

Overview:

On August 21st, Eric Brugger, David Camp, Hank Childs, Cyrus Harrison, and Brad Whitlock met at LLNL and discussed issues with adding hybrid parallel support to VisIt. This document summarizes the discussion, and outlines a direction for initial hybrid parallel support. This document is a living document and feedback for missing perspectives, improved planning, etc. is appreciated.

Tasks:

- 1) Infrastructure:
 - a. ThreadPool class that isolates details of pthreads, Windows threads, OpenMP (?), etc from developers.
 - b. Support in host profiles
 - i. Number of threads to create (may be more than # cores)
 - ii. Flag for tasks per NUMA section
 - iii. Thread depth
 - c. -debug and -timing
 - d. efficient load balancing
 - i. split single large vtkDataSet into chunk that threads can operate on. May be explicit split into multiple pieces or implicit split where each thread is assigned a piece (i.e. this thread to work on cells 1000-2000).
- 2) Filters
 - a. Embarrassingly parallel algorithms
 - i. Add core infrastructure to base classes for splitting an avtDataTree over threads
 - ii. Audit filters and ensure thread safe (i.e. data members aren't modified in ExecuteData)
 - b. Non-embarrassingly parallel algorithms
 - i. Streamlines
 - ii. Volume rendering
 - iii. Audit and fix other non-embarrassingly parallel algorithms
 1. Pos-CMFE
 2. Ghost zone generation
 3. Line scans
 4. ???
- 3) Threaded I/O requests
 - a. Implementation at avtGenericDatabase level
 - b. We identified that the appropriate threading for I/O may be different than the threading for filters. This has implications for host profiles.
- 4) Threaded rendering

Plan:

An initial phase will take place in the next few months. Camp and Childs will be carrying out the majority of the work & Whitlock and Harrison have volunteered to help as well. The initial phase will include 1a, 1b, 1c, 2a, 2bi, 2bii.

Subsequent phases (1d, 2biii, 3, 4, and other subsequently identified tasks) will take place after the initial phase is committed to the repo.

Discussion:

Additional discussion covered the following items:

- 1) We are going to ignore the issue of pinning data to threads on certain cores initially. David believes that we should not worry about NUMA effects in the first phase of the threading work. He thinks if the data is divide correctly we should not suffer from the NUMA effects. Also if we find issues, there are ways to create tasks that run per NUMA node and only use the threads on the NUMA section.
He believes we can have a bigger win by working on good load balancing, on the threads, to offset any hit by the NUMA effect. Good load balancing can improve our cache hits and provide better performance improvements to VisIt.
- 2) Brad suggested a new "ParallelContext" object that allows each thread to know its ID and may have future benefits with respect to multiple simultaneous pipeline executions, distinct MPI communicators, etc. Brad fleshed this out subsequent to our meeting and posted the discussion to [visitusers.org](http://www.visitusers.org). You can see the description here:
http://www.visitusers.org/index.php?title=Parallel_ideas
- 3) We decided to instantiate our threads one time per program, rather than instantiating and deleting with each filter execution, as the overhead for each thread create/destroy is of concern