ECSS Project: Prof. Bodony: CFD, Aeroacoustics

Robert McLay

The Texas Advanced Computing Center

June 19, 2012

ECSS Project: Bodony Aeroacoustics Program

- Program's name is `RocfloCM`
- It is mixture of Fortran 90 and C and C++ code
- The main science part is Fortran 90.
- This is a 3-D Finite Difference/Finite Volume Flow code.
- It uses a structured 3-D Cube as the computational grid.
- He and his team have been long-time TACC users.

ECSS Project

- The work (and this talk) divides into two parts.
  - Measuring Performance w/TAU, Improving MPI Communications.
  - Replacing per task I/O with parallel HDF5
- Conclusions

Performance Improvements

- Use TAU, the performance measuring tool.
- It provides per routine performance via instrumentation.
- Using TAU 1st pass the number one routine was `MPI_Barrier`.
- Removed several calls to `MPI_Barrier` from critical path.

Second Round Results from TAU

- The program solves the equations of state in each coordinate direction separately. The problem child routine did:
  - Post Receive Buffer for "left", Send to "right".
  - Wait for completion
  - Post Receive Buffer for "right", Send to "left".
  - Wait for completion
- I changed this to:
  - Post Receive Buffer for Both Neighbors.
  - Copy data to both Send buffers.
  - Send Data to both Neighbors.
  - Wait for completion.
- Re-running program showed a 27% improvement in performance.

`RocfloCM` Program I/O

- Originally used a per task output file for each time step.
- Any analysis required merging the task separate output files into global files.
- Parallel HDF5 provides an efficient way to write one global file per timestep.

HDF5 Overview

- An HDF5 file can be viewed as a file system inside a file.
- It uses a Unix style directory structure.
- It is a mixture of groups, datasets and attributes.
- This way each dataset can have a description (e.g. physical units)
- Typically one writes files in native machine format.
- It will automatically translate if required.
- Anybody run on a Big-Endian machine lately?

HDF5 Benefits

- It is easy to record many metadata items within a solution file.
- Adding attributes later won't break any program that reads the data.
- I am very big on testing and Q.A. With HDF5 it is easy to save with each solution file:
  - Computer Name, OS Version.
  - Compiler and MPI name and version.
  - Program Version.
  - Input file.
  - ...

Parallel HDF5 Benefits

- All tasks are writing their data at once.
- No sending all data to Proc 0.
- Users can control number of writers and stripes on output.
- Users can control number of readers on input.

Learning HDF5

- I learned HDF5 as part of this project.
- I attended the Parallel File System Tutorial at SC11.
- Almost of all of the HDF5 documentation is written for designers and not for users.
- The only bright spot is the HDF tutorial web pages.

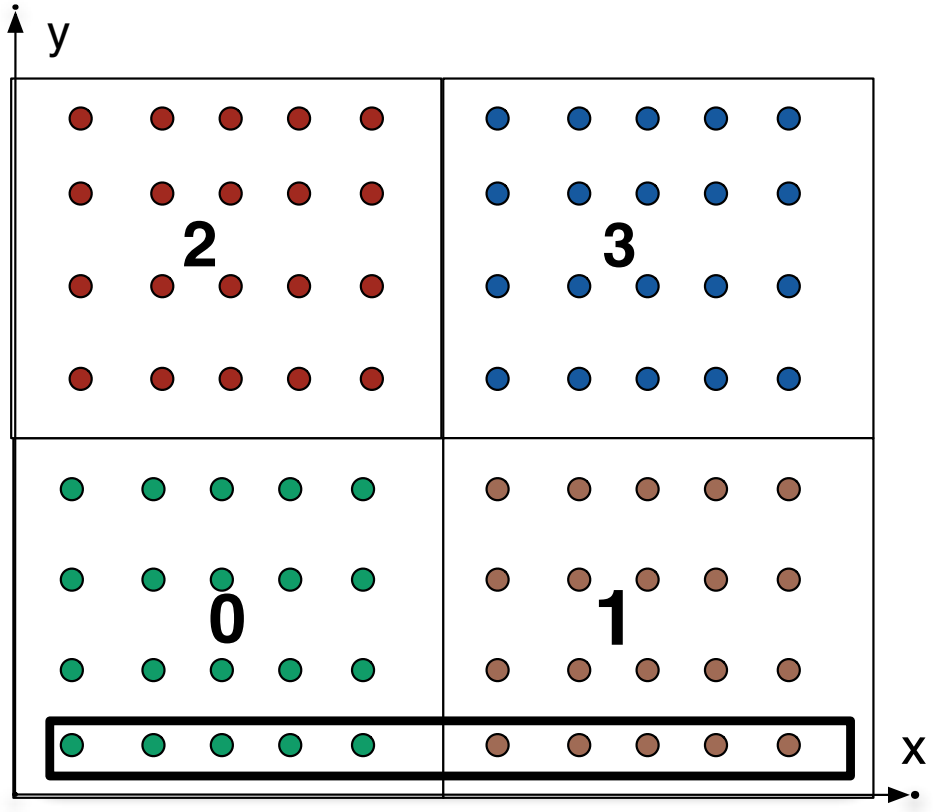
Lustre Parallel File Systems

- Parallel File Systems are magical.
- They make a collection of computers and disks look like 1 disk.
- Parallel File Systems are easy to abuse, break, slow down ...
- To obtain good performance one must chose the number of writers and stripes (at least with Lustre).
- They are a shared resource!!!

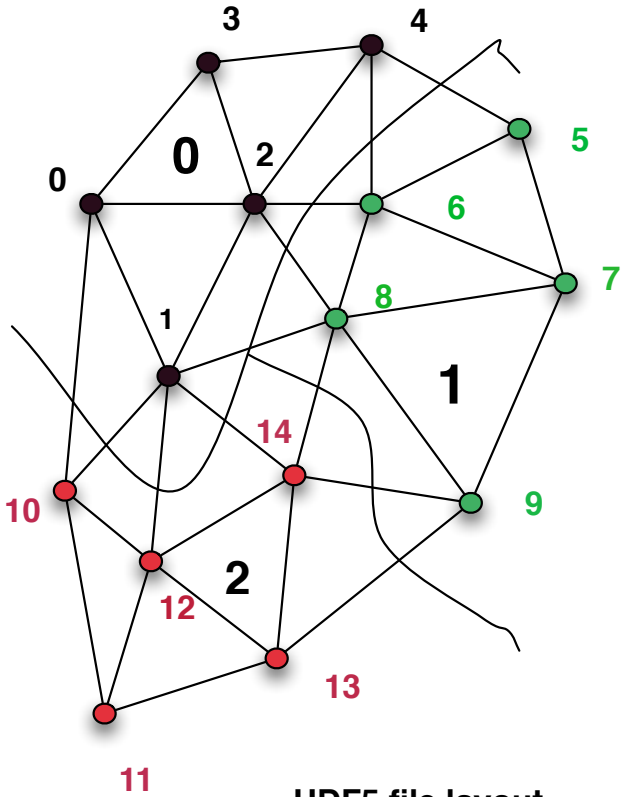
Parallel Data Layout Strategies

- Many large PDE codes use one of two methods of partitioning the problem:
  - A structured grid (2-D rectangle or 3-D shoebox) or a union of these.
  - A unstructured grid.

Structured Grid



Unstructured Grid



HDF5 file layout

0, 1, 2, 3, 4,	5, 6, 7, 8, 9,	10, 11, 12, 13, 14

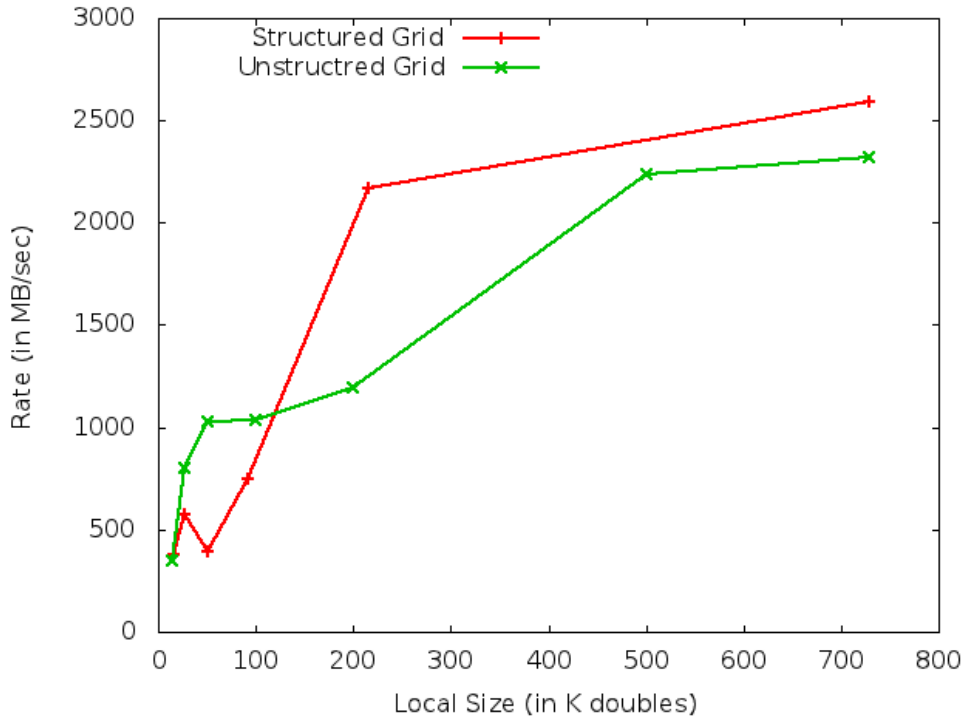
Parallel HDF5 Issues

- HDF5 uses MPI I/O (ROMIO) to handle the parallel parts.
- One uses `MPI_Info` to set the number of writers and stripes.
- There are two ways to define local portions of a global grid:
  - Hyperslabs: Original way.
  - Chunks: New Way.
- I tested both and always found Hyperslabs to work better.
- I would be interested to know if/when Chunks are better.

Two Test Codes

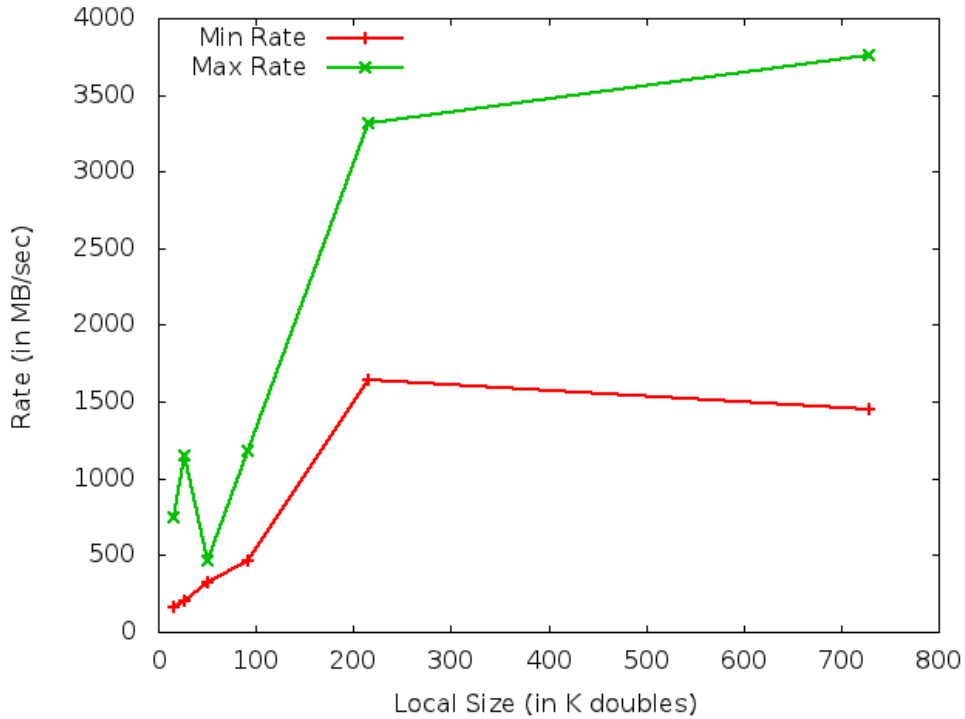
- `MPIIO`:
  - A fortran 90 code that uses Hyperslabs and Chunks.
  - 2-D rectangles or 3-D cubes.
  - This was delivered to customer.
- `H5test`:
  - A C++ code that uses Hyperslabs and Chunks.
  - Simulates an unstructured grid.
- They are designed to measure performance over grid size, writers and stripes

Averaged Structured vs Unstructured Grid



These results are the average of 10 runs for each size on the same nodes.

Structure Grid Min Max Results



Shows the fastest and slowest runs for the structured grid from before.

Lessons (Re-)Learned

- Our users are way more interested in science and not in performance.
- Less use of `MPI_Barrier` can improve performance.
- Better overlapping of messages can improve performance.
- Parallel HDF5 is ready for Prime-Time!

How many Writers and Stripes?

- A shared system means that using max stripes is *NOT* best.
- My rules for chosing writers and stripes:
  - Use 2 stripes per writer ($N_{stripes}$).
  - Use no more than 2 writers per node.
  - Use no more than 66% of max stripes available.
- Remember $OST_{active} \equiv T_{stripes} = N_{stripes} * N_{writers}$, where $N_{stripes} = 1$, 2 or 4.

How many Computational Nodes?

- MPI will tell me how many tasks.
- A node has a unique hostname.
- `MPI_Allgather` hostnames from each task.
- Count number of unique hostnames.

Max stripes available?

- This is a brute force way of finding out.
- I would like a better way:
  - Task 0 creates a file with the maximum number of stripes.
  - Count the number of stripes with a popen to the output of "`lfs getstripe -q`"
  - delete file and Bcast number to all tasks.