# Using ADIOS - a High-Level Library to Improve I/O Performance

Qing Gary Liu

NICS/RDAV

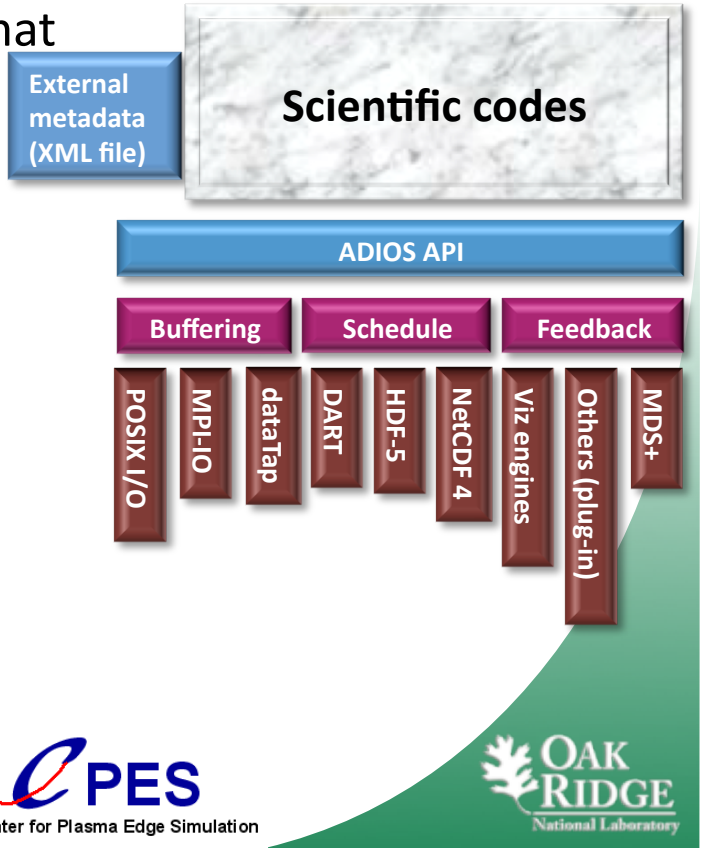
liuq@ornl.gov

*OLCF/NICS Spring Training*

*Oak Ridge, TN, March, 2011*

# ADIOS: ADaptable I/O System

- Overview
  - ***Componentization***: allows plug-ins for various I/O implementations
  - ***Abstraction***: abstracts the API from the method used for I/O
  - ***Innovative BP file format***: non-contiguous format
- Simple API, almost as easy as F90 write statement
- Both synchronous and asynchronous transports supported with no code changes
- Change your I/O by changing XML file only.
- ADIOS buffers data
- ADIOS allows multiple transport methods

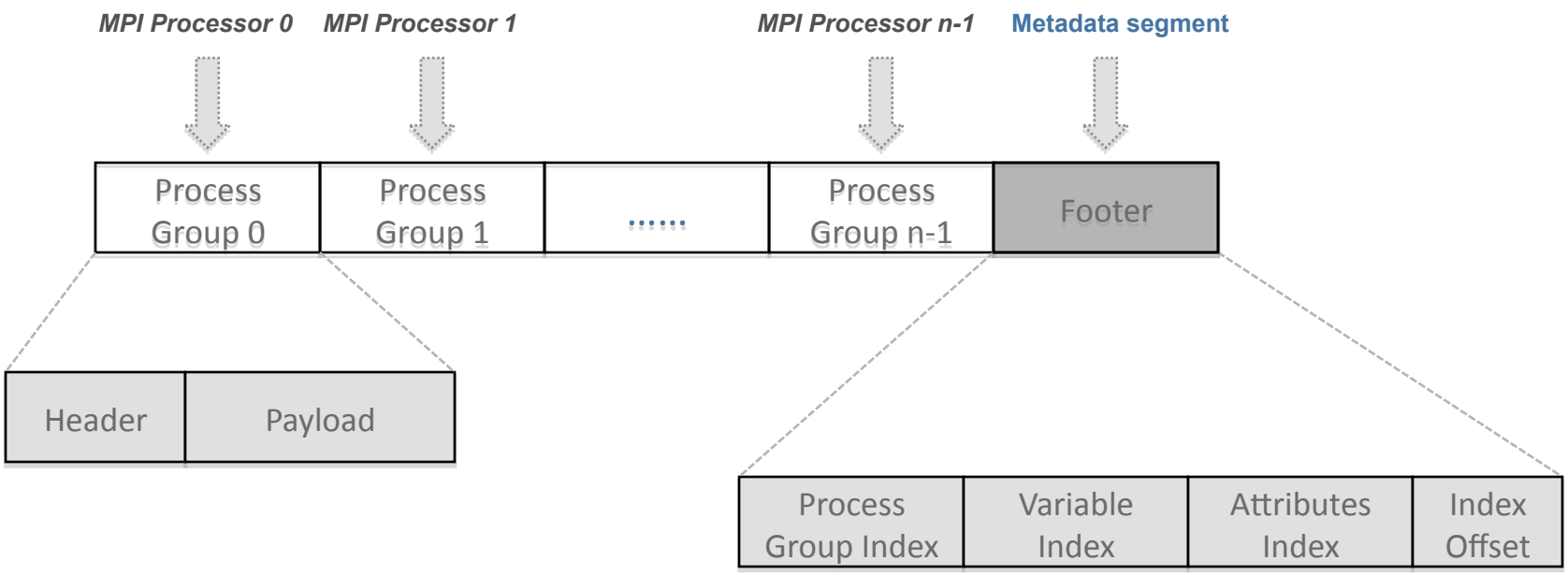



SDM CENTER

CPES Center for Plasma Edge Simulation

OAK RIDGE National Laboratory

# ADIOS BP File Format




1) ADIOS BP File Format – single file case

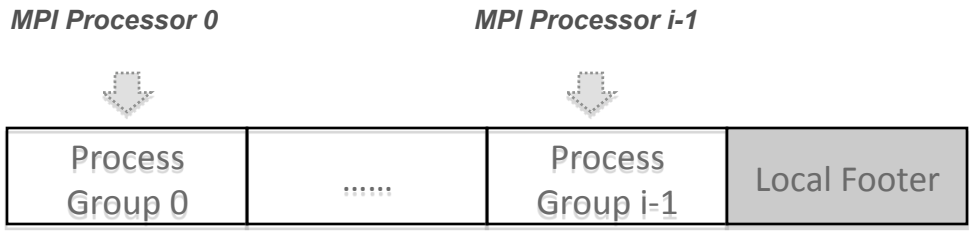
- Fault tolerance is critical for success of a parallel file format.
- Failure of a single writer is not fatal.
- Necessary to have a hierarchical view of the data (like HDF5).
- Tested at scale (140K processors for XGC-1) with over 20TB in a single file.

SDM CENTER

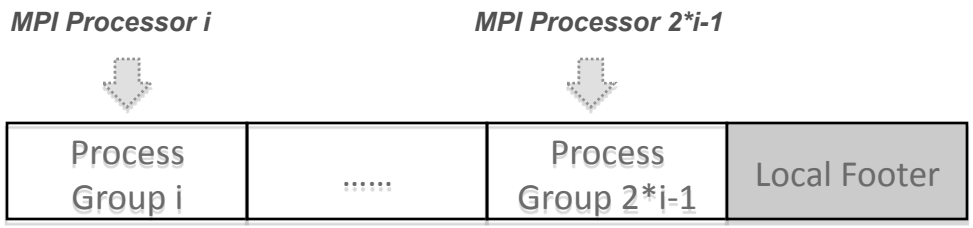
PES Center for Plasma Edge Simulation

OAK RIDGE National Laboratory

# ADIOS BP File Format

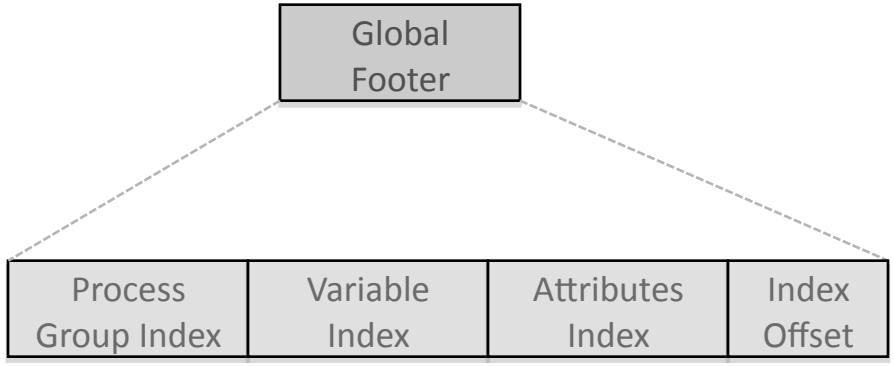



***Subfile 0***




***Subfile 1***

……




***Metadata file***

**1) ADIOS BP File Format – subfile case**



SDM CENTER
PES Center for Plasma Edge Simulation
OAK RIDGE National Laboratory

# What We have achieved

- ADIOS 1.2.1 was released on Aug 25th, 2010.
  - Can be downloaded from http://www.nccs.gov/user-support/adios/

- ADIOS Synchronous I/O Methods
  - MPI: single shared BP file.
  - MPI_LUSTRE: single shared BP file with Lustre stripe aligned
  - MPI_AMR: - new in ADIOS 1.2.1, output multiple BP subfiles
  - PHDF5: single shared HDF5 file
  - NC4: - new in ADIOS 1.2.1, output single share netcdf 4 file
  - POSIX: one file per process, generated files such as restart.bp.0, restart.bp.1,…..

- ADIOS Asynchronous I/O Methods
  - Datatap: Gatech
  - DART: Rutgers
  - DIMES: Rutgers
  - NSSI: Sandia

# What We have achieved

- More statistics supported, such as min/max/standard dev/mean
  - Automatically computed during write operation
  - Can be easily enabled/disabled in your XML file
- ADIOS Read API's
  - Able to read any slice from a multi-dimensional array.
  - Matlab BP Reader
- Useful tools
  - gpp: genenerate gwrite file for you.
  - bpdump: dump out the entire file
  - bpls: list all the vars with min/max, print out any slice of a multi-dimensional array
  - bp2h5: convert BP to HDF5 file
  - bp2nc: convert BP to netCDF file

# How to use ADIOS to write out data

- User Fortran code

```
call adios_open (adios_handle, "particles"//char(0), fname, "w", comm,
    err)
#include "gwrite_particles.fh" (automatically genereated by gpp)
call adios_close (adios_handle,err)
```

- Sample XML file

```
<adios-group name="particles" coordination-communicator="comm">
    <var name="mype" type="integer"/>
    <var name="nparam" type="integer"/>
    <var name="nspecies" type="integer"/>
    <var name="numberpe" type="integer"/>
    <var name="nparam*numberpe" type="integer" />
    <var name="nparam*mype" type="integer" />
    <var name="ntracke" type="integer" gwrite="ntrackp(2)"/>
<global-bounds dimensions="nparam*numberpe,ntracke" offsets="nparam*mype,0">
    <var name="electrons" type="real" dimensions="nparam,ntracke"
    gwrite="ptrackede(:,1:ntrackp(2))"/>
</global-bounds>
</adios-group>
<transport method="MPI" group="particles"/>
```

SDM CENTER

CPES Center for Plasma Edge Simulation

OAK RIDGE National Laboratory

# ADIOS Read API

```
typedef struct {
        uint64_t fh;                /* File handler                                          */
        int     groups_count;       /* Number of adios groups in file                        */
        int     tidx_start;         /* First timestep in file, usually 1                     */
        int     ntimesteps;         /* Number of timesteps in file. There is always at least one timestep   */
        char    ** group_namelist;  /* Names of the adios groups in the file (cf. groups_count)  */
 ......
} ADIOS_FILE;

typedef struct {
        uint64_t gh;                /* Group handler                                         */
        int     grpid;              /* group index (0..ADIOS_FILE.groups_count-1)            */
        int     vars_count;         /* Number of variables in this adios group               */
        char    ** var_namelist;    /* Variable names in a char* array                       */
        int     attrs_count;        /* Number of attributes in this adios group              */
        char    ** attr_namelist;   /* Attribute names in a char* array                      */
        ADIOS_FILE * fp;            /* pointer to the parent ADIOS_FILE struct               */
} ADIOS_GROUP;
```

SDM CENTER

CPES Center for Plasma Edge Simulation

OAK RIDGE National Laboratory

# ADIOS Read API

```
typedef struct {
    int      grpid;          /* group index (0..ADIOS_FILE.groups_count-1)                  */
    int      varid;          /* variable index (0..ADIOS_GROUP.var_count-1)                 */
    enum ADIOS_DATATYPES type;  /* type of variable                                         */
    int      ndim;           /* number of dimensions, 0 for scalars                         */
    uint64_t * dims;         /* size of each dimension                                      */
    int      timedim;        /* -1: variable has no timesteps in file, >=0: which dimension is time  */
    void     * value;        /* value of a scalar variable, NULL for array.                 */
    void     * gmin;         /* minimum value in an array variable, = value for a scalar    */
    void     * gmax;         /* maximum value of an array variable (over all timesteps)     */
    void     * mins;         /* minimum per each timestep (array of timestep elements)      */
    void     * maxs;         /* maximum per each timestep (array of timestep elements)      */
} ADIOS_VARINFO;
```

Center for Plasma Edge Simulation

Oak Ridge Leadership Computing Facility

SDM Center

Oak Ridge National Laboratory

# ADIOS Read API

1. open restart.bp file

```
ADIOS_FILE * f = adios_fopen ("restart.bp", MPI_COMM_WORLD);
```

2. open a ADIOS group called "temperature"

```
ADIOS_GROUP * g = adios_gopen (f, "temperature");
```

3. inquire the variable you want to read by its ID

```
for (i = 0; i < g->vars_count; i++) {
    ADIOS_VARINFO * v = adios_inq_var_byid (g, i);
}
```

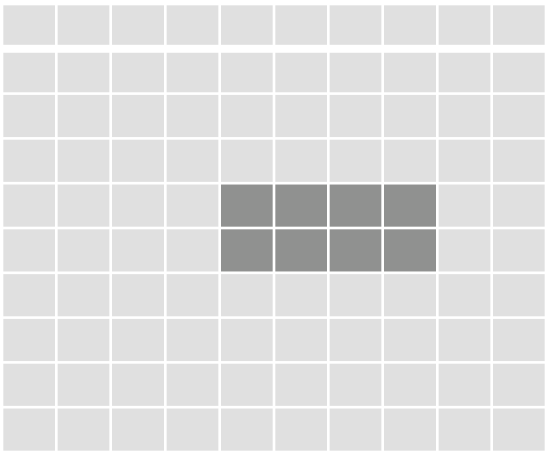
or a more common way is to inquire var by its name

```
ADIOS_VARINFO * v = adios_inq_var (g, "v2");
```

4. read data

```
bytes_read = adios_read_var (g, "v2", start, count, data);
```

10x10 2D array



start[0] = 4;
start[1] = 4;
count[0] = 2;
count[1] = 4;

# bpls

- **$ time bpls -l record.bp -v**

```
  of groups:    1
  of variables:  32
  of attributes: 0
  time steps:    10 starting from 1
  file size:     162 GB
  bp version:    1
Group record:
  double   /time          {10} = 0.003 / 0.03
  integer  /itime         {10} = 3 / 30
  double   /dt            {10} = 0.001 / 0.001
  integer  /nvar          scalar = 8
  integer  /dimensions/nxd+2  scalar = 1026
  integer  /dimensions/nyd+2  scalar = 514
  integer  /dimensions/nzd+2  scalar = 514
  double   /var/v1        {10, 514, 514, 1026} = 1 / 1
  double   /var/v2        {10, 514, 514, 1026} = -2.07946e-06 / 3.43263e-08
  double   /var/v3        {10, 514, 514, 1026} = -1.17581e-10 / 1.24015e-10
  double   /var/v4        {10, 514, 514, 1026} = -3.65092e-13 / 3.65092e-13
  double   /var/v5        {10, 514, 514, 1026} = -7.95953e-11 / 7.95953e-11
  double   /var/v6        {10, 514, 514, 1026} = -0.184178 / 0.0123478
  double   /var/v7        {10, 514, 514, 1026} = -0.000488281 / 0.984914
  double   /var/v8        {10, 514, 514, 1026} = 0 / 0
  byte     /name/v1_name  {20} = 32 / 111
  byte     /name/v2_name  {20} = 32 / 94
  byte     /name/v3_name  {20} = 32 / 94
  byte     /name/v4_name  {20} = 32 / 94
  byte     /name/v5_name  {20} = 32 / 94
  byte     /name/v6_name  {20} = 32 / 94
  byte     /name/v7_name  {20} = 32 / 94
  byte     /name/v8_name  {20} = 32 / 101
  integer  /bconds        {48} = -4 / 7
real    0m2.091s
```

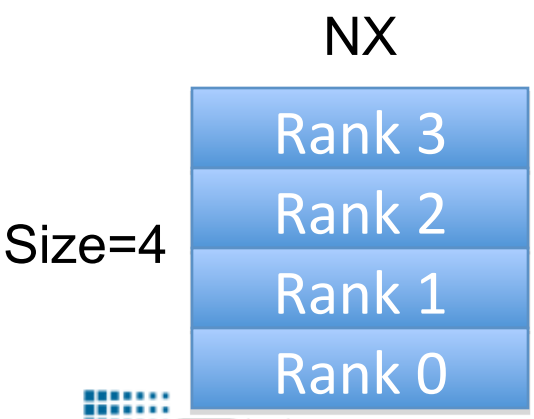
# Example (XML Configuration)

```
<?xml version="1.0"?>
<adios-config host-language="C">
    <adios-group name="temperature" coordination-communicator="comm">
    <var name="NX" type="integer"/>
    <var name="size" type="integer"/>
    <var name="rank" type="integer"/>
    <global-bounds dimensions="size,NX" offsets="rank,0">
      <var name="temperature" gwrite="t" type="double" dimensions="1,NX"/>
    </global-bounds>
    <attribute name="description" path="/temperature" value="Global array writte
n from 'size' processes" type="string"/>
</adios-group>

<method group="temperature" method="MPI_AMR">stripe_count=1;stripe_size=10485760
;num_aggregators=1;merging_pgs=0</method>

<buffer size-MB="2" allocate-time="now"/>

</adios-config>
```




Center for Plasma Edge Simulation

OLCF Oak Ridge Leadership Computing Facility

SDM Center

Oak Ridge National Laboratory

# Example (User Code)

```
#include <stdio.h>
#include <string.h>
#include "mpi.h"
#include "adios.h"
int main (int argc, char ** argv)
{
        char        filename [256];
        int         rank, size, i;
        int         NX = 10;
        double      t[NX];
        MPI_Comm    comm = MPI_COMM_WORLD;

        /* ADIOS variables declarations for matching gwrite_temperature.ch */
        int         adios_err;
        uint64_t    adios_groupsize, adios_totalsize;
        int64_t     adios_handle;

        MPI_Init (&argc, &argv);
        MPI_Comm_rank (comm, &rank);
        MPI_Comm_size (comm, &size);

        for (i = 0; i < NX; i++)
                t[i] = rank*NX + i;

        strcpy (filename, "adios_global.bp");

        adios_init ("adios_global.xml");

        adios_open (&adios_handle, "temperature", filename, "w", &comm);
        #include "gwrite_temperature.ch"
        adios_close (adios_handle);

        MPI_Barrier (comm);

        adios_finalize (rank);

        MPI_Finalize ();
        return 0;
}
```

I/O portion

Center for Plasma E

# Example (gwrite_temperature.ch)

```
adios_groupsize = 4 \
                + 4 \
                + 4 \
                + 8 * (1) * (NX);
adios_group_size (adios_handle, adios_groupsize, &adios_totalsize);
adios_write (adios_handle, "NX", &NX);
adios_write (adios_handle, "size", &size);
adios_write (adios_handle, "rank", &rank);
adios_write (adios_handle, "temperature", t);
```

# Example

- First run with “MPI” method

```
<method group="temperature" method="MPI">stripe_count=1;stripe_size=10485760;num_aggregators=1;merging_pgs=0</method>
```

```
qliu@jaguarpf-login1:/tmp/work/qliu/demo> aprun -n 16 ./adios_global
Application 3929776 resources: utime 0, stime 0
qliu@jaguarpf-login1:/tmp/work/qliu/demo> ls
adios_global  adios_global.bp  adios_global.xml
```

- Swith to “POSIX” method

```
<method group="temperature" method="POSIX">stripe_count=1;stripe_size=10485760;num_aggregators=1;merging_pgs=0</method>
```

```
qliu@jaguarpf-login1:/tmp/work/qliu/demo> aprun -n 16 ./adios_global
Application 3929834 resources: utime 0, stime 0
qliu@jaguarpf-login1:/tmp/work/qliu/demo> ls
adios_global  adios_global.bp  adios_global.bp.dir  adios_global.xml
qliu@jaguarpf-login1:/tmp/work/qliu/demo> ls adios_global.bp.dir
adios_global.bp.0   adios_global.bp.11  adios_global.bp.14  adios_global.bp.3  adios_global.bp.6  adios_global.bp.9
adios_global.bp.1   adios_global.bp.12  adios_global.bp.15  adios_global.bp.4  adios_global.bp.7
adios_global.bp.10  adios_global.bp.13  adios_global.bp.2   adios_global.bp.5  adios_global.bp.8
```

# Example

- First run with “MPI_AMR” method

```
<method group="temperature" method="MPI_AMR">stripe_count=1;stripe_size=10485760;num_aggregators=2;merging_pgs=0</method>
```

```
qliu@jaguarpf-login1:/tmp/work/qliu/demo> aprun -n 16 ./adios_global
Application 3929889 resources: utime 0, stime 0
qliu@jaguarpf-login1:/tmp/work/qliu/demo> ls
adios_global  adios_global.bp  adios_global.bp.dir  adios_global.xml
qliu@jaguarpf-login1:/tmp/work/qliu/demo> cd adios_global.bp.dir/
qliu@jaguarpf-login1:/tmp/work/qliu/demo/adios_global.bp.dir> ls
adios_global.bp.0  adios_global.bp.1
qliu@jaguarpf-login1:/tmp/work/qliu/demo/adios_global.bp.dir> lfs getstripe *

     adios_global.bp.0
             obdidx           objid          objid            group
                  0         8332409       0x7f2479                0

     adios_global.bp.1
             obdidx           objid          objid            group
                  1         8237559       0x7db1f7                0
```

Center for Plasma Edge Simulation

OLCF Oak Ridge Leadership Computing Facility

SDM Center

Oak Ridge National Laboratory

# Example (reading)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "mpi.h"
#include "adios_read.h"

int main (int argc, char ** argv)
{
    char        filename [256];
    int         rank, size, i, j;
    MPI_Comm    comm = MPI_COMM_WORLD;
    void * data = NULL;
    uint64_t start[2], count[2], bytes_read = 0;

    MPI_Init (&argc, &argv);
    MPI_Comm_rank (comm, &rank);
    MPI_Comm_size (comm, &size);

    ADIOS_FILE * f = adios_fopen ("adios_global.bp", comm);
    if (f == NULL)
    {
        printf ("%s\n", adios_errmsg());
        return -1;
    }

    ADIOS_GROUP * g = adios_gopen (f, "temperature");
    if (g == NULL)
    {
        printf ("%s\n", adios_errmsg());
        return -1;
    }

    ADIOS_VARINFO * v = adios_inq_var (g, "temperature");
```

# Example (reading)

```
uint64_t slice_size = v->dims[0]/size;
start[0] = slice_size * rank;
if (rank == size-1) /* last rank may read more lines */
    slice_size = slice_size + v->dims[0]%size;
count[0] = slice_size;

start[1] = 0;
count[1] = v->dims[1];


data = malloc (slice_size * v->dims[1] * sizeof (double));
if (data == NULL)
{
    fprintf (stderr, "malloc failed.\n");
    return -1;
}

bytes_read = adios_read_var (g, "temperature", start, count, data);

for (i = 0; i < slice_size; i++) {
    printf ("rank %d: [%lld,%d:%lld]", rank, start[0]+i, 0, slice_size);
    for (j = 0; j < v->dims[1]; j++)
        printf (" %6.2g", * ((double *)data + i * v->dims[1] + j));
    printf ("\n");
}

free (data);

adios_gclose (g);
adios_fclose (f);

MPI_Barrier (comm);

MPI_Finalize ();
return 0;
```

# ADIOS MPI_LUSTRE Method

- Improved version of MPI method.
- The file is written out with Lustre stripe-aligned.
- Automatically set Lustre I/O parameters from XML file.

i.e., stripe count, stripe size and write block size.

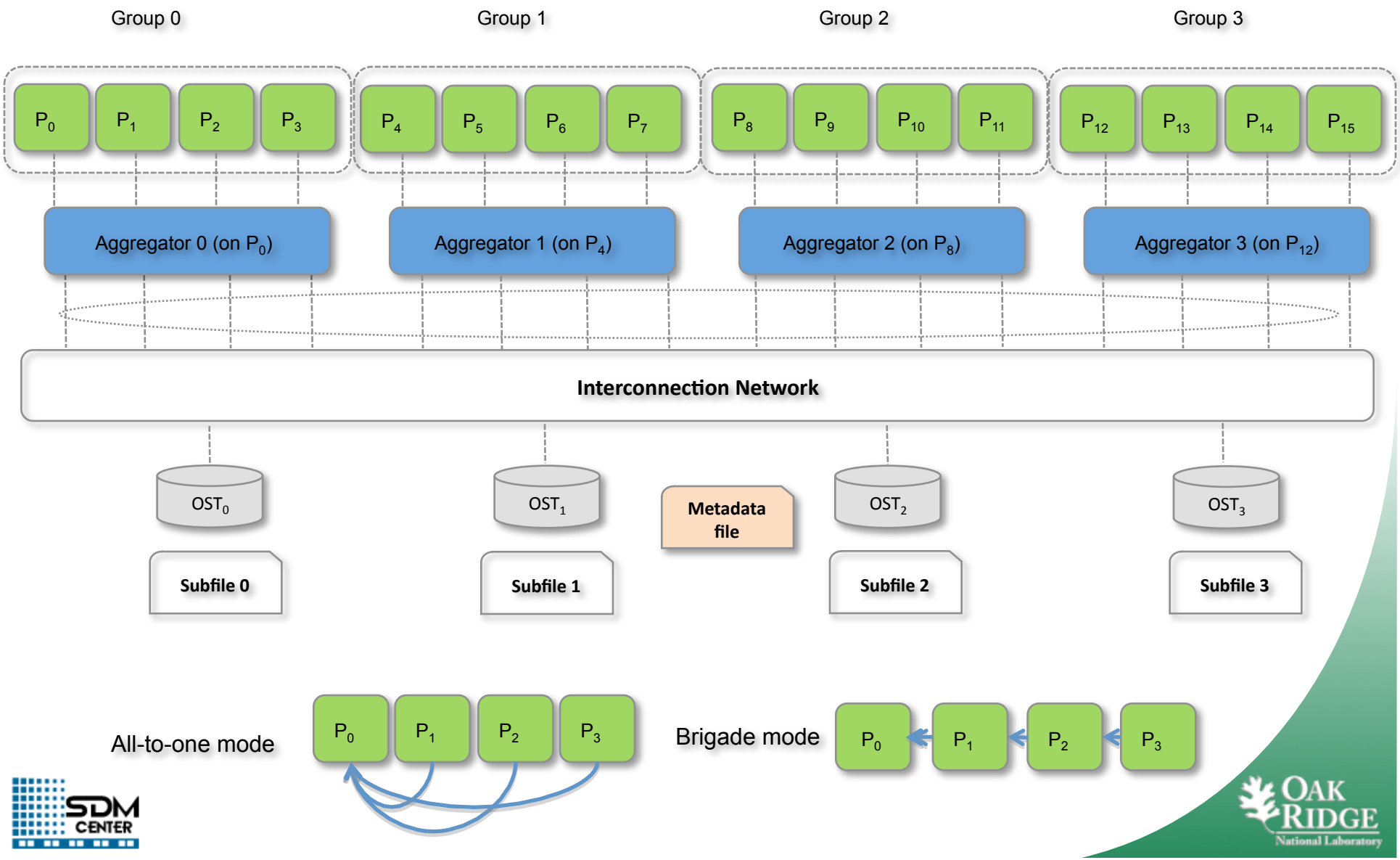
For example, to stripe your file on 16 OST's with stripe size 4MB and write block size 512KB,

```
<method group="temperature" method="MPI_LUSTRE">
  stripe_count=16;stripe_size=4194304;block_size=524288
</method>
```

# ADIOS MPI_AMR Method

- Newly developed method that further improves IO performance on Lustre parallel file system
- Key improvements:
  - Eliminate lock contention: Write out multiple subfiles with each file striped on 1 storage target (OST)
  - Aggregate IO among processors: Selected processors gathers all the data and write them out in a big chunk
  - Threaded file opens: Simulation can continue while waiting for file to be opened
  - Good usability: Other than telling ADIOS the # of aggregators to use, everything is the same as writing/reading one file to users.

# ADIOS MPI_AMR Method
# How does it work?



# ADIOS MPI_AMR Method
# How to use it?

- Suppose your code runs on 16 MPI processors with each output 8 MB
- If you want to an aggregation ratio of 2, you set num_aggregator to 16/2=8
- Each aggregator now will output 8*2=16MB

```
<method group="temperature" method="MPI_AMR">
  stripe_count=1;stripe_size=10485760;num_aggregators=8;merging_pgs=0;threading=0
</method>
```

- Output will look like

  restart.bp

  restart.bp.dir/restart.bp.0

  restart.bp.dir/restart.bp.1

……

  restart.bp.dir/restart.bp.7

- To read data back

  bpls restart.bp -lav

Center for Plasma Edge Simulation

Oak Ridge Leadership Computing Facility

SDM Center

Oak Ridge National Laboratory

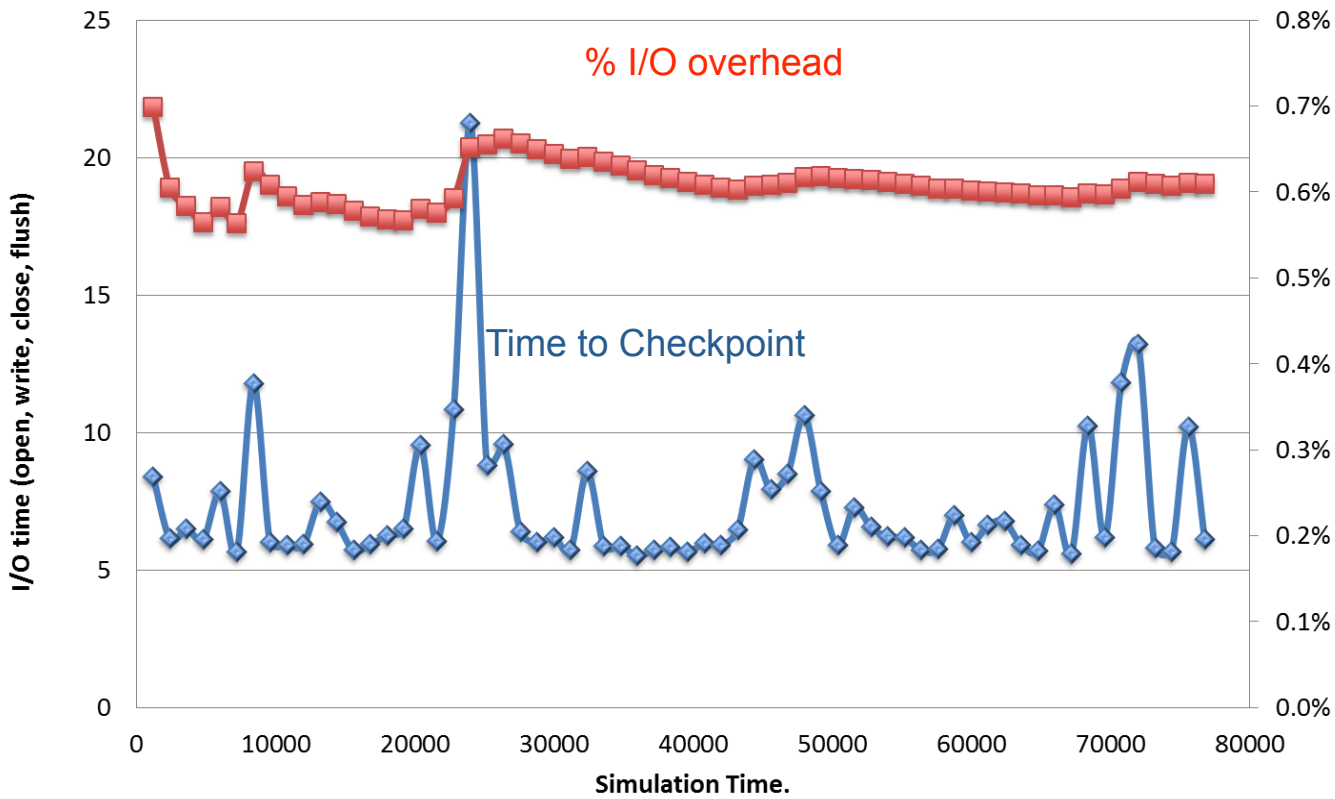
# ADIOS MPI_AMR Method How to use it?

- The best practice of using MPI_AMR
  - Post adios_open and adios_group_size calls as early as possible in your code.
  - Pick the right number of aggregators.

```
……
call adios_open (adios_handle, "temperature", filename, "w",
   comm, adios_err)
call adios_group_size (adios_handle, adios_groupsize,
   adios_totalsize, adios_err)
……
!Do you calculations here
……
call adios_write (adios_handle, "NX", NX, adios_err)
call adios_write (adios_handle, "size", size, adios_err)
call adios_write (adios_handle, "rank", rank, adios_err)
call adios_write (adios_handle, "temperature", t, adios_err)
call adios_close (adios_handle, adios_err)
```

# MPI_AMR Performance

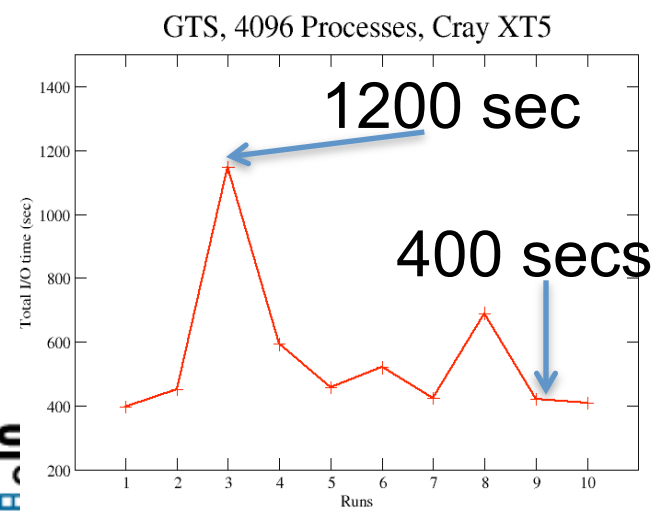
- S3D (Combustion code) ***production*** runs on Jaguar XT5 (96K core)



# I/O Variability Issue

- Motivations
  - Large-scale clusters are seeing significant I/O variability.
    - I/O interferences, e.g., multiple users hit the same OST at the same time.
    - Network resource congestions. Network bandwidth is limited and shared.
  - With more and more cores available on high-performance computer, the I/O subsystem becomes more and more variable. One “bad” user can slow down the overall I/O performance dramatically.
  - Unfortunately, Lustre parallel file system does not guarantee QoS.

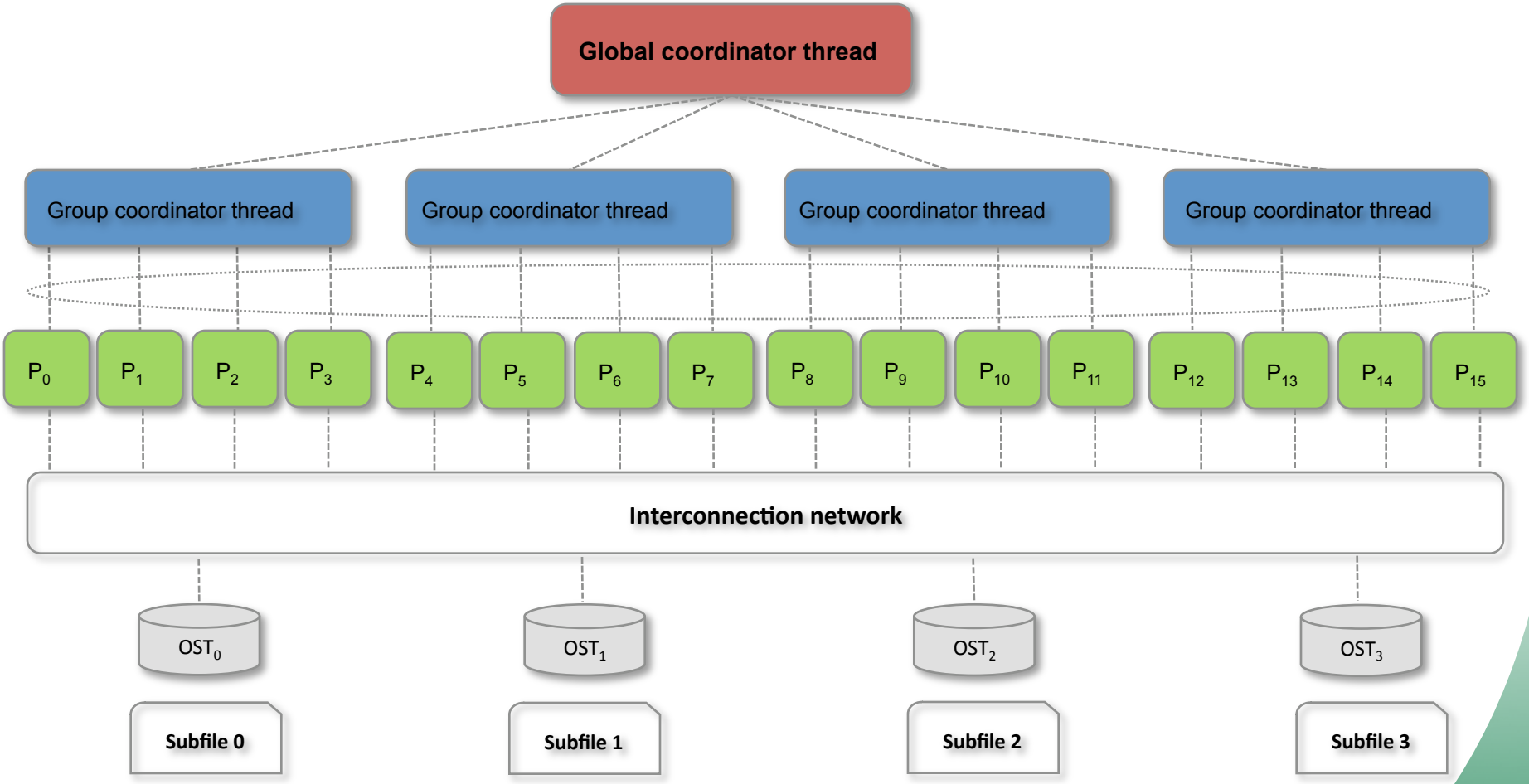
- Key ideas
  - Can we achieve better I/O performance by avoiding congestion??



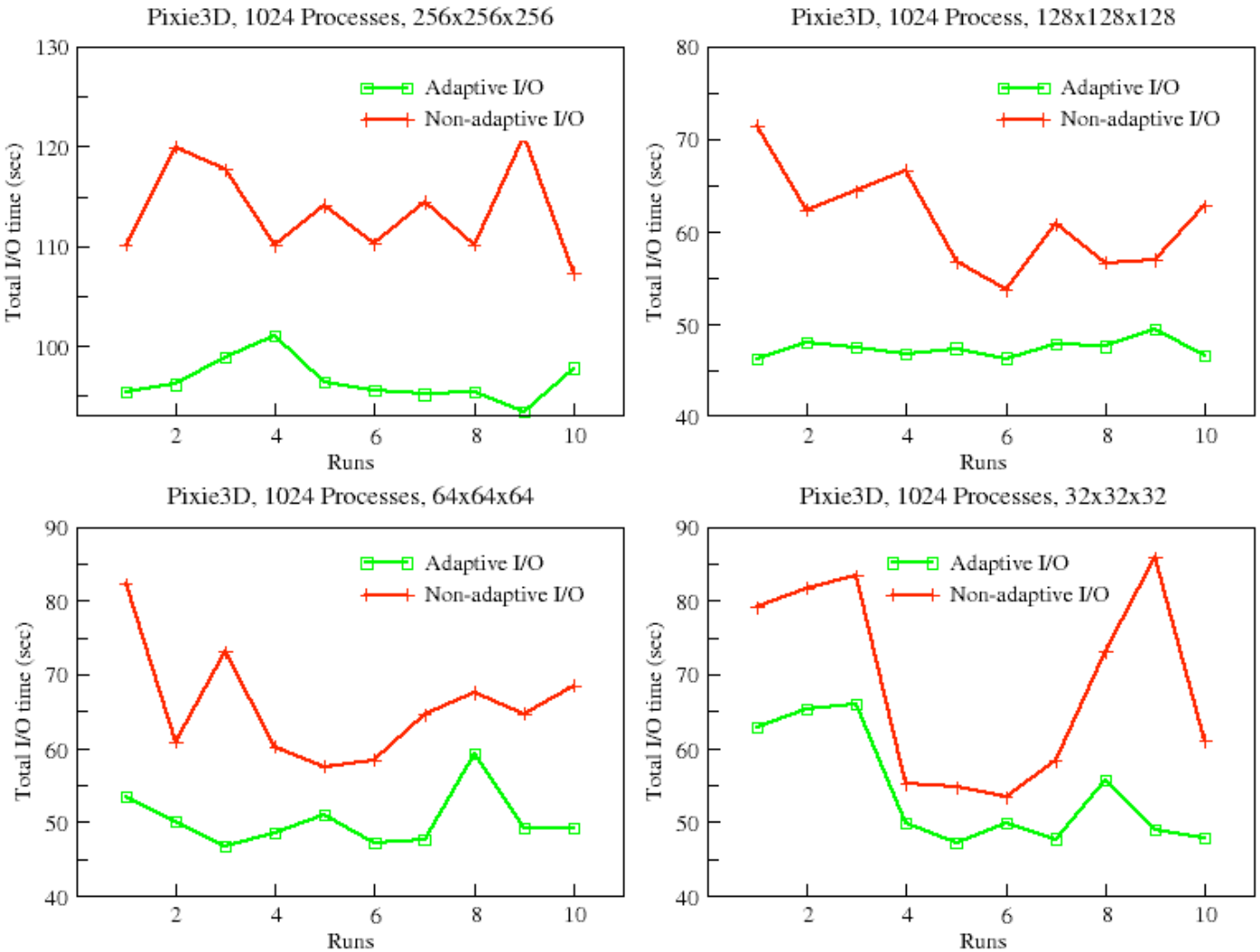

Center for Plasma Edge Simulation

OLCF Oak Ridge Leadership Computing Facility

# Adaptive I/O Control



# Results (I/O Time)



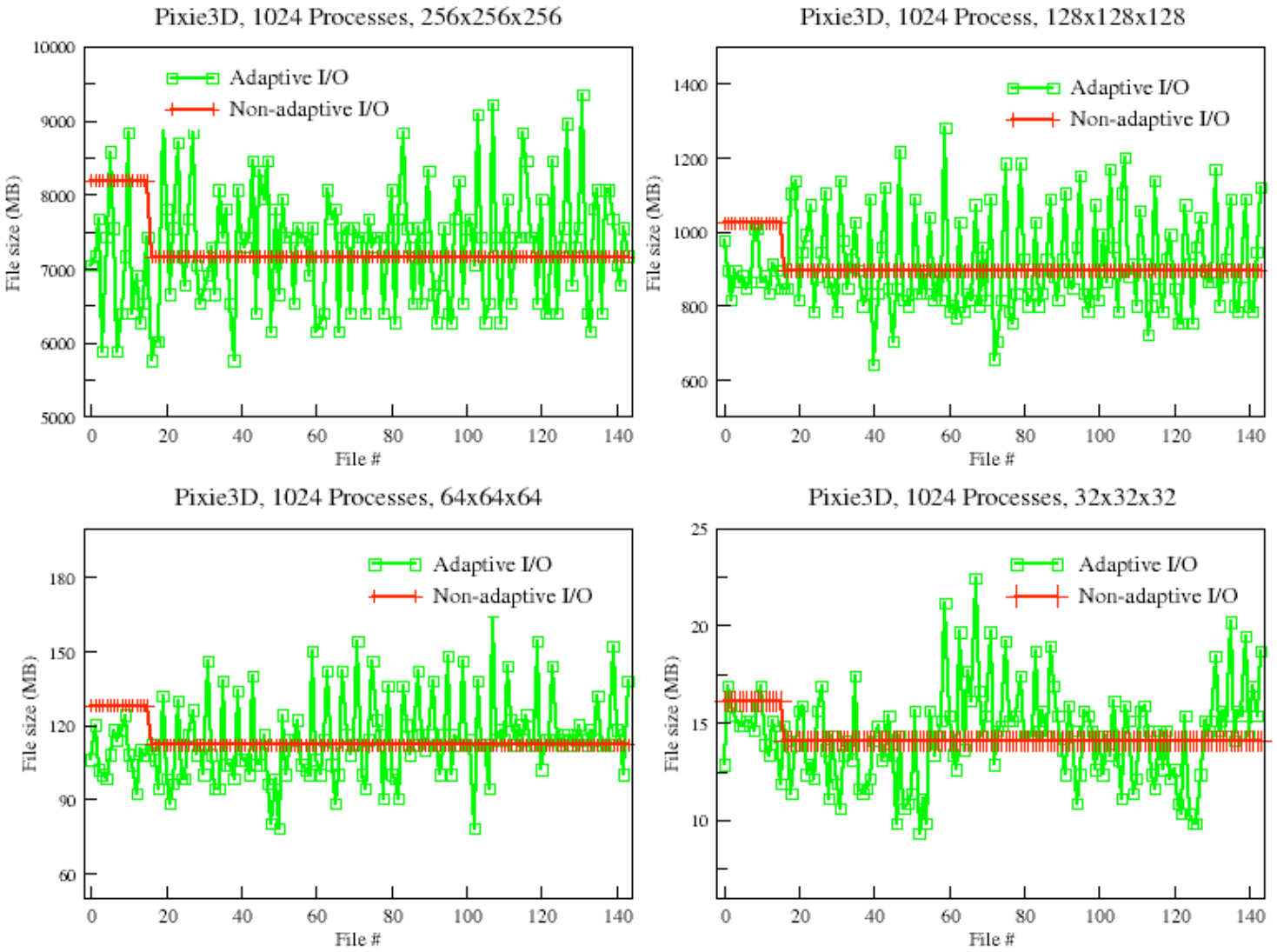

Center for Plasma Edge Simulation

OLCF Oak Ridge Leadership Computing Facility

SDM Center

Oak Ridge National Laboratory

# Results (File Size)



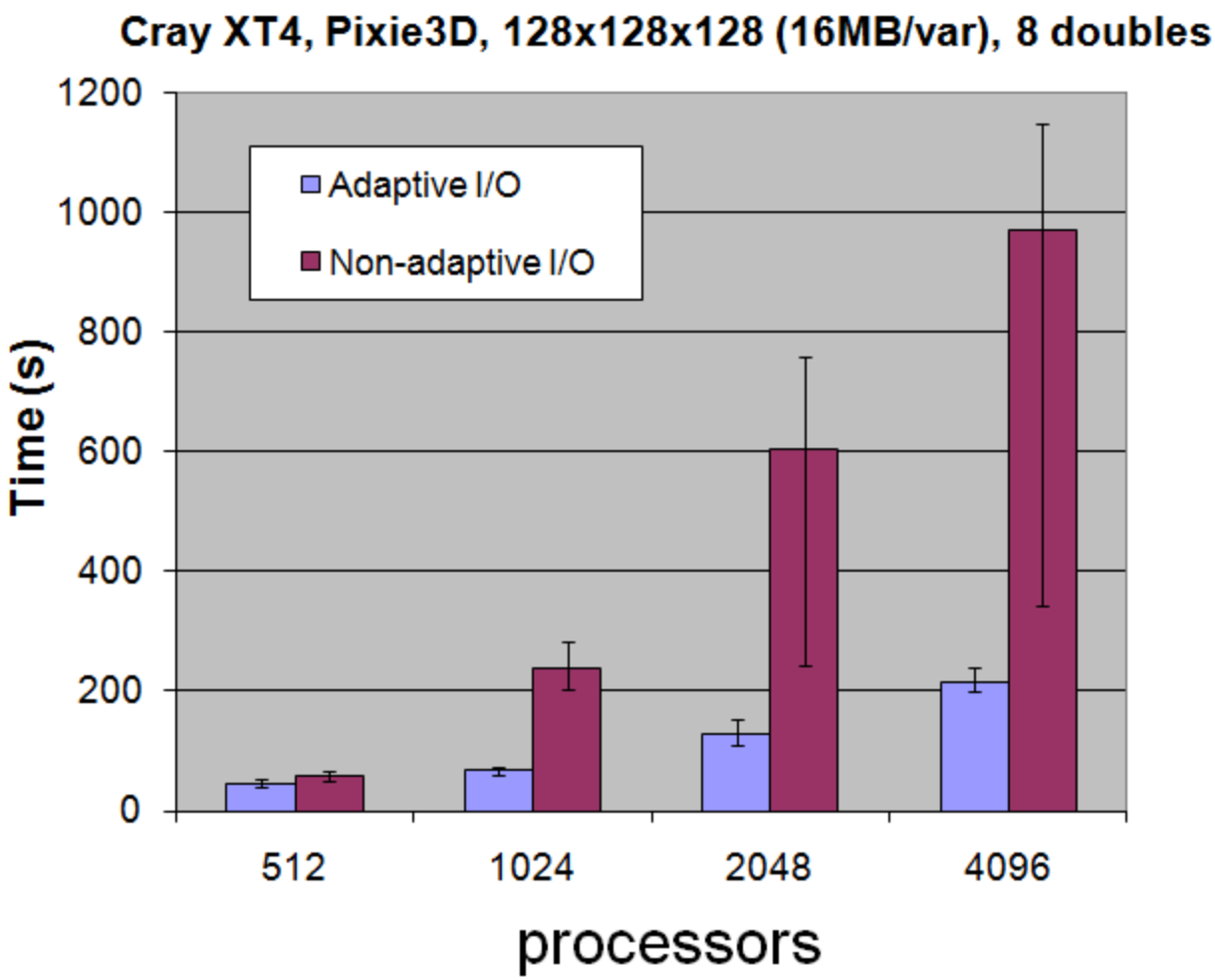

Center for Plasma Edge Simulation

Oak Ridge Leadership Computing Facility

CENTER

Oak Ridge National Laboratory

# I/O Time w. Manual Noise Injection




Center for Plasma Edge Simulation

Oak Ridge Leadership Computing Facility

SDM Center

Oak Ridge National Laboratory

# Conclusions

- ADIOS is an I/O componentization framework that
  - Has been proven for extreme scale performance on massively parallel systems
  - Allows for both synchronous and asynchronous I/O transports.
  - Contains a new, metadata rich, I/O format that can allow for extreme scale I/O.
- The ADIOS BP file format is a non-contiguous log-file format, which has shown extreme scalability for both write and read access.
- ADIOS 1.2.1 is available at http://www.nccs.gov/user-support/center-projects/adios/