OpenMosix のすすめ

松　本　正　和

Ⅰ．はじめに

大学本部の監査員が科研費購入品のリストを見て，いぶかしげに尋ねました。

監査員「先生，なんでこんなにたくさんパソコンばっかり買ってるんですか?!」
私「計算に使うからです」
監査員「本当に，こんなにたくさん，使えるんですか?」
私「・・・」

理論化学研究室の研究費による購入品リストには，パソコンがずらっと並びます。それも，ラックマウントサーバやクラスタといった名前ではなく，HP や Gateway の，事務所や家庭で使うようなパソコンばかり。パソコンはマウスとキーボードで操作するもの，と思っている人が見たら，研究室構成員の数の 10 倍もの台数のパソコンを本当に買っているのか，どうやって使っているのか，実は架空伝票でも作っているのでは，あるいは転売でもしているのではないか，と心配になる気持もわかります。

でも，実際にこれらのパソコンは，倉庫のような部屋の片隅の棚に並べられ，たった数人のユーザによって，今もその性能をフルに発揮して研究に活用されています。我々にとっては，パソコンもサーバも「計算機」にすぎません。DVD ドライブもグラフィックカードも USB も，さらにはキーボードもマウスも必要なく，速い CPU と大きなメモリ，速いネットワークカードだけが，計算機に必要な要素です。

理論化学研究室では，タンパク質などの生体分子や水といった，身近な物質の物性や反応を主に計算機シミュレーションによって研究しています。実験では観察が極めて困難な，超短時間かつ非常に小さな空間尺度でおこる現象も，計算機シミュレーションなら再現することができます。本研究室には，このような計算を行うための計算機が 400 CPU 以上あります。

最近の計算機シミュレーションを援用した研究の動向を見ると，計算規模は大きくなる一方ですが，最近ではパソコンの性能が非常によくなり，安いパソコンを買ってきてもそこそこの計算ができるので，これをたくさん並べた研究室専用コンピュータを作って大規模並列シミュレーションを行う，ということも可能です。

ただし，100 台を越える大きな計算機クラスタを研究室で運用するとなると，どうやって構築するか，どうやって維持管理するか，という点が問題になります。

Ⅱ．ハードウェアについて

構築のノウハウについて，ここでは詳しく述べませんが，100 台規模になると，箱から出してケーブルをつなぐだけでも重労働なので，それ以外に台数に比例して行わなければならない作業をいかに減らすか，ということが一番重要になります。また，管理コストもあまりかけたくないので，できるだけメンテナンスフリーになるようにあらかじめよく設計を練っておく必要があります。本研究室の場合，壊れやすいハードディスクの代わりにフラッシュメモリ（システムを入れるだけなら 512M バイトあれば十分）を使ったり，ディスクレスにしたりといった工夫をしています。

あと，壊れることを前提にして，パソコンやハブが数台壊れても，問題なく動くようにすると，管理者の手間が大幅に減ります。といっても，銀行のシステムのような堅牢さを求めるわけではありません。安いパーツを使うなら，スペアパーツを余分にあらかじめ入手しておくとか，計算機の台数を（止まることを想定して）あらかじめ多めにしておくといったことが効果的です。計算機センターのように，計算機を安定稼動させることに責任を負っているわけではないので，一部が止まることもあると割りきるとともに，一部の障害が全体に波及しないようにすることが，作業量の軽減につながります。(計算機の管理に時間を取られて研究する時間がなくなってしまっては本末転倒です。)

Ⅲ．ソフトウェアについて

100 台規模のクラスタを運用する場合，バッチキューシステムを導入するのが一般的だと思います。ユーザが計算したい内容の要望書（シェルスクリプト）を作り，それをバッチキューに投入すると，システムが空いている CPU をさがしてリモートプロセスを投入し，結果を知らせてくれる仕組みです。

私はバッチキューシステムにあまり良い思い出がありません。計算機センターの汎用機のバッチキューにジョブを投入し，長い長い間待たされたあと，いざ走りはじめたらスクリプトの書き損じですぐ止まってしまい，エラーを修正してまた再投入して待たされ，・・・この冊子をお読みの方であればこんなご経験があると思います。バッチキューは，高価な計算機資源を最大利用するための，いわば管理者のための仕掛けであり，ユーザのための仕掛けではないからです。私はシミュレーションで得られたデータの解析のためのプログラムをしょっちゅう書き，デバッグし，試し，実行しています。コンソールに向きあっている時間の多くはこのような作業に費しているので，速い計算機を使ってこれらの作業のサイクルを短縮できればと思うのですが，バッチキューはそもそもこのようなワークフローにフィットしませんし，バッチキューを使ってもコンパイル作業や試行の時間を短縮することはできません。むしろ，キューでの順番待ち時間のために，体感時間は遅くなりかねません。ユーザの立場からすれば，ターンアラウンドタイム（ジョブを投入してから結果が得られるまでの時間）が短かく，計算機クラスタの処理能力をシームレスに利用できること（コンソールで実行する雑多なコマンドやスクリプト，あるいはウィンドウ

システム上で動くアプリケーションでも，計算機クラスタの性能の恩恵を受けられること）が最も有用なのですが，バッチキューの仕組みはあきらかにこれに逆行しています。

せっかく自分たちのためのクラスタを作るのですから，計算機を使うのにいちいち要望書を書かなくても，手許のコンピュータを使うのとできるだけ同じ感覚で計算機クラスタを使えるようにできないだろうか，と私は常々考えてきました。コンピュータ上で行う作業のすべてがバッチキューに向いているわけではありません。例えば数千ファイルからなるプログラムをコンパイルするのに，バッチキューを使えないわけではありませんが，代償としてコンパイルのスケジューリングを全部自分で行わなければならなくなります。数万ファイルになるとこれは悪夢です。自分の使っている計算機がクラスタなのかどうかを意識せずに，どんなアプリケーションを走らせても負荷分散が自動的に行われるような仕組みはないものでしょうか？

Ⅳ．OpenMosix

インターネットであちこち探した結果，OpenMosix というシステムに出会いました。OpenMosix は，ネットワークでつながった複数のコンピュータ間で負荷を均衡させる仕組みです。バッチキューと違い，OpenMosix は Linux カーネルそのものを拡張し，負荷の重い計算機のプロセスを負荷の軽い計算機にネットワーク経由で動的に受け渡すことで負荷を均衡させます。

現在研究室で拡張中の計算機クラスタでは，各計算機はサーバから OpenMosix 拡張機能を組みこんだ Linux カーネルを読みこみ，network boot 機能を利用して起動します。network boot に対応しているパソコンを買ってきて，電源とネットワークケーブルをつなぎ，スイッチを入れて起動時に F12 キー（network boot キー）を押してやるだけで，直ちに計算機クラスタに組みこまれるようにしてあります。クラスタに組み込まれた計算機は，OpenMosix 拡張機能により，計算機の処理能力を自動的に互いに「融通」（migrate）します。つまり，ある計算機に重いプロセスをたくさん投入すると，それらは自動的に空いている計算機に移され，負荷の均衡が図られます。

処理能力の融通は，プロセスの負荷と計算機の処理能力にあわせて動的に行われます。モニタリングツールを使えば，融通されたプロセスがいろんな計算機ノードの間を放浪しているのを見ることができますが，プログラムを走らせているユーザは，実際にどの計算機で処理が行なわれているかを全く意識する必要がありません。つまり，OpenMosix で結合された計算機は，ユーザからは SMP（対称型マルチプロセッサ）機のように見えます。実際，CPU を 100％使用するプロセスを 100 個，100 台編成の OpenMosix クラスタの 1 ノードで走らせた場合，見掛け上の CPU の使用率の合計が 10,000％になります。

バッチシステムのように，プログラムを走らせるのにいちいちスクリプトを書くといった手間もなく，単にバックグラウンドプロセスとして走らせるだけで自動的に負荷分散が行われるのです。make コマンドを使っている人は-j オプションを追加するだけで直ちに負荷分散機能を利用できます。OpenMosix はカーネルの拡張機能なので，OpenMosix 上でプログラムを走らせる

ために再コンパイルする必要もありません。また，各計算機の処理能力が異なる場合には，処理能力の高い計算機に自動的に優先的に処理が融通されます。

例えば，研究室で学生が使っている端末全部に OpenMosix 拡張機能を入れてしまえば，学生が端末を使っていない間は，その処理能力を計算用に融通することができます。学生が端末で作業を始めると，自動的にその端末へのプロセス割り当ての優先順位が下がるので，そこで走っていたプロセスは別の計算機へ融通されます。

OpenMosix は大規模並列計算よりもむしろ細かいプログラムをたくさん走らせるような処理に向いています。しかし，並列プロセス間の通信がそれほど多くなければ，OpenMosix 上で並列計算を行うことも可能です。例えば，MPI を使う場合であれば，プログラムを走らせる計算機に 100 CPU 搭載されているという風に設定しておけば，実際の並列プロセスがどの CPU で走るかは OpenMosix にまかせてしまうことができます。

このように，OpenMosix は動的に負荷分散を行い，しかもユーザが意識せずに透過的に分散処理ができるという点で画期的な仕組みと言えます。ただし，バッチキューのように，ジョブ数を制限したりユーザごとの優先度を設定したりする機能は OpenMosix 自体にはないので，見知らぬ人同士が使う計算機に使うと，ジョブを入れた者勝ちになってしまう危険はありますが，研究室のような規模で使うには非常に便利だと思います。ぜひ一度，使ってみてください。

OpenMosix を体験する一番手っ取り早い方法は，ClusterKnoppix を使うことでしょう。ClusterKnoppix は，CD から Linux をブートさせる Knoppix という仕組みの上に OpenMosix を組み込んだものです。2 台のパソコンを同じネットワークに繋いで ClusterKnoppix で立ちあげ，いろんなプログラムを走らせてみれば，プロセスが融通される様子がわかると思います。

V．関連リンク

[1] 理論化学研究室 Wiki
http://www2.chem.nagoya-u.ac.jp/~og/wiki/wiki.cgi

[2] OpenMosix
http://openmosix.sourceforge.net/

[3] OpenMosix HowTo
http://www.linux.or.jp/JF/JFdocs/openMosix-HOWTO/

[4] ClusterBuild
http://www2.chem.nagoya-u.ac.jp/~og/wiki/wiki.cgi/ClusterBuild

[5] ClusterKnoppix
http://clusterknoppix.sw.be/

（まつもと　まさかず：名古屋大学物質科学国際研究センター）