二層タスク並列スケルトンプログラムの最適化に関する研究

数理第 7 研究室 大川 徳之
指導教員 武市 正人 教授

1 はじめに

スケルトン並列プログラミングに於いて，データ並列スケルトンとタスク並列スケルトンを併用する場合，プロセッサの割り当て方とタスク並列スケルトンの組み合わせ方によってプログラムの効率が大きく変化する．タスク並列スケルトンのみの場合に関しては Aldinucci と Danelutto による最適化手法 [1] が存在するが，データ並列スケルトンを併用する場合に関しては言及されておらず，通用するかは自明ではなかった．我々は，データ並列スケルトンを併用する場合に対しても最適化を行うアルゴリズムを提案し，実際に本手法を最適化機構として持つ並列スケルトンライブラリを実装した．

2 二層タスク並列スケルトンプログラム

本手法に於いて，対象とする並列スケルトンプログラム σ は以下の言語で与えられるものである．このようなデータ並列スケルトンとタスク並列スケルトンの併用モデルを二層モデルと呼ぶ．[2]

$$
\begin{array}{rcll}
\sigma & ::= & @f & \text{(atomic スケルトン)} \\
 & | & \sigma \mid \sigma & \text{(pipe スケルトン)} \\
 & | & \diamond\sigma & \text{(farm スケルトン)} \\
f & ::= & \mathrm{Seq} & \text{(逐次関数)} \\
 & | & f; f & \text{(関数結合)} \\
 & | & \mathrm{map}_f & \text{(map スケルトン)} \\
 & | & \mathrm{zip}_{\oplus} & \text{(zip スケルトン)} \\
 & | & \mathrm{reduce}_{\boxplus} & \text{(reduce スケルトン)} \\
 & | & \mathrm{scan}_{\boxplus} & \text{(scan スケルトン)}
\end{array}
$$

2.1 データ並列スケルトン

本手法で対象とするデータ並列スケルトンは，map，zip，reduce，scan の 4 つである．これらのデータ並列スケルトンが P 個のプロセッサによって実行されたときの並列計算時間 $\mathrm{T}_{\mathrm{comp}}[\![f]\!]_P$ は，次のように単純化できる．$\mathrm{T}_{\mathrm{comp}}[\![f]\!]_P = \mathrm{T}_{\mathrm{const}}[\![f]\!] + \mathrm{T}_{\mathrm{local}}[\![f]\!]/P + \lceil \log P \rceil \mathrm{T}_{\mathrm{comm}}[\![f]\!]$ 式中で，$\mathrm{T}_{\mathrm{const}}[\![f]\!]$ は並列実行される部分以外の計算時間，$\mathrm{T}_{\mathrm{local}}[\![f]\!]$ は各プロセッサで独立に並列実行される部分の計算時間，$\mathrm{T}_{\mathrm{comm}}[\![f]\!]$ はプロセッサ間通信を伴う並列実行部分の計算時間である．

2.2 タスク並列スケルトン

本手法で対象とするタスク並列スケルトンは，atomic，pipe，farm の 3 つである．これらのタスク並列スケルトンはストリームをひとつ入力に取り，ひとつのストリームを出力する．

タスク並列スケルトン atomic はストリームの各要素毎に関数を適用する．関数 f を適用する atomic を $@f$ と記述する．

タスク並列スケルトン pipe はタスク並列スケルトンを結合する．ふたつのタスク並列スケルトン σ_1，σ_2 を結合する pipe を $\sigma_1 \mid \sigma_2$ と記述する．

タスク並列スケルトン farm はタスク並列スケルトンを多重化する．タスク並列スケルトン σ を多重化する farm を $\diamond\sigma$ と記述する．タスク並列スケルトン $\diamond\sigma$ は，入出力部にそれぞれプロセッサがひとつづつ割り当てられ，中に σ を複数個配置する．複数個ある σ のうち，暇になったものが入力部からストリームの要素を取得し処理が行われる．ここで，各 σ を行うプロセッサ集合と入出力部に割り当てられるプロセッサは独立である．また，同時にふたつ以上の σ が入力部からストリームの要素を取得することはできないと仮定する．

3 サービスタイム

定義 1 (サービスタイム)**.** タスク並列スケルトンプログラム σ が，P プロセッサを持つ並列計算機環境に於いて，ストリーム $\langle x_1, x_2, \ldots \rangle$ を処理するとする．x_i の処理が終了し出力された時点を t_i とするとき，タスク並列スケルトンプログラム σ のサービスタイム $\mathrm{T}_{\mathrm{s}}[\![\sigma]\!]_P$ を

$$\mathrm{T}_{\mathrm{s}}[\![\sigma]\!]_P = \lim_{n\to\infty} \frac{\sum_{i=1}^{n}(t_{i+1} - t_i)}{n-1} = \lim_{n\to\infty} \frac{t_n - t_1}{n-1}$$

と定義する． □

ストリーム長が十分に長ければ，要素ひとつあたりの処理時間はサービスタイムへと漸近する．本手法ではこのサービスタイムを効率の指標として用いる．

タスク並列スケルトン $@f$ のサービスタイムを定式化するため，5 つのパラメータ $\mathrm{Procs}[\![f]\!]$，$\mathrm{T}_{\mathrm{in}}[\![f]\!]$，$\mathrm{T}_{\mathrm{const}}[\![f]\!]$，$\mathrm{T}_{\mathrm{local}}[\![f]\!]$，$\mathrm{T}_{\mathrm{comm}}[\![f]\!]$ を使う．ここで，$\mathrm{Procs}[\![f]\!]$ はプロセッサ数の制約であり，データを全てキャッシュあるいはメモリに載せきることのできる最低限のプロセッサ数を表わしている．$\mathrm{T}_{\mathrm{in}}[\![f]\!]$ は f の入力を取得するための時間である．これらのパラメータを用いて，P 個のプロセッサを用いたときのタスク並列スケルトン $@f$ のサービスタイム $\mathrm{T}_{\mathrm{s}}[\![@f]\!]_P$ を

$$\mathrm{T}_{\mathrm{s}}[\![@f]\!]_P = \mathrm{T}_{\mathrm{in}}[\![@f]\!] + \mathrm{T}_{\mathrm{comp}}[\![f]\!]_P$$

$$\mathrm{T}_{\mathrm{comp}}[\![f]\!]_P = \begin{cases} \infty & (P < \mathrm{Procs}[\![f]\!]) \\ \mathrm{T}_{\mathrm{const}}[\![f]\!] + \mathrm{T}_{\mathrm{local}}[\![f]\!]/P + \lceil \log P \rceil \mathrm{T}_{\mathrm{comm}}[\![f]\!] & (P \geq \mathrm{Procs}[\![f]\!]) \end{cases}$$

と定式化した．プロセッサ数が $\mathrm{Procs}[\![f]\!]$ 以下となったときサービスタイムを無限大とした．なぜなら，他の計

算に比してスワップやキャッシュミスには膨大な膨大な時間がかかると考えられるからである．

他のタスク並列スケルトン pipe や farm は，

$$\mathrm{T_s}[\![\sigma_1 \mid \sigma_2]\!]_P = \min_{1\le p<P} \max\left\{\mathrm{T_s}[\![\sigma_1]\!]_p, \mathrm{T_s}[\![\sigma_2]\!]_{P-p}\right\}$$

$$\mathrm{T_s}[\![\diamond\sigma]\!]_P = \max\left\{\mathrm{T_{in}}[\![\sigma]\!], \min_{1\le pw+2\le P} \frac{\mathrm{T_s}[\![\sigma]\!]_p}{w}\right\}$$

と定式化した．

4 最適化アルゴリズム

サービスタイムを最小化したいプログラム σ は，関数 f_0，f_1，…，f_{N-1} をこの順でストリームの各要素に適用するプログラムであるとする．また，各 f_i に関して，$@f_i$ のサービスタイムを定めるに足るパラメータ，$\mathrm{Procs}[\![f_i]\!]$ と $\mathrm{T_{in}}[\![f_i]\!]$，$\mathrm{T_{const}}[\![f_i]\!]$，$\mathrm{T_{local}}[\![f_i]\!]$，$\mathrm{T_{comm}}[\![f_i]\!]$ が与えられているとする．プロセッサを p 個まで用いることができるとき，サービスタイムが最小となり且つ関数 f_i，f_{i+1}，…，f_j をこの順でストリームの各要素に適用するようなタスク並列スケルトンプログラムを $\widetilde{\sigma}_{i,j,p}$ とする．プログラム $\widetilde{\sigma_{i,j,p}}$ のサービスタイム $st_{i,j,p} = \mathrm{T_s}[\![\widetilde{\sigma}_{i,j,p}]\!]_p$ は以下の漸化式で与えることができる．

$$st_{i,j,p} = \begin{cases} \min\left\{ \begin{array}{l} \mathrm{T_{in}}[\![f_i]\!] + cc_{i,i,p}, \\ \mathrm{T_{in}}[\![f_i]\!] + \min\limits_{1\le wp'<p-2} \dfrac{st_{i,i,p'}}{w} \end{array} \right\} & (i=j) \\ \min\left\{ \begin{array}{l} \mathrm{T_{in}}[\![f_i]\!] + cc_{i,j,p}, \\ \min\limits_{i\le k\le j} \min\limits_{1\le p'<p} \max\left\{ \begin{array}{l} st_{i,k,p'}, \\ st_{(k+1),j,(p-p')} \end{array} \right\}, \\ \mathrm{T_{in}}[\![f_i]\!] + \min\limits_{1\le wp'<p-2} \dfrac{st_{i,j,p'}}{w} \end{array} \right\} & (i<j) \end{cases}$$

$$cc_{i,j,p} = \begin{cases} \mathrm{T_{comp}}[\![f_i]\!]_p & (i=j) \\ cc_{i,i,p} + cc_{(i+1),j,p} & (i<j) \end{cases}$$

ただし，$cc_{i,j,p}$ はプロセッサ数 p を用いて関数 $f_i; f_{i+1}; \cdots; f_j$ を実行するときの並列計算時間である．

この漸化式は動的計画法で解くことができる．並列計算機環境で使用できるプロセッサ数を P 個とすると，$st_{0,N-1,P}$ を構築する際の記録を逆順に辿ることで，$\widetilde{\sigma}_{0,N-1,P}$ を構成できる．また，この漸化式を解く際の時間計算量は与えられた関数の個数 N と並列計算機環境のプロセッサ数 P に対して $O(N^3P^2)$ となる．

5 ライブラリの実装

本手法に基いて，実行時にサービスタイムが最小になる組み合わせへと自身を組み変えるタスク並列スケルトンライブラリを実装した．データ並列スケルトンは SkeTo のものを利用している．図 1 は本ライブラリの動作である．図 2 は実験結果である．

1. **テスト実行段階 (Pipelining & Test) :** ストリームの第 1 要素を処理し，プログラマが与えた各関数 f_0，f_1，…，f_{N-1} の実行時情報を収集する．
2. **最適化段階 (Optimize) :** 測定した実行時情報を用いて漸化式を解き，サービスタイムが最小となるタスク並列スケルトンプログラムを構成する．

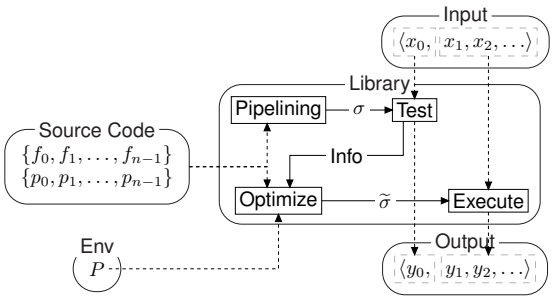

図 1: ライブラリの動作

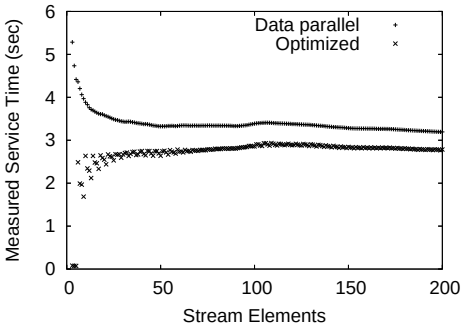

図 2: レイトレーシングを行うプログラムによる実験結果．データ並列スケルトンのみの実測サービスタイムと，最適化されたものによる実測サービスタイムの比較．

3. **本実行段階 (Execute) :** 構成したプログラムを用いてストリームの第 2 要素以降を処理する．

6 まとめと今後の課題

データ並列スケルトンとタスク並列スケルトンを併用した二層モデル上の並列スケルトンプログラムを対象として，サービスタイムを定式化した．また，その定式化の上でサービスタイム最小の並列スケルトンプログラムの導出法を与えた．さらに，本手法を最適化機構として持つ二層タスク並列スケルトンライブラリを実装した．

本手法にて扱うことが可能であるタスク並列スケルトンは，ひとつのストリームからひとつのストリームを生成するような，構造を持ったものに限られている．しかし，並列スケルトンの並列性に関する表現力を高めるためには，複数のストリームから複数のストリームを生成するようなタスク並列スケルトンや，ストリームの要素ひとつに対し出力の個数が増減するようなタスク並列スケルトンが扱えるとよい．このような並列スケルトンを導入した場合の最適化手法は課題のひとつである．

参考文献

[1] M. Aldinucci and M. Danelutto. Stream parallel skeleton optimization. In *Proceedings of the 11th IASTED ICPDCS*. IASTED/ACTA press, 1999.

[2] H. Kuchen and M. Cole. The integration of task and data parallel skeletons. In *Proceedings of 3rd International Workshop on CMPP*, pages 3–16, 2002.