大域分散並列コンピューティング

田浦健次朗

情報理工学系研究科電子情報学専攻

概 要

本研究課題では，インターネットに接続された多数の計算機群を並列計算に利用するための基盤ソフトウェアに関する研究を行っている．想定している条件は，計算は長時間 (数ヶ月またはそれ以上) 持続し，計算資源の状況 (ノードやネットワーク) は，それより短い時間スケール (数時間おきやそれ以下) で変化していくという環境であり，その条件の下で並列計算を持続させる枠組みが必要になる．本研究の主たる研究テーマはこれを実現するための並列計算の記述の枠組み「Phoenix モデル」，その実装方式，それを基にした効率的な，資源の再構成が可能な耐故障計算の枠組み，およびその応用である．また，大域環境で計算機を効率よく利用するための日常的なツール (シェルやファイル同期ツール) の開発も目標にする．

1 はじめに

本年度の主な成果は以下の通りである．

Phoenix システム第 0.9 版の実装: 提案モデル Phoenix の初期版を実装した．詳しくは 2 節で述べる．

ノード脱退を含めた移送プロトコル: Phoenix 上で，アプリケーションの状態と Virtual Processor の移送 (migration) を行うプロトコルを設計，実装した．これにより，並列計算をとめないまま，台数を減少させるプログラムの実行が可能になった．また，モデル検査ツールを用いて検証を行った．

Phoenix モデルでの並列アルゴリズム記述: Phoenix モデルはメッセージパッシングに準ずるが，並列アルゴリズムの記述にあたっては，通常のメッセージパッシングと異なる考え方をしなくてはならない部分もある．いくつかの並列アプリケーションを Phoenix モデルで実際に記述，実行することで，Phoenix モデルの並列プログラミングモデルとしての有用性，一般性を検証した．

Virtual Private Grid の第 1 版公開: 広域に分散した多数のノードを有効利用できるシェルで，技術的にも Phoenix と共通部分の多かった Virtual Private Grid について，ソフトウェアの公開を始めた．今後も改良し，継続して改版をしていく予定である．

多ノードファイル同期ツール NetSync の開発: 多数のノードに大きなファイルを耐故障的にコピーするツール NetSync を開発した．スケーラブル (ファイルのソースがボトルネックにならない)，耐故障的 (クラッシュしたノードは他のノードの進行に影響を与えない，また独立に復活して続きを実行できる)，効率よい転送路の自動構築などの特徴を持っている．

Virtual Private Grid に関する国際会議での発表: Virtual Private Grid に関する論文は，100 編以上の投稿中 23 本が採択された，「Grid およびクラスタ計算に関する国際会議 (CCGrid)」に採択，発表された．さらにその中から論文誌への投稿を推薦された 7 本の論文のうちのひとつに選ばれた．現在論

文誌は査読中である．

以下では，最初の 3 つの項目について説明する．

2 Phoenix モデルの実装

昨年度の実装と異なる，以下のような特徴を持った Phoenix モデルの実装 (公開できる基準を 1.0 版としたとき，第0.9版と言うべき版) を完了した．

- 自由なトポロジーでノード間を接続することを許す．IP フィルタリングによって接続がブロックされているノード対や，DHCP/NAT などの制限により，片方向にしか接続不可能なノード対があっても動作する．そして，そのトポロジーは自由であり，許される多数の接続を用いてバンド幅を有効利用できる．

- ルーティング表を構築･維持する．上記で作られたトポロジー上に，任意ノード宛てのメッセージ配送が可能になるようにルーティング表が構築される．メッセージのあて先ノードは仮想ノード名で指定され，物理ノード ↔ 仮想ノード間のマッピングを変更することで移送 (migration) が可能になる．

要約すると，当初の Phoenix の目的である，広域分散環境で，ノード数を増減させながらメッセージパッシング計算を行う枠組みが動作している．

3 移送･ノード追加･脱退プロトコル

計算中のノードの追加や脱退を可能にする上で基礎となるのが，物理ノードが担当する仮想ノードの集合を，アプリケーション状態とともに物理ノード間で移送するプロトコルである．提案プロトコルでは，

- 各ノードが自律的に移送を開始できる．もちろん他のノードと衝突した場合 (例えばほぼ同時に二つのノードが同じノードに対して移送を試みた場合) に失敗することはある．

- 移送を開始したノードは，他のノードに対して仮想ノード集合を送る (PUSH 型) 移送を行うことも出来るし逆に他のノードの仮想ノード集合を奪う (PULL 型) 移送を行うことも出来る．

- 仮想ノードの移送にあわせてアプリケーション状態を移送させることが出来る．そして，移送に関与しない全てのノードからはその移送を透明にすることが出来る．

1 番目と 2 番目の性質によって，各ノードは自律的に計算への参加・脱退を行うことが出来るようになる．参加を行う際には PULL 型の移送を開始し，脱退を行う際は PUSH 型の移送を開始すればよい．また，3 番目の性質によって，参加・脱退があるにも関わらず，アプリケーションプログラマは大部分において，通常の，ノード数一定を仮定した並列プログラムを記述することが可能になる．

さらなる性質として，設計したプロトコルでは，各ノードは仮想ノード集合として，ひとつの区間 (一連の連続する整数からなる集合) を担当するように保たれる．これは，後で述べる並列アルゴリズムの記述において，負荷分散などを容易にする．

設計したプロトコルは Phoenix 上で実装された．また，プロトコルの正しさに対する確信を高めるため，プロトコルをプロトコル記述言語 Promela で記述し，モデル検査器 SPIN によって検証を行った．状態数が多いため，プロセス数 10 程度でも完全な検証は不可能であったが，3 億以上の状態を探索してエラーは見つからなかった．

4 Phoenix モデルでの並列アルゴリズム記述

以下の 3 つの並列アプリケーションを実際に記述した．

1. 動的負荷分散に基づく Ray Tracing の並列化
2. 並列整数ソート
3. 並列 LU 分解

1. は Ray Tracing プログラムとして広く配布されている POV-ray というソフトウェアを並列化したものである．Work stealing による負荷分散を行っている．2 および 3 は配列や静的負荷分散などを用いた，一件 Phoenix で記述しにくそうに見える並列計算の例題である．どちらも，広域で長時間動き続けることを目的とした応用ではないが，Phoenix の並列プログラミングモデルとしての一般性を確認する，Phoenix と通常のメッセージパッシングで異なる考え方が必要な点を一般化して Phoenix での並列アルゴリズム記述法を確立する，および基本性能を他のシステムと比較する，などを目的として記述した，

以下では紙面の都合上，Ray Tracing と整数ソートに関して得られた知見を説明する．

動的負荷分散に基づく Ray Tracing の並列化 動的負荷分散手法として有効性の実証されている，Random Work Stealing に基づく並列化・負荷分散を行っている．各ノードがローカルな task queue を持ち，ローカルな task がなくなると，他のプロセッサに random に仕事の移送要求 (task steal) を出す．Phoenix でこの負荷分散を記述するのは容易である．Random なプロセッサに対する task steal 要求は random な仮想ノードに要求を出せばよい．

実験環境として，Sun Blade 1000 ワークステーション 120 ノードから成るクラスタを用いた．台数効果を測定したものが図 1 である．

並列整数ソート これは bucket ソートに基づくアルゴリズムである．詳しくは文献 [2] を参照されたい．

図 2 に，32 ノード Linux クラスタ (100Mbps リンク) における，整数ソートの台数効果と，MPI を用いた同じプログラムとの比較をしめす．MPICH と同等か，良い性能を得ていることがわかる．

また，紙面の都合上説明を省略しているが，並列 LU 分解についても，Sun Blade 1000 ワークステーションの 16 ノード 32 CPU のクラスタで台数効果と，MPICH および HPL との性能比較を行った．結果は図 3 にあり，ここでも MPICH や HPL と遜色ない性能が得られた．

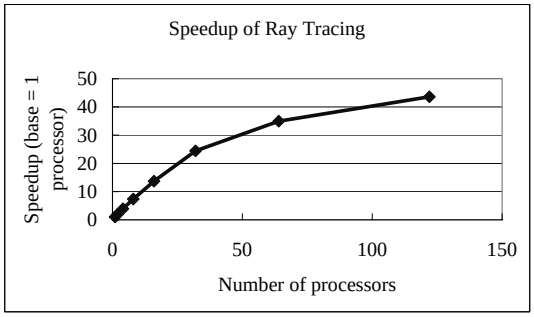

図 1: Speedup of Ray Tracing

5 おわりに

本年度の主な成果として，本格使用に耐えうる Phoenix システムの実装の，初期版が完成したこと，アプリケーション状態の移送を行うプロトコルを設計，実装，検証し，それを用いて動的にノードが増減するシステムの礎が出来たことを述べた．システムとしては当初の目的が達成されつつあるので，今後は，実装の改良を重ねるとともに，広域・異機種の環境で，非常に容易な設定と利用を可能にする，耐故障性などの広域環境で重要な要素を取り入れる，などの方向性に発展させていく予定である．

参考文献

[1] Kenji Kaneda, Kenjiro Taura, and Akinori Yonezawa. Virtual private grid: A command shell for utilizing hundreds of machines efficiently. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2002)*, 2002.

[2] Kenjiro Taura, Kenji Kaneda, Toshio Endo, and Akinori Yonezawa. Phoenix: a parallel programming model for accommodating dynamically joining/leaving resources. In *ACM SIGPLAN Symposium on Principles*

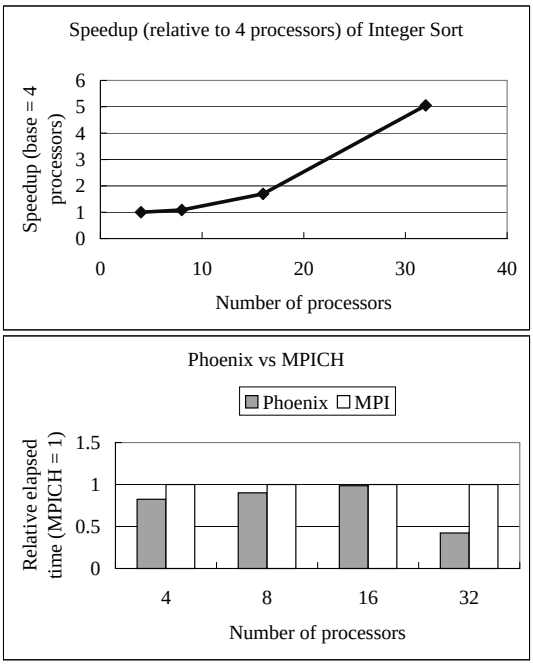

図 2: Speedup (left) and comparison to MPICH (right; the smaller the faster) of Integer Sort

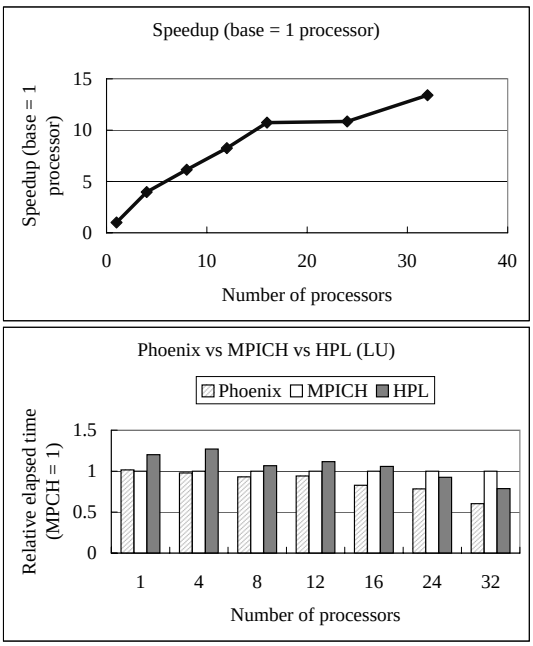

図 3: Speedup (left) and comparison to MPICH and HPL (right; the smaller the faster) of LU factorization

and Practice of Parallel Programming, 2003 (to appear).