Towards the Incorporation of Dynamic Adaptation into Operating Systems: Adaptive Disk I/O

Patricia J. (Pat) Teller
Professor, Computer Science
The University of Texas at El Paso
pteller@utep.edu
Outline

• Brief Introduction to Research
• Adaptive Disk I/O
  – How can experiments on small systems help w.r.t. extreme-scale systems?
  – What kinds of adaptations are applicable to disk I/O management?
  – How can adaptive disk I/O management help in terms of performance and productivity?
• Future Work
**Goal**

**Enhanced Performance**

- Generalized ➔ Customized
- Fixed ➔ Dynamically Adaptable

**Resource Management**

**OS/Runtime Services**
Dynamic Adaptivity in Support of Extreme Scale

Small-scale MPs ➔ Extreme-scale MPs
Commodity OS ➔ Dynamically Adaptable OS

• Currently developing prototypes for small-scale MPs running Linux
• Looking forward to applying DAiSES research to compute nodes and I/O nodes of extreme-scale systems

<table>
<thead>
<tr>
<th>System</th>
<th>Compute Nodes</th>
<th>I/O Nodes</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNL Intel Paragon</td>
<td>1,840</td>
<td>32</td>
<td>58:1</td>
</tr>
<tr>
<td>ASCI Red</td>
<td>4,510</td>
<td>73</td>
<td>62:1</td>
</tr>
<tr>
<td>Cray Red Storm</td>
<td>10,368</td>
<td>256</td>
<td>41:1</td>
</tr>
<tr>
<td>BG/L</td>
<td>63,536</td>
<td>1,024</td>
<td>64:1</td>
</tr>
</tbody>
</table>
Dynamic Adaptivity in Support of Extreme Scale

**Vision**

- Small-scale MPs → Extreme-scale MPs
- Commodity OS → Dynamically Adaptable OS

- Currently developing prototypes for small-scale MPs running Linux
- Looking forward to applying DAiSES research to compute nodes and I/O nodes of extreme-scale systems
  - Performance isolation research is applicable to I/O servers
  - I/O stream throttling research is applicable to improving the performance of applications with I/O programmed similarly to MADbench
Challenges

Determining

- **What** to adapt
  - policy
  - parameter value

- **When** to adapt
  - under what circumstances
  - definition of a heuristic, function, or table-driven decision map

- **How** to adapt
  - selection of policy or parameter value
  - mechanism to affect the adaptation

- **How** to measure effectiveness of adaptation
  - metric
Disk I/O cause for concern? adaptation target?

From HEC I/O Workshop, August 2005: HPCS I/O and Storage Issues, David Koester (Mitre) and Henry Newman (Instrumental)
Disk I/O Performance is dependent on …

- I/O request access times: seeks + rotational latencies
  - I/O request generation pattern
  - Number and type of concurrent I/O-generating processes/threads
  - Disk I/O scheduler: policy and parameters
  - Application I/O requirements, e.g., latency, utilization, fairness
  - I/O controller: order in which requests are serviced by disk
- Data transfer rate
- File system
Disk I/O Performance
what can be adapted?

- I/O request access times: seeks + rotational latencies
  - I/O request generation pattern
  - Number and type of concurrent I/O-generating processes/threads
  - Disk I/O scheduler: policy and parameters
  - Application data delivery requirements, e.g., latency, utilization, fairness
  - I/O controller: order in which requests are serviced by disk

- Data transfer rate
- File system
Disk I/O Performance
what can be adapted?

- I/O request access times: seeks + rotational latencies
  - I/O request generation pattern
  - Number and type of concurrent I/O-generating processes/threads
  - **Disk I/O scheduler: policy** and parameters
  - Application data delivery requirements, e.g., latency, utilization, fairness
  - I/O controller: order in which requests are serviced by disk

- Data transfer rate
- File system
Space of Data Delivery Requirements and Disk Schedulers

Schedulers in Linux

DCFQ, SCAN-EDF

Deadline
EDF

Max

Utilization

DCFQ
SCAN
LOOK
SSTF
FCFS

CFQ
SFQ
YFQ

Fairness

Latency

Strict
Adaptation of Disk Schedulers

- Linux 2.6 includes four schedulers (Anticipatory, Deadline, CFQ, and noop) with boot-time and run-time selection
- Match the scheduler to application or system data delivery requirements
  - Real-time database → Latency guarantees: Deadline
  - Multimedia database → Fairness: CFQ
  - Throughput/disk utilization: SSF
Disk Scheduling Challenge

Applications with different I/O needs

Different disk system configurations

Different disk schedulers

No silver bullet
Disk Scheduling
Current Solution

Applications with different I/O needs
Different disk system configurations

Different disk schedulers

? No silver bullet
Linux Solution

Shortcomings

- Only one scheduler active at a time
- Queue draining time
- Does not provide
  - fair allocation of disk resource among multiple concurrently executing applications with different data delivery requirements
  - I/O performance isolation / insulation
A Case for Adaptive Disk Scheduling

- Operating systems and storage systems service concurrently executing applications with different data delivery requirements
- There are various I/O schedulers that satisfy these requirements
- No single scheduler is likely to satisfy all the applications’ data delivery requirements
- Server consolidation and virtualization compound this problem
New Solution: Virtual I/O Scheduler (VIOS) Subsystem

Scheduler of Disk Schedulers
- A queue is associated with each application
- Each application queue is associated with an instance of an I/O scheduler that can meet its data delivery requirements
- The VIOS distributes I/O service quanta among the scheduler instances (applications) via a round-robin algorithm
Fairness and VIOS

• How can fairness be achieved?

• What is fairness in disk scheduling?

• What does VIOS achieve by providing fairness?
Fairness and VIOS

• How can fairness be achieved?
  – Using quanta as in round-robin process scheduling

• What is fairness in disk scheduling?
  – Depends on definition of quantum
    • Number of requests dispatched
    • Number of bytes transferred
    • Amount of allocated disk time

• What does VIOS achieve by providing fairness?
Fairness and VIOS

• How can fairness be achieved?
  – Using quanta as in round-robin process scheduling

• What is fairness in disk scheduling?
  – Depends on definition of quantum
    • Number of requests dispatched
    • Number of bytes transferred
    • Amount of allocated disk time

• What does VIOS achieve by providing fairness?
Disk Service Quantum
Number of Requests - Example

Apps I, II, III

- Applications essentially only generate I/O read requests
- App 1 and App 2 generate a total of \( n \) 4KB requests each time they run
- App 3 generates a total of \( n \) 512KB requests
- **Service is fair in terms of number of requests but it is not fair in terms of disk time allocated**
- **Unpredictable execution time** – it depends on the I/O characteristics (size of requests and seek characteristics) of the application with which it runs
- **No performance insulation**

29% execution time increase
Our Solution: Compensating Round Robin w.r.t. Disk Time

Provides disk-time fairness
- As requests are dispatched and serviced, request service time is subtracted from the quantum
- If remaining quantum ≤ 0, then
  - no more requests are dispatched from the queue and
  - the quantum of the next round is shortchanged by the over-compensation of this round
- Unused quantum in a round is not carried forward
VIOS Subsystem

Fairness given Different Request Sizes

- Each application consists of eight threads to ensure queues with enough requests to consume quantum
- One application performs 4KB reads and the other performs the same number of 256KB reads
- Since each queue has the same quantum, irrespective of the request sizes, approximately 50% of disk time is allocated to each application
VIOS Subsystem
Fairness given Different Seek Characteristics

Fairness in terms of disk-time allocation
- Each application consists of three threads to ensure queues with enough requests to consume quantum
- One application performs a 1M sector seek for each read while the other performs a 64M sector seek for each read
- Since each queue has the same quantum, irrespective of the seek characteristics, approximately 50% of disk time allocated to each application

VIOS Disk Access Time Partitioning among Applications with Different Seek Characteristics
Fairness and VIOS

• What is fairness in disk scheduling?
  – Round-robin scheduling
  – Quantum: amount of allocated disk time
  – CFQ-CRR

• What does VIOS achieve by providing fairness, i.e., what are the benefits of providing fairness and are there penalties as well?
Performance Isolation given Different Request Sizes

- Each instance of Application 1 has a fixed 4KB request size.
- Each instance of Application 2 generates a different fixed request size in the range of 4KB to 512KB.
- The execution times of both applications increase due to concurrent disk usage.
- Note that the execution time of Application 1 is not affected by the I/O characteristics (request size) of Application 2.

VIOS Execution Times of Different Instances of Applications 1 and 2. Application 1 always has a Fixed Request Size, while the Request Size of Application 2 Varies.
Dynamic Adaptivity in Support of Extreme Scale

2006 LACSI Symposium
Workshop on Performance and Productivity of Extreme-Scale Parallel Systems
Santa Fe, NM

Predictable Performance given Varying Seek Characteristics

- Each instance of Program 1
  - generates only 4KB requests
  - has a fixed inter-request seek distance of 1M sectors

- Each instance of Program 2,
  - generates only 4KB requests
  - has a fixed inter-request seek distance in the range of 1M to 64M sectors

- The execution times of both applications increase due to concurrent disk usage

- Note that the execution time of Program 1 is not affected by the I/O characteristics (seek characteristics) of Program 2

VIOs Execution Times of Different Instances of Programs 1 and 2. Program 1 always has a Fixed Inter-request Seek Distance, while that of Program 2 Varies.

executed concurrently

executed alone
Implicit Benefits of Providing Disk-Time Fairness

- Different applications can be given different quanta, i.e., disk-time allocations $\Rightarrow$ Performance Differentiation

- Disk-time fairness results in
  - Predictable disk-time allocation
    - Given the number of non-empty queues, the inter-service time interval for any queue is known within a specified bound
    - Unpredictability hinders disk performance guarantees
    - Can support QoS
  - Disk performance insulation / isolation
    - The I/O characteristics of one application cannot impact the disk time allocated to another application
      - An application cannot monopolize the disk system
      - Not provided by contemporary disk schedulers
• Performance improvements: but what does performance mean?
  - Improved utilization / decreased execution time / throughput
    - Checkpoints: schedule sync and async requests differently: 80% of I/O usage associated with checkpoints (async)
    - Match disk I/O scheduler to application data delivery requirements
    - Throttle number of concurrent I/O-generating processes / threads
• Performance improvements: but what does performance mean?
  - **Provide fairness, performance isolation, performance predictability**
    - Virtualized I/O
    - Service guarantees / contracts
    - Repeatable performance
    - Load balance in terms of I/O
• Performance improvements: but what does performance mean?
  - Provide **fairness, performance isolation, performance predictability**
    - Virtualized I/O   Can increase execution time
    - Service guarantees / contracts
    - Repeatable performance
    - Load balance in terms of I/O
• Performance improvements: but what does performance mean?
  - Provide fairness, performance isolation, performance predictability
    - Virtualized I/O
    - Service guarantees / contracts
    - Repeatable performance
    - Load balance in terms of I/O

Facilitates performance debugging / tuning => PRODUCTIVITY
Virtualized I/O
Performance vs. Productivity

WITHOUT

PRODUCTIVITY: application development time (including performance debugging / tuning)

PERFORMANCE: application execution time

WITH

PRODUCTIVITY: application development time (including performance debugging / tuning)

PERFORMANCE: application execution time

productivity improvements are not always performance improvements
Virtualized I/O
Execution Time vs. Fairness (QoS)

Application execution times of the threads that finished first and last among 32 concurrent threads; 1000 random 4KB requests/threads

Maximum and average latency of requests with different schedulers; each thread accesses disjoint areas of the disk

CFQ-CRR(P) CFQ-CRR CFQ-Linux Deadline Anticipatory Noop

Execution Time (sec)

0 50 100 150 200 250 300 350

Shared Virtualized

Fairest but at cost of execution time

Latency (ms)

0 500 1000 1500 2000 2500 3000 3500 4000 4500 5000

Average Latency Maximum Latency

>5% above 1 second latency

2.3%

Smallest Minimum Latency
MADbench

- dSdC – calculates signal correlation derivatives; writes to files
- invD – Calculates pixel-pixel data correlation matrix D; reads data
- W – Calculate dense matrix-matrix multiplications; reads data
- Each phases goes through several loops
MADbench-invD

Dynamic Adaptivity in Support of Extreme Scale

10/17/2006 36

2006 LACSI Symposium
Workshop on Performance and Productivity of Extreme-Scale Parallel Systems
Santa Fe, NM
Performance vs. Productivity
Sometimes in Conflict

- Algorithms/schedulers are built to improve the performance of the resource
- Designed in isolation
  - Disk schedulers
- On extreme scale systems
  - We need schedulers that provide performance isolation
    - Separating workloads
    - Performance analysis – productivity
  - But we need schedulers that provide performance
    - Unfair I/O scheduler on BG/L I/O node makes performance analysis of compute tasks that much difficult, e.g., load imbalance
Software on Extreme Scale

• Predictable systems improve productivity
• Opportunistic / unpredictable systems may improve performance
• Adaptation is not just for improved performance but it must be for improved productivity as well!
• Systems must be virtualized
  – Fault tolerance
  – Performance analysis
  – Quality of service
Future Related Work - 1

• Using VIOS framework, provide both fairness and latency guarantees
• Apply fairness, performance isolation, and performance predictability to parallel I/O systems
• Apply the framework to storage systems that support parallel access
• For quality of service, weights can be defined. Consider how these weights can be used in large systems with thousands of processors and multiple applications.
Future Related Work - 2

- Further consider adaptation w.r.t. stateful vs. stateless resources
- Identify other ways that resource management adaptivity can be applied successfully
  - Virtual memory management
  - Small/large page allocation
  - Multi-core scheduling
- Consider how fairness and performance insulation / isolation can be guaranteed in other parts of the system
  - Lightweight operating systems - CPU performance
Acknowledgments

• DOE, Office of Science (Grant # DE-FG02-04ER25622)
• IBM Corporation for IBM Shared University Research (SUR) Grants and IBM Faculty Awards
• UTEP and UT System for STAR (Science and Technology Acquisition and Retention) Program Award
• Seetharami Seelam, Ph.D.
Refereed Publications - 1

Dynamic Adaptivity in Support of Extreme Scale


Theses, Dissertations, Technical Reports


Publications/Patents in Review or Preparation

- Seetharami Seelam, Andre Kerstens, and Patricia Teller, “I/O Throttling to Improve the Performance of a Cosmology Application.”
- Seetharami Seelam and Patricia Teller, “Disk Scheduling with Performance Objectives.”