Bridging the complexity gap: Tracing and Replaying I/O

UIOP 2017, Hamburg, Mar. 22nd

Jean-Thomas Acquaviva, DDN Storage
Complexity: E.g 800 nodes Fat Tree

1PF Compute Cluster

EDR Infiniband N/w

Courtesy Philip Brighten, U. Illinois
Hierarchy mechanically increases predominance of patterns in the performance equation.
Spatial Locality Patterns

(a) Decomposed mesh
(b) File mapping

99% of the time the IO sub-system is stressed below 30% of its bandwidth
70% of the time the system is stressed under 5% of its peak bandwidth

Argone lab.
P. Carns, K. Harms et al., *Understanding and Improving Computational Science Storage Access through Continuous Characterization*
File contention is temporal
Spatial and Temporal Patterns

Pattern detection and prefetch logic

Sequential access

Strided access

Strided access with variable data block length
Storage devices are sensitive to pattern → increased life expectancy

1) Vendors estimate life time on 4K random write pattern
2) IME limits write amplification
3) Combo: better performance + longer life

Testbed:
Burning SSD with different patterns + monitoring SMART counters

<table>
<thead>
<tr>
<th>ID#</th>
<th>ATTRIBUTE_NAME</th>
<th>FLAG</th>
<th>VALUE</th>
<th>WORST</th>
<th>THRESH</th>
<th>TYPE</th>
<th>UPDATED</th>
<th>WHEN_FAILED</th>
<th>RAW_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Reallocated_Sector_Ct</td>
<td>0x0033</td>
<td>093</td>
<td>093</td>
<td>010</td>
<td>Pre-fail</td>
<td>Always</td>
<td>-</td>
<td>62</td>
</tr>
<tr>
<td>9</td>
<td>Power_On_Hours</td>
<td>0x0032</td>
<td>099</td>
<td>099</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td>-</td>
<td>1986</td>
</tr>
<tr>
<td>12</td>
<td>Power_Cycle_Count</td>
<td>0x0032</td>
<td>099</td>
<td>099</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td>-</td>
<td>7</td>
</tr>
<tr>
<td>177</td>
<td>Wear_Leveling_Count</td>
<td>0x0013</td>
<td>001</td>
<td>001</td>
<td>000</td>
<td>Pre-fail</td>
<td>Always</td>
<td>-</td>
<td>6242</td>
</tr>
<tr>
<td>179</td>
<td>Used_Rsvd_Blk_Cnt_Tot</td>
<td>0x0013</td>
<td>093</td>
<td>093</td>
<td>010</td>
<td>Pre-fail</td>
<td>Always</td>
<td>-</td>
<td>62</td>
</tr>
<tr>
<td>181</td>
<td>Program_Fail_Cnt_Total</td>
<td>0x0032</td>
<td>100</td>
<td>100</td>
<td>010</td>
<td>Old_age</td>
<td>Always</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>182</td>
<td>Erase_Fail_Count_Total</td>
<td>0x0032</td>
<td>100</td>
<td>100</td>
<td>010</td>
<td>Old_age</td>
<td>Always</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>183</td>
<td>Runtime_Bad_Block</td>
<td>0x0013</td>
<td>093</td>
<td>093</td>
<td>010</td>
<td>Pre-fail</td>
<td>Always</td>
<td>-</td>
<td>62</td>
</tr>
<tr>
<td>187</td>
<td>Reported_Uncorrect</td>
<td>0x0032</td>
<td>099</td>
<td>099</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td>-</td>
<td>21</td>
</tr>
<tr>
<td>190</td>
<td>Airflow_Temperature_Cel</td>
<td>0x0032</td>
<td>059</td>
<td>048</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td>-</td>
<td>41</td>
</tr>
<tr>
<td>195</td>
<td>Hardware_ECC_Recovered</td>
<td>0x001a</td>
<td>199</td>
<td>199</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td>-</td>
<td>21</td>
</tr>
<tr>
<td>199</td>
<td>UDMA_CRC_Error_Count</td>
<td>0x003e</td>
<td>100</td>
<td>100</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>235</td>
<td>Unknown_Attribute</td>
<td>0x0012</td>
<td>099</td>
<td>099</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>241</td>
<td>Total_LBAs_Written</td>
<td>0x0032</td>
<td>099</td>
<td>099</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

346288758224
11 Limited Write amplification → increased life expectancy

1) Assess devices Life expectancy
2) Understand deprecation rate → preventive maintenance
Parallel File System are sensitive to patterns

- Read
- Write
- Small file
- Large file
- High client concurrency
- Random
- Sequential
- Large IO
- Small IO
- metadata intensive
- IO intensive
- to many files
- to one file
- aligned
- not aligned
- highly threaded

WRITE DISTRIBUTION FOR MULTI-DISCIPLINARY HPC CLUSTER
EVEN LARGE HPC SITES DRIVE A LOT OF SMALL I/O

90% OF ALL I/O IN TYPICAL HPC DATA CENTERS IS <32KB IN SIZE
Community answers

- Real applications
- Synthetic benchmarks
- Analytical Model
- Mini-app
Community answers

► Real applications
  – Best estimation of production run
    • *Is the application going to run alone?*
  – Costly: require expertise and topnotch engineering

► Synthetic benchmarks
  – IOR
  – IOZONE
  – MDTEST
    • *Do not capture temporal locality*
Analytical Model: NEural Simulation Tool (NEST)

NEST, computational neuro-science

- Dynamics of interactions between nerve cells
- First step of wiring a neural network
- Next step simulate the network of spiking point neurons

- Developed for both local small experiments or deployment on leading super-computer for extreme-scale simulations
  → MPI + OpenMP
- I/O pattern burst of write at the end of every simulation step

Early Evaluation of the "Infinite Memory Engine" Burst Buffer Solution
WOPSSS '16, 2016, Frankfurt

Wolfram Schenck
Faculty of Engineering and Mathematics Bielefeld University of Applied Sciences Bielefeld, Germany

Salem El Sayed, Maciej Foszczynski, Wilhelm Homberg, Dirk Pleiter
Jülich, Germany Jülich Supercomputing Centre Forschungszentrum Jülich
NEST experimental results

Platform – JUDGE: IBM iDataPlex NodeDual Xeon Westmere – 48 GB RAM – Network IB-QDR
NEST modeling and analysis

→ hum... POSIX2IME and POSIX2DEVNULL show nearly identical behavior

If computation fully overlap I/O bandwidth is irrelevant

1 bit of I/O per FLOP is a rule of thumb [see Jim Gray]
Analytical model

► Allow to understand
  – Bottleneck isolation
    • Optimize both application and architecture

► Difficult to scale
  – Difficult to extend to an application portfolio
  – Difficult to model complex workload
    • Resource sharing
Mini-apps: Brain Simulation and Neuromapp

Brain simulator are large SW
- 3 decades of development
- 500 KLOC + DSL + src2src compiler

NeuroMapp
- Mini-app framework
- Each mini-app (1KLOC) represents a single critical neuron scientific algorithm
- Stay tuned, results to be published

Mini-apps can be assembled together
- Form the skeleton of the initial scientific application
- Experiment work-flow optimization opportunities

See github.com/PETTT
See github.com/BlueBrain/neuromapp
Mini-App: summary

► Easy to deploy
► Provide ‘reasonable’ estimations
► Provide ‘reasonable’ insight
► Costly to develop
  – Require code maintenance
  – Evolve jointly with the core application
DIO-pro

MPI OPERATIONS

- open/close
- write

MPI rank

Time (s)
I/O Profiling with DIO-pro

► Capture I/O traffic
  - Cope with high I/O loads (overhead <1%)
  - Support Posix and MPI-IO

► Characterize I/O patterns
  - Build a distributed accurate clock

► Evaluate I/O efficiency

► Extrapolate performances
  - Architectural prospection
  - Estimate IME perf. impact

► DIO-pro is not yet a product still a prototype
General overview of DIOPRO

Default scenario

Application

write()

I/O library

file

DIOPRO

dlsym trick:
#define MAP_FUNCTION(func) 
   if (!(__real_##func)) 
      __real_##func = dlsym(RTLD_NEXT, #func); 
   }

Read buffer

Produces compressed trace file

Read trace file

Produces analysis

Read trace file

Produces performance numbers

© 2016 DataDirect Networks, Inc. * Other names and brands may be claimed as the property of others. Any statements or representations around future events are subject to change.
$ mpirun -x LD_PRELOAD=/usr/lib/dio-pro.so -n 8 ./mpi-write-multi -f output/file -l 16

0: io_time = 0.078403
1: io_time = 0.093839
2: io_time = 0.086452
3: io_time = 0.167057
4: io_time = 0.065364
5: io_time = 0.111795
6: io_time = 0.167707
7: io_time = 0.150377

longest_io_time = 0.167707 seconds
total_number_of_bytes = 536870912

$ zcat /tmp/dio-pro/iotrace-jobid-*.bin.gz | dio-pro-xml | dio-pro-stat -p | grep 'Process\|write'

Process 1: ID: 12382 - MPI rank: 0
write 67.108864 MB 1 0.078378 s 816.558 MiB/s 816.558 MiB/s 816.558 MiB/s 0 B/s

Process 2: ID: 12383 - MPI rank: 1

Process 8: ID: 12390 - MPI rank: 7
write 67.108864 MB 1 0.150335 s 425.717 MiB/s 425.717 MiB/s 425.717 MiB/s 0 B/s

(\text{bytes*8})/\max(\text{time})/1024^2 = \text{[MiB/s]} \\
(67108864*8)/\max(0.078378,0.093811,0.086434,0.167011,0.065349,0.111760,0.167680,0.150335)/1024^2 = 3053.435 \text{ MiB/s}
Figure 4: Temporal overview of the .bigSharedFile file's write offset. 10 MPI Ranks write to the .bigSharedFile file in 13 steps, as shown in the figure. Each step contains 29 arrays. If one could zoom into the arrays, 10 MPI Ranks blocks of around 4MB would be visible (See Figure 5). The time-wise length (bottom arrows) of each I/O step is indicated between the green and red vertical dashed lines. The time between each step is shown (top arrows) between the red and green dashed lines.
## What To Do With I/O Traces?

<table>
<thead>
<tr>
<th>Tag</th>
<th>Ranks</th>
<th>Read</th>
<th>Write</th>
<th>Meta</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>%</td>
<td>%</td>
<td>%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sec</td>
<td>sec</td>
<td>sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bytes</td>
<td>bytes</td>
<td>bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MB/s</td>
<td>MB/s</td>
<td>MB/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>0.266</td>
<td>94.11</td>
<td>50564.613</td>
<td>359,101,407.640</td>
</tr>
<tr>
<td>X2</td>
<td>0.266</td>
<td>1.17</td>
<td>576,543.24</td>
<td>95,567,735.24</td>
</tr>
<tr>
<td>X3</td>
<td>0.266</td>
<td>1.44</td>
<td>375,890.49</td>
<td>25,046,129.09</td>
</tr>
<tr>
<td>X4</td>
<td>0.266</td>
<td>1.21</td>
<td>514,316.14</td>
<td>22,747,898.78</td>
</tr>
<tr>
<td>X5</td>
<td>0.266</td>
<td>0.81</td>
<td>426,277.34</td>
<td>28,538,811.32</td>
</tr>
<tr>
<td>X6</td>
<td>0.266</td>
<td>0.35</td>
<td>1311.705</td>
<td>4,967,095,534</td>
</tr>
<tr>
<td>X7</td>
<td>0.266</td>
<td>0.17</td>
<td>857.788</td>
<td>1,111,308.00</td>
</tr>
<tr>
<td>X8</td>
<td>0.266</td>
<td>0.14</td>
<td>736.05</td>
<td>1,111,308.00</td>
</tr>
<tr>
<td>X9</td>
<td>0.266</td>
<td>0.00</td>
<td>461.064</td>
<td>701.404.00</td>
</tr>
<tr>
<td>X10</td>
<td>0.266</td>
<td>0.05</td>
<td>231.268</td>
<td>1,056,905.76</td>
</tr>
<tr>
<td>X11</td>
<td>0.3</td>
<td>0.53</td>
<td>4.306</td>
<td>245,251,112</td>
</tr>
<tr>
<td>X12</td>
<td>0.3</td>
<td>0.38</td>
<td>3.065</td>
<td>13,208,908</td>
</tr>
<tr>
<td>X13</td>
<td>0.3</td>
<td>0.02</td>
<td>0.004</td>
<td>1,273.564</td>
</tr>
<tr>
<td>X14</td>
<td>0.3</td>
<td>0.01</td>
<td>0.013</td>
<td>2,699,705</td>
</tr>
<tr>
<td>X15</td>
<td>0.3</td>
<td>0.00</td>
<td>0.003</td>
<td>33,242.0</td>
</tr>
<tr>
<td>X16</td>
<td>0.3</td>
<td>0.00</td>
<td>0.002</td>
<td>150,080.0</td>
</tr>
<tr>
<td>X17</td>
<td>0.3</td>
<td>0.00</td>
<td>0.000</td>
<td>7,530.0</td>
</tr>
<tr>
<td>X18</td>
<td>0.3</td>
<td>0.00</td>
<td>0.000</td>
<td>39,294.0</td>
</tr>
<tr>
<td>X19</td>
<td>0.266</td>
<td>52.84</td>
<td>468,08.07</td>
<td>16,944,108,416</td>
</tr>
<tr>
<td>X20</td>
<td>0.266</td>
<td>5.83</td>
<td>507,783.0</td>
<td>1,067,315.52</td>
</tr>
<tr>
<td>X21</td>
<td>0.266</td>
<td>5.65</td>
<td>492,968.1</td>
<td>1,067,315.52</td>
</tr>
<tr>
<td>X22</td>
<td>0.266</td>
<td>5.58</td>
<td>489,048.2</td>
<td>1,067,315.52</td>
</tr>
<tr>
<td>X23</td>
<td>0.266</td>
<td>5.36</td>
<td>467,073.2</td>
<td>1,067,315.52</td>
</tr>
<tr>
<td>X24</td>
<td>0.266</td>
<td>5.16</td>
<td>449,212.1</td>
<td>1,067,315.52</td>
</tr>
<tr>
<td>X25</td>
<td>0.266</td>
<td>5.14</td>
<td>447,999.9</td>
<td>1,067,315.52</td>
</tr>
<tr>
<td>X26</td>
<td>0.266</td>
<td>5.01</td>
<td>436,792.2</td>
<td>1,067,315.52</td>
</tr>
<tr>
<td>X27</td>
<td>0.266</td>
<td>4.83</td>
<td>421,032.2</td>
<td>1,067,315.52</td>
</tr>
<tr>
<td>X28</td>
<td>0.266</td>
<td>4.58</td>
<td>398,656.3</td>
<td>1,067,315.52</td>
</tr>
</tbody>
</table>

Table 10: Per file read/write/meta activity as observed for MPI. Times indicated are the sum of the time spend by all ranks in that particular operation. Rel indicates the percentual share of that file in its read/write/meta category, time-wise. The 'orange tags' are files also present in the POSIX table.
What To Do With I/O Traces?

► Hot file show a pathological contention pattern
What To Do With I/O Traces?  Solve!

- Replication of the I/O pattern in Düsseldorf /Paris lab.
- HW optimization (IME) improves performance by x 64
- SW optimization (MPI-IO rewriting/ tuning) improve performance by x 1.2 (20% )
What To Do With I/O Traces? Replayer

- Instead of rewriting application I/O kernel replay its trace!
- Application characterization on large trace (2000+ ranks)
  - Isolate critical files
- Estimate control (metadata + synchronization) vs data cost
  - Distribution: metadata = 18.89%, barrier = 1.46%, data = 79.64%
- Search for symmetry
- Maintain group weight in downsizing
- Performance offset shift to avoid discontinuity artifact

\[ W_i = \frac{\# \text{ranks in group } i}{\text{total } \# \text{ranks}} \]
Replayer for architectural investigation
IME vs Spectrum Scale

IOR single shared file – 64 ranks on 8 clients 4MB block size

250 GB

45 sec
How to generate an I/O signature

► Structured part
  • Exploiting the deterministic behavior
  • Noise needs to be removed beforehand
  • Grammar analysis → identifying structure in sequences
    ◦ Nested Loop Recognition (University Of Strasbourg)
    ◦ Sequitur (Google, University of Waikato)

► Random part
  • Quantized via statistical analysis
    ◦ Continuous-time Markov chain modeling
    ◦ Long-range dependency matching
    ◦ Use of robust statistical methods

► Standardized I/O signature formulation
DIO-pro  an on-going effort

- Complexity is becoming unmanageable
- New app. challenge *our* (mine) expertise
- Fill a gap in the performance investigation stack
  - Full application  →  We are here!
  - Mini-app
  - Synthetic kernel
  - Analytical mode
Join us for a day of workshops dedicated to I/O at ISC High Performance 2017 on June 22, 2017.

WOPSSS

The Workshop On Performance and Scalability of Storage Systems (WOPSSS) aims to present state-of-the-art research, innovative ideas, and experience that focus on the design and implementation of HPC storage systems in both academic and industrial worlds, with a special interest on their performance analysis.

The arrival of new storage technologies and scales unseen in previous practice lead to significant loss of performance predictability. This will leave storage system designers, application developers and the storage community at large in the difficult situation of not being able to precisely detect bottlenecks, evaluate the room for improvement, or estimate the matching of applications with a given storage architecture. WOPSSS intends to encourage discussion of these issues through submissions of researchers or practitioners from both academic and industrial worlds.

All accepted papers will be published in the Proceedings by Springer. Extended versions of the best papers will be published in the ACM SIGOPS (http://www.sigops.org/osr.html) journal. Papers need to be submitted via EasyChair (http://easychair.org/conferences/?conf=wopsss2017).

Submission Deadline: March 31
Workshop: June 22

I/O in the Data Center Workshop

Managing scientific data at large scale is challenging for scientists but also for the host data center. The storage and file systems deployed within a data center are expected to meet users’ requirements for data integrity and high performance across heterogeneous and concurrently running applications.

With new storage technologies and layers in the memory hierarchy, the picture
Thank You!
Keep in touch with us

sales@ddn.com

@ddn_limitless

company/datadirect-networks

9351 Deering Avenue
Chatsworth, CA 91311

1.800.837.2298
1.818.700.4000