広域分散計算のための階層的負荷分散手法*

3ZB−07

川戸 正裕　蒲池 恒彦　妹尾 義樹†
日本電気 (株) インターネットシステム研究所‡

1 はじめに

近年のインターネットの高速化に伴い，地理的に分散した計算資源を組み合わせて大規模な計算を行なう **Grid コンピューティング** [1] が現実的になりつつある．今までに，Grid を構成するための基本ソフトウェアとして **Globus Toolkit**[2] などが開発されているものの，広域に分散した計算資源を効率的に利用する技術については未発展である．そこで，本稿では，Grid の有望なアプリケーションの 1 つである**タスクファーム (task farm)** について，高性能が得られる構成方法を検討する．タスクファームは，独立したタスクの集合を複数の計算ノードに割り当てて並列に実行するもので，代表例としてパラメータスイープやコンピュータグラフィックスによる動画のレンダリングがある．

2 階層的タスクファーム

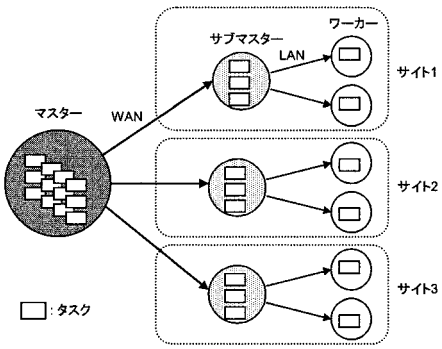

図 1: 階層的なタスクファームの構成

Grid 上にタスクファームを構成する方法として，大きく分けて次の 2 つが考えられる．

- 階層のない構成: 唯一の**マスター**ノードが，他のすべての**ワーカー**ノードにタスクを分配し，結果を収集する．ワーカー間の負荷分散のために，マスターが各ワーカーからの負荷情報の収集とタスクの移動の制御を行なう．
- 階層的な構成: 各サイトに**サブマスター**を設置し，マスター⇔サブマスター⇔ワーカーの順にタスクの分配および結果の収集を行なう (図 1 参照)．サブマスターは，同じサイト内のワーカーから負荷情報を収集し，サイト内でのタスクの移動を制御する．マスターは，各サブマスターから各サイトの負荷情報を収集し，サイト間のタスクの移動を制御する．

後者の方法は，前者に対して以下の利点を持つ．

- マスターの負荷が軽減され，マスターの負荷がボトルネックになることを避けることができる．
- 同一サイト内のワーカー間での負荷分散で，ワーカーでの負荷情報の取得からタスクの移動開始までの時間が短く，不適切なタスク移動が起こりにくい。
- マスター⇔サブマスター間で，複数のタスクについての入出力データを 1 度の通信で送ることで，WAN を通した通信のレイテンシを削減できる．

これらの利点から，以降では階層的な構成を前提とする．

3 実装方式の検討

次に，階層的タスクファームの実装方法として，各階層においてそれぞれどの通信手段を用いるべきかを考察する．考慮すべき点として，セキュリティ，通信速度，プロセスの生存期間が挙げられる．

セキュリティ　WAN 経由の通信を行なうマスター⇔サブマスター間では，セキュリティが重要な要素になる．Grid を利用したタスクファームでは，各サイトを異なる組織が管理している場合などに，サイトごとにセキュリティ機構が異なっていたり，同一ユーザについてのユーザ ID が異なる可能性がある．このような状況を扱うためには，Globus Toolkit に含まれるセキュリティ機構 **Grid Security Infrastructure** (GSI) と通信ライブラリ **Globus I/O** の組み合わせが適している．Globus I/O はソケットと類似した API に加え，GSI による認証のための API を提供する．一方，サブマスター⇔ワーカー間の通信については，同一サイト内での通信となるため，強力なセキュリティ機構は必ずしも必要ない．

通信速度　通信速度はどちらの階層でも重要であるが，マスター⇔サブマスター間の通信では，複数のメッセー

*Hierarchical Load Balancing for Wide-area Distributed Parallel Computation
†Masahiro Kawato, Tsunehiko Kamachi, Yoshiki Seo
‡Internet Systems Research Laboratories, NEC Corporation

ジを一度のコネクションで送ることが可能であるため，ある程度のレイテンシの増加は許容できる．サイト内の通信ではレイテンシが小さいことが重要であり，ソケットを直接使用した軽量プロトコルを作成することが有力な候補である．一方，Gridのための RPC システムである **Ninf**[3] は，軽量な通信プロトコルを用いており，サイト内通信手段の候補となりうる．ソケット通信と Ninf の性能については，4節での実験で比較する．

プロセスの生存期間　Ninf では，クライアントからの要求を処理するプロセスは，要求の処理が完了すると消滅する．この性質から，Ninf サーバをワーカーノードに設置することはできるが，マスターやサブマスターに Ninf サーバを用いることできない．他の通信手段ではこの制約はない．

4　サイト内通信手段の比較

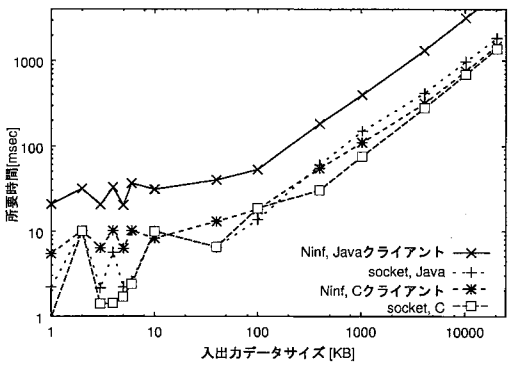

図 2: タスク実行の所要時間

前節で検討したサイト内通信の実装手段を比較するため，サブマスター⇔ワーカー間の通信部分を複数の方法で実装し，タスク実行の所要時間を測定した．この測定の目的は，以下の 2 点を明らかにすることである．

- サイト内通信に Ninf を使う場合とソケットを直接使う場合の性能差
- C 言語と Java 言語を使った場合の性能差

この目的のために，以下の 4 種類の通信方式について，単純なタスク実行 (配列の各要素に 1 を加える) の所要時間を測定した．

- Ninf，C 言語版 Ninf クライアント
- Ninf，Java 言語版 Ninf クライアント
- ソケット，C 言語
- ソケット，Java 言語

1000BASE/T で接続された 2 台の PC (Pentium III/866MHz, Linux 2.2.18) を使い，データサイズ (入力データサイズ = 出力データサイズ) を変えて測定した結果を図 2 に示す．タスク実行の所要時間はクライアント側で測定し，通信時間とタスク実行時間を含む．この結果，データサイズ 100KB 程度以下のときには C 言語とソケットの組み合わせが有利であるが，それ以上のデータサイズでは，Java 版 Ninf クライアント以外の 3 方式間で大きな差は出なかった．Java 版 Ninf クライアントでは他の方法に比べて 4 倍程度遅いという結果になっている．この原因は分かっていないが，Java 版 Ninf クライアントライブラリは α バージョンという位置付けであることから，最適化が十分でないことが予想される．

5　結論

本稿では，広域分散システム上で高い性能を得るための階層的タスクファームを提案し，その実現方法についての選択肢を検討した．

本稿で提案した階層的なタスクファームは，WAN を通した通信のレイテンシの影響が小さく，計算と通信の負荷を効果的に分散するという特長を持つ．階層的なタスクファームの実装する上で，サイト間とサイト内の通信手段を分けて考える必要がある．サイト間では，セキュリティ機能の観点から Globus Toolkit の利用が適している．サイト内では，性能面から，ソケットを直接使用した軽量な通信プロトコルを実装するのが有利である．実験では，各タスクの入出力データがある程度大きい場合には，Ninf でもソケットの直接使用と遜色ない性能が得られており，Ninf を選択することも現実的な選択肢である．

今後の課題として，試作システムを実装し，それを使ったより詳細な性能評価を行なうことが挙げられる．また，階層的なタスクファームにおける効果的な負荷分散手法の開発を進めていく予定である．

参考文献

[1] Ian Foster and Carl Kesselman (editors). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, 1998.

[2] Ian Foster and Carl Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *Intl J. Supercomputer Applications*, 11(2):pp.115-128, 1997.

[3] 中田 秀基, et al. Ninf による広域分散並列計算. 並列処理シンポジウム JSPP '97 論文集, pp.281-288, 1997.