Contribution ID: **167** Type: **Demonstration**

Using Spare Space in Worker Nodes and Desktop Disks to Build a gLite Storage Element

Wednesday 14 April 2010 17:10 (10 minutes)

We present how idle disk space available at desktops and worker nodes in computing elements can be used to implement the file system back-end of a gLite storage element. We developed the BeeFS to federate the distributed disks. Like some special-purpose file systems (eg. GFS), it uses a hybrid architecture that follows a client-server approach for serving metadata and manage file replicas, and a peer-to-peer one for serving data. This allows BeeFS to aggregate the spare space of disks to build a single logical volume on top of which a general purpose POSIX-compliant file system is implemented.

Detailed analysis

A BeeFS installation consists of a single queen-bee server that handles naming, metadata and replica management operations and a number of honeycomb servers that store the actual files. The queen-bee and the honeycombs provide service to many honeybee clients. The queen-bee server is deployed in a dedicated machine. It is responsible for providing a global file namespace with location-transparent access for files, access control, resource discovery and placement coordination services. On the other hand, it is not involved in data storage at all. Honeybee clients contact it in order to obtain the location of the honeycomb servers that store the files. After that, they fetch/send data directly from/to the appropriate honeycomb server. The role of the honeycomb servers is to collaboratively store files, providing basic read and write primitives. Honeycomb servers are conceived to be deployed over a set of desktop machines or a nodes in a cluster interconnected by a LAN. This hybrid architecture mixes aspects of client-server and peer-to-peer systems in a fashion that simplifies the design and facilitates the administration of the system.

Conclusions and Future Work

Our implementation has been checked against the version of Pawel Jakub Dawidek's POSIX file system test suite maintained by Tuxera and has successfully executed all the 3,061 tests that comprise the suite, giving us confidence that it is, indeed, fully POSIX-compliant. We are currently deploying a storage element in the EELA-2 infrastructure backed up by a BeeFS system that was able to harness more than 1.5Tbyte of spare disk in the desktops of our lab. Our future work includes the execution of MapReduce-like applications exploring the distributed implementation of this storage element.

Impact

We have developed an implementation of BeeFS that runs on Linux machines. BeeFS exposes the POSIX API for file system service; this is especially important for reasons of applications compatibility, allowing a standard gLite SE to use it. Programming a POSIX file system on Linux, usually requires coding at the VFS (Virtual File System) level. Instead, we have implemented BeeFS at the user level using the Java programming language. The coupling between the user level application and the Linux kernel file system modules was done via FUSE. In order to measure the file system performance in a wide range of typical operations, we ran the well-known Andrew benchmark. This benchmark emulates a software development workload. In average,

BeeFS outperforms NFS execution time in 74% for write operations and 30% for read operations in the best case. In the worst case, BeeFS results in a 56% improvement in write operations and 20% for read operations when compared with NFS. A storage element that uses such a distributed file system as the storage back-end is particularly suited to executed MapReduce applications in an efficient way, provided that appropriate scheduling mechanisms are in place.

Keywords

distributed file system; hybrid file system; POSIX; MapReduce applications

URL for further information

http://redmine.lsd.ufcg.edu.br/projects/show/ddg

Justification for delivering demo and/or technical requirements (for demos)

Running a demo showing how to efficiently execute MapReduce applications taking advantage of a storage element that uses the BeeFS as its back-end file system should be of interest to many EGEE users.

Author: Prof. BRASILEIRO, Francisco (UFCG)

Co-authors: SOARES, Alexandro (UFCG); SILVA, Jonhnny Wesley (UFCG); PEREIRA, Thiago Emmanuel (UFCG)

Presenter: Prof. BRASILEIRO, Francisco (UFCG)

Session Classification: Demo Session 2

Track Classification: Software services exploiting and/or extending grid middleware (gLite, ARC, UNICORE etc)