情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2009-SE-164 No.24
Vol.2009-EMB-13 No.24
Vol.2009-CSEC-45 No.24
2009/5/29

実行条件解析による並列化スケジューリングの効率化

南出英明[†]　　佐々木賢一[†]

概要：消費電力を抑えつつ高い実行性能を得るには，マルチプロセッサによる並列処理が有効である．プログラムの並列化では，プログラムをタスクに分割し，各タスクの実行時間とタスク間の依存関係を基に，総実行時間が最短となるようにスケジューリングを行う．一般に，タスクの実行時間は入力に依存し，各タスクの実行時間の間には相関がある．このため，ある入力に対して最適なスケジューリングが，他の入力に対しても最適であるとは限らない．そこで本研究では，全ての入力に対して，最悪実行時間が最短となるものを最適なスケジュールと定義する．スケジューリングや最悪実行パスの解析は NP 問題であるが，タスク間での実行時間の相関を解析し計算量を抑制する．

An Efficient Scheduling for Multiprocessor System using Condition Analysis

Hideaki Minamide[†] and Kenichi Sasaki[†]

Abstruct : Parallel processing on multiprocessors is a promising approach to perforr enhancement with low-power consumption. When parallelizing a program for multiprocessor program is first partitioned into a set of tasks, and then the tasks are scheduled based on execution times and inter-dependencies in order to minimize the total execution time o multiprocessors. In general, the execution times of tasks depend on their inputs, and there correlations among the execution times of the tasks. Therefore, an optimal scheduling for a c input is not always optimal for a different input. In this paper, we define an optimal schedule one whose worst-case execution time is the minimum. In order to obtain the optimal sch scheduling and worst-case path analysis, each of which is an NP problem, need to be simultaneously. This paper proposes an efficient technique for this problem, which car analyzes the correlations of tasks' execution times to reduce the solution space.

† 三菱電機株式会社　先端技術総合研究所
Mitsubishi Electric Corporation　Advanced Technology R&D Center

1. はじめに

近年，消費電力を抑えつつ，高い実行性能を得る方法として，マルチプロセッサやマルチコア構成による並列処理が注目されている．これらの並列処理に対応したハードウェア上でプログラムを効率良く実行するには，並列化スケジューリングが必要になる．この並列化スケジューリングでは，プログラムをタスクと呼ばれる部分プログラムへ分割し，各タスクの実行時間とタスク間の先行制約を基に，タスク全体の実行時間（スケジュール長）が最短となるように Processing Element（以下 PE）へ割り付けを行う．

一般に，機器に組み込み，その動作を制御するプログラムは，外部からの入力信号や内部状態を実行条件として扱い，その実行条件に対応する処理の組合せとして記述できる．本研究では，このようなプログラムの並列化スケジューリングを対象とする．本書では，入力信号や内部状態といった各処理の実行条件と，その実行条件下での処理の組合せを単位として部分プログラムへ分割し，１つのタスクとして扱う．ただし，このようなタスクで構成されるプログラムを並列化の対象にすると，実行条件によって個々のタスクの実行時間が異なるため，最適なスケジューリング結果を一意に決めることができない，という問題がある．実行条件を考慮せず，個々のタスクの最悪実行時間（以下 WCET）を用いて並列化スケジューリングを行う方法では，最適なスケジューリング結果は得られない．

このため本研究では，全ての実行条件下において WCET が最短になるスケジューリング結果を，最適なスケジュールと定義する．この最適なスケジュールは，全てのタスクの実行条件の真偽の組合せに対して並列化スケジューリングを行い，各々のスケジュールに対して全ての実行条件下におけるタスクの実行時間を割り当て，スケジュール長が最短となるスケジュールを選定することで得る．この方法では，プログラムが持つタスク数に対し，２のタスク数乗通りのスケジューリングを行うため，タスク数の増加に伴い多大な計算時間を必要とする．

この問題を解決するため，各タスクが持つ実行条件について，タスク間での相関関係を解析することで，並列化スケジューリングの対象となるタスクの組合せを削減し，タスク数の増加に伴う計算量の増大を抑制する方法を開発した．

本書では，条件解析によるスケジューリング対象の限定と，スケジューリングに要する計算量抑制の効果について述べる．また，条件解析の対象とするタスク数を削減し，計算量を抑える方法についても述べる．

Vol.2009-SE-164 No.24
Vol.2009-EMB-13 No.24
Vol.2009-CSEC-45 No.24
2009/5/29

2. 関連研究

プログラム内の一まとまりの処理をタスクとし，タスクを節点，タスク間の先行制約を有向辺で表すことで，並列計算をタスクグラフとしてモデル化することができる．タスクグラフの例を図 1 に示す. タスクグラフは, 閉路のない重み付き有向グラフである．タスク間の先行制約は，タスク間にデータ依存が存在する場合に生じる．また，() 内の数値はタスクの重みであり，タスクの実行時間を表す.

並列化スケジューリング問題とは，図 1 に示すようなタスクグラフを対象に，複数の PE に対して，割込み処理がないという条件の下で，スケジュール長を最短にすることを目的として，タスクの割り付けを決定する最適化問題である[1][2]．このように，一般的な並列化スケジューリング問題では，各タスクの実行時間は固定されている．

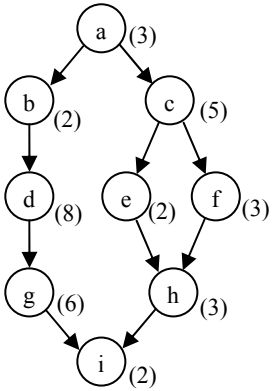

図 1　タスクグラフ

一方，タスクの実行時間が変化する状況を扱った並列化スケジューリングの研究[9]がある．タスクの実行時間を確率分布関数で表現し，遺伝的アルゴリズムを組合せたリストスケジューリングによって，並列化スケジューリングを行う．ただし，個々のタスクの実行時間は確率的に扱われており，タスク間で実行時間が相関関係を持つ場合については考慮されていない.

また，タスク間の相関関係を利用した WCET の研究として，シングルプロセッサ構成を対象に，WCET の精度を向上する目的で，非実行パスを考慮した WCET の計算方法が提案されている[4][5][6]．これらは，各基本ブロックの実行回数と，基本ブロックにつながる入力パスと出力パスの合計値が等しくなる性質を利用し，これらの関係を制約として与え，整数線形計画問題（ILP）として，WCET を求めるものである．ただし，各基本ブロックの実行条件となる論理式を ILP で解析するのではなく，論理式を演算した結果として得られる実行回数を，制約として与えている.

さらに，実行時間が変化するタスクを対象に，スケジュール可能性を解析する研究[7][8]がなされている．タスクの実行時間を確率分布関数で表現することで，変化する実行時間を扱っているが，各タスクの実行条件やタスク間での実行時間の相関関係については考慮されていない.

本研究は，個々のタスクが実行条件を持ち，実行条件に応じて実行時間が異なるタスクで構成されるプログラムを対象に, 実行条件の相関関係を考慮して WCET が最短となる並列化スケジューリングを求めるものである.

3. 並列化スケジューリング

3.1 対象プログラム

本書では，図 2 に示すように実行条件の判定とその実行条件下での処理の組合せで構成されるプログラムを，並列化スケジューリングの対象とする．そして，図 3 に示す複数の PE と共有メモリで構成される実行環境上で並列に実行する.

図 2 に示す様に，“条件判定 1”が真の場合は，then 節に相当する“処理 1－A”，“処理 1－B”，“･･･”の順に処理を実行する．一方，“条件判定 1”が偽の場合は，else 節に相当する”処理 1－a”や“処理 1－b”を実行する．そして，then 節あるいは else 節の処理を終えると，“条件判定 2”の判定処理へ進む．このように，“条件判定 1”の結果が，真の場合と偽の場合によって，処理内容の異なる then 節，else 節いずれかの処理を実行する．つまり実行条件によって，実行時間が異なるプログラムを対象とする.

本書では，図 2 に示すように，実行条件とその実行条件下での処理の組合せを，1 つのタスクとして扱う．更に，このタスクを並列化スケジューリングの基本単位として扱う．また，else 節に比べ，必ず then 節の実行時間が長いものとする.

実行環境は，図 3 に示す通り，複数の PE と共有メモリによって構成する．全ての PE がアドレス空間を共有し，かつ同一時間でアクセス可能な UMA 型（均一アクセスモデル）を想定する．分割後の部分プログラムやローカル変数は，ローカルメモリ上に配置する．部分プログラムに分割して，各 PE へ割当てたプログラムを並列に実行する際に

Vol.2009-SE-164 No.24
Vol.2009-EMB-13 No.24
Vol.2009-CSEC-45 No.24
2009/5/29

は，演算結果の整合性を保つために，部分プログラム間で同期処理が必要になる．本書で述べる並列化では，プログラムを並列化する際に必要に応じて同期イベントの受渡し処理を追加する．実行時には実行環境が持つ同期イベントの受渡し機能を用いて，異なる PE 上で実行している部分プログラム間で同期を取り，演算結果の整合性を保つ．

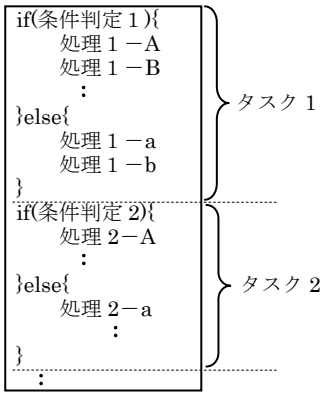

図 2　対象プログラム

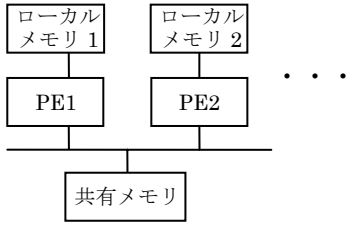

図 3　実行環境

3.2　実行時間の定式化

3.2.1 実行条件とタスクの実行時間

図 1 に示すタスクグラフを，次式で表現する．

$G = (\boldsymbol{V}, \boldsymbol{E}, w, c)$

v_i　：　i 番目のタスク　$1 \leqq i \leqq N$　N はタスク数

$\boldsymbol{V}$　：　タスク v_i の集合

e_{ij}　：　タスク v_i からタスク v_j への通信

$\boldsymbol{E}$　：　タスク間の通信 e_{ij} の集合

$w(v_i)$：　タスク v_i の実行時間

$c(e_{ij})$：　タスク v_i からタスク v_j への通信時間

スケジューリングの基本単位であり，実行条件を持つタスク v_i の実行時間 $w(v_i)$を，次式で表現する．

$$w(v_i) = w_T(v_i)k_i + w_F(v_i)\overline{k_i}$$
$$k_i \in \{0, 1\}$$

ここで，$w_T(v_i)$，$w_F(v_i)$は，それぞれタスク v_i の実行条件が真，および偽である場合の実行時間を表す．k_i は，そのタスク v_i の実行条件の真偽を表す変数であり，1 が then 節の実行(タスク v_i の条件判定の結果が真)，0 が else 節の実行(タスク v_i の条件判定の結果が偽)を表す．

実行条件の真偽値 k_i の集合を，実行条件列 $\boldsymbol{k}$ として，次式で表す．

$$\boldsymbol{k} = (k_1, k_2, \cdots, k_N)$$

$\boldsymbol{k}$ の集合を $\boldsymbol{K}$，タスク間の実行条件の相関関係を考慮した $\boldsymbol{k}$ の集合を $\boldsymbol{K_P}$ で表す．

実行条件列 $\boldsymbol{k}$ の中には，排他的に実行される 2 つのタスクの実行条件のように，タスクが持つ実行条件の相関関係から，実際にはとり得ない実行条件の組合せが存在する．

$\boldsymbol{K_P}$ は，実行条件列として，とり得る $\boldsymbol{k}$ の集合であり，$\boldsymbol{K}$ の部分集合となる．集合 $\boldsymbol{K}$, $\boldsymbol{K_P}$ 及び実行条件列 $\boldsymbol{k}$ の関係を図 4 に示す．

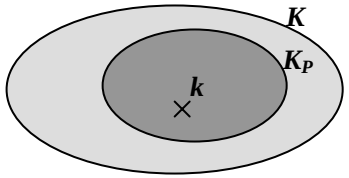

図 4 実行条件列 $\boldsymbol{k}$ と $\boldsymbol{K}$, $\boldsymbol{K_P}$

このとき，任意の実行条件列 $\boldsymbol{k}$ におけるシングルプロセッサ構成でのタスクの実行時間の総和 $T(\boldsymbol{k})$は，次式で表される．

$$T(\boldsymbol{k}) = \sum_{k_i \in \boldsymbol{k}} w(v_i) = \sum_{k_i \in \boldsymbol{k}} \left(w_T(v_i)k_i + w_F(v_i)\overline{k_i} \right)$$

3.2.2 並列化スケジューリング

複数のプロセッサ上で並列に実行するため，タスク v_i を実行するプロセッサを次のように表現する．

$$proc(v_i) \in \{PE_m\} \qquad m = 1,2,\cdots,M$$

ここで，PE はプロセッサであり，M はプロセッサ数である．また，$pred$, $succ$, t_s, t_f を以下のように定義する．

$pred(v_i)$　：　タスク v_i の先行タスクの集合

$succ(v_i)$　：　タスク v_i の後続タスクの集合

$t_s(v_i)$　：　タスク v_i の実行開始時刻

Vol.2009-SE-164 No.24
Vol.2009-EMB-13 No.24
Vol.2009-CSEC-45 No.24
2009/5/29

$t_f(v_i)$ ： タスク v_i の実行終了時刻

$$t_f(v_i) = t_s(v_i) + w(v_i)$$

最後のタスクが実行を終了する時刻 $max\ t_f(v_i)$を，タスク全体の実行時間（スケジュール長）と呼ぶ．

このとき，1 つの実行条件列 $\boldsymbol{k}$ についての WCET は，$\max_{\boldsymbol{k}\in \boldsymbol{K_P}}\{\max_{v_i\in V} t_f(v_i)\}$ で表現される．本研究では，全ての実行条件列に対する最短の WCET を求めるため，目的関数は次式となる．

$$minimize: \quad \max_{\boldsymbol{k}\in \boldsymbol{K_P}}\{\max_{v_i\in V} t_f(v_i)\}$$

このとき，以下の制約条件が存在する．

制約 1. 異なる PE への通信には，通信コストがかかる．

$\forall e_{ij}\in E$ について，$t_f(v_i) + c(e_{ij}) \leqq t_s(v_j)$

ここで，$proc(v_i) = proc(v_j)$ のとき $c(e_{ij})=0$ である．

つまり，同一 PE 上での実行では，通信コストがかからない．

制約 2. 1 つの PE 上では，同時に高々1 タスクのみ実行する．

$proc(v_i) = proc(v_j)$ のとき $\forall(v_i, v_j)$について，

$t_f(v_i) \leqq t_s(v_j)$ または $t_f(v_j) \leqq t_s(v_i)$ が成り立つ．すなわち，1 つの PE では，前のタスクが完了してから，次のタスクが実行可能になる．

4. 最適スケジュールの選定

並列化スケジューリングにおけるスケジュールとは，タスクを実行する PE と，その PE におけるタスクの実行順序を定めたものである．実行時に実行条件列 $\boldsymbol{k}$ は変化するが，スケジュール自体は変化しない．本章では，全ての $\boldsymbol{k}$ に対して WCET が最短となるスケジュールの選定について述べる．

4.1 総当り方式

本研究では，並列化スケジューリングに CP/DT/MISF 法[1]を用いている．CP/DT/MISF 法は，ヒューリスティックなアルゴリズムで，各タスクについて終端タスクまでの最長パス長を求め，この値を優先度とし，値が大きいものから順にリストスケジューリングを行う方法である．タスク間のデータ転送に要する時間を先行制約に含めてスケジューリングを行う．また，同じ優先度のタスクが複数存在する場合は，後続タスクの数が多いタスクを優先的に PE へ割り付ける．

ここで，実行条件列 $\boldsymbol{k}$ に対して得られるスケジュールを s_k，全条件 $\boldsymbol{K}$ に対するスケジュールの集合を $\boldsymbol{S}$ で表現する．

$$s_k \in \boldsymbol{S}$$

各実行条件列 $\boldsymbol{k}$ に対して得られるスケジュール s_k は，各 PE へのタスク割り当てと各 PE 上での実行順で示される．

総当り方式では，全ての実行条件列 $\boldsymbol{k}$ とそれに対応する全てのスケジュール s_k に対して，全ての実行条件の組合せ $\boldsymbol{K}$ を総当りで適用することで，WCET を求める．そして，$\boldsymbol{S}$ のうち全ての条件下において WCET が最短になるスケジュール s_k を，最適なスケジュールとして選択する．

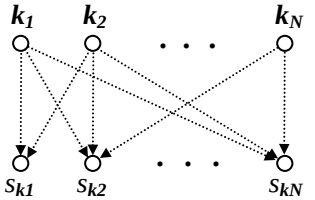

図 5 総当り方式

この方法では，タスク数を N とした場合，実行条件列 $\boldsymbol{k}$ の組合せの数は 2^N 通りとなる．さらに，それぞれの実行条件に対応した 2^N 通りのスケジュール s_k の各々に対して，2^N 通りの実行条件列を当てはめて WCET を探索するため，評価対象の数は 2^{2N} になる．このように，総当たり方式では，タスク数の増加に伴い計算量が指数関数的に増加するという問題がある．また，2 つのタスクの実行条件が排他的な関係にあるなど，タスク間の実行条件の関係によって，実際にはとり得ない実行条件の組合せも計算対象には含まれているものがある．

4.2 条件解析による対象の限定

タスク間の実行条件を解析することで，とりえない実行条件の組合せを計算対象から除き，計算量を抑制できる．タスクの実行条件 k_i は，実行条件に記述された真偽値の値をとる変数 $\boldsymbol{b} = \{b_1, b_2, \cdots\}$の論理式で構成される．以下に例を示す．

$k_1 = b_1 \ and \ b_2$

$k_2 = b_1 \ or \ b_3$

このような実行条件を持つタスク間で，実行条件の解析を次のように行う．図 6 に，

Vol.2009-SE-164 No.24
Vol.2009-EMB-13 No.24
Vol.2009-CSEC-45 No.24
2009/5/29

タスクが 2 つの場合の論理演算式と対応するベン図を示す．ここで，各タスクは，then 節と else 節を持つが，else 節に比べ必ず then 節の実行時間が長いものとする．図 6 の①は，タスク 1 とタスク 2 の両方が then 節を実行する実行条件列 $\boldsymbol{k}$ の集合を意味する．図 6 の①の領域に該当する実行条件列 $\boldsymbol{k}$ は複数存在し得るが，この領域内では実行条件列 $\boldsymbol{k}$によらず，各タスクの then 節を実行するため，実行時間は同一となる．②，③，④においても各領域内における各タスクの then 節/else 節の実行の組合せは同じであり，各々の領域における実行時間は同じである．CP/DT/MISF 法では，各タスクの実行時間を基にスケジューリングを行うため，各タスクの then 節/else 節の実行状態を示す図の各領域からそれぞれに対応した任意の実行条件列 $\boldsymbol{k}$ を 1 つ選べばよい．その $\boldsymbol{k}$ を求めるため，タスクの then 節/else 節実行の組み合わせを 2 分木で表現する．変数 $\boldsymbol{b}$ の論理式で表される各タスクの実行条件 k_i を BDD（Binary Decision Diagram）で表現し，各タスクの実行条件列 $\boldsymbol{k}$ に対して，充足可能性問題 SAT(Boolean Satisfiability Problem)を解く．

①$k_1 \cap k_2$
②$k_1 \cap \neg k_2$
③$\neg k_1 \cap k_2$
④$\neg k_1 \cap \neg k_2$

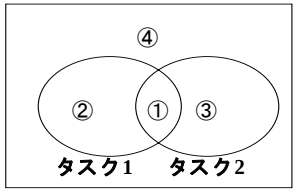

図 6　論理演算式とベン図

ある実行条件列 $\boldsymbol{k}$ の SAT が充足可能であれば，その実行条件列 $\boldsymbol{k}$ が組合せとして存在することを意味する．一方，SAT が充足可能でなければ，その組合せが存在しないことを意味する．

図 7 に SAT を用いた実行条件解析のアルゴリズムを示す．このアルゴリズムは，先頭のタスクから順に行う．先頭タスクからのタスク列の SAT が充足可能であれば，実行可能となる実行条件が存在する．そこで，それまでの実行条件をひとまとまりの実行条件として扱い，後続タスクの実行条件との間で論理演算を行い，2 分木の枝を拡張する．例として，図 6 のタスク 1，2 に続いてタスク 3 があり，タスク 1 とタスク 2 の両方を実行する図 6の①に関する2分木の枝を拡張する場合を考える．このとき，図 6①の条件$k_1 \cap k_2$が充足可能であれば，①の演算結果をひとまとまりとして扱い，タスク 3 との組合せ$(k_1 \cap k_2) \cap k_3$，$(k_1 \cap k_2) \cap (\neg k_3)$を 2 分木で表現する．

```
先頭タスクを取り出し基準タスクとする
while ( 全てのタスクを評価するまで ){
    基準タスクとその次のタスクの then 節/else 節実行の
                2 通りについて実行条件の組合せを評価する
    if (  SAT が充足可能  ) {
        評価に用いた 2 つのタスクを
            基準タスクとして 1 つのタスクにまとめる
    }else{
        後続のタスクとの実行条件の評価は行わない
    }
}
```

図 7　実行条件解析のアルゴリズム

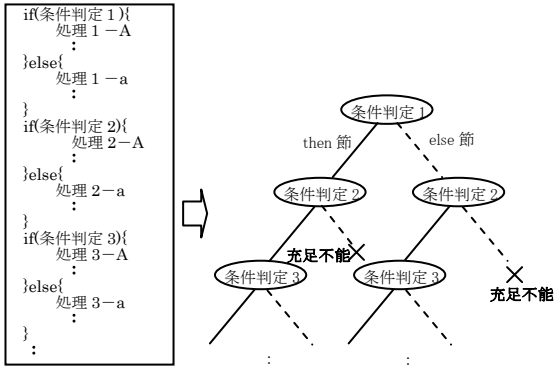

図 8　2 分木を用いた実行条件の組合せの限定

一方，先頭からのタスク列の SAT が充足不能であれば，タスクが実行可能となる実行条件が存在しないため，その実行条件に関する枝の作成は終了する(図 8)．例えば，条件判定 3=¬(条件判定 1)という関係があれば，条件判定 1 と条件判定 3 は同時に真にはならないため，それ以降はスケジューリングの対象から除く．

このように，実行条件を解析し，とりえない条件を除外することにより，スケジューリングの対象とする実行条件の組合せを限定することができる．全ての実行条件を解析した時点で，充足可能である実行条件の集合 $\boldsymbol{K_P} \subset \boldsymbol{K}$ を入力として，並列化スケジューリングを行う．充足可能な実行条件の組合せである $\boldsymbol{K_P}$ に含まれる各実行条件列 $\boldsymbol{k}$ を対象に並列化スケジューリングを行い，結果として得られるスケジュール $\boldsymbol{s_k}$ に対して，全ての条件の組合せ $\boldsymbol{K_P}$ を総当りで適用し，WCET が最短となる最適なスケジュールを求める．

Vol.2009-SE-164 No.24
Vol.2009-EMB-13 No.24
Vol.2009-CSEC-45 No.24
2009/5/29

なお，SAT の解析には zChaff [10]を用いている．

4.3 実行時間差による解析対象の限定

最適なスケジューリングを得るまでの計算量を更に抑えるため，各タスクの then 節，else 節の実行時間の差に注目し，この差が大きい上位指定個数のタスクのみを実行条件の解析対象として限定する．実行時間差が小さいものは，else 節に比べより長い時間である then 節の実行時間をそのタスクの実行時間として固定的に扱い，実行条件の解析対象から外す．実行時間の差が大きいタスクは，実行条件による全体のスケジュール長への影響が大きく，逆に実行時間の差が小さいタスクは，スケジュール長への影響が小さいためである．なお，解析対象とするタスク数は，任意の数を選択できる．

実行時間差に基づいて実行条件の解析対象タスク数を削減した後は，4.2 節の SAT 解析によって実行可能な $\boldsymbol{k}$ を求める．そして，4.2 節と同様に充足可能な実行条件の組合せを対象に総当りで並列化スケジューリングを行い，限定した範囲での最適なスケジュールを求める．

5. 結果及び考察

提案手法の有効性を確かめるため，PE 数が 2 の場合について，タスク数が 9～21 の 5 種類のサンプルプログラムを対象に，実行条件の組合せ数，スケジューリングに要する処理時間，WCET でのスケジュール長について評価を行った．

各タスクは実行条件を持ち，4.2 節に示した真偽値の値をとる変数 $\boldsymbol{b} = \{b_1, b_2, \cdots\}$ の論理式で構成される．また，then 節，else 節はそれぞれ 2 クロック～423 クロックの幅で実行時間を持つ．

評価環境は，CPU：Core2 1.86GHz，メモリ：2GB，OS：Windows XP Professional の計算機を用いた．また，並列化スケジューリング機能は，C++言語で作成し，Visual Studio 2005 を用い，リリース版でビルドしたものを実行した．

4.3 節に示した実行時間差による解析対象の限定を用いて，各タスクの実行時間差を求め，実行条件の解析対象を上位 6 タスクに限定した．各サンプルプログラムに含まれる実行条件の組合せと，並列化に要する時間を表 1 に示す．

表 1　組合せ数と並列化処理時間

プログラム	解析対象限定前		解析対象限定後		固定時間
	組合せ数 $\boldsymbol{K}$	処理時間(秒)	組合せ数 $\boldsymbol{K_P}$	処理時間(秒)	処理時間(秒)
sampleA	512	9.8	6	1.8	0.1
sampleB	131072	－	48	22.2	1.8
sampleC	262144	－	64	13.3	2.0
sampleD	1048576	－	24	16.4	0.9
sampleE	2097152	－	64	21.0	8.4

－：処理時間を計測できない項目

解析対象を限定する前は，2 のタスク数乗に相当する 512～2,097,152 通りの実行条件の組合せを持つが，4.3 節に示した方法により実行条件の解析対象を，実行時間差が大きい上位 6 タスクに限定することで，組合せ数を 64 通りに抑えている．これに対して，4.2 節に示した実行条件の解析を行うことで，sampleA は 6 通り，sampleB は 48 通り，sampleD は 24 通りまで組合せ数を抑えている．なお，sampleC, sampleE は，各タスクの実行条件に相関が無いため，条件解析を行っても組合せ数が削減できず，64 通りのままとなっている．このように，実行条件を解析しスケジューリング対象となる組合せを限定した後，これらの組合せについて，4.1 節に示した総当り方式により WCET が最短となるスケジュールを求める．この最適なスケジュールを得るまでの処理時間を組合せ数と共に表 1 に示している．表 1 に示す処理時間のように，解析対象を限定することで，実用的な時間で最適なスケジュールを得ることができる．なお，表中の“－”は，計算量が多大であるため，処理時間を計測できない項目である．

また，表 1 の固定時間の欄に示した処理時間は，実行条件を解析せず，全てのタスクについて then 節，else 節の実行時間のうち，長い方の then 節の実行時間をそのタスクの実行時間として固定的に扱ってスケジューリングを行い，WCET を求めるまでの時間である．各タスクの実行時間を固定しているため，スケジューリングのパターンは 1 通りしかない．ただし，この処理時間には，並列化スケジューリングに加え，得られたスケジューリング結果に対して，解析対象限定後のとりえる実行条件 $\boldsymbol{K_P}$ における各タスクの処理時間を，総当りで割当てた中で最長となるスケジュール長を選択するまでの時間を含む．この固定時間を用いる方法は，解析対象を限定する方法に比べ，全てのサンプルプログラムに対して処理時間が 1 桁短くなっている．

次に，表 1 に示したサンプルプログラムについて，最悪実行時のスケジュール長の短縮効果を表 2 に示す．

Vol.2009-SE-164 No.24
Vol.2009-EMB-13 No.24
Vol.2009-CSEC-45 No.24
2009/5/29

表 2　最悪実行時のスケジュール長

プログラム	固定時間(クロック)	解析後(クロック)	削減率(%)
sampleA	894	879	1.68
sampleB	889	461	48.1
sampleC	1292	1285	0.46
sampleD	2532	1722	32.0
sampleE	2549	2126	16.6

表 2 の固定時間でのスケジュール長は，先に述べたように，各タスクの実行時間を固定的に扱い，スケジューリングした結果に対して，とりえる実行条件 K_P での各タスクの実行時間を総当りで割当てた中で，最長となるスケジュール長を選択したものである．例えば上記 sampleA の場合，6 通りのタスクの実行時間を割当て，最長となるスケジュール長を選択している．一方，解析後のスケジュール長は，4.3 節で実行時間差を基に対象タスクを限定し 4.2 節の実行条件解析を行った結果，とりえる実行条件 K_P の中で 4.1 節の総当りにより求めた WCET である．

いずれのサンプルプログラムについても，実行条件を解析することにより，固定時間に比べ 1 桁程度処理時間が延びるものの，実用的な時間でスケジューリング結果が得られており，かつ，最悪実行時のスケジュール長を短縮できていることから，本手法の有効性を確認できる．

以上から，SAT を用いた各タスクの実行条件の解析と，各タスクの実行時間差に基づいた解析対象タスクの限定を組み合わせることで，計算量を抑えつつ，スケジュール長の短いスケジューリング結果が得られることが示された．

6. おわりに

実行条件を持つタスクの並列化スケジューリングに関し，最適なスケジュールを定義し，その求め方を提案した．最適なスケジューリングを総当りで探索する場合，タスク数の増加に伴い，指数関数的に計算量が増加する問題がある．そこで，実行条件を解析し，とりえない実行条件を計算対象から除くことで計算量を抑制した．さらに，タスクの実行時間に注目し，実行条件の解析対象を，実行条件による実行時間差が大きいタスクに限定することで，計算対象のタスク数を削減し，計算量を抑制した．本書ではタスク単位での実行時間の差に着目したが，タスクが実行条件として持つ変数とタスクの実行時間の差の関係を基にした計算量の削減が今後の課題である．

参考文献

[1]O. Sinnen, "*Task Scheduling for Parallel Systems*," Wiley-Interscience, 2007.
[2]笠原博徳, "並列処理技術," コロナ社, 1991.
[3]Y.N. Srikant and P. Shankar, "*The Compiler Design Handbook*," CRC Press, 2008.
[4]J.Engblom and A.Ermedahl, "Modeling Complex Flows for Worst-Case Execution Time Analysis," *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 2000.
[5]Y.S.Li and S.Malik, "Performance Analysis of Embedded Software using Implicit Path Enumeration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems(TCAD)* , Vol.16, No.12, 1997.
[6]S. Manolache, P. Eles, and Z. Peng, "The Worst-Case Execution-Time Problem Overview of Methods and Survey of Tools," *ACM Transactions on Embedded Computing Systems*, Vol.7, No.3, 2008.
[7] A. Colin and G. Bernat, "Scope-tree: a Program Representation for Symbolic Worst-Case Execution Time Analysis," *Proceedings of 14th Euromicro Conference on Real-Time Systems* (*ECTRS*02), 2002.
[8] S. Manolache, P. Eles, and Z. Peng, "Schedulability Analysis of Application with Stochastic Task Execution Times," *ACM Transactions on Embedded Computing Systems*, Vol.3, No.4, 2004.
[9] Y. S. Dandass, "Genetic List Scheduling for Soft Real-Time Parallel Applications," *Evolutionary Computation(CEC)*. , Vol.1, 2004.
[10] M. Moskewicz, C. Madigan, Y. Zhao, and L. Zhang, S. Malik, "Chaff:Engineering an Efficient SAT Solver, ", *38th Design Automation Conference (DAC)*, 2001.