Virtual-CPU Scheduling in the Quest Operating System

Matt Danish, Ye Li and Richard West

Contact: richwest@cs.bu.edu

Computer Science

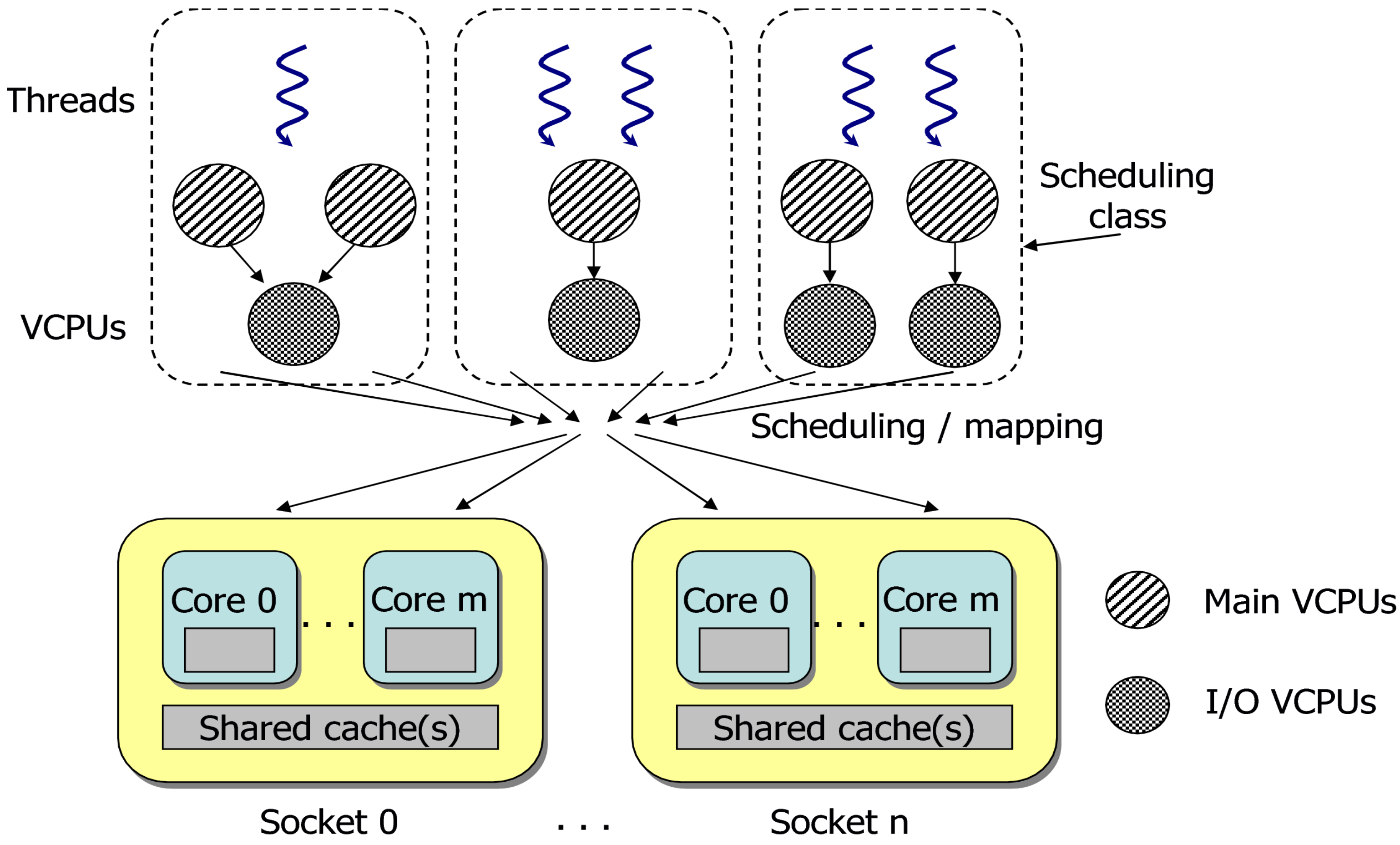
Goals

- Develop system with improved predictability
- Integrated management of tasks & I/O events
- Enforce *temporal isolation* between threads

Approach

- Introduce "virtual CPUs" for scheduling
  - Resource containers for CPU usage
  - Have *budgets* (reservations) and replenishment *periods*
  -
- Scheduling hierarchy
  - Threads mapped to VCPUs
  - VCPUs mapped to PCPUs

Big Picture


Threads
VCPUs
Scheduling
class
Scheduling / mapping
Core 0
. . .
Core m
Shared cache(s)
Core 0
. . .
Core m
Shared cache(s)
Socket 0
. . .
Socket n
Main VCPUs
I/O VCPUs

VCPUs in Quest

- Two classes
    - **Main** $\rightarrow$ for conventional tasks
    - **IO** $\rightarrow$ for IO event threads (e.g., ISRs)

- Scheduling policies
  - **Main** $\rightarrow$ sporadic server (SS)
  - **IO** $\rightarrow$ priority inheritance bandwidth-preserving server (PIBS)

SS Scheduling

- Model periodic tasks
  - Each SS has a pair (C,T) s.t. A server is guaranteed no more than C CPU cycles every period of T cycles
    - Guarantee applied at *foreground* priority
    - Can exceed this utilization at *background* priority
  - Rate-Monotonic Scheduling theory applies

PIBS Scheduling

- IO VCPUs have utilization factor, V_U
- IO VCPUs inherit priorities of tasks (or Main VCPUs) associated with IO events
  - Currently, priorities are f(T) for corresponding Main VCPU
  - IO VCPU budget is limited to:
    - $V_{T,main} * V_U$ for period $V_{T,main}$

PIBS Scheduling

- IO VCPUs have *eligibility* times, when they can execute
- $V_e = V_e + C_{actual} / V_U$

Quest Summary

- About 11,000 lines of kernel code
- About 175,000 lines including lwIP, drivers, regression tests
- SMP, IA32, paging, VCPU scheduling, USB, PCI, networking, etc

Experiments

- Intel Core2 Extreme QX6700 @ 2.66GHz
- 4GB RAM
- Gigabit Ethernet (Intel 8254x “e1000”)
- UHCI USB Host Controller
  - 1GB USB memory stick
- Parallel ATA CDROM in PIO mode

- Measurements over 5sec windows using bandwidth-preserving logging thread

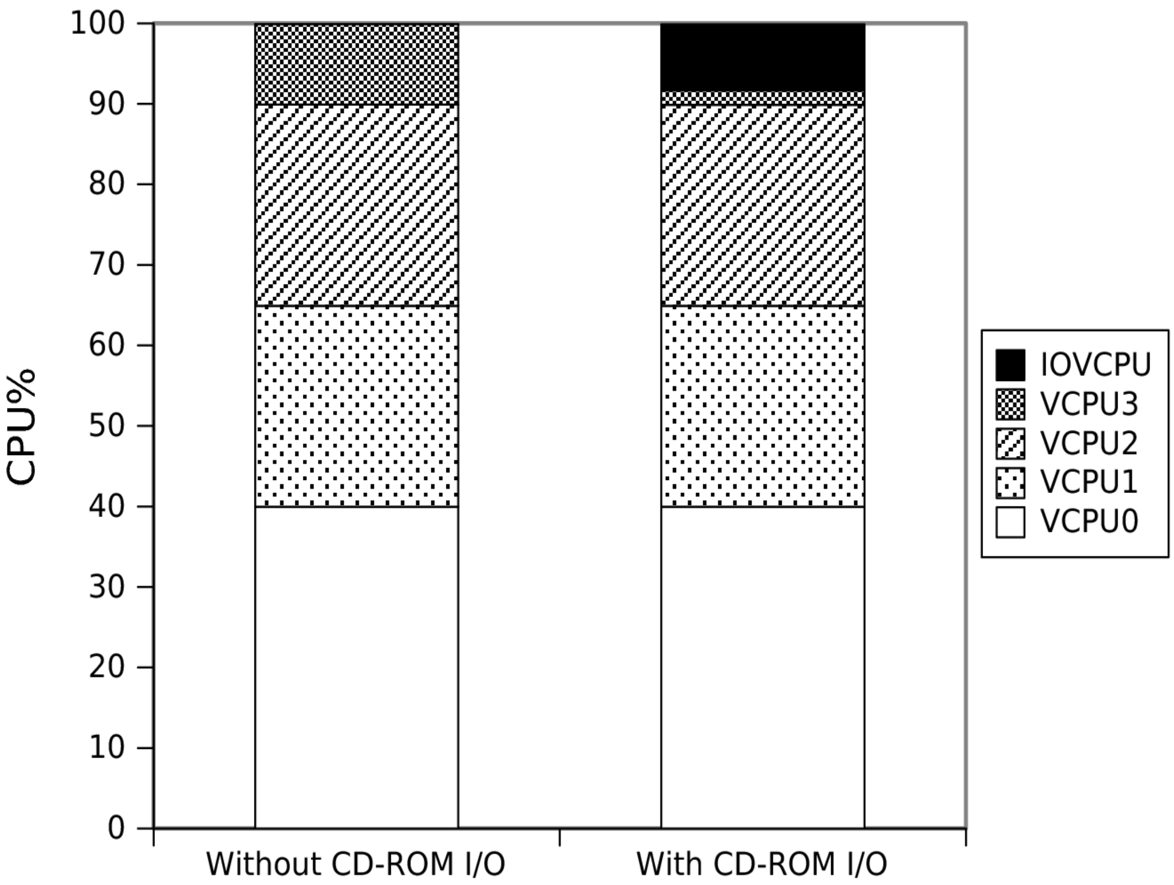
Experiments

- CPU-bound threads: increment a counter
- CD ROM/USB threads: read 64KB data from filesystem on corresponding device

I/O Effects on VCPUs

VCPU	V_C	V_T	threads
VCPU0	2	5	CPU-bound
VCPU1	2	8	Reading CD, CPU-bound
VCPU2	1	4	CPU-bound
VCPU3	1	10	Logging, CPU-bound
IOVCPU	10%	ATA	

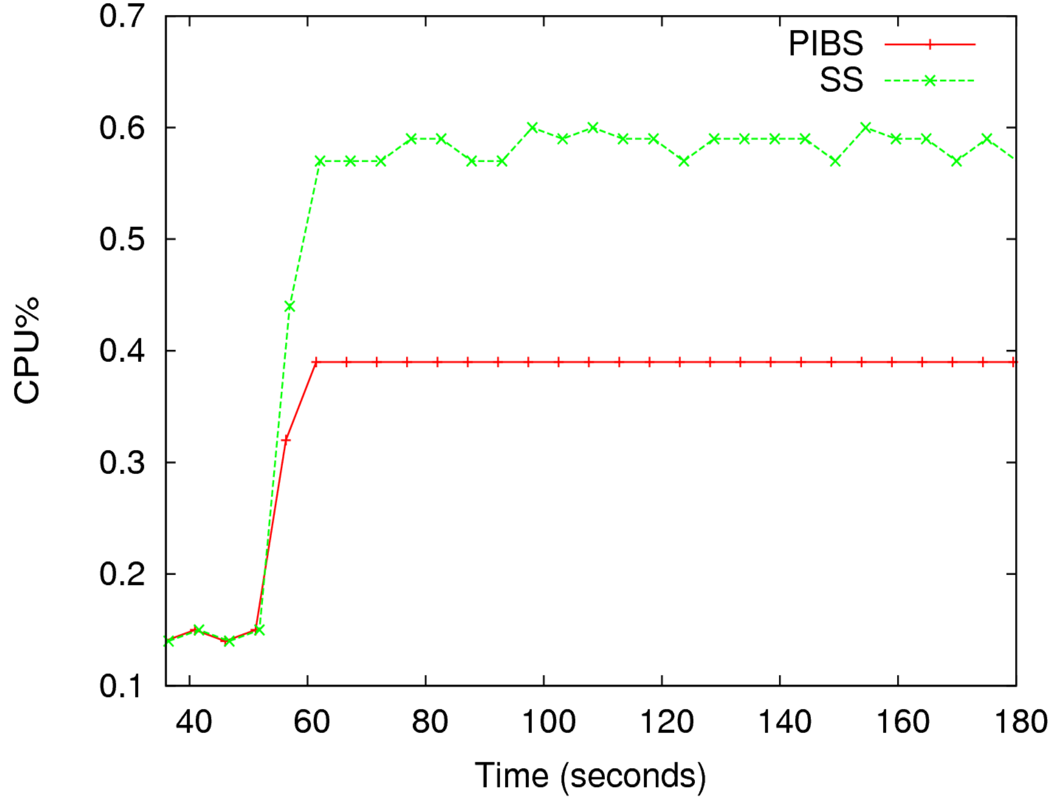
I/O Effects on VCPUs



PIBS vs SS IO VCPU Scheduling

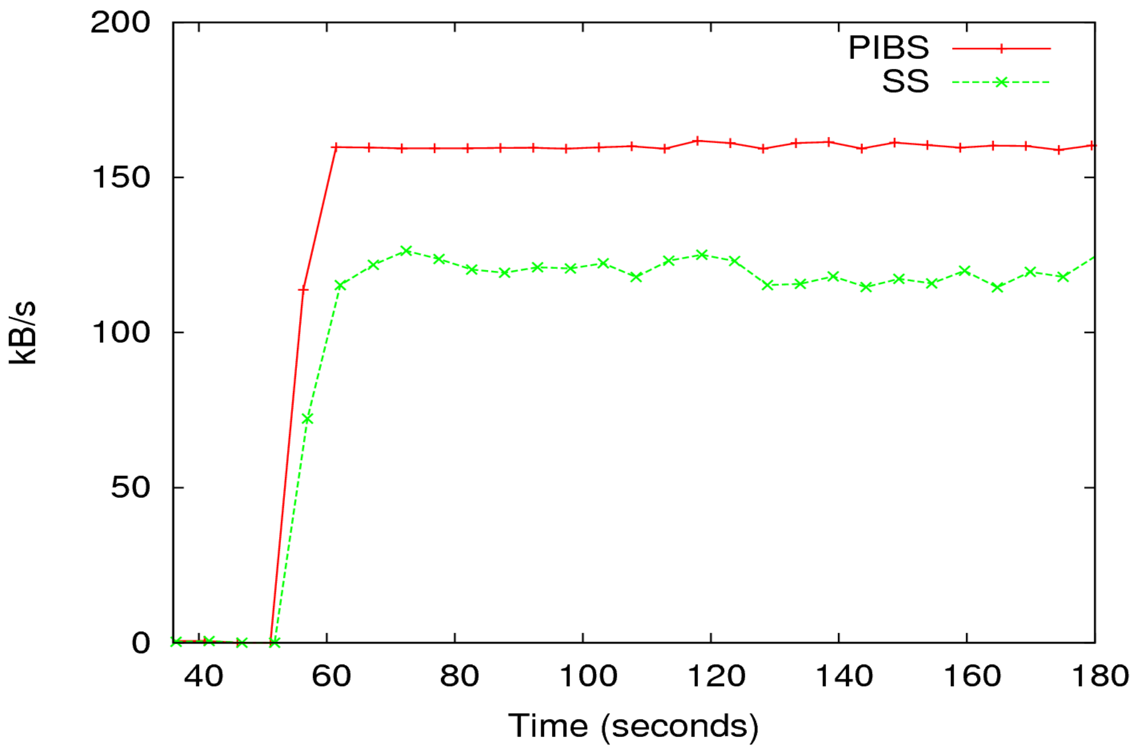
VCPU	V_C	V_T	threads
VCPU0	1	20	CPU-bound
VCPU1	1	30	CPU-bound
VCPU2	10	100	Network, CPU-bound
VCPU3	20	100	Logging, CPU-bound
IOVCPU	1%	Network	

PIBS vs SS IO VCPU Scheduling



t=50 start ICMP ping flood. Here, we see comparison overheads of two scheduling policies

PIBS vs SS IO VCPU Scheduling



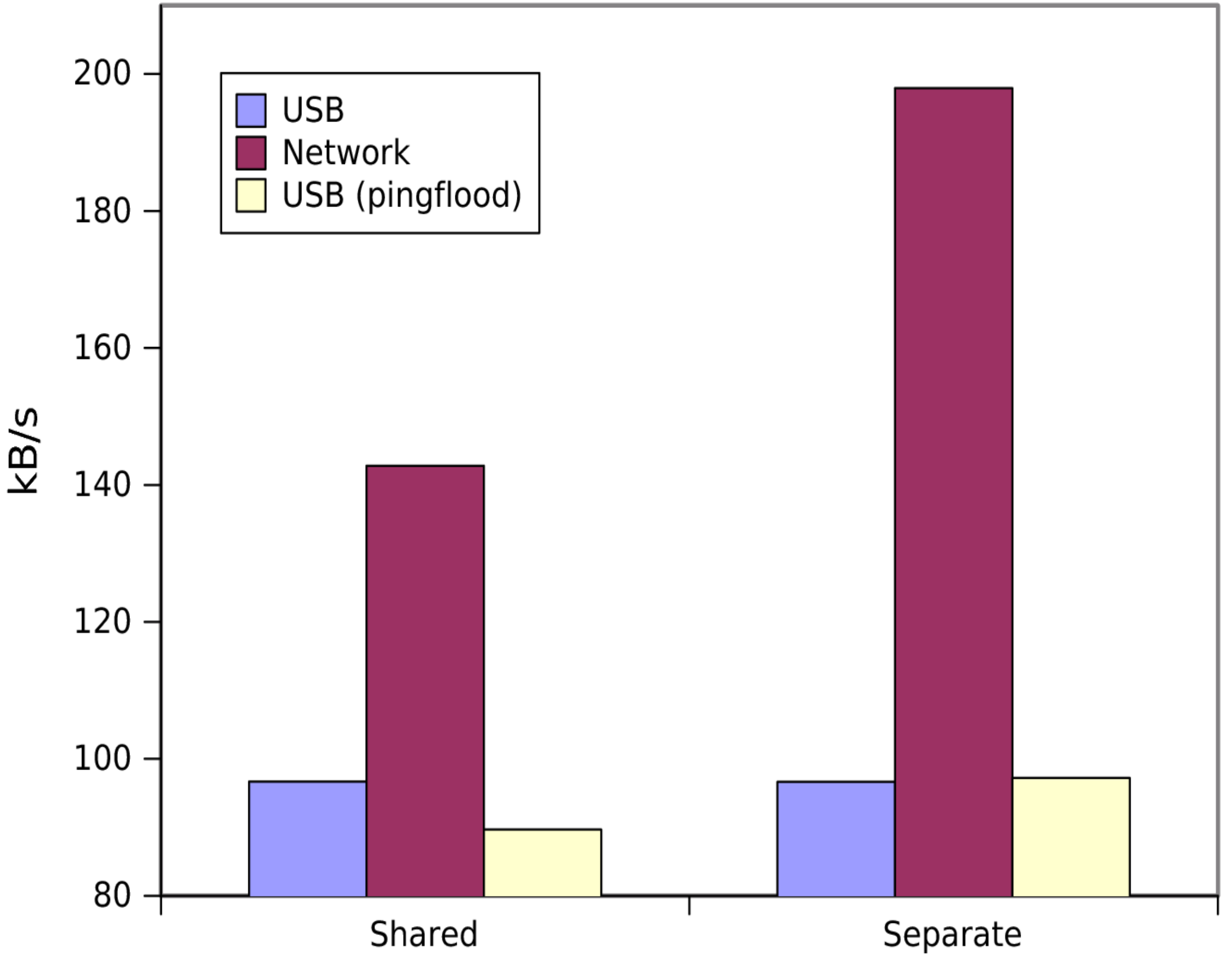
Network bandwidth of two scheduling policies

IO VCPU Sharing

VCPU	V_C	V_T	threads
VCPU0	30	100	USB, CPU-bound
VCPU1	10	110	CPU-bound
VCPU2	10	90	Network, CPU-bound
VCPU3	100	200	Logging, CPU-bound
IO VCPU	1%		USB,Network

VCPU0	30	100	USB, CPU-bound
VCPU1	10	110	CPU-bound
VCPU2	10	90	Network, CPU-bound
VCPU3	100	200	Logging, CPU-bound
IO VCPU1	1%		USB
IO VCPU2	1%		Network

IO VCPU Sharing



Conclusions

- Temporal isolation on IO events and tasks
- PIBS + SS Main & IO VCPUs can guarantee utilization bounds
- Future investigation of higher-level policies
- Future investigation of h/w performance counters for VCPU-to-PCPU scheduling