Introduction to Intel® Ethernet Flow Director and Memcached Performance

Problem Statement

Bob Metcalfe, then at Xerox PARC, invented Ethernet in 1973, over forty years ago. One of Ethernet's key strengths leading to its longevity has been its capacity to evolve. Metcalfe could not have possibly envisioned all the developments in computing that happened over the subsequent four decades. But he, and all the other early Ethernet pioneers, including Digital Equipment, Intel, and Xerox (the "DIX" group that developed the first specification submitted to the IEEE), laid a foundation solid enough and flexible enough to support Ethernet's dramatic evolution.

The problem that Metcalfe was trying to solve was how to interconnect multiple Xerox Alto "individual computers" (pre-PCs) and also connect them to shared Xerox laser printers. It was the latter requirement that drove Metcalfe to a 10 Mbps solution, staggeringly more bandwidth than anyone could envision using in 1973 for networking computers, but this headroom for the future was yet another factor making Ethernet solutions attractive. Key here is that Metcalfe's original problem was getting data packets between computers and to laser printers. Once the packets arrive at the Ethernet endpoint, Ethernet is done.

For about three decades this packet delivery model worked just great (as other aspects of Ethernet evolved). This is actually pretty remarkable, given the pace of technological change. But this model became challenged in the first decade of the 21st century with the mainstream deployment of multi-core processors from Intel and others. Now you have a problem. The Ethernet packet arrives at the Ethernet controller endpoint. Ethernet is done. But which core should receive the packet? In a nutshell, that is the problem that Intel® Ethernet Flow Director was designed to solve.

The details of how system software (operating systems or hypervisors) deal with this problem will vary, but typically an incoming block of data is DMA'd into host memory (actually, the data goes first to the top-level processor cache with Intel® Direct Data I/O (DDIO)-architected CPUs like the Intel® Xeon® processor families E5 and E7) by the Ethernet controller. When finished, the Ethernet controller, potentially after some delay for interrupt moderation, then interrupts a processor to say, "I just DMA'd a block of data into your host memory." In response, system software will then pick a single core to service the I/O interrupt, perform protocol processing, e.g., TCP/IP. However, in general, the core system software selects will not be the core where the consuming application resides. When the core servicing the Ethernet controller's interrupt has completed its tasks, it interrupts the core where the consuming application resides. That core, in turn, will then restart or continue the protocol processing (depending on system software) and deliver the data to the consuming app. Obviously, this is not a very efficient data transfer processor especially because an extra interrupt is required to get the incoming data to the application that wants it and potentially some duplication in protocol processing. But the problem is worse than that. Obviously, with any significant LAN traffic at all, the receiving core will get overloaded, choking incoming LAN traffic.

Receive Side Scaling: A Good Solution for Core Overload

Receive Side Scaling (RSS) is a technology defined by Microsoft and first implemented in their NDIS 6.0. It is now broadly supported in all major system software including Linux* and VMware's ESX. Intel has implemented RSS as an Intelligent Offload hardware acceleration for Intel® Ethernet Controllers and Converged Network Adapters. RSS doesn't solve the problem of getting incoming packets to the right consuming core, but at least RSS addresses the problem of a single core becoming the bottleneck to network performance.

The RSS Intelligent Offload in an Intel controller inspects every packet and computes a 32-bit hash using the Toeplitz algorithm. The information going into the hash calculation comes from the packet's header fields, including source and destination IP addresses, and source and destination TCP ports (96 bits for IPv4) for Windows*. For Linux, the 8-bit Ethertype is also included as input to the hash calculation. Some number of the Least Significant Bits (LSBs) of the computed hash becomes an index. (For Windows this must be at least 7 bits.) An index lookup in a redirection table will specify a queue for that index. Intel® Ethernet Converged Network Adapters X520 and X540 support a 128-entry RSS Indirection Table per port. A single MSI-X interrupt will be associated with each RSS queue that specifies a distinct core as the intended recipient of the data in the queue. Actually, at this point Windows only supports 16 queues. The reason the Indirection Table has so many entries, more than supported queues, is that it allows system software to actively load-balance between cores.

An example may help clarify how an Indirection Table larger than the number of available queues provide system software with the granularity to load-balance incoming traffic should a particular core become overloaded. In this example, indices 0 and 4 both go to queue 15.

LSBs of hash

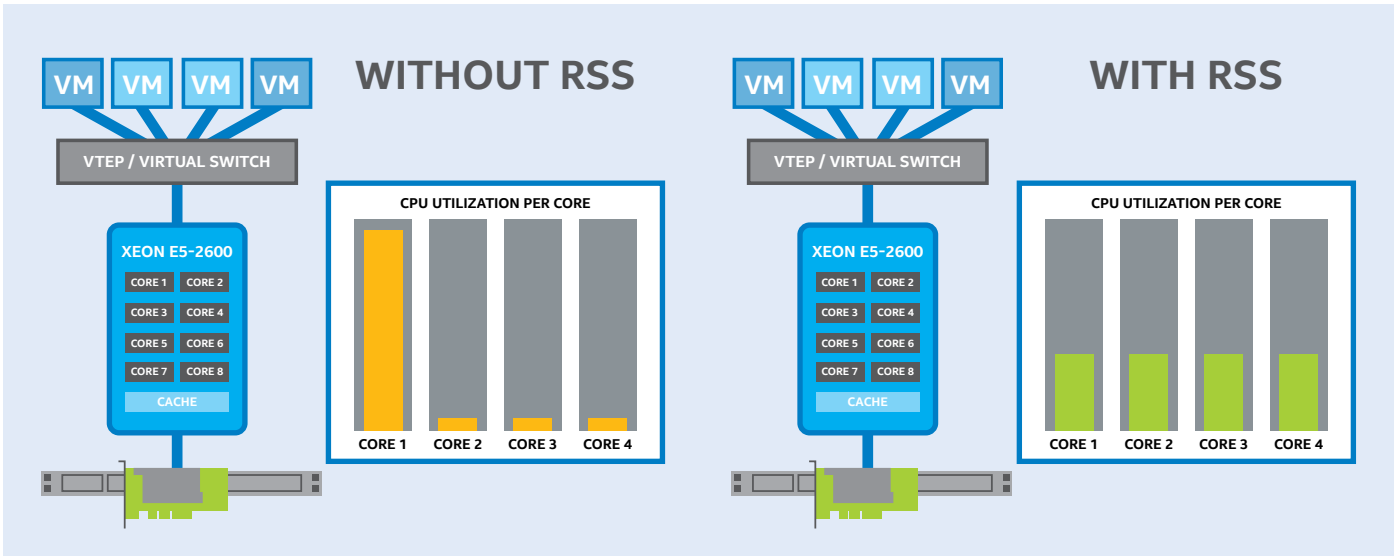
INDEX	DESTINATION QUEUE
0	15
1	2
2	8
3	11
4	15
5	7
6	5
...	...
127	2

Packet goes to queue 7

RSS DIRECTION TABLE

System software can change the destination corresponding to, say, index 4 to queue 13 if the core associated with queue 15 is overloaded. Note that only system software has access to the RSS Indirection Table. It is not user programmable.

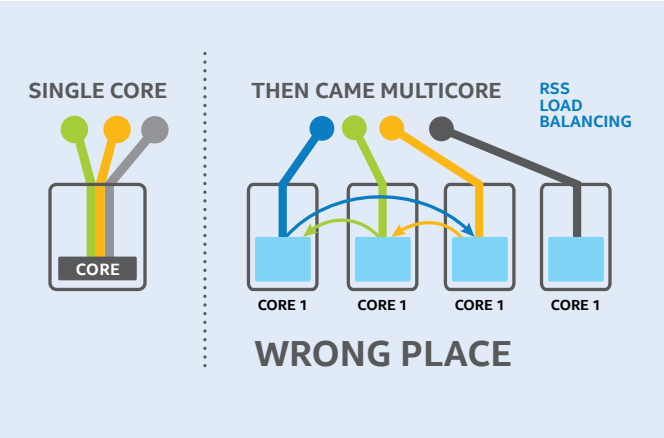
The RSS Intelligent Offload will place the first packet for each distinct hash/index into its assigned RSS queue as directed by the Redirection Table. Subsequent packets received with the same hash (and therefore, index) will be directed by this process to the same core, thereby assuring packets remain ordered. (If the packets from the same flow were spread across multiple cores, there would be no assurance that packet sequence order would be preserved.) The index is effectively a "filter" of network traffic into queues of data headed for the same destination. The index may not be unique for all the incoming flow, i.e., there may be collisions with multiple distinct flows going to the same RSS queue which system software sorts out later.



The important point here is that the RSS Redirection Table will *spread* the various queues with distinct hashes *across* the available cores, solving the problem of overloading a single core with incoming traffic.

It is possible to accomplish something functionally equivalent to Intel's RSS hardware-based Intelligent Offload in software in Linux with Receive Packet Steering (RPS) for controllers that lack hardware-based RSS support. As you would expect RPS does not deliver the performance of RSS because the actions performed by RSS have to be managed in software executing from code running in, most likely, yet another core. For this reason, all modern Ethernet controllers support RSS.

So RSS solves the problem of the overloaded core receiving all network traffic. Improvement in peak network bandwidth can be dramatic by enabling RSS. Where RSS falls short, however, is that RSS has no way of knowing anything about application locality. In general, RSS will send the data to the wrong core. RSS may have consistently sent all the data that an application in core 7 wants—to core 3. System software will have to interrupt core 7 to inform it that it needs to claim its data.



Intel® Ethernet Flow Director to the Rescue

Intel® Ethernet Flow Director is an Intelligent Offload intended to help solve the problem of getting incoming packets directly to the core with the application that will consume them. Flow Director has two fundamental operating modes using the same hardware: Externally Programmed (EP) and an automated Application Targeting Routing (ATR). EP mode might be preferred when a system administrator knows in advance what the primary data flows are going to be for the network. ATR, the default mode, implements an algorithm that samples transmit traffic and learns to send receive traffic with the corresponding header information (source and destination reversed) to the core where the transmitted data came from. ATR is the best solution when flows are unpredictable and the key action the user wants from Flow Director is simply pinning the flow to a consistent core.

Besides adding intelligence, Flow Director differs from RSS in several key respects starting with the nature of their tables. The values in an RSS indirection table are queue numbers. The values in a Flow Direct Perfect-Filter table are actions, only one of which is routing to a specific queue. Though that is the most frequently used action, there are others. For example "drop packet" could be useful in managing a denial of service attack. Incrementing the count in one of 512 packet counters can be useful in Externally Programmed mode. Other examples include tuning interrupt moderation for either low latency or high bandwidth and copying a software-defined value to a receive descriptor, also both potentially useful in Externally Programmed mode.

The second key difference stems from what each technology is trying to accomplish. RSS is trying to spread incoming packets across cores while directing packets from common flows to the same core. Unlike RSS, Intel Ethernet Flow Director is trying to establish a unique association between flows and the core with the consuming application. Indexing solely by a hash won't do that. Distinct flows must be uniquely characterized with a high probability which cannot be accomplished by a hash alone. And the Intel® Ethernet Flow Director Perfect-Match Filter Table has to be large enough to capture the unique flows a given controller would typically see. For that reason, Intel Ethernet Flow Director's Perfect-Match Filter Table has 8k entries.

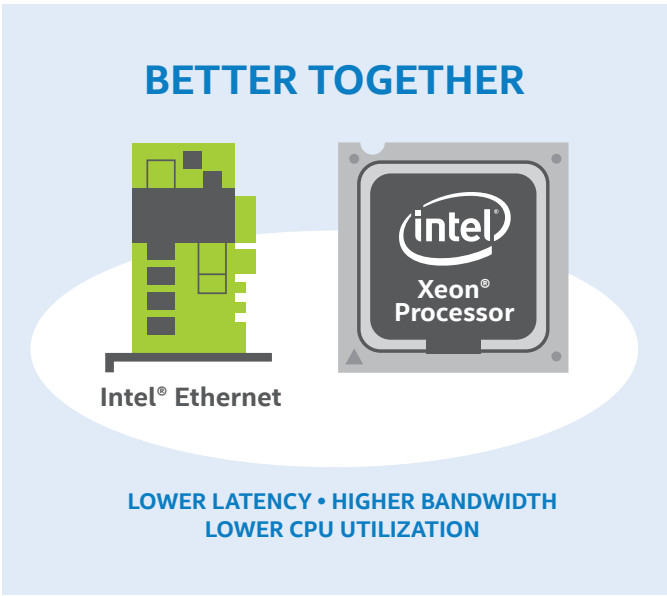
The basic idea is to intelligently select fields from the incoming packets and, in ATR mode, fields from outgoing packets that are likely to uniquely identify that packet's flow. The fields selected will depend on the packet type, e.g., different fields will be extracted from TCP/IP and UDP packets. A "Perfect-Match" value will be distilled from each packet: 12-bytes in the current generation of Intel Ethernet controllers, going to 48 bytes in future Intel Ethernet controllers. To populate the table in ATR mode, each sampled outgoing packet's Perfect-Match value will be hashed to create an index into the Filter Table. The full Perfect-Match value and identifier to the sending core will be placed in the Filter Table.

The hash from incoming packets will index into the Perfect-Match Filter Table. However, a hit to a populated entry does not guarantee uniqueness. Each incoming packet's Perfect-Match value will be compared to the Perfect-Match value pointed to by its hash into the Filter Table. Only if the Perfect-Match values are an exact match will the action specified in the Filter Table be taken. If there is an exact match, it is extremely likely that the incoming packet's consuming application resides in the core that corresponds

to the routing (or other) action specified in the Perfect-Match Filter Table. If the match fails, packet processing will revert to RSS mode. (Actually, Intel Ethernet Flow Director is a bit more sophisticated than this and can test against multiple Perfect-Match values corresponding to a common hash, but how this works is beyond the scope of this paper.) The only difference for Externally Programmable mode is that the user or software (e.g., OpenFlow) takes control of populating the Filter Table.

Just as Linux offers a software-based version of RSS (RPS), it also offers a software-based version of Application Targeting Routing-mode Flow Director called Receive Flow Steering (RFS). Though functionally equivalent, RFS cannot offer the performance of Intel Flow Director because it has to do its dispatching and steering from a core, and likely not in either the receiving core or in the core with the consuming app. ATR-mode Flow Director can be thought of as an Intelligent Offload hardware acceleration of RFS.

Where does Intel Ethernet Flow Director deliver the most value? Intel Ethernet Flow Director is not currently supported by VMware or by Microsoft. Intel Ethernet Flow Director delivers the most benefit in "bare metal" Linux applications where small packet performance is important and network traffic is heavy.



Memcached: An Application that Demonstrates the Power of Intel Ethernet Flow Director

Memcached is an open-source technology that uses the pooled DRAM from a server cluster as a storage cache, dramatically reducing latency. Specifically, "Memcached is an in-memory key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering.[1]" Memcached is used by many cloud and web-services companies, including Facebook, Reddit, Twitter, and YouTube. The important point here is that Memcached is an excellent stress test that shows what peak loads a network can handle. Memcached is effectively "networked DRAM." Memcached performance results are valuable even if you are not running Memcached because they illustrate what Flow Director can do to increase network peak performance.

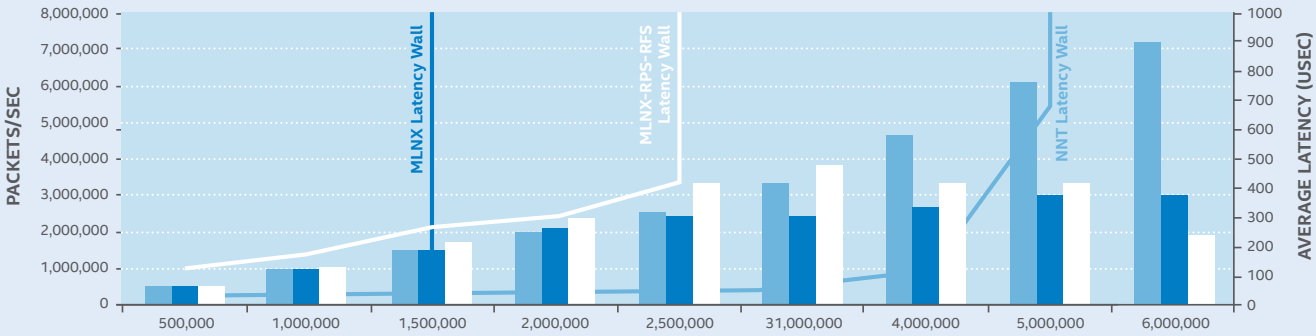
The following graph shows how Intel Ethernet Flow Director delivers superior Memcached throughput and latency. Tested are the Intel Ethernet CNA X520 ("NNT"), the Mellanox ConnectX-3* (MLNX), and the Mellanox ConnectX-3 with RPS and RFS enabled.

Memcached is requesting packets at increasing rates. What you see is that the Intel® X520 can respond handily to up to 5M requests per second. Only when the request rate gets up around 4 Mps does the latency start increasing. In contrast, the ConnectX-3 never gets above 3M responses per second, and its latency "goes up a wall" before it gets to 1.5M requests per second. RPS and RFS help the ConnectX-3, but it still cannot get above 4M responses per second, and it hits its latency wall before it gets to 1.5M requests per second. Even with the help of RPS and RFS, the Mellanox ConnectX-3 does not even come close to what Intel Ethernet can do with Flow Director.

In sum, Intel Ethernet Flow Director is a technology that successfully accelerates getting incoming packets to their consuming application. In applications that stress network performance, like Memcached, Intel Ethernet Flow Director offers a significant performance enhancement to an Ethernet network in terms of increased bandwidth and lower latency. Concomitant benefits include reduced CPU utilization and power consumption.

Intel Ethernet Flow Director is supported in the recently launched Intel® Ethernet Controller XL710 family that supports up to two ports of 40GbE, in addition to the Intel Ethernet Converged Network Adapters X520 and X540.

RECEIVED PACKETS/SEC AND LATENCY AS A FUNCTION OF REQUEST RATE



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, Intel does not control or audit the design or implementation of third party benchmark data or Web sites referenced in this document. Intel encourages all of its customers to visit the referenced Web sites or others where similar performance benchmark data are reported and confirm whether the referenced benchmark data are accurate and reflect performance of systems available for purchase. including the performance of that product when combined with other products.

TEST CONFIGURATION

Memcached v1.4.15, 32-thr
10 Clients, Mcblaster
Record Size = 64 Bytes (TCP)
Number of keys = 500,000
Threads per client (-t) = 12
Connections per thread (-c) = 1
Nagles Disabled
SUT: Rose City
2 Intel® Xeon® processors (JKT, E5-2687W, 3.1 GHz)
16 GB, 8-ch, DDR3, 1333 MHz
Patsburg C0
BIOS v46
Intel® x520 Adapter, ixgbe 3.18.7
Mellanox CX3 Adapter, mlx4_en 2.1.8
RHEL 6.3 x64, kernel v3.6.0
Clients
SuperMicro* 6016TT-TF (Thurley twin)
2 Intel® Xeon® processors (X5680, 3.33 GHz)
6 GB, DDR3, 1333 MHz
Intel® x520 Adapter
RHEL 6.3 x64, kernel v3.12.0
Network Configuration
Nexus 5020
Clients* connected @ 10G

For more information visit: www.intel.com/go/Ethernet

[1.] memcached.org.

 1014/MR/HBD/PDF 331109-001US