様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２９　年　　６　月　　９　日現在

機関番号：　１４３０１
研究種目：　若手研究(B)
研究期間：　2013～2016
課題番号：　２５７３００４１

研究課題名（和文）耐故障機能を備えたワークスティーリング計算フレームワークの開発

研究課題名（英文）Development of Fault Tolerant Work Stealing Based Computing Framework

研究代表者

平石　拓（Hiraishi, Tasuku）

京都大学・学術情報メディアセンター・助教

研究者番号：６０５２８２２２

交付決定額（研究期間全体）：（直接経費）　　3,100,000円

研究成果の概要（和文）：提案している動的負荷分散フレームワークTascellの実用性向上および耐故障実現に関する研究を行った．具体的には，実用的なグラフマイニングアルゴリズムの並列実装を行い，良好な並列性能を得られることを確認した．また，実行中のタスクの安全な中断などの耐故障機能を開発した．Tascellの実行モデルではsingle point of failureの問題が解決困難であると判断したため，全ワーカが同一の計算をそれぞれ任意の順序で実行しつつ，部分結果を保存・交換しあうことで動的負荷分散や耐故障性を実現するという新しい並列計算モデルの提案も行い，その試験実装および予備評価を行った．

研究成果の概要（英文）：We improved a dynamic load balancing framework called Tascell from the perspective of practicalicity and fault tolerance. We implemented a practical parallel algorithm for graph mining and ensured that it achieves good parallel performance. In addition, we implemented additional functionalities of Tascell such as safe abortion of running tasks. Because we found that the execution model of Tascell can hardly solve the single point of failure issue, we proposed a novel execution model, which realizes dynamic load balancing and fault tolerance by letting all computing workers execute a single program redundantly in arbitrary order and exchange partial results to omit completed parts of the computation. We conducted preliminary evaluations using a prototype implementation of this model.

研究分野：　計算機科学

キーワード：　ハイパフォーマンス・コン ピューティング　プログラミング言語　計算機システム　ソフトウェア開発　効率化・安定化　ネットワーク

1. 研究開始当初の背景

近年のスーパーコンピュータ等の大規模計算システムは，単一のプロセッサの性能向上ではなく，マルチコア／プロセッサノードのクラスタ化，グリッド化といった大規模並列化によって高い計算性能を実現している．そのため，計算機による応用計算を行っている多くの研究者が，分散環境にも対応した並列プログラミングへの移行を余儀なくされている．しかし，そのような並列計算環境でも効率良く計算を行えるプログラムを書くのは容易ではない．特に，充足可能性問題（SAT）やグラフアルゴリズムに代表される，事前に平等な仕事の配分を見積もることができない不規則アプリケーションの並列化は困難であり，可能であったとしても逐次版から根本的なプログラムの書き換えが必要となることも多い．

共有メモリ環境においては，たとえば Cilk 言語がそのようなアプリケーションの並列化を比較的容易に実現できる言語として有名である．Cilk では，多数の論理スレッドを（低コストで）生成して最古優先のワークスティールを行うことで全ワーカを有効活用し，良好な負荷分散を実現する．これに対し我々は，論理スレッドフリーのワークスティーリング計算フレームワーク Tascell を提案している． Tascell では，ワーカは他のアイドルなワーカからタスク要求を受けない限りは一切タスクを生成せず，逐次プログラムのワーカとほとんど同じ計算を実行する．タスク要求を受けると，自身の残りの仕事の一部をタスクとして生成して送信するが，この際，最適な仕事量のタスクを生成するため，バックトラック（後戻り計算）により最古のタスク生成可能状態を一時的に復元する．この手法は，論理スレッド生成・管理のコストを削減する．また，並行に計算を進める可能性がある論理スレッドそれぞれに対して前もって作業空間を準備する必要がないため，作業空間の遅延コピーによるコストの最小化に加え，参照局所性も向上し，Cilk より高い性能が得られる．さらに Tascell は Cilk とは異なり，クラスタ等の分散メモリ環境にも対応している．実際，Tascell は木探索問題で共有メモリ環境においては Cilk の 2 倍程度，分散メモリ環境でもほぼ理想的なスケーラビリティが得られている．さらにその後の研究において，複数のクラスタを WAN で接続した広域分散環境への適用や，グラフアルゴリズムや重力多体問題（Barnes-Hut アルゴリズム）といった実用アプリケーションでの評価を行うことにより実用性を高めてきた．

今後の並列計算環境のさらなる大規模化にともなって（不規則アプリケーションに限らず）懸念されている問題の 1 つとして，耐故障性の問題がある．すなわち，計算機規模が増大すると 1 つの計算に関わる部品点数も増えるため，部品の故障率が一定だとすれば，計算中にどこかの部品が故障する確率も増大する．将来的には，部品の信頼性向上の努力だけでは対応できず，計算中に故障が発生することを前提としたソフトウェア側の対応が不可欠になると予想されている．Tascell は，MPI などと比べて不均一な計算機環境にも対応しており，例えば異なるアーキテクチャをもつ計算ノードを接続したり，前述のように広域分散環境で動作させることもできる．さらに，計算中に計算ノードを新たに参加させることもできる．一方，参加ノードを途中で脱退させることには未対応であり，計算中に故障等で 1 ノードでも脱退してしまえば，計算は継続できず，それまでの途中結果も全て失われてしまうという問題がある．

2. 研究の目的

本研究では，充足可能性問題（SAT）やグラフアルゴリズムのような不規則アプリケーションの並列計算を主なターゲットとして，計算途中でノード等が故障等により脱退しても計算を継続できるような並列計算基盤の実現を目的とする．具体的には，開発中の動的負荷分散フレームワーク Tascell をベースとして，部分結果の保存・取出や失われた部分タスクの自動再計算，ノード間ネットワークの自動修復，およびデータ保存・復旧等を制御できるプログラミングインターフェースを開発することにより上記のシステムを実現する．

3. 研究の方法

平成 25 年度は，再計算ベース，すなわちダウンしたノードが行っていたタスクの部分結果は破棄し，生き残ったノードがそのタスクを再計算することで全体としての計算を完了させる機能を完成させる．平成 26 年度から 27 年度の前半にかけて，さらにワーカがタスクの部分結果を保存し，ノードダウン時には他のノードが取り出せるようにする機能を追加することで，より効率的な復旧が可能な耐故障機能を完成させる．平成 27 年度後半以降は，それまでに開発した拡張 Tascell を生体ネットワーク解析等の実アプリケーションに適用し，性能やプログラミングインターフェースの妥当性の評価およびフィードバックを行う．

Tascell の仕組みの本質的な問題により，Tascell 自体の設計・実装を大きく見直さなければ計画する耐故障機能が実現できないと判断した場合には，計算モデル自体を見直し，再開発することも有り得る．

4. 研究成果

(1) Tascell の機能拡張

耐故障機能開発の一環として，並列実行中のタスクのうち，指定されたものを安全に中断する機能の開発を行った．

```
f0 () {
  try {
    join {
      spawn e();
      f1();
    }
  } catch (E) {
    ...
  }
}
f1 () {
  join {
    spawn f2();
    e();
  }
}
f2 () {
  join {
    spawn e();
    e();
  }
}
e () {
  if (exceptional condition?) {
    throw (E);
  } else {
    ...
  }
}
```

図 1 例外を用いたタスク並列プログラムの例

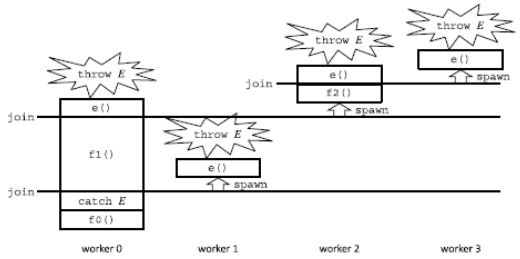

図 2 図 1 のプログラムの実行状態の例

図 3 図 2 例外 catch 後の実行状態

具体的には，Tascell に Java 言語などが持つ，try-catch 構文による例外処理機能を追加した．try ブロック実行中に例外が発生した場合，その例外が catch されるまで計算を巻き戻す点は通常の例外処理機構と同様だが，タスク並列言語である Tascell 固有の機能として，try ブロック内で実行中の全てのタスクが可能な限り早く中断されるような機能を提供する．たとえば，図 1 のプログラムを実行したとき，実行スタックが図 2 のような状態となるようなことがありえるが，この後 worker0 から worker3 のどのワーカが例外を投げたとしても，worker1 から worker3 が実行中だったタスクが中断され，かつ worker0 の実行が関数 f 内の catch を脱出するまで巻き戻される．例外が catch された直後の実行スタックの状態を図 3 に示す．

このような「できる限り早く並列タスクを中断する」例外処理機構は Cilk Plus や X10 などの他の主要なタスク並列言語では提供されていない．また，Tascell ではそれぞれのワーカが異なる計算ノードで動く分散並列環境でも，上記の中断処理は正常に動作し，別ノードのタスクも安全に中断させることができるが，このように分散環境に対応した中断機能を備えるタスク並列言語はこれまでに見当たらない．

なお，この実装は京都大学のスーパーコンピュータ上で性能評価を行い，妥当なオーバーヘッドおよび中断時間で動作することを確認した．

(2) Tascell の応用

上記の追加機能を含む Tascell の実応用に関する研究として，並列グラフマイニングアルゴリズムを Tascell で実装し，その性能を評価した．

対象としたグラフマイニングは，各頂点が複数のアイテムと紐付けられたようなグラフと閾値（整数）を入力とし，全頂点が閾値以上の数の共通アイテムを持つような部分グラフを全列挙するというものである．このようなグラフマイニングはソーシャルネットワーク解析や，タンパク質ネットワークから共通する複数の薬品に反応するような部分を抽出するという創薬研究などに応用可能である．

このグラフマイニング問題を解く効率的な逐次アルゴリズムは COPINE という名前ですでに提案されており，このアルゴリズムの並列化および Tascell によるプロトタイプ実装も本研究課題以前に行ってきた．本課題の研究では，中断処理も用いたさらなる高速化や耐故障に関する検討を行った．

COPINE は，探索中に得た知識をテーブルに書き込み，その知識を利用して不要な部分木の探索を行わないようにする（枝刈り）ことで，探索空間を大幅に削減するが，この仕組みを並列探索にも適用しようとすると，以下の 2 つの問題点がある．

一点目は，元の逐次アルゴリズムの枝刈りでは，探索木をある特定の順序で走査し，知識のテーブルへの登録もその順序で行われることを前提としているため，これをそのまま並列探索にも適用しようとすると過剰な枝刈りが行われてしまい，正しい抽出が行えないという問題である．

この問題を解決するため，テーブルに知識を登録する際にその知識がどの時点における登録かを示す ID（逐次探索において後に訪問される探索木のノードほど大きな ID となる）を付加し，参照時には自らの ID とに付与された ID を大小比較することにより参照できるタスクを制限するという手法を提案した．

この際，探索木の全てのノードに一意の ID を付加するのはノードの数が多すぎて困難であるため，ワークスティールにより獲得される各タスクに対して，minID と maxID の 2 つの符号なし 128 ビット整数（minID ≤ maxID）で表現される範囲 [minID, maxID] を ID として付加する．なお，ルートタスク（実行開始時に最初にワーカに割り当てられるタスク）には，符号なし 128 ビット整数で表現できる最大の範囲 $[0,2^{128}-1]$を付加する．ワークスティールによりタスクの分割が起こると，分割元のタスクに割り当てられている ID の範囲を半分に分割し，左半分と右半分の範囲をそ

れぞれ分割元と生成されるタスクの ID として付加する．知識テーブルに知識 I を登録する際には，実行中のタスクの minID の値を I に付加する．この知識 I を参照する際に，それに紐付けられた ID と自身のタスク ID を比較することで，利用してはいけない知識の利用を回避し，過剰な枝刈りを防ぐことができる．

なお，[minID，maxID]の範囲の分割を繰り返すと，ある時点で minID = maxID となり，それ以上の分割ができなくなる．そのようなタスクはそれ以上の分割を許さず，逐次実行されるようにする．分割できない逐次タスクの実行時間が長くなると負荷分散に悪影響を及ぼすが，たとえばタスク ID に符号なし 128 ビット整数を用いる場合には，ルートタスクから 128 回までの再帰的なタスク分割が可能であり，実際，この手法で逐次タスクの実行時間は十分短くなり，実用上の問題はないことを確認した．

並列探索における枝刈りにおける二点目の問題点について述べる．逐次探索においては，探索部分木に対する枝刈りは，当然その探索開始前に行われる．ところが，並列探索では，あるワーカがある部分木に対する枝刈りを実施しようとしたとき，その部分木が他のワーカによってすでに走査が開始されているということが起こりうる．これに対して何の対処も行わなければ，枝刈り対象であった部分木が完全に走査されてしまい，無駄な探索が無視できないほど増えてしまう．

この問題を解決するため，ワーカが探索部分木に対する枝刈りを行う際，別のワーカがその部分木を探索中でないかをチェックし，もし探索中であればそのワーカにその部分木は枝刈り対象であることを通知することで不要な探索の中断を促す仕組みを提案・実装した．通知を受けたワーカは，前節で述べた例外処理機構における例外を投げることで不要な探索を中断するようにした．すでに述べた通り，この例外処理機構は，脱出する try ブロックで実行されていた全ての並列タスクを早急に中断するものであるため，ここで中断される探索の一部を子タスクとして別のワーカに依頼していた場合でも，それら子タスクも全て即座に中断させることができる．

これらの仕組みを取り入れた COPINE 並列実装の性能を，京都大学のスーパーコンピュータの 28 コアを持つ 1 ノードを用いて評価した．その結果を図 4 に示す．評価の結果，並列版 COPINE でも枝刈りを利用する探索を正しく行うことができ，また，例外による中断機構を適用することで探索空間の削減ならびに実行時間の短縮を実現できることが確認できた．

この実装は，分散環境でも動作および中断による探索空間削減効果は確認できているが，スティールコストの増大などの影響により，満足する並列性能を得ることはできなかった．この解決は今後の課題である．

なお，ここで提案したタスク ID および中断の機構は耐故障性の観点でも有用である．すなわち，ノードが利用不可能になれば当該ノードで動いているタスクを例外で中断することが考えられる．また，中断後の再開の際には，再開するタスクを

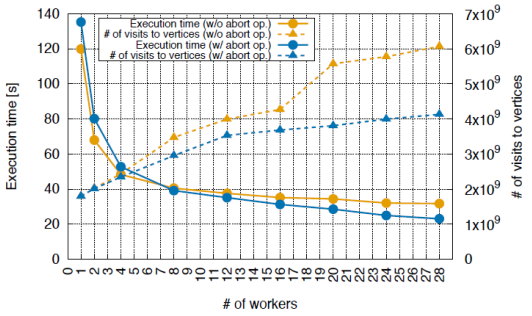

図 4 並列版 COPINE 実装の性能評価結果

何らかの方法で同定することが必要となるが，そのためにここで提案した ID を活用することが考えられる．

(3) 耐故障性を実現するための新たな計算モデルの設計

本研究の過程で，Tascell の実行モデルには，実行のルートに近いタスクを中断させると，ほぼ全ての実行をやり直さなければならないという，Single Point of Failure の問題があり，この問題は本質的に解決困難であるとの結論に至った．そのため，耐故障性を重視した Tascell にかわる新しい動的負荷分散の実行モデルを考案し，そのプロトタイプ実装を行った．

提案したモデルは，全ワーカが同一の計算をそれぞれ任意の順序で実行しつつ，部分結果を保存・交換しあうことで動的負荷分散や耐故障性を実現するというものである．

これまでの成果として，部分結果を保持・交換しあう機構である「メッセージ媒介システム」のプロトタイプ実装を行った．また，この実行モデルによる計算を記述する拡張 C 言語を(OpenMP のような)ディレクティブベースの言語として設計した(このコンパイラは現在開発中である)．さらに，実行モデルのプロトタイプ実装を行いメッセージ媒介システムと組み合わせることで，耐故障性および妥当な並列性能を得られることをマイクロベンチマークを用いた評価により確認した．

5．主な発表論文等

(研究代表者、研究分担者及び連携研究者には下線)

〔雑誌論文〕(計 4 件)

① Shingo Okuno, Tasuku Hiraishi, Hiroshi Nakashima, Masahiro Yasugi, Jun Sese. Parallelization of Extracting Connected Subgraphs with Common Itemsets in Distributed Memory Environments. Journal of Information Processing. 査読有, Vol.25, pp.256-267, 2017.
DOI: 10.2197/ipsjjip.25.256

② Tasuku Hiraishi, Shingo Okuno, Masahiro Yasugi. An Implementation of Exception Handling with Collateral Task Abortion. Journal of Information Processing. 査読有.

Vol.24, pp.439-449, 2016.

③ Shingo Okuno, Tasuku Hiraishi, Hiroshi Nakashima, Masahiro Yasugi, Jun Sese. Parallelization of Extracting Connected Subgraphs with C om m on Item sets. IPSJ Trans. Programing. 査読有, Vol.7, No.3, pp.22-39, 2014.

④ 田附 正充, 八杉 昌宏, 平石 拓, 馬谷 誠二. L-Closure の呼び出しコストの削減. 情報処理学会論文誌 プログラミング. 査読有, Vol.6, No.2, pp.13-32, 2013.

〔学会発表〕(計 24 件)

① Hiroshi Yoritaka, Ken Matsui, Masahiro Yasugi, Tasuku Hiraishi, Seiji Umatani. Extending a Work-Stealing Framework with Probabilistic Guards. Ninth International Workshop on Parallel Programming Models and Systems Software for High-End Computing P2S2 2016 (held in conjunction with ICPP2016). 2016 年 8 月 16 日. Temple University Center City, Philadelphia, PA, USA.

② Daisuke Muraoka, Masahiro Yasugi, Tasuku Hiraishi, Seiji Umatani. Evaluation of an MPI-Based Implementation of the Tascell Task-Parallel Language on Massively Parallel Systems. Ninth International Workshop on Parallel Programming

③ Models and Systems Software for High-End Computing P2S2 2016 (held in conjunction with ICPP2016). 2016 年 8 月 16 日. Temple University Center City, Philadelphia, PA, USA.

④ Tasuku Hiraishi, Shingo Okuno, Daisuke Muraoka, Masahiro Yasugi. Exception Handling with Collateral Task Abortion in Distributed Memory Environments. ISC 2016 HPC in Asia Posters. 2016 年 6 月 22 日. Frankfurt Messe, Germany.

⑤ Shingo Okuno, Tasuku Hiraishi, Hiroshi Nakashima, Masahiro Yasugi, Jun Sese. Reducing Redundant Search in Parallel Graph Mining using Exceptions. 21st International Workshop on High-Level Parallel Programming Models and Supportive Environments HIPS2016 (held in conjunction with IPDPS2016). 2016 年 5 月 23 日. Chicago Hyatt Regency, Chicago, IL, USA.

⑥ 重本 孝太, 八杉 昌宏, 平石 拓, 馬谷 誠二. HOPE コンパイラの実装に向けて. 日本ソフトウェア科学会プログラミング論研究会第 18 回プログラミングおよびプログラミング言語ワークショップ (PPL2016) カテゴリ3. 2016 年 3 月 7 日. 岡山県玉野市ダイヤモンド瀬戸内マリンホテル.

⑦ 諏訪 将大, 八杉 昌宏, 平石 拓, 馬谷 誠二. 分散進捗管理のためのメッセージ媒介システムにおける不要メッセージ削除機能. 並列/分散/協調処理に関するサマー・ワークショップ (SW oPP2015):第 106 回プログラミング研究会. 2015 年 8 月 6 日. 別府国際コンベンションセンター.

⑧ Shingo Okuno, Tasuku Hiraishi, Hiroshi Nakashima, Masahiro Yasugi, Jun Sese. Reducing Redundant Search using Exception Handling in a Task-Parallel Language. Annual Meeting on Advanced Computing System and Infrastructure (ACSI) 2015. 2015 年 1 月 28 日. 茨城県つくば市つくば国際会議場.

〔その他〕
ホームページ等
http://super.para.media.kyoto-u.ac.jp/tascell/

6. 研究組織
(1)研究代表者
平石 拓(HIRAISHI, Tasuku)
京都大学・学術情報メディアセンター・助教
研究者番号: 60528222