協調バックグラウンドタスクスペースに関する検討

3M-7　　山口 実靖 *　　瀧本 政雄 **　　相田 仁 ***　　齊藤 忠夫 *

* 東京大学大学院工学系研究科　** 東京大学工学部　*** 東京大学大学院新領域創成科学研究科
{sane,takimoto,aida,saito}@sail.t.u-tokyo.ac.jp

1　はじめに

近年, 個人用計算機も含め多くの計算機が長時間ネットワークに接続されている. これらの個人用計算機は常に高い負荷がかかっているわけではない思われる. 我々は文献 [1] において “ローカルの作業に弊害を出さずにアイドル状態にある計算資源を有効的に活用する” 手法を提案した. 本稿では提案手法の Java 言語による実装について述べ, 提案システムおよび実装システムの有効性を示す. ただし, ローカルの作業とはワードプロセッサなど対話的作業 (以後 “FG タスク”と呼ぶ) であり, 有効活用とはネットワークによりアイドル (ユーザが使っていない) 状態の資源を利用してバッチ的処理 (以後 “BG タスク”と呼ぶ) を行うことである.

2　関連研究と本研究の新規性

本研究で下記の分散環境を想定している. (1) 使用する計算資源群はヘテロジーニアスである, (2) 計算機群そのものを管理しておらず, 計算機にはプロセスを立ち上げるのみである, (3) システムの環境が動的である, (4) 構成する資源群は信頼性が低い, ここで (3), (4) はユーザのシステムへの参加, 脱退が頻繁に起こるからである. 上記の理由から整った環境である並列計算機, PC/ワークステーションクラスタ, 分散 OS とは想定している環境が大きく異なる. また, 連携処理専用のサーバを想定していないことや超大規模ネットワーク環境を想定しておらず GRID , Ninf などのグローバルコンピューティングとも想定している環境が異なる. 協調システムの規模は小規模 (数台) から中規模 (数十台) を想定している.

提案では特に以下の 4 目標を重要と考え, これらに特化したシステムを提案する. また, これらが本研究の新規性である. (1) “BG タスク”が “FG タスク”の作業を可能な限り妨げない, (2) 協調システムにユーザ認証機構があり認証ユーザレベルでの資源割り当てが可能である, (3) 協調システムの一部に障害が発生してもシステム全体がダウンしない高い耐障害性がある, (4) 動的な環境における適切な資源割当が可能である.

本稿では特に実装についてと, 目標 (4)“動的資源割当” について述べる. 目標 (1), (2) に関しては文献 [2] で, 目標 (1), (3) に関しては文献 [1] で述べている.

3　提案システム

提案手法は図 1の構造をしている. すなわち, 各計算機には “FG タスク”(図 1の “一般のプロセス”) と, 提案する協調システムのための “BG タスク”の実行環境 (図 1の “分散協調システムの 1 構成要素”, 以後 “BG タスクスペース”と呼ぶ) が存在する. 各計算機の “BG タスクスペース”同士は協調動作し, 全体として一つのプラットフォームを実現する.

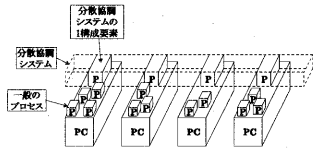

図 1: 提案システムの構造

3.1　BG タスクスペース

“BG タスクスペース”は “BG タスク”の実行環境である. クライアントは “BG タスクスペース”に対してタスクを投入することが可能であり, 必要ならば処理結果を受けとることが可能である.

各 “BG タスクスペース”は自ホストがアイドルであるかを監視しており, 計算機のユーザが席を外し計算機 (自ホスト) がアイドルになると他 “BG タスクスペース”より “BG タスク” 送信され, アイドルである間はその “BG タスク” を処理する. その計算機のユーザが戻り計算機がアイドルでなくなると “BG タスク” は他 “BG タスクスペース”へ移送される.

3.2　“BG タスク”の移送

実装を Java 言語で行ったため, 一般の方法ではプログラムの実行状態 (PCB など) を獲得することはできない[1]. 現段階の実装では “BG タスク” の移送は “BG タスク” のインスタンスの移送にとどまっている. すなわち, プログラムカウンタやスタック内の情報を持って移送させることができないのでスタック内のデータが空になった状態 (以下, “移送可能状態” と呼ぶ) でのみ移送が可能となる.

移送指示が送られると, “BG タスクスペース”内の移送フラグを立てる. 各 “BG タスク”は移送可能状態になるたびに “BG タスクスペース”の移送フラグを確認する. もし, 移送フラグが立っていなければ処理を続行し, フラグが立っていたら移送処理に移る. 基本的に移送先は “BG タスクスペース”が指定する[2]. このように, “BG タスクスペース”は移送フラグを指定するのみであり, フラグを立てた後にいつ “BG タスク”が移送されるかは “BG タスク”に依存することになる. 移送タスク数の詳細に関しては第 4章で後述する.

3.3　チェックポイント

チェックポイントは障害発生時における再開タスクの状態を更新することにより行う [1] が, 移送同様に移

[1] Native Code を用いる, トランスレータを作成しソースコードを変換するなどの方法を用いればこれを実現することが可能となる.

[2] “BG タスク”が自分の判断で移送することや, 移送先を決めることも可能である

送可能状態になったときにのみチェックポイントをとることが可能である.

3.4 状況の監視

計算機のアイドル状況の判断はユーザにより手動設定やスクリーンセーバにより判断などが実装されている. 計算機の負荷情報は OS から得る LoadAverage(Unix 系に限る) や "BG タスクスペース"が保持している "BG タスク"の数などにより判断する.

各 "BG タスクスペース"は定期的に (監視周期と呼ぶ) 自ホストの状況を調査する. また, 次節で述べる動的資源割当を実現するためにはこれらの計算機の状況を常時他 "BG タスクスペース"にアナウンスする必要がある. よって, 各 "BG タスクスペース"は定期的に (アナウンス周期と呼ぶ) 自ホストの負荷情報を他 "BG タスクスペース"にブロードキャストする. また, 自ホストの状態が著しく変化した場合はアナウンス周期以内でもアナウンスを行う.

3.5 動的資源割当

資源割当 / 負荷分散の方針として①高負荷状態を避ける, ②各 "BG タスクスペース"の負荷状態を均一にする, が考えられこれらを実装した. ここで ①"高負荷を避ける" がより重要とし②"均一化" は付加機能とした. なぜなら, 一般に過負荷状態では効率が著しく低下するため高負荷状態を避けることは重要であるが, 過負荷でない状態では効率は悪化せず負荷を均等化してもシステム全体のスループットは向上しない (向上が少ない) からである. 逆に②"均等化" は小数の "BG タスクスペース" に負荷が集中し結果として過負荷状態の "BG タスクスペース" を作成しないために有効である. ただし, 完全な均等化は重要でないため不必要な移送を行わないことが重要である.

自ホストの状態が過負荷と判断された場合は自 "BG タスクスペース"内のタスクをより負荷の低いホストの "BG タスクスペース"へと移送する. 過負荷の判断, 移送タスク量, 移送先は環境に依存する (第 4章参照).

4 考察と評価

提案システムを以下の環境において動作させ実験した. 負荷情報として LoadAverage およびタスク数を用る. 動作環境は SPARC Solaris, PC Solaris, FreeBSD, Windows 2000 の計算機を 100BaseTX Ethernet で接続した環境 (Windows 計算機では タスク数のみを使用). 過負荷の判断として過去 1 分の Load Average が 3.0 以上となった場合とした. 監視周期, アナウンス周期はともに 30 秒とした. また, 移送するタスクの数は①保持しているタスクの半分, と ②1 個, の 2 通りを実験した.

過去 1 分の Load Average を負荷として採用したため, "BG タスク"を他 "BG タスクスペース"に移送したことが負荷状態に反映されるには 1 分程度必要とされる. よって, 移送の判断の周期を短すぎる値 (30 秒未満) にすると負荷が下がる前に再度高負荷と判断して "BG タスク"を移送してしまうため過剰に "BG タスク"を移送してしまい, "BG タスク" が "BG タスクスペース" 間を振動する結果になった. 現実装では負荷の獲得は OS ごとに LoadAverage を獲得する機能を実装して獲得しているが, OS から LoadAverage を獲得する場合は 1 分より精度の高い負荷情報は獲得できない. また, LoadAverage を用いることにより (タスク数だけでは判断できない) そのホスト全体の負荷が獲得できた.

移送タスク数の判断は適切な負荷分散を行うのに非常に重要である. 現在の実装では, 過負荷判断の閾値を越えた場合の移送の方針として①1 回の判断につき保持タスク数の半分を他 "BG タスクスペース"に移送する, ②1 回の判断につき保持しているタスクの 1 個を他 "BG タスクスペース"へ移送する, の 2 通りを用意している. 方針①は保持しているタスク数に応じて変動するため一般的な解法と期待できるが, 移送判断間隔を十分に長くとらないと過制御状態に陥りタスクが "BG タスクスペース"間を往復する. 一般の利用においてタスク数は未定であり移送個数が固定である方針②は問題が多いと考えらるが, 実験した環境においてタスクの数は 10 以下が一般的であり (10 以上のタスクを処理してる場合は過負荷状態になり移送あるいは停止の必要が生じる)1 個ずつの移送でも問題がなく, 結果として①"半分" より良い結果が得られることが多いと言えた[3].

処理する "BG タスク"として頻繁に移送可能状態になるサンプルを用いて実験したところ移送要求を出してから移送処理が完了するまで 1 秒程度であり, "FG タスク"に弊害を出さずにアイドル資源を用いて "BG タスク" を処理することは実現できたと言える. しかし, 一般のアプリケーションを頻繁に移送可能状態になるように作成することは困難であることも確認された.

5 おわりに

本稿では, ネットワークに接続された計算機群のアイドル状態の計算資源を利用してバッチ的な処理を行う手法として, 特に第 2章であげた 4 目標に重点を置いたシステムを実装し, 評価した. 今後は, ①トランスレータの作成などを行い一般のアプリケーションを容易に移送できる環境を提供する, ②過負荷判断と移送タスク数を詳細に指定できるようにする (低度の過負荷状態の閾値とそのときの移送数や高度の過負荷状態の閾値とそのときの移送数などを指定できるよう改良を行う) , (3) 過負荷閾値 / 移送方針を動的に変動できるよう改良する. などを行っていく.

参考文献

[1] 山口 実靖, 相田 仁, 齊藤 忠夫, "協調バックグラウンドタスクスペースに関する検討", 第 8 回マルチメディア通信と分散処理ワークショップ論文集 pp.85-90, 2000 年 12 月

[2] 山口 実靖, 相田 仁, 齊藤 忠夫, "分散環境認証ベースによる資源割当", 第 61 回情報処理学会全国大会講演論文集 6J-08, 2000 年 10 月

[3] "半分" の場合は振動し収束しないことがある