Benchmarking Tool Insights

Scalable Tools Workshop 2026

GpuPerfSuite Overview

- **New benchmark suite: gpu-perf-suite, hosted in the HPC Toolkit repository**
  - Branch: master (main is automated; MR to main pending approval)
  - Directories: cuda/, hip/, sycl/
  - Top-level make auto-detects compiler on path: hipcc, mvcc, or icpx
- **Design: Makefile invokes a script defined by $(GPUPERFTOOL)**
  - bin/hpctoolkit.sh is the reference script; takes a target argument (ROCm, CUDA, Level Zero)
  - script runs HPCToolkit workflow on benchmark_$(model)
- **Currently 16 benchmarks**
  - sized to run ~1 second per kernel
- **Claude translated original HIP code into CUDA and SYCL versions**
- **Adjust the repetition parameters?**
  - NCU detail mode ran 8 passes per kernel, taking multiple minutes each

Benchmark Suite Design Goals

- **Goals**
  - collect performance data across varied kernels
  - evaluate what tools and architectures can observe
  - apples-to-apples comparison of insights across profilers
- **Not intended as a bug-reproducer suite intended to break things**
- **What is includes**
  - thread divergence
  - occupancy
  - ILP
  - memory coalescing
- **Covers host-device communication patterns, unified memory**
- **Fast and slow variants per kernel to provide known-good and known-bad signals**
  - Some “slow” versions currently outperform “fast” ones: benchmark needs refinement
- **Longer-term goal: scale from micro-benchmarks up to mini-apps and real workloads (e.g., adaptive mesh refinement)**

Related Benchmark Suites

- **Hetero-Mark: benchmark suite for CPU-GPU collaborative computing**
  - Sun et al., IISWC 2016; covers host-device communication patterns
  - Has OpenCL and CUDA implementations; no HIP
- **RAJA Performance Suite (rajaperf): performance portability analysis with Caliper and Thicket**
  - Pearce et al., SC24; native + RAJA implementations; geometric mean speedup 1.89x achieved by grad student using HPC Toolkit data + LLM suggestions
- **Velocity Bench: HPC, AI, and data analytics workloads; SYCL, CUDA, HIP**
  - https://github.com/oneapi-src/Velocity-Bench
- **MLPerf HPC: holistic benchmark suite for scientific ML on HPC systems**
  - Farrell et al., MLHPC 2021
- **HPCPerfBench (AI workloads): power-aware performance analysis for vision and language models**
  - Mayr et al., arXiv 2026 (arXiv:2603.16164)
- **Blackjack Bench (historical): long instruction chains, varying memory access strides; available online via SVN**

Tool Measurement Gaps and Observer Effect

- **Kernel serialization is forced when PC sampling is enabled on Intel GPUs**
  - HPC Toolkit documentation warns of this; destroys concurrent execution insight
  - Goal: lightweight measurement that does not perturb the application
- **Metadata recording is weak across tools: MPI rank count, thread count, library versions not captured automatically**
  - Workaround: infer from linked library paths in measurement directory
  - Desired: metadata dashboard surfacing what was actually measured
- **Profiler overhead benchmarking suite exists (from another participant) that hashes linked libraries and categorizes runs by build configuration; proposed as complementary infrastructure**
  -

Memory Hierarchy and TLB Profiling Discussion

- **Membench-style sweep (varying data size and stride) proposed as a strong addition**
  - Reveals L1/L2/L3 cache boundaries, TLB effects, stride sensitivity
  - Used as a student assignment to empirically characterize memory hierarchy
- **TLB miss visibility is currently limited**
  - PC sampling shows stall at the instruction address but not the TLB miss cause
  - TLB counters available separately but only at kernel granularity, not per-source-line
- **PC sampling smearing problem: multiple invocations of the same kernel with different arguments (e.g., varying stride/data size) collapse into one PC sample distribution**
  - AMD external correlation IDs can attribute samples back to specific invocations
  - NVIDIA lightweight PC sampling and Intel concurrent-kernel sampling both require kernel serialization to attribute to calling context

Nsight Compute Demo and Copilot

- **16 kernels loaded into a single NCU report; kernels named by their intended issue (divergence, atomics, etc.)**
  - Copilot noted: naming kernels after their issue makes it trivially easy for AI to guess the problem
- **NCU default page shows compute vs. memory throughput per kernel; expert system flags issues automatically**
  - Example findings: threads below hardware warp limit, unequal L2 slice targeting, occupancy issues
  - Copilot (beta, online NCU only) surfaced: low occupancy, bad grid size, recommended increasing grid size and tuning block size/register usage
- **Copilot backend: NVIDIA-provided token service; requires free NVIDIA account registration**
  - AI community shifting toward bring-your-own-AI; Nsight Systems AI skills already use that model
  - Copilot not included in the CUDA toolkit bundled version; available only in the standalone online download
- **Roofline model**
  - shows where you are but not how to fix it; useful for intuition on distance from peak but limited actionability

Next Steps and Contributions

- **Add portability model implementations: RAJA and Kokkos variants alongside native CUDA/HIP/SYCL**
- **Integrate existing suites (Hetero-Mark, rajaperf, Velocity Bench, MLPerf HPC) as submodules with a master makefile**
  - Organize by capability tested (memory hierarchy, threading, host-device comms) rather than by origin suite
- **Add an ncu.sh script to bin/ for Nsight Compute runs with recommended flag sets**
- **Improve kernel implementations where fast/slow signal is currently inverted or negligible**
  - ILP benchmark: chain of 2 instructions in a loop is insufficient; needs ~500 unrolled iterations
  - Consider volatile variables to prevent compiler from eliminating intended measurement patterns
- **Rerun suite on a higher-end GPU (A100/H100) with proper video setup for a recorded demo**
- **Explore mega-kernel / kernel fusion benchmarks (relevant to LLM workloads); ask grad student for pointers**

Next Steps

- **Add ncu.sh script to bin/ with recommended Nsight Compute flagsEnables apples-to-apples profiler comparison runs alongside the existing HPCToolkit.sh.**
- **Fix fast/slow kernel signal quality across benchmarksILP benchmark needs ~500 unrolled iterations; consider volatile variables to prevent compiler elision.**
- **Add RAJA and Kokkos implementations to the suiteCompare portability-model vs. native performance using the same profiling workflow.**
- **Integrate external suites as submodules with a unified makefileOrganize by capability (memory hierarchy, threading, host-device comms) rather than by suite origin.**