File-less Integration of PHASTA Solver and Mesh Adaptation: Design Proposal

Cameron Smith, Brian Orecchio, Min Zhou

December, 8 2010

The PHASTA software suite adaptively simulates complex fluid dynamic problems on a massively parallel scale [1]. It has two main components, the solver, implemented in phSolver, and the adapter, implemented in phParAdapt. phSolver solves a system of equations over a descretization of the computational domain, the mesh, with the finite element method. Each phSolver process solves a subset of these equations on 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 to adapt the current mesh based on user specified adaptation method controls. 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 funtional 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, 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 [2].

The integrated executable is driven by a function, described by pseudocode below, that repeatedly executes three procedures; flow solve, adaptation, and pre-processing. Before iteration over these procedures phSolver and phParAdapt are initialized on lines seven and eight with calls to phSolver_init() and phParAdapt_init(). Execution of the flow solver requires the phSolver mesh and solution files that are read on line 10 with the call to phSolver_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 16 and ends on line 38. phSolver_solve() is called on line 23 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 28 with the call to phParAdapt_adapt() using the mesh and flow solver solution fields produced from the time-stepping loop. After adaptation, the solution field projected on the adapted mesh, and the adapted mesh, is transformed into the form required by the flow solver and written into the flow solver data structures. This procedure is completed by the functions, phParAdapt_preprocess(), and a combination of the phParAdapt data accessor APIs and the phSolver data modifier APIs. Now the execution returns to the top of the main loop and the solver is ready to run time-steps once again on the adapted mesh.

An alternative approach, which is currently implemented, but not maintained since its initial development, differs from the above approach in the location of mesh pre-processing and adaptation in relation to the flow solver's time-stepping loop. In this alternative approach, they are embedded in the flow solver's time-stepping loop, implemented in phSolver itrdrv.f, and thus exploits access to global variables exposed at this level, which include the flow solver mesh and solution data structures. The proposed approach has the preprocessing and adaptation outside of the flow solver's time-stepping loop and uses protected data structures for the solver mesh and solution. These structures are passed into the various functions through pointers and have API's to obtain and modify the solver data. This approach will enable easier code maintainability while preserving performance. However, this approach will take more time to implement than the first approach due the refactoring of phSolver and phParAdapt.

```
main ()
{
   i = 1 // current timestep
   N //total timesteps to run

   //Initialize phSolver and phParAdapt
   phSolver_instance phS = phSolver_init()
   phParAdapt_instance phA = phParAdapt_init()

   phSolver_readMeshAndSolution_fromFiles(phS, <LIST OF FILE NAMES i.e.
geombc.*.dat, restart.*.*)
   phParAdpat_readMeshDatabase_fromFiles(phA, <meshDB file name i.e. FMDB or
Simmetrix mesh)

   //Time Stepping - Adaptation Loop
   While( i < N )
   {
      //Solve the remaining timesteps
      //phSolver_solve will exit if
      //1) done with timesteps
      //2) adaptation is needed due to bad mesh
      //3) Error
      phSolver_solve(phS, N - i )
      if( i < N )
      {
         if(Adapt Needed )
         {
            phParAdapt_Adapt(phA, getSolutionFields(phS))
            phParAdapt_preProcess(phA)
            if( OK )
            {
               // update phSolver mesh and solution data structures using a
               //  combination of the phParAdapt data accessor APIs and the
               //  phSolver data modifier APIs
            }
         }
      }
   }
   phSolver_finalize(phS)
   phParAdapt_finalize(phA)
}
```

References

[1] Onkar Sahni and Min Zhou. Scalable implicit finite element solver for massively parallel processing with demonstration to 160K cores. In *SC '09 Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009.

[2] MC Miller, JF Reus, RP Matzke, QA Koziol, and AP Cheng. Smart Libraries: Best SQE Practices for Libraries with an Emphasis on Scientific Computing. In *Proceedings of the Nuclear Explosives Code Developer's Conference 2004, vol. 1, N/A, December 15, 2004, unknown*. Lawrence Livermore National Laboratory (LLNL), Livermore, CA, 2004.