R&D 100 • 2013

SAND2013-2466P

Mantevo Suite 1.0

Exceptional service in the national interest

Sandia
National
Laboratories

Mantevo Suite 1.0

1. Organization Contact Information

For associated video,
CLICK HERE

A. Submitter/Product Developer

Organization:

Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185
www.sandia.gov

Submitter:

Michael A Heroux
18125 Kreigle Lake Road
Avon, MN 56310
(320) 845-7695
Fax: (320) 845-7846
maherou@sandia.gov
www.cs.sandia.gov/~maherou

B. Principal Developer Organization

Principal Developer Organization

Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185
www.sandia.gov

Other Supporting Organizations

- LANL and LLNL co-developed CoMD, one of the miniapps in the Mantevo Suite.
- AWE developed CloverLeaf, one of the miniapps in the Mantevo Suite.
- NVIDIA Corp. developed GPU-Optimized derivative version of MiniFE package.

Jamal Mohd-Yusof
Los Alamos National Laboratory (LANL)
P.O. Box 1663
Los Alamos, NM 87545
(505) 667-5061
jamal@lanl.gov
www.lanl.gov

David Richards
Lawrence Livermore National Laboratory (LLNL)
P.O. Box 808
Livermore, CA 94551
(925) 424-5140
Fax: (925) 422-1370
richards12@llnl.gov
www.llnl.gov

Andy Herdman
Atomic Weapons Establishment (AWE)
AWE, Aldermaston
Reading, Berkshire RG7 4PR
United Kingdom
0118 981 4111
Fax: 0118 981 5320
Andy.Herdman@awe.co.uk
awe.co.uk

Justin Luitjens
NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
(408) 486-2000
Fax: (408) 486-2200
jluitjens@nvidia.com
nvidia.com

C. Principal (Primary) Developer

Michael A. Heroux
Distinguished Member of Technical Staff
Sandia National Laboratories
18125 Kreigle Lake Road
Avon, MN 56310
(320) 845-7695
Fax: (320) 845-7846
maherou@sandia.gov

Richard F. Barrett (Co-lead)
Principal Member of Technical Staff
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM
87185-1319
(505) 845-7655
Fax: (505) 284-2518
rfbarre@sandia.gov

James M. Willenbring (Project Coordinator)
Member of Technical Staff
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM
87185-1320
(320) 685-0454
Fax: (320) 685-0454
jmwille@sandia.gov

2. Product Information

A. Product name
Mantevo Suite 1.0

B. Generic Description of Product
Computer Software

C. Product photo

Figure 1. Mantevo Suite 1.0

3. Executive Summary

Computing hardware enables rapid information processing via sequential execution, but clock speeds have stalled. Future performance gains come almost solely from running sets of instructions concurrently, precipitating fundamental changes for all computer components, making co-design (collaborative, simultaneous development of all system components) essential.

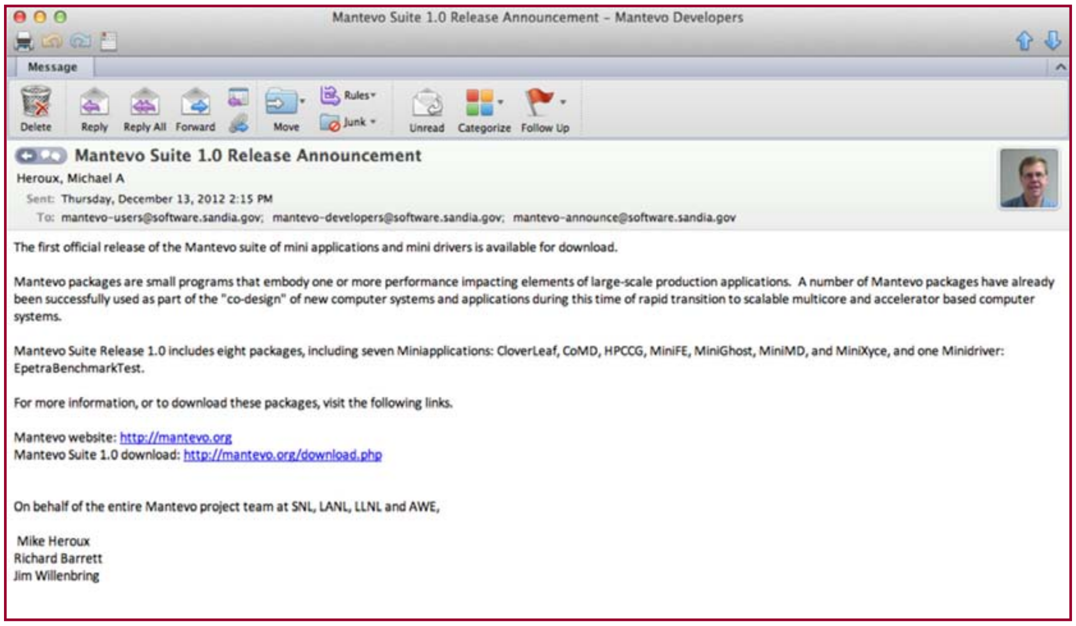
Miniapps have emerged as central components of co-design. Sandia's Mantevo Suite 1.0 is an integrated collection of small software programs (miniapps) whose performance characteristics model full-scale applications, yet require only a fraction of the lines of code, making miniapps easier to study, design, and rewrite.

The Mantevo project pioneered the miniapp concept, and the Mantevo Suite 1.0 is the first integrated collection of full-featured miniapps. Miniapps are essential tools to explore complex design spaces because they are exceptional performance predictors of full applications and allow earlier, informed design decisions. Major companies (e.g., Intel, IBM, NVIDIA, AMD, Cray), universities and laboratories use miniapps as essential tools.

Miniapps are essential to the computer system co-design process.

4. Introduction Date

December 13, 2012: Mantevo Suite 1.0 Release Announcement to Mantevo users and developers.



Mantevo Suite 1.0 Release Announcement
Heroux, Michael A
Sent: Thursday, December 13, 2012 2:15 PM
To: mantevo-users@software.sandia.gov; mantevo-developers@software.sandia.gov; mantevo-announce@software.sandia.gov

The first official release of the Mantevo suite of mini applications and mini drivers is available for download.

Mantevo packages are small programs that embody one or more performance impacting elements of large-scale production applications. A number of Mantevo packages have already been successfully used as part of the "co-design" of new computer systems and applications during this time of rapid transition to scalable multicore and accelerator based computer systems.

Mantevo Suite Release 1.0 includes eight packages, including seven Miniapplications: CloverLeaf, CoMD, HPCCG, MiniFE, MiniGhost, MiniMD, and MiniXyce, and one Minidriver: EpetraBenchmarkTest.

For more information, or to download these packages, visit the following links.

Mantevo website: http://mantevo.org
Mantevo Suite 1.0 download: http://mantevo.org/download.php

On behalf of the entire Mantevo project team at SNL, LANL, LLNL and AWE,

Mike Heroux
Richard Barrett
Jim Willenbring

Figure 2. Mantevo Suite 1.0 Release Announcement to Mantevo users and developers: December 13, 2012.

5. Previous R&D 100 Entry

This product has not been entered previously in the R&D 100 awards competition.

6. Product Price

The Mantevo Suite 1.0 is freely available as open source software at mantevo.org.

7. Patents

None.

8. Product Description

A. What Does the Mantevo Suite Do?

Application performance is determined by a combination of many choices: hardware platform, runtime environment, languages and compilers used, algorithm choice and implementation, and more. In this complicated environment, we find that the use of miniapps (small self-contained proxies for real applications) is an excellent approach for rapidly exploring the parameter space of all these choices. Furthermore, use of miniapps enriches the interaction between application, library, and computer system developers by providing explicit functioning software and concrete performance results that lead to detailed, focused discussions of design trade-offs, algorithm choices, and runtime performance issues.

Production-quality science and engineering applications are typically large, complicated and full-featured software products. As a result, they tend to be challenging to port to new computer platforms and require a well-trained user to do so. Although benchmarking of these applications on new platforms is essential as part of the design and implementation of a new computer system, the scope of this benchmarking is necessarily limited by the complexity of the software product, not to mention its demand for a full scope of system features that are only available after a new computer system reaches its near-production capabilities.

Characteristics that impact performance should be understood as early as possible in the analysis and design of new computers. Furthermore, it is often the case that there are multiple ways to design and implement the algorithms used in an application, and the choice can have a dramatic impact on application performance.

To address these needs, our recent work in application performance analysis takes advantage of two important properties of many applications. (i) Although an application may have one million or more source lines of code, performance is often dominated by a very small subset of lines. (ii) For the remaining code, these applications often contain many physics models that are mathematically distinct but have very similar performance characteristics.

Although an application may have one million or more source lines of code, performance is often dominated by a very small subset of lines.

To exploit the properties listed above, we have developed a growing collection of miniapps. Miniapps take advantage of the above two application properties by encapsulating only the most important computational operations, and consolidating physics capabilities that have the same performance profiles. The large-scale application developer, who is tasked with developing the miniapp, guides the decisions, resulting in a code that is a small fraction of the original application size, yet still captures the primary performance behavior.

Mantevo focuses on developing tools to accelerate and improve the design of high performance computers and applications by providing application and library proxies to the high performance computing community.

B. Principal Applications of Mantevo Suite

The Role of Miniapps.

Mantevo miniapps facilitate rapid and effective design decisions for the development of new high-performance computer systems by giving computer system developers — from those developing new memory systems, processor architectures, operating and runtime systems, languages, compilers, and the applications themselves — a concrete, executable performance proxy for a full application.

Performance proxies for full applications have been around for decades. Benchmarks and kernels in particular have received much attention. Miniapps are unlike previous efforts in that the explicit intent is to support co-design. Miniapps balance fidelity to the full application against ease-of-use, and balance providing a concrete description of how to perform computations against the flexibility to redesign and rewrite the software.

C. How Mantevo Suite Benefits the Market It Serves

Why we develop miniapps.

Not all applications require miniapps – some are small and accessible in their own right – but others are simply too large and complex to work with directly in early design studies. For example, Charon, a production multi-physics code, has more than 400,000 lines of code and 40 third-party library (TPL) dependencies, works only with message passing parallelism (MPI) and double-precision data, requires significant experience to build, and is export controlled. In contrast, MiniFE, a miniapp performance proxy for Charon, has 6,500 lines of code, does not have any required TPL dependencies, and works with MPI and many other parallel environments, including Pthreads, OpenMP, Threading Building Block (TBB), and CUDA. It supports multiple precisions and is open source and trivial to build. MiniFE has been used in dozens of performance design studies, informing numerous decisions.

D. Other Applications of Mantevo Suite

Uses for miniapps.

Miniapps provide a category of tools that help in the following situations:

- *Interaction with external research communities:* Miniapps are open source software, in contrast to many restricted access production applications.
- *Simulators:* Miniapps are the right size for use in simulated environments, supporting the study of processor, memory, and network architectures.
- *Early node architecture studies:* Scalable system performance is strongly influenced by the processor node architecture. Processor nodes are often available many months before the complete system. Miniapps enable an opportunity to study node performance very early in the design process.
- *Network scaling studies:* Miniapps are easily configured to run on any number of processors, providing a simple tool to test network scalability. Although not a replacement for production applications, miniapps can provide early insight into scaling issues.
- *New language and programming models:* Miniapps can be refactored or completely rewritten in new languages and programming models. Such working examples are a critical resource in determining if and how to rewrite production applications.
- *Compiler tuning:* Miniapps provide a focused environment for compiler developers to improve compiled code.

Miniapps are designed to be useful tools throughout the co-design space and have gained popularity among many of the Department of Energy (DOE) Advanced Scientific Computing Research and National Nuclear Safety Administration co-design centers. Unlike a benchmark, the output of a miniapp is information, which must be interpreted within an often-subjective context. Unlike a simplified model application, which is designed to capture physics behavior, miniapps are designed to capture some key performance issue in the full application. Unlike a skeleton application, which is designed for focusing on interprocess communication perhaps involving a “fake” computation, miniapps create a meaningful context in which to explore the key performance issues. Ideally miniapps are developed and owned by application code teams, with the assistance of other experts. Miniapps can be modified or even completely rewritten. Averaging 5,000 source lines of code, miniapps enable unconstrained modification.

Miniapps can be modified or even completely rewritten. Averaging 5,000 source lines of code, miniapps enable unconstrained modification.

9. Technology Description

A. How the Mantevo Suite Works

Working with Mantevo miniapps.

Mantevo miniapps are freely available software products, accessible from mantevo.org as C++, C, and Fortran source code, along with descriptions of capabilities and instructions for compiling and running relevant test cases. The base implementations include support for:

- Sequential execution
- Shared memory execution using OpenMP
- Distributed memory execution using MPI

In the hands of users, they are compiled for the target system or simulator environment and then executed over a range of problem sizes and configurations. In most cases, users will also modify the source code to better match the target platform (e.g., port to CUDA for GPUs [graphics processing unit]) and study the impact of those changes.

The results of these experiments are used to help system designers (e.g., should we increase register state on the next generation of GPUs?), application developers (e.g., are GPUs a good platform?), programming model developers (e.g., how can we make it easier to rewrite Charon for GPUs?) and users (e.g., how fast will Charon be in the future?).

B. What scientific theories are involved for Mantevo Suite?

Miniapps are small but predictive.

The fundamental principle underpinning miniapps is that performance profiles of applications can be described in terms of computational and data movement *patterns*. Although an application may have many modeling capabilities, these underlying patterns are repeated many times. For example, Charon uses the finite element method on hexahedral meshes to compute approximations to many physics models. In contrast, MiniFE computes very simple physics (heat diffusion) on synthetic hexahedral meshes, but the computational and data movement patterns are very similar. Charon uses a variety of preconditioned iterative solvers to solve the subsequent global finite element problem. MiniFE solves a simple problem using simple conjugate gradients, which have most of the patterns present in Charon's solvers.

Because MiniFE contains most of the same patterns as Charon, its performance behavior is very similar to Charon. Furthermore, transformations that change how MiniFE behaves will also be applicable to Charon, or any similar application. Extensive validation exercises show the strong correlation.

C. Building Blocks of Your Technology?

Mantevo common look and feel policies.

All miniapps share some common requirements despite modeling very different types of problems. Processes and tools for building, running, and collecting data can be leveraged across all miniapps. The Mantevo common look and feel policy supports effective use of an ever-growing collection of miniapps. All Mantevo miniapps share these features:

- *Four basic builds.* We also distribute any derived version (e.g., the CUDA version of MiniFE). All miniapps use a simple configuration and build system, which is essential when working in prototype environments and with simulators.
- *Common I/O format.* Mantevo provides utility functions to read and write YAML format. YAML is a *de facto* standard, human-readable text format that supports XML conversion. By using YAML, we can record Mantevo output directly into databases for later analysis. Each execution of a Mantevo miniapp generates a time-stamped output file.
- *Co-Pylot post-processing tools.* Performance studies with Mantevo can generate too much data to analyze by hand. The Mantevo tool Co-Pylot accepts Mantevo YAML output files and stores each as a record in a MySQL

database. Co-Pylot also displays results as graphs and charts.
- *YAML email reflector.* Because YAML is text output, it can be sent by email. We have email lists that accept embedded YAML output for use with Co-Pylot.

New miniapps leverage the common tools that Mantevo provides, reducing development time and improving usability.

Current miniapps.

Mantevo Suite Release 1.0 contains the miniapps listed below. Some of these miniapps have been available individually before, but this suite release is the first official release and includes the common look and feel features.

- *CloverLeaf:* Solves the compressible Euler equations on a Cartesian grid, using an explicit, second-order accurate method.
- *CoMD:* A simple proxy for the computations in a typical molecular dynamics application. The reference implementation mimics that of SPaSM. In addition, we provide an OpenCL implementation, which allows testing on multicore and GPU architectures, with both array-of-structures and structure-of-arrays data layouts.
- *MiniFE:* A proxy application for unstructured implicit finite element codes. It is similar to HPCCG but provides a much more complete vertical covering of the steps in this class of applications. MiniFE also provides support for computation on multicore nodes, including Pthreads and Intel's TBB for homogeneous multicore and CUDA for GPUs.
- *HPCCG:* Intended to be the "best approximation to an unstructured implicit finite element or finite volume application in 800 lines or fewer."
- *MiniMD:* A simple proxy for the force computations in a typical molecular dynamics application. The algorithms and implementation used closely mimics these same operations as performed in LAMMPS.
- *MiniGhost:* A finite difference proxy application that implements a difference stencil across a homogenous three-dimensional domain.
- *MiniXyce:* A portable proxy of some of the key capabilities in the electrical modeling Xyce.

10. Product Comparison

A & B. Comparison Matrix

	Full Application	Challenge Problem	Benchmark	Mantevo Miniapp
Size and Complexity	Large (up to millions of lines of code). Composed of multiple languages. Dependent on third-party components. Relies on proprietary data. Sometimes export controlled.	Much smaller than full application, but can still be large. Typically self-contained. Some export controlled.	Usually smaller than miniapp. Easy to build.	Smaller than challenge problem. Self-contained. Common look and feel across all.
Scientific Fidelity	Full.	Attempts to solve a simplified but still realistic problem. Leads to unnecessary work when only interested in performance modeling.	Very little. Focus is on a static collection of kernels.	Can solve a simplified problem with synthetic data, but mostly concerned about minimizing code size while retaining necessary computational and data movement patterns.
Performance Modeling Fidelity	Full.	Fairly good, but for simplified problem. Fidelity should hold under transformations.	Focused on performance of current computational approach. Transformations are generally not permitted.	Fairly good. Extensive validation studies show modeling strengths and weaknesses, including understanding of effect of transformations.
Ease of Refactoring	Very difficult. Successful deployment to next generation systems can be a multi-year effort. Without some prototyping, likelihood of success is low.	Much easier than full application, but adhering to a realistic problem means there are many repeated computational patterns that must be redundantly refactored.	Generally not permitted or only within a very narrow scope.	Much easier than challenge problem. Only the essential patterns are present, so refactoring is as efficient as possible.
Ease of Generating and Analyzing Performance Results	Possible, but only with current application implementation. No ability to try new approaches on new systems or with new programming environments.	Fairly easy to generate results, but no uniform output format that makes analysis easy.	Very easy to generate and collect across many systems. Data typically collected on common website. Easily analyzed.	Very easy to generate. Generally collected across many systems and thoroughly analyzed. Use of YAML, XML, and interoperability with post-processing tools (e.g., Co-Pylot, spreadsheets), makes analysis easy.
Availability	Sometimes very limited or export controlled. Sometimes costly to purchase.	More accessible, but also sometimes export controlled.	Downloadable. In some cases a license fee of $250-$800.	Free download. Open source under the GNU LGPL license.

[1] *DARPA UHPC, Challenge Problem Hydrodynamics: LULESH, Version 1.0.1*

[2] *Innovative Computing Laboratory, University of Tennessee, HPC Challenge Benchmark, Version 1.4.2 AND NASA Advanced Supercomputing Division, NAS Parallel Benchmarks, Version 3.3*

Mantevo miniapps provide the right balance of size, complexity, fidelity, ease of refactoring, generating and analysis of results, and availability for enabling the design of next generation computing environments.

C. Describe how your product improves upon competitive products or technologies.

Mantevo miniapps provide the right balance of size, complexity, fidelity, ease of refactoring, generating and analysis of results, and availability for enabling the design of next generation computing environments. In this time of rapid change in computing, Mantevo miniapps are an essential element of the co-design process.

- *Ease of refactoring.* Miniapps are designed and intended to be refactored. This is perhaps the most important feature of miniapps. The success of future high-end computing requires the redesign of all components, including the applications themselves.
- *Size and complexity.* Miniapps are the smallest possible performance proxies for full applications that still contain the most important performance-impacting features.
- *Fidelity.* Miniapp designs recognize that fidelity to computing the full application problem is not necessary for accurate performance modeling. On the contrary, fidelity leads to extra work when refactoring source code for new systems.
- *Generating and analyzing performance results.* Miniapps are designed for easy generation of both human- and machine-readable results. Use of YAML and XML formats enable the use of almost any post-processing tools, including Sandia's Co-Pylot tools (which permit sending of miniapp results to an email address for automatic insertion into a database) and standard spreadsheets and databases.
- *Availability.* Miniapps are free, open source downloads from the mantevo.org website.

D. Limitations of your product.

Miniapps have been controversial in the high-performance computing community. However, now that they have been successfully used in many settings, much of the criticism has subsided. Common concerns have been:

- *Miniapps are too simple to accurately represent real applications.* We get this comment from application developers who do not appreciate how frequently the same computational pattern appears in their full applications.
- *Miniapps are too specific.* This comment comes from people who think fundamentally new mathematical formulations are required. It is true that we assume a particular mathematical formulation. However, miniapps encode basic approaches for differential equations, particle physics, circuit simulation, and more. Although new mathematics may be required in some

application areas, the basic mathematical formulations encoded in Mantevo miniapps will continue to have importance for many years. Miniapps are intended to be refactored, allowing us to explore many new algorithms very quickly.

- *Miniapps are unnecessary overhead; I can work directly with my full application.* This criticism is true for some simpler applications that are fairly small and modular. In these settings, the full application can be refactored rather quickly and a miniapp is not necessary. However, in many application areas, the size and complexity of the full application requires one or more miniapps in order to explore future design choices.

11. Summary

The Mantevo project pioneered the miniapp collaboration model. Although many performance proxies have been available before, miniapps provide the right mix of ingredients to do rapid exploration of the design space for new supercomputers and applications. The release of the Mantevo Suite 1.0 provides access to a specific set of performance proxies that can be confidently used to guide design decisions by everyone involved in the development of the next generation of computing capabilities.

The entire computing community is facing a fundamental change in how to efficiently use new systems. Faster clock speeds have been the primary source for ever-increasing performance for decades. Clock speeds have stalled, but new, more challenging, opportunities are available via concurrency to once again enjoy regular performance improvements.

Supercomputing – the ability to perform 10^{15} (one million-billion) operations per second – allows us to do amazing things, and its importance cannot be overstated. From safer automobiles to enhanced oil recovery, new materials, better weather and climate forecasts, and breakthrough science results, supercomputing plays a unique role. However, because of stalled clock speeds, the next generation of supercomputers will be markedly different from the past. The new source for improved performance comes from high levels of concurrency using very different processor designs. This fundamental change in processor architecture requires changes to every other aspect of supercomputing.

Because supercomputers comprise an increasing number of components, the possibility of a component failing is dramatically rising. This also poses a daunting challenge for the next generation of supercomputers. How can applications remain resilient in the presence of intermittent failure?

With the number of components increasing and all components of supercomputing changing, co-design (the process of developing all components of the system simultaneously) becomes very important. No one can assume previous approaches will still be valid. In this era of co-design, miniapps have emerged as a central component for collaborative development.

12. Affirmation

By submitting this entry to *R&D Magazine* I affirm that all information submitted as a part of, or supplemental to, this entry is a fair and accurate representation of this product. I affirm that I have read the instructions and entry notes and agree to the rules specified in those sections.

[illegible]

Michael A. Heroux

For associated video,
CLICK HERE

APPENDICES

Appendix A: Development Team Information

Michael A. Heroux
Team Lead
Distinguished Member of Technical Staff
Sandia National Laboratories
18125 Kreigle Lake Road
Avon, MN 56310
(320) 845-7695
Fax: (320) 845-7846
maherou@sandia.gov

Richard R. Barrett
Team Co-Lead
Principal Member of Technical Staff
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1320
(505) 845-7655
Fax: (505) 284-2518
rfbarre@sandia.gov

James M. Willenbring
Member of Technical Staff
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1320
(320) 685-0454
Fax: (320) 685-0454
jmwille@sandia.gov

Jamaludin Mohd-Yusof
R&D Scientist
Applied Computer Science (CCS-7)
Mail Stop B287
Los Alamos National Laboratory
Los Alamos, NM 87545
(505) 665-2870
Jamal@lanl.gov

Appendix A: Development Team Information (cont.)

David F. Richards
Physicist
Lawrence Livermore National Lab
P.O. Box 808, L-045
Livermore, CA 94551
(925) 424-5140
Fax: (925) 422-2851
richards12@llnl.gov

Andy Herdman
High Performance Computing
+44 (0)118 98 55019
AWE, Aldermaston, Reading. RG7 4PR
Andy.Herdman@awe.co.uk

Justin Luitjens
NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
(408) 486-2000
Fax: (408) 486-2200
jluitjens@nvidia.com
nvidia.com

David Beckingsale
Performance Computing and Visualisation
Department of Computer Science
University of Warwick
+44 (0)247 65 22863
dab@dcs.warwick.ac.uk

Appendix A: Development Team Information (cont.)

James F. Belak
Senior Scientist
Lawrence Livermore National Laboratory
P.O. Box 808, L-045
Livermore, CA 94551
Fax: (925) 422-2851
Phone: 925-422-6061
Email: belak@llnl.gov

Mike Boulton
Microelectronics Group
Department of Computer Science
University of Bristol
mb8224@my.bristol.ac.uk

Wayne Gaudin
High Performance Computing
+44 (0)118 98 55028
AWE, Aldermaston, Reading. RG7 4PR
Wayne.Gaudin@awe.co.uk

Timothy C. Germann
R&D Scientist
Physics and Chemistry of Materials (T-1)
Mail Stop B214
Los Alamos National Laboratory
Los Alamos, NM 87545
(505) 665-9772
tcg@lanl.gov

Appendix A: Development Team Information (cont.)

Simon Hammond
Senior Member of Technical Staff
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1320
(505) 845-7897
Fax: (505) 845-7442
sdhammo@sandia.gov

Prof Stephen Jarvis
Performance Computing and Visualisation
Department of Computer Science
University of Warwick
+44 (0)247 65 24258
saj@dcs.warwick.ac.uk

Andrew Mallinson
Performance Computing and Visualisation
Department of Computer Science
University of Warwick
+44 (0)247 65 22863
acm@dcs.warwick.ac.uk

Simon McIntosh-Smith
Microelectronics Group
Department of Computer Science
University of Bristol
+44 (0)117 33 15324
simonm@compsci.bristol.ac.uk

Appendix A: Development Team Information (cont.)

Susan M. Mniszewski
R&D Scientist
Information Sciences (CCS-3)
Mail Stop B265
Los Alamos National Laboratory
Los Alamos, NM 87545
(505) 667 0790
smm@lanl.gov

Christopher Sewell
Staff Scientist
Mailstop B287
Los Alamos, NM 87545
Phone: 505-665-2563
Fax: 505-665-4939
csewell@lanl.gov

Sriram Swaminarayan
Group Leader
Applied Computer Science, CCS-7
Mail Stop B287
Los Alamos National Laboratory
Los Alamos, NM 87545
(505) 667-8647
sriram@lanl.gov

Appendix B: Marketing & Media Information

1. Marketing

Point of Contact for exhibits, banquet and publicity

Carol Adkins

Director, Research & Development, Science & Engineering

Sandia National Laboratories

P.O. Box 5800

Albuquerque, New Mexico 87185-0887

505-845-9119

Fax: 505-844-1583

cladkin@sandia.gov

2. Media

Point of Contact for media and editorial inquires

Carol Adkins

Director, Research & Development, Science & Engineering

Sandia National Laboratories

P.O. Box 5800

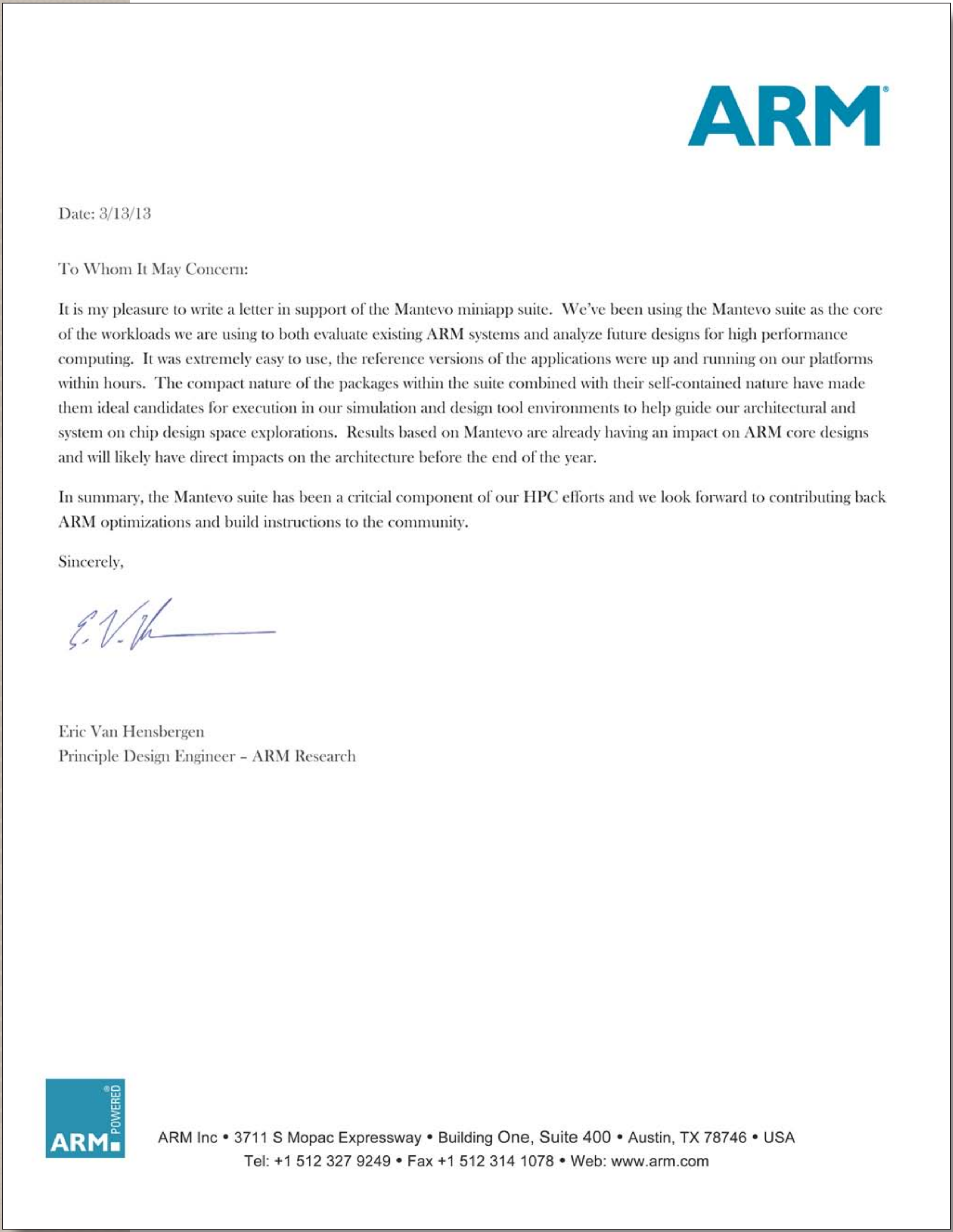
Albuquerque, New Mexico 87185-0887

505-845-9119

Fax: 505-844-1583

cladkin@sandia.gov

Appendix C: Letters of Support



ARM

Date: 3/13/13

To Whom It May Concern:

It is my pleasure to write a letter in support of the Mantevo miniapp suite. We've been using the Mantevo suite as the core of the workloads we are using to both evaluate existing ARM systems and analyze future designs for high performance computing. It was extremely easy to use, the reference versions of the applications were up and running on our platforms within hours. The compact nature of the packages within the suite combined with their self-contained nature have made them ideal candidates for execution in our simulation and design tool environments to help guide our architectural and system on chip design space explorations. Results based on Mantevo are already having an impact on ARM core designs and will likely have direct impacts on the architecture before the end of the year.

In summary, the Mantevo suite has been a criticial component of our HPC efforts and we look forward to contributing back ARM optimizations and build instructions to the community.

Sincerely,

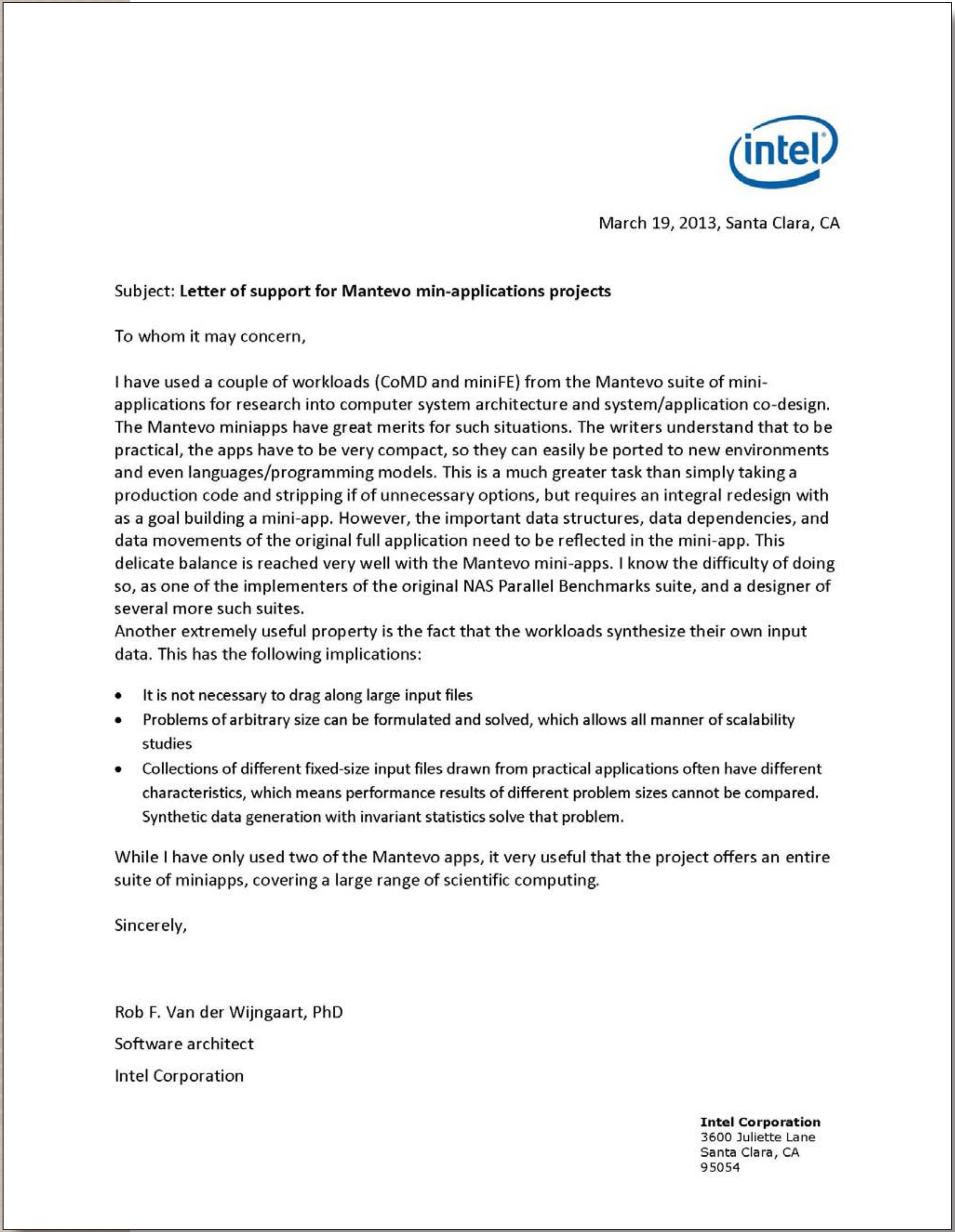
[illegible]

Eric Van Hensbergen
Principle Design Engineer – ARM Research

ARM POWERED

ARM Inc • 3711 S Mopac Expressway • Building One, Suite 400 • Austin, TX 78746 • USA
Tel: +1 512 327 9249 • Fax +1 512 314 1078 • Web: www.arm.com

Appendix C: Letters of Support (cont.)



March 19, 2013, Santa Clara, CA

Subject: **Letter of support for Mantevo min-applications projects**

To whom it may concern,

I have used a couple of workloads (CoMD and miniFE) from the Mantevo suite of mini-applications for research into computer system architecture and system/application co-design. The Mantevo miniapps have great merits for such situations. The writers understand that to be practical, the apps have to be very compact, so they can easily be ported to new environments and even languages/programming models. This is a much greater task than simply taking a production code and stripping if of unnecessary options, but requires an integral redesign with as a goal building a mini-app. However, the important data structures, data dependencies, and data movements of the original full application need to be reflected in the mini-app. This delicate balance is reached very well with the Mantevo mini-apps. I know the difficulty of doing so, as one of the implementers of the original NAS Parallel Benchmarks suite, and a designer of several more such suites.
Another extremely useful property is the fact that the workloads synthesize their own input data. This has the following implications:

- It is not necessary to drag along large input files
- Problems of arbitrary size can be formulated and solved, which allows all manner of scalability studies
- Collections of different fixed-size input files drawn from practical applications often have different characteristics, which means performance results of different problem sizes cannot be compared. Synthetic data generation with invariant statistics solve that problem.

While I have only used two of the Mantevo apps, it very useful that the project offers an entire suite of miniapps, covering a large range of scientific computing.

Sincerely,

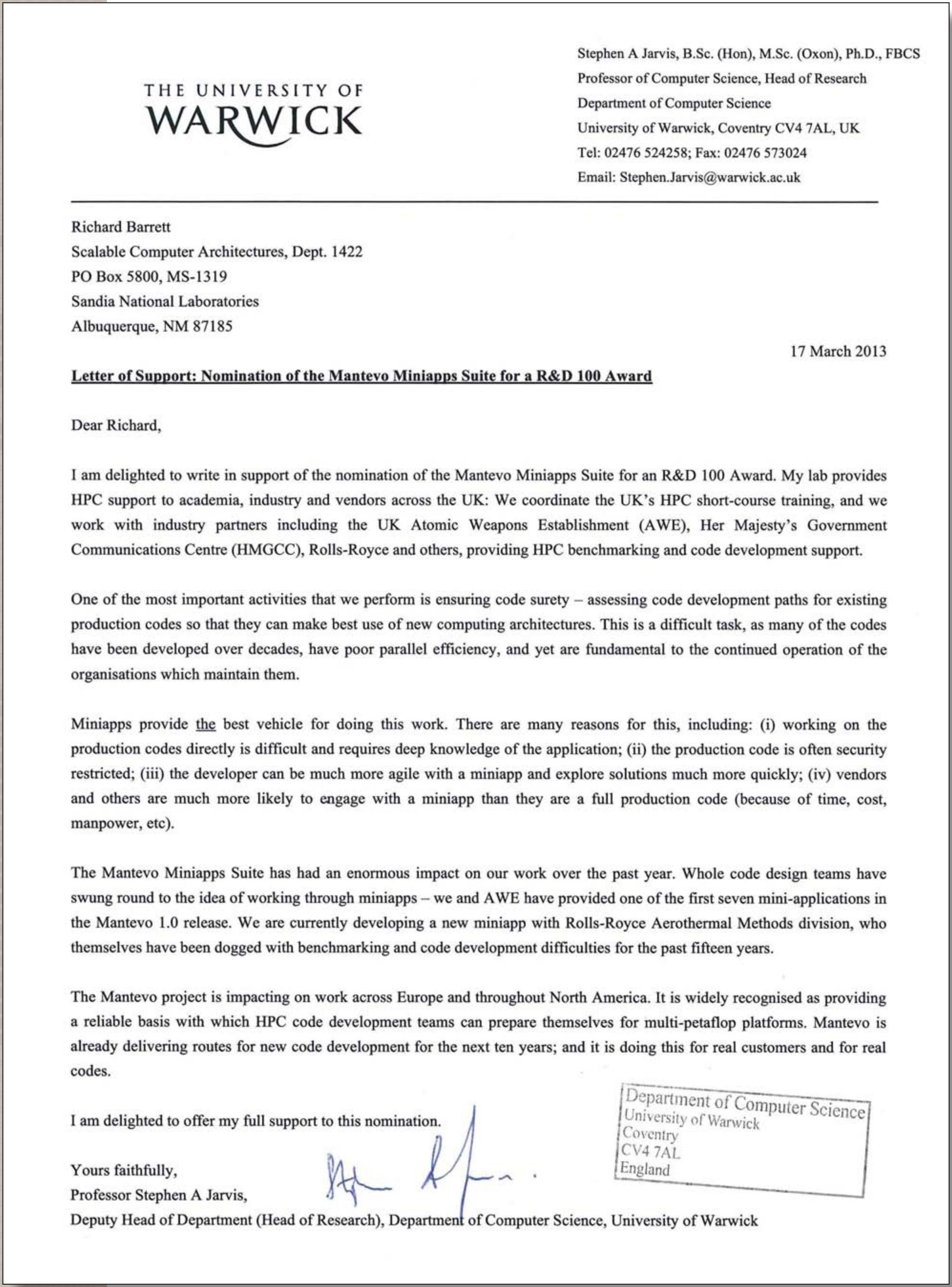
Rob F. Van der Wijngaart, PhD

Software architect

Intel Corporation

Intel Corporation
3600 Juliette Lane
Santa Clara, CA
95054

Appendix C: Letters of Support (cont.)



THE UNIVERSITY OF WARWICK

Stephen A Jarvis, B.Sc. (Hon), M.Sc. (Oxon), Ph.D., FBCS
Professor of Computer Science, Head of Research
Department of Computer Science
University of Warwick, Coventry CV4 7AL, UK
Tel: 02476 524258; Fax: 02476 573024
Email: Stephen.Jarvis@warwick.ac.uk

Richard Barrett
Scalable Computer Architectures, Dept. 1422
PO Box 5800, MS-1319
Sandia National Laboratories
Albuquerque, NM 87185

17 March 2013

Letter of Support: Nomination of the Mantevo Miniapps Suite for a R&D 100 Award

Dear Richard,

I am delighted to write in support of the nomination of the Mantevo Miniapps Suite for an R&D 100 Award. My lab provides HPC support to academia, industry and vendors across the UK: We coordinate the UK's HPC short-course training, and we work with industry partners including the UK Atomic Weapons Establishment (AWE), Her Majesty's Government Communications Centre (HMGCC), Rolls-Royce and others, providing HPC benchmarking and code development support.

One of the most important activities that we perform is ensuring code surety – assessing code development paths for existing production codes so that they can make best use of new computing architectures. This is a difficult task, as many of the codes have been developed over decades, have poor parallel efficiency, and yet are fundamental to the continued operation of the organisations which maintain them.

Miniapps provide the best vehicle for doing this work. There are many reasons for this, including: (i) working on the production codes directly is difficult and requires deep knowledge of the application; (ii) the production code is often security restricted; (iii) the developer can be much more agile with a miniapp and explore solutions much more quickly; (iv) vendors and others are much more likely to engage with a miniapp than they are a full production code (because of time, cost, manpower, etc).

The Mantevo Miniapps Suite has had an enormous impact on our work over the past year. Whole code design teams have swung round to the idea of working through miniapps – we and AWE have provided one of the first seven mini-applications in the Mantevo 1.0 release. We are currently developing a new miniapp with Rolls-Royce Aerothermal Methods division, who themselves have been dogged with benchmarking and code development difficulties for the past fifteen years.

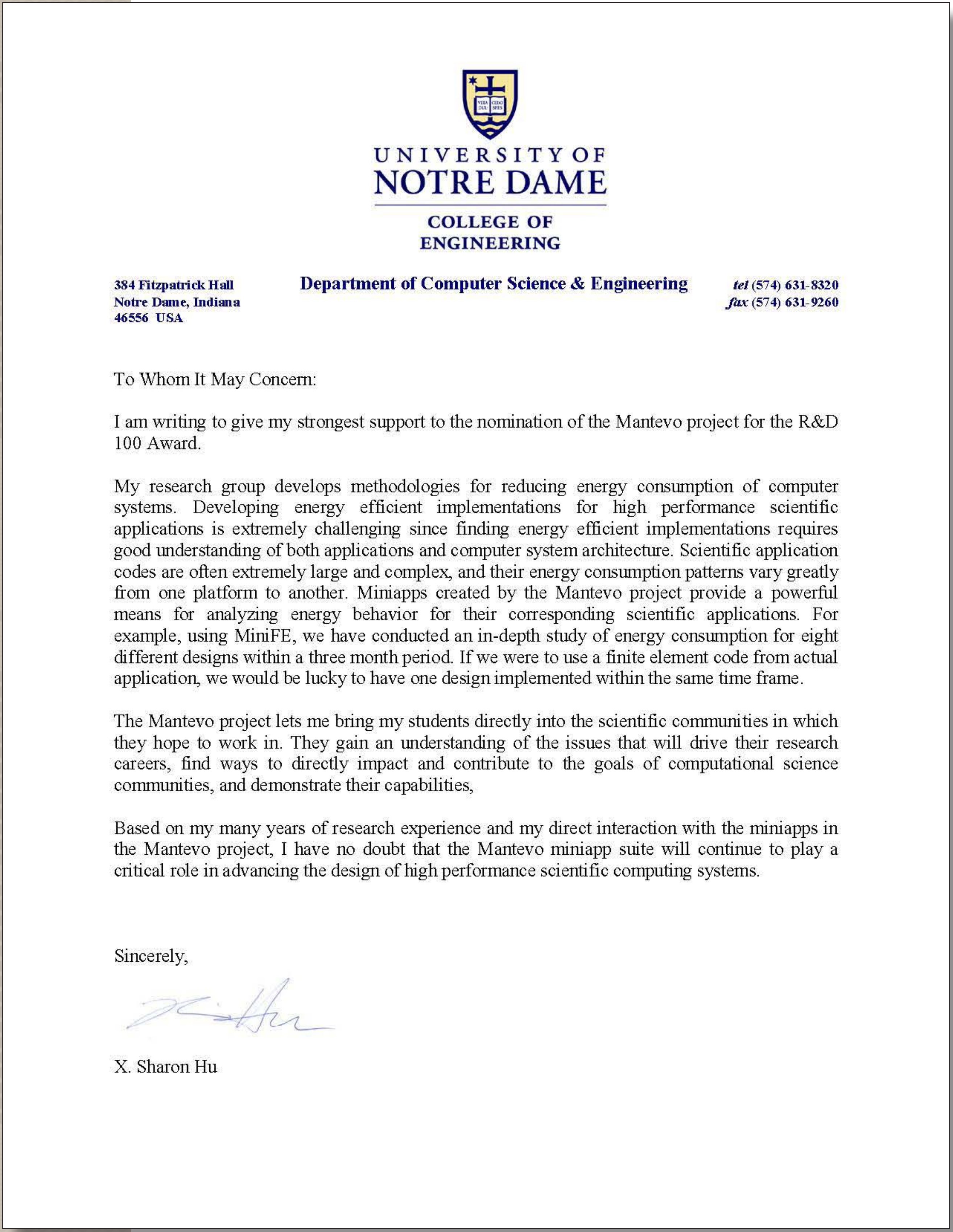
The Mantevo project is impacting on work across Europe and throughout North America. It is widely recognised as providing a reliable basis with which HPC code development teams can prepare themselves for multi-petaflop platforms. Mantevo is already delivering routes for new code development for the next ten years; and it is doing this for real customers and for real codes.

I am delighted to offer my full support to this nomination.

Department of Computer Science
University of Warwick
Coventry
CV4 7AL
England

Yours faithfully,
[illegible]
Professor Stephen A Jarvis,
Deputy Head of Department (Head of Research), Department of Computer Science, University of Warwick

Appendix C: Letters of Support (cont.)



UNIVERSITY OF
NOTRE DAME

COLLEGE OF
ENGINEERING

384 Fitzpatrick Hall
Notre Dame, Indiana
46556 USA

Department of Computer Science & Engineering

tel (574) 631-8320
fax (574) 631-9260

To Whom It May Concern:

I am writing to give my strongest support to the nomination of the Mantevo project for the R&D 100 Award.

My research group develops methodologies for reducing energy consumption of computer systems. Developing energy efficient implementations for high performance scientific applications is extremely challenging since finding energy efficient implementations requires good understanding of both applications and computer system architecture. Scientific application codes are often extremely large and complex, and their energy consumption patterns vary greatly from one platform to another. Miniapps created by the Mantevo project provide a powerful means for analyzing energy behavior for their corresponding scientific applications. For example, using MiniFE, we have conducted an in-depth study of energy consumption for eight different designs within a three month period. If we were to use a finite element code from actual application, we would be lucky to have one design implemented within the same time frame.

The Mantevo project lets me bring my students directly into the scientific communities in which they hope to work in. They gain an understanding of the issues that will drive their research careers, find ways to directly impact and contribute to the goals of computational science communities, and demonstrate their capabilities,

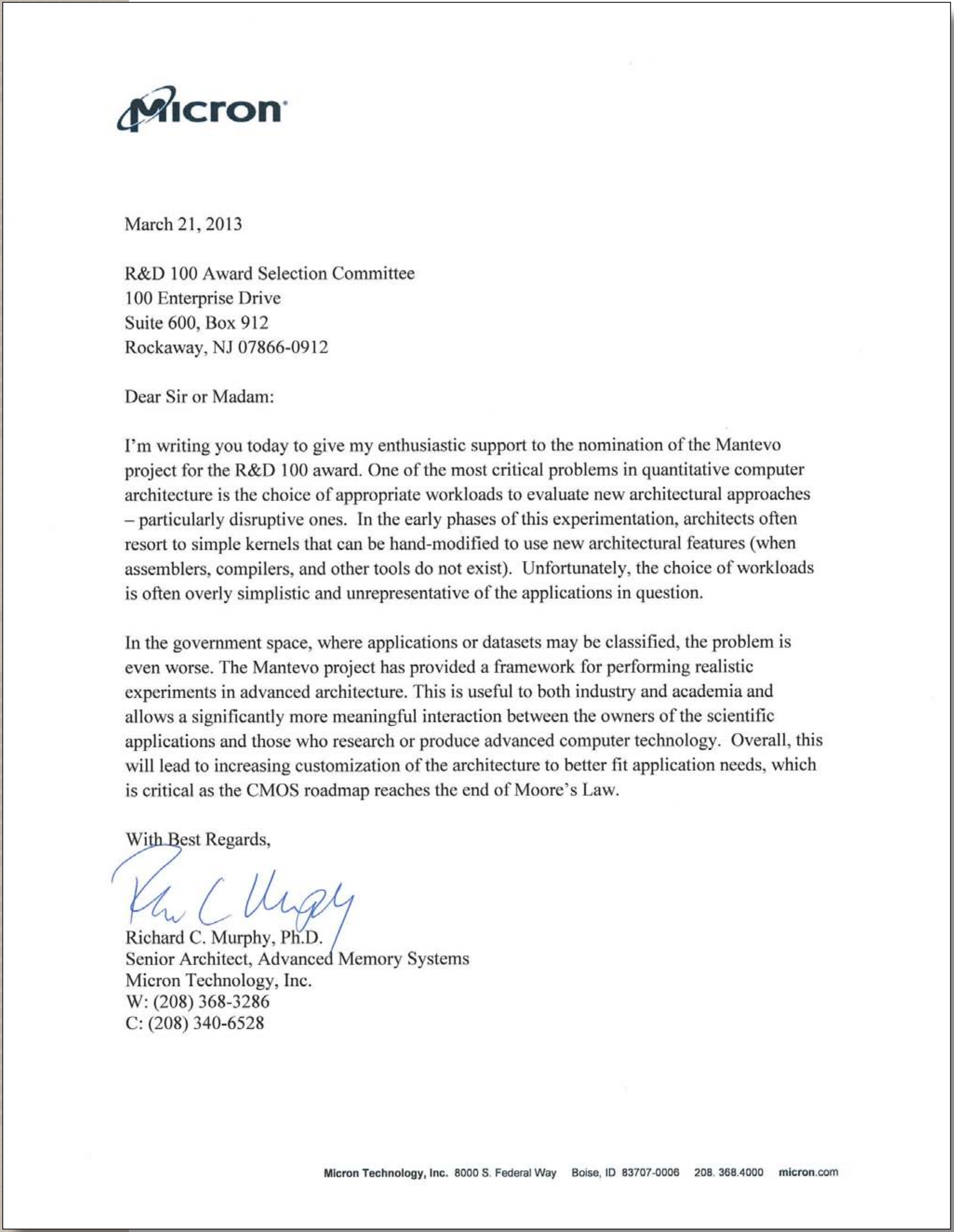
Based on my many years of research experience and my direct interaction with the miniapps in the Mantevo project, I have no doubt that the Mantevo miniapp suite will continue to play a critical role in advancing the design of high performance scientific computing systems.

Sincerely,

[illegible]

X. Sharon Hu

Appendix C: Letters of Support (cont.)



Micron

March 21, 2013

R&D 100 Award Selection Committee
100 Enterprise Drive
Suite 600, Box 912
Rockaway, NJ 07866-0912

Dear Sir or Madam:

I'm writing you today to give my enthusiastic support to the nomination of the Mantevo project for the R&D 100 award. One of the most critical problems in quantitative computer architecture is the choice of appropriate workloads to evaluate new architectural approaches – particularly disruptive ones. In the early phases of this experimentation, architects often resort to simple kernels that can be hand-modified to use new architectural features (when assemblers, compilers, and other tools do not exist). Unfortunately, the choice of workloads is often overly simplistic and unrepresentative of the applications in question.

In the government space, where applications or datasets may be classified, the problem is even worse. The Mantevo project has provided a framework for performing realistic experiments in advanced architecture. This is useful to both industry and academia and allows a significantly more meaningful interaction between the owners of the scientific applications and those who research or produce advanced computer technology. Overall, this will lead to increasing customization of the architecture to better fit application needs, which is critical as the CMOS roadmap reaches the end of Moore's Law.

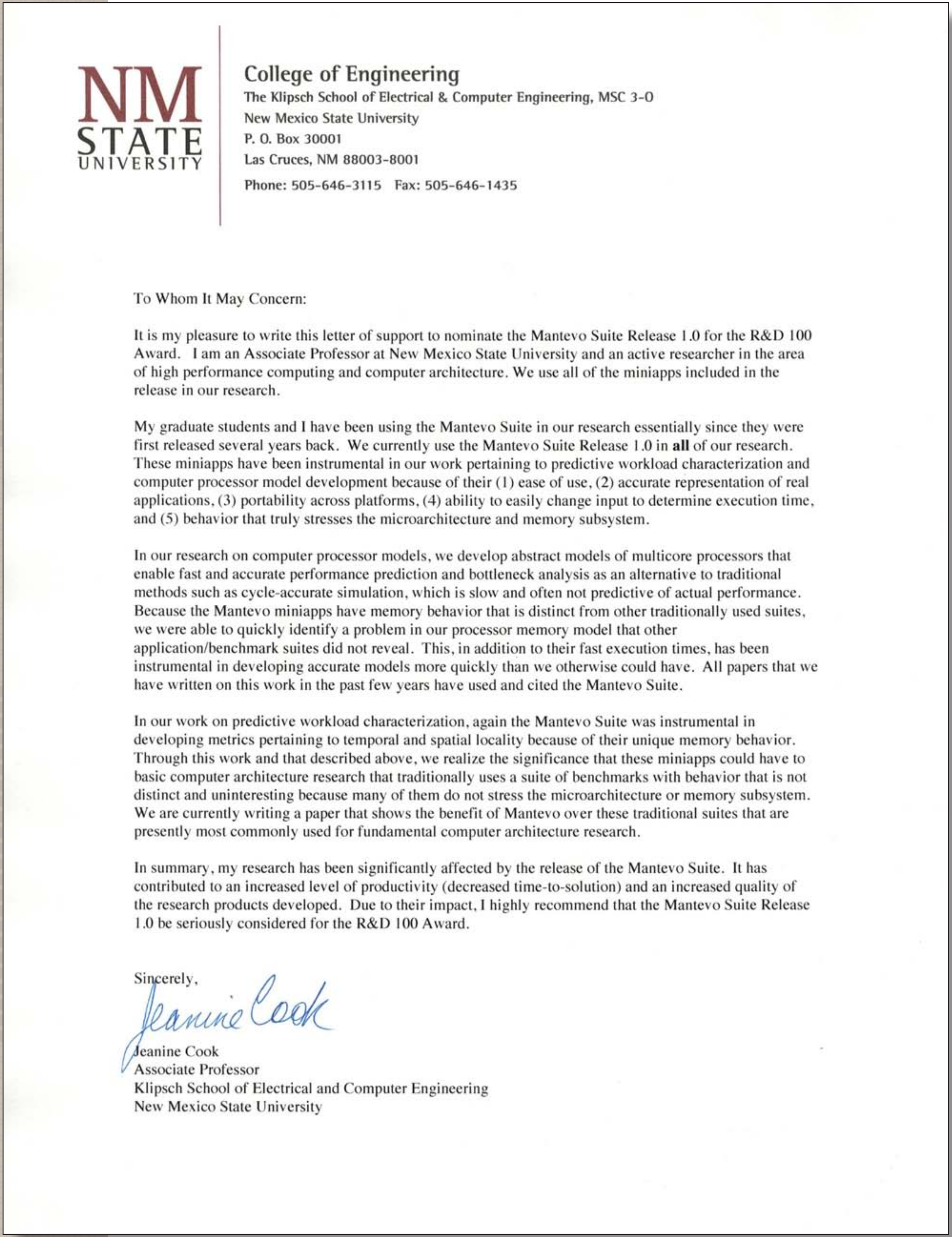
With Best Regards,

[illegible]

Richard C. Murphy, Ph.D.
Senior Architect, Advanced Memory Systems
Micron Technology, Inc.
W: (208) 368-3286
C: (208) 340-6528

Micron Technology, Inc. 8000 S. Federal Way Boise, ID 83707-0006 208. 368.4000 **micron**.com

Appendix C: Letters of Support (cont.)



NM STATE UNIVERSITY

College of Engineering
The Klipsch School of Electrical & Computer Engineering, MSC 3-O
New Mexico State University
P. O. Box 30001
Las Cruces, NM 88003-8001
Phone: 505-646-3115 Fax: 505-646-1435

To Whom It May Concern:

It is my pleasure to write this letter of support to nominate the Mantevo Suite Release 1.0 for the R&D 100 Award. I am an Associate Professor at New Mexico State University and an active researcher in the area of high performance computing and computer architecture. We use all of the miniapps included in the release in our research.

My graduate students and I have been using the Mantevo Suite in our research essentially since they were first released several years back. We currently use the Mantevo Suite Release 1.0 in **all** of our research. These miniapps have been instrumental in our work pertaining to predictive workload characterization and computer processor model development because of their (1) ease of use, (2) accurate representation of real applications, (3) portability across platforms, (4) ability to easily change input to determine execution time, and (5) behavior that truly stresses the microarchitecture and memory subsystem.

In our research on computer processor models, we develop abstract models of multicore processors that enable fast and accurate performance prediction and bottleneck analysis as an alternative to traditional methods such as cycle-accurate simulation, which is slow and often not predictive of actual performance. Because the Mantevo miniapps have memory behavior that is distinct from other traditionally used suites, we were able to quickly identify a problem in our processor memory model that other application/benchmark suites did not reveal. This, in addition to their fast execution times, has been instrumental in developing accurate models more quickly than we otherwise could have. All papers that we have written on this work in the past few years have used and cited the Mantevo Suite.

In our work on predictive workload characterization, again the Mantevo Suite was instrumental in developing metrics pertaining to temporal and spatial locality because of their unique memory behavior. Through this work and that described above, we realize the significance that these miniapps could have to basic computer architecture research that traditionally uses a suite of benchmarks with behavior that is not distinct and uninteresting because many of them do not stress the microarchitecture or memory subsystem. We are currently writing a paper that shows the benefit of Mantevo over these traditional suites that are presently most commonly used for fundamental computer architecture research.

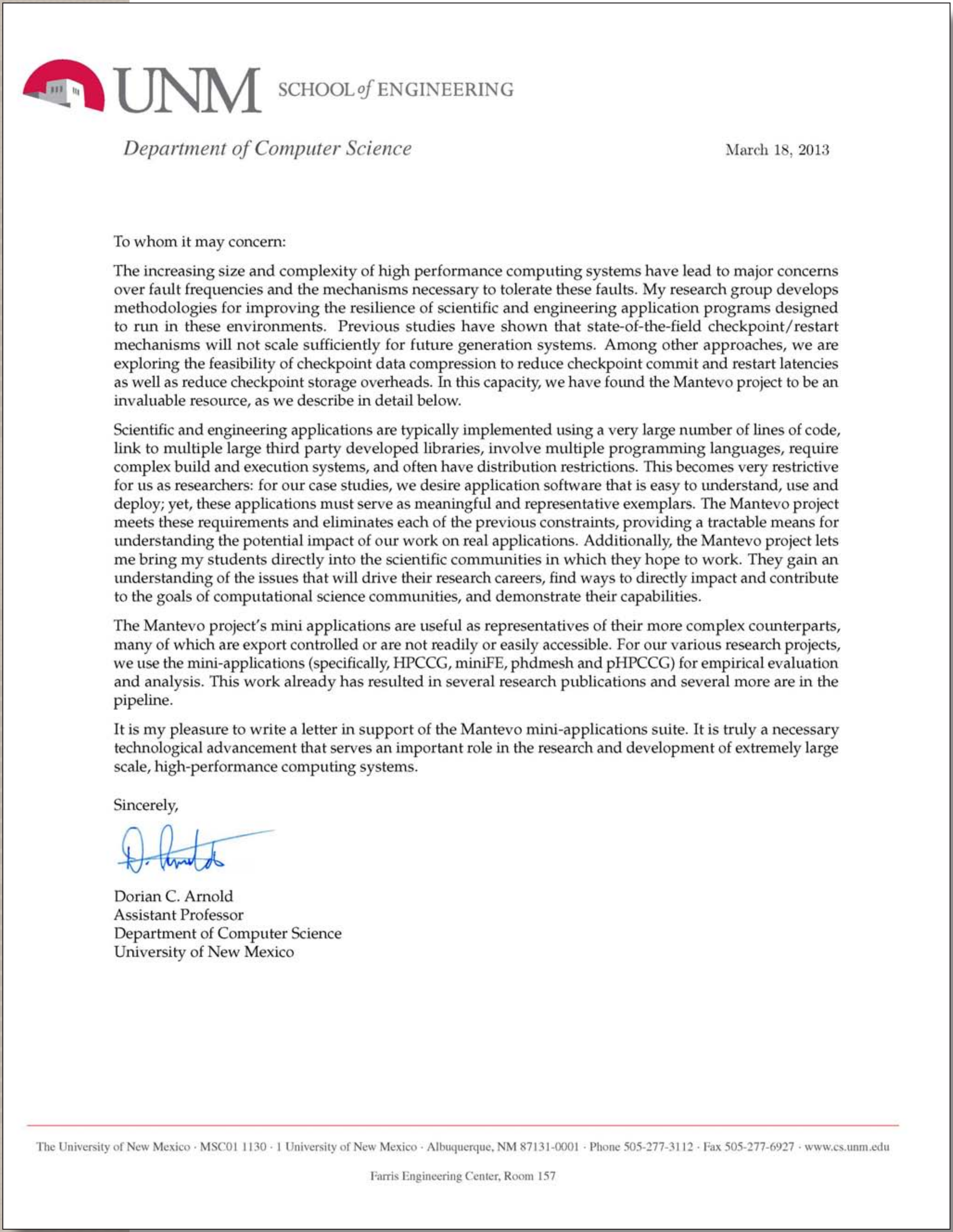
In summary, my research has been significantly affected by the release of the Mantevo Suite. It has contributed to an increased level of productivity (decreased time-to-solution) and an increased quality of the research products developed. Due to their impact, I highly recommend that the Mantevo Suite Release 1.0 be seriously considered for the R&D 100 Award.

Sincerely,

Jeanine Cook

Jeanine Cook
Associate Professor
Klipsch School of Electrical and Computer Engineering
New Mexico State University

Appendix C: Letters of Support (cont.)



UNM SCHOOL of ENGINEERING

Department of Computer Science

March 18, 2013

To whom it may concern:

The increasing size and complexity of high performance computing systems have lead to major concerns over fault frequencies and the mechanisms necessary to tolerate these faults. My research group develops methodologies for improving the resilience of scientific and engineering application programs designed to run in these environments. Previous studies have shown that state-of-the-field checkpoint/restart mechanisms will not scale sufficiently for future generation systems. Among other approaches, we are exploring the feasibility of checkpoint data compression to reduce checkpoint commit and restart latencies as well as reduce checkpoint storage overheads. In this capacity, we have found the Mantevo project to be an invaluable resource, as we describe in detail below.

Scientific and engineering applications are typically implemented using a very large number of lines of code, link to multiple large third party developed libraries, involve multiple programming languages, require complex build and execution systems, and often have distribution restrictions. This becomes very restrictive for us as researchers: for our case studies, we desire application software that is easy to understand, use and deploy; yet, these applications must serve as meaningful and representative exemplars. The Mantevo project meets these requirements and eliminates each of the previous constraints, providing a tractable means for understanding the potential impact of our work on real applications. Additionally, the Mantevo project lets me bring my students directly into the scientific communities in which they hope to work. They gain an understanding of the issues that will drive their research careers, find ways to directly impact and contribute to the goals of computational science communities, and demonstrate their capabilities.

The Mantevo project's mini applications are useful as representatives of their more complex counterparts, many of which are export controlled or are not readily or easily accessible. For our various research projects, we use the mini-applications (specifically, HPCCG, miniFE, phdmesh and pHPCCG) for empirical evaluation and analysis. This work already has resulted in several research publications and several more are in the pipeline.

It is my pleasure to write a letter in support of the Mantevo mini-applications suite. It is truly a necessary technological advancement that serves an important role in the research and development of extremely large scale, high-performance computing systems.

Sincerely,

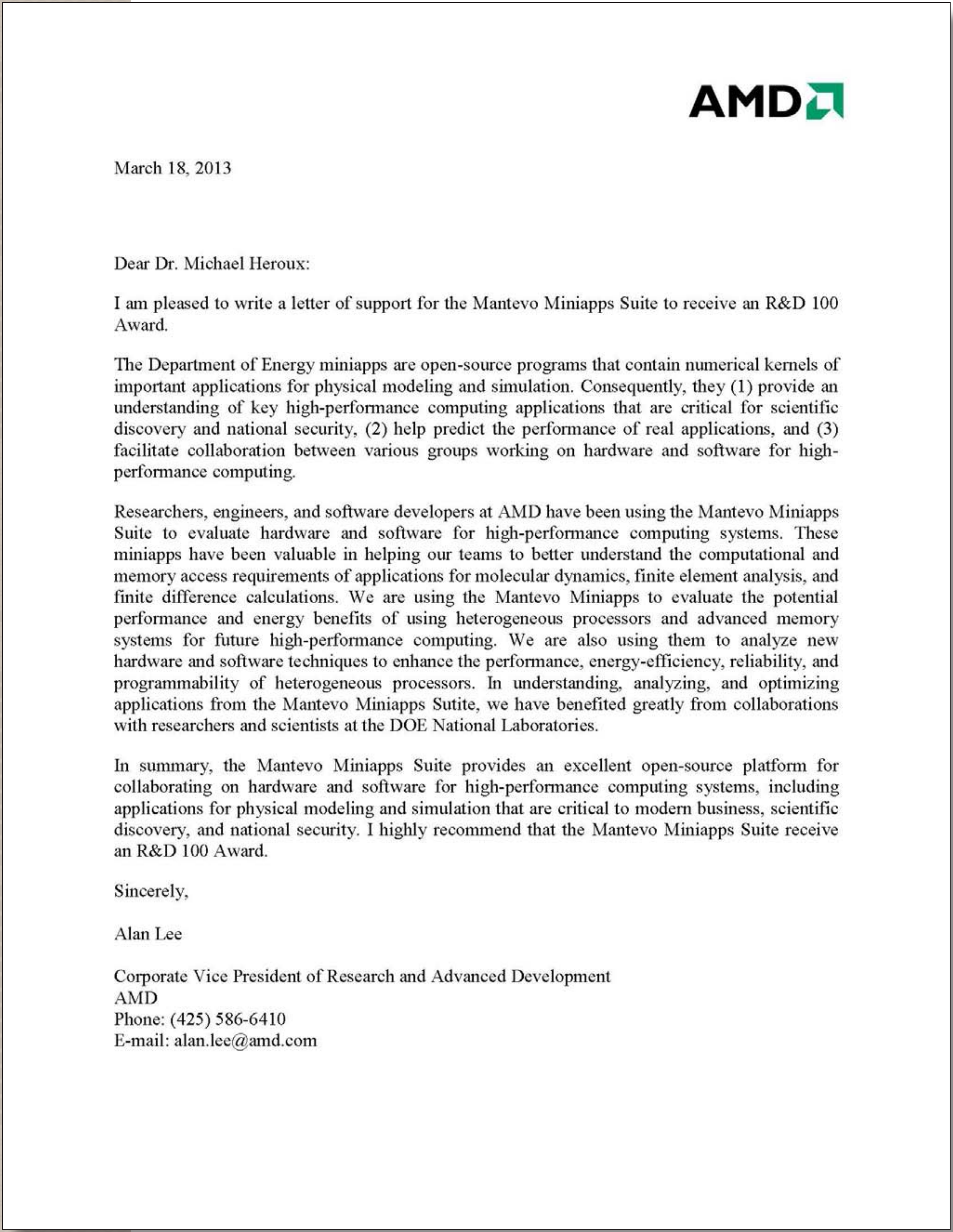
[illegible]

Dorian C. Arnold
Assistant Professor
Department of Computer Science
University of New Mexico

The University of New Mexico · MSC01 1130 · 1 University of New Mexico · Albuquerque, NM 87131-0001 · Phone 505-277-3112 · Fax 505-277-6927 · www.cs.unm.edu

Farris Engineering Center, Room 157

Appendix C: Letters of Support (cont.)



AMD

March 18, 2013

Dear Dr. Michael Heroux:

I am pleased to write a letter of support for the Mantevo Miniapps Suite to receive an R&D 100 Award.

The Department of Energy miniapps are open-source programs that contain numerical kernels of important applications for physical modeling and simulation. Consequently, they (1) provide an understanding of key high-performance computing applications that are critical for scientific discovery and national security, (2) help predict the performance of real applications, and (3) facilitate collaboration between various groups working on hardware and software for high-performance computing.

Researchers, engineers, and software developers at AMD have been using the Mantevo Miniapps Suite to evaluate hardware and software for high-performance computing systems. These miniapps have been valuable in helping our teams to better understand the computational and memory access requirements of applications for molecular dynamics, finite element analysis, and finite difference calculations. We are using the Mantevo Miniapps to evaluate the potential performance and energy benefits of using heterogeneous processors and advanced memory systems for future high-performance computing. We are also using them to analyze new hardware and software techniques to enhance the performance, energy-efficiency, reliability, and programmability of heterogeneous processors. In understanding, analyzing, and optimizing applications from the Mantevo Miniapps Sutite, we have benefited greatly from collaborations with researchers and scientists at the DOE National Laboratories.

In summary, the Mantevo Miniapps Suite provides an excellent open-source platform for collaborating on hardware and software for high-performance computing systems, including applications for physical modeling and simulation that are critical to modern business, scientific discovery, and national security. I highly recommend that the Mantevo Miniapps Suite receive an R&D 100 Award.

Sincerely,

Alan Lee

Corporate Vice President of Research and Advanced Development
AMD
Phone: (425) 586-6410
E-mail: alan.lee@amd.com

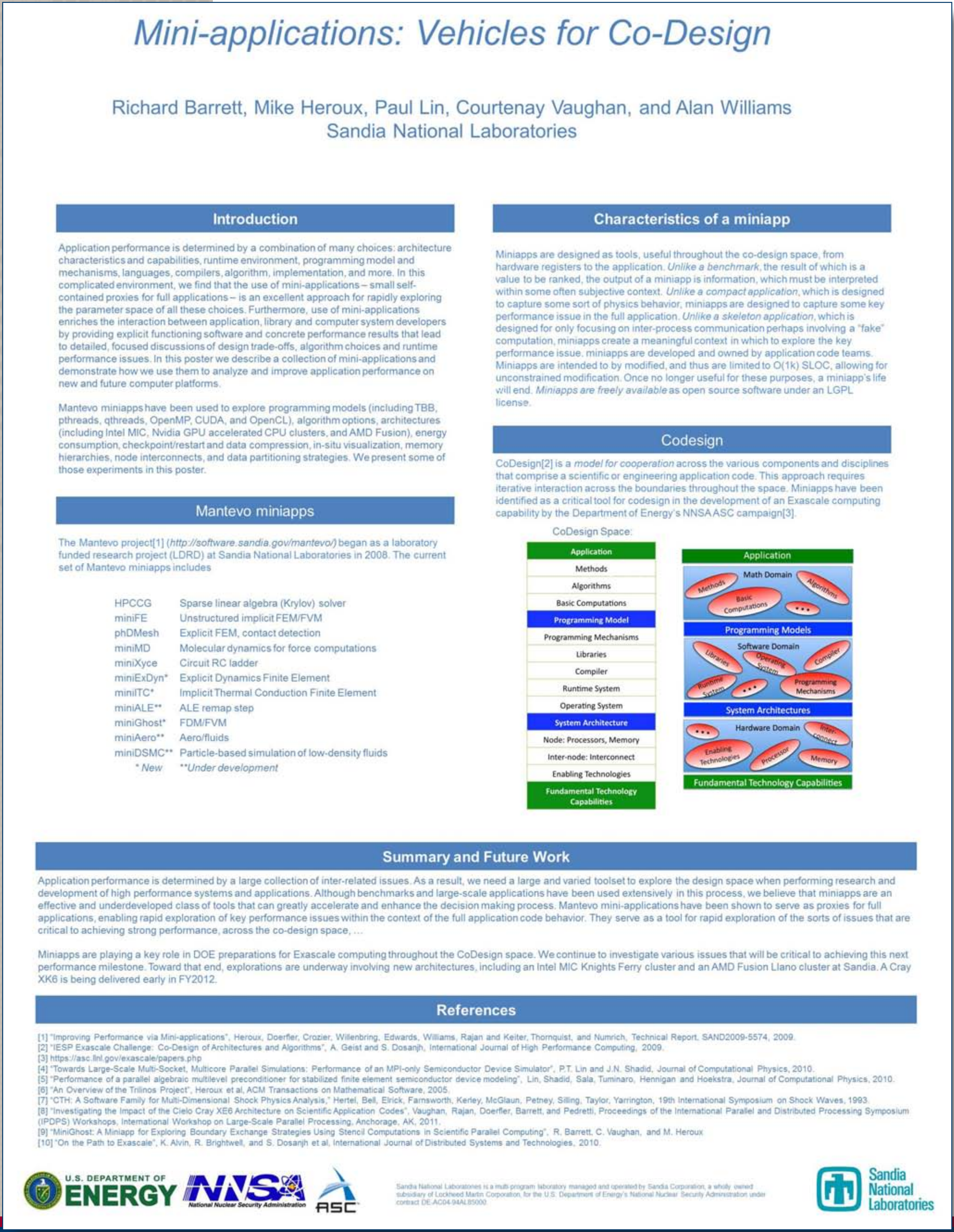
Appendix C: Letters of Support (cont.)

Appendix C: Letters of Support (cont.)

Appendix D: Publications

sc11.supercomputing.org/

SC11 is the International Conference for High Performance Computing, Networking, Storage and Analysis. This poster won the Best Poster prize.



Mini-applications: Vehicles for Co-Design

Richard Barrett, Mike Heroux, Paul Lin, Courtenay Vaughan, and Alan Williams
Sandia National Laboratories

Introduction

Application performance is determined by a combination of many choices: architecture characteristics and capabilities, runtime environment, programming model and mechanisms, languages, compilers, algorithm, implementation, and more. In this complicated environment, we find that the use of mini-applications – small self-contained proxies for full applications – is an excellent approach for rapidly exploring the parameter space of all these choices. Furthermore, use of mini-applications enriches the interaction between application, library and computer system developers by providing explicit functioning software and concrete performance results that lead to detailed, focused discussions of design trade-offs, algorithm choices and runtime performance issues. In this poster we describe a collection of mini-applications and demonstrate how we use them to analyze and improve application performance on new and future computer platforms.

Mantevo miniapps have been used to explore programming models (including TBB, pthreads, qthreads, OpenMP, CUDA, and OpenCL), algorithm options, architectures (including Intel MIC, Nvidia GPU accelerated CPU clusters, and AMD Fusion), energy consumption, checkpoint/restart and data compression, in-situ visualization, memory hierarchies, node interconnects, and data partitioning strategies. We present some of those experiments in this poster.

Mantevo miniapps

The Mantevo project[1] (*http://software.sandia.gov/mantevo/*) began as a laboratory funded research project (LDRD) at Sandia National Laboratories in 2008. The current set of Mantevo miniapps includes

HPCCG	Sparse linear algebra (Krylov) solver
miniFE	Unstructured implicit FEM/FVM
phDMesh	Explicit FEM, contact detection
miniMD	Molecular dynamics for force computations
miniXyce	Circuit RC ladder
miniExDyn*	Explicit Dynamics Finite Element
miniITC*	Implicit Thermal Conduction Finite Element
miniALE**	ALE remap step
miniGhost*	FDM/FVM
miniAero**	Aero/fluids
miniDSMC**	Particle-based simulation of low-density fluids
** New*	***Under development*

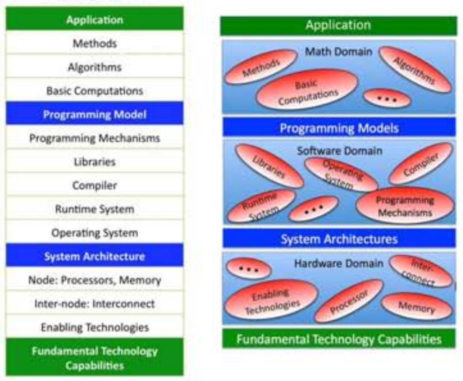
Characteristics of a miniapp

Miniapps are designed as tools, useful throughout the co-design space, from hardware registers to the application. *Unlike a benchmark*, the result of which is a value to be ranked, the output of a miniapp is information, which must be interpreted within some often subjective context. *Unlike a compact application*, which is designed to capture some sort of physics behavior, miniapps are designed to capture some key performance issue in the full application. *Unlike a skeleton application*, which is designed for only focusing on inter-process communication perhaps involving a "fake" computation, miniapps create a meaningful context in which to explore the key performance issue. miniapps are developed and owned by application code teams. Miniapps are intended to by modified, and thus are limited to O(1k) SLOC, allowing for unconstrained modification. Once no longer useful for these purposes, a miniapp's life will end. *Miniapps are freely available* as open source software under an LGPL license.

Codesign

CoDesign[2] is a *model for cooperation* across the various components and disciplines that comprise a scientific or engineering application code. This approach requires iterative interaction across the boundaries throughout the space. Miniapps have been identified as a critical tool for codesign in the development of an Exascale computing capability by the Department of Energy's NNSA ASC campaign[3].

CoDesign Space:



Summary and Future Work

Application performance is determined by a large collection of inter-related issues. As a result, we need a large and varied toolset to explore the design space when performing research and development of high performance systems and applications. Although benchmarks and large-scale applications have been used extensively in this process, we believe that miniapps are an effective and underdeveloped class of tools that can greatly accelerate and enhance the decision making process. Mantevo mini-applications have been shown to serve as proxies for full applications, enabling rapid exploration of key performance issues within the context of the full application code behavior. They serve as a tool for rapid exploration of the sorts of issues that are critical to achieving strong performance, across the co-design space, …

Miniapps are playing a key role in DOE preparations for Exascale computing throughout the CoDesign space. We continue to investigate various issues that will be critical to achieving this next performance milestone. Toward that end, explorations are underway involving new architectures, including an Intel MIC Knights Ferry cluster and an AMD Fusion Llano cluster at Sandia. A Cray XK6 is being delivered early in FY2012.

References

[1] "Improving Performance via Mini-applications", Heroux, Doerfler, Crozier, Willenbring, Edwards, Williams, Rajan and Keiter, Thornquist, and Numrich, Technical Report, SAND2009-5574, 2009.
[2] "IESP Exascale Challenge: Co-Design of Architectures and Algorithms", A. Geist and S. Dosanjh, International Journal of High Performance Computing, 2009.
[3] https://asc.llnl.gov/exascale/papers.php
[4] "Towards Large-Scale Multi-Socket, Multicore Parallel Simulations: Performance of an MPI-only Semiconductor Device Simulator", P.T. Lin and J.N. Shadid, Journal of Computational Physics, 2010.
[5] "Performance of a parallel algebraic multilevel preconditioner for stabilized finite element semiconductor device modeling", Lin, Shadid, Sala, Tuminaro, Hennigan and Hoekstra, Journal of Computational Physics, 2010.
[6] "An Overview of the Trilinos Project", Heroux et al, ACM Transactions on Mathematical Software, 2005.
[7] "CTH: A Software Family for Multi-Dimensional Shock Physics Analysis," Hertel, Bell, Elrick, Farnsworth, Kerley, McGlaun, Petney, Silling, Taylor, Yarrington, 19th International Symposium on Shock Waves, 1993.
[8] "Investigating the Impact of the Cielo Cray XE6 Architecture on Scientific Application Codes", Vaughan, Rajan, Doerfler, Barrett, and Pedretti, Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS) Workshops, International Workshop on Large-Scale Parallel Processing, Anchorage, AK, 2011.
[9] "MiniGhost: A Miniapp for Exploring Boundary Exchange Strategies Using Stencil Computations in Scientific Parallel Computing", R. Barrett, C. Vaughan, and M. Heroux
[10] "On the Path to Exascale", K. Alvin, R. Brightwell, and S. Dosanjh et al, International Journal of Distributed Systems and Technologies, 2010.

U.S. DEPARTMENT OF ENERGY NNSA National Nuclear Security Administration ASC

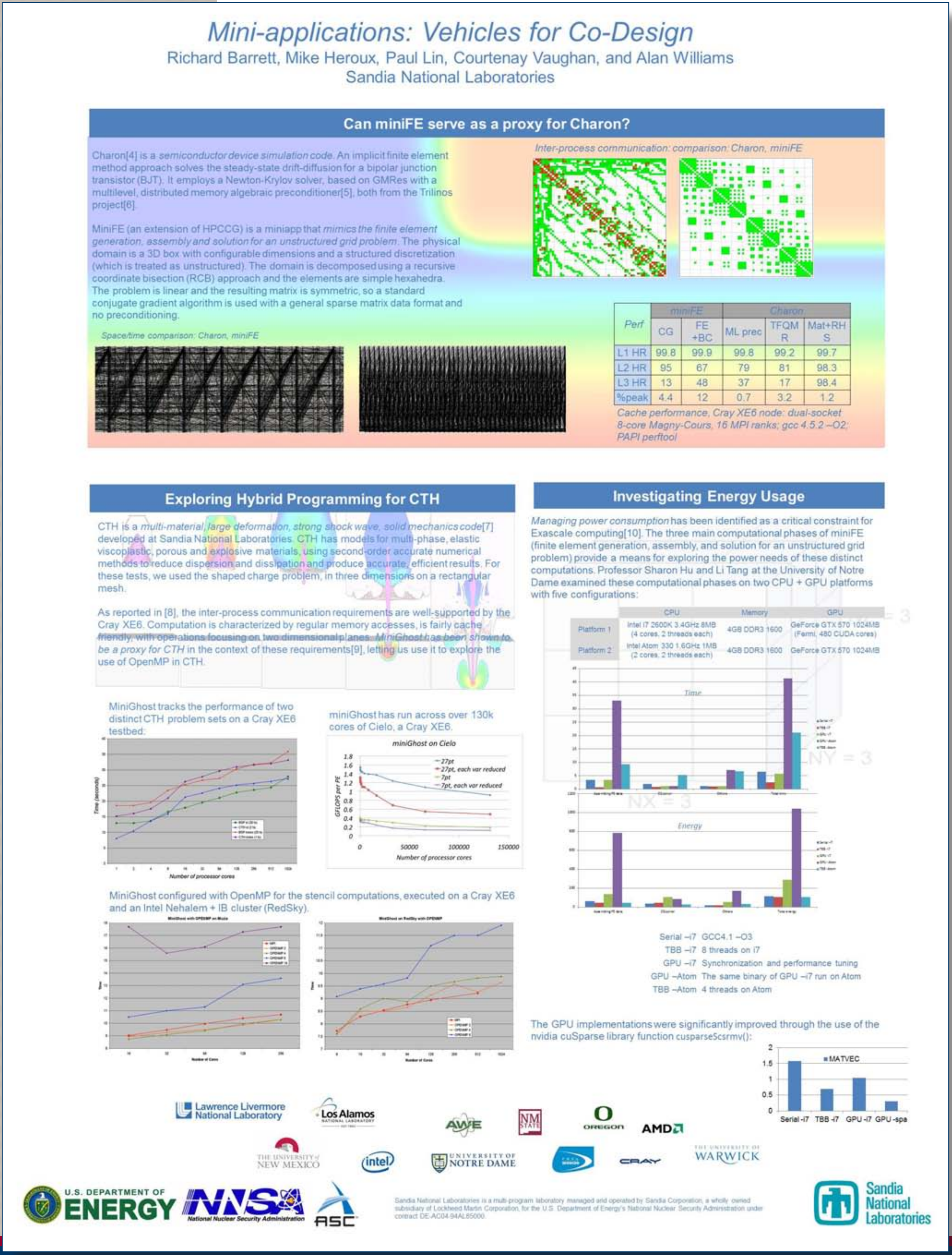
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Appendix D: Publications (cont.)

sc11.supercomputing.org/

Poster-2nd page.



Mini-applications: Vehicles for Co-Design

Richard Barrett, Mike Heroux, Paul Lin, Courtenay Vaughan, and Alan Williams
Sandia National Laboratories

Can miniFE serve as a proxy for Charon?

Charon[4] is a *semiconductor device simulation code*. An implicit finite element method approach solves the steady-state drift-diffusion for a bipolar junction transistor (BJT). It employs a Newton-Krylov solver, based on GMRes with a multilevel, distributed memory algebraic preconditioner[5], both from the Trilinos project[6].

MiniFE (an extension of HPCCG) is a miniapp that *mimics the finite element generation, assembly and solution for an unstructured grid problem*. The physical domain is a 3D box with configurable dimensions and a structured discretization (which is treated as unstructured). The domain is decomposed using a recursive coordinate bisection (RCB) approach and the elements are simple hexahedra. The problem is linear and the resulting matrix is symmetric, so a standard conjugate gradient algorithm is used with a general sparse matrix data format and no preconditioning.

Space/time comparison: Charon, miniFE

Inter-process communication: comparison: Charon, miniFE

Perf	miniFE		Charon		
	CG	FE +BC	ML prec	TFQMR	Mat+RHS
L1 HR	99.8	99.9	99.8	99.2	99.7
L2 HR	95	67	79	81	98.3
L3 HR	13	48	37	17	98.4
%peak	4.4	12	0.7	3.2	1.2

Cache performance, Cray XE6 node: dual-socket 8-core Magny-Cours, 16 MPI ranks; gcc 4.5.2 –O2; PAPI perftool

Exploring Hybrid Programming for CTH

CTH is a *multi-material, large deformation, strong shock wave, solid mechanics code*[7] developed at Sandia National Laboratories. CTH has models for multi-phase, elastic viscoplastic, porous and explosive materials, using second-order accurate numerical methods to reduce dispersion and dissipation and produce accurate, efficient results. For these tests, we used the shaped charge problem, in three dimensions on a rectangular mesh.

As reported in [8], the inter-process communication requirements are well-supported by the Cray XE6. Computation is characterized by regular memory accesses, is fairly cache friendly, with operations focusing on two dimensional planes. *MiniGhost has been shown to be a proxy for CTH* in the context of these requirements[9], letting us use it to explore the use of OpenMP in CTH.

MiniGhost tracks the performance of two distinct CTH problem sets on a Cray XE6 testbed:

miniGhost has run across over 130k cores of Cielo, a Cray XE6.

MiniGhost configured with OpenMP for the stencil computations, executed on a Cray XE6 and an Intel Nehalem + IB cluster (RedSky).

Investigating Energy Usage

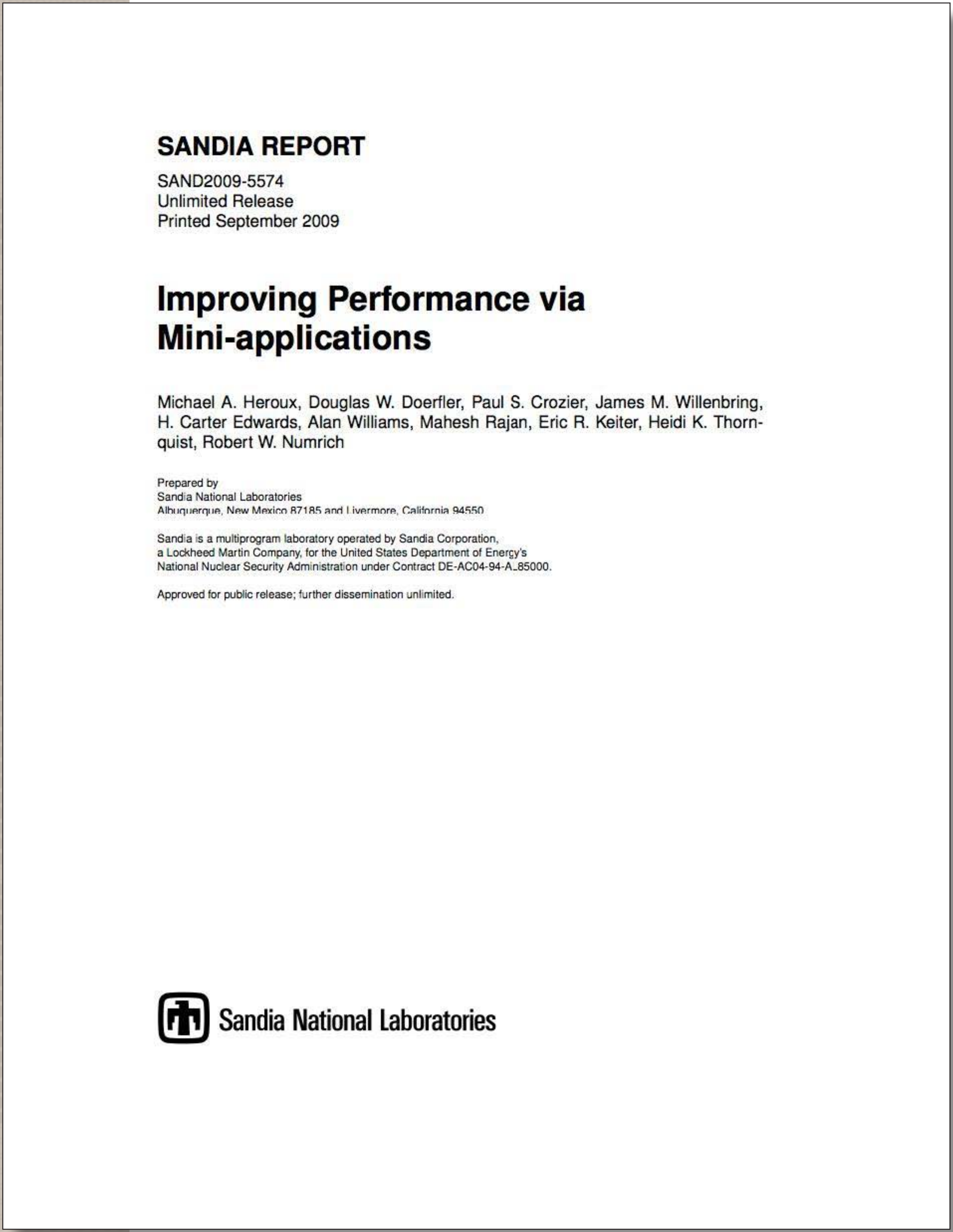
Managing power consumption has been identified as a critical constraint for Exascale computing[10]. The three main computational phases of miniFE (finite element generation, assembly, and solution for an unstructured grid problem) provide a means for exploring the power needs of these distinct computations. Professor Sharon Hu and Li Tang at the University of Notre Dame examined these computational phases on two CPU + GPU platforms with five configurations:

	CPU	Memory	GPU
Platform 1	Intel I7 2600K 3.4GHz 8MB (4 cores, 2 threads each)	4GB DDR3 1600	GeForce GTX 570 1024MB (Fermi, 480 CUDA cores)
Platform 2	Intel Atom 330 1.6GHz 1MB (2 cores, 2 threads each)	4GB DDR3 1600	GeForce GTX 570 1024MB

Serial –i7	GCC4.1 –O3
TBB –i7	8 threads on i7
GPU –i7	Synchronization and performance tuning
GPU –Atom	The same binary of GPU –i7 run on Atom
TBB –Atom	4 threads on Atom

The GPU implementations were significantly improved through the use of the nvidia cuSparse library function cusparseScsrmv():

Appendix D: Publications (cont.)



SANDIA REPORT

SAND2009-5574
Unlimited Release
Printed September 2009

Improving Performance via Mini-applications

Michael A. Heroux, Douglas W. Doerfler, Paul S. Crozier, James M. Willenbring, H. Carter Edwards, Alan Williams, Mahesh Rajan, Eric R. Keiter, Heidi K. Thornquist, Robert W. Numrich

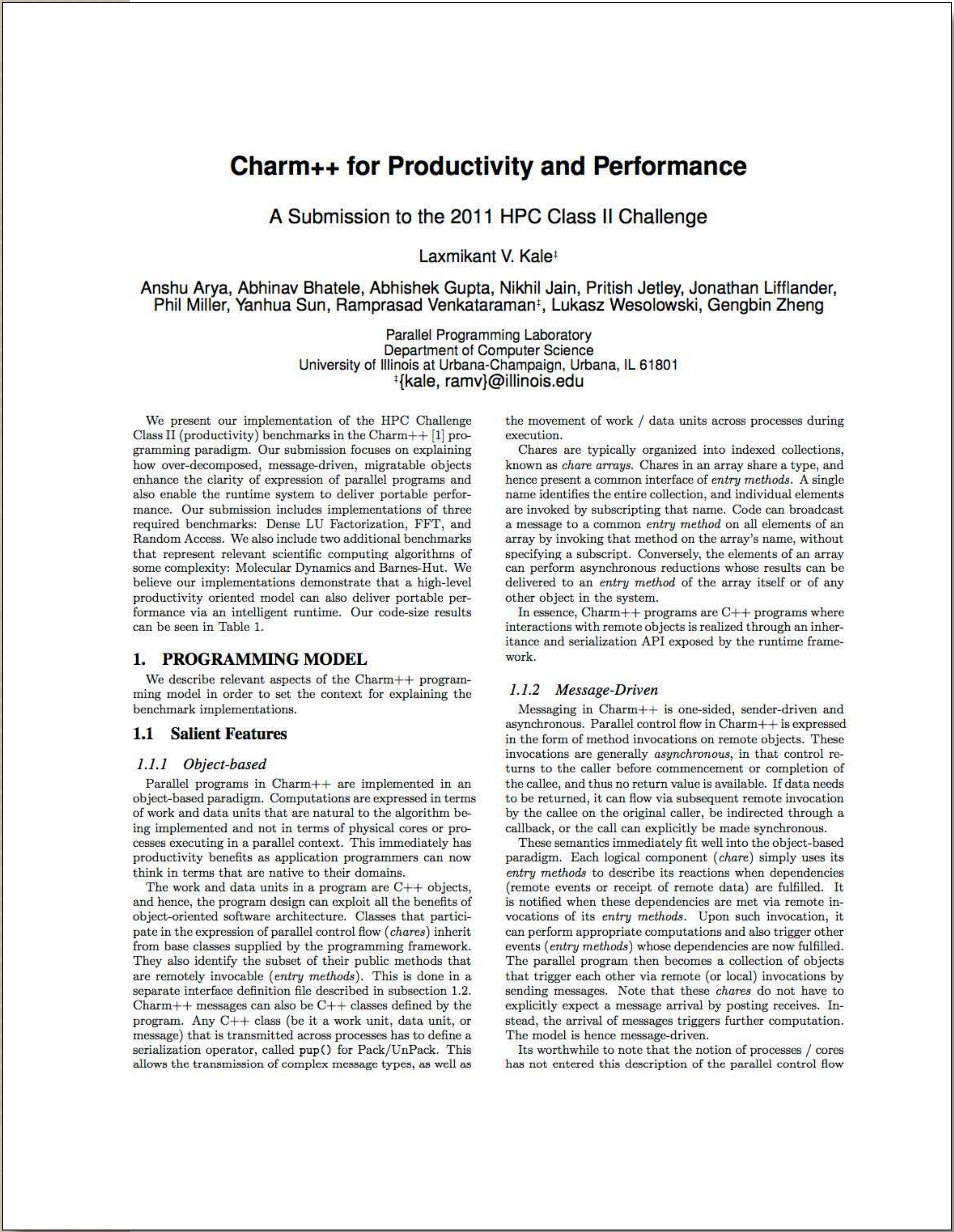
Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Appendix D: Publications (cont.)



Charm++ for Productivity and Performance

A Submission to the 2011 HPC Class II Challenge

Laxmikant V. Kale‡

Anshu Arya, Abhinav Bhatele, Abhishek Gupta, Nikhil Jain, Pritish Jetley, Jonathan Lifflander, Phil Miller, Yanhua Sun, Ramprasad Venkataraman‡, Lukasz Wesolowski, Gengbin Zheng

Parallel Programming Laboratory
Department of Computer Science
University of Illinois at Urbana-Champaign, Urbana, IL 61801
‡{kale, ramv}@illinois.edu

We present our implementation of the HPC Challenge Class II (productivity) benchmarks in the Charm++ [1] programming paradigm. Our submission focuses on explaining how over-decomposed, message-driven, migratable objects enhance the clarity of expression of parallel programs and also enable the runtime system to deliver portable performance. Our submission includes implementations of three required benchmarks: Dense LU Factorization, FFT, and Random Access. We also include two additional benchmarks that represent relevant scientific computing algorithms of some complexity: Molecular Dynamics and Barnes-Hut. We believe our implementations demonstrate that a high-level productivity oriented model can also deliver portable performance via an intelligent runtime. Our code-size results can be seen in Table 1.

1. PROGRAMMING MODEL

We describe relevant aspects of the Charm++ programming model in order to set the context for explaining the benchmark implementations.

1.1 Salient Features

1.1.1 Object-based

Parallel programs in Charm++ are implemented in an object-based paradigm. Computations are expressed in terms of work and data units that are natural to the algorithm being implemented and not in terms of physical cores or processes executing in a parallel context. This immediately has productivity benefits as application programmers can now think in terms that are native to their domains.

The work and data units in a program are C++ objects, and hence, the program design can exploit all the benefits of object-oriented software architecture. Classes that participate in the expression of parallel control flow (*chares*) inherit from base classes supplied by the programming framework. They also identify the subset of their public methods that are remotely invocable (*entry methods*). This is done in a separate interface definition file described in subsection 1.2. Charm++ messages can also be C++ classes defined by the program. Any C++ class (be it a work unit, data unit, or message) that is transmitted across processes has to define a serialization operator, called `pup()` for Pack/UnPack. This allows the transmission of complex message types, as well as the movement of work / data units across processes during execution.

Chares are typically organized into indexed collections, known as *chare arrays*. Chares in an array share a type, and hence present a common interface of *entry methods*. A single name identifies the entire collection, and individual elements are invoked by subscripting that name. Code can broadcast a message to a common *entry method* on all elements of an array by invoking that method on the array's name, without specifying a subscript. Conversely, the elements of an array can perform asynchronous reductions whose results can be delivered to an *entry method* of the array itself or of any other object in the system.

In essence, Charm++ programs are C++ programs where interactions with remote objects is realized through an inheritance and serialization API exposed by the runtime framework.

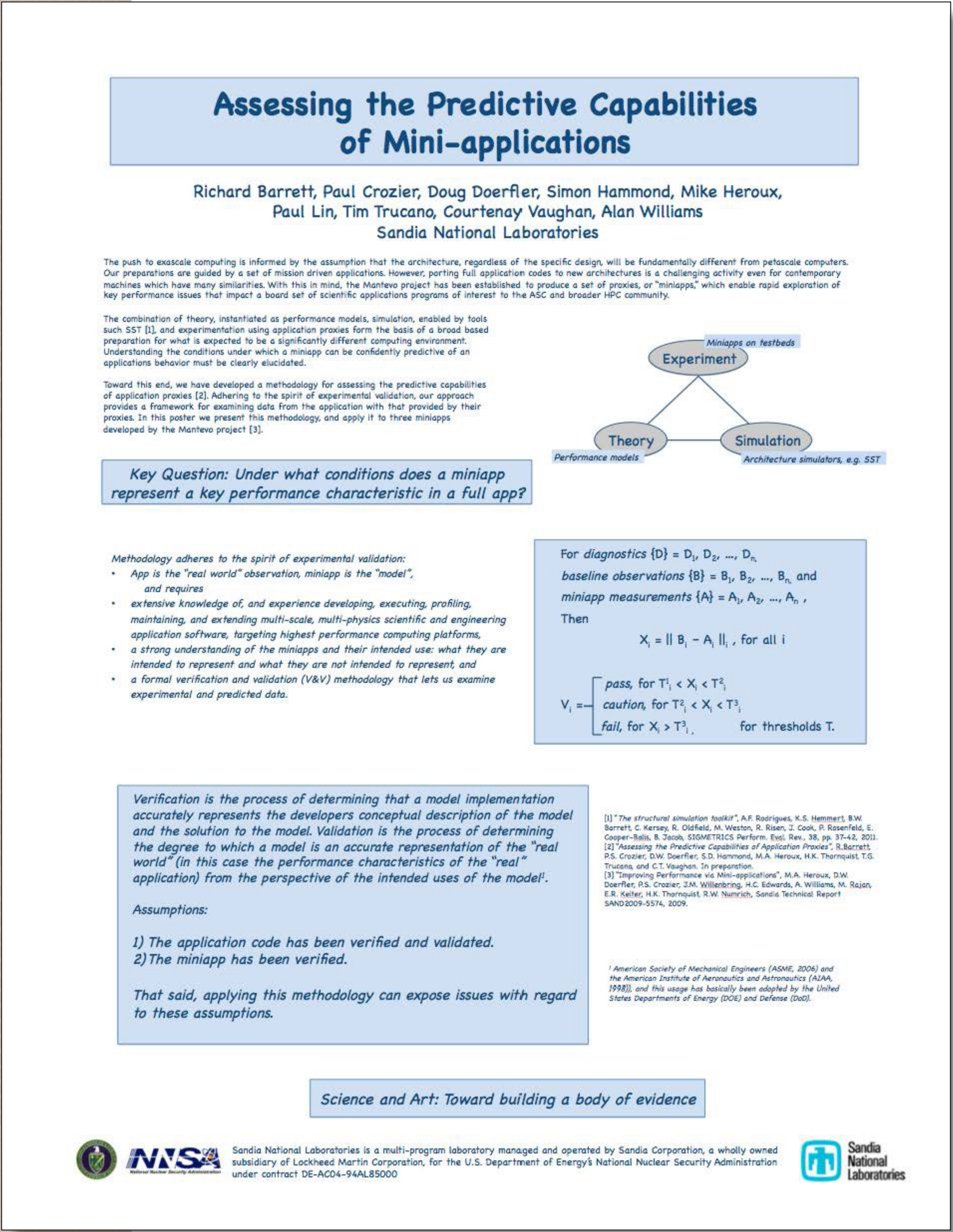
1.1.2 Message-Driven

Messaging in Charm++ is one-sided, sender-driven and asynchronous. Parallel control flow in Charm++ is expressed in the form of method invocations on remote objects. These invocations are generally *asynchronous*, in that control returns to the caller before commencement or completion of the callee, and thus no return value is available. If data needs to be returned, it can flow via subsequent remote invocation by the callee on the original caller, be indirected through a callback, or the call can explicitly be made synchronous.

These semantics immediately fit well into the object-based paradigm. Each logical component (*chare*) simply uses its *entry methods* to describe its reactions when dependencies (remote events or receipt of remote data) are fulfilled. It is notified when these dependencies are met via remote invocations of its *entry methods*. Upon such invocation, it can perform appropriate computations and also trigger other events (*entry methods*) whose dependencies are now fulfilled. The parallel program then becomes a collection of objects that trigger each other via remote (or local) invocations by sending messages. Note that these *chares* do not have to explicitly expect a message arrival by posting receives. Instead, the arrival of messages triggers further computation. The model is hence message-driven.

Its worthwhile to note that the notion of processes / cores has not entered this description of the parallel control flow

Appendix D: Publications (cont.)



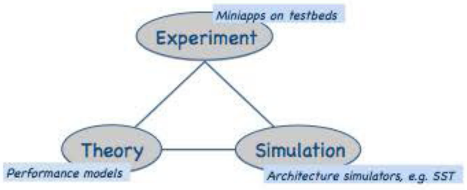
Assessing the Predictive Capabilities of Mini-applications

Richard Barrett, Paul Crozier, Doug Doerfler, Simon Hammond, Mike Heroux,
Paul Lin, Tim Trucano, Courtenay Vaughan, Alan Williams
Sandia National Laboratories

The push to exascale computing is informed by the assumption that the architecture, regardless of the specific design, will be fundamentally different from petascale computers. Our preparations are guided by a set of mission driven applications. However, porting full application codes to new architectures is a challenging activity even for contemporary machines which have many similarities. With this in mind, the Mantevo project has been established to produce a set of proxies, or "miniapps," which enable rapid exploration of key performance issues that impact a board set of scientific applications programs of interest to the ASC and broader HPC community.

The combination of theory, instantiated as performance models, simulation, enabled by tools such SST [1], and experimentation using application proxies form the basis of a broad based preparation for what is expected to be a significantly different computing environment. Understanding the conditions under which a miniapp can be confidently predictive of an applications behavior must be clearly elucidated.

Toward this end, we have developed a methodology for assessing the predictive capabilities of application proxies [2]. Adhering to the spirit of experimental validation, our approach provides a framework for examining data from the application with that provided by their proxies. In this poster we present this methodology, and apply it to three miniapps developed by the Mantevo project [3].



Key Question: Under what conditions does a miniapp represent a key performance characteristic in a full app?

Methodology adheres to the spirit of experimental validation:

- *App is the "real world" observation, miniapp is the "model", and requires*
- *extensive knowledge of, and experience developing, executing, profiling, maintaining, and extending multi-scale, multi-physics scientific and engineering application software, targeting highest performance computing platforms,*
- *a strong understanding of the miniapps and their intended use: what they are intended to represent and what they are not intended to represent, and*
- *a formal verification and validation (V&V) methodology that lets us examine experimental and predicted data.*

For *diagnostics* $\{D\} = D_1, D_2, \ldots, D_n$,
baseline observations $\{B\} = B_1, B_2, \ldots, B_n$ and
miniapp measurements $\{A\} = A_1, A_2, \ldots, A_n$,
Then

$$X_i = \| B_i - A_i \|_i \text{, for all } i$$

$$V_i = \begin{cases} \textit{pass}, \text{ for } T^1_i < X_i < T^2_i \\ \textit{caution}, \text{ for } T^2_i < X_i < T^3_i \\ \textit{fail}, \text{ for } X_i > T^3_i, \end{cases} \quad \text{for thresholds T.}$$

Verification is the process of determining that a model implementation accurately represents the developers conceptual description of the model and the solution to the model. Validation is the process of determining the degree to which a model is an accurate representation of the "real world" (in this case the performance characteristics of the "real" application) from the perspective of the intended uses of the model[1].

Assumptions:

1) The application code has been verified and validated.
2) The miniapp has been verified.

That said, applying this methodology can expose issues with regard to these assumptions.

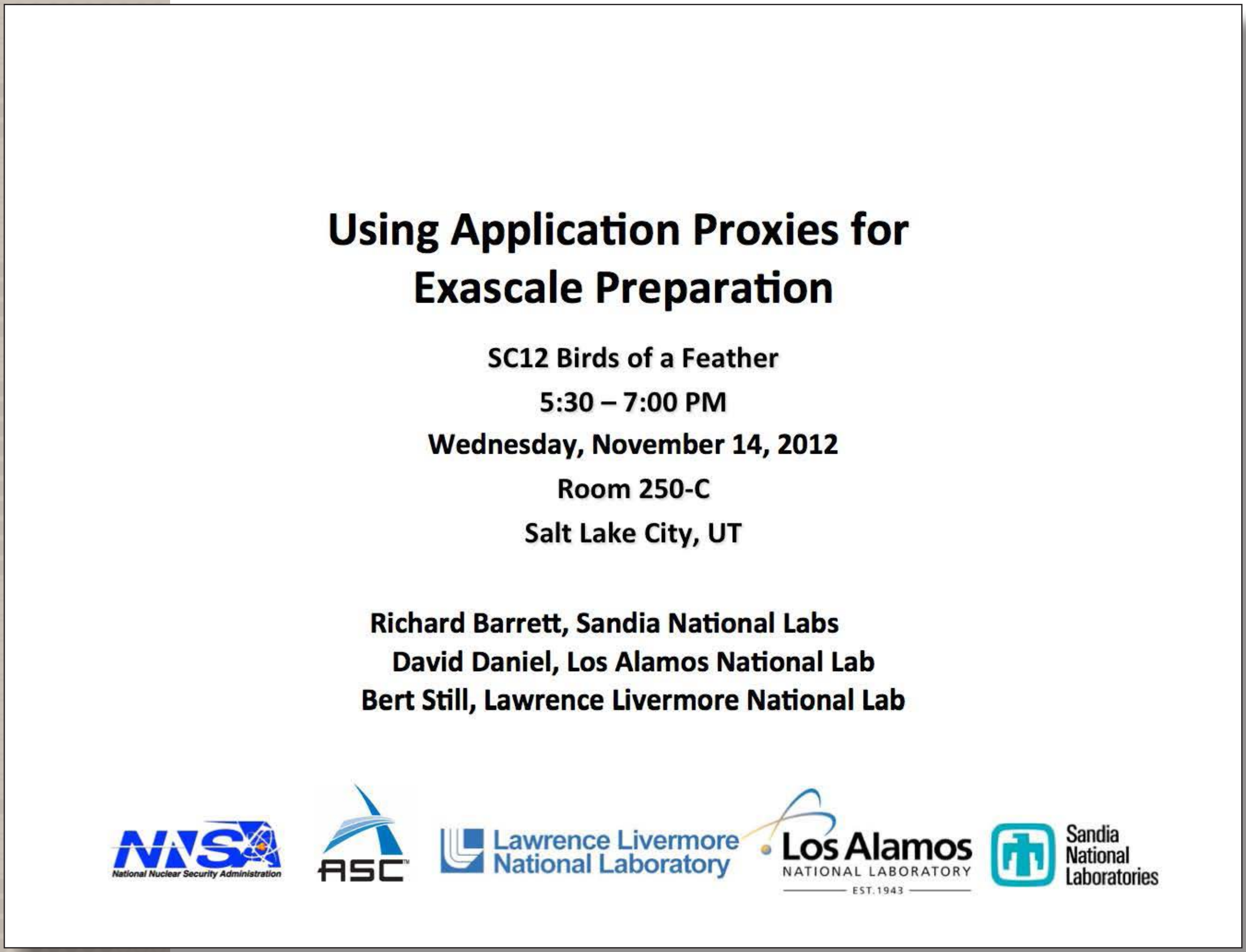
[1] "*The structural simulation toolkit*", A.F. Rodrigues, K.S. Hemmert, B.W. Barrett, C. Kersey, R. Oldfield, M. Weston, R. Risen, J. Cook, P. Rosenfeld, E. Cooper-Balis, B. Jacob, SIGMETRICS Perform. Eval. Rev., 38, pp. 37-42, 2011.
[2] "*Assessing the Predictive Capabilities of Application Proxies*", R.Barrett, P.S. Crozier, D.W. Doerfler, S.D. Hammond, M.A. Heroux, H.K. Thornquist, T.G. Trucano, and C.T. Vaughan. In preparation.
[3] "Improving Performance via Mini-applications", 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, Sandia Technical Report SAND2009-5574, 2009.

[1] American Society of Mechanical Engineers (ASME, 2006) and the American Institute of Aeronautics and Astronautics (AIAA, 1998)), and this usage has basically been adopted by the United States Departments of Energy (DOE) and Defense (DoD).

Science and Art: Toward building a body of evidence

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

Appendix D: Publications (cont.)



Using Application Proxies for Exascale Preparation

SC12 Birds of a Feather

5:30 – 7:00 PM

Wednesday, November 14, 2012

Room 250-C

Salt Lake City, UT

Richard Barrett, Sandia National Labs

David Daniel, Los Alamos National Lab

Bert Still, Lawrence Livermore National Lab

NNSA National Nuclear Security Administration

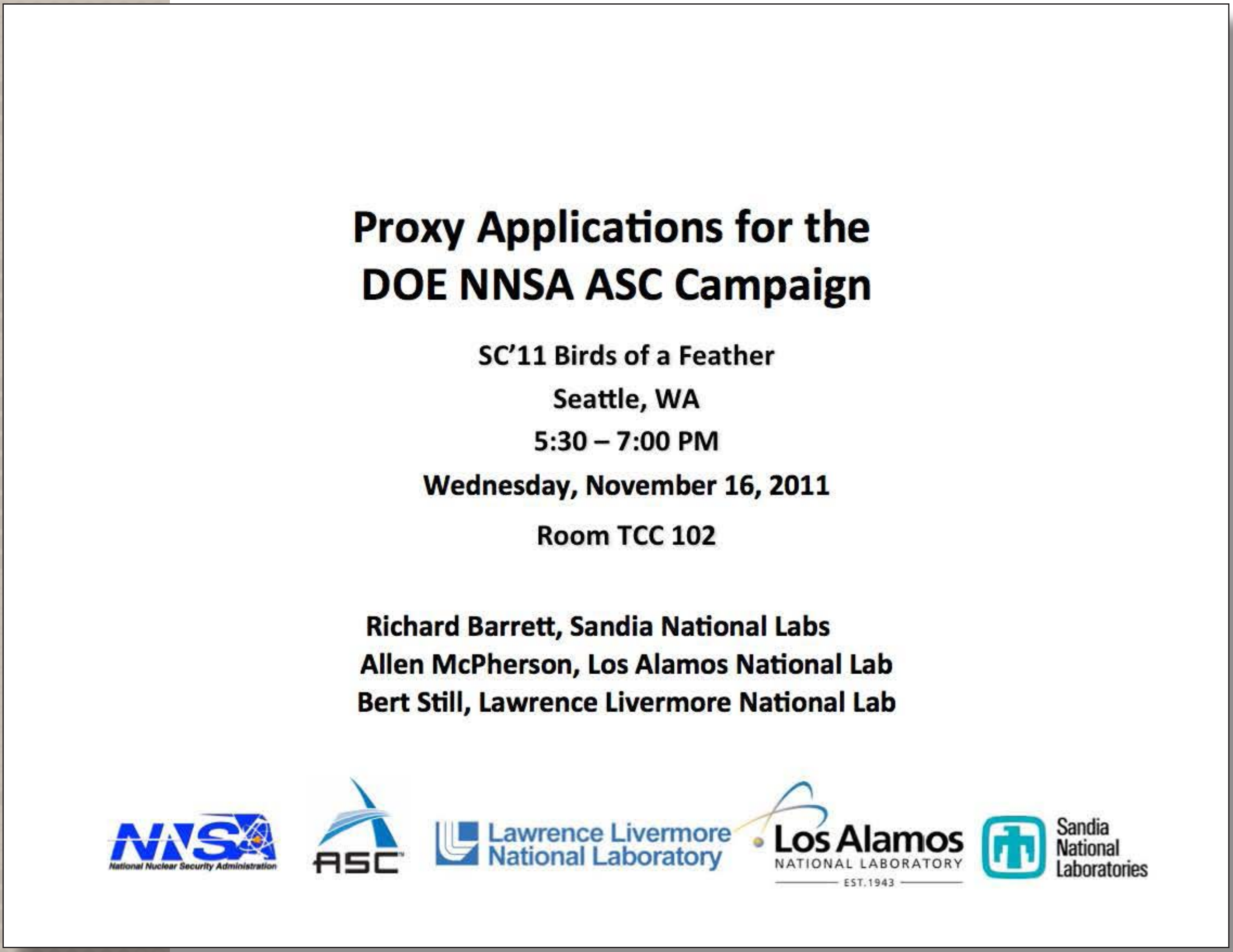
ASC

Lawrence Livermore National Laboratory

Los Alamos NATIONAL LABORATORY EST. 1943

Sandia National Laboratories

Appendix D: Publications (cont.)



Proxy Applications for the DOE NNSA ASC Campaign

SC'11 Birds of a Feather

Seattle, WA

5:30 – 7:00 PM

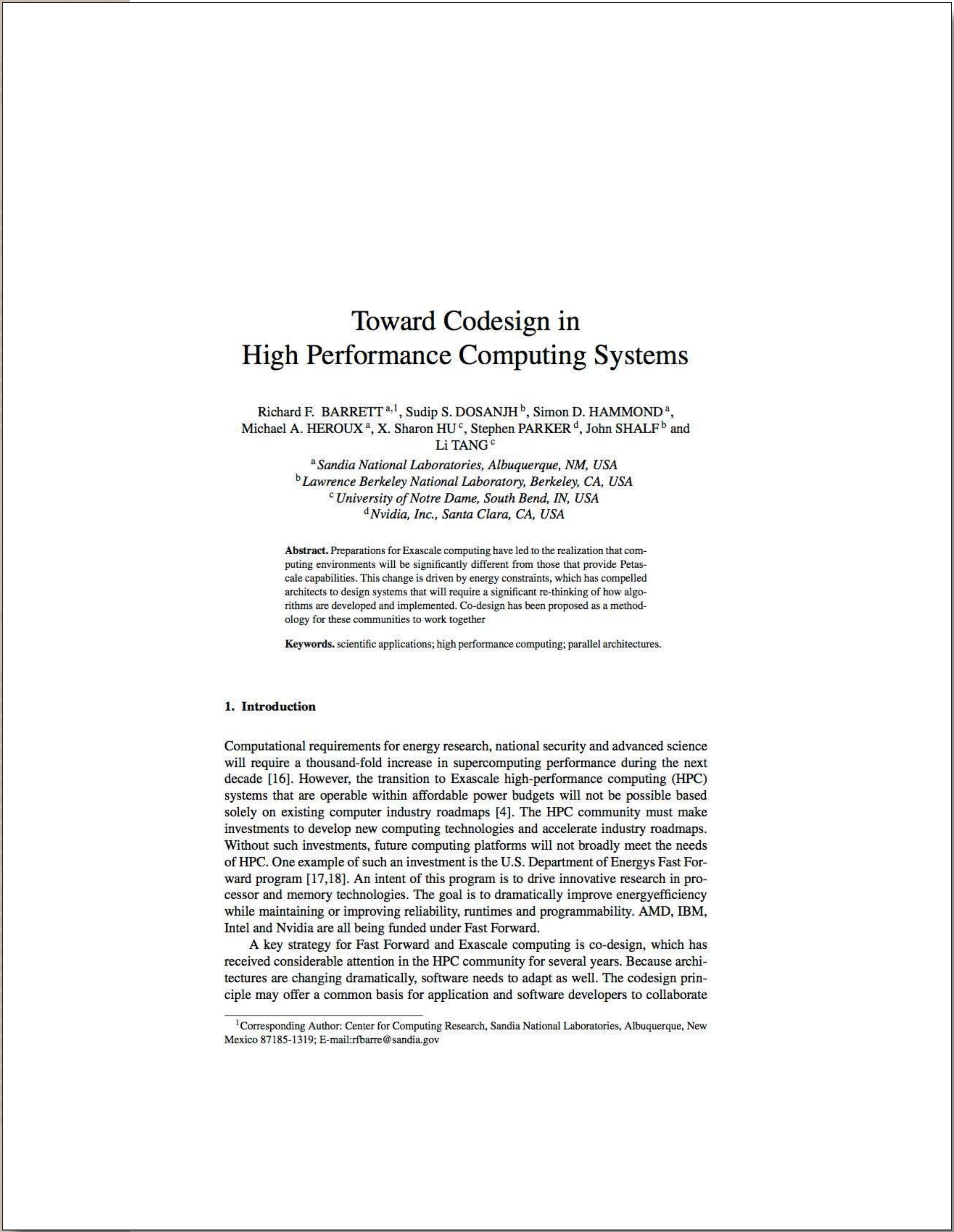
Wednesday, November 16, 2011

Room TCC 102

Richard Barrett, Sandia National Labs
Allen McPherson, Los Alamos National Lab
Bert Still, Lawrence Livermore National Lab

NNSA
National Nuclear Security Administration
ASC
Lawrence Livermore National Laboratory
Los Alamos NATIONAL LABORATORY EST.1943
Sandia National Laboratories

Appendix D: Publications (cont.)



Toward Codesign in High Performance Computing Systems

Richard F. BARRETT [a,1], Sudip S. DOSANJH [b], Simon D. HAMMOND [a], Michael A. HEROUX [a], X. Sharon HU [c], Stephen PARKER [d], John SHALF [b] and Li TANG [c]

[a] *Sandia National Laboratories, Albuquerque, NM, USA*
[b] *Lawrence Berkeley National Laboratory, Berkeley, CA, USA*
[c] *University of Notre Dame, South Bend, IN, USA*
[d] *Nvidia, Inc., Santa Clara, CA, USA*

Abstract. Preparations for Exascale computing have led to the realization that computing environments will be significantly different from those that provide Petascale capabilities. This change is driven by energy constraints, which has compelled architects to design systems that will require a significant re-thinking of how algorithms are developed and implemented. Co-design has been proposed as a methodology for these communities to work together

Keywords. scientific applications; high performance computing; parallel architectures.

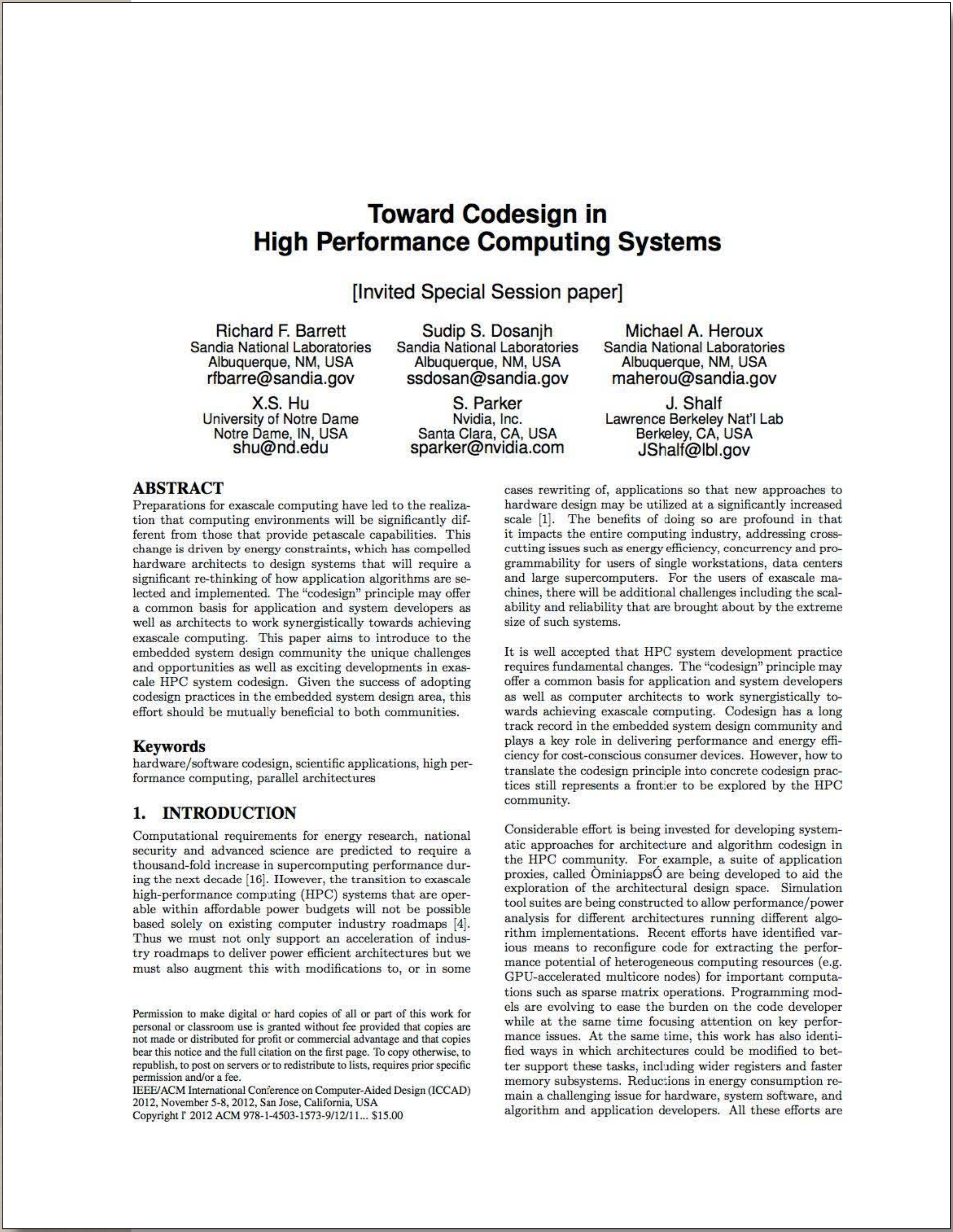
1. Introduction

Computational requirements for energy research, national security and advanced science will require a thousand-fold increase in supercomputing performance during the next decade [16]. However, the transition to Exascale high-performance computing (HPC) systems that are operable within affordable power budgets will not be possible based solely on existing computer industry roadmaps [4]. The HPC community must make investments to develop new computing technologies and accelerate industry roadmaps. Without such investments, future computing platforms will not broadly meet the needs of HPC. One example of such an investment is the U.S. Department of Energys Fast Forward program [17,18]. An intent of this program is to drive innovative research in processor and memory technologies. The goal is to dramatically improve energyefficiency while maintaining or improving reliability, runtimes and programmability. AMD, IBM, Intel and Nvidia are all being funded under Fast Forward.

A key strategy for Fast Forward and Exascale computing is co-design, which has received considerable attention in the HPC community for several years. Because architectures are changing dramatically, software needs to adapt as well. The codesign principle may offer a common basis for application and software developers to collaborate

[1]Corresponding Author: Center for Computing Research, Sandia National Laboratories, Albuquerque, New Mexico 87185-1319; E-mail:rfbarre@sandia.gov

Appendix D: Publications (cont.)



Toward Codesign in High Performance Computing Systems

[Invited Special Session paper]

Richard F. Barrett
Sandia National Laboratories
Albuquerque, NM, USA
rfbarre@sandia.gov

Sudip S. Dosanjh
Sandia National Laboratories
Albuquerque, NM, USA
ssdosan@sandia.gov

Michael A. Heroux
Sandia National Laboratories
Albuquerque, NM, USA
maherou@sandia.gov

X.S. Hu
University of Notre Dame
Notre Dame, IN, USA
shu@nd.edu

S. Parker
Nvidia, Inc.
Santa Clara, CA, USA
sparker@nvidia.com

J. Shalf
Lawrence Berkeley Nat'l Lab
Berkeley, CA, USA
JShalf@lbl.gov

ABSTRACT

Preparations for exascale computing have led to the realization that computing environments will be significantly different from those that provide petascale capabilities. This change is driven by energy constraints, which has compelled hardware architects to design systems that will require a significant re-thinking of how application algorithms are selected and implemented. The "codesign" principle may offer a common basis for application and system developers as well as architects to work synergistically towards achieving exascale computing. This paper aims to introduce to the embedded system design community the unique challenges and opportunities as well as exciting developments in exascale HPC system codesign. Given the success of adopting codesign practices in the embedded system design area, this effort should be mutually beneficial to both communities.

Keywords

hardware/software codesign, scientific applications, high performance computing, parallel architectures

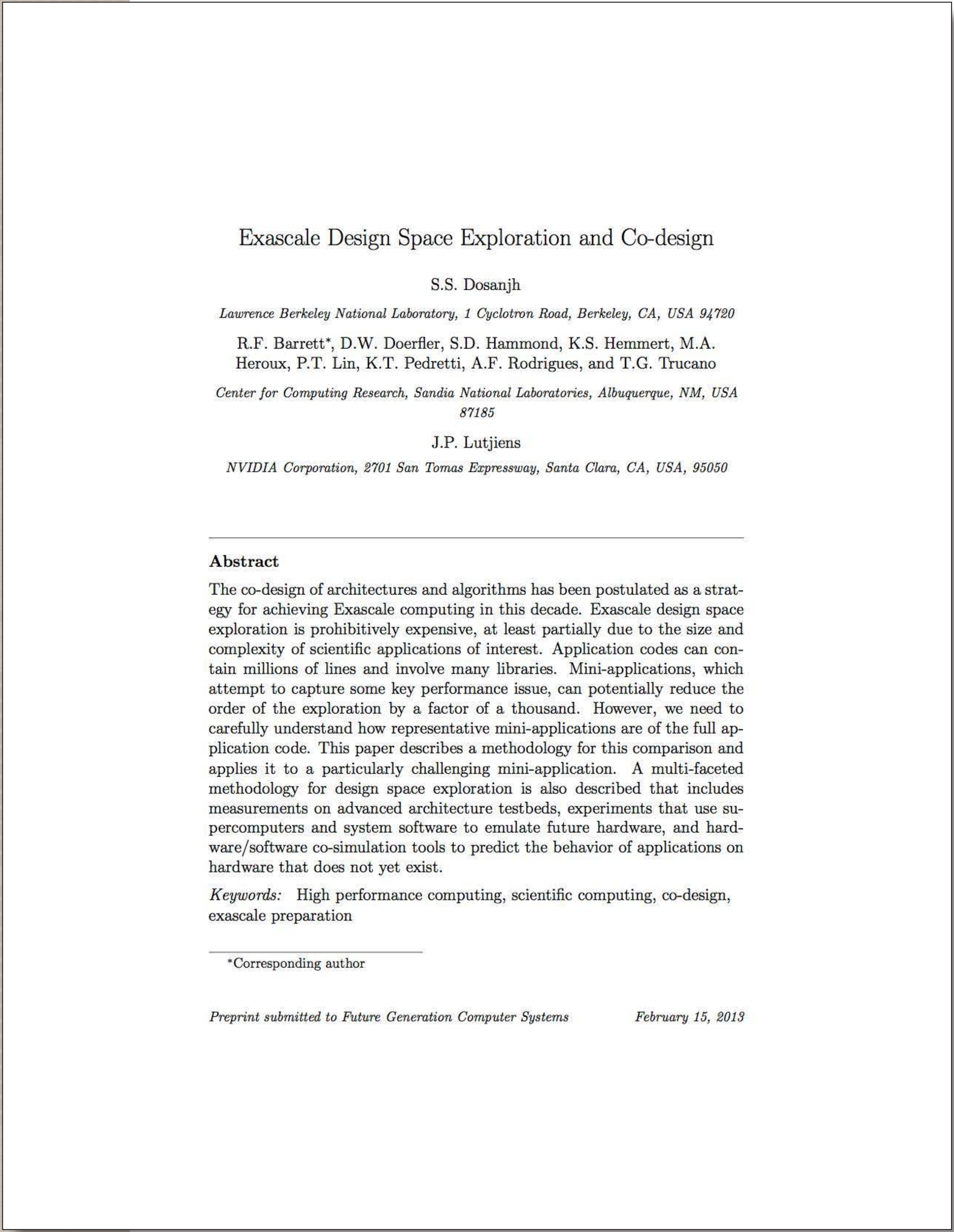
1. INTRODUCTION

Computational requirements for energy research, national security and advanced science are predicted to require a thousand-fold increase in supercomputing performance during the next decade [16]. However, the transition to exascale high-performance computing (HPC) systems that are operable within affordable power budgets will not be possible based solely on existing computer industry roadmaps [4]. Thus we must not only support an acceleration of industry roadmaps to deliver power efficient architectures but we must also augment this with modifications to, or in some cases rewriting of, applications so that new approaches to hardware design may be utilized at a significantly increased scale [1]. The benefits of doing so are profound in that it impacts the entire computing industry, addressing cross-cutting issues such as energy efficiency, concurrency and programmability for users of single workstations, data centers and large supercomputers. For the users of exascale machines, there will be additional challenges including the scalability and reliability that are brought about by the extreme size of such systems.

It is well accepted that HPC system development practice requires fundamental changes. The "codesign" principle may offer a common basis for application and system developers as well as computer architects to work synergistically towards achieving exascale computing. Codesign has a long track record in the embedded system design community and plays a key role in delivering performance and energy efficiency for cost-conscious consumer devices. However, how to translate the codesign principle into concrete codesign practices still represents a frontier to be explored by the HPC community.

Considerable effort is being invested for developing systematic approaches for architecture and algorithm codesign in the HPC community. For example, a suite of application proxies, called ÒminiappsÓ are being developed to aid the exploration of the architectural design space. Simulation tool suites are being constructed to allow performance/power analysis for different architectures running different algorithm implementations. Recent efforts have identified various means to reconfigure code for extracting the performance potential of heterogeneous computing resources (e.g. GPU-accelerated multicore nodes) for important computations such as sparse matrix operations. Programming models are evolving to ease the burden on the code developer while at the same time focusing attention on key performance issues. At the same time, this work has also identified ways in which architectures could be modified to better support these tasks, including wider registers and faster memory subsystems. Reductions in energy consumption remain a challenging issue for hardware, system software, and algorithm and application developers. All these efforts are

Appendix D: Publications (cont.)



Exascale Design Space Exploration and Co-design

S.S. Dosanjh

Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA, USA 94720

R.F. Barrett*, D.W. Doerfler, S.D. Hammond, K.S. Hemmert, M.A. Heroux, P.T. Lin, K.T. Pedretti, A.F. Rodrigues, and T.G. Trucano

Center for Computing Research, Sandia National Laboratories, Albuquerque, NM, USA 87185

J.P. Lutjiens

NVIDIA Corporation, 2701 San Tomas Expressway, Santa Clara, CA, USA, 95050

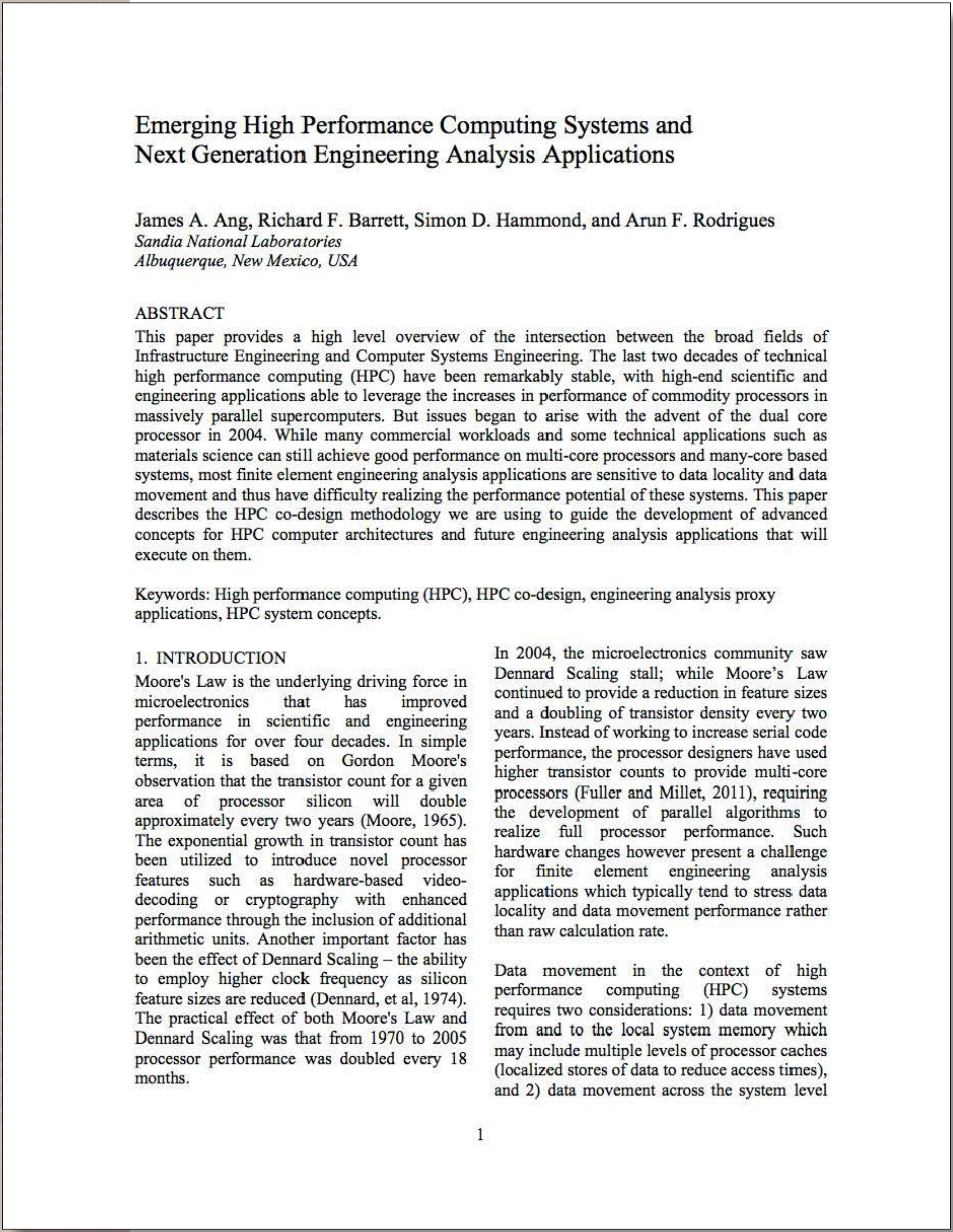
Abstract

The co-design of architectures and algorithms has been postulated as a strategy for achieving Exascale computing in this decade. Exascale design space exploration is prohibitively expensive, at least partially due to the size and complexity of scientific applications of interest. Application codes can contain millions of lines and involve many libraries. Mini-applications, which attempt to capture some key performance issue, can potentially reduce the order of the exploration by a factor of a thousand. However, we need to carefully understand how representative mini-applications are of the full application code. This paper describes a methodology for this comparison and applies it to a particularly challenging mini-application. A multi-faceted methodology for design space exploration is also described that includes measurements on advanced architecture testbeds, experiments that use supercomputers and system software to emulate future hardware, and hardware/software co-simulation tools to predict the behavior of applications on hardware that does not yet exist.

Keywords: High performance computing, scientific computing, co-design, exascale preparation

*Corresponding author

Appendix D: Publications (cont.)



Emerging High Performance Computing Systems and Next Generation Engineering Analysis Applications

James A. Ang, Richard F. Barrett, Simon D. Hammond, and Arun F. Rodrigues
Sandia National Laboratories
Albuquerque, New Mexico, USA

ABSTRACT

This paper provides a high level overview of the intersection between the broad fields of Infrastructure Engineering and Computer Systems Engineering. The last two decades of technical high performance computing (HPC) have been remarkably stable, with high-end scientific and engineering applications able to leverage the increases in performance of commodity processors in massively parallel supercomputers. But issues began to arise with the advent of the dual core processor in 2004. While many commercial workloads and some technical applications such as materials science can still achieve good performance on multi-core processors and many-core based systems, most finite element engineering analysis applications are sensitive to data locality and data movement and thus have difficulty realizing the performance potential of these systems. This paper describes the HPC co-design methodology we are using to guide the development of advanced concepts for HPC computer architectures and future engineering analysis applications that will execute on them.

Keywords: High performance computing (HPC), HPC co-design, engineering analysis proxy applications, HPC system concepts.

1. INTRODUCTION

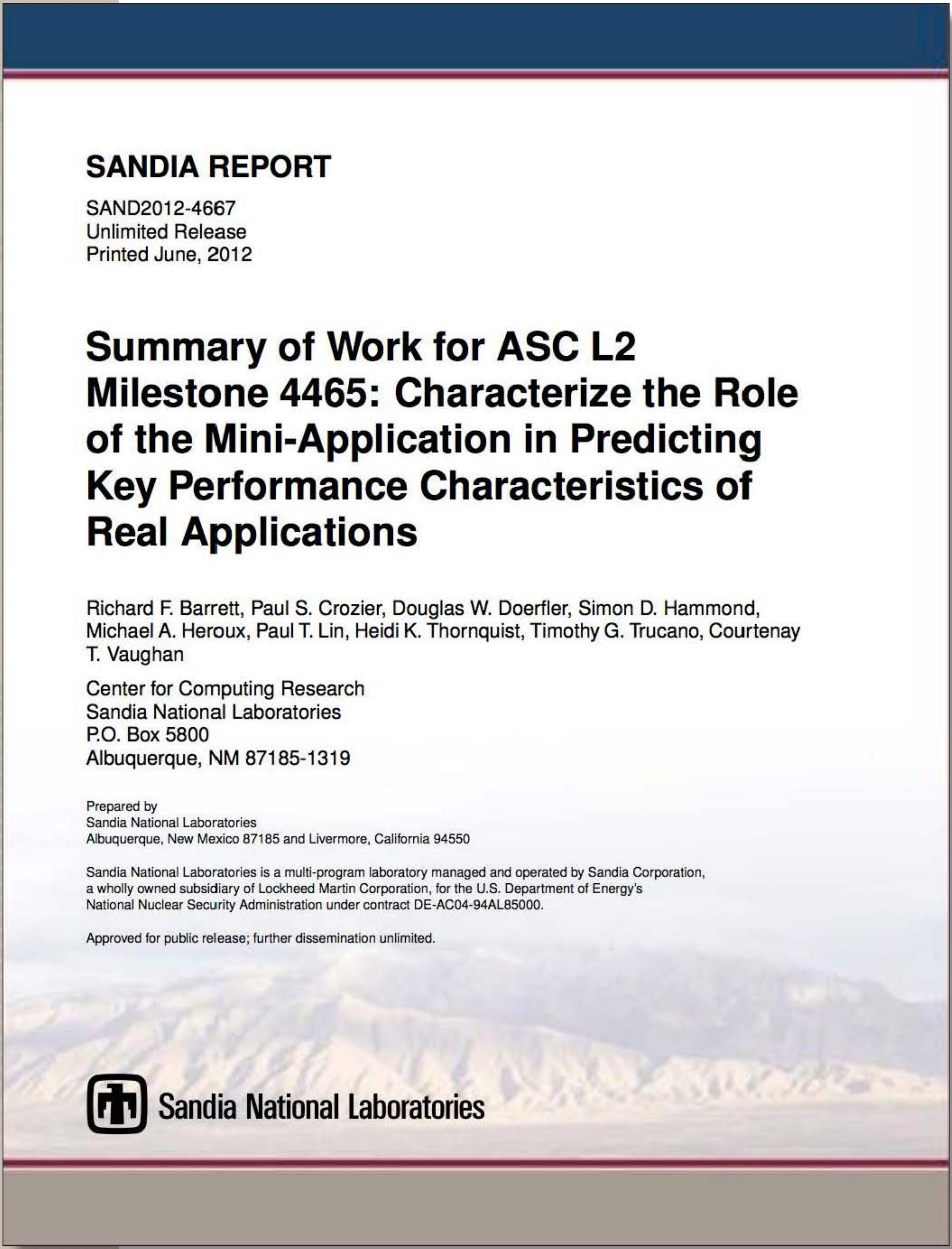
Moore's Law is the underlying driving force in microelectronics that has improved performance in scientific and engineering applications for over four decades. In simple terms, it is based on Gordon Moore's observation that the transistor count for a given area of processor silicon will double approximately every two years (Moore, 1965). The exponential growth in transistor count has been utilized to introduce novel processor features such as hardware-based video-decoding or cryptography with enhanced performance through the inclusion of additional arithmetic units. Another important factor has been the effect of Dennard Scaling – the ability to employ higher clock frequency as silicon feature sizes are reduced (Dennard, et al, 1974). The practical effect of both Moore's Law and Dennard Scaling was that from 1970 to 2005 processor performance was doubled every 18 months.

In 2004, the microelectronics community saw Dennard Scaling stall; while Moore's Law continued to provide a reduction in feature sizes and a doubling of transistor density every two years. Instead of working to increase serial code performance, the processor designers have used higher transistor counts to provide multi-core processors (Fuller and Millet, 2011), requiring the development of parallel algorithms to realize full processor performance. Such hardware changes however present a challenge for finite element engineering analysis applications which typically tend to stress data locality and data movement performance rather than raw calculation rate.

Data movement in the context of high performance computing (HPC) systems requires two considerations: 1) data movement from and to the local system memory which may include multiple levels of processor caches (localized stores of data to reduce access times), and 2) data movement across the system level

1

Appendix D: Publications (cont.)



SANDIA REPORT

SAND2012-4667
Unlimited Release
Printed June, 2012

Summary of Work for ASC L2 Milestone 4465: Characterize the Role of the Mini-Application in Predicting Key Performance Characteristics of Real Applications

Richard F. Barrett, Paul S. Crozier, Douglas W. Doerfler, Simon D. Hammond, Michael A. Heroux, Paul T. Lin, Heidi K. Thornquist, Timothy G. Trucano, Courtenay T. Vaughan

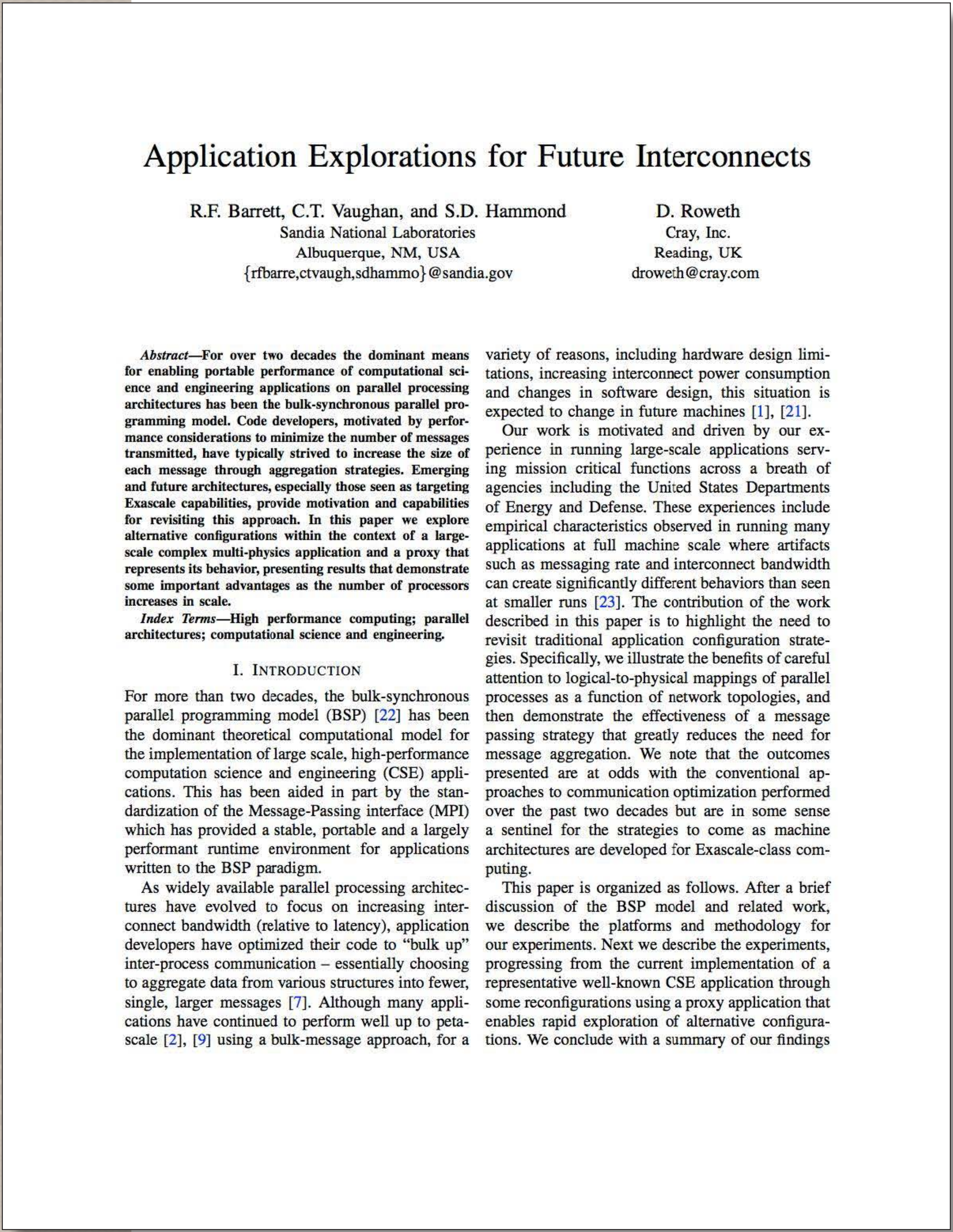
Center for Computing Research
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1319

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

Appendix D: Publications (cont.)



Application Explorations for Future Interconnects

R.F. Barrett, C.T. Vaughan, and S.D. Hammond
Sandia National Laboratories
Albuquerque, NM, USA
{rfbarre,ctvaugh,sdhammo}@sandia.gov

D. Roweth
Cray, Inc.
Reading, UK
droweth@cray.com

***Abstract*—For over two decades the dominant means for enabling portable performance of computational science and engineering applications on parallel processing architectures has been the bulk-synchronous parallel programming model. Code developers, motivated by performance considerations to minimize the number of messages transmitted, have typically strived to increase the size of each message through aggregation strategies. Emerging and future architectures, especially those seen as targeting Exascale capabilities, provide motivation and capabilities for revisiting this approach. In this paper we explore alternative configurations within the context of a large-scale complex multi-physics application and a proxy that represents its behavior, presenting results that demonstrate some important advantages as the number of processors increases in scale.**

***Index Terms*—High performance computing; parallel architectures; computational science and engineering.**

I. Introduction

For more than two decades, the bulk-synchronous parallel programming model (BSP) [22] has been the dominant theoretical computational model for the implementation of large scale, high-performance computation science and engineering (CSE) applications. This has been aided in part by the standardization of the Message-Passing interface (MPI) which has provided a stable, portable and a largely performant runtime environment for applications written to the BSP paradigm.

As widely available parallel processing architectures have evolved to focus on increasing interconnect bandwidth (relative to latency), application developers have optimized their code to "bulk up" inter-process communication – essentially choosing to aggregate data from various structures into fewer, single, larger messages [7]. Although many applications have continued to perform well up to petascale [2], [9] using a bulk-message approach, for a variety of reasons, including hardware design limitations, increasing interconnect power consumption and changes in software design, this situation is expected to change in future machines [1], [21].

Our work is motivated and driven by our experience in running large-scale applications serving mission critical functions across a breath of agencies including the United States Departments of Energy and Defense. These experiences include empirical characteristics observed in running many applications at full machine scale where artifacts such as messaging rate and interconnect bandwidth can create significantly different behaviors than seen at smaller runs [23]. The contribution of the work described in this paper is to highlight the need to revisit traditional application configuration strategies. Specifically, we illustrate the benefits of careful attention to logical-to-physical mappings of parallel processes as a function of network topologies, and then demonstrate the effectiveness of a message passing strategy that greatly reduces the need for message aggregation. We note that the outcomes presented are at odds with the conventional approaches to communication optimization performed over the past two decades but are in some sense a sentinel for the strategies to come as machine architectures are developed for Exascale-class computing.

This paper is organized as follows. After a brief discussion of the BSP model and related work, we describe the platforms and methodology for our experiments. Next we describe the experiments, progressing from the current implementation of a representative well-known CSE application through some reconfigurations using a proxy application that enables rapid exploration of alternative configurations. We conclude with a summary of our findings

Appendix D: Publications (cont.)

Assessing the Role of Mini-Applications in Predicting Key Performance Characteristics of Scientific and Engineering Applications

R.F. Barrett*, Paul S. Crozier, Douglas W. Doerfler, Simon D. Hammond, Michael A. Heroux, Paul T. Lin, Heidi K. Thornquist, Timothy G. Trucano, and Courtenay T. Vaughan

Center for Computing Research, Sandia National Laboratories, Albuquerque, NM, USA 87185

Abstract

High Performance Computing architectures are undergoing dramatic changes. Driven by power constraints as well as ..., the means for extracting the performance potential of these computers is, thus far, a research problem. Computational science and engineering application programs are typically large, complex, and dynamic, and often constrained by distribution limitations

As a means of making tractable rapid explorations of scientific and engineering application programs in the context of new, emerging, and future computing architectures, a suite of "mini-apps" has been created to serve as proxies for full scale applications. Each miniapp is designed to represent a key performance characteristic that does or is expected to significantly impact the runtime performance of a scientific or engineering application program. In this paper we introduce a methodology for assessing the ability of these mini-apps to represent these performance issues, and apply the methodology to a set of applications.

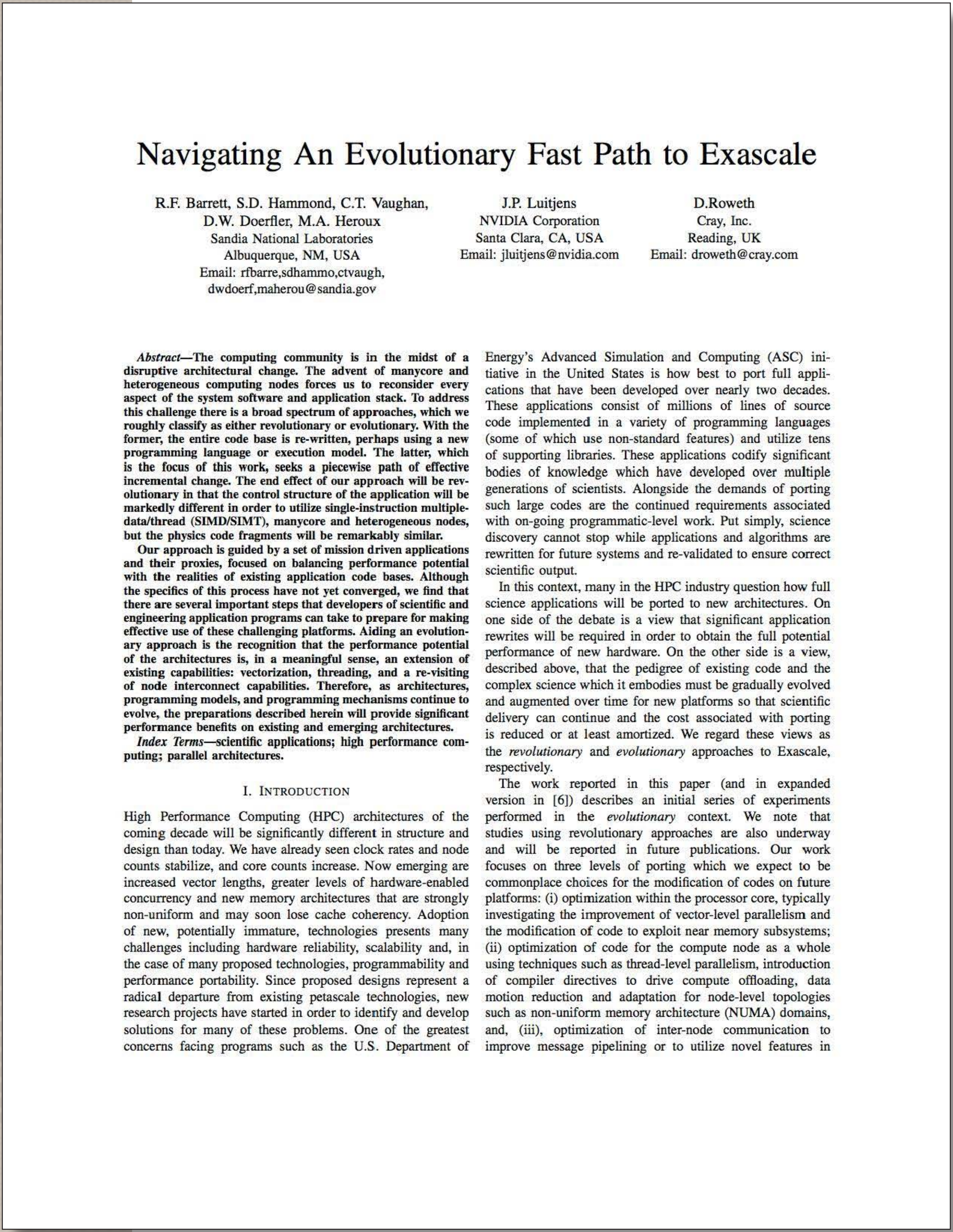
Keywords: High performance computing, scientific computing, co-design, exascale preparation

1. Introduction

Scientific and engineering applications typically out live the computing environments they originally targeted. Over the past several years computer architectures typically employed have remained relatively stable, with subsequent generations characterized by faster processors, memories, and interconnects, and have typically resulted in predictably faster runtimes. Emerging and expected future architectures, however, are presenting special challenges and opportunities that, if not effectively exploited, could result in slower runtimes. Driven

*Corresponding author

Appendix D: Publications (cont.)



Navigating An Evolutionary Fast Path to Exascale

R.F. Barrett, S.D. Hammond, C.T. Vaughan,
D.W. Doerfler, M.A. Heroux
Sandia National Laboratories
Albuquerque, NM, USA
Email: rfbarre,sdhammo,ctvaugh,
dwdoerf,maherou@sandia.gov

J.P. Luitjens
NVIDIA Corporation
Santa Clara, CA, USA
Email: jluitjens@nvidia.com

D.Roweth
Cray, Inc.
Reading, UK
Email: droweth@cray.com

***Abstract*—The computing community is in the midst of a disruptive architectural change. The advent of manycore and heterogeneous computing nodes forces us to reconsider every aspect of the system software and application stack. To address this challenge there is a broad spectrum of approaches, which we roughly classify as either revolutionary or evolutionary. With the former, the entire code base is re-written, perhaps using a new programming language or execution model. The latter, which is the focus of this work, seeks a piecewise path of effective incremental change. The end effect of our approach will be revolutionary in that the control structure of the application will be markedly different in order to utilize single-instruction multiple-data/thread (SIMD/SIMT), manycore and heterogeneous nodes, but the physics code fragments will be remarkably similar.**

Our approach is guided by a set of mission driven applications and their proxies, focused on balancing performance potential with the realities of existing application code bases. Although the specifics of this process have not yet converged, we find that there are several important steps that developers of scientific and engineering application programs can take to prepare for making effective use of these challenging platforms. Aiding an evolutionary approach is the recognition that the performance potential of the architectures is, in a meaningful sense, an extension of existing capabilities: vectorization, threading, and a re-visiting of node interconnect capabilities. Therefore, as architectures, programming models, and programming mechanisms continue to evolve, the preparations described herein will provide significant performance benefits on existing and emerging architectures.

***Index Terms*—scientific applications; high performance computing; parallel architectures.**

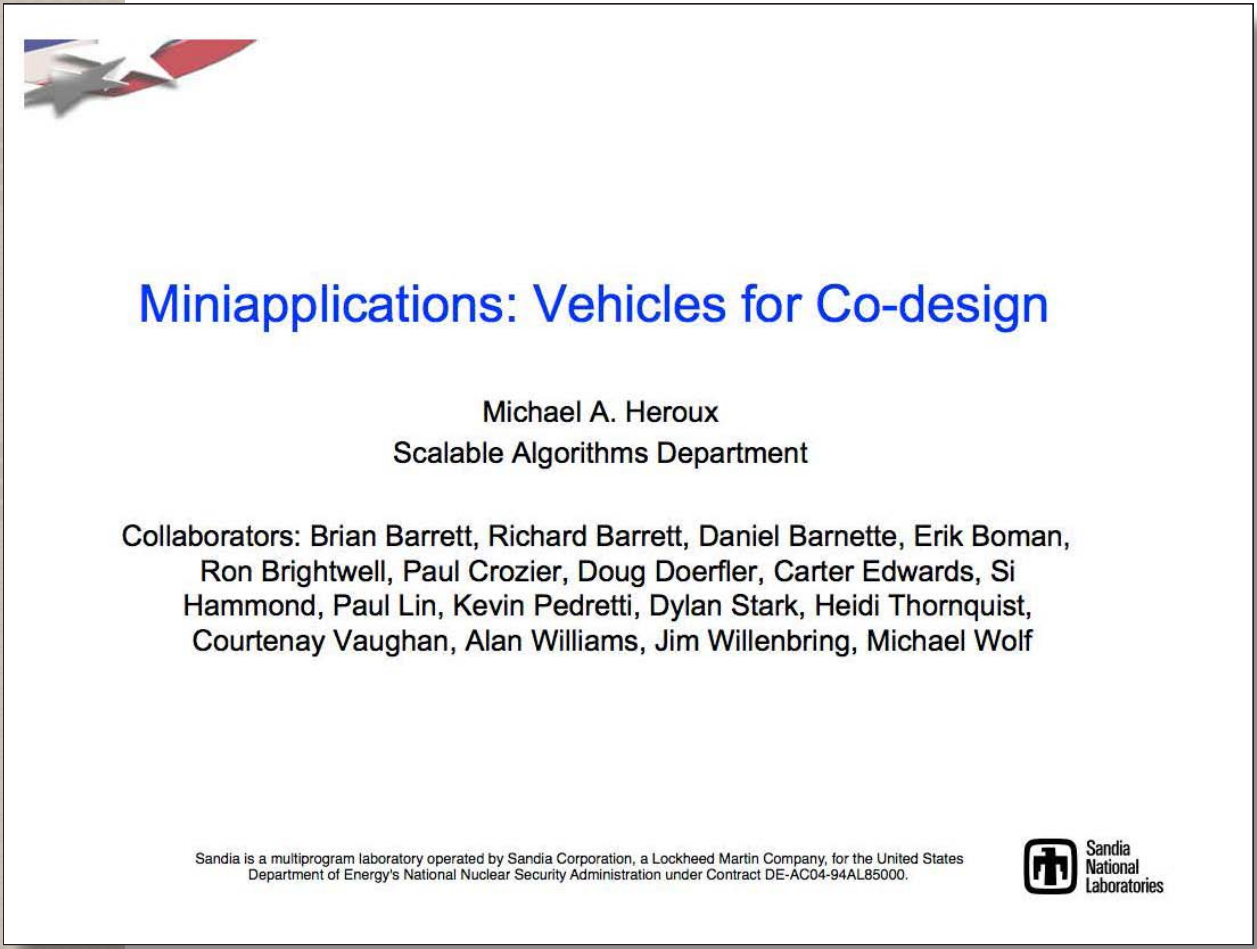
I. Introduction

High Performance Computing (HPC) architectures of the coming decade will be significantly different in structure and design than today. We have already seen clock rates and node counts stabilize, and core counts increase. Now emerging are increased vector lengths, greater levels of hardware-enabled concurrency and new memory architectures that are strongly non-uniform and may soon lose cache coherency. Adoption of new, potentially immature, technologies presents many challenges including hardware reliability, scalability and, in the case of many proposed technologies, programmability and performance portability. Since proposed designs represent a radical departure from existing petascale technologies, new research projects have started in order to identify and develop solutions for many of these problems. One of the greatest concerns facing programs such as the U.S. Department of Energy's Advanced Simulation and Computing (ASC) initiative in the United States is how best to port full applications that have been developed over nearly two decades. These applications consist of millions of lines of source code implemented in a variety of programming languages (some of which use non-standard features) and utilize tens of supporting libraries. These applications codify significant bodies of knowledge which have developed over multiple generations of scientists. Alongside the demands of porting such large codes are the continued requirements associated with on-going programmatic-level work. Put simply, science discovery cannot stop while applications and algorithms are rewritten for future systems and re-validated to ensure correct scientific output.

In this context, many in the HPC industry question how full science applications will be ported to new architectures. On one side of the debate is a view that significant application rewrites will be required in order to obtain the full potential performance of new hardware. On the other side is a view, described above, that the pedigree of existing code and the complex science which it embodies must be gradually evolved and augmented over time for new platforms so that scientific delivery can continue and the cost associated with porting is reduced or at least amortized. We regard these views as the *revolutionary* and *evolutionary* approaches to Exascale, respectively.

The work reported in this paper (and in expanded version in [6]) describes an initial series of experiments performed in the *evolutionary* context. We note that studies using revolutionary approaches are also underway and will be reported in future publications. Our work focuses on three levels of porting which we expect to be commonplace choices for the modification of codes on future platforms: (i) optimization within the processor core, typically investigating the improvement of vector-level parallelism and the modification of code to exploit near memory subsystems; (ii) optimization of code for the compute node as a whole using techniques such as thread-level parallelism, introduction of compiler directives to drive compute offloading, data motion reduction and adaptation for node-level topologies such as non-uniform memory architecture (NUMA) domains, and, (iii), optimization of inter-node communication to improve message pipelining or to utilize novel features in

Appendix D: Publications (cont.)

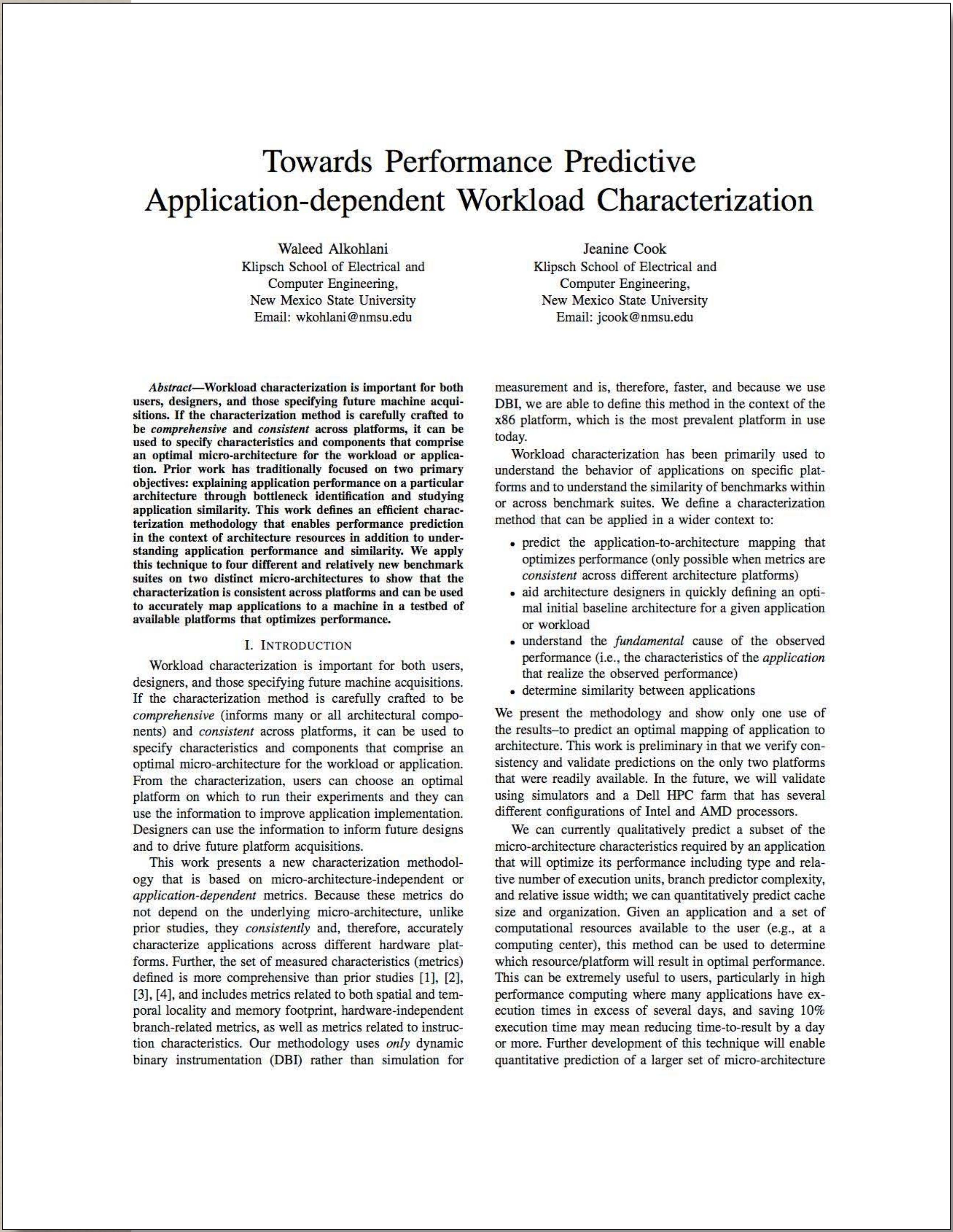


Miniapplications: Vehicles for Co-design

Michael A. Heroux
Scalable Algorithms Department

Collaborators: Brian Barrett, Richard Barrett, Daniel Barnette, Erik Boman, Ron Brightwell, Paul Crozier, Doug Doerfler, Carter Edwards, Si Hammond, Paul Lin, Kevin Pedretti, Dylan Stark, Heidi Thornquist, Courtenay Vaughan, Alan Williams, Jim Willenbring, Michael Wolf

Appendix D: Publications (cont.)



Towards Performance Predictive Application-dependent Workload Characterization

Waleed Alkohlani
Klipsch School of Electrical and Computer Engineering,
New Mexico State University
Email: wkohlani@nmsu.edu

Jeanine Cook
Klipsch School of Electrical and Computer Engineering,
New Mexico State University
Email: jcook@nmsu.edu

***Abstract*—Workload characterization is important for both users, designers, and those specifying future machine acquisitions. If the characterization method is carefully crafted to be *comprehensive* and *consistent* across platforms, it can be used to specify characteristics and components that comprise an optimal micro-architecture for the workload or application. Prior work has traditionally focused on two primary objectives: explaining application performance on a particular architecture through bottleneck identification and studying application similarity. This work defines an efficient characterization methodology that enables performance prediction in the context of architecture resources in addition to understanding application performance and similarity. We apply this technique to four different and relatively new benchmark suites on two distinct micro-architectures to show that the characterization is consistent across platforms and can be used to accurately map applications to a machine in a testbed of available platforms that optimizes performance.**

I. Introduction

Workload characterization is important for both users, designers, and those specifying future machine acquisitions. If the characterization method is carefully crafted to be *comprehensive* (informs many or all architectural components) and *consistent* across platforms, it can be used to specify characteristics and components that comprise an optimal micro-architecture for the workload or application. From the characterization, users can choose an optimal platform on which to run their experiments and they can use the information to improve application implementation. Designers can use the information to inform future designs and to drive future platform acquisitions.

This work presents a new characterization methodology that is based on micro-architecture-independent or *application-dependent* metrics. Because these metrics do not depend on the underlying micro-architecture, unlike prior studies, they *consistently* and, therefore, accurately characterize applications across different hardware platforms. Further, the set of measured characteristics (metrics) defined is more comprehensive than prior studies [1], [2], [3], [4], and includes metrics related to both spatial and temporal locality and memory footprint, hardware-independent branch-related metrics, as well as metrics related to instruction characteristics. Our methodology uses *only* dynamic binary instrumentation (DBI) rather than simulation for measurement and is, therefore, faster, and because we use DBI, we are able to define this method in the context of the x86 platform, which is the most prevalent platform in use today.

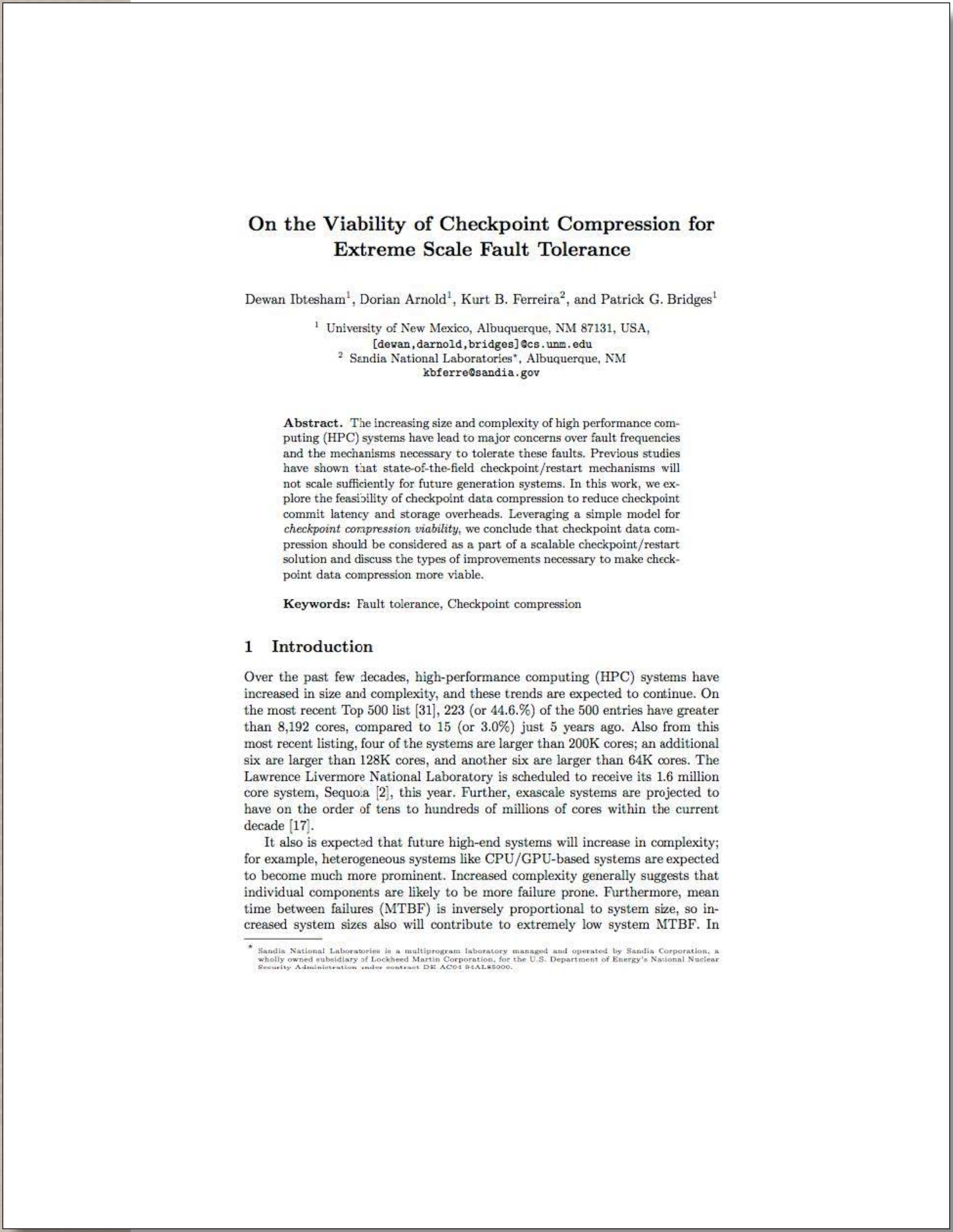
Workload characterization has been primarily used to understand the behavior of applications on specific platforms and to understand the similarity of benchmarks within or across benchmark suites. We define a characterization method that can be applied in a wider context to:

- predict the application-to-architecture mapping that optimizes performance (only possible when metrics are *consistent* across different architecture platforms)
- aid architecture designers in quickly defining an optimal initial baseline architecture for a given application or workload
- understand the *fundamental* cause of the observed performance (i.e., the characteristics of the *application* that realize the observed performance)
- determine similarity between applications

We present the methodology and show only one use of the results–to predict an optimal mapping of application to architecture. This work is preliminary in that we verify consistency and validate predictions on the only two platforms that were readily available. In the future, we will validate using simulators and a Dell HPC farm that has several different configurations of Intel and AMD processors.

We can currently qualitatively predict a subset of the micro-architecture characteristics required by an application that will optimize its performance including type and relative number of execution units, branch predictor complexity, and relative issue width; we can quantitatively predict cache size and organization. Given an application and a set of computational resources available to the user (e.g., at a computing center), this method can be used to determine which resource/platform will result in optimal performance. This can be extremely useful to users, particularly in high performance computing where many applications have execution times in excess of several days, and saving 10% execution time may mean reducing time-to-result by a day or more. Further development of this technique will enable quantitative prediction of a larger set of micro-architecture

Appendix D: Publications (cont.)



On the Viability of Checkpoint Compression for Extreme Scale Fault Tolerance

Dewan Ibtesham[1], Dorian Arnold[1], Kurt B. Ferreira[2], and Patrick G. Bridges[1]

[1] University of New Mexico, Albuquerque, NM 87131, USA,
`[dewan,darnold,bridges]@cs.unm.edu`
[2] Sandia National Laboratories*, Albuquerque, NM
`kbferre@sandia.gov`

Abstract. The increasing size and complexity of high performance computing (HPC) systems have lead to major concerns over fault frequencies and the mechanisms necessary to tolerate these faults. Previous studies have shown that state-of-the-field checkpoint/restart mechanisms will not scale sufficiently for future generation systems. In this work, we explore the feasibility of checkpoint data compression to reduce checkpoint commit latency and storage overheads. Leveraging a simple model for *checkpoint compression viability*, we conclude that checkpoint data compression should be considered as a part of a scalable checkpoint/restart solution and discuss the types of improvements necessary to make checkpoint data compression more viable.

Keywords: Fault tolerance, Checkpoint compression

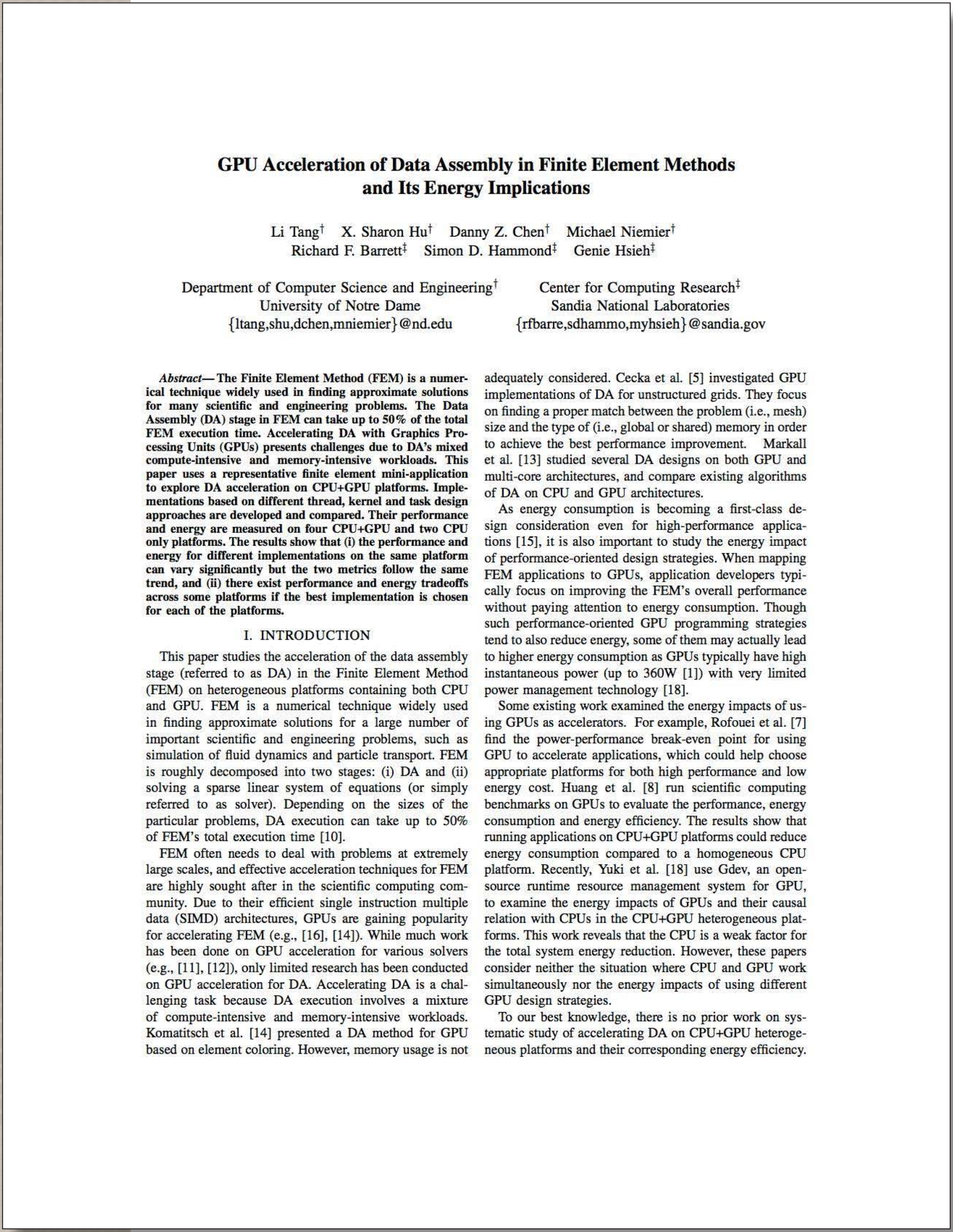
1 Introduction

Over the past few decades, high-performance computing (HPC) systems have increased in size and complexity, and these trends are expected to continue. On the most recent Top 500 list [31], 223 (or 44.6.%) of the 500 entries have greater than 8,192 cores, compared to 15 (or 3.0%) just 5 years ago. Also from this most recent listing, four of the systems are larger than 200K cores; an additional six are larger than 128K cores, and another six are larger than 64K cores. The Lawrence Livermore National Laboratory is scheduled to receive its 1.6 million core system, Sequoia [2], this year. Further, exascale systems are projected to have on the order of tens to hundreds of millions of cores within the current decade [17].

It also is expected that future high-end systems will increase in complexity; for example, heterogeneous systems like CPU/GPU-based systems are expected to become much more prominent. Increased complexity generally suggests that individual components are likely to be more failure prone. Furthermore, mean time between failures (MTBF) is inversely proportional to system size, so increased system sizes also will contribute to extremely low system MTBF. In

* Sandia National Laboratories is a multiprogram 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.

Appendix D: Publications (cont.)



GPU Acceleration of Data Assembly in Finite Element Methods and Its Energy Implications

Li Tang[†] X. Sharon Hu[†] Danny Z. Chen[†] Michael Niemier[†]
Richard F. Barrett[‡] Simon D. Hammond[‡] Genie Hsieh[‡]

Department of Computer Science and Engineering[†]
University of Notre Dame
{ltang,shu,dchen,mniemier}@nd.edu

Center for Computing Research[‡]
Sandia National Laboratories
{rfbarre,sdhammo,myhsieh}@sandia.gov

***Abstract*— The Finite Element Method (FEM) is a numerical technique widely used in finding approximate solutions for many scientific and engineering problems. The Data Assembly (DA) stage in FEM can take up to 50% of the total FEM execution time. Accelerating DA with Graphics Processing Units (GPUs) presents challenges due to DA's mixed compute-intensive and memory-intensive workloads. This paper uses a representative finite element mini-application to explore DA acceleration on CPU+GPU platforms. Implementations based on different thread, kernel and task design approaches are developed and compared. Their performance and energy are measured on four CPU+GPU and two CPU only platforms. The results show that (i) the performance and energy for different implementations on the same platform can vary significantly but the two metrics follow the same trend, and (ii) there exist performance and energy tradeoffs across some platforms if the best implementation is chosen for each of the platforms.**

I. INTRODUCTION

This paper studies the acceleration of the data assembly stage (referred to as DA) in the Finite Element Method (FEM) on heterogeneous platforms containing both CPU and GPU. FEM is a numerical technique widely used in finding approximate solutions for a large number of important scientific and engineering problems, such as simulation of fluid dynamics and particle transport. FEM is roughly decomposed into two stages: (i) DA and (ii) solving a sparse linear system of equations (or simply referred to as solver). Depending on the sizes of the particular problems, DA execution can take up to 50% of FEM's total execution time [10].

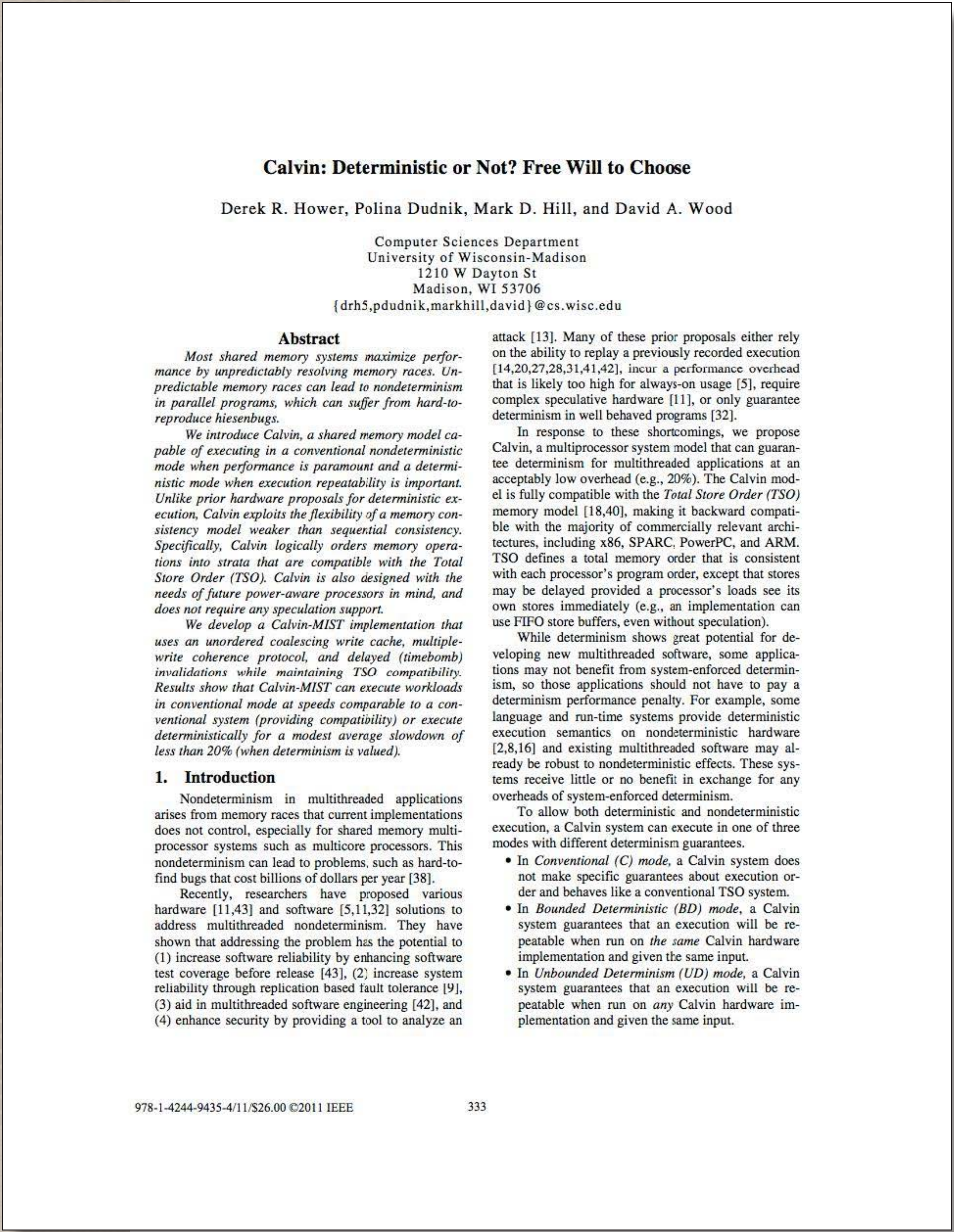
FEM often needs to deal with problems at extremely large scales, and effective acceleration techniques for FEM are highly sought after in the scientific computing community. Due to their efficient single instruction multiple data (SIMD) architectures, GPUs are gaining popularity for accelerating FEM (e.g., [16], [14]). While much work has been done on GPU acceleration for various solvers (e.g., [11], [12]), only limited research has been conducted on GPU acceleration for DA. Accelerating DA is a challenging task because DA execution involves a mixture of compute-intensive and memory-intensive workloads. Komatitsch et al. [14] presented a DA method for GPU based on element coloring. However, memory usage is not adequately considered. Cecka et al. [5] investigated GPU implementations of DA for unstructured grids. They focus on finding a proper match between the problem (i.e., mesh) size and the type of (i.e., global or shared) memory in order to achieve the best performance improvement. Markall et al. [13] studied several DA designs on both GPU and multi-core architectures, and compare existing algorithms of DA on CPU and GPU architectures.

As energy consumption is becoming a first-class design consideration even for high-performance applications [15], it is also important to study the energy impact of performance-oriented design strategies. When mapping FEM applications to GPUs, application developers typically focus on improving the FEM's overall performance without paying attention to energy consumption. Though such performance-oriented GPU programming strategies tend to also reduce energy, some of them may actually lead to higher energy consumption as GPUs typically have high instantaneous power (up to 360W [1]) with very limited power management technology [18].

Some existing work examined the energy impacts of using GPUs as accelerators. For example, Rofouei et al. [7] find the power-performance break-even point for using GPU to accelerate applications, which could help choose appropriate platforms for both high performance and low energy cost. Huang et al. [8] run scientific computing benchmarks on GPUs to evaluate the performance, energy consumption and energy efficiency. The results show that running applications on CPU+GPU platforms could reduce energy consumption compared to a homogeneous CPU platform. Recently, Yuki et al. [18] use Gdev, an open-source runtime resource management system for GPU, to examine the energy impacts of GPUs and their causal relation with CPUs in the CPU+GPU heterogeneous platforms. This work reveals that the CPU is a weak factor for the total system energy reduction. However, these papers consider neither the situation where CPU and GPU work simultaneously nor the energy impacts of using different GPU design strategies.

To our best knowledge, there is no prior work on systematic study of accelerating DA on CPU+GPU heterogeneous platforms and their corresponding energy efficiency.

Appendix D: Publications (cont.)



Calvin: Deterministic or Not? Free Will to Choose

Derek R. Hower, Polina Dudnik, Mark D. Hill, and David A. Wood

Computer Sciences Department
University of Wisconsin-Madison
1210 W Dayton St
Madison, WI 53706
{drh5,pdudnik,markhill,david}@cs.wisc.edu

Abstract

Most shared memory systems maximize performance by unpredictably resolving memory races. Unpredictable memory races can lead to nondeterminism in parallel programs, which can suffer from hard-to-reproduce hiesenbugs.

We introduce Calvin, a shared memory model capable of executing in a conventional nondeterministic mode when performance is paramount and a deterministic mode when execution repeatability is important. Unlike prior hardware proposals for deterministic execution, Calvin exploits the flexibility of a memory consistency model weaker than sequential consistency. Specifically, Calvin logically orders memory operations into strata that are compatible with the Total Store Order (TSO). Calvin is also designed with the needs of future power-aware processors in mind, and does not require any speculation support.

We develop a Calvin-MIST implementation that uses an unordered coalescing write cache, multiple-write coherence protocol, and delayed (timebomb) invalidations while maintaining TSO compatibility. Results show that Calvin-MIST can execute workloads in conventional mode at speeds comparable to a conventional system (providing compatibility) or execute deterministically for a modest average slowdown of less than 20% (when determinism is valued).

1. Introduction

Nondeterminism in multithreaded applications arises from memory races that current implementations does not control, especially for shared memory multiprocessor systems such as multicore processors. This nondeterminism can lead to problems, such as hard-to-find bugs that cost billions of dollars per year [38].

Recently, researchers have proposed various hardware [11,43] and software [5,11,32] solutions to address multithreaded nondeterminism. They have shown that addressing the problem has the potential to (1) increase software reliability by enhancing software test coverage before release [43], (2) increase system reliability through replication based fault tolerance [9], (3) aid in multithreaded software engineering [42], and (4) enhance security by providing a tool to analyze an attack [13]. Many of these prior proposals either rely on the ability to replay a previously recorded execution [14,20,27,28,31,41,42], incur a performance overhead that is likely too high for always-on usage [5], require complex speculative hardware [11], or only guarantee determinism in well behaved programs [32].

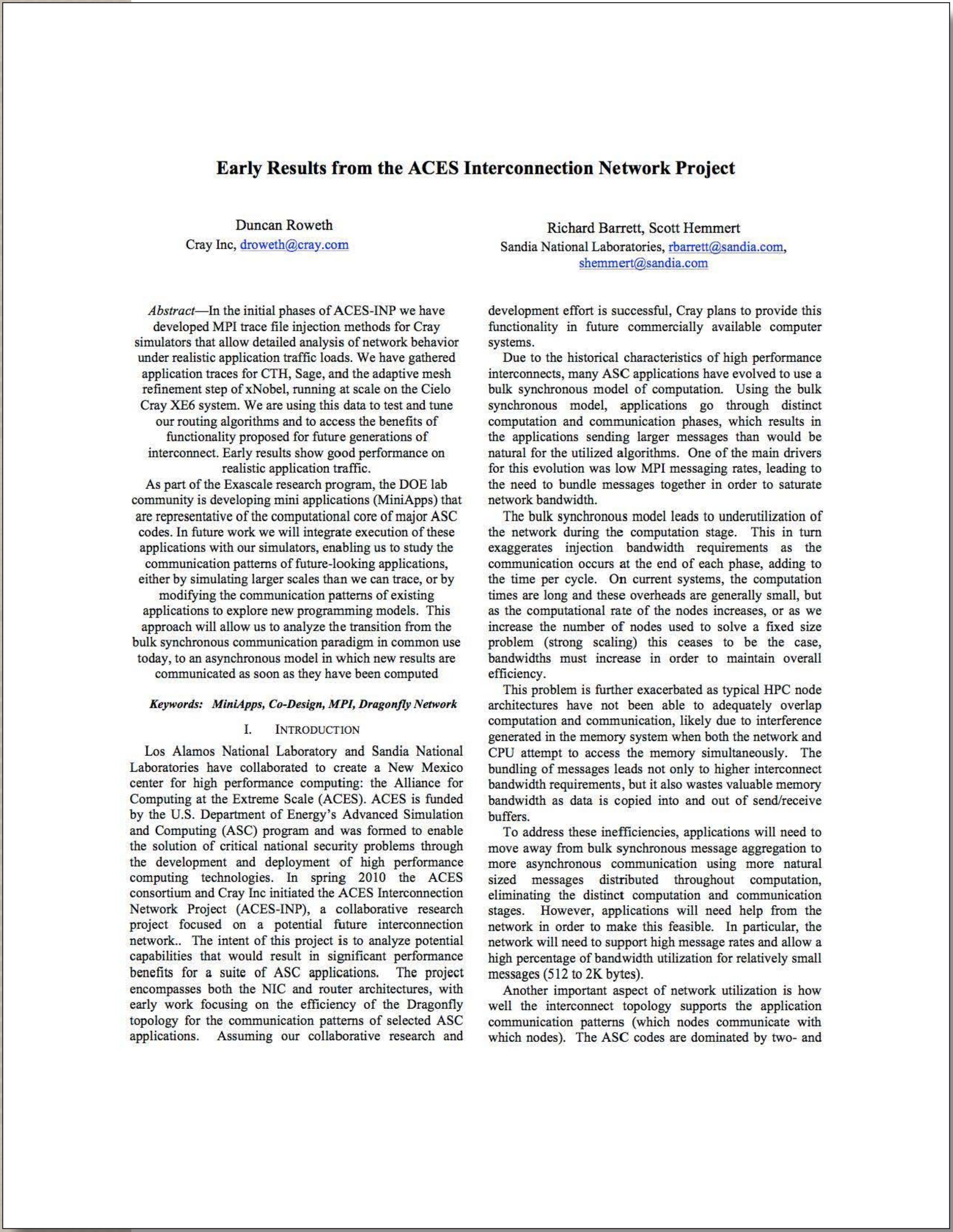
In response to these shortcomings, we propose Calvin, a multiprocessor system model that can guarantee determinism for multithreaded applications at an acceptably low overhead (e.g., 20%). The Calvin model is fully compatible with the *Total Store Order (TSO)* memory model [18,40], making it backward compatible with the majority of commercially relevant architectures, including x86, SPARC, PowerPC, and ARM. TSO defines a total memory order that is consistent with each processor's program order, except that stores may be delayed provided a processor's loads see its own stores immediately (e.g., an implementation can use FIFO store buffers, even without speculation).

While determinism shows great potential for developing new multithreaded software, some applications may not benefit from system-enforced determinism, so those applications should not have to pay a determinism performance penalty. For example, some language and run-time systems provide deterministic execution semantics on nondeterministic hardware [2,8,16] and existing multithreaded software may already be robust to nondeterministic effects. These systems receive little or no benefit in exchange for any overheads of system-enforced determinism.

To allow both deterministic and nondeterministic execution, a Calvin system can execute in one of three modes with different determinism guarantees.

- In *Conventional (C) mode,* a Calvin system does not make specific guarantees about execution order and behaves like a conventional TSO system.
- In *Bounded Deterministic (BD) mode*, a Calvin system guarantees that an execution will be repeatable when run on *the same* Calvin hardware implementation and given the same input.
- In *Unbounded Determinism (UD) mode,* a Calvin system guarantees that an execution will be repeatable when run on *any* Calvin hardware implementation and given the same input.

Appendix D: Publications (cont.)



Early Results from the ACES Interconnection Network Project

Duncan Roweth
Cray Inc, droweth@cray.com

Richard Barrett, Scott Hemmert
Sandia National Laboratories, rbarrett@sandia.com, shemmert@sandia.com

Abstract—In the initial phases of ACES-INP we have developed MPI trace file injection methods for Cray simulators that allow detailed analysis of network behavior under realistic application traffic loads. We have gathered application traces for CTH, Sage, and the adaptive mesh refinement step of xNobel, running at scale on the Cielo Cray XE6 system. We are using this data to test and tune our routing algorithms and to access the benefits of functionality proposed for future generations of interconnect. Early results show good performance on realistic application traffic.

As part of the Exascale research program, the DOE lab community is developing mini applications (MiniApps) that are representative of the computational core of major ASC codes. In future work we will integrate execution of these applications with our simulators, enabling us to study the communication patterns of future-looking applications, either by simulating larger scales than we can trace, or by modifying the communication patterns of existing applications to explore new programming models. This approach will allow us to analyze the transition from the bulk synchronous communication paradigm in common use today, to an asynchronous model in which new results are communicated as soon as they have been computed

Keywords: MiniApps, Co-Design, MPI, Dragonfly Network

I. Introduction

Los Alamos National Laboratory and Sandia National Laboratories have collaborated to create a New Mexico center for high performance computing: the Alliance for Computing at the Extreme Scale (ACES). ACES is funded by the U.S. Department of Energy's Advanced Simulation and Computing (ASC) program and was formed to enable the solution of critical national security problems through the development and deployment of high performance computing technologies. In spring 2010 the ACES consortium and Cray Inc initiated the ACES Interconnection Network Project (ACES-INP), a collaborative research project focused on a potential future interconnection network.. The intent of this project is to analyze potential capabilities that would result in significant performance benefits for a suite of ASC applications. The project encompasses both the NIC and router architectures, with early work focusing on the efficiency of the Dragonfly topology for the communication patterns of selected ASC applications. Assuming our collaborative research and development effort is successful, Cray plans to provide this functionality in future commercially available computer systems.

Due to the historical characteristics of high performance interconnects, many ASC applications have evolved to use a bulk synchronous model of computation. Using the bulk synchronous model, applications go through distinct computation and communication phases, which results in the applications sending larger messages than would be natural for the utilized algorithms. One of the main drivers for this evolution was low MPI messaging rates, leading to the need to bundle messages together in order to saturate network bandwidth.

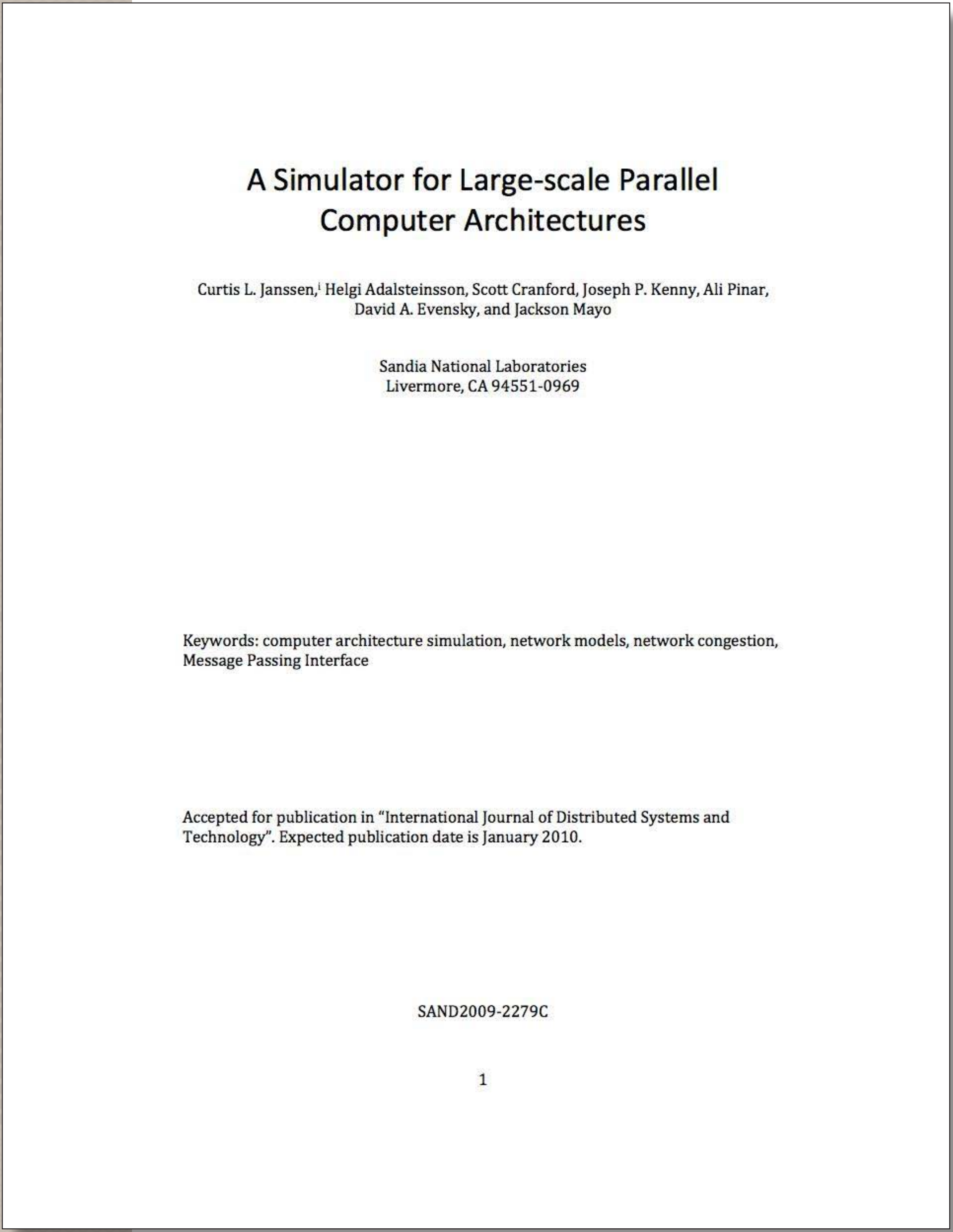
The bulk synchronous model leads to underutilization of the network during the computation stage. This in turn exaggerates injection bandwidth requirements as the communication occurs at the end of each phase, adding to the time per cycle. On current systems, the computation times are long and these overheads are generally small, but as the computational rate of the nodes increases, or as we increase the number of nodes used to solve a fixed size problem (strong scaling) this ceases to be the case, bandwidths must increase in order to maintain overall efficiency.

This problem is further exacerbated as typical HPC node architectures have not been able to adequately overlap computation and communication, likely due to interference generated in the memory system when both the network and CPU attempt to access the memory simultaneously. The bundling of messages leads not only to higher interconnect bandwidth requirements, but it also wastes valuable memory bandwidth as data is copied into and out of send/receive buffers.

To address these inefficiencies, applications will need to move away from bulk synchronous message aggregation to more asynchronous communication using more natural sized messages distributed throughout computation, eliminating the distinct computation and communication stages. However, applications will need help from the network in order to make this feasible. In particular, the network will need to support high message rates and allow a high percentage of bandwidth utilization for relatively small messages (512 to 2K bytes).

Another important aspect of network utilization is how well the interconnect topology supports the application communication patterns (which nodes communicate with which nodes). The ASC codes are dominated by two- and

Appendix D: Publications (cont.)



A Simulator for Large-scale Parallel Computer Architectures

Curtis L. Janssen,[i] Helgi Adalsteinsson, Scott Cranford, Joseph P. Kenny, Ali Pinar, David A. Evensky, and Jackson Mayo

Sandia National Laboratories
Livermore, CA 94551-0969

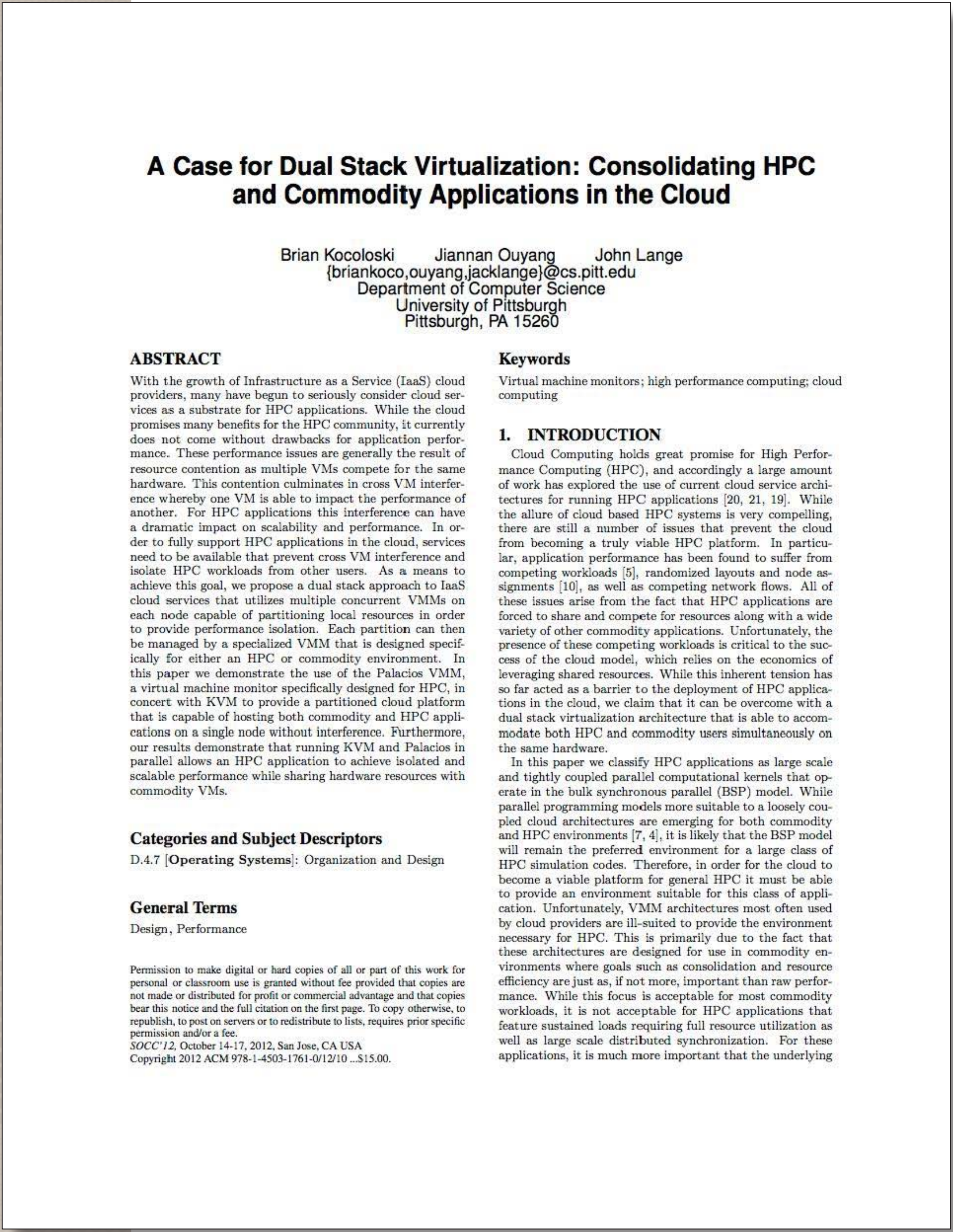
Keywords: computer architecture simulation, network models, network congestion, Message Passing Interface

Accepted for publication in "International Journal of Distributed Systems and Technology". Expected publication date is January 2010.

SAND2009-2279C

1

Appendix D: Publications (cont.)



A Case for Dual Stack Virtualization: Consolidating HPC and Commodity Applications in the Cloud

Brian Kocoloski Jiannan Ouyang John Lange
{briankoco,ouyang,jacklange}@cs.pitt.edu
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260

ABSTRACT

With the growth of Infrastructure as a Service (IaaS) cloud providers, many have begun to seriously consider cloud services as a substrate for HPC applications. While the cloud promises many benefits for the HPC community, it currently does not come without drawbacks for application performance. These performance issues are generally the result of resource contention as multiple VMs compete for the same hardware. This contention culminates in cross VM interference whereby one VM is able to impact the performance of another. For HPC applications this interference can have a dramatic impact on scalability and performance. In order to fully support HPC applications in the cloud, services need to be available that prevent cross VM interference and isolate HPC workloads from other users. As a means to achieve this goal, we propose a dual stack approach to IaaS cloud services that utilizes multiple concurrent VMMs on each node capable of partitioning local resources in order to provide performance isolation. Each partition can then be managed by a specialized VMM that is designed specifically for either an HPC or commodity environment. In this paper we demonstrate the use of the Palacios VMM, a virtual machine monitor specifically designed for HPC, in concert with KVM to provide a partitioned cloud platform that is capable of hosting both commodity and HPC applications on a single node without interference. Furthermore, our results demonstrate that running KVM and Palacios in parallel allows an HPC application to achieve isolated and scalable performance while sharing hardware resources with commodity VMs.

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organization and Design

General Terms

Design, Performance

Keywords

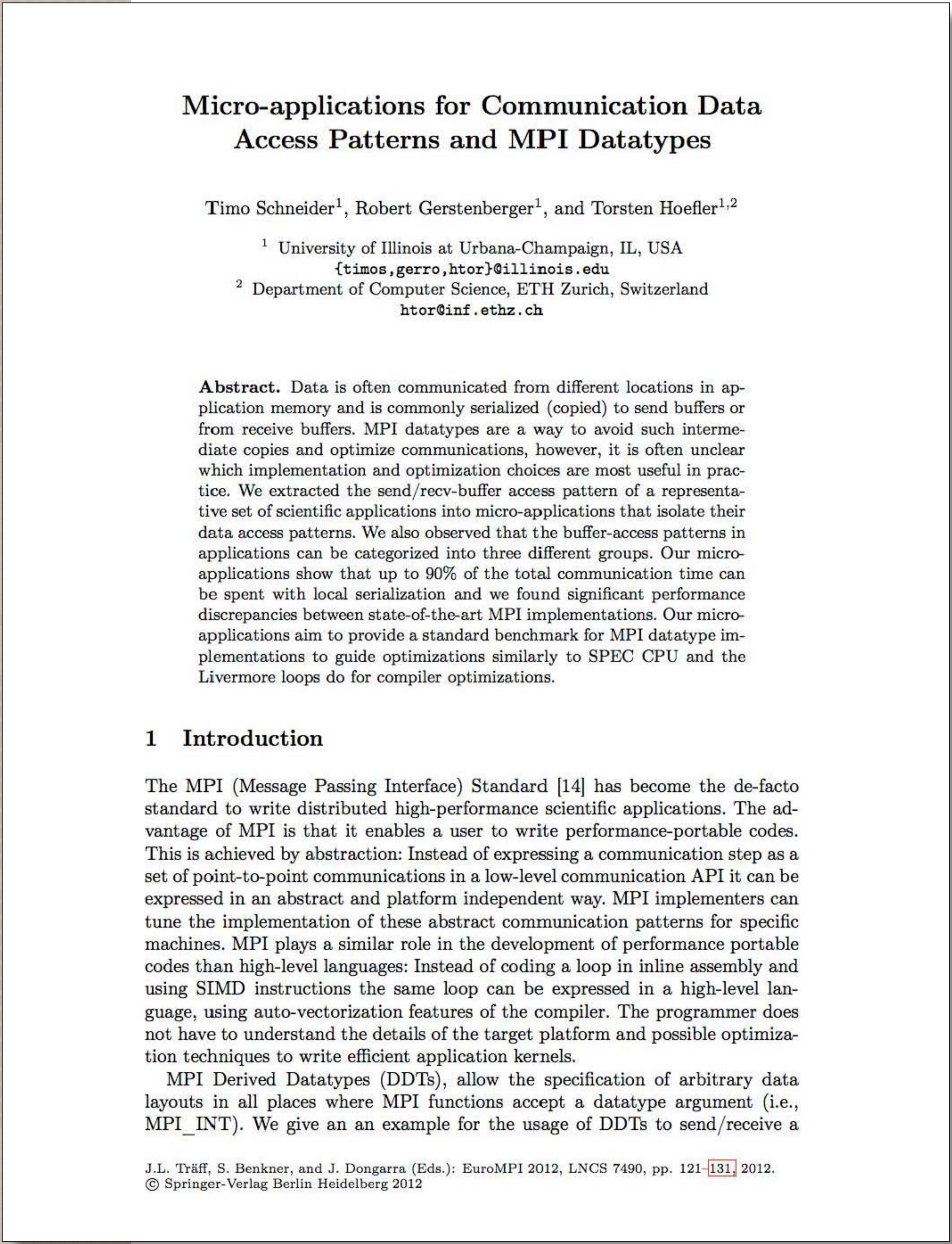
Virtual machine monitors; high performance computing; cloud computing

1. INTRODUCTION

Cloud Computing holds great promise for High Performance Computing (HPC), and accordingly a large amount of work has explored the use of current cloud service architectures for running HPC applications [20, 21, 19]. While the allure of cloud based HPC systems is very compelling, there are still a number of issues that prevent the cloud from becoming a truly viable HPC platform. In particular, application performance has been found to suffer from competing workloads [5], randomized layouts and node assignments [10], as well as competing network flows. All of these issues arise from the fact that HPC applications are forced to share and compete for resources along with a wide variety of other commodity applications. Unfortunately, the presence of these competing workloads is critical to the success of the cloud model, which relies on the economics of leveraging shared resources. While this inherent tension has so far acted as a barrier to the deployment of HPC applications in the cloud, we claim that it can be overcome with a dual stack virtualization architecture that is able to accommodate both HPC and commodity users simultaneously on the same hardware.

In this paper we classify HPC applications as large scale and tightly coupled parallel computational kernels that operate in the bulk synchronous parallel (BSP) model. While parallel programming models more suitable to a loosely coupled cloud architectures are emerging for both commodity and HPC environments [7, 4], it is likely that the BSP model will remain the preferred environment for a large class of HPC simulation codes. Therefore, in order for the cloud to become a viable platform for general HPC it must be able to provide an environment suitable for this class of application. Unfortunately, VMM architectures most often used by cloud providers are ill-suited to provide the environment necessary for HPC. This is primarily due to the fact that these architectures are designed for use in commodity environments where goals such as consolidation and resource efficiency are just as, if not more, important than raw performance. While this focus is acceptable for most commodity workloads, it is not acceptable for HPC applications that feature sustained loads requiring full resource utilization as well as large scale distributed synchronization. For these applications, it is much more important that the underlying

Appendix D: Publications (cont.)



Micro-applications for Communication Data Access Patterns and MPI Datatypes

Timo Schneider[1], Robert Gerstenberger[1], and Torsten Hoefler[1,2]

[1] University of Illinois at Urbana-Champaign, IL, USA
{timos,gerro,htor}@illinois.edu
[2] Department of Computer Science, ETH Zurich, Switzerland
htor@inf.ethz.ch

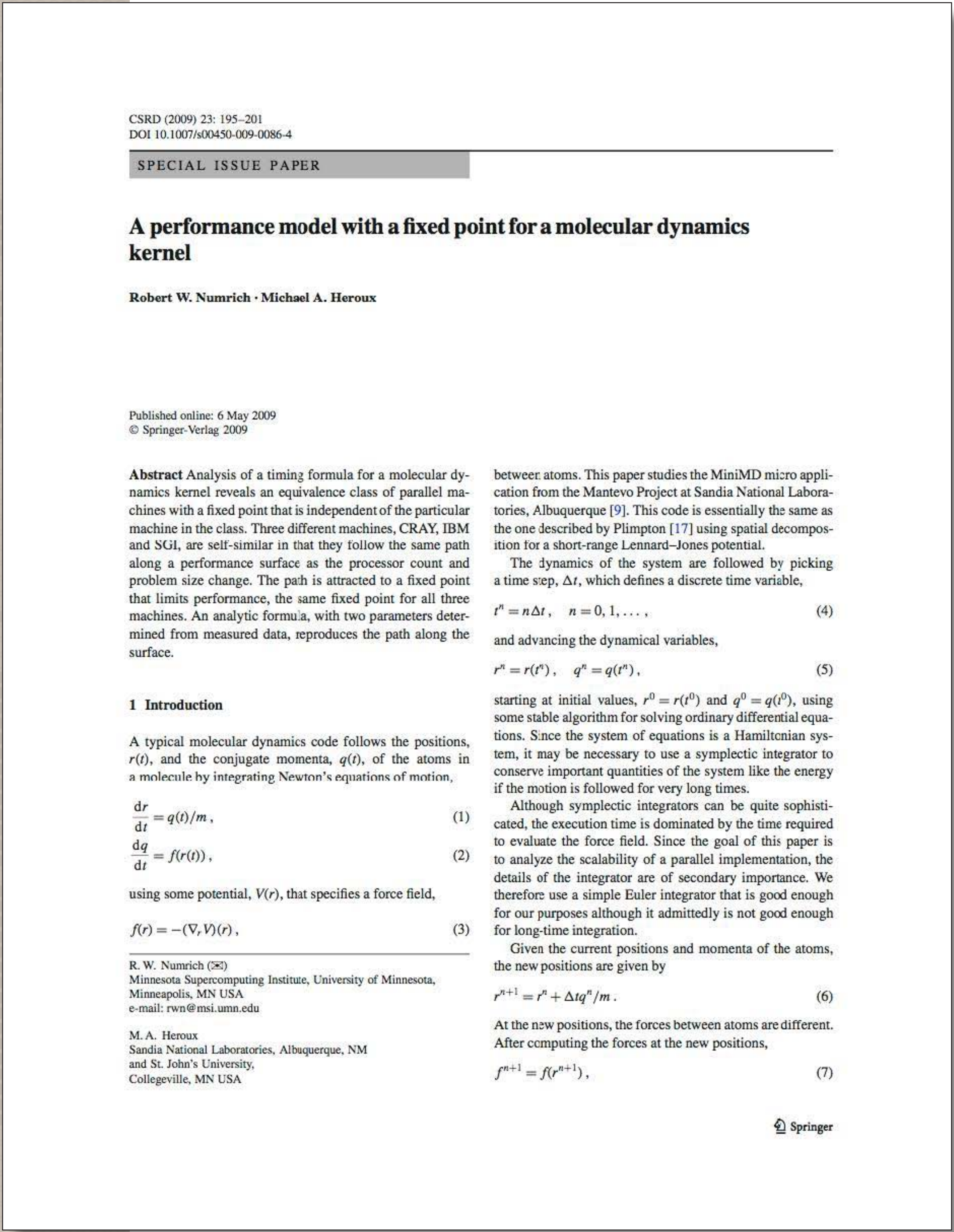
Abstract. Data is often communicated from different locations in application memory and is commonly serialized (copied) to send buffers or from receive buffers. MPI datatypes are a way to avoid such intermediate copies and optimize communications, however, it is often unclear which implementation and optimization choices are most useful in practice. We extracted the send/recv-buffer access pattern of a representative set of scientific applications into micro-applications that isolate their data access patterns. We also observed that the buffer-access patterns in applications can be categorized into three different groups. Our micro-applications show that up to 90% of the total communication time can be spent with local serialization and we found significant performance discrepancies between state-of-the-art MPI implementations. Our micro-applications aim to provide a standard benchmark for MPI datatype implementations to guide optimizations similarly to SPEC CPU and the Livermore loops do for compiler optimizations.

1 Introduction

The MPI (Message Passing Interface) Standard [14] has become the de-facto standard to write distributed high-performance scientific applications. The advantage of MPI is that it enables a user to write performance-portable codes. This is achieved by abstraction: Instead of expressing a communication step as a set of point-to-point communications in a low-level communication API it can be expressed in an abstract and platform independent way. MPI implementers can tune the implementation of these abstract communication patterns for specific machines. MPI plays a similar role in the development of performance portable codes than high-level languages: Instead of coding a loop in inline assembly and using SIMD instructions the same loop can be expressed in a high-level language, using auto-vectorization features of the compiler. The programmer does not have to understand the details of the target platform and possible optimization techniques to write efficient application kernels.

MPI Derived Datatypes (DDTs), allow the specification of arbitrary data layouts in all places where MPI functions accept a datatype argument (i.e., MPI_INT). We give an an example for the usage of DDTs to send/receive a

Appendix D: Publications (cont.)



CSRD (2009) 23: 195–201
DOI 10.1007/s00450-009-0086-4

SPECIAL ISSUE PAPER

A performance model with a fixed point for a molecular dynamics kernel

Robert W. Numrich · Michael A. Heroux

Published online: 6 May 2009
© Springer-Verlag 2009

Abstract Analysis of a timing formula for a molecular dynamics kernel reveals an equivalence class of parallel machines with a fixed point that is independent of the particular machine in the class. Three different machines, CRAY, IBM and SGI, are self-similar in that they follow the same path along a performance surface as the processor count and problem size change. The path is attracted to a fixed point that limits performance, the same fixed point for all three machines. An analytic formula, with two parameters determined from measured data, reproduces the path along the surface.

1 Introduction

A typical molecular dynamics code follows the positions, $r(t)$, and the conjugate momenta, $q(t)$, of the atoms in a molecule by integrating Newton's equations of motion,

$$\frac{\mathrm{d}r}{\mathrm{d}t} = q(t)/m\,, \tag{1}$$

$$\frac{\mathrm{d}q}{\mathrm{d}t} = f(r(t))\,, \tag{2}$$

using some potential, $V(r)$, that specifies a force field,

$$f(r) = -(\nabla_r V)(r)\,, \tag{3}$$

between atoms. This paper studies the MiniMD micro application from the Mantevo Project at Sandia National Laboratories, Albuquerque [9]. This code is essentially the same as the one described by Plimpton [17] using spatial decomposition for a short-range Lennard–Jones potential.

The dynamics of the system are followed by picking a time step, Δt, which defines a discrete time variable,

$$t^n = n\Delta t\,, \quad n = 0, 1, \ldots\,, \tag{4}$$

and advancing the dynamical variables,

$$r^n = r(t^n)\,, \quad q^n = q(t^n)\,, \tag{5}$$

starting at initial values, $r^0 = r(t^0)$ and $q^0 = q(t^0)$, using some stable algorithm for solving ordinary differential equations. Since the system of equations is a Hamiltonian system, it may be necessary to use a symplectic integrator to conserve important quantities of the system like the energy if the motion is followed for very long times.

Although symplectic integrators can be quite sophisticated, the execution time is dominated by the time required to evaluate the force field. Since the goal of this paper is to analyze the scalability of a parallel implementation, the details of the integrator are of secondary importance. We therefore use a simple Euler integrator that is good enough for our purposes although it admittedly is not good enough for long-time integration.

Given the current positions and momenta of the atoms, the new positions are given by

$$r^{n+1} = r^n + \Delta t q^n/m\,. \tag{6}$$

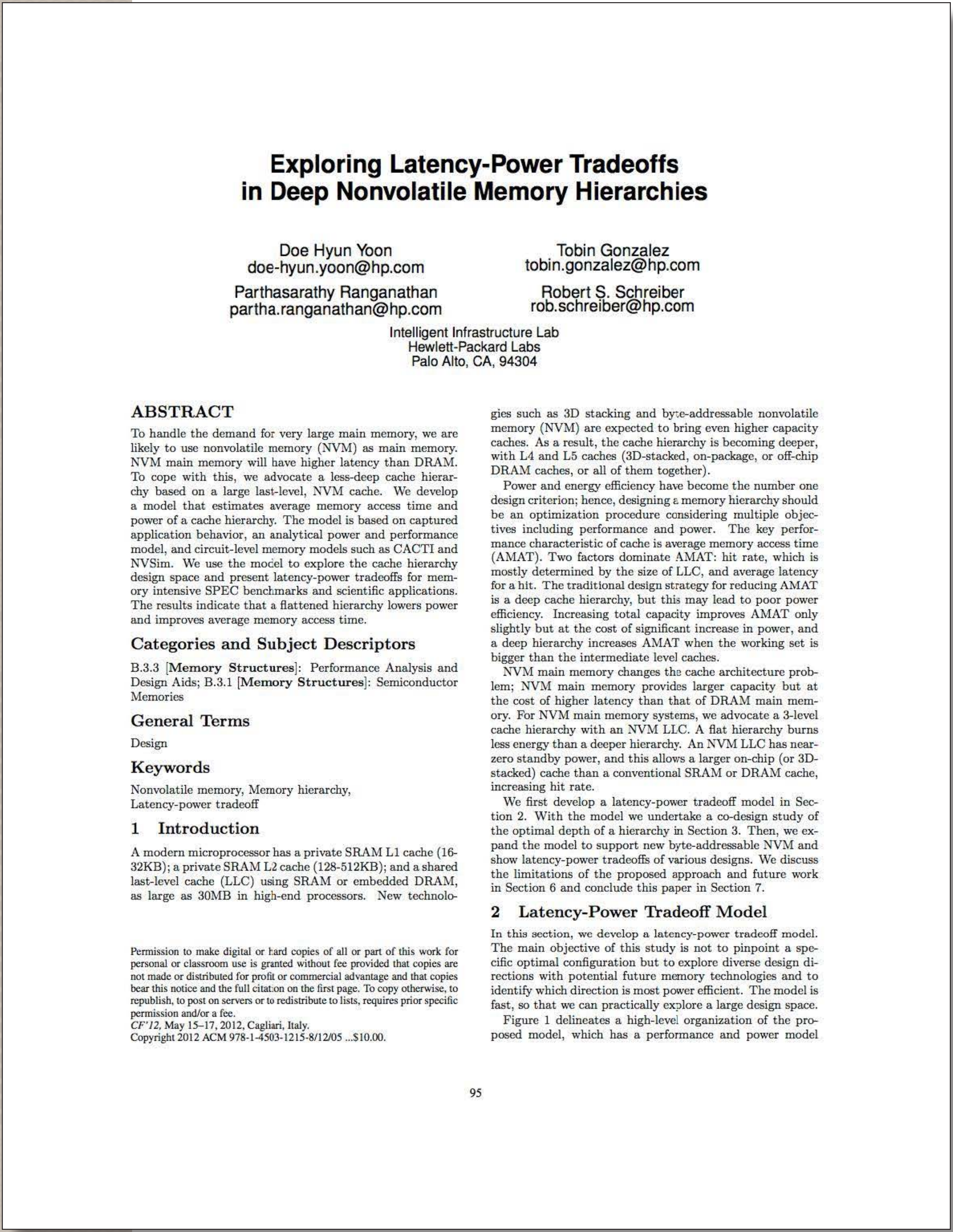
At the new positions, the forces between atoms are different. After computing the forces at the new positions,

$$f^{n+1} = f(r^{n+1})\,, \tag{7}$$

R. W. Numrich (✉)
Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, MN USA
e-mail: rwn@msi.umn.edu

M. A. Heroux
Sandia National Laboratories, Albuquerque, NM and St. John's University, Collegeville, MN USA

Appendix D: Publications (cont.)



Exploring Latency-Power Tradeoffs in Deep Nonvolatile Memory Hierarchies

Doe Hyun Yoon
doe-hyun.yoon@hp.com

Tobin Gonzalez
tobin.gonzalez@hp.com

Parthasarathy Ranganathan
partha.ranganathan@hp.com

Robert S. Schreiber
rob.schreiber@hp.com

Intelligent Infrastructure Lab
Hewlett-Packard Labs
Palo Alto, CA, 94304

ABSTRACT

To handle the demand for very large main memory, we are likely to use nonvolatile memory (NVM) as main memory. NVM main memory will have higher latency than DRAM. To cope with this, we advocate a less-deep cache hierarchy based on a large last-level, NVM cache. We develop a model that estimates average memory access time and power of a cache hierarchy. The model is based on captured application behavior, an analytical power and performance model, and circuit-level memory models such as CACTI and NVSim. We use the model to explore the cache hierarchy design space and present latency-power tradeoffs for memory intensive SPEC benchmarks and scientific applications. The results indicate that a flattened hierarchy lowers power and improves average memory access time.

Categories and Subject Descriptors

B.3.3 [**Memory Structures**]: Performance Analysis and Design Aids; B.3.1 [**Memory Structures**]: Semiconductor Memories

General Terms

Design

Keywords

Nonvolatile memory, Memory hierarchy, Latency-power tradeoff

1 Introduction

A modern microprocessor has a private SRAM L1 cache (16-32KB); a private SRAM L2 cache (128-512KB); and a shared last-level cache (LLC) using SRAM or embedded DRAM, as large as 30MB in high-end processors. New technologies such as 3D stacking and byte-addressable nonvolatile memory (NVM) are expected to bring even higher capacity caches. As a result, the cache hierarchy is becoming deeper, with L4 and L5 caches (3D-stacked, on-package, or off-chip DRAM caches, or all of them together).

Power and energy efficiency have become the number one design criterion; hence, designing a memory hierarchy should be an optimization procedure considering multiple objectives including performance and power. The key performance characteristic of cache is average memory access time (AMAT). Two factors dominate AMAT: hit rate, which is mostly determined by the size of LLC, and average latency for a hit. The traditional design strategy for reducing AMAT is a deep cache hierarchy, but this may lead to poor power efficiency. Increasing total capacity improves AMAT only slightly but at the cost of significant increase in power, and a deep hierarchy increases AMAT when the working set is bigger than the intermediate level caches.

NVM main memory changes the cache architecture problem; NVM main memory provides larger capacity but at the cost of higher latency than that of DRAM main memory. For NVM main memory systems, we advocate a 3-level cache hierarchy with an NVM LLC. A flat hierarchy burns less energy than a deeper hierarchy. An NVM LLC has near-zero standby power, and this allows a larger on-chip (or 3D-stacked) cache than a conventional SRAM or DRAM cache, increasing hit rate.

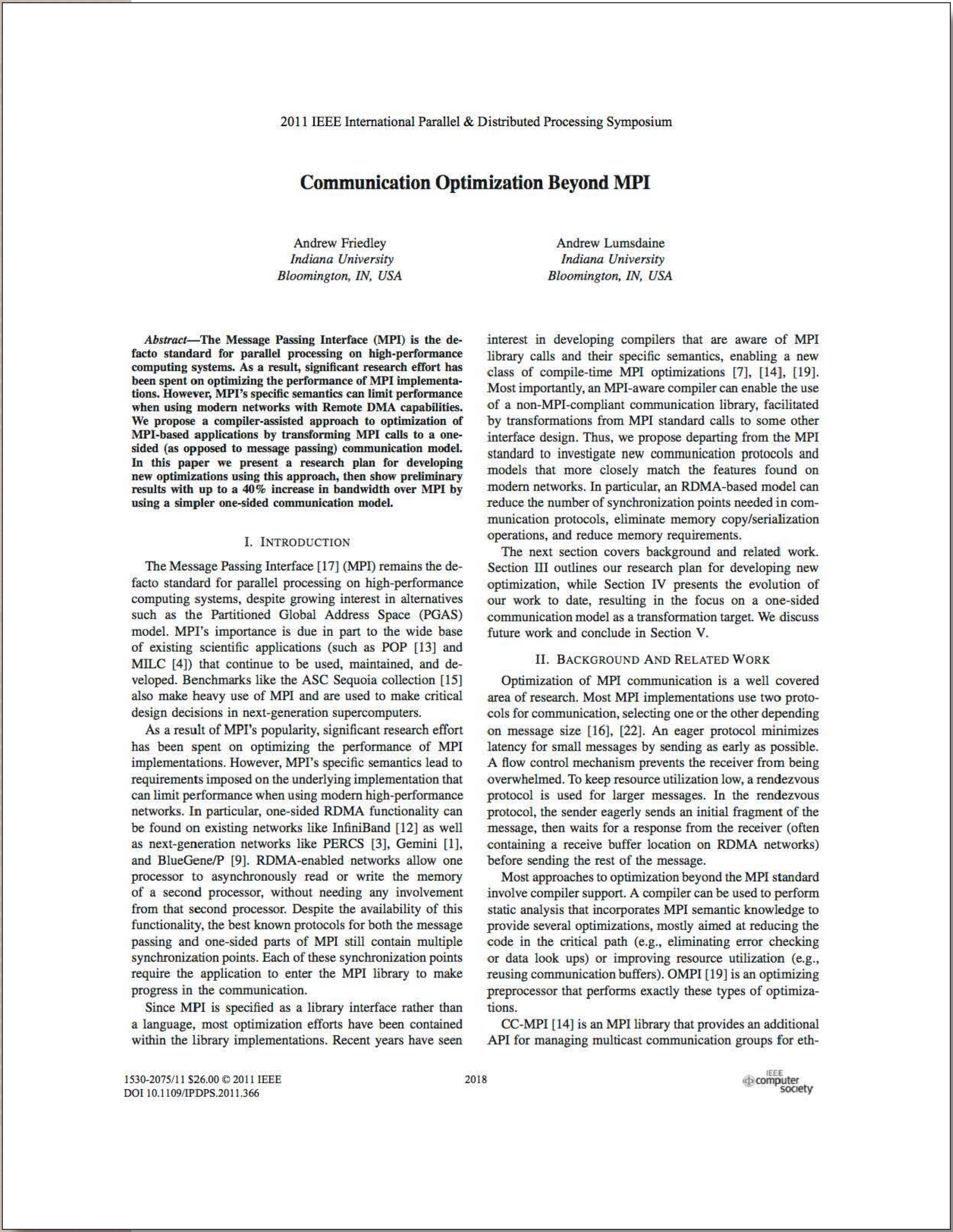
We first develop a latency-power tradeoff model in Section 2. With the model we undertake a co-design study of the optimal depth of a hierarchy in Section 3. Then, we expand the model to support new byte-addressable NVM and show latency-power tradeoffs of various designs. We discuss the limitations of the proposed approach and future work in Section 6 and conclude this paper in Section 7.

2 Latency-Power Tradeoff Model

In this section, we develop a latency-power tradeoff model. The main objective of this study is not to pinpoint a specific optimal configuration but to explore diverse design directions with potential future memory technologies and to identify which direction is most power efficient. The model is fast, so that we can practically explore a large design space.

Figure 1 delineates a high-level organization of the proposed model, which has a performance and power model

Appendix D: Publications (cont.)



2011 IEEE International Parallel & Distributed Processing Symposium

Communication Optimization Beyond MPI

Andrew Friedley
Indiana University
Bloomington, IN, USA

Andrew Lumsdaine
Indiana University
Bloomington, IN, USA

***Abstract*—The Message Passing Interface (MPI) is the de-facto standard for parallel processing on high-performance computing systems. As a result, significant research effort has been spent on optimizing the performance of MPI implementations. However, MPI's specific semantics can limit performance when using modern networks with Remote DMA capabilities. We propose a compiler-assisted approach to optimization of MPI-based applications by transforming MPI calls to a one-sided (as opposed to message passing) communication model. In this paper we present a research plan for developing new optimizations using this approach, then show preliminary results with up to a 40% increase in bandwidth over MPI by using a simpler one-sided communication model.**

I. Introduction

The Message Passing Interface [17] (MPI) remains the de-facto standard for parallel processing on high-performance computing systems, despite growing interest in alternatives such as the Partitioned Global Address Space (PGAS) model. MPI's importance is due in part to the wide base of existing scientific applications (such as POP [13] and MILC [4]) that continue to be used, maintained, and developed. Benchmarks like the ASC Sequoia collection [15] also make heavy use of MPI and are used to make critical design decisions in next-generation supercomputers.

As a result of MPI's popularity, significant research effort has been spent on optimizing the performance of MPI implementations. However, MPI's specific semantics lead to requirements imposed on the underlying implementation that can limit performance when using modern high-performance networks. In particular, one-sided RDMA functionality can be found on existing networks like InfiniBand [12] as well as next-generation networks like PERCS [3], Gemini [1], and BlueGene/P [9]. RDMA-enabled networks allow one processor to asynchronously read or write the memory of a second processor, without needing any involvement from that second processor. Despite the availability of this functionality, the best known protocols for both the message passing and one-sided parts of MPI still contain multiple synchronization points. Each of these synchronization points require the application to enter the MPI library to make progress in the communication.

Since MPI is specified as a library interface rather than a language, most optimization efforts have been contained within the library implementations. Recent years have seen interest in developing compilers that are aware of MPI library calls and their specific semantics, enabling a new class of compile-time MPI optimizations [7], [14], [19]. Most importantly, an MPI-aware compiler can enable the use of a non-MPI-compliant communication library, facilitated by transformations from MPI standard calls to some other interface design. Thus, we propose departing from the MPI standard to investigate new communication protocols and models that more closely match the features found on modern networks. In particular, an RDMA-based model can reduce the number of synchronization points needed in communication protocols, eliminate memory copy/serialization operations, and reduce memory requirements.

The next section covers background and related work. Section III outlines our research plan for developing new optimization, while Section IV presents the evolution of our work to date, resulting in the focus on a one-sided communication model as a transformation target. We discuss future work and conclude in Section V.

II. Background And Related Work

Optimization of MPI communication is a well covered area of research. Most MPI implementations use two protocols for communication, selecting one or the other depending on message size [16], [22]. An eager protocol minimizes latency for small messages by sending as early as possible. A flow control mechanism prevents the receiver from being overwhelmed. To keep resource utilization low, a rendezvous protocol is used for larger messages. In the rendezvous protocol, the sender eagerly sends an initial fragment of the message, then waits for a response from the receiver (often containing a receive buffer location on RDMA networks) before sending the rest of the message.

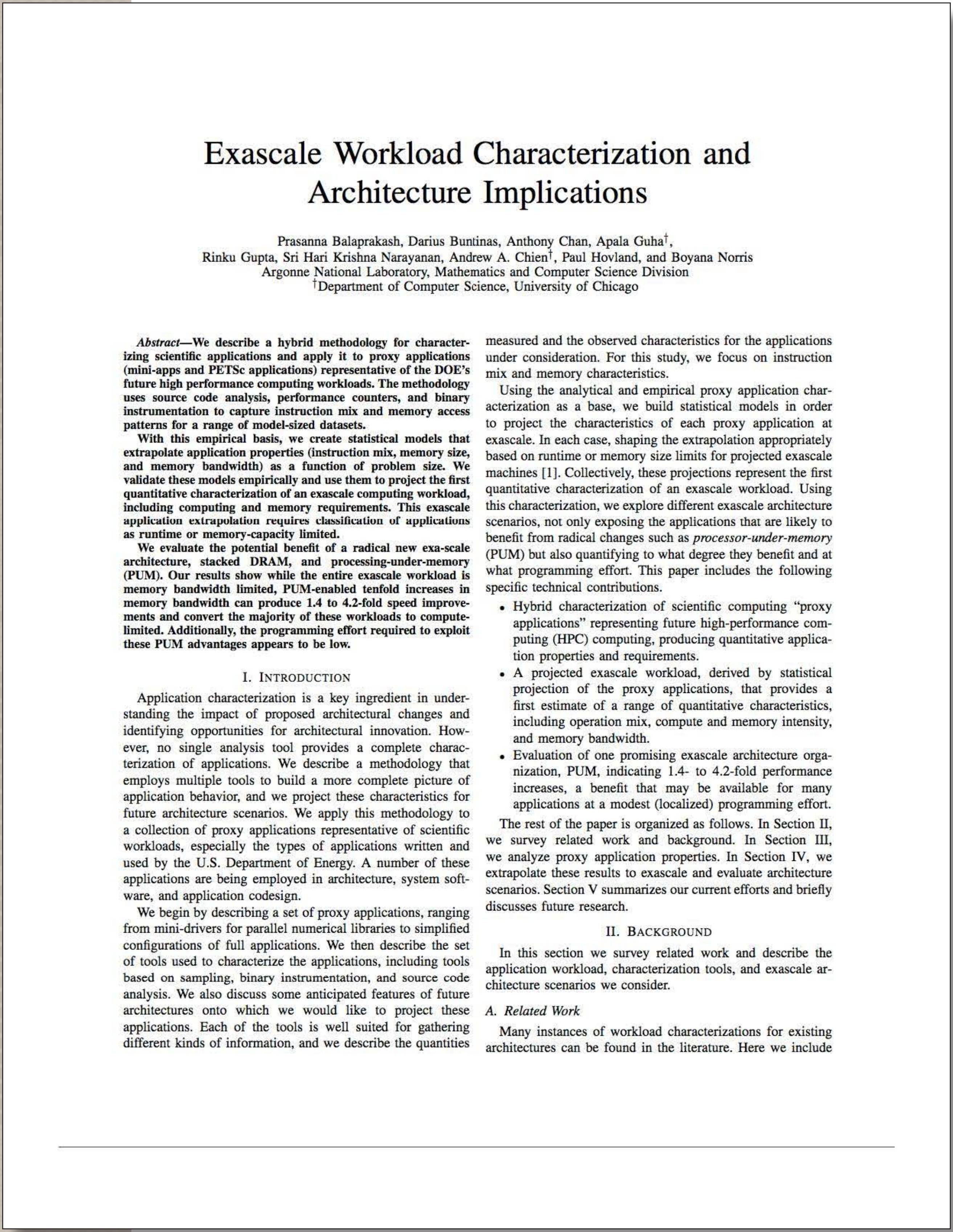
Most approaches to optimization beyond the MPI standard involve compiler support. A compiler can be used to perform static analysis that incorporates MPI semantic knowledge to provide several optimizations, mostly aimed at reducing the code in the critical path (e.g., eliminating error checking or data look ups) or improving resource utilization (e.g., reusing communication buffers). OMPI [19] is an optimizing preprocessor that performs exactly these types of optimizations.

CC-MPI [14] is an MPI library that provides an additional API for managing multicast communication groups for eth-

1530-2075/11 $26.00 © 2011 IEEE
DOI 10.1109/IPDPS.2011.366

IEEE computer society

Appendix D: Publications (cont.)



Exascale Workload Characterization and Architecture Implications

Prasanna Balaprakash, Darius Buntinas, Anthony Chan, Apala Guha[†],
Rinku Gupta, Sri Hari Krishna Narayanan, Andrew A. Chien[†], Paul Hovland, and Boyana Norris
Argonne National Laboratory, Mathematics and Computer Science Division
[†]Department of Computer Science, University of Chicago

***Abstract*—We describe a hybrid methodology for characterizing scientific applications and apply it to proxy applications (mini-apps and PETSc applications) representative of the DOE's future high performance computing workloads. The methodology uses source code analysis, performance counters, and binary instrumentation to capture instruction mix and memory access patterns for a range of model-sized datasets.**

With this empirical basis, we create statistical models that extrapolate application properties (instruction mix, memory size, and memory bandwidth) as a function of problem size. We validate these models empirically and use them to project the first quantitative characterization of an exascale computing workload, including computing and memory requirements. This exascale application extrapolation requires classification of applications as runtime or memory-capacity limited.

We evaluate the potential benefit of a radical new exa-scale architecture, stacked DRAM, and processing-under-memory (PUM). Our results show while the entire exascale workload is memory bandwidth limited, PUM-enabled tenfold increases in memory bandwidth can produce 1.4 to 4.2-fold speed improvements and convert the majority of these workloads to compute-limited. Additionally, the programming effort required to exploit these PUM advantages appears to be low.

I. Introduction

Application characterization is a key ingredient in understanding the impact of proposed architectural changes and identifying opportunities for architectural innovation. However, no single analysis tool provides a complete characterization of applications. We describe a methodology that employs multiple tools to build a more complete picture of application behavior, and we project these characteristics for future architecture scenarios. We apply this methodology to a collection of proxy applications representative of scientific workloads, especially the types of applications written and used by the U.S. Department of Energy. A number of these applications are being employed in architecture, system software, and application codesign.

We begin by describing a set of proxy applications, ranging from mini-drivers for parallel numerical libraries to simplified configurations of full applications. We then describe the set of tools used to characterize the applications, including tools based on sampling, binary instrumentation, and source code analysis. We also discuss some anticipated features of future architectures onto which we would like to project these applications. Each of the tools is well suited for gathering different kinds of information, and we describe the quantities measured and the observed characteristics for the applications under consideration. For this study, we focus on instruction mix and memory characteristics.

Using the analytical and empirical proxy application characterization as a base, we build statistical models in order to project the characteristics of each proxy application at exascale. In each case, shaping the extrapolation appropriately based on runtime or memory size limits for projected exascale machines [1]. Collectively, these projections represent the first quantitative characterization of an exascale workload. Using this characterization, we explore different exascale architecture scenarios, not only exposing the applications that are likely to benefit from radical changes such as *processor-under-memory* (PUM) but also quantifying to what degree they benefit and at what programming effort. This paper includes the following specific technical contributions.

- Hybrid characterization of scientific computing "proxy applications" representing future high-performance computing (HPC) computing, producing quantitative application properties and requirements.
- A projected exascale workload, derived by statistical projection of the proxy applications, that provides a first estimate of a range of quantitative characteristics, including operation mix, compute and memory intensity, and memory bandwidth.
- Evaluation of one promising exascale architecture organization, PUM, indicating 1.4- to 4.2-fold performance increases, a benefit that may be available for many applications at a modest (localized) programming effort.

The rest of the paper is organized as follows. In Section II, we survey related work and background. In Section III, we analyze proxy application properties. In Section IV, we extrapolate these results to exascale and evaluate architecture scenarios. Section V summarizes our current efforts and briefly discusses future research.

II. Background

In this section we survey related work and describe the application workload, characterization tools, and exascale architecture scenarios we consider.

A. Related Work

Many instances of workload characterizations for existing architectures can be found in the literature. Here we include

Appendix D: Publications (cont.)

- *Application Explorations for Future Interconnects*, R.F. Barrett, C.T. Vaughan, S.D. Hammond, and D. Roweth. In Workshop on Large Scale Parallel Processing, at the IEEE International Parallel & Distributed Processing Symposium (IPDPS) Meeting, 2013.
- *Assessing the Validity of the Role of Mini- Applications in Predicting Key Performance Characteristics of Scientific and Engineering Applications*, R.F. Barrett, P.S. Crozier, S.D. Hammond, M.A. Heroux, P.T. Lin, T.G. Trucano, and C.T. Vaughan. In preparation.
- A *Case for Dual Stack Virtualization: Consolidating HPC and Commodity Applications in the Cloud*, Brian Kocoloski, Jiannan Ouyang, and John Lange. In Proceedings of the Third ACM Symposium on Cloud Com- puting, SoCC '12, New York, NY, USA, 2012. ACM.
- *Charm++ for productivity and performance*, L. Kale et al., 2011.
- *Communication optimization beyond mpi*, Andrew Friedley and Andrew Lumsdaine. In Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum, IPDPSW '11. IEEE Computer Society, 2011.
- *Emerging High Performance Computing Systems and Next Generation Engineering Analysis Applications*, J.A. Ang, R.F. Barrett, S.D. Hammond, and A.F. Rodrigues. In Pacific Rim Workshop on Innovations in Civil Infrastructure Engineering. National Taiwan University of Science and Technology, 2013. SAND 2013-0054P.
- *Exascale Computing and the Role of Co-design. In High Performance Computing: From Grids and Clouds to Exascale*, chapter. J. Ang et al. IOS Press Inc, 2011.
- *Exascale Design Space Exploration and Co-design*. R.F. Barrett, D.W. Doerfler, S.S. Dosanjh, S.D. Hammond, K.S. Hemmert, M.A. Heroux, P.T. Lin, J.P. Lutjiens, K.T. Pedretti, A.F. Rodrigues, and T.G. Trucano. Under review, 2012.
- *Exascale Work-load Characterization and Architecture Implications*, P. Balaprakash, D. Buntinas, A. Chan, A. Guha, R. Gupta, S. H. K. Narayanan, A. A. Chien, P. Hovland, and B. Norris. Technical Report ANL/MCS-P3013-0712, Argonne National Laboratory, July 2012.
- *Exploring latency-power tradeoffs in deep non-volatile memory hierarchies*, Doe Hyun Yoon, Tobin Gonzalez, Parthasarathy Ranganathan, and Robert S. Schreiber. In Proceedings of the 9th conference on Computing Frontiers, CF '12, pages 95–102, New York, NY, USA, 2012. ACM.

Appendix D: Publications (cont.)

- *GPU Acceleration of Data Assembly in Finite Element Methods and Its Energy implications*, L.Tang, X.Sharon Hu, Danny Z. Chen, M. Niemier, R.F. Barrett, S.D. Hammond, and G. Hsieh. 2013.
- *Improving Performance via Mini-applications*. M.A. Heroux et al. Technical Report SAND2009-5574, Sandia National Laboratories, September 2009.
- *Mantevo views: A flexible system for gathering and analyzing data for the mantevo project*, Cameron Christesen. College of St. Benedict/St. John's University Senior Honors Thesis, 2007. Undergraduate Thesis.
- *Micro-applications for communication data access patterns and mpi datatypes*, Timo Schneider, Robert Gerstenberger, and Torsten Hoefler. In Proceedings of EuroMPI, Lecture Notes in Computer Science, volume 7490. Springer, 2012.
- *MiniGhost: A Miniapp for Exploring Boundary Exchange Strategies Using Stencil Computations in Scientific Parallel Computing*, R.F. Barrett, C.T. Vaughan, and M.A. Heroux. Technical Report SAND2011- 5294832, Sandia National Laboratories, May 2011.
- *Navigating An Evolutionary Fast Path to Exascale*, R.F. Barrett, S.D. Hammond, C.T. Vaughan, D.W. Doerfler, M.A. Heroux, J.P. Luitjens, and D. Roweth. In Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS12), 2012. Extended version: Technical Report SAND 2012-4667, Sandia National Laboratories, 2012. http://www.sandia.gov/~rfbarre/pubs_list.html.
- *On the viability of checkpoint compression for extreme scale fault tolerance*, Dewan Ibtesham, Dorian Arnold, Kurt B. Ferreira, and Patrick G. Bridges. In Proceedings of the 2011 international conference on Parallel Processing - Volume 2, Euro-Par'11. Springer-Verlag, 2012.
- *A Performance Model with a Fixed Point for a Molecular Dynamics Kernel*, Robert W. Numrich and Michael A. Heroux. . Computer Science - Research and Development, 23(3-4), June 2009.
- *Report of Experiments and Evidence for ASC L2 Milestone 4467 - Demonstration of a Legacy Application's Path to Exascale*, B.W. Barrett et al. Technical Report SAND2012-1750, Sandia National Laboratories, 2012.

Appendix D: Publications (cont.)

- ***Summary of Work for ASC L2 Mile- stone 4465: Characterize the Role of the Mini-Application in Predicting Key Performance Characteristics of Real Applications***, R.F. Barrett, P.S. Crozier, S.D. Hammond, M.A. Heroux, P.T. Lin, T.G. Trucano, and C.T. Vaughan. Technical Report SAND2012-4667, Sandia National Laboratories, 2012.
- ***Task Mapping for Noncontiguous Allocations***, S.P. Feer, Z.D. Rhodes, N.W. Price, D.P. Bunde, and V.J. Leung. Technical report, 2013. SAND 2011-7334C, Submitted.
- ***Toward codesign in high performance computing systems***, R.F. Barrett, X. S. Hu, S.S. Dosanjh, S. Parker, M.A. Heroux, and J. Shalf. In Proceedings of the International Conference on Computer- Aided Design, ICCAD '12, New York, NY, USA, 2012. ACM.
- ***Towards Performance Predictive Application-dependent Workload Characterization***, Waleed Alkohlani and Jeanine Cook, In Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS12), 2012.
- ***Using the Cray Gemini Performance Counters***, K.T. Pedretti, C.T.Vaughan, K.S.Hemmert, and R.F. Barrett. In Proc. 55th Cray User Group Meeting, 2013.

SAND2013-xxxx