gocachemark

Benchmark report for Go cache implementations. View on GitHub →

Generated: 2025-12-08 18:06:47 EST **Machine:** darwin/arm64 (16 CPUs) **Go Version:** go1.25.4

Command: gocachemark -all

Overall Winners

Ranked voting across all benchmarks

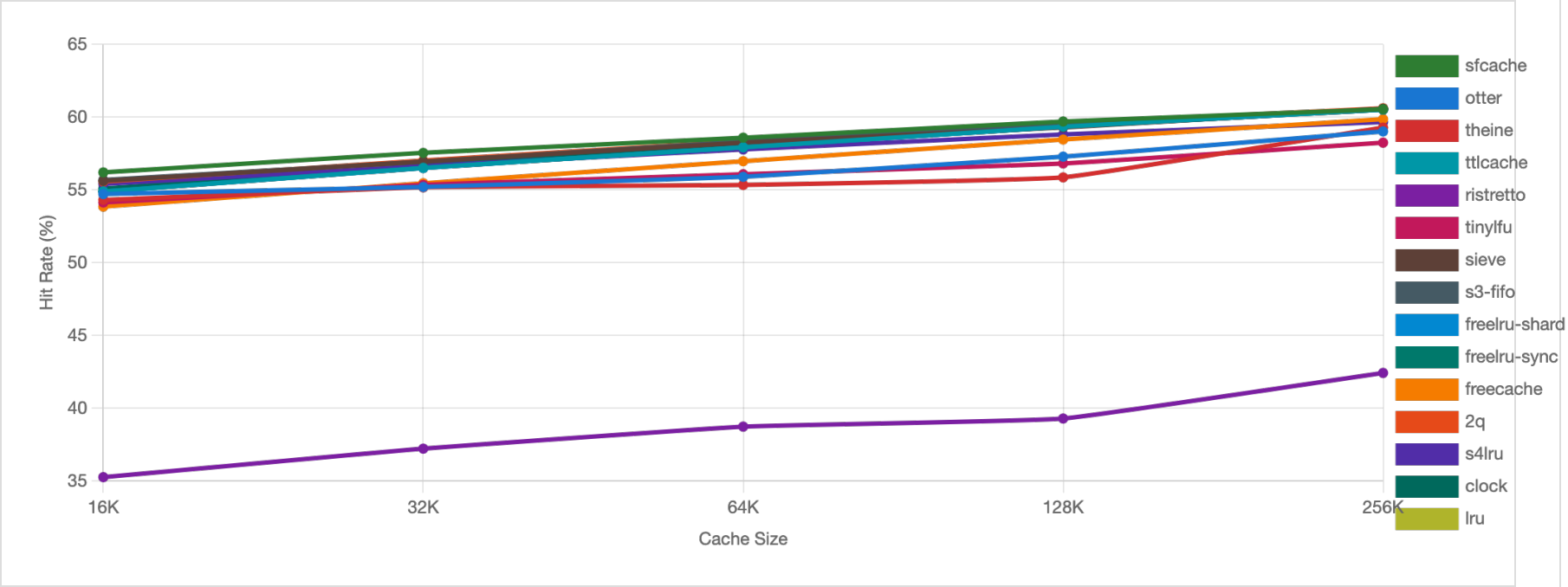
Table of Contents

Hit Rate Benchmarks

> Higher hit rates mean better cache effectiveness. Tests measure how often requested items are found in the cache across different cache sizes.

CDN Production Trace

> 2M operations, ~768K unique keys



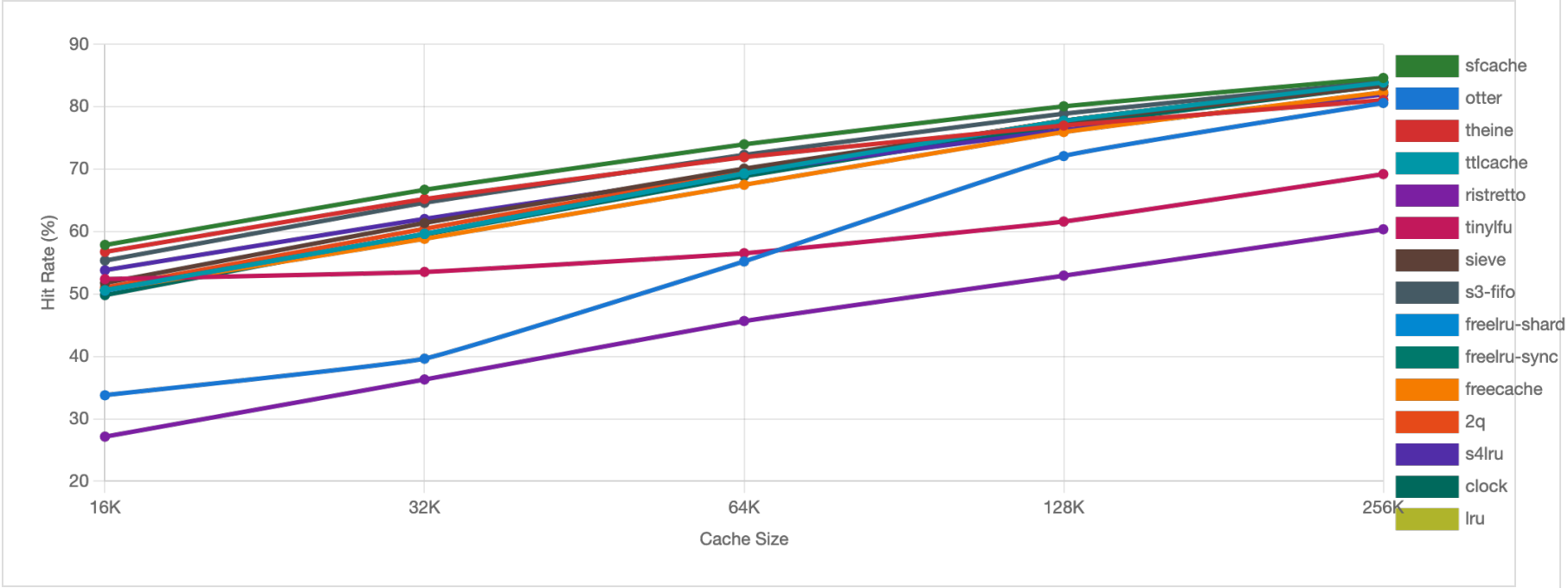
Results Table

Cache	16K	32K	64K	128K	256K	Avg
sfcache	56.21%	57.55%	58.59%	59.70%	60.55%	**58.52%**
s3-fifo	55.69%	56.95%	58.33%	59.61%	60.55%	**58.23%**
2q	55.57%	57.02%	58.35%	59.56%	60.61%	**58.22%**
sieve	55.63%	56.95%	58.25%	59.30%	60.59%	**58.14%**
clock	55.09%	56.68%	58.14%	59.42%	60.57%	**57.98%**
ttlcache	54.90%	56.52%	57.94%	59.35%	60.53%	**57.85%**
freelru-sync	54.90%	56.52%	57.94%	59.35%	60.53%	**57.85%**
lru	54.90%	56.52%	57.94%	59.35%	60.53%	**57.85%**

Cache	16K	32K	64K	128K	256K	Avg
freelru-shard	54.89%	56.51%	57.94%	59.34%	60.53%	**57.84%**
s4lru	55.42%	56.64%	57.79%	58.82%	59.67%	**57.67%**
freecache	53.86%	55.46%	56.98%	58.47%	59.87%	**56.93%**
otter	54.72%	55.21%	55.91%	57.29%	59.02%	**56.43%**
tinylfu	54.14%	55.35%	56.08%	56.82%	58.25%	**56.13%**
theine	54.32%	55.19%	55.35%	55.87%	59.29%	**56.01%**
ristretto	35.27%	37.23%	38.74%	39.30%	42.43%	**38.59%**

Meta KVCache Production Trace

> 5M operations



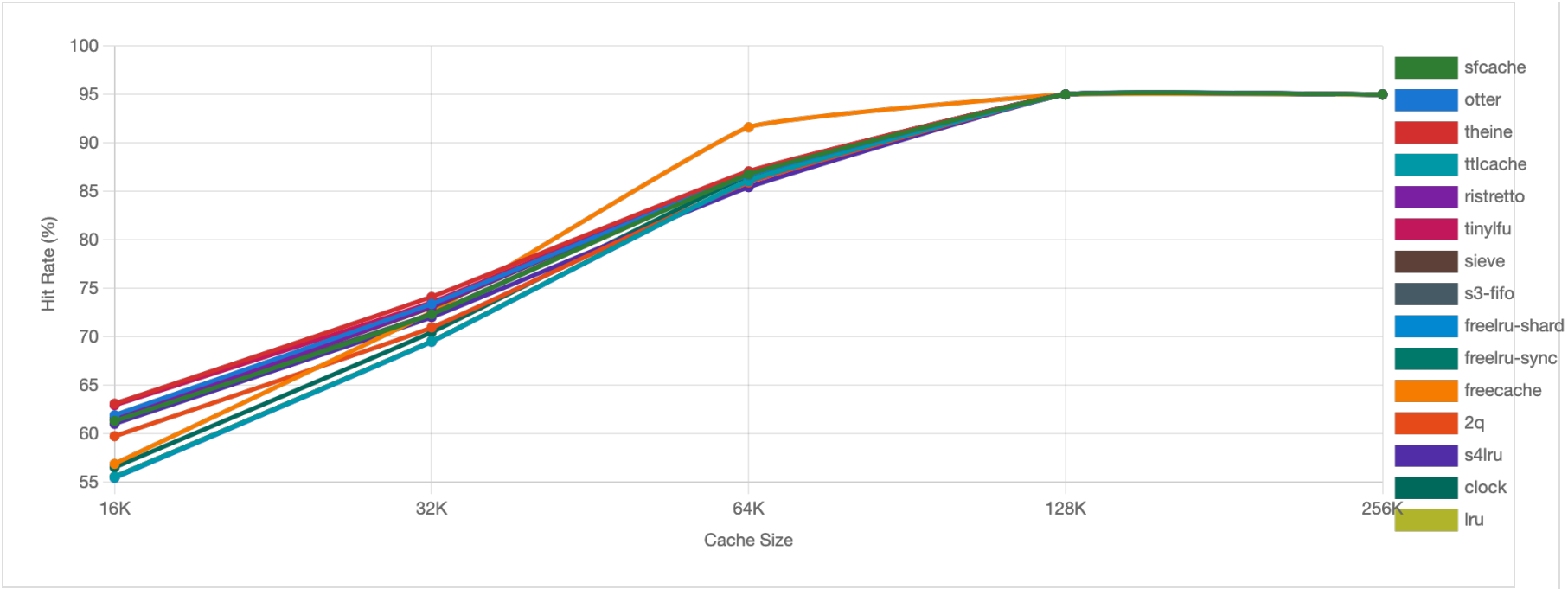
Results Table

Cache	16K	32K	64K	128K	256K	Avg
sfcache	57.87%	66.71%	74.00%	80.09%	84.64%	72.66%
s3-fifo	55.36%	64.63%	72.30%	78.90%	84.07%	71.05%
theine	56.76%	65.23%	71.94%	77.01%	81.10%	70.41%
sieve	51.77%	61.38%	70.09%	77.61%	83.36%	68.84%
s4lru	53.82%	62.04%	69.59%	76.27%	81.97%	68.74%
2q	51.04%	60.45%	69.58%	77.77%	83.83%	68.53%
ttlcache	50.63%	59.64%	69.34%	77.77%	83.90%	68.26%
freelru-sync	50.63%	59.64%	69.34%	77.77%	83.90%	68.26%

Cache	16K	32K	64K	128K	256K	Avg
lru	50.63%	59.64%	69.34%	77.77%	83.90%	**68.26%**
freelru-shard	50.54%	59.65%	69.30%	77.73%	83.89%	**68.22%**
clock	49.82%	59.51%	68.87%	76.99%	83.41%	**67.72%**
freecache	50.61%	58.85%	67.51%	75.96%	82.33%	**67.05%**
tinylfu	52.40%	53.54%	56.54%	61.62%	69.22%	**58.66%**
otter	33.79%	39.64%	55.23%	72.11%	80.61%	**56.28%**
ristretto	27.15%	36.31%	45.67%	52.95%	60.39%	**44.49%**

Zipf Synthetic Trace

> alpha=0.8, 2M operations, 100K keyspace



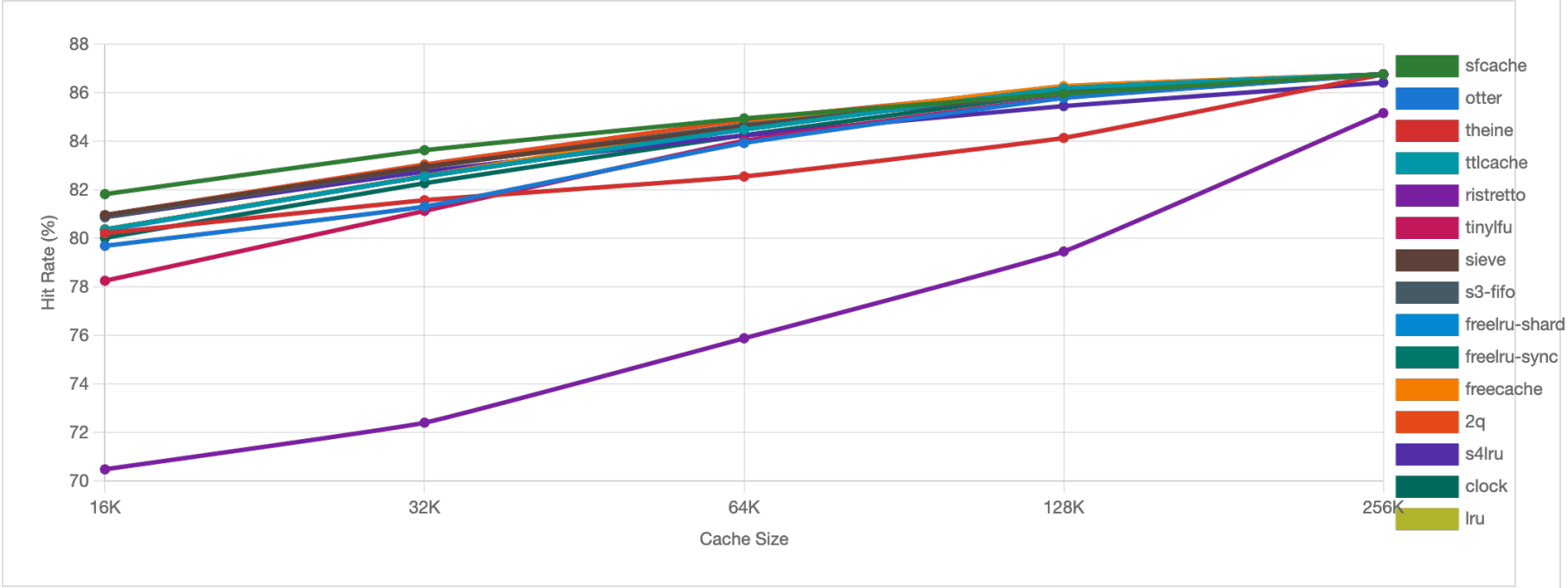
Results Table

Cache	16K	32K	64K	128K	256K	Avg
theine	63.11%	74.13%	87.08%	95.01%	95.01%	82.87%
tinylfu	62.96%	73.57%	86.92%	95.01%	95.01%	82.70%
otter	61.93%	73.40%	86.66%	95.01%	95.01%	82.40%
sieve	61.56%	73.07%	87.00%	95.01%	95.01%	82.33%
ristretto	61.44%	73.16%	87.01%	95.01%	95.00%	82.33%
freecache	56.91%	72.56%	91.63%	95.01%	95.01%	82.23%
s3-fifo	61.74%	72.40%	86.91%	95.01%	95.01%	82.22%
sfcache	61.33%	72.34%	86.78%	95.01%	95.01%	82.09%

Cache	16K	32K	64K	128K	256K	Avg
s4lru	61.04%	72.03%	85.44%	95.01%	95.01%	**81.71%**
2q	59.77%	70.97%	85.93%	95.01%	95.01%	**81.34%**
clock	56.54%	70.48%	86.60%	95.01%	95.01%	**80.73%**
ttlcache	55.57%	69.55%	86.11%	95.01%	95.01%	**80.25%**
freelru-sync	55.57%	69.55%	86.11%	95.01%	95.01%	**80.25%**
lru	55.57%	69.55%	86.11%	95.01%	95.01%	**80.25%**
freelru-shard	55.47%	69.50%	86.07%	95.01%	95.01%	**80.21%**

Twitter Production Cache Trace

> 2M operations, cluster001+cluster052



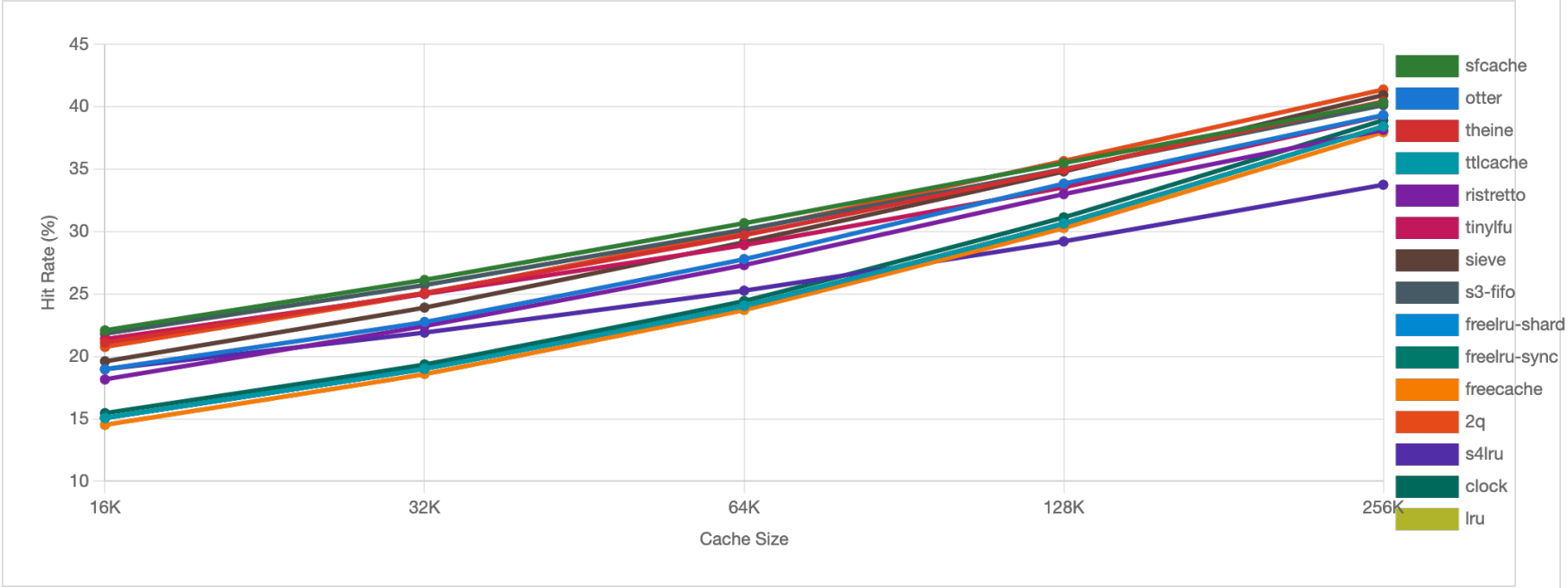
Results Table

Cache	16K	32K	64K	128K	256K	Avg
sfcache	81.83%	83.64%	84.95%	85.97%	86.77%	**84.63%**
2q	80.97%	83.05%	84.88%	86.17%	86.77%	**84.37%**
sieve	80.97%	82.96%	84.60%	86.08%	86.77%	**84.28%**
s3-fifo	80.89%	82.93%	84.68%	85.97%	86.77%	**84.25%**
freecache	80.38%	82.58%	84.71%	86.28%	86.77%	**84.14%**
ttlcache	80.37%	82.56%	84.50%	86.18%	86.77%	**84.08%**
freelru-sync	80.37%	82.56%	84.50%	86.18%	86.77%	**84.08%**
lru	80.37%	82.56%	84.50%	86.18%	86.77%	**84.08%**

Cache	16K	32K	64K	128K	256K	Avg
freelru-shard	80.34%	82.56%	84.49%	86.18%	86.77%	**84.07%**
s4lru	80.88%	82.76%	84.25%	85.46%	86.43%	**83.96%**
clock	80.03%	82.28%	84.26%	86.00%	86.77%	**83.87%**
otter	79.70%	81.32%	83.94%	85.79%	86.77%	**83.50%**
tinylfu	78.26%	81.14%	84.03%	85.82%	86.77%	**83.21%**
theine	80.22%	81.58%	82.56%	84.15%	86.77%	**83.06%**
ristretto	70.49%	72.41%	75.89%	79.47%	85.17%	**76.69%**

Wikipedia CDN Upload Trace

> 2M operations, upload.wikimedia.org



Results Table

Cache	16K	32K	64K	128K	256K	Avg
sfcache	22.09%	26.13%	30.67%	35.50%	40.32%	**30.94%**
2q	20.76%	25.03%	30.05%	35.66%	41.38%	**30.58%**
s3-fifo	21.83%	25.73%	30.17%	35.01%	40.14%	**30.57%**
theine	21.11%	25.10%	29.72%	34.97%	40.43%	**30.27%**
sieve	19.61%	23.92%	29.16%	34.83%	40.95%	**29.69%**
tinylfu	21.39%	25.01%	28.92%	33.56%	39.31%	**29.64%**
otter	18.98%	22.76%	27.80%	33.84%	39.34%	**28.54%**
ristretto	18.16%	22.43%	27.32%	33.00%	38.14%	**27.81%**

Cache	16K	32K	64K	128K	256K	Avg
clock	15.46%	19.36%	24.45%	31.15%	38.93%	**25.87%**
s4lru	18.97%	21.91%	25.27%	29.22%	33.75%	**25.82%**
ttlcache	15.08%	19.02%	24.07%	30.66%	38.43%	**25.45%**
freelru-sync	15.08%	19.02%	24.07%	30.66%	38.43%	**25.45%**
lru	15.08%	19.02%	24.07%	30.66%	38.43%	**25.45%**
freelru-shard	15.06%	19.00%	24.04%	30.65%	38.42%	**25.44%**
freecache	14.52%	18.59%	23.73%	30.29%	37.97%	**25.02%**

Latency Benchmarks (Single-Threaded)

> Lower latency means faster operations. Measures the time (in nanoseconds) for individual Get, Set, and SetEvict operations in a single-threaded environment.

String Keys

Results

Cache	Get (ns)	Get allocs	Set (ns)	Set allocs	SetEvict (ns)	SetEvict allocs	Avg (ns)
sfcache	13.0	0	52.0	0	131.0	0	32.5

Cache	Get (ns)	Get allocs	Set (ns)	Set allocs	SetEvict (ns)	SetEvict allocs	Avg (ns)
s4lru	18.0	0	63.0	1	65.0	1	40.5
clock	19.0	0	22.0	0	107.0	2	20.5
freelru-sync	24.0	0	24.0	0	36.0	0	24.0
freelru-shard	24.0	0	37.0	0	47.0	0	30.5
sieve	26.0	0	45.0	0	196.0	3	35.5
lru	27.0	0	28.0	0	100.0	1	27.5
s3-fifo	30.0	0	47.0	0	356.0	4	38.5
2q	31.0	0	30.0	0	239.0	1	30.5
otter	34.0	0	141.0	1	170.0	1	87.5
ristretto	58.0	1	268.0	3	116.0	4	163.0
theine	82.0	1	118.0	0	208.0	0	100.0
tinylfu	84.0	0	107.0	3	117.0	3	95.5
freecache	84.0	1	57.0	0	105.0	0	70.5
ttlcache	409.0	0	419.0	0	384.0	3	414.0

Int Keys

Results

Cache	Get (ns)	Get	Set (ns)	Set	SetEvict (ns)	SetEvict	Avg

		allocs		allocs		allocs	(ns)
freelru-shard	7.0	0	16.0	0	18.0	0	11.5
sfcache	9.0	0	24.0	0	97.0	0	16.5
otter	35.0	0	135.0	1	164.0	1	85.0
theine	78.0	1	110.0	0	177.0	0	94.0
ttlcache	428.0	0	419.0	0	354.0	3	423.5

String Keys - GetOrSet

> GetOrSet is an atomic operation that gets a value if it exists, or sets it if it doesn't. Only caches that support this operation are shown.

GetOrSet Results

Cache	GetOrSet (ns)	GetOrSet allocs
otter	54.0	1
sfcache	78.0	0
freecache	86.0	1
ttlcache	448.0	0

Int Keys - GetOrSet

> GetOrSet latency for integer keys. Only caches that support this operation are shown.

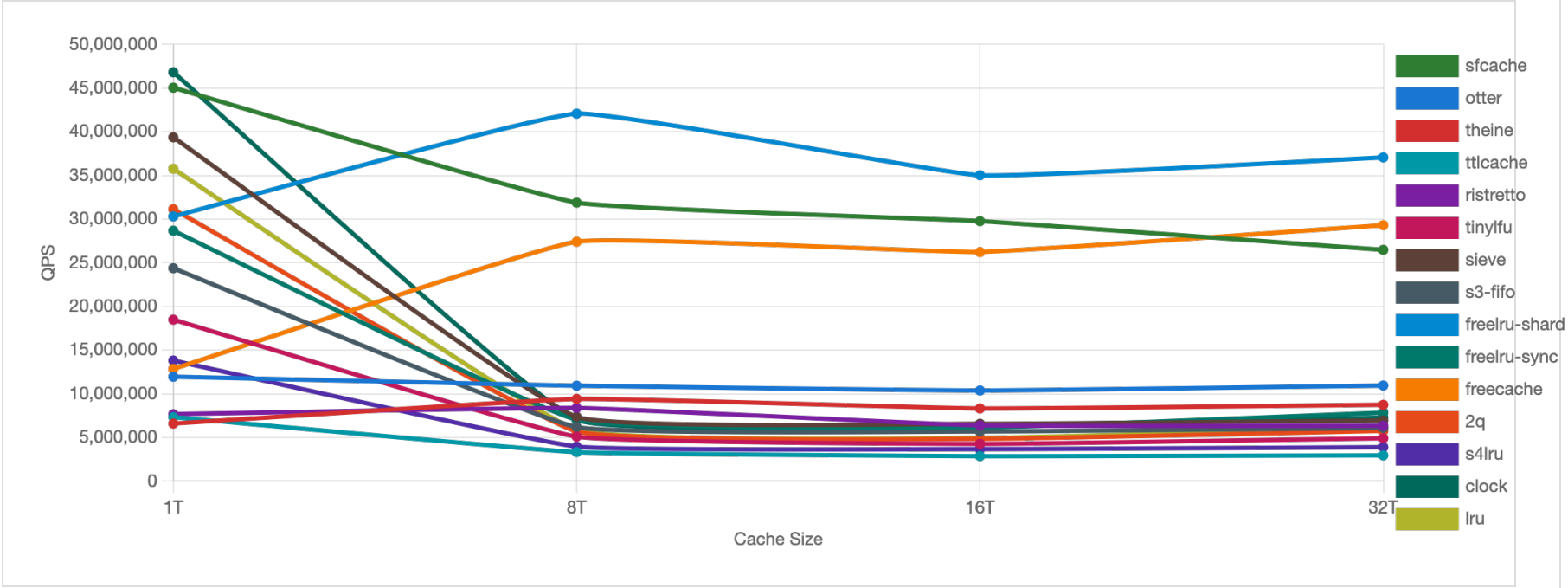
GetOrSet Results

Cache	GetOrSet (ns)	GetOrSet allocs
sfcache	10.0	0
otter	45.0	1
ttlcache	441.0	0

Throughput Benchmarks (Multi-Threaded)

> Higher throughput means more queries per second (QPS). Tests measure concurrent performance with a Zipf workload (75% reads, 25% writes) across different thread counts.

String Keys

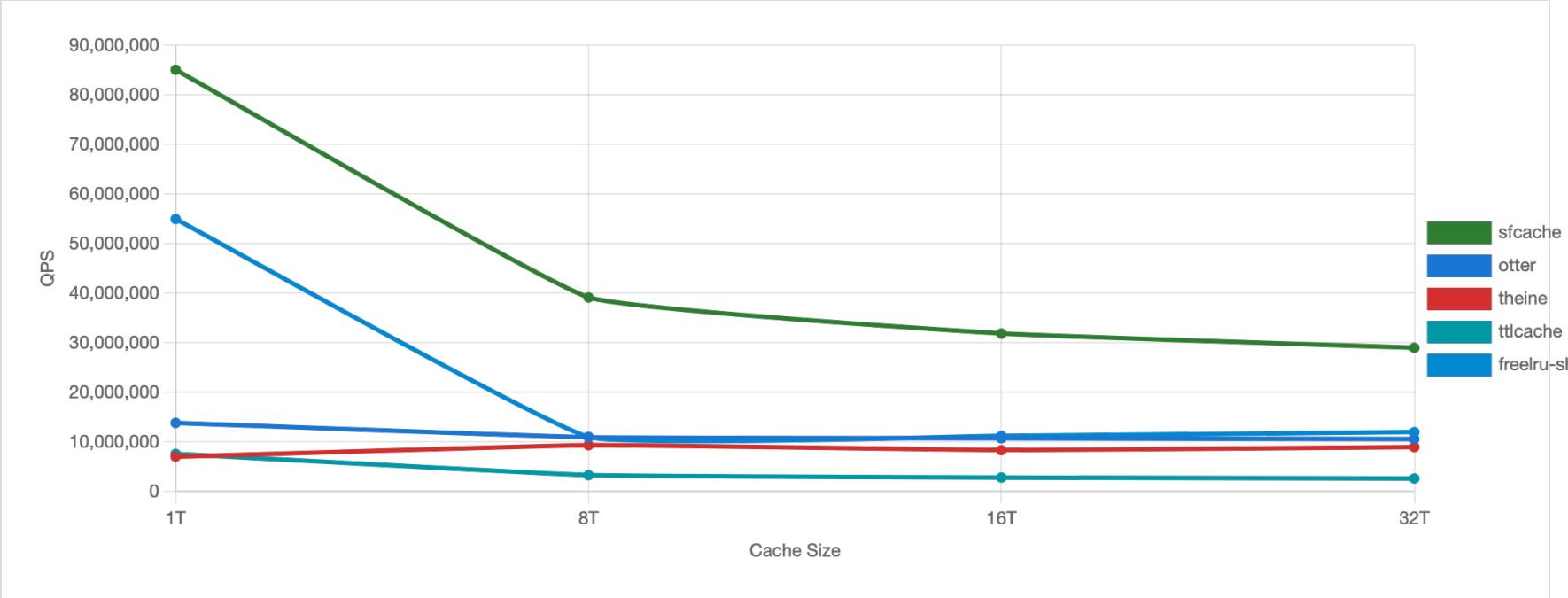


Results Table

Cache	1T	8T	16T	32T	Avg
freelru-shard	30.33M	42.09M	35.03M	37.07M	36.13M
sfcache	45.05M	31.90M	29.77M	26.49M	33.30M
freecache	12.85M	27.41M	26.24M	29.30M	23.95M
clock	46.82M	7.05M	6.41M	7.30M	16.90M
sieve	39.37M	7.37M	6.58M	6.96M	15.07M
lru	35.76M	5.86M	4.94M	6.10M	13.17M
freelru-sync	28.68M	7.05M	6.13M	7.86M	12.43M
2q	31.13M	5.70M	4.84M	5.75M	11.86M

Cache	1T	8T	16T	32T	Avg
otter	11.97M	10.94M	10.38M	10.95M	**11.06M**
s3-fifo	24.39M	6.21M	5.70M	6.10M	**10.60M**
theine	6.60M	9.41M	8.32M	8.74M	**8.27M**
tinylfu	18.50M	5.11M	4.28M	4.92M	**8.20M**
ristretto	7.68M	8.39M	6.38M	6.34M	**7.20M**
s4lru	13.82M	3.98M	3.68M	3.90M	**6.35M**
ttlcache	7.33M	3.33M	2.87M	2.96M	**4.12M**

Int Keys



Results Table

Cache	1T	8T	16T	32T	Avg
sfcache	85.09M	39.17M	31.89M	29.01M	**46.29M**
freelru-shard	54.98M	11.04M	11.23M	12.01M	**22.31M**
otter	13.83M	10.93M	10.73M	10.58M	**11.52M**
theine	7.01M	9.34M	8.37M	8.96M	**8.42M**
ttlcache	7.58M	3.30M	2.81M	2.63M	**4.08M**

Memory Overhead Benchmarks

> Capacity: 32768 items, Value size: 1024 bytes. Measured in isolated processes. Overhead compared to baseline `map[string][]byte`. Lower is better.

Memory Results

Cache	Items Stored	Memory (MB)	Overhead vs map (bytes/item)
freelru-sync	32768	35.19	-20
otter	32768	36.28	14
freelru-shard	31606	36.47	21
clock	32768	37.03	38

Cache	Items Stored	Memory (MB)	Overhead vs map (bytes/item)
sfcache	32667	37.17	43
2q	32768	37.50	53
lru	32768	37.52	54
s4lru	32768	38.27	78
theine	32768	38.47	84
sieve	32768	39.53	118
s3-fifo	32768	39.53	118
tinylfu	32767	39.83	128
ttlcache	32768	40.82	159
ristretto	32768	41.47	180
freecache	32764	44.77	286