DAMSEL - A Data Model Storage Library for Exascale Science

(This work is supported by Office of Advanced Scientific Computing Research under the program of X-stack Software Research)

Saba Sehrish

CScADS 2011

July 26, 2011

NORTHWESTERN UNIVERSITY

Outline

- Project Team
- Motivation
- Damsel I/O Library
- Usecases: FLASH, GCRM
- Proposed API and implementation, Data layout (In Progress)

Project Team

- **Northwestern University:** Alok Choudhary, Wei-keng Liao, Kui Gao, Saba Sehrish, Chen Jin, William Hendrix
- **Argonne National Laboratory:** Rob Ross, Rob Latham, Tim Tautges, Venkat Vishwanath
- **The HDF Group:** Quincey Koziol, Gerd Herber
- **NC State University:** Nagiza Samatova, Sriram Lakshminarasimhan

1. Motivation
   - Computational and Data Model Motifs
   - Existing I/O Libraries
   - Goals

Computational Model Motifs

Table 1: The expanded list of Computational Motifs (Dwarfs). Here, we have identified data models used in the motifs and provided illustrative examples. Some codes employ more than one motif. This project focuses on the top six (blue).

Motif	Data Model/ Data Structure	Examples
Dense Linear Algebra	**a**	BLAS, LAPACK, ScaLAPACK, Matlab,
Sparse Linear Algebra	**f**	OSKI, SuperLU, SpMV
Spectral Methods	**a**	FFT, Nek5000 (Nuclear Energy)
N-Body Methods	**b, e, j**	Molecular Dynamics, NN-Search
Structured Grids (+ AMR)	**a, b, c**	FLASH (Astrophysics), Chombo-based codes
Unstructured Grids (+ AMR)	**c**	UNIC, Phasta, SELFE numerical tsunami models
Monte Carlo, MapReduce	**a-l**	GFMC, EM, POV-Ray
Combinational Logic	**g, i**	RSA encryption, FastBit
Graph Traversal	**f, h**	Boost Graph Library (BGL), C4.5
Dynamic Programming	**a**	Smith-Waterman
String Searches	**d, e**	BLAST, HMMER
Backtrack and Branch-and-Bound	**f, i, g**	Clique, Kernel regression
Probabilistic Graphical Models	**h, k**	BBN, HMM, CRF
Finite State Machines	**l**	Collision detection

a–Multidimensional array, e.g., dense matrix in 2D; **b**–Point- or region-based quadtree, octree, compressed octree, or hyperoctree; **c**–Lattice model; **d**–Suffix tree, suffix array; **e**–R-tree, B-tree, X-tree, and their variants; **f**–Sparse matrix, e.g., block compressed sparse row (BCSR); **g**–Bitmap index, bitvector; **h**–Direct Acyclic Graph (DAG); **i**–Hash table, grid file; **j**–K-d tree; **k**–Junction tree; **l**–Transition table, Petri net.

Data Model Motifs

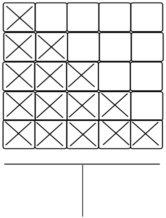
Data Models Exemplars			
Exemplar	Domain	Computational Motif	Data Model Motif
FLASH	Astrophysics	v	Structured Adaptive
MADNESS	Quantum Chemistry	i, iii, iv, vi	Unstructured Adaptive
Nek5000	Nuclear Energy	iii	Structured Adaptive
PHASTA	CFD	ii, vi	Unstructured Adaptive
S3D	Combustion	v	Structured Regular
UNIC	Neutron Transport	ii, vi	Unstructured Regular

i–Dense Linear Algebra; ii–Sparse Linear Algebra; iii–Spectral Methods; iv–N-Body Methods; v–Structured Grids (+ AMR); vi–Unstructured Grids (+ AMR).

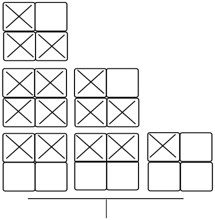
Existing I/O Libraries

- Storage data models developed in the 1990s; Network Common Data Format (netCDF) and Hierarchical Data Format (HDF)
- I/O library interfaces still based on low-level vectors of variables
- Lack of support for sophisticated data models, e.g. AMR, unstructured Grids, Geodesic grid, etc
- Require too much work at application level to achieve close to peak I/O performance

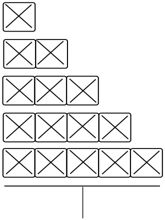
Example: Lower Triangle Matrix



netCDF: fixed dimensions

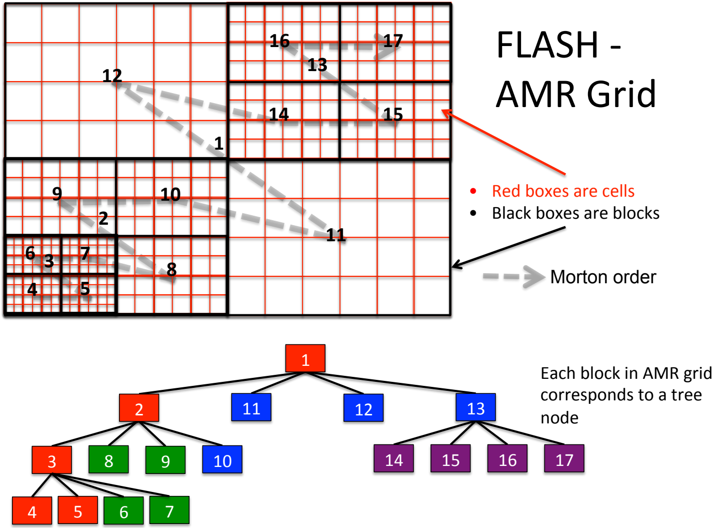


HDF5: Potential for odd interactions between application data layout and chunk allocation



Lower-triangular aware storage mode and layout

Example: FLASH



Example: FLASH

- Parallel adaptive-mesh refinement (AMR) code; Block structured - a block is the unit of computation
- **Tree information:** FLASH uses tree data structure for storing grid blocks and relationships among blocks, including lrefine, which_child, nodetype and gid.
- **Per-block metadata:** FLASH stores the size and coordinates of each block in three different arrays: coord, bsize and bnd_box
- **Solution Data:** Physical variables i.e. located on actual grid are stored in a multi-dimensional (5D) array e.g. UNK

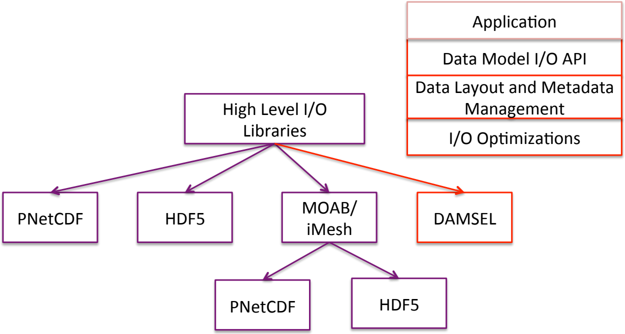
NORTHWESTERN UNIVERSITY

Goals

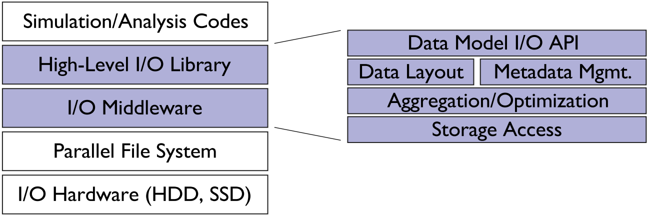
- Provide higher-level data model API to describe more sophisticated data models
- Enable exascale computational science applications to interact conveniently and efficiently with storage through the data model API
- Develop a data model storage library to support these data models, provide efficient storage data layouts
- Productizing Damsel and working with computational scientists to encourage adoption of this library by the scientific community

NORTHWESTERN UNIVERSITY

Big Picture


Application
Data Model I/O API
Data Layout and Metadata Management
I/O Optimizations
High Level I/O Libraries
PNetCDF
HDF5
MOAB/ iMesh
DAMSEL
PNetCDF
HDF5

Proposed Approach



Proposed Approach

- a set of data models I/O APIs relevant to computational science applications
- a data layout component that maps these data models onto storage efficiently,
- a rich metadata representation and management layer that handles both internal metadata and that generated by users and external tools,
- I/O optimizations: adaptive collective I/O, request aggregation, and virtual filing,

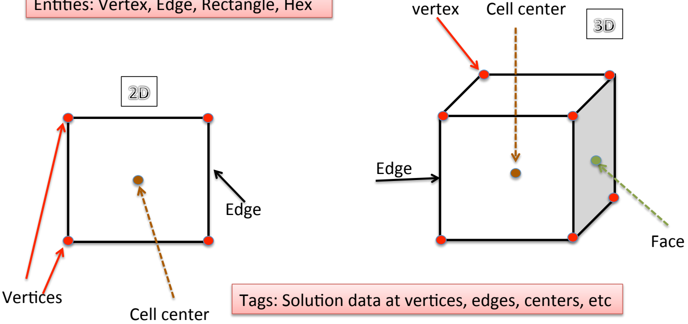
NORTHWESTERN UNIVERSITY

Data Model Components

- Describe structural/(hierarchical) and solution information through API
- To describe the structural information, i.e. Grid data
  Entity, Entity sets, Structured Blocks
- To describe the solution variable, i.e. Solution data
  Tags on Entities, Entity Sets, Structured Blocks

Example: Entity and Tags

Entities: Vertex, Edge, Rectangle, Hex

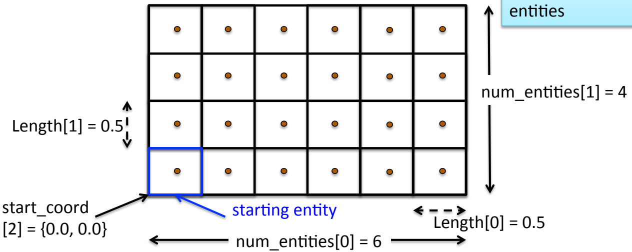


Tags: Solution data at vertices, edges, centers, etc

Example: Blocks and Tags

`Step 1: Creating the first/start entity`

Step 2: Defining start coordinates, lengths, number of entities

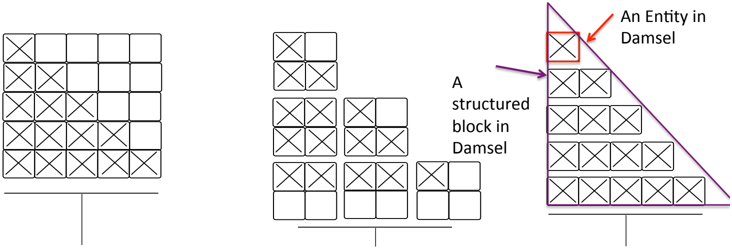


`Step 3: Creating a cartesian mesh/structured block`

`Step 4: Tag the centers of entities in cartesian mesh/structured block`

NORTHWESTERN UNIVERSITY

Example: Lower Triangle Matrix



netCDF: fixed dimensions

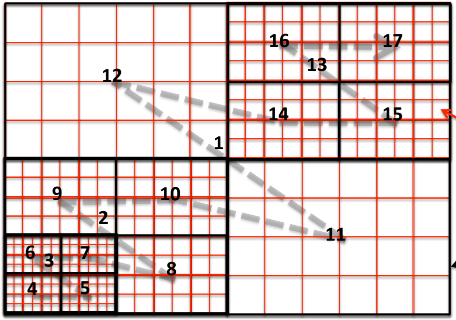
HDF5: Potential for odd interactions between application data layout and chunk allocation

Lower-triangular aware storage mode and layout

3 Usecases

- Usecase I: FLASH
- Usecase II: GCRM

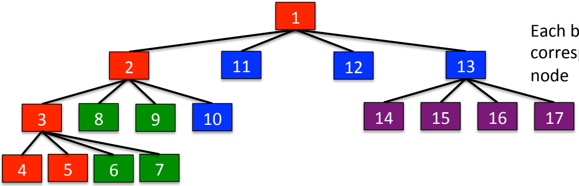
Introduction



FLASH - AMR Grid

- Red boxes are cells
- Black boxes are blocks

Morton order



Each block in AMR grid corresponds to a tree node

Introduction

- The FLASH is a modular, parallel multi-physics simulation code capable of handling general compressible flow problems found in many astrophysical environments.
- Parallel adaptive-mesh refinement (AMR) code; Block structured - a block is the unit of computation
- **Tree information:** FLASH uses tree data structure for storing grid blocks and relationships among blocks, including lrefine, which_child, nodetype and gid.
- **Per-block metadata:** FLASH stores the size and coordinates of each block in three different arrays: coord, bsize and bnd_box
- **Solution Data:** Physical variables i.e. located on actual grid are stored in a multi-dimensional (5D) array e.g. UNK

FLASH using existing I/O Libraries

FLASH in PnetCDF and MOAB

```
/*Step 1: Create data set*/
ncmpi_create_data()

/*Step 2: Define dimension*/
status = ncmpi_def_dim(ncid, "dim_tot_blocks",
(MPI_Offset)(*total_blocks), &dim_tot_blocks);

/*Step 3: Define variables*/
Status = ncmpi_def_var (ncid,
"runtime_parameters", NC_INT, rank, dimids,
&varid[id]);
status = ncmpi_def_var (ncid, "lrefine",
NC_INT, rank, dimids, &varid[id]);

/*Step 4: Create attributes for some
variables*/
status = ncmpi_put_att_int(ncid, 1,
intScalarNames[i], NC_INT, 1, &intScalarValues
[i]);

/*Step 5: Write structural & solution data*/
/* Write data from memory to file */
    err = ncmpi_put_vara_all(fileID, varID,
diskStart, diskCount, pData, memCountScalar,
memType);

/*Step 6: Close the dataset/file*/
ncmpi_close(fileID);
```

```
  moab::Core *mb = new moab::Core();
  moab::ErrorCode rval;
  moab::Range blk_handles;
  moab::Tag unkTH, lrefineTH, scalarsTH;

/*Step 1: Create an Entity Set*/

/*Step 2: Define/set tags for total_blocks,
runtime parameters, etc on the Entity set*/

/*Step 3: Create FLASH blocks as vertices in
MOAB*/
  rval = mb->create_vertices ( block_coords,
total_blocks, blk_handles);
  if (MB_SUCCESS != rval) return 1;

/*Step 4: Define tags for the structural
information per block and solution data*/
rval = mb->tag_create("lrefine", sizeof(int),
MB_TAG_DENSE, lrefineTH, lrefine);
rval = mb->tag_create("unk", 10*(nxb*nyb*nzb)
*sizeof(double), MB_TAG_DENSE, unkTH, unk);

/*Step 5: Set tags for tree & solution data*/
rval = mb->tag_set_data(lrefineTH, blk_handles,
lrefine);
rval = mb->tag_set_data(unkTH, blk_handles,
unk);

/*Step 6: HDF5 File I/O*/
/* Write data from memory to file */
```

FLASH using DAMSEL

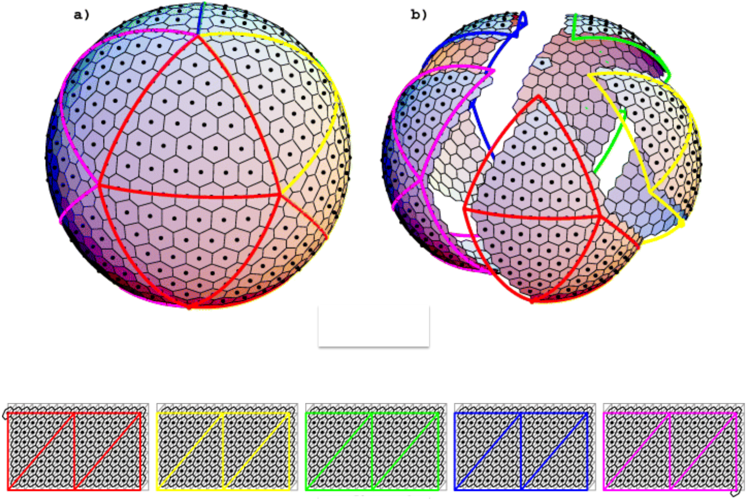
- Goal: to describe hierarchical/structural and solution information through API
- Entity
  - Cells as Rectangles
  - Blocks as Cartesian Mesh
- Entity Sets
  - Blocks assigned to entity sets to define hierarchical/structural information
- Tags
  - Only for solution data

FLASH using proposed DAMSEL API

```
Step 1: Creating the first/start entity
damsel_create_entity();
Step 2: Defining start coordinates, lengths, number
of entities
Step 3: Creating a cartesian mesh/structured block
damsel_cartesianmesh_create()
Step 4: Defining hierarchy using Entity sets
   damsel_create_entityset()
   damsel_addEntities()
   damsel_addChildren(EntityHandle , EntityHandle Children
   [])
Step 5: Define and set tags
   damsel_tag_define()
   damsel_tag_setval()
Step 6: Damsel I/O
```

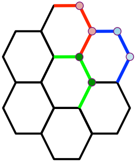

NORTHWESTERN UNIVERSITY

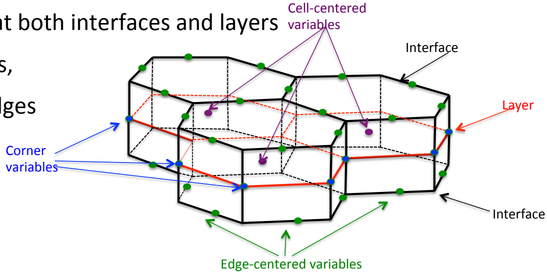
Introduction



Introduction

- Grid data
  - Cell corners (2/cell)
  - Cell edges (3/cell)
  - Layers and interfaces
- Solution data at both interfaces and layers
  - Cell centers,
  - corners, edges





NORTHWESTERN UNIVERSITY

GCRM using existing I/O Libraries

PNetCDF

- Grid Data:
  - Dimensions: Cells, edges, interfaces, etc
  - Variables: grid_center_lat(cells), grid_corner_lat(corners), cell_corners(cells, cellcorners)
- Solution Data:
  - float pressure(time, cells, layers)
  - float u(time, corners, layers)
  - float wind(time, edges, layers)

MOAB

- A Hexagonal Prism entity to describe a cell
- An unstructured mesh to describe GCRM grid (no hierarchical information)

GCRM using DAMSEL

- A Hexagonal Prism entity to describe a cell
- An unstructured mesh to describe GCRM grid (no hierarchical information)
- Or a structured mesh to describe GCRM grid

NORTHWESTERN UNIVERSITY

Summary

- Motivation
- DAMSEL Data Model
- Usecases: FLASH and GCRM
- API Implementation and data layout work is in progress