CircusTent Benchmark Evaluation Results Outline

1. We first introduce the results/graphs for each kernel in individual subsubsections and make some brief high-level comments regarding the performance/trends of each.
   1. Random Access
      1. Performance of i7-4980hq, Cortex-A72, Opteron and E5620 is the lowest
      2. 2695, i7-3930k, and 2698 are highest
      3. CAS performance is higher than add for same platforms
      4. Core i5-3210M is higher than most platforms for 1 and 2 threads
      5. Xeon E5-2698 performance drop across sockets is readily observable
      6. E5-2695 v4 has by far the best performance
      7. Xeon Phi performance is very low
      8. Up to 17.19 GAMS with E5-2695 CAS
   2. Stride-1
      1. Again, Xeon E5-2698 performance drop across sockets is readily observable
         1. Reaches highest performance with CAS at 32 threads = 66.9799 GAMS
      2. Opteron, E5620, Cortex-A53, Cortex-A72 lowest performers
         1. Why?
      3. Core i7 3930k, Xeon E5-2670, E5-2695, E5-2698 highest performance
         1. Why?
      4. Xeon phi performance is low again
         1. Why?
      5. Core i5-3210 performance is low
      6. Xeon Phi CAS and ADD both become erratic around 47-48 PEs
         1. High performance at 65 threads = 171.008 GAMs
      7. Little performance difference between CAS and Add within a platform
   3. Stride-N
      1. Big differences in performance by platform
         1. Some have scale well as number of PEs grows
         2. Other reach limit and flatten quickly
      2. E5-2695 does not scale as well as e5-2698
         1. Why?
      3. E5-2698
         1. Max GAMs: 106.819 at 32 PEs (CAS)
      4. I7-3930k, xeon E5-2698 perform well
      5. E5650, Opteron, cortex-A53, cortex-A72, core i5-3210M perform poorly
      6. Xeon phi
         1. CAS performance scales extremely well, best performance after approx. 15 threads
         2. ADD does okay, worse than e5-2698 for same number of cores
         3. 601.714 GAMs with CAS at 68 PEs
      7. Looks like total cache size forms upper bound for performance
      8. CAS and Add performance approximately equal except for Phi
   4. Pointer Chase
      1. Lower performance compared to StrideN, Stride1
         1. More random-access patterns like Random Access kernel
      2. Lowest Performance: Opteron, E5620, Cortex-A72, Core i5-3210M, Xeon Phi
      3. Highest Performance: i7-3930k(add), Cortex-A53(add), e5-2698
      4. Xeon-class processors do not outperform the smaller processors here
         1. No benefit from larger caches
      5. Xeon Phi
         1. One of the worst performers
            1. No benefit from cache + lower clock speed
         2. Erratic behavior, particularly for Add
            1. Why?
      6. Add performs better than CAS
   5. Central
      1. Contention forces GAMS much lower
         1. Never exceeds 2
      2. Worst Performance: Core i5-3210M, Cortex-A72, Xeon Phi
      3. Best Performance: Ryzen V1605B, Xeon E5-2670, E5-2695, E5-2698
      4. Ryzen performance?
         1. Check cache organization
      5. Fairly smooth and linear across platforms
         1. Contention is most limiting factor
      6. Add performance notably higher than CAS
      7. No noticeable cross socket performance penalty for any platform
         1. Already hindered by access to single memory location
   6. Scatter
      1. Worst Performance: Core I7-4980HQ (abysmal), Opteron, E5620, Xeon Phi
      2. Best Performance: Xeons (2695 better than 2698), Core i5-3210M
      3. Very smooth performance curves
      4. Cross-socket performance drop for E5-2698
         1. Observable for Add, but not CAS
      5. Add/CAS performance comparable within a system (CAS a bit more reliable)
   7. Gather
      1. Overall performance is very similar to Scatter
      2. Worst Performance: i7-4980HQ (abysmal), Opteron, E5620, Xeon Phi
      3. Best Performance: Core i5-3210M, Xeons (2695 better than 2698)
      4. Cross-socket performance drop for E5-2698
         1. Observable for Add, but not CAS
      5. Add/CAS performance comparable within a system (CAS a bit more reliable)
   8. Scatter/Gather
      1. Generally higher performance than individual scatter or gather for the same platforms, but otherwise very similar
         1. Particularly for higher thread counts
            1. Larger LLC comes into play
      2. Worst Performance: i7-4980HQ (abysmal), Opteron, E5620, Xeon Phi
      3. Best Performance: Core i5-3210M, Xeons (2695 better than 2698), i7-3930k
      4. Cross-socket performance drop for E5-2698
         1. Observable for Add, but not CAS
      5. Add/CAS performance comparable within a system (CAS a bit more reliable)
2. We then have an analysis/discussion subsection
   1. CircusTent elicits some behaviors that follow expectations and confirm previous studies/common knowledge
   2. I would suggest we first outline the obvious trends and analysis. Cache-to-performance correlation. Memory subsystem correlations, etc
      1. Kernels with predictable access patterns can have the highest performance
         1. Stride1, StrideN (depending on platform)
         2. Spatial locality - Cache can accelerate performance
         3. StrideN varies wildly by platform
            1. scalability seems directly dependent on total cache capacity
      2. Central Kernel and kernels with unpredictable access patterns have lowest performance
         1. Ptrchase, Random Access – Cache is of little help
         2. Central – Contention forms upper boundary
      3. Semi-random indexed operations typically fall somewhere in between
         1. Scatter, Gather, SG
         2. Cache helps some
   3. Size of the cache/LLC is an important factor for performance in many kernels
      1. Xeon processors perform the overall/most versatile
         1. Xeon E5-2670v2, E5-2695 v4, E5-2698 v3 have the largest caches. 25MiB, 45 MiB, 40 MiB
      2. Many of the lowest performers overall have smaller caches
         1. Cortex A-53: 2 MiB, Cortex-A-72: 1: MiB, Opteron 4130: 6 MiB, Xeon E5620 12MiB
   4. However, CircusTent also offers some new insights
   5. 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
   6. What can we observe that isn’t obvious in the trend lines?
   7. Why are some of the lines purely linear?
   8. 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?
   9. Why is the Stride-N CAS so much faster than the ADD? This is orthogonal to common knowledge given the data overhead of CAS.
      1. The OS/compiler may also play an important role
         1. Note higher performance of Ryzen, i5-3210M for some kernels despite the smaller caches
            1. I5-3120m performance could also be a consequence of smaller array size
         2. Also, notably uniform lower performance of E5620 system despite medium size cache – same cache size as i7-3930k, X5650, which perform much better
      2. As may the clock speed
         1. Note the i7-3930k performance
      3. CAS sometimes outperforms Add?
         1. Why?
         2. Doesn’t always need to perform store/stage
            1. Compare fails – CAS aborts
            2. No need to invalidate anything
            3. Particularly prominent for PHI because of so many caches
      4. Cortex-A53 performs almost uniformly better than newer Cortex-A72.
         1. Because the cache is larger
      5. I7-4980hq has worst performance in scatter, gather, sg, rand
         1. L4 is of little use
         2. Gpu forcing evictions?
      6. Cross-socket performance degradation is noticeable for E5-2698, but not other dual socket configurations
         * 1. Observable for:

Gather, Random Access, Scatter, SG, Stride1

* + - * 1. Why not in other kernels?

Central

Performance too constrained by contention

StrideN

PEs are operating on different memory segments?

Ptrchase

Too random?

Why doesn’t Random Access show degradation?

* + - * 1. Why is it not observable for other dual socket platforms?

OS/Compiler? Scheduling of threads within one socket instead?

* + 1. Differences in E5-2695 and E5-2698 performance
       1. Why does the E5-2698 scale so much better than the E5-2595 for the strideN?
    2. Xeon phi
       1. Overall, performance is very poor in comparison for same number of threads
          1. Suffers from lack of true LLC
          2. Has to go to MCDRAM to satisfy requests

Latency of MCDRAM cache is too high

Large size not as helpful as assumed

* + - * 1. Complex L2 system makes invalidations costly
      1. Performance is very erratic
         1. Particularly pronounced:

As number of threads increases

For the Stride1 benchmark

When using atomic add

Why? Something with L2 caches?

* + - 1. Xeon Phi – StrideN performance
         1. Why is CAS performance so high compared to Add?

Large MCDRAM “cache” helps CAS, but not add?

* 1. Conclusions/Recommendations
  2. 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.
     1. 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

overall, CAS is better for e5-2698

* + 1. Recommendations by platform/processor
       1. TBD