ADaptable IO System (ADIOS) for Scientific Codes

Jay Lofstead, Scott Klasky, Hasan Abbasi, Karsten Schwan
lofstead@cc.gatech.edu, klasky@ornl.gov, {habbasi, schwan}@cc.gatech.edu
Center for Experimental Research in Computer Systems
College of Computing, Georgia Tech & Oak Ridge National Laboratory

Scientific Codes Examined

- GTC & GTC_S (fusion)
  - Completed (7 types, ~50 vars total)
  - HDF-5, Fortran IO, MPI-IO all replaced
- Chimera (astrophysics/supernova)
  - Implemented and under testing/debugging (~475 vars)
  - Uses text and Fortran IO; will create both text and binary
- XGC0 & XGC1 (fusion)
  - In planning stage
  - New requirements for code coupling support
- S3D (combustion)
  - In planning stage
  - Lots of vars

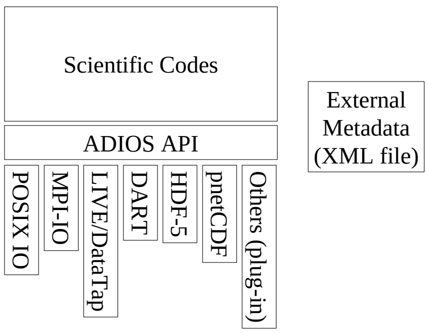
Project Goals

- Reduce ***IO overhead*** IO down to ***5%*** of total runtime or less.
- ***Simple API*** for Fortran and C nearly as simple as Fortran standard IO
- ***External metadata*** file describing data and transport selection that is parsed at runtime for greatest flexibility in transport selection and configuration. Uses ***XML*** for compatibility.
- Enable ***more frequent writing*** due to reduced IO costs.
- Both ***synchronous*** and ***asynchronous*** transports supported without code changes.
- ***Free*** hooks into ***visualization*** and ***workflow*** systems through the data flows.

Supported Transports

- POSIX IO
  - Both reading and writing
- MPI-IO
  - Using optimized approach from Steven Hodson (ORNL)
- LIVE/DataTap (Georgia Tech)
  - Asynchronous support
- DART (Rutgers)
  - Asynchronous support
- Parallel netCDF
  - In development
- HDF-5
  - In development

ADIOS API Architecture



Flexible I/O
– Scientific codes should choose the IO method that makes sense for their needs and change to what provides the level of functionality required later, without having to rewrite the IO routines.
– External metadata file documents I/O for more transparency and easier changes.
– Synchronous APIs support simple I/O while asynchronous can later be tested to determine how well it will work with the communication patterns.
– Optimized routines for free

Downstream Processing
– Perform expensive communication operations on cheaper resources
– Provide data safety while reducing files through offline consolidation—only if the data is verified.
– Integrate with Visit or Kepler by tapping into the datastream rather than rewriting the code
– Inline lossless or data aware lossy compression on the way to storage
– Automatic migration of completed data to offline storage to avoid running out of space.

Intelligent Data Movement
– Schedule movement off compute nodes according to compute/communication cycles.
– Manage storage actively to avoid contention issues to ensure best end-to-end times for all running codes.

API Example, Part 1

```
call adios_init ('config.xml')
…
! do main loop
call adios_begin_calculation ()
! do non-communication work
call adios_end_calcuation ()
…
! perform restart write
…
! do communication work
call adios_end_iteration ()
! end loop
…
call adios_finalize ()
```

API Example, Part 2

Restart Writing

```
call adios_get_type (io_type, 'restart')
call adios_group_by (io_type, 'mzeta',
      comm, previous, current, next)
…
call adios_open (group_handle,
   io_type, 'restart.01')
call adios_write (group_handle,
   'zion', zion)
…
ADIOS_WRITE (group_handle,
   mzeta)
…
call adios_close (group_handle)
```

Metadata XML File

```
<ioconfig><datatype name="restart">
<scalar name="mzeta" type="integer"
   write="no"/>
<dataset name="zion" type="double"
   dimensions="nparam,mi"/>
</datatype>
<method priority="1"
   method="PBIO" iterations="100"
   base-path="/tmp/work/ge1"
   type="restart">params</method>
<buffer size-MB="100" allocate-
   time="now"/></ioconfig>
```

Results

GTC
Initial performance results on the Cray XT3/XT4 at ORNL and Inifiniband cluster showed (asynchronous I/O):
25 GB/s on the Cray
2.5 GB/s on the Infiniband cluster

GTC_S
Initial performance results on the Cray XT3/XT4 at ORNL (1024 cores, ~1 GB filesize, every 10 iteration writes, synchronous non-collective MPI-IO, mzeta files): 12 GB/s

GTC (with DART)
Initial performance results on the Cray XT3/XT4 at ORNL (2048 cores): 0.4 GB/s streamed through to the Infiniband cluster

Chimera
Still testing

Summary

Initially, the Adaptable IO System simplifies the process of crafting I/O routines for scientific codes through the separation of the metadata and an API that is nearly as simple as standard Fortran I/O statements. The transport mechanisms underneath were crafted by experts for optimal performance giving the scientist the best performance possible for whatever I/O mechanism chosen. The separated metadata affords changing which I/O mechanism is employed by only changing the XML configuration file. This affords real apple-to-apple comparisons of various I/O techniques. Future benefits include tapping into the data stream for visualization (such as Visit), data aware data compression, and integration into workflow (such as Kepler)

References

1. Lofstead, Abbasi, Docan, Jin, Parashar, Schwan, Klasky, “Asynchronous I/O for Scientific Codes”, Petascale Data Storage Workshop at Supercomputing ’07, Reno, November 2007 (submitted).
2. Docan, Parashar, Lofstead, Schwan, Klasky “DART: An Infrastructure for High Speed Asynchronous Data Transfers”, IPDPS 08, Miami, April 2008 (submitted).
3. Hasan Abbasi, Matthew Wolf, Karsten Schwan. “Live data workspace: A flexible, dynamic and extensible platform for petascale applications.” In *Cluster Computing*, Austin, TX, September 2007. IEEE International.

http://www.cc.gatech.edu/~lofstead/adios