MOTIVATION FOR CHARACTERIZING PARALLEL I/O

Times are changing in HPC storage!

- Most scientific domains are increasingly data intensive: climate, physics, biology and much more
- Upcoming computing platforms include complex, hierarchical storage systems

How can we maximize productivity in this environment?

The NERSC burst buffer roadmap and architecture, including solid state burst buffers that can be used in a variety of ways

Example visualizations from the Human Connectome Project, CERN/LHC, and the Parallel Ocean Program
KEY CHALLENGES

- **Instrumentation:**
  - What do we measure?
  - How much overhead is acceptable and when?

- **Analysis:**
  - How do we correlate data and extract actionable information?
  - Can we identify the root cause of performance problems?

- **Impact:**
  - Develop best practices and tune applications
  - Improve system software
  - Design and procure better systems
CHARACTERIZING APPLICATION I/O WITH DARSHAN
**WHAT IS DARSHAN?**

**Darshan** is a scalable HPC I/O characterization tool. It captures an accurate but concise picture of application I/O behavior with minimum overhead.

- No code changes, easy to use
  - Negligible performance impact: just “leave it on”
  - Enabled by default at ALCF, NERSC, and NCSA
  - Used on a case-by-case basis at many other sites

- Produces a **summary** of I/O activity for each job, including:
  - Counters for file access operations
  - Time stamps and cumulative timers for key operations
  - Histograms of access, stride, datatype, and extent sizes

- Use cases: observe and tune individual applications, or capture a broad view of the platform workload
DARSHAN DESIGN PRINCIPLES

- The Darshan run time library is inserted at link time (for static executables) or at run time (for dynamic executables)
- Transparent wrappers for I/O functions collect per-file statistics
- Statistics are stored in bounded memory at each rank
- At shutdown time:
  - Collective reduction to merge shared file records
  - Parallel compression
  - Collective write to a single log file
- No communication or storage operations until shutdown
- Command-line tools are used to post-process log files
Example: Darshan-job-summary.pl produces a 3-page PDF file summarizing various aspects of I/O performance.

This figure shows the I/O behavior of a 786,432 process turbulence simulation (production run) on the Mira system.

- Percentage of runtime in I/O
- Access size histogram
- Access type histograms
- File usage

### File Count Summary (estimated by I/O access offsets)

<table>
<thead>
<tr>
<th>Type</th>
<th>Number of Files</th>
<th>Avg. Size</th>
<th>Max Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total opened</td>
<td>17</td>
<td>199G</td>
<td>1.6T</td>
</tr>
<tr>
<td>Read-only files</td>
<td>1</td>
<td>2.0K</td>
<td></td>
</tr>
<tr>
<td>Write-only files</td>
<td>13</td>
<td>260G</td>
<td>1.6T</td>
</tr>
<tr>
<td>Read/write files</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Created files</td>
<td>13</td>
<td>260G</td>
<td>1.6T</td>
</tr>
</tbody>
</table>
With a sufficient archive of performance statistics, we can develop heuristics to detect anomalous behavior.

This example highlights large jobs that spent a disproportionate amount of time managing file metadata rather than performing raw data transfer.

Worst offender spent 99% of I/O time in open/close/stat/seek.

This identification process is not yet automated; alerts/triggers are needed for greater impact.

---

Example of heuristics applied to a population of production jobs on the Hopper system in 2013:

<table>
<thead>
<tr>
<th>JOBS IDENTIFIED USING METADATA RATIO METRIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thresholds</td>
</tr>
<tr>
<td>Total jobs analyzed</td>
</tr>
<tr>
<td>Jobs matching metric</td>
</tr>
<tr>
<td>Unique users matching metric</td>
</tr>
<tr>
<td>Largest single-job metadata ratio</td>
</tr>
</tbody>
</table>

Darshan was conceived to address the need for greater understanding of I/O behavior in diverse scientific applications. It enabled unprecedented insight into the behavior of the most data-intensive scientific applications at Argonne National Laboratory.

- Darshan was generalized and ported to multiple computational platforms (IBM BG/Q, Cray XE and XC, Linux clusters) and deployed at every major ASCR facility.
- Widespread deployment enabled both cross-platform studies and targeted optimizations to improve scientific productivity.

Impact Going Forward
- Darshan is supported by the ALCF, NERSC, OLCF, and NCSA facilities on their largest systems.
- Vendors are contributing features.
- Darshan is being leveraged in new projects.
WHAT’S NEW?
MODULARIZED INSTRUMENTATION

- Frequently asked question: Can I add instrumentation for X?
- Darshan has been re-architected as a modular framework to help facilitate this, starting in v3.0
- Instrumentation modules:
  - Instruments a source of I/O data (I/O API, file system, etc.)
  - Creates/registersUpdates data records characterizing application I/O workload
- Darshan core library:
  - Exposes interface for modules to coordinate
  - Compresses and writes module data to log

We are using the modular framework to integrate more data sources and simplify the connections between various components in the stack.
**DARSHAN MODULE EXAMPLE**

- We are using the modular framework to integrate more data sources and simplify the connections between various components in the stack.

- Enables new insights into the link between application behavior and system behavior.
- In this case we observe how application I/O maps to Lustre file servers.
- Combined with system logs, this revealed a bottleneck caused by failover.

---

**Per-server I/O load on Edison for a 6,144 process simulation**

![Graph showing OSS occupancy and I/O time with data points and annotations for OSS failures.](image)
WHAT’S NEW?
DARSHAN EXTENDED TRACING (DXT)

- DXT module provides optional fine-grained instrumentation without recompiling
  - Detailed tracing of POSIX & MPI-IO interfaces
  - Enabled at runtime using environment variable
  - Individual I/O accesses can be mapped to specific Lustre OSTs

- First available in Darshan 3.1.3
- Contributed by Cong Xu and Intel’s High Performance Data Division (HPDD)

---

WHAT’S NEW?
DARSHAN EXTENDED TRACING (DXT)

- Future work in leveraging fine-grain instrumentation capability:
  - When should it be enabled? Dynamic triggers?
  - Can we analyze parts of it at run time?
THE NEED FOR ONGOING CHARACTERIZATION

- We’ve used Darshan to improving application productivity with case studies, application tuning, and user education
- ... But challenges remain:
  - What other factors influence performance?
  - What if the problem is beyond a user’s control?
  - The user population evolves over time; how do we stay engaged?
BEYOND APPLICATION CHARACTERIZATION:

HOLISTIC I/O CHARACTERIZATION FOR “TOTAL KNOWLEDGE OF I/O” (TOKIO)
“I OBSERVED PERFORMANCE XYZ.”

Is that normal?

- Consider a climate vs. weather analogy:
  “It is snowing in Atlanta, Georgia.” Is that normal?

- You need context to know:
  - Does it ever snow there?
  - What time of year is it?
  - What was the temperature yesterday?
  - Do your neighbors see snow too?
  - Should you look at it first hand?

- It is similarly difficult to understand a single application performance measurement without broader context

- How do we differentiate typical I/O climate from extreme I/O weather events?
HPC I/O VARIABILITY EXAMPLE

Consider the “I/O fingerprint” for two example systems on the right

- Data gathered with standardized, periodic, regression sampling of performance for diverse workloads
- More than just a single scalar number for expected I/O performance
- Indicates strengths, weaknesses, and susceptibility to variance

This data also confirms that I/O performance has extraordinary day to day variability.

- This preliminary example show the median performance of several workloads over time on two major computing platforms.
- Performance is normalized to the maximum observed rate on each system, to focus on trends rather than absolute throughput.
- Whiskers indicate minimum and maximum sample values.
WHAT CAUSES VARIABILITY?

- The load on the storage system plays is a major factor. HPC I/O workloads are:
  - Inherently bursty
  - With intervals of simulation or analysis punctuated intervals of data movement

- HPC facilities also optimize utilization and reduce cost by sharing major I/O systems across computing systems

- What about network contention, faults, etc.? *There may be many factors!*
CHARACTERIZING THE I/O SYSTEM

What were the contributing factors to a job’s performance?

- We need a big picture view
- No lack of instrumentation methods for system components…
  - but with wildly divergent data formats, resolutions, and scope
CHARACTERIZING THE I/O SYSTEM

What were the contributing factors to a job’s performance?

- We need a big picture view
- No lack of instrumentation methods for system components…
  - but with wildly divergent data formats, resolutions, and scope
- This is the motivation for the TOKIO (TOtal Knowledge of I/O) project:
  - Integrate, correlate, and analyze I/O behavior from the system as a whole for holistic understanding
  - Leverage and extend existing technology where possible

https://www.nersc.gov/research-and-development/tokio/
Collaboration between LBL and ANL
PI: Nick Wright
TOKIO DEPLOYMENT

- No new database and (probably) no new instrumentation tool either
- Index and query existing data sources in their native format
  - Infrastructure to align and link data sets
  - Adapters/parsers to produce coherent views on demand
  - Not all systems will have the same data
- Tools sharing a common format for analysis
  - correlation, data mining, dashboards, etc.
- Example of analysis tool: UMAMI
UMAMI
TOKIO Unified Measurements And Metrics Interface

- UMAMI is a pluggable dashboard that displays the I/O performance of an application in context with system telemetry and historical records.

Historical samples (for a given application) are plotted over time.

Each metric is shown in a separate row.

Box plots relate current values to overall variance.

(figures courtesy of Glenn Lockwood, NERSC)
UMAMI
TOKIO Unified Measurements And Metrics Interface

- Broader contextual clues simplify interpretation of unusual performance measurements

System background load is typical

Performance for this job is higher than usual

Server CPU load is low after a long-term steady climb

Corresponds to data purge that freed up disk blocks
UMAMI
TOKIO Unified Measurements And Metrics Interface

- UMAMI still requires some level of expert interpretation
- Can we automate the analysis?
  - Automatic root cause determination with Spark
  - Mathematical methods for quantifying correlations and variance
MODEL-BASED I/O OPTIMIZATION

We can also leverage holistic I/O instrumentation to develop broader metrics and methodology to understand parallel I/O variability

– Combines SDAV expertise on large-scale I/O with SUPER expertise on performance modeling and multi-metric inference
– Uses server-side and application-level monitoring to analyze job times of prototypical I/O-heavy applications on a BlueGene system at Argonne
– Previous studies ignored effect of parallel system state and workloads

TOKIO OUTCOME

- Standardized data formats and analysis tools
- Prototypes running at the ALCF (Mira IBM platform) and NERSC (Cori and Edison Cray platforms)
- Software components will be released open source as they are cleaned up from initial prototype
- Data repositories will be published as well
  - This is a secondary but important impact for TOKIO: enabling better data sharing between facilities for comparative studies and best practice development
MORE INFORMATION

- Darshan: http://www.mcs.anl.gov/research/projects/darshan
- TOKIO: http://www.nersc.gov/research-and-development/tokio/

carns@mcs.anl.gov
THANK YOU!

THIS WORK WAS SUPPORTED BY THE U.S. DEPARTMENT OF ENERGY, OFFICE OF SCIENCE, ADVANCED SCIENTIFIC COMPUTING RESEARCH, UNDER CONTRACT DE-AC02-06CH11357.

THIS RESEARCH USED RESOURCES OF THE ARGONNE LEADERSHIP COMPUTING FACILITY, WHICH IS A DOE OFFICE OF SCIENCE USER FACILITY SUPPORTED UNDER CONTRACT DE-AC02-06CH11357.