Advance Planning and Reservation for Jobs in a Cluster

Peneliti:
Dr. Ir. Ardi Pujiyanta, M.T.

ABSTRAK

Dalam lingkungan cluster, *resource broker* menjadi salah satu komponen yang paling penting di cluster middleware, karena memiliki tanggung jawab memilih sumber daya dan melakukan pekerjaan penjadwalan, sedemikian rupa sehingga persyaratan yang diberikan oleh user/aplikasi terpenuhi, dalam hal waktu keseluruhan pelaksanaan pekerjaan. Cluster adalah semacam generalisasi dari *World Wide Web*. Internet menyediakan akses ke informasi yang tersimpan dalam sejumlah besar komputer yang tersebar di berbagai lokasi. Ketika halaman Web dibaca, tak ada yang peduli di mana informasi tersebut berasal, didalam sesi Web yang khas adalah mungkin membaca halaman yang berada di komputer yang berbeda dan ditempatkan di negara-negara yang berbeda. Teknologi cluster membawa ide *World Wide Web* ke panggung lain, yang memungkinkan akses penggunaan pengolahan sumber daya dan data.

Didalam penelitian ini diusulkan keterbaruan strategi penjadwalan pekerjaan yang akan datang menggunakan strategi FCFS-LRH untuk meningkatkan ketermanfaatan cluster system. Strategi ini memetakan pekerjaan pada virtual node (logical view) dan memberkan garansi bahwa pekerjaan akan dikerjakan pada resource computer yang sebenarnya(physical view) untuk dieksekusi. Kami telah melakukan penelitian dan membandingkannya dengan FCFS-EDS hasilnya didalam model yang kami usulkan permutasi *Backward*(mundur) dapat mereduksi proses transpose yang ada di FCFS-EDF. Serta dari ketermanfaatan lebih baik dari model penjadwalan konvensional FCFS.

Kata Kunci: Cluster, penjadwalan, Metode FCFS-LRH

1.1 Latar Belakang

Dalam beberapa tahun terakhir, meningkatnya minat dalam penggunaan sumber daya berbasis jaringan untuk tujuan komputasi skala besar, paradigma komputasi Grid berasal dari infrastruktur komputasi baru untuk penelitian dan kerja sama ilmiah, dan menjadi sebuah teknologi yang dibangun untuk berbagi sumber daya dalam skala besar yang terdistribusi dan terintegrasikan. Pembangunan jaringan ditujukan untuk kegunaan yang beraneka ragam dengan pengelolaan yang efisien, didistribusikan secara geografis, serta ketersediaan sumber daya komputasi yang dinamis.

Dalam lingkungan grid, *resource broker* menjadi salah satu komponen yang paling penting di Grid middleware, karena memiliki tanggung jawab memilih sumber daya dan melakukan pekerjaan penjadwalan, sedemikian rupa sehingga persyaratan yang diberikan oleh user/aplikasi terpenuhi, dalam hal waktu keseluruhan pelaksanaan pekerjaan. Grid adalah semacam generalisasi dari *World Wide Web*. Internet menyediakan akses ke informasi yang tersimpan dalam sejumlah besar komputer yang tersebar di berbagai lokasi. Ketika halaman Web dibaca, tak ada yang peduli di mana informasi tersebut berasal, didalam sesi Web yang khas adalah mungkin membaca halaman yang berada di komputer yang berbeda dan ditempatkan di negara-negara yang berbeda. Teknologi Grid membawa ide *World Wide Web* ke panggung lain, yang memungkinkan akses penggunaan pengolahan sumber daya dan data.

Grid computing(komputasi grid) adalah penerapan sumber daya dalam jaringan untuk menyelesaikan satu masalah pada saat yang sama, biasanya digunakan untuk penyelesaian masalah ilmiah atau teknis, seperti fisika energi tinggi, observasi bumi, dan aplikasi biologi, yang membutuhkan sejumlah besar siklus pengolahan komputer atau akses ke data dalam jumlah besar.

Komputasi grid dapat dianggap sebagai komputasi cluster yang terdistribusi dengan skala besar dan sebagai bentuk pemrosesan jaringan paralel yang terdistribusikan[1].

Grid berisi sumber daya yang berbeda sifatnya seperti, misalnya, CPU, jaringan, data, atau perangkat lunak[2,3], dengan cepat menjadi tujuan penelitian utama untuk menawarkan pengguna akan akses transparan ke sumber daya. Transparansi adalah alasan untuk menggunakan istilah "Grid" yang mengacu kepada Jaringan Listrik yang hanya menyediakan pada permintaan daya listrik kepada semua pengguna, tanpa memerlukan wawasan lebih tentang bagaimana dan di mana daya yang sebenarnya telah dihasilkan. Demikian pula, komputasi grid memberikan daya komputasi sesuai permintaan, untuk semua pengguna tanpa pengetahuan tentang lokasi dari sumber daya yang dialokasikan. Secara umum, grid biasanya dinotasikan sebagai berbagi sumber daya yang didistribusikan secara geografis dan dimiliki oleh penyedia layanan yang berbeda dan berada di domain administrasi yang berbeda.

Mekanisme khas Grid adalah sebagai berikut[2]: user mengirimkan pekerjaan dengan melalui *Graphic User Interface*, dengan memberikan beberapa spesifikasi tingkat tinggi(Misalnya, jenis aplikasi yang akan digunakan). Grid memainkan peran mencari dan mengalokasikan *sumber daya*(sumber daya) yang layak(komputer, penyimpanan) untuk memenuhi permintaan user. Kemudian grid, memonitor pengolahan pekerjaan yang benar, dan memberitahu user bahwa sumber daya yang diperlukan tersedia. Perlu diperhatikan, dari analogi jaringan energi listrik bahwa pengguna tidak tahu sumber daya apa dan di mana yang terlibat dalam permintaannya. User hanya memperoleh daya komputasi dan ruang penyimpanan yang dibutuhkan melalui antarmuka standar.

Dua masalah yang paling penting dalam mengelola pekerjaan dari user yaitu pengalokasian sumber daya dan penjadwalan pekerjaan pada sumber daya yang dibutuhkan. Ketika pekerjaan dari pengguna disampaikan, pekerjaan tersebut akan dikelola oleh broker sumber daya, yang harus menemukan dan mengalokasikan sumber daya yang tepat untuk pekerjaan tersebut. Setelah tahap alokasi sumber daya, pekerjaan harus dijadwalkan pada sumber daya yang ada dan sesuai kebutuhan user akan sumber daya yang diperlukan.

Pengelolaan sumber daya Grid memiliki beberapa lapisan yang berbeda dari penjadwal. Pada tingkat tertinggi adalah pengelolaan sumber daya global yang mungkin memiliki pandangan yang lebih umum dari sumber daya, tetapi sangat jauh dari sumber, yang mana akhirnya aplikasi akan dieksekusi. Pada tingkat terendah adalah sistem pengelolaan sumber daya lokal, yang mengelola sumber daya tertentu atau mengatur sumber daya[5]. Dalam pengelolaan sumber daya lokal, sumber daya yang diakses, ditugaskan, dan dialokasikan sesuai dengan kriteria *Quality of Service* (QoS), seperti pemesanan, dan batas waktu.

Pada kebanyakan sistem grid dengan penjadwalan tradisional, pekerjaan diserahkan dan ditempatkan dalam antrian, menunggu jika sumber daya yang dimandatkan tersedia bagi mereka. Setiap sistem grid dapat menggunakan algoritma penjadwalan yang berbeda, misalnya *First Come First Serve*(FCFS), *Shortest Job First* (SJF), *Earliest Deadline First* (EDF)[11], dengan mengeksekusi pekerjaan berdasarkan parameter yang berbeda, seperti jumlah sumber daya, waktu pengiriman, dan durasi eksekusi. Dengan algoritma penjadwalan tersebut, tidak ada jaminan tentang kapan pekerjaan akan dilaksanakan[6].

Didalam penjadwalan *rigid*/kaku, bila pengguna meminta *sumber daya* untuk menjalankan pekerjaan mereka, mereka harus menyediakan tiga parameter, waktu mulai, waktu pelaksanaan dan jumlah *sumber daya*[7]. Sistem reservasi kemudian mencari ketersediaan *sumber daya* yang diminta dalam interval waktu yang ditentukan. Jika sumber daya yang dibutuhkan tidak tersedia, permintaan tersebut ditolak. Mekanisme ini dikenal sebagai reservasi kaku (GARA)[8][9].

Sebagai konsekuensi dari mekanisme ini(reservasi kaku), jika permintaan dari pengguna ditolak karena tidak tersedianya sumber daya pada waktu tertentu seperti yang diminta oleh pengguna, ia dapat mengajukan kembali permintaan baru dengan parameter dimodifikasi, seperti waktu mulai dan atau waktu pelaksanaan yang berbeda sampai sumber daya yang tersedia dapat

ditemukan. Jika ini sering terjadi, *scheduler* bekerja keras menangani permintaan pengguna yang sama karena permintaan sebelumnya ditolak. Pada akhirnya akan menyebabkan sumber daya menganggur antara pekerjaan, maka pemanfaatan sumber daya akan turun.

Sulistio et al. [10] telah mengusulkan reservasi elastis dengan mengambil parameter permintaan pengguna sebagai kendala lembut. Sistem reservasi bukannya menolak permintaan tersebut tetapi memberikan alternatif yang dapat dipilih oleh pengguna. Pendekatan ini memberikan fleksibilitas kepada pengguna untuk memilih pilihan terbaik atau dalam pemesanan pekerjaan mereka sesuai dengan kebutuhan *quality of Service* (QoS), seperti batas waktu. Setelah pengguna memilih salah satu pilihan alternatif yang diberikan, maka pengguna mengirimkan lagi permintaannya. Tapi, kali ini pengguna mengirimkan permintaan seperti cara di reservasi kaku, karena ada jaminan ketersediaan sumber daya untuk dirinya.

Smith et.al [7] telah memperkenalkan dampak pemesanan diawal dalam sistem penjadwalan. Mu'alem et.al [11] didalam penelitiannya menyimpulkan bahwa dengan menambahkan pemesanan diawal, akan meningkatkan rata-rata waktu tunggu permintaan *scheduler* dalam antrian sebesar 9% dengan adanya penimbunan.

Untuk mengatasi masalah penurunan pemanfaatan sumber daya yang disebabkan oleh pemesanan diawal[12] menggunakan slot waktu tumpang tindih, karena mereka percaya bahwa aplikasi cenderung melebih-lebihkan batas waktu reservasi untuk memastikan penyelesaian mereka. Strategi yang digunakan[13][14], pekerjaan pengguna dijadwalkan, bahkan jika ada pelanggaran pemesanan dalam tenggat waktu karena tumpang tindih pekerjaan. Mereka beranggapan bahwa beban kerja riil cenderung melebih-lebihkan batas waktu relatif dengan rata-rata nilai sebesar 35%.

Dalam pemesanan diawal yang *fleksibel*, pekerjaan pengguna dijadwalkan dan diberikan batasan yang fleksibel, waktu mulai tidak tetap dan dapat bervariasi dalam interval waktu. Dampak dari *backfilling algorithm* dalam pemesanan diawal yang *fleksibel* telah dilakukan penelitian oleh[15]. *Backfilling* diusulkan untuk meningkatkan pemanfaatan sistem. Strategi ini mengambil keuntungan dari pergeseran pemesanan yang dibuat lebih awal (tergantung pengingat kendala fleksibel) untuk membuat ruang agar reservasi baru masuk.

Chunming et.al [16] didalam penelitiannya memperkenalkan rentang waktu, ketika ada rentang waktu/range untuk waktu mulai dari pekerjaan. Rentang waktu ini disebut *slack-time*. Mereka mengusulkan mekanisme baru yang disebut FIRST(*Flexible Reservation using Slack Time*). Keuntungan dari *slack-time* adalah bahwa waktu mulai dari pekerjaan dapat digeser untuk meningkatkan pemanfaatan sumber daya dan untuk mengurangi tingkat penolakan. Jika reservasi baru datang, sistem reservasi menjadwal ulang semua reservasi yang tidak dieksekusi satu per satu, sesuai dengan aturan FIFO (*First In First Out*), untuk melihat apakah ada solusi atau tidak. Jika tidak ada solusi maka permintaan reservasi baru akan ditolak.

Perencanaan di awal dan strategi reservasi *First Come First Serve Ejecting Based Dynamic Scheduling* (FCFS-EDS) untuk meningkatkan pemanfaatan sumber daya dalam sistem grid pada *local scheduler*. Untuk mencapai hal ini pekerjaan pengguna akan dipetakan ke node komputasi virtual yang disebut pandangan logis(*logical view*), yang kemudian dipetakan ke node komputasi yang sebenarnya yang disebut pandangan fisik(*physical view*) pada saat eksekusi[17]. Strategi ini mengambil keuntungan dari pergeseran waktu mulai pekerjaan sebelumnya yang dibuat, dengan cara membuat ruang kosong untuk permintaan pekerjaan baru yang akan masuk. Didalam algoritma yang digunakan ada prosedure perhitungan matrik transpose, yang berfungsi untuk memetakan dari pandangan logis(*logical view*), ke node komputasi pandangan fisik(*physical view*) dengan adanya prosedure matrik transpose yang digunakan menyebabkan kinerja algoritmanya menjadi lebih lambat, karena *scheduler* terbebani harus melakukan proses perhitungan prosedure matrik transpose. Sehingga dengan adanya kelemahan ini masih dimungkinkan diusulkannya model algoritma lain, yang dapat mereduksi prosedure matrik transposenya.

Grandinetti et al. [36] telah melakukan penelitian pada penjadwalan *local scheduler,* di mana sekelompok pekerjaan *independent* telah dijadwalkan, dengan pembatasan waktu pemrosesan (Yaitu, batas waktu, waktu mulai paling awal, jumlah node) yang diberikan oleh pengguna. Diasumsikan bahwa semua node pengolahan adalah identik. beban kerja terdiri dari pekerjaan batch yang membutuhkan eksekusi ruang-sharing. Dengan demikian, setiap pekerjaan yang antri dapat dimulai hanya jika ada node yang cocok dengan kapasitas yang dibutuhkan. Masalah diselidiki dan dirumuskan menggunakan metode *Rectangle packing* yang mengacu pada penelitian[20], tetapi penelitian ini belum membahas *advance reservation.*

Struktur data digunakan untuk menyimpan ringkasan informasi permintaan reservasi dan menjadi dasar untuk kontrol masukan langsung dalam proses pemesanan sumber daya. Oleh karena itu, struktur data harus dapat memberikan akses cepat ke informasi dan menanganinya secara efisien. Sekitar 60 persen dari total waktu proses dibutuhkan untuk pengelolaan struktur data, 8 persen digunakan untuk pemilihan sumber daya yang tepat, dan 32 persen sisanya untuk pengelolaan sumber daya di dalam reservasi sumber daya[21]. Jika permintaan aplikasi disediakan untuk seluruh potensi layanan pemesanan diawal, maka lebih banyak waktu yang dibutuhkan. Misalnya, selama interval scanning dan deteksi sumber daya, waktu pemrosesan struktur data mencapai hingga 90% dari total waktu[52].

Dalam mengelola penjadwalan pekerjaan terlebih dahulu dalam sistem Grid, struktur data harus dapat menangani pekerjaan serial(*parametric job*) dan pekerjaan parallel(*MPI job*) yang memainkan peran penting dalam rangka untuk meminimalkan waktu untuk mencari sumber daya komputasi yang tersedia, menambah dan menghapus pekerjaan, sehingga perlu diusulkannya model struktur data yang dapat menangani pekerjaan serial(*parametric job*) dan pekerjaan parallel(*MPI job*) yang dapat mendukung model penjadwalan *advance reservation*. Dengan adanya struktur data yang dapat mendukung model penjawalan *advance reservation*, maka penjadwalan pekerjaan diawal akan mendapatkan waktu respon yang cepat, dalam rangka untuk memberikan hasil apakah permintaan pekerjaannya diterima atau tidak. sehingga dapat memaksimalkan efisiensi dalam hal penggunaan sumber daya.

Dari latar belakang masalah tersebut diatas maka diusulkannya model penjadwalan *advance reservation*, pada lingkungan *Local Scheduler*(LS) dan pekerjaan bersifat *independent job*, untuk mengatasi dan memastikan ketersediaan sumber daya tertentu pada waktu tertentu di masa depan, sehingga memberikan jaminan bahwa pekerjaan pengguna akan dilaksanakan. Penjadwalan pekerjaan diawal memungkinkan pengguna untuk meminta sumber daya dari beberapa sumber daya pada waktu tertentu di masa depan dan dengan demikian mendapatkan akses simultan ke sumber daya yang cukup untuk aplikasi mereka.

1.2. Perumusan Masalah

Didalam *advance reservation* adanya jaminan/garansi pekerjaan user bahwa pekerjaannya akan terlayani, merupakan hal yang sangat penting, bila ini tidak tertangani dengan baik akan menyebabkan penurunan ketermanfaatan sumber daya komputer. Berdasarkan alasan tersebut dan paparan permasalahan pada latar belakang, serta mengacu pada penelitian[17]. Didalam strategi algoritma yang digunakan ada prosedure perhitungan matrik transpose, yang berfungsi untuk memetakan dari pandangan logis(*logical view*), ke node komputasi pandangan fisik(*physical view*) dengan adanya prosedure perhitungan matrik transpose yang digunakan menyebabkan kinerja algoritmanya menjadi lebih lambat. Sehingga dengan adanya kelemahan ini masih dimungkinkan diusulkannya model algoritma lain, yang dapat mereduksi model matrik transposenya, oleh karena itu akan diusulkan model penjadwalan *advance reservation* dan model struktur data yang mampu memberikan jaminan pekerjaan akan dikerjakan pada waktu yang akan datang, sehingga dapat meningkatkan kinerja dan ketermanfaatan sumber daya, untuk waktu yang akan datang.

1.3. Batasan Masalah

Dalam melakukan penelitian ini, peneliti membatasi masalah atau ruang lingkup penelitian. Hal ini dimaksudkan agar penelitian dapat dilakukan pada batasan yang jelas. Adapun

batasan dalam penelitian ini difokuskan pada masalah penjadwalan lokal yang ditangani oleh masing-masing *Local Scheduler* di mana pekerjaan bersifat *independent job* yang dijadwalkan pada situs/domain tertentu, Dengan demikian, pekerjaan yang antri dapat dimulai hanya jika ada node pemrosesan yang cocok dengan kapasitas yang cukup dan sistem berjalan dalam kondisi normal dalam lingkungan grid.

1.4. Keaslian Penelitian

Penelitian yang berhubungan dengan sistem penjadwalan *advance reservation* [7,15][12,13,14,16], telah banyak dilakukan oleh para peneliti sebelumnya. Keaslian penelitian ini adalah diusulkannya model penjadwalan *advance reservation*, serta struktur data yang sesuai dengan model penjadwalan yang diusulkan, sehingga dapat memberikan jaminan pekerjaan akan dikerjakan pada masa yang akan datang, dan memaksimalkan ketermanfaatan sumber daya. Model penjadwalan yang diusulkan menggunakan model matrik permutasi maju dan mundur, serta unjuk kerjanya akan dibandingkan dengan penelitian sebelumnya, dengan cara membangun model simulasi di gridsim.

1.5. Tujuan Penelitian

1. Mengembangkan model penjadwalan baru yang optimal pada sumber daya komputer yang mampu mengaransi user bahwa pekerjaannya akan terlayani.
2. Mengembangkan model algoritma penjadwalan untuk optimasi penjadwalan pelayanan reservasi pada komputasi grid.
3. Didapatkan model struktur data waktu penjadwalan yang mampu mengaransi user bahwa pekerjaannya akan terlayani.
4. Didapatkan model *type job* yang mampu menangani struktur data waktu penjadwalan.

1.6. Manfaat Penelitian

1. Pengguna komputer mendapatkan waktu tanggap reservasi yang lebih cepat.
2. Pengguna komputer/pelanggan mendapatkan garansi/jaminan bahwa pekerjaannya akan dilaksanakan pada resource komputer.
3. Pemilik grid resource komputer dapat memanfaatkan resource komputer secara maksimal, karena adanya garansi pekerjaan pelanggan akan terlayani.
4. Memudahkan pemilik grid menjadwal pekerjaan untuk waktu yang akan datang.

2.1 Tinjauan Pustaka

2.1.1. Komputasi Grid

Popularitas Internet serta ketersediaan komputer yang powerful dengan Jaringan berkecepatan tinggi sebagai komponen komoditas mengubah cara kita menggunakan komputer saat ini. Peluang teknis ini telah menyebabkan kemungkinan penggunaan Sumber daya terdistribusi secara geografis untuk memecahkan masalah berskala besar di dalam Ilmu pengetahuan, teknik, dan perdagangan. Penelitian terbaru tentang topik ini telah mengarah pada munculnya sebuah paradigma baru yang dikenal sebagai komputasi Grid[35].

Komputasi grid adalah infrastruktur perangkat keras dan perangkat lunak yang disediakan dan dapat diandalkan, konsisten, jangkauwannya luas, dengan kemampuan komputasi yang tinggi[2], ini adalah penerapan lingkungan bersama melalui penyebaran sebuah Infrastruktur layanan berbasis standar yang secara terus-menerus mendukung penciptaan berbagi sumber daya, dengan komunitas terdistribusi. Sumber daya dapat berupa komputer, ruang penyimpanan, instrumen, aplikasi perangkat lunak dan data, semuanya terhubung melalui Internet dan lapisan perangkat lunak middleware yang menyediakan layanan dasar untuk keamanan, pemantauan, pengelolaan sumber daya, dan sebagainya. Kepemilikan Sumber daya terdiri dari berbagai organisasi administratif yang berbeda, terbagi berdasarkan kebijakan yang ditetapkan secara lokal untuk menentukan apa yang akan dibagikan, siapa yang diizinkan untuk mengakses, dan dalam kondisi bagaimana[4]. Masalah nyata dan spesifik yang mendasari konsep Grid adalah pembagian sumber daya yang terkoordinasi didalam pemecahan masalah pada organisasi virtual multi-institusional secara dinamis[31].

Kemajuan teknologi jaringan telah mendorong dan memberikan kesempatan pengunaan komputer yang terhubung dengan jaringan sebagai satu sistem komputasi terpadu tunggal, yang dikenal sebagai komputasi cluster. Cluster dapat digunakan dalam berbagai bentuk dengan keperluan yang berbeda, seperti komputasi kinerja tinggi untuk daya komputasi yang lebih dibandingkan komputer tunggal, ketersediaan keandalan yang tinggi dan lebih besar, dan *throughput* yang tinggi dengan kemampuan pemrosesan yang lebih lama dan lebih besar. Sebuah *cluster* memiliki beberapa karakteristik antara lain :

- Terdiri dari beberapa mesin-mesin bertipe sama.
- *Tightly-coupled* menggunakan koneksi jaringan yang *dedicated*
- Semua mesin berbagi sumber daya.
- Harus mempunyai *software* seperti implementasi MPI yang memungkinkan program-program dijalankan di semua *node*.

Cluster berbeda dengan Cloud dan Grid karena cluster adalah sekelompok komputer yang terhubung dengan jaringan area lokal (LAN), sedangkan cloud dan grid lebih luas dan dapat didistribusikan secara geografis. Cara lain untuk mengatakannya adalah dengan mengatakan bahwa sebuah cluster digabungkan secara *Tightly-coupled*, sedangkan Grid atau cloud digabungkan secara *Loosely-coupled*. Cluster terdiri dari kelompok mesin dengan perangkat keras yang serupa, sedangkan cloud dan grid terdiri dari mesin dengan konfigurasi perangkat keras yang mungkin sangat berbeda.

Perbedaan antara cloud dan grid dapat dinyatakan sebagai berikut: Distribusi sumber daya: Komputasi cloud adalah model terpusat sedangkan komputasi grid adalah model terdesentralisasi dimana perhitungan dapat terjadi di banyak domain administratif. Kepemilikan: Grid adalah kumpulan komputer yang dimiliki oleh banyak pihak di beberapa lokasi dan terhubung bersama sehingga pengguna dapat berbagi kekuatan gabungan sumber daya. Sedangkan cloud adalah kumpulan komputer yang biasanya dimiliki oleh satu pihak.

Grids merupakan pencapaian yang signifikan terhadap agregasi kelompok dan/atau Sumber daya jaringan lainnya untuk memecahkan aplikasi intensif, data intensif atau komputasi intensif skala besar[2]. Bergantung pada target domain dan tujuan aplikasi, Grids dapat dikelompokkan menjadi beberapa kategori [49] :

1. Komputasi Grids: menyediakan fasilitas komputasi terdistribusi untuk mengeksekusi Aplikasi komputasi intensif, seperti simulasi Monte Carlo[51], dan aplikasi Bag-ofTasks (BoT)[50].
2. Data Grids: menyediakan infrastruktur untuk akses, transfer dan pengelolaan dataset besar yang tersimpan dalam repositori terdistribusi[53,54]. Selain itu, data Grids fokus untuk memenuhi persyaratan kolaborasi ilmiah, di mana ada kebutuhan untuk menganalisis kumpulan data yang besar dan membagikan hasilnya.
3. *Application Service Provisioning*(ASP) Grid: Berkonsentrasi untuk penyediaan akses ke aplikasi jarak jauh, modul, dan perpustakaan di pusat data, mis. NetSolve [55].
4. Grid interaksi: Menyediakan layanan dan platform bagi pengguna untuk berinteraksi Satu sama lain dalam lingkungan real-time, mis. AccessGrid[56]. Dengan demikian, jenis Grids ini cocok untuk aplikasi multimedia, seperti konferensi video, dan ini membutuhkan jaringan cepat.
5. Grid Pengetahuan: Bekerja pada akuisisi pengetahuan, pengolahan data, dan pengelolaan data. Selain itu, mereka menyediakan layanan analisis bisnis yang didorong oleh layanan data mining terpadu. Proyek di bidang ini adalah *KnowledgeGrid*[57].
6. Utilitas Grids: Fokus untuk menyediakan satu atau lebih layanan Grid kepada pengguna akhir sebagai utilitas teknologi informasi(TI) dengan melakukan pembayaran untuk mengaksesnya. Selain itu, mereka membuat kerangka kerja untuk negosiasi dan pembentukan kontrak, dan alokasi sumber daya berdasarkan permintaan pengguna. Proyek yang ada di daerah ini adalah Utility Data Center [58] di tingkat perusahaan dan Gridbus[59] di tingkat global.

Berdasarkan skenario penggunaan diatas, dari sudut pandang pengguna, komputasi grid bisa dianggap sebagai menciptakan komputer virtual yang menggabungkan infrastruktur perangkat keras dan penyimpanan besar yang dikelola oleh berbagai organisasi di seluruh dunia[2]. Skenario ini juga mengidentifikasi beberapa fungsi atau komponen utama yang perlu ditangani oleh penyedia sumber daya Grid.

- User interface, dimana pengguna bisa mengirimkan dan melacak pekerjaan dengan menggunakan Antarmuka *command-line* atau login jauh, antarmuka pengguna grafis (misalnya, portal berbasis web, seperti Portal *P-GRADE*[60] dan Portal *BioGrid*[61]
- Keamanan dan manajemen akses, di mana pengguna perlu diautentikasi dan diberi wewenang sebelum mengirimkan pekerjaan dan menggunakan sumber daya masing-masing.
- Administrasi dan pemantauan, di mana administrator sumber daya dapat mengendalikan dan memantau keadaan sumber daya saat ini, dan pengguna dapat melacak atau melihat perkembangan Pekerjaan melalui sebuah antarmuka
- Penemuan sumber daya, di mana ketersediaan dan status sumber daya terdaftarkan di pusat Server, dengan demikian pengguna bisa bertanya tentang Sumber daya ini.
- Pengelolaan data, di mana sumber daya mengelola permintaan, replikasi dan penghapusan data Set. Selain itu, diterapkannya berbagai teknik replikasi.
- Pengelolaan sumber daya, di mana sumber daya dialokasikan, ditugaskan dan diakses sesuai dengan kriteria *Quality of Service*(QoS), seperti reservasi awal, batas waktu dan biaya.
- Penjadwalan pekerjaan, di mana penjadwalan sumber daya lokal, pekerjaan menunggu dieksekusi dalam antrian berdasarkan kriteria QoS, seperti yang disebutkan di atas.

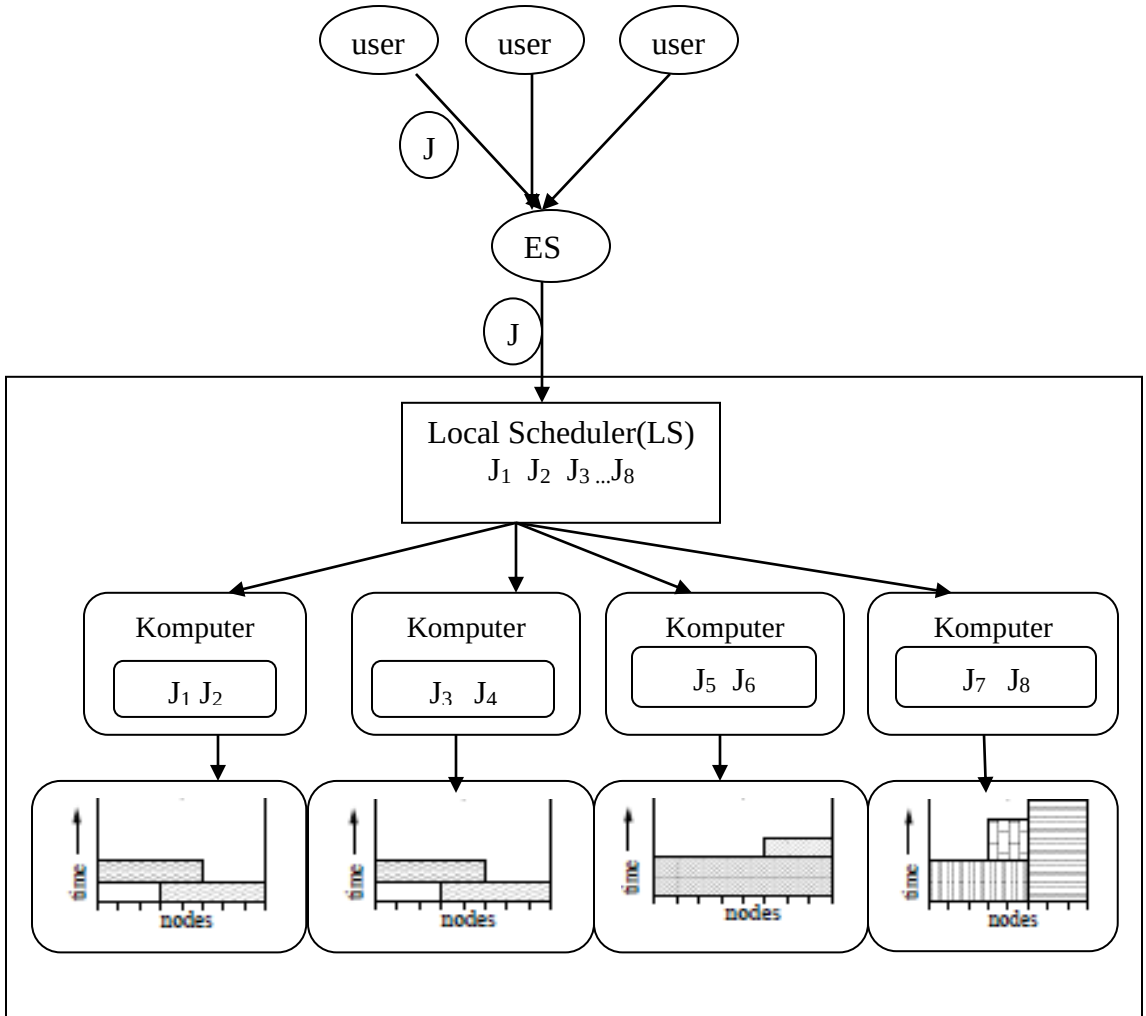
Pengelolaan sumber daya dan Penjadwalan pekerjaan merupakan komponen utama dari sistem grid. Tanggung jawab dasarnya adalah menerima pekerjaan dari pengguna, dan menjadwalkan pekerjaan ke sumber grid yang sesuai[4]. Namun, kinerja grid dapat ditingkatkan dalam hal waktu proses kerja dengan memastikan semua sumber daya yang ada di grid dimanfaatkan secara optimal dengan menggunakan algoritma penjadwalan pekerjaan yang baik. Dalam penjadwalan sistem komputasi tradisional masalah dipelajari dengan baik, ada banyak penjadwal pekerjaan didalam lingkungan komputasi. Sistem penjadwal ini dirancang untuk bekerja dengan asumsi bahwa mereka memiliki kontrol penuh terhadap sumber daya dan dengan demikian dapat menerapkan mekanisme dan kebijakan yang diperlukan untuk penggunaan sumber daya secara efektif.

Untuk mencapai potensi yang menjanjikan dari sumber daya terdistribusi yang luar biasa, algoritma penjadwalan yang efisien sangat penting. Sayangnya, algoritma penjadwalan tradisional dalam sistem paralel dan terdistribusi, tidak bisa bekerja dengan baik dalam situasi baru[12]. Penelitian ini terutama difokuskan pada penjadwalan pekerjaan yang akan datang dari komputasi Grid. Dengan tujuan untuk meningkatkan pemanfaatan sumber daya dan kepuasan pengguna dengan mempertimbangkan penjadwalan pekerjaan baru dan strategi penjadwalan pekerjaan untuk masa yang akan datang.

Solusi untuk masalah administrasi adalah menggunakan skema hirarkis. dimana dalam skema hirarkis, kebijakan lokal dan global dapat diterapkan pada satu tempat. Pekerjaan diserahkan ke eksternal scheduler(ES), yang kemudian mengirimkan pekerjaan ke penjadwal local yang ada di sebuah situs. Penjadwal lokal mengalokasikan pekerjaan tersebut ke mesin yang tergantung pada kebijakan lokalnya.

Di dalam penelitian ini, sumber daya komputasi di situs didedikasikan hanya untuk pengguna lokal (Gambar 2.1). Oleh karena itu, setiap situs memiliki beban kerja sendiri yang tidak dibagi dengan situs lain. Untuk simulasi dalam penelitian satu eksternal scheduler untuk satu situs yang akan disimulasikan menggunakan gridsim, Dalam penelitian ini diusulkan model penjadawalan FCFS-LRH.

External Scheduler (ES): Pengguna mengirimkan pekerjaan ke eksternal scheduler. kemudian ES memutuskan lokasi situs mana yang akan dikirimi pekerjaan tergantung pada beberapa algoritma penjadwalan yang digunakan. Mungkin perlu eksternal Informasi seperti beban di lokasi tertentu atau lokasi dataset untuk mengambil keputusan.
Local Scheduler (LS): Sekali pekerjaan ditugaskan untuk dikerjakan pada situs tertentu (Dan dikirim ke antrian pekerjaan yang masuk), kemudian dikelola oleh *Local Scheduler(* LS). *Local Scheduler* dari sebuah situs menentukan bagaimana menjadwalkan semua pekerjaan yang dialokasikan pada sumber dayanya.



Gambar 2.1. Interaksi antar komponen pada grid

2.1.2. *Advance Reservation System*

Pada kebanyakan sistem grid dengan penjadwalan tradisional, pekerjaan diserahkan ditempatkan dalam antrian menunggu jika sumber daya yang dimandatkan tersedia bagi mereka. Setiap sistem grid dapat menggunakan algoritma penjadwalan yang berbeda, misalnya *First Come First Serve* (FCFS), *Shortest Job First* (SJF), Earliest *Deadline First* (EDF),[11] dengan mengeksekusi pekerjaan berdasarkan parameter yang berbeda, seperti jumlah sumber daya, waktu pengiriman, dan durasi eksekusi. Dengan algoritma penjadwalan tersebut, tidak ada jaminan tentang kapan pekerjaan ini akan dilaksanakan[6].

Untuk mengatasi masalah waktu tunggu yang tidak beralasan dan memastikan bahwa sumber daya tertentu yang tersedia untuk aplikasi pada waktu tertentu di masa depan, kita perlu perencanaan diawal dalam sistem reservasi[18]. Pemesanan diawal memungkinkan pengguna untuk meminta sumber daya dari beberapa sistem penjadwalan pada waktu tertentu di masa depan dan dengan demikian mendapatkan akses simultan ke sumber daya yang cukup untuk aplikasi mereka[7]. Banyak literatur sebelumnya membahas tentang strategi pemesanan diawal untuk meningkatkan pemanfaatan sumber daya. Strategi pemesanan diawal yang dilaporkan dalam literatur sebagai berikut:

2.1.2.1. *Rigid Reservation*

Bila pengguna meminta *sumber daya* untuk menjalankan pekerjaan mereka, mereka harus menyediakan tiga parameter, waktu mulai, waktu pelaksanaan dan jumlah *sumber daya*[7]. Sistem reservasi kemudian mencari ketersediaan node yang diminta dalam interval waktu yang ditentukan. Jika node yang dibutuhkan tidak tersedia, permintaan tersebut ditolak. Mekanisme ini dikenal sebagai reservasi kaku dan telah diadopsi ke dalam Arsitektur Globus untuk Reservasi dan Alokasi (GARA)[8][9].

Sebagai konsekuensi dari mekanisme ini (reservasi kaku), jika permintaan dari pengguna ditolak karena tidak tersedianya penghitung node pada waktu tertentu seperti yang diminta oleh pengguna, ia dapat mengajukan kembali permintaan baru dengan parameter dimodifikasi, seperti waktu mulai dan atau waktu pelaksanaan yang berbeda sampai sumber daya yang tersedia dapat ditemukan. Jika ini sering terjadi, scheduler bekerja keras menangani permintaan pengguna yang sama karena permintaan sebelumnya ditolak. Pada akhirnya, ini akan menyebabkan fragmentasi, yang menyebabkan penghitung node menganggur antara pekerjaan, maka pemanfaatan penghitung node turun.

2.1.2.2. *Elastic Advance Reservation*

Dengan pemesanan kaku, ada kelemahan. Jika pengguna masih ingin memesan penghitung node harus mengubah parameter pekerjaan lagi dan permintaan untuk reservasi, sampai ada kecocokan antara ketersediaan penghitung node dan permintaan pengguna. Mekanisme ini membawa overhead karena lalu lintas komunikasi. Pemesanan elastis diusulkan oleh[10] mengambil parameter permintaan pengguna sebagai kendala lembut. Sistem reservasi bukannya menolak permintaan tersebut tetapi memberikan alternatif yang dapat dipilih oleh pengguna. Pendekatan ini memberikan fleksibilitas kepada pengguna untuk memilih pilihan terbaik atau dalam pemesanan pekerjaan mereka sesuai dengan kebutuhan *quality of Service* (QoS), seperti batas waktu.

Sulistio et al. [10] dalam mencari alternatif node komputasi yang tersedia untuk para pengguna, telah membandingkan antara *rigid reservation*(RR), algoritma *First Fit*(FF) dan algoritma *on-line strip packing*(OSP). Hasil penelitian menunjukkan bahwa metode *on-line*

strip packing(OSP) melakukan lebih baik daripada *first fit*(FF) dan FF melakukan lebih baik daripada *rigid reservation*(RR). Strategi ini telah diadopsi di GridSim[19].

2.1.2.3. *Overlapping/Relax Advance Reservation*

Smith et.al [7] telah memperkenalkan dampak pemesanan diawal dalam sistem penjadwalan. Mereka menyimpulkan bahwa dengan menambahkan pemesanan diawal, akan meningkatkan rata-rata waktu tunggu permintaan *scheduler* dalam antrian (non-reservasi query) sebesar 9% dengan adanya penimbunan[11]. Hal ini dilakukan dengan memilih 10% dari permintaan sebagai permintaan reservasi dan 90% adalah permintaan non-reservasi, seluruh model dengan beban kerja yang berbeda. Jika permintaan reservasi meningkat menjadi 20%, maka rata-rata waktu query non-reservasi akan menunggu dan meningkat sebesar 37%. Mereka juga menemukan dengan menerapkan pemesanan diawal dalam sistem penjadwalan, akan mengurangi pemanfaatan sumber daya antara 54 dan 59%. Hal ini karena fragmentasi atau kesenjangan waktu idle yang disebabkan oleh pemesanan diawal.

Untuk mengatasi masalah penurunan pemanfaatan sumber daya yang disebabkan oleh pemesanan diawal[12] menggunakan slot waktu tumpang tindih, karena mereka percaya bahwa aplikasi cenderung melebih-lebihkan batas waktu reservasi untuk memastikan penyelesaian mereka. Dalam strategi ini[13][14], pekerjaan pengguna dijadwalkan, bahkan jika ada pelanggaran pemesanan dalam tenggat waktu karena tumpang tindih pekerjaan. Mereka beranggapan bahwa beban kerja riil cenderung melebih-lebihkan batas waktu relatif dengan rata-rata nilai sebesar 35%.

2.1.2.4. *Flexible Advance Reservation* (Static)

Dalam pemesanan diawal yang *fleksibel*, pekerjaan pengguna dijadwalkan dan diberikan batasan yang fleksibel. Waktu mulai tidak tetap dan dapat bervariasi dalam interval waktu. Moaddeli et al. [15], telah meneliti dampak dari *backfilling algorithm* dalam pemesanan diawal yang *fleksibel*. *Backfilling* diusulkan untuk meningkatkan pemanfaatan sistem. Didalam *advance reservation*, yang fleksibel, pekerjaan pengguna dijadwalkan dengan batasan yang fleksibel. Waktu mulai tidak tetap dan dapat bervariasi dalam interval waktu. Idenya adalah mengidentifikasi node komputasi menganggur, yaitu saat di mana tidak ada pekerjaan yang ditugaskan untuk satu atau lebih node komputasi.

Pekerjaan di kepala antrian menunggu jika waktu yang dibutuhkan lebih besar dari node komputasi menganggur, sambil menunggu sampai ada node komputasi menganggur yang sesuai dengan pekerjaan ini. Backfilling memungkinkan pekerjaan di antrian tunggu yang lebih kecil dari pekerjaan di kepala antrian, yang cocok di node komputasi siaga untuk bergerak maju dan dieksekusi, pada node yang menganggur.

2.1.2.5. *Flexible Advance Reservation* (Dynamic, Physical View)

Strategi ini mengambil keuntungan dari pergeseran pemesanan yang dibuat lebih awal (tergantung pengingat kendala fleksibel) untuk membuat ruang agar reservasi baru masuk. Chunming et al. [16] didalam penelitiannya, telah memperkenalkan rentang waktu, ketika ada rentang waktu/range untuk waktu mulai dari pekerjaan. Rentang waktu ini disebut slack-time. Mereka mengusulkan mekanisme baru yang disebut FIRST (*FlexIble Reservation using Slack Time*). Keuntungan dari slack-time adalah bahwa waktu mulai dari pekerjaan dapat digeser untuk meningkatkan pemanfaatan sumber daya dan untuk mengurangi tingkat penolakan. Jika reservasi baru datang, sistem reservasi menjadwal ulang semua reservasi yang tidak dieksekusi satu per satu, sesuai dengan aturan FIFO (*First In First Out*), min slack, min-min, min-max, dan kebijakan hak pilih untuk melihat apakah ada solusi atau tidak . Jika tidak ada solusi maka permintaan reservasi baru akan ditolak.

Grandinetti et al. [36] didalam penelitian penjadwalan *local scheduler*, di mana sekelompok pekerjaan independen telah dijadwalkan pada situs tertentu, dengan pembatasan waktu pemrosesan (Yaitu, batas waktu, waktu mulai paling awal, reservasi) yang diberikan oleh sejumlah pengolah node. Diasumsikan bahwa semua node pengolahan adalah identik. beban kerja terdiri dari pekerjaan batch yang membutuhkan eksekusi ruang-sharing. Dengan

demikian, setiap pekerjaan yang antri dapat dimulai hanya jika ada node yang cocok dengan kapasitas yang dibutuhkan. dalam komputasi grid ditangani. Masalah diselidiki dan dirumuskan sebagai *Rectangle packing* dengan beberapa pendekatan heuristik yang dikembangkan untuk solusinya. Kinerja algoritma yang diusulkan dievaluasi di bawah skenario yang berbeda.

Metode dasar yang digunakan agar mudah untuk menerapkan dan pendekatan yang efisien maka didalam pendekatan penjadwalan mengacu pada *Rectangle packing* yang diusulkan[20]. Eksperimen menunjukkan bahwa strategi metode *Rectangle packing* mengungguli[20].

2.1.2.6. *Flexible Advance Reservation* (Dynamic, Logical View)

Perencanaan di awal dan strategi reservasi *First Come First Serve Ejecting Based Dynamic Scheduling* (FCFS-EDS) untuk meningkatkan pemanfaatan sumber daya dalam sistem grid. Untuk mencapai hal ini pekerjaan pengguna akan dipetakan ke node komputasi virtual yang disebut pandangan logis(*logical view*), yang kemudian dipetakan ke node komputasi yang sebenarnya yang disebut pandangan fisik(*physical view*), pada saat eksekusi. Sebuah lemma memastikan keberhasilan pemetaan tersebut dengan throughput meningkat. Pendekatan ini dapat dikategorikan sebagai *Flexible Advance Reservation* (*Dynamic, Logical View*)[17].

Strategi ini mengambil keuntungan dari pergeseran waktu mulai pekerjaan sebelumnya yang dibuat, dengan cara membuat ruang kosong untuk permintaan pekerjaan baru yang akan masuk. Didalam algoritma yang digunakan ada prosedure perhitungan matrik transpose, yang berfungsi untuk memetakan dari pandangan logis(*logical view*), ke node komputasi pandangan fisik(*physical view*) dengan adanya prosedure matrik transpose yang digunakan menyebabkan kinerja algoritmanya menjadi lebih lambat, karena *scheduler* terbebani harus melakukan proses perhitungan prosedure matrik transpose. Sehingga dengan adanya kelemahan ini masih dimungkinkan diusulkannya model algoritma lain, yang dapat mereduksi prosedure matrik transposenya.

2.1.5.2 *MPI job*

Untuk mengatasi aplikasi komputasi intensif yang lebih cepat kita bisa menggunakan tiga strategi: pertama, menggunakan hardware dengan prosesor kinerja tinggi(kerja keras), kedua; menggunakan algoritma yang lebih efisien dan lebih cepat(bekerja lebih cerdas), dan ketiga; menggunakan beberapa komputer untuk menyelesaikan tugas tertentu untuk mendapatkan bantuan[38]. Menggunakan hardware yang lebih cepat atau komputasi kinerja tinggi tampaknya menjadi solusi yang baik karena Hukum Moore[39] yang menyatakan bahwa jumlah transistor pada sirkuit terpadu ganda kira-kira setiap dua tahun. Tapi tetap komputer kinerja tinggi dengan algoritma yang efisien tidak cukup untuk mempercepat penyelesaian aplikasi komputasi intensif. Salah satu solusi yang mungkin untuk mempercepat penyelesaian aplikasi komputasi intensif adalah menghubungkan beberapa prosesor bersama-sama dan mengkoordinasikan komputasi mereka.

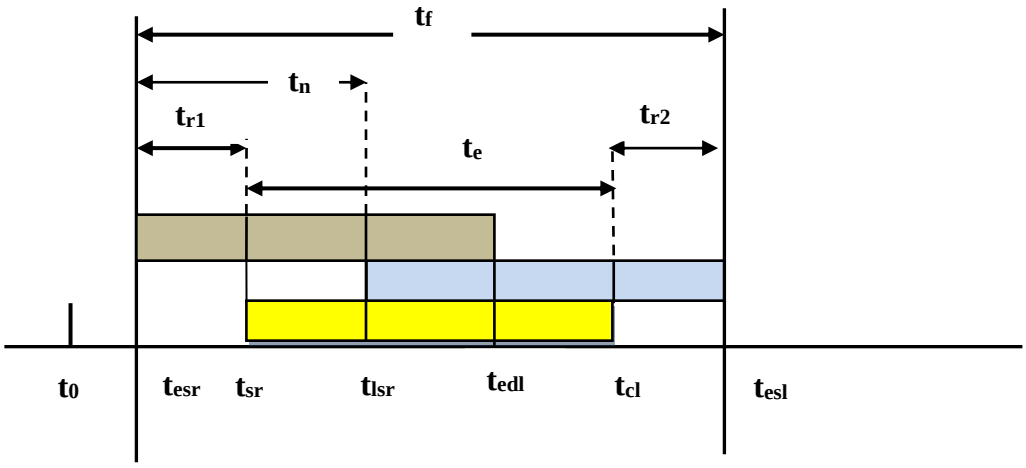
Alasan utama untuk menciptakan dan menggunakan komputer paralel adalah bahwa paralelisme adalah salah satu cara terbaik untuk mengatasi kecepatan bottleneck dari prosesor tunggal[40]. Ada tiga strategi untuk menciptakan aplikasi paralel. Strategi pertama didasarkan pada paralelisasi otomatis. Dengan strategi ini programmer tidak perlu repot-repot tentang tugas paralelisasi. kedua didasarkan pada penggunaan perpustakaan paralel. Dengan strategi ini kode paralel yang sama untuk beberapa aplikasi, dikemas ke dalam sebuah perpustakaan paralel. Strategi ketiga adalah recoding besar atau menyerupai kode dari awal dalam mengembangkan aplikasi paralel. programmer bebas untuk memilih bahasa dan model pemrograman yang digunakan untuk mengembangkan aplikasi paralel [41].

Message Passing Interface(MPI) adalah spesifikasi API yang memungkinkan aplikasi paralel untuk berkomunikasi satu sama lain dengan mengirim dan menerima pesan. Hal ini biasanya digunakan untuk program paralel yang berjalan pada cluster komputer dan

superkomputer. aplikasi paralel dapat disebut sebagai MPI aplikasi/job. Jika MPI job perlu menghitung node untuk menjalankan tugasnya, n pekerjaan ini harus mulai pada waktu yang sama.

2.2.1. Model Usulan penjadwalan

Diagram waktu pada Gambar 2.10 menunjukkan bagaimana permintaan ditangani. User Submite(numCN,t_{esr}, t_{lsr}, t_e,), Permintaan job dikirimkan pada waktu t_{esr}, dan permintaan job diterima pada waktu t_{sr}. Setelah permintaan job diterima, akan dicari apakah ada ruang kosong atau tidak, jika ada, maka job akan dieksekusi pada waktu t_{lsr}; serta sumber daya akan dialokasikan. Perbedaan antara t_{lsr} dan t_{esr} disebut dengan interval notifikasi/pemberitahuan.



Gambar 2.10. Usulan Penjadwalan advance reservation yang Fleksibel

t_0 : Waktu sekarang

t_{esr} : batas bawah untuk waktu mulai dari pekerjaan (waktu mulai paling awal)

t_{sr} : Waktu mulai untuk menjalankan pekerjaan ($t_{esr} \leq t_{sr} \leq t_{lsr}$)

t_{lsr} : Batas atas untuk memulai pelaksanaan pekerjaan (waktu mulai terakhir), didefinisikan sebagai $t_{lsr} = t_{esl} - t_e = t_n$

t_{esl} : Batas atas untuk mengakhiri waktu untuk menjalankan pekerjaan

t_e : Waktu eksekusi/durasi pekerjaan

t_n : Waktu notifikasi[34]

t_{r1},t_{r2} : t_{r1}(*left hole*), t_{r1}(*right hole*), t_r(waktu Relax), didefinisikan oleh $t_r = t_{r1} + t_{r2} = t_{esl} - t_{esr} - t_e$

t_{edl} : batas bawah sampai waktu untuk mengakhiri pelaksanaan pekerjaan, didefinisikan sebagai $t_{edl} = t_{esr} + t_e$

t_{cl} : Waktu penyelesaian untuk menjalankan pekerjaan ($t_{edl} \leq t_{cl} \leq t_{esl}$)

t_f : Waktu fleksibilitas, didefinisikan sebagai $t_f = t_{esl} - t_{esr}$

f : Tingkat fleksibilitas, didefinisikan sebagai $f = \frac{t_f}{t_e}$, di sini f≥1, (jika f= ∞, pekerjaan dianggap sebagai mode non job reservasi, Jika $t_s = t_0$ dan f = 1 pekerjaan untuk reservasi dipertimbangkan dengan prioritas paling atas mengarah ke modus pemesanan langsung yaitu, modus penjadwalan)[17].

userid : Identifikasi pengguna

jobid : Identifikasi pekerjaan

numj : Jumlah pekerjaan

numCN: Jumlah *sumber daya* komputer yang dibutuhkan

maxCN: Jumlah total *sumber daya* komputer

User Submite(numCN,t_{esr}, t_{lsr}, t_e,).

Fungsi t_{r1}(*left hole*), adalah memberikan ruang kosong, jika ada pekerjaan berikutnya masuk maka slot sebelumnya dapat digeser kekanan agar pekerjaan yang membutuhkan ruang kosong berikutnya dapat menempatinya.
Fungsi t_{r1}(*right hole*), misalkan user1 butuh waktu eksekusi pekerjaannya selama 10 menit, dimulai dari t=10 sampai t=11, sedangkan user2 butuh waktu untuk mengeksekusi pekerjaannya selama 15 menit, waktu mulai paling awal eksekusi dapat dimulai dari t=10 dan waktu mulai eksekusi pekerjaannya paling akhir t=12. kemudian user1 membatalkan pekerjaannya, sedangkan user3 masuk membutuhkan eksekusi pekerjaannya selama 10 menit, dimulai dari t=13 sampai t=14, maka user2 dapat digeser kekiri untuk memberi ruang user3 untuk menempati ruang t=13 sampai t=14.

2.1.5. *Type Job*

Perencanaan strategi penjadwalan untuk *advance reservation* akan diuji untuk dapat menangani dua jenis pekerjaan, yaitu pekerjaan serial(parametrik job) dan pekerjaan parallel(MPI Jobs).

2.1.5.1. *Parametric job*

Pekerjaan parametrik adalah pekerjaan yang serupa yang hanya berbeda dalam argumen atau file input/output. Dengan jenis pekerjaan parametrik, kita bisa mengirimkan sebagian besar pekerjaan sebagai pekerjaan tunggal.

Dalam sains dan teknik eksperimen komputasi parametrik menjadi sangat penting sebagai sarana untuk mengeksplorasi perilaku sistem yang kompleks. Misalnya, perilaku sayap di pesawat dapat dieksplorasi dengan menjalankan model komputasi dari airfoil beberapa waktu, di mana setiap kali menjalankan model parameter komputasi, seperti sudut serangan, kecepatan udara, yang dapat bervariasi[42].

2.1.5.2 *MPI job*

Untuk mengatasi aplikasi komputasi intensif yang lebih cepat kita bisa menggunakan tiga strategi: pertama, menggunakan hardware dengan prosesor kinerja tinggi(kerja keras), kedua; menggunakan algoritma yang lebih efisien dan lebih cepat(bekerja lebih cerdas), dan ketiga; menggunakan beberapa komputer untuk menyelesaikan tugas tertentu untuk mendapatkan bantuan[38]. Menggunakan hardware yang lebih cepat atau komputasi kinerja tinggi tampaknya menjadi solusi yang baik karena Hukum Moore[39] yang menyatakan bahwa jumlah transistor pada sirkuit terpadu ganda kira-kira setiap dua tahun. Tapi tetap komputer kinerja tinggi dengan algoritma yang efisien tidak cukup untuk mempercepat penyelesaian aplikasi komputasi intensif. Salah satu solusi yang mungkin untuk mempercepat penyelesaian aplikasi komputasi intensif adalah menghubungkan beberapa prosesor bersama-sama dan mengkoordinasikan komputasi mereka.

Alasan utama untuk menciptakan dan menggunakan komputer paralel adalah bahwa paralelisme adalah salah satu cara terbaik untuk mengatasi kecepatan bottleneck dari prosesor tunggal[40]. Ada tiga strategi untuk menciptakan aplikasi paralel. Strategi pertama didasarkan pada paralelisasi otomatis. Dengan strategi ini programmer tidak perlu repot-repot tentang tugas paralelisasi. kedua didasarkan pada penggunaan perpustakaan paralel. Dengan strategi ini kode paralel yang sama untuk beberapa aplikasi, dikemas ke dalam sebuah perpustakaan paralel. Strategi ketiga adalah recoding besar atau menyerupai kode dari awal dalam mengembangkan aplikasi paralel. programmer bebas untuk memilih bahasa dan model pemrograman yang digunakan untuk mengembangkan aplikasi paralel [41].
Message Passing Interface(MPI) adalah spesifikasi API yang memungkinkan aplikasi paralel untuk berkomunikasi satu sama lain dengan mengirim dan menerima pesan. Hal ini biasanya digunakan untuk program paralel yang berjalan pada cluster komputer dan superkomputer. aplikasi paralel dapat disebut sebagai MPI aplikasi/job. Jika MPI job perlu menghitung node untuk menjalankan tugasnya, n pekerjaan ini harus mulai pada waktu yang sama.

3.1. METODE PENELITIAN

Kebutuhan perangkat keras dan perangkat lunak yang diperlukan untuk menjalankan simulator Gridsim, didalam menguji usulan algoritma strategi penjadwalan *advance reservation* dalam Sistem grid, antara lain :

1. Perangkat Keras (*Hardware*).
   a. Prosesor : Amd A 10-5750 M APU 2.50 GHz
   b. Ram : 16 GB.
   c. Disk drive : 320 GB.
   d. Display : 12” Wide-screen.
2. Perangkat lunak (*software*)
   a. Sistem Operasi Windows 8 64 bit.
   b. Eclipse Kepler Build id:20130614-0229AppServ v2.5.8 : *Web Server*.

3.1.1. Metode Pengumpulan Data

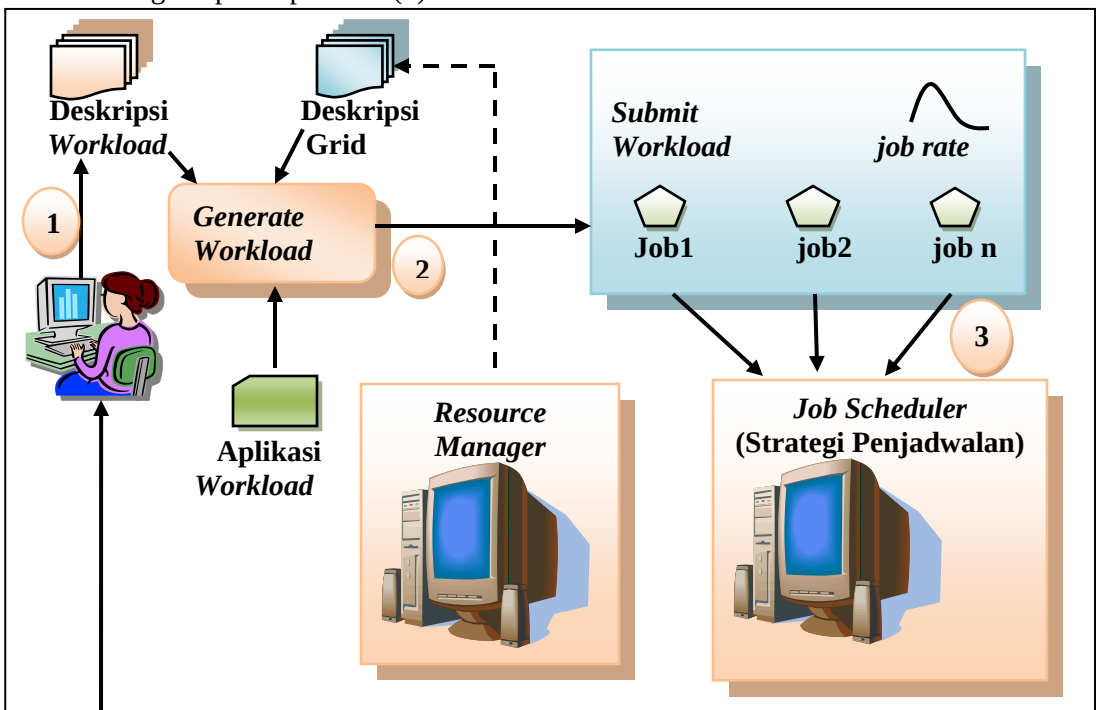
Metode pengumpulan data yang digunakan adalah metode studi literature yang yang mengacu pada data hasil penelitian[17,37].

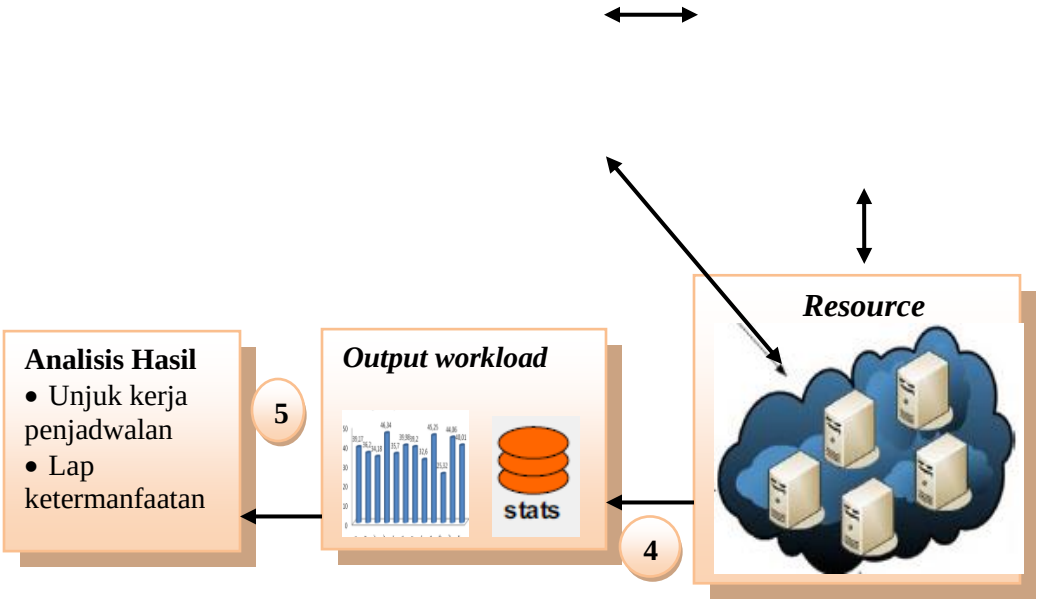
Workload Generator

Untuk memeriksa kinerja penjadwalan yang diusulkan dan strategi penjadwalan *advance reservation*, didalam penelitian ini menggunakan data[17,37] dan *workload generator* untuk membangkitkannya. Output dari *workload generator* ini digunakan sebagai masukan pada penjadwalan *advance reservation* yang `diusulkan. Untuk menghasilkan *workload generator* yang baik, maka kita harus menentukan karakteristik dari *workload generator* tersebut. Karakteristik *workload generator* didalam penelitian ini[37]:

1. Laju kedatangan pekerjaan(slot waktu) yang masuk mengikuti distribusi Poisson.
2. Rentang waktu eksekusi dari setiap permintaan pemesanan, terdistribusi secara merata.
3. waktu mulai paling awal dari setiap pemesanan, terdistribusi secara merata.
4. Persentase *advance reservation* yang fleksibel dipilih secara acak.
5. Kisaran waktu relax untuk setiap *advance reservation* yang fleksibel, terdistribusikan secara merata.
6. Jumlah sumber daya yang diperlukan, terdistribusikan secara merata.
7. Didalam penelitian ini lebar *timeslot* 5 menit

Gambar 3.3. menggambarkan penggunaan *workload generator*[37]. Pertama, user mensubmit deskripsi pekerjaan(1). Berdasarkan deskripsi pekerjaan user, dan informasi tentang deskripsi grid, kemudian dihasilkannya beban kerja oleh *workload generator*(2). Hasil dari *workload generator* kemudian diserahkan atau dikirimkan kembali ke grid(3). Lingkungan jaringan bertanggung jawab untuk melaksanakan pekerjaan dan mengembalikan hasil mereka(4). Dihasilkannya laporan secara rinci dari pekerjaan, dan. Akhirnya, pengguna memproses semua hasil dalam langkah pasca-produksi(5)





Gambar 3.3. Model proses *generate workload* data *advance reservation* pada grid computing.

3.1.2. Arsitektur Sistem Grid yang diusulkan

Arsitektur grid yang diusulkan ini menunjukkan interaksi antara berbagai komponen dalam model penjadwalan *advance reservation*. Arsitektur sistem grid ini berisi strategi model penjadwalan yang diusulkan, struktur data, serta mengakomodasi reservasi untuk pekerjaan serial dan parallel(MPI). Penelitian yang akan dilakukan menggunakan simulator gridsim, untuk menguji kinerja sistem yang diusulkan.

Gambar 3.4. menunjukkan interaksi antara komponen yang relevan dalam model, komponen yang terlibat dalam model ini adalah: Administrator, User, Modul perencanaan dan reservation, Alokasi Sumber Daya.

a. Administrator menetapkan kondisi awal untuk model penjadwalan, dan mempersiapkan laporan.
b. Pengguna mengirimkan deskripsi pekerjaan.
c. Modul Perencanaan menjadwal pekerjaan yang masuk dari pengguna, kemudian diproses menggunakan strategi penjadwalan yang diusulkan.
d. Modul penjadwalan pemesanan memetakan pekerjaan manual ke node sumber daya fisik yang diperlukan, menggunakan system yang diusulkan.

Penjelasan langkah-langkah interaksi antar komponen-komponen seperti di bawah ini:

LANGKAH-1 (Inisialisasi)

Administrator melakukan inisialisasi(1) pada modul Perencanaan. Dalam modul "Perencanaan", Administrator menentukan strategi penjadwalan yang diusulkan dan mendefinisikan parameter untuk pengajuan pekerjaan seperti: waktu mulai paling awal, waktu mulai, durasi pekerjaan, UserId dan Jobid, jumlah node komputasi yang dibutuhkan dan kapasitas sumber daya komputasi.

LANGKAH-2 (User mengirim pekerjaan job)

Pengguna mendefinisikan parameter untuk penjadwalan pekerjaan. Parameter tersebut: UserId, JobId, waktu mulai paling awal, waktu mulai, durasi pekerjaan, dan jumlah node komputasi yang dibutuhkan. Pengguna mengirimkan profil pekerjaannya pada Modul "Perencanaan".

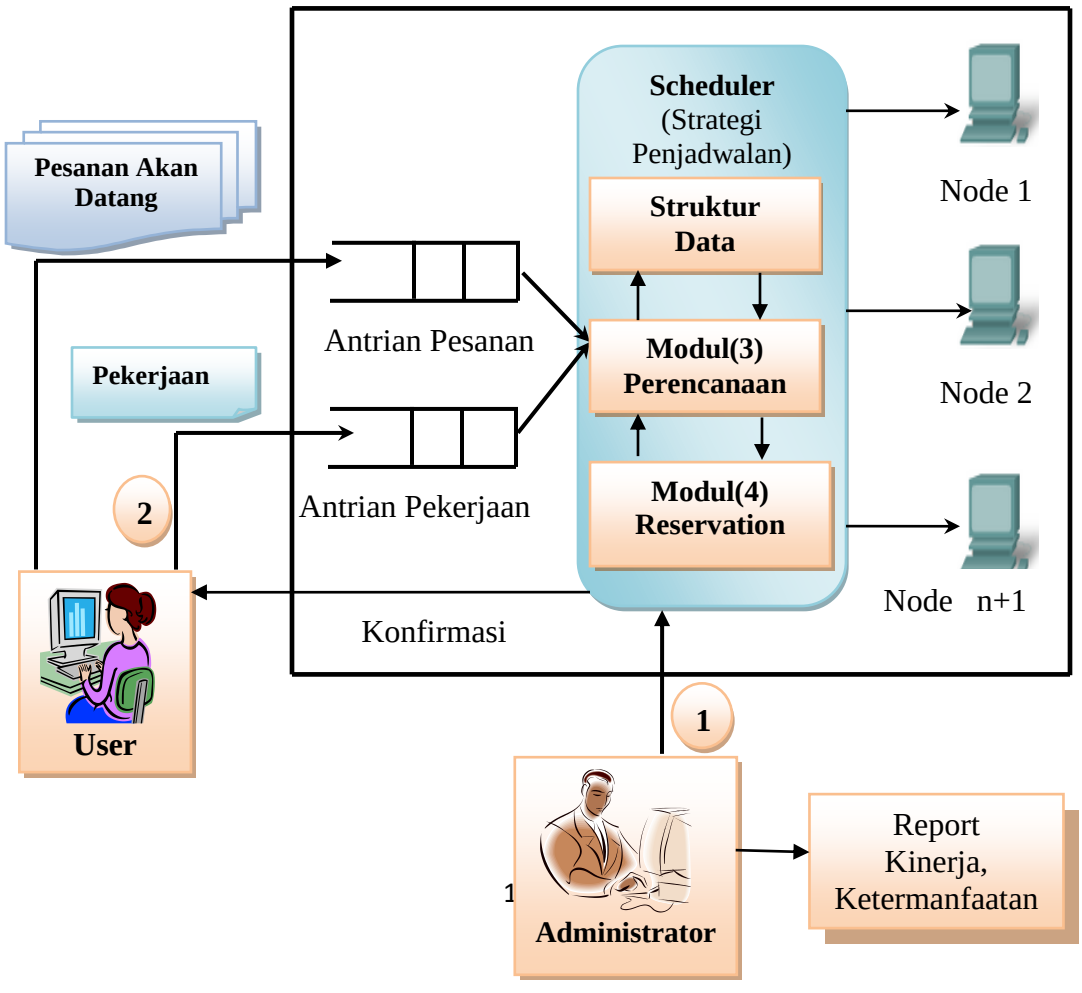
LANGKAH-3 (Perencanaan)

Dalam Modul "Perencanaan", pekerjaan yang diterima diproses untuk dilihat apakah pekerjaan dapat diterima berdasarkan uraian/parameter pekerjaan. Jika diterima, rincian pekerjaan diserahkan dan dikirim ke modul "Reservation" dan user di “konfirmasi” bahwa pekerjaannya

diterima. Selain itu status pekerjaan "Ditolak" karena sumber daya yang memadai (*compute node*) tidak tersedia. Pekerjaan tersebut akan dihapus dari daftar.

LANGKAH-4 (Proses eksekusi penjadwalan)

Di modul “Reservation”, dikonfirmasi kerja di LANGKAH-3 dan dijadwalkan pada sumber daya yang diperlukan secara fisik. Menurut strategi penjadwalan yang diusulkan, Modul "Reservation" selalu akan menemukan sumber daya untuk pekerjaan yang dijadwalkan dalam pandangan logis, dan kemudian Modul "Reservation" mengeksekusi pekerjaan pada node/resource komputasi yang sebenarnya.



Gambar 3.4. Model usulan sumber daya yang mendukung *advance reservation* pada grid.

3.1. Hasil

Untuk menjelaskan bagaimana FCFS-LRH bekerja, Jika diketahui Parameter permintaan job pesanan user seperti ditunjukkan pada table 8. maka MaxCn menunjukkan jumlah maksimum resource yang disediakan(physical node), maka maxCn=5(C0-C4), NumCn=5(v0-v4) menunjukkan jumlah resource yang dibutuhkan user yang disebut virtual node, $NumCn \leq MaxCn$. Sedangkan Numj menunjukkan jumlah pekerjaan yang dibutuhkan user. Jika Userid5 mengirim permintaan resource untuk mengerjakan pekerjaannya yang membutuhkan 3 time slot dari 18 sampai 20 dengan 1 resource yang independent job dan tidak dapat digeser(t_{esr} = t_{lsr} =18, t_e =3 , NumCn=1, Numj=1).

Tabel 8. Parameter permintaan job pesanan user

Userid	t_{esr}	t_{lsr}	t_e	NumCn	Numj
1	15	15	1	2	2
2	16	16	2	1	1
3	17	17	3	1	1
4	16	16	3	1	1
5	18	18	3	1	1
6	14	14	1	2	2
7	12	12	2	1	1
8	12	12	2	1	1
9	17	18	2	1	1
10	19	19	2	1	1
11	12	12	2	1	1
12	16	16	2	1	1

v4									
v3		11,1				12,1	4,1	9,1	
v2	11,1	8,1			12,1	4,1	9,1	3,1	5,1
v1	8,1	7,1	6,2	1,2	4,1	2,1	3,1	5,1	10,1
v0	7,1		6,1	1,1	2,1	3,1	5,1	10,1	
	12	13	14	15	16	17	18	19	20

Gambar 2. 12 permintaan pekerjaan telah ditempatkan pada Logical View

Gambar 2 menunjukkan 12 permintaan pekerjaan telah ditempatkan pada logical view, dimana sumbu x menunjukkan time slot, sedangkan sumbu y menunjukkan virtual compute node. Jika User17 datang dan membutuhkan 3 compute node dengan time slot yang dibutuhkan mulai dari 13 sampai 15 dan tiap pekerjaan dapat digeser dengan batas akhir pekerjaan 15(t_{esr} = 13, t_{lsr} =15, t_e =3 , NumCn=3, Numj=3), lihat gambar 3.

	17,3	17,3	17,3						
	17,2	17,2	17,2						
	17,1	17,1	17,1						
v4									
v3		11,1				12,1	4,1	9,1	
v2	11,1	8,1			12,1	4,1	9,1	3,1	5,1
v1	8,1	7,1	6,2	1,2	4,1	2,1	3,1	5,1	10,1
v0	7,1		6,1	1,1	2,1	3,1	5,1	10,1	
	12	13	14	15	16	17	18	19	20

Gambar 3.User17 melakukan permintaan pekerjaan.

Gambar 4. Menunjukkan hasil dari FCFS-LRH bahwa user17 telah dialokasikan sesuai dengan permintaan, jika menggunakan penjadwalan konventional reservation or rigid reservation hanya satu independent job dari user17 yang akan dialokasikan sedangkan dua lainnya akan ditolak. Dari gambar 4 menunjukkan pula bahwa permintaan pekerjaan user sukses dan user akan diberitahu bahwa pekerjaannya diterima.

v4		17,2	17,3	17,3					
v3		11,1	17,2	17,2	17,3	12,1	4,1	9,1	
v2	11,1	8,1	17,1	17,1	12,1	4,1	9,1	3,1	5,1
v1	8,1	7,1	6,2	1,2	4,1	2,1	3,1	5,1	10,1
v0	7,1	17,1	6,1	1,1	2,1	3,1	5,1	10,1	
	12	13	14	15	16	17	18	19	20

Gambar 4. User17 telah dialokasikan pada logical view

3.2. Pemetaan logical view ke physical view

Kami mengaransi bahwa pekerjaan pada logical view dipetakan dan akan dikerjakan pada Physical View(gambar 5) dan setiap masukan permintaan pekerjaan akan dikerjakan pada node komputasi yang sama.

C4		17,2							
C3		17,1			12,1		9,1		
C2	11,1		17,3			3,1			
C1	8,1		6,2	1,2	4,1			10,1	
C0	7,1		6,1	1,1	2,1		5,1		
	12	13	14	15	16	17	18	19	20

Gambar 5. Pemetaan node komputasi pada Physical View

4. Kesimpulan

Didalam penelitian ini kami mengusulkan keterbaruan strategi penjadwalan pekerjaan yang akan datang menggunakan strategi FCFS-LRH untuk meningkatkan

ketermanfaatan cluster system. Strategi ini memetakan pekerjaan pada virtual node (logical view) dan memberkan garansi bahwa pekerjaan akan dikerjakan pada resource computer yang sebenarnya(physical view) untuk dieksekusi. Kami telah melakukan penelitian dan membandingkannya dengan FCFS-EDS hasilnya didalam model yang kami usulkan permutasi *Backward*(mundur) dapat mereduksi proses transpose yang ada di FCFS-EDF. Serta dari ketermanfaatan lebih baik dari model penjadwalan konvensional FCFS.

DAFTAR PUSTAKA

[1] Carsten Franke_, Uwe Schwiegelshohn, and Ramin Yahyapour , Grid scheduling by on-line rectangle packing, network an international journal,wiley, 2004.

[2] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1999.

[3] M. Livny and R. Raman. High-Throughput Resource Management. In I. Foster and C. Kesselman, editors, *The Grid - Blueprint for a New Computing Infrastructure*, pages 311–337. Morgan Kaufmann,1999.

[4] I. Foster and A. Iamnitchi, *On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing*, in Proc. of 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), Berkeley, CA, USA, February 2003

[5] S. Uwe and Y. Ramin, "Attributes for Communication Between Grid Scheduling Instances," in *Grid Resource Management: State of the Art and Future Trends*, Norwell, MA, USA, Kluwer Academic Publishers, 2004.

[6] A. Sulistio and R. Buyya, "A Grid simulation infrastructure supporting advance reservation," in *16th International Conference on Parallel and Distributed Computing and Systems, ACTA Press*, Calgary, 2004.

[7] W. Smith, dkk, "Scheduling with Advanced Reservations," in *14th IEEE International Symposium on Parallel and Distributed Processing. IEEE Press*, Cancun, 2000.

[8] I. Foster, dkk, "A Distributed Resource Management Architecture that Supports Advance Reservation and Co-Allocation," in *7th IEEE International Workshop on Quality of Service, IEEE Press*, London,1999.

[9] K. Czajkowski, dkk, "A resource management architecture for metacomputing systems," in *4th Workshop on Job Scheduling Strategies for Parallel Processing. LNCS vol. 1459. Springer*, London,1998.

[10] A. Sulistio , dkk, "On Incorporating an On-line Strip Packing Algorithm into Elastic Grid Reservation-based Systems," in *13th International Conference on Parallel and Distributed Systems (ICPADS'07), IEEE Press*, Hsinchu,2007.

[11] A. W. Mu'alem and D. G. Feitelson, "Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling," *IEEE Transactions on Parallel and Distributed Systems,* vol. 12, 2001.

[12] P. Xiao, dkk,2008, "A Novel Statistic-based Relaxed Grid Resource Reservation Strategy," in *9th International Conference for Young Computer Scientists. IEEE Press*, Hunan.

[13] C. B. Lee, dkk, "On the user-scheduler dialogue: Studies of user-provided runtime estimates and utility functions," *International Journal of High Performance Computing Applications,* vol. 20,2006.

[14] C. Castillo, G. Rouskas and K. Harfoush, "On the design of online scheduling algorithms for advance reservations and QoS in grids," in *Parallel and Distributed Processing Symposium, IPDPS, IEEE International*, Long Beach, USA , 2007.

[15] C. Moaddeli, dkk, "Flexible Advance Reservation Impact on Backfilling Scheduling Strategies," in *7th International Conference on Grid and Cooperative Computing, IEEE Press*, Shenzhen, 2008.

[16] H. Chunming, H. Jinpeng and W. Tianyu , "Flexible Resource Reservation Using Slack Time for Service Grid," in *12th International Conference on Parallel and Distributed Systems, IEEE Press*, Washington, 2006.

[17] U. Rusydi, dkk, "Advance Planning and Reservation in a Grid System," in *NDT 2012. CCIS/LNCS, vol. 293. Springer, Heidelberg* , Dubai, 2012.

[18] J. MacLaren, "Advance Reservations: State of the Art," Working Draft, Global Grid Forum, 2003.

[19] R. Buyya and M. Murshed, "GridSim: A Toolkit for the Modeling and Simulation of Distributed Management and Scheduling for Grid Computing," *Concurrency and Computation: Practice and Experience (CCPE),* vol. 14, no. 13, 2002.

[20] Caramia, M., Giordani, S., Iovanella, A.: Grid scheduling by on-line rectangle packing. Networks **44**(2), 106–119, 2004.

[21] L. -O. Burchard, "Analysis of data structures for admission control of advance reservation requests," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 413 – 424,2005.

[22] L. -O. Burchard and H. -U. Heiss, "Performance Evaluation of Data Structures for Admission Control in Bandwidth Brokers," in *International Symposium of Performance Evaluation of Computer and Telecommunication Systems (SPECTS '02), pp. 652-659*, San Diego,2002.

[23] R. Guerin and A. Orda, "Networks with Advance Reservations: The Routing Perspective. In Proceeding of the Conference on Computer Communications," in *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Reaching the Promised Land of Communications (INFOCOM 2000), pp. 118-127, IEEE Press*, Tel Aviv,2000.

[24] O. Schelen, A. Nilsson, J. Norrgard and S. Pink, "Performance of QoS Agents for Provisioning Network Resources," in *Seventh International Workshop on Quality of Service (IWQoS '99), pp. 17-26, IEEE Press*, London,1999.

[25] A. Brodnik and A. Nilsson, "A static data structure for discrete advance bandwidth reservations on the internet," in *Swedish National Computer Networking Workshop (SNCNW)*, Stockholm,2003.

[26] R. Brown, "Calendar queues: A fast O(1) priority queue implementation for the simulation event set problem," *Communications of the ACM*, vol. 31, no. 10, pp. 1220-1227,1988.

[27] A. Sulistio, U. Cibej, S. Prasad and R. Buyya, "GarQ: An Efficient Scheduling Data Structure for Advance Reservations of Grid Resources," *International Journal of Parallel, Emergent and Distributed Systems (IJPEDS),* vol. 24, no. 1, pp. 1-19,2008.

[28] T. Wang and J. Chen, "Bandwidth tree – a data structure for routing in networks with advanced reservations," in *21st Interrnational Performance, Computing, and Communications Conference (IPCCC), pp. 37–44, IEEE Press*, Phoenix,2002.

[29] L. Yuan, C. -K. Tham and A. L. Ananda, "A probing approach for effective distributed resource reservation," in *the 2nd International Workshop on Quality of Service in Multiservice IP Networks, pp. 672–688, Springer-Verlag*, Milan,2003.

[30] K. Melhorn, Data structures and algorithms III: Multi-dimensional searching and computational geometry, New York : Springer-Verlag,1984.

[31] I. Foster, C. Kesselman and S. Tuecke, *The Anatomy of the Grid*: *Enabling Scalable Virtual Organizations*, in the International J. Supercomputer Applications, 15(3), pp. 200-220, fall 2001.

[32] Q. Xiong, C. Wu, J. Xing, L. Wu and H. Zhang, "A linked-list data structure for advance reservation admission control," in *the 3rd International Conference on Networking and Mobile Computing (ICCNMC). LNCS, vol. 3619, pp. 901-910,* Zhangjiajie, China,2005.

[33] W. Tao and C. Jianer, "Bandwidth Tree - A Data Structure for Routing in Networks with Advanced Reservations," in *21st IEEE International Conference on Performance, Computing, and Communications (IPCCC 2002), pp. 37-44, IEEE Press*, Phoenix,2002.

[34] Charlie Xu and J. W. Wong, *"*Scheduling algorithms for advance resource reservation", H. R. van As (ed.), *High Performance Networking,* Springer, New York,1998.
[35] R. Buyya and D. Abramson and J. Giddy and H. Stockinger, *Economic Models for Resource Management and Scheduling in Grid Computing,* in J. of Concurrency and Computation: Practice and Experience, Volume 14, Issue.13-15, pp. 1507-1542, Wiley Press, December 2002
[36] L. Grandinetti, Francesca Guerriero, Luigi Di Puglia Pugliese, Mehdi Sheikhalishahi, "Heuristics for the local grid scheduling problem with processing time constraints", J Heuristics 21:523–547, Springer, New York,2015.
[37] Alexandru Iosup and Dick H.J. Epema, Shynthetic grid workloads with Ibis, Koala, and Grenchmark, *Delft, The Netherlands.*2007.
[38] G. Pfister, In Search of Clusters, New Jersey: Prentice Hall PTR, NJ, 2nd Edition,1998.
[39] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics,* vol. 38, pp. 114-117,1965.
[40] K. Hwang and Z. Xu, Scalable Parallel Computing: Technology, Architecture, Programming, New York: WCB/McGraw-Hill,1998.
[41] P. T. Matthew and S. K. W. Johnny, "A Task Migration Algorithm for Heterogeneous Distributed Computing Systems," *System and Software,* vol. 41, pp. 175-188,1988.
[42] A. Sulistio, U. Cibej, S. Prasad and R. Buyya, "GarQ: An Efficient Scheduling Data Structure for Advance Reservations of Grid Resources," *International Journal of Parallel, Emergent and Distributed Systems (IJPEDS),* vol. 24, no.1,2008.
[43] Parallel Workloads Archive, http://www.cs.huji.ac.il/labs/parallel/workload/.,2004.
[44] Tuomas Ja'rvinen, Stride Permutation Networks for Array Processors, Journal of VLSI Signal Processing 49, 51–71,2007.
[45] Milovanovic´,E.R. Glogic´, Milovanovic, Bekakos, Stojcˇev, Permutation Matrices of Reverse r-th Stride, SER. A: APPL. MATH. INFORM. AND MECH. vol. 5.2, 79-84,2013.
[46] Buyya, R., Murshed, M.: Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurr. Comput.: Pract. Experience **14**(13-15), 1175–1220,2002
[47] Howell, F., McNab, R.: Simjava: a discrete event simulation library for java. Simul. Ser. **30**, 51–56,1998.
[48] R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. Proceedings of the IEEE, 93(3):698{714, 2005.
[49] C. S. Yeo, R. Buyya, M. D. de Assuncao, J. Yu, A. Sulistio, S. Venugopal, and M. Placek. *Utility Computing and Global Grids. In H. Bidgoli, editor, The Handbook of Computer Networks,* volume III Part 1, New York, USA, John Wiley & Sons,2007
[50] W. Cirne, F. Brasileiro, J. Sauve, N. Andrade, D. Paranhos, E. Santos-Neto, and R. Medeiros. Grid computing for bag of tasks applications. In Proceedings of the 3rd IFIP Conference on E-Commerce, E-Business and E-Government, Sao Paolo, Brazil, Sep. 2003.
[51] D. Abramson, J. Giddy, and L. Kotler. High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid? In Proceedings of the 14th International Parallel and Distributed Processing Symposium(IPDPS'00), Cancun, Mexico, May 1-5, 2000.
[52]. Libing Wu, Ping Dang, Tianshui Yu, and Lei Nie, *Research on Efficient Non-slotted Tree Structures for Advance Reservation*, J. Su et al. (Eds.): ICoC, CCIS 401, pp. 50–61,2013.
[53]. A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. Network and Computer Applications, 23:187-200, 2001.
[54] W. Hoschek, F. J. Ja´en-Mart´ınez, A. Samar, H. Stockinger, and K. Stockinger. Data management in an international data grid project. In Proceedings of the 1st International Workshop on Grid Computing (Grid'00), Bangalore, India, Dec. 17, 2000.

[55]. K. Seymour, A. YarKhan, S. Agrawal, and J. Dongarra. NetSolve: Grid Enabling Scientific Computing Environments. In L. Grandinetti, editor, Grid Computing and New Frontiers of High Performance Processing, volume 14 of Advances in Parallel Computing, pages 33-51,Netherlands, Elsevier,2005.

[56] L. Childers, T. Disz, R. Olson, M. E. Papka, R. Stevens, and T. Udeshi. Access Grid: Immersive Group-to-Group Collaborative Visualization. In Proceedings of the 4th International Immersive Projection Technology Workshop, Ames, USA, June 19-20, 2000.

[57] M. Cannataro and D. Talia. The Knowledge Grid. Communications of the ACM, 46(1):89-93, 2003.

[58] S. Graupner, J. Pruyne, and S. Singhal. Making the Utility Data Center a Power Station for the Enterprise Grid. Technical Report HPL{2003{53, HP Labs, Palo Alto, USA, 2003.

[59] R. Buyya and S. Venugopal. The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report. In Proceedings of the 1st International Workshop on Grid Economics and Business Models (GECON'04), Seoul, Korea, April 23, 2004.

[60] G. Sipos and P. Kacsuk. Multi-Grid, Multi-User Workflows in the P-GRADE Portal. Journal of Grid Computing, 3(3{4):221-238, Sep. 2005.

[61] H. Gibbins, K. Nadiminti, B. Beeson, R. Chhabra, B. Smith, and R. Buyya. The Australian BioGrid Portal: Empowering the Molecular Docking Research Community. In Proceedings of the 3rd APAC Conference and Exhibition on Advanced Computing, Grid Applications and eResearch (APAC'05), Gold Coast, Australia, Sep. 26-30, 2005.

[62] M. Murshed, R. Buyya, Using the GridSim Toolkit for Enabling Grid Computing Education, in proc of the conf on communication networks and distributed systems modeling and simulation,citeSeer,2002.