JURNAL MATEMATIKA DAN KOMPUTER
Vol. 6. No. 3, 137 - 143, Desember 2003, ISSN : 1410-8518

PERANCANGAN ARSITEKTUR PEMARALELAN UNTUK MENCARI SHORTEST PATH DENGAN ALGORITMA DIJKSTRA

Eko Adi Sarwoko
Jurusan Matematika FMIPA UNDIP

Abstrak

Perancangan arsitektur pemaralelan merupakan salah satu tahap penting dalam komputasi paralel. Tahap ini bertujuan agar kompleksitas komputasi dan komunikasi dapat efisien. Tulisan ini merupakan kajian perancangan arsitektur pemaralelan mencari Shortest Path dengan Algoritma Dijkstra. Rancangan ini ditinjau berdasarkan aspek analisis algoritma baik kompleksitas komputasi maupun komunikasi.

1. PENDAHULUAN

Solusi untuk mencari Shortest Path dengan Algoritma Dijkstra secara sekuensial telah banyak diteliti para pakar [McHugh, 1990]. Seperti diketahui aspek programming sekuensial mengalami kendala yang mencakup keterbatasan tranfer data, dan keterbatasan kecepatan perhitungan [Lewis, 1992]. Dengan perkembangan teknologi hardware dan software saat ini, alternatif yang saat ini dikembangkan adalah penyelesaian masalah dengan pendekatan proses secara paralel. Secara umum diharapkan dapat meningkatkan kinerja dan efisiensi dalam menangani suatu masalah. Apalagi dipihak user juga menginginkan adanya sistem penyelesaian masalah yang cepat dan dapat mengatasi masalah yang jauh lebih besar dan komplikated [Lewis, 1992],[Kumar,1994].

Demikian pula untuk solusi masalah shortest path dengan Algoritma Dijkstra memerlukan penyelesaian masalah dengan pendekatan proses secara paralel, apabila pendekatan solusi secara sekuensial belum mampu memberikan solusi yang cepat serta dihadapkan dengan jumlah vertek yang jauh lebih besar dan komplikated.

Dalam pemrograman paralel yang melibatkan banyak prosesor, selanjutnya beban masalah didistribusikan ke berbagai prosesor. Dengan melibatkan banyak prosesor hal ini akan berdampak pada aspek komunikasi. Isue penting yang harus diperhatikan adalah proses komunikasi tetap *low-overhead*.

Isue tersebut akan berdampak pada berbagai hal, antara lain mencakup pengaturan dan sinkronisasi arsitektur komputernya, proses komunikasi dan transfer data, dan metode keparalelan [Lewis, 1992],[Kumar,1994].

Tulisan ini mengkaji desain arsitektur keparalelan untuk masalah shortest path dengan Algoritma Dijkstra, agar diperoleh efisiensi dan peningkatan speedup dibandingkan dengan cara sekuensial.

2. KOMPUTASI PARALEL UNTUK SHORTEST PATH DENGAN ALGORITMA DIJKSTRA

Pandang graph berarah dengan bobot non negatip G=(V,E), masalah path terpendek dengan single-source adalah untuk mencari path terpendek dari suatu vertek $v \in V$ ke semua vertek yang lain di dalam V. Suatu shortest path dari u ke v adalah path dengan bobot minimal. Selain mencari shortest path dari vertek tunggal v ke setiap vertek yang lain, dapat juga untuk mencari shortest path diantara semua pasangan titik. Secara formal, masalah shortest path setiap pasang adalah untuk mencari shortest path diantara semua pasang vertek $v_I, v_j \in V$ sedemikian sehingga $i \neq j$. Untuk suatu graph dengan n vertek, outputnya adalah matriks berukuran nxn dari D=d(I,j) sedemikian sehingga $d_{I,j}$ adalah biaya shortest path dari vertek v_I ke vertek v_j.

Bobot garis dapat merepresentasikan waktu, biaya, pinalti, kerugian, atau beberapa kuantitas lain secara akumulatif.

Algoritma Dijkstra untuk mencari single-source shortest path dari suatu vertek tunggal s, dilakukan secara increment mencari shortest path dari s ke vertek yang lain di G dan selalu memilih suatu edge ke suatu vertek tertutup yang terdekat, dengan kompleksitas $\boldsymbol{\theta(n^2)}$**.** Sedang Algoritma pencarian all-pairs shortest path dari suatu vertek ke semua vertek yang lain, untuk semua pasangan

dengan mengeksekusi algoritma single-source pada setiap prosesor, untuk semua vertek v. Algoritma ini memerlukan kompleksitas $\theta(n^3)$.

Segmen program berikut menunjukkan Algoritma Dijkstra Sekuensial Untuk Shortest Paths Single Source [Brassard,1996]. Pada prosedur ini untuk setiap vertek $u \in (V-V_T)$, meletakkan l[u], sebagai biaya minimal untuk menjangkau vertek u dari vertek s dimana vertek-vertek berada di V_T.

```
1.  Procedure DIJKSTRA-SINGLE-SOURCE-SP(V,E,w,s)
2.  Begin
3.      V_T={s};
4.      For all v∈(V-V_T) do
5.          If(s,v) exists set l[v]=w(s,v);
6.          Else set l[v]=∞;
7.      While V_T≠V do
8.      Begin
9.          Find a vertex u such that l[v] = min { l[v] | v∈(V-V_T) };
10.         V_T=V_T∪{u};
11.         For all v∈(V-V_T) do
12.                 L[v] = min { l[v], l[u] + w(u,v) };
13.     Endwhile
14. End DIJKSTRA-SINGLE-SOURCE-SP
```

3. ARSITEKTUR KOMPUTASI PARALEL UNTUK SHORTEST PATH DENGAN ALGORITMA DIJKSTRA

Menurut [Kumar,1994], model arsitektur yang dipilih untuk implementasi paralelism harus disesuaikan dengan prosesor dan hardware, agar tercipta efisiensi proses. Hal ini harus diperhitungkan, karena bukan tidak mungkin masalah komunikasi ini, akan jauh lebih kompleks daripada masalah arsitektur, dan ini sering terabaikan pada perhitungan kinerja. Kemudian dari aspek software (sistem operasi, kompilator) dapat dilakukan secara dinamis (sistem mendeteksi sendiri) atau statis (pemrogram yang mesti menentukan letak keparalelannya) [Chaudhuri, 1992].

Selanjutnya untuk memperoleh hasil yang optimal selain desain algoritma paralel yang tepat, juga harus memperhatikan biaya komunikasi, karena terkadang kompleksitas komunikasi lebih tinggi dibandingkan kompleksitas komputasi, atau waktu yang dipakai untuk mengatur data antar prosesor lebih tinggi dibandingkan waktu untuk proses manipulasi data [Quinn, 1987]. Selain itu juga perlu

diperhatikan arsitektur komputer, ini penting untuk proses sinkronisasi antar prosesor dan pemrosesan.

3.1. Arsitektur Paralel untuk Single-Source Shortest Paths dengan Algoritma Dijkstra

Formulasi paralel untuk masalah ini prinsipnya adalah iterasi. Masing-masing iterasi mencari suatu vertek dengan pencapaian minimal dari suatu vertek asal, diantara vertek-vertek yang belum dikunjungi yang terhubung secara langsung dengan suatu vertek yang sudah dukunjungi. Pencapaian ini memungkinkan untuk memilih lebih dari satu vertek, jika terdapat lebih dari dua pilihan yang sudah dikunjungi berhungan langsung dengan vertek yang belum dikunjungi maka dipilih yang jaraknya paling dekat. Matriks adjacency berbobot dipartisi dengan menggunakan *block-striped mapping*.

Sehingga arsitektur masing-masing p prosesor ditugaskan secara berurut n/p kolom dari matriks matriks adjacent berbobot, dan menghitung nilai n/p pada array l.

3.2. Arsitektur Paralel All-pairs Shortest Path dengan Algoritma Dijkstra

Perancangan arsitektur untuk masalah all-pairs shortest path Dijkstra dapat diparalelisasikan dengan dua cara yang berbeda.

a. Source-Partitoned Formulation

Formulasi paralel source partitioned dari algoritma Dijkstra menggunakan n prosesor. Masing-masing p_I mencari shortest shortest path dari vertek v_I ke semua vertek yang lain dengan mengeksekusi Algoritma Dijkstra yang sekuensial dari single-source shortest path. Dengan demikian tidak ada proses komunikasi antar prosesor.

Sehingga, eksekusi paralel untuk formulasi ini adalah $\mathbf{T_p = \theta(n^2)}$. Komunikasi prosesor seperti tidak ada, ini merupakan formulasi paralel yang ekselen. Namun, ini tidaklah benar, karena algoritma menggunakan sebanyak n prosesor. Selanjutnya, fungsi isoefisiensi untuk proses konkurensinya adalah

$\theta(p^3)$, dimana ini merupakan rata-rata fungsi isoefisiensi algoritma ini. Jika jumlah prosesor yang tersedia untuk menyelesaikan masalah ini adalah kecil (bahwa $n=\theta(p)$), maka algoritma ini mempunyai kinerja yang baik. Namun, jika jumlah prosesor lebih besar dari n, algoritma lain biasanya mengadopsi algoritma ini sebab skalabilitasnya sangatlah kecil.

b. Source-Parallel Formulation

Masalah utama dengan formulasi source-partitioned formulation adalah bahwa jika hanya menggunakan n prosesor akan terjadi *busy* saat bekerja. Source parallel formulation adalah sama dengan source partitioned formulation, bedanya bahwa algoritma single-source berjalan pada subset prosesor yang terpisah.

Secara khusus, p(=16) prosesor dibagi menjadi n(=4) partisi, masing-masing dengan p/n prosesor (formulasi ini ditekankan hanya jika p>n). Masing-masing n problem single-source shortest path diselesaikan dengan satu dari n partisi. Dengan kata lain, pertama pemarelan problem all-pairs shortest path dengan menugaskan setiap vertek ke sebagian dari kumpulan prosesor, dan pemaralelan algoritma single-source dengan menggunakan p/n prosesor untuk menyelsaikannya. Jumlah total prosesor yang digunakan efisien dengan formulasi ini yaitu **$\theta(n^2)$.**

3.3. Evaluasi Overhead Komputasi dan Komunikasi

Asumsikan bahwa arsitektur yang dibangun memiliki p prosesor dengan struktur mesh, sedemikian sehingga $\sqrt{p}$ adalah perkalian pada $\sqrt{n}$. struktur mesh $\sqrt{p}x\sqrt{p}$ dipartisi menjadi n submesh yang masing-masing berukuran $\sqrt{(p/n)}x\sqrt{(p/n)}$.

Selanjutnya algoritma single source ini dieksekusi pada setiap submesh, maka waktu eksekusi paralel adalah **$T_p=\theta(n^3/p)+ \theta(\sqrt{(np)})$**, dimana waktu komputasinya **$\theta(n^3/p)$** dan waktu komunikasinya **$\theta(\sqrt{(np)})$.** Sedang waktu eksekusi sekuensial adalah **$W=\theta(n^3)$,** maka Speedup dan Efisiensinya adalah **$\theta(n^3)$ / { $\theta(n^3/p)+ \theta(\sqrt{(np)})$ },** dan **1 / { 1 + $\theta(p^{1.5}/n^{2.5})$ }.**

Dari hasil tersebut terlihat bahwa formulasi biaya minimal adalah $\mathbf{p^{1.5}/n^{2.5}=\theta(1).}$ Selanjutnya, formulasi ini dapat digunakan untuk menjadi $\mathbf{\theta(n^{1.66})}$ prosesor yang efisien. Hal ini menunjukkan terjadi isoefisiensi untuk komunikasi adalah $\mathbf{\theta(p^{1.8}).}$

Sedangkan untuk arsitektur formulasi paralel Dijkstra untuk semua pasangan, terlihat bahwa dalam formulasi source partitioned tidak ada komunikasi, dengan menggunakan prosesor yang jumlahnya tidak lebih dari **n prosesor**, dan menyelesaikan masalah dalam waktu $\mathbf{\theta(n^2).}$ Sangat bertentangan, pada formulasi source parallel menggunakan sampai $\mathbf{n^{1,66}}$ prosesor, memiliki waktu (overhead) komunikasi, dan menyelesaikan problem dalam waktu $\mathbf{\theta(n^{1.33})}$ bila digunakan prosesor sebanyak $n^{1,66}$. Sehingga, source parallel formulation lebih mengeksploitasi keparalelan dibandingkan source-partitioned.

4. KESIMPULAN

Arsitektur untuk Algoritma Dijkstra Single-Source Shortest Paths ini memerlukan masing-masing p prosesor ditugaskan secara berurut n/p kolom dari matriks matriks adjacent berbobot, dan menghitung nilai n/p pada array l. Pada algoritma single Source-Parallel Formulation dieksekusi pada arsitektur dimana setiap submesh, maka waktu eksekusi paralel adalah jumlahan waktu komputasinya $\mathbf{\theta(n^3/p)}$ dengan waktu komunikasinya $\mathbf{\theta(\sqrt{(np)}),}$ yaitu $\mathbf{T_p=\theta(n^3/p)+\theta(\sqrt{(np)})}$. Ini juga menunjukkan fungsi isoefisiensi untuk komunikasi adalah $\mathbf{\theta(p^{1.8}),}$ dengan isoefisiensi untuk proses yang konkuren adalah $\mathbf{\theta(p^{1.5}).}$

Speedup untuk arsitektur model Source-Partitoned Formulation adalah $\mathbf{\theta(n^3)/\theta(n^2)}$ dan efisiensinya adalah $\mathbf{\theta(1),}$ dimana tidak ada overhead komunikasi. Hal ini bukan merupakan formulasi paralel yang ekselen, karena bila menggunakan n prosesor, diperoleh fungsi isoefisiensi untuk proses konkurensi sebesar $\mathbf{\theta(p^3)}$.

Pada dua model arsitektur paralel untuk semua pasangan, untuk formulasi source partitioned tidak ada komunikasi, bila menggunakan prosesor yang jumlahnya tidak lebih dari n prosesor, dan menyelesaikan masalah dalam waktu

$\theta(n^2)$. Sedangkan pada formulasi source parallel menggunakan sampai $\mathbf{n^{1,66}}$ prosesor, memiliki waktu (overhead) komunikasi, dan menyelesaikan problem dalam waktu $\mathbf{\theta(n^{1.33})}$ bila digunakan prosesor sebanyak $\mathbf{n^{1,66}}$**.**

DAFTAR PUSTAKA

1. Akl, Selim G., ***The Design and Analysis of Parallel Algorithm***, Prentice-Hall International, Inc.,London, 1989
2. Brassard, G., dan Bratley, P., ***Fundamentals of Algorithmics***, Prentice Hall International, Inc., Singapore, 1996
3. Chaudhuri, P., ***Parallel Algorithms : Design and Analysis***, Prentice Hall Advances in Computer Science Series, 1992
4. Kumar, V., Grama, A., Gupta, A., dan Karypis, G., ***Introduction to Parallel Computing : Design and Analysis of Algorithms***, The Benjamin/Cummings Publishing Company, Inc., California, 1994
5. Lewis, Ted G., dan El-Rewini, H., ***Introduction to Parallel Computing***, Prentice Hall, Inc., Englewood Cliffs, NJ, 1992
6. McHugh, JA., ***Algorithmic Graph Theory***, Prentice Hall International, Inc., Englewood Cliffs, NJ., 1990
7. Quinn, NJ., ***Designing Efficient Algorithms for Parallel Computers***, Mc Graw Hill, International Editions, Singapore, 1987