TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 10/15 pp675-681
Volume 22, Number 6, December 2017

SCStore: Managing Scientific Computing Packages for Hybrid System with Containers

Wusheng Zhang, Jiao Lin, Weiping Xu, Haohuan Fu, and Guangwen Yang*

Abstract: Managing software packages in a scientific computing environment is a challenging task, especially in the case of heterogeneous systems. It is error prone when installing and updating software packages in a sophisticated computing environment. Testing and performance evaluation in an on-the-fly manner is also a troublesome task for a production system. In this paper, we discuss a package management scheme based on containers. The newly developed method can ease the maintenance complexity and reduce human mistakes. We can benefit from the self-containing and isolation features of container technologies for maintaining the software packages among intricately connected clusters. By deploying the SuperComputing application Strore (SCStore) over the WAN connected world-largest clusters, it proved that it can greatly reduce the effort for maintaining the consistency of software environment and bring benefit to achieve automation.

Key words: high performance computing; package management; container; hybrid system

1 Introduction

Due to the various package and dependency requirements of different scientific computing applications[1–3], providing a reliable scientific computing environment on a cluster can be a challenging task that involves the following issues: (1) Co-existing of multiple versions of same software packages[4]. There are often scenarios in which a single application may link to multiple versions of libraries in the same runtime environment; moreover, users from multiple application areas utilize various versions of numerous software packages to carry out their simulations[5]. (2) The requirement to keep consistency among all working servers[6]. We should keep a strictly consistent view of software configurations among all the servers involved in the computing farm. It is a nontrivial job to maintain multiple copies of same software packages and to synchronize among different nodes. (3) On-demand update of dynamically linked libraries. There are also cases that involve adding or changing the dynamically linked libraries according to the computation tasks. Those operations should be carried out without interfering the existing working copies or running instances[7, 8]. (4) Easy-to-use backup and recovery[9–11]. For safety and maintenance reasons, we are required to design a mechanism that can help the backup and recovery of the entire application store. We also prefer a backup and recovery scheme without re-configuration operations. (5) Performance. Scientific computing tasks usually are floating point arithmetic intensive applications. The installed packages should assure both high performance and consistent precision. To handle these issues, we developed a globally-shared, and container-managed SCStore in an out-of-the-box manner.

2 Mechanisms of the SCStore

Containers like docker, provide good isolation and self-

• Wusheng Zhang, Weiping Xu, and Guangwen Yang are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China, and National Supercomputing Center in Wuxi, Wuxi 210008, China. E-mail: {zws, xwp, ygw}@tsinghua.edu.cn.

• Jiao Lin is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: linjiao@tsinghua.edu.cn.

• Haohuan Fu is with the Department of Earth System Science, Tsinghua University, Beijing 100084, China, and National Supercomputing Center in Wuxi, Wuxi 210008, China. E-mail: haohuan@tsinghua.edu.cn.

* To whom correspondence should be addressed.

Manuscript received: 2017-10-18; accepted: 2017-11-16

containing service, which are useful for the management of the scientific computing software packages[4]. Our container-based SuperComputing application Strore (SCStore) is a set of daemon and tools, which provide services for handling user environment configuration, package management, execution context isolation, etc. It consists of several basic operations. The basic components of the SCStore are described as follows.

2.1 Application image

The application image is a docker image that provides a set of programs with similar functions. We use these images to pack real world applications or routine libraries and middle ware called by the real applications[12–16].

2.2 Reference graph

We define a reference graph for building the application images as

$$ref\_graph = \{< V, E >\},$$

where

$$V: docker\ image;$$

$$E: link\ between\ two\ docker\ images.$$

A set of application docker images are the vertices of the graph. For example, if an executable inside docker image X depends on a library inside docker image Y, then we have an edge from X to Y. The reference graph can be "calculated" at runtime to form a tree. Different application environments will calculate different trees based on the same reference graph.

2.3 Tracking the dependency

Operation

$$ldd\_map(app):$$

$$\{dependencies \mid app\_name \rightarrow library\_names\}.$$

As shown in Table 1, this operation will find all dependencies of an application executable, create a dependent library set, and then add the elements of the set into a description file called DockerFile. The dependencies of an executable are obtained by running Linux command ldd. The docker images link to each other according to the relation of reference.

Table 1 Operation sub_graph (executable).

(1) Execute the ldd tool, to get dependency outputs.
(2) Parse the outputs to construct a dependency tree.
(3) Follow the dependency tree to walk through the complete reference link of the docker image.
(4) Extract the libraries packed in the linked docker images and build the execution image using related binary files and executable(s).

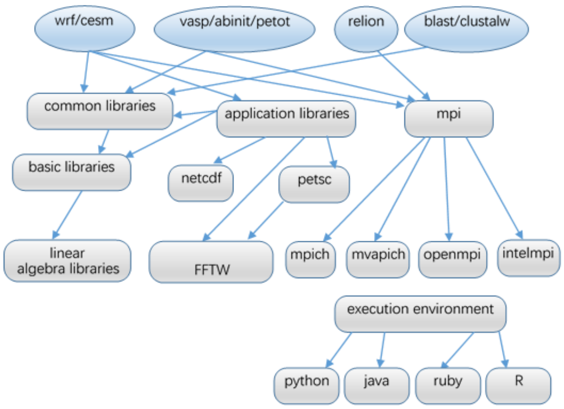


Fig. 1 Reference graph example.

Figure 1 shows part of contents from a reference graph. Each container is configured to export the general information (including the packages and libraries they provide, the configuration parameters a package or library requires, etc.) to their linked pair container.

We also have to define a mapping operation, which is the sub_graph operatoin:

$$sub\_graph\ (executable): \{< V, E >\},$$

where

$$V: the\ docker\ images\ that\ contain\ dependencies\ of\ the\ executable;$$

$$E: the\ link\ relation\ between\ these\ images.$$

This sub_graph operation will extract a sub tree from the reference graph, and the tree will form a final execution docker image. SCStore will export the image to the compute nodes for mounting. Table 1 gives the basic procedure of the sub_grap operation.

2.4 Stage-in and stage-out

The stage-in and -out operation is for synchronization between the docker image and the external file system. We may need to install some packages inside the SCStore image or in the actual OS running on their hosting systems. After finishing the validation and verification, we would use this operation to keep the consistency between the SCStore image and its hosting environment. The core function of stage-in and stage-out operation is to map between the directory structures in SCStore image and in the actual hosting OS. The mapping is guided by the configuration, which translates the directory structure upon stage-in and stage-out.

2.5 Tailoring of the user environment

This operation is intended to prepare execution environment for the users who logged into the computing system. It finds and mounts the docker image instances for the login user, and prepares the environment variables, compiling tools, job submission scripts, and some other user or application specific configurations. When the users login to the frontend server, the login shell preprocess mechanism will trigger the tailor tools to run before the login shell prompts. The tailor program is designed to be extensible. Both the users and the administrator can customize the tool chain and add their task-oriented implementations.

3 Managing Multiple Versions of Software or Libraries as a Unified Runtime Environment

In many scientific computing application areas, the same executables are needed to run on diverse environments (with distinct OS, CPU, network, etc.), and the researchers usually need to run different versions and have them linked to various libraries in order to carry out comparison and verification. Based on the container-aided package managing, we can easily maintain the relationship between executables and their linked libraries. With the reference graph, one can walk through the link relation among containers that define the related binaries, build a dependency tree, and finally create the new application image.

For example, we have a Vasp program running on a CentOS-6.5 and a CentOS-7.3 platform, which adopt Intel Xeon X5670 and E5-2690-V4 CPUs, respectively. The Vasp program links to different libraries and was generated by two different versions of Intel compilers on each platform. We need to provide a unified job submission scheme to each of the users from the above environments respectively, and to enable the program to run in different OS contexts. Figure 2 illustrates the usage of SCStore to hide the version differences of two type of environments from the users. According to the definition of the state-in operation, SCStore can map each version of the Vasp program into the two independent images with the same directory structure.

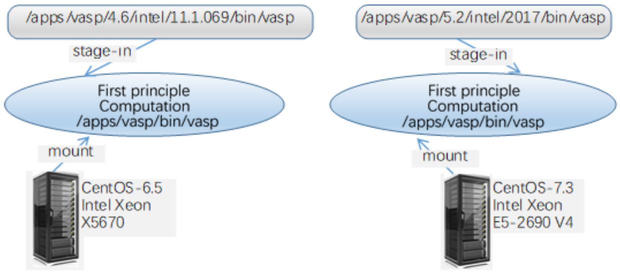

/apps/vasp/4.6/intel/11.1.069/bin/vasp
stage-in
First principle Computation /apps/vasp/bin/vasp
mount
CentOS-6.5 Intel Xeon X5670
/apps/vasp/5.2/intel/2017/bin/vasp
stage-in
First principle Computation /apps/vasp/bin/vasp
mount
CentOS-7.3 Intel Xeon E5-2690 V4

Fig. 2 Directory mapping and version management.

4 Maintaining Consistency Among Sophisticate Environments

In a complex application environment, users could be using different versions of the same software simultaneously in certain cases. The application executables may introduce numerous library dependencies, which can often lead to conflicts between different versions of the same libraries. To detect and solve the conflicts is not an easy task. Base on our container-based SCStore, we can derive the dependency for specific runtime environments, which will further stabilize the performance and certainty of the behavior of the applications. With SCStore, the images of the applications are created to be dependency aware, by tracking the actual package dependencies. Once the software binaries are packaged into the image, it will solidify the dynamic linking relationships; changes on the compute node will not affect the library loading.

5 Package Updating

To update the installed executables and libraries in place without interfering or interrupting the running tasks is another challenge in the management of scientific computing environments. The straightforward approach is to update the same library files located at the leaf of the linked dependency tree. However, if we have tasks that rely on these library files running during the updating process, there will be conflicts introduced. Waiting for the applications to finish is not a cost effective solution, as there could be different jobs that lock the same files for a long period of time.

The SCStore will provide us the ability to update software packages in-place with an “offline” way, and let the online application switch their linking gradually upon restarting.

For example, the climate simulation program CESM

(Fig. 3) will dynamically load a NETCDF library at runtime. Assume that we need to switch the library from old version 4.4.6 to a new version 4.6.7, but there are still some instances running with the old version. A common solution is to wait for the programs to finish, to put the services offline, to get the change done, and then to put the services online again. This may cause a long waiting time, and could be error prone. Instead of this "serial" updating scheme, the SCStore can perform the work in a parallel and pipelined way. While the CESM instances are running on NETCDF 4.6.6, we can finish the installation, testing, and verification with version 4.6.7 on a logically separated environment; then build up the image, with newly created libraries. After exporting the new image, compute nodes will auto-mounted to the appropriate docker instance. SCStore will inject an environment initialization hook into the user's profiling script chain, which will automatically inform and guide the user into the new application context.

6 Keep the Application Farm Trustable and Easy to Backup and Recovery

Scientific computing systems may sometimes have untrusted user logged in. The application binaries may be tampered with or replaced by malicious code. Scan virus in a large-scale application farm means we need to stop the service for a long time. Therefore a backup and recovery mechanism is also critical for a large-scale scientific computing system that supports wide range of applications and huge amount of concurrent users. By applying the container technology, the SCStore can provide a restricted and semi-offline sandbox to host the application binaries; furthermore, it is even able to configure them to be read-only at runtime. In SCStore, the backup and recovery operations are carried out in an out-of-the-box manner, i.e., transfer files in and out only on the servers the docker image is running, and usually located at a sub-network independent of compute nodes. After copying in and out the affected files at the background, the fresh healthy image can be rebuilt and re-exported to the compute nodes again, without interrupting the running system.

7 Implement User Oriented Deployment

Scientific computing environments are usually domain-specific. For example, there are not any common library dependencies between the climate simulation model and the first principle calculation, except that they may be commonly linked to some low level libraries such as glibc. It provides a cleaner environment if we isolate one application domain from another. SCStore can encapsulate related applications into partitioned container linking trees, and share the same low layer libraries among different application images. Such an approach can greatly simplify the users environment, and keep the runtime sane and clean. Figure 4 shows the application area oriented encapsulation and partitioning. Only a few packages are listed in the figure. For example, with the FFTW package, some applications may dynamically link to FFTW-3.3.3 generated by gcc-5.4.0, but some others may depend on the versions compiled with Intel compiler 2015.

In this example, User A comes from the Department of Physics and mainly carries out the material simulation work; and User B comes from the research group of biology, who mainly performs the bioinformatics computation work. Executables from both application areas may depend on the same underlying libraries with variant version number

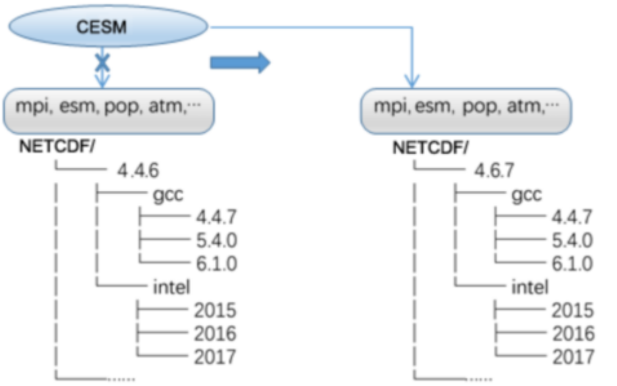


Fig. 3 Package updating and image switching.

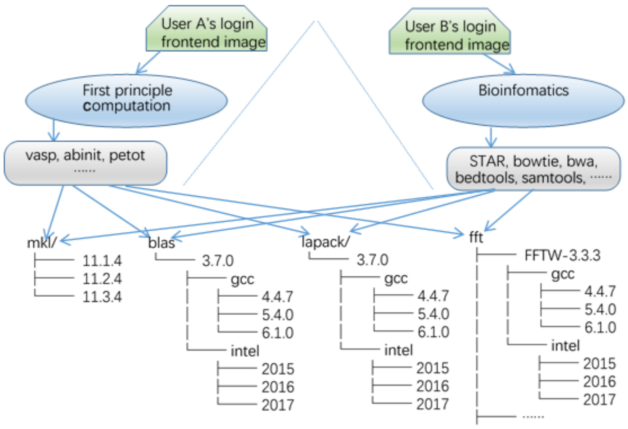


Fig. 4 Package updating and image switching.

or different compiling schemes. It is confusing to configure the global environment variables to accommodate the differences between the users, especially with the possibility of PATH conflicts. SCStore hides these conflicts by packing the executables and their dependencies into a unified-structured image; and furthermore, tools provided by the SCStore can track the dependencies and maintain an identical directory organization. This function is also helpful for license management, providing exclusive access to only authorized users.

8 Address Performance Evaluation and Verification Issues

When we update some underlying libraries, before switching to the newly installed package, we usually have to confirm if the computation can be performed as before. Users of such applications pay close attention to the correctness and performance. When we update and alter the package in-place, this could interfere with the running applications. SCStore is able to support the performance evaluation and correctness verification after in-place updating.

As shown in Fig. 5, assume we need to run new linpack testing with Mvapich version 2.3a, but there is already a test running on compute node set *x*, which links Intelmpi version 4.1. We shall not affect the task on compute node set *x*, but need to start a new test with compute node set *y*. To achieve this, we create a new docker image B based on the original version of image A, update the HPL binary with Mvapich version 2.3a, and then switch the mount point on compute node set *y* to the newly created docker instance B. The new testing will run simultaneously without affecting the running task on compute node set *x*. After the two test are finished, we can verify the results to ensure the performance and correctness. Depending on the utilization situation, the two versionss of HPL can be kept at the same time or the old version will be replaced with new one.

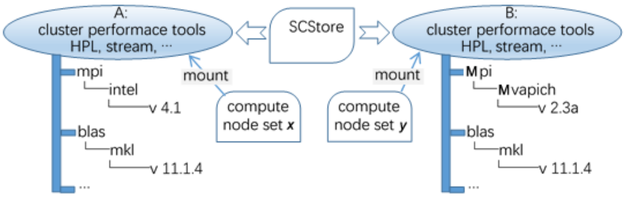


Fig. 5 On-the-fly verification and performance evaluation.

9 Deployment

SCStore was deployed on the supercomputing platform of Tsinghua University and National Supercomputing Center in Wuxi (WXSC); both are running large scale of computing clusters, which perform thousands of jobs per day submitted from multiple application areas. We built SCStore on the clusters with a linked graph of docker images.

SCStore consists of several SCStoreServers, management daemons, and a set of utilities. The SCStoreServers that host libraries, application executables, and tools are named Appserver[0-n] or psn[001-n]. Docker image instances which pack applications, and their dependencies are run among the Appservers according to load balance and management strategy. The background daemons running on Appservers are in duty of image managing.

Sunway TaihuLight supercomputer is a hybrid scientific computing system[10]. Table 2 shows the main hardware types in National Supercomputing Center in Wuxi (WXSC). It consists of 40 960 Sunway nodes, 980 2-way Intel xeon E5-2680 v3 nodes, 32 8-way Intel E7-8860 v3 nodes, and 64 Nvidia K40 nodes.

For the purpose of resource sharing, we also connect the campus cluster in Tsinghua University with WXSC via multiple CERNET tunnels (Fig. 6), with the two systems interoperable to each other. This means we should prepare several formats and/or versions of the same software, and train the user how to tell the differences between multiple versions.

Figure 7 shows a simplified view of our SCStore, which is deployed both on the supercomputing platform of Tsinghua University and WXSC, except that SCStore on WXSC contains special image instances for the Sunway system. The application software can be open source, commercial license, or self-developed, which requires different usage strategy

Table 2 Hardware summary from Tsinghua and WXSC.

	Node type	Amount	Feature
WXSC	Sunway	40 960	SUNWAY 26010
	Sugon 2-way	980	Intel E5-2680 v3
	Sugon 8-way	32	Intel E7-8860 v3
	Inspur GPU	64	NVidia K40
Tsinghua	Inspur 2-way	800	Intel X5670
	Dell 2-way	64	Intel E5-2670 v2
	Dell 2-way	12	Intel E7-4830 v2
	Nvidia GPU	32	Intel E7-8860 v3

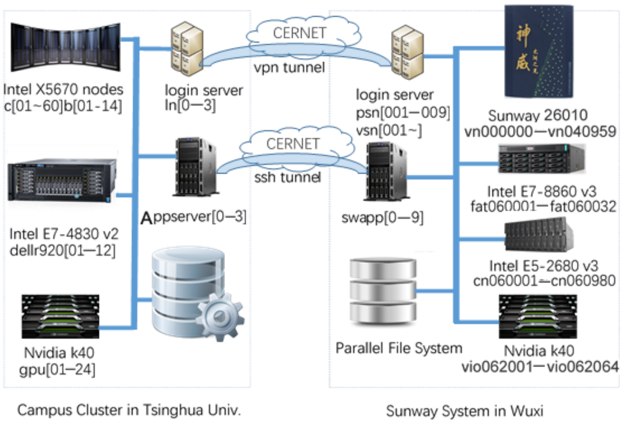


Fig. 6 Illustration of system deployment.

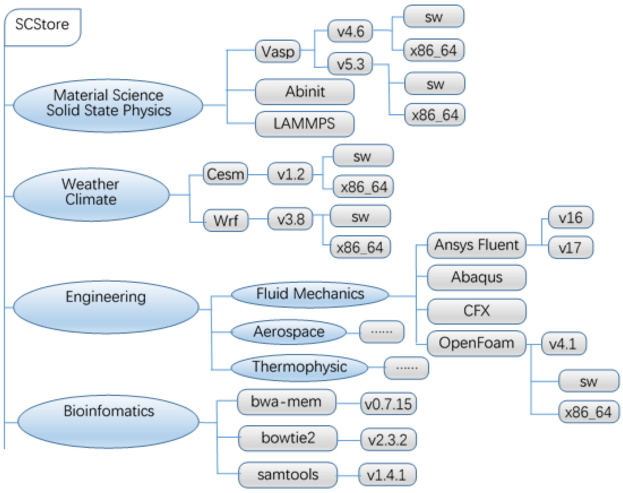


Fig. 7 SCStore demonstration.

and/or pattern. For example, users who develop program by themselves may need privacy protection from other users; commercial licensed software may not be allowed to be called by non-authorized users; administrators may not want the fluid mechanics programs to appear in their working area of users from bioinformatics. At the same time, we may hope users from the Material Science Department and the Thermo-physics Department can both have access to the first principle program Vasp. Some software codes have versions for both the Sunway platform and the x86_64 clusters, such as Vasp, Wrf, Cesm, LAMMPS, OpenFoam, etc. Running of these programs needs to distinguish Sunway jobs and x86_64 jobs. We use SCStore to deploy and dispatch the self-contained docker image as software farm for user login environment and execution environment. In above systems, application images are packed dynamically according to the user group and management strategy.

10 Conclusion

We developed a software package management system for supercomputer system. The SCStore utilizes the container technology to ease the installation and updating of software on a complex computing environment. With the SCStore, one can easily achieve strict consistency of the status of software packages among huge amount of computing nodes. It can help the administrator to obtain automation, and it also can bring benefit on DevOps of the supercomputer system. The deployment on the Taihu-light Supercomputer and the campus computing facility of Tsinghua University has proved that it can work smoothly in a WAN connected heterogeneous computing environment.

Acknowledgment

Thanks to Prof. Shimin Hu, who gives a lot of valuable comments and advises upon preparing this paper. This work was supported by the National Key R&D Program of China (No. 2016YFA0602100), the National Natural Science Foundation of China (No. 91530323), and Open Fund of Key Laboratory of Data Analysis and Applications, SOA (No. LDAA-2014-03).

References

[1] G. Vallee, T. Naughton, S. Bohm, and C. Engelmann, A runtime environment for supporting research in resilient HPC system software & tools, in *International Symposium on Computing and Networking (CANDAR)*, 2013, pp. 213–219.

[2] A. Morari and M. Valero, HPC system software for regular and irregular parallel applications, in *IEEE International Symposium on Parallel & Distributed Processing Workshops and PhF Forum*, 2013.

[3] K. Wang, A. Kulkarni, M. Lang, D. Arnold, and I. Raicu, Exploring the design tradeoffs for extreme-scale high-performance computing system software, *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 1070–1084, 2016.

[4] T. Adufu, J. Choi, and Y. Kim, Is container-based technology a winner for high performance scientific applications? in *Network Operations and Management Symposium (APNOMS)*, 2015, pp. 507–510.

[5] K. Lv, Z. Zhao, R. Rao, P. Hong, and X. Zhang, PCCTE: A portable component conformance test environment based on container cloud for avionics software development, in *International Conference on Progress in Informatics and Computing (PIC)*, 2017, pp. 664–668.

[6] T. Gamblin, M. LeGendre, M. R. Collette, G. L. Lee, A. Moody, B. R. de Supinski, and S. Futral, The spack package manager: Bringing order to HPC software chaos, in *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysisy*, 2015, pp. 1–12.

[7] G. Becker, P. Scheibel, M. L. Gendre, and T. Gamblin, Managing combinatorial software installations with spack, international workshop on Hpc user support tools, in *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysisy*, 2016, pp. 14–23.

[8] E. Dolstra, M. de Jonge, and E. Visser, Nix: A safe and policy-free system for software deployment, in *Proceedings of the 18th Large Installation System Administration Conference (LISA XVIII) LISA'04*, 2004, pp. 79–92.

[9] K. Hoste, J. Timmerman, A. Georges, and S. De Weirdt, Nix: A safe and policy-free system for software deployment, in *High Performance Computing Networking Storage and Analysis Proceedings*, 2012, pp. 572–582.

[10] G. Yang, W. Zhao, N. Ding, and F. Dun, "Sunway Taihulight" and its applications, *KEXUE*, vol. 69, no. 3, pp.12–16, 2017.

[11] A. DiGirolamo, The smithy software installation tool, http://github.com/AnthonyDiGirolamo/smithy, 2012.

[12] HashDist: Reproducible, Relocatable, Customizable, Cross-Platform Software Stacks for Open Hydrological Science, 2013, http://github.com/hashdist.

[13] M. Howell, Homebrew the Missing Package Manager for OS X, http://brew.sh.com, 2017.

[14] ROCKS: Open Source Toolkit for Real and Virtual Clusters, http://www.rocksclusters.org, 2017.

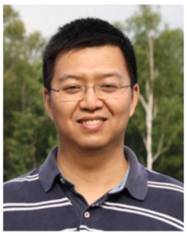
[15] The MacPorts Project Official Homepage, http://www.macports.org, 2017.

[16] About FreeBSD Ports, http://www.freebsd.org/ports, 2017.

Wusheng Zhang received the MEng and PhD degrees from Tianjin University and Tsinghua University, in 2007 and 2010, respectively. He focuses on research of parallel and distributed computing.

Jiao Lin obtained the MEng degree from Tsinghua University in 2005. Her current research interest is parallel computing.

Weiping Xu received the BEng degree from Tsinghua University in 1992. He is in duty of the system administration of National Supercomputing Center in Wuxi and the campus computing platform in Tsinghua University. He is interested in cluster computer system, parallel file system, and engineering computing.



Haohuan Fu recieved the BEng degree from Tsinghua University (2003), MPhil degree from City University of Hong Kong (2005), and PhD degree from Imperial College London (2009). He is the deputy director of the National Supercomputing Center in Wuxi, and also an associate professor in the Ministry of Education Key Laboratory for Earth System Modeling, and Department of Earth System Science in Tsinghua University. His research focuses on providing both the most efficient simulation platforms and the most intelligent data management and analysis platforms for geoscience applications.

Guangwen Yang is a professor in the Department of Computer Science and Technology at Tsinghua University and the director of the National Supercomputing Center in Wuxi. His research interests include parallel algorithms, cloud computing, and the earth system model. He received the PhD degree in computer science from Harbin Institute of Technology in 1996. He has received the ACM Gordon Bell Prize. He is a senior member of CCF.