Goldman Sachs

Java at Goldman Sachs

Sep 14th , 2011

A Consumers perspective of the Java Platform
for the JCP EC Face to face Meeting

Goldman Sachs

Agenda

- Technology at Goldman Sachs
  - Brief Introduction / Scene setting
- Java Engineering Function
  - Overview of role and challenges
- A Typical platform
  - Overview of our Middle office platform
- Questions / Discussion

Goldman Sachs

Java Engineering

- Who we are?
- Enterprise Java Challenges.
- What do we do?
- Items that would impact us.

Goldman Sachs

Who are we?

- Java Engineering provides support and consulting services to the firm on the Java stack
- Staff mostly former JVM engineers (e.g from IBM and SGI).
- We work with the JVM source code on a daily basis.
- Good contact with technologists at Vendors (e.g. Oracle and Red Hat).

Goldman Sachs

Enterprise Java challenges

- **Much of the firm's mission-critical processing is in Java**
- **Goldman Sachs pushes the limits of the JVM**
  - Routinely uncover never-before-seen bugs in the JVM
  - Mostly associated with extreme heap size or scaling
  - Breadth of language feature exploitation finds corner cases
- **Java must interoperate well with other languages**
  - C++, .Net, Proprietary languages, Scripting languages
- **Managing multiple instances of Java apps across the globe**
  - Timezone update:
    - How do you manage timezone update across the firm?
    - Why don't we use OS timezone data instead?
  - Date and Time API needs a overhaul: JSR-310 is important
- **Security Vulnerability Management**

Goldman Sachs

What we do

- **We cover the following technologies**
  - Java SE: Java Virtual Machines, Class libraries
  - Java EE: Java Application Servers, Middleware (e.g. Hibernate)
  - In house developed frameworks
- **Life cycle management**
  - Evaluate and enable adoption of new Java technology within GS
  - Security Vulnerability management
- **Performance tuning**
  - Working with application and infrastructure teams to maximize Java performance
- **Production support**
  - Analyzing crashes, providing workarounds, raising issues
- **Educating developers**
  - Best practices, tools, internal and external classes

Java deployment in Goldman Sachs

- **Heterogeneous environment**
  - Java on Windows and Linux
    - Windows Desktop and Server
    - RHEL 4, 5, 6
- **Java updates do cause issues**
  - Source compatibility is important
    - e.g. JDBC compilation issues
  - Binary compatibility
- **Certification suite to test GS-like environments**
  - Performance and scalability testing
  - Bug regression
  - Compatibility
- **Java 7 rollout**
  - Engaging, educating and supporting early adopters
  - Promoting the release on runtime and language improvements

Goldman Sachs

Supporting GS Java

- **What makes our life easier**
  - Accurate, complete, documentation
  - Source code access and developers' debug code
  - Discussions with developers
  - Bug reports and trackers
  - Standards
  - Debugging, monitoring, maintenance tooling
  - Logs – especially GC and JIT
  - Tracers, analyzers, profilers and visualizers
- **What makes our life harder**
  - Lack of any of the above
  - Extreme market conditions

Q&A

Goldman Sachs

HYDRA – A Java "Big App"

Goldman Sachs Operations Technology *September 2011*

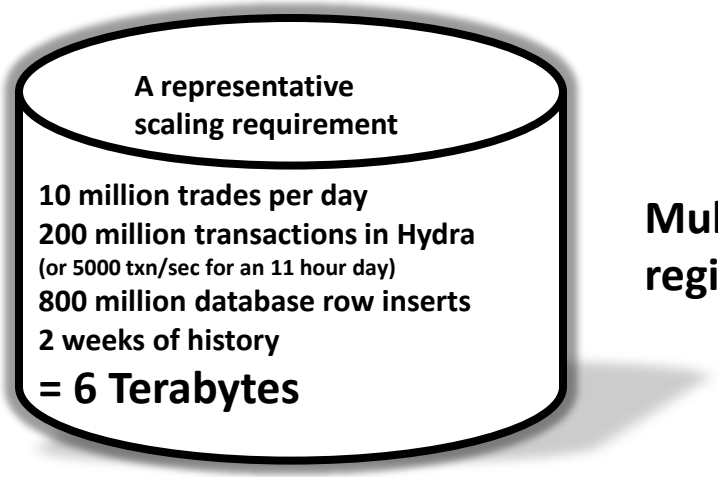
Hydra Requirements

Technology Requirements

A Middle Office Post Execution Trade Processing Platform which is ...

- **Available**
  Support business globally across a 24 hour x 5+ trading environment.
- **Reliable**
  Reliability and high uptime for timely processing of many of the firms flows.
- **Performant**
  Key goal to support high volumes within standard processing windows.
- **Agile**
  Fast, STP based processing with exception management to provide enhanced client experience and reduce firm risk.
- **Auditable**
  Clear and unambiguous audit of transaction lifecycle and processing with historic record.
- **Scalable**
  Support performance and capacity scalability to accommodate growing volumes and new markets.

What makes it a "Big App"?



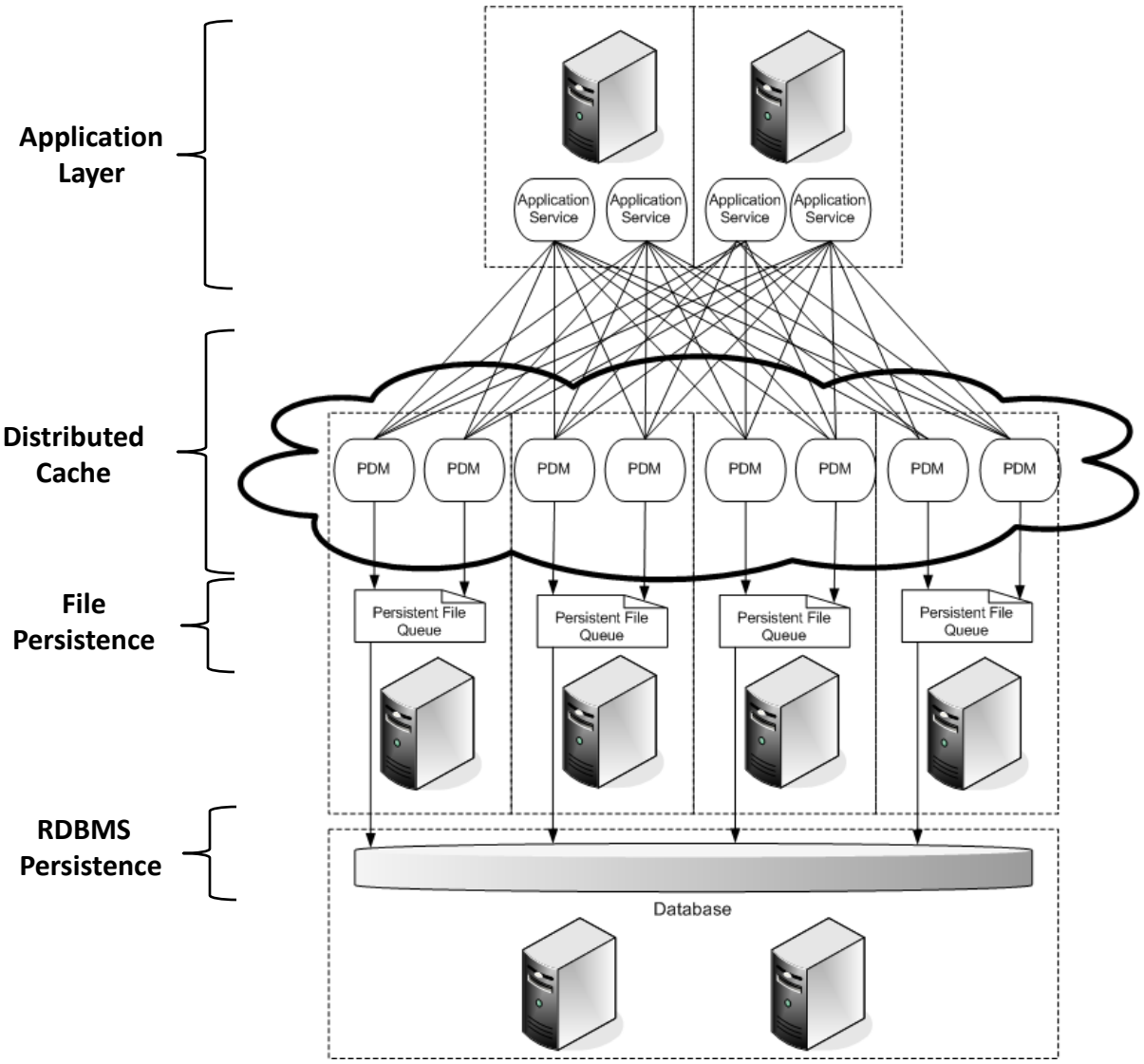
Multiple instances across regions and business lines.

Hydra Architecture
Physical Architecture

Overview
- Core Java software stack.
- Distributed in-memory cache.
- Hierarchical object / event driven data model.
- Real-time publish / subscribe IPC via RMI.
- Asynchronous RDBMS persistence layer.

Scalability Dimensions
- Instances
- Hosts
- Processes
- Threads

Concurrency Patterns
- Multi-threaded infrastructure processes with load balanced resource assignment.
- Notification & subscription queuing.
- Hashing to multiple application service instances.
- Thread pooling into thread safe application services instances.



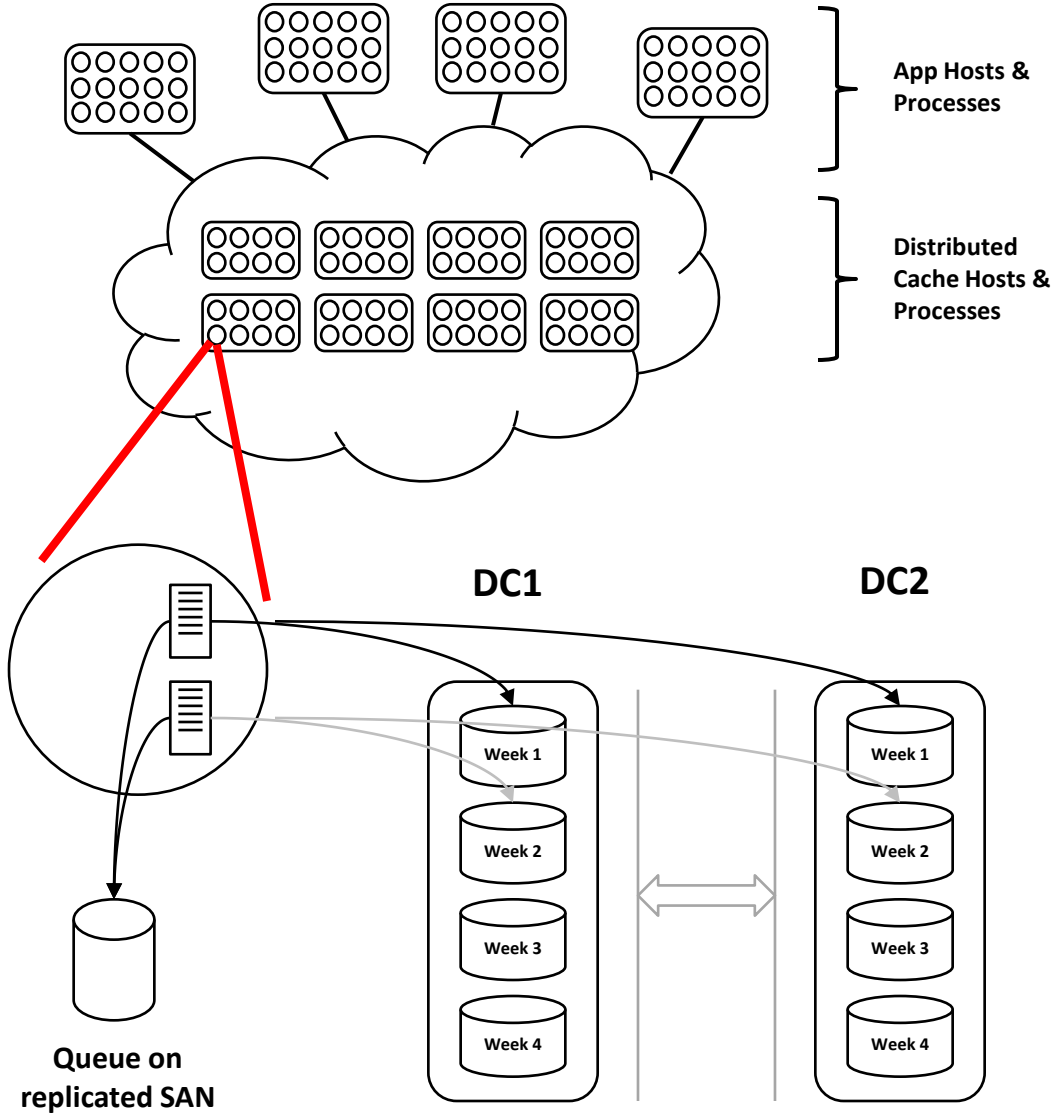
Hydra Architecture
Persistence Architecture

Resilience

Scalability

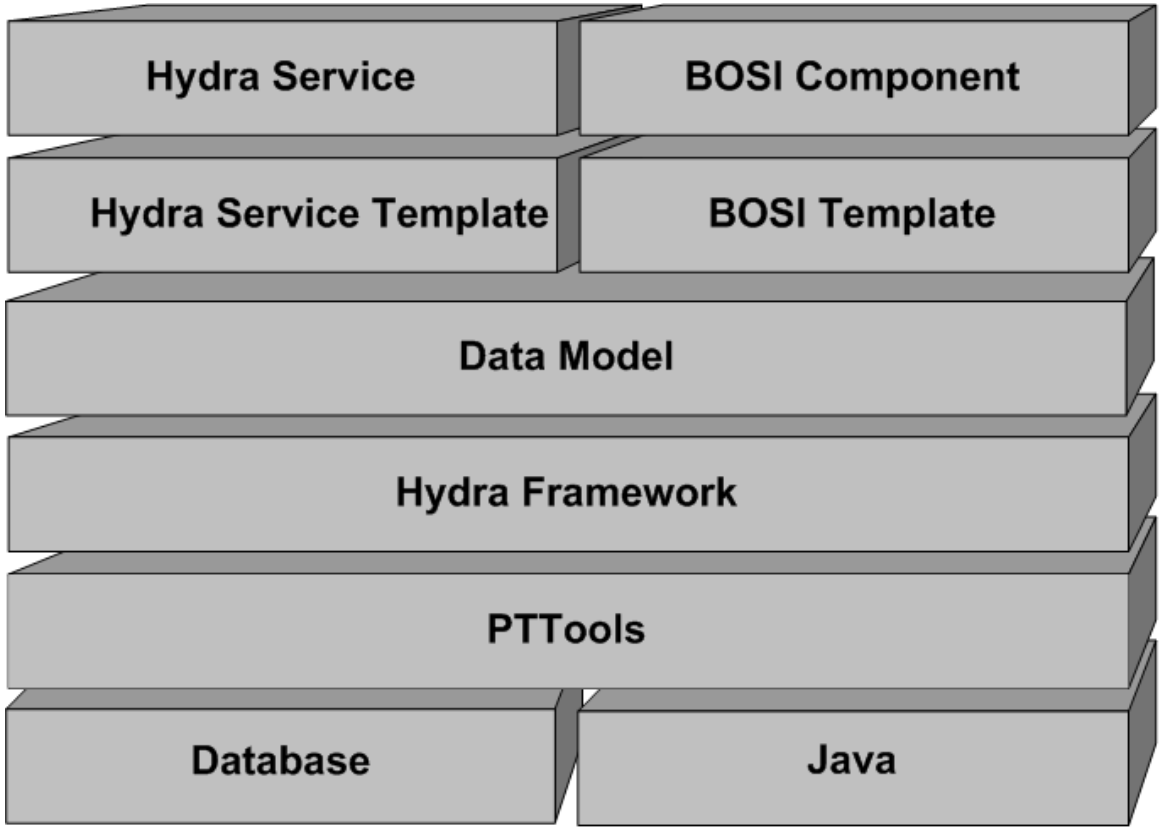
Performance

- Database writes decoupled from application.
- Queue files on replicated SAN.
- Queue writes batched and pre-zipped.
- Queue files have space pre-allocated.
- DB writes pre-compiled and batched.
- DB resilient live-live pairs.
- DB on cheaper and faster non-replicated SAN disk.
- DB reads auto fail-over and fail-back.
- DB striped in date ranges.
- DB purge eliminated.
- Queue tail pointer held in the database.
- Queue replays automagically on recovery.



Hydra Architecture
The Software Stack

All built using just Core Java ...



Hydra Applications Services
- Application specific configuration
- Application business logic

Hydra Service Template
- Standard application structure based on reader, processor & writer abstractions
- Configuration driven subscription implementation
- Common IO interfaces across HSF & PTTools
- Configurable implementations of standard connection types

Hydra Service Framework
- API component factory
- Subscription, query & commit APIs
- Thread pooling, notification queuing & stale data management support

PTTools Application Framework
- Application initialization
- Configuration property trees
- Logging
- Monitoring & metrics
- Rich connection, queue & database management

Core Java Software Stack
RDBMS Persistence Layer

Wrap Up

Hydra, Java & The Future

Platform is 10 years old and likely to evolve and remain active for a further 10+ years.

Maintaining Java as a suitable “Big App” platform & Hydra’s ability to meet the challenge of the next 10 years ...

The Java Language

- Programmer productivity / expressiveness
- Lambda’s, improved collection APIs, DSL support, concurrency models (STM, Actors)

Upgrade Cycles

- JVM stability through Java / OS revisions
- Ease of upgrade / backward compatible API evolutions
- Third-party compatibility

Running the Plant

- Building, bundling & deploying
- Configuration management
- Monitoring & Diagnostics

Architecture

- Service Oriented Architecture
- Dynamic compute, execution fabrics, storage as a service and the cloud

Performance & Capacity

- JVM performance
- Garbage Collection

Appendices

Hydra Architecture
Processing Architecture

Key Characteristics

- **Model –** Object graph of (semi) immutable parent / child related transactions & events.
- **Commit** – Write a transaction object or event. Generates notifications on interested subscriptions.
- **Subscribe** – Registers interests via predicates.
- **Predicates** – Java implemented encoding of filter / SARGs that can traverse the object graph.
- **Notify** – Generates a notification on a subscription (with optional transformer applied).
- **Transformer** – Java implemented return data transformer that can traverse object graph.
- **Query** – Static, point-in-time read from the cache and / or database (with optional predicates / transformers).