2.9 MVP/GMVPの並列計算

長家康展、中川正幸、森貴正、佐々木誠*
日本原子力研究所、*（株）日本総合研究所

Parallel computing by Monte Carlo codes MVP/GMVP

Yasunobu Nagaya, Masayuki Nakagawa, Takamasa Mori, Sasaki Makoto*
Japan Atomic Energy Research Institute, *Japan Research Institute

General-purpose Monte Carlo codes MVP/GMVP are well-vectorized and thus enable us to perform high-speed Monte Carlo calculations. In order to achieve more speedups, we parallelized the codes on the different types of parallel computing platforms or by using a standard parallelization library MPI. The platforms used for benchmark calculations are a distributed-memory vector-parallel computer Fujitsu VPP500, a distributed-memory massively parallel computer Intel Paragon and a distributed-memory scalar-parallel computer Hitachi SR2201, IBM SP2. As mentioned generally, linear speedup could be obtained for large-scale problems but parallelization efficiency decreased as the batch size per a processing element(PE) was smaller. It was also found that the statistical uncertainty for assembly powers was less than 0.1% by the PWR full-core calculation with more than 10 million histories and it took about 1.5 hours by massively parallel computing.

1 はじめに

MVP/GMVP コード [1] はそれぞれ連続エネルギー、多群形式の汎用中性子・光子輸送計算コードで、既に炉心解析を始めとし、遮蔽、臨界安全、核融合中性子工学等の幅広い分野で用いられている。MVP/GMVP コードの最大の特徴のひとつは高ベクトル化効率を達成しているということである。これまで、計算時間、計算コストがかかるのはモンテカルロ計算の最大の欠点であったが、MVP/GMVP ではベクトル計算機の発達とともにそれに適したアルゴリズムである事象駆動型アルゴリズムを採用し、精度の良い解を実用的な時間で得ることが可能となった。従来のスカラーモンテカルロコードではヒストリー駆動型アルゴリズムを用いており、ベクトル計算機上では飛躍的な高速化は望めないのに対して、MVP/GMVP では事象駆動型アルゴリズムを改良したスタック駆動型領域選択アルゴリズム [3],[4] を用い、ほとんどの場合ベクトル化率 95%以上を達成している。その結果、ベクトル計算機上では従来のスカラーコードに比べて多くの場合 10 倍以上の高速化を実現し、世界最速の汎用モンテカルロコードとなっている。

従って、MVP/GMVP はベクトルスーパーコンピュータが発達すればするほどその性能を発揮することが可能であるが、ベクトルスーパーコンピュータの 1CPU 当たりの性能はほぼ限界に達していると言われており、スーパーコンピュータは並列化により性能アップを図る傾向がある。MVP/GMVP コードも更に性能を上げるには並列化に対応する必要がある。もともとモンテカルロ法では粒子を一つ一つ独立に取り扱うことができるので並列化に適しており、MVP/GMVP も並列処理を考えて開発を進めてきた。

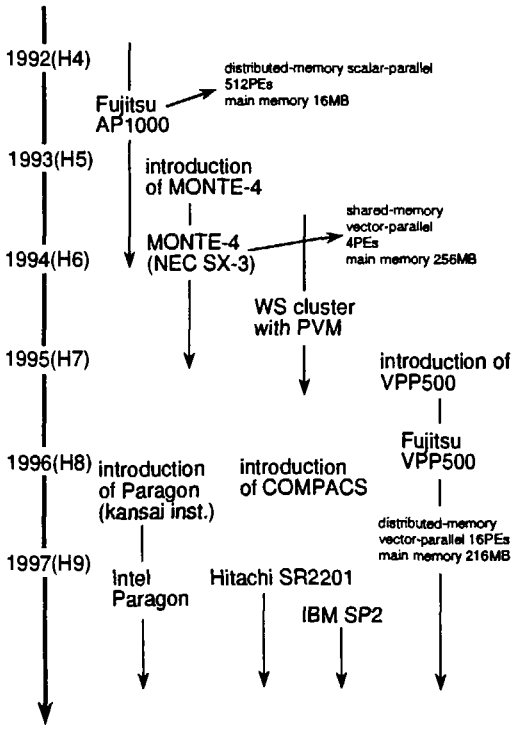


Fig.1 Parallelization of MVP/GMVP

Fig. 1は MVP/GMVP の並列化の歴史を振り返ったものである。並列化の初期の段階では GMVP を Fujitsu の超並列計算機 AP-1000 [5] に移植し、性能評価を行った。AP-1000 は MIMD アーキテクチャー、分散メモリー型の並列計算機で、RISC タイプの CPU を 512 ノード持っており、最大 512 ノードの並列計算が可能で

ある。GMVP ではAP-1000 上で512 ノード用いてNEACRP の3 次元中性子輸送計算ベンチマーク [8] を解き、約100%の並列化効率を達成したが [6],[7]、1 ノード当たり16MB のメモリー量では連続エネルギーコードを実用的に用いるにはまだまだ不十分であった。

このころ、ハードウェアの開発も盛んに行われ、原研では超高速モンテカルロ装置MONTE-4 が開発された。[9] MONTE-4 は共有メモリー型のベクトル並列計算機で4 つの計算ノードを有している。MONTE-4 上でMVP/GMVP コードのベクトル並列計算を行ったところ、高速炉の燃料集合体計算に対して、並列化による高速化倍率3.8、ベクトル並列化による高速化倍率18.7 が得られ、ベクトル並列計算機に対してもMVP/GMVP は非常に有効であることが示された。[6], [7]

しかしながら、AP-1000、MONTE-4 とも並列化するためにそれぞれ独自の並列化プログラミングを行う必要があり、移植性が悪かった。そこで、標準並列化ライブラリーPVM[10] を用いたワークステーションクラスターでの並列化も行った。ワークステーションクラスターではそれぞれのワークステーションの間をイーサネットでつなぐので、並列化効率は悪くなりがちであるが、4 台のワークステーションを用いて、3.7~4.0 倍の高速化倍率を得た。[6],[7]

最近では、最大16 のプロセッシングエレメント (PE) を使えるベクトル並列型のFujitsu VPP500 や、1PE 当たり128MB のメモリーを所有し、最大800PE まで使える分散メモリー型のIntel Paragon、最大56PE まで使える分散メモリー型のHitachi SR2201、最大48PE まで使える分散メモリー型のIBM SP2 が導入され、これらの並列処理プラットフォームにもMVP/GMVP の移植を行った。これらプラットフォーム上での並列化はVPP を除いてすべて標準並列化ライブラリーMPI[11], [12] を用いて行った。本研究ではこれらプラットフォームにおけるMVP/GMVP のベンチマーク計算を行い、大規模モンテカルロ計算の可能性を調べた。

2 MVP/GMVP コードの並列化手法

Fig. 2 にMVP/GMVP コードにおける並列処理の流れを示す。この並列処理の流れはどのプラットフォームにおいてもほとんど変わらない。ただ、各タスクを起動して並列処理を行う際にマスター/スレーブ型 (AP-1000) であるか対称並列型 (PVM,Paragon) であるかという違いがあるくらいである。まずタスク数を設定し、入力データを読み込む。その後、固定源問題では1 タスク当たりに処理するバッチ数、固有値問題では1 タスク当たりに処理するバッチサイズ (1 バッチ当たりのヒストリー数) を決定する。それから、乱数の種を発生、各タスクを起動し、データを送

信したのち、ランダムウォークを各タスクにおいて行う。全タスクでランダムウォークが終了すると、各タスクの計算結果を受信し、統計処理をする。

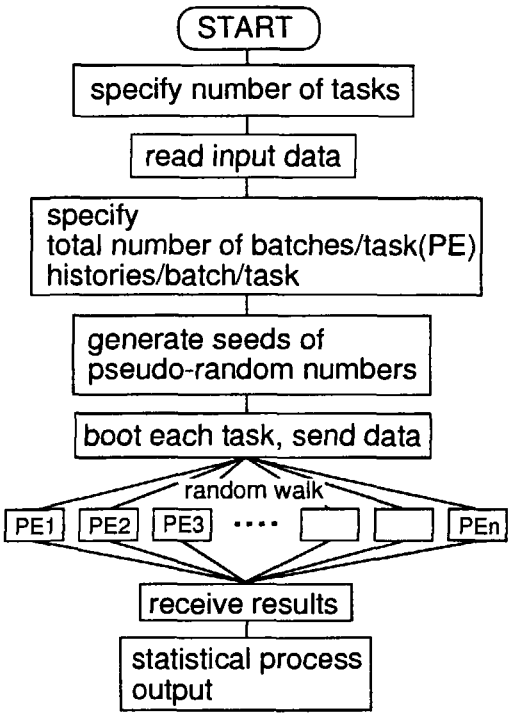


Fig.2 Flowchart of parallel processing

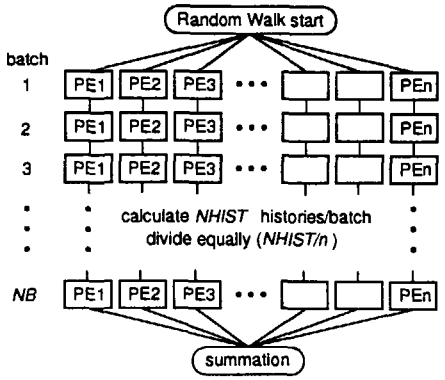


Fig.3 Parallel processing for eigenvalue problems

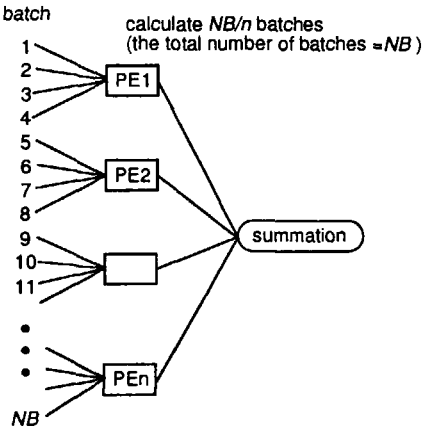


Fig.4 Parallel processing for fixed-source problems

2.1 固有値問題における並列処理

原子炉の炉心計算、臨界安全の問題などではしばしば実効増倍率、即ち固有値を求める必要がある。モンテカルロ法による固有値計算は総粒子数をあるバッチ数(NB)に分割し、各バッチで求められた固有値をすべてのバッチについて統計処理することによって行う。このとき、各バッチの粒子源となるのは前の世代で核分裂を起こした位置である。最初のバッチは核分裂源分布が分かっていないので何らかの仮定をして与える。従って、核分裂源が十分収束するように適当なバッチ数をとってやる必要があり、各バッチ独立に計算することはできない。

この固有値計算を並列処理で行うにはFig. 3に示すように、通常バッチサイズをPE数で分割する。この場合、核分裂源の処理に2通りの方法が考えられる。1つは各バッチの計算が終わる毎に核分裂源の情報を集め、次に世代ではその情報をまた各PEに分配して固有値計算を進める方法で、もうひとつは各PE毎に固有値計算を行い、最終バッチが終わった時点で各PEのデータを集計する方法である。前者の方法では1バッチ毎に通信する必要があるので、計算時間がかかるという欠点がある。また、後者の方法では各PE当たりのバッチサイズが小さくなり、計算された実効増倍率にバイアスがかかる可能性がある。MVP/GMVPでは後者の方法を採用しており、今回のベンチマーク計算では実効増倍率にバイアスがどうかも調べた。

2.2 固定源問題における並列処理

Fig. 4に固定源問題での並列処理を示す。固定源問題の場合は核分裂位置を考慮する必要はなく、一つ一つの粒子は固定源より独立に発生するので、固有値問題よりも並列処理は幾分簡単である。MVP/GMVPでは総粒子数をバッチ数(NB)に分割し、更にバッチ数を各PE数(n)に分割して並列処理を行う。従って、各PEでNB/nバッチずつ処理していくことになる。

3 計算結果

3.1 VPP500での並列計算

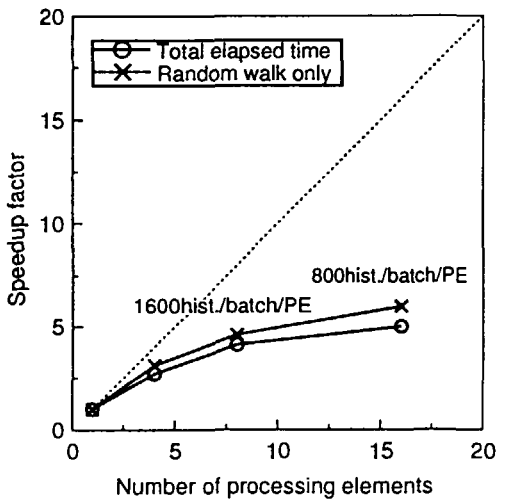


Fig.5 Efficiency of vector-parallel processing on VPP500 for an LMFBR assembly problem

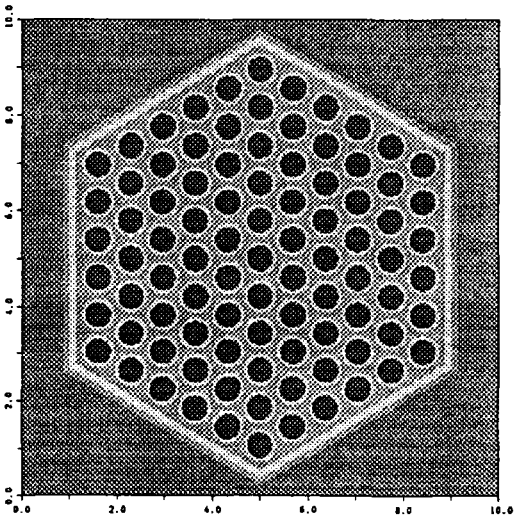


Fig.6 Geometry for a LMFBR assembly (91pins)

Fig. 5 に VPP500 上で固有値計算を行った結果を示す。計算体系は Fig. 6 に示すような高速炉燃料集合体が無限に配置された問題で、総ヒストリー数 640,000、バッチサイズ 12,800 で計算を行った。ベクトル並列計算の固有値計算ではバッチサイズ小さくなる、即ちベクトル長が短くなるために、PE 数に比例して計算速度が上がるということはなく、図に示すようになだらかな曲線となる。この場合、16PE を用いて約 5 倍の高速化率しか達成されていないが、もっとバッチサイズを大きくすることにより、効率の良い計算が可能であろう。

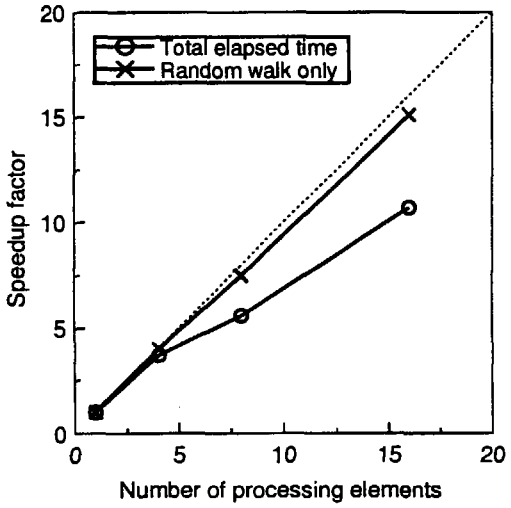


Fig.7 Efficiency of vector-parallel processing on VPP500 for a FNS bulk shielding problem

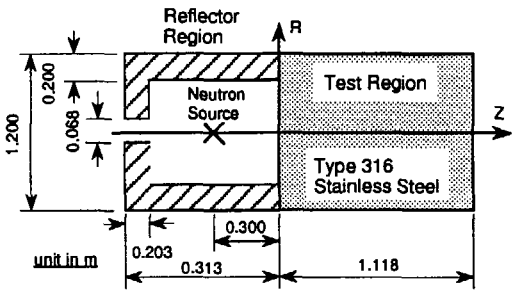


Fig.8 Geometry for a FNS bulk shielding experiment

次に固定源問題に対する VPP500 上での並列計算を Fig. 7に示す。計算に用いた体系は Fig. 8に示すような FNS バルク遮蔽実験の体系で、中性子光子結合問題である。総ヒストリー数 1,280,000、バッチサイズ 10,000 で計算を行った。固定源問題では固有値問題のようにベクトル長が短くなることはないので、ランダムウォークの部分ではほぼ理想的に並列化されている。しかし、全体の計算時間で見ると、16PE を用いても 10 倍程度のしか高速化されていない。この原因のひとつにデータ転送のオーバーヘッドが考えられるが、8PE, 16PE の場合に極端にデータ転送に時間がかかっており、システム固有の問題があるものと思われる。

3.2 Paragon での並列計算

Paragon で先ほどの無限高速炉燃料集合体問題に対して 2 つのケースについて PE 数を変えて並列計算した結果を Fig. 9、Fig. 10に示す。それぞれ総ヒストリー数 512,000、バッチサイズ 12,800、及び総ヒストリー数 6,400,000、バッチサイズ 128,000

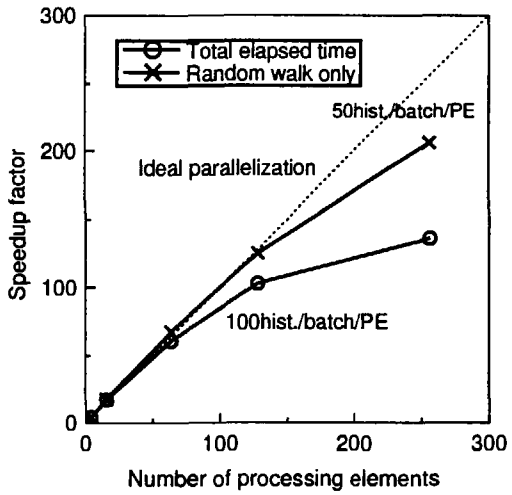


Fig.9 Efficiency of massive parallel processing on Paragon for an LMFBR assembly problem (case 1)

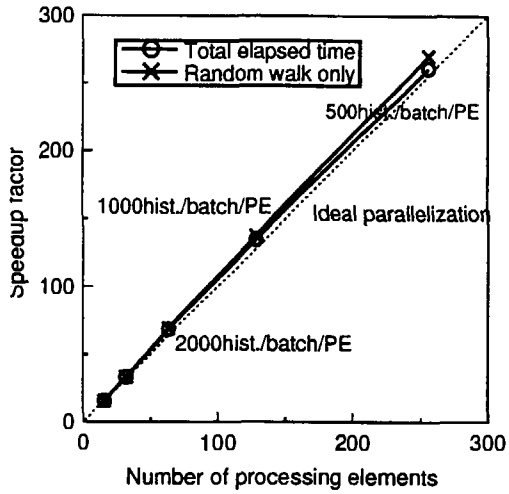


Fig.10 Efficiency of massive parallel processing on Paragon for an LMFBR assembly problem (case 2)

で計算を行った。バッチサイズの小さく、規模の小さい問題では100PEをこえたあたりから並列化効率は悪くなり、256PEでは50%程度まで下がってくる。また、256PEではランダムウォークの部分に対しても並列化効率が下がっており、各PEのアイドリング時間の影響があるものと思われる。それに対して、バッチサイズが大きく、規模の大きい問題ではほぼ理想的に並列化されている。これらのグラフより、高並列になるほど粒度(1PE当たりのバッチサイズ)が小さくなり、並列化効率が悪くなるのが分かる。

Table 1 Dependence of k_{eff} on granularity (Paragon)

PEs	histories/batch/PE	k_{eff}	1σ
4	32,000	1.36093	0.0082%
16	8,000	1.36086	0.0062%
32	4,000	1.36084	0.0068%
64	2,000	1.36088	0.0063%
128	1,000	1.36107	0.0072%
256	500	1.36102	0.0070%

Number of total histories = 6,400,000

batch size = 128,000

Table 1は無限高速炉燃料集合体問題を総ヒストリー数6,400,000、バッチサイズ128,000で計算を行ったときの、実効増倍率の粒度依存性を示している。1σの範囲では一致しないものもあるが、実効増倍率の粒度依存性は見られなかった。

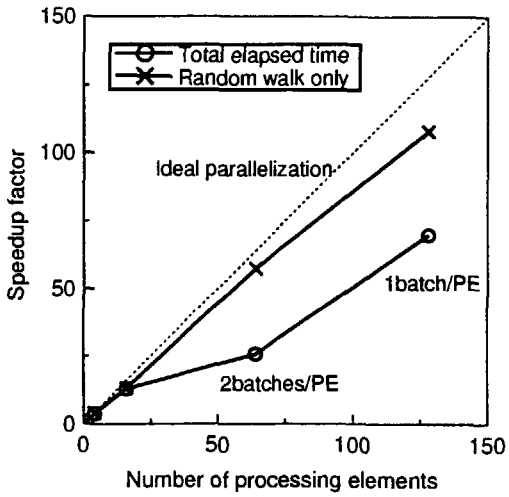


Fig.11 Efficiency of massive parallel processing on Paragon for an FNS bulk shielding problem (case 1)

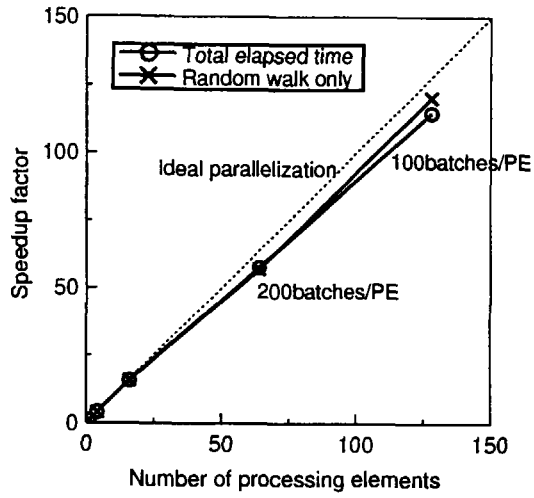
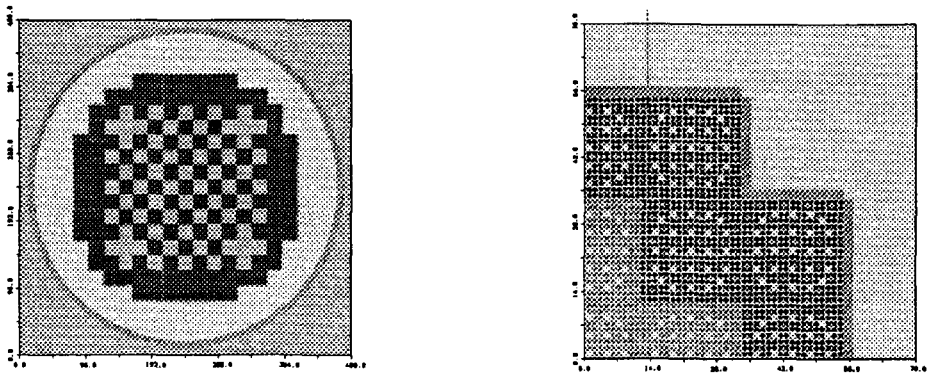


Fig.12 Efficiency of massive parallel processing on Paragon for an FNS bulk shielding problem (case 2)

Fig. 11、12は Fig. 8の FNS バルク遮蔽問題 (固定源問題) を Paragon で並列計算を行ったときの並列化効率を示した図である。それぞれ、総ヒストリー数 1,280,000、バッチサイズ 10,000 の小規模な計算、及び総ヒストリー数 6,400,000、バッチサイズ 500 の比較的大規模な計算である。小規模計算の方では 64PE 以上になると、とたんに並列化効率が悪くなったが、大規模計算の方ではほぼ理想的な並列化効率が得られた。これより固定源問題においても、効率の良い計算をしようとするなら、ある程度規模の大きな計算をしなければならないことが分かった。

3.3 SR2201 での並列計算



(a) Full-core model　　(b) Closeup view

Fig.13 Calculation model for PWR full core.

次にFig. 13に示すようなPWR全炉心体系について並列計算を行った。Fig. 14はSR2201でPWR全炉心体系について、総ヒストリー数1,280,000、バッチサイズ12,800で並列計算した結果である。問題の規模が小さいため、40PE用いても50%程度の並列化効率しか得られなかった。現在のところ、計算機資源の関係でこれ以上大規模、即ちヒストリー数の大きい問題を解くのは難しいが、そのような場合にはもっと並列化効率がよくなるだろう。

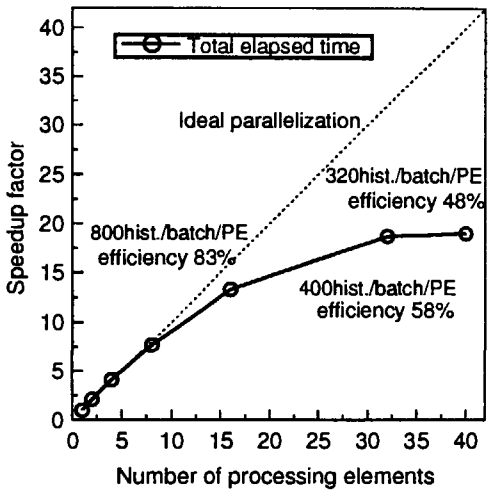


Fig.14 Efficiency of parallel processing on SR2201 for a PWR full-core problem

Table 2 Dependence of k_{eff} on granularity (SR2201)

PEs	histories/batch/PE	k_{eff}	1σ
1	12,800	1.00671	0.047%
2	6,400	1.00726	0.047%
4	3,200	1.00808	0.044%
8	1,600	1.00752	0.039%
16	800	1.00715	0.039%
32	400	1.00712	0.040%
40	320	1.00755	0.036%

Number of total histories = 1,280,000

batch size = 12,800

Table 2はPWR全炉心問題を総ヒストリー数1,280,000、バッチサイズ12,800で計算を行ったときの、実効増倍率の粒度依存性を示したものである。Paragonの場合と同様、実効増倍率の粒度依存性は見られないことが分かる。

3.4 SP2での並列計算

IBM SP2においても同じPWR全炉心体系について並列計算を行った。Fig. 15はSP2で総ヒストリー数1,280,000、バッチサイズ12,800で並列計算した結果である。SP2の場合も問題の規模が小さいため、40PE用いても64%程度の並列化効率しか得られなかったが、SR2201よりは良い結果が得られた。おそらくこれはPE間の結合方式やデータ転送速度の違いによるものと思われる。

Table 3は PWR 全炉心問題を総ヒストリー数1,280,000、バッチサイズ12,800で計算を行ったときの、実効増倍率の粒度依存性を示したものである。Paragon, SR2201 の場合と同様、実効増倍率の粒度依存性は見られないことが分かる。

Table 3 Dependence of k_{eff} on granularity (SP2)

PEs	granularity	k_{eff}	1σ
1	12,800	1.00750	0.042%
2	6,400	1.00696	0.043%
4	3,200	1.00772	0.046%
8	1,600	1.00762	0.045%
16	800	1.00746	0.040%
32	400	1.00695	0.045%
40	320	1.00697	0.038%

Number of total histories = 1,280,000
batch size = 12,800

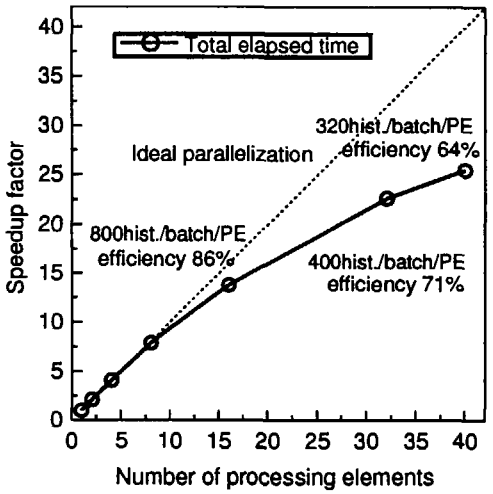


Fig.15 Efficiency of parallel processing on SP2 for a PWR full-core problem

3.5 Paragon における全炉心計算

以上に示したように SR2201, SP2 では計算機資源の関係上、PWR 全炉心問題について大規模な並列計算を行うことは不可能であった。そこで、Paragon において同様の問題に対して大規模な並列計算を行った。総ヒストリー数は12,800,000、バッチサイズは 128,000 で、512PE を用いて並列計算したときの結果を Table. 4に示す。

Table 4 PWR full-core calculation on Paragon

Location	0(core center)	1	7(core edge)
Enrichment	2.1%	2.6%	3.1%
	Assembly Power		
Power($\times 10^{-5}$)	257	295	131
1σ	0.7%	0.9%	0.7%
	Pin Power		
Power($\times 10^{-5}$)	1.06	1.18	1.04
1σ	4.4%	4.0%	6.0%

Total fission source is normalized to unity.

この表から分かるように、約1300万ヒストリーの全炉心計算で集合体出力は1%以内で、燃料棒出力は中心領域で約 4%、炉心外側領域で約 6%で計算できることが分

かった。この計算に要した時間は約1.5時間で、更に計算を進めることにより、全炉心計算で燃料棒出力を1%以内で求めることは十分可能であると思われる。

4 まとめ

MVP/GMVPコードを高速化するため、様々な並列計算プラットフォーム(VPP500、Paragon、SR2201、SP2)において並列化を行った。また、移植性を向上させるため、標準並列化ライブラリーMPIを用いて並列化を行った。(Paragon、SR2201、SP2)今回のベンチマーク計算においても、一般的に言われているように大規模計算ではどのプラットフォームにおいてもほぼPE数に比例するような並列化効率が得られたが、小規模計算ではデーター転送のためのオーバーヘッド及びアイドリング時間により並列化効率の悪化が見られた。

VPP500のようなベクトル並列計算機での固有値計算では、PE数が大きくなると1PE当たりのバッチサイズが小さくなり、ベクトル長が短くなるので、PE数に比例した並列化効率は得られない。固定源問題ではこのようなことは起こらないが、問題の規模が大きくなければ並列化効率はよくならない。

Paragon, SR2201, SP2のようなスカラー計算機においても、固有値計算では1PE当たりのバッチサイズが小さくなると必然的に問題の規模が小さくなり、並列化効率が悪くなる。従って、固有値問題の並列計算ではバッチサイズの最適化が重要である。

MVP/GMVPの並列計算ではバッチ毎に核分裂源を集めない方法を採用しているが、今回の固有値ベンチマーク計算において実効増倍率の粒度依存性は見られなかった。

Paragon上で1000万ヒストリー以上のPWR全炉心計算を行った結果、現実的な時間(2,3時間程度)で集合体出力は1σが0.1%以下の統計誤差で、燃料棒出力は中心領域で約4%、炉心外側領域で約6%の統計誤差で求められることが分かった。

参考文献

[1] 森、中川：MVP/GMVP連続エネルギー法及び多群法に基づく汎用中性子・光子輸送計算モンテカルロコード、*JAERI-Data/Code* 94-007 (1994)

[2] Mori, T., et. al. : "Reactor Engineering Department Annual Report", *JAERI-M* 92-125, 28 (1992)

[3] Mori, T., Nakagawa, M., and Sasaki, M. : "Vectorization of Continuous Energy Monte Carlo Method for Neutron Transport Calculation", *J. Nucl. Sci. Technol.* **29**, 183 (1992)

[4] Nakagawa, M., Mori, T., and Sasaki, M. : "Comparison of Vectrization Methods Used in a Monte Carlo Code", *Nucl. Sci. Eng.* **107**, 58 (1991)

[5] H. Ishihata, et. al. "An architecture of highly parallel computer AP-1000", *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, May 1991

[6] 中川、森、佐々木、鈴木：モンテカルロコード GMVP, MVP による並列計算、原子力学会 1994 春の大会

[7] Sasaki, M., Nakagawa, M., and Mori, T. : "An application study of parallel processing to the particle transport simulation", *Comp. Assisted Mech. and Eng. Sci.*, **1**, 177-189 (1994)

[8] Takeda, T. and Ikeda, H. : "3-D NEUTRON TRANSPORT BENCHMARKS", NEACRP-L-330 (1991)

[9] Asai, K., Higuchi, K., et. al. : "The JAERI Monte Carlo Machine", *Proc. Mathematical Methods and Supercomputing in Nuclear Applications*, 341, Karlsruhe, Apr 1993

[10] V. Sunderam. : "PVM : A framework for parallel distributed computing", *Concurrency : Practice and Experience*, **2**, No. 4, Dec 1990

[11] MPI Forum ; "MPI : A Message-Passing Interface Standard", June 12, 1995

[12] MPI Forum ; "MPI-2 : Extensions to the Message-Passing Interface", July 18, 1997