BAB III
METODE PENELITIAN

Dalam penyusunan skripsi ini, dirancang suatu sistem komputasi paralel untuk penghitungan integral dimensi tiga dengan metode Monte Carlo. Metode penelitian yang digunakan pada penyusunan skripsi ini dijelaskan dalam uraian berikut.

3.1 Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut meliputi:

1. komputasi paralel,
2. *cluster* Beowulf,
3. metode Monte Carlo untuk penghitungan integral definit,
4. pemrograman dengan Open MPI,
5. pembangkit bilangan acak semua pustaka standar di sistem operasi Linux.

3.2 Penentuan Spesifikasi Alat

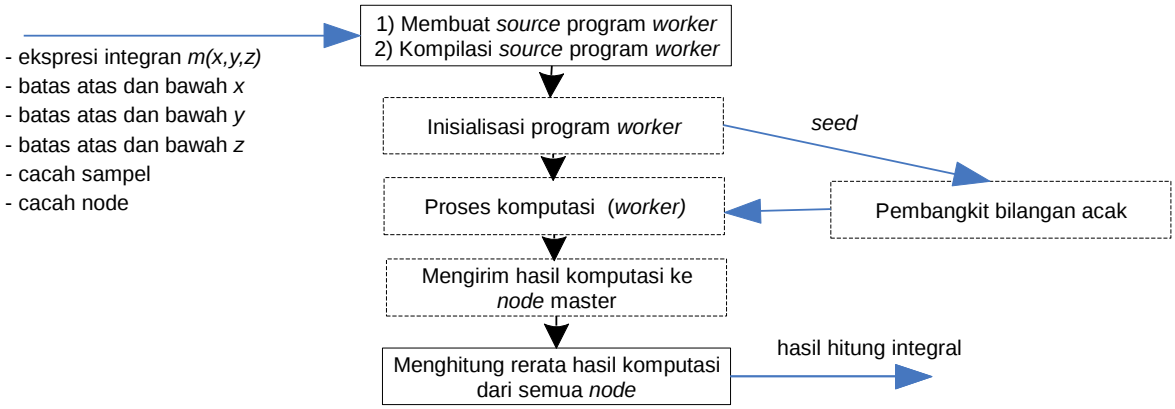
Perangkat yang digunakan untuk implementasi sistem komputasi paralel integral definit rangkap tiga metode Monte Carlo diuraikan sebagai berikut.

a) Perangkat keras:
   1) 10 unit komputer personal HP Compaq CQ3138L: Intel Core 2 Duo E7500, DDR3 2 GB,
   2) 6 unit komputer personal: Intel i3 550/540, DDR3 2 GB
   3) 1 unit komputer personal: Intel Dual Core E5400, DDR2 2 GB,
   4) *switch* DES 1024R dan perkabelan UTP Cat 5e,
   5) energimeter Wanf D02A

b) Perangkat lunak:
   1) GNU/Linux Salix64: kernel 3.2.29, glibc 2.15, gcc 4.7.1,
   2) GNU/Linux WattOS 12.04.1 LTS: kernel 3.2.0, glibc 2.15, gcc 4.6.5,
   3) Open MPI 1.7.

3.3 Abstraksi Sistem

Sistem komputasi paralel ini terdiri dari beberapa komponen yang dijelaskan dalam Gambar 3.3.



Gambar 3.3 Abstraksi Sistem Komputasi Paralel

Keterangan:

1)Pengguna memberikan masukan dalam bentuk argumen program yang berisi ekspresi integran *m(x,y,z)* dalam sintaks bahasa pemrograman C yang valid; batas atas dan bawah masing-masing untuk *x*, y, z; cacah sampel dan *proses*,

2)Program akan membuat berkas *source* program *worker* sesuai dengan masukan dari pengguna di poin 1, dan selanjutnya melakukan kompilasi. Jika kompilasi sukses program akan menjalankan program *worker* hasil kompilasi sebagai *proses* master.

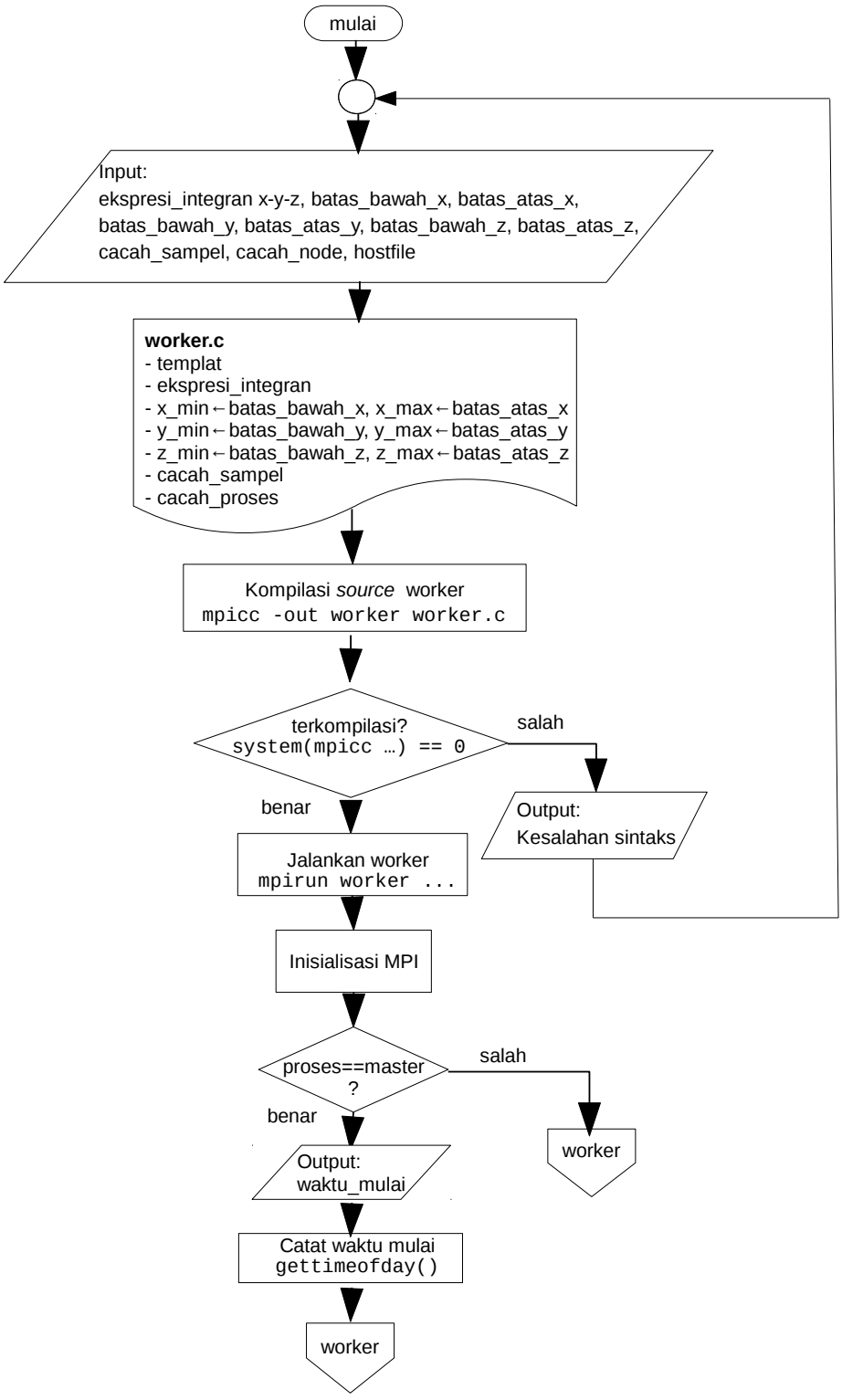
3)Di masing-masing *proses* komputasi, program *worker* akan meberi nilai *seed* pembangkit bilangan acak, kemudian menjalankan rutin komputasi dengan masukan dari pembangkit bilangan acak di tiap iterasi algoritma metode Monte Carlo.

4)Setelah proses komputasi selesai program *worker* mengirim hasil komputasi ke *proses* master.

5) *Proses* master akan menghitung rerata hasil komputasi dari semua *proses* dan menampilkan keluaran proses komputasi beserta informasi yang berkaitan.

3.4 Langkah Kerja Sistem

Inisialisasi rutin komputasi paralel dijelaskan oleh Gambar 3.4.1.

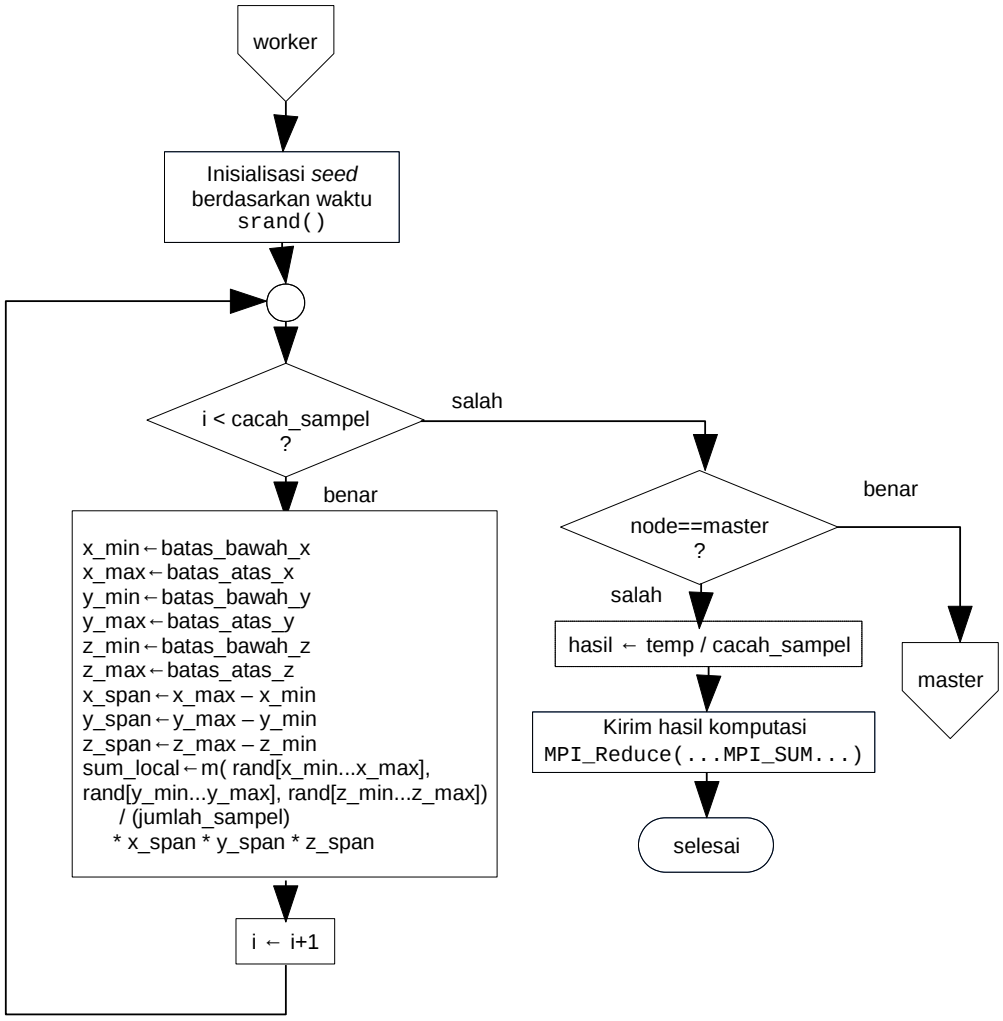


Gambar 3.4.1 Diagram Alir Inisialisasi Proses Komputasi Paralel di *Node* Master

Rutin program utama yang dijalankan di *node* master adalah sebagai berikut:

1) pengguna memberikan masukan dalam bentuk argumen program yang berisi: ekspresi integran sebagai fungsi *x*, *y*, atau *z* dalam sintaks bahasa pemrograman C yang valid; batas atas dan bawah masing-masing untuk *x*, y, z; cacah sampel; cacah *proses;* dan *hostfile* (berkas *plaintext* yang digunakan untuk mengidentifikasi *node* eksekusi MPI),
2) kode program (worker.c) akan disusun dengan konkatenasi templat-templat yang berisi *preprocessor directive*, dan kode berkaitan dengan konteks rutin program: cacah sampel, pemberian nilai batas atas dan bawah untuk *x, y, z*; ekspresi integran masukan pengguna,
3) kode program dikompilasi dengan mpicc (*wrapper* gcc dengan pustaka MPI*).* Jika kompilasi sukses program akan dilanjutkan dengan menjalankan program worker, jika kompilasi pengguna diberikan keluaran mengenai kesalahan sintaks dan dapat mengulang eksekusi program,
4) rutin utama program menjalankan program worker dengan *system call* `system(mpirun … worker)`,
5) inisialisasi MPI dilakukan dengan fungsi `MPI_Init().` Selanjutnya proses diidentifikasi dengan `MPI_Comm_rank()`. Rutin program dilanjutkan ke rutin komputasi di masing-masing *proses* worker (Gambar 3.3),
6) *proses* master (*rank 0)* akan mencatat waktu mulai proses komputasi dengan fungsi `gettimeofday()` sebelum melanjutkan ke rutin *proses* worker.

Gambar 3.4.2 menjelaskan rutin program worker yang dijalankan di *node* komputasi paralel.

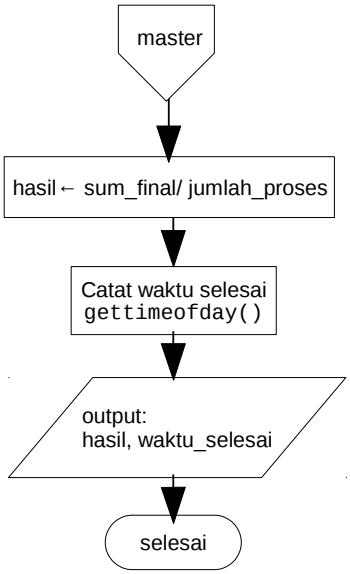


Gambar 3.4.2 Diagram Alir Rutin Program worker di tiap *Proses* Komputasi

Rutin komputasi paralel program worker di masing-masing *proses* komputasi adalah:

1) program worker menginisialisasi *seed* pembangkit bilangan acak pustaka standar dengan fungsi `srand()` dengan argumen yang diambil dari waktu *epoch*,
2) program worker melakukan iterasi penghitungan nilai fungsi *m(x,y,z)* sesuai batas atas dan bawah untuk *x*, *y*, dan *z*; sejumah cacah sampel, membagi dengan cacah sampel dan menjumlahkannya ke variabel sum_local.
3) setelah iterasi selesai, rerata fungsi *m(x,y,z)* dikirimkan sekaligus dijumlahkan ke *node* master dengan fungsi `MPI_Reduce()`.

Gambar 3.4.3 menjelaskan rutin tahap akhir proses komputasi paralel yang meliputi agregrasi hasil dan pencatatan waktu selesai.



Gambar 3.4.3 Diagram Alir Tahap Akhir Proses Komputasi Paralel yang Dijalankan *Node* Master

Rutin komputasi tahap akhir komputasi paralel yang dijalankan di *node* master adalah:

1) menerima hasil komputasi masinng-masing *proses* yang telah dijumlahkan dengan fungsi MPI_Reduce() dan menghitung rerata hasil komputasi dari semua *proses*,
2) mencatat waktu selesai proses komputasi paralel dengan fungsi `gettimeofday()`,
3) mencetak keluaran hasil dan waktu selesai proses komputasi.

3.5 Pengujian Sistem

Pengujian dilakukan untuk mengukur performansi sistem yang meliputi: waktu proses komputasi, ketelitian dan ketepatan hasil komputasi paralel integral definit terhadap variabel cacah sampel, pengingkatan kecepatan dan kesangkilan terhadap variabel cacah *proses* dan cacah *node* komputasi.

Waktu proses komputasi diukur dengan menghitung waktu komputasi dengan cacah sampel sebagai variabel bebas dengan cacah *proses* dan *node* yang sama.

Ketelitian diukur dengan membandingkan secara relatif beda hasil komputasi paralel integral definit rangkap tiga Metode Monte Carlo dan metode metode balok dengan hasil penghitungan integral definit secara analitis.

Ketepatan diukur dengan menghitung nilai varians dari 5 hasil komputasi dengan persamaan integral, batas-batas integrasi, dan cacah sampel yang sama, namun dengan nilai *seed* yang berbeda.

Peningkatan kecepatan diukur dengan membandingkan waktu eksekusi proses komputasi tunggal (1 *proses* di 1 *node*) terhadap waktu eksekusi proses komputasi dengan cacah *proses* atau *node* sebagai variabel bebas.

Kesangkilan (efisiensi) diukur dengan membandingkan peningkatan kecepatan dengan cacah *proses* atau *node.*

Penggunaan energi diukur dengan pengukuran daya dan waktu komputasi untuk tiap *node* komputasi tunggal tunggal. Penggunaan energi total diukur dengan perkalian penggunaan energi tiap *node* komputasi tunggal dan cacah *node* komputasi sejenis yang aktif.

3.6 Kesimpulan dan Saran

Kesimpulan analisis dari pengujian dipaparkan. Tahap selanjutnya adalah pembuatan saran untuk perbaikan dan pengembangan penelitian selanjutnya.