CircusTent: A Tool for Understanding the Performance of Atomic Memory Operations

Michael Beebe¹, Brody Williams¹, John D. Leidel²,¹, Xi Wang³, David Donofrio², Yong Chen¹

¹Texas Tech University, ²Tactical Computing Laboratories, ³RISC-V International Open-Source Laboratory
Motivation

- Increasingly **heterogenous** systems
- Distinct components may:
  - Utilize different ISAs
  - Necessitate the use of disparate APIs for interfacing
  - Include supplemental on-device memory hierarchies which may feature irregular organization
- Combination of these distinct memories leads to complex interconnected memory subsystems
- Behavior and performance of these memory subsystems is critical
Motivation Continued

• Parallel programming paradigms are ubiquitous to HPC and heterogeneous systems

• Multi-Core architectures are not without shortcomings
  • Scalability issues as number of cooperating PEs grow

• Unfortunately, these paradigms also necessitate the use of synchronization methods to ensure correctness

• Understanding and optimizing these synchronization methods is key
Atomic Memory Operations

• Atomic memory operations (AMOs) are used to realize these synchronization primitives
  • Barriers, mutex locks, cache coherence mechanisms often built upon atomic operations
  • Remote atomics often built on combination of RDMA verbs and local synchronization

• AMOs can also be used for “lock-free” synchronized memory accesses

• Scalability of AMOs and synchronization primitives decreases due to contention as the number of PEs rises
  • Bottleneck for existing systems
  • Important design consideration for future systems
Prevalence of AMOs in GAPBS

![Bar chart showing the proportion of AMOs in different benchmarks: BC, BFS, CC, PR, SSSP, TC.]


The GAP Benchmark Suite, Scott Beamer, Krste Asanović, and David Patterson, arXiv:1508.03619 [cs.DC], 2015.
Proportion of Remote AMOS in Distributed Scatter Operation

\[ A[B[i]] = C[i] \]

Local | Remote

<table>
<thead>
<tr>
<th>Number of Threads</th>
<th>Proportion (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td>64</td>
<td>1</td>
</tr>
</tbody>
</table>

CircusTent

• Atomic benchmark suite for read-modify-write (RMW) atomic memory operations

• Written in C/C++

• Targets API-level AMOs for physically shared and distributed shared memory paradigms
  • OpenMP
  • MPI RMA
  • OpenSHMEM
  • xBGAS RISC-V ISA Extension
  • OpenACC
  • Pthreads
  • OpenMP with Target Offloading
  • OpenCL

• Calculates Billions of Atomic Memory Operations per second (GAMS)

\[
GAMS = \frac{(PES \times Iters \times AMOs_{Per\_Iter})}{1e^9 \times time}
\]
Driving Requirement

• Ability to derive normalized results
  • Benchmark results are important to the design of future systems
  • Difficult to directly compare performance of varied systems across distinct kernels

• Support for a multitude of programming models
  • Different workloads and systems utilize a variety of programming models
  • Atomics are implemented differently in each model

• Allow the opportunity for system and model specific optimizations

• Provide pathological kernels that replicate a variety of common memory access patterns of interest
CircusTent Continued

• CircusTent consists of eight constituent kernels

• Two implementations of each kernel per back end:
  • Atomic add/fetch-and-add (FAA)
  • Atomic compare-and-swap (CAS)

• Each kernel executes N iterations of a loop for a given memory access pattern

• Each kernel uses two different arrays:
  • VAL
  • IDX

• Number of atomic operations varies between kernels

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>AMOs Per Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rand</td>
<td>1</td>
</tr>
<tr>
<td>Stride-1</td>
<td>1</td>
</tr>
<tr>
<td>Stride-N</td>
<td>1</td>
</tr>
<tr>
<td>Pointer Chase</td>
<td>1</td>
</tr>
<tr>
<td>Central</td>
<td>1</td>
</tr>
<tr>
<td>Scatter</td>
<td>3</td>
</tr>
<tr>
<td>Gather</td>
<td>3</td>
</tr>
<tr>
<td>Scatter/Gather</td>
<td>4</td>
</tr>
</tbody>
</table>

\[
GAMS = \frac{(PEs \times \text{Iter} \times \text{AMOs Per Iter})/10^9}{time}
\]
**Algorithm 1: Random Access Kernel**

\[
\text{for } i \leftarrow 0 \text{ to } \text{iters} \text{ by } 1 \text{ do}
\]
\[
\text{AMO(VAL[IDX[i]])}
\]
\[
\text{end}
\]
Algorithm 2: Stride-1 Kernel

\[
\text{for } i \leftarrow 0 \text{ to } \text{iters} \text{ by } 1 \text{ do }
\quad \text{AMO(VAL}[i]\text{)}
\text{end}
\]
Algorithm 3: Stride-N Kernel

\[
\text{for } i \leftarrow 0 \text{ to } \text{iters} \text{ by stride} \text{ do }
\begin{align*}
& \text{AMO(VAL[i])} \\
\text{end}
\end{align*}
\]
Algorithm 4: Pointer Chase Kernel

\[
\text{for } i \leftarrow 0 \text{ to } \text{iters} \text{ by } 1 \text{ do } \\
\quad \text{start} = \text{AMO(IDX[start])} \\
\text{end}
\]
Algorithm 5: Central Kernel

for $i \leftarrow 0$ to $iters$ by 1 do
  AMO(VAL[0])
end
Algorithm 6: Scatter Kernel

\[
\begin{align*}
\text{for } i & \leftarrow 0 \text{ to } \text{iters by } 1 \text{ do} \\
\text{dest} & = \text{AMO}(	ext{IDX}[i+1]) \\
\text{val} & = \text{AMO}(	ext{VAL}[i]) \\
\text{AMO}(&\text{VAL}[\text{dest}], \text{val}) \quad // \quad \text{VAL}[\text{dest}] = \text{val}
\end{align*}
\]
Algorithm 7: Gather Kernel

for $i \leftarrow 0$ to $\text{iters}$ by 1 do
    src = AMO(IDX[$i+1$])
    val = AMO(VAL[src])
    AMO(VAL[i], val) // VAL[i] = val
end
Algorithm 8: Scatter/Gather Kernel

\[
\text{for } i \leftarrow 0 \text{ to } \text{iters} \text{ by } 1 \text{ do}
\]

\[
\begin{align*}
\text{src} &= \text{AMO(IDX[i])} \\
\text{dest} &= \text{AMO(IDX[i+1])} \\
\text{val} &= \text{AMO(VAL[src])} \\
\text{AMO(VAL[dest], val)} &\quad // \text{VAL[dest]} = \text{val}
\end{align*}
\]

end
CircusTent Version 0.1
Usage: circustent [OPTIONS]

-b --bench TEST : Sets the benchmark to run
-m --memsize BYTES : Sets the size of the array
-p --pes PES : Sets the number of PEs
-i --iters ITERATIONS : Sets the number of iterations per PE
-s --stride STRIDE (elems) : Sets the stride in 'elems'

-h --help : Prints this help menu
-l --list : List benchmarks
<table>
<thead>
<tr>
<th>BENCHMARK</th>
<th>REQUIRED_OPTIONS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAND_ADD</td>
<td>No Arg Required</td>
<td>Random memory access pattern using FETCH+ADD</td>
</tr>
<tr>
<td>RAND_CAS</td>
<td>No Arg Required</td>
<td>Random memory access pattern using CAS</td>
</tr>
<tr>
<td>STRIDE1_ADD</td>
<td>No Arg Required</td>
<td>Stride-1 memory access pattern using FETCH+ADD</td>
</tr>
<tr>
<td>STRIDE1_CAS</td>
<td>No Arg Required</td>
<td>Stride-1 memory access pattern using CAS</td>
</tr>
<tr>
<td>STRIDEN_ADD</td>
<td>stride</td>
<td>Stride-N memory access pattern using FETCH+ADD</td>
</tr>
<tr>
<td>STRIDEN_CAS</td>
<td>stride</td>
<td>Stride-N memory access pattern using CAS</td>
</tr>
<tr>
<td>PTRCHASE_ADD</td>
<td>No Arg Required</td>
<td>Pointer chase memory access pattern using FETCH+ADD</td>
</tr>
<tr>
<td>PTRCHASE_CAS</td>
<td>No Arg Required</td>
<td>Pointer chase memory access pattern using CAS</td>
</tr>
<tr>
<td>CENTRAL_ADD</td>
<td>No Arg Required</td>
<td>Centralized point access using FETCH+ADD</td>
</tr>
<tr>
<td>CENTRAL_CAS</td>
<td>No Arg Required</td>
<td>Centralized point access using CAS</td>
</tr>
<tr>
<td>SG_ADD</td>
<td>No Arg Required</td>
<td>Scatter/Gather memory access pattern using FETCH+ADD</td>
</tr>
<tr>
<td>SG_CAS</td>
<td>No Arg Required</td>
<td>Scatter/Gather memory access pattern using CAS</td>
</tr>
<tr>
<td>SCATTER_ADD</td>
<td>No Arg Required</td>
<td>Scatter memory access pattern using FETCH+ADD</td>
</tr>
<tr>
<td>SCATTER_CAS</td>
<td>No Arg Required</td>
<td>Scatter memory access pattern using CAS</td>
</tr>
<tr>
<td>GATHER_ADD</td>
<td>No Arg Required</td>
<td>Gather memory access pattern using FETCH+ADD</td>
</tr>
<tr>
<td>GATHER_CAS</td>
<td>No Arg Required</td>
<td>Gather memory access pattern using CAS</td>
</tr>
<tr>
<td>Feature</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------------------</td>
<td></td>
</tr>
<tr>
<td>Architecture:</td>
<td>x86_64</td>
<td></td>
</tr>
<tr>
<td>CPU op-mode(s):</td>
<td>32-bit, 64-bit</td>
<td></td>
</tr>
<tr>
<td>Byte Order:</td>
<td>Little Endian</td>
<td></td>
</tr>
<tr>
<td>CPU(s):</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>On-line CPU(s) list:</td>
<td>0-11</td>
<td></td>
</tr>
<tr>
<td>Thread(s) per core:</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Core(s) per socket:</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Socket(s):</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>NUMA node(s):</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Vendor ID:</td>
<td>GenuineIntel</td>
<td></td>
</tr>
<tr>
<td>CPU family:</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Model:</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>Model name:</td>
<td>Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz</td>
<td></td>
</tr>
<tr>
<td>Stepping:</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>CPU MHz:</td>
<td>1201.406</td>
<td></td>
</tr>
<tr>
<td>CPU max MHz:</td>
<td>3200.0000</td>
<td></td>
</tr>
<tr>
<td>CPU min MHz:</td>
<td>1200.0000</td>
<td></td>
</tr>
<tr>
<td>BogoMIPS:</td>
<td>4794.48</td>
<td></td>
</tr>
<tr>
<td>Virtualization:</td>
<td>VT-x</td>
<td></td>
</tr>
<tr>
<td>L1d cache:</td>
<td>32K</td>
<td></td>
</tr>
<tr>
<td>L1i cache:</td>
<td>32K</td>
<td></td>
</tr>
<tr>
<td>L2 cache:</td>
<td>256K</td>
<td></td>
</tr>
<tr>
<td>L3 cache:</td>
<td>15360K</td>
<td></td>
</tr>
<tr>
<td>NUMA node0 CPU(s):</td>
<td>0-11</td>
<td></td>
</tr>
</tbody>
</table>
Future Work

• Implementations based on other PGAS models
  • Chapel
  • UPC
  • Coarray Fortran

• Device-specific models
  • CUDA

• Adding support for additional atomic primitives
Conclusion

• HPC is changing
  • Adoption of increasingly heterogeneous systems composed of novel device types

• New Challenges
  • Difficulty of measuring the performance of diverse platforms

• CircusTent is a tool for measuring the capabilities of distributed memory hierarchies within emerging heterogeneous system architectures
Repository and Contact Info

Code Repository:
• https://github.com/tactcomplabs/circustent

Contact Info:
• Michael Beebe – michael.beebe@ttu.edu
• Brody Williams – brody.williams@ttu.edu
• John D. Leidel – jleidel@tactcomplabs.com
• Yong Chen – yong.chen@ttu.edu
References


• The GAP Benchmark Suite, Scott Beamer, Krste Asanović, and David Patterson, arXiv:1508.03619 [cs.DC], 2015.