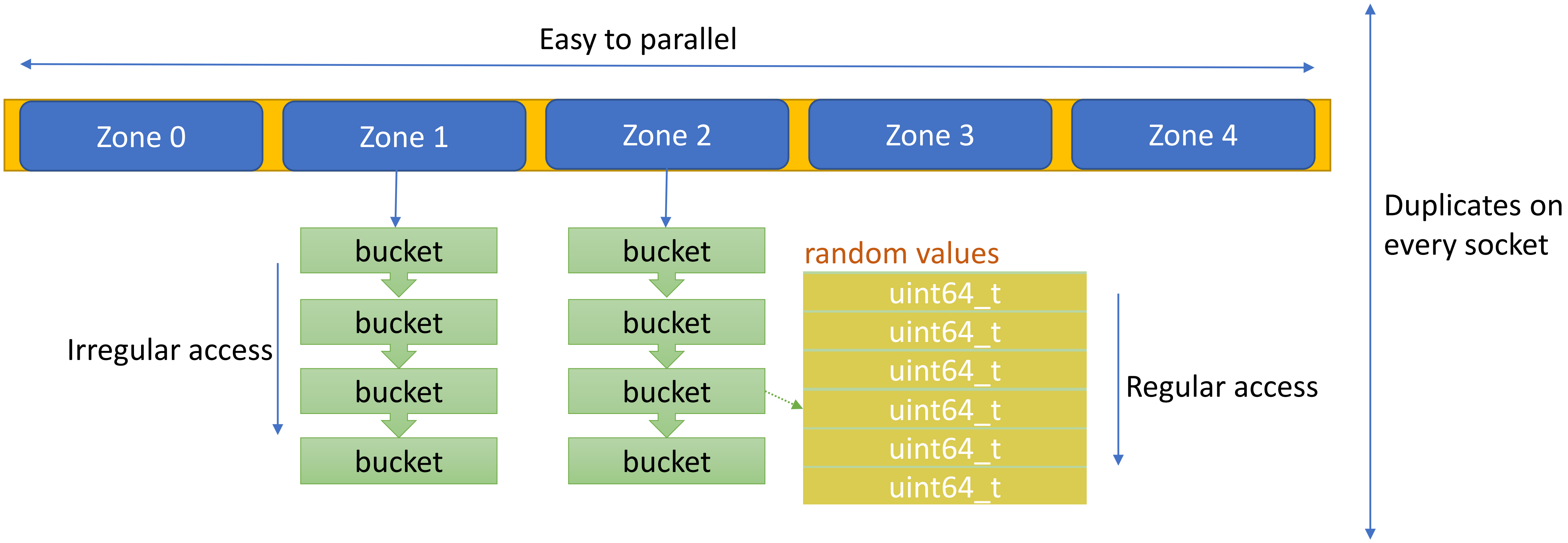
Mini benchmark – things to consider

- Cache: impact the performance across multiple runs
- CPU pipeline: memory latency might be hidden by the hardware prediction (prefetch).
- Compiler Optimization: some operations might not be actually executed if workload is too trivial
- NUMA effect: workload should always local to threads

Workload design

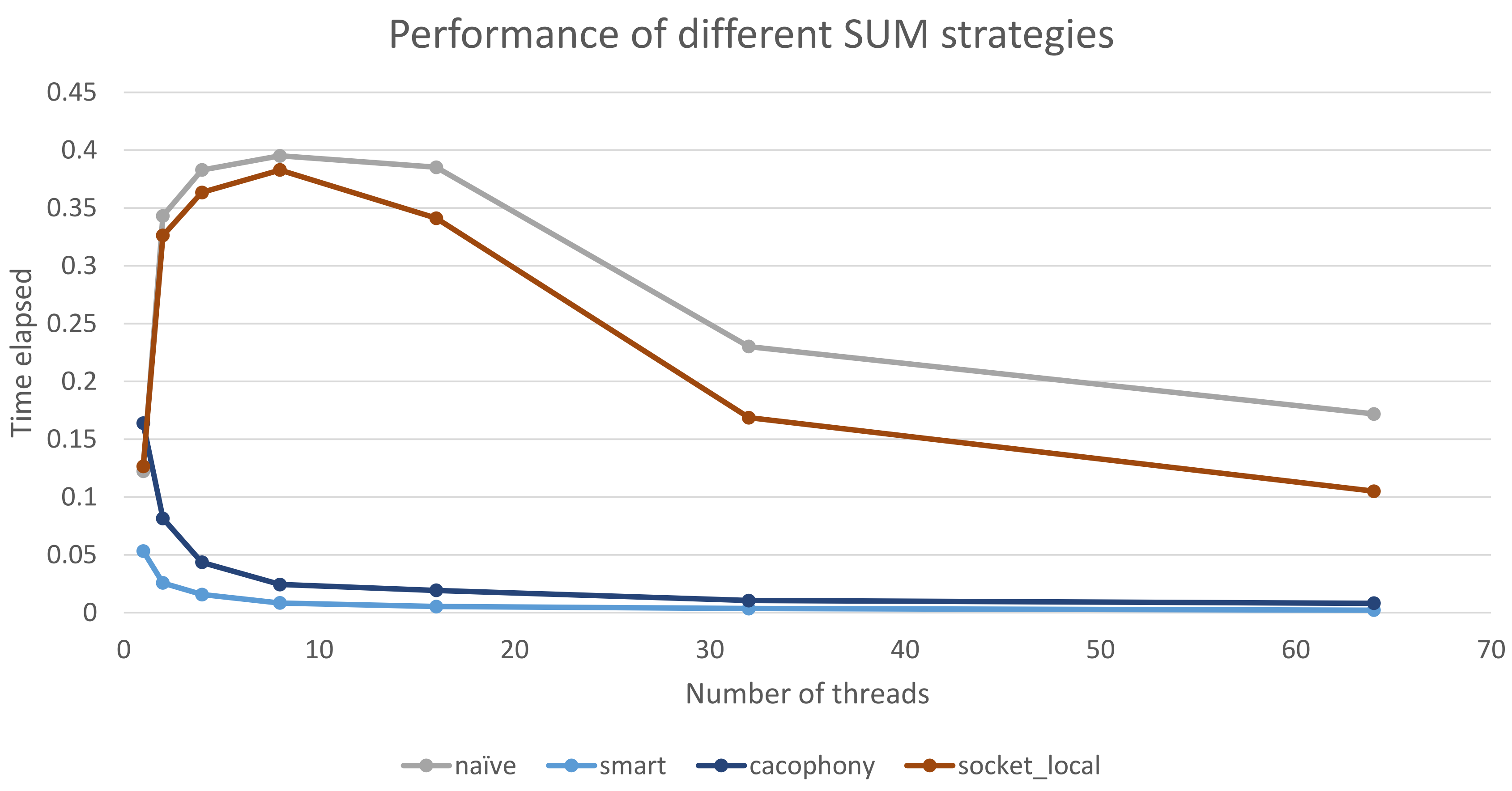


Workload goal: to max/sum all the uint64_t

Implementation Details

- All the experiments are repeated at least 3 times, the reported results are the average of them.
- Cache lines are evicted (using clflush) from all sockets across different runs.
- Thread creation/joining time not included, all threads are guaranteed to start at the same time.
- Each thread will only access its socket-local workload, in other words, no cross-socket memory access (except for the global sum, if applicable)

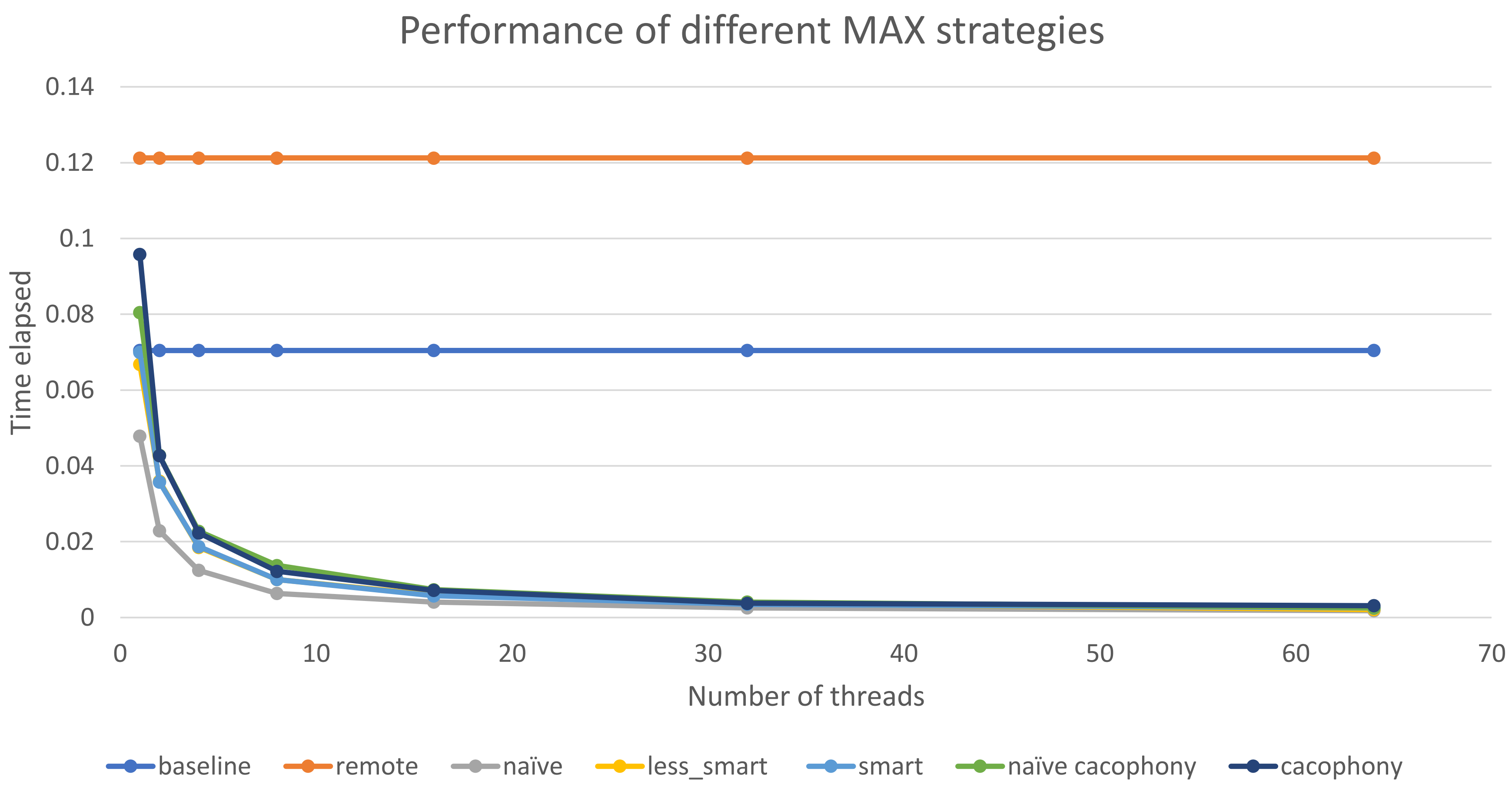
Result(continued) – socket local



Socket local:
cache coherence cost - YES
NUMA cost - NO

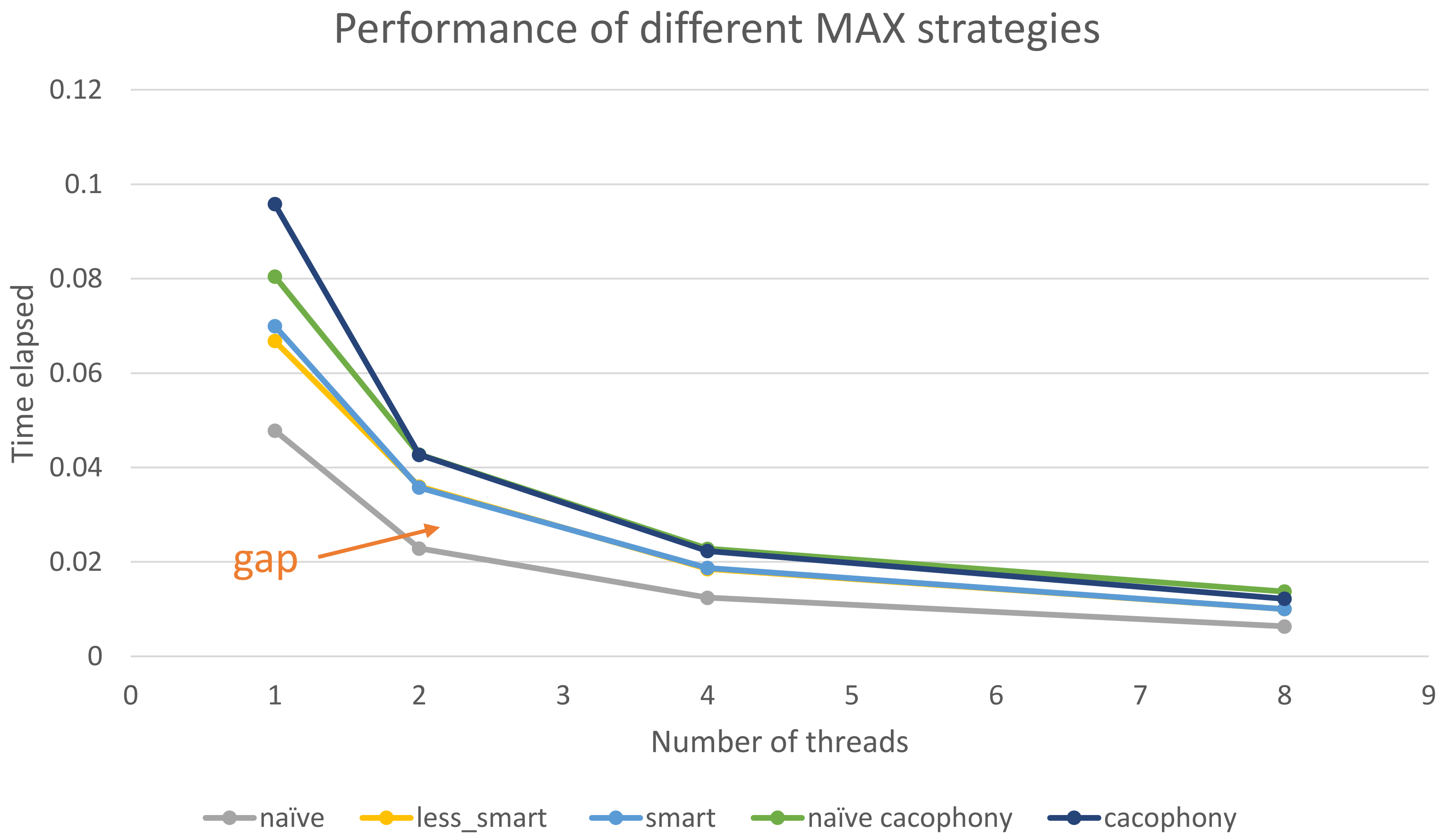
Cacophony:
cache coherence cost - NO
NUMA cost - NO

Results - overall



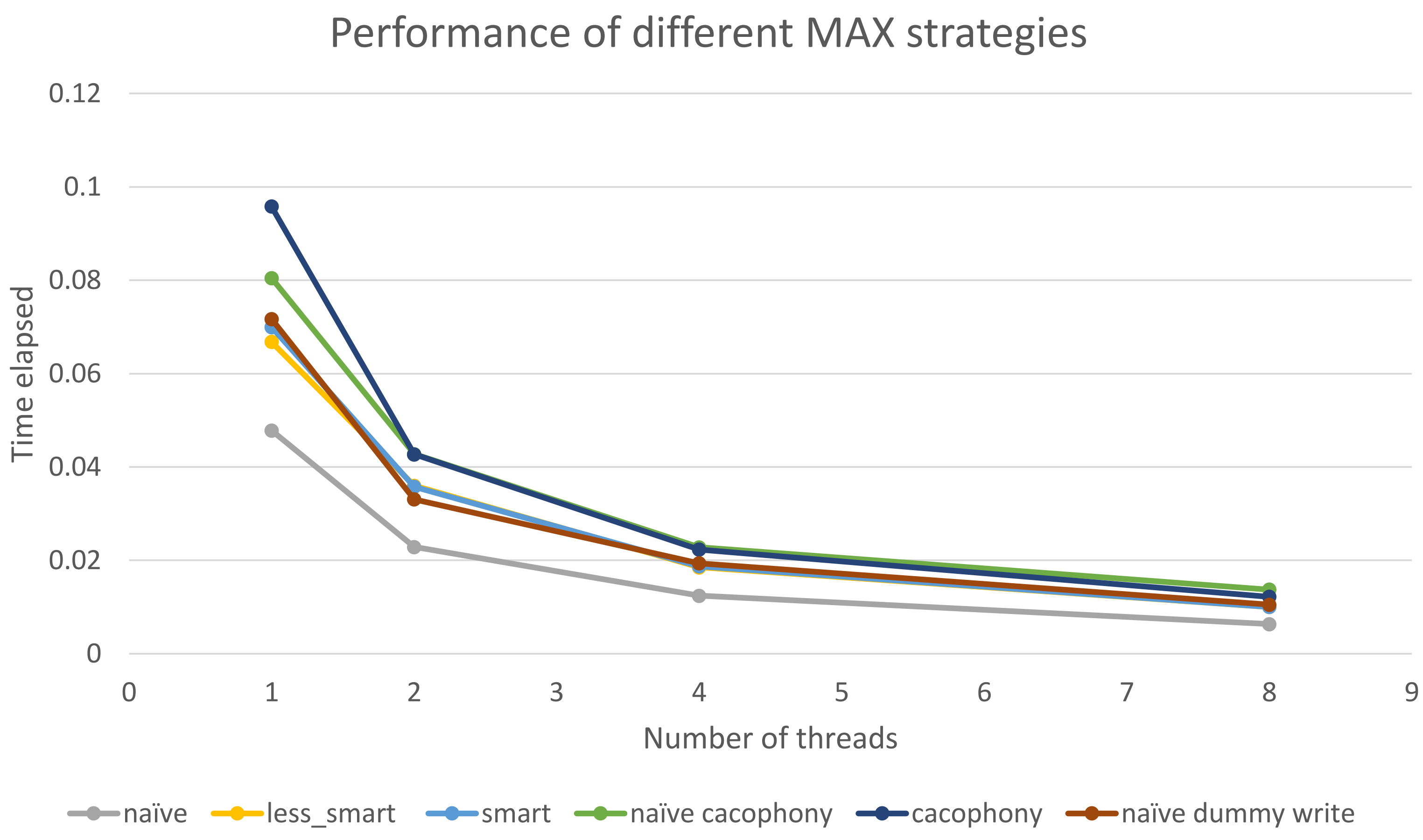
No significant difference among the implementations

Results – naïve vs smart



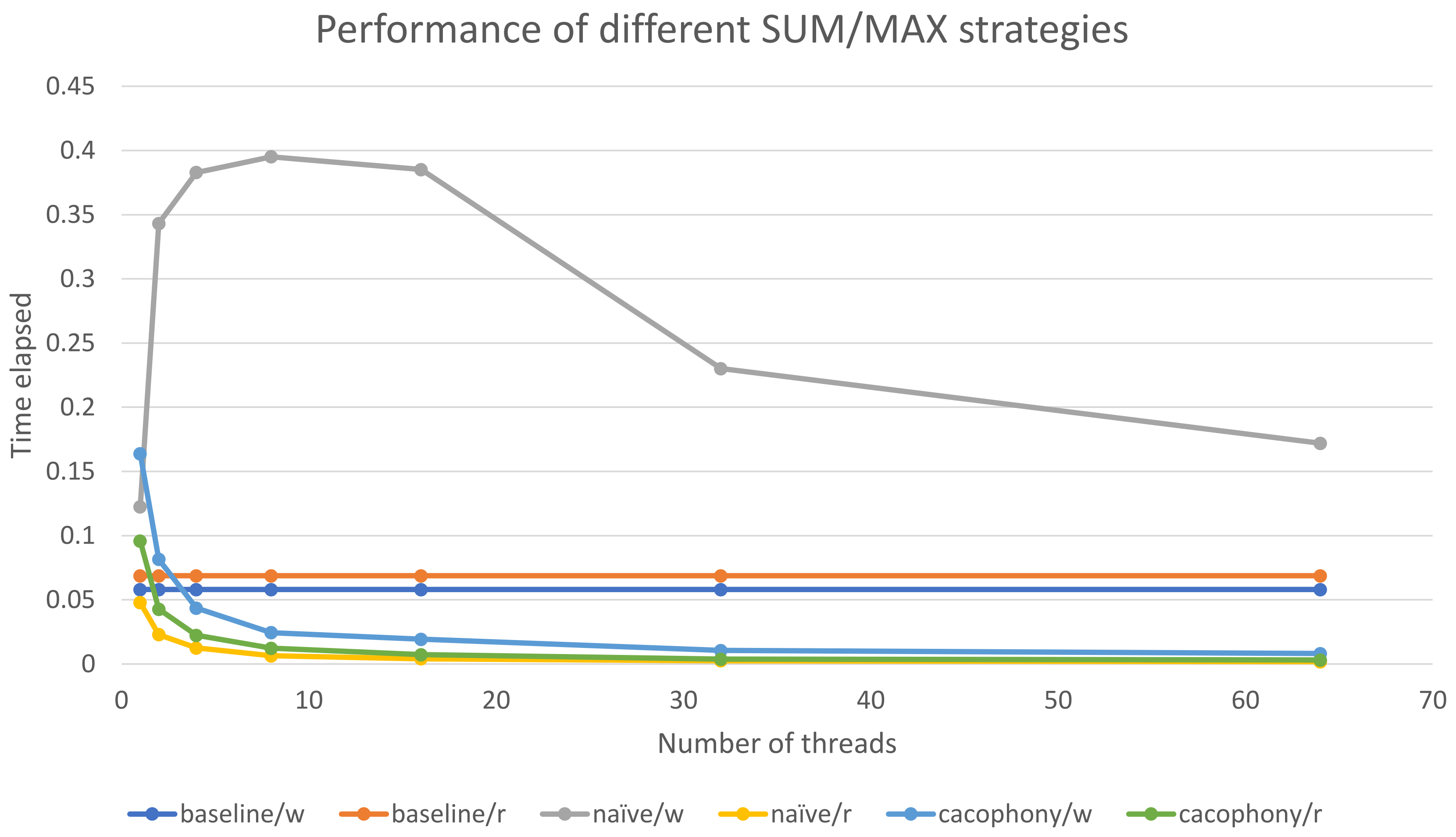
Naïve write is faster than smart

Results – naïve vs smart continued



Naïve dummy write:
Simulate smart behavior

Results – Sum(w) vs Max(r)



Baseline/w is faster than baseline/r is probably due to an extra branching in baseline/r

Naïve write is slower than naïve read because naïve write writes more.

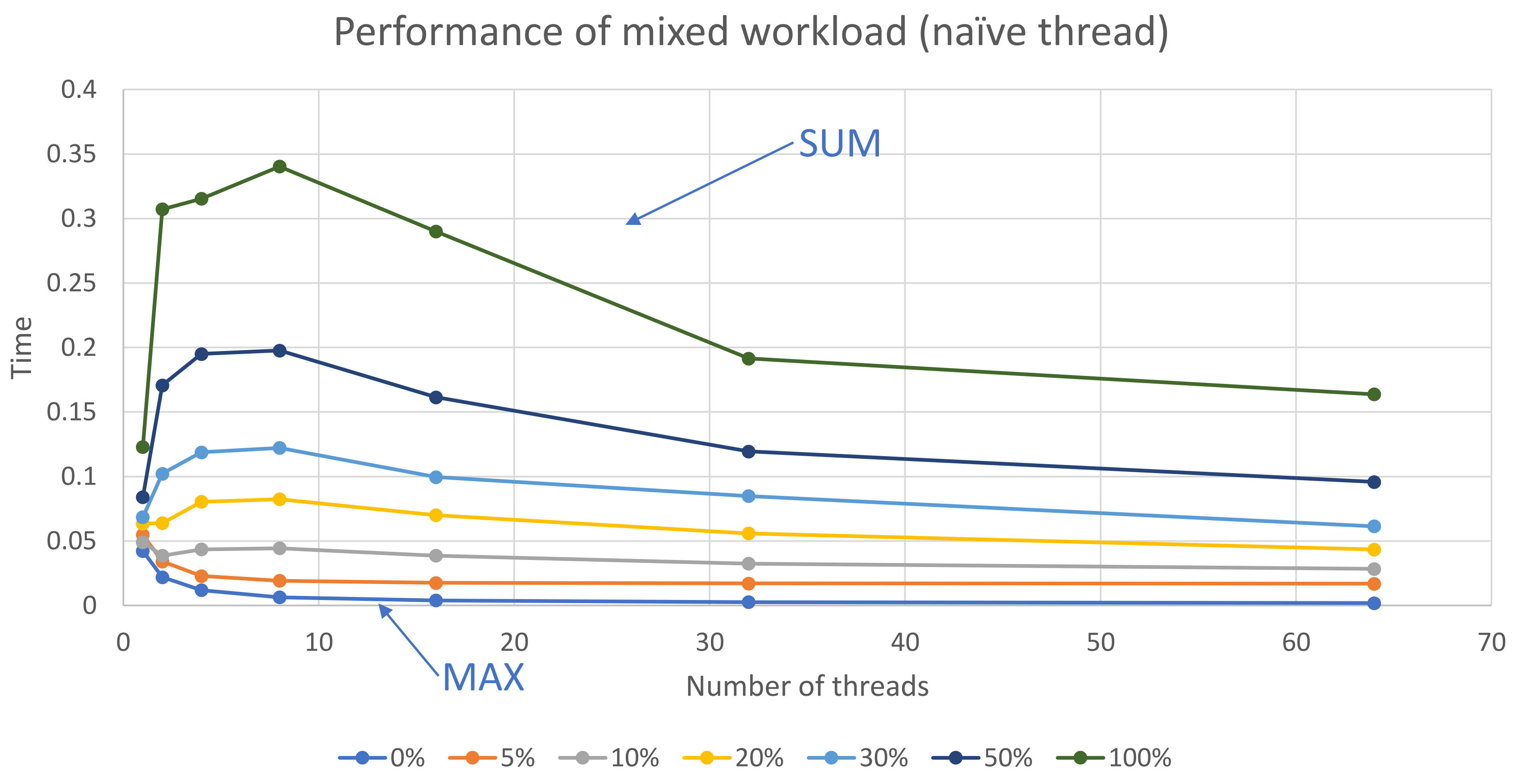
In both cases, Cacophony has consistent performance.

Characteristics of SUM and MAX

- Workload:
  - $80 * 2000 * 64 = 10240000$ integers.
- Sum:
  - 100% read + 100% write -> 10240000 writes
- Max:
  - 100% read + $\approx \frac{\ln(n) + \gamma + \frac{1}{2n} - \frac{1}{2n^2}}{n}$ write -> 16.717 writes

Experiments: 14 - 19

Results – performance vs write%



Conclusions

- NUMA effects (less) and cache coherence cost (more) are the most significant factors that impact the performance
- Naïve implementation only makes sense in (almost) all-read workload.