SCEAPI: A unified Restful Web API for High-Performance Computing

Cao Rongqiang, Xiao Haili, Lu Shasha, Zhao Yining, Wang Xiaoning and Chi Xuebin

Computer Network Information Center, Chinese Academy of Sciences, Beijing, China

E-mail: caorq@cnic.cn

Abstract. The development of scientific computing is increasingly moving to collaborative web and mobile applications. All these applications need high-quality programming interface for accessing heterogeneous computing resources consisting of clusters, grid computing or cloud computing. In this paper, we introduce our high-performance computing environment that integrates computing resources from 16 HPC centers across China. Then we present a bundle of web services called SCEAPI and describe how it can be used to access HPC resources with HTTP or HTTPS protocols. We discuss SCEAPI from several aspects including architecture, implementation and security, and address specific challenges in designing compatible interfaces and protecting sensitive data. We describe the functions of SCEAPI including authentication, file transfer and job management for creating, submitting and monitoring, and how to use SCEAPI in an easy-to-use way. Finally, we discuss how to exploit more HPC resources quickly for the ATLAS experiment by implementing the custom ARC compute element based on SCEAPI, and our work shows that SCEAPI is an easy-to-use and effective solution to extend opportunistic HPC resources.

1. Introduction

High Performance Computing (HPC) is increasingly moving to the web, and even to mobile applications. In our work, we have noticed that these science gateways and collaborative applications are now capable of supporting lots of applications for diverse research areas, and even more computational resources are needed. Many of these gateways and collaborative applications involve bridging interactive front-end interfaces with backend supercomputers and storage resources. However, it is difficult for developers to develop an easy-to-use, reliable and secure library for accessing computing resources.

With the development of grid computing and cloud computing, we have integrated massive HPC resources distributed geographically across China. On the basis of these massive resources and diversified demands, we present and maintain a bundle of web services called Scientific Computing Environment API (SCEAPI) that provides an essential and unified RESTful Web API for accessing HPC resources, including authentication, file transfer and management, job management and so on.

There are two important advantages in SCEAPI. First, SCEAPI is maintained by a professional team consisting of experienced experts and technicians in HPC and web services. They maintain the SCEAPI service availability, and update services transparently to deal with dynamic changes in computing resources, communication protocols and so on. Another important advantage of SCEAPI is to provide an easy-to-use way to access massive and economic HPC resources because SCEAPI
is built on the top of China National Grid (CNGrid) [1] and Scientific Computing Grid (ScGrid) [2] of Chinese Academy of Sciences (CAS). With the advances of SCEAPI, developers do not need to deal with the complexities caused by accessing the HPC resources, so that they can concentrate on building their own scientific applications themselves and workflows. Using SCEAPI, various third-party science gateways and collaborative applications have been developed and deployed in high energy physics, chemistry, bioinformatics and many other research areas for many scientific computing applications, for example, the ATLAS[3] Monte Carlo (MC) simulation application.

The structure of this paper is as follows: Section 2 presents related work about RESTful web services in HPC. Section 3 introduces CNGrid and ScGrid that provide massive HPC resources. Section 4 discusses the architecture of SCEAPI and describes functions of each API. Section 5 describes how ATLAS extends its resource pool to HPC resources quickly with SCEAPI and evaluates the effectiveness of the integration. Finally Section 6 summarizes the paper.

2. Related work
With grid computing and cloud computing middleware, computing environment aggregates massive resources consisting of supercomputers, clusters, and storages through high bandwidth networks. Many research areas and scientific collaborations need an easy-to-use and scalable way to access these massive resources aggregated by computing environment. Representational State Transfer (REST) is a software architecture style that aims to reduce the complexity of software and improve scalability [4]. RESTful web service is a good way to connect science gateways and collaborative applications to grid computing and cloud computing resources. With the benefits from REST and HTTP, RESTful web service can support cross-platform and cross language features. Besides, it is more lightweight in design, much simpler in style of API, and more convenient to implement than the big web service [5].

In HPC domain, several RESTful web services have emerged and become a popular way to leverage computing capabilities. National Energy Research Scientific Computing Center (NERSC) proposed a single RESTful web service called NERSC Web Toolkit (NEWT) that brings a heterogeneous collection of backend computing resources to the web applications via HTTP and JSON [6]. NERSC also proposed a customizable NEWT framework that can be deployed at a HPC center, to enable access to various backend resources and services through a common web API [7]. Under NEWT framework, several interfaces need to be implemented before providing a similar API as NEWT. SCEAPI is designed to provide a unified API for massive resources in ScGrid and CNGrid, and we do not need to write additional source code when a HPC resource is added or removed.

In recent years, Globus have transformed capabilities into web services and APIs, such as Globus Nexus [8] and Globus Galaxies platform [9]. These APIs allow external and third-party services to leverage Globus data management, authentication, and authorization capabilities as a platform when building computing portals. Compared with the REST API in Globus, SCEAPI is a lightweight and independent web API on the top of ScGrid and CNGrid, and specifically focused on core functions of the usage of HPC, including job submission and management, file transfer and management, and simple login/logout management. The Extreme Science and Engineering Discovery Environment (XSEDE) system is an integrating framework for lots of advanced digital resources, for example, supercomputers, collections of data, making them easier to use and helping more people use them. The XSEDE system provides not only its own REST API [10], for example, resource information services, but also provides some integrated REST API, for example, identity and group management providing API based on Globus nexus. Similar to XSEDE, SCEAPI also integrates API of China Science and Technology Network (CSTNET) passport to support single sign on for several kinds of clients.

3. Computing Environment
In recent years, HPC has developed rapidly in China. Several National Supercomputer Centers have built or upgraded very large supercomputers and the total performance of them is more than 150 petaflops. CAS also has built a petaflops supercomputer and deployed it in our center [18]. In addition, dozens of teraflops-scale supercomputers have been built and installed in many universities and
institutes. CNGrid has integrated these HPC resources into a virtual supercomputer by the lightweight middleware SCE developed by our center [11]. As the important part of CNGrid, ScGrid has integrated tens of clusters and supercomputers in CAS based on the SCE middleware.

3.1 CNGrid
CNGrid [1] is a key project launched in May 2002 and supported by the China National High-Tech Research and Development Program (863 program). The mission of CNGrid is to continually promote the construction of national information capabilities and speed up the development of relevant industries by technical innovation. CNGrid has integrated HPC resources across all China in the last more than 10 years, comprising 4 National Supercomputer Centers of Tianjin, Jinan, Changsha, and Shenzhen, and also dozens of teraflops-scale HPC resources from universities and institutes. At present, there are 16 supercomputers and tens of clusters connected by SCE middleware in CNGrid. In order to make full use of the existed HPC resources and provide better computing services for users in different research areas and disciplines, several collaborative communities are built based on SCEAPI in industry, medicine, and multi-media and other research areas.

3.2 ScGrid
ScGrid [2] is a general-purpose computing platform started from 2006 in CAS, which provided a problem solving environment for computing users through grid computing and cloud computing technologies. ScGrid is designed as a three-tier pyramidal structure. The top layer is a centralized large-scale supercomputer called ERA [18], which is a heterogeneous supercomputer and its peak performance is 2.3 petaflops. The middle layer contains 9 branch centers distributed among CAS. ScGrid has 18 sub-branch centers and 11 GPU centers in the bottom layer. All these HPC resources belonged to CAS are integrated into a virtual supercomputer by SCE middleware. ScGrid provides high performance computing services for scientists and students from different research areas in CAS.

4. Architecture and RESTful API
SCEAPI is a group of RESTful web services based on SCE middleware. Using SCEAPI, it is easy and quick to integrate massive computing capabilities with science gateways and collaborative applications. In architecture design, we focused on loosely-coupled style to acquire good scalability and shield heterogeneity and complexity of HPC resources and various demands. In implementation, we applied widely accepted knowledge for the definitions of the RESTful API and used a popular open source framework called Jersey to implement SCEAPI. For security and risk prevention, we adopted two methods to protect the API and HPC resources from malicious access, including the Uniform Resource Locator (URL) signature and parameters checking in format and variable type.

4.1 SCEAPI Architecture
The core rules of SCEAPI are simplicity, compatibility and transparency. SCEAPI provides easy and simple to use interfaces between third-party applications and HPC resources, such as user login/logout, job submission and management, file transfer and management, and so on. All these interfaces are easy-to-use RESTful API that is consistent with the usage of scientific computing and HPC. In addition, the API maintains compatibility with all older versions from 2013 when they started to provide unified computing capabilities. SCEAPI adapted a loosely-coupled hierarchical architecture to ensure good scalability and deal with frequent changes in demands, protocols and security limits. Compared to a classical Browser/Server (B/S) architecture, all science gateways and collaborative applications called clients are similar to different kinds of web browsers, which access HPC resources through SCEAPI by HTTP and AJAX in different programming language and platform. As shown in Figure 1, SCEAPI is a group of RESTful web services consisting of the unified RESTful API
interfaces, services and other necessary modules. Each layers and important modules are described in
detail as follows.

![Layered Architecture of SCEAPI](image)

**4.1.1 Clients layer.** Clients are applications developed by different third parties to diverse demands
from different disciplines and research areas. These clients could be grouped into several categories,
for example, science gateways, web and mobile collaborative, simple scripts and so on. SCEAPI
smoothes the learning curve of accessing massive heterogeneous HPC resources so that many entry-
level developers and even proficient users could be attracted to develop new clients for satisfying their
customized demands. Developers can choose their favorite languages and platforms firstly, and then
they focus on their demands because SCEAPI provides easy-to-use API for bridging different clients
to massive HPC resources.

**4.1.2 Web services layer.** This layer receives requests from the upper layer and returns responses. This
layer is composed of three combined groups.

- **The first group involves 3 modules including API definitions, user management and URL signature.** The API definitions module specifies a general message structure, input parameters and specified response structures for each API, detailed in 4.3. When updating a new version, we keep all existing definitions unchanged. If we should break the definition, a new definition will be added to keep compatibility so that existing clients do not need to be updated. The simple user management module provides login and logout functions and validates the request belonging to an authenticated user by the token carried in the request headers. The URL signature model validates whether a request is sent by a real user or forged by an attacker by calculating and comparing the md5 sequences, detailed in 4.2.

- **The second group contains several functional modules for job, HPC resources and data.** The format check module validates whether input parameters are strictly consistent with the definition of each API. The resource query module provides list for HPC resources and applications list, and available computing queues according the application and estimated wall time of each job. The job management module is designed to perform job submitting, status query, and job terminating. The file transfer module provides uploading and downloading file, viewing text remotely, listing directory and so on.

- **The last group comprises SCE library and LDAP library.** The former library encapsulates networks protocols and enables multi-threading support; the latter library provides unified interfaces to authenticate user identity in different persistence storages, such as relational database and LDAP.

**4.1.3 Foundation layer.** This layer provides HPC resources aggregated by SCE middleware and
account information stored in database. Besides, the web container that is composed of Apache HTTP
server and Tomcat is used to provide a runtime environment for SCEAPI. If necessary, more Tomcat
servers could be deployed dynamically to handle increasing requests coming from the HTTP server
that receives calls from clients and distributes these requests to any available Tomcat server.

**4.2 Implementation and security**
SCEAPI is a group of RESTful web services implemented on SCE middleware and Jersey RESTful Web Services framework. Several key issues in implementation and security are discussed as follows.

4.2.1 Framework for RESTful web services. Jersey is open source and production quality framework for developing RESTful web services in Java that provides support for JAX-RS API and serves as a JAX-RS (JSR 311 & JSR 339) reference implementation [12]. Besides, Jersey provides its own API that extends the JAX-RS toolkit to further simplify RESTful service and client development. Based on Jersey, we defined and implemented SCEAPI and provide easy-to-use and reliable RESTful web API for developers. In addition, we defined a group of strict constraints for all input parameters based on bean validation in Jersey to prevent malicious attacks. Moreover, all SCEAPI could return information in JSON or XML formats based on common media representations provided by Jersey according the accepted format of each request, except the file downloading request that returns text or binary data.

4.2.2 Interaction with computing resources. Developed by our supercomputer center belonged to CNIC (Computer Network Information Center) from 2007, SCE middleware is a lightweight grid middleware mainly based on the OPENSSH software package, and provides grid security, job management, data management and information service [11]. Using SCE middleware, SCEAPI is a group of efficient and reliable web services that maps local functions to remote HPC resources, and shields the difficult challenges of networks, rapidly evolving protocols and maintenance issues.

4.2.3 File transfer. Multiply-node Copy (MCP) was added to SCE middleware when 2013, which proposed a file transfer agent making data stream to flow along the specific intermediate nodes based on Secure File Transfer Protocol (SFTP) and provided secure data transfer service in wide area networks [13]. Using the MCP module of SCE middleware, SCEAPI implemented an independent micro-service to provide scalable data transfer service especially for data-intensive computing tasks. If necessary, more instances of the micro-service could be deployed to meet large-scale data transfer.

4.2.4 Security enhancement. In order to enhance security of SCEAPI and protect inadequate HPC resources, some measures are taken to protect sensitive data transferred in networks and prevent unauthorized access launched by malicious attackers. For login action, identity information including account, password and other credential is needed to send through unsafe networks in the HTTP request. The server would return some sensitive information, such as token and md5 secret sequences, if the login action is successful. So we suggest developers call the login action in POST method without credentials in URL and transfer data in HTTPS protocol to protect sensitive information.

Moreover, for other calls to SCEAPI, each URL must have two extra parameters called timestamp and MD5 signature. The client and server separately assemble a clear sequence consisting of HTTP method, URL, timestamp, the MD5 secret sequence and other parameters in specific order, and then calculate the signature with the MD5 algorithm. When a client sends a request, a signature is calculated and carried in URL. The server receives the request, calculates the signature again and checks whether the two signatures are the same character sequences. The server also checks whether the offset between the timestamp and current time is less than constant value to prevent replay attacks.

4.3 Unified RESTful Web API

Compared with traditional API encapsulated in a library, SCEAPI is a group of lightweight web services in essence and defined by standard HTTP methods that define function type, such as GET, POST, PUT, and DELETE, and a group of URI that defines signature, scope and parameters of each function. All functions of SCEAPI are as shown in table 1, for example, simple user management, resource query, job management, file transfer and other services. There are 3 variables represented by {\}, ‘ujid’ indicating the Identified (ID) number of a job, ‘app’ indicating the name of an application, and ‘file’ indicating the name of a file.

There are two typical scenarios for SCEAPI, job submission and job management. When submit a job, a client firstly calls resource query functions to select an application and a suitable computing queue. If a queue called cloud is selected, the SCE middleware will choose a best match queue for the job. Then the client submits all the parameters of the job to the server and gets an ID of the job. Then the client uploads zero or more input files to the server by calling files transfer functions with the ID.
And finally the client marks the job ready to start by calling a status API and then the job will be processed automatically in HPC resources. In the scenario for job management, a client could call different functions to get a group of jobs filtered by combined conditions, such as status, submission time, and application name. For a specific job, a client could call different functions to list files in cache and HPC storages, and terminate the job and/or view content of any text file when the job is running, and download output files when the job has finished.

Table 1. A list of important functions in SCEAPI

|                | Method | URL                                      | Description                               |
|----------------|--------|------------------------------------------|-------------------------------------------|
| User Management| POST   | /users/login                              | login to the system                       |
|                | GET    | /users/logout                             | logout from the system                    |
| Resource Query | GET    | /resources/hpcs                          | get a list of HPC nodes                   |
|                | GET    | /resources/applications                   | get a list of applications                |
|                | GET    | /resources/applications/{app}             | get a list of available HPC queue         |
| Job Management | PUT    | /jobs                                    | submit a job to the system                |
|                | DELETE | /jobs/{ujid}/status                       | set the status of the job                 |
|                | GET    | /jobs/{ujid}                              | kill the job                              |
|                | GET    | /jobs/ {ujid}/[{cs |hpc}]                | get a list of jobs filtered by conditions.|
| File Transfer  | GET    | /data/jobs/{ujid}/hpc/{file}/view        | view a text file of the job               |
|                | GET    | /data/jobs/{ujid}/filesUpload             | get a URL for uploading files             |
|                | POST   | URL for uploading files                   | upload files for the job                  |
|                | GET    | /data/jobs/{ujid}/fileDownload            | get a URL for downloading the file        |

5. Collaboration with ATLAS

Processing and analyzing data taken from the experiments at the Large Hadron Collider (LHC) is a major computational challenge. Data processing is based on the World wide LHC Computing Grid (WLCG), where computing sites were usually dedicated clusters specifically set up to meet the needs of the LHC [14]. Considering HPC resources, ATLAS showed the feasibility of running MC detector simulation jobs for the ATLAS in the Swiss National Supercomputing center [15]. The ATLAS team also implemented a SSH backend to the Advanced Resource Connector (ARC) middleware to enable HPC compliant access in compliance with the strict HPC policies [16]. Therefore, we proposed a solution to bridge the ATLAS job submission to massive HPC resources in CNGrid and ScGrid through SCEAPI and the ARC compute element.

5.1 SCEAPI client in Python

In order to reduce the difficulty and complexity, a library based on SCEAPI was implemented in Python, shown as figure 2. The library transforms all local calls to HTTP remote calls in order to shield the complexity in networks programming with HTTP. For simplicity, security signature with URI was implemented in the module called md5 generation for all functions except the login and logout functions. There are several services related to job submission and management in the library, for example, user authentication, submitting jobs, querying status of jobs, killing jobs, file uploading and downloading, listing workspaces, and viewing contents of a file when the file is existed. Some PBS-like commands were also implemented to verify whether the library could work and help developers learn how to use the library. With these examples, it is easy for developers to adjust their procedures to submit and manage jobs in CNGrid and ScGrid.
5.2 ARC Compute Elopement on SCEAPI

A Compute Element (CE) is used to receive workloads, handle user authentication, submit jobs to the backing computing resources and stage of input and output files in ARC middleware. In order to extend resources pool and exploit more HPC resources, a customized integration to the ARC-CE has been implemented with SCEAPI library to solve issues caused by the strict polices of HPC resources.

For authentication, one single account is enough to accessing all HPC resources in CNGrid and ScGrid because the SCE middleware will map this account to a local account of each HPC resource. In addition, the HPC resource cannot mount remote storage because of security policies. To stage files, input files are transferred to the ARE-CE server firstly. Then the customized ARE-CE submits a job, upload files to the job workspace, and mark the job ready to start through SCEAPI. When the job is completed, it downloads output files to its own server by SCEAPI and transfers them to the target storages. Moreover, the customized ARE-CE checks status of the unfinished jobs periodically.

As discussed above, developers from the ATLAS team and our supercomputer center implemented the customized ARC-CE with the SCEAPI client and integrated it into the ATLAS production system.

5.3 Evaluation

The customized ARC-CE was deployed on two servers that connected to the SCEAPI by CSTNET. We also deployed the MC softwares in the supercomputers ERA and TianHe-1A manually. ERA is the first petaflops-scale supercomputer in our center and its peak performance is 2.36 petaflops [18]. TianHe-1A also is a petaflops-scale supercomputer in Tianjin National supercomputer Center and its peak performance is 4.7 petaflops. Because supercomputers could not mount remote storage, software including MC Simulation and other applications were downloaded to an intermediate server. Then we uploaded the softwares to the supercomputers, installed, and tested to make sure the softwares work correctly. In addition, we update the installations when the ATLAS releases new versions.

The ATLAS production system started to submit jobs to CNGrid and ScGrid from May 2015 through SCEAPI. The ATLAS has successfully computed more than 11,000 of MC jobs that consumed 390,000 hours, shown as figure 3. The result showed that it is an effective way for the ATLAS to find and exploit much more opportunistic HPC resources. Furthermore, it also showed that it is available and reliable to provide HPC services for data-intensive computing jobs by SCEAPI.

6. Conclusion

SCEAPI is a unified RESTful web API for HPC and enables access to massive HPC resources aggregated by SCE middleware in ScGrid and CNGrid. SCEAPI provides developers a set of easy-to-use programming capabilities for accessing HPC resources, such as authentication file transfer and management, job management and so on. Using SCEAPI, developers don not need to deal with the
complexities in accessing HPC resources so that they can focus on building their own scientific collaborative applications, personalized script and so on for diverse demands. Using SCEAPI, a custom integration into the ATLAS job submission has been implemented easily and rapidly. The collaborative work shows that it is an effective and interesting work to extend and exploit massive HPC resources swiftly for the ATLAS experiment. In future, we will improve utilization of networks bandwidth to support data-intensive jobs much better.

Acknowledgments
Thanks are due to Mr. Andrej Filipci for integrating SCEAPI into the ATLAS production system. This paper was funded in part by the National Key Research and Development Program in the Ministry of Science and Technology (MST) under the contact number 2016YFB0201404, and by the national “863 Program” in MST under the contact number 2014AA01A302.

References
[1] Depei Q. CNGrid: A test-bed for grid technologies in China[C]//Distributed Computing Systems, 2004. FTDCS 2004. Proceedings. 10th IEEE International Workshop on Future Trends of. IEEE, 2004: 135-139.
[2] Xuebin C., Hai-li X., Rongqiang C., etc. Scientific Computing Grid and SCE Middleware[J]. Journal of Integration Technology, 2012, 1: 012.
[3] Goshaw A T. The ATLAS Experiment at the CERN Large Hadron Collider[J]. Center for Strategic and International Studies, 2009.
[4] Fielding R T. Architectural styles and the design of network-based software architectures[D]. University of California, Irvine, 2000.
[5] Pautasso C, Zimmermann O, Leymann F. Restful web services vs. big'web services: making the right architectural decision[C]//Proceedings of the 17th international conference on World Wide Web. ACM, 2008: 805-814.
[6] Cholia S, Skinner D, Boverhof J. NEWT: A RESTful service for building High Performance Computing web applications[C]//Gateway Computing Environments Workshop (GCE), 2010. IEEE, 2010: 1-11.
[7] Cholia S, Sun T. The NEWT platform: an extensible plugin framework for creating ReSTful HPC APIs[J]. Concurrency and Computation: Practice and Experience, 2015, 27(16): 4304-4317.
[8] Ananthakrishnan R, Bryan J, Chard K, et al. Globus Nexus: An identity, profile, and group management platform for science gateways and other collaborative science applications[C]//Cluster Computing (CLUSTER), 2013 IEEE International Conference on. IEEE, 2013: 1-3.
[9] Madduri R, Chard K, et al. The Globus Galaxies platform: delivering science gateways as a service[J]. Concurrency and Computation: Practice and Experience, 2015, 27(16): 4344-4360.
[10] Bachmann F, Foster I, et al. XSEDE architecture overview[R]. Technical Report, 2014.
[11] Dai Z, Wu L, et al. A lightweight grid middleware based on OPENSSH-SCE[C]//Grid and Cooperative Computing, 2007. Sixth International Conference on. IEEE, 2007: 387-394.
[12] Oracle Corporation, Jersey- RESTful Web Services in Java, 2016. https://jersey.java.net
[13] Yuhuan L, Xiaoning W, et al. Data Transfer Transmission for Cloud Service Environment of Scientific Computing[J]. e-Science Technology & Application, 2013, 4(5): 42-50.
[14] Bird I G. LHC computing (WLCG): Past, present, and future[J]. Grid and Cloud Computing: Concepts and Practical Applications, 2016, 192: 1.
[15] Filipic A, Haug S, Hostettler M, et al. ATLAS computing on CSCS HPC[C]//Journal of Physics: Conference Series. IOP Publishing, 2015, 664(9)
[16] Haug S, Hostettler M, Sciaccia F G, et al. The ATLAS ARC backend to HPC[C]//Journal of Physics: Conference Series. IOP Publishing, 2015, 664(6).
[17] CERN, ATLAS dashboard. http://dashb-atlas-job.cern.ch.
[18] CAS New Petascale Supercomputer “Era” Now Online, Computer Network Information Center. http://english.cnic.cn/ns/es/201407/t20140707_123836.html.