ISSN 1979-4657

SOLUSI NUMERIK PERSAMAAN DIFUSI NEUTRON PADA TERAS REAKTOR NUKLIR DENGAN METODE ITERASI JACOBI PARALEL MENGGUNAKAN OPENMP

Frans Madah Basoaro Wau, Imam Taufiq dan Afdal
Program Pascasarjana, Jurusan Fisika FMIPA Universitas Andalas
Kampus Unand, Limau Manis, Padang, 25163
email : frans_nisel07@yahoo.com

ABSTRAK

Perhitungan Persamaan difusi neutron 2-dimensi telah berhasil diselesaikan secara paralel dengan menggunakan metoda Jacobi. Program ditulis dalam bahasa C++ menggunakan OpenMP. Dari proses *benchmarking* nilai fluks neutron dengan program FI-ITBCHI diperoleh bahwa kode program yang dibuat valid dengan akurasi hingga 6 angka penting. Hasil running program menunjukkan bahwa program yang dibuat bersifat *scalable*, baik pada prosesor *single-core* maupun *multi-core*. Nilai *speedup* tertinggi dicapai dengan menggunakan jumlah *thread* sama dengan jumlah *core* prosesor. Untuk prosesor dengan jumlah *core* 2 dicapai nilai *speedup* 1,28 sedangkan untuk *core* 4 nilai *speedup*-nya adalah 1,53.

Kata Kunci : Persamaan difusi, metoda Jacobi, komputasi paralel, OpenMP, *speedup*

ABSTRACT

The calculation of 2-dimensional neutron diffusion equation has been succesfully done parallely using Jacobi method. The program is written in C++ using OpenMP. As a result of benchmarking process of neutron flux values by FI-ITBCHI, the written program is valid within 6 significant figures. The result of the running program showed that the written program is scalable, both on single-core processors and multi-core processors. Highest speedup is achieved by setting the number of threads the same as the number of cores. For processor with 2 core, the achieved speedup is 1.28 and for 4 cores the speedup is 1.53.

Key words : diffusion equation, Jacobi method, parallel computation, OpenMP, speedup

1. PENDAHULUAN

Seiring dengan kemajuan teknologi dibidang mikroprosesor, komputer-komputer saat ini telah memiliki prosesor yang lebih dari satu atau yang lebih dikenal dengan istilah *multi-core processor*. Komputer dengan *multi-core processor* ini sangat mendukung kebutuhan sebuah komputasi yang rumit, dengan ketentuan algoritma program atau aplikasi tersebut haruslah berbasis paralel. Gagasan dasar dari pemrograman paralel ini adalah membagi permasalahan sehingga menjadi lebih kecil untuk dikerjakan oleh setiap prosesor yang ada dalam waktu yang bersamaan. Dengan begitu, waktu pengeksekusian akan lebih singkat.

Dalam analisis reaktor, perhitungan persamaan difusi neutron seringkali dilakukan secara berulang-ulang. Karenanya secara keseluruhan waktu perhitungan persamaan difusi neutron menjadi cukup lama. Dengan demikian percepatan perhitungan persamaan difusi ini akan sangat membantu mempercepat analisis reaktor secara keseluruhan. Untuk dapat memperoleh nilai distribusi fluks neutron dari persamaan difusi, maka geometri teras

reaktor akan dibagi menjadi bagian-bagian (*mesh*) kecil. Persamaan difusi ini akan diintegrasi dan didiskritisasi sehingga persamaannya akan menjadi bentuk matriks. Matriks inilah yang akan diselesaikan menggunakan iterasi Jacobi, dimana metode Jacobi merupakan metode iterasi yang mudah diparalelisasikan dibandingkan metode iterasi lainnya, dan OpenMP sebagai paralelisasinya.

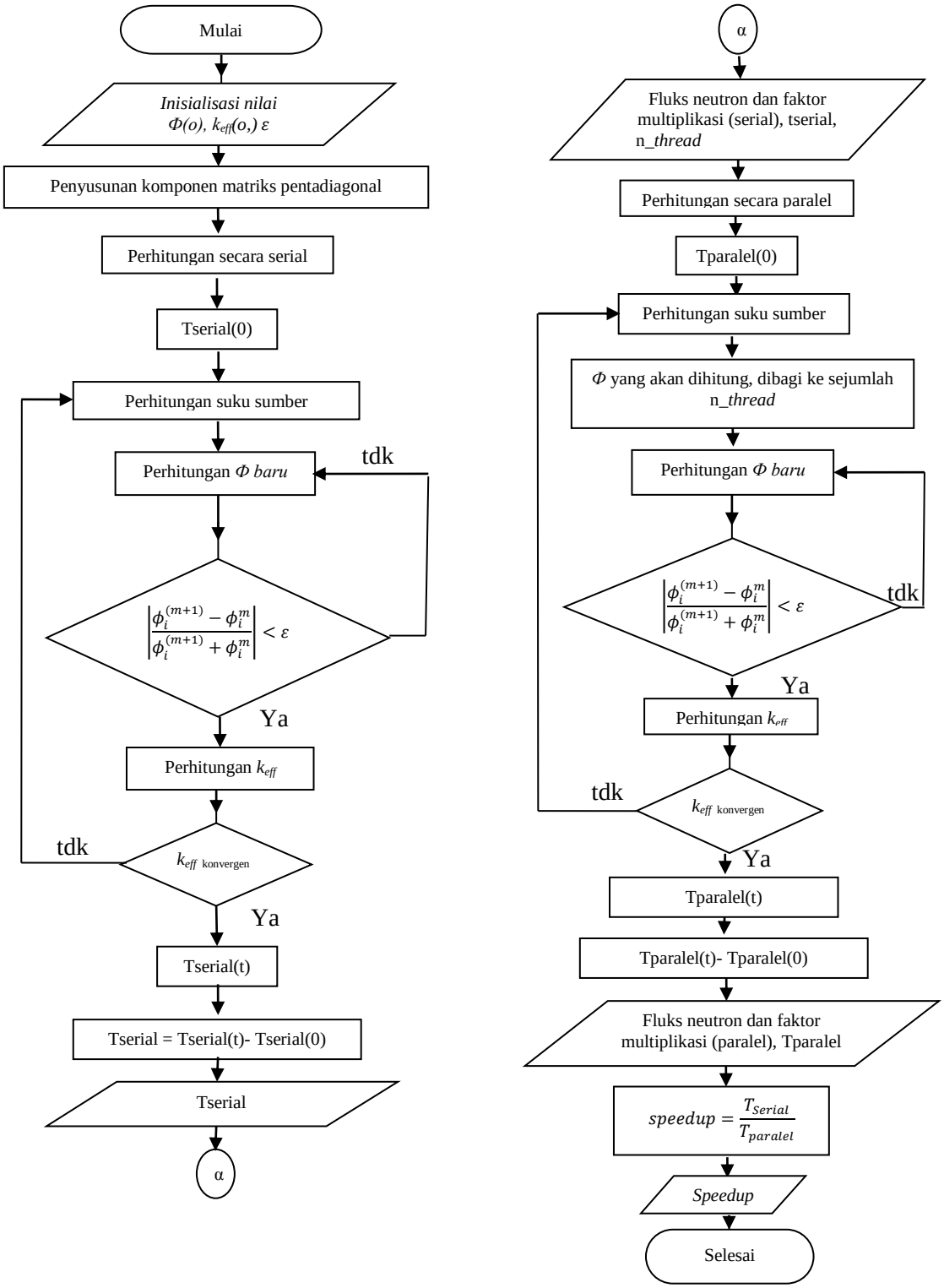
OpenMP (*Open Multi-Processing*) adalah program aplikasi antarmuka (*application programming interface*/API). OpenMP ini hanya sesuai untuk bahasa pemrograman C, C++ dan Fortran. OpenMP mendukung kegiatan yang penggunaan memorinya digunakan dalam waktu yang bersamaan (*shared-memory*).

Penelitian tentang komputasi parallel khususnya yang berhubungan tentang penelitian ini telah dilakukan antara lain, Su'ud, dkk (2001) yaitu perhitungan konstanta grup nuklir, dijalankan pada *cluster* multi-PC heterogen (tak-simetris) yang dihubungkan dengan *fast ethernet card* 100-*base* dan menggunakan PVM (*Parallel Virtual Machine*) pada *platform* Linux. Jenis prosesor yang digunakan adalah AMD K-6 dan Athlon dengan kecepatan *clock* CPU berkisar dari 300 MHz hingga 500 MHz. Penggunaan *fast ethernet* dengan kecepatan maksimum 100 Mbps masih mampu meningkatkan kinerja paralelisasi mengingat bahwa saat itu yang digunakan adalah prosesor dengan frekuensi *clock* CPU pada orde ratusan MHz, Taufiq (2010) melakukan paralelisasi perhitungan burnup bahan bakar pada reaktor cepat dengan bahan bakar UN-PuN (Uranium Nitrida dan Plutonium Nitrida) menggunakan pendingin Pb-Bi, dengan hasil bahwa dengan menggunakan Intel TBB (*Threading Building Block*) dan C++ telah berhasil dicapai *speedup* pada PC dengan prosesor *quad-core* sebesar 3,58 dan skalabilitas program dipenuhi. Sastri (2011) menyusun kode program perhitungan Persamaan difusi neutron 2-dimensi dengan metode Jacobi yang ditulis dalam bahasa C++ dan intel TBB, dengan hasil untuk prosesor intel i5 2320 3GHz, didapatkan *speedup* tertinggi sebesar 3,09, sedangkan untuk intel atom N550 didapatkan *speedup* tertinggi yaitu 1,51.

Tujuan dilakukannya penelitian ini adalah untuk membuat kode program solusi numerik Persamaan difusi neutron pada teras reaktor nuklir dengan metode iterasi Jacobi paralel menggunakan OpenMP, dan menganalisis tingkat ketelitian program dengan membandingkan hasil keluaran program dengan FI-ITBCHI yang sudah dianggap standar. Dengan dihasilkannya kode program ini, diharapkan dapat membantu mempercepat perhitungan Persamaan difusi neutron yang akhirnya juga dapat membantu perhitungan analisis reaktor nuklir.

2. METODE

Pada penelitian ini, perhitungan Persamaan difusi neutron dibatasi pada geometri teras berbentuk silinder dengan distribusi bahan bakar homogen konsentris. Dengan demikian persoalan geometri dapat direduksi menjadi 2-dimensi. Sedangkan tipe reaktor yang digunakan dalam perhitungan ini adalah reaktor cepat dengan bahan bakar UN-PuN. Program dibuat menggunakan bahasa pemrograman C++ pada *Microsoft Visual Studio* 2010 *Ultimate* dan OpenMP 2.0 yang telah terintegrasi di dalamnya. Program dijalankan pada komputer 2 *core* (intel *core* i3 3120M) dan 4 *core* (intel *core* i7 3770). Program di *benchmark* dengan program FI-ITBCHI. Diagram alir penelitian yang telah dilakukan seperti pada Gambar 1 berikut :



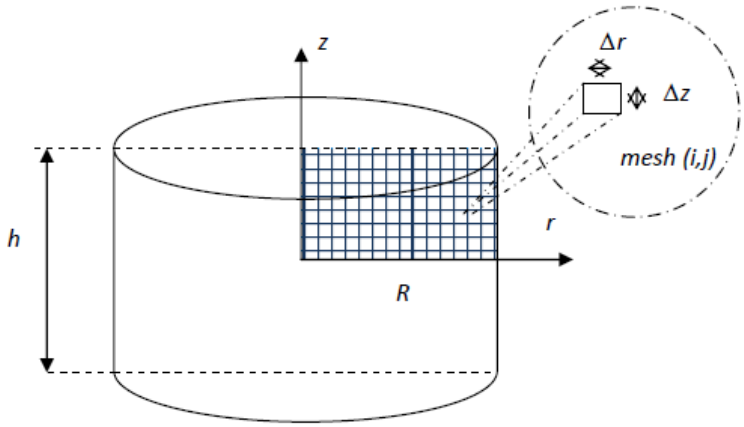
Gambar 1. Diagram alir perhitungan difusi neutron

Perhitungan Distribusi Fluks Neutron

Perhitungan distribusi fluks neutron akan dilakukan dengan metode penghampiran pada persamaan difusi. Geometri teras reaktor yang ditinjau berbentuk silinder sebagaimana tampak pada Gambar 2. Persamaan difusi neutron multigrup dapat dituliskan sebagai berikut (Stacey, 2001):

$$-\vec{\nabla}\cdot D_g(\vec{r})\vec{\nabla}\phi_g(\vec{r}) + \Sigma_{r,g}(\vec{r})\phi_g(\vec{r}) = \frac{\chi_g}{k_{eff}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r})\phi_{g'}(\vec{r})$$

$$+\sum_{g'=1}^{G}\Sigma_{s,g'\to g}(\vec{r})\,\phi_{g'}(\vec{r}) \qquad (1)$$



Gambar 2. Geometri teras reaktor berbentuk silinder

dengan indeks bawah g dan g' menunjukkan nomor grup neutron, χ_g adalah probabilitas bahwa neutron hasil fisi mempunyai energi dalam daerah grup g, D adalah tetapan difusi, ϕ adalah fluks neutron, Σ_r dan Σ_f masing-masing adalah tampang lintang *removal* makroskopik dan tampang lintang fisi makroskopik, k_{eff} adalah faktor multiplikasi efektif neutron. $\Sigma_{s,g'\to g}$bermakna probabilitas neutron dari grup g' terhambur ke grup g (dikenal dengan nama *group-transfer cross section*). Pada Gambar 2, jika diasumsikan distribusi bahan bakar bersifat simetris radial, maka perhitungan dapat dilakukan dengan menggunakan koordinat silinder 2-dimensi (r,z). Bagian teras reaktor yang ditinjau dibagi menjadi bagian-bagian kecil yang disebut *mesh* spasial yang posisinya ditandai dengan indeks (i,j). Dengan nilai i = 1, 2, ..., i-1, i dan j = 1, 2, ..., j-1, j. Persamaan (1) diintegrasikan terhadap ruang dan waktu menghasilkan Persamaan (2), (3), (4) dan (5).

$$-\gamma_{i,j}\phi_{g_{i,j-1}} + (\beta_{i,j} - \alpha_{1,j})\phi_{g_{1,j}} - \alpha_{2,j}\phi_{g,_{2,j}} - \gamma_{1,j+1}\phi_{g,_{1,j+1}} = S_{g,1,j} \qquad (2)$$

$$-\gamma_{I,j}\phi_{g,_{I,j-1}} - \alpha_{I,j}\phi_{g,_{I-1,j}} + \beta_{I,j}\phi_{g,_{I,j}} - \gamma_{I,j+1}\phi_{g,_{I,j+1}} = S_{g,I,j} \qquad (3)$$

$$-\alpha_{i,1}\phi_{g,_{i-1,1}} + \beta_{i,1}\phi_{g,_{i,1}} - \alpha_{i+1,1}\phi_{g,_{i+1,1}} - \gamma_{i,2}\phi_{g,_{i,2}} = S_{g,i,2} \qquad (4)$$

$$-\gamma_{i,J}\phi_{g,_{i,J-1}} - \alpha_{i,J}\phi_{g,_{i-1,J}} + \beta_{i,J}\phi_{g,_{i,J}} - \alpha_{i+1,J}\phi_{g,_{i+1,J}} - \gamma_{i,J+1}\phi_{g,_{i,J+1}} = S_{g,i,J} \qquad (5)$$

Kemudian Persamaan (2), (3), dan (4) dapat disusun ke dalam sebuah matriks *sparse* pentadiagonal **A** yang mengandung variabel (α, β, γ) yang berisi nilai fluks dan matriks **S** yang berisi suku sumber neutron, sehingga Persamaan difusi neutron dapat dituliskan sebagai Persamaan, yaitu:

$$A\Phi = S \quad (6)$$

Jika matriks A merupakan matriks non-singular maka penyelesaian Persamaan (6) adalah (Varga, 2009) :

$$\Phi = A^{-1} S \quad (7)$$

dengan A^{-1} adalah invers dari matriks A. Sedangkan matriks A dapat diurai (dekomposisi) menjadi penjumlahan dari matriks diagonal D, matriks segitiga-atas (*upper-triangular*) U, dan matriks segitiga-bawah (*lower-triangular*) L, seperti dituliskan di bawah ini.

$$A = D + U + L \quad (8)$$

Dengan demikian Persamaan (8) dapat dituliskan sebagai

$$\phi = -D^{-1}(U + L)\phi + -D^{-1}S \quad (9)$$

Persamaan iteratif dari Persamaan (9) dapat ditulis

$$a_{i,i}\phi_i^{m+1} = -\Sigma_{j=1}^{n} a_{i,j}\phi_j^m + S_i \, , 1 \leq i \leq n, m \geq 0 \quad (10)$$

dengan ϕ_i^0 adalah nilai tebakan awal. Kemudian Persamaan (10) dapat ditulis ke dalam bentuk

$$\phi_i^{m+1} = -\Sigma_{j=1}^{n} \frac{a_{i,j}}{a_{i,i}}\phi_j^m + \frac{S_i}{a_{i,i}} \, , 1 \leq i \leq n, m \geq 0 \quad (11)$$

Perhitungan iteratif seperti Persamaan (11) disebut sebagai metode Jacobi, atau dikenal dengan *total-step iterative method* (Varga, 2009). Persamaan in yang akan diparalelkan, dan dipakai untuk menyelesaikan persamaan difusi yang telah diubah dalam bentuk matriks.

Hukum Amdahl

Menurut Amdahl ada bagian kecil dari sebuah program yang tidak dapat lagi diparalelkan, akan membatasi peningkatan kecepatan yang dapat dicapai dari paralelisasi secara keseluruhan. Semua masalah mengandung bagian yang dapat diparalelkan dan bagian yang tidak dapat diparalelkan juga. Persamaan untuk mengetahui speedup yang didapatkan adalah :

$$SpeedUp = \frac{T_{serial}}{T_{paralel}} \quad (12)$$

dengan T_{serial} adalah waktu eksekusi program versi serial dan $T_{paralel}$ adalah waktu eksekusi program versi parallel.

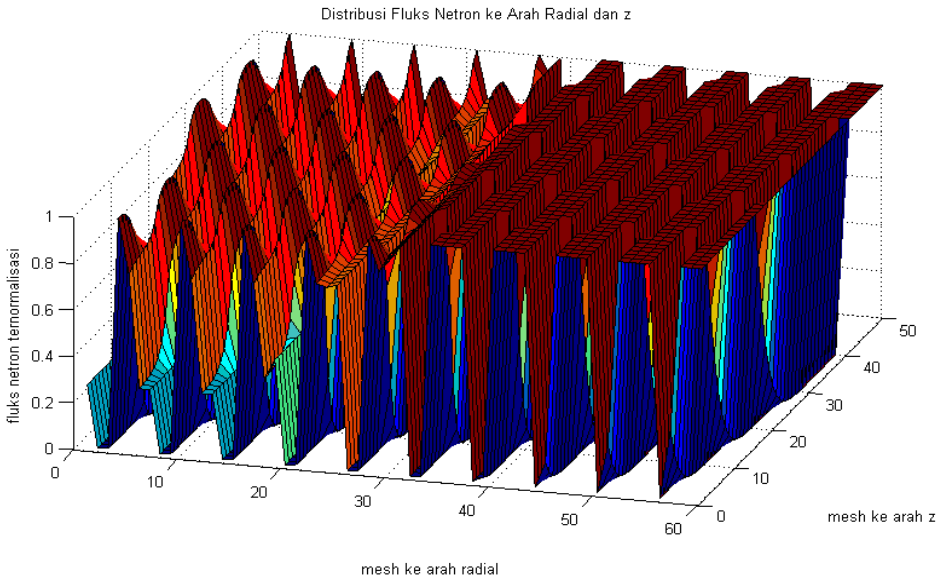
Komputasi Paralel

Komputasi paralel adalah proses atau pekerjaan komputasi di komputer dengan memakai suatu bahasa pemrograman yang dijalankan secara paralel pada saat bersamaan. Prinsip dari komputasi paralel adalah melakukan proses komputasi dengan menggunakan 2 atau lebih CPU/Processor dalam suatu komputer yang sama atau komputer yang berbeda (cluster of PCs). Setiap masalah dibagi kedalam beberapa instruksi kemudian dikirim ke prosesor yang tersedia dan eksekusi dilakukan secara serentak. Alasan utama dilakukannya paralelisasi adalah untuk mendapatkan waktu yang lebih singkat dalam menjalankan program. Penggunaan komputasi paralel merupakan pilihan yang cukup handal pada saat ini untuk tugas komputasi yang berat.

3. HASIL DAN DISKUSI

Hasil Benchmarking

Program yang ditulis terdiri atas bagian serial dan paralel. Masing-masing program mempunyai masukan dan keluaran yang sama. Hasil running program berupa distribusi fluks yang kemudian dinormalisasi, didapatkan grafik nya seperti pada Gambar 3.



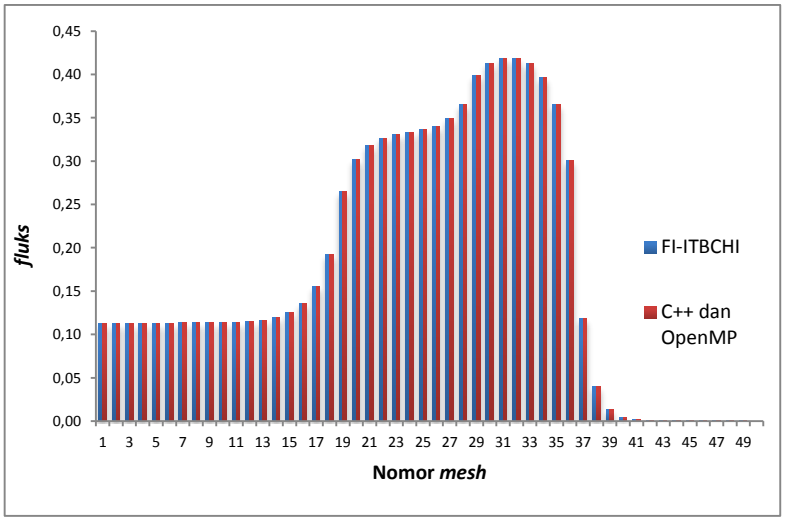
Gambar 3. Distribusi fluks neutron ke arah radial dan z

Hasil validasi program dengan program FI-ITBCHI yang dianggap menjadi standar dengan hasil perhitungan yang sesuai dengan batas konvergensi, dapat dilihat pada Tabel 1 dan Gambar 4.

Hasil keluaran program yang ditulis menggunakan C++ dan OpenMP ternyata memiliki tingkat kesamaan hingga 6 angka dibelakang koma dengan program FI-ITBCHI. Perbedaan yang paling mendasar adalah, untuk program FI-ITBCHI ditulis dengan menggunakan bahasa pemrograman Delphi dan menggunakan SOR sebagai metode iterasi dalam penyelesaian, sedangkan penulis menggunakan bahasa pemprograman C++ dan OpenMP serta menggunakan Jacobi sebagai metode iterasi.

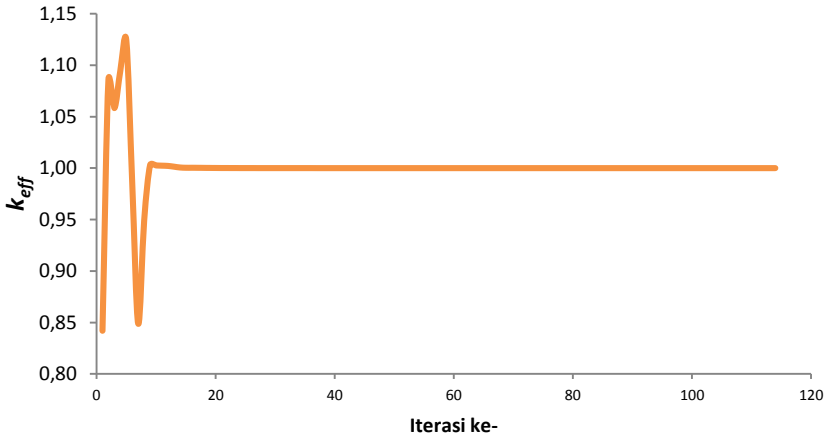
Tabel 1. Perbandingan nilai fluks untuk grup ke-8 pada arah-z

Fluks Neutron cepat (grup ke-8)					
No.	**C++ dan OpenMP**	**FI-ITBCHI dan Delphi**	**No.**	**C++ dan OpenMP**	**FI-ITBCHI dan Delphi**
1	0.1131181893	0.1131181435	**26**	0.3403146061	0.3403146714
2	0.1131195982	0.1131195768	**27**	0.3486065429	0.3486065167
3	0.1131228999	0.1131228965	**28**	0.3653142076	0.3653142086
4	0.1131292468	0.1131292354	**29**	0.398382275	0.398382209
5	0.1131409031	0.1131409131	**30**	0.4131312655	0.4131312977
6	0.1131621237	0.1131621326	**31**	0.4187450118	0.4187450215
7	0.1132008272	0.1132008675	**32**	0.4183964376	0.4183964865
8	0.1132717416	0.1132717865	**33**	0.4120964461	0.4120964876
9	0.1134023149	0.1134023124	**34**	0.3968014015	0.3968014243
10	0.1136438312	0.1136438653	**35**	0.3649431718	0.3649431987
11	0.1140923441	0.1140923857	**36**	0.3006734796	0.3006734153
12	0.1149281677	0.1149281893	**37**	0.1186721414	0.1186721657
13	0.11649048	0.116490421	**38**	0.03980231912	0.0398023187
14	0.119418455	0.119418497	**39**	0.01335426204	0.01335426143
15	0.1249185874	0.1249185853	**40**	0.004482038657	0.004482038875
16	0.1352716249	0.1352716121	**41**	0.001504763802	0.001504763176
17	0.1547948876	0.1547948164	**42**	0.0005053490706	0.0005053490153
18	0.191670834	0.19167089	**43**	0.0001697613581	0.0001697613985
19	0.2652663843	0.2652663865	**44**	0.00005704330857	0.0000570433678
20	0.3013861269	0.3013861153	**45**	0.0000191727196	0.0000191727053
21	0.3183256454	0.3183256545	**46**	0.000006445711853	0.000006445711856
22	0.326415672	0.326415665	**47**	0.000002167512122	0.000002167512942
23	0.3305578411	0.3305578673	**48**	0.0000007290398868	0.0000007290398841
24	0.3332280241	0.3332280013	**49**	0.0000002452652658	0.0000002452652854
25	0.3359726427	0.3359726415	**50**	0.00000008253002119	0.0000000825300534



Gambar 4. Perbandingan distribusi Fluks neutron pada C++ dan OpenMP dengan FI-ITBCHI

Persamaan difusi diselesaikan dengan menggunakan iterasi Jacobi dimana proses iterasi akan berhenti jika sampai pada batas konvergensi yaitu 10^{-6}. Ke konvergenan didapatkan setelah mencapai iterasi ke-114.



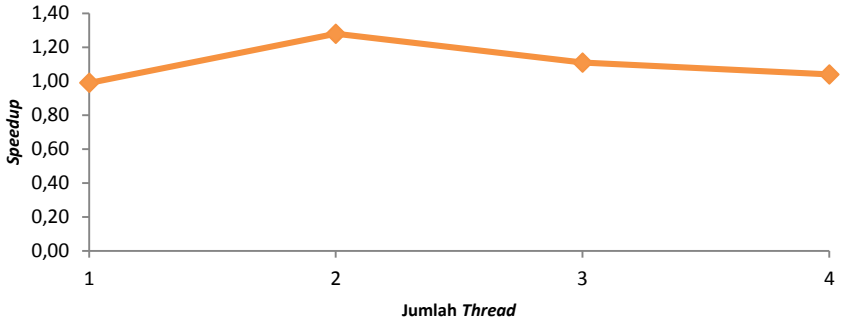
Gambar 5. Pola osilasi nilai keff hingga mencapai nilai konvergen

Hasil Analisis *Speedup*

Hubungan antara jumlah *thread* dan *speedup* yang didapatkan pada masing-masing komputer dapat dilihat di bawah ini.

Tabel 2. Perbandingan jumlah *thread* dengan *speedup* pada prosesor *dual core*

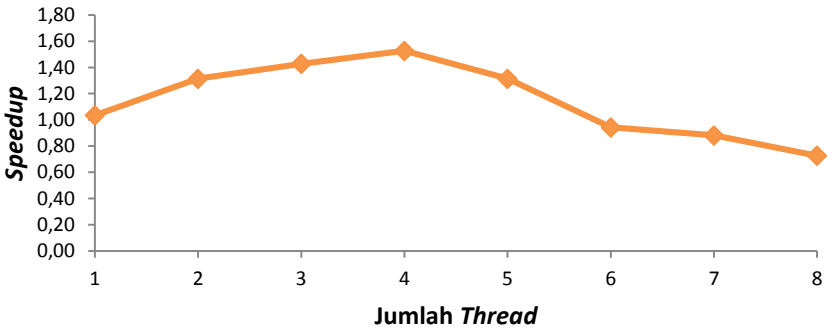
No.	Jumlah thread yang digunakan	Speedup yang dihasilkan
1	1	0,99
2	2	1,28
3	3	1,11
4	4	1,04



Gambar 6. Hubungan antara jumlah *thread* dengan *speedup* yang dihasilkan pada komputer dengan prosesor i3 3120M 2,5 GHz (2 *core*)

Tabel 3. Perbandingan jumlah *thread* dengan *speedup* pada prosesor *quad core*

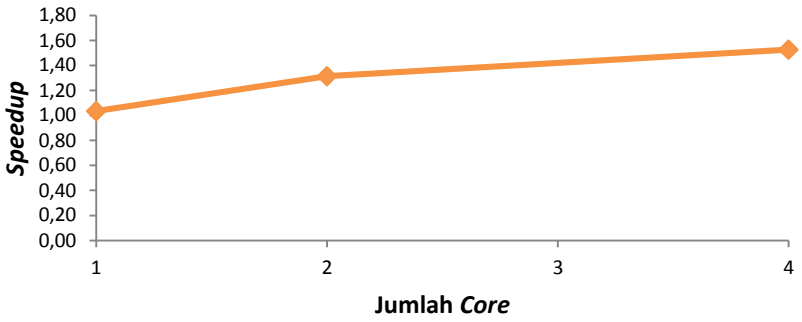
No.	Jumlah thread yang digunakan	Speedup yang dihasilkan
1	1	1,03
2	2	1,31
3	3	1,43
4	4	1,53
5	5	1,31
6	6	0,94
7	7	0,88
8	8	0,73



Gambar 7. Hubungan antara jumlah thread yang digunakan dan *speedup* yang dihasilkan pada komputer i7 3770 3,4 GHz (4 *core*)

Kedua komputer ini memiliki prosesor yang berkemampuan hyper-threading. Prosesor yang memiliki teknologi hyper-threading mampu menjalankan dua thread sekaligus dalam satu core. Sehingga misalnya untuk jenis prosesor dual core, prosesor tersebut mampu menjalankan 4 thread sekaligus. Itu alasan mengapa jumlah maksimal thread yang digunakan untuk masing-masing komputer ini adalah 4 dan 8 thread. Berdasarkan data di atas, untuk mendapatkan hasil speedup yang paling baik maka sebaiknya jumlah thread yang digunakan sama dengan jumlah prosesor yang dimiliki komputer tersebut.

Hubungan antara jumlah prosesor dengan speedup yang dihasilkan berdasarkan jumlah prosesor yang dimiliki sebuah komputer, dapat dilihat pada Gambar 8.



Gambar 8. Hubungan antara jumlah *core* dan *speedup* yang dihasilkan pada OpenMP

Tampak bahwa sifat skalabilitas dipenuhi, *speedup* meningkat dengan bertambahnya jumlah *core*. Dengan demikian diharapkan program ini dapat dijalankan dengan lebih cepat jika dipunyai prosesor atau PC dengan jumlah *core* yang lebih banyak. Saat ini telah dirilis produk prosesor dengan jumlah *core* 8 dan segera dirilis prosesor untuk server dengan jumlah *core* 16 yang mampu menjalankan 30 *thread* sekaligus .

4. KESIMPULAN

Kode program perhitungan persamaan difusi neutron telah berhasil disusun dalam bahasa C++ dan OpenMP. Hasil *benchmarking* dengan program FI-ITBCHI menunjukkan bahwa kode yang ditulis menghasilkan keluaran fluks yang valid dengan tingkat kesamaan hingga 6 angka penting. Nilai *speedup* tertinggi dicapai dengan menggunakan jumlah *thread* yang sama dengan jumlah *core* pada prosesor. *Speedup* tertinggi untuk komputer yang menggunakan prosesor i3 3120M 2,5 GHz (2*core*) sebesar 1,28, sedangkan untuk komputer yang menggunakan prosesor i7 3770 3,4 GHz (4 *core*) sebesar 1,53.

DAFTAR PUSTAKA

1. Stacey, W. M., 2001, *Nuclear Reactor Physics*, Jhon Wiley & Son Inc, Canada
2. Su'ud, Z., 2001, *Komputasi Paralel dalam Analisa Reaktor Nuklir*, Seminar Komputasi 2001, Bandung
3. Sastri, F., 2011, *Komputasi Paralel Persamaan Difusi Neutron Pada Reaktor Cepat Menggunakan Iterasi Jacobi*, Skripsi S1, Jurusan Fisika FMIPA UNAND, Padang
4. Taufiq, I., 2010, *Komputasi Paralel Persamaan Burnup Pada Reaktor Cepat dengan Pendingin Pb-Bi Menggunakan Pemrograman Multicore*, Disertasi S3, Jurusan Fisika FMIPA ITB, Bandung
5. Varga, R. S., 2009, *Matrix Iterative Analysis*, Springer Heidelberg Dordrecht London, New York