Adaptable IO System (ADIOS)

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

Cray User Group 2008

May 8, 2008

Chen Jin, Scott Klasky, Stephen Hodson, James B. White III, Weikuan Yu (Oak Ridge National Laboratory)

Jay Lofstead, Hasan Abbasi, Karsten Schwan, Matthew Wolf (Georgia Tech)

Wei-keng Liao, Alok Choudhary, North Western University

Manish Parashar, Ciprian Docan, Rutgers University.

Ron Oldfield, Sandia Labs

Outline

- ADIOS overview.
  - Design goals.
  - ADIOS files(bp).
- ADIOS APIs.
- ADIOS XML File Description.
- ADIOS Transport Methods.
  - Posix
  - MPI-IO
  - MPI-CIO
  - NULL
  - MPI-AIO
  - DataTap
  - DART
  - PHDF5
- Initial ADIOS Performance
- Future work
- Conclusions

ADIOS Overview – Design Goals

- Combine.
  - **Fast** I/O routines.
  - **Easy** to use.
  - **Scalable** architecture (100s cores) millions of processors.
  - **QoS.**
  - Metadata rich output.
  - Visualization applied during simulations.
  - Analysis, compression techniques applied during simulations.
  - Provenance tracking.
  - Methods to swap controlling apps (steering) vs. fast I/O.
- Use the largest, data producing codes at ORNL as test cases.
  - S3D, GTC, GTS, Chimera, XGC
- Support the 90% of the codes.
  - We haven't found a code which doesn't work, but we might.

Why Yet Another API?

- No special purpose API suitable for all purposes
  - complexities of programming interface
  - differences in file content support
  - performance differences depending on configuration of run
- No support for non-IO tasks as part of IO activity
  - To add Viz support, must add code
  - To integrate with steering or other feedback mechanism, must add code

A programmers perspective. (I/O options) until ADIOS came along! (PICK 1 please)

- Posix: F90 writes. “p” processors/files.
- MPI-IO: “n” writers, “p” procs, “f” files.
- Netcdf.
- HDF5
- Pnetcdf
- Phdf5

- **Asynchronous I/O**
- **Data streaming.**
- **VISIT APIs for steering.**
- **GT APIs for A/IO steering.**
- **Rutgers APIs for adios/steering.**

•We tried them all, but mainstream GTC used Posix F90 with p processors/files!

•Why?

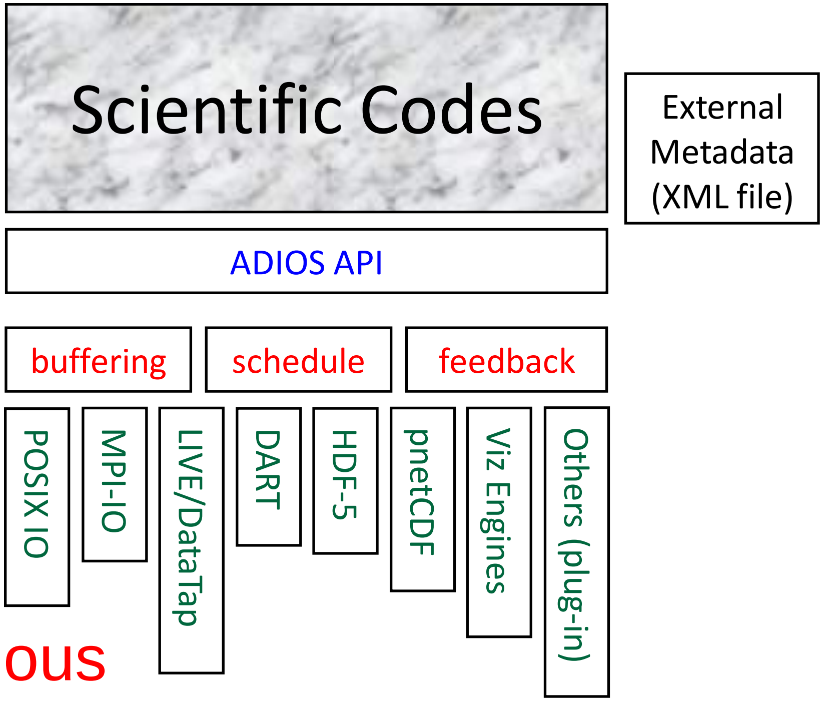
ADaptable IO System (ADIOS)

- Combines
  - High performance I/O.
  - In-Situ Visualization.
  - Real-time analytics.
- Collaborating with many institutions

	GTC	GTC_s	Flash	XGC1	Chimera	S3D	M3D	XGC0
MPI-IO/ORNL Jaguar	25 GBs	22GBs		15 GBs	20 GBs			
Async MPI-IO Jaguar								
DART Jaguar	1.2TB <1							
Datatap/jaguar								
Maviz/jaguar								
Visit/jaguar								
Paraview/jaguar								
Phdf5/jaguar								
Pnetcdf/jaguar								
BGP/IB/GPFS..								

% overhead

ADIOS Overview



Scientific Codes

External Metadata (XML file)

ADIOS API

buffering

schedule

feedback

POSIX IO

MPI-IO

LIVE/DataTap

DART

HDF-5

pnetCDF

Viz Engines

Others (plug-in)

- Overview
  - Allows plug-ins for different I/O implementations.
  - Abstracts the API from the method used for I/O.
- Simple API, almost as easy as F90 write statement.
- Both synchronous and asynchronous transports supported without code changes.
- Componentization.
  - Don’t worry about IO implementation.
  - Components for IO transport methods, buffering, scheduling, and eventually feedback mechanisms.
- Change IO method by changing XML file only!

ADIOS Overview

- Middleware between Applications and Transport Methods
- Abstract the data information (type, dimension, description etc) into XML file
- Clean and easy interface to app developers
- Webpage
  - http://www.cc.gatech.edu/~lofstead/adios
  - http://hecura.cs.unm.edu/doku.php?id=asynchronous_i_o_api

Benefits of ADIOS

- Simple API
  - As close to standard Fortran POSIX IO calls as possible
- External metadata in XML file
- Change IO transport/processing by changing XML file
- Best practices/optimized IO routines for all supported transports “for free”
- Supports both synchronous (MPI, MPI collective, netCDF, pnetCDF, HDF-5) and Asynchronous (GT: EVPath, Rutgers: DART) Transports
  - New transports for things like Visit and Kepler in the planning/development stages

ADIOS file format

- .bp format. (binary packed).
- Blocks of data are dumped with tags before each basic element.
- Will support a header in the released version of ADIOS.
  - Header will eventually be an index table.
  - No re-arrangement of the data when it touches disk.
- Utilities to dump .bp files to standard output. (like h5dump, ncdump).
- Converters from .bp to
  - Hdf5
  - Netcdf
  - Ascii (column data).

ADIOS Setup Code Example

XGC - Main.F90

```
# MPI Initialization
call my_mpi_init

# ADIOS Initialization
call adios_init ('config.xml', MPI_COMM_WORLD,
                 MPI_COMM_SELF,MPI_INFO_NULL)

# ADIOS resource release
call adios_finalize (sml_mype)

#MPI resource release
 call my_mpi_finalize
```

Example for asynchronous I/O scheduling

- Setup/iteration procedure

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

•For asynchronous operations ADIOS let’s programmers mark where no communication will occur in the code.

•We do this for ‘computational kernels’ in the code.

•XML file contains information for how quickly we must write out the data (how many iterations).

ADIOS APIs Example

GTC - restart.F90

```
call adios_get_group (grp_id, ’restart’)
call adios_open (buf_id, grp_id,
   ’restart.bp')
```

```
ADIOS_WRITE (buf_id,mpi_comm_world)
ADIOS_WRITE (buf_id,nparam)
ADIOS_WRITE (buf_id,mimax)
ADIOS_WRITE (buf_id,zion)
…
```

```
call adios_close (buf_id)
```

ADIOS XML Example

```
<adios-config host-language=“Fortran”>
<adios-group name=“restart”
         coordination-
   communicator="mpi_comm_world”>
    <var name=“mpi_comm_world” type=“integer*8”/>
    <var name=“nparam” type=“integer” write=“no”/>
    <var name=“mimax” type=“integer” write=“no”/>
    <var name=“zion” type=“double” dimensions=“nparam,mimax”/>
    <attribute name=“description” path=“/zion” value=“ion particle”/>
</adios-group>

<method priority=“1” method=“MPI”
   group=“restart”/>

<buffer size-MB=“100” allocate-time=“now”/>
</adios-config>
```

ADIOS Features

- Dataset/Array support
  - Local/global dimension:
    - Specify global space, local dimension(per mpi-process), and offsets (for this dataset).
    - We can specify ghost-zones too.
  - Support for specifying visualization meshes
    - VTK-like format used in XML code.
      - Structured/Unstructured data.
      - 1 mesh per ADIOS group.
    - No support for AMR for ADIOS 1.0
- Language Support
  - Fortran is the default.
  - C/C++ supported and tested.

ADIOS methods

- Posix.
  - ADIOS buffers data (user-definable) and writes out large blocks.
  - Posix writes out 1 file per MPI-process.

ADIOS MPI-IO method

- Simple – chained open requests dispatched sequentially, but with unknown time offset
    1. Process receives its starting file offset from previous rank
    2. Process calculates next file offset and sends to next rank
    3. Process opens file
- Robust – chained opens processed sequentially
    1. Process receives files offset from previous rank
    2. Process opens file
    3. Process calculates next file offset and sends to next rank
- Robust Timed – chained opens processed sequentially, attempts to maintain constant minimum offset between opens
    1. Process starts elapsed time
    2. Process receives files offset from previous rank
    3. Process opens file
    4. Process waits a specified interval minus elapsed time
    5. Process calculates next file offset and sends to next rank
- Each method will also offset the actual I/O data requests to a different degree.
- Similar methods could be used to control the data flow to OSTs

MPI-CIO (Collective MPI-method)

- Collective I/O enables process collaboration to rearrange I/O requests for better performance .
- The collective I/O method in ADIOS first defines MPI fileviews for all processes based on the data partitioning information provided in the XML configuration file.
- ADIOS also generated MPI-IO hints, such as data sieving and I/O aggregators, based on the access pattern and underlying file system configuration.
- The hints are supplied to the MPI-IO library for further performance enhancement.
- The syntax to describe the data partitioning pattern in the XML file uses <global-bounds dimensions offsets> tag which defines the global array size and the offsets of local subarrays in the global space.

MPI-AIO (Asynchronous MPI-IO method).

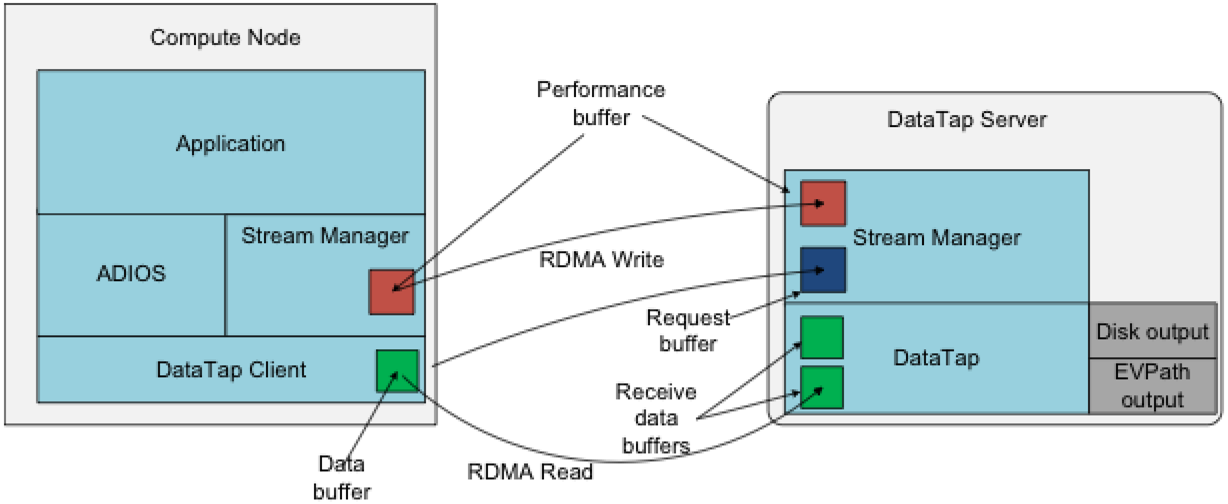
- Currently needs OpenMPI.
- Modify existing adios_mpi.c,
  - adios_mpi_do_write
  - adios_mpi_do_read
- Use asynchronous I/O if:
  - Buffer space available in adios (<buffer-MB=XXXX>) for current I/O request
  - Otherwise use synchronous call
- If asynchronous I/O not available in MPI-IO implementation, then request handled by MPI-IO synchronously
  - Unneeded async buffer allocation in adios consumed only for duration of synchronous I/O operation – a wash…
- **Currently only 1 outstanding async request allowed.**
  - Fine for large >>1MB I/O, but small I/O performance will benefit from queuing multiple requests.
- **Issue – deferred close**

ADIOS Method - NULL

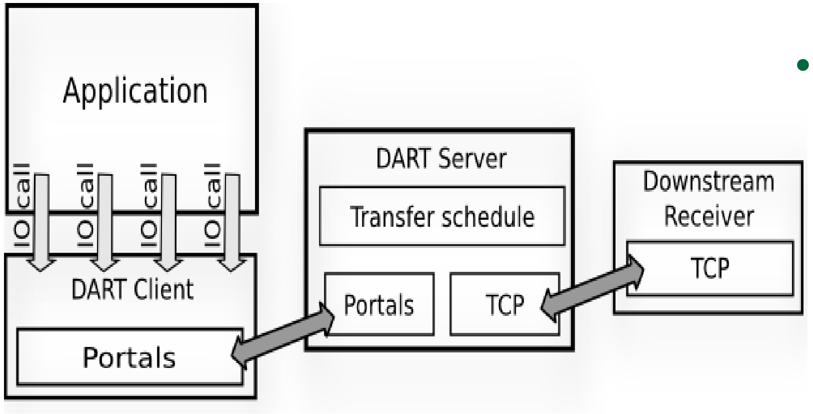
- Switching 1 line in the XML file enables the code NOT to write data to disk.
- Used for testing the speed of the I/O routine.

MPI method – asynchronous method with DataTap

- DataTap is an asynchronous data transport to ensure very high levels of scalability through sever-directed I/O, built by Georgia Tech.
- Data moves using RDMA methods from the compute nodes to the DataTap servers on login nodes.

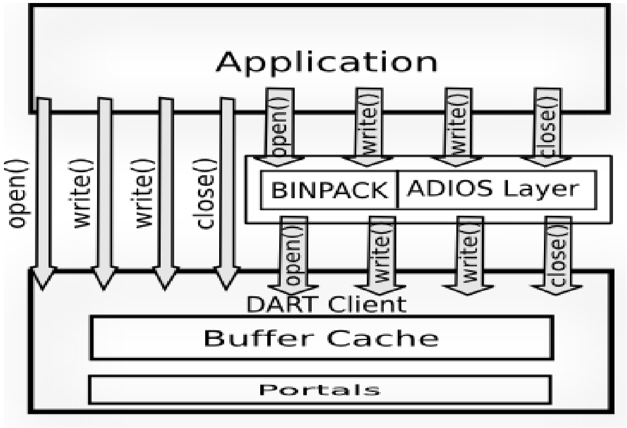


ADIOS DART Method.



- DART Asynchronous IO Substrate
  - Objectives
    - Minimize IO time & overheads
    - Maximize data throughput
    - Minimize latency of data transfers
  - Approach
    - Asynchronously schedule IO operations to overlap with computations
    - Dynamically manage buffering and mapping of IO to service nodes
    - Build on RDMA and Portals

- Evaluation using CPES Simulations
  - XGC-1, 1k nodes – stream to login node
    - transferred 380GB, in 3000sec with 5 sec spent in I/O - ~0.2% overhead
  - XGC-1, 1k nodes – save to local store
    - transferred 380GB, in 2900sec with 4.86 sec spent in I/O - ~0.16% overhead
  - GTC simulation on 1k and 2k nodes
    - transferred 1TB, in 3000sec with 12sec spent on I/O - ~0.6% overhead

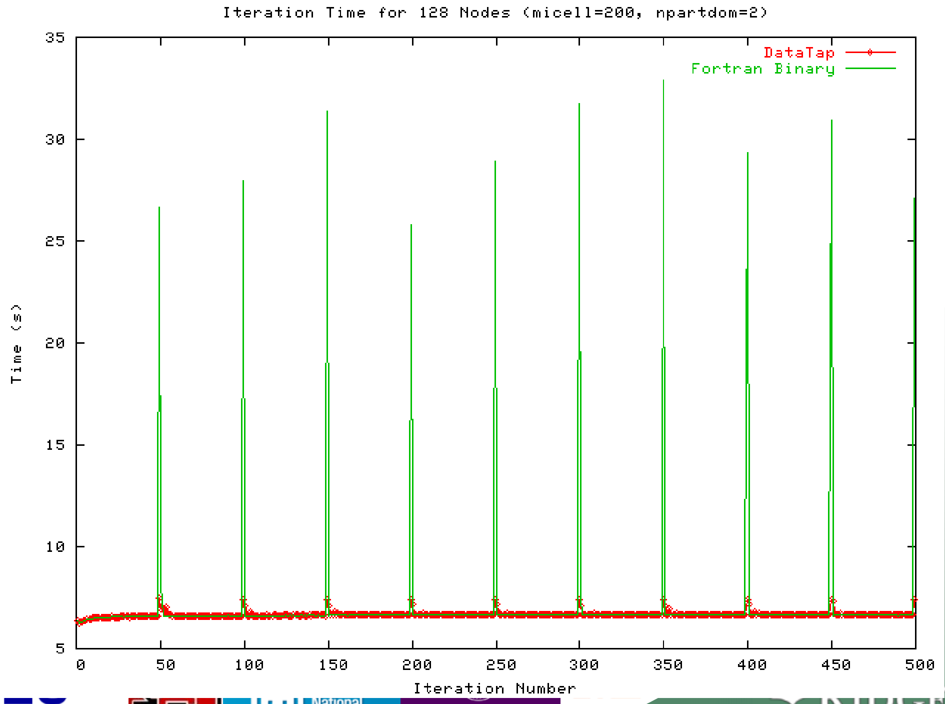


ADIOS methods not yet incorporated.

- Parallel HDF5.
- Parallel Netcdf.
- VISIT.
- Data multiplexing methods

Initial ADIOS performance.

- MPI-IO method.
  - GTC and GTS codes have achieved over 20 GB/sec on Cray XT at ORNL.
  - Chimera code speed up by 6.5% (overall time).
- DART: <2% overhead for writing 2 TB/hour with GTC code.
- DataTap vs. Posix
  - 5 secs for GTC computation.
  - ~25 seconds for Posix IO
  - ~4 seconds with DataTap



Future work: ADIOS 2.0

- Dynamic XML Generator within one simulation and reconfigure the simulation on the fly using the dynamic XML file.
- Index files built for fast reading.
- Metadata Catalogue.
  - One file will be produced per simulation which hyperlinks to all of the other files produced.
- Additional Readers developed.
- Additional Transport Methods.
  - Pnetcdf, ViSIT. (YOU TELL US).
- Hooks into workflow automation.
  - Metadata/EOF signals sent to Kepler workflow.

Conclusions & Future work

- ADIOS 1.0 will be released in 2008.
- Integrated for all IO in XGC1, GTC, GTS, Chimera, S3D, Flash.
- Fast asynchronous methods.
- ADIOS is supposed to be
  - Easy to use.
  - FAST.
  - Highly annotated.
- ADIOS 1.0 has hdf5, netcdf, ascii converters.

- ADIOS 2.0 will include
  - Parallel hdf5 methods.
  - Parallel netcdf methods.
  - Asynchronous schedulers optimized.
  - Data Multiplexing.
  - Faster methods to index files and read.
  - Harden routines on
    - Crays, BlueGene, Infiniband.
  - Bug fixing ☺
- More feedback from the File System.