Front. Comput. Sci., 2014, 8(3): 345–356
DOI 10.1007/s11704-014-3501-3

RESEARCH ARTICLE

MilkyWay-2 supercomputer: system and application

Xiangke LIAO (✉)[1,2], Liquan XIAO[2], Canqun YANG[1,2], Yutong LU[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 On June 17, 2013, MilkyWay-2 (Tianhe-2) supercomputer was crowned as the fastest supercomputer in the world on the 41th TOP500 list. This paper provides an overview of the MilkyWay-2 project and describes the design of hardware and software systems. The key architecture features of MilkyWay-2 are highlighted, including neo-heterogeneous compute nodes integrating commodity-off-the-shelf processors and accelerators that share similar instruction set architecture, powerful networks that employ proprietary interconnection chips to support the massively parallel message-passing communications, proprietary 16-core processor designed for scientific computing, efficient software stacks that provide high performance file system, emerging programming model for heterogeneous systems, and intelligent system administration. We perform extensive evaluation with wide-ranging applications from LINPACK and Graph500 benchmarks to massively parallel software deployed in the system.

Keywords MilkyWay-2 supercomputer, petaflops computing, neo-heterogeneous architecture, interconnect network, heterogeneous programing model, system management, benchmark optimization, performance evaluation

1 Introduction

The techniques of high performance computing (HPC) are developing very rapidly during these years. HPC is considered as a strategic infrastructure to fulfill industrial and societal requirements. There continues to be considerable demand greater computing power, which is important to foster innovation and improve competitiveness in science and technology. HPC community has broken through the barrier of petaflops, and is keen to take the race for exascale computing. Grand challenges facing exascale computing, coming from architecture, programming model, application and many other aspects, need to be solved before the anticipated arrival of the age of exascale computing near the end of this decade. Following the successes of research efforts in Tianhe1A [1] and many other major systems in the last two decades, we continue to address the issues of performance, scalability, programmability, energy efficiency, and reliability, and explore new supporting technologies to achieve world-class scale computing capabilities.

MilkyWay-2 (Tianhe-2, or simply TH-2) supercomputer is another important milestone developed by National University of Defense Technology (NUDT) in post petaflops era. It is sponsored by the National High-tech R&D Program (863 Program) administered by the Ministry of Science and Technology in China. It was ranked No.1 on the 41th Top500 list in June, 2013, retained lead on the 42th Top500 list in November, 2013, after Tianhe-1A getting on the Top1 in November 2010. MilkyWay-2 is installed in National Supercomputing Center of Guangzhou (NSCC-GZ), and services as a nationwide computing infrastructure to facilitate scientific discoveries and technical innovations, and tackle key issues that concern the local government.

This paper is organized as follows. We first present system

Received September 6, 2013; accepted December 30, 2013

E-mail: xkliao@nudt.edu.cn

overview of MilkyWay-2 hardware and software in Section 2. The highlights of system components are then described in Section 3. We present the system optimization for HPC benchmarks in Section 4, followed by a brief summary of the sample applications running on MilkyWay-2. We conclude this paper with a future perspective for applications and impacts of MilkyWay-2.

2 System overview

Although MilkyWay-2, like its predecessor, Tianhe1A, still employs accelerator based architectures, it is largely redesigned and managed to make a careful balance between the need for technological advancement of proprietary components and the availability of existing components. The goals of the design are to provide high computational performance under acceptable total of cost, power budgets, capabilities of supporting reliability, availability, and serviceability (RAS), complexity of application development and porting.

The hardware system is composed of five subsystems, covering compute, communication, storage, monitoring and diagnostic, and service. MilkyWay-2 has four compute frames per compute rack, and 32 compute nodes, one switch board and one monitor board in one compute frame, where they are electrically connected by the midplane and packaged in a compact structure. MilkyWay-2 consists of 125 compute racks, thereby 16 000 compute nodes. Each compute node is equipped with two Intel Xeon E5-2600 processors, three Intel Xeon Phi accelerators based on the many-integrated-core (MIC) architecture and 64 GB memory, delivering a peak performance of 3 432 GFLOPS. All compute nodes are connected to the top-level switches by virtue of their local switch boards, following a customized fat-tree topology. The heart of the communication subsystem is the proprietary high-speed interconnection chips, including high-radix switch chip and network interface chip. The interconnection chips achieve a high bi-directional bandwidth of each port, which can simultaneously send and receive at 10 GB/s, and supports both collective and global barrier functions. A thin communication software layer is built upon the chips and enables applications to achieve low-latency and high-throughput. The service subsystem is composed of 4 096 nodes, packaged in eight service racks. It is designed to accelerate emerging information service applications that call for high throughput instead of computation, e.g., big data processing. These service nodes also serve as front-end nodes that provide a gateway to application development and administration for the MilkyWay-2 system. They are installed with software development and scheduler tools, such as compilers, debuggers and job launch commands. Each service node is powered by a proprietary FT-1500 CPU, which implements SPARC V9 instruction set architecture, has 16 cores and runs at 1.8 GHz and 65 Watts, and delivers a peak performance of 144 GFLOPS. The storage subsystem contains 256 I/O nodes and 64 storage servers with a total capacity of 12.4 PB, packaged in 24 storage racks. It exploits hybrid hierarchy storage architecture, which enables the shared storage with large capacity, high bandwidth and low latency. The monitor and diagnosis subsystem adopts the centralized management architecture with distributed agents, achieving real-time monitoring, convenient control, and accurate diagnosis for MilkyWay-2.

The software stack of MilkyWay-2 consists of four environments, including system environment, application development environment, runtime environment and management environment. System environment consists of the operating system, the parallel file system and the resource management system. The operating system is a 64-bit Kylin OS, supporting massive applications. The parallel file system adopts large-scale hybrid hierarchy storage architecture, which supports I/O aggregate bandwidth of above 500 GB/s. The resource management system gives a unified view of the resources in MilkyWay-2 system. Various job scheduling policies and resource allocation strategies are implemented so that system throughput and resource utilization can be effectively improved. Application development environment supports multiple programming languages including C, C++, Fortran 77/90/95, and a heterogeneous programming model named OpenMC, and the traditional OpenMP and MPI programming models. The parallel development tools provide a unity GUI which supports application tuning and profiling effectively. Runtime environment consists of the parallel numerical toolkit for multi-field of scientific applications, the scientific data visualization system, and the HPC application service and cloud computing platform. The environment provides programming and runtime support to multiple fields, including scientific and engineering computing, big data processing and high throughput information service. Management environment implements fault monitoring and autonomic management. The autonomic fault tolerant management system provides real-time error probing, fault diagnosis and maintenance guide. These functions significantly simplify the system maintenance process and effectively improve the system usability.

3 Highlights of system components

3.1 Neo-heterogeneous compute node

A MilkyWay-2 compute node is built using two CPUs and three coprocessors, as shown in Fig. 1. The CPU is a 12-core Intel Xeon processor E5-2600 v2 product family based on Ivy Bridge architecture clocked at 2.2 GHz, and the coprocessor is a 57-core Intel Xeon Phi 3100 family clocked at 1.1 GHz, which is based on Intel MIC architecture. Two CPUs are directly interconnected by two full-width Intel QPI links, and each CPU is configured with two 16-lane PCI Express (PCI-E) interfaces, eight DIMMs within four memory channels. A compute node can be configured with up to 128 GB memory using 16 GB DIMMs. One CPU, the upper one in Fig. 1, is connected to the platform controller hub (PCH). One of the PCI-E interfaces in the upper CPU is connected to the proprietary high-speed interconnection network through the network interface chip (denoted by NIC), and the other is connected to one Intel Xeon Phi coprocessor. Each PCI-E interface in the other CPU, the lower one in Fig. 1, is connected to one Intel Xeon Phi. CPLD chip is responsible for administrating the whole computer node via IPMB bus.

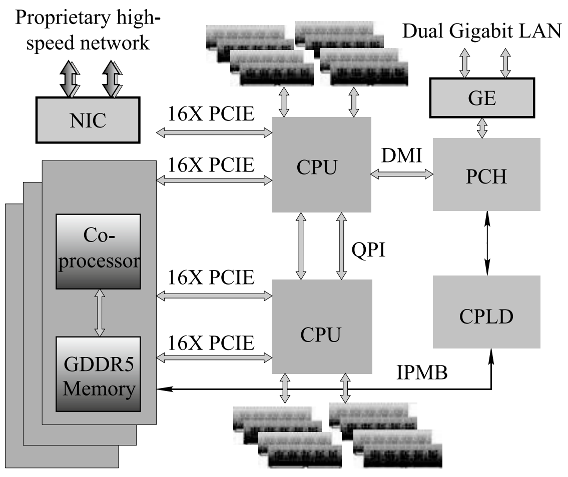

Fig. 1 The neo-heterogeneous architecture of compute node

Intel MIC architecture products run on standard, existing programming tools and methods, and use the same familiar programming model as the widely used Intel Xeon processors. That is, the same program source code (i.e., standard C, C++, FORTRAN, etc.) written for Intel MIC products can be compiled and run on a standard Intel Xeon processor. Comparing to multicore processor, Intel Xeon Phi offers many cores with wider vector processing units for greater floating point performance per watt. Furthermore, this coprocessor has a few advantages over general purpose graphics processing units (GPGPUs). For example, it can operate independently with CPUs and does not require special code. These features make Intel Xeon Phi allowing higher aggregate performance for today's most demanding applications in the most efficient and time-effective way. Therefore, Intel Xeon Phi coprocessor is selected in MilkyWay-2 to work with the Intel Xeon processor to increase developer productivity via common programming models and tools.

Since both Intel Xeon processor and Intel Xeon Phi coprocessor have similar instruction set architecture (ISA) but different arithmetic logic units (ALU), we call this hybrid architecture with a unified programming model *neo-heterogeneous* architecture. This kind of architecture has multiple classes of compute capabilities that are accessed by a common programming model, streamlining development and optimization processes, which are not possible when using a combination of CPUs and GPU accelerators.

3.2 TH Express-2 network

MilkyWay-2 uses proprietary interconnect, called TH Express-2 network for high-bandwidth and low-latency interprocessor communications. All network logic is developed and integrated into two specific ASIC chips, i.e., high-radix router chip and network interface chip. Both of them adopt efficient mechanisms to achieve high performance communications with regard to bandwidth, latency, reliability and stability. High-radix router chips are further used as basic building blocks to create switch boards of 48 ports and top-level switches of 576 ports. 32 compute nodes are packaged in one compute frame and connected by a switch board using an electrical backplane. A total of 500 compute frames are connected through top-level switches using active optical cables following a fat-tree topology, as shown in Fig. 2. TH Express-2 network leverages the features of high-radix router and network interface chips, and implements an optimized message passing interface (MPI) layer, and thus can support efficient execution of massively parallel message-passing programs from user space with minimal software overhead.

3.2.1 High-radix router

This high-radix router chip designed for MilkyWay-2, called HNR2, is an evolutionary upgrade of its predecessor in Tianhe-1A and offers improved performance. HNR2 switches data among 16 symmetric network ports. Each port has eight lanes with the bi-directional bandwidth of 160 Gbps. Some novel techniques are used in HNR2 such as dynamic buffer

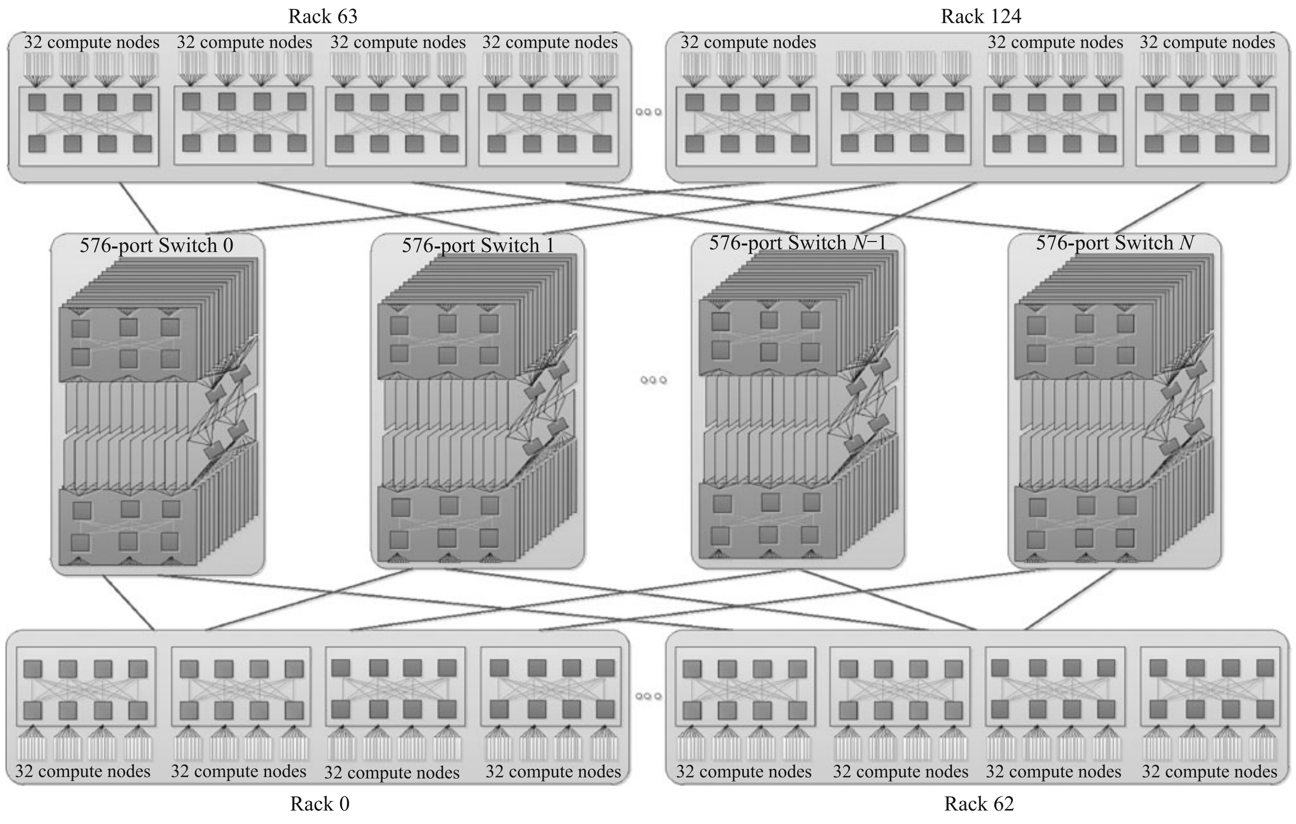

Fig. 2 TH Express-2 network architecture and topology

allocation, adaptive routing based on hierarchical look-up table, intelligent network management, low-latency scrambler, improved rolling CRC, etc.

Specifically, HNR2 adopts a hierarchical design methodology and splits the crossbar switch into interconnected smaller tiles that allow for efficient layout implementation. This design, however, calls for considerably more buffers compared to traditional input-queued switches with single crossbar. To mitigate the memory usage overhead, HNR2 uses customized dynamically allocated multi-queue (DAMQ) [2] to decrease the buffer size and improve the buffer utilization, while still meeting tight latency budgets.

HNR2 employs an adaptive routing method by using a hierarchical look-up table, and provides load balancing and fault tolerance. Upon the arrival of each packet head, multiple output ports are calculated based on the look-up table, and one of them is selected according to a utility function. These routing tables are constructed in a hierarchical fashion to reduce the storage overheads. Each HNR2 router or compute node in MilkyWay-2 supercomputer is assigned a unique ID which identifies its location in the packaging hierarchy. It is convenient to exploit hierarchical routing tables to support different routing methods in interconnection networks with regular or non-regular topology.

HNR2 provides a lot of management techniques to improve the RAS (reliability, availability and serviceability) capability of TH Express-2 network. Advanced functions, e.g., routing trace, link test, fault reporting, and topology discovery are supported by in-band management supports. Chip configuration and status query are achieved through IIC bus interface. These RAS features enable both real-time and historical status monitoring and alerting, facilitating fault locating.

3.2.2 Network interface chip

Network interface chip (NIC) provides the software-hardware interface for accessing the high-performance network. NIC contains several advanced mechanisms to support scalable high performance computing, including protected user-level communication, remote memory access engine, collective offload engine, reliable end-to-end communication etc.

A mechanism named virtual port (VP) is implemented in the NIC to support the protected user-level communications. Each virtual port is a combination of a small set of memory-mapped registers and a set of related on-chip and in-memory data structures, and the address range of registers in different virtual ports 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.

In order to speed-up collective communication in MPI, NIC provides offload mechanisms to accelerate collective operations. The software is required to construct collective algorithm tree and generate a collective descriptor sequence for NIC to initialize collective communication. However the collective descriptor sequence cannot be executed immediately, and should only be triggered to execute upon receiving a special control packet. When the descriptor sequence is executed, NIC can also perform a series of swap operations to modify the address and VP information of the descriptor in the descriptor sequence using the data from the control packet. To improve system reliability, a CRC in user level is used in data packets to ensure the data integrality. When user-level CRC errors occur, data packets will be retransferred.

3.2.3 Message passing service

To efficiently utilize the MilkyWay-2 interconnection fabric, we implemented the galaxy express (GLEX2) communication layer, which provides low-level user-space and kernel-space API to support the implementation of other software systems or popular programming models, such as TCP/IP protocol, parallel file systems, network booting system, MPI, global arrays, and etc.

The main programming model in MilkyWay-2 is MPI. The MPI implementation in MilkyWay-2 is mainly a high performance network module (Netmod) within MPICH's Nemesis channel. Thus highly optimized on-node messaging system in Nemesis can be used for intra-node message passing, while GLEX2 system is used for inter-node message transferring.

There are mainly two communication protocols in MPICH: eager and rendezvous. MilkyWay-2 MPI implements several RDMA data transfer channels which have different resource requirements and performance characteristics. Hybrid channel eager protocol will be used in runtime according to the message passing mode in application, to implement the balance of performance and scalability. Nemesis features a long message transfer (LMT) interface that facilitates implementation of zero-copy rendezvous protocol. In MilkyWay-2 MPI, LMT is implemented using zero-copy RDMA operation of GLEX2, to transfer bulk message data between application's sender buffer and receiver buffer directly. User space registration cache is also used to reduce the overhead of memory registration for message buffers.

Because RDMA data transfer may be out of order, MilkyWay-2 MPI implementation contains the message reorder mechanism and dynamic credit flow control for reliable and orderly message passing, to support the scalable running of large scale applications.

With the support of NIC collective offloading mechanisms, MilkyWay-2 MPI also implemented the offloaded MPI_Barrier and MPI_Bcast interfaces, which provide better latency than the collective interface using point-to-point operations. The offloaded optimization of other collective interfaces is also being investigated.

3.3 FT1500 CPU

FT1500 is a 40 nm 16-core processor designed for scientific computing. Each core of the FT1500 processor supports interleaved execution of eight threads and 256-bit wide SIMD processing. The SIMD processing unit supports fused multiply and add instructions. FT1500 works at 1.8 GHz and delivers a peak performance of 115.2 GFLOPS. Each FT1500 core owns a private 16 KB L1 instruction cache, a 16 KB L1 data cache and a 512 KB L2 cache.

The 16 cores are organized as four basic cells and each cell contains four cores, as shown in Fig. 3. A 4-MB L3 cache is distributed among the four cells. Inside each cell, four cores and a 1-MB L3 bank are connected through a crossbar. Two bi-directional ports are also supplied for the interconnection between adjacent cells, resulting in a 7×7 crossbar. Each port of the crossbar is 256 bit width, delivering a bandwidth of 448 GB/s. The four cells are connected through a line, forming a 1 × 4 mesh network. Meanwhile, the inter-chip interface is connected to the left port of the cell 0 and the I/O interface is

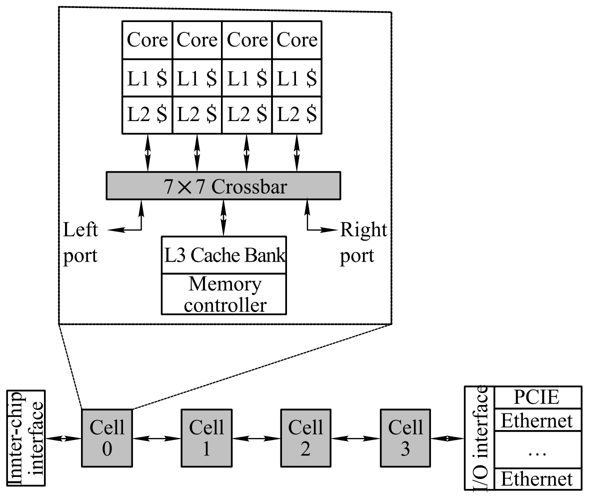

Fig. 3 Architecture of FT1500 processor

connected to the right port of the cell 3. The cache system works at 2.0 GHz while the core works at 1.8 GHz. The core clock is designed to be lower than the cache system for mainly two reasons. First, it makes a better balance between computation performance and memory bandwidth. Second, the dynamic power consumption of the cache system is less sensitive to the clock rate. Thus, such design establishes a good trade-off between performance and power consumption.

The L3 cache, being a 4 MB exclusive cache, is divided into four banks, each one working independently. The data is statically distributed among the four banks. Each bank is organized as a 32 way set-associative cache, and connected directly to a memory control unit (MCU). Each MCU manages a dedicated DDR3 memory channel. A directory based protocol is employed to maintain the coherence of different L2 caches. There are no coherence problems between different banks of the L3 cache since only one copy exists in the L3 cache for any data block.

FT1500 is a system-on-chip processor, which supplies direct links for inter-chip interconnection, four memory controllers to interface four fully buffered DIMM3 channels, two PCIE controllers and a 10 GB Ethernet ports. The SoC design reduces the total number of system components, resulting in improvements of power and reliability.

3.4 Hybrid hierarchy I/O system

We build a hybrid hierarchy file system for MilkyWay-2 system, named H2FS, which co-operates node-local storage and shared storage together into a dynamic single namespace to optimize I/O performance in data-intensive applications.

With the increasing data scale, typical global shared storage architecture, like lustre, is undergoing challenges for its inevitable I/O request contention, unbalanced performance and huge cost to achieve high I/O throughput. Compared with shared storage, node-local storage is decentralized and tightly coupled with compute node. It has natively a high degree of parallelism and has achieved great success in big data domain, but the proper way to utilize local storage in HPC field is still under exploration. For this reason, in H2FS we present a hybrid storage architecture and management scheme for not only hybrid of HDD and Flash, but also mixture of different storage architectures. H2FS provides locality-aware data layout and flexible management to serve diverse requirements of both data-intensive and computation-intensive applications. Figure 4 illustrates the file system architecture.

H2FS introduces data processing unit (DPU) and hybrid virtual namespace (HVN) to manage local and shared storage resources. DPU is a fundamental data processing unit, which tightly couples a compute node with its local storage. HVN provides a hybrid virtual namespace, which aggregates local disks of DPUs and global shared disks in a unified single namespace during application runtime. Each HVN is created with a unified namespace and can be opened and released at runtime. Application dataset can be distributed across DPUs

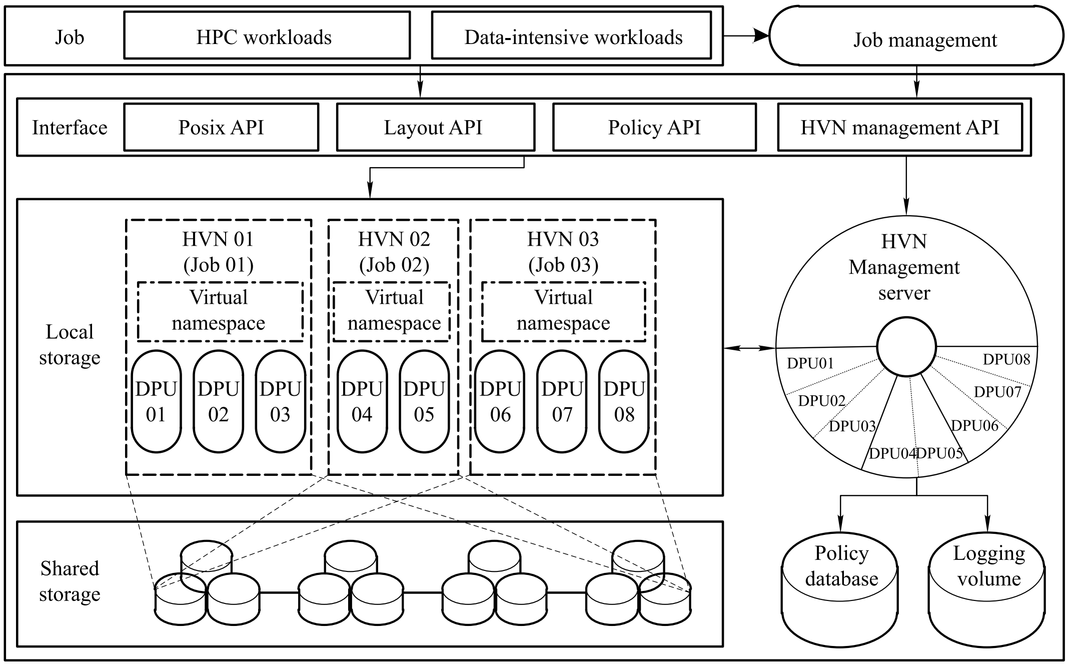

Fig. 4 Architecture of H2FS

inside an HVN at granularity of file or stripe. Locality or storage type aware policies are used to optimize dataset layout, and job scheduler can schedule task close to data for maximum locality. By employing HVNs, job working dataset can be isolated from each other for minimum interference, and exclusively accessed for appropriate concurrency. As for the interface, H2FS keeps POSIX interface for compatibility, and uses layout API, policy API and HVN API jointly for performance optimization and integration with other softwares, like HDF5, slurm and etc.

H2FS benefits both data-intensive applications and typical HPC I/O critical scenarios. Application can use local storage as performance booster in single namespace with benefits of spatial locality and ease of use. Furthermore, it can also provide hints to guide efficient cooperation between local and global storage to improve I/O performance. H2FS can easily delivery high I/O aggregate bandwidth with increasing nodes scale. More HPC applications can transparently achieve benefits of node-local storage with support of unified single namespace. It can reduce costs of building extremely fast storage system compared with conventional centralized one.

3.5 OpenMC programing model

Over the last few years, we have been looking for a suitable programming model for heterogeneous systems. In MilkyWay supercomputers, we rely on a hybrid model, in which MPI applies to inter-node programming and “OpenMP + X” to intra-node programming, where X is accelerator-specific, e.g., CUDA/OpenCL for GPUs [3] or “Offload” for Intel MIC [4]. While MPI may continue to be the primary model for inter-node programming, “OpenMP + X” is becoming less appealing as it makes intra-node programming ad hoc, fairly complex and quite difficult to obtain the degree of portability desired.

In search for a better intra-node programming model, we prefer to a directive-based solution over CUDA [3] and OpenCL [5], because the former provides a higher-level abstraction for programming accelerators and facilitates incremental parallelization [6]. Several directive-based models, including OpenACC [7], PGI Accelerator [8,9] and OpenHMPP [10], have recently been introduced. By providing directive-based interfaces, they enable programmers to offload computations to accelerator and manage parallelism and data communication.

To harness the full potential of MilkyWay-2 supercomputers, existing directive-based intra-node programming models are inadequate in three aspects. First, a single code region, when offloaded, is usually executed as a solo task in an entire accelerator, resulting in its low utilization when the task exhibits insufficient parallelism. Second, as accelerator-oriented computing is emphasized, the ever-increasing processing power of general-purpose multi-core CPUs is neglected and thus inadequately exploited, especially for some irregular applications. Finally, multiple devices, i.e., multiple multi-core CPUs and multiple many-core accelerators are nowadays found in a single node. We need a little more than just offering syntax to offload computations to accelerators. The ability to orchestrate the execution of multiple tasks efficiently across these devices becomes essential. We are not aware of any directive-based open-source compiler for MilkyWay-like heterogeneous systems accelerated by both GPUs and Intel MIC.

To overcome the aforementioned three limitations, we introduce a directive-based intra-node programming model, open many-core (OpenMC), towards simplifying programming for heterogeneous compute nodes, especially those used in the MilkyWay-2 supercomputers. OpenMC provides a unified abstraction of the hardware resources in a compute node as workers, where a worker can be a single multi-core device or a subset of its cores if this is permitted. It facilitates the exploitation of asynchronous task parallelism on the workers. As a result, OpenMC allows different types of devices to be utilized in a uniform and flexible fashion.

As shown in Fig. 5, OpenMC provides two levels of abstraction for the “software” running on the “hardware”. All the hardware resources in a compute node are available to an OpenMC program as a group of workers at the logical level. The tasks in an OpenMC program are organized at the software level in terms of a *master* thread, asynchronously executing *agents* that are dynamically spawned by the master (to run in slave threads) and *accs* (or accelerator regions) offloaded to workers from an agent. All the tasks between two consecutive global synchronization points will run in parallel unless their dependencies are explicitly annotated by programmers.

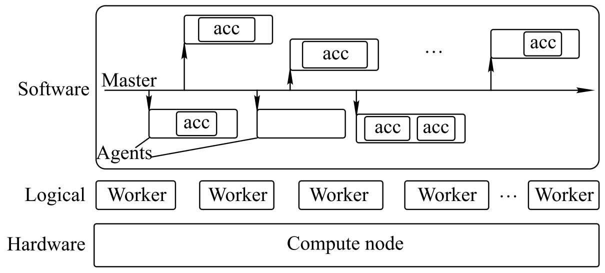

Fig. 5 Execution model of OpenMC

Essentially, OpenMC memory model is similar to those adopted by existing ones such as OpenACC [8] and OpenHMPP [10]. In an OpenMC program, the serial code in an agent runs on the host, where the master thread runs. As a result, their program states are mutually visible. However, programmers are able to create agent-private copies for certain variables, e.g., loop variables, for a particular agent.

The cores in a worker share the same memory space. If two workers have different memory spaces, then data movement between the two spaces is either annotated by programmers or deduced by compiler. An agent is responsible for managing data communication and synchronization required by its accs.

3.6 Intelligent system administration

As shown in Fig. 6, the intelligent management system is monitoring the whole system of MilkyWay-2, including the hardware and the software.

Besides the traditional monitoring functions, the system implement the error monitoring and the intelligent fault management. The system provides real-time error probing, precise fault diagnosis and in-time maintenance hint. The error probing is separated into the hardware part and the software part. The hardware error is collected by the customized hardware error monitoring facilities. The software error is digested by the OS kernel and the driver hooks. All errors are analyzed by a scalable fault management framework, which supports the evolution of the diagnosis ability and provides precise predictions of fault infection areas and infecting job list. Meanwhile, the system simplifies the maintenance process, thus expanding the system usability.

The key of fault management is managing the life cycle of all faults automatically and minimizing the cost of each stage, including fault detection, fault diagnosis, fault isolation, and task recovery. That is, the system makes the supercomputer to be self-managed, which is also defined as autonomous management. The system implements the following five autonomous features for fault management:

• Self-awareness: the supercomputer can be aware of the execution status and detect faults automatically.

• Self-diagnosis: the supercomputer can analyze the errors and locate the root cause of the errors automatically.

• Self-healing: the supercomputer can isolate the faulty hardware, reconfigure itself, and preserve high execution efficiency automatically.

• Self-protection: the supercomputer can protect the tasks running on it from the impact of faults automatically.

• Self-evolution: the supercomputer managing capability can be improved with more and more experiences gathered during system managing.

By autonomous management, our system greatly reduces the overhead of fault management. Therefore, it increases the usability and reliability of computer systems. In MilkyWay-2 supercomputer, our system speeds up the efficiency of fault management by two orders of magnitude.

In addition, our system is a general autonomous management frame work that can be extended for a broad range of computers beside supercomputers. We propose a self-similar system architecture for our system, by which the system is not sensitive the scale of target computer system. Moreover, the functional model of our system is an open interface that allows any developer to insert new function models such that the system supports dynamical and seamless extension. The architecture related and system specialized functions are

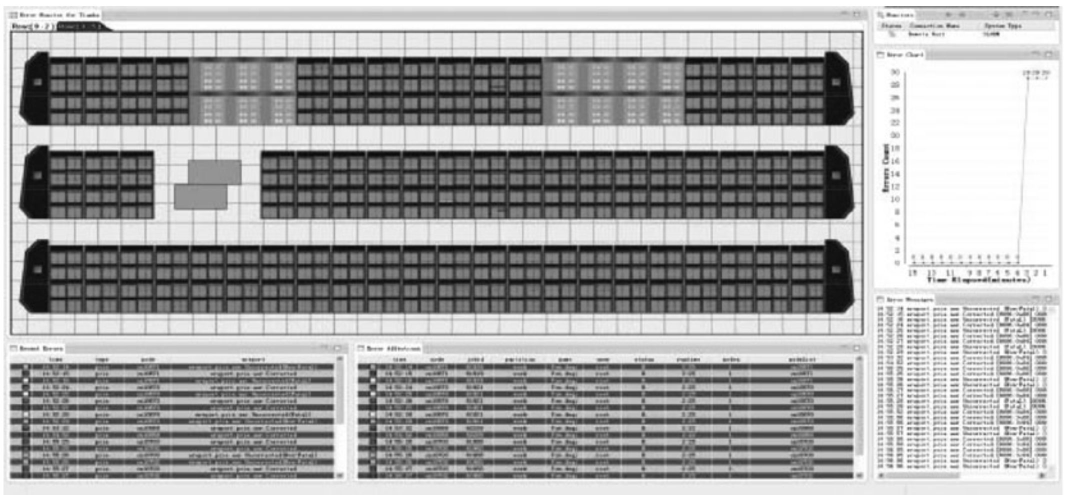

Fig. 6 The GUI of intelligent management system

confined into several models. Hence, the work for the transplantation, extension, or upgrade of our system is minimized. Besides this feature, the system can also support evaluative function. The system becomes smarter based on ever increasing knowledge gathered from normal days.

4 Benchmark evaluation

4.1 LINPACK benchmark

LINPACK is a widely recognized industry benchmark for system-level performance of high-performance computing systems. It solves a dense $N \cdot N$ system of linear equations of the form $Ax = b$ using parallel blocked LU decomposition method. With a problem size of $N = 9\ 960\ 000$ on the 124 racks of MilkyWay-2 system, a maximal performance of 33.86 PFLOPS is achieved, which corresponds to 62.16% of the theoretical peak performance.

The LINPACK implementation running on MilkyWay-2 is developed by Intel and delivered as massively parallel (MP) LINPACK benchmark, which is based on modifications and additions to high-performance LINPACK (HPL) 2.1 from innovative computing laboratories (ICL) at the University of Tennessee, Knoxville. Based on MP LINPACK, we do lots of performance tuning and optimization work to harness the massive compute power of the whole system.

Table 1 shows the parameters used in MilkyWay-2 LIPACK benchmark evaluation in details. We run 31 744 MPI processes as a two-dimention grid 248×128 on 15 872 compute nodes. There are two processes on each node to leverage the computer capacity of 2 Xeon CPUs and 3 Xeon Phi cards. Each process uses one CPU as the host and 1.5 Xeon Phi cards as the co-processors. The host is responsible to the initialization, controlling, communication and part of computing jobs. The co-processors deal with the majority of the computing jobs including DGEMM and DTRSM level 3 BLAS functions using the off-load mode.

One of the innovative optimization methods used in MilkyWay-2 is the "panel broadcast algorithm dynamic switching". There are six broadcast algorithms in the HPL 2.1, which are known as *1ring, 1ringM, 2ring, 2ringM, long and longM*. The choice of the algorithm is related to the network performance, the computer performance, the topology of the network, etc. We observed that in the MilkyWay-2 system the longM algorithm achieved higher performance at the beginning loops, however dropped fast during the end phase. The 1ringM algorithm showed the opposite trends, which began with lower performance and ended with flatter fluctuation. So we made a switching point during the running of LINPACK. Before that point longM algorithm is used, and after that the broadcast algorithm is switched to 1ringM. The experiment results show that at least 1.5% is achieved by this optimization method.

The output of the LINPACK benchmark is shown in Fig. 7. The benchmark was completed in 19 452 seconds (about five hours and 24 minutes), resulting in 33.86 PFLOPS and passed the residual check.

4.2 Graph500

Graph500 was introduced in 2010 as a complement to the TOP500 list. It aims to rank computers according to their capability of processing data-intensive tasks. BFS has been chosen as the primary benchmark for Graph500.

The performance of Graph500 benchmark is mainly affected by memory access, communication and computation. Therefore, we improve it from these aspects respectively. We presented several software strategies to improve performance of Graph500 benchmark on MilkyWay-2 from cost of memory access, communication and computation.

We evaluate the scalability of our algorithm on MilkyWay-2 by showing weak scaling, with 227 vertices per node, shown in Fig. 8(a). With this form of weak scaling, the memory usage per node increases as the scale becomes larger. It achieves good weak scaling as the size increases. The key performance metric is the traversed edges per second (TEPS). TEPS rate is proportional to nodes over the whole range of sizes. Our algorithm achieves 206 148 billion TEPS on a sub-system of MilkyWay-2 supercomputer with 8 192 nodes (196 608 cores), which is about 6.8 X faster than top-down

Table 1 HPL parameters used in MilkyWay-2 LINPACK benchmark evaluation

Matrix size (N)	9 960 000	Panel broadcast	longM & 1ringM
Block size (NB)	1 008	Look-ahead depth	1
Process mapping	Column-major	Swap	Binary-exchange
Process grid (P×Q)	248×128	Matrix form	L1:no-trans,U:no-trans
Panel factorization	Crout	Equilibration	No
Recursive panel factorization	Crout	Threshold	16.0
NBMIN, NDIV	4, 2	Alignment	eight double precision words

```
================================================================================
T/U                 N          NB      P       Q                 Time              Gflops
--------------------------------------------------------------------------------
WC15C2C4        9 960 000   1 008     248     128             19 452.04        3.386 27e+07
HPL_pdgesv( )   start    time   Mon   Jun   3   00:25:03    2013

HPL_pdgesv( )   end   time      Mon   Jun   3   05:49:16    2013

--------------------------------------------------------------------------------
||Ax-b||_00/(eps*(||A||_00*||x||_00+||b||_00)*N)=           0.001 966 1     ...    PASSED
================================================================================
```

Fig. 7 Output of LINPACK benchmark

algorithm [11] and is about 11.0 X faster than the 2D simple hybrid algorithm [12].

Fig. 8(b) shows the results from a strong scaling test. In contrast to weak scaling, the global problem size, i.e., the number of vertices in a given graph, remains constant while the number of processors increases. Therefore, the number of local vertices assigned to each processor decreases as the number of processors increases. We see that our algorithm is about 6.7 X~7.6 X faster than the 2D top-down algorithm [11], and is about 4.9 X~11.3 X faster than 2D simple hybrid algorithm [12]. Hence, our optimization improves performance and scalability of hybrid BFS algorithm greatly.

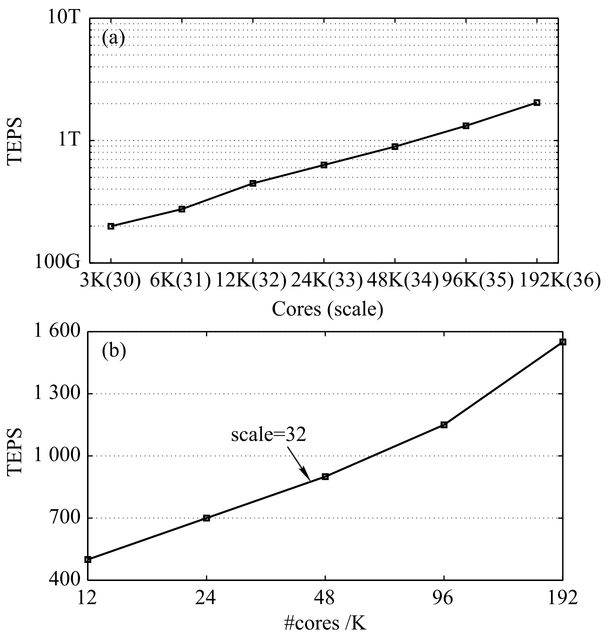

Fig. 8 Graph500 benchmark on MilkyWay-2. (a) Parallel weak scaling; (b) Parallel strong scaling

5 Applications

5.1 Gyrokinetic toroidal gode

The gyrokinetic toroidal code (GTC) is a global, three-dimensional particle-in-cell application developed to study microturbulence in tokamak fusion devices. GTC is optimized to use offload programming model for the Intel MIC architecture, and is massively tested on MilkyWay-2. Table 2 and Fig. 9 show parts of the test results, where we vary the number of used compute nodes, and run GTC on one MIC card, two MIC cards, three MIC cards, or two CPUs respectively. From Fig. 9, we can see GTC achieves linear scalability for both CPU and MIC devices. Figure 10 shows the performance comparison between MIC and CPU. The performance ratio of a MIC card to 2 CPUs is between 0.61 and 0.65. The performance ratio of two MIC cards to two CPUs is between 1.15 and 1.18. The performance ratio of three MIC cards to 2 CPUs is between 1.62 and 1.67.

Table 2 Billion electrons pushed per second

Device	Nodes					
	64	128	256	512	1 024	2 048
2CPUs/node	2.80	5.41	10.79	20.87	40.83	67.56
1MIC/node	1.72	3.42	6.79	13.49	24.92	41.87
2MICs/node	3.22	6.40	12.57	24.49	47.64	–
3MICs/node	4.54	9.01	17.74	–	–	–

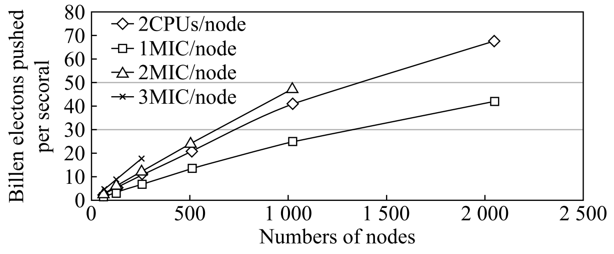

Fig. 9 Weak scaling performance of GTC

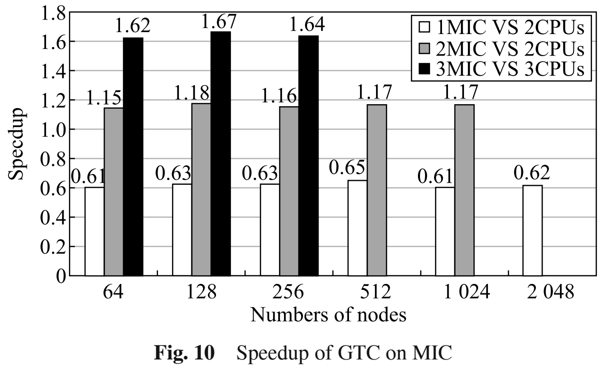

Fig. 10 Speedup of GTC on MIC

5.2 Parallel molecular docking

Virtual high throughput screening (vHTS) is one established computational method to identify drug candidates from large collection of compound libraries, and thus accelerate the drug discovery process [13–15]. Molecular docking puts small compounds into the active site of drug target to evaluate the binding affinity, and is a fairly valuable tool of vHTS due to its good balance between speed and accuracy. We adapt a modified version of DOCK6.5 [16] to MilkyWay-2. 303 826 small compounds conformations from SPECS[1)] are docked to 1 100 potential drug targets from potential drug target database (PDTD) [17] in parallel. Over 334 million docking calculations are finished using ten thousands of CPU cores in MilkyWay-2 within one week. The post docking analysis combined with experimental assays would yield many hit compounds against cancer, diabetes, infectious diseases and etc.

5.3 Global atmosphere simulation

Large-scale simulation of the global atmosphere is one of the most computationally challenging problems in scientific computing. There is an urgent need for a highly scalable framework that fits well into state-of-the-art heterogeneous systems equipped with both processors and accelerators. Among a variety of equation sets that are used to model the global atmosphere, shallow-water equations exhibit most of the essential dynamical characteristics of the atmosphere, thus can be used as a test bed for the development of new algorithms. On the Tianhe-1A, a peta-scalable CPU-GPU algorithm was proposed to enable high-fidelity atmospheric simulation at extreme scales [18]. The hybrid algorithm has been recently extended to MilkyWay-2. In the hybrid algorithm, an adjustable partitioning method for arbitrary CPU-MIC ratio is proposed to utilize equally and efficiently both CPU and MIC. A novel "pipe-flow" communication scheme for the cubed-sphere geometry is introduced to conduct balanced and conflict-free message passing. Systematic optimizations of both the CPU and MIC codes are done to achieve high double-precision performance. In the single node test on the MilkyWay-2, the MIC-to-CPU speedup for the $4\,096 \times 4\,096$ mesh is 1.18, 2.26 and 3.15 when one, two and three MICs are utilized. Large scale tests indicate that the average performance gained from SIMD vectorization on the MIC accelerator is about 4.5 X. Thanks to the successful communication-computation overlap, a nearly ideal weak-scaling efficiency of 93.5% is obtained when we gradually increase the number of nodes from six to 8 664 (nearly 1.7 million cores). In the strong-scaling test, the parallel efficiency is about 77% when the number of nodes increases from 1 536 to 8 664 for a fixed $65\,664 \times 65\,664 \times 6$ mesh (the total number of unknowns is 77.6 billion).

1) http://www.specs.net

6 Conclusions

The design of MilkyWay-2 system is driven by ambitious aims: high performance, energy efficiency, high availability, and user friendliness. MilkyWay-2 employs a proprietary high-speed communication network to connect neo-heterogeneous nodes that integrate CPUs and MIC accelerators, and yields impressive improvements in performance, efficiency and usability. MilkyWay-2 system has been installed at NSCC-GZ, and will provide services for a wide range of potential fields including basic science, major engineering, industrial upgrading and information infrastructure construction. For basic science, MilkyWay-2 can be applied to many fundamental researches, such as universe science, earth science, life science, nuclear science and etc. For major engineering, MilkyWay-2 will be an important supporting platform for solving challenge issues in national major engineering or projects, such as large aircrafts, oil exploration, nuclear power station, genetic engineering and large-scale equipment manufacturing. For industrial upgrading, the system will drive the technological advancements in industries directly related to HPC like microelectronics, optical communication and software development, and potentially promote the innovation and reconstruction of industries including automobile, shipbuilding, machinery manufacturing, and electronics. For the construction of information infrastructure, MilkyWay-2, combined with cloud computing, can provide the information infrastructure to support smart cities, e-government affairs, and Internet-of-things applications. We believe the full potential of MilkyWay-2 can be exploited to produce a major impact on Chinese HPC industry and contribute to the construction of an innovative country.

Acknowledgements This work was partially supported by the National High-tech R&D Program of China (863 Program) (2012AA01A301), and the National Natural Science Foundation of China (Grant No. 61120106005). The MilkyWay-2 project is a great team effort and benefits from the cooperation of many individuals at NUDT. We thank all the people who have contributed to the system in a variety of ways.

References

1. Yang X J, Liao X K, Lu K, Hu Q F, Song J Q, Su J S. The Tianhe-1a su-

percomputer: its hardware and software. Journal of Computer Science and Technology, 2011, 26(3): 344–351
2. Zhang H, Wang K, Zhang J, Wu N, Dai Y. A fast and fair shared buffer for high-radix router. Journal of Circuits, Systems, and Computers, 2013
3. Kirk D. Nvidia cuda software and GPU parallel computing architecture. In: Proceedings of the 6th International Symposium on Memory Management. 2007, 103–104
4. Sherlekar S. Tutorial: Intel many integrated core (MIC) architecture. In: Proceedings of the 18th IEEE International Conference on Parallel and Distributed Systems. 2012, 947
5. Gaster B, Howes L, Kaeli D R, Mistry P, Schaa D. Heterogeneous Computing with OpenCL. Morgan Kaufmann Publishers Inc., 2011
6. Lee S, Vetter J S. Early evaluation of directive-based GPU programming models for productive exascale computing. In: Proceedings of the 2012 International Conference for High Performance Computing, Networking, Storage and Analysis. 2012, 1–11
7. Wienke S, Springer P, Terboven C, Mey D. Openacc: first experiences with real-world applications. In: Proceedings of the 18th International Conference on Parallel Processing. 2012, 859–870
8. PGI Accelerator Compilers. Portland Group Inc, 2011
9. Yang X L, Tang T, Wang G B, Jia J, Xu X H. MPtoStream: an openMP compiler for CPU-GPU heterogeneous parallel systems. Science China Information Sciences, 2012, 55(9): 1961–1971
10. Dolbeau R, Bihan S, Bodin F. Hmpp: a hybrid multi-core parallel programming environment. In: Proceedings of the 2007 Workshop on General Purpose Processing on Graphics Processing Units. 2007, 1–5
11. Checconi F, Petrini F, Willcock J, Lumsdaine A, Choudhury A R, Sabharwal Y. Breaking the speed and scalability barriers for graph exploration on distributed-memory machines. In: Proceedings of the 2012 International Conference for High Performance Computing, Networking, Storage and Analysis. 2012, 1–12
12. Beamer S, Buluç A, Asanovic K, Patterson D. Distributed memory breadth-first search revisited: enabling bottom-up search. In: Proceedings of the 27th IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum. 2013, 1618–1627
13. Subramaniam S, Mehrotra M, Gupta D. Virtual high throughput screening (VHIS)–a perspective. Bioinformation, 2007, 3(1): 14–17
14. Tanrikulu Y, Krüger B, Proschak E. The holistic integration of virtual screening in drug discovery. Drug Discovery Today, 2013, 18(7): 358–364
15. Zhang X, Wong S E, Lightstone F C. Message passing interface and multithreading hybrid for parallel molecular docking of large databases on petascale high performance computing machines. Journal of Computational Chemistry, 2013, 34(11): 915–927
16. Lang P T, Brozell S R, Mukherjee S, Pettersen E F, Meng E C, Thomas V, Rizzo R C, Case D A, James T L, Kuntz I D. Dock 6: combining techniques to model RNA–small molecule complexes. RNA, 2009, 15(6): 1219–1230
17. Gao Z, Li H, Zhang H, Liu X, Kang L, Luo X, Zhu W, Chen K, Wang X, Jiang H. PDTD: a web-accessible protein database for drug target identification. BMC Bioinformatics, 2008, 9(1): 104
18. Yang C, Xue W, Fu H, Gan L, Li L, Xu Y, Lu Y, Sun J, Yang G, Zheng W. A peta-scalable CPU-GPU algorithm for global atmospheric simulations. In: Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2013, 1–12

Xiangke Liao received the BS in computer science from Tsinghua University, China and MS degree in computer science from the National University of Defense Technology (NUDT), China. Currently he is a professor at NUDT. His research interests include high performance computing system, operating system, parallel software. He is the chief designer of MilkyWay-2 system.

Liquan Xiao received his MS and PhD in computer science from National University of Defense Technology (NUDT), China. Currently he is a professor at the university. His research interests include architecture of high performance computing, high speed interconnect network, system integration, and power management. He is a deputy chief designer of MilkyWay-2 supercomputer.

Canqun Yang received his MS and PhD in computer science from National University of Defense Technology (NUDT), China in 1995 and 2008, respectively. Currently he is a professor at the university. His research interests include programming languages and compiler implementation. He is a director designer of the MilkyWay-2 supercomputer.

Yutong Lu received her MS and PhD in computer science from National University of Defense Technology (NUDT), China. Currently she is a professor at the university. Her research interests include parallel system management, high speed communication, distributed file systems, and advanced programming environments with MPI. She is a director designer of MilkyWay-2 supercomputer.