Benchmark Evaluation – Results Notes

1. Trends that fit with expected behavior
   1. I would suggest we first outline the obvious trends and analysis. Cache-to-performance correlation. Memory subsystem correlations, etc
   2. Kernels with predictable access patterns can have the highest performance
      1. Stride1, StrideN (depending on platform)
   3. Central Kernel and kernels with unpredictable access patterns have lowest performance
      1. Ptrchase, Random Access
   4. Indexed operations typically fall somewhere in between
      1. Scatter, Gather, SG
   5. In general, server-class processors tend to perform better than embedded/laptop/traditional desktop
      1. Large LLCs
2. New revelations
   1. The second portion of the analysis should focus on the less obvious or “hidden” trends. What happens for Phi beyond 45-ish PE’s? What happens in other results when you see hiccups, etc
   2. What can we observe that isn’t obvious in the trend lines?
   3. Why are some of the lines purely linear?
   4. For example, why does the Xeon Phi exhibit erratic behavior starting at ~45 PEs for AMO\_ADD? What does this tell us about the Phi’s memory infrastructure?
   5. Why is the Stride-N CAS so much faster than the ADD? This is orthogonal to common knowledge given the data overhead of CAS.
   6. StrideN scalability varies wildly by platform.
      1. Why? Prefetching?
         1. Size of cache seem to play a role
            1. Scalability limited by cache size?
      2. Why does the E5-2698 scale so much better than the E5-2595?
      3. Why is Phi CAS performance so high compared to Phi Add?
         1. Large “cache” helps CAS, but not add?
   7. Cortex-A53 performs almost uniformly better than Cortex-A72.
      1. Why?
   8. Core i5-3210m seems to perform very well in some tests by comparison?
      1. Why? Clang compiler?
   9. The Xeon phi performance is very erratic
      1. Particularly pronounced:
         1. As number of threads increases
         2. For the Stride1 benchmark
         3. When using atomic add
         4. Why? Something with L2 caches?
3. Conclusions
   1. Finally, draw some conclusions for our standard platforms (Xeons, etc), embedded (raspberry pi/ryzen) and others with respect to the best type of AMO to use for different memory access patterns.
   2. Server class Xeon processors seem to have the best all-around performance/most versatile
   3. Different kernels perform better with particular atomic primitives regardless of platform
      1. Add is better for some
         1. Ptrchase, Central
      2. CAS is better for others
         1. Random Access
      3. Some they seem to perform about the same
         1. Stride1, StrideN
         2. Scatter, Gather, SG
            1. overall, CAS is better for e5-2698
4. Expected or Unexpected?
   1. Xeon phi performance generally lower than other platforms with the same number of threads
      1. Expected
         1. It has no traditional LLC
         2. Has to go to MCDRAM to service requests
         3. Longer latency
      2. Unexpected:
         1. MCDRAM has 16GB capacity
         2. Should hold quite a bit of info and prevent accesses to main memory
   2. Effect of multiple sockets
      1. Expected:
         1. Cross socket degradation is clearly observable for E5-2698 in multiple benchmarks
            1. Gather, Random Access, Scatter, SG, Stride1
      2. Unexpected:
         1. Not observable for E5-2698 in other kernels
            1. Central

Performance too constrained by contention

* + - * 1. StrideN

PEs are operating on different memory segments

* + - * 1. Ptrchase

Too random?

Why does Random Access show degredation?

* + - 1. Not observable for other dual socket platforms

1. Behavioral mutterings/scratch notes by kernel
   1. Random Access
      1. Up to around 16 GAMs
      2. Fairly smooth performance curves
      3. CAS outperforms Add
      4. Xeon e5-2698 performance drops cross sockets
      5. E5-2695 v4 has by far the best performance
      6. Xeon Phi some of the worst
   2. Stride-1
      1. Up to approx. 170 GAMs
      2. I7-3930k has best performance for <= 5 threads
      3. I5-3210M does not do as well here
      4. E5620 and X5650 seem to have performance drops at sockets threads + 2-3
      5. Xeon e5-2698 has best performance, particularly within a single socket
         1. Huge drop off after crossing socket boundary
      6. Xeon Phi CAS and ADD both become erratic around 47-48 PEs
      7. Little performance difference between CAS and Add within a platform
   3. Stride-N
      1. Up to approx. 600 GAMs
      2. Big differences in performance by platform
         1. Some have scale well as number of PEs grows
         2. Other reach limit and flatten quickly
         3. E5-2695 does not scale, but e5-2698 scales very well
            1. Why?
      3. I7-3930k, xeon E5-2698 perform well
      4. E5650, opteron, cortex-A53, cortex-A72 perform poorly
      5. Xeon phi
         1. CAS performance scales extremely well, best performance after approx. 15 threads
         2. ADD does not scale well, worse than e5-2698 for same number of cores
   4. Pointer Chase
      1. Up to 14 GAMs
      2. Opteron, Xeon Phi has some of the lowest performance per core
      3. Add seems to perform a bit better
      4. Cortex A-53/i7-3930k perform best within their thread range
      5. Xeon e5-2598 has best performance per thread at higher counts
      6. Xeon phi is very erratic, particularly for add
         1. Has high point at approx. 51-52 PEs
   5. Central
      1. Overall GAMs Is much lower;
         1. Never reaches > 2 GAMs
      2. Cortex A-72 performance is particularly poor
      3. Add performance better than CAS
      4. Ryzen performance is better than others for same number of cores (small graph)
      5. Performance drops at 2 PEs and then rises again
         1. Ryzen is exception, doesn’t drop at 2, steadily increases
      6. Xeon E5-2695 performs particularly well
      7. Xeon E5-2670 add performance is good; CAS just okay
      8. Xeon Phi performance is one of the worst per thread count
         1. No shared LLC, lots of invalidations, always pulling from MCDRAM
      9. No noticeable cross socket performance penalty
         1. Already hindered by access to single memory location
   6. Scatter
      1. Up to approximately 32 GAMs
      2. I5-3210m performs best when thread count <= 2
      3. Add and CAS performance approximately equal
      4. Add performance degrades across sockets for e5-2698
      5. E5-2695 has best performance
      6. Xeon phi some of the worst
      7. Xeon phi add performance becomes erratic at around 56-58 pes
   7. Gather
      1. Up to 35 GAMS
      2. Very smooth curve, predictable
      3. I5-3210m has best performance for 1 and 2 threads
      4. I7-3930k has good performance in small graph
      5. Xeon E5-2695 best performance in medium graph
      6. Xeon e5-2698 add noticeable degrades at 17 threads
         1. Crossing sockets? – Yes
      7. Xeon phi one of the worst performers
         1. Add is more erratic
      8. For the most part, very little performance difference between Add and CAS
   8. Scatter/Gather
      1. Up to about 46 GAMs
      2. E5-2698 again shows cross socket-performance degradations
      3. E5-2695 has best performance
      4. Xeon phi Add performance more erratic than CAS