LIKWID 4: Lightweight Performance Tools

Jan Eitzinger, Thomas Röhl, Georg Hager and Gerhard Wellein
Erlangen Regional Computing Center (RRZE)
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstrasse 1, 91058 Erlangen, Germany
Email: jan.eitzinger@fau.de

***Abstract*—LIKWID is a collection of command-line tools for performance-aware programmers of multicore and manycore CPUs. It follows the UNIX design philosophy of "one task, one tool." Among its capabilities are system topology reporting, enforcement of thread-core affinity for threading, MPI, and hybrid programming models, setting clock speeds, hardware performance event counting, energy measurements, and low-level benchmarking. In this poster we describe the feature set of the current LIKWID version and elaborate on the developments added in recent years: a new software architecture with script language APIs, a LIKWID core library for tool development, systematic validation of hardware performance event counts, and more. We aim for LIKWID to provide a one-stop solution for running and analyzing high-performance software on current multi-and manycore systems and for developing more advanced tools on top of the library interface.**

I. Introduction

LIKWID ("Like I Knew What I'm Doing") is a set of easy to use command line tools that addresses several key problems encountered at performance-oriented programming in a Linux environment. Those are probing thread/core and cache topology and querying specifications of shared memory nodes, enforcing thread-core affinity on a program, easy access to and sensible interpretation of performance counter data, control and setup of configurable node properties and microbenchmarking for reliable upper performance bounds. It supports most recent Intel and AMD processor architectures. LIKWID development started in 2009 and it was first presented in a paper 2010 [1] and on a Supercomputing poster in 2011 [2]. LIKWID has been constantly improved and extended over the last few years. One distinct feature of LIKWID is that it has no external dependencies and implements everything (including hardware performance counter support) in user space. It is one of the few Open source HPM tools which is not based on the Linux perf interface. This allows us to provide full support for new architectures quick and independent from special kernel versions. This poster will introduce LIKWID to new users and give an update on new developments in LIKWID version 4 (latest release June 2016).

II. Tools

LIKWID comprises the following tools:

- `likwid-features` displays and alters the state of the hardware prefetchers on Intel x86 processors. This is may improve performance for some work loads and give insight on performance characteristics of an application.
- `likwid-topology` probes the hardware thread and cache topology in multi-core, multi-socket nodes. Knowledge like this is required to optimize resource usage like, e.g., shared caches and data paths, physical cores, and ccNUMA locality domains in parallel code.
- `likwid-perfctr` measures performance counter metrics over the complete runtime of an application or, with support from a simple API, between arbitrary points in the code. Predefined validated event sets with derived metrics, so called performance groups, are provided on all processors.
- `likwid-pin` enforces thread-core affinity in a multi-threaded application "from the outside," i.e., without changing the source code. It works with all threading models that are based on POSIX threads, and is also compatible with hybrid "MPI+threads" programming. An accessible and still powerful thread domain concept, shared among all LIKWID tools, is provided to capture topology features in a single systematic interface.
- `likwid-mpirun` allows to pin a pure MPI or hybrid MPI/threaded application to dedicated compute resources in an intuitive and portable way. It provides integrated support for hardware performance counter measurements.
- `likwid-bench` is a microbenchmarking framework allowing rapid prototyping of small assembly kernels. It supports very flexible threading, thread and memory placement, and performance measurement. likwid-bench comes with a wide range of typical benchmark cases and can be used as a stand-alone benchmarking application.
- `likwid-powermeter` gives access to RAPL energy data readings on recent Intel processors. Moreover it allows to query supported Turbo Mode Steps on Intel processors.
- `likwid-setFrequencies` is a user space frontend to the Linux kernel interfaces for enforcing specific fixed frequencies. This is crucial for meaningful benchmarking within shared memory nodes.
- `likwid-memsweeper` is a small tool which cleans up memory domains as well as the last level caches to enforce a well defined state before executing benchmarks independent from the execution history.

LIKWID is currently limited to x86-based processors. Given the strong prevalence of those architectures in the HPC market

(e.g., 90% of all systems in the latest Top 500 list are of x86 type) we do not consider this a severe limitation. Still, porting of LIKWID to POWER and ARM architectures is currently in progress and will be added soon.

An important concept shared by all tools in the set is logical numbering of compute resources inside so-called *thread domains*. Under the Linux OS, hardware threads in a compute node are numbered according to a scheme that may be unrelated to natural topological units like cache groups, sockets, etc. Since users typically think in terms of topological structures, LIKWID introduces a simple and yet flexible syntax for specifying processor resources. This syntax consists of a prefix character and a list of logical IDs, which can also include ranges. In addition to this list-based syntax we recently added a complementing expression-based syntax which allows for a more compact specification, especially on systems with many cores. It is also easier to handle in benchmarking scripts.

III. New features in LIKWID since [2]

Porting LIKWID to new processor architectures is a continuous effort. LIKWID currently supports 24 processor families including Intel Xeon Phi (KNC and KNL). Three new tools have recently been added to the suite: `likwid-setFrequencies` for reading and changing core frequency settings, `likwid-memsweeper` for eliminating the Linux file system buffer cache, and `likwid-powermeter` for reading power-related information and measuring energy consumption. The biggest change happened under the hood, however. In LIKWID 3, all tools were completely and separately implemented in C. In LIKWID 4 we introduced a new software architecture in which the core functionality is provided by a unified C library. This makes it much easier to integrate it in other frameworks and develop new tools on top of LIKWID. The command line tools are now written in Lua and use a Lua API for LIKWID. A scripting language implementation of the frontend tools allow us to provide more dynamic features, such as adding performance groups without recompilation. We chose Lua because the interpreter is small enough to be integrated in LIKWID. Also we now use the hwloc library, which is part of the OpenMPI project, for querying node topology information. Apart from the Lua Likwid API there are also APIs for Python and Java (markers only), which were contributed by the community.

To address the possibility of unreliable or broken events we now provide a user-extensible accuracy test suite. It uses `likwid-bench` to run tests with known results, which are then compared with the measured event counts. Test results are also published on the LIKWID wiki pages. A current effort aims at reducing the overhead of HPM measurements [3]. LIKWID relies on the MSR device file as well as the PCI subsystems of the Linux kernel. With the current MSR module every single read/write to a counter causes an OS context switch, introducing a significant overhead. We currently test alternative kernel module implementations to reduce the overhead.

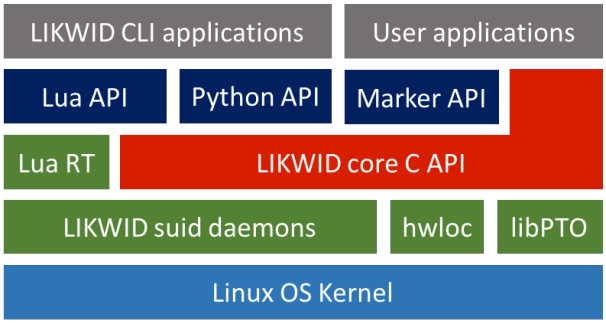

Figure 1. LIKWID 4 software architecture. Note that "libPTO" stands for the pthread-overload wrapper library, which is used for thread affinity enforcement.

IV. Future plans

LIWKID is a proven collection of small but powerful performance tools and has been extended considerably with the latest version 4. In the future we target wider applicability via, e.g., support for ARM and POWER in one of the next releases and a perf_event backend for users who cannot install the required access daemons with super-user privileges. Currently the profiling capabilities include HPM, RAPL energy data and TM/TM2 temperature data. We will also introduce a plugin mechanism to integrate other measurement facilities into LIKWID, such as network traffic or software-defined counters. The performance groups cover many metrics but are not embedded in a systematic performance analysis methodology based on bottleneck detection. We plan to offer and document performance groups supporting different performance analysis approaches such as our own pattern-based process [4] or the top-down Microarchitecture Analysis Methodology (TMAM, [5]) developed by Intel. Finally we will offer alternatives to the existing low-level Linux kernel HPM interfaces. We think there should be a low-overhead and simple interface to HPM as well as other profiling-related units (e.g., RAPL) to allow for easier user space implementations. This would create competition among tools and eventually increase the quality and diversity of profiling tool support.

LIKWID is released under GPL3. It can be downloaded at https://github.com/RRZE-HPC/likwid/.

References

[1] J. Treibig, G. Hager, and G. Wellein, "LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments," in *PSTI2010, the First International Workshop on Parallel Software Tools and Tool Infrastructures*. Los Alamitos, CA, USA: IEEE Computer Society, 2010, pp. 207–216. [Online]. Available: http://dx.doi.org/10.1109/ICPPW.2010.38

[2] J. Treibig, G. Hager, G. Wellein, and M. Meier, "Poster: Likwid: Lightweight performance tools," in *Proceedings of the 2011 Companion on High Performance Computing Networking, Storage and Analysis Companion*, ser. SC '11 Companion. New York, NY, USA: ACM, 2011, pp. 29–30. [Online]. Available: http://doi.acm.org/10.1145/2148600.2148616

[3] T. Röhl, J. Treibig, G. Hager, and G. Wellein, "Overhead analysis of performance counter measurements," in *2014 43rd International Conference on Parallel Processing Workshops*, Sept 2014, pp. 176–185.

[4] J. Treibig, G. Hager, and G. Wellein, "Performance patterns and hardware metrics on modern multicore processors: Best practices for performance engineering," in *Euro-Par 2012: Parallel Processing Workshops*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 7640, pp. 451–460. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36949-0_50

[5] Intel, "Intel 64 and IA-32 Architectures Optimization Reference Manual," http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html, Jun. 2016.