

# Performance Testing Application Device Queues (ADQ) with Memcached

**Innovative Intel® Ethernet technology improves open source memcached performance in benchmark testing.**

## Performance Terms

**Throughput** refers to how much data can be transferred from one location to another in a given amount of time, measured, for example, in transactions per second (TPS).

**Latency** in a network describes the average amount of delay in communication, measured, for example, in microseconds (μs).

**Tail latency** refers to the slowest response times (with the largest amount of delay) out of all the response times from a system. This is often measured as P(99) or 99th percentile latency, meaning the response time met by 99 percent of all requests.

**Jitter** is the variation in latency, or the opposite of response-time predictability.

*Increasing application response-time predictability by lowering jitter enables more compute servers to be assigned to a task and can allow more users to access the system, in addition to greater consistency in meeting customer service-level agreements (SLAs).*

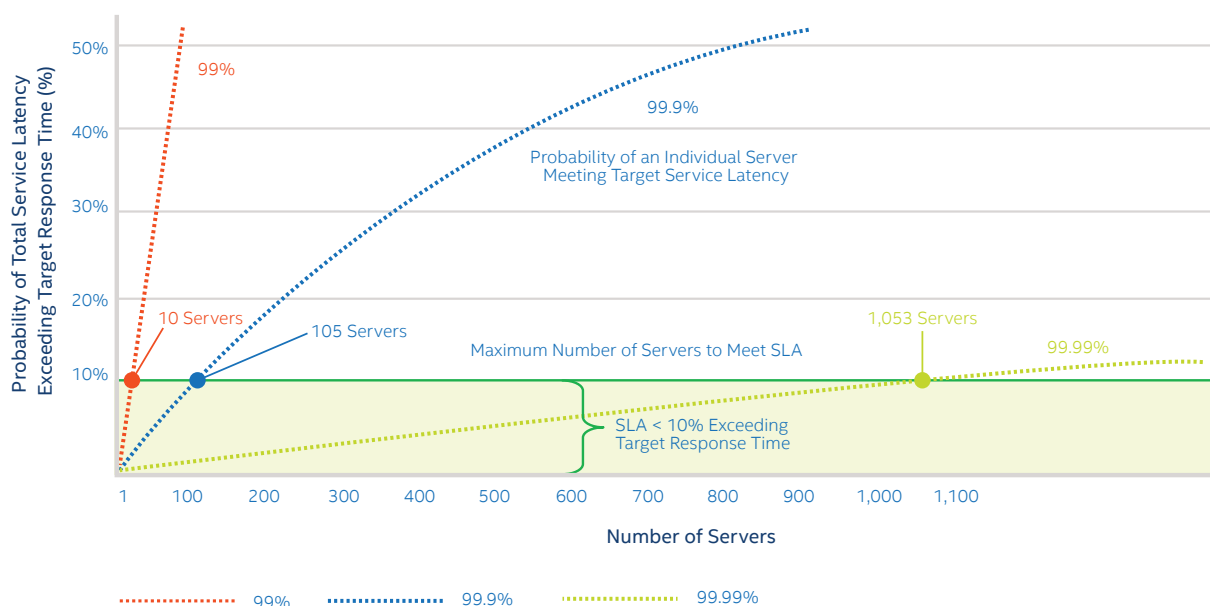
Modern distributed data center applications rely on powerful processors, massive storage, and fast connectivity to achieve high performance and availability. Connectivity performance has traditionally been measured in terms of latency and throughput—that is, the average speed and quantity of data movement. As data centers scale, a third metric becomes increasingly important: predictability.

## The Challenge of Predictability at Scale

When implementing a database application, end users typically expect a response in a given amount of time. Network architects will design the application and network system to respond within that response time. The predictability of an application's response time is typically measured in terms of jitter. Jitter refers to the *variability* of latency—the slight variation (earlier or later) in turnaround time when a response will be received—rather than the average latency in a server. The distribution of jitter increases with scale. Tail latency, the slow outliers in an otherwise fast system, becomes an increasingly significant factor for consideration as the data center scales with more servers.

For an analogy, consider flipping a coin. The odds of getting a heads result on the first flip are 1/2 or 50 percent. As you flip more coins, the probability of getting all heads goes down exponentially: 1/2, 1/4, 1/8, 1/16, and so on. Likewise with a server-based system, the probability of receiving every response within an acceptable response time goes down as the number of requests increases. While the likelihood of getting a response within the desired time in a server-based system is much higher (more than 99 percent), the magnitude of trials is also much higher, with tens, hundreds, thousands, or even tens of thousands of servers in use and tens, hundreds, or thousands of transaction requests per server. As a result, network designers must limit the number of servers assigned to a parallelized task, or limit the number of end users, to stay within the desired response time.





**Figure 1. Higher predictability enables more servers working in parallel within a desired response time<sup>1</sup>**

Jitter becomes a limiting factor to scalability when adding more servers, which causes system latency to exceed the user-experience requirements (see Figure 1). This is why, at scale, low latency and high throughput are not enough; predictability is also required. **Increasing the predictability of application response times by lowering jitter enables more compute servers to be assigned to a task and can allow more users to access the system, providing a better end-user experience.** Even applications that are not large can benefit from higher consistency, enabling them to meet service-level agreements (SLAs) more easily.

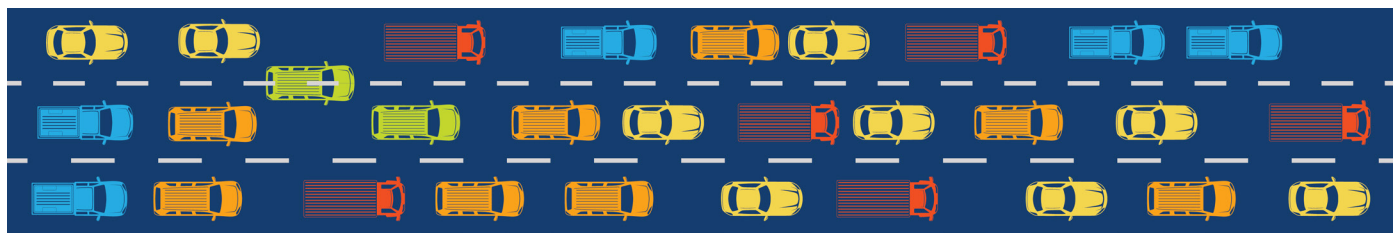
## Introducing Application Device Queues for Predictable, Scalable High Performance

Application Devices Queues (ADQ) is an open technology for system-level network input/output (I/O) performance that improves application response predictability and, thereby, data center scalability in a cost-effective manner. ADQ dedicates queues and shapes traffic for the transfer of

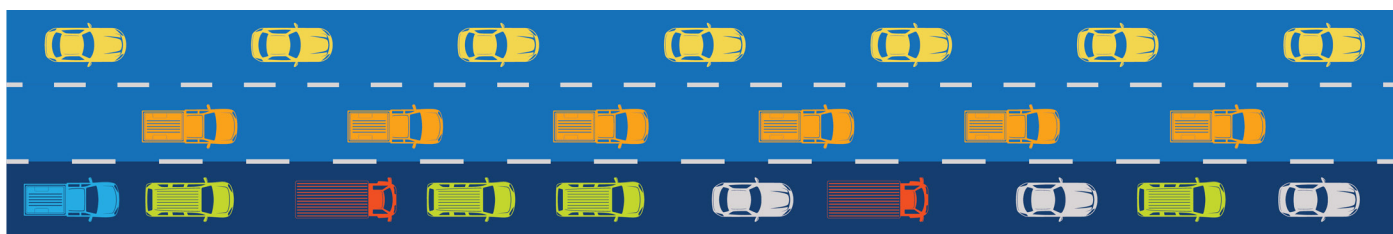
data over Ethernet for critical applications using standard operating system networking stacks and interfaces supplemented with Intel hardware technologies for improved performance. The goal of ADQ is to ensure that high-priority applications receive predictably high performance through dramatically reduced jitter.

### Data Express Lanes

A good way to understand ADQ is by analogy to freeway traffic. Imagine you want to get from your home to the airport in time to make your flight. If you take the freeway when traffic is light, the trip takes 30 minutes; but if the traffic is heavy, it might take as long as 90 minutes (see Figure 2). This means that, to be safe, you need to plan 90 minutes for the trip, which might waste up to an hour of your time. Likewise, a data-application developer who, not knowing how long it will take to receive requested data, must design around a worst-case scenario.



**Figure 2. Is there a traffic jam in your data center?**



**Figure 3. ADQ is like dedicated express lanes for your application data**

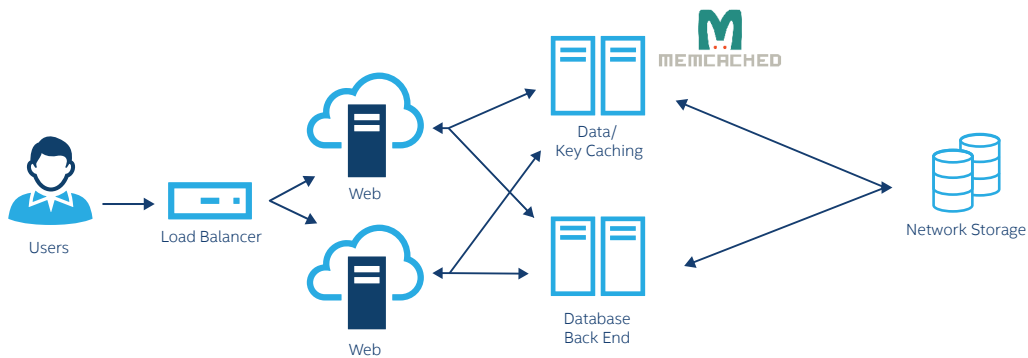


Figure 4. Memcached is a representative application for testing data and key caching

Now imagine dedicated express lanes available on the freeway allocated only to people traveling from your location to the airport (see Figure 3). Onramps are metered, offramps are limited, lanes are assigned, and speed is fixed so that the trip always takes 30 minutes—unaffected by any other traffic on its way to different destinations.

ADQ provides fast and predictable data-transfer performance that application developers can rely on so they, and network operators, can optimize their applications accordingly. By creating dedicated pipes between application threads and device queues, ADQ not only reduces contention for resources, but it also minimizes or eliminates synchronization operations such as locks and multithread sharing. Interrupts and context switching can also add traffic turmoil. ADQ avoids this additional turmoil by using busy polling to reduce the number of interrupts and context switches. ADQ uses a combination of these methods to improve application performance and reduce jitter.

ADQ offers application-specific, uncontended, smooth-flowing traffic, because there is no sharing of traffic from other applications on these queues. Traffic shaping reduces jitter by avoiding contention (no traffic jams), rate-limiting traffic (metered ramps), and reducing the number of interrupts and context switches per second (no lane changing).

ADQ Requirements

To take full advantage of ADQ, you'll need an application that has been enabled (if necessary) for ADQ, the latest Intel Ethernet 800 Series technology, and the updated Linux kernel, as shown in Table 1.

Table 1. ADQ deployment requirements

Component	Requirement
Application	Case 1: No change (for example, any single-threaded application)  Case 2: ISV ADQ-enabled application or reference code provided by the open source community (for example, memcached)
Configuration	Standard operating system tools and features (for example, iproute2, traffic control [TC], ethtool, and cgroup)
Operating System	Linux 4.19 or later
Ethernet Driver	Intel Ethernet 800 Series driver
Ethernet Network Interface Controller (NIC)	Intel Ethernet 800 Series

Testing Memcached with ADQ

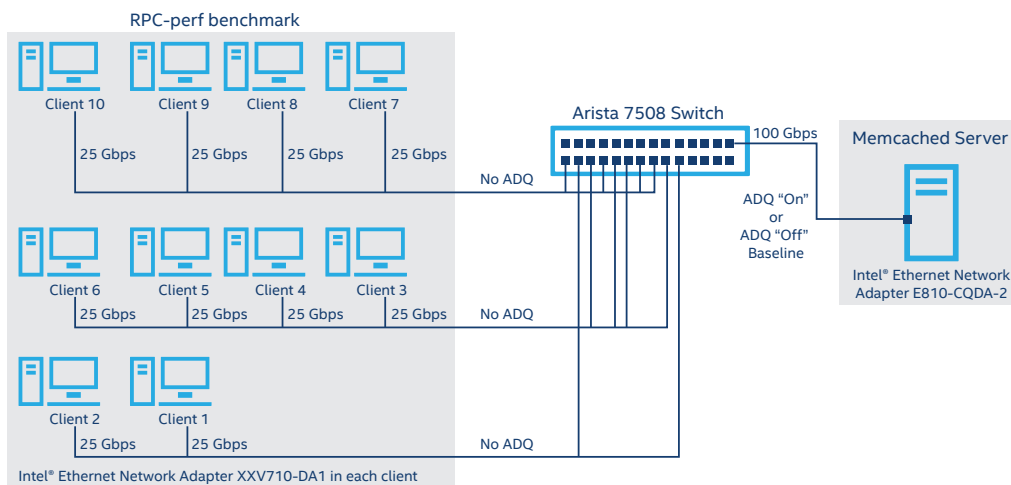
About Memcached

Memcached is an open source distributed memory caching system (see Figure 4). It is used for speeding up dynamic web applications or other workloads by storing data objects in dynamic memory to improve response times and reduce database traffic load. Memcached is used by companies such as YouTube, Twitter, Wikipedia, and Craigslist, among many others.<sup>2</sup>

When the client application requests a piece of data, memcached checks to see if that data is stored in cached memory. If it is, memcached returns the data to the client. If the data isn't stored in the cache, then the client application will query the database, retrieve the data, and subsequently store it in memcached. Whenever information is changed or an item expires, the client application updates memcached to ensure fresh content is delivered to the client.

Memcached components include both server-side and client-side software. A typical setup has multiple servers and multiple clients. Clients understand how to choose which server to read or write to for an item, using a hashing algorithm to help distribute the load among memcached servers. The servers understand how to store and fetch items. They also manage when to evict or reuse memory. The servers do not share data with each other, and, in fact, they are unaware of each other.

Memcached is a good representative caching application to test because its very nature as a distributed caching system puts a premium on the performance of the network. When caching is done internally on a server's own memory, there is no network latency involved. But when the caching is done on memory distributed among a number of other servers, the performance of the system will be highly dependent on the performance of the network connecting the client application to the cached data.



**Figure 5.** Memcached test setup

### Test Setup Details

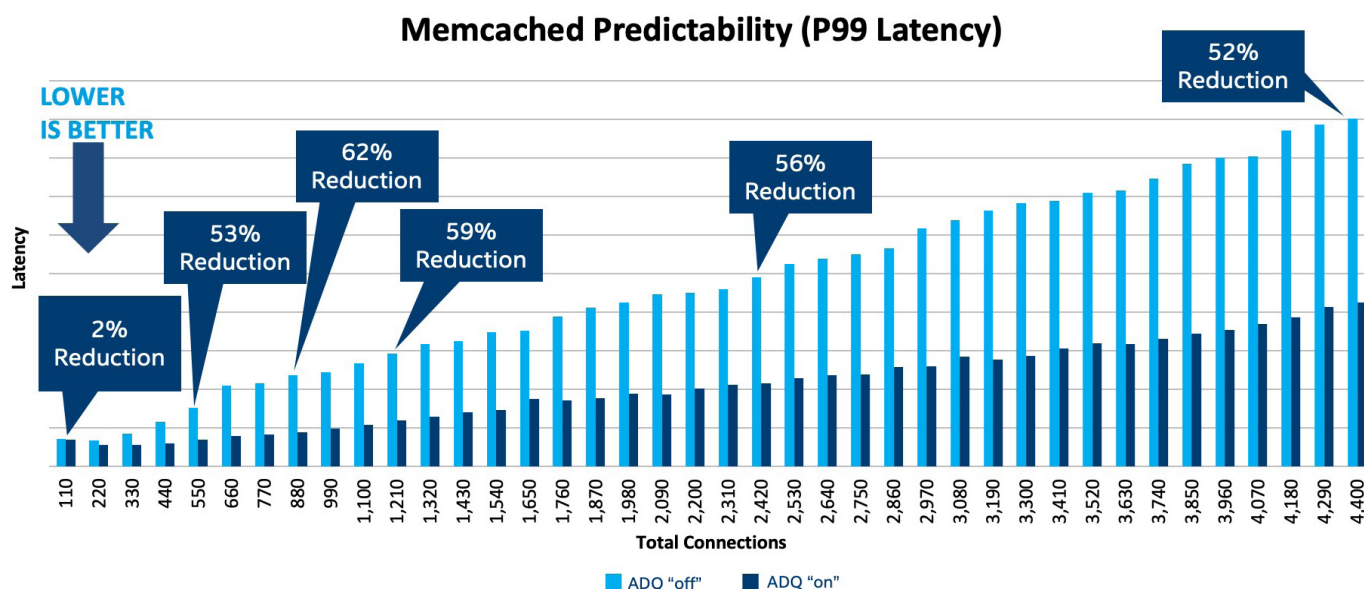
Intel tested memcached for performance with and without ADQ on the test configuration shown in Figure 5.

The memcached server was the system under test (SUT). 110 logical cores on the SUT were exposed, with an additional two logical cores used for system maintenance. Ten clients were configured with ten instances of the RPC-perf benchmark each. Tests were conducted at 11–44 connections per client (110 to 4,400 total connections) for a data size of 64 bytes, with an 8-byte key. This is a 100 percent read test with 20 million keys prepopulated at server startup.

### Test Results

Tests were performed for predictability, latency, and throughput, with results as shown in the following sections.

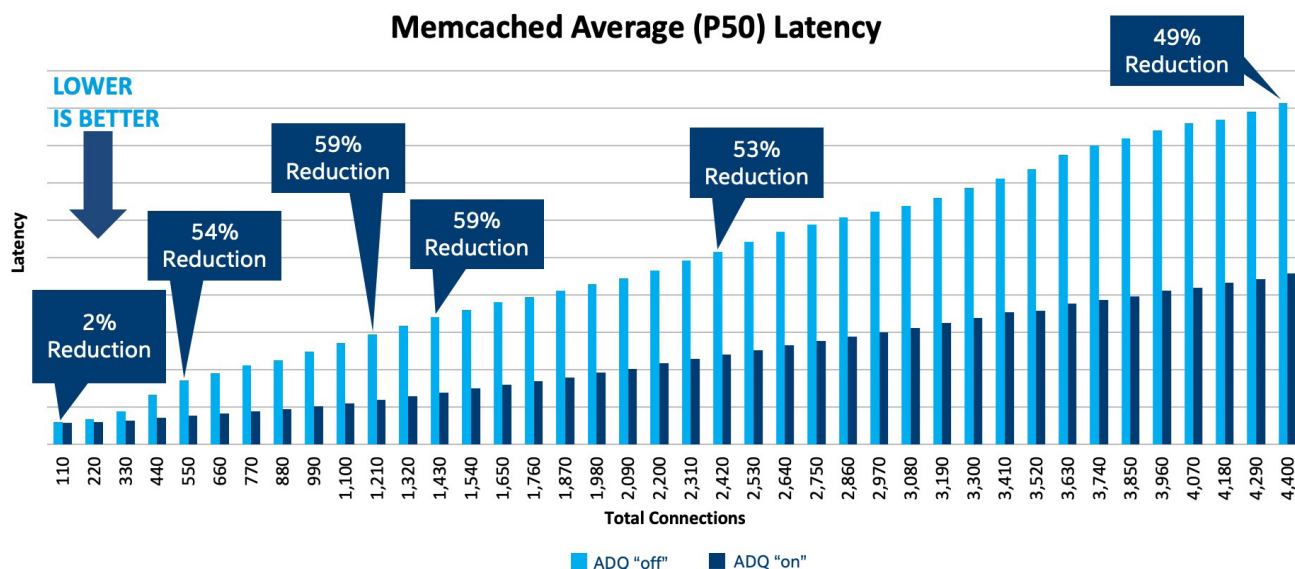
#### Predictability Results: Increased Up to 60 Percent



**Figure 6.** Predictability as measured by P99 latency with and without ADQ

Memcached predictability is measured by P99 latency, which is the probability that 99 percent of responses will be less than or equal to that value. At lower numbers of total connections, there is less difference between ADQ "on" versus ADQ "off," but as the number of connections increases, the difference (that is, the predictability) becomes more pronounced. P99 latency, or predictability, with ADQ "on" was more than half to 60 percent better than the latency with ADQ "off" at every connection rate above 550.

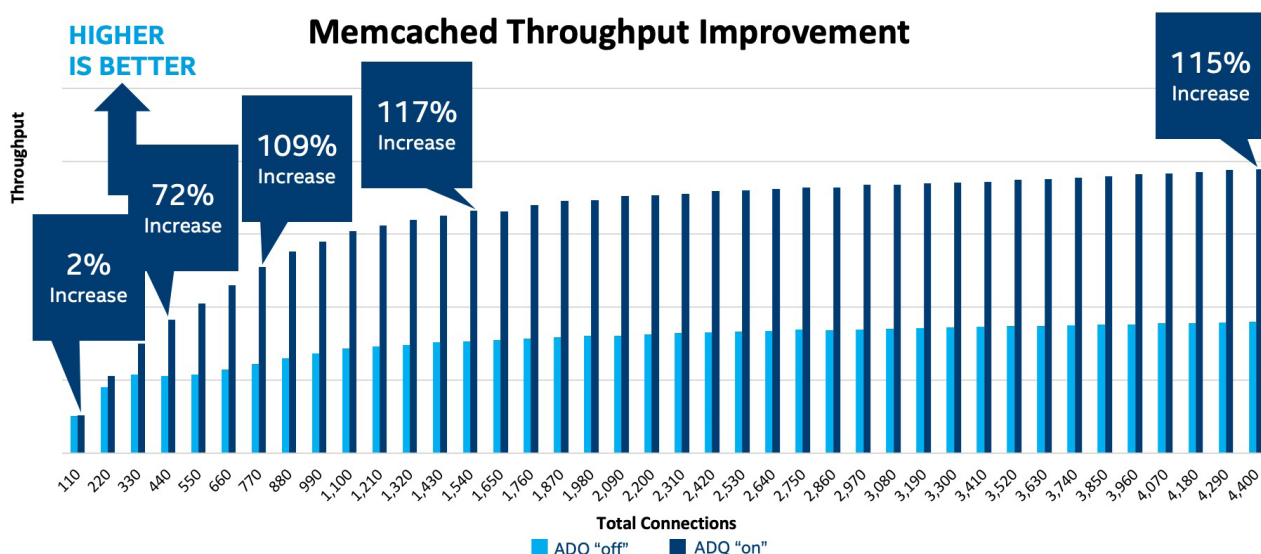
## Latency Results: Reduced Up to 60 Percent



**Figure 7.** Median average latency (P50) with and without ADQ

Figure 7 uses a P50 measurement (meaning 50th percentile) for latency. This represents the median average performance that is met by half the individual test cases for each connection level. Average latency with ADQ "on" was 49 to 60 percent lower than the latency with ADQ "off" at every connection rate above 550.

## Throughput Results: Greater Than 70 Percent Improvement with Higher Connections



**Figure 8.** Throughput improved substantially with ADQ turned "on"

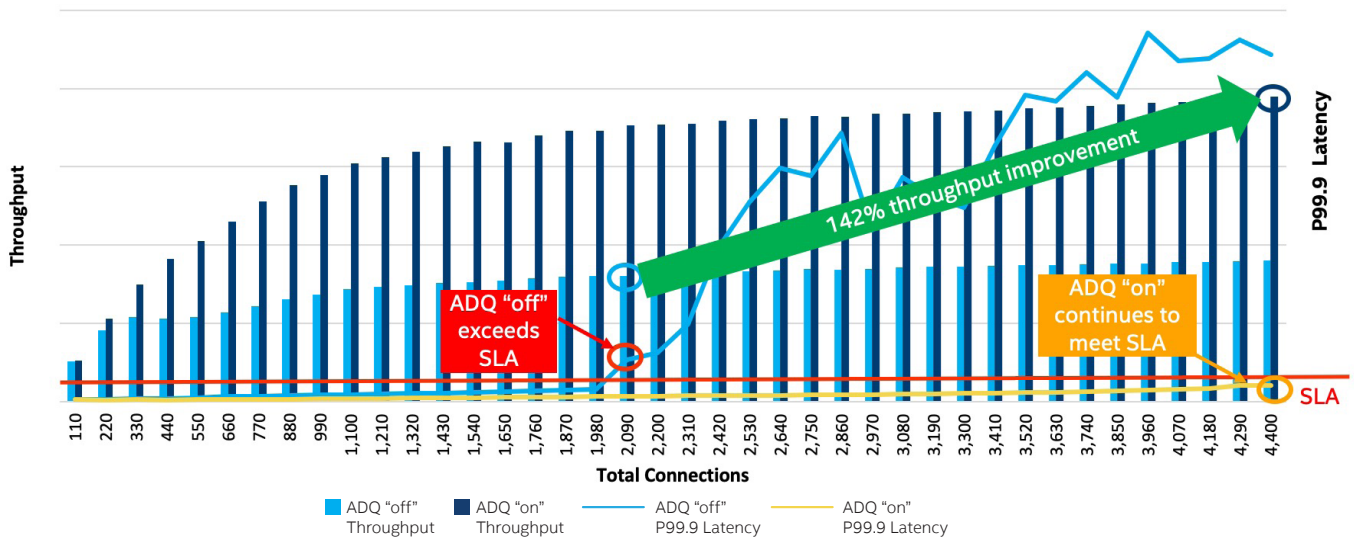
As displayed in Figure 8, the throughput improvement was greater than 70 percent with ADQ turned "on" for total connections of 440 or more.

## Use Case Example

One way to comprehend the impact of these performance improvements is to examine the impact on the customer SLA. Another common measurement is P99.9, or 99.9th percentile. P99.9 is often used in SLAs to promise that a certain maximum latency will be met 99.9 percent of the time. These SLAs are typically in the single-digit millisecond (ms) range. Figure 9 shows the P99.9 latency results, along with a hypothetical SLA maximum line drawn in red.



## Memcached Throughput Comparison while Meeting SLA for P99.9 Latency



**Figure 9.** Throughput improvement while meeting P99.9 SLA with ADQ "on" versus ADQ "off"

The solution with ADQ "off" exceeds the SLA at 2,090 connections, capping the throughput (request rate in transactions per second) at the amount shown in the light blue circle in order to meet the SLA. With ADQ "on," the P99.9 latency never exceeded the SLA, so testing did not identify a limiting cap for throughput. The highest number of connections tested was 4,400, with throughput at that number identified by the dark blue circle. This demonstrates a 142 percent throughput improvement compared to ADQ "off," while meeting the P99.9 SLA, though testing with more connections most likely would have shown even greater improvement.

The other remarkable result that Figure 9 illustrates is that with ADQ in this implementation, SLAs for latency offer the potential to be in the hundreds of microseconds range, compared to the typical millisecond ranges used previously. This could provide a new level of customer service in terms of application response times, or it could allow the system to be scaled even further within the current SLA.

## Final Analysis

Predictability increased by up to 60 percent, latency reduced by up to 60 percent, and throughput improved by more than 70 percent at higher connections in the memcached tests when ADQ was turned "on." The improvements were most dramatic at higher numbers of connections. In the hypothetical customer SLA scenario, more than a 140 percent throughput improvement was observed using ADQ while meeting a hypothetical P99.9 SLA and offering the potential to move to a hundreds-of- $\mu$ s SLA in this implementation.

The dramatic improvement in network predictability and reduction in jitter is of critical importance as data centers and applications scale, because it enables more servers to be added to a compute problem, more end users to be served, and greater consistency in meeting SLAs.

**INCREASES  
APPLICATION  
PREDICTABILITY**



**UP TO 60%**

**REDUCES  
APPLICATION  
LATENCY**



**UP TO 60%**

**IMPROVES  
APPLICATION  
THROUGHPUT**



**GREATER THAN 70%  
WITH HIGHER CONNECTIONS<sup>3</sup>**

## Learn More

Contact your Intel sales representative or distributor for more details about ADQ, and visit [intel.com/ethernet](https://intel.com/ethernet).

## Appendix: Test Configuration

**Table 2.** Details of the test equipment and configuration

	SUT	Client
Test By	Intel	Intel
Test Date	2/27/2020	2/27/2020
Platform	Intel Server Board S2600WFTF	Dell PowerEdge R630
# Nodes	1	10
# Sockets	2	2
CPU	Intel® Xeon® Platinum 8280 processor at 2.70 GHz	Intel Xeon processor E5-2699 v4 at 2.2 GHz
Cores/Socket, Threads/Socket	28 cores/socket, 56 threads/socket	22 cores/socket, 44 threads/core
ucode	0x500002c	0xb000038
Intel Hyper-Threading Technology (Intel HT Technology)	On	On
Intel Turbo Boost Technology	On	On
BIOS Version	SE5C620.86B.02.01.0010.010620200716	2.11.0
System DDR Memory Config: Slots/Cap/Run-Speed	12 slots, 16 GB, 2,933 megatransfers per second (MT/s) DDR4	8 slots, 8 GB, 2,666 MT/s DDR4
System Persistent Memory Config: Slots/Cap/Run-Speed	Not applicable (N/A)	N/A
Total Memory/Node (DDR+Persistent Memory)	192 GB	64 GB
Storage—Boot	1 x 512 GB Intel SSD	1 x 512 GB Intel SSD
Storage—Application Drives	N/A	N/A
NIC	1 x 100Gb Intel Ethernet Network Adapter E810-CQDA2	1 x 25Gb Intel Ethernet Network Adapter XXV710-DA1
Platform Controller Hub (PCH)	Intel C620 Series Chipset	Intel C612 Series Chipset
Other Hardware (Accelerator)	N/A	N/A
Operating System	Red Hat Enterprise Linux 8.1 (Ootpa)	CentOS 7.6
Kernel	4.19.106	3.10.0-957.el7.x86_64
Indirect Branch Restricted Speculation (IBRS) (0=Disabled, 1=Enabled)	1	1
Enhanced IBRS (eIBRS) (0=Disabled, 1=Enabled)	1	1
Retpoline (0=Disabled, 1=Enabled)	1	1
Indirect Branch Predictor Barrier (IBPB) (0=Disabled, 1=Enabled)	1	1
Page Table Isolation (PTI) (0=Disabled, 1=Enabled)	1	1
Mitigation Variants (1,2,3,3a,4, L1TF)	1,2,3,L1TF	1,2,3,L1TF
Workload and Version	Memcached ver. 1.5.22	RPC-perf benchmark ver. 3.1.0-pre
Memcached ADQ Patch	Pull request until put into the main branch: <a href="https://github.com/memcached/memcached/pull/610">https://github.com/memcached/memcached/pull/610</a>	N/A
Compiler	gcc (GCC) 8.3.1 20190507	gcc (GCC) 4.8.5 20150623
NIC Driver	Ice.0.14.0_rc55	i40e 2.8.43

Table 3. SUT network-adapter configuration settings

	ADQ "Off" Baseline	ADQ "On"
<b>System Settings</b>		
Interrupt Moderation	Fixed	Fixed
IRQ Balance	No	No
Interrupt Affinitization	Yes	Yes
<b>ADQ Settings</b>		
Epoll Busy Poll	Yes	Yes
Socket Option for NAPI ID	Yes	Yes
TC-Mqprio Hardware Offload and Shaper	No	Yes
TC- Cloud Filter Enabling with TC-flower	No	Yes
Symmetric Queueing	Yes	Yes



<sup>1</sup> Jeffrey Dean and Luiz André Barroso. "The Tail at Scale." *Communications of the ACM*. February 2013. <https://cseweb.ucsd.edu/~gmporter/classes/fa17/cse124/post/schedule/p74-dean.pdf>.

<sup>2</sup> DZone. "Memcached - A Distributed Memory Object Caching System." March 2016. <https://dzone.com/articles/memcached-a-distributed-memory-object-caching-syst>.

<sup>3</sup> 70 percent and better improved throughput on tests with 440 or higher total connections.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available security updates. See configuration disclosure for details. **No product or component can be absolutely secure.**

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No component or product can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com](https://intel.com).

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit [intel.com/benchmarks](https://intel.com/benchmarks).

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804

Your costs and results may vary.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.