Enhancing UNICORE Storage Management Using Hadoop Distributed File System

Wasim Bari[1], Ahmed Shiraz Memon[2], and Bernd Schuller[2]

[1] Rheinisch-Westfälische Technische Hochschule (RWTH)
52056 Aachen, Germany
[2] Institute of Advanced Simulation, Jülich Supercomputing Centre
Forschungszentrum Jülich GmbH
D-52425 Jülich, Germany

Abstract. UNICORE is a state of the art and well tested Grid middleware, designed for seamless and secure access to distributed resources, applications and data, in an easy to use fashion. A wide variety of UNICORE applications for example in bio-informatics generate and compute huge amounts of data. These large amounts of data are not easy to manage reliably and efficiently with the default UNICORE storage system which is using a standard file system. Hence, the current UNICORE does not support a scalable distributed storage system so far. We have integrated Apache Hadoop and its supported distributed storage/file systems into the UNICORE storage management service. Thus allows to build a UNICORE storage system providing data replication, disaster recovery, durability and elasticity. In this paper we will present the architecture and operation of a prototype called UniHadoop, which provides the integration of UNICORE and the distributed storage systems (DSS) supported by Hadoop and highlight its potential in usage scenarios.

1 Introduction

Computational power has increased significantly, typical desktop systems are now far superior to the super computers in the last years, and the current high-performance systems have reached the PetaFlop/s scale. This allowed to tackle new computational challenges in applications such as high-energy physics, nuclear physics, weather forecasting, data mining, environmental modeling, which can now be executed in 'feasible' time spans. These applications usually produce data with rapid growth and in large volumes. For example, the High Energy and Nuclear Physics (HENP) data volume will rise from hundreds of petabytes to exabytes (10^{18}) from 2012-2015 [1]. These huge chunks of data, produced from such applications, need to be stored, exchanged, visualized and analyzed for a variety of purposes. This requires specialized storage systems with high data availability, reliability and distribution.

UNICORE[1] is used as underlying Grid middleware by a number of scientific applications [2]. At the time of writing, its storage management interfaces do not

[1] **Uni**form **I**nterface to **Co**mputing **Re**sources.

H.X. Lin et al. (Eds): Euro-Par 2009 Workshops, LNCS 6043, pp. 345–352, 2010.

support distributed storage system interaction [3], but are limited to file system access. In this paper we present the design and integration of Apache Hadoop and its supported storage systems with UNICORE. The remainder of this paper is structured as follows. In Section 2 UNICORE's architecture is elaborated in detail. Section 3 gives an overview of Hadoop, its design and supported storage system. in Section 4 we present UniHadoop, i. e. the integration of UNICORE and Hadoop supported DSS. After discussing the usage scenarios, the paper ends with a brief conclusion and related work.

2 UNICORE Architecture

UNICORE is a ready-to-run, Open Source Grid middleware that enables seamless, secure and transparent access to resources. It hides unnecessary details from the users and provides single sign-on with well-established security mechanism. UNICORE offers a number of clients for job creation, submission and monitoring. In its most recent version UNICORE 6 conforms to the Open Grid Service Architecture (OGSA) [4] and several open Grid standards as mandated by the Open Grid Forum (OGF) [5]. It is a Web services based implementation using the Web Services Resource Framework (WSRF) [6]. The user can use various clients for jobs creation, submission and monitoring, both graphical and command-line oriented.

From the conceptual point of view, one can divide UNICORE in three tiers: Client, Server and Target. The user logs into the client and creates the job. The job along with user information is authenticated by the Gateway and forwarded to the Web services interfaces offered by the UNICORE/X server. These services interact with XNJS[2] which interacts with XUUDB[3] for authorization information. Finally, task is handed over to either Target System or Target System Interface (TSI) which then executes the job.

A Client layer: With growing use of UNICORE and emergence of new applications and problem domains, a set of clients is supported [7]. It helps in job definition and application integration from different domains in an easy to do manner. These are command line as well as GUIs. Java is used as development language, so it supports many platforms.UNICORE Commandline Client(UCC), Grid Programming-based Environment(GPE) Client, UNICORE Rich Client, Web Portal Client and High Level API (HILA) clients be used to communicate with UNICORE server and underlying resources.

Service layer: The core of UNICORE consists of services and components compliant with the OGSA model and is based on WSRF 1.2, SOAP [8] and WS-I standards [9]. Analyzing it from top to bottom, request first encounters with Gateway which acts as a secure entry point for any number of UNICORE sites. Client requests are authenticated and encrypted on this level using X.509 certificates [10]. Next the central job controlling entity of UNICORE is XNJS which provides a

[2] e**X**tended **N**etwork **J**ob **S**upervisor.
[3] e**X**tended **U**NICORE **U**ser **D**ata **B**ase.

set of services like job management, storage management and data transfer services. It communicates with the XUUDB for authorization of users, maps the jobs to the desired Target Systems using Incarnation Database (IDB),submits jobs and monitors its progress. It does not only support OGSA-* based open standards but also has its own proprietary interfaces named UNICORE Atomic Services (UAS). Storage Management Service (SMS) and File Transfer Service (FTS), part of UAS, exposes storage resources to users and provides different operations ranging from storage management to data transfer. Target System Registry Service (TSR) is another essential component like in any SOA system. Any client wishing to utilize UNICORE 6 must have information regarding available services and their description in a specific Grid. The detailed information is published in TSR. It works as a single point of entry for clients. A TSR is shareable between sites.

System layer: The Target System Interface (TSI) lies at bottom in UNICORE architecture. Its role is to communicate between the local operation system and XNJS. The TSI takes the concrete jobs from the XNJS and executes them on the target system as the local user determined by the XNJS. A temporary working directory named USpace is attached with each job. Every job has its own USpace which stores data related to the specific job and later user can transfer data to and from storage system (or the client) to the USpace.

3 Hadoop

The Web search engine giant Google processes and stores huge data and thus requires a storage system with special attributes for instant replies. In 2003, Google described its highly scalable, fault tolerant, dynamic distributed file system called the Google File System (GFS) [11]. This file system is designed to run on commodity machines and can respond to thousands of queries per second. Component failure was taken as the norm rather than the exception. Files are getting huge as compared to traditional files, increased computation power consistently producing more and more data which needs to be organized for future analysis. It was observed that many applications follow the "Write Once Read Many" paradigm. The GFS is proprietary software.

The Hadoop distributed file system (HDFS) is an open source implementation of the ideas and algorithms of Google's GFS, realised in Java [12]. It is used by many organizations, notably Yahoo, where it has been proven to scale up to 4000 nodes and 16PB disk size.

Apart from the HDFS, Hadoop offers an implementation of the MapReduce data processing framework, again as published in a seminal paper by Google [13].

3.1 HDFS Architecture and Supported File/Storage Systems

The Hadoop distributed file system follows a Master/Slave architecture. One node in the HDFS cluster works as NameNode, while the others serve as DataNodes. The NameNode manages all the meta data regarding filesystem like file

namespace, file to block mapping, block location information, access control version, block version numbers etc. As shown in Fig. 1, NameNode runs on a separate machine and periodically communicates with the DataNodes but this communication is least started by NameNode. It is the DataNode's responsibility to send regular heart beat messages to the NameNode.

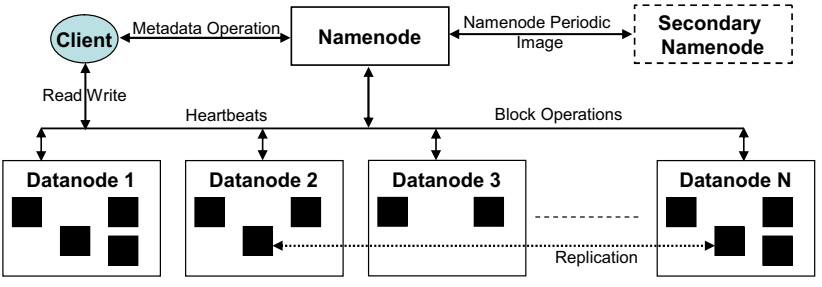

Fig. 1. HDFS Architecture showing node interdependencies

The NameNode starts up in "safe mode", no changes are possible in this mode and it remains in it as per configuration. As the NameNode starts up, each DataNode sends a heartbeat message to the NameNode. After that the NameNode asks each DataNode to send its block report. The DataNode computes the block report and returns it to the NameNode. After this the NameNode exits the safe mode and goes into "normal mode" if all conditions are fulfilled.

During the normal mode and with periodic communications NameNode gets certain pieces of information such as: if some DataNode is down, any disk failure on a DataNode, checks the Replica status of files, which DataNode stores which replica. Finally, Hadoop also supports a number of open source as well as commercial file/storage systems other than HDFS.

4 UniHadoop

UniHadoop is the integration of multiple Hadoop supported distributed storage systems with UNICORE 6. As discussed in section 2, UNICORE is a service oriented, flexible system with a number of proprietary and open standard services. The SMS manages the storage resources and initiates the data transfer with the FTS. In the UniHadoop prototype, SMS and FTS were extended to allow the integration of Hadoop with UNICORE. After the integration, user can access any of the Hadoop supported storage system without altering call interface to UNICORE.

4.1 UniHadoop Architecture

As UNICORE itself, UniHadoop is realized in Java and uses WS-RF based Web services. Fig. 2 shows the architecture of the UniHadoop prototype. An abstract

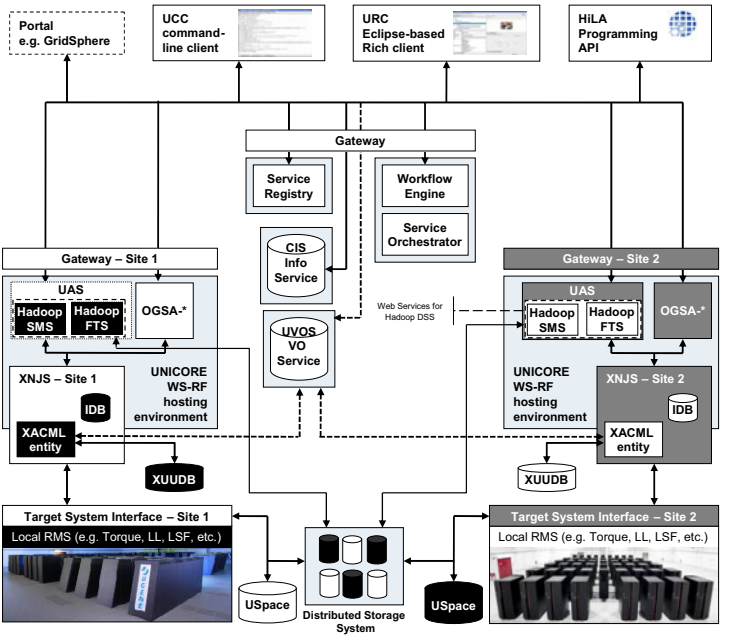

Fig. 2. UniHadoop Architecture

file system object is used for genericity. New Web services, HadoopSMS and HadoopFTS are created and deployed in the UNICORE/X component at the UNICORE service layer. With every client request originated from UNICORE clients and authenticated by the Gateway, Web service instance reads out the distributed storage system configuration files and instantiates the abstract filesystem object with the appropriate storage system. This filesystem instance holds the necessary information to communicate with the distributed storage system and directly performs operations on the DSS. Simple storage management operations are performed by HadoopSMS directly while for data transfer it initiates the HadoopFTS instance with desired data transfer protocol. This FTS service instance transfer data between DSS and USpace.

HDFS supports multi-user with its own permissions model for files and directories, a very similar one to POSIX model. UNICORE user is required to have a user account on the cluster machine with appropriate configuration settings for authorization. A user can be a single user or a group of users which is identified with the host operating system. A number of other mechanism for authentication like Kerberos, LDAP are also under consideration for future release [14].

Currently the UniHadoop prototype is being further refined and integrated into the core UNICORE server distribution.

5 Usage Scenarios

Hadoop supports a variety of back-end file systems: Hadoop Distributed File System (HDFS), CloudStore [15], Amazon Simple Storage System (Amazon S3) [16], FTP File System, Read-only HTTP and Read-only HTTPS can be used. Using UniHadoop, these can be used seamlessly in UNICORE.

The primary use case of UniHadoop is the realisation of a large, scalable storage on commodity hardware using the Hadoop HDFS.

User may need to have multiple UNICORE sites depending upon needs and resources availability. With UniHadoop integration, user can use multiple sites to access single distributed storage system. Same configuration files are only required to realized this scenario. Fig. 2 shows two sites namely site 1 and site 2 retrieve and store data on same Hadoop supported DSS. The number of sites to shares DSS may range from 1 to N.

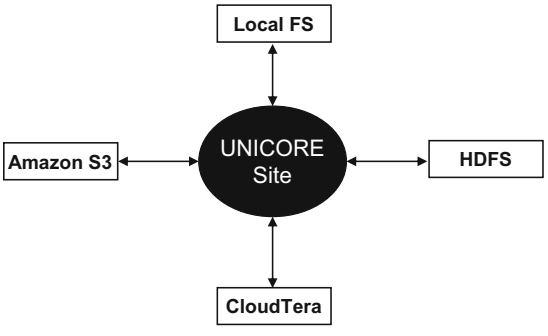

Fig. 3. Single UNICORE Site With Multiple Storage Systems

Since UniHadoop is not tied to any particular back-end, it can also be used to store data over Amazon Simple Storage System (S3) without any capacity restriction [17] and few other storage systems as shown in Fig. 3. Amazon Elastic Compute Cloud (Amazon EC2) can be used for running MapReduce jobs over data stored in Amazon S3 [18]. Data transfer among to Amazon EC2 is free which makes Amazon S3 attractive.

6 Conclusion and Outlook

In this paper, we presented UniHadoop, that represents the integration of Hadoop supported DSS with UNICORE. Now user can use the HDFS file system and some other prominent and state-of-the-art distributed storage systems as well like Amazon S3 and CloudTera with UNICORE. It enables users to administrate the previously mentioned distributed storage systems from UNICORE 6 and also to perform file transfer from these storage systems to UNICORE. A wide variety of scenarios can be realized with UniHadoop. It is possible to build

highly scalable distributed storage systems using HDFS. Multiple UNICORE sites can access the same DSS to achieve load-balancing and/or high-availabity. With multiple service instances, UNICORE sites can use multiple storage systems at the same time.

Storing data using the Hadoop storage system brings a number of advantages. Primarily one can build highly scalable, efficient and reliable storage systems on commodity hardware.

As an outlook, it will be interesting to leverage the MapReduce [13] framework available in Hadoop, using HDFS for simplified parallel data processing.

Examples for using MapReduce in Hadoop are demonstrated in Pig [19], a platform for analyzing large data sets. It uses Pig Latin [20], a very powerful language for writing jobs. Hive [21], a data warehouse infrastructure built over Hadoop, can be used with data stored in Hadoop for data summarization, ad hoc queries and analysis of large set of data. It provides a simple SQL based query language, Hive QL, for simplifying queries. Theoretically UNICORE can leverages from all the projects which Hadoop supports.

The Option to run MapReduce jobs directly on the data stored in the HDFS is very attractive, for example to index large volumes of user-generated data, in order to provide powerful search and metadata capabilities.

References

1. Newman, H.B., Ellisman, M.H., Orcutt, J.A.: Dataintensive e-science frontier research. Communications of the ACM 46, 68–77 (2003)
2. Breuer, D., Erwin, D., Mallmann, D., Menday, R., Romberg, M., Sander, V., Schuller, B., Wieder, P.: Scientific Computing with UNICORE. In: Wolf, D., Münster, G., Kremer, M. (eds.) NIC Symposium 2004, Forschungszentrum Jülich; proceedings (2004)
3. Streit, A., Erwin, D., Mallmann, D., Menday, R., Rambadt, M., Riedel, M., Romberg, M., Schuller, B., Wieder, P.: Unicore - from project results to production grids. In: Grid Computing and New Frontiers of High Performance Processing (2005)
4. Foster, I., Kesselman, C., Nick, J.M., Tuecke, S.: The physiology of the grid. In: Berman, F., Hey, G.C.F.A.J.G. (eds.) Grid Computing, vol. pages 217-249, John Wiley & Sons, Chichester (2003)
5. Riedel, M., Schuller, B., Mallmann, D., Menday, R., Streit, A., Tweddell, B., Shahbaz Memon, M., Shiraz Memon, A., Demuth, B., Lippert, T., Snelling, D., van den Berghe, S., Li, V., Drescher, M., Geiger, A., Ohme, G., Vanni, A., Cacciari, C., Lanzarini, S., Malfetti, P., Benedyczak, K., Bala, P., Ratering, R., Lukichev, A.: Web services interfaces and open standards integration into the european unicore 6 grid middleware. In: Proc. Eleventh International IEEE EDOC Conference Workshop EDOC '07, October 15–16, pp. 57–60 (2007)
6. Wsrf-technical committee, `http://www.oasisopen.org/committees/wsrf/`
7. Unicore clients, `http://www.unicore.eu/unicore/architecture/client-layer.php`. (accessed on 23.05.09)
8. Soap specification, `http://www.w3.org/TR/soap/`
9. Web services interoperability (ws-i), `http://www.ws-i.org`

10. Welch, V., Foster, I., Kesselman, C., Mulmo, O., Pearlman, L., Gawor, J., Meder, S., Siebenlist, F.: X.509 proxy certificates for dynamic delegation (2004)
11. Howard, S.G., Gobioff, H., tak Leung, S.: The google file system (2003)
12. Apache software foundation. hadoop distributed file system, `http://hadoop.apache.org/core/`
13. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters (2008)
14. Hadoop pemissions, `http://hadoop.apache.org/common/docs/` (accessed on 26-05-09)
15. Cloudstore, `http://kosmosfs.sourceforge.net/`
16. Ripeanu, M., Iamnitchi, A.: S4: A simple storage service for sciences
17. Amazon s3 with hadoop, `http://wiki.apache.org/hadoop/AmazonS3` (accessed on 26-05-09)
18. Hadoop with amazon ec2, `http://wiki.apache.org/hadoop/AmazonEC2` (accessed on 26-05-09)
19. Apache software foundation. pig software, `http://hadoop.apache.org/pig/` (accessed on 26-05-09)
20. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig latin: a not-so-foreign language for data processing (2008)
21. Apache software foundation. hive, `http://wiki.apache.org/hadoop/Hive` (accessed on 26-05-09)