Experience of a low-maintenance distributed data management system

Wataru Takase¹, Yoshimi Matsumoto¹, Adil Hasan², Francesca Di Lodovico³, Yoshiyuki Watase¹ and Takashi Sasaki¹

¹ High Energy Accelerator Research Organization (KEK), Japan
² University of Liverpool, UK
³ Queen Mary, University of London, UK

E-mail: wataru.takase@kek.jp

Abstract. In this paper we report on the setup, deployment and operation of a low-maintenance, policy-driven distributed data management system for scientific data based on the integrated Rule Oriented Data System (iRODS). The system is located at KEK, Tsukuba, Japan with a satellite system at QMUL, London, UK. The system has been running stably in production for more than two years with minimal management overhead. The management tool that was developed to support the production system is also described. In addition we describe a simple XOR-based approach to file backup that reduces the amount of storage space consumed. In situations of large data volumes this approach can be of great benefit.

1. Introduction

In order to perform precision tests of theoretical models High Energy Physics experiments need to produce very large data samples. These data samples require considerable computing power and storage space. These requirements far exceed the resources available at any one institution and have spurred the development of distributed computing and storage services. The integrated Rule Oriented Data System (iRODS) is a data grid software that provides distributed data management system. iRODS manages multiple servers in multiple locations capable of providing a large pool of storage.

The Tokai to Kamioka (T2K) long-baseline neutrino oscillation experiment based in Tokai and the data quality group need to monitor the quality of the data recorded by the detector to ensure the most precise measurements of the theoretical model quantities can be measured. The group is highly distributed and needs to provide rapid feedback to the detector. iRODS provides a low-threshold system for storage and access. The iRODS system at KEK has been running stably in production for more than two years. The system has been federated with an iRODS system at Queen Mary, University of London (QMUL) to allow sharing data for data analysis and backup with low-maintenance cost.

This paper shows the usage of iRODS in KEK and also describes developed management tool to visualizing the operational data. Additionally we provide the simple XOR-based backup using iRODS to resist any two servers fail with less storage overhead compared with full-file replication.

This paper follows with a description of the iRODS data management system in Section 2 and Section 3 describes usage of iRODS in KEK. Visualization tool for iRODS is described in
Table 1. KEK iRODS environment

| iRODS | Linux Distribution | DBMS           | Storage Resource          |
|-------|--------------------|----------------|---------------------------|
| 2.5 → 3.2 | RHEL 5.6       | PostgreSQL 9.1.1 | Disk system, HPSS         |

Section 4 and simple XOR-based backup method is provided in Section 5. Finally we summarize in Section 6.

2. iRODS overview

iRODS is a data grid software system developed by the Data Intensive Cyber Environments research group as the successor to Storage Resource Broker (SRB)[1]. iRODS provides a logical file system connecting geographically distributed physical devices. In iRODS a user interacts with the system through a logical name-space that insulate them from having to know the physical location of the data [2]. iRODS maintains the mapping between the logical namespace and physical location in a database (the iCAT) that serves as a metadata catalog.

A particular feature of iRODS is the **Rule Engine**. The Rule Engine allows policies to be enforced on the data [3]. The policies are encoded as rules which are interpreted by the rule engine and executed whenever the event that they are defined for occurs. Rules can be written for many different types of policies (such as data replication, checksumming, authorization etc). Each iRODS rule consists of a series of steps, each performed by an iRODS micro-service written in the C language. Rules can be invoked automatically in response to certain conditions or triggers and can also be invoked by command line.

An iRODS system (or zone) consists of an iCAT, an iRODS server that interfaces to the iCAT and zero or more iRODS storage servers. It is possible to federate iRODS zones allowing data to be copied from one iRODS zone to another and for data stored in one iRODS zone to be accessed from another zone.

3. iRODS usage at KEK

The KEK iRODS serves two communities: the T2K experiment [4] and the Materials and Live Science Facility (MLF) [5] both based at J-PARC (Japan Proton Accelerator Research Complex). For both communities the iRODS systems have been running for more than two years. The MLF group uses iRODS primarily as a back archive providing a more intuitive interface to the HPSS (High Performance Storage System) [6] tape system. The T2K group’s use of iRODS is more dynamic: as a cache for storing and accessing data and also as a long-term archive.

The iRODS servers are built on Red Hat Enterprise Linux (RHEL) distributions and use a PostgreSQL database system as the metadata catalogue. The storage resources consist of a disk cache and an HPSS are available for storage resources. The disk cache is used for fast access and the HPSS resource is used for long-term data storage. Table 1 shows information on the iRODS system.

The KEK iRODS system was initially setup with iRODS version 2.5 and was upgraded to 3.2 during the summer of 2013. The upgrade required significant testing as the iRODS rule syntax changed between version 2.5 and 3.2 [1].

Client side access to iRODS can be made from the command-line using the `icommands` or through a web interface (`iRODS Browser`).

The requirements for data access from the T2K experiment were: (1) access from any collaborating site; (2) timely access to recorded data; (3) low barrier to use; (4) minimal management overhead. The iRODS setup at KEK met these requirements [7] by keeping things
as simple as possible. Figure 1 shows the KEK iRODS system (KEK-T2K) setup. Cron jobs were run at the data production site that registered data in the KEK iRODS system. The data was then packaged (due to the small size of the data quality files) and archived in HPSS by automatic replication rules and then purged off the cache disk. The data packaging replicated data to a backup cache resource until the volume of data reached 1GB when it was tarred and put onto the HPSS resource. The data were regularly checksummed. Users were able to access the collected data as soon as they appeared in iRODS by downloading the data from the system. Collaborators in: Japan, UK, France, Spain and Canada regularly accessed the data without problems. Figure 2 shows the rate of storage of data quality data for the T2K experiment. The flat regions correspond to periods of no data taking.

The KEK iRODS system has been federated with a QMUL iRODS system to allow data to be replicated off-site. Analysis data are stored in the QMUL iRODS and replicated to KEK to also provide off-site backup of the data. In addition, federation allows users to select the iRODS system closest to them to allow more efficient data download.
4. SCALA: Visualization tool
The iRODS system collects a lot of useful information that can be used to understand the performance of the system and to plan further optimization as well as for debugging problems. Visualization of this information makes it possible to grasp an overview of the system easily, however, iRODS lacked such an interface for usage statistics and also for debugging problems.

The Statistical Charts And Log Analyzer (SCALA) is a web interface to visualize operational data for iRODS [8] to help to grasp an overview of the iRODS system status and debugging easily. SCALA provides two web pages, Statistical Charts page and Log Analyzer page. The Statistical Charts page displays iRODS operational data as charts. The Log Analyzer page provides an error debugging tool displaying error chart with error messages.

Figure 3 shows the SCALA workflow. SCALA consists of three processes: parse, summarize, and display. iRODS outputs resource usage and system log information daily. The necessary information for SCALA is extracted from the output in the parse process and is stored in a database. The summarize process sums up the extracted information grouped by date, user and storage resource. The summarize process reduces the time to draw the chart by the display process. When a user requests to see the SCALA web page, the display process handles the request and displays charts.

Figure 4 shows the Statistical Charts page. The page displays iRODS daily operational data, data volume, number of files, number of accesses, and number of errors. User can filter displayed content by changing select menus such user menu, resource menu, and data span menu. The page helps to grasp iRODS operational status overview. Figure 5 shows the Log Analyzer page. The page displays the iRODS daily errors as a bar chart with error messages. User can see detailed errors each day by clicking each bar and see log files containing the errors at selected date. The page helps iRODS administrators to debug iRODS errors easily.

SCALA has been installed in the KEK production iRODS system and allows the iRODS administrator in KEK to check operational data easily and detect and debug errors easier. Furthermore, SCALA shows that the summarize phase reduces the time to draw charts of KEK-T2K. Without the summarized phase, drawing charts takes about two minutes whereas using the summarized tables takes less than a second.

5. iRODS XOR-based backup
Reliable data management requires data replication to guard against data corruption and loss. However, data replication has a significant overhead: storage space needs to be reserved for the replicas. The more replicas required the less storage space is available for unique data. Erasure codes [9, 10, 11, 12, 13] reduce storage overhead comparing with full-file replication with same robustness [14, 15]. They split input file into $k$ data blocks and encode, then output $n$ the coded blocks. Reed-Solomon code [16] is one of erasure codes, however, it requires arithmetic
operations over a Galois field. Erasure codes require a non-trivial amount of computational time, a code requiring only exclusive OR (XOR) operations requires less computational cost and provides comparable space-savings.

Our code needs only XOR operations for encoding and decoding capable of resisting two disks/servers failure. We describe the case of four servers where the configuration is resilient to...
Table 2. Data Blocks and Redundant Blocks Locations

| Server | Data block | Redundant block |
|--------|------------|----------------|
| Server1 | D1         | R1 = D2 ⊕ D4   |
| Server2 | D2         | R2 = D3 ⊕ D4   |
| Server3 | D3         | R3 = D1 ⊕ D2   |
| Server4 | D4         | R4 = D1 ⊕ D3   |

* ⊕ is XOR operation

up to two servers failing as this is the most realistic scenario. The data file is split into four data blocks, $D_1$, $D_2$, $D_3$, and $D_4$. Four redundant blocks, $R_1$, $R_2$, $R_3$, and $R_4$, are then created using an XOR of data blocks. Data blocks and redundant blocks are distributed to storage systems: Server1, Server2, Server3, and Server4. Table 2 shows how to create redundant blocks and each block location. The full file can be reconstructed from the blocks on any set of two servers. For example, if Server1 and Server2 fail, data block $D_1$ can be reconstructed using $D_3$ and $R_4$, and data block $D_2$ can be reconstructed using $D_1$ and $R_3$. The full file can then be reconstructed from the data blocks.

Full-file replication needs three copies to resist any two servers fail. In contrast, our code needs four additional blocks to provide same robustness above so the necessary storage space only $(4+4)/4$. Our procedure saves the 1/3 of the total space compared with full-file replication. When one talks about Petabytes or Exabytes of data this can be quite significant. Of course, there is no free-lunch and there is some additional computational overhead in encoding/decoding in order to repair the broken blocks. However, the computational overhead can work out to be much cheaper than the cost of the additional storage.

Our XOR coding scheme-based service was implemented as iRODS microservices and rules. The system was implemented in a federated environment consisting of four iRODS systems each with one server: Server1, Server2, Server3, and Server4. Figure 6 shows the coding procedure. The data file is first split into four data blocks. Then four redundant blocks are created from the XOR of the data blocks. The MD5 checksum value for each block (coded and data) is computed and stored as metadata in iRODS. Each data block together with a coded block is then distributed to one of the federated iRODS servers (see Table 2). A metafile is created and copied to other servers. Finally, after distribution, all block locations and original data size are added to the metafile as user-defined key-value metadata.

6. Summary

The KEK iRODS system has been running stably in production for over two years. The system provides the researcher-friendly data access environment. To date more than 10 Terabytes of T2K data quality data has been stored. The system is also federated with QMUL and replicates data to QMUL and from QMUL. iRODS lacks an interface which visualizes operational data and support debugging. SCALA was developed for fulfilling the demand. It has been installed on the KEK iRODS allowing the administrator easy access to operational and debugging information.

We also provide simple XOR-based backup using iRODS. The backup can be incorporated to iRODS using iRODS rules and microservices. The backup guarantees 100% recovery when up to two servers fail. It reduces storage overhead compared with full file replication with same robustness.
Figure 6. iRODS XOR-based backup

Acknowledgments
We thank the generous help of the iRODS developers: Mike Wan and Wayne Schroeder. This work was also partially funded by the European Research Council, Grant agreement no. 207282-T2KQMUL.

References
[1] irods - the integrated rule-oriented data system http://www.irods.org/
[2] Hedges M, Hasan A and Blanke T 2007 Curation and preservation of research data in an irods data grid e-Science and Grid Computing, IEEE International Conference on pp 457 – 464
[3] Rajasekar A, Wan M, Moore R and Schroeder W 2006 A prototype rule-based distributed data management system HPDC workshop on Next Generation Distributed Data Management
[4] Itow Y et al. 2001 The jhf-kamioka neutrino project KEK Report
[5] Materials and life science experimental facility http://j-parc.jp/MatLife/en/
[6] Hpss - high performance storage systems http://www.hpss-collaboration.org/
[7] Lodovico F D, Hasan A, Iida Y and Sasaki T 2011 First irods experience in a neutrino experiment iRODS User Group Meeting
[8] Scala - statistical charts and log analyzer http://tgwww.kek.jp/scala/
[9] Wang J, Gong W, Varman P and Xie C 2012 Reducing storage overhead with small write bottleneck avoiding in cloud raid system Proc. of Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on pp 174 – 183
[10] Duminuco A and Biersack E 2009 A practical study of regenerating codes for peer-to-peer backup systems Proc. of Distributed Computing Systems, 2009. ICDCS ’09. 29th IEEE International Conference on pp 376 –384
[11] Luo J, Xu L and Plank J S 2009 An efficient xor-scheduling algorithm for erasure codes encoding Proc. of Dependable Systems Networks, 2009. DSN ’09. IEEE/IFIP International Conference on pp 504 –513
[12] Greenan K M, Li X and Wylie J J 2010 Flat xor-based erasure codes in storage systems: Constructions, efficient recovery, and tradeoffs Proc. of Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on pp 1 –14
[13] Aguilera M K, Janakiraman R and Xu L 2005 Using erasure codes efficiently for storage in a distributed system Proc. of Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on pp 336 –345
[14] Weatherspoon H and Kubiatowicz J D 2002 Erasure coding vs. replication: A quantitative comparison Proc. of the First International Workshop on Peer-to-Peer Systems (IPTPS)
[15] Rodrigues R and Liskov B 2005 High availability in dhts: Erasure coding vs. replication Proc. of the 4th International Workshop on Peer-to-Peer Systems (IPTPS) (Ithaca, New York)
[16] Reed L S and Solomon G 1960 Society for Industrial and Applied Mathematics 8 300–304