Autocorrelation with ADIOS

What is ADIOS

An extendable framework that allows developers to *plug-in*

- **I/O methods**: N-to-M, N-to-N, N-to-1, In Situ (aka Staging)
- **Transformations**: Compression, Decompression, Indexing
- **Self describing** data format: ADIOS-BP
- **Indexing/Querying**: MinMax, FastBit, Alacrity

Incorporates the “best” practices in the I/O middleware layer

Released twice a year, now 1.12, under the completely free BSD license

- https://www.olcf.ornl.gov/center-projects/adios
- https://github.com/ornladios/ADIOS

Available at ALCF, OLCF, NERSC, CSCS, Tianhe-1,2, Pawsey SC, Ostrava

Applications are supported through OLCF INCITE program

Outreach via on-line manuals, and live tutorials

How to use ADIOS

ADIOS is provided as a library to users; use it like other I/O libraries, except

ADIOS has a simple approach for I/O

- User defines in application source code: “what” and “when”
  - **Every process defines what data and when to output**
- ADIOS takes care of the “how”

Biggest hurdle for users:

- Forget all of your manual **tricks** to gain I/O performance on your particular target system and target scale and just say what you want to write/read
- Trust ADIOS to deliver the performance

Performance Portability:

- Write once, perform well anywhere
  - It comes naturally with ADIOS
  - ADIOS has many different I/O methods (strategies)

Data management tradeoffs at exascale → to hybrid staging

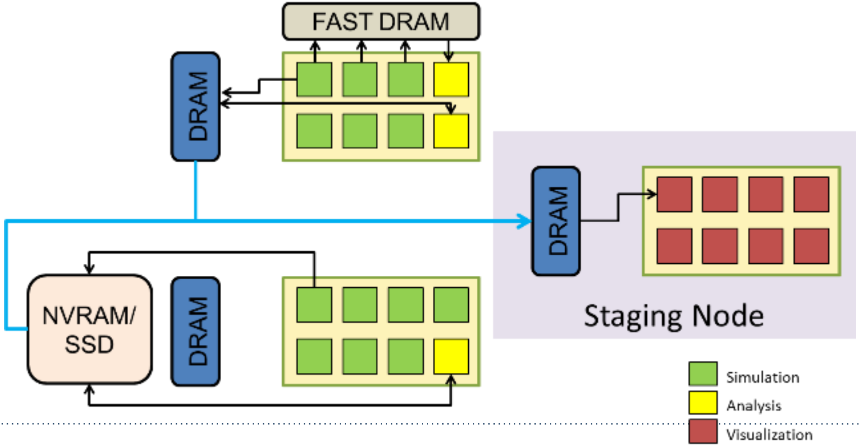
Explore node layout choices for data management

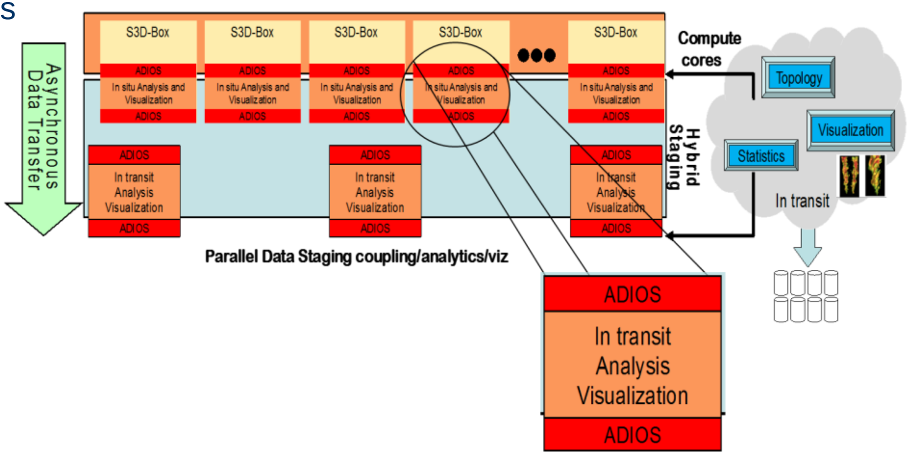
Balance of memory size and speed

Feedback for node designs with NVRAM, larger memory, on-chip NIC

Network throughput and latency impact on SDMA tasks

Placement of operations in concert with solver and network topology




Asynchronous Data Transfer
S3D-Box
S3D-Box
S3D-Box
S3D-Box
●●●
S3D-Box
ADIOS
In situ Analysis and Visualization
ADIOS
Compute cores
Hybrid Staging
ADIOS
In transit Analysis Visualization
ADIOS
Topology
Visualization
Statistics
In transit
Parallel Data Staging coupling/analytics/viz
ADIOS
In transit Analysis Visualization
ADIOS

Goals of the ADIOS Read API design

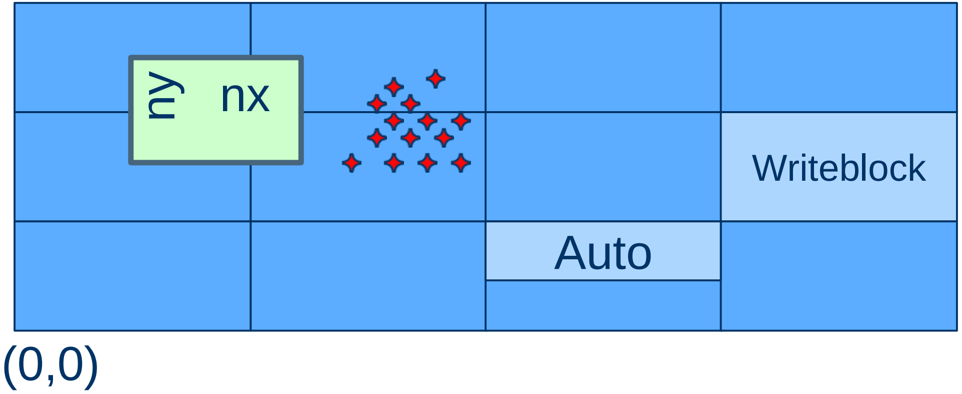
Staging I/O

- Insulate the scalable application from the variability inherent in the file system
- Enable the utilization of in situ and in-transit analytics and visualization

Same API for reading data from files and from staging

Allow for read optimizations:

- Multiple read operations can be scheduled before performing them
- Allow for blocking and non-blocking reads
- Use generic selections in the read statements instead of describing a bounding box
- Option to let ADIOS deliver data in chunks, with memory allocated inside ADIOS not in user-space



Selections

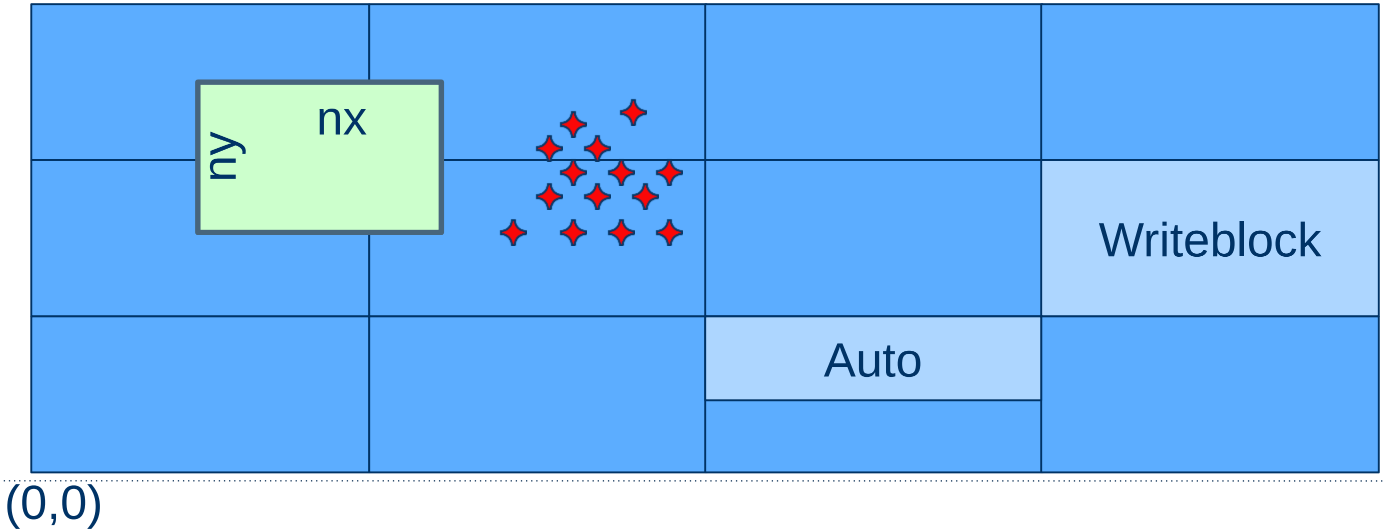
ADIOS_SELECTION *

adios_selection_boundingbox (int ndim, uint64_t * offsets, uint64_t * readsize)

adios_selection_points (uint64_t ndim, uint64_t npoints, uint64_t *points)

adios_selection_writeblock (int index)

adios_selection_auto (char * hints)



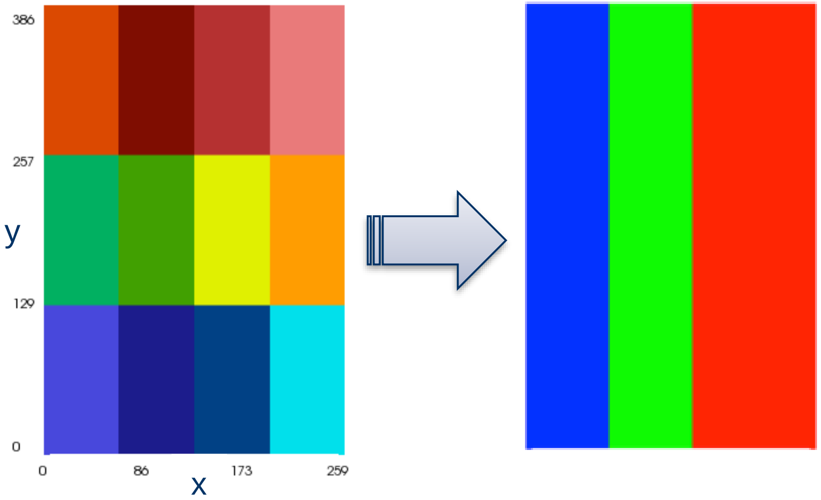
Example of Read API: read a variable step-by-step

```
int count[] = {10,10,10};
int offs[] = {5,5,5};
P = (double*) malloc (sizeof(double) * count[0] * count[1] * count[2]);
Q = (double*) malloc (sizeof(double) * count[0] * count[1] * count[2]);
ADIOS_SELECTION *sel = adios_select_boundingbox (3, offs, count);
while (fp != NULL) {
    adios_schedule_read (fp, sel, "P", 0, 1, P);
    adios_schedule_read (fp, sel, “Q", 0, 1, Q);
    adios_perform_reads (fp, 1, NULL);   // 1: blocking read
    // P and Q contains the data at this point
    adios_release_step (fp); // staging method can release this step
    // ... process P and Q, then advance the step
    adios_advance_step (fp, 0, 60.0);
    // 60 sec blocking wait for the next available step
}
// free ADIOS resources
adios_free_selection (sel);
```

N to M reorganization with stage_write

heat transfer + stage_write running together

- Write out 6 time-steps.
- Write from 12 cores, arranged in a 4 x 3 arrangement.
- Read from 3 cores, arranged as 1x3



N to M reorganization with stage_write

```
$ cd ~/Tutorial/heat_transfer
edit heat_transfer.xml  (vi, gedit)
set method to MPI
□<method group="heat" method="MPI"/>

$ mpirun -np 12 ./heat_transfer_adios1   heat  4 3    40 50   6 500
$ bpls -D  heat.bp   T
  double    T              6*{150, 160}
        step 0:
          block  0: [  0: 49,    0: 39]
          block  1: [  0: 49,   40: 79]
          ...
          block 11: [100:149, 120:159]


$ mpirun -np 3    stage_write/stage_write    heat.bp     h_3.bp     BP  ""    FLEXPATH ""    3
$ bpls -D   h_3.bp   T
  double    T              6*{150, 160}
        step 0:
          block 0: [  0:149,    0: 52]
          block 1: [  0:149,   53:105]
          block 2: [  0:149,  106:159]
```

Live demo

- Live demo on virtual machine