Accelerating Memcached Using Solarflare's Flareon™ Ultra server I/O adapter and OpenOnload® Kernel-Bypass TCP/IP stack

SolarflareWhitePaper

Introduction

Memcached is an open source, distributed, high-performance memory object caching system storing key-value pairs. It is designed to accelerate web applications by providing an object cache between a front-end web server and back-end database, for example. When run on a single Solarflare Flareon™ Ultra server I/O adapter leveraging Solarflare's OpenOnload® kernel-bypass TCP/IP stack in a dual-CPU Intel Ivy Bridge server (10 cores/CPU), the following headline performance increases were obtained with the Solarflare adapter compared to running on Intel's X520-DA2 10GbE adapter in the same system:

- Maximum batched query throughput (*get*) was **21.9 Mops***, a factor of 3 increase (200% faster).
- Maximum batched mixed throughput (*get/set* in a ratio of 9:1) was **13.5 Mops***, a factor of 2.1 increase (110% faster).

Solarflare's OpenOnload on Flareon Ultra Server I/O Adapters

Solarflare's OpenOnload is a Linux-based, open source, high-performance application accelerator that delivers lower and more predictable latency and higher message rates for TCP- and UDP-based applications. OpenOnload is ideal for applications that benefit from lower latency (with decreased jitter) and higher throughput such as, but by no means limited to, Memcached.

The latest OpenOnload release adds a number of new features including the ability to move sockets programmatically between OpenOnload stacks. This allows a significant potential performance boost for multi-threaded server applications, such as Memcached, that are architected with a single listener thread and multiple worker threads. Each worker thread can benefit from OpenOnload's ability to instantiate a dedicated stack per thread, with each stack having a direct path to the adapter.

Solarflare's Flareon Ultra server I/O adapters deliver industry-leading message rates with lowest latency and jitter over standard Ethernet along with low CPU utilization, enabling the industry's best performance and scalability for enterprise data center environments. They enable the deployment of more services to more users to fully leverage multi-core CPUs by providing hardware-assisted features to efficiently distribute I/O processing workloads, which eliminate bottlenecks and optimize CPU utilization. They also provide application compatibility and protocol compliance, bypassing kernel and networking overheads, while featuring binary compatibility with standard APIs and applications.

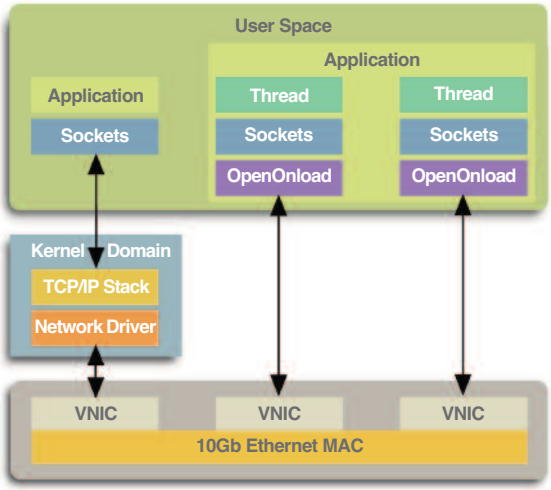

Figure 1. OpenOnload leveraging Flareon Ultra. Two OpenOnload Stacks/VNICs shown; 1024 available.

** Million Operations per Second*

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

They are specifically designed to complement OpenOnload by providing powerful packet switching functionality allowing incoming packets to be steered in adapter firmware directly to the OpenOnload stack to which they are destined. This architecture increases application processing efficiency significantly by removing potential bottlenecks and allowing each thread's stack to receive only traffic destined for its associated thread.

Server Configuration

Two Dell PowerEdge R620 Servers – each equipped with two 10-core Intel E5-2660 v2 processors and 32GB of RAM running Red Hat Enterprise Linux Server release 6.5 – were used. Hyper-threading was enabled on both. One server ran memslap configured to generate load on the other server running Memcached. The server running memslap utilized two Solarflare SFN6122F Onload server adapters while the other server running Memcached was tested in turn with an Intel® Ethernet Server Bypass Adapter X520-DA2 and a Solarflare Flareon Ultra server I/O SFN7122F adapter. In both configurations, the servers were connected together back-to-back from a single port on each SFN6122F adapter into both 10GbE ports on the X520-DA2/ SFN7122F; no switch was used.

All system threads and interrupts were bound to logical CPU 0 (first core on first CPU).

Memcached Architecture

Given Hyper-threading was enabled, 20 virtual cores were available per CPU for a total of 40 virtual cores. Within Memcached, when performing set operations with more than a single thread enabled, concurrency locks are used to maintain database consistency. To balance performance and scalability, more than 5 threads per Memcached instance are not recommended. Testing with various configurations confirmed this and the optimal Memcached configuration, on a Hyper-threaded CPU, was found to be 10 threads running on 5 physical/10 Hyper-threaded cores.

As 20 virtual cores were available, two Memcached instances were run per CPU. When running tests using both CPUs, four Memcached instances were run; two per CPU. Scalability tests were performed with each instance configured with a variable number of threads as per the following table:

CPUs Used	Total Memcached Instances	Physical CPU Cores Used	Memcached Threads Configured per Instance	Total Memcached Threads Configured	Logical CPU Cores Used
1	2	2	2	4	4
1	2	4	4	8	8
1	2	6	6	12	12
1	2	8	8	16	16
1	2	10	10	20	20
2	4	4	4	8	8
2	4	8	8	16	16
2	4	12	12	24	24
2	4	16	16	32	32
2	4	20	20	40	40

Figure 2. Memcached thread configurations for test runs.

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

For each Memcached instance, listener threads were bound to the first core on the CPU running it. Worker threads were distributed across all cores evenly, i.e. one thread per logical core, two threads per physical core. In the tests using a single CPU, Memcached instances were run on CPU 0 in the Solarflare case and CPU 1 in the Intel case. This was done as the Solarflare adapter was plugged into a slot directly connected to CPU 0 while the Intel adapter was plugged into a slot directly connected to CPU 1.

In total, 160MB of memory was pre-allocated to the Memcached instances: 80MB per instance in the dual instance case, 40MB per instance in the quad instance case. For both the read-only (*get*) tests as well as for the mixed tests, 70% of this was pre-populated with key/value pairs. For the mixed tests, the remaining 30% was populated during the tests.

Memcached has a separate maintenance thread to manage growing the internal hash table associating keys with values. The operation of this thread was supressed by ensuring that a hash table large enough to accommodate all the key/value pairs used during the test was created prior to the test runs.

For the purposes of these tests, the only commands used were value retrieval given the associated key (*get*) and value modification (*set*). Two sets of tests were run. The first set of tests measured read-only (*get*) performance from the prepopulated values. The second set ran a mixed workload of value retrieval (*get*) and value modification (*set*) in the ratio of 9:1. Tests were run both with retrieval of a single value for a single key and in batched mode with 48 values for 48 keys. Key sizes used were 20 bytes and value sizes were also 20 bytes. These values were chosen to stress the maximum capacity of the system in operations per second without being network bandwidth constrained.

For the Intel test, Intel Ethernet Flow Director was configured to ensure that packets were delivered to the consuming core to provide maximum performance.

Memslap Configuration

Given Hyper-threading was enabled, 20 virtual cores were available per CPU. It was determined that 20 memslap instances would be run across both CPUs. Each memslap instance was configured with two threads for a total of 40 threads. Each thread instance was configured to open 24 persistent TCP connections to the Memcached instances via OpenOnload. The total number of connections was identical across all tests and consisted of 24 connections/thread x 40 threads, equaling 960 total connections.

In the dual Memcached instance test case, half the TCP connections went to one instance and the other half to the other instance. In the quad Memcached instance test case, a quarter of the TCP connections went to each instance. All tests were run with the same number of connections independent of the number of Memcached instances and worker threads per instance.

Each thread's connections were distributed evenly across both Ethernet interfaces such that half the connections traversed each interface.

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

Memcached *Get* Results

The purpose of this test was to measure pure retrieval performance. Each of the 960 client connections pre-populated the Memcached instance that it was connected to with 1024 key/value pairs for a total of 983,042 key/value pairs, split across either two (single CPU test) or four (dual CPU test) Memcached instances. All operations requested existing values using pre-existing keys via the *get* primitive. Key choice was random within the pre-populated universe.

In the single operation case, only a single *get* operation was issued per transaction. In the multiple operations per transaction case, a single transaction of 48 *get* operations was issued.

Throughput was calculated by dividing the total number of operations obtained during a 30-second test run by the length of the run.

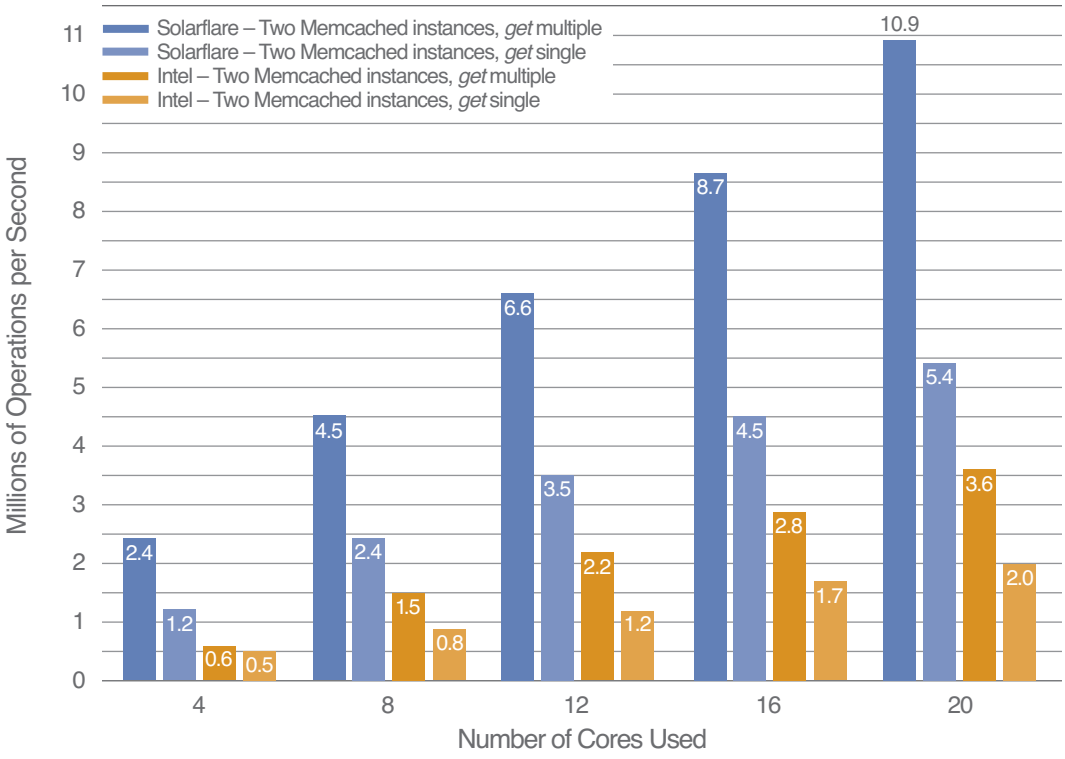

Figure 3. Single CPU throughput results for single *get* requests and batches of 48 *get* requests.

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

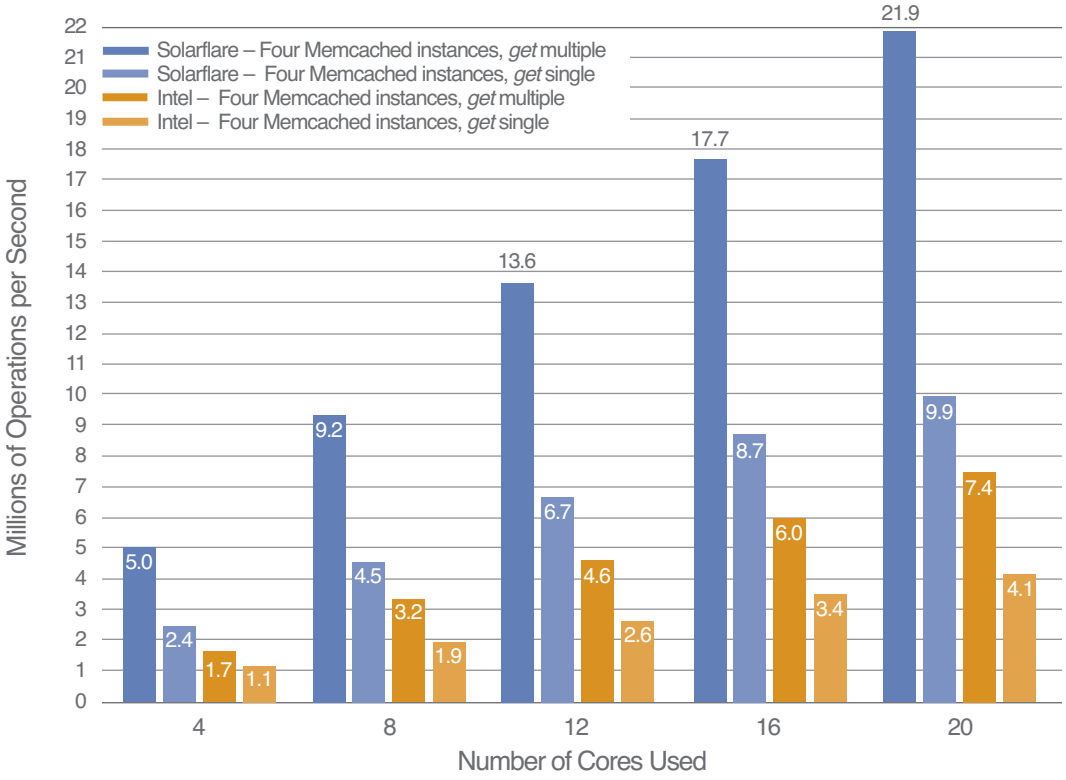

Figure 4. Dual CPU throughput results for single *get* requests and batches of 48 *get* requests.

Get Latency Results

These results were yielded from the same runs as the throughput tests. Latency was obtained by having memslap measure the time taken between issuing each transaction and obtaining the corresponding response. Memslap was specifically instrumented by Solarflare to do this. From each run, individual latency values were used to calculate the median and 99th percentile request/response latency.

N. B. Each latency value was measured per transaction, not per individual operation within a transaction.

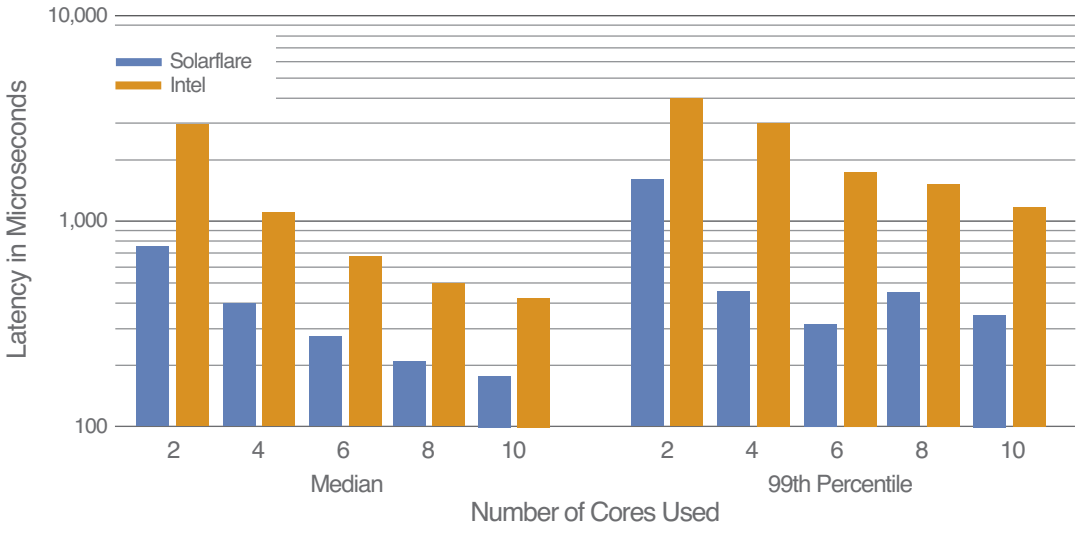

Figure 5. Single CPU latency results for single *get* requests.

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

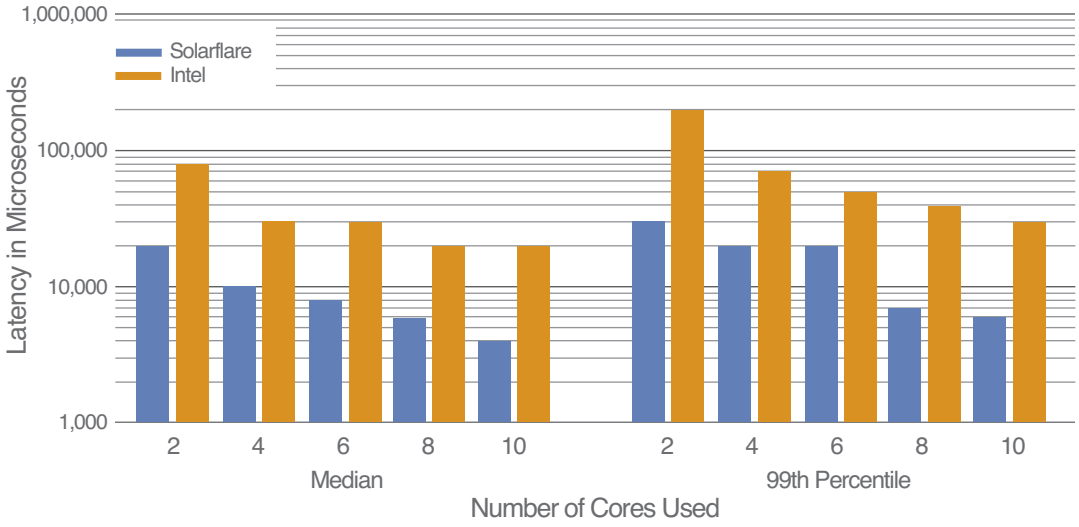

Figure 6. Single CPU latency results for batches of 48 *get* requests.

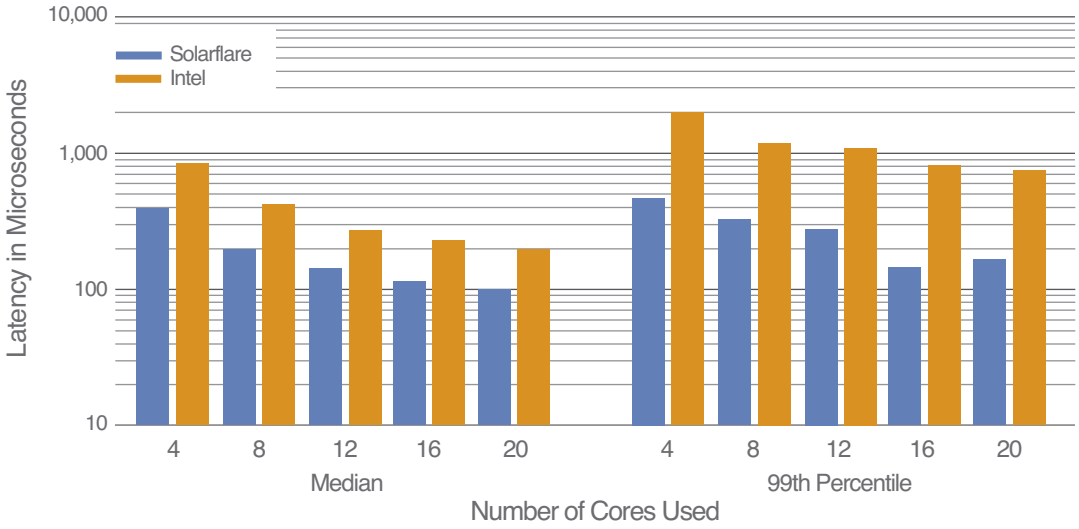

Figure 7. Dual CPU latency results for single *get* requests.

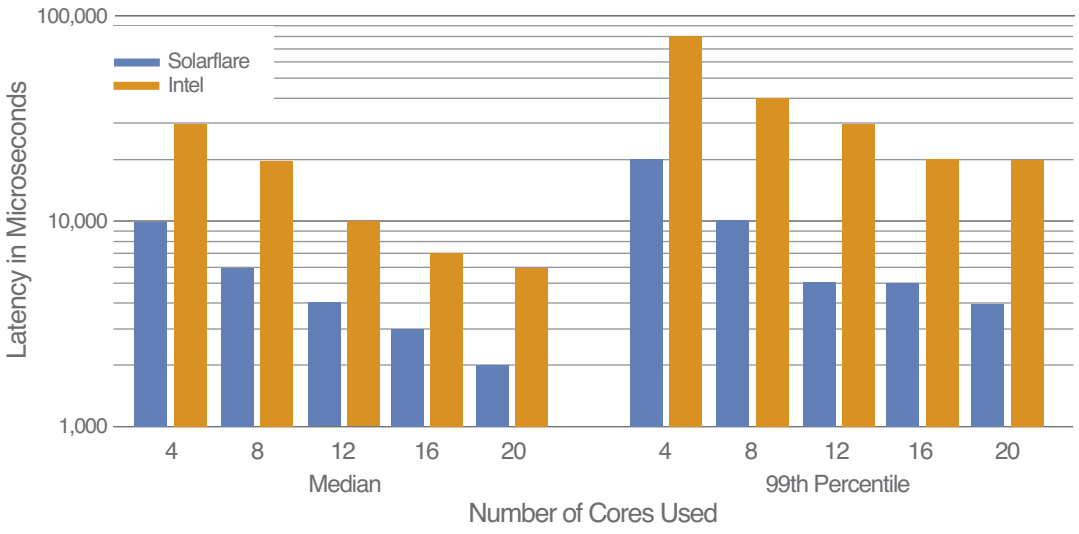

Figure 8. Dual CPU latency results for batches of 48 *get* requests.

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

Mixed *Get/Set* Results

This test is most representative of an actual deployment with the workload predominantly comprising retrieval of values interspersed with values being updated. A ratio of 9:1 was chosen, 10% of operations being a *set* operation, the remainder being *get* operations. As in the prior runs, the Memcached databases were pre-populated with 1024 key/value pairs by each memslap thread. For these tests however, each *set* operation overwrote the oldest of these 1024 entries.

In the single operation case where only a single operation was issued per transaction, each connection issued 9 randomly selected *get* operations followed by a single *set* operation and then repeated. In the multiple operations per transaction case, a single transaction of 48 *get* operations was followed by a number of single *set* operations (either 5 or 6) to ensure that on average, the 9:1 ratio was maintained.

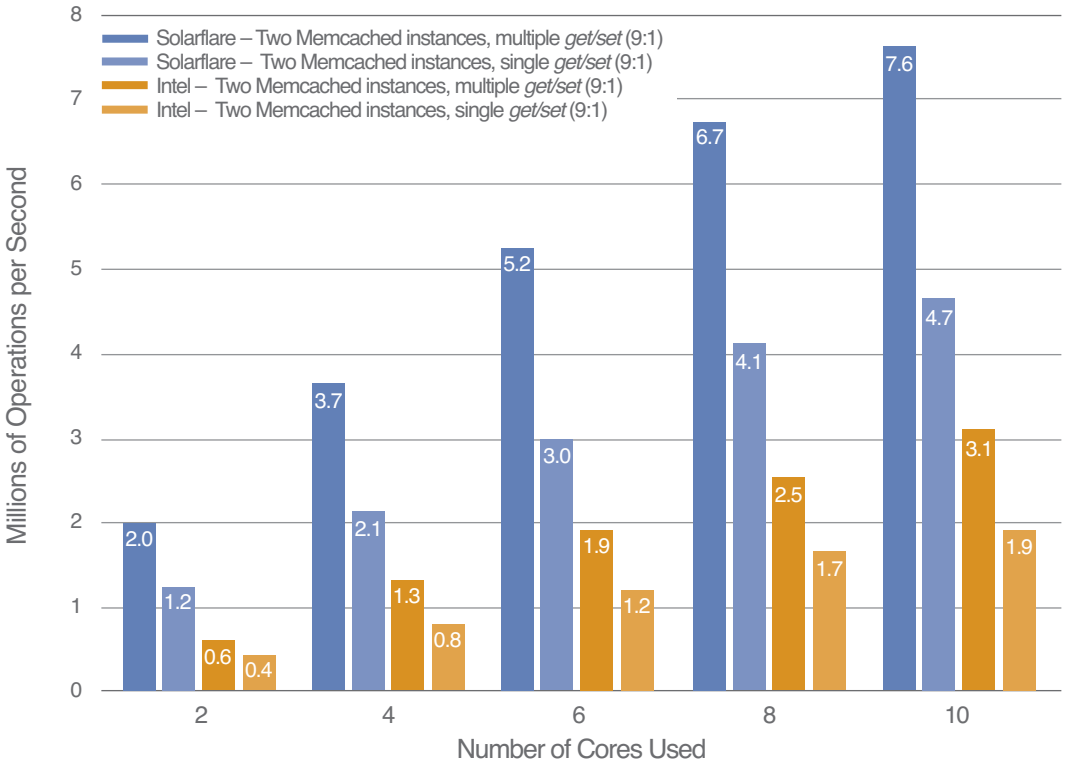

Figure 9. Single CPU mixed *get* and *set* requests in the ratio of 9:1 (*get:set*).

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

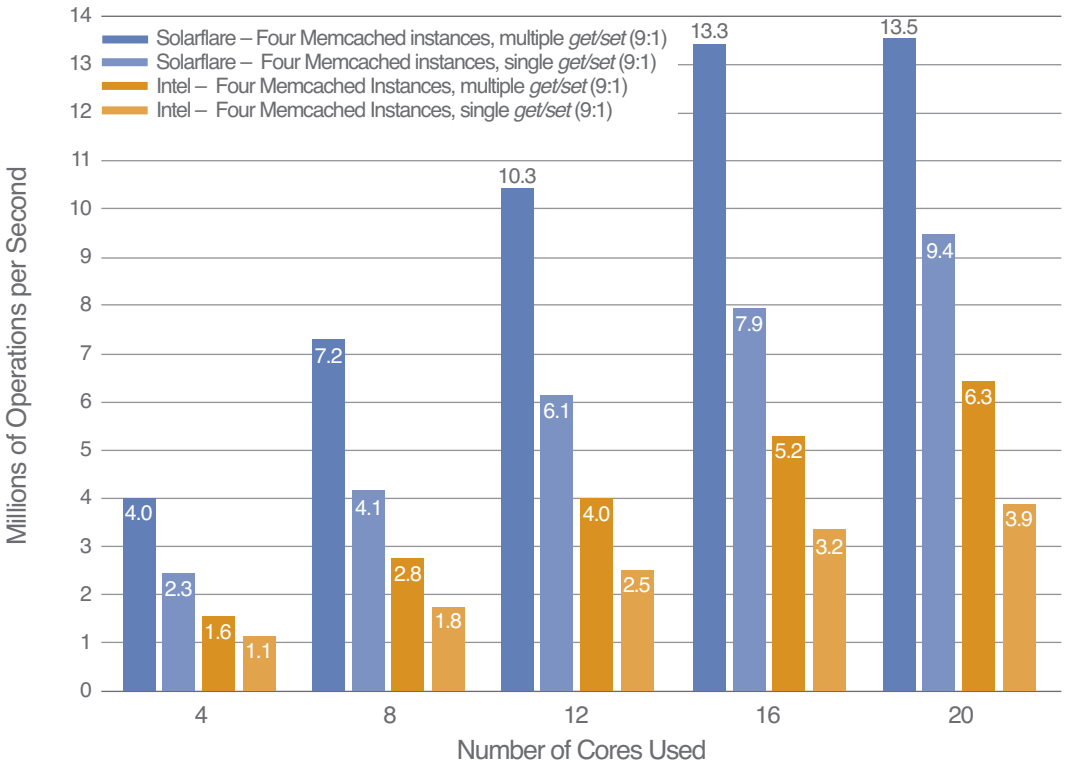

Figure 10. Dual CPU mixed *get* and *set* requests in the ratio of 9:1 (*get:set*).

Summary and Conclusions

Solarflare measured the performance of Memcached on a dual-CPU Ivy Bridge server using its own Flareon Ultra Server I/O Adapter SFN7122F and on Intel's x520-DA2 dual-port 10GbE server adapter.

Solarflare Performance Advantage

- The average system throughput improvement across all tests was a factor of 2.7.
- The lowest system throughput was a factor of 2.1 and the highest was a factor of 4.0.
- The average system latency reduction across all tests was 69.4%.
- The lowest latency reduction was 50% and the highest was 85%.

Translating this into practical terms, to match the performance of the Solarflare adapter with OpenOnload in a single server, between three and four equivalent servers would be required using Intel adapters.

Test Software

- A version of OpenOnload 201405-u1 was used with some as yet unreleased patches applied to fix some stability issues with the onload_move_fd() code which was new to this release.
- Intel's ixgbe-3.22.3 driver was used.
- Memcached 1.4.20 was modified as described above to call onload_move_fd(). Memslap was also modified to provide latency numbers as well as throughput.

A package containing the above software is available on demand to existing OpenOnload clients by sending an e-mail to: support@solarflare.com

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

SolarflareWhitePaper

Raw Throughput Numbers

		Single Operation per Transaction			Multiple Operations per Transaction (48)		
	Physical Cores Used	Solarflare	Intel	Factor Better Throughput	Solarflare	Intel	Factor Better Throughput
Single CPU *Get* (Mops)	2	1,215,710	450,534	2.7	2,449,984	615,924	4.0
	4	2,366,849	818,607	2.9	4,523,021	1,479,877	3.1
	6	3,465,582	1,215,681	2.9	6,623,495	2,186,840	3.0
	8	4,494,775	1,659,591	2.7	8,699,775	2,849,055	3.1
	10	5,357,072	2,042,140	2.6	10,866987	3,554,712	3.1
Dual CPU *Get* (Mops)	4	2,407,681	1,076,422	2.2	4,968,734	1,727,265	2.9
	8	4,549,616	1,854,242	2.5	9,249,270	3,236,525	2.9
	12	6,719,791	2,621,200	2.6	13,561,615	4,618,004	2.9
	16	8,728,365	3,419,506	2.6	17,690,756	6,033,233	2.9
	20	9,904,499	4,124,413	2.4	21,922,312	7,394,577	3.0
Single CPU Mixed *Get/Set* (9:1) (Mops)	2	1,157,047	448,624	2.6	2,028,781	605,871	3.3
	4	2,138,660	787,880	2.7	3,705,539	1,292,754	2.9
	6	3,037,058	1,159,893	2.6	5,203,493	1,933,115	2.7
	8	4,087,318	1,668,059	2.5	6,661,436	2,499,551	2.7
	10	4,738,219	1,923,718	2.5	7,607,754	3,070,689	2.5
Dual CPU Mixed *Get/Set* (9:1) (Mops)	4	2,266,809	1,058,190	2.1	4,047,791	1,550,256	2.6
	8	4,145,367	1,778,748	2.3	7,224,058	2,814,478	2.6
	12	6,097,402	2,480,687	2.5	10,269,060	3,980,335	2.6
	16	7,925,374	3,245,376	2.4	13,279,119	5,160,544	2.6
	20	9,421,963	3,912,335	2.4	13,519,899	6,300,232	2.1

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

SolarflareWhitePaper

Raw Latency Numbers

		Single Operation per Transaction			Multiple Operations per Transaction (48)		
	Physical Cores Used	Solarflare	Intel	Solarflare Latency Reduction	Solarflare	Intel	Solarflare Latency Reduction
Single CPU *Get* Median Latency (µs)	2	760	3,000	74.7%	20,000	80,000	75.0%
	4	400	1,100	63.6%	10,000	30,000	66.7%
	6	270	690	60.9%	8,000	30,000	73.3%
	8	210	510	58.8%	6,000	20,000	70.0%
	10	180	420	57.1%	4,000	20,000	80.0%
Dual CPU *Get* Median Latency (µs)	4	390	860	54.7%	10,000	30,000	66.7%
	8	210	420	50.0%	6,000	20,000	70.0%
	12	140	280	50.0%	4,000	10,000	60.0%
	16	110	240	54.2%	3,000	7,000	57.1%
	20	100	200	50.0%	2,000	6,000	66.7%
Single CPU *Get* 99th Percentile Latency (µs)	2	1,600	4,000	60.0%	30,000	200,000	85.0%
	4	460	3,000	84.7%	20,000	70,000	71.4%
	6	310	1,800	82.8%	20,000	50,000	60.0%
	8	460	1,600	71.3%	7,000	40,000	82.5%
	10	350	1,200	70.8%	6,000	30,000	80.0%
Dual CPU *Get* 99th Percentile Latency (µs)	4	470	2,000	76.5%	20,000	70,000	71.4%
	8	320	1,200	73.3%	10,000	40,000	75.0%
	12	290	1,100	73.6%	5,000	30,000	83.3%
	16	140	830	83.1%	5,000	20,000	75.0%
	20	170	780	78.2%	4,000	20,000	80.0%

SOLARFLARE®

sales@solarflare.com

US 1.949.581.6830 x2930

UK +44 (0)1223 477171

HK +852 2624-8868

www.solarflare.com

SF-113779-CD Issue 2
MEM_TB_012715