Vol.2011-HPC-130 No.20
2011/7/27

ヘテロジニアスな並列計算環境を応用した連成・連係計算の提案

押川雄大[†1] 小林泰三[†2] 森江善之[†2]
高見利也[†2] 青柳睦[†2]

「京」コンピュータや TSUBAME 2.0 に代表されるように，計算環境は飛躍的に大規模化し，且つ，GPGPU などのアクセラレータがついたヘテロジニアスな環境に進みつつある．このような大規模でヘテロジニアスな計算環境を効率よく利用する計算方法として，計算科学で需要が拡大しつつある連成・連携計算を提案する．ヘテロジニアスな計算環境として GPGPU と CPU を設定し，連成・連携計算の対象として金属微粒子の融点解析を行った．GPGPU で生成される一次データを CPU で平行してデータ解析をさせ，その解析結果に基づいて次に GPGPU で実行する計算を決定する事により，ストレージに書き出すデータの削減と，GPGPU で実行する計算量の最小化を実現した．

Coupled and Cooparated Simulation on Heterogeneous Parallel Computing Environment

Yuta Oshikawa ,[†1] Taizo Kobayashi ,[†2]
Yoshiyuki Morie ,[†2] Toshiya Takami [†2]
and Mutsumi Aoyagi [†2]

As represented by “K” computers and TSUBAME 2.0, large-scale computing environments are progressing dramatically and becomming heterogeneous with such as GPGPU. We propose coupled and cooparated simulation, which is growing demand for computational science, as a calculation method in order to efficiently utilize a large heterogeneous computing environments. A meltign point of a nano meter sized metal cluster is analyzed on a CPU and GPGPU heterogeneous computing environment. The primary data generated by the GPGPU is analyzed in parallel by the CPU, and the next calculations on the GPGPU are deciding based on the analysis result. Thus, reduction of the data to be written to storage, and minimizing the amount of calculation on the GPGPU are achieved.

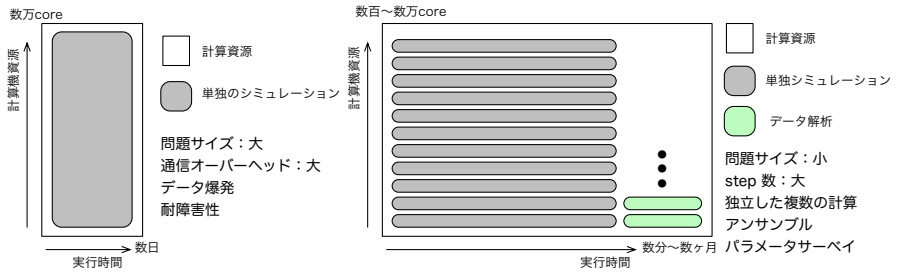

図 1 現在の典型的な計算例：(左) 大規模超並列計算と (右) データ解析を伴う embarrassingly prallel 計算

1. はじめに

1.1 大規模計算の現状

昨今の高性能計算機システムの多くは，メニーコアから成る CPU を複数個搭載したノードを互いにネットワークでつないだ並列計算環境である．そのノード数は今後も増加するのに加えて，GPGPU などのアクセラレータと呼ばれる演算加速機構を搭載することでシステム全体の演算性能を高める方法も広まりつつある．こういった計算機環境の状況を受けて，例えば「京」コンピュータを念頭に置いた超並列計算や，TSUBAME 2.0 の様なアクセラレータ付きのヘテロジニアスな計算環境を効率的に利用する研究が多方面で進められている[1)]．

この様な大規模計算の現状を端的に分類してみると，計算モデルは図 1 に示す 2 種類に大別できる． 左の図は，多数の演算ノードに問題サイズの大きな単独のシミュレーションを高度に並列処理させる計算方法で，大規模な超並列計算と云えば，現在ではこの計算方法をさすのが一般的である．しかし，大きな問題サイズの計算は，その並列度の増加にともなって様々な困難に直面する様になってきた．まず，並列度の増加に伴う通信コストの相対的な増大等によって，並列化による計算速度の向上が困難になってきている問題がある．次に，

†1 九州大学大学院
Kyushu University graduate school
†2 九州大学情報基盤研究開発センター
Research Institute for Information Technology

ノード数の増加に伴ってシステム全体の平均故障間隔 (MTBF) が下がる事により，ひとつのジョブの実実行時間を大きく取れなくなってきている問題もある．また，問題サイズの巨大化による計算量の増加から扱うデータ量も膨大になるために，データ爆発も問題になっている．これら 3 つの問題を解決するために，通信やフォールトトレランス，ストレージから数値計算のアルゴリズムに至迄さまざまな研究が続けられている．

その一方で，図 1 の右に示すように，パラメータサーベイやアンサンブル計算に代表される，単独のシミュレーションを多数独立して実行する計算科学の需要もある．これらの計算の特徴は，問題サイズは小さいものの多数の独立したシミュレーション（＝一次計算）を必要として，且つ，それらのシミュレーション結果をデータ解析する（＝二次，あるいは高次計算）ところまでが一組になっている所である．多くの場合，データ解析の結果を受けて新たな一次計算を実行するループ構造になるのも特徴である．これらの計算の具体的な対象としては，化学反応等の非定常・非平衡な系の数値計算から，一部分に固められた物質やエネルギー等が拡散していく緩和過程や，乱流の発生と消滅の様な過渡現象など様々に存在する．

実際に，これらの計算がどの程度の規模になるのかを，ナノメータースケールの微粒子（ナノ粒子）の物性を調べる為に分子動力学（MD）計算を行う場合で見積もってみよう．4 nm 程度のナノ粒子に含まれる原子の数はおよそ 1000 個のオーダーである．このナノ粒子を MD 計算するには時間刻みを 10^{-15} sec のオーダーにとる必要があり，また過渡現象を追うには $10^{-5} \sim 10^{-3}$ sec 程度まで計算する必要がある．これは，計算量だけで比較するならば，10^7 個の系を 10^{-11} sec まで計算するのと同じである．この一本の計算から生成されるデータ（一次データ）の量は，倍精度で 1000 step で平均したとすると

$$\text{総データ量} = \text{原子数} \times \text{次元数} \times \text{step 数}/10^3 \times \text{倍精度ビット数} \quad (1)$$

であるから，原子数を 1000，次元数を 6，step 数を 10^{12} とすると，総データ量はおよそ 50 PB にもなる．step 数を 3 桁落として μ sec までの計算にしたとしても，50 GB である．更に統計量をとるためには計算結果のアンサンブルを作る必要がある．これは，同一パラメータで初期条件のみが異なる一次計算のセットであり，パラメータサーベイとは異なる．このアンサンブルを作る為に，計算量は一桁上がる．実際にはこれに更にパラメータサーベイが加わる事になる．以上が一次計算の内容である．実際の研究では，この一次計算の結果である一次データを解析する（二次計算）ことにより研究に必要な物理量を抽出し，必要であればそれらの物理量を更に解析（三次計算以上の高次計算）して最終的に必要なデータを得ることになる．

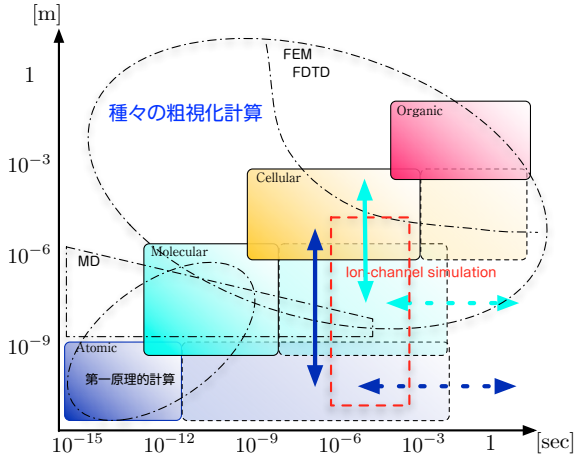

図 2　マルチスケール・マルチフィジックスの概念図

これらの系の数値計算の特徴を以下にまとめる．

- 一次計算の問題サイズが比較的小さい
- 一次計算は長時間（step 数が巨大）の時間発展計算であることが多い
- 多数の一次計算が必要（統計量をとる為にアンサンブルやパラメータサーベイが必要）
- 一次計算が作る一次データが膨大になる
- 一次計算同士は完全に独立である
- 一次計算の結果を解析する高次計算がある

ここでは「データ解析を伴う embarrassingly prallel 計算」（図 1 右）を取り上げ，その計算科学面での重要性を議論した．

1.2　連成・連携計算

ここでは「連成・連携計算」を概観する．連成・連携計算は，マルチスケール・マルチフィジックス（図 2）と呼ばれる対象に用いられる事が多い．マルチスケール・マルチフィジックスとは，時間空間的尺度が大きく異なったり，扱う物理量が粒子の位置座標と空間メッシュとの間での変換が必要であったり，それぞれの数値解法が分子動力学と有限要素法であったりと，複数の計算スキームを組み合わせる必要がある事である．例えば，化学反応

Vol.2011-HPC-130 No.20
2011/7/27

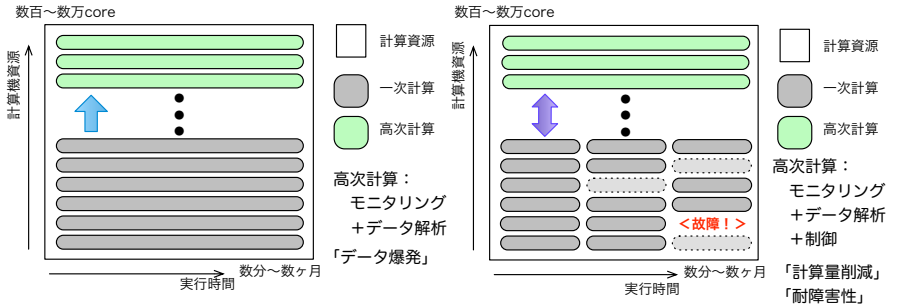

図 3　連成・連携計算としての多階層計算：(左) 高次計算としてモニタリングとデータ解析のみをする最も単純な例．高次計算の解析結果のみを出力するので「データ爆発」を緩和する．(右) 高次計算に一次計算の制御を含めた例．解析結果をもとに不要な一次計算を排除する（破線部分）などで計算量の削減を見込め，必要な一次計算を独立してスケジューリングすれば耐障害性も実現する．

をシミュレートする為に，まさに反応が進行している局所的な部分を量子力学に基づいた第一原理計算で行い，その他の大部分を古典力学に基づいた分子動力学計算で行って互いに連成させたり，エンジンの動作を調べる為に流体と燃焼と熱拡散を同時に扱う事，などが典型的な例である．この様なタイプの計算では，複数の異なる種類の計算とそれらの間のデータを取り持ったり制御したりする必要がある．このような計算をするために OCTA などの計算機構が考案されて成果も蓄積されてきている．

さて，マルチスケール・マルチフィジックスを対象にする連成・連携計算は非常に複雑な計算になるが，連成・連携計算の本質は異なる種類の計算を組み合わせるところにある．この観点から連成・連携計算の最も素朴な例を考えると，その一つとして，先に議論した「データ解析を伴う embarrassingly prallel 計算」のデータ解析と一次計算を同時平行で実行するもの（図 3 左）もその範疇に入ると看做せるであろう． これは，これまで直列に実行してきた一次計算とデータ解析を単純に並列実行した形でしかないが，一次計算の結果をストレージに保存する必要がなくなるなど，データ爆発対応への可能性がある．

次に考えるのは，データ解析の結果を一次計算にフィードバックする事である（図 3 右）．過渡現象を長時間計算していると，系が予期しないレアイベントに遭遇して，計算が続行できなくなったりする事がよくある．系がそのような異常な状態にあることを知るには高次計算としてのデータ解析が必要であるが，逆に言えば，データ解析を同時に実行していればリアルタイムに一次計算の状況が把握できるので，一次計算に対して例外処理を行える状況が整う事になる．そして，一次計算が独立したプロセスとして走っていれば，一次計算に修正をかけたり不要であれば止める事も可能である．従って，結果として不要な計算を避けることで計算量削減が可能になる．さらには一次計算を実行しているハードウェアが故障したとしても，代替計算を起動する事もこの枠組みの中では可能になる．つまり強力な耐障害性の実現を意味する事になる．

2. ヘテロジニアスな計算環境での連成・連携計算

ヘテロジニアスな計算環境として，今日もっとも身近なものの一つは，GPGPU + CPU の組み合わせである．そして CPU が GPGPU を制御するハードウェア上の構造が，図 3 の右の計算法での制御を実装するのに好都合である．GPGPU で計算を行う場合は，CUDA[2] や OpenCL[3] では，CPU で実行するプログラムから GPGPU で実行するカーネルと呼ばれる関数を呼ぶ形が取られている．従って，GPGPU での計算状況を CPU でモニタリングして必要であれば GPGPU での計算を制御すると云った一連の処理が，一つの C++ プログラムとして実装できるのが利点である．CUDA に付属しているサンプルに見られるインタラクティブなデモでは，GPGPU での一次計算を CPU がキーボードやマウスからの入力をもとにリアルタイムに制御していることからも，GPGPU での計算は CPU から制御しやすい事がわかる．

図 4 は CPU + GPGPU 環境へ連成・連携計算を適用した例である．ここでは流体と音波の連成計算を例として取り上げている．GPGPU 上では一次計算として流体力学計算 (CFD) と音波計算 (FDTD) のソルバーが同時に実行されていて，定期的に CPU 側にデータを送っている．CPU 上では高次計算として流体での音源になる渦運動解析と，音波のエネルギーが集中して流体との相互作用を考えなければならない位置を求める為の音場解析が実行されていて，それらの解析結果がそれぞれ連成相手の一次計算ソルバーに新たな計算条件としてフィードバックされる．また，CPU では計算制御も行われ，一次計算同士の同期や速度調節から，ソルバー切り替えなどの例外処理も行う．

3. 少数多体系問題への適用

上で我々が提案したヘテロジニアスな計算環境での連成・連携計算の素朴な例として，少数多体系問題の計算例を示す．

3.1 金属微粒子の融点解析

計算対象として，ナノメートルサイズの金属微粒子の融点を求める問題を考える．融点を

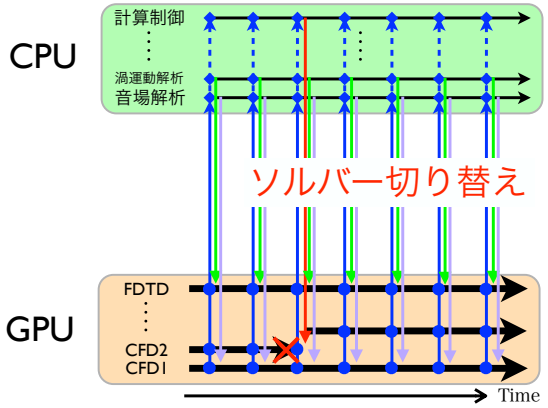

図 4　CPU+GPGPU への動的計算制御：下のクリーム色の部分が GPGPU で行う一次計算を，上の薄緑色の部分が CPU で行う高次計算をそれぞれ示している．ここでは流体と音波の連成計算を例として取り上げている．

求める計算方法は様々にあるが，今回は潜熱を直接観察する方法を採用する．潜熱は，固体から液体への相転移をする時などに，系に熱を与え続けても系の温度が変化しない現象であるから，系に様々なエネルギーを与えて全エネルギーに対する温度の対応関係（カロリック曲線）を調べればよい．

3.2 実　　装

本実装では CPU と GPGPU の環境で金属微粒子の融点を求めるため，与えた各エネルギーにおける温度を求めるプログラムを実装した．

計算の全体の流れは，まず CPU と GPGPU を用いて系の初期設定を行う．その後，GPGPU 側で一次計算として複数の系に関して各々の総エネルギーと温度を求める．次に CPU 側で解析計算として，各々の総エネルギーにおける温度をもとに，次に調べるエネルギーの範囲を決定する．決定したエネルギーの範囲において同様の計算を繰り返す．

3.2.1 前　処　理

計算を始めるにあたっての前処理は，系の初期設定である．以下にその内容を述べる．

まず，プログラム実行時に与えられる原子数に応じた原子の初期配置 (図 **5**) を，並列実行させる MD の数 n 個分用意する．

次にそれぞれの配置に乱数を加えることで原子をちらしていく．このとき，原子間の距離

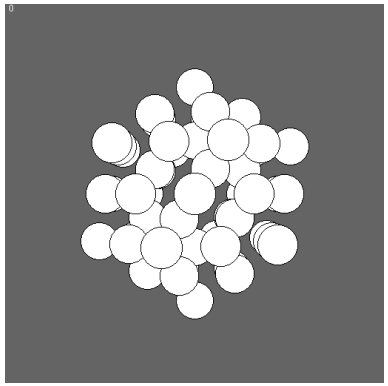

図 5　粒子数 55 個での初期配置の例

によっては内部エネルギーが大きくなり MD 計算を開始した瞬間にクラスターが崩壊してしまう可能性がある．このため，次に最急降下法を用いて quenching を行い系のポテンシャルエネルギーを低い方へ移動させる．最後に，系の温度を制御して所望の温度まで上昇させ，2000 ステップほど annealing を行う．

3.2.2 一次計算：GPGPU による分子動力学 (MD) 計算

計算環境は本実装では，GPU に NVIDIA 社の GPGPU 専用チップである「Tesla C2050」を，実行環境として「CUDA」（Compute Unified Device Architecture）を使用した．

一次計算は，個々の系の温度を求める計算であり，分子動力学 (MD) 計算を GPGPU 上で実行する．MD 計算の中で，系のポテンシャル・運動エネルギーを同時に算出しておく．MD 計算の詳細は以下である．

この実装ではモデルポテンシャルとして Morse ポテンシャルを採用した．ポテンシャルのパラメータには銅の値を用いた．以下に計算方法を示す．

粒子 i 粒子 j 間の Morse ポテンシャルは，2 粒子間の距離 r_{ij} を用いて式 2 で表される．

Morse ポテンシャル

$$U_{ij}(r) = \epsilon \left\{ e^{-2\beta_{ij}(r-r_{ij}^c)} - 2e^{-\beta_{ij}(r-r_{ij}^c)} \right\} \tag{2}$$

ここで

$$\begin{cases} \beta_{AA} = \beta_{BB} = \beta_{AB} = 1.358[A^{-1}] \\ r_{AA}^c = r_{AB}^c = r_{BB}^c = 2.866[A] \\ \epsilon = 0.3429[eV] \end{cases}$$

である．

粒子 i が他の粒子から受ける力 $\mathbf{F}_i$ は

$$\mathbf{F}_i(r_{ij}) = \sum_{j=1}^{N} (-\nabla U_{ij}) \tag{3}$$

になる．

ここで，質量 m_i の粒子 i の加速度 $\mathbf{a}_i$ は，

$$\mathbf{a}_i = \frac{\mathbf{F}_i}{m_i} \tag{4}$$

力 $\mathbf{F}_i$ を 3 次元ベクトル方向 x, y, z に分け，それぞれの加速度 a_{xi}, a_{yi}, a_{zi} を求める．x 軸方向では次のようになる．

$$a_{xi}(r_{ij}) = \sum_{j=1}^{N} \left[\frac{2\beta\epsilon x_{ij}}{m_i r_{ij}} \epsilon^{-\beta(r_{ij}-r_{ij}^c)} \{\epsilon^{-\beta(r_{ij}-r_{ij}^c)} - 1\} \right] \tag{5}$$

このようにして求めた $\mathbf{a}_i$ をもとに，時間ステップ t での粒子 i の速度 $\mathbf{v}_i$ と座標 $\mathbf{r}_i$ を $t-1$ ステップでの速度と座標をもとに計算する．

$$\mathbf{v}_i(t) = \mathbf{v}_i(t-1) + \int \mathbf{a}_i dt \tag{6}$$

$$\mathbf{r}_i(t) = \mathbf{r}_i(t-1) + \int \mathbf{v}_i dt \tag{7}$$

系の時間積分には，2 次のシンプレクティック積分法である速度ヴェルレ法[6] を用いた．

3.2.3 二次計算：CPU によるデータ解析と制御

本実装では，CPU に Intel Xeon X5650 を使用した．

CPU 側では GPGPU 側から一次計算結果を受け取る．その後次の系についての計算を GPGPU 側に実行させ，その間受け取った結果をもとにデータ解析を行う．データ解析では系を温度の範囲別に分けてそれぞれの範囲にある系の数を数え上げる．銅の融点付近では計算結果に多少のバラつきがでるため，全ての系の計算が終了した時点で，最もデータ数が多い範囲を融点付近であるとみなして，温度範囲をその付近に狭めて再計算を行う．これを繰り返すことで効率的にデータを集める．

3.2.4 全体の流れ

図 **6** に時間軸に沿ってプログラムが実行される流れを示す．

処理内容は以下のような流れとなる．

(1) 与えられた原子数に応じて初期配置を生成する．

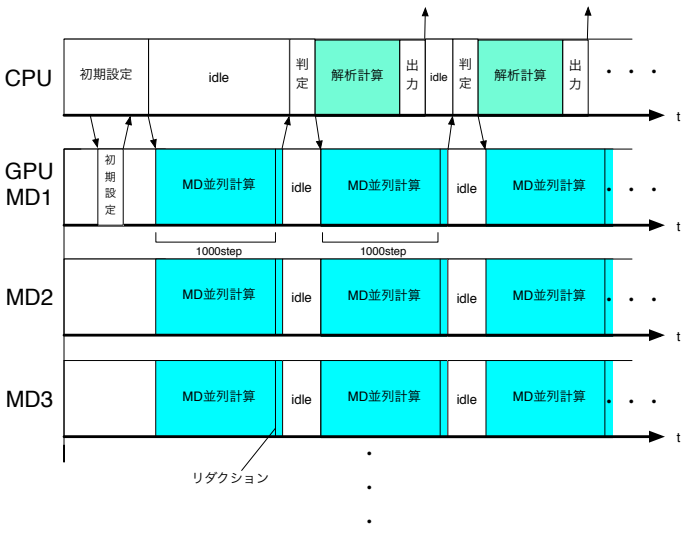

図 **6** 計算の流れ

(2) 生成した初期配置を並列実行数 n 個コピーする．

(3) n 個分の各原子の配置をそれぞれ乱数を使ってちらす．

(4) GPU 上では並列に実行できる処理の数に限りがあるため，用意した n 個のデータをいくつかのグループに分けて実行させる．加えて GPU 上では，グループ間の並列処理とグループ内での並列処理が実行される．まず最急降下法を用いてそれぞれのグループ内の各クラスターを所望のエネルギー値にする．このとき設定する各エネルギーの値は，グループ内のエネルギーの合計に偏りが出ないようなるべく均一に振り分ける．最急降下法の計算部分は GPU 上で行われ，計算処理を終了させる判定を CPU 側が行う．

(5) GPU 上で 2000 ステップの annealing を行う．

(6) 一次 MD 計算を 1000 ステップ実行し，全てのグループの系の総エネルギーと温度を求めるといった計算が終了するまで繰り返す．

(7) 融点付近の計算結果には多少のバラツキが出るため，融点付近での再計算を行うことでより精度を高める．そのためにここで融点の解析を行う．一つのグループの計算結

Vol.2011-HPC-130 No.20
2011/7/27

果について温度の範囲を 10 等分し，それぞれの範囲の中にあるデータの数を CPU 上で足し合わせていく．その中でデータ数が最も多い範囲を決定し，再計算ではその前後の範囲 3 個分に対応するエネルギーの範囲を用いる．

(8) ここで再び計算を開始するかどうかの判定を行う．再計算をする場合には設定エネルギーを新しく振り直して操作 (2) に戻る．GPU が再計算を進める間，CPU は次のグループの結果を現在の結果に足し合わせて並列に次の判定を行う．

(9) 得られた結果を出力する．

それぞれ（1）から（5）が初期設定,（6）が一次計算,（7）,（8）が解析計算,（9）が後処理に対応する．

このように，一次計算を GPGPU で実行すると並列に CPU が解析計算を実行するといったヘテロジニアスな並列計算環境を実装，連成・連携計算を行った．

3.3 数値計算とその結果

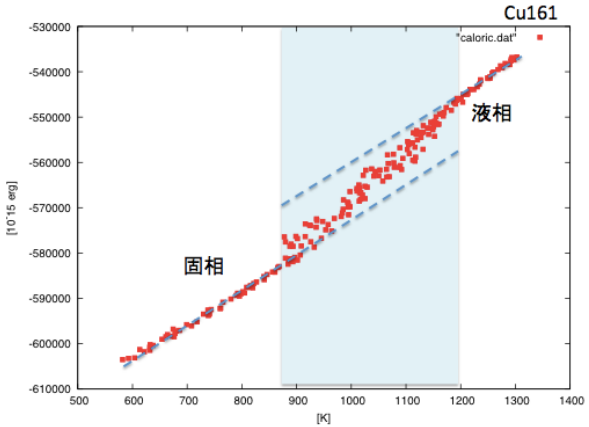

図 7 カロリック曲線

図 7 は数値計算結果であるカロリック曲線をグラフにしたものである．この図から融点は 1000 K 付近である事が見積もられた．

4. 結　　論

大規模な「京」コンピュータや TSUBAME 2.0 に代表される大規模且つヘテロジニアスな計算環境の効率的な利用方法として，計算科学で需要のある連成・連携計算の提案とその最も素朴な実装を行った．

本実装では，ヘテロジニアスな並列計算環境として GPGPU と CPU を設定し，連成・連携計算の対象として金属微粒子の融点解析を行った．融点解析は潜熱を利用してカロリック曲線を描いて行った．カロリック曲線を生成するには，固相から液相にわたるエネルギー領域を網羅する為に，各エネルギーでの小さな MD 計算を多数行う必要がある．この小さな MD 計算を一次計算として GPGPU 側で複数実行させることで処理を高速化した．さらに，次々に GPGPU から上がってくる一次データを CPU 側で平行して解析して，次に GPGPU で実行させる系のエネルギー値を割り出すことにより，最終的な一次計算の量を最小限に抑える事に成功した．

5. 今後の展望

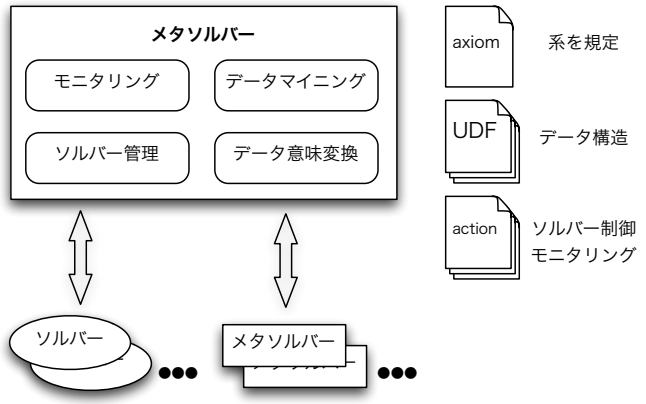

図 8 メタソルバーの概念図：連成・連携計算全体の管理を行う階層化された概念である．

本実装での計算速度向上を定量的に測る事も重要である．しかし，連成・連携計算は，リ

ファレンスになる計算が互いにバラバラであるために人手を介する必要があったり，そもそも連成・連携して初めて計算できる対象もあり，計算速度向上率を定義する事が困難な場合がある．

計算方法を発展させる方向では，既存の優秀なソルバーをできる限り利用した連成・連携計算機構の設計と，そのミドルウェア ToolKit として実装を目指す．既存のソルバーをまとめる役割としてのメディエイターは，既存の連成・連携計算スキームでは仲介役に留まっている．しかし，メディエイターを ToolKit 化してメタソルバーの位置づけにすることにより，各ソルバーが計算を続ける系のモニタリングからそのリアルタイムなデータマイニングとソルバーの制御などを行うフレームワークを構築することが重要である．いわば，メタソルバーは多細胞生物に於ける神経系としての役割を果たすものである．

メタソルバーの役割は，ソルバーが実行する一次計算をモニタリングしてその結果をデータマイニングなどで解析をし，必要なデータ意味変換を行った後に適切なソルバーにフィードバックする．それと同時に必要なソルバーを起動したり，不要なソルバーを停止させたりする．メタソルバーの動作は連成・連携計算全体の構造を規定する **axiom** ファイルに記述し，ソルバー間で交換するデータ構造は OCTA の **UDF** を拡張して利用する．ソルバーの制御やモニタリングの内容は **action** ファイルで指定する．メタソルバーは，ソルバーとしてメタソルバーを従える事も可能にする．

この独立したソルバーを管理するメタソルバーのもつ構造は，独立したサイトを管理する計算機グリッドの管理運用機構[12] と相似であり，本研究の方向性は，マルチスケール・マルチフィジックスの連携・連成計算と云う計算科学だけでなく，複数のサイトを跨ぐようなインタークラウドの管理運用などの計算機科学への貢献も期待できるであろう．

参 考 文 献

1) http://www.top500.org/
2) http://www.nvidia.co.jp/object/cuda_home_new_jp.html
3) http://www.khronos.org/opencl/
4) 加藤季広，青木尊之，額田彰，遠藤敏夫，松岡聡，長谷川篤史，姫野ベンチマークの GPU マルチノード実行における通信と演算のオーバーラップによる高速化 ～ 32GPU で 700GFLOPS 超を達成 ～，情報処理学会研究報告，Vol.2009-HPC-120 No.3
5) 青木尊之，額田彰，はじめての CUDA[クーダ] プログラミング，工学社，ISBN978-4-7775-1477-9
6) 岡崎進，コンピュータシミュレーションの基礎，化学同人，ISBN4-7598-0856-6
7) 清水寧，池田研介，澤田信一，2 元金属クラスターにおける自発的合金化現象とハミルトン系モデル（複雑系５），CiNii.http://ci.nii.ac.jp/naid/110006452093
8) 小林泰三，金属微粒子に於ける原子拡散，ー自発的合金化現象から金属微粒子の普遍的物性解明を目指してー，2002
9) NVIDIA．NVIDIA CUDA Programming Guide, CUDA ZONE.http://www.nvidia.co.jp/
10) NVIDIA．ホワイトペーパー NVIDIA の次世代 CUDA^{TM} コンピュートアーキテクチャ: Fermi^{TM} http://www.nvidia.co.jp/docs/IO/81860/NVIDIA_Fermi_Architecture_Whitepaper_FINAL_J.pdf
11) Lars Nyland，Mark Harris，Jan Prins，GPU Gems 3 Chapter 31 Fast N-Body Simulation with CUDA，NVIDIA，CUDA ZONE．http://www.nvidia.co.jp/
12) 小林泰三, 天野浩文, 青柳睦, 合田憲人,「大学間連携グリッド基盤の運用」情報処理学会誌 Vol.51, No.2 2010 年 2 月