粒子モデルのための動的負荷分散アルゴリズム

2 G − 2

武宮 博，川崎 琢治，樋口 健二

日本原子力研究所　計算科学技術推進センター

1. はじめに

粒子モデルによる数値計算は，十分に精度のよい解を得るために多大な計算時間を必要とするため，並列処理による効率的な計算が望まれている分野の一つである．効率的な並列処理を実現するために，領域分割や粒子分割に基づく並列計算アルゴリズムが多数提案されているが，並列計算の対象となるタスク(個々の粒子や領域の計算)を静的にプロセッサに割り付けた場合，計算負荷のインバランスが生じやすいことなどの理由で，動的な負荷分散を試みる研究が行われている．

本稿では，分散メモリ型スカラ並列計算機や異機種計算機クラスタ上での並列粒子計算に対するマスタ-スレーブ型の動的負荷分散アルゴリズムを提案し，粒子輸送モンテカルロシミュレーションコードMCNP4A[1]に対して適用した結果について述べる．

2. 動的負荷分散アルゴリズム

粒子計算の動的スケジュール問題は，マスタからスレーブにタスクをディスパッチする際，粒子輸送モンテカルロ計算のように転送すべきデータ量がタスクサイズによらず一定となる問題と，DSMC(Direct Simulation Monte Carlo)計算のようにタスクサイズに比例する問題の２種類に大別される．本稿で提案するアルゴリズムは，転送すべきデータ量がタスクサイズに依存しない問題を対象とする．

本手法では，スレーブとなる各プロセッサが，個々にスケジューリングコストとロードインバランスの和($H_{SCH}+H_U$)を最小にするように動的にスケジューリングを行う．具体的には，各スレーブはタスクの処理毎に通信時間と計算時間を測定し，そのデータを基に予想されるH_{SCH}とH_Uからコスト関数$H=H_{SCH}+H_U$を最小とするようなタスクのサイズを決定する．

まず，図1のようにHの評価に必要な幾つかの量を定義する．j番のスレーブプロセスにおいてk回目に処理されるタスクのサイズをN^{ini}_{jk}で表し，その処理に要した計算時間をT^{calc}_{jk}とする．また，全体計算はN^{Pe}個のスレーブプロセスにより実行されるものとし，全部で初期粒子N^{total}個の粒子計算を行うものとする．タスクサイズを決定し，マスタプロセスからタスクを獲得するためのスケジューリングコストはT^{sch}_{jk}とする．

H_Uは動的タスク割り当て手法を採用していることから高々１タスク計算時間程度である．各スレーブプロセスは，自分が最終計算終了プロセスになると仮定した場合に，どの程度のばらつきが生じるかを見積もる．すなわち，

H_U～単位サイズのタスクの平均計算時間

・タスクサイズ

$$\sim N^{uni}_{j\ k+1}\cdot\frac{\sum_k T^{calc}_{jk}}{\sum_k N^{ini}_{jk}}$$

一方，スケジューリングコストは

H_{SCH}～スケジューリング１回あたりの平均コスト

・スケジューリング回数

ここで，スケジューリング回数は(自分が最終的に処理する総タスクのサイズ／１タスクあたりの平均サイズ) で表され，自分が最終的に処理する総タスクのサイズはk回目の評価時点で全体で処理されているタスクのサイズと自分がそれまでに処理したタスクのサイズを基に予測する．すなわち，k回目の評価の時点で全体として$\sum_j\sum_k N^{ini}_{jk}$個処理されているのに対し，自分は$\sum_k N^{ini}_{jk}$個処理していることから，自分が最終的に処理する粒子数は$\frac{N^{total}\cdot\sum_k N^{ini}_{jk}}{\sum_j\sum_k N^{ini}_{jk}}$と予想される．従って，

$$H_{SCH}\sim\overline{T^{sch}_j}\cdot\frac{N^{total}\cdot\sum_k N^{ini}_{jk}}{\sum_j\sum_k N^{ini}_{jk}}\sim\frac{\sum_k T^{sch}_{jk}}{k}\cdot\frac{N^{total}\cdot\sum_k N^{ini}_{jk}}{\sum_j\sum_k N^{ini}_{jk}}/N^{uni}_{j\ k+1}$$

以上より，コスト関数Hは

$$H=N^{uni}_{j\ k+1}\cdot\frac{\sum_k T^{calc}_{jk}}{\sum_k N^{ini}_{jk}}+\frac{\sum_k T^{sch}_{jk}}{k}\cdot\frac{N^{total}\cdot\sum_k N^{ini}_{jk}}{\sum_j\sum_k N^{ini}_{jk}}/N^{uni}_{j\ k+1}$$

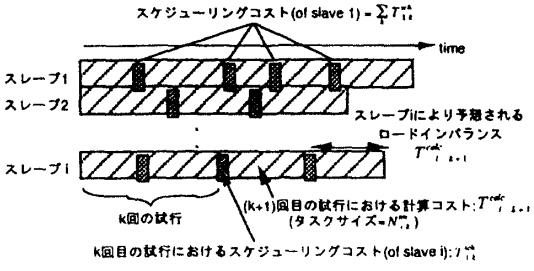

図１タスクサイズ決定に用いられる諸量

Dynamic Load Balancing Algorithm for Particle Simulation Model

Hiroshi TAKEMIYA, Takuji KAWASAKI, Kenji HIGUCHI
Center for Promotion of Computational Science and Engineering , Japan Atomic Energy Research Institute

この関数を最小とするように$N^{ins}_{j\ k+1}$を決定すればよい．よって

$$A=\frac{\sum_k T^{calc}_{jk}}{\sum_k N^{ini}_{jk}},\ B=\frac{\sum_k T^{sch}_{jk}}{k}\cdot\frac{N^{total}\cdot\sum_k N^{ins}_{jk}}{\sum_j\sum_k N^{ins}_{jk}}$$ とすれば

$$N^{ins}_{j\ k+1}=\sqrt{\frac{B}{A}}$$

上記の手法では，毎回各スレーブがタスク計算時に平均スケジューリングコストと平均タスク計算時間を計算し，それに基づきタスクサイズを計算するため，一般にタスクサイズは毎回変動する．その際，実際のスケジューリングコストと算出するスケジューリングコストの間に差異が生じる可能性がある．これは，スケジューリングコストの算出において，最初からタスクサイズ$N^{ins}_{j\ k+1}$で実行していると仮定しているためである．

例えば，あるプロセッサにおいてk回目まではタスクサイズN_0で処理を行い，k＋1回目に要求タスクサイズがN_1に変化したとする．この場合，最初からタスクサイズN_1で計算を行っていた場合の通信回数は$\frac{k\cdot N_0}{N_1}$だから，$N_0>N_1$であれば結果的にスケジューリングコストを削減したことになる．しかし，$N_0<N_1$の場合，実際にかかったスケジューリングコストは上記アルゴリズムで算出されるコストを上回り，結果的に並列実行効率を低下させる．

この並列実行効率の低下を防ぐために，TSSアルゴリズム[2]を応用して，最終的に上記コスト関数の評価が正しくなるようにタスクサイズを調整することにした．具体的には，以下の手法で調整を行う．

(1)上記アルゴリズムに基づき$N^{ins}_{j\ k+1}$を算出する．

(2)過去k回において本来計算しておくべきタスクサイズの和$k\cdot N^{ins}_{j\ k+1}$と，実際に計算を行ったタスクサイズの和$\sum_k N^{ins}_{jk}$を比較する．

(3)$\delta N=k\cdot N^{ini}_{j\ k+1}-\sum_k N^{ins}_{jk}<0$ならば次回要求タスクサイズとして$N^{ins}_{j\ k+1}$を採用する．$\delta N>0$ならば，最終要求タスクサイズを$N^{ini}_{j\ k+1}$とし，今後予測される通信回数内で$\delta N$を処理するようTSSスケジューリングを行う．すなわち，次回要求するタスクサイズは

$$N^{ins}_{j\ k+1}+\frac{2\cdot\delta N\cdot N^{ins}_{j\ k+1}}{N^{total}\cdot\sum_k N^{ins}_{jk}\Big/\sum_j\sum_k N^{ins}_{jk}-\sum_k N^{ins}_{jk}}$$

となる．

3. 性能評価

汎用粒子輸送モンテカルロシミュレーションコードMCNP4Aに上記アルゴリズムを適用し，Intel Paragon上で性能評価を試みた．今回性能評価に用いた問題は，人体に対する実効線量当量算出問題である．人体模型は，約200枚の二次曲面によっ記述された約100の3次元領域によって表現されている．初期入射粒子数は100万個，一粒子あたりの平均処理時間は5.43msecである．

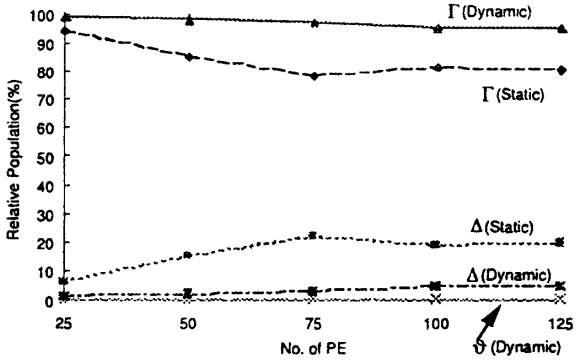

図2. Intel Paragon上での性能評価結果

性能評価のために，計算実行前にサイズN^{total}/N_{PE}のタスクを各スレーブに割り付ける静的負荷分散手法を比較対象とした．性能評価の指標として，

$$\Gamma=\frac{100\cdot\sum_j\sum_k T^{calc}_{jk}}{\sum_j\sum_k T^{calc}_{jk}+\sum_j\sum_k T^{sch}_{jk}+\Delta W}$$

$$\vartheta=\frac{100\cdot\sum_j\sum_k T^{sch}_{jk}}{\sum_j\sum_k T^{calc}_{jk}+\sum_j\sum_k T^{sch}_{jk}+\Delta W}$$

$$\Delta=\frac{100\cdot\Delta W}{\sum_j\sum_k T^{calc}_{jk}+\sum_j\sum_k T^{sch}_{jk}+\Delta W}$$

を採用した．$\Delta W=\sum_j(Max_k(T^{calc}_{jk}+T^{sch}_{jk})-(T^{calc}_{jk}+T^{sch}_{jk}))$である．$\Gamma,\vartheta,\Delta$は各々，粒子計算，スケジューリング，待ち状態にあるスレーブプロセスの相対比率を表す[cf.2]．

性能評価結果を図2.に示す．図2.より，本アルゴリズムは少ないスケジューリングコストで静的負荷分散手法と比較してロードインバランスを低減していることがわかる．

4. まとめ

局所メモリ型並列計算機や異機種計算機環境における並列粒子計算に関して動的な負荷分散手法を提案した．また，粒子輸送モンテカルロコードMCNPを対象としてIntel Praragon上で本手法の性能評価を行った．

謝辞

早稲田大学教授 笠原博徳氏には，本研究に関する議論において有用な助言を頂いた．ここに深謝する．

参考文献

[1] J. Briesmeister : MCNP-A General Monte Carlo N-Particle Transport Code, LA-12625-M, (1993)

[2] T. Tzen, M. Ni : Trapeziod Self-Scheduling : A Practical