5ZA-04 I/O要求の遅延を低減するプロセス・スケジューリング法

小堺 康之　中山 泰一
電気通信大学 情報工学科

1 はじめに

近年、CPUの速度は飛躍的に向上している。このことにより、ますますI/O機器との処理速度の差が開いてきている。

このような状況に対処するため、従来のOSではプロセスのスリープとウェイクアップによる同期機構やI/O処理の非同期化などの技術が用られてきた。一方、時分割を基本とした従来のプロセス・スケジューリング法はなんらI/Oの考慮がなされていない。そこで我々は、特にI/O要求の遅れというものに着目し、これを低減するプロセス・スケジューリング法を提案する。具体的には、プロセスがどの辺りを実行中にI/O要求を出すのかを、実行時にプロファイリングする。同時に、その情報を基にスケジューリングを行う。このスケジューリング法によりI/O要求の遅れを低減できる。特に、CPU処理とネットワークのI/Oがあるプログラムでは、マシン間の応答時間が短くなることが期待できる。

2 I/O要求の遅れ

ネットワークのI/O要求の遅れの例を図1に示す。マシン1のA、マシン2のDはCPU時間のみを消費するプロセスである。また、B,Cはネットワーク I/Oの処理も行ない、お互いに協調して動作するプロセスである。矢印はプロセス切替えの流れを示す。プロセスBは、(1)において、あと少しでI/O要求を出そうとしているのにも関わらず、強制的に再スケジューリングされている。これにより、再びこれらのプロセスがディスパッチされるまでI/Oの要求が遅れている。これがI/O要求の遅れである。I/O要求の遅れにより、プロセスCから見ると、プロセスBからの応答時間が悪くなっている。

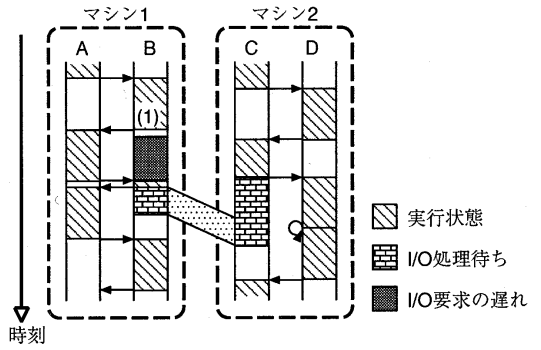

図 1 I/O要求の遅れ

A scheduling policy for reducing the latency of I/O requests
Yasuyuki KOZAKAI and Yasuichi NAKAYAMA
Department of Computer Science, The University of Electro-Communications
E-mail: kozaka-y@igo.cs.uec.ac.jp

3 プロファイリングを用いたプロセス・スケジューリング

I/O要求の遅れは、あと少しでI/O要求を出そうとしているプロセスがタイムスライスを使い果たすことが原因である。そこで、このタイムスライスを伸ばすことで解決できる。

しかし、これを実現するためには、いつI/O要求が起こるかを知らなければならない。そこで我々は、以下の2つをタイム・クオンタム毎にプロファイリングするようにした。

1. ユーザ・プロセスがどの辺りを実行しているのか。
2. そのときI/O要求が起こるかどうか。

1. は、プロセスのアドレス空間をいくつかのブロックに分け、そのブロック番号で表す。どのブロックを実行中かは、プロセスのプログラムカウンタから得る。2. は、ブロック番号ををキーとしたハッシュ表に次の2つの情報を保持し、これらを基に判断する。

- `intr_count`：
  ブロック内を実行していた時に起きたタイマ割り込みの回数。
- `io_count`：
  タイマ割り込み間に少なくとも1回、I/O要求が起こるとインクリメントされる。

(`io_count`/`intr_count`)は、ブロック内を実行中にそのタイムクオンタム内でI/O要求が起こる確率である。I/O要求が起こるかどうかは、この値で判断可能となる。

我々は、このハッシュ表を用いた次のスケジューリング・アルゴリズムをLinux 2.3.16のカーネルに追加した。

- タイマ割り込み時：
  タイマ割り込み時のプログラムカウンタを基に、ハッシュ表からエントリを得てその参照を保持しておく。もしハッシュ表に無ければ新たなエントリを作る。そしてそのエントリの`intr_count`をインクリメントする。

- I/O要求時：
  直前のタイマ割り込みが起こってから初めてのI/O要求ならば、io_countをインクリメントする。
- タイムスライスの減少時：
  タイムスライスが0になった場合、

$$\texttt{io\_count/intr\_count} > \texttt{th\_extend}$$

ならそのプロセスのタイムスライスを1クオンタムだけ延ばす。th_extendはしきい値である。

4 実験

実験に用いた計算機は、サーバがPentiumIII Xeon 500を4台持ち、メモリは256MBを搭載している。但し、CPUは1つのみ動作させた。クライアントはCeleron 300A, メモリは64MBを搭載している。サーバプログラムは5つの子プロセスを起動し、各々はクライアントからの要求を待ち受ける。クライアントからファイル名を受けとると、そのファイルに記載されている数字を昇順にソートし、クライアントに送り返す。

クライアントプログラムは、初めの5つの要求を1秒毎にサーバに出す。その後は結果が返ってくる度に新たな要求を出す。ソートの対象となるファイルのサイズは1.5MBである。

図2はブロックのサイズを変更したときの応答時間の推移である。th_extendは0.5に固定している。初めの4分間はハッシュ表の情報があまり正確ではないため、Linux 2.3.16のオリジナルのスケジューラに比べ応答時間が悪くなっている。しかし、その後は情報の精度が上がり、オリジナルより応答時間が良くなっている。

図3はth_extendを変更したときの応答時間の推移である。ブロックサイズは2^{10}Byteである。図2と同様のことがいえるが、th_extendが0.9のときは、0.5のときよりも応答時間が悪くなっている。これは、th_extendが大きすぎるため、I/O要求の遅れを見逃すことが多くなるためと考えられる。

なお、ブロックのサイズが2^{10}Byte,th_extendが0.5のとき、予測的中率は約40%であった。また、このとき、ハッシュ表のエントリを更新するコストが21.7μsであった。このことから、ハッシュ表の導入によるコストは十分小さいことが分かる。

5 関連研究

最近、I/Oのボトルネックが重要視されてきている。主な解決法はかなり低レベルな並列化技術を用いるものや、I/O操作の並列化などが挙げられる[1]。

また、谷口によって「プログラム指向スケジュール(POS:Program Oriented Schedule)」が提案されている[2][3]。POSでは、プロセスの振舞いを把握、保存し、

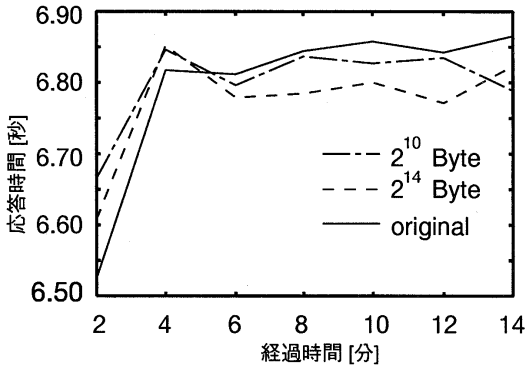

図 2 ex_addrサイズに対する応答時間

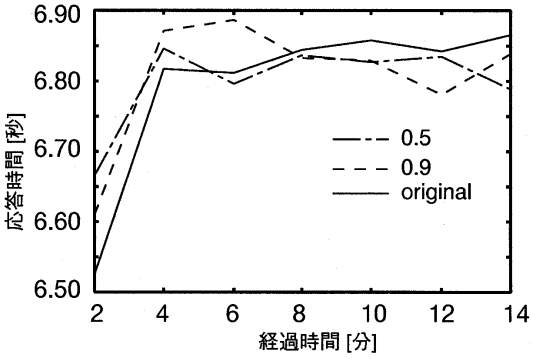

図 3 th_extendに対する応答時間

その情報を基にプロセスの実行制御法を変更していく。

このスケジューリング法では、プログラムの振舞いはプログラム終了後も2次記憶に保存されるため、何回もプログラムを実行することによって情報の精度を上げることができる。これに対して本論文で提案するスケジュール法では2次記憶には保存しない。しかし長時間動作し続けるサーバプログラムなどを対象とすれば、十分予測の精度を良くできると考えられる。

6 おわりに

I/O要求の遅れを低減するプロセス・スケジューリング法を提案した。また実験によってその有効性を示した。今後は、より情報の精度を上げるため、ブロックのサイズをエントリ毎に変更できるようにする予定である。

参考文献

[1] Ravi Jain, Kiran Somalwar, John Werth, and J.C.Browne: Heuristics for Scheduling I/O Operations,*IEEE Trans. on Parallel and Distributed Systems,* Vol 8, No.3, pp.310-320(1997).

[2] 谷口 秀夫: POS:プログラム指向スケジュールの提案, 情報処理学会コンピュータ・システムシンポジウム論文集, Vol.96, No.7, pp.123-130(1996).

[3] 谷口 秀夫: 入出力要求の流れの学習による入出力の効率化, 情報処理学会コンピュータ・システムシンポジウム論文集, Vol.97, No.8, pp.141-148(1997).