The TH Express high performance interconnect networks

Zhengbin PANG\textsuperscript{1,2}, Min XIE\textsuperscript{2}, Jun ZHANG\textsuperscript{2}, Yi ZHENG\textsuperscript{2}, Guibin WANG\textsuperscript{2}, Dezun DONG\textsuperscript{1,2}, Guang SUO\textsuperscript{2}

1 Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha 410073, China
2 College of Computer, National University of Defense Technology, Changsha 410073, China

Abstract Interconnection network plays an important role in scalable high performance computer (HPC) systems. The TH Express-2 interconnect has been used in MilkyWay-2 system to provide high-bandwidth and low-latency interprocessor communications, and continuous efforts are devoted to the development of our proprietary interconnect. This paper describes the state-of-the-art of our proprietary interconnect, especially emphasizing on the design of network interface. Several key features are introduced, such as user-level communication, remote direct memory access, offload collective operation, and hardware reliable end-to-end communication, etc. The design of a low level message passing infrastructures and an upper message passing services are also proposed. The preliminary performance results demonstrate the efficiency of the TH interconnect interface.

Keywords HPC, network interface chip (NIC), TH Express interconnect, offload collective operation

1 Introduction

As the second generation of massively parallel supercomputers in the TH series designed by National University of Defense Technology (NUDT), the MilkyWay-2 (TH-2) system topped the TOP500 list released in June 2013 [1]. The system consists of 16 000 compute nodes of each one equipped with two Xeon E5-2600 processors and three Xeon Phi accelerators. Such massive parallelism puts high pressure on the design of interconnection network. In order to fulfill the high communication requirement, the MilkyWay-2 system uses proprietary interconnect named TH Express-2 interconnect, which especially emphasizes on high bandwidth and low latency interprocessor communications [2].

While the computing nodes and system become more and more parallel due to architectural improvements and parallel programming paradigms, typically there is only a single network interface. Therefore, it is crucial to efficiently support such available parallelism in network interface [3,4]. We continually improve the design of network interface and message passing services based on the implementation in MilkyWay-2 system. Our goal is to provide highly scalable and efficient message passing services through the design at all levels, namely, hardware, operating system, and communication software. This paper describes the state-of-the-art of our proprietary interconnect, especially emphasizing on the design of network interface.

2 Network interface chip (NIC)

NIC provides the software-hardware interface for applications to access the high-performance network. As shown in Fig. 1, it contains a full width 16-Lane PCI-E 3.0 interface connected to the compute node, and uses its network port to communicate with interconnect fabric. NIC contains several advanced mechanisms to support scalable high performance computing, including protected user-level communication, remote direct memory access (RDMA), offloaded collective mechanism, etc.
Hardware resources virtualization plays an important role in implementing protected user-level communication [5–7], which provides each process an exclusive programming view for using communication hardware. When several processes run concurrently, communication operations from different processes are isolated without interference.

In order to support protected user-level communications, NIC exploits a mechanism named VP, which is a combination of a small set of memory-mapped registers and a set of in-memory data structures. The address ranges of registers in different VPs are spaced out at least the length of the physical page. All the data structures can be mapped to user space, so that it can be accessed in user space concurrently with protection. The related data structures are organized in several queues, including on-chip hardware descriptor queue (HDQ), and software descriptor queue (SDQ), minipacket queue (MPQ), event queue (EQ), interrupt queue (INTQ), error packet queue (EPQ) in memory. The detailed structure and usage will be described later.

2.2 Hybrid descriptor queue

To support multi-process simultaneously submitting descriptor, NIC supports up to 40 VPs in hardware. Among them, eight VPs receive descriptor through PIO (programmable IO) write and the other 32 VPs fetch descriptor through DMA (direct memory access). The two different types of VP are organized in HDQ and SDQ respectively.

Descriptor submission through PIO is easier by writing descriptors to the HDQ queue directly via PCI-E interface. However, the steps required to fetch descriptors through DMA in SDQ are more complicated. First, the software informs NIC to fetch descriptor by writing the number of descriptors prepared in host memory to a specific register within VP. Second, NIC schedules the candidate VPs which have descriptors to submit. Third, NIC requests to read descriptor in host memory through PCI-E interface. Finally, NIC receives the descriptor and stores it in the SDQ buffer.

Once the descriptors arrive at NIC, they are buffered internally, and then read out in round-robin sequence and submitted to corresponding processing units. Both the two descriptor queues have their specific advantages. Compared with SDQ, HDQ implementation has less PCI-E transactions and relatively lower latency from submitting descriptor to processing it. However, SDQ could accommodate much more descriptors. Therefore, the hybrid descriptor queue implementation supports both low latency and high capacity. It is up to system software to choose the proper VPs to submit descriptor.

2.3 Address translation and memory protection

In order to support RDMA in virtual address mode, NIC implements virtual address translation mechanism to translate virtual address into physical address. The whole address translation table (ATT) [8] is allocated in host memory, and NIC provides an on-chip Cache named ATC (address translation Cache) to reduce the long latency of main memory access.

Before address transformation, the legality of each virtual address must be checked. A memory checker is employed to detect the validity of virtual address. The memory checker contains 4 096 items, while each item is composed of several fields, including valid flag, key, read and write bit, base and top page address boundary. The memory checker serves several request sources, and adopts a round-robin arbiter to select one source each time.

The ATC is applied to bridge the gap between current long latency of main memory outside and high requirement for obtaining physical address. The Cache stores one million physical address items recently used. The Cache adopts eight ways set-associative structure and LRU (least recently used) replacement strategy. A deep non-blocking pipeline is implemented to improve performance and bandwidth. When the total number of physical address items used by software is smaller than one mega, the Cache can be configured to a large buffer to achieve substantial improvement on the performance of accessing the physical address. This buffer is initialized by writing all physical address items those will be used later, and then the latency of reading these items can be largely shortened.

It is important that the procedure of refreshing virtual and physical address items must be complete and correct. While the virtual and physical address items are newly added or