Get Off the Bus Gus, 50 Ways to Leave Your Mainframe

Rohit Kelapure
Pivotal Advisory Solutions Architect
Summary

• Before & After

• Four Strategies

• Mainframe Integration Patterns

• z/OS Connect 2.0

• Replacing Batch Processing

• Offloading Data
Modernizing Off The Mainframe

- Lift-and-Shift
- Greenfield Approach
- Incremental Replacement
- Application Tier Replacement
Strangler Pattern
Mainframe Integration Patterns

- **Service Oriented**: Loose coupling using network-based protocols such as SOAP/WSDL
- **Web Access**: Screen-scraping of 3270 applications
- **Messaging**: Using technologies such as IBM MQ
- **Tight Integration**: All components located on mainframe, and linkages are tightly coupled

**API Economy**: Integration based on a set of well-understood and easily-accessible APIs, increasingly based on REST/JSON patterns

*This is where z/OS Connect fits into the picture*
IBM's Answer
IBM z/OS Connect Enterprise Edition
API requests - mapping and transformation

GET
POST
PUT
DELETE

https://<host>:<port>/path?parameters +

Secure the request
Receive the request
Map the request
Audit and logging
Transform the request
Backend System

IBM z/OS Connect 2.0
Eclipse Tooling

- IBM CICS Explorer V5.3
- IBM IMS Explorer for Development V3.2
- IBM Explorer for z/OS Aqua V3.0

Eclipse project view, which is familiar to developers who have used Eclipse-tooling for other development projects

Assign API function based on HTTP verb

API projects can be exported and imported for portability between developers

Access query parameters from the URI

Provide data mapping definitions to the service

API definitions are created through the tool, which is consistent across backend systems (CICS, IMS, etc.)
SOA & Legacy Integration
Saving Money

• Rationalize Redundant Technologies

• Leverage latest features from zOS Transaction Managers

• Compilers Matter, Version Of Java matters

• Transaction Server Optimization

• Externalize Business Rules

• Offload processing to zIIP & zAAP Processors
Batch Modernization

- Batch programs concurrently executed with online transactional programs
- Ability to freely intermix technologies to enhance batch programs
- Reduction or Elimination of needless data movement
- Technical Solutions - I/O Acceleration Techniques …
- Don’t use batch for app integration
Spring Batch on PCF

- JSR 352 Specification on top of Spring Batch Implementation
- Auto scale slave nodes in the cloud based on the amount of data that the batch job processes
- SCDF PartitionHandlers execute slave partitions as Tasks on PCF
- v3 CAPI that exposes Diego Task API is experimental
- `cf cli Plugin` for v3 apps lacks ability to delete apps bound to services
- v3 apps do not show up in the AppsManager UI
Apache Geode - Offloading Mainframe data

- **OQL and Indexes**: Object Query Language allows distributed query execution on hot and cold data, with SQL-like capabilities, including joins. Multiple kinds of indexes can be defined and consistently maintained across the cluster.
- **Events**: Clients can be notified about server-side data events, and servers can react synchronously or asynchronously with guaranteed delivery of ordered events.
- **Clustering**: Highly scalable, robust advanced clustering technology with failure detection, dynamic scaling, and network-partition detection algorithms.
- **Multi-Cluster**: Geode clusters can be replicated over WAN in various topologies: active-active, active-passive, ring, hub-spoke, star, etc.
- **Continuous Query**: Clients can stay up to date by registering OQL queries with the Geode servers, making event-driven applications possible.
- **Clients**: Clients are available for Java. (C++, C# .NET and Node.js coming soon.) A REST API is available for all other languages.
- **Adapters**: Geode can be used as a drop-in replacement for Redis and memcached, allowing users of these caches to use Geode's server-side features like multi-cluster replication.
More Apace Geode

Replication and Partitioning
Data can easily be partitioned (sharded) or replicated between nodes allowing performance to scale as needed. Durability is ensured through redundant in-memory copies and disk-based persistence.

Persistence
Super fast write-ahead-logging (WAL) persistence with a shared-nothing architecture that is optimized for fast parallel recovery of nodes or an entire cluster.

Performance
Linear-scaling low latency for transactions, reads, writes and query processing of indexed or unindexed data.

In-Memory Storage
Blazing fast in-memory storage optimized for large heaps, with the option of using off-heap storage, compression and features such as disk-overflow, eviction and expiration of data.

Functions
Distributed location-aware user functions can be deployed and executed by the same nodes storing relevant sharded data for fast parallel processing. Failed operations can be retried on replicant nodes.

Transactions
ACID distributed transactions support efficient and safe coordinated operations on colocated data. Transactions can be initiated or suspended by either a client or a server.
Spring Cloud Data Flow

Geode
Stream/Task Spring Boot Apps
Offloading Mainframe Data into Hadoop

Data Architecture – Data Life Cycle
Thank You
Rohit Kelapure

rkelapure@pivotal.io
https://twitter.com/rkela
https://www.linkedin.com/in/rohitkelapure
http://cloud.rohitkelapure.com/