近況報告

遠藤 敏夫

1999 年 3 月 3 日

並列 GC とソフトウェア分散共有メモリの実装を並行して進めています。

1　並列 GC

共有メモリ用並列 GC を 98 年 10 月に公開してからも、デバグ、機能追加を何度か行っている。外注さん[1]に手伝ってもらって、「MicroSoft が脅威に思っているらしい Linux」への移植とか、 64 ビットコンパイラへの対応を行った。そろそろドキュメントを更新して公式バージョンアップを行いたい[2]。

今後の予定の一つは、将来 SMP クラスタが広まるだろう、ということで山本君の分散 GC と組み合わせること。クラスタ内の処理は共有メモリ用 GC が行い、クラスタをまたがる処理を分散 GC が行う。

2　ソフトウェア分散共有メモリ

秋頃から、ソフトウェア分散共有メモリシステムを作って遊んでいる。

ソフトウェア DSM 単体で見たときに、性能面で先発の研究に勝とうとはあまり考えていなくて、目標としては共有メモリで今動いている処理系が気楽に分散メモリで動くようになればいい、と考えている。

$\boxed{\text{プラットフォームの性質が変わる} \rightarrow \text{研究のネタが増える}}$

既存のソフトウェア DSM システムをアクセスミスの捕まえ方に注目して大まかに分類すると以下のようになる。

1. 仮想メモリを使用。 IVY, TradeMarks などの、老舗
2. コンパイラがミスチェックコードを出力。平木研 DSM, 脇田さん DSM など。 Emerald などの分散オブジェクトシステムもここに分類することができる
3. アセンブラコードを処理してミスチェックコードを挿入。 Shasta

3 番目のアセンブラコードを処理する方針をとっている。それぞれどんな長所短所があるかは、後藤君の話とほとんど同じ。

本システムは 2 つのモジュールからなる。ポストプロセッサとランタイムライブラリである。

---

[1] その正体は多くの人が知っている通り

[2] MicroSoft ではないが、頻繁にバージョンアップを繰り返すという戦略も、宣伝の機会を増やすという意味では「あり」かなぁ

2.1 ランタイムライブラリ

読み込みミス、書き込みミスが発生したとき、ライブラリ中のアクセスミスルーチンが呼ばれる。その中で通信を行って、最新のメモリ内容を入手する。今のところ、一貫性制御単位 64 バイトの sequential consistency を実装済みである。

しかし性能的には、もっとプロトコルを緩和し、せめて non-blocking write ができないと話にならなさそうな状況ではある。

2.2 ポストプロセッサ

SPARC のアセンブラコードを解析して、 read/write 命令にミスチェック命令を挿入するプログラムを作成し、動いている[3]。

各関数毎に使用レジスタ / 未使用レジスタを探して、未使用レジスタをミスチェックに使う。もっと細かいフロー解析はまだ行っていない。しかし、どうしてもレジスタが足りない場合は頻発するので、そのときは現在のレジスタの値をスタックの上に積んで待避しておく。

2.3 現状

Ultra Enterprise 上で MPICH を用いて実装を行っている[4]。

ミスチェック +polling 挿入、マルチスレッド対応した Sequential consistency の実装がほぼ終わっていて、並列 GC を使った簡単なプログラムが動き出している... という状況である。

まず逐次では、ポストプロセスしない場合に比べて 10 倍遅い速度で動く (挿入コードのオーバヘッド 900%。ちなみに Shasta は 35% 以下だそうな)。

2 プロセスで並列に動かすと、生の Ultra Enterprise 上で動かすのに比べて 1000 倍遅い。将来速くなる一方だと思えば気は楽か？

2.4 メモリモデルについて

現在は性能の低い sequential consistency を実装してあるが、将来は緩和したモデルを実装したい。その際、有名な release consistency そのままでは少し不満があるので修正したい。その不満とは、メモリ変更が伝播するには必ず lock/unlock が必要だということである。

ところが、共有メモリ用に書かれたプログラムは lock/unlock を使わない同期を行うこともある。

- p を共有領域へのポインタとしたとき、 while (*p == 0) ; で待つ
- Herlihy etc. の lock free アルゴリズム

こういう方法を使っているプログラムが、大きな変更なしで動くようにしたい。

[3]やっていることは後藤君のポストプロセッサのサブセットなのでそれを使わせてもらってもいいのだが、勉強のために自前でやっている

[4]共有メモリ上で分散メモリをエミュレートし、またその上で共有メモリをエミュレートしていることになる