Fusion Engineering and Design xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

# Utilizing cloud storage architecture for long-pulse fusion experiment data storage

Ming Zhang[a,b], Qiang Liu[a,b], Wei Zheng[a,b,*], Kuanhong Wan[a,b], Feiran Hu[a,b], Kexun Yu[a,b]

[a] State Key Laboratory of Advanced Electromagnetic Engineering and Technology, Wuhan, Hubei, China
[b] School of Electrical and Electronic Engineering, Huazhong University of Science and Technology, Wuhan, Hubei, China

ARTICLE INFO

*Article history:*
Received 30 May 2015
Received in revised form 15 January 2016
Accepted 15 February 2016
Available online xxx

*Keywords:*
MongoDB
Long-pulse
LVS
Data archiving
Scientific database

ABSTRACT

Scientific data storage plays a significant role in research facility. The explosion of data in recent years was always going to make data access, acquiring and management more difficult especially in fusion research field. For future long-pulse experiment like ITER, the extremely large data will be generated continuously for a long time, putting much pressure on both the write performance and the scalability. And traditional database has some defects such as inconvenience of management, hard to scale architecture. Hence a new data storage system is very essential.

J-TEXTDB is a data storage and management system based on an application cluster and a storage cluster. J-TEXTDB is designed for big data storage and access, aiming at improving read–write speed, optimizing data system structure. The application cluster of J-TEXTDB is used to provide data manage functions and handles data read and write operations from the users. The storage cluster is used to provide the storage services. Both clusters are composed with general servers. By simply adding server to the cluster can improve the read–write performance, the storage space and redundancy, making whole data system highly scalable and available.

In this paper, we propose a data system architecture and data model to manage data more efficient. Benchmarks of J-TEXTDB performance including read and write operations are given.

## 1. Introduction

With the development of data acquisition system and plasma control technology, more and more tokamaks are undertaking long-pulse experiments, leading data grows rapidly. Taking ASDEX Upgrade as an example, the acquired amount of data per shot increased from 4 GiB to 40 GiB in seven years [1]. In addition, data in Large Helical Device (LHD) increase 10 times in approximately 5 years [2]. Furthermore, the rate of data to be recorded at ITER is at times anticipated to 50GB/S at peak times [3]. However, traditional storage technology equipped with one server has its limit of read–write speed. Although Storage Area Network (SAN) offers high-speed block-level access, it's so expensive and complicated that only big enterprise can afford it. Therefore, we have to ask cluster and distributed system for assistance.

The purpose of a distributed file system (DFS) is to allow users of physically distributed computers to share data and storage resources by using a common file system. DFS offers user a high scalable and high-performance solution in general.

Research results about the distributed file system have been given in some laboratories and the achievements are relatively advanced. ASDEX Upgrade used AFS-OSD which is featuring a parallel storage concept and modified from AFS (Andrew File System) to store big files more efficient [1].

COMPASS and LHD have tested GlusterFS respectively and got some good results in read–write performance. COMPASS utilized “Distributed-Replicate” $6 \times 2$ type include six servers and six custom-made computers and the maximum of reading speed approaches a value 300MiB/s for 6 clients [4]. LHD also adopted GlusterFS as their 5th generation storage. Their test results indicated that SSD GlusterFS volumes can provide 300–900MB/s write, 500–1900MB/s read performances [5].

However, DFS is inflexible compared with database cluster. DFS lacks a uniform data model, thus it is not easy to manage the data especially when it grows too fast. When users want to search the

* Corresponding author at: State Key Laboratory of Advanced Electromagnetic Engineering and Technology, Wuhan, Hubei, China. Fax: +86 2787793060.
*E-mail addresses:* zhenghaku@gmail.com, zhengwei@hust.edu.cn (W. Zheng).

http://dx.doi.org/10.1016/j.fusengdes.2016.02.050
0920-3796/© 2016 Elsevier B.V. All rights reserved.

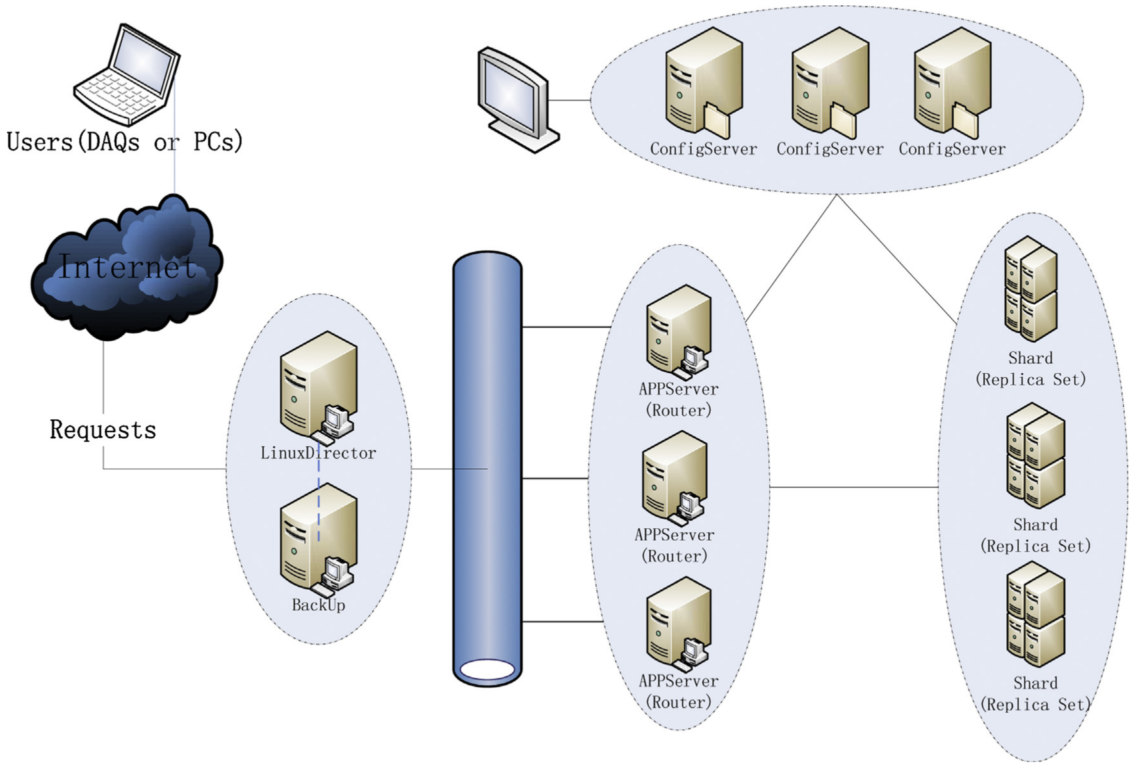

**Fig. 1.** Archtecture of J-TEXTDB.

data of a signal, it usually provides a block of data rather than the exact data we want.

Recently, a new technology about storage, named NoSQL (Not only SQL), has been used widely in Internet company such as Facebook and Twitter. NoSQL databases are increasingly applied in big data and real-time web applications, which are similar with what fusion research needed. Compared with DFS, NoSQL is more flexible and provides many functions including disaster recovery, security and scale horizontally. Hence, J-TEXTDB adopts MongoDB [6] to archive massive sized data. MongoDB is one of NoSQL databases and it is also an open-source, document-oriented database designed for ease of development and scaling.

This paper focuses on several issues below which are much vital in fusion research in our opinion.

- High availability and high scalability.
- Read–write speed must meet the demands of thousands DAQ cards.
- High concurrency of requests when worldwide researchers want to query data.
- Redundancy and failover.
- Load balance and security.

## 2. Architecture

The architecture of J-TEXTDB is made up by two parts. The first one is application cluster, it is oriented to users and plays a load balancer role in the meantime. Application cluster is composed of Linux Directors and APPServers. The second one is storage cluster, it is responsible for all the data transaction. Storage cluster is composed of ConfigServers and Shards. The architecture is depicted in Fig. 1.

### 2.1. Application cluster

Application cluster is a group of computers running different processes or services. Major functions of Application cluster are as follows:

1. Linux Director receives users requests include general users and DAQs requests, and then dispatches them to an idle APP Server with Least Connections Algorithm [7,8].
2. APP Server receives request and analyzes them to get a series of atomic operations. App Server sends operations to storage cluster, waiting for the response of it.
3. Storage cluster deals with the operations and sends the results to App Server.
4. App Server integrates the results and returns to the user according to users' demand.

The architecture of the Application cluster is fully transparent to end users, and the users interact as if it were a single high-performance virtual server. When the Director accepts request packets, it routes the packets to the chosen servers and the response packets can be sent to client directly rather than back to Director firstly then return client. In addition, J-TEXTDB adopts Backup server and Keepalived solution to handle director failover and provides a strong and robust health checking for LVS clusters. EAST has also used LVS as load balancer between users and Publish servers to lighten loads [9].

Apart from supporting WebScope of J-TEXT [10], another main function of App Server is to run a lightweight process named Mongos.

Mongos is a router for the storage cluster. In other words, if Linux Director is the first door between users and Application cluster, Mongos is the second gate between Application cluster and Storage cluster. Mongos processes and targets operations to storage cluster and then returns results to the App Server.