ハードリアルタイムサポートへ向けて

―ITRONハードリアルタイムサポート研究会中間報告―

16th July 1997

高田 広章
東京大学 理学部／ITRON専門委員会

hiro@is.s.u-tokyo.ac.jp
http://tron.um.u-tokyo.ac.jp/~hiro/

ソフトウェア部品の必要性と標準化

ソフトウェア部品の必要性

- 組み込みシステムの複合化・複雑化
  例) デジタルカメラ
  - システム内でカーネルの占める比率の低下
    - ITRONプロジェクトにおける標準化対象を拡大
      ～ カーネル仕様から周辺仕様へ ～
- ハードウェア部品のソフトウェア化
  例) ソフトモデム, JPEG, 音声圧縮/解凍

↓

標準化による部品の流通促進

標準化へ向けての２つの課題

(a) ソフトウェア部品の開発・流通を促す**前提条件の整備**

→ ハードリアルタイムサポート研究会

(b) ソフトウェア部品の**インタフェースの標準化**

- ソフトウェア部品の種類毎に標準化が必要

  → Embedded TCP/IP 技術委員会

  - 組み込みシステムに適した TCP/IP API の標準化を行う
  - TCP/IP API として広く使われているソケットインタフェースは組み込みには不適

  → その他についても重要分野から順に標準化

開発・流通を促す条件の整備

(1) カーネル仕様の標準化レベルを上げる

- 「弱い標準化」のコンセプト
  - これまでは成功
  - ソフトウェアのポータビリティを阻害
- ➡「弱い標準化」の利点との両立を目指す

(2) ハードリアルタイム性を持ったソフトウェア部品をサポートする

例) ソフトモデム, MPEG解凍

- アプリケーションシステムに組み込んだ場合にリアルタイム性を**保証**する仕組み
- 複数の部品を併用した場合の**保証**の仕組み

ハードリアルタイム性とは？

- リアルタイムシステムの中でも、特に厳格に時間制約を守ることを要求されるシステム
  - 「ハード」… 厳しい
  - **保証**が求められる

    ↓

    システムの動作が**予測可能**であること
- 単に実行速度が速いことや、レスポンス時間が短いことを意味するものではない

ITRONハードリアルタイムサポート研究会

位置づけ

- ITRON専門委員会が主催する研究会
- 貢献する意志のある方なら誰でも参加可能
- 成果はオープンに

活動の経緯

- 1996年11月にスタート (第1回ミーティング)
- 研究会への登録者は約40名
- 1997年4月までに5回のミーティング
  - ハードリアルタイムに関する理論の調査・紹介
  - 他のリアルタイムOS やツールの調査
  - アプリケーションからの要求事項の洗い出し

ワーキンググループ活動

- 成果の取りまとめへ向けて活動形態を見直す
- 1997年5月にスタート，毎月1回の会合
- 研究会登録者は自由に参加可能

アプリケーション構築ガイドライン作成WG

- リアルタイム性を持ったアプリケーションをリアルタイムOS上に構築する場合のガイドライン
- ハードリアルタイム性を持ったソフトウェア部品を用いる場合のリアルタイム性の保証の枠組み

➡ アプリケーション構築ガイドラインを作成して公表

カーネル仕様検討WG

➡ ITRONカーネル仕様の改良をITRON専門委員会に提案

カーネル仕様検討WG での検討内容

標準化レベルを上げる

- スタンダードプロファイルを定義
  - ソフトウェア部品の流通性を確保したいカーネルの実装が持つべき機能セット
  - 流通性を持たせたいソフトウェア部品が使って良い機能セット
- サブセット，拡張は自由に許される

ハードリアルタイムサポートに必要な機能の追加

- 優先度逆転の防止
  例) 優先度継承セマフォ
- オーバーランの検出と例外処理

ハードリアルタイムサポートのための性能指標の標準化

- **リアルタイム性の保証に必要な性能指標**の標準化

  例) タスクスイッチ時間の定義

カーネル構成記述の標準化

- 静的に生成するタスクや同期・通信オブジェクトなどの記述方法の標準化
- カーネルAPI を大文字で記述すると静的な実行を表す

  例)
```
CRE_TSK(TASK1, { 0, 1, func1,    … });
DEF_INT(EXTINT5, { 0, handler5 });
```

デバイスドライバの標準化

- デバイスドライバの呼び出し方法の枠組みの標準化
- デバイスドライバの構築方法のガイドライン

アプリケーション構築ガイドライン

- リアルタイム性を持ったアプリケーションをリアルタイムOS上に構築する場合
  - **タスク分割**の方法
  - **優先度**の付け方

  ➡ リアルタイム性確保の観点から1つの指針
- ソフトウェア部品を用いる場合のリアルタイム性の保証

  ソフトウェア部品の提供者

  … ソフトウェア部品の性質 (使用リソース, リアルタイム制約) を**パラメータ化**して提供

  ソフトウェア部品のユーザ

  … 提供された**パラメータ**を用いてシステム全体のリアルタイム性を検証

  ➡ パラメータの標準化

想定する読者

- リアルタイム制約をシステム設計時に考えず、アドホックな方法で設計している組み込みシステム設計者

  ➡ *ぴったりの読者*

- ハードリアルタイム性を持ったソフトウェア部品の提供者とユーザ

理論的枠組み

- **Rate Monotonic Analysis (RMA)** の手法を適用
  - 現在のリアルタイムOS と相性が良い
  - 現時点では最も完成度が高い理論体系
  - 市販のスケジュール可能性検証ツールあり

Liu and Layland の結果

- リアルタイムスケジューリング理論の古典的な結果

タスクセットに対する仮定

1. 周期タスク (periodic task) のみを考える
2. 各タスクの最大実行時間はあらかじめわかっている
3. 各タスクは周期の始めに実行可能に
4. 各タスクのデッドラインは周期の終わり
5. タスク間に同期・通信がない
6. タスクは自ら実行を中断することはない
7. タスク切替等のオーバヘッドは考えない
8. プロセッサが 1つのケースのみ考える

⇒ プロセッサに対する負荷が変動しないタスクセット

Rate Monotonic Scheduling

周期の短いタスクから順に高い優先度を割り付ける

- タスクセットがスケジュール可能になる**十分条件**

$$\left(\begin{array}{c}\text{システムの}\\ \text{最大負荷}\end{array}\right) = \sum_{j=1}^{n} \frac{C_j}{T_j} \leq n\,(2^{1/n} - 1)$$

n	右辺
2	0.83
3	0.78
∞	0.69

n : タスク数
T_j : 優先度が j 番目のタスクの周期
C_j : 優先度が j 番目のタスクの最大実行時間

- **最適な静的優先度割り付け**
- より正確な判定を可能にする**必要十分条件**の式も与えられている

前提条件を緩める

1. 非周期タスクを考える

- 最小起動間隔を周期とする周期タスクと考えてよい

4. タスクのデッドラインが周期の終わりに一致しない

- rate monotonic に代えて deadline monotonic を使う

デッドラインまでの時間が短いタスクから順に高い優先度を割り付ける

※ ただし、スケジュール可能性の判定条件は変わる

5. タスク間に同期・通信がある

- 優先度逆転を考慮してスケジュール可能性を判定

7. タスク切替のオーバヘッドを考える

- タスク切替 2回分を各タスクの最大実行時間に加える

ガイドラインにおけるアプリケーション構築の流れ

(1) アプリケーションシステムを構成する各処理の性質の抽出・整理

- 各**処理**に対して**リアルタイム性の解析に必要なパラメータ**を洗い出す
- 体系的に洗い出すためのガイドライン
- アプリケーションの処理への分割については、このガイドラインでは触れない

パラメータ(検討中)

- 処理の起動タイミング (起動要因，最大起動頻度)
- 処理の最大実行時間
- 処理に対するリアルタイム制約 (デッドライン，デッドラインの種別)

(2) タスクの分割方法・優先度の決定とスケジュール可能性の評価

- 各処理をどのタスクで実現するか？ (タスク分割)
- 処理のパラメータをタスクのパラメータに読み換え
- 優先度の決定 (deadline monotonic が基本)
- スケジュール可能性の解析 (式に当てはめる)
- パラメータの与えられたソフトウェア部品は、このフローの中間に合流する

(3) チューニング手法と QOS制御の適用

- 前のステップでスケジュール可能にならない時や必要となる資源が多すぎる場合に、このステップを適用
- このステップを適用したら (2) へ戻る

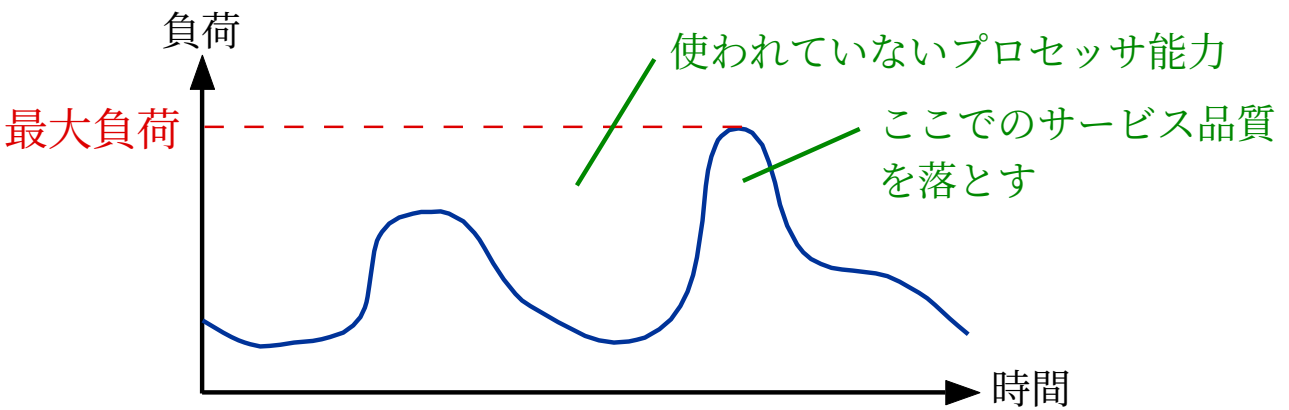
QOS制御とその必要性

※ QOS (Quality of Service) ＝ サービスの品質

- システムに対する負荷が想定した最大値を越えた時には、サービスの品質 (計算の精度, 制御の精度) を落とすことで負荷を下げる

➡ プロセッサの能力を有効に利用

- システムに対する負荷は変動する



ガイドライン中で考慮されるべきその他の事項

- タスク間の排他制御　　→ セマフォ, 優先度継承, etc.
- バッファによるシステム負荷の分散　→ メイルボックス
- タスクの起動条件の組合せ　→ イベントフラグ
- モードの扱い
- I/O待ちの扱い
- タスクと割り込みハンドラの使い分け
- タスクとライブラリの使い分け
- 最大負荷の不正確な見積りへの対処
- 最大実行時間の決定/計測
- 試験・検証との関連
- **適用例**

今後の予定

- 研究会の第一次の結果を年内 or 年度内には出す
  - アプリケーション構築ガイドライン 第0版
  - ITRONカーネル仕様に対する提案

ITRON専門委員会でカーネル仕様の改訂を議論

μITRON3.1仕様 (または、4.0)