Vol. 36 No. 8　　情　報　処　理　　Aug. 1995

解　説

2.1 並列OSの研究動向†

福　田　晃††

1. はじめに

マルチプロセッサハードウェアとユーザ（またはコンパイラ）の間に位置する並列OS（マルチプロセッサを対象としたOS）[1)~3)]の研究課題および研究動向を概説する．並列OSは古くて新しい分野であり，研究が徐々に進んではきているもののまだ熟してはいない．本稿で述べるすべての内容について程度の差こそあれ，現在研究・実証段階である．本稿の内容は，文献1）と重ならないようにした．文献1)と併せ読んでいただければ幸いである．

2. 並列OSの課題とソフトウェア技術

2.1 スケジューリング[4)]

マルチプロセッサの主な存在意義はプログラムの高速実行であることを考えると，スケジューリング問題が重要となる．

（1） 考慮すべきオーバヘッド

1） 同期にともなうオーバヘッド

排他制御のためのスピンロックでは，ロックを獲得しているスレッドがサスペンドされると，繁忙待機（busy waiting）しているスレッドはプロセッサを浪費することになる．また，同期関係が生産者/消費者の場合では，生産者がサスペンドされて消費すべきものがない場合には消費者はプロセッサを浪費することになる（逆の場合も同様に浪費となる）．さらに，この問題はバリア同期でも生じる．

2） コンテクストスイッチングにともなうオーバヘッド

コンテクストスイッチングは，レジスタ情報の退避およびローディング，さらには，プロセスがUNIXのように仮想アドレス空間を含む場合には，アドレス空間の切換えのオーバヘッドが生じる．アドレス空間の切換えは，ハードウェアにもよるが，TLB（Translation-Lookaside Buffer）およびキャッシュの無効化を必要とし，さらに，これらのミスヒットが生じやすい．

3） メモリアクセスのオーバヘッド

キャッシュミスヒットに加えて，分散共有メモリアーキテクチャでは，リモートメモリへのアクセスはオーバヘッドとなる．

上記のオーバヘッドを軽減するスケジューリングが必要となる．

（2） スケジューリングの分類

マルチプロセッサ上のプログラミング環境としてシングルプログラミング/マルチプログラミング環境の2つがあり，どちらの環境を考えるかは重要な問題である．望ましいことではないが，俗に言うコンパイラ研究者は前者，OS研究者は後者を想定することが多い．ここでは，OSの存在意義がよく現れるマルチプログラミング（プロセス）環境を考える．スケジューリングは次の2つの次元から大きく分類できる．

1） 計算機資源から見た分類

時分割（Time-multiplexing）方式と空間分割（Space-multiplexing）方式の2つに大別できる．時分割方式は，基本的には，システム全体のプロセッサ群を1つのプロセスに割り当て，プロセス内のスレッドを同時実行し，タイムスライスを使い切ったら次のプロセスに割り当てるものである．一方，空間分割方式は，プロセッサ群をグループに分割して，グループ対応にプロセスを割り当てるものである．上記は，大きな分類であり，その折衷方式も存在する．時分割方式は，同期のオーバヘッドを軽減できる可能性が高いが，スケジューラボトルネックおよびハードウェアキ

† A Survey of Multiprocessor Operating Systems by Akira FUKUDA (Nara Institute of Science and Technology).
†† 奈良先端科学技術大学院大学情報科学研究科

ャッシュが活用しにくいという問題がある．空間分割方式はその逆となり，また，カーネル空間とユーザ空間でスケジューラを分離でき，ユーザプログラムごとにスケジューリング方策を選択できるので，現在研究が活性化されている．

2） 割当て決定時期から見た分類

プログラム実行時にのみプロセッサ割当てを決定する静的方式と，実行中にも決定する動的方式の2つがある．スレッドの粒度にもよるが，割当てオーバヘッドを考慮しても動的方式が優れているという報告があり，盛んに研究されている．

また，その他に協調するスレッド群を考慮したスケジューリングかどうかによって分類したものもある[4]．

（3） 提案されているスケジューリング方策

同期にともなうオーバヘッドの軽減を目的としたものにHandoffスケジューリング，および典型的な時分割方式であるCoscheduling（Gang scheduling），キャッシュを活用するCache affinityスケジューリング，がある．また，同期およびコンテクストスイッチングのオーバヘッドを軽減する目的でスケジュール対象の仮想プロセッサ数を制御して物理プロセッサ数と大体等しくするProcess controlスケジューリングなどがある．さらに，空間分割方式の1つとして2レベルスケジューリングがある．まだ確定したスケジューリング方策はなく，ハードウェアアーキテクチャと絡んで，現在研究段階である．

2.2 同　期[5]

並列OSの実装方法にもよるが，OSコードを全プロセッサでも実行できるようにするには，同期機構が必要となる．同期アルゴリズムを考える場合，同期に参加するスレッド数が増加しても性能が低下しないこと（スケーラビリティ），軽い同期処理が重要となる．最も必要な同期機構は，OS内部の共有データ構造への排他制御のためのロック機構である．スピンロックについてはハードウェアキャッシュなどを意識したものなど，種々のアルゴリズムが提案されている．また，複数のリードアクセスを許し，ライトアクセスを排他制御するリード/ライトロック機構も必要である．さらに，ある時間スピンした後サスペンドするロック機構もある．

2.3 メモリ管理[6]

分散共有メモリアーキテクチャでは，メモリアクセスの不均一性を活用するため，ローカルメモリをキャッシュとして使う（動的）ページ割当て技法が研究されている．コヒーレンス制御方式としては，ハードウェアキャッシュにおけるものを基本的に流用して，ソフトウェアで実現することになる．ハードウェアキャッシュとの最大の違いは，制御単位がページと大きいため，仮想ページの移動，複製のコストが大きいことである．さらに，コヒーレンス制御にともない，仮想ページ自体の処理以外に，ページ表の書換え，TLBの無効化の操作が必要となる．したがって，ハードウェアキャッシュで採用されているコヒーレンス制御方式の流用の他に，頻繁なページ移動（または複製）を防止する（ページの凍結）方策を考える必要がある．さらに，割当て方策を見直す（ページの解凍）方策も必要となる（デーモンを走らせることになる）．また，ページ移動か複写かの決定には，当該仮想ページへのアクセス履歴を必要とし，この履歴はハードウェアが通常サポートしている参照ビット，修正ビットでは履歴が不十分で，これらを用いてソフトウェアで実現・管理しなければならない．また，どの時点でコヒーレンスを保証するかは，ハードウェアがサポートしているメモリコンシステンシモデルと密接な関係がある．

仮想ページの動的割当て管理は，現在研究段階であり，方策決定に関する種々のパラメータを設定して，評価している研究がある[6]．

その他にも，メモリ管理の構造，メモリ—2次記憶間の管理方法，などの研究課題がある．また，分散共有メモリアーキテクチャでは，ページ割当て方策とスケジューリングは連動すべきものであり（memory affinityと呼ぶべきかもしれない），今後の研究が待たれる[7]．

3. おわりに

参考文献は，原論文ではなく，孫引きできる文献を主に選んだ．本稿の詳細は，参考文献から参照していただければ幸いである．日本の並列OS事例は本特集に譲るとしても，誌面の都合で海外の並列OS事例（文献1）で紹介したものの他に，オブジェクト指向のChoices[8]

Renaissance[9], Elmwood[10], 最近では, Spring[11]などがある), マイクロカール[12]と並列処理, などは割愛した. また, 海外の商用OS一般についての1つの情報として, URL http://www.yahoo.com/Computers/Operating_Systems/がある. 最後に, 本稿が, 並列処理に興味のある方々の一助となれば幸いである.

参考文献

1) 福田 晃: 並列オペレーティング・システム, 情報処理, Vol. 34, No. 9, pp. 1139-1149 (1993).
2) Mukherjee, B. and Schwan, K.: A Survey of Multiprocessor Operating System Kernel, Technical Report GIT-CC-92/05, College of Computing, Geogia Institute of Technology (1992).
3) USENIX Symp. on Experiences with Distributed and Multiprocessor Systems (SEDMS I-IV) (1989, 1991-1993).
4) McCann, C., Vaswani, R. and Zahorjan, J.: A Dynamic Processor Allocation Policy for Multiprogrammed Shared-Memory Multiprocessors, ACM Trans. Comput. Syst., Vol. 11, No. 2, pp. 146-178 (1993).
5) Lim, B. and Agarwal, A.: Waiting Algorithms for Synchronization in Large-Scale Multiprocessors, ACM Trans. Comput. Syst., Vol. 11, No. 3, pp. 253-294 (1993).
6) LaRowe, R.P., Ellis, C.S. and Holliday, M.A.: Evaluation of NUMA Memory Management Through Modeling and Measurements, IEEE Trans. Parallel and Distributed Systems, Vol. 3, No. 6, pp. 686-701 (1992).
7) 甲斐久淳, 藤木亮介, 福田 晃: メモリ管理と協調動作する2レベルスケジューリング, 並列処理シンポジウムJSPP'94, pp. 319-326 (1994).
8) Campbell, R, H., Islam, N., Raila, D. and Madany, P.: Designing and Implementing Choices: An Object-Oriented System in C++, Commun. ACM, Vol. 36, No. 9, pp. 117-126 (1993).
9) Russo, V.F.: Process Scheduling and Synchronization in the Renaissance Object-Oriented Multiprocessor Operating System, Proc. the USENIX SEDMS II, pp. 117-132 (1991).
10) Leblanc, T., Mellor-Crummey, J., Gafter, N., Crowl, L. and Dibble, P.: The Elmwood Multiprocessor Operating System, Software-Practice and Experience, Vol. 19, No. 11, pp. 1029-1056 (1989).
11) Mitchell, J. et al.: An Overview of the Spring System, Proc. Compcon Spring 1994, pp. 122-131 (1994).
12) Proc. the USENIX Workshop on Micro-kernels and Other Kernel Architectures (1992, 1993).

(平成6年7月11日受付)

福田 晃 (正会員)

1954年生. 1977年九州大学工学部情報工学科卒業. 1979年同大学院修士課程修了. 同年NTT研究所入所. 1983年九州大学大学院総合理工学研究科助手. 1989年同大学助教授. 1994年より奈良先端科学技術大学院大学情報科学研究科教授. 工学博士. オペレーティング・システム, 並列化コンパイラ, 計算機アーキテクチャ, 並列/分散処理, 性能評価などの研究に従事. 本会平成2年度研究賞, 平成5年度Best Author賞受賞. 訳書「オペレーティングシステムの概念」(共訳, 培風館). ACM, IEEE Computer Society, 電子情報通信学会, 日本OR学会各会員.