システムソフトウェアと
オペレーティング・システム 65－7
（１９９４．７．２１）

超並列計算機におけるデータ分割/配置の最適化について

山家 陽　　村上和彰

九州大学 大学院総合理工学研究科 情報システム学専攻

〒816 春日市春日公園 6–1

E-mail: {yamaga, murakami}@is.kyushu-u.ac.jp

分散メモリマシンでは，グローバルなアドレス空間が存在しないため，分散メモリにプログラムデータ構造を分割・配置する必要がある．分割配置をどのように行なうかが，プロセッサ間のデータ転送および負荷分散等に影響を及ぼし，プログラムの実行時間の決定要因の一つとなり得る．本稿では，現在検討している，データの分割・配置問題について述べる．

Optimization of Data Partitioning and Allocation on Massively Parallel Processors

Akira Yamaga　　Kazuaki Murakami

Department of Information Systems

Interdisciplinary Graduate School of Engineering Sciences
Kyushu University

6–1 Kasuga-koen, Kasuga-shi, Fukuoka 816 Japan

E-mail: {yamaga, murakami}@is.kyushu-u.ac.jp

On the distributed memory multiprocessors , there is the absence of a single global address space. So, must distribute the program data structure on the distributed memories. The way of the data partioning and allocation influences the load balance,and data movement among processors , in turn decides the program excution time. We propose a new approach to the data pationing and allocation ploblem .

1 はじめに

1個のプログラムを（超）並列計算機上で並列実行する場合，

- **計　算**（*computation*）の　**分　割**（*partitioning, decomposition*），および，分割された計算（一般に，**タスク**（*task*）と呼ぶ）のプロセッサへの**スケジューリング**（各タスクをどのプロセッサで，どの時点で実行するかを決定すること）
- データの分割（*partitioning, decomposition*），および，分割されたデータのメモリへの**配置**（*allocation*）
- **データ転送**（*data transfer, data movement*）のための**コード生成**（*code generation*）および**スケジューリング**

といった操作が一般に必要である（図1参照）．

これらの操作は，対象とする（超）並列計算機が，

- **集中共有メモリ**（*centralized shared memory*）／**UMA**（*Uniform Memory Access*）モデル：物理的に一ヶ所に**グローバル・メモリ**として集中配置されたメモリを，プロセッサが通常の Read/Write 操作（LOAD/STORE，等）によりアドレス指定で直接的に読み書きすることで共有する．共有メモリ上の共有変数に対する読み書きという形でデータ授受を行なう．すべてのプロセッサから同一時間で集中共有メモリにアクセス可能である．
- **分散メモリ**（*distributed memory*）モデル：メモリをある単位（たとえば，プロセッサ）毎に**ローカル・メモリ**として分散配置する．
  - **分　散　共　有　メモリ**（*distributed shared memory*）／**NUMA**（*NonUniform Memory Access*）モデル：プロセッサが，それ自身のローカル・メモリだけでなく他のプロセッサに配置されたメモリ（**リモート・メモリ**）までも，通常の Read/Write操作（LOAD/STORE, PUT/GET，等）によりアドレス指定で直接的に読み書きすることで共有する．共有メモリ上の共有変数に対する読み書きという形でデータ授受を行なう．各プロセッサから見た共有メモリの距離はアドレスにより遠近が生じ，アクセス時間も異なる．
  - **メッセージ交換**（*message passing*）**NORA**（*NO Remote Access*）モデル：プロセッサが通常の Read/Write 操作によりアドレス指定で直接的に読み書きできるのは，それ自身のローカル・メモリだけで，リモート・メモリに対しては出来ない．すなわち，共有メモリ上の共有変数に対する読み書きという形でデータ授受を行なうのではなく，メッセージの交換という形でデータ授受を行なう．

のいずれでも必要である．

また，これらの操作は，

- プログラム実行前（**静的**）に，
  - **プログラマ**が，および／あるいは，

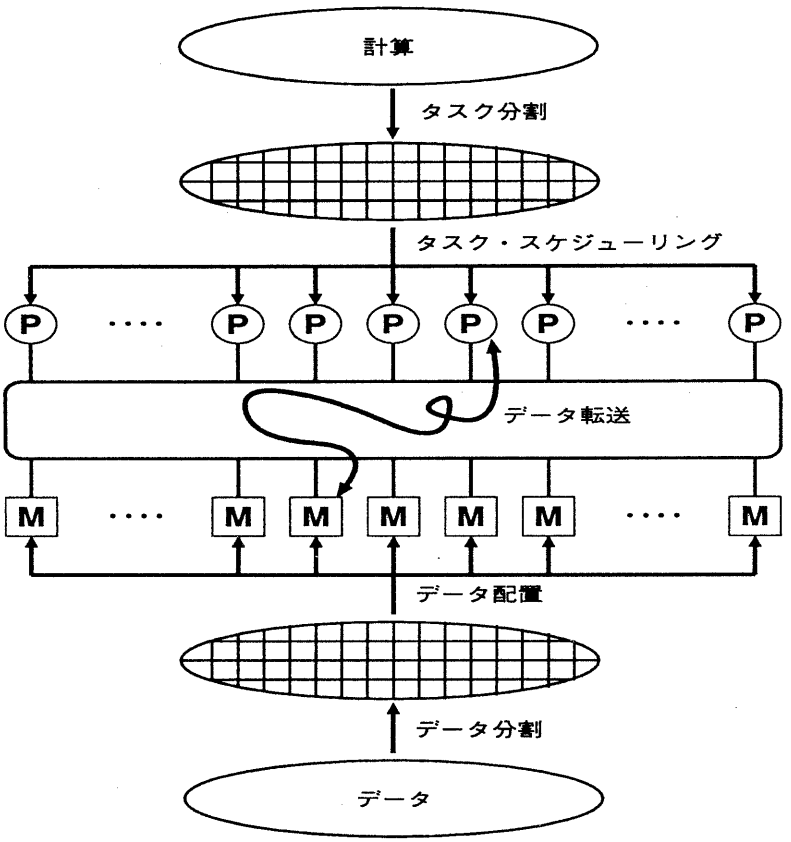

図 1: 並列実行に必要な操作

– コンパイラが

行なう，および／あるいは，

- プログラム実行時（動的）に，
  - プログラム自身が，
  - ランタイム・ソフトウェアが，および／あるいは，
  - ハードウェアが（それ自身の論理で）行なう

ことが選択肢として可能である．

さらに，これらの操作は，相互に密接に関わり合っている．すなわち，ある操作の結果が他の操作に影響を与え，その逆もまたしかりである．

現在，我々は，科学技術計算プログラムを対象にして，『コンパイラ（および，ランタイム・ソフトウェアならびにプログラム自身）によるデータ分割／配置およびデータ転送の最適化』について検討を行なっている [1]．本稿ではそのうち，『コンパイラによる静的なデータ分割／配置の最適化』について検討する．

まず，2章で，我々が現在検討している『コンパイラによる静的なデータ分割／配置およびデータ転送の最適化』問題を一般化した場合の定義を与える．3章で，本稿で取り上げる問題に関連する研究を紹介する．そして，4章で，我々が現在解こうとしている問題を定義する．

2 一般化された問題の定義

2.1 データ分割／配置の最適化

『コンパイラによる静的なデータ分割／配置の最適化』問題を一般化すると，次のように定義できる．

すなわち，プログラムおよびデータが与えられた時に，プログラム実行時間を最小とするように，データ分割／配置

$$\delta : \mathcal{D} \rightarrow \mathcal{M} \times \mathcal{T}$$

を求めることである．ここで，

- $\mathcal{D}$：データを要素とするデータ空間（*data space*）
- $\mathcal{M}$：メモリ・モジュールを要素とするメモリ空間（*memory space*）
- $\mathcal{T}$：時空間（*time sapece*）

である．

メモリ空間 $\mathcal{M}$ の各要素は必ずしも物理的なメモリ・モジュールに対応している必要はない．物理的なメモリ・モジュールと1対1に対応していない場合は，単に『データ分割』と呼ぶ．

プログラムの実行中，データ配置が一切変わらない場合は，データ分割／配置

$$\delta : \mathcal{D} \rightarrow \mathcal{M}$$

を求めればよい．

2.2 データ転送の最適化

『コンパイラによる静的なデータ転送の最適化』問題を一般化すると，次のように定義できる．

すなわち，タスク・スケジュール

$$\rho : \mathcal{C} \rightarrow \mathcal{P} \times \mathcal{T}$$

および，データ分割／配置δが与えられた時，プログラム実行時間を最小とするように，データ転送コードを生成し，スケジュール

$$\sigma : \mathcal{CC} \rightarrow \mathcal{P} \times \mathcal{T}$$

を求めることである．ここで，

- $\mathcal{C}$：計算を要素とする計算空間（*computation space*）
- $\mathcal{CC}$：$\mathcal{C}$に生成したデータ転送コードを加えた結果得られる計算空間
- $\mathcal{P}$：プロセッサを要素とするプロセッサ空間（*processor space*）

である．

データ転送コードの生成およびスケジューリングに当たっては，一般に以下の変数を決定する必要がある．

- 誰が転送するのか?: データの生産者，消費者，あるいは，第3者が転送するのか?
- いつ転送するのか？

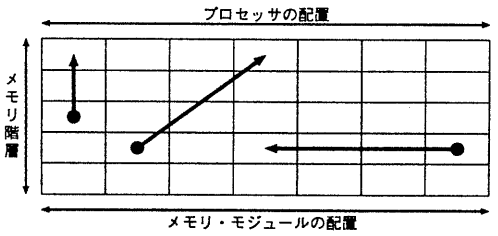

図 2: どこからどこへ？

- どのデータを転送するのか？
- どこからどこへ転送するのか?: 水平方向にはプロセッサおよびメモリ・モジュールの配置をとり，垂直方向にはメモリ階層（レジスタ，キャッシュ，メモリ，等）をとるような 2 次元空間（図??参照）において，どの点からどの点へ転送するのか？
- どのような経路で転送するのか？

3 関連研究

科学技術計算プログラムの並列実行を対象にした『コンパイラによる静的なデータ分割／配置の最適化』については，これまでに多くの研究が行なわれている．

Li と Chen[10] は，データ分割／配置問題を次の**『添字領域整列化**（*index domain alignment*）**』**問題の形で定義している．すなわち，配列間の相互参照に起因するデータ転送のコストを最小とするように，配列の添字領域（index domain）を 1 個の共通の添字領域に写像するための整列化（*alignment*）関数の集合を求める問題である．整列化関数としては，添字領域の異なる要素間での置換（permutation）や埋め込み（embedding），および，ある要素内でのシフトや反転（reflection）を用いる．本問題は，最初に添字領域の要素間での整列化（置換や埋め込み）を対象とし，次に要素内での整列化（シフトや反転）を対象とすることで解ける．そこで，問題の中心は，添字領域の異なる要素間の整列化，つまり，次の**要素整列化**（*component alignment*）問題に集約される．これは，添字領域の要素間のアフィニティ（affinity）を表す要素アフィニティ（component affinity）グラフ CAG（頂点は要素を表し，要素間にアフィニティがあれば対応する頂点間に枝を張る．枝には，非競合枝と競合枝があり，非競合枝には重み 1 が，競合枝には重み ϵ ($\ll 1$) が付けられる）が与えられた時，CAG の列（同一の添字領域に属する要素に対応する頂点から成る集合）の頂点数のうち最大のものを n として，CAG の頂点集合 V を互いに重複しない n 個の部分集合 V_1, V_2, ..., V_n に分割する問題である．この時，同じ列に属する頂点は同じ部分集合に属さないように，また，異なる部分集合に属する頂点の間に張られている枝の重みの合計が最小となるように分割する．

Gupta と Banerjee[8] は，上記の Li と Chen[10] のアイデアを基に，制約条件ベース（*constraint based*）のアプローチを提案している．個々のデータ分散（*distribution*）が満たした方が望ましい性質を制約条件と呼び，プログラム全体を通してこれらすべての制約条件が矛盾しないようにデータ分散を行なう．もし，ある 2 つのデータ分散に関する制約条件が互いに矛盾して競合を生じている場合は，それぞれの制約条件に付随しているクオリティ・

メジャー（*quality measure*：たとえば，その制約条件が満たされない場合に実行時間に与えるペナルティ）を基に，プログラム実行時間が最小となるように競合を解決する．

以上の研究は，プログラム全体を対象としたデータ分割／配置の最適化であり，また，その分割形態もブロックかサイクリックのいずれかである．

これに対して，Ramanujam と Sadayappan [12] は，1個のループ・ネストを対象として，通信を伴わない配列分割（*communication-free array partition*）が存在するか否かという問題を定義している．そして，配列添字がループ制御変数のアフィン関数であるような**doall** ループのクラスに対して，通信を伴わない配列分割が存在するための十分条件を示している．

以上のすべての研究において，計算の分割／配置（タスク分割およびタスク・スケジューリング）は，*Owner-Computes Rule*（代入式の左辺のデータを所有するプロセッサが，当該代入式の右辺の計算を行なう）に従っている．すなわち，データ分割／配置が決まると，それに従って計算の分割／配置も一意に決まる．

これに対して，Anderson と Lam[6]，および，Amarasinghe と Lam[5] は，データ分割（data decomposition）と計算分割（computation decomposition）とを同時に最適化することを目的とし，**doall** ループに対する問題の定式化および解法アルゴリズムを提案している．ただし，異なるループ・ネストに対するデータ分割が競合する場合，**データ再分割**（*data reorganization*）を行なう必要がある．

4 問題の定義

Li と Chen[10]，および，Gupta と Banerjee[8] で求めるデータ分割/配置は

$$\delta : \mathcal{D} \rightarrow \mathcal{M}$$

すなわち，プログラム全体を通して変わらない．データ再分割のオーバヘッドは生じないが，個々のループ・ネストに限って見ると必ずしも最適なデータ配置になっていない可能性がある．

一方，Anderson と Lam[6]，および，Amarasinghe と Lam[5] で求めるデータ分割は，

$$\delta : \mathcal{D} \rightarrow \mathcal{M} \times \mathcal{T}$$

であり，プログラム実行の途中でデータ分割が変わる，つまり，データ再分割の可能性がある．個々のループ・ネストに限って見ると最適なデータ分割になっているかも知れないが，データ再分散のオーバヘッドが加わるためプログラム全体を通すと必ずしも最適とはならない可能性がある．

そこで，我々は，以下のデータ分割/配置を求めることにする．

$$\delta : \mathcal{D} \times \mathcal{M} \rightarrow \mathcal{M}$$

すなわち，ある時間ステップにおけるデータ分割/配置から，次の時間ステップにおける最適なデータ分割/配置を求める．δは，再帰関数となる．これは，「データは移動するものである」という認識の下で，データ再分割をも最適化したデータ分割/配置を求めようというものである．

5 おわりに

以上，一般化されたデータ分割/配置問題を定義し，関連研究を整理して，筆者らが現在検討中のデータの分割・配置問題について述べた．今後は，本問題を定式化し，その解法アルゴリズムを開発する予定である．

謝辞

日頃ご討論頂く，九州大学 大学院総合理工学研究科 安浦寛人 教授，岩井原瑞穂 助手，ならびに，國貞勝弘 氏をはじめとする安浦研究室の諸氏に感謝します．資料を提供頂いた富士通研究所 進藤達也 氏に感謝致します．

本研究は一部，文部省科学研究費補助金 重点領域研究「超並列原理に基づく情報処理体系」による．

参考文献

[1] 國貞勝弘, 村上和彰, "超並列計算機におけるデータ転送の最適化について," 情処研報, OS-65-8, 1994 年 7 月.

[2] 進藤達也, 岩下英俊, 土肥実久, 萩原純一, 金城ショーン, "Twisted Data Layout," 並列処理シンポジウム *JSPP'94* 論文集, pp.161-168, 1994 年 5 月.

[3] 本多弘樹, "自動並列化コンパイラ," 情報処理, vol.34, no.9, pp.1150-1157, 1993 年 9 月.

[4] 吉田明正, 前田誠司, 尾形 航, 笠原博徳, "マルチグレイン並列処理におけるデータローカライゼーション手法," 並列処理シンポジウム *JSPP'94* 論文集, pp.145-152, 1994 年 5 月.

[5] Amarasinghe, S. P. and Lam, M. S., "Communication Optimization and Code Generation for Distributed Memory Machines," *Proc. 1993 Conf. on Programming Language Design and Implementation*, pp.126-138, Jun. 1993.

[6] Anderson, J. M. and Lam, M. S., "Global Optimization for Parallelism and Locality on Scalable Parallel Machines," *Proc. 1993 Conf. on Programming Language Design and Implementation*, pp.112-125, Jun. 1993.

[7] Balasundaram, V., Fox, G., Kennedy, K., and Kremer, U., "A Static Performance Estimator to Guide Data Partitioning Decisions," *Proc. 3rd Symp. on Principles & Practice of Parallel Programming*, pp.213-223, Apr. 1991.

[8] Gupta, M. and Banerjee, P., "Demonstration of Automatic Data Partitioning Techniques for Parallelizing Compilers on Multicomputers," *IEEE Trans. on Parallel and Distributed Systems*, vol.3, no.2, pp.179-193, Mar. 1992.

[9] Hudak, D. E. and Abraham S. G., "Compiler Techniques fot Data Partitioning of Sequentially Iterated Parallel Loops," *Proc. 1990 Int. Conf. on Supercomputing*, pp.187-200, Jun. 1990.

[10] Li, J. and Chen, M., "Index Domain Alignment: Minimizing Cost of Cross-Referencing Between Distributed Arrays," *Proc. Frontiers'90: The 3rd Symp. on the Frontiers of Massively Parallel Computation*, pp.424–433, Oct. 1990.

[11] Li, J. and Chen, M., "Compiling Communication-Efficient Programs for Massively Parallel Machines," *IEEE Trans. on Parallel and Distributed Systems*, vol.2, no.3, pp.361–376, Jul. 1991.

[12] Ramanujam, J. and Sadayappan, P., "Compile-Time Techniques for Data Distribution in Distributed Memory Machines," *IEEE Trans. on Parallel and Distributed Systems*, vol.2, no.4, pp.472–482, Oct. 1991.