UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Um framework para agrupar funções com base no comportamento da comunicação de dados em plataformas multiprocessadas

Rafael Ribeiro dos Santos

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

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Rafael Ribeiro dos Santos

Um framework para agrupar funções com base no
comportamento da comunicação de dados em plataformas
multiprocessadas

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. Vanderlei Bonato

USP – São Carlos
Agosto de 2018

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

S237f Santos, Rafael Ribeiro dos
Um framework para agrupar funções com base no
comportamento da comunicação de dados em plataformas
multiprocessadas / Rafael Ribeiro dos Santos;
orientador Vanderlei Bonato. -- São Carlos, 2018.
102 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, 2018.

1. Agrupamento de funções. 2. Plataformas
heterogêneas. 3. Distribuição de comunicação. 4.
Largura de banda variável. I. Bonato, Vanderlei,
orient. II. Título.

Rafael Ribeiro dos Santos

**A framework for clustering functions based on the behavior
of data communication on multiprocessed platforms**

Master dissertation submitted to the Institute of
Mathematics and Computer Sciences – ICMC-USP,
in partial fulfillment of the requirements for the
degree of the Master Program in Computer Science
and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Vanderlei Bonato

USP – São Carlos
August 2018

Este trabalho é dedicado para minha família que me apoiou durante todo o mestrado.

AGRADECIMENTOS

Gostaria de agradecer a minha mãe Rose, minha irmã Renata e meu pai Erasmo por sempre me apoiarem durante o mestrado, sempre me auxiliando durante as dificuldades encontradas.

Também gostaria de agradecer meu orientador Prof. Dr. Vanderlei Bonato por sempre me auxiliar nos momentos difíceis do projeto, sempre tentando solucionar o problema da melhor forma possível. Também gostaria de agradecer a Capes pelo apoio financeiro.

Finalmente, gostaria de agradecer a meus novos amigos do LCR, Carlos, Cláudio e Erinaldo, por sempre estarem presentes nos momentos de alegria e de dificuldades.

“Você não é derrotado quando perde, mais sim quando você desiste.”
(Vegeta)

RESUMO

SANTOS, R. R. **Um framework para agrupar funções com base no comportamento da comunicação de dados em plataformas multiprocessadas**. 2018. 102 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, 2018.

O aumento da demanda por sistemas computacionais mais eficientes para obter alto desempenho impôs novos desafios à comunidade de pesquisa, que precisou buscar por novas plataformas heterogêneas para grandes aplicações. Para utilizar todo o potencial dessas plataformas, pode-se agrupar a aplicação em grupos menores de modo que cada grupo seja executado em uma unidade de processamento específica, para reduzir o gargalo de comunicação, de acordo com o comportamento de comunicação durante a execução da aplicação. Com o propósito de oferecer um agrupamento mais eficiente, este projeto propõe a análise de agrupamento de uma aplicação levando em consideração não só o volume total de dados, mas também a distribuição desse volume durante o tempo de execução associado à restrição da banda e da taxa de transmissão. Embora alguns trabalhos considerem o volume total de dados para o agrupamento, não é evidenciado como esse volume é distribuído e como a restrição de banda afeta o agrupamento. Assim, neste projeto foi implementado um *framework* para sugerir um agrupamento considerando a distribuição do volume de comunicação e restrições de banda. Além disso, foi desenvolvido um módulo de extensão para a ferramenta externa MCProf (*Memory and Communication Profiler*) com o objetivo de obter a distribuição do volume de comunicação. A validação do *framework* foi realizada por meios de testes de agrupamentos de aplicações nos quais foram comparados o tempo de comunicação do agrupamento gerado pela execução do *framework* em relação aos resultados dos agrupamentos considerando os trabalhos da literatura. O uso desta abordagem apresentou um aumento no desempenho que variou de 1,117X a 2,621X para as aplicações usadas nos experimentos.

Palavras-chave: Agrupamento de funções, Plataformas heterogêneas, Distribuição de comunicação, Largura de banda variável.

ABSTRACT

SANTOS, R. R. **A framework for clustering functions based on the behavior of data communication on multiprocessed platforms**. 2018. 102 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, 2018.

The increased demand for more efficient computing systems to achieve high performance proposed new challenges to the research community, which needed to search for new heterogeneous platforms for large applications. To utilize the full potential of these platforms, the application can be grouped into small groups that runs on a specific processing unit to reduce the communication bottleneck according to the communication behavior during application execution . With the purpose of offering a more efficient clustering, this project proposes the analysis of clustering of an application taking into account not only the total volume of data, but also the distribution of that volume during the execution time associated to the band and restriction of rate transmission. Although some studies consider the total volume of data for the cluster, it is not clear how this volume is distributed and how the band constraint affects clustering. Thus, in this project was implemented a framework to suggest a cluster considering the distribution of the volume of communication and band restrictions. In addition, an extension module was developed for the external tool MCProf (Memory and Communication Profiler) in order to obtain the distribution of the communication. The validation of the framework was performed by clustering tests which used applications in which the communication time of the cluster generated by the execution of framework was compared to the results of the clusters considering the literature. The use of this approach showed an increase in performance ranging from 1.117X to 2.621X for the applications used in the experiments.

Keywords: Clustering functions, Heterogeneous platforms, Communication distribution, Variable bandwidth.

LISTA DE ILUSTRAÇÕES

Figura 1 – Diferença entre comunicação acumulada (a) e distribuída através do tempo (b)	25
Figura 2 – Exemplo de uma aplicação com 9 funções agrupadas em 3 grupos. Adaptado de (ASHRAF <i>et al.</i> , 2013).	28
Figura 3 – Algoritmo da técnica <i>K-means</i> . Fonte: Adaptado de (ZAKI; JR; MEIRA, 2014).	29
Figura 4 – Exemplo de grupos obtidos por meio de um agrupamento hierárquico. Fonte: Adaptado de (MIRKIN, 2012).	30
Figura 5 – Exemplo de uma plataforma heterogênea. Adaptado de (PHAM-QUOC <i>et al.</i> , 2015).	31
Figura 6 – Visão geral da ferramenta MCPProf. Fonte: Adaptado de (I. Ashraf and V.M. Sima and K.L.M. Bertels, 2015).	33
Figura 7 – Cálculo do <i>overhead</i> gerado pelo <i>framework</i> PIN. Adaptado de (INTEL, 2012)	34
Figura 8 – Exemplo de arquivo para a entrada da ferramenta PET. Fonte: Elaborado pelo autor.	35
Figura 9 – Esquema geral do <i>framework</i> desenvolvido.	42
Figura 10 – Algoritmo do <i>framework</i> desenvolvido.	43
Figura 11 – Exemplo de preenchimento do primeiro arquivo de configuração.	44
Figura 12 – Exemplo de matriz de comunicação de uma aplicação com 3 funções.	45
Figura 13 – Exemplo de utilização da janela de tempo.	46
Figura 14 – Saída obtida pela ferramenta PET, no qual é observado um agrupamento de 5 funções em 3 grupos. Fonte: Ferramenta PET	48
Figura 15 – Exemplo da comparação realizada neste projeto	49
Figura 16 – Exemplo de divisão de matrizes por um valor constante.	51
Figura 17 – Gráfico que representa o comportamento das funções de transferência através do tempo.	53
Figura 18 – Comunicação acumulada no final da aplicação.	55
Figura 19 – Distribuição da comunicação dos picos da aplicação KLT.	56
Figura 20 – Comparação de tempos entre cada função.	56
Figura 21 – Agrupamento ótimo encontrado para a aplicação KLT.	57
Figura 22 – Comunicação acumulada no final da aplicação.	57
Figura 23 – Distribuição dos picos de comunicação da aplicação Sparse.	58
Figura 24 – Comparação de tempos entre cada função.	59
Figura 25 – Agrupamento ótimo encontrado para a aplicação Sparse.	59

Figura 26 – Comunicação acumulada no final da aplicação.	60
Figura 27 – Distribuição dos picos de comunicação da aplicação Barnes.	61
Figura 28 – Comparação de tempos entre cada função.	61
Figura 29 – Agrupamento ótimo encontrado para a aplicação Barnes.	62
Figura 30 – Comunicação acumulada no final da aplicação.	62
Figura 31 – Distribuição dos picos de comunicação da aplicação FMM.	63
Figura 32 – Comparação de tempos entre cada função.	63
Figura 33 – Agrupamento ótimo encontrado para a aplicação FMM.	64
Figura 34 – Comunicação acumulada no final da aplicação.	64
Figura 35 – Distribuição dos picos de comunicação da aplicação Ocean.	65
Figura 36 – Agrupamento ótimo encontrado para a aplicação <i>Ocean</i>	65
Figura 37 – Comparação de tempos entre cada função.	66
Figura 38 – Esquema geral da primeira aplicação paralela emulada.	67
Figura 39 – Comparação dos tempos de comunicação obtidos através da emulação da aplicação paralela 1.	67
Figura 40 – Comparação dos tempos de comunicação obtidos através da emulação da aplicação paralela 2.	68
Figura 41 – Comparação dos tempos de comunicação obtidos através da emulação da aplicação paralela 3.	69
Figura 42 – Comparação dos tempos de comunicação obtidos através da emulação da aplicação paralela 4.	70
Figura 43 – Comparação dos tempos de comunicação obtidos através da emulação da aplicação paralela 5.	71

LISTA DE TABELAS

Tabela 1 – Comparação entre os trabalhos relacionados e o projeto proposto.	39
Tabela 2 – Média em segundos dos tempos encontrados no agrupamento da aplicação KLT utilizando 4 grupos.	55
Tabela 3 – Média em segundos dos tempos encontrados no agrupamento da aplicação Sparse.	58
Tabela 4 – Média em segundos dos tempos encontrados no agrupamento da aplicação Barnes.	60
Tabela 5 – Média em segundos dos tempos encontrados agrupando a aplicação FMM. .	62
Tabela 6 – Média em segundos dos tempos encontrados no agrupamento da aplicação <i>Ocean</i>	65
Tabela 7 – Resultados médios em segundos da emulação da aplicação paralela 1	67
Tabela 8 – Resultados médios em segundos da emulação da aplicação paralela 2	68
Tabela 9 – Resultados médios em segundos da emulação da aplicação paralela 3	69
Tabela 10 – Resultados médios em segundos da emulação da aplicação paralela 4	70
Tabela 11 – Resultados em segundos da emulação da aplicação paralela 5	71
Tabela 12 – Resultados em segundos dos tempos encontrados no agrupamento da aplicação KLT.	79
Tabela 13 – Resultados em segundos dos tempos encontrados no agrupamento da aplicação Sparse	80
Tabela 14 – Resultados em segundos dos tempos encontrados no agrupamento da aplicação Barnes	81
Tabela 15 – Resultados em segundos dos tempos encontrados agrupando a aplicação FMM	82
Tabela 16 – Resultados em segundos dos tempos encontrados no agrupamento da aplicação <i>Ocean</i>	83
Tabela 17 – Resultados médios em segundos da emulação da aplicação paralela 1	84
Tabela 18 – Resultados em segundos da emulação da aplicação paralela 2	85
Tabela 19 – Resultados em segundos da emulação da aplicação paralela 3	86
Tabela 20 – Resultados em segundos da emulação da aplicação paralela 4	87
Tabela 21 – Resultados em segundos da emulação da aplicação paralela 5	88
Tabela 22 – Valores de p obtidos comparando os resultados da comunicação acumulada.	89
Tabela 23 – Valores de p obtidos comparando os resultados da comunicação distribuída através do tempo.	89

Tabela 24 – Valores de p obtidos comparando os resultados da comunicação acumulada com os resultados da comunicação distribuída através do tempo.	89
--	----

LISTA DE ABREVIATURAS E SIGLAS

DCE	Data Collection Engines
dTDMA	dynamic Time Division Multiple Access
FMM	Fast Multiple Method
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
KISS	Keep it Simple Stupid
KLT	Kanade-Lucas-Tomasi feature tracker
MAT	Memory Access Tracer
MCProf	Memory and Communication Profiler
NoC	Network-on-Chip
PET	Partition Evaluation Tool
QUAD	Quantitative Usage Analysis of Data
RAM	Random Access Memory
SIFT	Scale-Invariant Feature Transform
SMU	Shadow Memory Unit
SoC	System-on-a-Chip
SPLASH-2	Stanford Parallel Applications for Shared-memory

SUMÁRIO

1	INTRODUÇÃO	23
1.1	Contextualização	23
1.2	Objetivo	24
1.3	Justificativa	24
1.4	Organização do trabalho	25
2	REVISÃO DE LITERATURA	27
2.1	Considerações iniciais	27
2.2	Agrupamento	27
2.2.1	<i>Técnicas de agrupamento</i>	28
2.3	Meio de comunicação	30
2.4	Linguagem de programação e ferramentas externas	31
2.4.1	<i>Linguagem R</i>	31
2.4.2	<i>Ferramenta MCPProf</i>	32
2.4.2.1	<i>Framework Intel PIN</i>	33
2.4.3	<i>Ferramenta PET</i>	34
2.5	Trabalhos relacionados	37
2.5.1	<i>Referentes à comunicação de uma aplicação.</i>	37
2.5.2	<i>Referentes aos meios de comunicação</i>	38
2.6	Considerações finais	39
3	FRAMEWORK DESENVOLVIDO	41
3.1	Considerações iniciais	41
3.2	Framework desenvolvido	42
3.2.1	<i>Inicialização do framework</i>	43
3.2.2	<i>Execução da ferramenta MCPProf</i>	45
3.2.2.1	<i>Módulo de extensão</i>	46
3.2.3	<i>Obtenção das matrizes de tempo</i>	47
3.2.4	<i>Execução da ferramenta PET</i>	47
3.2.5	<i>Análise dos agrupamentos</i>	48
3.3	Considerações finais	49
4	RESULTADOS E ANÁLISES	51
4.1	Considerações iniciais	51

4.2	Aplicação KLT	54
4.3	Aplicação Sparse 1.3a	57
4.4	Aplicações do <i>Benchmark SPLASH-2</i>	59
4.4.1	<i>Aplicação Barnes</i>	59
4.4.2	<i>Aplicação FMM</i>	61
4.4.3	<i>Aplicação Ocean</i>	64
4.5	Emulação de aplicação paralela	66
4.5.1	<i>Emulação da aplicação paralela 1</i>	66
4.5.2	<i>Emulação da aplicação paralela 2</i>	68
4.5.3	<i>Emulação da aplicação paralela 3</i>	68
4.5.4	<i>Emulação da aplicação paralela 4</i>	69
4.5.5	<i>Emulação da aplicação paralela 5</i>	70
4.6	Considerações Finais	71
5	CONCLUSÃO	73
5.1	Limitações e Trabalhos futuros	74
REFERÊNCIAS		75
APÊNDICE A	TABELAS COM OS TEMPOS OBTIDOS	79
APÊNDICE B	TABELAS COM OS VALORES DE P OBTIDOS	89
APÊNDICE C	CÓDIGO FONTE DO <i>FRAMEWORK</i> DESENVOLVIDO	91
APÊNDICE D	CÓDIGO FONTE DO MÓDULO DE EXTENSÃO DESENVOLVIDO	101

INTRODUÇÃO

1.1 Contextualização

Muitas aplicações, denominadas de *supercomputing applications* ou *superapplications*, demandam elevado poder computacional com tendência de crescimento (KIRK; HWU, 2010). Devido a isso, severas mudanças nas arquiteturas dos computadores têm ocorrido. Os projetos passaram a utilizar paralelismo em larga escala, incluindo plataformas heterogêneas compostas por aceleradores customizados, visando o ganho de desempenho computacional (BORKAR *et al.*, 2005; BORKAR; CHIEN, 2011; HOROWITZ; DALLY, 2004).

Diferente das plataformas homogêneas, que possuem várias unidades de processamento idênticas, as plataformas heterogêneas buscam explorar as características de cada elemento de processamento de acordo com o perfil do código a ser executado. GPUs e FPGAs são tradicionalmente utilizados como aceleradores e a escolha de qual parte do código é o mais adequado não é uma tarefa trivial devida a complexidade associada à exploração do espaço de projeto.

Há uma série de trabalhos que evidenciam o potencial de aceleradores de acordo com o perfil do código. No trabalho de Che *et al.* (2008), por exemplo, é traçado um perfil de aplicações que apresentam melhor desempenho em GPUs e em FPGAs baseado no custo de desenvolvimento, desempenho e restrições de *hardware*. Por exemplo, códigos com o perfil do método de eliminação de Gauss apresentam um desempenho melhor em GPUs pelo fato de não existir dependências no fluxo de dados, o que permite que a computação seja feita em paralelo. Por outro lado, códigos com o perfil do algoritmo *Data Encryption Standard* (DES) são mais adequados para FPGAs devido a grande quantidade de operações em nível de *bits*.

Códigos relacionados ao processamento de imagens também podem tirar proveito de arquiteturas FPGAs. No trabalho de (BONATO; MARQUES; CONSTANTINIDES, 2008), por exemplo, é proposta uma arquitetura de *hardware* paralela em FPGA para o processamento de

um algoritmo denominado SIFT, que é utilizado para a extração de características de imagens. A arquitetura gerada atua como um acelerador no processamento de imagens.

Uma forma de explorar o mapeamento em plataformas multiprocessadas é por meio da análise da comunicação entre trechos de código. Esses trechos de código, usualmente associado em nível funções, podem ser agrupados de acordo com a demanda da comunicação entre eles para que posteriormente cada grupo seja mapeado ao elemento de processamento mais adequado para o perfil de código. Porém, esse agrupamento não é uma tarefa trivial, pois além de analisar o código *offline*, é preciso entender a dinâmica da movimentação dos dados durante a execução da aplicação e conhecer o potencial dos compiladores para seus processadores dedicados.

O agrupamento das funções de uma aplicação pode ser realizado manualmente ou automaticamente. Na forma manual, o usuário utiliza ferramentas de *profile*, como Gprof (GRAHAM; KESSLER; MCKUSICK, 1982) ou Valgrind (NETHERCOTE; SEWARD, 2007), para obter informações que auxiliem na identificação de gargalos da aplicação, de modo que o usuário possa agrupar as funções como desejar. Já na forma automática, o usuário insere as informações da aplicação em ferramentas que retornam um agrupamento para as funções.

Entretanto, para realizar um bom agrupamento, deve-se analisar a capacidade da comunicação interna de uma plataforma heterogênea. Dependendo do meio de comunicação, o tempo de transmissão de dados pode variar, por exemplo, caso o meio de comunicação seja utilizado por muitos dispositivos, a comunicação de dados será mais lenta. A otimização desse tipo de comunicação beneficiaria a execução da aplicação agrupada, pois o meio de comunicação seria dedicado à essa aplicação.

1.2 Objetivo

O objetivo deste projeto é o desenvolvimento de um *framework* para agrupar funções em conformidade com o comportamento da comunicação de dados entre as mesmas durante a execução de uma aplicação, considerando para isso sistemas computacionais compostos por mais de uma unidade de processamento com características distintas entre si. O agrupamento deverá reduzir a demanda pelo uso do meio de comunicação entre as unidades de processamento e por consequência permitir o aumento de desempenho de tais sistemas.

1.3 Justificativa

O agrupamento eficiente de uma aplicação é fundamental para que seu mapeamento seja utilizado em plataformas heterogêneas. A decisão de qual função agrupar deve ocorrer de maneira precisa para que não ocorra perda no desempenho obtido. Além disso, deve-se considerar o perfil de comunicação da aplicação, para permitir a análise dos gargalos existentes na aplicação. Também deve-se considerar a largura de banda e a taxa de transmissão de dados disponível no

meio de comunicação, para que seja possível analisar como a aplicação se comportará ao utilizar um acelerador.

Atualmente, os trabalhos propostos na literatura só consideram o volume total da comunicação de uma aplicação finalizada e não analisam o meio de comunicação no qual a aplicação foi executada. Portanto, é importante verificar como a comunicação se comporta ao longo da execução da aplicação, pois é relevante analisar se os picos de comunicação ocorrem sequencialmente ou se a comunicação representada pelo pico é distribuída uniformemente durante a execução.

Na Figura 1, por exemplo, temos a comunicação das funções F1, F2 e F3 representadas no formato matricial. Em (a) é representada uma matriz de comunicação acumulada obtida no fim da execução de uma aplicação hipotética, na qual a função F1 troca 11 *bytes* com a função F2 e 8 *bytes* com a função F3 e a função F2 troca 16 *bytes* com a função F3. Essa matriz é resultado da soma das matrizes apresentadas em (b), sendo que cada matriz representam a comunicação distribuída através do tempo nos instantes T0, T1, T2 e T3.

				T0				T1			
					F1	F2	F3		F1	F2	F3
				F1	0	5	2	F1	0	0	2
				F2	0	0	4	F2	0	0	5
				F3	0	0	0	F3	0	0	0
				T2				T3			
					F1	F2	F3		F1	F2	F3
				F1	0	6	2	F1	0	0	2
				F2	0	0	7	F2	0	0	0
				F3	0	0	0	F3	0	0	0

	F1	F2	F3
F1	0	11	8
F2	0	0	16
F3	0	0	0

(a)

(b)

Figura 1 – Diferença entre comunicação acumulada (a) e distribuída através do tempo (b)

1.4 Organização do trabalho

Esta dissertação de mestrado está organizada da seguinte forma: no Capítulo 2 é apresentada a revisão da literatura necessária para a compreensão deste trabalho, os métodos e modelos considerados no trabalho e os trabalhos relacionados propostos na literatura. No Capítulo 3 é detalhado o *framework* proposto. No Capítulo 4 são apresentados os testes realizados para validar o *framework* e o resultado desses testes. Por fim, no Capítulo 5 é apresentada a conclusão, as principais contribuições, as limitações e os trabalhos futuros.

REVISÃO DE LITERATURA

2.1 Considerações iniciais

Neste capítulo são apresentados os conceitos necessários para a compreensão deste projeto. Na Seção 2.2 são apresentados os conceitos de agrupamento. Na Seção 2.4 é descrita a linguagem de programação e as ferramentas externas utilizadas para o desenvolvimento deste projeto. Finalmente, na Seção 2.5 são apresentados os trabalhos relacionados a este projeto.

2.2 Agrupamento

O termo agrupamento pode ser definido como a tarefa de aglomerar pontos semelhantes entre si em um mesmo grupo e separar em diferentes grupos pontos que são diferentes entre si (BERKHIN, 2006). O resultado de um agrupamento é denominado grupo. Neste projeto, um grupo representa uma unidade de processamento independente que existe em uma plataforma multiprocessada, sendo que cada grupo se comunica com outro grupo através de um meio de comunicação.

Segundo I. Ashraf and V.M. Sima and K.L.M. Bertels (2015), uma aplicação deve ser particionada em pequenos grupos para que seja possível explorar todo o paralelismo existente em uma plataforma multiprocessada, pois cada grupo é mapeado para um núcleo disponível na plataforma, de modo a alcançar um aumento no desempenho da aplicação.

O agrupamento de uma aplicação diminui o volume de dados trocados entre suas funções, pois durante o agrupamento busca-se diminuir a comunicação entre os grupos e maximizar a comunicação interna entre eles, de modo que em um mesmo grupo estejam as funções que apresentam grande troca de dados entre si. Ao mapear um grupo de funções em um processador, a comunicação intragrupo não necessita de um meio de comunicação externo entre os processadores, pois pode ocorrer através de memória compartilhada. Na Figura 2 é apresentado

o exemplo de um agrupamento realizado em uma aplicação com 9 funções.

Figura 2 – Exemplo de uma aplicação com 9 funções agrupadas em 3 grupos. Adaptado de (ASHRAF *et al.*, 2013).

Quando um agrupamento é realizado, deve-se verificar se o grupo encontrado é eficiente. Segundo (ASHRAF *et al.*, 2013), uma pequena parte das funções contribuem para o tempo total de execução da aplicação e entre essas funções, uma parte mais restrita de funções são responsáveis pela comunicação de dados na aplicação. Logo, é importante verificar se as funções corretas foram agrupadas. De acordo com (I. Ashraf and V.M. Sima and K.L.M. Bertels, 2015), agrupar as funções de uma aplicação em nível de processadores para explorar seu paralelismo é uma tarefa crítica que pode diminuir o desempenho que a aplicação apresentaria, pois o *overhead* do meio de comunicação seria elevado, o que tornaria o tempo de comunicação maior que o tempo de execução das funções.

Outra fragilidade que pode ser observada refere-se ao número de possibilidades de combinações para um agrupamento. Segundo (ZAKI; JR; MEIRA, 2014), há um total de k^n possíveis grupos, no qual k representa o número de grupos e n o número de funções da aplicação. Entretanto, o valor de k^n apresenta grupos com similaridades entre si, pois um grupo formado pelas funções A e B é similar a outro que apresenta as funções B e A. Logo, segundo (ZAKI; JR; MEIRA, 2014), o número de grupos distintos entre si é apresentado pelos Números de Segundo Tipo de *Stirling*, que é apresentado na equação (2.1).

$$S(n, k) = \frac{1}{k!} \sum_{t=0}^k (-1)^t \binom{k}{t} (k-t)^n \quad (2.1)$$

A seguir são apresentadas as técnicas de agrupamento existentes na literatura.

2.2.1 Técnicas de agrupamento

De acordo com Zaki, Jr e Meira (2014), as técnicas de agrupamento podem ser divididas em três categorias: agrupamento relativo, agrupamento hierárquico e agrupamento espectral.

No agrupamento relativo um conjunto de dados com n valores e k grupos desejados, são agrupados em grupos denominados $C = \{C_1, C_2, \dots, C_k\}$. Assim, cada grupo C apresenta um valor que representa a média de todos os dados existentes no grupo. Para o cálculo do agrupamento relativo é utilizada a técnica de particionamento denominada *K-means*.

Nessa técnica são escolhidos aleatoriamente k pontos entre os existentes no conjunto de dados, sendo que esses k pontos são utilizados como valores centrais nos K grupos. Posteriormente, é realizado um agrupamento de modo que cada grupo tenha valores semelhantes ao seu valor central. Em seguida, é calculada a média dos valores existentes no grupo para atualizar o valor central, que corresponde a média desses valores. Esse processo é repetido até seja encontrado um agrupamento ótimo. Segundo (XU; WUNSCH, 2005), a técnica *K-means* apresenta complexidade $O(kn)$, na qual k representa o número de grupos e n representa o número de pontos a serem agrupados. Na Figura 3 é apresentado o algoritmo dessa técnica, no qual D corresponde ao conjunto de dados que se deseja agrupar, C_i representa os grupos que serão formados e μ_i corresponde aos valores centrais de cada grupo.

```

K-MEANS ( $D, k, \epsilon$ ):
1  $t = 0$ 
2 Inicializar  $k$  centróides aleatoriamente:  $\mu_1^t, \mu_2^t, \dots, \mu_k^t \in \mathbb{R}^d$ 
3 repeat
4    $t \leftarrow t + 1$ 
5    $C_j \leftarrow \emptyset$  for all  $j = 1, \dots, k$ 
   // Passo para formação dos grupos
6   foreach  $\mathbf{x}_j \in D$  do
7      $j^* \leftarrow \arg \min_i \{ \|\mathbf{x}_j - \mu_i^t\|^2 \}$  // Assimilar  $\mathbf{x}_j$  para o centróide mais próximo
8      $C_{j^*} \leftarrow C_{j^*} \cup \{\mathbf{x}_j\}$ 
   // Atualização do centróide
9   foreach  $i = 1$  to  $k$  do
10     $\mu_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$ 
11 until  $\sum_{i=1}^k \|\mu_i^t - \mu_i^{t-1}\|^2 \leq \epsilon$ 

```

Figura 3 – Algoritmo da técnica *K-means*. Fonte: Adaptado de (ZAKI; JR; MEIRA, 2014).

K-means é uma técnica de particionamento muito conhecida. O trabalho de Pelleg, Moore *et al.* (2000) apresenta uma variação do *K-means*, denominada *X-means*. Essa técnica utiliza o *K-means* para obter um agrupamento inicial. Posteriormente, o valor central de cada grupo encontrado é dividido pela metade para aplicar a técnica *K-means* localmente. Com isso, é possível verificar o número ideal de grupos para um determinado conjunto de dados.

De acordo com (ZAKI; JR; MEIRA, 2014), o agrupamento hierárquico pode ser visualizado por meio de uma árvore hierárquica, no qual cada folha representa um dado em seu próprio grupo e a raiz representa todos os pontos em um único grupo. Esse agrupamento apresenta dois métodos principais para realizar o agrupamento, que são o método aglomerativo e o método divisivo.

O método aglomerativo consiste em aplicar uma estratégia *bottom-up* para construir

a árvore, sendo que cada ponto começa em uma folha e em seguida são unidos de acordo com sua similaridade até chegarem ao mesmo grupo. O método divisivo é o oposto do método aglomerativo, pois é aplicada uma estratégia *top-down*, na qual todos os pontos começam na raiz e em seguida são sucessivamente divididos até que cada um esteja em seu próprio grupo.

Segundo (XU; WUNSCH, 2005), o agrupamento hierárquico apresenta complexidade $O(n^2)$, no qual n representa o número de pontos a serem agrupados. Na Figura 4 é apresentado o exemplo de um agrupamento hierárquico, no qual são agrupados 8 pontos, sendo que o grupo B agrupa os pontos 5 e 6 enquanto o grupo E agrupa o ponto 4 e o grupo B.

Figura 4 – Exemplo de grupos obtidos por meio de um agrupamento hierárquico. Fonte: Adaptado de (MIRKIN, 2012).

Por fim, o agrupamento espectral pode ser utilizado para agrupar dados através de um grafo e de uma matriz que contém a similaridade entre cada vértice do grafo (SINGH, 2010), utilizando a matriz de similaridade como peso das arestas do grafo. Segundo (ZAKI; JR; MEIRA, 2014), o agrupamento espectral é similar ao método divisivo do agrupamento hierárquico, pois utiliza uma matriz de similaridade entre os nós como peso.

2.3 Meio de comunicação

Uma plataforma heterogênea usualmente contém um processador de propósito geral e um ou mais elementos de processamentos que são utilizados como aceleradores, que pode ser tanto em *hardware* quanto em *software*. Na Figura 5 é apresentado o esquema geral de uma plataforma heterogênea, que possui um processador de propósito geral denominado *host*, dispositivos de *hardware*, que são utilizados como aceleradores, denominados *kernels*, e uma infraestrutura de comunicação utilizada para troca de dados internamente na plataforma.

Um meio de comunicação muito utilizado é o barramento, que é um meio físico de comunicação utilizado para conectar vários dispositivos em uma plataforma heterogênea. Segundo (SANCHEZ; MICHELOGIANNAKIS; KOZYRAKIS, 2010) são simples, eficientes e de fácil utilização, mas possuem uma baixa *performance* devido a sua comunicação serializada. Neste projeto, considera-se que o meio de comunicação utilizado é um barramento *PCI Express*, pois é o meio de comunicação mais rápido disponível atualmente (BITTNER, 2012). Um barramento

Figura 5 – Exemplo de uma plataforma heterogênea. Adaptado de (PHAM-QUOC *et al.*, 2015).

PCI Express apresenta canais bidirecionais que são utilizados para a comunicação de dados, o que significa que um dispositivo pode tanto enviar como receber dados por meio desse canal. O número de canais nesse barramento varia entre 1 e 32, o que faz com que a velocidade de transferência de dados também varie dependendo do número de canais utilizados.

Um outro meio de comunicação é o NoC. Segundo Kumar *et al.* (2002), um NoC é um conjunto de *switchs* interconectados localmente entre si de modo a obter uma rede de *switches*. O NoC apresenta uma alta *performance*, no entanto possui um elevado custo de desenvolvimento devido a limitações de espaço em um *chip*.

2.4 Linguagem de programação e ferramentas externas

Nesta seção é apresentada a linguagem de programação R e as ferramentas MCProf e PET utilizadas para o desenvolvimento do *framework*. A MCProf retorna a matriz de comunicação total de uma aplicação durante a execução, enquanto a PET utiliza essa matriz para gerar um agrupamento.

2.4.1 Linguagem R

A linguagem de programação R¹ foi projetada para auxiliar na computação estatística e em trabalhos que envolvem grafos. Essa linguagem apresenta uma grande variedade de operadores que possibilitam a manipulação de matrizes, dessa forma foi escolhida para a implementação desse projeto, pois no *framework* desenvolvido houve a necessidade de manipular matrizes de comunicação ao longo do tempo. Além disso, sua característica *open source* também foi um atrativo para a utilização dessa linguagem, pois facilitaria a replicabilidade do projeto desenvolvido.

¹ <https://www.r-project.org/>

2.4.2 Ferramenta MCProf

MCProf² é uma ferramenta de *profile* dinâmico desenvolvida por (I. Ashraf and V.M. Sima and K.L.M. Bertels, 2015) para a análise do volume de comunicação existente entre funções de uma aplicação e entre funções e a memória. Segundo os autores, o desenvolvimento da MCProf foi motivado pelas fragilidades encontradas nas ferramentas de análise estática, pois essas ferramentas só podem ser utilizadas em aplicações bem estruturadas e assim tornam-se inapropriadas para uso em aplicações do mundo real. Além disso, os autores consideram também que as ferramentas de análise estática não apresentam informações sobre o volume que dados comunicados entre funções da aplicação, que é utilizado para realizar um agrupamento.

A MCProf foi desenvolvida para a análise dinâmica da comunicação interna de aplicações. É importante ressaltar que embora na literatura existam outras ferramentas de análise dinâmica, como, por exemplo, a ferramenta Valgrind (SEWARD; NETHERCOTE, 2005), tais ferramentas apresentam um grande *overhead* quando comparadas às de análise estática devido à carga extra de trabalho que essas ferramentas realizam para a análise das aplicações.

Para a inicialização da MCProf é disponibilizado para o usuário um arquivo nomeado como *Makefile*, no qual deve ser especificado o nome da aplicação a ser analisada. A aplicação é compilada e executada na MCProf com o objetivo de obter as informações desejadas pelo usuário, conforme as possibilidades oferecidas pela ferramenta. Para realizar a análise e execução da aplicação, a MCProf utiliza alguns parâmetros existentes no arquivo *Makefile*. Neste projeto foram escolhidos dois parâmetros; sendo um para rastrear funções de alocação e desalocação de memória e o outro para utilizar uma pilha que auxilia na análise das funções da aplicação.

A MCProf é dividida em três partes, conforme destacados em vermelho na Figura 6. Cada parte é utilizada para a execução de uma tarefa específica escolhida pelo usuário. Somente uma tarefa pode ser escolhida por vez. Cada uma das partes é detalhada a seguir.

- MAT : responsável por realizar uma instrumentação em nível de instrução, rotina e imagem e pela análise das leituras e escritas realizadas pela aplicação na memória com o auxílio do *framework* PIN, apresentado na Subseção 2.4.2.1. Por meio de uma análise em nível de instrução é realizado um rastreamento na quantidade de leituras e escritas que uma função realizou na memória. Um rastreamento em nível de rotina é realizado para manter a função atual que está sendo analisada.
- DCE : responsável pela execução do módulo escolhido pelo usuário, sendo que cada módulo apresenta uma funcionalidade específica. A seguir são destacados os três módulos implementados atualmente na MCProf, sendo que esses podem ser ser expandidos:
 - Módulo 1: apresenta a quantidade de leituras e escritas que cada função realiza na memória;

² <https://bitbucket.org/imranashraf/mcprof>

Figura 6 – Visão geral da ferramenta MCProf. Fonte: Adaptado de (I. Ashraf and V.M. Sima and K.L.M. Bertels, 2015).

- Módulo 2: esse módulo é o responsável pela obtenção das matrizes de comunicação, pois apresenta, em formato de matriz, a quantidade total de dados trocados entre as funções da aplicação. Além disso, também apresenta a porcentagem de tempo de execução de cada função na aplicação;
- Módulo 3: apresenta uma lista de chamadas de cada função, o que permite a análise de como cada função acessa a memória.
- SMU : responsável pela gravação de cada *byte* produzido. Em cada processo de escrita, informações como endereço, tamanho e ID de *thread* são armazenadas em uma pilha contendo o escritor (produtor) de cada *byte* na unidade da memória auxiliar. No processo de leitura de um *byte*, o consumidor do *byte* é a função em execução, enquanto que o escritor é recuperado da memória auxiliar.

O MAT envia para a DCE os acessos referentes à leitura/escrita realizados na memória. Por sua vez, a SMU é utilizada para analisar o produtor e consumidor dos dados enviados para a DCE. Posteriormente, a DCE retorna um conjunto de informações referentes à quantidade de dados comunicados, bem como a porcentagem de tempo de execução de cada função.

2.4.2.1 Framework Intel PIN

PIN (INTEL, 2012) é um *framework* dinâmico desenvolvido pela Intel para arquiteturas x86-64, que permite a criação de ferramentas que analisam aplicações dinamicamente e os analisa em tempo de execução, evitando a necessidade de recompilá-los.

Atualmente, o PIN oferece três funções principais: instrumentação, análise e retorno de chamada. A instrumentação ocorre quando a aplicação é carregada na memória, inserindo assim as rotinas de análise. As rotinas de análise são chamadas após o programa ser recompilado por um compilador do tipo *Just-In-Time*, fazendo com que instruções provenientes do PIN sejam executadas. Por fim, a rotina de retorno é chamada quando a condição de retorno estiver satisfeita, finalizando a execução do PIN.

Uma limitação do PIN é seu elevado *overhead*, dessa forma, as aplicações desenvolvidas com o mesmo devem ser otimizadas. Segundo (INTEL, 2012), o *overhead* pode variar entre 5% até 50% do tempo de execução da aplicação. Na Figura 7 é apresentado o cálculo do *overhead* do PIN.

$$\begin{aligned}
 \text{Overhead total} &= \text{Overhead PIN} + \text{Overhead da Ferramenta} \\
 &= \underbrace{\text{Overhead de Instrumentação de Rotinas} + \text{Overhead de Análise de Rotinas}}_{\text{Overhead PIN}} + \text{Overhead da Ferramenta} \\
 &= \underbrace{\text{Frequência de Chamadas da Análise de Rotina} \times \text{Trabalho Requerido da Análise de Rotina}}_{\text{Trabalho Requerido para a Análise de Rotinas}} + \text{Trabalho Realizado dentro da Análise de Rotinas}
 \end{aligned}$$

Figura 7 – Cálculo do *overhead* gerado pelo *framework* PIN. Adaptado de (INTEL, 2012)

Na ferramenta MCProf, o PIN é utilizado para rastrear a comunicação na execução de uma aplicação. Durante a compilação da aplicação, comandos específicos do PIN são inseridos na aplicação com o propósito de especificar o que deve ser analisado pelo *framework* PIN. Assim, no mesmo instante que a aplicação é executada, o PIN a analisa e armazena a comunicação interna da aplicação. Posteriormente a MCProf utiliza essas informações e as retorna em formato de matriz de comunicação.

2.4.3 Ferramenta PET

PET³ é uma ferramenta desenvolvida para a avaliação da qualidade de um agrupamento ótimo para uma aplicação, por meio da comparação de algoritmos de agrupamento (ASHRAF *et al.*, 2013). A motivação para o desenvolvimento da PET se deve ao fato de que encontrar uma partição ótima é um problema NP-complexo. Embora algumas heurísticas possam ser utilizadas para a resolução desse problema, muitas vezes o agrupamento resultante pode não ser ótimo. Assim, os autores notaram a necessidade de uma ferramenta capaz de avaliar a qualidade do agrupamento.

³ <http://imranashraf.github.io/PET/>

Para a utilização da PET, o usuário deve informar a quantidade de funções que a aplicação possui, o número de grupos desejados, a porcentagem de execução de cada função e a matriz de comunicação acumulada da aplicação. A porcentagem de execução de cada função é utilizada para o cálculo do custo de balanceamento e para distribuir de maneira uniforme as funções nos grupos. A matriz de comunicação acumulada é utilizada para o cálculo do custo, em *bytes*, de cada agrupamento.

Na Figura 8 é apresentado um exemplo da entrada da ferramenta PET, no qual o valor 5 indica a quantidade de funções da aplicação, o valor 2 indica a quantidade de grupos desejados para agrupar as funções da aplicação, os valores abaixo da palavra *Functions* representam a porcentagem de execução de cada função e os valores abaixo da palavra *Edges* representam a comunicação acumulada das funções da aplicação. É importante ressaltar que a soma das porcentagens de execução de cada função não corresponde a 100%, pois as funções referentes à alocação/desalocação de memória, impressões de resultados não estão contabilizadas nesse somatório.

5	2			
<i>Functions</i>				
10				
3				
48				
22				
14				
<i>Edges</i>				
6,97	5,19	4,03	5,18	6,88
5,05	8,97	5,78	6,26	7,60
6,32	9,84	7,70	6,68	3,81
4,31	7,67	7,59	3,62	1,58
5,39	2,99	3,23	7,44	8,32

Figura 8 – Exemplo de arquivo para a entrada da ferramenta PET. Fonte: Elaborado pelo autor.

A partir das informações fornecidas pelo usuário, a ferramenta considera a aplicação como sendo um grafo no qual cada vértice representa uma função e cada aresta representa a comunicação entre as mesmas. A partir desse grafo, é desenvolvida uma fórmula para calcular o custo de agrupar essa aplicação e o custo de balanceamento para que todos os grupos apresentem um número similar de funções agrupadas.

Posteriormente, a ferramenta encontra o melhor agrupamento para a aplicação do usuário. Para isso, a ferramenta possui cinco algoritmos utilizados para descobrir o melhor agrupamento, sendo que após a execução de cada um é apresentado qual o melhor agrupamento encontrado por cada algoritmo, bem como seu custo computacional em *bytes*. Os algoritmos são: Força Bruta, Busca Heurística, Busca Evolutiva, Recozimento Simulado e Busca Tabu.

O algoritmo de Força Bruta analisa todos os agrupamentos possíveis para encontrar qual

o melhor particionamento para a aplicação do usuário. Entretanto, como apresentado na Seção 2.2, uma aplicação com n funções e k grupos possui um total de k^n agrupamentos possíveis, tornando a execução do algoritmo de Força Bruta inviável para grandes aplicações, devido a sua complexidade exponencial. Por exemplo, a menor aplicação utilizada para validar esse projeto possui 42 funções e foi dividida em 4 grupos, sendo que através do Força Bruta seria necessário gerar mais de 834 trilhões de agrupamentos.

O algoritmo da Busca Heurística é utilizado para encontrar de maneira rápida um agrupamento. No entanto, o agrupamento encontrado pode não ser o melhor possível, pois como esse é o primeiro a ser encontrado, já é considerado como ótimo e não é realizada uma análise de sua qualidade.

Já o algoritmo de Busca Evolutiva utiliza elementos de biologia como mutação e recombinação para encontrar a melhor solução. Inicialmente, é utilizado o algoritmo *KISS* (MARSAGLIA; ZAMAN, 1993) para gerar três valores aleatórios que são combinados matematicamente para gerar um único valor aleatório que é utilizado pelo algoritmo de Busca Evolutiva. Após encontrar esse valor, o algoritmo verifica se esse valor aleatório é maior que um valor limite pré-definido, e caso seja, o algoritmo executa uma função de mutação e recombinação no agrupamento encontrado como forma de adquirir um agrupamento melhor que o previamente encontrado.

O algoritmo Recozimento Simulado é um algoritmo derivado da mecânica estatística que consiste em encontrar um agrupamento através de uma sequência de iterações, na qual cada iteração altera o agrupamento atual e desenvolve um novo agrupamento com semelhanças com agrupamento anterior. Quando um novo agrupamento é encontrado, uma função de custo é utilizada para analisá-lo, e caso seu custo computacional seja melhor que o custo atual, o agrupamento atual é substituído pelo novo agrupamento (PHAM; KARABOGA, 2012).

Finalmente, o algoritmo Busca Tabu (GLOVER, 1989) é um algoritmo utilizado para solucionar problemas de otimização combinacional. Segundo (GOMES, 2009), a heurística Tabu encontra várias soluções, denominadas tabu, e as armazena em uma lista. A melhor solução da lista é apontada como a solução ótima. A quantidade de soluções possíveis é definida pelo usuário da heurística.

Quando todos esses algoritmos encontram um agrupamento, o custo de cada agrupamento encontrado é avaliado pela função de custo, para que seja possível indicar qual o melhor agrupamento encontrado.

Essa ferramenta foi escolhida para ser utilizada neste projeto devido à qualidade dos agrupamentos encontrados. Entretanto, somente o algoritmo de Busca Evolutiva foi utilizado, pois foi o único algoritmo que conseguiu retornar agrupamentos corretos provavelmente devido ao grande número de funções das aplicações utilizadas. O algoritmo Força Bruta é inviável para encontrar um agrupamento e os outros algoritmos apresentaram agrupamentos em que o número

de grupos definidos pelo usuário não foi respeitado.

2.5 Trabalhos relacionados

Nesta seção são apresentados os trabalhos relacionados a este projeto. Como o projeto aborda a comunicação de uma aplicação e as características do meio de comunicação para realizar um agrupamento, os trabalhos relacionados estão divididos em duas subseções. Na Subseção 2.5.1 são apresentados os trabalhos que abordam a comunicação interna de uma aplicação e na Subseção 2.5.2 são apresentados os trabalhos sobre meios de comunicação.

2.5.1 Referentes à comunicação de uma aplicação.

Ostadzadeh *et al.* (2010) apresenta a ferramenta QUAD, que é utilizada para obter o volume de *bytes* trocados entre funções de uma aplicação. Essa ferramenta foi desenvolvida com o intuito de detectar quais funções de uma aplicação apresentam dependências de dados entre si, pois segundo os autores, ferramentas de *profile* não apresentam uma distinção entre tempo computacional e tempo de acesso à memória, de modo que não possam ser utilizadas para determinar gargalos da aplicação. Uma aplicação de processamento de vídeo foi utilizada para validar a aplicação. Com a ferramenta, foi possível encontrar a função que apresenta o maior volume de comunicação de dados para modificá-la e reduzir o volume de dados comunicados na aplicação. Essa ferramenta foi utilizada como base para o desenvolvimento da MCProf.

Pham-Quoc *et al.* (2015) propõem o desenvolvimento de um modelo automatizado para a implementação de uma aplicação em um acelerador de hardware interconectado com o sistema computacional. Inicialmente, utiliza-se a ferramenta QUAD e o *profiler* Gprof (GRAHAM; KESSLER; MCKUSICK, 1982) para analisar respectivamente, os padrões de comunicação e o custo computacional de cada função. Quando essas ferramentas retornam seus dados, uma ferramenta de agrupamento é utilizada para agrupar a aplicação em diversas partes. Após o agrupamento, é analisada qual parte da aplicação deve ser executada em *hardware* e *software*. Essa separação é realizada para que ocorra um aumento no desempenho do sistema.

Posteriormente, é implementado um meio de comunicação híbrido entre *hardware* e *software*, para que as partes da aplicação possam se comunicar, caso seja necessário uma troca de dados entre elas. Também é realizada uma reconfiguração no *hardware* para que todas partes que serão executadas da aplicação sejam aceleradas. Finalmente, todos os componentes são sintetizados e implementados na plataforma de destino, na qual a aplicação é executada entre *hardware* e *software*, sendo que toda a comunicação é realizada por meio da comunicação híbrida desenvolvida anteriormente.

O modelo proposto foi executado em duas aplicações escolhidas pelos autores. As aplicações utilizadas nos testes foram inicialmente executadas somente em *software* para adquirir seu tempo de execução. Em seguida, foram executadas em uma plataforma híbrida denominada

baseline, mas sem a utilização do meio de comunicação híbrido. Posteriormente, as aplicações foram executadas no modelo proposto. Finalmente, foi realizada uma comparação entre os tempos de execução gerado em cada sistema. Ao executar as aplicações no modelo proposto, houve uma redução no consumo de energia em relação aos demais.

O trabalho de [Zhao e Karypis \(2002\)](#) considera que as técnicas de agrupamento hierárquicas apresentam um melhor resultado em relação às técnicas aglomerativas. Para validar essa consideração são utilizadas doze conjuntos de dados para avaliar a *performance* de várias técnicas de agrupamento hierárquico e agrupamento aglomerativo. Para isso, cada conjunto de dados foi agrupado utilizando uma técnica hierárquica e aglomerativa e os resultados foram avaliados para verificar a qualidade dos mesmos. Após analisá-los, descobriu-se que os algoritmos aglomerativos apresentam um pior resultado devido ao erro que pode ocorrer no início do agrupamento.

Outros trabalhos relacionados que podem ser citados são as ferramenta MCProf e PET. Como descrito anteriormente, a MCProf é utilizada para obter a comunicação total da aplicação após sua execução, entretanto não apresenta como essa comunicação é distribuída durante a execução da aplicação. Enquanto a PET é utilizada para encontrar um agrupamento ótimo através de cinco algoritmos, porém só considera a comunicação total da aplicação para isso e não aborda possíveis gargalos de comunicação que podem apresentar um melhor agrupamento.

2.5.2 Referentes aos meios de comunicação

O trabalho de [Tsai et al. \(2010\)](#) apresenta um modelo de comunicação híbrido entre SoC's para reduzir a latência e o gargalo de comunicação. O modelo de comunicação híbrido desenvolvido foi testado em duas aplicações de processamento de vídeo e quatro matrizes aleatórias. O modelo de comunicação híbrido apresentou bons resultados para as aplicações utilizadas, pois diminuiu a latência na comunicação.

[Richardson et al. \(2006\)](#) apresentam uma arquitetura de comunicação que possui um barramento chamado dTDMA capaz de alocar dinamicamente espaços para transmissão de dados. Essa arquitetura apresentou um bom desempenho para a transmissão de dados, especialmente longas transmissões, apresentando uma redução de 66% na latência da comunicação e uma redução 20% no consumo de energia se comparada a uma arquitetura NoC.

É importante destacar que nenhum dos trabalhos apresentados na Seção 2.5 considera o período de tempo da comunicação entre as funções da aplicação e entre as funções e a memória. Essa informação é importante, pois permite a análise da distribuição dos gargalos durante o tempo de execução e não somente a comunicação final acumulada entre suas partes.

Na Tabela 1 é apresentada uma comparação entre trabalhos correlatos propostos na literatura e o projeto proposto.

O primeiro parâmetro de comparação considera se a comunicação acumulada da aplica-

ção é utilizada para realizar um agrupamento. É possível observar que a maioria dos trabalhos da literatura e o *framework* desenvolvido consideram essa informação, sendo que no *framework*, essa comunicação é utilizada para a comparação dos resultados obtidos.

O segundo parâmetro comparado considera as informações sobre o meio de comunicação para conseguir um melhor desempenho ao executar uma aplicação. Pode-se notar que a maioria dos trabalhos propostos não consideram o meio de comunicação. É importante destacar, que essa informação é relevante para avaliar o real comportamento do meio de comunicação, por isso, foi considerada no desenvolvimento do *framework*.

Finalmente, o terceiro parâmetro considera o comportamento da comunicação de uma aplicação durante sua execução. É possível perceber que nenhum dos trabalhos correlatos considera essa informação. No *framework* desenvolvido essa comunicação é considerada, pois permite a obtenção de um agrupamento melhor do que os propostos na literatura, pois com base nessa informação é possível analisar o momento em que ocorre um pico de comunicação da aplicação.

Tabela 1 – Comparação entre os trabalhos relacionados e o projeto proposto.

Trabalhos Relacionados	Abordagem	Considera a comunicação acumulada	Considera o meio de comunicação	Considera a comunicação através do tempo
(OSTADZADEH <i>et al.</i> , 2010)	Ferramenta QUAD	Sim	Não	Não
(PHAM-QUOC <i>et al.</i> , 2015)	Modelo híbrido de comunicação	Sim	Sim	Não
(ZHAO; KARYPIS, 2002)	Comparação entre técnicas de agrupamento	Não	Não	Não
(I. Ashraf and V.M. Sima and K.L.M. Bertels, 2015)	Ferramenta MCProf	Sim	Não	Não
(ASHRAF <i>et al.</i> , 2013)	Ferramenta PET	Sim	Não	Não
(TSAI <i>et al.</i> , 2010)	Modelo de comunicação	Não	Sim	Não
(RICHARDSON <i>et al.</i> , 2006)	Modelo de comunicação	Não	Sim	Não
PROJETO PROPOSTO	<i>Framework</i> desenvolvido	Sim	Sim	Sim

2.6 Considerações finais

Neste capítulo foram apresentados os conceitos necessários para a compreensão deste projeto.

Inicialmente, foram descritos os conceitos de agrupamento e meio de comunicação. Em seguida, foi apresentada a linguagem de programação R utilizada na implementação do *framework*. Além disso, foram apresentadas as ferramentas externas MCProf e PET, que foram utilizadas para no desenvolvimento do *framework*.

Finalmente, foram apresentados os trabalhos relacionados a este projeto. Esses trabalhos foram comparados com o *framework* desenvolvido, que é apresentado no próximo capítulo.

FRAMEWORK DESENVOLVIDO

3.1 Considerações iniciais

Este capítulo apresenta um *framework* para a obtenção de um agrupamento de funções mais eficiente para reduzir a demanda pelo uso do meio de comunicação entre as unidades de processamento e por consequência permitir o aumento de desempenho de tais sistemas. Esse *framework* tem como objetivo agrupar funções em conformidade com o comportamento da comunicação de dados entre as mesmas durante a execução da aplicação, considerando para isso sistemas computacionais compostos por mais de uma unidade de processamento com características distintas entre si. O *framework* foi desenvolvido no sistema operacional Ubuntu 15.04 na plataforma Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz 2.30GHz com 6.00GB de RAM utilizando o terminal R na versão 3.1.2. O código fonte do *framework* é apresentado no [Apêndice C](#).

No Capítulo 2 foram apresentadas as ferramentas MCProf e PET, destacando o funcionamento, vantagens e limitações de cada uma. Uma das fragilidades dessas ferramentas, é que sozinhas não apresentam resultados muito eficientes, pois não analisam todo o perfil de comunicação, porque consideram apenas o volume acumulado de dados comunicados durante a execução da aplicação. O *framework* utiliza essas ferramentas como um serviço externo para obter o comportamento da comunicação através do tempo e obter um agrupamento de funções considerando esse comportamento. Para a ferramenta MCProf, foi desenvolvido e incluído na mesma um módulo de extensão que tem como objetivo auxiliar a obtenção das matrizes de comunicação através do tempo, visto que a ferramenta não oferece essa alternativa.

Este capítulo está organizado da seguinte forma: na Seção 3.2 é descrito o *framework* desenvolvido; na Subseção 3.2.1 é apresentado o método de inicialização do *framework* e na Subseção 3.2.2 é descrito como a comunicação entre o *framework* e a ferramenta MCProf é realizada. Na Subseção 3.2.2.1 é apresentado o módulo de extensão para a ferramenta MCProf

ARRUMADO: Na Subseção 3.2.2.1 é apresentado o módulo de extensão para a ferramenta MCProf. Na Subseção 3.2.3 é descrito como as matrizes de tempo são obtidas, na Subseção 3.2.4 é apresentada a comunicação entre o *framework* e a ferramenta PET e na Subseção 3.2.5 é descrito como o agrupamento é analisado no *framework*. Finalmente na Seção 3.3 são apresentadas as considerações finais do capítulo.

3.2 Framework desenvolvido

Na Figura 9 é apresentado o esquema geral do *framework* desenvolvido. As partes destacadas em azul correspondem às informações de entrada e saída do *framework*, as em verde correspondem às ferramentas externas com as quais o *framework* se comunica e as vermelhas correspondem às principais contribuições deste trabalho.

Figura 9 – Esquema geral do *framework* desenvolvido.

O *framework* é inicializado por meio de arquivos de configuração e de comandos fornecidos pelo usuário para controlar a execução das aplicações a serem monitoradas. Em seguida, o *framework* comunica-se com a ferramenta MCProf com o objetivo de obter as matrizes de comunicação, utilizando para isso a janela de tempo disponibilizada pelo usuário em um dos arquivos de configuração. Posteriormente, as matrizes de comunicação são convertidas, pelo *framework*, em matrizes de tempo que são enviadas uma a uma para a ferramenta PET com o propósito de agrupar as mesmas. Esse agrupamento é realizado com o objetivo de obter dois ou mais grupos. Finalmente, a ferramenta PET retorna os grupos gerados ao *framework*, no qual são analisados com o objetivo de retornar ao usuário um único grupo baseando-se na comparação entre os tempos de execução obtidos.

Na Figura 10 é apresentado um algoritmo para o *framework* desenvolvido. Inicialmente, o *framework* chama a MCProf para obter as matrizes de comunicação através do tempo com

o auxílio do módulo de extensão desenvolvido. Em seguida, para cada matriz de comunicação, é realizada uma comparação entre a taxa e o valor máximo/mínimo de transferência, para verificar qual valor será utilizado para dividir cada matriz e assim obter as matrizes de tempo. Posteriormente, é realizado o agrupamento de cada matriz de tempo. Finalmente, após todas as matrizes de tempo serem agrupadas, todos os agrupamentos obtidos são analisados para encontrar o melhor.

Algoritmo 1: Algoritmo do <i>framework</i> proposto	
1.	Executar a ferramenta MCProf para obter as matrizes de comunicação
2.	Para cada matriz de comunicação
3.	Ler matriz de comunicação
4.	Analisar taxa de transferência
5.	Se taxa de transferência > banda máxima
6.	Dividir matriz por banda máxima
7.	Se taxa de transferência < banda mínima
8.	Dividir matriz por banda mínima
9.	Senão
10.	Dividir matriz por taxa de transferência
11.	Acumular o resultado em uma matriz auxiliar
12.	Enviar matriz de tempo para o agrupamento
13.	Enviar matriz auxiliar para o agrupamento
14.	Analisar qual melhor agrupamento encontrado

Figura 10 – Algoritmo do *framework* desenvolvido.

3.2.1 Inicialização do *framework*

O usuário é o responsável pelo início da execução do *framework*. Esse deve criar dois arquivos de configuração no formato *.txt* que são armazenados no próprio diretório do *framework* e são utilizados como parâmetros de entrada. O primeiro arquivo de configuração deve ser nomeado como *Input.txt* enquanto o segundo arquivo de configuração deve ser nomeado como *Paths.txt*.

O primeiro arquivo de configuração é formado por parâmetros relacionados à aplicação a ser agrupada e sobre as características do meio de comunicação. Os parâmetros relacionados à aplicação são:

- Janela de tempo: este parâmetro indica a janela de tempo em milissegundos em que cada matriz de comunicação é obtida através da ferramenta MCProf;
- Número de funções: este parâmetro indica o número de funções existentes na aplicação e é um dos utilizados pela ferramenta PET para a realização do agrupamento das matrizes de tempo;
- Número de grupos: este parâmetro indica o número de grupos que o usuário deseja usar para o agrupamento e é um dos utilizados pela ferramenta PET para a realização do agrupamento das matrizes de tempo.

Os parâmetros do meio de comunicação são:

- Largura de banda máxima: indica qual a taxa máxima de transferência permitida pelo meio de comunicação;
- Largura de banda mínima: indica qual a taxa mínima de transferência desejável para o meio de comunicação;
- Função de transferência: determina a taxa de transferência do meio de comunicação através do tempo.

É importante ressaltar que os parâmetros devem estar de acordo com a seguinte ordem: janela de tempo, número de funções, número de grupos desejados, largura de banda máxima do meio de comunicação, largura de banda mínima do meio de comunicação e uma função que representa a taxa de transferência do meio de comunicação. Essa ordenação foi definida, pois, conforme decisão de projeto, o *framework* lê e armazena cada parâmetro em sua respectiva variável global, que são utilizadas no decorrer da execução do mesmo.

Na Figura 11 é apresentado um exemplo de preenchimento do primeiro arquivo de configuração, no qual é considerada uma janela de tempo de 100 milissegundos, 42 funções, 4 grupos, largura de banda máxima de 250 MB/s = 250000000 B/s, largura de banda mínima de 50 MB/s = 50000000 B/s e uma função de transferência $f(t) = 250\log(t)$ MB/s = 250000000log(t) B/s na qual x corresponde ao tempo t em segundos.

```
100
42
4
250000000
50000000
250000000*log(x)
```

Figura 11 – Exemplo de preenchimento do primeiro arquivo de configuração.

O segundo arquivo de configuração deve conter três caminhos absolutos e um comando necessário para a execução da aplicação. Os três caminhos, na ordem em que devem ser fornecidos, são: caminho absoluto do diretório em que está armazenado o executável da MCPProf, caminho absoluto do diretório em que está armazenado o executável da ferramenta PET e caminho absoluto do diretório atual do *framework*.

Após a criação dos arquivos de configuração pelo usuário, é preciso que o mesmo utilize o comando `source('/caminho até o framework/framework.R')` no terminal R para a execução do *framework*, pois esse é o comando utilizado para executar um *script* produzido na linguagem R.

3.2.2 Execução da ferramenta MCProf

Após sua inicialização, o *framework* executa uma chamada de sistema para a ferramenta MCProf com o propósito de obter as matrizes de comunicação da aplicação do usuário. Essa chamada de sistema é formada pelo caminho absoluto do executável da MCProf, janela de tempo e o comando necessário para a execução da aplicação. Além desses, outros parâmetros específicos da MCProf também são considerados. Neste projeto, os parâmetros específicos considerados são: função de rastreamento de memória, módulo a ser utilizado e janela de tempo.

A MCProf retorna para o *framework* N matrizes de comunicação. A matriz de comunicação representa a quantidade de *bytes* trocados entre as funções da aplicação, sendo que as mesmas linhas e colunas da matriz representam funções iguais. Cada função presente nas linhas da matriz troca uma certa quantidade de *bytes* com cada função presente nas colunas. Essas matrizes são preenchidas pela MCProf por meio de comandos internos que executam o *framework* PIN. Na Figura 12 é apresentado um exemplo de uma matriz de comunicação com três funções sendo que a função F1 envia 11 *bytes* para a função F2 e 8 *bytes* para a função F3 enquanto a função F2 envia 16 *bytes* para a função F3. Embora na figura a comunicação é representada por uma matriz triangular superior, isso não representa que todas as matrizes de comunicação são dessa forma.

	F1	F2	F3
F1	0	11	8
F2	0	0	16
F3	0	0	0

Figura 12 – Exemplo de matriz de comunicação de uma aplicação com 3 funções.

Para a obtenção das matrizes de comunicação, foi desenvolvido um módulo de extensão para a MCProf com o objetivo de obter o volume da comunicação por janela de tempo, sendo que para cada janela haverá uma matriz associada. Na Figura 13 é apresentado um exemplo de retorno de janelas para a MCProf na versão original e na versão modificada. Neste caso, para a versão modificada há quatro janelas de tempo de 1 segundo cada, considerando uma aplicação que demore quatro segundos para ser executada. A matriz representada por (a) corresponde a matriz de comunicação original retornada pela MCProf, pois em sua versão original a ferramenta retorna somente uma única matriz acumulada no final da execução da aplicação contendo o volume total de comunicação entre as funções. Já (b) representa as quatro matrizes de comunicação da versão modificada.

Com o uso de janelas de tempo é possível obter mais detalhes do comportamento a respeito da comunicação entre as funções da aplicação. Na Figura 13 é possível perceber que em (a) a matriz de comunicação apresenta dois picos de comunicação, que ocorrem entre as funções F2x F3 (valor 16) e F1x F2 (valor 11). Enquanto que em (b), utilizando a janela de tempo, pode-se obter com maior precisão onde e quando a comunicação ocorre e com isso permitir que haja uma

Figura 13 – Exemplo de utilização da janela de tempo.

distribuição das respectivas funções de maneira mais eficiente em plataformas de processamento paralelo.

3.2.2.1 Módulo de extensão

Como descrito na Seção 2.4.2, a MCProf apresenta três módulos dentro do bloco *Data Collection Engines* (DCE), sendo que cada módulo é responsável pela execução de uma tarefa diferente. Neste projeto foi desenvolvido um quarto módulo de extensão que é utilizado para obter as matrizes de comunicação através do tempo. Esse módulo apresenta um funcionamento similar ao módulo de número 2, que é o responsável por imprimir a matriz de comunicação acumulada. No entanto, o módulo de extensão não apresenta a porcentagem de execução de cada função na aplicação e a matriz retornada no módulo apresenta todas as funções da aplicação, inclusive aquelas que não se comunicam em um determinado instante de tempo. Quando o *framework* executa uma chamada de sistema para a MCProf, é selecionado o módulo de extensão e a janela de tempo é atribuída para uma variável que é utilizada para a obtenção das matrizes de comunicação ao longo do tempo.

O arquivo principal da MCProf foi alterado com o objetivo de utilizar o módulo de extensão, que é utilizado da seguinte forma: sempre que uma rotina é retornada da tabela de símbolos usada pela MCProf, é feita uma verificação entre a variável e o tempo de retorno da rotina. Caso o valor de retorno seja menor que a janela de tempo, nenhuma matriz de comunicação é retornada. Caso o valor seja maior ou igual a janela de tempo, é retornada a matriz de comunicação para aquele instante de tempo, representado pela janela de tempo. Em seguida a matriz de comunicação da MCProf é zerada de modo a começar a acumular novamente a comunicação. O trecho de código da MCProf que foi alterado para utilizar o módulo de extensão está apresentado no [Apêndice D](#).

3.2.3 Obtenção das matrizes de tempo

As matrizes de comunicação obtidas anteriormente são utilizadas pelo *framework* para o cálculo e criação das matrizes de tempo, que correspondem a quantidade de tempo necessário para as funções trocarem a quantidade de *bytes* que compõem a matriz de comunicação, considerando para isso uma determinada largura de banda disponível no instante da comunicação. Essas novas matrizes de tempo são armazenadas em variáveis globais definidas no *framework*.

Para o cálculo do tempo, utiliza-se as larguras de banda e a função de transferência definidas pelo usuário no primeiro arquivo de configuração. As matrizes de comunicação são divididas pelo valor resultante do cálculo da taxa de transferência naquele instante de tempo. Para determinar o valor taxa de transferência em um determinado instante é utilizada a função de transferência, de modo que o valor de x da função é substituído por um valor que varia de 1 até N , sendo que N representa o número total de matrizes de comunicação obtidas dinamicamente pela MCPProf.

Antes da utilização do valor obtido na taxa de transferência, é realizada uma checagem entre o valor e as larguras de banda definidas pelo usuário. Essa checagem tem como objetivo evitar que a matriz de comunicação seja dividida por um valor acima da largura de banda máxima ou por um valor menor que a largura de banda mínima. Assim, caso o valor da transferência ultrapasse os limites máximos e mínimos, o valor utilizado será o respectivo limite.

Enquanto as matrizes de comunicação são convertidas uma a uma em matrizes de tempo, o *framework* cria automaticamente uma matriz auxiliar para armazenar a soma de todas as matrizes de comunicação obtidas pela MCPProf. Essa soma é realizada antes da divisão das matrizes de comunicação pelas suas respectivas taxas de transferência. O *framework* também cria automaticamente uma matriz acumulada de tempo formada pela soma das matrizes de tempo. O propósito dessa matriz é de acelerar a comparação dos agrupamentos, pois a comparação utilizando uma matriz única é mais rápida. É importante armazenar a matriz de comunicação acumulada para que após o término de todos os agrupamentos seja possível comparar o agrupamento acumulado existente na ferramenta PET com o agrupamento encontrado através do tempo, desenvolvido neste projeto.

3.2.4 Execução da ferramenta PET

Após a conversão das matrizes de comunicação em matrizes de tempo, o *framework* realiza uma chamada de sistema para a ferramenta PET, passando uma matriz de tempo por vez, com o propósito de agrupar cada uma. Essa comunicação é representada na Figura 9 pela seta denominada Matriz de tempo i , no qual i varia de modo incremental de 1 até N . Conforme apresentado no Capítulo 2, a ferramenta PET apresenta um formato de arquivo de entrada específico, assim o *framework* utiliza o número de funções e o número de grupos informados pelo usuário no primeiro arquivo de configuração para gerar a entrada de cada matriz utilizada

pela ferramenta PET.

Com o término do agrupamento da matriz de tempo i , a ferramenta PET retorna automaticamente para o *framework* um grupo para cada matriz, representado na Figura 9 pela seta denominada Agrupamento i . Na Figura 14 é apresentado um exemplo de três grupos (*cluster*) retornados pela ferramenta PET, no qual cada grupo apresenta um determinado número de funções. A sigla BP significa o custo de balanceamento dos grupos, CC representa o custo do agrupamento e Cost corresponde à soma dessas informações ($\text{Cost} = \text{BP} + \text{CC}$).

Após o término do agrupamento de todas as matrizes de tempo, o *framework* envia para a ferramenta PET a matriz de comunicação acumulada para que a ferramenta encontre um agrupamento que utiliza apenas o volume total de comunicação, que é usado neste projeto para comparar os resultados entre as diferentes abordagens.

```
Starting Simulation

=====
Application Generation (Save/Restore Mode)
=====
Restore Mode (Application will be restored from file)
Restoring Functions... Done !
Restoring Functions... Done !
=====
Evolutionary Search Summary
=====
Time Elapsed = (HH:MM:SS) 0:00:00

Details of Best Partition found by Evolutionary Search...
Cluster 0 has 2 functions ( 0 3 )
Cluster 1 has 2 functions ( 1 4 )
Cluster 2 has 1 functions ( 2 )
ES Cost = 291.666656
ES BP   = 26.666668
ES CC   = 265.000000

End of Simulation
```

Figura 14 – Saída obtida pela ferramenta PET, no qual é observado um agrupamento de 5 funções em 3 grupos. Fonte: Ferramenta PET

3.2.5 Análise dos agrupamentos

O *framework* analisa automaticamente todas os agrupamentos obtidos da ferramenta PET. Por decisão de projeto, o agrupamento obtido pela matriz de comunicação acumulada será testado inicialmente. O *framework* armazena o agrupamento em uma estrutura de dados na forma de lista encadeada, e o aplica na matriz de tempo acumulada com o propósito de obter o tempo

de execução total da aplicação. Em seguida, esse processo é repetido pelo *framework* para cada agrupamento encontrado para as N matrizes de tempo.

Na Figura 15 é apresentada um exemplo de como a comparação é realizada. O grupo A é obtido a partir da matriz de comunicação acumulada e possui um tempo de execução Z. Esse grupo é aplicado na matriz de tempo acumulada obtendo-se assim o tempo de execução Y. O grupo B é obtido através do melhor grupo encontrado pelas matrizes de tempo. Esse grupo é aplicado na matriz acumulada de tempo para obter o tempo X. Os valores temporais obtidos após a comparação são $X < Y < Z$, ou seja, o grupo B teve um valor temporal melhor, pois considera a comunicação distribuída ao longo do tempo. Esse resultado está de acordo com o esperado, pois o tempo obtido através das matrizes de tempo foi menor que encontrado utilizando o agrupamento da literatura.

Figura 15 – Exemplo da comparação realizada neste projeto

Após todos os agrupamentos serem analisados, o *framework* compara o valor de tempo obtido pelo agrupamento da matriz de comunicação acumulada com o menor tempo obtido utilizando os agrupamentos encontrados com as matrizes de tempo. Após tal comparação, o *framework* retorna como saída para o usuário os tempos obtidos, indicando qual o melhor agrupamento obtido entre os agrupamentos propostos na literatura, conforme Capítulo 2, e o desenvolvido neste projeto.

3.3 Considerações finais

Neste capítulo foi apresentado o *framework* desenvolvido. Esse *framework* integra as ferramentas MCProf e PET com o objetivo de obter um agrupamento de funções mais eficiente

para reduzir a demanda pelo uso do meio de comunicação entre as unidades de processamento de sistemas de computação paralelos e por consequência permitir o aumento de desempenho de tais sistemas. Além disso, foi apresentado o módulo de extensão desenvolvido para a ferramenta MCProf para obter matrizes de comunicação através do tempo.

RESULTADOS E ANÁLISES

4.1 Considerações iniciais

Neste capítulo é apresentada a avaliação realizada para validar o *framework* desenvolvido. Para a avaliação foram realizados testes com aplicações que manipulam imagens, realizam cálculos matriciais e simulam partículas em movimento. As aplicações analisadas possuem execução sequencial e não paralela, o que significa que duas ou mais funções da aplicação não acessam o meio de comunicação no mesmo intervalo de tempo. Considera-se que as aplicações serão executadas em uma plataforma heterogênea, na qual cada componente apresenta características distintas entre si, logo a aplicação será executada de forma mais eficiente se cada grupo for alocado para um acelerador específico. Cabe ressaltar que, se a plataforma for homogênea, não é vantajoso agrupar as aplicações que executem de maneira sequencial.

Inicialmente, foi realizada uma análise considerando uma taxa de transferência constante para o meio de comunicação. Caso isso ocorra, o resultado da análise das matrizes de comunicação através do tempo será o mesmo se for utilizada a matriz de comunicação acumulada. Na Figura 16, k é um valor constante que representa a taxa de transferência constante do meio de comunicação e M_1 , M_2 e M_N representam matrizes de comunicação. De acordo com as propriedades matriciais, se um valor constante for utilizado para multiplicar/dividir várias matrizes, o resultado é o mesmo se a soma dessas matrizes for multiplicada/dividida por esse valor constante. Assim, ao dividir várias matrizes de tempo por uma taxa de transferência constante, teremos o mesmo resultado caso fosse utilizada na divisão a matriz de comunicação acumulada proposta na literatura.

$$\frac{1}{k}M_1 + \frac{1}{k}M_2 + \dots + \frac{1}{k}M_N = \frac{1}{k}[M_1 + M_2 + \dots + M_N]$$

Figura 16 – Exemplo de divisão de matrizes por um valor constante.

Também foi considerado que todos os grupos possuem comunicação entre si e um grupo pode enviar ou receber dados para qualquer outro grupo. Os grupos se comunicam dessa forma, pois as técnicas de agrupamento não consideram o meio de comunicação existente, somente o volume total de dados da comunicação.

Nos testes foi considerada uma largura de banda máxima de 250 MB/s e uma largura de banda mínima de 50 MB/s e para emular diferentes taxas de transferências foram utilizadas três funções definidas em (4.1), (4.2) e (4.3). Os valores máximo e mínimo da largura de banda foram escolhidos para simular um meio de comunicação *PCI Express*, explicado no Capítulo 2, sendo que neste projeto foram considerados cinco canais de comunicação que transferem dados a 50 MB/s por cada canal.

As funções definidas para os testes simulam três tipos de transferência de dados: 1) uma taxa de transferência periódica (4.1), que representa uma comunicação de baixa taxa em seu início, mas que cresce e em seguida decresce com o passar do tempo, 2) uma taxa crescente (4.2), que simula uma baixa taxa de transferência inicial que aumenta com o tempo e 3) uma taxa decrescente (4.3) que simula uma alta taxa de transferência inicial que decai com o tempo. Em todas as funções, t corresponde ao tempo que o usuário define para a janela de tempo.

É importante destacar que a função (4.1) é uma senóide de meio ciclo devido ao módulo utilizado e seu período é aproximadamente $T = 6$ s. A função (4.2) não está definida para 0 devido ao fato de não existir $\log(0)$, nesse caso, o *framework* considera o valor da largura de banda mínima, definido como 50 MB/s. Já a função (4.3) não está definida para 0 e 1, devido ao fato de não existir $\log(0)$ e de que $\log(1) = 0$, impossibilitando a divisão por 0. Nesse caso, o *framework* considera o valor da largura de banda máxima, definido como 250 MB/s.

$$f(t) = 250|\sin(0,5t)|\text{MB/s, sendo } t \geq 0, \text{ com } t \text{ em radianos} \quad (4.1)$$

$$f(t) = 250\log(t)\text{MB/s, sendo } \{t \in \mathbb{R} \mid t > 0\} \quad (4.2)$$

$$f(t) = 100/\log(t)\text{MB/s, sendo } \{t \in \mathbb{R} \mid t > 1\} \quad (4.3)$$

No gráfico da Figura 17 é apresentado como essas funções se comportam ao longo do tempo. O gráfico foi desenvolvido com o auxílio da ferramenta R, utilizando 2000 pontos igualmente espaçados entre 0 e 20. É importante ressaltar que as função crescente e a função decrescente começam fora do limite máximo do gráfico, pois não estão definidas para $\log(0)$.

As diferentes funções de transferência foram escolhidas, pois no mundo real as transferências de dados dificilmente se comportam de maneira linear em um ambiente com recursos compartilhados.

Figura 17 – Gráfico que representa o comportamento das funções de transferência através do tempo.

Para cada aplicação, foram realizadas 30 execuções, considerando as funções de transferência apresentadas anteriormente, totalizando 180 resultados para cada aplicação. Além disso, para cada aplicação foi analisada tanto a comunicação acumulada quanto a comunicação distribuída através do tempo. Para a análise dos resultados, foi calculado o valor médio do tempo obtido para cada função de transferência. Em todos os testes, o módulo 2 da ferramenta MCProf foi utilizado para a obtenção da matriz de comunicação acumulada da aplicação e do número total de funções. Para cada matriz de comunicação criada foi elaborado um gráfico de superfície para realçar toda a comunicação. As tabelas com todos os resultados obtidos nos testes estão apresentadas no [Apêndice A](#).

Para validar os resultados obtidos, foram realizados os testes de hipótese **Teste t de Student** ([HAYNES, 2013](#)) e **Teste de Wilcoxon** ([WILCOXON, 1945](#)). Esses testes são utilizados para comparar duas amostras e verificar se elas são iguais ou diferentes entre si. Através deles, é possível obter um intervalo de mais de 95% de confiança de que as amostras são iguais ou diferentes entre si, podendo assim rejeitar ou não a hipótese nula do teste.

Para a realização dos testes, foi utilizada a linguagem de programação R que considera a hipótese nula H_0 e hipótese alternativa H_1 , sendo:

- H_0 = As duas amostras são iguais;
- H_1 = As duas amostras são diferentes.

Para decidir se a hipótese nula deve ser aceita ou rejeita, é preciso verificar o valor de p , que corresponde ao menor valor de significância que deve ser considerado para aceitar/rejeitar a hipótese nula. Caso esse valor seja menor que 5%, pode-se rejeitar a hipótese nula e aceitar a hipótese alternativa, que considera as amostras comparadas como sendo diferentes entre si.

Inicialmente, foram comparados os resultados obtidos através da comunicação acumulada. Para isso, os resultados de cada aplicação foram comparados da seguinte forma: resultados obtidos utilizando a função de transferência (4.1) X resultados obtidos utilizando a função de transferência (4.2), resultados obtidos utilizando a função de transferência (4.1) X resultados obtidos utilizando a função de transferência (4.3) e resultados obtidos utilizando a função de transferência (4.2) X resultados obtidos utilizando a função de transferência (4.3). Essa comparação foi realizada para cada aplicação utilizada para validar o projeto proposto. Posteriormente, também foram comparados os resultados obtidos com a comunicação distribuída através do tempo. Finalmente, para cada aplicação foram comparados os resultados acumulados e distribuídos obtidos através de cada função de transferência.

No [Apêndice B](#) são apresentadas as tabelas com os valores de p obtidos na comparação. Através das tabelas, é possível verificar que todos os valores são menores que 5%, fazendo com que a hipótese alternativa seja aceita, ou seja, as amostras comparadas são diferentes entre si.

Este capítulo apresenta a análise de cinco aplicações sequenciais escolhidas por apresentarem elevada comunicação de dados entre suas funções. Além disso, apresenta-se a emulação de aplicações paralelas com matrizes de comunicação geradas de maneira aleatória para explorar o potencial do *framework* proposto em situações de maior concorrência pelo meio de comunicação. Na Seção 4.2 é analisada a aplicação KLT. Na Seção 4.3 é apresentada a análise da aplicação Sparse 1.3a. Na Seção 4.4 são apresentadas análises de três aplicações do *benchmark SPLASH-2*. Na Seção 4.5 são apresentadas as análises das aplicações paralelas emuladas. Finalmente na Seção 4.6 são apresentadas as considerações finais do capítulo.

4.2 Aplicação KLT

A primeira aplicação escolhida para avaliar o *framework* foi a KLT ([LUCAS; KANADE et al., 1981](#)), que corresponde a uma técnica que utiliza a intensidade do gradiente espacial de uma imagem para encontrar pontos de similaridades entre duas imagens. Essa aplicação foi escolhida porque utiliza várias funções matemáticas para encontrar os pontos de semelhança, resultando em uma elevada comunicação entre suas funções.

Na Figura 18 é apresentado o gráfico gerado com a matriz de comunicação da aplicação. As regiões destacadas apresentam uma elevada comunicação. A aplicação da figura apresenta 42 funções e 3 picos de comunicação, representados pelas cores vermelho, amarelo e verde.

O *framework* foi utilizado para obter o agrupamento da aplicação considerando a sua comunicação através do tempo. Utilizando uma janela de tempo de 100 milissegundos foram obtidas 7 matrizes de comunicação. A aplicação foi agrupada em 3, 4 e 5 grupos para verificar qual a quantidade de grupos a ser utilizado. Posteriormente, a aplicação foi executada 30 vezes no *framework* utilizando cada função de transferência, totalizando 180 execuções para cada grupo.

Figura 18 – Comunicação acumulada no final da aplicação.

Na Tabela 2 é apresentado o tempo médio e o ganho obtidos ao utilizar a comunicação acumulada dividida pela comunicação através do tempo. A aplicação foi agrupada em 4 grupos com o objetivo de verificar se as funções que apresentam elevada comunicação seriam agrupadas no mesmo grupo. Por decisões de projeto, a quantidade de grupos utilizados nos testes corresponde ao número de picos mais um.

Tabela 2 – Média em segundos dos tempos encontrados no agrupamento da aplicação KLT utilizando 4 grupos.

	Função Periódica		Função Crescente		Função Decrescente	
	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída
Média	1,401	1,082	0,642	0,542	2,794	2,182
Desvio Padrão	0,356	0,390	0,202	0,179	0,897	0,870
Ganho	1,294		1,184		1,281	

Para avaliar os valores obtidos, foi elaborado um gráfico, no qual é possível analisar os picos de comunicação e verificar o comportamento da comunicação nas matrizes obtidas através da janela de tempo. Na Figura 19 é apresentado o gráfico da distribuição da comunicação dos picos da aplicação KLT através do tempo.

Na Figura 19, o primeiro pico de comunicação está no meio da aplicação, enquanto o segundo pico ocorre no fim da mesma. Já o terceiro pico de comunicação é repetido duas vezes após a metade da execução da aplicação. Nota-se que como ocorreu uma intensa comunicação após a metade da execução da aplicação e se a largura de banda neste período for alta, obtém-se um tempo de comunicação menor.

Por meio da análise dos valores médios de comunicação da aplicação na Tabela 2, observa-se que o maior ganho de velocidade ocorreu quando a função decrescente é utilizada,

Figura 19 – Distribuição da comunicação dos picos da aplicação KLT.

embora essa apresente o pior tempo de comunicação. O melhor tempo de comunicação ocorreu quando foi utilizada uma função crescente para simular a largura de banda. Na Figura 20 é apresentada uma comparação dos tempos médios de comunicação obtidos, na qual as colunas em azul apresentam o tempo obtido no agrupamento encontrado utilizando a matriz acumulada de comunicação aplicado nas matrizes de tempo e as colunas em vermelho apresentam o tempo obtido com o agrupamento encontrado utilizando as matrizes distribuídas de tempo.

Figura 20 – Comparação de tempos entre cada função.

Pelos resultados obtidos após os testes, conclui-se que ao realizar um agrupamento analisando a comunicação via janelas de tempo e o meio de comunicação, obtêm-se um resultado melhor, pois há um ganho de 1,184x em relação à comunicação acumulada, bem como um menor tempo de comunicação. Na Figura 21 é apresentado o agrupamento ótimo encontrado para a aplicação KLT.

```

Details of Best Partition found by Evolutionary Search...
Cluster 0 has 18 functions ( 0 5 7 8 9 12 14 19 20 21 27 29 30 34 38 39 40 41 )
Cluster 1 has 7 functions ( 1 10 11 18 24 35 36 )
Cluster 2 has 6 functions ( 2 15 23 25 28 31 )
Cluster 3 has 11 functions ( 3 4 6 13 16 17 22 26 32 33 37 )

```

Figura 21 – Agrupamento ótimo encontrado para a aplicação KLT.

4.3 Aplicação Sparse 1.3a

A segunda aplicação utilizada para a avaliação do *framework* foi a Sparse (KUNDERT; SANGIOVANNI-VINCENTELLI, 1988), que é utilizada para resolver sistemas lineares esparsos. Essa aplicação foi escolhida porque permite a realização de diversos cálculos matemáticos para resolver sistemas lineares esparsos, o que resulta em uma elevada troca de dados.

Na Figura 22 é apresentado o gráfico da matriz de comunicação da aplicação, que possui 170 funções. Na figura, existem 2 picos de comunicação, representados pelas cores vermelho e amarelo. Dessa forma, optou-se por agrupar a aplicação em 3 grupos, como forma de verificar se as funções que possuem elevada comunicação estariam no mesmo grupo.

Figura 22 – Comunicação acumulada no final da aplicação.

Utilizando uma janela de tempo de 100 milissegundos foram obtidas 5 matrizes de comunicação. Posteriormente, a aplicação foi executada 30 vezes no *framework* utilizando cada função de transferência totalizando 180 execuções. Os valores médios de tempo obtidos analisando o agrupamento encontrado utilizando a matriz de comunicação acumulada e o melhor agrupamento encontrado através das matrizes de tempo são apresentados na Tabela 3, bem como o ganho obtido ao utilizar a comunicação acumulada dividida pela comunicação através do tempo.

Com as matrizes de comunicação, foi gerado um gráfico para analisar como a comu-

Tabela 3 – Média em segundos dos tempos encontrados no agrupamento da aplicação Sparse.

	Função Periódica		Função Crescente		Função Decrescente	
	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída
Média	0,737	0,397	0,123	0,073	0,556	0,365
Desvio Padrão	0,230	0,136	0,024	0,011	0,101	0,066
Ganho	1,858		1,695		1,523	

nicação está distribuída durante a execução da aplicação. Na Figura 23 é apresentada essa comunicação, no qual os picos de comunicação ocorrem no começo da execução da aplicação, evidenciando uma elevada troca de dados entre duas funções.

Figura 23 – Distribuição dos picos de comunicação da aplicação Sparse.

Analisando os tempos obtidos na Tabela 3, o maior ganho de velocidade ocorreu quando foi utilizada a função periódica devido à alta disponibilidade de taxa de transferência nesse instante de comunicação. Na Figura 24 é apresentada uma comparação entre os tempos médios de comunicação obtidos, na qual as colunas em azul apresentam o tempo obtido utilizando a matriz acumulada de comunicação aplicado nas matrizes de tempo e as em vermelho apresentam o tempo obtido quando são utilizadas as matrizes de tempo.

Por meio dos resultados, pode-se concluir que ao realizar um agrupamento analisando a comunicação através do tempo e o meio de comunicação, obtêm-se um resultado melhor em relação aos métodos empregados na literatura, pois é obtido um ganho de 1,801x. Entretanto, obtemos um menor tempo de comunicação utilizando a função crescente. Logo, esse tipo de transferência é melhor para comunicação da aplicação. Na Figura 25 é apresentado o agrupamento ótimo encontrado para a aplicação Sparse.

Figura 26 – Comunicação acumulada no final da aplicação.

encontrado através das matrizes de tempo são apresentados na Tabela 4, bem como o ganho obtido ao utilizar a comunicação através da janela de tempo para realizar o agrupamento.

Tabela 4 – Média em segundos dos tempos encontrados no agrupamento da aplicação Barnes.

	Função Periódica		Função Crescente		Função Decrescente	
	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída
Média	0,0070	0,0030	0,0032	0,0012	0,0126	0,0048
Desvio Padrão	0,0014	0,0024	0,0015	0,0010	0,0066	0,0046
Ganho	2,295		2,587		2,621	

Com as matrizes obtidas anteriormente, foi gerado um gráfico no qual é possível analisar como a comunicação está distribuída durante a execução da aplicação. Na Figura 27 é apresentada essa comunicação. Pode-se observar que os picos de comunicação ocorrem no final da aplicação, o que evidencia uma elevada troca de dados entre duas funções.

Analisando os tempos obtidos na Tabela 4, o maior ganho de velocidade ocorreu quando foi utilizada a função crescente devido à alta disponibilidade de taxa de transferência no instante final da comunicação. Na Figura 28 é apresentada uma comparação entre os tempos médios de comunicação obtidos, na qual as colunas em azul apresentam o tempo obtido utilizando a matriz acumulada de comunicação aplicado nas matrizes de tempo e as colunas em vermelho apresentam o tempo obtido utilizando as matrizes de tempo. Através da figura, nota-se um ganho de 2,587x, bem como um menor tempo de comunicação.

Na Figura 29 é apresentado o agrupamento ótimo encontrado para a aplicação Barnes.

Figura 27 – Distribuição dos picos de comunicação da aplicação Barnes.

Figura 28 – Comparação de tempos entre cada função.

4.4.2 Aplicação FMM

A segunda aplicação do *benchmark* utilizada foi a FMM (GREENGARD, 1988), que simula um sistema de corpos em duas dimensões através do método *Fast Multiple Method*.

Na Figura 30 é apresentado o gráfico gerado com a matriz de comunicação da aplicação, que possui 470 funções. No gráfico existem 3 picos de comunicação representados pelas cores vermelho, amarelo e verde. Dessa forma, a aplicação foi agrupada em 4 grupos, para verificar se as funções que possuem elevada comunicação estariam no mesmo grupo. Utilizando uma janela de tempo de 100 milissegundos foram obtidas 4 matrizes de comunicação.

Posteriormente, a aplicação foi executada 30 vezes no *framework* utilizando cada função de transferência totalizando 180 execuções. O valor médio dos tempos obtidos analisando o agrupamento encontrado através da matriz de comunicação acumulada e o melhor agrupamento

Figura 31 – Distribuição dos picos de comunicação da aplicação FMM.

comparação entre os tempos médios de comunicação obtidos, na qual as colunas em azul apresentam o tempo obtido utilizando a matriz acumulada de comunicação aplicado nas matrizes de tempo e as colunas em vermelho apresentam o tempo obtido utilizando as matrizes de tempo.

Figura 32 – Comparação de tempos entre cada função.

Pelos resultados obtidos após os testes, conclui-se que ao realizar um agrupamento analisando a comunicação através do tempo e o meio de comunicação, obtêm-se um resultado melhor para o agrupamento, com um ganho de 2,621x. Entretanto, obtemos um menor tempo de comunicação utilizando a função crescente. Logo, esse tipo de transferência é melhor para comunicação da aplicação. Na Figura 33 é apresentado o agrupamento ótimo encontrado para a aplicação FMM.

Details of Best Partition found by Evolutionary Search...																																							
Cluster 0 has 112 functions (0 5 10 15 16 18 19 23 25 26 29 32 39 47 48 50 51 55 61 62 67 71 72 75 78 79 85 91 92 95 96 107																																							
109	116	117	120	121	123	126	129	133	139	144	145	147	155	158	159	164	165	167	168	171	176	177	182	183	188	196	199	204	207	208									
209	211	220	223	224	225	231	232	234	236	240	266	267	271	272	273	274	277	281	285	286	287	303	304	305	307	309	312	322	337	339									
341	343	344	351	354	359	360	367	370	375	389	393	395	396	402	403	404	406	)																					
Cluster 1 has 106 functions (1 4 6 8 13 14 24 31 43 44 49 53 58 63 65 77 81 84 87 88 89 90 98 104 105 110 111 118 128 132 138																																							
140	141	148	149	150	151	152	153	156	157	160	161	162	163	169	179	180	185	187	189	190	191	200	202	203	205	212	213	214	222	226									
237	239	245	246	250	251	255	256	260	261	280	291	293	296	299	301	302	313	314	316	317	321	327	328	329	333	334	340	353	357	362									
364	365	368	369	372	379	383	390	392	398	399	400	401	)																										
Cluster 2 has 87 functions (2 9 12 17 21 28 33 36 41 42 45 52 59 66 69 70 74 82 83 86 93 94 97 101 103 108 113 114 115 119 122																																							
124	130	131	143	146	170	172	181	184	195	197	198	221	227	228	229	230	235	241	243	244	247	249	252	254	257	259	262	263	265	270									
278	288	292	297	306	310	311	323	324	326	330	338	346	347	348	349	350	352	361	366	373	380	381	384	386	)														
Cluster 3 has 102 functions (3 7 11 20 22 27 30 34 35 37 38 40 46 54 56 57 60 64 68 73 76 80 99 100 102 106 112 125 127 134																																							
135	136	137	142	154	166	173	174	175	178	186	192	193	194	201	206	210	215	216	217	218	219	233	238	242	248	253	258	264	268	269									
275	276	279	282	283	284	289	290	294	295	298	300	308	315	318	319	320	325	331	332	335	336	342	345	355	356	358	363	371	374	376									
377	378	382	385	387	388	391	394	397	405	)																													

Figura 33 – Agrupamento ótimo encontrado para a aplicação FMM.

4.4.3 Aplicação Ocean

A última aplicação utilizada para avaliar o *framework* foi a *Ocean*, que é utilizada para simular o movimento dos oceanos através de partículas em 4D (WOO *et al.*, 1995).

Na Figura 34 é apresentado o gráfico gerado com a matriz de comunicação da aplicação, que possui 387 funções. É possível observar que existem 3 picos de comunicação, representados pelas cores vermelho, amarelo e verde. Dessa forma, a aplicação foi agrupada em 4 grupos, como forma de verificar se as funções que possuem elevada comunicação estariam no mesmo grupo. Utilizando uma janela de tempo de 100 milissegundos foram obtidas 5 matrizes de comunicação.

Figura 34 – Comunicação acumulada no final da aplicação.

Posteriormente, a aplicação foi executada 30 vezes no *framework* utilizando cada função de transferência totalizando 180 execuções. O valor médio de tempo obtido analisando o agrupamento encontrado utilizando a matriz de comunicação acumulada e o melhor agrupamento encontrado através das matrizes de tempo são apresentados na Tabela 6. No final da tabela, são apresentados os valores médios obtidos e o ganho obtido ao utilizar a comunicação acumulada dividido pela comunicação através do tempo.

A partir das matrizes, foi gerado um gráfico para analisar como a comunicação está

Tabela 6 – Média em segundos dos tempos encontrados no agrupamento da aplicação *Ocean*.

	Função Periódica		Função Crescente		Função Decrescente	
	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída
Média	0,0022	0,0017	0,0008	0,0006	0,0037	0,0028
Desvio Padrão	0,0003	0,0003	0,0001	0,0001	0,0004	0,0005
Ganho	1,265		1,265		1,312	

distribuída durante a execução da aplicação. Na Figura 35 é apresentado o gráfico da distribuição da comunicação, no qual os picos de comunicação ocorrem no fim da execução da aplicação, devido a elevada troca de dados entre duas funções.

Figura 35 – Distribuição dos picos de comunicação da aplicação *Ocean*.

Com base nos tempos obtidos na Tabela 6, pode-se verificar que ocorreu um ganho de velocidade quando foi utilizada a função crescente devido a alta disponibilidade de taxa de transferência nesse instante de comunicação. Na Figura 37 é apresentada uma comparação entre os tempos médios de comunicação obtidos, na qual as colunas em azul apresentam o tempo obtido utilizando a matriz acumulada de comunicação aplicado nas matrizes de tempo e as colunas em vermelho apresentam o tempo obtido utilizando as matrizes de tempo. Na Figura 36 é apresentado o agrupamento ótimo encontrado para a aplicação *Ocean*.

```

Details of Best Partition found by Evolutionary Search...
Cluster 0 has 86 functions ( 0 6 18 19 21 22 24 27 33 37 40 41 42 43 44 46 50 52 55 60 77 79 84 88 92 112 121 125 135 144 148
149 158 161 170 176 180 181 188 190 191 195 201 205 207 212 229 237 238 240 245 247 254 259 264 276 278 289 291 301 304 307
309 310 318 320 321 322 326 327 336 337 342 343 350 352 355 358 359 364 365 369 371 373 376 383 )
Cluster 1 has 100 functions ( 1 7 11 15 16 17 23 29 30 34 36 39 45 65 70 73 80 83 90 93 96 97 101 110 111 114 117 120 126 130
131 132 133 134 138 140 143 145 155 159 162 169 172 173 177 183 189 192 196 198 204 206 209 214 216 217 222 224 227 228 235
242 246 252 257 260 265 266 270 275 285 286 287 296 299 300 302 311 313 315 316 317 325 328 329 330 333 334 338 344 351 356
362 367 368 370 374 380 382 385 )
Cluster 2 has 105 functions ( 2 4 5 10 13 25 26 35 47 49 51 53 54 57 58 59 61 62 63 64 71 91 94 95 98 99 102 104 105 106 107
108 116 123 124 127 128 141 146 147 151 152 154 157 163 164 166 171 174 178 179 184 197 200 203 208 210 211 213 215 219 220
221 231 232 236 239 241 243 244 249 250 256 262 263 272 274 277 281 282 288 292 293 295 297 298 303 305 306 308 312 323 324
331 339 346 348 349 361 366 377 378 379 381 386 )
Cluster 3 has 96 functions ( 3 8 9 12 14 20 28 31 32 38 48 56 66 67 68 69 72 74 75 76 78 81 82 85 86 87 89 100 103 109 113 115
118 119 122 129 136 137 139 142 150 153 156 160 165 167 168 175 182 185 186 187 193 194 199 202 218 223 225 226 230 233 234
248 251 253 255 258 261 267 268 269 271 273 279 280 283 284 290 294 314 319 332 335 340 341 345 347 353 354 357 360 363 372
375 384 )

```

Figura 36 – Agrupamento ótimo encontrado para a aplicação *Ocean*.

Figura 37 – Comparação de tempos entre cada função.

4.5 Emulação de aplicação paralela

Embora neste projeto sejam consideradas apenas aplicações sequenciais para a validação do *framework*, foram emuladas cinco aplicações paralelas genéricas para analisar o comportamento de um agrupamento quando ocorre um paralelismo.

Para emular as aplicações, foram criadas matrizes de comunicação aleatoriamente para não induzir uma determinada situação. Também foram consideradas larguras de banda e funções de transferências genéricas. Nos testes realizados, as aplicações foram testadas utilizando uma taxa de transferência constante, periódica, crescente e decrescente.

4.5.1 Emulação da aplicação paralela 1

Inicialmente, foi considerada uma aplicação com 15 funções que se comunicam durante a execução da mesma. A aplicação foi dividida em níveis para simplificar a identificação do paralelismo. Cada nível apresenta um módulo em execução que possui todas as funções, sendo que essas necessariamente não se comunicam em um determinado módulo. Na Figura 38 é apresentado o esquema da aplicação paralela 1 emulada. O paralelismo ocorre todo no nível 2, conforme representado no esquema.

Para emular essa aplicação foram desenvolvidas 7 matrizes de comunicação aleatórias. Com isso, foram consideradas a comunicação acumulada e a distribuída durante o processo de agrupamento da aplicação. Dessa forma, a aplicação foi agrupada 30 vezes para cada uma das quatro taxas de transferência, considerando para uma a comunicação acumulada e a distribuída, totalizando 240 agrupamentos. Os resultados médios dessas emulações são apresentados na Tabela 17.

Por meio dos resultados obtidos pode-se perceber que o maior ganho ocorreu quando foi utilizada uma baixa taxa de transferência no final da execução da aplicação. No entanto, os

Figura 38 – Esquema geral da primeira aplicação paralela emulada.

Tabela 7 – Resultados médios em segundos da emulação da aplicação paralela 1

	Função Constante		Função Periódica		Função Crescente		Função Decrescente	
	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída
Média	170,74	159,25	230,25	228,30	96,84	94,10	1223,71	1143,47
Ganho	1,07		1,01		1,03		1,07	

piores tempos de comunicação ocorreram quando foi utilizada uma taxa de transferência baixa. Quando foi utilizada uma taxa de transferência crescente, foram obtidos os melhores tempos de comunicação, pois como o paralelismo ocorreu no final da aplicação, é importante que haja uma alta taxa de transferência para que os dados resultantes do paralelismo sejam enviados para o próximo módulo que será executado. Na Figura 39 é apresentada uma comparação das médias dos tempos de comunicação obtidos.

Figura 39 – Comparação dos tempos de comunicação obtidos através da emulação da aplicação paralela 1.

4.5.2 Emulação da aplicação paralela 2

A segunda aplicação paralela emulada possui 30 funções e 4 módulos executando em paralelo, sendo 2 módulos executados em paralelo no nível 1 e 2 módulos no nível 2. Para emular essa aplicação foram desenvolvidas 7 matrizes de comunicação aleatórias. Com isso, foram utilizadas a comunicação acumulada e a distribuída durante o processo de agrupamento da aplicação. Dessa forma, a aplicação foi agrupada 30 vezes para cada uma das quatro taxas de transferência, considerando para uma a comunicação acumulada e a distribuída, totalizando 240 agrupamentos. Os resultados médios dessas emulações são apresentados na Tabela 18.

Tabela 8 – Resultados médios em segundos da emulação da aplicação paralela 2

	Função Constante		Função Periódica		Função Crescente		Função Decrescente	
	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída
Média	801,99	798,37	858,39	851,88	542,19	539,99	4490,84	4476,04
Ganho	1,00		1,01		1,00		1,00	

Através dos resultados obtidos percebe-se que o maior ganho ocorreu quando foi utilizada uma taxa de transferência periódica, embora não apresente o melhor tempo de comunicação. Isso ocorre porque o paralelismo aconteceu durante a execução da aplicação e é importante que se tenha uma alta taxa de transferência para que os dados resultantes do paralelismo sejam enviados para o próximo módulo que será executado. Na Figura 40 é apresentada uma comparação das médias dos tempos de comunicação obtidos.

Figura 40 – Comparação dos tempos de comunicação obtidos através da emulação da aplicação paralela 2.

4.5.3 Emulação da aplicação paralela 3

A terceira aplicação paralela emulada possui 30 funções e 5 módulos executando em paralelo, sendo 2 módulos executados em paralelo no nível 1 e 3 módulos no nível 2. Para

emular essa aplicação foram desenvolvidas 7 matrizes de comunicação aleatórias. Com isso, foram utilizadas a comunicação acumulada e a distribuída durante o processo de agrupamento da aplicação. Dessa forma, a aplicação foi agrupada 30 vezes para cada uma das quatro taxas de transferência, considerando para uma a comunicação acumulada e a distribuída, totalizando 240 agrupamentos. Os resultados dessas emulações são apresentados na Tabela 19.

Tabela 9 – Resultados médios em segundos da emulação da aplicação paralela 3

	Função Constante		Função Periódica		Função Crescente		Função Decrescente	
	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída
Média	896,31	890,42	1024,02	1017,49	802,62	793,30	5483,79	5429,32
Ganho	1,01		1,01		1,01		1,01	

Embora todos os resultados obtidos apresentem o mesmo ganho, o menor tempo de comunicação ocorreu quando foi utilizada uma taxa de transferência crescente, pois grande parte do paralelismo aconteceu no final da execução da aplicação, assim é importante utilizar uma alta taxa de transferência para que os dados resultantes do paralelismo sejam enviados rapidamente para o próximo módulo que será executado. Na Figura 41 é apresentada uma comparação das médias dos tempos de comunicação obtidos.

Figura 41 – Comparação dos tempos de comunicação obtidos através da emulação da aplicação paralela 3.

4.5.4 Emulação da aplicação paralela 4

A quarta aplicação paralela emulada possui 30 funções e 5 módulos executando em paralelo, sendo 3 módulos executados em paralelo no nível 1 e 2 módulos no nível 2. Para emular essa aplicação foram desenvolvidas 7 matrizes de comunicação aleatórias. Com isso, foram utilizadas a comunicação acumulada e a distribuída durante o processo de agrupamento da aplicação. Dessa forma, a aplicação foi agrupada 30 vezes para cada uma das quatro taxas de

transferência, considerando para uma a comunicação acumulada e a distribuída, totalizando 240 agrupamentos. Os resultados dessas emulações são apresentados na Tabela 20.

Tabela 10 – Resultados médios em segundos da emulação da aplicação paralela 4

	Função Constante		Função Periódica		Função Crescente		Função Decrescente	
	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída
Média	927,96	913,30	1096,19	1085,85	623,57	617,66	5536,53	5473,73
Ganho	1,02		1,01		1,01		1,01	

Com base nos valores da tabela, nota-se que os piores tempos de comunicação ocorreram quando foi utilizada uma taxa de transferência decrescente. Como a aplicação apresenta um elevado paralelismo no começo de sua execução, deve-se utilizar uma alta taxa de transferência para que todos os módulos em paralelo possam enviar seus dados para os próximos módulos. Na Figura 42 é apresentada uma comparação das médias dos tempos de comunicação obtidos.

Figura 42 – Comparação dos tempos de comunicação obtidos através da emulação da aplicação paralela 4.

4.5.5 Emulação da aplicação paralela 5

A quinta aplicação paralela emulada possui 30 funções e 4 módulos executando em paralelo, sendo 2 módulos executados em paralelo no nível 1 e 2 módulos no nível 3. Para emular essa aplicação foram desenvolvidas 7 matrizes de comunicação aleatórias. Com isso, foram utilizadas a comunicação acumulada e a distribuída durante o processo de agrupamento da aplicação. Dessa forma, a aplicação foi agrupada 30 vezes para cada uma das quatro taxas de transferência, considerando para uma a comunicação acumulada e a distribuída, totalizando 240 agrupamentos. Os resultados dessas emulações são apresentados na Tabela 21.

Por meio dos valores da tabela, é possível observar que os piores tempos de comunicação ocorreram quando foi utilizada uma taxa de transferência decrescente. Na Figura 43 é apresentada uma comparação das médias dos tempos de comunicação obtidos.

Tabela 11 – Resultados em segundos da emulação da aplicação paralela 5

	Função Constante		Função Periódica		Função Crescente		Função Decrescente	
	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída
Média	629,08	624,41	995,58	992,73	400,51	396,37	3558,37	3515,77
Ganho	1,01		1,01		1,00		1,01

Figura 43 – Comparação dos tempos de comunicação obtidos através da emulação da aplicação paralela 5.

4.6 Considerações Finais

Neste capítulo foi apresentada a avaliação realizada para a validação do *framework* desenvolvido. Com base nos resultados dos testes, que utilizaram diferentes aplicações, pode-se observar que considerando o comportamento da comunicação durante a execução das aplicações e o meio de comunicação, pode-se obter um agrupamento aproximadamente duas vezes mais eficaz do que soluções que apenas consideram a largura de banda fixa e o montante de dados acumulados durante a aplicação.

CONCLUSÃO

A crescente exigência por sistemas computacionais mais eficientes para executar aplicações de alto desempenho impôs novos desafios à comunidade de pesquisa, que precisou buscar por novas formas de executar essas aplicações. Com isso, começou-se a utilizar plataformas heterogêneas para obter um ganho no desempenho ao executar essas aplicações.

Para utilizar todo o potencial dessas plataformas, pode-se agrupar a aplicação em grupos menores de modo que cada grupo seja executado em uma unidade de processamento específica, reduzindo assim o gargalo de comunicação, considerando o comportamento da comunicação durante a execução da aplicação, pois um determinado grupo pode executar de forma mais eficiente em um elemento de processamento do que em outro.

Com o propósito de explorar todo o potencial existente nas plataformas heterogêneas, existem na literatura métodos para agrupar aplicações que se baseiam na comunicação total existente na aplicação. Entretanto, esses métodos não consideram o meio de comunicação existente na plataforma que executará a aplicação. É importante destacar que o meio de comunicação é um fator que deve ser considerado em um agrupamento para que seja possível utilizá-lo de forma mais eficiente, evitando que a comunicação seja realizada quando ocorrer uma elevada concorrência para o uso do mesmo.

Além disso, outro fator que deve ser considerado é o comportamento da comunicação de uma aplicação durante sua execução e não apenas ao final de sua execução, pois com essa informação, é possível verificar de forma pontual os instantes que apresentam uma elevada comunicação entre funções. As abordagens propostas na literatura não consideram esse tipo de comunicação.

Nesse contexto, foi realizado um estudo sobre o agrupamento relacionado ao comportamento da comunicação interna de uma aplicação durante sua execução e sobre o meio de comunicação da plataforma. Por meio do estudo do comportamento da comunicação pode-se detectar de maneira eficaz os gargalos da aplicação.

É possível encontrar um agrupamento local para esse gargalo, de modo a reduzir o volume de dados comunicados no instante em que ocorre o gargalo. É importante considerar também o meio de comunicação e seu comportamento durante a execução da aplicação, pois durante um agrupamento, deve-se considerar qual a largura de banda que está disponível naquele instante para decidir qual a melhor forma de agrupar a aplicação.

As contribuições deste projeto são um *framework*, que pode ser utilizado no estudo de agrupamento considerando o comportamento da comunicação da aplicação e o comportamento do meio de comunicação, e um módulo extra para a ferramenta MCProf, que tem como objetivo a obtenção da comunicação da aplicação durante sua execução.

Por meio dos resultados dos testes realizados para validar o *framework*, pode-se observar que ao utilizar apenas o agrupamento obtido através da matriz de comunicação acumulada, obtém-se um resultado pior do que quando comparado aos resultados obtidos com as matrizes de comunicação distribuídas. Com o auxílio das funções de transferência de dados, observa-se como a clusterização se comporta quando a largura de banda muda durante a execução da aplicação. Algumas aplicações apresentam um elevado desempenho quando possuem uma alta largura de banda no início de sua execução, enquanto outras aplicações têm alto desempenho com uma banda alta no final de sua execução.

5.1 Limitações e Trabalhos futuros

Embora o *framework* possua diversos recursos que são úteis para a análise da distribuição de carga de trabalho em plataformas heterogêneas, ainda há algumas limitações que precisam ser levadas em consideração, como:

- Ao realizar o agrupamento considera que todos os grupos se comunicam entre si e possuem a mesma largura de banda e a mesma taxa de transmissão;
- Apenas as aplicações com funcionamento sequencial são consideradas;

Em relação a trabalhos futuros, pode-se considerar os diferentes métodos de comunicação existentes em uma plataforma heterogênea, bem como quais grupos que se comunicam entre si para desenvolver uma técnica de agrupamento que inclua mais possibilidades consistentes com o mundo real. Outro trabalho futuro é a realização de um estudo sobre aplicações paralelas com o objetivo de analisar como a comunicação se comporta durante suas execuções.

REFERÊNCIAS

ASHRAF, I.; OSTADZADEH, S. A.; MEEUWS, R.; BERTELS, K. Evaluation methodology for data communication-aware application partitioning. In: SPRINGER. **Euro-Par 2013: Parallel Processing Workshops**. [S.l.], 2013. p. 739–748. Citado nas páginas 15, 28, 34 e 39.

BERKHIN, P. A survey of clustering data mining techniques. In: **Grouping multidimensional data**. [S.l.]: Springer, 2006. p. 25–71. Citado na página 27.

BITTNER, R. Speedy bus mastering pci express. In: IEEE. **Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on**. [S.l.], 2012. p. 523–526. Citado na página 30.

BONATO, V.; MARQUES, E.; CONSTANTINIDES, G. A parallel hardware architecture for scale and rotation invariant feature detection. **Circuits and Systems for Video Technology, IEEE Transactions on**, v. 18, n. 12, p. 1703–1712, Dec 2008. ISSN 1051-8215. Citado na página 23.

BORKAR, S.; CHIEN, A. A. The future of microprocessors. **Commun. ACM**, ACM, New York, NY, USA, v. 54, n. 5, p. 67–77, maio 2011. ISSN 0001-0782. Disponível em: <<http://doi.acm.org/10.1145/1941487.1941507>>. Citado na página 23.

BORKAR, S. Y.; DUBEY, P.; KAHN, K. C.; KUCK, D. J.; MULDER, H.; RAMANATHAN, E. R. M.; THOMAS, V.; CORPORATION, I.; PAWLOWSKI, S. S. **Intel ® Processor and Platform Evolution for the Next Decade Executive Summary**. 2005. Citado na página 23.

CHE, S.; LI, J.; SHEAFFER, J.; SKADRON, K.; LACH, J. Accelerating compute-intensive applications with gpus and fpgas. In: **Application Specific Processors, 2008. SASP 2008. Symposium on**. [S.l.: s.n.], 2008. p. 101–107. Citado na página 23.

GLOVER, F. Tabu search—part i. **ORSA Journal on computing**, INFORMS, v. 1, n. 3, p. 190–206, 1989. Citado na página 36.

GOMES, A. Uma introdução à busca tabu. **Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo, SP, Brasil. Recuperado de: <http://www.ime.usp.br/~gold/cursos/2009/mac5758/AndreBuscaTabu.pdf>**, 2009. Citado na página 36.

GRAHAM, S. L.; KESSLER, P. B.; MCKUSICK, M. K. Gprof: A call graph execution profiler. **SIGPLAN Not.**, ACM, New York, NY, USA, v. 17, n. 6, p. 120–126, jun. 1982. ISSN 0362-1340. Disponível em: <<http://doi.acm.org/10.1145/872726.806987>>. Citado nas páginas 24 e 37.

GREENGARD, L. **The rapid evaluation of potential fields in particle systems**. [S.l.]: MIT press, 1988. Citado na página 61.

HAYNES, W. Student's t-test. In: **Encyclopedia of Systems Biology**. [S.l.]: Springer, 2013. p. 2023–2025. Citado na página 53.

HOROWITZ, M.; DALLY, W. How scaling will change processor architecture. In: IEEE. **Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International**. [S.l.], 2004. p. 132–133. Citado na página 23.

I. Ashraf and V.M. Sima and K.L.M. Bertels. Intra-application data-communication characterization. In: **Proc. 1st International Workshop on Communication Architectures at Extreme Scale**. Frankfurt, Germany: [s.n.], 2015. Citado nas páginas 15, 27, 28, 32, 33 e 39.

INTEL. **Pin - A Dynamic Binary Instrumentation Tool**. 2012. Disponível em: <<https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool>>. Citado nas páginas 15, 33 e 34.

KIRK, D. B.; HWU, W.-m. W. **Programming Massively Parallel Processors: A Hands-on Approach**. 1st. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2010. ISBN 0123814723, 9780123814722. Citado na página 23.

KUMAR, S.; JANTSCH, A.; SOININEN, J.-P.; FORSELL, M.; MILLBERG, M.; OBERG, J.; TIENSYRJA, K.; HEMANI, A. A network on chip architecture and design methodology. In: IEEE. **VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on**. [S.l.], 2002. p. 117–124. Citado na página 31.

KUNDERT, K. S.; SANGIOVANNI-VINCENTELLI, A. Sparse user's guide—a sparse linear equation solver version 1.3 a. **University of California, Berkeley**, v. 178, 1988. Citado na página 57.

LUCAS, B. D.; KANADE, T. *et al.* An iterative image registration technique with an application to stereo vision. Vancouver, BC, Canada, 1981. Citado na página 54.

MARSAGLIA, G.; ZAMAN, A. **The KISS generator**. [S.l.], 1993. Citado na página 36.

MIRKIN, B. **Clustering: a data recovery approach**. [S.l.]: CRC Press, 2012. Citado nas páginas 15 e 30.

NETHERCOTE, N.; SEWARD, J. Valgrind: a framework for heavyweight dynamic binary instrumentation. In: ACM. **ACM Sigplan notices**. [S.l.], 2007. v. 42, n. 6, p. 89–100. Citado na página 24.

OSTADZADEH, S. A.; MEEUWS, R. J.; GALUZZI, C.; BERTELS, K. Quad—a memory access pattern analyser. In: **Reconfigurable computing: architectures, tools and applications**. [S.l.]: Springer, 2010. p. 269–281. Citado nas páginas 37 e 39.

PELLEG, D.; MOORE, A. W. *et al.* X-means: Extending k-means with efficient estimation of the number of clusters. In: **Icml**. [S.l.: s.n.], 2000. v. 1, p. 727–734. Citado na página 29.

PHAM, D.; KARABOGA, D. **Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks**. [S.l.]: Springer Science & Business Media, 2012. Citado na página 36.

PHAM-QUOC, C.; ASHRAF, I.; AL-ARS, Z.; BERTELS, K. Heterogeneous hardware accelerators with hybrid interconnect: an automated design approach. 2015. Citado nas páginas 15, 31, 37 e 39.

- RICHARDSON, T. D.; NICOPOULOS, C.; PARK, D.; NARAYANAN, V.; XIE, Y.; DAS, C.; DEGALAHAL, V. A hybrid soc interconnect with dynamic tdma-based transaction-less buses and on-chip networks. In: IEEE. **VLSI Design, 2006. Held jointly with 5th International Conference on Embedded Systems and Design., 19th International Conference on.** [S.l.], 2006. p. 8–pp. Citado nas páginas 38 e 39.
- SANCHEZ, D.; MICHELOGIANNAKIS, G.; KOZYRAKIS, C. An analysis of on-chip interconnection networks for large-scale chip multiprocessors. **ACM Transactions on Architecture and Code Optimization (TACO)**, ACM, v. 7, n. 1, p. 4, 2010. Citado na página 30.
- SEWARD, J.; NETHERCOTE, N. Using valgrind to detect undefined value errors with bit-precision. In: **USENIX Annual Technical Conference, General Track.** [S.l.: s.n.], 2005. p. 17–30. Citado na página 32.
- SINGH, A. **Spectral Clustering**. 2010. Disponível em: <https://www.cs.cmu.edu/~aarti/Class/10701/slides/Lecture21_2.pdf>. Citado na página 30.
- SINGH, J. P. Parallel hierarchical n-body methods and their implications for multiprocessors. Stanford University, 1993. Citado na página 59.
- SINGH, J. P.; WEBER, W.-D.; GUPTA, A. Splash: Stanford parallel applications for shared-memory. **ACM SIGARCH Computer Architecture News**, ACM, v. 20, n. 1, p. 5–44, 1992. Citado na página 59.
- TSAI, K.-L.; LAI, F.; PAN, C.-Y.; XIAO, D.-S.; TAN, H.-J.; LEE, H.-C. Design of low latency on-chip communication based on hybrid noc architecture. In: IEEE. **NEWCAS Conference (NEWCAS), 2010 8th IEEE International.** [S.l.], 2010. p. 257–260. Citado nas páginas 38 e 39.
- WILCOXON, F. Individual comparisons by ranking methods. **Biometrics bulletin**, JSTOR, v. 1, n. 6, p. 80–83, 1945. Citado na página 53.
- WOO, S. C.; OHARA, M.; TORRIE, E.; SINGH, J. P.; GUPTA, A. The splash-2 programs: Characterization and methodological considerations. In: ACM. **ACM SIGARCH computer architecture news.** [S.l.], 1995. v. 23, n. 2, p. 24–36. Citado na página 64.
- XU, R.; WUNSCH, D. Survey of clustering algorithms. **IEEE Transactions on neural networks**, Ieee, v. 16, n. 3, p. 645–678, 2005. Citado nas páginas 29 e 30.
- ZAKI, M. J.; JR, W. M.; MEIRA, W. **Data mining and analysis: fundamental concepts and algorithms.** [S.l.]: Cambridge University Press, 2014. Citado nas páginas 15, 28, 29 e 30.
- ZHAO, Y.; KARYPIS, G. Evaluation of hierarchical clustering algorithms for document datasets. In: ACM. **Proceedings of the eleventh international conference on Information and knowledge management.** [S.l.], 2002. p. 515–524. Citado nas páginas 38 e 39.

TABELAS COM OS TEMPOS OBTIDOS

Tabela 12 – Resultados em segundos dos tempos encontrados no agrupamento da aplicação KLT.

Execução	Função Periódica		Função Crescente		Função Decrescente	
	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída
1	1,173	0,555	0,648	0,648	2,611	2,611
2	1,698	1,173	0,632	0,501	3,645	2,334
3	1,708	1,708	0,495	0,495	3,671	3,671
4	1,708	1,274	0,501	0,501	1,088	1,088
5	0,970	0,970	0,501	0,501	3,674	3,666
6	1,706	0,970	0,881	0,245	3,645	3,645
7	1,701	1,701	0,887	0,649	3,664	2,413
8	1,700	1,700	0,392	0,392	3,651	1,272
9	1,281	0,546	0,499	0,499	3,671	1,286
10	1,707	0,546	0,884	0,649	1,086	1,086
11	0,969	0,969	0,499	0,499	2,600	1,372
12	0,972	0,969	0,499	0,499	2,414	1,274
13	1,706	0,972	0,887	0,633	1,373	1,373
14	1,166	1,166	0,396	0,396	2,607	2,607
15	0,981	0,981	0,887	0,887	3,670	3,670
16	1,695	0,981	0,884	0,884	2,412	2,412
17	1,706	1,706	0,631	0,631	3,664	1,284
18	1,695	0,747	0,497	0,265	3,661	3,661
19	1,175	0,747	0,246	0,246	2,620	2,620
20	1,694	1,175	0,651	0,651	1,278	1,278
21	1,699	1,694	0,651	0,650	3,671	2,324
22	1,171	1,171	0,495	0,495	2,431	2,431
23	1,173	0,978	0,885	0,650	3,666	2,312
24	1,706	0,978	0,633	0,633	2,622	2,622
25	1,167	1,167	0,886	0,631	2,407	1,374
26	0,443	0,443	0,265	0,265	2,333	2,333
27	0,981	0,443	0,650	0,500	2,615	1,380
28	1,703	1,706	0,887	0,632	2,431	2,431
29	1,703	1,169	0,884	0,884	3,643	2,331
30	1,169	1,169	0,630	0,261	1,285	1,285
Média	1,401	1,082	0,642	0,542	2,794	2,182
Desvio Padrão	0,356	0,390	0,202	0,179	0,897	0,870
Ganho	1,294		1,184		1,281	

Tabela 13 – Resultados em segundos dos tempos encontrados no agrupamento da aplicação Sparse

Execução	Função Periódica		Função Crescente		Função Decrescente	
	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída
1	0,899	0,405	0,164	0,064	0,483	0,419
2	0,900	0,319	0,086	0,061	0,617	0,235
3	0,796	0,282	0,136	0,076	0,540	0,358
4	0,394	0,364	0,093	0,071	0,745	0,454
5	0,968	0,277	0,144	0,068	0,415	0,234
6	0,813	0,387	0,138	0,081	0,458	0,408
7	0,516	0,653	0,128	0,090	0,617	0,428
8	0,435	0,293	0,122	0,051	0,535	0,417
9	0,826	0,378	0,129	0,057	0,484	0,368
10	0,819	0,300	0,088	0,085	0,711	0,368
11	0,863	0,462	0,082	0,061	0,540	0,425
12	0,937	0,284	0,145	0,082	0,443	0,287
13	0,870	0,331	0,141	0,082	0,434	0,292
14	0,347	0,359	0,130	0,059	0,414	0,390
15	0,926	0,331	0,125	0,086	0,496	0,393
16	0,997	0,759	0,162	0,074	0,463	0,270
17	0,914	0,807	0,131	0,076	0,671	0,380
18	0,884	0,349	0,140	0,073	0,583	0,340
19	0,742	0,292	0,131	0,063	0,551	0,399
20	0,961	0,496	0,093	0,071	0,597	0,444
21	0,921	0,370	0,125	0,069	0,625	0,278
22	0,415	0,464	0,115	0,054	0,608	0,453
23	0,988	0,520	0,138	0,081	0,622	0,238
24	0,307	0,252	0,085	0,084	0,462	0,323
25	0,737	0,296	0,137	0,083	0,584	0,402
26	0,869	0,365	0,152	0,081	0,795	0,407
27	0,473	0,456	0,088	0,085	0,433	0,411
28	0,801	0,347	0,134	0,081	0,521	0,355
29	0,440	0,320	0,114	0,050	0,559	0,393
30	0,350	0,385	0,100	0,081	0,681	0,390
Média	0,737	0,397	0,123	0,073	0,556	0,365
Desvio Padrão	0,230	0,136	0,024	0,011	0,101	0,066
Ganho	1,858		1,695		1,523	

Tabela 14 – Resultados em segundos dos tempos encontrados no agrupamento da aplicação Barnes

Execução	Função Periódica		Função Crescente		Função Decrescente	
	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída
1	0,0078	0,0062	0,0039	0,0007	0,0187	0,0019
2	0,0022	0,0022	0,0044	0,0007	0,0183	0,0045
3	0,0073	0,0066	0,0042	0,0006	0,0181	0,0034
4	0,0071	0,0015	0,0044	0,0005	0,0041	0,0047
5	0,0076	0,0065	0,0039	0,0013	0,0158	0,0016
6	0,0075	0,0021	0,0039	0,0012	0,0019	0,0161
7	0,0074	0,0068	0,0036	0,0008	0,0041	0,0009
8	0,0070	0,0020	0,0038	0,0013	0,0177	0,0045
9	0,0068	0,0011	0,0040	0,0007	0,0162	0,0048
10	0,0075	0,0009	0,0041	0,0012	0,0045	0,0032
11	0,0078	0,0013	0,0013	0,0013	0,0030	0,0042
12	0,0077	0,0008	0,0043	0,0038	0,0169	0,0042
13	0,0070	0,0005	0,0043	0,0037	0,0182	0,0053
14	0,0077	0,0014	0,0008	0,0008	0,0176	0,0033
15	0,0019	0,0065	0,0047	0,0012	0,0160	0,0163
16	0,0078	0,0015	0,0013	0,0007	0,0174	0,0016
17	0,0073	0,0022	0,0041	0,0013	0,0165	0,0026
18	0,0078	0,0014	0,0013	0,0008	0,0028	0,0025
19	0,0074	0,0021	0,0011	0,0006	0,0163	0,0038
20	0,0078	0,0015	0,0013	0,0005	0,0034	0,0165
21	0,0077	0,0018	0,0046	0,0003	0,0164	0,0006
22	0,0068	0,0065	0,0008	0,0005	0,0165	0,0041
23	0,0071	0,0065	0,0041	0,0004	0,0179	0,0045
24	0,0068	0,0070	0,0012	0,0035	0,0045	0,0035
25	0,0069	0,0015	0,0042	0,0013	0,0156	0,0034
26	0,0068	0,0017	0,0043	0,0008	0,0032	0,0143
27	0,0073	0,0008	0,0005	0,0037	0,0036	0,0012
28	0,0069	0,0074	0,0045	0,0011	0,0182	0,0022
29	0,0074	0,0016	0,0013	0,0006	0,0187	0,0011
30	0,0072	0,0013	0,0042	0,0007	0,0158	0,0029
Média	0,0070	0,0030	0,0032	0,0012	0,0126	0,0048
Desvio Padrão	0,0014	0,0024	0,0015	0,0010	0,0066	0,0046
Ganho	2,295		2,587		2,621	

Tabela 15 – Resultados em segundos dos tempos encontrados agrupando a aplicação FMM

Execução	Função Periódica		Função Crescente		Função Decrescente	
	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída
1	0,000010	0,000008	0,000006	0,000008	0,000028	0,000027
2	0,000008	0,000008	0,000007	0,000008	0,000024	0,000024
3	0,000010	0,000006	0,000008	0,000008	0,000026	0,000023
4	0,000009	0,000007	0,000006	0,000008	0,000025	0,000024
5	0,000010	0,000008	0,000008	0,000009	0,000020	0,000022
6	0,000010	0,000008	0,000010	0,000006	0,000030	0,000019
7	0,000010	0,000008	0,000010	0,000008	0,000030	0,000017
8	0,000008	0,000006	0,000010	0,000006	0,000024	0,000026
9	0,000010	0,000006	0,000008	0,000008	0,000028	0,000020
10	0,000008	0,000007	0,000009	0,000007	0,000021	0,000023
11	0,000007	0,000007	0,000008	0,000006	0,000026	0,000024
12	0,000008	0,000008	0,000010	0,000007	0,000024	0,000016
13	0,000008	0,000007	0,000010	0,000008	0,000019	0,000021
14	0,000006	0,000005	0,000009	0,000008	0,000026	0,000025
15	0,000009	0,000006	0,000009	0,000009	0,000030	0,000022
16	0,000009	0,000008	0,000010	0,000006	0,000032	0,000019
17	0,000008	0,000007	0,000008	0,000006	0,000027	0,000020
18	0,000010	0,000008	0,000007	0,000007	0,000029	0,000024
19	0,000010	0,000009	0,000010	0,000006	0,000017	0,000022
20	0,000008	0,000008	0,000007	0,000009	0,000030	0,000026
21	0,000010	0,000008	0,000010	0,000007	0,000031	0,000017
22	0,000009	0,000008	0,000009	0,000007	0,000031	0,000022
23	0,000010	0,000007	0,000009	0,000008	0,000025	0,000022
24	0,000010	0,000006	0,000009	0,000007	0,000030	0,000023
25	0,000008	0,000007	0,000009	0,000007	0,000029	0,000019
26	0,000008	0,000007	0,000008	0,000007	0,000029	0,000025
27	0,000010	0,000008	0,000009	0,000009	0,000024	0,000022
28	0,000009	0,000007	0,000010	0,000008	0,000029	0,000019
29	0,000009	0,000008	0,000008	0,000009	0,000022	0,000024
30	0,000010	0,000008	0,000009	0,000008	0,000018	0,000018
Média	0,000009	0,000007	0,000009	0,000008	0,000026	0,000022
Desvio Padrão	0,000001	0,000001	0,000001	0,000001	0,000004	0,000003
Ganho	1,223		1,148		1,198	

Tabela 16 – Resultados em segundos dos tempos encontrados no agrupamento da aplicação *Ocean*

Execução	Função Periódica		Função Crescente		Função Decrescente	
	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída	Comunicação Acumulada	Comunicação Distribuída
1	0,0023	0,0011	0,0006	0,0005	0,0042	0,0021
2	0,0021	0,0017	0,0008	0,0007	0,0039	0,0031
3	0,0023	0,0015	0,0008	0,0006	0,0031	0,0029
4	0,0027	0,0016	0,0008	0,0007	0,0031	0,0027
5	0,0021	0,0015	0,0008	0,0007	0,0039	0,0023
6	0,0020	0,0020	0,0008	0,0007	0,0040	0,0030
7	0,0018	0,0018	0,0007	0,0006	0,0030	0,0021
8	0,0025	0,0018	0,0008	0,0007	0,0033	0,0032
9	0,0027	0,0019	0,0008	0,0007	0,0039	0,0028
10	0,0022	0,0019	0,0008	0,0006	0,0039	0,0014
11	0,0024	0,0016	0,0007	0,0006	0,0036	0,0030
12	0,0012	0,0016	0,0009	0,0008	0,0040	0,0030
13	0,0021	0,0017	0,0008	0,0008	0,0045	0,0023
14	0,0026	0,0023	0,0008	0,0006	0,0038	0,0032
15	0,0027	0,0018	0,0009	0,0007	0,0037	0,0030
16	0,0022	0,0017	0,0009	0,0005	0,0036	0,0025
17	0,0024	0,0013	0,0009	0,0005	0,0033	0,0022
18	0,0021	0,0017	0,0010	0,0006	0,0033	0,0033
19	0,0021	0,0022	0,0007	0,0007	0,0039	0,0029
20	0,0024	0,0018	0,0007	0,0007	0,0034	0,0034
21	0,0023	0,0022	0,0008	0,0007	0,0043	0,0031
22	0,0023	0,0019	0,0009	0,0007	0,0030	0,0034
23	0,0022	0,0015	0,0009	0,0006	0,0039	0,0020
24	0,0019	0,0018	0,0006	0,0006	0,0041	0,0029
25	0,0017	0,0018	0,0008	0,0006	0,0036	0,0032
26	0,0021	0,0014	0,0009	0,0005	0,0034	0,0032
27	0,0020	0,0022	0,0006	0,0006	0,0038	0,0031
28	0,0019	0,0010	0,0009	0,0005	0,0040	0,0032
29	0,0023	0,0018	0,0006	0,0008	0,0032	0,0030
30	0,0016	0,0017	0,0009	0,0006	0,0041	0,0029
Média	0,0022	0,0017	0,0008	0,0006	0,0037	0,0028
Desvio Padrão	0,0003	0,0003	0,0001	0,0001	0,0004	0,0005
Ganho	1,265		1,265		1,312	

Tabela 17 – Resultados médios em segundos da emulação da aplicação paralela 1

Execução	Função Constante		Função Periódica		Função Crescente		Função Decrescente	
	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída
1	170,04	170,04	230,54	230,54	85,07	85,07	804,54	804,54
2	170,04	170,04	230,54	230,54	111,10	103,69	1116,89	1116,89
3	154,81	160,98	267,66	238,07	111,10	122,81	1116,89	1116,89
4	172,50	160,98	210,08	238,07	105,74	105,74	1251,84	1034,83
5	172,50	172,56	210,08	232,09	105,74	105,74	1331,59	1034,83
6	178,67	164,42	208,02	232,09	105,02	84,60	1331,59	1134,93
7	178,67	164,42	208,02	208,02	105,02	84,60	1277,82	1134,93
8	162,60	112,58	236,65	235,87	97,87	87,24	1277,82	1150,87
9	162,60	112,58	236,65	236,65	97,87	87,24	1227,02	1199,78
10	181,44	144,50	231,72	236,65	123,01	114,40	1227,02	1048,38
11	181,44	158,69	231,72	232,88	91,05	114,40	1298,16	1231,05
12	170,13	164,77	216,21	236,86	91,05	106,11	1298,16	1231,05
13	170,13	164,77	216,21	236,86	97,57	106,11	1305,13	1219,34
14	184,77	162,60	255,24	254,51	97,57	94,52	1305,13	1219,34
15	184,27	162,60	255,24	254,51	114,43	94,52	1250,73	1246,80
16	184,27	170,17	223,97	255,09	114,43	100,76	1250,73	1246,80
17	180,67	170,17	223,97	255,09	68,70	70,41	1128,84	1128,84
18	180,67	173,54	231,71	187,05	68,70	70,41	1128,84	1128,84
19	175,92	173,54	231,71	187,05	122,57	89,34	1291,40	1126,24
20	175,92	175,92	237,38	217,45	122,57	89,34	1018,37	1126,24
21	143,40	175,92	245,72	217,45	91,12	116,26	1018,37	1018,37
22	143,40	148,79	245,72	245,72	82,74	116,26	1288,57	1018,37
23	179,13	148,79	237,36	245,72	82,74	96,67	1288,57	1101,95
24	179,13	179,13	237,36	206,98	95,02	96,67	1342,27	1101,95
25	164,85	164,85	234,86	206,98	95,02	94,98	1342,27	1280,54
26	164,85	125,58	234,86	234,86	87,54	87,54	1221,96	1221,96
27	178,06	125,58	202,59	202,59	87,54	87,54	1221,96	1221,96
28	178,06	155,00	202,59	202,59	65,45	65,45	1249,17	1249,17
29	149,65	172,00	236,58	202,59	65,45	65,45	1249,17	1249,17
30	149,65	172,00	236,58	247,69	116,42	79,28	1250,40	1159,29
Média	170,74	159,25	230,25	228,30	96,84	94,10	1223,71	1143,47
Ganho	1,07		1,01		1,03		1,07	

Tabela 18 – Resultados em segundos da emulação da aplicação paralela 2

Execução	Função Constante		Função Periódica		Função Crescente		Função Decrescente	
	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída
1	790,52	799,54	859,38	859,38	563,18	563,18	4720,18	4633,54
2	790,52	790,52	813,92	813,92	524,56	563,18	4720,18	4720,18
3	793,88	790,52	859,41	813,92	524,56	524,56	4587,97	4587,97
4	815,42	793,88	859,41	859,41	536,54	524,56	4141,73	4141,73
5	815,42	815,42	836,78	836,78	530,14	536,54	4141,73	4141,73
6	813,65	813,65	866,24	836,78	530,14	530,14	4551,32	4551,32
7	802,63	802,63	910,98	866,24	552,99	530,14	4482,86	4482,86
8	802,63	802,63	910,98	910,98	558,74	552,99	4482,86	4482,86
9	787,85	787,85	882,20	882,20	558,74	558,74	4689,88	4689,88
10	810,38	787,85	809,80	882,20	533,17	533,17	4410,28	4410,28
11	810,38	810,38	809,80	809,80	552,85	533,17	4410,28	4410,28
12	803,85	803,85	866,13	809,80	552,85	552,85	4539,08	4539,08
13	795,27	795,27	872,85	866,13	555,85	552,85	4502,74	4502,74
14	812,83	795,27	872,85	872,85	566,58	555,85	4502,74	4502,74
15	812,83	812,83	877,10	872,85	566,58	566,58	4359,62	4359,62
16	801,73	801,73	870,91	870,91	543,10	543,10	4386,80	4359,62
17	810,17	801,73	870,91	870,91	565,49	543,10	4503,22	4386,80
18	810,17	810,17	809,60	809,60	565,49	565,49	4503,22	4503,22
19	817,06	810,17	821,71	809,60	550,08	550,08	4471,63	4471,63
20	804,92	804,92	821,71	821,71	537,90	537,90	4591,08	4471,63
21	804,92	804,92	839,92	839,92	537,90	537,90	4591,08	4591,08
22	773,75	773,75	874,10	839,92	539,74	539,74	4685,41	4591,08
23	811,02	773,75	874,10	874,10	493,18	493,18	4604,54	4604,54
24	811,02	811,02	849,04	849,04	493,18	493,18	4604,54	4604,54
25	810,83	810,83	872,60	849,04	528,79	528,79	4495,06	4495,06
26	770,96	770,96	872,60	872,60	530,14	528,79	4051,98	4051,98
27	770,96	770,96	865,96	865,96	541,79	530,14	4051,98	4051,98
28	813,63	813,63	876,86	865,96	541,79	541,79	4691,74	4691,74
29	795,25	795,25	876,86	876,86	546,25	541,79	4624,75	4624,75
30	795,25	795,25	846,90	846,90	543,48	546,25	4624,75	4624,75
Média	801,99	798,37	858,39	851,88	542,19	539,99	4490,84	4476,04
Ganho	1,00		1,01		1,00		1,00	

Tabela 19 – Resultados em segundos da emulação da aplicação paralela 3

Execução	Função Constante		Função Periódica		Função Crescente		Função Decrescente	
	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída
1	940,60	940,60	1057,94	1057,94	844,20	844,20	5453,49	5453,49
2	873,52	873,52	1027,04	1027,04	806,10	806,10	5373,75	5373,75
3	873,52	873,52	1014,17	1014,17	768,13	768,13	5254,98	5254,98
4	937,25	937,25	966,43	1014,17	768,13	768,13	5571,77	5254,98
5	866,04	866,04	966,43	966,43	814,95	814,95	5571,77	5571,77
6	915,10	866,04	1038,05	1038,05	764,22	764,22	5316,48	5316,48
7	915,10	915,10	1085,26	1038,05	764,22	764,22	5470,91	5316,48
8	922,33	915,10	1018,45	1018,45	818,57	818,57	5609,20	5470,91
9	905,42	905,42	972,65	972,65	811,53	811,53	5640,87	5609,20
10	905,42	905,42	972,65	972,65	811,53	811,53	5640,87	5640,87
11	909,31	905,42	1033,95	1033,95	824,29	824,29	5478,13	5478,13
12	925,65	909,31	1020,39	1020,39	829,37	824,29	5426,07	5426,07
13	925,65	925,65	1048,57	1020,39	772,81	829,37	5583,73	5426,07
14	856,42	856,42	1043,42	1043,42	772,81	772,81	5583,73	5583,73
15	920,98	856,42	1065,52	1043,42	812,47	772,81	5425,19	5425,19
16	920,98	920,98	1065,52	1065,52	810,59	812,47	5561,02	5425,19
17	897,33	897,33	1004,63	1004,63	810,59	810,59	5438,92	5438,92
18	901,56	897,33	1064,14	1004,63	832,56	810,59	5447,60	5438,92
19	901,56	901,56	1015,87	1015,87	809,51	809,51	5447,60	5447,60
20	916,00	916,00	978,06	978,06	809,51	809,51	5421,91	5421,91
21	912,71	912,71	978,06	978,06	781,68	781,68	5551,71	5421,91
22	867,02	912,71	1004,25	1004,25	839,75	781,68	5534,41	5534,41
23	867,02	867,02	1035,30	1004,25	839,75	839,75	5534,41	5534,41
24	914,65	867,02	1061,58	1035,30	740,56	740,56	5170,18	5170,18
25	862,71	914,65	1070,71	1061,58	823,04	740,56	5304,54	5170,18
26	862,71	862,71	1070,71	1070,71	823,04	823,04	5425,49	5304,54
27	888,02	862,71	1036,19	1036,19	695,65	695,65	5489,99	5425,49
28	842,92	842,92	1012,47	1012,47	826,28	695,65	5489,99	5489,99
29	842,92	842,92	985,95	985,95	826,28	826,28	5731,15	5489,99
30	898,83	842,92	1006,31	985,95	826,35	826,35	5563,84	5563,84
Média	896,31	890,42	1024,02	1017,49	802,62	793,30	5483,79	5429,32
Ganho	1,01		1,01		1,01		1,01	

Tabela 20 – Resultados em segundos da emulação da aplicação paralela 4

Execução	Função Constante		Função Periódica		Função Crescente		Função Decrescente	
	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída
1	939,19	939,19	1124,91	1124,91	634,40	620,80	5658,18	5143,50
2	939,19	939,19	1069,28	1069,28	626,47	634,40	5658,18	5658,18
3	919,77	919,77	1069,56	1069,28	626,47	626,47	5663,85	5663,85
4	844,38	844,38	1114,37	1069,56	655,62	626,47	5560,76	5560,76
5	924,52	844,38	1088,07	1088,07	627,65	627,65	5639,78	5560,76
6	948,42	924,52	1088,07	1088,07	629,16	627,65	5680,20	5639,78
7	948,42	948,42	1101,56	1101,56	629,16	629,16	5680,20	5680,20
8	916,17	916,17	1072,98	1072,98	588,83	588,83	5409,63	5409,63
9	889,31	889,31	987,51	987,51	591,68	588,83	5568,20	5409,63
10	946,81	889,31	1106,05	987,51	646,06	591,68	5664,25	5568,20
11	946,81	946,81	1106,05	1106,05	622,46	622,46	5242,07	5242,07
12	943,81	943,81	1032,65	1032,65	622,46	622,46	5687,48	5242,07
13	858,88	858,88	1087,16	1032,65	650,92	650,92	5687,48	5687,48
14	956,27	858,88	1111,25	1087,16	644,19	644,19	5689,78	5689,78
15	897,75	897,75	1081,42	1111,25	620,99	620,99	5656,76	5656,76
16	897,75	897,75	1081,42	1081,42	645,80	645,80	5499,33	5499,33
17	954,06	954,06	1080,89	1080,89	618,90	618,90	5412,61	5499,33
18	937,81	937,81	1067,49	1067,49	646,92	618,90	5412,61	5412,61
19	927,00	927,00	1117,82	1067,49	615,60	615,60	5616,63	5616,63
20	961,98	927,00	1117,82	1117,82	615,60	615,60	5715,72	5616,63
21	961,98	961,98	1138,63	1138,63	575,28	575,28	5800,55	5715,72
22	954,23	954,23	1088,18	1088,18	588,67	575,28	5193,19	5193,19
23	873,60	873,60	1101,42	1088,18	652,16	588,67	5193,19	5193,19
24	946,73	873,60	1135,37	1101,42	631,03	652,16	5233,24	5233,24
25	946,73	946,73	1135,37	1135,37	631,03	631,03	5685,78	5233,24
26	975,46	975,46	1112,51	1112,51	620,84	620,84	5668,19	5668,19
27	955,48	955,48	1072,34	1072,34	617,19	617,19	5530,78	5530,78
28	860,23	860,23	1102,25	1102,25	617,19	617,19	5281,22	5281,22
29	932,98	860,23	1102,25	1102,25	607,18	607,18	5281,22	5281,22
30	932,98	932,98	1190,92	1190,92	607,18	607,18	5424,88	5424,88
Média	927,96	913,30	1096,19	1085,85	623,57	617,66	5536,53	5473,73
Ganho	1,02		1,01		1,01		1,01	

Tabela 21 – Resultados em segundos da emulação da aplicação paralela 5

Execução	Função Constante		Função Periódica		Função Crescente		Função Decrescente	
	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída	Acumulada	Distribuída
1	630,19	623,73	999,29	996,53	381,33	381,33	3619,76	3619,76
2	630,19	630,19	1000,15	999,29	405,73	381,33	3612,84	3612,84
3	634,38	634,38	1000,15	1000,15	405,73	405,73	3562,02	3562,02
4	643,48	634,38	1015,61	1015,61	378,24	378,24	3562,02	3562,02
5	655,04	643,48	1041,72	1015,61	401,68	378,24	3739,85	3739,85
6	655,04	655,04	995,75	995,75	394,51	394,51	3367,06	3367,06
7	647,15	647,15	995,75	995,75	419,44	394,51	3571,92	3367,06
8	636,96	636,96	1024,20	1024,20	419,44	419,44	3571,92	3571,92
9	650,56	636,96	975,34	975,34	422,37	419,44	3684,18	3684,18
10	650,56	650,56	990,90	975,34	395,79	395,79	3571,69	3571,69
11	646,38	646,38	1023,30	990,90	391,04	391,04	3550,74	3550,74
12	594,15	594,15	1023,30	1023,30	391,04	391,04	3590,37	3550,74
13	580,08	580,08	1023,84	1023,84	397,33	397,33	3590,37	3590,37
14	608,63	580,08	1004,26	1004,26	382,65	382,65	3256,09	3256,09
15	608,63	608,63	999,73	999,73	382,65	382,65	3883,66	3256,09
16	631,98	631,98	999,73	999,73	398,96	398,96	3593,49	3593,49
17	631,77	631,77	1018,39	1018,39	395,58	395,58	3593,49	3593,49
18	610,04	610,04	1036,24	1018,39	413,92	395,58	3631,56	3631,56
19	610,04	610,04	984,40	984,40	388,22	413,92	3275,23	3275,23
20	619,27	619,27	979,32	984,40	388,22	388,22	3644,82	3275,23
21	622,04	619,27	979,32	979,32	415,80	388,22	3376,81	3376,81
22	650,73	622,04	990,85	990,85	399,93	399,93	3376,81	3376,81
23	650,73	650,73	955,86	955,86	427,25	399,93	3460,69	3460,69
24	605,50	605,50	972,95	955,86	427,25	427,25	3679,72	3460,69
25	617,31	605,50	972,95	972,95	415,38	415,38	3788,84	3679,72
26	641,85	617,31	988,84	988,84	416,13	415,38	3496,82	3788,84
27	623,88	623,88	974,76	974,76	372,98	372,98	3496,82	3496,82
28	623,88	623,88	995,05	974,76	372,98	372,98	3557,89	3557,89
29	629,48	629,48	952,72	995,05	406,81	406,81	3527,01	3527,01
30	632,44	629,48	952,72	952,72	406,81	406,81	3516,51	3516,51
Média	629,08	624,41	995,58	992,73	400,51	396,37	3558,37	3515,77
Ganho	1,01		1,01		1,00		1,01	

TABELAS COM OS VALORES DE p OBTIDOS

Tabela 22 – Valores de p obtidos comparando os resultados da comunicação acumulada.

	Teste t de Student			Teste de Wilcoxon		
	Função (4.1) x Função (4.2)	Função (4.1) x Função (4.3)	Função (4.2) x Função (4.3)	Função (4.1) x Função (4.2)	Função (4.1) x Função (4.3)	Função (4.2) x Função (4.3)
KLT	3,08E-13	6,89E-09	1,87E-13	1,86E-09	2,05E-07	1,86E-09
Sparse 1.3a	2,24E-15	5,58E-04	2,20E-16	1,86E-09	9,52E-04	1,86E-09
Barnes	7,30E-10	1,41E-04	6,16E-09	4,46E-06	2,72E-04	6,04E-06
FMM	3,19E-03	2,20E-16	2,20E-16	3,94E-03	1,78E-06	1,78E-06
Ocean	2,20E-16	1,72E-14	2,20E-16	1,80E-06	1,80E-06	1,80E-06

Tabela 23 – Valores de p obtidos comparando os resultados da comunicação distribuída através do tempo.

	Teste t de Student			Teste de Wilcoxon		
	Função (4.1) x Função (4.2)	Função (4.1) x Função (4.3)	Função (4.2) x Função (4.3)	Função (4.1) x Função (4.2)	Função (4.1) x Função (4.3)	Função (4.2) x Função (4.3)
KLT	3,42E-08	5,97E-07	6,79E-11	5,48E-06	6,91E-07	1,86E-09
Sparse 1.3a	9,37E-14	2,49E-03	2,20E-16	1,86E-09	9,52E-03	1,82E-06
Barnes	1,25E-03	8,19E-03	3,01E-04	6,31E-04	1,99E-03	1,38E-05
FMM	4,40E-03	2,20E-16	2,20E-16	2,69E-03	1,80E-06	1,80E-06
Ocean	2,20E-16	1,19E-12	2,20E-16	1,77E-06	2,34E-06	1,79E-06

Tabela 24 – Valores de p obtidos comparando os resultados da comunicação acumulada com os resultados da comunicação distribuída através do tempo.

	Teste t de Student			Teste de Wilcoxon		
	Função (4.1)	Função (4.2)	Função (4.3)	Função (4.1)	Função (4.2)	Função (4.3)
KLT	4,04E-05	1,44E-03	3,45E-04	2,54E-04	2,51E-03	1,66E-03
Sparse 1.3a	2,28E-08	1,06E-11	7,76E-11	5,72E-07	1,82E-06	1,82E-06
Barnes	9,00E-08	4,85E-06	7,18E-05	1,02E-05	6,43E-05	3,19E-04
FMM	5,94E-09	1,49E-03	1,08E-04	7,86E-06	1,52E-03	5,12E-04
Ocean	2,83E-06	2,72E-06	1,10E-07	7,73E-05	7,63E-05	7,57E-06

CÓDIGO FONTE DO *FRAMEWORK* DESENVOLVIDO

```
1: #Paths que serao utilizados durante o projeto
2: mainDir <- 0 #path para o local desejado para criar a pasta
   que armazenara a entrada do PET e saidas
3: pathPet <- 0 #path contendo o caminho para o executavel do PET
4: pathMcprof <- 0 #path contendo o caminho para o executavel do
   MCProf
5: comApl <- 0 #comando para executar a aplicacao
6:
7: #Variaveis Globais
8: nFunctions <- 0      #armazena o numero de funcoes existentes
   na aplicacao
9: nClusters <- 0      #armazena o numero de grupos
10: Matrices <- list()  #lista que armazena as matrizes
11: MatrixTp <- list()  #lista que vai armazenar a acumulacao das
   matrizes de tempo
12: MatrixAc <- list()  #lista que vai armazenar a acumulacao das
   matrizes
13: matrixSum <- 0      #ira armazenar a soma da comunicacao
   total de todas as matrizes
14: vetClust <- list()  #lista contendo os agrupamentos
   encontradas
15: vetAcum <- list()   #lista contendo o agrupamento encontrada
   da matriz acumulada
16: resultClust <- list()#vetor que ira armazenar o resultado da
   comparacao entre grupos
```

```
17: timeF <- 0          #ira armazenar a funcao passada pelo
    usuario
18: maxLimit <- 0        #ira armazenar o valor da largura de
    banda (valor constante, maximo)
19: minLimit <- 0        #ira armazenar o valor da largura de
    banda (valor constante, minimo)
20: vetMat <- list()     #ira armazenar o tamanho de cada matriz
    lida
21: ResultadoAc <- 0     #armazena o agrupamento acumulado
22: JanTime <- 0         #armazena a janela de tempo
23:
24: #Funcao que le o arquivo de configuracao que contem os
    parametros iniciais
25: readParam <- function(){
26:   input <- readLines("/home/rafael/Documentos/Projeto/Input.txt
    ") #Le o arquivo de configuracao
27:   JanTime <- input[1]
28:   nFunctions <- input[2]
29:   nClusters <- input[3]
30:   maxLimit <- input[4]
31:   minLimit <- input[5]
32:   timeF <- input[6]
33:
34:   input <- readLines("/home/rafael/Documentos/Projeto/Paths.txt
    ")
35:   pathMcprof <- input[1]
36:   pathPet <- input[2]
37:   mainDir <- input[3]
38:   comApl <- input[4]
39:
40:   MatrixAc <- matrix(0, nrow = as.numeric(nFunctions), ncol =
    as.numeric(nFunctions))      #A matriz acumulada eh
    inicializada com zero
41:   MatrixTp <- matrix(0, nrow = as.numeric(nFunctions), ncol =
    as.numeric(nFunctions))      #A matriz acumulada eh
    inicializada com zero
42: }
43:
44: #Funcao para chamar a MCProf
45: chamaMCProf <- function(){
46:   command <- paste(c(pathPin,"-t",pathMcprof,"-RecordStack 0 -
    TrackObjects 1 -Engine 4 -TrackTime",JanTime,"--",comApl),
```

```

        collapse = " ")
47:   system(command, wait = TRUE)
48:
49: }
50:
51: #Funcao que le o arquivo com matrizes e as armazena
52: readMatrix <- function(i){
53:   arqName <- paste(c("/home/rafael/Documentos/Projeto/Matrix/
        Matrix_",i,".txt"), collapse = "")
54:   mat <- readLines(arqName)
55:   Matrices <- read.table(textConnection(gsub("\t"," ", mat)))
56:   vetMat[i] <- nrow(Matrices)
57:
58:   #se a matriz for menor que o numero de funcoes, adiciono
        linhas e colunas ate chegar no numero de funcoes
59:   if(nrow(Matrices) < as.numeric(nFunctions)){
60:     while(nrow(Matrices) < as.numeric(nFunctions)){
61:       vetRow <- matrix(0, ncol = ncol(Matrices), nrow = 1)
62:       Matrices <- rbind(Matrices, vetRow)
63:     }
64:     while(ncol(Matrices) < as.numeric(nFunctions)){
65:       vetCol <- matrix(0, ncol = 1, nrow = nrow(Matrices))
66:       Matrices <- cbind(Matrices, vetCol)
67:     }
68:   }
69:
70:   MatrixAc <- Reduce('+',list(MatrixAc, Matrices)) #somo as
        matrixes de tempo para gerar a acumulada de tempo
71:
72: }
73:
74: #Funcao que divide o volume de dados pela largura de banda para
        obter a matriz de tempo
75: timeMatrix <- function(i){
76:   auxFunc <- function(x) eval(parse(text = timeF))
77:   aux1 <- do.call(auxFunc, list(i))
78:   result <- abs(aux1)
79:
80:   if(result > as.numeric(maxLimit)){
81:     Matrices <- Matrices / as.numeric(maxLimit)
82:   }else if(result < as.numeric(minLimit)){
83:     Matrices <- Matrices / as.numeric(minLimit)

```

```
84:   }else{
85:       Matrices <- Matrices / result
86:   }
87:
88:   #soma as matrixes de tempo para gerar a acumulada de tempo
89:   MatrixTp <- Reduce('+',list(MatrixTp, Matrices))
90: }
91:
92: #Funcao que ira realizar o agrupamento
93: particiona <- function(i){
94:   gc()
95:   #cria o diretorio
96:   dir.create(file.path(mainDir))
97:   #verifica se o diretorio ja existe, se sim aparece um
   warning
98:   setwd(file.path(mainDir))
99:
100:  #path para um pre-arquivo de entrada do pet
101:  config <- readLines("/home/rafael/Documentos/Projeto/init.txt
   ")
102:
103:   filePet <- file(textPath,"w")          #cria o arquivo para
   escrita
104:   writeLines(paste(config), filePet)
105:   aux <- Matrices
106:
107:   write.table(aux, file=filePet, row.names=FALSE, col.names=
   FALSE, sep="\t")
108:
109:   close(filePet)          #fecha o arquivo
110:
111:   execMode <- 2
112:   command <- paste(c(pathPet,execMode,textPath,nClusters,">",
   saidaPath), collapse = " ")
113:
114:   system(command, wait = TRUE)
115:
116:   #copia o agrupamento encontrado para outro arquivo
117:   arqName <- paste(c(mainDir,"/saida_mat_",i,".txt"),
   collapse = "")
118:   fileRes <- file(arqName,"w")
119:   config2 <- readLines(saidaPath)
```



```
120:     writeLines(paste(config2), fileRes)
121:
122:     close(fileRes)
123: }
124:
125: particionaAc <- function(){
126:     gc()
127:     config <- readLines("/home/rafael/Documentos/Projeto/init.
        txt")
128:
129:     filePet <- file(textPath,"w")          #cria o arquivo para
        escrita
130:     writeLines(paste(config), filePet)
131:     aux <- MatrixAc
132:
133:     write.table(aux, file=filePet, row.names=FALSE, col.names=
        FALSE, sep="\t")
134:
135:     close(filePet)          #fecha o arquivo
136:
137:     execMode <- 2
138:     command <- paste(c(pathPet,execMode,textPath,nClusters,">",
        saidaPath), collapse = " ")
139:
140:     system(command, wait = TRUE)
141:
142:     #copia o agrupamento encontrado para outro arquivo
143:     arqName <- paste(c(mainDir,"/saida_acumulada.txt"),
        collapse = "")
144:     fileRes <- file(arqName,"w")
145:     config2 <- readLines(saidaPath)
146:     writeLines(paste(config2), fileRes)
147:
148:     close(fileRes)
149:
150: }
151:
152: #Funcao que ira comparar o agrupamento
153: compara <- function(){
154:     aux <- vector()
155:     k <- 1
156:     l <- 1
```

```
157:     index <- 1#auxiliar para o vetor de custos
158:     totalClust <- as.numeric(nMatrices) * as.numeric(nClusters)
159:     tamAux <- list() #ira armazenar o tamanho de cada elemento
na lista
160:     tamAux2 <- list()
161:     auxVet <- list()
162:     #ira armazenar um vetor auxiliar para completar as linhas
da matrix lida
163:     vetRow <- list()
164:     #ira armazenar um vetor auxiliar para completar as colunas
da matrix lida
165:     vetCol <- list()
166:     nclust <- as.numeric(nClusters)
167:     cont <- 0    #usado para auxiliar a leitura do grupo
168:     z <- 0      #ira armazenar o maior valor do tamanho de cada
linha na lista
169:     sumCl <- 0  #usado para armazenar a soma do custo de
execucao nos grupos
170:
171:     matrixSum <-<- sum(MatrixTp) #somo a comunicaÃ  ã o  inteira
da matrix acumulada
172:
173: #Calcula o agrupamento da matriz acumulada
174: arqName <- paste(c(mainDir, "/saida_acumulada.txt"), collapse
= "")
175:     clust <- readLines(arqName)
176:     #for comeca em 16 porque eh a linha do arquivo que contem o
inicio
177:     #do agrupamento encontrado pelo PET utilizando a Busca
Evolutiva
178:     for(j in 16:(15 + as.numeric(nClusters))){
179:         vetAcum[1] <-<- clust[j]
180:         l <- l + 1
181:     }
182:     for(i in 1:nclust){    #salvo todas os agrupamentos
encontrados em um vetor
183:         aux <- na.omit(as.numeric(unlist(strsplit(unlist(vetAcum[i
]), "[^0-9]+"))))
184:         tamAux2[i] <- length(aux)
185:         vetAcum[i] <-<- read.table(textConnection(gsub(" ", " ", aux
)))
186:     }
```

```

187:  z <- max(unlist(tamAux2))
188:  #completo cada linha da lista com 0 e depois converto a lista
      para matriz
189:  for(i in 1:nclust){
190:    aux <- length(unlist(vetAcum[i]))
191:    auxVet <- unlist(vetAcum[i])
192:    if(aux < z){
193:      for(j in (aux + 1):z){
194:        auxVet[j] <- 0
195:      }
196:    }
197:    vetAcum[i] <<- read.table(textConnection(gsub(" ", " ", (as
      .numeric(unlist(auxVet))))))
198:  }
199:  auxMat <- data.frame(matrix(unlist(vetAcum), ncol = z, byrow
      = T), stringsAsFactors = FALSE)
200:  vetAcum <<- auxMat
201:  sumCl <- 0
202:  for(j in 1:nclust){ #Acumulado
203:    #pega quantas funcoes foram colocadas em cada grupo, mas le
      um grupo por vez
204:    nf <- as.numeric(vetAcum[j,2])
205:    #pego o primeiro elemento do grupo encontrado
206:    for(x in 3:(nf + 2)){
207:      num1 <- as.numeric(vetAcum[j,x])
208:      #pego a comunicacao entre todos elementos do grupo
209:      for(y in 3:(nf + 2)){
210:        num2 <- as.numeric(vetAcum[j,y])
211:        sumCl <- sumCl + MatrixTp[(num1+1),(num2+1)]
212:      }
213:    }
214:  }
215:  ResultadoAc <<- abs(matrixSum - sumCl)
216:
217:  #Calcula o agrupamento das matrizes distribuidas
218:  for(i in 1:as.numeric(nMatrices)){
219:    arqName <- paste(c(mainDir,"/saida_mat_",i,".txt"),
      collapse = "")
220:    clust <- readLines(arqName)
221:    k <- 1
222:    for(w in 16:(15 + nclust)){
223:      vetClust[k] <<- clust[w]

```

```

224:         k <- k + 1
225:     }
226:     for(n in 1:nclust){
227:         aux <- na.omit(as.numeric(unlist(strsplit(unlist(vetClust
[n]), "[^0-9]+"))))
228:         tamAux[n] <- length(aux)
229:         vetClust[n] <- read.table(textConnection(gsub(" ", " ",
aux)))
230:     }
231:     z <- max(unlist(tamAux))
232:     for(r in 1:nclust){
233:         aux <- length(unlist(vetClust[r]))
234:         auxVet <- unlist(vetClust[r])
235:         if(aux < z){
236:             for(m in (aux + 1):z){
237:                 auxVet[m] <- 0
238:             }
239:         }
240:         vetClust[r] <- read.table(textConnection(gsub(" ", " ",
(as.numeric(unlist(auxVet))))))
241:     }
242:
243:     auxMat <- data.frame(matrix(unlist(vetClust), ncol = z,
byrow = T), stringsAsFactors = FALSE)
244:     vetClust <- auxMat
245:     sumCl <- 0
246:     for(j in 1:nclust){
247:         nf <- as.numeric(vetClust[j,2])
248:         for(x in 3:(nf + 2)){
249:             num1 <- as.numeric(vetClust[j,x])
250:             for(y in 3:(nf + 2)){
251:                 num2 <- as.numeric(vetClust[j,y])
252:                 sumCl <- sumCl + MatrixTp[(num1+1),(num2+1)]
253:             }
254:         }
255:     }
256:     resultClust[i] <- abs(matrixSum - sumCl)
257:     vetClust <- list()
258: }
259:
260: result <- which.min(unlist(resultClust))
261:

```

```
262:  print(paste0("Menor custo de agrupamento utilizando o grupo
      encontrado no arquivo saida_mat_",result,".txt"))
263:  print(paste0("Valor do agrupamento: ", resultClust[result]))
264:  print("Custo de agrupamento utilizando o grupo da matriz
      acumulada")
265:  print(paste0("Valor do agrupamento: ", ResultadoAc))
266:
267:  write.table(MatrixTp, file = "/home/rafael/Documentos/Projeto
      /Matrix_Tempo_Acumulada.txt", sep = "\t", row.names = FALSE,
      col.names = FALSE)
268:  write.table(MatrixAc, file = "/home/rafael/Documentos/Projeto
      /Matrix_Acumulada.txt", sep = "\t", row.names = FALSE, col.
      names = FALSE)
269: }
270:
271: #Funcao main
272: print("Lendo parametros de configuracao.")
273: readParam()
274: print("Obtendo as matrizes de comunicacao")
275: chamaMCProf()
276: print("Lendo matrizes de comunicacao.")
277: print("Criando as matrizes com o tempo de comunicacao.")
278: print("Agrupando as matrizes.")
279: for(i in 1:as.numeric(nMatrices)){
280:   readMatrix(i)
281:   timeMatrix(i)
282:   particiona(i)
283:   gc()
284: }
285: print("Agrupando a matriz acumulada.")
286: particionaAc()
287: print("Comparando os agrupamentos encontrados.")
288: compara()
```

CÓDIGO FONTE DO MÓDULO DE EXTENSÃO DESENVOLVIDO

Funções implementadas no arquivo "commatrix.cpp"

```
1: void Matrix2D::PrintMatrixTime(ostream &fout, double time){
2:     IDNoType TotalSymbols = symTable.TotalSymbolCount();
3:     CHECK_LT(TotalSymbols, Matrix.size());
4:
5:     u16 StartID;
6:     if(ShowUnknown)
7:         StartID = 0;//usar se quiser analisar funções
        desconhecidas
8:     else
9:         StartID = 1;//usar se não quiser analisar funções
        desconhecidas
10:
11:     for (IDNoType p=StartID; p<TotalSymbols; p++){
12:         if(IsFilledRow(p)){
13:             for(IDNoType c=StartID; c<TotalSymbols; c++){
14:                 if( IsFilledCol(c) ){
15:                     fout<< Matrix[p][c] << ALIGNMENT;
16:                 }
17:             }
18:             fout<<endl;
19:         }
20:     }
21:     fout<<endl;
22: }
```

```

23:
24: void Matrix2D::SetZero(){
25:     IDNoType TotalSymbols = symTable.TotalSymbolCount();
26:     CHECK_LT(TotalSymbols, Matrix.size());
27:
28:     u16 StartID;
29:     if(ShowUnknown)
30:         StartID = 0;
31:     else
32:         StartID = 1;
33:
34:     for (IDNoType p=StartID; p<TotalSymbols; p++){
35:         if(IsFilledRow(p)){
36:             for(IDNoType c=StartID; c<TotalSymbols; c++){
37:                 if( IsFilledCol(c) ){
38:                     Matrix[p][c] = 0;
39:                 }
40:             }
41:         }
42:     }
43: }

```

Função implementada no arquivo "pintrace.cpp"

```

1: if(KnobEngine.Value() == 4){
2:     stop_time = clock();
3:     total_time = (stop_time-start_time)/double(CLOCKS_PER_SEC)
        *1000;
4:
5:     if(total_time > KnobTime.Value()){
6:         result.open("Matrix_"+std::to_string(i)+".txt", std::
            ios_base::out);
7:         ComMatrix.PrintMatrixTime(result,0);
8:         result.close();
9:         ComMatrix.SetZero();
10:        i++;
11:        start_time = clock();
12:    }
13: }

```