WRITE ONCE.
SCALE ANYWHERE.

NoSQL, One size fits all: A concept whose time has come and gone

Nati Shalom,
CTO & Founder GigaSpaces
John Davies Incept5
Frank Greco Kaazing, NYJavaSig

.

About GigaSpaces Technologies

Enabling applications to run a distributed cluster as if it was a single machine…

75+ Cloud Customers | **Among Top 50 Cloud Vendors** | **300+ Direct Customers**

Bild 2

a2

1. Replaced the number of deployments (+2000) with the number of Cloud customers.
- Details about the specifics of the numbers are in the notes below.

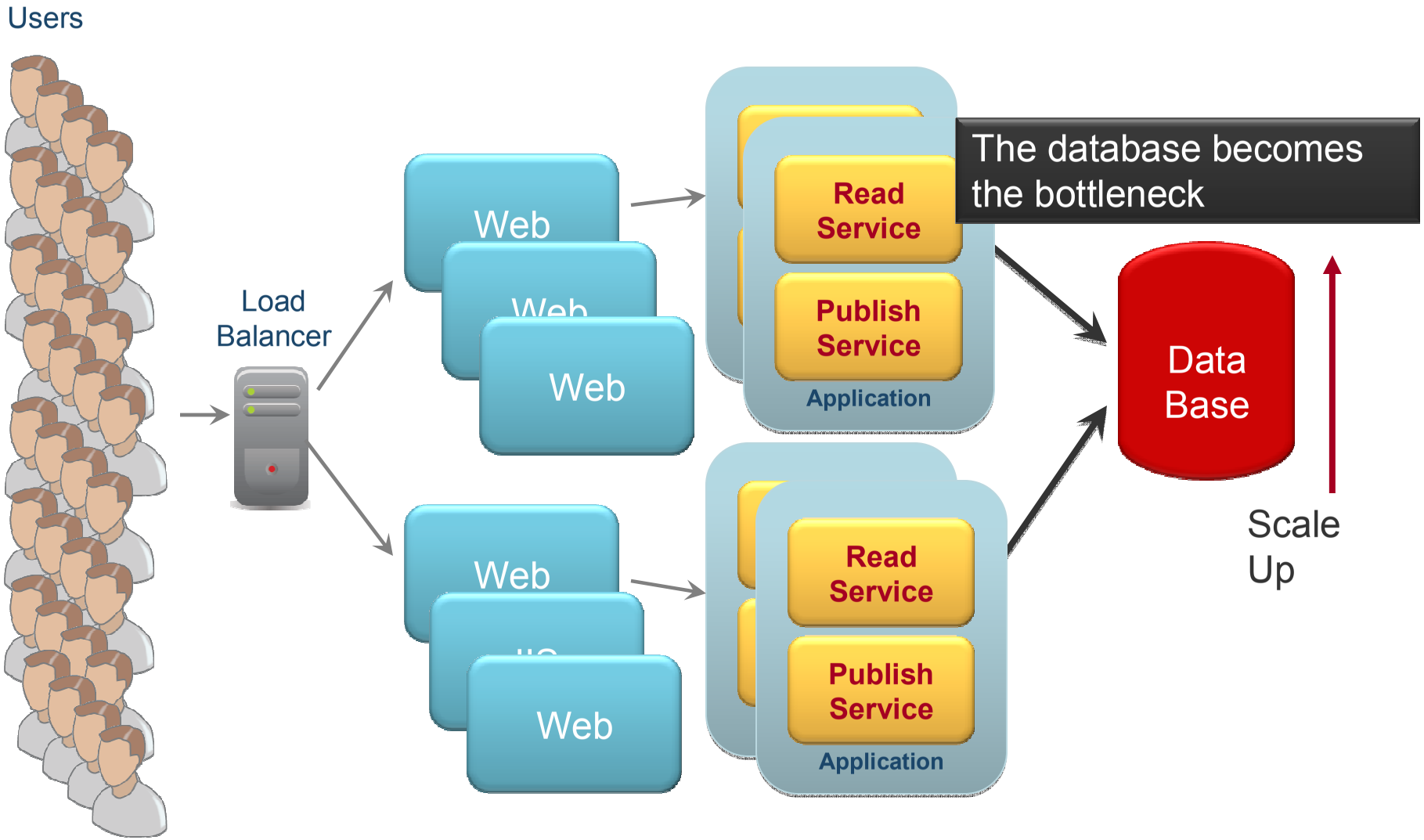
2. added logo's of Cloud Customers and partners (included Spring in there) at the bottom part of the slide

alit, 23/08/2009

Agenda

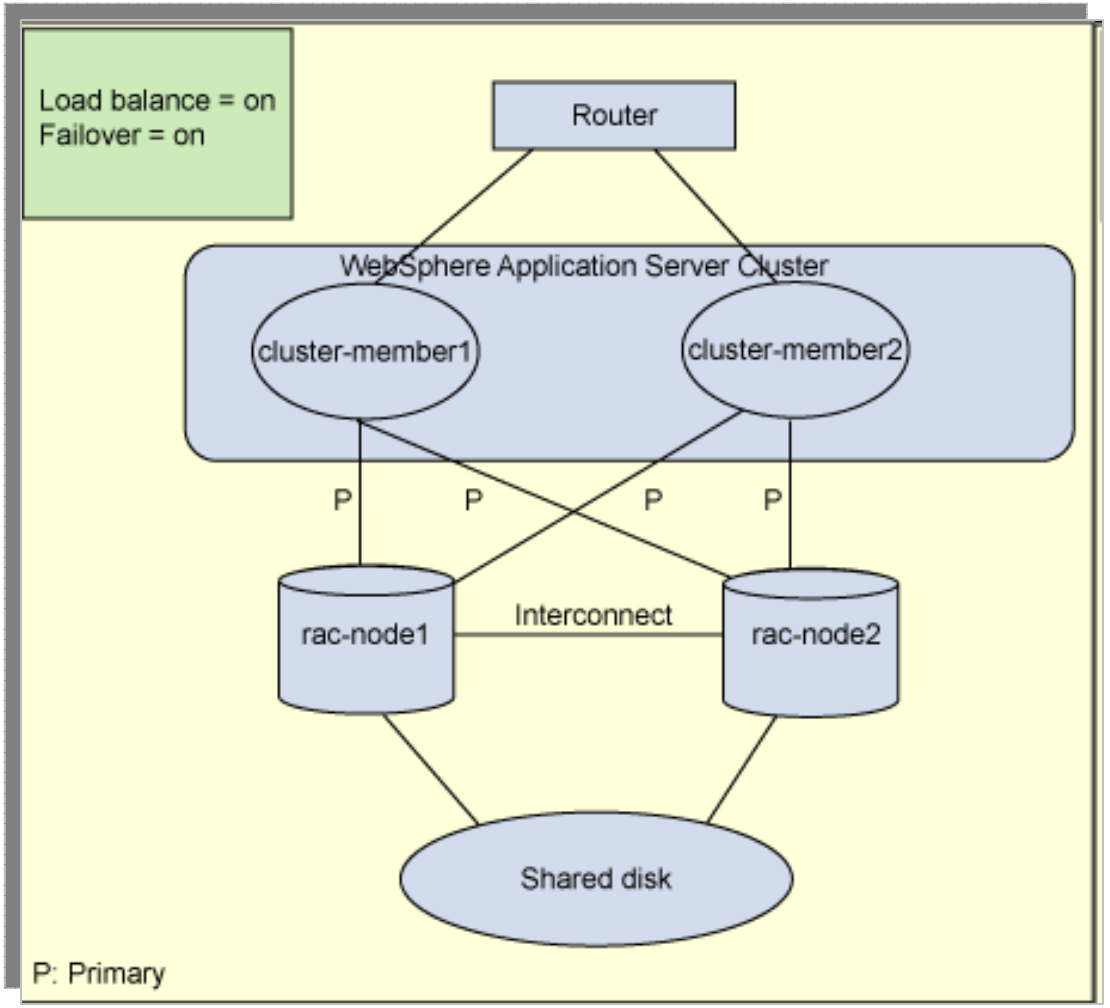
- Traditional Data Base Scaling approach
  - Typical database clusters
  - Other data scalability approaches
- Drive for a change
  - Social media and cloud computing effect
- Exercise: Writing your own scalable twitter
- NoSQL alternatives
  - Common principles
  - Query models
- The best of both world
  - JPA on top of Key/Value store

A typical scaling approach : share nothing



Traditional database scaling 1/2: Shared storage

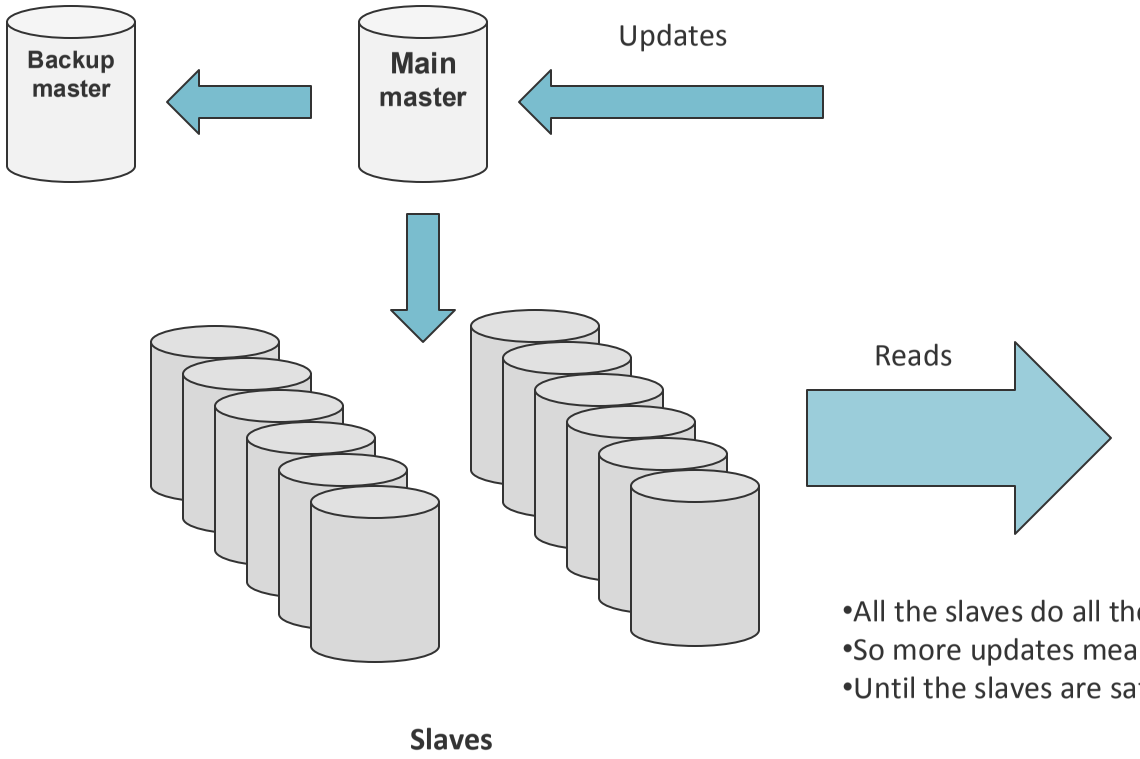
- Reliability at the cost of performance and scalability
- Fairly Expensive
  - Strong reliance on expensive hardware
- Doesn't scale well



Source: IBM Websphere

Traditional Database scaling 2/2 – Master/Slave

- MySQL replication is quite powerful and flexible
- But the model does not scale well



- All the slaves do all the updates
- So more updates means more slaves, even at the same read traffic
- Until the slaves are saturated with just updating!

Learning from the Vasa ship failure

Why the Vasa Sank ?

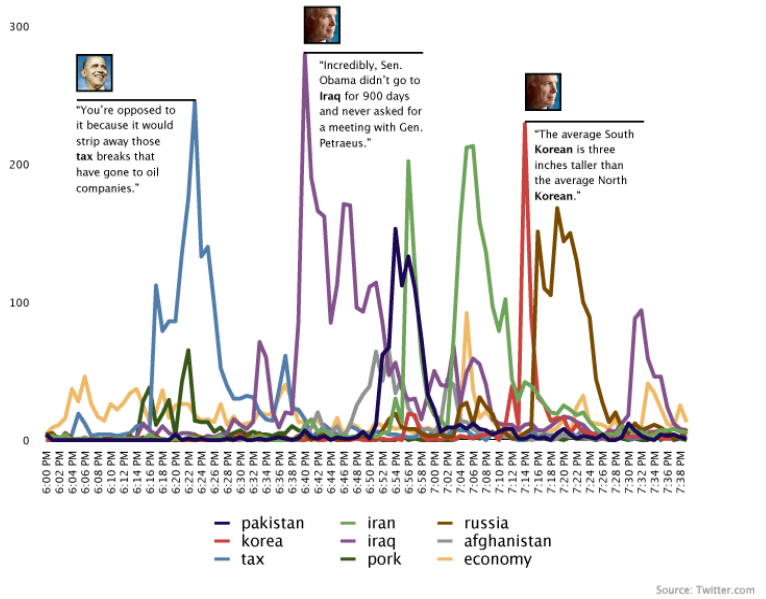
The Vasa was designed to be one of the premier warships of the 17th century. Unfortunately, the ship sank two hours after its initial launch in an 8-knot wind

Agenda

- Traditional Data Base Scaling approach
  - Typical database clusters
  - Other data scalability approaches
- Drive for a change
  - Social media and cloud computing effect
- Exercise: Writing your own scalable twitter
- NoSQL alternatives
  - Common principles
  - Query models
- The best of both world
  - JPA on top of Key/Value store

The need for a change

- Social network changes the web experience:
  - Read mostly => Read/write
  - Predictable traffic => Viral traffic
- SaaS
  - Application PER customer => Application for ALL customers
- Economy meltdown
  - Throwing more expensive hardware at the problem doesn't work anymore



- Core assumptions behind the current database design are proven to be broken (See next slides)

Our data is safe once it is saved on disk?

- Disk failure/year - 3% vs. 0.5 - 0.9% estimated 600% more then estimated
- No correlation between failure rate and disk type - SCSI, SATA, or fiber channel
- Lower temperatures are associated with higher failure rates

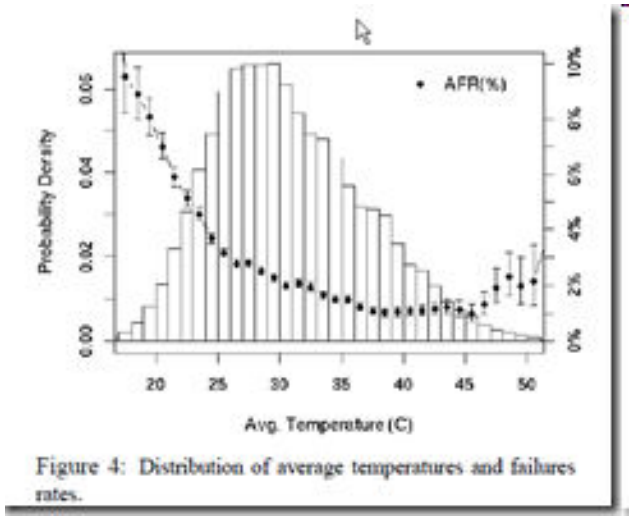


Figure 4: Distribution of average temperatures and failures rates.

Memory is more expensive and not reliable?

- Memory can be x100, x1000 more efficient than disk

> *"RAMClouds become much more attractive for applications with high throughput requirements. When measured in terms of cost per operation or energy per operation,* ***RAMClouds are 100-1000x more efficient than disk-based systems*** *and 5-10x more efficient than systems based on flash memory (Source: Stanford University research – The Case For RAM Cloud)"C*

- New hardware makes it possible to store the entire set of data in memory

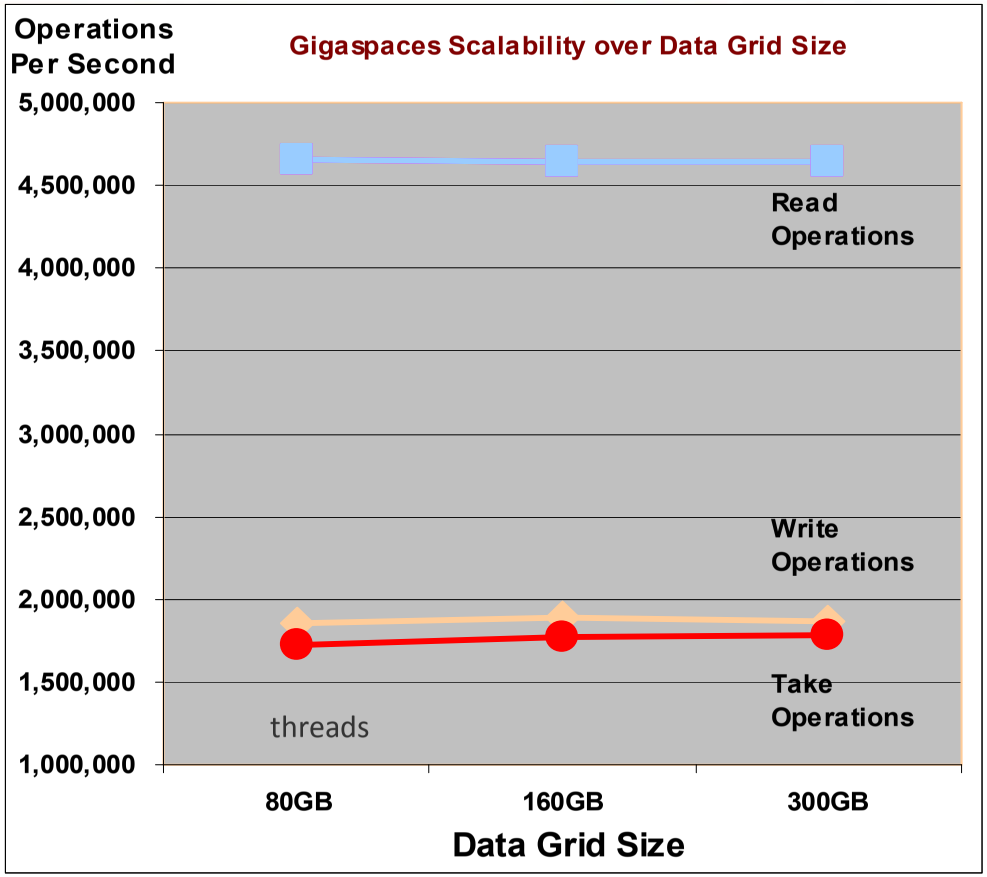
Online Retailer		Airline Reservations	
Revenues/year:	$16B	Flights/day:	4000
Average order size	$40	Passengers/flight:	150
Orders/year	400M	Passenger-flights/year:	220M
Data/order	1000 - 10000 bytes	Data/passenger-flight:	1000 - 10000 bytes
Order data/year:	400GB - 4.0TB	Passenger data/year:	220GB - 2.2 TB
RAMCloud cost:	$24K-240K	RAMCloud cost:	$13K-130K

Table 3. Estimates of the total storage capacity needed for one year's customer data of a hypothetical online retailer and a hypothetical airline. In each case the total requirements are no more than a few terabytes, which would fit in a modest-sized RAMCloud. The last line estimates the purchase cost for RAMCloud servers, using the data from Table 1.

Available memory capacity grows exponentially

- Current (ops/sec)
  - 1.8M read
  - 1.1M write
- UCS (ops/sec)
  - 4.5M read
  - 2M write

- 100% improvements on throughput
- A Single JVM can hold up to 270GB data without performance drop. (A typical deployment today is 5G-10G per VM)
- => Save HW cost by a factor of x20 (compared with typical deployment today) => huge saving in operational and admin cost.



Cisco Extended Memory Technology

Agenda

- Traditional Data Base Scaling approach
  - Typical database clusters
  - Other data scalability approaches
- Drive for a change
  - Social media and cloud computing effect
- Exercise: Writing your own scalable twitter
- NoSQL alternatives
  - Common principles
  - Query models
- The best of both world
  - JPA on top of Key/Value store

Exercise: Writing your own scalable twitter

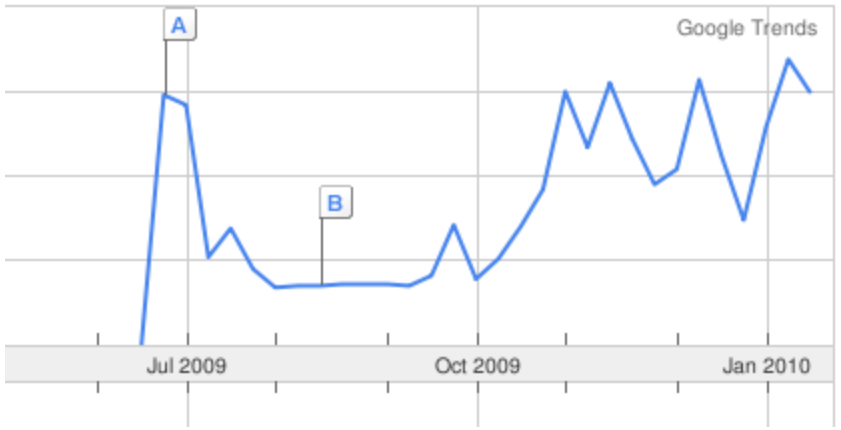
- Design a scalable twitter application
  - Performance requirements:
    - 150 req/hour per user
  - Scalability requirements
    - 1M users

Agenda

- Traditional Data Base Scaling approach
  - Typical database clusters
  - Other data scalability approaches
- Drive for a change
  - Social media and cloud computing effect
- Exercise: Writing your own scalable twitter
- NoSQL alternatives
  - Common principles
  - Query models
- The best of both world
  - JPA on top of Key/Value store

The NOSQL movement

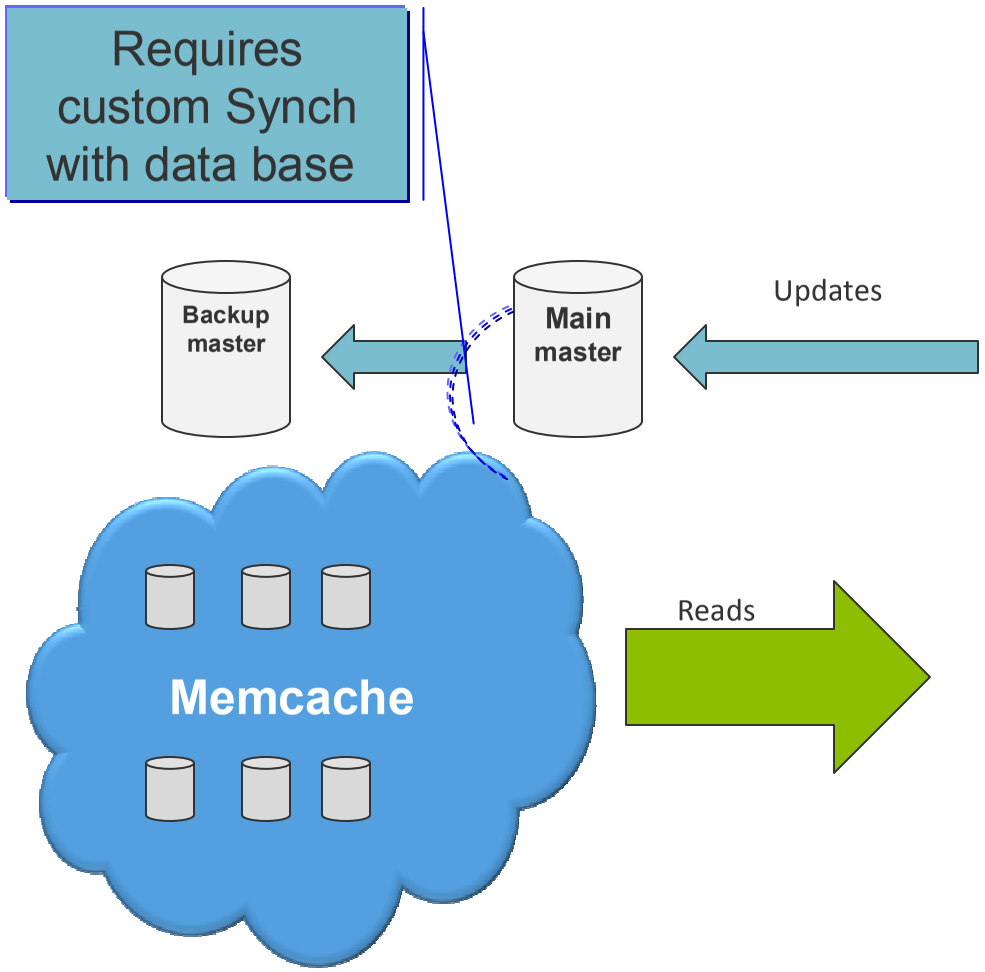
- Distributed key/value store (#NOSQL) –
  - New hot trend, influenced by Google, Amazon, facebook,..
  - Use Distributed commodity HW then expensive HW
  - Designed for massive scale
  - Examples
    - FIlesystem based implementation
      - Amazon - Dynamo/SimpleDB
      - Google- Big Table
      - Facebok –Casandra
      - LinkedIn - Voldemort
    - Memory based implementation
      - GigaSpaces - XAP
      - Gemstone - Gemfire
      - IBM - extremeScale
      - JBoss - infinispan
      - Oracle - Oracle Coherence
      - Terracotta - EHcache



What about Memcached?

- Key value store
- Extremely simple
- Extremely scalable
- Language independent

- Missing
  - Reliability solution
  - Consistency model (read/write)
  - Limited query support



Memcached relies on RDBMS for reliability and consistency: Doesn't fit with the NoSQL core principles

Common principle behind NoSQL alternatives

- Design for failure
- Scale through partitioning of the data
- Maintain reliability through replication
- Provide flexibility between Consistency, Availability and Partition failure
- Dynamic scaling

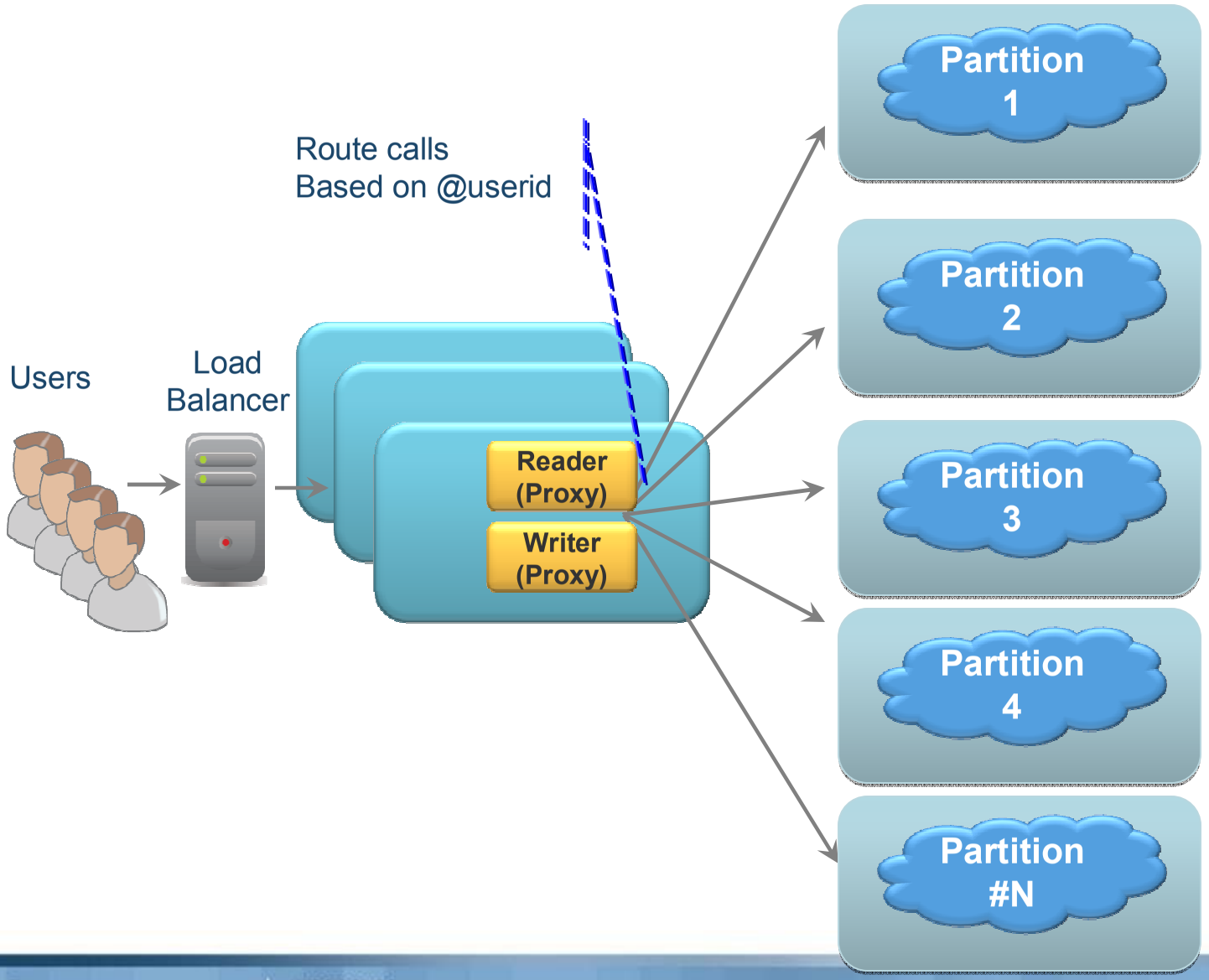
- Other principles not as common
  - Document model support
  - SQL Query support
  - Aggregated Query - Map reduce support
  - Transaction management
  - Security

Design for failure

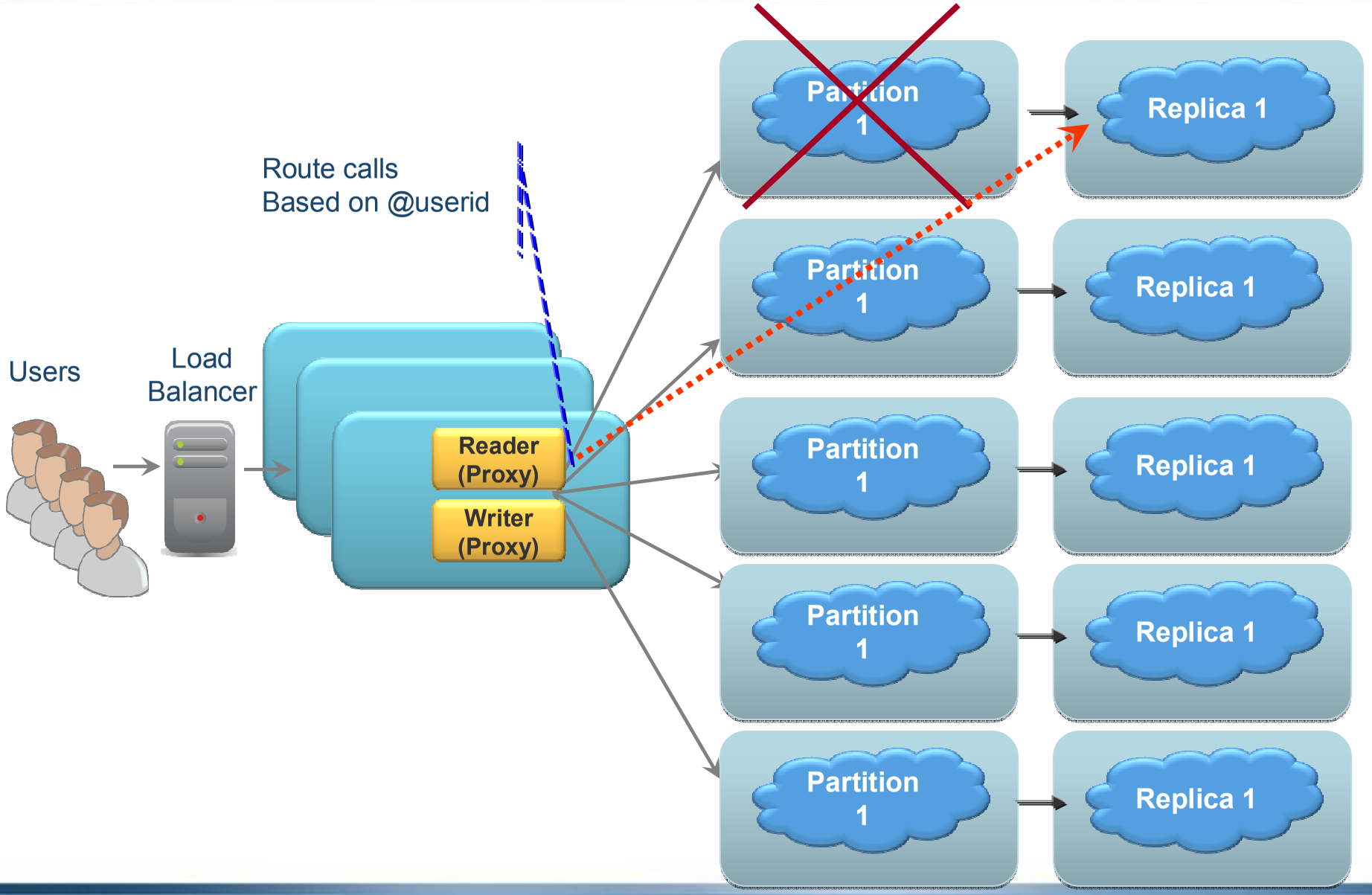
Design Principles Recap

- Expect and tolerate failures
- Code for large scale failures
- Expect and handle data and message corruption
- Code for elasticity
- Monitor, extrapolate and react
- Code for frequent single machine failures
- Game days

Scale through partitioning of the data



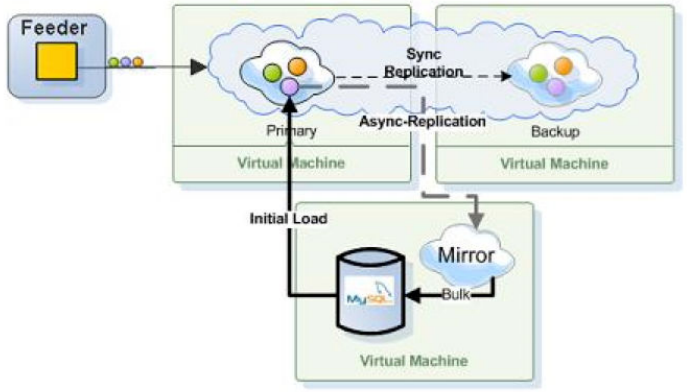
Maintain reliability through replication



Flexibility between Consistency, Availability and Partition failure

- Maintain configurable number of replicas
- Eventual consistency (Asynchronous replication)
  - Better performance
  - Weaker Consistency: read/write may not be consistent
  - Reliability – may loose data
- Strong consistency - (Synchronous replication)
  - Performance could be lower by 50% (Two network calls)
  - Guaranteed consistency and availability

- Combining the two: Write behind
  - Synchronous to memory
  - Asynchronous to disk



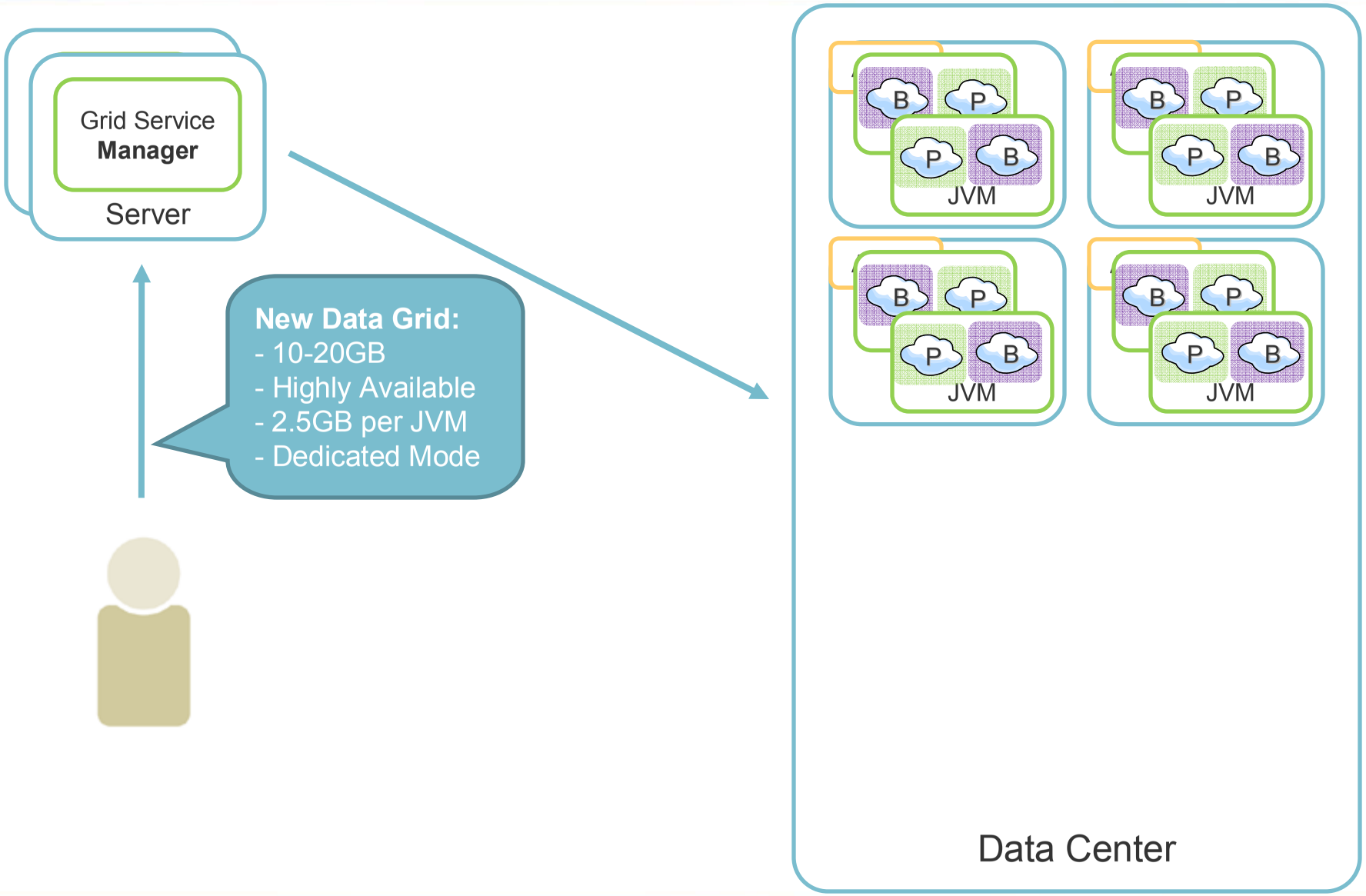
Choosing the right consistency/availability

- Design a scalable twitter application
  - Performance requirements:
    - 150 req/hour per user
  - Scalability requirements
    - 1M users
- Analysis
  - For 1M users that means 40,000 req/sec
  - The actual data that needs to be accessed in real-time is only the window of time between interactions (Last 10/30 min should be more then sufficient)
  - The rest of the data can be stored in file-system storage (For users who just logged in)
  - Assuming a ratio of 20% writes and the size per post is 128 bytes:
    - Data produced = 40k*20% * 128 = 1Mb/Sec
    - We can keep a buffer of an hour in less then 5G
- Solution:
  - Use In Memory Data Grid for the real-time access and files-ystem storage for long term, search and other analytics requirements

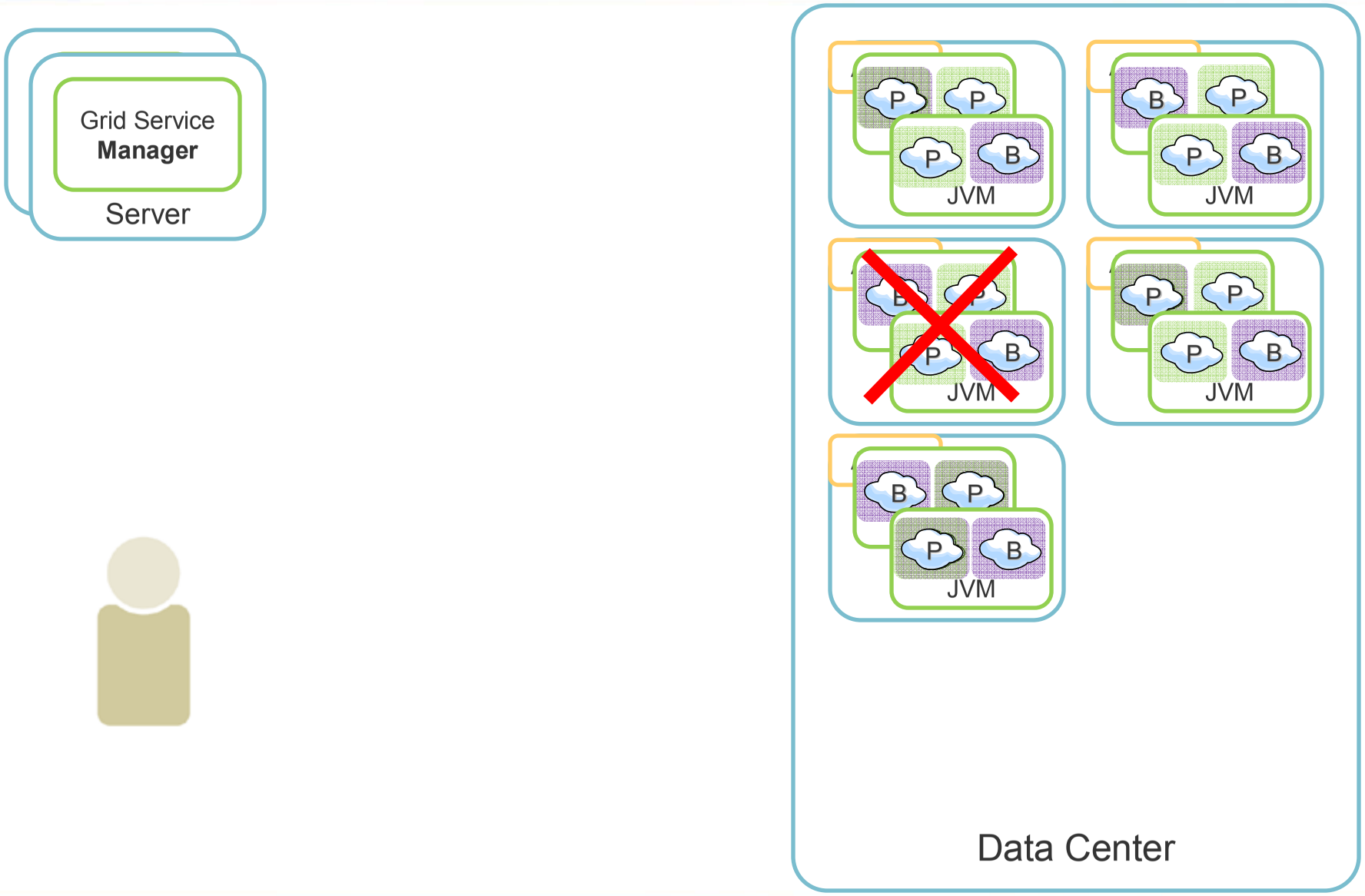
Dynamic Data Scaling

- Two common algorithms
  - Logical partitions
    - Mostly used by In Memory Data Grids
    - The Number of partitions is fixed
    - Number of partition per physical node vary based on scale
    - No rehashing
    - Extremely deterministic during failure and scaling event
    - Fairly simple algorithm
  - Consistent hashing
    - Used by Cassandra, Valdimore,..
    - Number of partitions can vary
    - Requires re-hashing
    - Failure impact can be minimized but are not deterministic
    - Fairly complex

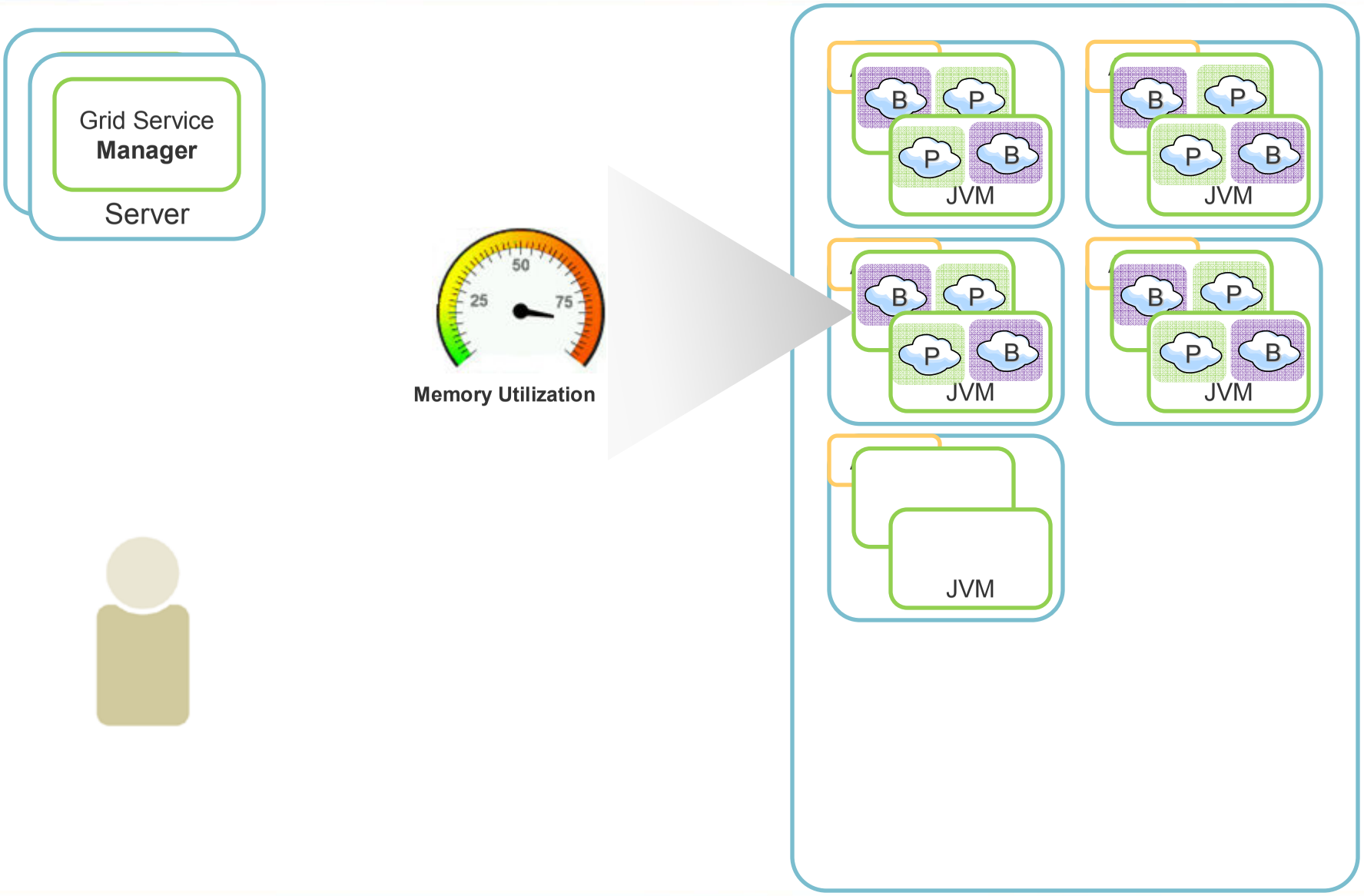
Dynamic Data Scaling (Logical partitions)



Dynamic Data Scaling (Logical partitions) – Failover



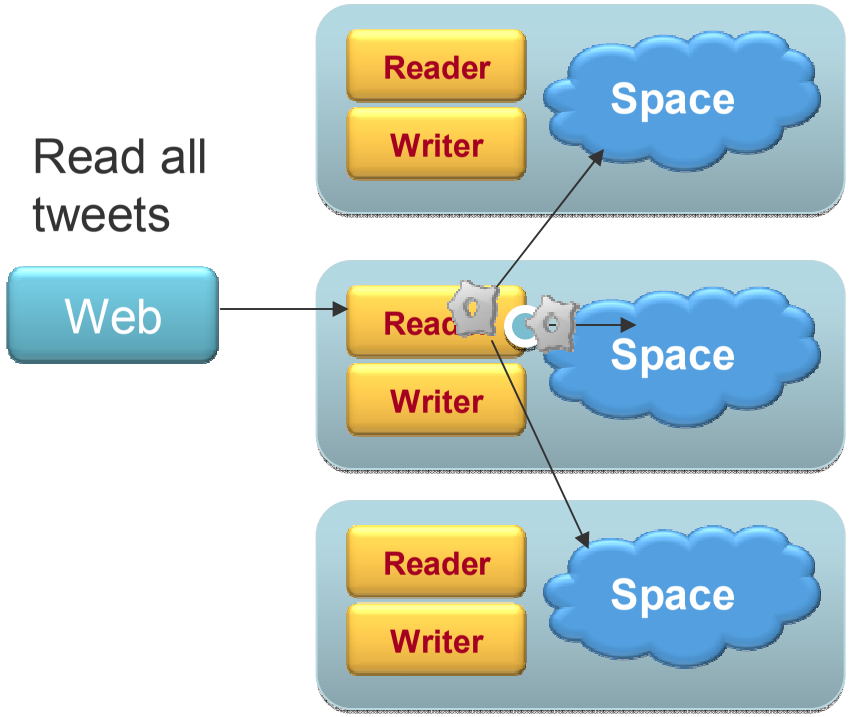
Dynamic Data Scaling (Logical partitions) – Automatic Scale Out



Query support

- Key/Value
  - Same as in hashTable
  - Least common denominator
- Document model
  - Search engine style
  - Blob model with varying number of indexes per raw
- SQL Like
  - Use portion of SQL sytax in combination
  - Similar indexing model as database but in-memory and distributed
- Aggregation - Map/Reduce
  - Included in some of the IMDG (GigaSpaces, Coherence)
  - Supported partially through Hadoop (Cassandra,..)

Map/Reduce support (GigaSpaces)



Read- Map/Reduce

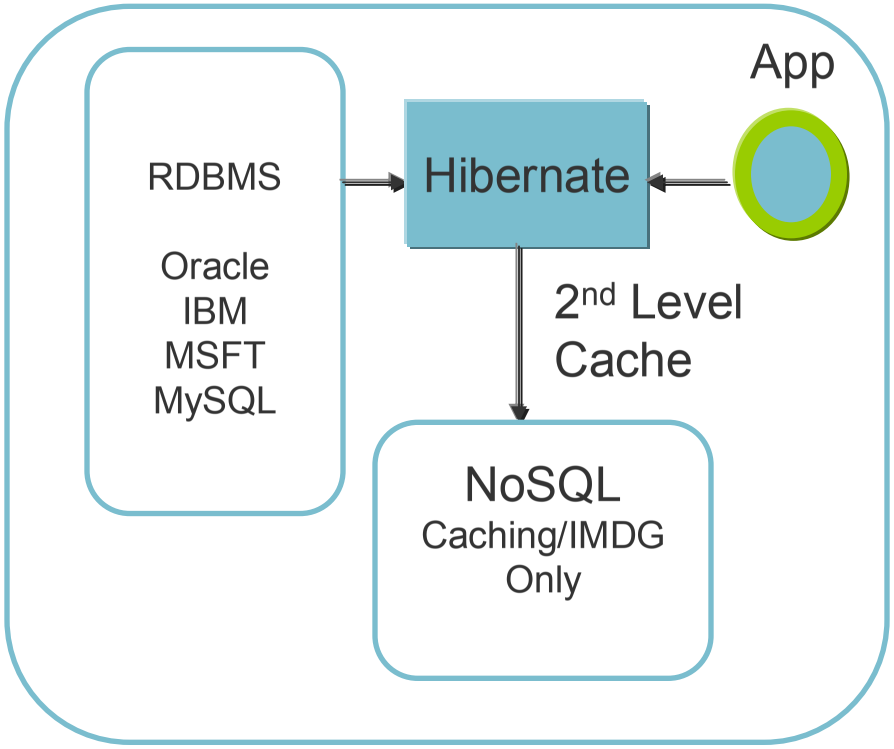
Implicit Map/Reduce: Tweet[] t = GigaSpaces.readMultiple(new Tweet(UerId)); Or
Explicit Map/Reduce Tweets t = Gigaspace.execute(new GetTweetsTask(userId));

Agenda

- Traditional Data Base Scaling approach
  - Typical database clusters
  - Other data scalability approaches
- Drive for a change
  - Social media and cloud computing effect
- Exercise: Writing your own scalable twitter
- NoSQL alternatives
  - Common principles
  - Query models
- The best of both world
  - JPA on top of Key/Value store

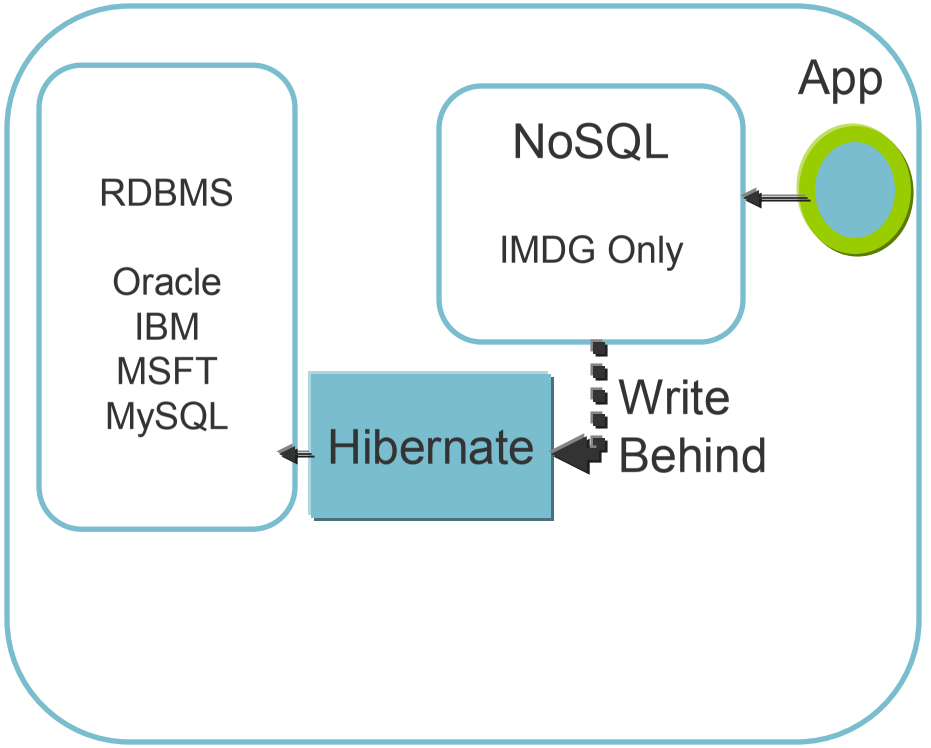
The best of both worlds: Migration strategies

2nd Level Cache (Read mostly)



- Seamless
- Limited value (Read mostly)
- Limited scalability

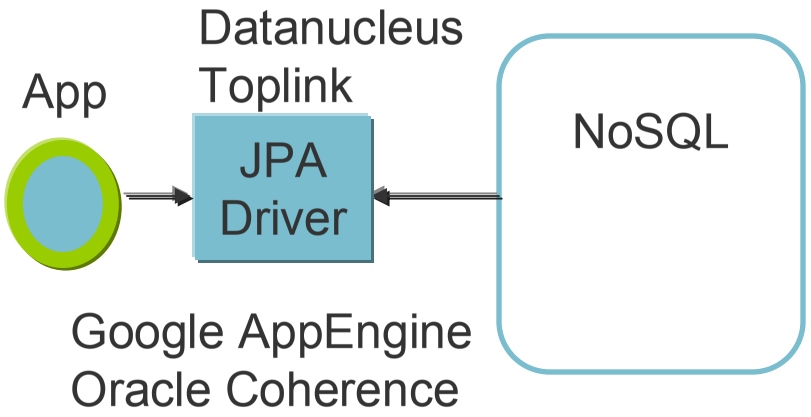
Write Behind (Read/Write)



- Data base remains the same
- Application need to be modifed
- Read/Write scalability

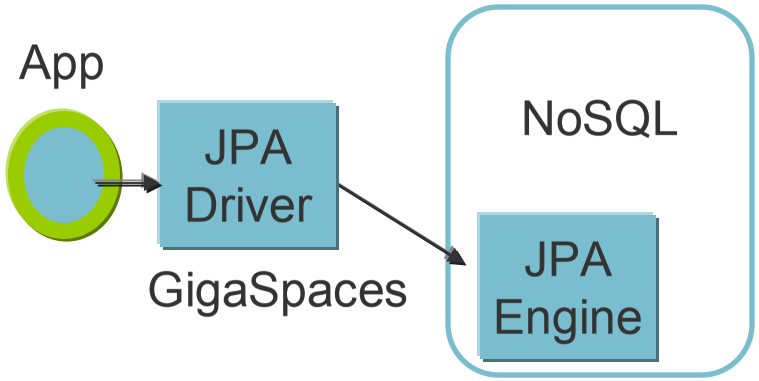
The best of both worlds: Mapping Strategies

External Mapping layer



- Standard API – Avoid lock-in
- Migrating from existing centralized RDBMS would require a change
- Performance overhead
- Lots of limitation especially around queries/transactions

Native Mapping layer



- Standard API – Avoid lock-in
- Migrating from existing centralized RDBMS would require a change
- No or little perforamcne overhead
- Minimal set of limitations

Summary

- RDBMS are not going a way anytime soon
- The concept of one size fit it all was and is wrong
- NoSQL represent a major paradigm shift
  - Scale after -> Scale First
- NoSQL alternative are hear to stay
  - Can argue with success: Google, Amazon, Facebook, Linkedin,..
  - Lots of DataGrid deployments running in mission critical apps
- Migration is expected to get simpler over time

Questions?