

# Merits of Data Reuse Hints

## White Paper

**Merits of Data  
Reuse Hints**

## Introduction

Traditionally, much of the I/O system architecture and design was done with the assumption that I/O data rates and latencies are very large when compared to the throughput and latencies observed inside a CPU core or the memory subsystem. This has changed with the advent of higher data rates, for example 10 GbE, approaching cache miss latencies on the wire. Current device to CPU communication does not take full advantage of system capabilities such as the system cache hierarchy and effective utilization of system resources (memory bandwidth, etc.). We can improve I/O features to allow for better performance/power in future systems. Relevant system metrics include CPU utilization, memory bandwidth, latency, and power. This paper discusses how, by providing explicit cache management hints, we can significantly reduce I/O to memory bandwidth utilization, system interconnect bandwidth, and associated power consumption.

**Mahesh Wagh**  
**Anil Vasudevan**  
Intel Corporation

August 2008

## White Paper: Merits of Data Reuse Hints

### CPU accesses data DMA'd in and out by an I/O device

- Control data, for example, descriptors
- Headers for protocol processing
- Payload for data copies

The table below describes the problem statement and the power/performance impact to system architecture for I/O access:

Problem	Impact
<b>CPU Execution</b> CPU makes multiple "forced" trips to memory. (For example, during the network I/O processing data flow)	Higher CPU utilization due to "compulsory" cache misses. High bus bandwidth consumption. Per core throughput reduction.
<b>System Resource Consumption</b> Memory bandwidth consumed	Reduces available bandwidth for other applications. Increased power consumption.

For many applications there is temporal reuse of these data structures, for example recycling of data buffers, updates to descriptors, etc. The rate of data control exchanged between I/O device and CPU depends on application and other system variables. For example, at 10 GbE, the rate of control exchange is such that there is a very high likelihood that the accessed data is resident in the system cache hierarchy when accessed by I/O device or CPU, if this is allowed by the cache system implementation. However, keeping I/O data in system cache is not always desirable. So, many existing system architectures perform to "invalidate and write to memory" for device initiated accesses, for example, I/O initiated reads and writes cause cache evictions if addressed data present in the cache hierarchy. For the case where keeping data in cache is desirable, the penalty caused by the above behavior is in current systems on the order of ~50-80 ns. For reference, packet arrival rate using 10 GbE for 64 bytes is ~67 ns.

Note that in many cases it is not desirable to cache all I/O traffic. For example, modern operating systems typically implement a speculative read-ahead buffer for the disk. Caching such data, which may easily be many megabytes, would needlessly pollute system caches with data that may never be used.

To implement a successful I/O cache management policy it is necessary to have information about the intended use of I/O data.

CPU execution and system resource utilization can be significantly improved by providing I/O system architecture enhancements that result in reducing cache misses; for example, information about the I/O to CPU access patterns can be utilized by cache management entity to maintain data residency in caches. Lack of temporal information on I/O requests results in sub-optimal power/performance caused by inappropriate speculative memory access or undesired serialization of memory accesses. The impact is higher power consumption, and potentially increased memory and system fabric bandwidth. Information about the intended use (that is, a hint provided by device) enables system architecture data flows to be optimized based on request hints.

### Potential Solution: Data Reuse Hints (DRH)

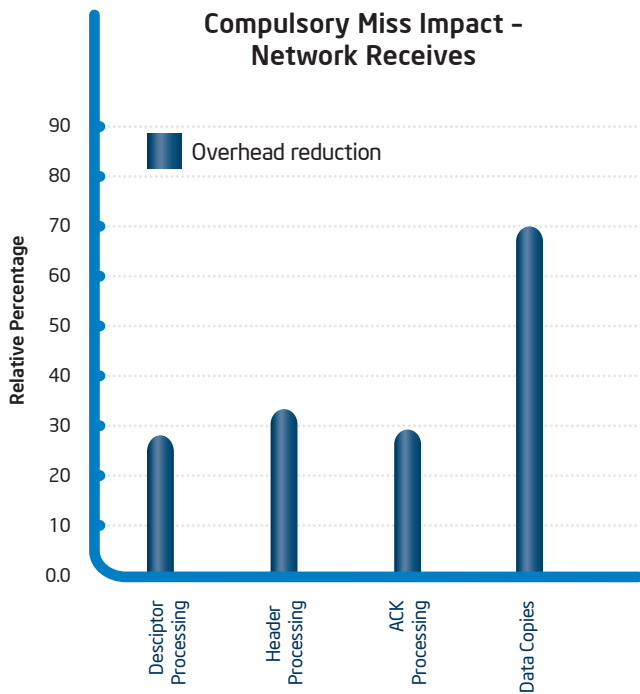
To reduce latency, increase throughput, reduce power, and improve platform efficiency for data transfers to and from I/O devices requires knowledge of the device/CPU communication patterns. The I/O device/application (device or device driver) are the best entities to provide this information, and Data Reuse Hints (DRH) are proposed as the associated PCI Express\* (PCIe\*) protocol mechanism.

For example, in network I/O data flows, these hints would be provided for following data structures:

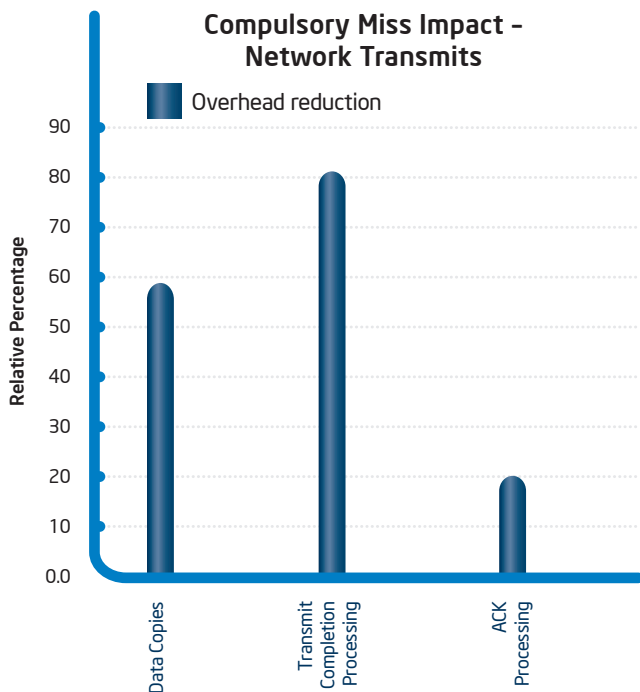
- Control structures such as headers and descriptors for inbound and outbound.
- Optionally, for payloads that are copied to multiple locations.

The figures below show the CPU execution improvements as a result of system hardware effectively utilizing the DRH provided by an I/O device on PCIe requests. This data has been generated from measurements using highly optimized hardware and software, and represents an accurate apples-to-apples comparison of the differences one would see in a well designed system.

**Overhead reduction = % reduction in time spent on CPU on specified tasks**



**Figure 1:** Compulsory Miss Impact - Network Receives on Linux\*. Source: Intel Corporation.



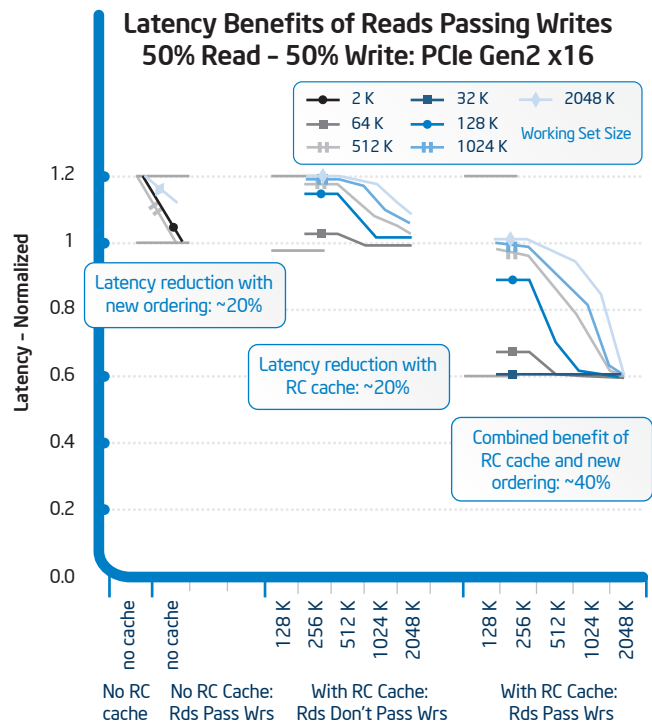
**Figure 2:** Compulsory Miss Impact - Network Transmits on Linux\*. Source: Intel Corporation.

As shown in the figures above, the DRH can be used to significantly reduce the overhead for network operations caused by cache miss-induced memory access penalties. The improvement for network packet receive is in the range from 25 to 70 percent for various processing primitives (descriptors, data copies, etc.), similarly the overhead reduction for

network transmit is in the range of 20 to 80 percent for transmit processing primitives (ack processing or data copies). This reduction in overhead is achieved by maintaining data residency in system caches based on communication pattern hints provided by PCIe devices. The net effect of these combined improvements is a double-digit percentage improvement in the performance of the overall flow.

In addition to reducing the CPU overhead DRH provides a capability for system data flows to be optimized for effective utilization of resources such as memory and system interconnect bandwidth. For example, when a request is most appropriately serviced out of cache, this frees memory bandwidth to allow improved performance for other activities. There is also an implied power reduction by lowering the utilization of critical platform resources such as memory and system interconnect bandwidth.

Appropriate caching of I/O data can significantly improve I/O read latencies. This improvement is largely additive when combined with ordering improvements, as discussed elsewhere. The data in Figure 3 was taken from an accurate system simulator using a synthetic traffic pattern, and shows that the incremental reduction in I/O read latencies is about 20 percent. The system modeled included a relatively shallow memory topology, and the benefits would be larger in a system with a longer latency memory system.



**Figure 3:** Example Read Latency Benefit to Reads Passing Writes. Source: Intel Corporation.

# Summary

DRH can be applied in a variety of scenarios (applications) to improve multiple system metrics at the same time.

- DRH provides device communication pattern information
  - For example, control structure information such as headers and descriptors
- DRH can be effectively used by I/O system architecture to
  - Facilitate effective cache management, improve application performance
  - Reduce memory bandwidth, system bandwidth, and associated power
- DRH can be widely applied to I/O (network and storage) and to emerging accelerator device/applications such as visualization, math, high-performance computing, etc.
- Supporting the cache hierarchy to comprehend I/O accesses reduces variability in access latency to system memory
- Caching provides significant latency benefits from I/O devices' perspective

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order. Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web site at [www.intel.com](http://www.intel.com).

Copyright © 2008 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

\*Other names and brands may be claimed as the property of others.

Printed in USA

0808/VP/HBD/PDF

 Please Recycle

