分散環境認証ベースによる資源管理

6J-08　　山口 実靖 *　　相田 仁 **　　齊藤 忠夫 *

* 東京大学大学院工学系研究科　** 東京大学大学院新領域創成科学研究科

{sane,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. 本稿では “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” 手法を提案する. ローカルの作業とはワードプロセッサや表計算など対話的作業 (以後このフォアグラウンドのタスクを “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状態の資源を利用してバッチ的処理 (以後このバックグラウンドのタスクを “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[1], Ninf[2] 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.

本稿では特に以下の 3 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, さらに, (4) 動的な環境における適切な資源割当が可能である, も実現が推奨されると考える.

3　提案手法

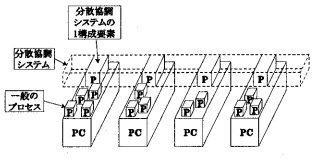

図 1: 提案システムの構造

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.

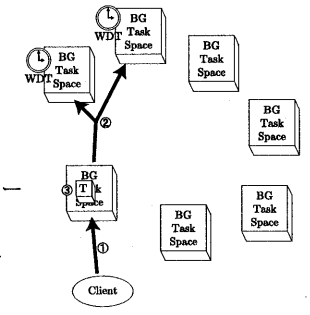

図 2: “BG タスク”投入

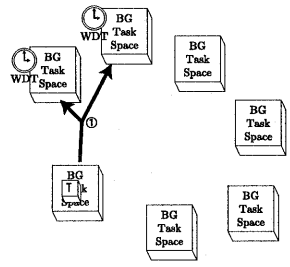

図 3: “BG タスク”処理

“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.

3.1　アカウンティング

協調システム全体で管理されるアカウント情報を構築し全ての処理要求に要求者情報をつける. “BG タスクスペース”は “現在までだれからの要求をどれだけ処理したか” と “現在誰からの処理をどれだけ処理しているか” を保持する. そして, “BG タスクスペース”は定期的にこれらの情報を交換し常に最新の情報に更新しておく. これにより, アカウントごとの累積処理量やアカウントごとのスループットを等しくすることが可能となる. 処理量の計測は陽に求まるとき[1]はその値を用いるが, 求まらないときはベンチマークにより現在の処理速度を推測しその値を用いる.

3.2　“BG タスク”の処理

本節では “BG タスク”の処理について述べる.

システムに対する “BG タスク”の要求は以下の様に行う (図 2参照). ①クライアントが “BG タスク”要求をいずれかの “BG タスクスペース”に送信する, ②その “BG タスク”のための Watch Dog タイマーを 1 つ以上起動する, ③その “BG タスク”を 1ヶ所以上の “BG タスクスペース”で開始する, Watch Dog タイマーはその “BG タスク”の処理に発生した障害を検出するためのものであり, 要求された “BG タスク”のコピーを保持している. “BG タスク”を起動するよりも先に Watch Dog タイマーを起動することにより, ②までの処理が成功す

[1]例えば, メール配送システムにおいて使用するネットワーク資源

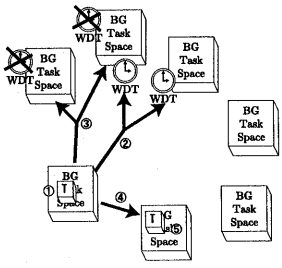

図 4: “BG タスク”移送

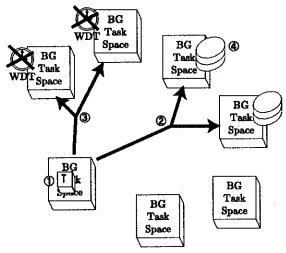

図 5: “BG タスク”終了

れば全 “BG タスク”の処理と全 Watch Dog タイマーが停止しない限り処理を正常に終わらせることが可能である. “BG タスク”処理中は定期的に Watch Dog タイマーをリセットする (図 3参照).

タスクの移送は以下の様に行う (図 4参照). ①“BG タスク”の移送要求が発生する, ②新しい Watch Dog タイマーを起動する, ③古い Watch Dog タイマーを停止する, ④別の “BG タスクスペース”に “BG タスク”を移動する, 同様にして②までの処理が成功すれば全 “BG タスク”の処理と全 Watch Dog タイマーが停止しない限り処理を正常に終わらせることが可能である.

タスクの終了処理は以下の様に行う (図 5参照). ①“BG タスク”が終了する, ②“BG タスク”の処理結果を結果格納ストレージに格納するなどのその “BG タスク”に定められた終了処理を行う, ③Watch Dog タイマーを停止する, ④結果格納ストレージに納められている結果は有限時間で削除される, Watch Dog タイマーを停止する前にストレージへの格納を試みているため, (全 Watch Dog タイマーに障害が発生しない限り) 冗長に複数回実行され複数回ストレージに伝えられることがあっても, 処理の格納がされないまま処理が停止してしまうことはない.

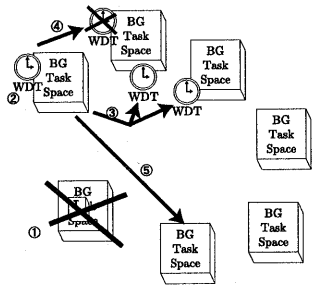

図 6: Watch Dog タイマーのタイムアウト

障害が発生したときは Watch Dog タイマーにより障害が検出される (図 6参照). ①障害が発生しその “BG タスク”を処理している “BG タスクスペース”がなくなる, ②Watch Dog タイマーがタイムアウトし障害を検出する, ③Watch Dog タイマーが再実行のための Watch Dog タイマーを起動する, ④古い Watch Dog タイマーを停止する, ⑤別の “BG タスクスペース”に再度 “BG タスク”を要求する, 前述のように Watch Dog タイマーは “BG タスク”の要求のコピーを保持している. 古い Watch Dog タイマーを停止する前に新しい Watch Dog タイマーを起動しているため, 処理が重複実行されてしまう可能性があるが, 処理が障害により停止してしまう危険性は低くなる. Watch Dog タイマーのタイムアウト時間を異なる値にしておくことにより重複実行の可能性を軽減できる.

3.3 “FG タスク”への影響

第 1章で述べたように, “BG タスク”が “FG タスク”の作業を妨げてはならない. 資源の解放としては以下の 4種類の “BG タスク”の処理方法が考えられる.

1. システム内の他のアイドル計算機に移送する
2. サスペンド状態にする
3. ローカル OS での優先度を下げる
4. 破棄する

方法 1は “BG タスク”にとっては理想的な方法であるが移送の処理があり速やかに資源を解放することができず, 厳密には “FG タスク”にとって理想的な方法ではない. 方法 2, 方法 3は速やかな移行が可能であることが予想されるが, “BG タスク”にとってのデメリットが多い. また, CPU 資源は解放してもメモリなどの資源は解放していないことになる. “FG タスク”の処理を妨げることが全く許されない場合は方法 4も必要であると考える. この場合, 破棄を行ってもその “BG タスク”の処理が正しく行われること, 破棄の被害を最小限に抑えることが必要となる (第 3.4節参照).

3.4 耐障害性

前述のように個々の “BG タスクスペース”は信頼性が低い. これらの環境で高い信頼性を実現するために以下のことが考慮されている. (1)Watch Dog タイマー, (2)“BG タスク”の移送, (3) チェックポイント. 上記の (1) は障害が発生しても正しく処理が行われることを実現し, (2), (3) は障害時の被害を抑える. ただし, 提案システムでは移送がチェックポイント処理を含んでおり, 同一 “BG タスクスペース”へ移送すればチェックポイントとなる.

4 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 (1) バックグラウンドのタスクがフォアグラウンドのタスクを妨げないこと, (2) アカウントベースの公平性があること, (3) 高い耐障害性があること, に重点を置いた手法を提案した. 今後は, 動的な環境変化に対応した適切な資源割り当ての実現, 提案手法の実装を行い実用による評価を進めて行く.

参考文献

[1] Ian Foster, Carl Kesselman: The GRID Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999

[2] http://ninf.etl.go.jp/