

CS4220 Embedded Systems CS6235 Real-Time Systems

7A: Feedback Scheduling

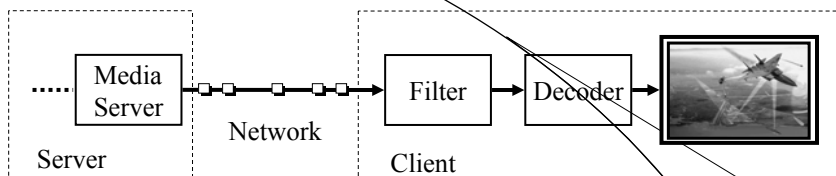
Instructor: Calton Pu

calton.pu@cc

TA: Younggyun Koh (young@cc)

1

Motivation



- Real-rate applications
 - ◆ Real-world performance demands
- Automated rate matching
 - ◆ fine-grain adjustment of allocation
 - ◆ dynamic response to variable rates
 - ◆ low programming complexity

2

Priority Schedulers

- Give CPU to highest priority task
- Scheduling policy
 - ◆ Assigns priorities to tasks
- Scheduling mechanism
 - ◆ Sort runnable task queue according to priorities
 - ◆ Run the task at the head of the queue
- Example of problems
 - ◆ Priority inversion

3

Proportional Scheduler

- Give tasks a proportional share of CPU
- Scheduling policy
 - ◆ Assign proportions to tasks
- Scheduling mechanism
 - ◆ Round-robin queue
 - ◆ Time slice allocated according to the proportion
- Example of problems
 - ◆ Need to cycle through queue fast

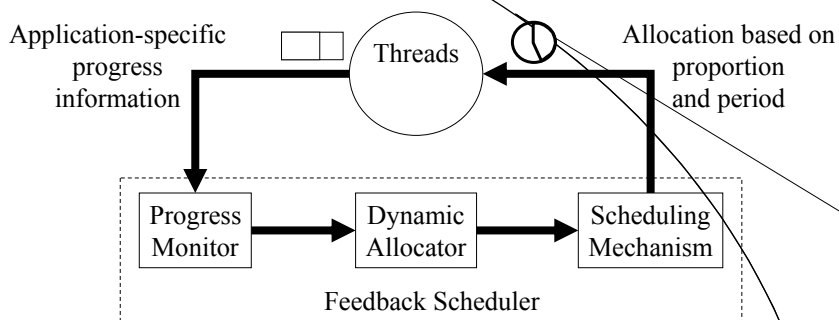
4

Current Schedulers

- Priority-based schedulers
 - ◆ all-or-nothing or equal share: coarse grained
 - unless aided by the application programmer: high complexity
- Proportional-share schedulers
 - ◆ require program to specify resource needs: high complexity
 - ◆ hard to know for variable rate apps: lack dynamic response

5

Feedback Approach



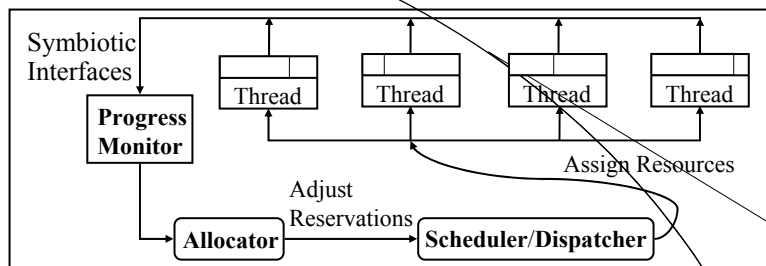
6

Simple Idea

- Real-rate task model
 - ◆ Known CPU requirements
 - ◆ Known deadline
- Dynamic scheduler
 - ◆ Monitors task progress towards deadline
 - ◆ Insufficient CPU: increase allocation
 - ◆ Do it fast enough before miss

7

Our Approach



- Application progress monitor
- Resource reservation scheduler
- Dynamic allocator:
 - ◆ Feedback based allocator samples progress
 - ◆ Calculates resource needs, assigns allocation

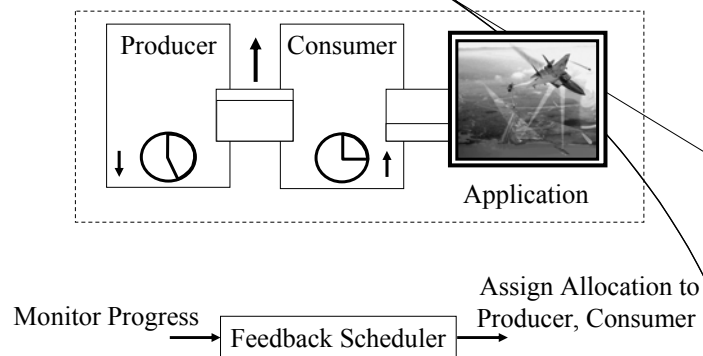
8

Progress Estimation

- Automated monitoring feasible
 - ◆ Symbiotic interfaces between system and application
- Concrete examples
 - ◆ Server: consumer of a bounded buffer
 - ◆ Data rate of shared memory queues, sockets, pipes, etc
 - ◆ I/O intensive: consumers of the I/O subsystem
 - ◆ Interactive: listen to tty instead of sockets

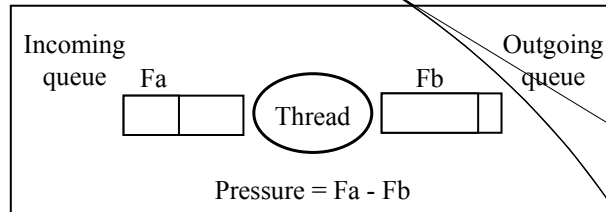
9

Queue Example



10

Progress Monitor

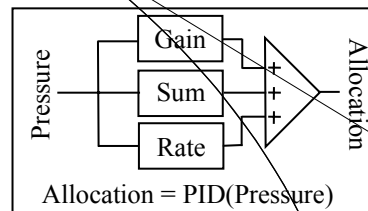


- Fill levels show data flow pressure
 - ◆ High pressure: increase CPU allocation/priority
 - ◆ Low pressure: decrease allocation/priority
 - ◆ Goal: keep data flowing smoothly

11

Dynamic Allocator

- Real-rate: pressure fed to PID controller
- Real-time: progress towards reservation fed to PID controller
- Other: constant positive pressure (bypass adaptation)



12

Overload Allocation Policy

- Real-time:
 - ◆ renegotiate reservations, admission control
- Others:
 - ◆ weighted fair sharing, proportional squishing
 - ◆ Transient load: jobs regain allocation
 - ◆ Long-term load: quality exceptions to applications
 - Applications can shed load
 - Increase job importance

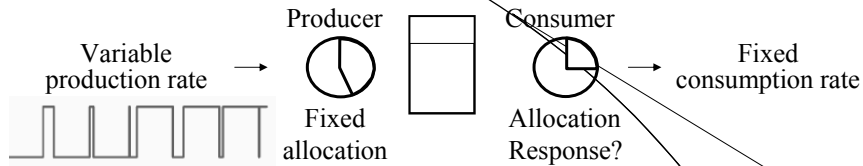
13

Scheduling Mechanism

- Reservation scheduler
 - ◆ based on proportion and period
 - ◆ uses rate-monotonic scheduling
 - ◆ allocations enforced during process dispatch
 - ◆ allows fine granularity allocation adjustment
 - ◆ low overhead for changing reservations
 - ◆ respects reservations for applications that specify them

14

Allocator's Response

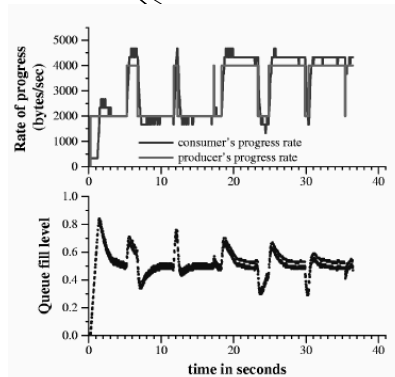


- Response to varying progress
- Vary the progress rate of the producer
 - ◆ fixed allocation, variable production rate
- Measure the progress rate of the consumer
 - ◆ variable allocation, fixed consumption rate
- Progress rate = allocation X (production/consumption rate)

15

Results on Idle System

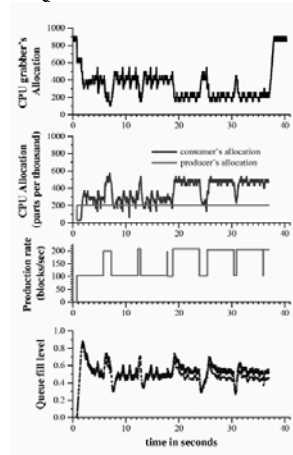
- Allocator response
 - ◆ adjusts consumer's allocation rapidly
 - ◆ matches producer's progress rate



16

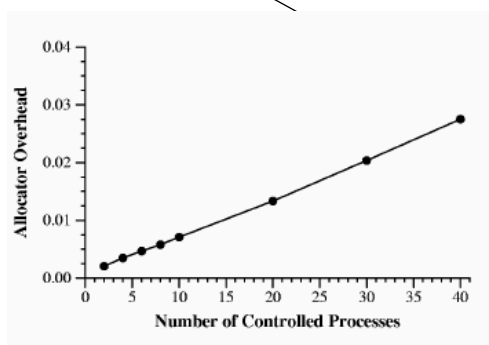
Results on Loaded System

- The “other” load does not have a progress metric
- “Other” load loses allocation to the real-rate consumer job



17

Allocator Overhead



- Linear with number of threads under our control
- Small slope of prototype - 0.06% per process

18

Benefits of Our Approach

- Finer grained allocation
- Avoids starvation and priority inversion
- Easy admission control
- Automatic estimation of reservations
- Implementation in Linux
source code available at
<http://www.cse.ogi.edu/DISC/projects/quasar/releases>

19

Related Work

- Real-time priorities
- Proportional allocation [Waldspurger]
- Reservation Schedulers [Nieh, Jones]
Previous work focuses on satisfying requirements, rather than inferring requirements accurately
- Balancing between real-time and normal jobs
- Pipeline based allocation [Jeffay]

20

Adaptive Controller

Proportion Specified	Progress Metric	Period Specified	Period Unspecified
Yes	N/A	Real-time	Aperiodic Real-time
No	Yes	Real-Rate	
	No	Miscellaneous	

- **Real-time:** controller typically doesn't touch specification
- **Aperiodic Real-time:** controller assigns default period
- **Real-Rate:** controller uses progress metric to estimate pressure
- **Miscellaneous:** heuristic, constant pressure

21

Conclusion

- **Unified scheduling mechanism**
 - ◆ real-rate, traditional and real-time jobs
 - ◆ expose application-level progress
- **Move scheduling to a higher level of abstraction**
 - ◆ think progress rather than allocation
- **Dual strategy**
 - ◆ proportional share scheduler: gear box
 - ◆ dynamic allocator: automatic transmission

22