Performance Analysis with LIKWID on IBM POWER8 chips

Thomas Röhl (HPC @ Uni Erlangen)
Agenda

- LIKWID
- LIKWID on POWER8
  - Porting of LIKWID
    - Problems
    - Nice features
  - Hands on: Triangular Matrix-Vector-Multiplication
LIKWID
Overview

- „Like I knew what I am doing“
- Set of tools:
  - Topology information
  - Process/Thread pinning
  - Hardware performance monitoring
  - Low-level benchmarking
  - CPU frequency manipulation
  - CPU feature manipulation (prefetchers)

Ported to POWER8
System topology with LIKWID

**likwid-topology**

- Thread topology
- Cache topology
- NUMA topology
- Graphical topology

Socket 0:

```
+-----------------------------------------+
| +-------+ +-------+ +-------+ +-------+ |
| |  0 4  | |  1 5  | |  2 6  | |  3 7  | |
| +-------+ +-------+ +-------+ +-------+ |
| +-------+ +-------+ +-------+ +-------+ |
| | 32kB | | 32kB | | 32kB | | 32kB | |
| +-------+ +-------+ +-------+ +-------+ |
| +-------+ +-------+ +-------+ +-------+ |
| | 256kB | | 256kB | | 256kB | | 256kB | |
| +-------+ +-------+ +-------+ +-------+ |
| +-------+ +-------+ +-------+ +-------+ |
| | | | | 8MB | | | |
| +-------------------------------------+
```

Not a POWER8 system 😊
Affinity with LIKWID

**likwid-pin**

- LIKWID defines affinity domains:
  - Node (N:0-19)
  - Last Level Cache (C0:0-5)
  - Socket (S0:0-5)
  - NUMA domain (M0:0-5)
- Also physical, logical, function-based selection
LIKWID
Hardware performance monitoring (HPM)

- Measure hardware behavior
- Should be accurate (no vendor guarantees)
- Difficult management
  - Access to counters restricted
  - Programming the valid event configurations
  - What to do with the raw counts?
  - Each (micro-)architecture has different registers / events
Event names are mostly not intuitive -> difficult selection

Performance groups combine event set and derived metrics

Easy event set management

Get counter results and derived metrics (bandwidths, ratios, …)

Examples:

- likwid-perfctr –c S0:0-3@S1:0-3 –g L3 ./a.out
- likwid-perfctr –C E:N:10:1:2 –g FLOPS_VSU ./a.out
LIKWID
Low-level benchmarking

- Benchmarking is tough! (timing, warmup effects, …)
- Don‘t forget thread pinning and memory locality
- What does the compiler do with the code?
  - Optimize loops away
  - Use different instructions as expected

- LIKWID‘s benchmark suite uses abstracted assembly
- Takes care about thread and data placement
Enable LIKWID requires:

- Arch-specific access mechanism (on POWER8 sysfs files)
- CPU model related control files (events, counters, config functions)
- Adjustments to topology, NUMA and affinity module

Not all LIKWID tools can be ported:

- No CPU frequency manipulation
- CPU feature control available but not at user-space
LIKWID on POWER8

Porting of LIKWID (nice features)

- Very detailed events (stalls caused by various units)
- Fine-grained freeze counters selection
- Combine multiple counters

- Helpful but not implemented in LIKWID (by now)
  - Three sample modes
  - Machine state or hardware events trigger counting
Some registers have special access privileges

Official documentation misleading
  - Register sizes (32 vs 64 bit)
  - Event definitions in application specific syntax

Too many events are restricted to a single counter
  - Result: cannot measure L3 hits and misses simultaneously, both require counter 1

Big- and Little-Endian fun
LIKWID on POWER8

Porting of LIKWID (Problems for groups)

- No clear differentiation of events:
  - PM_DATA_ALL_FROM_L2 does not contain all data!
  - After long search: No event for SP scalar FP ops
- Invalid metrics:
  - Completion Stall Cycles = PM_CMPLU_STALL / PM_RUN_INST_CMPL and PM_CMPLU_STALL >> PMC_RUN_CYC (???)
- Events too fine grained:
  - No event to count FP ops of any FP unit
Hands On
Triangular Matrix-Vector-Multiplication

- Often performed operation
- Work sharing problem

```c
#pragma omp parallel for
for (i = 0; i < N; i++) {
    current = 0; off = (i*N);
    for (j = i; j < N; j++)
        current += mat[off+j] * bvec[j];
    cvec[i] = current;
}
```

FMA
Hands On
Triangular Matrix-Vector-Multiplication

- Thread0 runs at full performance
- New threads never reach the performance of T0
- Linear decrease in performance
Hands On
Triangular Matrix-Vector-Multiplication

- Also data volumes show similar behavior
Hands On
Triangular Matrix-Vector-Multiplication

- Positive compared to x86: CPI not misleading

POWER8 CPI: Lower is better

18.10.2016 | Performance Analysis with LIKWID on POWER8 | Thomas Röhl
Hands On Triangular Matrix-Vector-Multiplication

- Default OpenMP scheduler
  - static distributes work equally
    - One chunk per thread
    - Round-robin based on thread ID
  - Proper partitioning required
- Or keep it simple 😊

Load Imbalance!
Hands On
Triangular Matrix-Vector-Multiplication

- Set reasonable chunk size for static scheduler
- No optimal distribution but much better
- Never split cache lines between threads
  \((16 \times 8 \text{ Bytes} = 128 \text{ Bytes})\)
Hands On
Triangular Matrix-Vector-Multiplication

Upper triangular matrix vector multiplication
One thread per physical core, OpenMP schedule(static) vs. schedule(static,16)

#Threads

FP Performance [MFLOP/s]

Imbalanced schedule(static)
Balanced schedule(static,16)
What has to be done

- Not all groups integrated (IBM CPI stack)
- Events not verified properly
- More benchmarks for likwid-bench
- Support for instruction sampling
- Add runtime changable CPU features if possible
  (prefetcher on/off)
Next plans

- Integrate 'LIKWID for POWER8' in 'LIKWID for x86'
- Add some missing features to LIKWID generally
- Release LIKWID with new major version 4.2 (hopefully until SC16)
- Extend benchmark kernels for verification
Thank you for your attention
Questions?

- LIKWID: [https://github.com/RRZE-HPC/likwid](https://github.com/RRZE-HPC/likwid)
- LIKWID on POWER8: Email me Thomas.Roehl@fau.de