How to Buy a Supercomputer for Scientific Computing

Compute Ontario Colloquium

James Willis (SciNet)

April 17, 2024

Outline

- Motivation
- Performance Metrics
- SSI Metric
- Benchmark Applications
- File System Performance
- Summary

Motivation

- Choosing a new HPC system that suits the needs of our users is a very difficult decision
- How do we choose the hardware? AMD? Intel? Nvidia? DDN? VAST?
- What filesystem shall we use? What should the total storage be? 20PB? Minimum bandwidth performance?
- Do we want all CPU nodes? All GPU nodes? A combination of CPU + GPU nodes? What ratio then?
- How many total nodes?
- What interconnect should we use to connect the nodes? Infiniband? Slingshot? Dragonfly+?
- Will it fit our power budget?
- Will it fit in our current datacentre?
- How should we cool it? Fans? Liquid cooled? Direct? Immersion?

Motivation

- What is the budget?
- How much does it cost to run? Power usage?
- How do we balance all of the above constraints and purchase the best possible system for our users?
- How do we compare new potential systems to decide the winner? What performance metric do we use?

Performance Metric

- We need a way to quantify how well a system performs
- A single performance metric that evaluates all aspects of the system which can be used to rank new potential systems
- Performance metrics exist that measure raw throughput
- High Performance Linpack (**HPL**) for example:
  - Solves a (random) dense linear system in double precision (64 bits)
  - Measures total FLOP/s (**FL**oating point **OP**erations per second)
  - Used to rank HPC systems on the **Top500** list
- This metric is good to measure *raw* performance
- *However*, it is not representative of the vast majority of workloads run on our system

Performance Metric

- Another issue we face is that systems are becoming more heterogeneous when it comes to node types
- Every application will not necessarily run on each node type and if they do they might not perform well
- For example, on Niagara (CPU) and Mist (GPU) we have 2024 **CPU** nodes and 54 **GPU** nodes
- Therefore we need a metric that will:
  - Represent common workloads that will run on the system
  - Combine the performance of each node type that makes up the system

SSI Performance Metric

- The National Energy Research Scientific Computing center (**NERSC**) has developed such a metric
- Called the Scalable System Improvement (**SSI**) metric:

$$\boxed{SSI = \langle w_i U_i S_i \rangle}$$

- Where:
  - U is known as the *utilisation factor*
  - S is the *speedup factor*
  - w is the benchmark weight
  - i refers to each benchmark application
- SSI is computed as a mean over all benchmarks
- Lets go through each term in the equation

Utilisation Factor

- U_i is defined as:

$$U_i = \frac{N/n_i}{N^{ref}/n_i^{ref}}$$

- Where:
  - N is the total no. of nodes in the system
  - n is the total no. of nodes used in the benchmark
  - i refers to each benchmark application
  - ref refers to a reference machine, e.g. Niagara
- The machine *utilisation* term was introduced to reward application throughput and system size relative to a reference system
- In other words if the proposed system can run the benchmark on a smaller fraction of the total system compared to the reference it will score higher

Speedup Factor

- S_i is defined as:

$$S_i = \frac{t_i^{ref}}{t_i}$$

- Where:
  - t is the runtime of the benchmark (or other application-specific figure-of-merit such as time-per-iteration) e.g. s, FLOP/s, ns/day
  - i refers to each benchmark application
  - ref refers to a reference machine, e.g. Niagara
- This term is simpler, it's just the speedup of the benchmark running on the proposed system compared to the reference system, i.e. a newer system improves the time-to-solution
- This constraint is necessary because, for example, U could be maximised by running the benchmarks on the fewest number of nodes to avoid scaling inefficiencies
- This could lead to a system that sacrifices network performance for increased memory per node

Weights

- Each benchmark can be prioritised by setting optimal values for the weights, where $0 < w_i \leqslant 1$
- Say a benchmark represents a particular workload that takes up a large proportion of the jobs run on the system
- w_i can be set accordingly to increase the reliance of the SSI score on that benchmark

Mean Calculation

- We also have to decide on what type of mean calculation we use
- We have a few to choose from:
  - Arithmetic mean, to be used when performance metrics are in units of time e.g. s
  - Harmonic mean, to be used when performance metrics are expressed as a rate e.g. FLOP/s
  - **Geometric mean, is best used when normalising against a specific baseline**
- Geometric mean was chosen because we are normalising against Niagara as a baseline
- Calculation:

$$\left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}} = \sqrt[n]{x_1 x_2 \ldots x_n}$$

- Where n in our case is the total number of benchmarks to run

Final Form

- For Niagara we equally weighted all benchmarks, i.e. set all $w_i = 1$
- This gives:

$$\boxed{SSI = \langle U_i S_i \rangle = \left\langle \frac{N/n_i}{N^{ref}/n_i^{ref}} \frac{t_i^{ref}}{t_i} \right\rangle}$$

Benchmark Applications

- To get a good representative SSI score for our system we need to choose a set of benchmarks carefully
- We can do this by looking at the current workloads running on Niagara and Mist
- The top workloads on Niagara are:
  - Matrix/Linear Algebra
  - Computational Fluid Dynamics (CFD)
  - Quantum Chemistry (Density Functional Theory - DFT)
  - Molecular Dynamics
  - Atmospheric Research (Weather Forecasting)
- The top workloads on Mist are:
  - Molecular Dynamics
  - Machine Learning

Benchmark Applications

- Based on this list we chose the following benchmark applications:
  1. HPCG - **H**igh **P**erformance **C**onjugate **G**radients
  2. NekRS
  3. Quantum ESPRESSO - Op**E**n-**S**ource **P**ackage for **R**esearch in **E**lectronic **S**tructure, **S**imulation, and **O**ptimization
  4. NAMD - **NA**noscale **M**olecular **D**ynamics
  5. WRF - **W**eather **R**esearch and **F**orecasting Model
  6. SPEChpc 2021 - **S**tandard **P**erformance **E**valuation **C**orporation HPC
  7. GROMACS (GPU) - **GRO**ningen **MA**chine for **C**hemical **S**imulations
  8. MLPerf (GPU) - **M**achine **L**earning **P**erformance Benchmark
- You can find the full list of benchmarks with instructions here: https://github.com/SciNetHPC/RFP-Benchmark-Suite
- Lets take a brief look at each application

HPCG

- The High Performance Conjugate Gradients benchmark is an open source project developed by Mike Heroux, Jack Dongarra and Piotr Luszcze
- Source: https://www.hpcg-benchmark.org
- It was designed to exercise computational and data access patterns that closely match a broad set of important scientific applications
- Features:
  - Sparse matrix-vector multiplication
  - Vector updates
  - Global dot products
  - Local symmetric Gauss-Seidel smoother
  - Sparse triangular solve (as part of the GS smoother)
  - Driven by multigrid preconditioned conjugate gradient algorithm that exercises the key kernels on a nested set of coarse grids
  - Written in C++ with MPI and OpenMP support

NekRS

- Open source, fast and highly scalable computational fluid dynamics (CFD) solver targeting HPC applications
- Developed at Argonne National Laboratory
- Source: https://github.com/Nek5000/nekRS

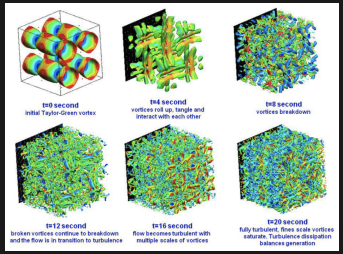


Figure 1: Taylor-Green Vortex simulated with NekRS

- Features:
  - Incompressible and low Mach-number Navier-Stokes + scalar transport
  - High-order curvilinear conformal spectral elements in space
  - Variable time step 2nd/3rd order semi-implicit time integration
  - MPI + OCCA (backends: CUDA, HIP, OPENCL, SERIAL/C++)
  - LES and RANS turbulence models
  - Arbitrary-Lagrangian-Eulerian moving mesh
  - Lagrangian phase model

Quantum ESPRESSO

- Integrated suite of open source computer codes for electronic-structure calculations and materials modeling at the nanoscale
- Based on density-functional theory (DFT), plane waves, and pseudopotentials
- Source: https://www.quantum-espresso.org

- Features:
  - Written in Fortran with MPI + OpenMP, GPU version available
  - PWscf: structural optimisation and molecular dynamics on the electronic ground state, with self-consistent solution of DFT equations
  - CP: Car-Parrinello molecular dynamics
  - PHonon: vibrational and dielectric properties from DFPT (Density-Functional Perturbation Theory)
  - TD-DFPT: spectra from Time-dependent DFPT
  - HP: calculation of Hubbard parameters from DFPT
  - EPW: calculation of electron-phonon coefficients, carrier transport, phonon-limited superconductivity and phonon-assisted optical processes
  - PWCOND: ballistic transport

NAMD

- Parallel molecular dynamics code designed for high-performance simulation of large biomolecular systems
- Highly scalable to 10^5 cores
- Developed at the University of Illinois
- Source: www.ks.uiuc.edu/Research/namd

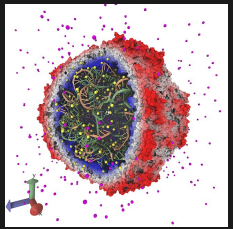


Figure 2: Satellite Tobacco Mosaic Virus

- Features:
  - Written using the Charm++ parallel programming model
  - Support for GPUs
  - Constant temperature via rescaling, coupling, or Langevin dynamics
  - Constant pressure via Berendsen or Langevin Nose-Hoover methods
  - Particle mesh Ewald full electrostatics for periodic systems

WRF

- Open source Weather Research and Forecasting (WRF) Model
- Next-generation mesoscale numerical weather prediction system designed for both atmospheric research and operational forecasting needs
- Source: https://www.mmm.ucar.edu/models/wrf
- Features:
  - Written in Fortran with MPI + OpenMP
  - Meteorological studies
  - Real-time NWP (Numerical Weather Prediction)
  - Idealized simulations
  - Data assimilation
  - Earth system model coupling
  - Model training and educational support

SPEChpc

- The Standard Performance Evaluation Corporation SPEC maintains a standardized set of benchmarks and tools to evaluate performance for the newest generation of computing systems
- They offer well-selected science and engineering codes that are representative of HPC workloads and are portable across CPU and accelerators, along with certain fair comparative performance metrics
- Source: https://www.spec.org/hpc2021
- SPEChpc 2021 Large Workload Benchmark:
  - D2Q37 is a Computational Fluid Dynamics code for simulating 2D-fluids using the Lattice Boltzmann Method (LBM)
  - TeaLeaf is a mini-app that solves the linear heat conduction equation on a spatially decomposed regular grid using a 5 point stencil with implicit solvers
  - CloverLeaf is a mini-app that solves the compressible Euler equations on a Cartesian grid, using an explicit, second-order accurate method
  - POT3D computes potential field solutions used to approximate the 3D solar coronal magnetic field using observed photospheric magnetic fields as a boundary condition
  - High Performance Geometric Multigrid (HPGMG-FV) is a benchmark designed to proxy the finite-volume based geometric multigrid linear solvers found in adaptive mesh refinement (AMR) codes
  - The miniWeather code mimics the basic dynamics seen in atmospheric weather and climate

GROMACS

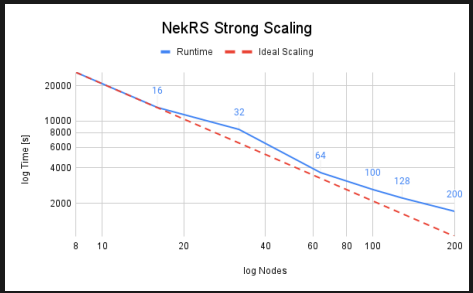
- GROningen MAchine for Chemical Simulations is a versatile open source package used primarily for molecular dynamics simulations of biomolecules, such as proteins, lipids, and nucleic acids
- It is designed to perform simulations of large biomolecular systems with high efficiency on CPUs, GPUs, and specialized hardware
- Source: https://www.gromacs.org/
- Features:
  - Written in C++ with MPI + OpenMP
  - Support for GPUs
  - Energy minimization
  - Molecular dynamics simulations
  - Free energy calculations
  - Analysis tools for studying the dynamics and properties of biomolecules at the atomic level
  - Widely used in various fields such as biochemistry, biophysics, pharmaceutical research, and materials science

MLPerf

- MLPerf is an industry-standard benchmark suite for measuring the performance of machine learning tasks
- It is developed by MLCommons, a consortium of AI leaders from academia, research labs, and industry
- Source: https://mlcommons.org/

Benchmark Problem Sizes

- We made sure to choose problem sizes that were sufficiently large so that:
  1. They occupied a significant proportion of the RAM available on each node to test memory bandwidth
  2. The problem could be scaled up to high core counts
  3. There is enough work for sustained computation and we are not network bound
- To do this we performed strong-scaling tests to confirm scability

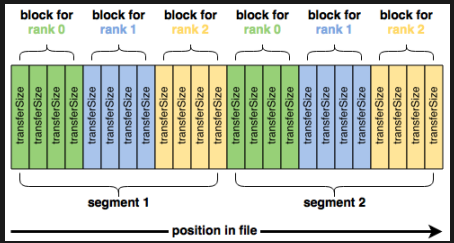


File System Performance

- So far we have tested the system for memory, compute and network performance
- But if we can't get data to/from the compute nodes in a timely manner this is irrelevant
- The other major component that we need to consider is the *file system*
- We measure this with by looking at bandwidth, i.e. GB/s and IOPS performance, i.e. **I**nput/Output **O**perations **P**er **S**econd
- The IOR benchmark comes in handy here

IOR Benchmark

- **I**nterleaved **O**r **R**andom is an open source benchmark for testing the performance of parallel storage systems
- It uses various interfaces and access patterns to excercise the file system by reading/writing files in parallel
- List of backends:
  - POSIX (default)
  - MPIIO
  - HDF5
  - HDFS
  - S3



- Features:
  - Uses a common parallel IO abstraction backend and MPI for process synchronisation
  - Can modify block size, files per process, transfer size, segment size, memory allocated
  - Can modify file access patterns to mimic application workloads, e,g. file offsets, random acces

Summary

- We looked at the considerations that go into to buying a new supercomputer
- We described the SSI metric for measuring supercomputer performance
- Listed our benchmark suite for the refresh of Niagara
- Also looked at measuring file system performance
- Please email any questions to: **support@scinet.utoronto.ca**

References

- SSI: https://sc18.supercomputing.org/proceedings/workshops/workshop_files/ws_pmbsf117s2-file1.pdf
- NekRS: https://www.researchgate.net/publication/335147096_ECP_Milestone_Report_Public_release_of_CEED_20
- NAMD: https://en.wikipedia.org/wiki/Tachyon_(software)
- IOR: https://ior.readthedocs.io/en/latest/userDoc/tutorial.html#first-steps