Los Alamos

NATIONAL LABORATORY

EST. 1943

Los Alamos
NATIONAL LABORATORY
EST. 1943

Delivering science and technology
to protect our nation
and promote world stability

Cluster Benchmarking

Presented by CSCNSI

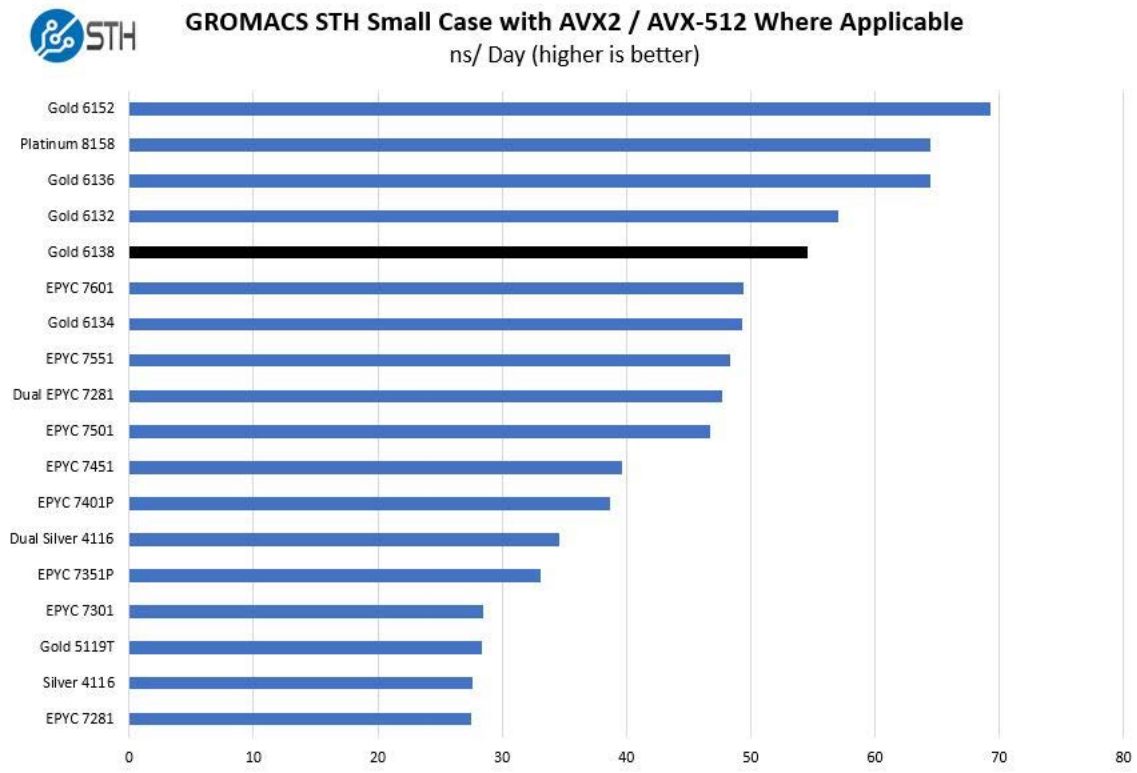
Why Benchmark?

Benchmarking is the processes of collecting quantifiable data on system performance

- Benchmarks can help us:
  - Evaluate comparative system performance
  - Identify bottlenecks for application performance
  - Estimate performance of an application on a given system
- Benchmarks are often a required as part of the “acceptance” process
  - Newly purchased systems are often contracted to reach a certain SSI
    - SSI = “Scalable System Improvement”, i.e. how much faster is this system than the old one?
- Benchmarks can sometimes be an early warning sign that something is wrong with a system, either in design or in function

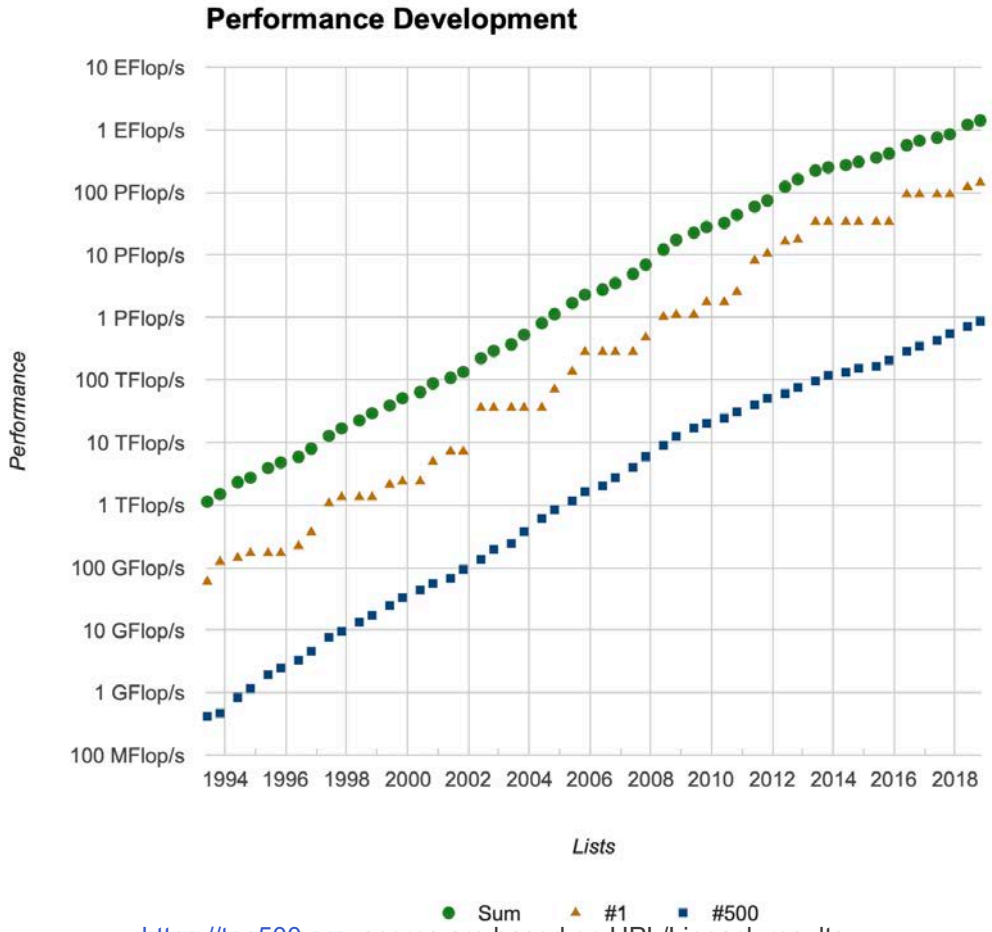
“Real” benchmarks

- The best way to know how a system will perform with an application is to run the application!
- Often, the data we collect with a real benchmark is simply the time it takes to run an application
- There can be problems with real benchmarks:
  - We may not have tuned or ported our application for the new platform
  - Results with one specifically build or data set may differ from another
  - We may not have the software available
  - They don’t always tell us why an application performance better or worse



Synthetic Benchmarks

- Synthetic benchmarks tend to focus on a class of algorithm, or particular hardware function
- Synthetic benchmarks can help understand which components perform better and worse
- Synthetic benchmarks have problems too:
  - It can be hard to predict how a specific complex application will perform
  - Systems can be tuned for synthetic benchmarks and ignore performance that really matters



https://top500.org: scores are based on HPL/Linpack results

Hybrid benchmarking

- Generally we want a combination of real and synthetic benchmarks
- Using a hybrid approach can give us:
  - High confidence that specific applications will perform
  - Understanding of which components help/hurt performance

Synthetic “micro” benchmarking

- Test a specific component or metric
- Examples:
  - Test network fabric latency/bandwidth
  - Test memory bandwidth
  - Test CPU vector operations
  - Test disk seek time
  - GPU performance
  - …

- Common Tools:
  - Netperf (TCP networks point-to-point)
  - ib_{send/recv}_{bw/lat} (Infiniband point-to-point)
  - Bonnie++ (filesystem performance)
  - Stream (memory performance)
  - Intel MPI Benchmarks (“IMB”, MPI operations bencharking)

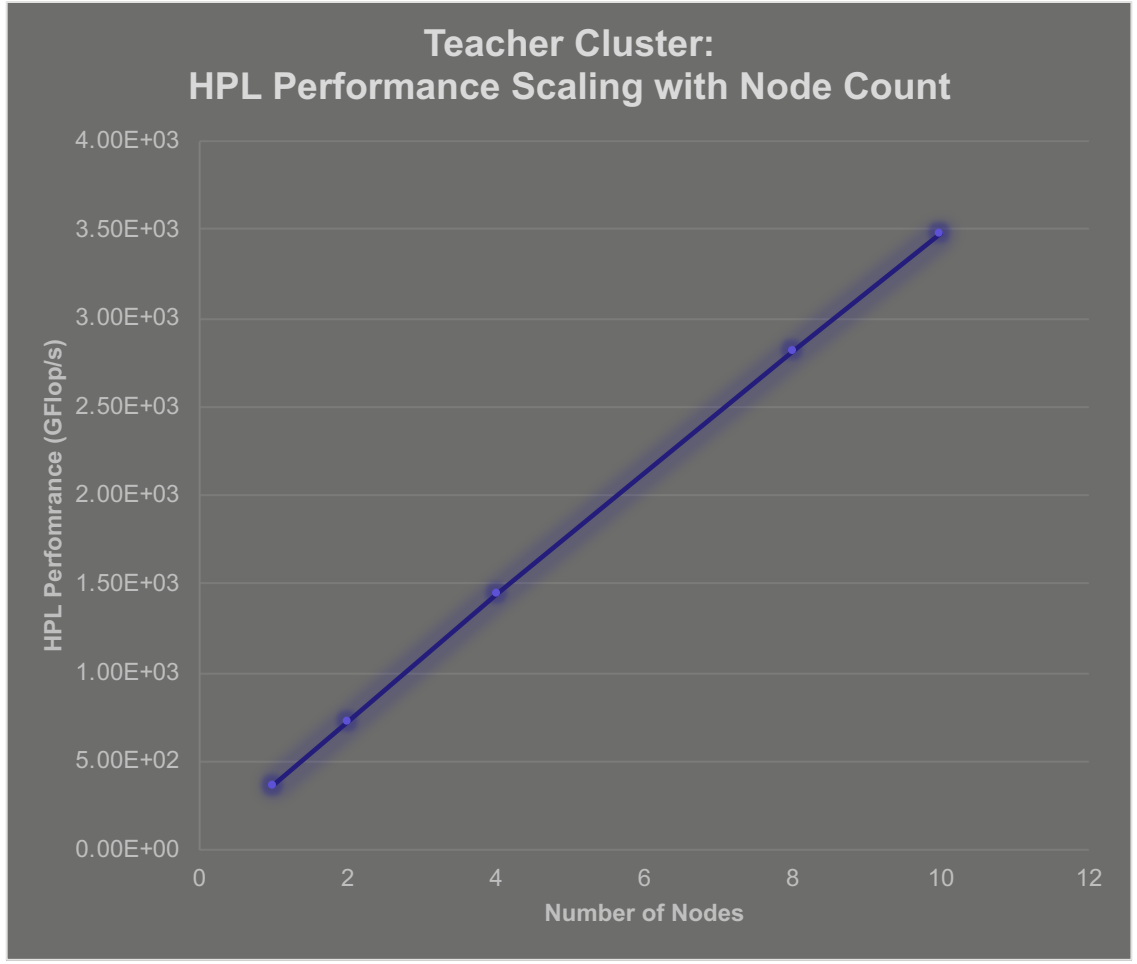
Synthetic “macro” benchmarking

- Macro benchmarks tend to try to solve a typical *kind* of problem that can tell about overall performance
- Often solving mathematical problems:
  - Linear algebra
  - Differential Equations
  - Graph theory
- High scores on these when you places on the big lists like:
  - Top500.org
  - Green500.org
  - Graph500.org
  - …

- Common Tools:
  - HPL (“High Performance Linpack”, distributed matrix operations, standard for top500.org)
  - HPCG (“High Performance Conjugate Gradients”, tests various linear algebra applications)
  - DGEMM (dense matrix, low communications overhead)
  - SMG2000 (Linear PDE solver)
  - …

Benchmarking scalability

- A common technique in both real and synthetic (mostly macro) benchmarking
- Focuses on how linear the quantity scales while increasing parallelism
- Is often critical to knowing how big to build a system
- Often coupled with profiling tools (when feasible) to understand why scaling breaks down



Some things to keep in mind…

- You probably want to build benchmarks for the hardware you're on
- Understand what the tool does, and how to tune it
- Keep your results well organized
- Try building against different libraries/tools:
  - Linear algebra: OpenBLAS, ATLAS…
  - Compilers: GCC, Intel, PGI…
  - MPI: OpenMPI, IntelMPI, MPICH…

Questions?