アルゴリズム 71－2
（2000．1．17）

タスク複製率とプロセッサアイドル率に着目したBSPスケジュール生成手法の提案

森 雅博，橋本貴至，西村晃一，藤本典幸，萩原兼一
大阪大学大学院基礎工学研究科 情報数理系専攻
{masahiro, t-hasmot, k-nisimr, fujimoto, hagihara}@ics.es.osaka-u.ac.jp

計算はスーパーステップでのみ行い，通信はスーパーステップ間でしか行わない並列計算モデルBSP(Bulk Synchronous Parallel)に則ったスケジュールを，BSPスケジュールと呼ぶ．BSPスケジュールから生成した並列プログラムには，通信一括化を適用した効率的な集合通信を行えるという利点がある．BSPスケジュールの性能を決める特性量として，プロセッサアイドル時間および通信量，タスク複製量が考えられる．しかし，性能とこれらの特性量との関係は，今まで良く知られていない．本稿では，特性量と性能との関係を調査するため，パラメータを与えることで特性量を指定できるBSPスケジュール生成アルゴリズムASSERT(Algorithm for Synchronous Schedule which Engages for Ratios-Temperance)を提案する．ASSERTに与えるパラメータを変化させてスケジュールを生成し，スケジュールから生成した並列プログラムの性能が最も良いスケジュールを分析した．この分析結果と，特性量を指定できない既存のBSPスケジュール生成アルゴリズムBCSHのスケジュールの分析結果を比較した．その結果，通信量と複製量のトレードオフの実行性能への影響など，性能と特性量との関係を確認できた．また，並列化の効果を表すスピードアップ率は，最大でBCSHに比べて37.5%上昇したことから，ASSERTの有効性もわかった．

On Generating a Bulk Synchronous Schedule Controled by a Task-Duplication Ratio and a Processor-Idle Ratio

Masahiro Mori, Takashi Hashimoto, Kouichi Nishimura,
Noriyuki Fujimoto, Kenichi Hagihara
Department of Informatics and Mathematical Science,
Graduate School of Engineering Science, Osaka University

A bulk synchronous schedule is a schedule based on BSP(Bulk Synchronous Parallel) computation model. In a parallel program generated from this schedule, communication can be implemented efficiently as collective communication of packaged messages. There seem to be three properties which influence the performance of the generated program, i.e. the number of communications, the number of duplicated tasks, and the length of idle-time. However there was no knowledge about relations between these properties and the performance. In this paper, in order to investigate the relations, we propose a scheduling algorithm ASSERT(Algorithm for Synchronous Schedule which Engages for Ratios-Temperance) which generates a bulk synchronous schedule according to user specified parameters which control the properties in the generated schedule. For various parameters, we generate schedules with ASSERT, and analyze schedules from which highest-performance parallel programs are generated. Then, we compare the result of analysis with one for the existing algorithm BCSH which generates a bulk synchronous schedule without user specified parameters. As a result of this comparison, we found some relations between properties and the performance. Further, a speedup ratio, a measure of the effect of parallelism, is improved 37.5% by ASSERT toward BCSH.

1 はじめに

Valiantにより提案された並列計算モデルである**BSP(Bulk Synchronous Parallel)**[9]に則ったスケジュールを**BSPスケジュール**[6]という．BSPスケジュールには，BSPモデルのスーパーステップに相当する計算層があり，BSPスケジュールに基づく並列プログラムでは，計算層の間でしか通信を行わないようにできる．このため，BSPスケジュールと等価な並列プログラムを生成する際には，送受信するプロセッサが同じであれば，まとめて送受信する

通信一括化[7]を適用した上で，集合通信命令を用いることで，効率的に通信できる[5][6]．また，このようにプログラム化することで計算層にかかる時間をある程度予測できるため，さらに集合通信にかかる時間を見積もれば，全体の実行時間もおおよそ予測でき，かつ予測性能と並列プログラム実行性能との乖離が少ないという特長がある[5][6]．

実行性能の良い並列プログラムを生成しうるBSPスケジュールが，満たしていると考えられる特性が3つある．P1) 各計算層における各プロセッサの負荷が均等であり，プロセッサアイドル時間が少ないこと．P2) 計算層が厚いこと．P3) 複製タスクが多すぎないことである．P2とP3はトレードオフの関係にある．ところが，スケジュールのこれらの特性量と，スケジュールから生成した並列プログラムの実行性能との関係は，今まで良く知られていない．

本稿では，タスク複製率，プロセッサアイドル率の上限値，計算層の厚さの下限値をパラメータで与えることで，各特性量を指定できるBSPスケジュール生成アルゴリズム **ASSERT(Algorithm for Synchronous Schedule which Engages for Ratios-Temperance)** を提案する．また，ASSERTを実装してスケジュールを生成する．さらに，C言語と通信ライブラリMPI[4]を用いた生成スケジュールと等価な並列プログラムの実行性能を，分散メモリ型並列計算機Cenju-3[2]を用いて評価する．生成するスケジュールの特性量を指定できない既存のBSPスケジュール生成アルゴリズム **BCSH(Bulk Communication Scheduling Heuristic)** のスケジュールと，ASSERTのスケジュールとの比較を，並列プログラムの実行性能に対応付けて行うと，プロセッサアイドル時間，タスク複製量を相対的に多くする事で，計算層が厚くなり，実行性能が向上することがわかった．また，計算層の厚さが同程度の場合には，タスク複製量の少ない方が実行性能が良く，逆にタスク複製量が多くなりすぎると，計算層が厚くても実行性能が悪くなることがわかった．また，並列化の効果を表すスピードアップ率は，最大でBCSHに比べて37.5%上昇し，ASSERTの有効性も確認できた．

2 諸定義

2.1 タスクグラフとスケジュール

重み付きの **DAG(Directed Acyclic Graph)** を $G=(V,E,\lambda,\tau)$ と記す．ここで V は節点の集合，E は有向辺の集合，λ は節点から節点の重みへの関数，

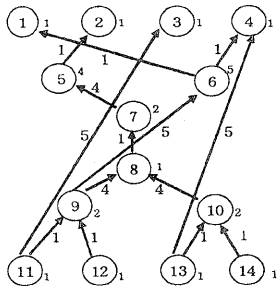

図 1: タスクグラフの例

τ は有向辺から有向辺の重みへの関数である．節点 u から節点 v への有向辺を (u,v) と記す．並列計算をモデル化したDAGをタスクグラフと呼ぶ．図1にタスクグラフの例を示す．タスクグラフの節点は並列計算のタスクを表す．節点 u が表すタスクを T_u と記す．値 $\lambda(u)$ は，T_u の計算時間が $\lambda(u)$ 単位時間であることを意味する．有向辺 (u,v) は，T_v の計算のためには T_u の計算結果が必要であることを意味する．値 $\tau(u,v)$ は，T_u を計算するプロセッサ p から T_v を計算するプロセッサ q へのプロセッサ間通信遅延は，p と q が異なる場合高々 $\tau(u,v)$ 単位時間であることを意味する．p と q が同一なら通信遅延は0とする．

$(u,v)\in E$ のとき，v は u の**直接後続節点**と呼ぶ．v のすべての直接後続節点の集合を $Succ(v)$ と記す．直接後続節点のない節点を**出力節点**と呼ぶ．**パスの長さ**を，そのパス上の節点の重みの和と定義する(有向辺の重みは加えない)．節点 u の**レベル** $level(u)$ は，u から出力節点への最長パスの長さである．レベル l_1 (l_1 から l_2) のすべての節点の集合を $V(l_1)$ ($V(l_1..l_2)$) と記す．G の節点のレベルの最大値を $l_{max}(G)$ と記す．G の有向辺の重みの最大値を $\tau_{max}(G)$ と記す．

タスクグラフ $G=(V,E,\tau,\lambda)$ の使用プロセッサ数 p のときの**スケジュール**とは，以下の条件R1，R2，R3を満たす3つ組 $\langle v,q,t\rangle$ の有限集合をいう[8]．ここで，$v\in V$，q は $1\le q\le p$ を満たす整数，t は非負整数である．3つ組 $\langle v,q,t\rangle\in S$ は，プロセッサ q がタスク T_v を時刻 t から時刻 $t+\lambda(v)$ の間に計算することを意味する．

- R1 各 $v\in V$ に対して，3つ組 $\langle v,q,t\rangle$ が少なくとも1つ S に存在する．
- R2 $v\neq v'$ かつ $\max(t,t')<\min(t+\lambda(v),t'+\lambda(v'))$ となる2つの3つ組 $\langle v,q,t\rangle,\langle v',q,t'\rangle$ は S に存在しない．
- R3 $(u,v)\in E$ かつ $\langle v,q,t\rangle\in S$ ならば $t'\le t-\lambda(u)$ となる3つ組 $\langle u,q,t'\rangle$ が S に存在するか，ま

たは $t'' \leq t-\lambda(u)-\tau(u,v)$ かつ $q \neq q'$ となる3つ組 $\langle u,q',t''\rangle$ が S に存在する．

スケジュール S の**メイクスパン**は，S のすべてのタスクの計算が完了する時刻 $\max\{t+\lambda(v)|\langle v,q,t\rangle \in S\}$ である．

2.2 BSP スケジュール

本節ではBSPスケジュールの定義[6]について述べる．タスクグラフ $G=(V,E,\lambda,\tau)$ のプロセッサ数 p のときのスケジュールを S とする．以下の条件C1，C2を満たすとき，またそのときに限り，$S'(\subseteq S)$ は S の**計算層**であるという．

C1 $q \neq q'$ かつ $(u,v) \in E$ となる3つ組 $\langle u,q,t\rangle, \langle v,q',t'\rangle$ が S' に存在するならば，$t'' \leq t'-\lambda(u)$ となる3つ組 $\langle u,q',t''\rangle$ が S' に存在する．

C2 $t_{min}(S') \leq t$ かつ $t+\lambda(v) \leq t_{max}(S')$ となる3つ組 $\langle v,q,t\rangle$ が S に存在するならば，3つ組 $\langle v,q,t\rangle$ が S' に存在する．ここで，$t_{min}(S')=\min\{t|\langle v,q,t\rangle \in S'\}$, $t_{max}(S')=\max\{t+\lambda(v)|\langle v,q,t\rangle \in S'\}$ ．

つまり，S' 中のあるタスクの計算結果が必要となる場合，そのタスクが他プロセッサで実行されていても，**複製タスク**[3]として各プロセッサ自身でも計算しているので，S' において通信は必要ない．また，S' はSの時間による分割の1要素である．ここで，$t_{max}(S_i)-t_{min}(S_i)$ を S_i の**局所メイクスパン**という．S が以下の条件C3を満たす計算層系列 $\langle S_1,S_2,\ldots,S_n\rangle$ へと分割できるとき，またそのときに限り，S は**BSPスケジュール**であるという．

C3 任意の $i(1 \leq i < n)$ について $t_{max}(S_i)+\tau_{suff}(S_i,S_{i+1}) \leq t_{min}(S_{i+1})$ ，ここで，$\tau_{suff}(S_1,S_2)=\max\{\tau(u,v)|u \in S_1, v \in S_2\}$．ただし任意の $u \in S_1, v \in S_2$ について，$(u,v) \notin E$ のときは，便宜上 $\tau_{suff}(S_1,S_2)=\infty$ とみなす．

t_{suff} は連続する計算層間の通信遅延時間を十分に見積もったものである．

2.3 BSP スケジュール構成法

この節ではBSPスケジュールの一構成法[5][6]について述べる．

2.3.1 独立グループ集合

タスクグラフ $G=(V,E,\lambda,\tau)$ の任意の連結成分を $G'=(V',E',\lambda,\tau)$ とする．V' を G の**節点グループ**と呼ぶ．C_j と C_k をそれぞれ G の節点グループとする $(C_j \neq C_k)$．$v \in C_j$ かつ $v \in C_k$ であるような節点

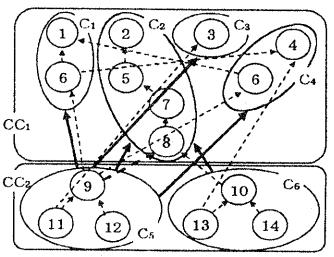

図 2: 独立グループ集合の例

v を C_j と C_k の**共通節点**と呼ぶ．複数の節点グループに属する節点は複製タスクに相当する．C_j, C_k の共通節点の重みの総和 $\sum_{v \in \{C_j \cap C_k\}} \lambda(v)$ を C_j と C_k の**親和度**と呼ぶ．$v \in C_j$ かつ $u \in C_k$ かつ $u \notin C_j$ である (u,v) が E に存在するとき，かつそのときに限り C_j は C_k に**直接依存する**という．以下の条件E1またはE2が満たされるとき，C_j は C_k に**依存する**という．

E1 C_j は C_k に直接依存する．

E2 C_j が C_i に直接依存し，かつ C_i が C_k に依存するような G の節点グループ C_i が存在する．

互いに依存のない節点グループの集合を**独立グループ集合**と呼ぶ．図1のタスクグラフに対する独立グループ集合の例を図2に示す．

2.3.2 独立グループ集合を用いたBSPスケジュールの構成法

以下の方法でBSPスケジュールを生成できる．

1. タスクグラフ G を節点のレベルで部分グラフに分割し，(必要ならば複製タスクを用いて) 各部分グラフから独立グループ集合を構成する．例えば図2は，図1のタスクグラフをレベル4と5の間で分割し，タスク T_6 を複製して構成した2つの独立グループ集合を表している．
2. 各独立グループ集合が1つの計算層に対応するように，節点グループ単位で節点をプロセッサに割り当ててBSPスケジュールを構成する．例えば図3は，図2の独立グループ集合に基づいて構成されたBSPスケジュールを表している．

この方法でBSPスケジュールを生成するためには，タスクグラフをどのレベルで分割するか，独立グループ集合をどのように構成するかを決めれば良い．

任意のタスクのレベルを l とする．同レベルのタスク間には依存が無いので，$V(l)$ から $\{\{v\}|v \in V(l)\}$ として独立グループ集合を作ることができる．k を非負整数として，すでに $V(l..l+k)$ から独立グループ集

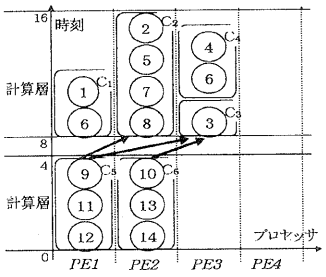

図 3: BSP スケジュールの例

合が構成されているものとする．u を $V(l+k+1)$ の任意の節点とすると，$\{C \cup \{u\} | C \in CC, C \cap Succ(u) \neq 0\} \cup \{C | C \in CC, C \cap Succ(u) = 0\}$ のように，節点 u を直接後続節点として持つグループすべてに，u を複製して収容することにより，$V(l..l+k) \cup \{u\}$ から独立グループ集合を構成できる．$V(l..l+k+1)$ の全節点に対してこの処理を繰り返せば (**全複製**)，$V(l..l+k+1)$ から独立グループ集合を構成できる．全複製の際には，必要以上に節点の複製が行われる可能性がある．そこで，グループをマージして複製の削減を図る．ただし，マージの際には**マージ判定**を行い，判定条件を満たさないマージは行わない．

出力節点から順に，1レベルずつタスク複製とマージを繰り返し，節点グループを構成して行く．もし，$V(l..l+k+1)$ の独立グループ集合が計算層採用判定条件を満たすなら，レベル $(l+k+1)$ が $l_{max}(G)$ になるまで進み，独立グループ集合を計算層として処理を終了する．もし，計算層採用判定条件を満たさなくなったら，$V(l..l+k)$ からの独立グループ集合を計算層とし，レベル $l+k+1$ から新たな独立グループ集合を構成する．

3 BSP スケジュールの評価指標

3.1 BSP スケジュールが満たすべき特性

BSP スケジュールには，通信の一括化を適用した1回の集合通信命令により，計算層間の通信を効率的に行う並列プログラムを生成できるという利点がある[7]．一般に複数回の一対一命令によるより，1回の集合通信命令による方が，多対多の通信を効率的に行える．しかし，集合通信には弊害がある．計算層における各プロセッサの負荷バランスが良ければ，効率的に通信を行えるが (図4·左部)，負荷バランスが悪い場合，最も負荷の高いプロセッサの計算が終了するまで通信命令が発行されない，あるいは最も負荷の高いプロセッサからの計算結果を受け取るまで

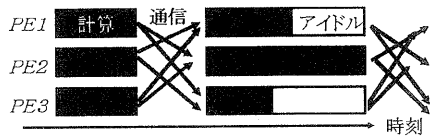

図 4: 集合通信の弊害によるアイドル

は，次の計算が開始できないなどの理由により，早く計算終了したプロセッサにアイドルが発生してしまう (図4·右部)．よって，BSP スケジュールは次の特性を満たすべきである．

P1 計算層で各プロセッサの負荷が均等であること．

BSP スケジュールにおいては，P1 を満たす限り効率よく通信を行うことができる．しかし，通信には細粒度タスク1個の計算時間の 15 から 86 倍のオーバヘッドをともないコストが高い[5]ため，通信回数が多すぎると実行性能が低下してしまう．同じタスクグラフの BSP スケジュールであれば，局所メイクスパンが長いほど計算層数が少ないので，通信回数も少ない．ところで，局所メイクスパンの長い BSP スケジュールほど，2.3.2節の方法のように，一般にタスクの複製が必要となる．しかし，複製タスクが多すぎると，同じタスクを何度も計算することとなり，計算効率が悪くなる．つまり，通信回数と複製タスク数の間にはトレードオフの関係がある．以上のことを踏まえると，BSP スケジュールがさらに満たすべき特性は以下の2点である．

P2 各局所メイクスパンが長いこと．

P3 複製タスクが多すぎないこと．

前述の通り，R2 と R3 はトレードオフの関係にある．

3.2 特性充足度の測定指標

前節の各特性をどの程度満たしているかを測るための指標として，プロセッサアイドル率，プロセッサ使用率，タスク複製率を定義する．それぞれ，P1，P2，P3 に対応している．タスクグラフ $G = (V, E, \lambda, \tau)$ のプロセッサ数 p のときの BSP スケジュールを $S = \langle S_1, S_2, \ldots, S_n \rangle$ とする．S_i の局所メイクスパンを $MaxLM(S_i)$ と記す．また，$\max\{t+\lambda(v) | \langle v, q, t \rangle \in S_i\} - \min\{t | \langle v, q, t \rangle \in S_i\}$ を $LM(S_i, q)$ と記す．

・ プロセッサアイドル率

$$IR(S_i) = \frac{\sum_{\{q | LM(S_i,q) \neq 0\}} (MaxLM(S_i) - LM(S_i,q))}{MaxLM(S_i) \cdot |\{q | LM(S_i,q) \neq 0\}|}$$

・ プロセッサ使用率

$$UR(S) = \frac{\sum_{i=1}^{n} MaxLM(S_i)}{\sum_{i=1}^{n} MaxLM(S_i) + \sum_{i=1}^{n-1} \tau_{suff(S_i, S_{i+1})}}$$

・ タスク複製率

$$DR(S_i)=\frac{\sum_{\langle v,q,t\rangle\in S_i}\lambda(v)}{\sum_{v\in V}\lambda(v)}$$

プロセッサアイドル率は，計算層に占めるアイドルの比率であり，計算層におけるプロセッサ間の負荷が均等であるほど低くなる．$\{q|LM(S_i,q)\neq 0\}$により，全くタスクを割り当てられていないプロセッサを対象外としている．プロセッサ使用率は，スケジュール全体に占める計算層の比率であり，各計算層の局所メイクスパンが長く，通信回数が少なく，計算層間の通信遅延が小さいスケジュール程高い．タスク複製率は，計算層中の全タスクの計算時間の和が，計算層中の複製タスクを各1つのみ残して除去した全タスクの計算時間の和の何倍かを表す．複製タスクが少ない程この値は低くなる．

$IR(S_i), UR(S_i), DR(S_i)$の理想値は$0,1,1$である．しかし，特殊なタスクグラフでなければ，複製は不可欠であり，$DR(S_i)=1$は達成できない．また，現実的には通信遅延は0ではないので，計算層数1のスケジュールでなければ，$UR(S_i)=1$は達成できない．計算層数1は複製を多用すれば実現できるが，$DR(S_i)$が上昇してしまう．つまり，$UR(S_i)$と$DR(S_i)$の是非は両者の兼ね合いで決まる．$IR(S_i)$については，$UR(S_i)$を1に近づける際に犠牲になり得る．以上より，$IR(S_i), UR(S_i), DR(S_i)$は，総合的に評価されるべきである．また，各々がどのような関係にあれば良いのかは，少なくとも現時点では不明である．

4 アルゴリズム ASSERT

この章では2.3節に述べた構成法によるスケジューリングアルゴリズム ASSERT を記す．ただし本稿においては，アルゴリズムのキーとなる独立グループ集合からの計算層形成およびマージ判定条件，計算層採用判定条件のみを記す．アルゴリズムの詳細については文献[6]を参照されたい．ASSERTへの入力はタスクグラフ，プロセッサ数，集合通信にかかる予測時間であるパラメータτおよび，3つのパラメータ，許容プロセッサアイドル率δ，最小局所メイクスパンζ，制限タスク複製率ρである．また，出力はスケジュールである．ASSERTでは，タスクグラフの分割レベルと複製タスクをヒューリスティックに決定する際に，3.1節に記した特性を満たすため，δ,ζ,ρを用いて，出力するスケジュールのプロセッサアイドル率，プロセッサ使用率，タスク複製率を制御する．

```
条件判定: Balance
入力: 独立グループ集合 CC
出力: CC のバランスが良いとき真，そうでないとき偽
  return ((IR(CC) ≤ δ)
                or (MaxLM(CC,p) ≤ ζ));
```

図 5: バランス判定条件

```
条件判定: MergeSize
入力:1. 節点グループ C1, C2
     2. 独立グループ集合 CC
出力: C1 と C2 をマージするとき真，しないとき偽
  {M ∈ CC} を ω(M) の降順にソートし，
  ⟨M1, M2, ..., Mk⟩ とする;
  i を C1 ≥ Mi を満たす最大のものとする;
  j を C2 ≥ Mj を満たす最大のものとする;
  return (((ω(C1) < ω(M_P(CC)))
                or (i > P(CC)))
          and ((ω(C2) < ω(M_P(CC)))
                or (j > P(CC))));
```

図 6: マージサイズ判定条件

4.1 独立グループ集合からの計算層形成

節点グループをC，独立グループ集合をCCとし，$\omega(C)=\sum_{v\in C}\lambda(v)$，$W(CC)=\sum_{v\in(\cup_{C\in CC}C)}\lambda(v)$とする．計算層$S_i$は，$P(CC)(\leq p)$プロセッサに，全ての$C\in CC$を$\omega(C)$の降順で，もっとも負荷の軽いプロセッサに割り当るGrahamのアルゴリズムLDSH[1]を用いて形成する．ここで，$P(CC)$はプロセッサ節約[5][6]の考えに基づいて，$MaxLM(CC,P(CC))=\min\{MaxLM(CC,x)|lb\leq x\leq p,\ lb=\min(p,\lceil W(CC)/\max\{\omega(C)|C\in CC\}\rceil)\}$を満たす最小の正整数とする．このときの$S_i$の局所メイクスパンを$MaxLM(CC,p)$と記す．

4.2 マージ判定条件

マージ判定条件は，バランス判定条件とマージサイズ判定条件との論理積である．図5のバランス判定条件が真のとき，独立グループ集合CCはバランスが良いという．ここで，$IR(CC)$は，4.1節のLDSHによって，CCを計算層にしたときのプロセッサアイドル率である．プロセッサアイドル率による判定に加え，第2の判定によりプロセッサ使用率による判定を行う．ASSERTでは，タスクグラフ中の有向辺の重みは全て$\tau_{max}(G)$とみなす[6]ので，プロセッサ使用率は，計算層数と各計算層の局所メイクスパンで決まる．計算層数はスケジューリングの過程ではわからないので，局所メイクスパンのみでプロセッサ使用率の下限制御を試みる．マージ判定条件内部にバランス判定条件を含むことで，マージによってバランスが悪くなるのを防ぐ．次に，マージサイズ判定条件を図6に示す．$\omega(C)$の大きな節点グループ

条件判定: **Acceptable**
入力: 独立グループ集合 CC
出力: CC を計算層として採用するとき真，しないとき偽
$bl := min\{level(v)|v \in C, C \in CC\}$;
$tl := max\{level(v)|v \in C, C \in CC\}$;
$notmuchdup := (W(CC)/\omega(V(bl..tl)) \leq \rho)$;
$balanced := Balance(CC)$;
return *notmuchdup* **and** *balanced*;

図 7: 計算層採用判定条件

表 1: $\tau_{max}(G)$ と ASSERT におけるパラメータ

問題	p	$\tau_{max}(G)$	δ	ζ	ρ
LU128	4	771	0.2	1300	1.4
	8	1440	0.3	1800	1.4
	16	2760	0.3	2800	1.4
Jacobi512	4	640	0.1	500	1.4
	8	1200	0.1	1000	1.4
	16	2310	0.1	2000	1.4
FFT16k	4	839	0.1	500	1.4
	8	1570	0.1	1800	1.4
	16	3000	0.1	2000	1.1

は，さらに突出して大きくなりやすく，バランスが悪くなる原因となるため，プロセッサ台数個までの節点グループをなるべく均等に成長させる．

4.3 計算層採用判定条件

最後に計算層採用判定条件を図 7に示す．ここで，$W(CC)/\omega(V(bl..l))$ が複製率にあたる．ASSERT では，マージした結果，より多くの複製を削減できるように，マージは親和度の高いクラスタ対から順に適用する．そこで，計算層採用判定条件では，一連のマージ処理によって，制限複製率を下回るまで複製タスクが削減されたかを判定する．マージが一度も起こらなくても複製率が低い場合に，バランスの悪い計算層を採用しないため，ここでもバランス判定を行っている．

5 評価実験

5.1 タスクグラフ生成

逐次プログラムからタスクグラフを生成する．全てのタスクの実行時間を 1 単位時間とし，有向辺の重みは表1とする[5][6]．$\tau_{max}(G)$ は ASSERT，BCSH で共通とする．

5.2 並列プログラム生成

ASSERT，BCSH を用いてプロセッサ数に応じたスケジュールを，タスクグラフから生成する．次に生成スケジュールから，それと等価な C 言語と MPI による並列プログラムを生成する．通信には通信の一括化を適用した上で，全対全通信命令 (MPI_Alltoallv()) を用いる．

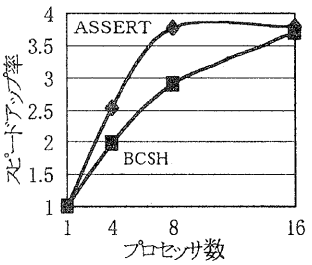

図 8: LU128 のスピードアップ率

5.3 評価指標

G をタスクグラフとし，$T_p(G)$ をプロセッサ数 p としたスケジュールから生成されたプログラムの実行時間とする．このとき，**スピードアップ率**は $Sp(p) = T_1(G)/T_p(G)$ と表される．$Sp(p)$ は分散メモリ型並列計算機において，プロセッサ数 p であるときのスケジュールの性能評価指標となる．

5.4 MPI プログラムの性能比較

サイズ 128 の LU 分解とサイズ 16384 の FFT，サイズ 512 の Jacobi 解法 (ループのボディのみ) を用いて，ASSERT と BCSH によるスケジュールから生成した，MPI プログラムの実行性能の比較実験を行った．ASSERT においては，与えるパラメータを基礎性能測定に基づいて変動させ，ピーク性能を得たスケジュールを使用した (表1参照)．実行環境は，分散メモリ型並列計算機 Cenju-3 を用いた．Cenju-3 は 150MIPS，50MFLOPS の VR4400 プロセッサ最大 256 台の構成が可能である．Cenju-3 では，ローカルメモリは 64MB，ネットワークトポロジは多段接続であり，帯域は 40MB/sec である．

5.4.1 LU128

ASSERT はプロセッサ数 4 台，8 台，16 台において，BCSH のスピードアップ率を上回る (図8参照)．特に性能向上が著しいプロセッサ数 8 台における両手法のスケジュールを解析すると，各計算層の複製率，アイドル率の総和を計算層数で割った値である平均 DR，平均 IR は，ASSERT の方が BCSH より高くなっている事がわかる (表2参照．層数は計算層数)．しかし局所メイクスパンは大きくなり層数が減っている (図9参照)．これによりプロセッサ使用率 UR は，BCSH で 0.77 なのに対し，ASSERT では 0.91 と高くなっている．視覚化した実行状況を図10に示す．各段が各プロセッサに対応し，横軸は時刻である．両手法とも時刻軸の縮尺は同じである．ASSERT では高い UR の実現により，実行時間を短縮できている．

表 2: LU128 の平均複製率と平均アイドル率

手法	p	層数	平均 DR	UR	平均 IR
ASSERT	8	11	1.41	0.91	0.07
BCSH	8	22	1.14	0.77	0.03

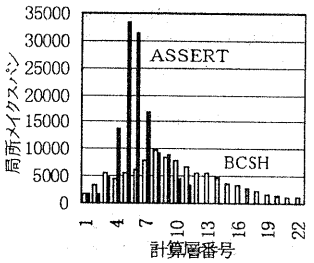

図 9: LU128 の計算層の厚さ

5.4.2 Jacobi512

ASSERT, BCSH ともプロセッサ数によらず層数は 2 となっていて, UR も両手法とも同程度であるので, 平均 DR を低く抑えている ASSERT の方が計算効率が良く, BCSH を若干上回るスピードアップを得た (表3, 図11参照).

5.4.3 FFT16k

ASSERT と BCSH の差異はほとんど見られなかった (図12参照). これは, サイズ 16k の FFT のタスクグラフの, 並列度の高さに起因する (図13参照). 他プロセッサと通信が起きないように, バタフライ演算に関連するタスクを, まとめて各プロセッサに割り当てることで, あまり複製することなく, かつバランスの良い計算層形成が可能であり, ある程度 DR, IR の低いスケジュールしか生成できない BCSH においても, 高い性能が得られている. ところで, 使用プロセッサ台数が 4 台の時には, ASSERT が BCSH を上回っている. これは, ASSERT において複製により計算層数 1(UR=1) の無通信を実現しているためである (図14, 表4参照). しかし, 通信をなくすためとはいえ, 複製を行いすぎると逆に実行効率が低下し, スピードアップ率 Sp が低下してしまう. (図15, 表5参照). どちらも FFT16k のタスクグラフのプロセッサ数 16 のときの ASSERT によるスケジュールだが, 一方は複製率制限をしていない. 複製率制限をしない場合には, 層が成長して 1 層になり UR は 1 となるが, DR が 2.14 となる (表5参照).

6 まとめと今後の課題

プロセッサアイドル率, プロセッサ使用率, タスク複製率を指定可能な BSP スケジュール生成アルゴリズム ASSERT を提案し, 実装した. ASSERT によりスケジュールを生成し, 生成スケジュールと等価な並列プログラムを生成実行した. 既存手法との比較の結果, 複製率, アイドル率を相対的に高くする事で, 局所メイクスパンの大きい計算層が形成され, プロセッサ使用率が上昇し, 実行性能が向上することがわかった. また, プロセッサ使用率が同程度の場合には, 複製率の低い方が実行性能が良く, 逆にプロセッサ使用率が高くても, 複製率が高くなりすぎると実行性能が悪くなることがわかった. 総じて実行性能は LU128, Jacobi512 においては既存手法を上回り, FFT16k においても既存手法と同程度か上回ったことから, ASSERT の有効性も確認できた.

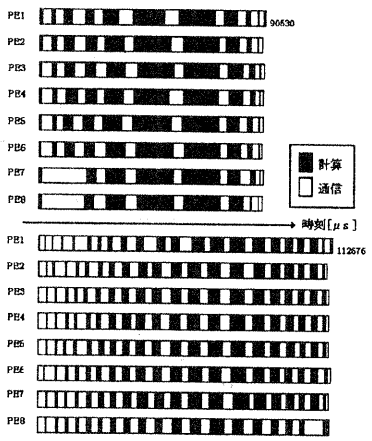

図 10: LU128 の実行状況 (上:ASSERT, 下:BCSH)

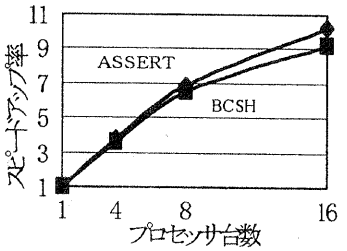

図 11: Jacobi512 のスピードアップ率

今後の課題は, 現在ユーザが与えているアイドル率, プロセッサ使用率, 複製率に関するパラメータを, 並列計算機の通信特性や, タスクグラフの特性から求める手法を確立することと, マージサイズ判定のヒューリステッィックの改良である. マージサイズ判定における改良点としては, 成長させる節点グループの個数の変更や, 統計的手法の導入などが考えられる.

表 3: Jacobi512 の平均複製率と平均アイドル率

手法	p	層数	平均 DR	UR	平均 IR
ASSERT	4	2	1	0.99	0.00
	8	2	1	0.97	0.12
	16	2	1.00	0.89	0.06
BCSH	4	2	1.01	0.99	0.04
	8	2	1.01	0.96	0.04
	16	2	1.02	0.89	0.28

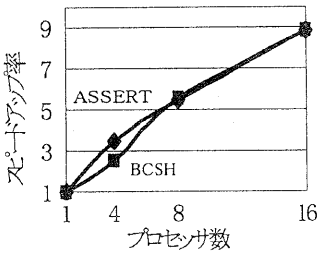

図 12: FFT16k のスピードアップ率

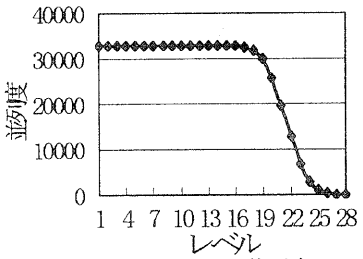

図 13: FFT16k の並列度

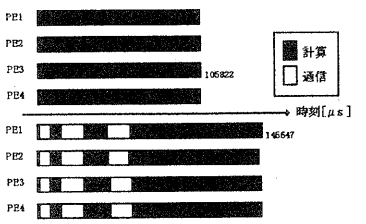

図 14: FFT16k の実行状況 1 (上 : ASSERT, 下 : BCSH)

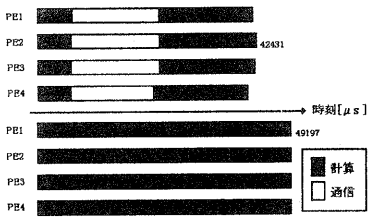

図 15: FFT16k の実行状況 2 (上 : ASSERT, 下 : ASSERT(複製率制限無し))

表 4: FFT16k において複製をした方が良い例 (表上)
表 5: FFT16k において複製が多すぎる例 (表下)

手法	p	層数	平均 DR	UR	平均 IR	Sp
ASSERT	4	1	1.14	1	0	3.52
BCSH	4	4	1.06	0.99	0.00	2.56

手法	p	層数	平均 DR	UR	平均 IR	Sp
ASSERT	16	2	1.05	0.94	0	8.89
ASSERT(※)	16	1	2.14	1	0	7.66

※複製率制限なし

謝辞

本研究は一部平成 11～12 年度文部省科学研究費補助金 · 基盤研究 (C) (11680357) およびＰＤＣ (並列 · 分散処理研究推進機構) の補助による．並列計算機 Cenju-3 を使用させて頂いた NEC(株) に感謝します．

参考文献

[1] Graham, R.L. :"Bounds on multiprocessing timing anomalies", SIAM Journal of Applied Mathematics,Vol.17, pp.416-429, 1969.

[2] 広瀬 哲生, 細見 岳生, 丸山 勉, 加納 健: "並列コンピュータ Cenju-3 のプロセッサ間通信方式とその評価", 情報処理学会論文誌, vol.37, No.7,pp1378-1387, July 1996.

[3] Kruatrachue, B. : "static task scheduling and packing in parallel processing systems", Ph.D. diss., *Department of Electrical and Computer Engineering*, Oregon State University, Corvallis, 1987.

[4] Message Passing Interface Forum: "*MPI: A Message-Passing Interface Standard Version 1.1*", 1995.

[5] Fujimoto, N., Baba, T., Hashimoto, T., and Hagihara, K.: "A Task Scheduling Algorithm to Package Messages on Distributed Memory Parallel Machines", the proceedings of the International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99), pp. 236-241 , Fremantle, 1999.

[6] Fujimoto, N., Hashimoto, T., Mori, M., and Hagihara, K.: "On the Performance Gap between a Task Schedule and Its Corresponding Parallel Program", the proceedings of the International Workshop on Parallel and Distributed Computing for Symbolic and Irregular Applications, to be published, 1999.

[7] Skillicorn, D.B., Hill, J.M.D., and McColl, W. F. : "Questions and answers about BSP" , Scientific Programming, 6(3): pp.249–274, 1997.

[8] Thurimella, R. and Yesha, Y.: "A scheduling principle for precedence graphs with communication delay", *International Conference on Parallel Processing*, 3, pp.229-236, 1992.

[9] Valiant, L.G.: "A Bridging Model for Parallel Computation", *Communication of ACM*, Vol.33,No8, 1990.