Pemodelan dan Penjadwalan Tugas dalam Sistem Terdistribusi dengan Pendekatan Algoritma Path-Planning A-Star

Nicholas Liem - 13521135
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521135@std.stei.itb.ac.id

***Abstract*—** Masalah penjadwalan tugas pada sistem terdistribusi memerlukan solusi yang optimal dalam pembagian tugas untuk mempercepat komputasi dan penyelesaian tugas yang ada. Untuk mencapai hasil tersebut, suatu algoritma penjadwalan haruslah suatu penjadwalan yang cepat dan mudah beradaptasi dan sebaiknya tidak mengetahui keseluruhan informasi. Oleh sebab itu, algoritma penjadwalan sebaiknya algoritma yang heuristik. Salah satu algoritma heuristik adalah algoritma perencanaan jalur A-Star. Namun, masalah penjadwalan yang ada tidak bisa langsung diselesaikan menggunakan algoritma A-Star tetapi haruslah dilakukan suatu pemetaan masalah dari masalah penjadwalan menjadi masalah perencanaan jalur. Dalam makalah ini, akan ditunjukkan bagaimana caranya secara dasar dalam bentuk prototipe.

***Keywords*—** *sistem terdistribusi, algoritma, path-planning, a-star*

I. Sistem Terdistribusi

Sistem terdistribusi adalah sekumpulan komputer-komputer yang terpisah secara fisik tetapi terhubung melalui koneksi tersentralisir yang dilengkapi oleh sebuah *distributed-system software*. Masing-masing dari komputer ini dapat berkomunikasi dan melakukan komputasi melalui suatu jalur komunikasi di mana semua *resources*nya dapat dipakai bersama. Perangkat lunak yang mengatur dan melakukan manajemen terhadap sumber daya ini dapat menugaskan komputer-komputer yang terhubung untuk mengerjakan tugas-tugas tertentu. Pengerjaan tugas yang diatur oleh perangkat lunak ini tentunya bermanfaat sebab suatu tugas yang besar dapat dipecah menjadi beberapa sub-tugas sehingga tugas dapat diselesaikan dalam waktu yang cepat.

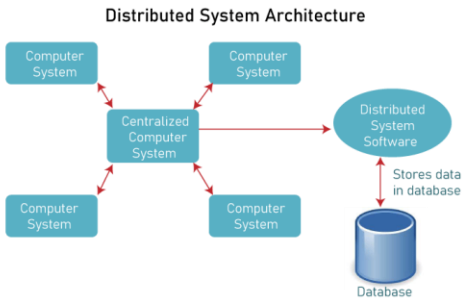


Fig. 1. *Skema dasar sistem terdistribusi*
Sumber: https://www.javatpoint.com/

A. Karakteristik dari Sistem Terdistribusi

Karakteristik dari suatu sistem terdistribusi adalah pembagian sumber daya, heterogenitas, skalabilitas, konkurensi, transparansi, keterbukaan, dan toleransi terhadap kesalahan. Setiap sistem terdistribusi dikatakan suatu sistem yang terdistribusi jika setiap komponen komputer pada sistem memiliki suatu hal yang dapat dibagi bersama. Ketika mereka terhubung pada koneksi tersebut, mereka dapat menggunakan sumber daya bersama, beberapa hal dari sumber daya tersebut bisa dalam bentuk file, perangkat keras, dan data.

Heterogenitas dalam sistem terdistribusi artinya setiap komponen komputer pada sistem terdistribusi itu bisa tidak sama, sehingga komputer-komputer yang terhubung bisa memiliki banyak variansi dan perbedaan. Perbedaan ini bisa dalam bentuk perangkat keras, bahasa pemrograman, sistem operasi, dan lain sebagainya.

Skalabilitas juga merupakan salah satu hal penting dari sistem terdistribusi. Aspek skalabilitas artinya sistem terdistribusi harus dapat dikembangkan dan diperluas dengan menambah jumlah pengguna maupun unit komputasi. Artinya, sistem terdistribusi harus dapat mengatur penambahan jumlah pengguna dan komputer.

Aspek konkurensi juga salah satu hal yang penting dari sistem ini. Setiap komputer yang terhubung pada sistem ini harus dapat mengerjakan beberapa aktivitas maupun tugas secara konkuren atau *simultaneously*.

Aspek transparansi dalam sistem ini berarti bahwa seluruh jaringan dalam komputasi itu tersembunyi bagi pengguna sehingga pengguna tidak tahu tentang *inner workings* dari sistem ini.

Fault tolerance adalah salah satu aspek lain yang berarti bahwa sistem ini harus bisa meng*handle* kesalahan dan kegagalan implikasi dari aspek ini adalah sistem ini dikenal sebagai sistem yang memiliki *availability* yang tinggi. Aspek terakhir adalah aspek keterbukaan di mana sistem ini harus

terbuka terhadap penambahan komponen dan menjadi suatu sistem yang *extensible*.

B. Kelebihan dan Kekurangan dari Sistem Terdistribusi

Kelebihan dari sistem terdistribusi adalah terdapatnya aplikasi yang terdistribusi, informasi yang ada pada sistem dapat disebar pada pengguna dari berbagai tempat di seluruh dunia, pembagian sumber daya, memiliki rasio *price-performance* dan fleksibilitas yang tinggi, memiliki waktu respons yang singkat dan memiliki *throughput* yang tinggi, meningkatnya *availability* walaupun terjadi kesalahan (*fault tolerance*), dan sistem ini dapat dikembangkan atau di*extend* dengan mudah.

Kekurangan dari sistem terdistribusi adalah masalah sekuritas yang disebabkan oleh pembagian sumber daya kepada banyak *instance*, terdapat *lag* pada sistem yang disebabkan oleh banyaknya transfer data. Sistem basis data yang menggunakan sistem terdistribusi cukup sulit untuk di*manage* serta dikembangkan.

C. Tipe dari Sistem Terdistribusi

Tipe-tipe dari sistem terdistribusi ada banyak. Beberapa di antaranya yang terkenal adalah tipe *client-server*, *peer-to-peer*, *three-tier*, dan *N-tier*. Masing-masing dari tipe ini memiliki keunikannya sendiri-sendiri. Pada tipe *client-server*, jaringan terdiri dari komputer server dan komputer klien, masing-masing dan komputer klien terhubung melalui suatu koneksi terhadap komputer server dan masing-masing perangkat ini berkomunikasi melalui skema *request-response*.

Pada tipe *peer-to-peer*, setiap komputer yang terhubung pada jaringan ini disebut sebagai *node* bedanya sistem ini dengan tipe yang lain adalah sistem ini tidak memiliki server yang tersentralisir sehingga setiap dari *node* ini adalah setara dan mereka melakukan penjadwalan maupun pembagian tugas bersamaan.

Pada tipe *three-tier*, sistem ini sebenarnya mirip dengan skema *client-server*, tetapi yang membedakannya adalah adanya satu komponen tambahan, yakni *data source* atau sumber data. Pada skema *three-tier*, arsitektur ini membagi tipe komputasinya menjadi tiga lapisan yakni *presentation*, *application*, dan *data*. Lapisan *presentation* digunakan sebagai skema untuk menunjukkan *user interface*, lapisan *application* adalah lapisan untuk memperolah informasi dan melakukan pemrosesan terhadap basis data, dan lapisan *data* adalah lapisan yang menampung data.

Tipe sistem terdistribusi yang terakhir adalah tipe *N-tier*. Tipe ini sebenarnya mirip dengan tipe arsitektur *three-tier*, tetapi jumlah lapisannya tidak hanya tiga tetapi bisa banyak.

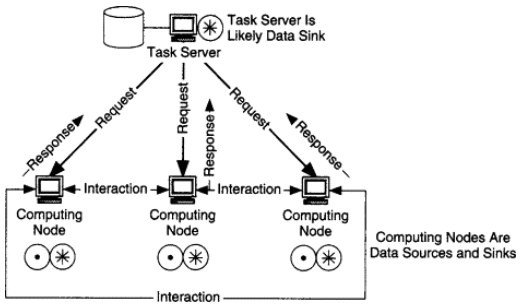


Fig. 2. *Skema sistem client-server*
Sumber: https://www.ridge.co/blog/what-is-distributed-computing/

D. Contoh dari Implementasi Sistem Terdistribusi

Sistem terdistribusi pada dewasa ini diaplikasikan pada banyak sistem. Beberapa aplikasinya adalah sebagai berikut.

1. Sistem *Real-time*
2. Pemrosesan parallel
3. *Artificial intelligence* terdistribusi
4. Basis data terdistribusi
5. Aplikasi web *multilevel*
6. *Multiplayer game*

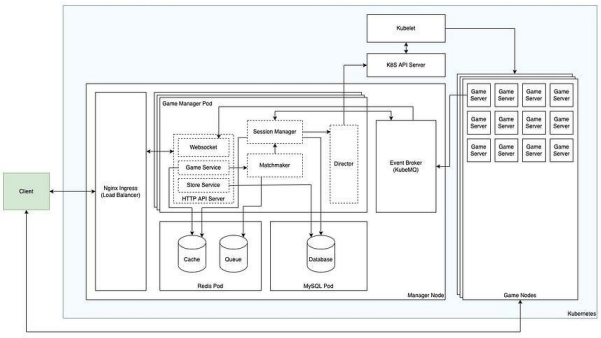


Fig. 3. *Skema sistem server MMORPG*
Sumber: https://theredrad.medium.com/designing-a-distributed-system-for-an-online-multiplayer-game-architecture-part-3-f9483ebbe5ac

II. PENJADWALAN TUGAS DALAM SISTEM TERDISTRIBUSI

Pada bagian I, telah disebutkan bahwa salah satu aspek dari sistem terdistribusi adalah *resource sharing*. Sumber daya yang dapat dipakai bersama ini harus dapat dibagi secara tepat dan efektif kepada setiap komputer yang terhubung supaya waktu penyelesaian tugas bisa semakin cepat. Oleh sebab itu, diperlukan suatu sistem dan algoritma yang disediakan oleh sistem utama untuk mengatur pendistribusian tugas secara tepat. Algoritma ini biasa sering disebut sebagai *scheduling algorithms*.

Ada banyak jenis algoritma penjadwalan yang pernah dilampirkan, misalnya seperti *First-Come, First-Served* (FCFS), *Shortest-Job-Next*, *Priority Scheduling*, *Shortest Remaining Time*, *Round-Robin Scheduling*, dan lain

sebagainya. Semua algoritma ini lalu dibagi menjadi dua macam, algoritma *non-preemptive* dan *preemptive*.

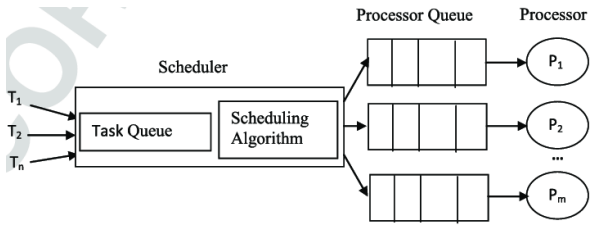


Fig. 4. *Ilustrasi Proses Penjadwalan Tugas pada Sistem Terdistribusi*
Sumber: https://www.researchgate.net/figure/Scheduling-model-for-distributed-systems_fig1_312650222

Perbedaan di antara keduanya adalah pada algoritma *non-preemptive* jika suatu tugas sedang dikerjakan, maka tugas ini harus diselesaikan hingga waktu yang dialokasikannya habis. Algoritma yang *preemptive* berbeda, tugas yang belum selesai bisa disalip dengan tugas lain yang memiliki tingkat prioritas yang lebih tinggi. Kedua pendekatan ini memiliki kelebihan dan kekurangannya sendiri-sendiri. Misalnya, algoritma *preemptive* dapat menyebabkan *starvation* di mana suatu tugas bisa saja ditidak dijalankan sebab selalu digantikan posisinya dengan tugas dengan prioritas yang lebih tinggi sehingga tugas yang tergantikan ini tidak akan pernah selesai dijalankan. Di sisi lain, algoritma *non-preemptive* dapat menyebabkan *response-time* yang buruk sebab suatu tugas harus diselesaikan dulu baru bisa tugas lain untuk dijalankan. Jika tugas yang ingin diselesaikan memakan waktu yang sangat lama untuk dikerjakan, maka tugas-tugas lain akan terbengkalai juga dan tidak dikerjakan juga akan menyebabkan *starvation*.

Ada beberapa karakteristik atau kriteria algoritma penjadwalan yang dapat disebut baik. Kriteria pertama adalah algoritma yang baik seharusnya tidak harus tahu pengetahuan sebelumnya tentang suatu tugas yang akan diberikan, misalnya berapa banyak sumber daya yang dibutuhkan atau waktu alokasinya. Lalu, kriteria selanjutnya adalah algoritma penjadwalan yang baik dapat berdaptasi terhadap perubahan secara dinamis. Misalnya, alokasi sebuah sumber daya dapat berubah ketika adanya perubahan yang disebabkan ketidakseimbangan *load* pada sistem terdistribusi. Selain itu, program harus dapat menentukan pilihan secara cepat. Oleh sebab itu, biasanya pendekatan algoritma heuristik sering dipakai untuk masalah penjadwalan tugas dalam kasus ini.

III. Algoritma Path-Planning A-Star dan Relevansinya

Ada beberapa algorithma *path-planning*, misalnya seperti DFS (*Depth-first search)*, BFS (*Breadth-first search*), UCS (*Uniformed-cost search*), *Greedy BFS*, dan Algoritma A-Star (A*). Masing-masing algoritma memiliki keunikannya masing-masing. Misalnya, pada algoritma BFS dan DFS, *path-planning* atau perancangan jalur dapat dilakukan secara menyeluruh menggunakan BFS tetapi memakan waktu yang sangat lama dan tidak efisien sebab setiap simpul pada graf ditelusuri. Pada DFS, kita dapat menemukan jalur dengan cepat tetapi, masalah dari algoritma ini adalah hasilnya yang selalu tidak optimal karena hasil pertama yang dicari ialah yang dipakai.

Algoritma-algoritma lain yang biasanya sering dipakai adalah algoritma UCS, algoritma Greedy BFS, dan algoritma A-Star. Masing-masing algoritma ini memiliki kelebihan dan kekurangannya tersendiri. Tetapi, hal yang menyatukan mereka semua adalah penggunaan struktur data *prioqueue*. Hal yang membedakan ketiganya adalah bagaimana penentuan traversal grafnya. Masing-masing dari ketiga algoritma ini menggunakan suatu fungsi f(n) yang terdefinisi unik pada masing-masing algoritma dan berdasarkan nilai f(n) kita dapat mengurutkan simpul mana yang harus ditelurusi oleh graf (pada *prioqueue*).

Algoritma UCS memiliki nilai f(n) = g(n) di mana g(n) adalah suatu fungsi yang menunjukkan jarak sejauh ini yang ditempuh dari simpul awal ke simpul n. Algoritma Greedy BFS menggunakan fungsi f(n) = h(n), di mana h(n) adalah suatu fungsi heuristik yang dapat ditentukan misalnya menggunakan jarak euclidean ataupun jarak Manhattan. Lalu, algoritma A-Star menggunakan nilai f(n) = g(n) + h(n), yakni nilai g(n) dan h(n) sesuai dengan definisi sebelumnya.

Algoritma A-Star bekerja dengan menggunakan traversal graf. Pertama-tama, harus didefinisikan terlebih dahulu simpul-simpul atau status yang akan dilewati, dari beberapa status tersebut minimal terdiri dari dua status utama yakni status awal dan status akhir. Kemudian, harus ditentukan fungsi f(n) yang terdiri dari h(n) dan g(n), seperti yang telah didefinisikan sebelumnya. Traversal graf akan dimulai melalui simpul pertama, kemudian daftarkan simpul-simpul yang bersisian dengan simpul awal dan untuk setiap simpul tentukan nilai f(n)nya dan masukkan setiap simpul ini ke dalam larik simpul hidup dan masukkan simpul awal sebagai simpul ekspan. Urutkan nilai pada larik simpul hidup sesuai dengan nilai f(n) dimulai dengan nilai f(n) terkecil. Selanjutnya, lakukan proses ini hingga salah satu hasil dari simpul ekspan adalah simpul tujuan atau simpul akhir.

Algorithm	Complexity	Reso.	Costs	Time (ms)
A* algorithm	$\mathcal{O}(\|E\|)$	640x480	611	608 ($\pm$ 18.2)
	$\mathcal{O}(\|E\|)$	320x240	305	58 ($\pm$ 1.7)
Wavefront	$\mathcal{O}(\|E\| + \|V\|)$	640x480	611	1778 ($\pm$ 53.3)
	$\mathcal{O}(\|E\| + \|V\|)$	320x240	305	199 ($\pm$ 6.0)
BFS	$\mathcal{O}(\|E\| + \|V\|)$	640x480	611	1943 ($\pm$ 58.3)
	$\mathcal{O}(\|E\| + \|V\|)$	320x240	305	212 ($\pm$ 6.4)
Dijkstra's	$\mathcal{O}(\|E\| + \|V\|log\|V\|)$	640x480	611	4875 ($\pm$ 146.3)
	$\mathcal{O}(\|E\| + \|V\|log\|V\|)$	320x240	305	489 ($\pm$ 14.7)

Fig. 5. *Perbandingan algoritma-algoritma perencanaan jalur*
Sumber: https://www.researchgate.net/figure/Comparison-of-path-finding-algorithms_tbl1_308034798

Seperti yang telah dijelaskan sebelumnya, algoritma A-Star adalah salah satu algoritma dalam perencanaan jalur yang memanfaatkan penggunaan fungsi heuristik yang selalu melakukan estimasi yang *underestimate* dan telah dibuktikan dapat menemukan solusi yang optimal. Hal ini tentunya dinilai cocok untuk masalah penjadwalan tugas pada sistem terdistribusi seperti yang dijelaskan pada bagian III. Namun, tentunya masalah penjadwalan tidak bisa langsung diselesaikan begitu saja dengan algoritma A-Star, tetapi harus dilakukan

suatu pemetaan atau konversi masalah dari masalah penjadwalan menjadi suatu masalah *path-planning*.

IV. PEMETAAN PENJADWALAN TUGAS PADA ALGORITMA PATH-PLANNING

Pertama-tama, akan didefinisikan metode yang akan digunakan untuk melakukan pemetaan masalah penjadwalan tugas menjadi masalah yang dapat diselesaikan menggunakan algoritma A-Star. Berikut adalah langkah-langkahnya:

1. Definisikan ruang statusnya
2. Tentukan aksi-aksi atau transisi yang dapat ditentukan dari suatu *state* ke *state* lain.
3. Tentukan fungsi *cost*nya g(n)
4. Tentukan fungsi heuristiknya h(n)
5. Gunakan algoritma A* untuk menyelesaikan masalahnya

Dari langkah-langkah yang disebutkan di atas, ada beberapa hal yang perlu diperhatikan. Hal pertama adalah tentang pendefinisian aksinya. Dalam mendefinisikan aksi, kita dapat menentukan aksi-aksi ini misalnya dalam bentuk pemberian tugas kepada sumber daya yang ada atau perpindahan sumber daya. Untuk pendefinisian fungsi *cost*, kita dapat menggunakan nilai-nilai seperti *execution time* dan metrik-metrik lainnya yang memengaruhi perpindahan dari suatu state ke state lain. Lalu, untuk fungsi heuristiknya, harus didefinisikan sedemikian sehingga estimasi yang dilakukan harus *underestimate*. Salah satu caranya adalah dengan melakukan estimasi dengan bentuk *remaining cost* untuk menyelesaikan semua tugas yang ada dari state yang diberikan. Selain itu, status awal adalah ketika belum ada tugas yang diselesaikan dan status terakhir dalam kasus ini dapat didefinisikan sebagai semua tugas telah selesai dilaksanakan.

Berikut akan dijelaskan salah satu contoh pemetaan masalah ini. Diberikan sebuah data sebagai berikut:

Diberikan sebuah tugas yang memiliki beberapa submodul yakni Task = {m_{11}, m_{21}, m_{31}} dan diberikan tiga buah komputer dalam sistem terdistribusi Processor = {P_0, P_1, P_2}. Berikut adalah ilustrasi graf yang sesuai dengan data ini.

Tahap 1 dan 2: Definisikan ruang statusnya, tentukan aksi atau transisi yang dpaat ditentukan dari satu status ke status lain. Dalam hal ini akan dibuat sebuah ruang status dalam bentuk $X_1X_2X_3$, masing-masing dari nilai X_i adalah modul ke-*i* nilainya diisi dengan proses apa yang mengerjakannya. Contohnya, modul m_{11} akan ditugaskan ke prosessor P_1 maka notasinya akan menjadi seperti ini $P_1X_2X_3$.

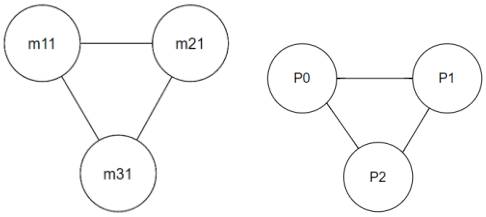


Fig. 6. *Graf komunikasi antar modul (IMC) dan prosessor*
Sumber: Dokumen pribadi

Tahap 3 dan 4: Tentukan fungsi *cost* dan heuristiknya. Dalam hal ini dapat dihitung fungsi *cost*nya adalah besarnya biaya penugasan modul ke prosessor. Lalu, fungsi heuristiknya adalah besarnya biaya komunikasi antar modul (ambil nilai minimum dari modul yang belum ditugaskan)

TABLE I. BIAYA PENUGASAN MODUL KE PROSESSOR G(N)

	P_0	P_1	P_2
M_{11}	17	13	11
M_{21}	16	14	10
M_{31}	18	15	8

TABLE II. BIAYA KOMUNIKASI ANTAR MODUL H(N)

	M_{11}	M_{21}	M_{31}
M_{11}	0	9	5
M_{21}	9	0	7
M_{31}	5	7	0

Tahap 5: Gunakan algoritma A* untuk menyelesaikan masalahnya.

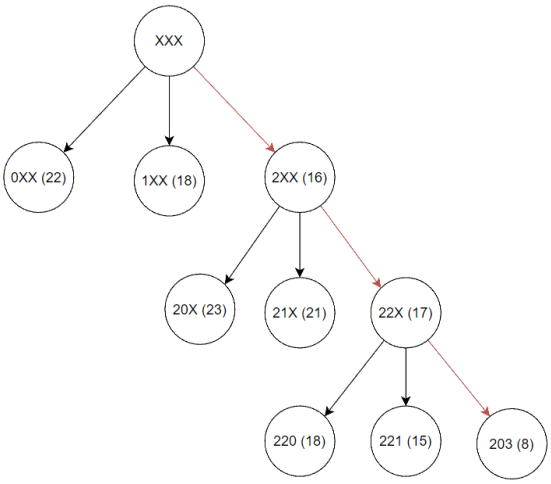


Fig. 7. *Graf ruang status dan penentuan pilihan*
Sumber: Dokumen pribadi

TABLE III. PERHITUNGAN

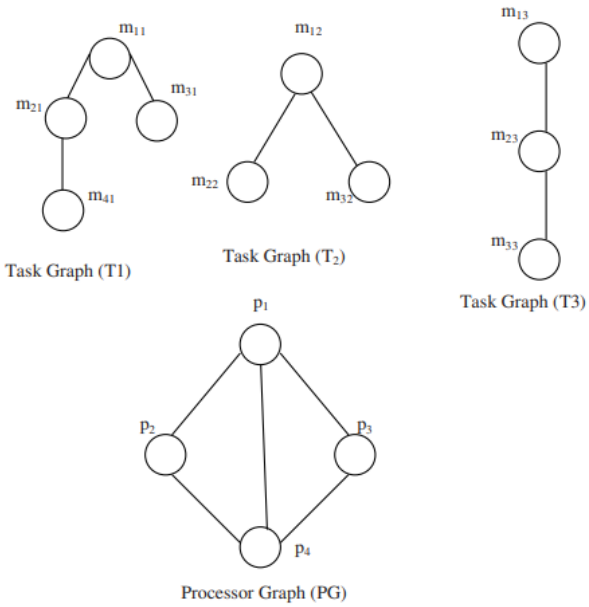
Iterasi	Simpul Ekspan	Simpul Hidup
1	$X_1X_2X_3$	$P_2X_2X_3$ (11+5 = 16), $P_1X_2X_3$ (13+5 = 18), $P_0X_2X_3$ (17+5 = 22)
2	$P_2X_2X_3$ (16)	$P_2P_2X_3$ (10 + 7 = 17), $P_1X_2X_3$ (18), $P_2P_1X_3$ (14 + 7 = 21), $P_0X_2X_3$ (22), $P_2P_0X_3$ (16 + 7 = 23),
3	$P_2P_2X_3$ (17)	$P_2P_2P_2$ (8), $P_2P_2P_1$ (15), $P_2P_2P_0$ (18), $P_1X_2X_3$ (18), $P_0X_2X_3$ (22), $P_2P_0X_3$ (16 + 7 = 23), $P_2P_1X_3$ (14 + 7 = 21),
4	$P_2P_2P_2$ (Solusi ditemukan)	

V. IMPLEMENTASI PROTOTIPE PENJADWALAN TUGAS SISTEM MENGGUNAKAN ALGORITMA A-STAR

Berikut adalah kodenya dalam pseudo-code

```
function taskScheduling(initialAssignment):
    PrioQueue simpulHidup = new PrioQueue
    add initialAssignment with cost 0 to simpulHidup
    while openList is not empty:
        current ← remove the node with the lowest cost from simpulHidup
        currentAssignment ← current.assignment
        currentCost ← current.cost

        if currentAssignment is the goal state:
            return currentAssignment and currentCost

        for each unassigned module i in currentAssignment:
            for each processor j:
                if module i can be assigned to processor j:
                    assignmentCost ← assignment cost for module i and processor j
                    communicationCost ← communication cost between module i and assigned modules on processor j
                    newCost ← currentCost + assignmentCost + communicationCost
                    newAssignment ← replace module i in currentAssignment with processor j
                    add newAssignment with cost newCost to simpulHidup
        return "Tidak ada assignment yang valid"

    initialAssignment ← "XXXXX"
    optimalAssignment, optimalCost <- taskScheduling(initialAssignment)

print "Optimal Assignment:", optimalAssignment
print "Optimal Cost:", optimalCost
```

Algoritma ini tentunya dapat dikembangkan lebih jauh dengan memanfaatkan beberapa tugas sekaligus. Dalam contoh yang disebutkan di atas, tugas yang diberikan adalah satu dengan submodul tugas sebanyak 3 dan jumlah prosessor sebanyak 3. Banyaknya tugas nanti juga harus diseimbangkan dengan pembuatan beberapa tabel baru serta tabel penggunaan memori yang juga haris diperhatikan, misalnya seperti ini.



m_{11}	m_{21}	m_{31}	m_{41}	m_{12}	m_{22}	m_{32}	m_{13}	m_{23}	m_{33}
5	3	2	4	3	2	1	4	2	3

Memory Requirement of Modules in Units

Processor	Max. no. of Modules	Memory Capacity	Modules Assigned	Remaining No. of Modules	Remaining Memory
p_1	4	10	m_{21} m_{41}	4	3
p_2	3	8		3	8
p_3	4	9		4	9
p_4	5	12	m_{11} m_{31}	3	5

Fig. 8. *Graf kasus 3 task dan 4 processor dan hasilnya*
Sumber: Referensi [10]

VI. KESIMPULAN

Masalah penjadwalan tugas pada sebuah sistem terdistribusi dapat diselesaikan menggunakan algoritma perencanaan jalur menggunakan algoritma A-Star karena sifatnya yang menggunakan fungsi heuristik menghasilkan suatu algoritma yang cepat dan efisien dalam menyelesaikan masalah penjadwalan ini. Hal ini dicapai dengan memematakan masalah tersebut menjadi sebuah masalah graf. Bagian yang menantang dalam permasalahan ini adalah proses bagaimana sebuah masalah penjadwalan bisa dipetakan menjadi sebuah masalah perencanaan *path-finding*. Walaupun demikian, jika solusi yang ditemukan optimal masih banyak hal yang harus dikerjakan, yakni bagaimana rencana pendistribusian tersebut disitribusikan secara simultan ke semua sumber daya atau komputer terhubung serta bagaimana sistem dapat merespon kepada perubahan dalam jumlah besar.

UCAPAN TERIMA KASIH

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena berkat dan karuniaNya sehingga penulis dapat menyelesaikan makalah dengan judul "Pemodelan dan Penjadwalan Tugas dalam Sistem Terdistribusi dengan Pendekatan Algoritma Path-Planning A-Star" dengan tepat waktu. Selain itu, penulis ingin berterima kasih kepada dosen-dosen pengampu mata kuliah IF2210 – Strategi Algoritma, yakni pak Rinaldi Munir, bu Nur Ulfa Maulidevi, dan pak Rila Mandala yang telah mengajarkan materi kuliah dengan baik.

REFERENSI

[1] https://www.geeksforgeeks.org/starvation-and-aging-in-operating-systems/. Diakses pada tanggal 21 Mei 2022.

[2] https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm. Diakses pada tanggal 21 Mei 2022.

[3] https://www.geeksforgeeks.org/scheduling-and-load-balancing-in-distributed-system/. Diakses pada tanggal 21 Mei 2022.

[4] https://www.geeksforgeeks.org/scheduling-and-load-balancing-in-distributed-system/ . Diakses pada tanggal 21 Mei 2022.

[5] https://www.confluent.io/learn/distributed-systems/. Diakses pada tanggal 21 Mei 2022.

[6] https://www.comp.nus.edu.sg/~bleong/hydra/related/assiotis06mmorpg.pdf. Diakses pada tanggal 21 Mei 2022.

[7] https://theredrad.medium.com/designing-a-distributed-system-for-an-online-multiplayer-game-architecture-part-3-f9483ebbe5ac. Diakses pada tanggal 21 Mei 2022.

[8] https://www.javatpoint.com/advantages-and-disadvantages-of-distributed-system. Diakses pada tanggal 21 Mei 2022.

[9] https://www.cis.upenn.edu/~lee/07cis505/Lec/lec-ch1-DistSys-v4.pdf. Diakses pada tanggal 21 Mei 2022.

[10] Scheduling in Distributed Computing Systems Analysis, Design & Models. Deo Prakash Vidyarthi, Biplab Kumer Sarker, Anil Kumar Tripathi, Laurence Tianruo Yang. ISBN: 978-0-387-74480-3.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2022

Nicholas Liem - 13521135