A Fabric That Adapts to Big Data On-The-Fly

Overview

Big Data applications are growing exponentially in volume as the computing public's reliance on online commerce and entertainment increases. But the diversity of Big Data applications is also expanding, as businesses sort through volumes of market data to discover trends, and smart machines are being linked to share production information. In fact, many promising data-intensive computing trends, such as the "Internet of Things" (IoT), are just starting to get traction. With this growth have come some difficult challenges for the architects and users of the computer systems that analyze the data. It may seem paradoxical, but one characteristic of the Big Data analysis problem is that the information that one typically wants to obtain from the data depends on the data itself. It often isn't possible to pre-process data to simplify the data analysis problem. Systems need to be able to respond to the data in a dynamic way. These factors combine to make efficient processing of Big Data among the greatest problems in computing today.

Simply applying more processors to the problem of Big Data processing is not a solution. In fact, adding processors can make the processing problem worse, as bottlenecks develop in the inter-processor communication links as the data is shared. Application performance is limited by the ability to allocate resources and move data, not the time required to process it. Lightfleet has the solution, but it required a complete re-thinking of the way that fabrics are built.

Modern distributed computing systems typically use messaging to transmit data packets between processors using standard network fabrics such as Ethernet or InfiniBand. These networks can experience heavy congestion when multiple processors need to access a shared memory. Implementing shared memory with maximum performance, using today's networking technologies, requires building as much of the data transfer mechanism into the hardware as possible. This need has spawned network protocols that bypass the local CPU, such as iWARP and RDMA over Converged Ethernet (RoCE), to transfer data packets into processors' local memories. Congestion is still an issue, however, and data packets can be dropped when the system becomes overwhelmed. Because of this, most existing fabrics have evolved a means of recovering from message failures, but system performance and complexity suffer.

Rigidity Impacts Determinism and Flexibility

For typical network fabric architectures, keeping performance, cost and hardware/software complexity in balance requires that a single method of memory-sharing be implemented between cooperating processing elements. It is typically impractical to reconfigure or reallocate memory spaces between processing tasks, which would otherwise provide a performance advantage in applications where the processing workload varies. In order to improve application performance, processors typically use local caching of data, but that increases overhead and software complexity when such data must be shared.

Dealing with these problems has produced technologies such as ccNUMA, where inter-processor communication between the cache controllers is used to keep a consistent memory image when more than one cache stores the contents of the same memory address. The overhead and software complexity problems persist, however, and these factors can severely impact multi-application environments where maintaining time determinism is a factor. Such applications include embedded-computing systems that combine real-time control functions with data processing. If the integrity of critical control parameters is not precisely maintained due to an unresponsive network, the system may crash or at least be unreliable.

Simply designing smarter network switches won't fix latency and reliability problems in the long run, however. Providing scalability and adaptability within the environment of today's fabric architectures presents difficult software problems whose workaround solutions extend latencies and place limits on throughput. But a more serious problem is the fact that the problem fixes can't easily be scaled to meet future needs, many of which can't be known today.

A Flexible Fabric for the Future of Big Data

The solution is an intelligent network fabric architecture that enables flexible groupings of processor hosts that can be reconfigured in real-time as the processing workload changes. Rather than typical network architectures that are optimized for point-to-point transfers, and managed by a network control hierarchy, a more flexible architecture supports information transfers that are data-directed, requiring no control plane to intervene in setting up transfer paths. No control plane means no switching overhead to extend transfer latencies in large fabrics. Multicast transfers would be the default mode, while unicast transfers would be handled as single-channel multicast transfers. The structure of the fabric would be continuously adaptive according to the needs of the data. The data flow control function (the protocol stack), which used to be implemented in software, would be built into the hardware of the fabric connections, and different data-flow types would be handled automatically.

The result of such an architecture would be a huge reduction in software size and complexity and dramatically-reduced latency and network congestion. Memory access would appear to users as though all memory is shared, and the access latency would stay low even as the message rate increases.

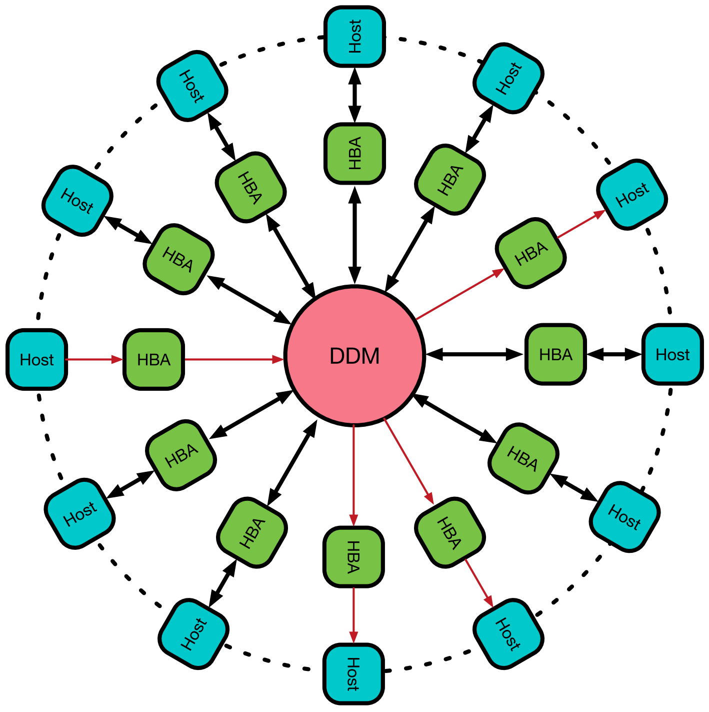
Without the need for a control plane, the system would have built-in scalability. Fabric nodes can be snapped together as the data environment grows and the data can still find its way. Maintaining memory coherency, which is a problem for switched networks, can be handled simply. Implementations of content-addressable memory are easier to construct. In computing environments handling diverse processing workloads, the fabric can react to the problem being handled in order to provide optimal connectivity for the application so that available processing power can be used most effectively.

The Lightfleet DDS

Just such a fabric is the Lightfleet data-distribution system (DDS). The DDS is an adaptive interconnect designed to support parallel and distributed computing applications, based on the concept of self-directed data flow. The Lightfleet DDS consists of one or more interconnected data-distribution modules (DDMs) that send and receive message packets to and from host-bus adapters (HBAs) mounted in the commodity computers or blades comprising the distributed computing system.

The functionality of the DDM and the companion HBAs extends across the first four layers defined by the Open Systems Interconnection (OSI) abstract model which conceptually divides an ideal communication system into seven layers. Since Lightfleet's interconnect system is an ab initio design meant to provide high performance with low transport latency for closed systems, the OSI model served only as a rough guide in its development. Strict adherence would have forced compromises with the stated goals. For this reason, while the hardware and firmware comprising the DDS properly implement Layer 1 and Layer 2 of the OSI model, they also include functionality that is typically associated with Layers 3 and 4. In this sense, functionality of the protocol stack is largely present in the underlying DDS hardware, with only the top layers relegated to the operating system or kernel of the interconnected hosts. Hence this design relieves the application programmer of the details of interprocessor communications, and avoids the performance penalties that software-intensive solutions place on data transfers.

Figure 1.
The Lightfleet DDS



Memory Flexibility Increases Efficiency

The purpose of the Lightfleet DDS is to ensure that the transport of information between processors is as efficient as possible. This is accomplished in hardware by a datagram protocol that directly supports the various messaging and direct-memory updates needed to implement all applications involving multiple processing workloads and multiple memory spaces.

To increase computational efficiency and to simplify the task of the application programmer, the communication between processes over the Lightfleet fabric occurs at the lowest possible OSI layer. As a result, the DDS design enables a memory-virtualization model wherein memory access appears to the user as if it were physically shared, simplifying the programming model. In addition to messaging, there are two other methods used to implement a shared memory on a distributed system. Each of these methods makes use of the direct-memory access (DMA) capabilities inherent in the HBA and allows remote DMA (RDMA) transfers between memory spaces of different processors.

The versatility of the DDS design does not limit a particular memory model to a particular computing architecture. For example, multiple publish-subscribe (PS) applications in a high-performance computing environment can operate simultaneously on a distributed multiprocessor by defining a working group of processors specific to each application. Any particular PS application may specify its own method of memory access during initialization depending on memory size and access requirements.

Flexible Messaging

The DDS hardware paired with the Lightfleet datagram protocol supports all messaging and message-passing techniques that are essential to distributed object-oriented programming. In addition, concurrency models such as the message-passing interface (MPI) are also supported. MPI is a communications protocol often used in high-performance computing applications. Messaging is mainly used to invoke behaviors and coordinate processes. Messages may be sent and received by memory-to-memory transfers where the destination is a memory location or messaging queue specific to a messaging middleware layer. Alternatively, network sockets may be used to send, receive, and process messages. This latter approach can easily be made consistent with Internet protocols such as used in client-server applications.

Summary

At the end of the day, Big Data applications are all about the data. Conventional network fabrics and evolutionary enhancements of same can’t adapt enough to handle the processing needs of Big Data applications in the future, without paying a large toll in software complexity and processing efficiency. A new fabric architecture is needed, and Lightfleet has responded with its new DDS architecture.

Lightfleet’s DDS delivers ease of shared-memory programming and scalable message passing in a single interconnect fabric, eliminating transfer bottlenecks and adding a new dimension of flexibility to high-performance computing. The fabric is self-adapting, allowing the use of versatile and efficient memory models that specifically address the needs of each application type. A distributed multiprocessor based on Lightfleet’s DDS may be partitioned into independent machines, each with its own and possibly different computing architecture. The memory-access protocols can support each of these architectures simultaneously in the same system, greatly increasing the flexibility and utility of a large distributed multiprocessor and eliminating the performance penalty due to conventional network switching operations.

Interested in evaluating Lightfleet technology? Lightfleet will provide all the details to qualified organizations. Please call Ivy Yap at 360.816.2840 or email iyap@lightfleet.com to order or discuss specific requirements.

Lightfleet®

For more information, visit
www.lightfleet.com

Lightfleet Corporation
4800 NW Camas Meadows Dr.
Camas, WA 98607-7671
USA