Project no. 032691

KnowARC

Grid-enabled Know-how Sharing Technology Based on ARC Services and Open Standards

Specific Targeted Research Project

Information Society Technologies

D2.1-1 SELF-HEALING FLEXIBLE GRID STORAGE FOR ARC FIRST PROTOTYPE

Due date of deliverable: July 31, 2008 ***Actual submission date:*** July 31,2008

Start date of project: June 1, 2006 ***Duration:*** 39 months

Organisation name of lead contractor: NIIFI

Revision: draft

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

CONTENTS

1 Introduction

This is the first out of two KnowARC deliverables, which are called together 'Self-healing flexible Grid storage for ARC, first prototype and final release'. This deliverable describes the first prototype. The deliverable document is just a short overview of the prototype software; there is a technical documentation[1] describing the interfaces as well as the internal mechanism of the services, and also an archive file containing the source code[2] of the storage-related services.

2 Overview

One of the goals of KnowARC is to provide high-level, self-healing data storage capability along with an easy-to-use user interface. To achieve this goal we wanted to avoid single points of failures and bottlenecks and to provide a redundantly planned grid storage system. The system shows a global namespace of files and collections (which could contain files and sub-collections) to the user. This way it can be used almost as simple as local file systems. The files are replicated and the broken replicas are repaired automatically. The files are referred by Logical Names (LNs), which are paths in the global namespace.

This capability is provided by four main services called A-Hash, Librarian, Shepherd and Bartender. The typical system architecture consists of several storage nodes having storage space and bandwidth to share, one Shepherd service for each storage node (not necessarily installed on the node itself), and other machines having CPU time to share hosting the other services. In this prototype release the A-Hash service is still centralized, but in the final release it will be distributed. It is possible to install any number of Librarian and Bartender services to support high-availability and load-balancing.

The A-Hash service is a database storing objects containing key-value pairs. The Librarian service uses the A-Hash as its database.

The Librarian service manages the hierarchy and metadata of files and collections, handling Logical Names and monitoring the Shepherd services.

The Shepherd service manages a particular storage node and provides uniform interface for storing and accessing file replicas. On a storage node there must be at least one transfer service (e.g. HTTP(S), FTP(S), GridFTP, ByteIO[3], etc.) which is capable to perform the actual file transfer. Transfer services can either be KnowARC or third-party services. For each kind of transfer service we need a module for the Shepherd service which makes the Shepherd service capable of communicating with the transfer service, e.g. to initiate file uploads, downloads and removal.

The Bartender service provides a high-level interface for the storage system to the users. The users can create and remove collections (directories), create, get and remove files, move files and collections within the namespace using Logical Names. The Bartender communicates with the Librarian and Shepherd services to accomplish

[1] http://www.knowarc.eu/download/D2.1-1_documentation.pdf

[2] http://www.knowarc.eu/download/D2.1-1_code.zip

[3] https://forge.gridforum.org/sf/projects/byteio-wg

the user's requests. The actual file data does not go through the Bartender; file transfers are directly performed between the storage nodes and the clients.

There is a client script included in this release, which can be used to access the storage system. It can be used on any system that has Python interpreter installed. It has no other dependency.

This prototype has no security-related solutions. The process of authentication and authorization is mainly designed, but not implemented yet.

3 Prototype code and testing

The services, which have SOAP interface, are written using the ARC1 HED web-service based hosting environment[4]. The services are written in Python. The HED hosting environment is written in C++, but due to its language binding feature service developers can write services in other languages, like Python and Java.

The source code included is not usable on its own. It contains all the source code related to the storage services, but it needs the HED hosting environment to run. The archive file contains a README describing install and usage instructions, but successful compilation is possible only from the whole ARC1 source tree.

The prototype has only been tested in a single machine environment where all the services were running on one host. It was tested using a simple HED-based HTTP server as the transfer service. Tests with multiple machines on LAN and WAN are planned but have not started yet.

4 Files accompanying this deliverable

This archive file includes the source code of the four main services:

```
http://www.knowarc.eu/download/D2.1-1_code.zip
```

There is a detailed technical documentation here:

```
http://www.knowarc.eu/download/D2.1-1_documentation.pdf
```

[4] The ARC container - https://www.knowarc.eu/documents/Knowarc_D1.2-2_07.pdf