Building scalable applications using Pivotal GemFire/Apache Geode

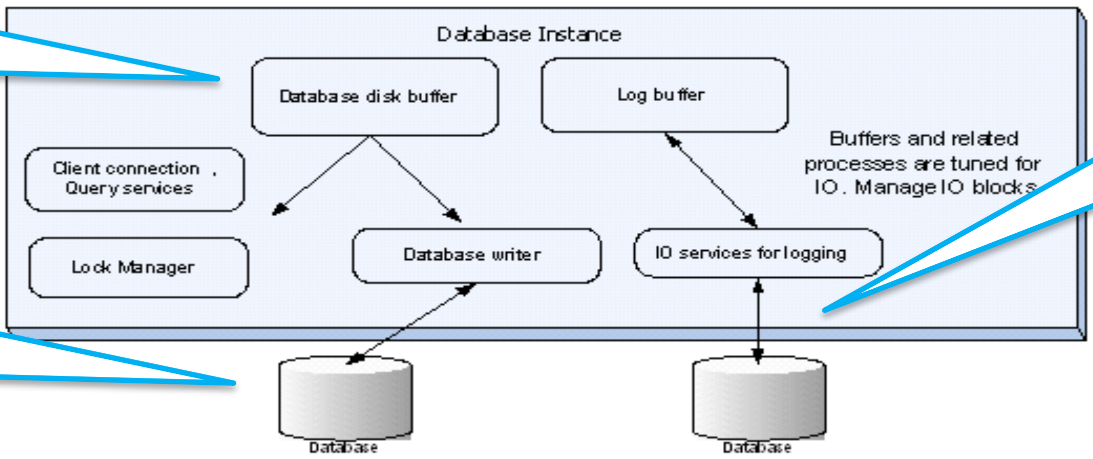
Yogesh Mahajan
ymahajan@apache.org

Pivotal™

Eliminate disk access in the real time path

We Challenge the traditional RDBMS design NOT SQL

Typical relational database architecture



Buffers primarily tuned for IO

First write to Log

Second write to Data Files

Too much I/O

Design roots don’t necessarily apply today

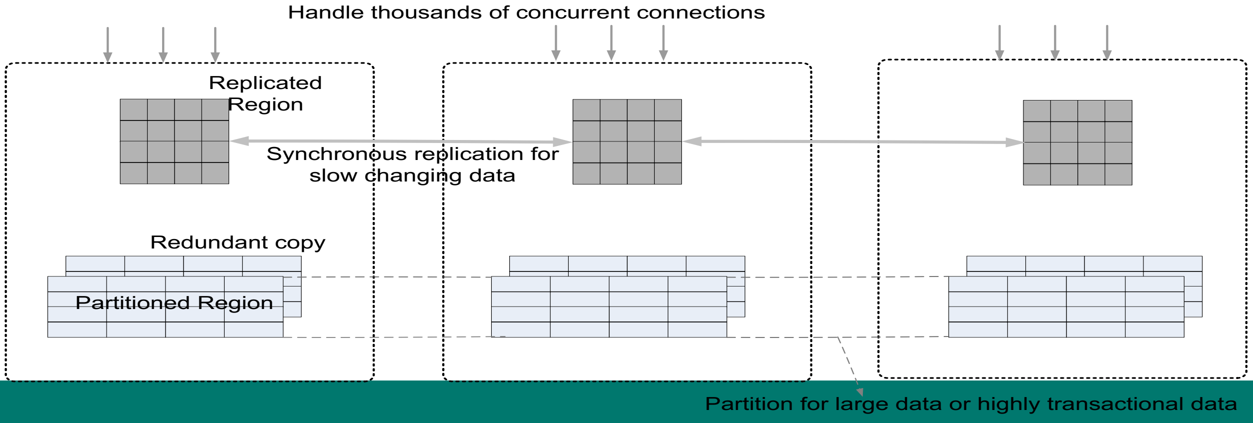
- **Too much focus on ACID**
- **Disk synchronization bottlenecks**

Pivotal

IMDG basic concepts

- Distributed memory oriented store
  - KV/Objects or JSON
  - Queryable, Indexable and transactional
- Multiple storage models
  - Replication, partitioning in memory
  - With synchronous copies in cluster
  - Overflow to disk and/or RDBMS

- Parallelize Java App logic
- Multiple failure detection schemes
- Dynamic membership (elastic)
- Vendors differentiate on
  - Query support, WAN, events, etc

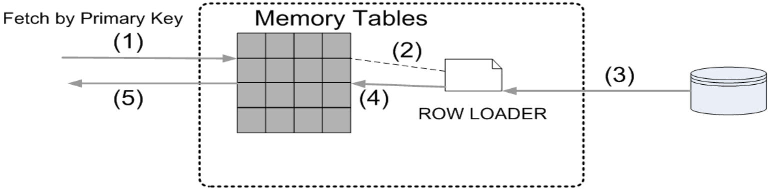

Handle thousands of concurrent connections
Replicated Region
Synchronous replication for slow changing data
Redundant copy
Partitioned Region
Partition for large data or highly transactional data

Low latency for thousands of clients

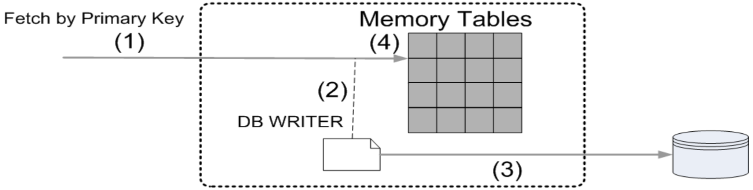
Key IMDG pattern - Distributed Caching

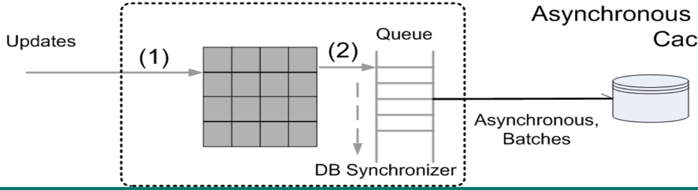
- Designed to work with existing RDBs
  - Read through: Fetch from DB on cache miss
  - Write through: Reflect in cache IFF DB write succeeds
  - Write behind: reliable, in-order queue and batch write to DB

Read through Cache



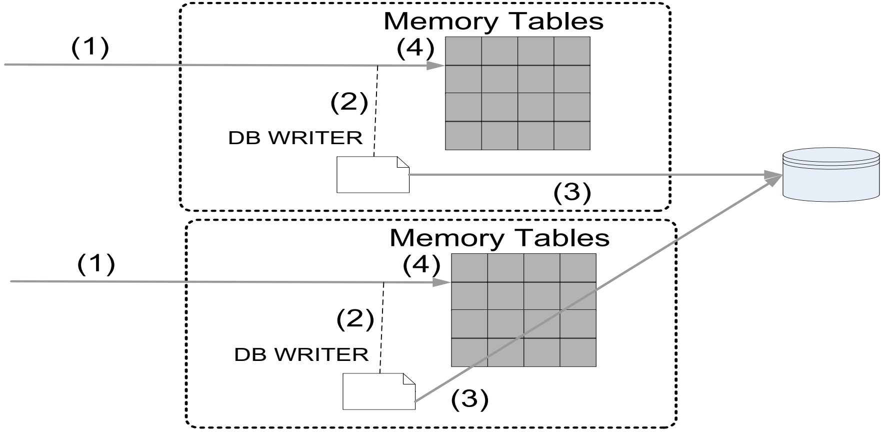
Write through Cache

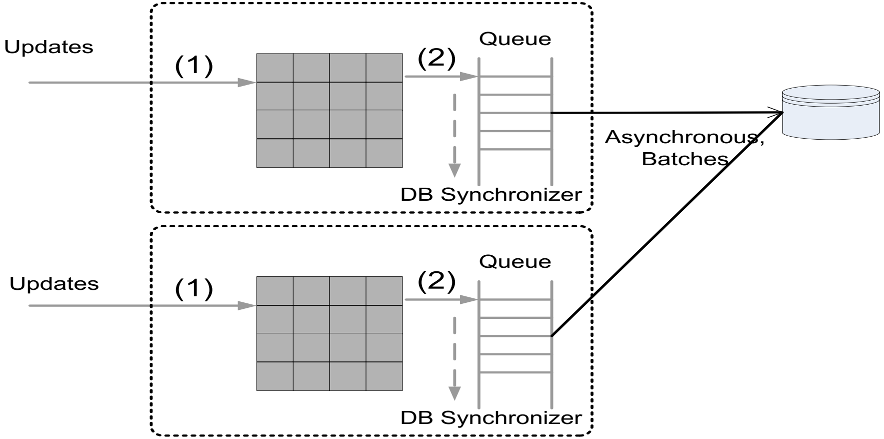




Asynchronous 'write-behind' Cache

Traditional RDB integration can be challenging





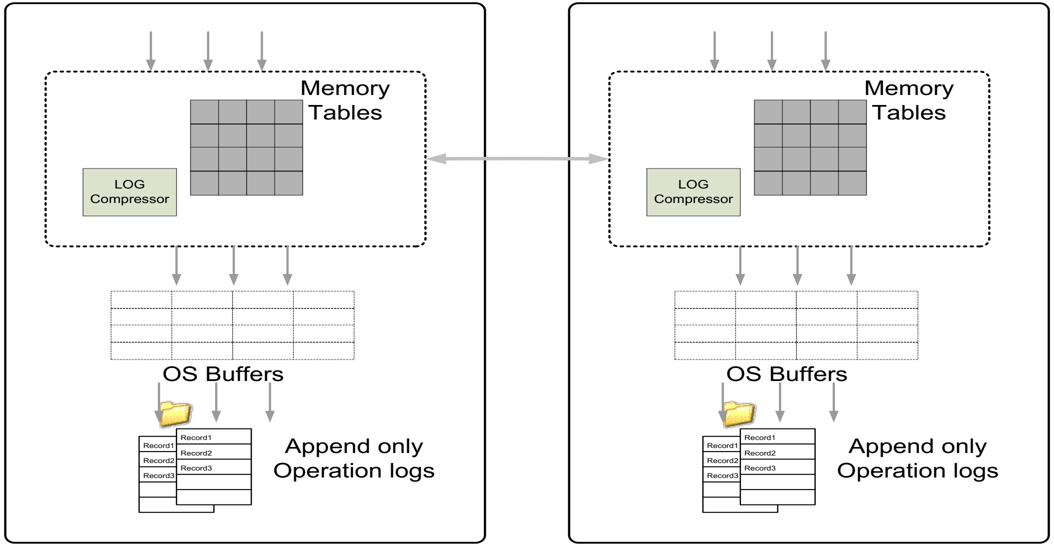
Synchronous “Write through”

Single point of bottleneck and failure
Not an option for “Write heavy”
Complex 2-phase commit protocol
Parallel recovery is difficult

Asynchronous “Write behind”

Cannot sustain high “write” rates
Queue may have to be persistent
Parallel recovery is difficult

Some IMDG, NoSQL offer 'Shared nothing persistence'



- Append only operation logs
- Fully parallel
- Zero disk seeks

- But, cluster restart requires log scan

- Very large volumes pose challenges

GemFire

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 steet 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

→ Hybrid Transactional /Analytics grids

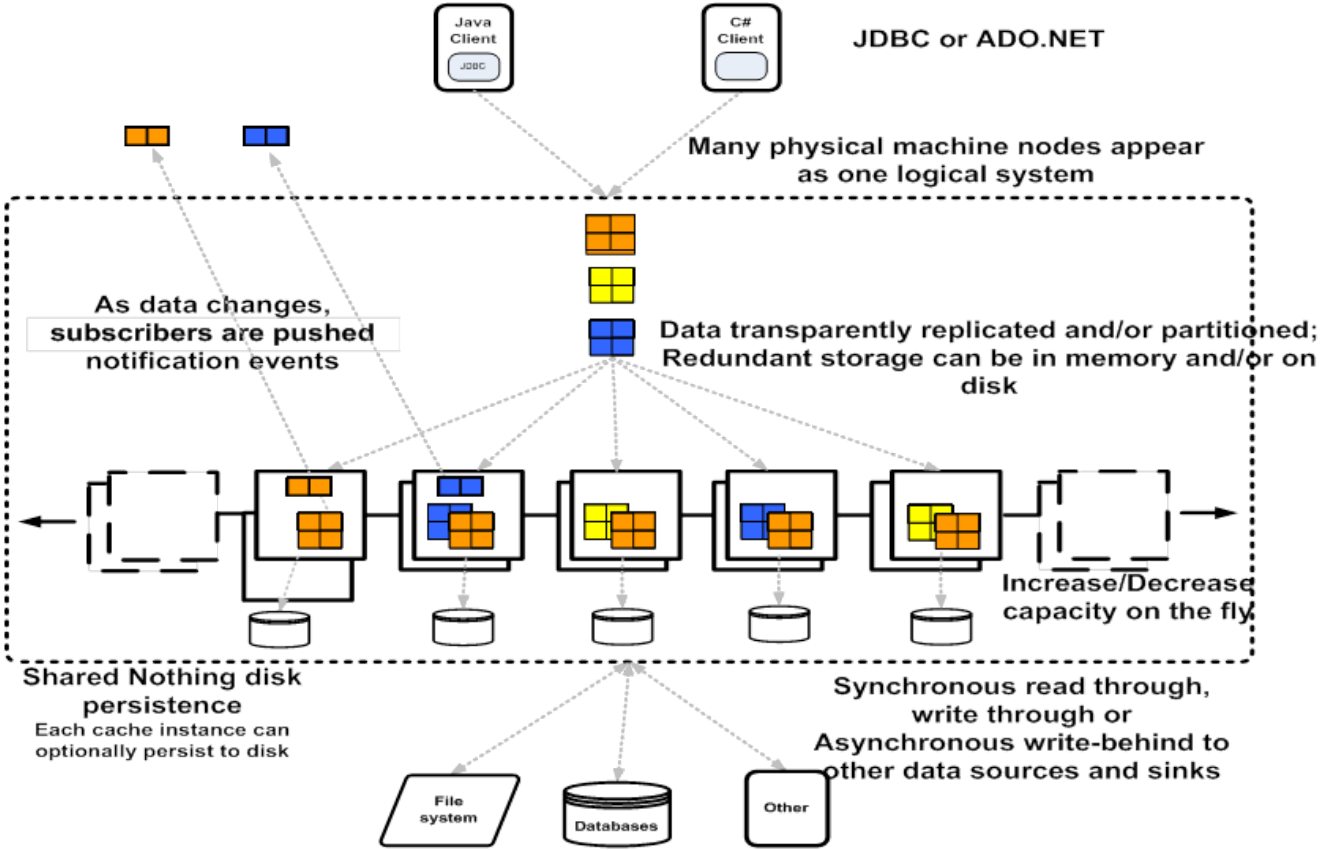
Why OSS? Why Apache?

- Open Source Software is fundamentally changing buying patterns
  - Developers have to endorse product selection (No longer CIO handshake)
  - Community endorsement is key to product visibility
  - Open source credentials attract the best developers
  - Vendor credibility directly tied to street credibility of product
- Align with the tides of history
  - Customers increasingly asking to participate in product development
  - Resume driven development forces customers to consider OSS products
  - Allow product development to happen with full transparency
- Apache is where you go to build Open Source street cred
  - Transparent, meritocracy which puts developers in charge

Geode Will Be A Significant Apache Project

- Over a 1000 person years invested into cutting edge R&D
- 1000+ 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

Gemfire High Level Architecture



JDBC or ADO.NET

Many physical machine nodes appear as one logical system

As data changes, subscribers are pushed notification events

Data transparently replicated and/or partitioned; Redundant storage can be in memory and/or on disk

Increase/Decrease capacity on the fly

Shared Nothing disk persistence

Each cache instance can optionally persist to disk

Synchronous read through, write through or Asynchronous write-behind to other data sources and sinks

What makes it fast?

- Minimize copying
  - Clients dynamically acquire partitioning meta data for single hop access
  - Avoid JVM memory pools to the extent possible
- Minimize contention points .. avoid offloading to OS scheduler
  - Highly concurrent data structures
  - Efficient data transmission – Nagle's Algorithm
- Flexible consistency model
  - FIFO consistency across replicas but NO global ordering across threads
  - Promote single row transactions (i.e no transactions)

Pivotal™

What makes it fast?

- Avoid disk seeks
  - Data kept in Memory – 100 times faster than disk
  - Keep indexes in memory, even when data is on disk
  - Direct pointers to disk location when offloaded
- Tiered Caching
  - Eventually consistent client caches
  - Avoid Slow receiver problems
- Partition and parallelize everything
  - Data. Application processing (procedures, callbacks), queries, Write behind, CQ/Event processing

“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"
```

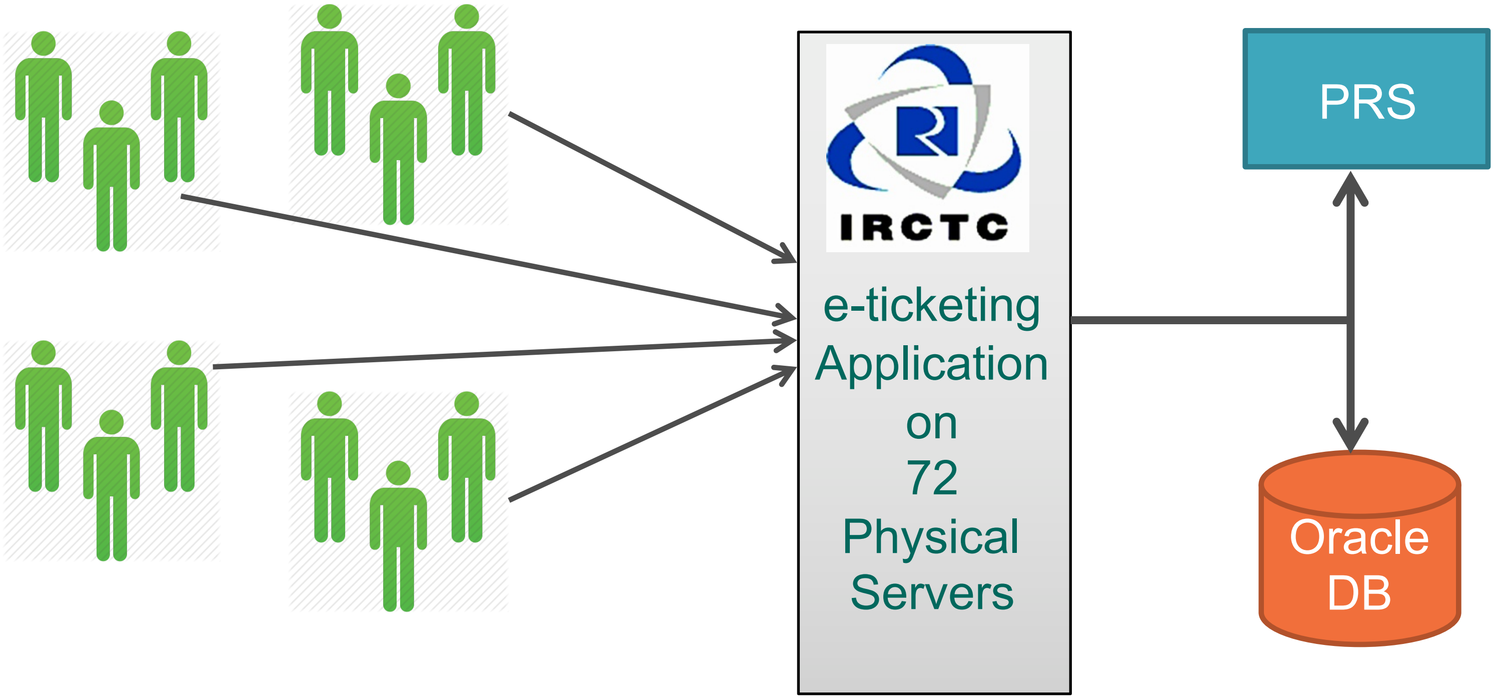
Pivotal™

A GemFire customer use case : IRCTC

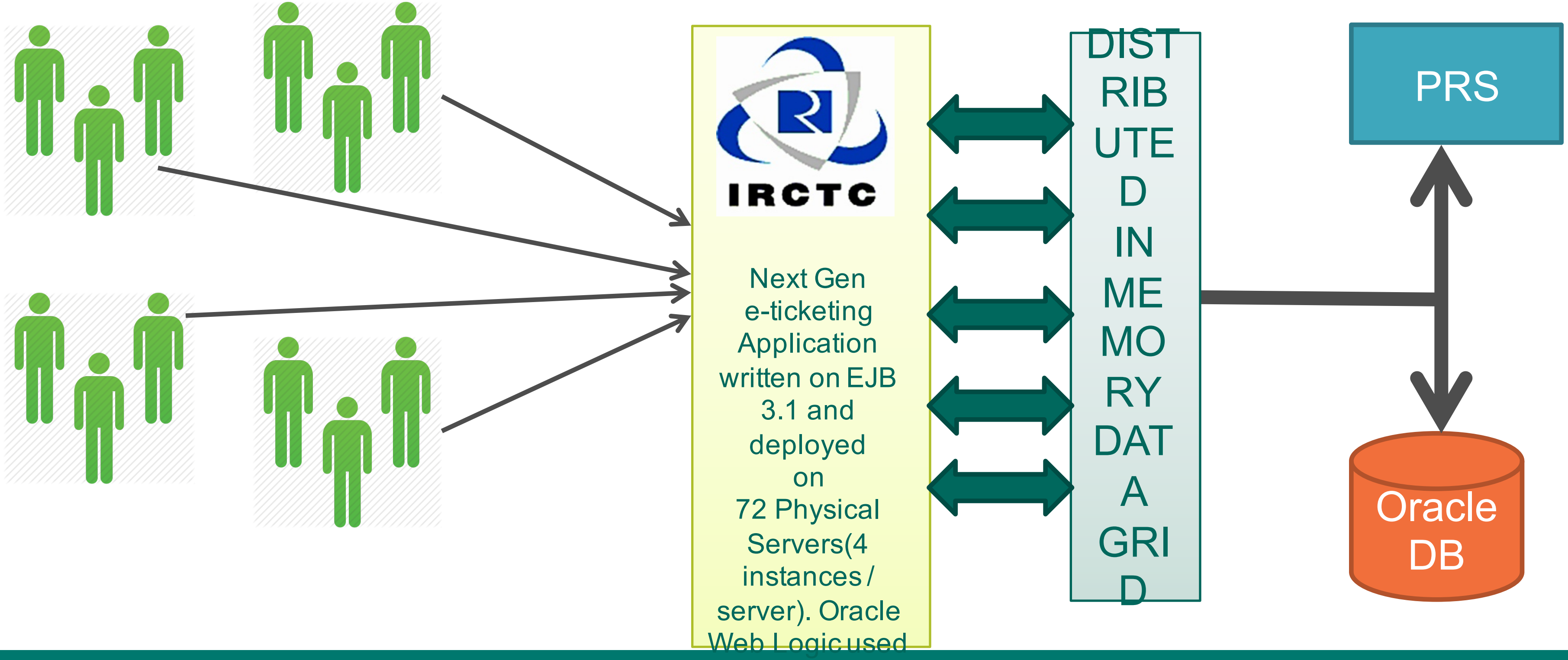
- World’s second largest railway network, 7000 stations, 30 million users,12000 trains
- Longer queues at railway booking counters
- Not able to scale during peak hours, 8AM, 10AM
- System designed back in 2005/2006
- Frequent downtimes, more than 10 mins delay to book a ticket, or timeout.

Pivotal™

Old Architecture



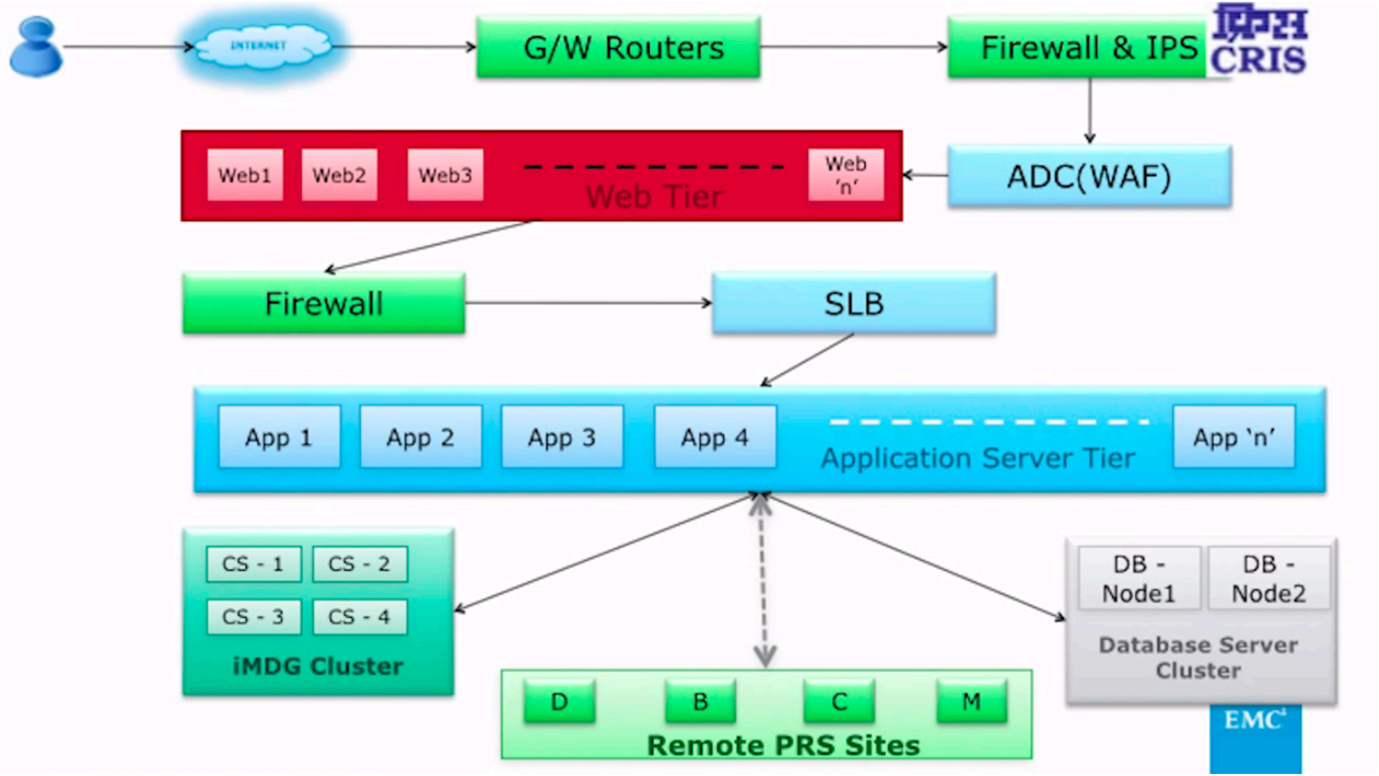
Architecture Using GemFire


IRCTC
Next Gen e-ticketing Application written on EJB 3.1 and deployed on 72 Physical Servers(4 instances / server). Oracle Web Logic used
DIST RIB UTE D IN ME MO RY DAT A GRI D
PRS
Oracle DB

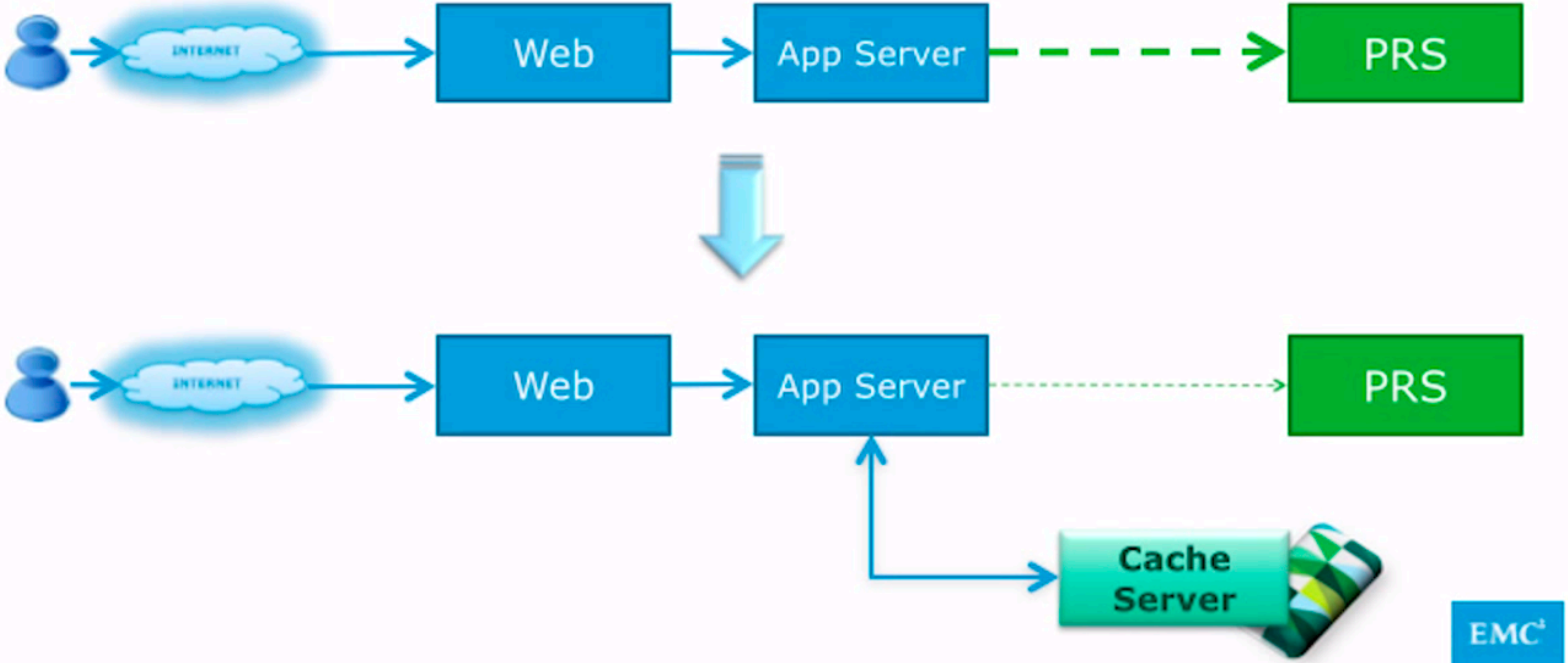
Challenges

- Social infrastructure site
- Migrating 30 million registered users
- Booking transaction checkpoints because of supply demand gaps
- Journey Planner, user authentication migration to in memory
- Capable of scaling up as the demand increases in future.
- High number of concurrent users at the peak times

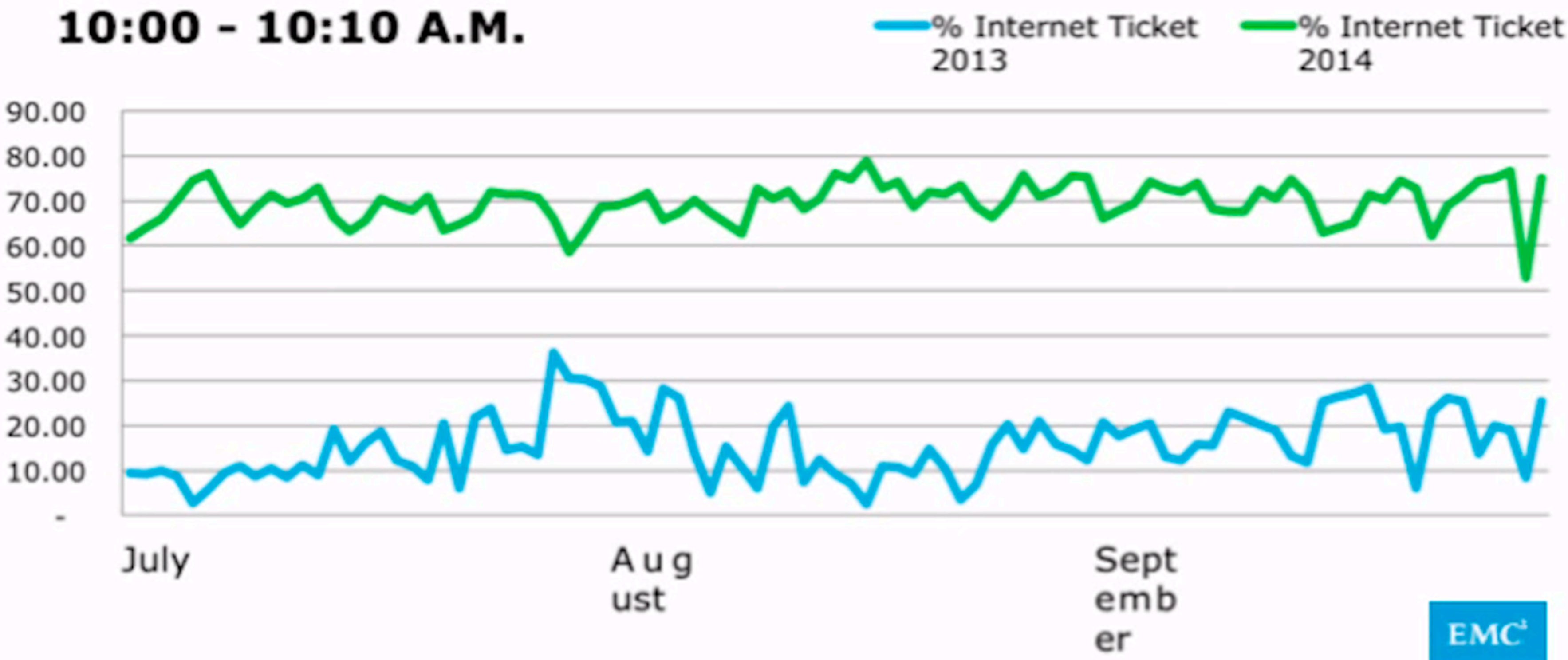
Architecture Using GemFire



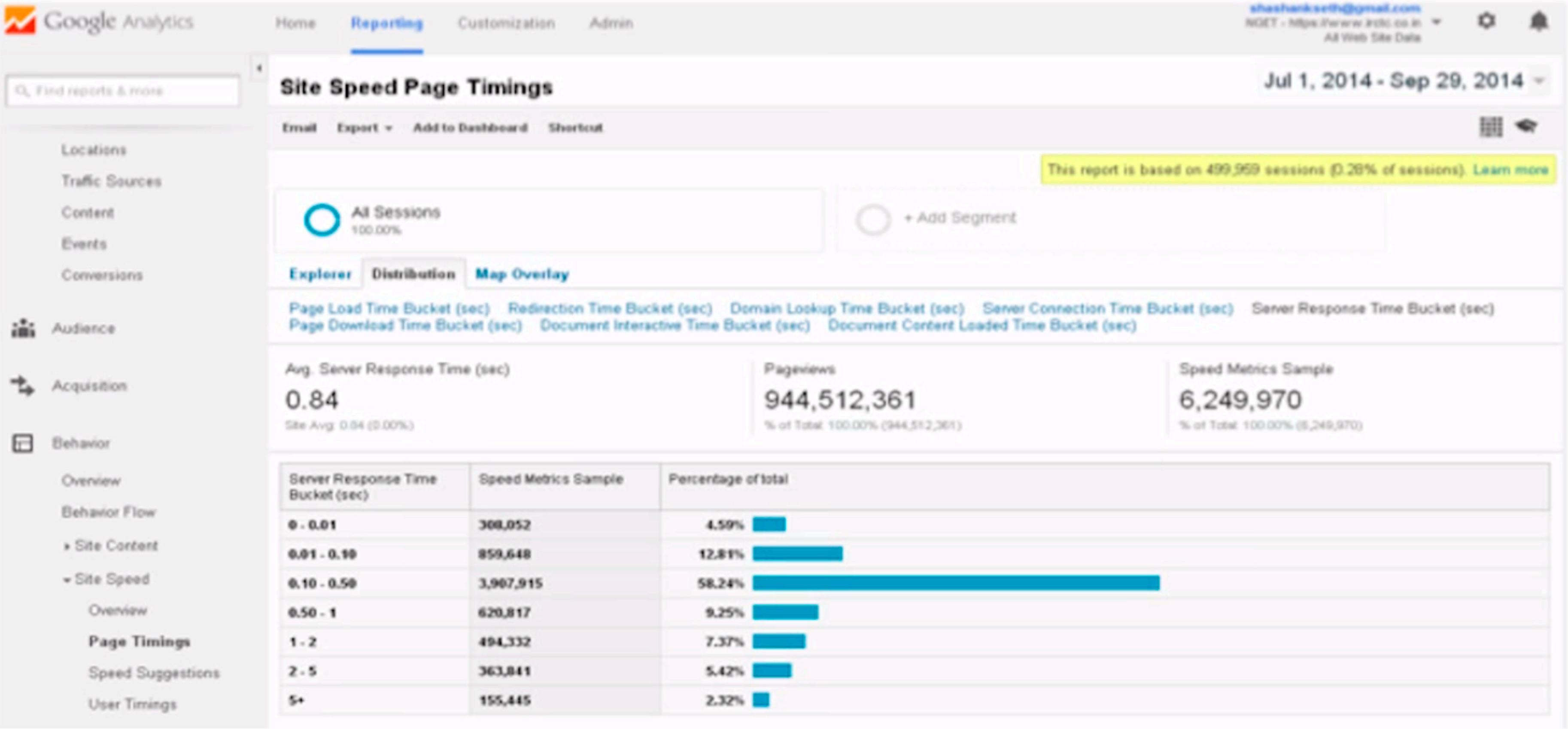
Journey Planning & Availability Enquiry



Increase in e-ticket booking In initial 10 minutes



Average response time < 1 sec



Site Speed Page Timings

Jul 1, 2014 - Sep 29, 2014

This report is based on 499,959 sessions (0.28% of sessions). Learn more

Avg. Server Response Time (sec)
0.84
Site Avg: 0.84 (0.00%)

Pageviews
944,512,361
% of Total: 100.00% (944,512,361)

Speed Metrics Sample
6,249,970
% of Total: 100.00% (6,249,970)

Server Response Time Bucket (sec)	Speed Metrics Sample	Percentage of total
0 - 0.01	308,052	4.59%
0.01 - 0.10	859,648	12.81%
0.10 - 0.50	3,907,915	58.24%
0.50 - 1	620,817	9.25%
1 - 2	494,332	7.37%
2 - 5	363,841	5.42%
5+	155,445	2.32%

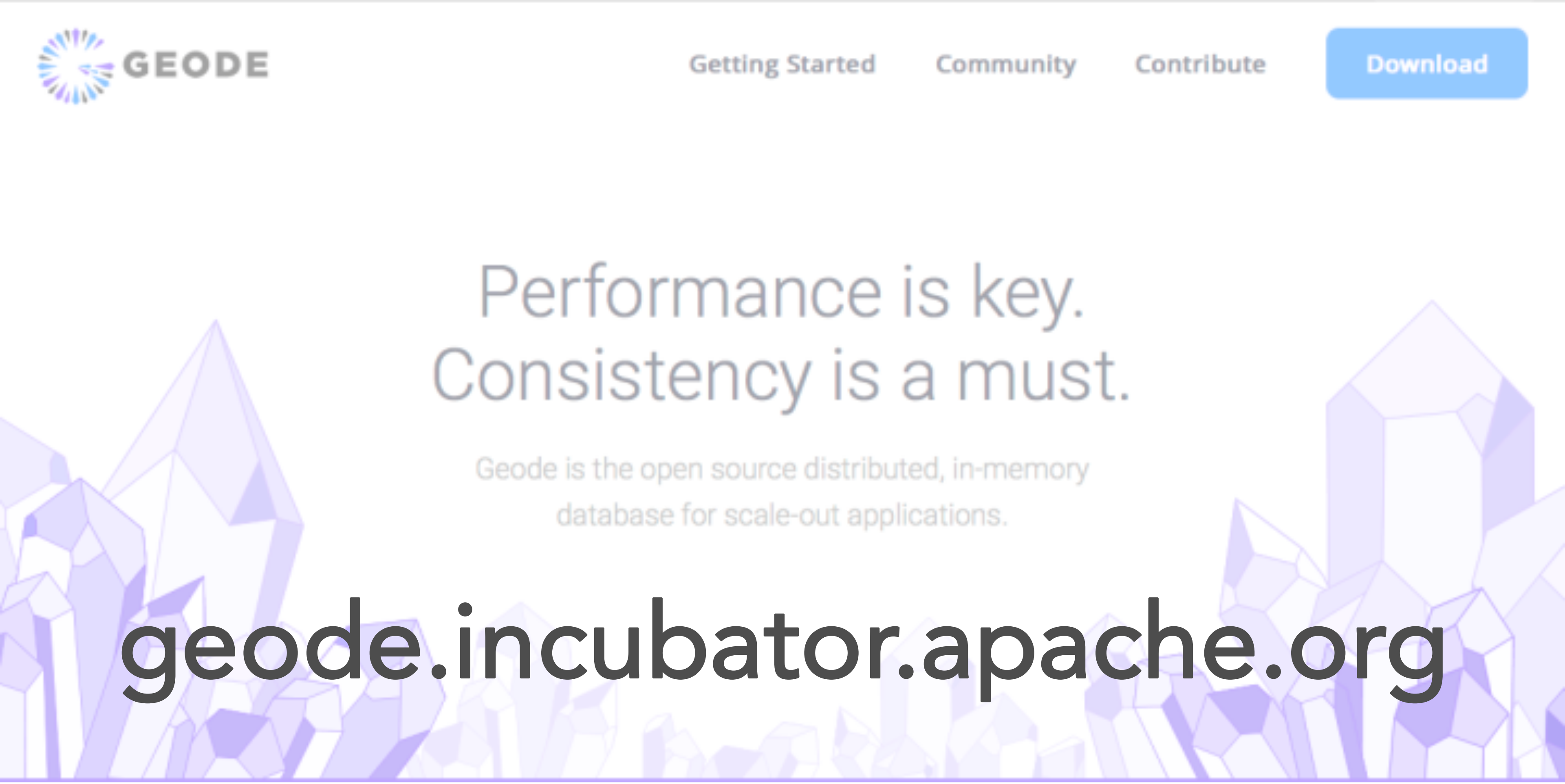
Benefits

- Supports More than 200,000 Concurrent Purchases
- Provide Stable Performance to Book Approximately 150,000 TPH, Compared to 60,000 in the Old System
- Transformed Customer Experience so Reservation Transactions Complete in Seconds Instead of 15 minutes
- Shifted Online Purchasing From 50% of Tickets Sold to 65%
- Boosting Revenue Generated From E-ticket Sales to INR600 Million Daily
- Capable of scaling up as the demand increases in future.
- CPU Usage during peak hours (**Tatkaal)** is less than 9%

Roadmap

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

…and other ideas from the Geode community!



Performance is key.
Consistency is a must.

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

geode.incubator.apache.org

Geode community

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

Our in-memory computing journey

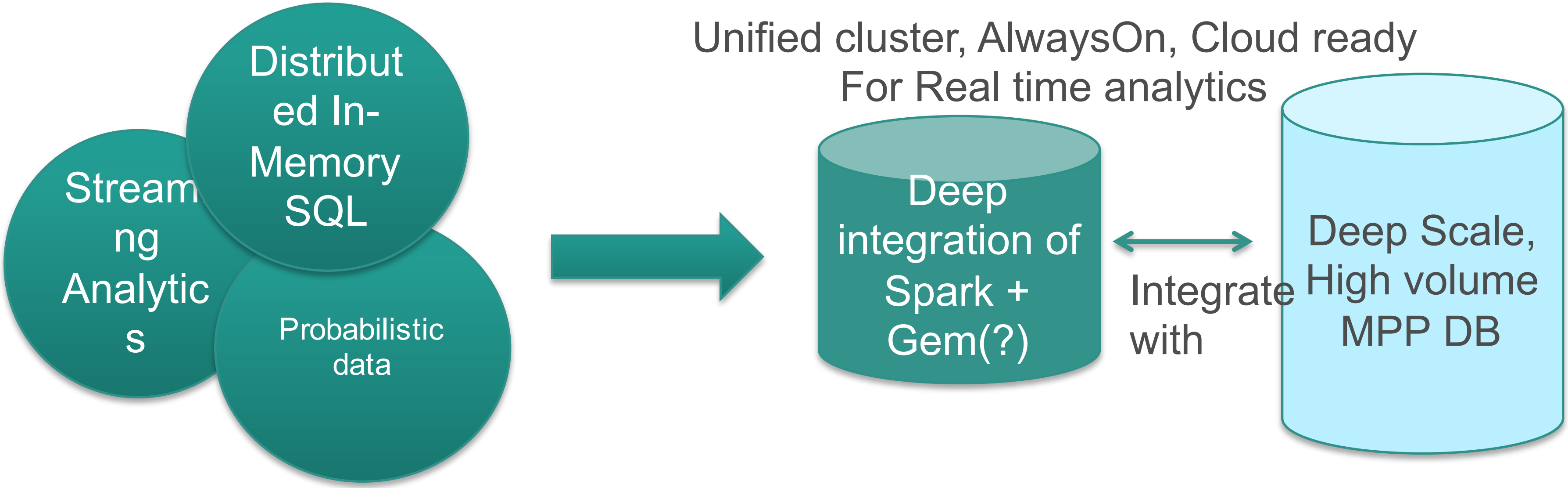
- We started GemFire team in Pune in 2005, the core team remains the same over the last decade

- We build a new product out of Pune , GemFire XD, In memory distributed SQL with GemFire and Apache Derby.

- We are now working on a new initiative, SnappyData.io, a startup funded by Pivotal, building a product based on Spark(Streaming/SQL), GemFire and Approximate Query Engine.

- And we are hiring

SnappyData Positioning (snappydata.io)



Unified cluster, AlwaysOn, Cloud ready
For Real time analytics

Distributed In-Memory SQL

Streaming Analytics

Probabilistic data

Deep integration of Spark + Gem(?)

Integrate with

Deep Scale, High volume MPP DB

Vision – Drastically reduce the cost and complexity in modern big data. …Using fraction of the resources

10X better response time, drop resource cost 10X, reduce complexity 10X