# BAB II

# TINJAUAN PUSTAKA

## 2.1 Tinjauan Pustaka

Penelitian ini mengacu pada dua penelitian atau publikasi terdahulu dengan judul *Analisis Kinerja Algoritma Reduksi Siklis untuk Sistem Persamaan Linier dengan Matriks Tridiagonal berbasis PVM* (Seminar Nasional Riset Teknologi Informasi, 2013) dan *Karakteristik Kinerja Algoritma Recursive Decoupling pada Multiprosesor berbasis PVM* (Seminar Nasional Teknologi Informasi dan Multimedia, 2015).

Meskipun dua metode tersebut diterapkan pada persoalan yang sama namun memiliki hasil yang tidak sama, hal ini disebabkan karena dua metode tersebut memiliki karakteristik yang berbeda. Berdasarkan perbedaan karakteristik tersebut maka perlu diteliti lebih lanjut pada level implementasi.

## 2.2 Sistem Persamaan Linier Tridiagonal

Diberikan suatu sistem persamaan linier dari sistem tridiagonal Au = d, dengan A matriks tridiagonal dan dapat ditulis sebagai berikut:

$$\begin{pmatrix} b1 & c1 & & & & \\ a2 & b2 & c2 & & 0 & \\ & a3 & b3 & c3 & & \\ & & a4 & b4 & c4 & \\ & & & \cdot & \cdot & \\ & & & \cdot & \cdot & \\ 0 & & & & bn-1 & cn-1 \\ & & & & an & bn \end{pmatrix} \begin{pmatrix} u1 \\ u2 \\ \\ \cdot \\ \\ un-1 \\ un \end{pmatrix} = \begin{pmatrix} d1 \\ d2 \\ \\ \cdot \\ \\ dn-1 \\ dn \end{pmatrix} \quad (2)$$

Persoalannya adalah mencari vektor u yang memenuhi sistem persamaan Au = d, dan dapat ditulis $u = A^{-1} d$, jika matriks A nonsingular [9]. Menurut [4] dan [5] sistem

persamaan tersebut dapat diselesaikan dengan metode reduksi siklis dan metode pemecahan rekursif. Selanjutnya dalam studi komparasi ini untuk menyamakan penerapan akan dipilih sistem persamaan linier berukuran n dengan $n = 2^m$.

### 2.2.1 Metode Reduksi Siklis Untuk Sistem Tridiagonal

Karena matriks koefisien memiliki struktur yang spesifik, maka dimungkinkan pemakaian metode lain yang penyelesaiannya dapat dikerjakan secara paralel. Metode yang dibahas adalah metode reduksi siklis. Untuk menyederhanakan persoalan pada kasus ini akan ditinjau elemen matriks berupa tridiagonal konstan simetri. Maka dengan tidak menghilangkan sifat umum sistem tersebut, sistem (1) dapat ditulis sbb :

$$\begin{pmatrix} b & a & & & & & \\ a & b & a & & & 0 & \\ & a & b & a & & & \\ & & a & b & a & & \\ & & & & . & . & \\ & & & & . & . & \\ & 0 & & & a & b & a \\ & & & & & a & b \end{pmatrix} \begin{pmatrix} u1 \\ u2 \\ \\ . \\ \\ . \\ un-1 \\ un \end{pmatrix} = \begin{pmatrix} d1 \\ d2 \\ \\ . \\ \\ . \\ dn-1 \\ dn \end{pmatrix} \qquad (3)$$

ide dasar metode reduksi siklis adalah menurunkan baris-baris independen dengan cara reduksi pada baris yang berindeks ganjil atau genap. Metode ini cocok dikembangkan untuk matriks berukuran $n = 2^m-1, 2^m, 2^m+1$.

Secara umum prosedur metode reduksi siklis dapat diturunkan sebagai berikut:

Perhatikan 3 baris yang berdekatan, yaitu baris ke i-1, i, dan i+1

$$\begin{array}{lccccc} \text{i-1} & a & b & a & & \\ \text{i} & & a & b & a & \\ \text{i+1} & & & a & b & a \end{array} \qquad (4)$$

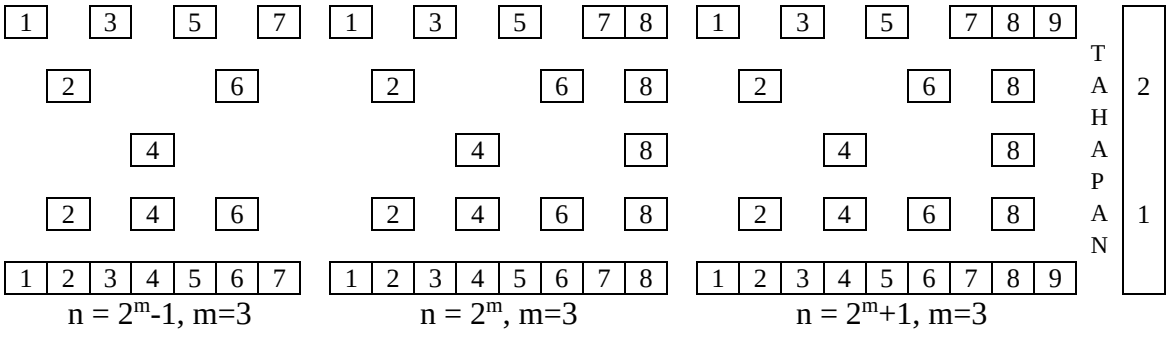
secara singkat bentuk (4) dapat dinyatakan dengan (…… a, b, a, ……). Dengan melakukan operasi baris elementer terhadap baris tengahnya, yaitu baris i dikalikan dengan konstanta b, kemudian ditambahkan dengan –a kali jumlahan baris ke i-1 dan i+1, maka diperoleh bentuk :

$$\begin{array}{cccccc} i-1 & a & b & a & & \\ i & a^{[1]} & 0 & b^{[1]} & 0 & a^{[1]} \\ i+1 & & & a & b & a \end{array} \qquad (5)$$

dengan memperhatikan baris tengah persamaan diatas diperoleh sistema persamaan baru yang dapat dinyatakan sebagai (….. $a^{[1]}$ , 0, $b^{[1]}$ , 0, $a^{[1]}$ …). Bentuk ini merupakan sistem tridiagonal dengan n = $2^{m-1}$-1, $2^{m-1}$, $^{2m-1}$ + 1 yang unsur-unsurnya hanya terdiri atas baris-baris berindek ganjil atau genap saja dari matrik A. Operasi ini juga dikerjakan terhadap vector ruas kanan d.

Misalkan dari bentuk persamaan (5) sistem persamaan baru yang diperoleh terdiri atas baris berindeks genap saja. Sedangkan untuk baris ganjil dapat dieliminasi oleh baris genap dengan substitusi balik. Jika proses tersebut dilakukan berulang-ulang, akhirnya akan diperoleh sistem persamaan baru yang minimal, dimana solusinya dapat dihitung secara langsung. Proses substitusi balik bertujuan untuk mencari nilai $u_j$ yang lain, yaitu dengan melakukan substitusi terhadap variabel yang telah diketahui. Metode reduksi siklis akan memberikan komputasi yang sangat cepat jika elemen subdiagonal dan superdiagonal sama dengan satu [9]. Hal ini dapat diperoleh dengan melakukan proses normalisasi sehingga persamaannya berbentuk (……, 1, b, 1, ……).

Secara umum tahapan solusi metode reduksi siklis terdiri dari dua tahapan utama yaitu (1) proses reduksi baris ganjil atau genap, dan (2) proses substitusi balik. Untuk lebih jelasnya dapat dilihat diagram berikut (n = $2^m$-1, $2^m$, $2^m$+1 dengan m = 3).




Gambar 2.1. Tahapan Solusi Algoritma Reduksi Siklis

### 2.2.2 Metode Pemisahan Rekursif Untuk Sistem Tridiagonal

Metode pemisahan rekursif berdasarkan pada strategi perubahan rank-satu (*rank-one updating*) dan proses partisi berulang pada sistem matriks sehingga didapat bentuk matriks diagonal blok yang masing-masing blok berukuran 2x2. Metode ini akan dipakai untuk menyelesaikan bentuk matriks tridiagonal sembarang, dan untuk sederhananya dipilih matriks dengan ukuran $n = 2^k$ dengan k = bulat positif.

**Proses Partisi**

Pandang sistem persamaan (1) yang selanjutnya dinyatakan sebagai :

$$\begin{pmatrix} b1 & c1 & & & & & \\ a2 & b2 & c2 & & & 0 & \\ & a3 & b3 & c3 & & & \\ & & a4 & b4 & c4 & & \\ & & & & \ddots & \ddots & \\ & & & & \ddots & \ddots & \\ & 0 & & & & bn-1 & cn-1 \\ & & & & & an & bn \end{pmatrix} \begin{pmatrix} u1 \\ u2 \\ \\ \cdot \\ \\ \cdot \\ un-1 \\ un \end{pmatrix} = \begin{pmatrix} d1 \\ d2 \\ \\ \cdot \\ \\ \cdot \\ dn-1 \\ dn \end{pmatrix} \qquad (6)$$

Dengan |bi|≥ |ai|+ |ci| untuk setiap i = 1, 2, 3, .....n, syarat ini menjamin proses reduksinya berjalan stabil. Proses partisi dilakukan terhadap matriks koefisien A sedemikian sehingga diperoleh bentuk blok matriks berukuran 2x2 sebagai berikut

$$\begin{pmatrix} e1 & c1 & & & \\ a2 & e2 & & 0 & \\ & e3 & c3 & & \\ & a4 & e4 & & \\ & & & & \\ & & & & \\ 0 & & & en-1 & cn-1 \\ & & & an & en \end{pmatrix} + \sum_{j=1}^{m-1} \begin{pmatrix} x1 \\ x2 \\ \cdot \\ \cdot \\ xn-1 \\ xn \end{pmatrix}^{j} X \begin{pmatrix} y1 \\ y2 \\ \cdot \\ \cdot \\ yn-1 \\ yn \end{pmatrix}^{(j)T} \quad (7)$$

Dengan $e_1 = b_1$, $e_{2j-1} = b_{2j-1} - c_{2j-2}$, j = 2, 3, .....m

$e_n = b_n$, $e_{2j} = b_{2j} - a_{2j+1}$, j =1, 2, .....m-1, dan m = n/2

Sedangkan vektor $x^{(j)}$ dan $y^{(j)}$merupakan vektor dengan elemen tidak nol hanya pada elemen ke 2j dan 2j+1 yang didefinisikan sebagai

$$x_k = \begin{cases} 1, untuk\ k = 2j, 2j+1 \\ 0,\ untuk\ yang\ lain \end{cases} \quad (8)$$

$$y_k = \begin{cases} a2j+1,\ untuk\ k = 2j \\ c2j, untuk\ k = 2j+1 \\ 0,\ untuk\ yang\ lain \end{cases} \quad (9)$$

Atau dapat ditulis sebagai

$x(j) = (0, 0, ....., 1, 1, .....0)^T$ (10)

$y(j) = (0, 0, .....,a_{2j+1}, c_{2j}, .....0)^T$ (11)

Persamaan (4) secara sederhana dapat ditulis menjadi $A= J + x^{(j)} y^{(j)T}$

Dengan J berbentuk

$$J = \begin{pmatrix} J1 & & & & \\ & J2 & & & \\ & & J3 & & \\ & & & & \\ & & & Jm & \\ & & & & Jm \end{pmatrix} \quad (12)$$

Sedangkan J1, J2, .....Jm terdiri dari submatrik 2x2 berbentuk

$\begin{pmatrix} e2j-1 & c2j-1 \\ a2j & e2j \end{pmatrix}$ dengan j = 1, 2, .....m dan m = n/2

Ide dasar partisi ini mengacu pada metode inversi Sherman-Morrison yang bertujuan untuk mereduksi kompleksitas komputasi $A^{-1}$ secara langsung. Menurut Golub [7] metode inversi Sherman-Morrison memberikan formula perhitungan invers matrik A berbentuk $A = (J + uv^T)$ dengan A elemen $R^{nxn}$ , Sedangkan u dan v vektor berukuran nx1, Invers matrik A didefinisikan sebagai

$$A^{-1} = (J + uv^T)^{-1} = J^{-1} - ( J^{-1} u v J^{-1})/ (1 + v^T J^{-1} u) \quad (13)$$

Dengan memakai persamaan tsb, maka invers matrik

$A = J + xy^T$ adalah

$$A^{-1} = (J + xy^T) = J^{-1} - \alpha ( J^{-1} x) (y^T J^1) \quad (14)$$

Dengan $\alpha = 1/ (1 + y^T J^{-1} x)$

Kemudian dengan menghitung invers A, solusi persamaan (3) adalah

$$\begin{aligned} U &= A^{-1} d = (J + xy^T)^{-1} d \\ &= \{J^{-1} - \alpha ( J^{-1} x) (y^T J^1)\} d \\ &= J^{-1} d - \alpha J^{-1} x y^T J^1 d \\ &= J^{-1} d - \alpha (J^{-1} x) y^T (J^{-1} d) \end{aligned} \quad (15)$$

dengan $\alpha = 1/ (1 + y^T J^{-1} x)$

Dengan menghitung nilai-nilai (10), yaitu $J^{-1}x$, $J^{-1}$ d dan α diperoleh solusi dar sistem persamaan (3).

**Proses Pemisahan Rekursif.**

Proses ini bertujuan mengubah vektor u dan d sedemikian hingga elemen-elemennya menjadi berpasangan (*couple*) yang dapat ditulis

$$u = \begin{pmatrix} u1 \\ u2 \\ u3 \\ u4 \\ . \\ . \\ un-1 \\ un \end{pmatrix} = \begin{pmatrix} u1 \\ u2 \\ \\ \\ . \\ . \\ dm \end{pmatrix} \qquad (16)$$

$$\text{dan} \quad d = \begin{pmatrix} d1 \\ d2 \\ d3 \\ d4 \\ . \\ . \\ dn-1 \\ dn \end{pmatrix} = \begin{pmatrix} d1 \\ d2 \\ \\ \\ . \\ . \\ dm \end{pmatrix} \qquad (17)$$

dari persamaan (15) solusi Au = d dapat dinyatakan

$$u = \left(J + \sum_{j=1}^{m-1} x^{(j)} y^{(j)T}\right)^{-1} d$$

$$= \left(J^{-1} - \alpha J^{-1} \sum_{j=1}^{m-1} x^{(j)} y^{(j)T}\right)^{-1} d$$

$$= (Jd - \alpha J^{-1} \sum_{j=1}^{m-1} x^{(j)} y^{(j)T})^{-1} d \quad (18)$$

dengan

$$\alpha = 1/(1 + \sum_{j=1}^{m-1} y^{(j)T}) J^{-1} \sum_{j=1}^{m-1} x^{(j)})$$

Dengan menyatakan $J^{-1}$ d' = u' dan $J^{-1}$ $x^{(j)} = g^{(j)}$ maka bentuk (18) dapat disederhanakan menjadi

$$u = (u' - \alpha g^{(j)} y^{(j)T} u') = (1 - \alpha g^{(J)} (y^{(j)T}) u' \ldots \quad (19)$$

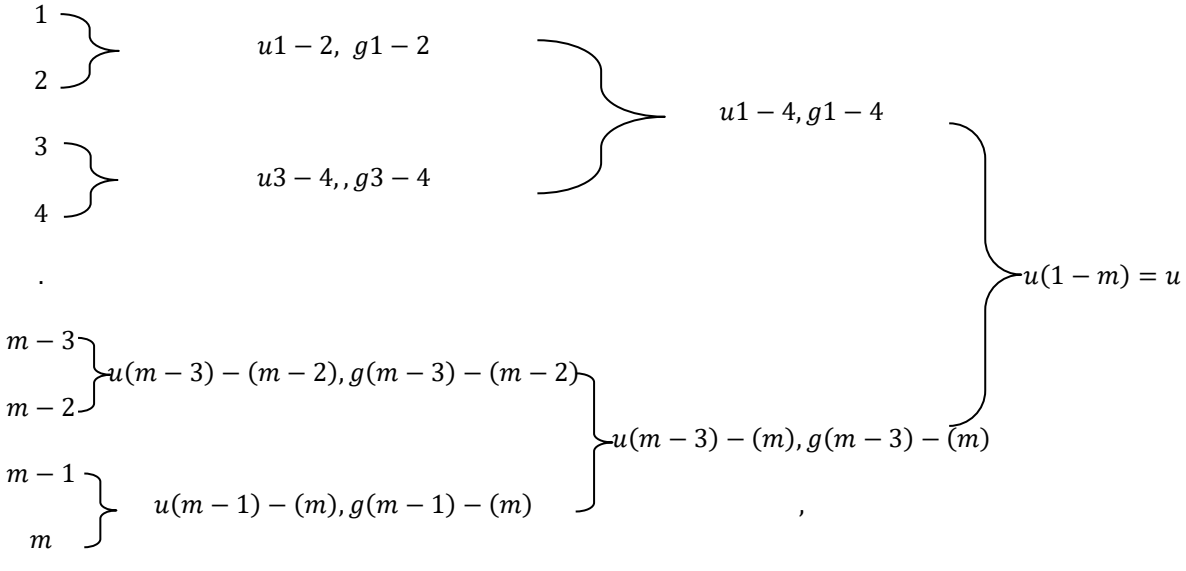
dengan $\alpha = 1/( 1 + y^{(j)T} g^{(j)})$ dan j = 1, 2, ….m-1

Persamaan (14) merupakan bentuk formula perubahan rank satu, maka tahapan solusi untuk persamaan diatas adalah

(a) Mencari solusi vektor u' dari persamaan u' = $J^{-1}$ d

(b) Mencari solusi vektor $g^{(j)}$ dari persamaan $g^{(j)} = J^{-1} x^{(j)}$

(c) Melakukan perubahan rank satu (19)




Gambar 2.2 Tahapan Penyelesaian Metode Pemisahan Rekursif ($n = 2^m$)

## 2.3 Model Sistem Komputasi Paralel

Kebutuhan akan kecepatan penyelesaian suatu masalah sangat diperlukan, terutama untuk persoalan yang cukup besar dan kompleks serta hasil prosesnya segera diperlukan. Beberapa aplikasi yang memerlukan penanganan seperti ini, misalnya : komputasi pada prakiraan cuaca, komputasi aerodinamik, kecerdasan buatan, keamanan pada reaktor nuklir, pengindraan jauh [21]. Untuk memenuhi kebutuhan seperti ini dapat ditempuh dengan cara menyelesaikan persoalan secara cepat, kontribusi mesin (komputer) dan jika memungkinkan dikerjakan secara paralel.

Ada beberapa cara untuk meningkatkan kecepatan komputasi, antara lain mengembangkan pembuatan mesin yang lebih cepat, tetapi perlu teknologi tinggi dan biaya mahal atau secara arsitektural dapat dilakukan dengan membangun sistem paralel dari mesin yang ada, murah tetapi relatif lambat [12] [22]. Selain itu dapat dilakukan dengan cara memanipulasi banyaknya operasi komputasi yang ada sedemikian hingga dapat dikerjakan secara paralel [21].

Suatu sistem paralel dapat digambarkan sebagai teknik pemrosesan secara simultan pada subproses yang independen. Menurut Hwang dan Briggs [13] pemrosesan paralel didefinisikan sebagai bentuk pemrosesan yang efisien dengan menitik beratkan pada eksploitasi kejadian-kejadian yang bersamaan. Tujuan utamanya adalah untuk mereduksi waktu proses yang dibutuhkan maupun menyelesaikan masalah yang sebelumnya di pandang terlalu besar [10].

Sistem komputer paralel merupakan gabungan beberapa prosesor saling terhubung dan bekerja sama menyelesaikan masalah secara simultan pada submasalah yang berbeda. Tipe komputer yang sering dipertimbangkan sebagai komputer paralel dikenal

sebagai sistem vektor prosesor atau multiprosesor. Sedangkan konfigurasi arsitekturnya dibedakan menjadi *pipeline, array processor*, dan *multiprocessor system* [10].

Menurut [3] sistem multiprosesor dapat dibedakan berdasarkan tipe dan jumlah prosesor yang dipakai. Komputer paralel dengan banyak prosesor disebut *massively parallel*, sedangkan sedikit prosesor *coarse-grained paralelism*. Berdasarkan mekanisme kerjanya, prosesor dapat bekerja secara sinkron atau asinkron. Pembedaan lain didasarkan pada interkoneksi prosesor dan mekanisme komunikasinya, yang dibedakan menjadi model *shared memory* dan *distributed system*.

### 2.3.1 Klasifikasi Komputer Paralel

Pada dasarnya pemrograman baik secara sekuensial maupun paralel adalah proses manipulasi data. Prosesnya dapat digambarkan sebagai aliran instruksi (algoritma) yang merupakan perintah kepada komputer secara bertahap. Menurut taksonomi Flynn dalam [1] [21] komputer dapat dibedakan menjadi 4 model.

Komputer SISD (*Single Instruction Single Data*). Semua komputer sekuensial termasuk dalam klas SISD. Meskipun eksekusi dari instruksi dapat dilakukan secara *pipeline* akan tetapi bersifat sekuensial karena hanya memiliki satu unit pemroses.

Komputer MISD (*Multiple Instruction Single Data*). Secara definitif komputer MISD memiliki sejumlah prosesor, kontrol unit dan aliran instruksi, tetapi hanya memiliki satu aliran data. Data dalam memori bersama dimanipulasi oleh semua prosesor secara bersamaan. Kendalanya terjadi jika tiap prosesor melakukan proses *update* data bersamaan, padahal data lama masih dipakai untuk proses lain. Kecuali hal-hal khusus komputer ini jarang dibicarakan, dan belum ada jenis komputer MISD [10] [21].

Komputer SIMD (*Single Instruction Multiple Data*). Pada komputer SIMD terdapat sejumlah prosesor dan aliran data, tetapi hanya memiliki satu aliran instruksi. Setiap

prosesor memiliki memori lokal dan duplikasi program yang sama. Sehingga semua prosesor akan mengeksekusi instruksi yang sama, tetapi pada data yang berbeda, dan prosesor akan bekerja secara sinkron.

Komputer MIMD (*Multiple Instruction, Multiple Data*). Pada komputer MIMD terdapat sejumlah prosesor, aliran instruksi, dan aliran data. Setiap prosesor memiliki kontrol unit dan memori lokal. Prosesor akan bekerja sesuai dengan instruksi program dan pada data yang berbeda. Sedangkan mekanisme kerja prosesornya dapat bekerja secara asinkron.

### 2.3.2 Topologi Interkoneksi Jaringan Prosesor

Topologi jaringan prosesor adalah suatu bentuk interkoneksi sejumlah prosesor yang terhubung sebagai suatu jaringan. Bentuk jaringan prosesor dapat di gambarkan sebagai suatu graph dari sejumah titik (prosesor) dan garis hubungan (*edge*) antar titik sebagai link prosesor. Maka *path* dalam graph tersebut merupakan jalur komunikasi antar prosesor.

Pada sistem komputasi paralel, faktor komputasi relatif terhadap kecepatan komunikasi akan selalu diperhatikan. Karena kecepatan eksekusi suatu algoritma pada sistem paralel dihitung berdasarkan perbandingan waktu eksekusi sekuensial terhadap waktu eksekusi paralel termasuk waktu untuk berkomunikasi. Secara implisit faktor interkoneksi jaringan prosesor akan mempengaruhi waktu komunikasi yang diperlukan.

Menurut [3] untuk mengevaluasi bentuk topologi jaringan prosesor, kriteria yang seringkali dipakai adalah diameter jaringan dan tingkat konektivitas. Jarak sepasang titik adalah jumlah minimum link yang harus dilalui diantara kedua titik tersebut. Pada suatu jaringan prosesor berdiameter *r,* maka waktu untuk suatu paket data dapat ditransfer antara dua titik adalah *0(r).* Sedangkan konektivitas jaringan merupakan

ukuran dari jumlah *edge* independen pada sepasang titik. Tingkat konektivitas yang tinggi sangat diharapkan untuk menjaga kehandalan proses komunikasi. Jika suatu jaringan prosesor mempunyai tingkat konektivitas *k,* maka komunikasi antara dua titik dapat diparalelkan paling sedikit dengan *k* path.

Ada beberapa pola topologi jaringan prosesor yang cukup dikenal, antara lain pola *ring, mesh, tree*, dan pola *hypercube.* Pemilihan pola topologi jaringan prosesor akan dipengaruhi oleh bentuk persoalan yang dihadapi dan model komputer paralel ada. Jika komputer paralel memiliki jumlah prosesor terbatas, maka topologi jaringan prosesor yang dapat dibuat akan terbatas pula.

Model topologi ring merupakan pola sederhana dan seringkali dipakai. Pada model ring terdapat satu path komunikasi untuk tiap pasang titik prosesor. Model ini memberikan kemudahan dalam menyelesaikan masalah komunikasi. Jika pola ring dengan p prosesor maka path komunikasi yang terjadi sebesar p edge, artinya berdiameter p.

Model topologi *tree* dengan *p* prosesor akan memberikan path komunikasi pada tiap parang prosesor minimal *(p-1)* edge, dan diameternya *(p-1)*. Kekurangan topologi tree adalah mempunyai faktor konektivitas rendah. Pada saat terjadi dua atau lebih subset tree, maka prosesor-prosesor tidak dapat saling berkomunikasi. Kasus khusus topologi tree adalah pola star yang mempunyai diameter minimal.

Untuk suatu persoalan yang cukup besar akan efisien jika persoalan tersebut dapat dipetakan secara geometris. Maka pola topologi yang cocok adalah topologi mesh, yang dipandang sebagai pola prosesor *array* terhubung. Sekumpulan prosesor pola *mesh* berdimensi d dapat dinyatakan sebagain titik-titik dalam ruang berdimensi d, dan terjadi komunikasi langsung antara titik yang berdekatan.

Model lain yang cukup dikenal adalah *hypercube*. Struktur pola ini terbatas pada aplikasi dengan jumlah prosesor yang ada. Pada dasarnya struktur prosesor hypercube mirip dengan pola mesh terhubung. Secara umum model *hypercube* dapat di generalisasi. Suatu kumpulan titik dalam ruang berdimensi d dengan nilai koordinat 0 atau 1. Asumsikan bahwa titik-titik tersebut sebagai titik sudut atau suatu *cube* dimensi d, yang berkorespondensi dengan suatu himpunan jaringan prosesor. Suatu link komunikasi terjadi jika terdapat dua titik yang memiliki satu perbedaan pada koordinatnya. Maka pola jaringan tersebut akan menghasilkan suatu model *cube*.

## 2.4 Aspek Komunikasi Pada Sistem Paralel

### 2.4.1 Metode Komunikasi

Sifat utama pada algoritma sisitem paralel adalah adanya aktivitas proses-proses yang berlangsung secara bersamaan. Karena proses tersebut merupakan bagian dari penyelesaian suatu masalah, maka diperlakukan suatu koordinasi yang baik antar proses yang terjadi. Hal ini menunjukkan bagaimana suatu proses pada suatu prosesor saling berkomunikasi dengan suatu proses pada prosesor yang lain. Dari model dan jenis komputer paralel yang telah ada, maka komunikasi antarprosesor dapat berlangsung secara *shared memory system* atau *distributed system*. Maka teknik pemrogaman pada kedua model komunikasi ini dikenal dengan model *shared memory* dan *model message passing*.

Model *shared memory* merupakan perluasan model sistem sekuensial, dengan mengasumsikan bahwa prosesor-prosesor akan mengakses data pada unit memori yang sama. Sedangkan model *message passing* mengasumsikan bahwa setiap prosesor

memiliki memori tersendiri untuk menyimpan data masing-masing. Sehingga proses pertukaran informasi pada model *message passing* akan berlangsung secara eksplisit.

### 2.4.2 Pinalti Komunikasi

Pada algoritma sistem paralel, waktu yang dipakai untuk komunikasi merupakan bagian dari total waktu untuk menyelesaikan persoalan. Dari pengertian tersebut tersirat bahwa algoritma mengalami penundaan waktu eksekusi akibat terjadinya komunikasi [3]. Komunikasi pinalti (*communication pinalty*) dihitung sebagai rasio waktu eksekusi dan komputasi.

$$CP = T_{total} / T_{komputasi}$$

$T_{total}$ = total waktu yang dipakai algoritma menyelesaikan persoalan, dan

$T_{komputasi}$ = waktu yang diperlukan untuk tahap komputasi.

Untuk mengevaluasi persoalan komunikasi terlebih dahulu perlu dipahami masalah sistem terdistribusi sebagai suatu jaringan prosesor terhubung. Pertukaraan informasi antarprosesor yang berisi paket kerja, terdiri dari pengunaan memori, ukuran data dan masalah lain yang berkaitan, proses transmisinya berlangsung melalui saluran komunikasi. Ukuran paket kerja dipetakan dalam bit. Selama pertukaran informasi berlangsung proses transmisi bit-bit paket kerja tidak dapat diinterupsi. Hal seperti ini merupakan salah satu penyebab adanya penundaan waktu terhadap eksekusi algoritma.

Berdasarkan mekanisme kerjanya, komunikasi pinalti meliputi waktu pemaketan data, waktu antrian data, waktu transmisi data, dan waktu propagasi antara bit terakhir dikirim sampai bit terakhir diterima. Beberapa faktor lain yang juga berpengaruh terhadap komunikasi pinalti adalah algoritma yang dipakai untuk mengontrol jaringan komunikasi, topologi prosesor dan struktur persoalan dengan algoritma yang sesuai.

### 2.4.3 Tradeoffs Komunikasi Dan Paralelisasi

Faktor paralelisasi dan penggunaan sejumlah prosesor secara aktif untuk menyelesaikan persoalan secara simultan akan saling bergantung. Sehingga tingkat paralelisasi akan bergantung pada proses pembagian beban kerja untuk setiap prosesor. Agar lebih efisien sebaiknya waktu kerja setiap prosesor relatif sama, dan waktu *idle* prosesor minimum. Untuk maksud ini dapat dilakukan pembagian beban kerja yang sama agar diperoleh waktu eksekusi algoritma yang optimal dan efisien. Proses ini disebut sebagai *load balancing*. Pada proses *load balancing* yang baik, banyaknya proses komputasi dan komunikasi pada setiap tahapan waktu akan bersesuaian [11].

Untuk melihat proses-proses yang dapat dikerjakan secara simultan, langkah pertama adalah melakukan proses dekomposisi persoalan sehingga diperoleh bagian-bagian yang independen. Selanjutnya bagian-bagian ini akan dipetakan ke dalam model komputasi sistem paralel. Menurut [2] proses dekomposisi dapat dilakukan secara geometris dan algoritmis. Dekomposisi geometris adalah membagi persoalan secara fisik agar setiap prosesor hanya menangani bagian kecil persoalan, agar diperoleh efisiensi serta memudahkan implementasi. Sedangkan dekomposisi algoritmis adalah setiap prosesor akan menangani bagian kecil proses dari seluruh algoritma, misalnya model *pipeline.*

Dari kedua proses dekomposisi tersebut seringkali masih terjadi kesulitan dalam menuliskan algoritma. Sehingga diperlukan suatu pendekatan lain yang dapat dipakai sebagai bahan pertimbangan maupun perbandingan. Pertama membuat algoritma paralel dengan jalan memodifikasi algoritma sekuensial yang ada tanpa mempertimbangkan mesin yang akan dipakai. Kedua membuat algoritma paralel

dengan memfokuskan pada mesin yang akan dipakai agar diperoleh algoritma yang sesuai, terutama menyangkut aspek komunikasi [3].

Pada umumnya dari suatu persoalan dapat dibuat beberapa algoritma, namun hanya dipilih yang efisien. Secara teoritis tidak ada formula yang baku untuk menganalisa efisiensi suatu algoritma, akan tetapi selalu berkaitan dengan pembuktiaan, intuisi, dan pengalaman. Untuk tujuan ini dapat dipakai suatu pendekataan secara empiris, teoritis atau *hybrid*. Pendekataan empiris adalah mengimplementasikan algoritma dan mengujinya dengan beberapa input yang berbeda. Sedangkan pendekataan teoritis adalah menganalisa algoritma secara matematis terhadap sejumlah parameter yang dipakai dan menyatakan algoritma sebagai suatu fungsi dari parameter tersebut. Pendekatan teoritis akan independen terhadap mesin, bahasa pemrogaman yang dipakai, dan kemampuan pemrograman. Pendekatan hybrid adalah membentuk fungsi algoritma secara teoritis, kemudian diperlukan parameter numeris untuk menguji secara empiris, sehingga akan diperoleh bentuk yang dapat dipakai sebagai prediksi terhadap kasus yang lebih besar.

## 2.5 Ukuran Kinerja Algoritma Sistem Paralel

Karena sistem komputasi paralel memberikan peningkatan kecepatan penyelesaian, maka perlu didefinisikan suatu besaran yang merupakan ukuran peningkatan kecepatan yang sebenarnya. Agar diperoleh gambaran mengenai kriteria algoritma paralel yang baik dan memiliki fleksibilitas yang cukup terhadap sebarang mesin.

### 2.5.1 Waktu Eksekusi

Secara empiris perilaku algoritma paralel dapat dilihat pada hasil pengukuran terhadap waktu eksekusi (*running time*). Waktu eksekusi didefinisikan sebagai waktu

yang diperlukan algoritma untuk menyelesaikan suatu masalah secara paralel, yang dihitung pada saat algoritma dimulai sampai algoritma berakhir. Jika sejumah prosesor memulai bekerja dan perhitungan paralel tidak bersamaan, maka waktu eksekusi dihitung mulai saat prosesor paling awal bekerja hingga saat prosesor terkahir selesai.

### 2.5.2 Percepatan

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. Mengingat bahwa komputasi pada sistem multi prosesor akan lebih cepat dibanding dari sistem komputer biasa (yang disebabkan adanya tambahan prosesor), maka perlu didefinisikan suatu besaran yang merupakan ukuran peningkatan kecepatan yang sebenarnya. Besaran ini adalah efisiensi dan peningkatan kecepatan (*speed-up*) dari sistem multiprosesor yang selanjutnya menjadi ukuran kualitas algoritma paralel.

Kualitas algoritma paralel dapat dilihat dengan menghitung rasio waktu eksekusi algoritma dengan satu prosesor terhadap waktu eksekusi algoritma pada multiprosesor. Menurut Freeman dan Phillips [10]. Percepatan (*algorithmatic speed-up ratio* ) didefinisikan sebagai berikut

$$S_p(n) = \frac{T_{1\ (n)}}{T_{p\ (n)}}$$

$T_1(n)$ = waktu eksekusi algoritma pada satu prosesor, $T_p(n)$ = waktu eksekusi algoritma pada *p* prosesor, dan n = Ukuran data.

Secara analitis dengan bertambahnya jumlah prosesor yang dipakai, percepatannya menjadi naik. Kenaikan ini akan mengikuti pola hukum *Amdahl*, maka percepatannya akan memenuhi persamaan berikut.

$$S_{p(n)} \leq \frac{1}{s + r/p}$$

Dengan *r* = bagian dari program yang dapat diparalelkan, dan s = 1 – r, merupakan bagian tidak dapat diparalelkan, serta p = jumlah prosesor.

### 2.5.3 Biaya dan Efisiensi

Dari implementasi algoritma pada sistem multiprosesor mengakibatkan adanya beban biaya yang harus ditanggung. Biaya (*cost)*, sebagai salah satu ukuran kualitas algoritma paralel, didefinisikan sebagai hasil kali waktu eksekusi secara paralel dan banyaknya prosesor yang dipakai, atau dapat ditulis

$$\text{Cost} = T_p(n) \;\; x \;\; p.$$

Dengan kata lain besaran biaya sama dengan jumlah tahapan solusi yang dieksekusi oleh semua prosesor dalam menyelamatkan persoalan.

Disamping itu kualitas algoritma pada sistem paralel dapat dilihat dengan menghitung efisiensinya. Efisiensin didefinisikan sebagai

$$E_{(\square fficiency)} = \frac{S_{p\,(n)}}{p}$$

$S_p(n)$ = percepatan (*Speed-up*), p = jumlah prosesor, n = ukuran data. Jika biaya algoritma sistem paralel sama dengan waktu eksekusi pada satu prosesor, maka efisiensinya akan mencapai 100% [1].

## 2.6 Prinsip Invarian

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 independen terhadap mesin, bahasa pemrogaman dan pemrogam. Untuk maksud tersebut perlu ditentukan suatu unit ukuran sebagai dasar untuk menyatakan efisiensi agoritma, umumnya dinyatakan sebagai bentuk fungsi parameter. Prinsip ini disebut 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 o 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.