J. Parallel Distrib. Comput. 75 (2015) 107–122

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

# Assessing the role of mini-applications in predicting key performance characteristics of scientific and engineering applications

R.F. Barrett *, P.S. Crozier, D.W. Doerfler, M.A. Heroux, P.T. Lin, H.K. Thornquist, T.G. Trucano, C.T. Vaughan

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

## HIGHLIGHTS

- Proxies are being used to examine the performance of full application codes.
- We present a methodology for showing the link between full application codes and their proxies.
- We demonstrate this methodology using four applications and their proxies.

ARTICLE INFO

*Article history:*
Received 10 December 2013
Received in revised form
7 September 2014
Accepted 8 September 2014
Available online 28 September 2014

*Keywords:*
High performance computing
Validation
Scientific computing

ABSTRACT

Computational science and engineering application programs are typically large, complex, and dynamic, and are 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 "miniapps" 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 an application program. In this paper we introduce a methodology for assessing the ability of these miniapps to effectively represent these performance issues. We applied this methodology to three miniapps, examining the linkage between them and an application they are intended to represent. Herein we evaluate the fidelity of that linkage. This work represents the initial steps required to begin to answer the question, "Under what conditions does a miniapp represent a key performance characteristic in a full app?"

## 1. Introduction

Over the past several years computer architectures commonly employed by the computational science and engineering communities have remained relatively stable, with subsequent generations characterized by faster processors, memories, and interconnects. These changes have typically resulted in predictably faster runtimes for application programs. Emerging and expected future architectures, however, are presenting special challenges and opportunities that, if not effectively exploited, could result in slower runtimes. Driven by stagnant clock speeds, memory constraints, and power consumption restrictions [3,9], architects are exploring designs involving wider vector units, significantly more but less powerful processor cores, increased threading, more complex memory hierarchies, differently balanced node interconnects, etc.

The means for exploiting these capabilities must be investigated within the relevant context of their use. However, application programs targeting these machines are typically large, complex, dynamic, and often constrained by distribution limitations, and typically outlive the computing environments they originally targeted. They may be constructed using hundreds of thousands to millions of source lines of code, written in multiple programming languages and linking in several third-party libraries, and developed over decades by multiple generations of computational scientists. Thus examining and addressing the issues that will enable effective execution on these machines are prohibitive.

As a means of making tractable rapid explorations of scientific and engineering application programs in this context, 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,

* Corresponding author.
*E-mail address:* rfbarre@sandia.gov (R.F. Barrett).

http://dx.doi.org/10.1016/j.jpdc.2014.09.006
0743-7315/© 2014 Elsevier Inc. All rights reserved.

impact the runtime performance of a scientific or engineering application program.

These miniapps enable rapid exploration of key performance issues that impact a broad set of scientific application programs. Within the Department of Energy (DOE) they are being used to explore the above issues by a broad set of participants, including staff at DOE laboratories, universities, and vendors. Yet how can we be sure that these proxies adequately represent that which they are intended?

The key contribution of the work described herein is a methodology, rooted in formal verification and validation (V&V) efforts that have been developed for experimental science, for determining the quality of the miniapp as it pertains to a large, complex application code. We applied this methodology to three miniapps, examining the linkage between them and an application they are intended to represent. Herein we evaluate the fidelity of that linkage. This work represents the initial steps required to begin to answer the question, "Under what conditions does a miniapp represent a key performance characteristic in a full app?"

### 1.1. Related work

Application proxies have been part of the code developers' tool kit for many years. The LINPACK benchmark came into existence [12] as what we are now calling a miniapp. Sweep3d [1], sPPM [4], and the NAS benchmarks [5] in some sense may also be viewed in these terms. Large scale proxies, such as LULESH [18], are serving related purposes. The three DOE Office of Advanced Scientific Computing Research (ASCR) Co-Design Centers[1] (ExMatEx, CESAR, and ExaCT) have identified the development of proxy applications as a key component of their efforts. Miniapps, and other kinds of application proxies, are being used as part of machine procurements.[2] The Mantevo project [16] solidifies the application proxy idea, bringing a community-based focused effort to bear on the wide variety of explorations that application proxies can enable.

The validation methodology presented herein is the first formal means, that we are aware of, for understanding if, and how, an application proxy may be used to represent the behavior of a full application program. This work is strongly informed by techniques developed for experimental validation, as will be discussed in the following sections.

## 2. Overview of the Mantevo project

The Mantevo project [16] was motivated by questions arising from the Trilinos project [15]. These questions concerned the direction of some coding implementations targeting emerging and expected future architectures, including multi-core, many-core, and GPU-accelerated high performance computers. The goal was to create a suite of tools that placed important algorithms into an application-relevant context, enabling rapid exploration of issues and options and their mapping to computing platforms.

Mantevo miniapps are designed and developed to be a tool, useful throughout the co-design space [13], enabling agile exploration of a variety of issues that impact performance. Unlike a compact application, which is designed to capture some sort of physics behavior, miniapps are designed to capture some key performance issues in full applications. 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 key performance issues. Miniapps are developed and owned by application code teams. Miniapps are intended to be modified, and thus are generally limited to a few thousand source lines of code (SLOC), allowing for unconstrained modification. Once no longer useful for these purposes, a miniapp will be discarded. Mantevo miniapps are freely available as open source software under an LGPL license.

**Table 1**
List of Mantevo miniapps, release 1.0.

| Miniapp | Description |
|---|---|
| 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. |
| HPCCG | Intended to be the best approximation to an unstructured implicit finite element or finite volume application in 800 lines or fewer. |
| miniFE | A proxy 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. |
| miniGhost | A difference stencil across a homogeneous three dimensional domain, targeting the inter-process communication halo exchange operation. |
| miniMD | The force computations in a typical molecular dynamics applications. The algorithms and implementation used closely mimic these same operations as performed in LAMMPS. |
| miniXyce | SPICE-style circuit simulator [24]. |

The miniapps in the first release of the Mantevo project are listed in Table 1. The first miniapp was HPCCG, which formed and solved a sparse linear system of equations. Although it provides an important capability, it was soon realized that in order to provide a stronger tie to applications of interest, the context in which the linear system is formed needed strengthening. The result was miniFE, putting the linear system into the context of an implicit finite element solver. Thus although HPCCG continues to serve an important role, miniFE will be examined in detail for purposes herein. We anticipate that this sort of situation will continue to occur as these miniapps are used in different situations.

## 3. Methodology

Miniapps are designed to provide a predictive capability for some key performance issue in a full application. Ensuring that a miniapp completely fulfills its intent is a difficult and probably an ongoing task. Further, the runtime behavior of a complex scientific application is typically problem dependent, and therefore it is important to understand the different ways that a code can be used and have a means for configuring the miniapp to mimic the important features under consideration. Thus our approach is to build up a "body of evidence" in support of the goals of a miniapp, combining formal verification and validation (V&V) techniques with our knowledge and experience bases.

*Verification* is the process of determining that a model implementation accurately represents the developer's 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. These terms are so defined by the 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 DOE and Department of Defense (DoD). That is, within the context of the intent of the comparisons of a model

[1] http://science.energy.gov/ascr/research/scidac/co-design/.

[2] http://www.nersc.gov/systems/trinity-nersc-8-rfp/draft-nersc-8-trinity-benchmarks/.