大型計算実施報告書
09-07: 極高エネルギーでの空気シャワーの精密シミュレーション

2009 年 11 月 3 日

1 研究組織

1. 笠原克昌（かさはらかつあき）　早稲田大学・理工学術院・理工学研究所・客員教授 (専任待遇)
2. 櫻井信之（さくらいのぶゆき）　東大宇宙線研究所・助教
3. 武多昭道（たけたあきみち）　東大 (宇宙線研究所) D4
4. 木戸英治（きどえいじ）　東大 (宇宙線研究所) D3

2 実施の詳細

2.1 はじめに

最終的な目標は A)1 cpu で行えば 2 年以上の時間が必要な 10^{19}eV を越える極高エネルギー領域の空気シャワーシミュレーションを実用時間のうちに行う手法の MPI 環境での実現，B) これによりアメリカ・ユタ州で現在遂行されている TA 実験に資する, 事である.

ただ，今回の申請は，4 月から 9 月までの期間が対象であったが，宇宙線国際会議の関係から夏後半から使用をめざし，コードの MPI 化と KEK BG での性能テストの側面が強いものになるという前提で採択されている．当初 7 月後半から MPI 化に取り組む予定であったが，実際には 8 月初めからの着手となった.

2.2 MPI 化の実際

コード (Cosmos) は MPI などの分散並列処理機能をサポートしない PC クラスタ環境で分散処理をするためにすでに特別な方法を持っている (D-Skeleton=Distributed-Skeleton 法). それは処理を

1. 空気シャワーのスケルトンを作る (skeleton)
2. それを分割する (smash)
3. 分割されたスケルトンを個別の cpu で肉付けする (flesh)
4. 肉付けされた分割シャワーを 1 つのシャワーに構成する (assemble)

という 4 段階のステップに分けることで実現している．計算に時間を要するのは flesh 部分であり，他は全体からみれば微少である．flesh 部分では cpu 間でまったく通信が不要である．このような構成のため分散する個数が N なら計算時間はほぼ完全に $1/N$ になり，N の増加で性能が低下するという要因はないと言ってよい．

今回 MPI 環境にコードを適合させるため，以下のような構成とした．

1),2) はそれぞれ別 job として実行する．共に 10 分の job で実行可能である．1) では MPI の特徴を活かし、1 度に 10 シャワー程度の skeleton を作成して，そのうち最適なものを選ぶようにする．空気シャワー・シミュレーションには揺らぎが重要であるが，今回のシャワー発生は標準的なものが対象なので，このような選択が可能である．ここでのシャワー発生は全く独立なので，通信は不要である．

2) は分散処理するのが困難なのと、通常 10 分以内に終わるので分散処理の必要もなく 1 ノード job として実行する．3),4) は MPI の特性を活かして 1 つの job で処理する．N はなるべく大きい方がいいが，プログラム・サイズが 340MB 程度あるので co-processor モードを選択せざるをえない．テスト環境なので 128 ノードで実行をする．メモリーを各ランクに分散させるのも困難なので，co-processor モードでの実行は今のところ不可避である．

各ランクでの flesh は数% 以内の時間差で終了するので全ランクの flesh が終了後 assemble 作業に入る．これは disk の read/ write が主作業になり，各ランクが作成した file をランク 0 が sequential に処理して 1 つのファイルにまとめあげる作業となる．

2.3 問題点

このような方式で，MPI 化は予想通り容易に行えたが，予想外であったのは，計算時間が PC クラスタでの場合より ∼ 10 倍掛かる事であった．また，下記の “report” 例に示されるように MA 値も異常に小さい．

```
-----------------1 例目-----------------
     Unix Username: kasahara
          JobClass: qb128b
             Nodes: 128
           JobTime: 3774.69819732031
      FMAs average: 866854707.117188
      FMAs minimum: 749826339
      FMAs maximum: 955503502
        FMAs total: 110957402511
MFLOPS(MA) average: 0.9185950703125
MFLOPS(MA) minimum: 0.794566
MFLOPS(MA) maximum: 1.012520
-----------------2 例目-----------------
    Unix Username: kasahara
          JobClass: qb128b
             Nodes: 128
           JobTime: 6687.11463273437
      FMAs average: 760103522.171875
      FMAs minimum: 607885705
      FMAs maximum: 888021645
        FMAs total: 97293250838
MFLOPS(MA) average: 0.454667546875
MFLOPS(MA) minimum: 0.363617
MFLOPS(MA) maximum: 0.531175
```

このため，このような状況がどうして生ずるのかをテストする必要が出てきた．まず，実際の 3) 4) の過程てどのようなことが起こっているのかを調べた．

通信負荷が原因でない事は既に明らかで，通常の cpu 消費型の計算部分と I/O が絡む部分について着目して調べることにした．

Assemble の段階は I/O が主でかなりの実時間を消費している．ディスクは/bgwork0 を使用．このファイルシステムに同じ時期にアクセスが集中するような状況が生じている (?) と同じようなサイズのファイルを処理しても倍以上の時間がかかることがあると推察される．report の 2 例目はそのような状況で生じていると疑われる．

3),4) を行う job が 3 時間程度かかる場合，3),4) それぞれに半々程度の時間がかかっている場合がある．実時間として I/O にかなりの時間が割かれていることは確かである．各ランクが書き出すファイルは $N = 128$ のときバイナリで 150MB 程度，この値は N がさらに大きくなると小さくなる．このファイルをランクゼロが順次読み，処理してアスキーファイルとして書き出す．この最終ファイルは 20 GB 程度である．入射粒子のエネルギーを上げると、3) に要する時間が増えるが，file サイズはある限度以上増えないように書き出しを制限しているので 4) にかかる時間のエネルギー依存性はほとんどないとはずである．

3) の部分にも各ノードが 150MB のデータを随時書き出す I/O 時間が含まれる．各ノードがデータを書き出すタイミングはランダムなので，アクセスの競合はほとんど起こらないと想像され，実時間に対する割合は 1% 以下のはずである．従って I/O 部分を除いても PC クラスタより ~ 10 倍遅いという現象は説明困難である．

2.4 特別 job による原因究明

Cosmos は Fortran(ごく一部 C) で書かれており，Fortran77/90/95 がミックスされている．また，Fortran77 時代に正式にはサポートされてなかった構造体を使用している． この構造体は VAX 拡張仕様の構造体と言われるもので，サポートしてないコンパイラもある．例えば，KEK A 系の日立のコンパイラではサポートされてない．この VAX 拡張仕様の構造体が原因で異常に実行時間が掛かっている可能性がある．また I/O がどのように関係しているかも調べる必要がある．

そこで，Cosmos の中に組み込まれているハドロン反応の QGSJET2[1] の部分を取り出し，時間測定のプログラムを新しく作った．QGSJET2 は a)Cosmos が選択するハドロン反応モデルの典型の 1 つであること，b)VAX 拡張仕様は全く含まないこと，c) 初期化時に 200MB 超のアスキーファイルを読み込むので I/O に消費する時間を独立に計れること，d)Fortran77 相当の形式で書かれており，多くのその他の相互作用モデルのプログラムと同様で特に変わった点がない．e) 行数は 12300 行あまりで，これもハドロン反応モデルのプログラムとしては標準的である．

時間計測のプログラムは QGSJET2 が初期化にファイルを読むときと，各段階で時間を計測してその経過時間を書き出す以外はまったく I/O を含まないものとした．経過時間の測定には `MP I_WTIME()` を用いた．

2.4.1 テストの詳細

テストでは，pN(N=窒素) 衝突を使い、エネルギー一定にして，衝突イベントを生成するのに要する時間を測定し、KEK BG 以外の環境での結果と比較した．マシーンや Fortran コンパイラは表の通り．

あらかじめ分っていた事は，BG でカスケードを生成すると

- tasim501 シリーズでの計算時間より 10 倍以上の cpu 時間がかかる．例えば，1TeV の電子の起こすカスケードは tasim501 では 6s 程度で終わるが，BG では 60∼70 s 程度かかる．
- どうも I/O が相当遅そうである．job がスタートしてから計算が始まるまでに手間取っているようである．

[1] S.Ostapchenko, Nucl.Phys.Proc.Suppl.B151(2006)143

表 1: テスト環境

ハード	clock(GHz)	コンパイラ	注
KEKB powerPC	?	IBM BG 用	結果は正しそう
tasim501; AMD athlon	2	intel 10.1.022. 32 bit	
tasim599: AMD opteron	2.8	intel 10.1.022. 32 bit	
//	//	intel 10.1.022. 64 bit	注 1
MacBook; intel core 2 duo	2.4	intel 10.1.012 32 bit	注 2

注 1: 現在の Cosmos バージョンでカスケードの計算まで行うと 32bit モードの 100 倍以上の時間がかかる．実質使えない．
注 2: 現在の Cosmos バージョンでカスケードの計算まで行うと segmentation fault を起こす．少し前のバージョンでは dummy の write(0,*) ” ”を一行プログラムの最初の方に書けば動いた．スピードは遅くない．

そこで今回，qgsjet2 特有の大きなファイル (∼214MB, 329 万行) を初期化時に読む時の経過時間も計測した．

表 2: qgsjet2 用ファイルの読み込みにかかる時間

ハード	経過時間 (s)	ファイルの在処
KEK BG	52	NFS 経由のはず
tasim501	11	local raid5 disk
tasim599	5.2	上記を NFS で使う
MacBook	9.3	local system disk

BG 以外では競合する job/process はない．BG の場合，MPI により，同時に走る process(rank) 数を 32,16,4,1 と変えた．ただし互いの通信は一切無し．process 数 1 の場合は BG 以外と似た環境になると思われる．process 数が増えると同一のファイルを読むため，HD アクセスの競合が起こり経過時間が急激に増す可能性がある．また，ファイルの存在するシステムは他の Job もアクセスしているはずで，そのための性能低下も考えられる．しかしながら，不思議な事に，52 秒という経過時間 (読み込み開始から終了までの実時間) は、そうした環境にはほとんどよらずに一定であった．そして他のマシーンより格段に遅い．BG 特有の機構が関係しているのであろうか．

次に $Ep = 10^{15}, 10^{16}, 10^{17}, 10^{18}, 10^{19}, 10^{20}$eV についてそれぞれ数 100 イベント程度を生成し，発生粒子数と時間の相関を取った．結果の一部を図 1 に載せる．

最後のグラフから，BG は tasim501 の 4 倍 MacBook の 10 倍程度遅いことが分る．BG では、今回のプログラムに，発生データを Cosmos 用の VAX 構造体にセーブする部分を追加してみた．しかし，所要時間に目立った変化はなく，構造体が直ちに遅延につながるとは言えなそうである．Cosmos としての計算では BG は tasim501 より 10 倍遅いので，2.5 倍のファクタがどこから来るかは今の所不明である．MacBook が最高性能なのは意外である．

なお，コンパイルオプションとして最適化のレベルはデフォルトと-O0 をためしたが，問題になる差は見いだされなかった．

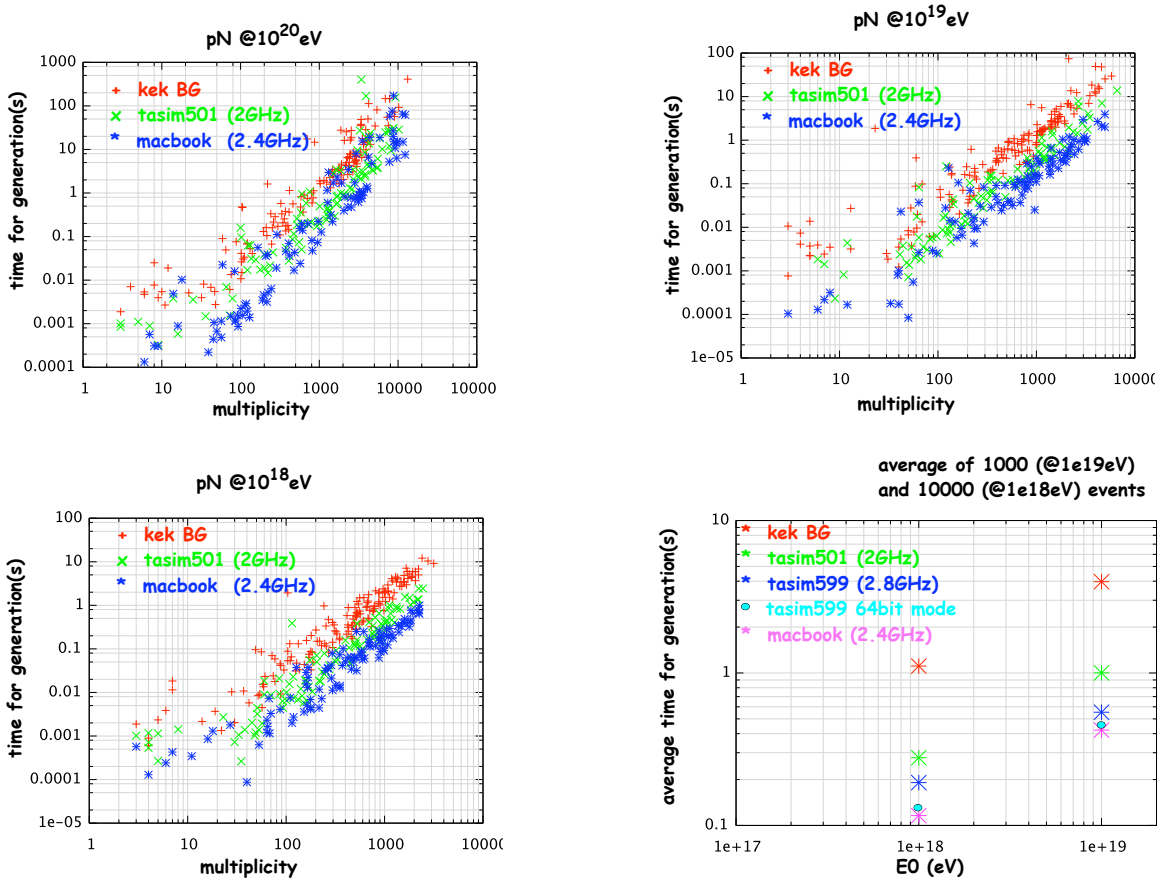

図 1: 上段および下段左: $10^{20}, 10^{19}, 10^{18}$ eV での多重度と所用時間の関係．下段右． $10^{19}, 10^{18}$ eV での平均所要時間/event(それぞれ 1000, 10000 イベントの平均).

3 結論

これまでの結果では、BG での実行が他環境より 10 倍程度の時間がかかる原因は VAX 拡張仕様ではない．I/O が主の部分も cpu 消費型の部分も 10 倍前後の時間がかかっている．テストした Fortran はごく普通のもので，Cosmos 全体に使われている．Cosmos 全体としては 20 万行を越えるので，細かいチューニングなどは不可能である．なにかが原因であると判明したとして，それが機械的に変換できるようなものであれば，別であるが，当面 BG 環境での実用になる速度は望めないと残念ながら結論せざるを得ない．

4 謝辞

今回の実施にあっては KEK 計算科学センターの松古 栄夫氏に一方ならぬお世話になりました．また、申請に当たって同じく KEK の橋本省二氏にもいろいろご助言いただきました．紙面を借りてお礼申し上げます.