HPCG

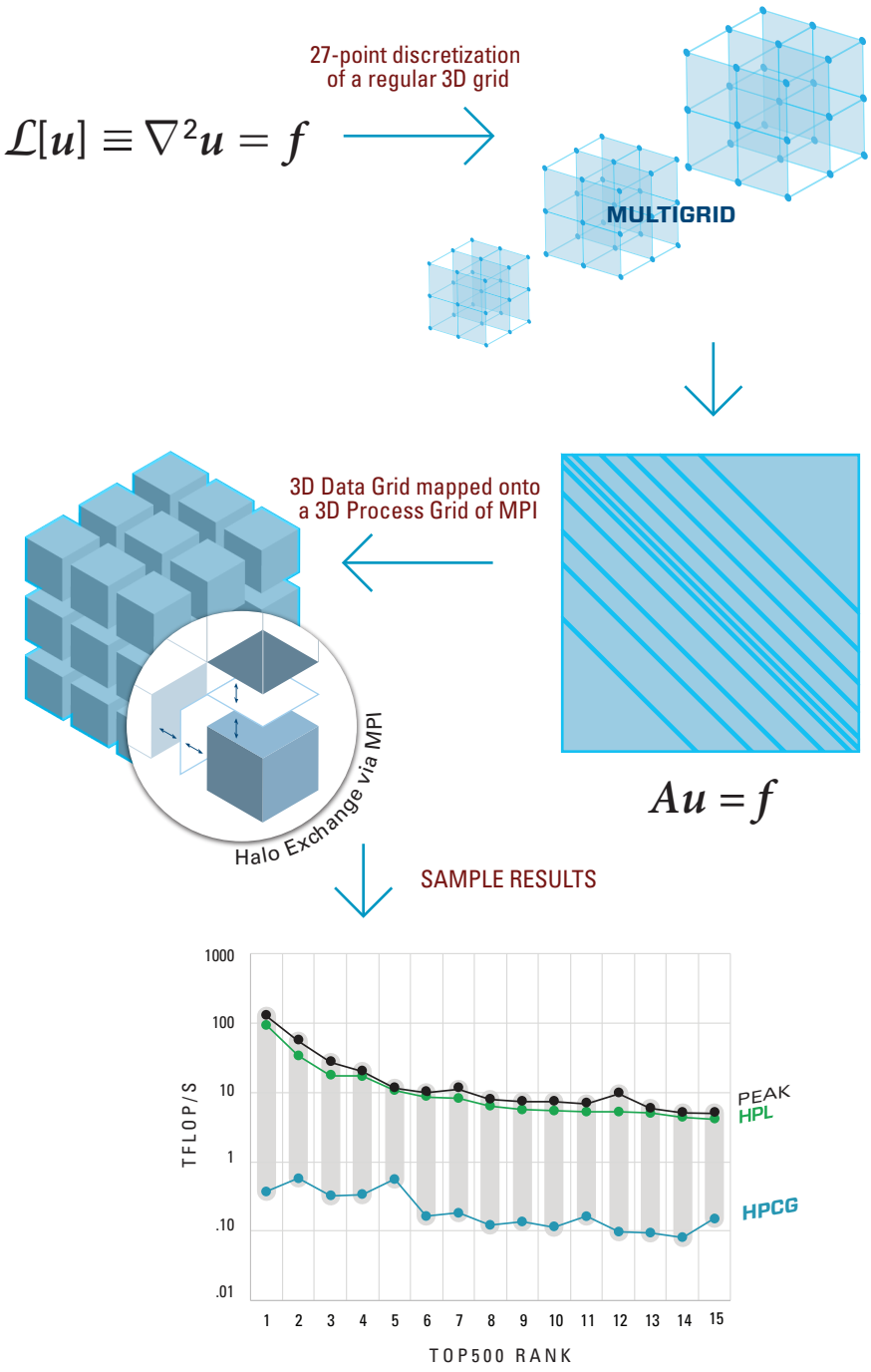
INNOVATIVE COMPUTING LABORATORY

The HPC Conjugate Gradient (HPCG) benchmark uses a Preconditioned Conjugate Gradient (PCG) algorithm to measure the performance of HPC platforms with respect to frequently observed, and yet challenging, patterns of execution, memory access, and global communication.

The PCG implementation uses a regular 27-point stencil discretization in 3 dimensions of an elliptic Partial Differential Equation (PDE). The 3D domain is scaled to fill a 3D virtual process grid of all available MPI process ranks. The CG iteration includes a local and symmetric Gauss-Seidel preconditioner, which computes a forward and a back solve with a triangular matrix. All of these features combined allow HPCG to deliver a more accurate performance metric for modern HPC hardware architectures.

PRECONDITIONED CONJUGATE GRADIENT SOLVER

$p_0 \leftarrow x_0$, $r_0 \leftarrow b - Ap_0$
for $i = 1, 2,$ to max_iterations **do**
  $z_i \leftarrow M^{-1} r_{i-1}$
  if $i = 1$ **then**
    $p_i \leftarrow z_i$
    $\alpha_i \leftarrow$ dot_prod(r_{i-1}, z_i)
  else
    $\alpha_i \leftarrow$ dot_prod(r_{i-1}, z_i)
    $\beta_i \leftarrow \alpha_i / \alpha_{i-1}$
    $p_i \leftarrow \beta_i p_{i-1} + z_i$
  end if
  $\alpha_i \leftarrow$ dot_prod(r_{i-1}, z_i)/dot_prod(p_i, Ap_i)
  $x_{i+1} \leftarrow x_i + \alpha_i p_i$
  $r_i \leftarrow r_{i-1} - \alpha_i Ap_i$
  if $\|r_i\|_2 <$ tolerance **then**
    STOP
  end if
end for



ICL UT INNOVATIVE COMPUTING LABORATORY THE UNIVERSITY of TENNESSEE

IN COLLABORATION WITH Sandia National Laboratories

SPONSORED BY U.S. DEPARTMENT OF ENERGY, NNSA National Nuclear Security Administration

CITR CENTER FOR INFORMATION TECHNOLOGY RESEARCH

FIND OUT MORE AT http://icl.cs.utk.edu/**hpcg**

HPCG

A New Yardstick for Supercomputers

The High Performance Conjugate Gradients (HPCG) benchmark implements the preconditioned Conjugate Gradient method with a local symmetric Gauss-Seidel preconditioner and a three-level multigrid and is quickly becoming the new HPC metric of choice.

In doing so, HPCG is designed to measure performance that is representative of many important scientific codes, or computational patterns, with low compute-to-data-access ratios. To simulate these patterns, commonly found in real applications, HPCG exhibits the same irregular accesses to memory as well as fine-grain recursive computations that dominate many important HPC workloads.

In contrast to the new HPCG metric, HPL factors and solves a large dense system of linear equations using Gaussian elimination with partial pivoting. Thus, the dominant calculations in the HPL implementation are dense matrix-matrix multiplication and related kernels. With proper organization of the computation, data access is predominantly unit-stride and is mostly hidden by concurrently performing computations on previously retrieved data. This kind of algorithm strongly favors computers with very high floating-point performance and only adequate streaming memory systems.

In general, a well-rounded computer system should be designed to execute both computational patterns efficiently, as this combination allows the system to run a broad mix of applications and run them well. For a metric to test the true capabilities of a general-purpose computer, it should stress both patterns. However, HPL only stresses the floating-point oriented patterns, and, as a metric, is incapable of measuring the irregular data access patterns.

Another issue with the existing performance metrics stems from the emergence of accelerators, which are extremely effective (relative to multicore CPUs) with compute-intensive patterns, but much less so with the irregular patterns. For many users, HPL results show a skewed picture relative only to compute-bound application performance. This is especially true on machines that are heavily biased toward irregular codes, and not so on machines that use accelerators for the majority of the computational power.

HPCG Algorithmic Design

The HPCG Benchmark alleviates many of the problems described above with help of the following design principles:

Provide coverage of the major communication and computational patterns: The major communication (global and neighborhood collectives) and computational patterns (vector updates, dot products, sparse matrix-vector multiplications, and local triangular solves) from our production differential equation codes, both implicit and explicit, are present in this benchmark. Emerging asynchronous collectives and other latency-hiding techniques can be explored in the context of HPCG and aid in their adoption and optimization on future systems.

Represents a minimal collection of the major patterns: HPCG is the smallest benchmark code containing these major patterns, while at the same time representing a real mathematical computation, which aids in Validation and Verification efforts.

Rewards investment in high-performance of collectives: Neighborhood and all-reduce collectives represent essential performance bottlenecks for our applications that can benefit from high-quality system design. Improving the performance of HPCG will improve the performance of these production codes.

Rewards investment in local memory system performance: The local processor performance of HPCG is largely determined by the effective use of the local memory system. Improvements in the implementation of HPCG data structures, compilation of HPCG code, and the performance of the underlying system will improve HPCG benchmark results and real application performance, *and* will inform application developers on new approaches to optimization of their own implementations.

Detects and measures variance values for bitwise identical computations: It is widely believed that future computer systems will not be able to provide deterministic execution paths for floating-point computations. Because floating-point addition is not associative, this means we will generally not have bitwise reproducible results, even when running the same exact computation twice on the same number of processors of the same system. This is in contrast with many of our MPI-only applications today, and presents a big challenge to applications that must certify their computational results and debug in the presence of bitwise variability. HPCG will make the deviation from bitwise reproducibility apparent. The reference code will be implemented in C++ (a commonly implemented subset of the C++11 standards and mostly compatible with C++98).

HPCG Overview

The HPCG code base performs the following steps:

Problem setup: Generates a synthetic symmetric positive definite (SPD) matrix A using the compressed sparse row format, and a corresponding right-hand-side vector b, and initial guess for x.

Preconditioner setup: Initializes the data structures for the local symmetric Gauss-Seidel preconditioner. The reference version uses a simple compressed sparse row representation for the lower and upper triangular matrices, each one as a separate matrix.

Verification and validation setup: Computes pre-conditions, post-conditions, and invariants that will aid in the detection of anomalies during the iteration phases due to, e.g., optimization errors.

Iteration: Performs m iterations, n times, using the same initial guess each time, where m and n are sufficiently large to test system uptime. By doing this, we can compare the numerical results for "correctness" at the end of each m-th iteration phase.

Post-processing and reporting: Reports a single timing result and other metrics. The HPCG benchmark generates a synthetic discretized three-dimensional partial differential equation model problem, and computes preconditioned conjugate gradient iterations for the resulting sparse linear system. The model problem can be interpreted as a single degree of freedom heat diffusion model with zero Dirichlet boundary conditions. The global domain dimensions are $n_x \times P_x \times n_y \times P_y \times n_z \times P_z$ where $n_x \times n_y \times n_z$ are the local sub-grid dimensions in the x, y, and z dimensions, respectively, assigned to each MPI process. These values are read from the data file hpcg.dat, or are passed in as command line arguments. The dimensions $P_x \times P_y \times P_z$, are a factoring of the MPI process space that is computed automatically in the HPCG setup phase. We impose ratio restrictions on both the local and global x, y, and z dimensions, which are enforced in the setup phase of HPCG.

The validation includes a symmetry test for the sparse matrix multiply with discretization matrix A: $|x^tAy - y^tAx|$, and for the symmetric Gauss-Seidel preconditioner M: $|x^tMy - y^tMx|$. Also included is a test for fast convergence of the CG algorithm on a modified matrix A that is close to being diagonal.

Future Work Directions

Two major undertakings are currently being considered for short-term inclusion in the code base. The first is an optimized version of the symmetric Gauss-Seidel preconditioner based on a generic multicoloring algorithm. The second, further down the line, is support for unassembled matrix data in order to aid in better mapping to performance characteristics of science applications.