



Implementing High Performance Directory Services

By

Paul Mullen

Systems Administrator

HEAnet, Schools Team



Background

- Approached by the NCTE to build a login service
- Scalability challenges
 - 800,000 students
 - 50,000 staff
 - 4,000 schools
 - Multiple roles per school



Overview

- What are Directory Services?
- What is LDAP and why use it?
- Three Keys to an Efficient Directory
- Choosing Hardware
- Evaluating Directory Services
- High Availability and High Performance
- Dealing with Legacy Systems
- Some Benchmark Examples

What are Directory Services?

- Anything that provides an authoritative list of a group of users
- Allows searching for, and authentication of, users
- Think of a Phonebook that has a way of confirming which user is checking it.

What is LDAP?

- Lightweight Directory Access Protocol
- Not a directory – A way to access one
- Developed from X500
- Industry standard for access user information
- Examples
 - Active Directory
 - OpenLDAP
 - Sun Java Directory

Why Use LDAP?

- An Industry standard
- Employed by many applications
- Designed for efficient access to data
- Builds on experience of X500
- Optimised for searching of hierarchical data



The Three Keys



The Three Keys

- Know Your Data!



The Three Keys

- Know Your Data!
- Know Your Applications!



The Three Keys

- Know Your Data!
- Know Your Applications!
- Know How they Interact!

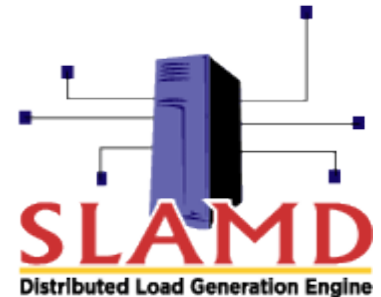


Hardware

- General observations
- I/O bound service
 - Fast disks help
 - RAM is better
- 1 x 4-core CPU is better than 2 x 2-core CPU
 - Scaling is better within the socket than across sockets
- Memory bandwidth is vital
 - Favours AMD over Intel and UltraSparc T1

Evaluation

- Benchmark *your* workload.
- Freely available tools
 - **SLAMD** - <http://www.slamd.com>
 - Slapd-auth – Part of the OpenLDAP test suite
- Other useful tools
 - **DirectoryMark** - <http://www.mindcraft.com/directorymark/>



High Availability and High Performance

- Many Designs are possible
- Single Master – Single Slave
 - Simple set up and architecture
 - No write redundancy
- Multiple Master – No slave
 - Read and write redundancy
 - Complex change conflict resolution

High Availability and High Performance

- Multiple Master – Multiple Slave
 - Search and Bind only on slaves
 - Slaves chain all writes to designated write master
 - Non LDAP fail-over required when write master fails
 - Larger outlay on hardware initially
 - Virtual Machines are typically not suitable for high performance directory services

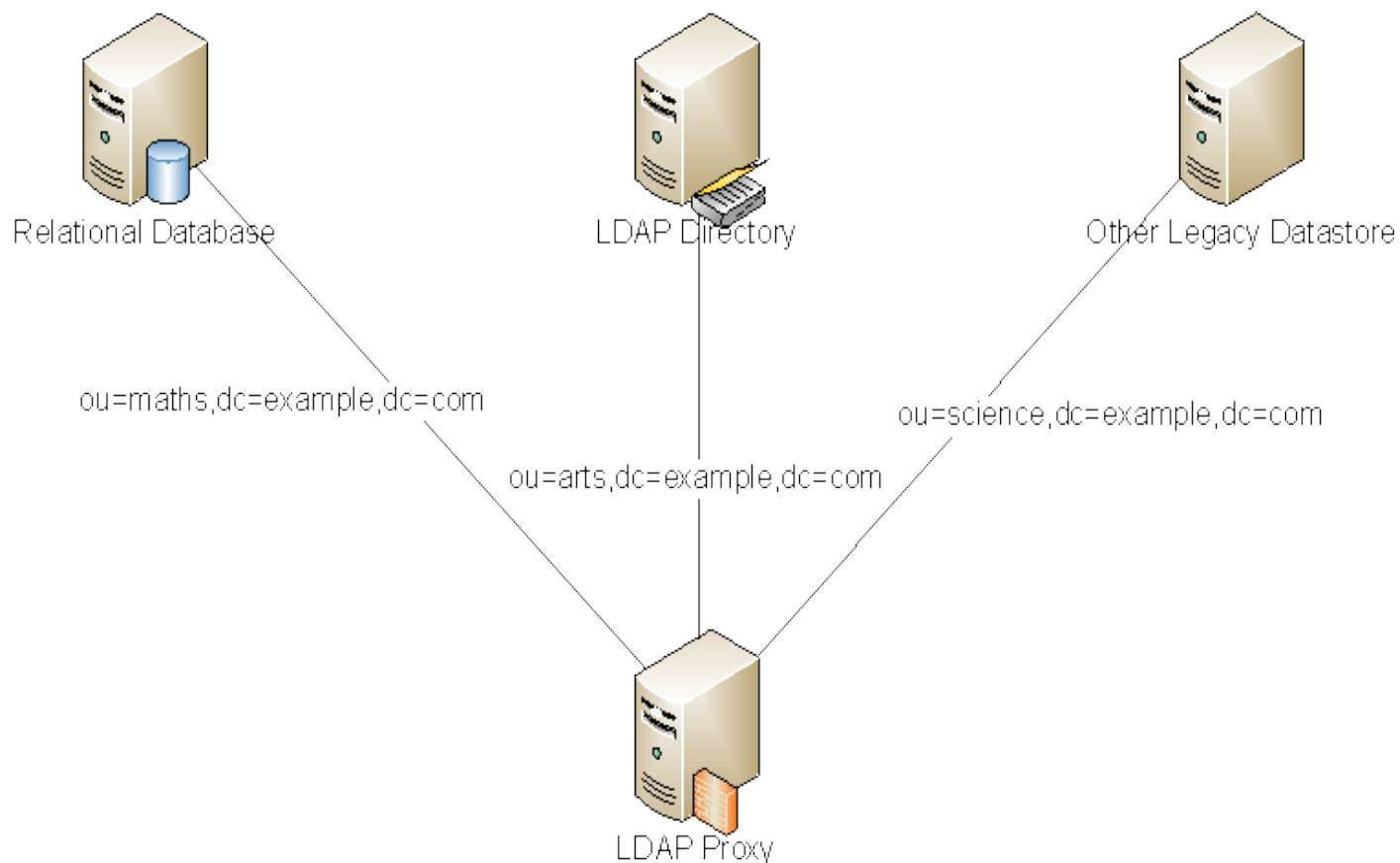
High Availability and High Performance

- Multimaster does NOT increase performance
 - Every write must be written to every master
- Single 'Write Master' is always preferable
 - Synchronising atomically is impossible at speed
 - Conflict resolution may not do what you expect
 - Always observe the principal of least surprise

Legacy Systems

- Multiple directories – Multiple technologies
 - Different name spaces
 - Can be 'glued' together with LDAP proxies
 - Caching LDAP proxies can increase performance
 - Complicated to set up but can bring a unified view of multiple incompatible directories together

Legacy Systems

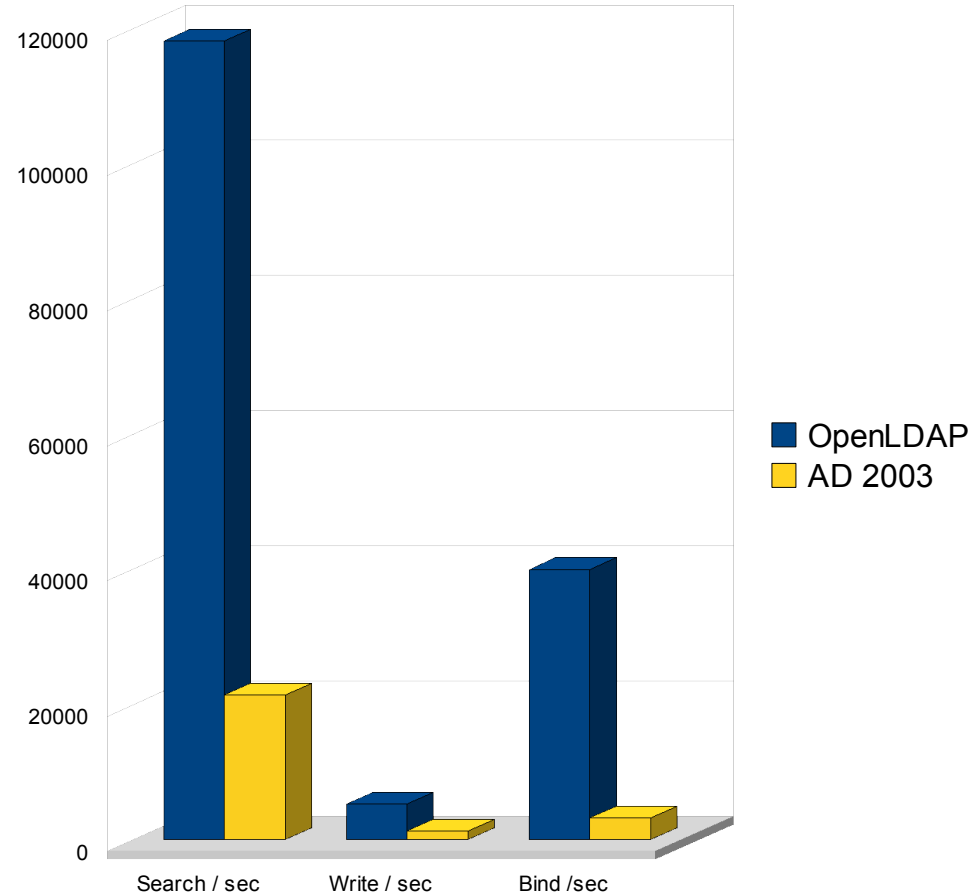


Empirical Evidence

- ~ 800K accounts ~ 3KB each ~= 2.4GB of Data
- 2 x Dell PowerEdge 2970
 - 2 x 2 Core AMD Opteron
 - 16 GB RAM
 - 5 x 73 GB 15K RPM SAS Drives
- Each capable of 40,000 Auths per second
 - Each Auth is 1 search + 1 bind
- Capable of > 4000 writes per second

Benchmark Results

- Active Directory results are Microsoft's numbers
- OpenLDAP results are our numbers
- Similar hardware
 - **Active Directory**
 - 4 x Opteron 850
 - 32GB RAM
 - **OpenLDAP**
 - 2 X Opteron 2220
 - 16GB RAM





Finally

- What you need to know before you start
- What you need to think about during a project
- How directories interact with hardware
- How you test your directory
- How to think about scalability and resilience
- A strategy for dealing with legacy data