



Content Aware Routing

A NEW ARCHITECTURE FOR MAXIMIZING MULTICORE PERFORMANCE

By Yehiel Engel, Chief Architect, Commex Technologies

yehiele@commextech.com

Challenge

Currently all traffic arriving from I/O ports (e.g. network port or storage port) to server/appliance passes through the platform hub (chipset) and is sent to the server processor(s)/core(s). The processor that receives the data is responsible for classifying and directing the data frames to their final destination. This destination can be another I/O agent, either a network port or local/remote storage (i.e., disk).

In most cases the server processor only needs to inspect and process a small portion of the data frame (header or header fields), or even make automatic decisions (i.e., send compressed packets to decompress engine).

Due to the current nature of I/O traffic, all frame data is sent to the processor(s), resulting in the following:

- Increase traffic via the platform hub bus interface (PCIe or HT)
- Increase traffic towards the processor host memory (in NUMA architecture) or towards the host shared memory (in non-NUMA architecture)
- Increase cache pollution caused by redundant data loaded to the processor cache
- Increase processing latency as more data is written to the host memory which in turn, becomes a bottleneck
- Increase power consumption due to increased access to the host memory, more traffic on local bus, more processing on processors, etc.

In recent years, these phenomena have worsened due to the introduction of multi-core/multi-processor environments and the increase in I/O BW from networks and storage, currently network ports of 10Gb/s, with up to to 40Gb/s and 100Gb/s in the future.

The overall effect is that most users discovered that their I/O performance did not scale well after moving from a single processor single core to a multi-processor multi-core system. Moreover, they found it difficult to scale their system to process and handle traffic in the range of 10Gb/s and higher.

Recommended Solution

To solve the problems described herein, we recommend improving platform architecture, making it the base for future chipsets. Architecture is based on content aware routing of incoming traffic. As part of our solution, we will first address the previous and current generations of platform architecture.

Previous generation chipsets

In the past, chipsets were aimed at supporting three main functions:

- Interface to external memory (SDRAM, DDR)
- Interface to graphic card
- Bridge to local bus (PCI, PCI express) used to connect to various peripherals (i.e., networking and storage)

Figure 1 below describes initial platform architecture. We see a processor connected to a chipset (generally based on two devices - The Northbridge (Memory Controller Hub (MCH)), and The Southbridge, (I/O Controller Hub (ICH))). The chipset contains an integral memory controller, graphic interface, and a bridge to the platform's local bus and peripheral interfaces.

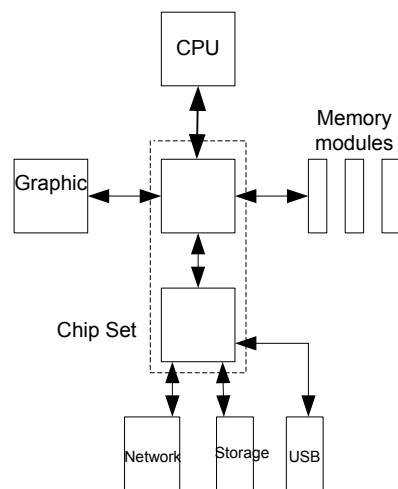


Figure 1: Initial platform architecture

In recent years, we have witnessed a migration towards NUMA architecture (Non-Uniform Memory Architecture), where the memory controller becomes part of the processor. This is carried out in order to solve shared memory bottleneck problems that have arisen due to the increase in the number of cores.

Additional trends include:

- Integrating the graphic core as part of the processor
- Allowing direct connection to the processor (i.e., using HyperTransport in AMD processors)

Based on these trends, we currently have only bridge functionality in the chipset.

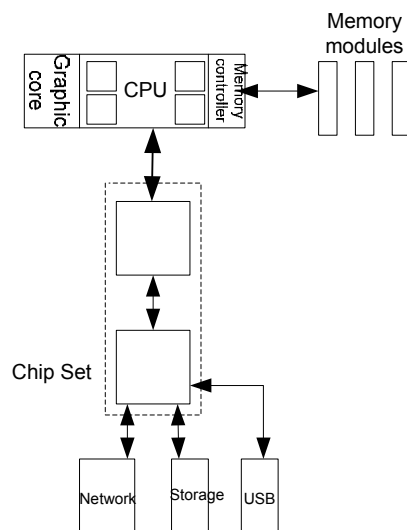


Figure 2: Current platform architecture

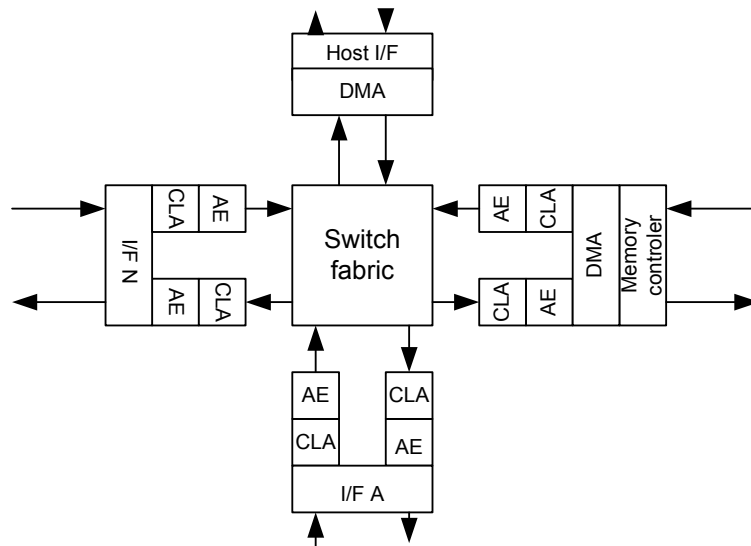
In parallel, the amount of traffic now handled by some interfaces (LAN or SAN) has increased dramatically (from 1G to 10G), and as noted above, the current trend is to move towards multi-processor multi-core environments.

These trends have led to the need to define new chipset architecture that will handle not only data transfer, but more complex tasks, such as load balancing, higher levels of traffic classification, persistency, bandwidth management, and other tasks that will result in enhanced performance and improved I/O scaling.

New Chipset Architecture

We recommend new chipset architecture to effectively handle the challenges described above.

The new chipset is detailed in Figure 3, below.



*Figure 3: New chipset - recommended architecture
(based on Commex Technologies' patent pending technology)*

The new chipset includes the following:

1. **Switch fabric** – Previously, incoming traffic was first sent to the host memory, processed by the host processor(s) and then (if required), sent back to other I/O ports. We recommend a different approach: transfer “decision” elements to the chipset (via programming), so it will be able to transfer traffic that does not require any processing or traffic that needs external processing (decryption or decompression) prior to host processing to the relevant port. For this reason, we require a switch fabric in the chipset that will allow such forwarding between all ports.
2. **Classification engines and action engines** – “Smart” classification that will enable better understanding of the traffic's nature in fine grain resolution (i.e., per packet or per flow base). This will allow improved and more accurate traffic management and decisions. Further, it will result in rich sets of decisions that will not only include filter/direct decisions, but also the option to send traffic to another port, plus better granularity about the specific destination core and the portion of data needed to be sent (all packets, headers, and descriptors). This will achieve improved destination core granularity and superior data transfer). Action engines must carry out various

classification decisions, including filtering traffic, forwarding traffic to its destination, and modifying packet fields (add, remove, or modify various fields). In addition, action engines are used to offload tasks for outgoing traffic (i.e., checksum, time stamping).

3. **DMA engine** – The DMA engine is used to handle traffic sent to the various processors and cores. Physical functions or DMA channels per core with different programmable parameters (i.e., priority, discard policy, interrupt policy, etc.), allow flexibility and QoS. Descriptor-based DMA enables sending as part of the descriptor information on incoming traffic extracted during classification, and reduces processing time (if only the descriptor is sent to the host memory, it saves I/O and memory traffic).
4. **Memory interface** – Adding physical memory to store traffic that does not require further processing by the host processor saves I/O and host memory BW., The processor should only receive the traffic parameters (i.e., from the descriptor or header) and send direction decisions to the chipset. This memory is used to handle new problems that arise from NUMA architecture where traffic, required processing or not, is sent to the host memory.

The following sections include detailed descriptions of the chipsets' parts (function, requirements, etc.).

Classification engines

In order to perform content aware routing on incoming traffic, one of the main components is to add a "smart" classification engine per I/O interface.

Current I/O cards (i.e., network cards) perform simple classification (L2 MAC, VLAN, or RSS hash function), however, classification is limited and decisions based on classification results are mainly to drop (filter) or forward the packet.

"Smart" classification should provide the following enhancements:

- Fine grain resolution that will allow it to classify and direct specific sessions/flows to a specific core.
- Ability to use different QoS for different protocols (i.e., low latency for real-time applications)
- Better knowledge (per flow) of the data required to be sent to the host processor/core.
- Ability to have more decisions, including sending traffic to one of the other ports and not directly to the host port (with/without modification). An encrypted packet, for example, may first be sent to a decryption engine and only then, will it be sent to the host memory.

Implementing classifiers to support higher BW and PPS rates is a challenge, and working with different types of classifiers is a difficult task. Today, however, there is a trend towards Converged Network Adapters, (CNAs) that support both data networking (TCP/IP) and storage networking (FC) traffic on a single I/O adapter, reducing the number of classifiers required.

Switch fabric

The switch fabric enables fast, low latency data transfer from any input port to any output port.

The switch fabric should provide the following:

- Allow any-to-any traffic in parallel
- Transfers should be non-blocking
- Inter-connect mechanism should be low latency and easy to implement
- Multiple queues to support QoS (i.e., allow bypass of urgent traffic).
Each queue should include input and/or output memory buffers to compensate bursts and allow smooth traffic flow
- High speed and high BW

DMA engine

The DMA engine is used to handle the traffic to/from the host processor(s)/core(s).

The DMA engine should implement the following:

- Descriptor-based DMA to provide easy and improved control of data buffer, and allow insert traffic information to ease processing and (optionally) reduce traffic to the host and host memory
- Multi-buffer should support sending traffic segment (i.e., header) to one memory location and remaining traffic (i.e., payload) to other another location
- Multi-channel should be supported with a minimum requirement of one channel per existing core
- Each channel should be independent from one another
- Each channel should be programmable and allow for dynamic changes

External memory

External memory is used to store data buffers that are not required to send to the host memory for further processing. External memory should be mapped as part of the processor's physical memory (i.e., as part of the PCI memory I/O addresses space). The chipset should include mechanisms to synchronize the redirection commands derived from the processor with the stored data.

Other issues – interrupt mechanism

Since the problem is a platform problem, other issues should also be considered, One of the major issues is the interrupt mechanism. The chipset should be able to send interrupt messages to specific cores. It should also be able to send interrupts per a (programmable) amount of traffic (i.e., once per 100 Ethernet packets) or per amount of time to reduce the amount of interrupts in the system. Current MSI-X mechanism enables per core interrupt, while the coalescing mechanism (per packet, per time) allows the requirement to reduce the number of interrupts.

Summary

In recent years, we have seen new technologies and solutions that aim to solve specific problems in the platform hub (chipset). These solutions include multiple DMA channels to support the multi-queue environment, MSI-X interrupts that allow per core interrupts, SR-IOV and MR-IOV in the PCI express arena that attempt to solve virtual function issues, platform hub switches for improved data convergence, and more.

We assume that these new technologies play an important role in improving overall performance, while a comprehensive solution is needed for the chipset. The new chipset should handle new tasks, including higher level classification, improved service quality, sophisticated routing, and per core data transfer persistency. These tasks, many of which are derived from the networking area, will allow better I/O scalability, reduce the unnecessary traffic and cache pollution, and reduce power and latency. The recommended architecture described herein provides a full solution using innovative content aware routing and a higher level of classification to address the overall platform challenge and not only one aspect of the problem.