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

Michael Beebe[1], Brody Williams[1], John D. Leidel[2,1], Xi Wang[3], David Donofrio[2], Yong Chen[1]

[1]Texas Tech University, [2]Tactical Computing Laboratories, [3]RISC-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

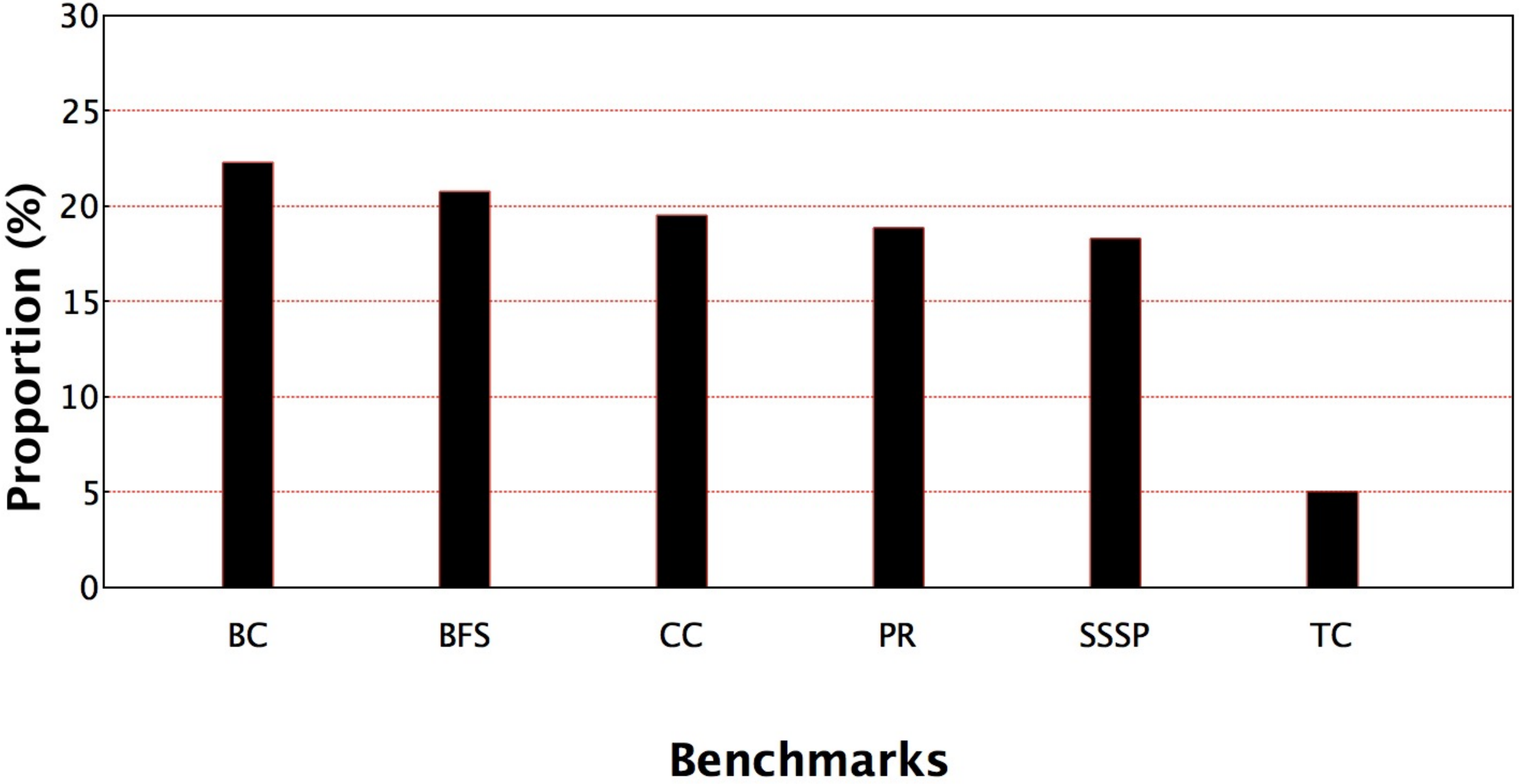
- Parallel programming paradigms are ubiquitous to HPC and heterogeneous systems
- Multi-Core architectures are not without shortcomings
  - Scalability issues as number of cooperating PEs grow
- Unfortunately, these paradigms also necessitate the use of synchronization methods to ensure correctness
- 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

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

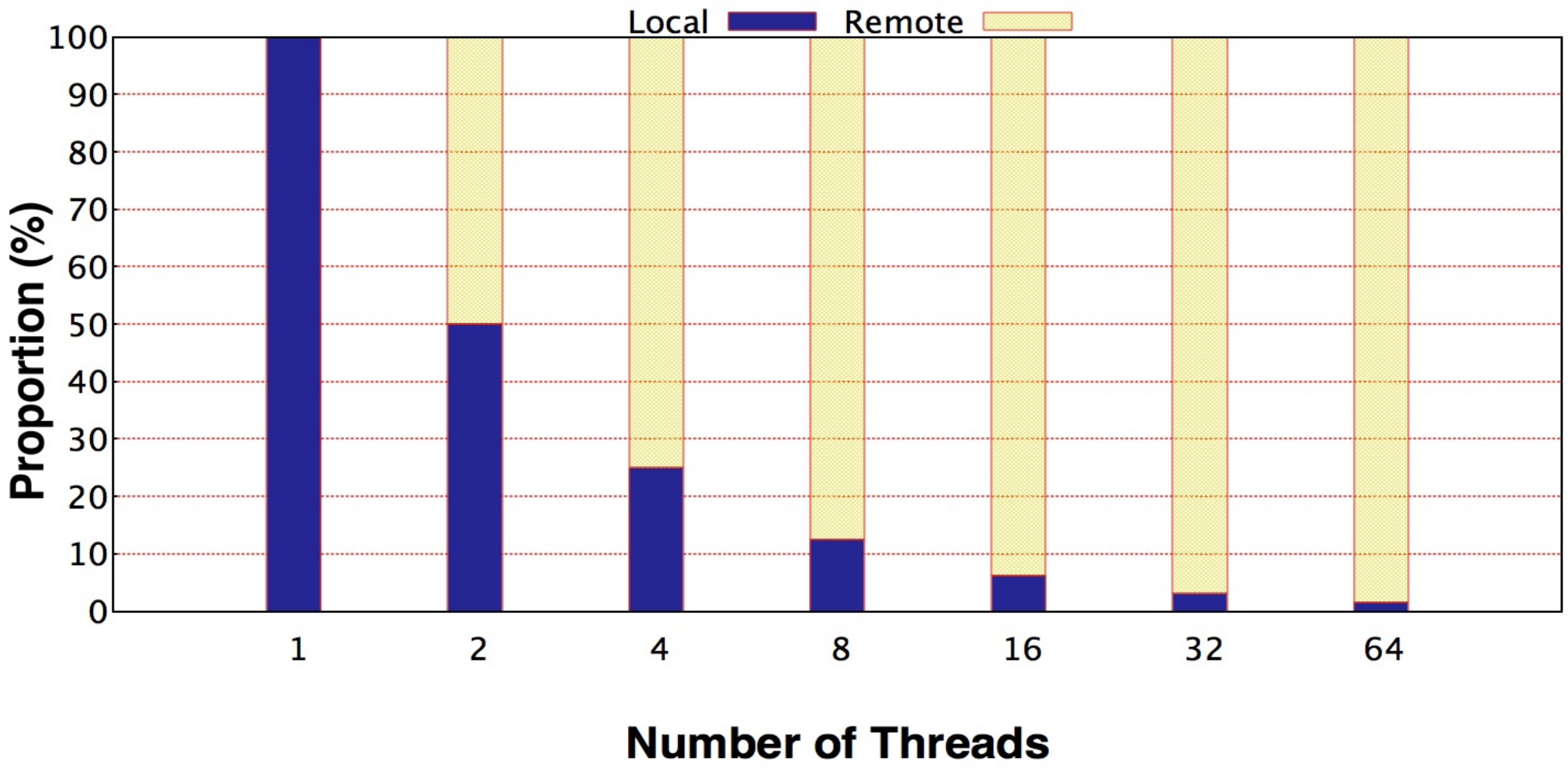
Prevalence of AMOs in GAPBS



Xi Wang, Brody Williams, John D. Leidel, Alan Ehret, Michel Kinsy, and Yong Chen. 2020. Remote Atomic Extension (RAE) for Scalable High Performance Computing. In Proceedings of the 57th Annual Design Automation Conference 2020 (DAC ’20).

Proportion of Remote AMOS in Distributed Scatter Operation

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



Xi Wang, Brody Williams, John D. Leidel, Alan Ehret, Michel Kinsy, and Yong Chen. 2020. Remote Atomic Extension (RAE) for Scalable High Performance Computing. In Proceedings of the 57th Annual Design Automation Conference 2020 (DAC ’20).

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)

$$GAMS = \frac{(PEs \times Iters \times AMOs\_Per\_Iter)/1e^9}{time}]$$

Driving Requirement

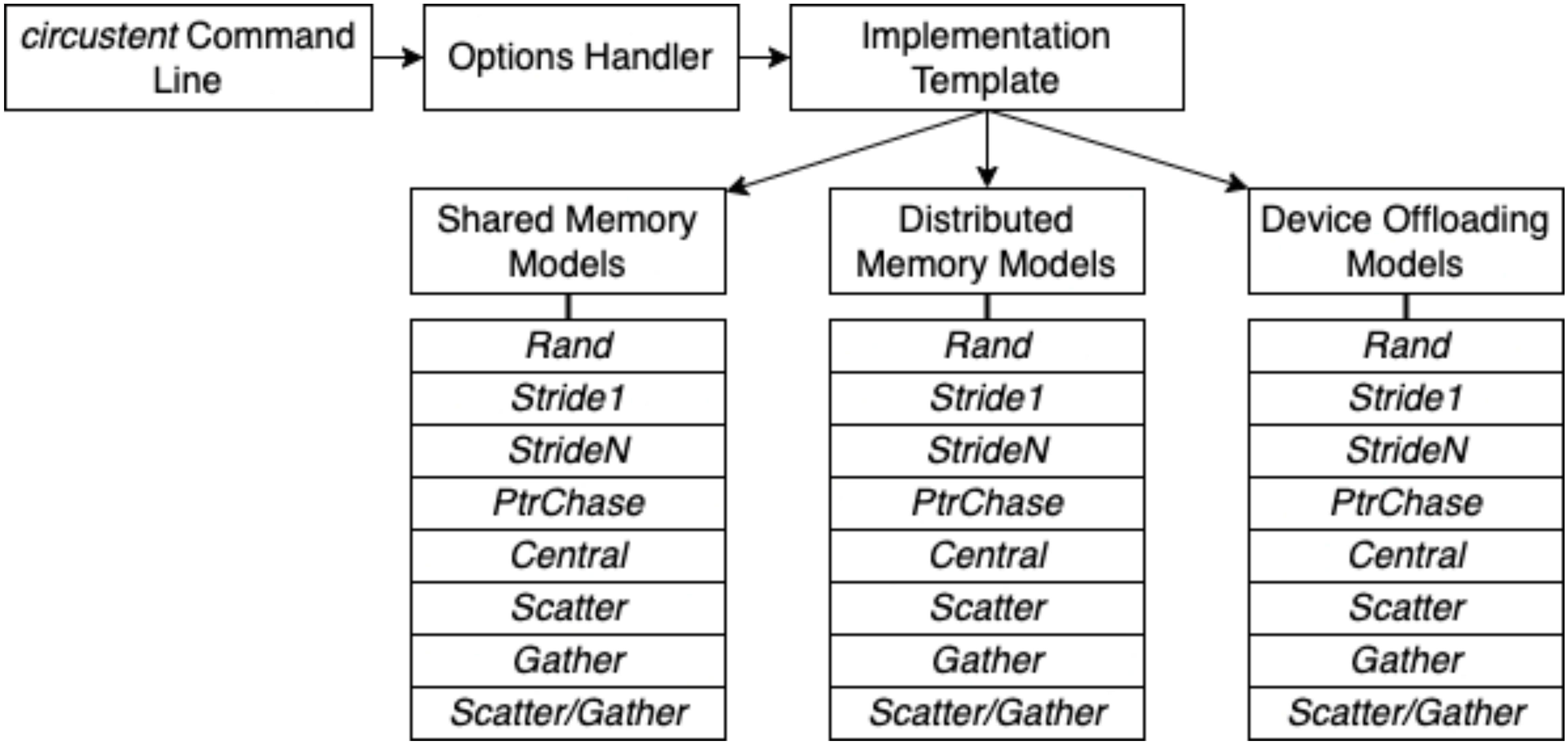
- Ability to derive normalized results
  - Benchmark results are important to the design of future systems
  - Difficult to directly compare performance of varied systems across distinct kernels
- Support for a multitude of programming models
  - Different workloads and systems utilize a variety of programming 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

Benchmark	AMOs Per Iteration
Rand	1
Stride-1	1
Stride-N	1
Pointer Chase	1
Central	1
Scatter	3
Gather	3
Scatter/Gather	4

$$GAMS = \frac{(PEs \times Iters \times AMOs\_Per\_Iter)/1e^9}{time}]$$


circustent Command Line
Options Handler
Implementation Template
Shared Memory Models
Distributed Memory Models
Device Offloading Models
Rand
Stride1
StrideN
PtrChase
Central
Scatter
Gather
Scatter/Gather

Williams, B., Leidel, J., Wang, X., Donofrio, D., Chen, Y.: Circustent: A bench- mark suite for atomic memory operations. In: The International Symposium on Memory Systems

Algorithm 1: Random Access Kernel

for $i \leftarrow 0$ **to** *iters* **by** 1 **do**

| AMO(VAL[IDX[i]])

end

Williams, B., Leidel, J., Wang, X., Donofrio, D., Chen, Y.: Circustent: A bench- mark suite for atomic memory operations. In: The International Symposium on Memory Systems

Algorithm 2: Stride-1 Kernel

for $i \leftarrow 0$ **to** $iters$ **by** 1 **do**
  | AMO(VAL[i])
end

Williams, B., Leidel, J., Wang, X., Donofrio, D., Chen, Y.: Circustent: A bench- mark suite for atomic memory operations. In: The International Symposium on Memory Systems

Algorithm 3: Stride-N Kernel

for $i \leftarrow 0$ **to** *iters* **by** *stride* **do**
  | AMO(VAL[i])
end

Williams, B., Leidel, J., Wang, X., Donofrio, D., Chen, Y.: Circustent: A bench- mark suite for atomic memory operations. In: The International Symposium on Memory Systems

Algorithm 4: Pointer Chase Kernel

```
for i ← 0 to iters by 1 do
|   start = AMO(IDX[start])
end
```

Williams, B., Leidel, J., Wang, X., Donofrio, D., Chen, Y.: Circustent: A bench- mark suite for atomic memory operations. In: The International Symposium on Memory Systems

Algorithm 5: Central Kernel

for $i \leftarrow 0$ **to** *iters* **by** 1 **do**
  AMO(VAL[0])
end

Williams, B., Leidel, J., Wang, X., Donofrio, D., Chen, Y.: Circustent: A bench- mark suite for atomic memory operations. In: The International Symposium on Memory Systems

Algorithm 6: Scatter Kernel

for $i \leftarrow 0$ **to** $iters$ **by** 1 **do**
    dest = AMO(IDX[i+1])
    val = AMO(VAL[i])
    AMO(VAL[dest], val) // VAL[dest] = val
end

Williams, B., Leidel, J., Wang, X., Donofrio, D., Chen, Y.: Circustent: A bench- mark suite for atomic memory operations. In: The International Symposium on Memory Systems

Algorithm 7: Gather Kernel

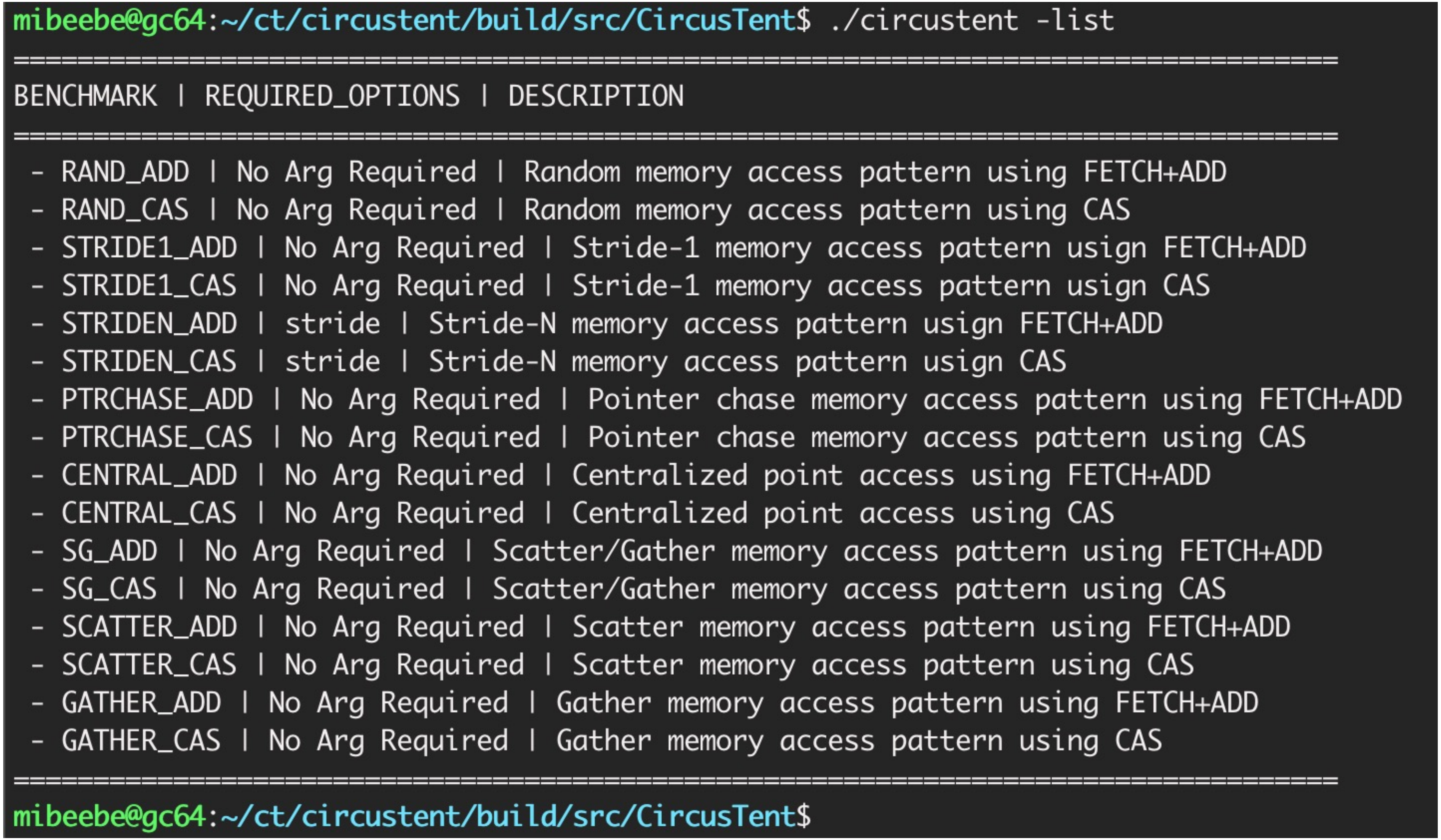
```
for i ← 0 to iters by 1 do
    src = AMO(IDX[i+1])
    val = AMO(VAL[src])
    AMO(VAL[i], val) // VAL[i] = val
end
```

Williams, B., Leidel, J., Wang, X., Donofrio, D., Chen, Y.: Circustent: A bench- mark suite for atomic memory operations. In: The International Symposium on Memory Systems

Algorithm 8: Scatter/Gather Kernel

```
for i ← 0 to iters by 1 do
    src = AMO(IDX[i])
    dest = AMO(IDX[i+1])
    val = AMO(VAL[src])
    AMO(VAL[dest], val) // VAL[dest] = val
end
```

```
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$ ./circustent -help
=======================================================================
 CircusTent Version 0.1
 Usage: circustent [OPTIONS]
=======================================================================
 -b|-bench|--bench TEST                     : Sets the benchmark to run
 -m|-memsize|--memsize BYTES                : Sets the size of the array
 -p|-pes|--pes PES                          : Sets the number of PEs
 -i|-iters|--iters ITERATIONS               : Sets the number of iterations per PE
 -s|-stride|--stride STRIDE (elems)         : Sets the stride in 'elems'
=======================================================================
 -h|-help|--help                            : Prints this help menu
 -l|-list|--list                            : List benchmarks
=======================================================================
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$
```


mibeebe@gc64:~/ct/circustent/build/src/CircusTent$ ./circustent -list
BENCHMARK | REQUIRED_OPTIONS | DESCRIPTION
- RAND_ADD | No Arg Required | Random memory access pattern using FETCH+ADD
- RAND_CAS | No Arg Required | Random memory access pattern using CAS
- STRIDE1_ADD | No Arg Required | Stride-1 memory access pattern usign FETCH+ADD
- STRIDE1_CAS | No Arg Required | Stride-1 memory access pattern usign CAS
- STRIDEN_ADD | stride | Stride-N memory access pattern usign FETCH+ADD
- STRIDEN_CAS | stride | Stride-N memory access pattern usign CAS
- PTRCHASE_ADD | No Arg Required | Pointer chase memory access pattern using FETCH+ADD
- PTRCHASE_CAS | No Arg Required | Pointer chase memory access pattern using CAS
- CENTRAL_ADD | No Arg Required | Centralized point access using FETCH+ADD
- CENTRAL_CAS | No Arg Required | Centralized point access using CAS
- SG_ADD | No Arg Required | Scatter/Gather memory access pattern using FETCH+ADD
- SG_CAS | No Arg Required | Scatter/Gather memory access pattern using CAS
- SCATTER_ADD | No Arg Required | Scatter memory access pattern using FETCH+ADD
- SCATTER_CAS | No Arg Required | Scatter memory access pattern using CAS
- GATHER_ADD | No Arg Required | Gather memory access pattern using FETCH+ADD
- GATHER_CAS | No Arg Required | Gather memory access pattern using CAS
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$

```
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$ ./circustent -b RAND_ADD -m 16488974000 -p 24 -i 10000000
RUNNING WITH NUM_THREADS = 24
=================================================
 Timing (secs)        : 3.14958
 Giga AMOs/sec (GAMS) : 0.0762006
=================================================
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$ ./circustent -b RAND_ADD -m 16488974000 -p 24 -i 20000000
RUNNING WITH NUM_THREADS = 24
=================================================
 Timing (secs)        : 6.21284
 Giga AMOs/sec (GAMS) : 0.0772594
=================================================
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$ ./circustent -b RAND_ADD -m 16488974000 -p 24 -i 30000000
RUNNING WITH NUM_THREADS = 24
=================================================
 Timing (secs)        : 9.27618
 Giga AMOs/sec (GAMS) : 0.0776181
=================================================
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$ ./circustent -b RAND_ADD -m 16488974000 -p 24 -i 40000000
RUNNING WITH NUM_THREADS = 24
=================================================
 Timing (secs)        : 12.3172
 Giga AMOs/sec (GAMS) : 0.07794
=================================================
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$ ./circustent -b RAND_ADD -m 16488974000 -p 24 -i 50000000
RUNNING WITH NUM_THREADS = 24
=================================================
 Timing (secs)        : 15.4594
 Giga AMOs/sec (GAMS) : 0.0776229
=================================================
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$
```

```
mibeebe@gc64:~/ct/circustent/build/src/CircusTent$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                12
On-line CPU(s) list:   0-11
Thread(s) per core:    2
Core(s) per socket:    6
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 63
Model name:            Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz
Stepping:              2
CPU MHz:               1201.406
CPU max MHz:           3200.0000
CPU min MHz:           1200.0000
BogoMIPS:              4794.48
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              15360K
NUMA node0 CPU(s):     0-11
```

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

- Williams, B., Leidel, J., Wang, X., Donofrio, D., Chen, Y.: Circustent: A bench- mark suite for atomic memory operations. In: The International Symposium on Memory Systems. p. 144–157. MEMSYS 2020, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3422575.3422789, https://doi.org/10.1145/3422575.3422789
- Wang, X., Leidel, J.D., Williams, B., Ehret, A., Mark, M., Kinsy, M.A., Chen, Y.: xbgas: A global address space extension on risc-v for high performance computing. In: 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS). pp. 454–463 (2021). https://doi.org/10.1109/IPDPS49936.2021.00054
- Tudor David, Rachid Guerraoui, and Vasileios Trigonakis. 2013. Everything You Always Wanted to Know about Synchronization but Were Afraid to Ask. In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP ’13). Association for Computing Machinery, New York, NY, USA, 33–48. https://doi.org/10.1145/2517349.2522714
- Message Passing Interface Forum. 2012. MPI: A Message-Passing Interface Standard Version 3.0. Chapter author for Collective Communication, Process Topologies, and One Sided Communications.
- The GAP Benchmark Suite, Scott Beamer, Krste Asanović, and David Patterson, arXiv:1508.03619 [cs.DC], 2015.