Rate-Based Execution Models For Real-Time Multimedia Computing

On the Duality of Proportional Share and Liu & Layland Style Resource Allocation

Kevin Jeffay
Department of Computer Science
University of North Carolina at Chapel Hill
jeffay@cs.unc.edu
September 26, 1997

http://www.cs.unc.edu/~jeffay/courses/pisa

---

Proportional Share Resource Allocation

Outline

- Fluid-flow resource allocation models
  » Packet scheduling in a network

- Proportional share resource allocation models
  » CPU scheduling in an operating system

- On the duality of proportional share and traditional real-time resource allocation models
  » How to make a provably real-time general purpose operating system
On Proportional Share Allocation vs. Traditional Real-Time Scheduling

- **Proportional share allocation**
  - Uniform rate of execution
  - “Firm” real-time response
  - Provides fault containment in the time domain
  - Easy to implement in an operating system

- **Traditional real-time scheduling**
  - Hard-real-time response
  - Isolation from non-real-time processes

The Essence of Real-Time Resource Allocation

- Real-time processes are allocated a *fraction* of the CPU’s capacity (an *absolute share*)
  - Canonical real-time, periodic process model: \( f_i = \frac{c_i}{p_i} \)
    - \( c_i \) is the execution time of process \( i \)
    - \( p_i \) is the period of process \( i \)

- Process \( i \) executes \( c_i \) time units every \( p_i \) time units
Integrating Proportional Share &
Traditional Real-Time Resource Allocation

- Weights and shares are *duals*

\[ f_i = \frac{w_i}{\sum_j w_j} \]

» Fixing the weight \( w \) results in proportional share allocation
» Fixing the share \( f \) results in real-time execution

Therefore, characterize each process by a pair \((w, f)\), where

» \( w \) — weight, the cost the process is willing to pay for execution
» \( f \) — fraction (share) of the CPU the process should receive

Exploiting the Weight/Share Duality

**Predictability v. Cost**

- Interpret \( w \) as the rate at which a process is charged for service
  » A process with a fixed weight is charged \( w \times \Delta t \) to use the resource over a time interval of length \( \Delta t \)

- Under proportional share resource allocation, *cost* is fixed and *execution rate* is variable
  » a process knows how much it is charged over any future time interval, but doesn’t know how much service time it will receive

- Under traditional real-time allocation, *execution rate* is fixed and *cost* is variable
  » a process knows how much service time it will receive over any future time interval, but doesn’t know how much it will be charged
Exploiting the Weight/Share Duality
Trading off cost for quality-of-service

◆ Fix a weight, receive a variable share
  » Process 1’s:
    service time = 3/4
    cost = 20

◆ Fix a share, pay a variable cost
  » Process 1’s:
    service time = 3/4
    cost = 24

Exploiting the Weight/Share Duality
Trading off cost for response time

Process 1: Service time = 1
Cost = 20

Process 1: Service time = 1
Cost = 15
Exploiting the Weight/Share Duality

Scheduling class hierarchy

\[
\frac{W_{PS}}{W_{RT}} = \frac{F_{PS}}{F_{RT}}
\]

\[
(W, 1) \\
(\sum W_{PS}, \sum f_{PS}) \\
(w_1, f_1) \quad \ldots \quad (w_n, f_n)
\]

Proportional share class

\[
(\sum W_{RT}, \sum f_{RT}) \\
(w_{n+1}, f_{n+1}) \quad \ldots \quad (w_{n+m}, f_{n+m})
\]

Real-time class

Scheduling Hierarchy Example

Admitting a new proportional share process

Adding a PS process halves the execution rate of other PS processes

» doubles the cost of real-time processes
Scheduling Hierarchy Example

Admitting a new real-time process

Adding a real-time process has the same effect

- PS processes halve their share
- Real-time processes pay twice as much

Exploiting the Scheduling Hierarchy

Layering a real-time scheduler on top of a PS scheduler

- Proportional share scheduling can provide a virtual CPU abstraction to other operating systems

- Example: Execute a real-time operating system as a process within a general purpose, proportional share system
Exploiting the Scheduling Hierarchy
Layering a real-time scheduler on top of a PS scheduler

- Feasability test for a set of periodic tasks scheduled with an earliest deadline first scheduler on top of a proportional share scheduler:

Experimental Evaluation
EEVDF Implementation in FreeBSD

- **Platform**
  - PC compatible, 75 Mhz Pentium processor, 16 MB RAM

- **Implementation**
  - Replaced FreeBSD CPU scheduler
  - Time quantum = 10 ms

- **Experiments**
  - Non-real-time tasks making uniform progress
  - Speeding up and slowing down task progress by manipulating weights
  - Real-time execution (of non-real-time programs!)
Proportional Share Scheduling Example
Uniform allocation to non-real-time processes

Proportional Share Scheduling Example
Dynamic share redistribution
Proportional Share Scheduling Example
Mixed workload: 1 real-time & 2 ps processes

\[ \sum w_i = 6 \]
\[ \sum w_i = 13.5 \]

Number of Dhrystone Iterations (x 1,000)

Elapsed Time (secs)

Process 2’s share remains fixed throughout at 1/3

MPEG Player Example
2 Proportional share players, 1 real-time player

Player 1
\[ w = 3 \]

Player 2
\[ f = 1/2 \]

Player 3
\[ w = 2 \]
Summary & Conclusions

Proportional share v. traditional real-time scheduling

- Weights and shares are duals

- There exists a simple framework to integrate proportional share and real-time resource allocation
  - Subsumes traditional priority and real-time scheduling

- By using EEVDF, we’ve implemented a CPU scheduler that provides support for
  - real-time
  - interactive, &
  - batch applications

Rate-Based Execution Models For Real-Time Multimedia Computing

Summary

- Multimedia services are greatly enhanced by the existence of real-time communication and computation support

- Traditional approaches to real-time OS support are too hard to apply and don’t fit requirements well

- We’re experimenting with new programming models and new implementation paradigms

- Stay tuned!
Rate-Based Execution Models For Real-Time Multimedia Computing

**Summary**

- Rate-based execution models are more robust models for real-time multimedia computing
  - Seamless integration of real-time & non-real-time requirements
  - Simple “tuning knobs”
  - Graceful degradation
  - A dual of existing periodic models

- **Easy to implement**
  
  (In the case of proportional share allocation)