

**RESOLUÇÃO PARALELA
VERIFICADA DE SISTEMAS DE
EQUAÇÕES LINEARES: UMA
ABORDAGEM PARA
EFICIÊNCIA ENERGÉTICA
UTILIZANDO DVFS**

VIVIANE LINCK LARA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Luiz Gustavo Leão Fernandes
Coorientadora: Prof^a. Mariana Luderitz Kolberg

Dados Internacionais de Catalogação na Publicação (CIP)

L318r Lara, Viviane Linck

Resolução paralela verificada de sistemas de equações lineares :
uma abordagem para eficiência energética utilizando DVFS / Viviane
Linck Lara. – Porto Alegre, 2013.

92 p.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Luiz Gustavo Leão Fernandes.
Coorientadora: Profa. Mariana Luderitz Kolberg.

1. Informática. 2. Processamento de Alto Desempenho. 3. Energia
Elétrica - Conservação. I. Fernandes, Luiz Gustavo Leão. II. Kolberg,
Mariana Luderitz. III. Título.

CDD 004.22

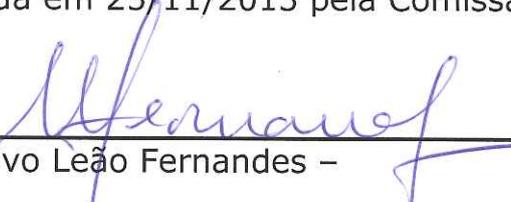
**Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS**



Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

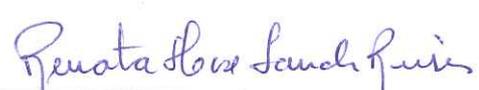
TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "Resolução Paralela Verificada de Sistemas de Equações Lineares: Uma Abordagem para Eficiência Energética Utilizando DVFS" apresentada por Viviane Linck Lara como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Processamento Paralelo e Distribuído, aprovada em 25/11/2013 pela Comissão Examinadora:


Prof. Dr. Luiz Gustavo Leão Fernandes - PPGCC/PUCRS
Orientador


Prof. Dr. Paulo Henrique Lemelle Fernandes - PPGCC/PUCRS


Profa. Dra. Mariana Luderitz Kolberg - UFRGS
Coorientadora


Profa. Dra. Renata Hax Sander Reiser - UFPEL

Homologada em 18/12/2015, conforme Ata No. 023 pela Comissão Coordenadora.


Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.

PUCRS

Campus Central

Av. Ipiranga, 6681 - P32- sala 507 - CEP: 90619-900

Fone: (51) 3320-3611 - Fax (51) 3320-3621

E-mail: ppgcc@pucrs.br

www.pucrs.br/facin/pos

DEDICATÓRIA

Dedico este trabalho à minha família, principalmente aos meus pais e avós, os quais sempre me apoiaram, me consolaram e foram a minha base em momentos difíceis.

“Se você encontrar um caminho sem obstáculos,
ele provavelmente não leva a lugar nenhum.”
(Frank A. Clark)

AGRADECIMENTOS

Agradeço, principalmente, à minha mãe Sônia Marisa Linck Lara e ao meu pai José Ivan Marques Lara que deram todo suporte que precisei durante estes anos de dedicação ao Mestrado, inclusive, com conselhos, carinho, amor e tudo que uma filha mimada precisa. Eles sempre fizeram tudo e estiveram presentes e isso foi extremamente fundamental para mim.

À minha avó materna Honorina Ruth Meira Linck que sempre cuidou de mim, me protegeu, sempre rezou, foi paciente comigo, fez ótimas comidinhas para mim e torceu por mim.

Ao meu avô materno Sady Antonio Linck que não esteve presente fisicamente durante estes anos, mas que me ajudou espiritualmente.

Ao meu avô paterno Ivon Lara e à minha avó paterna Jane Marques Lara que sempre me apoiaram, torceram por mim e me animaram nas festinhas da família.

Aos meus padrinhos e madrinhas que sempre torceram por mim e me ajudaram.

Ao Tio Juca que me ofereceu abrigo no início do Mestrado e a sua família que me recebeu muito bem e me acolheu.

Ao meu namorado Lucas H., que sempre me apoiou em momentos de dificuldade e tornou meus dias mais felizes.

Aos meus grandes amigos de Carazinho: Giane, Cássio, Gabriel, Mateus, Juliana, Ana, Jéssica e Aline. Que foram sempre queridos e parceiros comigo e me fizeram esquecer o *stress* da dissertação com ótimos momentos.

Aos meus grandes amigos de Passo Fundo: Alessandra, Maria Luiza, Rafael, Alisson, Paloma, Mônica e Aline. Que antes mesmo de eu iniciar o Mestrado, já me ajudaram em momentos importantes com apoio, força e amizade.

Aos amigos do mestrado: Eli, Claiton, Lucas O., Bernardo, Eduardo e Daiane. Estes me aturaram quando eu estava de mau humor, me apoiaram e foram amigos queridos durante esse tempo sofrido. Especialmente a Luciana Espindola, que foi extremamente paciente comigo e com minhas dúvidas em física.

Aos meus orientadores Luiz Gustavo Leão Fernandes e Mariana Kolberg que foram extremamente pacientes comigo e me auxiliaram quando precisei.

Aos meus colegas do grupo GMAP, que insistiram veementemente para eu mencionar seus nomes: Andriele, Mateus, Dalvan, Silvana, Victoria e Eduardo. Eles me ajudaram muito e foram capazes de aguentar as bobagens que eu falei e fiz durante o tempo que passei com eles. Especialmente a Andriele que me ajudou em inúmeros momentos de dúvidas, leu meu texto inúmeras vezes e teve paciência em me aconselhar e conversar comigo.

Agradeço ao Ricardo que elaborou este modelo em Latex e foi prestativo e paciente com as correções que pedíamos.

Agradeço aos meus professores do Mestrado e da Graduação da UPF, sem eles eu nunca teria conseguido chegar ao Mestrado.

É incrível este momento de agradecimento, porque lembro de tantas pessoas que ajudaram, ou não, em todo caminho que tracei, no entanto, sei que todos os momentos bons e ruins que passei fizeram eu me tornar uma pessoa mais esforçada na tentativa de ser uma pessoa sempre melhor.

Agradeço ao Grêmio, pois as partidas que fui e xinguei muito o juiz me ajudaram a manter a calma fora do estádio Olímpico Monumental e Arena Monumental.

Certamente esquecerei de alguém então gostaria de agradecer a todos que me auxiliaram durante esta jornada: Muito Obrigado.

RESOLUÇÃO PARALELA VERIFICADA DE SISTEMAS DE EQUAÇÕES LINEARES: UMA ABORDAGEM PARA EFICIÊNCIA ENERGÉTICA UTILIZANDO DVFS

RESUMO

A resolução de Sistemas de Equações Lineares Algébricas (SELAs) é importante em diversos domínios do conhecimento. Em muitos casos, o uso de Computação Verificada é necessário para garantir que os resultados sejam confiáveis. Com o auxílio da Computação de Alto Desempenho, a resolução mais eficiente de SELAs de grande porte com o uso da Computação Verificada tornou-se possível. Atualmente, a área de Alto Desempenho tem buscado soluções que considerem, além do desempenho, a eficiência energética. Nesse sentido, o objetivo do trabalho é utilizar a técnica DVFS (*Dynamic Voltage and Frequency Scaling*) para modificar a frequência do processador na execução de um *solver* de SELAs de Alto Desempenho com verificação do resultado. Além disso, realizar um estudo de caso que permita avaliar se o uso de DVFS reduz o consumo de energia, bem como avaliar de que maneira o desempenho e a exatidão podem ser comprometidos. Inicialmente, foi realizado um estudo de caso sobre o *solver* FastPILSS, analisando exatidão, desempenho e consumo de energia. Depois disso, verificou-se que a utilização de DVFS não afetou a exatidão. Com a análise dos resultados, observou-se que não houve redução do consumo de energia ao utilizar o governador em *powersave* se comparado ao consumo de energia com o governador em *performance*. Esse comportamento pode ser atribuído ao significativo aumento no tempo de execução. Ao realizar a alteração de frequência em pontos isolados no algoritmo do *solver*, observou-se que tendo como entrada matrizes do tipo densas durante a realização do cálculo da inversa aproximada, obtém-se redução de no máximo 3,29% no consumo de energia.

Palavras-Chave: DVFS, Sistemas de Equações Lineares, Alto Desempenho, FastPILSS.

PARALLEL VERIFIED RESOLUTION SYSTEMS OF LINEAR EQUATIONS: AN APPROACH TO ENERGY EFFICIENCY USING DVFS

ABSTRACT

Solving Systems of Linear Equations is important in several domains. In many cases, it is necessary to employ verified computing to achieve reliable results. With the support of High Performance Computing (HPC), solve efficiently huge linear systems with Verified Computing has become possible. Recently, HPC researchers have started to investigate solutions focused not only in performance but also in energy efficiency as well. In this context, the main goal of this work is to propose the use of DVFS (Dynamic Voltage and Frequency Scaling) technique to change the CPU frequency during the execution of a solver that employs Verified Computing. Furthermore, this work intends to present a case study aiming at verifying if the use of DVFS can provide a reduction on energy consumption without performance and accuracy being compromised. Initially, a study about the FastPILSS solver was carried out to evaluate its accuracy, performance and energy consumption over several different input matrices. After that, we observed that the use of DVFS does not affect accuracy. Analysing the results, no reduction in energy consumption using the *powersave* governor was observed if compared to the energy consumption using the *performance* governor. This occurs due to the significant increase in execution time. When the frequency was changed in isolated steps of the solver algorithm, it was possible to reduce up to 3,29% the energy consumption for dense matrices during the approximate inverse calculation.

Keywords: DVFS, Systems of Linear Equations, High Performance, FastPILSS.

LISTA DE FIGURAS

Figura 3.1 – Requisitos para a obtenção da Computação Verificada. Baseado em: [9], [24]	34
Figura 3.2 – Variável de alta precisão. Fonte: [31]	36
Figura 4.1 – Comparação entre o tempo de execução do solver para a matriz de entrada Gen1.	44
Figura 4.2 – Comparação entre o tempo de execução do solver para a matriz de entrada DingDong.	45
Figura 4.3 – Comparação entre o tempo de execução do solver para as matrizes reais de entrada Fidapm29 e Fidap019.	45
Figura 6.1 – Localização dos valores não nulos das matrizes	56
Figura 6.2 – Comparação entre consumo e tempo para a matriz de entrada Gen1.	60
Figura 6.3 – Comparação entre consumo e tempo para a matriz de entrada DingDong.	61
Figura 6.4 – Comparação entre consumo e tempo para a matriz de entrada Fidapm29	61
Figura 6.5 – Comparação entre consumo e tempo para a matriz de entrada Fidap019	62
Figura 6.6 – Comparação entre consumo e tempo para a matriz de entrada Bcsstk17	62
Figura 6.7 – Comparação entre consumo e tempo para a matriz de entrada Fidapm37	63
Figura 6.8 – Comparação entre consumo e tempo para a matriz de entrada Dw8192	63
Figura 6.9 – Comparação entre consumo e tempo para a matriz de entrada Utm5940	64
Figura 6.10 – Comparação entre as matrizes por tempo e por consumo	65
Figura 6.11 – Comparação entre consumo e tempo para a matriz de entrada Gen1	66
Figura 6.12 – Comparação entre consumo e tempo para a matriz de entrada DingDong	67
Figura 6.13 – Comparação entre consumo e tempo para a matriz de entrada Fidapm29	67
Figura 6.14 – Comparação entre consumo e tempo para a matriz de entrada Fidap019	68
Figura 6.15 – Comparação entre consumo e tempo para a matriz de entrada Bcsstk17	68
Figura 6.16 – Comparação entre consumo e tempo para a matriz de entrada Fidapm37	69
Figura 6.17 – Comparação entre consumo e tempo para a matriz de entrada Dw8192	69
Figura 6.18 – Comparação entre consumo e tempo para a matriz de entrada Utm5940	70
Figura 6.19 – Comparação entre as matrizes por tempo e por consumo	71

LISTA DE TABELAS

Tabela 3.1 – Tipos de Dados do C-XSC	35
Tabela 4.1 – Características das matrizes de entrada.	42
Tabela 4.2 – Tempos de execução dos passos no <i>solver</i> de Zimmer com 8 processos . . .	43
Tabela 4.3 – Tempos de execução dos passos no <i>solver</i> de Zimmer com 24 processos . .	43
Tabela 4.4 – Resultados dos <i>solvers</i> para a matriz Gen1 de ordem 45.000 com 32 processos.	46
Tabela 4.5 – Resultados dos <i>solvers</i> para a matriz DingDong de ordem 45.000 com 32 processos.	46
Tabela 4.6 – Resultados dos <i>solvers</i> para a Matriz Fidap019 com 32 processos	46
Tabela 4.7 – Resultados dos <i>solvers</i> para a Matriz Fidapm29 com 32 processos	47
Tabela 4.8 – Consumo de energia em <i>joules</i> com frequência máxima de 2.40Ghz com 8 processos.	48
Tabela 4.9 – Consumo de energia em <i>joules</i> com frequência máxima de 2.40Ghz com 24 processos.	48
Tabela 6.1 – Características das matrizes de entrada.	56
Tabela 6.2 – Resultados do <i>solver</i> de Zimmer para a Matriz Gen 1 com 4 processos. . . .	57
Tabela 6.3 – Resultados do <i>solver</i> de Zimmer para a Matriz Fidamp29 com 4 processos.	57
Tabela 6.4 – Valores medidos de potência das matrizes	58
Tabela 6.5 – Tempo de execução do <i>solver</i> de Zimmer.	59
Tabela 6.6 – Consumo de energia e tempo de execução do <i>solver</i> de Zimmer	59
Tabela 6.7 – Potência (<i>watts</i>) dos passos em <i>performance</i>	60
Tabela 6.8 – Potência (<i>watts</i>) dos passos em <i>powersave</i>	60
Tabela A.1 – Tempos de execução dos passos no <i>solver</i> de Zimmer com 16 processos . .	83
Tabela A.2 – Tempos de execução dos passos no <i>solver</i> de Zimmer com 32 processos . .	83
Tabela A.3 – Consumo de energia em <i>joules</i> com frequência máxima de 2.40Ghz com 16 processos.	83
Tabela A.4 – Consumo de energia em <i>joules</i> com frequência máxima de 2.40Ghz com 32 processos.	84
Tabela B.1 – Resultados do <i>solver</i> de Zimmer para a Matriz DingDong com 4 processos	85
Tabela B.2 – Resultados do <i>solver</i> de Zimmer para a Matriz Fidap019 com 4 processos .	85
Tabela B.3 – Resultados do <i>solver</i> de Zimmer para a Matriz Bcsstk17 com 4 processos. .	85
Tabela B.4 – Resultados do <i>solver</i> de Zimmer para a Matriz Fidapm37 com 4 processos	86
Tabela B.5 – Resultados do <i>solver</i> de Zimmer para a Matriz Dw8192 com 4 processos . .	86
Tabela B.6 – Resultados do <i>solver</i> de Zimmer para a Matriz Utm5940 com 4 processos .	86

Tabela B.7 – Consumo de energia e tempo de execução reduzindo a frequência no Passo 2	87
Tabela B.8 – Consumo de energia e tempo de execução reduzindo a frequência no Passo 3	87
Tabela B.9 – Consumo de energia e tempo de execução reduzindo a frequência no Passo 4	88
Tabela B.10 – Consumo de energia e tempo de execução reduzindo a frequência no Passo 5	88
Tabela B.11 – Consumo de energia e tempo de execução reduzindo a frequência no Passo 6	89
Tabela B.12 – Consumo de energia e tempo de execução reduzindo a frequência nos Passos 2 e 3	89
Tabela B.13 – Consumo de energia e tempo de execução reduzindo a frequência nos Passos 2 e 5	90
Tabela B.14 – Consumo de energia e tempo de execução reduzindo a frequência nos Passos 2, 3 e 4	90
Tabela B.15 – Consumo de energia e tempo de execução dos <i>solvers</i> reduzindo a frequência nos Passos 3, 4 e 6	91
Tabela B.16 – Consumo de energia e tempo de execução reduzindo a frequência nos Passos 4, 5 e 6	91
Tabela B.17 – Consumo de energia e tempo de execução reduzindo a frequência nos Passos 5 e 6	92

LISTA DE SIGLAS

SELAS – *Sistemas de Equações Lineares Algébricas*

BLAS – *Basic Linear Algebra Subprograms*

LAPACK – *Linear Algebra PACKage*

SCALAPACK – *Scalable Linear Algebra PACKage*

DVFS – *Dynamic Voltage and Frequency Scaling*

API – *Application Programming Interface*

CFD – *Computational Fluid Dynamics*

GPU – *Graphics Processing Unit*

LPMG – *Low-Power MPI_Gather*

LPMS – *Low-Power MPI_Scatter*

PLASMA – *Parallel Linear Algebra for Scalable Multi-core Architectures*

C-XSC – *C for eXtended Scientific Computation*

CPU – *Central Processing Unit*

LAD – *Laboratório de Alto Desempenho*

LISTA DE ALGORITMOS

1	Algoritmo do <i>solver</i> FastPILSS. Adaptado de: [31]	40
---	---	----

SUMÁRIO

1	INTRODUÇÃO	25
1.1	OBJETIVOS	26
1.2	ORGANIZAÇÃO DO TRABALHO	27
2	TRABALHOS RELACIONADOS	29
2.1	MEDIÇÕES	29
2.2	ESTIMATIVAS	30
2.3	CONSIDERAÇÕES	32
3	REFERENCIAL CONCEITUAL	33
3.1	SCALAPACK	33
3.2	COMPUTAÇÃO VERIFICADA	34
3.2.1	C-XSC	35
3.3	EFICIÊNCIA ENERGÉTICA EM ALTO DESEMPENHO	36
3.4	TÉCNICA DVFS	36
4	ANÁLISE PRELIMINAR DO SOLVER FASTPILSS	39
4.1	FASTPILSS	39
4.2	CENÁRIO	41
4.3	DESEMPENHO	42
4.4	EXATIDÃO	46
4.5	ESTIMATIVA DE CONSUMO DE ENERGIA	47
4.6	CONSIDERAÇÕES	49
5	ESTRATÉGIAS PARA ESTUDO DE EFICIÊNCIA ENERGÉTICA EM SELAS VERIFICADOS	51
5.1	DEFINIÇÃO DA FREQUÊNCIA EM UM ÚNICO PONTO	51
5.2	DEFINIÇÃO DA FREQUÊNCIA EM PONTOS ISOLADOS	52
5.3	DEFINIÇÃO DA FREQUÊNCIA EM PONTOS DIVERSOS	53
6	AVALIAÇÃO DAS ESTRATÉGIAS PROPOSTAS	55
6.1	CENÁRIO	55
6.2	EXATIDÃO	57
6.3	CONSUMO DE ENERGIA E DESEMPENHO	58

6.3.1	DEFINIÇÃO DA FREQUÊNCIA EM UM ÚNICO PONTO	58
6.3.2	DEFINIÇÃO DA FREQUÊNCIA EM PONTOS ISOLADOS	59
6.3.3	DEFINIÇÃO DE FREQUÊNCIA EM PONTOS DIVERSOS	66
6.4	CONSIDERAÇÕES	71
7	CONCLUSÃO	73
7.1	TRABALHOS FUTUROS	74
	REFERÊNCIAS	77
	APÊNDICE A – Capítulo 4	83
A.1	DESEMPENHO	83
A.2	ESTIMATIVA DE CONSUMO DE ENERGIA	83
	APÊNDICE B – Capítulo 6	85
B.1	EXATIDÃO	85
B.2	CONSUMO DE ENERGIA E DESEMPENHO	87

1. INTRODUÇÃO

A resolução de SELAs (Sistemas de Equações Lineares Algébricas) é importante em diversas áreas como, por exemplo, no cálculo de redução de ruído da Física ou desenho da asa de um avião da Engenharia [9], [13]. Entretanto, por ser uma tarefa custosa, a resolução de SELAs de grande porte só é viável se usada com Computação de Alto Desempenho [42]. Para isso, existem bibliotecas voltadas à programação numérica de Alto Desempenho, tais como, BLAS (*Basic Linear Algebra Subprograms*) [51], LAPACK (*Linear Algebra PACKage*) [52], ScaLAPACK (*Scalable Linear Algebra PACKage*) [50], entre outras. A utilização destas bibliotecas auxilia no desenvolvimento de aplicações de Alto Desempenho, especialmente aplicações que resolvem SELAs.

Embora existam ferramentas específicas para a resolução de SELAs, normalmente elas não fornecem garantia de que os resultados estejam corretos, pois todos os cálculos no computador são feitos utilizando a aritmética de ponto flutuante, podendo ocorrer erros de representação ou arredondamento. O sistema de representação em ponto flutuante é orientado pelo padrão IEEE 754 [1]. Este padrão define a forma como os números em ponto flutuante devem ser representados em precisão simples e dupla.

Como a representação é definida por uma quantidade finita de números, erros de arredondamento podem acontecer. Estes erros são causados porque algumas vezes o resultado de uma operação não tem representação em ponto flutuante, sendo necessário arredondar o valor para representá-lo. Claudio [9] também comenta que além dos erros de arredondamento existem os erros de truncamento, que acontecem quando se substitui qualquer processo infinito por um processo finito ou discreto.

Estes tipos de erros podem afetar de forma drástica os resultados numéricos. No caso de SELAs, até mesmo sistemas pequenos podem ser afetados por tais erros, e ter como resposta um resultado errado. Por exemplo [28]:

$$A = \begin{pmatrix} 64919121 & -159018721 \\ 41869520.5 & -102558961 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Ao resolver este sistema obtêm-se os seguintes resultados para x_1 e x_2 :

$$x_1 = \frac{a_{22}}{a_{11}a_{22} - a_{12}a_{21}} = 205117922, \quad x_2 = \frac{-a_{21}x_1}{a_{22}} = 83739041.$$

Entretanto, ao resolver este sistema em uma máquina com processador Itanium2, com padrão IEEE de dupla precisão aritmética, o resultado obtido para x_1 e x_2 é:

$$\tilde{x}_1 = 102558961 \quad \tilde{x}_2 = 41869520.5$$

Nesse contexto, a Computação Verificada aparece como uma alternativa para contornar as limitações da representação de ponto flutuante, pois garante o rigor matemático do resultado

de uma computação fornecendo um intervalo que, certamente, contém o resultado exato. Este intervalo é chamado de *enclosure* [28],[32]. Embora a Computação Verificada permita representar os resultados com maior exatidão, seu uso aumenta o tempo de execução se comparada a uma mesma aplicação sem verificação.

Existem alguns trabalhos que apresentam *solvers*¹ para a resolução verificada de Sistemas Lineares com Alto Desempenho [28], [31]. Este trabalho foca, particularmente, no *solver* de Zimmer [31], um *solver* verificado para a resolução de SELAs utilizando a biblioteca C-XSC [4]. A proposta de Zimmer [31] tem como objetivo calcular o resultado verificado de um Sistema Linear com programação paralela. Zimmer optou por utilizar o ambiente fornecido pela biblioteca C-XSC, possibilitando assim a obtenção de resultados verificados.

Atualmente, a Computação de Alto Desempenho tem buscado soluções que considerem, além do desempenho, o consumo de energia durante a execução de aplicações. Devido a esta preocupação com a redução do consumo de energia, a maioria dos processadores atuais permite a utilização da técnica DVFS (*Dynamic Voltage and Frequency Scaling*). Esta técnica visa a alteração da frequência e voltagem do processador, já que a escala adequada da frequência e da voltagem pode reduzir o consumo de energia das máquinas.

Entretanto, calcular o consumo de energia de uma aplicação não é trivial. Além disso, muitos conceitos da área de Computação Verde são recentes e as ferramentas que permitem medir e analisar o consumo de energia em ambientes de Alto Desempenho muitas vezes são proprietárias. Em função disso, o cálculo do consumo de energia das aplicações pode ser realizado através de estimativas baseadas no tempo de execução e no consumo em *Watts* do nodo.

Neste contexto, uma abordagem para buscar a redução do consumo de energia na utilização de *solvers* paralelos verificados para a resolução de Sistemas Lineares se faz necessária. Este trabalho apresenta um estudo que permite avaliar o uso da técnica DVFS em um *solver* de SELAs Verificado de Alto Desempenho, cujo objetivo é reduzir o consumo de energia sem comprometer o desempenho de forma significativa e mantendo a garantia numérica dos resultados.

1.1 Objetivos

O objetivo central deste trabalho é a **realização de um estudo sobre a utilização da Técnica DVFS em um *solver* de Alto Desempenho que soluciona Sistemas de Equações Lineares com Computação Verificada**. Os objetivos secundários deste trabalho são:

- um estudo da possibilidade do uso da técnica de DVFS em um *solver* de SELAs com Computação Verificada com relação à exatidão dos resultados e à manutenção da garantia;
- proposta e avaliação de uma estratégia de redução de energia em SELAs Verificados sem comprometer significativamente o desempenho;

¹*Solver* é uma ferramenta computacional com o objetivo de resolver um problema matemático.

1.2 Organização do Trabalho

O trabalho está organizado da seguinte forma: os trabalhos relacionados são descritos no Capítulo 2, divididos em trabalhos que utilizam medidores para calcular o consumo e trabalhos que fazem estimativas. O Capítulo 3 contextualiza as bibliotecas de Alto Desempenho, Computação Verificada, eficiência energética em Alto Desempenho e apresenta a técnica DVFS. O Capítulo 4 apresenta os SELAs Verificados e um estudo do comportamento do *solver* de Zimmer (FastPILSS) em um *cluster* do Laboratório de Alto Desempenho da PUCRS. O Capítulo 5 apresenta a estratégia proposta para a realização dos experimentos. No Capítulo 6 é introduzido o novo cenário de testes e são apresentados os experimentos de consumo de energia e desempenho do *solver* com DVFS, apresentando uma análise sobre os resultados obtidos. Por fim, o Capítulo 7 apresenta as conclusões e os trabalhos futuros.

2. TRABALHOS RELACIONADOS

Diversos trabalhos analisam o consumo de energia durante a execução de aplicações de acordo com o uso de diferentes algoritmos de escalonamento. Esta seção apresenta alguns destes trabalhos, que estão subdivididos de acordo com o tipo de teste realizado para medir o consumo de energia, isto é, utilizando medidores (Seção 2.1), ou estimativas (Seção 2.2). A Seção 2.3 apresenta as considerações finais do capítulo.

2.1 Medições

Uma série de trabalhos apresenta o cálculo do consumo de energia através de medidores e ferramentas auxiliares. Freeh et al. [19] por exemplo, investigam o consumo de energia e o tempo de execução de aplicações do *benchmark* NAS em um *cluster* de energia escalável. Este tipo de *cluster* permite o escalonamento de energia e assim pode-se economizar energia escalonando a frequência das CPUs para um valor menor. O consumo é medido com multímetros e simulação nos “gargalos” de memória e comunicação. O consumo de energia e o tempo de execução entre diferentes quantidades de nós também é comparado com testes e simulação. Através desse estudo Freeh et al. concluíram que o *cluster* escalável tem potencial para economizar energia diminuindo a frequência do processador.

A técnica DVFS (*Dynamic Voltage and Frequency Scaling*) é uma técnica para reduzir o consumo de energia em microprocessadores [55], [8]. Os trabalhos de Sharma et al. [60], Choi et al. [8], Kimura et al. [27], Ge et al. [20] e Castillo et al. [6] utilizaram esta técnica em seus trabalhos.

Sharma et al. [60] implementaram um algoritmo no kernel do Linux que altera a voltagem e a frequência em servidores *web* com QoS (*Quality of Service*), o objetivo era reduzir o consumo de energia durante o atraso do serviço.

Choi et al. [8] implementaram uma modificação no uso da técnica DVFS que não só considera o tempo ativo, como é considerado em outros trabalhos, mas também considera o tempo da máquina em *stand-by*. Foram realizados experimentos na plataforma BitsyX e informaram o consumo mensurado.

Kimura et al. [27] propõem um novo algoritmo que reduz o consumo de energia de um programa paralelo em um *cluster* de energia escalável utilizando DVFS, sem impactar no desempenho da aplicação. O algoritmo seleciona a voltagem e a frequência apropriada quando a aplicação está em *slack time*, ou seja, há uma espera por sincronização de tarefas. Neste tempo o algoritmo diminui a frequência e a voltagem, o que permite a redução do consumo de energia sem influenciar no desempenho. Foi desenvolvida a ferramenta PowerWatch com um conjunto de medidores para monitorar o consumo de energia.

A maioria das ferramentas que são utilizadas para calcular o consumo de energia utilizam um *hardware* acoplado à máquina. Com base no valor gerado, é realizado o cálculo de eficiência

energética da aplicação. Por exemplo, o *framework* PowerPack [58] contém um conjunto de ferramentas de *hardware* e *software*. Os componentes de *hardware* incluem sensores, medidores, circuitos e dispositivos de aquisição de dados que permitem medição de potência direta e instrumentação. Os componentes de *software* incluem *drivers* para vários medidores e sensores, e em nível de usuário APIs (*Application Programming Interface*) para criação de perfis de controle de energia e sincronização de código [21]. Este *framework* foi utilizado no trabalho de Latief et al. [33] para criar perfis de desempenho e eficiência energética em algoritmos de Álgebra Linear Densa em arquiteturas *multicore* para as bibliotecas PLASMA e LAPACK. Ge et al. [20] também utilizaram este *framework* para desenvolver estratégias de escalonamento com DVFS em *clusters*.

Castillo et al.[6] analisaram o impacto do uso da técnica DVFS em funções da biblioteca ScaLAPACK, além de realizar testes com Computação Híbrida. Foram feitos experimentos com os quatro governadores disponibilizados pelo pacote *CpuFreq* em algumas funções da ScaLAPACK. Com estes experimentos, Castillo concluiu que ao reduzir a frequência em operações que usam intensivamente o processador (*CPU-bounded*), o tempo de execução aumenta e resulta em um consumo maior. Por outro lado, operações que acessam de forma intensiva a memória (*memory-bounded*), conseguem reduzir o consumo de energia ao ter a frequência reduzida durante a operação.

2.2 Estimativas

Calcular o consumo de energia de uma aplicação paralela não é trivial. Por ser um assunto recente, não é simples obter ferramentas voltadas à análise do consumo de aplicações. Desta forma, algumas pesquisas consideram um cenário levando em consideração que todos os processadores estão ativos e consumindo a mesma quantidade de energia continuamente. Para isso fazem uso de simulação, como o trabalho de Mór et al. [46].

Mämmelä et al. [45], através de modelos de consumo de energia baseados no trabalho de Fan et al. [16], desenvolveram um modelo genérico para calcular o consumo de um servidor. Fan et al. [16] apontam os principais contribuidores para consumo de energia de uma máquina: o processador com 37%; placa de interface de rede com 23%; a memória com 17%; a placa-mãe com 12%; o disco rígido com 6%; e, por fim, os *coolers* com a menor porcentagem, 5%. Para cada componente citado há um modelo relacionado e, em cada modelo de consumo são considerados na fórmula vários fatores para cada componente. O modelo genérico de Mämmelä et al. [45] calcula o consumo total do servidor somando o consumo de energia de todos os modelos anteriores, multiplicado pelo tempo.

O trabalho de Anzt et al. [2] avalia a possibilidade do uso de algoritmos de precisão mista em diferentes plataformas de *hardware*. Os autores realizaram um estudo de caso com dinâmica dos fluidos computacional (C.F.D., do inglês *Computational Fluid Dynamics*) e analisaram a eficiência energética de dois tipos de *clusters*, um com nós comuns e híbridos e um *cluster* com placa aceleradora GPU (*Graphics Processing Unit*). Para estimar os valores de consumo, utilizou-se os valores

de consumo de um nodo considerando que a sua utilização foi completa, e multiplicou-se o valor pelo tempo de execução em segundos.

A métrica *FLOPS/WATT* foi desenvolvida para medir a eficiência energética de uma aplicação. Com a finalidade de aperfeiçoar esse tipo de medida, o trabalho de Bekas et al. [3] propõe uma nova métrica de desempenho que leva em consideração a redução do consumo de energia e a minimização do tempo para encontrar a solução. Assim, foi proposta a FTTSE ($f(\text{Time To Solution}) \cdot \text{energy}$) como métrica de desempenho, onde o consumo de energia é multiplicado pela função $f(T)$, que depende do tempo total da execução (*time to solution*) T . Esta função $f(T)$ pode ser uma simples função linear ou até uma função exponencial, dependendo do tipo de medida para a função definida.

Assim como há trabalhos que medem o consumo de energia, há trabalhos que utilizam a técnica DVFS com estimativas do valor de consumo de energia. Os trabalhos de Dong et al. [12], Chen et al. [7], Etinski et al. [15] e Ding et al.[11] utilizaram esta técnica.

Segundo Dong et al.[12], as operações coletivas de comunicação MPI ocupam a maior parte do tempo, e reduzir a frequência e voltagem nessas operações que não são críticas pode, efetivamente, reduzir o consumo de energia. Com base nisso, o trabalho de Dong et al. [12] propõe os algoritmos LPMG (*Low-Power MPI_Gather*), LPMS (*Low-Power MPI_Scatter*) e também pretende estendê-lo para todas as operações coletivas da biblioteca MPI. Para realizar os experimentos e analisar os resultados, o Intel MPI *Benchmark* foi utilizado com inserções de instruções de escalonamento de voltagem. Os resultados experimentais mostram que estes algoritmos diminuem o consumo de energia.

Etinski et al. [15] analisam quais características são necessárias para o sucesso na relação entre desempenho e energia em aplicações paralelas. Este trabalho apresenta um modelo que fornece o limite superior de perda de desempenho obtido através da alteração da frequência e voltagem com a técnica DVFS. Etinski et al. indicam que a quantidade de economia de energia depende das características do *cluster*.

Chen et al. [7] desenvolveram um trabalho com objetivo de reduzir o consumo de energia alterando a voltagem e a frequência dos processadores que não estão em momentos críticos. O trabalho mostrou que o uso desta estratégia reduziu o consumo de energia, que dependendo da aplicação foi de 5% a 30% de redução.

O trabalho de Ding et al. [11] propõe modelos analíticos para avaliar a escalabilidade de energia e eficiência energética. Estes modelos servem também para compreender as tendências de consumo de aplicações com grandes quantidades de dados em execução em um grande número de processadores. Os resultados deste estudo mostraram que a execução de uma mesma aplicação em CPUs com menor frequência pode reduzir 40% o consumo de energia dos casos de estudo analisados.

2.3 Considerações

Analisando os trabalhos apresentados, percebe-se que independente da forma de medição (uso de estimativas ou medidores), há redução no consumo de energia com o uso da técnica DVFS. Nota-se que o uso desta técnica está se tornando relevante para a comunidade científica e, com isso, novos estudos sobre a aplicação da escala de frequência e voltagem são importantes.

Dentre os trabalhos pesquisados, muitos abordam o consumo de energia em aplicações de Alto Desempenho, porém não foram encontrados trabalhos que utilizavam a técnica de DVFS para a redução no consumo de aplicações que utilizem algoritmos verificados. O trabalho de Castillo et al. [6] realizou um estudo interessante do uso da técnica DVFS para as funções *pdgemm* e *pdgemv* da ScaLAPACK, realizando comparações entre os governadores disponibilizados pelo pacote *cpufrequtils* para o consumo de energia e desempenho. O trabalho de Castillo et al. [6] é de especial interesse para esta dissertação, pois a função *pdgemm* é utilizada em alguns passos do *solver* de Zimmer [31], descrito no Capítulo 4.

A análise dos trabalhos relacionados indica que uma forma de alterar a frequência do processador é utilizando o pacote *cpufrequtils*, como nos trabalhos de Ge [20], Dong et al. [12], Kimura et al. [27] e Castillo et al.[6]. Visto que dentre os trabalhos relacionados não foram encontrados estudos que focam no uso da técnica DVFS em *solvers* verificados voltados para a resolução de Sistemas de Equações Lineares, é interessante realizar um estudo da técnica DVFS neste contexto.

3. REFERENCIAL CONCEITUAL

Este capítulo apresenta alguns conceitos importantes para o desenvolvimento deste trabalho. A Seção 3.1 apresenta a biblioteca ScaLAPACK, utilizada no desenvolvimento do *solver* descrito no Capítulo 4. A Seção 3.2 descreve a Computação Verificada e a biblioteca de alta exatidão C-XSC. A Seção 3.3 aborda eficiência energética em Alto Desempenho. Por fim, a Seção 3.4 trata da técnica DVFS e os pacotes que possibilitam seu uso.

3.1 ScaLAPACK

Existe um conjunto de bibliotecas para Álgebra Linear com objetivo de obter alto desempenho. Estas bibliotecas contêm rotinas de Álgebra Linear para diferentes arquiteturas de computadores. O uso destas bibliotecas possibilita a resolução de Sistemas Lineares Densos e do tipo Banda [50].

Estas bibliotecas são baseadas na BLAS (*Basic Linear Algebra Subprograms*) [51], que contém rotinas que auxiliam na realização de operações básicas de matrizes e vetores. O nível 1 da BLAS trabalha com operações de escalares e vetores, o nível 2 opera com operações entre matrizes e vetores e o nível 3 trabalha com operações entre matrizes.

Além da BLAS, há a BLACS (*Basic Linear Algebra Communication Subprograms*) [47], uma interface de Álgebra Linear orientada ao paradigma de troca de mensagens com intuito de ser implementada de forma eficiente e uniforme em uma ampla gama de plataformas de memória distribuída. O pacote BLACS existe para fazer aplicações de Álgebra Linear tanto mais fáceis de programar quanto mais portáteis. Também existe uma versão paralela para multicomputadores, a PBLAS (*Parallel Basic Linear Algebra Subprograms*) [49], desenvolvida com base na versão sequencial da BLAS e da biblioteca BLACS [42].

Com o desenvolvimento e popularização da BLAS, novos pacotes como a LINPACK [48], LAPACK (*Linear Algebra PACKage*) [52], PLASMA [25], MAGMA [10] e ScaLAPACK (*Scalable Linear Algebra PACKage*) [50] foram implementados com base nas suas rotinas.

LINPACK é uma coleção de subrotinas em *Fortran* que exploram o nível 1 da BLAS para analisar e resolver Equações Lineares. Ao contrário da LINPACK, que utiliza algoritmos orientados à coluna, a LAPACK usa algoritmos que executam as operações de matrizes nos blocos mais internos. LAPACK foi projetada para ter alta eficiência em processadores vetoriais, estações de trabalho super escalares de Alto Desempenho e multiprocessadores de memória compartilhada.

PLASMA (*Parallel Linear Algebra for Scalable Multi-core Architectures*) [25], é um pacote de Álgebra Linear que foi projetado para oferecer o melhor desempenho possível a partir de um sistema com múltiplos *sockets* de processadores *multicore*. Esta biblioteca é disponibilizada livremente e pode ser incorporada em *softwares* comerciais. A MAGMA [10] é uma biblioteca de Álgebra Linear

Densa semelhante à LAPACK, mas para arquiteturas heterogêneas e híbridas, como por exemplo, a união de sistemas *multicores* com GPU.

O pacote ScaLAPACK é destinado a adaptar a LAPACK para outras arquiteturas como as máquinas massivamente paralelas SIMD (*Single Instruction Multiple Data*) ou máquinas de memória distribuída MIMD (*Multiple Instruction Multiple Data*) [13]. Conforme o pacote LAPACK, as rotinas do ScaLAPACK são baseadas nos algoritmos de particionamento em blocos. Esta distribuição cíclica em blocos provê um mecanismo simples para distribuição de dados com algoritmos particionados em blocos para arquiteturas de memória distribuída. Os níveis 1, 2 e 3 da PBLAS são utilizados no ScaLAPACK juntamente com a BLACS [42]. A ScaLAPACK possibilita o desenvolvimento de aplicações de Alto Desempenho que resolvem SELAs.

3.2 Computação Verificada

Resultados de computações em ponto flutuante, em alguns casos, não são representados corretamente devido a erros de arredondamento. Estes erros ocorrem pela impossibilidade de se gerar certos números no sistema de ponto flutuante, que é implementado pelo padrão IEEE 754. Conforme previamente dito, a Computação Verificada garante o rigor matemático de um resultado de uma computação fornecendo um intervalo (chamado de *enclosure*) que, certamente, contém o resultado exato [28], [32]. Neste tipo de computação, se a solução não for encontrada na execução, o algoritmo deve permitir que o usuário saiba [29]. Entretanto, encontrar o resultado verificado muitas vezes aumenta drasticamente o tempo de execução [28].

Na Computação Verificada, técnicas de análise numérica foram desenvolvidas para tornar possível que o computador verifique a corretude dos resultados computados em inúmeros problemas e aplicações [22]. Para garantir essa verificação, existem alguns métodos chamados de métodos auto-validados (*self validated*). De acordo com Rump [57], os objetivos destes métodos são: produzir resultados rigorosos e incluir a prova da existência da solução. Uma maneira de produzir resultados rigorosos é a utilização da Aritmética Intervalar e arredondamentos direcionados para representar estes resultados. A Aritmética Intervalar é uma aritmética definida para intervalos, ao invés de números reais [53].

A Figura 3.1 apresenta os requisitos necessários para a obtenção da Computação Verificada, sendo eles: algoritmos apropriados, aritmética intervalar e arredondamentos direcionados.



Figura 3.1 – Requisitos para a obtenção da Computação Verificada. Baseado em: [9], [24]

3.2.1 C-XSC

A biblioteca C-XSC (*C for eXtended Scientific Computation*) [4] foi implementada em C++ e é direcionada ao desenvolvimento de algoritmos que necessitam de resultados em alta exatidão. Esta biblioteca possibilita trabalhar com tipos de dados de alta precisão e com verificação de resultados [22].

Tabela 3.1 – Tipos de Dados do C-XSC

Tipo de Dado	Descrição
<i>rvector</i>	vetor de números reais
<i>ivector</i>	vetor de números reais em intervalos
<i>cvector</i>	vetor de números complexos
<i>civector</i>	vetor de números reais em intervalos complexos
<i>rmatrix</i>	matriz de números reais
<i>imatrix</i>	matriz de números intervalares
<i>cmatrix</i>	matriz de números complexos
<i>cimatrix</i>	matriz de números reais em intervalos complexos

Esta biblioteca trabalha com classes que implementam vários tipos de dados numéricos, os quais foram desenvolvidos para trabalhar com valores em alta exatidão e intervalos. Os tipos de dados convencionais não suportam estes tipos de dados como, por exemplo, o tipo de dado *interval* (intervalo de reais), *complex* (número complexo) e *cinterval* (intervalos complexos) [23]. Com base nestes tipos de dados, o C-XSC possui tipos de dados especiais para matrizes e vetores apresentados na Tabela 3.1 com sua descrição.

Para estes tipos de dados também foram desenvolvidos operadores especiais. O C-XSC utiliza a sobrecarga de operadores para implementar métodos específicos e, desta forma, possibilita que as operações matemáticas entre as matrizes e vetores sejam realizadas de forma simplificada quando possível. A sobrecarga de operadores torna possível multiplicar duas matrizes sendo A , B e C variáveis do tipo *rmatrix* utilizando o operador de multiplicação conforme a Equação 3.1:

$$C = A * B \quad (3.1)$$

Além dos tipos de dados descritos na Tabela 3.1, o C-XSC possui variáveis de alta precisão, também denominadas acumuladores de alta exatidão. São elas: *dotprecision*, *cdotprecision*, *idotprecision* e *cidotprecision*. A Figura 3.2 apresenta estes acumuladores de ponto fixo, onde t é o tamanho da mantissa, $emax$ e $emin$ representam o expoente máximo e o mínimo e g corresponde aos dígitos de guarda (*guard digits*), que evitam o *overflow* [31]. Um acumulador *dotprecision* ocupa um espaço de armazenamento, com $g=31$, de no mínimo 4227 bits [24]. Estes acumuladores de alta exatidão (*dotprecision*) podem armazenar resultados de uma expressão sem erro de arredondamento, possibilitando a realização do produto escalar ótimo com apenas um arredondamento no resultado final baseado no padrão IEEE-754 [24].

g	$2e_{max}$	t	t	$2 e_{min} $
-----	------------	-----	-----	--------------

Figura 3.2 – Variável de alta precisão. Fonte: [31]

3.3 Eficiência Energética em Alto Desempenho

Tópicos acerca de Computação Verde estão tendo maior importância nos últimos anos. De acordo com Murugesan et al [44], a Computação Verde beneficia o meio ambiente, uma vez que ela impulsiona o intuito de pesquisar formas de manter o constante desenvolvimento tecnológico aliado a métodos de preservação ambiental.

Aumentar a eficiência energética é importante, independentemente do *hardware* adotado, em todo o contexto computacional. Ou seja, para todas as arquiteturas e implementações físicas de máquinas paralelas, a eficiência energética é um fator crucial a ser analisado, inclusive, utilizando melhor os recursos computacionais e assim, melhorando o desempenho [46]. Mór et al [46] considera que utilizar melhor os recursos computacionais melhora também a eficiência energética.

Quando se trata da execução de um programa paralelo em uma plataforma de memória distribuída (e.g. *cluster*), o consumo energético é basicamente multiplicado pela quantidade de nós. Portanto, a execução paralela valerá a pena em termos de gasto de energia somente se a aceleração é linear em relação ao número de nós, a fim de compensar a sobrecarga de energia. Para a maioria das aplicações isso será verdade apenas até um certo número [5].

Porém, quanto mais máquinas se utiliza para a solução de um problema, maior o consumo de energia. E se não há preocupação com este consumo no momento de paralelizar um problema, pode-se ter desperdício, diminuindo assim a eficiência energética deste programa. De acordo com Wang et al [61], Computação Verde e o baixo consumo de energia estão ligados. Sendo assim, a indústria e a pesquisa da área de Tecnologia da Informação precisam focar no desenvolvimento de soluções nas quais diminuir o consumo de energia é prioridade.

O desenvolvimento da arquitetura do processador paralelo também pode ajudar ainda mais na redução do consumo de energia, com novas técnicas de aumento de desempenho [46]. Da mesma forma, o hardware pode permitir a redução do consumo, ao possibilitar o desligamento de unidades que não estão sendo utilizadas, ou também permitindo a alteração da frequência e a voltagem do processador através da técnica DVFS.

3.4 Técnica DVFS

A técnica DVFS (*Dynamic Voltage and Frequency Scaling*) é uma técnica moderna para reduzir o consumo de energia em microprocessadores ou controlar o calor gerado pelo circuito [55]. De acordo com Rizvandi, [55] esta técnica é comumente usada em dispositivos como *notebooks* e

celulares, onde diminuir o consumo de energia da bateria é necessário. Além disso, DVFS é utilizada na área de Alto Desempenho, para diminuir o processamento dos nós e assim reduzir energia ou esfriá-los.

Segundo estudos realizados por Feng et al. [18], com o *benchmark* SPEC CPU2000, o consumo de energia de uma CPU (*Central Processing Unit*) de um nodo durante a execução corresponde a aproximadamente 35% do total da energia consumida por esse nodo. Por outro lado, quando o nodo está ocioso, o consumo de energia da CPU é reduzido para o valor de 15%. O restante é dividido entre memória, *coolers*, fonte de alimentação, interface de rede, disco e *chipset*.

O consumo dinâmico do processador é definido pela Equação 3.2.

$$P = c * v^2 * f \quad (3.2)$$

onde, c é a capacitância do circuito, v a voltagem e f a frequência. A relação entre frequência e voltagem pode ser definida através da equação 3.3, descrita por Dong [12]:

$$f \propto \frac{(V - V_t)^\gamma}{V} \quad (3.3)$$

onde, $\gamma = 2$ e a frequência f é diretamente proporcional a voltagem V .

Deste modo, a voltagem pode ser expressa como uma função linear da frequência, isto é $V = \alpha f$. Desta forma, de acordo com Dong et al. [12] e Elnozahy et al. [14], considera-se que o consumo de energia dinâmico é aproximadamente proporcional ao cubo da frequência (Equação 3.4).

$$P = f^3 \quad (3.4)$$

Os processadores da maioria das máquinas atuais permitem o uso da técnica DVFS [54]. Conforme visto no Capítulo 2, já existem na literatura trabalhos que usam a técnica DVFS para reduzir a frequência e consumo. Dentre estes destacam-se trabalhos que utilizam DVFS. Lim et al. [34] introduz um sistema que reduz dinamicamente a frequência da CPU durante as fases de comunicação MPI em programas paralelos. Dong et al. [12] utilizou DVFS com MPI para reduzir a frequência durante as operações de *Gather* e *Scatter* do MPI. Já o trabalho de Kimura et al. [27] implementou um algoritmo com DVFS durante o *slack time* de um programa MPI. Kappiah et al. [26] desenvolveu um sistema denominado Jitter, que reduz a frequência em nodos nos quais são atribuídos menos computações, ajustando o *slack time*.

Existem bibliotecas e pacotes que permitem o uso de técnica DVFS em computadores, tais como: *cpufrequtils*, *cpupower*, *powernowd*, *cpudyn*, *cpufreqd*, entre outras. Este trabalho destaca o pacote *cpufrequtils*.

Para fazer uso da técnica DVFS em ambientes *Linux* pode-se instalar o pacote *cpufrequtils* (conhecido também como CPU-Freq), para alterar a frequência dos processadores. Através deste pacote é possível alterar os governadores do processador, que controlam a política de frequência do processador. Existem cinco tipos de governadores que podem ser definidos:

- *Ondemand*: governador padrão, que altera a frequência conforme a necessidade do sistema;
- *Userspace*: permite ao usuário alterar a frequência do processador manualmente;
- *Powersave*: altera a frequência do processador para o valor mínimo;
- *Conservative*: similar ao *ondemand*, sendo que a frequência é alterada gradualmente conforme a média de utilização do sistema;
- *Performance*: altera a frequência do processador para o valor máximo;

Para definir estes governadores e outras configurações pode-se utilizar o comando *cpufreq-set* em modo *root*. Este comando possui os seguintes parâmetros:

- *-g*: usado para alterar o governador;
- *-c*: usado para definir o número do processador que será alterado. Se esse valor não for definido, por padrão será o processador 0;
- *-u*: tem a finalidade de alterar o limite superior da frequência;
- *-d*: altera o limite inferior da frequência;
- *-f*: altera o valor da frequência. Porém, para ser possível alterar a frequência, o governador do processador deve ser definido para *Userspace*;

O comando *cpufreq-info* permite monitorar a frequência dos processadores, apresentando informações como a frequência corrente, frequências possíveis, intervalo de frequência corrente, governador corrente, dentre outros.

O pacote *cpufrequtils* já foi utilizado em diversos trabalhos para gerenciar a frequência dos processadores, como por exemplo os trabalhos de Ge et al. [20], Lim et al. [34], Sharma et al. [60], Minartz et al. [43] e Castillo et al. [6].

4. ANÁLISE PRELIMINAR DO SOLVER FASTPILSS

Este capítulo apresenta um estudo cujo objetivo é analisar as características de desempenho, exatidão e consumo de energia do *solver* de Zimmer [31]. A Seção 4.1 apresenta o *solver* de Zimmer, intitulada FastPILSS. A Seção 4.2 descreve o ambiente de testes e as matrizes de entrada utilizadas para os teste realizados com o *solver* de Zimmer. As Seções 4.3, 4.4 e 4.5 apresentam os resultados de desempenho, exatidão e consumo de energia. Por fim, a Seção 4.6 apresenta as considerações sobre as análises realizadas.

4.1 FastPILSS

Os primeiros *solvers* que utilizavam a biblioteca C-XSC obtinham alta exatidão, porém, eram bastante lentos [31]. O *solver* *FastPILSS* (*Fast Parallel Interval verified Linear System Solvers*) [17] foi desenvolvido para ter melhor desempenho que o *solver* anterior, cujo objetivo é calcular o resultado verificado de um Sistema Linear. De acordo com Krämer [31], este *solver* foi desenvolvido para suportar todos os tipos básicos de variáveis do C-XSC descritas no Capítulo 2. Desta forma, foi utilizado o ambiente da biblioteca C-XSC que fornece as ferramentas necessárias (aritmética intervalar, acumuladores de alta precisão, etc.) para computação verificada, possibilitando assim, obter resultados com alta exatidão.

Este *solver* utiliza o algoritmo DotK [31] proposto por Rump [56]. De acordo com Krämer [31], o objetivo principal do algoritmo DotK é fornecer um tratamento similar ao das classes *dotprecision*, que são utilizadas como acumuladores de alta precisão, e melhorar o desempenho. Desta forma, foram criadas novas classes DotK (RDotK para produtos de números reais, IDotK para produtos de números intervalares, CDotK para produtos de números complexos, CIDotK para produtos de números intervalares complexos). Cada nova classe possui um conjunto de construtores, operadores e alguns métodos básicos de *get* e *set*. Cada classe oferece três métodos importantes: *addDot*, que recebe dois vetores em ponto flutuante como parâmetro e calcula o produto destes vetores na precisão desejada de acordo com o valor de K fornecido como entrada; *res*, que retorna o resultado ignorando os limites de erros; *res_enclosure*, que retorna o *enclosure* do resultado utilizando os limites de erros, lembrando que *enclosure* é um intervalo que, certamente, contém o resultado exato [28], [32].

O *solver* é baseado no algoritmo descrito por Rump [56] baseado no método de Krawczyk [30], também desenvolvido com o objetivo de utilizar o *solver* em sistemas mal condicionados. O Algoritmo 3 apresenta o modelo básico do algoritmo autoverificado utilizado neste *solver* para resolver Sistemas Lineares.

Para aperfeiçoar o cálculo de C , contido no Algoritmo 4.1, foram utilizadas as classes DotK, onde também utilizou-se o método *addDot* e o método *res_enclosure*. As rotinas da BLAS e LAPACK foram utilizadas no cálculo de C e no cálculo da inversa aproximada de R . No cálculo

Algorithm 1 Algoritmo do *solver* FastPILSS. Adaptado de: [31]

Entrada: Matriz quadrada de A e um vetor b

Saída: Solução em intervalo para $Ax = b$

// PASSO 1 - Calcula o inverso aproximado R de A

$R \approx A^{-1}$

// PASSO 2 - Calcula solução aproximada

$\tilde{x} := Rb$

// Iteração

repita

| $\tilde{x} := \tilde{x} + R(b - A\tilde{x})$

até que \tilde{x} alcance máxima precisão ou alcance o número máximo de iterações;

// PASSO 3 -Cálculo do Z

$Z := R \diamond (b - A\tilde{x})$

// PASSO 4 - Cálculo do C

$C := \diamond(I - RA)$

// PASSO 5 -Verificação

$Y := Z$

repita

| $Y_A := blow(Y, \varepsilon)$ // ε -inflação

| $Y := Z + C \cdot Y_A$

até $Y \subset interior(Y_A)$ ou até alcançar o máximo de iterações);

// Checar Resultados

se $Y \subset interior(Y_A)$ **então**

| Solução única em $x \in \tilde{x} + Y$

senão

se Inversa aproximada de comprimento duplo ainda não utilizados **então**

| // Segundo estágio

| $R_1 := R$

| $S := R_1 \cdot A$

| Calcular o inverso S_1 de S

| $S := S_1 \cdot R_1$

| $R_2 := S_1 \cdot R_1 - S$

| $R_1 := S$

| Reiniciar o algoritmo com a nova inversa aproximada de $R = R_1 + R_2$ (A soma não deve ser computada)

fim se

| Algoritmo falha. A é singular ou mal condicionado.

fim se

da inversa aproximada foram utilizadas as rotinas *xgetrf* e *xgetri*. No cálculo de C e no cálculo do produto entre matrizes foi utilizada a rotina *dgemm* da BLAS [31]. Com estes aperfeiçoamentos, o *solver* sequencial obteve um melhor desempenho comparado ao *solver* em C-XSC anterior [31].

A nova versão paralela utilizou as rotinas da biblioteca ScaLAPACK ao invés das rotinas da LAPACK. Em busca de maior eficiência, as matrizes utilizadas são distribuídas igualmente entre todos os processos, sendo que cada processo guarda uma parte das matrizes A , R e C . As matrizes são distribuídas da mesma forma que a distribuição pela ScaLAPACK, em blocos cíclicos bi-dimensionais.

A biblioteca MPI também foi utilizada para possibilitar a troca de mensagens entre os processos [31].

A paralelização de memória distribuída realizada pelo *solver* permite o cálculo de uma solução verificada para Sistemas Lineares de grande porte (de dimensão 100.000 ou até maior), com os dados distribuídos. Desta forma, a dimensão do sistema é limitada apenas pelo tamanho da memória dos nodos do *cluster* utilizado [17].

4.2 Cenário

Os experimentos foram realizados no *cluster* Atlântica do Laboratório de Alto Desempenho da Pontifícia Universidade Católica do Rio Grande do Sul (LAD). O *cluster* possui 16 máquinas Dell PowerEdge R610 sendo que cada nodo consiste em dois processadores Intel Xeon Quad-Core E5520.

Para avaliação do consumo de energia é necessário identificar o consumo de um nodo. Os valores de potência em execução dos nodos, com o governador em *performance* e em *powersave* dos nodos foram obtidos com o uso do multímetro digital EZ-735, ficando em 270 watts a potência de execução em *performance* e 219 watts executando em *powersave* quando executado com o *benchmark* gromacs [59].

Para os dados de entrada, foram utilizadas matrizes geradas e matrizes reais. Os valores de entradas das matrizes geradas são obtidos através de fórmulas matemáticas. Já os valores de entradas das matrizes reais são obtidos com dados de matrizes utilizadas em aplicações reais. Foram executados testes com as matrizes geradas pelas Equações 4.1 e 4.2 correspondentes à Matriz Gen1 e à Matriz DingDong. Sendo, n a ordem da matriz e M_PI o número Pi .

$$Gen1_{i,j} = \text{sqrt}(2.0/(n + 1)) * \sin((i + *j * M_PI)/(n + 1)); \quad (4.1)$$

$$DingDong_{i,j} = 0.5/((n - i - j) + 1.5); \quad (4.2)$$

Além das matrizes de entrada geradas através de equações, foram testadas duas matrizes de aplicações reais. Elas foram obtidas através do site Matrix Market [35], que contém um repositório de dados de teste para uso em estudos comparativos de algoritmos para Álgebra Linear Numérica, com cerca de 500 matrizes esparsas de uma variedade de aplicações. As matrizes utilizadas foram Fidapm29 e Fidap019, que pertencem a um pacote de testes de equações da área de Fluidodinâmica computacional (C.F.D.) [39] [38]. Reitera-se que o algoritmo fornece um resultado intervalar, que contém o resultado correto, porém os valores de entrada A e b são reais.

O valor do vetor b foi definido como $b[i]=1$, para todos valores de i , onde i varia de 1 até a dimensão da matriz. O tamanho das matrizes testadas pode ser visto na Tabela 4.1. Nesta tabela também é possível visualizar o número de condicionamento de cada matriz. O condicionamento é um número que aponta o quanto as modificações dos dados influenciam na qualidade do resultado.

Sendo uma matriz A , o seu número de condicionamento é obtido através do cálculo do produto da norma¹ de A pela norma de sua inversa.

Tabela 4.1 – Características das matrizes de entrada.

Matriz	Ordem	Número de condicionamento	Esparsa/Densa	Real/Gerada	Aplicação
Gen1	45.000/ 30.000/ 15.000	1,21E+04/ 2,43E+04/ 3,64E+04	Densa	Gerada	Teste
DingDong	45.000/ 30.000/ 15.000	1,50E+03/ 2,26E+03/ 2,86E+03	Densa	Gerada	Teste
Fidamp29	13.668	7,32E+05	Esparsa	Real	C.F.D.
Fidap019	12.005	3,92E+12	Esparsa	Real	C.F.D.

Foram executados testes utilizando 8, 16, 24 e 32 processos em 4 nodos. Foi escolhida uma estratégia de escalonamento que prioriza a alocação dos processos em nodos físicos. Sendo assim, devido a estratégia de escalonamento priorizar os processadores físicos, o número de processos por nó é de 2 para 8 processos, 4 para 16 processos, 6 para 24 processos e 8 para 32 processos. Reiterando que a divisão dos processos físicos dentro do nó é balanceada, ou seja, como são dois processadores físicos dentro de um nó, se são 2 processos por nó, cada processador recebe um processo.

O *solver* testado foi subdividido em passos. Os passos do *solver* foram definidos com base na organização dos algoritmos descrita na Seção 4.1, apenas com a inclusão da alocação das matrizes e variáveis na memória (Passo 1). Esta organização auxiliará nas análises dos resultados de desempenho do *solver*. Os passos do *solver* foram definidos como segue:

- **Passo 1:** alocação;
- **Passo 2:** cálculo da inversa;
- **Passo 3:** cálculo do x ;
- **Passo 4:** cálculo do z ;
- **Passo 5:** cálculo do C ;
- **Passo 6:** verificação.

4.3 Desempenho

Para fazer a análise de desempenho será utilizada a medida de tempo de execução. O objetivo é analisar o tempo de execução obtido pelo *solver* com matrizes de diferentes tamanhos,

¹Mais informações sobre norma e número de condicionamento podem ser encontradas em Claudio [9].

utilizando 1 ou 4 nodos e diferentes números de processos. Esta análise será importante para o cálculo estimado do consumo de energia.

A Tabela 4.2 apresenta os tempos de execução, em segundos, referente a cada passo do método e a soma do tempo de execução de cada passo em segundos. Para cada matriz foram feitos experimentos com 8 processos, sobre 1 e 4 nodos. Para os casos em que as matrizes apresentavam ordem maior que 15.000 não foi possível realizar o teste usando apenas 1 nodo por limitações de memória. Na descrição da Tabela 4.2 foi omitido o tempo de execução do Passo 1, relativo à alocação das matrizes e variáveis na memória. As tabelas com os valores do tempo de execução com 16 e 32 processos estão no Apêndice A.

Tabela 4.2 – Tempos de execução dos passos no *solver* de Zimmer com 8 processos

Solver			Zimmer					
Matrizes	Ordem	Nodos	Passo 2	Passo 3	Passo 4	Passo 5	Passo 6	Total
Gen1	45.000	4	3.508,36	51,28	25,91	8.575,44	567,99	12.728,97
DingDong	45.000	4	3.478,90	19,46	25,30	8.629,60	496,96	12.650,23
Gen1	30.000	4	1.297,52	24,93	12,39	1.883,39	7,81	3.226,03
DingDong	30.000	4	1.050,69	8,85	11,52	1.758,37	7,02	2.836,43
Gen1	15.000	1	210,07	5,87	2,85	377,45	2,64	598,87
Gen1	15.000	4	155,67	6,06	3,05	226,83	1,77	393,38
DingDong	15.000	1	178,85	2,16	2,81	372,18	1,99	558,00
DingDong	15.000	4	148,88	2,30	2,98	225,81	1,81	381,78
Fidapm029	13.668	1	176,43	2,72	3,75	281,94	2,15	466,99
Fidapm029	13.668	4	114,57	1,92	2,55	179,42	1,38	299,85
Fidap019	12.005	1	96,57	2,04	1,90	124,58	1,08	226,16
Fidap019	12.005	4	79,17	2,13	1,98	123,85	1,03	208,15

Tabela 4.3 – Tempos de execução dos passos no *solver* de Zimmer com 24 processos

Solver			Zimmer					
Matrizes	Ordem	Nodos	Passo 2	Passo 3	Passo 4	Passo 5	Passo 6	Total
Gen1	45.000	4	1.849,34	21,16	10,37	6.672,18	352,04	8.905,09
DingDong	45.000	4	1.755,75	7,94	10,15	6.468,02	381,90	8.623,75
Gen1	30.000	4	610,42	10,31	5,08	683,44	3,20	1.312,44
DingDong	30.000	4	582,20	3,89	4,99	656,15	3,09	1.250,32
Gen1	15.000	4	90,63	3,47	1,65	93,05	0,85	189,65
DingDong	15.000	4	85,62	1,40	1,56	91,43	0,81	180,83
Fidapm029	13.668	4	78,52	1,11	1,40	68,09	0,73	149,84
Fidap019	12.005	4	56,30	2,73	1,11	46,52	0,50	107,16

Ao observar as tabelas, cabe ressaltar que há diferença no tempo de execução ao utilizar quantidades de nodos diferentes com 8 processos. Percebe-se que quando foi utilizado apenas um nodo, o tempo de execução aumentou. Assume-se que isto ocorre devido à concorrência no acesso à memória e na quantidade de processos quando se utiliza apenas um nodo.

A Figura 4.1 apresenta uma comparação entre o tempo de execução do *solver* de Zimmer para a matriz de entrada Gen1 com ordens de 45.000, 30.000 e 15.000. Para cada ordem foram executados testes com 8, 16, 24 e 32 processos. Observa-se que conforme o número de processos aumenta, o tempo de execução de Zimmer diminui em todos os casos. Além disso, é possível perceber que em matrizes com menor ordem a diferença no tempo de execução é menor, como é

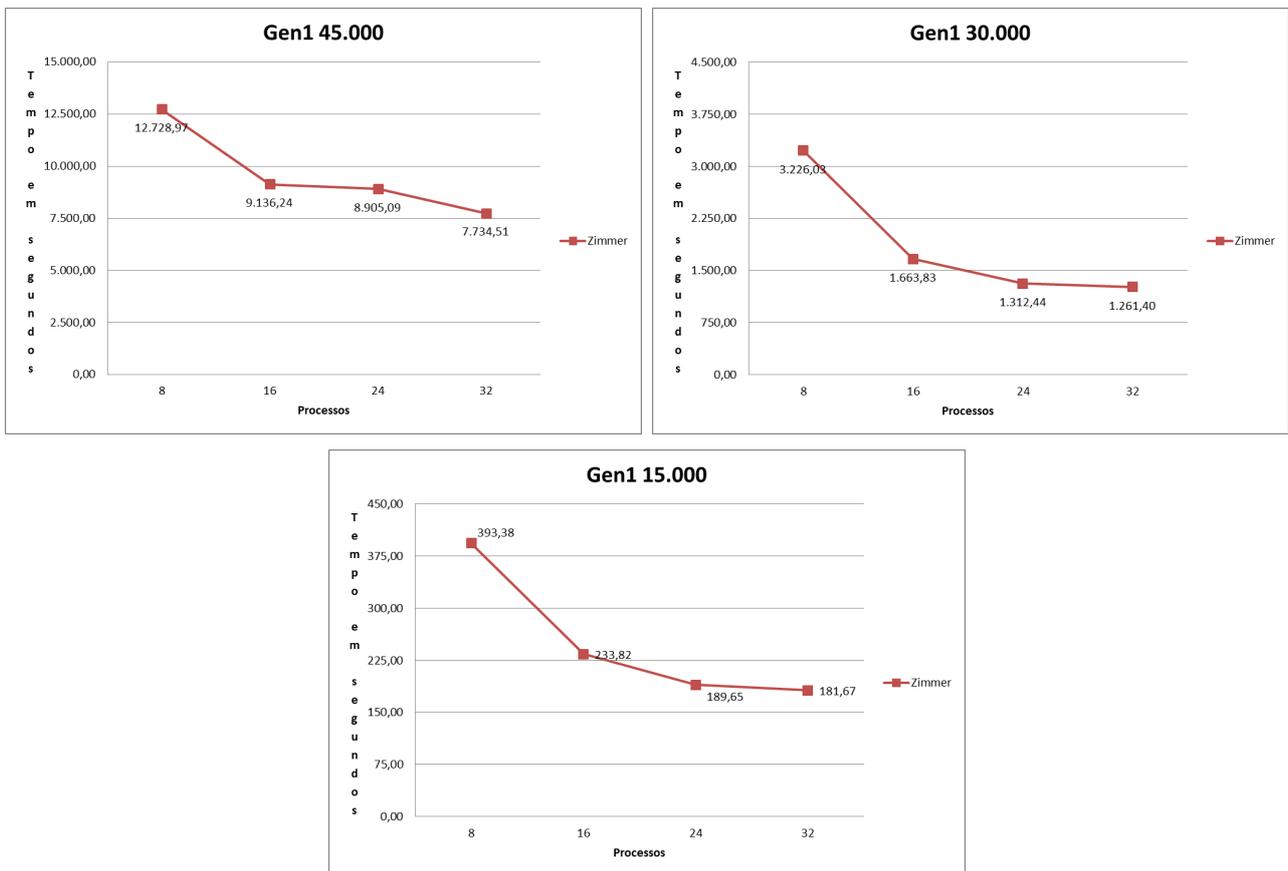


Figura 4.1 – Comparação entre o tempo de execução do solver para a matriz de entrada Gen1.

o caso entre 24 e 32 processos. Este comportamento já era esperado, visto que com matrizes de menor ordem o tempo de processamento também é menor e, por isso, não compensa o uso de mais unidades de processamento. Para este caso específico, 32 processos é o limite ideal para tornar a paralelização viável.

A Figura 4.2 apresenta uma comparação dos tempos de execução do *solver* para a matriz de entrada DingDong. A forma de execução é a mesma que a matriz de entrada Gen1, com as mesmas ordens e a mesma quantidade de processos. Foi observado o mesmo comportamento do experimento realizado com a matriz Gen1.

A Figura 4.3 apresenta o tempo de execução do *solver* para as matrizes reais Fidapm29 e Fidap019, com a mesma quantidade de processos utilizada nos testes com as matrizes densas: 8, 16, 24 e 32 processos. Da mesma forma que ocorreu para as matrizes de ordem 15.000, ao executar o *solver* com 32 processos para a matriz Fidapm29 pode-se notar que há uma pequena diminuição no tempo de execução. Já para a matriz Fidap0419, que apresenta menor tamanho de ordem, com 32 processos há aumento no tempo de execução em relação ao resultado do tempo com 24 processos. Com isso, percebe-se que a paralelização com 32 ou mais processos não é vantajosa para matrizes com esta ordem ou semelhantes.

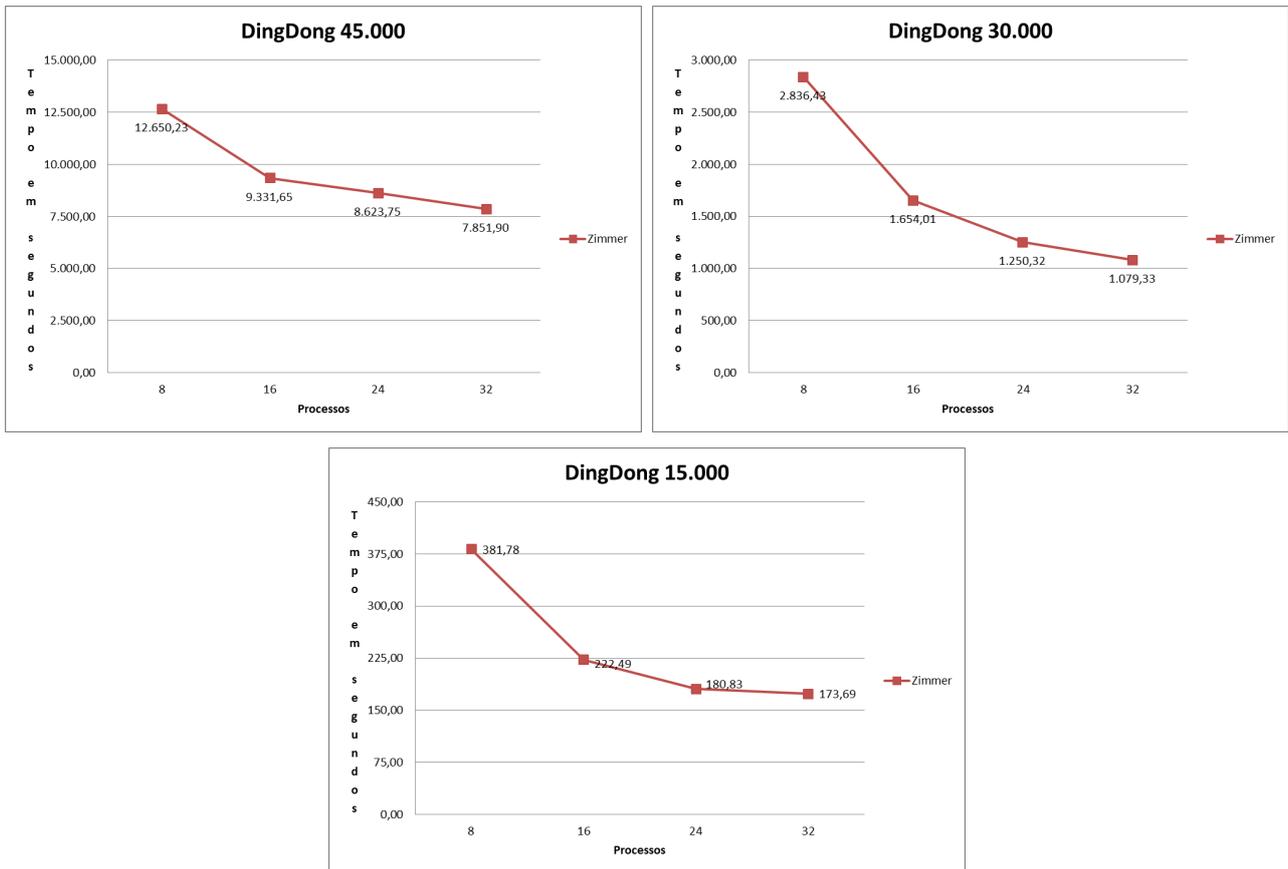


Figura 4.2 – Comparação entre o tempo de execução do solver para a matriz de entrada DingDong.

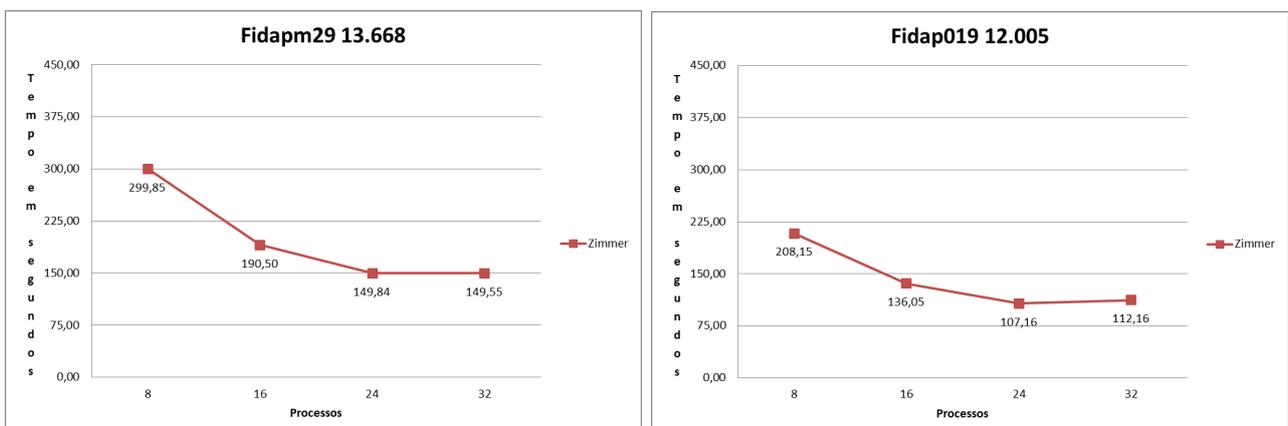


Figura 4.3 – Comparação entre o tempo de execução do solver para as matrizes reais de entrada Fidapm29 e Fidap019.

4.4 Exatidão

Em Computação Verificada o foco não está só no desempenho da aplicação, mas no controle dos erros que podem ocorrer durante as computações numéricas que prejudicam a qualidade do resultado. Desta forma, analisar o resultado numérico de uma aplicação verificada é importante, já que o resultado obtido deve ser um intervalo que contém a resposta correta. Os resultados de Zimmer [31] são apresentados para avaliar a exatidão. As Tabelas 4.4, 4.5, 4.6 e 4.7 apresentam os resultados intervalares obtidos para cada matriz de entrada.

Tabela 4.4 – Resultados dos *solvers* para a matriz Gen1 de ordem 45.000 com 32 processos.

x[]	Zimmer
x[0]	[1.909880536868255E+002, 1.909880536868257E+002]
x[1]	[-1.440394679963117E-014, 2.885165611176025E-014]
x[2]	[6.366268435542821E+001, 6.366268435542827E+001]
x[3]	[-3.861489088129587E-014, 4.825807158652983E-015]
x[4]	[3.819761036504061E+001, 3.819761036504066E+001]
x[5]	[-2.359263404056874E-014, 1.995640499233034E-014]
x[6]	[2.728400713765435E+001, 2.728400713765441E+001]
x[7]	[-1.917478541573541E-014, 2.445167871969569E-014]
x[8]	[2.122089416460189E+001, 2.122089416460195E+001]
x[9]	[-3.678556161724684E-014, 6.899836232456892E-015]

Tabela 4.5 – Resultados dos *solvers* para a matriz DingDong de ordem 45.000 com 32 processos.

x[]	Zimmer
x[0]	[2.393647033379822E+002, 2.393647033379827E+002]
x[1]	[1.196810218503079E+002, 1.196810218503082E+002]
x[2]	[8.975976901263635E+001, 8.975976901263650E+001]
x[3]	[7.479897635538266E+001, 7.479897635538280E+001]
x[4]	[6.544837704212434E+001, 6.544837704212447E+001]
x[5]	[5.890288478868657E+001, 5.890288478868668E+001]
x[6]	[5.399371104617194E+001, 5.399371104617204E+001]
x[7]	[5.013646024156724E+001, 5.013646024156734E+001]
x[8]	[4.700240913461814E+001, 4.700240913461822E+001]
x[9]	[4.439067085435308E+001, 4.439067085435317E+001]

Tabela 4.6 – Resultados dos *solvers* para a Matriz Fidap019 com 32 processos

x[]	Zimmer
x[0]	[4.571964621165768E+003, 4.571964628737764E+003]
x[1]	[-1.663813454000574E+002,-1.663813423072850E+002]
x[2]	[-6.871703922032086E+002,-6.871703852777510E+002]
x[3]	[3.769679744360134E+002, 3.769679762750578E+002]
x[4]	[-5.519618296494698E+001,-5.519617543827807E+001]
x[5]	[-8.451577041524918E+001,-8.451576915166161E+001]
x[6]	[-1.592901891095857E+002,-1.592901831088274E+002]
x[7]	[1.126918243015329E+002, 1.126918252184748E+002]
x[8]	[-7.335637844620139E+001,-7.335637334380605E+001]
x[9]	[-4.422795444781231E+001,-4.422795381188728E+001]

Como é possível verificar nos resultados de exatidão, os diâmetros² dos intervalos dos resultados de Zimmer [31] são bem pequenos. Por exemplo, na Tabela 4.4 para a matriz de entrada

²Diâmetro de um intervalo (d): número real não negativo $d = a_2 - a_1$ [53]

Tabela 4.7 – Resultados dos *solvers* para a Matriz Fidapm29 com 32 processos

$x[]$	Zimmer
$x[0]$	[1.530081256362627E+001, 1.530081256362800E+001]
$x[1]$	[-6.208148168356456E-001,-6.208148168356214E-001]
$x[2]$	[1.579105158878873E+001, 1.579105158879001E+001]
$x[3]$	[-1.467169571729604E+000,-1.467169571729538E+000]
$x[4]$	[1.447675288907328E+001, 1.447675288907424E+001]
$x[5]$	[-2.316536672396893E+000,-2.316536672396800E+000]
$x[6]$	[1.260962590807345E+001, 1.260962590807429E+001]
$x[7]$	[-3.132407229768991E+000,-3.132407229768878E+000]
$x[8]$	[1.038044496201669E+001, 1.038044496201753E+001]
$x[9]$	[-3.895651110979420E+000,-3.895651110979293E+000]

Gen1, o diâmetro do resultado $x[0]$ de Zimmer é $2,00E-13$. Lembrando que na Aritmética Intervalar quanto menor o diâmetro do intervalo, mais exato é o resultado. Desta forma, os resultados intervalares de $x[0]$ de Zimmer possuem uma boa exatidão. Reiterando que os resultados são inclusões da resposta correta.

O resultado intervalar para a matriz gerada DingDong manteve o mesmo padrão. O diâmetro do resultado intervalar do $x[0]$ de Zimmer manteve o expoente e é de $5,00E-13$. Entretanto para a matriz real Fidap019 apresentada na Tabela 4.6, o diâmetro do resultado intervalar de Zimmer é de $7,57E-06$. Acredita-se que o diâmetro deste resultado foi maior devido ao condicionamento da matriz Fidap019 ser próximo das matrizes mal condicionadas ($3.92E+12$), o que dificulta a resolução do sistema. Por fim, o diâmetro do resultado intervalar de $x[0]$ para a matriz real Fidapm29 é de $1,73E-12$, conforme apresentado na Tabela 4.7.

Acredita-se que o *solver* de Zimmer obteve intervalos com diâmetro pequeno devido ao uso da biblioteca C-XSC e da implementação da técnica proposta por Rump (DotK) [56].

4.5 Estimativa de Consumo de Energia

Tendo em vista que na plataforma do LAD não foi possível o acesso aos medidores, a análise do consumo será realizada com base no valor de frequência máxima de um nodo, conforme trabalho de Anzt [2]. Desta forma, considera-se que durante a execução normal, os processadores estão com frequência e voltagem máximas.

O consumo de energia (C) do *solver* é avaliado com base no consumo do nodo. Conforme apresentado na Seção 4.2, o valor da potência (P) em alto desempenho de um nodo é de 270 watts . De acordo com a Equação 4.3 sabe-se que:

$$P = WATTS = \text{joule}/\text{segundos} \quad (4.3)$$

Assim, o consumo de energia em *joule* (EC) é obtido multiplicando o tempo de execução (t) da aplicação em segundos com a potência (P), apresentado pela seguinte equação:

$$EC = P * t \quad (4.4)$$

Para a obtenção do consumo, é necessário saber a quantidade mínima de nós necessária para executar a aplicação. Então para obter o consumo de energia (EC_{total}) dos nós utilizados, multiplica-se o valor de consumo (EC) de um nodo pela quantidade de nós mínimos (a):

$$EC_{total} = EC * a \quad (4.5)$$

Desta forma, obtêm-se o consumo de energia em *joule*. Por exemplo, se a aplicação demorou 10 segundos para executar e foram utilizados dois nodos. Sabendo-se pela Equação 4.3 que o valor de potência é de 270 *watts* (também conhecido como 270 *joules* por segundo). Com base na Equação 4.4 multiplica-se o valor de 270 pelo tempo de duração em segundos (10 segundos) e obtêm-se o consumo de energia de um nodo (EC) de 2.700 *joules*. Para obter o consumo de energia total (EC_{total}) multiplica-se o valor de consumo obtido pela quantidade de nodos utilizados, neste caso foram 2 nodos. Por fim, obtêm-se o valor de consumo de energia de 5.400 *joules*.

Tabela 4.8 – Consumo de energia em *joules* com frequência máxima de 2.40Ghz com 8 processos.

Matriz	Nodos	Ordem	Consumo (J)
Gen1	4	45.000	13.747.286,25
DingDong	4	45.000	13.662.245,81
Gen1	4	30.000	3.484.116,72
DingDong	4	30.000	3.063.342,83
Gen1	1	15.000	161.695,93
Gen1	4	15.000	424.849,35
DingDong	1	15.000	150.661,19
DingDong	4	15.000	412.318,51
Fidapm29	1	13.668	126.086,13
Fidapm29	4	13.668	323.833,39
Fidap019	1	12.005	61.064,54
Fidap019	4	12.005	224.806,98

Tabela 4.9 – Consumo de energia em *joules* com frequência máxima de 2.40Ghz com 24 processos.

Matriz	Nodos	Ordem	Consumo (J)
Gen1	4	45.000	9.617.498,44
DingDong	4	45.000	9.313.653,97
Gen1	4	30.000	1.417.439,26
DingDong	4	30.000	1.350.346,28
Gen1	4	15.000	204.827,25
DingDong	4	15.000	195.292,12
Fidapm29	4	13.668	161.826,81
Fidap019	4	12.005	115.729,03

Tendo como base os valores estimados de consumo de energia, que estão diretamente relacionados com o tempo de execução do *solver* de Zimmer, é possível perceber nas Tabelas 4.8 e 4.9 que o *solver* tem um consumo de energia significativo para as matrizes de maior ordem. Cabe

ainda salientar que a quantidade de nodos influencia no consumo de energia, pois apesar do tempo de execução com 8 processos usando somente um nodo, ser maior que o tempo de execução, de 8 processos usando quatro nodos, o consumo de energia ao utilizar um nodo é menor que o consumo ao utilizar quatro nodos.

Para avaliar se os valores obtidos são considerados de alto consumo, realizou-se uma comparação dos valores com o consumo de aparelhos populares, como por exemplo, um chuveiro elétrico e um secador de cabelo. Conforme o INMETRO³, um chuveiro comum de uma determinada marca de 127V de voltagem apresenta potência em torno de 4.500 watts e em uma hora de uso apresenta consumo de energia de 16.200.000 joules. Já um secador de cabelo⁴ apresenta potência em torno de 1.000 watts e consumo de energia em uma hora de uso no valor de 3.600.000 joules.

Então, em média, apenas uma execução com a matriz de entrada Gen1 de ordem 45.000, com 24 processos e 4 nodos para o *solver* de Zimmer consome em joules mais que o consumo de meia hora de uso do chuveiro (8.100.000 J). Já uma execução com a mesma matriz de entrada de ordem 30.000 com 24 processos, o *solver* de Zimmer consome em *joules* um pouco menos que o consumo de meia hora de uso do secador de cabelo (1.800.000 J).

4.6 Considerações

Com base nesta análise dos resultados, considera-se que o *solver* de Zimmer apresenta uma boa escalabilidade à medida que os processos são utilizados. Além disso, os resultados numéricos são bastante precisos com intervalos de diâmetro pequeno. Acredita-se que o uso da biblioteca C-XSC, um ambiente completo para Computação Verificada, contribui para a exatidão obtido no *solver* de Zimmer, de acordo com os experimentos realizados.

Constatou-se que o tamanho da matriz de entrada influencia no consumo de energia. Observou-se também que se a redução do consumo de energia for prioridade, para matrizes que podem ser armazenadas em um nodo, convém utilizar apenas esse nodo, já que o aumento no tempo compensa energeticamente. Isto ocorre pois o consumo de energia na execução com um nodo é menor que o consumo de energia da execução da mesma matriz com mais nodos.

Observa-se através da Tabela 4.2 que os tempos de execução do Passo 2 e do Passo 5 são significativos em relação aos outros passos para todos os casos de teste. Por causa disso, nota-se que estes dois passos influenciam significativamente no tempo de execução e conseqüentemente no consumo de energia do *solver*.

Observando o alto consumo de energia do *solver* de Zimmer e a relevância da resolução de Sistemas de Equações Lineares em diversas áreas, considerou-se importante a realização de estudos sobre a redução do consumo de energia neste *solver*. Sendo assim, com base nos estudos apresentados no Capítulo 2, verificou-se a possibilidade de utilizar a técnica DVFS com o objetivo de reduzir

³<http://www.inmetro.gov.br/consumidor/pbe/chuveiro.pdf>

⁴<http://www.inmetro.gov.br/consumidor/produtos/secador.asp>

o consumo de energia e analisar se este consumo é reduzido sem comprometer significativamente o desempenho e a exatidão do *solver*.

5. ESTRATÉGIAS PARA ESTUDO DE EFICIÊNCIA ENERGÉTICA EM SELAS VERIFICADOS

Este capítulo apresenta as estratégias utilizadas para o estudo do comportamento de um *solver* de SELAs, FastPILSS, com o uso da técnica DVFS (*Dynamic Voltage and Frequency Scaling*). Este estudo tem com o objetivo avaliar o quanto a técnica melhora a eficiência energética, e qual seu impacto no desempenho da solução paralela e na exatidão dos resultados. Nesse contexto foram desenvolvidas estratégias para a definição de frequência em SELAS Verificados.

O estudo foi dividido em três etapas. A primeira etapa descrita na Seção 5.1 é a etapa de redução da frequência durante a execução do *solver* de Zimmer. Na segunda etapa, apresentada na Seção 5.2, a frequência é reduzida em pontos isolados. Por fim, a Seção 5.3 apresenta a redução de frequência em conjuntos de pontos do *solver*.

5.1 Definição da Frequência em um Único Ponto

Nesta etapa foi realizado um experimento com o objetivo de alterar a frequência dos processadores durante toda a execução do *solver* FastPILSS. Foram realizados dois tipos de testes. Inicialmente o valor do governador de frequência é alterado com objetivo de utilizar a máquina em seu máximo desempenho (*performance*). Em seguida, o governador da frequência foi alterado para o valor mínimo (*powersave*), e os mesmos testes foram realizados.

Esta estratégia foi elaborada com a finalidade de obter os valores médios de consumo de energia durante a execução do *solver* FastPILSS com o governador em *powersave*, e também os valores médios com o governador em *performance* para as matrizes testadas. Desta forma, é possível avaliar o impacto da alteração de frequência no comportamento do *solver* e assim analisar se há redução no consumo de energia. Além disso, objetiva-se apresentar o impacto da redução da frequência no tempo de execução do *solver*, cuja análise auxiliará no desenvolvimento de novos experimentos.

De acordo com Castillo [6], ao reduzir a frequência em aplicações que utilizam intensivamente o processador, o aumento no tempo de execução é tão grande que torna o consumo de energia maior em relação ao consumo de energia da execução com frequência no máximo. Já quando se reduz a frequência em aplicações que acessam intensamente a memória, Castillo indica que há redução no consumo de energia. Como o *solver* utiliza uma ampla quantidade de funções distintas, não é possível classificá-la como pertencente ao grupo de uso intensivo de processador (*CPU-bounded*) ou ao uso intensivo de memória (*memory-bounded*). Assim, os valores obtidos nesta etapa servirão de base para as etapas futuras, já que os resultados permitirão avaliar se a redução da frequência diminuirá de fato o consumo de energia.

5.2 Definição da Frequência em Pontos Isolados

Após realizar testes alterando a frequência do *solver*, optou-se por dividi-los em passos com base na organização dos algoritmos descrita na Subseção 3.2, apenas com a inclusão da alocação das matrizes e variáveis na memória (Passo 1). Os passos dos *solver* foram definidos como segue:

- **Passo 1:** alocação;
- **Passo 2:** cálculo da inversa;
- **Passo 3:** cálculo do x ;
- **Passo 4:** cálculo do z ;
- **Passo 5:** cálculo do C ;
- **Passo 6:** verificação.

Neste trabalho, optou-se por não utilizar o tempo de alocação, pois o foco do trabalho é o consumo de energia em operações que solucionam SELAs, e não no tempo de alocação de memória. A frequência em cada passo foi alterada isoladamente para o valor mínimo de consumo (*powersave*), enquanto os outros passos permaneciam no máximo. Assim, identificam-se os passos em que processador é utilizado intensivamente. Essa estratégia funciona da seguinte forma: quando a frequência do Passo 2 é alterada para o valor mínimo (*powersave*), a execução dos próximos passos é realizada com a frequência no máximo (*performance*). Depois, é realizado outro experimento onde apenas a frequência do Passo 3 é reduzida e as frequência dos outros passos, inclusive do Passo 2, permanecem no máximo (*performance*). Este procedimento também é realizado nos passos seguintes (Passo 4, Passo 5 e Passo 6).

Estes experimentos foram realizados com o objetivo de analisar o consumo de energia e o impacto no desempenho da alteração de frequência com o governador *powersave* em cada passo isoladamente. Essa estratégia permitirá a avaliação do impacto destas alterações durante a execução do *solver* em cada passo e também será possível comparar os resultados obtidos com os de experimentos anteriores. Também será possível avaliar em qual característica o passo se aproxima (uso intensivo de processador ou uso intensivo de memória). Além disto, será possível detectar em quais passos a redução de frequência com o governador diminui de fato o consumo de energia e quais passos a redução de frequência não contribui para a diminuição do consumo de energia no *solver*.

É de especial interesse observar o comportamento do Passo 5, pois utiliza a função `pdgemm` da ScaLAPACK, analisada por Castillo em [6]. Neste trabalho, Castillo identificou na função `pdgemm` características de uso intensivo de processador (*CPU-bound*), o que justificou o aumento do tempo de execução usando o governador *powersave* o que gera um aumento no consumo de energia. A

comparação dos resultados obtidos na alteração de frequência do Passo 5 com os resultados do trabalho de Castillo enriquecerá as análises dos resultados.

O Passo 2, além de utilizar algumas funções do C-XSC, utiliza a função `pdgetrf`. Já o Passo 3 ao invés de utilizar a função `pdgemv` para a multiplicação entre matrizes e vetores, o *solver* FastPILSS utiliza iterações com funções de acumuladores da biblioteca C-XSC. Isso também acontece para o cálculo do Z. No Passo 6, da verificação, também são utilizados estes acumuladores.

Castillo também analisou o comportamento da função `pdgemv`. Neste trabalho Castillo identificou que a função `pdgemv` utiliza intensivamente a memória (*memory-bound*) e, por isso, justificou a redução do consumo de energia em *powersave*. Entretanto, o *solver* de Zimmer optou pela utilização de iterações e acumuladores do C-XSC nos Passos 3 e 4. Acredita-se que a comparação destas diferenças também enriquecerá a análise dos resultados obtidos nesta etapa de definição de frequência em pontos isolados. Acredita-se que teoricamente o comportamento deve ser semelhante ao da função `pdgemv`.

5.3 Definição da Frequência em Pontos Diversos

Nesta etapa optou-se por realizar testes alterando a frequência com o governador em *powersave* em determinados conjuntos de passos, e não apenas um só como foi realizado anteriormente. A escolha de agrupamento dos passos se baseou em dois critérios: tempo de execução e proximidade. O critério de tempo de execução, foi escolhido uma vez que quando o impacto dos passos com uma porcentagem maior no tempo total da execução é significativo, o seu tempo influencia no consumo de energia de toda a aplicação. O critério de proximidade entre eles foi escolhido pois há custo para alterar a frequência nos passos, então optou-se pela alteração em agrupamentos de passos próximos. Os agrupamentos escolhidos e suas respectivas descrições são apresentados a seguir:

- **Passos 2 e 3:** este conjunto foi organizado com um dos passos de tempo de execução significativo (Passo 2) e o próximo passo. A finalidade dessa organização é a de analisar se há melhora na redução do consumo de energia em relação aos testes com os passos isolados da segunda etapa. Acredita-se que com o acréscimo do Passo 3, aproveita-se o custo da alteração do governador sem aumentar o custo que se teria ao realizá-los separadamente;
- **Passos 2, 3 e 4:** este conjunto foi organizado com o Passo 2 (tempo de execução significativo) e os dois próximos passos, e então analisar se há melhora no consumo de energia em relação aos testes anteriores. Cabe salientar que os Passos 3 e 4 utilizam funções da biblioteca C-XSC e iterações para multiplicação de matriz por vetor. Castillo definiu esta operação como *memory bound* quando testou a ScaLAPACK;
- **Passos 2 e 5:** o conjunto foi organizado com o objetivo de reunir os passos significativos em relação ao tempo de execução total. Desta forma, é possível comparar o impacto da

alteração de frequência para o valor mínimo apenas nestes passos que demandam mais tempo de processamento com os outros testes. Reiterando que o Passo 5 utiliza a função *pdgemm* e o Passo 2 utiliza a função *pdgetrf* da ScaLAPACK;

- **Passos 5 e 6:** este conjunto foi organizado com um dos passos significativos (Passo 5). O objetivo é analisar se há melhora no consumo de energia em relação aos testes anteriores, pois acredita-se que com o acréscimo do Passo 6 aproveita-se o custo de apenas uma alteração do governador para diminuir a frequência de dois passos. Outra razão da escolha deste conjunto é que de acordo com Castillo [6], a alteração da frequência para o *powersave* apenas no Passo 5 não reduziria o consumo de energia, pois utiliza a função *pdgemm* que foi classificado como *CPU-bound*.
- **Passos 4, 5 e 6:** este conjunto foi organizado com o Passo 5 (tempo de execução significativo) e dois passos próximos de menor complexidade e também para analisar se há melhora na redução do consumo de energia em relação aos testes anteriores. Acredita-se que com os acréscimos dos dois passos próximos, pode-se aproveitar o custo de uma alteração para diminuir a frequência de três passos próximos sem aumentar o custo. Lembrando que de acordo com Castillo apenas a alteração no Passo 5 não reduziria o consumo de energia;
- **Passos 3, 4 e 6:** organizado com o objetivo de reduzir a frequência dos passos com menor tempo de execução em relação ao tempo total da aplicação. Desta forma é possível analisar o impacto destas alterações na frequência em comparação com os testes anteriores. Além disso, contém os passos que utilizam os acumuladores da biblioteca C-XSC com iterações. Será possível observar se a alteração do governador em *powersave* neste conjunto reduz o consumo de energia do *solver*. Também com a exclusão dos Passos 2 e 5, que segundo Castillo são *CPU-bound*, buscando agrupar apenas os passos que seriam *memory-bound*.

Esta estrutura permite uma análise aprofundada do uso da técnica DVFS no *solver* de Zimmer, além de buscar a alteração em passos isolados ou o conjunto de passos mais eficiente em consumo de energia cujo tempo de execução não foi comprometido significativamente.

6. AVALIAÇÃO DAS ESTRATÉGIAS PROPOSTAS

Este capítulo apresenta os resultados dos testes e a avaliação das estratégias de uso da técnica DVFS no *solver* FastPILSS. Na Seção 6.1 é apresentado o cenário em que os experimentos foram realizados. A Seção 6.2 apresenta os resultados de exatidão. A Seção 6.3 apresenta os resultados de desempenho e consumo de energia. Por fim, na Seção 6.4, são apresentadas as considerações finais do capítulo.

6.1 Cenário

Os experimentos foram realizados em um servidor DELL PowerEdge T310, com processador Intel Xeon X3470 e 8GB de memória RAM. Os valores de consumo de energia foram obtidos com o multímetro digital EZ-735.

Os testes foram realizados com Sistema Operacional Ubuntu Server 12.04.2. Foram instaladas todas as bibliotecas necessárias para utilização do *solver*: MPICH versão 1.2.7, Scalapack versão 2.02, Lapack versão 3.4.2, Blas, Blacs, PBLAS e a C-XSC na versão 2.5.3.

O pacote `cpufrequtils` (versão 007) foi instalado para possibilitar alteração da frequência dos processadores. Com a instalação do pacote, a frequência do processador pode ser alterada para os governadores *powersave*, *ondemand* e *performance*.

Foram definidos testes com entradas para as matrizes geradas pelas equações 6.1 e 6.2, correspondentes às Gen1 e DingDong.

$$Gen1_{i,j} = \sqrt{2.0/(n+1)} * \sin((i + *j * M_PI)/(n+1)); \quad (6.1)$$

$$DingDong_{i,j} = 0.5/((n-i-j)+1.5); \quad (6.2)$$

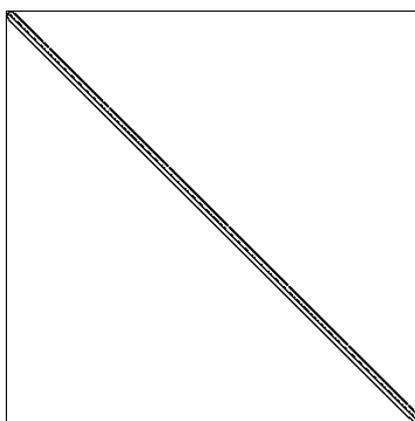
Além destas matrizes, foram utilizadas 6 matrizes do Matrix Market [35]:

- as matrizes Fidapm29, Fidap019 e Fidapm37 pertencem a um pacote de testes de equações da área de Fluidodinâmica Computacional (C.F.D.) [39], [38], [40].
- a matriz Bcsstk17 surgiu de problemas decorrentes de aplicações do código de engenharia estrutural GT-Strudl. Eles foram coletados como um conjunto de *benchmarks* para fatoração em computadores de memória limitada (por exemplo, a série CDC 6600) [36].
- a matriz Dw8192 é criada a partir de problemas de canais em guias de ondas dielétricos em aplicações de circuitos integrados [37].
- a matriz Utm5940 vem da coleção de SPARSKIT é originária de aplicações da física nuclear [41].

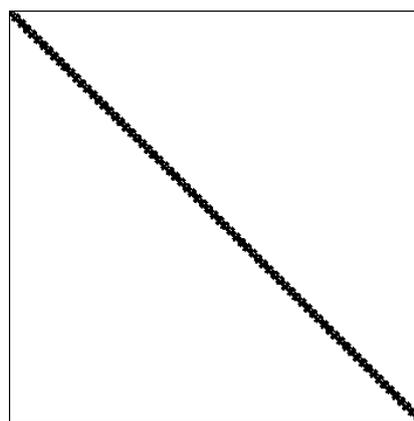
Para a realização dos testes, além da matriz A , é necessário definir o valor de b . Este valor de entrada foi definido para todos os casos testados, o valor foi definido como $b[i]=1$, O índice i varia de um a n . É importante ressaltar que os valores de entrada para A e b não são valores intervalares, porém os resultados do *solver* de Zimmer são em números intervalares, que contém a resposta correta.

Tabela 6.1 – Características das matrizes de entrada.

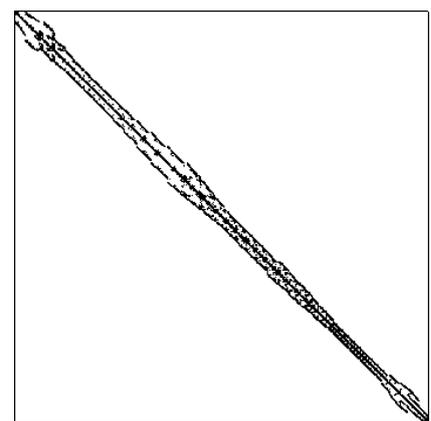
Matriz	Ordem (n)	Número de condicionamento	Esparsa/Densa	Real/Gerada	Área de Aplicação	% não nulos
Gen1	15.000	3,64E+04	Densa	Gerada	Teste	-
DingDong	15.000	2,86E+03	Densa	Gerada	Teste	-
Fidamp29	13.668	7,32E+05	Esparsa	Real	C.F.D.	0,099
Fidap019	12.005	3.92E+12	Esparsa	Real	C.F.D.	0,018
Bcsstk17	10.974	1.42E+10	Esparsa	Real	Engenharia Estrutural	0,018
Fidapm37	9.152	3.04E+10	Esparsa	Real	C.F.D	0,914
Dw8192	8.192	1.13E+07	Esparsa	Real	Engenharia Elétrica	0,062
Utm5940	5.940	3.68E+09	Esparsa	Real	Física Nuclear	0,237



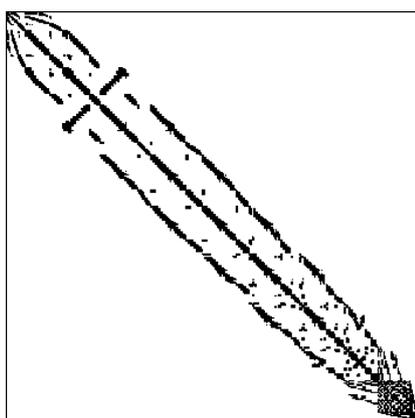
(a) Fidapm29



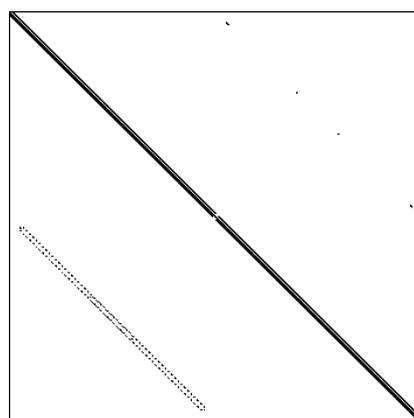
(b) Fidap019



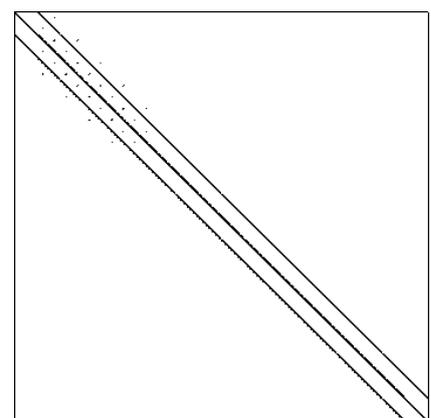
(c) Bcsstk17



(d) Fidapm37



(e) Dw8192



(f) Utm5940

Figura 6.1 – Localização dos valores não nulos das matrizes

A Tabela 6.1 mostra informações sobre o número de condicionamento das matrizes de entrada utilizadas nos experimentos, bem como suas respectivas ordens e informações sobre as características destas matrizes.

Para melhor exemplificar a posição dos valores não nulos das matrizes esparsas, a Figura 6.1 contém as imagens com a localização dos valores não nulos das matrizes esparsas. Pode-se notar que a maioria dos valores não nulos concentram-se próximos à diagonal principal

Por questões de acesso limitado a recursos físicos, neste caso específico o multímetro, foram realizadas médias de três execuções para todos os testes com todas as matrizes. Os valores apresentados foram obtidos com base na média destas execuções.

6.2 Exatidão

Esta seção apresenta os resultados de exatidão dos experimentos descritos na Seção 6.1 e suas respectivas análises. O objetivo é verificar se o uso da técnica DVFS causa algum impacto nos resultados numéricos. As Tabelas 6.2 e 6.3 apresentam os resultados intervalares do *solver* de Zimmer sem o uso da técnica DVFS, e ao lado os resultados do mesmo *solver* com o uso da técnica DVFS para as matrizes: Gen1 e Fidamp29. Os demais resultados do *solver* para as matrizes de entrada DingDong, Fidap019, Bcsstk17, Fidapm37, Dw8192 e Utm5940 podem ser vistos no Apêndice B.

Tabela 6.2 – Resultados do *solver* de Zimmer para a Matriz Gen 1 com 4 processos.

x[]	Zimmer	Zimmer com DVFS
x[0]	[1.102694541460449E+002, 1.102694541460450E+002]	[1.102694541460449E+002, 1.102694541460450E+002]
x[1]	[-8.203213255146324E-015, 1.666269437753182E-014]	[-8.203213255146324E-015, 1.666269437753182E-014]
x[2]	[3.675648364061169E+001, 3.675648364061174E+001]	[3.675648364061169E+001, 3.675648364061174E+001]
x[3]	[-9.416142615602100E-015, 1.553291826926338E-014]	[-9.416142615602100E-015, 1.553291826926338E-014]
x[4]	[2.205388889468306E+001, 2.205388889468310E+001]	[2.205388889468306E+001, 2.205388889468310E+001]
x[5]	[-8.225056918732845E-015, 1.677263313740173E-014]	[-8.225056918732845E-015, 1.677263313740173E-014]
x[6]	[1.575277640011220E+001, 1.575277640011224E+001]	[1.575277640011220E+001, 1.575277640011224E+001]
x[7]	[-1.126736918785614E-014, 1.376506452609243E-014]	[-1.126736918785614E-014, 1.376506452609243E-014]
x[8]	[1.225215798932723E+001, 1.225215798932726E+001]	[1.225215798932723E+001, 1.225215798932726E+001]
x[9]	[-7.875031459591726E-015, 1.718380513444427E-014]	[-7.875031459591726E-015, 1.718380513444427E-014]

Tabela 6.3 – Resultados do *solver* de Zimmer para a Matriz Fidamp29 com 4 processos.

x[]	Zimmer	Zimmer com DVFS
x[0]	[1.530081256362627E+001, 1.530081256362800E+001]	[1.530081256362627E+001, 1.530081256362800E+001]
x[1]	[-6.208148168356456E-001,-6.208148168356214E-001]	[-6.208148168356456E-001,-6.208148168356214E-001]
x[2]	[1.579105158878873E+001, 1.579105158879001E+001]	[1.579105158878873E+001, 1.579105158879001E+001]
x[3]	[-1.467169571729604E+000,-1.467169571729538E+000]	[-1.467169571729604E+000,-1.467169571729538E+000]
x[4]	[1.447675288907328E+001, 1.447675288907424E+001]	[1.447675288907328E+001, 1.447675288907424E+001]
x[5]	[-2.316536672396893E+000,-2.316536672396800E+000]	[-2.316536672396893E+000,-2.316536672396800E+000]
x[6]	[1.260962590807345E+001, 1.260962590807429E+001]	[1.260962590807345E+001, 1.260962590807429E+001]
x[7]	[-3.132407229768991E+000,-3.132407229768878E+000]	[-3.132407229768991E+000,-3.132407229768878E+000]
x[8]	[1.038044496201669E+001, 1.038044496201753E+001]	[1.038044496201669E+001, 1.038044496201753E+001]
x[9]	[-3.895651110979420E+000,-3.895651110979293E+000]	[-3.895651110979420E+000,-3.895651110979293E+000]

Ao analisar os resultados intervalares do *solver* apresentados nas Tabelas 6.2 e 6.3 constatou-se que a utilização da técnica DVFS para o *solver* de Zimmer (que utiliza a biblioteca C-XSC) não interferiu na exatidão dos resultados para todas as matrizes testadas. Esta confirmação torna possível a realização de estudos com alterações de frequência no *solver* sem modificar o resultado numérico.

Como a técnica DVFS não interfere na garantia numérica e exatidão dos resultados, é possível estudar maneiras de reduzir o consumo de energia, buscando não impactar significativamente no tempo de execução do *solver* de Zimmer.

6.3 Consumo de Energia e Desempenho

Como exemplificado anteriormente o consumo de energia e o tempo de execução do *solver* estão diretamente relacionados. Esta seção apresenta os resultados de consumo e tempo de execução obtidos pela alteração da frequência em diversos passos do *solver* de Zimmer e suas respectivas análises.

6.3.1 Definição da Frequência em um Único Ponto

Inicialmente o governador do processador foi alterado para manter o valor máximo de frequência (*performance*). Na sequência, o governador foi alterado para manter o valor mínimo de frequência (*powersave*). Para avaliação do consumo de energia foi realizada a medição durante a execução de todas as matrizes na FastPILSS. A Tabela 6.4 apresenta a potência (P) obtida para os governadores *performance* e *powersave*. Além disso, é calculada a porcentagem da redução de potência, quando utilizado o governador *powersave*, em relação ao governador em *performance*.

Tabela 6.4 – Valores medidos de potência das matrizes

Solver		Zimmer			
Matrizes	Ordem	Performance (w)	Powersave (w)	Diferença	% da redução
Gen1	15.000	137,37	81,81	55,56	40,45
DingDong	15.000	134,93	80,91	54,02	40,03
Fidapm29	13.668	111,18	75,15	36,02	32,40
Fidapm019	12.005	119,65	74,98	44,68	37,34
Bcsstk17	10.974	119,69	74,94	44,74	37,38
Fidapm37	9.152	123,38	76,03	47,35	38,38
Dw8192	8.192	125,53	79,01	46,52	37,06
Utm5940	5.940	116,59	71,88	44,72	38,35

A unidade de potência (P) é *watts*, também conhecida com a representação *joule/segundo*. Sendo assim, para obter o consumo de energia (em *joule*), multiplica-se o valor da potência em watts pela quantidade de segundos executados. Assim, com base na tabela observou-se que os valores de potência são similares para as matrizes densas. Porém, entre as matrizes esparsas há mais disparidade. Nota-se também que os valores de potência (*watts*) apresentados para as matrizes geradas (densas) no *solver* de Zimmer são mais altos do que os valores para as matrizes reais (esparsas). Com base nessas observações presume-se que a característica da matriz influencia no consumo de energia do *solver*.

Baseados nos valores de potência e no tempo de execução de cada matriz, foram obtidos os resultados do consumo de energia do *solver* de Zimmer. Os valores de tempo de execução e consumo de energia (em *joules*) são apresentados nas Tabelas 6.6 e 6.5, respectivamente.

Tabela 6.5 – Tempo de execução do *solver* de Zimmer.

Solver		Zimmer		
Matrizes	Ordem	Tempo Perf. (s)	Tempo Pow. (s)	(% do aumento de tempo)
Gen1	15.000	4.354,22	7.766,63	78,37
DingDong	15.000	4.350,57	7.785,16	78,95
Fidapm29	13.668	54,50	107,70	97,61
Fidap019	12.005	38,33	75,01	95,70
Bcsstk17	10.974	28,70	56,78	97,79
Fidapm37	9.152	41,35	80,49	94,66
Dw8192	8.192	28,66	51,54	79,84
Utm5940	5.940	15,67	34,88	122,67

Ao analisar a Tabela 6.5, é possível notar um comportamento padronizado com relação a porcentagem de aumento no tempo de execução do *solver* quando é usado *powersave* para as matrizes geradas (densas). Já para as matrizes reais (esparsas) há maior disparidade entre essa porcentagem. Acredita-se que isto ocorre devido a diferença de características destas matrizes como, por exemplo, a quantidade de valores não nulos em relação ao tamanho e também a localidade destes valores.

Tabela 6.6 – Consumo de energia e tempo de execução do *solver* de Zimmer

Solver		Zimmer		
Matrizes	Ordem	Consumo Perf. (J)	Consumo Pow. (J)	(% do aumento de consumo)
Gen1	15.000	598.144,67	635.381,29	6,23
DingDong	15.000	587.033,37	629.926,70	7,31
Fidapm29	13.668	6.059,50	8.094,33	33,58
Fidapm019	12.005	4.586,17	5.624,08	22,63
Bcsstk17	10.974	3.435,53	4.254,94	23,85
Fidapm37	9.152	5.101,39	6.119,24	19,95
Dw8192	8.192	3.597,65	4.072,43	13,20
Utm5940	5.940	1.826,58	2.507,38	37,27

Com base nos valores de consumo obtidos para todas as matrizes, na Tabela 6.6 é possível perceber que a alteração do governador para o *powersave* durante toda a execução da FastPILSS não diminuiu o consumo de energia em relação a alteração do governador para *performance*. Neste caso não houve redução pois o tempo de execução em *powersave* foi bem maior que o consumo em *performance* e devido a isso a utilização do governador em *powersave* não foi vantajosa.

6.3.2 Definição da Frequência em Pontos Isolados

Para calcular o consumo de energia de cada passo foram realizadas medições em todos os passos para todas as matrizes. As Tabelas 6.7 e 6.8 apresentam os valores de potência em watts de todos os passos isoladamente, em *performance* e *powersave* para todas as matrizes.

As Figuras 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8 e 6.9 apresentam a comparação da porcentagem de tempo e consumo em relação ao consumo com o governador *performance*, para as matrizes Gen1, DingDong, Fidapm29, Fidap019, Bcsstk17, Fidapm37, Dw8192 e Utm5940, respectivamente.

Ao observar as Figuras 6.2 e 6.3, nota-se que o comportamento dos resultados das matrizes Gen1 e DingDong em relação ao aumento da porcentagem de tempo em cada passo isolado é semelhante. Estima-se que este comportamento se assemelha, pois ambas as matrizes são densas

Tabela 6.7 – Potência (watts) dos passos em *performance*.

Solver		Zimmer				
Matrizes	Ordem	Passo 2	Passo 3	Passo 4	Passo 5	Passo 6
Gen1	15.000	138,9834	131,8966	135,2233	138,8774	65,0654
DingDong	15.000	137,0442	140,7478	130,3274	136,9068	63,5723
Fidapm29	13.668	112,6960	124,4177	132,7573	103,4838	124,2483
Fidapm019	12.005	113,4558	123,0207	127,5503	123,2081	107,0356
Bcsstk17	10.974	113,6968	121,6025	127,8043	124,0155	119,8880
Fidapm37	9.152	121,9149	122,9360	128,2065	124,7775	130,1750
Dw8192	8.192	130,3287	128,2700	138,0490	124,1848	87,6300
Utm5940	5.940	116,1704	122,6820	122,6820	124,2060	108,2675

Tabela 6.8 – Potência (watts) dos passos em *powersave*.

Solver		Zimmer				
Matrizes	Ordem	Passo 2	Passo 3	Passo 4	Passo 5	Passo 6
Gen1	15.000	83,7589	74,3005	75,1263	81,2771	61,6091
DingDong	15.000	83,0170	76,5334	73,8124	80,2336	61,0772
Fidapm29	13.668	73,8427	73,5330	74,1934	78,9940	75,5015
Fidapm019	12.005	73,5729	72,8504	74,2406	78,2205	74,9935
Bcsstk17	10.974	73,9087	73,9987	74,0410	77,5596	70,6755
Fidapm37	9.152	76,5254	73,4060	76,2508	76,1827	67,1195
Dw8192	8.192	79,0267	75,8613	75,4803	81,2959	79,6290
Utm5940	5.940	70,5764	72,3265	72,3265	77,4954	78,6765

e possuem número de condicionamento aproximado. O comportamento de ambas as matrizes em relação ao consumo de energia também é semelhante, é possível notar uma redução no consumo de energia de no até -3,29% com a redução da frequência no Passo 2 para ambas matrizes. Verificou-se que neste caso a diferença de potência do governador em *performance* com o *powersave* no Passo 2 é em torno de 55 watts (representando uma redução de aproximadamente 40% do valor em *performance*). Nota-se também que o uso de *powersave* no Passo 5 impactou mais no consumo de energia e no tempo de execução se comparado aos demais passos isolados. Entretanto, apesar do tempo de execução ter aumentado em torno de 56%, o consumo aumentou no máximo 8,49%, isso indica que se a diferença de potência entre o governador *powersave* e *performance* for maior, há tendência de reduzir o consumo de energia no Passo 5. Verificou-se que neste passo, a diferença entre a potência dos governadores foi em torno de 57 watts (representando uma redução de aproximadamente 41% do valor em *performance*).

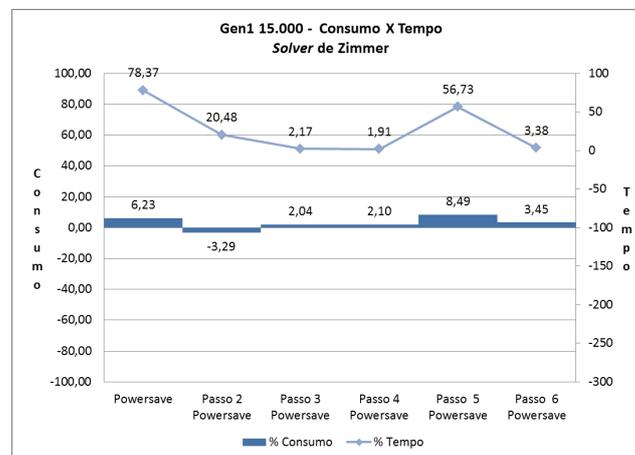


Figura 6.2 – Comparação entre consumo e tempo para a matriz de entrada Gen1.

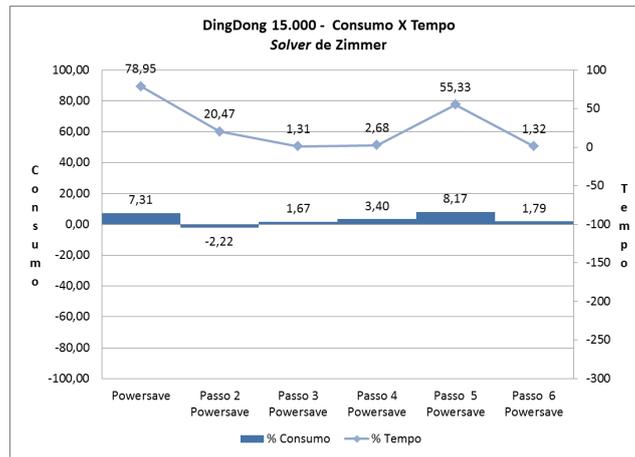


Figura 6.3 – Comparação entre consumo e tempo para a matriz de entrada DingDong.

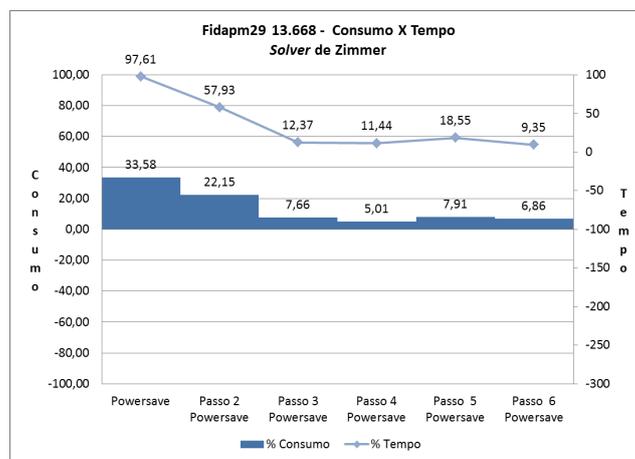


Figura 6.4 – Comparação entre consumo e tempo para a matriz de entrada Fidapm29

A Figura 6.4 apresenta os percentuais de consumo e tempo de execução tendo como entrada a matriz Fidapm29. Ao examinar a figura, pode-se verificar que o Passo 2 apresenta um aumento na porcentagem de tempo de execução se comparado ao Passo 5, o contrário do observado para as matrizes densas. Ao avaliar o consumo de energia percebe-se que ao alterar o governador para o *powersave*, durante toda a execução, o consumo é maior se comparado às alterações de frequência realizadas nos passos isolados. O Passo 2 consumiu mais energia dentre as alterações nos passos. Dentre os passos restantes, apesar do Passo 5 ter apresentado porcentagem de tempo de execução maior que os demais passos, não consumiu na mesma proporção e apresentou um percentual de consumo de energia próximo dos demais passos. O que indica que, da mesma forma que observou-se para as matrizes densas, há a tendência de reduzir o consumo de energia no Passo 5 se a diferença entre a potência dos governadores em média for maior. Verificou-se que a diferença de potência do governador *performance* com o *powersave* no Passo 5 foi em torno de 24 watts (representa uma redução de aproximadamente 23% em relação ao valor em *performance*).

A Figura 6.5 apresenta os resultados tendo como entrada a matriz Fidap019. Ao analisar a figura, pode-se perceber que a interferência no tempo de execução e no consumo de energia na alteração de frequência no Passo 2 é maior do que nos outros passos, como ocorreu com a

matriz Fidapm29. Como pode-se observar, a alteração de frequência não reduziu o consumo de energia para a maioria dos passos. No entanto, o aumento do consumo para os Passos 3, 4, 5 e 6 foi de no máximo 3,84%. Isso indica que se a diferença de potência for maior, é possível ter redução no consumo de energia. Cabe ressaltar que no servidor de testes, a diferença de consumo da matriz Fidap019 com o governador *performance* em relação ao *powersave* é em média de 44 watts (aproximadamente 36% do valor em *performance*).

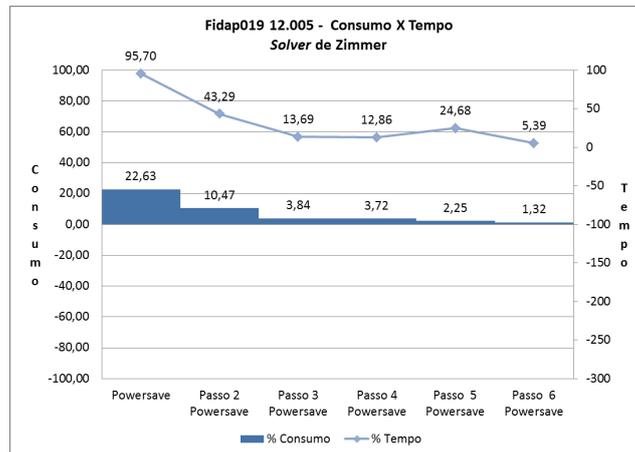


Figura 6.5 – Comparação entre consumo e tempo para a matriz de entrada Fidap019

A Figura 6.6 mostra os resultados do *solver* tendo como entrada a matriz Bcsstk17. É possível notar um aumento no percentual de tempo de execução na alteração de frequência no Passo 2. Apesar de se aproximar do percentual do Passo 5, ainda se manteve alto em relação ao percentual dos demais passos também para a matriz Bcsstk17. Observou-se também que apesar de haver uma diferença significativa no Passo 5 em relação aos Passos 3, 4 e 6, no que se refere ao percentual de tempo de execução, o consumo se manteve semelhante e até menor.

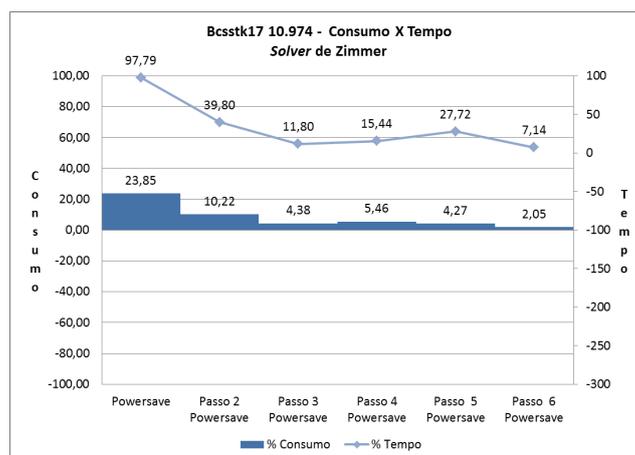


Figura 6.6 – Comparação entre consumo e tempo para a matriz de entrada Bcsstk17

Na Figura 6.7 percebe-se um comportamento semelhante da matriz Fidapm37 em relação às demais matrizes esparsas. Ao alterar a frequência no Passo 2, pode-se observar um aumento no percentual de consumo de energia e de tempo de execução se comparado às alterações realizadas

nos demais passos. Dentre as alterações de frequência nos passos restantes (Passos 3, 4, 5 e 6), a alteração realizada no Passo 5 manteve o percentual de tempo de execução e consumo de energia alto, da mesma maneira que a matriz Fidapm29. Os Passos 3, 4 e 6 apresentaram aumento no percentual de consumo de energia de no máximo 2,53%. Nota-se que o consumo de energia do Passo 5 apresentou diferença maior de 3% em relação aos Passos 3, 4 e 6. Esse comportamento só foi observado nas matrizes densas. Esse comportamento é atribuído ao fato de que dentre as matrizes reais, a matriz Fidapm37 apresenta maior porcentagem de valores não nulos.

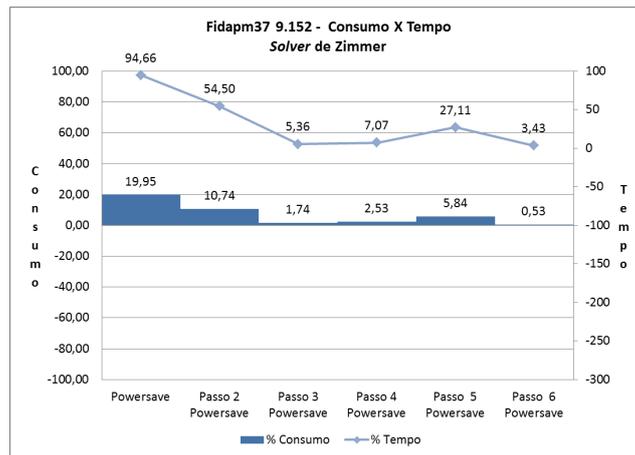


Figura 6.7 – Comparação entre consumo e tempo para a matriz de entrada Fidapm37

Como ilustrado na Figura 6.8, para a matriz Dw8192 é possível identificar um comportamento às demais matrizes, onde o percentual de tempo de execução e consumo de energia do Passo 2 é maior em relação aos demais passos. Constatou-se também que o aumento no consumo de energia dos Passos 3, 4, 5 e 6 foi de no máximo 4,73%. Assim, do mesmo modo que para as demais matrizes esparsas, se a diferença de potência entre o *powersave* e o *performance* fosse maior que a diferença de 42 watts (percentual de 34% em relação ao valor da potência em *performance*), possivelmente haveria redução no consumo de energia no Passo 5.

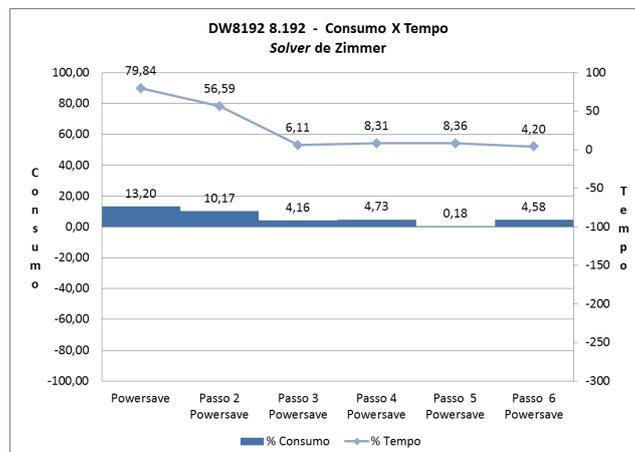


Figura 6.8 – Comparação entre consumo e tempo para a matriz de entrada Dw8192

Por fim, a Figura 6.9 apresenta os resultados para a matriz Utm5940. Da mesma forma que ocorre para as demais matrizes reais, percebe-se a alta interferência do Passo 2 no tempo de

execução em relação aos demais passos, quando há alteração de frequência. Nota-se que, neste caso, o aumento da porcentagem de tempo de execução em *powersave* ultrapassou o valor de 100% quando usado *powersave* em toda a execução. Também é possível identificar que o valor do percentual do tempo de execução do Passo 2 ficou próximo de 100%. Acredita-se que este comportamento pôde ser observado apenas para esta matriz devido à sua ordem ser a menor dentre as matrizes testadas. Por isso, a alteração de frequência influenciou muito no consumo de energia e no tempo de execução para o Passo 2, passo este significativo no tempo de processamento. Já ao alterar a frequência nos demais passos, pode-se detectar que o percentual de aumento no consumo de energia foi de no máximo 5,09% no Passo 4, e no tempo de execução foi de no máximo 12,79% no Passo 5.

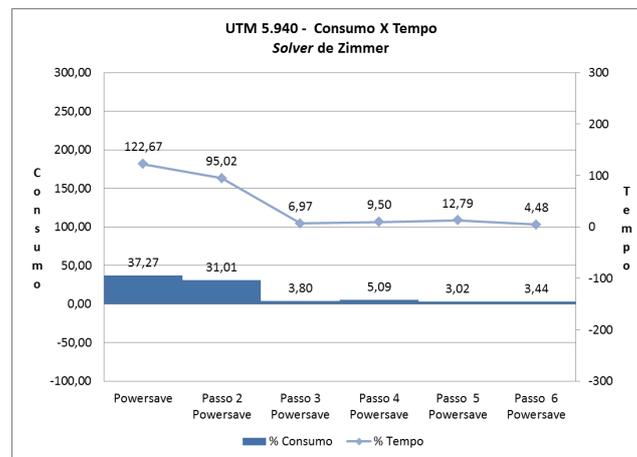


Figura 6.9 – Comparação entre consumo e tempo para a matriz de entrada Utm5940

Analisando os resultados das matrizes em relação à porcentagem de aumento no tempo de execução, pode-se separá-las em dois grupos: o grupo das matrizes densas e o grupo das matrizes esparsas. No grupo das matrizes densas, que possuem número de condicionamento aproximado, a alteração no Passo 5 é significativa em relação aos demais passos no que se refere ao aumento no tempo de execução. Em seguida tem-se a alteração no Passo 2, que é significativa se comparada aos passos restantes (Passos 3, 4 e 6). Entretanto, apesar de o Passo 2 ter um significativo aumento de tempo de execução, há redução no consumo de energia para ambas as matrizes. Por outro lado, para as matrizes esparsas a alteração de frequência no Passo 2 apresenta valores altos de aumento no tempo de execução se comparada aos demais passos. Já dentre os passos restantes, a alteração no Passo 5 apresenta aumento significativo no tempo de execução.

Ao analisar as matrizes esparsas, nota-se que a única matriz de entrada que não apresentou aumento acima de 90% no percentual de tempo de execução foi a matriz Dw8192. Estima-se que isto ocorreu, devido à localização dos valores não nulos. Na matriz Dw8192 há valores próximos à diagonal principal e também distantes, diferente das demais matrizes esparsas, em que os valores não nulos estão localizados bem próximos à diagonal principal. Então, como ocorreu nas matrizes densas, a porcentagem de aumento no tempo de execução foi próximo de 79%. Observou-se também que as matrizes Bcsstk17, Fidapm37 e Fidap019, que apresentam número de condicionamento acima do Expoente E^{+09} (E^{+10} e E^{+12}), apresentaram porcentagem de aumento no tempo de execução

entre 27% e 24% para o Passo 5. Já as demais matrizes esparsas, com condicionamento abaixo do expoente E^{+10} (E^{+05} , E^{+07} e E^{+09}) apresentam porcentagem de aumento no tempo de execução abaixo de 20%.

O servidor onde foram realizados os testes possui apenas um processador Intel Xeon X3470, com diferença média entre os consumos dos governadores em *performance* e *powersave* de 46 watts (aproximadamente 36% em relação ao valor da potência em *performance*). Este valor é resultado da média das diferenças entre os valores de consumo em *performance* e *powersave* para cada matriz de entrada.

Para melhor ilustrar o comportamento das matrizes em relação à alteração de frequência em pontos isolados do *solver*, a Figura 6.10 apresenta os grupos de matrizes cujos comportamentos são semelhantes. A classificação é feita de acordo com o impacto no tempo de execução e no consumo de energia. Pode-se observar dois grupos relacionados ao impacto no tempo de execução: um grupo contém as matrizes esparsas e o outro grupo as matrizes densas. Já em relação ao impacto no consumo de energia, há três grupos: um grupo com as matrizes densas e dois grupos com matrizes esparsas. Foram definidos dois grupos de matrizes esparsas, pois como já foi mencionado, as matrizes Fidapm37 e Fidapm29 apresentaram algumas diferenças no impacto do consumo de energia ao se comparar com as outras matrizes testadas.

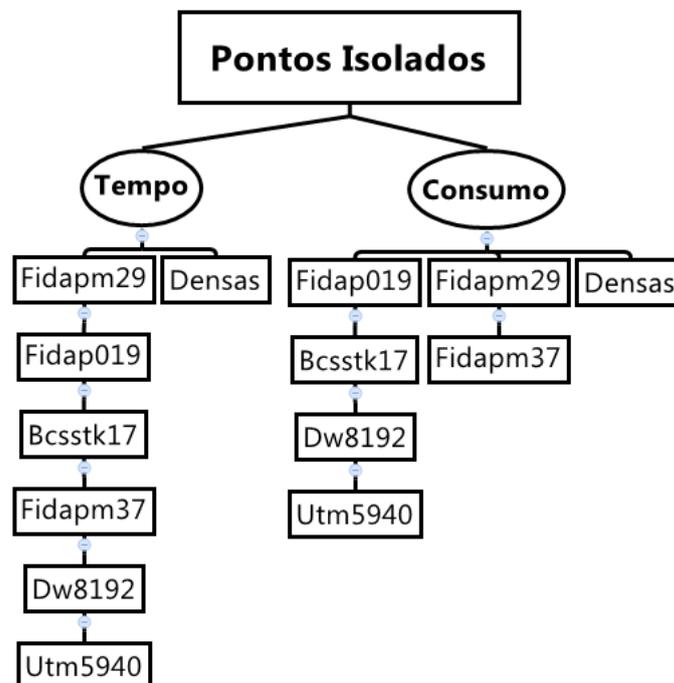


Figura 6.10 – Comparação entre as matrizes por tempo e por consumo

6.3.3 Definição de Frequência em Pontos Diversos

As Figuras 6.11, 6.12, 6.13, 6.14, 6.15, 6.16, 6.17 e 6.18 apresentam a comparação do tempo de execução e consumo de energia para a alteração da frequência em pontos diversos. As matrizes testadas são: Gen1, DingDong, Fidapm29, Fidap019, Bcsstk17, Fidapm37, Dw8192 e Utm5940. Os valores de consumo dos passos são apresentados nas Tabelas 6.7 e 6.8 da subseção 6.3.2 .

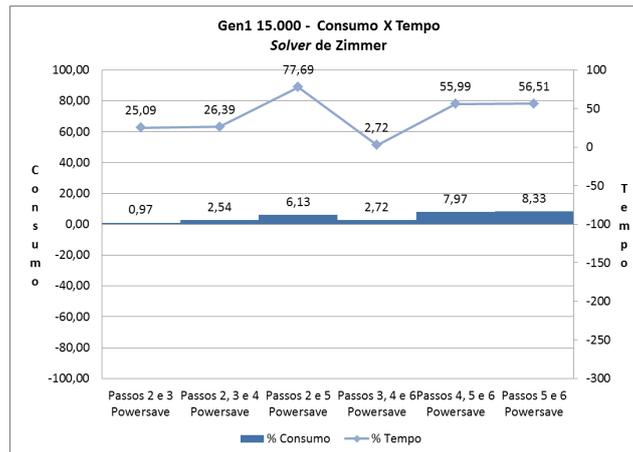


Figura 6.11 – Comparação entre consumo e tempo para a matriz de entrada Gen1

A Figura 6.11 apresenta a comparação do consumo de energia e do tempo de execução dos conjuntos de passos definidos. A matriz de entrada é a Gen1. Pode-se observar que o comportamento em relação ao percentual de tempo de execução do *solver* se manteve semelhante aos resultados desta matriz apresentados na Subseção 6.3.2. Sendo assim, o conjunto com os Passos 2 e 5 apresentou o maior percentual de tempo de execução. Como foi visto na Subseção 6.3.2, houve redução de consumo do Passo 2 na etapa de definição de frequência em pontos isolados. Já nesta etapa pode-se notar que os conjuntos que possuem o Passo 2 não apresentam redução do consumo de energia.

Na Figura 6.12 percebe-se que da mesma forma que ocorreu na primeira etapa de comparação da alteração de frequência em pontos isolados, o comportamento das alterações de frequência no tempo de execução e no consumo de energia da matriz DingDong se manteve semelhante ao da matriz Gen1. Isto acontece devido à semelhança entre as características das duas matrizes. É importante salientar também que o conjunto com os Passos 2 e 5 apresentou maior percentual de aumento no tempo de execução, assim como ocorreu com a matriz Gen1. Acredita-se que este comportamento se mantém para outras matrizes densas com número de condicionamento próximo. Cabe ressaltar que para ambas as matrizes o percentual de aumento de consumo na alteração de frequência no Passo 2 foi menor do que os percentuais das alterações em conjuntos que contêm o Passo 2.

Conforme ilustrado na Figura 6.13, os conjuntos que possuem o Passo 2 apresentam aumento significativo no percentual de tempo de execução (acima de 60%) e de consumo de energia

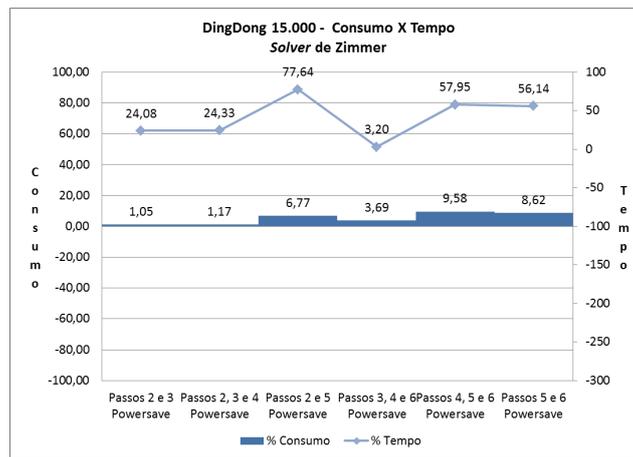


Figura 6.12 – Comparação entre consumo e tempo para a matriz de entrada DingDong

(acima de 19%), em que a entrada é a matriz Fidapm29. Dentre os conjuntos que continham o Passo 2, o conjunto que obteve maior percentual de aumento no tempo de execução e no consumo de energia foi o conjunto com os Passos 2 e 5. E entre os demais conjuntos que não continham o Passo 2, o que obteve maior percentual de aumento no tempo de execução e no consumo de energia foi o conjunto com os Passos 3, 4 e 6. Identificou-se também que entre os conjuntos, o que apresentou menor percentual de aumento de consumo e tempo de execução foi o conjunto de Passos 5 e 6. Na etapa anterior, verificou-se que se a diferença entre a potência dos governadores em *performance* e *powersave* fosse maior, a tendência seria reduzir o consumo de energia no Passo 5. Nesta etapa de conjuntos acredita-se que pode haver redução no conjunto com os Passos 5 e 6.

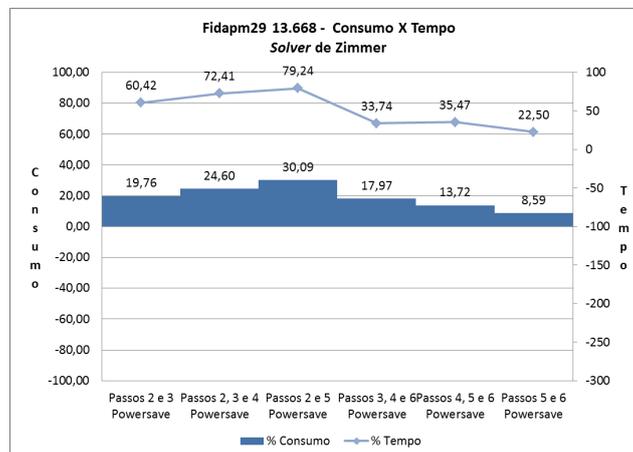


Figura 6.13 – Comparação entre consumo e tempo para a matriz de entrada Fidapm29

Ao observar a Figura 6.14, verifica-se que como a matriz Fidap019, os conjuntos de passos que contêm o Passo 2 apresentam valores de consumo de energia altos em relação aos outros conjuntos, assim como ocorreu com a matriz Fidapm29. Percebe-se que, assim como foi apresentado para as matrizes densas, o conjunto que apresentou maior valor no percentual de aumento no tempo de execução foi o conjunto com os Passos 2 e 5. No entanto, diferente do que ocorreu para as matrizes densas, este conjunto também apresentou consumo de energia superior aos outros conjuntos de passos. Como pode-se observar, a alteração de frequência não reduziu o consumo de energia

para a maioria dos passos. No entanto, o aumento do consumo para o conjunto de Passos 5 e 6 foi de apenas 3,52%. Conforme mencionando na descrição dos resultados obtidos na etapa de definição de frequência em pontos isolados, se a diferença entre a potência dos governadores *performance* e *powersave* fosse maior, a tendência seria de reduzir o consumo de energia no Passo 5. Com base nos experimentos realizados, verificou-se que se este comportamento fosse mantido, o consumo também reduziria no conjunto que contém apenas os Passos 5 e 6.

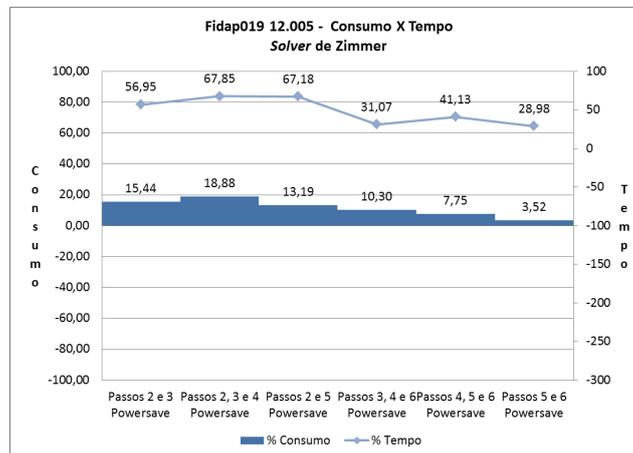


Figura 6.14 – Comparação entre consumo e tempo para a matriz de entrada Fidap019

Ao observar a Figura 6.15, nota-se que os conjuntos que contêm o Passo 2 também apresentam considerável aumento no percentual de tempo de execução e consumo de energia, tendo como entrada a matriz Bcsstk17. Observa-se um comportamento semelhante aos resultados dos testes cuja entrada é a matriz Fidap019, pois dentre os conjuntos que possuem o Passo 2, o que apresentou maior percentual de aumento no consumo foi o conjunto com os Passos 2, 3 e 4. Da mesma forma, entre os demais conjuntos, o que apresentou maior percentual de aumento no consumo de energia foi o conjunto com os Passos 3, 4 e 6.

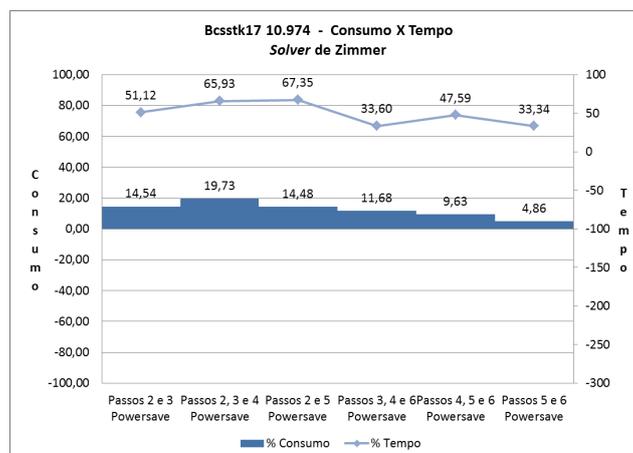


Figura 6.15 – Comparação entre consumo e tempo para a matriz de entrada Bcsstk17

A Figura 6.16 apresenta os resultados do *solver* tendo como entrada a matriz Fidapm37. Em relação ao percentual de aumento no tempo de execução, observa-se um aumento acentuado

do conjunto com os Passos 2 e 5. Nota-se que este comportamento se assemelha ao das matrizes densas. Acredita-se que isto ocorre, devido à grande quantidade de valores não nulos da matriz Fidapm37.

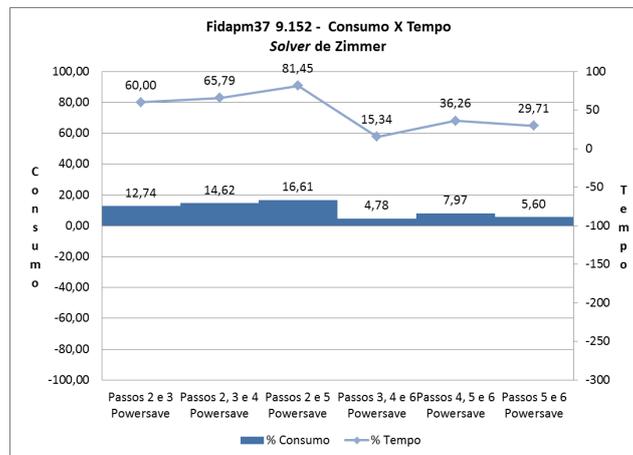


Figura 6.16 – Comparação entre consumo e tempo para a matriz de entrada Fidapm37

Ao examinar a Figura 6.17, nota-se que o comportamento da matriz Dw8192 é semelhante ao comportamento da matriz Bcsstk17 e Fidap019 em relação ao consumo de energia. Em relação ao aumento no tempo de execução há semelhança com as matrizes Fidap019 e Bcsstk17. Conforme mencionado anteriormente, nestes resultados o percentual de aumento no tempo de execução dos conjuntos que contêm o Passo 2 são altos se comparados aos demais conjuntos. Além disso, o conjunto com os Passos 4, 5 e 6 apresenta maior percentual de aumento no tempo de execução dentre os passos que não contêm o Passo 2.

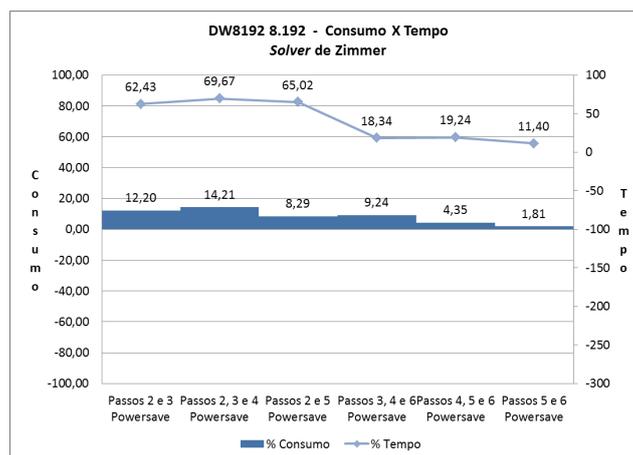


Figura 6.17 – Comparação entre consumo e tempo para a matriz de entrada Dw8192

Por fim, a Figura 6.18 apresenta os resultados tendo como entrada a matriz Utm5940. Os conjuntos que continham o Passo 2 apresentaram um percentual de aumento no tempo de execução próximo à 100%. Estes comportamentos são justificados devido ao tamanho da matriz Utm5940 ser menor que o tamanho das demais matrizes testadas. Entretanto, apesar do grande aumento de percentual no tempo de execução, o comportamento do consumo de energia foi semelhante ao das

matrizes Fidap019, Bcsstk17 e Dw8192. Pode-se observar que o conjunto com os Passos 2 e 5, entre os conjuntos que continham o Passo 2, apresentou maior percentual de consumo de energia. Já entre os demais conjuntos, o que apresentou maior percentual foi o conjunto com os Passos 3, 4 e 6.

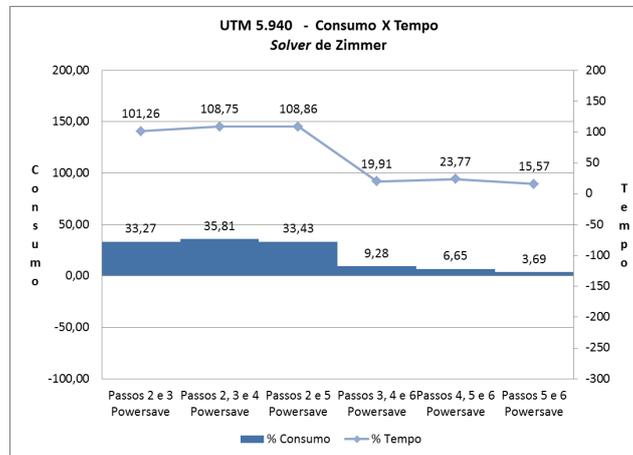


Figura 6.18 – Comparação entre consumo e tempo para a matriz de entrada Utm5940

É importante salientar que as matrizes Fidap019, Bcsstk17, Dw8192 e Utm5940 apresentaram comportamentos no percentual de consumo de energia e tempo de execução semelhantes. Já as matrizes Fidapm37 e Fidapm29 apresentaram um comportamento um pouco diferente. Para estas duas matrizes o conjunto que contém os Passos 2 e 5 apresenta percentual de tempo de execução e consumo de energia superiores em relação aos demais conjuntos. Enfatiza-se também que na etapa anterior de definição de frequência em pontos isolados, estas duas matrizes apresentaram um consumo de energia com um percentual maior no Passo 5 em relação aos Passos 4 e 6. Acredita-se que a matriz Fidapm37 apresenta esse comportamento devido à quantidade de valores não nulos ser maior que as outras. Já para a matriz Fidapm29, isso acontece pois a diferença entre a potência do *powersave* e do *performance* é pouco representativa, apresentando um valor de 24 watts (representando em torno de 23% se comparado ao valor em *performance*) no Passo 5. Além disso, as matrizes pertencem ao mesmo repositório.

Ao observar os resultados das matrizes densas (Gen1 e DingDong), nota-se que as alterações de frequências em conjuntos de pontos diversos se comparados com a alteração apenas no Passo 2 isoladamente, apresentam percentual de consumo de energia maior. Desta forma, a alteração de frequência apenas no Passo 2 comportou-se melhor em relação à redução do consumo de energia do que em conjuntos que continham o Passo 2.

O padrão de comportamento das matrizes quando pontos isolados são agrupados pode ser observado na Figura 6.19. Pode-se destacar três grupos relacionados ao tempo de execução e ao consumo de energia. Um grupo contém as matrizes densas e os outros dois grupos contêm as matrizes esparsas. As matrizes Fidapm37 e Fidapm29 foram separadas das demais matrizes esparsas por apresentarem valores diferentes.

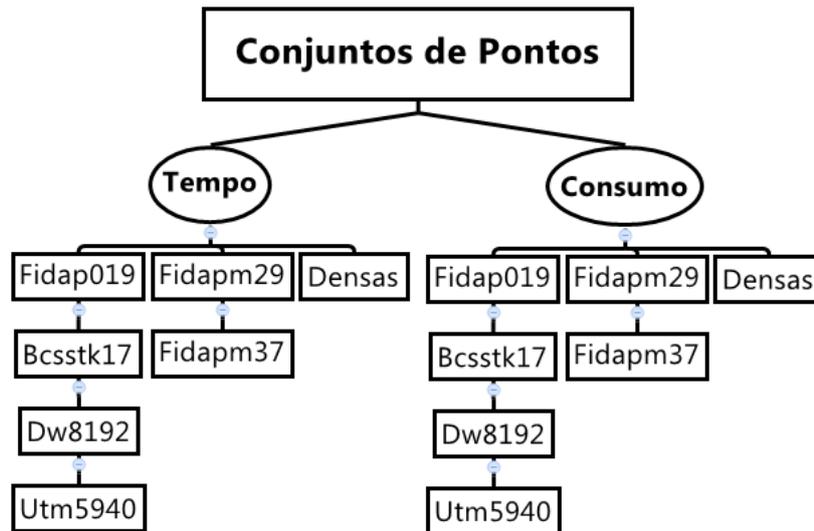


Figura 6.19 – Comparação entre as matrizes por tempo e por consumo

6.4 Considerações

Com base nos resultados obtidos, confirmou-se que utilização da técnica DVFS através da alteração de frequência com o governador em *powersave* ou *performance*, para os casos estudados, não interferiu na exatidão dos resultados.

A execução do *solver* com o governador em *powersave* durante todo o tempo, não apresentou redução no consumo de energia em relação a execução com o governador em *performance*. Observou-se que os valores médios de potência dos passos das matrizes Gen1 e DingDong apresentaram valores semelhantes, o que sugere que o comportamento dessas matrizes é similar, já que ambas são densas, bem condicionadas e possuem número de condicionamento próximo. Já os valores médios de potência das matrizes esparsas diferem em alguns casos.

Na etapa de definição de frequência em pontos isolados, observou-se que em relação a porcentagem de aumento no tempo de execução, os resultados das matrizes testadas podem ser separados em dois grupos: o grupo das matrizes densas e o grupo das matrizes esparsas. Para as matrizes densas, a alteração de frequência no Passo 5 apresentou considerável aumento de percentual no tempo de execução e no consumo de energia se comparada com a alteração de frequência nos demais passos. Além disso, observou-se redução no consumo de energia ao alterar a frequência no Passo 2. De acordo com Castillo [6], quando a aplicação é *CPU-bound* pode haver aumento no consumo de energia ao alterar a frequência com o governador em *powersave*. Desta forma, acredita-se que o cálculo da inversa aproximada, para as matrizes densas, não apresenta apenas características de uso intensivo de CPU durante a execução, pois obteve-se um percentual de redução de consumo. Já para as matrizes esparsas, observou-se aumento no consumo de energia ao alterar a frequência para o *powersave* em passos. Nota-se que para a maioria das matrizes esparsas a alteração de

frequência ocasionou aumento no tempo de execução e consumo de energia no Passo 2 (cálculo da inversa).

Na etapa de alteração de frequência em pontos diversos observou-se que o comportamento das matrizes permaneceu semelhante ao comportamento dos passos da etapa de alteração de frequência em pontos isolados na Subseção 6.3.2. Entretanto, ao analisar os resultados obtidos para as matrizes densas, verificou-se que o percentual de redução de energia em alguns conjuntos que continham o Passo 2 apresentaram menor redução, e em alguns casos não apresentaram. Isso indica que alterar a frequência com o governador em *powersave* apenas no Passo 2 é a melhor abordagem dentre as testadas para reduzir o consumo de energia para estes tipos de matrizes densas.

Para as matrizes esparsas na etapa de alteração de frequência em pontos diversos não houve redução no consumo de energia em relação ao comportamento observado com o governador em *performance*. Observou-se que o comportamento das matrizes Fidap019, Bcsstk17, Dw8192 e Utm5940 se mantiveram semelhantes. As matrizes Fidapm29 e Fidapm37 se diferenciaram um pouco das demais matrizes, pois os conjuntos 2 e 5 apresentaram um consumo de energia e tempo de execução um pouco maior que os outros conjuntos. Estima-se que isso aconteceu pois a matriz Fidapm37 apresenta uma quantidade de valores não nulos maior que as outras. A matriz Fidapm29 também é diferente, pois apresenta a diferença entre o consumo de energia do *powersave* e do *performance* com pouco representatividade, apresentando um valor de 24 *watts* no Passo 5. Essas duas matrizes também pertencem ao mesmo repositório.

Com base nas análises dos resultados apresentados, conclui-se que para reduzir o consumo de energia nas matrizes densas, pode-se alterar a frequência durante a execução apenas do Passo 2. A diferença de potência entre a execução com o governador em *performance* e *powersave* ao realizar a média entre os passos é de aproximadamente 46 *watts* (representando aproximadamente 33% do valor em *performance*) para as matrizes densas. Porém, no Passo 2, para essas matrizes, a diferença entre o consumo do *performance* e o *powersave* é em média de 54 *watts* (representando em torno de 39% do valor em *performance*).

Para as matrizes esparsas acredita-se que não é possível economizar energia com configurações de máquina semelhantes ao servidor de testes utilizado. Com base nos resultados obtidos, espera-se que se o comportamento se mantiver o mesmo em outras arquiteturas e a diferença entre a potência durante a execução em *performance* e *powersave* for maior, pode-se reduzir o consumo de energia ao alterar a frequência no Passo 5 para as matrizes Fidapm29, Fidap019, Bcsstk17, Dw8192 e Utm5940. Assim, conclui-se que para as matrizes esparsas a alteração de frequência para o *powersave*, compensaria se a diferença entre o *performance* e o *powersave* fosse maior.

7. CONCLUSÃO

Devido à crescente preocupação com questões ambientais, diversas pesquisas estão sendo desenvolvidas com o objetivo de reduzir o consumo de energia em diferentes ambientes computacionais. Neste trabalho é apresentado um estudo sobre a possibilidade de redução do consumo de energia utilizando a técnica DVFS (*Dynamic Voltage and Frequency Scaling*) em um *solver* de Alto Desempenho com verificação do resultado, que soluciona Sistemas de Equações Lineares. Desta forma, o foco deste trabalho é a redução do consumo de energia procurando manter a exatidão dos resultados com menor impacto no desempenho da aplicação.

Além do estudo, foi desenvolvida uma metodologia para avaliar o consumo de energia nestes *solvers* de Sistemas de Equações Lineares. Esta metodologia apresentou três estratégias de verificação de consumo de energia. Assim, as principais contribuições deste trabalho são: a verificação da possibilidade de utilização da técnica DVFS em um *solver* paralelo com verificação do resultado, respeitando a manutenção da exatidão obtida nos resultados numéricos, a avaliação do impacto do uso desta técnica em relação ao tempo de execução e ao consumo de energia e a metodologia para avaliação do consumo de energia do *solver* através de estratégias.

Antes de utilizar a técnica, foi realizado um estudo de caso utilizando o *cluster* Atlântica do Laboratório de Alto Desempenho da PUCRS. Este estudo permitiu avaliar o tempo de execução e o consumo de energia do *solver* FastPILSS [31]. A avaliação foi realizada com matrizes de diferentes tamanhos (até 45.000), onde o cenário dos experimentos utilizou 1 e 4 nodos do *cluster*. Foi possível constatar um alto consumo de energia do *solver* para as matrizes testadas. Com base nos resultados e análises, considerou-se importante a realização de um estudo sobre a possibilidade de redução do consumo de energia neste *solver*.

Posteriormente, verificou-se a possibilidade de utilizar a técnica DVFS neste *solver* com o objetivo de reduzir o consumo de energia e analisar se este consumo é reduzido sem comprometer a exatidão dos resultados. Constatou-se que o uso da técnica não afeta o diâmetro do resultado intervalar, mantendo exatamente o mesmo resultado numérico e, por consequência, não afeta a exatidão dos resultados.

Foram desenvolvidas três estratégias para a alteração de frequência do processador em SELAS Verificados. A primeira etapa consiste na alteração do governador de frequência para o tipo *performance* (frequência no máximo) e para o tipo *powersave* (redução de frequência) durante toda a execução do *solver*. Na segunda etapa, o governador de frequência é definido como *performance* e é alterado para o *powersave* em pontos isolados do *solver*. Por fim, alteração do governador é realizada em conjuntos de pontos do *solver*.

Ao analisar os resultados dos experimentos na primeira etapa, observou-se que a execução do *solver* com frequência reduzida (*powersave*) durante todo o tempo, não apresentou redução no consumo de energia em relação à execução com o governador em *performance*. Isso acontece, pois a diferença entre os valores de consumo em *powersave* e *performance* é pequena. Quando os valores de consumo de energia em *powersave* e *performance* são semelhantes, o uso do *powersave*

não traz benefícios, uma vez que o tempo de execução aumenta e a redução de consumo não compensa esta perda de desempenho. Observou-se que os valores de consumo de energia em *powersave* e *performance* das matrizes densas (Gen1 e DingDong) são semelhantes, e também que as características dessas matrizes são similares, já que ambas são densas, bem condicionadas e possuem número de condicionamento próximo. Já os valores de consumo de energia em *performance* e *powersave* para as matrizes esparsas diferem, pois as matrizes esparsas apresentam características diferentes de porcentagem de valores não nulos, número de condicionamento e localização dos valores.

Observando os resultados da segunda etapa constatou-se que o comportamento do *solver* tendo como entrada as matrizes geradas é diferente do comportamento do *solver* tendo como entrada matrizes reais extraídas do site MatrixMarket. Observou-se que é possível reduzir o consumo de energia nas matrizes geradas (densas) ao alterar a frequência para o *powersave* durante a realização do cálculo da inversa aproximada da matriz de entrada (Passo 2). Nestes experimentos, observou-se também que se a diferença entre o *performance* e o *powersave* for maior e o comportamento se mantiver o mesmo, haveria redução no consumo de energia.

Na terceira etapa de definição de frequência em pontos diversos constatou-se que o comportamento das matrizes permaneceu semelhante ao observado na etapa anterior. Em relação ao consumo, observou-se que o percentual de redução de energia em alguns conjuntos que continham o Passo 2 é menor se comparado com o percentual de redução apenas no Passo 2 e, em alguns casos, esses conjuntos não apresentaram redução.

Conforme Castillo [6] et al., nem todas as operações que utilizam a técnica DVFS apresentam redução no consumo de energia. Castillo indica que o uso desta técnica em operações com uso intensivo de memória (*memory-bound*) apresentam redução, enquanto operações que utilizam intensivamente o processador (*CPU-bound*) não apresentam redução no consumo.

Com base nestas considerações, observou-se que é possível utilizar a técnica DVFS em *solvers* verificados de SELAs sem afetar a exatidão do resultado. Porém, na maioria dos casos, o tempo de execução aumenta consideravelmente e o consumo de energia não reduz devido ao uso intensivo de processamento. Entretanto, ao utilizar a estratégia de dividir o algoritmo em passos e estudar a alteração de frequência isoladamente em cada passo, foi possível observar que ao alterar a frequência do Passo 2 para o *powersave* em matrizes densas é possível reduzir, ainda que sutilmente, o consumo de energia. Além disso, acredita-se que se a diferença entre o *performance* e o *powersave* for maior, haverá uma redução maior no consumo de energia no Passo 2.

7.1 Trabalhos Futuros

Ao desenvolver este trabalho, observou-se novas oportunidades para continuar e aperfeiçoar a pesquisa. Inicialmente buscou-se realizar experimentos com uma quantidade maior de nodos, com objetivo de analisar o comportamento do *solver* ao utilizar a técnica com matrizes de grande porte.

Entretanto, não foi possível o uso da técnica DVFS em *clusters*, devido a necessidade de permissão *root*, concorrência de acesso e restrições de tempo de utilização das máquinas. Futuramente, poderia ser realizada uma comparação dos resultados obtidos neste trabalho com experimentos utilizando máquinas de grande porte.

Neste trabalho foram realizados experimentos com matrizes densas e esparsas em um *solver* voltado à resolução de matrizes densas. Acredita-se que é importante realizar um estudo de *solvers* para matrizes esparsas, com ou sem computação verificada para a inclusão da técnica DVFS, pois matrizes esparsas são mais utilizadas em problemas reais de diversas áreas.

Este trabalho é voltado à resolução de SELAs e optou-se por não realizar a análise do tempo de alocação das matrizes e variáveis de memória. Acredita-se que seria interessante um estudo para averiguar o consumo de energia e o impacto da utilização da técnica DVFS durante este passo de alocação.

Atualmente várias pesquisas estão sendo desenvolvidas voltadas à redução do consumo de energia. Assim, é interessante realizar um estudo de novas técnicas e propor um modelo de economia de energia para o *solver* de Zimmer, onde o usuário poderá definir o percentual máximo de perda de desempenho aceitável.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] American National Standards Institute Institute of Electrical and Eletronics Engineers. "IEEE Standard for Binary Floating-Point Arithmetic". Capturado em: <ftp://ftp.openwatcom.org/pub/devel/docs/ieee-754.pdf>, Fevereiro 2013.
- [2] ANZT, H.; ROCKER, B.; HEUVELINE, V. "Energy Efficiency of Mixed Precision Iterative Refinement Methods Using Hybrid Hardware Platforms", *Computer Science - Research and Development*, vol. 25, Ago 2010, pp. 141–148.
- [3] BEKAS, C.; CURIONI, A. "A New Energy Aware Performance Metric", *Computer Science - Research and Development*, vol. 25, Ago 2010, pp. 187–195.
- [4] C-XSC. "Universität Wuppertal: Wissenschaftliches rechnen / softwaretechnologie". Capturado em: <http://www2.math.uni-wuppertal.de/~xsc/xsc/cxsc.html>, Dezembro 2011.
- [5] CARISSIMI, A.; GEYER, C. F. R.; MAILLARD, N.; NAVAUX, P. O. A.; CAVALHEIRO, G. G. H.; PILLA, M. L.; YAMIN, A.; CHARÃO, A. S.; STEIN, B.; ROSE, C. A. F. D.; FERNANDES, L. G. L.; FERRETO, T. C.; ZORZO, A. "Energy-Aware Scheduling of Parallel Programs". In: Conferencia Latino Americana de Computación de Alto Rendimiento (CLCAR), 2010, pp. 95–101.
- [6] CASTILLO, M.; FERNANDEZ, J.; MAYO, R.; QUINTANA-ORTI, E.; ROCA, V. "Analysis of Strategies to Save Energy for Message-Passing Dense Linear Algebra Kernels". In: 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2012, pp. 346–352.
- [7] CHEN, G.; MALKOWSKI, K.; KANDEMIR, M.; RAGHAVAN, P. "Reducing Power with Performance Constraints for Parallel Sparse Applications". In: 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2005, pp. 1–8.
- [8] CHOI, K.; LEE, W.; SOMA, R.; PEDRAM, M. "Dynamic Voltage and Frequency Scaling Under a Precise Energy Model Considering Variable and Fixed Components of the System Power Dissipation". In: IEEE/ACM International Conference on Computer Aided Design (ICCAD), 2004, pp. 29–34.
- [9] CLAUDIO, D.; MARINS, J. "Cálculo Numérico Computacional: Teoria e Prática". Editora Atlas S. A., 2000, 464p.
- [10] Computational Algebra Group. "Magma computational algebra system". Capturado em: <http://magma.maths.usyd.edu.au/magma/>, Março 2013.
- [11] DING, Y.; MALKOWSKI, K.; RAGHAVAN, P.; KANDEMIR, M. "Towards Energy Efficient Scaling of Scientific Codes". In: IEEE International Symposium on Parallel and Distributed Processing, 2008, pp. 1–8.

- [12] DONG, Y.; CHEN, J.; YANG, X.; YANG, C.; PENG, L. "Low Power Optimization for MPI Collective Operations". In: 9th International Conference for Young Computer Scientists, 2008, pp. 1047–1052.
- [13] DONGARRA, J. J.; FOSTER, I.; FOX, G.; GROPP, W.; KENNEDY, K.; TORCZON, L.; WHITE, A. "Sourcebook of Parallel Computing". Morgan Kaufmann Publishers, 2003, 842p.
- [14] ELNOZAHY, E. N.; KISTLER, M.; RAJAMONY, R. "Energy-Efficient Server Clusters". In: 2nd International Conference on Power-aware Computer Systems, 2003, pp. 179–197.
- [15] ETINSKI, M.; CORBALAN, J.; LABARTA, J.; VALERO, M. "Understanding the Future of Energy-performance Trade-off via DVFS in HPC Environments", *Journal of Parallel and Distributed Computing*, vol. 72–4, Abr 2012, pp. 579–590.
- [16] FAN, X.; WEBER, W.; BARROSO, L. "Power Provisioning for a Warehouse-sized Computer". In: 34th Annual International Symposium on Computer Architecture, 2007, pp. 13–23.
- [17] FastPILSS. "Universität Wuppertal: Wissenschaftliches rechnen / softwaretechnologie". Capturado em: http://www2.math.uni-wuppertal.de/~xsc/xsc/cxsc_software.html#fastpilss, Agosto 2011.
- [18] FENG, X.; GE, R.; CAMERON, K. W. "Power and Energy Profiling of Scientific Applications on Distributed Systems". In: 19th IEEE International Parallel and Distributed Processing Symposium, 2005, pp. 1–34.
- [19] FREEH, V. W.; PAN, F.; KAPPIAH, N.; LOWENTHAL, D. K.; SPRINGER, R. "Exploring the Energy-Time Tradeoff in MPI Programs on a Power-Scalable Cluster". In: 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2005, pp. 1–10.
- [20] GE, R.; FENG, X.; CAMERON, K. W. "Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters". In: ACM/IEEE Conference on Supercomputing, 2005, pp. 1–34.
- [21] GE, R.; FENG, X.; SONG, S.; CHANG, H.; LI, D.; CAMERON, K. "Powerpack: Energy Profiling and Analysis of High-Performance Systems and Applications", *IEEE Parallel and Distributed Systems*, vol. 21–5, Maio 2010, pp. 658 –671.
- [22] HAMMER, R.; HOCKS, M.; KULISH, U.; RATZ, D. "C++ Toolbox for Verified Computing". Springer, 1995, 382p.
- [23] HOFSCHUSTER, W.; KRÄMER, W.; WEDNER, S.; WIETHOFF, A. "C–XSC 2.0 a C++ Class Library for Extended Scientific Computing", *Relatório Técnico, Instituto de Ciência da Computação, UW*, 2001, pp. 1–25.

- [24] HÖLBIG, C. “Ambiente de Alta Exatidão com Alto Desempenho para a Resolução de Problemas”. Tese de Doutorado, Programa de Pós-Graduação em Computação, UFRGS, 2005, 123p.
- [25] Innovative Computing Laboratory Academic Research in Enabling Technology; High Performance Computing. “Plasma”. Capturado em: <http://icl.cs.utk.edu/plasma/>, Março 2013.
- [26] KAPPIAH, N.; FREEH, V. W.; LOWENTHAL, D. K. “Just in Time Dynamic Voltage Scaling: Exploiting Inter-Node Slack to Save Energy in MPI Programs”. In: ACM/IEEE Supercomputing Conference, 2005, pp. 1–33.
- [27] KIMURA, H.; SATO, M.; HOTTA, Y.; BOKU, T.; TAKAHASHI, D. “Emprical Study on Reducing Energy of Parallel Programs Using Slack Reclamation by DVFS in a Power-scalable High Performance Cluster”. In: IEEE International Conference on Cluster Computing, 2006, pp. 1–10.
- [28] KOLBERG, M. “Parallel Self-Verified Solver for Dense Linear Systems”. Tese de Doutorado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2009, 124p.
- [29] KOLBERG, M.; DORN, M.; FERNANDES, L. G.; BOHLENDER, G. “Parallel Verified Linear System Solver for Uncertain Input Data”. In: 20th International Symposium on Computer Architecture and High Performance Computing, 2008, pp. 89–96.
- [30] KRAWCZYK, R. “Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken”, *Computing*, vol. 4–3, Set 1969, pp. 187–201.
- [31] KRÄMER, W.; ZIMMER, M. “Fast (Parallel) Dense Linear System Solvers in C-XSC Using Error Free Transformations and BLAS”. Springer, 2009, vol. 5492, pp. 230–249.
- [32] KULISCH, U.; MIRANKER, L. “Computer Arithmetic in Theory and Practice”. Academic Press, 1981, 268p.
- [33] LATIEF, H.; LUSZCZEK, P.; DONGARRA, J. “Profiling High Performance Dense Linear Algebra Algorithms on Multicore Architectures for Power and Energy Efficiency”, *Computer Science - Research and Development*, vol. 27-4, Nov 2012, pp. 277–287.
- [34] LIM, M. Y.; FREEH, V. W.; LOWENTHAL, D. K. “Adaptive, Transparent Frequency and Voltage Scaling of Communication Phases in MPI Programs”. In: ACM/IEEE Conference on Supercomputing, 2006, pp. 1–14.
- [35] Mathematical and Computational Sciences Division. “Matrix Market”. Capturado em: <http://math.nist.gov/MatrixMarket/>, Janeiro 2012.
- [36] Mathematical and Computational Sciences Division. “Matrix BCSSTK17”. Capturado em: <http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/bcsstruc2/bcsstk17.html>, Janeiro 2013.

- [37] Mathematical and Computational Sciences Division. "Matrix DW8192". Capturado em: <http://math.nist.gov/MatrixMarket/data/NEP/dwave/dw8192.html>, Maio 2013.
- [38] Mathematical and Computational Sciences Division. "Matrix FIDAP019". Capturado em: <http://math.nist.gov/MatrixMarket/data/SPARSKIT/fidap/fidap019.html>, Maio 2013.
- [39] Mathematical and Computational Sciences Division. "Matrix FIDAPM29". Capturado em: <http://math.nist.gov/MatrixMarket/data/SPARSKIT/fidap/fidapm29.html>, Maio 2013.
- [40] Mathematical and Computational Sciences Division. "Matrix FIDAPM37". Capturado em: <http://math.nist.gov/MatrixMarket/data/SPARSKIT/fidap/fidapm37.html>, Maio 2013.
- [41] Mathematical and Computational Sciences Division. "Matrix UTM5940". Capturado em: <http://math.nist.gov/MatrixMarket/data/SPARSKIT/tokamak/utm5940.html>, Maio 2013.
- [42] MILANI, C. R. "Computação Verificada Aplicada à Resolução De Sistemas Lineares Intervalares Densos em Arquiteturas Multicore". Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2010, 92p.
- [43] MINARTZ, T.; LUDWIG, T.; KNOBLOCH, M.; MOHR, B. "Managing Hardware Power Saving Modes for High Performance Computing". In: Green Computing Conference and Workshops (IGCC), 2011, pp. 1–8.
- [44] MURUGESAN, S. "Harnessing Green IT: Principles and Practices.", *IT Professional*, vol. 10, Jan-Fev 2008, pp. 24–33.
- [45] MÄMMELÄ, O.; MAJANEN, M.; BASMADJIAN, R.; MEER, H.; GIESLER, A.; HOMBERG, W. "Energy-Aware Job Scheduler for High-Performance Computing", *Computer Science - Research and Development*, vol. 27–4, Ago 2011, pp. 265–275.
- [46] MÓR, S. D. K.; ALVES, M. A. Z.; LIMA, J. V. F.; MAILLARD, N. B.; NAVAUX, P. O. A. "Eficiência Energética em Computação de Alto Desempenho: Uma Abordagem em Arquitetura e Programação para Green Computing", *XXXVII Seminário Integrado de Software e Hardware (SEMISH)*, 2010, pp. 346–360.
- [47] National Science Foundation. "Blacs". Capturado em: <http://www.netlib.org/blacs/>, Janeiro 2012.
- [48] National Science Foundation. "Linpack". Capturado em: <http://www.netlib.org/linpack/>, Janeiro 2012.
- [49] National Science Foundation. "Pblas home page". Capturado em: http://netlib.org/scalapack/pblas_qref.html, Janeiro 2012.
- [50] National Science Foundation. "Scalapack". Capturado em: <http://www.netlib.org/scalapack/>, Janeiro 2012.

- [51] National Science Foundation. “Blas (*Basic Linear Algebra Subprograms*)”. Capturado em: <http://netlib.org/blas/>, Setembro 2011.
- [52] National Science Foundation. “Lapack”. Capturado em: <http://www.netlib.org/lapack/>, Setembro 2011.
- [53] OLIVEIRA, P.; DIVERIO, T.; CLAUDIO, D. “Fundamentos de Matemática Intervalar”. Sagra Luzzatto, 1997, 90p.
- [54] RIZVANDI, N. B.; TAHERI, J.; ZOMAYA, A. Y. “Some Observations on Optimal Frequency Selection in DVFS-based Energy Consumption Minimization”, *Parallel Distributed Computing*, vol. 71–8, Ago 2011, pp. 1154–1164.
- [55] RIZVANDI, N. B.; ZOMAYA, A. Y.; LEE, Y. C.; BOLOORI, A. J.; TAHERI, J. “Multiple Frequency Selection in DVFS-Enabled Processors to Minimize Energy Consumption”. John Wiley and Sons, 2012, pp. 443–463.
- [56] RUMP, S. “Kleine Fehlerschranken bei Matrixproblemen”. Tese de Doutorado, Programa de Pós Graduação em Matemática, Karlsruhe, 1980, 98p.
- [57] RUMP, S. “Self-validating methods”, *Linear Algebra and Its Applications*, vol. 324–1-3, Fev 2001, pp. 3–13.
- [58] Scalable Performance Lab. “Powerpack 3.0, scape lab”. Capturado em: <http://scape.cs.vt.edu/software/powerpack-3-0/>, Janeiro 2012.
- [59] ScalaLife. “Gromacs”. Capturado em: <http://www.gromacs.org/>, Agosto 2013.
- [60] SHARMA, V.; THOMAS, A.; ABDELZAHER, T.; SKADRON, K.; LU, Z. “Power-aware QoS Management in Web Servers”. In: 24th IEEE International Real-Time Systems Symposium, 2003, pp. 1–10.
- [61] WANG, D. “Meeting Green Computing Challenges”. In: 07th International Symposium on High Density Packaging and Microsystem Integration, 2007, pp. 1–4.

APÊNDICE A – CAPÍTULO 4

A Seção A.1 apresenta as tabelas restantes de desempenho da Seção 4.3 do Capítulo 4. A Seção A.2 apresenta as tabelas restantes da Seção 4.5 do Capítulo 4.

A.1 Desempenho

Tabela A.1 – Tempos de execução dos passos no *solver* de Zimmer com 16 processos

Solver			Kolberg					
Matrizes	Ordem	Nodos	Passo 2	Passo 3	Passo 4	Passo 5	Passo 6	Total
Gen1	45.000	4	2.383,89	29,91	14,86	6.387,01	320,57	9.136,24
DingDong	45.000	4	2.341,04	11,63	14,51	6.700,40	264,07	9.331,65
Gen1	30.000	4	728,37	14,34	7,15	910,16	3,82	1.663,83
DingDong	30.000	4	723,07	5,49	7,02	914,62	3,81	1.654,01
Gen1	15.000	4	107,94	4,49	2,15	118,02	1,21	233,82
DingDong	15.000	4	100,33	1,73	2,13	117,21	1,10	222,49
Fidapm029	13.668	4	92,85	1,46	1,87	93,52	0,80	190,50
Fidap019	12.005	4	69,27	1,64	1,48	62,96	0,69	136,05

Tabela A.2 – Tempos de execução dos passos no *solver* de Zimmer com 32 processos

Solver			Zimmer					
Matrizes	Ordem	Nodos	Passo 2	Passo 3	Passo 4	Passo 5	Passo 6	Total
Gen1	45.000	4	1.485,23	17,16	9,23	5.759,21	463,68	7.734,51
DingDong	45.000	4	1.407,13	6,27	7,99	6.121,96	308,56	7.851,90
Gen1	30.000	4	520,54	8,11	3,99	726,51	2,24	1.261,40
DingDong	30.000	4	459,15	3,08	3,93	611,06	2,11	1.079,33
Gen1	15.000	4	83,62	5,82	1,35	90,29	0,60	181,67
DingDong	15.000	4	79,03	2,28	1,29	90,39	0,69	173,69
Fidapm029	13.668	4	75,81	2,11	1,26	69,92	0,46	149,55
Fidap019	12.005	4	59,94	2,67	0,92	48,13	0,50	112,16

A.2 Estimativa de Consumo de Energia

Tabela A.3 – Consumo de energia em *joules* com frequência máxima de 2.40Ghz com 16 processos.

Matriz	Nodos	Ordem	Consumo (J)
Gen1	4	45.000	9.867.136,28
DingDong	4	45.000	10.078.181,19
Gen1	4	30.000	1.796.939,68
DingDong	4	30.000	1.786.334,96
Gen1	4	15.000	252.524,80
DingDong	4	15.000	240.284,31
Fidapm29	4	13.668	205.742,01
Fidap019	4	12.005	146.929,83

Tabela A.4 – Consumo de energia em *joules* com frequência máxima de 2.40Ghz com 32 processos.

Matriz	Nodos	Ordem	Consumo (J)
Gen1	4	45.000	8.353.270,72
DingDong	4	45.000	8.480.053,10
Gen1	4	30.000	1.362.308,57
DingDong	4	30.000	1.165.672,29
Gen1	4	15.000	196.202,87
DingDong	4	15.000	187.584,95
Fidapm29	4	13.668	161.517,74
Fidap019	4	12.005	121.136,15

APÊNDICE B – CAPÍTULO 6

A Seção B.1 apresenta as tabelas restantes da Seção 6.2 sobre exatidão do Capítulo 6. A Seção B.2 apresenta as tabelas restantes da Seção 6.3 sobre os resultados do consumo de energia e desempenho do Capítulo 6.

B.1 Exatidão

Tabela B.1 – Resultados do *solver* de Zimmer para a Matriz DingDong com 4 processos

x[]	Zimmer	Zimmer com DVFS
x[0]	[1.381965081461679E+002, 1.381965081461681E+002]	[1.381965081461679E+002, 1.381965081461681E+002]
x[1]	[6.909595072116979E+001, 6.909595072116989E+001]	[6.909595072116979E+001, 6.909595072116989E+001]
x[2]	[5.182023546935216E+001, 5.182023546935224E+001]	[5.182023546935216E+001, 5.182023546935224E+001]
x[3]	[4.318208986685972E+001, 4.318208986685978E+001]	[4.318208986685972E+001, 4.318208986685978E+001]
x[4]	[3.778306886193443E+001, 3.778306886193449E+001]	[3.778306886193443E+001, 3.778306886193449E+001]
x[5]	[3.400362814352547E+001, 3.400362814352552E+001]	[3.400362814352547E+001, 3.400362814352552E+001]
x[6]	[3.116895308404320E+001, 3.116895308404325E+001]	[3.116895308404320E+001, 3.116895308404325E+001]
x[7]	[2.894163412077508E+001, 2.894163412077513E+001]	[2.894163412077508E+001, 2.894163412077513E+001]
x[8]	[2.713187710972111E+001, 2.713187710972116E+001]	[2.713187710972111E+001, 2.713187710972116E+001]
x[9]	[2.562369596764498E+001, 2.562369596764503E+001]	[2.562369596764498E+001, 2.562369596764503E+001]

Tabela B.2 – Resultados do *solver* de Zimmer para a Matriz Fidap019 com 4 processos

x[]	Zimmer	Zimmer com DVFS
x[0]	[4.571964621165769E+003, 4.571964628737764E+003]	[4.571964621165769E+003, 4.571964628737764E+003]
x[1]	[-1.663813454000575E+002,-1.663813423072849E+002]	[-1.663813454000575E+002,-1.663813423072849E+002]
x[2]	[-6.871703922032086E+002,-6.871703852777510E+002]	[-6.871703922032086E+002,-6.871703852777510E+002]
x[3]	[3.769679744360134E+002, 3.769679762750579E+002]	[3.769679744360134E+002, 3.769679762750579E+002]
x[4]	[-5.519618296494673E+001,-5.519617543827833E+001]	[-5.519618296494673E+001,-5.519617543827833E+001]
x[5]	[-8.451577041524914E+001,-8.451576915166165E+001]	[-8.451577041524914E+001,-8.451576915166165E+001]
x[6]	[-1.592901891095856E+002,-1.592901831088274E+002]	[-1.592901891095856E+002,-1.592901831088274E+002]
x[7]	[1.126918243015328E+002, 1.126918252184748E+002]	[1.126918243015328E+002, 1.126918252184748E+002]
x[8]	[-7.335637844620141E+001,-7.335637334380604E+001]	[-7.335637844620141E+001,-7.335637334380604E+001]
x[9]	[-4.422795444781230E+001,-4.422795381188728E+001]	[-4.422795444781230E+001,-4.422795381188728E+001]

Tabela B.3 – Resultados do *solver* de Zimmer para a Matriz Bcsstk17 com 4 processos.

x[]	Zimmer	Zimmer com DVFS
x[0]	[9,999999999999997E-001, 1,000000000000001E+000]	[9,999999999999997E-001, 1,000000000000001E+000]
x[1]	[4,388641548221828E-008, 4,388641548221831E-008]	[4,388641548221828E-008, 4,388641548221831E-008]
x[2]	[2,537201051038633E-006, 2,537201051038635E-006]	[2,537201051038633E-006, 2,537201051038635E-006]
x[3]	[5,432124915630344E-008, 5,432124915630347E-008]	[5,432124915630344E-008, 5,432124915630347E-008]
x[4]	[9,999999999999997E-001, 1,000000000000001E+000]	[9,999999999999997E-001, 1,000000000000001E+000]
x[5]	[9,999999999999997E-001, 1,000000000000001E+000]	[9,999999999999997E-001, 1,000000000000001E+000]
x[6]	[9,999999999999997E-001, 1,000000000000001E+000]	[9,999999999999997E-001, 1,000000000000001E+000]
x[7]	[2,326690494580464E-008, 2,326690494580466E-008]	[2,326690494580464E-008, 2,326690494580466E-008]
x[8]	[2,518160073152223E-008, 2,518160073152225E-008]	[2,518160073152223E-008, 2,518160073152225E-008]
x[9]	[1,479269753447750E-008, 1,479269753447752E-008]	[1,479269753447750E-008, 1,479269753447752E-008]

Tabela B.4 – Resultados do *solver* de Zimmer para a Matriz Fidapm37 com 4 processos

x[]	Zimmer	Zimmer com DVFS
x[0]	[8.407943991146310E-003, 8.407943991146416E-003]	[8.407943991146310E-003, 8.407943991146416E-003]
x[1]	[5.200329181135330E-003, 5.200329181135678E-003]	[5.200329181135330E-003, 5.200329181135678E-003]
x[2]	[4.691350917810609E-003, 4.691350917810917E-003]	[4.691350917810609E-003, 4.691350917810917E-003]
x[3]	[-1.565593537228262E-002,-1.565593536702447E-002]	[-1.565593537228262E-002,-1.565593536702447E-002]
x[4]	[9.678853868641161E-003, 9.678853874157123E-003]	[9.678853868641161E-003, 9.678853874157123E-003]
x[5]	[3.505235410065610E-002, 3.505235410493132E-002]	[3.505235410065610E-002, 3.505235410493132E-002]
x[6]	[-6.511807091045279E-002,-6.511807089953318E-002]	[-6.511807091045279E-002,-6.511807089953318E-002]
x[7]	[2.885121596688119E-002, 2.885121597465974E-002]	[2.885121596688119E-002, 2.885121597465974E-002]
x[8]	[5.037189733044104E-002, 5.037189733714455E-002]	[5.037189733044104E-002, 5.037189733714455E-002]
x[9]	[-1.001425211242189E-001,-1.001425211076431E-001]	[-1.001425211242189E-001,-1.001425211076431E-001]

Tabela B.5 – Resultados do *solver* de Zimmer para a Matriz Dw8192 com 4 processos

x[]	Zimmer	Zimmer com DVFS
x[0]	[-2.864672775197476E+002,-2.864672775197362E+002]	[-2.864672775197476E+002,-2.864672775197362E+002]
x[1]	[-5.527341332529428E+002,-5.527341332529209E+002]	[-5.527341332529428E+002,-5.527341332529209E+002]
x[2]	[-7.930382698018557E+002,-7.930382698018245E+002]	[-7.930382698018557E+002,-7.930382698018245E+002]
x[3]	[-9.691268119378880E+002,-9.691268119378499E+002]	[-9.691268119378880E+002,-9.691268119378499E+002]
x[4]	[-1.045084623233168E+003,-1.045084623233126E+003]	[-1.045084623233168E+003,-1.045084623233126E+003]
x[5]	[-9.904979148655038E+002,-9.904979148654658E+002]	[-9.904979148655038E+002,-9.904979148654658E+002]
x[6]	[-7.796527395350170E+002,-7.796527395349883E+002]	[-7.796527395350170E+002,-7.796527395349883E+002]
x[7]	[-4.524886083278131E+002,-4.524886083277984E+002]	[-4.524886083278131E+002,-4.524886083277984E+002]
x[8]	[-7.330121864918234E+001,-7.330121864918061E+001]	[-7.330121864918234E+001,-7.330121864918061E+001]
x[9]	[2.975915982223568E+002, 2.975915982223744E+002]	[2.975915982223568E+002, 2.975915982223744E+002]

Tabela B.6 – Resultados do *solver* de Zimmer para a Matriz Utm5940 com 4 processos

x[]	Zimmer	Zimmer com DVFS
x[0]	[-1.414213420810485E+000,-1.414213420810483E+000]	[-1.414213420810485E+000,-1.414213420810483E+000]
x[1]	[-1.568932286783809E+000,-1.568932286773491E+000]	[-1.568932286783809E+000,-1.568932286773491E+000]
x[2]	[-1.414213562373091E+000,-1.414213562373089E+000]	[-1.414213562373091E+000,-1.414213562373089E+000]
x[3]	[-1.414213562373091E+000,-1.414213562373089E+000]	[-1.414213562373091E+000,-1.414213562373089E+000]
x[4]	[-1.414213562373111E+000,-1.414213562373109E+000]	[-1.414213562373111E+000,-1.414213562373109E+000]
x[5]	[4.583008259837063E+002, 4.583008260153979E+002]	[4.583008259837063E+002, 4.583008260153979E+002]
x[6]	[-2.573672608138092E+001,-2.573672607981001E+001]	[-2.573672608138092E+001,-2.573672607981001E+001]
x[7]	[-4.169695789747562E+001,-4.169695789482314E+001]	[-4.169695789747562E+001,-4.169695789482314E+001]
x[8]	[-5.138092613710361E+002,-5.138092613360599E+002]	[-5.138092613710361E+002,-5.138092613360599E+002]
x[9]	[3.285101944605738E+005, 3.285101944830531E+005]	[3.285101944605738E+005, 3.285101944830531E+005]

B.2 Consumo de Energia e Desempenho

Tabela B.7 – Consumo de energia e tempo de execução reduzindo a frequência no Passo 2

Solver		Zimmer	
Matrizes	Ordem	Consumo Passo 2 (J)	(%)
Gen1	15.000	578.455,17	-3,29
DingDong	15.000	574.014,66	-2,22
Fidapm29	13.668	7.401,46	22,15
Fidapm019	12.005	5.066,12	10,47
Bcsstk17	10.974	3.786,65	10,22
Fidapm37	9.152	5.649,32	10,74
Dw8192	8.192	3.963,66	10,17
Utm5940	5.940	2.392,92	31,01
Matrizes	Ordem	Tempo Passo 2 (s)	(%)
Gen1	15.000	5.246,17	20,48
DingDong	15.000	5.241,23	20,47
Fidapm29	13.668	86,08	57,93
Fidapm019	12.005	54,92	43,29
Bcsstk17	10.974	40,13	39,80
Fidapm37	9.152	63,88	54,50
Dw8192	8.192	44,88	56,59
Utm5940	5.940	30,55	95,02

Tabela B.8 – Consumo de energia e tempo de execução reduzindo a frequência no Passo 3

Solver		Zimmer	
Matrizes	Ordem	Consumo Passo 3 (J)	(%)
Gen1	15.000	610.344,35	2,04
DingDong	15.000	596.843,45	1,67
Fidapm29	13.668	6.523,37	7,66
Fidapm019	12.005	4.762,18	3,84
Bcsstk17	10.974	3.585,96	4,38
Fidapm37	9.152	5.189,98	1,74
Dw8192	8.192	3.747,37	4,16
Utm5940	5.940	1.896,03	3,80
Matrizes	Ordem	Tempo Passo 3 (s)	(%)
Gen1	15.000	4.448,90	2,17
DingDong	15.000	4.407,44	1,31
Fidapm29	13.668	61,24	12,37
Fidapm019	12.005	43,58	13,69
Bcsstk17	10.974	32,09	11,80
Fidapm37	9.152	43,56	5,36
Dw8192	8.192	30,41	6,11
Utm5940	5.940	16,76	6,97

Tabela B.9 – Consumo de energia e tempo de execução reduzindo a frequência no Passo 4

Solver		Zimmer	
Matrizes	Ordem	Consumo Passo 4 (J)	(%)
Gen1	15.000	610.692,75	2,10
DingDong	15.000	607.011,16	3,40
Fidapm29	13.668	6.363,05	5,01
Fidapm019	12.005	4.756,71	3,72
Bcsstk17	10.974	3.622,96	5,46
Fidapm37	9.152	5.230,62	2,53
Dw8192	8.192	3.767,86	4,73
Utm5940	5.940	1.919,59	5,09
Matrizes	Ordem	Tempo Passo 4 (s)	(%)
Gen1	15.000	4.437,38	1,91
DingDong	15.000	4.467,22	2,68
Fidapm29	13.668	60,74	11,44
Fidapm019	12.005	43,26	12,86
Bcsstk17	10.974	33,14	15,44
Fidapm37	9.152	44,27	7,07
Dw8192	8.192	31,04	8,31
Utm5940	5.940	17,15	9,50

Tabela B.10 – Consumo de energia e tempo de execução reduzindo a frequência no Passo 5

Solver		Zimmer	
Matrizes	Ordem	Consumo Passo 5 (J)	(%)
Gen1	15.000	648.947,16	8,49
DingDong	15.000	635.007,04	8,17
Fidapm29	13.668	6.538,97	7,91
Fidapm019	12.005	4.689,42	2,25
Bcsstk17	10.974	3.582,06	4,27
Fidapm37	9.152	5.399,17	5,84
Dw8192	8.192	3.604,16	0,18
Utm5940	5.940	1.881,80	3,02
Matrizes	Ordem	Tempo Passo 5 (s)	(%)
Gen1	15.000	6.824,56	56,73
DingDong	15.000	6.757,80	55,33
Fidapm29	13.668	64,61	18,55
Fidapm019	12.005	47,79	24,68
Bcsstk17	10.974	36,66	27,72
Fidapm37	9.152	52,55	27,11
Dw8192	8.192	31,06	8,36
Utm5940	5.940	17,67	12,79

Tabela B.11 – Consumo de energia e tempo de execução reduzindo a frequência no Passo 6

Solver		Zimmer	
Matrizes	Ordem	Consumo Passo 6 (J)	(%)
Gen1	15.000	618.762,19	3,45
DingDong	15.000	597.562,22	1,79
Fidapm29	13.668	6.475,31	6,86
Fidapm019	12.005	4.646,69	1,32
Bcsstk17	10.974	3.506,06	2,05
Fidapm37	9.152	5.128,59	0,53
Dw8192	8.192	3.762,45	4,58
Utm5940	5.940	1.889,47	3,44
Matrizes	Ordem	Tempo Passo 6 (s)	(%)
Gen1	15.000	4.501,35	3,38
DingDong	15.000	4.407,85	1,32
Fidapm29	13.668	59,60	9,35
Fidapm019	12.005	40,39	5,39
Bcsstk17	10.974	30,75	7,14
Fidapm37	9.152	42,76	3,43
Dw8192	8.192	29,86	4,20
Utm5940	5.940	16,37	4,48

Tabela B.12 – Consumo de energia e tempo de execução reduzindo a frequência nos Passos 2 e 3

Solver		Zimmer	
Matrizes	Ordem	Consumo Passo 23 (J)	(%)
Gen1	15.000	603.973,37	0,97
DingDong	15.000	593.209,21	1,05
Fidapm29	13.668	7.256,64	19,76
Fidapm019	12.005	5.294,28	15,44
Bcsstk17	10.974	3.935,08	14,54
Fidapm37	9.152	5.751,39	12,74
Dw8192	8.192	4.036,71	12,20
Utm5940	5.940	2.434,27	33,27
Matrizes	Ordem	Tempo Passo 23 (s)	(%)
Gen1	15.000	5.446,63	25,09
DingDong	15.000	5.398,20	24,08
Fidapm29	13.668	87,43	60,42
Fidapm019	12.005	60,16	56,95
Bcsstk17	10.974	43,38	51,12
Fidapm37	9.152	66,15	60,00
Dw8192	8.192	46,55	62,43
Utm5940	5.940	31,53	101,26

Tabela B.13 – Consumo de energia e tempo de execução reduzindo a frequência nos Passos 2 e 5

Solver		Zimmer	
Matrizes	Ordem	Consumo Passo 25 (J)	(%)
Gen1	15.000	634.809,23	6,13
DingDong	15.000	626.769,54	6,77
Fidapm29	13.668	7.882,55	30,09
Fidapm019	12.005	5.191,02	13,19
Bcsstk17	10.974	3.932,87	14,48
Fidapm37	9.152	5.948,48	16,61
Dw8192	8.192	3.896,03	8,29
Utm5940	5.940	2.437,18	33,43
Matrizes	Ordem	Tempo Passo 25 (s)	(%)
Gen1	15.000	7.737,02	77,69
DingDong	15.000	7.728,17	77,64
Fidapm29	13.668	97,69	79,24
Fidapm019	12.005	64,08	67,18
Bcsstk17	10.974	48,04	67,35
Fidapm37	9.152	75,02	81,45
Dw8192	8.192	47,29	65,02
Utm5940	5.940	32,72	108,86

Tabela B.14 – Consumo de energia e tempo de execução reduzindo a frequência nos Passos 2, 3 e 4

Solver		Zimmer	
Matrizes	Ordem	Consumo Passo 234 (J)	(%)
Gen1	15.000	613.366,91	2,54
DingDong	15.000	593.906,43	1,17
Fidapm29	13.668	7.549,88	24,60
Fidapm019	12.005	5.452,07	18,88
Bcsstk17	10.974	4.113,34	19,73
Fidapm37	9.152	5.847,07	14,62
Dw8192	8.192	4.108,79	14,21
Utm5940	5.940	2.480,68	35,81
Matrizes	Ordem	Tempo Passo 234 (s)	(%)
Gen1	15.000	5.503,13	26,39
DingDong	15.000	5.408,87	24,33
Fidapm29	13.668	93,97	72,41
Fidapm019	12.005	64,34	67,85
Bcsstk17	10.974	47,63	65,93
Fidapm37	9.152	68,55	65,79
Dw8192	8.192	48,63	69,67
Utm5940	5.940	32,70	108,75

Tabela B.15 – Consumo de energia e tempo de execução dos *solvers* reduzindo a frequência nos Passos 3, 4 e 6

Solver		Zimmer	
Matrizes	Ordem	Consumo Passo 346 (J)	(%)
Gen1	15.000	614.404,94	2,72
DingDong	15.000	608.697,26	3,69
Fidapm29	13.668	7.148,31	17,97
Fidapm019	12.005	5.058,67	10,30
Bcsstk17	10.974	3.836,76	11,68
Fidapm37	9.152	5.345,25	4,78
Dw8192	8.192	3.930,11	9,24
Utm5940	5.940	1.996,01	9,28
Matrizes	Ordem	Tempo Passo 346 (s)	(%)
Gen1	15.000	4.472,51	2,72
DingDong	15.000	4.489,95	3,20
Fidapm29	13.668	72,89	33,74
Fidapm019	12.005	50,24	31,07
Bcsstk17	10.974	38,35	33,60
Fidapm37	9.152	47,69	15,34
Dw8192	8.192	33,92	18,34
Utm5940	5.940	18,78	19,91

Tabela B.16 – Consumo de energia e tempo de execução reduzindo a frequência nos Passos 4, 5 e 6

Solver		Zimmer	
Matrizes	Ordem	Consumo Passo 456 (J)	(%)
Gen1	15.000	645.805,90	7,97
DingDong	15.000	643.256,52	9,58
Fidapm29	13.668	6.891,01	13,72
Fidapm019	12.005	4.941,61	7,75
Bcsstk17	10.974	3.766,44	9,63
Fidapm37	9.152	5.507,91	7,97
Dw8192	8.192	3.754,05	4,35
Utm5940	5.940	1.948,09	6,65
Matrizes	Ordem	Tempo Passo 456 (s)	(%)
Gen1	15.000	6.792,21	55,99
DingDong	15.000	6.871,73	57,95
Fidapm29	13.668	73,84	35,47
Fidapm019	12.005	54,09	41,13
Bcsstk17	10.974	42,36	47,59
Fidapm37	9.152	56,34	36,26
Dw8192	8.192	34,17	19,24
Utm5940	5.940	19,39	23,77

Tabela B.17 – Consumo de energia e tempo de execução reduzindo a frequência nos Passos 5 e 6

Solver		Zimmer	
Matrizes	Ordem	Consumo Passo 56 (J)	(%)
Gen1	15.000	647.955,49	8,33
DingDong	15.000	637.615,20	8,62
Fidapm29	13.668	6.580,04	8,59
Fidapm019	12.005	4.747,77	3,52
Bcsstk17	10.974	3.602,42	4,86
Fidapm37	9.152	5.386,94	5,60
Dw8192	8.192	3.662,87	1,81
Utm5940	5.940	1.894,02	3,69
Matrizes	Ordem	Tempo Passo 56 (s)	(%)
Gen1	15.000	6.814,82	56,51
DingDong	15.000	6.792,93	56,14
Fidapm29	13.668	66,77	22,50
Fidapm019	12.005	49,43	28,98
Bcsstk17	10.974	38,27	33,34
Fidapm37	9.152	53,63	29,71
Dw8192	8.192	31,93	11,40
Utm5940	5.940	18,11	15,57