FUTURE OF I/O

ROB LATHAM
PHIL CARNES
Argonne National Laboratory

5:00-5:30pm, August 11, 2016
St. Charles IL
THE CHANGING LANDSCAPE OF COMPUTER SCIENCE

- Software tools are evolving to support new applications and new data-intensive methodologies:
  - In situ analysis
  - Workflows
  - Advanced programming models

- ... and the systems themselves are undergoing key revolutions, particularly in storage technology

- Notable architectural features on Aurora (2019):
  - On-node persistent memory (NVRAM)
  - Burst buffers
  - Omnipath network with silicon photonics
UPDATING I/O LIBRARIES
Evolving Parallel I/O Libraries

- An assortment of libraries are widely adopted already to help translate data models to storage
  - We've talked about MPI-IO, Parallel NetCDF, and HDF5 today
  - The APIs and data models that they present will continue to be critical on future systems
  - So how do we adapt?
EVOLVING PARALLEL I/O LIBRARIES

Future enhancements

- New collective algorithms and optimizations that take into account the new realities of HPC architectures (e.g., low latency, local or semi-local storage)

- Intermediate storage locations offer a convenient staging area for alternative file formats:
  - Optimize for ingest when writing data, reorganize for analysis when transferring to longer-term storage

- Compression
  - Depends on the application use case, but we have a new opportunity for data transforms in the I/O path
NEW DATA SERVICES
BEYOND PARALLEL FILE SYSTEMS

How will storage software change in the future?

- How exactly are we going to use the various levels of storage available in upcoming systems?
- Observation: successful HPC applications are composed of software components that provide only the communication, concurrency, and synchronization needed for the task at hand.
- Why not storage services too? It doesn’t have to be a one-size-fits all global file system for all purposes.

The community is exploring **software defined storage** principles to provide more specialized, composable storage services for key applications.
Specialized data services are already here!

<table>
<thead>
<tr>
<th>Service</th>
<th>Function</th>
<th>MPI ranks</th>
<th>MPI</th>
<th>RAM</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADLB</td>
<td>Data store and pub/sub.</td>
<td>(MPI ranks)</td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>DataSpaces</td>
<td>Data store and pub/sub.</td>
<td>Indep. job</td>
<td>Dart</td>
<td>RAM (SSD)</td>
<td>Under devel.</td>
</tr>
<tr>
<td>DataWarp</td>
<td>Burst Buffer mgmt.</td>
<td>Admin./ sched.</td>
<td>DVS/ Inet</td>
<td>XFS, SSD</td>
<td>Ext. monitor</td>
</tr>
<tr>
<td>FTI</td>
<td>Checkpoint/restart mgmt.</td>
<td>MPI ranks</td>
<td></td>
<td>RAM, SSD</td>
<td>N/A</td>
</tr>
<tr>
<td>Kelpie</td>
<td>Dist. in-mem. key/val store</td>
<td>MPI ranks</td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>SPINDLE</td>
<td>Exec. and library mgmt.</td>
<td>Launch MON</td>
<td>TCP</td>
<td>RAMdisk</td>
<td>N/A</td>
</tr>
</tbody>
</table>
COMPOSING HPC STORAGE SERVICES

- Vision: Specialized HPC storage services composed from building blocks that provide required data abstractions, communication, synchronization, resilience, access control, etc.
  - Match service to science requirements and technology available
  - Scope of coherence, etc. constrained to application(s) using service
  - Extend, not replace, the existing storage ecosystem
  - Don’t pay for resilience or durability until you need it

- Approach: Lightweight, user-space services that can be quickly instantiated and torn down in response to application needs

- These “micro” services can also be developed and experimentally evaluated more quickly

- The above concept is being pursued by a collaborative effort between Argonne National Laboratory, Carnegie Mellon University, the HDF Group, and Los Alamos National Laboratory
SDS: STATUS AND USE CASES

- Identifying initial use cases
- Assembling building blocks
- Building prototypes
- Not necessarily files and directories, could use objects, key/value pairs, or other constructs

Example use cases:
- Shared, in-system databases
  - Equation of state
  - Opacity
  - Out of core computation
- On-demand metadata service allocation
- Workflow coordination
  - Coupling
  - In-situ
- Analysis
  - Data subsetting
  - Indexing
- Checkpoint/restart
BETTER INSTRUMENTATION
DARSHAN FUTURE: EXPANDING COVERAGE

- Darshan has to track advancements in I/O architectures as well
- Darshan 3.x has been redesigned to enable the addition of new instrumentation modules
- Turning Darshan into a more flexible instrumentation platform
- Modules can gather data from a variety of sources
  - I/O libraries (e.g., POSIX, MPI-IO, HDF5, PnetCDF)
  - FS interfaces (e.g., Lustre API)
  - System-specific data (e.g., BG/Q or Cray runtime environment)

- Instrumentation modules are only recognized by Darshan when they are actually activated
  - Darshan assigns memory to modules for storing I/O data
  - Instrumentation module provide callback functions so Darshan can interface with them at shutdown time
DARSHAN FUTURE: EXPANDING COVERAGE

- At shutdown, Darshan:
  - Retrieves I/O data from each module
  - Compresses data
  - Collectively writes data to Darshan log
DARSHAN FUTURE: EXPANDING COVERAGE

- Applications that do not shut down cleanly cause a gap in coverage
  - Darshan’s normal shutdown procedure hooks into MPI_Finalize()
  - If you run to the wall clock limit or the application crashes, then Darshan cannot store its data
- We are now developing low-overhead techniques to store ongoing data and retrieve it after application exit

Applications that don’t exit cleanly are more common than you might think: example from NCSA.


These observations are corroborated by internal metrics at NERSC
DARSHAN FUTURE: DATA MINING AND HOLISTIC I/O CHARACTERIZATION

- Mining Darshan data
  - Ongoing work led by M. Winslett (UIUC)
  - Mining Darshan logs for interactive visualization and generation of representative workloads

- Holistic I/O characterization framework
  - Ongoing work led by N. Wright (LBL/NERSC)
  - Deploy best-in-class characterization tools for applications, servers, devices, network, etc.
  - Combine and correlate data for deeper insight into the system as a whole

Thanks to Huong Luu (UIUC) for providing this figure.

Web dashboard for ad-hoc analysis: https://github.com/huongluu/DarshanVis
1. Component-level monitoring data fed into RabbitMQ
   - Slurm plugins (kernel counters, Darshan, IPM)
   - Native support (procmon, collectd)
THANK YOU!

THIS CONCLUDES THE "FUTURE OF I/O" SESSION