Rate-Based Resource Allocation Models for Real-Time Computing

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

Steve Goddard  
Computer Science & Engineering  
University of Nebraska – Lincoln  
goddard@cse.unl.edu

http://www.cs.unc.edu/Research/dirt

Rate-Based Resource Allocation

Overview

◆ The problem:
  » How to allocate resources in an environment wherein…
     ◆ Work arrives at well-defined but highly variable rates
     ◆ Tasks may exceed their execution time estimates
  » … and still guarantee adherence to deadlines

◆ The thesis:
  » Static priority scheduling is the wrong tool for the job (existing task models are too simplistic)
  » Rate-based scheduling abstractions can simplify the design and implementation of many real-time systems and improve performance and resource utilization

Rate-Based Resource Allocation

The case against static priority scheduling

◆ Static priority scheduling in general, and Rate Monotonic scheduling in particular, dominates in the real-time systems literature
  » VxWorks, VRTX, QNX, pSOS, LynxOS all support static priority scheduling

◆ Does one size fit all?
  » “When you have a hammer, everything looks like a nail”

◆ Problems with static priority scheduling
  » Feasibility is dependent on a predictable environment and well-behaved tasks.

The Case Against Priority Scheduling

Example: Display-side multimedia processing

◆ The problem: Receive frames from the network and deliver to a display application so as to ensure...
  » Continuous playout
  » Minimal playout latency

◆ The theory: Multimedia is easy — it’s periodic!
  » Apply existing theory of periodic or sporadic tasks
Nothing is periodic in a distributed system!

The effects of distributed systems pathology:
- Variable message transmission times
- Out-of-order message arrivals
- Lost & duplicate messages

Packets fragmented in the network must be reassembled
- Messages have deadlines, packets do not
- Applications know about messages, operating systems do not

The Case Against Priority Scheduling

Example: Signal processing data flow graphs

Node u has a period of 3
Node v has a period of 5

U
produce = 3
consume = 5

Node v has a period of 5

V
produce = 4

Node v has a period of 5

W
produce = 4
consume = 4

Rate-Based Computing

Approaches

- Extend the Liu and Layland model of real-time tasks to allow the expression of real-time rates
  - Hierarchical “server-based” scheduling — Create a “server” process that is scheduled as a periodic task and internally schedules the processing of aperiodic events
  - Event-based scheduling — Process aperiodic events as if they were generated by a virtual “well behaved” periodic process

- Adapt “fluid-flow” models of resource allocation developed in the networking community for bandwidth allocation to CPU scheduling
  - Provide a “virtual processor” abstraction wherein each task logically executes on a dedicated processor with \(1/f(n)\) the capacity of the physical processor
An Event-Based Rate Model
The Rate-Based Execution (RBE) model

- Tasks make progress at the rate of processing $x$ events every $y$ time units and each event is processed within $d$ time units (in the best case).
- 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

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

- $D(i, j)$ gives the earliest possible deadline for the $j^{th}$ instance of task $i$ ($\geq t_{i,j} + d_i$)

The RBE Task Model
Example: Periodic arrivals, periodic service

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

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

- Deadlines separated by at least $y = 2$ time units
- Deadlines occur at least $d = 2$ time units after a job is released

The RBE Task Model
Example: Periodic arrivals, deadline $\neq$ period

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

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

- Deadlines separated by at least $y = 2$ time units and occur at least $d = 6$ time units after a job is released

The RBE Task Model
Bursty arrivals

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

- Deadlines separated by at least $y = 2$ time units and occur at least $d = 6$ time units after a job is released
The RBE Task Model

Bursty arrivals

- Task with rate specification \((x = 3, y = 6, d = 6)\)
  - Deadlines separated by at least \(y = 6\) time units and occur at least \(d = 6\) time units after a job is released.

The RBE Task Model

RBE features/properties

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

The RBE Task Model

Comparison of rate specifications

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

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

The RBE Task Model

RBE features/properties

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

Rate specification \((x = 3, y = 6, d = 6)\)
The RBE Task Model

**RBE features/properties**

- Provides isolation from arrival rates that exceed the rate specification
  - (But does not provide isolation from tasks exceeding their stated execution time)

**Fluid Flow Resource Allocation**

**Proportional share resource allocation**

- Tasks are allocated a share of the processor’s capacity
  - Task $i$ is assigned a weight $w_i$
  - Task $i$’s share of the CPU at time $t$ is
    $$f_i(t) = \frac{w_i}{\sum_{j \neq i} w_j}$$

- If tasks’ weights remain constant in $[t_1, t_2]$ then task $i$ receives
  $$S_i(t_1, t_2) = \int_{t_1}^{t_2} f_i(t) \, dt = \frac{w_i}{\sum_{j \neq i} w_j} (t_2 - t_1)$$
  units of execution time in $[t_1, t_2]$
Proportional Share Resource Allocation
Task scheduling metrics & goals

Schedule tasks so that their performance is as close as possible to that in the fluid system.

Why is fluid allocation important?
» What about real-time allocation?!

Approximating Fluid Allocation
Why is this so important?

- Fluid allocation implies real-time progress.
- Weights are used to allocate a relative fraction of the CPU’s capacity to a task:
  \[ f_i(t) = \frac{w_i}{\sum_j w_j} \]
- Real-time progress requires a constant fraction of the CPU’s capacity:
  \[ \forall t, \quad f_i(t) = \text{execution cost}_i \times \text{execution frequency}_i \]
  » If a task must execute for 16 ms every 33 ms then allocating \( f = 0.5 \) ensures real-time execution.
- Thus real-time performance can be achieved by adjusting weights dynamically so that the share remains constant.

Proportional Share Resource Allocation
Real-time scheduling example

- Periodic tasks allocated a share equal to their processor utilization.
  » Round-robin scheduling with infinitesimally small quantum.
  \[ T_1 = (2, 8), \quad T_2 = (3, 6) \]
  » With unit-sized quantum.

Goal: Schedule tasks so that their performance is as close as possible to that in the fluid system.

Define the allocation error for task \( i \) at time \( t \) as:
\[
\text{lag}_i(t) = \left\{ \begin{array}{l}
\text{allocation the task would have received in the fluid system} \\
\text{allocation the task has received in the quantum system}
\end{array} \right\}
= S_i(t, t) - s_i(t, t)
\]

Schedule tasks so that the lag is bounded for all tasks over all time intervals.
» What is the least upper bound on lag?
Proportional Share Resource Allocation

Timing analysis

Is a task guaranteed to complete before its deadline?
» How late can a task be?

Theorem: Let $c$ be the size of the current request of task $T$. Task $T$’s lag is bounded by

$$-q < lag_T(t) < q$$

Rate-Based Resource Allocation

Summary

- There’s life beyond rate monotonic scheduling
- Rate-based resource allocation simplifies systems wherein
  » Work is generated at non-periodic but structured rates
  » Tasks may “misbehave”
- Liu and Layland extensions
  » Rate models demonstrate a fundamental distinction between static priority and deadline scheduling methods
- Fluid flow models
  » Real-time ± quantum
  » No fundamental distinction between real-time and non-real-time tasks
  » Provide strict isolation between tasks