Brandenburg
University of Technology
Cottbus - Senftenberg

ROME Workshop, August 23, 2016

DEALING WITH LAYERS OF OBFUSCATION
IN PSEUDO-UNIFORM MEMORY

Robert Kuban, Mark Simon Schöps, Jörg Nolte,
Randolf Rotta rottaran@b-tu.de

[1] Research supported by German BMBF grant 01IH13003.

PROBLEM: MEMORY LATENCY ON INTEL XEON PHI KNC

Example: Measuring avg. time is unstable between restarts

Affects: micro-benchmarks,
algorithm tuning,
developer's sanity. . .
also application performance?

⇒ Outline

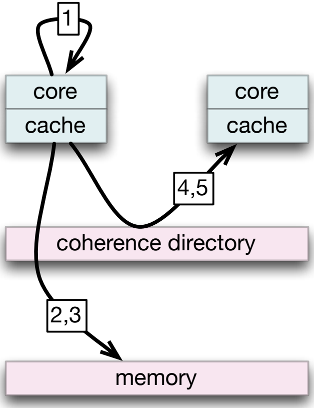
1. Causes?
2. Solutions?
3. Is it worthwhile?

OUTLINE

1. **Causes?**
2. Solutions?
3. Is it worthwhile?
4. Conclusions

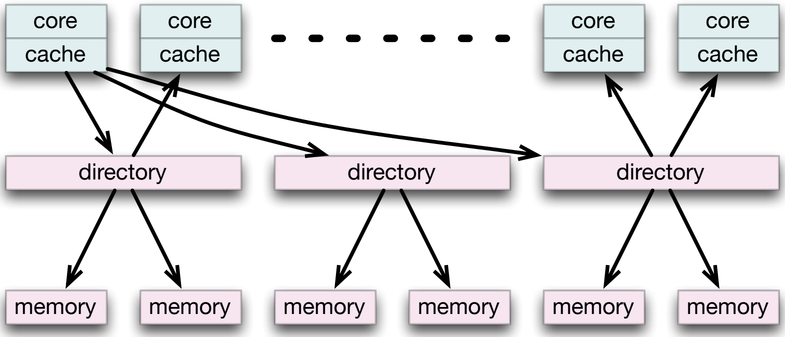
CAUSES: MULTIPLE PERFORMANCE BOTTLENECKS

1. compute bound
2. **memory throughput**:
   streaming, matrix alg.
3. **memory latency**:
   key-value stores, graphs
4. **coherence latency**:
   synchronisation variable
5. **coherence throughput**:
   many sync. variables



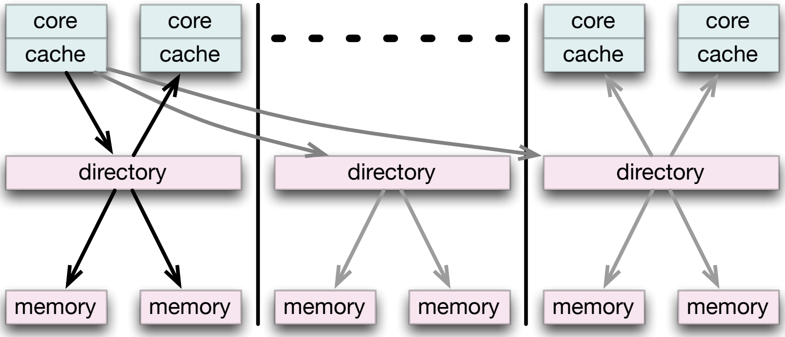
HW SOLUTION: STRIPING TO MAXIMISE THROUGHPUT

1. striping over memory channels, banks, and coherence directories
2. past: NUMA throughput bottlenecks ⇒ mostly local striping
3. many-cores: no throughput bottlenecks but larger network



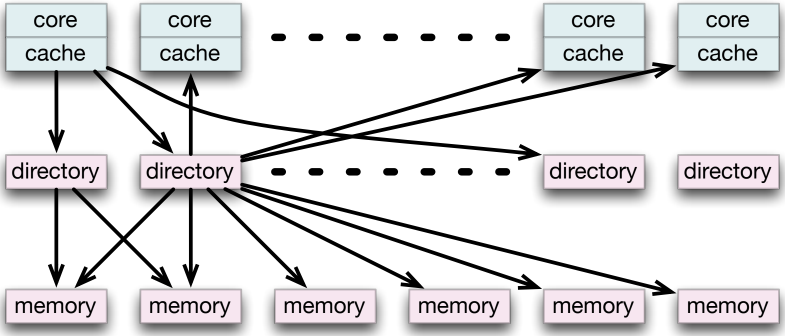
HW SOLUTION: STRIPING TO MAXIMISE THROUGHPUT

1. striping over memory channels, banks, and coherence directories
2. past: NUMA throughput bottlenecks ⇒ mostly local striping
3. many-cores: no throughput bottlenecks but larger network

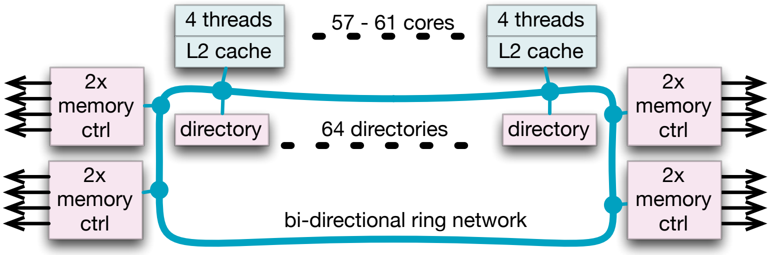


HW SOLUTION: STRIPING TO MAXIMISE THROUGHPUT

1. striping over memory channels, banks, and coherence directories
2. past: NUMA throughput bottlenecks ⇒ mostly local striping
3. many-cores: no throughput bottlenecks but larger network



INTEL XEON PHI KNC IN DETAIL



- memory striping by `(PhysAddr/62)&0xF`.[1]
- avg. remote L2 read ≈ 240 cycles, contention >16 threads.[2]
- some lines near to memory, up to 28% app. speedup possible.[3]

[1] John McCalpin: `https://software.intel.com/en-us/forums/intel-many-integrated-core/topic/586138`

[2] Ramos et al: Modeling communication in cache-coherent SMP systems: A case-study with Xeon Phi.

[3] Balazs Gerofi et al: Exploiting Hidden Non-uniformity of Uniform Memory Access on Manycore CPUs

OUTLINE

Brandenburg
University of Technology
Cottbus - Senftenberg

1. Causes?

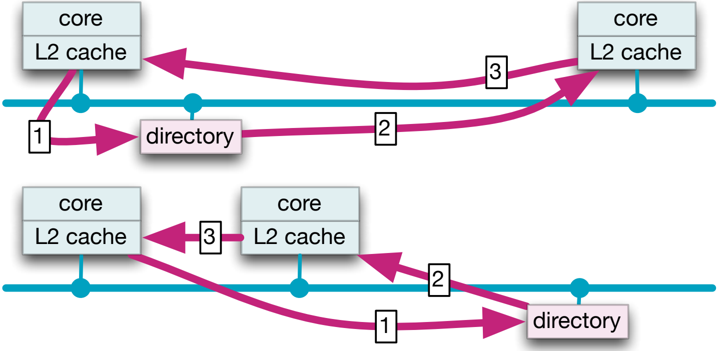
2. **Solutions?**

3. Is it worthwhile?

4. Conclusions

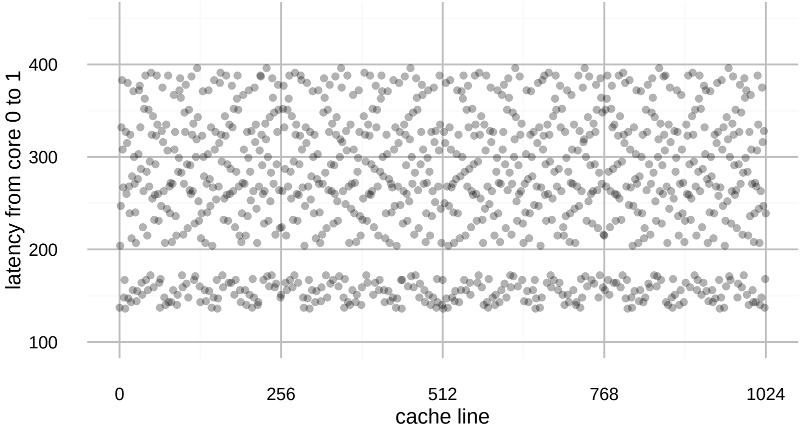
REVERSE ENGINEERING KNC'S DIRECTORY STRIPING

- measure: fetch line currently owned by neighbour L2
- two cores, two lines: one for measurement, other for coordination
- minimum RDTSC cycles, MyThOS kernel as bare-metal env.



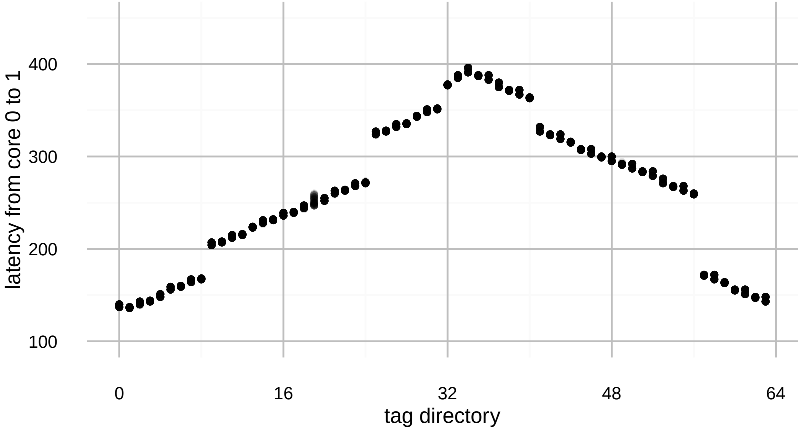
RESULTS: PSEUDO-RANDOMLY SCATTERED

≈140 cycles best case vs. ≈400 cycles worst case



RESULTS: RECONSTRUCTED MAPPING OF LINES TO DIRECTORIES

Enables quick initialisation without measurements



IMPLICATIONS

Support in the MyThOS kernel

- per page: base address for line $\mapsto$ directory
- per node: balanced mapping for directory $\mapsto$ nearby core
- kernel objects can allocate local lines for sync. vars.

Application challenges

- avoid $>$16 threads accessing same line
- co-locate dependent tasks
- squeeze synchronisation into cache lines
- no easy migration after allocation

OUTLINE

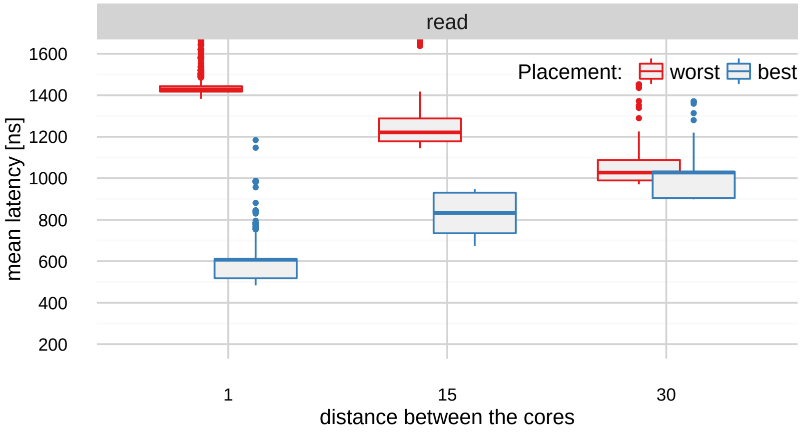
1. Causes?

2. Solutions?

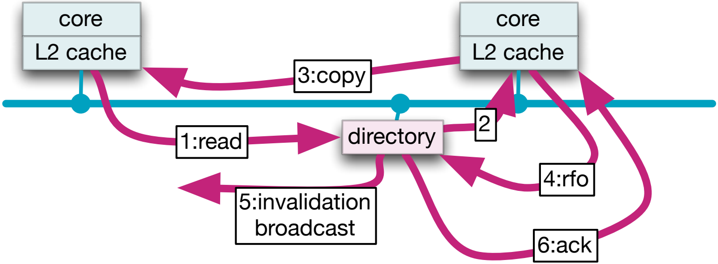
3. **Is it worthwhile?**

4. Conclusions

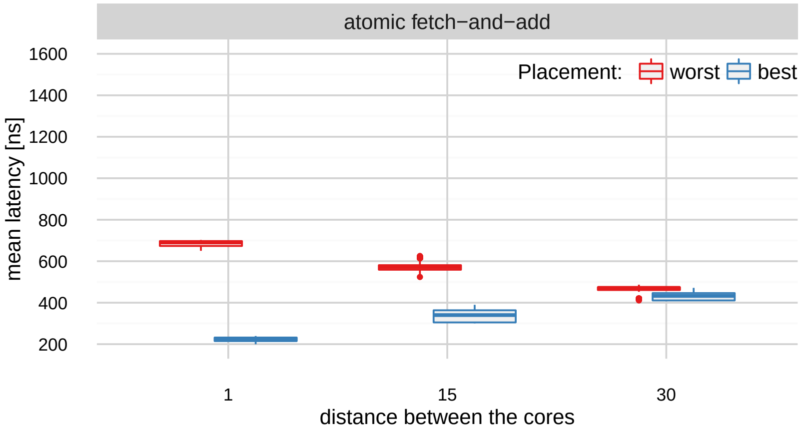
PING-PONG BENCHMARK: BUSY POLLING, THEN WRITE



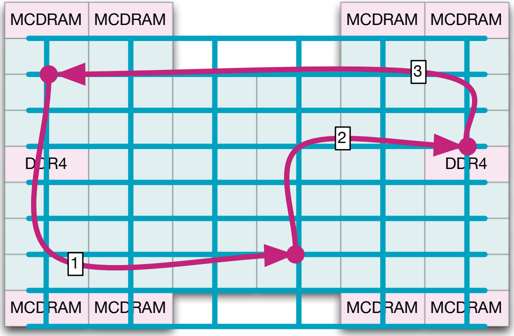
PING-PONG BENCHMARK: TIMES DON'T ADD UP



PING-PONG BENCHMARK: AVOID INVALIDATION BROADCASTS!



INTEL XEON PHI KNL: DOES IT APPLY?



- modes: all2all, quadrant, sub-numa; as memory or L3 cache
- benchmarks[4]: quadrant > all2all > sub-numa

⇒ memory + directory striping persists
smaller latency? overhead of Y-X crossing?

[4] Carlos Rosales: A Comparative Study of Application Performance and Scalability on the Intel Knights Landing Processor

OUTLINE

1. Causes?
2. Solutions?
3. Is it worthwhile?
4. **Conclusions**

CONCLUSIONS

memory striping $\neq$ directory striping

- good for throughput-bound computations,
  bad for latency- and synchronisation-bound computations

Intel KNC: pseudo-uniform

- up to 3x synchronisation latency
  but avoiding broadcasts and contention equally important
- benchmarks: average over multiple random allocations

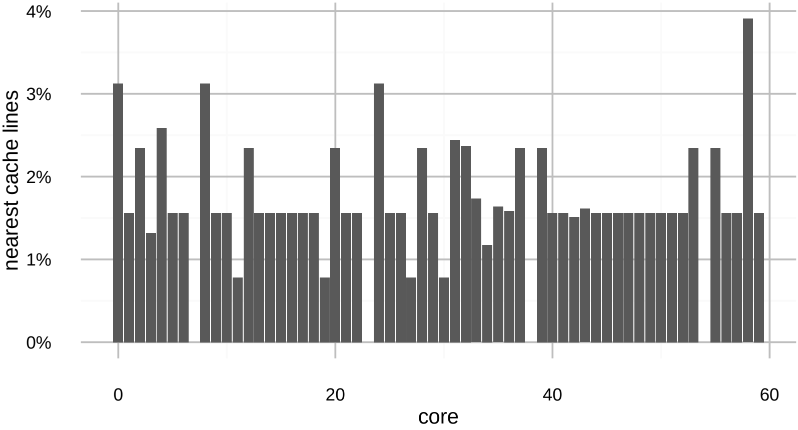
Future. . .

- MyThOS: evaluate impact on in-kernel synchronisation
- Intel KNL: latency and contention benchmarks
- HW: dedicated memory/network for synchronisation !?

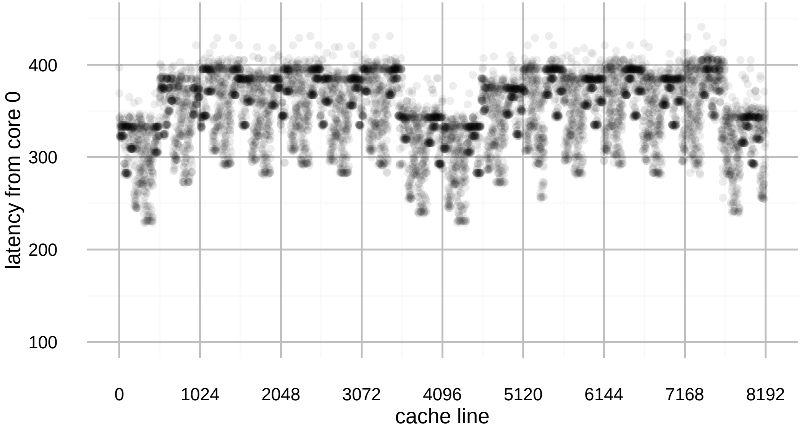
Brandenburg
University of Technology
Cottbus - Senftenberg

5. Appendix

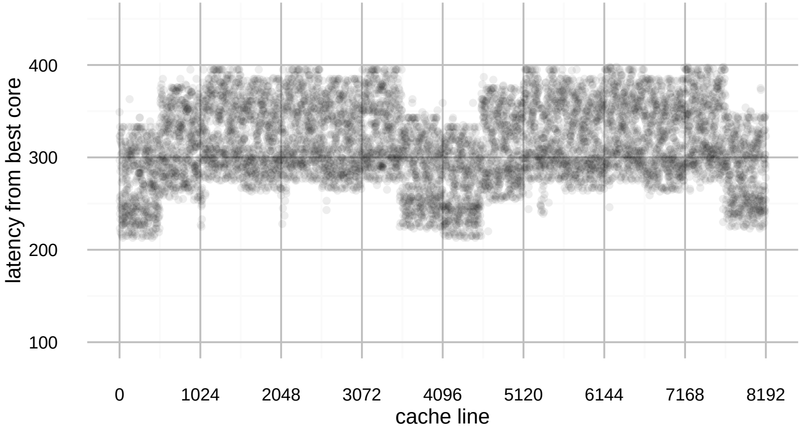
RESULTS: UNEVEN MAPPING, DEPENDS ON ENABLED CORES!



READING FROM MEMORY: LATENCY FROM CORE 0



READING FROM MEMORY: LATENCY FROM BEST CORE



“PSEUDO-UNIFORM” MEMORY ARCHITECTURES

Good for throughput bound computations

- HW maximises average throughput over large data sets, average latency hidden by prefetching & many threads

$\Rightarrow$ no need for data partitioning and placement, can focus on computation balance

Bad for latency and synchronisation bound computations

- most synchronisation variables are very small, prefetching does not help
- average latency does not apply, permanent overhead depending on placement