研究課題別評価

１．研究課題名：

分散管理された計算機の高度な協調利用

２．研究者氏名： 田浦 健次朗

研究員： 遠藤 敏夫 （ **研究期間** H.14.12～H.16.03 ）

３．研究の狙い：

今日、計算機による大規模な問題の解決は、多数の計算機を協調(並列)処理させて行われている。協調・並列処理を行うためのハードウェアは共通化、廉価化しており、多数の PC やワークステーションを、Gigabit Ethernet などのネットワークで結合した、いわゆるクラスタ計算機が容易に手に入るようになっている。そして広域ネットワークの高性能化に伴って、複数の管理ドメインにまたがった多数の拠点を結合した協調処理や、Web 上の情報源を統合した計算サービスへの関心が高まっている。したがって、そのような広域に分散する計算資源を統合し、協調処理に利用するための環境が重要となる。しかし現実には、管理ドメインをまたがった計算機資源の統合には現在でも多大な人的コストがかかる。協調処理のためのソフトウェアはさまざまなものが提案・実装されているが、単一の管理ドメイン、単一の LAN を想定して設計されているものが多く、広域環境での利用が困難であったり、導入に多大な労力を要したりするものが多い。

我々はこの研究で、分散管理された多数の計算資源を、**設定や導入を含め、非常に少ない人的コストで容易に利用可能にするための環境を構築する**ことを目指した。より具体的には、(1)ユーザが、ネットワークに存在する全計算機の情報を知らなくても、システムがそれを発見・提示し、ユーザが容易にそれらを選択可能なシステム、(2) 利用可能な計算機群全体を接続するグラフを構築し、それを利用して全計算機に瞬時にコマンドを投入することができる、(3)多数のジョブをスケーラブルかつ耐故障性を持って管理、スケジュールすることができる、ことを狙いとした。それらを、複数管理ドメイン・広域にまたがった資源に対し、小さな導入・設定コストで可能にすることを目指した。

４．研究成果

4.1 GXP 並列・分散シェル

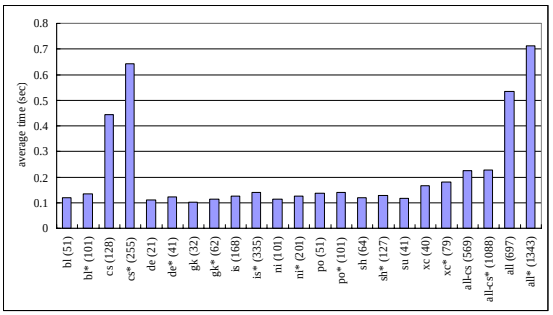
「研究のねらい」の節で述べた目標は、GXP (Grid Explorer)という実用的な**分散環境のためのシェル**として実現され、公開されている。

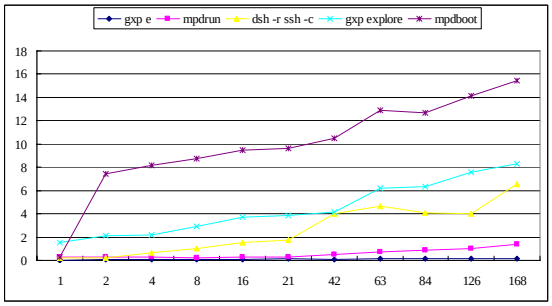
(http://www.logos.ic.i.u-tokyo.ac.jp/phoenix/ gxp_quick_man_ja.shtml)

研究終盤には計算機の状態をグラフィカルに表示して、利用者が容易に資源の混雑状態や、動いているユーザやプロセスを把握できるようにするためのインタフェースを開発した。「狙い」で述べた目標を達成するために以下のような特徴を持って設計・実装されている。

- Firewall や NAT が存在する、今日典型的なネットワーク設定の下で動作する。
  Firewall な NAT によって直接接続することが禁止されているホストへも、多段の接続を維持することで到達することができる。
- GXP が動作するための事前用件(依存するソフトウェア)を極力小さくしている。
  SSH と Python が動作するホストであれば動作する。
- GXP を多拠点に導入するためのコストを極力小さくする。
  ファイルシステムによってプログラムが共有できないホストへは、GXP 自身が自動的に自分のプログラムをコピーする。つまり、各ホストへの明示的なインストールが不要である。
- 分散管理された資源を統合利用するのに管理者権限を必要としない。
- 高速に動作する。短いコマンドに対して速い/対話的な応答を提供する。

下図左は、各種クラスタで、プロセスの立ち上げから終了までの時間を計測したものである。時間は環境によっても異なるが、all-cs* のバーは、本郷、柏、大岡山、つくば、徳島などにまたがる 9 つのクラスタにおいて 1088 のプロセスの起動と終了を 0.2 秒強で行えることを示している。





上図右は、類似ソフトウェアとの性能比較で、1 つのクラスタ内の各種台数(横軸)での、プロセス起動+終了にかかる時間を比較している。一番下の青い線が GXP のそれであり、特に台数が大きいほど、他のシステムとの差が大きくなる。dsh はプロセス起動のたびに認証を必要とするため、また mpdrun はプロセス間の接続を木構造ではなくリング構造としているために、それぞれ GXP のプロセス起動に比べると時間を要する。総じて、GXP は多数の計算機からなる環境下においても対話的な操作を快適に行えるだけの応答性を実現している。

4.2 GXP を用いたジョブスケジューラとその活用事例

GXP の基本機能は、上で述べたように同じコマンドを多数ノードで一斉に起動することである。GXP は、そのわずかな延長として、多数コマンドの出力をマージして、ローカルなプロセスに入力する、逆にあるローカルなプロセスの入力を多数のプロセスすべてに入力する(ブロードキャスト)ための機能を備えており、これによって単純なプロセス間通信、プロセスの制御を行うこともできる。これを利用すると、ネットワークプログラミングを一切用いることなく、単純な協調・並列処理を行うことができる。我々はこの仕組みの上に単純なジョブスケジューラを設計した。インタフェースは、ファイルに実行したいコマンドを羅列すると、あいたホスト上で順に実行していくだけのものであり、既存のバッチスケジューラの機能を単純化したものである。しかし、複数の管理ドメインにまたが

った資源を用いて大量のコマンドを分散実行するための、非常に導入コストの低い方法として意義がある。

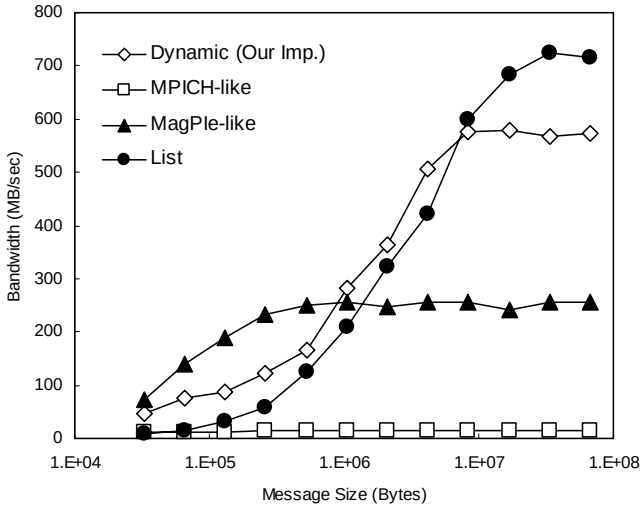
我々は、東京大学辻井研究室と共同で、このツールを用いた自然言語処理実験を行った。辻井研究室が研究開発した英語の HPSG 構文解析器 enju を用い、生物医学系の文献データベースである MEDLINE の全アブストラクト(約 1500 万件。7000 万文)を解析した。本郷キャンパスと柏キャンパスにまたがった 340 のプロセッサを用いて約 1 週間をかけ、全件の構文解析が終了した。この間、我々のプロセスは優先度最低の状態で実行しており、他のユーザを排除して実験を行ったわけではない。

4.3 適応的通信ライブラリ

分散環境ではネットワーク性能が均一ではなく、特に LAN をまたがるリンクは、多数のノード間で共有されている外部トラフィックと共有されているなど、さまざまな理由でボトルネックとなることが多い。一般に分散環境で協調処理を行う場合、このようなネットワーク性能の非均一性を考慮に入れて、通信パターンや負荷分散方式を設計しなくてはならない。

我々は、並列プログラムが起動した環境の測定と、それに基づいたブロードキャスト・リダクションの転送路の自動構築アルゴリズムの研究を行った。アルゴリズムは単純な分散アルゴリズムで、各プロセスが自律的に転送木の親を探していく。基本は、自分により近いプロセスを親とすることであるが、これに加え、(1) 短いメッセージに対しては、木が深くなりすぎないこと、(2) 長いメッセージに対しては、プロセスの子の数を少なくする(1 に近づける)ように親を選択する。前者は、プロセス間のメッセージの遅延の積み重ねを避けることを狙ったものである。後者は、ホスト間の負荷分散、すなわち同じメッセージが同じホストから多数回送られるのを避けることを狙ったものである。

下図はさまざまな大きさのメッセージ(横軸)に対し、さまざまな方式によるブロードキャストのバンド幅を測定したものである。4 つのクラスタで、合計 137 のプロセスを用いている。MPICH-like (下図の□)は、通信性能の均一性を仮定した転送木を構築するアルゴリズムで、これを分散環境で動作させると、非常に低い性能しか得られないことが明確に現れている。MagPie-like (図の▲)は LAN の内外の区別をし、起点となるプロセスが各 LAN の一プロセスにメッセージを配送し、各 LAN の中では MPICH と同様のアルゴリズムを用いるものである。これは LAN 内・外というグループ情報をユーザが与える必要がある上に、メッセージサイズが大きいか、LAN の数が多いと最適なアルゴリズムではなくなる。List (図の●)は、全プロセスをリスト上につないだ転送路(つまり、子の数が 1 の木)を構築するもので、かつ list 上の隣り合うプロセスが LAN をまたがる回数を最小(つまり、この実験では 3)にしたもので、大きなメッセージに対して最適な転送路を手動で設定したものである。Dynamic (図の◇)は我々のアルゴリズムで、手動の情報を与えられない状態で、小さなメッセージから大きなメッセージまで、平均して高い性能を示している。



５．自己評価

分散処理の研究を効率的に進めるために不可欠で、かつ他の研究者にも有用なソフトウェア(GXP)を開発できたことには満足している。これは今後どんな研究をするにあたっても必要な、基本的成果といえると思う。一方で耐故障性を持つ並列処理・タスクスケジューラに関しては、実践的に有用なツールの開発には成功したものの、深みのある研究成果を出すにはいたらなかったと思っている。故障の存在下で失われる仕事量と、通信量のトレードオフやその限界について基礎的な結果を出し、それを実用的なアルゴリズムの開発に反映させることを今後の課題としたい。
ポスドク研究員と一緒に研究を進められたことは非常に有意義であり、研究のアプローチも多様化するし、個々のアプローチも議論・フィードバックによって深化させることができた。

６．研究総括の見解

並列計算に対するプログラミングはこれまで一般利用者にとっては困難であったが、本研究は多くの計算機資源をできるだけ少ない人的コストで統合し並列処理をよりたやすく実行できるソフトウエアの研究を行い顕著な成果を挙げた。ごく少数の標準的ツール(SSH/Python)のみを使って多数のPCの同時使用による効率的並列計算を誰にでも可能にする並列分散環境用のシェルGXP を開発し、さらに、起動した並列計算環境の測定とそれを用いた転送路の自動構築アルゴリズム(Phenix)を立ち上げ並列計算の高効率化を図ってこの分野の進歩に大きく貢献したことは高く評価される。

７．主な論文等

1. Toshio Endo and Kenjiro Taura. Highly Latency Tolerant Gaussian Elimination. Grid 2005 - 6th IEEE/ACM International Workshop on Grid Computing. (Grid2003), Seattle, pp.91-98, 2005.11.13-14.
2. Hideo Saito, Kenjiro Taura, and Takashi Chikayama. Collective Operations for Wide-Area

Message Passing Systems Using Adaptive Spanning Trees. Grid 2005 - 6th IEEE/ACM International Workshop on Grid Computing. (Grid2003), Seattle, pp.40-48, 2005.11.13-14.

3. Yuuki Horita, Kenjiro Taura, and Takashi Chikayama. A Scalable and Efficient Self-Organizing Failure Detector for Grid Applications. Grid 2005 - 6th IEEE/ACM International Workshop on Grid Computing. (Grid2003), Seattle, pp.202-210, 2005.11.13-14.
4. Kenji Kaneda, Toshio Endo, Kenjiro Taura, Akinori Yonezawa. Routing and Resource Discovery in Phoenix Grid-Enabled Message Passing Library. IEEE/ACM Symposium on Cluster Computing and the Grid (CCGrid2004). pages 670-677, 2004.
5. Kenjiro Taura. GXP : An Interactive Shell for the Grid Environment. Proceedings of Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'04). pages 59-67, 2004.
6. Kenjiro Taura, Toshio Endo, Kenji Kaneda, Akinori Yonezawa. Phoenix : a Parallel Programming Model for Accommodating Dynamically Joining/Leaving Resources. ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP 2003), pages 216-229.

他論文 12 件、口頭発表 26 件