Project plan

Final (*CVS Revision* : 1.22)

April 28, 2001

RTD-Log

Dan Patrick Doepel
Matti Johansson
Marko Moisio
Juha Saarelainen
Minna Vasankari

University of Helsinki
Department of Computer Science
P.O. Box 26 (Teollisuuskatu 23)
FIN-00014 UNIVERSITY OF HELSINKI

Contents

1 General description

RTD-Log project is working with the real-time database pilot, called RODAIN. RODAIN is a distributed real-time main-memory database, where each database node consists of two database units: Primary and Mirror. Both database units keep their own (identical) copy of the database. Primary Unit is responsible of executing all transactions, and it delivers all log information to the mirror. Mirror Unit uses the log to update its own local database and stores then this log information to the permanent storage on the disk-based Secondary Storage Subsystem (SSS).

When either of the database units crash, the remaining unit takes the role of Transient Unit. Transient Unit then executes all transactions and writes logs directly to SSS while the other unit is recovering. Recovering Unit uses log information on SSS to build up it's own database. When the recovering unit is ready, it takes the role of Mirror Unit and Transient Unit becomes a new Primary Unit.

Currently the RODAIN system logging is based on incremental logs. This means that each new log entry is appended to the log file on SSS. Although simple and easy to use in recovery, its performance is somewhat poor and it may cause relatively long periods of total unavailability (neither of the two database units are executing transactions) if both database units crash simultaneously.

The goal of the RTD-Log project is to design and implement a more sophisticated way to store log information to the Secondary Storage Subsystem so that the log mechanism will enable fast recovery of crashed unit.

2 Objective

In order to minimize the time needed in recovery and the total unavailability time of the node, logs must be somehow indexed so that the latest value of any given object can be accessed directly from the log. This index must also be periodically stored to the SSS, so that it may be re-read after crash. In addition to logs, it may be useful to periodically store complete database images to SSS.

The design and implementation are carried out with special emphasis on modularity. This is to enable easy modification of the system later on so that different design solutions and their effect on recovery performance can be evaluated.

3 Environment

Both database nodes run on ChorusOS but the Secondary Storage Subsystem runs on Linux. The implementation of the new log system requires coding both for database nodes and for SSS. But because the RTD-Log project developers has no experince with ChorusOS, all coding will be done on Linux environment using the C-language. Porting from Linux to Chorus should be quite straightforward.

4 Equipments

Project will use CVS as version control system. Documents, source codes and all other relevent files will be kept there. Documentation will be done with LaTeX.

5 Project organization

The customer on behalf of the RTD-Pilot -project is Tiina Niklander. Mika Karlstedt is the instructor of the RTD-Log project group and the projecmaster is Raine Kauppinen.

5.1 Project developers and their areas of responsibility

Matti Johansson	project manager
Patrick Doepel	coding
Marko Moisio	coding
Juha Saarelainen	coding
Minna Vasankari	documentation & testing

6 Outline of project schedule

The project started 16.1.2001 in a gathering meeting. The duration of the project was chosen to be 12 weeks but it was later extended to be 16 weeks. The deadline of the project is 4.5.2001. Meetings are held Tuesdays and Fridays.

The project follows the waterfall method consisting of four stages. The first stage, requirement analysis, should sort out the clients requirements for the product. The requirements will be classified as necessary or complementary. The necessary requirements are an essential part of the product, while the complementary requirements are implemented only if the projects resources allow it. Deadline for the requirement analysis is set to 20.2.2001 resulting in a specification document. A preliminary version of the specification document should be ready 13.2.2001, giving one week for further modifications and specifications of requirements.

Next stage is planning. The Planning stage determines what data structures, structure interfaces and reference methods are used. Also a general description of data and control flows is outlined along with a module hierarchy. The result of the planning stage is a design document based on the specification document. It should containd the data plan, the architecture plan, the interface plan and the module plan. We aim at finishing a preliminary version of the design document at the end of the second week of the planning stage (23.2.2001). The deadline of the planning is set to 30.3.2001, resulting in final versions of the design document. The date for techical review of the design document has been set to 9.3.2001.

When planning is done, the implementation of the product is ready to start. Estimated time for implementation is approximately 4 weeks and the deadline of implementation is set to 20.4.2001.

After (and partially during) the implementation stage, intensive testing is carried out to ensure the robustness of the product. The deadline for testing is set to 27.4.2001 resulting in a combined implementation and testing document.

During the project, testing is carried out after and during each stage. Purposeful testing is essential to the project so that errors and vulnerable parts are found in an early stage, reducing work later on.

The outline of the projects schedule is shown in Figure 1.

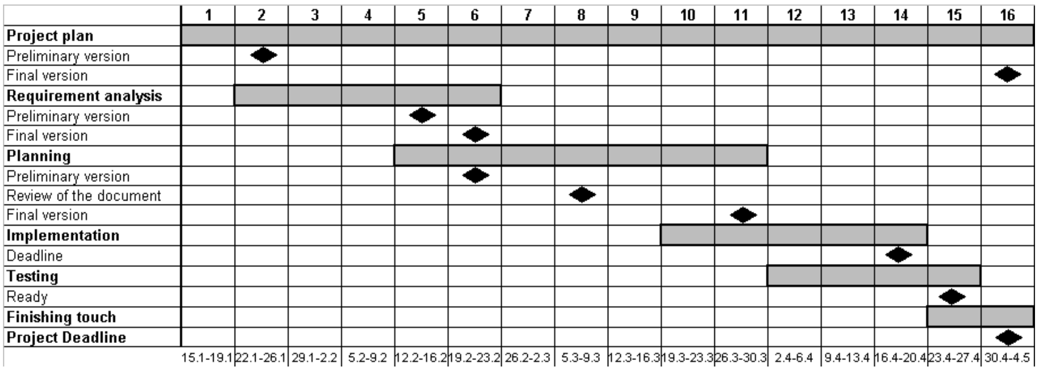

Figure 1: Outline of project schedule