CMT-bone: A Proxy App for Compressible Multiphase Turbulence Simulation Software

Conference Paper · September 2015

DOI: 10.1109/CLUSTER.2015.134

8 authors, including:

Nalini Kumar
University of Florida
4 PUBLICATIONS **8** CITATIONS
SEE PROFILE

Mrugesh Shringarpure
ExxonMobil, Spring, Tx, United States
10 PUBLICATIONS **50** CITATIONS
SEE PROFILE

Jason F. Hackl
University of Florida
17 PUBLICATIONS **78** CITATIONS
SEE PROFILE

Sivaramakrishnan Balachandar
University of Florida
391 PUBLICATIONS **8,734** CITATIONS
SEE PROFILE

CMT-bone: A mini-app for Compressible Multiphase Turbulence Simulation Software

Nalini Kumar[1], Mrugesh Sringarpure[2], Tania Banerjee[3], Jason Hackl[2], S Balachandar[2], Herman Lam[1], Alan George[1]

[1]Department of Electrical and Computer Engineering
[2]Department of Mechanical and Aerospace Engineering
[3]Department of Computer Information Science and Engineering
University of Florida, Gainesville, FL
[1]*{nkumar, hlam, george}@hcs.ufl.edu*
[2]*{mrugeshs, jason.hackl, bala1s}@ufl.edu*
[3]*tmishra@cise.ufl.edu*

***Abstract*—Designed with the goal of mimicking key features of real HPC workloads, mini-apps have become an important tool for co-design. An investigation of mini-app behavior can provide system designers with insight into the impact of architectures, programming models, and tools on application performance. Mini-apps can also serve as a platform for fast algorithm design space exploration, allowing HPC application developers to evaluate their design choices before significantly redesigning the application codes. Consequently, it is prudent to develop a mini-app alongside the full blown application it is intended to represent. In this paper, we present CMT-bone a mini-app for the compressible multiphase turbulence (CMT) application, CMT-nek, being developed to extend the physics of the CESAR Nek5000 application code. CMT-bone represents the most computationally intensive kernels of CMT-nek and the communication operations involved in nearest-neighbor updates and vector reductions. The mini-app represents CMT-nek in its most mature state and going forward it will be developed in parallel with the CMT-nek application. We describe these kernels and discuss the role that CMT-bone has played in enabling interdisciplinary collaboration by allowing application developers to work with computer scientists on performance optimization on current architectures and performance analysis on notional future systems.**

Keywords — miniapp, CMT-nek, CMT-bone, co-design, performance analysis, compressible multiphase turbulence

I. Introduction

Computational scientists and engineers use High-Performance Computing (HPC) systems to simulate increasingly complex models of real world problems such as climate modeling, fluid dynamics, and nuclear reaction simulations. Already running on massive supercomputers, more complex and detailed simulations are needed to gain insight into a particular domain, which in turn requires even faster supercomputers. Emerging and future system architectures will be able to provide the computational power necessary for these insights that scientists demand. However, these systems will likely have significantly different node and system architecture from the systems that exist today. Identifying likely exascale architectures, and optimizing these applications for these architectures is crucial for scientists to prepare for exascale systems.

One such application being prepared to be deployed on exascale systems is *CMT-nek*. It is a compressible multiphase turbulence (CMT) simulation software being developed by researchers and scientists at PSAAP-II Center for Compressible Multiphase Turbulence at University of Florida. CMT-nek is being developed to perform simulation of instabilities, turbulence, and mixing in particulate-laden flows under conditions of extreme pressure and temperature. CMT has applications in many environmental, industrial, and national security areas. Medical applications such as needleless drug delivery, lithotripsy, and micro-bubble-enhanced ultrasound imaging rely upon compression wave or shock interaction with particles and bubbles. CMT dominates the behavior of natural flows such as explosive volcanic eruptions, supernovae, and dust explosions in coal mines and grain silos. In many applications of national defense and security CMT plays an important role in our ability to accurately predict and control explosive dispersal of particles.

The CMT-nek development team faces the challenging task of developing optimized software for exascale machines whose architectures are as yet unknown and are not going to be available before the end of this decade. There are too many variables such as system architecture, programming models etc. that can affect the application performance. Since both the exascale application and architecture are being explored currently, it a great opportunity for co-design. Traditionally, architecture DSE is carried out by computer engineers, who are generally not very familiar with the application, before making system design decisions. Application code developers can benefit from early algorithm DSE before making changes to the production codes. Since the application codes are quite large and complex, using the entire application code for carrying out early DSE will be time and resource intensive. A smaller, self-contained, representative mini-app can serve as a vital tool for collaboration between researchers from different domains.

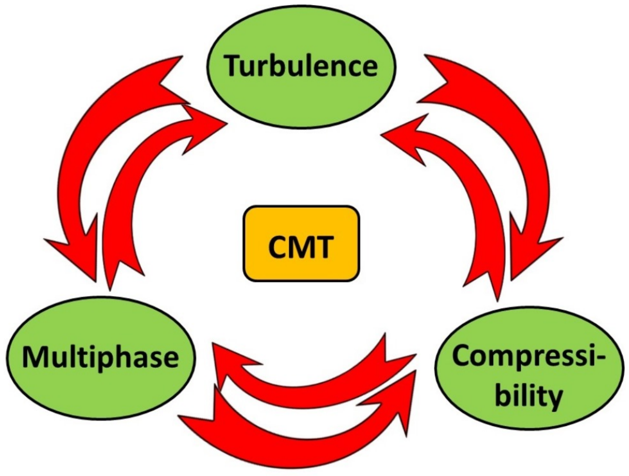

Fig. 1. CMT-nekscope

In this paper we introduce a new mini-app called *CMT-bone*, created as a performance proxy for the CMT-nek application, discuss its key features, demonstrate its use for performance optimization, and explain its role in architecture modeling and simulation. CMT-bone represents the most computationally intensive portion of CMT-nek, the spectral element solver, and the key communication operations such as the nearest-neighbor exchanges and vector reductions. Since the application code developers are the most knowledgeable about CMT-nek application, they are responsible for developing and maintaining the mini-app and ensuring that CMTbone represents the most-mature state of the application.

Mini-apps cannot be a perfect representation of the application performance and thus, it is important to understand the link between the application and the mini-app. We describe a conceptual model of CMT-nek and explain how the key kernels can be abstracted into simpler representative linear algebraic operations and communication routines in CMT-bone. The aim of this qualitative description is to establish that CMT-bone adequately represents CMT-nek and is a good starting point for application performance analysis.

As mentioned earlier, CMT-bone has been designed to serve as a vehicle for collaboration between application code developers, performance engineers, and computer engineers. The computer scientists in our group have used the mini-app to evaluate various optimization strategies for key kernels. We discuss the optimization results for one such kernel - the spectral element solver. Since the mini-app can be used to characterize the computation and communication behavior of CMT-nek, it can help computer engineers, who are interested in modeling and predicting performance on notional future systems, select architecture models with appropriate fidelity. We present the performance characteristics of CMT-bone, results obtained from optimization of key kernels, and insights gained for modeling application communication.

The rest of the paper is organized as follows: Section II discusses other mini-app development efforts, their use for co-design, and validation of mini-apps. Section III provides an overview of the parent CMT-nek application and a conceptual model for the key application features. In section IV we describe the abstractions of CMT-nek kernels that are included in CMT-bone. Section V presents an analysis and preliminary results of optimizing the most expensive computation kernel in CMT-bone, the spectral element solver. In section VI we discuss the communication behavior of the mini-app that are necessary for building robust network models for system simulation. Finally, Section VII summarizes our experience in developing and working with the CMT-bone mini-app and lays out our vision for the future of CMT-bone mini-app.

II. Background

Scientific simulation and data analysis requirements are fast approaching exascale. As part of the its Exascale Initiative, DOE is trying to address the challenges in producing exascale capabilities by engaging in public/private partnership with the leaders in computing industry. The DOE FastForward and DesignForward projects are aimed at advancing research and development in key areas. As part of these two programs, industry will work towards meeting the requirements of future extreme-scale applications. In order for this R&D effort to succeed it is vital that the architectures are designed with the knowledge of target applications and their requirements. This strategy of co-design has long been used in the embedded systems community but its adoption in HPC is more recent and has faced several challenges.

Effective co-design requires collaboration among software developers, application developers, and hardware architects in order to streamline the design process. In [7] the authors stress on the importance of co-design as a key strategy for exascale computing. Since DOE applications cannot be shared with the vendors and vendors are reluctant to share detailed information on architectures, mini-apps and proxy architectures are almost a necessity for co-design. The authors discuss the results of applying co-design to various problems and the unique insights gained into the solutions for problems faced in the HPC community.

The Mantevo project provides a set of proxies or mini-apps, for use as the central element in co-design efforts for next-generation scalable computers and applications [1]. Each mini-app in the Mantevo Miniapp Suite consists of some or all the dominant application kernels and is meant to serve as performance proxy of the original application. These mini-apps can be used to study various multi-core performance issues and provide an ideal testbed for application programming model studies. Prolego, a configurable mini-app, can be calibrated to a target application as opposed to the other mini-apps which are designed for a specific application. Initial testing of Prolego has shown a promising ability to mimic the performance of application-specific mini-apps [1].

While mini-apps are a useful tool, it is important to treat them as guidelines and not targets for optimization. It is important to understand the relationship between the application and its mini-app. A verification and validation methodology for identifying and understanding this relationship is described in [8] and [9]. It was useful in showcasing the link between four mini-apps and the corresponding full-scale application and exposed several issues in the mini-apps representation of the applications that will have to be addressed in the future [9]. Validation of miniFE against the Charon application showed that the mini-app was accurate in predicting sensitivity to memory bandwidth, but less accurate for predicting cache behavior. We understand the importance of such rigorous validation and we plan on performing such analysis with CMT-bone in near future. With some confidence and understanding of the predictive capabilities of mini-apps they can be used with simulation tools such as the Structural Simulation Toolkit (SST) to enable co-design. [8][16]

A mini-app is developed to capture one or few key performance features of an application and cannot capture its behavior under all circumstances. It is possible and beneficial to have more than one mini-app to better represent an application. Also, majority of mini-apps are created after the application has matured. While this makes the development of the mini-app easier, if the mini-app development keeps pace with the key algorithm changes in the parent application it may shorten the co-design loop. This is also our goal in developing and using CMT-bone, the CMT-nek mini-app.

III. CMT-NEK

The goal of CMT-nek is to advance predictive simulation science by solving a complex multiscale problem combining three complex physics: compressibility, multiphase flow, and turbulence (figure 1), at an unprecedented level of physical detail and integration. CMT occurs often under extreme conditions of pressure and temperature, and as a result is not easily amenable to high-fidelity experiments and diagnostics. It presents a fascinating array of poorly-understood instabilities, their transition to turbulence, and coupling between dispersed species and turbulence. Typically these processes manifest over a wide range of strongly interacting length and time scales. Such characteristics greatly limit our ability to perform high fidelity simulations on current state-of-the-art (petascale) supercomputers. A direct consequence of this is that the current computational approaches are based on models and closures that were developed from simplified experiments and simulations with reduced physics, and as a result are largely empirical. Therefore, fully validated exascale simulation that incorporates all the relevant physics is perhaps the only path to fundamental breakthroughs that can lead us out of current empiricism.

A. *CMT-nek development plan*

To develop a novel software that can perform high-fidelity simulations of compressible multiphase turbulent flows on massively parallel supercomputers in a short time frame (3 years) is an ambitious project. Several choices regarding spatial discretization, numerical schemes for time integration, spatial derivatives and shock capturing, strategies to partition the computational domain on a distributed system, and MPI

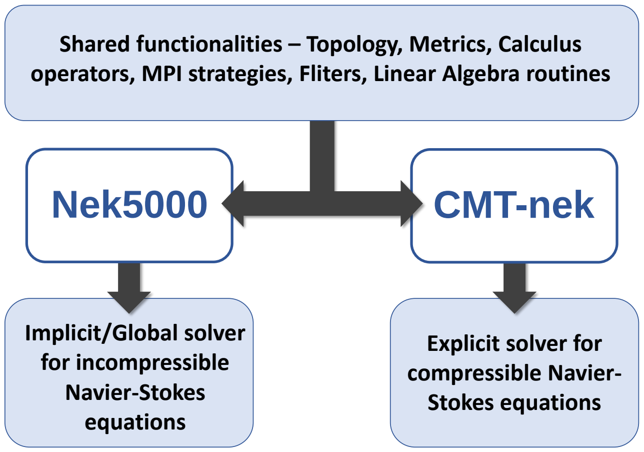
Shared functionalities – Topology, Metrics, Calculus operators, MPI strategies, Fliters, Linear Algebra routines
Nek5000
CMT-nek
Implicit/Global solver for incompressible Navier-Stokes equations
Explicit solver for compressible Navier-Stokes equations

Fig. 2. Schematic showcasing the association of CMT-nek and Nek5000 - their shared functionalities and fundamental difference, i.e., the governing equations involved.

strategies for exchanging data play a crucial role in making an efficient petascale or exascale application. It was decided that CMT-nek will leverage the choices made by existing petascale codes for spatial discretization, spatial derivatives, and MPI strategies to speed up the code development and focus the effort on implementation of specific features like shock capturing and point particle tracking. Therefore, CMT-nek is being developed from a production release of Nek5000. Nek5000 is a Gordon Bell prize winning open-source spectral element computational fluid dynamics code developed at Argonne National Laboratory for simulating unsteady incompressible fluid flow with thermal and passive scalar transport [11]. It is a highly scalable code, with demonstrated strong scaling to over a million MPI ranks on ALCF BG/Q Mira. CMT-nek aims to take advantage of this sustained performance by inheriting the MPI strategies used in Nek5000. Also, since Nek5000 is an active project, any changes and optimizations made to the Nek5000 code can be leveraged for CMT-nek.

Nek5000 has demonstrated superior parallel performance and high-order accuracy in simulating unsteady incompressible flows associated with complex systems. However, the existing framework of Nek5000 is not suitable for simulating compressible multiphase turbulent flows. To extend Nek5000 to solve compressible multiphase flows, two fundamental modifications are required - solve compressible multiphase flow equations [12] and implement discontinuous Galerkin method for solving the governing equations. The development plan for CMT-nek is to incorporate these modifications within Nek5000 so that CMT-nek can leverage common functionalities and maximize code reuse. Figure 2 shows the relationship between Nek5000 and CMT-nek, the similarities and major differences. CMT-nek is designed such that it hooks into Nek5000 as a branch of execution and source code. This development plan enables CMT-nek to inherit broad range of common functionalities like element topology, approximation polynomials, high-order spatial discretization, MPI strategies, and suite of

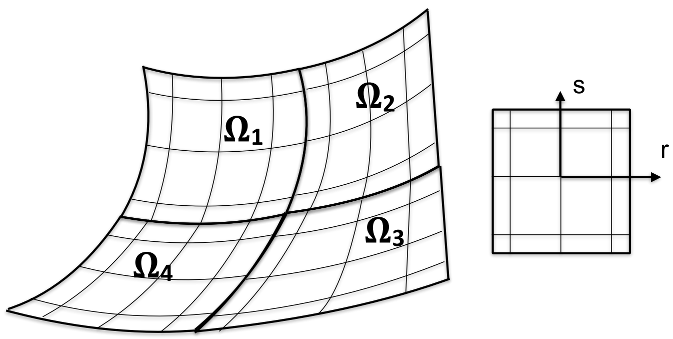

Fig. 3. Representation of partitioning of computational domain in CMT-nek

linear algebra operations from Nek5000. As a consequence CMT-nek will inherit the superior parallel performace, higher-order accuracy, and optimized code base for linear algebra from Nek5000.

CMT-nek will be developed over a period of three years. The current version of CMT-nek is an explicit solver for compressible Navier-Stokes equations with limited multiphase coupling in the form of a nozzling term in the momentum equation. In the following years complete multiphase coupling, shock capturing, lagrangian point particle tracking, and real gas models will be added.

B. Conceptual model of CMT-nek

Compressible multiphase flow equations solved by CMT-nek can be represented using the following conservation law

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot f(\mathbf{U}, \nabla \mathbf{U}) = R \qquad (1)$$

In the above equation, **U** is a vector of conserved variables (mass, momentum, and energy). The two terms on the left hand side are the rate of change of the conserved variable and the flux of conserved variable. The term on the right hand side represents the source term which captures the multiphase coupling. This conservation law is solved for each component of the vector of conserved variables (**U**). Since CMT-nek is based on discontinuous Galerkin spectral element method, the computational domain is partitioned into hexahedral elements (figure 3). Each element is discretized by N^3 Legendre Gauss-Lobatto grid points. Inside each element the conserved variables are approximated by tensor product of N^{th} order polynomials in three co-ordinate directions.

CMT-nek solves the variational formulation of the conservation law shown above. In general the variational formulation for discontinuous Galerkin method can be written as

$$\sum_{k=1}^{N} \left\{ \int_{\Omega_k} \phi \frac{\partial \mathbf{U}}{\partial t} d\mathbf{x} + \int_{\Omega_k} \phi \nabla \cdot f(\mathbf{U}, \nabla \mathbf{U}) d\mathbf{x} - \int_{d\Omega_k} (f - f^*) \cdot \hat{n} \phi d\mathbf{A} = \int_{\Omega_k} R d\mathbf{x} \right\}, \qquad (2)$$

where ϕ is the test function, Ω_k is the k^{th} hexahedral element and $d\Omega_k$ is the surface of k^{th} hexahedral element. In the above equation all the terms are volume integrals apart from the third term on the left-hand-side which is a surface integral. In the surface integral term f^* is the numerical flux which is informed by the physics of compressible flow and analogous to their role in finite volume methods. These terms in the variational formulation are evaluated using quadrature rules. In summary, CMT-nek involves computing the (1) source terms, (2) flux divergence, and (3) numerical flux for all the elements.

C. Preparing CMT-nek for future machines

The upcoming revolutionary transformation of computer architectures and systems to exascale poses a significant challenge for this project. We must craft numerical algorithms and computational/software techniques for CMT-nek to target systems as yet undefined, while concurrently advancing predictive simulation science.

Our strategy is to first perform state-of-the-art CMT-nek simulations using our existing software on present-day and near-future petascale platforms (as they become available) to demonstrate progressive improvements in the predictive capability of the demonstration problem. Second, we plan to position ourselves to extract maximum performance on futuristic exascale architectures through a co-design effort that combines fast and scalable Behavioral Emulation leveraging the unique Novo-G facility at the NSF-supported UF Center for High-Performance Reconfigurable Computing (CHREC) to emulate and evaluate a series of candidate exascale architectures [13].

Both performance optimization on current architectures as well as evaluation of promising architectures requires extensive understanding of and experimentation with CMT-nek and its kernels. It is hard for computer engineers and computer scientists to understand and work with the entire application code, especially since it is constantly being modified. Hence, based on our early experiences with performance optimization and architecture simulation we realized the need for a smaller version of the CMT-nek application, a mini-app, that represents the key kernels and performance features but which is simple enough to understand and modify as necessary for performance, power, energy analysis and architecture simulations.

IV. CMT-bone Mini-app

CMT-bone is a mini-app for CMT-nek that will be used to evaluate CMT-nek's performance on near-future and promising exascale architectures. CMT-bone will also be used for analyzing and optimizing existing algorithms for current architectures. To ensure that the mini-app represents the parent application in its most mature state, CMT-bone and CMT-nek development will happen concurrently. As new capabilities are added and tested in CMT-nek, their abstractions will be added in CMT-bone.

CMT-bone will adequately represent CMT-nek if the operations involved in computing source terms, flux divergence, and numerical flux, described in the CMT-nek model, are abstracted into CMT-bone. Flux divergence term is a dot product of gradient operator and the flux vector. Numerically,

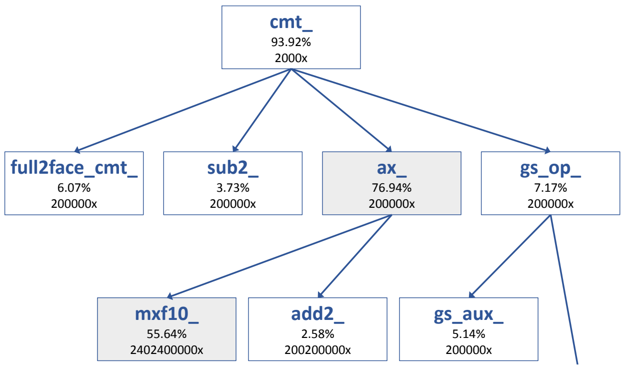

Fig. 4. Partial CMTBone call graph and execution profile on 8-core Intel i5-2500 processor obtained using gprof

the flux divergence can be abstracted into matrix multiplication operations where the derivative matrix of size *(N, N)* operates over a 3D data *(N, N, N, Nel)*. Here *N* is the degree of polynomial approximation for conserved variables and *Nel* is the total number of elements in the computational domain. The numerical flux term is evaluated on the surface of the elements which involves surface data exchange between nearest neighbors. Since the latest version of CMT-nek has limited mutiphase coupling, the source terms in the conservation law are set to zero. Thus, the current version of CMT-bone abstracts CMT-nek behavior as matrix-multiplication and nearest neighbor surface data exchanges to represent the flux divergence term and the numerical flux term respectively.

Therefore, the key kernels in the current version of CMT-bone are the matrix-multiplication of derivative operators and 3D data and the nearest-neighbor data exchanges. Key application parameters that affect the execution time are degree of the polynomial $N-1$, number of elements per processor Nel, and the number of MPI processes P used to run the simulation. The degree of polynomial determines the number of grid points in each element or the 3D volume data (cube of size N^3) and the size of the derivative operator (square matrix of size N^2).

Both single-node and multi-node CMT-bone profiling is needed to measure the relative impact of various CMT-bone kernels on application performance and identify candidates for potential optimization. Extensive performance measurement is also necessary for building robust performance models for architecture simulation. As shown in figure 4, the majority of application time is spent in derivative calculation ($ax\_$ routine, for flux divergence). The partial call graph was obtained using the gprof utility for CMT-bone with 8 MPI processes on a 4-core Intel i5-2500 processor running at 3.30GHz with hyper-threading enabled. We observed a similar profile on different processors and large cluster runs. Since it is the most expensive computation intensive portion of the application, optimizing this kernel can have significantly improve the execution time of CMT-bone. In section V, we present preliminary results of applying loop transformation techniques to this kernel.

Besides the derivative calculation, the other key kernels are $full2face\_cmt$ and $gs\_op\_$. The first routine creates an array of surface data, that needs to be transferred to the neighbors, from the volume data for each element. The second routine handles the communication between MPI processors using a communication strategy selected from a library of gather-scatter algorithms. Understanding the size, frequency, average distance etc. of these communication routines is important for improving the scaling behavior of the software as well as for developing robust performance models. In section VI, we present a brief analysis of CMT-bone MPI communication calls that is being used to guide the network modeling and simulation effort.

V. Optimizing spectral element solver

As we saw in the previous section, the majority of application time is spent in small matrix multiplication used mainly for computing partial derivatives in the spectral element solver and for dealiasing reference elements, where an element is first mapped to a finer mesh and later mapped back to the regular mesh. The elements and derivative operator matrices are fairly small, with N ranging between 5 and 25, where N is the number of grid points along any one direction in a cubic reference element. The derivative calculation is an $O(N^4)$ operation. Since derivative computation is a key operation in spectral element method, we profiled the derivative computing kernel separately for the partial derivatives along the three co-ordinate directions (r, s, t).

Figure 5 shows the performance statistics of the derivative computing kernel obtained on AMD Opteron 6378 platform operating at 2.4GHz. The size of both L1 data cache and L1 instruction cache is 48KB and the code was compiled using gfortran. The kernel was setup to process 1563 elements (Nel) and the simulation was run through 1000 time steps. The total instruction and total cycles statistics were collected using PAPI [14].

The derivative computing kernel in CMT-bone inherits the loop transformation techniques of loop fusion and loop unroll from CMT-nek which in turn inherits it from Nek5000. Specifically, the innermost for-loop is completely unrolled for all the three derivatives, and the two outermost for-loops are

Derivatives	Runtime (seconds)	Total instructions	Total Cycles
dudt	4.89	1,158,978,395	762,267,174
dudr	8.60	2,402,189,302	1,355,354,404
duds	9.45	2,595,078,699	1,468,462,190

Fig. 5. Execution time for different partial derivative calculations with loop transformation techniques applied to the compute kernels

Derivatives	Runtime (seconds)	Total instructions	Total Cycles
dudt	11.3	3,219,865,483	1,695,229,754
dudr	8.89	2,428,697,316	1,394,120,803

Fig. 6. Execution time for partial derivative calculations using a basic implementation

fused for the derivatives with respect to r and t. Both loop fusion and loop unroll allow better vectorization of the code leading to a smaller number of total instructions processed and hence faster runtime. Figure 6 shows the runtimes for a basic implementation of computing partial derivatives along r and t. In the basic implementation, optimizations such as loop fusion or unroll, are not applied. Comparison of figure 5 ad figure 6 shows how the loop optimization techniques improve performance in CMT-bone. For $dudt$ and $dudr$ CMT-bone is 2.31 and 1.03 times faster with loop optimizations that the basic implementation without these loop optimizations. For $duds$, there was no noticeable improvement over the basic implementation. This is primarily because the array access pattern while computing the matrix mutiplication forbids implementation of loop fusion which lowers the potential of vectorization. Further, unrolling the innermost for-loop in $duds$ computation does not improve performance since the benefit of vectorization is offset by a large number of cache misses due to poor data locality.

VI. Mini-app Communication profiling

Since CMT-nek inherits the MPI communication strategies from Nek5000, CMT-bone does too. The major communication routines in CMT-bone are vector reductions, nearest-neighbor exchanges, and generalized all-to-all. Nearest-neighbor exchanges are used to ensure continuity and take place using a specialized gather-scatter library. The initialization step in CMT-bone involves setting up the communication groups and determining which of the three algorithms (pairwise exchange, crystal router, all_reduce) is used by the application for the nearest neighbor exchanges. In addition, spectral element coefficients are stored redundantly (and locally) on each processor instead of maintaining a global matrix and each processor is given index sets containing the global ids of the elements using gs_setup. This requires a discovery phase using all-to-all communication to identify for every global index i on processes p, all the processes q that also have i.

At the beginning of each CMT-nek and CMT-bone simulation, three gather-scatter methods are evaluated to determine which one performs the best for the given problem setup and machine. These three exchange strategies are : (1) pairwise exchange, (2) crystal-router, and (3) all_reduce onto a big vector. Figure 7 compares the results of these tests for both CMT-bone and Nekbone mini-apps for the same problem setup on Compton machine, a 42-node ASC cluster at Sandia National Laboratories, Albuquerque, NM with two 8-core Sandy Bridge Xeon E5-2670 connected with Mellanox Infiniscale IV QDR.

Even though the application parameters for both CMT-bone and Nekbone are the same, since the two mini-apps are meant

Setup:

Number of processors: 256	Dimensions = 3
Number of elements per process = 100	Processor Distribution (x,y,z) = 8, 8, 4
Total elements = 25600	Element Distribution (x,y,z) = 40, 40, 16
Number of gridpoints per element = 10	Local Element Distribution (x,y,z) = 5, 5, 4

Mini-app	All-to-all method	Time (avg) seconds	Time (min) seconds	Time (max) seconds
CMT-bone	pairwise exchange	0.000318934	0.000244498	0.000353503
	crystal router	0.000799977	0.000788808	0.000808311
Nekbone	pairwise exchange	0.000638981	0.000557685	0.000685811
	crystal router	0.000663779	0.000657296	0.000669909

Fig. 7. Comparison of the two communication algorithms candidates (pairwaise exchange and crystal router) used in CMT-bone and Nekbone on 256 MPI processes and a single timestep

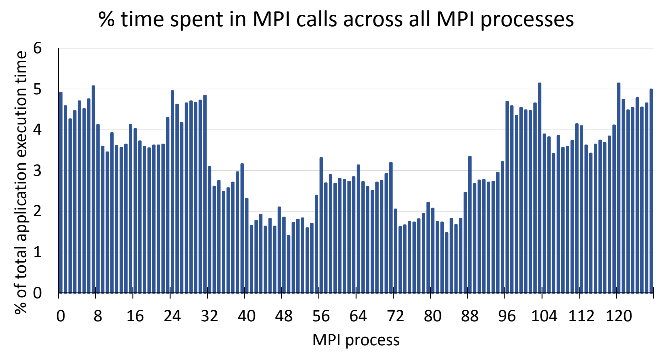

Fig. 8. Time spent by each CMT-bone MPI process in communication routines

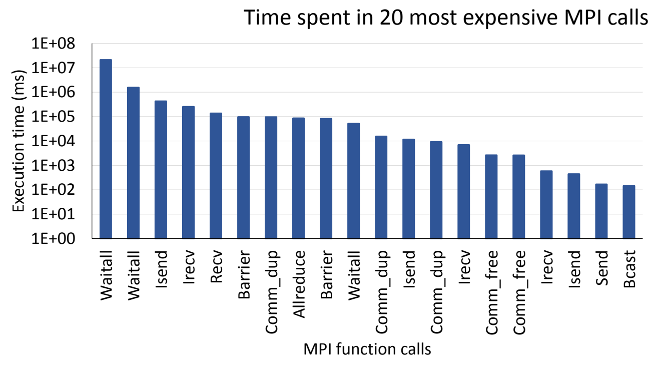

Fig. 9. Time spent in the top 20 MPI calls in CMT-bone

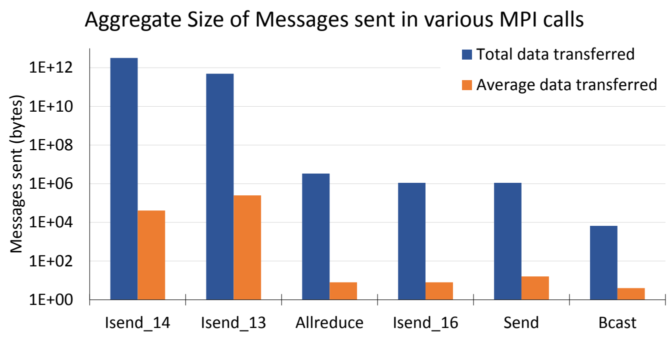

Fig. 10. Total and average size of messages sent in the most frequently called MPI calls

to represent different physical problems, we observe that different methods are chosen for communication. All_reduce is too expensive for both the mini-apps for this problem setup. While Nekbone run uses an implementation of the crystal router algorithm, originally developed for all-to-all communication in hypercubes, CMT-bone execution run uses a simple pairwise exchange strategy. All-to-all communication using the crystal router exchange is guaranteed to complete in $log_2 P$ stages. While this routine has not been used in any of our CMT-bone test runs with different system and problem sizes, as new kernels get added to the mini-app and the problem setup changes, it is possible that crystal router may be used instead of pairwise exchange. This observation is of importance to both performance optimization and performance modeling efforts.

A further breakdown of communication time is presented in figure 8. This plot generated from data collected using mpiP shows the percentage of total execution time spent in MPI routines by each MPI rank. mpiP is a lightweight, task-local, and scalable profiling library for MPI applications [15]. The plot in figure 9 shows the time spent in the twenty most expensive MPI calls. From this plot we see that a large amount of time is spent in MPI_Wait for synchronization. It demonstrates the need for better load balancing in the applicaiton. It is also an important observation for the computer engineers who are interested in developing performance models and simulating the application behavior. In order to model this network behavior, we would need to model time spent in MPI_Wait which is hard to do with analytical models and may require timing-based simulations. To perform network simulations we also need appropriate latency and bandwidth models for the machines and data transfer characteristics for the application. Figure 10 shows the total and average size of messages transferred in the most expensive MPI calls.

VII. Conclusions and Future Work

Mini-apps can be an important aid for early architecture and algorithm design space exploration. In this paper we have introduced a new mini-app - CMT-bone - which represents key communication and computation kernels in the compressible multiphase turbulence application CMT-nek. By developing CMT-bone in parallel with CMT-nek, application developers can work with system architects and performance engineers to design specialized architectures and also evaluate alternate algorithms which better utilize the architectural features. CMT-bone has allowed faster performance analysis and optimization of key kernels on current systems and identification of key performance characteristics that will be needed in the future systems.

A key focus in the near term will be extensive validation of the relationship between CMT-bone and CMT-nek on different architectures based on performance metrics. As mentioned earlier, CMT-bone will be developed in parallel with the CMT-nek code. In the future, as CMT-nek is extended to include discontinuous Galerkin methods, real gas equations, adaptive grid density, adaptive time stepping etc., corresponding key kernels will also be added to CMT-bone. There is also a possibility of making available single-core codes representative of the key computation kernels inside CMT-nek. Major version updates and releases of CMT-bone will occur annually.

Acknowledgment

This work is partially supported by the I/UCRC Program of the National Science Foundation under Grant Nos. EEC-0642422 and IIP-1161022 and the U.S. Department of Energy, National Nuclear Security Administration, Advanced Simulation and Computing Program, as a Cooperative Agreement under the Predictive Science Academic Alliance Program, under Contract No. DE-NA0002378. We would like to acknowledge Sandia National Laboratories (SNL), Albuquerque, NM for providing us access to Compton, an ASC testbed at SNL for communication analysis. We would also like to thank Chris Hajas and Kasim Alli for assisting in the mini-app performance analysis and for their feedback on the paper.

References

[1] M.A. Heroux, D.W. Doerfler, P.S. Crozier, J.M. Willenbring, H.C. Edwards, A. Williams, M. Rajan, E.R. Keiter, H.K. Thornquist, R.W. Numrich, *Improving Performance via Mini-applications*, Technical Report, SAND2009-5574, 2009.

[2] Project for Characterization of the DOE Mini-apps, http://portal.nersc.gov/project/CAL/doe-mini-apps.htm

[3] M.A. Heroux, *Design issues for numerical libraries on scalable multi-core architectures*

[4] Proxy-apps for thermal hydraulics URL: https://cesar.mcs.anl.gov/content/software/thermal_hydraulics

[5] P.F. Fischer, J.W. Lottes and S.G. Kerkemeier, *Nek5000*, http://nek5000.mcs.anl.gov

[6] *The CESAR Codesign Center: Early Results*, Technical paper for Exascale Research Conference, April 2012.

[7] R.F. Barrett, S. Borkar, S.S. Dosanjh, S.D. Hammond, M.A. Heroux, X. S.Hu, J.Luitjens, S.G. Parker, J. Shalf, and L. Tang, *On the role of co-design in high performance computing.* vol 24 (2013): 141-155.

[8] S.S. Dosanjh, R.F. Barrett, D.W. Doerfler, S.D. Hammond, K.S. Hemmert, M.A. Heroux, P.T. Lin, K.T. Pedretti, A.F. Rodrigues, T.G. Trucano, J.P. Luitjens , *Exascale design space exploration and co-design*, Future Generation Computer Systems, Volume 30, January 2014, Pages 46-58, ISSN 0167-739X

[9] R.F. Barrett, P.S. Crozier, D.W. Doerfler, M.A. Heroux, P.T. Lin, H.K. Thornquist, T.G. Trucano, C.T. Vaughan, *Assessing the role of mini-applications in predicting key performance characteristics of scientific and engineering applications*, Journal of Parallel and Distributed Computing, Volume 75, January 2015, Pages 107-122, ISSN 0743-7315

[10] I. Karlin, J. McGraw, J. Keasler, B. Still, *Tuning the LULESH Mini-app for Current and Future Hardware*, NECDC Proceedings, NECDC 2012, Livermore, CA, United States

[11] Paul F. Fischer, James W. Lottes, Stefan G. Kerkemeier, *nek5000 Web page*, 2008

[12] J. M. Powers, *Two-phase viscous modeling of compaction of granular materials*, Phy. Fluids, Volume 16, 2004

[13] N. Kumar, C. Pascoe, D. Rudolph, H. Lam, A. George, G. Stitt, *Multi-scale, Multi-objective, Behavioral Modeling & Emulation of Extreme-scale Systems* , Workshop on Modeling and Simulation of Systems and Applications, August 13-14, 2014, Seattle, WA

[14] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci, *A Portable Programming Interface for Performance Evaluation on Modern Processors*, International Journal of High Performance Computing Applications, Volume 14, Number 3, Pages 189204, August 2000.

[15] J. Vetter, J & C. Chambreau, *mpiP: Lightweight, Scalable MPI Profiling*, 2004. URL http://mpip. sourceforge.net.

[16] A.F. Rodrigues, K.S. Hemmert, B.W. Barrett, C. Kersey, R. Oldfield, M. Weston, R. Risen, J. Cook, P. Rosenfeld, E. Cooperballs, & B. Jacob, *The structural simulation toolkit*, ACM SIGMETRICS Performance Evaluation Review, 2011, 38(4), 37-42.