オペレーションシステム

久保田知恵

2018 年 5 月 11 日

1　第 1 回

特定の OS に注目するものではない資源：計算を行うのに利用できる存在公平性と効率性のバランス API : ABI: 仮装計算機資源（メモリ、ディスク）の違いの差を全て意識してつくらなくてもよい？コンテナと VM のメリットの違い

UNIX Multics →マルチ　→　ユニ基本的な考えは同じベル研究所（C 言語開発したところ）

Apple　macOS もともと UNIX とは無関係→ macOSX の時に設計を刷新

2　第 2 回

3　第 3 回

4　第 4 回

4.1　ジョブの性質

4.2　スケジューリング

4.3　ポリシーの比較基準

- 公平かどうか
- 単純かどうか
- CPU の利用率
- スループット

このほかに、レスポンスタイムとターンアラウンドタイムが比較基準となる。

レスポンスタイム

ターンアラウンドタイム　バッチジョブの場合はこちらを重視

それぞれどう言う尺度を求められている。

- 先着順
- 最短時間順

- 後着順
- 優先度順
- ランダム（抽選型）
- ラウンドロビン
- 複合型ほとんどはこれ。ラウンドロビン＋何か
- 経過処理時間順
- 残余処理時間順

優先順位の高いプロセスに対して、いつディスパッチすべきか。今の OS はプリエンティブ・・・仕事の途中であってもディスパッチ？される。ノンプリエンプティブ・・・実行中のプロセスが CPU を手放さないとディスパッチされない

先着順　First-Come,First-Served:FCFS ともよばれる。FIFO とも。とてもシンプル。先に並んだ方が有利なので、ディスパッチはない。例：バッチジョブ。デメリットは、時間のかかる仕事が先に並ぶとキューが伸びる。図__の例では、ターンアラウンドタイムは A=4,B=9,C=11（単位時間）であり、平均は 8 時間である。

最短時間順　Shortest Job First:SJF 短い仕事ほど優先順位が高い。平均待ち時間は（理論上）最短になる。しかし、実際には利用ができない。なぜなら、同じアルゴリズムでもプロセス終了にかかる時間が異なることがある（むしろ多い）からである。理論値であるからこそ、ベンチマークの指標として使われることが多い。ターンアラウンドタイムは A=7,B=12,C=3 で、平均は 7.3 単位時間である。

優先度順　(fixed:固定された) priority scheduling あらかじめ優先度を割り振り、その優先度が優先順位となる。固定でない（プロセス途中で優先度が上下する）場合もある。

ラウンドロビン　総当たり戦、円卓会議のこと。優先順位はなく、公平。そのかわり、一定時間（＝タイムスライス）ごとにプロセスを切り替えるタイムスライスを長くすれば、FCFS に、短くすれば、最短順に近く。あまりタイムスライスを短くすると、プロセスを切り替える際のオーバーヘッドが増加してしまう。

その他

- 経過処理時間順 shortest elapsed time first CPU の利用時間を累積させ、経過時間の短いものを優先。公平性がある。利用時間を保持しなければいけない。
- 残余処理時間順 shortest remaing time first 残り時間が短いものから優先。残りがどのくらいかわからないと言う問題
- 降着順

4.4　CPU 待ち時間

待ち時間が長ければ長いほど優先度を上げてあげよう。→公平性 resouce starvation　CPU 利用時間に応じて優先度を下げよう。

実装例：多段フィードバックキュー

多段フィードバックキュー　nice 値を用いて、優先度を設定（windows は 32 段階）。基本はラウンドロビン・先着順で処理する。CPU 利用時間・待ち時間で優先度を上下してやる

4.5 CFS

Completely Fair Scheduling スッゲー完璧なスケジューリング。二分探索木を作りながら、仮装経過時間のみで決定する。

5 第 5 回