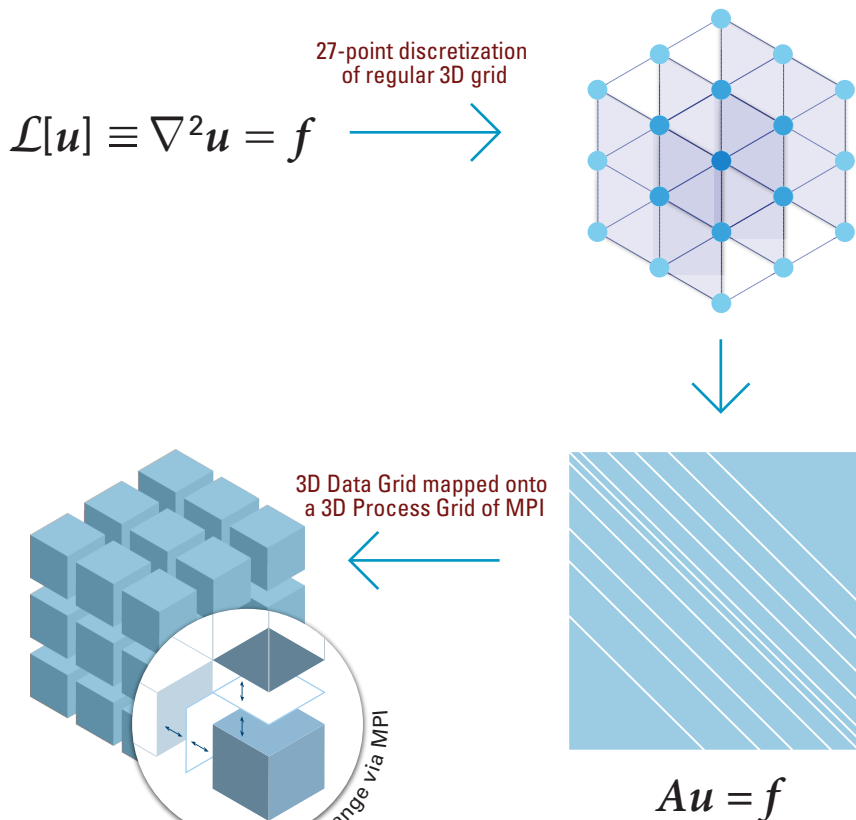


HPCGG

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 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.



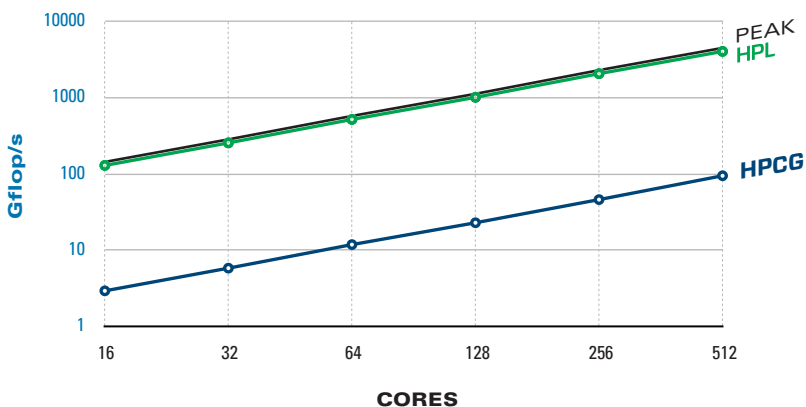
PRECONDITIONED CONJUGATE GRADIENT SOLVER

```

p0 ← x0, r0 ← b - Ap0
for i = 1, 2, to max_iterations do
  zi ← M-1ri-1
  if i = 1 then
    pi ← zi
    αi ← dot_prod(ri-1, zi)
  else
    αi ← dot_prod(ri-1, zi)
    βi ← αi/αi-1
    pi ← βipii-1 + zi
  end if
  αi ← dot_prod(ri-1, zi)/dot_prod(pi, Api)
  xi+1 ← xi + αipi
  ri ← ri-1 - αiApi
  if ||ri||2 < tolerance then
    STOP
  end if
end for
    
```

Halo Exchange via MPI

SAMPLE RESULTS



IN COLLABORATION WITH



SPONSORED BY



HPCG

We Need a New Yardstick

As a candidate for a new HPC metric, the HPC Preconditioned Conjugate Gradient Benchmark (HPCG) implements the preconditioned conjugate gradient method with a local symmetric Gauss-Seidel preconditioner.

In doing so, HPCG is designed to measure performance that is representative of many important calculations, like Type 1 patterns, with low computation-to-data-access ratios. To simulate patterns commonly found in real applications, HPCG exhibits the same irregular accesses to memory and fine-grain recursive computations that dominate so many scientific workloads.

In contrast to the new HPCG metric, the older HPL standard is a program that factors and solves a large dense system of linear equations using Gaussian Elimination with partial pivoting. The dominant calculations in this algorithm are dense matrix-matrix multiplication and related kernels, Type 2 patterns. 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 computation rates and adequate streaming memory systems.

In general, a well-rounded computer system should be designed to execute both Type 1 and Type 2 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 Type 1 and Type 2 patterns. However, HPL only stresses Type 2 patterns, and, as a metric, is incapable of measuring Type 1 patterns.

Another issue with existing performance metrics stems from the emergence of accelerators, which are extremely effective (relative to CPUs) with Type 2 patterns, but much less so with Type 1 patterns. For many users, HPL results show a skewed picture relative to Type 1 application performance, especially on machines that are heavily Type 2 biased, like a machine running accelerators for the majority of the computational power.

HPCG Can Help

The HPCG Benchmark can help alleviate many of the problems described above using the following principles:

Provides 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) in 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 real applications.

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 optimizing their own implementations.

Detects and measures variances from 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++.

The code does the following:

Problem setup: Generate 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: Set up data structures for the local symmetric Gauss-Seidel preconditioner. The reference version will use a simple compressed sparse row representation for the lower and upper triangular matrices, each as a separate matrix.

Verification and validation setup: We will compute preconditions, post-conditions, and invariants that will aid in the detection of anomalies during the iteration phases.

Iteration: We will perform 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 iteration phase.

Post-processing and reporting: We will report 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 subgrid 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 $A: [x^t A y - y^t A x]$, and for the symmetric Gauss-Seidel preconditioner $M: [x^t M y - y^t M x]$. Also included is a test for fast convergence of the CG algorithm on a modified matrix A that is close to being diagonal.