SANDIA REPORT

SAND2015-11038
Unlimited Release
Printed December 18, 2015

Parallel scaling analysis for explicit solid dynamics in ALEGRA

John H. J. Niederhaus, Richard R. Drake, and Christopher B. Luchini

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2015-11038
Unlimited Release
Printed December 18, 2015

Parallel scaling analysis for explicit solid dynamics in ALEGRA

John H. J. Niederhaus[1], Richard R. Drake[2], Christopher B. Luchini[3]

[1] Multiphysics Applications
[2] Computational Multiphysics
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185
jhniede@sandia.gov

[3] Sci Tac Research Associates, LLC
Los Alamos, NM
luchini@strallc.com

Abstract

Weak scaling studies were performed for the explicit solid dynamics component of the ALEGRA code on two Cray supercomputer platforms during the period 2012-2015, involving a production-oriented hypervelocity impact problem. Results from these studies are presented, with analysis of the performance, scaling, and throughput of the code on these machines. The analysis demonstrates logarithmic scaling of the average CPU time per cycle up to core counts on the order of 10,000. At higher core counts, variable performance is observed, with significant upward excursions in compute time from the logarithmic trend. However, for core counts less than 10,000, the results show a 3× improvement in simulation throughput, and a 2× improvement in logarithmic scaling. This improvement is linked to improved memory performance on the Cray platforms, and to significant improvements made over this period to the data layout used by ALEGRA.

Acknowledgment

Acknowledgment

Compute time on Cielo was provided by DOE's ASC Cielo Capability Computing Campaign (CCC-5) under the administration of Joel Stevenson (SNL-9326). Compute time on Excalibur was provided by DoD's ARL DSRC during early-access testing time and subsequent dedicated time, administered by Thomas Kendall and Phillip Matthews (ARL). Early versions of the ALEGRA simulations and test configuration were developed with the help of David Hensinger (SNL-1555). Special thanks to Richard Barrett (SNL-5638) and Robert Doney (ARL) for reviewing this report.

This work was done under the support of the U.S. Army Research Laboratory.

Contents

Acknowledgment 4

1 Introduction **9**

2 Supercomputing platforms for testing **11**

3 ALEGRA hypervelocity impact simulation **13**

4 Weak scaling concept and scaling analysis configuration **15**

Mesh refinement 15

Performance metrics 16

Cores per node 16

5 ALEGRA parallel performance **17**

Cielo test results, 2012 17

Excalibur test results, 2015 18

Comparison of multiple platforms 20

Alternate representation 22

Discussion of excursions 22

6 Conclusions **25**

References **26**

Appendix

A Run Time Data **29**

List of Figures

1.1 Oblique impact ("Whipple") simulation used here for ALEGRA parallel performance testing. 9

5.1 Test results for Cielo (2012). 17

5.2 Test results for Excalibur (2015) for 32, 24, and 16 cores per node, including four repeated identical tests at 12,288 cores. 18

5.3 Test results for both platforms compared directly. 19

5.4 Simulation time of two hydrodynamics benchmark problems run with a version of ALEGRA as it existed at dates from January 2013 to July 2014. The times of each simulation are shown with an "X" and the solid lines run through the minimum time for each date. 21

5.5 Simulation time of the hypervelocity impact problem run on a single compute node of Cielo using the 2012 release of ALEGRA, the 2014 release, and on a single compute node of Excalibur using the 2014 release. Ten runs of each were made and are marked with an "X". 21

5.6 Scaling trends for ALEGRA on Excalibur (2015), using the grind-time representation, for the tests with 32, 24, and 16 cores per node. 22

List of Tables

2.1 Characteristics of computing platforms used in ALEGRA performance testing. . . . 11

A.1 Selected run-time statistics from Raptor in 2011 [6] with 16 cores per node. 29

A.2 Run-time statistics from Cielo in 2012 with 16 cores per node. 29

A.3 Run-time statistics from Excalibur in 2015 with 16 cores per node. 30

A.4 Run-time statistics from Excalibur in 2015 with 24 cores per node. 30

A.5 Run-time statistics from Excalibur in 2015 with 32 cores per node. 31

Chapter 1

Introduction

ALEGRA is a finite-element multiphysics simulation tool designed for high performance computing using the massively parallel, MPI-based model that has been prevalent since the 1990's. Written primarily in C++ and Fortran, it is widely used for modeling shock hydrodynamics phenomena including terminal ballistics and material failure as shown in Figure 1.1 [17, 7], as well as systems involving resistive magnetohydrodynamics, particularly in electromagnetic pulsed power [9, 10, 15].

Because of the physical and mathematical complexity of these problems, and the large numbers of unknowns typically involved (millions to tens of millions on the average), ALEGRA relies heavily on parallelism to provide reasonable solution times at the users' required level of accuracy. Therefore, significant technical effort has been devoted throughout its lifetime to ensuring that ALEGRA can run successfully at the highest available levels of parallelism, and exploit the parallelism effectively.

Tests of the code's parallel scaling and serial-parallel consistency are built in to the software development environment and executed regularly at very modest degrees of parallelism (8 or 16 processors). However, typical uses cases involve much greater numbers of parallel processors (hundreds to thousands on the average), which are not available for regular testing. Therefore, it is necessary to carry out test campaigns examining parallel scaling in these environments, and in environments anticipated for future use cases.

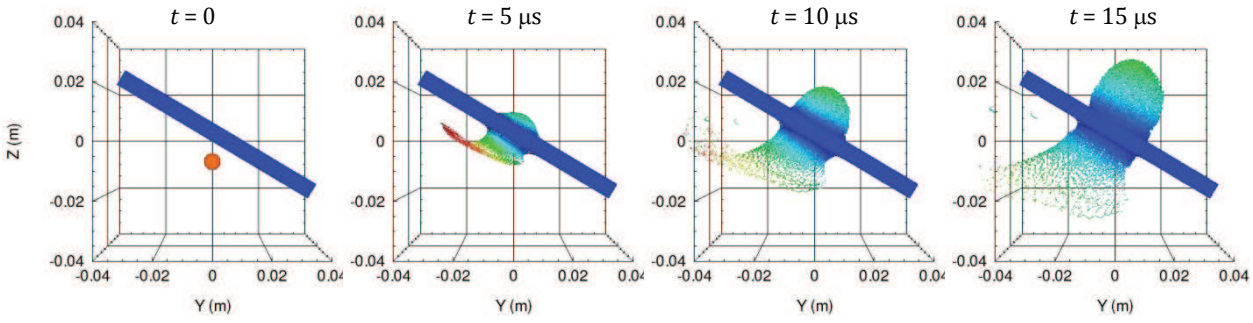

Figure 1.1: Oblique impact ("Whipple") simulation used here for ALEGRA parallel performance testing.

This was done for ALEGRA and the Eulerian shock hydrodynamics code CTH in 2011, demonstrating scaling out to about 10,000 cores on the Cray XE6 system Raptor [6]. The 2011 study is extended here to other Cray systems in order to provide a current performance snapshot as well as an updated test configuration and comparison to previous performance. In this report, we provide a description of the machines subject to testing in Section 2, followed by information on the particular solid dynamics problem used for testing here in Section 3. This is the same as one of the two problems used in the 2011 study, shown in Figure 1.1. We provide a brief review of the concept of "scaling" and in particular "weak scaling" in Section 4, describing the configuration of the tests used here. We then examine and discuss the performance test results in Section 5 and provide final thoughts in Section 6.

Chapter 2

Supercomputing platforms for testing

Two machines were tested in this study. The first series of tests was conducted in December of 2012 on Cielo, a Cray XE6 machine housed at Los Alamos National Laboratory (LANL) [1]. The second series was conducted in April and July of 2015 on Excalibur, a Cray XC40 housed at the U.S. Army Research Laboratory (ARL) [2]. Both machines had been in operation for less than 18 months at the time of the testing. The properties of the two platforms are summarized here in Table 2.1.

Table 2.1: Characteristics of computing platforms used in ALEGRA performance testing.

	Cielo	*Excalibur*
System	Cray XE6	Cray XC40
Online date	November, 2011	April, 2015
Core type	AMD Opteron	Intel Xeon E5
Core speed	2.4 GHz	2.3 GHz
Cores per node	16	32
Memory type	DDR3-1333 MHz	DDR4-2133 MHz
Memory per node	32 GB	128 GB
Total compute cores	143,104	99,136
Interconnect type	Gemini 3D Torus	Aries/Dragonfly

The Cielo Cray XE6 system is owned by the Department of Energy (DOE). Design, procurement and deployment were accomplished by the New Mexico Alliance for Computing at Extreme Scale (ACES), a joint partnership between Los Alamos and Sandia. It is built on AMD Opteron chip technology and the Gemini interconnect, which uses a 3D torus topology. Each compute node has 16 cores, for a total of approximately 140,000 compute cores. The machine came online in 2011 and is still in use currently within DOE.

The Excalibur Cray XC40 system is owned and developed by the Department of Defense High Performance Computing Modernization Program (DoD HPCMP), and is operated by the ARL Dedicated Shared Resource Center (DSRC). It is built on the Intel Xeon E5 chip technology with the Aries Dragonfly interconnect. The compute nodes have 32 cores each, and there are approximately 100,000 total compute cores available. The machine came online in April, 2015 and is currently in use under the DoD HPCMP.

To contrast the hardware, note that the CPU clock frequencies are very similar between the

two, but the memory speed and type is significantly improved. Also, the Opteron has the ability to do at most two vector operations simultaneously, while the Xeon has an 8-wide vector unit. As for the interconnect, improvements have resulted in lower latency communications and higher bandwidth (some details are in Reference [4]). Finally, while Cielo has more total cores, the total problem size that can fit on Excalibur is more than a third larger than can fit on Cielo.

A comparison of the two machines only became possible by coincidence, because of dedicated time allocations and the availability of the parallel scaling test configuration on both machines. The 2012 testing on Cielo used a version of ALEGRA that was current as of June 20, 2012, and was built using the Portland Group (PGI) compiler. The 2015 testing on Cielo used the Release_18Jun2014 version of ALEGRA, which was the most recently released code at the time, and was built using the Intel compiler.

Chapter 3

ALEGRA hypervelocity impact simulation

Our test considers the oblique hypervelocity impact of a copper sphere on a steel plate, based on the experiments of Grady and Kipp. [11] This problem exists as a test case for both ALEGRA and CTH, and is representative of production-scale impact and penetration problems that are modeled using the explicit solid dynamics capability in ALEGRA. The same problem was also considered in the 2011 scaling study of Doney et al. [6]

The problem is sometimes referred to as the "Whipple" problem, in reference to hypervelocity impact shields used for spacecraft protection. In the problem, a 3.18-mm-diameter sphere of copper impacts a 5.6-mm-thick plate of steel, at an impact velocity of 4520 m/s and an impact angle of 30.8°. Because of the obliquity, the problem must be modeled in 3D. The impact event results in pervasive particulation of both the plate and projectile material, and the formation of debris clouds propagating in the forward and backward directions, as seen in Figure 1.1. The problem is simulated out to a time of 15 μs, or about 14.5 μs after initial impact.

In the ALEGRA test, the copper is modeled using the Johnson-Cook yield strength model and the standard Sesame 3320 equation of state table. The steel is modeled using the Johnson-Cook yield strength model and the Sesame 2150 (iron) equation of state table. For both materials, dynamic fracture is modeled by use of a simple void insertion model with a trigger set at a tensile pressure of -0.345 GPa for copper and -0.1379 GPa for steel.

This problem is of interest here because it includes dynamics and constitutive models that are typical of ALEGRA use cases, and so should provide a realistic indication of the behavior of ALEGRA for everyday users. However, this problem does not exercise ALEGRA's multiphysics and brittle failure modeling capabilities. These are also of interest, but they lie outside the scope of this study.

The hypervelocity impact simulation is set up on a uniform 3D Eulerian finite-element mesh consisting of 8-node hexahedrons with unit aspect ratio. We apply simple uniform mesh refinement through the course of the parallel scaling study, with no geometry modifications. The mesh has at a minimum approximately 2 elements per projectile radius, and at a maximum about 20 elements per projectile radius. During mesh refinement, we rely on the built-in CFL-based stability criterion, so that the time step size also diminishes as the element dimension decreases.

Chapter 4

Weak scaling concept and scaling analysis configuration

To test the parallel performance of ALEGRA, we scale the oblique impact problem up in overall size in such way that the work per processor is held approximately constant as the number of cores is increased. Called "weak scaling" or "scaled speedup", this technique was introduced early on in distributed parallel computing by Gustafson et al. [12] as more relevant to scientific computing than "strong scaling", which suffers from *Amdahl's Law* [3]. In weak scaling, as the overall problem size grows, the amount of parallel work increases, thus counteracting the degredation of serial-to-parallel work ratio. Weak scaling appears throughout the computational physics literature, for example in the 2009 work work of Lin et al. [14] A brief description and comparison of weak and strong scaling is given in the 2007 paper of Colella et al. [5], along with a justification for the choice of weak scaling for certain types of analyses and certain types of codes.

Mesh refinement

For weak scaling here, the workload per processor is set by choosing an approximate number of elements per processor. Based on previous studies with ALEGRA for this type of problem, values near 10,000 elements per core are optimal in terms of minimizing communication cost. For each core count tested on each platform, a mesh resolution and corresponding element count is chosen which should result in 10,000 elements per core.

Although the meshing is done "inline" using the Pamgen capability, and is controlled with a single input parameter, the desired core count per processor can only be achieved to within several percent because of rounding involved in configuring the mesh to the dimensions of the problem domain. Because the mesh is not body-fitted and includes mixed-material elements, the time of initial impact also varies slightly with the mesh resolution.

Performance metrics

For each test, the performance of the code is evaluated by recording the total elapsed wall clock time, less time spent in input/output, and the total number of explicit time integration cycles (taken as the final values of the history variables CPUNOIO and CYCLE). The total element count must also be recorded since this is not known *a priori*.

From these data, a mean compute time per cycle is then tabulated, which is the ratio of the final elapsed time to the final cycle number. This is a representative performance metric that can be compared across the various mesh resolutions, time step sizes, and core counts, and takes into account the performance of the code throughout the entire calculation.

In some studies, such a metric has also been weighted by the total number of zones or elements. However, there is no need for such weighting so long as the core count on each processor is approximately constant. The analysis could be improved by disabling output to data files, which could have an indirect effect, however this was not done here. The simulations here include all output that would be requested in typical use cases, including inline visualization (Spymaster), data files (Exodus), history/diagnostic data (Hisplt), and restart files. The time spent writing these files was excluded from the performance metric.

In perfect weak scaling, the mean compute time per cycle would remain constant as the number of processors increases; but in practice, this would only occur if no communication took place. However, deviation from this ideal is commonly examined as a performance quantity of interest. And since the weak scaling behavior of the dominant communications occurring in domain decomposed, hyperbolic and parabolic PDE solvers is most often logarithmic in the number of processors, we expected logarithmic weak scaling behavior for ALEGRA as well. Such logarithmic scaling arises from limits in the underlying network as well as the MPI algorithm efficiency [13].

Cores per node

In the tests on Excalibur, the effect of idling a subset of cores on each node is explored. This should help to clarify whether competition for access to memory between processes on a node with shared memory could be negatively affecting performance. To do this, it is necessary to set an `MPI_OPT` environment variable `N`, and use batch scheduling node/core selection criteria to ensure only the requested number of cores is used on each node. In this way, weak scaling is studied on Excalibur for the full set of 32 cores per node, and for 24 and 16 cores per node.

Chapter 5

ALEGRA parallel performance

Performance testing was done on Cielo in December of 2012, and on Excalibur in April and July of 2015. The testing was done with the oblique impact problem described above, and the results are described below.

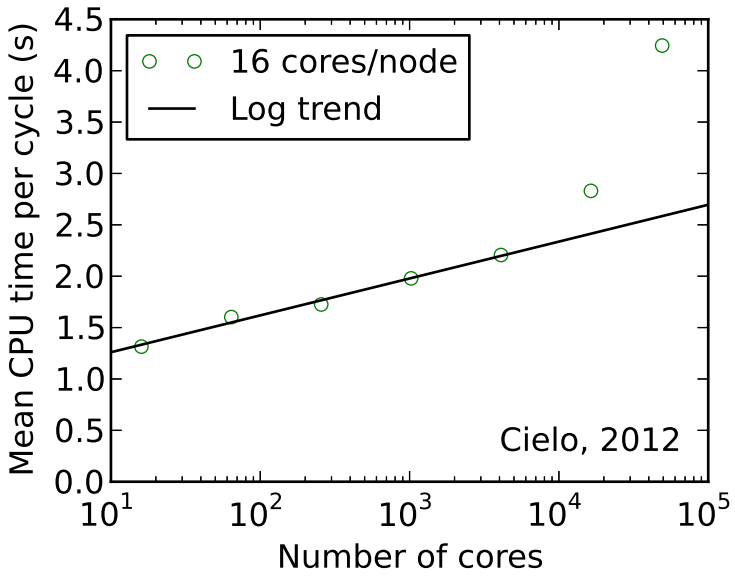

Figure 5.1: Test results for Cielo (2012).

Cielo test results, 2012

Testing was done on Cielo for core counts ranging from 16 to 49,152 and element counts ranging from 118,416 to 586,984,320. In each case, the full set of 16 cores was used on each node. The mean element count per core for these cases ranged between a minimum of 11,500 and a maximum of 12,116, which is higher than the target of 10,000. An additional test at 65,536 cores and 819,492,884 elements failed due to integer overflow, since some components of ALEGRA are limited to 32-bit integers.

The mean compute time per cycle from these weak scaling tests on Cielo in 2012 is shown in

Figure 5.1. The data are also tabulated in Table A.2. We observe logarithmic scaling up through 4,096 cores, as indicated by a logarithmic trend which is fit to these data points and shown on the plot. The slope of this curve is 0.199. At 16,384 cores, which was the next sampling point, the beginning of a significant upward excursion in compute time can be seen, while at 49,152 cores, there is a 50% upward excursion in mean compute time per cycle relative to the logarithmic trend. These last two points are excluded from the fit.

Excalibur test results, 2015

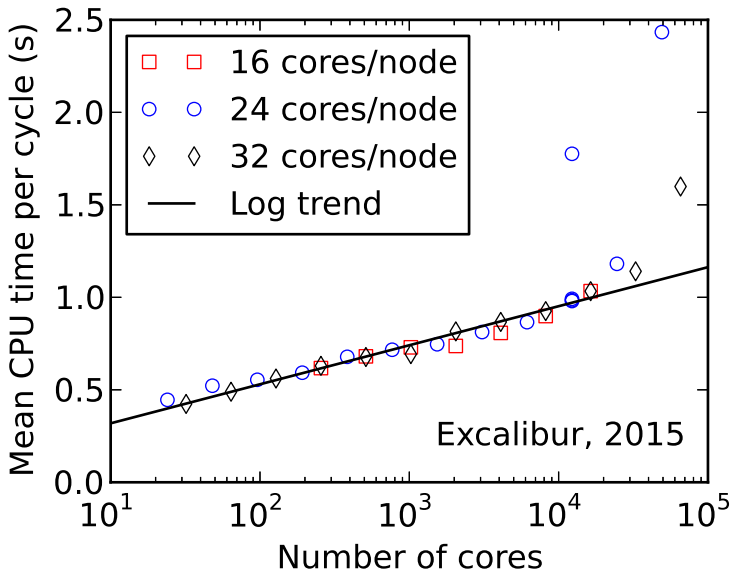

Figure 5.2: Test results for Excalibur (2015) for 32, 24, and 16 cores per node, including four repeated identical tests at 12,288 cores.

Testing was done on Excalibur for core counts ranging from 16 up to 65,536 and element counts ranging from 154,800 to 646,677,108. Three test series were conducted, using the full set of 32 cores per node, and reduced sets of 24 and 16. The mean element count per core for these cases ranged between a minimum of 8,861 and a maximum of 10,164. Several additional tests were conducted to find the largest element count possible in 3D under the 32-bit limitation. The largest successful element count was 691,109,952 on 61,488 cores, and a test at approximately 740 million elements resulted in failure.

The mean compute time per cycle from the tests on Excalibur in 2015 is shown in Figure 5.2, with data for 16, 24, and 32 cores per node. These data are also tabulated in Tables A.3-A.5. We observe logarithmic scaling up through 16,384 cores for 16 and 32 cores per node, and through 12,288 cores for 24 cores per node. As seen in the plot, the variation in cores per node had almost no perceptible influence on the overall scaling behavior. This indicates that competition for access to memory (mentioned in Section 4) is likely not a significant contributor to the performance

behavior at these scales. The logarithmic trend shown in Figure 5.2 fits the data up to those limits and has a slope of 0.092, showing significant improvement over the 2012 results from Cielo.

For higher core counts, significant excursions are again observed on Excalibur, as they were on Cielo. These are seen at core counts of 12,288, 24,576, and 49,152 for 24 cores per node, and at 65,536 cores for 32 cores per node. Core counts larger than 16,384 were not attempted at 16 cores per node. The magnitude of these excursions relative to the level predicted by the logarithmic scaling trend ranges from 13% to 120%. The origin of these excursions is discussed in Section 5.

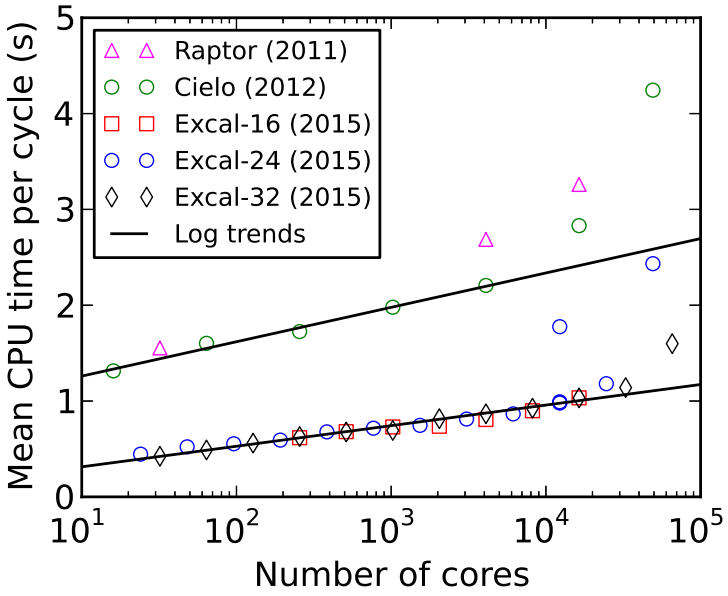

Figure 5.3: Test results for both platforms compared directly.

During the testing campaign on Excalibur, the appearance of seemingly unpredictable excursions at high core counts suggested the need for tests of repeatability. Therefore, the test at 12,288 cores for 24 cores per node was run a total of four times with the identical setup. The tests are visible in Figure 5.2. While the first run gave results on the trend line, one of the three subsequent runs, which were concurrent, showed a significant increase in compute time. This run leaped by more than 70% in mean compute time and is clearly visible as an outlier in the plot. This happened even though none of the characteristics of the solution showed any significant difference between the four tests. This provides an potentially important clue to the origin of some of the outlying data points in the Cielo and Excalibur testing, discussed in Section 5.

There was one additional difficulty in the testing on Excalibur, which is that jobs allocated to a certain node consistently became unresponsive after a short time. Since only a limited window of dedicated time was available to do the performance testing, this node had to be isolated by recursive bisection and excluded from subsequent job submissions.

Comparison of multiple platforms

The weak scaling trends from both machines are compared directly in Figure 5.3. Also included in this plot are the three data points from the 2011 study of Doney et al. [6] for which the element count per node was within 20% of the 10,000 element-per-core target used here. (See Table A.1.) These tests were run on the Raptor platform, which was a Cray XE6 system with 16 cores per node, operated by the U.S. Air Force Research Laboratory DSRC. (It has since been consolidated into another DoD HPCMP platform with a different name.)

Noticeable in Figure 5.3 is a significant improvement in the overall throughput of the ALEGRA code for the oblique impact problem, for the platforms tested here and in Reference [6] over the period 2011-2015. For core counts less than 12,000, the improvement in compute time is approximately a factor of 3. For larger core counts, the performance on Excalibur in 2015 still exceeds that observed on Cielo in 2012 or Raptor in 2011. Also notable is that the slope of the weak scaling trend decreased from 0.199 in 2012 to 0.092 in 2015, indicating an improvement in scaling that is roughly a factor of 2.

The majority of this remarkable improvement in overall performance of ALEGRA on these platforms is most likely due to improvements in hardware, particularly the improved memory speeds and the performance of the interconnect (see Table 2.1). The upgrade from 16 to 32 cores per node may also play a role because more of the communication could stay on each compute node. However, an extensive series of performance-oriented modifications were also made to the code infrastructure and some of the algorithms over this period of time, which contributed to the observed performance increase.

The performance effect of the code modifications can be seen in a study performed on a machine consisting of 16-core Intel Xeon E5-2760 processors in June of 2014 [8]. Two solid-dynamics problems were run using a version of the code as it existed at dates ranging from January 2013 to July 2014. One problem was a hypervelocity impact problem very similar to that studied here, and the other a 2D simulation for formation of a shaped charge jet. Each code version was run nine times for each date and each simulation used all 16 cores of a single compute node. As seen in Figure 5.4, an overall reduction of 27% in wall clock time was observed between January 2013 and July 2014 on this platform.

Additional information is provided by a single-compute-node study on the Cielo and Excalibur machines. The hypervelocity impact problem was run using all cores of a single compute node (16 cores for Cielo and 32 for Excalibur) and at two problem sizes (10,000 and 20,000 elements per core). In order to disentangle the performance contribution of the ALEGRA code and the hardware improvements, the 2013 ALEGRA software would have to be run at scale on the Excalibur platform. Due to limited time allocation, this test could not be run, and the relative contribution to the increased performance must be extrapolated from the single-node runs.

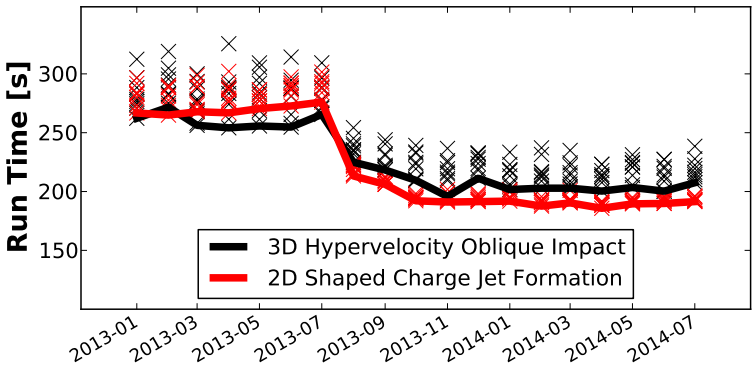

Figure 5.4: Simulation time of two hydrodynamics benchmark problems run with a version of ALEGRA as it existed at dates from January 2013 to July 2014. The times of each simulation are shown with an "X" and the solid lines run through the minimum time for each date.

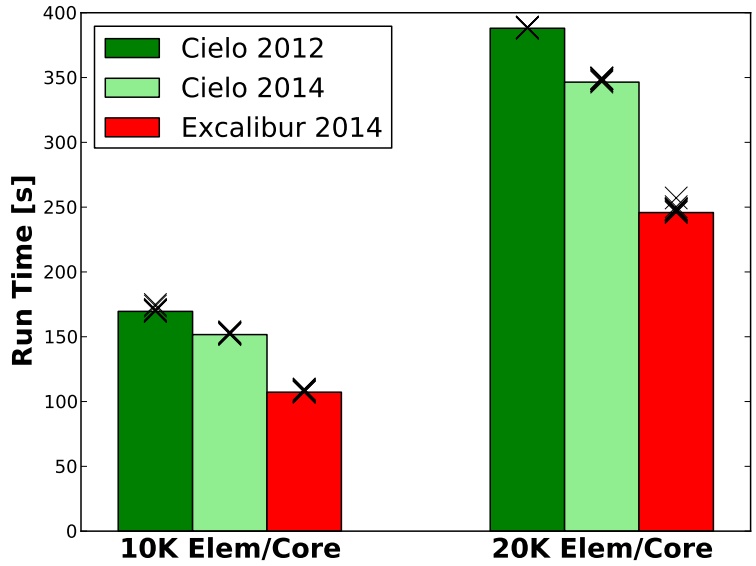

Figure 5.5: Simulation time of the hypervelocity impact problem run on a single compute node of Cielo using the 2012 release of ALEGRA, the 2014 release, and on a single compute node of Excalibur using the 2014 release. Ten runs of each were made and are marked with an "X".

The results shown in Figure 5.5 indicate a 10% speedup on Cielo due to code modifications, and a further 30% speedup due to hardware improvements between Cielo and Excalibur. Together this information indicates that improvements in both the hardware and the software contributed to the factor-of-3 speedup seen in the performance data on Excalibur at core counts under 10,000.

Alternate representation

The scaling trends shown in Figures 5.1-5.3 can also be represented in terms of the grind time rather than the mean compute time per cycle. By "grind time" we mean the time spent by the code per computational cycle *per element*. The per-element weighting results in a quantity that should diminish proportionally with mesh refinement, in the ideal case where communication costs are vanishingly small. This alternate representation of parallel scaling was used for CTH data appearing in Reference [6].

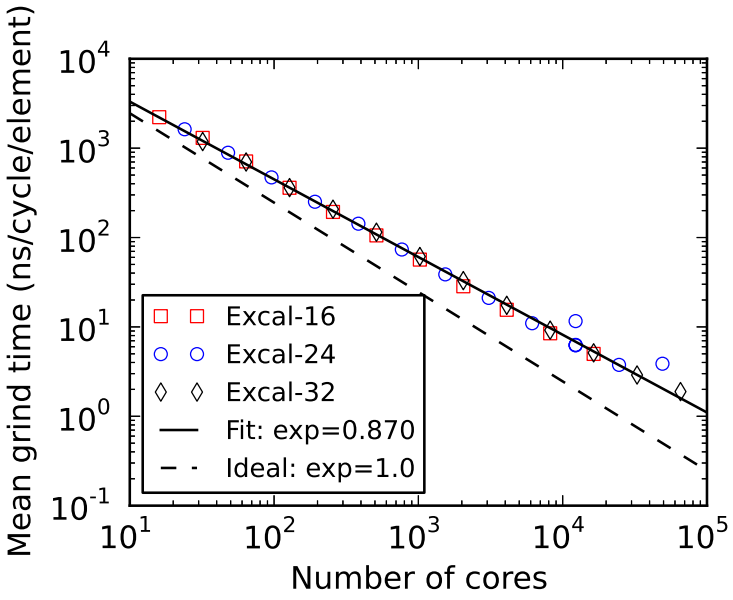

Figure 5.6: Scaling trends for ALEGRA on Excalibur (2015), using the grind-time representation, for the tests with 32, 24, and 16 cores per node.

The grind time is also recorded by ALEGRA on every timestep, excluding time spent in I/O operations, in the "GRINDNOIO" variable. For the tests on Excalibur, a mean grind time was obtained by averaging this quantity over all timesteps in each simulation. These data are plotted against the core count in Figure 5.6. A least-squares fit to all of the log-log data produces a slope of -0.87, indicating good parallel efficiency. These fits include the excursions at high core counts. The excursions are much less pronounced in this representation.

Discussion of excursions

Although most production-level ALEGRA calculations ALEGRA are currently done with thousands of cores, the excursions in the weak scaling trends above 10,000 cores are troubling. Examination of the physical characteristics of the solutions has ruled out any mechanical or thermal effect that could be to blame. Fracture patterns, time step sizes, and global energy balances in

the simulations indicate no deviation from the expected trends.

The conventional explanation of performance variability on massively parallel hardware is usually that periodic operating system daemon process issues, or resource contention is the major source of variability. [16]. Alternatively:

- **Bad Compute Nodes**: The existence of a compute node that would hang and not complete lends credence to the idea that a defective compute node operating at a reduced effective speed could slow down the entire calculation. For instance, the Xeon E5 system supports Adaptive Thermal Throttling, which will slow the clock rate if the DDR memory overheats. The Xeon also has a thermally managed Turbo Boost and AVX (vector computation) mode that will alter the clock speed to manage chip temperature. A defective fan, or other reduction in cooling could force an otherwise nominal performance node into reduced performance.

- **Mismatched compute nodes**: Figure 5.4 was generated by submitting a single node, multicore job nine times into the batch system. The resulting spread of times is likely indicative of the intrinsic spread of serial performance of nodes on that machine.

  In the "Whipple" problem, there are a few processor regions directly in the path of the impact that will have a very largly disproportionate amount of serial computation to perform. While the vendors for these large scale platforms such as Excalibur and Cielo make an effort to match processor performance, there is a manufacturing variation intrinsic to each type of chip set. If a high intensity computational region is randomly matched to a processor that happens to be at the bottom of the distribution of intrinsic serial speed, that combination will force the rest of the calculation to wait on that one node.

- **Bad or damaged Interconnect NIC or fiber**: If the communications interconnect to a particular node or set of nodes is compromised, this can also act to slow down the communication: if a fiber channel has a high level of noise, the detection of errors in the transmitted data will generate a request to re-send that data, with corresponding latency. Alternatively, the interconnect may route around a damaged node, adding hops (and delay) across the communication fabric. This re-routing may lead to localized overcrowding and "data storms" that will effectively slow down any computation using that section of the interconnect.

- **Node Layout**: It is standard practice when running performance studies on small numbers of nodes to run the analysis jobs multiple times. Experience has shown that the specific nodes allocated to a particular run of a job may have less than optimal node topology in the communication interconnect. These effects tend to diminish as the number of nodes requested increase. Though unlikely, the large excusion on Excalibur at 12288 nodes may be a result of an 'optimally bad' node allocation by the node manager.

It is also not possible to rule out issues in the ALEGRA code, or in its communication patterns and frequency, as playing a role in this behavior at very large core counts. Further investigaton and research is needed to unravel the true origin of this variability.

Chapter 6

Conclusions

In this scaling study, ALEGRA shows excellent parallel scaling performance for core counts less than about 10,000, which is typical of current production calculations. The total CPU time per computational cycle shows consistent logarithmic scaling (in the weak scaling sense) across three decades of core counts. This is true even for a production-style simulation environment that includes choices of dimensions, materials, and I/O settings that are characteristic of typical ALEGRA use cases for solid dynamics. Therefore, we expect that most users should see very good parallel scaling of ALEGRA for their problems of interest in this area.

However, the performance deteriorates somewhat for very large scales, between 10,000 and 65,536 cores. The CPU time per computational cycle shows significant excursions as high as 120% from the logarithmic trends. The behavior also becomes more variable, with some jobs at the same core count differing in performance by as much as 70%, while others lie on the logarithmic trend. We speculate that these excursions could be due to a number of hardware- and software-related issues, most of which are not easily addressed. The outcome at these scales indicates there is still work to be done in order to assess these behaviors, and to ensure users of ALEGRA can operate in these regimes in the future.

The most remarkable outcome of the study emerges from the comparison of the code's behavior at different stages of development over a period of three years, on similar Cray platforms designed and built over nearly the same period. Since the same scaling study was performed on both machines – on Cielo in 2012 and Excalibur in 2015 – we can assess the improvement of the ALEGRA code and the supercomputing hardware. We find that the overall rate of throughput of ALEGRA on these machines increased by approximately a factor of 3 over this time, and the slope of the weak-scaling trend decreased by a factor of approximately 2. This is true for core counts less than about 10,000. This means the time-to-solution seen by users is dramatically better than it was three years ago, and the benefit seen by using higher parallelism is also much better.

These improvements are partially due to noticeable improvements in the hardware – particularly the memory speed and interconnect performance. They are also due to extensive improvements made to the data layout and infrastructure used by the ALEGRA code. In the future, it is hoped that these improvements will also be extended beyond the 10,000-core limit experienced here.

References

[1] Cielo: NNSA capability supercomputer. http://www.lanl.gov/projects/cielo/. Accessed: 2015-09-04.

[2] DoD HPC unclassified systems. http://centers.hpc.mil/systems/unclassified.html. Accessed: 2015-09-04.

[3] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.

[4] R. F. Barrett, D. T. Stark, C. T. Vaughan, R. E. Grant, S. L. Olivier, and K. T. Pedretti. Toward an evolutionary task parallel integrated MPI + X programming model. In *Proceedings of the Sixth International Workshop on Programming Models and Applications for Multicores and Manycores*, PMAM '15, pages 30–39, New York, NY, USA, 2015. ACM.

[5] P. Colella, J. Bell, N. Keen, T. Ligocki, M. Lijewski, and B. van Straalen. Performance and scaling of locally-structured grid methods for partial differential equations. *Journal of Physics: Conference Series*, 78(1):012013, 2007.

[6] R. Doney, S. Schraml, M. Love, J. Niederhaus, D. Hensinger, S. Schumacher, and S. Carroll. Scaling exercises for armor mechanics using CTH and Alegra. Technical Report ARL-TR-5642, U.S. Army Research Laboratory, Aberdeen Proving Grounds, MD, August 2011.

[7] R. Doney and J. Stewart. Computational predictions of rear surface velocities for metal plates under ballistic impact. Technical Report ARL-TR-7327, U.S. Army Research Laboratory, Aberdeen Proving Grounds, MD, June 2015.

[8] R. R. Drake. The ALEGRA production application: strategy, challenges, and progress toward next generation platforms. Presented at the Algorithms and Abstractions for Assembly in PDE Codes workshop, May, 2014.

[9] A. C. Robinson et al. Alegra: An arbitrary Lagrangian-Eulerian multimaterial, multiphysics code. Reno, Nevada, January 2008. Proceedings of the 46th AIAA Aerospace Sciences Meeting. AIAA-2008-1235.

[10] R. W. Lemke et al. Effects of mass ablation on the scaling of X-ray power with current in wire-array Z pinches. *Physical Review Letters*, 102(025005), 2009.

[11] D. E. Grady and M. E. Kipp. Experimental and computational simulation of the high velocity impact of copper spheres on steel plates. *International Journal of Impact Engineering*, 15(5):645–660, 1994.

[12] John L. Gustafson, Gary R. Montry, and Robert E. Benner. Development of parallel methods for a 1024-processor hypercube. *SIAM Journal on Scientific and Statistical Computing*, 9(4):609–638, 1988.

[13] Torsten Hoefler, William Gropp, Rajeev Thakur, and Jesper Larsson Träff. Toward performance models of mpi implementations for understanding application scaling issues. In *Proceedings of the 17th European MPI Users' Group Meeting Conference on Recent Advances in the Message Passing Interface*, EuroMPI'10, pages 21–30, Berlin, Heidelberg, 2010. Springer-Verlag.

[14] P. T. Lin, J. N. Shadid, M. Sala, R. S. Tuminaro, G. L. Hennigan, and R. J. Hoekstra. Performance of a parallel algebraic multilevel preconditioner for stabilized finite element semiconductor device modeling. *Journal of Computational Physics*, 228:6520–6267, 2009.

[15] P. O'Malley and C. J. Garasi. Understanding the electrical interplay between a capacitive discharge circuit and exploding metal. Technical Report SAND2015-1132, Sandia National Laboratories, Albuquerque, NM, February 2015.

[16] David Skinner and W. Kramer. Understanding the causes of performance variability in hpc workloads. In *Workload Characterization Symposium, 2005. Proceedings of the IEEE International*, pages 137–149, Oct 2005.

[17] M. B. Zellner and G. B. Vunni. Photon Doppler velocimetry (PDV) characterization of shaped charge jet formation. *Procedia Engineering*, 58:88–97, 2013. (Proceedings of the 12th Hypervelocity Impact Symposium).

Appendix A

Run Time Data

Below are the raw run-time statistics used here from the scaling studies on Raptor, Cielo, and Excalibur.

Table A.1: Selected run-time statistics from Raptor in 2011 [6] with 16 cores per node.

Cores	CPUNOIO (s)	NSTEPS	CPUNOIO/NSTEPS (s)
32	3.1400e+02	202	1.554455
4096	2.9160e+03	1085	2.687558
16384	1.9717e+04	6047	3.260625

Table A.2: Run-time statistics from Cielo in 2012 with 16 cores per node.

Cores	CPUNOIO (s)	NSTEPS	CPUNOIO/NSTEPS (s)
16	2.3400e+02	178	1.3146
64	4.3900e+02	274	1.6022
256	7.2800e+02	422	1.7251
1024	1.3220e+03	668	1.9790
4096	2.3520e+03	1066	2.2064
16384	4.9900e+03	1763	2.8304
49152	1.5619e+04	3680	4.2443

Table A.3: Run-time statistics from Excalibur in 2015 with 16 cores per node.

Cores	CPUNOIO (s)	NSTEPS	CPUNOIO/NSTEPS (s)
16	6.7800e+01	167	4.059880e-01
32	9.8940e+01	208	4.756731e-01
64	1.4250e+02	259	5.501931e-01
128	1.4896e+02	308	4.836364e-01
256	2.4817e+02	402	6.173383e-01
512	3.4110e+02	501	6.808383e-01
1024	4.5529e+02	624	7.296314e-01
2048	5.8224e+02	790	7.370126e-01
4096	8.0029e+02	991	8.075580e-01
8192	1.136780e+03	1265	8.986403e-01
16384	1.753680e+03	1697	1.033400e+00

Table A.4: Run-time statistics from Excalibur in 2015 with 24 cores per node.

Cores	CPUNOIO (s)	NSTEPS	CPUNOIO/NSTEPS (s)
24	8.4680e+01	190	4.456842e-01
48	1.2477e+02	239	5.220502e-01
96	1.6402e+02	296	5.541216e-01
192	2.1682e+02	366	5.924044e-01
384	3.0862e+02	455	6.782857e-01
768	4.1058e+02	573	7.165445e-01
1536	5.3726e+02	720	7.461945e-01
3072	7.2855e+02	897	8.122073e-01
6144	9.8857e+02	1142	8.656480e-01
12288	1.408020e+03	1437	9.798330e-01
12288	1.414550e+03	1437	9.843772e-01
12288	2.551880e+03	1437	1.775838e+00
12288	1.425230e+03	1437	9.918093e-01
24576	2.314760e+03	1960	1.181000e+00
49152	6.541610e+03	2688	2.433635e+00

Table A.5: Run-time statistics from Excalibur in 2015 with 32 cores per node.

Cores	CPUNOIO (s)	NSTEPS	CPUNOIO/NSTEPS (s)
32	8.5550e+01	202	4.235149e-01
64	1.2286e+02	251	4.894821e-01
128	1.7950e+02	320	5.609375e-01
256	2.4804e+02	394	6.295431e-01
512	3.2848e+02	485	6.772784e-01
1024	4.1365e+02	597	6.928811e-01
2048	6.3145e+02	774	8.158269e-01
4096	8.4352e+02	973	8.669270e-01
8192	1.137230e+03	1229	9.253295e-01
16384	1.643630e+03	1591	1.033080e+00
32768	2.419160e+03	2121	1.140575e+00
65536	6.394080e+03	3999	1.598920e+00

DISTRIBUTION:

1 MS 0899	Technical Library (electronic copy), 9536

Sandia National Laboratories