Understanding scalability and performance requirements of I/O intensive applications on future multicore servers

Shoaib Akram, Manolis Marazakis, and Angelos Bilas

Presentation: Polyvios Pratikakis

Foundation for Research and Technology – Hellas (FORTH)
Institute of Computer Science (ICS)
Demand for Data Grows Fast

• ...Faster than storage capacity
  – Storage capacity grows faster than Moore’s law
• Need to store and can store a lot of data
• Can we access and process data at the same rate?

Figure 5: The Emerging Gap
Information Creation > Storage Available

Source: IDC Digital Universe Study, sponsored by EMC, May 2010

A Decade of Digital Universe Growth: Storage in Exabytes

Today Low “I/O Density”

• Typical server configuration
  – 4-8 cores
  – 8-32 GBytes
  – 2-4 disks
  – 2 cores to keep up with 1 disk-performance

• Emerging needs: process large amounts of data
  – Bring data to memory, process (data centric)
  – Compared to compute from main memory
  – Keeping up with data growth requires increasing I/O density

• So far slow disks limitation to increasing I/O density
Towards Higher “I/O Density”

• New device technologies (SSDs) allow higher access rate with fewer devices and better latency (IOPS)
• This allows and requires increasing #cores per server
• Broadly, what is the role of storage I/O?
Goals

• This presentation centered around 3 questions
  1. Does I/O scale with cores?
  2. How much I/O in ten years?
  3. How energy (in)efficient is application I/O?

• Contribute to methodology
  – How can we characterize I/O across applications?
  – We measure using real applications, workloads
  – We project to large numbers of cores
Outline

✓ Motivation and Goals
  • Metrics & Methodology
  • Applications & Platforms
  • Does I/O scale?
  • How much I/O?
  • How much Energy?
  • Conclusions
Methodology

• Get a number of applications
  – Data-centric, I/O intensive
• Figure out parameters and configurations
• Run them on a real system
• Examine how much I/O they require
• Methodology is interesting by itself
cpi: Abstract I/O behavior

- We use cycles per I/O (cpi) as a metric
  - Used in the past in certain cases
  - Recently used more in networking as cycles per packet
- System-level metric
  - Not related to application output
  - Includes both CPU and I/O
- Computing cpi
  - Calculate execution time breakdown
  - Count number of I/Os – 512 bytes
  - \( cpi = \frac{(\text{system} + \text{user})}{\#\text{ios}} \)
- Ignore idle and iowait time
  - Energy proportionality -> idle+iowait not a problem
  - Not straight-forward to distinguish idle form iowait
Use Experimental Approach

- Server-type specs with aggressive I/O subsystem
  - 24 SSDs, 4x LSI controllers, 6 SSDs per controller
- Two configurations: More, less aggressive (CPU, I/O)

<table>
<thead>
<tr>
<th>DISKS</th>
<th>SSDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Intel Xeon E5620 (Quad-core)</td>
<td>2 Intel Xeon E5405 (Quad-core)</td>
</tr>
<tr>
<td>No Hyper-threading</td>
<td>Hyper-threading</td>
</tr>
<tr>
<td>8 GB RAM</td>
<td>12 GB RAM</td>
</tr>
<tr>
<td>1 Storage Controller (8 Disks)</td>
<td>4 Storage Controllers (24 SSDs)</td>
</tr>
<tr>
<td>XFS on Hardware RAID 0</td>
<td>XFS on Software RAID 0</td>
</tr>
<tr>
<td>1 GB/s Storage Throughput</td>
<td>6 GB/s Storage Throughput</td>
</tr>
<tr>
<td>CentOS distribution; 2.6.18</td>
<td>CentOS distribution; 2.6.32</td>
</tr>
</tbody>
</table>
Benchmarks and Applications

• Applications from diverse domains
  – Benchmarks (zmIO, fsmark, IOR)
  – OLTP workloads (TPC-C, TPC-E)
  – NoSQL Data Stores (HBase, BDB)
  – HPC Domain (Ferret, BLAST)
  – Backend Applications (Deduplication, Psearchy, Metis)
  – Data Streaming (Borealis)
  – Business Intelligence (Tariff)

• Applications are tuned to perform large amounts of I/O
  – Applications and runtime parameters available at [www.iolanes.eu]
Two Broad Categories

• Sweep
  – Do a pass over the data to calculate metadata
  – E.g. indexing, deduplication, streaming

• Metadata
  – Quickly calculate metadata
  – Operate mostly from metadata and only access necessary data
  – OLTP, OLAP, key-value stores, image processing
- Range from 1K to 2M
- cpio not appropriate in absolute terms to say “good” or “bad”
  - Memory caching plays an important role
- Captures behavior assuming same amount of work to devices
- Can be as tricky as speedup
I/O Characterization

• Breakdown of execution time (user, system, idle, iowait)

**DISKS**

Average system time: 3%

**SSDS**

Average system time: 26%
cpio Sensitivity to Devices

• cpio largely independent of configuration
• Spinning effects in IOR, HBase and TPC-E
Outline

✓ Motivation and Goals
✓ Applications and Metrics
✓ Test System Configurations
  • Does I/O Scale?
  • How much I/O?
  • How much Energy?
  • Conclusions
Does I/O Scale?

- cpio does not scale with cores
- Overhead/work for a single I/O increases – ideally constant
- hw threads = cores (80% of perf at much less area)
Application Scaling

- Hard to saturate cores with a single application
- Much bigger problem in the future
Outline

✓ Motivation and Goals
✓ Applications and Metrics
✓ Test System Configurations
✓ I/O Scalability Trends
  • How much IO?
  • How much Energy?
  • Conclusions
We Project via cpio

• How do we calculate I/O requirements with increasing #cores?
• Once we know cpio
• Available cycles = #cores*freq
• Divide cycles with cpio
  – We get IOPS requirement for given #cores
  – Multiply with I/O size to get required I/O xput for #cores
• Which cpio do we use?
Various Projection Scenarios

• cpio
  – Measured with 16 cores (optimistic)
  – Measured with 1 core (desired)
  – Linear projection to N cores (pessimistic)

• CPU Utilization
  – 30%-40% range
    • Low utilization common today
  – 80%-100% (full) utilization
    • Desirable for better efficiency
## How much I/O?

### Millions of IOPS for 4096 Cores

<table>
<thead>
<tr>
<th>utilization&amp;cpio</th>
<th>Average</th>
<th>TPC-E</th>
<th>HBase</th>
<th>PSearchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low&amp;Projected</td>
<td>7.5</td>
<td>9</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>High&amp;Projected</td>
<td>59</td>
<td>14</td>
<td>107</td>
<td>65</td>
</tr>
<tr>
<td>Low&amp;Today</td>
<td>476</td>
<td>535</td>
<td>563</td>
<td>2207</td>
</tr>
<tr>
<td>High&amp;Today</td>
<td>818</td>
<td>743</td>
<td>1405</td>
<td>4509</td>
</tr>
<tr>
<td>Low&amp;Desired</td>
<td>969</td>
<td>1743</td>
<td>644</td>
<td>2540</td>
</tr>
<tr>
<td>High&amp;Desired</td>
<td>1810</td>
<td>2469</td>
<td>1652</td>
<td>5941</td>
</tr>
</tbody>
</table>

3/1/16

MASCOTS 2012
I/O Bandwidth

• Once we know cpio
• \#ios = (#cores*freq) / cpio
• required I/O bw = \#ios * iosize
• Per core
  – 100K – 500K IOPS
  – 1 GBit/s
How much I/O as #Cores Increases?

- GB/s on Y-axis
- Low utilization (left) and High utilization (right)
I/O Requirements: Quick Summary

• Requirements per core
  – 100K IOPS
  – 1 GBit/s I/O bandwidth
  – 1 GBytes/s memory bandwidth

• At 128 cores
  – 10M IOPS
  – 10 GBytes/s I/O bandwidth
  – 100 GBytes/s memory bandwidth

• Difficult to saturate systems with single application

• More work per I/O as # cores increases
Energy Requirements

• cpio easy to convert to energy
• BkWH to sweep over 35 ZettaBytes of data
• Calculate number of cores and translate to energy
  – 0.5W/core at 4K cores/server (2.5KW/server)
  – Idle power 0% -> perfect energy proportionality

<table>
<thead>
<tr>
<th>Power Assumptions</th>
<th>Projected cpio</th>
<th>Today’s cpio</th>
<th>Desired cpio</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5 Watts per core (2.5 KW)</td>
<td>29</td>
<td>0.27</td>
<td>0.175</td>
</tr>
<tr>
<td>1.25 KW</td>
<td>17.5</td>
<td>0.16</td>
<td>0.107</td>
</tr>
<tr>
<td>2006 Level (0.675 KW)</td>
<td>9.5</td>
<td>0.09</td>
<td>0.057</td>
</tr>
</tbody>
</table>

• Between 0.1 – 0.3 BkWH for a single pass
  – A city of 200K, energy for a year
• Close to energy star projections
  – But we are using applications whereas they use market growth
Conclusions

• A methodology for characterizing I/O
• Scalability of I/O stack with cores
  – More overhead per I/O as number of cores increase
  – Contention and interference in the system stack
  – A single server is not saturated
• I/O requirements
  – At 128 cores (10M IOPS)
• Opportunity to save energy by better scalability
Thank you for your attention!

Questions?

Polyvios Pratikakis for Shoaib Akram

{polyvios, shbakram}@ics.forth.gr

Foundation for Research and Technology – Hellas (FORTH)
Institute of Computer Science (ICS)
Hyper-threading

- Effectively h/w threads = cores (for these apps)
- 80% of perf at much less area
Memory Bandwidth

- Today systems overprovisioned for memory
  - Base: 1.3 GB/s/core
  - At 0.8 GB/s/core only 25% increase in cpio
- Going forward: 1 Gbytes/s/core memory bandwidth