Linux Kernel における CPU Resource Reservation の比較･検証

日立製作所
Lineo Solutions, Inc.

内容

- 概要
- 調査対象
- 評価
- 機能比較
- まとめ

概要

- 過去の Jamboree でも既に何回か紹介あり。
- Linux kernel のスケジューラはスケジューリング・ポリシとプライオリティでプロセスをスケジュールする。
- CPU Resource Reservation は最低限のCPU割り当てを保証する機能をスケジューラに加える。
- コミュニティにはいくつかの手法が存在。
- CPU Reservation の詳細な内容は、Jamboree #4 の早稲田大学、菅谷さんのプレゼンを参照。

調査対象

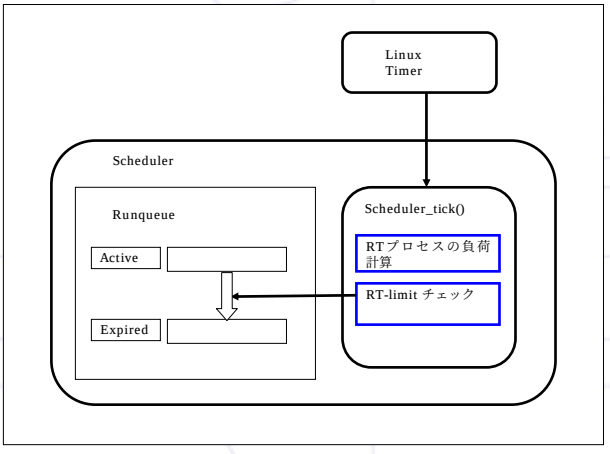
- Rt-limit
- Isochronous scheduler
- CABI （CPU Accounting and Blocking Interface）
- CKRM (Class-based Kernel Resource Management)

Rt-limit

1．処理概要

- RTプロセスに対する優先的なスケジューリングに制限。
- パッチ適用時、defaultでRTプロセスは80%使用可能。
- 単純にRTプロセスのCPU使用率が閾値を超えたらRT以外のプロセスにリスケジューリング。
- 負荷計算（CPU使用率）
  - RTプロセス：rt_cpu_avg = （rt_cpu_avg * (HZ/10-1) + HZ)/(HZ/10)
  - 通常プロセス：rt_cpu_avg = rt_cpu_avg * (HZ/10-1)/(HZ/10)

Rt-limit ブロック図

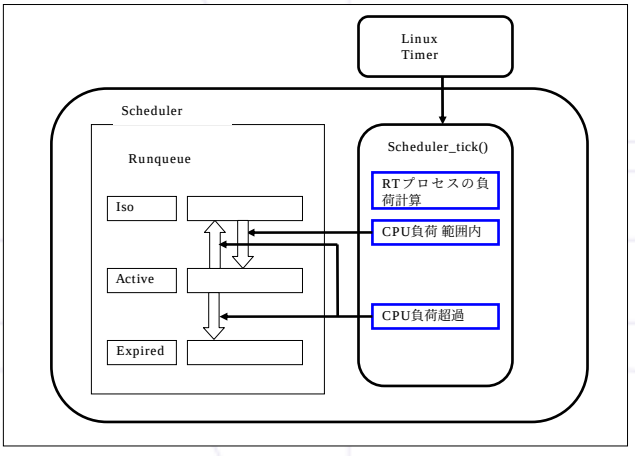


Isochronous scheduler

1. 処理概要

- 新たなスケジューリング ポリシSCHED_ISO_RR、SCHED_ISO_FIFOを追加。
- 特権のないユーザがRTプロセスを生成するとSCHED_RR、SCHED_FIFOは上記ポリシに置き換えられる。
- 追加したポリシに独自の優先順位を与える。優先順位はRTプロセスとNORMALプロセスの中間。
- CPUの使用限界を超えると、CPU使用限界の90%に低下するまでSCHED_ NORMALとして動作する。

Isochronous scheduler ブロック図



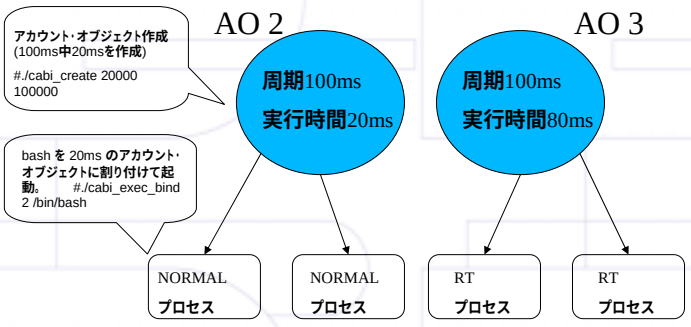
CABI

1. 処理概要

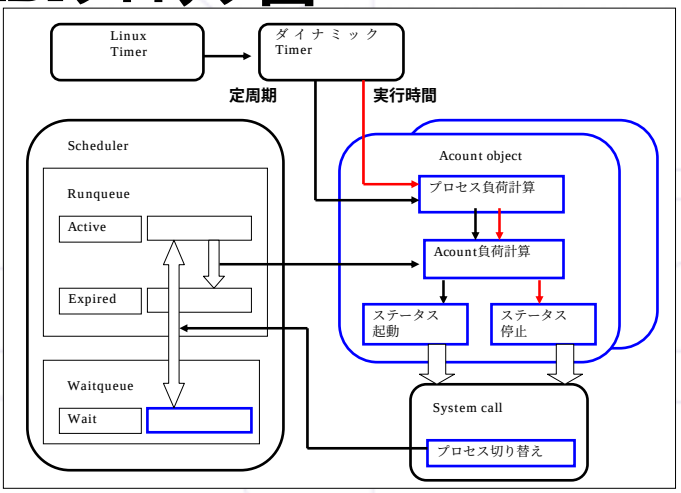
- 周期時間と実行時間を設定するアカウント･オブジェクトを作成。
- アカウント･オブジェクトにRTプロセスを結び付けることで実行を制限。
- アカウント･オブジェクトの設定は追加されたシステムコールを使用。
- HRTを使用することで実行時間と周期時間の精度を向上することができる。

CABI cont.

2. アカウント･オブジェクト



CABIブロック図



CKRM

1. 処理概要

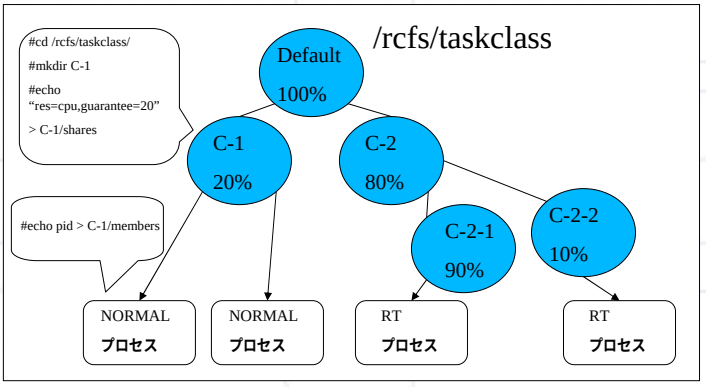
- Linux カーネルの CPU 時間、メモリサイズ、ディスク I/O 帯域幅、TCP 接続に優先順位をつけるような資源の管理を一般化するための枠組である。
- リソースコントローラはリソース単位で用意される。
- 今回は“CPU時間“のリソースコントローラを利用する。
- 資源管理はクラスで行う。
- クラスは階層的に定義できる。
- クラスの生成、パラメータ変更はFS (rcfs) を通して設定。

CKRM cont.

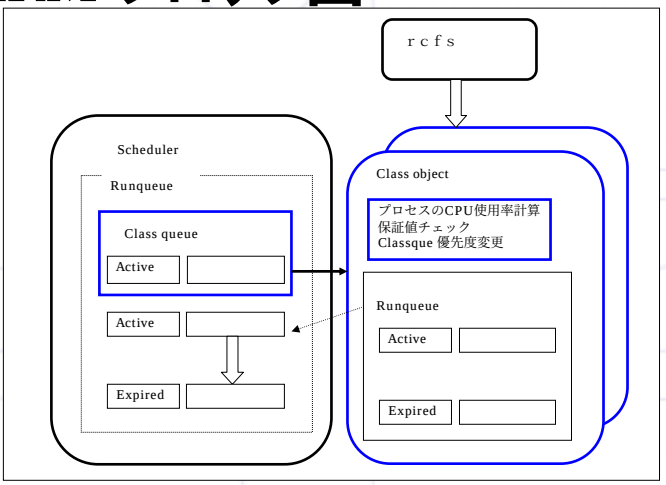
- rcfsでディレクトリをmountするとその下に Taskclassディレクトリが自動的に作成され親クラスになる。さらにその下にディレクトリを作成すると子クラスとなる。Taskclassを最上位クラスとし、それ以下のディレクトリの階層がクラスの階層と同じになる。
- 下位クラスのリソースは上位クラスのリソースに対する割合を指定する為、管理方法は簡単。

CKRM cont.

2. クラス



CKRM ブロック図



評価

1. 環境

   2.6.x kernel x86

2. 評価方法

- 評価プログラム + コンソールによるキー入力の応答性
  - ✓ 評価プログラムは “write スレッド” と“read スレッド” から構成され、リングバッファからデータを書き込み/読み込みを行う。
  - ✓ この2つのスレッドを作成する “main スレッド” は起動時に自分自身と2つのスレッドのポリシーを RTスレッド (SCHED_RR) に切り替える。write スレッドはバッファに空きがあれば書き込み、read スレッドはバッファにデータがあればバッファから読み捨てる。各スレッドは “バッファが一杯で書き込めない” 又は “バッファが空で読み込めない” ときは CPU を解放することなくwait する。

評価結果

	キー入力応答性	備考
Rt-limit	○	
Isochronous scheduler	△	コンソールの処理が重い。
CABI	×	Busy-loopするRTプロセスはCPUを占有する。
CKRM	○	

機能比較

	メリット	デメリット
RT-limit	•実装が軽い •タイマ割り込みでプロセスの切り替えを行う為、確実に切り替えが可能	•RTプロセス全体の使用率しか指定できない為、個々のプロセスを管理できない •RTプロセスが連続実行することをブロックできるが、切り替わった通常プロセスの実行制限が無い為、リアルタイムプロセスに影響が出る可能性がある •RTプロセスのCPU使用率の指定が0から90%しかできない
CABI	•カーネルにシステムコールを追加し使用する為、スケジューラに対して変更量は少ない •プロセス単位に使用率、周期時間を設定できる •RTプロセス、通常プロセスの区別無くアカウントオブジェクトに関連付けできる •複数のプロセスを一つのアカウントに関連付けできる	•システムコールが呼ばれた後、プロセスに切り替わる際にプロセスをブロックしている為、システムコールを呼ばないプロセスはブロックされない •アカウントオブジェクトのタイムリミット判定は正確であるが、プロセスの切り替えまでには次回システムコール実行終了までのタイムラグがある

機能比較 cont.

	メリット	デメリット
CKRM	・rcfsを通してプロセスのリソースを設定できる為、ユーザプログラムの変更が不要 ・複数のプロセスをクラスに関連付けできる ・クラスを親子の関係で管理できる ・CABIに比べ処理が速い	・現状、起動しているプロセスに対してのみリソースの設定ができる ・処理が複雑
Isochronous scheduler	・実装が軽い。 ・ブロックされたくないRTプロセスを実行することが可能。	・NORMALプロセスの処理が重い ・CKRMと比べ、処理が遅い

まとめ

使い分けるならば・・・

- RTプロセスをブロックしRT以外のプロセスにリソースを割り当てたい
  RT-limitを使用
- RTプロセスをブロックしRT以外のプロセスにリソースを割り当てたいが他にブロックされたくないRTプロセスがある
  Isocronuc schedulerを使用
- プロセス、プロセスグループ毎にリソースを割り当てたい
  CABI or CKRM を使用
- プロセス、プロセスグループ毎にリソースを割り当てたい。さらに、周期的に実行して欲しいプロセスがある。
  CABIを使用

まとめ

- CPU Resource Reservation の必要性
  - RTプロセスはシステム設計者が管理
  - Kernelが制限を加えることについて

その他

- CPU時間に制限加える方法、アイディアなど

参考までに

1. 入手先

RT-limit:

http://people.redhat.com/mingo/rt-limit-patches/

Isochronous scheduler:

http://ck.kolivas.org/patches/SCHED_ISO/

CABI:

http://tree.celinuxforum.org/CelfPubWiki/PatchArchive

http://sourceforge.jp/projects/cabi

CKRM:

http://sourceforge.net/projects/ckrm

HITACHI LINEO

参考までに cont.

2. diffstat の結果

RT-limit:

5 files changed, 169 insertions(+), 7 deletions(-)

SCHED_ISO :

5 files changed, 323 insertions(+), 78 deletions(-)

CABI :

56 files changed, 4756 insertions(+), 9 deletions(-)

CKRM :

69 files changed, 15125 insertions(+), 101 deletions(-)