Indonesian Journal of Physics
Vol. 21 No. 1, January 2010

Parallel Burnup Analysis of Long-life Fast Reactors Using Multi-core Programming

Imam Taufiq[a,b] ,Zaki Su'ud[a], Abdul Waris[a], and Mitra Djamal[a]
[a]Physics Department, Bandung Institute of Technology, Jl. Ganesha 10, Bandung 40123, Indonesia.
[b]Physics Department, Andalas University, Kampus Limau Manis, Padang 25163, Indonesia.

Received 12 December 2009, Revised 2 January 2010, Accepted 12 January 2010

Abstract
A scalable parallel program for burnup analysis of long-life fast reactor has been succesfully built. The scalability test was conducted by comparing the program performance on dual-core and quad-core platforms. The best speedup gain were 1.72 on dual core 2.0 GHz, 3.58 on quad-core 2.4 GHz and 3.05 on quad-core 3.2 GHz. The slight drop of the speedup on quad-core 3.2 GHz could be well explained in connection with communication-time needed in running parallel version of the program.

Keywords: *Burnup, parallel, multi-core .*

1. Introduction

Burnup analysis for long-life nuclear reactors is relatively time-consuming calculation. Neutron diffusion equation coupled with burnup calculation which is executed repeatedly for 30 years of reactors life time need about 2.3 hours on 2.0 GHz processor of Desktop PC. The trend of computer's microprocessor development towards multi-core processors is both an opportunity and headache for computer programmer. The great potential of computational power exists in multi-core technology, but it can only be exploited using multi-core programming technique. Parallel programming is no longer optional because parallel computers have become personal for consumers. Now, parallel programming is mandatory for performance-sensitive applications in multicore processor envirnment.

Many paradigms of parallelization have been proposed and have been set in the form of new languages, language extensions, and libraries. One of such paradigm defines tasks that can optimally run in shared memory. This paradigm is well suited to achieving parallel speedup on multi-core chips. The key notion is to separate logical task patterns from physical threads, and to delegate task scheduling to the system. Threading Building Blocks is a library that supports scalable parallel programming using standard C++ code. It does not require special languages or compilers[1)].

The aim of this research is to produce a scalable parallel program for burnup analysis and analize its performance on dual-core and quad-core processors.

2. Parallel Computing

Parallel computing is basically based on the principle that large problems can often be divided into smaller ones, which are then solved concurrently. The way to analyze the problem into the serial and parallel parts is often called problem decomposition. There are two kinds of decompositions, functional decomposition and domain/data decomposition.

The potential speed-up of an algorithm on a parallel computing platform is given by Amdhal's Law, originally formulated by Gene Amdhal in the 1960s. It states that the portion of the program which cannot be parallelized will limit the overall speed-up available from parallelization. In this research, the actual speedup is the ratio of the running-time of serial version to the running-time of parallel version of the program on the same platform of parallel machine.

3. Neutron Diffusion Equation

For many aspect of nuclear reactor analysis, multigroup neutron diffusion approximation is sufficient to calculate the distribution of neutron on the core of reactors. In this research, two dimensional R-Z geometry 8-energy groups neutron diffusion calculation has been solved using parallel programming in multicore processor environment. Multigroup neutron diffusion equation, can be written as follows[2-4)]

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

where the subscript g and g' represent energy groups; D_g is diffusion constant of group-g; ϕ_g stands for neutron flux of group-g, and S_g is neutron source of group-g. Σ_{rg} represent the macroscopic removal cross section of group g, Σ_{sg} represent the macroscopic scattering cross section out of group-g and $\Sigma_{sg'g}$ represent the macroscopic scattering cross-section from group-g' to group-g.

In this research, the geometry of reactor core is chosen to be cylindrical with annular fuel arrangement. For this case, 2-dimensional diffusion

equation in cylindrical coordinate (r,z) is sufficient to describe the problem. After some rearrangement, equation (1) could be written as

$$\mathbb{A}\Phi = \mathbb{S} \qquad (2)$$

where Φ represent column matrix of ϕ_g, $\mathbb{S}$ is column matrix of S_g and $\mathbb{A}$ represent penta-diagonal matrix containing the rest variables and constants of diffusion equation for the 8-group of neutron[3]. Successive over relaxation (SOR) method is chosen to get the solution of equation (2).

4. Burnup Calculation

The depletion of fuel in reactors core could be described by basic equation that govern the burnup of nuclides[4]

$$\frac{dN_A}{dt} = -\lambda_A N_A - \left[\sum_g \sigma_{ag}^A \phi_g\right] N_A + \lambda_B N_B + \left[\sum_g \sigma_{cg}^C \phi_g\right] N_C \,. \qquad (3)$$

Equation (3) could be read as

$-\lambda_A N_A$ = loss due to radioactive decay of A
$-\left[\sum_g \sigma_{ag}^A \phi_g\right] N_A$ = loss due to neutron absorbtion by A
$\left[\sum_g \sigma_{cg}^C \phi_g\right] N_C$ = gain due to transmutation of C to A via neutron capture by C
$\lambda_B N_B$ = gain due to decay of B to A

where N_A is the density of nuclide A, λ_A is decay constant of nuclide A, σ_{ag}^A is microscopic absorption cross section of nuclide A, σ_{cg}^C is transmutation cross section for the production of nuclide A by neutron capture in nuclide C and λ_B is decay constant of nuclide B into nuclide A.

The depletion calculation was performed by using numerical method using variable time steps.

The reactor's core is divided into 6 regions along the r-axis and 6 regions along the z-axis. The fact that each mesh could be treated as mutually independent part to each other, inspiring the used of parallel techniques to calculate the burnup on each mesh.

5. Results and Discussion

In diffusion calculation, the SOR acceleration coefficient need to be optimized to get the fastest convergence time. Figure 1. shows that the best convegence condition achieved by setting the SOR acceleration coefficient about 1.35. This value will be used in the whole diffusion calculation for the rest of the research.

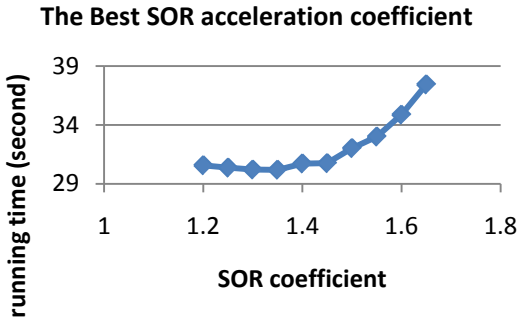

Figure 1. The best value of SOR acceleration coefficient for diffusion calculation is found to be 1.35

Stability of the reactors performance for the whole life-time could be shown from its k_{eff} values along the designated time. The fast reactor under consideration is designed to produce electricity power up to 30 years. In figure 2. it is shown that k_{eff} values are around the acceptable critical value for more than 30 years.

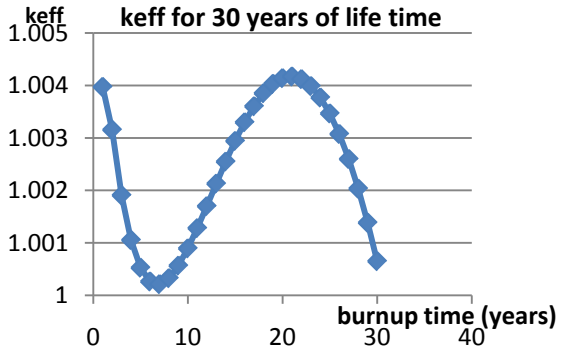

Figure 2. k_{eff} of the reactors calculated for 30 years of reactors life-time. The value swing between 1.0002 and 1.0042.

According to Amdhal's Law, the serial portion of the program will be the critical part that determine the speedup of the whole parts. In this case, the diffusion equation part of the calculation will be the main serial portion that affect the ultimate speedup gain for burnup analysis. By changing the portion of the parallel part of the whole calculation, it could revealed how the speedup of parallel burnup analysis depend on portion of the serial or parallel parts. Table 1. revealed the behavior of the program against the changing of burnup time-steps. On 2.4 GHz quad-core processor, the best speedup is 3.58 at 1/1000 time mesh scale.

Table 1. Comparison of running-time (Rt) of serial version and parallel version of the burnup analysis running on 2.4 GHz quad-core processor for several values of time-mesh scale of burnup's numerical calculation. The best speedup is achieved for 1/1000 time-mesh setup, that is 3.58.

Time-mesh scale	Rt Serial version	Rt Parallel version	Speedup
1	27.08894078 second	20.5734217 second	1.316695937
1/10	87.38172347 second	43.90326625 second	1.990323977
1/100	686.958958 second	253.691866 second	2.707847787
1/1000	6683.381458 second	1865.943597 second	3.581770354

To be scalable, a program have to pass the requirement that it could be run on different number of cores or processors or machines. The scalability assures that the result of the program will be the designated value independent of the platform of parallel computer used. From table 2., it could be concluded the program is relatively scalable. Figure 3., Give a better description of how program run on different platform, dual-core 2.0 GHz and quad-core 2.4 GHz.. The last two column of table 2., need more explanation.

Table 2. The scalability of the program could be drawn from the comparison of speedup on dual-core and quad-core platforms. Speedup on dual-core (2.0 GHz) and quad-core (2.4 GHz) is easily understandable. But slight decrease of speedup of quad-core (3.2 GHz) from speedup of quad-core (2.4 GHz) need more explanation.

	Rt Dual Core 2.0 GHz	Rt Quad Core 2.4 GHz	Rt Quad Core 3.2 GHz
Serial version	8178.376055 second	6683.381458 second	4573.861326 second
Parallel version	4750.268945 second	1865.943597 second	1498.996614 second
Speedup	**1.721665899**	**3.581770354**	**3.051281959**

On quad-core 2.4 GHz the resulting speedup is 3.58, but on a better processor, quad-core 3.2 GHz, the speedup is decreased by 0.5 points. To explain this result, it is better to refer to the running-time of the two cases. Actually, the running–time of the parallel version of the program on quad-core 3.2 GHz is faster (1.24 times) than on quad-core 2.4 GHz. But, the running-time of the serial version on quad-core 3.2 GHz far more faster (1.46 times) than on quad-core 2.4 GHz. Because speedup is calculated as the ratio of running-time of the serial version to running-time of parallel version on the same platform, the speedup on quad-core 3.2 GHz is lower than on quad-core 2.4 GHz. The parallel version needs some communication time between each parallel process running on each processor's cores, which is not exist in the serial version.

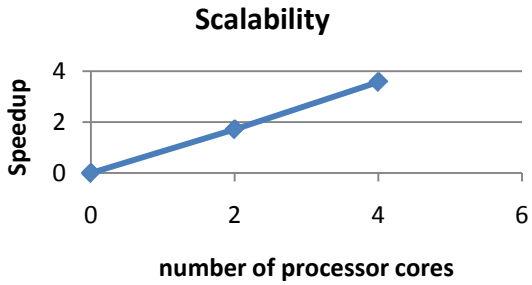

Figure 3. Scalability of the program on dual-core and quad-core processors. The speedup is 1.72 on dual-core (2.0 GHz) and 3.58 an quad-core (2.4 GHz).

6. Conclusion

A scalable parallel program for burnup analysis of long-life fast reactor has been built. By comparing the program performance on dual-core and quad-core platforms it is obtained that the program is

scalable. The best speedup gain were 1.72 on dual core 2.0 GHz, 3.58 on quad-core 2.4 GHz and 3.05 on quad-core 3.2 GHz. The slight drop of the speedup on quad-core 3.2 GHz could be well explained in connection with communication-time needed in running parallel version of the program.

Acknowledgments

Part of this research is funded by Directorate of Higher Education of the Ministry of National Education of the Republic of Indonesia by 'Hibah Penelitian Doktor ' grant 2009.

References

1. J. Reinders, *Intel Threading Building Blocks: Outfitting C++ for Multi-core Processor Parallelism*, O'Reilly, Cambridge, 2007, pp. 2-20.
2. J.J. Duderstadt and L.J. Hamilton, *Nuclear Reactors Analysis*, John Wiley and Sons, New York, 1976, pp. 25-30.
3. Z. Su'ud and H. Sekimoto, *Journal of Nuclear Science and Technology*, 32(9), pp. 834-845 (1995).
4. W. M. Stacey, *Nuclear Reactors Physics*, 2nd.ed., Wiley-VCH, Weinheim, 2007, pp.197-210.