Terakreditasi SINTA Peringkat 2
Surat Keputusan Dirjen Penguatan Riset dan Pengembangan Ristek Dikti No. 10/E/KPT/2019
masa berlaku mulai Vol. 1 No. 1 tahun 2017 s.d Vol. 5 No. 3 tahun 2021

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 4 No. 5 (2020) 957 - 963 ISSN Media Elektronik: 2580-0760

Implementasi Algoritma Improvised Prioritized Deadline Scheduling Algorithm (IPDSA) pada Grid Environment Menggunakan PVM3

Haidar Hendri Setyawan[1], Wisnu Widiarto[2], Ardhi Wijayanto [3]
[123]Informatika, Universitas Sebelas Maret Surakarta
[1]haidarhendri@student.uns.ac.id, [2]wisnu.widiarto@staff.uns.ac.id*, [3]ardhi.wijayanto@staff.uns.ac.id

Abstract

Resource Scheduling is one of the most challenging parts of grid computing. A number of algorithms have been designed and developed to create effective resource scheduling. In this research, the algorithms that have been used are the improvised prioritized deadline scheduling algorithm (IPDSA), and the parallel virtual machine version 3 (PVM3) has been used for efficient task execution, with a deadline limit for each task. PVM3 is a software library that optimizes resources flexibly and heterogeneously on a computer. These resources have been connected to various architectures in parallel, so that they can complete tasks well, even though they are very large and complex. This research has implemented the IPDSA resource scheduling algorithm to optimize scheduling and Grid resources in a computer laboratory as a grid environment, where the computers (hosts) are the Grid resource. This research has also developed an IPDSA resource scheduling algorithm by giving priority to each task and implemented using PVM3. The IPDSA resource scheduling algorithm has been successfully implemented using PVM3, with average Tardiness showing a stable value and getting a Non-Delayed Task value above 97.3%, because the resources and tasks that are carried out can be distributed evenly according to the number of hosts used.

Keywords: Grid Computing, Resource Scheduling, Two-level Hierarchy Scheduling Model, PVM3, IPDSA

Abstrak

Resource Scheduling adalah salah satu bagian yang menantang dalam komputasi grid. Sejumlah algoritma telah dirancang dan dikembangkan untuk membuat resource scheduling yang efektif. Dalam riset ini, algoritma yang telah digunakan adalah algoritma improvised prioritized deadline scheduling algorithm (IPDSA), dan parallel virtual machine versi 3 (PVM3) telah digunakan untuk eksekusi task yang efisien, dengan batasan deadline dari setiap task. PVM3 merupakan library perangkat lunak yang mengoptimalkan sumber daya secara fleksibel dan heterogen pada komputer. Sumber daya tersebut telah saling terhubung dengan beragam arsitektur secara paralel, sehingga telah dapat menyelesaikan tasks dengan baik, meskipun berukuran sangat besar dan kompleks. Riset ini telah mengimplementasikan algoritma resource scheduling IPDSA untuk mengoptimalkan penjadwalan dan sumber daya Grid pada suatu laboratorium komputer sebagai grid environment, dimana komputer – komputer (hosts) tersebut sebagai sumber daya Grid. Penelitian ini juga telah mengembangkan algoritma resource scheduling IPDSA dengan memberikan prioritas pada setiap task dan diimplementasikan menggunakan PVM3. Algoritma resource scheduling IPDSA telah berhasil diimplementasikan menggunakan PVM3, dengan Tardiness rata-rata menunjukkan nilai yang stabil dan mendapatkan nilai Non-Delayed Task diatas 97,3%, karena sumber daya dan task yang dikerjakan dapat didistribusikan secara merata sesuai dengan jumlah host yang digunakan.

Kata kunci: Grid Computing, Resource Scheduling, Two-level Hierarchy Scheduling Model, PVM3, IPDSA

1. Pendahuluan

Grid merupakan suatu sumber daya komputer yang saling terhubung dan tersebar untuk mendistribusikan suatu layanan kepada pengguna. Komputasi pada *grid* pada dasarnya digunakan untuk mengkombinasikan kemampuan, *bandwidth* komunikasi dan kapasitas pemrosesan yang dapat bervariasi dalam sebuah sistem yang ada pada *grid* [1,2,3]. Semakin besarnya kebutuhan penggunaan teknologi informasi dan komunikasi dalam kehidupan sehari-hari membuat beban komputasi semakin besar. Hal ini diakibatkan oleh *task* atau tugas yang semakin banyak dan kompleks karena kebutuhan pengguna yang sangat tinggi dan dinamis. Karena keterbatasan *resourece* atau sumber daya komputasi yang dimiiliki, perusahaan mengandalkan teknologi komputasi *grid* untuk meninggatkan kualitas pelayanan

Diterima Redaksi: 10-09-2020 | Selesai Revisi : 18-10-2020 | Diterbitkan Online : 30-10-2020

dan menyelesaikan kebutuhan komputasi mereka [4,5,6].

Salah satu bagian dari komputasi Grid adalah *resource scheduling* [7]. *Resource scheduling* merupakan cara untuk menentukan *task* mana yang harus dikerjakan dan memilih *resource* untuk mengerjakan *task* tersebut berdasarkan pada parameter Q*uality of Service* (QoS) yang dibutuhkan. Bagian ini merupakan salah satu bagian terpenting pada *grid*, karena bertanggung jawab untuk memilih *virtual machine* yang optimal untuk melaksanakan tugas menggunakan algoritma *resource scheduling* dan bertanggung jawab untuk memeriksa apakah ada kendala dalam memenuhi QoS.

Terdapat beberapa algoritma *resource scheduling* yang terbagi menjadi 3 (tiga) skema, yaitu *heuristic scheduling*, *meta-heuristic scheduling*, dan *hybrid scheduling*. Algoritma dari masing-masing skema tersebut diataranya *Round Robin, First Come First Serve (FCFS), Deadline Constraint based, QoS based, Artificial Honey Bee, Genetic Algorthm, ACO+Max-min* [8]. Setiap algoritma memiliki kekurangan dan kelebihan masing-masing yang digunakan untuk menyelesaikan masalah tertentu dan hasil yang efisien dan optimal.

Penelitian-penelitian untuk menentukan *resource scheduling* telah banyak dilakukan, yaitu membandingan sebuah algoritma *resource scheduling* dan mengembangkannya serta membandingkan algoritma tersebut dengan algoritma-algoritma lainnya dengan mengujinya pada suatu perangkat simulasi. Perangkat simulasi yang banyak digunakan peneliti adalah CloudSim atau GridSim [9]. Mengembangkan algoritma *resource scheduling* dengan pendekatan pada perilaku semut yang mencari makanan dan diberi nama *Ant Algorithm-based* [10]. Mengembangkan Algoritma *Prioritized Deadline Scheduling Algorithm* (PDSA) dan *Earlier Deadline First* (EDF) dengan pendekatan pada batas tenggat waktu pekerjaan untuk dieksekusi yang diberi nama *Improvisied Prioritized Deadline Scheduling Algorithm* (IPDSA) [11].

Pada penelitian ini, *resource scheduling* dengan algoritma *resource scheduling* IPDSA diimplementasikan dalam teknologi *grid computing* menggunakan PVM3 pada studi kasus *parallel computing* untuk mengetahui kemampuan dari algoritma tersebut dalam menyelesaikan permasalahan *resource scheduling*.

2. Metode Penelitian

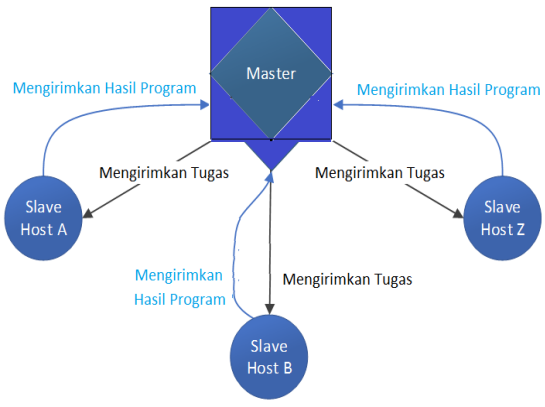
Pada Gambar 1 digambarkan proses jalannya program pada tahap pengujian ini. Pertama dari komputer-komputer yang dijalankan, ada satu komputer yang berperan sebagai *master*, yang bertugas untuk menjadwalkan *task*, menerima *tasks* dan membagi *task* tersebut untuk *host* lainnya yang berperan sebagai *slave*. *Host* yang berperan sebagai *slave* memiliki tugas untuk menyelesaikan bagian *scheduling tasks* yang diberikan oleh *master* dan mengirimkan hasilnya ke *master* lagi untuk digabungkan. *Master* bertugas untuk menggabungkan hasil pengerjaan task dari *slave* dan yang dikerjakan oleh *master*, sehingga mendapatkan hasil yang sesuai dengan desain program yang dibuat.

Pada penelitian ini dilakukan percobaan dengan jumlah *host* yang beragam, yaitu 7 host, 10 host, 15 host, 25 host, 27 host, 30 host, 33 host dan 40 host, berdasarkan sumber daya yang digunakan dengan jumlah dan spesifikasi sumber daya yang heterogen dan kondisi yang berbeda.

Pada bagian ini dijelaskan mengenai metode yang digunakan pada penelitian ini yaitu algoritma *resource scheduling* IPDSA dan PVM3 (*Parallel Virtual Machine* versi 3).

2.1. Algoritma resource scheduling IPDSA

Algoritma *Resource Scheduling* IPDSA (*Improvised Prioritized Deadline Scheduling Algorithm*) dikembangkan oleh Anubha Chauhan [11]. Model *resource scheduling* pada algoritma IPDSA pada langkah awal mengumpulkan resource kedalam sebuah list yang dihimpun sebagai *resource* untuk memproses *job* dan jumlah dari elemen pemrosesan digunakan untuk desain sistem tertentu. *Time Delay* mengalokasikan *resource* dari *task* yang merupakan perbedaan antara waktu eksekusi yang diharapkan dan *deadline* dari *task* yang dijalankan. Kebutuhan dari *task* mengenai tipe properti apa yang diinginkan sebagai *resource* dan berapa banyak prosesor yang diperlukan untuk melaksanakan *task* tersebut maka dibuatlah daftar dari *resource* untuk *task* yang dijalankan. Pemilihan yang tepat dilakukan berdasarkan kecepatan pemrosesan untuk *task-task* yang terdaftar. Prioritas diberikan kepada resource yang memiliki kecepatan pemrosesan maksimum untuk melaksanakan tugas sebelumnya.



Gambar 1. Ilustrasi Pengujian Implementasi Program

Algoritma *resource scheduling* IPDSA mengelompokkan *task* sesuai yang dijelaskan di atas dengan menggunakan nilai *Time Delay* yang merupakan perbedaan antara *deadline* yang diberikan kepada sebuah *task* dengan waktu eksekusi yang diharapkan dari *task* tersebut. Nilai *Time Delay* tersebut diperoleh dari persamaan (1).

$$TD_i = d_i - ET_i \quad (1)$$

Dimana d_i merupakan nilai *deadline* pada *task i*, dan ET_i merupakan *Expected Execution Time* pada satu kali proses program dijalankan. Kemudian menghitung *Total Tardiness* yang merupakan penjumlahan dari *tardiness* untuk setiap *task* dan membaginya dengan jumlah *total task*, lalu didapatkan *average tardiness*, yang diberikan oleh persamaan (2).

$$ATR = \frac{\sum_{i=1}^{n} TR_i}{n} \quad (2)$$

Dimana ATR_i (*average tardiness*) yaitu rata – rata *tardiness* pada pada setiap *task i* yang dijalankan dalam satu kali proses program dijalankan, TR_i adalah nilai *Tardiness* dan n adalah banyaknya *task* yang dijalankan dalam satu kali proses program dijalankan. Sedangkan *Tardiness* (*TR*) merupakan waktu yang tertunda antara *deadline task* dan waktu selesai *task* tersebut. Nilai *Tardiness* tersebut didapatkan dari persamaan (3).

$$ALM_i = \frac{1}{m}\sum_{i=1}^{m} ALC_i \quad (3)$$

Dimana ALMi (average load machine system) yaitu rata – rata load pada level machine, ALCi (average load cluster) yaitu rata – rata load pada setiap cluster, dan m adalah banyaknya cluster pada suatu sistem di level machine. Nilai ALCi didapatkan dari persamaan (4).

$$TR_i = d_i - F_i \quad (4)$$

Dimana d_i merupakan nilai *deadline* pada *task i*, dan F_i merupakan *Finish Time* pada setiap *task i* yang dijalankan dalam satu kali proses program dijalankan. Kemudian parameter yang perlu diperhatikan adalah Non-delayed tasks yang merupakan jumlah dari *task* yang waktu penyelesaiannya kurang dari *deadline task* tersebut atau dapat diartikan sebagai *"Total jumlah task yang diselesaikan sebelum deadline"*.

Pada algoritma ini parameter yang dibutuhkan antara lain: *taskID, expected execution time, deadline* dan *number of host.* Time delay untuk setiap *task* dihitung menggunakan persamaan (1). *Task* pada antrian yang telah siap disusun secara *ascending* berdasarkan pada *computed time delay* dari setiap task. Jika *task* memiliki *computed time delay* yang sama, maka *task* tersebut harus disusun berdasarkan metode *First Come First Serve* (FCFS) untuk setiap antrian yang telah siap. Kemudian mengurutkan *task* berdasarkan *time delay* pada urutan secara *ascending* (*task* dengan *time delay* paling sedikit diberikan prioritas) dan memilih *task* yang memiliki *time delay* paling sedikit untuk dieksekusi. Akhirnya, *finish time* dan *tardiness* dari setiap *task* didapatkan. Langkah–langkah algoritma *resource scheduling* IPDSA [11], dapat ditunjukkan pada algoritma:

```
Algorithm : Resource Scheduling IPDSA
Input: daftar proses yang siap untuk dieksekusi
dengan taskID, arrival time, expected execution
time, number of host dan deadline
Output: Finish Time, Time Delay, Average
Tardiness, Non-Delayed Tasks
1. Begin
2. For all tasks in the queue
   Menghitung time delay dari seluruh task
   menggunakan persamaan (1)
   Menyusun task sebagai ready queue yang
   diurutkan secara ascending berdasarkan pada
   computed time delay.
3. If (TDi = TDj)
4. Menyusun Ti, Tj berdasarkan pada FCFS
5. End if
6. While (ready queue tidak kosong)
7. do
8. {
9. Menampilkan daftar prosesor yang sesuai
   untuk task tersebut
10.Memilih prosesor dengan kecepatan pemrosesan
   terbaik di antara prosesor yang sesuai
11.Menjalankan task sesuai dengan computational
   length
12.Menghitung nilai Finising Time (Fi) dari
   task
13.Menghitung nilai Tardiness menggunakan
   persamaan (3)
14.If (di > Fi)
15.Then
16.Increment the Non-Delayed job counter
17.End if
18.}
19.Menghitung average tardiness menggunakan
   persamaan (2)
20.End
```

2.2. *Parallel Virtual Machine* versi 3 (PVM3)

PVM3 atau *Parallel Virtual Machine version* 3 adalah seperangkat alat dan *library* perangkat lunak terintegrasi yang mengemulsikan kerangka kerja konkuren yang pertujuan umum, fleksibel, dan heterogen pada komputer yang saling berhubung dengan beragam arsitektur [12,13]. Secara singkat, prinsip-prinsip yang mendasari PVM3 meliputi:

1. *User-configured host pool*, Tugas komputasi aplikasi dijalankan pada serangkaian mesin yang dipilih oleh pengguna untuk menjalankan program PVM3 tertentu. Baik mesin CPU tunggal dan multiprosesor perangkat keras (termasuk komputer memori bersama dan memori terdistribusi) dapat menjadi bagian dari kumpulan host. Host pool dapat diubah dengan menambah dan menghapus mesin selama operasi (fitur penting untuk toleransi kesalahan).
2. *Translucent access to hardware*, Program aplikasi dapat memandang lingkungan perangkat keras sebagai kumpulan elemen pemrosesan virtual yang tidak dapat dikaitkan atau dapat memilih untuk mengeksploitasi kemampuan mesin tertentu di kumpulan host dengan memposisikan tugas komputasi tertentu pada komputer yang paling tepat.

3. *Process-based computation*, Unit paralelisme dalam PVM adalah task, sebuah rangkaian kontrol berurutan independen yang berganti-ganti antara komunikasi dan komputasi. Tidak ada pemetaan proses-ke-prosesor yang tersirat atau ditegakkan oleh PVM; khususnya, banyak tugas dapat dijalankan pada satu prosesor.
4. *Explicit message-passing model*, Kumpulan tugas komputasi, masing-masing melakukan bagian dari workload aplikasi menggunakan data-, fungsional-, atau dekomposisi hibrid, bekerja sama dengan secara eksplisit mengirim dan menerima pesan satu sama lain. Ukuran pesan hanya dibatasi oleh jumlah memori yang tersedia.
5. *Heterogeneity support*, Sistem PVM mendukung heterogenitas dalam hal mesin, jaringan, dan aplikasi. Sehubungan dengan pengiriman pesan, PVM memungkinkan pesan yang mengandung lebih dari satu tipe data untuk dipertukarkan antara mesin yang memiliki representasi data berbeda.
6. *Multiprocessor support*, PVM menggunakan fasilitas native message-passing pada multi-prosesor untuk mengambil keuntungan dari perangkat keras yang mendasarinya. Vendor sering menyediakan PVM mereka sendiri yang dioptimalkan untuk sistem mereka, yang masih dapat berkomunikasi dengan versi PVM publik.

Sistem PVM terdiri dari dua bagian. Bagian pertama adalah *daemon*, yang disebut *pvmd3* dan kadang-kadang disingkat *pvmd*, yang berada di semua komputer yang membentuk *virtual machine*. Pvmd3 dirancang setiap *user* dengan login yang valid dapat menginstal *daemon* ini pada mesin. Ketika *user* menjalankan aplikasi PVM, ia pertama kali menciptakan *virtual machine* dengan memulai PVM. Aplikasi PVM kemudian dapat dimulai dari *prompt (command promt* atau *terminal)* Unix pada *host* manapun. Beberapa *user* dapat mengonfigurasi *virtual machine* yang tumpang tindih, dan setiap *user* dapat menjalankan beberapa aplikasi PVM secara bersamaan.

Bagian kedua dari sistem adalah *library* dari PVM *interface routines*. Ini berisi *functionally complete reporoire of primitives* yang diperlukan kerja sama antara *task* aplikasi. *Library* ini berisi *routines* yang dapat dipanggil *user* untuk pemindahan message, *spawning process*, koordinasi *tasks*, dan modifikasi *virtual machine.*

3. Hasil dan Pembahasan

Bagian ini membahas hasil percobaan implementasi algoritma *resource scheduling* IPDSA menggunakan PVM3 pada *Grid Environment.*

3.1. *Grid Environment*

Algoritma IPDSA diimplementasikan pada percobaan ini dilakukan menggunakan 40 komputer yang berada di Laboratorium Komputer S1 Informatika Universitas Sebelas Maret. 40 komputer tersebut dioperasikan menggunakan sistem operasi Ubuntu 18.04.4 LTS *(Long Term Support)* yang ter-*install native* menggunakan desktop environment GNOME *(GNU Network Object Model Environment).* Spesifikasi komputer yang digunakan yaitu CPU 64-bit, RAM 8 GB, Harddisk 1 TB, dengan sistem operasi Ubuntu 18.04.4 LTS. Konfigurasi jaringan dalam pengujian ini menggunakan network interface setiap unit komputer dan terkoneksi antar komputer secara lokal.

3.2. Hasil Percobaan Implementasi Program

Pada percobaan tahap ini yaitu menguji Algoritma *Scheduling* IPDSA menggunakan 1 program dengan variasi jumlah *task* dan variasi jumlah *host* yang terhubung. Program yang dijalanakan berupa program yang melakukan proses generator 100.000 angka secara acak, lalu angka-angka tersebut diurutkan menggunakan algoritma *sorting* Bubble Sort. Pada tahap ini, hasil percobaan diukur menggunakan *Expected Execution Time, Deadline, Time Delay, Finishing Time, Tardiness,* nilai *Average Tardiness* dan nilai *Non-Delayed Job.*

Tabel 1 menunjukkan jumlah ATR, jumlah NDT dan persentase NDT dari *task* yang dijalankan untuk satu kali proses dengan 7 *host*. Percobaan dilakukan sebanyak 5 kali proses yang dijalankan pada satu kali eksekusi program. Tabel 2 menunjukkan jumlah ATR, jumlah NDT dan persentase NDT dari *task* yang dijalankan untuk satu kali proses dengan 15 *host*. Percobaan dilakukan sebanyak 5 kali proses yang dijalankan pada satu kali eksekusi program. Demikian juga untuk Tabel 3 dengan 33 *host* dan Tabel 4 dengan 40 *host*.

Berdasarkan hasil pengujian program diperoleh hasil pada Tabel 1, Tabel 2, Tabel 3 dan Tabel 4, terdapat improvisasi terhadap metode penjadwalan algoritma *task scheduling* IPDSA yaitu pada *environment* yang digunakan, dimana pada penelitian sebelumnya menggunakan GridSim maupun CloudSim lalu pada penelitian ini dimodifikasi menggunakan PVM3. Pada hasil pengujian menunjukkan, bahwa dalam menjalankan algoritma *resource scheduling* memiliki 3 kriteria.

Tabel 1. Hasil Percobaan dengan 7 Host

No.	Uji ke-	Jumlah Task	ATR (ms)	Jumlah NDT	NDT
1	1	1000	19651,00	987	98,700%
2		2000	17832,89	1966	98,300%
3		3000	17688,07	2975	99,167%
4		4000	21101,64	3967	99,175%

5	2	1000	21250,30	987	98,700%
6		2000	21841,76	1991	99,550%
7		3000	17753,47	2981	99,367%
8		4000	19393,60	3964	99,100%
9	3	1000	17590,15	969	96,900%
10		2000	19160,68	1979	98,950%
11		3000	19575,62	2973	99,100%
12		4000	19651,25	3971	99,275%
13	4	1000	21653,23	987	98,700%
14		2000	185549,14	2000	100,000%
15		3000	21425,12	2976	99,200%
16		4000	19509,29	3974	99,350%
17	5	1000	19353,84	982	98,200%
18		2000	16782,18	1970	98,500%
19		3000	188551,11	2999	99,967%
20		4000	20594,01	3960	99,000%

5	2	1000	15388,99	973	97,300%
6		2000	17770,56	1975	98,750%
7		3000	17907,23	2970	99,000%
8		4000	19299,30	3953	98,825%
9	3	1000	19727,10	977	97,700%
10		2000	20130,09	1981	99,050%
11		3000	22695,10	2988	99,600%
12		4000	18865,52	3960	99,000%
13	4	1000	19489,30	984	98,400%
14		2000	16137,84	1976	98,800%
15		3000	18481,45	2976	99,200%
16		4000	163404,09	4000	100,000%
17	5	1000	22957,76	992	99,200%
18		2000	13884,84	1954	97,700%
19		3000	16736,78	2842	94,733%
20		4000	21091,60	3972	99,300%

Tabel 2. Hasil Percobaan dengan 15 Host

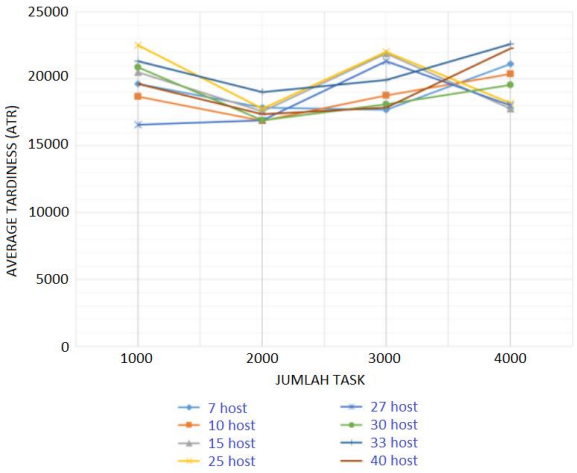
No	Uji ke-	Jumlah Task	ATR (ms)	Jumlah NDT	NDT
1	1	1000	20508,93	980	98,000%
2		2000	17556,10	1960	98,000%
3		3000	21903,55	2987	99,567%
4		4000	17785,73	3952	98,800%
5	2	1000	19723,05	990	99,000%
6		2000	18961,75	1987	99,350%
7		3000	19010,31	2975	99,167%
8		4000	17320,19	3955	98,875%
9	3	1000	22986,59	987	98,700%
10		2000	18678,41	1978	98,900%
11		3000	15693,97	2967	98,900%
12		4000	17951,24	3941	98,525%
13	4	1000	19790,07	994	99,400%
14		2000	19568,10	1979	98,950%
15		3000	22923,81	2988	99,600%
16		4000	18289,47	3941	98,525%
17	5	1000	18159,54	985	98,500%
18		2000	19194,18	1977	98,850%
19		3000	166838,36	3000	100,000%
20		4000	18764,55	3964	99,100%

Tabel 4.Hasil Percobaan dengan 40 Host

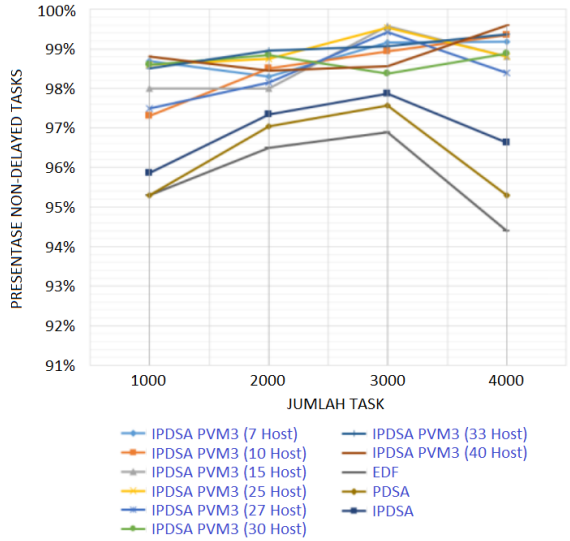
No	Uji ke-	Jumlah Task	ATR (ms)	Jumlah NDT	NDT
1	1	1000	19624,42	988	98,800%
2		2000	17363,25	1969	98,450%
3		3000	17834,10	2957	98,567%
4		4000	22266,70	3984	99,600%
5	2	1000	21462,58	988	98,800%
6		2000	20102,87	1974	98,700%
7		3000	19949,54	2967	98,900%
8		4000	19252,97	3973	99,325%
9	3	1000	22151,54	983	98,300%
10		2000	15640,64	1964	98,200%
11		3000	20839,90	2955	98,500%
12		4000	17375,90	3952	98,800%
13	4	1000	22706,54	991	99,100%
14		2000	16740,59	1968	98,400%
15		3000	16687,57	2929	97,633%
16		4000	16534,43	3929	98,225%
17	5	1000	17746,71	983	98,300%
18		2000	20064,67	1982	99,100%
19		3000	20162,91	2965	98,833%
20		4000	15514,09	3925	98,125%

Tabel 3. Hasil Percobaan dengan 33 Host

No.	Uji ke-	Jumlah Task	ATR (ms)	Jumlah NDT	NDT
1	1	1000	21310,73	985	98,500%
2		2000	19007,21	1979	98,950%
3		3000	19906,53	2972	99,067%
4		4000	22621,65	3975	99,375%



Gambar 2. Grafik *Average Tardiness* (ATR) dengan Jumlah *Task*



Gambar 3. Grafik *Non-Delayed Task* dengan Jumlah *Task*

Kriteria pertama, *task* dijalankan berdasarkan prioritas yang telah diberikan yaitu Prioritas 1, Prioritas 2 dan Prioritas 3, jumlah task yang masuk kedalam golongan prioritas tersebut didapatkan dari parameter nilai *Time Delay* (TD), antrian *task* diurutkan secara *ascending* berdasarkan nilai TD pada masing-masing *task*, pengelompokan *task* untuk setiap kelompok prioritas didapatkan dari nilai rentang tiap golongan prioritas dalam satu kali program berjalan. Beberapa hasil percobaan menunjukkan pengelompokkan prioritas *task* yang tidak wajar berupa kelompok prioritas task yang memiliki 0 task, hal ini disebabkan oleh faktor yang lain seperti koneksi jaringan yang tidak stabil sehingga terjadi *bottle neck* pada jaringan yang digunakan, dan dapat disebabkan oleh proses-proses yang lain yang berjalan di latar belakang sistem operasi.

Kriteria kedua, *Average Tardiness* (ATR) yang didapatkan pada percobaan menggunakan PVM3 ini menunjukkan nilai yang stabil, karena PVM3 dapat menggunakan sumber daya yang dapat terdistribusi secara optimal dengan jumlah komputer (*host*) yang terhubung dengan PVM3 pada setiap kelompok berbeda-beda [14]. Pada setiap percobaan yang dilakukan pada penelitian ini memiliki jumlah *host* yang beragam, yaitu 7 host, 10 host, 15 host, 25 host, 27 host, 30 host, 33 host dan 40 host, meskipun jumlah dan spesifikasi sumber daya yang heterogen dan dalam kondisi yang berbeda sebagaimana Gambar 2 dan Gambar 3. Oleh karena itu, PVM3 memungkinan kumpulan komputer (*host*) heterogen berfungsi sebagai mesin paralel kinerja tinggi tunggal (virtual), melalui mesin virtualnya ini memiliki *daemon* yang berjalan di semua komputer yang membentuk mesin virtual. PVM3 memiliki sifat portabilitas, heterogenitas dan juga memungkinkan untuk mengatasi kesalahan komunikasi antar komputer.

Kriteria ketiga, jumlah task yang tidak tertunda atau *Non-Delayed Task* (NDT) memiliki nilai diatas 97,3 % dari total *task* yang dijalankan. Persentase NDT yang didapatkan pada percobaan menggunakan PVM3 lebih tinggi dibandingkan dengan penelitian sebelumnya yaitu pada garis EDF, PDSA dan IPDSA, karena *task* dijalankan berdasarkan prioritas yang telah diberikan, nilai *Time Delay* yang didapatkan dan penggunaan PVM3 memungkinkan integrasi multi-prosesor antara komputer-komputer dalam satu jaringan yang saling terhubung [9].

4. Kesimpulan

Algoritma *resource scheduling* IPDSA telah berhasil diimplementasikan pada *grid environment*, yaitu mengunggnakan beberapa komputer sebagai *Grid Resource*. Percobaan yang dilakukan dengan memvariasikan jumlah task sebanyak 1000, 2000, 3000 dan 4000. Algoritma *scheduling* IPDSA diimplementasikan menggunakan PVM3, yang mampu memberikan performa *Grid Resource* yang optimal ketika digunakan untuk menyelesaikan *task* yang ditugaskan. Untuk menugaskan *task* mana yang dieksekusi, algoritma *scheduling* IPDSA memilih *task* yang memiliki nilai *time delay* minimum lalu *task* dikirimkan ke *host* yang dituju. Algoritma *scheduling* IPDSA juga berhasil dioptimalkan dengan memberikan nilai prioritas pada setiap *task* yang ditugaskan untuk satu kali proses penjalanan program. Hal ini ditunjukkan dengan nilai rata-rata keterlambatan (*Average Tardiness/ATR*) yang stabil dalam mengerjakan *task*, dan persentase *Non-Delayed Task* yang tinggi dibanding dengan penelitian sebelumnya yaitu memiliki nilai diatas 97,3 % dari total *task* yang dijalankan. Indikator performa *Average Tardiness* (*ATR*) dan *Non-Delayed Task* (*NDT*) digunakan untuk tujuan perbandingan.

Daftar Rujukan

[1] Foster, Kesselman, C. and Tuecke, S., 2001, The anatomy of the grid: Enabling scalable virtual organizations, Int. J. High Perform. Comput. Appl., vol. 15, no. 3, pp. 200–222.
[2] Goswami, S. and De Sarkar, A., 2013, A comparative study of load balancing algorithms in computational grid environment, in

2013 Fifth International Conference on Computational Intelligence, Modelling and Simulation, pp. 99–104.
[3] Matani, A., Naji, H.R. and Motallebi, H., 2020, A Fault-Tolerant Workflow Scheduling Algorithm for Grid with Near-Optimal Redundancy, Journal of Grid Computing, volume 18, pp. 377–394
[4] Patel, D. K., Tripathy, D. and Tripathy, C. R., 2016, Survey of load balancing techniques for grid, J. Netw. Comput. Appl., vol. 65, pp. 103–119
[5] Balasangameshwara, J. and Raju, N., 2012, Performance-driven load balancing with a primary-backup approach for computational grids with low communication cost and replication cost, IEEE Trans. Comput., vol. 62, no. 5, pp. 990–1003
[6] Midya, S., Roy, A., Majumder, K. and Phadikar, S., 2018, Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach, Journal of Network and Computer Applications, Volume 103, 1 February 2018, pp. 58-84, https://doi.org/10.1016/j.jnca.2017.11.016
[7] Topcuoglu, H., Hariri, S., and Wu, M., 2002, Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions Parallel Distribution Systems 13(3), pp. 260–274
[8] Kumar, M., Sharma, S. C., Goel, A. and Singh, S. P., 2019, A comprehensive survey for scheduling techniques in cloud computing, J. Netw. Comput. Appl..
[9] Hao, Y., Liu, G. and Wen, N., 2012, An enhanced load balancing mechanism based on deadline control on GridSim, Future Gener. Comput. Syst., vol. 28, no. 4, pp. 657–665
[10] Xu, Z., Hou, X. and Sun, J., 2003, Ant algorithm-based task scheduling in grid computing, in CCECE 2003-Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology (Cat. No. 03CH37436), vol. 2, pp. 1107–1110.
[11] Chauhan, A., Singh, S., Negi, S. and Verma, S. K., 2016, Algorithm for deadline based task scheduling in heterogeneous grid environment, in 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), pp. 219–222.
[12] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V. S., 1994, PVM: Parallel Virtual Machine :a Users' Guide and Tutorial for Networked Parallel Computing. MIT Press.
[13] Nanthiya, D. and Keerthika, P., 2013, Load balancing GridSim architecture with fault tolerance, in 2013 International Conference on Information Communication and Embedded Systems (ICICES), pp. 425–428
[14] Sampath S., Nanjesh B. R., Sagar, B. B. and Subbaraya, C. K., 2014, Performance optimization of PVM based parallel applications using optimal number of slaves, in 2014 International Conference on Reliability Optimization and Information Technology (ICROIT), Feb. 2014, pp. 388–392, doi: 10.1109/ICROIT.2014.6798360.