Apache Sqoop: Highlights of Sqoop 2

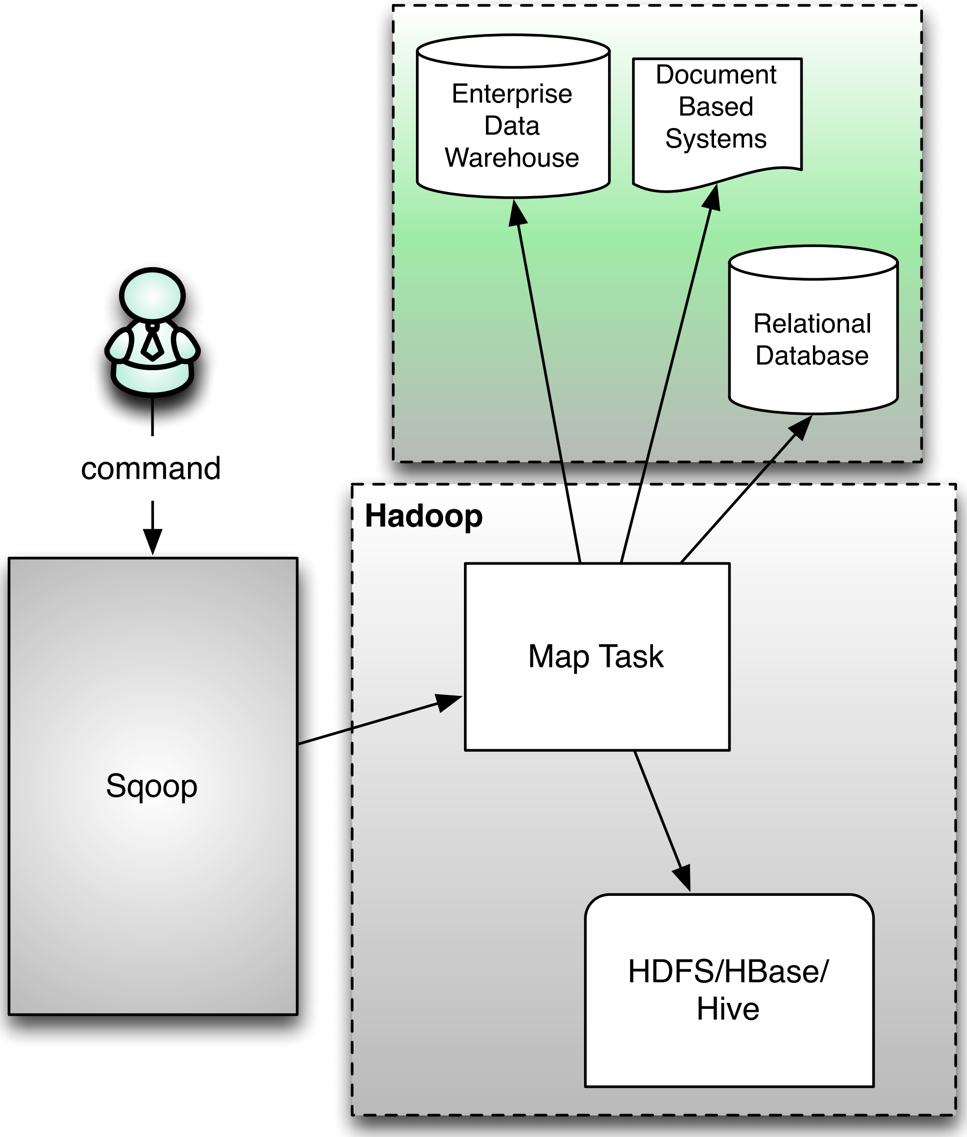
Arvind Prabhakar

Sqoop Meetup, April 2012

What is Sqoop?

- Bulk data transfer tool
  - Import/Export from relational database, enterprise data warehouse, NoSQL systems
  - Populate tables in Hive, HBase
  - Schedule Oozie automated import/export tasks
  - Support plugins via Connector based architecture

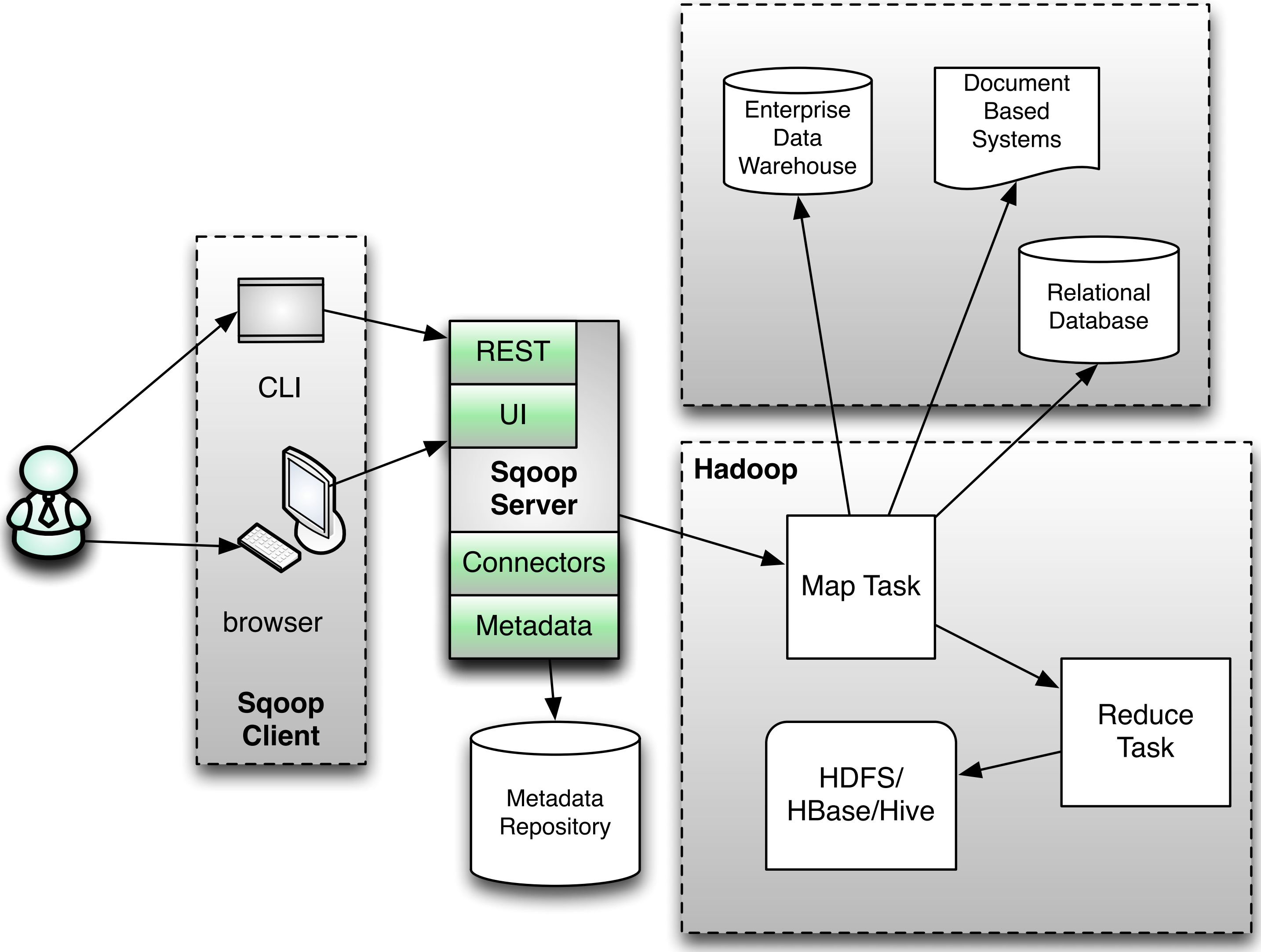
Sqoop 1 Architecture



Sqoop 1 Challenges

- Cryptic, contextual command line arguments
- Tight coupling between data transfer and serialization format
- Security concerns with openly shared credentials
- Not easy to manage config/install
- Not easy to monitor map job
- Connectors are forced to follow JDBC model

Sqoop 2 Architecture



Agenda

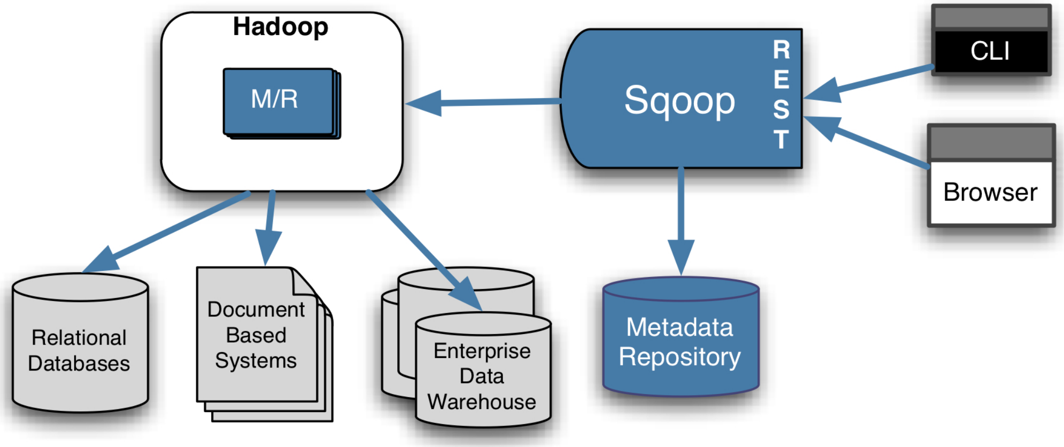
- Ease of Use
  - Sqoop 1: Client-side Tool
  - Sqoop 2: Sqoop as a Service
  - Client Interface
  - Sqoop 1: Service Level Integration
  - Sqoop 2: Service Level Integration
- Ease of Extension
  - Sqoop 1: Implementing Connectors
  - Sqoop 2: Implementing Connectors
  - Sqoop 1: Using Connectors
  - Sqoop 2: Using Connectors
- Security
  - Sqoop 1: Security
  - Sqoop 2: Security
  - Sqoop 1: Accessing External Systems
  - Sqoop 2: Accessing External Systems
  - Sqoop 1: Resource Management
  - Sqoop 2: Resource Management

Sqoop 1: Client-side Tool

- Sqoop 1 is a client-side tool
  - Client-side installation + configuration
    - Connectors locally installed
    - Local configuration, requiring root privileges
    - JDBC drivers needed locally
    - Database connectivity needed locally

Sqoop 2: Sqoop as a Service

- Server-side installation + configuration
  - Connectors configured in one place, managed by Admin/run by Operator
  - JDBC drivers in one place
  - Database connectivity needed on the server



Client Interface

- Sqoop 1 client interface:
  - Command-Line Interface (CLI) based, thus scriptable
- Sqoop 2 client interface:
  - CLI based, thus scriptable
  - Web based, thus accessible
  - REST API exposed for external tool integration

Sqoop 1: Service Level Integration

- Hive, HBase
  - Requires local installation
- Oozie
  - von Neumann(esque) integration:
    - Packaged Sqoop as an action
    - Then ran Sqoop from node machines, causing one MR job to be dependent on another MR job
    - Error-prone, difficult to debug

Sqoop 2: Service Level Integration

- Hive, HBase
  - Server-side integration
- Oozie
  - REST API integration

Ease of Use (summary)

Sqoop 1	Sqoop 2
Client-side install	Server-side install
CLI based	CLI + Web based
Client access to Hive, HBase	Server access to Hive, HBase
Oozie and Sqoop tightly coupled	Oozie finds REST API

Agenda

- Ease of Use
  - Sqoop 1: Client-side Tool
  - Sqoop 2: Sqoop as a Service
  - Client Interface
  - Sqoop 1: Service Level Integration
  - Sqoop 2: Service Level Integration
- Ease of Extension
  - Sqoop 1: Implementing Connectors
  - Sqoop 2: Implementing Connectors
  - Sqoop 1: Using Connectors
  - Sqoop 2: Using Connectors
- Security
  - Sqoop 1: Security
  - Sqoop 2: Security
  - Sqoop 1: Accessing External Systems
  - Sqoop 2: Accessing External Systems
  - Sqoop 1: Resource Management
  - Sqoop 2: Resource Management

Sqoop 1: Implementing Connectors

- Connectors forced to follow JDBC model
  - Connectors limited/required to use common JDBC vocabulary (URL, database, table, etc)
- Connectors must implement all Sqoop functionality that they want to support
  - New functionality not avail for old connectors

Sqoop 2: Implementing Connectors

- Connectors are not restricted to JDBC model
  - Connectors can define own vocabulary
- Common functionality abstracted out of connectors
  - Connectors only responsible for data transport
  - Common Reduce phase implements functionality
  - Ensures that connectors benefit from future dev of functionality

Different Options, Different Results

Which is running MySQL?

```
$ sqoop import --connect jdbc:mysql://localhost/db \
--username foo --table TEST

$ sqoop import --connect jdbc:mysql://localhost/db \
--driver com.mysql.jdbc.Driver --username foo --table TEST
```

- Different options can lead to unpredictable results
    - Sqoop 2 requires explicit selection of connector thus disambiguating the process

Sqoop 1: Using Connectors

- Choice of connector is implicit
  - In a simple case, based on the URL in the --connect string used to access the database
  - Specification of different options can lead to different connector selection
  - Error-prone but good for power users

- Requires knowledge of database idiosyncrasies
  - e.g. Couchbase doesn't need to specify a table name, which is required causing --table to get overloaded as backfill or dump operation
  - e.g. --null-string representation not supported by all connectors

- Functionality limited to what the implicitly chosen connector supports

Sqoop 2: Using Connectors

- User makes explicit connector choice
  - Less error-prone, more predictable
- User need not be aware of the functionality of all connectors
  - Couchbase users need not care that other connectors use tables
- Common functionality available to all connectors
  - Connectors need not worry about downstream functionality, transformations, integration with other systems

Ease of Extension (summary)

Sqoop 1	Sqoop 2
Connector forced to follow JDBC model	Connector given free rein
Connectors must implement functionality	Connectors benefit from common framework of functionality
Connector selection is implicit	Connector selection is explicit

Agenda

- Ease of Use
  - Sqoop 1: Client-side Tool
  - Sqoop 2: Sqoop as a Service
  - Client Interface
  - Sqoop 1: Service Level Integration
  - Sqoop 2: Service Level Integration
- Ease of Extension
  - Sqoop 1: Implementing Connectors
  - Sqoop 2: Implementing Connectors
  - Sqoop 1: Using Connectors
  - Sqoop 2: Using Connectors
- Security
  - Sqoop 1: Security
  - Sqoop 2: Security
  - Sqoop 1: Accessing External Systems
  - Sqoop 2: Accessing External Systems
  - Sqoop 1: Resource Management
  - Sqoop 2: Resource Management

Sqoop 1: Security

- Inherits/propagates Kerberos principal for the jobs it launches
- Access to files on HDFS can be controlled via HDFS security
- Sqoop operates as command line Hadoop client
- No support for securing access to external systems
  - E.g. relational database

Sqoop 2: Security

- Inherits/propagates Kerberos principal for the jobs it launches
- Access to files on HDFS can be controlled via HDFS security
- Sqoop operates as server based application
- Support for securing access to external systems via role-based access to Connection objects
  - Admins create/edit/delete Connections
  - Operators use Connections
- Audit trail logging

Sqoop 1: Accessing External Systems

- Every invocation requires necessary credentials to access external systems (e.g. relational database)
  - Workaround: Admin creates a limited access user in lieu of giving out password
    - Doesn't scale
    - Permission granularity is hard to obtain
- Hard to prevent misuse once credentials are given

Sqoop 2: Accessing External Systems

- Sqoop 2 introduces Connections as First-Class Objects
  - Connection encompass credentials
  - Connections created once, then used many times for various import/export Jobs
  - Connections created by Admin, used by Operator
    - Safeguard credential access from end user
- Restrict scope: connections can be restricted based on operation (import/export)
  - Operators cannot abuse credentials

Sqoop 1: Resource Management

- No explicit resource management policy
  - User specifies number of map jobs to run
  - Can’t throttle load on external systems

Sqoop 2: Resource Management

- Connections allow specification of resource policy
  - Admin can limit the total number of physical Connections open at one time
  - Connections can be disabled

Security (summary)

Sqoop 1	Sqoop 2
Support only for Hadoop security	Support for Hadoop security and role-based access control to external systems
High risk of abusing access to external systems	Reduced risk of abusing access to external systems
No resource management policy	Resource management policy

Takeaway

Sqoop 2 Highights:

- Ease of Use: Sqoop as a Service
- Ease of Extension: Connectors benefit from shared functionality
- Security: Connections as First-Class objects, Role-based Security

Current Status: work-in-progress

- Sqoop 2 Development: https://issues.apache.org/jira/browse/SQOOP-365

- Sqoop 2 Blog Post: https://blogs.apache.org/sqoop/entry/apache_sqoop_highlights_of_sqoop

- Sqoop 2 Design: https://cwiki.apache.org/confluence/display/SQOOP/Sqoop+2