PEMODELAN ALGORITMA GAXPY PADA SISTEM MULTIPROSESOR

Tri Prabawa[1)]
Program Studi Teknik Informatika, STMIK AKAKOM Yogyakarta
Jl. Raya Janti 143, Yogyakarta 55198
e-mail : *tprabawa@akakom.ac.id*

ABSTRAK

Dalam komputasi matriks sering dijumpai operasi GAXPY yang dapat dinyatakan sebagai Y = Y + AX, dengan X, Y $\in R^n$ *dan A* $\in R^{nxn}$*. Dalam tulisan ini dibahas perancangan algoritma GAXPY untuk sistem multiprosesor. Beberapa pendekatan yang dapat dipakai untuk membuat algoritma pada sistem multiprosesor antara lain dengan cara mengeksploitasi sifat-sifat paralelisasi dari algoritma sekuensial, menggunakan algoritma paralel yang sudah ada untuk menggantikan beberapa instruksi pada algoritma sekuensialnya, atau dengan cara membuat algoritma paralel yang baru yang lebih sesuai dengan persoalan yang dihadapi.*

Sedangkan cara mengeksploitasi letak paralelisasi dari masalah yang ada, dipakai metode dekomposisi algoritmik atau dekomposisi geometrik. Untuk memperoleh ukuran kinerja yang baik pada sistem multiprosesor perlu diperhatikan beberapa hal, antara lain topologi jaringan, pembagian beban kerja prosesor (load balancing), serta partisi data untuk mendukung proses dekomposisi yang optimal.

Akhirnya untuk permodelan algoritma GAXPY digunakan pendekatan dengan membuat algoritma paralel yang baru yang lebih sesuai dengan persoalan yang dihadapi, memakai prinsip pipeline, memperhatikan pembagian beban kerja yang sama dan partisi data secara kolomial.

***Kata Kunci**: algoritma sekuensial, algoritma sistem multiprosesor, komputasi matriks, metode GAXPY*

ABSTRACT

Gaxpy method is one of the operations in a computing matrix that can be expressed as Y = Y + AX, with X, Y $\in R^n$ *and A* $\in R^{nxn}$*. In this paper discussed the design of algorithms GAXPY for multiprocessor systems. Several approaches can be used to create an algorithm on a multiprocessor system among others by exploiting the properties of parallelization of sequential algorithm, using parallel algorithms that already exist to replace some instructions on algorithms sekuensialnya, or by creating a parallel algorithm that is more suitable to problems encountered.*

How to exploiting the parallelization of the problem lies there can be used an algorithmic decomposition method or geometric decomposition. To obtain a good measure of performance on a multiprocessor system to note a few things, including network topology, load sharing of the processor (load balancing), and partition data to support optimal decomposition process.

Finally, to the design of algorithms used GAXPY approach by creating a new parallel algorithms that is more in keeping with the problems faced, the principle of a pipeline, pay attention to the division of the same workload and partition data by column.

***Keywords**: sequential algorithms, multiprocessor systems algorithms, matrix computation, GAXPY method.*

I. PENDAHULUAN

Dalam komputasi numerik, khususnya yang melibatkan operasi-operasi matriks, banyak ditemukan operasi dasar yang berbentuk Y = Y + AX. Operasi ini sering disebut sebagai operasi gaxpy, atau *general a x, plus y* [6]. Untuk matriks A yang berukuran atau berdimensi besar, maka diperlukan proses komputasi yang cepat. Penyelesaian yang diinginkan dapat dikerjakan secara cepat, jika dikerjakan dengan kontribusi komputer dan jika memungkinkan diproses secara paralel (memakai banyak prosesor).

Komputer paralel adalah suatu perangkat komputer yang mempunyai sejumlah alat pemroses (disebut prosesor) yang saling bekerja sama dalam suatu koordinasi program kendali [1]. Adanya arsitektur seperti ini memungkinkan suatu masalah diselesaikan secara paralel. Menurut taksonomi Flynn, model komputer yang sering dipertimbangkan sebagai sistem multiprosesor didefinisikan sebagai SIMD (*single instruction multiple data*) dan MIMD (*multiple instruction multiple data*).

Dalam menggunakan arsitektur komputer yang demikian maka kecepatan algoritma sangat ditentukan oleh jumlah prosesor yang dipakai serta pola hubungan interkoneksi antara prosesor yang satu dengan yang lain. Untuk itu perlu dibuat suatu algoritma yang sesuai untuk menyelesaikan masalah tersebut. Ada beberapa pendekatan yang dapat dipakai untuk membuat algritma pada sistem multiprosesor antara lain (1) mengeksploitasi sifat-sifat paralelisasi dari algoritma sekuensial yang ada,

yaitu mengidentifikasi instruksi-instruksi yang mungkin bisa dilakukan secara bersamaan tetapi dapat dikerjakan oleh prosesor berbeda, (2) menggunakan algoritma paralel yang sudah ada untuk menggantikan beberapa instruksi pada algoritma sekuensialnya, dan (3) membuat algoritma paralel baru yang lebih sesuai dengan persoalan yang dihadapi [10].

II. METODE PENELITIAN

Dalam tulisan ini metode penelitian berupa studi pustaka yang meliputi beberapa langkah, antara lain (1) melakukan formulasi persoalan GAXPY, (2) merumuskan algoritma sekuensial GAXPY, dan (3) merancang algoritma GAXPY untuk sistem multiprosesor.

2.1 Formulasi Operasi GAXPY.

Permasalahannya bersumber pada operasi GAXPY yang dituliskan sebagai vektor X,Y, dan matriks A. yang dinyatakan sebagai Y = Y + AX, dimana X, Y $\in R^n$ dan A $\in R^{nxn}$. dengan notasi sebagai berikut.

$$\begin{bmatrix} y1 \\ y2 \\ . \\ . \\ . \\ yn \end{bmatrix} = \begin{bmatrix} y1 \\ y2 \\ . \\ . \\ . \\ yn \end{bmatrix} + \begin{bmatrix} a11 & a21 & \cdots & an1 \\ a12 & a22 & \cdots & an2 \\ . & . & \cdots & . \\ . & . & \cdots & . \\ . & . & \cdots & . \\ a1n & a2n & \cdots & ann \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ . \\ . \\ . \\ xn \end{bmatrix} \quad (1)$$

atau dapat ditulis :

$$y_i = y_i + \sum_{j=1}^{n} a_{ij}\ x_i,\ \ dengan\ \ 1 \le i \le n \quad (2)$$

2.2 Algoritma GAXPY Sekuensial

Menurut Dompierre (2010) ada dua algoritma GAXPY yang dapat dikembangkan, yaitu (1) berbasis baris atau *row version*, dan (2) berbasis kolom atau *column version* [3].

Algoritma Gaxpy (*Row Version*)

Algoritma ini secara sederhana digunakan untuk menghitung perkalian matriks-vektor y = y + Ax dengan cara memperbarui komponen satu demi satu.

$$y_i = y_i + \sum_{j=1}^{n} a_{ij}\ x_i,\ \ dengan\ \ 1 \le i \le n \quad (3)$$

(sebagai bentuk umum dari *saxpy operation*)

Algoritma Gaxpy: Row Version

```
{ jika  A ∈ R^nxn, x ∈ R^n, dan y ∈R^n,  maka algoritma ini update nilai y
dengan cara  y = y + Ax}.
  for i = 1 : n
     for j = 1 : n
         y(i) = A(i,j)x(j) + y(i)
     end
  end
```

Algoritma Gaxpy (*Column Version*)

Jika perkalian matriks-vektor Ax diasumsikan sebagai kombinasi linear dari kolom A, maka dapat di-peroleh Gaxpy versi kolom sebagai berikut.

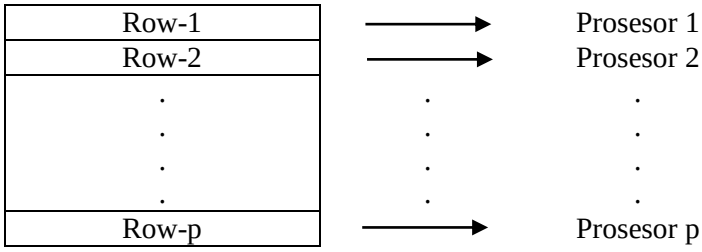
Algoritma Gaxpy : *Column Version*

```
{ jika  A ∈ R^nxn, x ∈ R^n, dan y ∈R^n,  maka algoritma ini update nilai y
dengan cara  y = y + Ax}
  for j = 1 : n,
    for i = 1 : n,
      y(i) = A(i,j)x(j) + y(i),
    end,
  end,
```

2.3 Metode Partisi Data

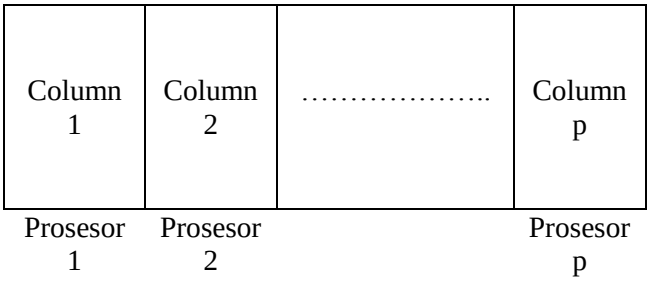
Partisi data dimaksudkan untuk mencari cara bagaimana data (beban) dipecah-pecah menjadi sub-data, disimpan kemudian dibagikan ke setiap prosesor yang bersesuaian. Misalkan jumlah prosesor yang digunakan sebanyak p, dimensi matriks A adalah n dan k = n/p untuk k bilangan bulat positif. Ada dua metode yang seringkali menjadi pertimbangan dalam partisi data.

a) *Metode Row Storage*, yaitu metode dimana prosesor ke i hanya menyimpan k baris dari submatriks, mulai baris ke (i-1)k + 1 sampai baris ke ik. Demikian juga vektor x, jika prosesor ke i akan menghitung komponen ke (i-1)k + 1 sampai ik dari Ax. Dengan mengamsumsikan bahwa Ax yang telah dihitung adalh prosesor tertentu yang akan digunakan oleh prosesor lain, maka tiap prosesor akan mengirimkan nilai yang telah diupdate ke prosesor lain. Masalah ini sering disebut sebagai *multimode broadcast problem*. Metode ini digambarkan sebagai berikut:



Gambar 1. Model *Row Storage* Untuk Partisi Data

b) *Metode Column Storage*, dimana prosesor ke i hanya menyimpan k kolom dari submatriks mulai kolom ke (i-1)k + 1 sampai kolom ke ik. Demikian juga vektor x. jadi prosesor ke i akan menghitung 1 komponen penjumlahan vector yang ke i. setelah itu hasilnya dikirim ke prosesor lain, dimana prosesor lain melakukan akumulasi terhadap vektor tadi. Masalah inilah yang disebut dengan *multimode accumulation problem*. Metode ini digambarkan sebagai berikut.



Gambar 2. Model *Column Storage* Untuk Partisi Data

2.4 Faktor-faktor Pendukung Desain Algoritma GAXPY Pada Sistem Multiprosesor

Dalam pemecahan menggunakan sistem multiprosesor/komputasi parallel, langkah awalnya mengeksploitasi letak paralelisasi dari masalah. Setelah itu melakukan dekomposisi terhadap masalah menjadi submasalah yang dapat diselesaikan secara simultan. Proses ini disebut dengan proses dekompisi, yang meliputi (1) dekomposisi algoritmik, yaitu melakukan dekomposisi algoritma sekuensial yang ada atas beberapa blok instruksi, dimana tiap blok instruksi akan dikerjakan oleh prosesor yang berbeda, dan (2) dekomposisi geometrik, yaitu melakukan dekomposisi masalah yang ada (beban) menjadi beberapa submasalah dengan ukuran tertentu dimana tiap submasalah bisa dipecahkan secara parallel dan independen.

Strategi ini cenderung melihat struktur data pada masalah yang dihadapi. Untuk memperoleh ukuran kinerja yang baik pada sistem multiprosesor perlu diperhatikan beberapa hal, antara lain (1) topologi jaringan, (2) pembagian beban kerja prosesor (*load balancing*), serta (3) partisi data untuk mendukung proses dekomposisi yang optimal [7].

Untuk masalah GAXPY strategi yang digunakan adalah dekomposisi secara geometrik karena lebih banyak mempertimbangkan faktor struktur datanya. Kemudian persoalan GAXPY bersifat proses *updating* dan bersifat *pipeline* maka topologi yang digunakan adalah berbentuk ring atau melingkar.

Selanjutnya algoritma Gaxpy Column version akan dikerjakan dengan terlebih dahulu melakukan dekomposisi matriks A menjadi matriks kolom yang dapat dikerjakan pada prosesor berbeda. Setelah proses updating selesai baru dikirimkan ke prosesor berikutnya. Dengan demikian tiap prosesor akan

memiliki proses perhitungan Algoritma Gaxpy : *Column Version* berikut, dengan ukuran matriks yang lebih kecil (matriks kolom).

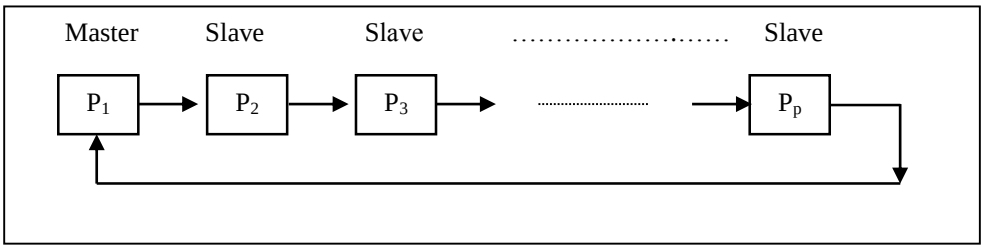
Algoritma Gaxpy : *Column Version*

```
{ jika  A ∈ R^(n×n), x ∈ R^n, dan y ∈R^n,  maka algoritma ini update nilai y
dengan cara  y = y + Ax)
  for j = 1 : n,
    for i = 1 : n,
       y(i) = A(i,j)x(j) + y(i),
    end,
  end,
```

III. HASIL

Berdasarkan berbagai pertimbangan diatas, yang meliputi (1) topologi ring yang dipilih karena model komunikasinya *pipeline* (proses *update* lebih mudah), dan (2) kemudahan pemaketan data dan *assembling vector solution* maka dipilih algoritma GAXPY *model column version*, karena akan memudahkan pengiriman paket data dan paket hasil per kolom. Berikut disajikan algoritma GAXPY versi kolom dengan menyesuaikan banyak prosesor yang akan dipakai.

Proses komunikasi data, yang meliputi proses pemaketan/pengiriman data dan assembling solusi, akan mengikuti model topologi ring yang dapat dijelaskan seperti gambar berikut.



Gambar 2. Rangkaian Prosesor dan Model Komunikasi Data

Algoritma Gaxpy Paralel

Misalkan X, Y $\in R^n$ dan A $\in R^{nxn}$ diketahui dan akan dicari vektor Y = Y + AX. Misalkan n = rp, p menyatakan banyak prosesor, dan r bilangan bulat. Jika tiap prosesor sudah memiliki A_loc dan X_loc, dan Y di prosesor 1, maka setelah selesai eksekusi program berikut, prosesor ke-1 akan menyimpan vektor Y yang sudah dilakukan proses *updating*.

Algoritma Gaxpy : *Column Version*

```
{ jika  A ∈ Rn×n, x ∈ Rn, dan y ∈Rn,  maka algoritma ini update nilai y
dengan cara  y = y + Ax}
     for j = 1 : n,
        for i = 1 : n,
          y(i) = A(i,j)x(j) + y(i),
          { →  untuk proses (lokal) tiap-tiap prosesor}
       end,
     end,
```

Pada prosesor ke – k dimana k=1 (Master)

Langkah - 1	{local.init}
Langkah - 2	(i) hitung Y = Y + A_loc * X_loc (ii) send (Y, right, k+1) →kirimkan ke prosesor berikutnya (iii) recv (Y, left, p) →menerima kiriman dari prosesor sebelumnya (terakhir)

Pada prosesor ke – k, dimana 1 < k ≤ p (Slave)

Langkah - 1	{local.init}
Langkah - 2	(1) hitung : w = A_loc X_loc (2) receive (Y, left, k) → menerima kiriman dari prosesor sebelumnya (3) update Y = Y + W → update nilai Y (4) Send (y, right, k+1) → kirimkan ke prosesor berikutnya

Local.init dinyatakan sebagai	Local.init[p, id, left, right, n, r=n/p, col = 1+(k-1)r:kr, A_loc=A(:,col), X_loc=X(col)]

dengan A(a:b, c:d} adalah submatriks baris ke – a sampai baris ke – b, dan dari kolom ke – c sampai kolom ke – d, dari matriks A. Local.init adalah tahapan inisialisasi variabel yang diperlukan, antara lain inisialisasi jumlah prosesor yang diperlukan, id_prosesor, id_prosesor_tetangga, ukuran data dan nilai awal yang lain.

IV. PEMBAHASAN

Analisis algoritma diperlukan sebagai ukuran prediksi unjuk kerja algoritma terhadap parameter yang digunakan, misalnya : ukuran memori, komunikasi bandwith, atau waktu komputasi. Untuk melakukan analisis algoritma dipakai pendekatan teoritis untuk menyatakan efesiensinya, yang akan independen terhadap mesin, bahasa pemrogaman dan pemrogam. Untuk maksud tersebut perlu ditentukan suatu unit ukuran sebagai dasar untuk menyatakan efisiensi algoritma, umumnya dinyatakan sebagai bentuk fungsi parameter, yang disebut sebagai prinsip invarian [4].

Meskipun dari hasil pengukuran diperoleh waktu komputasi berbeda , tetapi efisiensi algoritma tidak akan berubah. Misalkan dari dua kali implementasi, diperoleh hasil pengukuran waktu eksekusi sebesar t_1 (n) dan t_2(n) untuk input n, maka terdapat konstanta positif n sedemikian hingga berlaku t_1 (n) $\leq t_2$(n) untuk n cukup besar. Maka dikatakan algoritma menyelesaikan persoalan memiliki order t(n), t suatu fungsi.

Prinsip utama perancangan algoritma sekuensial menjadi algoritma sistem parallel adalah untuk menghasilkan waktu eksekusi yang lebih sedikit sehingga dapat diukur kenaikan percepatan dan tingkat efisiensi dengan order yang sama.

V. SIMPULAN DAN SARAN

Berdasarkan hasil pembahasan dapat disimpulkan bahwa dapat disusun model algoritma Gaxpy sejenis yang dirancang untuk diimplementasikan pada sistem multiprosesor. Meskipun model algoritma pada sistem multiprosesor lebih rumit namun diharapkan tetap dapat mereduksi waktu eksekusi algoritma tersebut. Algoritma Gaxpy tetap menerapkan prinsip *pipeline*, algoritma versi kolom, dan topologi ring untuk mendukung proses *updating.*

Sebagai saran algoritma Gaxpy parallel dapat diimplementasikan pada sistem multiprosesor yang sesungguhnya, agar dapat diukur tinggi kinerja dan efisiensi algoritma tersebut.

REFERENSI

[1] Akl, Selim G. *The Design and Analysis of Parallel Algorithms*, New York: Prentice Hall International Inc. 1989.

[2] Berstsekas and Tsitsiklis. *Parallel and Distributed Computation, Numerical Methods*, New Jersey: Prentice Hall. 1989.

[3] Dompierre, Julien. *Algoritmic Complexity of Matrix Operations*, Matrix Computations. Department of Mathematics and Computer-Science Laurentian University. Sudbury: Laurentian University. 2010.

[4] Evans, DJ. *Design of Parallel Numerical Algorithms.* London : Elsevier Science Publisher. 1992.

[5] Freman and Phillips. *Parallel Numerical Algorithms.* London : Prentice Hall. 1992.

[6] Golub and Van Loan. *Matrix Computation*, Third Edition. Baltimore: The John Hopkins University Press. 1996.

[7] Hwang, Kai and Briggs, FA. *Computer Architecture and Parallel Pocressing*. McGraw-Hill. Book Company. 1984.

[8] Mitchell and Griffiths. *The Finite Difference Method in partial Differetial Equations*, John Wiley & Sons. 1989.

[9] Poyrazoglu, Gokturk. *Matrix Multiplication*, New York: The State University of New York at Buffalo BEST Group – Winter Lecture Series, 2009.

[10] Quinn, M.J. *Designing Effisient Algorithms for Parallel Computer*. McGraw-Hill Book Company. 1987. 1-135.

[11] Tanembaum. *Structured Computer Organization*, Prentice Hall International Inc. 2002.

[12] Tri Prabawa, *Analisis Kinerja Algoritma Reduksi Siklis untuk Sistem Persamaan Linier dengan Matriks Tridiagonal berbasis PVM*. Proceeding Seminar Nasional Riset Teknologi Informasi Yogyakarta: STMIK Akakom. 2013.

[13] Tri Prabawa, *Karakteristik Kinerja Algoritma Rekursive Decoupling pada Multiprosesor berbasis PVM*. Proceeding Seminar Nasional Teknologi Informasi dan Multimedia 2015, tanggal 6-8 Februari 2015. Yogyakarta : STMIK Amikom. 2015.