

Storing LDAP data in MySQL Cluster

Howard Chu

CTO, Symas Corp. hyc@symas.com

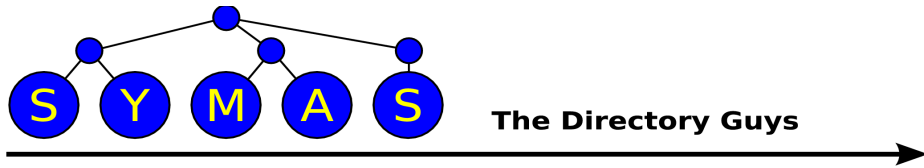
Chief Architect, OpenLDAP hyc@openldap.org

Ludovic Poitou

Architect, Sun Microsystems.

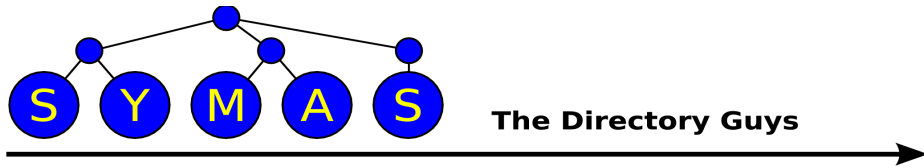
OpenDS Community Manager

ludovic.poitou@sun.com



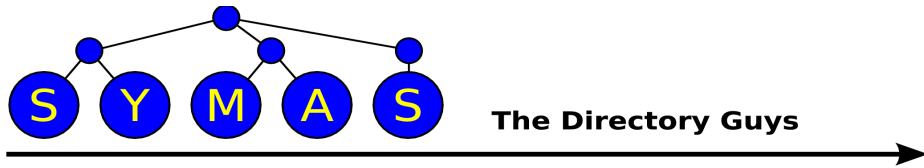
OpenLDAP Project

- Open source code project
- Founded 1998
- Three core team members
- A dozen or so contributors
- Feature releases every 12-18 months
- Maintenance releases roughly monthly



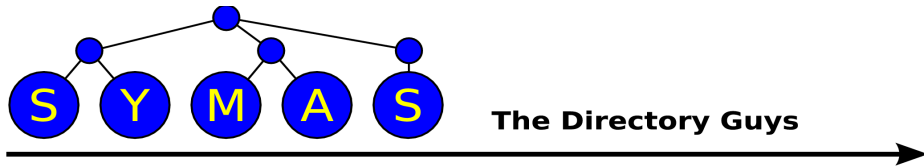
A Word About Symas

- Founded 1999
- Founders from Enterprise Software world
 - *platinum* Technology (Locus Computing)
 - IBM
- Howard joined OpenLDAP in 1999
 - One of the Core Team members
 - Appointed Chief Architect January 2007



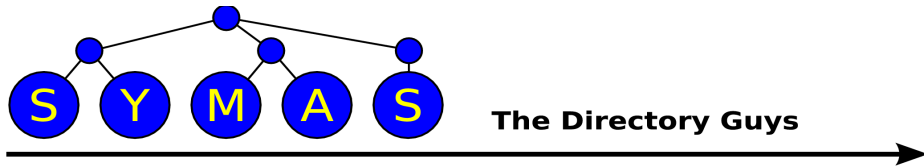
OpenDS Project

- Open source code project
- Founded in July 2006
- Sponsored by Sun Microsystems
- 43 committers, 16 full time, all Sun paid
- Half dozen contributors
- OpenDS 1.0 in July 2008
- OpenDS 1.2 in Feb 2009
- OpenDS 2.0 in July 2009



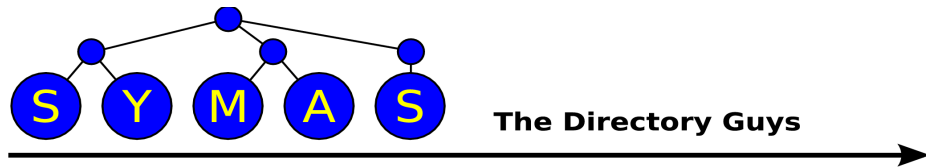
Topics

- Overview
- Relational vs Hierarchical Data models
- Accessing Relational data from LDAP
- A design for LDAP to Relational
- OpenLDAP Back-NDB Backend
- OpenDS NDB Backend
- Future Directions



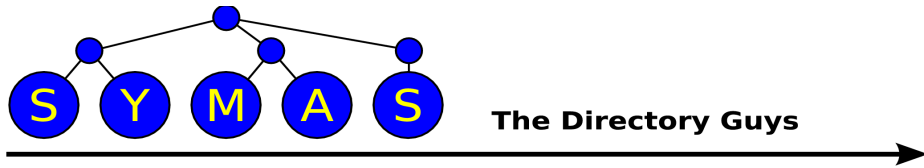
Overview

- LDAP directory servers scalability and reliability have greatly contributed to the success of the technology.
- LDAP directory servers have been proven to scale to billions of objects and terabytes of data, with performance in excess of 100,000 queries/second at sub-millisecond latencies.
- Some deployments have demonstrated 5 Nines availability.



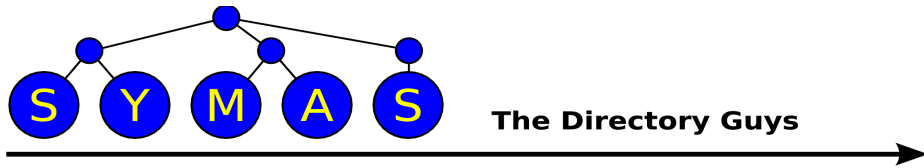
Overview

- The current design depends on having a very powerful single machine to achieve maximum scaling.
- The trend in data centers has been to scale using clusters that can be grown incrementally.
- A cluster-friendly backend design was needed.
- As luck would have it, MySQL released a cluster-based database engine while we were beginning our own cluster-oriented design effort.
- Leveraging MySQL's relational database engine in LDAP is not straightforward.



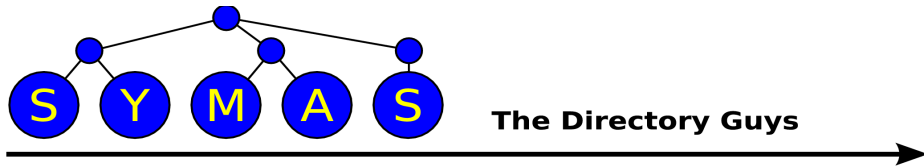
Overview

- The hierarchical data model of the directory and the tabular data model of relational databases (RDBMSs) are fundamentally different
- Both are ubiquitously useful
- Access to one from the other is frequently desired
- Solutions for providing cross-access exist but tend to be sub-optimal
- The new LDAP solutions developed in cooperation with MySQL leverages the strengths of both technologies



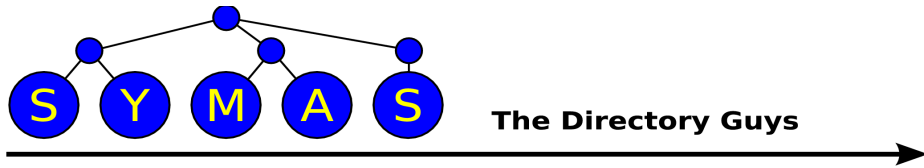
Relational vs Hierarchical

- RDBMSs are built on tables of rows and columns
 - One “record” is one row of columns
 - One value is stored per cell of the table
 - Values have predefined size
- Directories are built from trees of objects
 - One “record” is an object with arbitrarily many attributes
 - An attribute has arbitrarily many values
 - Values have arbitrary size



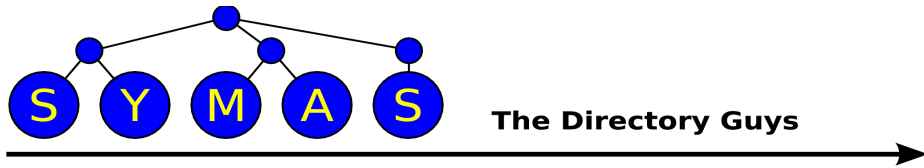
Storing LDAP data in RDBMS

- RDBMSs generally don't support multiple values for a single field/attribute
 - Normalization requires only one value per field
 - Supporting multi-valued attributes requires dedicating a separate table per attribute
 - Combining values across multiple tables typically requires many disk seeks and thus performs poorly



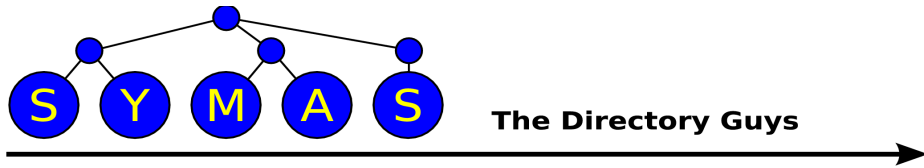
Storing LDAP data in RDBMS

- LDAP uses Distinguished Names (DNs) as primary key
 - The directory namespace is inherently hierarchical, but the RDBMS namespace is inherently flat, so the DN cannot be used directly as an RDBMS primary key



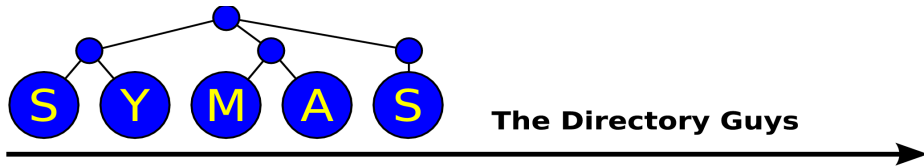
Cross Access

- LDAP access to RDBMS
 - OpenLDAP has provided back-sql since release 2.0
 - It requires a lot of manual setup, and performance is poor because it goes thru many translation layers
- RDBMS access to LDAP
 - Generally there's no direct access: export the LDAP data, massage it, import to RDBMS



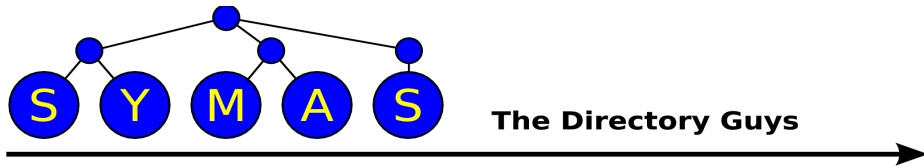
Open Source to the Rescue

- OpenLDAP is the world's most powerful LDAP software
- MySQL is the world's most popular open source relational database
- Open development models allow seemingly intractable obstacles to be overcome



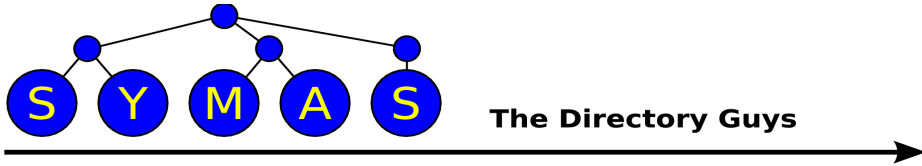
Back-NDB Design

- Uses a DN to ID table to map DNs to numeric IDs
- Numeric IDs are used as the primary key of the main data tables
- Generally uses a separate table per objectclass
- LDAP entries that have multiple objectclasses may have their data split across many tables
- The list of objectclasses for an entry must be known, to identify which tables hold the entry's data



DN Mapping

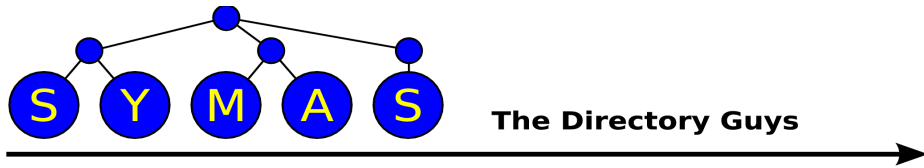
- DN2ID table
 - 16 column primary key, one column per RDN of a DN (thus, the directory tree is limited to 16 levels deep)
 - 1 column numeric ID (generated by autoincrement)
 - 1 column objectclass (contains multiple class names, delimited by spaces)



DN Mapping

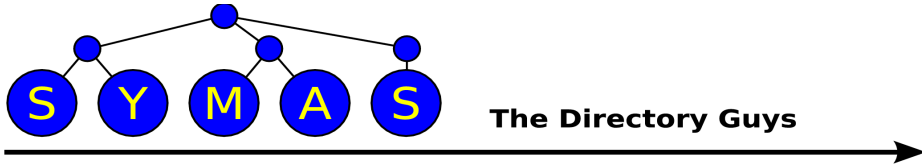
- DN2ID table example

a0 ... a15					eid	objectclasses
dc=com	dc=example	(null)	(null)	(null)	1	dcObject organization
dc=com	dc=example	ou=users	(null)	(null)	2	organizationalUnit
dc=com	dc=example	ou=groups	(null)	(null)	3	organizationalUnit
dc=com	dc=example	ou=groups	cn=staff	(null)	4	groupOfNames
dc=com	dc=example	ou=users	cn=Joe M	(null)	5	person inetOrgPerson



ObjectClass Mapping

- Data is distributed in a separate table per objectclass
 - Since NDB is memory-resident, disk seeks are not an issue
- But, attributes may only appear in one table
 - Inherited attributes only appear in the parent class's table
 - "Attribute Sets" are used to collect attributes that have multiple unrelated references
 - Attribute Sets are defined in slapd config



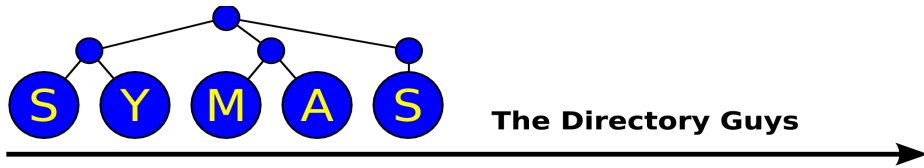
ObjectClass Mapping

- attrset Common cn,sn,uid

eid	cn	sn	uid
4	staff	(null)	(null)
5	Joe M	Mudd	joem

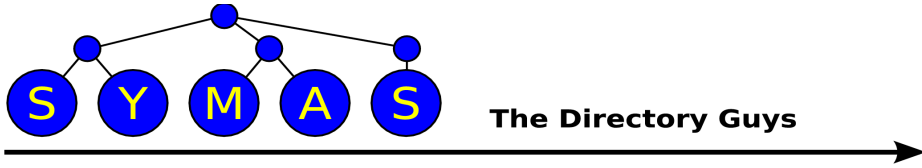
- objectClass person

eid	userPassword	telephoneNumber
5	MyGoodSecret	+1-818-555-1212



Attribute Mapping

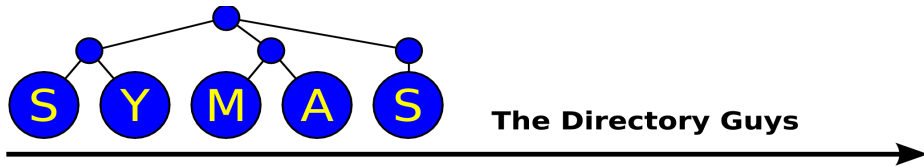
- LDAP schema imposes no size limits on schema elements, but RDBMS table columns must be of explicitly configured size
- LDAP schema allows for advisory lengths
- Backends uses advisory lengths as column size, if present
- Sizes may be explicitly configured
- Otherwise a default size of 1024 is used for DNs, 128 for everything else
- Widths of any existing columns are used as-is



Attribute Mapping

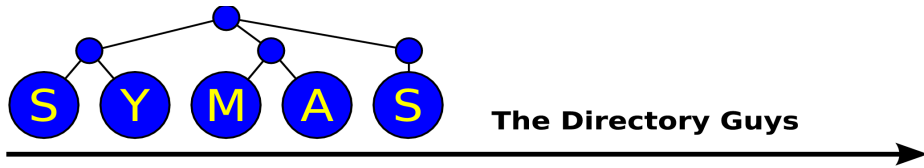
- Multi-valued attributes require a compound primary key (eid,vid)

eid	vid	cn	sn	uid
4	0	staff	(null)	(null)
5	0	Joe M	Mudd	joem
5	1	Joseph	(null)	(null)



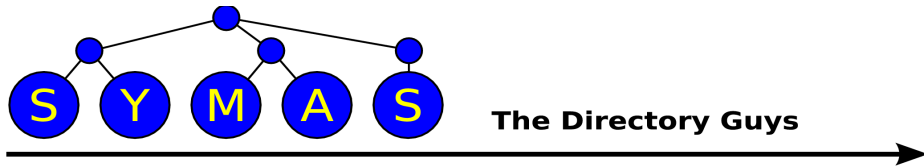
Attributes, Misc...

- Currently Attributes are stored either as VARCHARs or as BLOBs; BLOBs must be explicitly chosen in the slapd config
- NDB indexing only supports equality and inequality matching, no substring matching



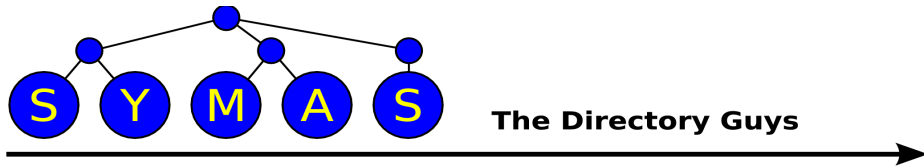
Design Wrap-Up

- The table design is minimally constrained; while NDB backend cannot be dropped in place on an existing database, the database can be adapted with minimal changes
- SQL apps are able to use the new tables as easily as before, so data can be shared directly with no duplication/waste
- Hard limits are imposed where LDAP has no limits, but most LDAP apps won't notice



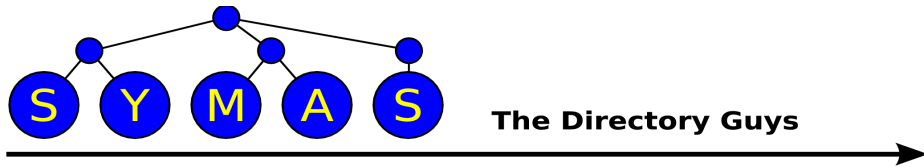
OpenLDAP implementation

Back-NDB



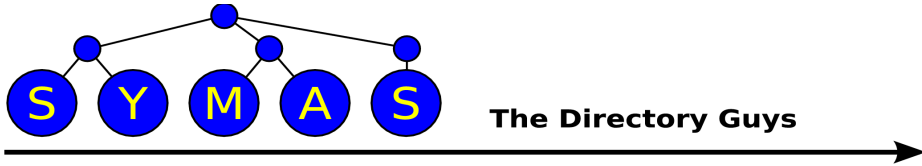
Introducing Back-NDB

- Back-NDB is a new OpenLDAP backend that uses native MySQL APIs for direct access to a MySQL NDB data store
- Released in OpenLDAP 2.4.12
- NDB is MySQL's carrier-grade cluster database engine
 - Fully transactional, scales across multiple data nodes
 - Memory-based for high performance
 - Provides automatic replication/failover



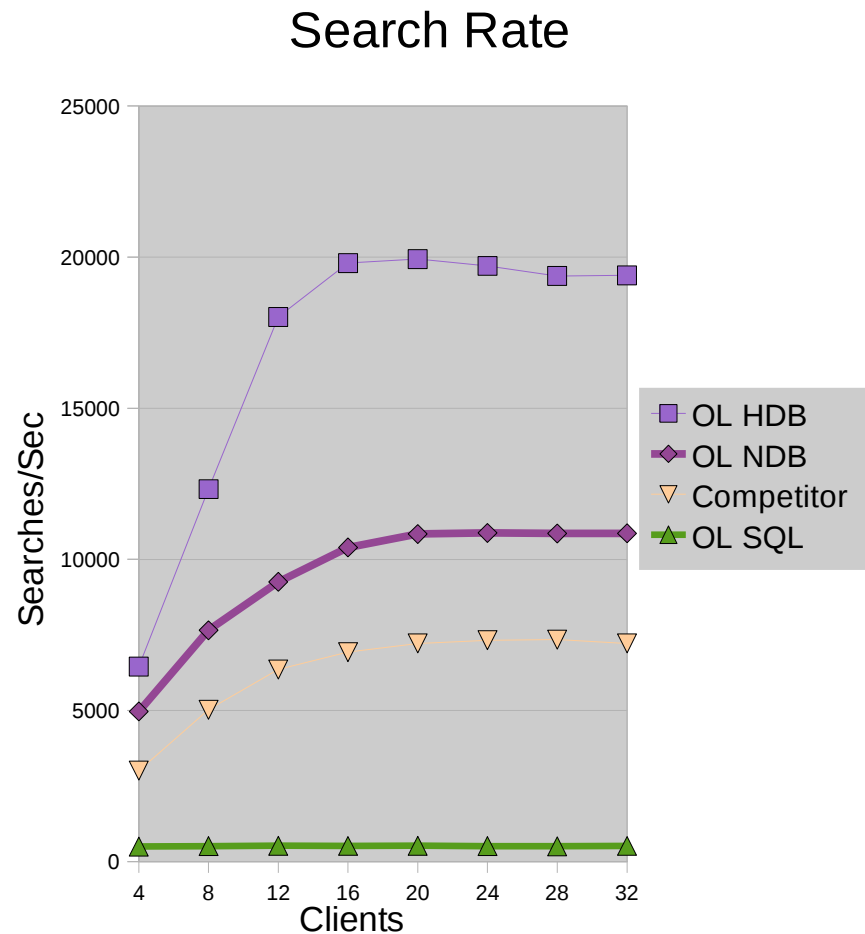
Back-NDB

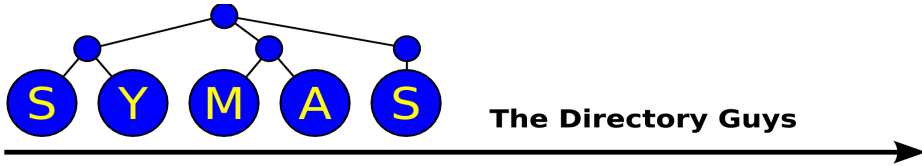
- Uses NDB APIs, bypasses ODBC and SQL layers
- Allows multiple slapd processes to operate on the same NDB databases concurrently
- Also allows multiple concurrent SQL clients
- Automatically maps LDAP schema to RDBMS schema
- Automatically detects RDBMS schema changes and maps to LDAP



Early Results

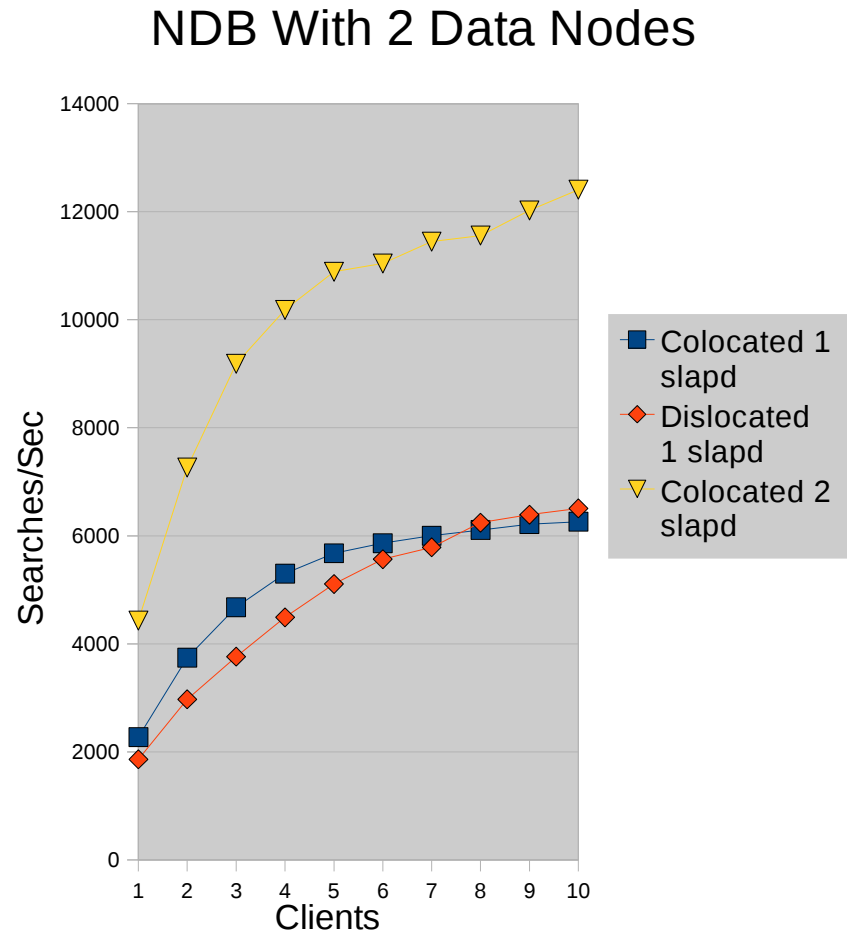
- Orders of magnitude faster than Back-SQL
- Not as fast as BerkeleyDB on a single node, but that's not the point...

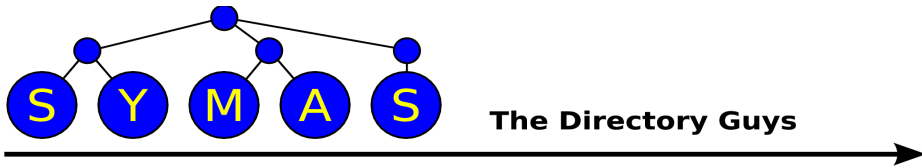




Scaling Horizontally...

- Cluster engine allows DB to be spread across multiple data nodes
- Multiple slapds can access the same DB simultaneously
- Performance scales linearly with number of nodes

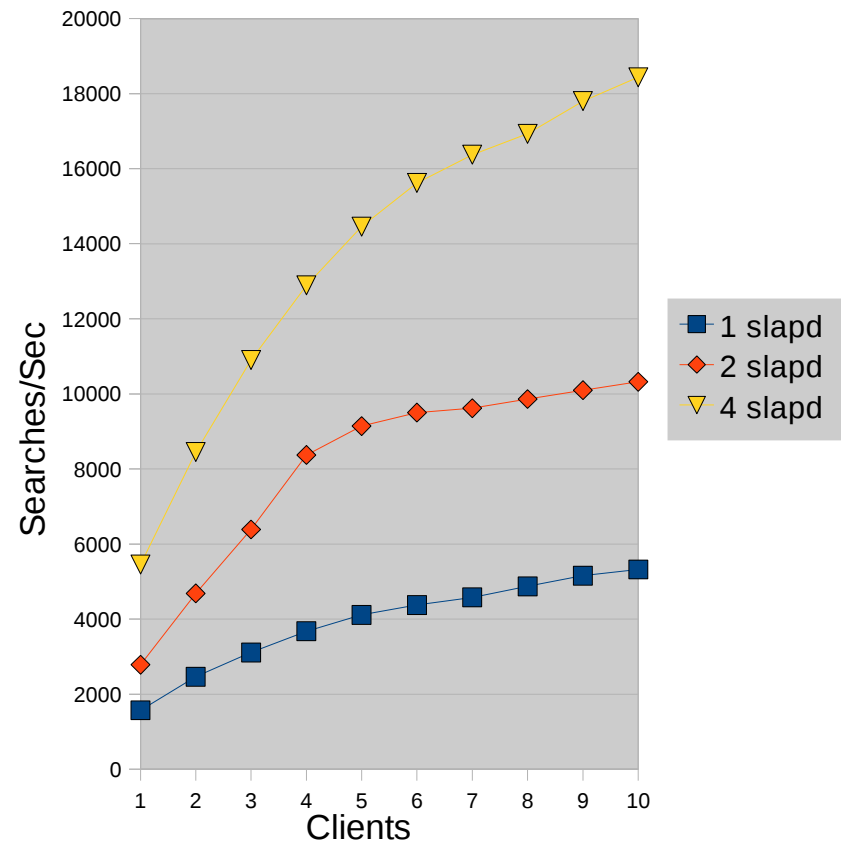




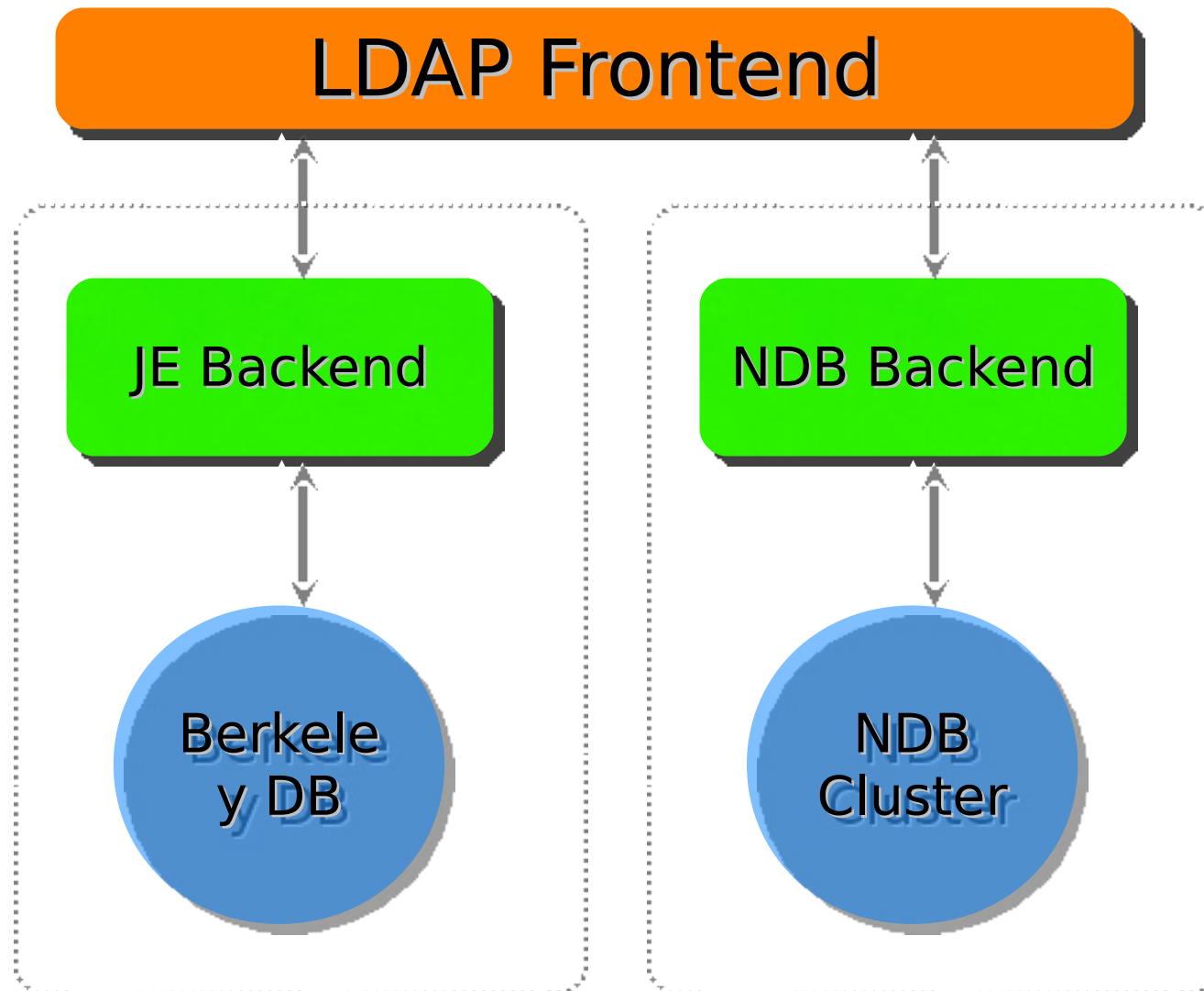
Scaling Horizontally...

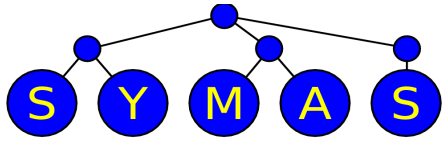
- Ideal for cluster and blade deployments
- Whenever more capacity or throughput are needed, just add more data nodes or slapd frontends

NDB With 4 Data Nodes



OpenDS Implementation





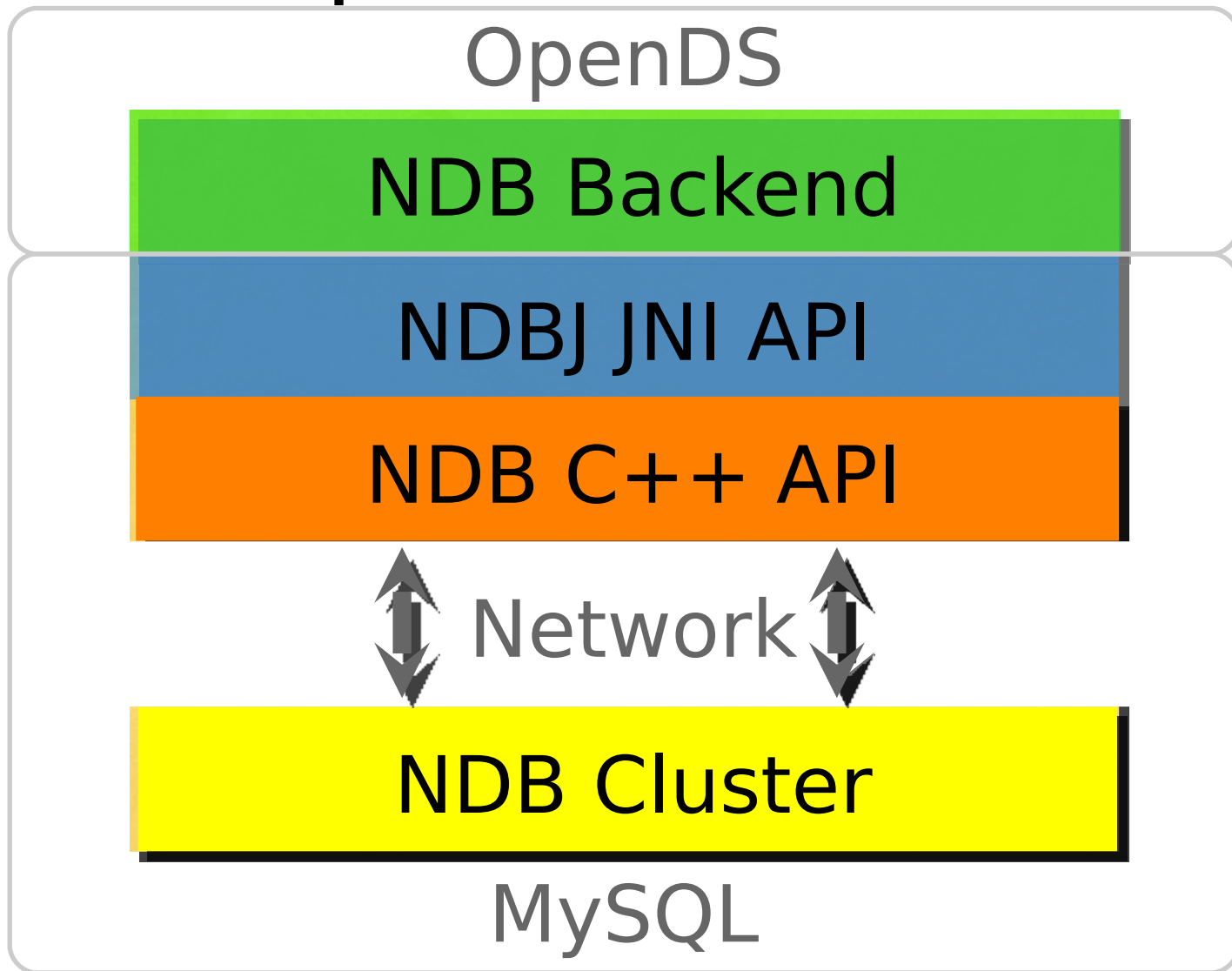
The Directory Guys

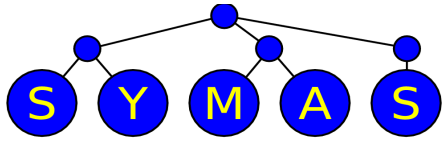


OpenDS

Open Source. Open Standards. Open Directory Service.

OpenDS Stack





The Directory Guys



OpenDS

Open Source. Open Standards. Open Directory Service.

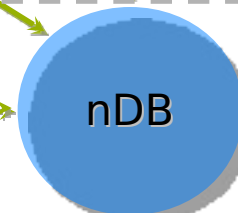
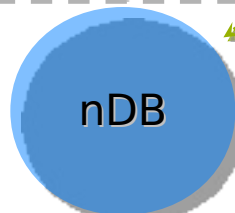
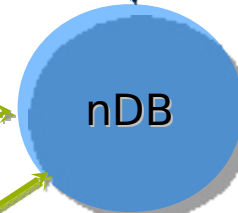
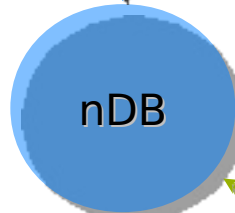
Typical Deployment

Co-Located

Co-Located

OpenDS / MySQL

OpenDS / MySQL



OpenDS / MySQL

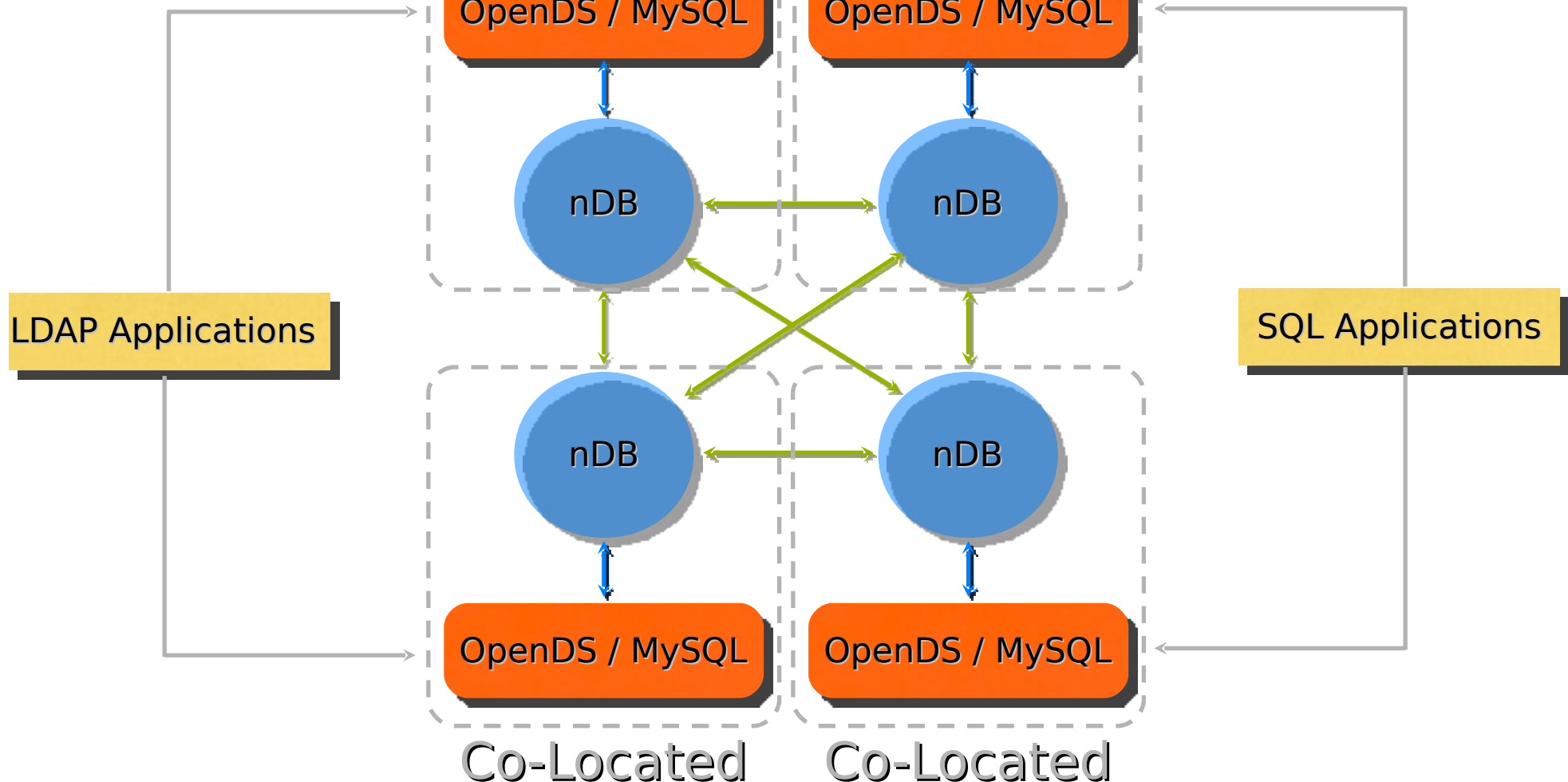
OpenDS / MySQL

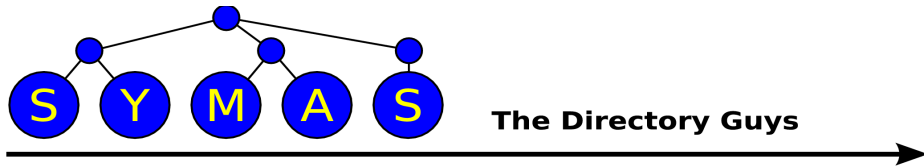
Co-Located

Co-Located

LDAP Applications

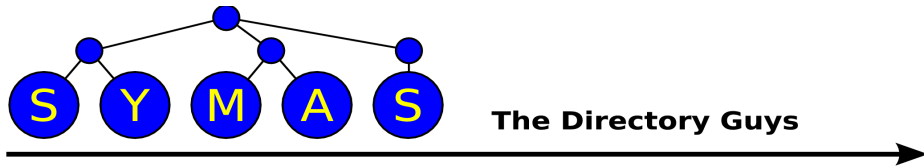
SQL Applications





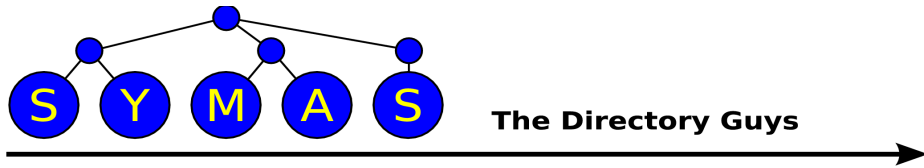
OpenDS NDB Backend

- Similar design as OpenLDAP back-ndb
- All LDAPv3 core features supported
- Scales by adding nodes
- Performances
 - Stable low latency response times
 - But some bottleneck in the NDB Java Bindings
- Some Limitations:
Entry level ACI, VLV Persistent Search,
Virtual attributes, Dynamic groups...



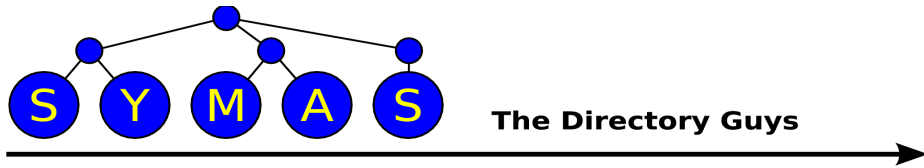
Differences with OpenLDAP NDB

- OpenDS has an extra table to deal with attribute options
- OpenDS has an additional column in the data model table for extensible objects
- There are a few differences in the naming of tables and columns



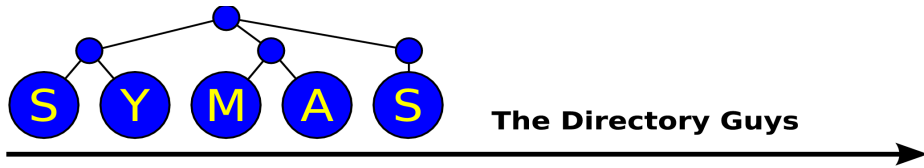
Future Directions

- Align OpenLDAP and OpenDS data models
- With a common data model, a MySQL Cluster database could be access by both servers at the same time



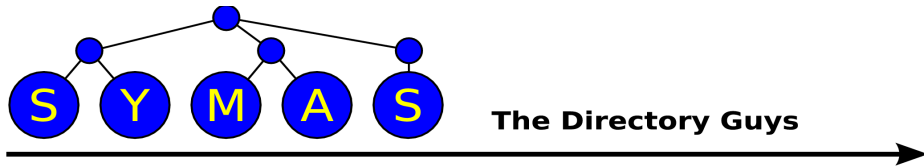
Future Directions

- Cache DN2ID table
 - Currently no local caching is done
 - Every reference to an entry requires two network roundtrips - one to the DN2ID table, and one to all of the relevant data tables
 - Reduce network roundtrips in half, double throughput



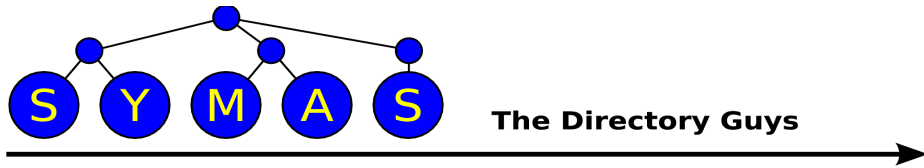
Future Directions

- Redesign DN2ID table to use HDB-style hierarchical layout
 - Increase storage efficiency - current approach wastes significant space on redundant copies of RDNs
 - Support subtree renames - current approach requires $O(n)$ time to rename a subtree; HDB style is $O(1)$



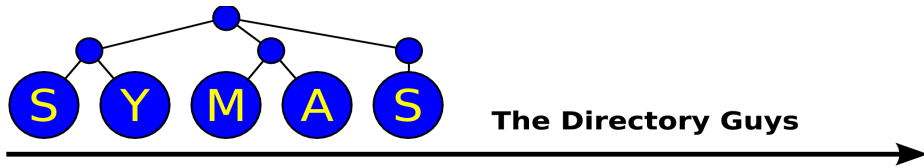
Future Directions

- Investigate possible future enhancements to MySQL
 - Support for substring indexing - currently no support at all
 - Support for consolidated filter/index processing - currently the NDB filter engine is separate from the index mechanism



Conclusion

- Growth of databases is inevitable; they never shrink
- The importance of data management and data sharing continues to increase as distributed applications proliferate
- Per-app databases are untenable as the cost of maintaining duplicate data and guaranteeing its consistency grows



Conclusion

- Admins shouldn't be forced into an either-or situation for LDAP vs RDBMS
- With OpenLDAP Back-NDB and OpenDS NDB backend, both approaches will work equally well
- OpenLDAP/OpenDS and MySQL give you the best of both worlds
- Getting started material:
<http://www.severalnines.com/blog/openldap.php>
<https://www.opensds.org/wiki/page/EnableNDBBackend>