

Runtime System Support for Software-Guided Disk Power Management

Seung Woo Son, Mahmut Kandemir

Department of Computer Science and Engineering

Pennsylvania State University

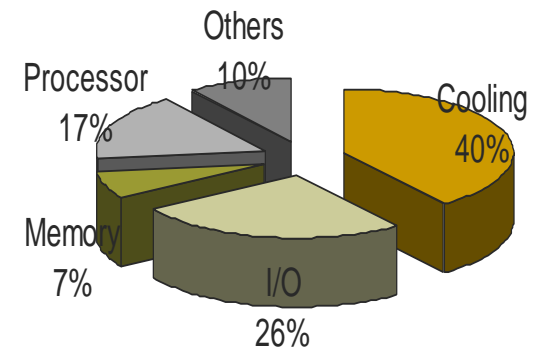
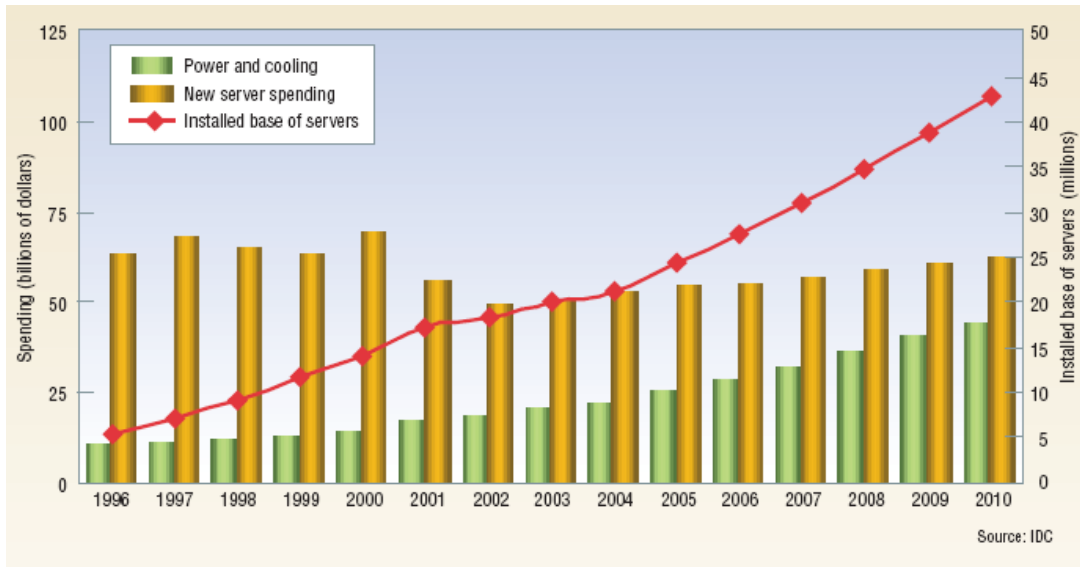
sson@cse.psu.edu

Cluster 2007

September 17-20 2007, Austin, TX, USA

Motivation

- Scientific applications are data-intensive
 - Out-of-core, visualization, checkpointing, etc.
- Energy cost is growing by **25%** annually.
- Disks can consume **~30%** of the total energy consumed by a server
 - Storage demands are growing by **60%** annually



Source: ACEED, 2003.

Prior Work

- OS or System Level Approaches
 - PA-LRU [HPCA'04], PB-LRU [ICS'04], Energy-aware caching and prefetching [USENIX'04], RIMAC [EuroSys'06]
 - MAID [SC'02], PDC [ICS'04]
 - Hibernator [SOSP'05], FS2 [SOSP'05]
 - EERAID [SIGOPS'04], PA-RAID [USENIX'06, FAST'07], Exploiting redundancy [SIGMETRICS'06]
 - Pros: suit for **server class workloads**
 - Cons: **reactive** in nature

Prior Work – Cont'd

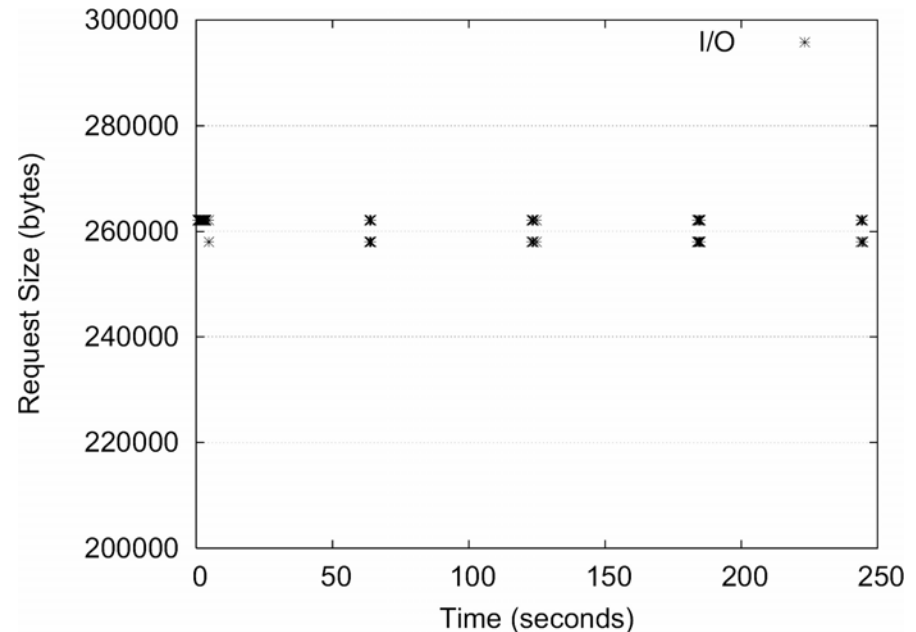
- Software-Guided Approaches
 - Code restructuring for laptop disks [PACT'02]
 - Compiler-directed disk power management for scientific applications [IPDPS'05]
 - Pros: **proactive** power management
 - Cons: **lack of runtime information** (suitable for single application)
- ⇒ **Runtime system support to make software-guided approach more effective**

Outline

- ✓ Motivation
- ✓ Prior Work
 - OS or System Level vs. Software-Guided
- Our Approach
 - Compiler support
 - Runtime system support
- Experimental Evaluation
- Conclusion

Compiler Support – Example

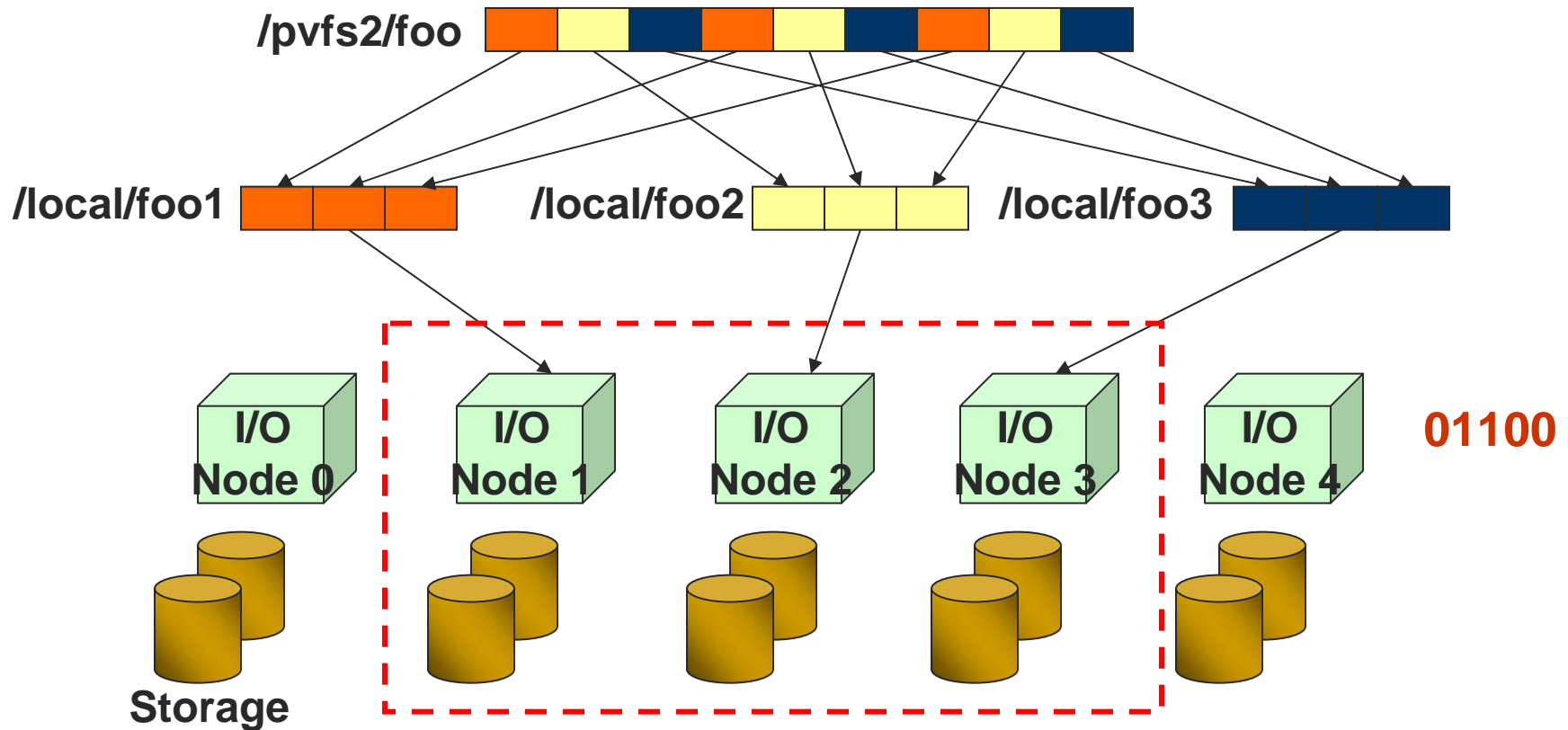
```
for ih = 1, R, 1
  MPI_File_read ( ... fh_A ...);
  for iv = 1, R, 1
    MPI_File_read ( .. fh_B ...);
    for i = 1, Nb, 1
      for j = 1, Nb, 1
        for k = 1, Nb, 1
          C[i][j] += A[i][k] * B[k][j];
        MPI_File_write ( ... fh_C ...);
```



- Many scientific applications exhibit explicit **I/O phase** followed by **computation phase**
 - I/O is done by MPI-IO
- How to detect the disk access pattern during I/O phase?
- What is the most appropriate management policy?

Disk Layout – File Striping

`MPI_File_read (“/pvfs2/foo”, start_offset, req_size);`



- * File striping (disk layout) is exposed to the software (e.g., compiler)
- * Disk usage is represented as bitmap (1: used, 0: unused)

Compiler Support – Insert Hints

```
for ih = 1, R, 1
  MPI_File_read ( ... fh_A ...);
  for iv = 1, R, 1
    MPI_File_read ( .. fh_B ...); ← Lower RPM
    for i = 1, Nb, 1
      for j = 1, Nb, 1
        for k = 1, Nb, 1
          C[i][j] += A[i][k] * B[k][j]; Boost RPM
    MPI_File_write ( ... fh_C ...); ←
```

```
for ih = 1, R, 1
  MPI_File_read ( ... fh_A ...);
  for iv = 1, R, 1
    MPI_File_read ( .. fh_B ...);
    set_speed (01100, 6000, 0);
    for i = 1, 3Nb/4, 1
      for j = 1, Nb, 1
        for k = 1, Nb, 1
          C[i][j] += A[i][k] * B[k][j];
    set_speed (01100, 15000, 1);
    for i = 3Nb/4+1, Nb, 1
      for j = 1, Nb, 1
        for k = 1, Nb, 1
          C[i][j] += A[i][k] * B[k][j];
    MPI_File_write ( ... fh_C ...);
```

* RPM: 6000 (lowest), 15000 (highest)

Compiler Support – Summary

- Determine **disk access patterns**
 - Walk through all I/O calls in the code
 - Exploit disk layout exposed to compiler
- Determine suitable **disk speeds**
 - Disk speed dictates loop splitting point
- **Transform** application code
 - Split loop in order to be proactive
 - Insert explicit `set_speed()` hints

Multiple Applications?

A1

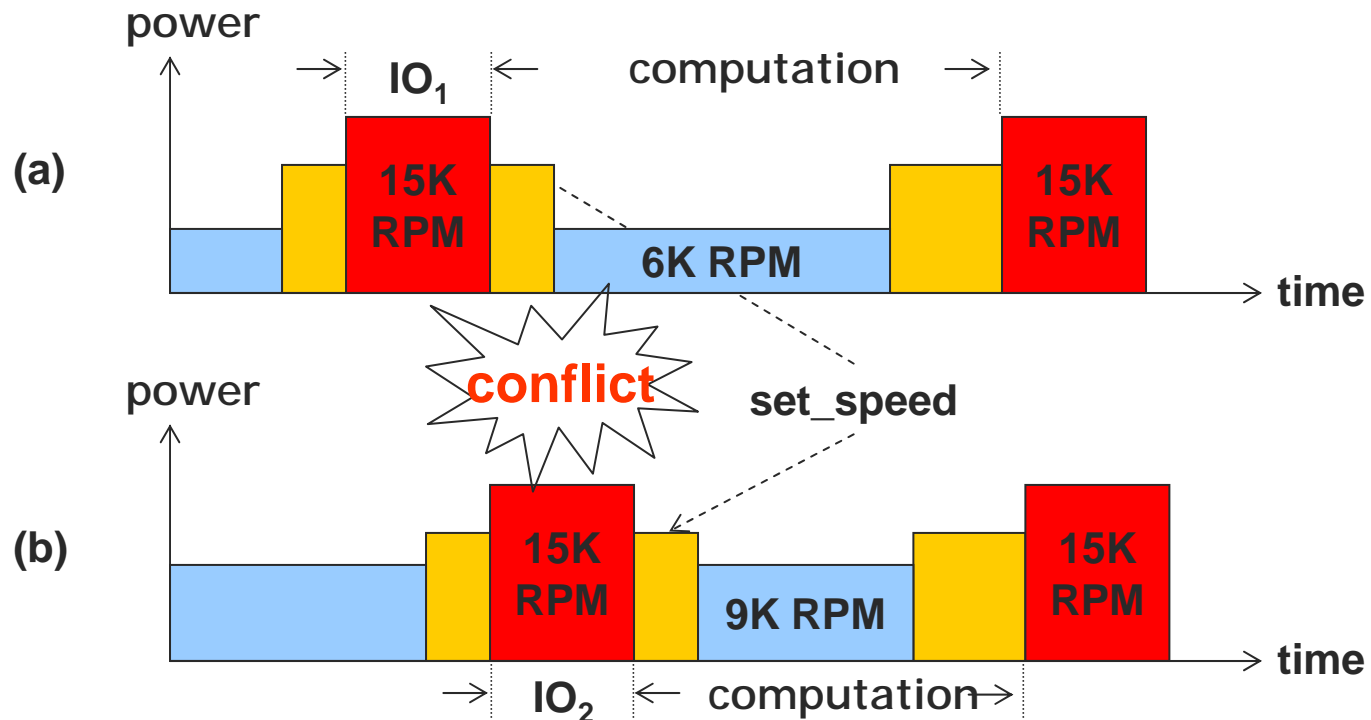
loop
body

```
set_speed ( 1110, 6K, 0 );  
... <Computation Phase>  
set_speed ( 1110, 15K, 1 );  
... <I/O Phase>
```

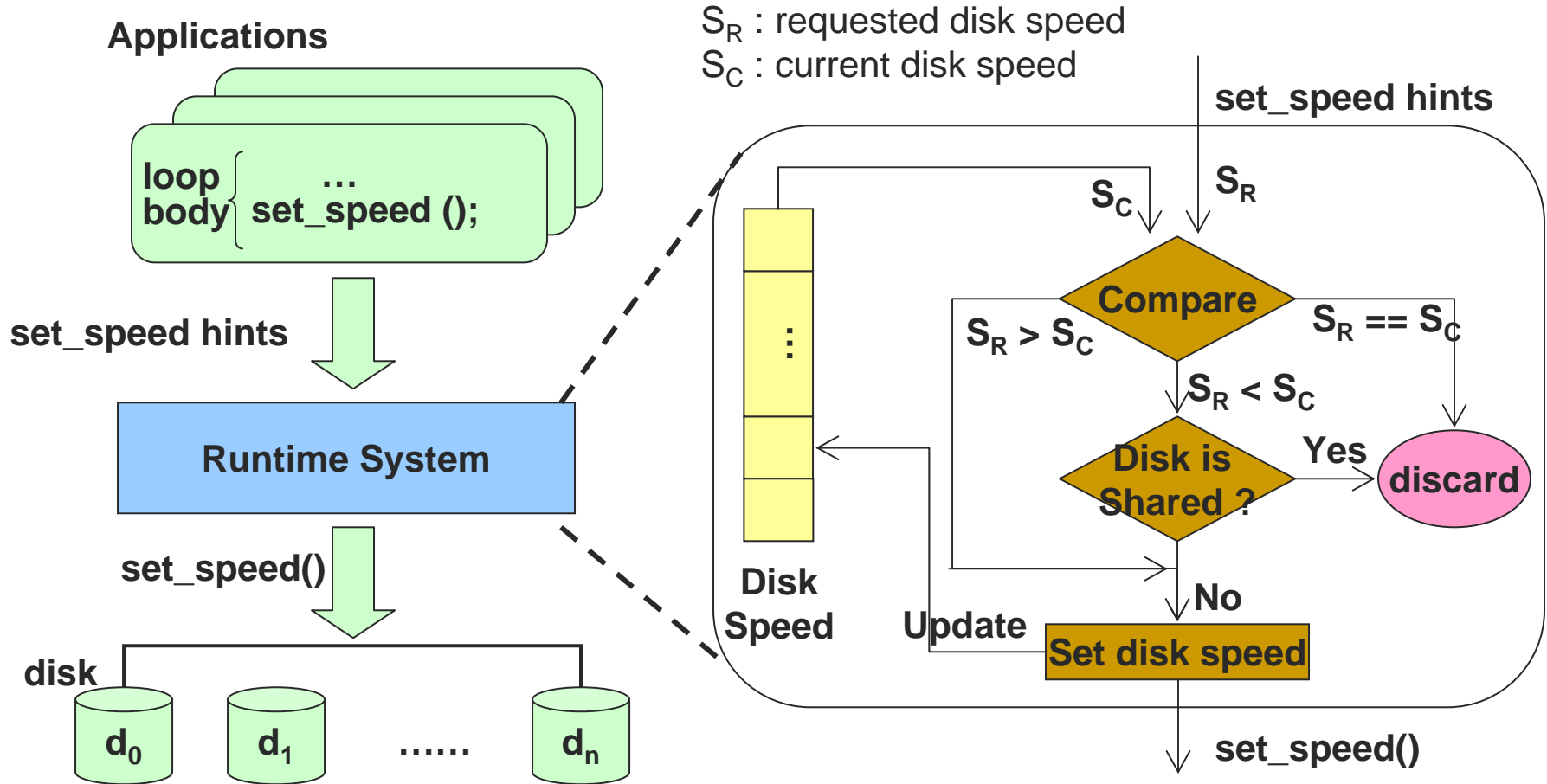
A2

loop
body

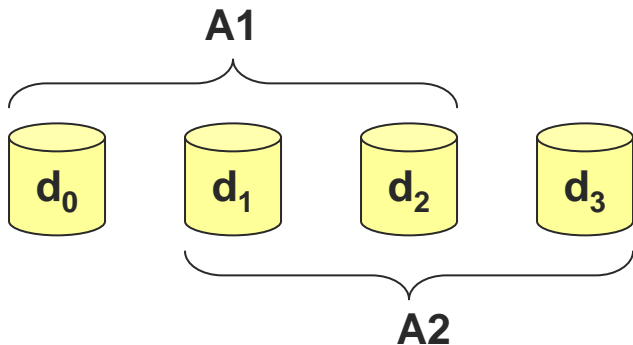
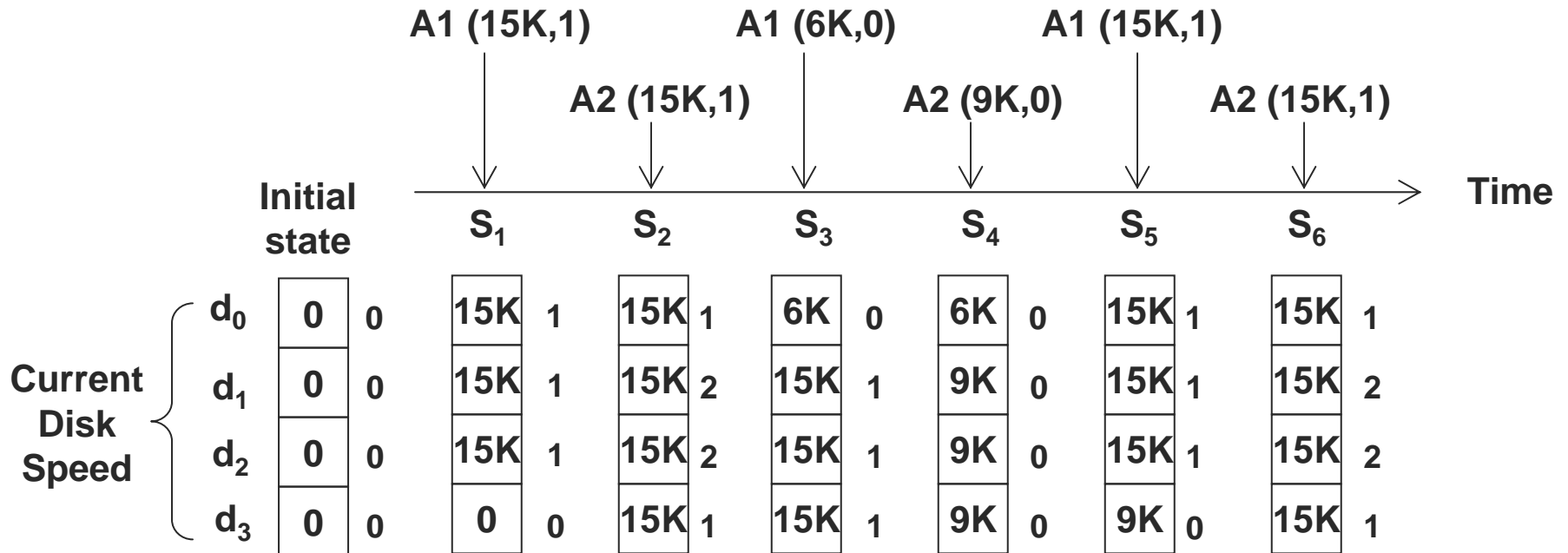
```
set_speed ( 0111, 9K, 0 );  
... <Computation Phase>  
set_speed ( 0111, 15K, 1 );  
... <I/O Phase>
```



Runtime System Support



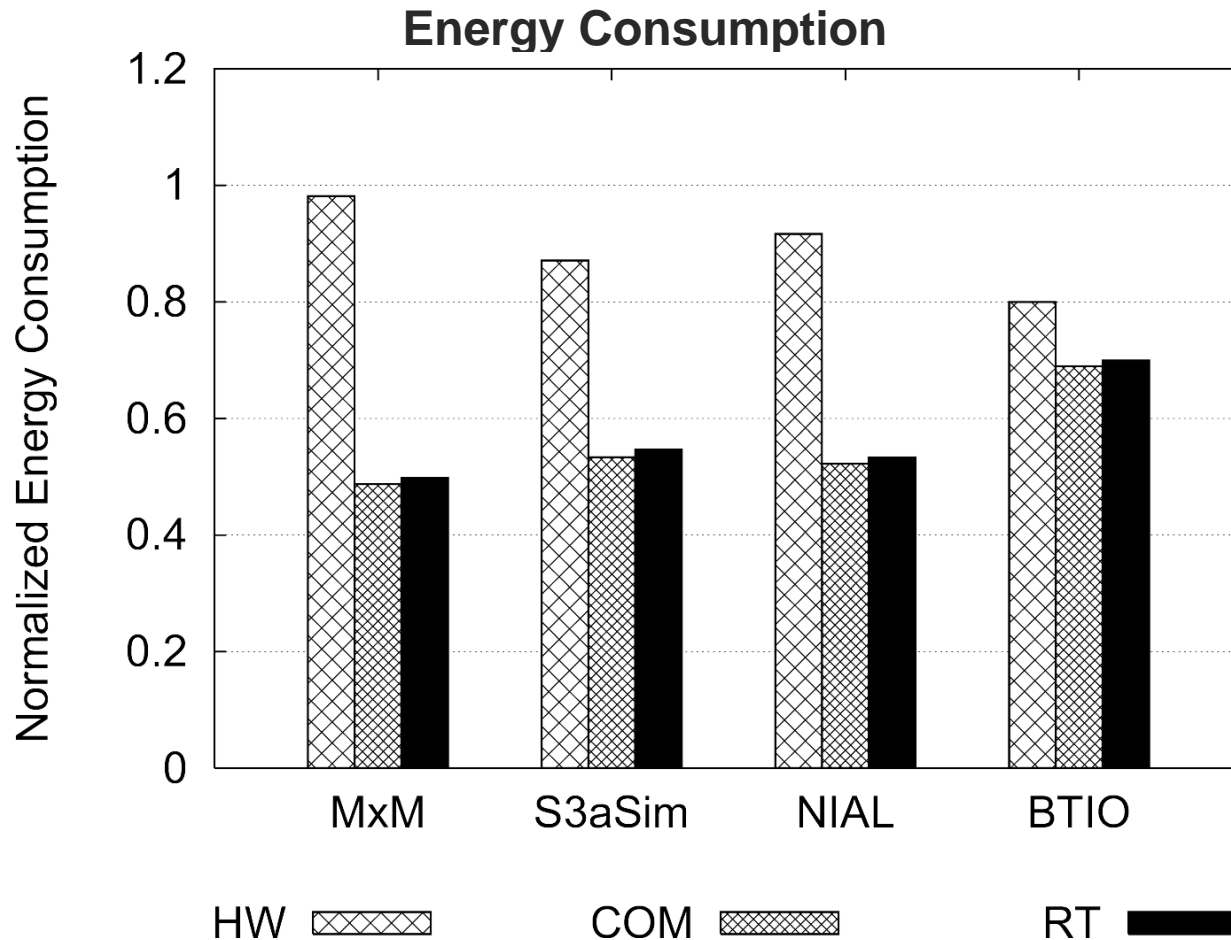
Runtime Support – Example



Experimental Methodology

- Ran all programs on Linux cluster
 - MPICH2 + PVFS2
 - I/O traces are collected using **Pablo** toolkit
- Used custom energy simulator
 - Emulate DRPM (Dynamic RPM) disks within PVFS2
 - Quadratic disk energy model: $P \propto (\text{RPM})^2$
- Evaluated three different schemes
 - **COM**, **HW**, and **RT** (this paper)
- Used 4 MPI-IO based applications
 - From different problem domains
 - Perform I/O for different purposes

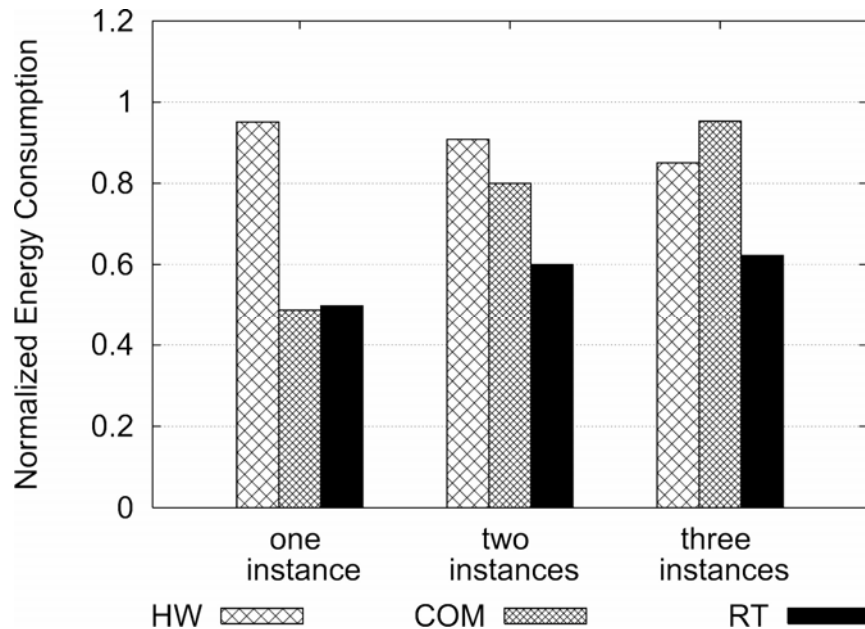
Result – Single Application



Average energy savings
HW: 14.5 %, COM: 44.2 %, RT: close to COM

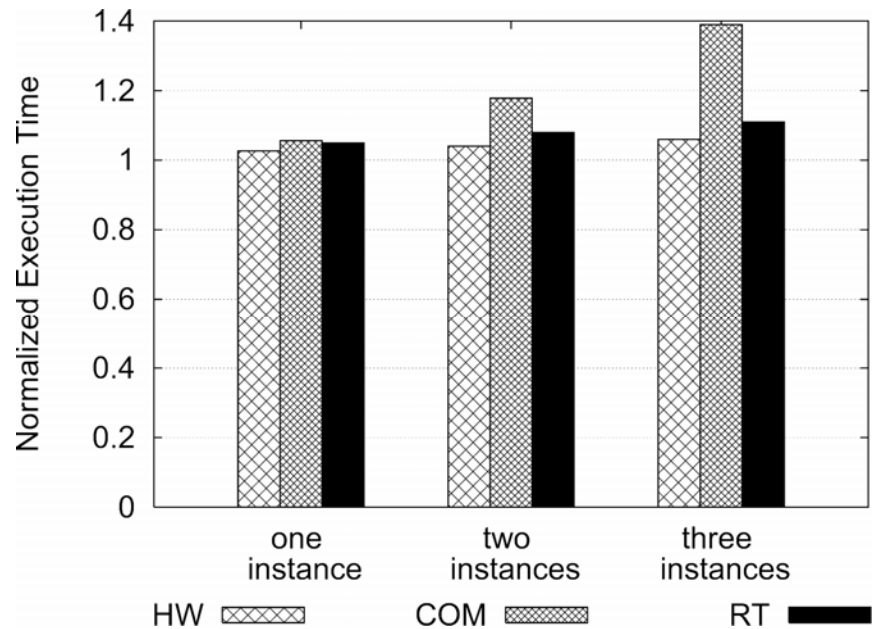
Result – Multiple Applications (1)

Energy Consumption



Average energy savings
HW: well adapt to workload
COM: significant energy increase
RT: 30.9 %

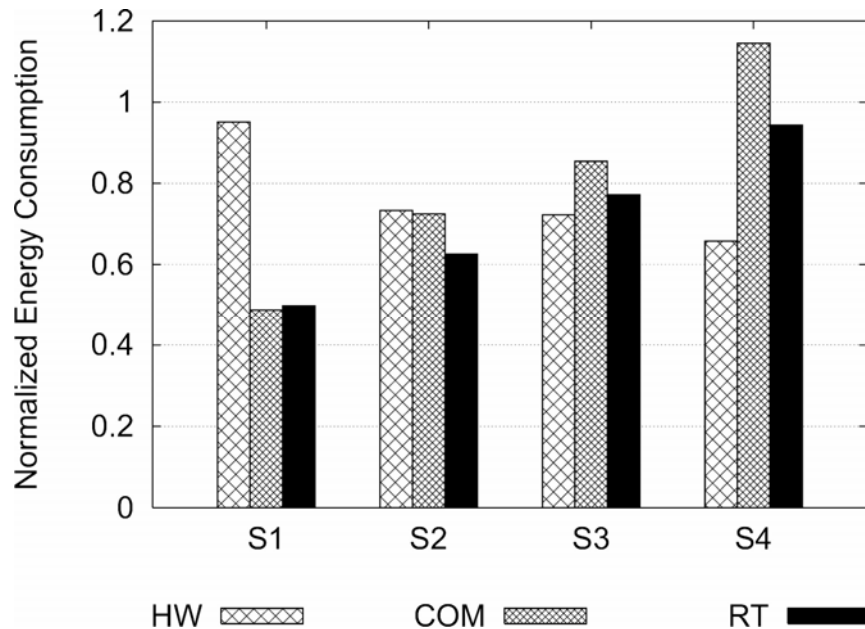
Execution Time



Average Performance Penalty
HW: 4.2 %
COM: significant slowdown
RT: ~5.0%

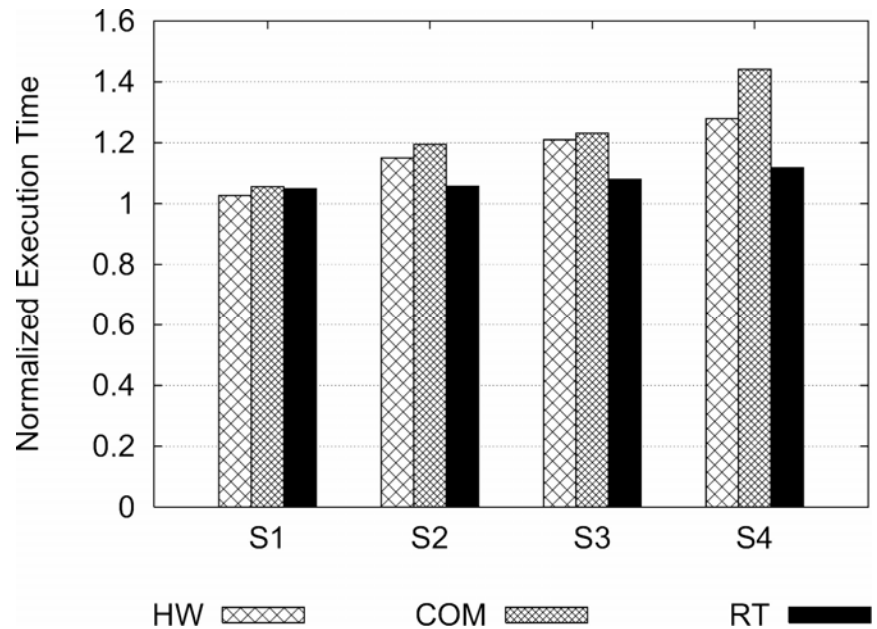
Results – Multiple Applications (2)

Energy Consumption



Average Energy Savings
RT: 19.4% (HW), 39.9% (COM)

Execution Time



Average Performance Penalty
HW: 16.7 %
COM: 23.1 %
RT: 7.6 %

Conclusion

- Proposed a **runtime system** support for software-guided disk power management
 - Compiler provides hints for future disk access patterns
 - Runtime system uses this information to make decisions that would be agreeable to all application running concurrently
- Evaluated our approach using PVFS2 file system
 - Used 4 I/O-intensive applications
 - **19.4%** and **39.9%**, on average, over pure-hardware and pure-software based approach



Thank you!

sson@cse.psu.edu