Automatic I/O scheduler selection through online workload analysis

Ramon Nou, Jacobo Giralt, Toni Cortes
Outline

- Background
- Motivation
- Proposal
- Solution
- Evaluation
- Conclusion
Background

- IOLanes EU Project ([www.iolanes.eu](http://www.iolanes.eu))
  - Computers have more and more cores
  - There are unused cores
  - I/O is a noticeable bottleneck
  - We try to use unused CPU power to improve I/O
Background

I/O PATH

CPU

DISK
Background
Background
Background

![Diagram showing CPU, JOB1, JOB2, JOB3, and I/O PATH to DISK]

- CPU
- JOB1
- JOB2
- JOB3
- I/O PATH
- DISK
Background

![Diagram showing a CPU with three jobs (JOB1, JOB2, JOB3) and an I/O path to a disk](image)

- JOB1
- JOB2
- JOB3
- CPU
- I/O PATH
- DISK
Motivation

- I/O Scheduler
  - Fixed per Linux Distribution
  - Normally not changed
  - Four available: NOOP, Anticipatory, Deadline, CFQ

- Is there a “better one”? 
  - No, it depends on:
    - Workload, CPU, Disk, Memory
    - Depends on a lot of components
Motivation
Motivation

<table>
<thead>
<tr>
<th>BENCH 1 / DISK 1</th>
<th>BENCH 2 / DISK 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>NOOP</td>
<td>NOOP</td>
</tr>
<tr>
<td>AS</td>
<td>AS</td>
</tr>
<tr>
<td>DL</td>
<td>DL</td>
</tr>
<tr>
<td>CFQ</td>
<td>CFQ</td>
</tr>
</tbody>
</table>

Normalized Execution Time

- BENCH 1 / DISK 1: NOOP > AS > DL > CFQ
- BENCH 2 / DISK 1: NOOP > AS > DL > CFQ
- 5x increase

BENCH1 / DISK 2

BENCH2 / DISK 2
Motivation

![Diagram showing normalized execution time for different benchmarks and disk configurations.](image-url)
Motivation

Inverted
Proposal

- Use spare cores to analyse workload
  - Take small I/O traces (e.g. 5 seconds) and find differences
  - We use Dynamic Time Warping (DTW) Algorithm

![Graph showing %Similarity]

%Similarity
Proposal – I/O Trace Similarity
80% of the trace have more than 70% similarity
Solution

- Each t seconds, capture I/O trace, performance and actual I/O scheduler.
- Compare old trace with new trace
- **No Match:**
  - Store data
  - Update next trace pointer on previous found trace.
- **Match:**
  - Search from next trace pointer the most probable and beneficial scheduler in the future (5-10 seconds)
Solution

- Low-Level requirements
  - Capture I/O Traces
    - Integrated in Linux Kernel (blktrace)
      - Non-linear Overhead
    - Own-logger in elevator code (kernel mods)
      - Simple, no noticeable overhead
  - Capture performance
    - Normal system utilities
  - Change scheduler of a device
Evaluation

How well it works?

- Selects good I/O scheduler without interaction.
- We improve performance if the workload is dynamic
  - Benefits depend on the application
- Tested (analytical) with well-known traces (cello99, deasna)
- Tested in real environment (parallel apps)
- Microbenchmarks
- TPC-H, TPC-E, TPC-W, LinearRoad
- Applications inside VM
- Different Hardware
Evaluation - Microbenchmarks

- 4 Parallel processes (Kernel read and sequential reading)
  - Expected: Apparently random I/O
  - Found: Similarity, heterogeneous I/O
  - Better performance than the best static scheduler.
  - Learning phase: 15 iterations of the test (~ 10 hours)
Evaluation – Real Applications

- Applications Inside VM
  - 4 VM
  - 2 Different image formats (QCOW2, and RAW) - same application
  - I/O Behaviour totally different
  - Results:
    - Between 2 best schedulers with RAW format
    - Better performance with QCOW2 format
Evaluation
Evaluation

QCOW2 Format

- AS
- NOOP
- CFQ
- DL
- AUTOMATIC

reports/sec

0 - 20000 - 40000 - 60000 - 80000

Barcelona Supercomputing Center
Centro Nacional de Supercomputación
Evaluation

![Bar chart showing performance comparison between AS, NOOP, CFQ, DL, and AUTOMATIC.](RAW Format)
Evaluation - Overheads

- Only works when I/O is ongoing
  - (i.e. No CPU Bound phases).
  - Low priority threads.
    - We have additional mechanism to reduce the interferences but they are not needed.

- With an overloaded LinearRoad => 5% of improvement.

- Always near the 2nd best scheduler
Conclusions

- The method works in a non-intrusive way
  - Tested extensively

- Major problem is the need of kernel modifications
  - Solved with an user-mode version using blktrace
  - Being used in several places. (But still need testers ;))
Thank you!

For further information please contact

ramon.nou@bsc.es