UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Avaliação do Ensino Antecipado de Programação Paralela a Alunos sem Formação Prévia em Computação

Davi José Conte

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Davi José Conte

Avaliação do Ensino Antecipado de Programação Paralela a Alunos sem Formação Prévia em Computação

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Paulo Sérgio Lopes de Souza

USP – São Carlos
Agosto de 2020

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

C761a Conte, Davi José
 Avaliação do Ensino Antecipado de Programação
Paralela aAlunos sem Formação Prévia em Computação /
Davi José Conte; orientador Paulo Lopes Sérgio de
Souza. -- São Carlos, 2020.
 184 p.

 Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2020.

 1. Programação paralela. 2. Computação paralela.
3. Ensino. 4. Aprendizagem. I. Souza, Paulo Lopes
Sérgio de, orient. II. Título.

Davi José Conte

Evaluation of Early Parallel Programming Teaching to Students without Previous Knowledge in Computing

Dissertation submitted to the Institute of Mathematics
and Computer Sciences – ICMC-USP – in
accordance with the requirements of the Computer
and Mathematical Sciences Graduate Program, for
the degree of Master in Science. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Paulo Sérgio Lopes de Souza

USP – São Carlos
August 2020

Aos meus pais, noiva e a toda família e amigos que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa da minha vida.

AGRADECIMENTOS

Nesses anos de pesquisa, muito estudo, esforço e empenho, gostaria de agradecer primeiramente a Deus por ter me capacitado para chegar até aqui e também às pessoas que me acompanharam nessa jornada. Agradeço aos meus pais pelo constante apoio, que me sustentou nos momentos de dificuldade, à minha noiva, Paula Caires, por estar junto dando forças e incentivo em todos os momentos, aos amigos de laboratório que contribuíram com meu crescimento e desenvolvimento até aqui, em especial, ao Guilherme Martins e Ana Spengler, os quais sem a colaboração não poderia ter finalizado esse projeto. Agradeço imensamente ao meu orientador, Paulo Sérgio Lopes de Souza, o qual, com suas valiosas orientações, guiou-me com sabedoria e destreza no decorrer de todo o trabalho de mestrado. Agradeço pela oportunidade de participar do excelente programa de pós-graduação do ICMC e à CAPES pelo suporte financeiro.

“Se a educação sozinha não transforma a sociedade, sem ela, tampouco, a sociedade muda.”
(Paulo Freire)

RESUMO

CONTE, D. J. **Avaliação do Ensino Antecipado de Programação Paralela a Alunos sem Formação Prévia em Computação**. 2020. 184 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

A necessidade de soluções de *software* que fazem uso de computação de alto desempenho é cada vez maior na sociedade atual. Essas soluções podem ser obtidas com o uso de processadores *multi-core* e até mesmo supercomputadores. Desse modo, é necessário que profissionais da computação sejam capacitados para explorar os benefícios da programação paralela da melhor forma possível. Entretanto, devido à complexidade do conteúdo, este é abordado apenas nos períodos finais dos cursos de graduação, postergando o contato dos alunos com a computação paralela. Dado esse cenário, este projeto investiga a antecipação do ensino de programação paralela para alunos sem formação prévia em computação, como por exemplo alunos do primeiro ano de cursos de ensino superior de computação. Com isso, podemos verificar se é possível superar alguns dos principais desafios do ensino impostos nesse contexto. Neste trabalho são analisados os conceitos definidos como essenciais por currículos cancelados pela ACM e IEEE, que definem as grades curriculares da computação. Nesse sentido, este trabalho também apresenta um estudo inicial sobre os conteúdos a serem ministrados a alunos sem conhecimentos prévios em computação, utilizando-se diferentes abordagens de ensino, inclusive baseadas em metodologias ativas. Obtivemos resultados de aprendizagem positivos com os estudantes que não possuíam conhecimento prévio em computação nos experimentos. O aproveitamento destes foram estatisticamente similares ou superiores quando comparados com o aproveitamento de alunos que possuíam conhecimento prévio. Os resultados demonstraram que foi possível ensinar programação paralela desde o início da graduação para os alunos participantes dos experimentos, com qualidade, aferida pelas avaliações dos algoritmos feitos e pelas avaliações dos alunos sobre o conteúdo ensinado.

Palavras-chave: Programação paralela, computação paralela, ensino, aprendizagem .

ABSTRACT

CONTE, D. J. **Evaluation of Early Parallel Programming Teaching to Students without Previous Knowledge in Computing**. 2020. 184 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

The need for software solutions that make use of parallel computing is a reality nowadays. The use of platforms with multi/many-core processors and even clusters demand for parallel software and professionals with consistent skills. Thus, it is necessary to train future computing professionals as soon as possible to explore the benefits of parallel programming. However, due to the complexity of the content, it is only addressed in the final periods of the undergraduate courses, postponing the students' contact with parallel computing. Given this scenario, this project investigates the anticipation of teaching parallel programming for students without previous training in computing, such as students in the first year of higher education courses in computing. We hope to overcome some of the main challenges of teaching in this context. In this work, we analyze the concepts defined as essential by curricula certified by ACM and IEEE, which represent the curricular grids of computing. In this way, this project also presents an initial study on topics to be taught to students without previous knowledge in computing, using different teaching approaches, including based on active methodologies. We obtained positive learning results in the experiments with students who had no previous knowledge in computing. Their performance was statistically similar or superior when compared to the performance of students who had previous knowledge. The results showed that it was possible to teach parallel programming from the beginning of graduation to the students participating in the experiments, with quality, measured by the evaluations of the algorithms made and the evaluations of the students on the content taught.

Keywords: Parallel programming, parallel computing, teaching, learning .

LISTA DE ILUSTRAÇÕES

Figura 1 – Abordagem, design e procedimento (RICHARDS; RODGERS, 1982)	32
Figura 2 – A sala de aula do ensino tradicional, denominado por Freire (1987) como “ensino bancário” (SOUZA <i>et al.</i> , 2011)	33
Figura 3 – Etapas da metodologia TBL (adaptado de Inuwa (2012))	37
Figura 4 – Relação das tecnologias mais utilizadas no âmbito do ensino de programação paralela	42
Figura 5 – Quantidade de estudos publicados por ano	51
Figura 6 – Relação das <i>keywords</i> mais frequentes	51
Figura 7 – <i>Word cloud</i> de tecnologias/ferramentas mais aplicadas no ensino de paralelismo	52
Figura 8 – Estrutura dos principais tópicos da grade curricular (PRASAD <i>et al.</i> , 2012) .	65
Figura 9 – Tópicos referentes à arquitetura (PRASAD <i>et al.</i> , 2012)	66
Figura 10 – Tópicos referentes à programação (PRASAD <i>et al.</i> , 2012)	66
Figura 11 – Tópicos referentes a algoritmos (PRASAD <i>et al.</i> , 2012)	67
Figura 12 – Tópicos referentes a tópicos avançados (PRASAD <i>et al.</i> , 2012)	67
Figura 13 – Camadas de computação paralela	70
Figura 14 – Classificação dos tópicos referentes às diferenças de programação paralela e sequencial considerando o contexto de alunos do primeiro ano.	73
Figura 15 – Distribuição de aproveitamento na etapa de garantia de preparo individual .	80
Figura 16 – Histograma das pontuações obtidas na etapa de garantia de preparo em equipe	81
Figura 17 – Pontuações médias por atividade	81
Figura 18 – Qualidade de código média por turma	82
Figura 19 – Qualidade de código média por atividade prática	82
Figura 20 – Avaliação qualitativa - questão 1	83
Figura 21 – Avaliação qualitativa - questão 3	83
Figura 22 – Avaliação qualitativa - questão 5	84
Figura 23 – Avaliação qualitativa - questão 8	84
Figura 24 – Avaliação qualitativa - questão 10	85
Figura 25 – Avaliação qualitativa - questão 11	85
Figura 26 – Avaliação qualitativa - questão 12	86
Figura 27 – Alunos por curso e período - OpenMP/2018	92
Figura 28 – Classificação de alunos de acordo com formação prévia - OpenMP/2018 . .	92
Figura 29 – Comparação da distribuição de notas de cada métrica de avaliação - OpenMP/2018	94
Figura 30 – Comparação das porcentagens de aproveitamento do curso - OpenMP/2018	94

Figura 31 – Resultado avaliação qualitativa - OpenMP/2018	97
Figura 32 – Respostas da questão 2 da avaliação qualitativa - OpenMP/2018	99
Figura 33 – Respostas da questão 4 da avaliação qualitativa - OpenMP/2018	99
Figura 34 – Respostas da questão 6 da avaliação qualitativa - OpenMP/2018	101
Figura 35 – Respostas da questão 7 da avaliação qualitativa - OpenMP/2018	101
Figura 36 – Respostas da questão 5 da avaliação qualitativa - OpenMP/2018	102
Figura 37 – Respostas da questão 8 da avaliação qualitativa - OpenMP/2018	102
Figura 38 – Respostas da questão 9 da avaliação qualitativa - OpenMP/2018	103
Figura 39 – Alunos por curso e período - Pthreads/2018	107
Figura 40 – Classificação de alunos de acordo com formação prévia - Pthreads/2018	108
Figura 41 – Comparação da distribuição de notas de cada métrica de avaliação - Pthreads/2018	109
Figura 42 – Comparação das porcentagens de aproveitamento do curso - Pthreads/2018	111
Figura 43 – Resultado avaliação qualitativa - Pthreads/2018	113
Figura 44 – Respostas da questão 2 da avaliação qualitativa - PThreads/2018	116
Figura 45 – Respostas da questão 4 da avaliação qualitativa - PThreads/2018	116
Figura 46 – Respostas da questão 6 da avaliação qualitativa - PThreads/2018	118
Figura 47 – Respostas da questão 7 da avaliação qualitativa - PThreads/2018	118
Figura 48 – Respostas da questão 5 da avaliação qualitativa - PThreads/2018	119
Figura 49 – Respostas da questão 8 da avaliação qualitativa - PThreads/2018	119
Figura 50 – Respostas da questão 9 da avaliação qualitativa - PThreads/2018	120
Figura 51 – Alunos por curso - OpenMP/2019-1	127
Figura 52 – Período no curso de graduação dos alunos - OpenMP/2019-1	127
Figura 53 – Pré-teste x pós-teste (questões sobre programação sequencial)	128
Figura 54 – Pré-teste x pós-teste (questões sobre paralelismo)	128
Figura 55 – Evolução Pré-teste x pós-teste (questões sobre paralelismo)	129
Figura 56 – Distribuição das notas por aula	131
Figura 57 – Aproveitamento por métrica no curso	131
Figura 58 – Médias finais dos alunos no curso	132
Figura 59 – Resultado de aproveitamento do desafio extra - cálculo paralelo do Pi	132
Figura 60 – Distribuição de notas das categorias de motivação - OpenMP 2019-1	133
Figura 61 – Questão 30 do CIS	135
Figura 62 – Questão 31 do CIS	136
Figura 63 – Resultado geral da análise de motivação	137
Figura 64 – Caracterização dos participantes por curso	141
Figura 65 – Porcentagem de acertos, erros e "não sei" dos pré e pós-testes	142
Figura 66 – Comparação entre as médias do pré-teste e pós-teste	142
Figura 67 – Médias finais por equipe considerando os desafios de programação paralela	143
Figura 68 – Médias de qualidade de código por desafio	143

Figura 69 – Distribuição de notas das categorias de motivação - OpenMP 2019-2	144
---	-----

LISTA DE TABELAS

Tabela 1 – Sete boas práticas de ensino (CHICKERING; GAMSON, 1987)	34
Tabela 2 – Comparação de concepções do ensino tradicional e ensino ativo	35
Tabela 3 – Palavras-chave e sinônimos deste mapeamento	47
Tabela 4 – Resultado da pesquisa.	48
Tabela 5 – Estudos selecionados após a execução dos critérios de inclusão e exclusão.	49
Tabela 6 – Critérios de qualidade	49
Tabela 7 – Número de artigos por base de pesquisa	52
Tabela 8 – Quantidade de Publicações por tipo	52
Tabela 9 – Quantidade de Publicações por abordagem desenvolvida	53
Tabela 10 – Quantidade de publicações por método	54
Tabela 11 – Quantidade de publicações por recurso educacional	54
Tabela 12 – Principais trabalhos relacionados I	55
Tabela 13 – Principais trabalhos relacionados II	56
Tabela 14 – Configuração das turmas: Experimento I	78
Tabela 15 – Médias das métricas considerando a amostra de alunos sem conhecimento prévio OpenMP 2018	93
Tabela 16 – Médias das métricas considerando a amostra de alunos com conhecimento prévio OpenMP 2018	93
Tabela 17 – Teste de Wilcoxon para amostras independentes	95
Tabela 18 – Planejamento de curso	106
Tabela 19 – Médias das métricas considerando a amostra de alunos sem conhecimento prévio Pthreads 2018	110
Tabela 20 – Médias das métricas considerando a amostra de alunos com conhecimento prévio Pthreads 2018	110
Tabela 21 – Teste de Wilcoxon para amostras independentes	111
Tabela 22 – Atividades práticas OpenMP 2019-1	123
Tabela 23 – Conteúdo das aulas OpenMP 2019-1	126
Tabela 24 – Teste de Wilcoxon para amostras pareadas	130
Tabela 25 – Médias de aproveitamento de cada aula por métricas de avaliação	130
Tabela 26 – Planejamento de curso: sexto experimento.	139
Tabela 27 – Aproveitamento de aprendizagem geral dos alunos nos experimentos realizados.	146
Tabela 28 – Teste de Wilcoxon: média de todos os experimentos	147
Tabela 29 – Teste de Wilcoxon: média dos experimentos I, II e V	148

SUMÁRIO

1	INTRODUÇÃO	25
1.1	Considerações iniciais	25
1.2	Contexto	25
1.3	Problema	26
1.4	Hipótese	27
1.5	Motivação	27
1.6	Objetivos	28
1.7	Estrutura da dissertação	29
2	FUNDAMENTAÇÃO TEÓRICA	31
2.1	Metodologia, Abordagem e Técnica de Ensino	31
2.2	Metodologia de Ensino Tradicional	32
2.3	Metodologias de Aprendizagem Ativa	33
2.3.1	<i>Aprendizagem Baseada em equipes</i>	35
2.4	Ensino de Programação Paralela	37
2.4.1	<i>Conceitos Essenciais Ministrados</i>	38
2.4.2	<i>Metodologias e Tecnologias no Ensino de Programação Paralela</i> . .	41
3	REVISÃO DA LITERATURA	45
3.1	Considerações iniciais	45
3.2	Mapeamento Sistemático	45
3.2.1	<i>Objetivo do Mapeamento</i>	46
3.2.2	<i>Questões de Pesquisa</i>	46
3.2.3	<i>Bases de pesquisa e Palavras-chave</i>	47
3.2.4	<i>String de busca</i>	47
3.2.5	<i>Seleção</i>	48
3.2.6	<i>Metodologia de Avaliação</i>	48
3.2.7	<i>Avaliação de qualidade</i>	49
3.2.8	<i>Extração de Dados</i>	50
3.2.9	<i>Resultados do Mapeamento</i>	50
3.2.10	<i>Limitações</i>	53
3.2.11	<i>Conclusões</i>	53
3.3	Principais Trabalhos Relacionados	55

3.3.1	<i>Lacuna</i>	62
4	DESENVOLVIMENTO DO PROJETO	63
4.1	Objetivos	63
4.1.1	<i>Objetivos específicos</i>	64
4.2	Análise dos conceitos fundamentais de programação paralela	64
4.3	Mapas Conceituais	65
4.4	Estruturação em camadas	67
4.5	Definição do escopo de ensino	72
5	ESTUDOS EXPERIMENTAIS	75
5.1	Considerações iniciais	75
5.2	Relação de experimentos	75
5.3	Experimento I: Disciplinas de Programação Concorrente baseadas em Aprendizagem Baseada em Equipes (ABE)	77
5.3.1	<i>Definição do experimento</i>	77
5.3.2	<i>Planejamento</i>	78
5.3.3	<i>Execução</i>	79
5.3.4	<i>Apresentação dos resultados</i>	79
5.3.5	<i>Análise dos resultados</i>	86
5.4	Experimento II: Curso de Difusão Cultural de Introdução ao <i>OpenMP</i> 87	
5.4.1	<i>Definição do experimento</i>	87
5.4.2	<i>Planejamento</i>	88
5.4.3	<i>Execução</i>	89
5.4.4	<i>Apresentação dos resultados</i>	90
5.4.5	<i>Análise dos resultados</i>	98
5.5	Experimento III: Curso de Difusão Cultural de Introdução a <i>Pthreads</i> 103	
5.5.1	<i>Definição do experimento</i>	104
5.5.2	<i>Planejamento</i>	104
5.5.3	<i>Execução</i>	105
5.5.4	<i>Apresentação dos resultados</i>	107
5.5.5	<i>Análise dos resultados</i>	114
5.6	Experimento IV: Curso de Difusão Cultural de Introdução ao <i>OpenMP</i> para alunos iniciantes em computação.	120
5.6.1	<i>Definição do experimento</i>	121
5.6.2	<i>Planejamento</i>	122
5.6.3	<i>Execução</i>	125
5.6.4	<i>Apresentação dos resultados</i>	126
5.6.5	<i>Análise dos resultados</i>	133

5.7	Experimento V: Curso de Difusão Cultural de Introdução ao <i>OpenMP</i> baseado em <i>Problem Based Learning</i>	137
5.7.1	<i>Escopo</i>	137
5.7.2	<i>Planejamento</i>	138
5.7.3	<i>Execução</i>	140
5.7.4	<i>Apresentação dos resultados</i>	140
5.7.5	<i>Análise dos resultados</i>	144
5.8	Análise dos resultados dos experimentos	145
6	CONCLUSÕES	151
6.1	Considerações iniciais	151
6.2	Caracterização da Pesquisa Realizada e seus Resultados	151
6.3	Principais Contribuições	152
6.4	Limitações	153
6.5	Produção Científica	154
6.6	Trabalhos Futuros	155
	REFERÊNCIAS	157
APÊNDICE A	QUESTIONÁRIO QUALITATIVO - EXPERIMENTO 1	163
APÊNDICE B	QUESTÕES DO PRÉ-TESTE - CURSO DE OPENMP PARA INICIANTE EM COMPUTAÇÃO	167
APÊNDICE C	QUESTÕES DO PÓS-TESTE - CURSO DE OPENMP PARA INICIANTE EM COMPUTAÇÃO	173
APÊNDICE D	PRÉ-TESTE - CURSO DE INTRODUÇÃO AO OPENMP BASEADO EM PROBLEM BASED LEARNING . . .	177
APÊNDICE E	PÓS-TESTE - CURSO DE INTRODUÇÃO AO OPENMP BASEADO EM PROBLEM BASED LEARNING . . .	181

INTRODUÇÃO

1.1 Considerações iniciais

Este Capítulo contém a introdução a este trabalho, apresentando o contexto onde o projeto se insere, problema encontrado, hipótese, motivação em realizá-lo, objetivos e, por fim, mostra a estrutura da monografia. Os detalhes teóricos abordados pelo projeto e outras particularidades serão apresentados em capítulos posteriores.

1.2 Contexto

A necessidade de soluções de software que façam uso apropriado da computação paralela é uma realidade já há algum tempo na nossa sociedade. Hardware com *multi* e *many-cores*, e *clusters* com processadores heterogêneos (uso conjunto de processadores escalares - ou CPUs (*Central Processing Unit*) - e Unidades de Processamento Gráfico - ou GPUs (*Graphical Processing Unit*), são populares na academia e em atividades comerciais, financeiras e industriais (FERNÁNDEZ *et al.*, 2019). Devido a essa presença, o desempenho de soluções de *software* que exploram o paralelismo cresceu consideravelmente em diferentes aplicações como a mineração de dados, processamento de linguagem natural, documentos fiscais eletrônicos, acessos a grandes bancos de dados corporativos, engenharias, matemática, física, pesquisa operacional, e muitos outros (BOURGOIN; CHAILLOUX; LAMOTTE, 2014; FIORE; BAKHOUYA; SMARI, 2018).

A programação paralela permite o desenvolvimento de programas que, ao executarem na forma de processos concorrentes, consideram o uso de múltiplas tarefas independentes, ainda que em máquinas com um único elemento de processamento (processadores, núcleos, aceleradores gráficos ou outros), onde estas atividades disputam recursos computacionais, no intuito de solucionar um problema específico. Consideram-se processos concorrentes aqueles

cujas execuções começaram e em um instante de tempo ainda não terminaram (mesmo que em um único elemento de processamento), e paralelos aqueles processos concorrentes cujas execuções ocorrem, de fato, simultaneamente em elementos de processamento distintos de uma arquitetura paralela (ALMASI; GOTTLIEB, 1989). Esta definição estabelece, portanto, que processos paralelos são especificações dos processos concorrentes. Para efeito do ensino de programação paralela ou programação concorrente abordado neste trabalho, assumimos que ambos os estilos de programação possuem construções e características análogas. Portanto, os termos paralelo e concorrente serão usados como sinônimos ao longo deste texto.

Apesar da demanda por profissionais capacitados em usar a computação paralela (ERNST *et al.*, 2009), o ensino de programação paralela não acompanha a demanda, pois não é considerado algo trivial de ser realizado (BI; BEIDLER, 2007). As disciplinas de programação paralela são, geralmente, dispostas em semestres ou anos mais avançados das grades curriculares dos cursos de computação, muitas vezes como disciplinas eletivas (BACHIEGA *et al.*, 2017).

Isso faz com que os alunos desconheçam a programação paralela no início das suas formações e fiquem restritos ao desenvolvimento de soluções puramente sequenciais, em diferentes áreas da computação onde a computação paralela já poderia ser empregada (contribuindo ao aprendizado de se extrair melhores desempenhos a custos menores das soluções propostas). Enquanto o paralelismo das aplicações pode ser empregado usualmente na maioria dos computadores presentes no mundo atual, a ausência de conhecimento precoce dessa área da computação impede que os alunos exercitem-na enquanto estão sendo treinados para o atuarem no mercado de trabalho ou em centros de pesquisa. Portanto, este trabalho defende que é imperativo à formação de nossos futuros egressos em cursos de computação antecipar o ensino da programação paralela desde o início da formação profissional.

1.3 Problema

O ensino de programação paralela não é algo trivial (BI; BEIDLER, 2007). A principal dificuldade dos alunos é alterar a estratégia de programação focada em soluções sequenciais para a programação paralela, esta diferente, pois envolve a divisão de um problema em tarefas independentes, geração de diferentes processos e distribuição destes a processadores. Além disso, existem outros tópicos relacionados à distribuição de tarefas, estratégias de mapeamento e balanceamento da carga de trabalho que os estudantes necessitam compreender para realmente melhorar o desempenho de suas aplicações (MURESANO; REXACHS; LUQUE, 2010).

Devido a isso, o ensino da programação paralela para alunos sem formação prévia em computação torna-se mais complexo de ser implementado quando comparado com o ensino de programação sequencial. Um dos primeiros desafios a ser superado é como abordar este assunto a alunos que não possuam os pré-requisitos necessários de outras disciplinas básicas como organização e arquitetura de computadores, sistemas operacionais, redes de computadores e a

própria programação sequencial. A ACM e a IEEE [Prasad *et al.* \(2012\)](#) apresentam os conceitos fundamentais de cursos de graduação, mostrando as dependências de cada um com outras disciplinas relacionadas à computação, sustentando a ideia do desafio de ensinar programação paralela para alunos no início da graduação.

Nesse contexto, acredita-se que o ensino de programação paralela deva ser gradual e sistemático desde as primeiras aulas na graduação e, por isso, faz-se necessário analisar metodologias que possam ser aplicadas com este objetivo, incentivando os alunos a compreenderem programação paralela.

1.4 Hipótese

Investigamos neste trabalho a hipótese de que alunos sem formação básica em computação são capazes de compreender e aplicar conceitos de paralelismo, podendo, inclusive, desenvolver códigos com qualidade satisfatória em termos de quão corretos estão em relação aos fundamentos de programação paralela. Acredita-se que o ensino de programação paralela para estudantes neste contexto pode ter um excelente resultado, análogo ao observado por discentes em estágios mais avançados de cursos computação, estes com um conhecimento prévio e básico maior.

1.5 Motivação

A motivação do ensino de programação paralela para alunos ingressantes nos cursos superiores de computação se dá pela importância de se consolidar tal conhecimento atualmente como ferramenta usual de desenvolvimento de soluções computacionais. Acadêmicos e profissionais do mercado acreditam que o paralelismo é um dos principais conteúdos de ensino da grade curricular de cursos de computação ([GHAFOOR; BROWN; ROGERS, 2017](#); [JOHN, 1994](#); [KO; BURGSTALLER; SCHOLZ, 2013](#)). Instituições como ACM e IEEE defendem o fortalecimento do ensino da programação paralela nos cursos de graduação, mesmo com a dependência de outros conceitos ([CURRICULA, 2013](#); [PRASAD *et al.*, 2012](#)). Acredita-se que a introdução da programação paralela antecipada possibilite aos alunos relacionarem o paralelismo em outras disciplinas como por exemplo: estrutura de dados, projeto de *hardware* e computação gráfica. Além disso, soluções paralelas desenvolvidas por profissionais com tal conhecimento podem apresentar maior desempenho, melhores relações custo x benefício e menores custos de manutenção, pois serão projetadas com a visão de melhorar a qualidade com otimização de métricas de desempenho como o tempo de resposta.

Segundo [McMaster, Rague e Anderson \(2010\)](#), alunos no início da graduação conhecem os paradigmas de programação apenas como abstrações, onde há uma ausência de padrões de desenvolvimento, sendo mais propensos à aprendizagem do "pensamento paralelo" do que alunos

em nível mais avançado. Isto aplica-se também a discentes de cursos de outras áreas além da computação, visto que, na maioria das vezes, estes não possuem conhecimentos consolidados em programação de computadores.

Entender o processamento paralelo está se tornando cada vez mais importante para graduados em computação e também em outras áreas. Aprender esses conceitos paralelos desde o início pode dar aos alunos habilidades importantes para o início da sua vida profissional. Por outro lado, a programação paralela continua a ser um assunto mais complexo do que a sequencial na maioria dos ambientes de programação contemporâneos. À medida que as abordagens pedagógicas mais bem-sucedidas são desenvolvidas para tornar o processamento paralelo acessível aos programadores mais novos e os paradigmas para a programação paralela se tornam mais comuns, espera-se que mais elementos de programação paralela apareçam no início dos cursos (CURRICULA, 2013).

A literatura disponível usualmente relata estudos de caso sobre o uso de metodologias ou ferramentas aplicadas ao ensino de programação paralela. Não foram encontrados estudos que definam quais tópicos poderiam ou deveriam ser apropriados para o ensino de programação paralela a alunos no início da graduação, com uma avaliação abrangente e sistemática do aprendizado desses alunos (GHAFOOR; BROWN; ROGERS, 2017).

As metodologias ativas podem colaborar neste processo de ensino-aprendizagem. Elas se diferem da tradicional por considerarem o aluno como a gente ativo na aula, envolvendo-o mais no processo de ensino e aprendizagem. O aluno busca e constrói seu conhecimento, vendo o professor como um facilitador e orientador do ensino, que estimula o aluno a pensar no que está aprendendo (BONWELL; EISON, 1991).

Por isso, este projeto considera o ensino de programação paralela usando diferentes metodologias de ensino, no caso, a metodologia tradicional, a Aprendizagem Baseada em Equipes (ABE ou TBL - *Team Based Learning*) e o aprendizado baseado em problemas (*Problem Based Learning*). Busca-se neste trabalho avaliar o uso de diferentes estratégias educacionais para superar o desafio de ensino de um conteúdo complexo para alunos que não possuem conhecimento de conceitos abordados em disciplinas introdutórias como sistemas digitais, organização e arquitetura de computadores, sistemas operacionais e redes. O uso de diferentes metodologias permite verificar a interação entre alunos e também com o professor, para que o conhecimento de programação paralela seja construído naturalmente.

1.6 Objetivos

O principal objetivo deste trabalho é investigar a antecipação do processo de ensino-aprendizagem de programação paralela por alunos que não possuam formação prévia em computação, abrangendo discentes ingressantes em cursos superiores de computação, ou seja, no primeiro ano de graduação, e de outras áreas de conhecimento fora da computação. Além disso,

espera-se determinar a abrangência do conteúdo de programação paralela que seria ministrado para alunos neste contexto, de forma que estes superem obstáculos da ausência de conhecimento prévio e causem boa receptividade ao assunto ministrado.

1.7 Estrutura da dissertação

O presente documento está dividido em seis capítulos, os quais estão dispostos da seguinte forma:

O Capítulo 1 compreende o texto introdutório, dividido pelas seções de Contexto, Problema, Hipótese, Motivação, Objetivos e Estrutura da Monografia. Neste capítulo o projeto é apresentado de forma geral e sucinta para que o leitor possa compreender o conteúdo do trabalho.

O Capítulo 2 contempla a fundamentação teórica do trabalho, onde são apresentados os conceitos fundamentais para o projeto e o embasamento da proposta do mesmo. Neste capítulo estão incluídas seções sobre metodologia tradicional de ensino, metodologias de aprendizagem ativa e o ensino de programação paralela, discutindo sobre os principais conceitos ensinados e as tecnologias utilizadas.

O Capítulo 3 apresenta o mapeamento sistemático da literatura realizado para este projeto, demonstrando o estado da arte no ensino de programação paralela e os principais trabalhos relacionados.

O Capítulo 4 refere-se ao desenvolvimento do projeto e demonstra uma análise de iniciativas curriculares voltadas para o ensino de programação paralela. Além disso, nesse capítulo é definido o escopo de ensino para alunos sem formação prévia em computação.

O Capítulo 5 descreve os experimentos que foram realizados, apresentando as etapas do processo de planejamento e execução, além dos resultados de aprendizagem obtidos.

Por fim, no Capítulo 6, analisam-se os resultados desse projeto de forma geral, apontando para as principais contribuições, limitações, produções e propondo trabalhos futuros.

FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada a fundamentação teórica do trabalho definindo os princípios da área que o projeto explora e visa atuar. Inicialmente são apresentadas discussões sobre a metodologia de ensino tradicional e metodologias de aprendizagem ativa, definindo as principais ideias e concepções. Em seguida é discutido sobre o ensino atual de programação paralela, tratando dos conceitos essenciais que devem ser ministrados na disciplina e as metodologias mais utilizadas, de acordo com o que foi encontrado na literatura.

Com os assuntos explorados neste capítulo é possível notar a dificuldade do ensino de programação paralela relativa à necessidade do conhecimento de conceitos provenientes de outras áreas da computação. Isso dificulta a introdução de tópicos de programação paralela para alunos que não possuam formação básica em computação.

2.1 Metodologia, Abordagem e Técnica de Ensino

Há definições de diferentes autores para os termos metodologia, abordagem e técnica de ensino. O presente projeto será orientado pela taxonomia de [Richards e Rodgers \(1982\)](#), em que é feita uma reformulação dos termos de [ANTHONY \(1963\)](#). Embora trata-se de uma referência já antiga, as definições são satisfatórias para os objetivos do presente projeto. A reformulação da taxonomia envolve os termos metodologia, abordagem, design e procedimento, definidos da seguinte forma:

- **Metodologia:** termo que engloba a especificação e inter-relação entre teoria e prática.
- **Abordagem:** engloba as teorias dos conceitos e do aprendizado dos mesmos.
- **Design:** inclui o conteúdo da instrução (ou seja, o programa), os papéis do aluno e do professor no sistema e tipos de materiais de instrução.

- **Procedimento:** o foco é sobre as técnicas no momento de atuação no ensino e na aprendizagem de acordo com um método particular.

A Figura 1 ilustra a hierarquia reformulada pelos autores.

Figura 1 – Abordagem, design e procedimento (RICHARDS; RODGERS, 1982)

2.2 Metodologia de Ensino Tradicional

O ensino de forma tradicional é centrado no professor. Mizukami (1986) diz que “esse tipo de ensino é externo ao aluno”, isto é, a aprendizagem ocorre através do programa, disciplinas e professor. Em outras palavras, o aluno aprende apenas o que é transmitido para ele pelo professor, dirigido por um programa de ensino.

Diversos autores como Saviani (1986) sugerem que o professor tem a função de garantir que o conhecimento seja adquirido, independentemente do interesse e da vontade do aluno. O indivíduo, de início, é considerado uma *tábula rasa*, onde são impressas imagens e informações ao longo de sua vida (MIZUKAMI, 1986).

O autor Freire (1987) denomina, ainda, o ensino tradicional como um ensino baseado na “educação bancária”, ou seja, a educação caracterizada por “depositar” no aluno conhecimentos, dados, informações, fatos, ideias, etc. A Figura 2 ilustra a concepção de ensino tradicional que o aluno apenas recebe o conhecimento transmitido pelo professor. Ainda segundo o autor, “o educador é o que pensa e os educandos os pensados; o educador é o que diz a palavra e os

educandos, os que escutam; o educador é o que opta e prescreve sua opção e os educandos os que seguem a prescrição;”, concluindo que o educador é o sujeito ativo do processo.

Figura 2 – A sala de aula do ensino tradicional, denominado por Freire (1987) como “ensino bancário” (SOUZA *et al.*, 2011)

O trabalho intelectual do aluno usualmente é iniciado após a aula, quando o professor, provavelmente, sugere exercícios para treinamento e memorização. Uma das dificuldades desse método é o atendimento individual por parte do professor, pois o resto da classe deverá esperar o atendimento de um aluno finalizar. Tende-se a tratar todos os alunos igualmente, seguindo o mesmo ritmo de trabalho para todos.

A avaliação nesse sistema de ensino caracteriza-se por ser pontual, realizada por meio de provas e exames que medem a reprodução do conteúdo ensinado em sala de aula. Com isso, define-se inteligência como a capacidade de armazenar informações, englobando conhecimentos sobre mundo físico e social, considerando desde os mais simples até os mais complexos (MI-ZUKAMI, 1986). Este ensino reflete a sociedade atual na questão da valorização da variedade de conceitos, noções e informações que um indivíduo conhece ao invés do seu pensamento reflexivo.

2.3 Metodologias de Aprendizagem Ativa

A metodologia tradicional tem sido amplamente utilizada para o ensino em diferentes áreas de estudo há muito tempo, porém tem sido criticada por diversos educadores por levar o aluno a ter uma atitude passiva durante as aulas. Muitos professores afirmam que toda aprendizagem é ativa, pois, mesmo que apenas ouvindo um monólogo em sala de aula, os estudantes estão ativamente envolvidos. Entretanto, uma análise da literatura mostra que, para estarem mais ativamente envolvidos na aula, os alunos devem estar engajados em atividades como leitura, escrita, discussão em pares ou grupos, resolução de problemas, elaboração de análises ou resumos e até mesmo avaliações (CHICKERING; GAMSON, 1987).

Os autores Chickering e Gamson (1987) definem sete boas práticas consideradas para o ensino em nível de graduação (Tabela 1). Estas práticas foram definidas como um guia para

professores e alunos melhorarem o processo de ensino-aprendizagem, uma vez que os mesmos estão embasados em uma pesquisa de 50 anos sobre a maneira que os professores ensinam nas universidades, como os alunos aprendem e se comunicam e de que forma ocorre a comunicação aluno-professor.

Tabela 1 – Sete boas práticas de ensino ([CHICKERING; GAMSON, 1987](#))

1	Encorajar o contato entre alunos e professores
2	Desenvolver reciprocidade e cooperação entre os alunos
3	Encorajar a aprendizagem ativa
4	Fornecer <i>feedback</i> imediato de acertos ou erros
5	Enfatizar o gerenciamento do tempo necessário para a tarefa
6	Demonstrar altas expectativas dos alunos como motivação
7	Respeitar a diversidade de talentos e formas de aprendizado

Considerando as práticas expostas na Tabela 1, fazem-se necessárias estratégias de ensino que promovam a aprendizagem ativa por meio de atividades que permitam a construção do conhecimento, possibilitando que o estudante participe da mesma. Dessa forma, a aprendizagem ativa busca cumprir esses objetivos.

William Reginald Revans é considerado o pioneiro da aprendizagem ativa, que em 1955 na Universidade de *Manchester* foi o primeiro professor a espalhar suas ideias inovadoras de aprendizagem ativa e filosofia de "aprender fazendo". Entretanto, somente em meados da década de 60 suas ideias alcançaram reconhecimento internacional. A partir de então, aprofundaram-se os estudos de aprendizagem ativa incluindo o surgimento de novas técnicas e metodologias em livros publicados pelo próprio Revans ([BARKER, 2010](#)).

Na literatura, não é encontrada uma definição precisa e concisa sobre o termo "aprendizagem ativa", sendo, na maioria das vezes, apenas evidenciadas suas propriedades. Por isso, os autores ([BONWELL; EISON, 1991](#)) listam algumas características gerais comumente associadas a estratégias que instigam o aprendizado ativo em sala de aula, descritas abaixo.

- Os alunos estão mais envolvidos e deixam de ser apenas ouvintes da aula.
- Menos foco na transmissão do conhecimento e mais no desenvolvimento das habilidades dos alunos.
- Os alunos estão envolvidos em um pensamento de ordem superior (análises, sínteses, avaliações).
- Os alunos devem estar engajados nas atividades (leitura, discussão, escrita).
- Maior ênfase é dada para o aluno explorar seus conhecimentos, atitudes e valores.

Em suma, analisando as características apresentadas, pode-se concluir que a aprendizagem ativa busca envolver o aluno na realização de tarefas e pensar sobre o que estão fazendo (BONWELL; EISON, 1991).

A Tabela 2 demonstra as principais diferenças entre a metodologia tradicional e as metodologias ativas, sendo utilizada para efeito de comparação.

Tabela 2 – Comparação de concepções do ensino tradicional e ensino ativo

Metodologia Tradicional	Metodologia Ativa
Centrada no professor.	Centrada no aluno.
Professores capacitados em conteúdo.	Professores capacitados em educação integral.
Visa o trabalho isolado.	Visa o trabalho em equipe.
Orientada por conteúdos.	Orientada por competências.
Transmissão de informações e memorização.	Construção do conhecimento, desenvolvimento de habilidades e atitudes.
Ensino-aprendizagem com observação passiva dos estudantes.	Aprender a aprender com participação ativa dos estudantes.
Avaliação somativa ao final das unidades.	Avaliação formativa com retornos constantes ao longo das unidades.

Com o desenvolvimento e estudo da aprendizagem ativa, foram criadas novas metodologias de ensino que enfatizam este tipo aprendizagem. Essas metodologias, chamadas de metodologias ativas, fazem uso de diferentes técnicas para alcançar os objetivos de ensino de forma alternativa à tradicional. Algumas das metodologias utilizadas para o ensino de programação paralela são descritas no Capítulo 2.4.

Uma das metodologias que integra os conceitos da aprendizagem ativa e colaborativa é a aprendizagem baseada em equipes, que destaca o aprendizado com discussões e realização de atividades em grupo visando um objetivo compartilhado. Isso faz com que a metodologia apresente fortes características de avaliação formativa. Destaca-se que no levantamento bibliográfico realizado neste projeto não foram encontradas iniciativas da aplicação desta abordagem para o ensino de computação paralela. Portanto, iremos analisar os impactos causados no ensino de programação paralela pela aplicação dessa metodologia em sala de aula no primeiro experimento deste projeto, detalhado posteriormente na Seção 5.3. Dessa forma, a seção a seguir apresenta um detalhamento maior sobre a referida metodologia.

2.3.1 Aprendizagem Baseada em equipes

A metodologia da aprendizagem baseada em equipes, cujo termo equivalente em inglês se dá por *Team Based Learning*, comumente referenciada pela sigla (TBL), tem como fundamentação teórica a abordagem do construtivismo, na qual o professor se torna um facilitador de aprendizagem. Além dos alunos construírem seu conhecimento e "aprenderem a aprender", estes

são levados a praticar a comunicação e a argumentação com as atividades propostas para pequenos grupos de aprendizagem, compostos de cinco a sete alunos escolhidos de forma totalmente aleatória. As experiências e os conhecimentos prévios dos alunos devem ser evocados na busca da aprendizagem significativa (BOLLELA *et al.*, 2014).

A metodologia descrita a seguir é dividida em três etapas: preparação individual, avaliação de garantia de preparo e aplicação de conceitos.

1. Preparação individual

A preparação individual é uma atividade que deve ocorrer anteriormente à aula. Nesta etapa, os estudantes devem ser responsáveis por se prepararem individualmente para as atividades que serão realizadas em sala de aula por meio de textos, vídeos, imagens ou outros recursos definidos pelo professor com antecedência (BOLLELA *et al.*, 2014). Dessa forma, os alunos poderão compreender o assunto antes da aula e o professor terá função apenas de auxiliar e responder possíveis dúvidas durante a aula. Essa fase prepara o aluno para poder compartilhar suas ideias sobre o assunto estudado com um grupo de outros alunos.

2. Avaliação de garantia de preparo

A segunda etapa é a Garantia de Preparo, mecanismo básico que garante a responsabilidade individual pela preparação pré-classe. A garantia de preparo é feita em duas fases: garantia de preparo individual e em grupo. Na fase individual, um teste, contendo de 10 a 15 questões sobre o material enviado previamente para estudo pelo professor, deve ser respondido sem consulta a qualquer material bibliográfico ou didático. As questões devem ser referentes aos principais tópicos ou conceitos que devem ser alcançados pelo estudo prévio realizado. Após o término do teste individual, deve ser realizado o teste em grupo, dividindo os alunos em grupos de cinco a sete integrantes aleatoriamente. Os grupos deverão se reunir para resolver o mesmo conjunto de testes, também sem consulta (garantia de preparo em grupo). Os alunos devem discutir os testes e cada membro defende e argumenta as razões para sua escolha até o grupo decidir qual é a melhor resposta. Como resultado, os alunos percebem que são explicitamente responsáveis perante seus pares, não só no preparo pré-classe, mas também por ter que explicar e fundamentar suas respostas, exercitando suas habilidades de comunicação, argumentação e convencimento (BOLLELA *et al.*, 2014). Nessa fase, os grupos recebem *feedback* imediato se a resposta definida pelo grupo está correta ou não e, se não estiver correta, permite que tentem novamente.

Ao final dessa etapa, o professor responde o questionário conferindo as respostas com os alunos e dando abertura para questionamentos e dúvidas dos grupos. Neste ponto, as dúvidas mais pontuais que os alunos possam ter podem ser sanadas.

3. Aplicação de conceitos

O professor deve proporcionar aos estudantes, reunidos em suas equipes, a oportunidade de aplicar conhecimentos para resolver questões apresentadas na forma de cenários/problemas relevantes e presentes na prática profissional diária. Os estudantes devem ser desafiados a realizarem interpretações, inferências, análises ou sínteses (BOLLELA *et al.*, 2014). Esta é a etapa em que deve ser proporcionada a maior parte do tempo para que os alunos possam aplicar os conceitos aprendidos. A aplicação depende dos tópicos de cada área tratados em aula e de como o professor pretende que estes sejam desenvolvidos, podendo ser um problema real que mostre a prática do que foi aprendido, alguma tarefa escrita, modelagem de problema ou, no caso da computação, desenvolvimento de um *software*.

A Figura 3, a seguir, demonstra as etapas do TBL descritas previamente, desde o estudo individual, até a garantia de preparo e aplicação de conceitos. A Figura 3 também mostra que, para as fases dois e três, em sala de aula, são necessários por volta de 40 a 60 minutos.

Figura 3 – Etapas da metodologia TBL (adaptado de Inuwa (2012))

2.4 Ensino de Programação Paralela

Tradicionalmente, cursos de computação assumiram a disponibilidade de um único processador, um único processo e um único segmento, com a execução do programa sendo completamente conduzida pelas instruções do programador direcionando à execução sequencial. Os recentes desenvolvimentos de hardware e software levaram os educadores a repensar esses pressupostos, mesmo em níveis mais introdutórios, pois os processadores *multicore* são agora uma realidade presente na maior parte das máquinas, além do surgimento de novos paradigmas, como *big data*, que requerem processamento paralelo e armazenamento distribuído. Esse desenvolvimento fez ser necessário que o ensino específico de programação paralela fosse abordado como uma nova área de estudo em cursos de computação, direcionando os alunos a pensarem em execuções simultâneas e em paralelo e no alto desempenho que pode ser obtido.

Neste capítulo, inicialmente, serão vistos os conceitos considerados essenciais a serem ensinados em um curso de programação paralela de acordo com currículos aprovados e publicados na literatura. Em seguida, são demonstradas as metodologias de ensino mais utilizadas no ensino de programação paralela, assim como as tecnologias mais utilizadas.

2.4.1 *Conceitos Essenciais Ministrados*

A programação paralela nem sempre foi considerada como disciplina essencial em uma grade curricular de cursos de computação, sendo considerada uma área pequena dentro do curso. Até o final da década de 2000, buscava-se integrar os conteúdos de paralelismo em outras disciplinas como tópicos optativos ou complementares, não aprofundando o ensino do conteúdo aos alunos (ERNST; STEVENSON, 2008). A partir do crescimento da área, foi publicado, em 2013, um currículo para o curso de Ciência da Computação (CURRICULA, 2013), proposto por um grupo de especialistas e chancelado pela ACM juntamente com a IEEE, o qual considera Programação Paralela e Distribuída uma área de conhecimento essencial, sugerindo tópicos fundamentais e optativos que, segundo os autores, espera-se que sejam estudados por um graduando em nível de bacharel em um curso de ciência da computação. Nas diretrizes curriculares da ACM dos anos de 2001, 2005 e 2008 havia pouca menção a este ensino (The Joint Task Force on Computing Curricula, CORPORATE, 2001; SHACKELFORD *et al.*, 2006; CASSEL *et al.*, 2008). Já na versão de 2013 (CURRICULA, 2013), a preocupação com o ensino do paralelismo mostra-se mais evidente. A área se estruturou em torno de assuntos que podem ser abordados em diferentes disciplinas ao longo da trajetória dos alunos na graduação.

Para este trabalho, tomou-se como base os tópicos definidos como necessários no ensino de programação paralela e distribuída de acordo com o currículo em Curricula (2013). Para a definição de tais tópicos, houve grande contribuição de outro documento relacionado à programação paralela apresentado em Prasad *et al.* (2012). Desta forma, neste texto, o detalhamento de Prasad *et al.* (2012) será utilizado para complementar o currículo apresentado em Curricula (2013).

Para o desenvolvimento do referido currículo, inicialmente, foi definido um Comitê de Direção pela ACM e pela Sociedade de Computação da IEEE, que deu início ao trabalho de revisar as áreas de ensino determinadas no currículo anterior, devido a necessidade de criação de novas áreas de conhecimento deficientes no currículo de 2008.

Após a definição de um conjunto de áreas para o novo "corpo de conhecimento", que são as áreas da computação consideradas no currículo, um subcomitê foi elencado para revisar ou criar conceitos de cada área de conhecimento específico. Além disso, obteve-se *feedback* da sociedade acadêmica, incluindo professores, pesquisadores, alunos de pós-graduação e estudantes universitários, sobre a aplicação dos currículos anteriores e os conceitos definidos para cada área de ensino por meio de apresentações em conferências, revisões diretas e pesquisa *online*. A sociedade científica da área da computação contribuiu significativamente para a definição das

áreas de conhecimento que deveriam ser consideradas. Cada subcomitê trabalhou para resolver conflitos, eliminar redundâncias e categorizar corretamente cada tópico abordado no currículo de 2013 ([CURRICULA, 2013](#)).

O “corpo de conhecimento” pertencente ao currículo de 2013 está organizado em uma série de 18 “áreas de conhecimento”, correspondentes a áreas de estudo da computação. As áreas de conhecimento compreendidas pelo currículo são as seguintes, dispostas na ordem apresentada pelo documento: Algoritmos e Complexidade (AL), Arquitetura e Organização (AR), Ciência Computacional (CN), Estruturas Discretas (DS), Gráficos e Visualização (GV), Interação Homem-Computador (HCI), Garantia de Informação e Segurança (IAS), Gestão da Informação (IM), Sistemas Inteligentes (IS), Rede e Comunicações (NC), Sistemas Operacionais (OS), Desenvolvimento Baseado em Plataforma (PBD), Computação Paralela e Distribuída (PD), Linguagens de Programação (PL), Fundamentos de Desenvolvimento de Software (SDF), Engenharia de Software (SE), Fundamentos de Sistemas (SF), Questões Sociais e Prática Profissional (SP).

Grande parte das áreas de conhecimento são derivadas dos currículos de ciência da computação desenvolvidos em 2001 ([The Joint Task Force on Computing Curricula, CORPORATE, 2001](#)) e 2008 ([CASSEL et al., 2008](#)), porém submetidos à revisão para o currículo de 2013. Além das áreas submetidas à revisão, novas áreas foram incluídas no currículo de 2013, destacando o crescimento da importância de outras áreas desde os currículos anteriores e, atualmente, são integradas aos estudos em computação.

Versões anteriores do currículo possuíam tópicos de paralelismo espalhados em diferentes áreas de conhecimento. Dada a crescente importância da computação paralela e distribuída, foi crucial, para o currículo apresentado, abordar conceitos essenciais da área e promover a devida importância a esses tópicos ([CURRICULA, 2013](#)). Por isso, o currículo de 2013 dedica uma área de conhecimento específica para a programação paralela.

Os tópicos no currículo são classificados em “principais” (*Core*) ou “eletivos” *Elective*, com a divisão dos principais em duas camadas: “Camada Principal 1” (*Tier-1*) e “Camada Principal 2” (*Tier-2*). De acordo com ([CURRICULA, 2013](#)), uma grade curricular deve incluir:

- Todos os tópicos principais da camada 1 e garantir que todos os alunos estudaram os materiais.
- Todos ou quase todos tópicos principais da camada 2 e garantir que todos os estudantes compreendam a maior parte deste material.
- Material optativo relevante, pois cobrir apenas conceitos principais é insuficiente para uma grade completa.

Além disso, ao final da apresentação dos tópicos de cada subárea de ensino no documento,

são descritos os resultados de aprendizagem desejáveis resultantes de cada subárea que serão considerados como objetivo de ensino principal de cada subárea.

Conforme dito anteriormente, houve diversas contribuições da comunidade científica para a elaboração do referido currículo. Na área de Programação Paralela e Distribuída, destaca-se a contribuição do estudo realizado em [Prasad et al. \(2012\)](#), referenciado no próprio currículo de 2013 como trabalho provedor de valioso *feedback*. Durante o desenvolvimento do [Curricula \(2013\)](#), houve grande interação com a Força Tarefa responsável pela elaboração do [Curricula \(2013\)](#), provendo contribuições significantes na área de Programação Paralela e Distribuída no currículo.

Este trabalho, produzido por especialistas na área e chancelado pela IEEE, apresenta uma série de conceitos sobre programação paralela com determinados níveis de conhecimento que, segundo os autores, um aluno de Ciência ou Engenharia da Computação deve alcançar como resultado do curso. Como alguns desses conceitos são resultados de ensino de outras disciplinas, a proposta do currículo citado é que os conceitos sejam abordados de forma que possibilite a oportunidade do aluno "pensar em paralelo". Como um exemplo, podemos citar o ensino de estruturas de repetição que sejam paralelas ou operações independentes em algoritmos de busca que permitam o paralelismo, mesmo em disciplinas tradicionais do curso. A maior parte dos tópicos descritos no [Curricula \(2013\)](#) é abordada com maior detalhamento e profundidade no currículo em [Prasad et al. \(2012\)](#), além de outros conceitos não explorados em [Curricula \(2013\)](#).

Tais conceitos são classificados em quatro áreas, sendo estas: arquitetura, programação, algoritmos e interesses transversais. O conteúdo de cada área referida é composto pela descrição dos conceitos pertinentes à área e o grau desejável de conhecimento pelo aluno em determinado conceito, graus estes definidos de acordo com a taxonomia de compreensão de Bloom ([BLOOM, 1956](#)). Ainda, são descritos os resultados de aprendizado esperado, a disciplina a qual o conteúdo é oferecido e sugestão de número horas utilizado para cada conceito, além de apêndices sobre como ensinar cada tópico.

Por fim, neste projeto são considerados os conceitos de programação paralela definidos em [Curricula \(2013\)](#) como essenciais em cursos de computação e a iniciativa curricular em [Prasad et al. \(2012\)](#) como uma base de aprofundamento, mostrando mais detalhes e requisitos do ensino dos tópicos fundamentais. Além disso, é possível perceber a dificuldade na antecipação do ensino dessa disciplina por considerar diversos tópicos que possuem dependência direta com outras áreas como Organização e Arquitetura de Computadores, Sistemas Operacionais, Estrutura de Dados etc.

2.4.2 Metodologias e Tecnologias no Ensino de Programação Paralela

Diferentes metodologias têm sido utilizadas para o ensino de programação paralela em diferentes contextos universitários, conforme demonstrado no capítulo Trabalhos Relacionados (3). Os estudos mais relevantes levantados pelo mapeamento sistemático realizado incluem, para o ensino de programação paralela, abordagens construtivistas, aprendizagem ativa, aprendizagem colaborativa, avaliação contínua, demonstrações, experiências, entre outros. Além disso, foram somados os resultados do mapeamento sistemático realizado por [Bachiega et al. \(2017\)](#), por englobar trabalhos relacionados ao ensino teórico e prático de programação paralela.

Abaixo, destacam-se, com uma breve descrição, as principais metodologias encontradas nos trabalhos relacionados deste projeto juntamente com a revisão sistemática em [Bachiega et al. \(2017\)](#):

- **Sala de aula invertida:** trata-se de um modelo de aula que inverte o modelo tradicional. Atividades como leitura de materiais de instrução, estudo por meio de vídeos instrucionais e exercícios são realizados pelos alunos anteriormente à aula, dedicando o tempo em sala de aula para discussões e aplicação do tema estudado ([MOORE; DUNLOP, 2016](#)).
- **Feedback:** visa guiar e encorajar os alunos a refletirem sobre seus pensamentos e respostas. Assim, possibilita a mudança de pensamento, do modo de agir e do processo de ensino-aprendizagem. O *feedback* pode contribuir para a motivação do aluno por meio de respostas positivas e sugestões de aprendizagem ([GONÇALVES, 2017](#)).
- **Aprendizagem baseada em projeto:** a aprendizagem baseada em projetos (*Project-Based Learning*) refere-se a uma metodologia centrada no aluno na qual são apresentados problemas e desafios reais diretamente ao aluno, tornando o processo de ensino mais dinâmico e propiciando conhecimento mais realístico para os alunos ([ZARESTKY; BANGERTH, 2014](#)).
- **Programação por Padrões:** a programação por padrões (*pattern programming*) utiliza padrões que provêm um conjunto de regras detalhado para a resolução de um problema. O número de iterações em uma estrutura de repetição e um pseudocódigo podem ser tidos como exemplo. Entretanto, determinar padrões que sejam efetivos implica na necessidade de constante remodelagem dos mesmos ([CLANCY; LINN, 1999](#)).
- **Abordagem Analítica:** trata-se de uma abordagem fundamentada no construtivismo, na qual os alunos devem compreender os conceitos fundamentais e, então, aplicar o conhecimento adquirido na resolução de problemas mais complexos ([UL-ABDIN; SVENSSON, 2015](#)).

- **Aprendizagem Baseada em Problemas:** possui características semelhantes à Aprendizagem Baseada em Projeto, utilizando problemas específicos como um desafio para os estudantes, organizados de forma incremental em complexidade e relação dos assuntos abordados. Os alunos devem identificar outros problemas relacionados da área e solucionam de acordo com os critérios propostos pelo professor (CUENCA; GIMÉNEZ, 2016).
- **Aprendizagem Centrada em Modelo:** a Aprendizagem Centrada em Modelo (*Model-Center Learning*) sugere que o aluno construa conhecimento, realize experimentos com o ambiente, observe resultados e infira conclusões. Este modelo define que o propósito da instrução do professor é auxiliar os alunos na construção do conhecimento sobre objetos e eventos no ambiente vivenciado (GIBBONS, 2001).
- **Aprendizagem Colaborativa:** valoriza o conhecimento coletivo, gerado a partir de diferentes conhecimentos individuais. Sua definição mais comum refere-se a duas ou mais pessoas que aprendem algo em conjunto, buscando compreensão mútua, soluções, significados ou conquista de um objetivo compartilhado pelo grupo do qual fazem parte (DILLENBOURG, 1999). Isso implica em atividades que sejam compartilhadas por um grupo ao invés de tarefas individuais isoladas, de forma que os indivíduos possam interagir com ideias diferentes de outros integrantes, produzindo um resultado que somente pode ser alcançado com a contribuição e junção das ideias de todos os envolvidos (STAHL; KOSCHMANN; SUTHERS, 2006).

Além das metodologias citadas, outro aspecto que julga-se relevante ao ensino de programação paralela, trata-se da frequência da utilização de tecnologias, estimando a relação entre as linguagens de programação, APIs, *frameworks* e o processo de ensino das disciplinas de programação paralela e concorrente. Foi realizado um agrupamento das tecnologias mais utilizadas de acordo com os trabalhos levantados na revisão sistemática realizada, relatada no Capítulo 3, que buscou agrupar trabalhos sobre o ensino de programação paralela de forma ampla. Desta forma, a Figura 4 demonstra o agrupamento considerado por meio de um *word cloud*, mostrando, pelo destaque gráfico da palavra, as tecnologias utilizadas com maior frequência.

Figura 4 – Relação das tecnologias mais utilizadas no âmbito do ensino de programação paralela

Analisando a Figura 4 é possível notar que a ferramenta OpenMP têm sido a mais aplicada no contexto em questão, seguida pelo MPI, da plataforma CUDA e do OpenCL. As linguagens de programação mais utilizadas para o mesmo fim tratam-se de Java e a linguagem de programação em blocos Scratch.

REVISÃO DA LITERATURA

3.1 Considerações iniciais

Neste capítulo, busca-se levantar o estado da arte do ensino de programação paralela na literatura, elencando os trabalhos que possuem maior relação com o presente projeto.

Inicialmente, são descritos os procedimentos realizados para o mapeamento sistemático, assim como a metodologia de pesquisa que tomou-se como base para sua realização. Os resultados obtidos por meio do mapeamento sistemático são apresentados ao final da seção.

Em seguida, são relatados pontualmente os trabalhos que melhor se relacionam com a proposta de pesquisa deste projeto, as principais contribuições e limitações em relação ao trabalho.

3.2 Mapeamento Sistemático

Define-se como mapeamento sistemático a metodologia de pesquisa eficiente, rígida e reprodutível, que presta-se a oferecer procedimentos para a obtenção de material acadêmico relevante para um determinado tópico de interesse e a classificação do mesmo (KITCHENHAM, 2004).

Resultados relevantes tem sido publicados no que se refere à aplicabilidade e eficiência do desenvolvimento de revisões ou mapeamentos sistemáticos da literatura, quando comparados com revisões tradicionais. Obtêm-se uma proporção superior de textos relevantes em um tópico específico quando aplicada uma das metodologias citadas do que quando praticado uma metodologia *ad hoc* (BRERETON *et al.*, 2007).

Este mapeamento foi realizado inicialmente com a colaboração de Guilherme Martins no primeiro semestre de 2018. No segundo semestre de 2019, foi feita a atualização do mesmo

por Thiago Durães, além da realização do processo de *snowballing*.

3.2.1 *Objetivo do Mapeamento*

Espera-se com este mapeamento sistemático conhecer o estado da arte tratando-se de metodologias de ensino empregadas em disciplinas de computação ou programação paralela. Assim, desejamos elencar estudos que considerem o aperfeiçoamento o ensino do referido conteúdo, identificando as melhores práticas de ensino.

O principal objetivo deste mapeamento sistemático foi o reconhecimento do estado da arte referente às metodologias de ensino e recursos educacionais aplicados a aprendizagem de *High Performance Computing*. Portanto, desejamos elencar os estudos que sugiram a implementação de melhorias na qualidade do ensino deste conteúdo.

3.2.2 *Questões de Pesquisa*

Com base no objetivo especificado, foram levantadas as questões de pesquisa, cuja elaboração foi baseada na estratégia PICO. O método PICO é um mecanismo utilizado para guiar a formulação das questões de pesquisa, baseadas nos seguintes elementos ([PREGUNTA, 2007](#)):

- *Population* (População): define população, contexto e/ou situação-problema;
- *Intervention* (Intervenção): contexto da revisão ou mapeamento sistemático;
- *Comparison* (Comparação): No caso de pesquisa clínica e se necessário, uma intervenção de comparação com o contexto da revisão/mapeamento;
- *Outcomes* (Resultados): define os resultados desejados ou indesejados.

Esta análise permite garantir que a questão mantém o foco na necessidade (no problema) e facilita sua análise crítica e avaliação. Para este trabalho, esta análise será considerada da seguinte forma:

- *Population* (População): Disciplinas inseridas em um contexto de cursos superiores de computação referentes ao conteúdo de Programação Concorrente e Programação Paralela;
- *Intervention* (Intervenção): Metodologias de ensino e ferramentas utilizadas para o ensino de programação concorrente e paralela;
- *Comparison* (Comparação): não aplicável neste mapeamento;
- *Outcomes* (Resultados): Resultados ou consequências da aplicação destes recursos e métodos no ensino e aprendizado de programação concorrente/paralela.

Desta maneira, foram elaboradas questões de pesquisa visando alcançar os propósitos deste mapeamento sistemático:

QP1: Quais abordagens têm sido utilizadas no ensino de programação concorrente?

QP2: Quais abordagens têm sido utilizadas no ensino de programação paralela?

QP3: Qual o impacto causado pela aplicação de tais abordagens de ensino (considerando aquelas que têm sido utilizadas)?

3.2.3 Bases de pesquisa e Palavras-chave

Considerou-se como bases de pesquisa as fontes de literatura mais expressivas no contexto da computação: *Scopus*, *IEEE Xplore* e *ACM Digital Library* (BRERETON *et al.*, 2007).

As seguintes palavras-chave foram estabelecidas de modo a abranger o tema e as questões de pesquisa. Elas foram classificadas em função de sua importância e sinônimos considerados.

Tabela 3 – Palavras-chave e sinônimos deste mapeamento

<i>Concurrent Programming</i>	<i>Concurrent Computing</i>
<i>Parallel Programming</i>	<i>Parallel Computing</i>
<i>Teach</i>	<i>Educational Resource, ER</i>

3.2.4 String de busca

Após a definição da temática de interesse e das palavras-chave a serem consideradas, definiu-se, conforme apresentado no Quadro 1, a *string* de busca para investigação nas bases de dados destacadas anteriormente.

Quadro 1 – *String* de busca

(“concurrent programming” OR “concurrent computing” OR “parallel programming” OR “parallel computing”) AND (“teach” OR “educational resource” OR “ER”)
--

Por meio da submissão da *string* de busca definida no Quadro 1 nas bases elencadas anteriormente, foram obtidos os resultados demonstrados na Tabela 4, a qual ilustra a quantidade de artigos obtida em cada uma das bases referenciadas e, finalmente, a quantidade total de artigos obtida na primeira execução deste processo em 2018.

Tabela 4 – Resultado da pesquisa.

Base de Dados	Resultado
<i>Scopus</i>	370
<i>IEEE Xplore</i>	127
<i>ACM Digital Library</i>	198
<i>Proquest</i>	2
Total	697
Total de estudos sem repetição	515

A ferramenta utilizada para registrar a quantidade de estudos sem repetição foi o *Mendeley* (<<https://www.mendeley.com/>>), que permite a exportação das publicações extraídas de cada base de pesquisa, na configuração de um *bibtex*, a fim de analisar as duplicatas.

3.2.5 Seleção

No intuito de estabelecer os critérios de inclusão e exclusão dos materiais obtidos através das buscas em bases de dados, define-se, na lista a seguir, os parâmetros que regem a inclusão ou exclusão dos referidos trabalhos.

- **Critérios de Inclusão**

- CI 1 Estudos relacionados ao ensino de programação concorrente;
- CI 2 Estudos relacionados ao ensino de programação paralela.

- **Critérios de Exclusão**

- CE 1 Estudos que não tenham relação com os temas de ensino de programação concorrente ou paralela;
- CE 2 Estudos incompletos (chamadas de evento, resumos, etc);
- CE 3 Estudos que não sejam focados em quaisquer subáreas da computação;
- CE 4 Estudos classificados como literatura cinza,
- CE 5 Estudos cujo idioma não seja inglês;
- CE 6 Estudos que considerem apenas o recurso educacional, não a metodologia.

3.2.6 Metodologia de Avaliação

No intuito de refinar os resultados obtidos inicialmente e, efetivamente, classificar os estudos com base na relevância observada, utilizou-se um método de depuração em 3 fases, de acordo com as indicações de [Kitchenham e Charters \(2007\)](#). O referido processo ocorre por meio da aplicação sucessiva, em 3 etapas, dos critérios de inclusão e exclusão em partes diferentes dos estudos obtidos inicialmente.

- **Fase I: Leitura dos títulos e resumos:** Em um primeiro momento, foram analisados os títulos e resumos (*abstracts*) de todo material retornado inicialmente. Aplicaram-se os critérios de inclusão e exclusão com o intuito de refinar os resultados;
- **Fase II: Leitura dos resultados e conclusões:** Em um segundo momento, a partir dos resultados obtidos na primeira fase, aplicou-se os critérios de inclusão e exclusão, de forma similar à praticada anteriormente, nas seções de resultados e conclusões de cada um dos estudos;
- **Fase III: Leitura dos estudos completos:** Finalmente, os estudos remanescentes das fases anteriores são examinados integralmente, realizando a leitura de todas as seções. Ao final, obtemos a listagem final de artigos para o mapeamento.

A Tabela 5 demonstra os estudos selecionados em cada etapa de leitura executada.

Fase	Quantidade de Estudos Selecionados
<i>I</i>	256
<i>II</i>	136
<i>III</i>	93

Tabela 5 – Estudos selecionados após a execução dos critérios de inclusão e exclusão.

3.2.7 Avaliação de qualidade

Com o objetivo de classificar, efetivamente, os resultados em função da relevância observada, Dybå e Dingsøyr (2008) propõem uma série de critérios binários que classificam a qualidade dos trabalhos de uma amostra no contexto de uma revisão ou mapeamento sistemático. A partir deste fundamento, definiram-se parâmetros relacionados a qualidade, retratados na Tabela 6.

Tabela 6 – Critérios de qualidade

Número	Critério
1	O estudo é baseado em pesquisa (ou trata-se de um relatório de lições aprendidas)?
2	Os objetivos são bem definidos?
3	Foi desenvolvida ou utilizada uma metodologia de ensino diferente do modelo tradicional?
4	Foram desenvolvidos novos recursos de ensino?
5	Os resultados são exibidos de forma clara?
6	O trabalho é conclusivo?
7	O estudo é reproduzível?
8	O trabalho tem comprovação estatística?

Em cada estudo, atribui-se o valor de 1 (sim) ou 0 (não) para cada um dos referidos parâmetros. Finalmente, a pontuação dos trabalhos são somadas individualmente e os estudos são, dessa forma, classificados em conformidade com os parâmetros de qualidade. A ordenação das publicações por pontuação nesses critérios está disponível em <http://bit.do/pontuacao_MS>.

É válido destacar que os valores atribuídos não representam quaisquer afirmações acerca da validade e qualidade total do material analisado. Tais elementos, meramente, representam a medida de distância e adequação às questões de pesquisa definidas para o presente trabalho. Desta forma, os valores mais altos representam maior grau de similaridade e relevância, considerando os objetivos aqui definidos.

3.2.8 Extração de Dados

O procedimento para a extração de dados foi conduzido a partir da instituição de um formulário que especifica os atributos considerados mais pertinentes de cada trabalho investigado. Tais atributos, aplicados em cada um dos estudos considerados, são: data de extração dos dados, autores, título da publicação, ano de publicação, *abstract*, *keywords*, fonte (onde foi publicada), base de dados em que foi obtida, tipo de publicação, abordagem/metodologia, tecnologias utilizadas e conclusões. As informações extraídas de cada trabalho pode ser acessada em http://bit.do/dadosbrutos_MS.

3.2.9 Resultados do Mapeamento

Depois de empregar progressivamente os parâmetros tanto de inclusão como de exclusão, discutidos na Seção 3.2.5, chegou-se ao número final de 93 estudos. Em seguida, foi feita a classificação destes trabalhos de acordo com sua significância, conteúdo e características, com base nos fatores apresentados na Seção 3.2.7. Os dados obtidos a partir da composição remanescente de artigos são dispostos nesta seção.

A primeira informação obtida trata-se da quantidade de estudos publicados, por ano. A Figura 5 ilustra esta distribuição.

Por meio do gráfico anterior é possível perceber que o interesse no problema de pesquisa definido, existe aproximadamente desde a década de 1990, portando um crescimento significativo a partir de 2010. Tal ocorrência coincide e associa-se com o crescimento do interesse em *High Performance Computing* para processamento de algoritmos com alto custo computacional. Além disso, coincide com o período próximo a publicação da grade curricular da ACM em 2013 e ao desenvolvimento de novos *frameworks* e tecnologias, por exemplo CUDA e novos modelos arquiteturais *many-core*.

Além disso, as *keywords* dos artigos foram agrupadas e dispostas em um diagrama *word-cloud*, demonstrando os termos de indexação mais frequentes, que pode ser visto na Figura 6. A dimensão de cada termo ilustrado caracteriza a relação da frequência de sua utilização.

Analisando o diagrama, podemos perceber que as palavras-chave mais comuns são referentes ao ensino e educação. Em seguida, verificamos a maior frequência de termos relacionados a programação ou computação paralela. Finalmente, nota-se termos referentes a tecnologias

Figura 5 – Quantidade de estudos publicados por ano

Figura 6 – Relação das *keywords* mais frequentes

utilizadas, tal como OpenMP ou Cuda. É possível verificar, dessa forma, o foco da maior parte dos artigos.

Além disso, foi contabilizado o número de artigos indexados por base de pesquisa. A Tabela 7 demonstra a quantidade por bases.

Foi classificada, também, o número de artigos por categoria (artigo em periódico, *paper*

Por fim, julgamos relevante categorizar o aspecto relativo à principal contribuição do artigo. Os trabalhos foram separados em duas classificações: os que apresentam metodologias de ensino para programação/computação paralela e os que apresentam recursos educacionais. A Tabela 9 demonstra o resultado da classificação.

Tabela 9 – Quantidade de Publicações por abordagem desenvolvida

Abordagem	Quantidade de Publicações
Metodologia de ensino	64
Recursos educacionais	29

Percebe-se que uma parte expressiva dos trabalhos é relacionada ao estudo de metodologias de ensino de programação paralela. Os recursos educacionais e materiais de suporte ao aprendizado, especificamente os que possuem acesso público, são, em comparação com o número de trabalhos referentes às metodologias de ensino, consideravelmente reduzidos.

3.2.10 Limitações

O processo de determinação da *string* de busca e questões de pesquisa, componentes do protocolo de revisão praticado no presente trabalho, que embora trate-se de uma atividade criteriosa e bem fundamentada, pode gerar incoerências nos dados extraídos e polarização (*bias*) do material obtido, assim como documentado em (KITCHENHAM; CHARTERS, 2007).

Especificamente, nas etapas mencionadas, sujeita-se à eventualidade de imprecisões, dada a intrínseca subjetividade dos procedimentos de definição e classificação praticados neste trabalho. Desta forma, admite-se que estudos podem ser ignorados em função dos atributos e bases de dados utilizados para pesquisa.

3.2.11 Conclusões

Por meio do levantamento destes estudos, foi possível ilustrar o estado da arte referente ao ensino de computação e programação paralela, em específico, as metodologias e recursos educacionais aplicados em disciplinas deste conteúdo. Dessa forma, pode-se concluir alcançamos os objetivos elencados para este mapeamento, estabelecidos na Seção 3.2.1, possuindo, assim, resultados para as questões de pesquisa levantadas inicialmente na Seção 3.2.2.

A listagem final de trabalhos obtidos possibilitou examinar técnicas, padrões, mecanismos e procedimentos utilizadas no ensino do conteúdo de programação paralela. A Tabela 10 exhibe o número de trabalhos encontrados, identificados por estratégia educacional (metodologia, abordagem e técnica) utilizada. As estratégias não contabilizadas na tabela não apresentam uma estratégia de aprendizagem determinada.

É possível perceber que existem diversas iniciativas que propõem a aplicação de metodologias ativas no ensino de programação paralela, principalmente Aprendizado baseado em

Tabela 10 – Quantidade de publicações por método

Método	Quantidade
Metodologia tradicional	12
Aprendizado baseado em problemas	9
Programação por padrões	8
Aprendizado orientado a exemplo	4
Aprendizado baseado em modelo	4
Sala de aula invertida ou parcialmente invertida	3
Aprendizado baseado em projetos	3
Gameificação	2
Aprendizado baseado em módulos	2
Aprendizado baseado em pesquisa	2

problemas e Programação por padrões. Entretanto, apesar destes trabalhos, há uma quantidade significativa de estudos que consideram o ensino do tema utilizando a metodologia tradicional. Tais estudos tratam da formação de disciplinas específicas para o ensino de programação paralela, assim como a reestruturação de ementas com o objetivo de abranger estes tópicos. Além disso, são apresentadas experiências de cursos de extensão realizados em universidades. Destaca-se que não foram encontradas iniciativas de aplicação da abordagem de Aprendizagem Baseada em Equipes dentro desta temática.

De forma complementar, acessando o endereço http://tiny.cc/class_metodologia, encontra-se uma tabela que apresenta a distribuição completa dos estudos por metodologia/abordagem de ensino.

Subsequentemente, foram classificados os recursos educacionais encontrados. As quantidades de publicações por tipo de recurso são exibidas na Tabela 11.

Tabela 11 – Quantidade de publicações por recurso educacional

Recurso	Quantidade
Simuladores ou ferramentas gráficas	13
Cluster ou infraestrutura	5
Framework ou API	3
Repositório ou curso online	4
Correção automática	2
Maratona/desafio de programação	2

No endereço http://tiny.cc/class_RE, são apresentadas, em uma tabela, as referências dos trabalhos relacionados aos recursos educacionais por classificação.

Referente aos resultados descritos nos artigos sobre o emprego das metodologias e recursos, podemos verificar que são favoráveis e otimistas, destacando o crescimento da curiosidade pelo assunto por parte dos alunos, atenuação de evasão em cursos deste tema, crescimento no desempenho dos alunos em qualidade de código e crescimento das aprovações nas disciplinas.

Vale ressaltar que, considerando produtos decorrentes de fatores subjetivos, a principal ameaça a validade é, conseqüentemente, a possibilidade de subjetividade na análise das informações e resultados obtidos, que, em sua maioria, são autenticados mediante testes escritos, questionários e outras ferramentas subordinadas a pontos de vista, pensamentos, consensos, ideologias e falhas humanas.

3.3 Principais Trabalhos Relacionados

Esta seção apresenta os principais trabalhos relacionados ao tema deste projeto. Com intuito de analisar as metodologias utilizadas no ensino de programação concorrente e evitar desconsiderar trabalhos relevantes da área, o levantamento mostra trabalhos sobre o ensino de programação paralela com alunos de diferentes estágios da graduação. Diversos estudos, principalmente entre os anos de 2010 a 2015, demonstram a alteração das grades curriculares de cursos de graduação em computação de forma a abranger os conceitos de computação paralela em diferentes estágios da graduação, principalmente tendo como base as iniciativas curriculares da IEEE ([PRASAD et al., 2012](#)) e ACM ([CURRICULA, 2013](#)) em 2012 e 2013 respectivamente.

Entretanto, no que refere-se aos principais trabalhos relacionados, foram encontrados 14 estudos com iniciativas de ensino que propõem o ensino de programação paralela em estágios iniciais ou anteriores ao curso de graduação direcionado a estudantes sem formação sólida nos conhecimentos de computação. Estes estudos são demonstrados nas Tabelas 12 e 13, ordenadas por ano de publicação, e descritas, sinteticamente, em seguida.

Tabela 12 – Principais trabalhos relacionados I

Título	Autor	Keywords	Ano
Is Teaching Parallel Algorithmic Thinking to High School Students Possible? One Teacher's Experience	Shane Torbert, Uzi Vishkin, Ron Tzur, David J. Ellison	High school, Parallel algorithmic thinking, PRAM algorithms, XMT	2010
Measuring CS1 perceptions of parallelism	Brian Rague	Assessment, Concurrent Programming, Content Inventory, Parallelism	2011
Breadth in depth: A 1st year introduction to parallel programming	Thomas R. Gross	CS education, Parallel programming, Software engineering	2011
An Experience of Early Initiation to Parallelism in the Computing Engineering Degree at the University of Murcia, Spain	Manuel E. Acacio, Javier Cuenca, Lorenzo Fernández, Ricardo Fernández-Pascual, Joaquín Cervera, Domingo Giménez, M. Carmen Garrido, Juan A. Sánchez Laguna, José Guillén, Juan Alejandro Palomino Benito, María-Eugenia Requena	Computing Engineering Degree, Parallel algorithms, Parallel architecture, Parallel programming, Parallelism teaching	2012
Parallel from the Beginning: The Case for Multicore Programming in the Computer Science Undergraduate Curriculum	Yousun Ko, Bernd Burgstaller, Bernhard Scholz	Computer science education, Course design, Introductory programming, Multicore computing development/education	2013

Tabela 13 – Principais trabalhos relacionados II

Título	Autor	Keywords	Ano
Minimum time, maximum effect: Introducing parallel computing in CS0 and STEM outreach activities using scratch	Russell Feldhausen, Scott Bell, Daniel Andresen	Scratch, HPC, Parallel Programming, Outreach, K-12, CS0	2014
Habanero-Java library: A Java 8 framework for multicore programming	Shams Imam, Vivek Sarkar	Habanero-Java Library, Java Task Parallel Library, Lambda Expressions, Task Parallelism	2014
Early learning in parallel programming	Igor N. Skopin	Activity-based approach to learning, Cognitive development, Computational model, Parallel programming, Problem tasks, Resource constraints, Teaching patterns	2014
HPC/PDC Immunization in the Introductory Computer Science Sequence	David Valentine	Concurrent Programming, Parallel Programming, Types of Simulation, Monte Carlo, Computer and Information System Education	2014
Injecting parallel computing into CS2	Joel C. Adams	CS2, Multithreading, OpenMP, Parallel, Patterns, Shared memory	2014
Experiences with teaching a second year distributed computing course	Rizos Sakellariou	Abstracting, Curricula, Distributed computer systems, Economic and social effects, Education	2016
Survey of software visualization systems to teach message-passing concurrency in secondary school	Cédric Libert, Wim Vanhoof	Concurrency, Message passing, Teaching	2017
Integrating Parallel Computing in Introductory Programming Classes: An Experience and Lessons Learned	Sheikh Ghafoor, David W. Brown, Mike Rogers	Introductory programming undergraduate education, Parallel and distributed computing	2017
Pervasive parallel and distributed computing in a liberal arts college curriculum	Tia Newhall, Andrew Danner, Kevin C. Webb	CS curriculum, Parallel and distributed computing	2017

O estudo realizado em [Torbert et al. \(2010\)](#) apresenta um modelo baseado no método tradicional para o ensino de conceitos de computação paralela para estudantes do ensino médio utilizando, no primeiro semestre, passagem de mensagens com MPI e, no segundo semestre, pthreads, OpenMP e Cuda com Explicit Multi-Threading (XMT). Participaram deste trabalho estudantes de ensino médio com experiência prévia em programação com Java e Python matriculados em um curso optativo de computação paralela extra-classe. O autor demonstra as técnicas e exercícios utilizados e suas experiências após a realização da disciplina, concluindo que o pensamento paralelo (*parallel thinking*) pode ser inserido de maneira eficaz a estudantes do ensino médio. Esse artigo fornece evidências significativas sobre a capacidade de ensino por meio do sistema XMT e defende seu uso no ensino de Ciência da Computação. Apesar do artigo considerar estudantes do ensino médio, a proposta de ensino é para alunos que possuam

algum conhecimento prévio em computação e programação com Java e Python, o que constitui a principal diferença para com este projeto, no qual propomos tópicos de ensino para alunos sem conhecimento prévio em computação. Além disso, o principal fator avaliado no artigo é a possibilidade de ensino com a abordagem XMT PAT. Em nosso trabalho, validamos o ensino utilizando diferentes abordagens, técnicas e contextos.

O autor [Rague \(2011\)](#) demonstra em seu trabalho a viabilidade estatística de ministrar os conceitos introdutórios de programação paralela para alunos do primeiro ano de graduação em ciência da computação. Apresenta-se o questionário POPS (*Perceptions of Parallelism Survey*), instrumento de avaliação da percepção do paralelismo e conteúdo associado pelos alunos. Os resultados desse estudo mostram a importância de atividades práticas em laboratório que foram utilizadas em cursos de programação paralela. Esse trabalho difere do nosso, pois buscamos ensinar programação paralela enquanto o autor apresenta um instrumento para medir a percepção de paralelismo dos indivíduos.

O trabalho de [Gross \(2011\)](#) descreve a metodologia proposta para o ensino de programação paralela no segundo semestre do primeiro ano de graduação em ciência da computação, no contexto da universidade ETH de Zurique, Suíça. O autor propõe um modelo cronológico, onde conceitos fundamentais da abstração são apresentados aos alunos em uma estrutura teórico/prática. O autor afirma que o experimento foi aprovado pelos alunos inscritos na turma experimental, porém considera o ensino apenas de abstrações de paralelismo, fazendo tão somente uma introdução superficial ao pensamento paralelo e deixando a programação em si para períodos mais avançados. A principal distinção desse trabalho para o nosso é que neste visamos despertar o pensamento paralelo de forma prática, possibilitando que os alunos desenvolvam código paralelo desde o início.

Uma experiência de ensino de programação paralela em estágios iniciais da graduação também é descrita em [Acacio et al. \(2012\)](#). O contexto deste estudo ocorre em uma disciplina para o segundo ano de uma turma de graduação em Engenharia de Computação utilizando OpenMP e MPI. O conteúdo referente a conceitos e tecnologias de programação paralela ministrados no curso foram propostos com base nos tópicos sugeridos pelo Comitê Técnico de Processamento Paralelo da IEEE. Além disso, é apresentado o plano de aula utilizado e a divisão dos tópicos por período e disciplina. Os autores afirmam que esta experiência inicial pode servir de exemplo para melhorar o ensino do paralelismo em aulas de computação em outras universidades. Na data da publicação, a disciplina estava em andamento, portanto o artigo não apresenta resultados ou avaliações de aprendizagem. Este trabalho é diferente do nosso porque apresenta tópicos de programação paralela para alunos no segundo ano da graduação, ou seja, que possuem um conhecimento considerável em computação. Além disso, por se tratar de um experimento em andamento, não apresenta nenhum tipo de avaliação técnica ou qualitativa do aprendizado dos participantes.

Baseado na recomendação curricular da ACM de 2008 ([CASSEL et al., 2008](#)) de que é

necessário inserir paralelismo na graduação, um curso de programação paralela é apresentado em [Ko, Burgstaller e Scholz \(2013\)](#). O curso é dividido em 5 módulos: paralelismo e indeterminismo de execução, programação baseada em thread-and-lock, desempenho de programas paralelos, aceleração de hardware usando OpenCL, e fluxos paralelos com StreamIt. Cada módulo compreende um conceito fundamental de programação paralela utilizando as ferramentas StreamIt e OpenCL em GPGPUs. Os experimentos realizados obtiveram a participação de alunos no segundo e terceiro semestre da graduação em computação. A diferença deste trabalho para o nosso é que nele os autores apresentam, para cada módulo, os exemplos e exercícios usados enquanto nós apresentamos a proposta de agrupar conceitos de paralelismo que possam ser ensinados para alunos sem formação prévia, inclusive no primeiro semestre do curso, e procuramos validar essa proposta por meio de 5 experimentos. Além disso, o artigo não fundamenta a escolha desses 5 módulos que representam os conceitos de programação paralela, inclusive não considera nem cita alguma relação com os tópicos de computação paralela definidos pela IEEE ([PRASAD et al., 2012](#)) ou ACM ([CURRICULA, 2013](#)), iniciativas fundamentais para comunidade acadêmica em relação ao ensino de programação paralela. Por fim, o nosso projeto diferencia-se desse trabalho pois demonstra uma avaliação técnica do desempenho dos alunos e uma avaliação qualitativa para cada curso realizado.

Um curso de introdução à programação paralela é apresentado em [Feldhausen, Bell e Andresen \(2014\)](#). Nesta abordagem, utiliza-se a linguagem de programação em blocos Scratch visando definir um ambiente amigável e visual, que permite a representação teórica fundamental em alto nível, de forma que os alunos não sejam intimidados pela complexidade do conteúdo. As atividades práticas são descritas e disponibilizadas publicamente, caracterizando alguns recursos educacionais abertos fornecidos pelo trabalho. Os autores reportam uma considerável satisfação dos alunos quando submetidos à metodologia, com um aumento no número de estudantes que consideram entender o que é programação paralela. Os alunos também se consideram capazes de aprender a escrever programas paralelos, promovendo a popularização de computação de alto desempenho e despertando o interesse dos mesmos nesta temática. A principal distinção para nosso projeto é que introduzimos programação paralela de forma prática com foco no desenvolvimento de código e melhora do desempenho de algoritmos, não somente nas abstrações em alto nível, que constitui o principal objetivo do Scratch.

[Imam e Sarkar \(2014\)](#) apresentam um recurso educacional para o apoio ao ensino de programação paralela em dispositivos *multicore* chamado de Habanero-Java Library (HJlib). Trata-se de um *framework*, desenvolvido na linguagem de programação Java que facilita o ensino de conceitos fundamentais da disciplina de programação paralela ao oferecer uma abstração simplificada dos mecanismos de *threads* e sincronização. O recurso foi aplicado em uma turma de "Fundamentos de Programação Paralela" no segundo ano de graduação em ciência da computação. Os resultados demonstram que o *framework* permitiu que os alunos, considerando a base de conhecimento prévio em Java, se concentrassem nos fundamentos da programação paralela e no desenvolvimento de algoritmos, ao invés das complexidades específicas do uso de uma

API como a `java.util.concurrent`. O HJlib permite que programadores com um conhecimento básico de Java iniciem o aprendizado de conceitos de programação paralela. Neste projeto nós sugerimos a organização dos conceitos de programação paralela visando ao ensino antecipado do conteúdo utilizando linguagem C, enquanto que nesse artigo os autores propõem um instrumento facilitador para a introdução de programação paralela utilizando Java.

O trabalho de Skopin (2014) discute a implementação de uma metodologia baseada na rejeição de padrões de pensamento no ensino à disciplina de programação paralela nos primeiros períodos do curso de graduação em ciência da computação. A principal ideia da metodologia proposta é a separação de dois tipos de atividade na construção de um programa paralelo: o desenvolvimento de um esquete do programa, que não leva em consideração restrições de recursos, e o mapeamento deste esboço para um computador real. O autor destaca que a principal dificuldade foi encontrar problemas que ofereçam soluções com um paralelismo natural e que possam ser mapeados em um computador real, principalmente em se tratando de abstrações de paralelismo. A principal diferença deste artigo para nosso projeto é que o autor propõe a construção de esquetes de algoritmos paralelos em alto nível independentemente de qualquer linguagem de programação. Nós sugerimos a codificação de algoritmos permitindo que o aluno desenvolva programas paralelos de fato, desde o início de sua aprendizagem. Como o próprio autor afirma, por se tratarem de modelos abstratos, os esquetes podem ser diferentes da codificação em si posteriormente. Além disso, o autor não apresenta tópicos ou conceitos a serem ensinados, mas sim exemplos que podem ser esquematizados de acordo com a metodologia. Por fim, não há relatos de experimentos, aulas ou cursos que utilizaram a metodologia que demonstrem resultados reais de ensino em sala de aula e validem a metodologia.

No trabalho de Valentine (2014) é realizada uma experiência de apresentar conceitos iniciais de programação paralela definidos na iniciativa curricular da IEEE (PRASAD *et al.*, 2012) em disciplinas introdutórias do curso de bacharelado em ciência da computação. O autor propõe uma abordagem orientada a exemplos em que são apresentadas aos estudantes, em sequência, a versão estruturada e paralela de algoritmos que os mesmos já conheçam ou tenham implementado, para que os alunos possam visualizar as diferenças estruturais entre as implementações e conceitos tais como *Speed Up* e Eficiência. Os resultados demonstram que a recepção dos alunos à metodologia foi positiva e o modelo proposto permitiu implantar com êxito os tópicos de HPC na grade curricular principal do curso. Os alunos puderam obter a percepção da importância do paralelismo nas aplicações atuais. O autor também afirma que o contato com paralelismo em estágios iniciais da graduação, especificamente primeiro e segundo ano, foi o fator fundamental para a mudança na perspectiva sobre paralelismo por parte dos alunos. A diferença desse trabalho para o nosso é que no primeiro as concepções de paralelismo são apenas demonstradas por exemplos de códigos, não envolvendo a prática de código paralelo por parte do aluno. Neste projeto, além de apresentarmos exemplos de algoritmos e códigos paralelos, os estudantes no primeiro ano de graduação são estimulados a desenvolver suas próprias soluções paralelas. Além disso, o artigo não apresenta nenhuma definição de conteúdos para o ensino

antecipado.

A partir da experiência demonstrada em [Adams \(2014\)](#), o autor afirma que o segundo ano do curso de graduação de computação é o ideal para expor o conteúdo de programação paralela, pois os alunos não são mais novatos, facilitando a resolução de problemas mais complexos. O trabalho demonstra a experiência de sessões de ensino em laboratório com metodologia baseada em padrões de paralelismo e mostra que foi possível implementar com os alunos *loops* paralelos, o conceito de *fork-join*, operação de redução e o algoritmo de mestre-escravo. Esse trabalho é diferente do nosso, pois não apresenta fundamentação na literatura acadêmica dos conceitos selecionados para ensino nas aulas ou qual foi a base para defini-los. Além disso, o autor visa o ensino de alunos no segundo ano e nós visamos ensinar alunos totalmente novatos em computação.

O trabalho de [Sakellariou \(2016\)](#) apresenta uma disciplina de programação paralela para o segundo ano da graduação da Universidade de Manchester, juntamente com as experiências adquiridas ao ministrar o curso. Tal disciplina é mandatória para os alunos após concluírem as disciplinas de Arquitetura de Computadores. Os alunos participantes possuíam conhecimento prévio de programação em Java e conhecimento básico em linguagem C, portanto, apesar de cursarem o segundo ano de graduação, não caracterizavam-se como alunos totalmente principiantes. O autor conclui que o ensino de computação paralela para estudantes de graduação deixou de ser opcional, tornando-se um conteúdo mandatório nas grades dos cursos, inclusive desde os primeiros anos. A diferença desse trabalho para o nosso é que nós visamos o ensino de alunos totalmente iniciantes em computação, ou seja, anterior ao segundo ano de graduação, onde já possuem alguma experiência. Desta forma, os módulos da disciplina propostos neste artigo consideram alunos que já possuem conhecimento prévio em computação.

Em [Libert e Vanhoof \(2017\)](#) são analisados 27 sistemas de visualização de software de acordo com 8 critérios estabelecidos no intuito de suportar cursos introdutórios de programação baseados em concorrência com passagem de mensagens. Dois sistemas foram identificados como os melhores nesta categoria devido a seus paradigmas e capacidade de depuração: *ToonTalk* e *Snap!*. Entretanto, por se tratar de um estudo preliminar, ainda é necessário avaliar precisamente o quão difícil seria adicionar um modelo de passagem de mensagens adequado ao *Snap!* e coletar a percepção dos alunos adolescentes sobre o *ToonTalk*, além de avaliar os impactos causados pelo uso dos recursos. Nesse artigo o foco da análise são as ferramentas de auxílio para o ensino introdutório de programação paralela, enquanto neste projeto de mestrado nós estudamos o conteúdo ensinado de programação paralela em si e a viabilidade de ensino para estudantes sem formação prévia em computação.

No trabalho de [Ghafoor, Brown e Rogers \(2017\)](#) os autores introduziram, com diferentes níveis de profundidade, conceitos de paralelismo em algumas disciplinas dos primeiros semestres do curso de computação na Universidade de Tecnologia do *Tennessee*. Além disso, um a dois dias de palestra por semestre foram dedicados a introduzir porque a programação paralela é necessária,

arquitetura paralela, conceitos básicos e como esta programação difere da codificação sequencial. Na disciplina do primeiro semestre, Introdução à resolução de problemas e programação de computadores, o objetivo é apresentar aos alunos codificação básica com OpenMP, o modelo *fork-join* de processamento paralelo, assim como tornar o aluno mais familiarizado com as ideias de memória compartilhada vs. distribuída, projetar programas paralelos e definir as diferenças entre concorrência e paralelismo. O conteúdo foi dividido em dois módulos: O primeiro consiste em uma simples demonstração de somatório *fork-join* e permitir que os alunos criem um programa paralelo básico e observem o *Speed-up*. O segundo módulo, mais complexo, guia os alunos à manipulação de *arrays* paralelos para a manipulação de imagem. Os dois módulos ajudam a reforçar os conceitos abordados na palestra principal. No segundo semestre, na disciplina Estruturas de Dados e Algoritmos, os docentes reforçaram o conteúdo abordado no semestre anterior e, ao mesmo tempo, expandiram sua capacidade de pensar em paralelo. Novamente, o conteúdo foi dividido em dois módulos: O primeiro permite que os alunos voltem a observar *Speed-up* em programação paralela, implementando o algoritmo *Bubble Sort* paralelizado. O segundo trabalha com modificação de imagens, mas desta vez utilizando *pipelining* e o modelo produtor-consumidor de paralelização. Os resultados obtidos mostram que, ao longo do ano letivo, os estudantes puderam compreender o conteúdo de paralelismo apresentado com mais dificuldade em alguns tópicos do que outros. Os autores afirmam que essas implementações, por mais limitadas que possam ser, têm tido algum sucesso e apontam para vários resultados promissores que avançam.

O trabalho de [Ghafoor, Brown e Rogers \(2017\)](#) é o que mais se assemelha a este projeto devido à exposição de programação paralela para alunos no primeiro ano de computação inserindo alguns conceitos e exemplos de algoritmos paralelos. A diferença para nosso trabalho é que o conteúdo de paralelismo é introduzido de forma mais profunda, direta e prática em cursos específicos de programação paralela para alunos neste contexto e, inclusive, de outras áreas além da computação, comparando com o desempenho de alunos que já possuem conhecimento prévio em computação. Além disso, os autores destacam no artigo que é necessário formalizar a introdução ao conteúdo. Em nosso trabalho, sugerimos uma organização dos tópicos de paralelismo propostos pela IEEE ([PRASAD et al., 2012](#)), que serviu de base para ensinarmos programação paralela para os alunos. Por fim, o artigo não considera uma avaliação qualitativa da introdução do conteúdo de paralelismo antecipadamente. Nos experimentos realizados neste projeto avaliamos o impacto qualitativo em relação à sensação e impressão dos alunos quanto ao formato do curso e conteúdo apresentado.

Em [Newhall, Danner e Webb \(2017\)](#) são apresentadas as alterações realizadas na grade curricular de computação do *Swarthmore College*, sendo que o conteúdo de determinadas disciplinas foi retificado, inserindo-se uma nova disciplina na grade curricular, chamada Introdução a Sistemas Computacionais. Esta disciplina, em um nível intermediário da grade, tem o intuito de garantir que todos os alunos possuam conhecimentos básicos de sistemas de computação, além de oferecer uma introdução à computação paralela, preparando os estudantes para disci-

plinas futuras. O único requisito para sua realização é "Introdução a computação", ministrada no primeiro ano de graduação. A iniciativa buscou adequar os tópicos de computação paralela sugeridos em [Prasad *et al.* \(2012\)](#) às disciplinas do curso de ciência da computação e, criando uma nova disciplina intermediária, suprir a necessidade de conhecimentos prévios em diversas áreas como representação de dados binários, circuitos, *assembly*, hierarquia de memória, compilação e execução de programas, abstrações de sistemas operacionais, arquiteturas paralelas e programação paralela. Tal disciplina antecipa o ensino de paralelismo combinado com conceitos de outras áreas da computação. Entretanto, diferentemente da iniciativa deste projeto de mestrado, a disciplina descrita no artigo, além de não ser exclusiva para o ensino de programação paralela, não é realizada em um estágio totalmente inicial da graduação. Outras modificações realizadas na grade curricular também foram feitas em disciplinas de nível intermediário na grade, permanecendo o desafio de antecipar o ensino deste tema para alunos sem conhecimento prévio em computação, ou seja, o primeiro ano.

3.3.1 *Lacuna*

A partir dos trabalhos descritos na seção 3.3, é possível verificar que a antecipação do ensino de programação paralela é uma preocupação da comunidade acadêmica. Em diferentes estudos e iniciativas, autores enfatizam que o ensino de programação paralela deve ser antecipado na graduação e apresentam suas contribuições com experimentos que foram implementados, como modificações em grades curriculares, distribuição de tópicos de paralelismo em disciplinas e cursos de extensão em universidades.

Muitos trabalhos encontrados na literatura relatam experiências do uso de metodologias de ensino ou ferramentas aplicadas ao ensino de programação paralela, mas não apresentam de forma clara uma avaliação estatística dos seus resultados usando métricas quantitativas e qualitativas quando aplicáveis. Este cenário dificulta uma validação e análise mais consistente dos resultados.

Percebe-se que ainda é necessário identificar uma classificação dos tópicos de computação paralela que podem ser antecipados aos alunos ([GHAFOOR; BROWN; ROGERS, 2017](#)) de forma satisfatória em um contexto prático e reproduzível. No presente projeto, buscamos avaliar esta antecipação em diferentes contextos de ensino, alterando sistematicamente fatores como metodologias, formação dos alunos, tamanho das turmas, e modelos de programação paralela.

Destaca-se que a maioria dos trabalhos consideram o currículo definido em [Prasad *et al.* \(2012\)](#) como base para o ensino de programação paralela, integrando os conceitos definidos no documento em seus cursos ou grades curriculares que aprovam seu uso. Dessa forma, fundamentam-se os conceitos definidos em [Prasad *et al.* \(2012\)](#) como base para os tópicos de estudo relacionados ao paralelismo, porém, considerando o contexto de alunos que não tenham conhecimento básico em computação, devem ser selecionados tópicos que sejam compreensíveis pelos mesmos.

DESENVOLVIMENTO DO PROJETO

4.1 Objetivos

O objetivo geral deste projeto de pesquisa é investigar o aprendizado de programação paralela para alunos que não possuam formação prévia em computação, incluindo alunos ingressantes (primeiro ano) em cursos superiores de computação e alunos de outras áreas de conhecimento fora da computação.

Esse objetivo geral constitui-se um grande desafio e uma pesquisa em si: como abordar um assunto complexo e denso como a programação paralela para alunos que não possuem pré-requisitos sólidos de outras matérias essenciais como programação sequencial, sistemas digitais, organização e arquitetura de computadores, sistemas operacionais e redes de computadores.

A antecipação do ensino de programação paralela para alunos do primeiro ano da graduação tem como motivação criar nos mesmos a cultura de se pensar em soluções paralelas como padrão, e não como uma eventual alternativa quando o desempenho não é alcançado. Espera-se que o "**pensar paralelo**" passe a ser algo natural aos futuros profissionais, indo ao encontro das demandas com o uso de plataformas paralelas e distribuídas. Espera-se, também, que futuras soluções computacionais desenvolvidas pelos alunos tenham melhores desempenhos, melhores relações custo x benefício e os códigos paralelos desenvolvidos apresentem um menor custo futuro de manutenção, pois já serão projetados com a visão do paralelismo.

Como subproduto do objetivo geral citado, este trabalho também propõe um estudo inicial sobre quais conteúdos podem ser transmitidos aos alunos sem conhecimentos prévios em computação, de modo que os mesmos estejam aptos a relacionar o paralelismo com o desenvolvimento de soluções computacionais ainda na graduação, em disciplinas como: projeto de hardware, estruturas de dados, banco de dados, inteligência artificial, computação gráfica, processamento de imagem, engenharia de software, entre outras. Devido às características específicas dessa análise do conteúdo, a mesma não é explorada exaustivamente e espera-se que

ela seja aprofundada em estudos posteriores.

Dado este cenário, acredita-se que o ensino da programação paralela deva ser antecipado, gradual e sistematicamente, de modo a formar futuros profissionais da computação com habilidades e competências sólidas em computação de alto desempenho, em especial, programação paralela.

4.1.1 Objetivos específicos

Os objetivos específicos deste projeto são:

- Averiguar se alunos sem formação sólida em computação são capazes de compreender conceitos de paralelismo e desenvolver programas paralelos com qualidade e corretos, com múltiplas tarefas que cooperam para solucionar problemas computacionalmente.
- Determinar a profundidade e a abrangência do conteúdo da programação paralela ministrado nas aulas, de forma que os alunos sejam apresentados a conceitos essenciais em nível de programação, abstraindo dependências diretas com conceitos de outras disciplinas não cursadas.
- Verificar quão receptivos são os alunos em relação a este aprendizado antecipado da programação paralela, visto que esta programação ainda é considerada como um tópico avançado em computação.

4.2 Análise dos conceitos fundamentais de programação paralela

Os conceitos fundamentais de programação paralela são descritos e definidos, essencialmente, em [Prasad *et al.* \(2012\)](#). Na grade curricular da ACM de 2013 ([CURRICULA, 2013](#)) os tópicos de programação paralela estão organizados em uma estrutura disciplinar.

Com o intuito de conhecer o cenário completo dos conceitos de programação paralela, elaboramos uma listagem de todos os tópicos e respectivos conceitos definidos em ambas diretrizes curriculares. A partir dessa listagem foi construída uma unificação de todos os conceitos, criando uma relação fundamentada em tópicos de programação paralela e conceitos de ambas iniciativas curriculares. Essa unificação permitiu a análise dos conceitos estabelecidos por ACM e IEEE, conceitos estes aceitos pela comunidade internacional como fundamentais no ensino de programação paralela.

A partir da análise dessa unificação das grades curriculares e comparação dos tópicos de conhecimento de cada uma, foi possível perceber que na iniciativa da IEEE ([PRASAD *et al.*, 2012](#)) é apresentado um detalhamento mais abrangente e profundo dos conteúdos que devem

ser ensinados, contendo, inclusive, resultados de aprendizagem esperados de cada conceito de paralelismo. Isso deve-se ao fato de que o documento foi elaborado especificamente para as diretrizes curriculares da disciplina de Programação Paralela e Distribuída, enquanto que a iniciativa da ACM propõe diretrizes curriculares para todo o curso de ciência da computação. Examinando as referências, verificou-se que a disciplina de programação paralela em [Curricula \(2013\)](#) originou-se do estudo feito em [Prasad et al. \(2012\)](#) e, por isso, os tópicos estruturados para a disciplina de Programação Paralela e Distribuída na primeira iniciativa curricular estão englobados na segunda. Por estas razões serão considerados os aspectos e conceitos apresentados na iniciativa curricular da IEEE ([PRASAD et al., 2012](#)) como elementares na aprendizagem deste domínio, sendo a base bibliográfica para definição de conteúdo disciplinar.

4.3 Mapas Conceituais

Com a definição dos conceitos fundamentais da área de programação paralela fundamentados na literatura, foram elaborados mapas conceituais com o intuito de organizar e esquematizar os tópicos e sub-tópicos definidos em [Prasad et al. \(2012\)](#), ilustrando as relações entre áreas e conceitos do documento. Os mapas conceituais de cada sub-área representam graficamente todos os conceitos e tópicos de paralelismo descritos pela iniciativa curricular, o que permitiu a análise da estrutura de tais conceitos, a organização e relação do conteúdo de forma visual, facilitando a investigação e apresentação do mesmo. A Figura 8 apresenta o mapa conceitual que demonstra as quatro principais divisões da grade curricular: *Architecture Topics*, *Programming Topics*, *Algorithm Topics* e *Cross Cutting and Advanced Topics*.

Figura 8 – Estrutura dos principais tópicos da grade curricular ([PRASAD et al., 2012](#))

Cada sub-área de conhecimento pode ser expandida, exibindo os sub-tópicos e conceitos relacionados à ela. Dessa forma, pôde-se obter uma visão completa dos conceitos de programação paralela reconhecidos internacionalmente como essenciais para o ensino e elaborar uma estrutura

que representa visualmente cada tópico de ensino com os respectivos conceitos ligados a ela. As Figuras 9, 10, 11 e 12 exibem os mapas conceituais elaborados e permitem a apresentação dos conceitos de forma objetiva e categórica referente às sub-áreas *Architecture topics*, *Programming Topics*, *Algorithm Topics* e *Cross Cutting and Advanced Topics* respectivamente. Todos os mapas conceituais estão disponíveis publicamente acessando a url <http://bit.do/mapas_conceituais>.

Figura 9 – Tópicos referentes à arquitetura (PRASAD *et al.*, 2012)

Figura 10 – Tópicos referentes à programação (PRASAD *et al.*, 2012)

Figura 11 – Tópicos referentes a algoritmos (PRASAD *et al.*, 2012) Figura 12 – Tópicos referentes a tópicos avançados (PRASAD *et al.*, 2012)

4.4 Estruturação em camadas

O ensino de computação em geral é realizado em camadas. Podemos considerar como exemplo o ensino de programação sequencial que normalmente ocorre no primeiro ano dos cursos de computação. Os alunos aprendem a programar computadores sequencialmente usando uma linguagem sem ter conhecimentos prévios de organização de computadores, sistemas opera-

cionais ou complexidade de algoritmos, os quais estão em outras camadas de abstração. Estes alunos abstraem conceitos que não possuem e desenvolvem programas mesmo desconhecendo tais conceitos. Após o primeiro ano e já sabendo programar, os discentes aprendem os assuntos que foram encapsulados em camadas situadas em outros níveis (normalmente abaixo) de um sistema hierárquico de camadas. Ao aprenderem os conteúdos de outros níveis de abstração, compreenderão a motivação para conceitos importantes, os quais fornecerão qualidade e desempenho à programação que estão aprendendo no primeiro ano da graduação. Essa mesma organização do conhecimento é praticada no ensino de arquitetura de computadores, sistemas operacionais e redes de computadores (TANENBAUM; GOODMAN, 1998; TANENBAUM; BOS, 2014; TANENBAUM; WETHERALL, 2010).

Apesar dessa abordagem em camadas já ser aplicada em várias áreas da computação, o mesmo não ocorre com o ensino de computação paralela. Ao organizar os conceitos de computação paralela em camadas, possibilita-se identificar quais assuntos pertinentes à programação paralela podem ser ensinados a alunos sem outros conhecimentos prévios. As camadas hierárquicas permitem separar conteúdos e abstrair conceitos mais complexos em um primeiro momento, antecipando assim o ensino de programação paralela simultaneamente ao ensino de programação sequencial. Esta antecipação permite que os alunos já aprendam a programar computadores em paralelo criando nos futuros profissionais um forte "pensar paralelo" em suas soluções computacionais.

Os autores em Rauber e Rünger (2013) defendem que uma solução sistemática para esta representação é considerar modelos que representem detalhes do sistema de computação usado e forneçam uma visão abstrata para projeto e análise de programas paralelos. Os autores apresentam quatro modelos hierárquicos aplicados à computação paralela de acordo com Heywood e Ranka (1992):

- **Modelo de máquina:** é o nível mais baixo de abstração e consiste na descrição de *hardware* e sistema operacional, como registradores e *buffers* de entrada e saída. Linguagens Assembly são baseadas em modelos deste nível (RAUBER; RÜNGER, 2013).
- **Modelo arquitetural:** os conceitos abordados neste nível de abstração incluem redes de interconexão, organização de memória, processamento síncrono ou assíncrono e modo de execução por arquiteturas SIMD ou MIMD (RAUBER; RÜNGER, 2013).
- **Modelo computacional:** fornece uma concepção abstrata (modelo formal) do modelo arquitetural correspondente. O modelo fornece abstrações que refletem o tempo necessário para executar um programa considerando os recursos de um modelo arquitetural. O modelo computacional provê um método analítico para desenvolver e avaliar algoritmos, cuja complexidade reflete no desempenho em um computador real.
- **Modelo de programação:** trata-se do nível mais alto de abstração segundo Rauber e

Rünger (2013) e refere-se a programas paralelos em termos de semântica da linguagem e ambiente de programação. O modelo de programação paralela deve especificar a visão do programador diante de computadores paralelos, definindo como deve ser o desenvolvimento e codificação de algoritmos paralelos (RAUBER; RÜNGER, 2013). Esta visão é influenciada pelo modelo de arquitetura, linguagem, compilador, bibliotecas de compilação etc. Portanto, existem diversos modelos de programação paralela diferentes para uma mesma arquitetura. Os modelos de programação, de acordo com Rauber e Rünger (2013), podem diferenciar-se segundo alguns critérios:

- O nível de paralelismo que é explorado na execução paralela (nível de instrução, nível procedural ou *loops* paralelos);
- A maneira como os módulos paralelos do programa são especificados;
- O modo de execução de unidades paralelas (SIMD ou SPMD, síncrono ou assíncrono);
- O padrão de comunicação entre as unidades de computação para troca de informações (comunicação explícita ou variáveis compartilhadas);
- Mecanismos de sincronização para organizar o processamento e a comunicação entre unidades paralelas.

O objetivo de um modelo de programação é fornecer mecanismos com os quais o programador possa escrever programas paralelos, especificando processamentos que possam ser realizados em paralelo. De acordo com o modelo de programação, o processamento paralelo pode ser definido em diferentes níveis: (i) uma sequência de instruções que executam operações aritméticas ou lógicas, (ii) uma sequência de diretivas de compilação em que cada diretiva pode incluir várias instruções ou (iii) chamadas de funções ou métodos que geralmente consistem em diversas instruções. Os módulos ou tarefas que serão executados de forma independente e paralela devem ser mapeados nos processadores da plataforma explicitamente pelo programador ou implicitamente por uma biblioteca de tempo de execução.

Como foi demonstrado por meio dos mapas conceituais na Seção 4.3, os conceitos de programação paralela podem estar ligados a outras áreas específicas da computação além da programação, como arquitetura, algoritmos, organização de computadores e sistemas operacionais. Tendo o objetivo de ensinar a programação paralela, a divisão dos tópicos de computação paralela em diferentes níveis deve representar a abstração completa de um sistema computacional, agrupando-os em suas camadas correspondentes.

A partir dos conceitos propostos e organizados por Prasad *et al.* (2012), da estrutura de modelos de abstração apresentados por Rauber e Rünger (2013) e análise do conteúdo estudado em computação paralela, propusemos a possibilidade de classificação dos conceitos de

forma hierárquica em cinco camadas: **Organização e Arquitetura de Computadores**, **Sistema**, **Modelo Computacional**, **Programação** e **Aplicação**. Essa estrutura busca permitir que os conceitos relacionados à computação paralela sejam estudados separadamente em níveis distintos, de forma hierárquica, sistemática e modular, propiciando a observação individual de cada camada e direcionando o estudo para os tópicos de ensino inseridos na mesma.

A Figura 13 exibe a organização geral das camadas. Os conceitos de computação paralela foram mapeados nas camadas de acordo com sua correspondência às abstrações. Vale ressaltar que alguns conceitos podem ser compreendidos a partir de diferentes pontos de vista (por exemplo, uma perspectiva arquitetural ou de programação quanto ao conceito de memória compartilhada ou distribuída) e, por isso, podem estar associados a mais de uma camada.

Figura 13 – Camadas de computação paralela

A seguir são discutidos sinteticamente os tópicos abordados em cada camada, apresentando um panorama dos conceitos englobados em cada uma delas.

Na camada (**Organização e Arquitetura de Computadores**) são compreendidos vários aspectos de *hardware* como recursos de processamento, memória, redes e dispositivos de E/S. Mais especificamente, conceitos de paralelismo relacionados a processadores (taxonomia de Flynn, fluxo de dados, pipelines, processamento heterogêneo, entre outros) e memória (memória compartilhada e distribuída, barramentos, passagem de mensagem, latência, organização de Cache, entre outros) além de outros relacionados ao desempenho e redes de conexão (MIPS/FLOPS, ciclos por instrução, distribuição de dados, entre outros).

Na segunda camada (**Sistema**) os conceitos são relacionados principalmente a sistemas operacionais, compiladores e bibliotecas, que viabilizam a execução da aplicação paralela sobre

a arquitetura, como por exemplo processos e *threads*, *deadlock*, disputa por dados, atomicidade de uma operação, consistência e coerência de dados. Os tópicos da primeira e segunda camada estão relacionados ao sistema computacional em que será executado o programa paralelo. As principais abstrações das camadas são relacionadas ao *hardware*, especificamente ao processador e suas unidades, memória e consistência dos dados, além de bibliotecas e compiladores. À vista disso, os conceitos da primeira e segunda camadas estão relacionados e podem ser unificados como referentes à **plataforma** de execução, originando uma categoria que abrange as duas camadas.

A terceira camada (**Modelo Computacional**) apresenta os conceitos decorrentes da plataforma, proporcionando uma visão estruturada dos mesmos. Os tópicos nesse nível refletem os recursos oferecidos pela plataforma, abstraindo a complexidade dos detalhes dos níveis mais baixos de arquitetura e de sistema. Os conceitos desta camada, como modelos de memória, *speedup* e eficiência, lei de Amdahl, isoeffiência, tempo vs. espaço, escalabilidade, PRAM, não-determinismo, consumo de energia e decomposição de um problema, permitem investigar a influência da plataforma na execução de programas paralelos, abstraindo particularidades dos outros níveis. Além disso, por meio do conhecimento desta camada, é possível avaliar o desempenho de um programa paralelo e analisar custos computacionais, podendo propor melhorias no desempenho do programa com algumas estratégias, como balanceamento de carga e alternativas de escalonamento.

A quarta camada (**Programação**) possui conceitos relacionados à visão do programador quanto ao programa paralelo. Isso envolve desde o planejamento do programa, considerando aspectos da plataforma, até modelos distintos de programação para o desenvolvimento do programa. De uma forma geral, a camada de Programação engloba conceitos como paralelismo de dados e de tarefas, dependências, noções dos paradigmas SPMD (*Single Program Multiple Data*) e MPMD (*Multiple Program Multiple Data*), geração de processos e *threads*, comunicação entre processos, memória compartilhada (linguagens, compiladores e bibliotecas), regiões críticas, sincronização, estratégias de decomposição, operações atômicas, memória distribuída (passagem de mensagem), balanceamento de carga, entre outros. Os conceitos deste nível apontam para o conhecimento e princípios que um programador deve ter em vista no desenvolvimento de *software* paralelo. Percebe-se que o estudo de alguns dos conceitos, por exemplo memória compartilhada ou distribuída, sincronização, escalonamento e mesmo o conceito de *threads*, pode ser mais abrangente em outras camadas, no entanto a de programação visa mostrar os conceitos deste nível de forma suficiente para que o aluno possa desenvolver código paralelo.

Finalmente, a última camada (**Aplicação**) apresenta o uso do paralelismo em soluções de problemas, o que envolve desde exemplos reais que utilizam paralelismo (como previsão do tempo, solução de cálculos complexos na física e aplicações nas áreas de biologia, astronomia e jogos) até soluções de algoritmos simples que possam ser paralelizáveis. Neste nível, onde abstraem-se aspectos inclusive da linguagem de programação, atuam principalmente usuários

finais e gestores de dados e de negócio, preocupados com o uso da aplicação paralela.

A relação integral, contendo todos os conceitos distribuídos em suas respectivas camadas, está disponível publicamente e pode ser acessada na seguinte *url*: <<https://bit.do/camadas>>.

Considerando que os conceitos são oriundos de uma iniciativa curricular (PRASAD *et al.*, 2012), acredita-se que estas camadas podem ser aplicadas na organização do ensino de computação paralela, permitindo focar em diferentes níveis conforme o escopo e objetivo que se deseja alcançar no processo de aprendizagem. Dessa maneira, a divisão em camadas possibilita que o conteúdo seja direcionado de acordo com o tempo disponível e propósito do ensino, permitindo que o conteúdo tenha focos diferentes, como programação e conceitos relacionados à plataforma, por exemplo, ou possa abranger completamente os conceitos das camadas, apresentando o conteúdo de forma integral e com visões dos diferentes níveis de paralelismo.

4.5 Definição do escopo de ensino

A partir do conhecimento do conteúdo de programação paralela esperado que um aluno em nível de graduação compreenda, apresentado na Seção 4.2, foram elencadas, com base na observação das iniciativas curriculares mais atuais da IEEE e ACM (PRASAD *et al.*, 2012; CURRICULA, 2013), as distinções mais relevantes entre programação paralela e sequencial que um aluno essencialmente deve perceber e compreender na graduação, enumerados abaixo.

1. Criação de processos;
2. Distribuição de carga de trabalho;
3. Ferramentas de programação;
4. Identificação de paralelismo;
5. Interação (comunicação e sincronização);
6. Impacto da plataforma;
7. Teste e depuração; e
8. Visão multidimensional.

Deve-se considerar que, visando ensinar alunos sem conhecimentos dos conceitos fundamentais da computação, os assuntos elencados devem ser ponderados para que sejam compreensíveis. Dessa maneira, sugerimos classificações desses tópicos em níveis que representem a relevância da abordagem do tema em sala de aula como essenciais, importantes ou desconsiderados, considerando o objetivo de que sejam aplicados especialmente para o ensino de alunos

sem formação prévia em computação. Nessa perspectiva, a Figura 14 apresenta a classificação de cada tópico referente às diferenças de programação paralela e sequencial de acordo com os níveis de relevância do ensino.

Figura 14 – Classificação dos tópicos referentes às diferenças de programação paralela e sequencial considerando o contexto de alunos do primeiro ano.

<ul style="list-style-type: none">++ Criação de processos (iniciar e finalizar);++ Ferramentas de programação;++ Interação (comunicação e sincronização);+ Distribuição de carga de trabalho;+ Identificação de paralelismo;+ Visão multidimensional;- Impacto da plataforma; e- Teste e depuração.	<ul style="list-style-type: none">++ Essencial+ Importante- Desconsiderado
--	--

Com base nestes tópicos que apontam para temas de programação paralela abordados no processo de ensino-aprendizagem, almeja-se que os alunos em cursos de computação tenham conhecimentos introdutórios em tópicos essenciais e importantes desde o princípio da graduação. Esses assuntos podem ser aprofundados posteriormente na graduação, quando outros tópicos vierem a ser ensinados, como arquiteturas paralelas, complexidade de algoritmos, e engenharia de software. Acredita-se que estes tópicos iniciais também podem ser ensinados para estudantes de outras áreas além da computação que possuam interesse em programação paralela. Neste caso, por não cursarem computação, teriam uma visão mais superficial do desenvolvimento de tais sistemas, fato que já ocorre com a programação sequencial.

A partir dos estudos realizados e tendo em vista o ensino de indivíduos no contexto descrito, acreditamos que os conceitos ensinados para estes alunos possam ter foco na camada de **Programação**, buscando apresentar uma visão **algorítmica e prática** de programação paralela. A camada de Programação permite antecipar a apresentação de conceitos que serão vistos nos próximos anos com enfoque no desenvolvimento de código, permitindo que o aluno possa conhecer aspectos de paralelismo e desenvolver aplicações paralelas antes mesmo de conhecer detalhes das outras camadas, proporcionando, assim, o contato com programação paralela e superando as limitações da ausência de conhecimento básico prévio em computação. Desta forma, não advogamos que a teoria de programação paralela é desnecessária, visto que é primordial para o entendimento mais profundo e abrangente do assunto, no entanto deseja-se inverter a ordem tradicional de ensino dos conteúdos de teoria e prática.

ESTUDOS EXPERIMENTAIS

5.1 Considerações iniciais

Este capítulo descreve cinco experimentos elaborados para avaliar o ensino de programação paralela para alunos sem conhecimento prévio básico de computação e com foco na camada de programação, definida anteriormente na Seção 4.4, buscando um ensino algorítmico e prático dos conceitos de paralelismo. Além disso, procurando validar a proposta em diferentes cenários, comparamos o aprendizado de alunos quando estes são expostos aos tópicos de paralelismo com diferentes abordagens e contextos de ensino. As abordagens escolhidas para os experimentos foram, em sua maioria, aquelas frequentemente encontradas no mapeamento sistemático realizado neste trabalho.

Primeiramente, são listados os experimentos, os quais são descritos de forma geral. Nas seções subsequentes são descritos, detalhadamente, os processos de elaboração, planejamento e os resultados provenientes da execução de cada curso, assim como uma análise precisa de cada aspecto destes resultados.

Por fim, os resultados de todos os experimentos são comparados e analisados criticamente com o intuito de confrontar e, portanto, validar a hipótese e os objetivos, conforme propostos inicialmente neste trabalho, pontuando as principais contribuições do mesmo.

5.2 Relação de experimentos

Esta Seção apresenta os experimentos realizados, os quais são descritos com base na estrutura descrita em (WOHLIN *et al.*, 2012). A estrutura seguida visa permitir reprodutibilidade do experimento e garantir a consistência científica dos resultados. Os experimentos analisam o processo de ensino/aprendizagem pelo desempenho dos alunos nos cursos e pela satisfação ao aprender o conteúdo de programação paralela. No decorrer deste capítulo, os experimentos

detalham a definição do estudo, planejamento, operações realizadas, descrição dos resultados e, por fim, a análise destes resultados.

A execução destes experimentos contou com o auxílio de alunos monitores para as aulas, Guilherme Martins e Ana Spengler, que colaboraram na preparação das aulas e prestaram suporte aos alunos quando necessário. Além disso, Paulo Sérgio Lopes de Souza dispôs-se como professor responsável pelos cursos, participando ativamente da elaboração das atividades e definição de conteúdo, o que permitiu a execução destes experimentos.

Com o intuito de facilitar a comparação dos resultados entre os experimentos, foram unificadas as métricas de avaliação dos estudantes para os cinco experimentos, em relação ao desempenho dos alunos no processo de aprendizagem. Neste quesito, definiram-se as seguintes métricas de avaliação:

- **Uso correto de paralelismo:** 40%;
- **Uso correto do modelo de programação:** 30%;
- **Saída correta/esperada:** 20%;
- **Legibilidade e boas práticas de programação.** 10%.

Em vista dos objetivos de ensino, o peso das métricas foi distribuído dessa maneira, pois desejamos que as notas dos códigos desenvolvidos representem principalmente o quão corretos estão em função do uso do paralelismo e do modelo de programação, sendo a saída correta e legibilidade aspectos integrantes da nota, no entanto não suas maiores composições.

A análise estatística feita nesse trabalho foi baseada em [Armitage, Berry e Matthews \(2008\)](#).

Os experimentos estão organizados em ordem cronológica e são listados a seguir:

- Experimento I: Disciplinas de Programação Concorrente baseadas em Aprendizagem Baseada em Equipes (ABE) (2018);
- Experimento II: Curso de Difusão Cultural de Introdução ao *OpenMP* (2018);
- Experimento III: Curso de Difusão Cultural de Introdução a *PThreads* (2018);
- Experimento IV: Curso de Difusão Cultural de Introdução ao *OpenMP* para alunos iniciantes em computação (2019); e
- Experimento V: Curso de Difusão Cultural de Introdução ao *OpenMP* baseado em *Problem Based Learning* (2019).

5.3 Experimento I: Disciplinas de Programação Concorrente baseadas em Aprendizagem Baseada em Equipes (ABE)

Este experimento foi composto de 141 alunos, que participaram das aulas realizando um curso abrangente de programação concorrente, considerando: aspectos conceituais e práticos de programação paralela em pthreads, OpenMP, MPI, CUDA e Erlang. Também foram abordados temas como avaliação de desempenho, arquiteturas paralelas e teste de programas concorrentes. Apesar desta abrangência, relatamos apenas os resultados do ensino de OpenMP neste primeiro experimento, visto que o OpenMP foi utilizado nos outros experimentos.

5.3.1 Definição do experimento

Objetivo

Determinar o desempenho dos alunos em relação à qualidade do código desenvolvido e aplicação dos conceitos de paralelismo considerando o conteúdo de *OpenMP*, afim de comparar tais resultados com os outros experimentos, cujos estudantes não tinham conhecimento prévio ou fundamentação teórica similar à amostra deste estudo.

Objeto

Iniciativa de ensino nas disciplinas de programação concorrente baseada na Aprendizagem Baseada em Equipes (ou *Team Beased Learning* - TBL), divididos em três turmas, ministrada no ICMC USP, no segundo semestre de 2018.

Foco qualitativo

Avaliar a capacidade dos estudantes em desenvolver de códigos paralelos em *OpenMP* corretos e com qualidade.

Perspectiva

Prática de uma metodologia de ensino ativa e colaborativa, associada à recursos educacionais que apoiam sua execução e que visa o desenvolvimento de atividades em grupos, a perspectiva é que este experimento seja conduzido em ambiente de sala de aula teórica com recursos e espaço físicos suficientes para as atividades colaborativas previstas.

Contexto

Realizado entre agosto a dezembro, em função do calendário letivo do segundo semestre de 2018, no contexto da disciplina obrigatória de Programação Concorrente, oferecida aos cursos de Engenharia da Computação (SSC0742), Bacharelado em Ciências de Computação (SSC0143) do ICMC USP nos 7º e 6º período da graduação, respectivamente, e, como disciplina eletiva, para a pós-graduação (SSC5795).

5.3.2 Planejamento

Definição das Questões de Pesquisa

Considerando os objetivos estabelecidos para este projeto de mestrado, definiram-se as seguintes questões de pesquisa:

QP1 Os estudantes com formação prévia relevante em computação foram capazes de compreender o conteúdo teórico de *OpenMP*?

QP2 Os estudantes foram capazes de desenvolver aplicações paralelas usando *OpenMP* corretamente e com qualidade?

QP3 Os estudantes ficaram motivados com a forma como o conteúdo foi abordado?

Seleção de Sujeitos

A população neste estudo foi constituída de discentes regularmente matriculados na disciplina de programação concorrente distribuídos em 3 turmas, compostos conforme ilustrado na Tabela 14. Todos os estudantes matriculados nas turmas da disciplina já haviam cursado Sistemas Operacionais, Arquitetura de Computadores e outras disciplinas consideradas fundamentais para o aprendizado de programação paralela. As grades dos cursos de Bacharelado em Ciências de Computação e Engenharia da computação estão disponíveis nos links <http://bit.do/grade_bcc_2018> e <http://bit.do/grade_ec_2018>, respectivamente.

Tabela 14 – Configuração das turmas: Experimento I

Curso	Quantidade de Alunos
Engenharia da Computação	34
Bacharelado em Ciência de Computação	112
Pós-Graduação	5

Descrição da Instrumentação

O aprendizado colaborativo com TBL foi a base para a metodologia empregada neste experimento (PARMELEE *et al.*, 2012). As aulas possuíam as seguintes etapas: estudo prévio, avaliação individual, avaliação em equipe e aplicação prática (desafio de programação).

O estudo prévio foi realizado individualmente pelos alunos, antes do início da aula presencial. A etapa de avaliação individual foi a aplicação de teste com 10 questões de múltipla escolha e 4 alternativas cada. Na etapa de avaliação em equipe os alunos foram reunidos em grupos que puderam discutir sobre as respostas que cada um forneceu para as questões na etapa de avaliação individual. Após a discussão, cada grupo escolheu as respostas corretas do grupo e as enviou novamente ao professor. Na etapa de aplicação prática, um desafio de programação

foi passado aos alunos. As submissões das respostas ao desafio foram feitas pelo *software* de maratona de programação *PC²* que auxiliou nas correções automáticas dos programas e forneceu *feedback* instantâneo se as saídas esperadas dos códigos submetidos estavam corretas.

Foi elaborado também um questionário qualitativo para avaliar a satisfação dos alunos relativa a metodologia, materiais e conteúdo ministrado no curso. O questionário está disponível no Apêndice A.

Planejamento de Curso

O curso de OpenMP foi planejado para ser abrangente e incluiu aspectos históricos do OpenMP, principais diretivas (`parallel`, `for`, `sections`, entre outras), cláusulas (`num_threads`, `shared`, `private`, `firstprivate`, `lastprivate`, `nowait`, entre outras), funções OpenMP e variáveis de ambiente.

5.3.3 Execução

A preparação das aulas TBL seguiu o planejamento já descrito (PARMELEE *et al.*, 2012). Os desafios foram elaborados a partir de exercícios práticos de codificação com OpenMP, os quais continham o enunciado de cada atividade, códigos sequenciais para auxiliar a compreensão e a implementação paralela pelos alunos. Os testes de múltipla escolha, usados na etapa de garantia de preparo individual e de equipe também foram elaborados previamente à aula.

Os experimentos foram realizados dentro do período de aula com uma duração de 2:40 horas. Seguindo as etapas do TBL, o tempo de aula foi dividido da seguinte forma:

- Etapa 1: 20 min;
- Etapa 2: 30 min;
- Etapa 3: 20 min;
- Etapa 4: 60 min.

5.3.4 Apresentação dos resultados

Em um primeiro momento, apresentamos os resultados da primeira etapa do TBL, com 10 questões teóricas referentes ao aprendizado teórico do tema ministrado.

Cada questão do teste era valorada em 4 pontos. Este modelo de pontuação da questão permite aos alunos distribuírem a pontuação às 04 alternativas disponíveis, i.e., mais de uma alternativa pode receber de 01 a 04 pontos, conforme a certeza que o estudante tem para a alternativa correta na questão. Se distribuir os 04 pontos em mais alternativas, pois está inseguro qual é a correta, ele poderá receber um valor menor de acordo com a sua certeza da alternativa correta. Para a nota final nesta etapa, o total de pontos atribuídos às alternativas corretas era

somado e, de acordo com a pontuação máxima total possível, calculava-se o aproveitamento nesta fase. Esta atividade correspondia a 40% da nota final do aluno.

A Figura 15 representa a distribuição de aproveitamento obtido pelos participantes na primeira etapa do TBL.

Figura 15 – Distribuição de aproveitamento na etapa de garantia de preparo individual

A segunda etapa do TBL, garantia de preparo em equipe, representa o conhecimento teórico do aluno após a discussão das questões em grupo. A porcentagem da nota nesta atividade foi de 18% da composição da nota final do aluno. A Figura 16 mostra a distribuição das médias das pontuações obtidas pelos participantes na segunda etapa do TBL. A Figura 17 ilustra a evolução da pontuação média dos alunos nas duas primeiras etapas da aula.

Figura 16 – Histograma das pontuações obtidas na etapa de garantia de preparo em equipe

Figura 17 – Pontuações médias por atividade

Pode-se perceber um aproveitamento de 75,4% da turma na etapa de garantia de preparo individual (AI) e de 97,2% na etapa de garantia de preparo em equipe (AE), o que significa uma melhora satisfatória no desempenho teórico após discutir as questões com outros estudantes.

Na última etapa do TBL os alunos implementaram a atividade prática (desafio de programação paralela), valorada em 10 pontos, considerando as métricas definidas na Seção 5.2. Essa nota correspondia a 42% do valor final do aluno.

A Figura 18 representa as médias de qualidade de código desenvolvido na atividade prática separadas por turma e a Figura 19 a distribuição unificada das notas, representando o aproveitamento na atividade prática pela amostra de alunos total. Essas Figuras serão analisadas na Seção 5.3.5.

Figura 18 – Qualidade de código média por turma

Figura 19 – Qualidade de código média por atividade prática

Avaliação qualitativa

Após a conclusão de todas as aulas planejadas, um questionário *online* foi divulgado, por meio do sistema *Moodle*, para a avaliação qualitativa por parte dos alunos, no que se refere à disciplina. As respostas ao formulário foram feitas de forma anônima e optativa. O formulário está disponível no Anexo A.

Foram obtidas 100 respostas ao formulário, i.e., 66% dos participantes. As respostas encontram-se em http://bit.do/exp1_qualitativo.

A seguir apresenta-se as respostas para as principais questões do teste qualitativo. A análise das respostas obtidas será feita em um segundo momento.

1) O planejamento do curso foi claro?

Figura 20 – Avaliação qualitativa - questão 1

3) Foi possível compreender como funciona TBL (Team Based Learning)?

Figura 21 – Avaliação qualitativa - questão 3

5) Qual método pedagógico você acredita ser melhor?

Figura 22 – Avaliação qualitativa - questão 5

8) Você gostou das aulas com TBL?

Figura 23 – Avaliação qualitativa - questão 8

10) O que você achou das Avaliações Contínuas (ACs)?

Figura 24 – Avaliação qualitativa - questão 10

11) Qual das etapas do TBL você acredita que mais auxilia no aprendizado?

Figura 25 – Avaliação qualitativa - questão 11

12) Você preferiria que a disciplina voltasse ao método tradicional?

Figura 26 – Avaliação qualitativa - questão 12

5.3.5 Análise dos resultados

QP1: Os alunos com formação prévia relevante em computação foram capazes de compreender o conteúdo teórico de *OpenMP*?

O conhecimento teórico foi avaliado em dois momentos na metodologia: na primeira e na segunda etapa, garantia de preparo individual (AI) e garantia de preparo em equipe (AE), respectivamente.

Foi possível notar, através das Figuras 15 e 16, que os estudantes apresentaram a maior parte das notas positivas em ambas as etapas citadas.

A Figura 17 demonstra que houve grande evolução da AI para AE, caracterizando um avanço significativo de 75,4% para 97,2% no conhecimento teórico.

Desta forma, afirma-se que os estudantes foram capazes de compreender conteúdo teórico ministrado neste experimento.

QP2: Os estudantes nesse contexto foram capazes de desenvolver aplicações paralelas usando *OpenMP* corretamente e com qualidade?

Os resultados de codificação exibidos na Figura 18 nos mostram que, nas quatro turmas analisadas, o padrão de desenvolvimento manteve-se elevado. Podemos observar que a segunda turma teve sua média prejudicada devido a alguns *outliers* muito baixos, no entanto a mediana nos mostra que pelo menos 50% dos valores da amostra são iguais ou superiores a 90% de aproveitamento. Segundo demonstrado na Figura 19, a média considerando os valores da média dos estudantes de todas as turmas é 82,1%, sendo que a mediana demonstra que mais da metade das notas foram superiores a 87,0% de aproveitamento.

Esses resultados nos permitem afirmar que, em relação à capacidade de aplicação do

paralelismo, uso do modelo de programação, saída correta e legibilidade de código, os alunos puderam desenvolver aplicações paralelas corretas, com qualidade e utilizar apropriadamente os recursos do modelo de paralelismo.

QP3: Os alunos sentiram-se confortáveis e motivados a aprender os conceitos propostos?

Como foi possível perceber, por meio das Figuras 20 a 26, os alunos se sentiram mais satisfeitos com a disciplina ministrada com *Team Based Learning*, do que em formato tradicional. Além disso, os alunos se sentiram motivados e aprovaram o aprendizado com o uso de TBL, avaliações contínuas e desafios de programação.

5.4 Experimento II: Curso de Difusão Cultural de Introdução ao *OpenMP*

A principal motivação para este experimento foi analisar se alunos sem formação preliminar em computação, como em sistemas operacionais, redes de computadores, organização de computadores e, até mesmo, sem formação sólida em programação sequencial, poderiam aprender programação paralela. O experimento expôs os alunos ao aprendizado de conceitos sobre paralelismo, com foco em programação, possibilitando a execução paralela e concorrente de *threads* para a solução de problemas. A turma, neste experimento, estava composta de alunos sem e com formação prévia em computação, permitindo, desta forma, verificar a diferença no desempenho e aprendizado entre ambos durante o curso. As seções a seguir descrevem o experimento detalhadamente com base nas etapas do processo experimental definido em (WOHLIN *et al.*, 2012).

5.4.1 Definição do experimento

Objetivo

O objetivo deste estudo experimental é analisar se alunos sem formação básica em computação, incluindo alunos no primeiro ano da graduação, são capazes de compreender e aplicar corretamente os conceitos de paralelismo. Observamos neste experimento o quão corretos estavam os algoritmos paralelos desenvolvidos e a aplicação prática de fundamentos da programação paralela. Durante o experimento nós verificamos se os estudantes sem base conceitual prévia foram capazes de compreender e aplicar programação paralela da mesma maneira que alunos que possuíam formação prévia em computação.

Objeto

O objeto de estudo deste experimento é a implementação de um curso de programação paralela voltado a alunos sem formação básica em computação. Essa implementação segue

as iniciativas da ACM ([CURRICULA, 2013](#)) e IEEE ([PRASAD et al., 2012](#)) selecionando os tópicos inseridos nas camadas de conceitos definidas na Seção 4.4, especificamente a de Programação.

Foco Qualitativo

O foco qualitativo deste experimento é medir o desempenho obtido por alunos de diferentes contextos ao serem expostos ao ensino da programação paralela e seus paradigmas. Além disso, mensurar qual o impacto na satisfação com o curso e o aprendizado por parte dos alunos, mediante suas experiências.

Perspectiva

Por se tratar de um estudo experimental com processo de ensino-aprendizagem específico e, baseando-se no modelo de ensino tradicional, a perspectiva do experimento considerou o ambiente real de sala de aula. Ainda, por se tratar de um curso voltado à prática de programação, a perspectiva é de que o curso seja conduzido em um laboratório de computação.

Contexto

Mediante a perspectiva de aulas práticas em laboratório, o experimento foi realizado em novembro de 2018 no Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo no contexto de um laboratório de programação, como curso de difusão cultural, cujas vagas foram abertas à comunidade. O curso teve duração total de 8 horas, divididas entre 4 tópicos de 2 horas cada. O único requisito determinado para o curso era já possuir algum conhecimento básico em programação sequencial na linguagem C. De fato, os alunos não precisariam ter conhecimentos sólidos em programação C, porém, deveriam saber como compilar, executar e utilizar estruturas de programação simples em C.

5.4.2 Planejamento

Definição das questões de pesquisa

Em vista dos objetivos deste experimento e do projeto, foram elencadas as seguintes questões de pesquisa:

QP1: Os alunos sem formação básica prévia foram capazes de compreender e aplicar de forma prática os tópicos de ensino tanto quanto os alunos com formação básica prévia?

QP2: Os códigos desenvolvidos pelos alunos demonstram aplicação dos conceitos de paralelismo?

QP3: Os alunos foram capazes de usar corretamente o modelo de programação (OpenMP)?

QP4: Os códigos desenvolvidos pelos alunos funcionavam/rodavam corretamente?

QP5: Os materiais e conteúdos abordados no curso foram suficientes, insuficientes ou excessivos?

QP6: Qual o sentimento dos alunos em relação ao curso? Os alunos demonstraram interesse no tema ensinado?

QP7: Os alunos sem formação prévia sentiram que puderam aprender programação paralela?

Seleção de sujeitos

Os alunos participantes do experimento foram provenientes de uma amostra inteiramente aleatória devido a matrícula ser aberta à comunidade. O curso não possuía caráter obrigatório e, devido ao único requisito ser o de conhecimento básico em linguagem C, os alunos matriculados apresentavam diferentes características, como nível de profundidade em programação, escolaridade e área de conhecimento.

Descrição de instrumentação

A instrumentação utilizada durante o curso foi: apresentação de slides, formulários online para submissão dos exercícios e o teste qualitativo.

Os exercícios aplicados durante o curso foram:

Ex1 - Soma de um intervalo de números de 0 a N

Ex2 - Soma de um intervalo de números de 0 a N com Reduction

Ex3 - Pi paralelo com redução (opcional)

Ex4 - Soma de um intervalo de números de 0 a N com Reduction e paralelismo de estruturas de repetição com a primitiva for

Ex5 - Pi paralelo com redução e loop (opcional)

Ex6 - Soma de um intervalo de números de 0 a N usando Sections

5.4.3 Execução

Preparação

Após a definição do experimento e respectivo planejamento, foi submetido o pedido para realização do curso de difusão cultural no ICMC com carga horária e tema propostos nesse experimento. O pedido foi aprovado e, então, reservou-se o laboratório onde ocorreria o curso. A proposta de curso visava os seguintes objetivos:

Objetivos do curso:

- Estimular o pensar em soluções paralelas: familiarização do estudante com a execução paralela e concorrente de *threads* para otimizar o desempenho de soluções computacionais;

- Compreensão do modelo de programação OpenMP e suas implicações;
- Aplicação das primitivas do OpenMP para redução do tempo de execução em algoritmos “paralelizáveis” de maneira prática; e
- Capacitação na criação e sincronização de *threads*.

As apresentações de *slides* para cada aula foram preparadas cuidadosamente buscando obter clareza, sequência e boa didática para a transmissão do conteúdo. Ainda, foram preparadas as descrições e soluções codificadas de cada exercício que seria proposto durante as aulas, além dos formulários online os quais seriam utilizados pelos alunos para submissão dos exercícios implementados em cada aula.

Operação

O quadro 2 apresenta uma visão geral dos tópicos abordados durante as aulas.

5.4.4 Apresentação dos resultados

Caracterização dos participantes

Considerando a aleatoriedade da amostra devido ao caráter do curso com vagas abertas à comunidade, a população de estudantes para o experimento foi altamente heterogênea, abrangendo alunos de áreas de conhecimento diversificadas, tais como engenharia civil, engenharia elétrica, estatística, matemática e cursos de computação como engenharia da computação, sistemas de informação e bacharelado em computação. Além disso, o grupo de alunos destaca-se pelos diferentes níveis de formação, contemplando desde alunos no primeiro ano de graduação até alunos de doutorado ou formados.

A Figura 27 exibe a quantidade de alunos por formação e período. Destaca-se nesta Figura a categorização, de acordo com o que indica a formação acadêmica, em dois grupos: sem conhecimento prévio consolidado dos fundamentos da computação (destacados em laranja) e com conhecimento prévio (destacados em azul). Alunos de computação no segundo período (primeiro ano) foram classificados como sem conhecimento prévio em função das disciplinas ainda não cursadas pelos mesmos.

A Figura 28 apresenta a quantidade total de alunos nas duas categorias. Destaca-se a maior porcentagem no grupo sem conhecimento prévio, indicando que o tema é atrativo para os alunos neste contexto, sejam no primeiro ano da graduação ou de outras áreas além da computação.

Aprendizado Técnico

As avaliações do conhecimento técnico adquirido foram feitas de forma gradual no decorrer do curso, por meio das soluções dos exercícios desenvolvidas individualmente. As soluções solicitadas deveriam empregar as técnicas de programação paralela apresentadas no

Quadro 2 – Planejamento de curso

Tópico	Conceitos
1- Introdução OpenMP	Motivação Computação sequencial x paralela Exemplos de soluções paralelas do dia a dia. Exemplos de paralelismo na computação. O que são <i>Threads</i> ? Regiões paralelas Dados Compartilhados e Locais <i>OpenMP</i> (O que é, Como funciona e Diretivas de compilação).
2- Tipos de variáveis e estruturas	Rotinas do OpenMP Espaço compartilhado vs privado Atributos (<i>Private, shared, default, firstprivate, lastprivate, nowait</i>). Cláusulas que podem ser aplicadas (<i>private, shared, firstprivate, lastprivate, reduction, schedule, nowait and ordered</i>) Criação e monitoramento de <i>threads</i> <i>#pragma omp parallel reduction</i> <i>Speedup & Eficiência</i>
3- Paralelismo de dados usando laços de repetição	Paralelismo de dados usando <i>#pragma omp for [clause list]</i> Aplicação de diferentes tipos de variáveis: <i>private, firstprivate, lastprivate, reduction, schedule, nowait and ordered</i> Aplicação de diferentes tipos de escalonamento: <i>static, dynamic, runtime and guided</i> Mostrar o impacto de diferentes tipos de escalonamento no desempenho.
4 - Paralelismo de tarefas ou funcional	Diretiva <i>sections</i> Diretiva <i>barrier</i> Exclusão Mútua Avaliação sobre conteúdo aprendido e formato das aulas

módulo correspondente. Dessa forma, cada avaliação corresponde a aplicação de diretivas ou conceitos diferentes do modelo de programação estudado no módulo em questão.

As Tabelas 15 e 16 apresentam as médias de aproveitamento para cada métrica de avaliação em cada atividade prática realizada em sala de aula durante o curso, contemplando, respectivamente, os grupos de alunos sem conhecimento prévio e com conhecimento prévio em conteúdos de ciência da computação. Além do aproveitamento por métrica de avaliação, exibem a quantidade de submissões de cada grupo para a mesma atividade.

A Figura 29 apresenta um diagrama de caixas que demonstra a dispersão das médias apresentadas nas Tabelas 15 e 16. Os intervalos das caixas indicam a distribuição das médias das

Figura 27 – Alunos por curso e período - OpenMP/2018

Contagem de aluno por curso e período

Figura 28 – Classificação de alunos de acordo com formação prévia - OpenMP/2018

Alunos por conhecimento prévio em computação

atividades em todo o curso. Foram apresentadas conjuntamente, com o intuito de compará-las, as representações da dispersão de aproveitamento dos alunos sem conhecimento prévio e sem conhecimento prévio.

Analisando os resultados, percebe-se que a distribuição dos valores está acima de 85%

Tabela 15 – Médias das métricas considerando a amostra de alunos sem conhecimento prévio | OpenMP 2018

Alunos sem conhecimento prévio				
	Paralelismo + legibilidade	Uso do modelo de programação	Saída correta	Quantidade de submissões
Ex1	88.7%	89.0%	78.0%	15
Ex2	91.5%	94.6%	88.5%	13
Ex3	96.7%	93.3%	86.7%	9
Ex4	100.0%	97.7%	100.0%	11
Ex5	87.5%	94.5%	57.5%	4
Ex6	90.8%	89.6%	84.2%	12
Média	92.5%	93.1%	82.5%	Total: 64

Tabela 16 – Médias das métricas considerando a amostra de alunos com conhecimento prévio | OpenMP 2018

Alunos com conhecimento prévio				
	Paralelismo + legibilidade	Uso do modelo de programação	Saída correta	Quantidade de submissões
Ex1	95.0%	91.4%	68.6%	7
Ex2	100.0%	94.3%	100.0%	7
Ex3	96.7%	86.7%	100.0%	3
Ex4	100.0%	97.1%	100.0%	7
Ex5	90.0%	100.0%	100.0%	1
Ex6	94.3%	90.0%	85.7%	7
Média	96.3%	93.9%	93.7%	Total: 32

de aproveitamento em todas as métricas, exceto a saída correta/esperada. Isso justifica-se por erros relacionados à programação sequencial ou falta de atenção em detalhes no código que não resultaram na saída esperada dada a pouca experiência com programação, sendo a minoria dos casos falha de execução ou saída que representasse que a solução está incorreta. Ainda assim, o valor da mediana indica que, no grupo sem conhecimento prévio, em 50% das atividades foi obtido um aproveitamento equivalente ou superior a 85,5% e de 100% no grupo com conhecimento prévio. Considerando as médias, todas apresentam valores valores de aproveitamento elevado.

Comparando as amostras de alunos sem conhecimento prévio e com conhecimento prévio, percebe-se que a maior parte das notas estão na mesma região, caracterizando um bom desempenho para ambos os grupos.

Finalmente, a Figura 30 exibe, por meio de um diagrama de caixas, a dispersão das médias de aproveitamento dos alunos ao final do curso. A porcentagem de aproveitamento total representa a média final que o aluno obteve considerando a avaliação do aproveitamento de todos os exercícios realizados no curso.

Figura 29 – Comparação da distribuição de notas de cada métrica de avaliação - OpenMP/2018

Figura 30 – Comparação das porcentagens de aproveitamento do curso - OpenMP/2018

Analisando a dispersão dos percentuais de aproveitamento total dos alunos no curso, percebe-se que, apesar das médias dos alunos sem formação prévia estarem abaixo, o valor

das médias de ambos os grupos são muito próximos e superiores a 90% de aproveitamento, o que avalia-se como um aproveitamento excepcional. Como é possível verificar na Tabela 15 e na Tabela 16, houve um número maior de submissões das atividades por parte de alunos sem formação prévia, o que aumentou a amostra deste grupo, ampliando a dispersão das médias. Por isso, pode-se dizer que os valores são proporcionalmente equivalentes em razão do tamanho das amostras.

Foi realizado o teste de Wilcoxon para amostras independentes com o intuito de investigar se as amostras são estatisticamente equivalentes ou se possuem diferença significativa. Para isso, foram incluídas no teste todas as notas de exercícios realizados e definidas as seguintes hipóteses, onde μ_1 representa a amostra de alunos sem conhecimento prévio e μ_2 , com conhecimento prévio:

Hipótese Nula: Não há diferença estatisticamente significativa no aproveitamento total dos experimentos realizados entre os grupos de alunos com e sem conhecimento.

$$H_0 : \mu_1 - \mu_2 = 0$$

Hipótese Alternativa: Há diferença estatisticamente significativa no aproveitamento total dos experimentos realizados entre os grupos de alunos com e sem conhecimento.

$$H_1 : \mu_1 - \mu_2 \neq 0$$

A partir dessas hipóteses, foi obtido o resultado exibido na Tabela 17.

Tabela 17 – Teste de Wilcoxon para amostras independentes

Estatística do Teste (Wilcoxon)	
Informações	Valores
Estatística	79,5
P-valor	0,5204
Hipótese Nula	0
Limite Inferior	-0,399953685
(Pseudo) Mediana	0,199985793
Limite Superior	1,199996688
Nível de Confiança	0,95

Como o P-valor é igual a 52,04%, aceitamos a hipótese nula a uma significância de 95%. Assim, existem evidências suficientes para aceitar a hipótese nula e afirmar que não há diferença estatística significativa no aproveitamento total dos experimentos realizados entre os grupos de alunos com e sem conhecimento prévio.

Avaliação qualitativa

Ao final do curso, os alunos presentes responderam, de forma anônima e opcional, um formulário contendo questões com o intuito de avaliar a satisfação dos alunos relativa a organização, materiais e conteúdo ministrado no curso e, além disso, conhecer o contexto

técnico prévio dos estudantes, assim como o grau de domínio de programação sequencial. As respostas desse formulário permitiram observar a avaliação dos participantes sobre o próprio conhecimento obtido durante o curso e se o material foi eficaz e suficiente para suprir o processo de aprendizagem. As respostas de cada questão davam-se pela escala Likert, ou seja, opções com valores de 1 a 5, em que 1 representa a pior nota e 5 a melhor nota.

Abaixo são listadas as questões do formulário qualitativo.

- Q1:** Como você classifica seu conhecimento em programação sequencial (como a C, por exemplo) antes da realização deste curso?
- Q2:** Como você classifica seu conhecimento em programação paralela antes da realização deste curso?
- Q3:** Você considera importante aprender programação paralela, desde o começo do curso de computação?
- Q4:** Como você classifica seu conhecimento em programação paralela após a realização deste curso?
- Q5:** Você considera que este curso o estimulou a pensar em paralelo na construção de algoritmos?
- Q6:** Você acha que os exemplos/exercícios propostos foram suficientes para uma introdução ao OpenMP?
- Q7:** Como você classifica seu conhecimento em OpenMP após a realização deste curso?
- Q8:** O curso contribui para você entender a programação de *threads* para máquinas paralelas?
- Q9:** De uma forma geral, como você avalia o conteúdo ministrado neste curso?
- Q10:** Você acha que o tempo/duração do curso foi suficiente para abordar o conteúdo proposto?

A Figura 31 demonstra a média das respostas para cada questão do formulário, na mesma sequência listada anteriormente, comparando as respostas dos alunos sem conhecimento prévio e com conhecimento prévio. As questões são identificadas por palavras-chaves referentes ao seu conteúdo.

Observamos na Figura 31 que, por meio das respostas da questão 1, os alunos julgavam possuir algum conhecimento em linguagem de programação procedural, no entanto, por meio da média das respostas da questão 2, podemos afirmar que possuíam um conhecimento de programação paralela relativamente baixo antes do curso em ambos os grupos. A média da questão 4 nos mostra que, ao término do curso, os estudantes, tanto com conhecimento prévio quanto sem conhecimento prévio, sentiam que houve uma evolução considerável no conhecimento de

Figura 31 – Resultado avaliação qualitativa - OpenMP/2018

programação paralela. No entanto, os resultados da questão 7 demonstram que uma porção dos participantes não sentiram-se confiantes no domínio do modelo de programação. Analisando comentários feitos pelos alunos, isso deve-se ao tempo do curso não ter possibilitado uma maior dedicação e profundidade dos tópicos abordados.

Além das questões mencionadas acima, o questionário disponibilizava uma área de texto para o participante descrever comentários, críticas ou sugestões. Destacamos, na lista a seguir, as críticas positivas e negativas recebidas de maior relevância.

- "Recebi uma introdução e base boa para programação paralela. O professor e os monitores foram prestativos e bons."
- "Mais cursos assim aqui no ICMC."
- "Não tinha conhecimento nenhum sobre programação paralela e openMP, agora tenho noção sobre em que contextos implementar e da eficiência proporcionada."

- “O conteúdo de Openmp é muito amplo. Para um minicurso introdutório foi excelente. Acredito que serei capaz, com os nortes apresentados no minicurso, de utilizar e pesquisar sobre openmp.”
- “Eu gostei bastante do curso, mas acho que o aproveitamento poderia ser melhor se houvesse um pouco mais de tempo para abordar os conteúdos.”
- “talvez ter um curso em apenas um período (manhã/tarde) durante uma semana inteira seja mais interessante pois assim, o conteúdo poderia ser estendido.”
- “Um curso um pouco maior e dividido em mais dias.”
- “Eu gostei bastante do curso, mas acho que o aproveitamento poderia ser melhor se houvesse um pouco mais de tempo para abordar os conteúdos.”

5.4.5 *Análise dos resultados*

Tendo em vista o objetivo desse experimento e a hipótese do presente trabalho, o desempenho técnico dos alunos com formação básica prévia e sem formação básica prévia foram comparados. Além disso, são avaliados por meio das respostas do questionário qualitativo: a satisfação dos alunos durante o processo de aprendizagem, o conteúdo ministrado e o sentimento se de fato obtiveram um conhecimento novo. A seguir é realizada uma análise dos resultados com base nas questões de pesquisa.

QP01: Os alunos sem formação básica prévia foram capazes de compreender e aplicar de forma prática os tópicos de ensino tanto quanto os alunos com formação básica prévia?

O gráfico na Figura 30 representa a distribuição das médias de aproveitamento total de todos os alunos participantes com e sem conhecimento básico prévio no curso, considerando todos os exercícios aplicados. Pode-se perceber observando o gráfico que todas as médias, em ambos os grupos, estão acima de 83% de aproveitamento total, o que indica que todos os alunos obtiveram um desempenho adequado nas atividades práticas aplicadas e foram capazes de compreender e aplicar o conteúdo ensinado. Comparando a dispersão de notas entre a amostra de alunos, percebemos que, apesar da média e o mínimo serem inferiores na primeira amostra, a maior parte dos valores detém-se no mesmo intervalo de porcentagem, demonstrando forte intersecção entre o aproveitamento total de ambos os grupos. Além disso, as Tabelas 15 e 16 apresentam boas médias para todos os exercícios em cada critério de avaliação.

Assim, podemos concluir que os alunos, mesmo sem formação básica prévia em conceitos de computação, foram capazes de compreender o conteúdo de paralelismo e, logo no primeiro contato, produzir soluções de código paralelo aplicando os conceitos ensinados tanto quanto alunos que já possuíam experiência anterior com a computação.

Além disso, as respostas do questionário qualitativo em relação ao nível de conhecimento de paralelismo que os estudantes julgavam possuir antes e após o curso indica um aumento médio de mais de 24% após o curso no geral. As respostas a essas questões são inteiramente representadas de forma visual nas Figuras 32 e 33 por meio de gráficos de barras. Portanto, pode-se concluir que os alunos compreenderam o conteúdo que foi ministrado, sendo suficiente ao ponto de não ultrapassar a limitação da falta de formação básica prévia.

Figura 32 – Respostas da questão 2 da avaliação qualitativa - OpenMP/2018

Como você classifica seu conhecimento em programação paralela antes da realização deste curso?

23 respostas

Figura 33 – Respostas da questão 4 da avaliação qualitativa - OpenMP/2018

Como você classifica seu conhecimento em programação paralela após a realização deste curso?

23 respostas

Esse resultado demonstra que os tópicos selecionados a partir das camada de Programação, especificada na seção 4.4, foram compreensíveis por alunos sem formação prévia, validando que o enfoque nos tópicos da camada de Programação pode colaborar para o aprendizado prático do aluno e não implica em limitações ao ensino.

QP02: Os códigos desenvolvidos pelos alunos demonstram aplicação dos conceitos de paralelismo?

Observando a Figura 29, pode-se verificar a correção das atividades práticas de acordo com as métricas de avaliação isoladas. A primeira métrica (paralelismo + legibilidade) está relacionada à aplicação correta e demonstração de aspectos paralelos do problema/aplicação e boas práticas de programação. A análise do aproveitamento nesta métrica indica a distribuição de notas acima 87,5% de aproveitamento em ambos os grupos, demonstrando que as soluções desenvolvidas pelos estudantes revelavam um aproveitamento satisfatório dos conceitos de paralelismo. Portanto, pode-se concluir que, ao longo do experimento, os participantes aplicaram corretamente o paralelismo em seus códigos, apontando para o desenvolvimento do pensamento paralelo.

QP03: Os alunos foram capazes de usar corretamente o modelo de programação (OpenMP)?

A segunda métrica de avaliação exibida na Figura 29 identifica o aproveitamento das atividades práticas de acordo com o uso do modelo de programação, ou seja, uso adequado da plataforma disponível. Analisando a dispersão do aproveitamento desta métrica, verifica-se a distribuição de notas acima 86,7% de aproveitamento em ambos os grupos, mostrando que os alunos, em geral, souberam usar adequadamente o recurso de programação ensinado. Portanto, pode-se concluir que os estudantes foram capazes de utilizar corretamente em seus códigos o modelo de programação proposto.

QP04: Os códigos desenvolvidos pelos alunos funcionavam/rodavam corretamente?

A terceira métrica de avaliação exibida na Figura 29 avalia se a saída do código está correta e conforme o esperado. A dispersão de aproveitamento nessa métrica demonstra a distribuição em um intervalo maior. Uma parte dos alunos cometeram erros relacionados à programação sequencial ou falta de atenção à saída que era esperada, prejudicando, desta forma, a nota da atividade e comprometendo a saída correta da aplicação, o que justifica a maior dispersão dessa nota. Ainda assim, o gráfico demonstra que pelo menos 75% das notas obteve um aproveitamento superior a 73% nessa métrica. Portanto, pode-se afirmar que, salvo exceções, os códigos desenvolvidos funcionavam corretamente.

QP05: Os materiais e conteúdos abordados no curso foram suficientes, insuficientes ou excessivos?

De acordo com as respostas obtidas no teste qualitativo, conforme exibido na Figura 34, os alunos ficaram satisfeitos com os exemplos e exercícios aplicados em sala de aula e julgaram suficientes para um curso de introdução ao OpenMP, no entanto, de acordo com o resultado da questão 7, apresentado na Figura 35, parte deles não se sentiu confiante quanto ao domínio do conhecimento adquirido. Uma justificativa plausível, apontada pelos próprios alunos participantes do experimento, é o tempo que tinham disponível para implementar as atividades

e estudarem o conteúdo, considerando que o curso foi realizado com 8 horas de duração total em 1 dia. Nos comentários sobre o curso, surgiram muitas sugestões de que o curso tivesse uma duração maior. Apesar disso, conforme dito anteriormente, foi possível perceber que os alunos, mesmo sem conhecimento prévio, não relataram limitações maiores durante o processo de ensino-aprendizagem.

Figura 34 – Respostas da questão 6 da avaliação qualitativa - OpenMP/2018

Você acha que os exemplos/exercícios propostos foram suficientes para uma introdução ao OpenMP?

23 respostas

Figura 35 – Respostas da questão 7 da avaliação qualitativa - OpenMP/2018

Como você classifica seu conhecimento em OpenMP após a realização deste curso?

23 respostas

Além disso, percebe-se pelas respostas das questões 5 e 8, exibidas nas Figuras 36 e 37 respectivamente, que o objetivo de apresentar os conceitos de programação paralela, estimulando o aluno a pensar em paralelo nas soluções de problemas computacionais foi alcançado. Portanto, pode-se afirmar que os tópicos de ensino definidos nas camadas de conceitos, devidamente ordenados e organizados para um curso introdutório, foram suficientes para os participantes compreenderem a execução de algoritmos paralelos, independente de seu contato anterior com a computação.

Figura 36 – Respostas da questão 5 da avaliação qualitativa - OpenMP/2018

Você considera que este curso o estimulou a pensar em paralelo na construção de algoritmos?

23 respostas

Figura 37 – Respostas da questão 8 da avaliação qualitativa - OpenMP/2018

O curso contribui para você entender a programação de threads para máquinas paralelas?

22 respostas

QP06:Qual o sentimento dos alunos em relação ao curso? Os alunos demonstraram interesse no tema ensinado?

Observando a Figura 36, é possível verificar que os participantes consideram que foram estimulados a pensar em soluções paralelas por meio da construção dos algoritmos propostos durante o curso. De acordo com o resultado apresentado na Figura 37, eles puderam compreender os princípios da execução de *threads* paralelas/concorrentes em um mesmo programa, especificamente usando o modelo de programação OpenMP.

Além disso, as respostas da questão 9, apresentadas na Figura 38, demonstram que a maior parte dos alunos avaliou de forma positiva o conteúdo ministrado no curso, indicando que os mesmos aprovaram o tema ensinado e possuem interesse no assunto.

Figura 38 – Respostas da questão 9 da avaliação qualitativa - OpenMP/2018

De uma forma geral, como você avalia o conteúdo ministrado neste curso?

23 respostas

QP07: Os alunos sem formação prévia sentiram que puderam aprender programação paralela?

Na Figura 31 podemos ver as médias das respostas do questionário qualitativo agrupadas por alunos sem conhecimento prévio e com conhecimento prévio. Podemos comparar as médias obtidas nas questões 2 e 4 para avaliar se os participantes acreditam que houve um crescimento no conhecimento de programação paralela. A questão 2 demonstra que os alunos possuíam pouco ou nenhum conhecimento em programação paralela antes da participação no curso. A questão 4, contudo, demonstra que, após a participação no curso, eles acreditavam possuir um conhecimento significativamente maior. Diante desse cenário, houve um crescimento médio de 32,5% em relação à compreensão de programação paralela anterior e posterior à participação no curso, indicando que os estudantes sem conhecimento prévio sentiram que puderam compreender o conteúdo ministrado.

5.5 Experimento III: Curso de Difusão Cultural de Introdução a Pthreads

Após a realização do segundo experimento averiguamos quais seriam os resultados de aprendizagem aplicando um curso no mesmo formato, porém visando o ensino de paralelismo com um modelo de programação diferente: Pthreads. A turma neste experimento também estava composta de alunos sem e com formação prévia em computação. Assim, foi possível verificar a diferença no desempenho e aprendizado entre ambos os grupos considerando o uso de um modelo de programação paralela diferente. As seções a seguir descrevem o experimento detalhadamente com base nas etapas do processo experimental definido em Wohlin *et al.* (2012).

5.5.1 Definição do experimento

Objetivo

O objetivo do presente experimento é examinar se discentes sem formação prévia básica em computação, incluindo alunos no primeiro ano da graduação, podem compreender e aplicar no desenvolvimento de código os conceitos fundamentais de paralelismo com *threads*. Durante o experimento, foi analisado se os estudantes sem conhecimento prévio foram capazes de compreender e aplicar a programação paralela da mesma maneira que alunos que possuíam formação prévia em computação.

Objeto

O objeto de estudo deste experimento é a implementação de um curso de programação paralela usando Pthreads para o desenvolvimento de código voltado a alunos sem formação básica em computação. Essa implementação segue as iniciativas da ACM ([CURRICULA, 2013](#)) e IEEE ([PRASAD et al., 2012](#)) selecionando os tópicos inseridos nas camadas de conceitos definidas na Seção 4.4, especificamente a de Programação.

Foco qualitativo

Deseja-se medir o desempenho no processo de ensino-aprendizagem obtido por alunos de diferentes contextos ao serem expostos a aspectos práticos e teóricos de programação paralela e seus paradigmas.

Perspectiva

Considerando os objetivos de ensino e tendo como base o método de ensino tradicional, temos como perspectiva desse experimento o ambiente real de sala de aula. Ainda, por se tratar de um curso voltado à prática de programação, a perspectiva é de que o curso seja conduzido em um laboratório de computação.

Contexto

Mediante a perspectiva descrita, o experimento foi realizado em dezembro de 2018 no Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo no contexto de um laboratório de programação, como curso de difusão cultural, cujas vagas foram abertas à comunidade. O curso teve duração total de 8 horas, divididas entre 4 tópicos de 2 horas cada. O único requisito determinado para o curso era possuir conhecimento básico em programação sequencial na linguagem C. De fato, os alunos não precisariam ter conhecimentos sólidos em programação C, porém, deveriam saber como compilar, executar e utilizar estruturas de programação simples em C.

5.5.2 Planejamento

Definição das questões de pesquisa

Em vista dos objetivos deste experimento e do projeto, foram elencadas as seguintes questões de pesquisa:

- QP1:** Os alunos sem formação básica prévia foram capazes de compreender e aplicar de forma prática os tópicos de ensinado tanto quanto os alunos com formação básica prévia?
- QP2:** Os códigos desenvolvidos pelos alunos demonstram aplicação dos conceitos de paralelismo?
- QP3:** Os alunos foram capazes de usar corretamente o modelo de programação (Pthreads)?
- QP4:** Os códigos desenvolvidos pelos alunos funcionavam/rodavam corretamente?
- QP5:** Os materiais e conteúdos abordados no curso foram suficientes, insuficientes ou excessivos?
- QP6:** Qual o sentimento dos alunos em relação ao curso? Os alunos demonstraram interesse no tema ensinado?
- QP7:** Os alunos sem formação prévia sentiram que puderam aprender programação paralela?

Seleção de sujeitos

Os alunos participantes do experimento foram provenientes de uma amostra inteiramente aleatória devido a matrícula ser aberta à comunidade. O curso não possuía caráter obrigatório e, devido ao único requisito ser o de conhecimento básico em linguagem C, os alunos matriculados apresentavam diferentes características, como nível de profundidade em programação, escolaridade e área de conhecimento.

Descrição de instrumentação

A instrumentação utilizada durante o curso foi: apresentação de slides, formulários online para submissão dos exercícios e o teste qualitativo.

Os exercícios aplicados durante o curso foram: Exercício 1 - Soma de um intervalo números

Exercício 2 - Cálculo do pi com soma local

Exercício 3 - Cálculo do pi com soma global

Exercício 4 - Produtor consumidor

Exercício 5 - Alerta, variáveis de condição

5.5.3 Execução

Preparação

Após a definição do experimento e respectivo planejamento, foi submetido o pedido para realização do curso de difusão cultural no ICMC com carga horária e tema propostos nesse experimento. O pedido foi aprovado e, então, reservou-se o laboratório onde ocorreria o curso. A proposta de curso visava os seguintes objetivos:

- Familiarização do estudante com a execução de processos/*threads* em paralelo para otimizar desempenho de soluções computacionais;
- Compreensão do modelo de programação Pthreads e suas implicações;
- Aplicação das primitivas do Pthreads e obtenção de desempenho em algoritmos paralelizáveis de maneira prática; e
- Capacitação na criação e sincronização de *threads*.

As apresentações de *slides* para cada aula foram preparadas cuidadosamente buscando obter clareza, sequência e boa didática para a transmissão do conteúdo. Ainda, foram preparadas as descrições e soluções codificadas de cada exercício que seria proposto durante as aulas, além dos formulários online os quais seriam utilizados pelos alunos para submissão dos exercícios implementados em cada aula.

Operação

A Tabela 18 apresenta uma visão geral dos tópicos abordados durante as aulas.

Tabela 18 – Planejamento de curso

Tópico	Conceitos
1- Introdução ao Paralelismo e às PThreads.	<ul style="list-style-type: none"> - Motivação - Computação sequencial vs. paralela - Atividades sequenciais & paralelas: exemplos no dia-a-dia e na computação; - Threads: conceitos e exemplos; - PThreads: histórico, funcionamento e exemplos; e - Desenvolvimento de um programa C com PThreads usando blocos estruturados em paralelo.
2 – Manipulação de Dados Locais, Paralelização de Tarefas com Threads e Avaliação de Desempenho.	<ul style="list-style-type: none"> - Criando e sincronizando a finalização de Pthreads: exemplos práticos; - Avaliação de desempenho (tempo de resposta, speedup e eficiência); e - Exercícios práticos com desenvolvimento de código, avaliando resultados e desempenho.
3 – Trabalhando com dados compartilhados protegidos por semáforos binários.	<ul style="list-style-type: none"> - Regiões críticas, condições de disputa e exclusão mútua; - Protegendo regiões críticas com <code>pthread_mutex_lock()</code> e <code>pthread_mutex_unlock()</code>; - Análise de desempenho
4 – Sincronização de threads com variáveis de condição e aspectos avançados de sincronização	<ul style="list-style-type: none"> - Variáveis de condição (sleep/wake-up); - Usando variáveis de condição com C e PThreads; e - Exercícios práticos avaliando resultados e desempenho;

5.5.4 Apresentação dos resultados

Caracterização dos participantes

Devido ao fato das vagas serem abertas à comunidade e por tratar-se de um curso de extensão, a população participante do experimento derivou-se de alunos matriculados em diferentes cursos e estágios de formação. A Figura 39 mostra a quantidade de alunos agrupados por curso e nível de formação. Além disso, apresenta a categorização sem formação prévia em computação (em azul) e com formação prévia em computação (em laranja). Analisando o referido gráfico, é possível constatar que participaram do experimento discentes de cursos relacionados à computação, como Bacharelado em Ciência da computação, Sistemas da Informação e Engenharia da Computação, assim como de áreas não relacionadas à computação, como Física e matemática. No entanto, vale ressaltar a participação expressiva de discentes sem formação básica prévia em computação, em especial, dos alunos do primeiro ano (segundo período) de cursos relacionados à computação, que consistiram a maior parte da amostra. Alunos de computação no segundo período (primeiro ano) foram classificados como sem conhecimento prévio em função das disciplinas ainda não cursadas pelos mesmos.

A população considerada para essa análise engloba os indivíduos que completaram o questionário qualitativo disponibilizado ao final do experimento. Portanto, o número total de participantes dessa amostra é de 17 alunos.

Figura 39 – Alunos por curso e período - Pthreads/2018

Contagem de aluno por curso e período

A partir disso, a Figura 40 apresenta a quantidade total de alunos nas duas categorias. Destaca-se que a população sem conhecimento prévio em conceitos computação é predominante, compondo mais de 82% da amostra. O tamanho da amostra de cada grupo deve ser levado em

consideração durante as avaliações, visto que, com uma população maior, os dados avaliados tendem a apresentar maior dispersão.

Figura 40 – Classificação de alunos de acordo com formação prévia - Pthreads/2018

Aprendizado Técnico

Conforme descrito anteriormente, no decorrer do curso foram realizadas atividades práticas de programação visando a aplicação do conteúdo técnico apresentado. Por meio dessas atividades práticas, pôde-se avaliar sistematicamente e de forma gradual o desempenho técnico dos estudantes em relação ao tópico ensinado. Dessa forma, possibilita-se inferir, pela análise de aproveitamento das atividades, se os alunos puderam, de fato, compreender e aplicar o conteúdo ministrado durante os módulos do curso. Após o estudo destes resultados, podemos afirmar se o processo de ensino-aprendizagem foi eficaz para os participantes do curso.

A Figura 41 apresenta um diagrama de caixas que demonstra a dispersão de notas de acordo com as métricas de avaliação estabelecidas para as atividades práticas. O intervalo de dispersão de cada caixa representa a distribuição das médias de aproveitamento global em cada métrica, ou seja, engloba totalmente as avaliações das atividades práticas do curso. Foram apresentadas conjuntamente, com o intuito de compará-las, as representações da dispersão de aproveitamento dos alunos sem conhecimento prévio e com conhecimento prévio. Assim, apresenta-se uma visão completa do resultado obtido por cada métrica avaliada no curso.

Figura 41 – Comparação da distribuição de notas de cada métrica de avaliação - Pthreads/2018

Os resultados exibidos na Figura 41, demonstram que o valor das médias é superior a 95% de aproveitamento em todas as métricas de avaliação para ambos os grupos de alunos avaliados. Pode-se observar que o menor valor de aproveitamento em cada métrica é superior a 85% e que pelo menos mais da metade dos valores estão acima de 90% de aproveitamento. Esses resultados indicam que os estudantes tiveram um bom aproveitamento em cada uma das três métricas definidas durante todo o curso, uma vez que essas notas envolvem todos as atividades práticas realizadas, e que desenvolveram soluções com qualidade adequada, fazendo o uso correto dos conceitos de paralelismo visando o desempenho mediante o modelo de programação apresentado.

A Tabela 19 e a Tabela 20 contemplam, respectivamente, os grupos de alunos sem conhecimento prévio e com conhecimento prévio em conteúdos de ciência da computação, apresentando as médias de aproveitamento para cada métrica de avaliação em cada atividade prática realizada em sala de aula durante o curso. Além do aproveitamento pelas métricas, exibe a quantidade de submissões de cada grupo para a mesma atividade.

Podemos observar nas tabelas que há diferença entre a quantidade de submissões de exercícios entre as amostras. Inclusive, nos exercícios 2 e 3 foram obtidas submissões, ainda que em menor número, somente de alunos sem conhecimento prévio. Acredita-se que esse número reduzido de submissões nesses exercícios seja devido à ausência de tempo para entregar uma solução satisfatória ou os alunos ainda não se sentiam confiantes para entregar as soluções que elaboraram. Pelo fato de haver uma quantidade maior de participantes considerados sem conhecimento prévio em computação e, conseqüentemente, de submissões, a dispersão das notas

foi maior neste grupo. Ainda assim, podemos visualizar na Figura 41 que, comparando as duas categorias, ambas possuem a maior parte das notas acima de 90% de aproveitamento.

Tabela 19 – Médias das métricas considerando a amostra de alunos sem conhecimento prévio | Pthreads 2018

Alunos sem conhecimento prévio				
	Paralelismo + legibilidade	Uso do modelo de programação	Saída correta	Quantidade de submissões
Ex1	96.2%	98.5%	93.8%	14
Ex2	100.0%	100.0%	100.0%	2
Ex3	95.0%	87.5%	100.0%	2
Ex4	87.1%	95.7%	89.3%	14
Ex5	100.0%	95.5%	96.4%	11
Média	95.7%	95.4%	95.9%	Total: 43

Tabela 20 – Médias das métricas considerando a amostra de alunos com conhecimento prévio | Pthreads 2018

Alunos com conhecimento prévio				
	Paralelismo + legibilidade	Uso do modelo de programação	Saída correta	Quantidade de submissões
Ex1	99.5%	100.0%	92.5%	4
Ex2	-	-	-	0
Ex3	-	-	-	0
Ex4	98.0%	100.0%	94.0%	5
Ex5	100.0%	100.0%	100.0%	3
Média	99.2%	100.0%	95.5%	Total: 12

Por fim, a Figura 42 exibe, por meio de um diagrama de caixas, a dispersão das médias de aproveitamento dos participantes ao final do curso. A porcentagem de aproveitamento total representa a média final que o aluno obteve no curso considerando a avaliação do aproveitamento de todos os exercícios realizados no curso.

Analisando o gráfico da Figura 42, percebe-se que a média no grupo sem conhecimento é inferior ao outro. No entanto, como pontuado anteriormente nesta seção, esse fato é justificado devido à diferença na quantidade de submissões possuir uma ampla desigualdade. A média final de todos os alunos participantes do experimento está acima de 90% de aproveitamento total, ou seja, obtiveram um desempenho esplêndido no curso.

Foi realizado o teste de Wilcoxon para amostras independentes com o intuito de investigar se as amostras são estatisticamente equivalentes ou se possuem diferença significativa. Para isso, foram incluídas no teste todas as notas de exercícios realizados e definidas as seguintes hipóteses, onde μ_1 representa a amostra de alunos sem conhecimento prévio e μ_2 , com conhecimento prévio:

Hipótese Nula: Não há diferença estatisticamente significativa no aproveitamento total dos experimentos realizados entre os grupos de alunos com e sem conhecimento.

Figura 42 – Comparação das porcentagens de aproveitamento do curso - Pthreads/2018

$$H_0 : \mu_1 - \mu_2 = 0$$

Hipótese Alternativa: Há diferença estatisticamente significativa no aproveitamento total dos experimentos realizados entre os grupos de alunos com e sem conhecimento.

$$H_1 : \mu_1 - \mu_2 \neq 0$$

A partir dessas hipóteses, foi obtido o resultado exibido na Tabela 21.

Tabela 21 – Teste de Wilcoxon para amostras independentes

Estatística do Teste (Wilcoxon)	
Informações	Valores
Estatística	57
P-valor	0,168
Hipótese Nula	0
Limite Inferior	-0,099974698
(Pseudo) Mediana	0,199952652
Limite Superior	0,799938666
Nível de Confiança	0,95

Como o P-valor é igual a 16,8%, aceitamos a hipótese nula a uma significância de 95%. Assim, existem evidências suficientes para aceitar a hipótese nula e afirmar que não há diferença estatística significativa no aproveitamento total dos experimentos realizados entre os grupos de alunos com e sem conhecimento prévio.

Avaliação qualitativa

Ao final do experimento foi disponibilizado um questionário respondido de maneira anônima e opcional, com o objetivo de avaliar a satisfação dos alunos relativa a organização, materiais e conteúdo ministrado no curso, visando concluir se foi suficiente para o processo de ensino-aprendizagem segundo a ótica dos alunos. Além disso, foi possível examinar se os próprios alunos julgavam ter adquirido conhecimento de paralelismo e o nível de domínio sobre programação paralela com *Pthreads*. As respostas de cada questão davam-se pela escala Likert, ou seja, opções com valores de 1 a 5, onde 1 representa a pior nota e 5 a melhor nota.

Abaixo são listadas as questões do formulário qualitativo.

- Q1:** Como você classifica seu conhecimento em programação sequencial (como a C, por exemplo) antes da realização deste curso?
- Q2:** Como você classifica seu conhecimento em programação paralela antes da realização deste curso?
- Q3:** Você considera importante aprender programação paralela, desde o começo do curso de computação?
- Q4:** Como você classifica seu conhecimento em programação paralela após a realização deste curso?
- Q5:** Você considera que este curso o estimulou a pensar em paralelo na construção de algoritmos?
- Q6:** Você acha que os exemplos/exercícios propostos foram suficientes para uma introdução ao *Pthreads*?
- Q7:** Como você classifica seu conhecimento em *Pthreads* após a realização deste curso?
- Q8:** O curso contribuiu para você entender a programação de *threads* para máquinas paralelas?
- Q9:** De uma forma geral, como você avalia o conteúdo ministrado neste curso?
- Q10:** Você acha que o tempo/duração do curso foi suficiente para abordar o conteúdo proposto?

A Figura 43 demonstra a média das respostas para cada questão do formulário, na mesma sequência listada anteriormente, comparando as respostas dos alunos sem conhecimento prévio e com conhecimento prévio. As questões são identificadas por palavras-chaves referentes ao seu conteúdo.

Figura 43 – Resultado avaliação qualitativa - Pthreads/2018

Analisando a Figura 43, a média das respostas da questão 1 indica que os alunos se consideravam experientes em programação procedural utilizando linguagem C. No entanto, em razão do baixo valor da média na questão 2 em ambos os grupos, podemos concluir que os participantes possuíam pouco ou nenhum conhecimento em programação paralela antes do curso. Na questão 4, podemos observar que, após o curso, eles consideravam possuir um conhecimento maior em programação paralela, ou seja, que houve um aprendizado significativo durante o experimento. Destaca-se o grande avanço principalmente relacionado à população sem conhecimento prévio que, apesar de não possuir formação prévia em computação, demonstrou um alto nível de aprendizagem do conteúdo considerando a carga horária disponibilizada. Além disso, vale ressaltar as médias elevadas para as respostas das questões 6, 8 e 9. Isso demonstra que os participantes ficaram satisfeitos com o conteúdo ministrado e o formato do experimento em si, validando que os alunos, especificamente sem conhecimento prévio em computação, podem aprender o conteúdo de programação paralela superando os obstáculos impostos pela ausência de formação prévia.

Além das questões mencionadas acima, o questionário disponibilizava uma área de texto para o aluno descrever comentários, críticas ou sugestões. Destacamos, na lista a seguir, as críticas positivas e negativas recebidas de maior relevância.

- "Gostei muito do curso. Foi fluido num ritmo bom. Os exemplos dados foram bons e as explicações foram claras. O curso não sobrecarrega conteúdo e dá bastante espaço para práticas simples, para entender os conceitos básicos e funcionalidades iniciais. Ainda sim, ao fim do curso da pra sair bastante feliz com o que se aprendeu rs. O melhor do curso foram os códigos pré programados, facilmente legíveis e compreensíveis para se trabalhar um pequeno conceito nele, economizando muito tempo. Sem falar que os exemplo preparados foram muito bem pensados e focados no necessário."
- "Gostei muito do curso, a dinâmica é bem interessante. O tempo e a escolha dos exercícios foi apropriada para a duração. A quantidade de slides, conteúdo discursivo e exercícios também foi bem equilibrada. Excelente. Ouvi inclusive comentários de alunos da graduação relatando que deveriam ter mais cursos curtos com esta qualidade."
- "(...) a metodologia para empregar os exercícios poderia ser melhorada. Isto é, alguns exercícios repetiu o uso de conceitos (por exemplo, a parte sobre uso do mutex nas contagens de PI e soma de array). Essa questão é importante e poderia ter exemplos mais diferentes, porém ainda simples."
- "Creio que, devido à longa duração, o final acabou ficando um pouco cansativo, sugeriria que fosse ministrado em duas manhãs, mas ainda assim foi muito interessante e consegui aprender bastante!"
- "As pessoas que trouxeram conteúdo possuem boa didática e se esforçaram em facilitar a aprendizagem da turma. Como o assunto do curso é bem abrangente, o tempo de curso, ao meu ver foi insuficiente, pois com mais tempo, mais exemplos/exercícios poderiam ser abordados."
- "Foi muito bom. Talvez, se fosse dividido em dois dias (ou diminuído a quantidade de horas totais) seria mais agradável."
- "O número de exercícios parecia grande. Foi demorado adquirir um pouco de familiaridade com o pthreads no primeiro exercício e não foi possível fazer alguns."

5.5.5 Análise dos resultados

A partir da apresentação dos resultados na seção 5.5.4, estes devem ser analisados e investigados no intuito de contribuir com o estudo das hipóteses do presente trabalho. Alguns dos principais resultados procedentes deste experimento são o desempenho técnico de alunos com e sem formação prévia em computação, a comparação entre eles e a satisfação dos alunos quando expostos ao aprendizado do conteúdo de paralelismo com *Pthreads*, principalmente os que não possuem os conceitos fundamentais da computação. A seguir, a análise será realizada com base nas questões de pesquisa elencadas na definição deste experimento, buscando respondê-las de

forma objetiva, completa e apontando para os resultados quantitativos e qualitativos descritos anteriormente.

QP1: Os alunos sem formação básica prévia foram capazes de compreender e aplicar de forma prática os tópicos de ensinado tanto quanto os alunos com formação básica prévia?

O gráfico na Figura 42 representa a distribuição das médias de aproveitamento total de todos os alunos participantes com e sem conhecimento básico prévio no curso, considerando todos os exercícios aplicados. Primeiramente, percebe-se que a distribuição das médias dos do grupo com conhecimento prévio é pouco dispersa, contendo médias entre 97,8% e 100% de aproveitamento. Isso é justificado pela quantidade reduzida de alunos nessa amostra. Portanto, podemos concluir que o aproveitamento de todos os discentes sem conhecimento prévio no curso aproximou-se de 100% nos exercícios que foram entregues, conforme demonstrado na Tabela 20. Embora tratando-se de poucos alunos e poucas submissões, podemos afirmar que estes não apresentaram dificuldade com o aprendizado e aplicação dos conceitos de paralelismo utilizando *Pthreads*.

Já a amostra sem conhecimento prévio, apresenta uma dispersão das médias relativamente maior, contendo médias entre 90,1% e 100% de aproveitamento. Entretanto, esse resultado indica que, mesmo em quantidade maior, todos os alunos sem conhecimento prévio obtiveram uma média de aproveitamento total entre 90% e 100%, caracterizando um desempenho satisfatório na implementação dos exercícios que foram aplicados no experimento.

Após essa comparação, podemos concluir que os discentes, mesmo sem formação básica prévia em conceitos de computação, foram capazes de compreender o conteúdo de paralelismo tanto quanto os que possuem formação prévia em computação, considerando o alto desempenho obtido por ambos os grupos e, também, a diferença entre os tamanhos das populações. Ambos foram capazes de produzir soluções de código paralelo aplicando os conceitos de paralelismo ministrados.

Além disso, o questionário qualitativo possui questões com o objetivo de verificar o nível de conhecimento de paralelismo antes e após o curso que os indivíduos julgavam possuir, especificamente as questões 2 (conhecimento anterior ao curso) e 4 (conhecimento após a participação do curso). O resultado dessas questões mostram um aumento total de 40% após o curso, sendo o maior crescimento para os alunos sem conhecimento prévio, fortalecendo a conclusão de que estes podem aprender tanto quanto os que possuem formação prévia. As respostas às questões 2 e 4 são inteiramente representadas de forma visual nas Figuras 44 e 45 por meio de gráficos de barras.

Este resultado demonstra que os tópicos de ensino selecionados foram compreensíveis por alunos sem formação prévia, validando que, dentre as camadas de conceitos de paralelismo apresentadas na seção 4.4, o enfoque nos tópicos da camada de Programação pode colaborar

para o aprendizado prático do aluno e não implica em limitações ao ensino.

Figura 44 – Respostas da questão 2 da avaliação qualitativa - PThreads/2018

Como você classifica seu conhecimento em programação paralela antes da realização deste curso?

17 respostas

Figura 45 – Respostas da questão 4 da avaliação qualitativa - PThreads/2018

Como você classifica seu conhecimento em programação paralela após a realização deste curso?

17 respostas

QP2: Os códigos desenvolvidos pelos alunos demonstram aplicação dos conceitos de paralelismo?

O resultado da correção de todos os exercícios que foram submetidos pelos alunos de acordo com a primeira métrica de avaliação (Paralelismo e legibilidade) são exibidos na Figura 41. Podemos verificar que, no que se refere aos alunos sem conhecimento prévio, a média, em cada exercício, das notas das soluções que foram submetidas obtiveram um aproveitamento superior a 87% nesta métrica, sendo a maior parte deles acima de 90%. Referente aos com conhecimento prévio, observamos que a distribuição está acima de 98% de aproveitamento nesta métrica. A partir desses resultados, concluímos que, mediante as médias das soluções nas atividades considerando a primeira métrica de avaliação, os participantes demonstraram saber

aplicar corretamente os conceitos de paralelismo ensinados nos códigos utilizando Pthreads, expressando o desenvolvimento do pensamento paralelo.

QP3: Os alunos foram capazes de usar corretamente o modelo de programação (Pthreads)?

O resultado da correção dos exercícios consoantes com a segunda métrica de avaliação (uso do modelo de programação) é exibido na Figura 41. Podemos averiguar que a amostra de estudantes com conhecimento prévio obteve médias de 100% de aproveitamento em todas as atividades propostas, considerando essa métrica. Isso evidencia que, apesar de poucos, os alunos que submeteram as soluções dos exercícios fizeram o uso correto do modelo de programação nas atividades entregues. A amostra de discentes sem conhecimento prévio obteve médias acima de 87,5% de aproveitamento em todas as atividades propostas, considerando essa métrica, sendo pelo menos 75% dessa distribuição superior a 90% de aproveitamento, demonstrando um uso do modelo de programação de excelência nos códigos desenvolvidos. A partir desses resultados, concluímos que, mediante as médias das soluções nas atividades considerando a segunda métrica de avaliação, os participantes demonstraram ter obtido conhecimento suficiente para usar de forma ideal o modelo de programação proposto no curso.

QP4: Os códigos desenvolvidos pelos alunos funcionavam/rodavam corretamente?

O aproveitamento da terceira métrica de avaliação (saída correta/esperada) das soluções submetidas é apresentado na Figura 41. É possível verificar que em ambos os grupos, sem conhecimento prévio e com conhecimento prévio, as distribuições de aproveitamento nesta métrica encontram-se em um intervalo de valores muito próximos e, mesmo apresentando uma dispersão maior, o grupo de alunos sem conhecimento prévio obteve uma média geral ligeiramente superior ao alunos com conhecimento prévio, apesar de possuir uma amostra maior. A maioria dos valores em ambos os grupos situam-se no intervalo de 90% a 100% de aproveitamento. A partir desses resultados, concluímos que, mediante as médias das soluções nas atividades considerando a terceira métrica de avaliação, os participantes demonstraram desenvolver códigos que, em sua maioria, apresentavam a saída conforme o esperado. Com isso, podemos dizer que os códigos compilavam e executavam corretamente, mostrando que as soluções propostas tinham o funcionamento correto.

QP5: Os materiais e conteúdos abordados no curso foram suficientes, insuficientes ou excessivos?

O questionário qualitativo avalia a opinião dos participantes, por meio da questão 6, relacionada aos exemplos e exercícios aplicados no curso. A Figura 46 apresenta a distribuição de repostas para essa questão. Como pode-se verificar, todas as avaliações estão entre 4 ou 5, sendo que 76,5% das respostas são 5 de acordo com a escala Likert. Por meio dessa questão, podemos concluir que os alunos não sentiram-se pressionados com os exercícios propostos de um conteúdo totalmente inédito para eles e os exemplos fornecidos durante o processo de ensino-

aprendizagem foi suficiente para ilustrar os conceitos ensinados. Ainda, podemos observar nas repostas da questão 7, representadas na Figura 47, que 88,3% dos estudantes consideravam seu conhecimento em Pthreads igual ou superior a 3, o que representa uma boa avaliação, mostrando que estes puderam adquirir um conhecimento introdutório do modelo de programação com os recursos utilizados em sala de aula.

Figura 46 – Respostas da questão 6 da avaliação qualitativa - PThreads/2018

Você acha que os exemplos/exercícios propostos foram suficientes para uma introdução a PThreads?

17 respostas

Figura 47 – Respostas da questão 7 da avaliação qualitativa - PThreads/2018

Como você classifica seu conhecimento em PThreads após a realização deste curso?

17 respostas

Além disso, analisando as respostas das questões 5 e 8, exibidas nas Figuras 48 e 49 respectivamente, percebe-se que os participantes do curso acreditam que foram estimulados a pensar em soluções paralelas, visto que 94,1% das resposta da questão 5 concentram-se entre os valores 4 e 5. A questão 8 ainda demonstra que 88,2% consideram que adquiriram conhecimento e compreensão de programação com threads em máquinas paralela.

Figura 48 – Respostas da questão 5 da avaliação qualitativa - PThreads/2018

Você considera que este curso o estimulou a pensar em paralelo na construção de algoritmos?

17 respostas

Figura 49 – Respostas da questão 8 da avaliação qualitativa - PThreads/2018

O curso contribuiu para você entender a programação de threads para máquinas paralelas?

17 respostas

Portanto, mediante a apresentação dos resultados, considera-se que os tópicos ministrados puderam atingir o principal objetivo do curso de introduzir o pensamento paralelo para soluções computacionais de modo que os alunos pudessem aplicar os conceitos de forma prática, comparando o desempenho de soluções sequenciais e paralelas e, conseqüentemente, sendo suficientes para uma introdução ao paralelismo com Pthreads, não ultrapassando as limitações de conhecimento de alunos sem conhecimento prévio em computação.

QP6: Qual o sentimento dos alunos em relação ao curso? Os alunos demonstraram interesse no tema ensinado?

As questões 5 e 8 do questionário, Figuras 48 e 49 respectivamente, demonstram que os participantes puderam sentir que foram estimulados a pensar em soluções paralelas por meio dos recursos oferecidos pelo curso. Além disso, puderam compreender a programação e execução de *threads* paralelas/concorrentes em núcleos de processamento diferentes.

As respostas da questão 9, exibidas na Figura 50, apresenta que 94,1% dos alunos avaliaram o curso de forma geral com notas 4 e 5, de acordo com a escala Likert, o que caracteriza uma nota de avaliação satisfatória para o curso como um todo. Portanto, os participantes apreciaram o curso ministrado e veem de forma positiva os resultados de aprendizagem que este proporcionou, indicando que possuem interesse no tema. Vale ressaltar que a maior parte da população deriva-se de estudantes sem conhecimento prévio em computação e, deste modo, esse resultado está fortemente direcionado a estes alunos.

Figura 50 – Respostas da questão 9 da avaliação qualitativa - PThreads/2018

De uma forma geral, como você avalia o conteúdo ministrado neste curso?

17 respostas

QP7: Os alunos sem formação prévia sentiram que puderam aprender programação paralela?

As questões 2 e 4 do questionário qualitativo, representado na Figura 43, são referentes ao conhecimento em programação paralela anterior à realização do curso e após participar do curso, respectivamente. Podemos observar que as médias referentes a essas questões demonstram que os alunos puderam perceber um avanço, cujo valor médio é de 47,5% de aumento, no conhecimento antes e após o curso, ou seja, sentiram que puderam aprender, de fato, programação paralela através do conteúdo ministrado.

5.6 Experimento IV: Curso de Difusão Cultural de Introdução ao *OpenMP* para alunos iniciantes em computação.

A maior parte dos comentários dos participantes nos experimentos anteriores refere-se ao fato de a carga horária do curso ser distribuída em apenas um dia. Alguns comentários, inclusive, sugerem a realização do curso em um período mais extenso para que seja possível absorver melhor o conteúdo e permitir a fixação do conhecimento por meio de mais exercícios. Em

vista disso, este experimento segue a proposta de distribuir os tópicos de ensino em uma carga horária maior e dividida em mais de um dia de curso. Com isso, averiguamos a evolução dos participantes antes e após a participação de um curso um pouco mais longo de programação paralela, além de averiguar qual o interesse dos alunos no tema ensinado ao longo das aulas. Espera-se permitir que os estudantes possam compreender os conceitos de paralelismo de forma consolidada, intensificar a prática de código e sanarem dúvidas ao longo das aulas do curso. O público alvo deste experimento foi restringido a alunos do primeiro ano de graduação para investigar o ensino de programação apenas para discentes sem formação básica prévia em computação, particularmente com o desafio de ensinar programação paralela para alunos com pouco ou nenhum conhecimento em linguagem de programação estruturada. As seções a seguir descrevem o experimento detalhadamente com base nas etapas do processo experimental definido em [Wohlin et al. \(2012\)](#).

5.6.1 Definição do experimento

Objetivo

O objetivo desse estudo experimental é verificar se alunos no primeiro ano de graduação em cursos de computação são capazes de compreender os conceitos de paralelismo e desenvolver soluções aplicando corretamente esses conceitos utilizando OpenMP logo no primeiro bimestre da graduação, antecipando o ensino de tais conceitos que seriam abordados somente em estágios mais avançados da graduação. Isso implica no fato de que grande parte deles pode não conhecer os princípios de desenvolvimento mesmo com linguagem estruturada. Portanto, espera-se que os alunos iniciantes possam compreender programação paralela juntamente com os conceitos básicos de programação em C.

Objeto

O objeto de estudo desse experimento é a proposta de ensino de programação paralela baseando-se em tópicos selecionados das camadas de programação definidas na seção 4.4, especificamente a de Programação, distribuídos em uma carga horária maior de curso e considerando que os participantes são alunos totalmente iniciantes em computação.

Foco qualitativo

O foco qualitativo deste experimento é a mensuração do desempenho dos estudantes do primeiro ano de graduação, de acordo com as métricas de avaliação definidas, no desenvolvimento das atividades práticas propostas pelo experimento. Além disso, averiguar, utilizando a Pesquisa de Interesse de Curso definida em [Keller \(2010\)](#), os graus de atenção, relevância, confiança e satisfação por parte dos alunos, segundo o modelo ARCS, referentes ao curso ministrado e conhecimento adquirido.

Perspectiva

A perspectiva desse experimento é que o curso seja feito sob o ponto de vista de um indivíduo que acaba de ingressar na graduação, portanto, em uma sala de aula acadêmica conduzido como curso de extensão com recursos suficientes para realização do experimento. Considerando que serão avaliados os códigos desenvolvidos pelos participantes, o ambiente ideal é um laboratório contendo computadores disponíveis para os alunos individualmente.

Contexto

Mediante essa perspectiva, o experimento foi realizado no contexto de um curso de difusão cultural em março de 2019 no Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo em um laboratório de computação. Não havia pré-requisitos de conhecimento para matrícula. O experimento foi elaborado com uma carga horária total de 18 horas divididas em 6 aulas, logo, cada aula teve a duração de 3 horas.

5.6.2 Planejamento

Definição das questões de pesquisa

Em vista dos objetivos do projeto e deste experimento, foram elencadas as seguintes questões de pesquisa:

- QP1:** Os alunos do primeiro ano foram capazes de compreender os tópicos de ensino que foram selecionados para as aulas?
- QP2:** Os conceitos de paralelismo foram bem aplicados nas atividades práticas desenvolvidas pelos alunos?
- QP3:** Os alunos foram capazes de usar corretamente o modelo de programação (OpenMP) nas atividades práticas?
- QP4:** Os códigos desenvolvidos pelos alunos funcionavam/rodavam corretamente?
- QP5:** Houve evolução dos alunos no conhecimento de paralelismo e programação paralela após a participação no curso?
- QP6:** Os materiais e conteúdos abordados no curso foram suficientes, insuficientes ou excessivos?
- QP7:** Os alunos demonstraram motivação para com o tema ensinado?

Seleção de sujeitos

Com o intuito de alcançar os objetivos do presente experimento, as matrículas para o curso foram divulgadas somente a alunos no primeiro ano de cursos de graduação da área computação. Deste modo, foi obtida uma população homogênea no sentido do nível de formação

em conceitos fundamentais da computação e, além disso, que pelo menos a maioria dos sujeitos participantes do curso não possuem conhecimento prévio em programação com linguagem procedural. Sendo assim, o ensino de programação paralela deve ocorrer ao mesmo tempo que os alunos aprendem programação sequencial.

Descrição de instrumentação

Para apoiar a execução do experimento, alguns materiais foram produzidos e outros recuperados de experimentos anteriores. Para auxiliar no ensino da teoria e na condução de cada aula, foram elaboradas apresentações de *slides* que direcionavam o conteúdo da aula, além de possibilitar o uso de elementos lúdicos que facilitam o entendimento do tema, como ilustrações, gráficos, diagramas etc. Ainda, em alguns momentos do curso, o código contendo a solução esperada para determinado exercício poderia ser projetado para que os alunos pudessem comparar as soluções desenvolvidas.

As atividades práticas propostas para os alunos foram compostas de exercícios referentes ao tema ministrado na aula; em determinados momentos, sobre programação sequencial ou paralela. A Tabela 22 exibe, em uma visão geral, os exercícios que foram propostos em cada aula do experimento. Vale ressaltar que a última linha da tabela apresenta um exercício extra, disponibilizado com a clareza de que não iria compor a nota final do aluno no curso, entretanto permitiria que os participantes pudessem praticar fora da sala de aula e submeter o código para correção com o objetivo de aperfeiçoar o conhecimento e praticar o uso do modelo de programação.

Tabela 22 – Atividades práticas | OpenMP 2019-1

	Exercícios
Aula 1	1. Hello word sequencial 2. Hello word paralelo
Aula 2	1. Calcular média aritmética (sequencial) 2. Encontrar números perfeitos (sequencial) 3. Encontrar números perfeitos (paralelo)
Aula 3	1. Encontrar números perfeitos (com escalonamento) 2. Soma de vetores (sequencial) 3. Soma de um intervalo de números
Aula 4	1. Soma de um intervalo de números (sem reduction) 2. Operações em vetores com funções (sequencial) 3. Calcular perímetro, área e diagonal de retângulos (paralelo)
Aula 5	1. Soma dos elementos de um vetor (região crítica) 2. Encontrar números perfeitos (região crítica)
Aula 6	Maratona de programação
Desafio extra	Cálculo de Pi em paralelo

Além disso, com a intenção de comparar o conhecimento dos alunos antes e após a realização do experimento, foram elaborados questionários aplicados como pré-teste e pós-teste. Estes possuíam dez questões diferentes, entretanto sobre o mesmo conteúdo: conceitos de

paralelismo, programação sequencial e programação paralela. Na elaboração do pré-teste, foram incluídas cinco questões de programação sequencial para medir o conhecimento dos alunos exclusivamente em C. No pós-teste, o número de questões sobre programação sequencial foi reduzido para apenas uma, em vista de focar no conhecimento de programação paralela obtido. Em ambos os testes, buscou-se avaliar o conhecimento teórico e prático de paralelismo usando o modelo de programação abordado no curso. As questões do pré e do pós-teste podem ser encontradas nos Apêndices B e C respectivamente.

O referencial teórico para medir o grau de motivação dos participantes deste estudo foi embasado no modelo **ARCS**, desenvolvido em Keller (1979) como um modelo para analisar categorias motivacionais dividido em quatro categorias: Atenção (A), Relevância (R), Confiança (C) e Satisfação (S). Essas categorias, de acordo com Keller (2010), suprem muitos dos conceitos e variáveis específicas que caracterizam a motivação humana. O modelo ARCS é resultado de um estudo baseado na literatura acadêmica sobre motivação e práticas bem-sucedidas, sendo validado pela análise vários estudos experimentais (KELLER, 1999). Um dos instrumentos de avaliação que exerce o modelo ARCS é a Pesquisa de Interesse de Curso, conhecido como CIS (do inglês *Course Interest Survey*), também desenvolvido por Keller. Esta instrumentação foi elaborada para medir a motivação dos alunos de acordo com os quatro elementos do modelo ARCS no contexto específico de cursos presenciais ou à distância (KELLER, 2010).

Com o intuito de avaliar a motivação dos estudantes em aprender o conteúdo de paralelismo, optou-se pela aplicação do CIS ao final experimento. O instrumento CIS é constituído por trinta e quatro afirmações, as quais os alunos podem atribuir cinco opções de resposta, no formato de escala Likert, que varia entre 1 até 5, sendo: (1) não verdadeiro; (2) pouco verdadeiro; (3) moderadamente verdadeiro; (4) na maior parte verdadeiro; (5) muito verdadeiro. Isso significa que a pontuação mínima na pesquisa é 34 pontos, e a máxima é 170 pontos. As trinta e quatro questões do modelo conseguem mensurar completamente as categorias de motivação do modelo ARCS: atenção, relevância, confiança e satisfação. Oito questões do instrumento mensuram a atenção, nove medem a relevância, oito determinam a confiança e nove estimam a satisfação do usuário. De acordo com Keller (2010), as pontuações são determinadas pela soma das respostas de cada categoria. Entre as questões do CIS, há algumas que possuem afirmações negativas e, por conta disso, precisam receber pontuações invertidas durante a análise. Assim, se o aluno avaliar em um ponto para uma questão invertida, durante a análise será necessário transformar esse valor em cinco pontos. Keller (2010) recomenda a seguinte transformação para as questões invertidas: 1 = 5, 2 = 4, 3 = 3, 4 = 2, 5 = 1. Com isso, o formulário contendo as questões do CIS foi elaborado para ser respondido pelos alunos participantes ao final do experimento. Assim, ao final do experimento, será possível avaliar, por meio de um material com embasamento teórico científico, a motivação dos estudantes em participar do curso.

5.6.3 Execução

Preparação

Após a definição do experimento e respectivo planejamento, foi submetido o pedido para realização do curso de difusão cultural no ICMC com carga horária e tema propostos nesse experimento. O pedido foi aprovado e, então, reservou-se o laboratório onde ocorreria o curso. A proposta de curso visava os seguintes objetivos:

- Estimular o pensar dos alunos em soluções paralelas;
- Introduzir aspectos práticos da programação concorrente com *threads*;
- Permitir que alunos de começo de curso, mesmo sem conhecimentos básicos de arquitetura de computadores, sistemas operacionais e redes de computadores, possam desenvolver algoritmos paralelos;
- Introduzir conceitos práticos de avaliação de desempenho através de soluções paralelas;

As apresentações de *slides* para cada aula foram preparadas cuidadosamente buscando obter clareza, sequência e boa didática para a transmissão do conteúdo. Ainda, foram preparadas as descrições e soluções codificadas de cada exercício que seria proposto durante as aulas, além dos formulários online os quais seriam utilizados pelos alunos para submissão dos exercícios implementados em cada aula.

Na última aula do curso foi realizada uma maratona de programação paralela. Os exercícios que foram aplicados passaram por processos de análise, implementação e descrição do problema a ser implementado como etapas de preparação para a maratona. Preparou-se um guia de como deveriam ser feitas as submissões para a maratona e o material de apoio necessário para a implementação dos exercícios.

Após o planejamento das questões e a elaboração dos documentos, foram impressas cópias dos questionários de pré-teste e pós-teste suficientes para todos os alunos matriculados no curso realizarem ambos os testes.

O instrumento para análise da motivação, o CIS, foi originalmente escrito no idioma inglês. Devido a isso, uma versão em língua portuguesa do CIS foi elaborada e disponibilizada para coleta de dados via formulário online do Google, distribuído por meio de uma URL de compartilhamento.

Operação

Cada aula do experimento possuía tópicos de ensino que deviam ser alcançados com a contribuição do material previamente preparado. Em diversos momentos, buscou-se ensinar conceitos de programação sequencial com a visão de soluções paralelas. A Tabela 23 apresenta uma visão geral dos tópicos abordados durante as aulas. Ao decorrer do curso, percebeu-se que a

maior parte dos alunos já possuía algum conhecimento em programação sequencial, no entanto, outros ainda possuíam um conhecimento mínimo e, por isso, foi indispensável a realização de alguns exercícios com programação sequencial. Logo na primeira aula foi introduzido o conceito de algoritmos seguido dos aspectos paralelos de soluções algorítmicas, estimulando, desde o início, os alunos a pensarem em soluções paralelas que busquem obter maior desempenho de execução. Outros conceitos fundamentais de programação sequencial como estruturas de repetição, vetores, funções e escopo de variáveis, quando apresentados, foi possível demonstrar a perspectiva de paralelismo que poderia aplicar-se sobre estes conceitos, apresentando soluções computacionais paralelas desde o início.

Tabela 23 – Conteúdo das aulas | OpenMP 2019-1

Aulas	Tópicos abordados
Aula 1	<ul style="list-style-type: none"> - Introdução ao pensamento algorítmico; - Pensamento paralelo em soluções algorítmicas; - Desenvolvimento de código em linguagem C; e - Apresentação de algoritmos paralelos com OpenMP.
Aula 2	<ul style="list-style-type: none"> - Estruturas de decisão; - Estruturas de repetição; e - Paralelismo em estruturas de repetição com OpenMP.
Aula 3	<ul style="list-style-type: none"> - Escalonamento no paralelismo em laços de repetição; - Conceito de vetores; - Visão de paralelismo usando vetores; - Operação de Redução do OpenMP;
Aula 4	<ul style="list-style-type: none"> - Funções em C; - Escopo de variáveis; e - Execução de funções em paralelo com OpenMP (seções)
Aula 5	<ul style="list-style-type: none"> - Comunicação e sincronização entre threads; - Definição de região crítica; e - Uso de Critical e Atomic em OpenMP.
Aula 6	<ul style="list-style-type: none"> - Semáforos: lock e unlock em OpenMP.

5.6.4 Apresentação dos resultados

Caracterização dos participantes

O presente experimento visa investigar o ensino de programação paralela para alunos cursando o primeiro ano de graduação em cursos relacionados a computação. Com o intuito de obter esta população, as matrículas para participação do curso foram restringidas a alunos nesse contexto.

A amostra analisada a seguir engloba os alunos com frequência superior a 75% no curso e que realizaram ambos pré-teste e pós-teste, portanto a população analisada foi de 29 alunos. A Figura 51 exibe a porcentagem de alunos por curso de graduação que participaram do experimento. Nota-se que a maior parte dos alunos cursavam Bacharelado em Ciência da

Computação, compondo 86,2% da amostra, o que corresponde a 25 alunos. Alunos de Engenharia da Computação compõem 10,3% da amostra, correspondente a 3 alunos e houve apenas 1 aluno de Matemática Aplicada e Computação Científica, correspondente a 3,4% da amostra final.

Figura 51 – Alunos por curso - OpenMP/2019-1

Alunos por curso

Por fim, por meio do gráfico exibido na Figura 52, podemos confirmar que toda a amostra de alunos era ingressante na graduação considerando que estavam cursando o primeiro período e, supostamente, não possuem formação básica em computação anterior ao curso.

Figura 52 – Período no curso de graduação dos alunos - OpenMP/2019-1

Alunos por período

Tanto o pré-teste quanto o pós-teste tinham o objetivo de mensurar o conhecimento do ponto de vista técnico dos participantes em programação estruturada e paralela, incluindo OpenMP. Para melhor análise dos resultados, as questões foram analisadas separadamente em dois grupos: relacionadas a programação sequencial e questões sobre paralelismo. O resultado obtido da correção das questões de programação sequencial no pré e pós-teste são exibidos na Figura 53 e das questões sobre paralelismo, na Figura 54.

Figura 53 – Pré-teste x pós-teste (questões sobre programação sequencial)

Figura 54 – Pré-teste x pós-teste (questões sobre paralelismo)

Analisando a Figura 53, é possível perceber que a maioria dos alunos possuía um conhecimento satisfatório em programação sequencial, visto que tiveram um bom resultado no pré-teste, exceto alguns *outliers* demonstrados nos gráficos. No pós-teste esse conhecimento demonstrou-se consolidado, obtendo-se 100% de aproveitamento.

Na Figura 54, podemos perceber a distribuição de notas extremamente baixa, demonstrando que no momento anterior ao curso, a maioria dos alunos realmente não possuía conhecimento sobre os conceitos de paralelismo e programação paralela, exceto por alguns *outliers*, que, mesmo assim, não foram valores altamente discrepantes. Entretanto, observando os resultados do pós-teste, é possível verificar um imenso progresso nas notas, onde as notas aproximam-se da média oito. A evolução dessas notas para cada aluno é demonstrada visualmente na Figura 55. Podemos afirmar que todos os alunos demonstraram uma evolução significativa se compararmos os resultados do pré-teste com o pós-teste. Vale destacar que uma grande porção dos alunos cuja nota foi 0, em um primeiro momento, alcançaram nota 8,9 no pós-teste e dois alunos obtiveram um crescimento partindo da nota 0 no pré-teste para 10 no pós-teste.

Figura 55 – Evolução Pré-teste x pós-teste (questões sobre paralelismo)

Foi realizado o teste de Wilcoxon para amostras pareadas com o intuito de investigar se as amostras são estatisticamente equivalentes ou se possuem diferença significativa. Para isso, foram incluídas no teste todas as notas de exercícios realizados e definidas as seguintes hipóteses, onde μ_1 representa a amostra de alunos sem conhecimento prévio e μ_2 , com conhecimento prévio:

Hipótese Nula: Não há diferença estatisticamente significativa entre o número de acertos do pré-teste e do pós-teste.

$$H_0 : \mu_1 - \mu_2 = 0$$

Hipótese Alternativa: Há diferença estatisticamente significativa entre o número de acertos do pré-teste e do pós-teste.

$$H_1 : \mu_1 - \mu_2 \neq 0$$

A partir dessas hipóteses, foi obtido o resultado exibido na Tabela 24.

Tabela 24 – Teste de Wilcoxon para amostras pareadas

Estatística do Teste (Wilcoxon)	
Informações	Valores
Estatística	≈ 0
P-valor	≈ 0
Hipótese Nula	0
Limite Inferior	-4,499974936
(Pseudo) Mediana	-3,999907939
Limite Superior	-3,000032535
Nível de Confiança	0,95

Como o P-valor é aproximadamente igual a 0, rejeitamos a hipótese nula a uma significância de 95%. Assim, existem evidências suficientes para rejeitar a hipótese nula e afirmar que existe uma diferença significativa em termos do número de acertos no Pré-teste e no Pós-teste.

Além dos resultados do pré e pós-teste, foram analisadas as notas dos alunos nas atividades práticas desenvolvidas. Primeiramente, a Figura 56 exibe a distribuição de notas na segunda, terceira, quarta e quinta aula, as quais tiveram exercícios relevantes e inseridos no contexto da aula. A primeira aula foi desconsiderada por ter havido somente exemplos, mas nenhum exercício, de fato, e, na sexta aula, ocorreu a maratona de programação.

Examinando de forma mais profunda os exercícios submetidos de cada aula, temos como resultado, exibido na Tabela 25, a média das notas dos exercícios de cada aula separadas pelas métricas de avaliação. Além disso, é possível verificar a quantidade de alunos que submeteram os exercícios em cada aula.

Tabela 25 – Médias de aproveitamento de cada aula por métricas de avaliação

Média de aproveitamento de cada aula por métricas de avaliação				
	Paralelismo + legibilidade	Uso correto do modelo de programação	Saída correta	Qtde de alunos que submeteram
Aula 2	96,4%	94,7%	92,2%	32
Aula 3	97,2%	92,9%	94,3%	31
Aula 4	93,4%	96,8%	96,2%	30
Aula 5	93,0%	90,6%	95,8%	26
Média	95,0%	93,8%	94,6%	Total = 119

Com isso, a Figura 57 exibe a dispersão de dados de cada métrica de avaliação, permitindo examinar os resultados de forma conjunta e comparar as distribuições.

Figura 56 – Distribuição das notas por aula

Figura 57 – Aproveitamento por métrica no curso

Após a realização do curso, foram calculadas as médias finais de cada aluno considerando todos os exercícios que foram implementados. A Figura 58 representa a distribuição dessas médias finais dos alunos no curso, permitindo analisar, de forma geral e unificada, qual foi o desempenho deles no curso.

Por fim, os resultados do exercício extra, sugerido para que os alunos codificassem

Figura 58 – Médias finais dos alunos no curso

fora da sala de aula de forma não obrigatória, são apresentados na Figura 59. São exibidas a distribuição de notas de cada métrica de avaliação e, por último, a dispersão das notas de aproveitamento total, proveniente das métricas. Foi obtido um total de 18 submissões para esse exercício.

Figura 59 – Resultado de aproveitamento do desafio extra - cálculo paralelo do Pi

Avaliação qualitativa

A análise da motivação foi feita considerando a descrição para uso do instrumento relatada no estudo de Keller (2010). Assim, durante a análise foram consideradas as categorias que fazem parte da motivação. São elas: atenção, relevância, confiança e satisfação. A Figura 60 representa a distribuição das notas de todas as questões segundo cada categoria. Desta forma, as notas atribuídas, de acordo com a escala Likert, pelos alunos para as 34 questões foram agrupadas em suas categorias correspondentes.

Figura 60 – Distribuição de notas das categorias de motivação - OpenMP 2019-1

Também foi calculada a média geral, na mesma escala de 1 a 5, que indica o grau de motivação dos participantes de acordo com o questionário CIS. Calculando a média das médias das categorias apresentadas na Figura 60, obtemos o valor de 3,9 que retrata a motivação geral dos participantes.

5.6.5 Análise dos resultados

A partir da apresentação dos resultados, estes serão analisados em função das questões de pesquisadas elencadas inicialmente, respondendo-as de forma objetiva e completa com base nos dados técnicos e qualitativos resultantes do experimento.

QP1: Os alunos do primeiro ano foram capazes de compreender os tópicos de ensino que foram selecionados para as aulas?

Como é possível observar na Figura 56, a distribuição das notas, desconsiderando os outliers, se manteve acima de 85% de aproveitamento, tendo mais da metade dos alunos com aproveitamento acima de 90% em todas as aulas. Os valores das médias de cada aula mantêm-se muito próximos em um intervalo de 9,3 a 9,6, ou seja, com uma variância de apenas 3 décimos. Esse resultado mostra que os alunos puderam elaborar exercícios satisfatórios de acordo com o conteúdo de cada aula, mantendo constância na qualidade de desenvolvimento e aplicação

do paralelismo, e puderam compreender o conteúdo ministrado sem expressar dificuldades significativas. Complementar a isso, a Figura 58 mostra que, considerando a média final de cada aluno, o aproveitamento total no curso teve uma média de 9,4 e, desconsiderando os *outliers*, não possui médias finais abaixo de 8,5.

Esses resultados juntamente com o bom desempenho demonstrado no pós-teste servem de sustentação para afirmar que os alunos demonstraram compreender e usar corretamente os tópicos de paralelismo ministrados.

QP2: Os conceitos de paralelismo foram bem aplicados nas atividades práticas desenvolvidas pelos alunos?

Uma das métricas de avaliação das atividades era o desenvolvimento paralelo. A Figura 57 exibe a média geral de cada métrica utilizada. A primeira coluna mostra que, referente ao desenvolvimento paralelo, as médias de todos os exercícios aplicados no curso foram superiores a 92,0% de aproveitamento. A média de aproveitamento, de forma geral, foi de 95,0%. Essa média demonstra que, de forma geral, a aplicação dos conceitos de paralelismo nos códigos ao longo do curso foi ideal.

QP3: Os alunos foram capazes de usar corretamente o modelo de programação (OpenMP) nas atividades práticas?

Outra métrica de avaliação das atividades era o uso do modelo de programação. A Figura 57 exibe, na segunda caixa do gráfico, que, referente ao uso do modelo, a média de aproveitamento dos exercícios de todas as aulas foi de 93,7%. Analisando a distribuição, as médias dessa métrica em todas as aulas obteve um aproveitamento acima de 90,0%. Esses resultados mostram que, no geral, o uso de OpenMP nos códigos ao longo do curso foi adequado e satisfatório.

QP4: Os códigos desenvolvidos pelos alunos funcionavam/rodavam corretamente?

Os códigos desenvolvidos pelos alunos foram avaliados de acordo com a métrica de saída correta do exercício. A Figura 57 exibe que, referente a saída correta, a média de aproveitamento nesta métrica de todas as aulas foi de 94,6% e que as médias de todas as aulas foram superiores a 92,0%. Esses resultados mostram que, no geral, os códigos desenvolvidos ao longo do curso apresentavam a saída esperada de acordo com o exercício, ou seja, executavam corretamente.

QP5: Houve evolução dos alunos no conhecimento de paralelismo e programação paralela após a participação no curso?

As questões do pré-teste e do pós-teste exploraram conceitos de paralelismo e de programação paralela. O resultado do pós-teste aponta um visível crescimento no conhecimento de paralelismo e programação paralela. Essa evolução pode ser vista na Figura 55, que mostra, pontualmente, a diferença das notas de pré e pós-teste de cada aluno. Com isso, podemos ver que todos os alunos obtiveram grandes progressos de aprendizagem, inclusive alguns com 0,0%

de aproveitamento no pré-teste foram capazes de obter 100,0% no pós-teste, o que indica que puderam compreender o fluxo de execução de software paralelo e os principais conceitos de desenvolvimento com *threads*. É evidente que todos os alunos alcançaram uma evolução significativa do início ao final do curso, identificando que foi possível transmitir percepções de paralelismo e introduzir o pensamento paralelo para que fosse aplicado de forma prática em soluções com OpenMP.

QP6: Os materiais e conteúdos abordados no curso foram suficientes, insuficientes ou excessivos?

A evolução demonstrada pelos alunos no pós-teste, o bom desempenho na resolução de exercícios durante as aulas, a implementação correta do desafio extra por alguns alunos e o resultado positivo de satisfação dos alunos com o curso demonstra que os tópicos de ensino selecionados foram suficientes e não ultrapassaram as limitações impostas pela ausência de formação prévia.

Esse resultado mostra que os tópicos da camada de conceito de Programação, descrita na Seção 4.4 podem ser compreensíveis por alunos já no primeiro ano de graduação, e que o enfoque nesta camada pode colaborar para o aprendizado do aluno, não implicando em limitações ao ensino.

Além disso, as respostas das questões da avaliação de motivação demonstradas nas Figuras 61 e 62 reforçam que a maioria dos participantes acredita que a dificuldade, ou nível de desafio, do curso foi razoável e que a quantidade de exercícios propostos foram apropriadas para o curso.

Figura 61 – Questão 30 do CIS

30. Eu acho que o nível de desafio neste curso é o correto: nem muito fácil nem muito difícil.

29 respostas

QP7: Os alunos demonstraram motivação para com o tema ensinado?

Figura 62 – Questão 31 do CIS

33. A quantidade de atividades que tenho que fazer no curso é apropriada para esse tipo de curso.

29 respostas

Analisando o gráfico na Figura 60, observa-se que as médias de cada categoria possuem variância de apenas 3 décimos, todas situando-se entre o intervalo de 3,8 e 4,1. Por esses valores, verificamos que a concentração de notas permaneceu alta em todas as categorias. Ademais, os valores das medianas são equivalentes em todas as categorias, ou seja, são iguais a quatro. Esses resultados indicam que 50% dos valores encontram-se superiores ou iguais a 4, ou seja, apesar da distribuição possuir valor mínimo 1, indicando que um ou mais participantes não expressavam satisfação em algumas questões, pelo menos metade das opções selecionadas foram 4 ou 5, valores máximos da escala, sendo essa análise válida para as áreas de atenção, relevância, confiança e satisfação dos alunos referente ao curso ministrado.

Após a análise individual das categorias, podemos analisar a motivação de forma geral a partir do gráfico na Figura 63. A média geral de motivação incluindo todas as questões da pesquisa foi de 3,9. Esse resultado aponta para uma grande motivação geral com o curso. Era esperado que nem todos os alunos estariam completamente satisfeitos com todas as questões, por isso houve a atribuição do valor mínimo 1 por um ou mais alunos em algumas questões. No entanto, da mesma forma ocorrida na análise individual de cada categoria, o valor da mediana indica que pelo menos 50% das notas dadas foram 4,0 ou 5,0 e 75% acima de 3,0.

Ainda, além do resultado demonstrado pela pesquisa de interesse de curso, o fato de mais da metade da população de alunos implementarem o desafio extra-classe, cientes de que o mesmo não contribuiria para a nota final do curso, demonstra que os alunos possuíam interesse no assunto e estavam motivados em aprender.

Mediante a apresentação dos resultados acima, podemos concluir que o ensino do conteúdo de paralelismo para alunos ingressante na graduação e sem base nos conceitos fundamentais de computação, causou motivação para com o tema ministrado de acordo com as métricas do

Figura 63 – Resultado geral da análise de motivação

modelo ARCS. Portanto, em geral, os participantes indicaram estar atentos às aulas, entender que o conteúdo ministrado é relevante para eles, sentir confiança no seu entendimento e estarem satisfeitos com a aprendizagem no curso.

5.7 Experimento V: Curso de Difusão Cultural de Introdução ao *OpenMP* baseado em *Problem Based Learning*.

O experimento descrito nessa Seção foi planejado e dirigido por um membro do mesmo laboratório de pesquisa baseado na proposta de conteúdo similar ao Experimento IV, ou seja, fundamentado nas camadas de conceitos estabelecidas na Seção 4.4 deste projeto, entretanto no contexto de uma metodologia de ensino ativa, *Problem Based Learning* (PBL). O experimento busca avaliar, em aspectos práticos, o ensino baseado em desafios de programação em cada aula auxiliado por um *software* de maratona de programação paralela para alunos com e sem conhecimento preliminar em computação. Portanto, o desenvolvimento e preparação das aulas e desafios propostos têm como base os conceitos determinados na Seção 4.5 deste projeto.

5.7.1 Escopo

Objetivo

Por meio deste experimento, busca-se estudar o processo pedagógico em um curso de programação paralela totalmente baseado em desafios de programação, independente do estágio de formação dos estudantes. Pretende-se estabelecer uma relação de causa-efeito entre a metodologia e recursos utilizados e o aprendizado e motivação dos estudantes.

Objeto

O objeto de estudo deste experimento constitui-se no processo de aprendizagem de discentes de diferentes cursos e períodos da graduação e os resultados de ensino obtido com a aplicação da metodologia do *Problem Based Learning* para alunos com e sem conhecimentos prévios relevantes em computação.

Foco qualitativo

Buscamos, com este experimento, analisar o conhecimento teórico e prático de programação paralela demonstrado pelos alunos na resolução dos desafios. Além disso, deseja-se mensurar, por meio do modelo ARCS, a motivação dos alunos no curso.

Perspectiva

Em função da realização de desafios de programação, a perspectiva deste experimento é sua execução em um ambiente computacional adequado, contemplado pelos recursos de *hardware* e *software* necessários para o desenvolvimento e execução dos desafios.

Contexto

Assim, projetou-se um curso de difusão cultural oferecendo inscrições gratuitas para a comunidade. Tal curso ocorreu no mês de junho em 2019 no Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo em uma sala de aula que fornecia a infraestrutura requerida para a execução do curso. A duração total do curso foi de 12 horas disposta em 4 aulas de 3 horas. O conteúdo estudado em cada aula correspondia ao que foi proposto e elencado na Seção 4.5 deste projeto.

5.7.2 Planejamento

Definição das questões de pesquisa

- QP1:** Os estudantes puderam compreender os tópicos de ensino que foram selecionados para as aulas?
- QP2:** Os códigos desenvolvidos pelos estudantes em *OpenMP* possuíam qualidade levando em consideração a aplicação adequada do paralelismo e modelo de programação para melhorar o desempenho?; e
- QP3:** Os estudantes sentiram-se motivados e receptivos em relação ao aprendizado dos conceitos propostos?

Seleção de sujeitos

Os sujeitos foram escolhidos de forma randômica, visto que a matrícula foi aberta à comunidade, ainda que houve a preferência pela inscrição de alunos sem conhecimento prévio em programação paralela, estágio inicial da graduação ou cursando graduação/pós em cursos não relacionados à computação.

Descrição de instrumentação

A metodologia empregada no experimento deu-se pelo Aprendizado Baseado em Problemas, ou *Problem Based Learning* (PBL). Nesta metodologia, apresenta-se o conteúdo teórico como método para a resolução para os desafios apresentados na aula. As maratonas de programação foram realizadas em todas as aulas e com o apoio de um sistema específico para maratonas, *Mooshak 2* (<<https://mooshak2.dcc.fc.up.pt/>>). O módulo final consistiu na realização de uma maratona de programação paralela que incluiu a entrega de prêmios para as equipes com aproveitamento maior.

O *software* de maratona, *Mooshak*, foi executado no *Google cloud* por meio de uma máquina virtual. Os materiais de suporte para conteúdo diretrizes para instalar e utilizar o software está disponível em <http://bit.do/material_de_apoio>.

Ao todo, foram propostos dezesseis desafios durante o curso. Os códigos em suas versões sequenciais e paralelas podem ser encontradas em <http://bit.do/desafios_omp_2019_2>.

Além disso, foram elaborados cadernos contendo o detalhamento de cada desafio de programação a ser implementado, juntamente com o conteúdo teórico necessário para a solução dos algoritmos e, finalmente, as implementações sequenciais dos desafios. Foi fornecido um caderno impresso e online para cada equipe no começo das aulas.

Os documentos desenvolvidos para utilização neste experimento estão disponíveis em <http://bit.do/caderno_desafios_curso>.

Planejamento de curso

A seguir, na Tabela 26, encontram-se os assuntos lecionados e conceitos relacionados.

Tabela 26 – Planejamento de curso: sexto experimento.

Aula	Conteúdo
1	Desafios de programação Diretiva <code>parallel</code> Escopo de variáveis (<code>shared</code> , <code>private</code> , <code>firstprivate</code> , <code>lastprivate</code> , <code>threadprivate</code> , <code>copyin</code>)
2	Desafios de programação Diretiva <code>for</code> (loops paralelos) Redução (diretiva <code>reduction</code>) Escalonamento (<code>static</code> , <code>dynamic</code> , <code>guided</code> , <code>runtime</code> , <code>auto</code>)
3	Desafios de programação Seções (<code>sections</code>) Sincronização (<code>nowait</code> , <code>critical</code> , <code>atomic</code> , <code>barrier</code> , <code>flush</code>)
4	Maratona de programação paralela

5.7.3 Execução

Preparação

Após a definição do experimento e respectivo planeamento, foi submetido o pedido para realização do curso de difusão cultural no ICMC com carga horária e tema propostos nesse experimento. O pedido foi aprovado e, então, reservou-se o laboratório onde ocorreria o curso.

Em seguida, foi feita a escolha dos desafios que seriam aplicados em cada aula e, com isso, as descrições e implementações tanto estruturada como paralela de cada um. A infraestrutura na nuvem do *Google* foi configurada corretamente para utilização do *Mooshak*. Os computadores da sala de aula foram verificados, garantindo o desenvolvimento e execução em ambiente local dos desafios, além de permitir a submissão remota para o *software* de maratonas. Foram adquiridos, anteriormente a aula, a premiação que seria entregue aos alunos das equipes com maiores pontuações.

Operação

No primeiro dia do curso, os alunos foram convidados a formar grupos de até 3 membros, a se inscreverem na plataforma e a se agruparem em diferentes lugares da classe. Em seguida, os alunos receberam a impressão, assim como o acesso à alternativa *online*, do caderno com o detalhamento e versão estruturada dos exercícios propostos para a turma, bem como uma breve apresentação do conteúdo teórico.

No contexto de cada aula, os professores faziam uma breve apresentação do conteúdo e o projetor era utilizado para exibir as notas da equipe, enquanto os professores respondiam dúvidas dos estudantes e gerenciavam o sistema de maratonas.

Ainda, as implementações enviadas manualmente examinadas e os times melhor classificados foram anunciados, recebendo a premiação ao final de cada aula.

5.7.4 Apresentação dos resultados

Caracterização dos participantes

A amostra participante do curso foi composta de 10 alunos com e 23 alunos sem conhecimentos prévios em computação, dos quais 11 cursavam o primeiro semestre de graduação. A quantidade de alunos por grau de formação é demonstrada na Figura 64.

Figura 65 – Porcentagem de acertos, erros e "não sei" dos pré e pós-testes

Figura 66 – Comparação entre as médias do pré-teste e pós-teste

- $p\text{-valor} = 6,091 \cdot 10^{-5}$, $p\text{-valor} < 0,05$. Conclusão: Rejeita-se h_0 , mostrando que os estudantes tiveram melhores resultados no pós-teste.

Foram aplicados 09 desafios aos alunos durante o curso, sendo que os desafios de cada módulo precisavam ser resolvidos com os conceitos e recursos ensinados no respectivo módulo de ensino. Além destes desafios, foi realizada uma maratona de programação paralela no último módulo do curso com mais 07 desafios, totalizando 16 desafios ao todo. Os alunos agruparam-se em equipes de desenvolvimento para a solução dos desafios. A Figura 67 mostra a média das avaliações feitas sobre os 16 desafios propostos por equipe, considerando scores em uma escala entre 0 e 10. Os resultados mostram que 81,3% das equipes tiveram scores acima da média da

turma de 9,3. Apenas 03 equipes (18.7%) ficaram abaixo da média da turma.

Figura 67 – Médias finais por equipe considerando os desafios de programação paralela

No geral, percebe-se que os alunos implementarão soluções corretas que apresentam qualidade. Somente quatro times obtiveram médias menores que 85%.

Além disso, observando a Figura 68, pode-se verificar que, em geral, os participantes tiveram bons resultados de desempenho nos exercícios sugeridos, dos quais em somente 5 desafios a média de todas as equipes apresentou-se menor que 85%. Vale destacar que as notas correspondem ao grau de dificuldade dos desafios ao longo do curso, assim, nos últimos exercícios, e consequentemente mais complexos, a qualidade do código produzido, no geral, foi menor.

Figura 68 – Médias de qualidade de código por desafio

Avaliação qualitativa

Para medir a receptividade dos alunos, utilizamos um questionário de pesquisa fundamentado no modelo ARCS chamado de Pesquisa de Interesse de Curso definido em Keller (2010). As questões referentes a esta pesquisa podem ser visualizadas em <http://bit.do/qualitativo_openmp_2>.

Dezenove participantes submeteram suas respostas. Os diagramas representando todas as respostas referente a cada uma das 34 questões estão disponíveis para acesso em <http://bit.do/graficos_qualitativo_openmp>.

A fim de resumir as respostas recebidas, foi elaborado um gráfico na escala *likert*, considerando as 4 áreas avaliadas pelo modelo ARCS: Atenção, relevância, confiança e satisfação. O gráfico é demonstrado na Figura 69. Pode-se perceber pela análise que a avaliação do curso, em geral, foi favorável, resultando em uma satisfação média total de 83%.

Figura 69 – Distribuição de notas das categorias de motivação - OpenMP 2019-2

5.7.5 Análise dos resultados

Em função das questões de pesquisa definidas para este experimento, apresenta-se as análises dos resultados obtidos, visando responde-las.

QP1: Os estudantes puderam compreender os tópicos de ensino que foram selecionados para as aulas?

Como pode ser visto por meio da avaliação da Figura 66, e da análise estatística correspondente, os alunos participantes deste experimentos obtiveram médias significativamente superiores no pós-teste quando comparadas com o pré-teste, ilustrando a aprendizagem do conteúdo relacionado a teoria que foi ministrado. A porcentagem de acertos mensurado foi de

70,4% no pós-teste e 5,7% no pré-teste. Ainda, a porcentagem de seleção da alternativa que sinalizava que os participantes não tinham conhecimento da resposta correta foi reduzida de 81,9% para 8,5% no pré e pós-teste, respectivamente.

Dessa forma, mediante estes resultados, podemos dizer que ocorreu uma aprendizagem considerável referente ao tema alvo do experimento e que os tópicos de ensino puderam ser compreendidos ao decorrer do curso.

QP2: Os códigos desenvolvidos pelos estudantes em *OpenMP* possuíam qualidade levando em consideração a aplicação adequada do paralelismo e modelo de programação para melhorar o desempenho?

A nota média relativa a qualidade de código dessa turma foi de 85%. Além disso, avaliou-se a qualidade de código por equipes, ilustrada na Figura 67, e, na maior parte dos desafios, a qualidade média de código encontra-se equivalente ou excedente a 93%, como pode ser observado na Figura 68. Além disso, considerando as limitações de tempo, os times puderam implementar 74% do total de desafios do curso, demonstrando o interesse por parte dos alunos no tema e desenvolvimentos propostos.

Assim, foi possível notar que os estudantes puderam implementar programas corretos e com qualidade satisfatória considerando o uso de paralelismo.

QP3: Os estudantes sentiram-se motivados e receptivos em relação ao aprendizado dos conceitos propostos?

Em função das respostas da pesquisa de interesse, pode-se constatar que o nível de satisfação dos alunos participantes do experimento foi relevante. A Figura 69 indica que houve um grau de motivação satisfatório e uma satisfação média de 83%.

Foi possível também notar que os alunos se sentiram motivados e receptivos ao conteúdo, temas de programação paralela, metodologia de ensino, recursos educacionais, didática e outras ferramentas pedagógicas utilizadas neste curso.

5.8 Análise dos resultados dos experimentos

Com o intuito de validar a proposta deste projeto, projetamos e executamos 05 experimentos com 243 alunos ao todo para o ensino de programação paralela. Foram criadas 08 turmas para o ensino de programação paralela com enfoque prático de desenvolvimento de algoritmos, porém, sem esquecer da fundamentação teórica para tanto. Sete turmas consideraram o ensino de OpenMP e uma turma o ensino de Pthreads, usando diferentes abordagens de ensino como tradicional, PBL (*Problem Based Learning*) e TBL (*Team Based Learning*).

Analizando o curso de graduação e estágio de formação, os estudantes, em todos os experimentos, foram classificados em dois grupos distintos: com conhecimento prévio e sem conhecimento prévio em computação. No total, considerando a população presente em todos

os experimentos realizados, obtivemos uma amostra de 164 estudantes no primeiro grupo e 88 no segundo. O foco dos experimentos foi analisar se alunos sem conhecimentos prévios de computação poderiam aprender programação paralela no mesmo nível de aprendizado verificado por alunos que já possuem conhecimentos prévios como programação sequencial, arquitetura de computadores e sistemas operacionais. Do ponto de vista técnico os estudantes foram avaliados por exercícios correspondentes ao tópico de ensino e o modelo de programação proposto no curso.

A Tabela 27 apresenta conjuntamente os percentuais de aproveitamento médio dos algoritmos paralelos desenvolvidos pelos alunos em todos os experimentos separados pelos grupos de alunos mencionados. As notas de todas as atividades práticas foram compostas pelas métricas de avaliação, definidas na Seção 5.2, considerando os respectivos pesos atribuídos à média ponderada. Ao final da Tabela é possível verificar a média geral de aproveitamento em todos os experimentos em ambos os grupos e especificamente dos experimentos II, III e V, onde houve a participação de alunos com e sem conhecimentos prévios nas mesmas aulas.

Tabela 27 – Aproveitamento de aprendizagem geral dos alunos nos experimentos realizados.

Formato	Experimento	Aproveitamento	
		Alunos com conhecimento prévio	Alunos sem conhecimento prévio
Disciplina	Programação Paralela BCC Turma 1 (E-I)	83,0%	-
	Programação Paralela BCC Turma 2 (E-I)	79,8%	-
	Programação Paralela Engenharia da Computação (E-I)	84,5%	-
	Programação Paralela Pós-graduação (E-I)	80,0%	-
Curso de extensão	Curso de Difusão Cultural de Introdução ao OpenMP (E-II)	93,8%	88,9%
	Curso de Difusão Cultural de Introdução a Pthreads (E-III)	98,8%	94,6%
	Curso de Difusão Cultural de Introdução ao OpenMP para alunos iniciantes em computação (E-IV)	-	94,1%
	Curso de Difusão Cultural de Introdução ao OpenMP baseado em Problem Based Learning (E-V)	95,4%	95,8%
	Média de todos os experimentos:	87,9%	93,3%
	Média dos experimentos II, III e V:	96,0%	93,1%

Analisando os experimentos podemos ver que o método, os materiais teóricos e práticos, o modelo de aula e a abordagem proposta para os cursos foram satisfatórios para um bom resultado de aprendizado de alunos experientes e não experientes em relação ao conhecimento em computação. De forma geral, os alunos sem experiência conseguiram desenvolver programas paralelos e compreender o funcionamento e execução de *threads* concorrentes.

Foi realizado o teste de hipóteses de Wilcoxon para amostras independentes com o intuito de investigar se o aproveitamento das amostras nos experimentos são estatisticamente equivalentes ou se possuem diferença significativa. Para isso, foram incluídas no teste todas as notas de exercícios realizados nos cursos e definidas as seguintes hipóteses, onde μ_1 representa a amostra de alunos sem conhecimento prévio e μ_2 , com conhecimento prévio:

Hipótese Nula: Não existe diferença significativa no aproveitamento total entre os grupos de alunos com e sem conhecimento prévio.

$$H_0 : \mu_1 - \mu_2 = 0$$

Hipótese Alternativa: O aproveitamento total dos alunos sem conhecimento prévio nos experimentos possui valor significativamente maior.

$$H_1 : \mu_1 - \mu_2 > 0$$

A partir dessas hipóteses, foi obtido o resultado exibido na Tabela 28.

Tabela 28 – Teste de Wilcoxon: média de todos os experimentos

Estatística do Teste (Wilcoxon)	
Informações	Valores
Estatística	10302
P-valor	≈ 0
Hipótese Nula	0
Limite Inferior	0,600015184
(Pseudo) Mediana	0,799950148
Limite Superior	Inf
Nível de Confiança	0,95

Sendo o P-valor aproximadamente igual a 0, rejeitamos a hipótese nula a uma significância de 95%. Dessa forma, podemos afirmar que existem evidências suficientes para rejeitar a hipótese nula e afirmar que o aproveitamento total dos alunos sem conhecimento prévio nos experimentos possui valor significativamente maior que os alunos com conhecimento prévio.

Nota-se que a média de aproveitamento das disciplinas de graduação (Experimento I) são menores em todas as turmas se comparadas com as médias obtidas nos cursos de extensão em que houve participação de alunos sem conhecimento prévio em computação, inclusive no Experimento IV, o qual obteve participação unicamente de estudantes do primeiro ano da graduação. Acredita-se que podem haver três possíveis fatores para este resultado: 1) os alunos das disciplinas obrigatórias em períodos mais avançados da graduação possuíam menos interesse no tema do que os alunos em cursos de extensão, nos quais participaram alunos realmente interessados no aprendizado; 2) diferentes metodologias de ensino podem ter influenciado no resultado, pois nas disciplinas obrigatórias do Experimento I as aulas foram teóricas, enquanto que as aulas nos cursos de extensão foram em laboratório, focadas no desenvolvimento de código; e 3) os alunos estão mais engajados em aprender programação paralela no início da graduação do que em semestres mais avançados. Por este motivo, também examinamos e comparamos

estatisticamente as médias obtidas somente nos experimentos II, III e V, nos quais houve a participação de alunos com e sem conhecimentos prévios nas mesmas aulas.

As notas dos experimentos II, III e V mostram que os os alunos com conhecimento prévio obtiveram uma média de 96.0% de aproveitamento, enquanto que os alunos sem conhecimento obtiveram uma média de 93.1%. Foi realizado o teste de hipóteses de Wilcoxon para amostras independentes comparando se elas possuem diferença estatisticamente significativa, caracterizando as seguintes hipóteses:

Hipótese Nula: Não existe diferença significativa no aproveitamento total entre os grupos de alunos com e sem conhecimento prévio.

$$H_0 : \mu_1 - \mu_2 = 0$$

Hipótese Alternativa: O aproveitamento total dos alunos sem conhecimento prévio nos experimentos possui uma diferença significativa.

$$H_1 : \mu_1 - \mu_2 \neq 0$$

A Tabela 29 exibe os resultados para o teste de Wilcoxon realizado.

Tabela 29 – Teste de Wilcoxon: média dos experimentos I, II e V

Estatística do Teste (Wilcoxon)	
Informações	Valores
Estatística	724,5
P-valor	0,383
Hipótese Nula	0
Limite Inferior	-0,099962596
(Pseudo) Mediana	0,099974759
Limite Superior	0,200008129
Nível de Confiança	0,95

O teste para estas amostras mostram $P = 0.383$. Portanto, aceitamos a hipótese nula a uma significância de 95%. Dessa forma, podemos afirmar que existem evidências suficientes para aceitar a hipótese nula e afirmar que o aproveitamento em ambos os grupos são estatisticamente equivalentes.

Além das comparações realizadas anteriormente, destaca-se o quarto experimento, "Curso de Difusão Cultural de Introdução ao OpenMP para alunos iniciantes em computação", descrito na Seção 5.6, o qual foi possível obter uma população integralmente composta por alunos do primeiro semestre de cursos de computação. Os alunos foram capazes de aprender programação paralela ao mesmo tempo que compreendiam os conceitos básicos de programação sequencial. Os estudantes puderam desenvolver códigos que fazem uso de *threads* executadas em paralelo e medir o desempenho de execução obtido em seus programas, apesar de não conhecerem os detalhes operacionais e técnicos de hardware, memória, avaliação de desempenho ou sistemas operacionais. Dessa maneira, os alunos que participaram do experimento poderão, desde o início

de seu curso, pensar em soluções algorítmicas paralelas e explorar, na prática, o desempenho obtido por sua aplicação, diferentemente de alunos regulares que, provavelmente, irão obter o primeiro contato prático com paralelismo somente em outras disciplinas dispostas em anos mais avançados da graduação de programação concorrente. Observa-se que a porcentagem média de aproveitamento no grupo de alunos deste experimento (do primeiro semestre de graduação) apresenta-se maior se comparada com as médias das 4 turmas da disciplina ocorrida regularmente no sexto semestre, com alunos não iniciantes em computação, tratando-se do ensino do mesmo conteúdo (OpenMP).

Esse resultado, somado com as comparações realizadas nos demais experimentos, demonstra que os alunos sem conhecimento prévio em computação podem alcançar um aproveitamento de aprendizado na mesma proporção ou maior que os alunos que possuem conhecimento prévio.

O quinto experimento, "Curso de Difusão Cultural de Introdução ao OpenMP baseado em *Problem Based Learning*" (seção 5.7), foi planejado, elaborado, conduzido e avaliado com a participação de outro integrante do laboratório de pesquisa de forma independente, adotando, inclusive, uma metodologia de ensino diferente (*Problem Based Learning*). O experimento também apresentou resultados satisfatórios no processo de ensino-aprendizagem, conforme descritos na Seção 5.7.4. Os resultados dos alunos sem conhecimentos prévios foram considerados bons (acima de 90% de aproveitamento).

Analisando as pesquisas qualitativas realizadas nos cursos, verificamos que os participantes não demonstraram problemas à antecipação do ensino da programação paralela e, de forma geral, foram receptivos ao aprendizado. Esse resultado é essencial principalmente considerando os alunos sem formação prévia em computação, mostrando que o ensino de programação paralela pode ocorrer independentemente dos obstáculos impostos pela pouca experiência em computação. A análise de motivação segundo o modelo ARCS somente com discentes do primeiro ano, no Experimento IV (seção 5.6), intensifica a concepção de que o conteúdo de programação paralela possa ser antecipado para estudantes neste contexto e fornecer bons resultados de aprendizagem, sem bloqueio ou oposição por parte dos alunos.

CONCLUSÕES

6.1 Considerações iniciais

Este capítulo apresenta os principais resultados obtidos com a realização deste projeto de mestrado, destacando as principais contribuições científicas, tecnológicas e de formação de recursos humanos abrangidas por este trabalho. Este capítulo também descreve as limitações desta pesquisa, as quais devem ser levadas em consideração durante a interpretação dos resultados obtidos. As produções científicas geradas como fruto dos resultados desta pesquisa também são apresentadas. Por fim, com o intuito de estimular a continuidade do trabalho, são apresentadas sugestões para o desdobramento desta pesquisa.

6.2 Caracterização da Pesquisa Realizada e seus Resultados

O principal foco do presente projeto foi investigar o processo de ensino-aprendizagem de programação paralela para alunos que não possuam formação prévia em computação, abrangendo discentes ingressantes em cursos superiores de computação, ou seja, no primeiro ano de graduação e de outras áreas de conhecimento fora da computação. O principal desafio advém do fato deste assunto possuir diversos aspectos dependentes de outras áreas da computação e, por isso, é usualmente estudado em estágios mais avançados da graduação em computação.

Visando alcançar os objetivos do projeto, realizamos um estudo sobre os conceitos considerados como essenciais no ensino de programação paralela pela comunidade internacional e que possuam embasamento teórico acadêmico. Com isso, elencamos os tópicos determinados pela ACM e IEEE em diretrizes curriculares para o ensino de programação paralela, os quais foram a base para o desenvolvimento deste projeto. Propusemos uma iniciativa de categorização desses tópicos em camadas de conceitos relativos a um mesmo tema (Organização e arquitetura,

Sistema, Modelo Computacional, Programação e Aplicação). Tais camadas visam introduzir uma possibilidade de estudo direcionado de acordo com a visão do nível de paralelismo de cada camada. Baseado nisso, foi possível sugerir escopos que incluíam apenas conceitos de uma mesma categoria.

A partir disso, foram planejados experimentos para validar o ensino dos conceitos especificamente da camada de Programação para alunos iniciantes em computação. Foram executados 5 experimentos incluindo estudantes que possuíam alguma formação prévia em computação e que não possuíam tal formação ou conhecimento prévio na área. Os resultados provenientes desses experimentos permitiram avaliar e comparar o desempenho dos participantes em ambos os grupos.

Os nossos resultados mostraram que os alunos participantes dos experimentos foram capazes de aprender programação paralela desde o primeiro ano dos cursos de graduação em computação. Estes alunos foram capazes de aprender conceitos essenciais e práticos de paralelismo, estimulando-os ao desenvolvimento de soluções com melhores desempenho e qualidade de código. Os resultados dos experimentos mostram altas pontuações em médias finais dos cursos, tanto para alunos com conhecimentos prévios quanto alunos sem conhecimentos prévios. Tais resultados mostraram que os alunos participantes dos experimentos que eram iniciantes em computação ou eram de outros cursos tiveram aproveitamentos análogos aos observados em alunos mais maduros em cursos de ciências da computação, estes últimos com um *background* maior.

Concluímos, com base nos nossos resultados, que a falta de base teórica em computação não foi um fator impeditivo para a antecipação do ensino de programação paralela nos nossos experimentos e que foi possível despertar o pensamento paralelo naqueles alunos, nosso futuros profissionais da computação. Esta antecipação foi considerada pelos alunos dos nossos experimentos como um aspecto motivador ao desenvolvimento de soluções com foco em qualidade e desempenho.

6.3 Principais Contribuições

Como as contribuições mais relevantes deste projeto podemos citar em ordem de prioridade:

- A falta de base teórica em computação não é um fator impeditivo para a antecipação do ensino de programação paralela e a possibilidade de despertar o pensamento paralelo em jovens, futuros profissionais da computação é, não apenas possível, mas também um forte motivador a se desenvolver soluções com foco em qualidade e desempenho.
- Apresentar programação para os alunos no início da graduação desperta a curiosidade dos mesmos no tema, incentivando-os a buscarem conhecimento e estimulando a pesquisa na

área.

- Diferentes metodologias de ensino podem ser usadas nas aulas de Programação Paralela. O uso de TBL neste contexto, por exemplo, não foi encontrado na literatura.
- Implantação da metodologia *Team Based Learning* em aulas de programação paralela, algo que não foi identificado no levantamento bibliográfico;
- Análise introdutória sobre uma organização do conteúdo ministrado de programação paralela em camadas de conceitos, possibilitando o estudo objetivo em diferentes níveis.
- Desenvolvimento de Mapas Conceituais que esquematizam os tópicos compreendidos na iniciativa curricular da IEEE (PRASAD *et al.*, 2012), permitindo uma visualização estruturada dos tópicos.
- Mapeamento sistemático sobre as principais técnicas de ensino e aprendizado, além dos recursos educacionais, usados no ensino de computação e programação paralela;
- O presente trabalho abre possibilidades novas de pesquisa em termos de metodologias de ensino para a programação paralela, uso da programação paralela em outras disciplinas da computação como Estrutura de dados, Banco de Dados, Processamento de Imagens, Inteligência Artificial etc.

6.4 Limitações

O presente estudo possui algumas limitações que caracterizam ameaças à validade, descritas a seguir:

- Subjetividade das correções e notas atribuídas: considerando a natureza deste trabalho, é razoável destacar que as notas atribuídas para as atividades, ainda que provenientes de verificação por diferentes avaliadores, pode acarretar em subjetividade devida às diferentes visões, conceitos e julgamento de cada um dos pesquisadores. Visando reduzir o impacto destes elementos, padronizamos os critérios de avaliação que permitiram uma comparação mais uniforme;
- Classificação do conhecimento prévio em computação: apesar de considerarmos o conhecimento prévio em computação dos alunos pelo estágio da formação, é possível que, devido à aleatoriedade da amostra de estudantes nos experimentos, um aluno, ainda que no primeiro ano de graduação, possa ter experiência prévia com computação ou programação proveniente de formações anteriores. Dessa forma, é desejável que sejam executados experimentos semelhantes aos desta dissertação possuindo uma amostra controlada de estudantes que permita garantir a ausência de conhecimento prévio em computação ou programação.

- Representatividade das amostras: As amostras, ainda que compostas de uma quantidade considerável de alunos, podem não ser representativas dentro da população de todos os alunos iniciantes em computação, visto que todos os sujeitos selecionados eram alunos da USP ou de outras instituições localizadas em São Carlos, São Paulo. Por isso, é desejável que os experimentos sejam replicados em outras instituições e contextos para garantir uma maior representatividade da amostra;
- Seleção de sujeitos: Embora nos experimentos realizados tenha-se preferido a inscrição de indivíduos sem conhecimento prévio em programação paralela, em apenas um experimento foi possível constituir uma turma composta apenas por alunos com as características desejadas. Ainda sim, isso possibilitou expandir os experimentos para estudantes de outros cursos de graduação e pós graduação.
- Impacto da metodologia e dos recursos utilizados: Embora este trabalho tenha tido sucesso em comprovar que estudantes sem conhecimento prévio são capazes de aprender os conceitos propostos e desenvolver soluções paralelas com boa qualidade, não foi analisado de forma profunda o impacto das diferentes metodologias e recursos educacionais utilizados sobre a qualidade do aprendizado. Desta forma, é necessário estabelecer ainda quais recursos pedagógicos podem ser melhor aplicados no contexto dessa dissertação, de forma a maximizar os resultados obtidos.

6.5 Produção Científica

Com o intuito de divulgar os resultados obtido neste projeto, foram submetidos 3 artigos completos para a conferência *Frontiers in Education* (FIE) em 2020, intitulados:

1. *Teaching Parallel Programming for Beginners in Computer Science;*
2. *Learning Parallel Programming Through Programming Challenges;* e
3. *Research on Parallel Computing Teaching: state of the art and future directions.*

Os dois primeiros artigos descrevem os experimentos e os analisam a partir de pontos de vistas diferentes: 1) o ensino de programação paralela para alunos sem formação prévia em computação e 2) o ensino de programação paralela por meio de desafios de programação paralela. O terceiro descreve o estado da arte com as etapas realizadas por este mapeamento sistemático e resultados apresentados. Tais artigos tiveram os *abstracts* submetidos previamente e foram aprovados pelos avaliadores da conferência.

6.6 Trabalhos Futuros

Considerando as contribuições do presente projeto, relacionamos possibilidades de estudos para serem realizados futuramente no mesmo âmbito:

- Investigar as diferenças do ensino de programação paralela usando metodologias de ensino distintas, principalmente a metodologia tradicional e metodologias ativas de ensino.
- Investigar problemas reais da indústria e transcrevê-los em atividades práticas para serem utilizadas como exercícios no ensino de programação paralela, visando estimular a utilização de paralelismo em problemas do “mundo real”.
- Estudar qual é o impacto durante toda a graduação de aprender programação paralela desde o início do curso. Qual o efeito em outras disciplinas posteriores?

REFERÊNCIAS

- ACACIO, M. E.; CUENCA, J.; FERNÁNDEZ, L.; FERNÁNDEZ-PASCUAL, R.; CERVERA, J.; GIMÉNEZ, D.; GARRIDO, M. C.; LAGUNA, J. A. S.; GUILLÉN, J.; BENITO, J. A. P.; REQUENA, M.-E. An experience of early initiation to parallelism in the computing engineering degree at the University of Murcia, Spain. In: **Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2012**. [S.l.: s.n.], 2012. p. 1289–1294. Citado na página 57.
- ADAMS, J. C. Injecting parallel computing into cs2. In: **Proceedings of the 45th ACM Technical Symposium on Computer Science Education**. New York, NY, USA: ACM, 2014. (SIGCSE '14), p. 277–282. ISBN 978-1-4503-2605-6. Disponível em: <<http://doi.acm.org/10.1145/2538862.2538883>>. Citado na página 60.
- ALMASI, G. S.; GOTTLIEB, A. **Highly Parallel Computing**. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1989. Citado na página 26.
- ANTHONY, E. M. Approach, method, and technique. **ELT Journal**, XVII, n. 2, p. 63–67, 1963. Citado na página 31.
- ARMITAGE, P.; BERRY, G.; MATTHEWS, J. N. S. **Statistical methods in medical research**. [S.l.]: John Wiley & Sons, 2008. Citado na página 76.
- BACHIEGA, N. G.; SOUZA, P. S. L.; BRUSCHI, S.; DO, S.; SOUZA, R. S. D. Mapeamento sistemático do ensino teórico e prático de programação paralela. In: **VI Congresso Brasileiro de Informática na Educação**. [s.n.], 2017. p. 1089–1098. Disponível em: <<http://www.br-ie.org/pub/index.php/wcbie/article/viewFile/7498/5293>>. Citado nas páginas 26 e 41.
- BARKER, A. E. Remembering reg revans: Action learning's principal pioneer. In: _____. **Action Learning: History and Evolution**. London: Palgrave Macmillan UK, 2010. p. 29–47. ISBN 978-0-230-25073-4. Disponível em: <https://doi.org/10.1057/9780230250734_2>. Citado na página 34.
- BI, Y.; BEIDLER, J. A visual tool for teaching multithreading in java. **Journal of Computing Sciences in Colleges**, v. 22, p. 156–163, 06 2007. Citado na página 26.
- BLOOM, B. S. **Taxonomy of Educational Objectives, Handbook 1: Cognitive Domain**. New York: Longman, 1956. Citado na página 40.
- BOLLELA, V. R.; SENGER, M. H.; TOURINHO, F. S. V.; AMARAL, E. Team-based learning: from theory to practice. 2014. Citado nas páginas 36 e 37.
- BONWELL, C. C.; EISON, J. A. **Active Learning: Creating Excitement in the Classroom. 1991 ASHE-ERIC Higher Education Reports**. [S.l.]: ERIC, 1991. Citado nas páginas 28, 34 e 35.
- BOURGOIN, M.; CHAILLOUX, E.; LAMOTTE, J.-L. Efficient abstractions for gpgpu programming. **International Journal of Parallel Programming**, v. 42, n. 4, p. 583–600, 2014. Citado na página 25.

- BRERETON, P.; KITCHENHAM, B. A.; BUDGEN, D.; TURNER, M.; KHALIL, M. Lessons from applying the systematic literature review process within the software engineering domain. **Journal of systems and software**, Elsevier, v. 80, n. 4, p. 571–583, 2007. Citado nas páginas 45 e 47.
- CASSEL, L.; CLEMENTS, A.; DAVIES, G.; GUZDIAL, M.; MCCAULEY, R.; MCGETTRICK, A.; SLOAN, B.; SNYDER, L.; TYMANN, P.; WEIDE, B. W. **Computer Science Curriculum 2008: An Interim Revision of CS 2001**. New York, NY, USA, 2008. Citado nas páginas 38, 39 e 57.
- CHICKERING, A. W.; GAMSON, Z. F. Seven principles for good practice in undergraduate education. **AAHE bulletin**, ERIC, v. 3, p. 7, 1987. Citado nas páginas 19, 33 e 34.
- CLANCY, M. J.; LINN, M. C. Patterns and pedagogy. **SIGCSE Bull.**, ACM, New York, NY, USA, v. 31, n. 1, p. 37–42, mar. 1999. ISSN 0097-8418. Disponível em: <<http://doi.acm.org/10.1145/384266.299673>>. Citado na página 41.
- CUENCA, J.; GIMÉNEZ, D. A parallel programming course based on an execution time-energy consumption optimization problem. In: **2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. [S.l.: s.n.], 2016. p. 996–1003. Citado na página 42.
- CURRICULA, A. Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science. 01 2013. Citado nas páginas 27, 28, 38, 39, 40, 55, 58, 64, 65, 72, 88 e 104.
- DILLENBOURG, P. What do you mean by collaborative learning? In: DILLENBOURG, P. (Ed.). **Collaborative-learning: Cognitive and Computational Approaches**. Oxford: Elsevier, 1999. cap. 1, p. 1–19. Citado na página 42.
- DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. **Information and software technology**, Elsevier, v. 50, n. 9, p. 833–859, 2008. Citado na página 49.
- ERNST, D.; WITTMAN, B.; HARVEY, B.; MURPHY, T.; WRINN, M. Preparing students for ubiquitous parallelism. **ACM SIGCSE Bulletin**, v. 41, p. 136, 03 2009. Citado na página 26.
- ERNST, D. J.; STEVENSON, D. E. Concurrent cs: Preparing students for a multicore world. **SIGCSE Bull.**, ACM, New York, NY, USA, v. 40, n. 3, p. 230–234, jun. 2008. ISSN 0097-8418. Disponível em: <<http://doi.acm.org/10.1145/1597849.1384333>>. Citado na página 38.
- FELDHAUSEN, R.; BELL, S.; ANDRESEN, D. Minimum Time, Maximum Effect: Introducing Parallel Computing in CS0 and STEM Outreach Activities Using Scratch. In: **Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment**. New York, NY, USA: ACM, 2014. (XSEDE '14), p. 75:1—75:7. ISBN 978-1-4503-2893-7. Citado na página 58.
- FERNÁNDEZ Álvaro; FERNÁNDEZ, C.; MIGUEL-DÁVILA, J. Ángel; CONDE, M. Ángel; MATELLÁN, V. Supercomputers to improve the performance in higher education: A review of the literature. **Computers & Education**, v. 128, p. 353 – 364, 2019. ISSN 0360-1315. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0360131518302756>>. Citado na página 25.

FIGLIORE, S.; BAKHOUYA, M.; SMARI, W. W. **On the road to exascale: Advances in High Performance Computing and Simulations—An overview and editorial**. [S.l.]: Elsevier, 2018. Citado na página 25.

FREIRE, P. **Pedagogia do Oprimido**. Rio de Janeiro: Paz e Terra, 1987. Citado nas páginas 15, 32 e 33.

GHAFOOR, S.; BROWN, D.; ROGERS, M. Integrating parallel computing in introductory programming classes: An experience and lessons learned. In: . [S.l.: s.n.], 2017. Citado nas páginas 27, 28, 60, 61 e 62.

GIBBONS, A. S. Model-centered instruction. **Journal of Structural Learning and Intelligent Systems**, v. 14, n. 4, p. 511–540, 2001. Citado na página 42.

GONÇALVES, R. Q. **Ensino de Gerenciamento de Projetos de Software Mediado por Ferramentas**. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, 2017. Citado na página 41.

GROSS, T. R. Breadth in depth: A 1st year introduction to parallel programming. In: **Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education**. New York, NY, USA: ACM, 2011. (SIGCSE '11), p. 435–440. ISBN 978-1-4503-0500-6. Disponível em: <<http://doi.acm.org/10.1145/1953163.1953291>>. Citado na página 57.

HEYWOOD, T.; RANKA, S. A practical hierarchical model of parallel computation. In: . [S.l.: s.n.], 1992. p. 18 – 25. ISBN 0-8186-2310-1. Citado na página 68.

IMAM, S.; SARKAR, V. Habanero-Java Library: A Java 8 Framework for Multicore Programming. In: **Proceedings of the 2014 International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools**. New York, NY, USA: ACM, 2014. (PPPJ '14), p. 75–86. ISBN 978-1-4503-2926-2. Citado na página 58.

INUWA, I. Perceptions and attitudes of first-year medical students on a modified team-based learning (tbl) strategy in anatomy. v. 12, p. 336–43, 08 2012. Citado nas páginas 15 e 37.

JOHN, D. J. Nsf supported projects: Parallel computation as an integrated component in the undergraduate curriculum in computer science. **SIGCSE Bull.**, Association for Computing Machinery, New York, NY, USA, v. 26, n. 1, p. 357–361, mar. 1994. ISSN 0097-8418. Disponível em: <<http://doi.org/10.1145/191033.191167>>. Citado na página 27.

KELLER, J. Motivation and instructional design: A theoretical perspective. **Journal of Instructional Development**, v. 2, p. 26–34, 06 1979. Citado na página 124.

_____. Motivation in cyber learning environments. **International Journal of Educational Technology**, v. 1, p. 7–30, 01 1999. Citado na página 124.

_____. **Motivational Design for Learning and Performance: The ARCS Model Approach**. [S.l.: s.n.], 2010. 1-353 p. Citado nas páginas 121, 124, 133 e 144.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1–26, 2004. Citado na página 45.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. 2007. Citado nas páginas 48 e 53.

KO, Y.; BURGSTALLER, B.; SCHOLZ, B. Parallel from the beginning: The case for multicore programming in the computer science undergraduate curriculum. In: **Proceeding of the 44th ACM Technical Symposium on Computer Science Education**. New York, NY, USA: ACM, 2013. (SIGCSE '13), p. 415–420. ISBN 978-1-4503-1868-6. Disponível em: <<http://doi.acm.org/10.1145/2445196.2445320>>. Citado nas páginas 27 e 58.

LIBERT, C.; VANHOOF, W. Survey of software visualization systems to teach message-passing concurrency in secondary school. In: . [S.l.: s.n.], 2017. p. 386–397. ISBN 978-3-319-60284-4. Citado na página 60.

MCMMASTER, K.; RAGUE, B.; ANDERSON, N. Integrating mathematical thinking, abstract thinking, and computational thinking. In: **2010 IEEE Frontiers in Education Conference (FIE)**. [S.l.: s.n.], 2010. p. S3G–1–S3G–6. ISSN 0190-5848. Citado na página 27.

MIZUKAMI, M. da G. N. **Ensino: as abordagens do processo**. São Paulo: Epu, 1986. Citado nas páginas 32 e 33.

MOORE, S. V.; DUNLOP, S. R. A flipped classroom approach to teaching concurrency and parallelism. In: **2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. [S.l.: s.n.], 2016. p. 987–995. Citado na página 41.

MURESANO, R.; REXACHS, D.; LUQUE, E. Learning parallel programming: a challenge for university students. **Procedia Computer Science**, Elsevier, v. 1, n. 1, p. 875–883, 2010. Citado na página 26.

NEWHALL, T.; DANNER, A.; WEBB, K. Pervasive parallel and distributed computing in a liberal arts college curriculum. **Journal of Parallel and Distributed Computing**, v. 105, 01 2017. Citado na página 61.

PARMELEE, D.; MICHAELSEN, L. K.; COOK, S.; HUDES, P. D. Team-based learning: a practical guide: Amee guide no. 65. **Medical teacher**, Taylor & Francis, v. 34, n. 5, p. e275–e287, 2012. Citado nas páginas 78 e 79.

PRASAD, S.; COORDINATOR; CHTCHELKANOVA, A.; DAS, S.; DEHNE, F.; GOUDA, M.; GUPTA, A.; JAJA, J.; KANT, K.; LA, A.; NSF, S.; LEBLANC, R.; LUMSDAINE, M.; PADUA, D.; PARASHAR, M.; PRASANNA, N.; ROBERT, Y.; ROSENBERG, A.; SAHNI, S.; WU, J. Nsf/ieee-tcpp curriculum initiative on parallel and distributed computing: Core topics for undergraduates. In: **Proceedings of the 45th ACM Technical Symposium on Computer Science Education**. New York, NY, USA: ACM, 2012. (SIGCSE '14). ISBN 978-1-4503-2605-6. Disponível em: <<https://grid.cs.gsu.edu/~tcpp/curriculum/>>. Acesso em: 01/02/2018. Citado nas páginas 15, 27, 38, 40, 55, 58, 59, 61, 62, 64, 65, 66, 67, 69, 72, 88, 104 e 153.

PREGUNTA, L. A estratégia pico para a construção da pergunta de pesquisa e busca de evidências. **Rev Latino-am Enfermagem**, SciELO Brasil, v. 15, n. 3, 2007. Citado na página 46.

RAGUE, B. Measuring cs1 perceptions of parallelism. In: **2011 Frontiers in Education Conference (FIE)**. [S.l.: s.n.], 2011. p. S3E–1–S3E–6. ISSN 0190-5848. Citado na página 57.

RAUBER, T.; RÜNGER, G. **Parallel Programming for Multicore and Cluster Systems, 2nd Edition**. [S.l.: s.n.], 2013. ISBN 978-3-642-37800-3. Citado nas páginas 68 e 69.

- RICHARDS, J. C.; RODGERS, T. Method: Approach, design, and procedure. **TESOL Quarterly**, [Wiley, Teachers of English to Speakers of Other Languages, Inc. (TESOL)], v. 16, n. 2, p. 153–168, 1982. ISSN 00398322. Disponível em: <<http://www.jstor.org/stable/3586789>>. Citado nas páginas 15, 31 e 32.
- SAKELLARIOU, R. Experiences with teaching a second year distributed computing course. In: . [S.l.: s.n.], 2016. p. 28–37. ISBN 978-3-319-58942-8. Citado na página 60.
- SAVIANI, D. **Escola e Democracia**. Campinas, SP: Autores Associados, 1986. 29 p. Citado na página 32.
- SHACKELFORD, R.; MCGETTRICK, A.; SLOAN, R.; TOPI, H.; DAVIES, G.; KAMALI, R.; CROSS, J.; IMPAGLIAZZO, J.; LEBLANC, R.; LUNT, B. Computing curricula 2005: The overview report. **SIGCSE Bull.**, Association for Computing Machinery, New York, NY, USA, v. 38, n. 1, p. 456–457, mar. 2006. ISSN 0097-8418. Disponível em: <<https://doi.org/10.1145/1124706.1121482>>. Citado na página 38.
- SKOPIN, I. Early learning in parallel programming. p. 219–229, 01 2014. Citado na página 59.
- SOUZA, D. B.; JESUS, D. L. S. D.; DALTRO, E. dos S.; BISPO, J. M. M.; SANTOS, L. S.; SANTOS, R. de C. **Educação Segundo a Abordagem Tradicional**. 2011. Disponível em: <<http://educacaotradicional.blogspot.com.br/p/metodologia.html>>. Citado nas páginas 15 e 33.
- STAHL, G.; KOSCHMANN, T.; SUTHERS, D. Computer-supported collaborative learning: An historical perspective. In: SAWYER, R. K. (Ed.). **Cambridge handbook of the learning sciences**. Cambridge, UK: Elsevier, 2006. p. 409–426. Citado na página 42.
- TANENBAUM, A. S.; BOS, H. **Modern Operating Systems**. 4th. ed. USA: Prentice Hall Press, 2014. ISBN 013359162X. Citado na página 68.
- TANENBAUM, A. S.; GOODMAN, J. R. **Structured Computer Organization**. 4th. ed. USA: Prentice Hall PTR, 1998. ISBN 0130959901. Citado na página 68.
- TANENBAUM, A. S.; WETHERALL, D. J. **Computer Networks**. 5th. ed. USA: Prentice Hall Press, 2010. ISBN 0132126958. Citado na página 68.
- Computing curricula 2001. **J. Educ. Resour. Comput.**, ACM, New York, NY, USA, v. 1, n. 3es, set. 2001. ISSN 1531-4278. Disponível em: <<http://doi.acm.org/10.1145/384274.384275>>. Citado nas páginas 38 e 39.
- TORBERT, S.; VISHKIN, U.; TZUR, R.; ELLISON, D. J. Is Teaching Parallel Algorithmic Thinking to High School Students Possible?: One Teacher’s Experience. In: **Proceedings of the 41st ACM Technical Symposium on Computer Science Education**. New York, NY, USA: ACM, 2010. (SIGCSE ’10), p. 290–294. ISBN 978-1-4503-0006-3. Citado na página 56.
- UL-ABDIN, Z.; SVENSSON, B. Towards teaching embedded parallel computing: An analytical approach. In: ACM. **Proceedings of the Workshop on Computer Architecture Education**. [S.l.], 2015. p. 8. Citado na página 41.
- VALENTINE, D. HPC/PDC Immunization in the Introductory Computer Science Sequence. In: **Proceedings of the Workshop on Education for High-Performance Computing**. Piscataway, NJ, USA: IEEE Press, 2014. (EduHPC ’14), p. 9–14. ISBN 978-1-4799-7021-6. Citado na página 59.

WOHLIN, C.; RUNESON, P.; HöST, M.; OHLSSON, M.; REGNELL, B.; WESSLÉN, A. Experimentation in software engineering. In: _____. [S.l.: s.n.], 2012. p. 123–151. ISBN 978-3-642-29043-5. Citado nas páginas [75](#), [87](#), [103](#) e [121](#).

ZARESTKY, J.; BANGERTH, W. Teaching high performance computing: Lessons from a flipped classroom, project-based course on finite element methods. In: IEEE PRESS. **Proceedings of the Workshop on Education for High-Performance Computing**. [S.l.], 2014. p. 34–41. Citado na página [41](#).

QUESTIONÁRIO QUALITATIVO - EXPERIMENTO 1

1a Avaliação Qualitativa das Aulas de Programação Concorrente

Esta avaliação tem por objetivo saber a sua opinião sobre as aulas de Programação Concorrente já ministradas, principalmente em relação ao uso da metodologia de ensino.

As respostas não são anônimas. Queremos considerar apenas respostas de alunos matriculados na disciplina.

As respostas aqui não valem nota.

Responda de maneira construtiva: elogie no que acredita haver mérito, dê sugestões caso tenha e critique aquilo que possa ser melhorado.

1. O planejamento do curso foi suficientemente claro até o início da segunda aula? *

Foi possível compreender o funcionamento das aulas no semestre com os exemplos dados na primeira aula e começo da segunda?

- ☐ Foi bem claro e eu entendi como o curso iria funcionar.
- ☐ Foi claro, embora eu tenha ficado com dúvidas pontuais resolvidas na sequência.
- ☐ Não foi claro e eu não entendi nada.
- ☐ Não sei opinar.

2. Caso a resposta sobre o Planejamento do Curso não tenha sido "Foi bem claro e eu entendi como o curso iria funcionar", explique o porquê.

3. Foi possível compreender o funcionamento do TBL (Team Based Learning), suas etapas e características até o início da segunda aula? *

- ☐ Sim, eu compreendi.
- ☐ Sim, mas eu fiquei com algumas dúvidas pontuais resolvidas na sequência.
- ☐ Não, eu não compreendi.
- ☐ Não sei opinar.

4. **Caso a resposta sobre a compreensão do TBL não tenha sido "Sim, eu compreendi", explique o porquê.**

5. **Qual método de ensino/aprendizado você acredita ser melhor? ***

Baseado em suas experiências de aprendizado e com as aulas ministradas até agora , com qual método você se sente mais confortável e/ou facilita sua compreensão do conteúdo?

- ☐ TBL (Team Based Learning)
- ☐ Tradicional
- ☐ Outro: _____

6. **Justifique a resposta da questão anterior. ***

Baseado na opção selecionada na questão anterior, justifique sua escolha.

7. **Você está gostando das aulas com TBL? ***

- ☐ Sim, estou gostando.
- ☐ Não estou gostando.
- ☐ Isso é indiferente para mim.
- ☐ Achei muito ruim e gostaria das aulas expositivas, tradicionais.

8. **Justifique a resposta sobre gostar das aulas com TBL. ***

9. **Qual sua opinião sobre a aplicação de avaliações contínuas (ACs)? ***

Quando comparado com o modelo de provas bimestrais, com conteúdo cumulativo, você acredita que as avaliações contínuas favorecem ou não seu aprendizado?

Marcar apenas uma oval.

- ☐ As ACs são melhores e favorecem o aprendizado.
- ☐ Não vi diferença
- ☐ As ACs são piores e não favorecem o aprendizado.
- ☐ Não sei opinar.
- ☐ Outro: _____

10. Em relação às quatro etapas do TBL, qual você acredita que melhor favorece seu aprendizado? *

As quatro etapas do TBL são: estudo prévio do conteúdo, garantia de preparo individual (avaliação individual e sem consulta), garantia de preparo em equipe (mesma avaliação em grupo) e aplicação prática (exercício prático na última porção da aula). Qual delas mais auxilia o seu processo de aquisição de conhecimento?

- ☐ Estudo prévio do conteúdo
- ☐ Garantia de preparo individual
- ☐ Garantia de preparo em equipe
- ☐ Aplicação prática
- ☐ Nenhuma delas
- ☐ Todas as quatro etapas.
- ☐ Outro: _____

11. Você gostaria que o curso voltasse a ser ensinado pelo método tradicional de ensino, com aulas expositivas? *

- ☐ Sim, aprendo mais com o método tradicional de ensino, com aulas expositivas.
- ☐ Não, estou aprendendo mais com o TBL.

12. Dê suas sugestões, críticas e/ou elogios sobre o curso em andamento.

Este espaço é livre para você colocar os seus comentários em relação ao curso. Sugira melhorias, mudanças ou faça seus elogios/críticas.

QUESTÕES DO PRÉ-TESTE - CURSO DE OPENMP PARA INICIANTE EM COMPUTAÇÃO

Considere este código em C e responda às questões 1 e 2:

```
1  \#include<stdio.h>
2  int main(void) {
3      int idade;
4
5      scanf("%d", &idade);
6
7      if (idade < 0)
8          printf("Idade inválida! Precisa ser um número natural. \n");
9
10     else if (idade < 18)
11         printf("Você é menor de idade, pois tem menos de 18 anos. \n");
12
13     else if (idade < 65)
14         printf("Você é maior de idade e tem menos de 65 anos. \n");
15
16     else if (idade < 99)
17         printf("Você tem mais de 65 anos e menos de 99 anos. \n");
18
19     else
20         printf("Você tem mais de 99 anos. \n");
21
```

```
22 return(0);  
23 }
```

1. Este programa utiliza:

- a) estruturas de repetição
- b) chamadas de funções e procedimentos
- c) estruturas de decisão
- d) estruturas de repetição e de decisão
- e) não sei

2. Qual das seguintes diretivas do OpenMP poderia ser usada para paralelizar o programa descrito?

- a) #pragma omp parallel sections
- b) #pragma omp parallel for
- c) #pragma omp ordered
- d) #pragma omp parallel
- e) não sei

3. (FUNCAB) Considerando este programa em C:

```
1 #include<std.io>  
2 main(){  
3 int i, d;  
4 scanf("%d", &d);  
5 for (i = 0; i < 2; i++)  
6 d = d * d;  
7  
8 printf("%d \n", d);  
9 }
```

Caso seja fornecido como entrada o valor 3, a saída será:

- a) 3
- b) 9
- c) 27
- d) 81
- e) não sei

4. A diretiva #pragma omp parallel for do OpenMP permite:

- a) replicar as iterações previstas no for sequencial sobre um número de threads
- b) dividir as iterações previstas no for sequencial sobre um número de threads
- c) gerar mais iterações paralelas que as previstas no for sequencial, porém, otimizando o código
- d) reduzir o número total de iterações previstas no for sequencial para a solução do problema
- e) não sei

5. Analise o seguinte código em C:

```
1  \#include <stdio.h>
2  void calcula_resultado (int n1, int n2)
3  {
4  soma += n1 + n2;
5  printf("%d \n", soma);
6  }
7
8  int main()
9  {
10 int i, soma = 0;
11 int vet_1[10];
12 int vet_2[10];
13
14 for (i=0; i< 10; i++){
15 vet_1[i] = i;
16 vet_2[i] = i+1;
17
18     calcula_resultado(vet_1[i], vet_2[i]);
19 }
20
21 return 0;
22 }
```

5.1. O código acima utiliza as seguintes estruturas:

- a) estruturas de decisão
- b) estruturas de decisão e de repetição
- c) chamadas de funções e estrutura de repetição

- d) chamadas de funções recursivas
- e) não sei

5.2. É correto afirmar que:

- a) O valor do conteúdo de cada posição dos vetores está relacionado com o índice do loop.
- b) Os dois vetores possuem 10 posições com índice de 1 a 10.
- c) Os dois vetores são utilizados para armazenar o resultado de soma.
- d) O programa está incorreto, pois o índice do vetor não é definido em "vet_1[i] = i;".
- e) não sei

5.3. Uma forma de paralelizar a execução desse código com OpenMP seria utilizar:

- a) #pragma omp parallel sections
- b) #pragma omp parallel for
- c) #pragma omp ordered
- d) #pragma omp parallel
- e) não sei

5.4. Considerando que o código seja paralelizado, a variável "soma":

- a) Será acessada por todas as threads na ordem que o programador definir.
- b) Não necessita de alterações, pois será acessada por todas as threads na linha 4.
- c) Não deve ser acessada pelas threads, pois trata-se de uma região crítica.
- d) Deve ter seu acesso protegido, pois trata-se de um dado compartilhado.
- e) não sei

6. Um computador deve calcular as três operações aritméticas dadas abaixo. Suponha que a, b, c, d, e, f, i, j sejam todos inteiros. O tempo necessário para calcular cada instrução individual é de 1 nseg (nano segundo).

- (I) $c = a + b;$
- (II) $d = e + f;$
- (III) $j = i - c;$

6.1. Suponha que uma unidade de processamento esteja disponível para realizar o cálculo. Qual é o tempo mínimo necessário para calcular as três declarações?

- a) 1 nsec
- b) 2 nsec

- c) 3 nsec
- d) 4 nsec
- e) não sei

6.2. Suponha que três unidades de processamento estejam disponíveis para realizar o cálculo. Qual é o tempo mínimo necessário para calcular as três declarações?

- a) 1 nsec
- b) 2 nsec
- c) 3 nsec
- d) 4 nsec
- e) não sei

QUESTÕES DO PÓS-TESTE - CURSO DE OPENMP PARA INICIANTES EM COMPUTAÇÃO

1) A linha de código `#pragma omp parallel for` implica que:

- a) O bloco de comandos abaixo desta linha necessariamente deve ser um loop for, cujas iterações serão, caso possível, divididas entre as threads disponíveis.
- b) O bloco de comandos abaixo desta linha necessariamente deve ser um loop for que será replicado e executado em processos paralelos.
- c) O bloco de comandos abaixo desta linha pode ou não representar um loop for que será replicado e executado em paralelo. Neste caso, o primeiro for a ser encontrado será paralelizado.
- d) A diretiva irá paralelizar todos os laços de repetição for que forem encontrados pelo compilador.
- e) Não sei

2) A linha de código `#pragma omp parallel sections`:

- a) Permite que seções de memória sejam acessadas em paralelo.
- b) Indica que a seguir devem ser usadas seções paralelas de códigos sequenciais em diferentes threads.
- c) Faz com que laços for sejam distribuídos a diferentes seções.
- d) Indica que as seções especificadas serão executadas sequencialmente.
- e) Não sei

3) A cláusula `private`:

a) Identifica blocos de computação local (privada) que precisam de dados locais para executarem.

b) Identifica quais variáveis devem ser locais (privadas) às threads em questão.

c) Identifica threads que precisam executar localmente em um núcleo do processador.

d) identifica as instruções que fazem parte de uma região crítica.

e) Não sei

4) A linha de código `#pragma omp critical` permite indicar que o bloco de instruções abaixo desta linha:

a) É crítico para o desempenho e que o mesmo precisa ser paralelizado com o máximo de eficiência e speedup.

b) Deve ser mapeado sempre no mesmo núcleo para evitar problemas de desempenho e erros de funcionalidade.

c) Representa uma região crítica e deve ser executado pelas threads garantindo-se a exclusão mútua entre elas.

d) Garante que os dados dessa região são críticos e não serão executados em nenhuma thread.

e) Não sei.

5) Considere o seguinte grafo:

A ilustração pode ser a representação de:

a) Uma operação de redução.

b) Um paralelismo funcional com `#pragma omp parallel for`.

c) A execução de um `#pragma omp parallel for` com a cláusula master.

- d) A execução sequencial da operação de soma entre diferentes valores inteiros.
- e) Não sei.

6) A estratégia de paralelização de um código pode ser classificada em dois tipos: paralelismo por dados e paralelismo por funções. As diretivas do OpenMP que possibilitam essas estratégias são, respectivamente:

- a) Uso de variáveis nas cláusulas private e shared.
- b) #pragma omp parallel reduction e #pragma omp parallel for.
- c) #pragma omp parallel for e #pragma omp sections.
- d) #schedule(dynamic) e #schedule(guided)
- e) Não sei.

7) Considere o seguinte fragmento de código:

```

1  \#include<stdio.h>
2  int main()
3  {
4  int i, soma = 0;
5  int vet_1[10];
6  int vet_2[10];
7
8  for (i=0; i< 10; i++){
9  vet_1[i] = i;
10 vet_2[i] = i+1;
11
12     soma += vet_1[i] + vet_2[i] ;
13 printf("Soma: %d na iteração %d \n", soma, i);
14 }
15
16 return 0;
17 }
```

7.1) Uma forma correta de paralelizar a execução desse código com OpenMP seria utilizar a diretiva:

- a) #pragma omp sections para alternar as iterações do for entre as threads disponíveis.
- b) #pragma omp parallel for para dividir as iterações do for em threads distintas.
- c) #pragma omp ordered para executar as iterações do for pela thread mestre.
- d) #pragma omp parallel critical para executar com diferentes threads, as quais otimizam

as operações críticas (aquelas que demoram mais tempo).

e) Não sei.

7.2) Considerando que o código seja paralelizado usando uma variável global / compartilhada chamada “soma”, esta variável:

a) É protegida automaticamente pelo OpenMP e o programador não precisa fazer nada mais.

b) Não necessita de proteção e o programador não precisa fazer nada.

c) Representa um erro de programação, pois trata-se de uma região crítica.

d) Deve ter seu acesso protegido pelo programador, pois trata-se de uma região crítica.

e) Não sei.

8) Um computador deve calcular as três operações aritméticas dadas abaixo. Suponha que a, b, c, d, e, f, i, j sejam todos inteiros. O tempo necessário para calcular cada instrução individual é de 1 nsec (nano segundo).

1 $d = e + f;$

2 $c = a + b;$

3 $j = c - i;$

8.1) Suponha que uma unidade de processamento esteja disponível para realizar o cálculo. Qual é o tempo mínimo necessário para calcular as três declarações?

a) 1 nsec.

b) 2 nsec.

c) 3 nsec.

d) 4 nsec.

e) não sei.

8.2) Suponha que três unidades de processamento estejam disponíveis para realizar o cálculo. Qual é o tempo mínimo necessário para calcular as três declarações?

a) 1 nsec.

b) 2 nsec.

c) 3 nsec.

d) 4 nsec.

e) não sei.

PRÉ-TESTE - CURSO DE INTRODUÇÃO AO OPENMP BASEADO EM PROBLEM BASED LEARNING

Programação Paralela em Máquinas Multicore: um curso prático com
OpenMP
Pré-Teste - 05/06/2019

Nome: _____

Curso: _____ Período: _____

Observações importantes:

- Esta prova é individual e sem consulta.
- Esta prova não vale nota. Você não será prejudicado com uma nota baixa. Nosso objetivo é determinar o nível de conhecimento dos alunos para personalizar o curso.
- Há apenas uma alternativa correta para cada questão.
- Se não souber a resposta, por favor, **não “chute”**; marque a opção **e) não sei**.
- Se tiver alguma dúvida em relação à prova, levante a mão e pergunte ao professor.

Um computador deve calcular as três operações aritméticas dadas abaixo. Suponha que a, b, c, d, e, f, i, j sejam todos inteiros. O tempo necessário para calcular cada instrução individual é de 1 nsec (nano segundo).

(I) $c = a + b$;

(II) $d = e + f$;

(III) $j = i - c$;

1) Suponha que três unidades de processamento estejam disponíveis para realizar o cálculo. Qual é o tempo mínimo necessário para calcular as três declarações?

- (A) 1 nsec
- (B) 2 nsec**
- (C) 3 nsec
- (D) 4 nsec
- (E) 6 nsec

2) A linha de código contendo ***#pragma omp parallel*** significa que:

- a) As instruções do bloco de comandos que está na sequência à linha acima serão executadas todas concorrentemente.
- b) O bloco de instruções e as instruções serão ambos executados em paralelo.
- c) O bloco de instruções que está na sequência à linha acima será replicado em diferentes threads e que cada thread executará concorrentemente.**
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

3) A função **omp_in_parallel()** permite:

- a) Determinar que a *thread* entrará em uma região paralela do código.
- b) Identificar se a *thread* está executando ou não uma região paralela do código.
- c) Configurar parâmetros da execução paralela de um processo *OpenMP*.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

4) A linha de código **#pragma omp parallel for** implica que:

- a) O bloco de comandos abaixo desta linha necessariamente deveria ser um **loop for**, cujas iterações serão, caso possível, divididas entre as *threads* disponíveis;
- b) O bloco de comandos abaixo desta linha necessariamente deveria ser um **loop for** que será replicado e executado em paralelo.
- c) O bloco de comandos abaixo desta linha pode ou não representar um **loop for** que será replicado e executado em paralelo. Neste caso, o primeiro **for** a ser encontrado será paralelizado.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

5) A função **omp_set_lock()** permite:

- a) Impedir o escalonamento de *threads* em núcleos específicos.
- b) Fixar critérios para o acesso às variáveis.
- c) Proteger a entrada em uma região crítica, no caso de variáveis compartilhadas.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

6) A cláusula **schedule** determina como será a distribuição:

- a) De *threads* aos núcleos de um processador.
- b) De variáveis locais às *threads*.
- c) Das iterações de um **for** para as *threads*.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

7) A cláusula **nowait** do OpenMP:

- a) Evita a sincronização das *threads* ao final do bloco, deixando o código mais rápido.
- b) Evita que as *threads* esperem para serem executadas.
- c) Evita que as *threads* bloqueiem no final de suas execuções com variáveis persistentes.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

8) A cláusula **private**:

- a) Identifica blocos de computação local que precisam de dados locais para executarem.
- b) Identifica quais variáveis devem ser locais às *threads* em questão.
- c) Identifica *threads* que precisam executar localmente em um núcleo do processador.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

9) A cláusula **if()** no OpenMP:

- a) É semelhante ao **if ()** do C.
- b) Permite verificar se uma execução de *thread* finalizou com sucesso.
- c) Permite deixar condicional a criação de *threads*.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

10) Considere o seguinte grafo:

A ilustração pode ser a representação de:

- a) Uma operação de redução.
- b) Um paralelismo funcional com **#pragma omp parallel for**.
- c) A execução de um **#pragma omp parallel for** com a cláusula **master**.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

PÓS-TESTE - CURSO DE INTRODUÇÃO AO OPENMP BASEADO EM PROBLEM BASED LEARNING

Nome: _____

Curso: _____ Período: _____

Observações importantes:

- Esta prova é individual e sem consulta.
- Esta prova não vale nota. Você não será prejudicado com uma nota baixa. Nosso objetivo é determinar o nível de conhecimento dos alunos para personalizar o curso.
- Há apenas uma alternativa correta para cada questão.
- Se não souber a resposta, por favor, não “chute”; marque a opção **e) não sei**.
- Se tiver alguma dúvida em relação à prova, levante a mão e pergunte ao professor.

1) As variáveis em regiões paralelas do *OpenMP* devem ser identificadas como:

- a) *Int*, *string*, *boolean*, *byte* e *float*.
- b) 16, 32 ou 64 bits.
- c) *Shared*, *private*, *firstprivate*, *lastprivate*, *reduction*, *threadprivate* ou *copyin*.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

2) A função *omp_get_thread_num()* retorna:

- a) O número de *threads* em execução na porção paralela do código.
- b) O número da *thread* em execução, sendo que *thread* mestre recebe o número zero e as demais um sequencial > 0 .
- c) O número da *thread* em execução, sendo que *thread* mestre recebe o *PID* do processo e as demais um sequencial a partir do *PID*.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

3) A cláusula *num_threads()* no *OpenMP* permite:

- a) Determinar o número de *threads* a serem executadas em uma *#pragma omp parallel*.
- b) Determinar o número máximo de *threads* que podem ser criadas.
- c) Determinar o número da *thread* atual.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

4) A linha de código contendo *#pragma omp for* significa que:

- a) O bloco de comandos abaixo desta linha necessariamente representa um *loop for*, cujas iterações serão distribuídas às *threads*, caso esteja dentro de uma região paralela.
- b) O bloco de comandos abaixo desta linha necessariamente representa um *loop for* que será totalmente replicado e executado em paralelo.
- c) O bloco de comandos abaixo desta linha pode ou não representar um *loop for* que será totalmente replicado e executado em paralelo.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

5) O escalonamento dinâmico difere do escalonamento estático por:

- a) Realizar uma diminuição exponencial do número de iterações de um **for** a serem atribuídas às *threads* em função da quantidade de iterações que faltam executar.
- b) Realizar a atribuição de *threads*, e não de iterações de um **for**, aos núcleos que estiverem ociosos em tempo de execução.
- c) Realizar a atribuição de iterações de um **for** às *threads* que estiverem ociosas em tempo de execução.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

6) A estratégia de paralelização de um código pode ser classificada em dois tipos: **paralelismo por dados e paralelismo por tarefas/funcional**. As diretivas do *OpenMP* que possibilitam essas estratégias são, respectivamente:

- a) Uso de variáveis nas cláusulas **private** e **shared**.
- b) **schedule(static,100)** e **#pragma omp parallel for**.
- c) **#pragma omp parallel for** e **#pragma omp parallel sections**.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

7) A linha de código **#pragma omp parallel sections**:

- a) Permite que seções de memória sejam acessadas em paralelo.
- b) Indica que o código a seguir deve conter seções paralelas executadas em diferentes *threads*.
- c) Faz com que laços **for** sejam distribuídos a diferentes seções.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

8) A linha de código **#pragma omp critical** permite indicar que o bloco de instruções abaixo desta linha:

- a) É crítico para o desempenho e que o mesmo precisa ser paralelizado com o máximo de eficiência e speedup.
- b) Deve ser mapeado sempre no mesmo núcleo para evitar problemas de desempenho e erros de funcionalidade.
- c) Representa uma região crítica e deve ser executado pelas *threads* garantindo-se a exclusão mútua entre elas.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

Considere o seguinte fragmento de código:

```
1  #include<stdio.h>
2  int main()
3  {
4      int i, soma = 0;
5      int vet_1[10];
6      int vet_2[10];
7
8      for (i=0; i< 10; i++){
9          vet_1[i] = i;
10         vet_2[i] = i+1;
11
12         soma += vet_1[i] + vet_2[i] ;
13         printf("Soma: %d na iteração %d \n", soma, i);
14     }
15
16     return 0;
17 }
```

9) Uma forma correta de paralelizar a execução desse código com *OpenMP* seria utilizar a diretiva:

- a) **#pragma omp sections** para alternar as iterações do **for** entre as *threads* disponíveis.
- b) **#pragma omp parallel for** para dividir as iterações do **for** em *threads* distintas.
- c) **#pragma omp ordered** para executar as iterações do **for** pela *thread* mestre.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.

10) Considerando que o código seja paralelizado usando uma variável global / compartilhada chamada "**soma**", esta variável:

- a) É protegida automaticamente pelo *OpenMP* e o programador não precisa fazer nada mais.
- b) Não necessita de proteção e por isso o programador não precisa fazer nada.
- c) Deve ter seu acesso protegido pelo programador, pois trata-se de uma região crítica.
- d) Nenhuma das alternativas anteriores.
- e) Não sei.