5S−9

ソフトウェア分散開発環境における負荷分散プログラムの実現*

二上俊嗣　三浦史光†
NTTソフトウェア研究所‡

1　はじめに

最近、ソフトウェア開発環境としてワークステーション（WS）を利用した分散開発環境が広まりつつある。これは、高い処理能力を持つCPUを内蔵した多数のWSを用いて、自身のディスク上またはファイルサーバ上にあるソースファイルをアクセスしながら、組織的に大規模なソフトウェア開発を行う環境である。

ここで、WSの処理能力は単なるデータ処理に用いられるだけではなく、エディタ、ウィンドウシステム、ネットワーク通信機能等にも用いられている。そのため、一時的な高負荷にも十分耐えられるように、かなり能力に余裕を持たせているのが一般的である。

しかし、このような余剰能力はWS台数が増大するにつれて、無視しえないものとなってくる。一時的に大きな処理能力を必要とする場合には、これを積極的に利用することで、環境を大きく向上することができる。

そこで、本論文ではこのようなソフトウェア分散開発環境における負荷分散方式に関して検討を行い、その試作プログラムに関して報告する。

2　現状の問題点と改善の方針

ソフトウェア開発において特にCPU資源を必要とする工程の1つに、全ソースファイルの一括コンパイル・リンクがある。現在、このような作業の進行を自動的に管理するためにmakeが使用されている。

従来の開発環境ではmakeはコンパイル対象ファイルを選び出し、パラメータを組み立て、それを順次実行していた。この時、多くのコンパイルジョブ(cc)が発生し、その大多数は他のコンパイルジョブと独立に実行出来るものであったが、現状はこれを1つ1つ逐次的に処理する方法をとっている。

一方、分散開発環境において効果的な負荷分散を行うためには、makeは実行可能なコマンド列を作成し、適当なWSを選択し実行を指示した後、その実行完了を待たずに、次のステップへ進むことが要求される。

ここで必要となる機能は、

1. ジョブ割付機能: 負荷の軽いWSを選び出し、ジョブを割り付ける
2. 実行順序制御機能: 並列に行えるジョブは並列に実行させ、並列に行えないジョブは、他のジョブの完了を待たせる

の2つにまとめることができる。

ここでは従来の環境との互換性を確保し、移行を容易にするために、makeの改造は最小限にし、必要な機能は外付けする方法で試作を行うこととした。試作するコマンドは単独でも利用出来るようにし、make以外のコマンドとの連携にも対応できるよう考慮した。(図1)

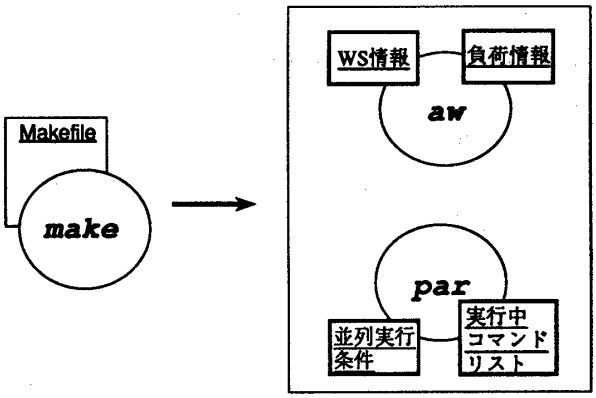

図1：負荷分散プログラムの構成

3　ジョブ割付コマンド

ジョブ割付コマンドは、以下の機能で構成されている。

1. 負荷データ測定機能: 適当なタイミングで利用可能な全WSの負荷状況を測定する
2. 負荷データ補正機能: 測定した負荷データに、各WSごとのデータ（CPU種別、ローカルディスクの有無、登載メモリ量等）を重ね合わせて補正する

*A Load Balancing Program for Software Development Environment
†Toshitsugu Futagami, Fumiaki Miura
‡NTT Software Laboratories

3. 最軽負荷 WS 選択機能: 最も負荷の軽い WS の候補リストを常時準備する
4. 最適 WS 選択機能: ジョブの実行条件 (ツール及びデータファイルの利用不可等) に合った WS を選択する
5. 遠隔ジョブ実行機能: 選択された WS に対し、コマンド、パラメータ、環境変数、標準入出力を渡し、実行を依頼する

このような機能を持つジョブ割付コマンドを作成し、aw(anywhere) と名付けた。

4 実行順序制御コマンド

実行順序制御コマンドは、以下の機能で構成されている。

1. 並列化判断機能: 実行すべきコマンドが、他のコマンドと同時に実行可能かを判断する
2. 実行状況モニター機能: それ以前に起動されたコマンドのリストを保持し、それの実行完了状態をモニターする
3. 並列実行機能: 並列実行可能なコマンドをバックグラウンドジョブとして実行し、逐次的にしか実行できないコマンドは、他のコマンドの実行終了を待って開始する

これらの機能を持つ実行順序制御コマンドを作成し、par(parallel) と名付けた。par は、特に指定しなければ、連続する cc だけが並列に実行できるとする方針を採用する。この方針は悲観的であり、すべての並列性を検出できるわけではないが、コンパイルに適用する限り、実用上問題にならない。通常の使い方では、正当性 (fairness) を破壊することはない。この方針はユーザによって変更することが可能である。(図 2)

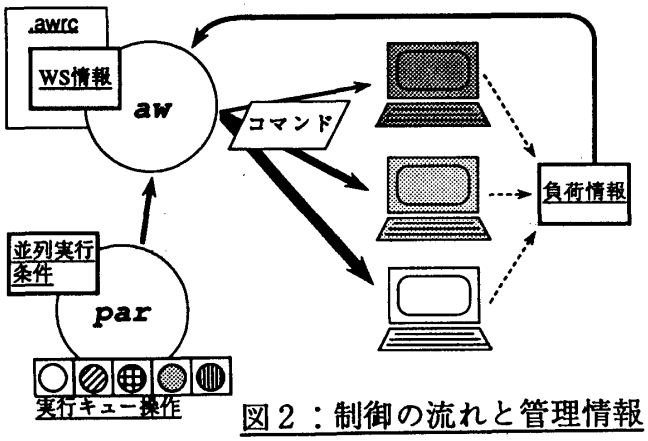

図 2 : 制御の流れと管理情報

5 make の改造

make が作り出すコマンドを並列的に他の WS で実行するために、make 内部のプロセス生成部を修正し、全てのコマンド及びパラメータを par に渡すようにした。

makefile に関しては、単純な構成の場合には修正の必要がなかった。

6 評価

現在、par,aw の評価用試作版を作成し、この方式の評価を行っている。14 個のソースファイル (計 7000 行) を 3 台の WS(Sun-3/470 1 台,Sun-3/280 2 台) を使用して make した結果では、従来方式の make に比べて約 2 分の 1 の時間で全ジョブを実行した。リモートファイルアクセス等の通信処理オーバヘッドを考えると、十分妥当な速度と考えられる。

7 さいごに

本論文では、ソフトウェア分散開発環境における負荷分散の一方式を提案し実現した。この方式の長所として以下の項目があげられる。

1. 簡単な機構で負荷分散を実現できる
2. コマンド自体のオーバヘッドが比較的小さい
3. 既存のプログラムとの連携が容易にできる
4. OS の一般的な機能だけを利用しているので、他機種への移植が容易である

また、短所としては、以下の項目があげられる。

1. 並列実行における正当性が必ずしも保証されない。
2. 負荷分散の単位がプロセスであるため、make 以外の利用分野が見つけにくい

今後、分散開発環境には多種類の WS が数多く導入されるであろうが、この方式はそれらの持つ能力を有効に活用するための一手段となると考えられる。

最後にこれからの課題として、

1. 並列実行性の自動抽出法の検討
2. par,aw 個別の効果の評価
3. 多様な負荷状況におけるツールの性能評価

があげられる。そして、これらをもとにしてより効果的な負荷分散方式を検討していく予定である。