PHASTA File-less Integration with Mesh Adaptation

Tuesday, May 03, 2011

Cameron Smith, Brian Orecchio

The PHASTA software suite adaptively simulates complex fluid dynamic problems on a massively parallel scale \cite{Sahni2009}. It has two main components, the flow solver, implemented in phSolver, and the error estimator and mesh adaptation driver, implemented in phParAdapt. phSolver solves a system of equations over a discretization of the computational domain, the mesh, with the finite element method. Each phSolver process solves a subset of these equations on the mesh subdomain that is assigned to it during a pre-processing stage. After simulating the flow for a certain number of time steps, files containing solution data are output. phParAdapt reads these files, computes the mesh metric field based on the solution field, and adapts the current mesh to satisfy the metric field. After adaptation, phParAdapt writes files for the solver to continue the simulation. In massively parallel simulations requiring frequent adaptation, the reading and writing of files by phSolver and phParAdapt introduces significant computational overheads largely due to the latency of writing/reading to disk. Thus, enabling phSolver and phParAdapt interfacing without writing and reading files for adaptation would greatly increase the performance of the application. An implementation is discussed that integrates phParAdapt and phSolver via functional interfaces for execution control, data passing and data transformation.

phSolver and phParAdapt are required to provide functional interfaces to support initialization of the interfaces, flow solve and adaptation procedures, respectively, data transformation and exchange, and finalization of the interfaces. The initialization procedures encapsulate existing mechanisms for processing input parameters as specified by the user and allocation of requisite data structures to maintain the values of these parameters. The finalization procedures deallocate these structures. These initialization and finalization procedures provide a starting point for API users such that the state of the interfaces is guaranteed before any other procedures are invoked \cite{Miller2004}.

The integrated executable is driven by a function, described by the pseudo-code in Figure 1, that repeatedly executes three procedures; flow solve, adaptation, and pre-processing. Before iteration over these procedures phSolver and phParAdapt are initialized on lines four and five with calls to the phSolver() and phParAdapt() constructors. Execution of the flow solver requires the phSolver mesh and solution files that are read on line seven with the call to phS->readMeshAndSolution_fromFiles(…). It is assumed in the workflow that an initial set of phSolver mesh and solution files are prepared prior to execution. The time stepping loop begins on line 14 and ends on line 34. phS->solve(…) is called on line 21 for solving the flow for a certain number of time steps. Here the loop is constructed with the

assumption that the flow solver has the ability to stop when either all time steps have been executed, adaptation is required, or there has been an error. Next, mesh adaptation is executed on line 25 with the call to phA->adapt() using the mesh and flow solver solution fields produced from the time-stepping loop. Local solution transfer methods are applied at each mesh modification operation during mesh adaptation. The adapted mesh and solution fields are then transformed into the form required by the flow solver, phA->preProcess(), and written into the flow solver data structures, phS->updateMeshAndSolution (…). Execution then returns to the top of the main loop and the solver is ready to run time-steps again on the adapted mesh.

```
1  main ()
2  {
3    //Initialize phSolver and phParAdapt
4    phSolver* phS = new phSolver(<configuration file name>)
5    phParAdapt* phA = new phParAdapt(<configuration file name>)
6
7    phS.readMeshAndSolution_fromFiles(<LIST OF FILE NAMES e.g. geombc.*.dat, restart.*.*> )
8    phA.readMeshDatabase_fromFiles(<mesh database file e.g. FMDB or Simmetrix Mesh>)
9
10   int currentTimeStep = 0
11   int N //total timesteps to run
12
13   //Time Stepping - Adaptation Loop
14   while( currentTimeStep < N )
15   {
16     //Solve the remaining timesteps
17     //phS.solve(...) will exit if
18     //1) done with timesteps
19     //2) adaptation is needed due to bad mesh
20     //3) Error
21     currentTimeStep = phS->solve(N-currentTimeStep)
22     if( currentTimeStep < N && Adaptation Required)
23     {
24       phA->loadAndAttachSolution(phS.getSolverFields())
25       phA->adapt()
26       phA->preProcess()
27       phS->updateMeshAndSolution(phA.getSolverFields())
28     }
29     else if ( Error )
30     {
31        exitWithError()
32     }
33   }
34 }
```

Figure 1. Pseudo code for file-less solve-adapt loop.

In the last period SCOREC researchers have refactored the single entry point phParAdapt executable into a C++ class. Towards supporting file-less coupling with phSolver the monolithic adaptation function was separated into one function which supports loading phSolver solution, and solution-derived, data and attaching it to the mesh entities of the mesh database, loadAndAttachSolution(…), and another function which computes the mesh metric field and drives mesh adaptation procedures, adapt(…). The pre-processing and the file based mesh database loading member functions were extracted from source code with minimal modifications.

The phParAdapt API supports the file dependent mode supported by the original single entry phParAdapt executable. During the next period the extension of the functions loadAndAttachSolution(..) and preprocess(…) to respectively support the reading and writing of phSolver formatted data structures will enable phParAdapt API support for the pseudo code listed in Figure 1. A performance study of the file based and file-less API modes for a test case that pre-processes phSolver data then performs mesh adaptation will be performed.

During the last period SCOREC researchers investigated the implementation of the current single entry point phSolver executable and initial API designs have been proposed. Refactoring of phSolver for implementation of a functional interface will begin in the next period.

References:

@inproceedings{Sahni2009,

author = {Sahni, Onkar and Zhou, Min},

booktitle = {SC '09 Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis},

doi = {1654059.1654129},

title = {{Scalable implicit finite element solver for massively parallel processing with demonstration to 160K cores}},

year = {2009}

}

@inproceedings{Miller2004,

author = {Miller, MC and Reus, JF and Matzke, RP and Koziol, QA and Cheng, AP},

booktitle = {Proceedings of the Nuclear Explosives Code Developer's Conference 2004, vol. 1, N/A, December 15, 2004, unknown},

publisher = {Lawrence Livermore National Laboratory (LLNL), Livermore, CA},

title = {{Smart Libraries: Best SQE Practices for Libraries with an Emphasis on Scientific Computing}},

url={http://scholar.google.com/scholar?hl=en\&btnG=Search\&q=intitle:Smart+Libraries+:+Best+SQE+Practices+for+Libraries+with+Emphasis+on+Scientific+Computing\#0},

year = {2004}

}