TECHNOLOGY BRIEF

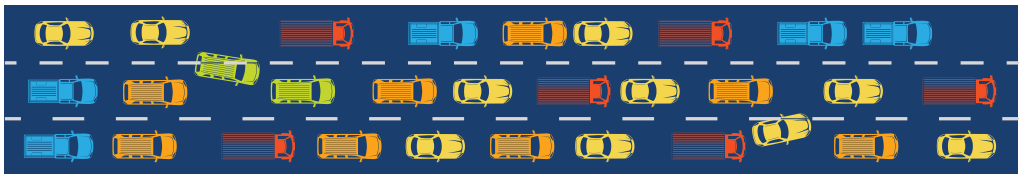
Intel Connectivity
Ethernet Networking

Faster, More Predictable Ethernet with the Intel® Ethernet 800 Series with Application Device Queues (ADQ)

Achieve greater consistency in meeting customer service-level agreements (SLAs).

As your data center scales to handle larger quantities of data and more complex workloads, connectivity becomes increasingly important to overall system performance. Ethernet is like a freeway system for data traveling between different places in the data center. Like a freeway at rush hour, the network can slow to a crawl when there's too much traffic competing for access.

Is there a traffic jam in your data center?



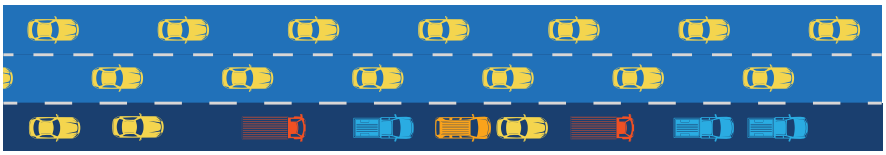
Ethernet needs express lanes to prevent traffic jams from slowing the performance of critical applications in your data center. Intel® Ethernet 800 Series network adapters include Application Device Queues (ADQ) technology, which can serve as express lanes for critical application data. ADQ can significantly improve the performance of your key applications, providing greater consistency in meeting customer SLAs.

Predictability Is the Key

When measuring performance in a data center, most people initially think of metrics like throughput and latency—how much data can be processed per second, and how long does an operation take? Throughput and latency are certainly important metrics, but they have one thing in common: they represent averages of performance.

Predictability, on the other hand, is about the outliers and the laggards. This is called *tail latency*, referring to the instances that appear at the slow end, or tail end, of the curve. If an operation usually takes 1 millisecond, but sometimes takes 10 milliseconds, then 10 milliseconds is the tail latency. The critical question is about predictability: how often do slow operations occur? If only one in a million operations is slow, then it's highly predictable, even if scaled to many systems. But if one in a hundred operations is slow, while that might be fine for an individual system, when scaled to many systems, the predictability is not very good.

Think of planning your trip to the airport on the freeway. If it usually takes half an hour, but sometimes takes an hour, when do you need to leave? To be safe, you always need to allow for an hour. If only you had a dedicated express lane to the airport that would predictably get you there in half an hour! Then you could consistently have a shorter trip. That is essentially what ADQ is: dedicated express lanes on the freeway for your most important applications.



ADQ express lanes keep Ethernet traffic running smoothly for your critical applications.

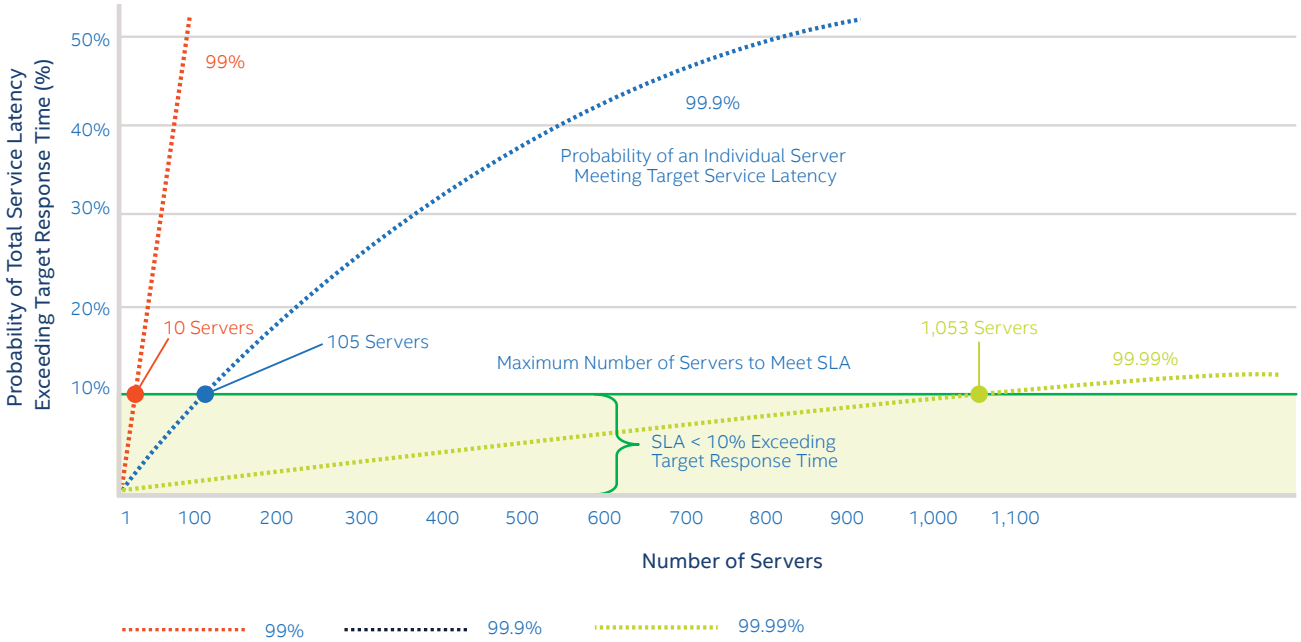


Figure 1. Probability of exceeding SLA target latency

But a data center is more complicated than the airport freeway analogy, because a data center uses parallel computing. A single job might be split up, and its parts sent to be completed by dozens or hundreds of different servers at the same time. And there's the problem. The job will not be completed until the *last and slowest* thread completes and returns its part of the job. This means that the more servers you scale to, the higher the likelihood that one piece of the result will be slow to return and thereby slow the completion of the entire job. Therefore, it is imperative that each split job is extremely predictable to get a reasonable overall predictability for the job as a whole. The network and networking software stack can play a significant role in influencing this turnaround time.

Although scaling out data centers can bring many advantages in performance and economies of scale, one of the challenges is that more servers make it more difficult to consistently meet customer SLA requirements.

Figure 1 illustrates how adding more servers increases the mathematical likelihood of missing your SLA targets. It also shows that improving the predictability of each server alleviates that problem and lets you use more servers without exceeding your target response times.

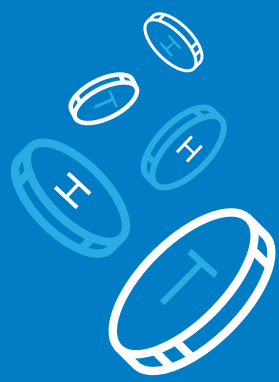
The larger the data center, the more important predictability becomes. Average latency is not good enough; tail latency must also be managed. That's why ADQ is important.

Flipping Coins

To understand why adding more servers makes it more difficult to meet an SLA for system latency, it might help to think about flipping coins. Let's say that *heads* is success and *tails* is failure. You might offer an SLA based on an expected 50-percent failure rate when repeatedly flipping a single coin.

But now let's introduce parallel processing by flipping more than one coin at a time. If you flip two coins, the chances of success (that is, all *heads*) go down to 1 in 4. And adding more coins makes the probability of success go down exponentially: 1 in 8, 1 in 16, 1 in 32, 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 parallel requests increases.



Will these all land on heads?

INCREASES APPLICATION PREDICTABILITY

MORE THAN 50%

Based on tests using open source Redis* software[1]

REDUCES APPLICATION LATENCY

MORE THAN 45%

Based on tests using open source Redis* software[1]

IMPROVES APPLICATION THROUGHPUT

MORE THAN 30%

Based on tests using open source Redis* software[1]

ADQ Increases Predictability

ADQ is an open technology designed to help address network traffic challenges by improving throughput and latency, and, most importantly, by enabling greater predictability in application response times. ADQ is enabled by ingredients that Intel has made available to the industry in the Linux* kernel. Intel's first fully realized implementation of ADQ is in the Intel Ethernet 800 Series.

Tests show significant improvements in application performance when ADQ is turned on versus without ADQ, including no busy polling by Linux. Tests using open source Redis* database software, for example, showed that with ADQ enabled, predictability increased by more than 50 percent, application latency was reduced by more than 45 percent, and application throughput improved by more than 30 percent.[1]

How ADQ Works

How are such big improvements accomplished while using the same software running on the same hardware? The answer lies in how ADQ prevents traffic jams in the data center network. To return to the airport freeway analogy, what ADQ does is like reserving express lanes on the freeway solely for use by guests at your hotel traveling to the airport. You don't need to share the lanes with other traffic going other places, so your trip is not slowed down by rush-hour traffic. Similarly, ADQ lets software applications reserve lanes, or queues, directly to the destination hardware devices in your data center—without sharing or competing with other applications.

Each Intel Ethernet 800 Series adapter has 2,048 dedicated hardware queues that can be configured as dedicated ADQs or used for standard traffic. The system administrator defines how many queues to assign to a given application, thus giving more queues to higher priority applications to help ensure their predictable high performance.

This level of performance was previously only achievable by bypassing the kernel using proprietary TCP/IP software stacks for specific applications using user-mode networking. Using ADQ, performance that was once only possible with user-mode networking is now possible with kernel-mode networking. Kernel-mode networking is open source and takes inherent advantage of enhancements in the Linux kernel such as extended Berkeley Packet Filter (eBPF) with containers, netfilters, and other TCP enhancements.

What Is Needed to Implement ADQ?

ADQ is straightforward to implement with the following ingredients.

Table 1. Requirements for using ADQ

Component	Requirement
Application	Case 1: No change (for example, open source Redis*, Netperf*) Case 2: ADQ-enabled application
Configuration tools	Standard operating system tools
Operating system	Linux 4.19* or later
Ethernet driver	Intel® Ethernet 800 Series driver
Ethernet NIC	Intel Ethernet 800 Series

Intel has upstreamed key patches to the Linux kernel to enable ADQ; these patches have been accepted by the Linux community in version 4.19 and later. Plans for other operating systems are currently under development.

Standard Linux operating system tools are used to configure ADQ—specifically, iproute2*, traffic control (TC), ethtool*, and cgroup*.

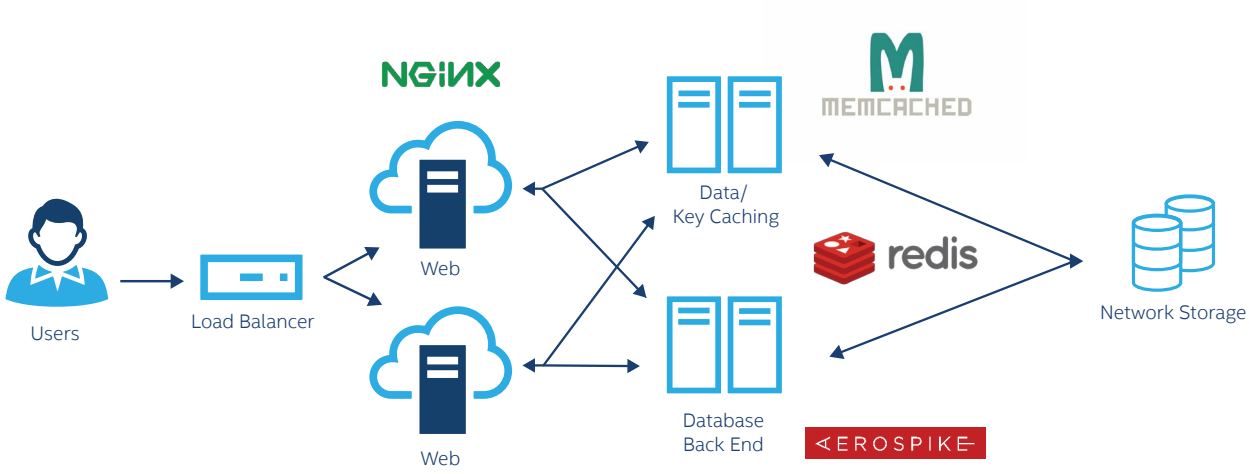


Figure 2. Representative applications for testing ADQ performance

The first set of specific applications targeted for the Intel Ethernet 800 Series with ADQ are NGINX*, memcached*, Redis, and Aerospike*.

In most cases, ISVs will add some code (typically less than 50 lines of C code) to enable an application for ADQ, although single-threaded applications like open source Redis or the Netperf* benchmark require no modification. ISVs who make modifications and optimize their code to support ADQ will provide documentation for how to implement their solutions. For the open source Redis and memcached applications, Intel provides a best-known configuration (BKC) guide that shows how system administrators can assign dedicated ADQs to latency-intolerant applications. Many more applications will be enabled in the areas of communications, storage (for example, NVM Express* [NVMe*] over TCP), and containers.

Get on the ADQ Expressway

Ethernet traffic jams can slow critical workloads in the data center. As data centers scale out with more servers, the likelihood increases that a single server experiencing such a slowdown will slow the whole application. The ability to offer and meet strong SLAs depends on the consistency of application performance, and that depends on the predictability of your system to meet latency targets. ADQ acts like express lanes on the Ethernet freeway that give critical applications fast, dedicated lanes and keep them out of traffic jams. ADQ has been shown to achieve greater predictability in application performance in addition to lower latency and higher throughput.

Learn More

Learn more about how Intel Ethernet 800 Series network adapters let you use ADQ to achieve greater consistency in meeting customer SLAs:

- **Learn about the Intel Ethernet 800 Series**
- **Read the open source Redis ADQ test results**

[1] Source: Intel. "Performance Testing Application Device Queues (ADQ) with Redis." March 2019. intel.com/content/www/us/en/architecture-and-technology/ethernet/application-device-queues-with-redis-brief.html?wapkw=redis+solution+brief. Based on Intel internal testing as of February 2019; open source Redis on 2nd Generation Intel® Xeon® Scalable processors and the Intel® Ethernet 800 Series, with 100 gigabit Ethernet (GbE) on the Linux 4.19.18* kernel.

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

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.

Intel technologies may require enabled hardware, specific software, or services activation. Check with your system manufacturer or retailer.

Printed in USA 0919/TK/PRW Please Recycle 341311-001US