Brief Introduction of TianHe Exascale Prototype System

Ruibo Wang  
*College of Computer, National University of Defense Technology, Changsha 410072, China*

Kai Lu  
*College of Computer, National University of Defense Technology, Changsha 410072, China*

Juan Chen  
*College of Computer, National University of Defense Technology, Changsha 410072, China*

Wenzhe Zhang  
*College of Computer, National University of Defense Technology, Changsha 410072, China*

Jinwen Li  
*College of Computer, National University of Defense Technology, Changsha 410072, China*

*See next page for additional authors*

---

Follow this and additional works at: [https://tsinghuauniversitypress.researchcommons.org/tsinghua-science-and-technology](https://tsinghuauniversitypress.researchcommons.org/tsinghua-science-and-technology)

Part of the *Computer Sciences Commons, and the Electrical and Computer Engineering Commons*

---

**Recommended Citation**


This Research Article is brought to you for free and open access by Tsinghua University Press: Journals Publishing. It has been accepted for inclusion in Tsinghua Science and Technology by an authorized editor of Tsinghua University Press: Journals Publishing.
Brief Introduction of TianHe Exascale Prototype System

Authors
Ruibo Wang, Kai Lu, Juan Chen, Wenzhe Zhang, Jinwen Li, Yuan Yuan, Pingjing Lu, Libo Huang, Shengguo Li, and Xiaokang Fan
Brief Introduction of TianHe Exascale Prototype System

Ruibo Wang, Kai Lu*, Juan Chen, Wenzhe Zhang, Jinwen Li, Yuan Yuan, Pingjing Lu, Libo Huang, Shengguo Li, and Xiaokang Fan

Abstract: Facing the challenges of the next generation exascale computing, National University of Defense Technology has developed a prototype system to explore opportunities, solutions, and limits toward the next generation Tianhe system. This paper briefly introduces the prototype system, which is deployed at the National Supercomputer Center in Tianjin and has a theoretical peak performance of 3.15 Pflops. A total of 512 compute nodes are found where each node has three proprietary CPUs called Matrix-2000+. The system memory is 98.3 TB, and the storage is 1.4 PB in total.

Key words: TianHe exascale system; prototype; proprietary CPU; Matrix-2000+

1 Background

The development of high-performance computing has always been a strategic goal for many countries. The current state-of-the-art system is at the 100 P level, and mainstream research institutions and manufacturers are aiming to achieve exascale level in 2020.

However, exascale computing poses many challenges, and simply expanding the current system is unrealistic[1]. For example, TianHe-2A[2] has nearly 20,000 nodes and a peak performance of 100 P. If expansion is necessary to reach the exascale level (1000 P), the system needs more than 200,000 nodes. Such a system is neither programmable nor stable, with the potential Mean Time Between Failures (MTBF) being less than hours, and the energy consumption being unacceptable. All these problems cannot be solved with existing technologies.

Facing the challenges of the next generation exascale computing, National University of Defense Technology (NUDT) has developed a prototype system to explore opportunities, solutions, and limits toward the next generation Tianhe system.

The research and development of the system were launched in early 2016 and completed in late 2018. It is deployed at the National Supercomputer Center in Tianjin and has a theoretical peak performance of 3.15 Pflops. A total of 512 compute nodes exist, and each node has three proprietary CPUs called Matrix-2000+. The system memory is 98.3 TB, and the storage is 1.4 PB in total.

The rest of this paper is organized as follows: Section 2 introduces our proprietary CPU called Matrix-2000+. Section 3 presents the compute node structure. Section 4 introduces proprietary interconnection. Section 5 describes the monitoring and diagnostic subsystem. Section 6 discusses the programming environment, and Section 7 shows several typical high-performance applications.

2 Proprietary CPU: Matrix-2000+

Matrix-2000+, as a verification chip shown in Fig. 1,
adopts a regional autonomous parallel architecture. The chip is composed of several regions, and each region as a functionally independent processor. The intra-region is composed of multiple core groups through intra-region interconnection interfaces. This architecture aims to make full use of the locality of programs and features high performance, high scalability, and flexible physical implementation[3].

Matrix-2000+ has 128 compute cores running at 2 GHz and achieving 2.048 TFlops peak performance. Figure 2 illustrates its conceptual structure. The CPU is configured with four Super-Nodes (SNs), connected through a scalable on-chip communication network. Each SN has four panels and each panel containing eight cache-coherent compute cores. The CPU supports eight DDR4-2400 channels and is integrated with PCIe 3.0 interface. The compute core is an in-order 8 to 12-stage Reduced Instruction Set Computing (RISC) pipeline extended with vector Instruction Set Architecture (ISA), resulting in eight Double Precision (DP) flops/cycles. The peak performance of Matrix-2000+ can then be calculated as 4 SNs × 32 cores × 8 flops × 2 GHz = 2.048 Tflops.

Data exchange among cores is accomplished by the data exchange engine in a panel. This module is responsible for protocol bridging, arbitration between local and remote accesses, and private directory controlling. Different panels communicate with each other through a hierarchical network-on-chip[4].

3 Compute Node

Each compute node (Figs. 3 and 4) is equipped with three Matrix-2000+ CPUs with an operating frequency of 2 GHz and a peak performance of 6 Tflops. Each CPU is equipped with eight-channel DDR4 controllers. With eight DDR4 DIMM slots designed in a CPU, it comes with 24 DDR4 DIMMs per compute node with maximum memory access bandwidth of 614.4 Gbps and capacity of 768 GB. The proprietary interconnect network interface chips are embedded to achieve high-speed network interconnection among CPUs and nodes. A single network port uses a four-lane high-speed serial transmission link. The link rate is up to 25 Gbps. The one-port one-way bandwidth is 100 Gbps and the external one-way bandwidth of a single compute node is 400 Gbps in total. The CPU and the network interface chip are connected through the PCIe Gen3 16x interface. In addition, multi-voltage power supply and energy consumption dynamic management are adopted to achieve high energy efficiency.

4 Proprietary Interconnect

The system adopts a proprietary interconnect network. The network logic is developed and integrated into

![Fig. 2 Conceptual structure of Matrix-2000+.](image-url)
two specific ASIC chips, the network interface chip (named HFI-E) and the network router chip (named HFR-E), as displayed in Figs. 5 and 6, respectively. Both chips implement efficient mechanisms to achieve high-performance communication with regard to bandwidth, latency, reliability, and stability. The link rate is upgraded to 25 Gbps from 14 Gbps of the TianHe-2A supercomputer system. Collective offload\cite{ref5} accelerates collective operations, effectively improves the throughput of a single chip.

HFI-E provides the software-hardware interface for accessing the high-performance network, implementing the proprietary Mini Packet/Remote Direct Memory Access (MP/RDMA) communication and collective offload mechanism. HFI-E contains a 16-lane PCIe 3.0 interface\cite{ref6} and connects with interconnect fabric via network ports. Each network port has an eight-lane 25 Gbps SerDes interface. HFI-E adopts FC-PBGA\cite{ref7} packaging technology and supplies 692 pins. The die size of HFI-E is 10 mm × 10 mm.

HFR-E contains 24 network ports. Each port has an eight-lane 25 Gbps SerDes interface, with 200 Gbps unidirectional bandwidth. The throughput of a single HFR-E chip is up to 9.6 Tbps, and the Message Passing Interface (MPI) latency is 1.1 us. HFR-E also adopts
Flip Chip-Plastic Ball Grid Array (FC-PBGA) packaging technology and supplies 2816 pins. The die size of HFR-E is 19 mm × 19 mm.

The interconnection system network adopts a two-dimensional tree network topology on the basis of opto-electronic hybrid interconnection, as shown in Fig. 7. A total of 72 compute frames are connected by four communication frames using active optical cables with a two-dimensional tree network topology. Such a topology is more improved than n-D-Torus topology. The links between the adjacent nodes on each dimension are replaced with tree switches that enable the scaling of the two-dimensional tree network topology to interconnect more than 150,000 nodes, thus meeting the interconnection needs of exascale supercomputers. The system implements the world’s highest density switches and optical modules, and 864 optical ports are integrated into a single chassis.

The system implements intelligent network management. By monitoring the network resource status of the system, flexible allocation of network resources and traffic autonomous management functions are supported for the communication domain. An underlying hardware management firmware is implemented, and an Application Programming Interface (API) interface for system resource management scheduling is provided. Based on this interface, computing and network resource collaborative management can be realized. High-speed network fault location, self-healing, and self-isolation through software-defined intelligent network management control technology are realized, greatly improving the reliability of the interconnection system.

5 Monitoring and Diagnostic Subsystem

To support the high reliability, availability, and serviceability of exascale supercomputer systems, we propose a hierarchical regional autonomous architecture for the monitoring and diagnostic subsystem. Figure 8 illustrates that the subsystem is composed of three hierarchical management units, including Board Management Unit (BMU), Chassis Management Unit (CMU), and System Management Unit (SMU). These units are uniformly interconnected through a dedicated monitoring and diagnostic network, which enables the subsystem to work and be maintained independently. On the basis of these units, we propose a three-level management framework to achieve elaborate, autonomous, and intelligent monitoring and diagnostic for exascale systems.

Elaborate BMU management. Each BMU is integrated into a functional board as a module.

Fig. 7 Interconnection network topology.
BMUs adopt a software-defined approach to meet the monitoring demands of the functional boards in computing, interconnection, and storage subsystems. Each BMU monitors the temperature, voltage, and current of devices and supports functions, such as power control, out-of-band firmware update, networking management, and in-band data aggregation, which satisfies the demand of board-level elaborate management.

**Autonomous CMU management.** Each chassis is equipped with two CMUs for fault tolerance. All monitoring data in a chassis are aggregated from the BMUs to the two CMUs. According to tasks, we divide all the chassis into several autonomous regions, which are managed by differentiated policies determined by the head CMU. This regionally autonomous design significantly reduces system-level monitoring complexity and improves system flexibility.

**Intelligent SMU management.** We configure multiple SMUs in the subsystem with an active/inactive mode. The SMUs are connected with the monitoring and diagnostic network and the high-speed interconnection network. SMUs collect the massive monitoring data from all the nodes and the power and cooling subsystems, and then use machine learning methods to provide system-level intelligent fault prediction and diagnosis and energy prediction and management. These factors comprehensively improve system maintenance efficiency and reduce energy consumption.

Currently, our monitoring and diagnostic subsystem satisfies the prototype system and is considered a potential solution for our next generation exascale supercomputer system.

### 6 Programming Environment

#### 6.1 Parallel development environment


#### 6.2 OpenMP multi-thread optimization for many-core architecture

To meet the needs of parallel programs in different fields and architectures, the OpenMP 4.5 standard has added new features, such as new Single Instruction Multiple Data (SIMD) extensions, depend clauses, and task-group directives.

We implement new task clauses and directives, such as depend and task-group, to support task migration. For irregular programs, such as recursive function calls and graph searches, we use Near-Data Computing (NDC) to improve data locality. With rich environment variable support, we can also achieve thread affinity scheduling at different levels of granularities.

Figure 9 illustrates the specific optimizations conducted to improve the performance of OpenMP programs for the Matrix–2000+ many-core architecture.

1. **Locality optimization for Non-Uniform Memory Access (NUMA) architecture**

   The compiler can locate the physical position of data by using the first-touch mechanism. When performing
loop partitioning, we first analyze the data distribution of the processed loop, and then perform the task division according to the NDC principle.

(2) Optimization for the group-shared cache
General many-core processors use a group-shared L2 cache. The compiler analyzes the data access patterns and ensures that threads accessing data with spatial locality are scheduled to the same set of cores as much as possible. At the same time, by using the thread-sharing feature, the data required by the remaining threads scheduled to the same core group can be loaded into the cache in advance through a thread pre-fetch operation.

(3) Synchronization optimization
The synchronization includes barrier, critical section, and implicit synchronizations of reduction operations. The performance of barrier synchronization is limited by the efficiency of atom accesses to synchronous variables by different threads. To improve the performance of atom accesses, the compiler ensures that threads are in a bound state when the barrier is synchronized.

The OpenMP 4.0 standard supports array and user-defined reduction operations. However, traditional lock-based reduction operations have limited performance when the number of threads is large. We use a $k$-ray-based reduction operation to improve the performance.

6.3 Efficient implementation of an OpenCL parallel compiler
We implement MOCL, which is an OpenCL programming model for the Matrix-2000+ many-core CPU. MOCL comprises a kernel compiler and a runtime system. The kernel compiler is responsible for compiling the kernel code. The runtime system implements the OpenCL API and manages the runtime context.

To improve the performances of OpenCL programs on the Matrix-2000+ many-core CPU, MOCL performs the following two optimizations for the kernel compiler and the runtime system.

1) Kernel compiler with “lock-free” atom accesses
The MOCL kernel compiler is built on LLVM-5.0 and conforms to the OpenCL v1.2 specification. Unlike GPUs, Matrix-2000+ supports Portable Operating System Interface of UNIX (POSIX) threads. Therefore, we customize the OpenCL kernel compiler for Matrix-2000+. The basic idea is to schedule different workgroups to run on different hardware threads while keeping work items in the same workgroup running on the same hardware thread. Work items within the same workgroup are executed in a determined order, indicating that no concurrent access to work items exists when the atomic variable is in local storage. We propose a “lock-free” implementation of atomic operations. Our implementation can avoid the synchronization overhead between different work items in the same workgroup and significantly improve the performance of OpenCL kernel programs.

2) Runtime system with a “push-first” task dispatch strategy
The runtime system implements the API of OpenCL. The host and the device communicate in a command manner. The commands include buffer allocating and releasing commands, kernel function compiling and executing commands, data moving commands, and synchronizing commands.

An API works as follows: The host initially issues a command to the command queue of the device. Subsequently, the device reads the command from the queue and performs the assigned work. When the runtime system on the device side receives a task, the manner of assigning the task to an idle thread on the device is the key of the runtime system. MOCL uses a “push-first” task allocation strategy. The runtime system first determines the required number of worker threads and the number of tasks each worker thread needs to process according to the specific task size.

A corresponding number of idle threads is taken from the thread pool, and task information is set for these idle threads, which wake up to perform tasks. Compared with the traditional “pull-first” task allocation strategy, our strategy can reduce the synchronization overhead between worker threads and thus is more suitable for OpenCL programs.

7 Application
The system has provided high-performance computing
services to more than 10 users in China, and more than 40 applications have been experimented on that system, including framework programs, nuclear physics, computational fluid dynamics, large-scale equipment electromagnetic simulations, biological information, engine combustions, nonlinear flows, mechanical strengthen analyses, and other fields[17–21]. We achieve a series of good scientific research results. Specifically, we introduce two Computational Fluid Dynamics (CFD) applications and one electromagnetic simulation in detail.

Example 1. The sub-surface supersonic CFD software platform (named TRIP) is a large-scale structural grid numerical simulation software developed by the China Aerodynamics Research and Development Center. This software includes all the mainstream functions of structural grid computing software and has recently added aeroelastic and aerodynamic noise modules. On the NUDT system, the CFD/Comparehensive Structural Dynamics (CSD) tight coupling strategy is used to complete the numerical simulation of a wing transonic flutter (Fig. 10). The maximum parallel scale can reach 197 000 CPU cores and the parallel efficiency is 52%. It has excellent scalability.

Example 2. The multi-scale combustion in leading-edge engines (named MuSCLE) is a large-scale CFD parallel software developed by Beijing Institute of Spacecraft Systems Engineering. The software is oriented to structured multi-block, dynamic adaptive, and unstructured grids[22–24]. MuSCLE can be used to guide engineering designs and study key scientific issues in major research projects through high-precision multi-scale algorithms. On the NUDT system, MuSCLE successfully realizes the integrated simulation of the internal and external flows of a hypersonic aircraft, which uses the MPI + OpenMP parallel computing and up to 197 000 cores (Fig. 11). MuSCLE obtains the external flow field and engine internal combustion of the aircraft. The software also acquires the distribution characteristics of the field and the aerodynamic characteristics of the aircraft before and after the ignition of the engine. The parallel efficiency is up to 72%.

Example 3. Shaanxi Key Laboratory of Large-scale Electromagnetic Computing (Computational Electromagnetics (CEM) Team of Xidian University) has designed and developed a complex dense matrix Local Block Pivoting LU (LBPLU) decomposition software system for a new generation of high-performance computing systems. The software system can be used for the algorithm and application research of massive parallel Method of Moments (MoM)[25, 26]. On the NUDT system, LBPLU with local pivoting is used to solve the large dense matrix equation generated by MoM, and the electromagnetic scattering of an aircraft is simulated (Fig. 12). By using LBPLU, 50.16% of parallel efficiency can be obtained when the parallel scale is extended from 24 576 cores to 196 608 cores.

The two typical benchmarks (High Performance Linpack (HPL) and Graph500) have also been optimized and tested on that system[27].

The run of the HPL on the 512 nodes of the exascale prototype uses approximately 144 GB out of the 192 GB...
memory of each node and achieves 2,469.36 Pflops out of a theoretical peak of 3,145.728 Pflops, resulting in a 78.4988% efficiency of the peak performance.

The performance of Graph500 is closely related to memory capacity and bandwidth. Compared with most of the top 10 machines in the Graph500 list, the E-class prototype system of NUDT has smaller memory. By using the data compression technique, the storage space reduces up to 70% and the test scale on the system reaches 27. Finally, the Graph500 tested on the system is 2130.98 GTEPS, which ranks the 10th in the June 2018 Graph500 list.

Acknowledgement

This work was supported by the National Key Research and Development Program of China (No. 2016YFB0200401).

References


Ruibo Wang et al.: Brief Introduction of TianHe Exascale Prototype System


Ruibo Wang received the BS and PhD degrees in computer science from National University of Defense Technology (NUDT) in 2003 and 2011, respectively. He is now an associate professor at NUDT. His research interests include high performance computing and operating system.

Kai Lu received the BS and PhD degrees from NUDT in 1995 and 1999, respectively. He is now a professor in the College of Computer, NUDT. His research interests include parallel programming and operating system and security.

Juan Chen received the PhD degree from NUDT, China in 2007. She is now an associate professor at NUDT. Her research interests include supercomputer systems and energy-efficient software optimization method.

Wenzhe Zhang received the PhD degree from NUDT, China in 2017. He is now an assistant professor at NUDT. His research interests include supercomputer systems and operating system.

Jinwen Li received the BS, MS, and PhD degrees in electronic engineering from NUDT in 1996, 1999, and 2003, respectively. He is now a professor in the College of Computer, NUDT. His research interests include high speed digital circuit and microprocessor technology.

Yuan Yuan received the PhD degree in computer science from NUDT in 2011. He is currently an associate professor at NUDT. His research interests include High Performance Computing (HPC) monitoring and diagnosis and parallel storage systems.

Pingjing Lu received the BS, MS, and PhD degrees in computer science from NUDT, China in 2004, 2006, and 2010, respectively. Currently, she is an associate professor at NUDT. Her current research interests include computer architecture and computer networks.

Libo Huang received the BS and PhD degrees in computer engineering from NUDT, China in 2005 and 2010, respectively. He is an associated professor in the College of Computer, NUDT. His research interests include computer architecture, hardware/software co-design, Very Large Scale Integration (VLSI) design, and on-chip communication. He authored more than 50 papers in internationally recognized journals and conferences.

Shengguo Li received the PhD degree in computational mathematics from NUDT, China in 2013. Currently, he is an assistant professor at NUDT. His research interests include numerical linear algebra, parallel computing, and Computational Fluid Dynamics (CFD).

Xiaokang Fan received the BS and MS degrees from NUDT in 2010 and 2012, respectively. He received the PhD degree from the University of New South Wales, Australia in 2018. Currently he is an assistant professor at the College of Computer, NUDT. His research interests include compiler and program analysis and optimization.