Idletime Scheduling with Preemption Intervals

Lars Eggert (NEC) and Joe Touch (USC/ISI)

20th ACM Symposium on Operating Systems Principles, Brighton, United Kingdom

October 26, 2005

Motivation

- use available bandwidth "in the background"
- best effort vs. "least effort"
- IP differentiated services extensions OK
- but what if the bottleneck is in the host?
- diffserv forwarding at routers is ineffective if there are no queues
- need diffserv-like mechanism in the OS

Challenges

- system bottleneck resource depends on current workload & changes dynamically
- scheduler of current bottleneck resource dominates overall system behavior
- don’t want to change all schedulers & queues
    - some may be in hardware
- don’t want to modify apps for FG or BG

Idletime Service

- system-wide, least-priority service class
  - default service class = FG
    (inverse of traditional QoS)
- in some sense, generalization of POSIX “idprio” CPU scheduling
- ideal: utilize all available capacity of all resources for BG work – with zero impact on FG work
- talk focuses on temporally shared resources, paper discusses ideas for spatially shared (storage)

Applications & Benefits

- prefetching/caching = reduce access costs
  - disk: block replication, arm movement
  - network: IKE, DNS, PMTU, “prefetch means”
  - currently: conservative limits to avoid overload
- system optimization & maintenance
  - fsck, defrag, virus scan, update, etc.
- process/data migration systems
  - Condor, Sprite, x@home, Mether, etc.
  - coarse-grained, single-resource, remote benefit

Goals

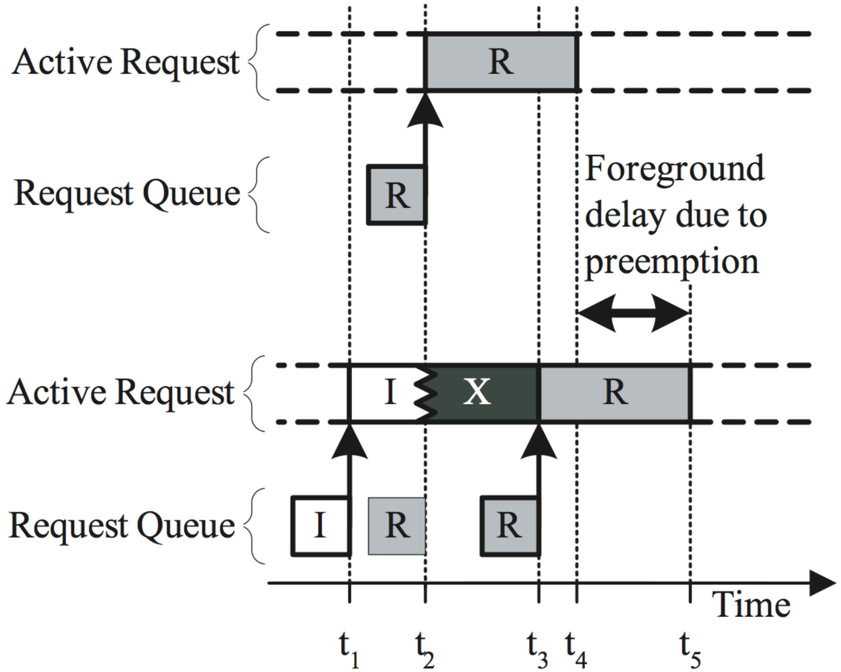
- ideal: utilize all available capacity of all resources for BG work with zero impact on FG
- difficult: high preemption costs without hardware support, always some preemption costs
- primary goal: minimize FG impact
  - or people won't use it
- secondary goal: reasonable BG throughput

Idletime Principles

1. isolation
   - BG side effects interfere with FG
   - can affect FG correctness & performance
2. prioritization
   - never serve BG if FG queued
3. preemptability
   - preempt/abort BG when FG arrives
   - preemption cost – main factor delaying FG

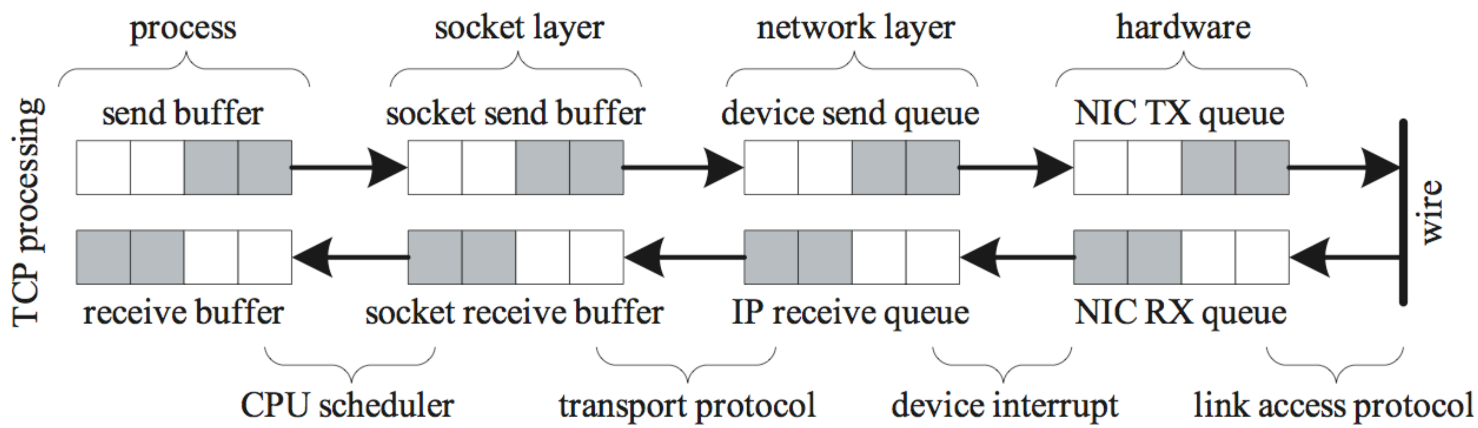
Preemption Cost

- BG → FG switch: <u>delay</u>
  main cause of FG performance reduction
- limit preemption cost = limit FG impact



Work Conservation

- never remain idle with work queued (and never destroy completed work)
- challenge: OS + hardware = queue hierarchy
- hierarchy level implies priority
- causes lower-level BG to delay higher-level FG

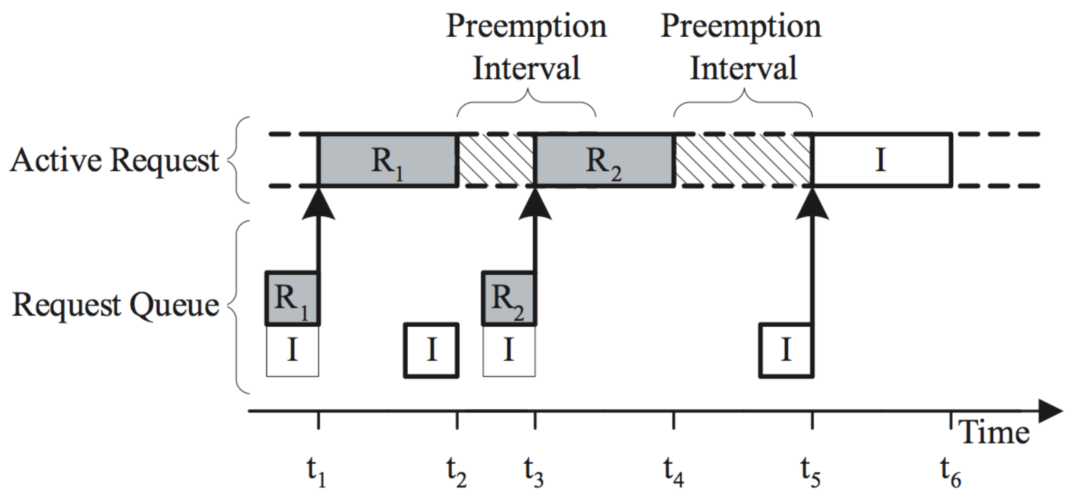


Idea

- work conservation for BG <u>creates</u> idletime worst-case
  - preemption before each FG request
  - up to 50% impact when active BG must run to completion
- <u>idea: relax work conservation for BG only</u>
  - limit preemptions = limit FG impact

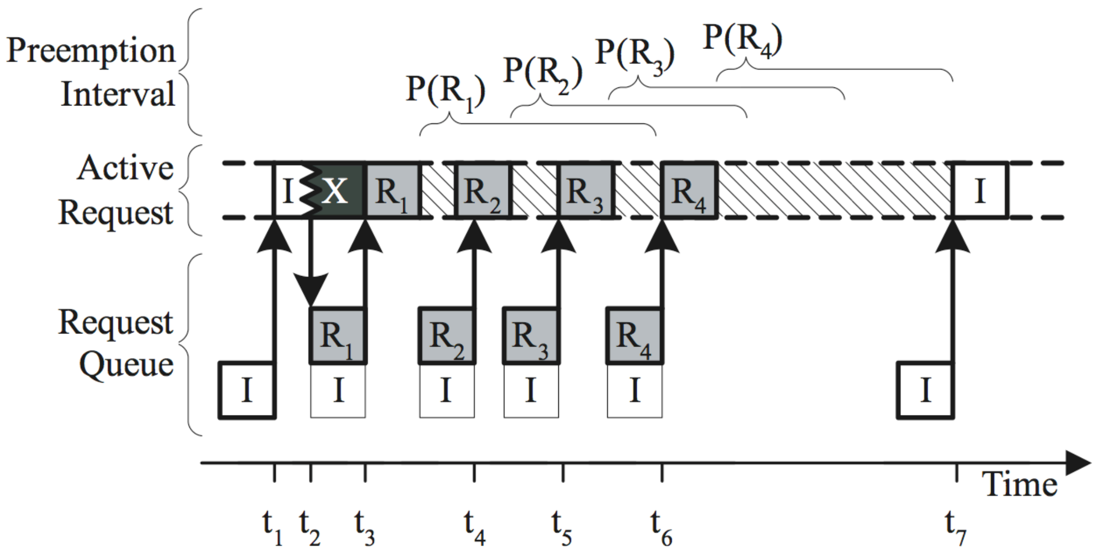
Preemption Interval

- preemption interval = period of relaxed BG work conservation
  - new FG → start immediately
  - BG → if in PI, delay until PI ends
- enter PI before each FG → BG switch



Behavior

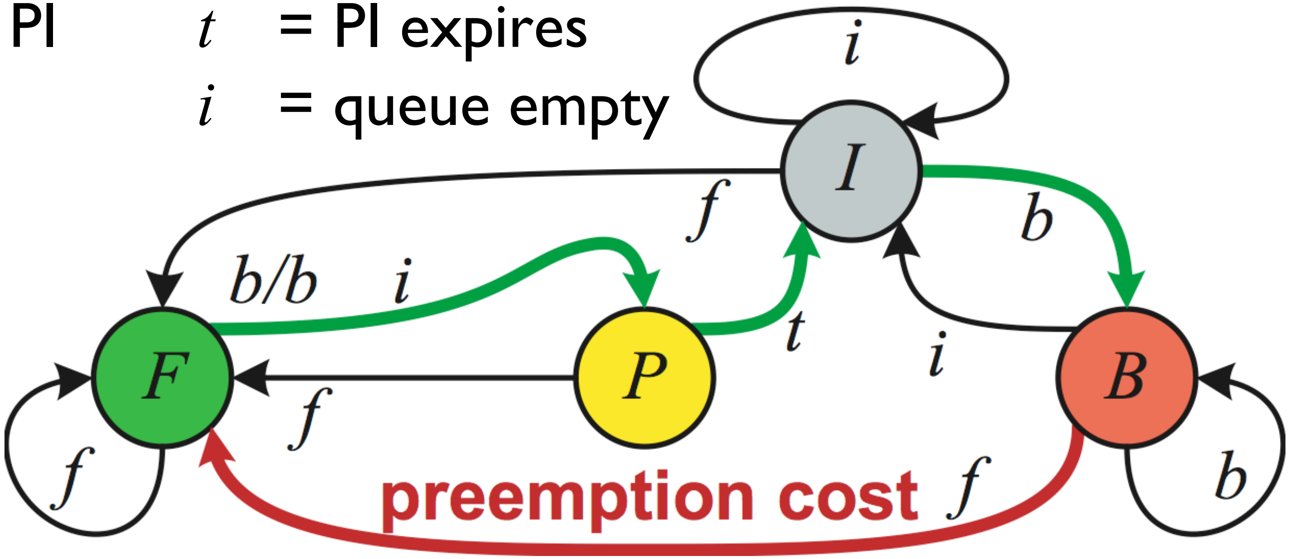
- PI creates bursts of FG requests
- max. 1 preemption/burst
- limits FG impact



Idletime Scheduler

- priority queue + PI scheduling

<u>states</u>	<u>events</u>
F = FG active	f = FG in queue
B = BG active	b = BG in queue
P = idle in PI	t = PI expires
I = idle	i = queue empty



Consequences

- idletime scheduling suspends BG work at higher levels
    - can’t interfere with FG at lower levels
- can be implemented as localized modifications – extend traditional OS
- how long to suspend BG work conservation for?

PI Length

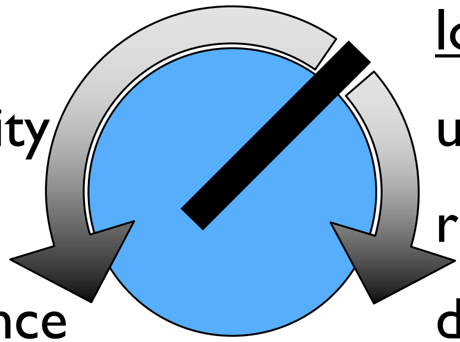
- parameter: controls scheduler
  - FG impact ~ BG performance
  - effective PI length?

<u>shorter PI</u>

utilize more idle capacity

higher FG impact

increase BG performance



<u>longer PI</u>

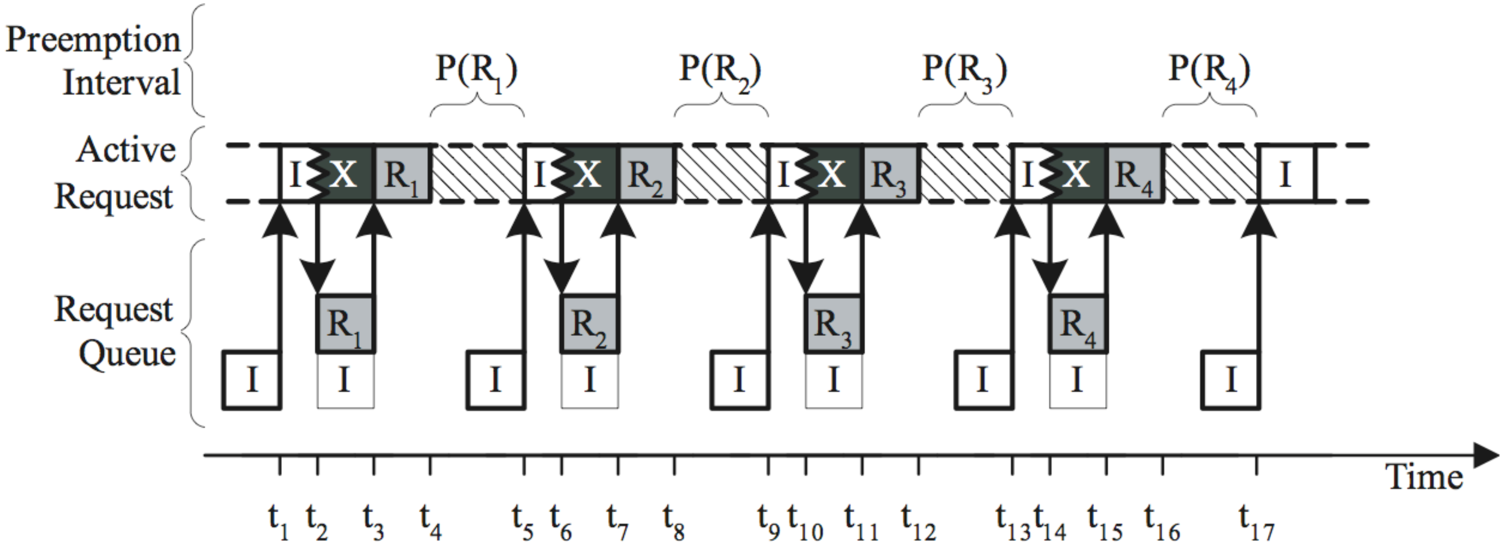
utilize less idle capacity

reduce FG impact

decrease BG performance

Short PIs

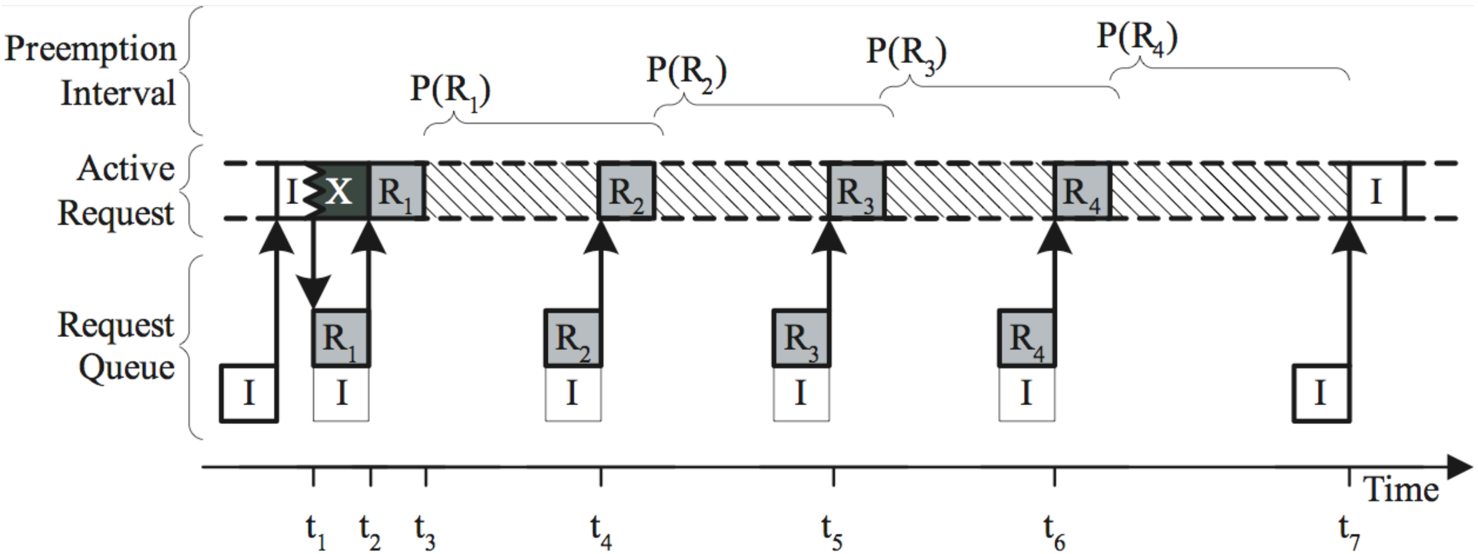
- too short = ineffective
- mechanism degenerates into priority queue
- no cost limit = no FG impact reduction



Lars Eggert <eggert@netlab.nec.de>

Long PIs

- too long = waste idle capacity
- poor BG throughput
- limited usefulness



Effective PI Lengths

- factors: resource, workload, user policy
- lower bound: create FG burst length > 1
  - otherwise: no cost amortization
- upper bound: FG inter-arrival gap
  - otherwise: BG starvation

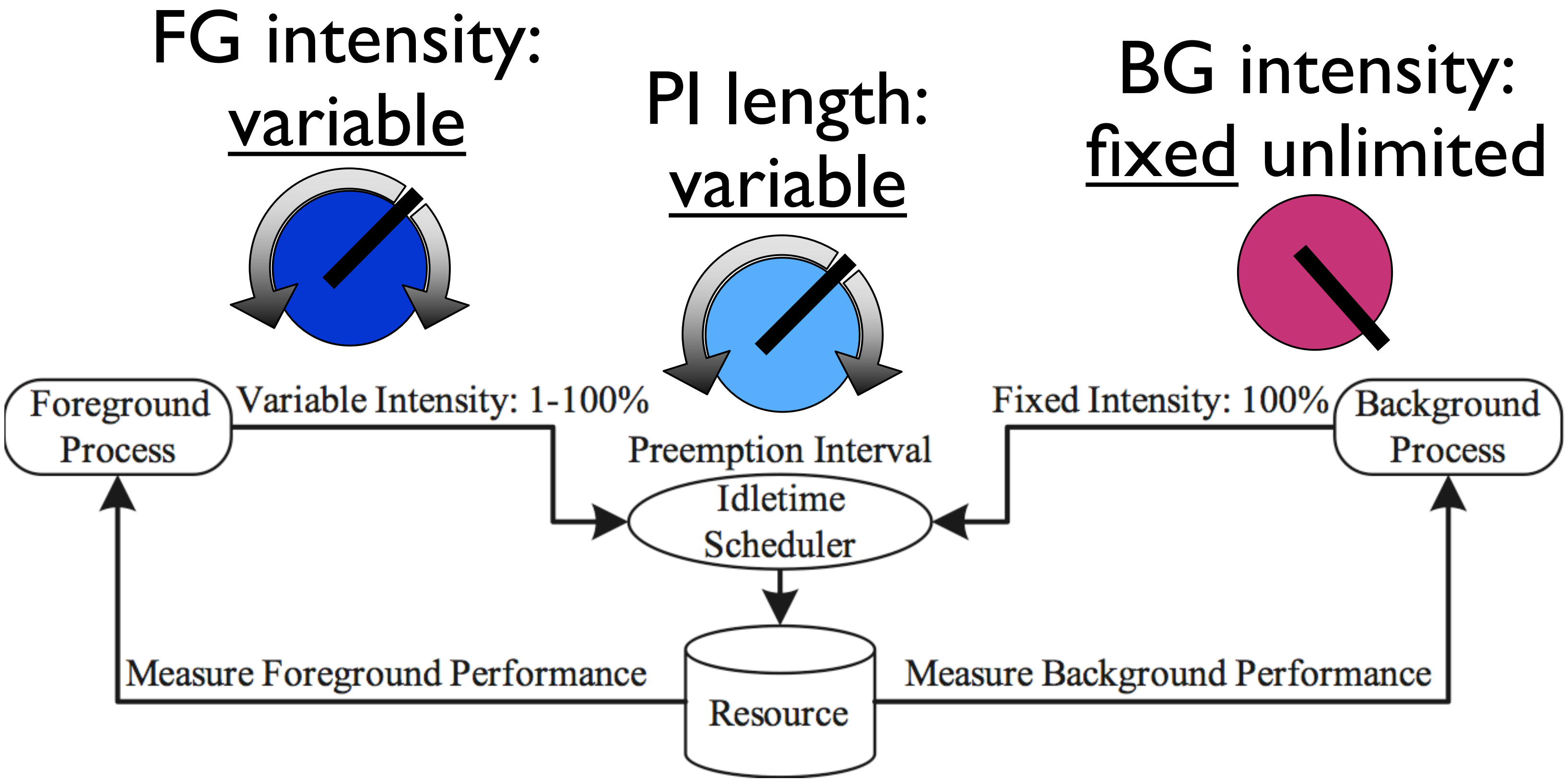
Future Extensions

- automatic PI length adaptation
  - preemptions before FG → lengthen PI
  - FG without preemption → shorten PI
  - TCP-like AIMD scheme: preemption ~ loss
- tolerate limited FG impact
  - skip PI after FG burst = increase BG throughput

Implementation

- localized modifications to FreeBSD 4.7
  - disk: replace disksort()
  - network: new ALTQ discipline, tag packets
- begin PI with FG request, not at end = simplify code
  - expectation: PI < service time ineffective
  - PI expires while FG active, system degenerates into priority queue

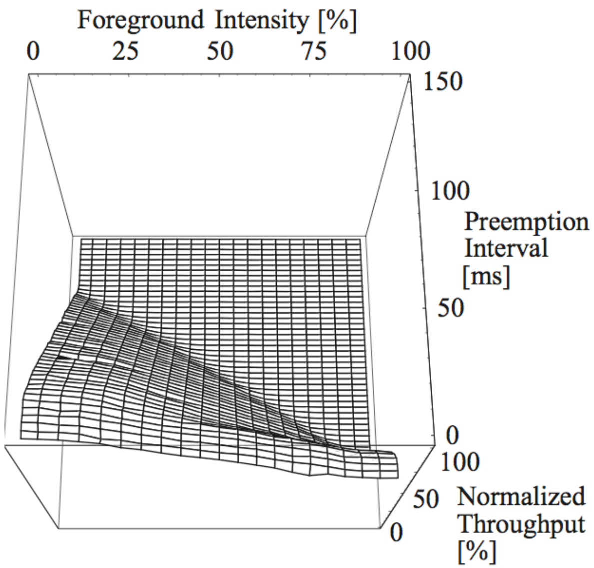
Experimental Setup

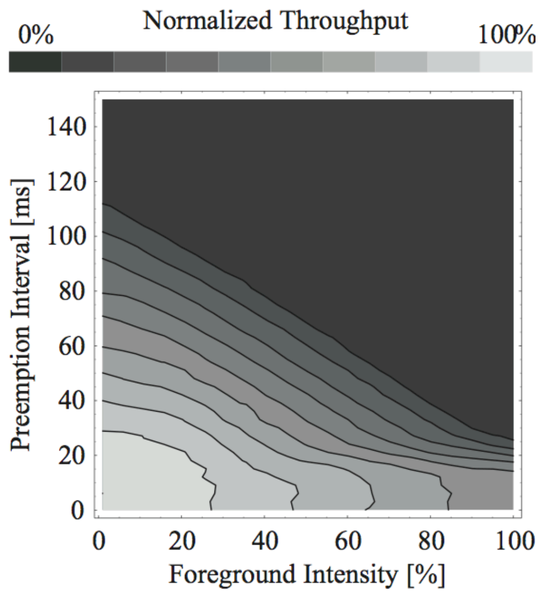


(intensity = % cycles used to generate load)

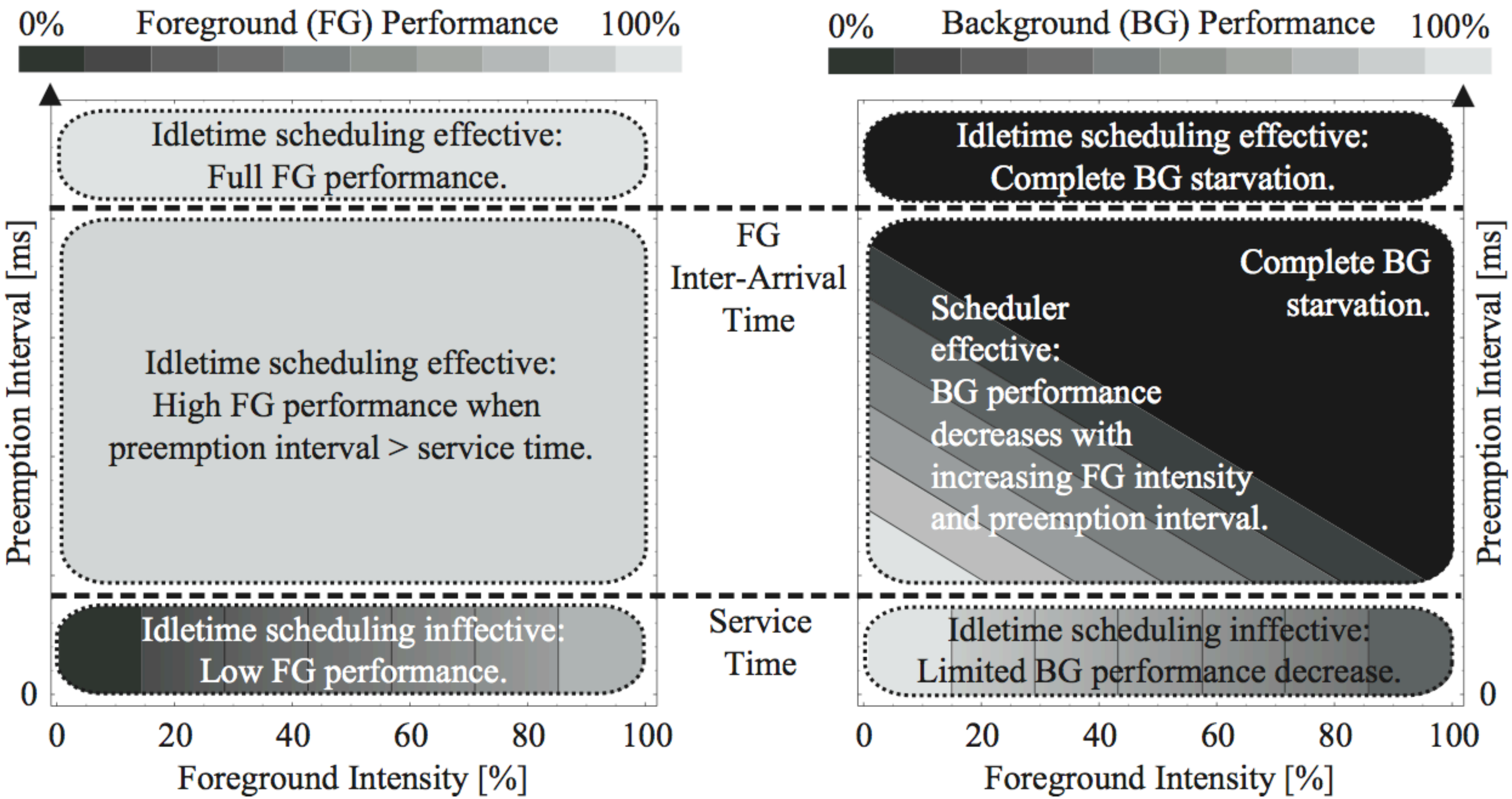
Metrics

- FG/BG throughput
- normalize against baseline (= no BG)
- contour plot: lighter shades = better





Expectations

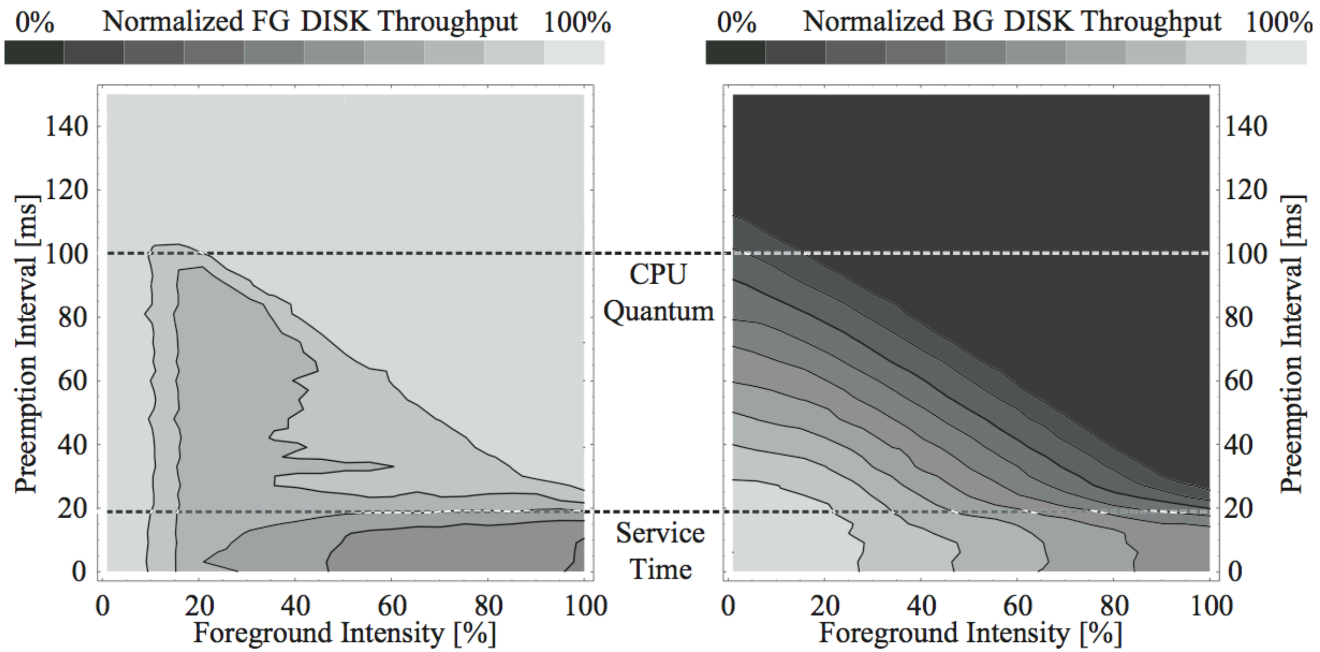


Disk Setup

- UFS file system, random data
- single disk, isolated ATA channel
    - 8.2GB Western Digital Caviar AC28200
    - 15ms maximum seek + 5ms latency (mean)
- FG + BG randomly read 512-byte blocks
- Pentium III SMP, 733Mhz, 512MB RAM

Disk: Random Access

- FG > 80%, BG ≤ 90% of baseline throughput
- 20% impact ~ 1 BG request (20ms seek)

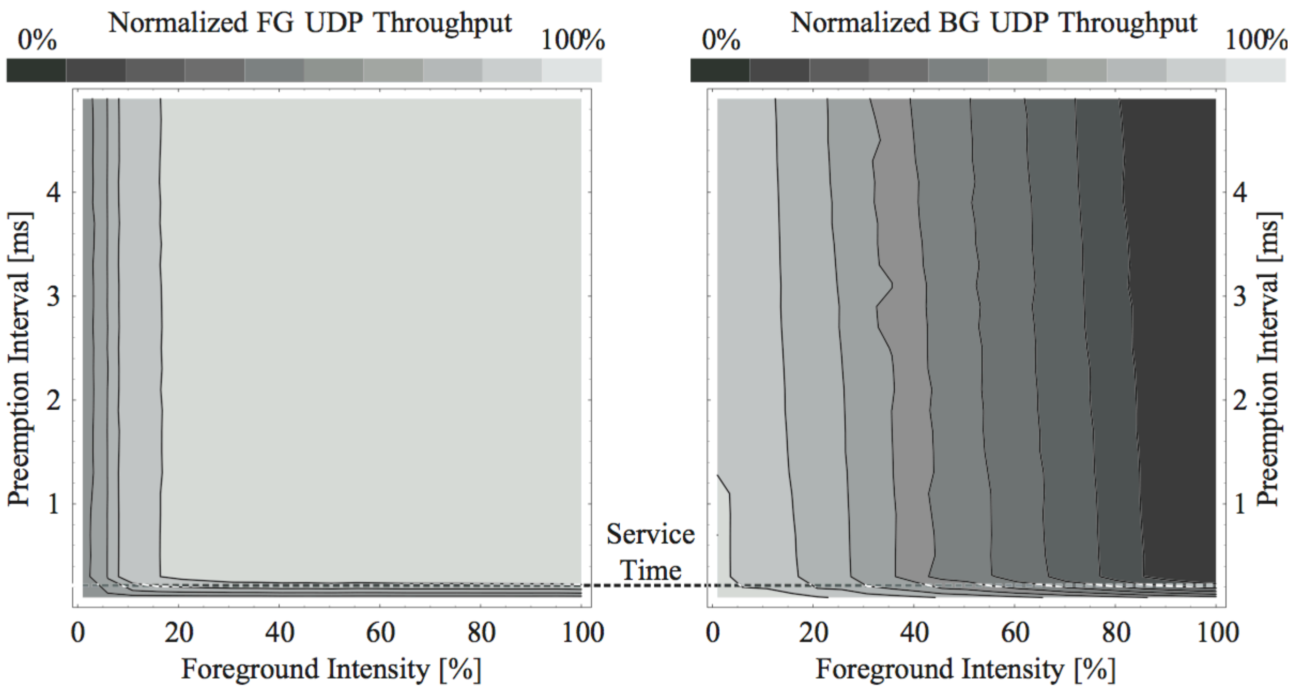


Network Setup

- direct, isolated LAN link (cross-over cord)
  - Intel PRO/1000F 1Gb/s Ethernet fiber
- source + sink hosts
  - Pentium III SMP, 733Mhz, 512MB RAM
- combinations of UDP + TCP

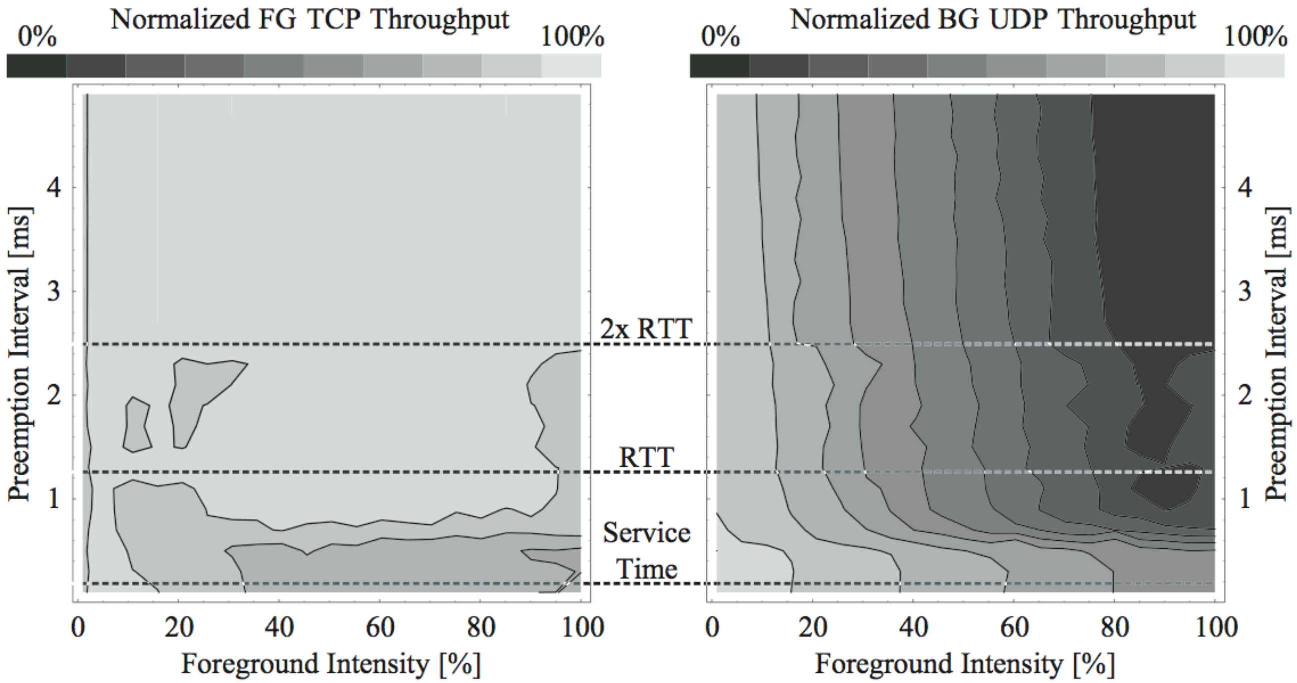
Network: UDP/UDP

- FG > 90% + BG ≤ 90% of baseline
- except: low intensity = short FG burst



Network TCP/UDP

- worst case: FG TCP vs. greedy BG UDP
- PI length ~ RTT? (1.25ms minimum RTT)



Lars Eggert <eggert@netlab.nec.de>

Related Work

- realtime systems – resource reservations
  - computation deadlines, predictability
- network diffserv/QoS – many mechanisms
  - L2: drop priority flag (ATM, FR)
  - L3: IP TOS, diffserv, intserv, prop-share
  - L4: TCP-LP, TCP Nice, MulTCP
  - L7: Mozilla, BITS, LSAM, push-polite

Related Work (2)

- idle capacity consumers
  - process & data migration
  - prefetching & caching
- anticipatory scheduling
  - disk performance through locality
- MS Manners
  - reactive monitoring, app cooperation
- other OS can be configured/extended (Scout, etc.)

Additional Work

- paper
  - ideas for extending this to storage resources
  - idletime networking improvements (inbound processing)
- elsewhere
  - analytical model of idletime scheduling predicts behavior with >85% accuracy
  - experimental analysis of FG/BG latency

Conclusion

- generic idletime scheduler based on relaxing work conservation for BG work during preemption intervals
- resource + workload independent
- disk + network implementation
  - FG > 80%
  - BG < 90%

baseline throughput