Performance Characteristics of Curvilinear Multi-Block Structured ALEGRA on Red Storm

David Hensinger, *Sandia National Laboratories; and*
Robert Mackinnon, *Sandia National Laboratories*

ABSTRACT: *The hydrodynamics/solid dynamics code ALEGRA.was used to run a test problem on up to 25 processors of the Red Storm and Janus machines. Timing results were collected from these runs. These results will be compared and discussed in the context of the test problem. The Red Storm machine ran all simulations more than 8 times faster than Janus.*

KEYWORDS: Red Storm, Janus, timing results

1. Introduction

The ALEGRA shock-physics / multi-physics code uses an arbitrary Lagrangian-Eulerian (ALE) formulation on a finite element mesh. This formulation allows the user to determine whether material should flow through a stationary mesh (pure Eulerian), whether the mesh should move with the material (pure Lagrangian), or whether the mesh should move independently from the material motion (Arbitrary). ALEGRA supports parallel execution through message passing using MPI.

The curvilinear multi-block structured capability of ALEGRA has significant storage and cycle time benefits over the unstructured formulation. It also supports inline specification of solution domains which allows specification of problems without reliance on input mesh files.

2. ALEGRA's Communication Pattern

ALEGRA timesteps proceed in two sequential sub-steps. A Lagrangian sub-step in which the mesh follows material movement, and a remesh/remap sub-step in which nodes are arbitrarily moved and element and nodal quantities are advected through the mesh in correspondence with this mesh movement. A pure Eulerian timestep results when the nodes are moved back to their original coordinates during the remesh sub-step.

During the Lagrangian sub-step, the curvilinear multi-block version of ALEGRA uses swap and add operations of nodal quantities on block boundary nodes. These swap and add operations are applied across block boundaries whether the blocks are on or off processor. During the Lagrangian sub-step, vectors are swapped and added twice for each block boundary node.

During the Eulerian sub-step ALEGRA uses ghost elements as locations into which to sum and from which to accrue fluxes. Four times per each Eulerian sub-step element quantites are updated on ghost elements.

In addition to these communications there are a collection of global sum, global maximum, and global minimum calls made throughout each timestep.

3. The Test Problem

The test problem (Input deck attached as Appendix A) used for gathering this timing comparison consisted of a pure Eulerian cube of discretized space filled with material that was uniformly advecting through the mesh in the +x,+y,+z direction. This problem ensured that every element would advect material in each coordinate direction during every timestep. This maximizes the amount of work performed during the Eulerian sub-step of ALEGRA. The work done during a Lagrangian sub-step of ALEGRA depends almost exclusively on domain size.

The problem domain was specified within the ALEGRA input deck in terms of its spatial extent and the numbers of elements and blocks in each coordinate direction. The total number of elements in a domain is the product of the number of elements in each coordinate direction in each block multiplied by the product of the number of blocks in each coordinate direction.

Curvilinear multi-block structured ALEGRA is capable of running with zero or more blocks of structured mesh on each processor. By default, blocks of mesh are dealt out to available processors in numerical order. For the purposes of this comparison the number of blocks was always set equal to the number of processors. This eliminates on-processor communication between blocks.

4. Running on Red Storm

Successful efforts to run on the Red Storm machine were limited by the availability of the machine. One preliminary (serial) run was carried out on Red Storm using some of the utility software undergoing testing (xtshowmesh etc...). After this preliminary run, there were no further opportunities to use large portions of Red Storm.

The timing runs reported here were performned on a 28 node cluster that was being adminstrated separately from the bulk of the Red Storm machine. Only serial and 25 processor jobs were run. Job submission was done interactively using a yod -sz command. Jobs launched very quickly and there appeared to be a well furnished suite of unix utilities available on the interactive node from which the jobs were launched.

5. Running on Janus

The runs performed on Janus were submitted to interactive nodes using a 'yod' command almost identical to that used on Red Storm. Since no output files were generated and only a single input file was used, the input/output capabilities of Janus did not come into play. Janus placed a relatively tight limit on the number of elements that could be loaded on each processor. Curvilinear Multi-Block ALEGRA requires a minimum of 820 bytes of memory per element [1.2 million elements / Gb].

6. Results

The data collected from these runs consisted of the time spent in advancing the solution for a total of 10 cycles. This metric excludes startup time and shutdown time. The timing results are summarized below in Table 1. All times are given in micro-seconds. The duration of these runs was fairly short, and successive tests showed that the timing results were consistent to two significant figures.

\

Problem Size	Janus	Red Storm	Janus/Red Storm
Serial 125,000 Elements	500	61	8.2
25 processors 125,000 elements 5,000 / processor	540	57	9.5
25 processors 1000,000 elements 40,000 / processor	520	61	8.5
25 processors 3,125,000 elements 125,000 / processor	520	62	8.3

Table 1. Cycle time $x10^{-6}$s/element/processor

For the 125,000 total element problems, on Janus ALEGRA performance scaled less than linearly when the number of processors was increased from 1 to 25 (540 vs. 500 an 8% departure). Red Storm had a suprising super-linear performance on 125,000 elements on 1 vs 25 elements (57 vs. 61 ~6% speedup). This may be attributed to the signifncantly larger cache memory available on the Red Storm processors and the relatively small number (5,000) of elements per processor during the 25 processor runs.

For larger problems Janus apparently became more efficient on a per element basis . This could be explained by the relative decrease in the communication burden as the element blocks' surface to volume ration decreased. Red Storm cycle times per element increased slightly. This may have been due to loss of cache efficiencies.

Conclusion

A hydrodynamics advection problem that required constant work per element was used to test and compare the speed of ALEGRA running on Janus and Red Storm. As the number of processors was increased, performance on Red Storm scaled more closely to linear than performance on Janus. For all simulations Red Storm was more than eight times faster than Janus.

Acknowledgments

Special thanks to Suzanne Kelly for granting excellent support to neophyte users.

About the Authors

David Hensinger dmhensi@sandia.gov and Robert Mackinnon rjmacki@sandia.gov are staff members at Sandia Nation Laboratories in the Computational Physics Research and Development group. P.O. Box 5800, Albuquerque NM, 87185-0378

Appendix A: Typical Input Deck, 125,000 Elements 1 Block

```
TITLE
Two bar impact ,1 material, all IG

termination cycle 10

structured solid dynamics
  structured mesh
    amr 3dr
      nx =  50
      ny =  50
      nz =  50
      bx =  1
      by =  1
      bz =  1
      gmin = 0.0 0.0  0.0
      gmax = 10.0 10.0 10.0
    end
  end

  initial block velocity, block 1,vector,
x=1.0e+05 y=1.0e+05 z=1.0e+05

  block 1
    eulerian mesh
    material 1
    remesh frequency 1
  end

  domain
    new smyra interface tracker
  end
end

material 1
  model 1
end

model 1 KEOS IDEAL GAS
  GM1 0.4
  CV 1.723459e+07
END
EXIT
```