Extensions to Liu & Layland Scheduling Models For Rate-Based Execution

Kevin Jeffay
Department of Computer Science
University of North Carolina at Chapel Hill
jeffay@cs.unc.edu
September 23, 1997

Outline

- Rate Based Execution: The case against Liu & Layland style models of real-time computing
- A Liu & Layland extension for rate-based execution?
- Fluid-flow models of resource allocation for real-time services
- Proportional share CPU scheduling
- On the duality of proportional share and traditional Liu & Layland style resource allocation
Extensions of the Liu & Layland Model

Objectives

- Support notions of execution rate that are more general than periodic execution
- Support integrated real-time device and application processing
- Support responsive non-real-time computing

Rate-Based Computing

Concept

- Schedule tasks at the average rate at which they are expected to be invoked
  - Make buffering a first-class concept in the model
  - Understand the fundamental relationships between feasibility, latency, and processing rate

- Develop a model of tasks wherein:
  - Tasks complete execution before a well-defined deadline
  - Tasks make progress at application-specified rates
  - No constraints are placed on the external environment
Rate-Based Computing
Beyond periodic & sporadic models

◆ An event-based model — *rate-based execution*
  » Process make progress at the rate of processing \( x \) events every \( y \) time units, each event is processed within \( d \) time units

◆ A time-sharing model — *proportional share resource allocation*
  » Processes make progress at a precise, uniform rate — as if executing on a dedicated processor with \( 1/n^{th} \) original capacity

Rate-Based Computing
Overview of results

◆ We will demonstrate that
  » the theory of dynamic priority task systems extends nicely to handle rate-based execution
  » unless constraints are placed on the external environment, *no* static priority scheduling algorithm can guarantee that a set of rate-based tasks execute in real-time
Rate-Based Execution

Formal model

- Process make progress at the rate of processing $x$ events every $y$ time units, each event is processed within $d$ time units.

- For task $i$ with rate specification $(x_i, y_i, d_i)$, the $j^{th}$ event for task $i$, arriving at time $t_{i,j}$, will be processed by time

\[
D(i, j) = \begin{cases} 
    t_{i,j} + d_i & \text{if } 1 \leq j \leq x_i \\
    \max(t_{i,j} + d_i, D(i, j-x_i)+y_i) & \text{if } j > x_i
\end{cases}
\]

- Deadlines separated by at least $y$ time units
- Deadlines occur at least $y$ time units after a job is released

Rate-Based Execution

Example: Periodic arrivals, periodic service

- Task with rate specification $(x = 1, y = 2, d = 2)$

\[
D(i, j) = \begin{cases} 
    t_{i,j} + d_i & \text{if } 1 \leq j \leq x_i \\
    \max(t_{i,j} + d_i, D(i, j-x_i)+y_i) & \text{if } j > x_i
\end{cases}
\]
Rate-Based Execution

Example: Periodic arrivals, \textit{deadline} \neq \textit{period}

- Task with rate specification \((x = 1, y = 2, d = 6)\)

\[
D(i, j) = \begin{cases} 
  t_{i,j} + d_i & \text{if } 1 \leq j \leq x_i \\
  \text{MAX}(t_{i,j} + d_i, \ D(i, j-x_i)+y_i) & \text{if } j > x_i
\end{cases}
\]

Rate-Based Execution

Bursty arrivals

- Task with rate specification \((x = 1, y = 2, d = 6)\)
Rate-Based Execution

Bursty arrivals

- Task with rate specification \((x = 3, y = 6, d = 6)\)

Rate-Based Execution

Comparison

Rate specification
\((x = 1, y = 2, d = 6)\)

Rate specification
\((x = 3, y = 6, d = 6)\)
Using RBE Tasks
What problems do they solve?

◆ Provides better response time for non-real-time activities by integrating application-level buffering with the system run queue.

![Diagram showing network reception, display, and a processing pipeline.](image)

Rate specification \((x = 1, y = 2, d = 6)\)

Using RBE Tasks
What problems do they solve?

◆ Provides a more natural way of modeling inbound packet processing of fragmented messages.

![Diagram showing acquire and display with display initiation time.](image)

Rate specification \((x = 3, y = 6, d = 6)\)
Rate-Based Execution

Conjectures

- Captures the essence of real-time computing on the desktop
- Provides a framework for tuning application performance to network performance
- Minimizes response time for non-real-time activities
- One can precisely characterize the conditions under which a rate-specification is realizable

Is it new?

- RBE is an amalgam of three technologies
  - the Synthesis operating system (Columbia)
    - software phased-lockedloops
  - the Dash operating system (Berkeley)
    - a “leaky bucket” model applied to operating system processes
    - processes characterized by an average rate and a “burst” size
  - the YARTOS real-time operating system (UNC)
    - the producer/consumer data-flow model of computation
- Novel aspects
  - separation of throughput and response time specifications
  - provably real-time
A Theory of Rate-Based Execution

Goal and basic concepts

- The goal is to develop conditions on model parameters which, if satisfied by a set of tasks, imply that every job of every task will complete execution before its deadline.

- Feasibility and schedulability analysis
  - Feasibility — conditions under which a set of tasks are guaranteed to execute correctly: an absolute measure of correctness
  - Schedulability — conditions under which a set of tasks are guaranteed to execute correctly when scheduled by a given algorithm: a relative measure of correctness

Review

- Schedulability analysis of periodic tasks \( T_i = (c_i, p_i) \)
  - Static priority assignment: “Level \( i \) busy period analysis”
    \[
    \forall i, 1 \leq i \leq n, \exists L, 1 \leq L \leq p_i: L \geq \sum_{j=1}^{i} \left\lfloor \frac{L}{p_j} \right\rfloor c_j
    \]
  - Dynamic priority assignment: “Processor demand analysis”
    \[
    \forall L, L > 0: L \geq \sum_{i=1}^{n} \left\lfloor \frac{L}{p_i} \right\rfloor c_i
    \]
Feasibility analysis of periodic tasks with \textit{deadline \neq period}

- Under earliest deadline first scheduling

\[ \forall L, L > 0: \quad L \geq \sum_{i=1}^{n} \left( \frac{L - d_i + p_i}{p_i} \right) c_i \]

Consider a set of \textit{RBE} tasks with rate specification \((x, y, c, d)\)

- Feasibility conditions are precisely the same as for periodic tasks

\[ \forall L, L > 0: \quad L \geq \sum_{i=1}^{n} \left( \frac{L - d_i + y_i}{y_i} \right) x_i c_i \]
What is the maximum number of jobs of an RBE task with deadlines in an interval \([0, L]\), \(L \geq d\)?

\[
x + \left(\frac{L - d}{y}\right)x = \left(\frac{L - d}{y} + 1\right)x = \left(\frac{L - d + y}{y}\right)x
\]

When scheduled by an EDF scheduler

\[
\sum_{i=1}^{n} \left(\frac{t_{d} - t_{0} - d_i + y_i}{y_i}\right)x_i c_i > t_{d} - t_{0}
\]
A Theory of Rate-Based Execution
On the relationship to periodic tasks

What is the maximum number of jobs of an RBE task with deadlines in an interval \([0, L]\), \(L \geq d\)?

» It can never be greater than the corresponding periodic task

» In the RBE model, “early” task invocations receive the same deadlines they would have had they been invoked “on time”

But can’t an RBE task be modeled as \(x\) instances of a periodic task (with some appropriate precedence relationship between instances)?
A Theory of Rate-Based Execution

A corollary on static priority scheduling

- Under a static priority scheduling scheme, the processor demand in any interval can be unbounded
  - thus event driven, rate-based execution is not possible under static priority scheduling schemes

A Theory of Rate-Based Execution

Feasibility analysis under preemption constraints

- When preemption is allowed at arbitrary points, feasibility conditions are precisely the same as for periodic tasks

\[
\forall L, L > 0: \quad L \geq \sum_{i=1}^{n} \left( \frac{L - d_i + y_i}{y_i} \right) x_i c_i
\]

- The same holds for non-preemptive systems

\[
\forall i, \quad 1 < i \leq n \quad \forall L, \quad d_1 < L < d_i \quad L \geq c_i + \sum_{j=1}^{i-1} \left( \frac{L - 1 - d_j + y_j}{y_j} \right) x_j c_j
\]
A Theory of Rate-Based Execution
Feasibility analysis under preemption constraints

\[ \forall i, \ 1 < i \leq n \ \Rightarrow \ L \geq c_i + \sum_{j=1}^{i-1} \frac{\left\lfloor \frac{L - 1}{p_j} \right\rfloor c_j} \]

\[ \forall L, \ p_1 < L < p_i \]

\[ \forall L, \ d_1 < L < d_i \]

\[ L \geq c_i + \sum_{j=1}^{i-1} \frac{L - 1 - d_j + y_j}{y_j} x_j c_j \]
A Theory of Rate-Based Execution

Summary

◆ There exists an efficient (pseudo-polynomial time) decision procedure for determining both feasibility and schedulability
  » If processor utilization less than 1.0

◆ The earliest-deadline-first scheduling algorithm is optimal

◆ The feasibility and schedulability of a set of “periodic tasks” was never inherently tied to the fact that tasks are invoked strictly periodically
  » The only requirement is that deadlines be separated by at least a constant amount of time

Rate-Based Execution

Applying the theory

◆ Kernel issues
  » RBE task implementation
  » admission control
  » rate enforcement
  » rate negotiation

◆ Application issues
  » rate specifications
  » mechanisms for rate feedback and adaptation
Applying the Theory
Latency comparison (latency vs. CPU utilization)

RBE Execution

Periodic Server

Applying the Theory
Non-real-time task response time comparison

75% Real-time task utilization

50% Real-time task utilization

25% Real-time task utilization
Rate-Based Execution

Warts

◆ Requires extensive kernel modifications to support
  » Defining a new, event-based programming model

◆ Intel: *This is really great stuff. Will it work in Windows?*