RESTful Filesystems

James Westland Cain, Ph.D.
Principal Software Architect
Quantel Limited

Menu

- Who are Quantel?
- What is REST?
- REST in HTTP
- REST in a FileSystem?
- Demos
- Conclusion – RESTful Filesystems

Who are Quantel?

- You have heard of our broadcast customers:
  - BBC
  - ESPN
  - DirectTV
  - Fox Sports
- You have seen some films Quantel helped to make:
  - Avatar
  - Lord of the Rings
  - Star Wars

Quantel's Object Model

- Pictures and Sound are immutable
- Quantel has a reference counted scheme in the file system
- Quantel adds ids to all media elements in the file system
- Edit logical descriptions of runs of pictures & sound using the identities
- Simple editing Demo

What is REST?

- Representation State Transfer (REST)
  - Representations of resources
- An architecture
  - Invented by Roy Fielding during his Ph.D.
  - Roy is a principal author of HTTP 1.0 & 1.1
- It is NOT RPC
  - The client and server are not tightly coupled

What is REST?

- A set of principles
  - Resources
  - Self contained requests
  - HATEOAS – Hypermedia as the engine of application state

What is REST?

- A set of constraints
  - Client – Server
  - Servers are stateless
  - Replies are cacheable
  - Servers can be layered
  - Code on demand (optional)
  - Uniform Simple API for CRUD
    - Eg in HTTP: GET, POST, PUT, DELETE

Benefits of REST

- Supports Redundancy
- Massively Scalable
- Loosely Coupled
- Services are composable
- Supports massive growth and client freedom

REST in HTTP

- Browsers follow paths to resources
- Paths contain ids of the resources
- Browsers negotiate MIME types to choose representation
- HTML / XML contain links (HATEOAS)
- Links contain IDs to other resources
- The user choosing a link is a change in application state

REST in a FileSystem?

- Only a partial fit
  - Large bespoke APIs
  - More like RPC
- Question: is there a useful subset?
- What properties of a filesystem can be retained whilst support a useful subset of REST?

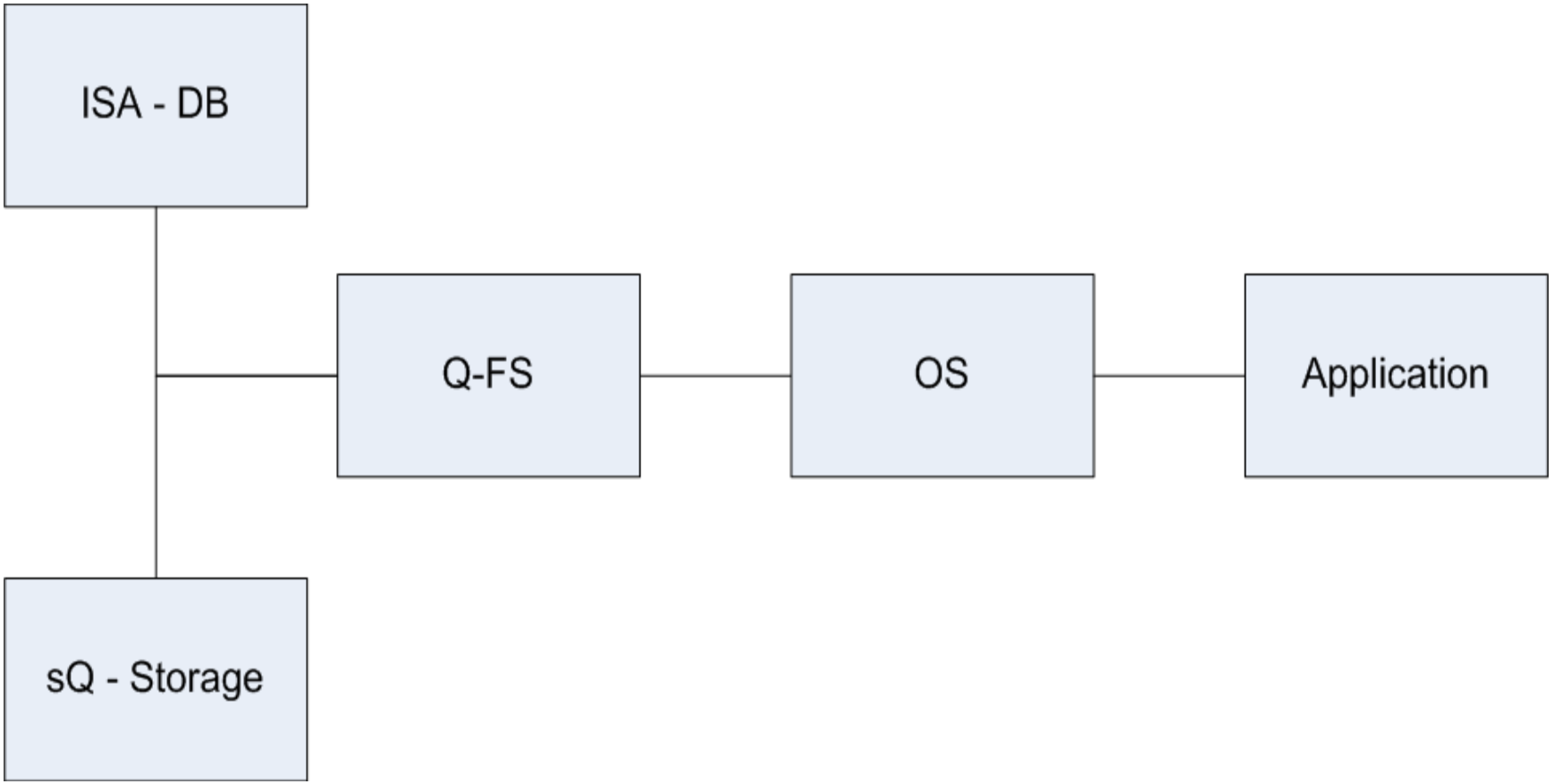
REST in a FileSystem?

- Properties of DAS / SAN / NAS world.
  - Speed per node
  - Many standards
  - Limited scale (not easily composable)
- Representation of resources
  - Implies file / folder understanding
  - Therefore use NAS

DEMO 1

- Resource Identity
  - Folders are normally a way to discover immutable copies of pre-formatted files
  - Cifs / SMB (and other NAS protocols) do not stipulate this
  - WARNING: Were not in Kansas anymore!

Tech Description



Q-FS

- Quantel have implemented a small subset of Cifs
  - Simple Files and Folders
  - Engineered for high speed – no memcpys, uses IOCompletionPorts
  - No AD, no signing, very little else!
- It runs on Windows
- It replaces the built in Windows SMB server

DEMO 2

- Self contained requests
- Follow a path containing a resource identity
- What form should the representation of the resource take?
- HTTP uses MIME types
- Cifs uses file extensions

DEMO 3

- HATEOAS
- Embedding resource locators in files such that application state changes are communicated to the server.

Problems

- The NAS protocols are too broad for full REST constraint support (c.f. WebDAV)
- CIFS supports shared editing using OpLocks
- CIFS is very stateful
- CIFS clients do not easily failover

Problems - Mutation

- Mutation
  - Locking is stateful (ask NFS!)
  - Don't support shared editing
    - No external update!
  - Use reference counting to make all files "read only"
- Create, Read & Delete
- No Shared Update in the file system

Problems – Shared Views

- Applications will update files – Cifs supports it.
- Give users their own sandpit
  - Per user views composed on top of shared immutable files.
  - Breaks REST cachability – in a transient manner.

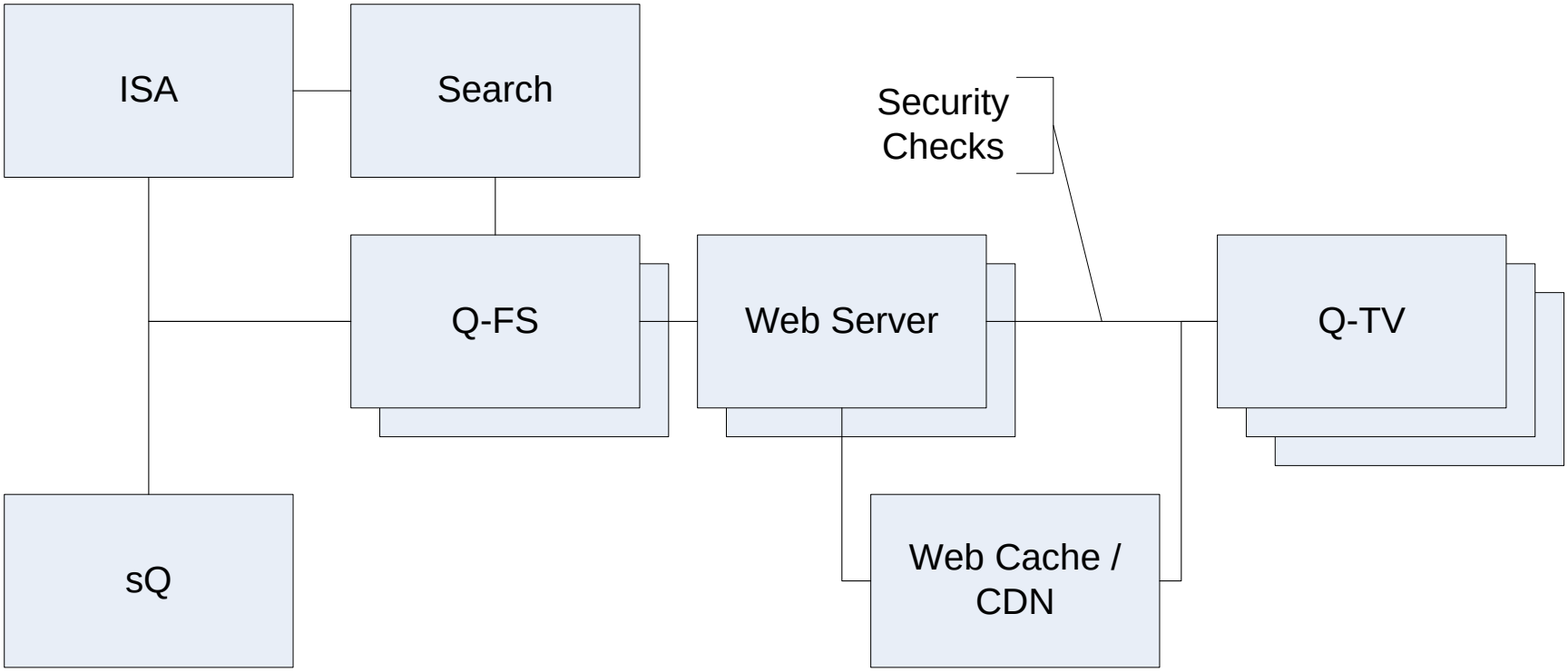
DEMO 4

- Creating new files
  - Translates to creating new REST resources
  - Use a private view while the resource being created

Code on Demand

- RIAs run anywhere
- Quantel uses RIAs to supply application glue – simple search apps that allow 3rd party COTS Thick clients to integrate
- Used Flex to add support for Apple FCP
- Used Silverlight to support SmoothStreaming

DEMO – Remote Viewing



REST in a Filesystem - Analysis

- Architectural Principles apply
- Constraints Partially honoured:
  - ☑ Client – Server
  - ☑ Servers are stateless
  - ☑ Replies are cacheable
  - ☑ Servers can be layered
  - ☑ Code on demand (optional)
  - ☑ Uniform Simple API for CRD
- But not for Updates!

Conclusion

- These techniques can be used to allow workflows exploiting all modern OS's – Windows, Linux, OS-X.
- A constrained form of REST can be supported in a filesystem
- The constraints probably are onerous for generic storage
- The constraints are fine for the media production industry