LIKWID 5.0 | quick reference
Erlangen Regional Computing Center (RRZE)

Introduction

LIKWID is a simple to install and simple to use tool suite of command line applications for performance-oriented programming. It currently works for Intel, AMD, ARM and POWER processors on the Linux OS.

likwid-topology Print the node topology, including cache information, NUMA structure, and the mapping of hardware threads to resources

likwid-pin Pin threaded applications (POSIX threads and all threading models built on pthreads, such as Intel and GCC OpenMP) to dedicated processors

likwid-mpirun Wrapper for starting MPI/Hybrid MPI/OpenMP applications with likwid-perfctr integration

likwid-perfctr Count hardware performance events, including energy, in wrapper, timeline, or stethoscope mode; works with marker API to restrict counting to code regions; includes likwid-pin functionality

likwid-perfscope Frontend to the timeline mode of likwid-perfctr, plots live graphs of performance metrics using gnuplot

likwid-powermeter Read out RAPL Energy information and get info about Turbo Mode steps; can be used for end-to-end energy measurements

likwid-bench Microbenchmarking platform; allows easy design of multi-threaded assembly language benchmarking loops with full affinity control

likwid-memsweeper Sweep memory of NUMA domains and evict cache lines from the last level cache

likwid-setFrequencies Control the CPU core and Uncore frequencies, set the scaling governor

likwid-genTopoCfg Dump topology information to a file

Download, Build and Install

You can get the releases of LIKWID at:
http://ftp.fau.de/likwid/ or
https://github.com/RRZE-HPC/likwid/releases
For build and installation hints see the INSTALL file or the build instructions in the Wiki:
https://github.com/RRZE-HPC/likwid/wiki/Build

Contact

If you have any questions about LIKWID, please open a topic at https://groups.google.com/forum/#!forum/likwid-users.
If you think you found a bug, please open an issue with as much information as possible: https://github.com/RRZE-HPC/likwid/issues.

Generic options (all tools)

`-h, --help`	Help message
`-v, --version`	Version information

likwid-topology

Syntax:	`likwid-topology [options]`
`-V, --verbose <level>`	Set verbosity
`-c, --caches`	List cache information
`-C, --clock`	Measure processor clock
`-O`	CSV output
`-o, --output <file>`	Store output to file
`-g`	Graphical output (ASCII art)

likwid-pin

Syntax:	`likwid-pin [options] your_binary [args]`
`-V, --verbose <level>`	Verbose output
`-i`	Set NUMA interleave policy across domains selected by `-c`
`-m`	Set NUMA membind policy across domains selected by `-c`
`-S, --sweep`	Sweep memory & LLC of involved NUMA nodes
`-c, -C <list>`	Specify core ID list
`-s, --skip <hex>`	Bitmask with threads to skip
`-p`	Print available domains with mapping on physical IDs
`-d <string>`	Delimiter in physical processor list
`-q, --quiet`	Silent without output
Example: physical numbering (as in likwid-topology)	
`-c 7,4,12-14`	Cores 7, 4, 12, 13, and 14
Examples: logical numbering (physical cores first)	
`-c S1:0-3`	First four physical cores on socket 1
`-c M0:0-3@M1:0-3`	First four physical cores each on NUMA domains 0 and 1
`-c M:scatter`	Scattered binding, physical cores first, across all NUMA domains
Examples: expression syntax (compact numbering)	
`-c E:M0:24`	First 24 SMT threads in NUMA domain 0
`-c E:N:120:2:4`	Pin 120 threads in chunks of 2 with stride 4 in whole node

likwid-memsweeper

Syntax:	`likwid-memsweeper [options]`
`-c <list>`	Specify NUMA domain ID(s) to sweep (default: all)

likwid-setFrequencies

Syntax:	`likwid-setFrequencies [options]`
`-c <dom>`	Domain to apply settings to (default all)
`-g <gov>`	Set governor (conservative, ondemand, powersave, performance, turbo)
`-f, --freq <f>`	Set fixed core frequency (min/cur/max), implicitly sets userspace governor
`-t, --turbo <0|1>`	(De-)activate turbo mode
`-x, --min <f>`	Set min core frequency
`-y, --max <f>`	Set max core frequency
`--umin <f>`	Set min Uncore frequency
`--umax <f>`	Set max Uncore frequency
`-p`	Print current frequencies
`-l`	List available frequencies
`-m`	List available governors

likwid-bench

Syntax:	`likwid-bench [options]`
`-a`	List all available benchmark kernels
`-d`	Delimiter used for physical core list
`-p`	List available thread domains
`-s <TIME>`	Minimum time to run the test [sec]
`-i <ITERS>`	Specify the number of iterations per thread manually.
`-l <TEST>`	List properties of benchmark
`-t <TEST>`	Type of test

`-w <GROUP>`	Specify thread group:
	`<dom>:<size>[:<nThreads>[:<chunk>:<stride>]]`
	`[-<streamId>:<dom_id>[:<offset>]]`
	`<size>` in kB, MB or GB

Example: STREAM Triad, AVX w/FMA, 4 cores in socket 0	
`likwid-bench -t stream_avx_fma -w S0:100MB:4:1:2`	
Example: load-only, AVX-512, 64 cores, 2 threads/core	
`likwid-bench -t load_avx512 -w N:28MB:64:2:4`	
Example: cross-NUMA STREAM Copy	
`likwid-bench -t copy -w M0:100MB:7:1:2-0:M1,1:M1`	

likwid-perfctr

Count hardware performance events. Can be used as wrapper application without modifying the source of the monitored code or with a marker API to restrict counting to parts of the code.

Syntax: `likwid-perfctr [options] [your_binary [args]]`	
`-V, --verbose <level>`	Verbose output
`-c,-C <list>`	Core ids to count events on. `-C` pins the threads
`<list>`	Like `-c` but also pin threads
`-G <list>`	GPU ids to count events on
`-g, --group <string>`	CPU performance group or custom event
`-W, --gpugroup <string>`	GPU performance group or custom event
`-H`	Get group help
`-s, --skip <hex>`	Bitmask with threads to skip for pinning
`-M <0|1>`	Set how MSR registers are accessed
`-a`	List available performance groups
`-e`	List available events & counter registers
`-E <string>`	List available events & corresponding counters that match `<string>`
`-i, --info`	Print CPU info
`-T <time>`	Switch to next event set after `<time>`
`-f, --force`	Force overwrite of in-use registers
Modes:	
`-S <time>`	Stethoscope mode with duration (in s or ms)
`-t <time>`	Timeline mode, measure after `<time>`
`-m, --marker`	Recognize LIKWID markers in code
Output options:	
`-o, --output <file>`	Store output to file
`-O`	CSV output
`--stats`	Always print statistics table
Event set syntax (multiple `-g` options allowed):	
`-g <group>`	Count performance group
`-g <event>:<counter>`	Count event `<event>` with counter `<counter>`
`-g <e1>:<c1>,<e2>:<c2>,..`	Combine multiple events using ‘,’
`-g <event>:<counter>:<opt>`	Count event `<event>` with counter `<counter>` and additional option `<opt>`

MarkerAPI

Instrument code region for C/C++. Get MarkerAPI macros from LIKWID header `<likwid.h>`. Link code to the LIKWID library and define `LIKWID_PERFMON` during build.
LIKWID Marker API bindings exist also for Fortran, Python, and Java.

Macro:	Comment
`LIKWID_MARKER_INIT`*	Initialize LIKWID Marker API.
`LIKWID_MARKER_THREADINIT`	Add thread to Marker API
`LIKWID_MARKER_START(tag)`	Start code region named `tag` (string)
`LIKWID_MARKER_STOP(tag)`	Stop code region named `tag`
`LIKWID_MARKER_CLOSE`*	Finalize LIKWID Marker API.
Optional Macro:	Comment
`LIKWID_MARKER_REGISTER(tag)`	Register code region identifier `tag` (less `START` overhead)
`LIKWID_MARKER_GET(tag)`	Get results for code region `tag`
`LIKWID_MARKER_SWITCH`*	Switch to next event set

* must be called in a serial region

Markers are recognized if the application is wrapped by **likwid-perfctr** with the **-m** option.

likwid-powermeter

Syntax: `likwid-powermeter [options] [your_binary [args]]`	
`-V, --verbose <level>`	Verbose output
`-M <0|1>`	Set how MSR registers are accessed
`-c <list>`	Specify socket(s) to measure on
`-i, --info`	Print power-related processor info
`-s <time>`	Measure for specified time
`-p`	Print dynamic clocking & CPI values (uses **likwid-perfctr** with ENERGY group)
`-t`	Print current core temperatures [°C]
`-f`	Print current core temperatures [°F]

When used as a wrapper, **likwid-powermeter** does not do any pinning of application threads.

likwid-genTopoCfg

Syntax:	`likwid-genTopoCfg [options]`
`-o, --output <file>`	Use `<file>` instead of default

likwid-mpirun

Syntax: `likwid-mpirun [options] your_binary [args]`	
`-d, --debug`	Debugging output
`-n, -np <count>`	Set the number of processes
`-nperdomain <domain>:<count>`	Set the number of processes per node by affinity domain and count; see **likwid-pin** for domains
`--pin <list>`	Specify pinning of threads
`-d,--dist <count>(:<order>)`	Specify distance between MPI processes. `<order>`: ‘close’ or ‘spread’.
`-t,-tpp <count>`	Specify threads per process
`-s, --skip <hex>`	Bitmask with threads to skip
`--mpi <id>`	Specify which MPI should be used
`--omp <id>`	Specify which OpenMP should be used
`--hostfile`	Use custom hostfile
`-g, --group <string>`	Activate event counting; see **likwid-perfctr**
`-m, --marker`	Activate marker API mode
`-O`	CSV output
`-f, --force`	Force overwrite of in-use registers

Example: 2 processes per host, 1 per socket, 2 threads

```
likwid-mpirun -pin S0:0-1_S1:0-1 ./a.out
```

Example: 2 processes per socket, count MEM group

```
likwid-mpirun -nperdomain S:2 -g MEM ./a.out
```

likwid-perfscope

Syntax: `likwid-perfscope [options] your_binary [args]`	
`-V, --verbose <level>`	Verbose output
`-a`	Print all preconfigured plot configurations for the current system.
`-c <list>`	Core ids to count events on
`-C <list>`	Like `-c` but also pin threads
`-g, --group <string>`	Preconfigured plot group or custom event set string with plot config
`-t, --time <time>`	Update interval (default: 1 s)
`-f, --force`	Force overwrite of in-use registers
`-d, --dump`	Print data as it is sent to feedGnuplot
`-p, --plotdump`	Use dump functionality of feedGnuplot. Outputs plot configurations plus data to directly feed to gnuplot
`--host <host>`	Execute command and measurements on remote host using SSH