CABI (CPU Resource Management) in Embedded Linux

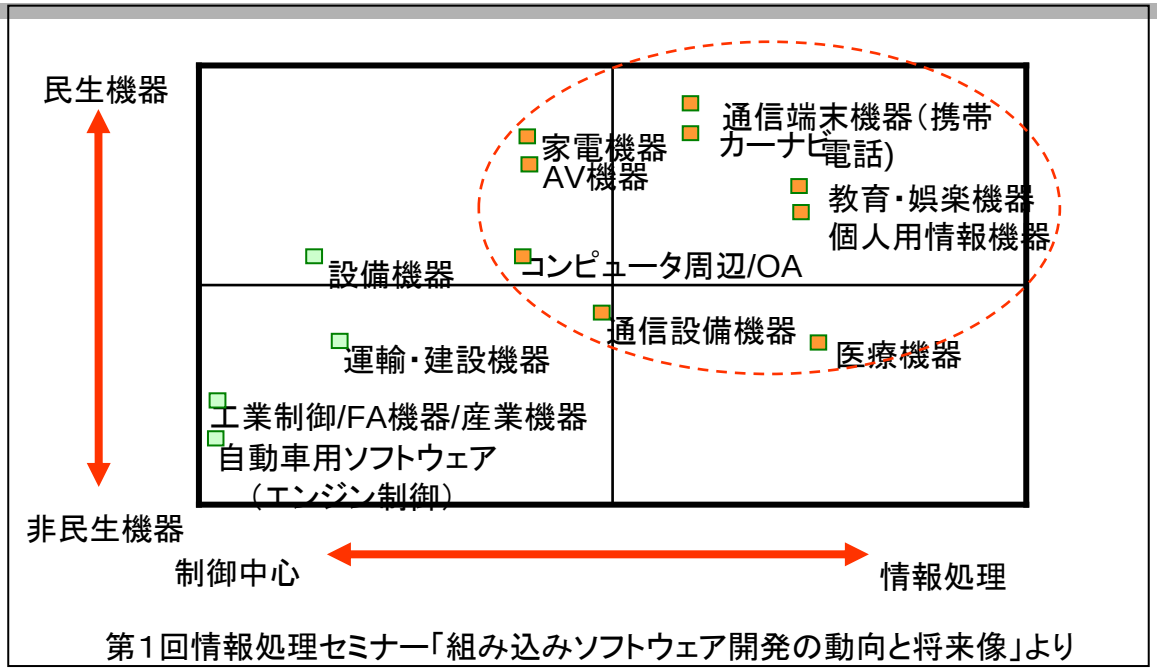
早稲田大学理工学術院
菅谷みどり

CE Linux Forum in 第9回 組込みシステム開発技術展（ESEC)
2006

発表内容

- Linux 上の資源管理
  - 組込みシステムの制約
  - リソース管理とは
  - Linux の資源管理の制限
  - CABIシステムの提案
  - システムの公開
- CABI の応用例
  - クラススケジューリング、資源保護、オーバーロード管理
- Priority Boost の紹介

組込みLinux の適用分野



第1回情報処理セミナー「組み込みソフトウェア開発の動向と将来像」より

Linux-powered robots

Linux-based mobile phones and smart phones

最近の傾向

- 適用分野の拡大
  - 携帯電話、カーナビ、PDA 、etc
  - 制御から情報処理へ
- PCレベルの処理機能
  - 大規模化、複雑化
  - GUIの高度化, ネットワーク化
- 開発期間の短縮

Linux

- 多様なカーネルコンフィグレーション
- マルチベンダーサポートのツール
- POSIX API
- プロセスモデル
- デバイスドライバ

情報系組込みシステムへのサービス要求

- **多様化、複雑化**
  - アプリケーション規模の増大
- 携帯電話、PDA、カーナビシステム



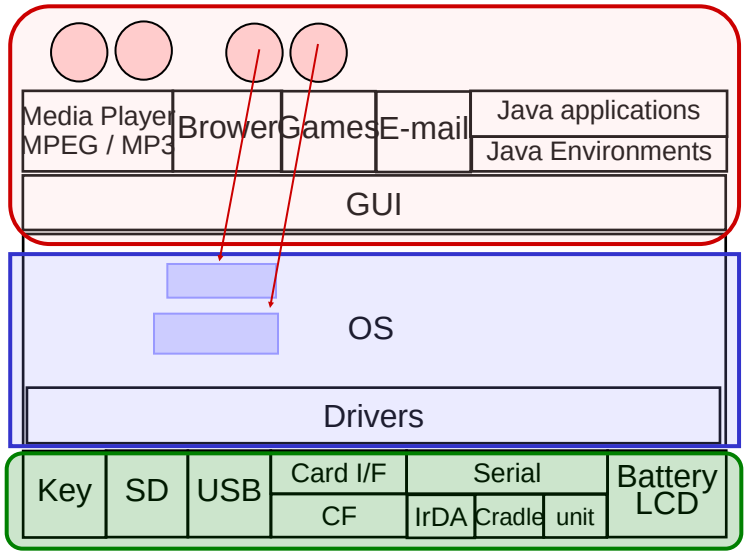


- **ビデオ・オーディオ再生**
  - 高負荷時の再生
  - マルチメディア再生と同時に、メールを受信させたい
- **メディア再生時のキー入力への応答性**
  - メディア再生しつつユーザーからの入力への応答も同時に確保したい
- **資源占有の防止**
  - ダウンロードしたアプリケーションが安全ではない場合
  - CPU を占有してしまうことを防ぎたい
  - リアルタイムアプリケーションの動作を制限して利用したい

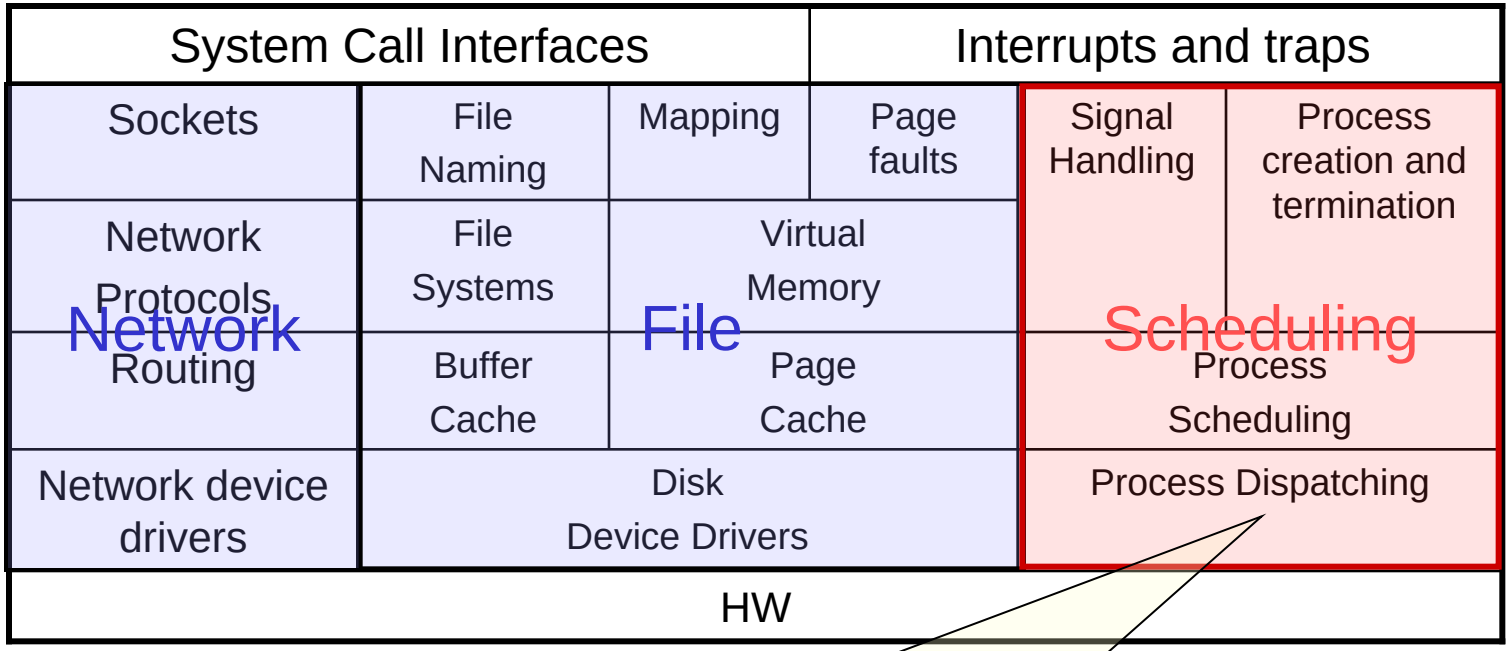
Linux上で様々なアプリケーションを同時に動かし、かつ、
サービスのクオリティを守るためには、資源管理（リソースマネージメント）が必要

リソースマネジメントとは

- 計算機上の資源
  - システムの目的に応じて、アプリケーションに適切に配分
- システムの目的
  - リアルタイム性能、応答性、保護
- 資源管理の提供は
  - ハードウェア　（資源提供）
  - アプリケーション　(資源要求）
  - オペレーティングシステム（調停）
    - 各処理の実行条件は OS が集中管理
- Linux による資源管理
  - 抽象化されたインターフェイス
    - プロセス：プログラムの実行単位
    - ファイル：ハードウェア資源を扱うためのインターフェイス
    - POSIX API の利用



Linux リソースマネジメントの構成



特にスケジューラは、プロセスの実行を決定
- プロセスの実行時間の管理
- 優先度（実行順序）の管理

Linux とリアルタイムOS の設計目的の違い

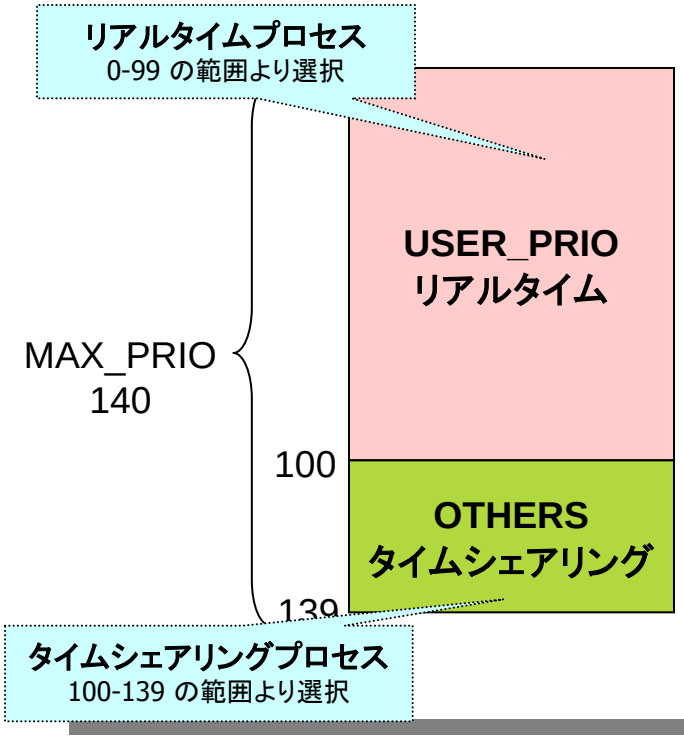
- UNIX に代表される汎用機向けOS
  - 組み込み向けリアルタイムOSとは性能に関する基準が異なる

	General Purpose Operating System	Real-Time Operating System
スケジューリング	システム志向、平均性能	ユーザー志向、リアルタイム性能
性能目標	**スループット** **公平性**	**時間制約、予測可能性** **応答性**
リソース	豊富	制約
ネットワーク接続	あり	なし → あり
セキュリティ脅威	あり	なし → あり

Linux スケジューラ

- Linux のスケジューリングクラス

タイプ	クラス	内容	優先度
リアルタイム	SCHED_FIFO	FIFO方式	0−99
	SCHED_RR	ラウンドロビン方式	
タイムシェアリング	SCHED_OTHER	時分割方式	100−139



– **リアルタイム:** RT **(固定優先度)**
  - Static Priority (POSIX 1003.1b)

– **タイムシェアリング：** TS **(動的優先度)**
  - 定期的(一定時間おき)にプリエンプション
    実行プロセスをスイッチ
  - 実行＋sleep 時間に応じて優先度を決定
    インタラクティブプロセスに高い優先度

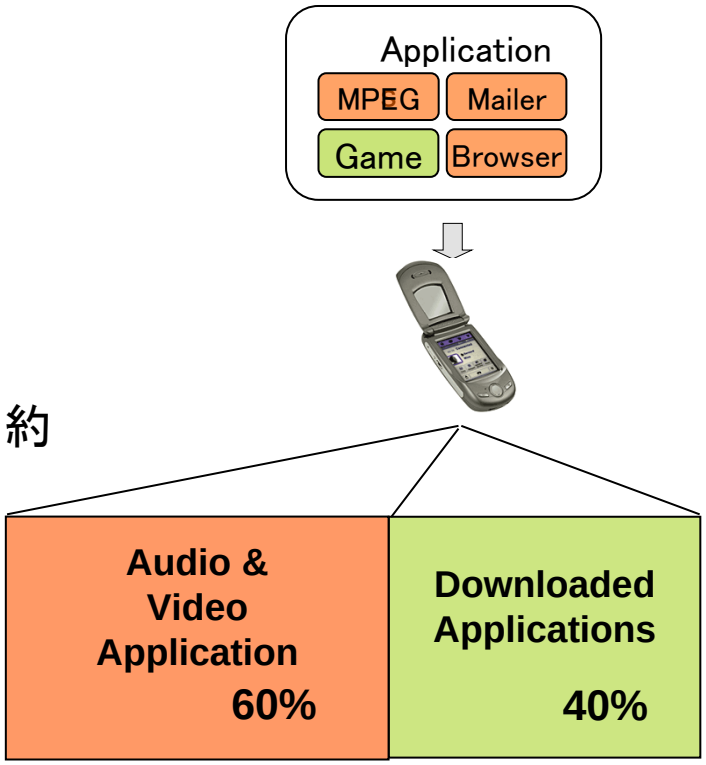
デフォルトはタイムシェアリング
リアルタイムはシステムコールによって利用

CPUリソースマネジメントの方針

問題区分	技術的な要求事項	機能要件
ビデオ・オーディオ再生 ↓ ソフト リアルタイム性	リアルタイムクラスの利用時 ↓ ・ 優先度と組み合わせた資源予約を行いたい ・ デッドラインを守らせるための仕組みがほしい	資源予約 負荷時のQoS
キー入力 ↓ 応答性	低優先度クラスのプロセスの応答性の低下 ↓ ・高優先度プロセスの利用上限値の制限をしたい	上限値制限
資源占有の防止 ↓ 保護	リアルタイムプロセスの資源占有 ↓ ・ リアルタイムプロセスの上限制限	上限値制限

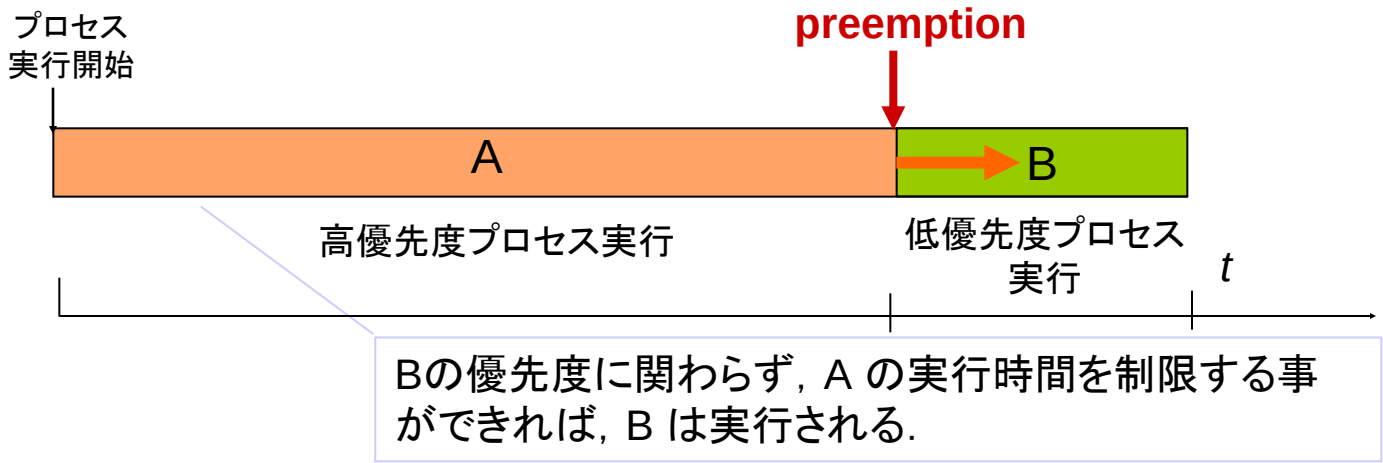
CABI (CPU Accounting and Blocking Interfaces)

- CABI とは
  - 組み込み向け CPU リソースマネジメントシステム
    - アプリケーション毎の実行時間制御のためのシステム
      - 定量的に周期と実行時間を与える事
      - e.g)
        - » マルチメディアアプリケーション 60%
        - » ダウンロードアプリケーション　40%
- 目的
  - **定量的 CPU リソースマネジメント**
    - マルチメディアアプリケーションへの資源予約
  - **応答性の向上**
    - キー入力時の応答性向上
  - **資源占有の防止**
    - 高優先度プロセスの資源占有の防止



CABI における非優先度割り込みのアプローチ

- **強制的なプリエンプション**
  - プロセス A, B （優先度 A > B)
  - ある周期内で，プロセスAが利用できる最大実行時間を制御
    - e.g. q = 60ms，周期 = 100ms (60%)
    - 優先度に依存しないバンド幅（時間）のみを対象とする.



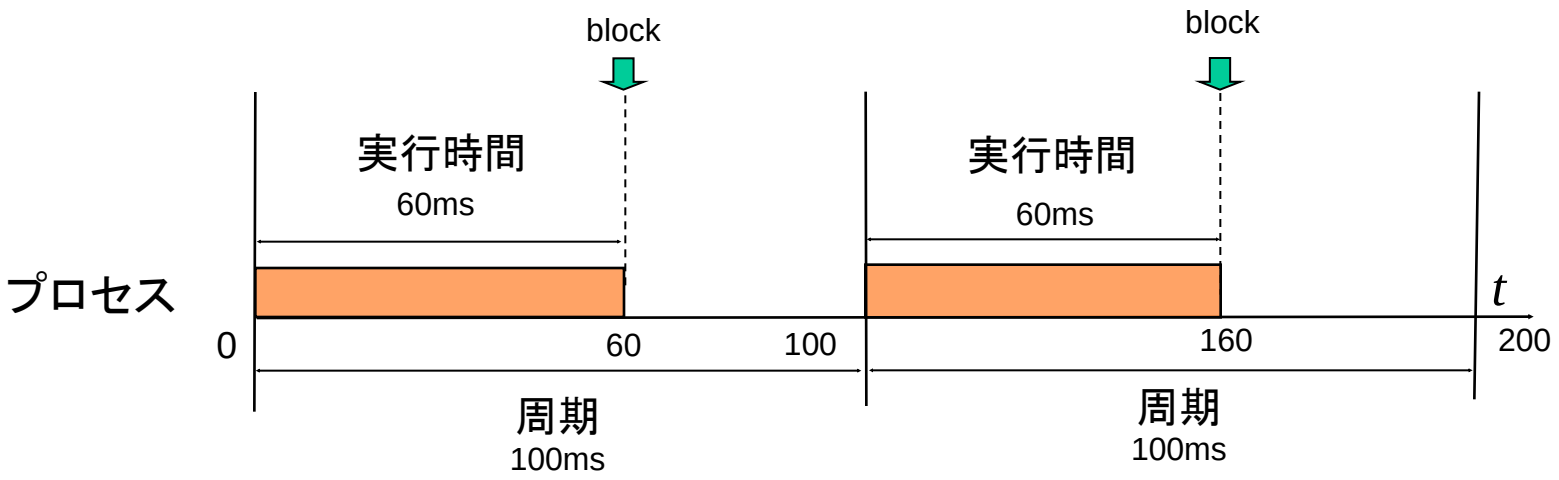
- 特徴
  - 優先度に関わらず，定量的な値（バンド幅）を確保する
    - 静的なパラメータを設定する

CABI の時間指定

- プロセスの実行時間は周期と実行時間で指定

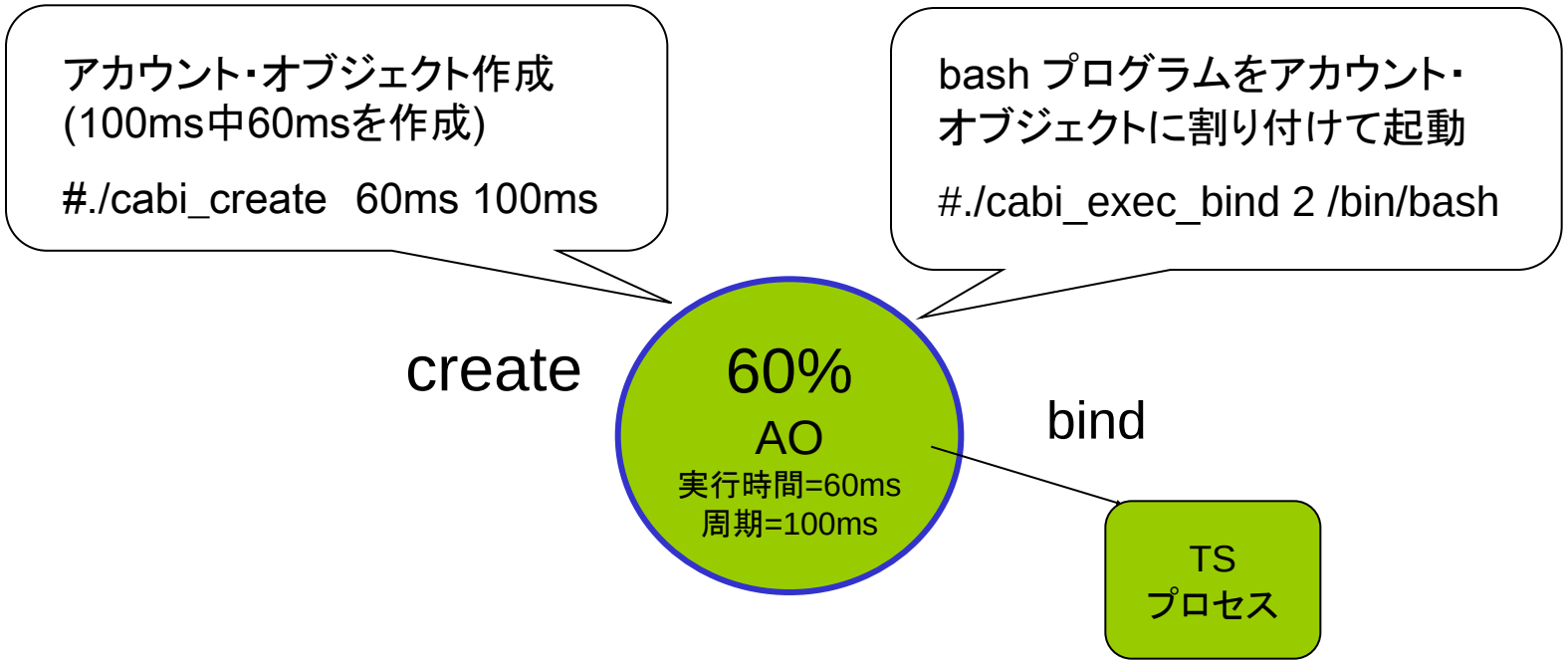
$$\text{CPU使用率}\ (\%) = \frac{\text{実行時間}}{\text{周期}} \times 100$$

e.g. 60% (C=60ms, T=100ms)の使用率を設定



CABI 利用方法

- Accounting Object（AO: アカウンティングオブジェクト)
  - 単数、複数のプロセスを統一的に扱うためのインターフェイス
    - AO にCPU使用率(%) を設定
      - 周期 (T), 実行時間(C) 、終了動作(block/signal) 、AO毎に識別子(object_id)
    - AOに制限 / 資源保証したいプロセスをバインドする



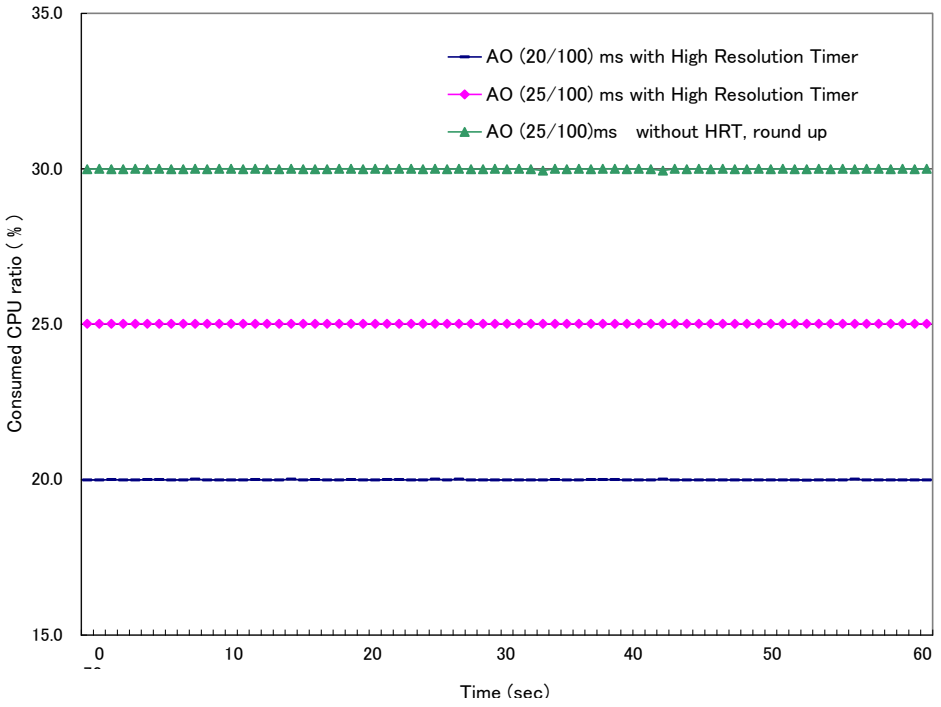
CABI による制御

- 試験環境
  - CPU　　Celeron 300MHz
  - MEMORY　128MB

- 基本的な API　の実行時間
  - ユーザ ⇔ カーネル

API	User (μ sec)
create	39.3
bind	35.1
unbind	9.8
destroy	24.0
set	4.9
get	3.5

- 検証結果
  - 20 % (20/100ms) に制限した場合
  - 25 % (25/100ms) に制限した場合
  - 高解像度タイマの有無検証



- 与えられた比率で、正確にアカウンティングを行う

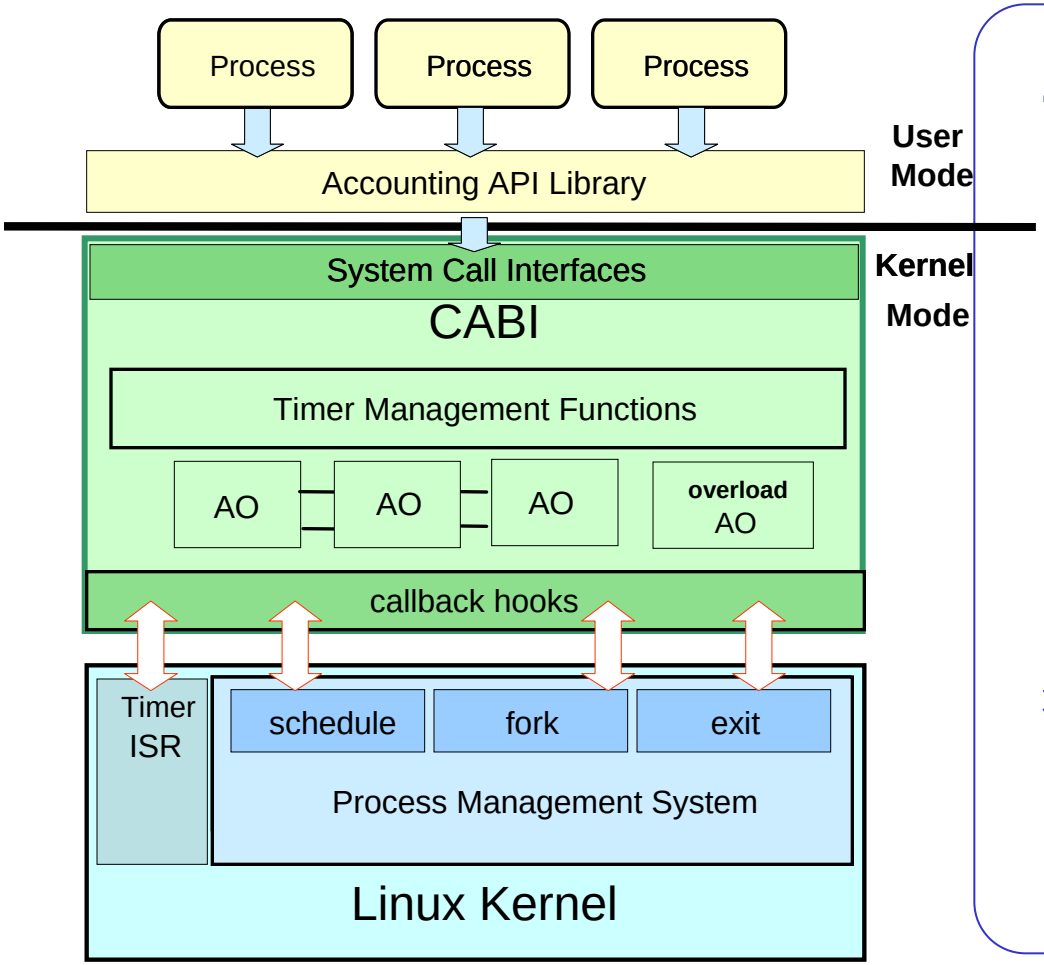
アカウンティングシステムが提供するAPI

- システムコール API

システムコールインターフェイス	内容
int cabi_account_create (obj_id, &object_attr)	オブジェクトの作成
int cabi_account_destroy (obj_id)	オブジェクトの削除
int cabi_account_pid_bind (obj_id, &object_attr)	オブジェクトにプロセスをバインド
int cabi_account_pgid_bind (obj_id, &object_attr)	オブジェクトにプロセスグループをバインド
int cabi_account_set(obj_id, &object_attr)	オブジェクトの設定値の変更
int cabi_account_get(obj_id, &object_attr)	オブジェクトの設定値の取得

- ツールも充実
  - 実行するアプリケーションをそのまま指定
    - exec_bind
  - /proc/cabi/id/status
    - /proc 以下に統計情報、現在制御されているプロセスの情報などが表示される

システムアーキテクチャ



単純さ
- シンプルな構造とする
- インターフェイス
  ユーザ/カーネル

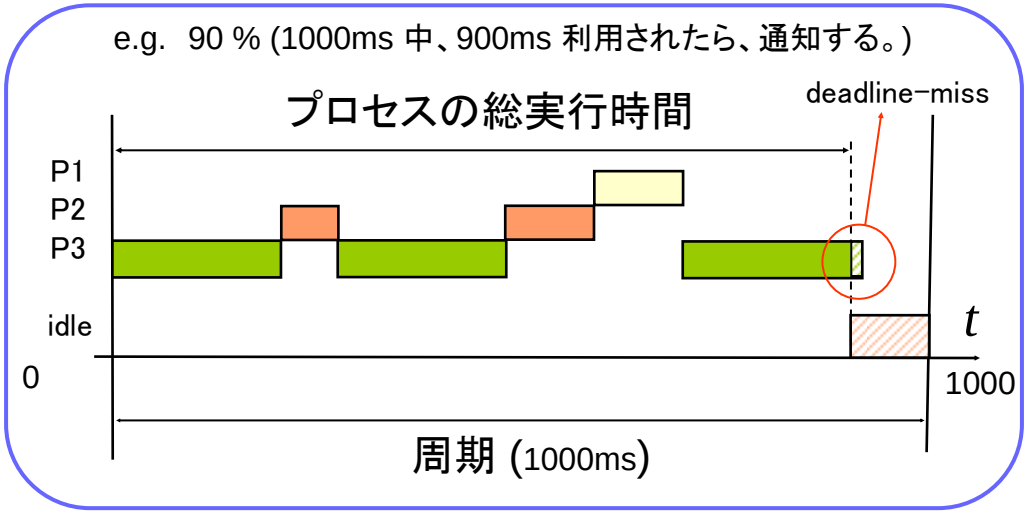
正確さ
- 独自の時間管理機構
- 高解像度タイマ 利用

独立性
- スケジューラ無変更
- カーネルに最小限のフック

応用例 1: オーバーロード監視機能

- オーバーロードとは
  - システムの負荷が高まり，不安定な状態
    - RTプロセス：デッドラインをミス, TSプロセス： 実行遅延
- オーバーロード通知機能
  - 全体が指定した比率(%) を超えると通知
    - idle プロセスを AO にバインド
      - idle プロセスの監視（システム全体の使用率と逆の比率で動作）

API
- Overload 用の AO の作成，削除
- overload_create (&object_attr, pid)
- overload_destroy (void)



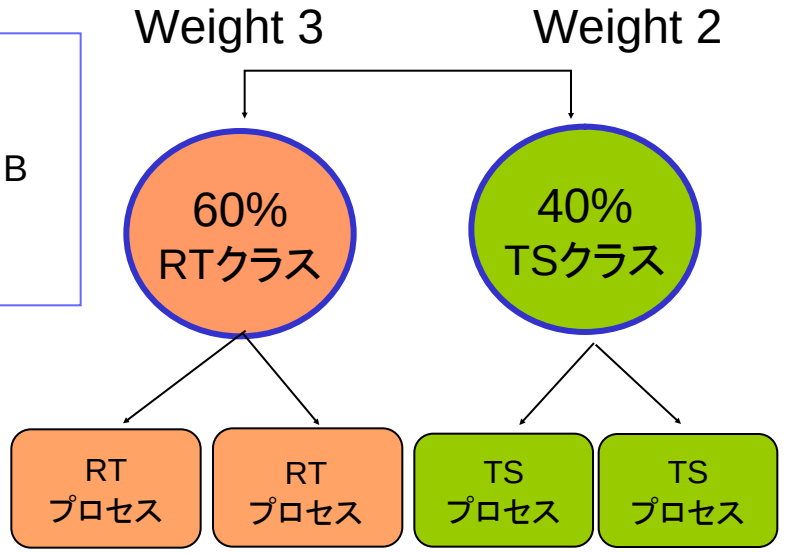
応用例 2：クラス別アカウンティング

- リアルタイムのプロセス全てに実行比率で制限をかけたい
  - 全ての RT プロセスを取りこぼしなく制限する

- RT/TS の各クラスにウェイト（比率）を指定する
  - リアルタイムクラス（RT），タイムシェアリング（TS）クラス
    - e.g) リアルタイムクラス、タイムシェアリングクラスの比率(2:3)
      RT: 40%，TS : 60 %

B ：全体の資源量
Bi ：クラス i が受取る資源量
ri ：i 番目のウェイト

$$B_i = \frac{r_i}{\sum_{j=1}^{n} r_j} \times B$$



- メリット
  - 100% で調整を行うため，確実にクラスに割り当てられた資源が確保される
  - 逆のクラスからみた場合、制限となる

リファレンス実装の公開

- ppc, sh, mips, arm, x86へ対応
- Linux 2.6 (arm, sh) 対応
- Sourceforge にて開発バージョン公開
  - http://www.sourceforge.net/cabi/
  - http://sourceforge.jp/

- Emblix
  - http://www.emblix.org

Priority Boost の取り組み

日立製作所
Lineo Solutions, Inc.

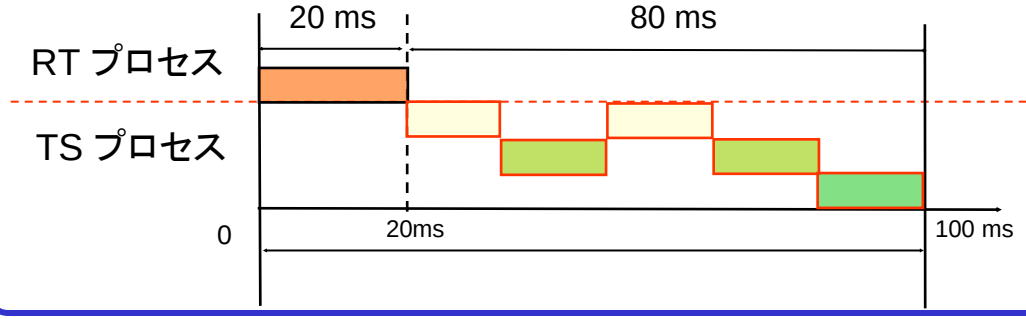
CE Linux Forum in 第9回 組込みシステム開発技術展（ESEC)
2006

プロセス下限保証の必要性

- CABI 機能
  - RT プロセスの実行時間保証（資源予約）
    - オーバーロード時の保証は予測可能性の保証のみ
  - TS プロセスの実行時間保証(下限保証) は行っていない
    - Linux では、ユーザーモードスレッドは通用TSクラスを利用

実行中（TASK_RUNNING）状態のTS プロセスが多数存在していると、RTプロセスをブロックしていても実行権が渡されるプロセスは不確定

e.g. 20 % で RT プロセスを制限、残り 80% 利用できるはずだが。。



- RT以外のプロセスの実行順序保証（下限保証)を行うには ？

最低割り当て時間に着目したプロセスのCPUリソース管理機能提供

下限保証機能の実装検討

- TS (NORMAL)プロセスへの確実なCPUの割り当て

TS プロセスを一時的にRTプロセスとする
(Boost priority)

- Priority　Boost
  - CABI のアカウンティング・オブジェクト(AO) でプロセス指定
  - Boost 周期は AO の定周期時間を利用
  - Boost 時間は AO の実行時間を利用
  - Boostする RT の priority 指定は新規に追加する

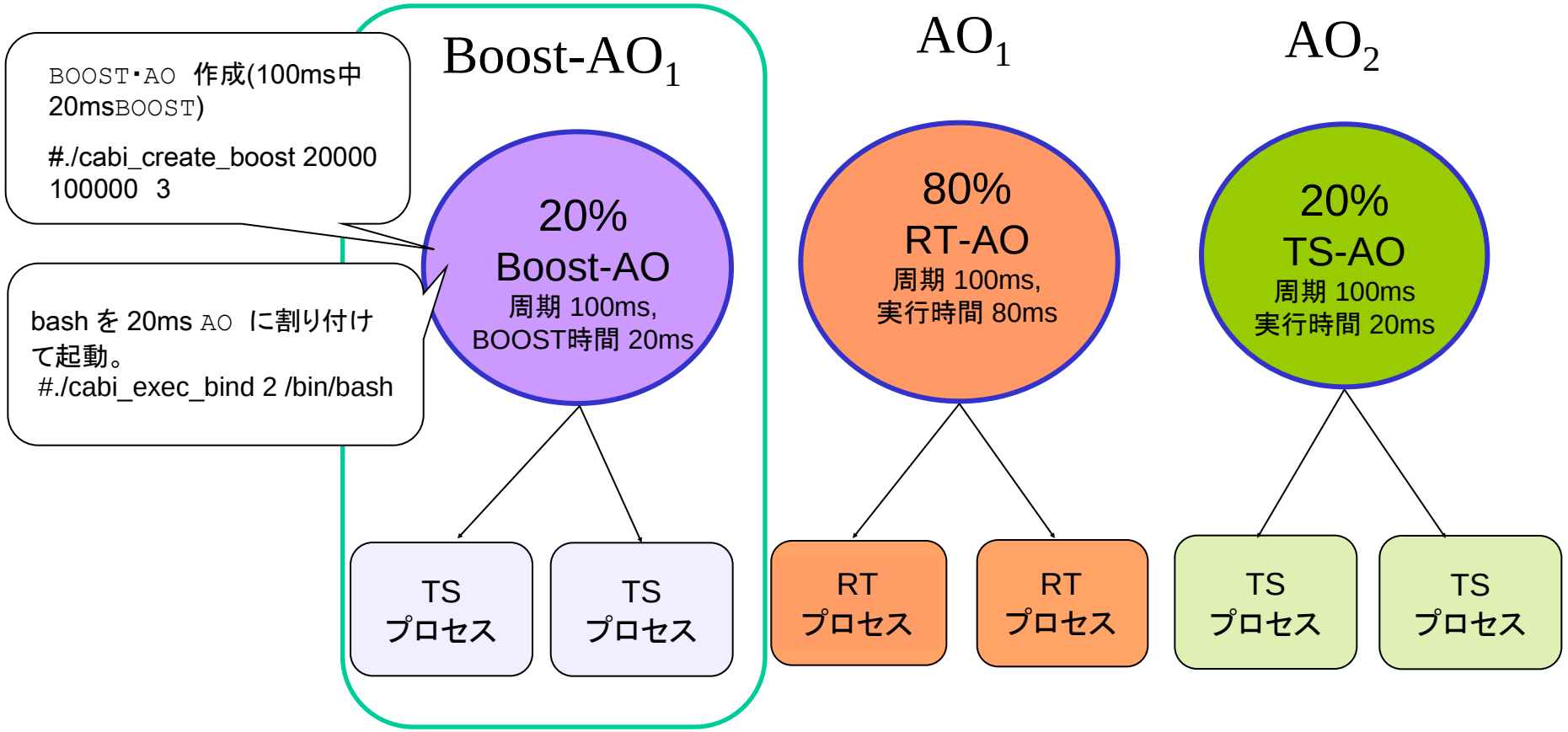
Boost 実現におけるCABIへの新規追加点

- CABIの設定値変更

設定値	CABI	Priority Boost
Boost priority		Boostt時のpriority指定（新規追加）
実行時間	アカウンティングオブジェクトの最大実行時間	Boostしたプロセスの最大Boost時間（最小保証値）
周期時間	アカウント・オブジェクトの周期	Boost するプロセスの周期時間

Boosting Accounting Object

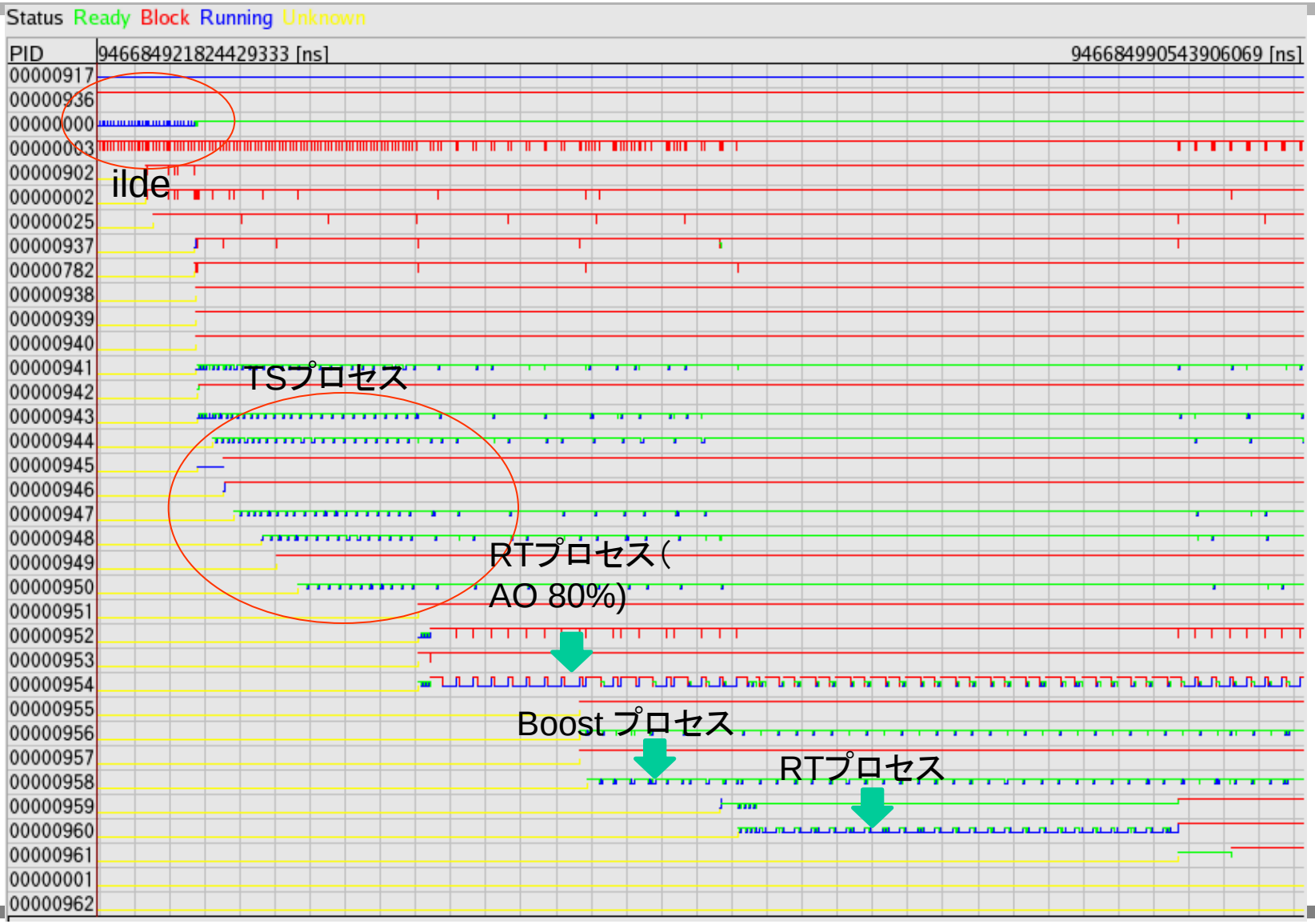
- Boost を行うアカウンティング・オブジェクトの追加



Demo

- 環境
  - Kernel 2.6.9
  - SH 7751R
- ペンギン
  - 黒: TS (タイムシェアリング)プロセス/ ノーマル
  - 青: CABI プロセス 80 % (800ms/10000ms)
  - 緑: RT (リアルタイム) プロセス
  - 赤: Boost プロセス

評価


Status Ready Block Running Unknown
PID
946684921824429333 [ns]
946684990543906069 [ns]
00000917
00000936
00000000
00000003
00000902
00000002
00000025
00000937
00000782
00000938
00000939
00000940
00000941
00000942
00000943
00000944
00000945
00000946
00000947
00000948
00000949
00000950
00000951
00000952
00000953
00000954
00000955
00000956
00000957
00000958
00000959
00000960
00000961
00000001
00000962
ilde
TSプロセス
RTプロセス(
AO 80%)
Boost プロセス
RTプロセス

リソースマネジメントワーキンググループ

- 発足　：　2004年6月
- 目的　：
  - Linux上のQoSを支援するための標準API の決定
  - 標準化の仕様書の策定
    - Emblix 標準化指標に準拠したQoSシステムの推奨と推進
- 参加　：
  - モンタビスタ ソフトウエア ジャパン(株)，兵庫県立大学，岩崎通信機(株)，(株)アックス，ヤマハ(株)，三菱電機(株)，名古屋大学，(株)KDDI研究所，富士通(株),早稲田大学

目的

- リソースマネジメントに関する標準化 API の策定
  - 様々なベンダーで利用できる標準化されたインターフェイスを定めます
- 方針
  - 実用性　：現状のLinuxの制限を考慮
  - 汎用性　：多様なアーキテクチャ上での動作
- 特徴
  - 割合による制御

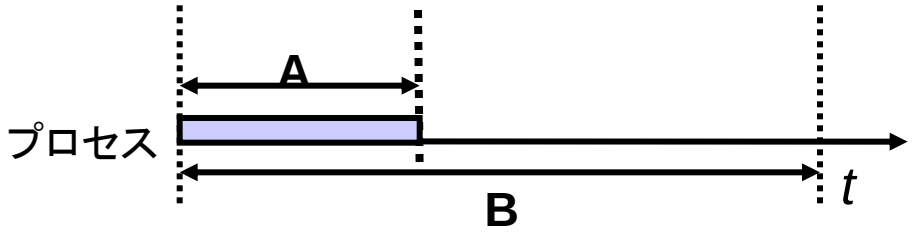
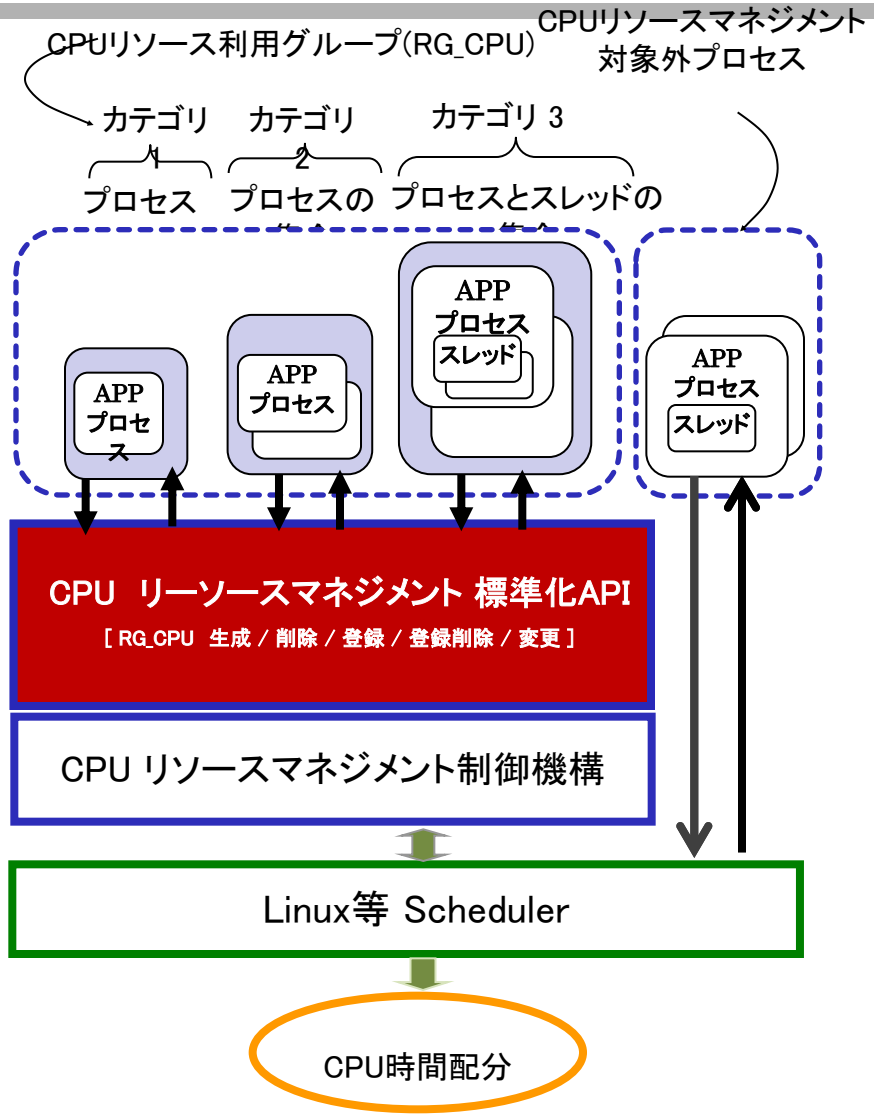


図　割合によるプロセスの制御

【Linux の制限の考慮】

Linux上でのリアルタイム性能は、カーネルの構造上、時間制約に関する定義を厳密に行う事が困難です。そのため、ワーキンググループでは、実装に対する負担を軽減するため、実装者が自由に解釈できる2つのパラメータを指定し、その割合で制御するよう規定を行いました。

標準化仕様内容



■ カテゴリ区分

カテゴリ	制御対象		セキュリティ	機能リソースグループ	
	対象	構成メンバ		自動消滅	自動継承
カテゴリ1	RG_CPU	単一プロセス	無し	有り	無し
カテゴリ2	RG_CPU	プロセス集合	権限設定	無し	有り
カテゴリ3	RG_CPU	プロセス, スレッド集合	権限設定	無し	有り

【解説】
本仕様では、CPU リソースマネジメントの機能仕様を、対象システムの用途や目的、実装の複雑さ等により、機能レベルが選択できるように3種類のカテゴリを用意する。

終わりに

- 資源管理について
  - 組込みシステムの制約
  - Linux の資源管理の制限
  - 提案するCABIシステムについて
- 応用例の紹介
  - クラススケジューリング、資源保護、オーバーロード管理
  - Priority Boost (日立製作所、Lineo Solutions, Inc)
  - リソースマネジメントワーキンググループ(Emblix)
    - 標準化仕様書

についてご紹介致しました。ご意見、ご質問等ありましたら、宜しくお願いいたします。