QuickPath Interconnect: Considerations in Packet Processing Applications

An Industry Whitepaper

Contents

Executive Summary ................................... 1
Introduction to QuickPath Interconnect ........ 2
Considerations for Stateful Packet Processing... 4
Processor and Core Affinity ...................... 4
    Network Routing Asymmetry and Processor Migration .......................... 5
Cabling ............................................ 6
Other Compounding Factors ...................... 6
Memory Look-Ups ................................ 6
Packet Size ....................................... 6
Ensuring Core Affinity ............................ 7
Redesigning the Network ....................... 7
Flow Director .................................... 7
Network Processing Unit ....................... 8
Conclusions ...................................... 9
Questions to Ask About Packet-Processing Platforms .................................. 9
Additional Resources ............................. 9
Invitation to Provide Feedback ............... 9

Executive Summary

An issue of critical importance to stateful packet-processing applications, including deep-packet inspection (DPI) and network policy control, is Intel’s QuickPath Interconnect architecture.

In an architecture update, Intel introduced a separate, integrated memory controller and high performance memory cache for each processor in a multiprocessor architecture. With this design, whenever closely-related tasks are assigned to different processors, processors rely on an interconnection of quick paths (the QuickPath Interconnect) to access each other’s caches and memory for the related information - an architecture called NUMA.

However, when QPI memory checks between processors occur frequently, as they do in common network policy control applications, processing throughput suffers immensely. Many packet-processing solutions are vulnerable to massive performance degradation as a result of QPI memory checks, and only by understanding the issue can an operator properly evaluate competing alternatives.

To maximize packet-processing performance in multiprocessor environments, QPI memory checks must be kept to a minimum.

In the worlds of policy control and packet-processing, the only way to completely avoid QPI memory checks is to maintain processor affinity by ensuring all packets associated with a flow, session, and subscriber are processed by the same CPU.

While there are architectural accommodations that can slightly decrease the number of QPI memory checks, only a solution that steers packets to a particular core (by flow, session, and subscriber) will maintain complete affinity.
Introduction to QuickPath Interconnect

When choosing vendors and computing platforms for complex tasks such as applying policy control to a communications network, most evaluators examine product datasheets and subject short-listed vendors to lab trials. It is rare for an evaluator to investigate a potential solution to a degree such that the internal CPU architecture is a point of discussion; however, the CPU architecture can have a significant impact on real-world performance.

By understanding these issues, evaluators are able to scrutinize vendors’ datasheet claims and construct lab tests that reveal true performance. One such issue of critical importance to packet-processing applications, including deep-packet inspection (DPI) and network policy control, is Intel®’s QuickPath Interconnect architecture.

Prior to 2008, Intel processing technology employed a shared bus system in which all traffic was sent across a single shared bi-directional bus. Also known as a front-side bus (FSB), this bi-direction bus bought in multiple data bytes at a time for processing. As Intel attempted to increase the frequency of the wide-source synchronous buses, this design encountered electrical constraints.1

Over time, the FSB evolved and ultimately arrived at a four-FSB design with individual high-speed interconnects to each CPU, as shown in Figure 1.2

![Figure 1 - Intel front-side bus (FSB) memory access design, circa 2007](https://www.intel.com/content/www/us/en/io/quickpath-technology/quick-path-interconnect-introduction-paper.html?wapkw=quickpath)

In 2008, with its Nehalem and Tukwila processors, Intel introduced a new technology called QuickPath Interconnect (QPI). To promote better scalability, modularity, and resilience, QPI introduced a separate, integrated memory controller and high performance memory cache for each processor in a multiprocessor architecture.3 With this design, computational tasks are assigned to different cores that each access an integrated local memory cache.

---
3 Ibid, page 8.
Performance in this architecture is better than the FSB design because the processors no longer compete over memory access. However, whenever closely-related tasks are assigned to different processors (e.g., processing multiple packets of a single application flow), the processors must access each other’s caches and memory for the related information. To do so, they rely on an interconnection of quick paths between processors, as shown in Figure 2 - an architecture called non-uniform memory access (NUMA).

Stateful packet processing devices (e.g., NAT, firewall, DPI, or network policy control) perform multiple memory operations per packet (e.g., flow-state lookup, signature analysis, stateful modifications, counters, etc.), and these memory operations depend on previous packets. These memory operations in turn deal with a specific memory controller, location, and cache. If all processors access the same memory location, then a severe bottleneck is created due to cache pollution and latency/interlock on the interconnect bus.

Intel refers to the inter-processor memory check communication that is required to maintain memory state as “snoop traffic”. When QPI memory checks between processors occurs frequently, as they do in common network policy control applications, processing throughput suffers immensely.

Therefore, understanding the conditions that cause these QPI memory checks and cache pollution, and the potential design and deployment workarounds are of vital importance to any telecommunications network engineer who is tasked with evaluating or deploying DPI and policy control appliances.

---

4 Ibid.
Considerations for Stateful Packet Processing

The QPI memory snooping issue essentially means that the expected throughput of a packet-processing appliance reliant on these CPUs is wildly variable, making deployment dimensioning a difficult task. As an added complication, while some tests will reveal the performance drops, many others will not; as a consequence, two evaluations of the same platform might reveal dramatically different throughput numbers.

Real-world network policy control deployments are characterized by bidirectional flow of transactional traffic between senders and receivers, with many activities relating to previous packets (e.g., flow-state lookup, signature analysis, stateful modifications, counters, etc.), and these conditions are practically a worst-case scenario for QPI memory checks.

Processor and Core Affinity

To maximize packet-processing performance in multiprocessor environments, QPI memory checks, cache pollution, and inter-cache references must be kept to a minimum. To achieve this objective, a solution must maintain “Flow Affinity”.

Packets belonging to a specific flow should be processed by the same core. Flow affinity is especially important for TCP. TCP is a connection-oriented protocol, and it has a large and frequently accessed state that must be shared and protected when packets from the same connection are processed. Ensuring that all packets in a TCP flow are processed by a single core reduces contention for shared resources, minimizes software synchronization, and enhances cache efficiency.

In other words, to process flows as quickly as possible, flow packets moving in both directions should be processed by the same core. This scenario is illustrated in Figure 3.

---

QuickPath Interconnect: Considerations in Packet Processing Applications

The opposite scenario, processor migration, is characterized by a processor needing access to state information that is stored in the memory of another processor.

But this view is overly simplistic - in the world of network policy control, the importance of processor affinity extends beyond flows to sessions and subscribers, and beyond processor affinity to core affinity. That is, for maximum throughput performance it is required that all flows in a multi-flow protocol, and all traffic attached to a single subscriber, be processed by the same core. Only in this manner is all the relevant state information available in the local memory, and only in this manner can QPI memory checks be avoided.

As an example, consider the SIP protocol: with SIP, there are multiple flows for data and one for control; attempting to do anything stateful with this flow (e.g., applying QoS, measuring duration, measuring quality, etc.) without maintaining session affinity will cause QPI memory checks to spike dramatically.

This strategy applies to any packet-processing device: whether it’s a firewall, NAT element, session border controller, or DPI platform, maximum throughput is only achieved when core affinity is maintained for the largest stateful group (e.g., flow, flow-bundle, session, session-bundle, subscriber). In any situation in which core affinity cannot be guaranteed, device performance is completely dependent upon the appearance or not of conditions that cause core and processor migration.

Network Routing Asymmetry and Processor Migration

By design, all communications networks include routing asymmetry - an environment in which packets take different routes between the same endpoints.9

Unfortunately for packet-processing devices, unless architectural care is taken (see “Ensuring Core Affinity”, later in this whitepaper) routing asymmetry all but guarantees that QPI memory checks will degrade performance.

Recall the ideal flow conditions of Figure 3; now, let’s examine the same processing architecture under conditions of asymmetric routing (Figure 4). Under these conditions, different packets from the same flow are seen by different cores and will cause QPI memory checks (i.e., snoop traffic).

![Figure 4](image)

Figure 4 - Under conditions of routing asymmetry, packets from the same flow are seen by different cores, requiring QPI memory checks

9 A comprehensive explanation of routing asymmetry and its implications for network policy control is available in the Sandvine whitepaper Applying Network Policy Control to Asymmetric Traffic: Considerations and Solutions
Cabling
While routing asymmetry is the most obvious example of a condition that causes packets from the same flow to go to different CPUs, the reality is that QPI memory checks will result from any case in which packets from the same flow enter ports assigned to different CPUs.

For instance, consider the case of performing aggregate quota management for a subscriber in a network in which the upstream and downstream directions are handled by ports assigned to different cores: with hundreds of flows to count on a per-subscriber basis, the amount of latency-inducing QPI memory checking between cores is considerable.

Other Compounding Factors
Whenever core or processor affinity is compromised, device performance suffers. The degradation is compounded when the high-level tasks involve frequent memory look-ups or the processing of many small packets.

Memory Look-Ups
Many packet processing applications require the device to maintain memory state: for instance, subscriber-aware policy control must take into account the history and entitlement of each and every user, and complex measurements like subscriber quality of experience apply to the lifetime of a flow rather than just the packet that is currently being examined.

Any activity that leads to a memory look-up via QPI memory check will contribute to further performance impairment.

Since many or most activities associated with policy control require statefulness, it is impossible to avoid these memory checks; consequently, the optimum strategy to maintain performance is to ensure core affinity so that the memory access is local.

Packet Size
Packet size on the Internet varies: while some protocols typically utilize relatively large packets, others (e.g., DNS, latency-sensitive video-streaming and voice applications) use much smaller packets.

When considering the throughput implications of packet size, not all sizes are equal.

Figure 5 shows how the CPU utilization of a packet-processing device is impacted by packet size: at smaller packet sizes, the CPU utilization is higher, even when accounting for the variation in throughput. Since the CPU cores are working relatively harder to process the small packets, this compounds the issue around QPI memory checks.

In an Internet packet-processing environment, one can’t predict or control the size of the packets that the appliance will see, so again the optimum strategy is to ensure core affinity.
Ensuring Core Affinity
To ensure core affinity in a packet-processing application, there are two main approaches:

1. External to the packet processing device: design a deployment that minimizes the risk of QPI memory checks and cache pollution
2. Within the packet processing device: architect such that core affinity is always maintained

Redesigning the Network
To limit QPI memory checks, a network architect can attempt to deploy the packet-processing devices in such a manner that processor affinity is maintained. For instance, the deployment can be at the subscriber service edge, where there should be no asymmetry.

In this scenario, one would have to take care also to avoid cabling in which the upstream and downstream are connected to different CPU cores.

The most obvious downside to this approach is a much higher cost of deployment, since deploying to avoid asymmetry will typically require hundreds of packet-processing devices.

A secondary and more long-term problematic downside is that architecting around a problem only works if all future network modifications don’t inadvertently reintroduce any of the conditions in which processor migration occurs.

Attempting to design around the problem ultimately requires a network engineer to take on significantly more cost and to sacrifice future flexibility.

Flow Director
In a multiprocessor environment, flow packets are typically assigned to individual cores by an Intel network interface card (NIC) technology called Ethernet Flow Director. Flow Director tries to ensure that all packets from the same flow are assigned to the same processor.

---

QuickPath Interconnect: Considerations in Packet Processing Applications

On the surface, this approach seems to address the issue; however, the IEEE Communication Letter *Why Does Flow Director Cause Packet Reordering* conducted experiments that showed a significant amount of packet reordering due to process migration across cores when multiprocessing is configured for peak performance, so it seems that this approach does not provide a complete solution.\(^\text{11}\)

Intel is continuing to update Flow Director and related technologies\(^\text{12}\), but until major advancements are made there will continue to be performance problems.

**Network Processing Unit**

At present, the only way to completely avoid QPI memory checks in a packet-processing application is to ensure that all packets associated with a flow, session, and subscriber, are processed by the same CPU. To achieve this result, two conditions must be met:

1. There must be an aggregate solution to resolve network asymmetry by ensuring all packets relating to a particular flow, session, and subscriber go to the same packet-processing device\(^\text{13}\)
2. The packet-processing device must include functionality that specifically directs associated packets to a common processor core

The functionality described in the second condition can be generally referred to as a network processing unit (NPU). The NPU is the first point of inspection for incoming packets, and is dedicated to maintaining flow, session and subscriber affinity for maximum element throughput.

In the case of specific cabling requirements, the NPU also automatically removes this local asymmetry within packet-processing element by steering packets from the same flow to a single core, and then back out through the appropriate exit port.

Functionally, here is how an NPU would work:

1. Incoming packets are first inspected by the NPU to determine whether the traffic even needs to be inspected (i.e., passed to a CPU). For example, depending on the policy, traffic belonging to certain VLANs may not be inspected, which may be desired if the service provider chooses not to inspect traffic that belongs to a wholesale customer or business customer. Simply performing this task in the NPU already achieves performance advantages over equipment that requires CPU examination of all traffic.
2. For those packets that should be sent to a CPU, the NPU creates and relies upon a map that determines which core will process particular flows, sessions, and subscribers, and directs the packets appropriately. This mapping ensures that the same core is always used for all packet-processing relating to a specific flow, session, and subscriber. To preserve performance, the map must scale by the number of cores in the system, rather than packets per second.
3. Once the CPU has completed its tasks, the NPU returns the packet through the appropriate exit path.

In essence, the NPU can be thought of as a Flow Director that is specifically designed for policy control and packet processing applications, and which completely eliminates QPI memory checks and maximizes device throughput.

---

\(^{11}\) Ibid.
\(^{12}\) Including receive side scaling (RSS)
\(^{13}\) To learn about all the ways this can be achieved, please refer to the Sandvine whitepaper *Applying Network Policy Control to Asymmetric Traffic: Considerations and Solutions*
QuickPath Interconnect: Considerations in Packet Processing Applications

Conclusions
When designing a packet processing deployment (e.g., DPI or network policy control), device performance of Intel-based chips can vary wildly due to the impact of QPI memory checks. Even if two devices both rely on Intel chips, and are deployed in precisely the same way, the performance can vary due to internal architectures.

To appropriately plan deployments and evaluate vendors/equipment, engineers and evaluators must understand the circumstances and design decisions that lead to QPI memory checks.

The only way to completely avoid QPI memory checks for packet processing applications is to employ an architecture in which network routing asymmetry is resolved outside of an individual device and a network processing unit is used to direct packet flow within a device.

Questions to Ask About Packet-Processing Platforms
To determine whether or not specific equipment is vulnerable to the unpredictable, but potentially debilitating, performance degradation that results from QPI memory checks, an evaluator can start by asking some simple questions:

- Does this equipment rely on an Intel CPU that has the QPI feature?
  - If so, then how is core affinity maintained?
- Does the configuration of the cabling impact device performance?
- Does routing asymmetry impact device performance?
- How does a multi-device deployment address routing asymmetry?

Conveniently, many of these questions can be answered within the lab.

Additional Resources
There are several resources available beyond those which were cited in a footnote:

- To hear this issue explained and animated in a video, please click here
- To learn about Sandvine’s solution, please read the technology showcase: Maximizing Performance with Processor and Core Affinity

Invitation to Provide Feedback
Thank you for taking the time to read this whitepaper. We hope that you found it useful, and that it contributed to a greater understanding of the nuances of stateful packet processing on multi-core architectures.

If you have any feedback at all, then please get in touch with us at whitepapers@sandvine.com.