研　究　速　報

研 究 速 報

流体計算における計算機性能の比較

Comparision of Computer Performance on the Flow Calculation

伊　藤　裕　一*・谷　口　伸　行**・小　林　敏　雄***

Yuichi ITOH, Nobuyuki TANIGUCHI and Toshio KOBAYASHI

1. は　じ　め　に

数値シミュレーションにおいて実行するコンピュータの計算速度は，より大きな問題，より複雑な問題へ適用する上で非常に重要である．特に乱流の数値計算においては長時間の演算が不可欠であり，計算機の演算性能と記憶容量は常に制約となっている．

この様な環境の中，実際にシミュレーションプログラムを製作・実行するユーザにとって，適切な計算機環境を選んでいるか，また，使用した計算機が十分に性能を発揮しているかどうかを調べることは非常に重要となる．しかし，計算機性能を評価するデータには，計算機開発者の視点から示されたものが多く，利用環境によっては実際にそぐわないケースも見られ，数値シミュレーションなどのアプリケーションソフトウェア実行性能を推定するには必ずしも利用しやすいとは言えない．

一方で，近年の計算機環境の発達に伴い，乱流の数値計算の工学的応用は，レイノルズ平均乱流モデル（RANS：Reynolds Averaged Navier-Stokes）に基づく3次元定常解析から，乱流が本質的に持つ非定常現象を直接的に扱いうるラージ・エディ・シミュレーション　（LES: Large Eddy Simulation）によるシミュレーションに移行しつつあるが，LES計算ではRANS計算より格子点数を多くとる必要があるので，計算負荷が大きな問題となってくる．

そこで本報では，流れ場数値シミュレーション研究用として当研究室で実際に使用しているLESプログラムと小規模なベンチマークプログラムを，各種PC，EWS，スーパーコンピュータでテストし，その計算速度の比較を行なう．

*東京大学生産技術研究所　人間・社会大部門
**東京大学情報基盤センター
***東京大学生産技術研究所　情報・システム大部門

2. 支　配　方　程　式

非圧縮性流体の支配方程式は以下のような運動方程式と連続の式である．

運動方程式：

$$\frac{\partial u_i}{\partial t}+\frac{\partial u_i u_j}{\partial x_j}=-\frac{\partial p}{\partial x_i}+\frac{\partial}{\partial x_j}\left(\nu\frac{\partial u_i}{\partial x_j}\right) \quad \cdots\cdots(1)$$

連続の式：

$$\frac{\partial u_i}{\partial x_i}=0 \quad \cdots\cdots(2)$$

LESの基礎方程式を得るために，これらの式にフィルタリング操作を施し，SGS成分のモデルとしてSmagorinskyモデルを用いると以下のようになる．

運動方程式：

$$\frac{\partial \bar{u}_i}{\partial t}+\frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j}=-\frac{\partial \bar{p}}{\partial x_i}+\frac{\partial}{\partial x_j}\left(\nu\frac{\partial \bar{u}_i}{\partial x_j}-\tau_{ij}\right) \quad \cdots\cdots(3)$$

$$\tau_{ij}\equiv\overline{u_i u_j}-\bar{u}_i\bar{u}_j=-2\nu_{SGS}\overline{S_{ij}}, \quad \cdots\cdots(4)$$

$$\nu_{SGS}=(C_S\Delta)^2|\bar{S}| \quad \cdots\cdots(5)$$

圧力方程式:

$$\frac{\partial^2 p}{\partial x_i \partial x_j}=-\frac{\partial D}{\partial t}+\frac{\partial^2}{\partial x_i \partial x_j}\left(\nu\frac{\partial u_i}{\partial x_j}\right)-\frac{\partial^2 u_i u_j}{\partial x_i \partial x_j} \quad \cdots\cdots(6)$$

$$D\equiv\frac{\partial u_i}{\partial x_i} \quad \cdots\cdots(7)$$

このうち，式（3）は．$(x_1, x_2, x_3)=(x, y, z)$ の

各座標方向それぞれについて与えられる．LESにおいては，式（3），(6）から速度u_iと圧力pの時間変化を得る．

ここで時間微分について，式（3）には2次精度Adams-Bashforth法，式（6）には1次精度Euler法を用いて離散化すると，

$$u_i^n - u_i^{n-1} = \delta t \{1.5 f_i(u^{n-1}, p^{n-1}) - 0.5 f_i(u^{n-2}, p^{n-2})\} \quad \cdots\cdots(8)$$

$$D^{n+1} - D^n = \delta t \{\nabla^2 p^n + g_i(u_i^n)\} \quad \cdots\cdots(9)$$

を得る．ここで，δtは時間きざみ，上付き添字n，n–1，n–2は離散化したときの時間stepをあらわす．式（8）は速度成分u_i^nの3成分について，式（9）は連続の条件

$$D^{n+1} = 0 \quad \cdots\cdots(10)$$

を代入した後，圧力p^nについての代数方程式とそれぞれみなして，これらを順次計算することで時間ステップ$n+1$の数値解を得る．以上の数値解法はMAC法と呼ばれ，非圧縮性流れ場解析に広く用いられている方法である．

上記のような非圧縮性流れの非定常解析において，計算時間の大半は式（6），(7）から得られる圧力に関するポアソン方程式の求解（連立一次方程式の逆行列の求解）に費やされ，その割合は全計算時間の90％以上とも言われている．本報では上記に基づく直交直線座標系LESコード（LES-FDM）と曲線一般座標系LESコード（LES-BFC）を用いてテストを行なう．また，乱流LESのように格子点数の多い問題のポアソン方程式（連立一次方程式）の解法には反復解法が用いられるが，そこに現れる典型的な演算式を用いたベンチマークテストが姫野［1］，白山［2］らより提案されている．これらは数十行程度のプログラムによるポータビリティの良いテストであり，PCを含む多くのシステムでの実行結果が報告されている［1］ことも踏まえ，今回のテストでも合わせて取り上げた．表1にテストプログラムの概要を示す．

テストにおける計算条件を表2に，テストした計算機環境を表3に示す．ここで用いた計算機は全て東京大学内に設置されているもので，情報基盤センタ，生産技術研究所電子計算機室，あるいは著者らの所属する研究室にて運用されている．表中，計算機名の欄の括弧にホスト名を示した．実行テストにおいては，実際の実行環境を想定して，マルチユーザ環境のままテストを行なった．なお，コンパイルオプションには標準的なものを用いている．

Table 1　テストプログラム概要

	格子分割法	ポアソン式の解法
姫野ベンチ	一般曲線座標	Point Jacobi 法
LES-FDM	直交直線座標	ICCGS 法
LES-BFC	一般曲線座標	BiCGStab 法

Table 2　計算条件

ケース名	離散化手法	計算点数	計算時間 step 数
姫野ベンチ	差分法	128 × 64 × 64	100～1000
LES-FDM	差分法	32 × 64 × 32	30
LES-BFC	差分法	32 × 64 × 32	30

3. 実行結果

表4にテスト結果を示す．LES-FDM，LES-BFCの結果には単位が秒（sec），ベンチマークの場合には，FLOPS[1]を用いていることに注意されたい．よって，姫野ベンチにおいては数字が大きい方が，LES-FDM，LES-BFCにおいては，数字が少ない方が，高性能であることを示す．

図1に表4をグラフ化したものを示す．図中LES-FDM，LES-BFCについての結果は，計算時間（sec）の逆数をとり，Sun Enterprise4500（ホスト名yamba）の値を100としたときの換算性能で示している．

表4，図1を見ると，実用コードであるLES-FDM，LES-BFCと姫野ベンチを比較しても高い相関が見られ，このベンチマークが計算機性能評価として妥当なものであると言える．

表4中，No.1とNo.2を比較すると，SPECfp95における差は20％程度なのにも関わらず，実行結果の差は2倍以上になっている．このことからSPEC値から性能を推定する場合には注意する必要があると言える．（SPECfp95はキャッシュに対する評価が過大であることが知られており，それに代わるものとして最近からSPECfp2000による性能表示に変わっていることを付記しておく）

No.5とNo.6とは，同じ計算機でOS，コンパイラが違う環境である．両者を比較すると，市販のOS・コンパイラを使っているNo.5の方がより良いパフォーマンスを示している．ちなみにNo.6はフリーのOS・コンパイラを使っている[2]．

No.7とNo.8を比較するとクロック周波数はほとんど同じなのに対して,実際の計算速度は3～4倍程度違うことがわかる．計算機の演算速度はCPUのクロック周波数だ

[1]FLOPS：Floating Operation Per Second
[2]GNU Fortran（g77）を用いた．

研　究　速　報

Table 3　テスト環境

	計算機名	CPU	OS
1	Sun Enterprise 4500 (yamba)	Ultra SPARC-II (400MHz)	Solaris2.6
2	COMPAQ AlphaServer DS20-6/500 (emu)	α21264 (500MHz)	Tru64 UNIX 4.0E
3	DEC AlphaStation 600-5/333	α21164 (300MHz)	OSF1v3.2c
4	COMPAQ XP1000	α21264 (500MHz)	Tru64 UNIX 4.0E
5	VT Alpha 600	α21164A (600MHz)	Digital UNIX 4.0D
6	VT Alpha 600	α21164A (600MHz)	RedHat Linux 5.1
7	SGI O2	MIPS R10000 (175MHz)	IRIX 6.5
8	SGI Origin2000	MIPS R10000 (195MHz)	IRIX6.5
9	SGI Origin2000	MIPS R10000 (400MHz)	IRIX6.5
10	Fujitsu VX (旧 fuji)		UXP/V
11	Hitachi SR8000 (sr8000-s,p)	250MHz	HI-UX /MPP
12	DELL XPS450	PentiumII (450MHz)	Windows98
13	Proside B750	PentiumIII (750MHz)	Windows98

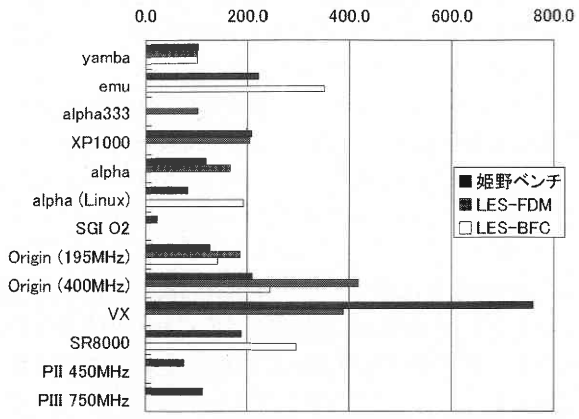

Fig. 1　計算結果の比較

Table 4　テスト結果

	姫野 (MFLOPS)	FDM (sec)	BFC (sec)	理論速度
1	101	389	572	SPECfp95 63.9
2	222	161	163	SPECfp95 76.1
3		385		
4	209	190		SPECfp95 58.7
5	117	235		SPECfp95 27.0
6	82		298	SPECfp95 27.0
7	22.8			SPECfp95 8.9
8	124	210	409	390M FLOPS
9	210	93	234	800M FLOPS
10	760	100		2G FLOPS
11	188		194 (86)	8G FLOPS
12	74			
13	112			

けでなく，2次キャッシュやメモリバンド幅など様々な要素が関わってくる．これに対し，No.8とNo.9では同じマシンにクロック周波数が違う場合の比較になっており，クロックスピードにほぼ比例してパフォーマンスが向上している．

No.10は生産技術研究所において99年度まで稼働していたベクトル型スーパーコンピュータである．姫野ベンチはベクトル計算に適したプログラムであるので，とても高速に演算できていることがわかる．

No.12，13はPCにおける結果である．近年，PCの性能向上が著しく，また価格も急激に安価になっていることから，数値シミュレーションの分野でも期待されている．そ

の結果は高価なEWSにもひけをとらず，性能対コストで見るとPCの方が優れる．しかしPCの場合はOS，ハードウェアの信頼性が若干低下しがちであるので，注意が必要である．なお，OSに限って言えば，LinuxやFreeBSDなどのUNIX系OSを導入することで解決できることを指摘しておく．

4. 並列計算ベンチマーク

RISCチップの高速化・低価格化と，高速な通信環境が提供されるようになり，近年の計算機環境は，高価なベクトル計算型の大形計算機からより安価なCPUを高速なネットワークで接続した並列型計算機へと移行しつつある．

並列計算機には大きく分けて，共有メモリ型並列計算機と分散メモリ型並列計算機の2種類が挙げられるが，分散メモリ型並列計算機上で並列計算を行なう場合，他のCPUが持つデータを参照するために専用のメッセージパッシングライブラリ[3]などを使う必要があり，ソースコードの変更を伴う．これが利用者にとって大きな負担となることが少なくない．それに対し，共有メモリ型並列計算機の場合，コンパイラによる自動並列化が期待できるので，プログラムがうまく適合すれば，ソースコードの変更なしに並列計算による高速化の恩恵を受けることができる．

今回テストした計算機のなかにも共有メモリ型並列計算機が含まれるので，そのうちのOrigin2000，SR 8000についてコンパイラの自動並列化による並列計算を実施した．（SR 8000についてはnode間で見ると分散メモリ型であるが，node内に8CPUを持ち，それらは共有メモリ型並列計算機として扱える）なお，LES-FDMで用いているICCGS法はそのままでは自動並列できないタイプの行列ソルバであるので，ここではテストしていない．

その結果をまとめたものを，表5，6に示す．

Origin2000（8CPU）における姫野ベンチの結果を見ると，1CPUの8倍以上の結果が出ている．これは，並列実行した際に1CPUあたりの実行モジュールの大きさが4MB程度となり，実行モジュールが完全に2次キャッシュに納まってしまうため，実メモリとの通信なしに計算が進んでしまうためと考えられる．（逆に言うとキャッシュによる高速化の例とも言える）

SR 8000（8CPU），LES-BFCにおいて加速率が低いのは，CPUあたりの問題サイズ（格子点数）が小さいために計算時間が非常に短く，相対的にスレッドの起動時間が大きくなってしまったためと考えられる．ここには載せていないが，格子点数を増やして計算すると，良好な加速率が得られている．

[3]MPIやPVM

Table 5　並列テスト結果（姫野ベンチ）

	速度 (MFLOPS)	加速率
Origin2000 1CPU	210	1.0
8CPU	2676	12.7
SR8000 1CPU	188	1.0
8CPU	1055	5.6

Table 6　並列テスト結果（LES-BFC）

	計算時間 (sec)	加速率
Origin2000 1CPU	305.1	1
8CPU	53.1	5.75
SR8000 1CPU	194	1
8CPU	86	2.3

5. おわりに

実用流体計算LESコードとベンチマークコードを用いて計算性能の比較を並列計算機を含むいくつかの計算機で行なった．

本報の実行テストは少数の特定プログラムのみを対象に行なったものであり，実際のプログラム実行環境（マルチユーザ，マルチジョブ）のままで実施したため，計測値の一般性や再現性は必ずしも保証できない．また，報告に記載されていない機種の詳細な仕様や，運用されているソフトウェアによっても結果が変わる恐れがある．本報のデータを利用する際には，これらの点に十分に注意されたい．

（2000年11月20日受理）

参考文献

1) http://w3 cic. riken. go. jp/HPC/HimenoBMT/
2) CFDnetにおける議論
3) 谷口伸行，生産研究48-8(1997), pp. 345-348.