Addressing Reproducibility Challenges through Software Design, Benchmarking and Editorial Policies

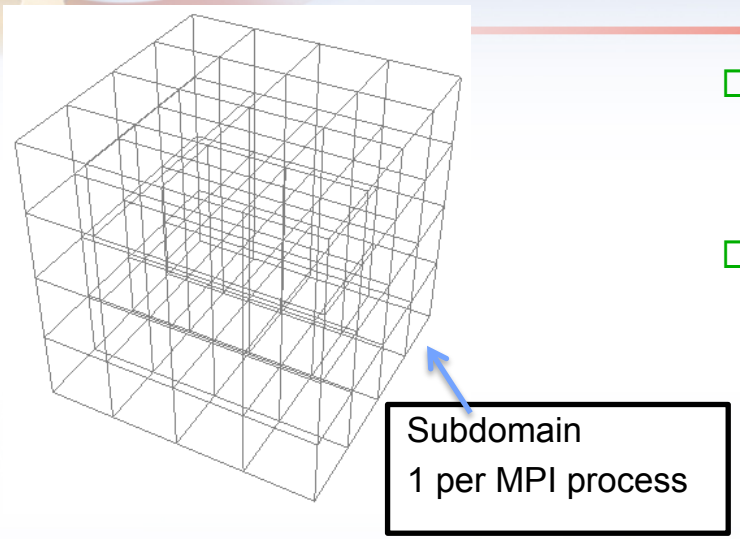
Michael Heroux
Sandia National Laboratories

Outline

- Tasking: New control layer in apps.
- C++ Templates: Paired data types.
- Containers: Docker.
- Benchmarks: HPCG.
- Publications: ACM TOMS.

Dynamic Tasking: Issues, no comprehensive solution

Classic HPC Application Architecture

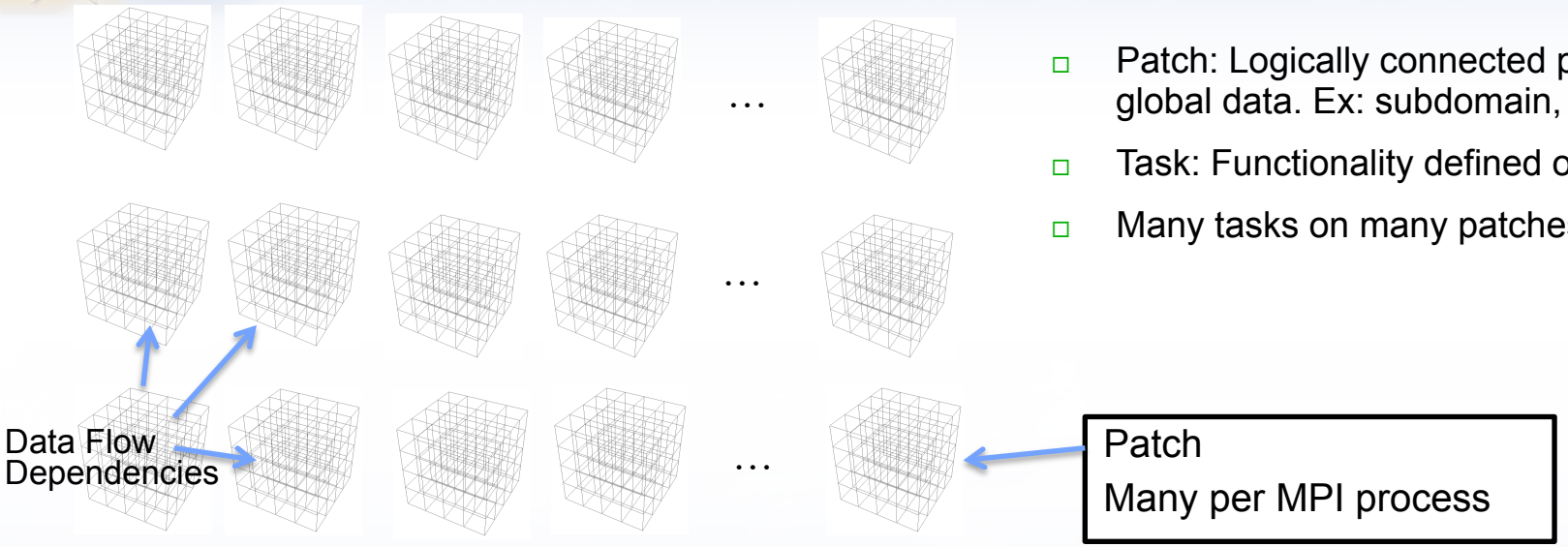


- Logically Bulk-Synchronous, SPMD
- Basic Attributes:
  - Halo exchange.
  - Local compute.
  - Global collective.

- Strengths:
  - Portable to many specific system architectures.
  - Separation of parallel model (SPMD) from implementation (e.g., message passing).
  - Domain scientists write sequential code within a parallel SPMD framework.
  - Supports traditional languages (Fortran, C).
  - Many more, well known.

- Weaknesses:
  - Not well suited (as-is) to emerging manycore systems.
  - Unable to exploit functional on-chip parallelism.
  - Difficult to tolerate dynamic latencies.
  - Difficult to support task/compute heterogeneity.

Task-centric/Dataflow Application Architecture



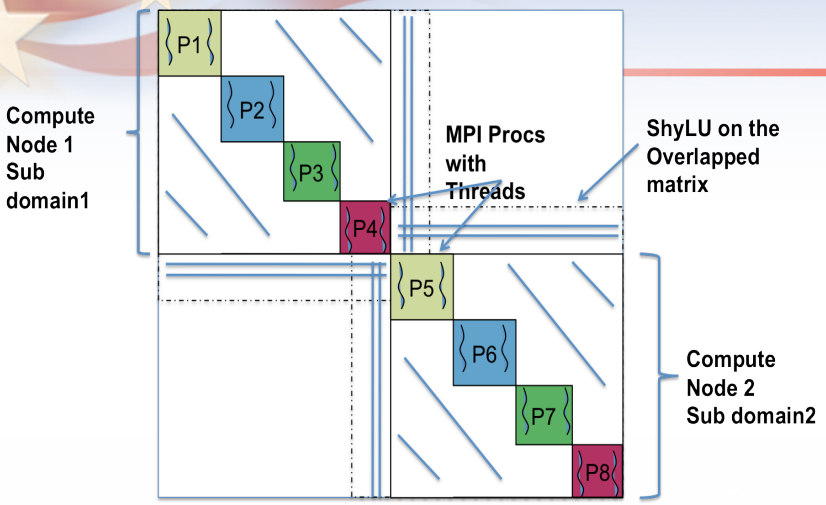
- Patch: Logically connected portion of global data. Ex: subdomain, subgraph.
- Task: Functionality defined on a patch.
- Many tasks on many patches.

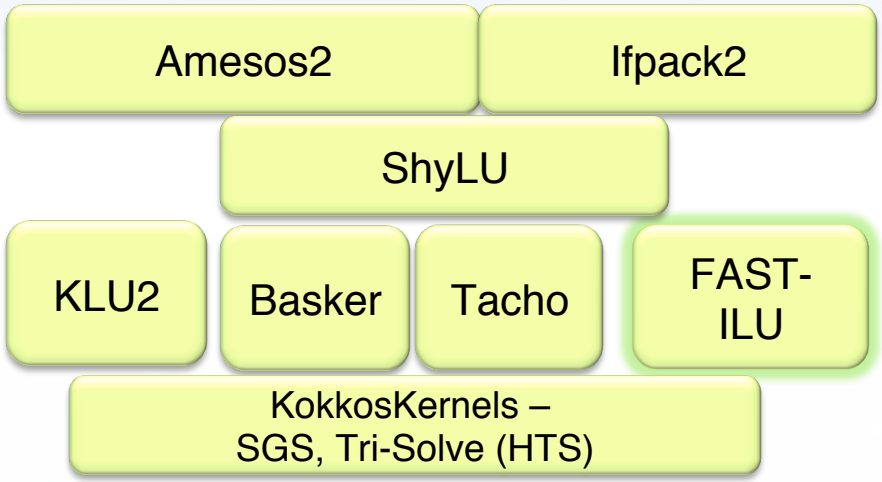
- Strengths:
  - Portable to many specific system architectures.
  - Separation of parallel model from implementation.
  - Domain scientists write sequential code within a parallel framework.
  - Supports traditional languages (Fortran, C).
  - Similar to SPMD in many ways.

- More strengths:
  - Well suited to emerging manycore systems.
  - Can exploit functional on-chip parallelism.
  - Can tolerate dynamic latencies.
  - Can support task/compute heterogeneity.

Trilinos/ShyLU and Subdomain Solvers : Overview

Led by Siva Rajamanickam, Sandia





- MPI+X based subdomain solvers
  - Decouple the notion of one MPI rank as one subdomain: Subdomains can span multiple MPI ranks each with its own subdomain solver using X or MPI+X
  - Epetra based solver, Tpetra interface still being developed
    - Trilinos Solver Factory a big step forward to get this done (*M. Hoemmen*)
- Subpackages of ShyLU: Multiple Kokkos-based options for on-node parallelism
  - Basker : LU or ILU (t) factorization (J. Booth)
  - Tacho: Incomplete Cholesky - IC (k) (K. Kim)
  - Fast-ILU: Fast-ILU factorization for GPUs (A. Patel)
- KokkosKernels: Coloring based Gauss-Seidel (M. Deveci), Triangular Solves
- **Experimental code base under active development.**

A Little Templating

Utilizing Scalar Type Templates

- Tpetra is a templated version of the Petra distributed linear algebra model in Trilinos.
  - Objects are templated on the underlying data types:

```
MultiVector<scalar=double, local_ordinal=int,
            global_ordinal=local_ordinal> …
CrsMatrix<scalar=double, local_ordinal=int,
            global_ordinal=local_ordinal> …
```

  - Examples:

```
MultiVector<double, int, long int> V;
CrsMatrix<float> A;
```

- Audi crankshaft model
- Dimension: 943,695
- Nonzeros: 77,651,847
- Belos, 10 RHS, Pseudo Block PCG, 1e-5, 4 core, MPI only

Speedup of float over double in Belos linear solver.

float	double	speedup
18 s	26 s	1.42x

Scalar	float	double	double-double	quad-double
Solve time (s)	2.6	5.3	29.9	76.5
Accuracy	10^{-6}	10^{-12}	10^{-24}	10^{-48}

Arbitrary precision solves using Tpetra and Belos linear solver package

FP Accuracy Analysis: FloatShadowDouble Datatype

- Templates enable new analysis capabilities
- Example: Float with "shadow" double.

```
class FloatShadowDouble {

public:
  FloatShadowDouble( ) {
    f = 0.0f;
    d = 0.0;  }
  FloatShadowDouble( const FloatShadowDouble & fd) {
    f = fd.f;
    d = fd.d;  }
…
inline FloatShadowDouble operator+= (const FloatShadowDouble & fd ) {
    f += fd.f;
    d += fd.d;
    return *this;  }
…
inline std::ostream& operator<<(std::ostream& os, const FloatShadowDouble& fd) {
  os << fd.f << "f " << fd.d << "d";  return os;}
```

FloatShadowDouble

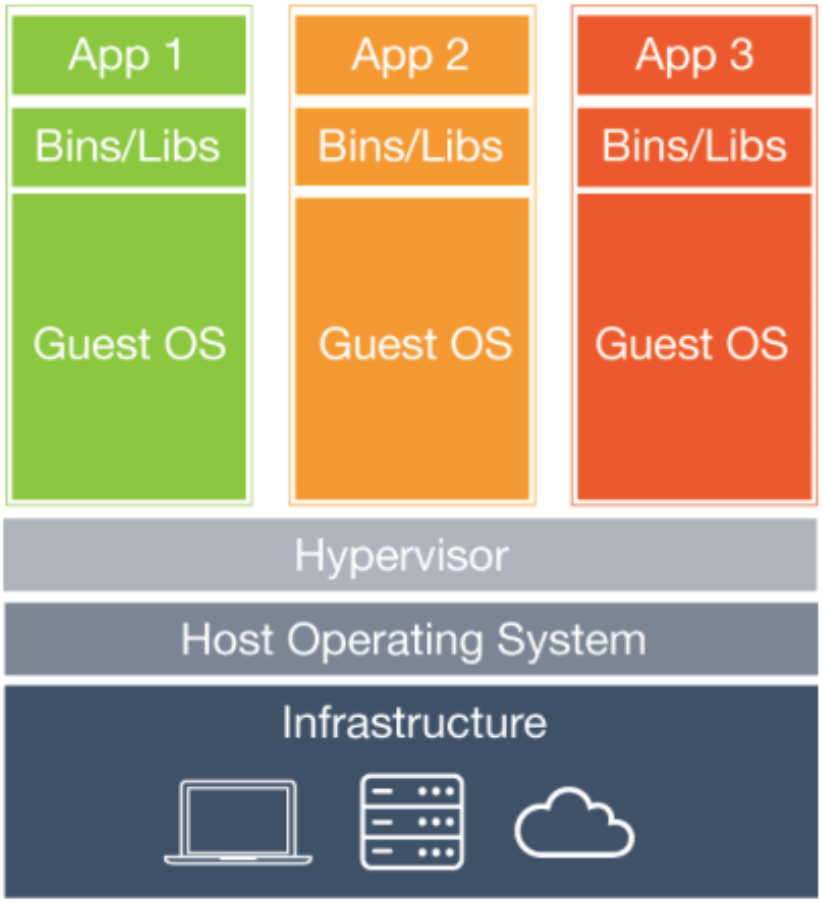
Sample usage:

```
#include “FloatShadowDouble.hpp”
Tpetra::Vector<FloatShadowDouble> x, y;
Tpetra::CrsMatrix<FloatShadowDouble> A;
A.apply(x, y);  // Single precision, but double results also computed, available
```

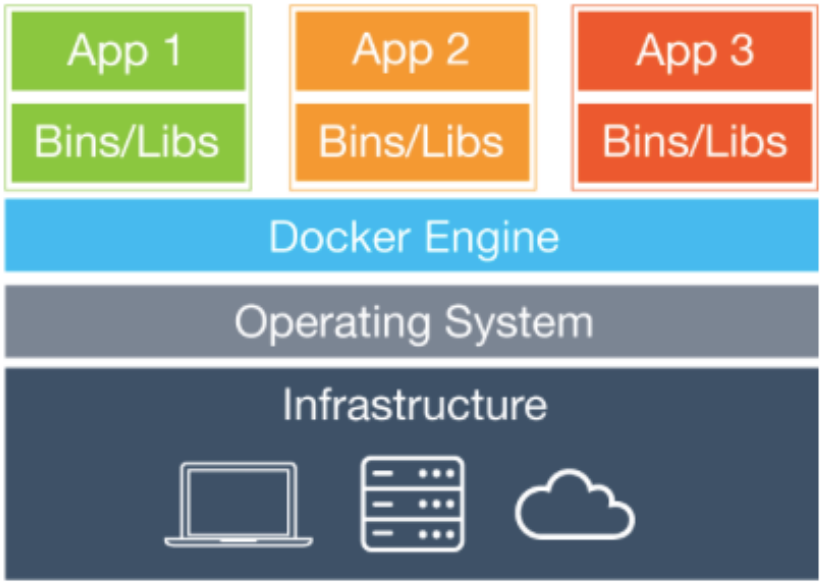
```
Initial Residual =          455.194f        455.194d
Iteration = 15   Residual = 5.07328f         5.07618d
Iteration = 30   Residual = 0.00147022f   0.00138466d
Iteration = 45   Residual = 5.14891e-06f  2.09624e-06d
Iteration = 60   Residual = 4.03386e-09f  7.91927e-10d
```

Containers: Reproducibility, and more!

Using Docker



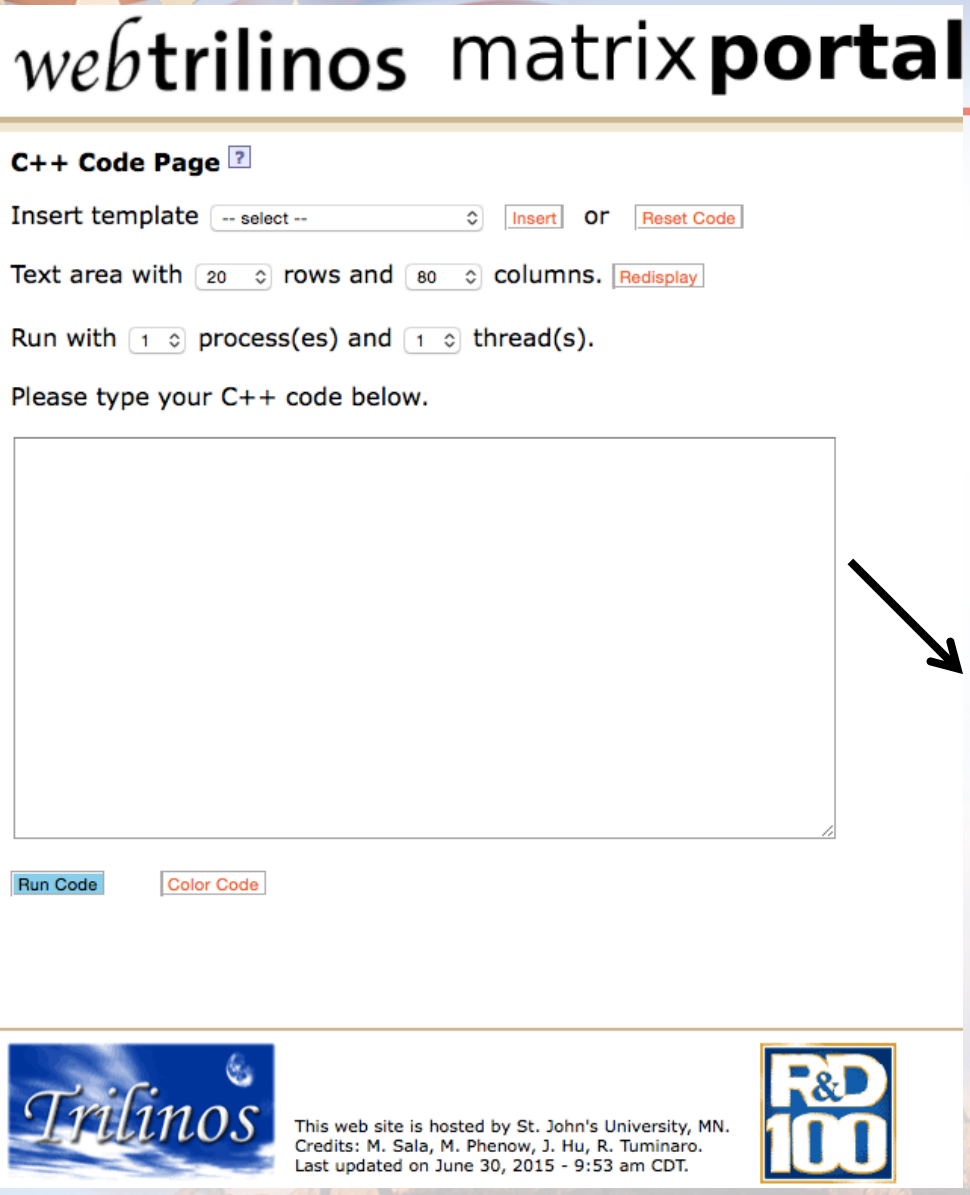
http://docker.com

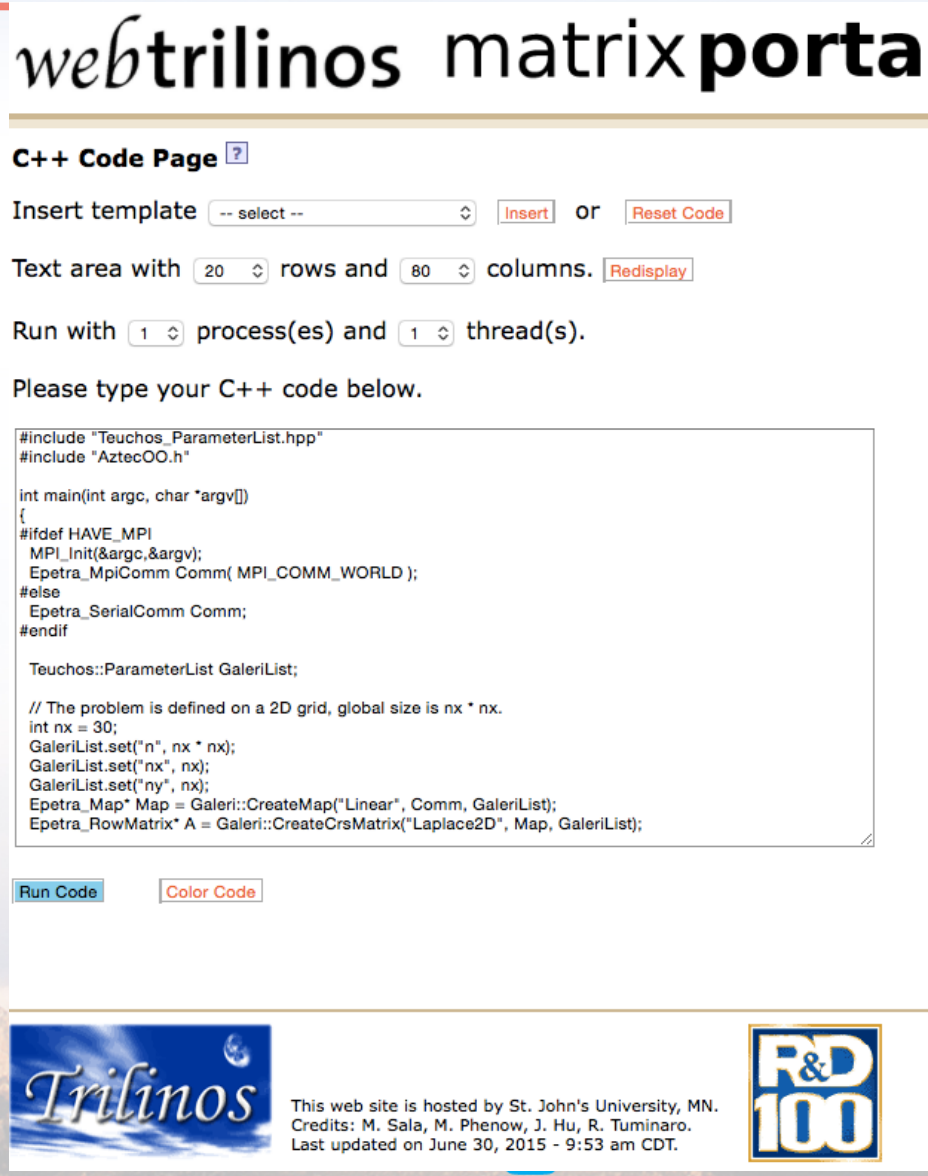


Typical Trilinos Cmake Script (edison)

```
cmake \
-D MPI_CXX_COMPILER="CC" \
-D MPI_C_COMPILER="cc" \
-D MPI_Fortran_COMPILER="ftn" \
-D Teuchos_ENABLE_STACKTRACE:BOOL=OFF \
-D Teuchos_ENABLE_LONG_LONG_INT:BOOL=ON \
-D Trilinos_ENABLE_Tpetra:BOOL=ON \
-D Tpetra_ENABLE_TESTS:BOOL=ON \
-D Tpetra_ENABLE_EXAMPLES:BOOL=ON \
-D Tpetra_ENABLE_EXPLICIT_INSTANTIATION:BOOL=ON \
-D Teuchos_ENABLE_EXPLICIT_INSTANTIATION:BOOL=ON \
-D TPL_ENABLE_MPI:BOOL=ON \
-D CMAKE_INSTALL_PREFIX:PATH="$HOME/opt/Trilinos/tpetraEval" \
-D BLAS_LIBRARY_DIRS="$LIBSCI_BASE_DIR/gnu/lib" \
-D BLAS_LIBRARY_NAMES="sci" \
-D LAPACK_LIBRARY_DIRS="$LIBSCI_BASE_DIR/gnu/lib" \
-D LAPACK_LIBRARY_NAMES="sci" \
-D CMAKE_CXX_FLAGS="-O3 -ffast-math -funroll-loops" \
\
..
```

WebTrilinos





Trilinos usage via Docker

- WebTrilinos Tutorial
  - https://hub.docker.com/r/sjdeal/webtrilinos
- http://johntfoster.github.io/posts/peridigm-without-building-via-Docker.html
  - docker pull johntfoster/trilinos
  - docker pull johntfoster/peridigm
  - docker run --name peridigm0 -d -v `pwd`:/output johntfoster/peridigm \
    Peridigm fragmenting_cylinder.peridigm
  - Etc…

Docker Features and Trends

- Initial motivator:
  - Containers essential to bridge HPC-HPDA gap.
- Many attractions:
  - MPI execution in Docker is scalable (evidence so far).
  - Improved reproducibility of results in several ways.
  - Smaller base system image: Only what Docker needs.
  - Lower barrier for complex SW.
  - And more.
- Docker container: New “tarfile”.
- DOE NERSC Cori System: “Shifter” supports Docker.

Reproducibility and Benchmarking

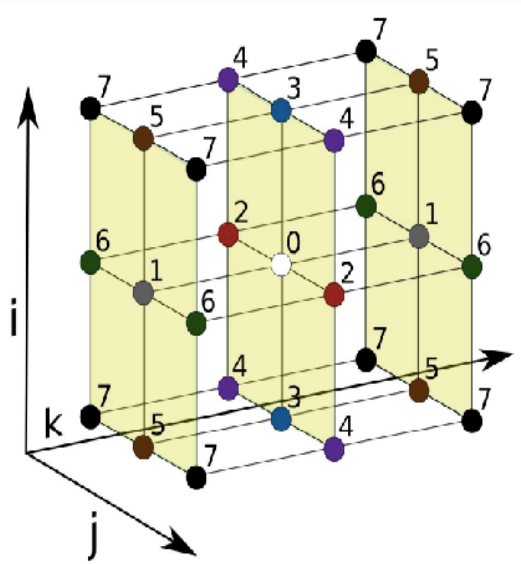
HPCG Snapshot

- High Performance Conjugate Gradients (HPCG).
- Solves $Ax=b$, A large, sparse, b known, x computed.
- An optimized implementation of PCG contains essential computational and communication patterns that are prevalent in a variety of methods for discretization and numerical solution of PDEs

- Patterns:
  - Dense and sparse computations.
  - Dense and sparse collectives.
  - Multi-scale execution of kernels via MG (truncated) V cycle.
  - Data-driven parallelism (unstructured sparse triangular solves).
- Strong verification (via spectral properties of PCG).

Model Problem Description

- Synthetic discretized 3D PDE (FEM, FVM, FDM).
- Zero Dirichlet BCs, Synthetic RHS s.t. solution = 1.
- Local domain: $(n_x \times n_y \times n_z)$
- Process layout: $(np_x \times np_y \times np_z)$
- Global domain: $(n_x * np_x) \times (n_y * np_y) \times (n_z * np_z)$
- Sparse matrix:
  - 27 nonzeros/row interior.
  - 8 – 18 on boundary.
  - Symmetric positive definite.



27-point stencil operator

Merits of HPCG

- Includes major communication/computational patterns.
  - Represents a minimal collection of the major patterns.
- Rewards investment in:
  - High-performance collective ops.
  - Local memory system performance.
  - Low latency cooperative threading.
- Detects/measures variances from bitwise reproducibility.
- Executes kernels at several (tunable) granularities:
  - nx = ny = nz = 104 gives
  - nlocal = 1,124,864; 140,608; 17,576; 2,197
  - ComputeSymGS with multicoloring adds one more level:
    - 8 colors.
    - Average size of color = 275.
    - Size ratio (largest:smallest): 4096
  - Provide a “natural” incentive to run a big problem.

HPL vs. HPCG: Bookends

- Some see HPL and HPCG as “bookends” of a spectrum.
  - Applications teams know where their codes lie on the spectrum.
  - Can gauge performance on a system using both HPL and HPCG numbers.

HPCG 3.0 Release, Nov 11, 2015

- Available on GitHub.com
  - Using GitHub issues, pull requests, Wiki.
- Intel, Nvidia optimized 3.0 version available.
- IBM has 2.4 optimized version.
- For ISC'16, HPCG 3.0 any new results should be obtained using 3.0 unless not possible.
- Quick Path option will make this easier.

Other Items

- Reference version on GitHub:
  - https://github.com/hpcg-benchmark/hpcg
  - Website: hpcg-benchark.org.
  - Mail list hpcg.benchmark@gmail.com
- HPCG used in SC15 Student Cluster Competition.
- HPCG-optimized kernels going into vendor libraries.
- Next event: ISC'16:
  - 63 entries so far (42 – ISC15, 25 – SC14, 15 – ISC14)
  - Quick Path option should accelerate adoption.

Detecting FP Variations (Reproducibility)

Residual=4.250796408610557858**83**e-08 (0x1.6d240066fda73p-25)
Residual=4.250796408610322939**54**e-08 (0x1.6d240066fd910p-25)
Residual=4.250796408610790792**89**e-08 (0x1.6d240066fdbd3p-25)
Residual=4.250796408610545285**68**e-08 (0x1.6d240066fda60p-25)
Residual=4.250796408610684913**77**e-08 (0x1.6d240066fdb33p-25)
Residual=4.250796408610590946**05**e-08 (0x1.6d240066fdaa5p-25)

- Nvidia version of HPCG:
  - Deterministic run optional, cost quantifiable.
- HPCG runs can be arbitrarily long (--rt=#secs).
  - Stress test for reproducibility.
- Nvidia, Intel versions: Heterogeneous.
  - Expose FP variability issues.

Creating Incentives to Improve Reproducibility via Publication Policies

Reproducibility & Independent Verification Requirement

- In order to publish a paper: *Someone other than the authors must be able to reproduce the computational results.*
- Latitude in “reproduce”:
  - Exactly the same numerical results?
  - Exactly the same runtime?
  - Close, in the opinion of an expert reviewer?
- What about:
  - Access to the same computing environment?
  - High end systems?
- Lots of challenges.
- But just the *expectation [threat]* can drive efforts…

Fruits of the Threat

- **Source management tools:** In order to guarantee that results can be reproduced, the software must be preserved so that the exact version used to produce results is available at a later date.
- **Use of other standard tools and platforms:** In order to reduce the complexity of an environment, standard software libraries and computing environments will be helpful.
- **Documentation:** Independent verification requires that someone else understand how to use your software.
- **Source code standards:** Improves the ability of others to read your source code.
- **Testing:** Investment in greater testing makes sense because the software will be used by others.
- **High-quality software engineering environment:** If a research team is serious about producing high-quality, reproducible and verifiable results, it will want to invest in a high-quality SE environment to improve team efficiency.

Evidence:
Cover letter excerpt from RCR candidate paper

Thank you for taking the time to consider our paper for your journal.

XXX has agreed to undergo the RCR process should the paper proceed far enough in the review process to qualify. ***To make this easier we have preserved the exact copy of the code used for the results (including additional code for generating detailed statistics that is not in the library version of the code).***

ACM's First "Artifact Evaluation"

- ACM Algorithms
  - Algorithm Section of CACM established Feb 1961
    - First Editor: J.H. Wegstein, NBS
  - Submissions refereed
  - Program text printed in pages of CACM
    - Algorithm 1: QuadI (R.J. Herbold, NBS), Feb, 1961
  - Continued until 1975 when programs became too large to print

Slide courtesy Ron Boisvert

ACM TOMS

- *Transactions on Mathematical Software* (TOMS)
  - First in ACM's *Transactions* series (est. 1975)
- Focus on numerical software engineering
  - Theoretical foundations of numeric, symbolic, algebraic, and geometric computing applications, as well as practical aspects of analysis and construction of mathematical software, and the interaction of programs and architecture.
- Assumed ownership of ACM Algorithms.

Slide courtesy Ron Boisvert

http://toms.acm.org/

TOMS “Algorithms”

- 1/3 of published papers are “Algorithm” papers with code
  - Software focused on solving a generic mathematical problem
  - First: Algorithm 493: Zeros of a Real Polynomial
  - > 450 published since 1975
- Refereed
  - For: completeness, portability, usability/extensibility, documentation
- Distribution
  - Originally: Algorithms Distribution Service (magnetic tape)
  - Late 1980s: netlib (first email, then ftp, then web)
  - Today: ACM Digital Library

Slide courtesy Ron Boisvert

TOMS “Algorithms”: Context

- Numerical software community has always had a strong focus on sharing and reuse
  - *Sharing artifacts has not been a hard sell.*
- Software is typically a reusable component
  - *Packaging for reuse/replay has not been difficult*
  - *Zip/tar file with code, makefile, sample drivers, data*
- Emphasis on portability has made code long-lasting
  - In 2002 Tim Hopkins replicated results of all 300 published algorithms
  - Many were straightforward, some required extensive work
  - Many code improvements implemented to extend lifetimes

Slide courtesy Ron Boisvert

Reproducibility Terminology

V. Stodden, D. H. Bailey, J. Borwein, R. J. LeVeque, W. Rider, and W. Stein. 2013. Setting the Default to Reproducible: Reproducibility in Computational and Experimental Mathematics. (2013). http://icerm.brown.edu/html/programs/topical/tw12 5 rcem/icerm report.pdf

- **Reviewable Research**. The descriptions of the research methods can be independently assessed and the results judged credible. (This includes both traditional peer review and community review, and does not necessarily imply reproducibility.)
- **Replicable Research**. Tools are made available that would allow one to duplicate the results of the research, for example by running the authors' code to produce the plots shown in the publication. (Here tools might be limited in scope, e.g., only essential data or executables, and might only be made available to referees or only upon request.)
- **Confirmable Research**. The main conclusions of the research can be attained independently without the use of software provided by the author. (But using the complete description of algorithms and methodology provided in the publication and any supplementary materials.)
- **Auditable Research**. Sufficient records (including data and software) have been archived so that the research can be defended later if necessary or differences between independent confirmations resolved. The archive might be private, as with traditional laboratory notebooks.
- **Open or Reproducible Research.** Auditable research made openly available. This comprised well-documented and fully open code and data that are publicly available that would allow one to (a) fully audit the computational procedure, (b) replicate and also independently reproduce the results of the research, and (c) extend the results or apply the method to new problems.

ACM Transactions on
Mathematical Software

ACM TOMS

- TOMS RCR Initiative: Referee Data.
- Why TOMS? Tradition of real software that others use.
- Two categories: Algorithms, Research.
- TOMS Algorithms Category:
  - Software Submitted with manuscript.
  - Both are thoroughly reviewed.
- TOMS Research Category:
  - Stronger: Previous implicit “real software” requirement is explicit.
  - New: Special designation for replicated results.

RCR Process: Step by step

1. RCR review request:
   - When authors submit a manuscript for review,
   - optional request for RCR review
   - Conducted independently from the standard process.
2. Standard reviewer assignment: No change.
3. RCR suitability review:
   - Concurrent with assigning standard reviewers
   - EiC/AE briefly review for RCR suitability.
   - RCR suitability decision may be delayed until the first round of standard reviews is complete.

RCR Process

4. RCR reviewer assignment (assuming suitable):
   - RCR review concurrent with standard peer review.
   - EiC/AE assigns RCR reviewer.
   - RCR reviewer sole responsibility replicate manuscript computational results.
   - Open review.
5. RCR review process:
   - RCR review requires multi-faceted approach.
   - Editors advise RCR reviewer on acceptable approaches.
   - BUT: Ultimately the RCR reviewer has the responsibility to declare whether or not computational results in the manuscript are replicated.
   - RCR reviewers document replication details.

RCR Process

6. RCR Determination:
   - We anticipate (in this intro phase) that RCR manuscripts will almost surely receive RCR designation.
   - Important: Reduce authors' risk as volunteers.
7. RCR Review Failure:
   - Some risk now and in the future that RCR efforts will fail.
   - If so, must acknowledge manuscript is not ready.
   - Intro phase: EiC/AE personally manages this situation, work with authors to avoid rejecting manuscript outright.
   - Later: No special treatment.

RCR Process

8. Publication:
   - Published with a special RCR designation.
   - RCR referee will be acknowledged in published paper.
   - RCR referee's report will be published as a TOMS article.
     - Incentive for RCR reviewers.
     - Written record of process for understanding/improving.

Replication Methods

1. **Independent replication:** The authors provide the RCR reviewer access to, or sufficient description of, the computational platform used to produce the manuscript results. Access could be:
   - A direct transfer of software to the reviewer or a pointer to an archive of the software, and a description of a commonly available computer system the reviewer can access.
   - A guest account and access to the software on the system used to produce the results.
   - Detailed observation of the authors replicating the results.

Replication Methods

2. Review of computational results artifacts:
   - In some situations, authors may not be able to readily replicate computational results.
   - Results may be from a system that is no longer available, or may be on a leadership class computing system to which access is very limited.
   - In these situations, careful documentation of the process used to produce results could be sufficient for an RCR designation.
   - In this case, the software should have its own substantial verification process to give the reviewer confidence that computations were performed correctly.
   - If timing results are reported, the authors' artifacts should include validation testing of the timers used to report results.

Announcement

Editorial: ACM TOMS Replicated Computational Results Initiative

MICHAEL A. HEROUX, Sandia National Laboratories

The scientific community relies on the peer review process for assuring the quality of published material, the goal of which is to build a body of work we can trust. Computational journals such as the *ACM Transactions on Mathematical Software* (TOMS) use this process for rigorously promoting the clarity and completeness of content, and citation of prior work. At the same time, it is unusual to independently confirm computational results.

ACM TOMS has established a *Replicated Computational Results* (RCR) review process as part of the manuscript peer review process. The purpose is to provide independent confirmation that results contained in a manuscript are replicable. Successful completion of the RCR process awards a manuscript with the Replicated Computational Results Designation.

This issue of ACM TOMS contains the first [Van Zee and van de Geijn 2015] of what we anticipate to be a growing number of articles to receive the RCR designation, and the related RCR reviewer report [Willenbring 2015]. We hope that the TOMS RCR process will serve as a model for other publications and increase the confidence in and value of computational results in TOMS articles.

CCS Concepts: • **General and reference** → **Verification;** *Validation*; • **Software and its engineering** → **Formal software verfication**

General Terms: Verification

Additional Key Words and Phrases: Replicated computational results, reproducibility, validation, publication

ACM Reference Format:
Michael A. Heroux. 2015. Editorial: ACM TOMS replicated computational results initiative. ACM Trans. Math. Softw. 41, 3, Article 13 (May 2015), 5 pages.
DOI: http://dx.doi.org/10.1145/2743015

14

BLIS: A Framework for Rapidly Instantiating BLAS Functionality

FIELD G. VAN ZEE and ROBERT A. VAN DE GEIJN, The University of Texas at Austin

The BLAS-like Library Instantiation Software (BLIS) framework is a new infrastructure for rapidly instantiating Basic Linear Algebra Subprograms (BLAS) functionality. Its fundamental innovation is that virtually all computation within level-2 (matrix-vector) and level-3 (matrix-matrix) BLAS operations can be expressed and optimized in terms of very simple kernels. While others have had similar insights, BLIS reduces the necessary kernels to what we believe is the simplest set that still supports the high performance that the computational science community demands. Higher-level framework code is generalized and implemented in ISO C99 so that it can be reused and/or reparameterized for different operations (and different architec-

DL includes data from replication experiment

Replicated Computational Results (RCR) Report for "BLIS: A Framework for Rapidly Instantiating BLAS Functionality"

JAMES M. WILLENBRING, Sandia National Laboratories

"BLIS: A Framework for Rapidly Instantiating BLAS Functionality" by Field G. Van Zee and Robert A. van de Geijn (see: http://dx.doi.org/10.1145/2764454) includes single-platform BLIS performance results for both level-2 and level-3 operations that is competitive with OpenBLAS, ATLAS, and Intel MKL. A detailed description of the configuration used to generate the performance results was provided to the reviewer by the authors. All the software components used in the comparison were reinstalled and new performance results were generated and compared to the original results. After completing this process, the published results are deemed replicable by the reviewer.

15

Sandia National Laboratories

Status

- First RCR paper available: TOMS 41:3
  - Editorial introduction.
  - van Zee & van de Geijn, BLIS paper.
  - Referee report.
- Goal: 1 RCR paper per TOMS issue.
  - Hogg & Scott next.

Big Picture of TOMS RCR

- Improve science.
  - Quality of prose in publications: Good.
  - Quality of data: (Very!) poor.
- So bad now:
  - Trust comes from seeing a “cloud” of similar papers with similar results.
  - Which could still be wrong (built on a common bad piece).
  - Replicability: First step toward improvement.
- Engage a “dark portion” of the R&D community.
  - Reviewers not among typical reviewer pool.
  - Practitioners, users. Expert at use of Math SW.

Remaining Issues

- Replication initiative
  - “Replicated” branding in the PDF and in the DL
    - Would like uniformity across ACM
  - Can the process be sustained?
  - Will replicability reports have an (artificial) negative impact
    - on journal impact factor?
- Archiving of TOMS software/data in the DL
  - Separate DOIs?
  - Additional metadata?

Summary

- Containerization: New (for HPC) tool for reproducibility (and much more).
- HPCG can be a reproducibility tester.
- Journal, funding agency policies can provide productivity incentives.
- Replicability expectations:
  - Better productivity practices are a natural reaction.
- Funding Proposals:
  - We expect data management plans.
  - Can we start expecting a SW quality management plan?

Final Thought: Commitment to Quality

Canadian engineers' oath (taken from Rudyard Kipling):

My Time I will not refuse; my Thought I will not grudge; my Care I will not deny toward the honour, use, stability and perfection of any works to which I may be called to set my hand.

http://commons.bcit.ca/update/2010/11/bcit-engineering-graduates-earn-their-iron-rings