

LAN上での分散計算における効率と粒度について

荒木啓二郎 岡村耕二¹ 藤井義巳 平原正樹

九州大学工学部

高性能ワークステーションとLANで構成される疎結合分散環境は、並列処理を行うには通信のオーバーヘッドが大きいのだが、粒度を適切に設定することで並列化による処理速度の向上が期待できる。並列処理の効率は、処理単位の粒度と関係があるといわれる。しかし、この粒度の定量的な定義は与えられていない。本論文では分散計算のモデルを定義し、そのモデルに基づいて「粒度の値」という定量的な定義を示した。さらに実際の並列処理における粒度と効率の関係を明らかにするために、Ethernetで結合された30台のワークステーションからなる疎結合分散環境で分散計算の実験を行った結果を掲げ、その効率と粒度の値との関係について考察した。

Efficiency and Granularity of Distributed Computation on LAN

Keijiro ARAKI, Koji OKAMURA, Yoshimi FUJII and Masaki

HIRABARU

Faculty of Engineering, Kyushu University

Hakozaki, Higasi-ku, Fukuoka 812 JAPAN

Though a loosely-coupled distributed environment consisting of workstations and a local area network(LAN) has high communication cost for parallel computation, speed-up can be expected with adequate granularity. The efficiency of parallel computation is closely related with the granularity in the parallel computation. But there is no general quantitative definition of granularity. In this paper we propose a model of distributed computation and present two kinds of quantitative definitions of the *value of granularity*. We made experiments with distributed computation upon a distributed system with 30 workstations connected by Ethernet, and discuss the relations between the granularity and the efficiency.

1 はじめに

多数の高性能ワークステーションで構成される LAN は多くの大学、機関に普及してきた。現在 LAN では主にファイルサーバやプリントサーバの様な機能分散のサービスが提供されているのだが、並列計算の様な負荷分散のサービスは提供されていない。疎結合分散環境は通信のオーバーヘッドが大きいのだが、適切な問題を選び、並列実行の単位を適切に設定すれば、並列化の効果が期待できると考える。本研究では、九州大学工学部情報工学科に設置されている、Ethernet で結合されたワークステーションから構成される疎結合分散環境を利用して実際に並列計算を行い [岡村 89]、その効果を確かめると同時に並列実行の単位としての粒度について考察を行った。本論文では、まず 2 節で我々が基づく分散計算のモデルを定義する。3 節でこのモデルに基づいて、並列処理の単位である粒度に対する定量的な定義として、粒度の値を導入する。4 節では Ethernet で結合された 30 台のワークステーションからなる疎結合分散環境で、実際に分散計算を行い、その結果について考察する。5 節では本研究のまとめを述べる。

2 分散計算のモデル化

我々はまず、分散計算が進行する過程のモデル化を行った。ホストと呼ばれる 1 台の計算機が、計算に必要なすべてのデータを持っている状態からスタートする。計算の過程は次の 3 つのフェーズから成る。

- データの分配：ホストは並列化に必要な前処理を施した後、エージェントと呼ばれる複数台の計算機にデータを配る。
- 並列計算：それぞれのエージェントはホストからデータを受け取ると、必要なら他のエージェントと通信を行いながらデータの処理を行う。

- データの収集：エージェントは処理が終了するとホストにデータを送信し、ホストは計算結果としてまとめる。

3 粒度の値の定義

3.1 粒度の値

分散計算の効率は粒度と呼ばれる概念と深い関係があると言われている。粒度とは並列処理の単位で、CPU が通信無しに連続して一度に処理することのできる仕事の大きさ [小柳 89] である。しかしながら、この連続して一度に処理される仕事のみを取り上げて、その大きさを具体的に何ステップとか何秒とか言ってもあまり意味がない。処理する仕事の大きさと、通信に要するコストとの相対的な関係で、粒度というものを評価する必要がある。このことから我々は、処理時間と通信などの並列化に伴うオーバーヘッドの関係から、粒度の値というものを以下のように定義した。

$$\text{粒度の値} = \frac{(\text{全処理時間} - \text{並列化によるオーバーヘッド})}{\text{全処理時間}}$$

並列処理の過程では、通信無しに連続して一度に処理される仕事がいくつも現れ得る。上掲の粒度の値の定義においては、それらの個々の大きさについて論じるのではなく、全体としての平均値に着目する。しかも上述のように並列化に伴うオーバーヘッドとの相対的な比率で考える。前節で定義した分散計算のモデルには次の 2 種類の通信形態が存在する。

- データの分配、収集に関する通信 (ホスト ↔ エージェント)
- 並列計算の途中で発生する通信 (エージェント ↔ エージェント)

これらの通信に対応する粒度をそれぞれ

- マクロな粒度 (macro.granularity)
- ミクロな粒度 (micro.granularity)

と定義する。マクロな粒度は、全体の仕事を複数のエージェントで並列処理する場合の各エージェントが分担する仕事の大きさを表すもので、並列処理の単位の一つと考える。ミクロな粒度は各エージェントで処理される仕事の内、通信を含まない部分の大きさを表すものである。これら二種類の粒度を定義するために、2節に掲げたモデルに基づいて進行する分散計算において以下のものを定義する。

all.time: 分散計算の実行に要する時間。すなわち、2節で述べた、データの分配、並列計算、データの収集に要する時間の合計。

agent.time: 並列計算に要する時間。すなわち、それぞれのエージェントがホストからデータを受け取った後、各エージェントでの処理が完了するまでの時間の平均。

interaction.time: 並列計算中における通信のオーバーヘッド。すなわち、それぞれのエージェントにおける処理の途中でエージェント同士が相互に通信するために要する時間をエージェントについて平均したもの。

ここでは、通信によって生じるオーバーヘッドには、同期による待ち時間も含まれることにする。我々は同期による待ち時間も並列化に伴うオーバーヘッドであると解釈して、粒度の値を計算する際、オーバーヘッドに含めるべきであると考えた。この時、二種類の粒度の値は次のように定義される。

$$\begin{aligned} \text{macro.granularity} &= \frac{\text{agent.time}}{\text{all.time}} \\ \text{micro.granularity} &= \frac{(\text{agent.time} - \text{interaction.time})}{\text{agent.time}} \\ &\times \frac{\text{number.of.agents}}{\text{number.of.agents} + 1} \end{aligned}$$

ここで *number.of.agents* はエージェントの台数である。また、*micro.granularity* の値に $\frac{\text{number.of.agents}}{\text{number.of.agents} + 1}$ の因子が付いているのは、ここでの分散計算には *number.of.agents* 台のエージェントの外にホストも使われているということを考慮に入れるからである。

3.2 分散計算の効率

分散計算の効率は以下のように定義される。[Modi 88][小柳 89] 並列処理の効率は、複数台の演算装置を使って処理時間をどれだけ短縮できたかで評価される。あるアルゴリズムを *p* 台の演算装置で処理した時に要する時間を T_p とすると、加速率 S_p は、

$$S_p = \frac{T_1}{T_p}$$

と定義される。1台での計算時間を *single.time*、加速率を *speedup* とすると、本モデルでは

$$\text{speedup} = \frac{\text{single.time}}{\text{all.time}}$$

single.time とは、*n* 台のエージェントで計算したのと同じ量の計算を、1台のエージェントで行った場合にかかる計算時間である。

$$\begin{aligned} \text{single.time} &= \\ &(\text{agent.time} - \text{interaction.time}) \\ &\times \text{number.of.agents} \end{aligned}$$

効率 E_p は、加速率 S_p を演算装置の台数 *p* で割った値で、演算装置が無駄なく働いている度合を示す。 E_p は1を越えない。

$$E_p = \frac{S_p}{p}$$

本モデルでは効率 *efficiency* は、

$$\text{efficiency} = \frac{\text{speedup}}{\text{number.of.agents} + 1}$$

number.of.agents はエージェントの数である。分母が *number.of.agents* + 1 となっているのは、ホストもこの分散計算のために使用した計算機として勘定するからである。

3.3 粒度の値と効率との関係

我々の分散計算モデルでは、次の関係式が成り立つ。

$$\text{効率} = \text{マクロな粒度} \times \text{ミクロな粒度}$$

証明は以下の通り。

効率は定義より、

$$\begin{aligned} \text{efficiency} &= \frac{\text{speedup}}{\text{number.of.agents} + 1} \\ &= \frac{\text{single.time}}{\text{all.time}} \\ &\quad \times \frac{1}{\text{number.of.agents} + 1} \\ &= \frac{(\text{agent.time} - \text{interaction.time})}{\text{all.time} \times (\text{number.of.agent} + 1)} \\ &= \frac{(\text{agent.time} - \text{interaction.time})}{\text{all.time}} \\ &\quad \times \frac{\text{number.of.agents}}{\text{number.of.agents} + 1} \end{aligned}$$

一方、2種類の粒度の定義から、

$$\begin{aligned} \text{macro.granularity} \times \text{micro.granularity} &= \frac{\text{agent.time}}{\text{all.time}} \\ &\quad \times \frac{(\text{agent.time} - \text{interaction.time})}{\text{agent.time}} \\ &\quad \times \frac{\text{number.of.agents}}{\text{number.of.agents} + 1} \\ &= \frac{(\text{agent.time} - \text{interaction.time})}{\text{all.time}} \\ &\quad \times \frac{\text{number.of.agents}}{\text{number.of.agents} + 1} \end{aligned}$$

よって

$$\text{efficiency} = \text{macro.granularity} \times \text{micro.granularity}$$

証明終り。

4 定義したモデルに基づく分散計算の実験

4.1 実験に用いた疎結合分散環境

我々が分散計算の実験に用いた環境は、DEC社製のワークステーション VAXstation 2000(メインメモリ 4MB、CPU の処理能力約 1MIPS)が単一のイーサネット (10Mbit / sec) で接続されている。通信のプロトコルは TCP/IP を使用した。具体的には UNIX のソケットを用いて C 言語でコーディングした。

4.2 Block Odd-Even Sort のアルゴリズム

我々は定義した分散計算のモデルを用いて、疎結合分散環境上で実際に並列計算を行った。分散計算の例題としては Block Odd-Even Sort[Bitton 84]を取り上げた。Block Odd-Even Sort のアルゴリズムは次の通りである。

S-1. ホストは n 台のエージェントにデータを分配する。

S-2. エージェントはホストから受け取ったデータをそれぞれ独立に ローカルソートする。

S-3. i 番目のエージェントは以下の操作を n 回繰り返す。

1. 偶数回目には

(a) 偶数番目のエージェントは $i + 1$ 番目のエージェントにデータを渡す。

(b) 奇数番目のエージェントは受け取ったデータと自分のデータをマージし、分割して半分を $i - 1$ 番目のエージェントに返す。

2. 奇数回目には

(a) 奇数番目のエージェントは $i + 1$ 番目のエージェントにデータを渡す。

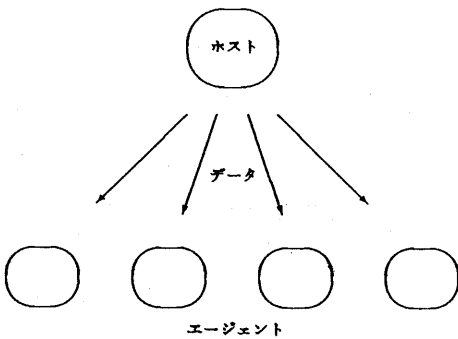
- (b) 偶数番目のエージェントは受け取ったデータと自分のデータをマージし、分割して半分をi-1番目のエージェントに返す。

S-4. ホストはエージェントからソートされたデータを収集し、一つのデータにまとめる。

前節で掲げた分散計算のモデルに基づいて、処理の過程を図で示すと次のようになる。

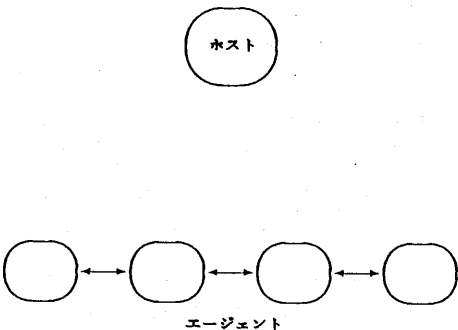
データの分配

ホストは前処理をした後、エージェントにデータを分配する。(S-1)



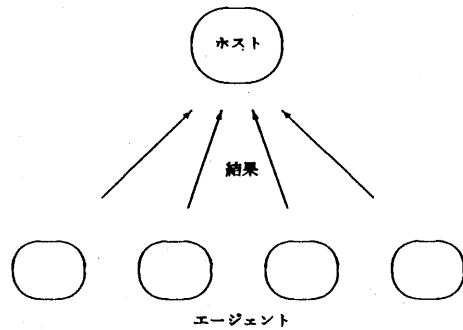
並列計算

エージェントはローカルソートの後、マージングを行う。(S-2,S-3)



データの収集

ホストはソート済みのデータをエージェントから回収する。(S-4)



このアルゴリズムでは、S-1とS-4で、ホストとエージェントの間での通信が発生する。また、S-3でエージェント同士の通信が発生する。これらは直接、粒度や効率に影響を与える。S-2の処理では通信が全く発生しない。全体の通信の量は、エージェント間の同期を考慮しなければ、データの量だけに比例する。エージェントの数が増えた時、同期のコストが大きく変化するなら、粒度や効率に大きな影響を与えるであろう。また、ホストはエージェントにデータを分配した後、エージェントからソート済みのデータを受け取るまで、何もしないで待っている。

4.3 実験1：数値データの Block Odd-Even Sort

前節のアルゴリズムに従って、数値データをソートする実験を行った。実験に使用したデータは整数の乱数で、その数は5000個から50000個まで、5000個おきに測定した。また、エージェントの数も8台から28台まで変化させた。図4.3.1～図4.3.4において、グラフ中にプロットされているアルファベットa～jは、データの個数を表し、それぞれa:5000, b:10000, c:15000, d:20000, e:25000, f:30000, g:35000, h:40000, i:45000, j:50000であることを示す。

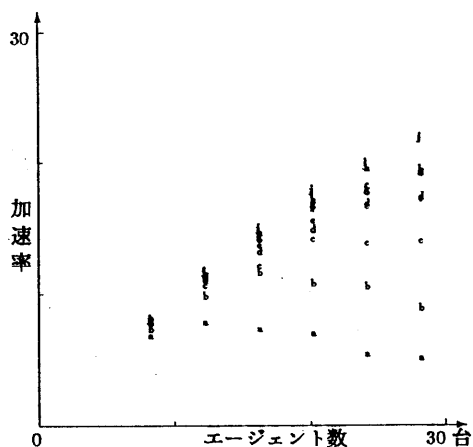


図 4.3.1 エージェント数と加速率

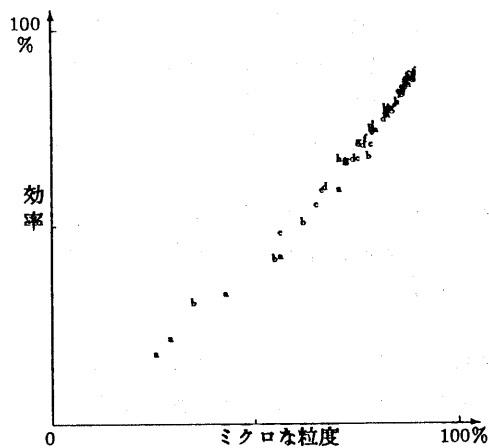


図 4.3.4 ミクロな粒度と効率の分布

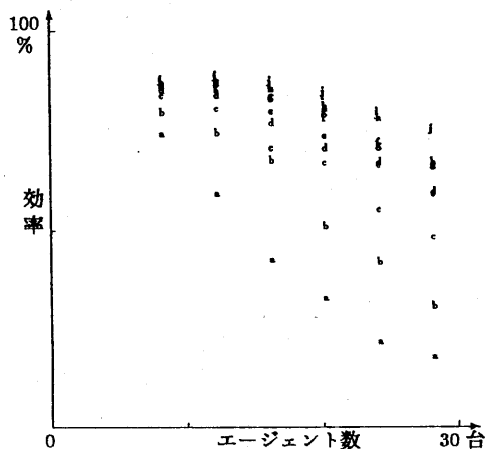


図 4.3.2 エージェントの数と効率

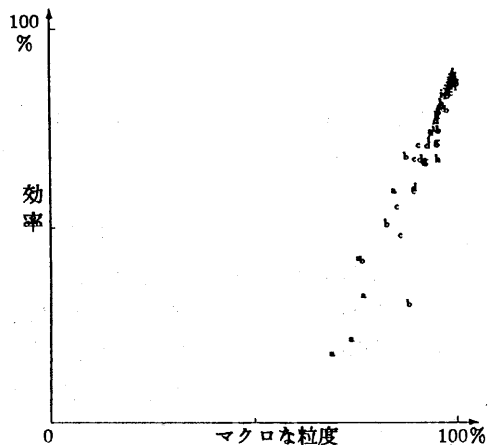
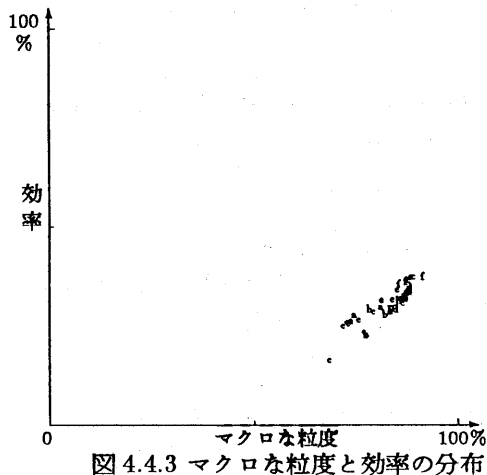
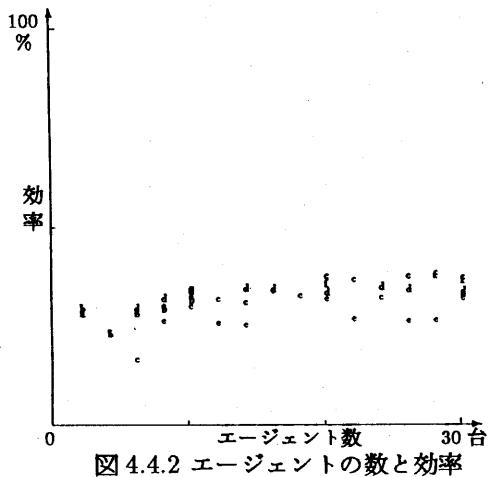
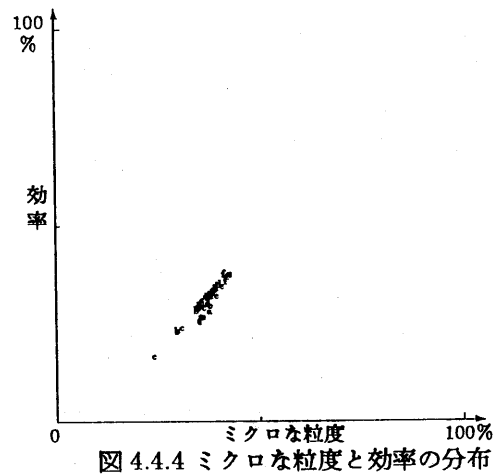
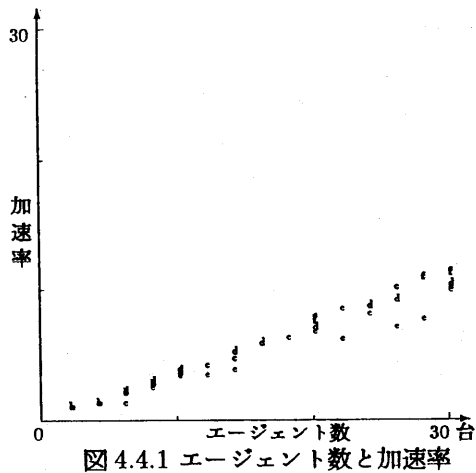


図 4.3.3 マクロな粒度と効率の分布

4.4 実験2：大量文書データの Block Odd-Even Sort

次に、数値データとは粒度の分布が異なると予想される大量の文書データ中の各文を一つのレコードとして、それらのソートを行った。実験では約1Mバイトから約42Mバイトまでの長さの文書を用いた。データが大きいため、オンメモリでは処理できず、ディスクへのアクセスが行われる点が実験1とは異なる。このことが粒度の値に影響するかも知れない。一つの文の大きさの平均は約40バイトである。実験1と同じく、図4.4.1～図4.4.4においてプロットされているアルファベットa～fはデータの大きさを示しており、それぞれ約1MB, 2MB, 3MB, 10MB, 20MB, 42MBである。



4.5 実験結果の考察

2つの実験結果のグラフを比較してみよう。

- エージェント数と加速率、効率
(図 4.3.1, 図 4.3.2, 図 4.4.1, 図 4.4.2 参照)

実験1はだいたい予測される通り、エージェントの台数が増えるに従って少しずつ効率が低下している。実験2では、効率はほぼ横ばいであるが、実験1に比べてバラツキが大きい。バラツキの大きい原因は2つ考えられる。

- － 文書データを分割する際、丁度同じ大きさ同士にはならないため、エージェント間の処理時間にバラツキが生じ、エージェント間の同期のロスが大きくなる。(処理の不均一 [小柳 89].)
- － 全処理時間に占める通信時間の割合が大きいため(ミクロな粒度が小さいため)、通信回線の込み具合の影響を受けやすい。

- マクロな粒度と効率(図 4.3.3, 図 4.4.3 参照)

実験1も実験2もマクロな粒度はかなり

高い所に分布が集中している。このことは、どちらもデータの分配や収集にあまりコストがかからない処理であることを示している。効率、実験1がかなり高い所に多く分布しているのに対して、実験2は50%を超えたものは一つも無い。これはソートされるデータの性質を反映している。実験1の数値データに比べて実験2の文書データは一つ一つのレコードが大きい、ローカルソートよりもエージェント間の通信の時間の方がずっと長くなるのである。このことは次のミクロな粒度と効率の関係のグラフに現れている。

● ミクロな粒度と効率 (図 4.3.4, 図 4.4.4 参照)

実験1と実験2は、マクロな粒度の分布はほとんど同じなのだが、ミクロな粒度の分布には大きな差が現れている。この差が効率の差となって現れるのである。

実験1も実験2も、マクロな粒度、ミクロな粒度にかかわらず、効率との間に正の相関関係が成り立っている。このことから、実験2の処理の効率を上げるためには、S-3の部分の改良すべきであると言える。

5 まとめ

Lukらは、4~8台のワークステーションから構成されるLAN上でソーティングを並列に実行し、使用したシステムに依存した形で結果を提示した[Luk 89]。我々は、より多数のワークステーションからなるLAN上で、分散計算の一般的な特性について調べた。本論文では、分散計算における粒度の値の定量的な定義を提示した。また、疎結合分散環境上で実際に分散計算の効率を測定し、その結果を基に考察を行った。その結果、データの分配、収集にあまり通信のコストがかからず、通信のコストが大きい

疎結合分散環境でも、並列化による処理速度の向上が期待できることが判った。また、分散計算を行ってみたが処理速度が向上しない場合、効率を測定しただけでは処理時間のネックがどこに存在するのか特定できない。本研究のようにミクロな粒度とマクロな粒度に着目すると、分散計算のどの部分がネックなのかを探す目安となる。

今後の課題としては、別の種類の問題についても分散計算を行い、粒度の値と効率とに関する特性について考察することや、粒度の分析結果に基づいて、分散計算の効率を向上させる方法について検討すること等がある。

参考文献

- [Bitton 84] Bitton, D., DeWitt, D.J., Hsaio, D.K. and Menon, J.: "A Taxonomy of Parallel Sorting", ACM Computing Surveys, Vol.16, No.3, pp.287-318, 1984.
- [Luk 89] W.S.Luk and Franky Ling: "An Analytic/Empirical Study of Distributed Sorting on a Local area Network", IEEE Transaction on Software Engineering, Vol.15, No.5, pp.575-586, May 1989.
- [Modi 88] Modi, J.J.: "Parallel Algorithms and Matrix Computation", Oxford University Press, 1988.
- [岡村 89] 岡村, 平原, 荒木: "UNIX ワークステーションによる 分散環境上での並列計算に関する実験", 第14回 jus シンポジウム, 1989年11月.
- [小柳 89] 小柳義夫: "数値計算の並列アルゴリズム", bit, Vol.21, No.4, pp.378-385, 1989.