Managing Network Data: Lessons from LDAP Directories

Divesh Srivastava
Database Research Department
AT&T Labs–Research
http://www.research.att.com/~divesh/
Overview

- Part I: A tutorial on directories
- Part II: Selected research on directories
Where do Directories Fit in?

- **Database technology driven by applications**

<table>
<thead>
<tr>
<th>APPLICATIONS</th>
<th>DATABASE SYSTEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Mining, Analysis</td>
<td>Data Warehousing</td>
</tr>
<tr>
<td>OLAP Applications</td>
<td>Bitmap Join Indexes</td>
</tr>
<tr>
<td><strong>DEN, Policy-based QoS</strong></td>
<td><strong>Distributed Directories</strong></td>
</tr>
<tr>
<td><strong>Network Applications</strong></td>
<td><strong>Scalable Namespaces</strong></td>
</tr>
<tr>
<td>Data integration</td>
<td>Semi-structured data</td>
</tr>
<tr>
<td>Web site management</td>
<td>Mediators</td>
</tr>
<tr>
<td>Replication</td>
<td>Recovery</td>
</tr>
<tr>
<td>Storage Management</td>
<td></td>
</tr>
</tbody>
</table>

- **Data-intensive network applications**
  - white pages, sendmail, authentication
  - policy-based connectivity, QoS, DEN

Databases used by network applications.
What is a Network Directory?

- Conceptually centralized information about all types of entities on the network
- Physically distributed (partitioned, replicated) over many directory servers

Distributed and shared network database.
Directory Features: Highlights

- Standard schemas
  - cn, mail, tel
- Heterogeneous objects and collections
  - role assignments, matching mail
- Flexible hierarchies
  - product categories, organizations
- Fine granularity access control
  - telephone number, SSN, password
- Distributed namespace
  - att.com, research.att.com, cs.wisc.edu

Critical for managing network data.
Application: Outbound Communication

- Goal: callers reach callees by name
  - reach Jim, not abc123@isp.com
  - reach Mary, not 212-555-1234

- Enabling technology: directories
  - name → telephoneNumber, mail
  - white pages, corporate directory

Standard schemas.
Application: Mobility Management

- Users are highly mobile
  - laptop user connected to any subnet
  - cell phone user in any one of many cells
- Goal: callers reach mobile callees
- Enabling technology: directories
  - callee’s home location registry (HLR) maintains callee’s current location

Standard schemas, distributed namespace.
Application: Inbound Communication

- Users access multiple communication devices
  - cell phone, fax machine, voicemail
- Goal: callees control access
  - if Bob calls, connect him to my cell phone, but if it’s someone named Paul . . . voicemail!
- Enabling technology: directories
  - callee’s policies specify communication preferences based on caller’s profile

Heterogeneity, flexible access control.
Application: DNS

- Applications use network names, but routing is based on IP addresses
- Goal: translate network name to IP address
  - location service
- Enabling technology: directories
  - designed to scale to very large numbers of entries, while remaining lightweight

Distributed namespace, hierarchies.
Application: Virtual Private Networks

- Internet used for business-business e-commerce
  - performance, privacy issues
- Goal: use the Internet as a private network
- Enabling technology: directories
  - customer policies specify QoS and security guarantees on provisioned pipes

Representative customer-level DEN application.
Application: Supporting QoS

- Routers support different traffic classes
- Goal: use capabilities of network elements to support customer-level DEN applications
- Enabling technology: directories
  - router policies specify actions on packets satisfying specific profiles

Representative network-level DEN application.
Types of Directories

- **Network OS directories**
  - Novell NDS, Microsoft Active Directory
- **Application-embedded directories**
  - Lotus Notes address book, Novell GroupWise, Microsoft Exchange directory
- **Service-specific directories**
  - DNS
- **General-purpose, standards-based directories**
  - X.500, LDAP

Trend towards standards.
Evolution of Directories

- General-purpose directories are abstractions of application-specific directories
  - work done in the networking community
- Relational databases were Codd-given: a fundamental departure from predecessors
  - work done in the database community

Evolution, not creationism.
Roadmap of Part I

- Standards
- Data model
- Query language
- Distribution
- Access control
- OpenLDAP
- Lessons
What is X.500?

- X.501: the model
- X.509: authentication framework
- X.511: abstract service definition
- X.518: distributed operation
- X.519: protocol specification
- X.520: selected attribute types
- X.521: selected object classes
- X.525: replication

Complex ITU standard.
What is LDAP?

- Protocol for accessing directory servers
  - based on X.500 DAP
  - better suited for the Internet
- Object-based data model
- Selection-based query language

IETF standard: RFC 1777 (v2), RFC 2251 (v3).
Roadmap of Part I

- Standards
- Data model
- Query language
- Distribution
- Access control
- OpenLDAP
- Lessons
LDAP Data Model: Highlights

- Heterogeneous entries with:
  - multi-valued attributes
  - missing attribute values
  - flexible schema evolution

- Hierarchical namespace of entries

Simple, flexible data model.
LDAP Data Model: Why Schemas?

- Schemas are necessary
  - distinguish databases from arbitrary data
  - important for database design, database administration, query optimization, etc.
- Schemas can be flexible
  - mandatory + optional schema elements
  - important for heterogeneous data

LDAP schemas are flexible.
LDAP Data Model: Schema

- Attributes (config/slapd.[user-]at.conf)

<table>
<thead>
<tr>
<th>name</th>
<th>alternative name(s)</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>cn</td>
<td>commonName</td>
<td>cis</td>
</tr>
<tr>
<td>member</td>
<td></td>
<td>dn</td>
</tr>
<tr>
<td>objectClass</td>
<td></td>
<td>cis</td>
</tr>
</tbody>
</table>

- Classes (config/slapd.[user-]oc.conf)

<table>
<thead>
<tr>
<th>objectclass person mandatory</th>
<th>objectclass groupOfNames mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>objectClass, cn, sn</td>
<td>objectClass, cn</td>
</tr>
<tr>
<td>optional description</td>
<td>optional member description</td>
</tr>
</tbody>
</table>

Cleanly decouples attributes from classes.
LDAP Data Model: Schema Classes

• **Core and auxiliary classes**

<table>
<thead>
<tr>
<th>Core Class</th>
<th>Auxiliary Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>objectclass person</td>
<td>objectclass inetAccount</td>
</tr>
<tr>
<td></td>
<td>mandatory</td>
</tr>
<tr>
<td></td>
<td>uid</td>
</tr>
<tr>
<td>objectclass groupedNames</td>
<td>optional</td>
</tr>
<tr>
<td></td>
<td>mail</td>
</tr>
<tr>
<td></td>
<td>labeledURI</td>
</tr>
</tbody>
</table>

• **Single inheritance core class hierarchy**

<table>
<thead>
<tr>
<th>objectclass top</th>
</tr>
</thead>
<tbody>
<tr>
<td>objectclass person</td>
</tr>
<tr>
<td>superior top</td>
</tr>
<tr>
<td>objectclass groupedNames</td>
</tr>
<tr>
<td>superior top</td>
</tr>
</tbody>
</table>

Core + auxiliary $\approx$ multiple inheritance.
LDAP Data Model: Entry

- Set of \((\text{attribute}, \text{value})\) pairs

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cn</td>
<td>divesh srivastava</td>
</tr>
<tr>
<td>sn</td>
<td>srivastava</td>
</tr>
<tr>
<td>uid</td>
<td>divesh</td>
</tr>
<tr>
<td>mail</td>
<td><a href="mailto:divesh@research.att.com">divesh@research.att.com</a></td>
</tr>
<tr>
<td>mail</td>
<td><a href="mailto:divesh@acm.org">divesh@acm.org</a></td>
</tr>
<tr>
<td>objectClass</td>
<td>person</td>
</tr>
<tr>
<td>objectClass</td>
<td>inetAccount</td>
</tr>
</tbody>
</table>

- An LDAP entry must
  - belong to one most-specific core class
  - have $\geq 1$ value for mandatory attributes

- An LDAP entry may
  - belong to multiple auxiliary classes
  - have $\geq 0$ value for optional attribute

Key elements of heterogeneity.
Conventional Data Models: Entries

• ODMG data model
  • multi-valued LDAP attribute = set-valued ODMG attribute
  • mandatory LDAP attribute = nonemptiness constraint on set-valued ODMG attribute
  • LDAP objectclasses = blow-up in number of ODMG classes

• Relational data model
  • LDAP entry split across multiple tables
  • in extreme case: represent each attribute of an LDAP entry in a separate table

A consequence of rigid schemas.
**LDAP Data Model: Instance**

- **Forest (hierarchical space) of named entries**
  
  ![Tree Diagram]

- **Class and instance hierarchies are orthogonal**
  - objectclass person does not inherit from objectclass organization
  - a person entry can be a descendant of an organization entry

Heterogeneous entries in a subtree.
Conventional Data Models: Instance

- **ODMG data model**
  - instance = set of homogeneous class extents
  - can model (parent, child) relationships in hierarchical namespace using object references

- **Relational data model**
  - instance = set of homogeneous tables
  - can model (parent, child) relationships in hierarchical namespace using foreign keys

No first-class hierarchical namespace.
Example: Policies, Profiles, Actions

- **Policy**: correlates profiles with actions
  - limit traffic flow between subnets
- **Profile**: identifies objects relevant to the policy
  - users, flows, network elements
- **Action**: specifies the desired behavior
  - drop packets

Arises in multiple network applications.
LDAP Data Model: Example

Relationships using embedded references.

Divesh Srivastava

Directories
LDAP Data Model: Example

Relationships using pure hierarchies.
Roadmap of Part I

- Standards
- Data model
- Query language
- Distribution
- Access control
- OpenLDAP
- Lessons
LDAP Query Language: Highlights

- Selection queries
  - no joins, no aggregation, no methods
- Hierarchically scoped, boolean filters

Philosophy: useful and efficiently evaluable.
LDAP Query Language: Boolean Filters

- Atomic filters: compare attributes with values
  - $cn=*$divesh*, $mail=*$, $SLAPriority<2$
  - can match entries of different objectclasses
  - class conditions: $objectClass=person$

- Combined using AND ($\&$), OR ($|$), NOT ($!$)

- Filter answer = entries whose (attribute, value) pairs satisfy the boolean filter
  - existential semantics

Filter answer can be heterogeneous.
LDAP Query Language: Scoping

- LDAP query: baseDN ? scope ? booleanFilter
  - baseDN: identifies an entry uniquely
  - scope: base, one, sub

Focus search within a subspace.
**LDAP Query Language: Example**

- **Query:** retrieve Divesh’s list of friends
  - `dc=att, dc=com ? sub ? cn=divesh*
  - `uid=divesh, ... ? one ? listName=friends`

Common to pose sequence of LDAP queries.
Roadmap of Part I

- Standards
- Data model
- Query language
- Distribution
- Access control
- OpenLDAP
- Lessons
Data Model: Physical Organization

- Each directory server manages a subspace of the directory information forest

- Conceptual unity achieved via (superior and subordinate) knowledge references

Scalable and efficient.
Distributed Evaluation by Referrals

• \( Q\): r2 \( ? \) sub \( ? \) objectClass=SLAPolicy

1. Client: \( Q \) -> S4
2. S4: referral to S1 -> client
3. Client: \( Q \) -> S1
4. S1: referral to S2 -> client
5. Client: \( Q \) -> S2
6. S2: local answers to \( Q \) -> client
   S2: referral to S3 -> client
7. Client: \( Q' \) (modified base) -> S3
8. S3: local answers to \( Q' \) -> client

• LDAP client combines answers from servers

Distribution logic at LDAP client.
Distributed Evaluation by Chaining

- $Q$: r2 ? sub ? objectClass=SLAPolicy

1. Client: Q -> S4
2. S4: Q -> S1
3. S1: Q -> S2
4. S2: Q' (modified base) -> S3
5. S3: local answers to Q' -> S2
6. S2: local answers to Q + remote answers from S3 -> S1
7. S1: remote answers from S2 -> S4
8. S3: remote answers from S1 -> client

- LDAP client unaware of distribution

Distribution logic at servers.
Distributed Evaluation: A Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Referrals</th>
<th>Chaining</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total query/answer traffic</td>
<td>Less</td>
<td>More</td>
</tr>
<tr>
<td>High inter-server bandwidth</td>
<td>No use</td>
<td>Useful</td>
</tr>
<tr>
<td>Server resource usage</td>
<td>Less</td>
<td>More</td>
</tr>
<tr>
<td>Client resource usage</td>
<td>More</td>
<td>Less</td>
</tr>
<tr>
<td>Client control of query plan</td>
<td>High</td>
<td>None</td>
</tr>
<tr>
<td>Client aware of distribution</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

- In the Internet (wide area network)
  - autonomous directory servers
  - client-server bandwidth same as server-server bandwidth
  - sufficiently powerful clients

LDAP directories implement referrals.
Distributed Relational Databases

- Distributed query evaluation uses server-server communication
- All servers belong to the same organization
  - high inter-server bandwidth
- Richer topologies, more complex queries
  - topologies: tree, star, ...
  - queries: joins

Distributed SQL evaluation \(\approx\) chaining.
Roadmap of Part I

- Standards
- Data model
- Query language
- Distribution
- Access control
- OpenLDAP
- Lessons
LDAP Directory: Access Control

• Many users access directory data

<table>
<thead>
<tr>
<th>cn:</th>
<th>divesh srivastava</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn:</td>
<td>123-45-6789</td>
</tr>
<tr>
<td>salary:</td>
<td>$500</td>
</tr>
<tr>
<td>uid:</td>
<td>divesh</td>
</tr>
<tr>
<td>mail:</td>
<td><a href="mailto:divesh@research.att.com">divesh@research.att.com</a></td>
</tr>
<tr>
<td>password:</td>
<td>who-knows?</td>
</tr>
<tr>
<td>objectClass:</td>
<td>AT&amp;T employee</td>
</tr>
</tbody>
</table>

• Read privileges differ per-attribute
  - password: only self
  - ssn: only self, department administrator
  - salary: only self, department manager
  - all else: universal access

Need for fine granularity access control.
Access Control: Mechanics

- A user binds to a DN
  - many authentication methods supported
- Access control determined by bind DN’s rights

User profiles need to be present in directory.
Access Control: Key Features

- No standard yet for LDAP access control
  - draft-ietf-ldapext-acl-model-06.txt
  - products support similar mechanisms
- Access control rule (ACR) specifies:
  - subjects: DNs of users, user groups
  - objects: attributes of LDAP query answer
- ACRs associated with object query’s base entry
  - ACR associated with child node takes precedence over ACR associated with parent node

Makes critical use of hierarchical namespace.
Access Control: Example

- Secretary access to all data in subtree
- Administrator access to ssn

ou: calendar appointments
objectClass: organizationalUnit

ou: personal address book
objectClass: organizationalUnit

listName: friends
listDefinition: ...
objectClass: list

listName: neighbors
listDefinition: ...
objectClass: list

- Self-access to all data in subtree
- Spouse access to all data in subtree

cn: divesh srivastava
ssn: 123-45-6789
mail: divesh@research.att.com
password: who-knows?
uid: divesh
salary: $500
objectClass: AT&T employee

Divesh Srivastava

ACRs apply to heterogeneous set of related entries.
Access Control Philosophies

- **Mandatory control**
  - subjects and objects assigned security levels
  - basis for multi-level security models

- **Discretionary control**
  - subjects given access to views of objects
  - more flexibility

Directory proposal based on discretionary control.
Roadmap of Part I

- Standards
- Data model
- Query language
- Distribution
- Access control
- OpenLDAP
- Lessons
Directory Products

- **Microsoft**: [http://www.microsoft.com/adsi/](http://www.microsoft.com/adsi/)
- **OpenLDAP**: [http://www.openldap.org/](http://www.openldap.org/)

And many, many more . . .
OpenLDAP

- Open source implementation of LDAP
  - slapd: stand-alone LDAP server
  - slurpd: stand-alone LDAP replication server
  - libraries: implementing LDAP protocol
  - utilities, tools, sample clients

- OpenLDAP release roadmap
  - 1.0: released 26 August 1998
  - 1.2.9: derived from U Mich release 3.3
  - 2.0: LDAPv3 support, in alpha testing

http://www.openldap.org/
OpenLDAP: slapd Architecture

Clean software architecture.
OpenLDAP: Storing LDAP Entries

- Each LDAP entry has an internal unique id

<table>
<thead>
<tr>
<th>Entry</th>
<th>DN</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>id</td>
<td>pid</td>
</tr>
<tr>
<td>entry</td>
<td>dn</td>
<td>cid</td>
</tr>
<tr>
<td>1279</td>
<td>1279</td>
<td>1043</td>
</tr>
<tr>
<td>...</td>
<td>ou=userProfiles, dc=att, dc=com</td>
<td>1279</td>
</tr>
<tr>
<td>2871</td>
<td>2871</td>
<td>1279</td>
</tr>
<tr>
<td>...</td>
<td>uid=divesh, ou=...</td>
<td>2871</td>
</tr>
</tbody>
</table>

- Entry attributes vertically partitioned

<table>
<thead>
<tr>
<th>mail</th>
<th>objectClass</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>id</td>
</tr>
<tr>
<td>value</td>
<td>value</td>
</tr>
<tr>
<td>2871</td>
<td>2871</td>
</tr>
<tr>
<td><a href="mailto:divesh@acm.org">divesh@acm.org</a></td>
<td>person</td>
</tr>
<tr>
<td>2871</td>
<td>2995</td>
</tr>
<tr>
<td><a href="mailto:divesh@research.att.com">divesh@research.att.com</a></td>
<td>person</td>
</tr>
</tbody>
</table>

- Indexes on various columns for fast access

Simplest possible mapping.
OpenLDAP: LDAP Query Evaluation

- Query translation: LDAP → pure boolean
  - $(B \ ? \ sub \ ? \ bf) \rightarrow (\& \ (dn=*B) \ bf)$
- Use available indexes on queried attributes
  - equality, substring, presence, approx
  - returns sorted id-blocks
- Use id-block merging for boolean operators
  - linear-time, block level optimizations
- Retrieve query answers using id2entry()
  - project attributes, create LDAPMessage

Very efficient.
OpenLDAP: Supporting Distribution

- Query translation: need to retrieve relevant subordinate knowledge references

  \[ (B \ ? \ sub \ ? \ bf) \rightarrow (\& (dn=*B) (| bf (objectClass=referral))) \]

- LDAP client can hide distribution from user.

Only referrals, no chaining.
OpenLDAP: Access Control Checking

- Access control rules maintained in memory
  - assumption: few ACRs at server
- A two phase approach to checking ACRs

```
Attribute-level access control
<table>
<thead>
<tr>
<th>Boolean query evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry-level access control</td>
</tr>
<tr>
<td>Atomic query evaluation</td>
</tr>
</tbody>
</table>
```

Clean design, permits query extensibility.
Summary of Part I

• What was covered
  • LDAP data model, query language, distribution, access control
  • how these are realized in the OpenLDAP open source system

• What was not covered
  • LDAPv3 protocol, database design, query extensibility, updates, caching, replication, ...

Divesh Srivastava

Directories
Roadmap of Part I

- Standards
- Data model
- Query language
- Distribution
- Access control
- OpenLDAP
- Lessons
Lesson 1: Standard Schemas

- Need standard schemas for typical objects
  - IETF: inetOrgPerson, calendar
  - flights, traffic data, stocks
- Need standard schemas for meta-data

Critical for network applications.
Lesson 2: Flexible Schemas

- LDAP classes are not rigid tables
  - optional attributes
- Attributes can have multiple values
  - email addresses, role assignments
  - incrementally specify attribute values

Important for managing heterogeneous data.
Lesson 3: Graceful Schema Evolution

- LDAP classes can evolve
  - add new attributes
- Attributes can evolve
  - one value → multiple values
- Existing applications should not break

Important for managing evolving data.
Lesson 4: Hierarchies

- The world is not flat
- Representing hierarchies in a relational model is difficult
- Constructing hierarchies from relational tables requires many joins

Many complex hierarchies in real world.
Lesson 5: Heterogeneous Collections

- Allow queries to range over heterogeneous sets of objects
  - match email against users and groups
- Allow queries to return heterogeneous sets of objects
  - distinguish in client application

Real world is not homogeneous.
Lesson 6: Scalable Namespaces

- Large scale physical distribution of data
  - namespace partitioned across sites
  - use hierarchy, range, or hash
  - decentralized namespace assignment
- Conceptual unity
  - use namespace referrals

Cooperation, yet autonomy.
Lesson 7: Flexible Access Control

- Allow fine granularity access control
  - per attribute
- ACRs on heterogeneous set of related objects
- Access control policy = ACRs + delegation + ACR prioritization

Controlled sharing of information.
Lessons from LDAP Directories

- Lesson 1: Standard schemas
- Lesson 2: Flexible schemas
- Lesson 3: Graceful schema evolution
- Lesson 4: Hierarchies
- Lesson 5: Heterogeneous collections
- Lesson 6: Scalable namespaces
- Lesson 7: Flexible access control
Additional Resources

- Innosoft’s LDAP world
- LDAPv3 standards documents
- Shukla & Deshpande’s SIGMOD’00 tutorial

Just a few starting points.
Directory Limitations

- Limited query language capability
  - no joins, no aggregation
- Transactional update consistency
  - loose consistency.

Not a general purpose database.
Roadmap of Part II: Research

- Query language extensions
- Centralized query evaluation
- Distributed query evaluation
- Bounding schemas
- Other research
LDAP Query Language: Revisited

- Selection queries
  - no joins, no aggregation, no methods
- Hierarchically scoped, boolean filters

Philosophy: useful and efficiently evaluable.
QoS Application: Revisited

Policies, profiles, actions.
LDAP Query Language: Limitations

- Conditions on correlated entries
  - find applicable SLAPolicy
  - find highest priority applicable SLAPolicy
- Application poses sequence of LDAP queries

User inconvenience + potential inefficiency.
Highlights of Contributions

- Useful query language constructs
  - multiply scoped queries
  - hierarchical selection operators
  - aggregate selection operators
- Efficient centralized operator evaluation
  - linear I/O and CPU complexity
- Efficient distributed evaluation
  - scales linearly, topology-independent

Query language matches LDAP data model.
Simple Hierarchical Selection

- find applicable SLAPolicy

Simple selection using structural semi-joins.
Simple Aggregate Selection

- find highest priority applicable SLAPolicy

\[ \text{SLAPolicyName: dso} \]
\[ \text{SLAPriority: 2} \]
\[ \text{SLAPolicyScope: ...} \]

\((g \quad \text{SLAPriority} = \max(\text{SLAPriority}))\)

Aggregate selection, not aggregate computation.
Hierarchical Aggregate Selection

- find policies with more than one traffic profile

Aggregate selection, not aggregate computation.
Roadmap of Part II

- Query language extensions
- Centralized query evaluation
- Distributed query evaluation
- Bounding schemas
- Other research
Why Not Merging?

- Straightforward quadratic algorithms
- Linear merging works for boolean operators
  - sorted id-blocks $\rightarrow$ sorted id-blocks
- Problem with parent, child: inversion

Need to go beyond merging.
Stack-Based Efficient Evaluation

- Sorted lists of DNs $\rightarrow$ sorted list of DNs
  - not sorted id-blocks $\rightarrow$ sorted id-blocks
- Sequence of stack DNs $\leftrightarrow$ entries on a path
  - (parent, child) $\Rightarrow$ stack adjacency
  - stack adjacency $\Rightarrow$ (ancestor, descendant)
- Incremental aggregate computation
  - $\text{descendants}(e) \subseteq \text{descendants}(p(e))$

Linear I/O and CPU complexity.
Stack-Based Evaluation: Example

Amortized linear I/O and CPU complexity.
Stack-Based Evaluation: Example

(c count() > 1)

Operand 1       Stack        Merged operands

Aggregate selection preserves complexity.
Implementation in OpenLDAP

• Problem: working with DNs is expensive
  o long DNs, id2dn(), dn2id()
• Solution: use IDNs (root to node id sequence)
  o shorter IDNs, id2idn(), no idn2id()
• Additional costs:
  o \[ \text{id-blocks } \xrightarrow{\text{sorting}} \text{ IDN-blocks: before any operator evaluation} \]
  o \[ \text{IDN-blocks } \xrightarrow{\text{sorting}} \text{ id-blocks: after all operator evaluations, to use cache and id2entry()} \]

OpenLDAP is flexible.
Roadmap of Part II

- Query language extensions
- Centralized query evaluation
- Distributed query evaluation
- Bounding schemas
- Other research
Evaluation by Referrals: Revisited

- **LDAP query** $Q$: $r_2$ ? sub ? $bf$

1. Client: $Q$ -> $S_4$
2. $S_4$: referral to $S_1$ -> client
3. Client: $Q$ -> $S_1$
4. $S_1$: referral to $S_2$ -> client
5. Client: $Q$ -> $S_2$
6. $S_2$: local answers to $Q$ -> client
   $S_2$: referral to $S_3$ -> client
7. Client: $Q'$ (modified base) -> $S_3$
8. $S_3$: local answers to $Q'$ -> client

- LDAP client combines answers from servers

Servers can compute local answers independently.
Advanced Query Evaluation: Goals

• Problem: query answer in one server, with its witness in a different server
  • example query $Q$: $(d \ Q_1 \ Q_2)$
  • cannot compute answers independently

• Goal: a distributed query evaluation such that:
  • no server-server communication
  • $|\text{total traffic}| = |\text{query answer}| + \text{some function of } (|\text{query}|, |\text{distribution topology}|)$

As efficient as distributed LDAP query evaluation.
A Key Observation

- Two components of distributed evaluation:
  - identification of relevant servers
  - distributed answer computation

Interleaved in distributed LDAP query evaluation.
Identification of Relevant Servers

- Each directory server maintains the distributed topology of directory servers
  - consistency: topology rarely changes
- A separate directory server maintains the distributed topology
  - pointed at from each directory server
- Dynamically compute using an LDAP query

Cost amortized over multiple queries.
Answer Computation: Example

• $Q$: (d AQ1 AQ2)

Answers from S3/S4
1. Client: $Q \rightarrow S3/S4$
2. S3/S4: local answers $\rightarrow$ client

Answers from S2
1. Client: exists(AQ2) $\rightarrow$ S3
2. S3: yes/no $\rightarrow$ client
3. if (no) Client: (d AQ1 AQ2) $\rightarrow$ S2
   else Client: (d AQ1 (| AQ2 rr3)) $\rightarrow$ S2
4. S2: local answers $\rightarrow$ client

Answers from S1
1. Client: exists(AQ2) $\rightarrow$ S2/S3/S4
2. S2/S3/S4: yes/no $\rightarrow$ client
3. Client: (d AQ1 (| AQ2
   if (yes@S2 OR yes@S3) rr2
   if (yes@S4) rr4 )) $\rightarrow$ S1
4. S1: local answers $\rightarrow$ client

• Subordinate references as pseudo-witnesses

Coordination, not computation, at client.
Distributed Answer Computation

- Each answer to $Q$: $(d \ Q_1 \ Q_2)$ belongs to a particular server
  - **union** plan for evaluating $Q$
- A witness entry satisfying $Q_2$ may belong to same server or any descendant server
  - yes/no subqueries to descendant servers
- Generate distributed query evaluation plan in a bottom-up fashion
  - use topology knowledge for composing plans

Non-(query answer) traffic is data independent.
Witness Query Cache at LDAP Client

- Problem: a yes/no witness query may be posed at a server multiple times
  - proportional to topology depth
- Solution: maintain a small witness query cache at LDAP client
  - proportional to query size * topology size

Linear scale-up, independent of topology.
Distributed SQL Evaluation

- $R_1 @ S_1 \bowtie R_2 @ S_2$:
  - communicate table $R_2$ to site $S_1$
  - use semijoins for optimization
- $|\text{Intermediate answers}| \gg |\text{Query answers}|$

Tradeoff: communication versus computation.
Roadmap of Part II

- Query language extensions
- Centralized query evaluation
- Distributed query evaluation
- Bounding schemas
- Other research
LDAP Content Schema: Revisited

- Primacy of attributes
- Each objectclass specifies
  - mandatory attributes
  - optional attributes
- Core and auxiliary objectclasses
  - mandatory: core objectclasses
  - optional: auxiliary objectclasses

Provides [lower, upper] bounds on instances.
LDAP Schema: Limitations

- Two components of a directory instance
  - contents of an entry
  - hierarchical structure of entries
- LDAP schema has focused on entry content

No schema to guide hierarchical structure.
Structure Schema: Highlights

- **Goal**: impose some order on hierarchical structure of instance, without losing flexibility
- Specify mandatory structure schema
  - every profile must have a parent orgUnit
- Specify forbidden structure schema
  - no SLAPolicy can have a child profile

Flexible schema for hierarchical structure.
Mandatory Structure Schema

• Specify hierarchical structure that must occur

• Specified in terms of objectclasses

A lower-bound on hierarchical structure.
Forbidden Structure Schema

- **Specify hierarchical structure that cannot occur**

- **Specified in terms of objectclasses**

An upper-bound on hierarchical structure.
Testing Legality of Instances

- Does a directory instance conform to a bounding structure schema?
  - consistent information representation

- Legality testing \(\equiv\) checking emptiness of a hierarchical selection query
  - each bounding schema constraint can be separately translated to a query
  - bulk legality testing is efficient

Effective use of query evaluation algorithms.
Schema Constraint Translation

- **Mandatory structure schema constraint**

  ```
  SLAPolicy
  (-(objectClass=SLAPolicy)
   (d (objectClass=SLAPolicy)
    (objectClass=profile)))
  ```

- **Forbidden structure schema constraint**

  ```
  SLAPolicy
  (x (-(objectClass=SLAPolicy)
     (c (objectClass=SLAPolicy)
      (objectClass=profile)))
  ```

Other translations exist as well.
Incremental Legality Testing

- Atomic updates permitted: only leaf entries can be inserted to or deleted from forest
  - testing legality after update not robust
- Update transaction: sequence of updates
  - test legality after update transaction
  - too complex for analysis
- Update granularity for legality testing: insertion and deletion of disjoint subtrees

Desirable modularity and robustness properties.
Incremental Testing: Highlights

- Trivial: no test required
  - forbidden structure schema under deletions
- Purely incremental: all tests on $\Delta D$
  - child and descendant mandatory structure schema under insertions
- Partly incremental: some tests on $\Delta D$
  - forbidden structure schema under insertions
- Non-incremental: all tests on $D - \Delta D$
  - child and descendant mandatory structure schema under deletions

Translation to emptiness query permits elegance.
Research Summary

- Focus on significant new class of data-intensive network applications
- Efficiently evaluable extensions to the LDAP query language to support applications
- Natural bounding schema extensions to LDAP data model to support consistency

Right abstraction level simplifies query specification + enhances data integrity
Roadmap of Part II

- Query language extensions
- Centralized query evaluation
- Distributed query evaluation
- Bounding schemas
- Other research
What Have I Not Covered?

- Meta-directories for data integration
- Triggers for LDAP directories
- Combining LDAP with XML
- Dereferencing embedded DNs
- Caching LDAP query templates
- ...

A lot of interesting and useful research.
LDAP Colleagues and Collaborators

- Sihem Amer-Yahia: AT&T Labs–Research
- Sungran Cho: Stevens Institute of Technology
- Sophie Cluet: INRIA Rocquencourt, France
- Richard V. Huber: AT&T Labs
- H. V. Jagadish: University of Michigan, Ann Arbor
- Mark A. Jones: AT&T Labs–Research
- Olga Kapitskaia: Pôle Universitaire Léonard de Vinci, France
- Laks V. S. Lakshmanan: IIT–Bombay, India
- Tova Milo: Tel-Aviv University, Israel
- Raymond T. Ng: University of British Columbia, Canada
- Dan Suciu: University of Washington, Seattle
- Dimitra Vista: Cigna