3

システムソフトウェア特論 試験予想問題

- 並列OS
  1. マルチプロセッサのアーキテクチャをメモリアーキテクチャの観点から分類し，その各々について，OSが考慮しなければならない問題を述べよ．
  2. カーネルデータ構造の統一性について，下記の問題に答えよ．対象マシンは，単一プロセッサシステムとする．
     i. どのような場合に，統一性が壊れる可能性があるか？
     ii. また，その契機を分類し，各々の場合への対処方法を示せ．
     iii. 共有メモリ型マルチプロセッサでのOSの実装方法を分類し，各々の方式について，実現，性能の観点から利点，欠点を述べよ．
     iv. プログラム内のスケジューリング，データ配置など，高度化している自動並列化コンパイラと絡めて，並列OSの果たすべき役割について，思うところを述べよ．
- プロセス管理とスケジューリング
  1. 共有メモリ型マルチプロセッサにおいて，(初期の)UNIXを実装したとする．本UNIXでは，1つの下層アドレス空間と1つのコンテクストを一体化させたUNIXプロセスのみをユーザに提供しているとする．この環境において，並列処理の軽さの観点から，問題点を述べよ．また，軽い並列処理環境を提案し，その利点，欠点を述べよ．

     解説）スライド二章7～10ページ
     問題点:UNIXプロセスの生成には、仮想アドレス空間の生成およびファイル資源などの管理を伴うので、比較的粒度の小さなアクティビティを並列実行させる場合、UNIXプロセス操作のオーバーヘッドが無視できなくなり、効率的な実行ができない

     提案：スレッドを用いた並列処理環境
     利点：生成/消滅、同期などオーバーヘッドが小さい
     欠点：スレッドの管理するレベルによって各々の問題に対する対処が必要

  2. スレッドモデルを3つあげ，説明せよ．また，各々のモデルの利点，欠点を述べよ．

     解説：スライド二章11～18ページ

     カーネルレベルスレッドモデル
     カーネルが仮想プロセッサとしてスレッドを提供し、ユーザアクティビティはカーネルスレッドに1対1にマッピングされて実行されるモデル
     利点：ユーザアクティビティのカーネル内での動きがユーザ空間に見えやすい
     欠点：高い性能を引き出せない。柔軟性にかける。オーバースペックになりやすい

     ユーザレベルスレッドモデル
     ユーザ空間でスレッドの生成、消滅及びコンテクストスイッチなどのスレッド管理を行うもの
     利点：関数呼び出し程度の軽さでスレッド操作を実現できる
     欠点：ユーザ空間から仮想プロセッサの物理プロセッサへのマッピング状況を知ることができない

     協調型ユーザレベルスレッドモデル
     ユーザ空間とカーネル空間の間で情報伝達を行うことができるユーザレベルスレッドモデル
     利点：カーネルレベルスレッドモデルの持つ機能性、ユーザレベルスレッドモデルの持つ柔軟性を兼ね備えている
     欠点：カーネルの再構築に多大な労力がかかる

  3. set_jmp関数，long_jmp関数を用いて，胡ルーチンを実現する概略フローチャートを示せ．

  4. マルチプロセッサのスケジューリングについて，考慮すべき問題点を述べよ．

     マルチプロセッサのスケジューリングについては、同期する際のタスク毎のロック取得や、各プロセッサにおけるコンテキストスイッチングのオーバヘッドを考慮する必要がある。
     また、プロセッサに割り振られるタスクの負荷がなるべく均等になるように負荷分散を行うこと、クリティカルパス（最も実行時間が長いパス）の優先度に留意することが重要となる。

  5. コスケジューリング(Coscheduling)を説明し，実現方法を示す．

     コスケジューリングとは、協調するプロセス群を同時に実行しようとするスケジューリング方式であ

る。
特定のプロセス群を同時に実行するため、同期のオーバヘッド軽減を期待できる。

実現の際には、タスクフォース内のプロセス数分のプロセッサ群を割り当てる。
実現方法は、以下の３つがある。
プロセッサの台数をｐ、各プロセッサが実行できるプロセス数をｑとする。
・行列を用いた方式
ｐ×ｑ個のスロットをｐ×ｑの２次元配列とする。
各行から１つのタスクフォースに含まれるプロセス数だけの空きスロットが有るかを判定していき、あればプロセスをスロットに割り当てる。
各プロセッサは各行をラウンドロビン方式で実行する。
利点：
アルゴリズムが簡単
欠点：アイドルプロセッサが生じやすい
タスクフォースを同時に切り替えるためにページングデバイスなどの共有機能を同時に使う確率が増加し、ボトルネックとなる。

・連続アルゴリズム
スケジューリング空間は１次元。
スロットとプロセッサを１対１に対応させる。
プロセッサをスロットに割り当てるには、
１．スロット系列の幅がｐのウインドを用意する
２．ウインド内の空きスロットの数を見て、１つのタスクフォース内のプロセス数以上のスロットがあれば割り当てる。
無い場合は、ウインド内の最左端のスロットが空きかつ、最左端の一つ前のスロットが空きでないようにウインドを移動する。

各プロセッサはプロセスをラウンドロビン方式で実行する。
タイムスライスを使いきればウインドを実行されなかったタスクフォースの先頭のプロセスまでずらす。

利点：
行列を用いた方式よりも空きスロット数が減少する
全てのタスクフォースにプロセッサを割り当てることが可能
欠点：
ウインド間で散在する空きスロットを有効するために、パッキングが必要
ウインドの移動速度が非分割アルゴリズムよりも遅い

・非分割アルゴリズム
一つのタスクフォースを連続して、プロセスを割り当てる。
ウインド間で散在する空きスロットを有効するために、パッキングが必要

利点：
連続アルゴリズムよりもウインドの移動速度が速い
欠点：
連続アルゴリズムよりも連続した空き領域が必要となり、パッキングの頻度が多くなる

6. NUMA型マルチプロセッサにおけるスケジューリング方式を自分で提案し，その利点，欠点を述べよ．

- 同期機構
  1. キャッシュが装備されている共有メモリ型において，単にセットアンドセット命令を用いたスピンロックでは，スピードの観点から効率が悪い．なぜか？その理由を述べよ，また，これを改善した方法を示せ．

理由
スヌーピングキャッシュを用い，ライトバック方式でデータを書き込み，書き込み時に他のキャッシュ内の値を無効化する場合を考える．
Test&Set命令では書き込み操作を伴う．
書込み操作では，他のキャッシュのlockをメモリに書き戻し，自分のキャッシュへデータを移動させ，他のキャッシュのlockを無効化させる．

この3つの操作それぞれでバスを使用するため，ループして定期的にTest&Set命令でlockをチェックすると，バストラフィックが増加し効率が悪くなる．
(第3章:P.7～)

改善方法
ループして定期的にロックが獲得できるかどうかチェックし，獲得できる可能性が有る場合Test&Set命令でロックを取りに行く．
ロック獲得可能性の判断はlock変数のリード命令のみなので，単にTest&Set命令を道いた場合と比べバストラフィックが増加しない
(第3章:P.9～)

2. 共有メモリ型マルチプロセッサにおいて，スピンロックを用いたアルゴリズムを3つあげ，説明せよ．また，各々のアルゴリズムの利点，欠点を述べよ．

1. スヌーピングロック方式
キャッシュを意識したロック方式
ロックを獲得できる可能性があるとき，Test&Set命令でロックを取りに行く．
ロックが獲得できるかどうかはロック変数のリード命令のみで判断．
利点：単純ロック方式と比べバストラフィックが削減できる
欠点：ロック解除時における解除待ちプロセッサのバースト的アクセスが発生

(第3章:P.9～)

2. 衝突回避ロック方式
複数のプロセッサがロック待ちしているとき，同時にTest&Set命令を実行するために起こる衝突を遅延を挿入することにより回避する
利点：Test&Set命令の衝突が起きにくくなる
欠点：各プロセッサの遅延の値によっては飢餓状態(全くロックを取得できないプロセス)が発生する可能性がある

(第3章:P.12～)

3. トーナメントロック方式
ロック変数を分散させる
リーフノードからスタートし，上位ノードのロックを取得していく．
最終的に，ルートノードのロックを取得する．
利点：1つの親ノードに対して子ノードが2つしかないので，Test&Set命令の衝突がおきにくい?
欠点：ロック変数が多いため消費メモリが多くなる

(第3章:P.15～)

3. バリア同期において，次の問いに答えよ．
i. 別紙に示すアルゴリズムTは正しいか，正しくないか，証明せよ．このとき，バリア同期は次の2つを満たさなければならないとする．
   - バリア同期の働きをする(足並みをそろえる)
   - 再初期化問題に対処している

また，プロセッサ速度に関して何の制限もないとする．アルゴリズムT内の各ブロックはむろん，クリティカルセクションである．

<<proof>>
プロセス数2個p1,p2(異なるプロセッサ上N=2)とする。
このとき、
a、p1がバリア同期点に到着、3で待つ
b、p2が到着、2を実行後プロセッサが横取りされ、4の前でとまる
c、p1がバリア同期点を通過でき、さらに次のバリア同期点に入れる
上記の実行系列により、バリア同期の再初期化問題が解決されていない。
よって、正しいアルゴリズムではない。

(以下解説)
4で初期化する前にp1が抜けてしまうと、countが0にされる前に、p1が次の同期点に入ってしまう可能性があり、そのままだと次の同期点も抜けてしまう。
また、count=Nのときだけ3を抜けるとしても、上記のことが起きると、p1は次の同期点でと

まっているが、p2が到着した際、count=1となっているため、p1とp2はどちらもバリア同期点から抜けることができなくなってしまう。

ii. バリア同期を用いる具体例を示せ.

行列の掛け算について
各要素の計算をプロセッサに分割して行い、各計算結果が出そろうまで、バリア同期で同期をとる。

4. 共有メモリ型マルチプロセッサにおいて，バリア同期のアルゴリズムを3つあげ，説明せよ．また，各々のアルゴリズムの利点，欠点を述べよ．

集中型バリア同期(3章30、31枚目)
コンバイニングツリー(3章33，34枚目)
トーナメントバリア(3章35，36、37枚目)
ツリーバリア(3章38，39枚目)

比較：3章40枚目
集中型　　利点：他のものに比べてO(N)と高速、わかりやすい　欠点：不可分命令がある
コンバイニング　　利点：実装が簡単(?)　欠点：不可分命令がある
トーナメント　　利点：不可分命令がない　欠点：メモリ使用量が多い(?)
ツリー　　利点：不可分命令がない　欠点：メモリ使用量が多い(?)

o メモリ管理

1. キャッシュを装備している共有メモリ型マルチプロセッサにおいて，次の問いに答えよ．

i. キャッシュコヒーレンス問題とは何か？
共有メモリ型並列計算機では，実メモリとキャッシュ，あるいは，キャッシュ相互の内容が一致しない可能性が出てくる．

ii. キャッシュコヒーレンス問題を解決する方法を分類して，その方法を列挙し，各々の方式の利点，欠点を述べよ．

1）書込み時無効化方式（write invalidate）
利点：元の情報処理装置はキャッシュに書き込みが可能となる。各情報処理装置上にキャッシングされたデータは、ある情報処理装置が書き込みを行なうことで再び無効化される。以上のような手続きを繰り返して、各キャッシュの一貫性を保持する。
欠点：キャッシュの無効化要求やデータ転送が、主記憶のページの無効化要求やページデータの転送になり、ネットワークに大きな負荷を掛ける大量のメッセージ転送が発生する。転送の帯域がかなり低い。情報処理装置の負荷も重くなり、処理能力が極端に低下するという問題があった。

2）書込み時更新方式（write update）
利点：主メモリーにデータの最新コピーが維持される、新しい値がキャッシュに早めに反映されるため、待ち時間が短くなる。
欠点：すみません、できません。

2. 共有メモリ型マルチプロセッサにおいて，TLBコヒーレンス問題に関して次の問いに答えよ．

i. TLBコヒーレンス問題とは何か？具体例を挙げて説明せよ．

TLBは仮想ページ(ページ)と物理ページ(フレーム)の対をキャッシングし、保持している。
いま、複数のプロセッサがある同じページ・フレーム対の情報を各々のTLBに保持しているとする。
(講義資料の例ではページ0とフレーム2)
もし、あるプロセッサ0上で動いているスレッド0がこのページ・フレームのマッピングを変更した場合、
(講義資料ではページ0とフレーム2のマッピングからページ9とフレーム2のマッピングに変更しています)
プロセッサ0はこの変更を認識し、自分のTLB内の該当するページ・フレーム対の情報を変更する

か、無効化することができるが、
ほかのプロセッサはページ・フレーム対の情報が古いままである。よって、TLB情報の一貫性を保つ必要がある。
これがTLBコヒーレンス問題である。(具体例もいるので、講義資料の図4.1を参照してください)

ii. TLBコヒーレンスを解決するアルゴリズムを示し，その利点，欠点を述べよ．

○シュートダウンアルゴリズム(2．詳細を主に書けばいいと思います)
1．概要
ページ表エントリを変更したいプロセッサ(イニシエータ)が、まず、当該エントリをキャッシングしているプロセッサ
(レスポンダ)に、変更したい旨の通知を出し、全レスポンダから応答が返ってくると、イニシエータは、ページ表エントリ
を書き換える。
2．詳細
・イニシエータの処理
ⅰ．イニシエータは、コヒーレンス制御のための準備を行う。具体的には、割り込み禁止の設定、制御に参加していることを示す
ためのフラグ設定(active[self]=0)、当該ページ表のロック、無効化メッセージのキューイングがある。
ⅱ．レスポンダにプロセッサ間割り込みを用いて、無効化を依頼する。
ⅲ．自TLBを初期化する。
ⅳ．全レスポンダカからのⅱの割り込み通知に対する応答を待つ。(レスポンダrに対してはactive[r]=0で判断する)
ⅴ．全レスポンダから応答が返ると、ページ表エントリを変更して、当該ロックを解除して、全レスポンダに完了を知らせる。
ⅵ．割り込みレベルを元に戻す。

・レスポンダの処理
ⅰ．イニシエータからのプロセッサ間割り込みを受け付けると、割り込み禁止に設定し、応答を返す。(active[self]=0)
ⅱ．ページ表エントリの変更が完了するまで、ページ表対応のロック変数上でスピンして待つ。
ⅲ．ロックが解除されると、無効化依頼メッセージを削除し、自TLBを無効化する。
ⅳ．フラグの初期化、割り込みレベルの復帰の後処理をして、割り込み前に戻る。

○利点
・完全なコヒーレンスを保証できる。
・イニシエータが起動中のコヒーレンスを保証(レスポンダをアイドルにする)
○欠点
・同期の際のオーバーヘッドが大きい。
(イニシエータは全レスポンダのアイドルを確認した後、ページ表を変更するし、レスポンダはページ表変更完了まで繁忙待機する)
・プロセッサ間通信レベルの低いハードウェア上では、割り込み受信が遅れる。

3. コピーオンライと(Copy-on-write)とは何かを説明せよ．また，ページ表エントリ属性の設定と絡めて，その実現方法を述べよ．

コピーオンライトとはプロセスの複製を要求されたときには、複製を行わず、書き込み処理が行われた時に初めて複製を行う仕組み。

親プロセスのコピーを作成(fork)する時、子プロセスのページ表では、親プロセスのアドレスを参照する。このときの親プロセスのデータは書き込み制限を行う。
データの書き込みがあると、親データの複製を行い、子プロセスのデータは、複製されたデータを参照する。このとき、親プロセス、子プロセスのデータの書き込みを可能にすることによってコピーライトを実現している。

4. 分散共有メモリ(Distributed Shared Memory)とは何が？説明せよ．また，DSMを実現する方法を挙げて具体的に説明し，その方式の利点，欠点を述べよ．

DSM(分散共有メモリ)とは、物理的に分散されているメモリをユーザには共有に見せる仕掛けである。(4章P24)

実現方法として、ソフトウェアで実現する方法がある。
全てのプロセッサ上のDSMブロックに一意の識別子を割り振り、仮想の共有メモリを作成する。
プロセッサは自分のメモリにしかアクセスできないので、他ノードが持っているDSMブロックにアクセスする場合、ローカルメモリへのキャッシングを行う。

書き込みアクセスを検知は、ソフトウェアによる方法(コヒーレンスが必要な書き込みアクセスの直後にコヒーレンス制御ルーチンを入れる)を使用する。
キャッシュコヒーレンス方法には書き込み時更新方式(書き込みアクセスが生じた際に当該DSMブロックを他ノードが持っていればそれを更新する方式)で行う。
ブロック書き換えアルゴリズムにはLRUを使用する。

書き込みアクセス検知をソフトウェアにより行うことで、コンパイラを修正する必要があるという欠点があるが、ブロックのサイズをユーザが任意に設定できるのでフォールスシェアリングが軽減される、システムコールの発行回数が削減されるなどの利点がある。

5. NUMA型マルチプロセッサでは，ローカルメモリ，リモートメモリとの間で，メモリアクセス時間が不均一となる．これを意識したメモリ管理方式を提案せよ．

- その他
  1. OSの夢を語れ．

- 組込みシステム
- 概論
  1. 組込みシステムを，自分なりに分類軸を考えて，分類せよ．
  2. リアルタイムシステムにおいて，タスクがデッドラインを守れなかった場合に当該システムにどのような影響があるかの観点から，リアルタイムシステムを分類し，各々の例を示せ．

- リアルタイムスケジューリング
  1. 優先度固定のスケジューリングの中で，レートモノトニックスケジューリング(Rate Monotonic Scheduling)は最適なスケジューリングである．その根拠を示せ．

* レートモノトニックスケジューリング：
優先度を固定する静的スケジューリング方式の一つ。周期の短いタスクに、より高い優先度をつけるスケジューリング。詳細は、システムソフトウェア特論の資料「リアルタイムスケジューリング」のスライドNo.14~25を参照。

解答：
全てのタスクが周期的であるとき、周期の長いタスクの優先度を高くしてスケジュールしたものがスケジュール可能であれば、レートモノトニックスケジューリングでも必ずスケジュール可能である。これは、以下のようにタスク数を一般化した場合(タスク数:n個)でも成立する。

n個タスク(t1, t2…tn)があり、それらのタスクの周期について、T1<T2<…<Tnとする。ここで、Tiはタスクtiの周期。Tiの優先度をP(ti)とする。このとき、次の2つのスケジューリングを考える。
・スケジューリング1：P(t1)>P(t2)…>P(tn)なる優先度を各タスクに付けてスケジューリングする(レートモノトニックスケジューリング)。
・スケジューリング2：上記スケジューリング1以外の優先度を付けてスケジューリングする。
このとき、スケジューリング2でスケジューリング可能であれば、必ずスケジューリング1で可能であることが成立する。

これは、周期タスク・固定優先度が前提の下では、レートモノトニックスケジューリングによってスケジュールできないタスク集合をスケジュールできる方式はない、ということを示す。よって、レートモノトニックスケジューリングは最適なスケジューリングの方式である。

2. 2つの周期タスクP1，P2のタスクセットを考える．ここで，P1=(3,5)，P2=(2,7)とする．但し，(実行時間，周期)．このとき，このタスクセットに関して，次のリアルタイムスケジューリングでスケジュール可能か否かを判定せよ．また，スケジュールの時間的推移を示せ．

i. レートモノトニックスケジューリング(Rate Monotonic Scheduling)
ii. EDFスケジューリング(Earliest Deadline First Scheduling)

- その他
  1. 組み込みシステムの夢を語れ.