UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA

PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO

Programa Institucional de Bolas de Iniciação Científica

Integração da biblioteca ScaLAPACK em código para cálculo de modelagem molecular usando Mecânica Quântica

Bolsista: Lucas Santana Carneiro

Orientador: Angelo Amâncio Duarte

Modalidade: CNPq/UEFS

Agosto, 2015

UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
Programa Institucional de Bolas de Iniciação Científica

Integração da biblioteca ScaLAPACK em código para cálculo de modelagem molecular usando Mecânica Quântica

Relatório final sobre as atividades desenvolvidas como bolsista do PIBIC apresentado à Coordenação do Programa, como parte dos requisitos exigidos na Resolução Normativa 017/2006 do CNPq.

Bolsista: Lucas Santana Carneiro
Orientador: Angelo Amâncio Duarte
Modalidade: CNPq/UEFS

Feira de Santana, Agosto/2015

Apresentação

O presente relatório descreve o trabalho de Iniciação Cientifica realizado pelo estudante de Engenharia de Computação, Lucas Santana Carneiro, bolsista do PIBIC/CNPq, no Laboratório de Computação de Alto Desempenho (LaCAD) da Universidade Estadual de Feira de Santana (UEFS), sob a orientação do Professor Doutor Angelo Amâncio Duarte.

Lista de Figuras

Lista de Tabelas

Sumário

Resumo

Este relatório apresenta a metodologia, os resultados e as discussões acerca do trabalho desenvolvido para integração da biblioteca de software ScaLAPACK (Scalable Linear Algebra Package) ao código computacional que está sendo desenvolvido pela professora Fernanda Castelo Branco, mestranda do Programa de Pós-Graduação em Mecatrônica da Universidade Federal da Bahia, o qual implementa métodos de Modelagem Molecular utilizando a teoria da Mecânica Quântica.

A biblioteca de software ScaLAPACK possui rotinas de alto desempenho para resolução de problemas de álgebra linear em arquiteturas computacionais com memória distribuída. Tais rotinas são responsáveis pela resolução de equações lineares, resolução de mínimos quadrados, autovalores e autovetores e resolução de sistemas lineares, podendo ser utilizados em física computacional, modelagem molecular etc, em matrizes de elementos reais ou complexos, com a possibilidade de escolha entre precisão simples ou dupla, permitindo uma maior convergência dos resultados.

Este relatório apresenta detalhes sobre o ambiente no qual a biblioteca foi implementada e sobre a utilização de uma de suas funções para a busca por autovalores de uma matriz, que serão utilizados no cálculo da configuração da modelagem molecular de três átomos. Serão apresentados também resultados dos testes realizados para estudar a eficiência e a aceleração alcançada pela biblioteca para o problema em que foi utilizada.

1 Introdução

A utilização de bibliotecas de software, que são uma coleção de subprogramas utilizados no desenvolvimento de algoritmos, proporciona uma importante redução do tempo de codificação de programas, visto que, o programador se concentrará nas operações essenciais do seu algoritmo, sem precisar ater-se à implementação de funções básicas, como multiplicação de matrizes, soluções de sistemas de equações ou busca de autovalores e autovetores. Além disso, existem bibliotecas públicas de código que já foram historicamente testadas e aceitas pela comunidade acadêmica, o que reduz significativamente a possibilidade de erros de codificação.

A biblioteca de software Scalable Linear Algebra Package (ScaLAPACK), baseada na biblioteca Basic Linear Algebra Subprograms (BLAS), é utilizada como base na implementação de bibliotecas comerciais como a Intel MKL (INTEL, 2015), a NAG Parallel Library e a IBM Parallel ESSL (BLACKFORD et al., 2015d). Tal biblioteca possui funções otimizadas para resolução de sistemas de equações, resolução de equações lineares, busca por autovalores e autovetores, sendo utilizadas em diversos problemas em áreas como física computacional (JOIOSO et al., 2008), modelagem molecular (BLUM et al., 2009), estatística (YOGINATH et al., 2005) etc.

O presente relatório descreve o trabalho de instalação, configuração, manutenção e disponibilização de uso da biblioteca ScaLAPACK, como uma ferramenta no Laboratório de Computação de Alto Desempenho (LaCAD) da UEFS, buscando também uma forma eficiente de integrar a biblioteca no trabalho da mestranda Fernanda Castelo Branco (orientada pelo coordenador do projeto), relacionado à implementação de métodos computacionais de alto desempenho para o cálculo da configuração molecular de três átomos, através do uso da teoria da Mecânica Quântica.

Visando melhorar o entendimento do trabalho realizado, este relatório foi subdividido em seções. Na próxima seção, Revisão de Literatura, são apresentados os conceitos fundamentais para o entendimento das atividades desenvolvidas. Na seção Materiais e Métodos são explicitadas as ferramentas e a metodologia utilizadas no desenvolvimento das tarefas, o processo de desenvolvimento das atividades e o modo como as atividades previstas no plano de trabalho foram realizadas. Na seção Resultados são apresentados os resultados obtidos através de pesquisas e das atividades realizadas, que são discutidos na seção Discussão, na qual é feita uma análise dos trabalhos realizados e dos resultados obtidos. Finalmente, na última seção são feitas as considerações finais acerca das atividades realizadas.

2 Revisão de Literatura

A resolução de modelos matemáticos que descrevem fenômenos físicos, como a interação molecular, possuem soluções analíticas complexas. Para se chegar a uma solução de forma mais simples, métodos numéricos podem ser utilizados. Tais métodos podem ser resolvidos com a utilização de simulação por computador.

O uso de modelos matemáticos sofisticados requer grande poder computacional para alcançar resultados com precisão aceitável e num intervalo de tempo útil ao trabalho que está sendo desenvolvido. Para conseguir esse poder computacional, é necessário a utilização de arquiteturas paralelas, o que dificulta o desenvolvimento e o teste dos códigos usados para a simulação.

Uma forma de facilitar o desenvolvimento de simulações em arquiteturas paralelas é usar bibliotecas numéricas padronizadas, já testadas e otimizadas, buscando facilitar a implementação dos códigos, pois o programador pode utilizar diversas funções pré-codificadas e, principalmente, testadas para garantir a obtenção dos resultados corretos.

Entre as bibliotecas numéricas existentes a ScaLAPACK (BLACKFORD et al., 1997) é uma das bibliotecas de álgebra linear com foco em sistemas de memória distribuída mais conhecida, desenvolvida como uma continuação do projeto LAPACK (ANDERSON et al., 1999), tendo como objetivos eficiência, escalabilidade, confiabilidade de resultados, portabilidade, flexibilidade e facilidade de utilização. Porém a ScaLAPACK é otimizada para ambientes distribuídos, possuindo, além das funções para resolução de problemas de álgebra linear, funções de comunicação e divisão dos dados entre os processadores. Baseada na biblioteca padrão para Álgebra Linear, BLAS (BLACKFORD et al., 2002) e na LAPACK, a biblioteca de software ScaLAPACK, depende dessas bibliotecas para realizar operações em cada processador.

As bibliotecas BLACS (Basic Linear Algebra Communication Subprograms) (BLACKFORD et al., 1997) e PBLAS (Parallel BLAS) (BLACKFORD et al., 1997), foram criadas juntamente com a ScaLAPACK, buscando torná-la parecida com a LAPACK, e possuem funções paralelas para realizar os cálculos e a comunicação. Para realizar a passagem de mensagens na comunicação entre os processadores a BLACS utiliza-se de outra biblioteca, sendo que no trabalho realizado foi empregada a OpenMPI(OPENMPI, 2015), uma biblioteca de código aberto utilizada para para compilação e execução de códigos paralelos, baseada no padrão de comunicação MPI (Message Passing Interface)(MPI, 2015), que é uma especificação para bibliotecas de passagem de mensagem. No modelo MPI, os dados são movidos de um processador para outro através da cooperação entre os processos. A aplicação é composta por um ou mais processos que se comunicam através de envio e recebimento de mensagens. A

hierarquia completa das bibliotecas quem compõem a ScaLAPACK pode ser vista na Figura 1.

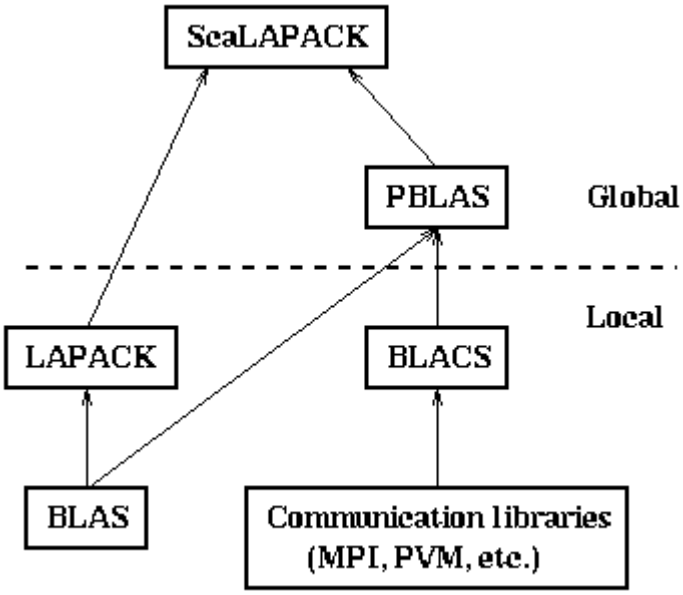

Figura 1: Hierarquia de Software da ScaLAPACK.

Pode-se verificar na Figura 1 as dependências locais e globais da ScaLAPACK, sendo que o funcionamento da mesma depende da instalação das bibliotecas BLAS, BLACS, LAPACK e a biblioteca de comunicação em cada computador no qual a aplicação deve executar.

As funções da biblioteca ScaLAPACK possuem uma padronização nos nomes semelhante a padronização da biblioteca LAPACK, com o acréscimo do prefixo P. Todas as funções ScaLAPACK possuem seus nomes da seguinte forma:

PXYYZZZ

Na qual:

- P, indica que é uma função ScaLAPACK
- X, indica o tipo de dado utilizado que pode ser S (real com precisão simples), D (real com precisão dupla), C (complexa com precisão simples) e Z (complexa com precisão dupla).
- YY, indica o tipo de matriz que vai ser operada. Podendo ser Simétrica, Geral, Triangular etc.
- ZZZ, indica o tipo de cálculo que vai ser realizado. Podendo ser autovalor, autovetor, sistemas de equação etc.

A função utilizada para a realização dos testes, foi a função PDSYEVR (BLACKFORD et al., 2015c), usada para encontrar os autovalores e autovetores em uma matriz simétrica real de precisão dupla (P – Função do ScaLAPACK, D – real com precisão dupla, SY – Matriz simétrica, EVR – Autovalores), desenvolvida utilizando o algoritmo MRRR (Multiple Relatively Robust Representations) (DHILLON; PARLETT; VÕMEL, 2006) que realiza a busca dos autovalores ortogonais de uma matriz triangular simétrica, sem necessitar de uma ortogonalização da matriz e possui complexidade O(n^2), garantindo um menor custo computacional. Esta função foi desenvolvida por um contribuidor e incorporada posteriormente ao ScaLAPACK.

A ScaLAPACK utiliza algumas convenções para realizar a distribuição dos dados entre os processadores(BLACKFORD et al., 2015a). Os processadores são distribuídos em uma grade de processos especificada pelo usuário na qual os processadores disponíveis são distribuídos em uma matriz. A partir da grade de processos e de descritores das matrizes, que guarda informações sobre as matrizes como tamanho, tipo e fator de divisão, os dados são distribuídos entre os processadores. Cada processador recebe um ou mais blocos da matriz, gerando uma matriz local para ser computada, informações sobre a localização de cada bloco da matriz são obtidos pelo descritor permitindo que a função recupere os resultados.

A ScaLAPACK foi implantada em um Cluster, que é um grupo de computadores interconectados por rede e com um Sistema Operacional (SO) distribuído. O Cluster utilizado é do tipo Beowulf (BEOWULF, 2015), ou Cluster dedicado, no qual as máquinas são utilizadas apenas para cômputo dispensando monitores e teclados, podendo ser homogêneas (os computadores possuem a mesma configuração) ou heterogêneas (os computadores possuem configurações diferentes).

3 Materiais e Métodos

O projeto foi desenvolvido utilizando a infraestrutura instalada no LaCAD, localizado na sala I12 do Labotec III, na Universidade Estadual de Feira de Santana (UEFS), utilizando um Cluster heterogêneo, que possui dois conjuntos de computadores, um com 16 máquinas com processadores i3 quad-core e o outro com 8 máquinas com processadores Pentium dual-core. O Cluster utiliza o sistema de gerenciamento Rocks Cluster versão 6.1.1(ROCKS, 2015), que tem como sistema operacional o CentOS 6.5(CENTOS, 2015).

Para desenvolver os códigos foram utilizados os editores de texto Gedit versão 2.28.4 (GEDIT, 2015), em modo gráfico, e Vim 7.2.411 (VIM, 2015), em modo terminal, utilizado para alterações remotas por SSH, também gratuitos e presentes na instalação do sistema operacional CentOS. Os códigos foram compilados e executados com o OpenMPI 1.8.1. Os gráficos apresentados neste relatório foram obtidos com o software Octave 3.8.2 (OCTAVE, 2015).

A ScaLAPACK é uma biblioteca de domínio público e pode ser encontrada para download gratuito na Internet(BLACKFORD et al., 2015e). A versão utilizada da ScaLAPACK foi a 2.0.2, contendo as bibliotecas BLACS e PBLAS. Além do ScaLAPACK, outras bibliotecas foram necessárias para a execução dos códigos, sendo a BLAS e a LAPACK 3.5.0, que podem ser baixadas no site da Netlib (NETLIB, 2015) ou juntamento com o ScaLAPACK durante a instalação.

Na primeira etapa do trabalho teve início a revisão bibliográfica acerca da biblioteca ScaLAPACK, seu funcionamento, características e funcionalidades e como utilizá-la em ambientes distribuídos. Após adquirir conhecimentos sobre a biblioteca, teve início a instalação da mesma, sendo implantada em um Cluster do LaCAD, nesta etapa do trabalho também foi desenvolvido um tutorial sobre o processo de instalação. Como o Cluster onde a biblioteca ScaLAPACK está instalada é do tipo heterogêneo, buscando aumentar a eficiência dos códigos, a biblioteca foi configurada separadamente em cada arquitetura distinta do Cluster visando a otimização da biblioteca. Durante o processo de instalação e utilização da ScaLAPACK, manutenções na infraestrutura do Laboratório, como atualizações de software ou configurações de rede, foram realizados para manter o mesmo em funcionamento permitindo uso da biblioteca.

Depois de instalar e configurar a ScaLAPACK, códigos de exemplos disponibilizados no site da biblioteca e um código para decomposição LU desenvolvido pelo bolsista, foram utilizados como testes para entender as funções, a organização dos códigos da biblioteca e verificar o funcionamento da mesma no Cluster. Após a realização dos testes, tomando como base o problema a ser resolvido no trabalho da mestranda Fernanda Castelo Branco, relacionado à implementação de métodos computacionais de alto desempenho para o cálculo da configuração

molecular de três átomos, foram feitas pesquisas na documentação do ScaLAPACK por funções de busca por autovalores, descobrindo a função de melhor desempenho. Definida a função PDSYEVR, que possui a menor complexidade dentre as funções disponíveis, foi realizado um estudo sobre suas características para facilitar a implementação posterior da mesma.

Com a escolha da função PDSYEVR testes de escalabilidade foram realizados buscando descobrir o funcionamento e o comportamento da mesma, servindo como forma de sintonizar parâmetros da função e testar a execução no ambiente disponível. Os testes realizados com a função mostraram a necessidade de um aumento de memória para a execução do algoritmo, pois quando o tamanho da matriz é aumentado os computadores não possuem espaço para alocar todas as variáveis da algoritmo. Como esse aumento não foi possível, a utilização de memória *swap* e uma reconfiguração do Cluster para aumentar a mesma em todos os computadores fez-se necessária para a execução do algoritmo. Durante o processo de testes da biblioteca um tutorial sobre sua utilização foi desenvolvido, como forma de facilitar o uso por outros pesquisadores.

4 Resultados

Para desenvolver o trabalho foi preciso fazer a instalação da ScaLAPACK, atualmente a biblioteca encontra-se instalada e configurada em um dos Cluster do LaCAD (UEFS). A ScaLAPACK está configurada em cada arquitetura distinta do Cluster, deste modo tornando-a otimizada e está disponível para utilização dos demais pesquisadores do laboratório.

Uma das atividade realizadas, foi o desenvolvimento de tutoriais sobre a biblioteca ScaLAPACK, disponibilizados no site do LaCAD(LACAD, 2015). O primeiro tutorial descreve a instalação da biblioteca e objetiva facilitar o processo em outros Clusteres ou uma possível reinstalação podendo ser encontrado no Anexo 1 do presente relatório. No Anexo 2 podemos verificar o tutorial sobre a utilização da biblioteca, que busca facilitar a pesquisa e implementação de códigos de Álgebra Linear para os pesquisadores, tanto do laboratório quanto para pesquisadores externos.

Um estudo sobre funções do ScaLAPACK para busca de autovalores foi realizado, objetivando encontrar a função de menor complexidade para auxiliar na resolução do problema desenvolvido pela professora Fernanda Castelo Branco, mestranda do Programa de Pós-Graduação em Mecatrônica da Universidade Federal da Bahia, o qual implementa métodos de Modelagem Molecular utilizando a teoria da Mecânica Quântica. A função, encontrada (PDSYEVR) utiliza um algoritmo para a busca de autovalores chamado de MRRR e tem complexidade $O(n^2)$, reduzindo o custo computacional da execução do código.

Depois de encontrar a função PDSYEVR, alguns testes foram realizados buscando verificar a escalabilidade do algoritmo e da biblioteca. Foram realizadas três classes de testes modificando a quantidade de elementos da matriz(25000000, 100000000 e 400000000 elementos) na qual os autovalores devem ser encontrados.

Para cada caso de teste o número de processadores (*computer nodes* ou nós) é aumentado e o tempo de execução da função é calculado para cada quantidade de nós (2, 4, 8, 16). Devido aos tamanhos das matrizes, o último caso de teste só foi executado com 8 e 16 nós, em virtude de problemas com a memória necessária para a execução da função. Após mensurar os tempos, a aceleração (speedup) e a eficiência do aumento dos processadores é verificada, buscando encontrar os melhores parâmetros para a execução da função.

Como a ScaLAPACK é baseada em uma execução paralela e uma versão serial dos testes não foi desenvolvida a aceleração e a eficiência são tomadas levando em consideração a quantidade de nós da primeira execução de cada classe de testes (dois para os dois primeiros casos e oito para o último).

Abaixo são mostrados os resultados obtidos nos testes realizados. Os tempos apresentados

são a média dos três valores intermediários entre cinco testes realizados (o maior e o menor valor de cada teste é desconsiderado da média).

Tabela 1: Resultados obtidos para uma matriz quadrada com 5000x5000 elementos.

Nós	Média(s)	Aceleração (em relação a 2 nós)	Eficiência (em relação a 2 nós)
2	186,9450156667	1	1
4	109,7106353333	1,7039826184	0,8519913092
8	95,9052223333	1,9492683622	0,4873170906
16	100,2580543333	1,8646383765	0,2330797971

Na Tabela 1 pode-se verificar a execução da função PDSYEVR, na busca por autovalores em uma matriz de 5000x5000 elementos. Devido a comunicação intensa entre os nós, provocado pela troca de mensagens, nem sempre um aumento no número de processadores reduz o tempo de cômputo, como é possível observar na tabela na qual, a execução com 8 nós teve uma média de tempo melhor que a execução com 16 nós. Na Tabela 2 é apresentado o resultado da execução da função com uma matriz de 10000x10000 elementos.

Tabela 2: Resultados obtidos para uma matriz quadrada com 10000x10000 elementos.

Nós	Média(s)	Aceleração (em relação a 2 nós)	Eficiência (em relação a 2 nós)
2	1218,7855326667	1	1
4	630,1721386667	1,9340517581	0,967025879
8	504,8001443333	2,4143922032	0,6035980508
16	402,6640146667	3,0268051981	0,3783506498

Na Tabela 2 é possível verificar que para o tamanho de matriz utilizado, o aumento na quantidade de nós reduziu o tempo de execução, porém a eficiência da execução é reduzida para menos de 50%. Essa redução da eficiência fica mais evidenciada na Figura 3. Já na Tabela 3 pode-se observar o resultado da execução da função com uma matriz de 20000x20000 elementos.

Tabela 3: Resultados obtidos para uma matriz quadrada com 20000x20000 elementos.

Nós	Média(s)	Aceleração (em relação a 8 nós)	Eficiência (em relação a 8 nós)
8	3133,6985403333	1	1
16	1801,4344633333	1,7395573384	0,8697786692

Com o aumento na quantidade de elementos o consumo de memória aumenta e mais computadores são necessários para executar o algoritmo. Na Tabela 3 percebe-se que devido ao tamanho da matriz, são realizados testes apenas com 8 e 16 nós. Nesta tabela, pode-se verificar que para matrizes muito grandes a aceleração do algoritmo é alta e a eficiência não reduz muito, sendo que o aumento de 8 para 16 nós possui uma eficiência em torno de 86% no caso em estudo.

A partir dos resultados mostrados nas tabelas, os gráficos apresentados nas Figuras 2 e 3 foram gerados utilizando o software Octave, mostrando a aceleração (Speedup) e a eficiência da função levando em consideração o número de nós do primeiro teste (2 ou 8 nós) em cada caso de teste:

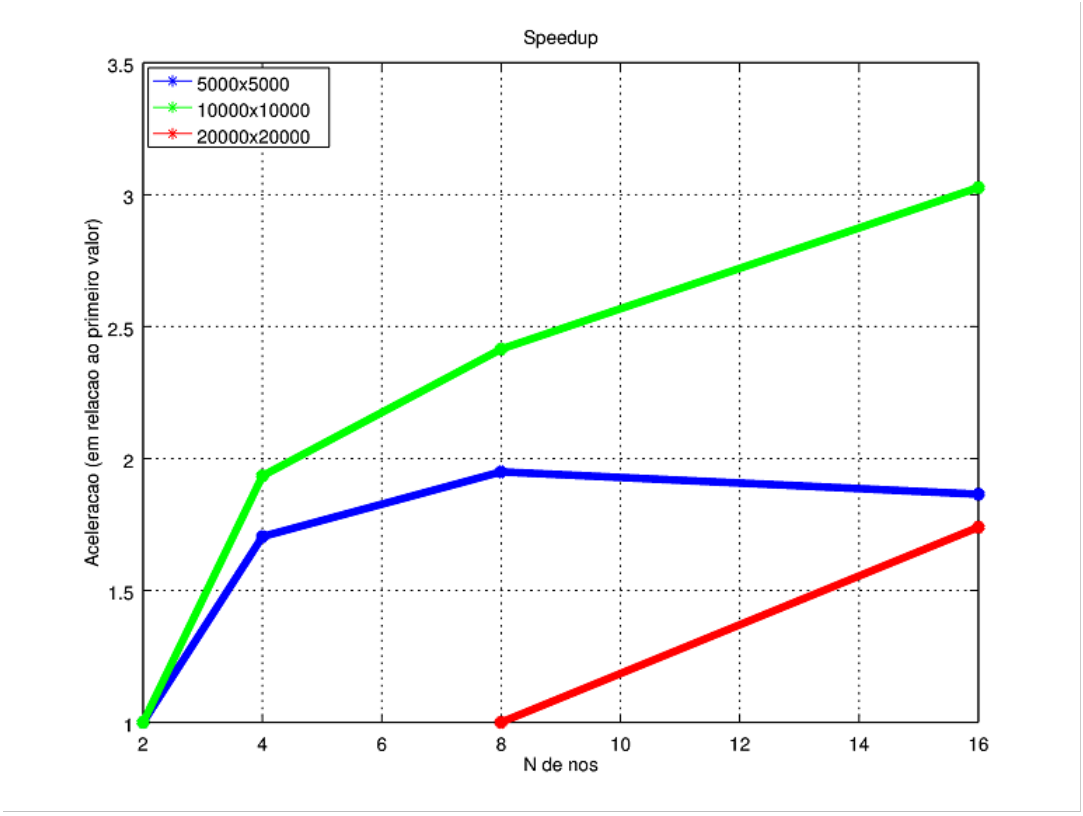

Figura 2: Gráfico da aceleração da função com o aumento do número de nós.

O primeiro caso de cada classe de teste inicia com aceleração igual a 1, pois ele é tomado como base para os próximos testes. Em todos os testes realizados os parâmetros utilizados foram os mesmos. Como observado no gráfico da aceleração (Figura 2) e na Tabela 2, o aumento do número de nós pode piorar o desempenho (queda na aceleração de 8 para 16 nós para matrizes 5000x5000). Essa piora pode ser explicada pelo aumento na comunicação entre os nós, provocada pelo aumento de nós. Pode-se verificar também que os resultados da aceleração são maiores para a matriz 10000x10000, mostrando que em matrizes pequenas um aumento da quantidade de recursos utilizados nem sempre é vantajosa. Ajustes nos parâmetros da função podem melhorar ou piorar o desempenho diminuindo a comunicação entre processos.

Apesar de existir uma aceleração mostrada na Figura 2, a Figura 3 nos mostra que o aumento no número de nós reduzem muito a eficiência do algoritmo, principalmente em matrizes menores. No caso de matrizes 5000X5000, por exemplo, a eficiência caiu para menos de 50% com o aumento de 2 para 8 nós. Essa redução da eficiência mostra que os recursos alocados, não estão melhorando muito no resultado e poderiam estar sendo utilizados para resolver

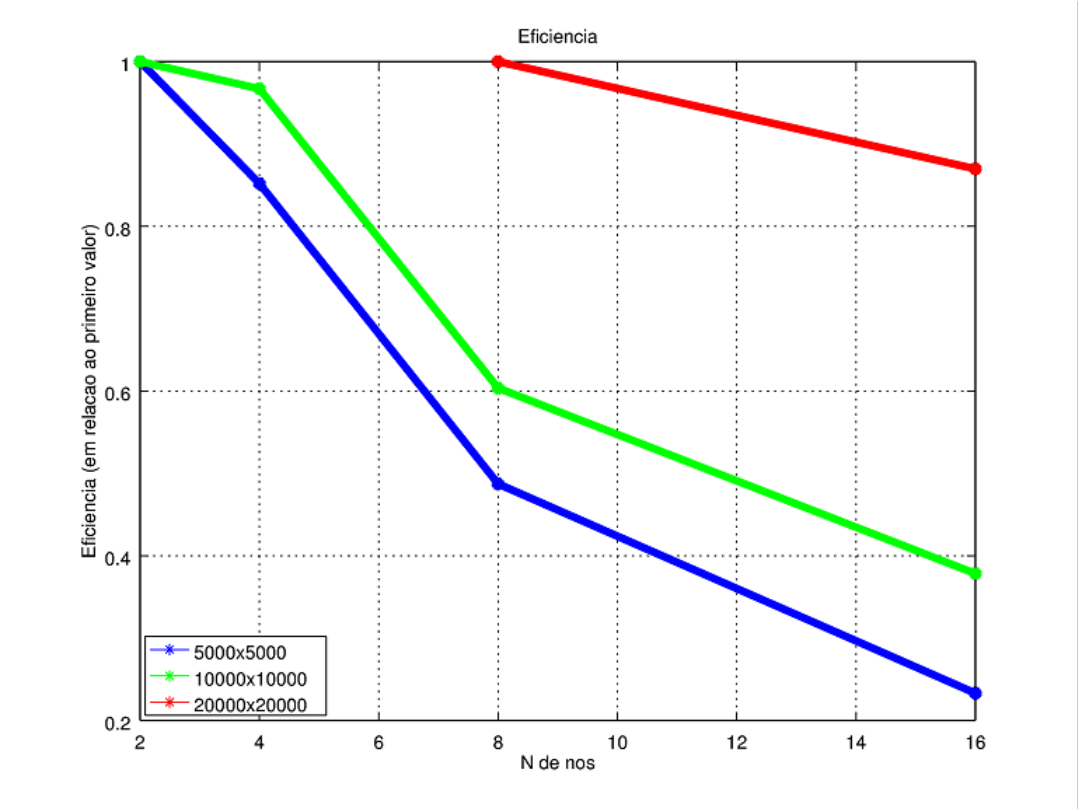

Figura 3: Gráfico da eficiência da função com o aumento do número de nós.

outros problemas.

Os testes realizados com a função mostraram que a quantidade de nós utilizados durante a execução do algoritmo, irá depender do tamanho que a matriz pode alcançar, sendo que utilizar quantidades menores pode ser mais eficiente, desde que a memória dos computadores suporte alocar a matriz a ser operada. Para alcançar resultados rapidamente de forma eficiente diversos parâmetros da função podem ser manipulados para melhorar a execução, entre eles o tamanho e a organização da grade de processos e o fator de bloco da distribuição da matriz.

5 Discussão

Este trabalho foi desenvolvido como o intuito de incluir a biblioteca ScaLAPACK entre as ferramentas do LaCAD, buscando estudar e desenvolver competências na aplicação das funções da biblioteca no desenvolvimento de códigos para computação científica, além de investigar a forma mais eficiente de integrar a biblioteca no trabalho da mestranda Fernanda Castelo Branco (orientada pelo coordenador do projeto), relacionado à implementação de métodos computacionais de alto desempenho para o cálculo da configuração molecular de três átomos, através do uso da teoria da Mecânica Quântica. Um dos objetivos do trabalho era também criar tutoriais e prover treinamentos para facilitar a utilização da biblioteca pelos futuros usuários do Laboratório.

A instalação e configuração da biblioteca de software ScaLAPACK no LaCAD permite que os usuários a utilizem sem se preocupar futuramente com as etapas de instalação, configuração e teste, além de eventuais problemas que esses passos possam causar, permitindo que a biblioteca seja utilizada diretamente por usuários com acesso ao Cluster. O uso posterior da biblioteca também foi facilitado pelo desenvolvimento dos tutoriais, os quais permitem que usuários sem experiencia com a mesma possam resolver seus problemas sem necessitar de novas pesquisas sobre seu funcionamento, encontrando as informações diretamente nos tutoriais. A compilação separada para as diferentes arquiteturas do Cluster permite que o mesmo trabalhe de forma otimizada em todos os nós, possibilitando também, o uso do ScaLAPACK para executar, de forma otimizada, diferentes trabalhos nas diferentes arquiteturas.

A infraestrutura do LaCAD auxiliou o desenvolvimento das atividades em aspectos como: Cluster montado e configurado pronto para uso; possibilidade de acesso remoto ao laboratório permitindo realizar tarefas externamente. Alguns problemas também podem ser citados, como: falta de acesso ao Cluster devido a falta de energia e problemas com a rede do Campus.

A função PDSYEVR foi escolhida para resolver o problema de autovalores da matriz simétrica por ser desenvolvida com o algoritmo com melhor complexidade para matrizes simétricas, reduzindo o tempo de execução do algoritmo. O ScaLAPACK possui outras funções para busca por autovalores, desenvolvidas com algoritmos diferentes que possuem complexidades maiores, para diferentes tipos de matrizes, que podem ser encontradas em sua documentação (BLACKFORD et al., 2015b).

Nos testes realizados com a função foi possível verificar que a aceleração nem sempre é proporcional ao número de processadores, podendo sofrer variações, às vezes até negativas, com o aumento no número de nós. Problemas como uso de *swap* ou alta taxa de comunicação pela rede podem afetar o desempenho da biblioteca. Para se alcançar o melhor desempenho

possível para um determinado tamanho de matriz, testes e ajustes de parâmetros devem ser realizados até que sejam encontrados os melhores parâmetros para esse tamanho determinado. Além do número de nós, o ScaLAPACK permite que o fator de bloco, tamanho do maior bloco de dados distribuídos, e a grade de processos possam ser alterados, podendo ocasionar mudanças significativas na performance do algoritmo em desenvolvimento.

Outro fator importante a ser considerado é a eficiência no uso dos recursos, pois mesmo apresentando aumento na aceleração, a eficiência no uso dos recursos pode ser muito baixa, não justificando o aumento do número de nós. Uma justificativa para a utilização de mais nós seria que, para tamanho da matriz muito grande, a memória dos nós não conseguem alocar a matriz local quando poucos computadores são utilizados, mesmo utilizando a *swap*. Outra justificativa seria diferença da velocidade entre memória e disco, então mais nós podem ser adicionados para tentar reduzir o uso da *swap*.

6 Conclusão

O trabalho, aqui descrito, foi desenvolvido durante a vigência da bolsa de iniciação científica PIBIC-CNPq, no qual foi realizada a instalação, configuração e viabilização da biblioteca ScaLAPACK em um Cluster do LaCAD, além da pesquisa e testes da função PDSYEVR, para busca de autovalores em matrizes, a ser utilizada no trabalho da mestranda Fernanda Castelo Branco. Os tutoriais disponibilizados nos anexos 1 e 2 e no site do LaCAD(LACAD, 2015), podem ser utilizados para quem está começando no ambiente fazer a instalação e utilizar a biblioteca.

O uso de uma biblioteca como a ScaLAPACK permite uma maior velocidade na codificação e na resolução de problemas de computacionais, devido as funções já previamente desenvolvidas e testadas. O programador pode resolver seu problema sem se preocupar com problemas como resolução de sistemas lineares ou busca de autovalores de uma matriz e questões como divisão dos dados e comunicação entre processos, no caso do ScaLAPACK.

Os resultados apresentados mostram que certos cuidados devem ser tomados na utilização dos recursos disponíveis para se conseguir uma aceleração, buscando garantir a eficiência na utilização dos mesmos. Para se descobrir a configuração com desempenho e eficiência, testes devem ser realizados levando em consideração o ambiente e o problema a ser resolvido.

O próximo passo desse trabalho é a utilização de outras bibliotecas de álgebra linear para outros tipos de sistema (Memória compartilhada ou placa gráfica), e bibliotecas que resolvem outros tipos de problema como resolução de equações diferencias, criando no LaCAD um ambiente propício para desenvolvimento e execução rápida de problemas de modelagem matemática que envolvem métodos numéricos. Outra possibilidade de estudo seria a utilização de uma ferramenta de monitoramento que permitisse encontrar os melhores parâmetros para a execução da biblioteca melhorando o desempenho do algoritmo em execução.

Referências

ANDERSON, E. et al. *LAPACK Users' Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999. ISBN 0-89871-447-8 (paperback).

BEOWULF. *Beowulf – Discussion of topics related to Beowulf clusters*. 2015. Disponível em: ⟨http://www.beowulf.org/mailman/listinfo/beowulf⟩. Acesso em: 17 Jul. 2015.

BLACKFORD, L. S. et al. *ScaLAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997. ISBN 0-89871-397-8 (paperback).

BLACKFORD, L. S. et al. *An Updated Set of Basic Linear Algebra Subprograms (BLAS)*. [S.l.]: ACM Trans. Math. Soft., 2002.

BLACKFORD, L. S. et al. *Data Distributions and Software Conventions*. 2015. Disponível em: ⟨https://www.netlib.org/scalapack/slug/node68.html\#SECTION04400000000000000000⟩. Acesso em: 02 Ago. 2015.

BLACKFORD, L. S. et al. *Eigenvalue Problems*. 2015. Disponível em: ⟨https://www.netlib.org/scalapack/slug/node119.html⟩. Acesso em: 20 Jul. 2015.

BLACKFORD, L. S. et al. *PDSYEVR. ScaLAPACK's Parallel MRRR Algorithm for the Symmetric Eigenvalue Problem*. 2015. Disponível em: ⟨https://www.netlib.org/lapack/lawnspdf/lawn168.pdf⟩. Acesso em: 02 Ago. 2015.

BLACKFORD, L. S. et al. *ScaLAPACK Scalable Linear Algebra PACKage*. 2015. Disponível em: ⟨http://www.netlib.org/scalapack/faq.html\#1.3⟩. Acesso em: 08 Ago. 2015.

BLACKFORD, L. S. et al. *ScaLAPACK Scalable Linear Algebra PACKage*. 2015. Disponível em: ⟨http://www.netlib.org/scalapack⟩. Acesso em: 17 Jul. 2015.

BLUM, V. et al. Ab initio molecular simulations with numeric atom-centered orbitals. *Computer Physics Communications*, v. 180, n. 11, p. 2175–2196, Novembro 2009.

CENTOS. *CentOS Project*. 2015. Disponível em: ⟨http://www.centos.org/⟩. Acesso em: 18 Jul. 2015.

DHILLON, I. S.; PARLETT, B. N.; VÕMEL, C. The design and implementation of the mrrr algorithm. *ACM Transactions on Mathematical Software*, v. 32, n. 4, p. 533–560, Dezembro 2006.

GEDIT. *Gedit*. 2015. Disponível em: ⟨https://wiki.gnome.org/Apps/Gedit⟩. Acesso em: 18 Jul. 2015.

INTEL. *ScaLAPACK Routines*. 2015. Disponível em: ⟨https://www.software.intel.com/pt-br/node/468348⟩. Acesso em: 03 Ago. 2015.

JOIOSO, A. L. B. et al. *Aplicação de computação em grade a simulações computacionais de estruturas semicondutoras*. [S.l.]: Universidade de São Paulo, 2008.

LACAD. *Lacad, Laboratório de Computação de Alto Desempenho*. 2015. Disponível em: ⟨https://www.sites.google.com/site/labsupercomp/⟩. Acesso em: 18 Jul. 2015.

MPI. *Message Passing Interface Forum*. 2015. Disponível em: ⟨http://www.mpi-forum.org/⟩. Acesso em: 17 Jul. 2015.

NETLIB. *Netlib Repository at UTK and ORNL*. 2015. Disponível em: ⟨http://www.netlib.org/⟩. Acesso em: 08 Ago. 2015.

OCTAVE. *Gnu Octave*. 2015. Disponível em: ⟨https://www.gnu.org/software/octave/⟩. Acesso em: 18 Jul. 2015.

OPENMPI. *Open MPI: Open Source High Performance Computing*. 2015. Disponível em: ⟨http://www.open-mpi.org/⟩. Acesso em: 17 Jul. 2015.

ROCKS. *Rocks: Open-Source Toolkit for Real and Virtual CLusters*. 2015. Disponível em: ⟨http://www.rocksclusters.org/wordpress/⟩. Acesso em: 18 Jul. 2015.

VIM. *Vim the editor*. 2015. Disponível em: ⟨http://www.vim.org⟩. Acesso em: 18 Jul. 2015.

YOGINATH, S. et al. Rscalapack: High-performance parallel statistical computing with r and scalapack. *Proceedings of the ISCA 18th International Conference on Parallel and Distributed Computing Systems*, 2005.

Declaração

Declaro que desenvolvi as atividades constante do plano de trabalho do projeto Integração da biblioteca ScaLAPACK em código para cálculo de modelagem molecular usando Mecânica Quântica submetido ao PIBIC/CNPq cujos resultados finais constam do presente relatório.

Feira de Santana, Agosto de 2015

Bolsista

Declaração

Declaro que acompanhei as atividades do bolsista Lucas Santana Carneiro do plano de trabalho do projeto Integração da biblioteca ScaLAPACK em código para cálculo de modelagem molecular usando Mecânica Quântica submetido ao PIBIC/CNPq, cujos resultados finais constam do presente relatório.

Feira de Santana, Agosto de 2015

Orientador

Anexo 1

Instalação do ScaLAPACK

1. Instalação

   (a) Baixar ScaLAPACK

   ```
   wget http://www.netlib.org/scalapack/scalapack_installer.tgz
   ```

   (b) Descompactar o arquivo

   ```
   tar -vzxf scalapack_installer.tgz
   ```

   (c) Entrar na pasta descompactada

   ```
   cd scalapack_installer_1.0.2/
   ```

   (d) Executar o scritp (–downall (Download das dependências) –prefix (Diretório para a instalação))

   ```
   ./setup.py --downall --prefix=/export/apps/scalapack
   ```

2. Executando com ScaLAPACK

   (a) Compilando

   ```
   mpicc -o example example.c -I/export/apps/scalapack/include
   -L/export/apps/scalapack/lib  -lscalapack -ltmg -lreflapack
   -lrefblas -lgfortran
   ```

   (b) Executando

   ```
   mpirun -np 4 example
   ```

Anexo 2

Tutorial de Utilização de Pacotes de Álgebra Linear para Alto Desempenho (ScaLAPACK)

1. ScaLAPACK (Scalable Linear Algebra PACKage)

   (a) Os quatro passos básicos requeridos para a chamada de rotinas ScaLAPACK

   - Inicializa a grade de processos
   - Distribui a matriz na grade de processos
   - Chamada a rotina ScaLAPACK
   - Liberar a grade de processos

   (b) Passo 1:

   Inclusão de bibliotecas necessárias para o funcionamento das funções ScaLAPACK no código

   - mpi.h

   (c) Passo 2: Inicializar a Grade de Processos;

   - `Cblacs_pinfo( &mype, &npe);`

     Captura informação sobre o processador, mype, e o numero de processadores, npe.

   - `Cblacs_gridinit( &context, "c", ncolunas, nlinhas);`

     Inicializa a grade de processos, com numero de linhas, nlinhas, e numero de colunas, ncolunas, definidas pelo usuário.

   (d) Passo 3: Distribuir a matriz na grade de processos

   - Inicializa descritores da matriz (descinit())
   - O descritor depende do tipo de matriz, variando o tamanho do descritor e os parâmetro divididos em:
     - Matrizes densas in-core:
       ⟨http://netlib.org/scalapack/slug/node77.html⟩
     - Matrizes de banda estreita ou tridiagonais:
       ⟨http://netlib.org/scalapack/slug/node86.html⟩
     - Matrizes de vetores RHS:
       ⟨http://netlib.org/scalapack/slug/node87.html⟩

  - Matrizes densas out-core:

    ⟨http://netlib.org/scalapack/slug/node91.html⟩

- Distribuir a matriz com a função

  `pdelset_()`

(e) Passo 4: Chamada a rotina ScaLAPACK (nome da função LAPACK precedida pela letra p.) Ex LAPACK:

`dgetrf_()`

Ex ScaLAPACK:

`pdgetrf_()`

(f) Passo 5:

- Liberar matrizes utilizados com

  `free()`

- Liberar a grade de processos

```
Cblacs_gridexit(context);
Cblacs_exit( 0 );
```

(g) Passo 6: Executando o código com ScaLAPACK

- Compilando

```
mpicc -o exemplo exemplo.c \
-I/opt/scalapack/lib \
-L/opt/scalapack/lib -lscalapack \
-ltmg -lreflapack -lrefblas -lgfortran
```

- Executando

```
nohup mpirun -np 4 -machinefile machinefile exemplo &
```