Thesis proposal: Enhancing the distributed file storage system for SORCER SILENUS - Sorcer Integrated Local Enhanced New User's Storage

Max Berger

January 28, 2006

Based on the existing solution in Sorcer and Globus, and my experience this semester in understanding and working with the Sorcer Framework, I would like to work on a new file storage system for the Sorcer environment as a thesis.

Requirements elicitation

Problem statement

The Sorcer environment currently provides a file storage system that can be used from Sorcer clients. This file storage system, however has some drawbacks.

The current filesystem in Sorcer is not very distributed. Although it allows access to resources from different clients, these have to have physical access to the file system. We want a new, robust file system that is accessible from within Sorcer, easy to set up, secure to failures, fully distributed, ...

Timeline

In a sense, this project has already started since I have done some analysis of existing solutions. This project should start in September 2004 and should be completed by December 2006.

Functional requirements

- Upload files from classical and Sorcer applications
- Download files from classical and Sorcer applications
- Modify files online from classical and Sorcer applications
- Selecting files for hoarding from a special application

Nonfunctional requirements

Decentralization

- Each node should be able to work on its own.
- There should be no other connection requirements other than being able to connect to other Sorcer Services.

Transparencies

- Location transparent: it shouldn't matter where the file is stored
- Access transparent: All elements in the filestore should be accessible from classical, non-Sorcer programs.
- Replication transparent: There should be no difference on what replication the user works
- Failure transparent: The system should still work even if a significant number of hosts is down.
- Read concurrency transparent: Multiple users should be able to read the same file at the same time
- Write concurrency transparent: Multiple users should be able to write to same file at the same time
- Migration transparent: The system or the user should be able to migrate the physical presence of a file without interrupting any work.

Security

- User Security: There should a secure way to identify users, and to make sure that no-one can read/write files without permissions, not even the superuser on a system.
- System Security: Each system that wants to become part of a particular filestore must somehow identify itself so that no systems can just come up and steal data.

Compability

- The applications and its parts should be implemented in java utilizing only the Sorcer framework.
- All programs written for the old filestore interface should work without modification.

User friendliness

- The application should be easy to install
- The application should require as little actual user interaction as possible.

Automaticity

The software should be itself

- keep important files on multiple machines.
- migrate files to the machines where they are mostly used.

Completeness of implementation

Each of the Requirements will be given a priority. The implementation should focus on the most important features first, then go to the lesser important features. If the time runs out it should be a complete working solution.