Using the Adaptable I/O System (ADIOS)

Joint Facilities User Forum on Data-Intensive Computing
June 18, 2014
Norbert Podhorszki

Subtle message of the forum agenda
What is ADIOS?

• ADaptable I/O System

• As Wes Bethel said in his talk on Monday morning:
  – ADIOS is an In-situ framework

• Don’t think about it as just a portable I/O library that indeed does scale with data size and number of writers
ADIOS aspiration

Cloud interface

Big Data Cluster

WAN interface

Analysis workflow

Analytics Site

Analysis workflow

Collaborator Site

Remote data movement by ADIOS

Sensor or Instrument

Computer

Workflow built from plugins

Local data movement by ADIOS
R&D100 award for what?

Adaptable I/O System for Big Data (ADIOS)

ADIOS is a portable, scalable, easy-to-use software framework conceived to solve “big data” problems. For scientists making use of high performance computers, ADIOS significantly reduces the input or output complexities typically encountered and reduces the time to solution, so researchers spend less time managing data. The software streamlines workflows and lays the foundation for exascale supercomputers to be able to run multiple tasks simultaneously.

The research was funded by DOE’s Oak Ridge Leadership Computing Facility, the Office of Advanced Scientific Computing Research, the Office of Fusion Energy Science, and the National Science Foundation.

The ORNL team consisted of (seated) Norbert Podhorszki, Gary Liu, Yuan Tian; (standing) Jong Youl Choi, Hasan Abbasi, Jeremy Logan, Scott Klasky; and (not pictured) Roselyne Tchoua. Also not pictured are Karsten Schwab and Matthew Wolf (Joint Faculty; Georgia Institute of Technology), Manish Parashar (Rutgers University), Naziza Samatova (Joint Faculty; North Carolina State University), and Jay Lofstead (Sandia National Laboratories).
Quantum Physics – QLG2Q

- QLG2Q is a quantum lattice code developed in a DoD project.
- George Vahala (William & Mary), Min Soe (Rogers State)
- Large data size + many processors, > 50 MB per core, >100K cores

QLG2Q MPI-IO performance on JaguarPF

Isosurface visualization of QLG2Q data in Visit

Thanks to Dave Pugmire
Quantum Physics – QLG2Q

- ADIOS version removed their I/O bottleneck completely
  - 45GB/s on half of JaguarPF (110k cores)
- Recent releases of ADIOS achieve 98 GB/sec on ERDC, Garnet

Garnet performance with $32^3=32k$ cores with $3200^3$ data space 6 double complex arrays, 2.8TB, it takes 31 seconds to write.
How do they do that? We told them to...

- Avoid latency (of small writes)
  - **Buffer** data for large bursts
- Avoid accessing a file system target from many processes at once
  - **Aggregate** to a small number of actual writers
    - proportionate to the number of file system targets, not MPI tasks
- Avoid lock contention
  - by **striping correctly**
  - or by writing to subfiles
- Avoid global communication during I/O
  - **ADIOS-BP file format**
<?xml version="1.0"?>
<adios-config host-language="Fortran">
  <adios-group name="spin1">
    <var name="xdim" gwrite="lg" type="integer"/>
    <!– ... Similar definitions for ydim, zdim, ipe, jpe, kpe, lx, ly, lz, start_indices* -->
    <global-bounds dimensions="xdim,ydim,zdim" offsets="offx,offy,offz">
      <var name="qab1" gwrite="phia1(is:ie,js:je,ks:ke)"
           type="double complex" dimensions="lx,ly,lz"/>
      <!– ... Similar definitions for qab2, qab3, qab4, qab5, qab6 -->
    </global-bounds>
  </adios-group>
</adios-config>
Source code to declare output action

call mpi_init (ierror)
call adios_init ("spin1.xml", mpi_comm_world, ierror)
...
call adios_open (adios_handle, "spin1", fname1, "w", group_comm, ierr)
#include "gwrite_spin1.fh"
call adios_close (adios_handle, ierr)
...
call adios_finalize (rnk, ierr)
call mpi_finalize (ierror)
XML file to set runtime parameters

<method group="spin1" method="MPI_AGGREGATE">
    num_aggregators=1024;num_ost=512
</method>

<buffer size-MB="256">
    allocate-time="now"/
</buffer>

</adios-config>
ADIOS Approach

• I/O calls are of **declarative** nature in ADIOS
  – which process writes what
    • add a local array into a global space (virtually)
  – adios_close() indicates that the user is done declaring all pieces that go into the particular dataset in that timestep

• I/O **strategy is separated** from the user code
  – aggregation, number of subfiles, target filesystem hacks, and final file format not expressed at the code level

• This allows users
  – to **choose the best method** available on a system
  – without modifying the source code

• This allows developers
  – to **create a new method** that’s immediately available to applications
  – to push data to other applications, remote systems or cloud storage instead of a local filesystem
XML file to set runtime parameters on Mira

```xml
<method group="spin1" method="BGQ">
  <buffer size-MB="60"
         allocate-time="now"/>
</method>
</adios-config>
```

- Topology-aware data movement was needed on BGQ
- With ADIOS 1.6’s BGQ method, QLG2Q achieves 120 GB/sec on 16 racks of Mira
call adios_read_init_method (ADIOS_READ_METHOD_BP, 
    group_comm,"",ierr);

call adios_read_open (f, filename, 0, group_comm, 
    ADIOS_LOCKMODE_CURRENT, 60.0, ierr)

do while (ierr != err_stream_terminated)
    call adios_get_scalar (f,"gdx",gdx, ierr)
    call adios_get_scalar (f,"gdy",gdy, ierr)

    ! ... calculate offsets and sizes of xy to read in...

    call adios_selection_boundingbox (sel, 2, offset, readsize)
    call adios_schedule_read (f, sel, "xy", 1, 1, xy, ierr)
    call adios_perform_reads (f, ierr)
    call adios_advance_step (f, 0, 60.0, ierr)
endo
call adios_read_close (f, ierr)
ADIOS Read API

Code which reads stream data

```fortran
call adios_read_init_method (ADIOS_READ_METHOD_DATASPACE,
                           group_comm, "", ierr);
call adios_read_open (f, filename, 0, group_comm,
                      ADIOS_LOCKMODE_CURRENT, 60.0, ierr)

do while (ierr != err_stream_terminated)
    call adios_get_scalar (f, "gdx", gdx, ierr)
    call adios_get_scalar (f, "gdy", gdy, ierr)

! … calculate offsets and sizes of xy to read in…

    call adios_selection_boundingbox (sel, 2, offset, readsize)
    call adios_schedule_read (f, sel, "xy", 1, 1, xy, ierr)
    call adios_perform_reads (f, ierr)
    call adios_advance_step (f, 0, 60.0, ierr)
enddo

call adios_read_close (f, ierr)
```
Introduction to Staging

- Initial development as a research effort to minimize I/O overhead
- Draws from past work on threaded I/O
- Exploits network hardware for fast data transfer to remote memory
- ADIOS contains 3 staging methods: DataSpaces, DIMES, FlexPath

1. Define Staging

2. Using Staging for writing. Think of burst buffers++

3. Allow workflow composition

Diagram showing the flow of data through computational and I/O nodes, with stages for defining, using, and allowing workflow composition.
Workflow composition with ADIOS+staging

ADIOS + DataSpaces/DIMES/FLEXPATH
+ asynchronous communication
+ easy, commonly-used APIs
+ fast and scalable data movement
+ not affected by parallel IO performance
- data aggregation/transformation at the coupler

Interactive visualization pipeline of fusion simulation, analysis code and parallel viz. tool
Hybrid Staging

- Use compute and deep-memory hierarchies to optimize overall workflow for power vs. performance tradeoffs
- Abstract complex/deep memory hierarchy access
- Placement of analysis and visualization tasks in a complex system
- Impact of network data movement compared to memory movement

Asynchronous Data Transfer

Parallel Data Staging coupling/analysis/viz
Harvesting Idle Time for In-Situ Computation

- Fine-Grain idle resource monitor and resource scheduler to concurrently schedule analytics with simulations on the same node
- GoldRush extends OpenMP schedulers and executes in-situ tasks during periods of serial processing in OpenMP application
- For many-core exascale nodes the same technique can identify low utilization cores
- GoldRush dynamically assesses resource contention in memory hierarchy and throttles the analytics execution rate to mitigate interference to simulation

- Evaluating utility of using additional core vs performing analytics inline using the parallel volume rendering
  - Additional core method executes 1.1% extra instructions but performs 5.1% less memory operations and finishes first
  - Inline operation imposes 48% more L1, and 69% more L2 cache misses

- Ongoing research to heterogeneous environment

GoldRush reduces analytics overhead by interference-aware asynchronous execution (*Runtime overheads of GoldRush (gold) and shared memory I/O (red) are negligible*)
Schema: unstructured example

```
<mesh name="trimesh" type="unstructured" time-varying="no">
  <nspace value="2" />
  <points-single-var value="points" />
  <uniform-cells count="num_cells" data="cells" type="triangle" />
</mesh>

<global-bounds dimensions="num_cells" offsets="oc">
  <var name="C" type="double" dimensions="lc" mesh="trimesh" center="cell" />
</global-bounds>
```
$ bpls -lm tri2d.bp

Mesh info:

$trimesh
  type: unstructured
  npoints: 144
  points: single-var:
    "points"
  ncsets: 1
  cell set 0:
    cell type: 3
    ncells: 240
    cells var: "cells"
  nspaces: 2
  time varying: no

integer nproc scalar = 12
integer npoints scalar = 144
integer num_cells scalar = 240
integer nx_global scalar = 16
integer ny_global scalar = 9
integer offs_x scalar = 0
integer offs_y scalar = 0
integer nx_local scalar = 4
integer ny_local scalar = 3
integer lp scalar = 12
integer op scalar = 0
integer lc scalar = 24
integer oc scalar = 0
double N \{144\} = 0 / 11
double C \{240\} = 0 / 11
double points \{144, 2\} = 0 / 25.6667
integer cells \{240, 3\} = 0 / 143
SPECFEM3D_GLOBE simulation and VisIt

- ADIOS file: three layers of the Earth which are being simulated
  - Crust-Mantle, Outer Core, and Inner Core
- Just another (actually 3) unstructured meshes
  - Hexa cells
- Mesh: 17GB file
  - this is current resolution (86M cells)
  - 9s resolution will produce ~450GB
What are Data Transformations?

- **Data transformations** are a class of technologies that change the format/encoding of data to optimize it somehow
  - Improve write performance
  - Reduce storage space
  - Accelerate read performance for analysis

<table>
<thead>
<tr>
<th>Data Transformation</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compression</td>
<td>Reduce I/O time and storage footprint</td>
</tr>
<tr>
<td>Filtering/sampling</td>
<td>Downsampling data to reduce I/O and storage</td>
</tr>
<tr>
<td>Indexing</td>
<td>Speed up query-driven analytics/visualization</td>
</tr>
<tr>
<td>Level-of-detail encoding</td>
<td>Fast approximate reads, high-precision drilldown</td>
</tr>
<tr>
<td>Layout optimization</td>
<td>Speed up various read access patterns</td>
</tr>
</tbody>
</table>
The ADIOS Transforms Framework

- Intrinsic support in ADIOS for **data transforms as a service**
- ADIOS as a **data transforms deployment platform** due to:
  - Wide acceptance and existing integration with scientific codes
  - Positioning in the I/O pipeline

![Diagram of ADIOS Transforms Framework]
Key Benefits of the Transforms Framework

1. Ad hoc integration with scientific codes is **avoided**
   – Well-defined plugin API for transform developers

2. ADIOS I/O pipeline **compatibility** is generally maintained

3. Transforms are **easily configured** via the ADIOS XML

4. **Read-optimizing transforms** can benefit applications
   – E.g., lower precision under level-of-detail can reduce read times
Applying Transforms to Data Variables

• Transforms are applied to individual variables with the ADIOS XML

• Parameters (e.g., zlib compression level) may be specified

• Example: applying “zlib” at compression level “5”:

```xml
<var name="pressure" type="double"
     dimensions="NX,NY,NZ" transform="zlib:5"/>
```
ADIOS Transforms Framework Overview

ADIOS XML config: `<var name="temperature" ... transform="zlib">27`
Data Transforms in ADIOS 1.6

• **zlib, bzip2, szip** lossless compression

• **ISOBAR** [1] adaptive lossless compression*
  – Selectively compresses parts of data based on entropy
  – Can improve both compression ratio and throughput

• **APLOD** [2] precision level-of-detail encoding*
  – Allows precision – access time tradeoff, including lossless access
  – Guaranteed bounded per-point error for each level


* request ISOBAR and APLOD libraries from Nagiza Samatova at North Carolina State University
ISOBAR Lossless Compression

- ISOBAR (In-Situ Orthogonal Byte Aggregate Reduction) Compression is a preconditioner-based, high-throughput lossless compression technique for hard-to-compress scientific datasets.

**ISOBAR compared to best standard lossless alternative (zlib or bzip2)**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ΔCR</th>
<th>TPc</th>
<th>SPc</th>
<th>TPd</th>
<th>SPd</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3D</td>
<td>32.6</td>
<td>105</td>
<td>31</td>
<td>425</td>
<td>63</td>
</tr>
<tr>
<td>GTS</td>
<td>10.2</td>
<td>112</td>
<td>8</td>
<td>552</td>
<td>5</td>
</tr>
<tr>
<td>XGC1</td>
<td>14.1</td>
<td>77</td>
<td>21</td>
<td>389</td>
<td>52</td>
</tr>
<tr>
<td>FLASH</td>
<td>17.2</td>
<td>456</td>
<td>36</td>
<td>1617</td>
<td>14</td>
</tr>
</tbody>
</table>

Measurement on Lens @ ORNL

**ΔCR**: compression ratio improvement (%)
**TPc**: compression throughput (MB/sec); **TPd**: decompression throughput
**SPc**: compression speedup; **SPd**: decompression speedup
Transform Layer

- We started with “none”

```
$ du *.bp
78624  writer00.bp
78624  writer01.bp
```

**ZLIB**

```
$ mpirun -np 12 ./writer
  ts= 0
  ts= 1
$ du *.bp
73820  writer00.bp
74112  writer01.bp
```

**BZIP2**

```
$ mpirun -np 12 ./writer
  ts= 0
  ts= 1
```

**ISOBAR**

```
$ mpirun -np 12 ./writer
  ts= 0
  ts= 1
$ du *.bp
60060  writer00.bp
60072  writer01.bp
```
Coming soon: Query

• Reading API is for reading data where you know what you want to read
• Query: find the interesting pieces and read them only
• So far Indexing/Query has been separate from I/O libraries
  – e.g. FastQuery implemented for HDF5, NetCDF, ADIOS
• Idea:
  – index can be generated on the fly, going along with the data in a pipeline
  – generic query API that (again) supports multiple indexing solutions
ADIOS is a complex project

• ADIOS started as a project to solve I/O + analysis + visualization for fusion, but evolved

• Involves multiple institutions, multiple projects, many application areas

• ADIOS is a very complex project, but
  – We are moving closer to service oriented architecture to place containers around services

• ADIOS is a framework
  – CS researchers have a platform to place new I/O methods and try them for real codes
  – Application scientist can use “known” I/O methods as a backup when more advanced methods fail on new machines

• ADIOS is our research platform
  – Example: SC 2013: 4 papers, 5 posters
Questions