Node-Level Performance Analysis with LIKWID

Learning Objectives

The objectives of this tutorial are:

- To learn how to explore machine configurations for performance optimizations using LIKWID
- To learn how to use LIKWID to measure performance limitations of common computer architectures

Tools

- LIKWID

Contents

In this tutorial we use LIKWID Profiling toolsuite to evaluate the performance of specific computational kernel's. Using the Roofline model, identify bottlenecks and computation and bandwidth boundedness of the kernels. Suggest Optimization strategy based on the Emperical Roofline Model.
NOTE: Tasks for this tutorial should be performed in compute nodes.

Using LIKWID Topolgy 1: Tutorial (10 min)

Use LIKWID to determine the NUMA topology of a compute node.

Steps

1. Start an interactive Slurm session for a single node
2. Load LIKWID module
3. Run `likwid-topology` from the command line.
4. Determine the number of threads, cores per socket, and sockets
5. Determine cache sizes, cache types, cache associativity and cache line sizes

Hints

- Use the `-c` option of `likwid-topology`.

Using LIKWID Process Affinity Tool 2: Tutorial (20 min)

- Determine the thread domains in the selected compute node.
- Pin the given program to the hardware threads 0,4,5 and 6
- Pin the given program to 8 cores numbered randomly
- Pin the given program to 2 MPI processes with 2 threads per process.

Steps

1. Compile the given source code, 'pchpc_tut_likwid.c'
2. Run the program with `likwid-pin` providing a comma separated list for the `-c` option.
3. For MPI process pinning, wrap `likwid-pin` with `srun`.

Hints

- Use the `-p` option of `likwid-pin`.
- Use the `-h` option to determine possibilities of providing threads to processor list.

Using LIKWID Performance Counters Tool 3: Tutorial (20 min)

- Determine the list of performance metrics or groups supported by LIKWID
- List the hardware events or counters available in the selected compute node, and show the events or counters used to calculate a selected performance group.
- Find the formula used by LIKWID to derive the DRAM bandwidth metric.
- Measure the L2 and L3 cache miss ratios for 4 selected cores or processors.
- (Optional) Perform task 4 using 2 MPI processes and 2 threads.

Steps

1. Compile the given source code, 'pchpc_tut_likwid.c'
2. Run the program with `likwid-perfctr` providing necessary options

Hints

- Use `likwid-perfctr -a` to list the available performance metrics.
- Use the `-C` to pin threads.

Using LIKWID MPI Wrapper 4: Tutorial (20 min)

Select and measure four performance metrics for the given program using `likwid-mpirun`.

Steps

1. Compile the given source code, 'pchpc_tut_likwid.c'
2. Run the program with `likwid-mpi` providing necessary options

Hints

- Use `likwid-mpi -h` to list the available options.

(Optional) Emperical Roofline Model 5: Tutorial (20 min)

Use LIKWID to measure the performance of the given program. Is the program computation or communication bound? Can you identify any bottleneck? What are the possible optimization strategy?

Steps

1. Compile the given source code, 'dmvm.c'
2. Run the program with LIKWID and measure the necessary metrics for the Roofline model.
3. Compare measured performance to vendor's specifications.
4. Measure performance metrics for specific regions in the given source code using LIKWID Marker API.

Hints

- Use `likwid-perfctr` and/or `likwid-mpirun`
- Use the LIKWID predefined performance groups, e.g.
  - $FLOPS\_DP$ and $FLOPS\_SP$ for $FLOPs$
  - HDM_CACHE for MCDRAM data movement in KNL
  - $L2$ for L2 cache, and
  - $DATA$ for L1 cache, etc
- Read about LIKWID Marker API in the LIKWID documentation.

Further Reading

- LIKWID Performance Tools - https://hpc.fau.de/research/tools/likwid/
- Samuel W. Williams (2008) Auto-tuning Performance on Multicore Computers, *University of California at Berkeley, Technical Report No. UCB/EECS-2008-16*