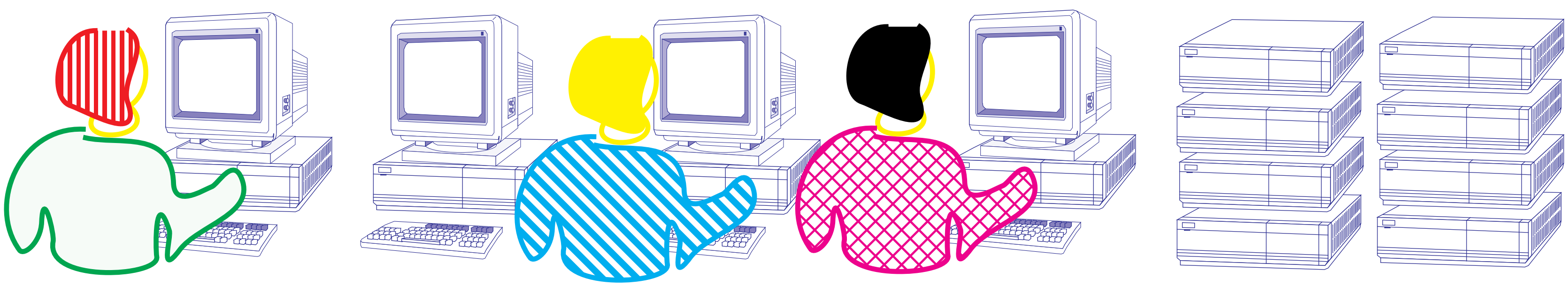
Towards Global Process Management in NOW

Remzi H. Arpaci
Andrea C. Dusseau
Amin M. Vahdat

Goal: Determine issues when scheduling sequential jobs in a pool of workstations

Motivation

- Desktop or Basement: Manage workstations globally
- Assume complete control of placement of every job
  - Jobs are never migrated
- Known: Placement depends upon state of system
  - Load and memory usage of workstation
- Show: Scheduling also depends on job characteristics
  - Some jobs run noticeably faster on some models
  - Jobs with locality should be run on same workstation
  - Multiple I/O-bound jobs should not be scheduled



Methodology

Construct 4 sets of 10 min. workloads with real applications

- Production: Independent runs from SPEC-fp
- Development: Compile-edit-debug cycle
- I/O: Jobs with I/O dependencies
- Interactive: Simulated typing and shell commands

Run jobs and measure sensitivity of workload to:

- Workstation model
- File cache locality
- Network sharing
- *(Sharing workstations)*
- *(Additional workstations)*

Future Work: Run workloads with scheduling policies

Instead: Enumerate issues learned from measurements

Heterogeneous Cluster

Scenario: Workstation cluster consists of different models

Which workstation is the fastest for each job?

	SS-10 (Model 51)	SS-10 (Model 41)	SS-20
Clock Speed	50 MHz	40 MHz	50 MHz
L1 Cache	4 KB (1-way)	16 KB (4-way)	16 KB (4-way)
L2 Cache	1 MB	1 MB	None
Main Memory Access Time	1.0 -1.4 us	1.1 - 1.8 us	0.4 - 1.0 us

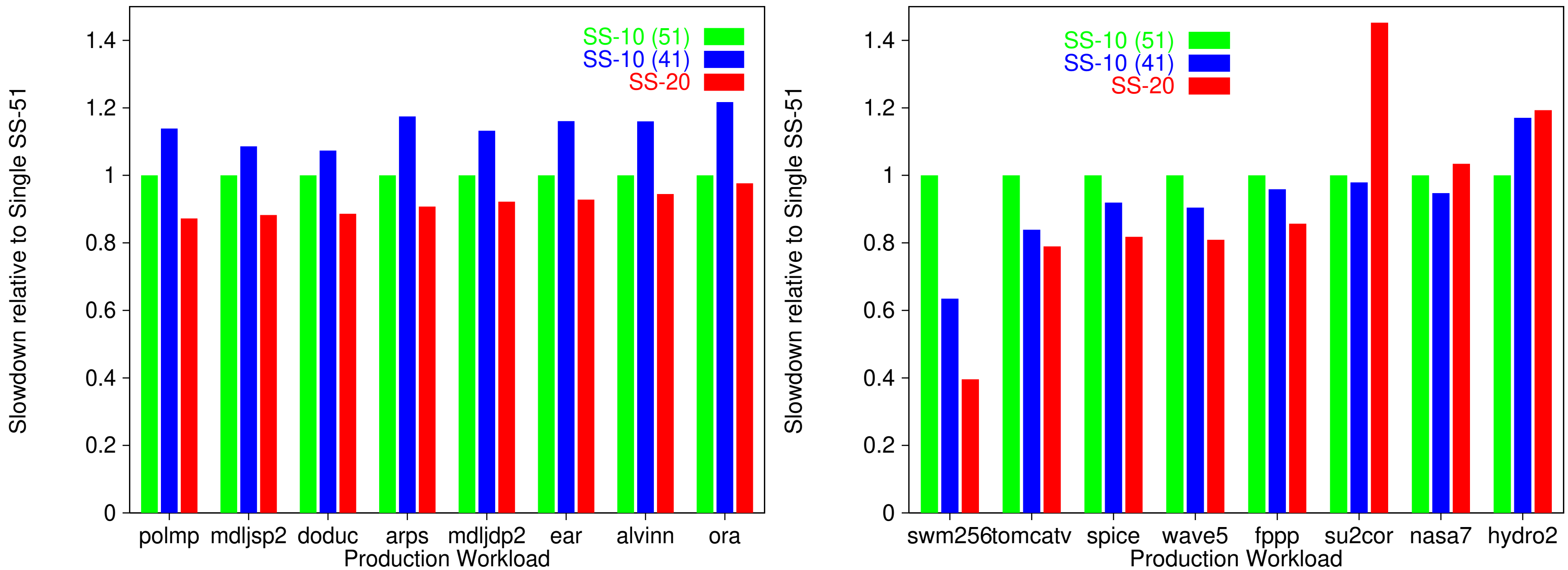
Experiment:

- Run each user workload on three SS-10 and SS-20 models

Heterogeneous Cluster

Measurement Results:

- Production Workload: No ordering of performance



- Interactive Workloads: Almost no sensitivity

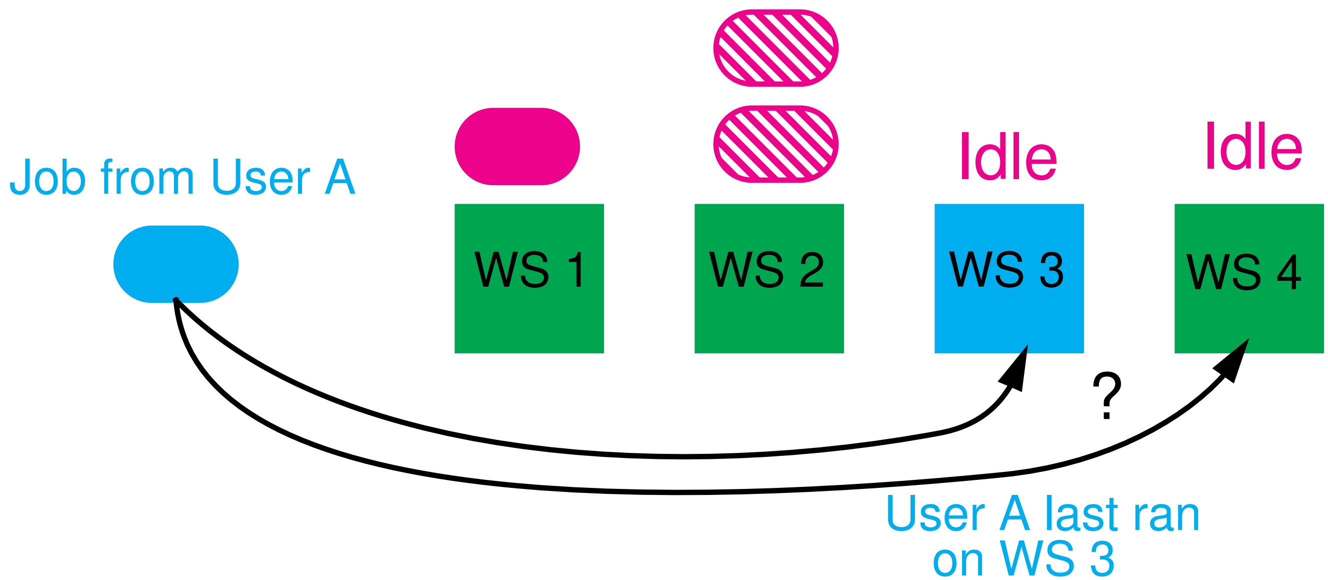
Scheduling Implications:

- Run light CPU jobs on “slower” machines
- Prediction (or user specification) for CPU-intensive jobs

Workstation Affinity

Scenario: Job is executed on least-loaded/fastest machine

Should jobs from same user have affinity for workstations?



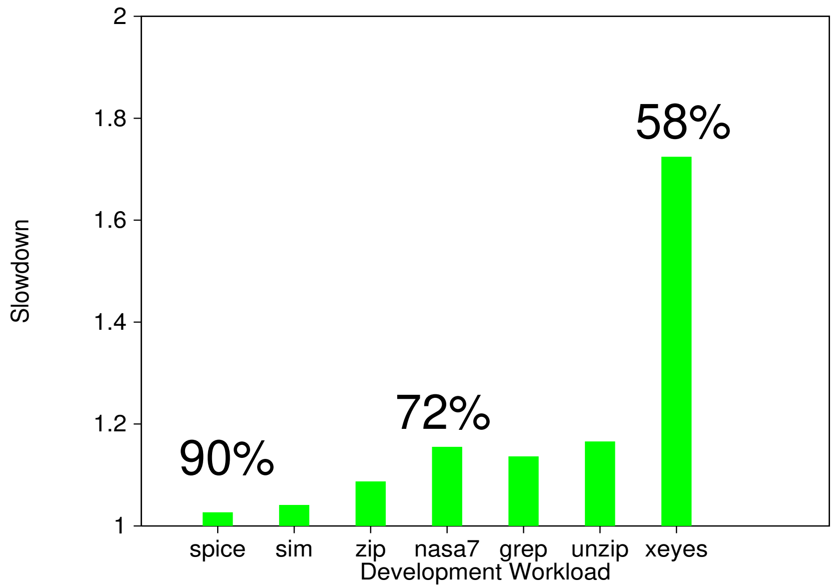
Experiment:

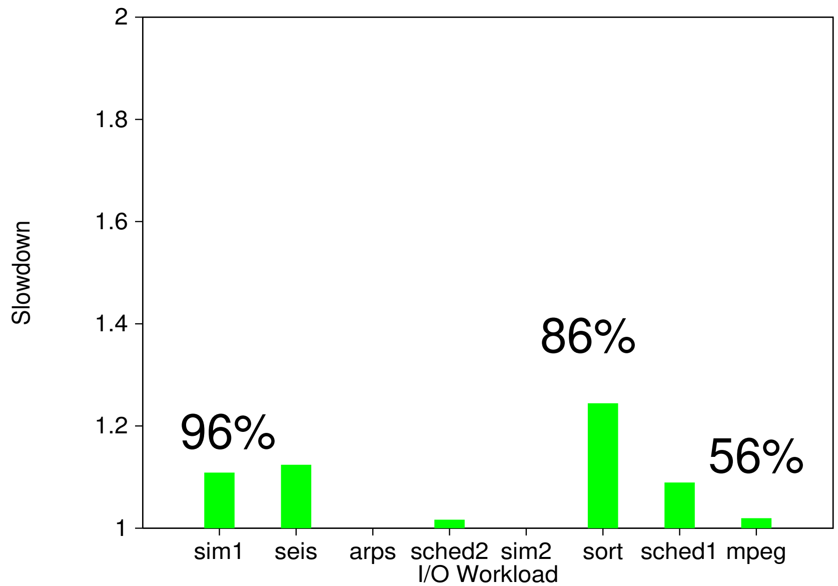
- Simulate execution on new workstation by flushing file cache between each job in workload

Workstation Affinity

Measurement Results:

- Development and I/O: Slowdown related to utilization





- Production and Interactive: Negligible slowdown

Scheduling Implications:

- Use processor affinity when scheduling jobs with large input sets

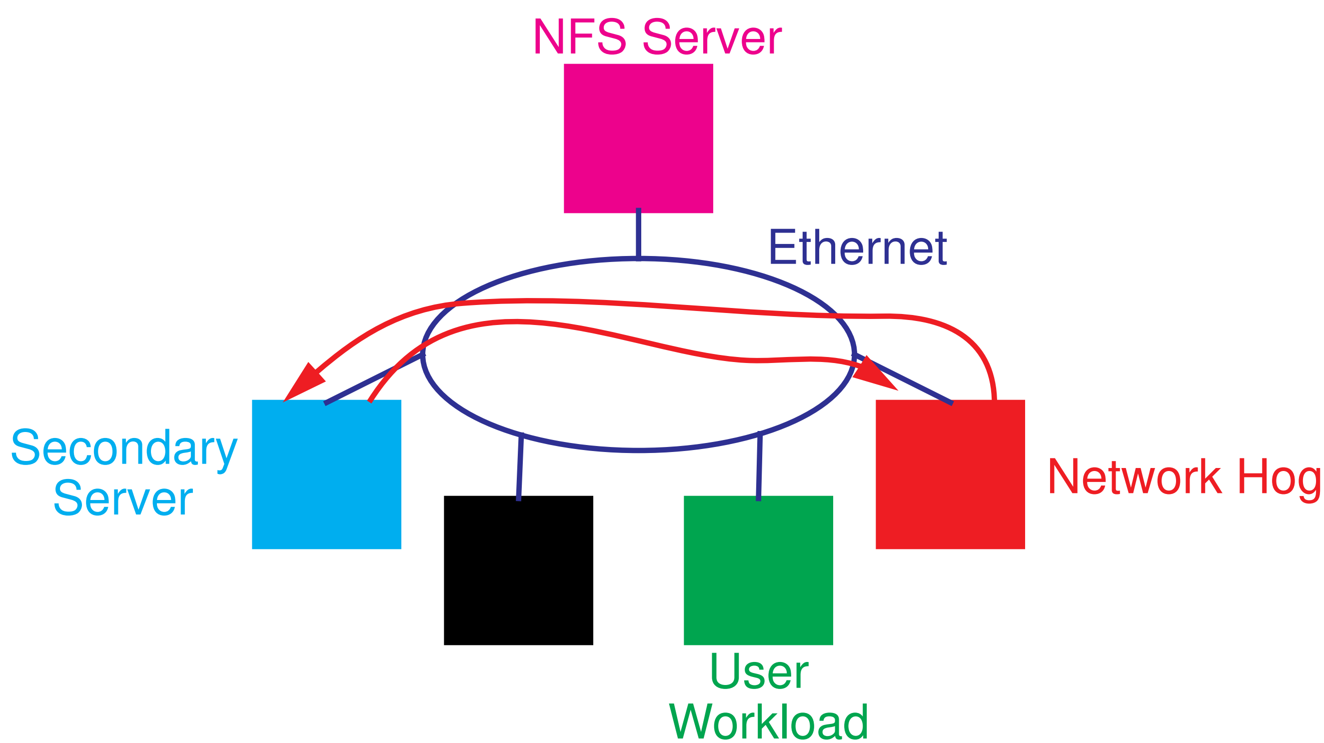
Network Sharing

Scenario: One process in cluster performing I/O over network

Should other I/O-bound jobs be scheduled?

Experiment:

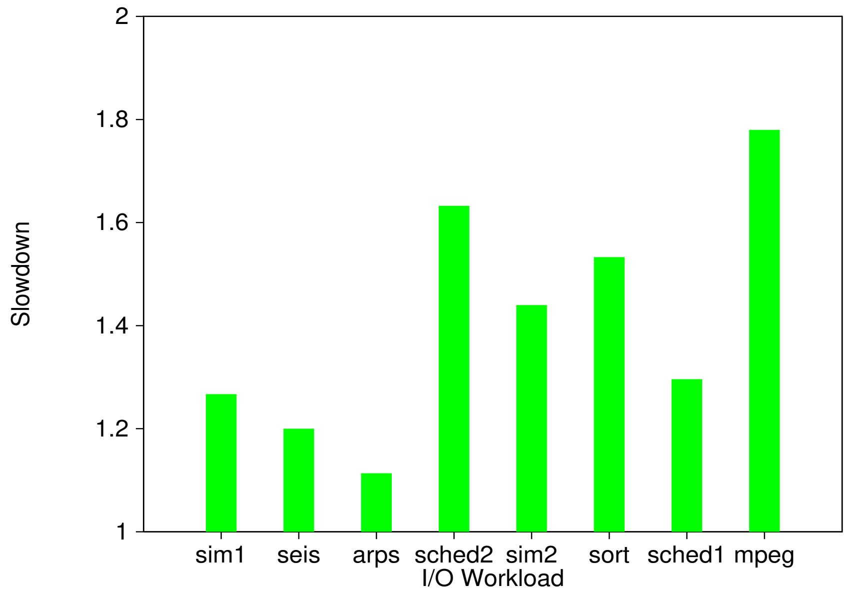
- Run workload with job repeatedly reading 100 MB file over Ethernet from secondary server

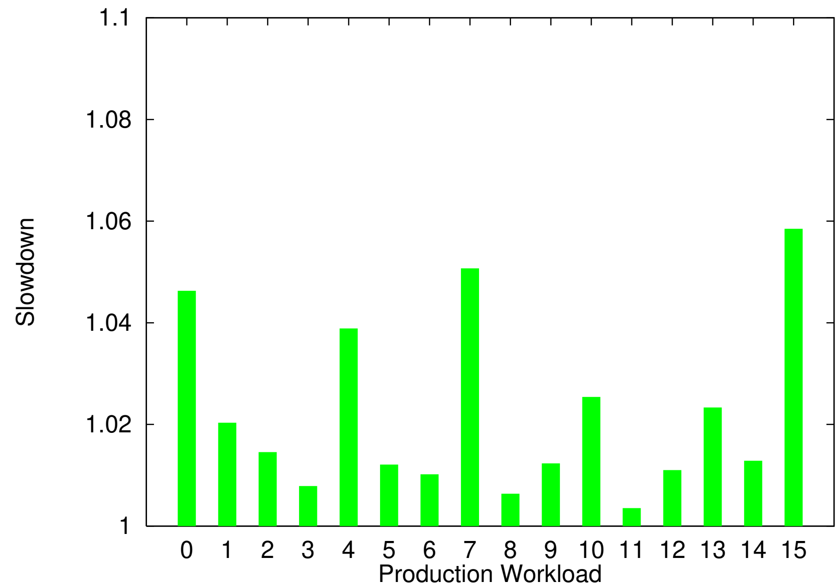


Network Sharing

Measurement Results:

- I/O and Production Workloads:





Scheduling Implications:

- Charge I/O-intensive workloads for network usage by lowering priority
- Avoid scheduling multiple I/O jobs simultaneously

Future Work

Develop scheduling policy that considers:

- Heterogeneity:
  - CPU jobs sensitive to model; others not
  - Find best model for CPU jobs; run others on available
- Locality:
  - I/O-intensive jobs are slowed down 1.0 - 1.8
  - Schedule user's I/O jobs on same workstation
- Network usage:
  - I/O-bound jobs affect all users
  - Decrease priority of I/O jobs; schedule one at a time

Knowlege of workload helpful when scheduling

- [Devarakonda&Iyer89] predict CPU, file, & memory from job names