BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Implementasi algoritma paralel untuk aplikasi pengujian perkalian matriks dan pengurutan data (*sorting*) pada prosesor *quadcore* mendapatkan hasil-hasil analisis sebagai berikut :

1) Implementasi algoritma paralel dengan MPICH2 pada *cluster* PC *quadcore* menghasilkan percepatan relatif maksimal 5,987 pada perkalian matriks 1024 x 1024 dan jumlah proses 16 sedangkan percepatan relatif minimal 1,180 dicapai pada perkalian matriks 256 x 256 dan jumlah proses 4. Efisiensi terendah 0,120 pada perkalian matriks 256 x 256 dan jumlah proses 16 sedangkan efisiensi tertinggi 0,967 pada perkalian matriks 1024 x 1024 dan jumlah proses 4.
2) Implementasi algoritma paralel dengan Cilk++ pada PC *quadcore* menghasilkan percepatan relatif maksimal 25,126 pada perkalian matriks 1024 x 1024 dan jumlah *thread* 4 sedangkan percepatan relatif minimal 1,394 dicapai pada perkalian matriks 256 x 256 dan jumlah *thread* 1. Efisensi terendah 1,170 pada perkalian matriks 256 x 256 dan jumlah *thread* 4 sedangkan efisiensi tertinggi 6,672 pada perkalian matriks 1024 x 1024 dan jumlah *thread* 1.
3) Implementasi algoritma paralel dengan MPICH2 menghasilkan percepatan relatif maksimal 4,175 pada *sorting* data 50.000.000 *integer* dan jumlah proses 16 sedangkan percepatan relatif minimal 2,617 dicapai pada *sorting* data 50.000.000 *integer* dan jumlah proses 4. Efisiensi terendah 0,202 pada *sorting* data 200.000.000 dan jumlah proses 16 sedangkan efisiensi tertinggi 0,721 pada pada *sorting* data 25.000.000 dan jumlah proses 4.
4) Implementasi algoritma paralel dengan Cilk++ menghasilkan percepatan relatif maksimal 9,105 pada *sorting* data 25.000.000 dan jumlah *thread* 4 sedangkan percepatan relatif minimal 2,569 dicapai pada *sorting* data 25.000.000 dan jumlah *thread* 1. Efisensi terendah 2,569 pada *sorting* data

25.000.000 dan jumlah *thread* 1 sedangkan efisiensi tertinggi 9,105 pada *sorting* data 25.000.000 dan jumlah *thread* 4.

Dari hasil-hasil analisis di atas maka dapat ditarik kesimpulan sebagai berikut :

1) Algoritma paralel dengan MPICH2 baik untuk aplikasi perkalian matriks maupun *sorting* mampu meningkatkan performansi komputasi namun belum maksimal memanfaatkan *core processor* yang tersedia. Pada *cluster* PC *quadcore* yang terdiri dari 16 *core*, percepatan relatif maksimal yang dicapai baru 5,987 pada jumlah proses 16 atau 37,42% dari percepatan maksimal yang diharapkan. Sedangkan algoritma paralel dengan Cilk++ mampu meningkatkan performansi komputasi mendekati jumlah *core* yang ada. Pada PC *quadcore*, percepatan maksimalnya 3,766 pada jumlah *thread* 4 atau 94,15% dari percepatan yang diharapkan.
2) Kinerja program aplikasi yang diimplementasikan pada komputer paralel sangat dipengaruhi oleh algoritma paralel yang digunakan. Pada penelitian ini terbukti bahwa penggunaan algoritma paralel dengan MPICH2 kurang efisien bila dibandingkan dengan algoritma paralel dengan Cilk++ baik pada *cluster* PC *multicore* maupun PC *multicore* itu sendiri. Untuk aplikasi yang sama percepatan yang dihasilkan oleh algoritma paralel Cilk++ jauh melampui percepatan algoritma paralel MPICH2. Walaupun demikian Cilk++ masih mempunyai keterbatasan terutama karena lebih efektif bila digunakan pada PC *multicore* belum pada *cluster* PC.

5.2 Saran

Studi kinerja ini baru diterapkan pada PC *quadcore* dan *cluster* PC *quadcore* dengan aplikasi pengujian sederhana, hasilnya tentu belum bisa menjadi gambaran implementasi riil dari algoritma paralel MPICH2 dan Cilk++. Penelitian berikutnya hendaknya menggunakan aplikasi pengujian dengan operasi yang lebih kompleks sehingga memberikan gambaran sebenarnya dari pengaruh penggunaan algoritma paralel tersebut pada prosesor *multicore* seperti yang banyak diterapkan oleh para pengguna HPC.

Rencana awal penulis untuk menggabungkan dua algoritma paralel MPICH2 dan Cilk++ ternyata menghadapi banyak kendala sehingga tidak dapat terwujud

padahal hasil penelitian ini menunjukkan kinerja algoritma paralel Cilk++ sangat sesuai diterapkan pada PC *multicore* sehingga asumsi bahwa kinerja algoritma gabungan antara keduanya pada *cluster* PC *multicore* bisa meningkatkan performansi komputasi paralel secara lebih optimal merupakan keniscayaan. Karenanya perlu menjadi catatan untuk penelitian selanjutnya sehingga mendapatkan hasil yang lebih maksimal dan manfaat yang lebih luas bagi masyarakat umumnya dan pengembangan IPTEK khususnya.