Benchmarking

Benchmarking

- High-Performance Computing (HPC) benchmarking tools are used to assess and measure the performance of HPC systems, including supercomputers and high-end clusters.
  These tools provide standardized metrics and tests to evaluate various aspects of the system, such as compute power, memory performance, interconnect efficiency, and storage capabilities.

- The results obtained from benchmarking can help compare different systems, identify performance bottlenecks, optimize configurations, and gauge system improvements over time.

Benchmarking

- High-Performance Computing (HPC) benchmarking tools are used to assess and measure the performance of HPC systems, including supercomputers and high-end clusters.
  These tools provide standardized metrics and tests to evaluate various aspects of the system, such as compute power, memory performance, interconnect efficiency, and storage capabilities.

- The results obtained from benchmarking can help compare different systems, identify performance bottlenecks, optimize configurations, and gauge system improvements over time.

- HPC benchmarking tools can be broadly categorized into two types:
  a) Micro Benchmarks b) Macro Benchmarks

Benchmarking

- **Micro Benchmarks:**
  Micro benchmarks focus on measuring the performance of specific system components or low-level operations. These benchmarks are designed to isolate and evaluate individual aspects of the HPC system, such as CPU, memory, disk I/O, network interconnects, and parallel file systems. They provide insights into the raw capabilities of these components and help identify any weaknesses or areas for improvement.

- Some commonly used micro benchmarking tools include:
  HPL, STEAM, IOR, OSU, Iperf, NTTTCP, LIKWID

Benchmarking

- **Macro Benchmarks:**
  Macro benchmarks, also known as application benchmarks or real-world benchmarks, focus on evaluating the performance of complete applications or workloads that are representative of real-world usage scenarios. These benchmarks typically simulate complex scientific, engineering, or data-intensive tasks that are common in HPC environments. This enables a more accurate assessment of real-world performance and helps in making informed decisions about system configurations or optimizations.

- Some commonly used micro benchmarking tools include:
  SPEC MPI, HPCG, NAMD, GROMACS, WRF, HPCC, PTRANS

LIKWID

- **LIKWID (Like I Knew What I'm Doing)** is a performance analysis tool for parallel applications and hardware architectures. It provides a set of command-line tools and libraries that assist in analyzing the performance of applications running on modern processors.
  LIKWID focuses primarily on x86-based architectures, including CPUs from Intel and AMD.

- Here are some key features and components of LIKWID:
  Performance monitoring counters, Event based sampling, Topology discovery, Resource allocation, Access modes, Energy monitoring, Output formats and Visualization.

  LIKWID is often used by developers, researchers, and system administrators who need to understand the performance characteristics of their applications.

LIKWID

- **Install LIKWID:**

  Download and install the LIKWID package from the official LIKWID GitHub repository (https://github.com/RRZE-HPC/likwid).
  Follow the installation instructions provided in the repository's README file.

- Check for details on the following tools:

  Hardware Topology
  Checking Events
  Profiling Application and Analyzing Performance Data
  Energy Monitoring

LIKWID

- **Hardware Topology:**
  Use the **likwid-topology** tool to discover and explore the hardware topology of your system. It provides information about the available sockets, cores, caches, and threads.

  Run the following command:

```
$) likwid-topology
```

LIKWID

- **Check Available Events:**
  Use the **likwid-perfctr** tool to check the available performance events supported by your processor.

  Run the following command:

```
$) likwid-perfctr  -a
```

LIKWID

- **Profile Application:**
  Use the **likwid-perfctr** tool to profile your application and collect performance data. Specify the events you want to monitor and the command or application you want to profile.

  For example, to profile a program called ‘myprogram’, run the following command:

```
$) likwid-perfctr  -g  <events>  -m  ./myprogram
```

LIKWID

- **Analyze Performance Data:**
  After running your application with LIKWID, you will have performance data to analyze. LIKWID provides various tools for analyzing the collected data.

  Some of the commonly used tools include:
  - **likwid-perfctr**: Generates a summary report of the collected performance data.
  - **likwid-plot**: Visualizes the collected data in various plots and graphs.
  - **likwid-pin**: Binds your application to specific cores or memory controllers for analyzing performance in different execution scenarios.

LIKWID

- **Energy Monitoring:**
  For energy monitoring, use the **likwid-powermeter** tool to measure the power consumption of your application.

  Run the following command:

  ```
  $) likwid-powermeter  -C  <cores>  -g  <granularity>  -s  <duration>
  ```

Workout 1

A) Simply run the commands listed above and check the data that is being made available.

B) Profile LAMMPS application using LIKWID tool. Check if you are able to gain any insights.

C) Experiment with the available tools for processor / core specific action for profiling the application.

Execute the listed commands for LIKWID tools.

1) Run LIKWID's likwid-perfctr tool to monitor core '0' for event FLOPS_SP and execute likwid-plot to plot graphs based on collected data
2) Execute likwid-powermeter, and parallely execute LAMMPS or WRF application. Note the energy consumption during the same rungs.
3) Execute an OpenMP application using likwid-pin.
4) Benchmark your system using likwid-bench tool.

HPL Benchmark

- The **HPL (High-Performance Linpack) benchmark** is a widely used benchmark for measuring the floating-point performance of computer systems, particularly supercomputers and high-performance computing (HPC) clusters.
  It solves a dense system of linear equations using the LINPACK library and is designed to stress the computational capabilities of the system.

- The benchmark solves a dense linear system of equations represented as **Ax = b**, where A is a large square matrix, x is the solution vector, and b is the right-hand side vector.
  The LU factorization with partial pivoting decomposes the matrix A into the product of a lower triangular matrix (L), an upper triangular matrix (U), and a permutation matrix (P).
  The benchmark performs repeated iterations of the LU factorization to obtain accurate results.

HPL Benchmark

- The performance metric of the HPL benchmark is typically measured in terms of the number of **floating-point operations per second** (FLOPS).
  It provides a measure of the system's computational power, specifically its ability to perform matrix computations efficiently.

- The HPL benchmark is often used to assess the scalability of parallel computing systems.
  It measures the performance of the system as the problem size and the number of processing elements (e.g., CPU cores) are varied.

HPL Benchmark

- The HPL benchmark is typically implemented using the MPI standard for distributed memory parallelism.
  It relies on highly optimized numerical linear algebra libraries, such as BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra Package), to perform the matrix computations efficiently.

- The HPL benchmark plays a significant role in the **TOP500** list, which ranks the world's most powerful supercomputers. The list is based on the performance results obtained from running the HPL benchmark on each system.

HPL Benchmark

- The HPL benchmark uses a file called "**HPL.dat**" as its input file.
  The "HPL.dat" file contains the parameters and configuration settings for running the benchmark.
  It specifies various parameters such as the matrix size, block size, distribution of processes, and other tuning parameters that define the problem size and execution environment.

- Most important parameters:
  Problem Size(N)
  Block Size(NB)
  Process Grid Dimension(P x Q)

HPL Benchmark

- Problem Size (N):
  This parameter determines the size of the square matrix in the linear system of equations. It represents the number of rows or columns in the matrix.
- Block Size (NB):
  The block size determines how the matrix is divided into smaller blocks for computation. It affects the memory access pattern and can influence the performance of the benchmark.
- Process Grid Dimension(P x Q):
  The process grid dimensions determine how the processes are distributed across nodes or compute resources. The dimensions P and Q represent the number of rows and columns in the process grid, respectively.

More Details: https://icl.utk.edu/hpl/faq/index.html#286

Workout 2

A) Compile and Install HPL benchmark.

B) Tune the HPL.dat file based on the system that you are using.

C) Execute the benchmark and observe the performance value.

D) Compare the value to the TOP500 website list.

Use the HPL benchmark to benchmark your system.

1) Manually download and install HPL.
2) Observe the Makefile and make changes as necessary.
3) Set correct paths for MPI and Libraries.
4) Check the parameters necessary for HPL.dat file.
5) Understand how the values are computed.
6) Execute the benchmark.

Workout 3

A) Compile and Install HPCC benchmark.

B) Tune the HPL.dat file based on the system that you are using.

C) Execute the benchmark and observe the performance value.

D) Compare the value to the TOP500 website list.

Compile and execute HPCC (HPC Challenge) benchmarks

“Benchmarking is like stepping on a scale after a week of eating pizza. You may not like the numbers, but at least you know where you stand... and how much pizza you've enjoyed!”

Questions?

Thanks!

Contact:

Vineet More
vineet.more.bfab@gmail.com
[Make sure to use HPCAP23 as the subject line for your queries]

LinkedIn Handle:
https://www.linkedin.com/in/vineet-more-c-programmer/