Requirement specification document

Final version (*CVS Revision* : 1.25)

February 23, 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 Introduction

This section outlines why this new software is needed. Then some terminology is fixed and finally the contents of the rest of the document is outlined.

1.1 The background and purpose of the software

The RTD-Log project is going to build a new log handling subsystem for the distributed Rodain 2000 Real-Time Database System that is being studied and developed by the RTD-Pilot project. Each database node in the database system consists of two units, Primary and Mirror, and of a storage subsystem. The Primary Unit handles the clients' requests and passes the generated log records to the Mirror Unit. The Mirror Unit then updates its copy of the database and after that sends log records to Secondary Storage Subsystem where they are saved to nonvolatile storage. The log records contain full after-images of the data objects so that the log records on the disk together with a periodically written image of the whole database form a nonvolatile copy of the database in a recent state.

In case one of the units fails, the other becomes Transient and takes over processing the clients' transactions. In a normal situation transactions are allowed to commit when the Primary knows that the Mirror has received log records. But now the response time is somewhat slower because the unit has to store the logs itself. When the failed unit has recovered enough, it can start handling the logs and the Transient Unit becomes a normally functioning Primary.

The failed unit always becomes a Mirror Unit after recovering. The recovering process is made more efficient by periodically storing an image of the whole database to the disk. After reading the image and the log records stored after it the Recovering Unit has an up-to-date copy of the database in its memory. The goal is to turn the log flow through the Recovering Unit as early as possible, because the Transient Unit is constantly appending new records to the log file.

It is possible that the Transient Unit fails while the other unit is still recovering and has not yet read the full image. It should start processing clients' transactions although it does not know about all the objects in the database. To avoid this kind of situation the Mirror should maintain an index with an entry for each data object. The index points to the latest disk representation of each data object. That can be a log record or an object inside the latest image. The index must be written to disk periodically just like the image.

Hence the new log handling subsystem should in a normal situation receive log records from the Primary, update the database copy in the memory, write the log records to disk, maintain an index and write the image and the index periodically to the disk. In the recovering the goal is to build up the database in main memory as fast as possible in order to be able to handle clients' requests if necessary. For research purposes the system should be implemented with high degree of modularity so that some parts of the system can be rewritten if another design solution is desired.

1.2 Terminology

The following abbreviations and terms are used throughout the document:

SSS	Secondary Storage Subsystem, the disk-based permanent data storage
PU	Primary Unit, the active database unit that processes the clients' requests
MU	Mirror Unit, the secondary unit that maintains a copy of the main memory database and stores the log to the SSS
TU	Transient Unit, the remaining unit after either PU or MU has failed
RU	Recovering Unit, the failed unit that is recovering to become a Mirror Unit
Log mediator	The part of the already implemented RODAIN 2000 Database Prototype that transmits log records to Log system
Log system	The whole system this project is about to implement
Log manager	The whole system excluding SSS

1.3 Overview of the document

This document collects the results of the requirements analysis. The document states the client's requirements so that the outcome of the project can easily be evaluated. More importantly, the document serves as a basis for further design.

In Section 2 general description of the software is given. Section 3 covers the architecture of the system with more detail. Section 4 discusses the information flow requirements in the new log system and Section 5 lists the other requirements.

2 General description of the software

This section briefly outlines the environment, and tasks that this software is responsible of. Details are covered in sections 3, 4 and 5.

2.1 Description of the environment

The system consists of three individual servers; Primary Unit, Mirror Unit and Secondary Storage Subsystem (while logically just a disk unit, in reality the SSS is a Linux machine). All these three are connected via IP-network.

The RODAIN 2000 Database Prototype itself is running on a real-time operating system Chorus. Instead of Chorus, Linux-machines will be used in RTD-Log project, because it would be too hard to learn Chorus in such a short time. Linux is highly POSIX-compliant, and therefore it should be fairly easy to port our software to Chorus environment.

2.2 General functional description

The goal of the RTD-Log project is to create a log subsystem for RODAIN 2000 Database Prototype. Under normal circumstances this Log system is running on Mirror Unit. When either one of the database units has collapsed, the Log system will run on the remaining unit (Transient Unit). General functionality of the Log system can be divided as follows:

- Writing the log to SSS.
- Updating the database.
- Updating the index.
- Periodically writing the index and database image to SSS.
- Recovering (reading log, index and image information from SSS, and rebuilding the database).

Some of these functions are not performed, while the Log system is running on Transient Unit. More detailed description of this functionality will be given in the following section.

3 Functional description

The basic structure of the system and the modules that are going to be impelemented have been described in this section.

3.1 General architecture of the software

System consists of Primary and Mirror Unit and of Secondary Storage Subsystem. In the normal situation the Primary Unit handles the clients' requests and generates log records about the database transactions. The Primary then sends the records to the Mirror Unit that uses them to update its copy of the database. Mirror maintains an index where is kept information about the latest disk representation of every database object. After Mirror Unit has updated the index it sends the log records to SSS where they are stored to nonvolatile disk.

The tasks of every unit are outlined with more detail in the next sections.

3.1.1 Primary Unit

Primary Unit generates log records about transactions that has altered the content of the database. The log records are then sent to Mirror Unit. Mirror sends acknoledgement when it receives a record. After that Primary Unit can commit the transaction in question. When Mirror Unit has stored the log record to SSS it informs the Primary Unit so that the Primary can remove all the logs concernig the transaction in question from its buffers.

If Primary Unit notices that Mirror Unit has failed Primary switches its role to Transient Unit.

3.1.2 Mirror Unit

Mirror Unit contains a module (Log mediator) that receives log records from the Primary Unit and delivers them to Log manager in the Log system. Log manager updates the Mirror's copy of the database and the index and then sends the records to SSS. When a log record has been sent Log manager informs the Log mediator about this. Log mediator then informs the Primary Unit. From time to time Mirror Unit stores its copy of the database to SSS. It also stores regularly the index structure in order to speed up the recovering.

If Mirror Unit notices that Primary Unit has collapsed it will become a Transient Unit.

3.1.3 Transient Unit

Transient Unit handles clients' requests as Primary Unit but it sends log records directly to SSS. If the unit was previously a Mirror Unit it may also maintain the index. When the Transient Unit receives information that the other unit is recovering and can accept the log flow, it stops sending log records to SSS and starts sending them to Recovering Unit instead. At the same time it switches its role to Primary Unit.

3.1.4 Recovering Unit

When unit for some reason has failed it will recover and become a Mirror Unit. When the unit is recovering and hasn't yet become a fully operational Mirror Unit it is a Recovering Unit. The recovery has two phases. In the first phase the unit reads the newest index that is stored on SSS. In the second phase the Transient Unit switches its role to Primary Unit and turns the log flow from SSS to Recovering Unit. Then the Recovering Unit reads log records that have been saved to SSS after the latest index. Finally the Recovering Unit uses the index to fetch those data objects from the SSS that haven't yet been added to database. At the same time it also stores to SSS log records it receives from the Primary Unit. After the unit has fully recovered it starts acting as a Mirror Unit.

3.1.5 Secondary Storage Subsystem

SSS receives log records from the Mirror or from the Transient Unit and takes care of saving those records to disk. SSS maintains status information about all the relevent files. This information is needed for the Recovering Unit so that it can get the up-to-date status of SSS. Log records are written to a file in the order they are received. When Mirror Unit decides to store an image the SSS then opens a new file where the image and the log records are written.

Indexes are stored in a file of their own. When the SSS is storing an index no log records are written at the same time. Only information about successfully stored index will be saved to a log file.

4 Interfaces and information flow

Log system will be a subsystem of an existing program called RODAIN 2000 Database Prototype. That's why all the interfaces will be software interfaces.

Description about these interfaces and the information that is transferred through them is given in this chapter.

4.1 Software interfaces

In Figure 1 the interfaces are represented with arrows. There are actually two different interface types. There is an interface to Log mediator and to the database.

The Log mediator in the Mirror Unit receives log records from Primary Unit and delivers them to Log system. In Transient Unit the Log mediator delivers log records that have been generated in that unit (interface 1).

In both cases the Log manager informs Log mediator about each log record it has written to SSS. Log mediator needs that information so it can be sure that each log record has been written permanently to disk (interface 3).

In the Mirror Unit the Log manager has to update the database using received log records. This is done by using interface number two.

In the Mirror Unit the Log system has to store the database image from time to time to SSS. That's why an interface is needed that can be used to fetch database objects from the database (interface 4).

4.2 Information description

Log mediator delivers log records to Log system. Log records include information about commited transactions either in Primary Unit or in Transient Unit. Each log record includes at least the identifier and the after-image of the data object.

In the Mirror Unit the Log system updates Mirror's copy of the database using log records. The newest value of the concerned object is copied into

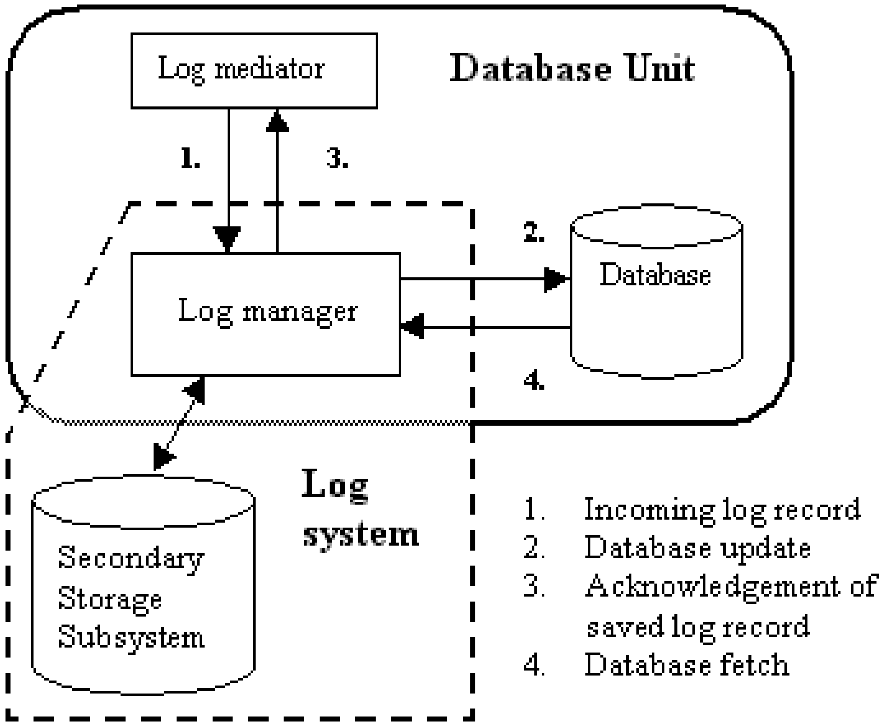

Figure 1: Software interfaces

the main-memory database. After Log manager has sent the log record to SSS it informs the Log mediator using mutual communication channel.

From time to time the Log manager saves an image of the database to SSS. It fetches a copy of every object in the database to build up the image.

5 Other requirements

5.1 Performance requirements

The goal of the RTD-Log project, in aspect of performance, is to make the log subsystem as fast as possible. The time required for a log record to be sent from Primary to Mirror, stored in the SSS, and receiving a acknoledgement for the log record in question, should be as short as possible. A major issue, and probably the most important feature, is making the recovery process fast and transparent to the Transient/Primary Unit. By regularly saving an image of the database and the index maintained in the Log Manager, the Recovering Unit can start accepting the log flow in a very early stage (in fact as soon as the index has been loaded into memory). Another requirement to the Log system is making it as modular as possible. Changing a module, for example the module managing storage to the SSS, should be easy and have no or as little as possible effect on the rest of the system.