SANDIA REPORT

SAND2013-8748
Unlimited Release
Printed October 14, 2013

Simulation Information Regarding Sandia National Laboratories' Trinity Capability Improvement Metric

Anthony M. Agelastos, Paul T. Lin

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

SAND2013-8748
Unlimited Release
Printed October 14, 2013

Simulation Information Regarding Sandia National Laboratories' Trinity Capability Improvement Metric

Anthony M. Agelastos, Paul T. Lin

Abstract

Sandia National Laboratories, Los Alamos National Laboratory, and Lawrence Livermore National Laboratory each selected a representative simulation code to be used as a performance benchmark for the Trinity Capability Improvement Metric. Sandia selected SIERRA Low Mach Module: Nalu, which is a fluid dynamics code that solves many variable-density, acoustically incompressible problems of interest spanning from laminar to turbulent flow regimes, since it is fairly representative of implicit codes that have been developed under ASC. The simulations for this metric were performed on the Cielo Cray XE6 platform during dedicated application time and the chosen case utilized 131,072 Cielo cores to perform a canonical turbulent open jet simulation within an approximately 9-billion-element-unstructured-hexahedral computational mesh. This report will document some of the results from these simulations as well as provide instructions to perform these simulations for comparison.

Acknowledgment

This document's authors acknowledge the contributions from the following individuals: Douglas W. Doerfler, Micheal W. Glass, and Stefan P. Domino. The work contained herein leverages the work from many Sandians across many organizations, many of whom are in the following list: Matthew T. Bettencourt, Ryan B. Bond, Eric C. Cyr, Karen D. Devine, Travis C. Fisher, Michael A. Heroux, Robert J. Hoekstra, Mark F. Hoemmen, Jonathan J. Hu, Kyran D. Mish, Brent M. Perschbacher, Eric T. Phipps, Kendall H. Pierson, Andrey Prokopenko, Sivasankaran Rajamanickam, Christopher Siefert, James M. Willenbring, and Alan B. Williams. Additionally, this document's authors acknowledge the support of the entire ACES Cielo dedicated application time team during the Trinity Capability Improvement Metric simulations, especially the late-night/ early-morning support from Jim Williams and the management efforts from Robert A. Ballance and Cory Lueninghoener. Joel Stevenson and John Noe are acknowledged for their effort with transferring and reviewing the data and disseminating it to all parties for rapid post processing.

Contents

Nomenclature **8**

1 Introduction **9**

2 Simulation Description **11**

2.1 Code Description 11

2.2 Problem Description 12

2.3 Environment Description 13

3 Package Description **15**

4 Simulation Instructions **17**

5 Determination of Capability Improvement Metric **19**

References **21**

Appendix

A Example Input File **23**

B Example Job Script **29**

C Nalu Log Excerpt **31**

List of Figures

2.1 This depicts the “R1” mesh with culled frontface and trimmed top. 13

List of Tables

2.1 This lists the names and sizes of the relevant computational meshes. 13

5.1 This table lists the maximum single-core memory consumption and the 2 figures of merit for 4 different computational meshes. 20

Nomenclature

DOE U.S. Department of Energy

NNSA DOE's National Nuclear Security Administration

SNL Sandia National Laboratories

ASC NNSA's Advanced Simulation and Computing Program

ACES NNSA's New Mexico Alliance for Computing at Extreme Scale

CIM Trinity Capability Improvement Metric

Nalu SIERRA Low Mach Module: Nalu

DAT Dedicated Application Time

PE Processing Element

CLE Cray Linux Environment

ALPS CLE's Application Level Placement Scheduler

CADE Cray Application Developer's Environment

PBS Portable Batch System

Chapter 1

Introduction

Sandia National Laboratories (SNL) selected SIERRA Low Mach Module: Nalu (Nalu), which is a fluid dynamics code that solves many variable-density, acoustically incompressible problems of interest spanning from laminar to turbulent flow regimes, to be utilized for the Trinity Capability Improvement Metric (CIM) since Nalu is fairly representative of implicit codes that have been developed under the U.S. Department of Energy's (DOE) National Nuclear Security Administration (NNSA) for the Advanced Simulation and Computing Program (ASC). The simulations for this metric occurred on Cielo [1], which is also funded by ASC and operated by the NNSA's New Mexico Alliance for Computing at Extreme Scale (ACES), during dedicated application time (DAT) and the chosen case utilized 131,072 Cielo cores to perform a canonical turbulent open jet simulation within an approximately 9-billion-element-unstructured-hexahedral computational mesh. This report will document some of the results from these simulations during the DAT, which took place from May 21 until May 22, 2013, as well as provide instructions to perform these simulations for comparison.

This report will begin with a description of the codes utilized and the simulation they perform (Chapter 2), the example files provided (Chapter 3), the instructions of how to perform the simulations (Chapter 4), and how to compute the overall figure of merit (Chapter 5).

Chapter 2

Simulation Description

This chapter describes the application code utilized for this work; the description of the Nalu code base is within Chapter 2.1, the simulation Nalu performs is described within Chapter 2.2, and the operating environment during the CIM DAT is described within Chapter 2.3.

2.1 Code Description

Nalu performs the fluid-dynamics simulation utilizing a computational mesh provided from STK_Adapt which performs the on-the-fly mesh refinement and decomposition. Nalu and STK_Adapt are built atop the Sierra Toolkit [2]. Nalu additionally leverages solvers from the Trilinos Project [3]. Sierra and Trilinos development is primarily funded by ASC. An excerpt from "Trinity Capability Improvement Metric" [4] that further describes Nalu is given below.

> The SIERRA Low Mach Module (internal code name: Nalu) solves a wide variety of variable density acoustically incompressible flows spanning from laminar to turbulent flow regimes. This generalized unstructured code base supports both elemental (control volume finite element) and edge (edge-based vertex centered) discretizations in the context of an approximate pressure projection algorithm (equal order interpolation using residual-based pressure stabilization). The generalized unstructured algorithm is second order accurate in space and time. A variety of turbulence models are supported, however, all are classified under the class of modeling known as Large Eddy Simulation (LES). The chosen coupling approach (pressure projection, operator split) results in a set of fully implicit sparse matrix systems. Linear solves are supported by the Trilinos Tpetra interface. Finally, this multi-physics simulation tool is built under the Sierra Toolkit.
>
> By deploying a code base written to leverage both Trilinos/Tpetra and the Sierra Toolkit, i.e., code bases that have been demonstrated to be 64-bit compliant and represent the path towards advanced architectures; the Nalu simulation tool can support mesh and degree-of-freedom counts well above the 2.14 billion count. The calculations are computationally intensive and require good cache usage. In typical applications, hundreds of thousands of time steps must be used.

> Communication patterns include both point-to-point exchanges typical of sparse graphs, consistent with assembly of partial sums, and collective reduction operations including global minimums, maximums, and summations. This code base is fairly representative of a wide range of implicit codes that have been developed in support of the ASC Integrated Codes (IC) project.

Nalu's input files currently adhere to the YAML 1.2 specification [5]. The version of Nalu and Trilinos used for the CIM were pulled from their respective development branches on May 11, 2013. The Nalu that is distributed for CIM-related activities is from version 4.29.4, which was tagged on June 11, 2013. The Nalu input files that are discussed in Section 3 and Appendix A are compatible with Nalu 4.29.4 which has a different syntax than the Nalu used for the CIM.

2.2 Problem Description

An excerpt from "Trinity Capability Improvement Metric" [4] that describes the problem of interest is given below.

> The test problem of interest is a turbulent open jet (Reynolds number of ~6,000) with passive mixture fraction transport using the one-equation Ksgs LES model. The problem is discretized on unstructured meshes with hexahedral elements. The baseline problem mesh consists of nine billion elements, with total degree-of-freedom count approaching 60 billion. Given the pressure projection scheme in the context of a monolithic momentum solve, the maximum matrix size is ~27 billion rows (momentum) followed by a series of smaller 9 billion row systems, i.e., for the continuity system (elliptic Pressure Poisson), mixture fraction and turbulent kinetic energy.

The computational domain consists of a cylindrical fluids region with an inner, smaller cylinder removed. The top of this cutout is where the turbulent open jet emanates. The initial discretization of this domain was created as the basis for all of the other mesh sizes. These mesh characteristics, including their names, are given below within Table 2.1. A figure depicting the R1 mesh is given below within Figure 2.1. The mesh nomenclature is "R<level of uniform refinement>;" the uniform refinement is relative to a reference mesh containing 34,215 elements and each level of refinement will increase the number of elements eightfold.

For the CIM, STK_Adapt read in the R4 mesh, performed 2 levels of uniform mesh refinement, decomposed the final problem for 131,072 processing elements (PEs), and then wrote the corresponding 131,072 Exodus files to disk, which collectively form the R6 mesh, that were eventually read in by Nalu to perform its simulation.

Table 2.1. This lists the names and sizes of the relevant computational meshes.

Mesh Name	File Name	# Elements	Size (MB)
R1	`1cm_ped_35KR1.g`	273,720	19
R2	`1cm_ped_35KR2.g`	2,189,760	144
R3	`1cm_ped_35KR3.g`	17,518,080	1,140
R4	`1cm_ped_35KR4.g`	140,144,640	9,124
R5	`1cm_ped_35KR5.g`	1,121,157,120	~73,024
R6	`1cm_ped_35KR6.g`	8,969,256,960	~584,448

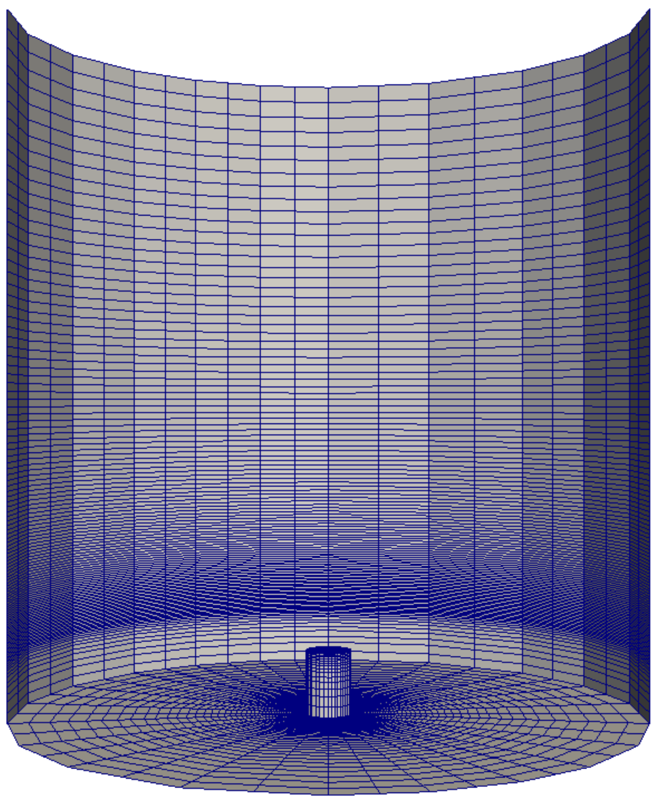

Figure 2.1. This depicts the "R1" mesh with culled front-face and trimmed top.

2.3 Environment Description

The CIM DAT began on May 21 and ended on May 22, 2013. During this time, Cielo had Cray Linux Environment (CLE) version 4.1.UP01 [6] installed with Cray Application Developer's Environment (CADE) version 6.16 [7]. This version of CADE contains the Cray

Message Passing Toolkit 5.6.0, which is Cray's MPICH-2 implementation; this particular version is based on Argonne National Laboratory's MPICH-2 version 1.5b1 [8].

Chapter 3

Package Description

The package that accompanied this document contains the R1, R2, R3, and R4 Exodus-format computational meshes, Nalu input files for performing simulations utilizing R1, R2, R3, R4, R5, and R6 meshes, and sample job scripts for executing Nalu with the R3, R5, and R6 meshes. These example files needed to perform these simulations are within the `Examples/Nalu_v4.29.4` directory. A list of this directory's files, each accompanied with a brief description, is below.

`1cm_ped_35KR1.g.gz` : This is a gzip-compressed Exodus file that contains the R1 computational mesh.

`1cm_ped_35KR2.g.gz` : This is a gzip-compressed Exodus file that contains the R2 computational mesh.

`1cm_ped_35KR3.g.gz` : This is a gzip-compressed Exodus file that contains the R3 computational mesh.

`1cm_ped_35KR4.g.gz` : This is a gzip-compressed Exodus file that contains the R4 computational mesh.

`nalu-turbElemOpenJet_R1_noReset-25step-dt16e-5-muelu_k110-mom_mixf_tke_sgs.i` : This is a Nalu input file that references the R1 mesh.

`nalu-turbElemOpenJet_R2_noReset-25step-dt8e-5-muelu_k110-mom_mixf_tke_sgs.i` : This is a Nalu input file that references the R2 mesh.

`nalu-turbElemOpenJet_R3_noReset-25step-dt4e-5-muelu_k110-mom_mixf_tke_sgs.i` : This is a Nalu input file that references the R3 mesh.

`nalu-turbElemOpenJet_R4_noReset-25step-dt2e-5-muelu_k110-mom_mixf_tke_sgs.i` : This is a Nalu input file that references the R4 mesh.

`nalu-turbElemOpenJet_R5_noReset-25step-dt1e-5-muelu_k110-mom_mixf_tke_sgs.i` : This is a Nalu input file that references the R5 mesh.

`nalu-turbElemOpenJet_R6_noReset-25step-dt05e-5-muelu_k110-mom_mixf_tke_sgs.i` : This is a Nalu input file that references the R6 mesh.

`milestone.xml` : This is an XML file that contains solver settings that Nalu reads in.

`run3.pbs` : This is a BASH script containing PBS directives and ALPS [9] commands that utilizes the R3 mesh with Nalu (this assumes that STK_Adapt has already been used to create the R3 mesh files).

`runR2_R3.pbs` : This is a BASH script containing PBS directives and ALPS commands that utilize STK_Adapt to perform one level of mesh refinement on the R2 mesh and to write out the decomposed R3 mesh to be used with Nalu.

`runR4_R5.pbs` : This is a BASH script containing PBS directives and ALPS commands that utilize STK_Adapt to perform one level of mesh refinement on the R4 mesh and to write out the decomposed R5 mesh to be used with Nalu.

`runR4_R6.pbs` : This is a BASH script containing PBS directives and ALPS commands that utilize STK_Adapt to perform two levels of mesh refinement on the R4 mesh and to write out the decomposed R6 mesh to be used with Nalu.

Chapter 4

Simulation Instructions

This chapter will describe the procedure for carrying out the CIM simulations. It is assumed that Nalu and STK_Adapt have already been installed; if this is not the case, then please install them first.

As previously described in Chapter 2, STK_Adapt performs on-the-fly refinement and decomposition. For an eventual n-PE simulation, STK_Adapt will read in a single mesh, e.g., R4, perform on-the-fly mesh refinement as many times as the user specifies, with each time resulting in an eight-fold increase in the number of elements, and then write to disk n mesh files. These n files are then read in by Nalu and the simulation begins. Because of this process, it is possible for all of the mesh refinement and decomposition to occur prior to any of the Nalu simulations beginning. This decision is left to the vendor, however it should be noted that this was the path chosen for the CIM simulations. Additionally, each CIM simulation read in its own copy of the computational mesh.

A description of how to perform the large CIM simulation is given below. These directions can be changed to accommodate other targeted PE counts. This simulation begins with the R4 mesh, STK_Adapt performs 2 levels of refinement and then writes out a decomposed R6 mesh, and then Nalu reads these mesh files in and performs the simulation. The first step is to prepare the current working directory, which is assumed to be the directory that will contain all of the simulation-specific files. To do this, the R4 mesh, R6 input file, and `milestone.xml` files must be copied and uncompressed if necessary.

```
$ cp /path/to/Examples/Nalu_v4.29.4/1cm_ped_35KR4.g.gz .
$ cp /path/to/Examples/Nalu_v4.29.4/\
      nalu-turbElemOpenJet_R6_noReset-25step-dt05e-5-\
      muelu_k110-mom_mixf_tke_sgs.i .
$ cp /path/to/Examples/Nalu_v4.29.4/milestone.xml .
$ gunzip 1cm_ped_35KR4.g.gz
$ mkdir results
$ mkdir 01
$ mv 1cm_ped_35KR4.g 01
```

Now, STK_Adapt can be utilized to create the partitioned R6 mesh from the R4 mesh.

The example below, since the CIM simulation was run on Cielo, utilizes ALPS commands; please convert these commands to their counterparts within the current environment.

```
$ aprun -n 131072 /path/to/stk_adapt_exe \
   --ioss_read_options="large,auto-decomp:yes" \
   --ioss_write_options="large,auto-decomp:no" \
   --input_mesh=01/1cm_ped_35KR4.g \
   --output_mesh=01/1cm_ped_35KR6.g \
   --refine=DEFAULT --load_balance=0 \
   --print_memory_usage=0 --estimate_memory_usage=0 \
   --query_only=0 --respect_spacing=0 --number_refines=2 \
   >> text_stkadaptout.log 2>&1
```

The decomposed R6 mesh should now be saved within the current working directory, with file names resembling `1cm_ped_35KR6.g.131072.000000` to `1cm_ped_35KR6.g.131072.131071`. Nalu can now be executed; the syntax to do this is given below.

```
$ aprun -n 131072 /path/to/nalu -d ‘pwd‘/ -i \
   nalu-turbElemOpenJet_R6_noReset-25step-dt05e-5-\
   muelu_k110-mom_mixf_tke_sgs.i \
   -o text_nalulog.log >> text_naluout.log 2>&1
```

As previously mentioned in Chapter 3, the job scripts that will perform these steps are included as part of the package. To learn more about these scripts please refer to Appendix B. Additionally, please refer to Appendix A for more information about Nalu's input file and what changes are made for the R1 - R6 simulations should any modifications be deemed necessary.

Chapter 5

Determination of Capability Improvement Metric

This chapter describes how to extract and compute the relevant figures of merit from the Nalu log file, which was `text_nalulog.log` in the example within Chapter 4. An excerpt from "Trinity Capability Improvement Metric" [4] that describes the figures of merit is below.

> Two figures of merit will be employed; both involve the solution of the momentum equations. The speedup of the two metrics will be weighted to produce a single speedup factor for SIERRA. The first figure of merit will be the average "solve" time per linear iteration. The second will be the average matrix "assemble" time per nonlinear step. Speedup will be defined as: Speedup = $\text{Speedup}_{solve} * 0.67 + \text{Speedup}_{assemble} * 0.33$.

The first figure of merit, FOM_S, which is the average "solve" time per linear iteration, is computed by extracting the average solve time for the momentum equations and then dividing it by the total number of linear iterations performed in the simulation. The second figure of merit, FOM_A, which is the average matrix "assemble" time per nonlinear step, is computed by extracting the average assemble time for the momentum equations and then dividing it by the total number of nonlinear iterations performed in the simulation; please refer to Appendix C for more details about how to extract these figures of merit. Table 5.1 lists Nalu's figures of merit for a weak scaling trend computed from the results obtained during the CIM DAT, performed on the Cielo Cray XE6 platform with 2.4 GHz AMD Opteron 6136 processors. All of these values are provided to facilitate comparisons from small-to-large computational scales.

Table 5.1. This table lists the maximum single-core memory consumption and the 2 figures of merit for 4 different computational meshes.

Mesh Name	# PEs	Loading (# Elem/ PE)	Max. Memory per Core (MB)	FOM_A (s)	FOM_S (s)
R3	256	68.7k	761	2.419	0.499
R4	2,048	68.7k	789	2.422	0.494
R5	16,384	68.7k	858	2.430	0.517
R6	**131,072**	**68.7k**	**1,081**	**2.422**	**0.521**

To provide an example to determine the capability improvement (CI) metric, suppose a simulation with the R7 mesh is performed, resulting in $FOM_S = 0.425$ and $FOM_A = 2.012$. The baseline CIM DAT R6 mesh figures of merit from Table 5.1 are $FOM_S = 0.521$ and $FOM_A = 2.422$. Please note that the R7 mesh would be 8x larger than the R6 mesh, which would increase the problem size by a factor of 8. The CI metric, for this example, is shown to be 9.7 from the equations below.

$$CI = [\texttt{increase in problem size}] \times [\texttt{Speedup}] \tag{5.0.1}$$

$$= [\texttt{increase in problem size}] \times [0.67 \times \texttt{Speedup}_{solve} + 0.33 \times \texttt{Speedup}_{assemble}] \tag{5.0.2}$$

$$= [8] \times [0.67 \times (0.521/0.425) + 0.33 \times (2.422/2.012)] \tag{5.0.3}$$

$$= 9.7 \tag{5.0.4}$$

References

[1] Los Alamos Lab: High-Performance Computing: Cielo supercomputer. http://www.lanl.gov/orgs/hpc/cielo/index.shtml, May 2013.

[2] Harold C. Edwards, Todd S. Coffey, Daniel Sunderland, and Alan B. Williams. Sierra Toolkit Tutorial. Technical report SAND2011-0814C, Sandia National Laboratories, Albuquerque, New Mexico 87185 and Livermore, California 94550, February 2011.

[3] Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P. Pawlowski, Eric T. Phipps, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. An overview of the Trilinos project. *ACM Trans. Math. Softw.*, 31(3):397–423, 2005.

[4] Douglas W. Doerfler. Trinity Capability Improvement Metric. http://www.nersc.gov/assets/Trinity--NERSC-8-RFP/Documents/Trinity-Capability-Improvement-v2.4.pdf, June 2013.

[5] Oren Ben-Kiki, Clark Evans, and Ingy döt Net. YAML Ain't Markup Language (YAML) Version 1.2. http://www.yaml.org/spec/1.2/spec.html, October 2009.

[6] Cray Inc. CLE 4.1.UP01 Software Release Announcement. http://docs.cray.com/books/S-9402-4101/S-9402-4101.pdf, April 2013.

[7] Cray Inc. Cray Application Developer's Environment User's Guide. http://docs.cray.com/books/S-2396-610/S-2396-610.pdf, May 2012.

[8] MPICH: A High-Performance, Portable Implementation of MPI. http://www.mcs.anl.gov/project/mpich-high-performance-portable-implementation-mpi, August 2013.

[9] Cray Inc. Workload Management and Application Placement for the Cray Linux Environment. http://docs.cray.com/books/S-2496-42/S-2496-42.pdf, April 2013.

Appendix A

Example Input File

This appendix contains an example Nalu input file for the R6 mesh. All of these examples are within `Examples/Nalu_v4.29.4`; the input file shown below is named `nalu-turbElemOpenJet_R6_noReset-25step-dt05e-5-muelu_k110-mom_mixf_tke_sgs.i`.

The only differences between the input files within `Examples/Nalu_v4.29.4` are from the `mesh:` (line 86), `termination_time:` (line 211), and `time_step:` (line 212) fields. These fields have been highlighted within the example below. The number of time steps, N, can be computed by

$$N = \frac{\texttt{termination\_time}}{\texttt{time\_step}}. \tag{A.0.1}$$

All of the input files contained herein are set for 25 time steps and 2 nonlinear steps per time step.

```
1  Simulations:
2    - name: sim1
3      time_integrator: ti_1
4      optimizer: opt1
5      error_estimator: errest_1
6
7  linear_solvers:
8
9    - name: solve_mom
10     type: tpetra
11     method: gmres
12     preconditioner: sgs
13     tolerance: 1e-4
14     max_iterations: 100
15     kspace: 100
16     output_level: 0
17
18   - name: solve_cont
19     type: tpetra
20     method: gmres
21     preconditioner: muelu
22     tolerance: 1e-4
23     max_iterations: 100
24     kspace: 110
```

```
    output_level: 0
    recompute_preconditioner: false
    ML_options_int:
      - name: "coarse: max size"
        value: 1000
      - name: "repartition: enable"
        value: 0
      - name: "repartition: min per proc"
        value: 1000
      - name: "max levels"
        value: 10
      - name: "repartition: Zoltan dimensions"
        value: 3
      - name: "smoother: sweeps"
        value: 2
      - name: "eigen-analysis: iterations"
        value: 15
      - name: "ML output"
        value: 11
      - name: "output"
        value: 11
    ML_options_real:
      - name: "repartition: max min ratio"
        value: 1.327
      - name: "aggregation: damping factor"
        value: 1.333
      - name: "aggregation: threshold"
        value: 0.02
    ML_options_string:
      - name: "aggregation: type"
        value: "Uncoupled"
      - name: "repartition: partitioner"
        value: "Zoltan"
      - name: "smoother: type"
        value: "Chebyshev"
      - name: "smoother: pre or post"
        value: "both"
      - name: "eigen-analysis: type"
        value: "power-method"

  - name: solve_tke
    type: tpetra
    method: gmres
    preconditioner: sgs
    tolerance: 1e-4
    max_iterations: 100
    kspace: 100
    output_level: 0

  - name: solve_mix
    type: tpetra
    method: gmres
    preconditioner: sgs
    tolerance: 1e-4
```

```
    max_iterations: 100
    kspace: 100
    output_level: 0

regions:

  - name: region_1
    mesh: 01/1cm_ped_35KR6.g
    use_edges: no

    equation_systems:
      name: theEqSys
      max_iterations: 2
      convergence_tolerance: 1.e-5

      solver_system_specification:
        velocity: solve_mom
        turbulent_ke: solve_tke
        mixture_fraction: solve_mix
        pressure: solve_cont

      systems:
        - LowMachEOM:
            name: myLowMach
            max_iterations: 1
            convergence_tolerance: 1e-5

        - TurbKineticEnergy:
            name: myTke
            max_iterations: 1
            convergence_tolerance: 1.e-5

        - MixtureFraction:
            name: myZ
            max_iterations: 1
            convergence_tolerance: 1.e-5

    initial_conditions:
      - constant: ic_1
        target_name: block_1
        value:
          pressure: 0
          velocity: [0,0,0]
          turbulent_ke: 0.0
          mixture_fraction: 0.0

    material_properties:
      target_name: block_1
      specifications:
        - name: density
          type: constant
          value: 1.1814e-3
        - name: viscosity
          type: constant
```

```
     value: 1.79e-4

  boundary_conditions:

  - wall_boundary_condition: bc_bottom
    target_name: surface_1
    wall_user_data:
      velocity: [0,0,0]
      turbulent_ke: 0.0

  - inflow_boundary_condition: bc_inflow
    target_name: surface_2
    inflow_user_data:
      velocity: [0,0,500]
      turbulent_ke: 150.0
      mixture_fraction: 1.0

  - wall_boundary_condition: bc_pipe
    target_name: surface_3
    wall_user_data:
      velocity: [0,0,0]
      turbulent_ke: 0.0

  - open_boundary_condition: bc_side
    target_name: surface_4
    open_user_data:
      velocity: [0,0,0]
      pressure: 0.0
      turbulent_ke: 1.0e-16
      mixture_fraction: 0.0

  - open_boundary_condition: bc_top
    target_name: surface_5
    open_user_data:
      velocity: [0,0,0]
      pressure: 0.0
      turbulent_ke: 1.0e-16
      mixture_fraction: 0.0

  solution_options:
    name: myOptions
    options:
      - hybrid_factor:
          velocity: 1.0
          turbulent_ke: 1.0
          mixture_fraction: 1.0

      - alpha_upw:
          velocity: 1.0
          turbulent_ke: 1.0
          mixture_fraction: 1.0

      - laminar_schmidt:
          turbulent_ke: 0.9
```

```
            mixture_fraction: 0.5

        - turbulent_schmidt:
            turbulent_ke: 1.0
            mixture_fraction: 1.0

    output:
      output_data_base_name: results/turbElemOpenJet.e
      output_frequency: 0
      output_node_set: no
      output_variables:
       - velocity
       - pressure
       - turbulent_ke
       - effective_viscosity_tke
       - dkdx
       - mixture_fraction
       - scalar_variance
       - scalar_dissipation

Time_Integrators:
  - StandardTimeIntegrator:
      name: ti_1
      start_time: 0
      termination_time: 12.5e-05
      time_step: 0.5e-05
      time_stepping_type: fixed
      time_step_count: 0

      regions:
        - region_1
```

Appendix B

Example Job Script

This appendix contains an example job script that will perform 2 levels of uniform, on-the-fly mesh refinement by STK_Adapt of the R4 mesh which will effectively create the R6 mesh, decompose it, write it to the file system, and then execute Nalu on this resultant R6 mesh. This setup is what was used for the final CIM data.

This script was written with PBS directives and will need to be modified so that the variables that define the paths to the installed STK_Adapt (`stk_adapt` and `stk_adapt_exe`) and Nalu (`fuego` and `fuego_exe`) executables. Additionally, this script is essentially what was used for the CIM simulations on Cielo; please substitute commands such as `aprun` with a necessary counterpart if needed.

```
#!/bin/bash

#PBS -l nodes=8192:ppn=16
#PBS -l walltime=0:30:00
#PBS -N R3
#PBS -j oe

cd $PBS_O_WORKDIR

size=131072

run=2

stk_adapt=stk_adapt_exe-code20130725-co4.29.4-amagela-PE4.1.40-icc11.0.72-build20130724
stk_adapt_exe=/lscratch2/ptlin/fuego/bin/$stk_adapt

textout=text_output-run$run

date > $textout
echo aprun -n $size $stk_adapt_exe --ioss_read_options="large,auto-decomp:yes" --ioss_write_options="large,auto-decomp:no" --input_mesh=01/1cm_ped_35KR4.g --output_mesh=01/1cm_ped_35KR6.g --refine=DEFAULT --load_balance=0 --print_memory_usage=0 --estimate_memory_usage=0 --query_only=0 --respect_spacing=0 --number_refines=2 >> $textout 2>&1
aprun -n $size $stk_adapt_exe --ioss_read_options="large,auto-decomp:yes" --ioss_write_options="large,auto-decomp:no" --input_mesh=01/1cm_ped_35KR4.g --output_mesh=01/1cm_ped_35KR6.g --refine=DEFAULT --load_balance=0 --print_memory_usage=0 --estimate_memory_usage=0 --query_only=0 --respect_spacing=0 --number_refines=2 >>
```

```
     $textout 2>&1
date >> $textout

sleep 5

# turn off auto-decomp (files are left in split state after stk_adapt_exe)
export IOSS_PROPERTIES="INTEGER_SIZE_API=8:INTEGER_SIZE_DB=8"

fuego=nalu-code20130725-co4.29.4-amagela-PE4.1.40-icc11.0.72-build20130724
fuego_exe=/lscratch2/ptlin/fuego/bin/$fuego

basename=nalu-turbElemOpenJet_R6_noReset-25step-dt05e-5-muelu_k110-mom_mixf_tke_sgs
outfile=$basename-p${size}-$fuego-run$run
textout=text_output-$outfile

date > $textout
echo aprun -n $size $fuego_exe -d `pwd`/ -i $basename.i -o $outfile >> $textout 2>&1
aprun -n $size $fuego_exe -d `pwd`/ -i $basename.i -o $outfile >> $textout 2>&1
date >> $textout
```

Appendix C

Nalu Log Excerpt

This appendix will describe the relevant parts of the Nalu log file and extract the necessary information to compute the figures of merit. The following parameters should be examined for each simulation: the time step count, the average momentum assemble time (t_A), the average momentum solve time (t_S), and the average number of linear iterations for the momentum equations (i). To assist with this description, the last 93 lines of the log file for the CIM is given below. In this file, there were 480 linear iterations and 50 nonlinear iterations.

The time step count is important to check because each of the input files distributed with this document were set to perform 25 time steps. If 25 time steps are not performed, then the performance numbers will be askew. The Nalu log file provides a time step status update; to check how many time steps elapsed, simply go to the last such update. This update line is highlighted in the reference log file below and is on line 1,295.

The remaining data to extract is near the end of the Nalu log file within the section "`Timing for Eq  MomentumEQS`." The average momentum assemble time, in seconds, is the first time given on the line with "`assemble --`;" this is highlighted within the reference log file on line 1,355 and is 121.08 seconds for the CIM. The average momentum solve time is the first time given on the line with "`solve --`;" this is highlighted within the reference log file on line 1,357 and is 250.077 seconds for the CIM. The average number of linear iterations for the momentum equations is the first number given on the line with "`linear iterations --`;" this is highlighted within the reference log file on line 1,359 and is 9.6 for the CIM.

With these values, the figures of merit can be calculated with the following equations:

$$FOM_S = \frac{t_S}{25 \times 2 \times i}, \tag{C.0.1}$$

$$FOM_A = \frac{t_A}{25 \times 2}. \tag{C.0.2}$$

The equations to compute FOM_S and FOM_A contain "25×2" in the denominator. This is in reference to each simulation requiring 25 time steps and each time step contains 2 nonlinear loops; for more information about this, please refer to Appendix A.

```
*******************************************************
Time Step Count: 25 Current Time: 0.000125 dt: 5e-06
 Max Courant: 2.15066 Max Reynolds: 9.44179
   Region Nonlinear Iteration: 1/1

  Equation System Iteration: 1/2
   myLowMach Iteration: 1/1
EqSystem Name:       MomentumEQS
        iters      = 9
linearResidNrm     = 1.14218e-05
nonlinearResidNrm  = 2.46289e-06

EqSystem Name:       ContinuityEQS
        iters      = 23
linearResidNrm     = 2.72972e-05
nonlinearResidNrm  = 1.44646e-05

   myTke Iteration: 1/1
EqSystem Name:       TurbKineticEnergyEQS
        iters      = 8
linearResidNrm     = 1.50749e-05
nonlinearResidNrm  = 2.82666e-06

   myZ Iteration: 1/1
EqSystem Name:       MixtureFractionEQS
        iters      = 8
linearResidNrm     = 2.87205e-08
nonlinearResidNrm  = 6.1534e-09

  Equation System Iteration: 2/2
   myLowMach Iteration: 1/1
EqSystem Name:       MomentumEQS
        iters      = 9
linearResidNrm     = 1.0301e-05
nonlinearResidNrm  = 1.35252e-07

EqSystem Name:       ContinuityEQS
        iters      = 24
linearResidNrm     = 2.22366e-05
nonlinearResidNrm  = 1.36126e-05

   myTke Iteration: 1/1
EqSystem Name:       TurbKineticEnergyEQS
        iters      = 8
linearResidNrm     = 1.49953e-05
nonlinearResidNrm  = 2.20165e-07

   myZ Iteration: 1/1
EqSystem Name:       MixtureFractionEQS
        iters      = 8
linearResidNrm     = 2.87802e-08
nonlinearResidNrm  = 6.21797e-11
```

```

Begin Timer Overview for Region: region_1
--------------------------------
Timing for Eq: myLowMach
         assemble --    avg: 0  min: 0  max: 0
    load_complete --    avg: 0  min: 0  max: 0
            solve --    avg: 0  min: 0  max: 0
             misc --    avg: 30.9048  min: 29.5858  max: 31.974
Timing for Eq: MomentumEQS
         assemble --    avg: 121.08   min: 119.383  max: 140.421
    load_complete --    avg: 57.3265  min: 37.8664  max: 58.8717
            solve --    avg: 250.077  min: 249.6  max: 250.196
             misc --    avg: 17.6281  min: 17.1331  max: 18.2331
linear iterations --    avg: 9.6  min: 4  max: 11
Timing for Eq: ContinuityEQS
         assemble --    avg: 35.6083  min: 35.1582  max: 40.4265
    load_complete --    avg: 27.3833  min: 22.5294  max: 27.8217
            solve --    avg: 1602.35  min: 1600.58  max: 1602.54
             misc --    avg: 16.3106  min: 16.133   max: 17.3331
linear iterations --    avg: 35.92  min: 23   max: 101
Timing for Eq: myTke
         assemble --    avg: 44.0112  min: 43.3627  max: 48.463
    load_complete --    avg: 20.4834  min: 16.001   max: 21.1413
            solve --    avg: 77.8191  min: 77.6768  max: 77.8729
             misc --    avg: 14.3252  min: 13.7849  max: 15.493
linear iterations --    avg: 9.22   min: 8  max: 11
Timing for Eq: myZ
         assemble --    avg: 43.3186  min: 42.0306  max: 111.259
    load_complete --    avg: 124.731  min: 56.9236  max: 126.292
            solve --    avg: 369.845  min: 369.347  max: 369.931
             misc --    avg: 58.9038  min: 58.4077  max: 59.9637
linear iterations --    avg: 8.78   min: 8  max: 10
Timing for IO:
     io read mesh --    avg: 62.4344  min: 3.1802   max: 69.9324
 io output fields --    avg: 0  min: 0  max: 0
Timing for Simulation:
           main() --    avg: 4214.94  min: 4092.27  max: 4234.7
Memory Overview:
stk_nalu memory: total (over all cores) current/high-water mark=        115452 G    122240 G
stk_nalu memory:   min (over all cores) current/high-water mark=       828.953 M    879.68 M
stk_nalu memory:   max (over all cores) current/high-water mark=       1.02418 G    1.08121 G
```

DISTRIBUTION:

1	MS 0807	Anthony M. Agelastos, 9326
1	MS 0845	Micheal W. Glass, 1545
1	MS 1319	Douglas W. Doerfler, 1422
1	MS 1320	Paul T. Lin, 1426
1	MS 0899	Technical Library, 9536 (electronic copy)

Sandia National Laboratories