Data staging for in-situ processing and parallel IO/Coupling of HPC applications

John Biddiscombe
Jerome Soumagne
Project/Funding

NextMuSE
(http://nextmuse.cscs.ch)

Next generation Multi-mechanics Simulation Environment

European Community’s Seventh Framework Programme (FP7/2007-2013)
Information and Communication Technologies (ICT)
Small or medium-scale focused research project - Specific Targeted Research Project (STREP)
Future and Emerging Technologies Open Scheme (FET-Open)
Project: Naval Partners: Particle Methods
Project: Mechanical heart valves

- Hinges
- Leaflets
- Diameter: 25 mm

Reynolds number $\approx 6000$

Design static since early 1980s
Project: TurboMachinery Partners
Requirements

 In-Situ Visualization
 Computational Steering
 Two way exchange of potentially large data
  (simulation has no meshing capability)
 Multiple codes/languages
 Scale solvers but still …
  ▪ Treat Cluster/Supercomputer like workstation extension

Constraints

 Partners who are far away
 Principally scientists as opposed to programmers
Ideal Solution

Visit/ParaView

Use this approach for in-situ visualization

Memory pressures
*(especially for time-dependent data)*

If Vis/Analysis code doesn’t scale as well as simulation code, possible problem
Parallel IO (c.f. Lustre)

HDF5 Parallel IO already looks like this;
And we encourage HDF5 usage

Loose Coupling

All IO goes through OSS nodes
DSM Implementation (à la Data-Staging)

DSM Model could look like this

Just replace the IO layer in HDF5

HDF IO intercepted and routed to DSM

IO Forwarding c.f. ADIOS etc
Loose Coupling: Configurations

1. **Simulation Machine** with DSM connected to (Infiniband) Switch with DSM, then to **Analysis/Vis Machine** with Render GUI, and finally to **Workstation**.

2. **Simulation Nodes** with DSM, connected to **Analysis/Vis Nodes** with Render GUI, then to **Workstation**.

3. **Simulation Machine** with DSM connected to **Analysis/Vis Workstation**.
In-Situ Visualization – How to use H5FDdsm driver

Create a file access property list

```c
fapl_id = H5Pcreate(H5P_FILE_ACCESS);
```

Instead of using the MPI-IO driver

```c
H5Pset_fapl_mpio(fapl_id, MPI_COMM_WORLD, MPI_INFO_NULL);
```

Insert our DSM driver

```c
H5Pset_fapl_dsm(fapl_id, MPI_COMM_WORLD, NULL);
```

Then carry on as usual

```c
file_id = H5Fcreate(filename, H5F_ACC_TRUNC, H5P_DEFAULT, acc_plist_id);
H5Pclose(acc_plist_id);
H5Fclose(file_id);
```

Comment out any MPIO_COLLECTIVE dataset transfer properties
Data sent into HDF5 is then automatically redirected to this driver
- Transfer - ‘plug-in’ communicators MPI / MPI_RMA / DMAPP / TCP sockets
- Two/One sided use different synchronization methods

\[
\text{start} \quad 0 \quad \text{local length} \quad 2 \times (\text{local length}) \quad (N - 1) \times (\text{local length}) \quad N \times (\text{local length}) \quad \text{end of file (eof)}
\]

Network

more links = more BW

network layer = plugin
TCP / MPI / MPI_RMA
(Gemini/DMAPP)

EOF + metadata = total length of DSM

more links = more BW
HDF5 File Structure (Single Large Dataset)

- **Dataset header**
- **Metadata cache**
- **Datatype**
- **Dataspace**
- **Attributes**

**Application memory**

- **Single Dataset**: Not so common
- **Dataset data (distributed in parallel)**

**“File”**

- **Metadata**
- **Dataset data (linear HDF5 address space)**

- **Mapped to DSM**
  - 1
  - 2
  - ...
  - N

Parallel write = about N DSM links used at the same time
InfiniBand QDR 4X cluster—DSM size of 4GB and distributed among 16 processes (4 nodes).

Cray XK6 system—DSM size of 40GB and distributed among 160 processes (40 nodes).

Write transfer rate of an (in-memory) HDF5 file composed of one single dataset using a contiguous distribution—Comparison between MPI point-to-point and passive RMA transfers.
HDF5 File Structure (Multiple Datasets)

- **Dataset header**
  - Datatype
  - Dataspace
  - Attributes

- **Metadata cache**

- **Each dataset distributed over sim nodes**
  - Dataset data
  - Dataset data
  - Dataset data

- **“File”**
  - Metadata
  - Dataset data
  - Dataset data
  - Dataset data

- **Mapped to DSM**
  - 1
  - 2
  - ... N

Parallel write = only ~N/3 DSM links used at the same time
Redistribution Strategies

- Needed to add a more elaborate redistribution strategy to optimize the number of DSM links used at the same time

- Contiguous

- Block Cyclic

- Random Block
  - Same as previous one but blocks are shuffled (uses table lookup)
  - Other methods possible (see future work section)
Inter-node micro-benchmark (OSU): Cray XK6

Inter-node micro-benchmark using MPI two sided transfers.

Inter-node micro-benchmark using passive one-sided MPI transfers.
Inter-node micro-benchmark (home) : Cray XK6

Inter-node micro-benchmark using DMAPP put operations (blocking) with an FMA/BTE switch at 4KB, which effectively corresponds to a bandwidth drop point.
Bandwidth writing a N smaller datasets

Write transfer rate on Cray XK6 using MPI RMA and DMAPP communicators of an HDF5 file composed of 10 datasets—DSM size of 40GB and distributed among 160 processes (40 nodes).
Compare and Contrast

DataSpaces (Rutgers + ORNL)

- Don’t use HDF5 as their intrinsic mapping layer
- Can Map Data Objects using any method they choose
- (Hashing functions based on geometric properties: Map N dimensional entries to 1D address space)

- With HDF5
  We use Hyperslabs
  Library does the work
- only 2 connections
A “Real” Example – Coupling with SPH-Flow

![3D simulation of a flow coupling example with SPH-Flow.]
Part 2: Application to solvers

Visualization easy
In-Situ Visualization – Create Template

- Built on top of Xdmf model and format
- Generate Xdmf files to drive Xdmf reader plugin
- Uses in-memory H5dump to get the missing (metadata) information
- Much simpler to write
- Allows variable number of elements

```xml
<Grid>
  <Topology TopologyType="Polyvertex">
  </Topology>
  <Geometry GeometryType="XYZ">
    <DataItem>/fluid/position</DataItem>
  </Geometry>
  <Attribute>
    <DataItem>/fluid/density</DataItem>
  </Attribute>
  <Attribute>
    <DataItem>/fluid/velocity</DataItem>
  </Attribute>
  <Attribute>
    <DataItem>/fluid/vmob</DataItem>
  </Attribute>
  <Attribute>
    <DataItem>/fluid/w</DataItem>
  </Attribute>
</Grid>
```
The DSM is 2 way read/writeable (not simultaneously)

ParaView Servers

Set Steering Commands and Disabled Objects in metadata section

Read simulation data – write user defined steering commands and data into HDF5 “file” section

Write output data – Read user defined steering commands and data

Parallel Simulation

Get Steering Commands and Disabled Objects
Computational Steering

Each simulation requires custom controls

And different analysis requirements
Automatic GUI Generation: Interactions XML

Reuse ServerManager syntax from ParaView

Added some extra Properties

```xml
<DataExportProperty
    name="ModifiedBodyNodes"
    command="SetSteeringArray"
    label="Modified Body Node Data">
<DataExportDomain name="data_export"
    full_path="/Mesh_DataSet"
    geometry_path="/Mesh_Nodes#1/NewXYZ"
    topology_path="/Mesh_Nodes#1/NewCo...
    command_property="ReloadFreeBodyMesh">
</DataExportDomain>
</DataExportProperty>

Can also link GUI 3D widgets using ‘hints’
No ‘app-specific’ knowledge required

Generate ServerManager XML from template
parse hints/commands and other stuff
Register it internally on the fly
No need to (re)compile paraview plugin

Unless you have some custom heart valve adjustment widget that is specially designed for that app

Engineer only modifies simulation ‘Names’ and XML template
Wait Mode

Computation/Analysis not overlapped

Define some pipeline in the ParaView GUI which does something interesting (to send data back)

Can switch between modes arbitrarily (or every N iterations)

H5FD_dsm_steering_wait();
Free Mode

Computation/Analysis overlapped

Can switch between modes arbitrarily (or every N iterations)

Can store N time steps

File

Write Initialization

Analysis

Analysis

Analysis

GUI

File Create wipes DSM clean
So may want to add more sync options

Example - Erosion on Pelton Bucket
Parallel Simulation/Analysis + serial Client
Steering demo (Benny Hill Version)
Animated Wedge
Part 3 : Limitations

Future development must address

- Memory
- Locality - Piece Coherence
- Locality - Array Sharing
- Multiple clients
- Longer term storage (time domain comparisons)
- Trees of Data Objects (ownership)
Memory Copies: 3 possibly existing at once

H5DCreate
Allocate directly

HDF5 VOL Proposal
(DAOS from lustre talk)

Only see addresses - not objects
Piece Coherence

Dataset Pressure

Dataset Velocity

Dataset Density

Block Mapping (non-random block)

vtkDataArray

vtkDataArray

vtkDataArray
Locality: Data Sharing

General Purpose Node

Simulation Core
Simulation Core
Accelerator
Analysis Core
Analysis Core

Simulation

PGAS-like memory manager
e.g. OpenShmem

Analysis

Pressure array global: mapped to HDF5, mapped to Analysis

Time Series Analysis, Piece Locality, Comparisons, What-Ifs, Reference Counting
Final concluding remarks

Created
Easy to interface Computational Steering Framework

Need Memory Mapped File Array system
- Locality / Coherence / Sharing / Atomic / Index / Ref Counts / Persistence