# Pivotal

A NEW PLATFORM FOR A NEW ERA

# Evolution of Pivotal Gemfire

## Which way might the "Apache Way" take It?

Roman Shaposhnik
rvs@apache.org
Director of Open Source, Pivotal Inc.
@rhatr

Milind Bhandarkar
milind@ampool.io
CEO, Ampool Inc.
@techmilind

Pivotal™

# Topics

- GemFire history, architecture, and use cases
- Geode as the open source core of GemFire
- Requirements of the modern data infrastructure
- Butterfly architecture
- Geode as an engine for in-memory data exchange layer

It's not the size of

# DATA

it's how you use it!

Pivotal™

# Our GemFire Journey Over The Years

**2004**

- Massive increase in data volumes
- Falling margins per transaction
- Increasing cost of IT maintenance
- Need for elasticity in systems

- Financial Services Providers (every major Wall Street bank)
- Department of Defense

**2008**

- Real Time response needs
- Time to market constraints
- Need for flexible data models across enterprise
- Distributed development
- Persistence + In-memory

- Largest travel Portal
- Airlines
- Trade clearing
- Online gambling

**2014**

- Global data visibility needs
- Fast Ingest needs for data
- Need to allow devices to hook into enterprise data
- Always on

- Largest Telcos
- Large mfrers
- Largest Payroll processor
- Auto insurance giants
- Largest rail systems on earth

Pivotal

# Apps at Scale Have Unique Needs

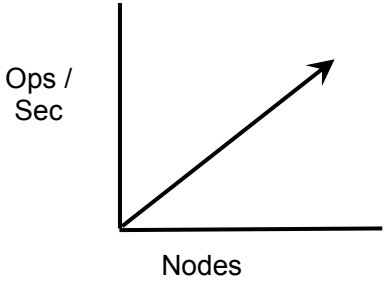
Pivotal GemFire is the distributed, in-memory database for apps that need:

1. Elastic scale-out performance
2. High performance database capabilities in distributed systems
3. Mission critical availability and resiliency
4. Flexibility for developers to create unique applications
5. Easy administration of distributed data grids

Pivotal™

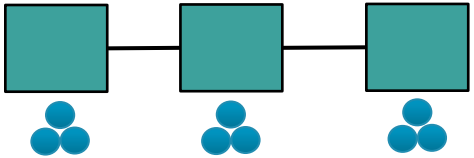
# 1. Elastic Scale-Out Performance

Linear scalability




Elastic capacity


Latency-minimizing data distribution



## China Railway Corporation

*"The system is operating with solid performance and uptime. Now, we have a reliable, economically sound production system that supports record volumes and has room to grow"*

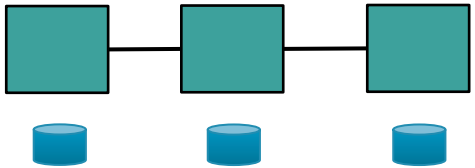
**Dr. Jiansheng Zhu, Vice Director of China Academy of Railway Sciences**

- **4.5 million** ticket purchases & **20 million** users per day.
- Spikes of **15,000** tickets sold per minute, **40,000** visits per second.

Pivotal

# 2. High Performance Database Capabilities

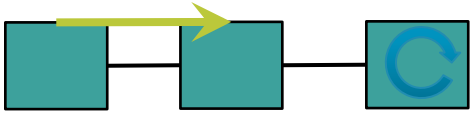
Performance-optimized persistence



Configurable consistency

Partitioned Replicated Disabled

Distributed & continuous queries



Distributed transactions, indexing, archival

## Newedge

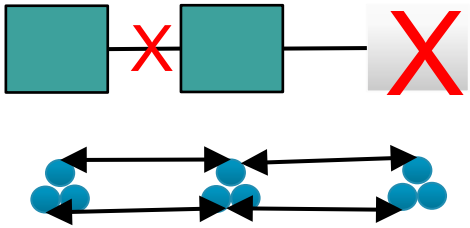
*"Our global deployment of Pivotal GemFire's distributed cache gives me a single version of the trade – resolving hard-to-test-for synchronization issues that exist within any globally distributed business application architecture"*

**Michael Benillouche, Global Head of Data Management**

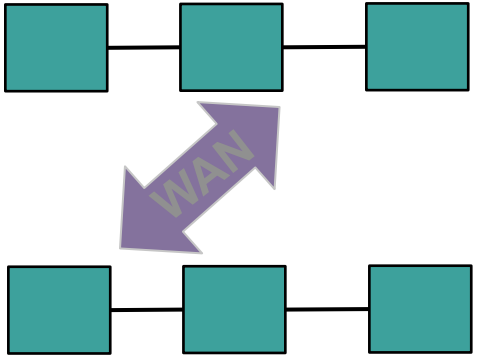
Pivotal

# 3. Mission Critical Availability and Resiliency

Cluster resilience & failover



Cluster to cluster WAN connectivity




## Gire

*"We can track and collect money at our 4,000+ kiosks and branches – even without a reliable Internet connection. GemFire provides the core data grid and a significant amount of related functionality to help us handle this unreliable network problem"*

**Gustavo Valdez, Chief of Architecture and Development**

- **19 million** payment transactions per month
- **4000+** points of sale with intermittent Internet connectivity

# 4. Flexibility for Developers to Create Unique Apps

- Data Structures:
  - User-defined classes
  - Documents (JSON)
- Native language support:
  - Java, C++, C#
- API's
  - Java: Hashmap
  - Memcache
  - Spring Data GemFire
  - C++ Serialization API's
- Embedded query authoring
  - Object Query Language (OQL)
- Publish & subscribe framework for continous query & reliable asynchronous event queues
- App-server embedded functionality:
  - Web app session state caching
  - L2 Hibernate

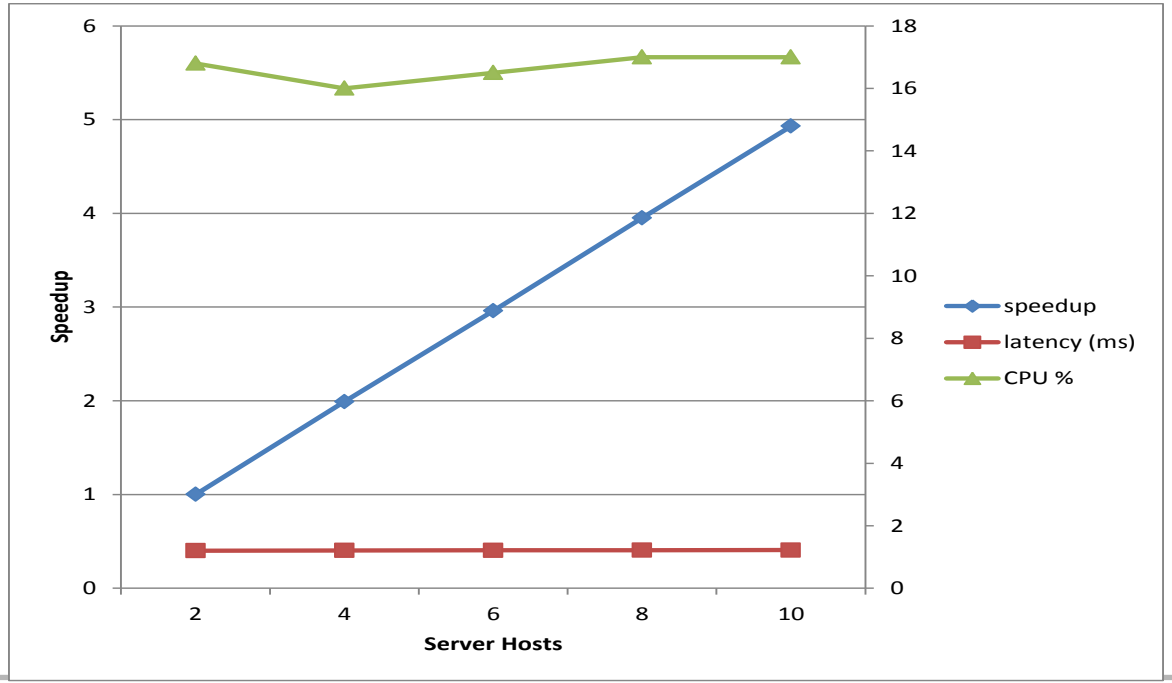
Pivotal

# What makes it fast?

- Design center is RAM not HDD
- Really demanding customers
- Avoid and minimize, particularly on the critical read/write paths:
  - Network hops, copying data, contention, distributed locking, disk seeks, garbage
- Lots and lots of testing
  - Establish and monitor performance baselines
  - Distributed systems testing is difficult!

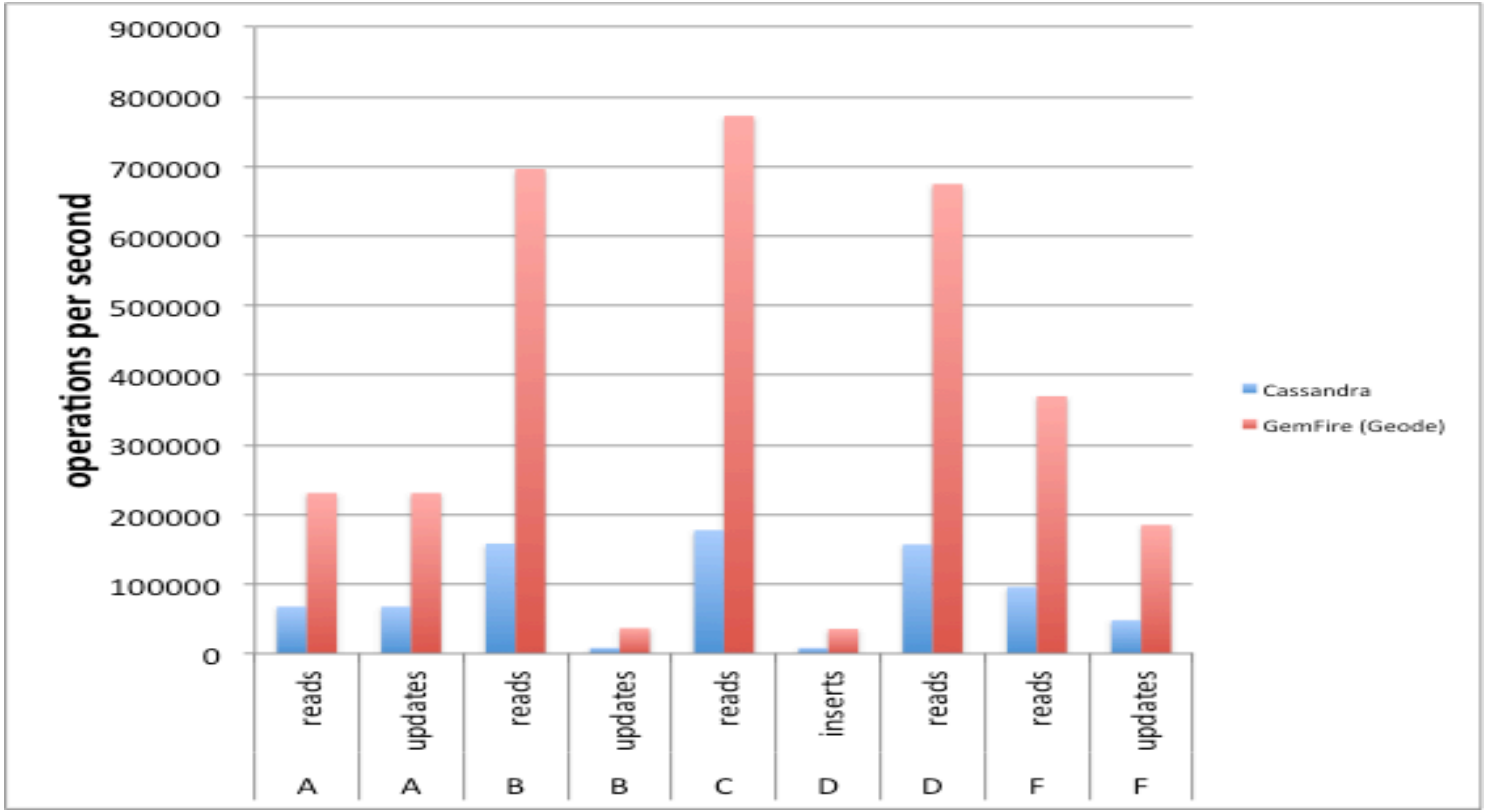
Pivotal™

# Horizontal Scaling for GemFire (Geode) Reads With Consistent Latency and CPU




- Scaled from 256 clients and 2 servers to 1280 clients and 10 servers
- Partitioned region with redundancy and 1K data size

# GemFire (Geode) 3.5-4.5X Faster Than Cassandra for YCSB




Pivotal

# Apache Geode (incubating)

## The open source core of GemFire

Pivotal™

# Geode Will Be A Significant Apache Project

- Over a 1000 person years invested into cutting edge R&D
- Thousands of production customers in very demanding verticals
- Cutting edge use cases that have shaped product thinking
- Tens of thousands of distributed , scaled up tests that can randomize every aspect of the product
- A core technology team that has stayed together since founding
- Performance differentiators that are baked into every aspect of the product

Pivotal™

# Geode or GemFire?

- Geode is a *project*, GemFire is a *product*
- We donated everything but the kitchen sink*
- More code drops imminent; going forward all development happens OSS-style (“The Apache Way”)

* Multi-site WAN replication, continuous queries, and native (C/C++) client driver

Pivotal™

GEODE

Getting Started Community Contribute Download

Performance is key.
Consistency is a must.

Geode is the open source distributed, in-memory database for scale-out applications.

# geode.incubator.apache.org

Pivotal

# Why OSS? Why Now? Why Apache?

- Open Source Software is fundamentally changing buying patterns
  - Developers have to endorse product selection (No longer CIO handshake)
  - Open source credentials attract the best developers
  - Open Source has replaced standards
- Align with the tides of history
  - Customers increasingly asking to participate in product development
  - Allow product development to happen with full transparency
- Apache Way
  - “Community over code”
  - Use cases far beyond Pivotal Gemfire’s

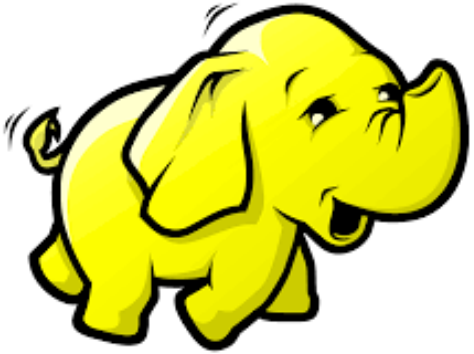
Pivotal™

# Beyond Gemfire’s core

Pivotal™

# Roadmap

- HDFS persistence
- Off-heap storage
- Lucene indexes
- Spark integration
- Cloud Foundry service

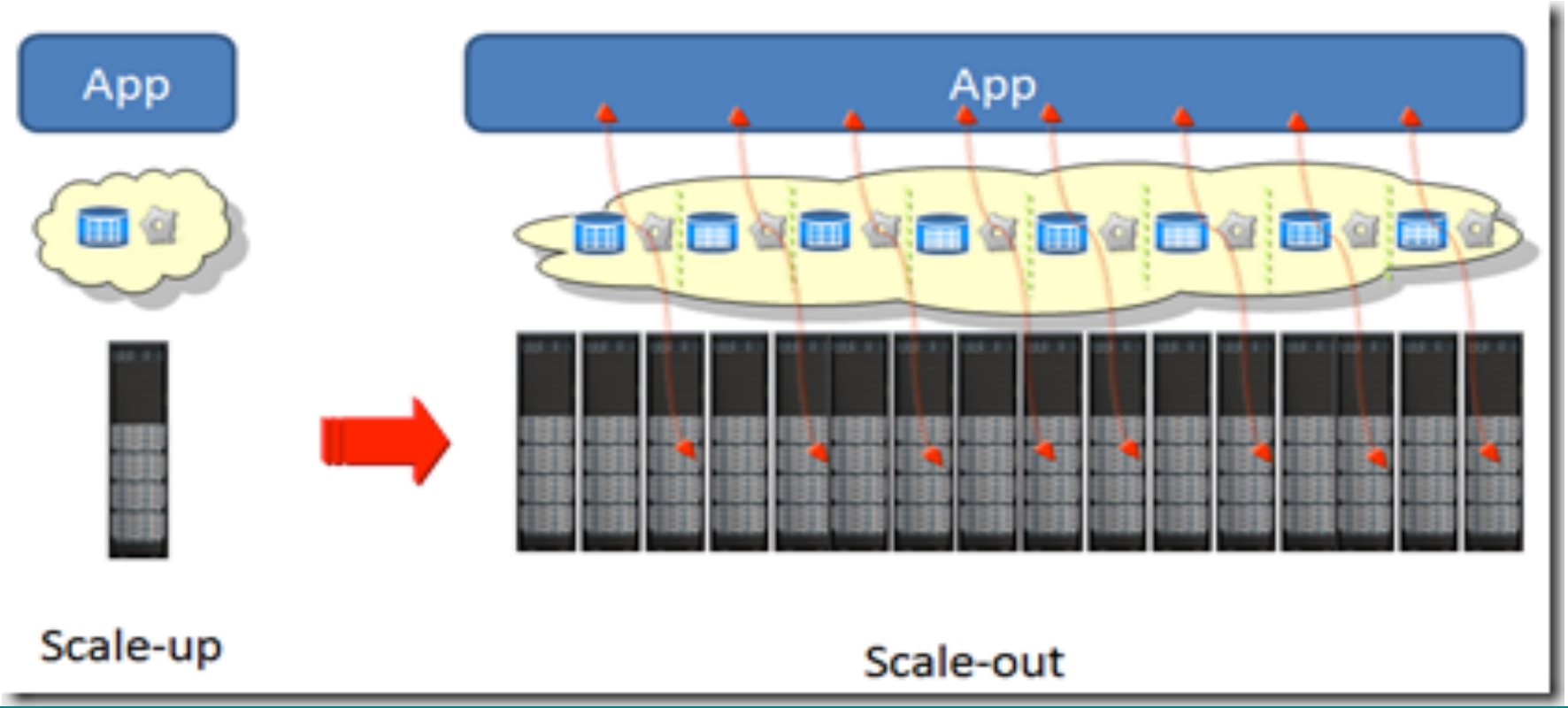


…and other ideas from the Geode community!

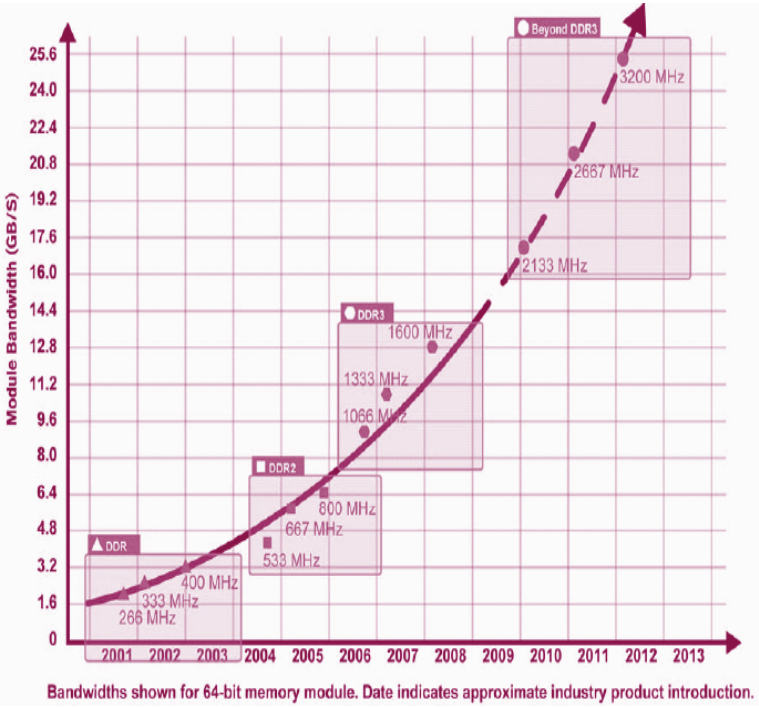
# Geode in modern data infrastructure

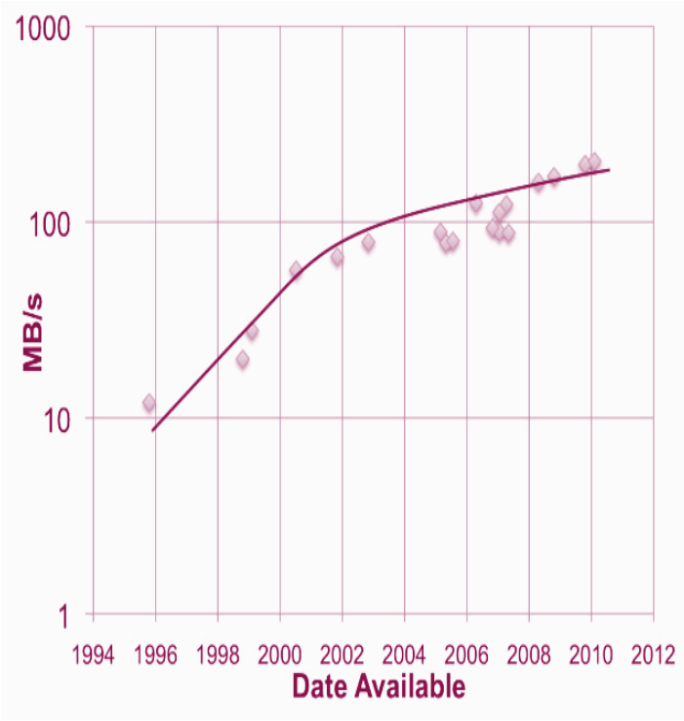
Pivotal™

# Infrastructure is increasingly Scale-Out



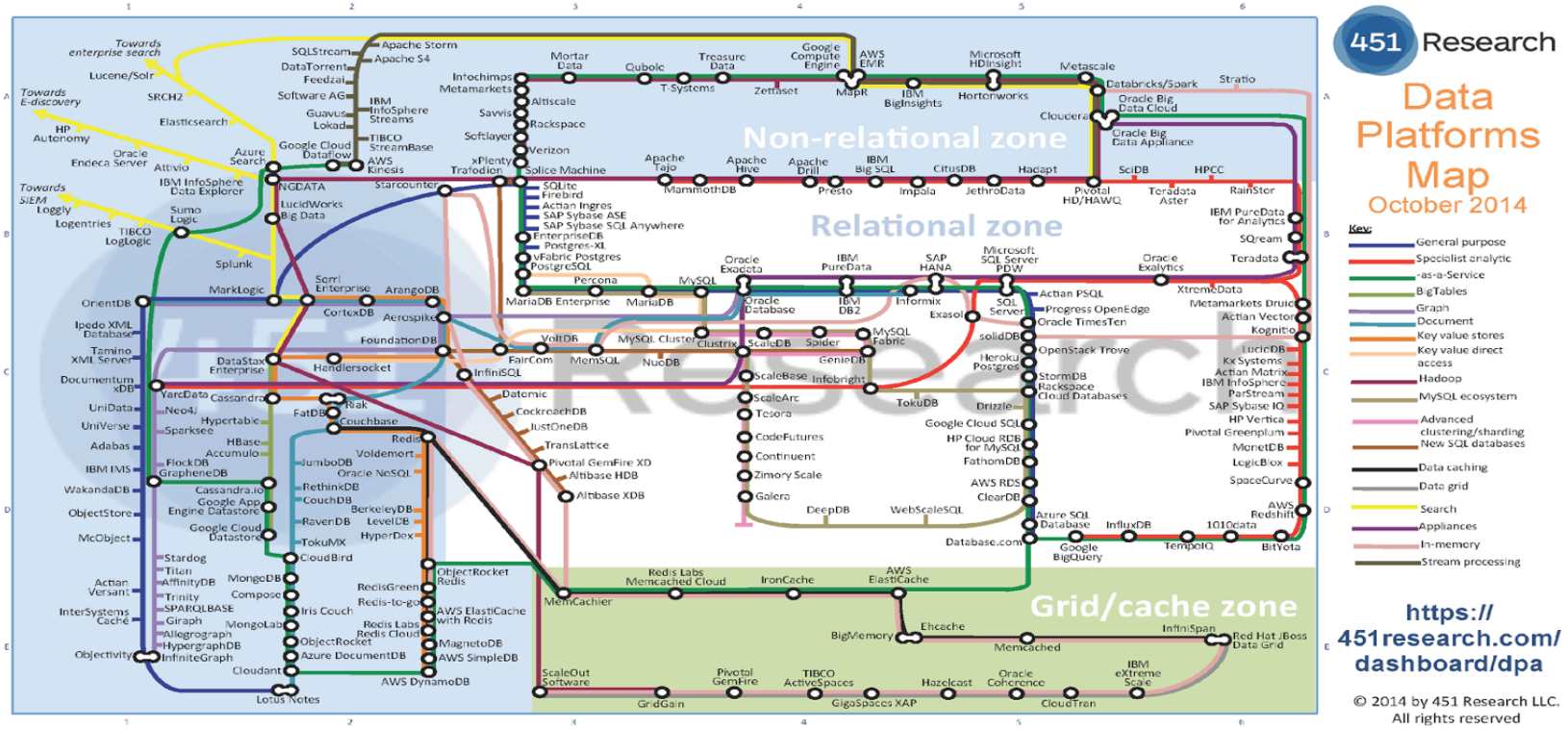
# Memory throughput growth



# Memory hierarchy getting deeper

# One-size-fits-all Data Platform Era is Over



# Analytics moving from Batch to Real-Time

# What happened?

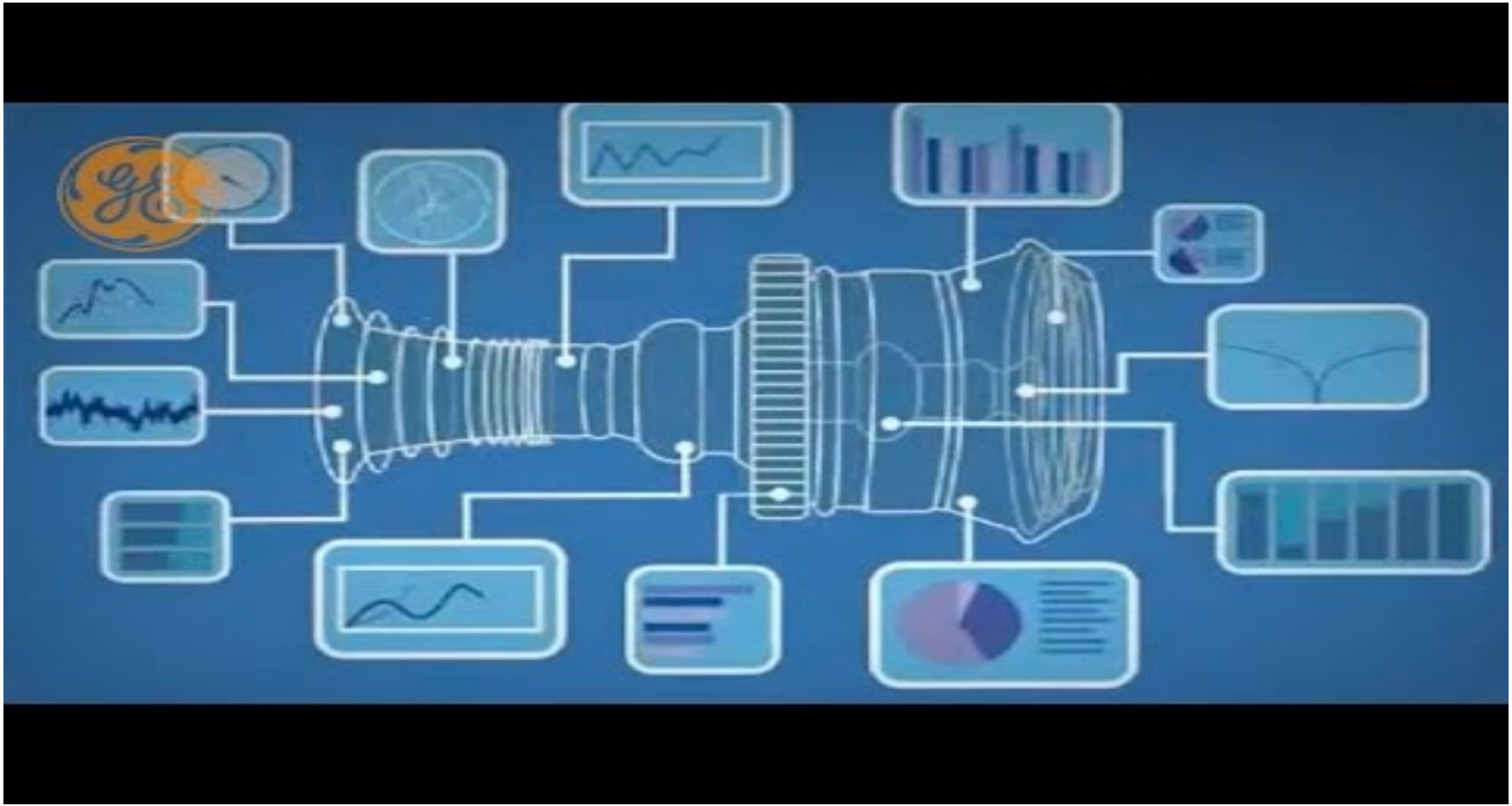
Pivotal™

Aggregators Folksonomy Wikis
Blogs Participation Six Degrees User Centered Joy of Use
Pagerank XFN Usability Widgets
Recommendation Social Software FOAF Browser
Sharing Collaboration Perpetual Beta Simplicity
Videocasting Podcasting AJAX
Audio IM Video Design
Web 2.0
Convergence CSS Pay Per Click
UMTS Mobility Atom XHTML SVG Ruby on Rails VC Trust Affiliation
OpenAPIs RSS Semantic Web Standards SEO Economy
OpenID Remixability REST Standardization The Long Tail
DataDriven Accessibility XML
Microformats Syndication
Modularity SOAP

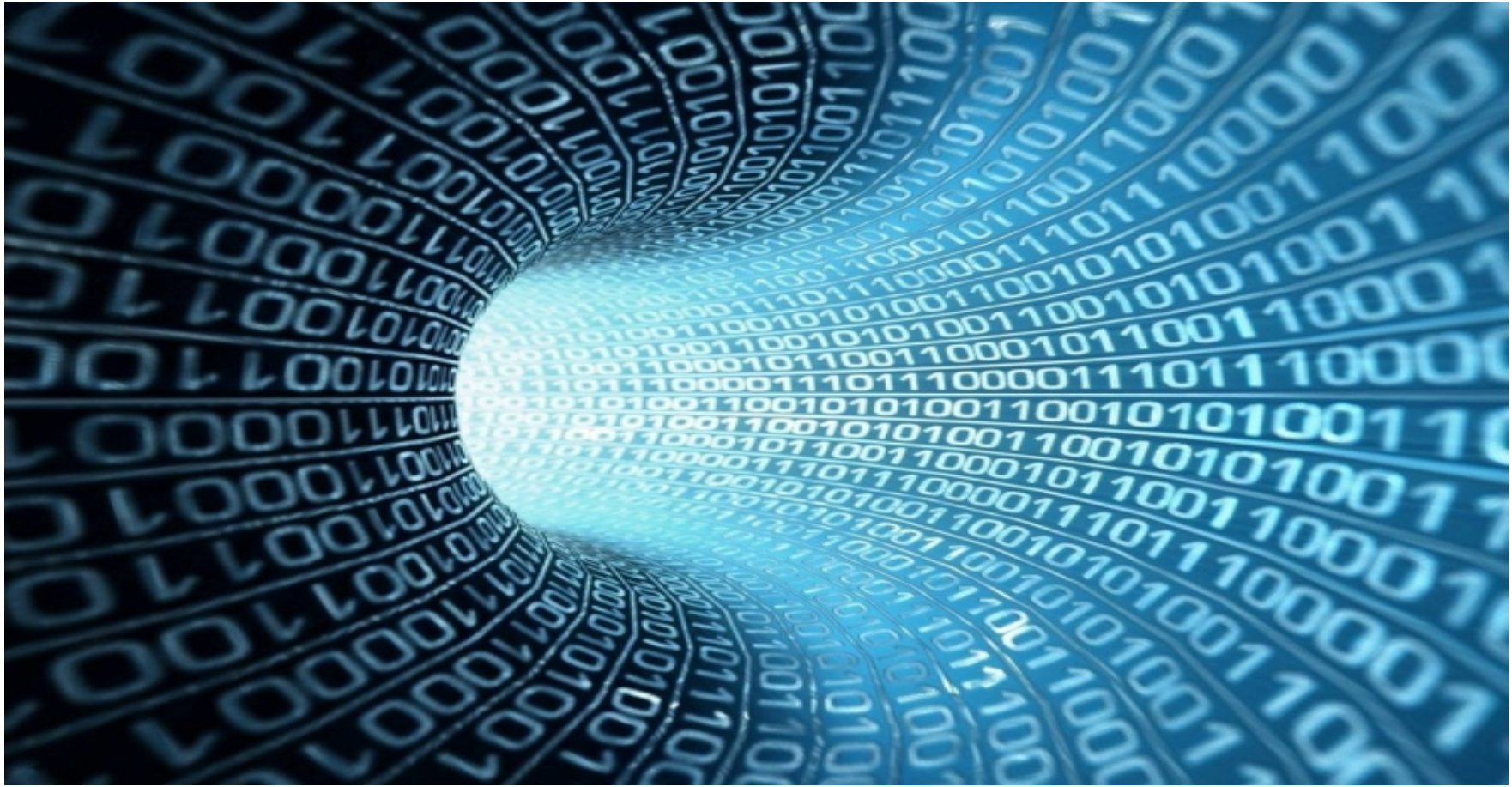
Pivotal™


Pivotal


Pivotal



Pivotal



Pivotal


Pivotal

# And, then…

MapReduce

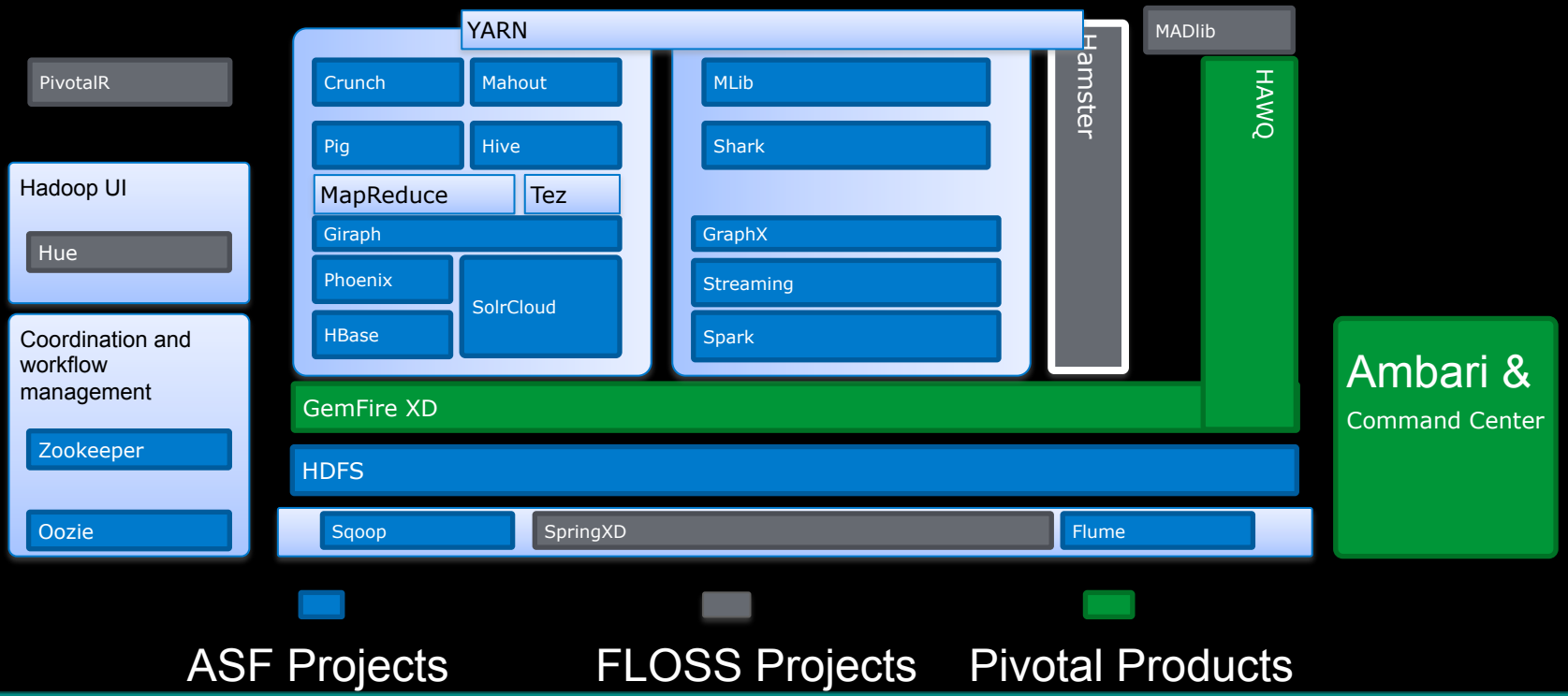
HDFS

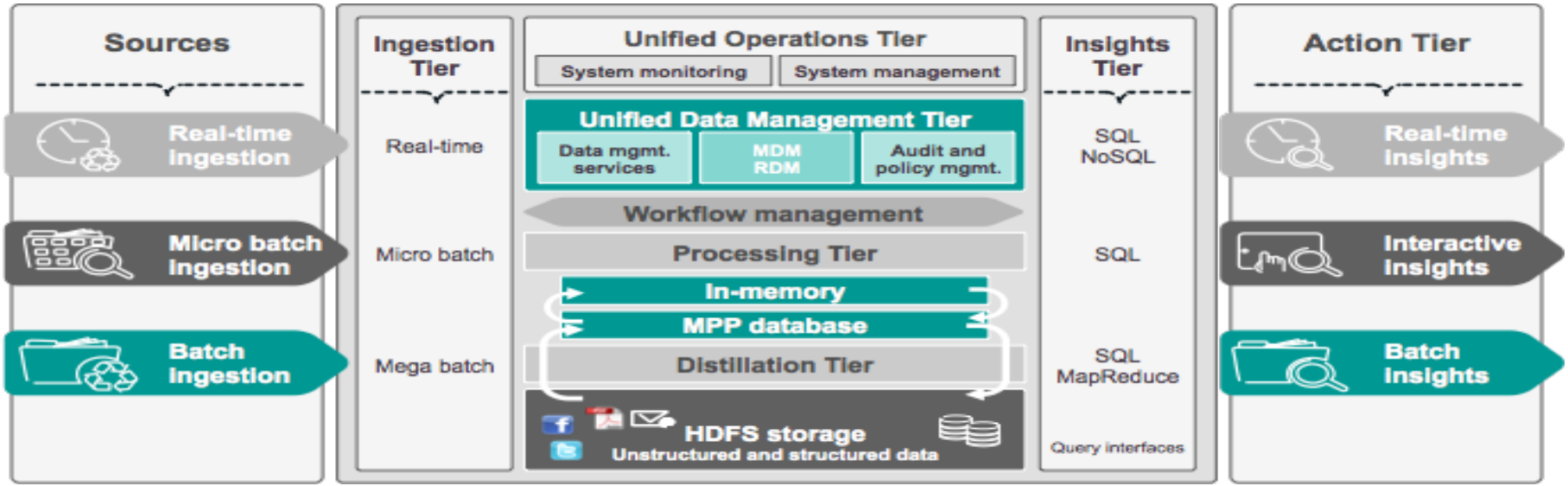
ASF Projects

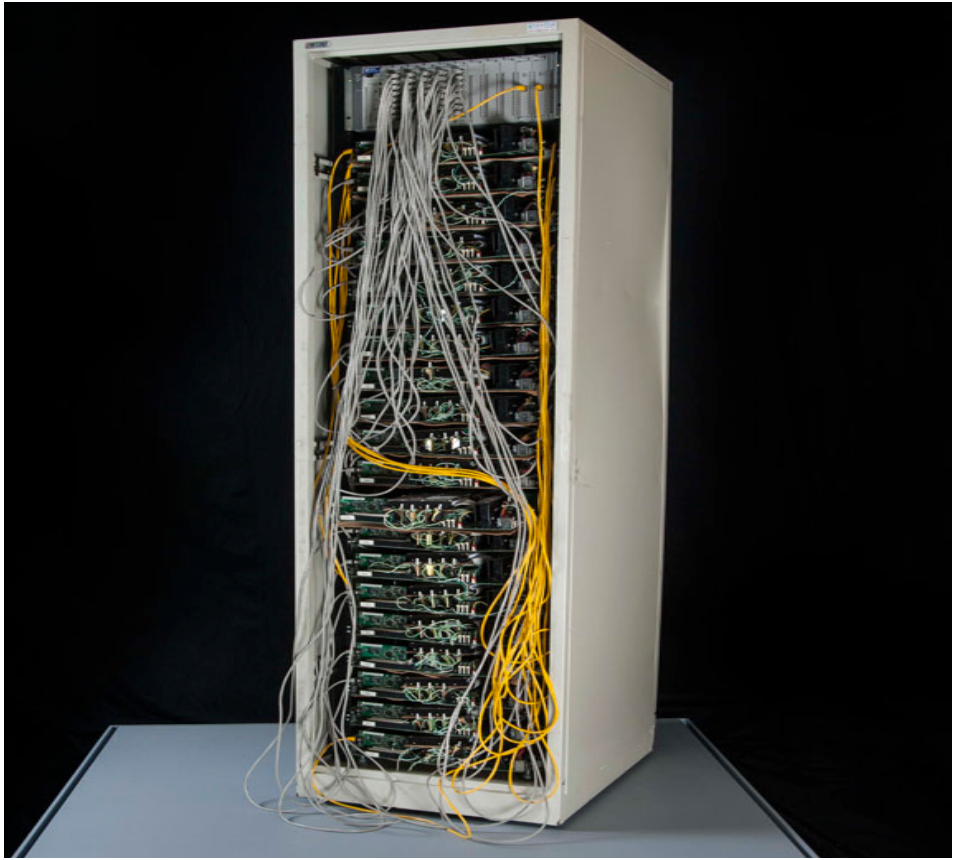
FLOSS Projects

Pivotal Products

# In a blink of an eye…







Pivotal

Pivotal™

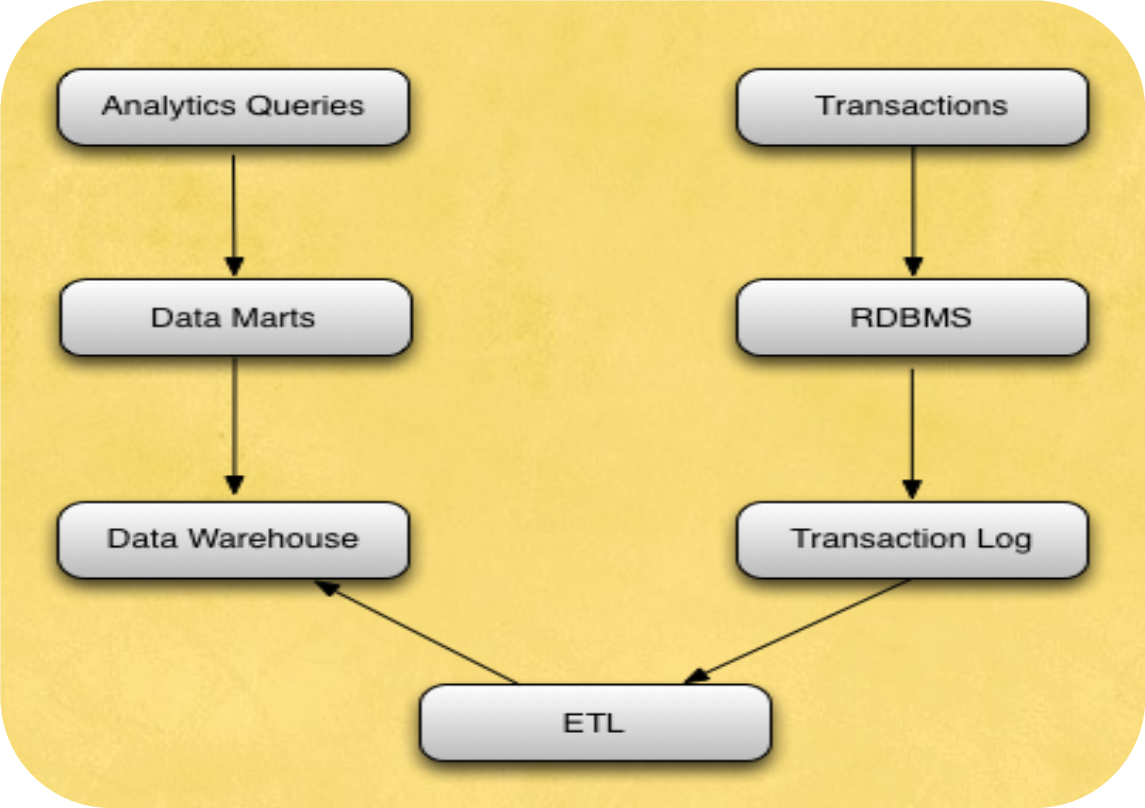
# Tape is Dead
# Disk is Tape
# Flash is Disk
# RAM Locality is King

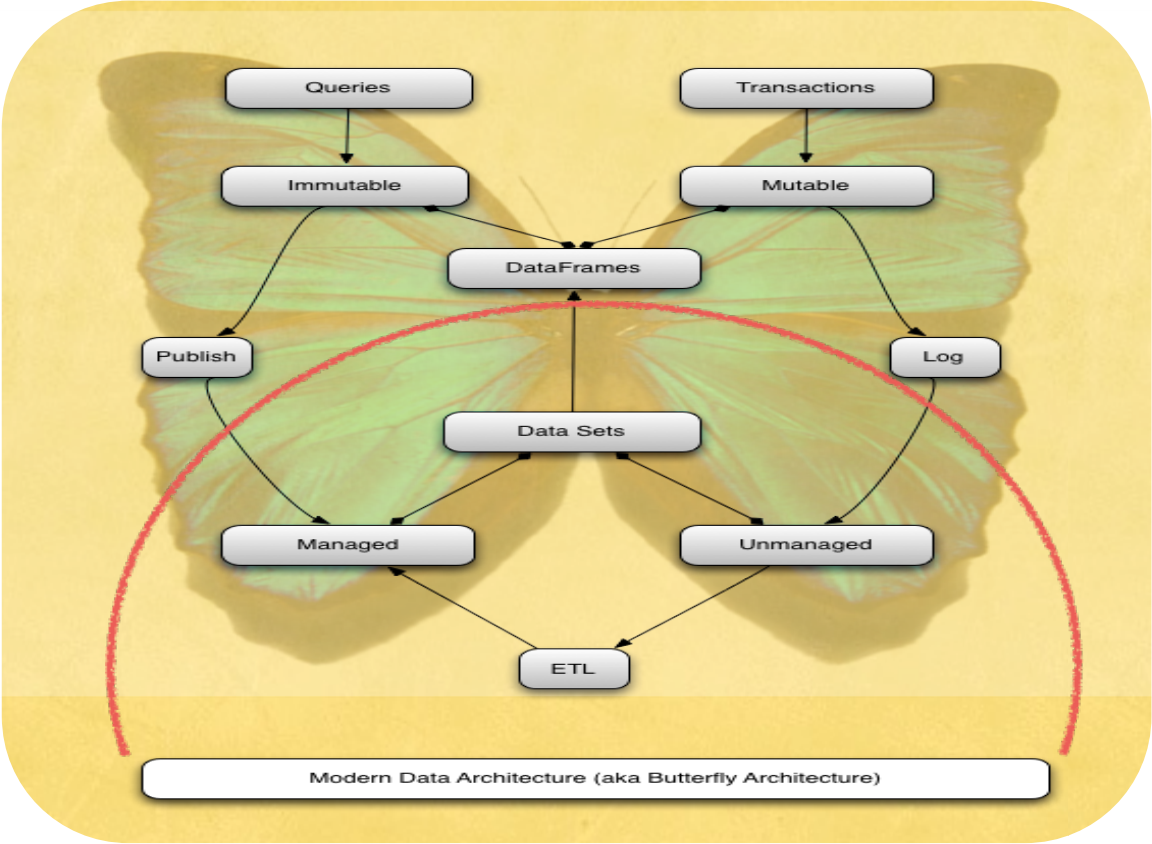
Jim Gray

Microsoft

December 2006

Pivotal

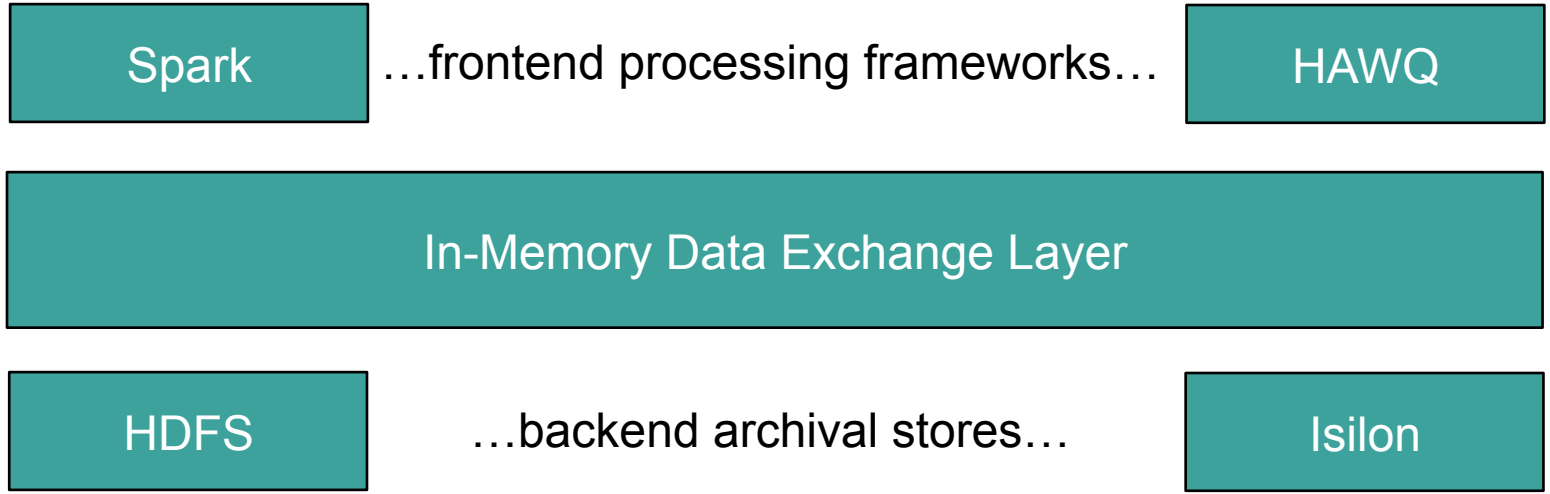




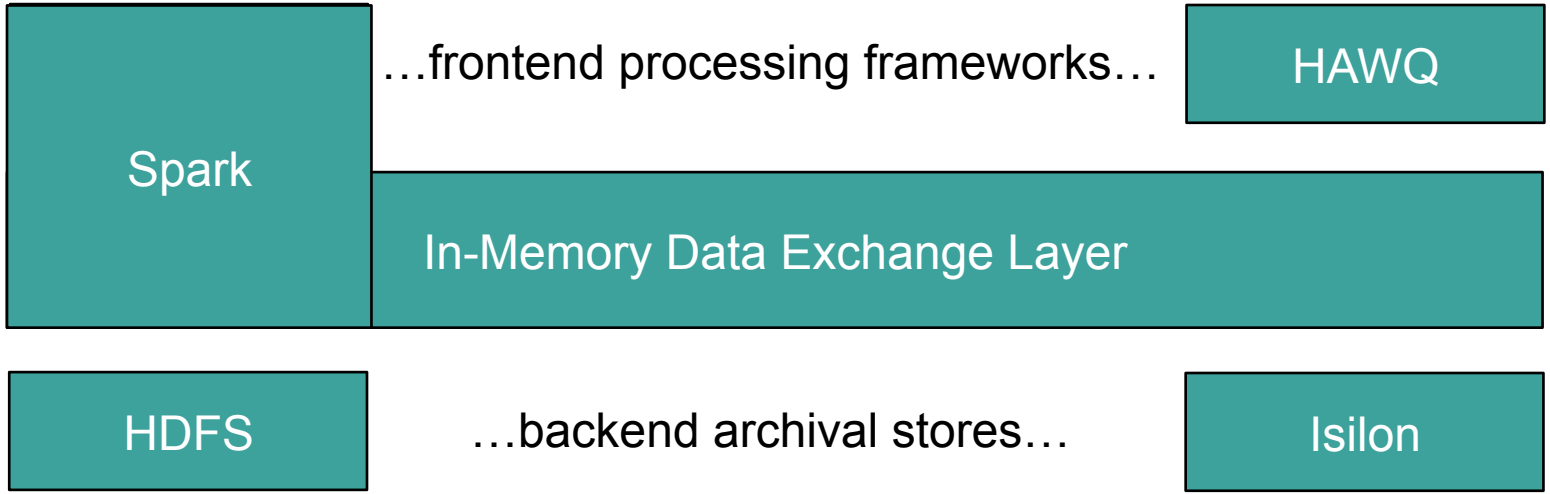
# The *missing* building block

Pivotal™

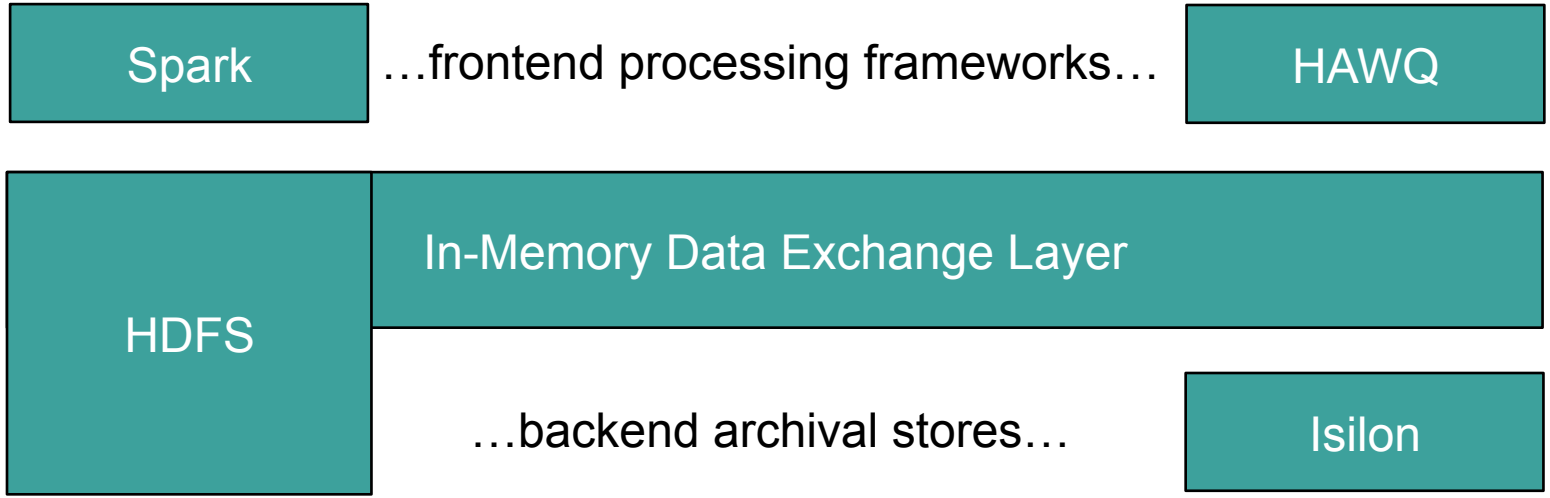
# Sharing operational data at the speed of RAM



# Spark's view on how to lock you in



# HDFS's view on how to lock you in



# How is open source solving this?

Pivotal™

# Short list of open source contenders

- Tachyon
- Infinispan
- Apache Ignite (incubating)
- Apache Geode (incubating)

Pivotal™

# Short list of open source contenders

- ~~Tachyon~~
- ~~Infinispan~~
- Apache Ignite (incubating)
- Apache Geode (incubating)

Pivotal™

# Geode's secret sauce

- Community!
- Maturity
- Scalability
- Building blocks

Pivotal™

# A few key building blocks

- PDX Serialization
- Asynchronous Events

Pivotal™

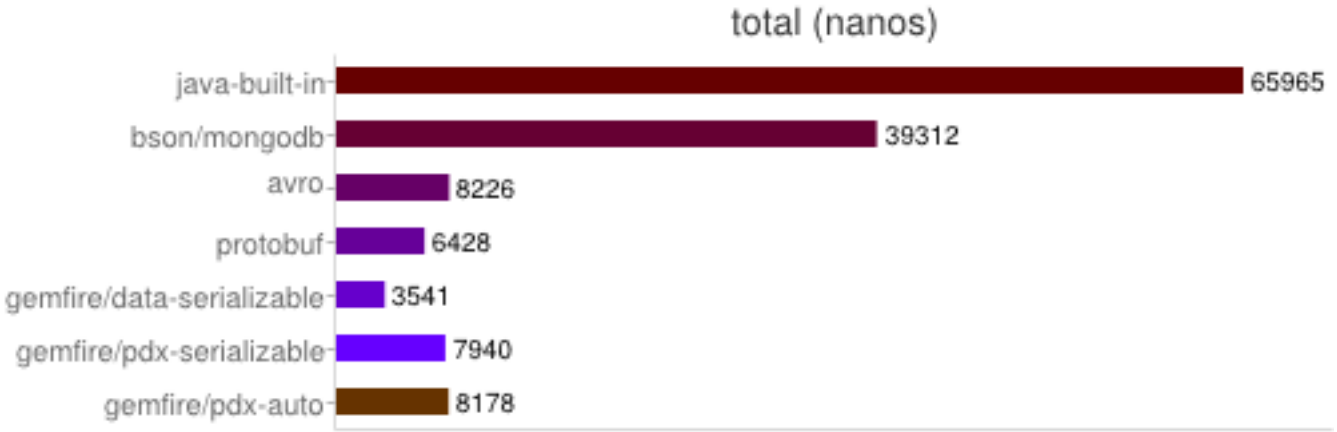
# Fixed or flexible schema?

| id | name | age | pet_id |
| --- | --- | --- | --- |
| | | | |
| | | | |

OR

```
{
  id   : 1,
  name : "Fred",
  age  : 42,
  pet  : {
    name : "Barney",
    type : "dino"
  }
}
```

# But how to serialize data?




http://blog.pivotal.io/pivotal/products/data-serialization-how-to-run-multiple-big-data-apps-at-once-with-gemfire

Pivotal

C#, C++, Java, JSON

# Portable Data eXchange

```
|                header                |       data        |
| pdx | length | dsid | typeId | fields | offsets |
```

Pivotal

# Efficient for queries

```
SELECT p.name from /Person p WHERE p.pet.type = "dino"
```

```
{
  id   : 1,
  name : "Fred",
  age  : 42,
  pet  : {
    name : "Barney",
    type : "dino"
  }
}
```

single field deserialization

# Easy to use

- Access from Java, C#, C++, JSON
- Domain objects not required
- Automatic type definition
  - No IDL compiler or schema required
  - No hand-coded read/write methods

Pivotal™

# Asynchronous Events – Design Goals

- High availability
- Low latency, high throughput
- Deliver events to a receiver without impacting the write path

Pivotal™

# Questions?

- http://geode.incubator.apache.org
- dev@geode.incubator.apache.org
- user@geode.incubator.apache.org
- http://github.com/apache/incubator-geode

Pivotal

# Bonus Content

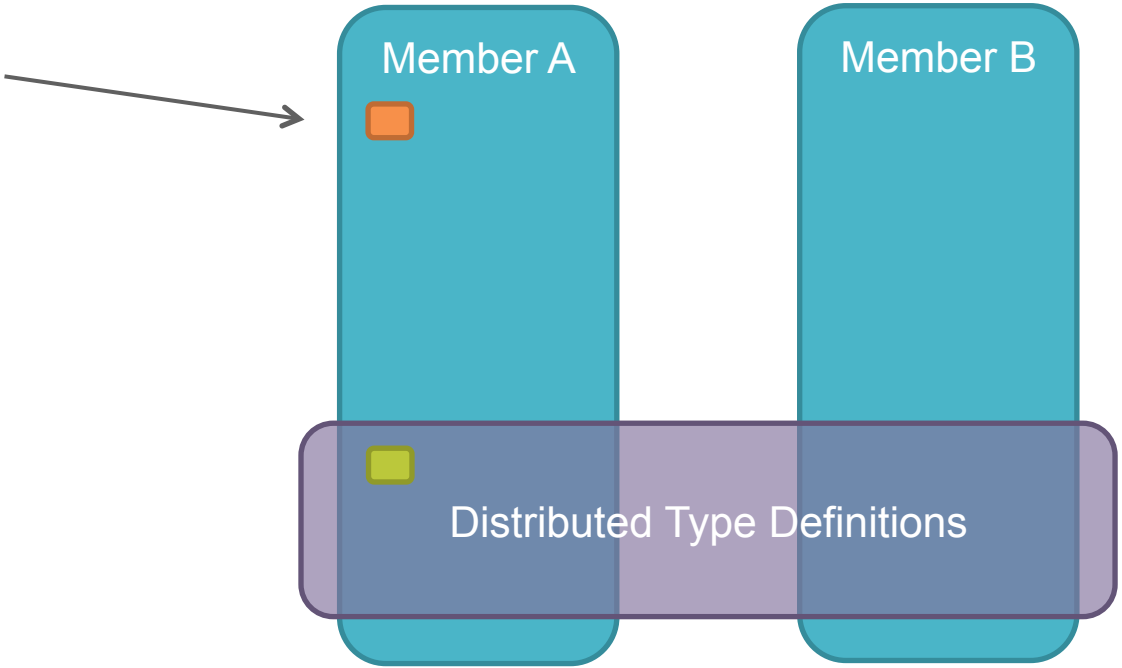
Pivotal™

# PDX

Pivotal™

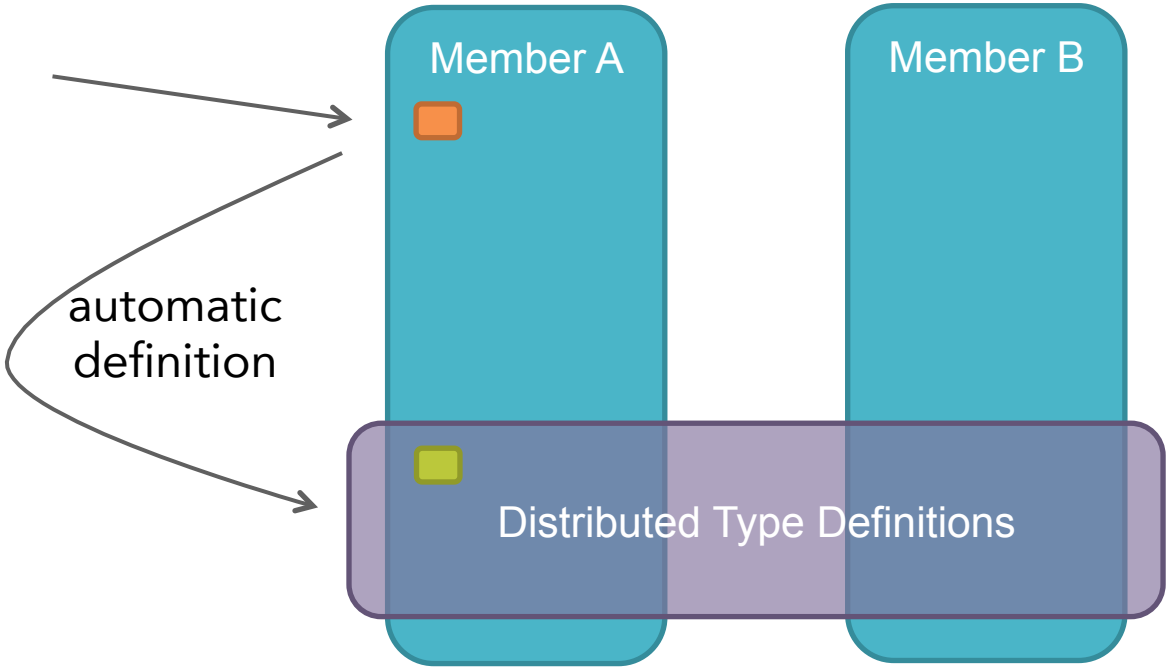
# Distributed type registry

```
Person p1 = …
region.put("Fred", p1);
```



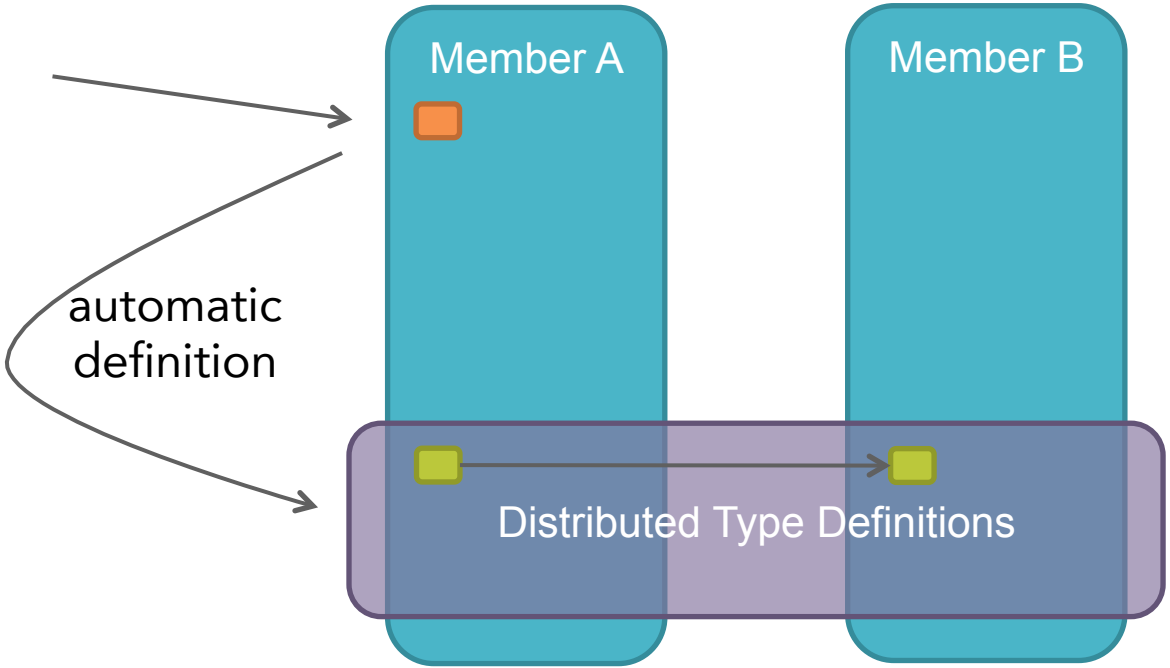
# Distributed type registry

```
Person p1 = …
region.put("Fred", p1);
```



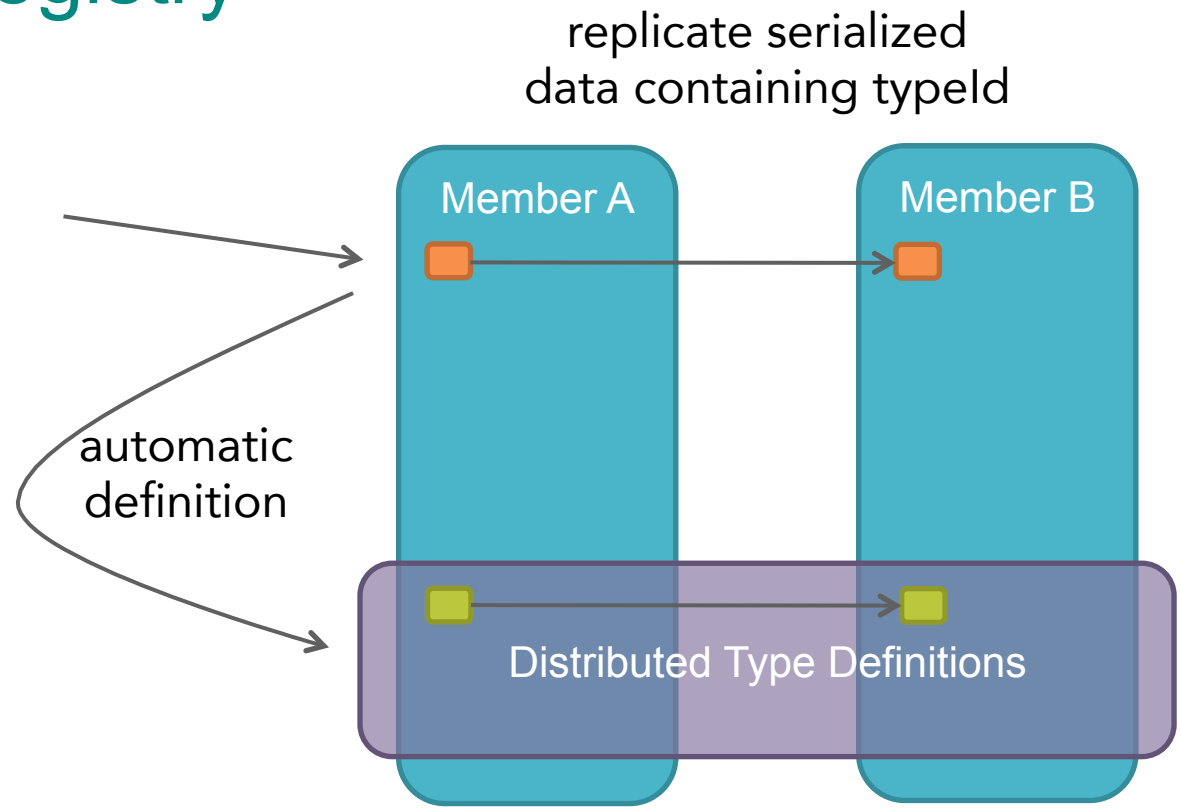
# Distributed type registry

```
Person p1 = …
region.put("Fred", p1);
```



# Distributed type registry

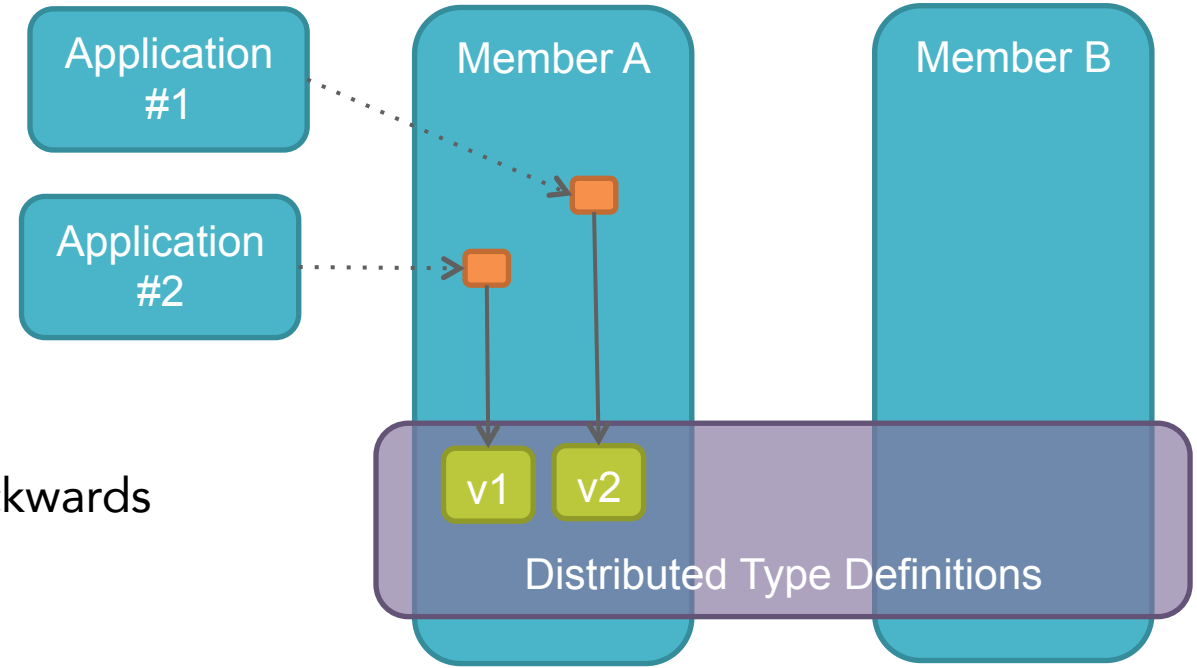
```
Person p1 = …
region.put("Fred", p1);
```



# Schema evolution

v2 objects preserve data from missing fields

v1 objects use default values to fill in new fields

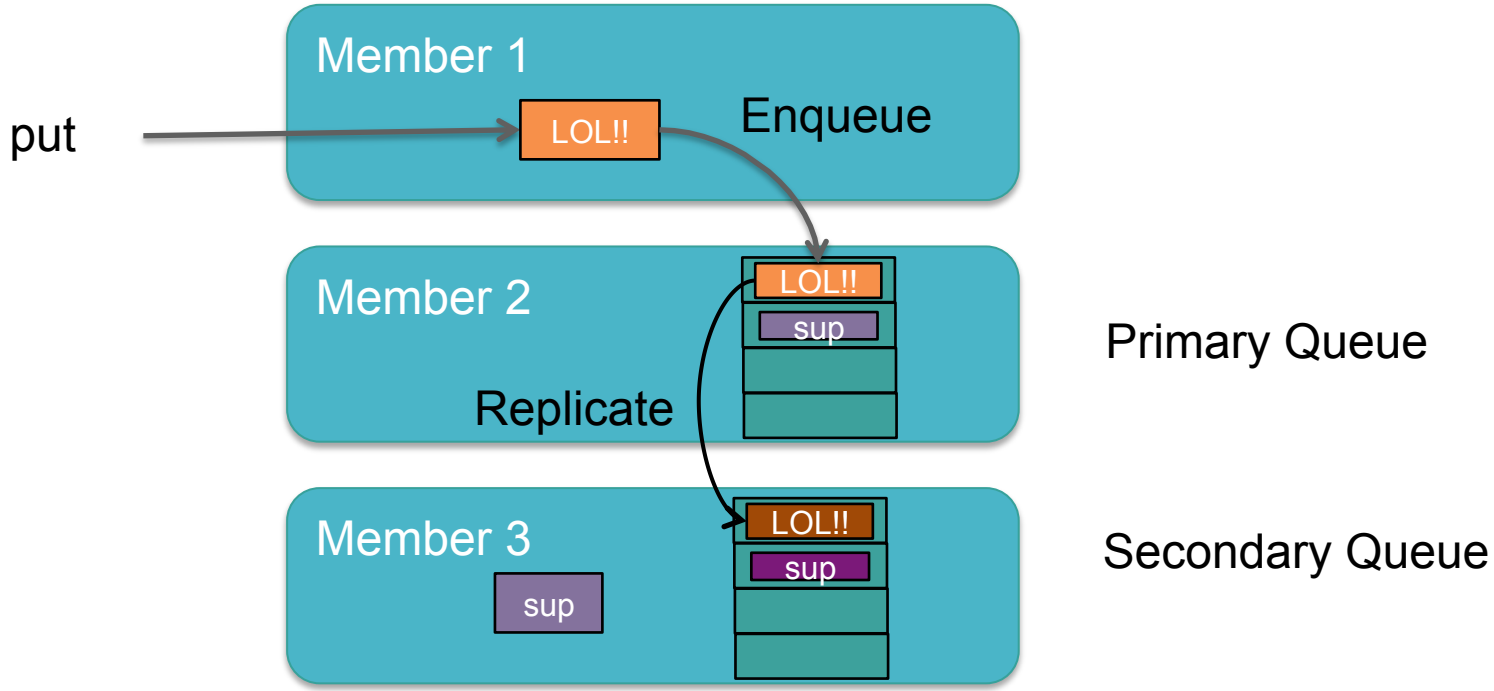



PDX provides forwards and backwards compatibility, no code required

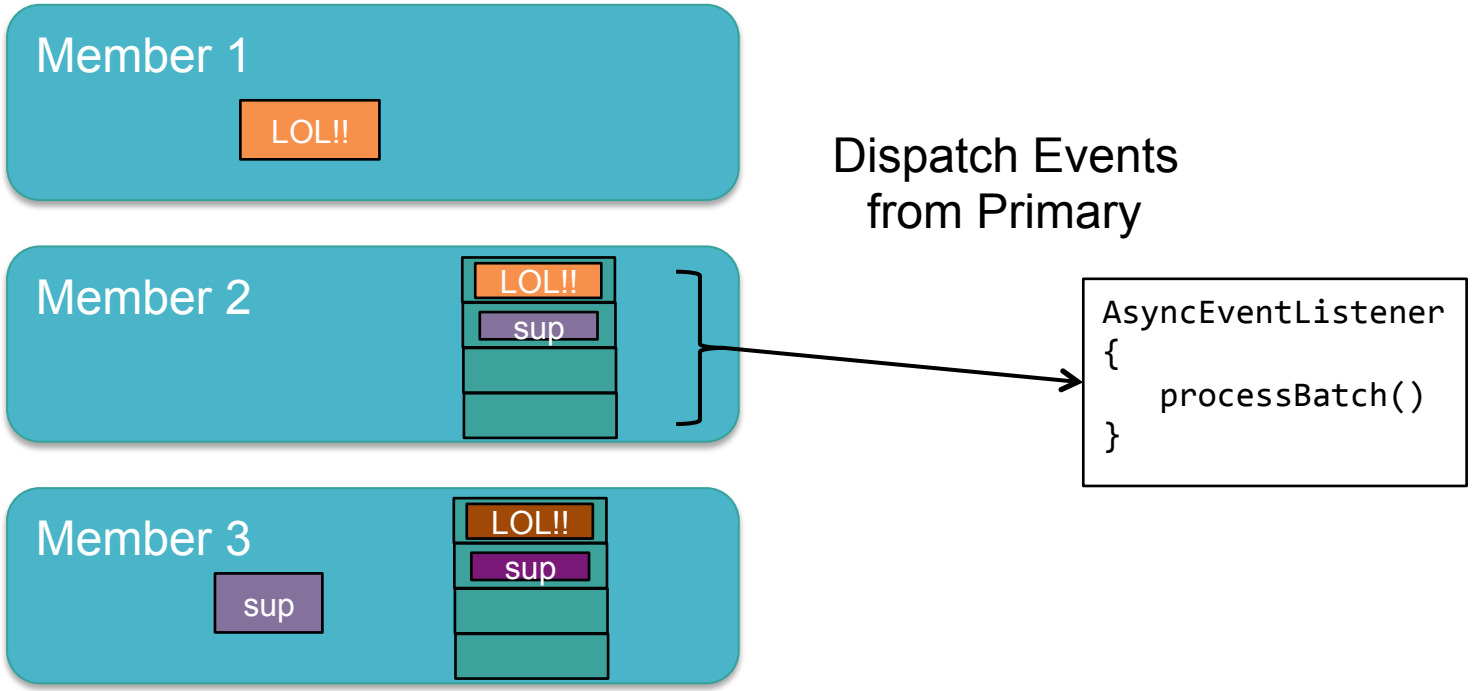
# Asynchronous Events

Pivotal™

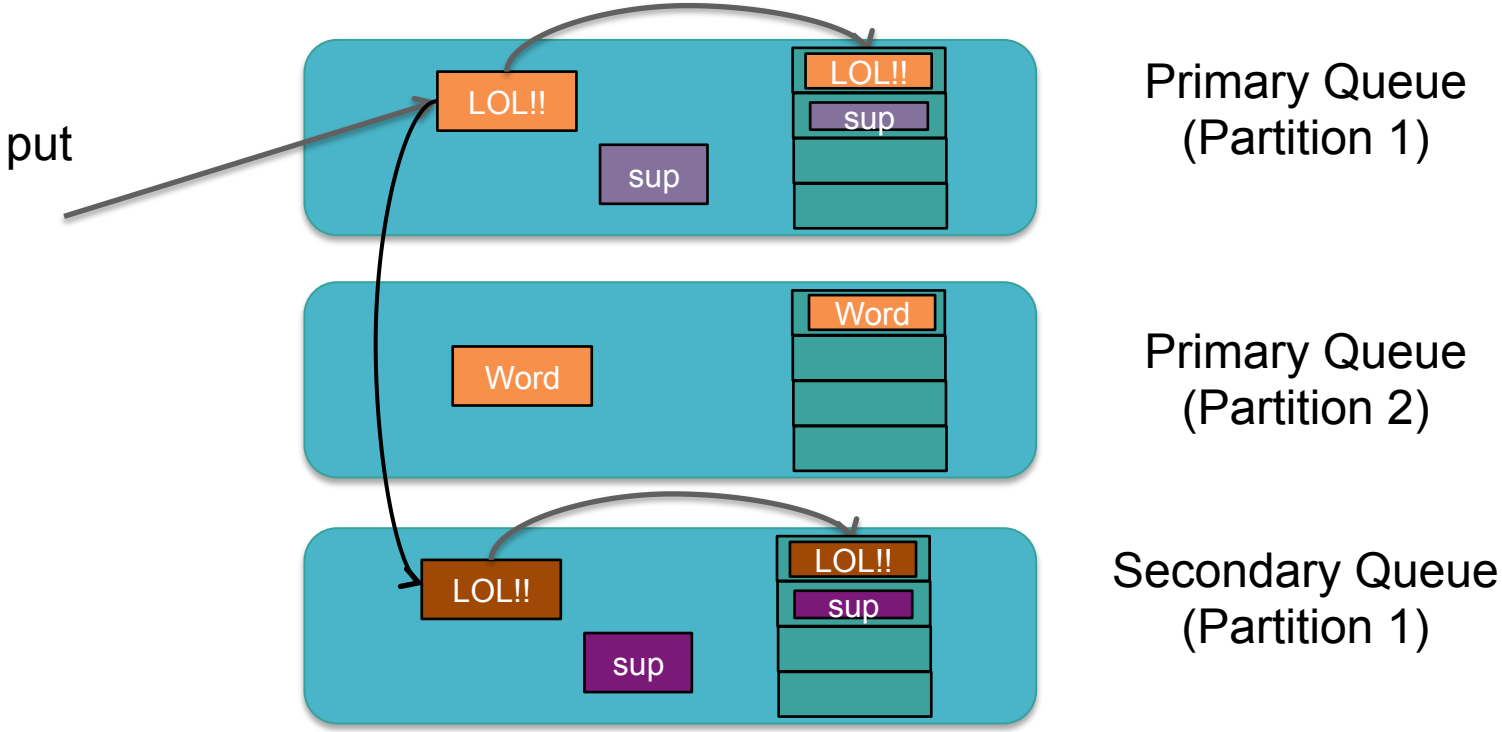
# Serial Queues



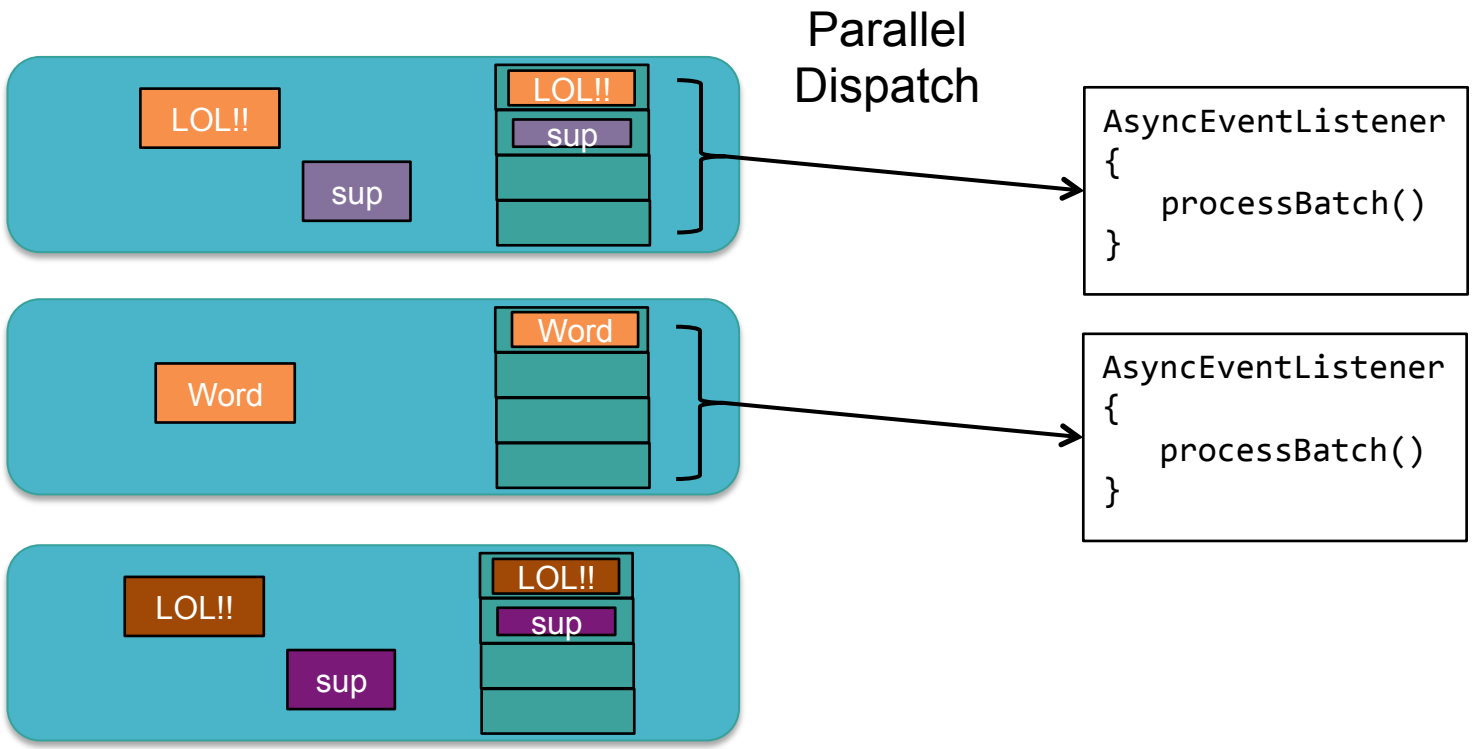
# Serial Queues



# Parallel Queues



# Parallel Queues



"There are only two hard things in Computer Science: **cache invalidation**, naming things, and off-by-one errors."

– the internet

Pivotal™

# Client driver

- Intelligent – understands data distribution for single hop network access
- Caching – can be configured to locally cache data for even faster access
- Events – `registerInterest()` in keys to receive push notifications

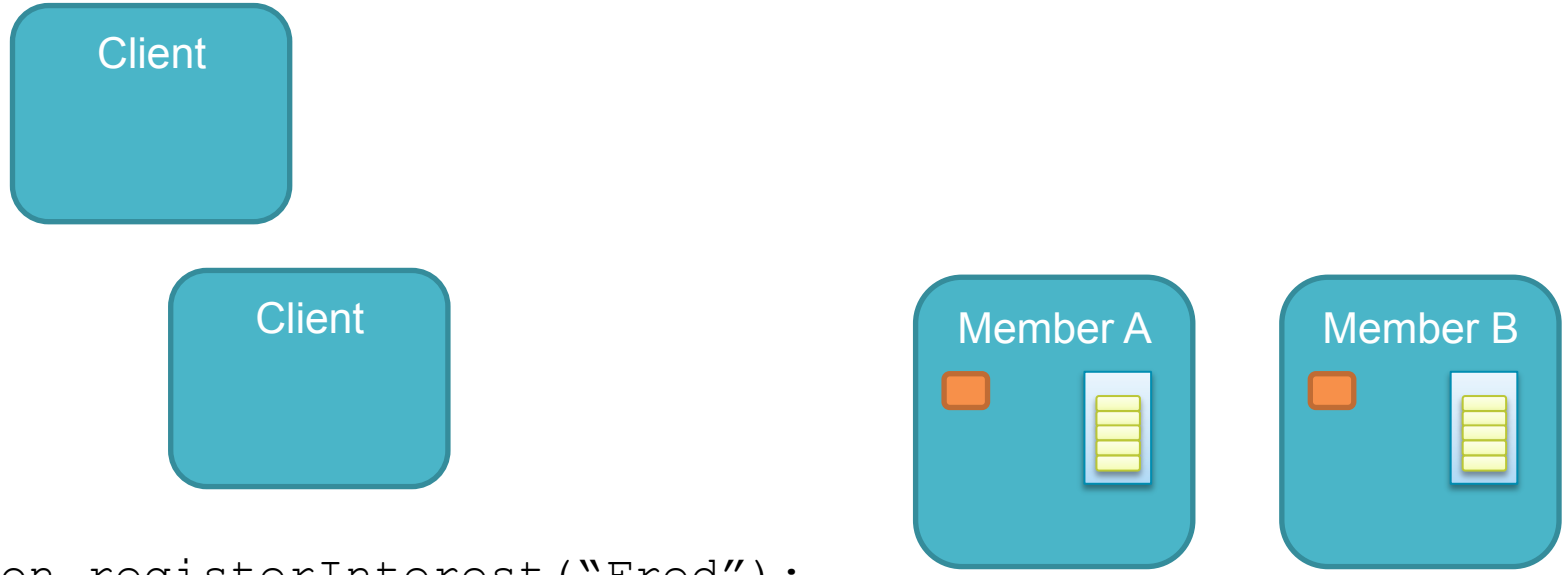
Pivotal

# Client subscriptions

- Useful for refreshing client cache, pushing events (“topics”) to multiple clients
- Highly available & scalable via in-memory replicated queues
- Events are ordered, at-least-once delivery
- Durable subscriptions, conflation optional

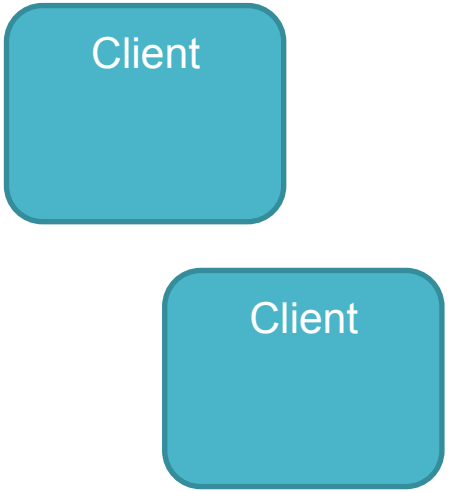
Pivotal™

# Client subscriptions

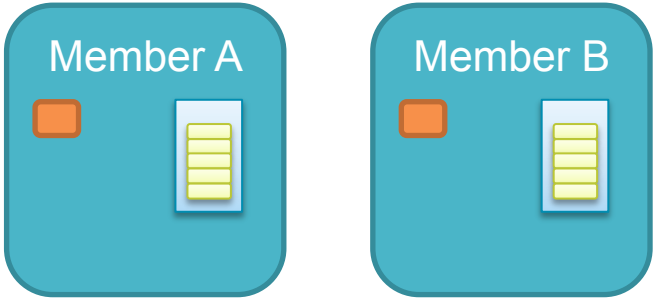



```
region.registerInterest("Fred");
```

# Client subscriptions

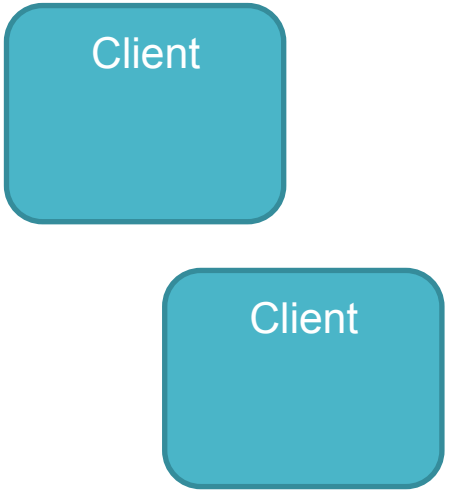



```
Person p1 = …
region.put("Fred", p1);
```

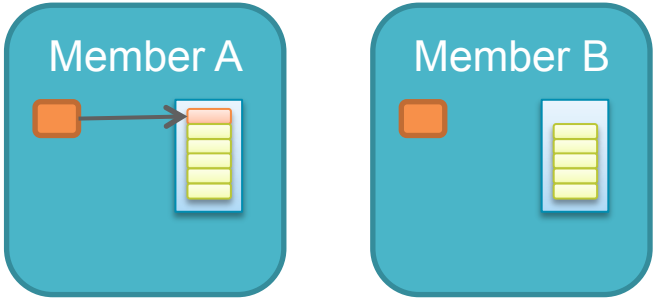



```
region.registerInterest("Fred");
```

# Client subscriptions

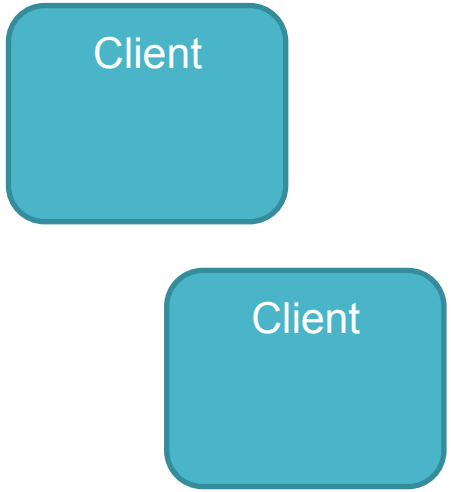



```
Person p1 = …
region.put("Fred", p1);
```

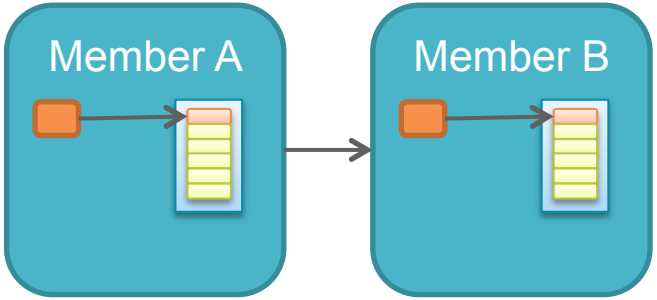



```
region.registerInterest("Fred");
```

# Client subscriptions

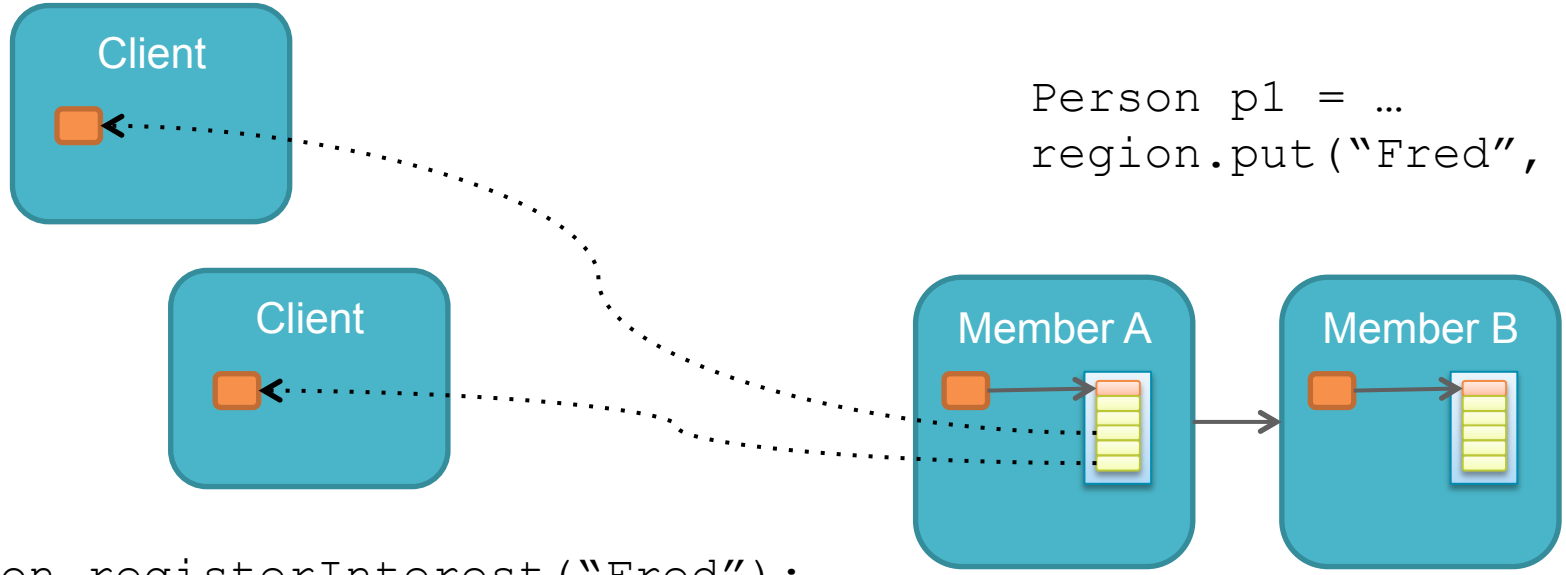



```
Person p1 = …
region.put("Fred", p1);
```




```
region.registerInterest("Fred");
```

# Client subscriptions




```
Person p1 = …
region.put("Fred", p1);
```

```
region.registerInterest("Fred");
```

# Use cases and patterns

Pivotal™

# “Low touch” Usage Patterns

**HTTP Session management**

**Simple template for TCServer, TC, App servers**

**Shared nothing persistence, Global session state**

**Hibernate L2 Cache plugin**

Set Cache in hibernate.cfg.xml

**Support for query and entity caching**

**Memcached protocol**

**Servers understand the *memcached* wire protocol**

**Use any *memcached* client**

**Spring Cache Abstraction**

```
<bean id="cacheManager"
class="org.springframework.data.gemfire.support.GemfireCacheManager"
```

Pivotal

# Application Patterns

- Caching for speed and scale
  - Read-through, Write-through, Write-behind
- Geode as the OLTP system of record
  - Data in-memory for low latency, on disk for durability
- Parallel compute engine
- Real-time analytics

Pivotal™

# Development Patterns

- **Configure** the cluster programmatically or declaratively using cache.xml, gfsh, or SpringDataGemFire

```
<cache>
  <region name="turbineSensorData" refid="PARTITION_PERSISTENT">
    <partition-attributes redundant-copies="1" total-num-buckets="43"/>
  </region>
</cache>
```

Pivotal™

# Development Patterns

- **Write** key-value data into a Region using { create | put | putAll | remove }
  - The value can be flat data, nested objects, JSON, …

```
Region sensorData = cache.getRegion("TurbineSensorData");

SensorKey key = new SensorKey(31415926, "2013-05-19T19:22Z"); // turbineId, timestamp
sensorData.put(key, new TurbineReading()
    .setAmbientTemp(75)
    .setOperatingTemp(80)
    .setWindDirection(0)
    .setWindSpeed(30)
    .setPowerOutput(5000)
    .setRPM(5));
```

Pivotal

# Development Patterns

- **Read** values from a Region by key using { get | getAll }

```
TurbineReading data = sensorData.get(new SensorKey(31415926,"2013-05-19T19:22Z"));
```

- **Query** values using OQL

```
// finds all sensor readings for the given turbine
SELECT * from /TurbineSensorData.entrySet WHERE key.turbineId = 31415926

// finds all sensor readings where the operating temp exceeds a threshold
SELECT * from /TurbineSensorData.entrySet WHERE value.operatingTemp > 120
```

- Apply indexes to optimize queries

Pivotal

# Development Patterns

- Execute **functions** to operate on local data in parallel
- Respond to updates using **CacheListeners** and **Events**

- Automatic redundancy, partitioning, distribution, consistency, network partition detection & recovery, load balancing, …

Pivotal™