Studi Komputasi Paralel dan Implementasinya pada Kasus Komputasi Matriks Besar

(*Study and Implementation of Parallel Computing on Computation Case of Big Size/Sparse Matrix*)

Mayzar Annas[1], I Gede Pasek Suta Wijaya[2], dan L. A. Syamsul Irfan[1]
[1]Jurusan Teknik Elektro, Fakultas Teknik, Universitas Mataram
Jl. Majapahit 62, Mataram, Lombok, NTB-INDONESIA
[2]Prodi. Teknik Informatika, Fakultas Teknik, Universitas Mataram
Jl. Majapahit 62, Mataram, Lombok, NTB-INDONESIA
Email: mayzar23@gmail.com, gpsutawijaya@unram.ac.id, irfan@te.ftunram.ac.id

***Abstract*—Computation is the third pillar in the world of science to solve problems accurately and quickly. This study is an easy and inexpensive alternative of parallel computing using the library and utility of separate memory and processor of computers in local area network. The parallel computing is proposed instead of using traditional programming in a serial CPU instructions (can not be broken). In this case, the parallel computing is applied to solving problems on a big/sparse matrix such as determining inverse, solving a simultaneous linear equations, and eigen analysis. The experimental results show that the proposed system can break mentioned cases accurately and quickly compared with standalone programming (decreasing standalone computational time averagely by about 96.62% of all tested cases).**

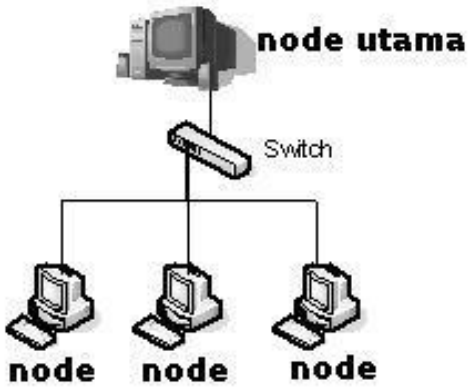
Key Words: Parallel computing, MPI (Message Passing Interface), Matrix, Amdahl Law.

I. PENDAHULUAN

Seiring dengan perkembangan pemikiran manusia, teknologi kemudian berpaling dari perangkat analog menjadi perangkat digital yang multifungsi dan memiliki kehandalan yang tinggi. Pada dunia sains, komputasi disetarakan sebagai pilar ketiga setelah analisis teorema dan eksperimen sains. Komputasi memungkinkan penyelesaian problem yang tidak bisa diselesaikan melalui eksperimen tradisional maupun teoritis, seperti prediksi iklim, pencarian formula obat-obatan hingga prediksi gempa.

Era tahun 80-an sampai dengan pertengahan 90-an, komputasi tingkat lanjut sangat tergantung pada super komputer (*supercomputer*). Seiring dengan meningkatnya kebutuhan komputasi yang cepat dan naiknya harga peralatan super komputer, maka pengembangan aplikasi-aplikasi berbasis paralel termasuk komputasi paralel didalamnya, untuk menyelesaikan permasalahan waktu komputasi pada model komputasi standalone semakin banyak dilakukan.

Pemanfaatan resource-resource komputasi yang tersedia secara paralel akan mempercepat eksekusi program. Program yang dieksekusi pada komputer master akan dibagi kepada node-node komputer untuk dimanfaatkan resourcenya. Komputasi ini sangat berguna untuk algoritma-algoritma yang memiliki tingkat pertumbuhan fungsi yang kuadratik dan eksponensial.



Gambar 1. Contoh sistem paralel.

Dengan konsep komputasi paralel, biaya investasi untuk super komputer bisa ditekan dan konsumsi energi listrik, biaya instalasi serta pemeliharaan juga menjadi lebih rendah. Lebih penting lagi, sistem ini fleksibel terhadap perubahan teknologi komputer yang sangat cepat. Alternatif populer saat ini adalah computer clustering (kelompok komputer) atau parallel *computer* (komputer paralel). Sistem ini merupakan penggabungan beberapa PC (disebut node) menjadi seolah-olah satu komputer dengan kemampuan yang lebih besar, seperti yang ditunjukkan pada Gambar 1.

II. TINJAUAN PUSTAKA

Komputasi paralel merupakan teknik komputasi menggunakan 2 atau lebih processor untuk meningkatkan performa komputasi dalam mengolah data besar/banyak[1]. Tidak mengherankan bila banyak industri jaringan, perusahaan besar dan developer tentunya menggunakan teknologi ini sebagai opsi mengatasi permasalahan yang ada. Penggunaannya sebagai alternatif disaat superkomputer dari segi harga tidak murah/terjangkau.

Beberapa perangkat bantu untuk SDK (sumber daya komputasi) berkinerja tinggi dan murah[2] adalah Parallel Virtual Machine (PVM), Message Passing Interface (MPI), Java Remote Method Invocation (RMI), serta Java

Common Object Request Broker Architecture (CORBA). Perangkat bantu tersebut diuji coba pada perangkat komputer pribadi (PC) untuk menghasilkan suatu kesimpulan perangkat mana yang cocok untuk dipakai oleh calon user nantinya.

Sementara, penggunaan teknologi Grid Computing untuk pemecahan suatu masalah science telah berhasil dilakukan[3]. Contoh kongkrit penggunaan komputasi GRID seperti analisa data high energy physics dan nuklir, riset iklim, serta analisa dan pencarian data pada masalah penemuan obat. Pada riset yang ia lakukan, dipaparkan penggunaan teknik pemrograman untuk komputasi paralel GRID dengan bantuan MPI dan GridPC, berikut implementasinya.

Thesis[4] yang berisi tentang implementasi komputasi paralel dengan komunikasi satu arah MPICH2 pada Infiniband. Infiniband adalah arsitektur komunikasi berkecepatan tinggi yang bertujuan digunakan untuk alat interkoneksi, seperti server, secondary storage, dan switch jaringan. Dalam thesis-nya ini, diterangkan pemetaan dari MPI versi 2 untuk komunikasi satu arah kepada Infiniband Remote Direct Access Memory (RDMA) dengan melihat performa yang dihasilkan

A. *Komputasi Paralel*

Komputasi paralel adalah salah satu teknik melakukan komputasi secara bersamaan dengan memanfaatkan beberapa komputer independen secara bersamaan. Ini umumnya diperlukan saat kapasitas komputasi yang diperlukan sangat besar, baik karena harus mengolah data maupun karena tuntutan proses komputasi yang banyak. Kasus lain yang umum ditemui yakni komputasi numerik untuk menyelesaikan persamaan matematis di bidang fisika (fisika komputasi), kimia (kimia komputasi), teknik sipil, dan bidang lainnya.

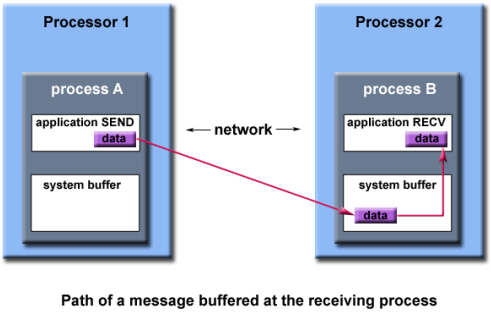
B. *Mesin Paralel*

Untuk melakukan aneka jenis komputasi paralel ini diperlukan infrastruktur mesin paralel (lihat Gambar 2) yang terdiri dari banyak komputer yang dihubungkan dengan jaringan dan mampu bekerja secara paralel untuk menyelesaikan satu masalah. Untuk itu diperlukan aneka perangkat lunak pendukung yang biasa disebut sebagai middleware yang berperan untuk mengatur distribusi pekerjaan antar node dalam satu mesin paralel.

C. *Pemrograman Paralel*

Pemrograman parallel[5], [6], [7], [8], [9] adalah teknik pemrograman komputer yang memungkinkan eksekusi perintah/ operasi secara simultan (komputasi paralel), baik dalam komputer dengan satu (prosesor tunggal) ataupun banyak (prosesor ganda dengan mesin paralel) CPU. Bila komputer yang digunakan secara bersamaan tersebut dilakukan oleh komputer-komputer terpisah yang terhubung dalam suatu jaringan komputer lebih sering istilah yang digunakan adalah sistem terdistribusi (*distributed computing*). Komunikasi data pada sistem pemrograman paralel di tunjukkan pada Gambar 3

Gambar 2. Mesin paralel.



Gambar 3. Aliran data pada mesin paralel.

D. *MPI*

MPI (Message Passing Interface)[10] adalah bahasa independen untuk protokol komunikasi yang digunakan untuk pemprograman paralel pada komputer. MPI dibuat oleh William Gropp, Ewing Lusk dan lainnya. MPI diterima banyak komunitas pemrograman paralel karena sifatnya independen sehingga realisasinya beragam.

E. *BCCD*

BCCD (*Bootable Cluster CD*) merupakan suatu live cd berbasis Debian yang berisi *tool-tool clustering*. *Cluster live CD* ini diciptakan untuk memfasilitasi kebutuhan konstruksi komputasi paralel dan keperluannya. Seringkali penggunaan resource untuk komputasi paralel yang rumit dan mengorbankan sistem operasi yang telah terpakai didalam node, oleh sebab itu kurangnya sumber daya dari server komputasi paralel dapat diakomodasi dengan penggunaan live CD cluster tanpa mengubah sistem yang ada. Live cd ini menyediakan fitur-fitur overlay untuk menjalankan lingkungan komputasi paralel secara penuh pada setiap komputer workstation baik PC maupun MAC.

III. METODE PENELITIAN

A. *Alat dan Bahan*

Alat dan bahan yang diperlukan pada penelitian ini dibagi menjadi dua bagian besar yaitu perangkat keras dan perangkat lunak. Perangkat keras yang diperlukan adalah:

1) Sebuah notebook komputer Lenovo T61 sebagai perangkat pembantu penelitian dalam pembuatan program standalone dan MPI.
2) Satu buah perangkat Switch, dalam penelitian ini digunakan Allied Telesyn-ATFS716 Fast Ethernet Switch.
3) Kabel-kabel UTP straight through, CAT 5E sebanyak 4 buah dengan tipe male to male.
4) Empat buah komputer sebagai node pengujian (1 node master dan 3 node slave), dalam hal ini dipergunakan 4 komputer yang bersifat homogen pada Laboratorium Komputasi Fakultas Teknik Universitas Mataram.
5) Empat buah Live CD, BCCD.

Sedangkan perangkat lunaknya adalah sebagai berikut:

1) Perangkat lunak coding (IDE) dan compiler C.
2) MPI library untuk C dan keperluannya.
3) SSH (Secure Shell).
4) Octave.
5) BCCD atau bootable cluster

Sepesifikasi perangkat keras untuk master dan klien dapat di tunjukkan pada Tabel I dan II .

Tabel I: Spesifikasi Komputer Master.

Perangkat	Keterangan
Processor	*Processor Intel Core2Duo T7300 2.00 GHz*
Memori	2048 Mbyte
Harddisk	500 Gbyte
Kartu *Ethernet*	Intel 82566MM Gigabit Network Connection
Sistem Operasi	Linux Mint 14 (Nadia)
Software Pendukung	GCC GNU C Compiler CodeBlocks IDE untuk C.OpenMPI 1.4 C library

Tabel II: Spesifikasi Komputer Klien.

Perangkat	Keterangan
Processor	*Processor Intel Core2Duo, E7300 2..66 GHz* (2667Mhz) 3072 KB Cache. ACPI Bios
Memori	2 Gbyte
Harddisk	160 Gbyte (Optional)
Kartu Ethernet	Realtek PCIe FE Family Controller RTLE8023
Drive Optik	LG HL-DT-ST DVDRAM GH22NS40
Sistem Operasi	BCCD Live CD

B. Prosedur Penelitian

Dalam penelitian ini digunakan tiga kasus penyelesain aljabar linear dengan ordo besar diatas 1000, yaitu:

1) Invers Matriks,
2) Penyelesaian Eigenvalue, dan
3) Penyelesaian persamaan linear simultan Ax=b.

Untuk ketiga kasus diatas menggunakan matriks yang dibangkitkan secara random. Metode yang dipakai untuk menyelesaikan kasus invers matriks dan Ax=b yakni metode gauss jordan, sementara pada penyelesaian eigenvalue digunakan metode power method[7]. Secara umum proses penelitian dapat dilahat pada diagram alir pada Gambar 4. Langkah yang dilalui dalam penelitian ini yakni:

1) Mempelajari algoritma untuk setiap kasus yang diberikan, melakukan perhitungan menggunakan Octave, mendeklarasikan data dan variabel yang dibutuhkan.
2) Menulis program untuk kasus yang diberikan dalam dalam bahasa C dan menyimpannya sebagai proses kompilasi A.
3) Mengujicoba program C apakah sudah tepat dan menghasilkan output yang benar dengan pembanding hasil perhitungan dengan menggunakan Octave.
4) Apabila sudah benar, maka tahap selanjutnya memodifikasi program kompilasi A dalam bahasa C dengan tambahan library MPI dan menyimpannya sebagai proses kompilasi B.
5) Waktu komputasi untuk setiap kasus dicatat untuk membuktikan bahwa sistem paralel dapat mempercepat proses komputasi dibandingan dengan komputasi standalone .

IV. Hasil dan Pembahasan

A. Penyusunan Perangkat

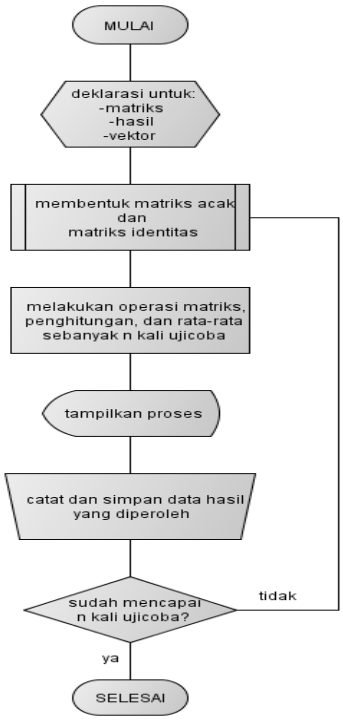
Perangkat penelitian yang digunakan berupa 4 buah komputer, KVM, Switch, beserta Kabel Jaringan disusun seperti pada Gambar 5.

B. Pengujian Node

Pengujian ini bertujuan untuk mengetahui apakah node yang terkoneksi dalam sistem paralel pada network bejalan atau tidak, dengan cara mengirimkan pesan secara berantai antar rank, mulai dari rank 0 hingga kembali ke rank 0, seperti yang ditunjukkan pada Gambar 6.

Fungsi yang digunakan dalam pengujian ini adalah MPI_Send pada pengiriman dan MPI_Recv pada sisi penerima pesan. Hasil dari pengujian dapat dilihat pada gambar screen shoot seperti pada Gambar 7.

Hasil pengujian menunjukkan bahwa setiap rank dapat saling berkomunikasi yang ditunjukkan dengan adanya tanda paket berhasil diterima dan rank aktif. Hal ini berarti bahwa sistem parallel sudah siap digunakan untuk menyelesaikan kasus numerik dengan matrik besar dengan komputasi parallel.

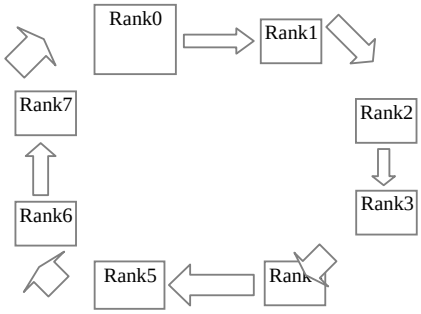


Gambar 4. Diagram alir proses penelitian.

Gambar 5. Susun perangkat keras paralel komputing pada penelitian ini.

C. Pengujian Kasus Invers Matriks

Pada kasus ini, invers matrik diselesaikan menggunakan metode gauss jordan dimana setiap iterasi sebanyak *N* ordo matriks dihitung dengan menggunakan baris pivot yang telah dikirimkan ke masing-masing rank. Dari pengujian terhadap program standalone dan program MPI didapatkan grafik, seperti pada Gambar 8.



Gambar 6. Aliran proses pengiriman pesan antar rank pada sistem paralel.

```
-bccd/testermpio-world$ mpirun -n 8 -machinefile ~/machines-openmpi
ah Processor Aktif : 8
 Processor : node010.bccd.net , Versi MPI 2.1

 Processor : node010.bccd.net , Versi MPI 2.1
l proses 8, pada rank 1 menerima pesan : rank master aktif! dari 0
 Processor : node011.bccd.net , Versi MPI 2.1
l proses 8, pada rank 2 menerima pesan : rank 1 aktif! dari 1
 Processor : node011.bccd.net , Versi MPI 2.1
l proses 8, pada rank 3 menerima pesan : rank 2 aktif! dari 2
 Processor : node000.bccd.net , Versi MPI 2.1
l proses 8, pada rank 4 menerima pesan : rank 3 aktif! dari 3
 Processor : node000.bccd.net , Versi MPI 2.1
l proses 8, pada rank 5 menerima pesan : rank 4 aktif! dari 4
 Processor : node009.bccd.net , Versi MPI 2.1
l proses 8, pada rank 6 menerima pesan : rank 5 aktif! dari 5
 Processor : node009.bccd.net , Versi MPI 2.1
l proses 8, pada rank 7 menerima pesan : rank 6 aktif! dari 6
 0 menerima pesan akhir dari ring broadcast : "rank 7 aktif!"
@node011:~/Hello-world$ xwd | convert - /tmp/flashdisk/s1.png
```

Gambar 7. *Screen shoot* hasil uji node.

D. Pengujian Kasus Matriks Eigenvalue

Untuk kasus kedua, nilai eigenvalue dihitung menggunakan metode power method dengan normalisasi kuadrat dari tiap baris matriks. Dari pengujian yang dilakukan didapatkan grafik seperti pada Gambar 9.

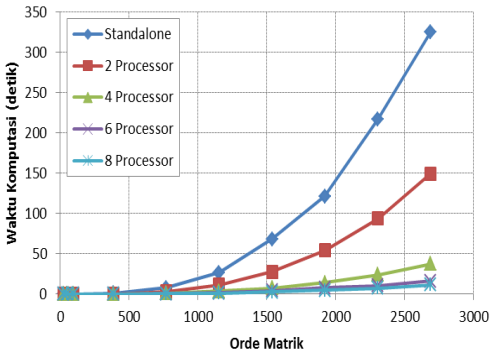
E. Pengujian Kasus Ax=b

Pada kasus ketiga ini, metode yang digunakan sama seperti pada kasus pertama yakni Gauss Jordan, namun data yang dikirimkan kembali oleh rank merupakan vektor bukan berupa matriks. Grafik yang dihasilkan dari pengujian pada kasus ketiga ini ditunjukkan Gambar 10.
.

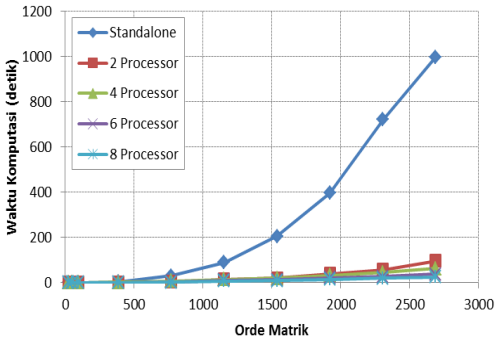
F. Diskusi

Program paralel untuk semua kasus pada penelitian ini berkonsep sama yakni memecah banyak data matriks yang kemudian disebar ke masing-masing node dan melakukan operasi aritmatika di tiap-tiap node.

Suatu program paralel tidak serta merta dapat menghasilkan *speedup* yang tinggi, ada beberapa bagian serial dari program yang tidak bisa dipecah dalam bentuk paralel. Sebagai contoh, untuk kasus 2 dengan menggunakan metode iterasi Jacobi. Metode ini menggunakan iterasi yang berkesinambungan artinya matriks dengan ukuran NxN yang diproses dalam iterasi harus menjalani fungsi-fungsi berkaitan satu sama lain. Sehingga peluangnya



Gambar 8. Waktu komputasi untuk penyelesaian invers matrik.



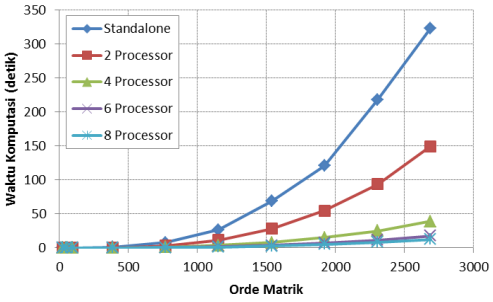
Gambar 9. Waktu komputasi penghitung nilai eigenvalue.

kecil untuk memecah data kemudian melakukan proses tersebut secara terpisah.

Metode pengalokasian memori secara dinamis dengan pointer dapat diterapkan pada kedua jenis pemrograman (standalone maupun paralel). Penggunaan pointer mengatasi keterbatasan array kedalam stack yang dapat menyebabkan kesalahan memori yakni segmentation fault error.

Pengukuran waktu komputasi pada program paralel menggunakan fungsi MPI_Wtime() untuk mencari selisih waktu komputasi dari masing-masing rank, kemudian ditotalkan dan dibagi dengan jumlah rank, sehingga didapatkan waktu rata-rata komputasi dari seluruh rank. Sementara pada program standalone, waktu komputasi dapat dilihat pada hasil running dari GCC secara langsung di terminal pada bagian akhir program dengan output execution time.

Untuk ketiga kasus yang diujicobakan, waktu komputasi yang diperlukan oleh pemrograman paralel sangatlah kecil dibandingkan dengan waktu komputasi standalone seperti yang ditunjukkan pada Gambar 8, 9, dan 10. Hasil uji tersebut juga menunjukkan bahwa semakin banyak processor yang digunakan dalam menyelesaikan persoalan yang diberikan maka semakin kecil waktu komputasi yang diperlukan (menurunkan waktu komputasi standalone untuk kasus invers matriks, eigen value



Gambar 10. Waktu komputasi penyelesaian Ax=b.

dan persamaan linesar simultan, secara beurutan sebesar 96.06%, 97.40%, dan 96.39%, untuk orde data 2688). Hal ini sebabkan oleh beban komputasi dibagi kedalam beberapa bagian yang desebar ke masing-masing rank. Hasil dari masing-masing rank di kombine di komputer master.

Dalam penelitian ini data yang digunakan belum sampai mengakibatkan titik jenuh dari hukum amdhal itu sendiri, apabila titik jenuh dicapai dari matriks berordo N maka jumlah percepatan antara prosesor (rank yang aktif) bisa diabaikan.

Kedua program baik program standalone maupun program paralel berbasis MPI yang dibuat dapat terjadi *segmentation fault*, biasanya terjadi karena besar data yang dideklarasikan melebihi batas memori maupun dikarenakan terdapat porsi data yang kosong dan kesalahan kirim-terima data pada rank (program MPI).

V. KESIMPULAN DAN SARAN

A. *Kesimpulan*

1) Sistem komputasi paralel menggunakan MPI telah berhasil diwujudkan untuk kasus penyelesaian numerik pada matrik besar.
2) Waktu komputasi yang dibutuhkan cenderung meningkat seiring bertambahnya jumlah ordo mariks yang diberikan, namun waktu komputasi yang diperlukan sangatlah kecil jiga dibandingkan dengan waktu komputasi standalone, rata-rata untuk ketiga kasus sebesar 16.80 detik (turun 96.62% dari waktu komputasi standalone) untuk orde data 2688.
3) Penggunaan lebih banyak rank (pekerja) dalam komputasi dapat memangkas waktu komputasi selaras dengan hukum Amdahl.
4) Program paralel memecah data yang besar dan diproses oleh rank-rank yang dipakai sehingga instruksi dapat dijalankan bersamaan (simultan).
5) Tidak semua bagian dari aplikasi *standalone* dapat diparalelkan. Sebagai contoh studi kasus yang memerlukan batch processing untuk suatu penyelesaian iterasi, peluang menjadikan program *standalone* tersebut ke pemrograman paralel adalah kecil.

B. Saran

1) Penggunaan Live CD (BCCD) bersifat portable, tidak direkomendasikan untuk penggunaan jangka panjang di Laboratorium komputasi. Sehingga untuk system yang lebih terpadu, disarankan melakukan instalasi.
2) Diperlukan studi mendalam untuk kasus-kasus lain dan pengimplementasiannya dalam paralel, sehingga dapat membantu mahasiswa dan para akademis dalam mencapai hasil penelitian.
3) Sistem ini terbuka untuk diimplementasikan pada sistem yang memerlukan komputasi yang besar seperti pada sub-space analisis dan sistem cerdas.

UCAPAN TERIMA KASIH

Terimakasih yang sebesar-besarnya kami ucapkan kepada Lab. Komputasi Fakultas Teknik Universitas Mataram atas bantuan peralatan untuk menyelesaikan penelitian ini.

DAFTAR PUSTAKA

[1] M. Susmikanti and W. Dewayatna, "Komputasi paralel eigenvalue dalam penyelesaian difusi multigroup menggunakan metoda householder deflasi dan divide conquer," in *Lokakarya Komputasi dalam Sains dan Teknologi Nuklir*, Oktober 2012, pp. 341–352.

[2] H. Suhartanto, "Kajian perangkat bantu komputasi parallel pada jaringan pc," *Makara, Teknologi*, vol. 10, no. 2, pp. 72–81, 2006.

[3] S. Pahlevi, "Komputasi grid dan paralel," in *Lokakarya Komputasi dalam Sains dan Teknologi Nuklir*, Agustus 2008, pp. 15–24.

[4] W. Jiang, "High performance mpich2 one-sided communication implementation over infiniband," Master's thesis, The Ohio State University, 2004.

[5] K. Agus, *Pemrograman Parallel dengan MPI dan C*. ANDI Yogyakarta, 2010.

[6] K. G. Em, I. I. Kirby, and M. Robert, *Parallel Scientific Computing in C++ and MPI*. Cambridge University Press, 2003.

[7] Jeroen, *Parallel implementation of the power method algorithm*. University of Antwerp, 2013.

[8] M. Quinn, *Designing Efficient Algorithms for Parallel Computers*. Singapore: Mc Graw Hill, International Editions, 1987.

[9] M. J. Quinn, *Parallel Computing (2Nd Ed.): Theory and Practice*. New York, NY, USA: McGraw-Hill, Inc., 1994.

[10] S. Dedy, *Uji Kinerja Octave dengan MPITB Menggunakan Pustaka LAM/MPI dan OpenMPI*. Yogyakarta: Universitas Gadjah Mada, 2010.