In-Memory Computing SUMMIT EUROPE 2018

DESIGNING MULTI-CLUSTER APPLICATIONS

LUCAS BEELER

GRIDGAIN SYSTEMS, INC.

WHO AM I?

- Professional Services Architect at GridGain Systems
- A post-sales role—I help customers design and build GridGain-based applications
- Worked at GigaSpaces in the same role before joining GridGain
- This presentation is the result of hard-won experience designing and building systems for my largest clients
- The approach I'm going to present may seem antithetical to received wisdom about "correct" or "best practices" IMDG application design

WHAT IS THIS TALK ABOUT?

- Building multi-cluster applications
  - Single applications whose functionality is implemented over multiple IMDG clusters
- A **cluster** is a distributed in-memory data grid that supports collocated processing
- An **application** is a deployable set of code artifacts that deliver a coherent, unified collection of business functions; when these artifacts are executed, they run over a cluster
- We usually think of deploying a code artifact—like a JAR—to a cluster, but here we're talking about deploying multiple artifacts to multiple clusters
  - We do this for separation of concerns and orthogonal evolution and scalability

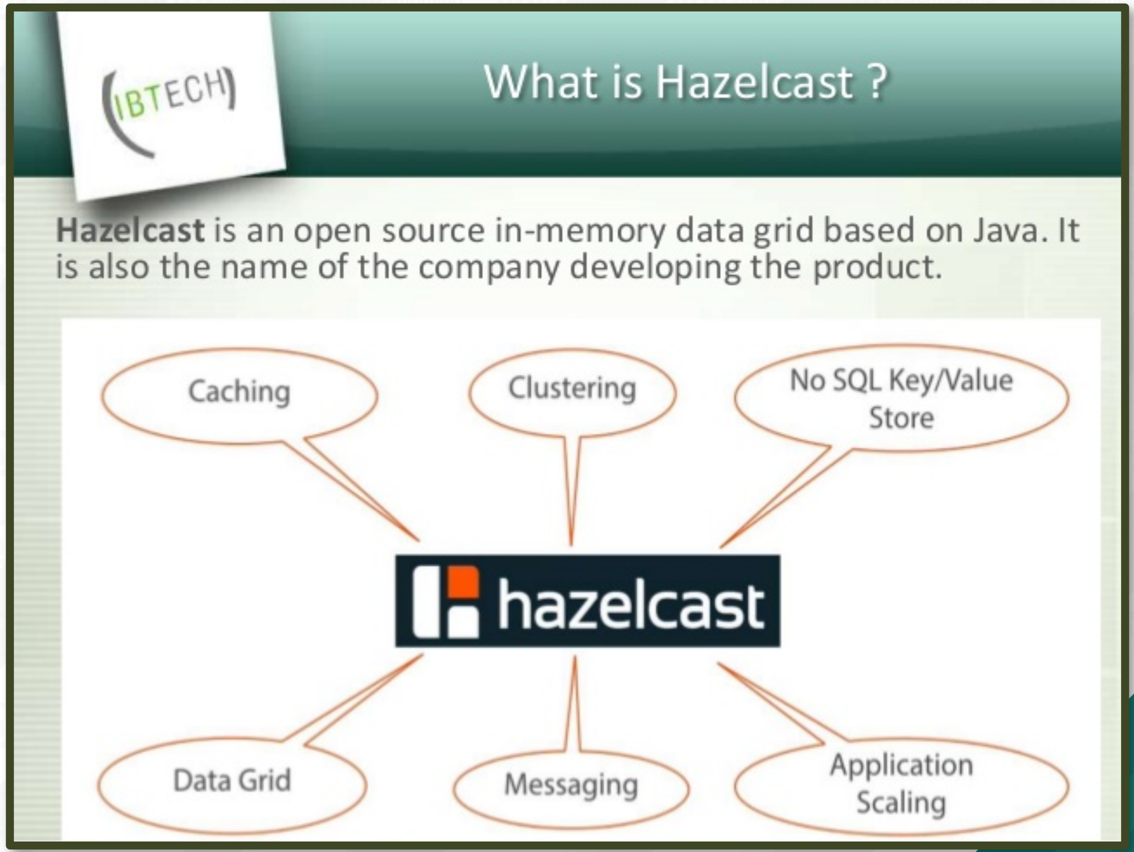
AGENDA

- Conceptual overview of modern IMDG functionalities
- An example single-cluster application and its limitations
- The advantages of transitioning our application to a multi-cluster design
- Case study implementations
- Question & answer

FUNCTIONS AN IMDG APPLICATION MIGHT PERFORM…

- In-memory caching of data in its native object form (obviously!)
  - No ORM
- Analytics
  - Type classification (what kind of a thing is this?)
  - Domain-specific performance analysis (how well did that work?)
  - Statistical learning
- Stream data ingestion and ETL operations
- Event processing ("reactive programming", "triggers")
- Provide a local mirror of SaaS data—possibly in a different form—or push data to a SaaS service
- Integrate data many sources in a unified queryable store

HOW DO IMDGs ACCOMPLISH SUCH A RANGE OF TASKS?



What is Hazelcast ?

Hazelcast is an open source in-memory data grid based on Java. It is also the name of the company developing the product.

Image from and may be copyrighted by IBTech QNB A.Ş and Hazelcast, Inc.

HOW DO IMDGs ACCOMPLISH SUCH A RANGE OF TASKS?

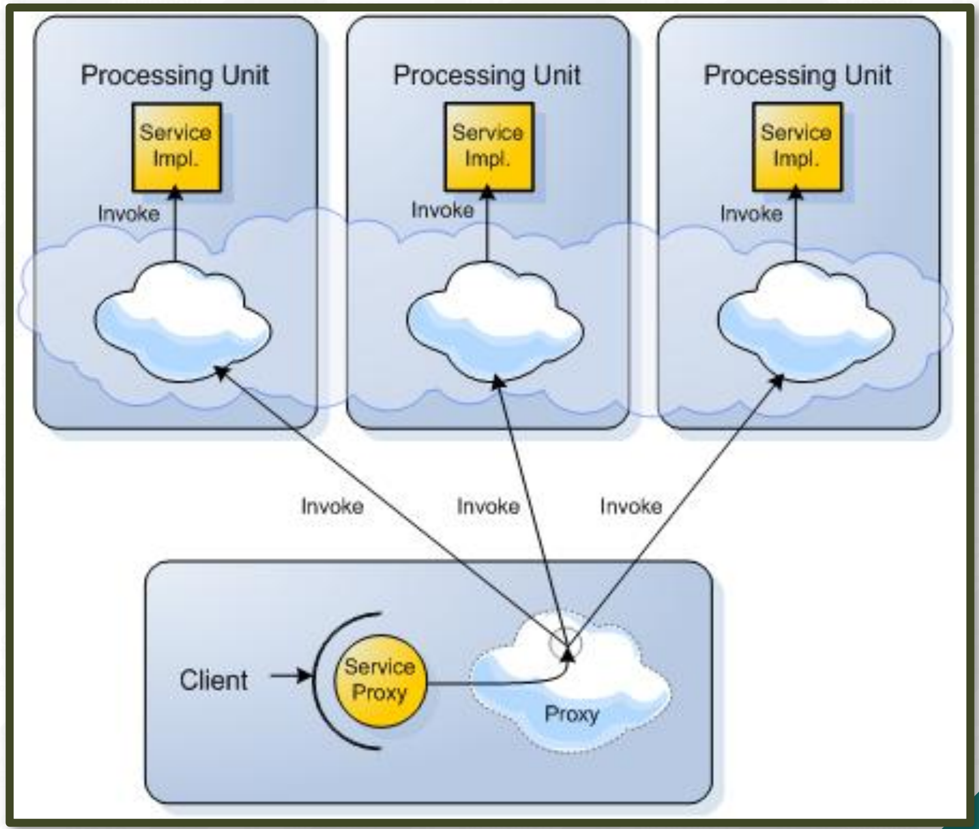
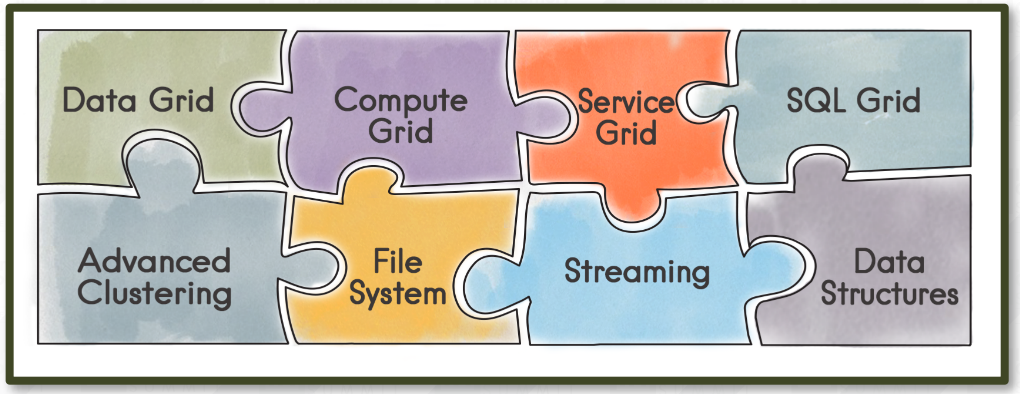
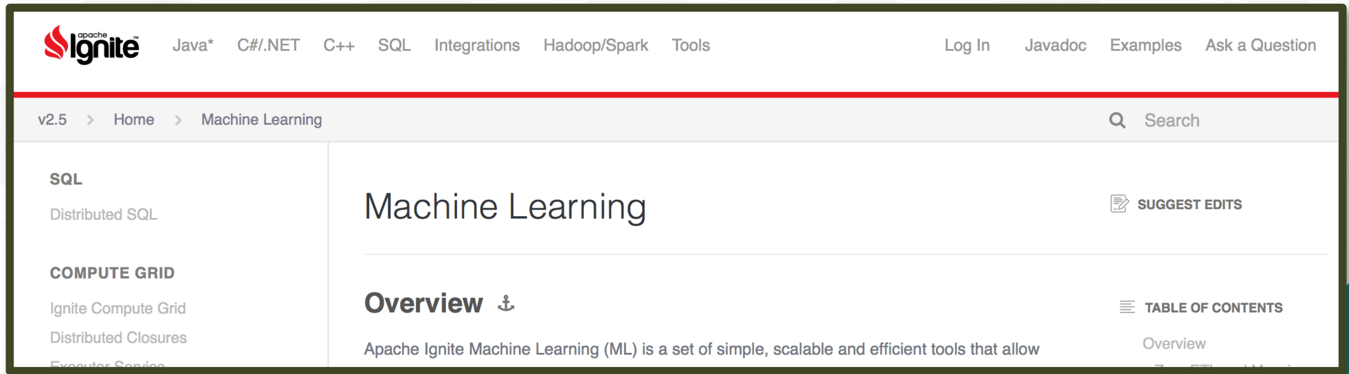


Image from and may be copyrighted by GigaSpaces, Inc.

HOW DO IMDGs ACCOMPLISH SUCH A RANGE OF TASKS?





Machine Learning

Overview

Apache Ignite Machine Learning (ML) is a set of simple, scalable and efficient tools that allow

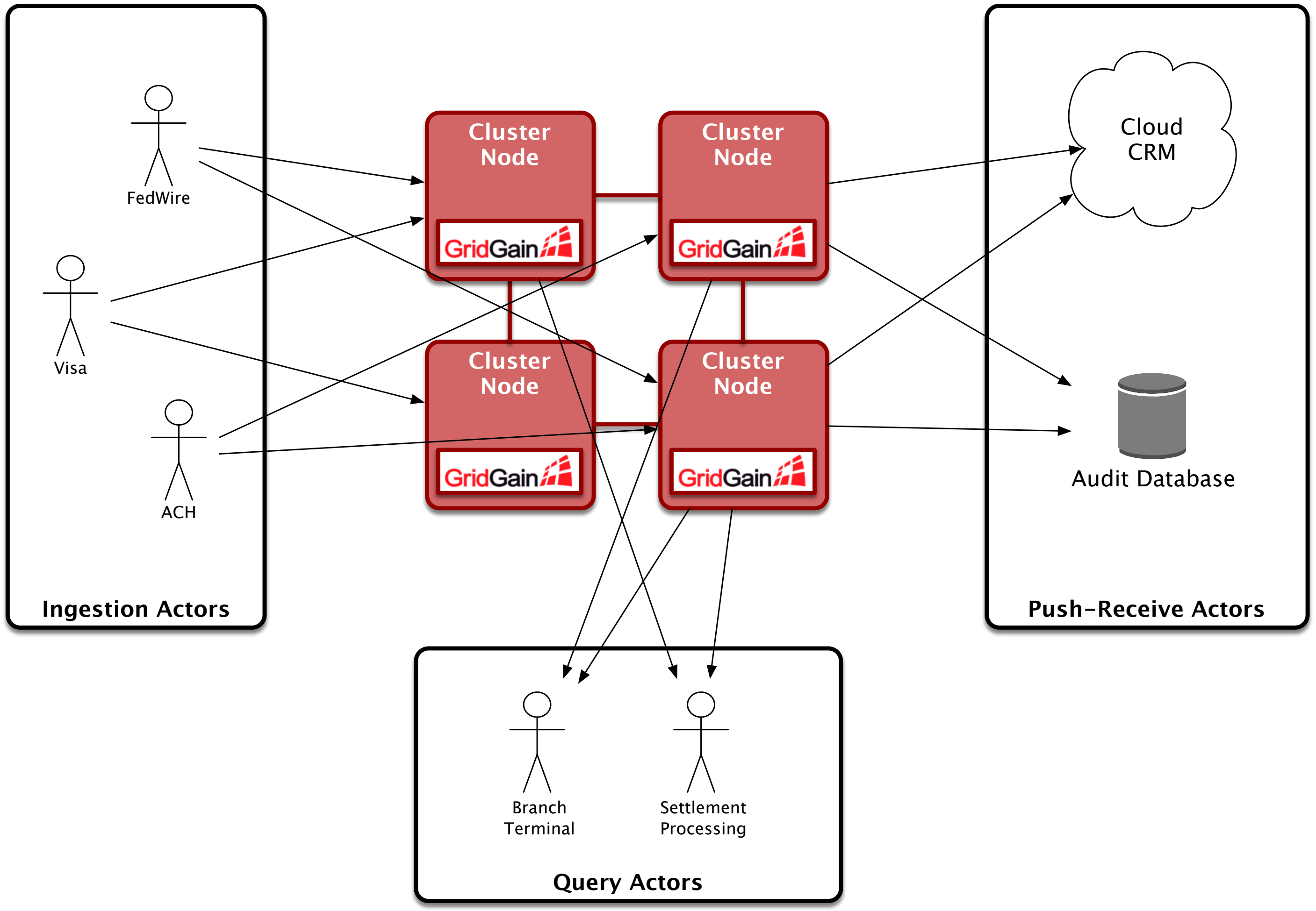
FORRESTER®

"IMDGs, in general, suffer from a caching stereotype. **GridGain smashes that stereotype** by offering a general purpose compute grid. **If you need an IMDG and a super-computer all in one then take a look at GridGain Systems**." (Mike Gualtieri, Forrester)

In-Memory Computing SUMMIT EUROPE 2018

AN APPLICATION: LET'S IMAGINE…

- We're a regional bank
  - We have 12 branch offices and 104,000 customers
- We do all the things that commercial banks do:
  - We take deposits: cheques, cash, electronic direct deposit, etc.
  - We allow withdrawals: cash, cheques, debit card, etc.
  - We issue credit cards associated with a processing network (e.g., Visa or MasterCard)
  - We make loans
  - We manage relationships with our customers and would like them to engage with ***more*** of our products
  - Assume we're US-based: we comply with both state and federal banking regulations


FedWire
Visa
ACH
Ingestion Actors
Cluster Node
GridGain
Cluster Node
GridGain
Cluster Node
GridGain
Cluster Node
GridGain
Cloud CRM
Audit Database
Push-Receive Actors
Branch Terminal
Settlement Processing
Query Actors

CAN WE TALK ABOUT HOW AWESOME THIS IS?

- Yeah, we process transactions.
- But also…
  - When a customer uses his or her ATM/debit card to get cash at a foreign cash point, we push a notification to our cloud CRM system
  - If that customer doesn't already have our SkyPoints™ Airline Rewards MasterCard (issued in conjunction with Alaska Airlines and Emirates), we send him or her a spiffy marketing email
  - The system provides us with rich primitives for building functionality—amazing!
    - Class `StreamReceiver` and `StreamTransformer`
    - Compute, service, and event processing APIs make it easy to implement a rules engine

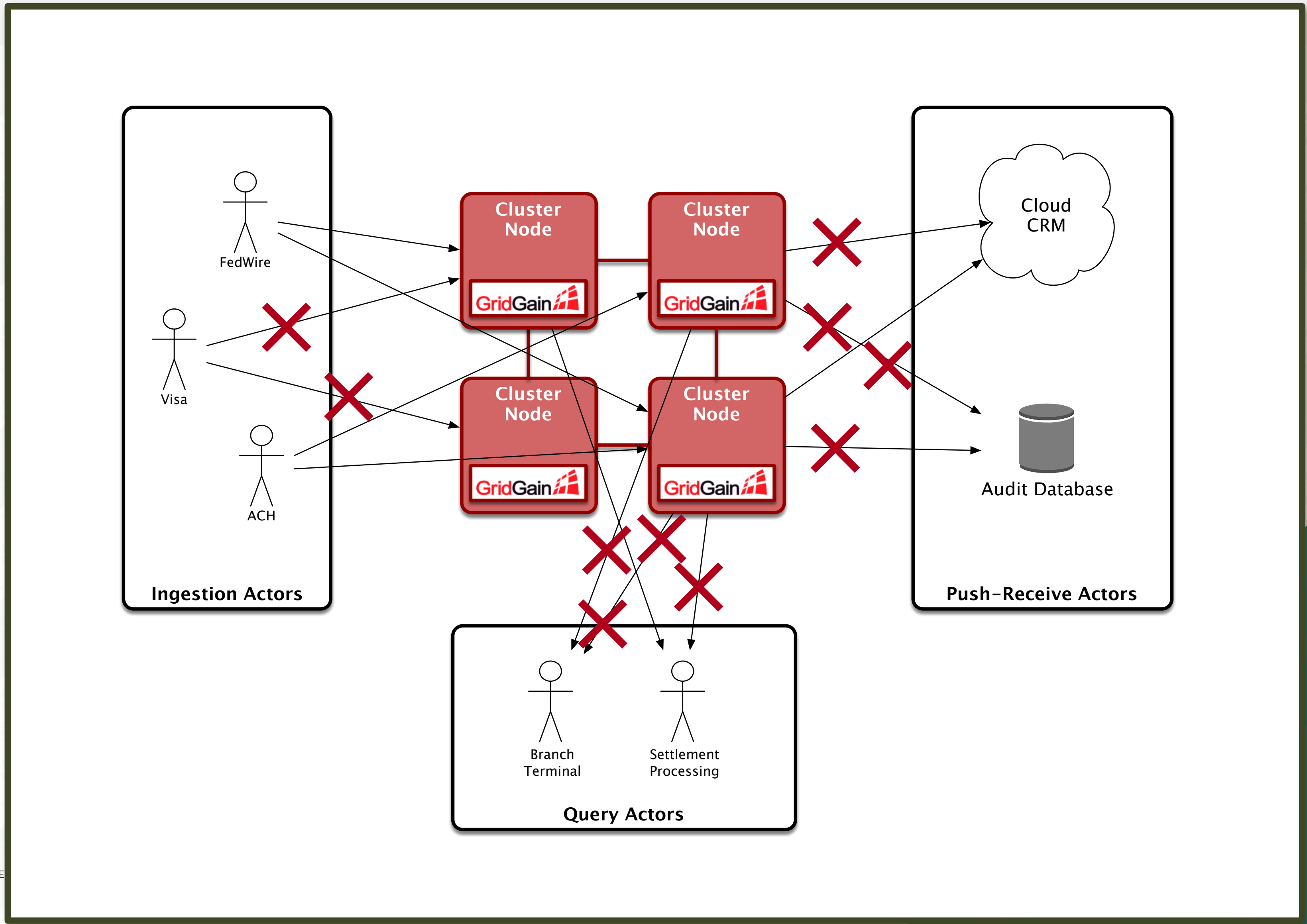
HOW DID WE IMPLEMENT THIS?

- We followed "best-practices" design protocols for clustered IMDG applications
  - Assume that inside the grid, one important datatype is a `CustomerTransaction` POJO
  - **Shared-nothing for maximum collocation**
    - All cluster nodes are identical in configuration and can **complete our entire business workflow within a single memory image**
      - There is no shared state between nodes, so network movement between nodes is necessary
    - We can stream FedWire transaction receipts to the same node where the final `CustomerTransaction` POJO that corresponds to the receipt will be stored and ETL them "in-place" from FedWire XML to `CustomerTransaction` POJOs
    - Nodes holding `CustomerTransasction` instances of importance for marketing purposes can transform them into a REST JSON message and push that message out directly, from their local memory images, to our cloud-hosted, SaaS CRM service
    - If we want to increase total system capacity, all we have to do is **add another node**

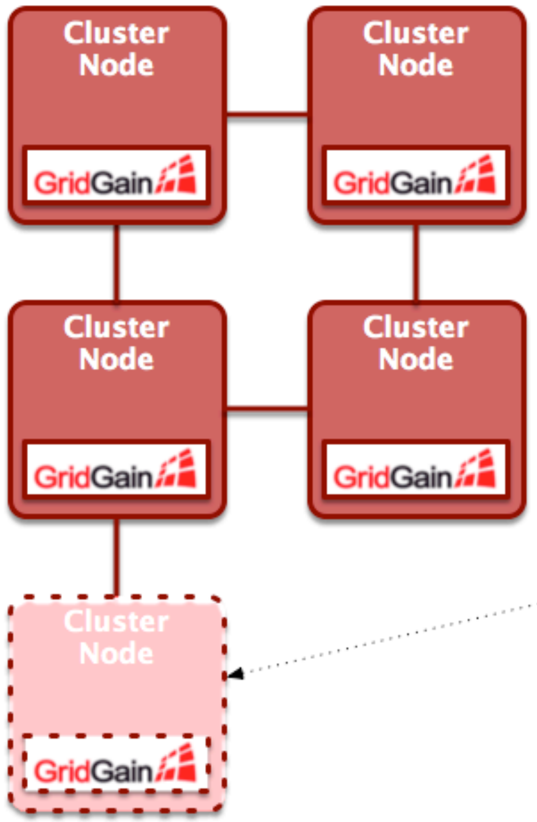
SO WHAT’S ON EVERY NODE?

- We followed “best-practices” design protocols for clustered IMDG applications—“shared nothing”
  - `CustomerTransaction` POJO instances
  - FedWire `StreamReceiver`
  - Visa Processing Network `StreamReceiver`
  - ACH `StreamReceiver`
  - Audit Database `CacheStoreAdapter`
  - Cloud CRM Pusher Service
  - Branch Terminal Query Service
  - Settlement Processing Query Service

COOL. BUT WHAT HAPPENS AT MIDNIGHT EASTERN TIME?



IT'S MIDNIGHT. LET'S ADD EXTRA CAPACITY…



- Holds CustomerTransaction POJOs
- Has a Cloud CRM Pusher Service
- Has a Branch Terminal Query Service

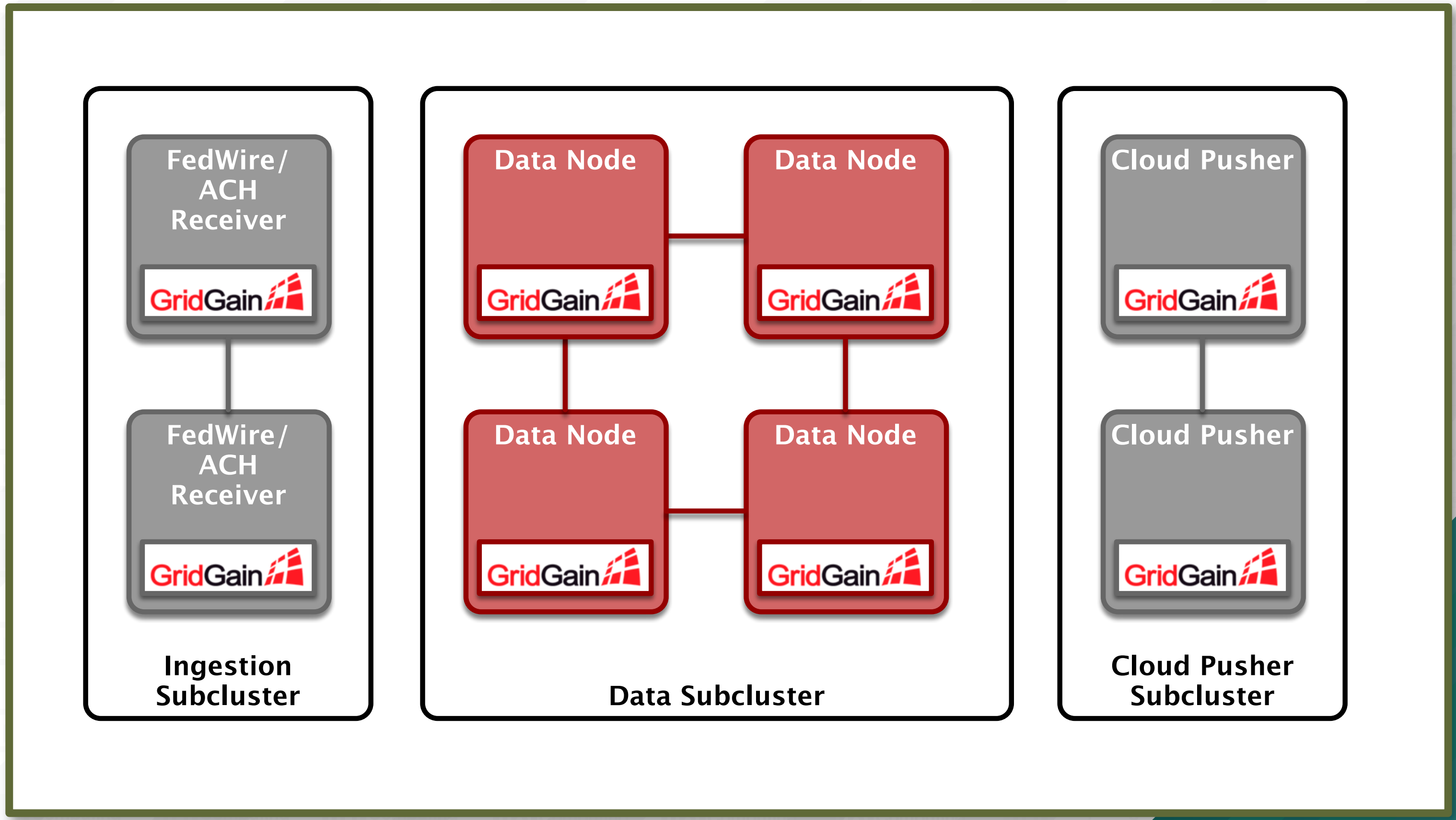
…

- Has a FedWire StreamReceiver
- Has an ACH StreamReceiver

SO LET’S LEAVE “SHARED NOTHING” BEHIND…

- We will have multiple clusters (physical or logical) made up of different kinds of nodes
  - Often called “subclusters”
    - “Shared nothing,” uniform provisioning and horizontal scaling (add another node) will still apply ***within*** a subcluster. It’s just that now we have multiple subclusters.
- What we give up
  - DevOps isn’t simple anymore—we no longer have just one cluster node image; we have several
  - We can no longer scale the entire system by adding a node, instead we scale along different ***scaling axes***
  - Each subcluster represents a different scaling axis
  - We can’t complete our entire business process without sharing data between nodes and performing network operations—we’ve broken shared nothing across the entire cluster.
- We’ll talk about what we **gain** over the next few slides

WHAT WILL WE DO?



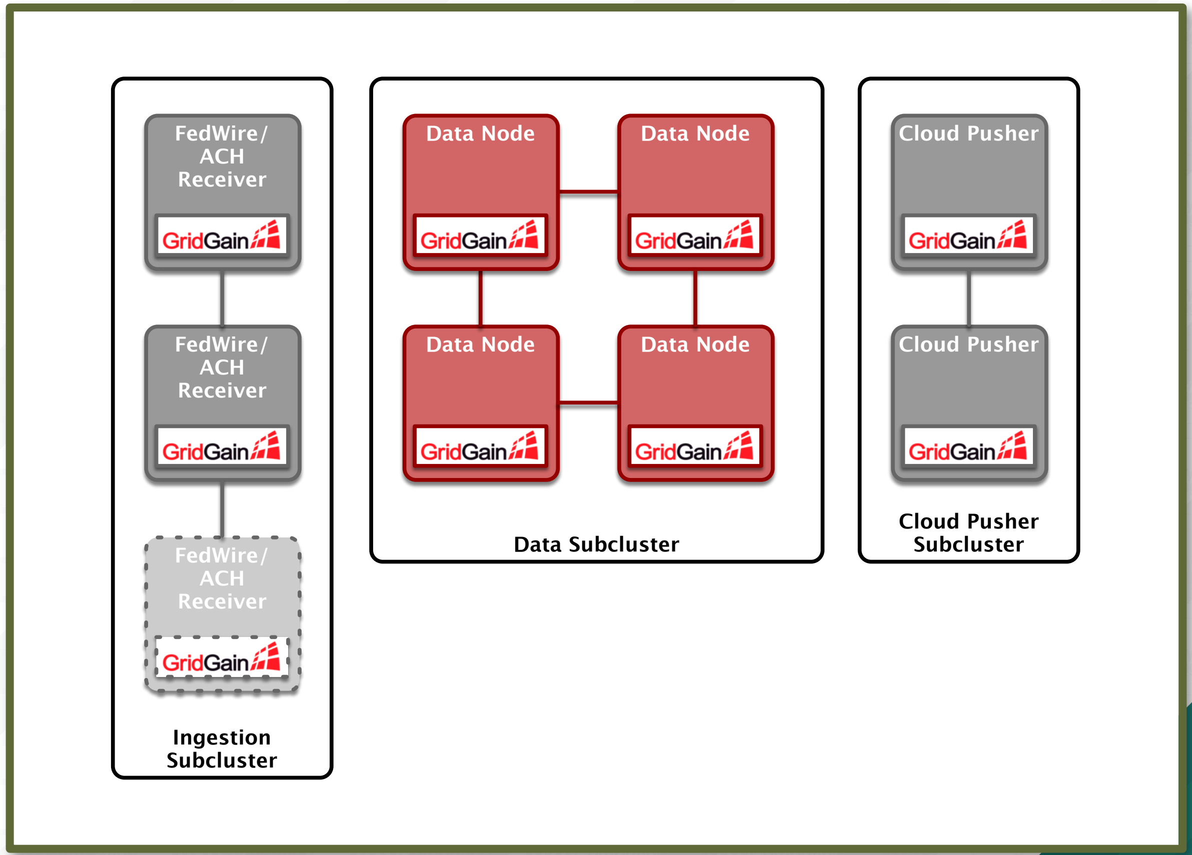
HOW WILL WE DO IT?

```
<bean class="org.apache.ignite.IgniteConfiguration">
...
    <property name="userAttributes">
        <map>
            <entry key="role" value="fedAchReceiver"/>
        </map>
    </property>
 ...
</bean>
```

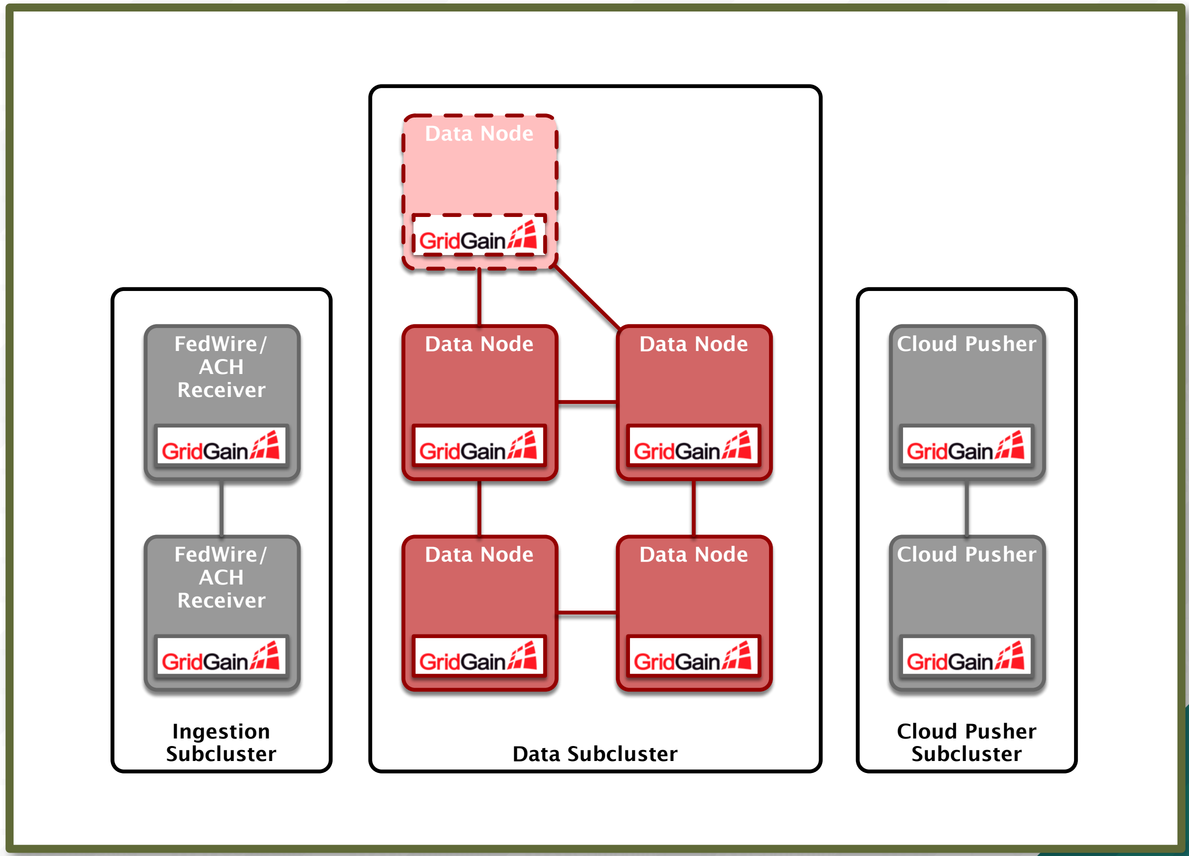
IMMEDIATE ADVANTAGES

- All FedWire and ACH ETL operations are separated from nodes that store data
  - Queries don't slow down when FedWire and ACH message volume is high
- We can bring down our Ingestion Subcluster and upgrade our FedWire and ACH ETL logic without affecting our Data Subcluster at all
  - No data movement, no rebalancing, no network saturation
  - Data cluster never enters an intermediate topology state
  - This is "orthogonal evolution"

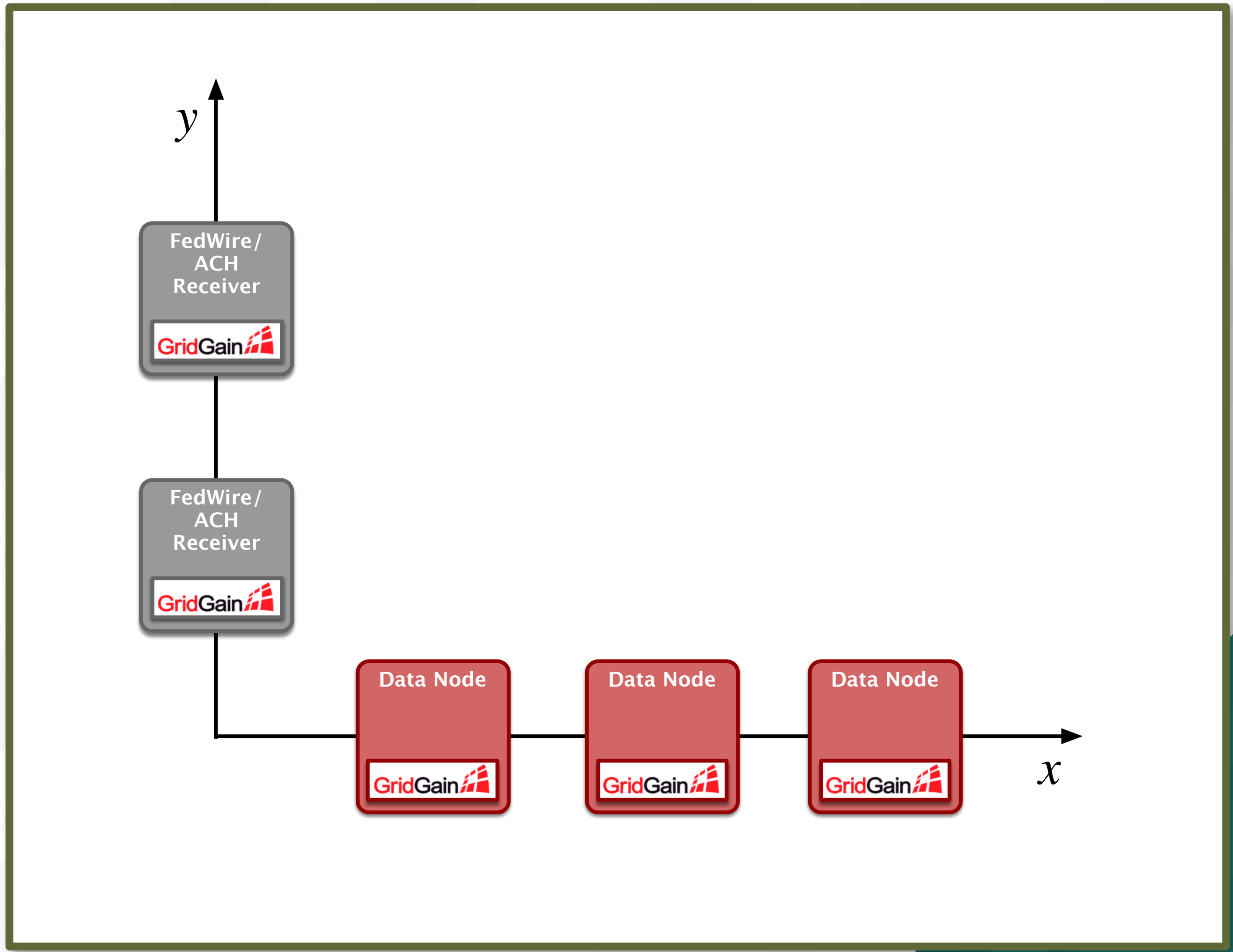
ORTHOGONAL SCALABILITY



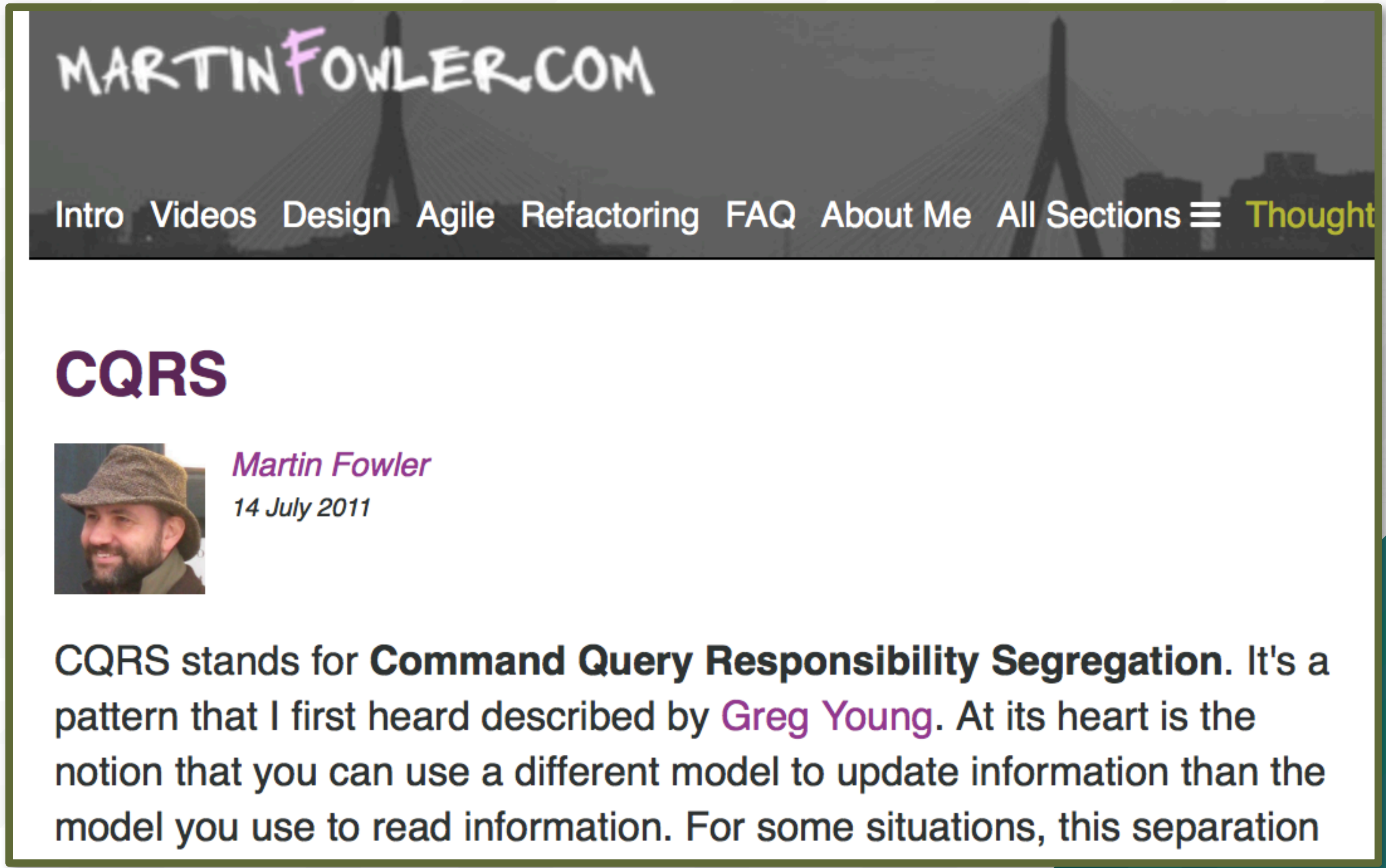
ORTHOGONAL SCALABILITY



ORTHOGONAL SCALABILITY



CASE STUDIES: DESIGN THEMES


MARTINFOWLER.COM
Intro Videos Design Agile Refactoring FAQ About Me All Sections ≡ Thought

CQRS

Martin Fowler
14 July 2011

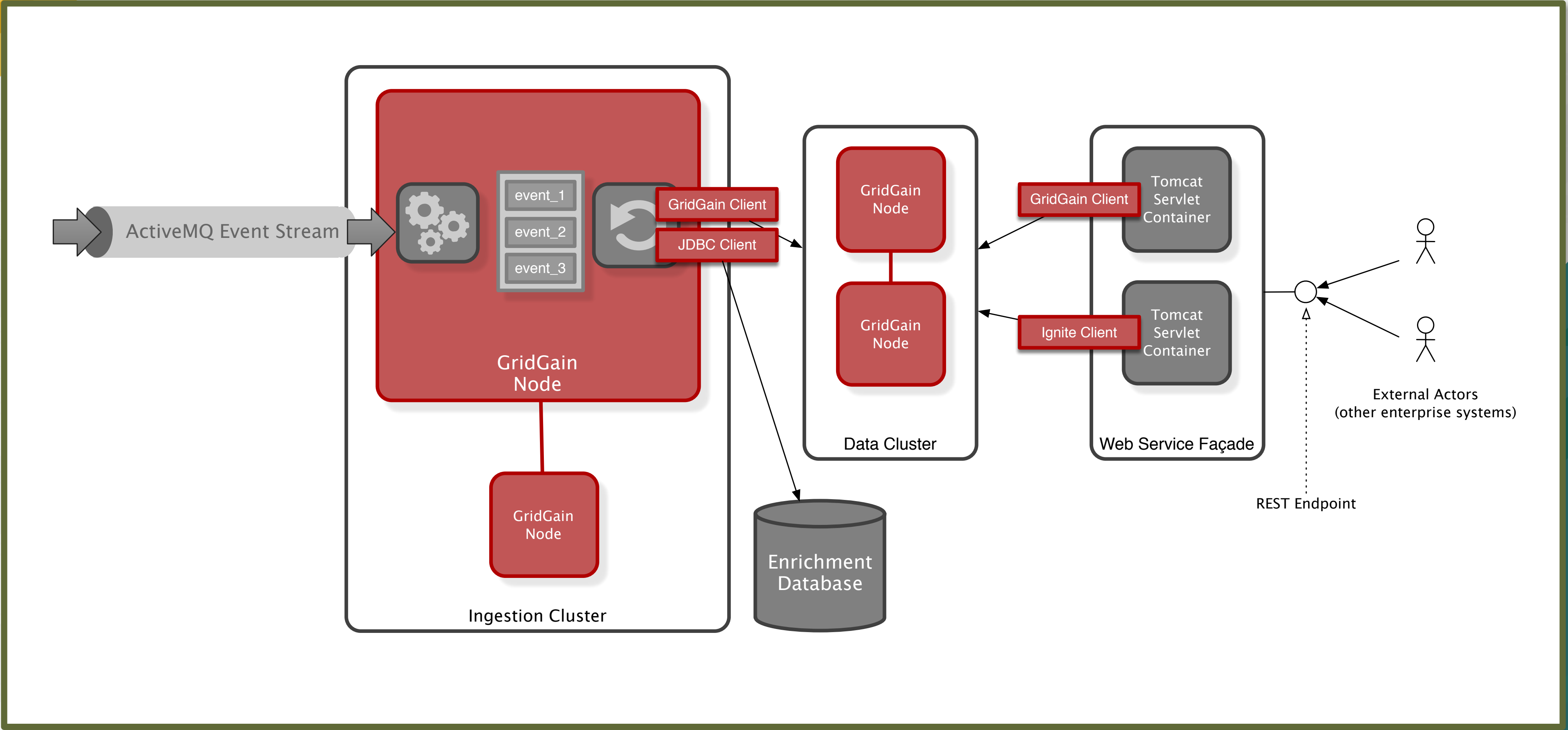
CQRS stands for **Command Query Responsibility Segregation**. It's a pattern that I first heard described by Greg Young. At its heart is the notion that you can use a different model to update information than the model you use to read information. For some situations, this separation

Image from and may be copyrighted by Martin Fowler or ThoughtWorks, Inc.

CASE STUDY 1

- Customer is a large wealth management firm:
  - About $1 trillion in assets under management
  - Actively trades in financial markets to help institutional investors like pension funds grow wealth over time
  - Had a requirement that queries had to be very fast (on the order of microseconds) to feed other enterprise systems
  - However, trade tickets arrive in the cluster via a queue in a primitive format that is useless for data consumers without enrichment

CASE STUDY 1



ActiveMQ Event Stream
event_1
event_2
event_3
GridGain Client
JDBC Client
GridGain Node
GridGain Node
Ingestion Cluster
Enrichment Database
GridGain Node
GridGain Node
Data Cluster
GridGain Client
Ignite Client
Tomcat Servlet Container
Tomcat Servlet Container
Web Service Façade
REST Endpoint
External Actors (other enterprise systems)

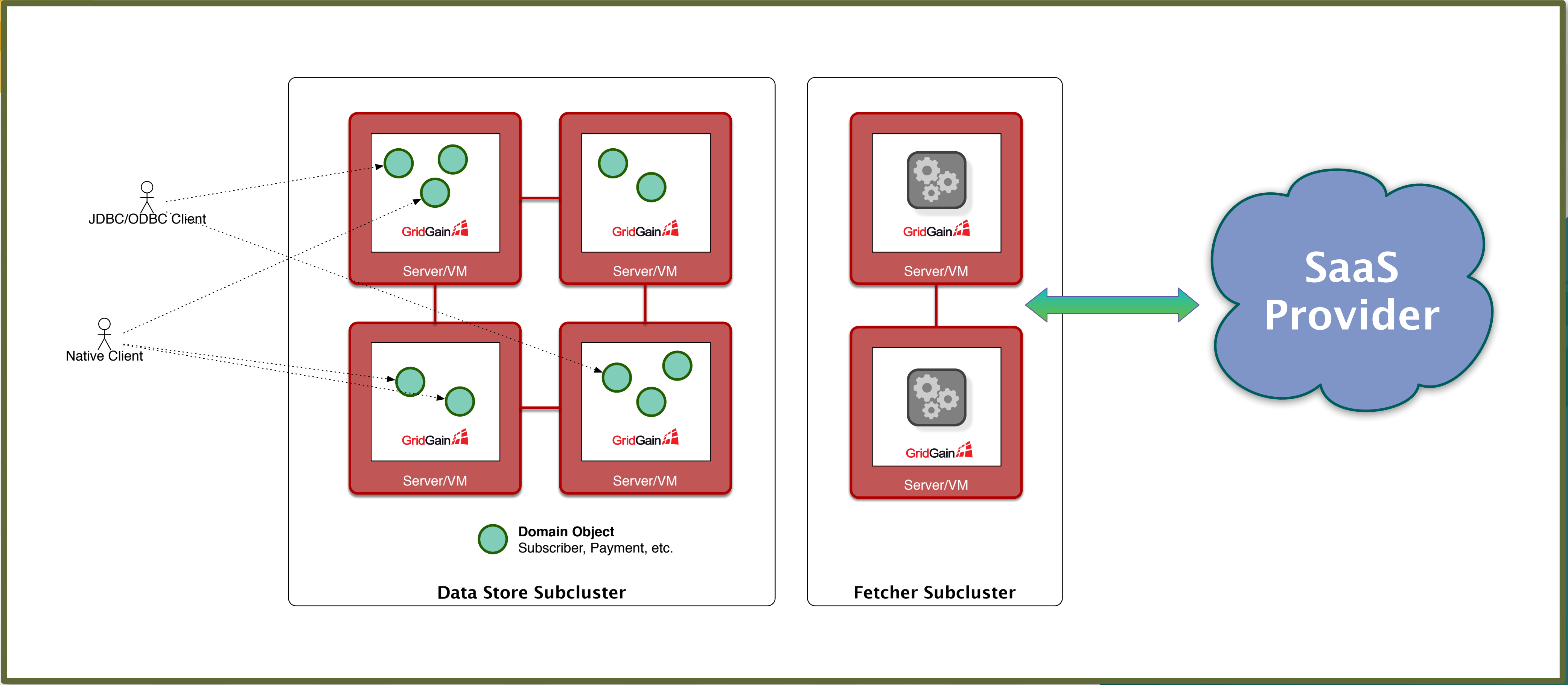
CASE STUDY 2

- Customer is one of the largest fitness center operators in the US
  - About 300 gym clubs and 4 million members
  - A decision was taken to move customer engagement management (marketing campaigns, etc.) and subscription management (billing, etc.) from internal, on-premise systems into a cloud-based SaaS CEM/subscription management provider
    - Lowers costs: software engineers are expensive, as is hardware and the people who maintain it. It was cheaper to pay a fee to the SaaS provider than it was to maintain the staff required to develop, operate, patch, and maintain internal systems for the same purpose.
    - Subscription management is an ancillary business activity. The company would much rather devote its time to improving quality of its facilities, creating innovative subscription offers, pricing tiers, etc.

CASE STUDY 2

- However…
  - The customer had concerns about 24-7 availability of data stored by the SaaS provider. The SaaS data is crucial to business functions and cannot be unavailable, even in the event of network disruptions, DDoS attacks, outages, etc.
  - Outages do happen, even for very well maintained web providers: both Netflix and Gmail have suffered recent, high-profile outages
  - The SaaS provider has its own data model. This model is not fully compatible with the customer's internal business object model.
  - The SaaS provider only supports storing certain kinds of data that match its data model. If you can't express your data within the constraints of the SaaS provider's model, you're out of luck.
  - Hence, a lot of business data still had to be kept on-premises
  - The customer wanted the ability to join on-premise data with SaaS hosted data by querying a single, internal system

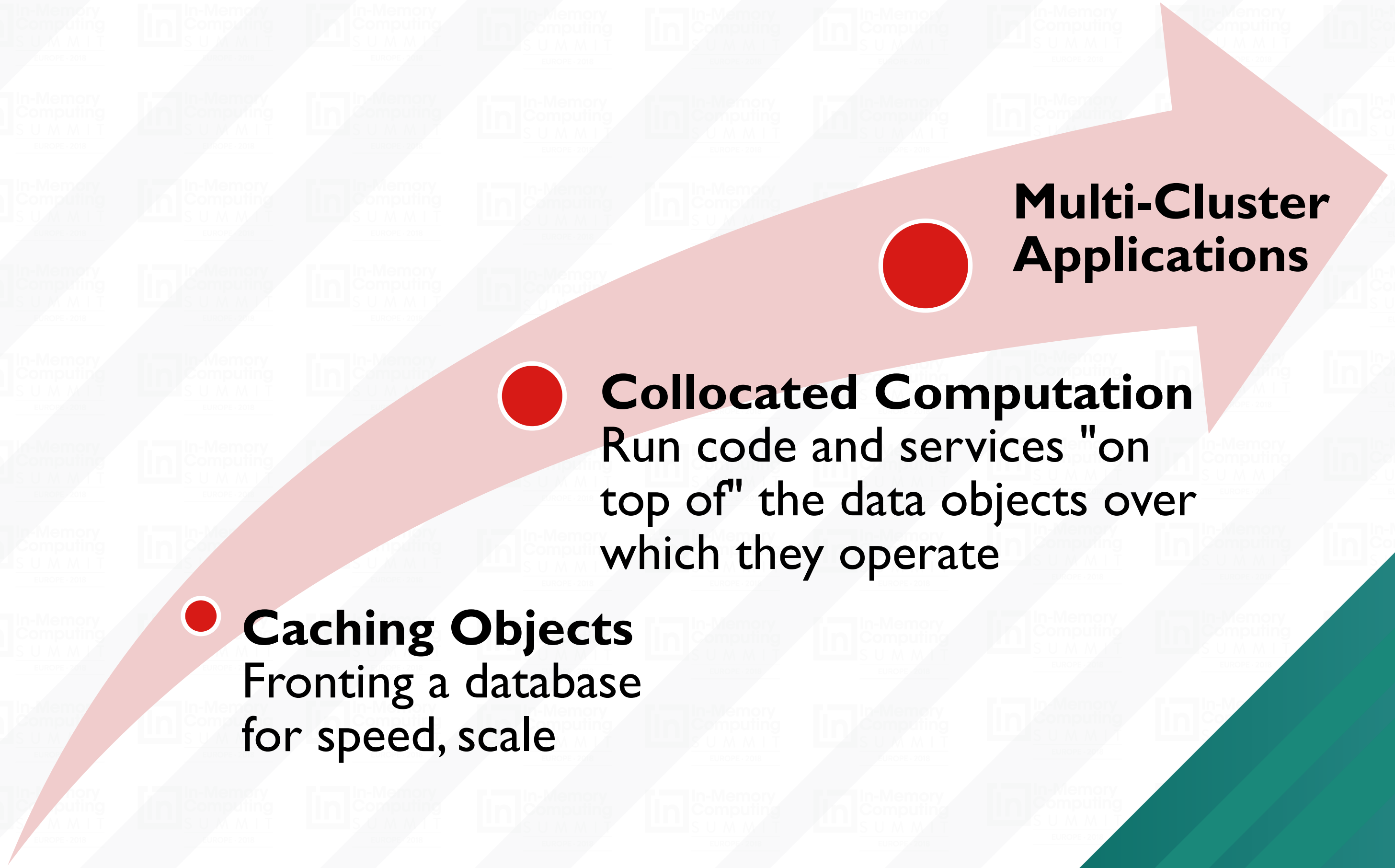
CASE STUDY 2


JDBC/ODBC Client
Native Client
GridGain
Server/VM
Domain Object
Subscriber, Payment, etc.
Data Store Subcluster
Fetcher Subcluster
SaaS Provider

CASE STUDY THEMES

- In both cases, interacting with external systems was critical to the business functions of the system
  - The time required to communicate with external systems (e.g., the WAN link to the cloud SaaS provider) dwarfed any time costs associated with local data movement, so by breaking shared nothing, we weren't going to degrade performance
- In one case, compute-intensive ETL operations were required
  - The time required to complete these computations were several orders of magnitude greater than network movement time
  - The same is true for other compute-intensive tasks like machine learning and statistical classifaction
    - Sometimes, you can't do HTAP in one cluster; you need a multi-cluster design

A CAPABILITY-MATURITY MODEL FOR IMC



Multi-Cluster Applications

Collocated Computation
Run code and services "on top of" the data objects over which they operate

Caching Objects
Fronting a database for speed, scale