Architecting the Modern LDAP Renaissance:

The Apache Directory Vision

Abstract

Directory technology is an indivisible cornerstone in computing science and LDAP specifically is essential in several industries however it is severely underutilized. One would expect the demand for LDAP to increase as infrastructures and the Internet grow, with more boundaries, nodes, services, users and ways of doing business emerging rapidly. Directories inherently solve integration problems yet more integration problems appear while complexities of existing problems are compounded. We're not witnessing a proportional increase in the adoption rate of directories; namely LDAP directories. This is not a coincidence. It is the result of a lack of several factors: tooling support, courses on directory technology in academia, qualified domain experts and rich integration constructs. More specifically, information architects and key decision makers incorrectly choose to apply ad hoc solutions to problems rather than opting to using directories. If these limitations are removed then there would be greater comfort, flexibility and adoption. LDAP could do more than it does today namely in the area of provisioning and workflow. LDAP could potentially experience a Renaissance with renewed interest due to increased demand to solve the classical integration problems it was designed for and beyond. We discuss these limitations, and the proposed means to remove them, all in an effort to express our vision at the Apache Directory Project. Our aim is clear; we intend to influence and incite other directory implementers and projects by our example to trigger what we envision as the Modern LDAP Renaissance.

Drivers Leading to the Birth of the Directory

Several systems slowly appeared in the arsenal of tools used daily by those within and across organizations. Data architects quickly could characterized two logical categories of information spanning across all domains:

1. Static information shared across applications and systems read frequently yet seldom altered.

2. Information specific to an event within an application (a transaction) read infrequently however written rapidly in volume (data seldom accessed by external systems).

The Directory relates to this first type of data above which has the following general properties:

- Accessed by many clients
- Partitioned across different locations in different organizations owned by separate authorities
- Structure data elements with data types (syntaxes) with semantics for matching
- Data easily fits into hierarchies
- Highly cross referenced

The telecommunication industry in Europe, in particular, felt this burden while managing shared information, specifically subscriber directories. Telephone companies in Europe, some national providers and other international providers, managed profiles for subscribers, a subset of which, needed to be published for public access.

The ITU responded with the X.500 series of specifications. The Directory specifications addressed these specific concerns for accessing information shared across organizations, located in separate stores and managed by several different authorities. The access model and authoritative model enabled organizations to publish subscriber information and allow others to access it while delegating the management of access control, schema and other aspects to the proper authorities. As a standard it was way ahead of its time.

The X.500 specifications reveal a unique data access and authoritative model designed specifically for the category of static shared data we defined above. It is a cornerstone technology as is the relational database for the second category.

Demand for Directories Should Be High

Directories expose a unique model for efficiently accessing relatively static information shared across multiple systems. Directory protocol designers intended the protocol to address a class of specific problems through a standard means to access shared information centrally while providing high availability. Existing storage paradigms failed to solve these problems due to inherent limitations in their design.

These data management problems fall under the domain of data integration. To integrate systems one must integrate both the data and the processes. Interoperating systems often need to access the same information however many systems simply duplicate that information (in relational databases) rather than accessing it from a centralized service (the directory). As a result multiple copies of the same information exist across these systems. Data synchronization technologies emerged to address these issues of duplication such as meta-directories (not real directories). These technologies never really solved all the issues of data duplication: they merely facilitated the export, transformation and import of data across systems. The serious issue of determining the authoritative copy still remained.

For some time, commercial entities avoided addressing these integration problems by centralizing access to this shared information. The market fueled the push of OLTP systems since the transaction equated to money. As markets became saturated as several competitors appeared, businesses started focusing on gains through efficiency. To streamline development and maintenance better integration was needed while allowing transactions to span across interoperating systems. This increased the need for directories.

Aspects of growth naturally contribute to the demand for Directory technologies. As the number of users, organizations, systems, services and machines expand the problems of integration only increase without applying the correct remedy. In fact, the vast majority of today's problems are primary integration problems. Hence one would expect the utilization of Directories to increase proportionally with the increased demand. Unfortunately, a proportional increase in demand for Directories has not been observed.

Barriers of Adoption

Although Directories, namely LDAP Directories, have proliferated over the past two decades, their adoption has not been commensurate with the demand one would expect. Several factors may be attributed to this less than optimal adoption rate.

Directories still feel alien to most developers. Many admit, in theory, to a directory as the ideal access model for shared information in their applications however most fall back to using a relational database instead. Why?

First there is no formal education around directory technologies while several courses around relational database systems exist. This might explain in part why developers are much more comfortable with keeping their Directory information stored and accessed from an RDBMS. As a result, finding qualified Directory experts is difficult. Every generalist knows how to use an RDBMS, yet costly specialists are needed to

design, operate and maintain directories. The cost and availability of specialized engineers factor into design decisions especially if the process is formally conducted.

Developers may also fall back to using an RDBMS instead of a Directory because of a lack of rich integration tier constructs (like triggers, views and stored procedures). These very useful constructs are missing in Directories yet have been available in relational databases for decades. Today's heavy integration needs demand these constructs. For example the demand for the event driven provisioning of information could be elegantly solved using triggers and stored procedures as entries are added, updated and deleted. Data architects and developers in this respect have been left in the dark-ages with LDAP while the RDBMS has given them what they wanted/needed at the price of compounding their problems with data duplication.

Perhaps the most significant factor driving these faulty decisions is the lack of LDAP tooling. There are myriads of RDBMS tools for various aspects: tuning, accessing, and designing. Tools for the RDBMS appeared with a vengeance to enable designers to rapidly prototype new OLTP systems and thus bring them to market faster. Less confident generalists could even design databases and manipulate them thanks to these tools which abstracted away the details to just get the job done. The tools made engineers and non-engineers productive in building database driven applications rapidly. Unfortunately, there barely are any good browsers for LDAP much less topology/schema, design and tuning tools. The barren tooling landscape leaves developers and data architects exposed enough to turn back to the RDBMS for immediate comfort. This temporary gain is at the cost of long term data bottle necks or synchronization issues.

Regardless of which factor prevents most the uptake of LDAP, the fact remains: the wrong decisions are being made more often than the right ones. It's a matter of time before Directories are considered antiquated technologies unless they can accommodate modern needs. Directories must be compelling enough for data architects and developers to decide to utilize them for their strengths in theory and in practice.

Renovating (LDAP) Directories for the 21st Century

Several factors are not under the immediate control of the LDAP community however enough are to bring LDAP to the 21st century. First and foremost, LDAP tooling must improve. We need tools to model LDAP directories properly for topology and schema. Designers should be guided on how they can properly use more of the esoteric constructs to control the naming, and hierarchy of the directory. There's more to directory design than just objectClasses and attributeTypes.

Secondly the protocol must expand LDAP to include standard mechanisms for declaring triggers, stored procedures and views. These two aspects alone contribute to the vast majority of reasons why relational databases store information best accessed from a directory. Allowing these new constructs for a directory will enable virtualization, provisioning, referential integrity, and server interaction with the external environment. Rather than only exposing access to static information (white pages) the directory could be utilized to solve many more problems confronting designers and developers in today's integration era. The Directory would truly become the Swiss Army Knife™ of integration tools in the data architect's tool chest.

Conclusion: The Aim of the Apache Directory Project

At the Apache Directory Project, there are two LDAP specific projects. The first is a pure Java LDAP Server called Apache Directory Server (ApacheDS) which has been written and certified by the Open Group for LDAPv3. It was started in reaction to the often brittle code that was too hard to manage: the code was written in C and had preprocessor directives strewn all over it for porting.

The other LDAP related project is called Apache Directory Studio. It is intended to provide the missing LDAP tooling support in addition to Apache Directory Server specific management utilities.

The general motives for forming the Apache Directory Project can be summarized with the points below:

1. Avoid low level compiled language for:
    a. Reducing complexity, and increase productivity
    b. Enabling portability to other platforms with minimal effort
    c. Attracting more contributors with most common language (Java)
2. Devise a more flexible dynamic server designed for:
    a. Extension
    b. Protocol experimentation
    c. Hot plug-ability
    d. Simplicity
3. Experiment with and introduce new integration tier constructs for directories:
    a. Triggers

    b. Views
    c. Stored Procedures
    d. Queues
4. Provide the missing tooling critical for:
    a. LDAP Adoption (general for all LDAP servers)
    b. Increasing user comfort with LDAP
    c. Lessening the learning curve for those using LDAP
5. Stimulate commercial vendors to adopt these new capabilities to remain competitive.

One of our aims is to induce thought in the LDAP community while stimulating its growth through simple useful products. The more satisfied LDAP users are, the bigger the community using and interested in LDAP. This is where simple intuitive tooling and a server come in hand. More than that, these projects serve as an experimentation platform for devising new constructs in the protocol and evaluating their utility without breaking with the existing protocol. This is why certification is very important to us as we devise new constructs to make LDAP much more useful to its users.

Ultimately we want to see changes appearing in the protocol standard which make LDAP a more familiar hence appealing, and easier to use technology. With standards around these new constructs interoperability could be achieved. Users will not be bound to a specific server. We want to demonstrate the viability and profound effects of these new constructs in the protocol through our example while stimulating commercial vendors to do the same to remain competitive. It does not matter to us which LDAP server is used so long as the users' situation improves with these new productive protocol features to make LDAP serve its intended purpose and go beyond. We strive, in this way, to increase LDAP awareness, comfort and adoption to bring forth what we call the Modern LDAP Renaissance.