Deploying and Scaling Massive Digital Archive Repositories

Sacha Arnoud

Architect
Sun Microsystems, Inc.
http://www.sun.com

TS-19460

Deploy and Make Use of A Petabyte-Scale Digital Repository

Go through the steps of designing, implementing, and deploying a real-life, petabyte-scale digital repository using Java™ technologies.

Agenda

1. Is fixed data important?
2. Let’s go and build our digital repository!
   - What components do I need?
   - Which existing technologies can I reuse?
   - How do I design and grow my storage?
3. Did I get something wrong? …or how real life can bite you
4. Case Study: fedora.info and Sun StorageTek™ 5800

Agenda

1. **Is fixed data important?**
2. Let’s go and build our digital repository!
   - What components do I need?
   - Which existing technologies can I reuse?
   - How do I design and grow my storage?
3. Did I get something wrong? …or how real life can bite you
4. Case Study:
   fedora.info and Sun StorageTek™ 5800

What’s Fixed Data ?

What’s Fixed Data ?

Fixed content data

Reference data

What’s Fixed Data ?

Fixed content data

Reference data

Archival data

What’s Fixed Data ?

Fixed content data

Reference data

Archival data

Basically any data that never gets modified

What’s Fixed Data ?

Fixed con
data

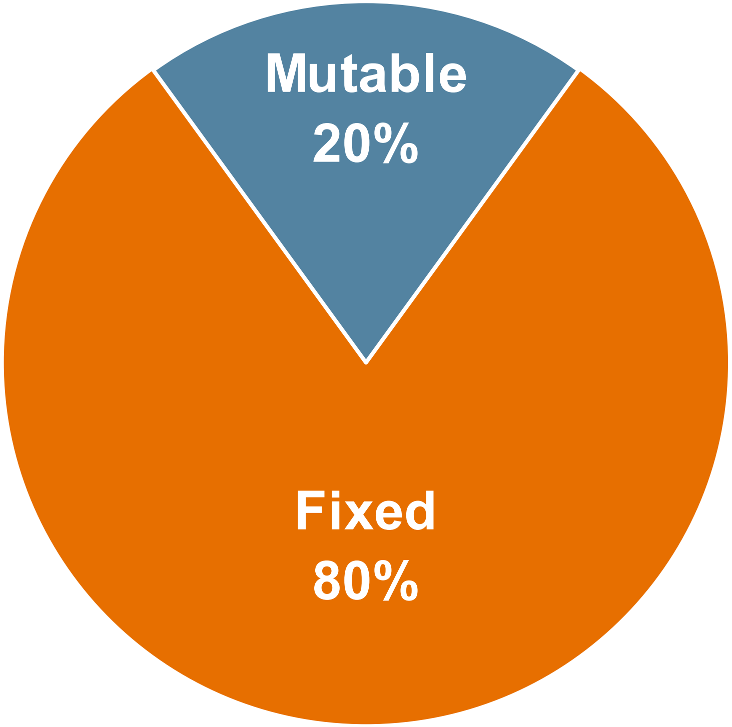
Reference
data

Archival
data

Is it a lot of data?

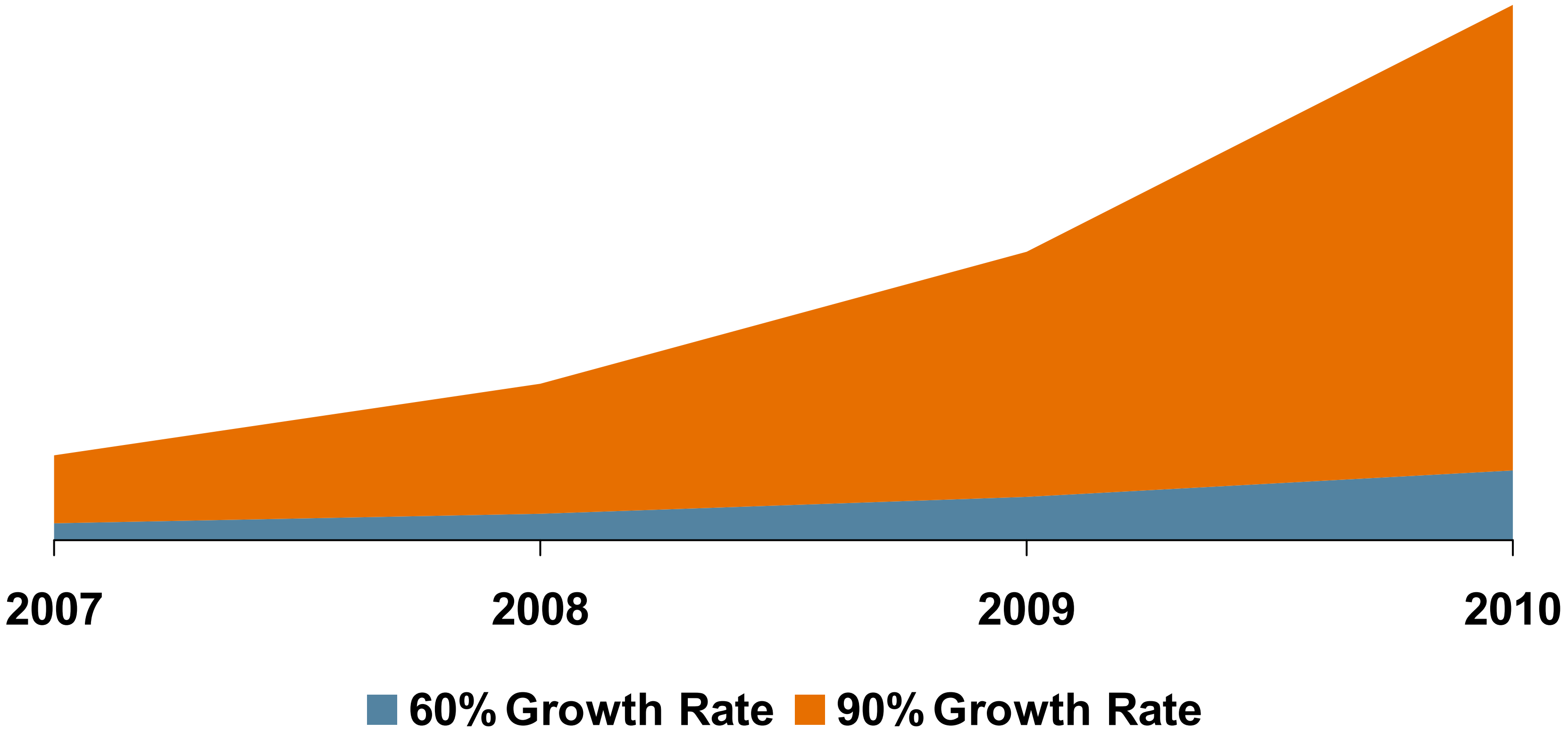
Basically any data that never gets modified

Yes !



Source: Hal Varian, UC Berkeley

And Even More Tomorrow



Reference Data Is Everywhere

- Most of the new digital content is born fixed
  - Video (1 million YouTube uploads per day[1])
  - Pictures (top sites generate multiple PB a year)
    - Flickr, Kodak EasyShare Gallery, …
  - Commodity storage (xdrive, box.net, Amazon S3, …)
  - Digital libraries (all major universities)
  - Healthcare industry
  - …
- Through the archival process mutable data becomes fixed
- Federal regulations

[1] Wired, December 2006, “The Rise of YouTube,” p. 22.

Let’s go, build, and use our own digital repository, right now!

Agenda

1. Is fixed data important?
2. Let’s go and build our digital repository!
    - **What components do I need?**
    - Which existing technologies can I reuse?
    - How do I design and grow my storage?
3. Did I get something wrong? …or how real life can bite you
4. Case Study:
   fedora.info and Sun StorageTek 5800

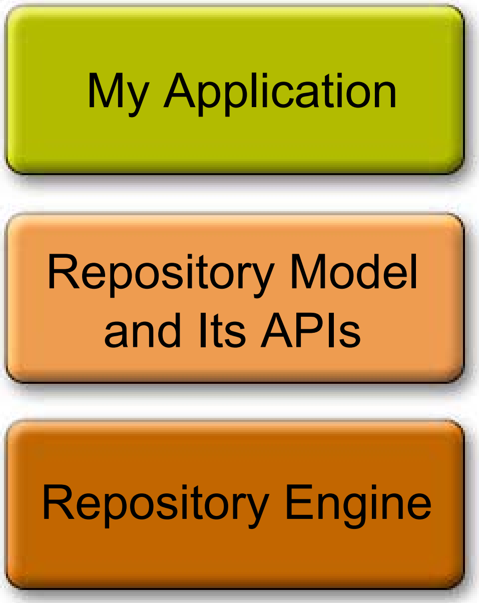
What Components Do I Need?

My Application

You still need to think about your application

- What kind of data do you generate?
- What data model for your repository?

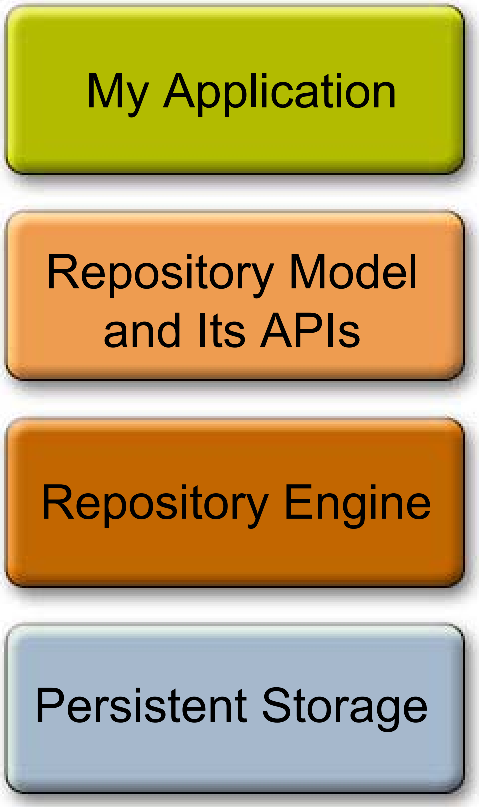
What Components Do I Need?



Tap into existing digital repository solutions

- Look at their data models
- Their APIs
- And a lot of them are based on Java technology!

What Components Do I Need?



My Application

Repository Model and Its APIs

Repository Engine

Persistent Storage

- What storage APIs do these implementations rely on?
- Do you have to deploy a database?
- Can you change the repository engine to use other storage solutions?

Agenda

1. Is fixed data important?
2. Let's go and build our digital repository!
   - What components do I need?
   - **Which existing technologies can I reuse?**
   - How do I design and grow my storage?
3. Did I get something wrong? …or how real life can bite you
4. Case Study:
   fedora.info and Sun StorageTek 5800

DSpace

- http://dspace.org/
  - MIT Libraries
  - Hewlett-Packard Labs
- Emphasizes the curation of digital content
- The reference implementation relies on:
  - A filesystem
  - Tomcat
  - A relational DB (PostgreSQL, Oracle)

fedora.info

- http://fedora.info/
  - Cornell University Information Science
  - University of Virginia Library
- Based on objects that encapsulate:
  - Data streams;
  - Metadata;
  - Code that implements behaviors (disseminators)
- The reference implementation relies on:
  - A filesystem
  - Tomcat
  - A relational DB (MySQL or mckoi)
- VTLS is a commercial offering on top of Fedora

Content Repository for Java Technology API (JSR 170)

- http://jcp.org/aboutJava/communityprocess/review/jsr170/index.html
- Lower level
  - Requires a higher level application to implement a workflow
- Documents are nodes in a hierarchical structure
  - With data
  - And properties (metadata)
- R.I. is Apache Jackrabbit
  - http://jackrabbit.apache.org/
- Day software is a compliant commercial implementation
  - http://www.day.com/site/en/index.html

JSR = Java Specification Request

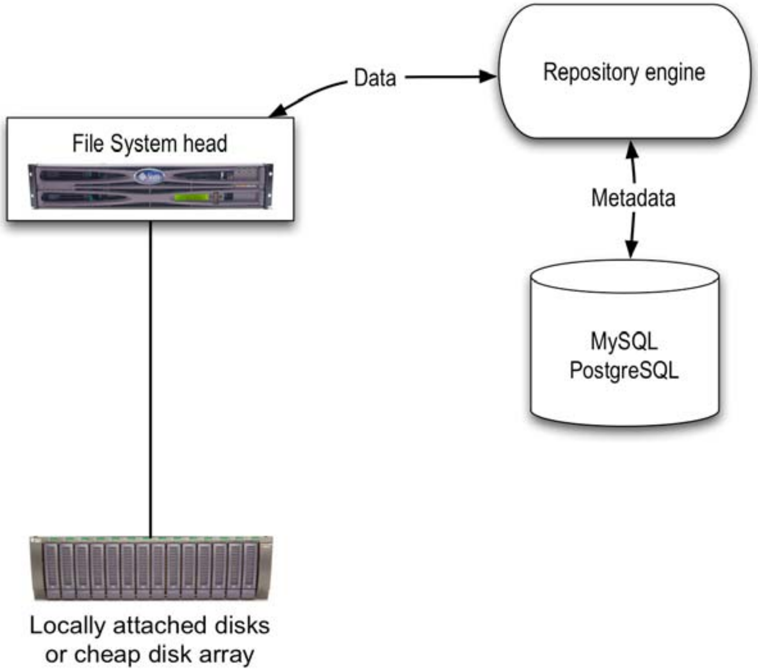
XAM (FCAS TWG-SNIA)

- http://www.snia.org/xam/home
- Focuses on a abstraction at the storage layer
  - SNIA standard
- Demonstrates the industry commitment to fixed content storage
- Work on both a C and a Java API

Agenda

1. Is fixed data important?
2. Let’s go and build our digital repository!
   - What components do I need?
   - Which existing technologies can I reuse?
   - **How do I design and grow my storage?**
3. Did I get something wrong? …or how real life can bite you
4. Case Study:
   fedora.info and Sun StorageTek 5800

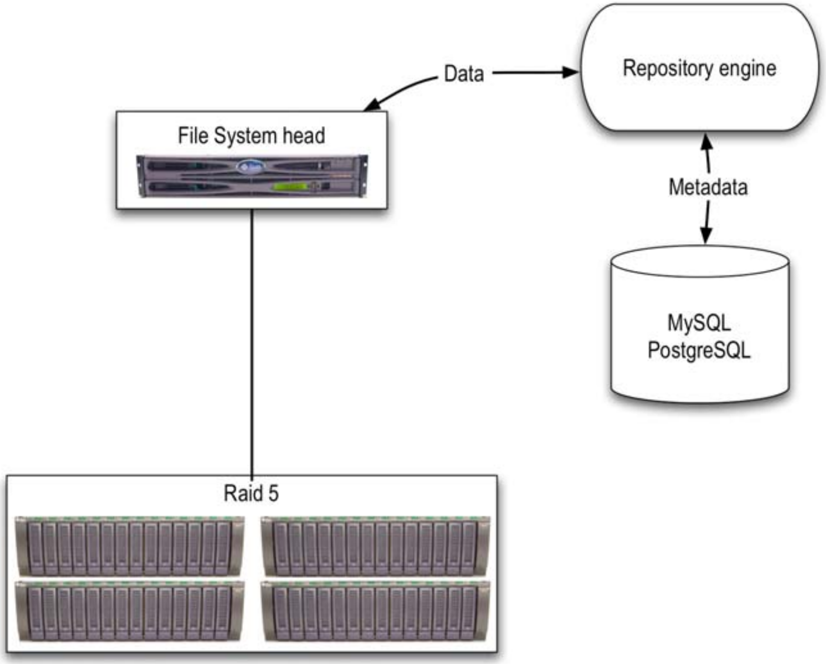
Let’s Build Our Storage Solution



Use simple components required by most R.I.:

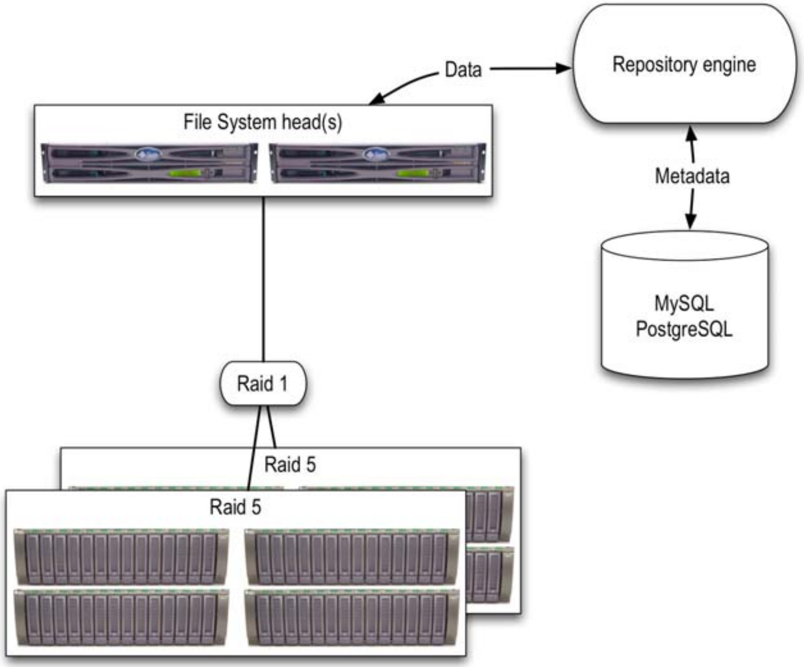
- Directly attached diska
- An open source database

We Need More Storage!



We all know the virtues of RAID!

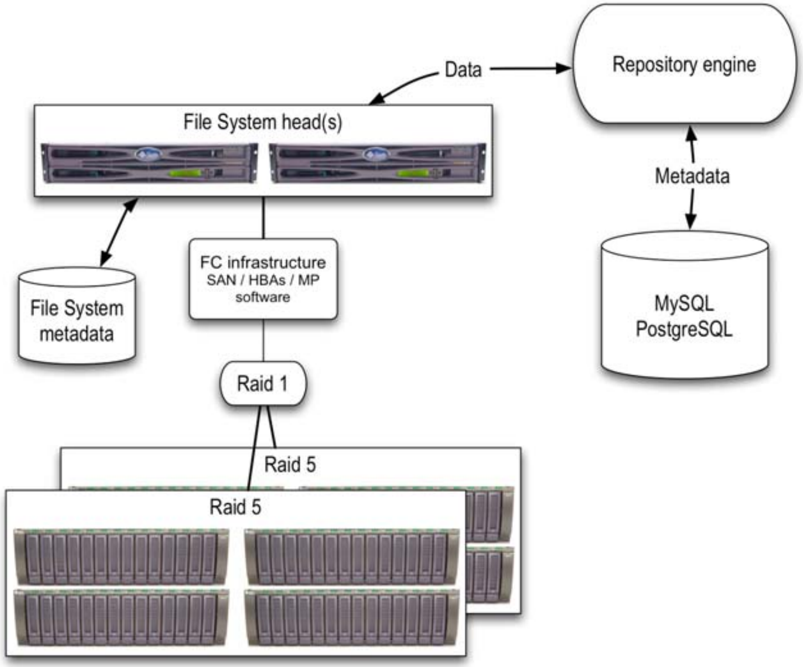
Even More Storage



Keep adding cheap DATA disks

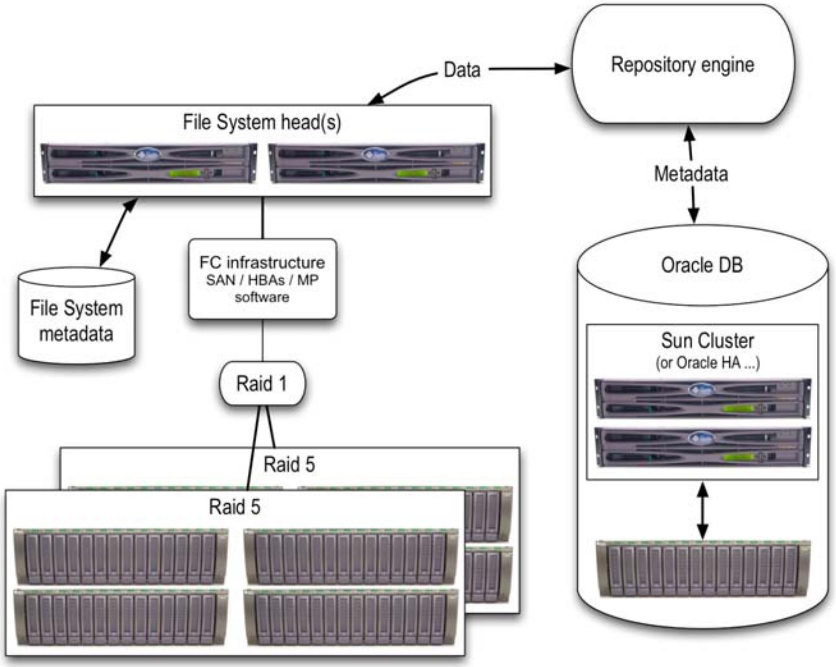
- Do RAID 5.1 for reliability
- Put a distributed FS for availability and performance

I Need to Scale Better



- Switch to fiber channel
- Switch to a high-performance FS

And What About the Database ?



Switch to a distributed DB:

- Oracle DB
- Scale the DB storage

Agenda

1. Is fixed data important?
2. Let’s go and build our digital repository!
   - What components do I need?
   - Which existing technologies can I reuse?
   - How do I design and grow my storage?
3. **Did I get something wrong? …or how real life can bite you**
4. Case Study:
   fedora.info and Sun StorageTek 5800

I Just Built a Complex Solution

- Expensive to acquire
  - Fiber channel infrastructure
  - RAID 5.1
  - Costly RDBMS license
- Technically advanced
  - High-performance filesystem
  - Distributed filesystem
- Managing it is complex

And I Am Not Using It!

- My data is never modified
    - Why paying the price of a consistent distributed FS?
- I probably don’t need a HPC kind of performance
- I am not really using the FS hierarchical structure

But I need to:

- Scale my application with my storage;
- Get reliability and high-availability to my data at a low cost per GB

Maybe I Chose the Wrong Model?

- Storage is evolving
  - From block-based storage
  - To file- and filesystem-based storage
  - To objects
- A storage object
  - Has data
  - Has searchable metadata
  - Has code that can be executed against the data
- It is up to the repository model to organize them logically!
  - Using enhanced new generation storage capabilities

How Does That Affect Me?

- If you develop digital archives frameworks
  - Abstract storage using an object based approach
  - Don’t use **`java.io.File`**
  - Stay tuned and expect new Java technology standards in that space
- If you develop digital archive applications
  - Use an existing framework that will do the abstraction work for you
  - Focus on your workflow, business logic, …

Agenda

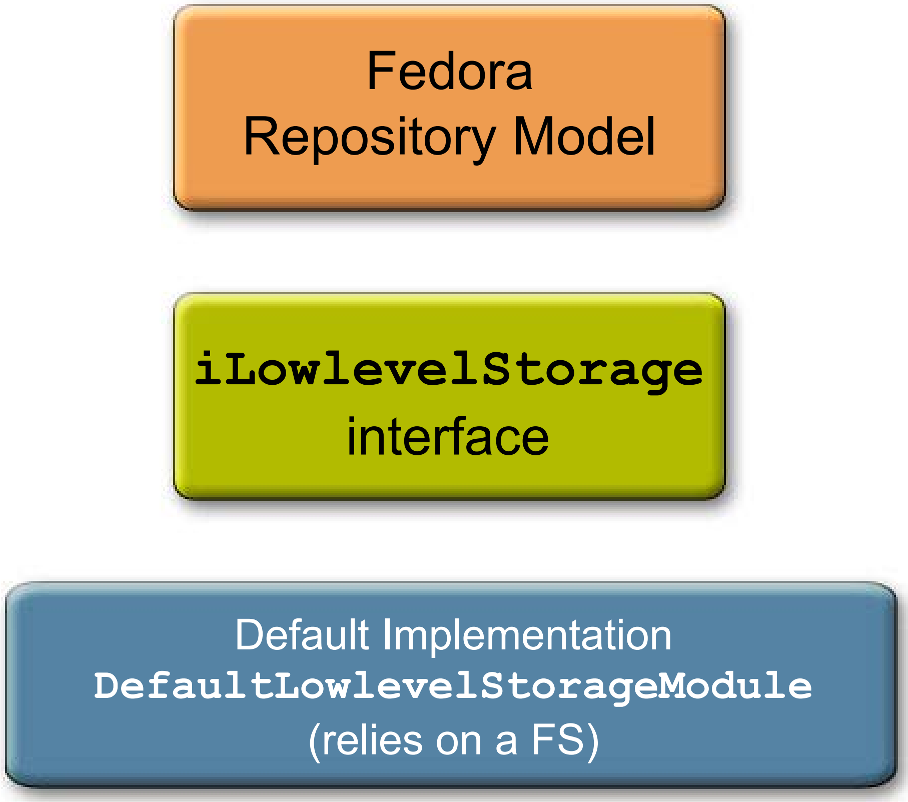
1. Is fixed data important?
2. Let’s go and build our digital repository!

   What components do I need?

   Which existing technologies can I reuse?

   How do I design and grow my storage?
3. Did I get something wrong? …or how real life can bite you
4. **Case Study: fedora.info and Sun StorageTek 5800**

The Fedora Stack



Fedora
Repository Model

`iLowlevelStorage`
interface

Default Implementation
`DefaultLowlevelStorageModule`
(relies on a FS)

Sun StorageTek 5800

- a.k.a. Honeycomb
- Fixed content, object-based storage
  - Cheap
  - Reliable
  - Available
  - Scalable
  - With metadata support, including search
- Check out booth **POD-986**

StorageTek 5800 APIs

- Built-in Java APIs
- Store

```
objectHandler =
   storeObject(
   java.nio.channels.ReadableByteChannel,  # Data
   java.util.Map);                          # Metadata
```

- Retrieve

```
retrieveObject(objectHandler,
    java.nio.channels.WritableByteChannel);
```

- Query

```
objectHandler List = query(String);
```

Map Fedora Objects Into Storage Objects

- Fedora objects have a pid (string)
- Each object in storage has a metadata field

```
fedora.pid = “my value”;
```

- To retrieve, first lookup the fedora pid in storage

```
object handle = query(“fedora.pid=‘my value’”);
retrieve(object handle);
```

- Fedora FOXML metadata can be pushed to the storage

  …and the storage can take care of queries

Example: addObject

```
public void addObject(String pid,
                      InputStream content)
       throws LowlevelStorageException {
       LOG.info("HCStorage: store ["+pid+”]");
       try {
           NameValueRecord record = oa.createRecord();
           record.put(PID_FIELD_NAME, pid);
           record.put(TYPE_FIELD_NAME, type);
           SystemRecord res =
  oa.storeObject(Channels.newChannel(content),
                                             record);
           LOG.info("HCStorage: stored object”);
       } catch (…) { }
```

Summary

- The explosion of fixed content data applications is a huge opportunity
- Most (all?) existing solutions heavily rely on Java technology
- Storage and Java technology are going to integrate more!
- Sun Microsystems is committed to make both integrations converge

…and Sun is committed to open source

Stay tuned!

For More Information

Sun StorageTek 5800

http://www.sun.com/storagetek/disk_systems/enterprise/5800/index.xml
http://www.sun.com/emrkt/innercircle/newsletter/0606edchoice_nontrad.html

- **Check out Booth POD-986**

fedora.info

http://fedora.info/

The expanding digital universe

http://www.emc.com/about/destination/digital_universe/pdf/Expanding_Digital_Universe_IDC_WhitePaper_022507.pdf

Q&A

Deploying and Scaling Massive Digital Archive Repositories

Sacha Arnoud

Architect
Sun Microsystems, Inc.
http://www.sun.com

TS-19460