Analysis of X.500 Distributed Directory Refresh Strategies

David W. Bachmann  
The University of Michigan

Michael A. Bauer  
The University of Western Ontario

J. Michael Bennett  
The University of Western Ontario

Guy A. Fasulo  
Chevron Information Technology Corporation

Michael H. Kamlet  
Bell Laboratories

Kevin H. Klinge  
Bellcore

Sailesh Makkapati  
The University of Michigan

Jacob Slonim  
IBM Canada Laboratory - Toronto

Toby J. Teorey  
The University of Michigan

Refresh strategies for distributed database directories, commonly recommended for the X.500 standard, are defined and analytically modeled for variations on push/pull and total/differential options under ideal asynchronous control conditions. The models are implemented in a HyperCard-based tool called DirMod (DDirectory ModelO). Experimental test results show an important elapsed time/performance trade-off among the different strategies, and test data contribute to the verification of the models.

Overview

The trend towards distributed systems is rapidly increasing. In particular, distributed database systems are becoming not only popular, but necessary, as applications and users become more distributed and demand all the services that are currently only available in centralized computing systems (Teor, 1990). One of the most important components of distributed systems in general, and distributed database systems in particular, is the directory. The directory is responsible for the management of the names of entities or objects in the distributed environment, their location, and other information about them. It is clearly central to the database administration function.

The objectives of a directory and the system it supports include location transparency, distribution transparency, expandability, performance control, high reliability and availability, and access control. Other desirable properties include both a procedural and nonprocedural user interface, data sharing, and an open architecture framework (BSS, 1989). Focusing on the performance control and data sharing issues, we find that the problem of replicating directory information and maintaining and updating the data is still being investigated (Bauer, 1990).

To examine this problem, we chose to study the emerging X.500 standard for directories. Quipu, an implementation of X.500, is readily available (Kille, 1989; KRRT, 1990; Rose, 1989a; Rose, 1989b), and therefore ideal for experimentation to discover “the good, the bad, and the ugly” about directories (BBS, 1989; ISO, 1989a; ISO, 1989b; ISO, 1990). The experiments we conducted form the basis of the parametric modeling of the refresh strategies.

The X.500 directory system is comprised of a set of interconnected open systems that cooperate to provide directory services (Bennett, 1989). X.500 uses an attribute-based naming system to identify objects within its name space. The attribute name space is built on a hierarchy, which allows the system to support a unique name for each object. X.500 accommodates access to objects even when
information is incomplete, by associating a set of attributes with each object and allowing users to browse the attributes at each node in the hierarchy [Bauer89].

The refresh strategies of the X.500 distributed directory system involve three main parameters (Bauer, 1990; LHMHPW, 1986):

1) Initiation Source
   - Pull: Initiated by a Shadow
   - Push: Initiated by the Supplier/Master

2) Size - Total vs Differential

3) Consistency - Synch vs Asynch. Synch implies that multiple levels of the DSA hierarchy are refreshed; asynch implies that only one level is refreshed.

There are therefore, eight possible strategies:

1) Pull, Total, Asynch
2) Pull, Diff, Asynch
3) Push, Total, Asynch
4) Push, Diff, Asynch
5) Pull, Total, Synch
6) Pull, Diff, Synch
7) Push, Total, Synch
8) Push, Diff, Synch

This paper presents our analysis of strategies 1-4, in which asynchronization (asynch) is the parameter in common. (The four synch strategies have not yet been analyzed; the intent is to do so in the future. All of these strategies are implemented in a HyperCard-based analytical model (see Section 3: The Distributed Directories Modeling Tool).

### Basic Definitions

The following are definitions of basic terms in X.500 (ISO, 1989a):

DIT: Directory Information Tree (Database). Each node in the DIT refers to a directory.

DSA: Directory Service Agent — a server responsible for manipulating the directory data.

Master DSA: The DSA that has administrative authority over a set of data. Updates to the data can only be done at the master.

Shadow DSA: This DSA maintains a read-only copy of the data and must get changes from the master. It can also be called a slave, a consumer, or a shadow-consumer.

Supplier DSA: This type of DSA cannot update itself, but it can update shadows (that is, it acts like both a master and a shadow).

Refresh: The process whereby the shadow receives a current copy of the data.

Pull: Shadow-initiated refresh. The update schedule is held at the shadow. (This is the implementation of Quipu.)

Push: Master-initiated refresh. The update schedule is held at the master.

Total refresh: The entire data set (whole table) is sent to the shadow. (This is the implementation of Quipu.)

Differential refresh: The set of changes to the data is sent to the shadow.

Synch: Multi-level refresh. In the case of a Push, all copies of the data will be consistent. In the case of a Pull, if a shadow receives refreshes from a supplier, then the supplier and the shadow will be refreshed.

Asynch: Single-level refresh. In the case of a Push, only shadows of the master will be refreshed. (This is the implementation of Quipu.)

Snap Time: The last time the shadow received a refresh. In Quipu, this is implemented with a version number.

To keep the terminology consistent, DSAs will be referred to here as either a master or shadow, although the DSA could be a supplier in any of the alternatives discussed.

### Basic Steps in the Refresh Process

#### Asynch Pull

The basic steps followed in the Asynch Pull refresh process are as follows:

1) The shadow prepares a message to the master indicating that it desires a refresh.
   1a) The shadow sends it to the CP (Communication Processor).
   1b) The CP then transmits the message over to the network.
   1c) The message is propagated over the network.
   1d) The message is deserialized by the CP of the master.

2) The master processes the message (i.e., it identifies which shadow is making the request, determines if there have been any changes to the data since the last refresh to the requesting shadow, etc.), and determines if a refresh is warranted (i.e., if the time of the last change to the data is later than snap time).

3) If a refresh is warranted, the master prepares the data for transmission (selection and packing time).
11 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the product's webpage:
www.igi-global.com/article/analysis-500-distributed-directory-refresh/51086?camid=4v1

This title is available in InfoSci-Journals. Recommend this product to your librarian:
www.igi-global.com/e-resources/library-recommendation/?id=2

Related Content

GR-OLAP: Online Analytical Processing of Grid Monitoring Information
www.igi-global.com/chapter/olap-online-analytical-processing-grid/20757?camid=4v1a

Antecedents of Online Game Dependency: The Implications of Multimedia Realism and Uses and Gratifications Theory
www.igi-global.com/article/antecedents-online-game-dependency/42086?camid=4v1a

Requirement Specification and Conceptual Modeling for Data Warehouses
www.igi-global.com/chapter/requirement-specification-conceptual-modeling-data/20689?camid=4v1a

Sanitization and Anonymization of Document Repositories
www.igi-global.com/chapter/sanitization-anonymization-document-repositories/8026?camid=4v1a