!SQL - Augmenting the RDBMS with a Distributed Key Value Store in the Real World

or

"Consistency, schmistency...."

Geir Magnusson Jr
V.P. Platform and Architecture
Gilt Groupe Inc.
geir@pobox.com

Agenda

- About Me
- The Talk in One slide
- Gilt : What We Do and Why We Needed !SQL
- Goal : Turning off the RDBMS at Peak
- Project Voldemort : What it is and why we chose it
- Summary

About Me

- VP, Platform and Architecture at Gilt Groupe
- Commercial developer for 20+ years
  - Bloomberg, Intel, IBM, Gluecode, Adeptra, Joost, 10gen
- Open source practitioner and advocate for 10 years
  - Apache Software Foundation
    - Member, Director, Officer
    - Apache Geronimo, Apache Harmony, Apache DB, Apache Velocity, Jakarta Commons, etc
  - Codehaus
  - Project Voldemort
- *Not* a database domain expert

The Talk in One Slide

Modern data-oriented apps are forcing us - programmers, architects, and C[I|T]Os - rethink our applications and data models.

Thankfully, databases are changing in response.

You should go investigate these new technologies.

(It turns out this is ok, since as object oriented programmers, we want to get away from this relational hooey anyway.)

The Summary Slide From the End

- The RDBMS is great - it's served us well for almost 40 years.
- We're in a kind of "renaissance" for databases
  - New problems challenge status quo architectures
  - Advances in distributed computing gives us powerful alternatives
- This is changing how we approach data in our apps
  - Different APIs
  - Different responsibilities as programmers
- This stuff works - people use it in anger
- Expand your professional toolbox - go play and learn

Gilt : What we do and why we needed !SQL

(and where I learned what a “Louboutin” was)

About Gilt Groupe (not a sales pitch)

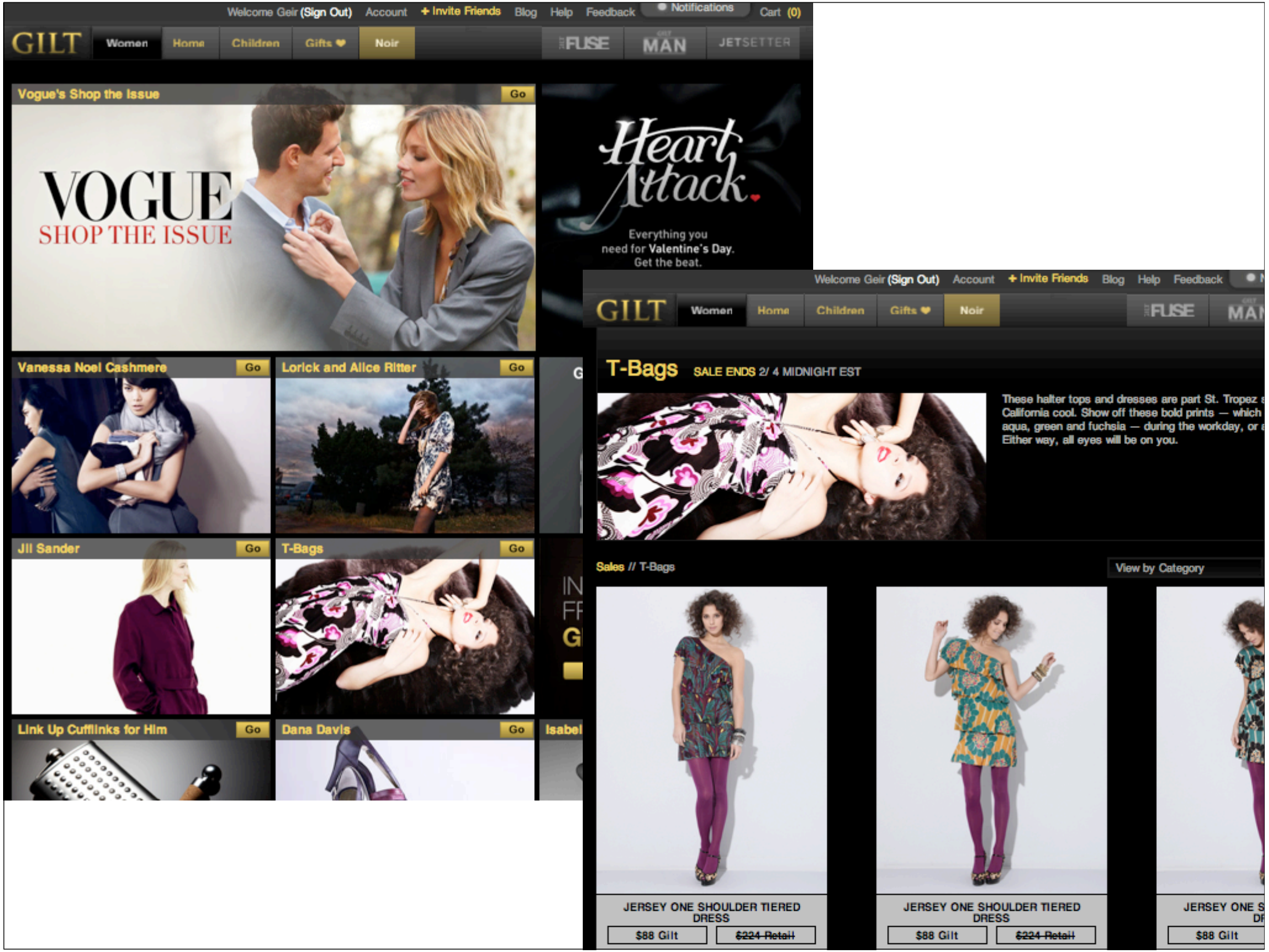
http://www.gilt.com/

Gilt Groupe provides access, by invitation only, to the world's best brands at prices up to 70% off retail.

Each sale lasts 36 hours and features hand selected styles from a single designer.

How does it work?

- Every day we run 10-20 sales of limited-inventory luxury goods
- Members know who the designers are, but not the specific items
- Sales begin at 12 o'clock sharp (EST)
- Members scramble to get items into shopping carts - can reserve for 10 minutes only


Welcome Geir (Sign Out) Account + Invite Friends Blog Help Feedback Notifications Cart (0)
GILT
Women
Home
Children
Gifts
Noir
FUSE
MAN
JETSETTER
Vogue's Shop the Issue
Go
VOGUE
SHOP THE ISSUE
Heart Attack
Everything you need for Valentine's Day. Get the beat.
Vanessa Noel Cashmere
Go
Lorick and Alice Ritter
Go
Jil Sander
Go
T-Bags
Go
Link Up Cufflinks for Him
Go
Dana Davis
Go
Isabel
Welcome Geir (Sign Out) Account + Invite Friends Blog Help Feedback
GILT
Women
Home
Children
Gifts
Noir
FUSE
T-Bags
SALE ENDS 2/ 4 MIDNIGHT EST
These halter tops and dresses are part St. Tropez
California cool. Show off these bold prints — which
aqua, green and fuchsia — during the workday, or
Either way, all eyes will be on you.
Sales // T-Bags
View by Category
JERSEY ONE SHOULDER TIERED DRESS
$88 Gilt
$224 Retail
JERSEY ONE SHOULDER TIERED DRESS
$88 Gilt
$224 Retail
$88 Gilt

Welcome Geir (Sign Out) Account + Invite Friends Blog Help Feedback Notifications Cart (0)

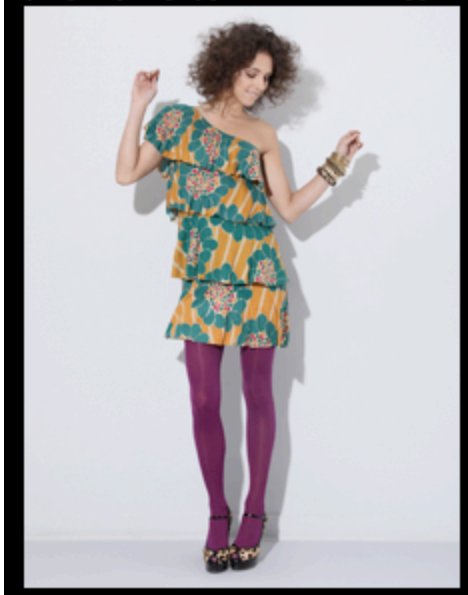
GILT Women Home Children Gifts ♥ Noir FUSE MAN JETSETTER

T-Bags SALE ENDS 2/ 4 MIDNIGHT EST

« Click here for sale page

Email to friend

JERSEY ONE SHOULDER TIERED DRESS



Move mouse over to zoom:

CLICK BOX TO SELECT SIZE:

p s m l

COLOR: NADIA 4

GILT PRICE
$88.00

ORIGINAL PRICE
~~$224.40~~

Buy Now

Sizing

STYLE INFO
Jersey flutter
and tiered par
from a size S
35 inches.

Color: Nadia
Material: 10
Fit: This styl
Origin: Unite

ESTIMATED

AUTHENTIC
We guarantee
Every brand t

RETURN PO
This item may

YOU WOULD

Welcome Geir (Sign Out) Account + Invite Friends Blog Help Feedback

GILT Women Home Children Gifts ♥ Noir FUSE MAN JETSETTER

Confirm your order

1. Pay with: American Express
Ends with: 1007
Exp date: 11/13
Change

2. Ship to: Geir Magnusson Jr
74 Old Belden Hill Rd
Wilton, CT 06897
Change

Estimated Delivery:
Mon 2/22/10 To Wed 3/3/10

- (•) UPS Ground
- ( ) UPS Overnight

Want to include a gift message?

Submit Order

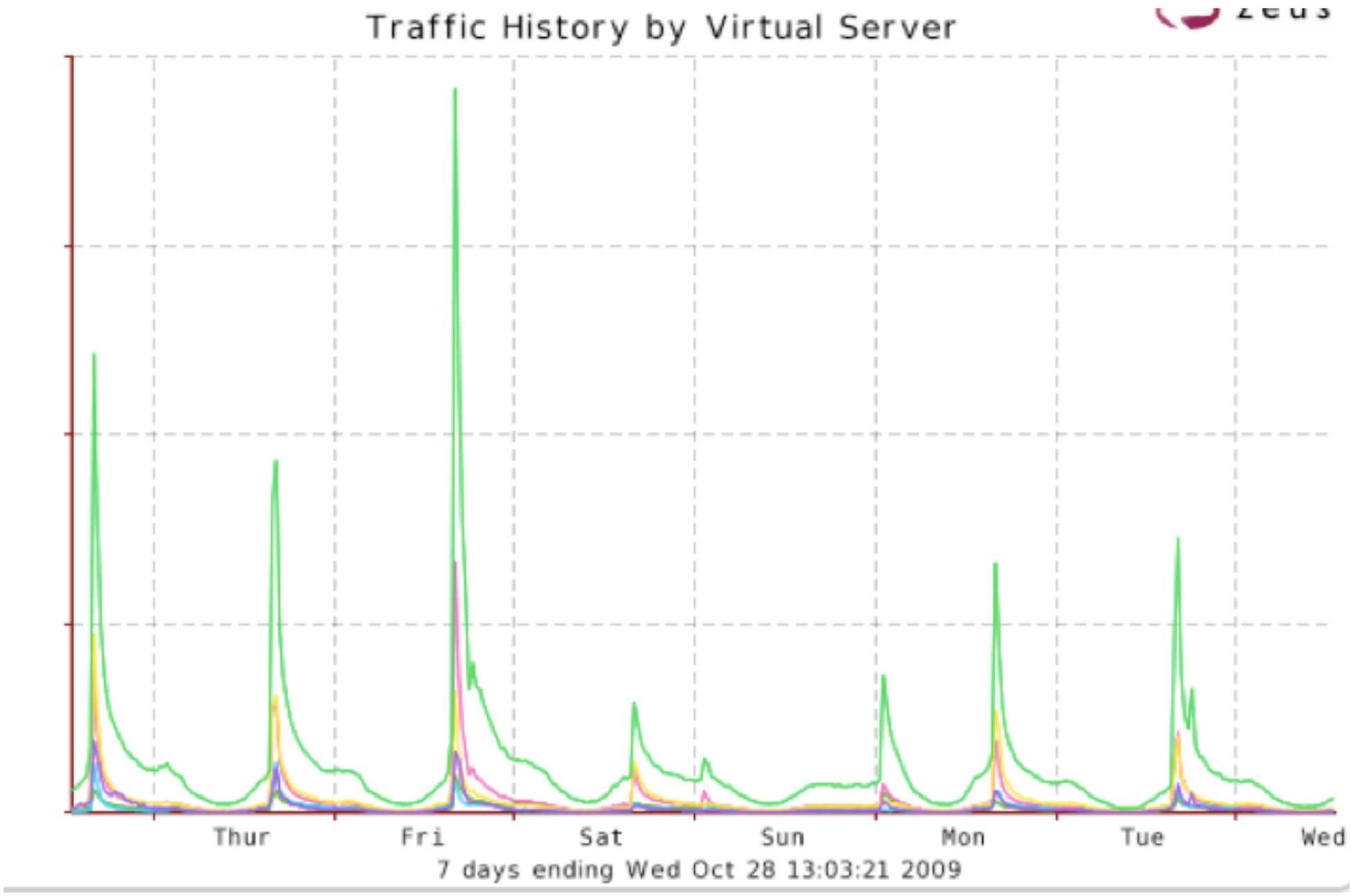
Your Order

Order Subtotal:	$51.00
Shipping:	$10.00
Sales Tax:	$0.00
Credit:	-$61.00
TOTAL:	$0.00

T-Bags
Jersey One Shoulder
Tiered Dress
Color:nadia 4
Size:s
Qty:1
Price:$51.00
Estimated Arrival:
Mon 2/22/10 to Wed 3/3/10

ABOUT CAREERS SIZING FAQ

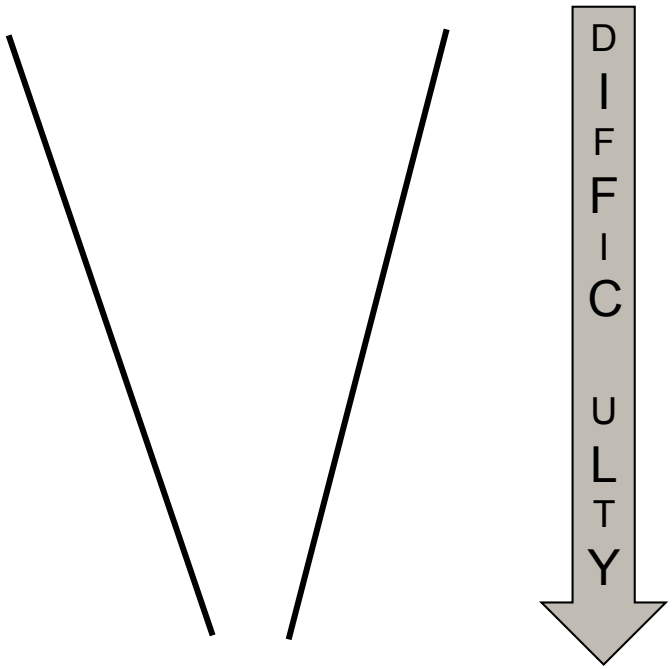
RETURN POLICY SHIPPING & TAX TERMS OF SERVICE PRIVACY


Traffic History by Virtual Server
Thur
Fri
Sat
Sun
Mon
Tue
Wed
7 days ending Wed Oct 28 13:03:21 2009

From an actual member...

“Today was round II of Gilt Groupe's Final Sale.
[…]
I clicked BUY NOW, and it was in someone's shopping cart so I proceeded to click BUY NOW, BUY NOW, BUY NOW, BUY NOW, BUY NOW, BUY NOW, BUY NOW, BUY NOW, BUY NOW, BUY NOW,BUY NOW, BUY NOW, BUY NOW, BUY NOW, BUY NOW, for the next 5 minutes and then.....
a shopping angel reigned down from heaven and it was in my shopping cart! I scored the ADAM find that was normally $375 for $68.”

Activity Funnel

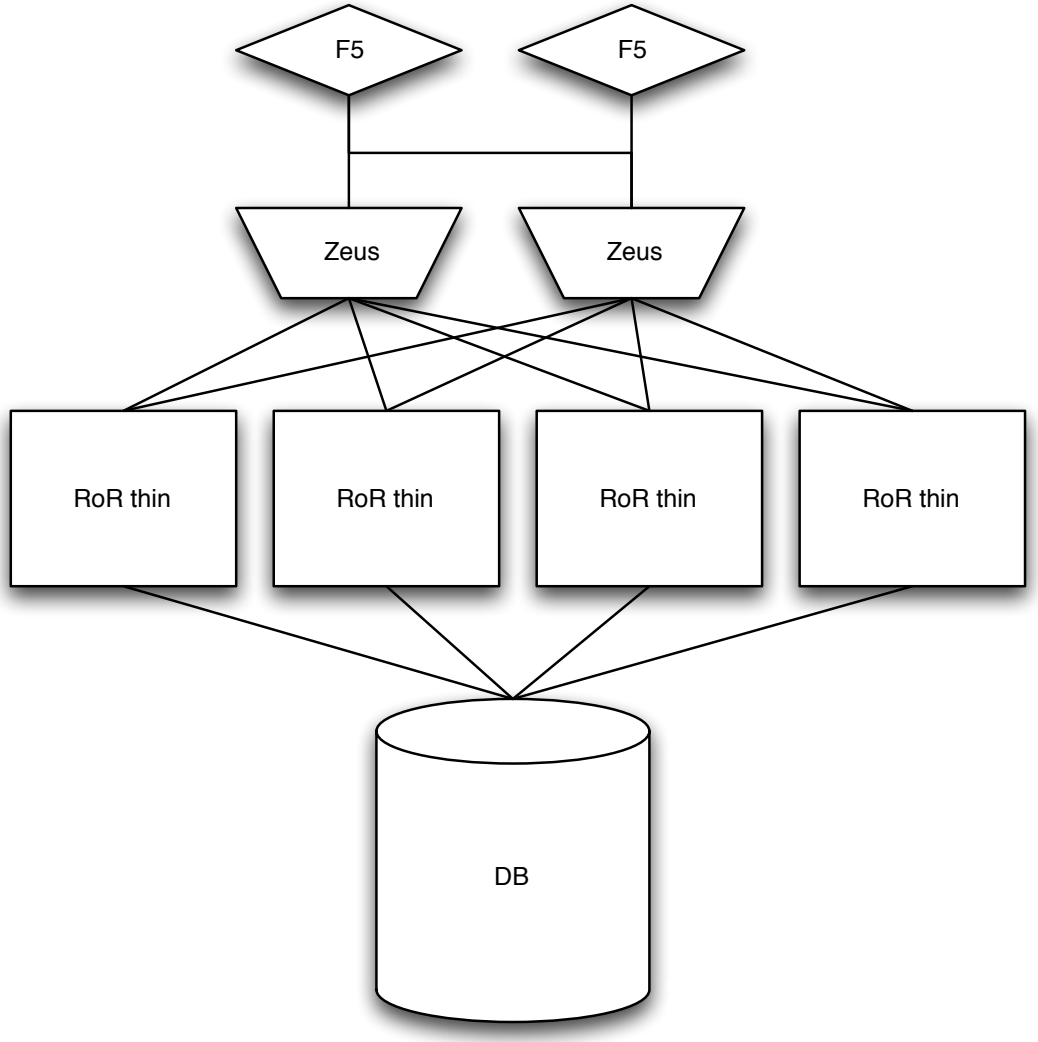


A) Millions of page views / hour, fast ramp up

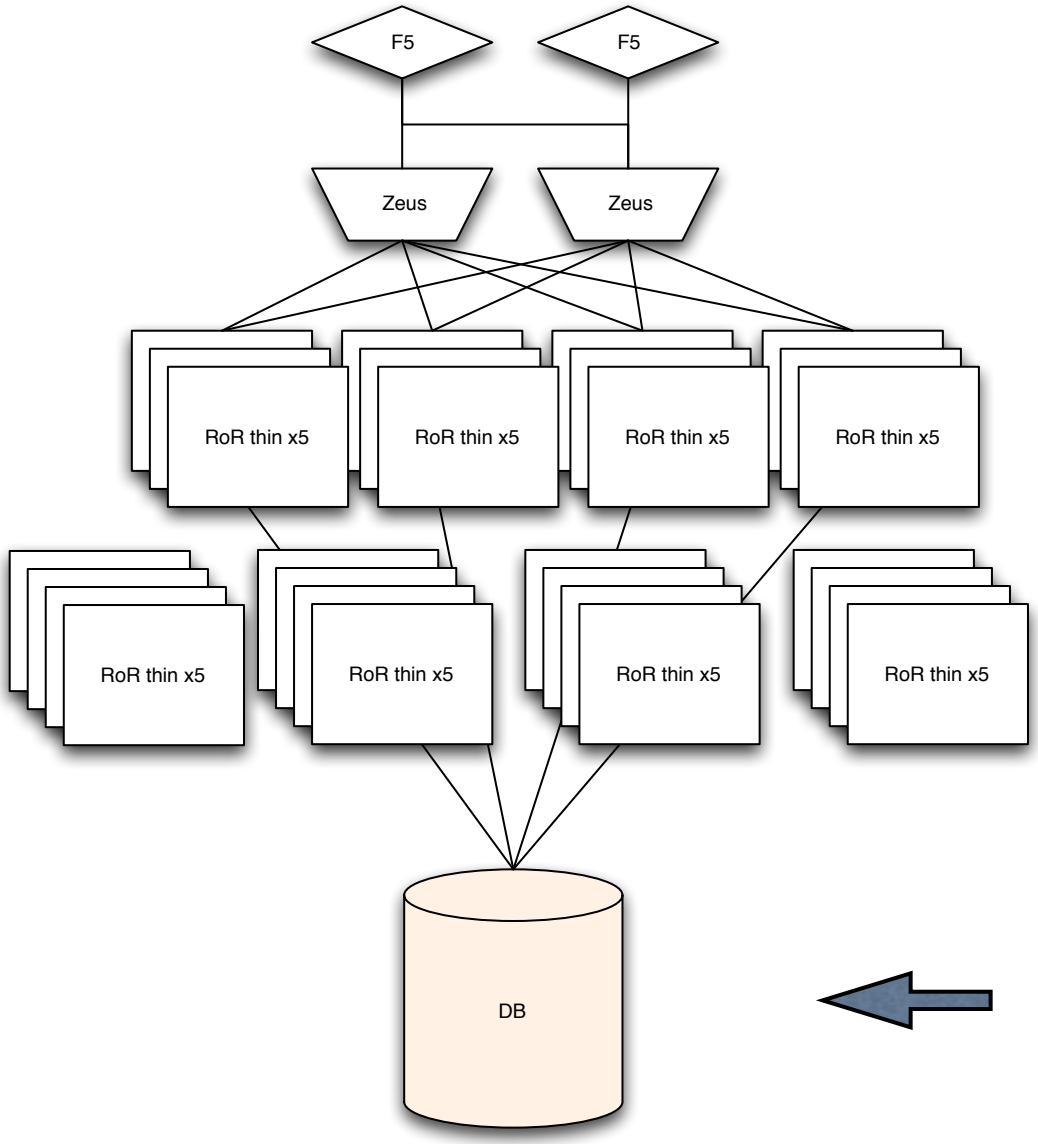
B) High volume transactions (registration, login, wait list)

C) High volume, shared state (add to cart, checkout)

“Shared nothing” Architecture



Are you sure it's "Shared-Nothing"?

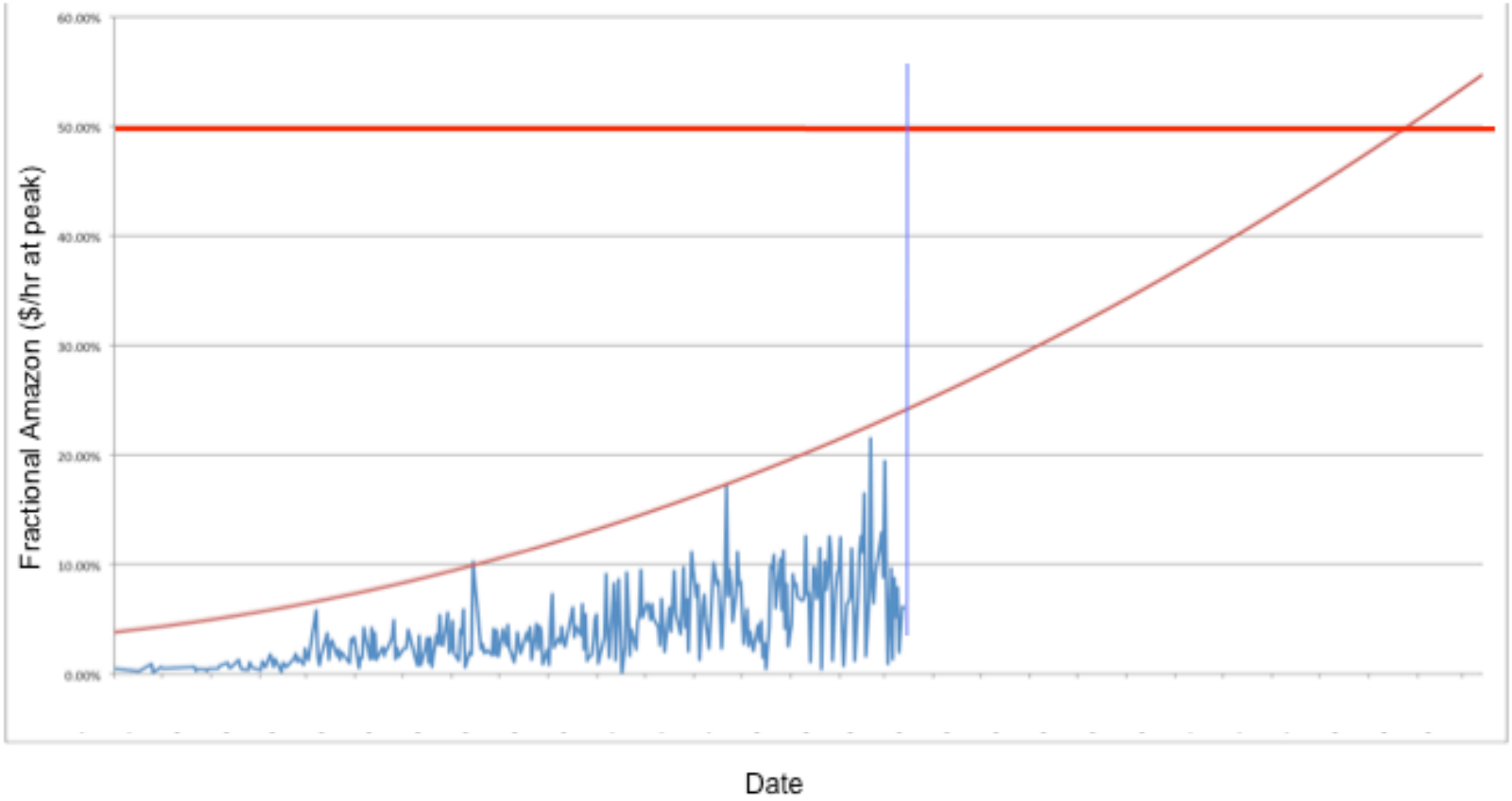


Nothing is shared!

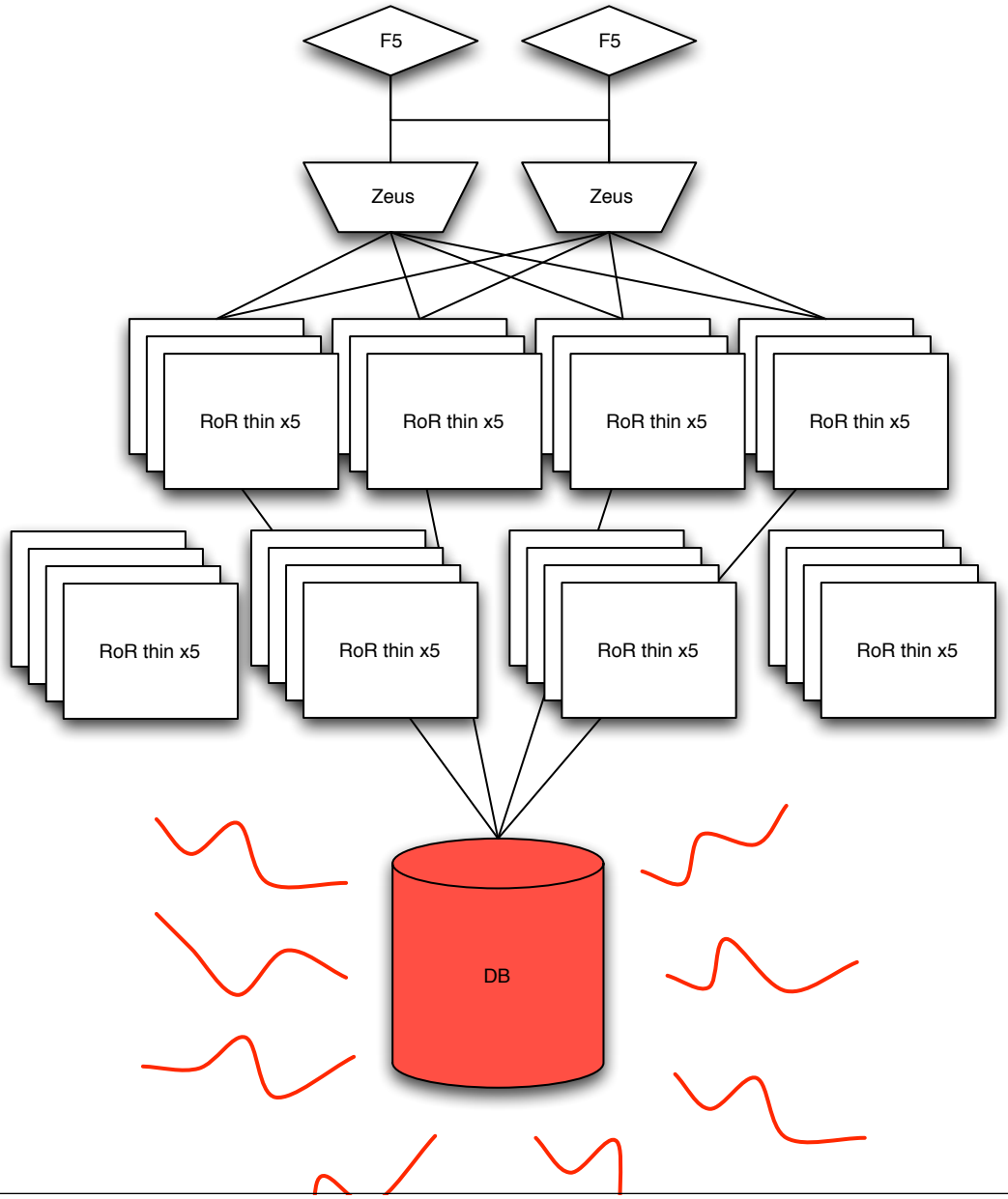
Don't look here. Nothing to see here. Move along.


Paris

“Half an Amazon”

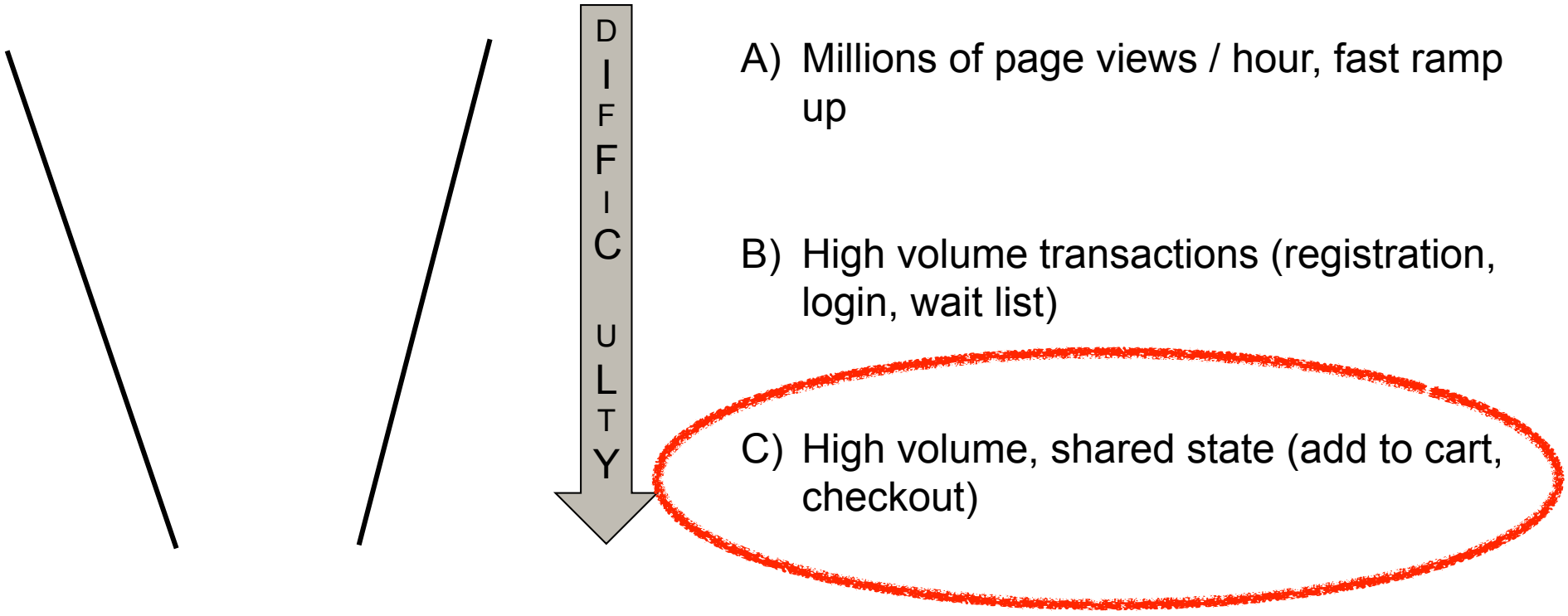


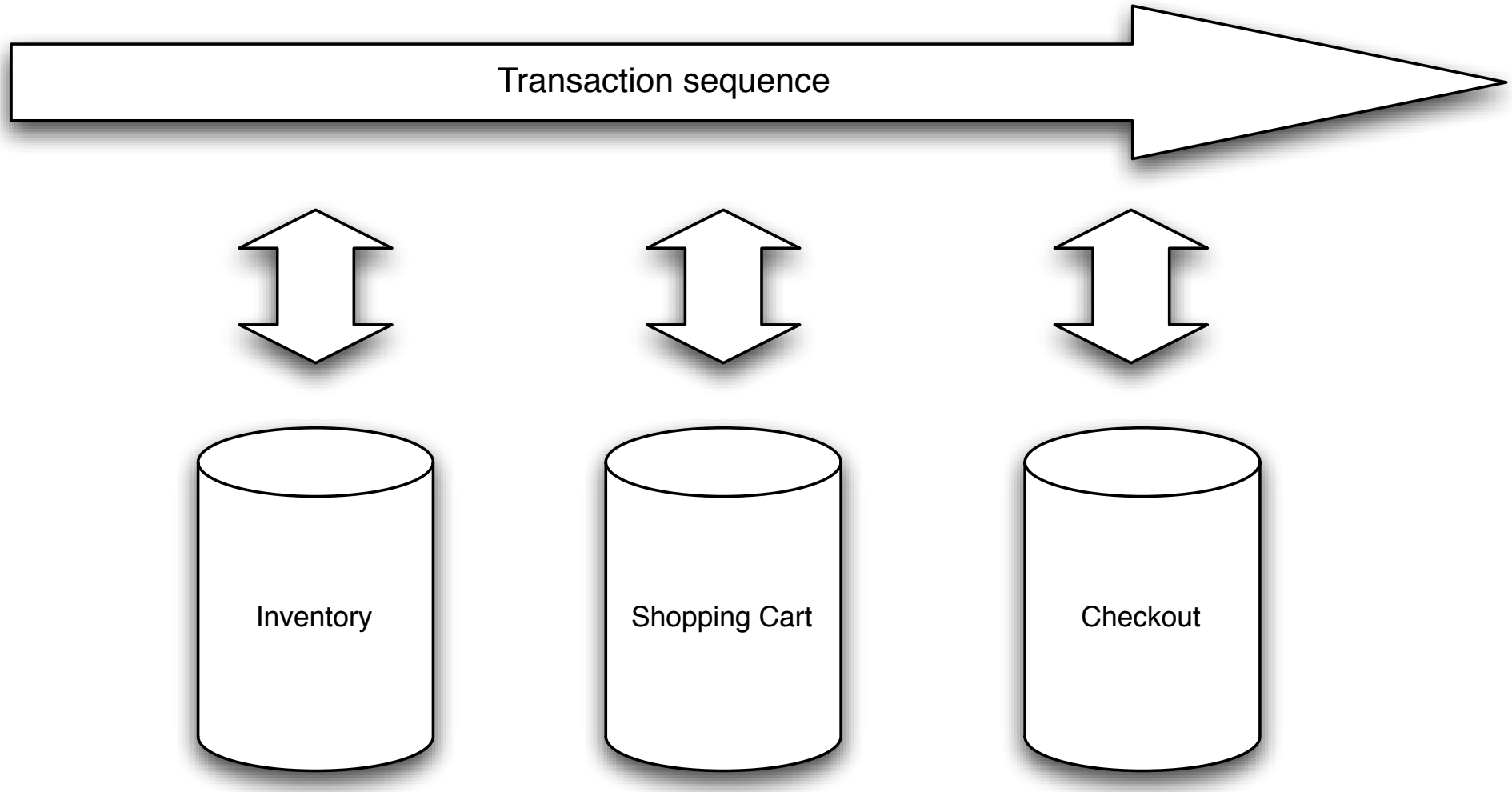
“What’s that burning smell?”



Goal : Turn off the RDBMS at peak

Activity Funnel



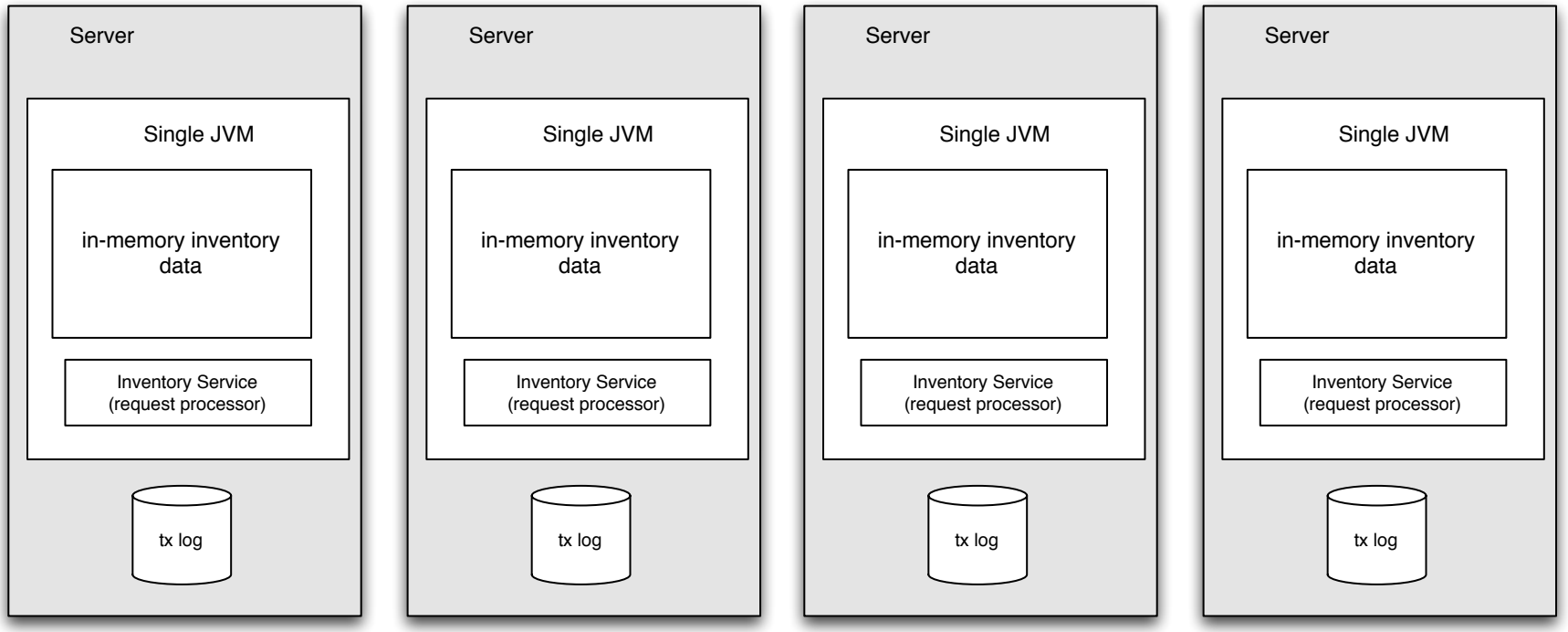

Transaction sequence
Inventory
Shopping Cart
Checkout

Inventory Management

- This is our highest transactional load
- Must be sure to provide a ‘reservation’ to a product unit once and only once
- Must be fast *and* durable

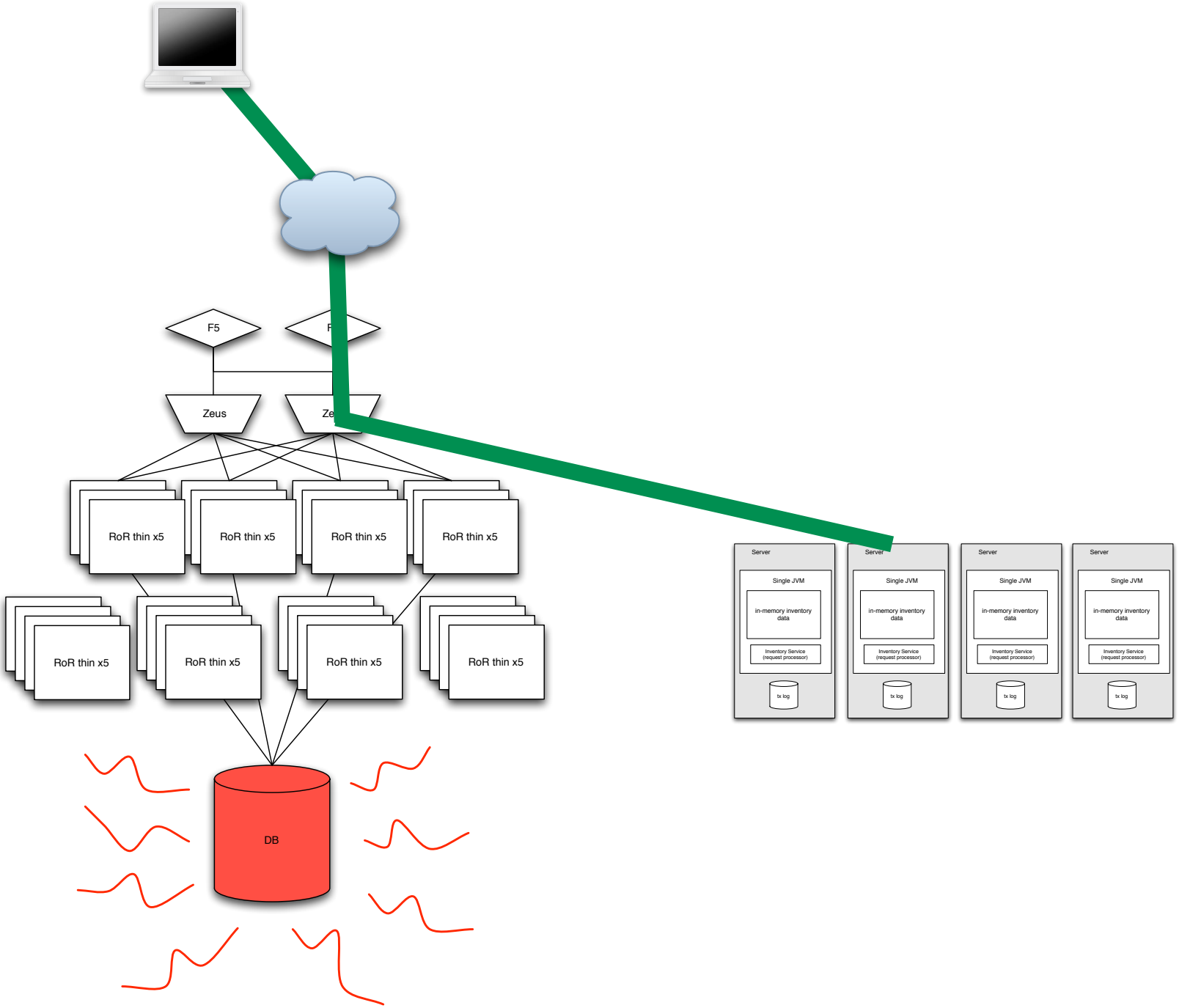
Inventory Solution

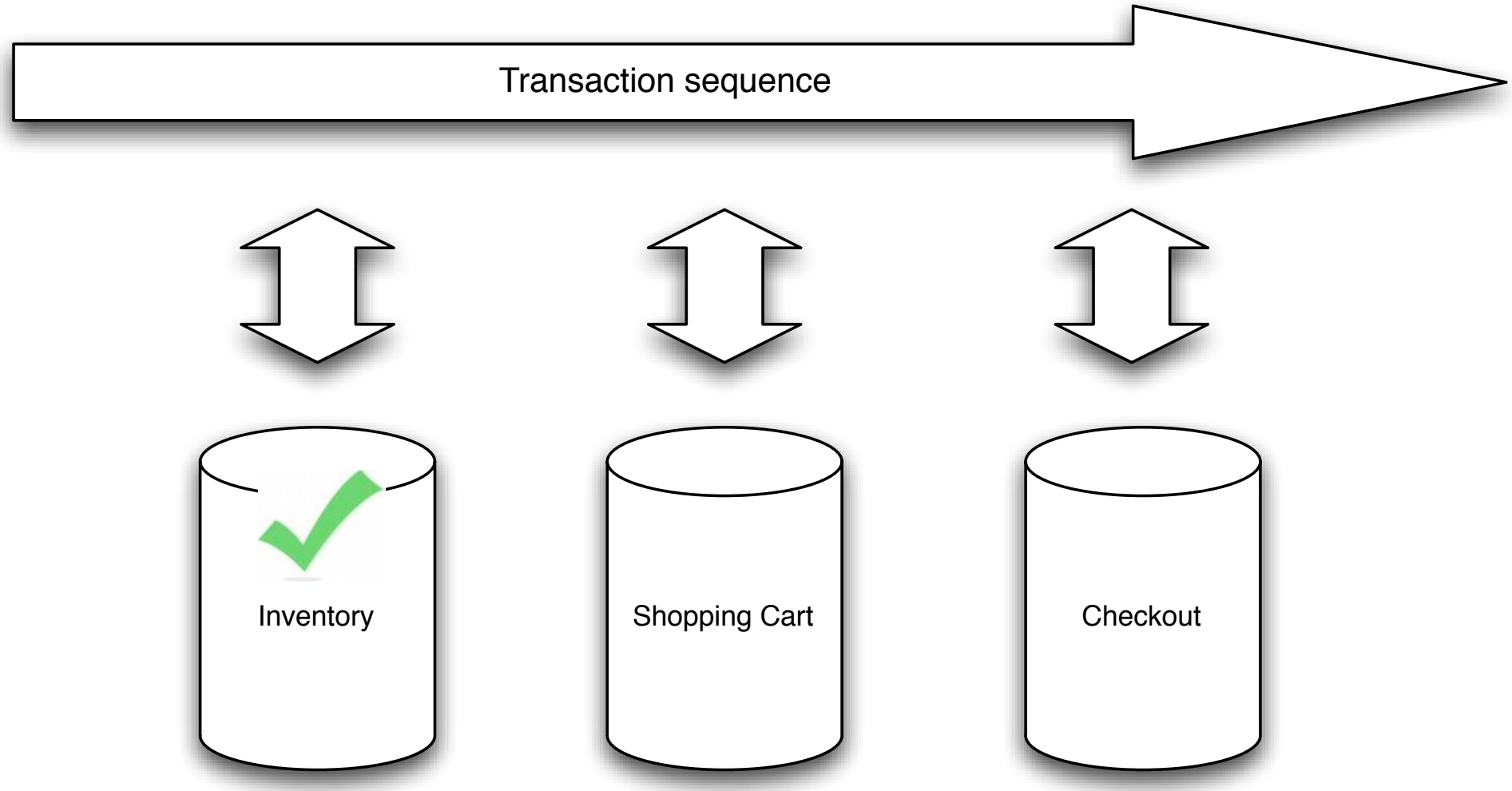
- Partition inventory so horizontally scalable
- Custom server keeps all assigned inventory in memory
- All operations are in memory, transactional - lock at SKU level
- Local write-behind transaction log for recovery



partition 0 partition 1 partition 2 partition 3

DB Shielded from Inventory Requests




Transaction sequence
Inventory
Shopping Cart
Checkout

Shopping Cart and Order Processing

- Shopping Cart : High tx activity and churn on hundreds of thousands of ~5k documents. Speed and availability important, less worried about losing data. (single write)
- Order processing : Lower tx activity, need high-availability and multi-copy writes (we don't want to lose them!)

Need availability and speed

- We decided early on that Amazon's Dynamo approach was the way to go

  http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf

- Project Voldemort was the only implementation at the time in production that we could find

  http://project-voldemort.com

- I'm a Java Weenie (tm) so I like the fact that it's written in Java

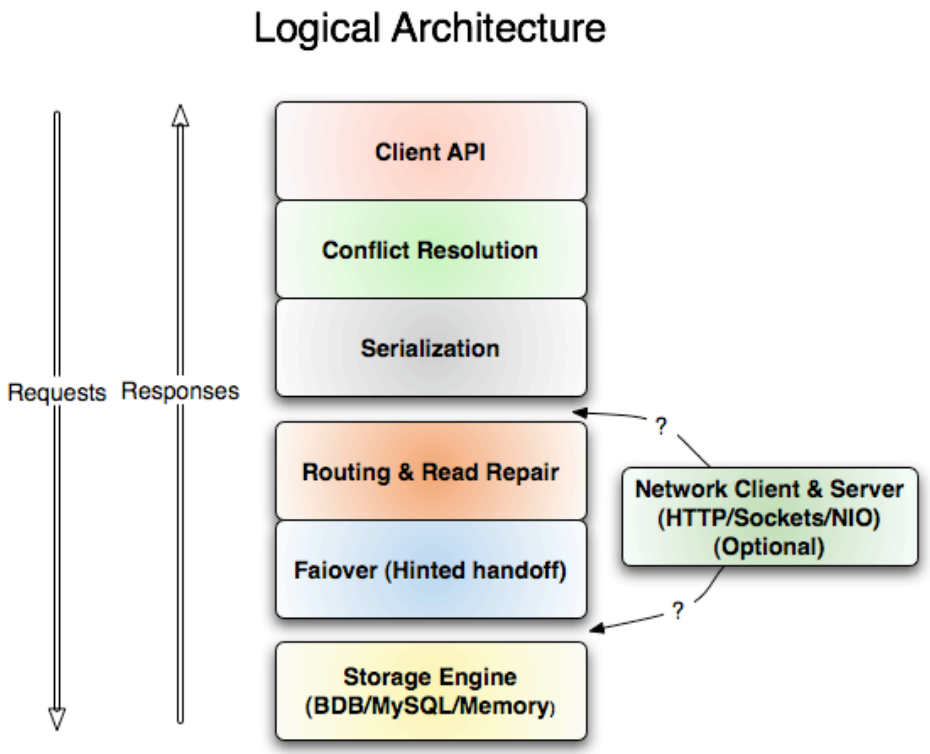
What is Project Voldemort

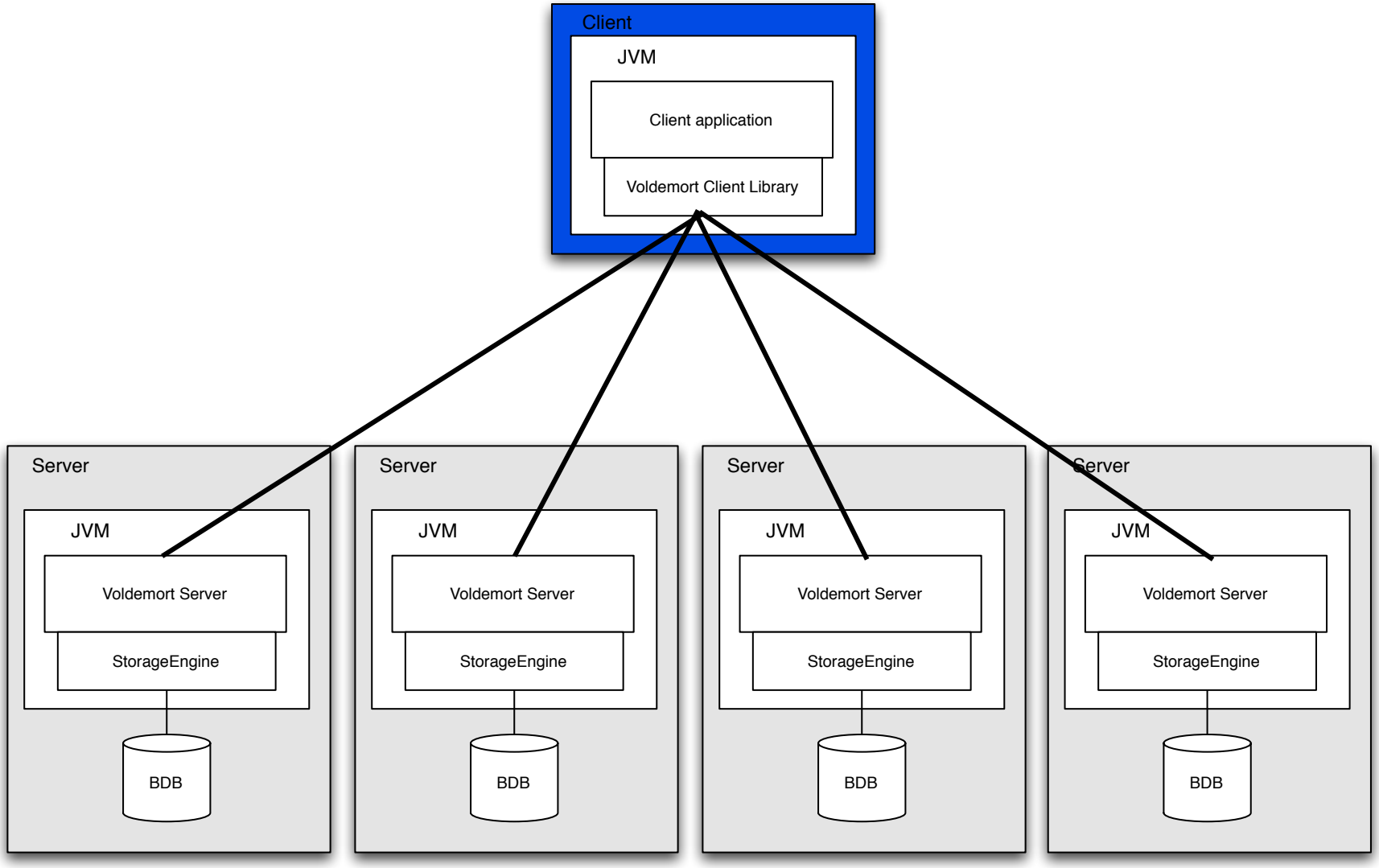
- Distributed "key value" store designed for *availability*

  Survive server failures and network partitions

- Combines several techniques :
  - Decentralized architecture - no master
  - Data partitioned and replicated via *consistent hashing*
  - Multi-node reads and writes for redundancy
  - Objects are versioned for *consistency*
  - Pluggable persistence

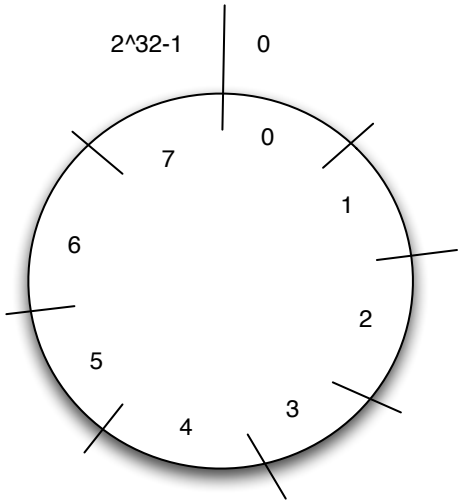
Basic Architecture

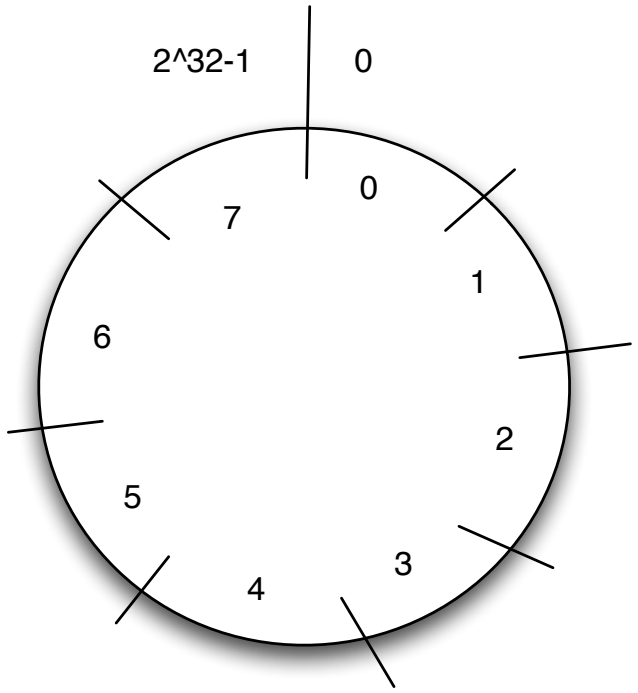


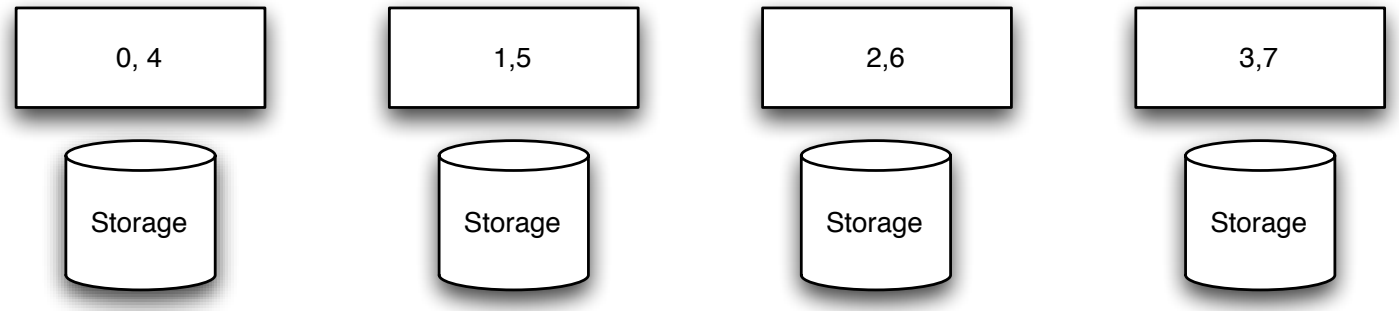

Client
JVM
Client application
Voldemort Client Library
Server
JVM
Voldemort Server
StorageEngine
BDB
Server
JVM
Voldemort Server
StorageEngine
BDB
Server
JVM
Voldemort Server
StorageEngine
BDB
Server
JVM
Voldemort Server
StorageEngine
BDB

Consistent Hashing

- Keys hash to a point on fixed circular space
- Circular space is divided into a large set of ordered buckets, called nodes
- Nodes are distributed across servers




2^32-1
0
0
1
2
3
4
5
6
7


0, 4
1,5
2,6
3,7
Storage
Storage
Storage
Storage

Vector Clocks

- Mechanism to disambiguate between versions of the same object
- Non-locking optimistic locking
- A vector clock is a list of (nodeID, counter) tuples
- Every object has a vector clock, which is updated on each write, and examined on each read
- Explicit in the client API

Vector Clocks

When an object is read, if there are multiple versions and Voldemort can't figure it out... you have to!

3 servers : Sx, Sy, Sz

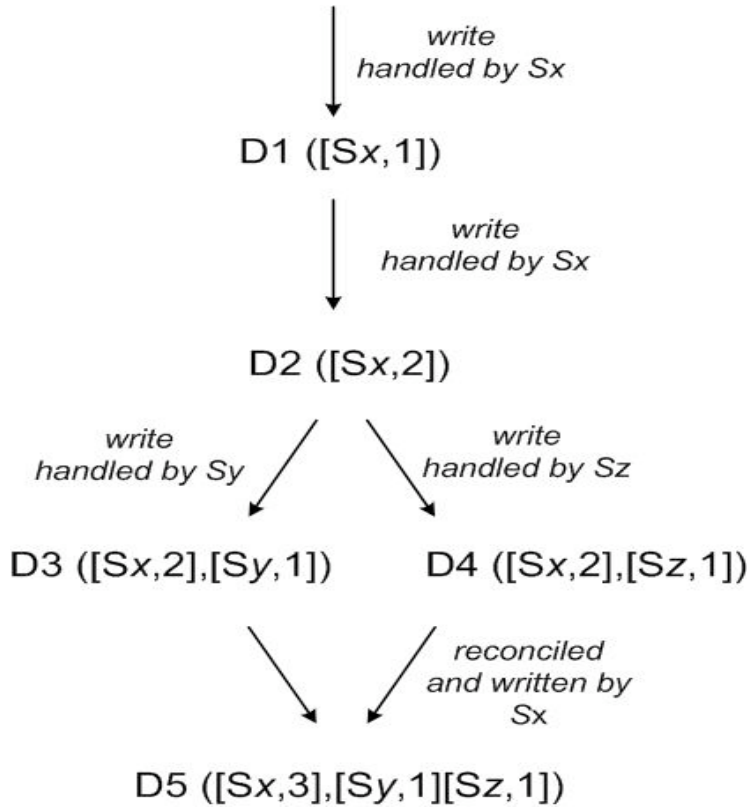
Sequence of writes :

D1

D2

D3 / D4

D5



Storage and Serialization

- Voldemort is local storage and serialization agnostic. Both are pluggable
- Different needs require different solutions
- Storage choices of :
  - ➡ BDB, MySQL, memory, Hadoop (RO), MongoDB
- Serialization choices of :
  - ➡ String, JSON, Protobuf, Thrift...

Storage Configuration

Data organized into named stores that have independent configurations

- storage engine
- request routing parameters

  R : num reads required,

  W : num writes required

  N : replication factor

Client API

[(value, version), ...] get (key)

[[(value, version), ...]] getAll([key1, key2, ...])

put(key, value, version)

delete(key)

delete(key, version)

Doing a get(key)

- hash the key and figure out what node it maps to.
- Starting with the *next* node that is live, get sequential list of N nodes that are live
- Read from nodes until you get R responses back.
- Compare results (compare vector clocks) and return one or more responses to client

Doing a put(key, value, version)

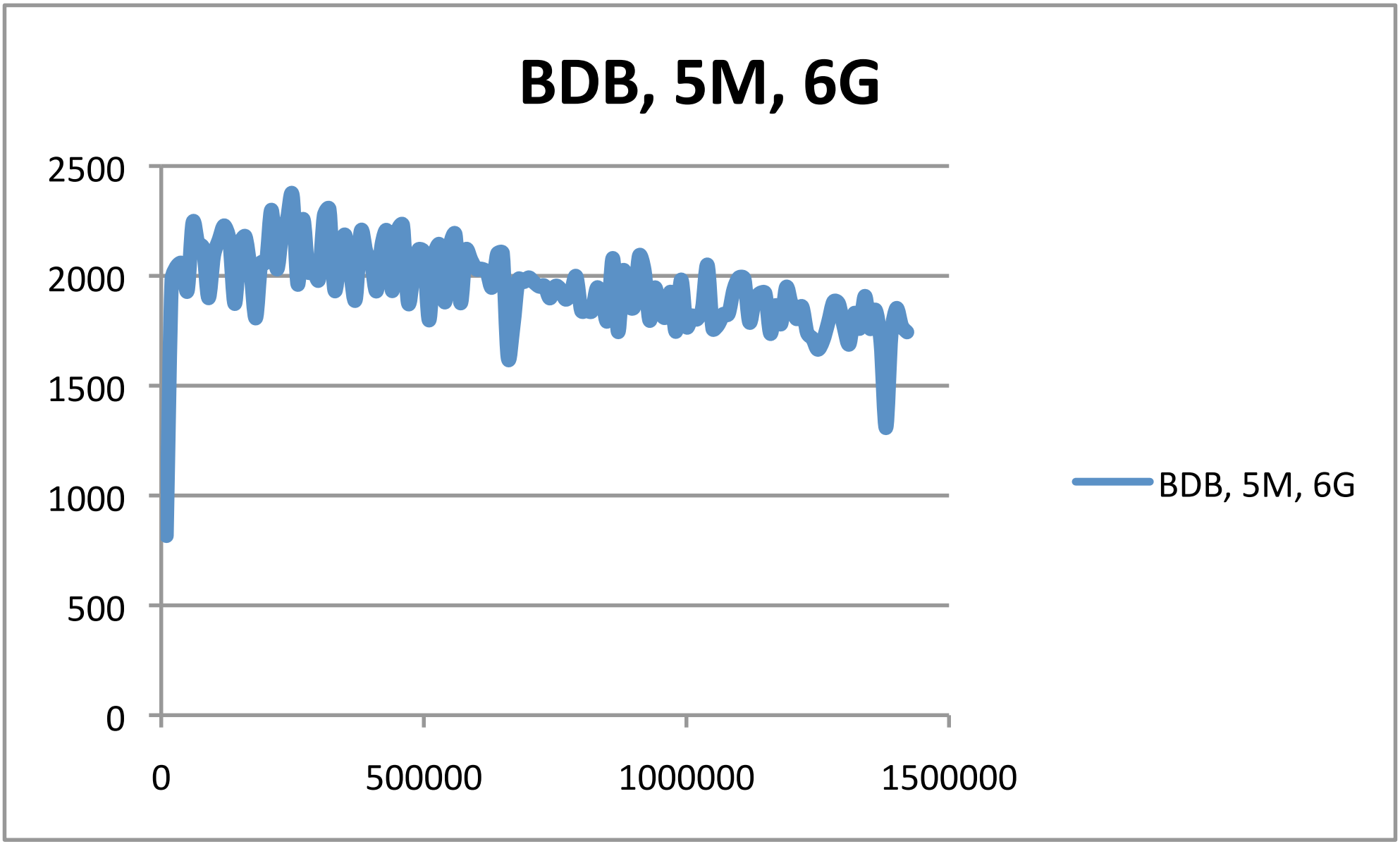
- hash the key and figure out what node it maps to.
- Starting with the *next* node that is live, get sequential list of N nodes that are live
- Write to all N nodes and then wait for W successful responses back

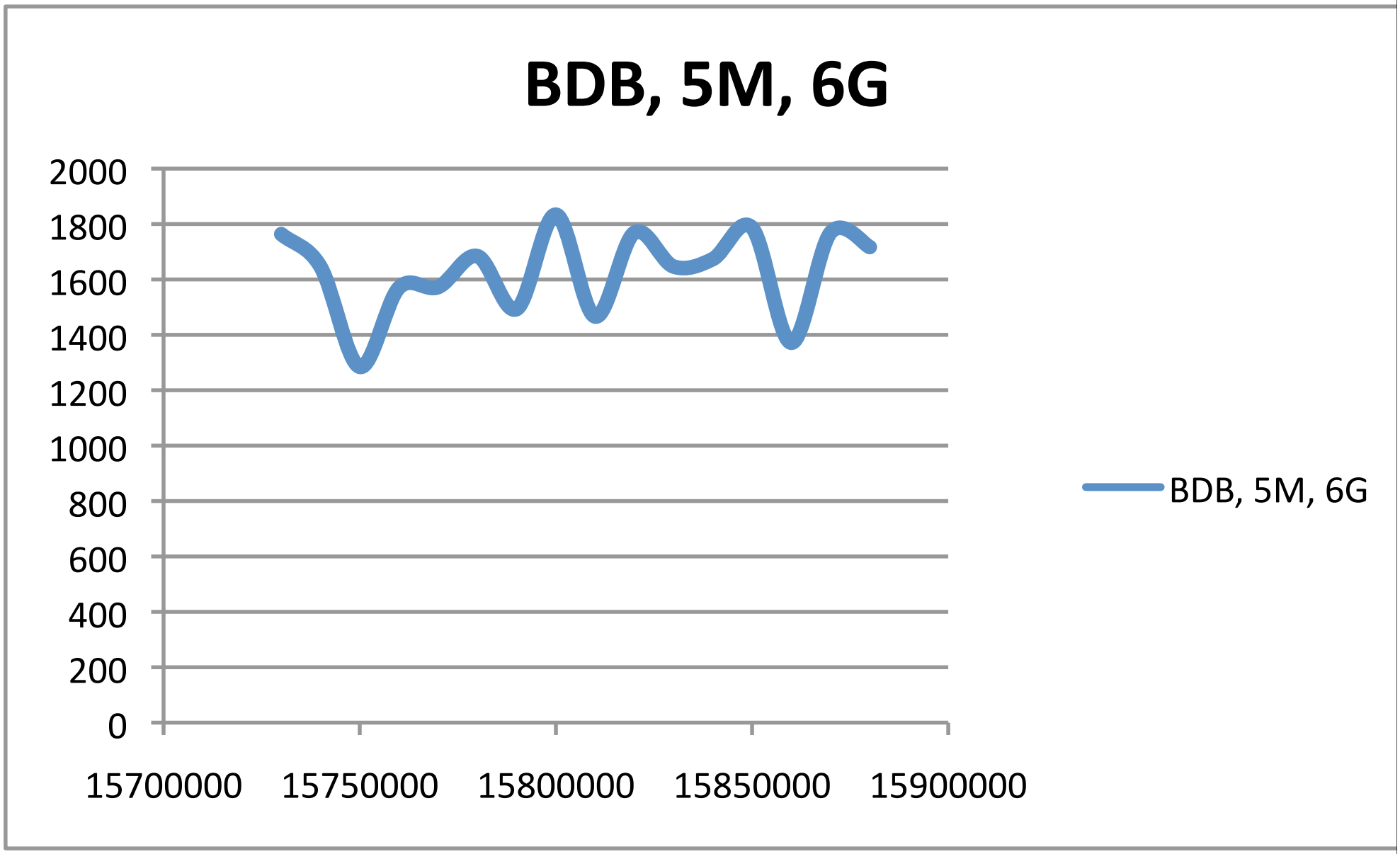
Choosing A Store

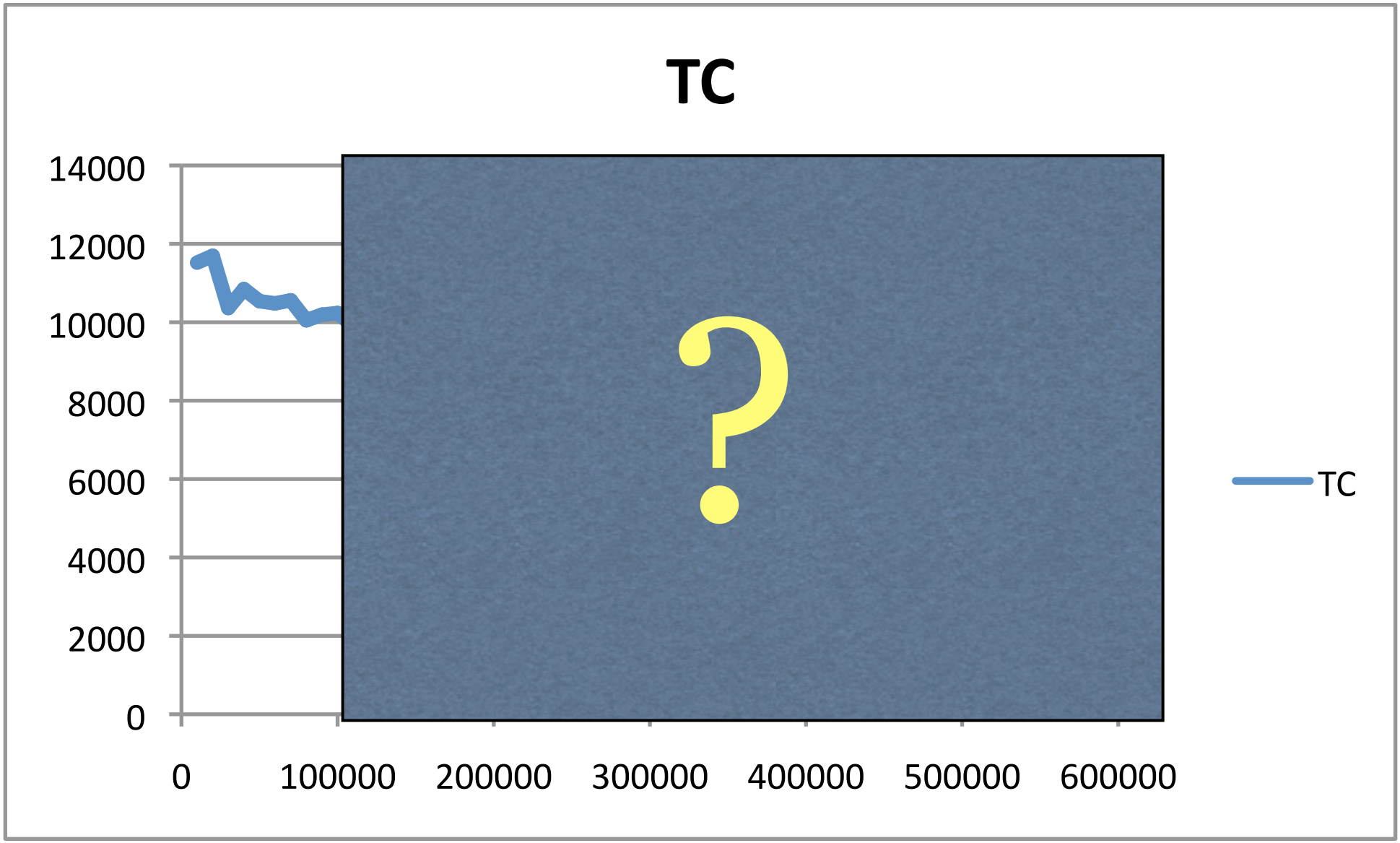
- Goal : for shopping cart (5k JSON doc), find store that has predictable, consistent, low-maintenance behavior
- We looked at
  - BDB-J
  - BDB-C
  - MySQL
  - MongoDB
  - Tokyo Tyrant
  - H2

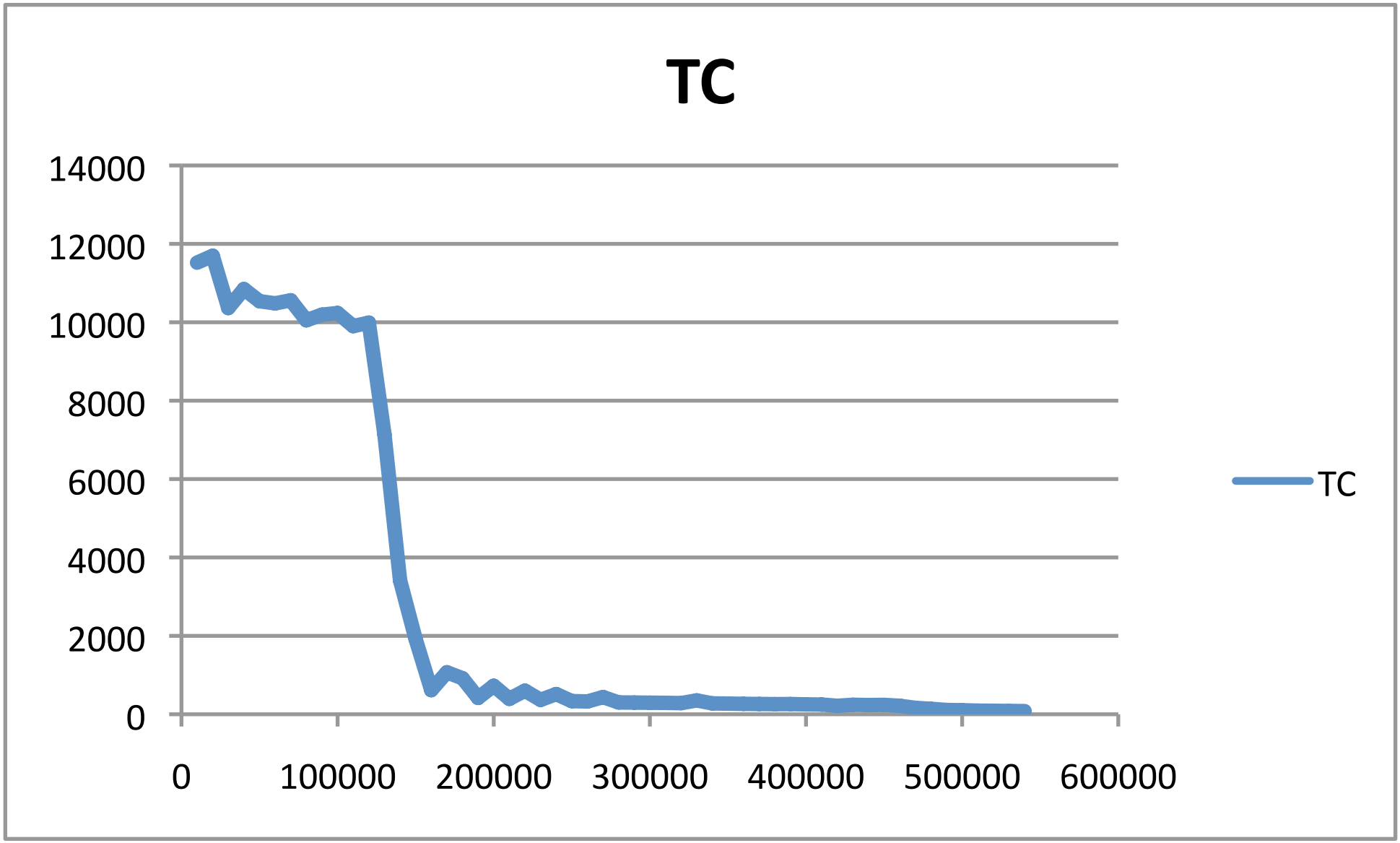
Cart Simulations

- 5KB-sized documents
- fill store with 1MM documents
- keys are [1,1000000]
- choose key at random from range (x, y)
- do get(key), put(key, value)


BDB, 5M, 6G
2500
2000
1500
1000
500
0
0
500000
1000000
1500000
BDB, 5M, 6G


BDB, 5M, 6G
2000
1800
1600
1400
1200
1000
800
600
400
200
0
15700000
15750000
15800000
15850000
15900000
BDB, 5M, 6G


TC
14000
12000
10000
8000
6000
4000
2000
0
0
100000
200000
300000
400000
500000
600000
?
TC


TC
14000
12000
10000
8000
6000
4000
2000
0
0
100000
200000
300000
400000
500000
600000
TC

Our KV Store

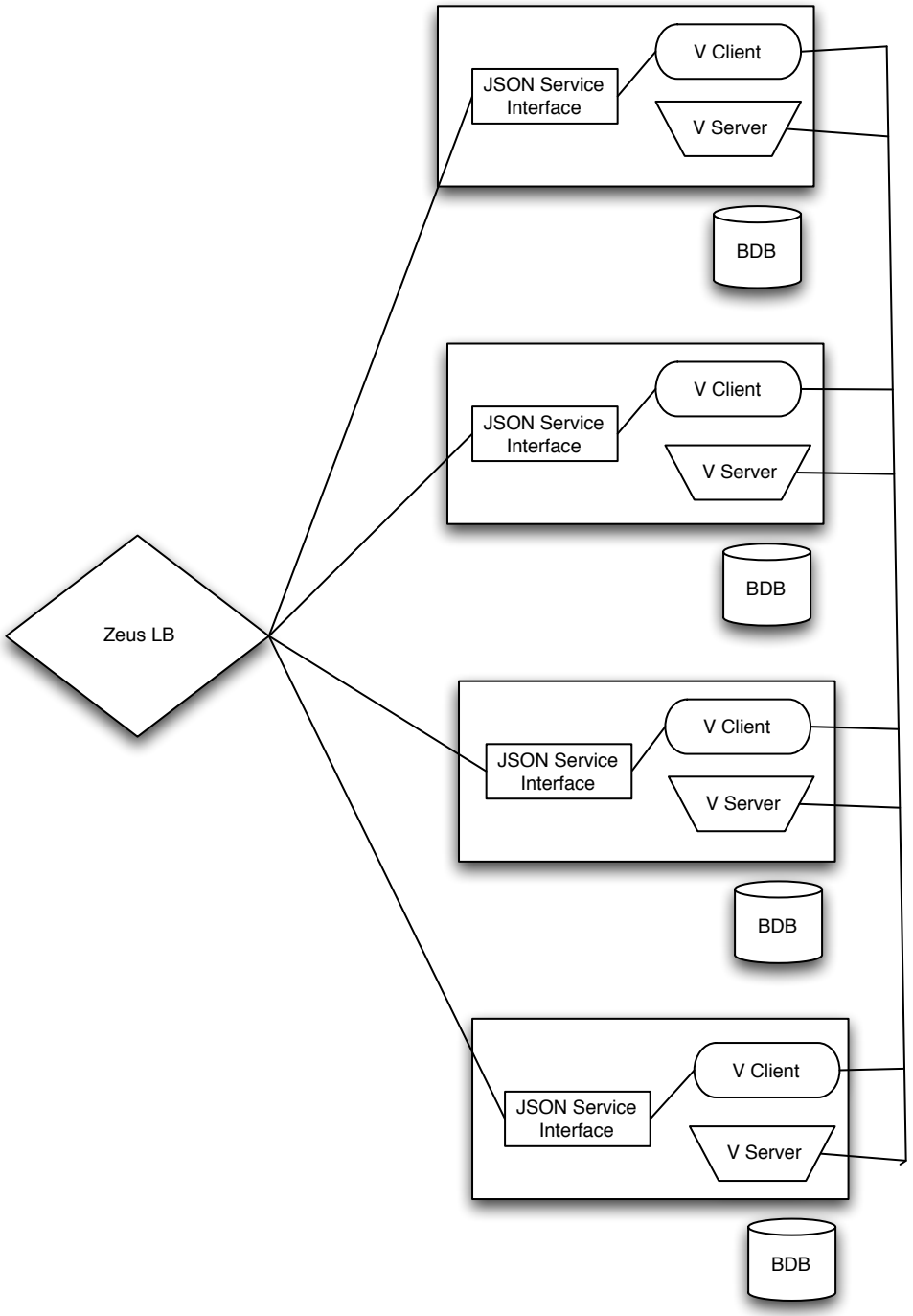
- KV persistence service in our architecture
- JSON over HTTP
- Embed V client as well as V server in the service container

/kvstore/get?store=cart&key=12312312

2009-10-22 15:58:02,868 [1986749137@qtp-1533779554-0] INFO
com.gilt.svc.framework.servlet.ServiceServlet - 17 ms : GET : 127.0.0.1 : - : /kvstore/get?store=cart&key=1 : {
  "status" : 0,
  "msg" : "ok",
  "request" : "/kvstore//get?store=cart&key=1",
  "timestamp" : "Thu, 22 Oct 2009 19:58:02 UTC",
  "nodename" : "pthbbb-2",
  "nodeID" : 0,
  "data" : {
    "values" : [{
      "value" : "{\n \"sku_info\" : {\n },\n \"cart_id\" : \"1\"   \"sku_id\" : 1,\n   \"sale_id\" : 1\n }]\n}",
      "success" : true,
      "version" : "0001010000020000001247dd230a4"
    }],
    "store" : "cart",
    "key" : "1"
  }
}

/kvstore/put?store=cart&key=123123123&version=X&value=Y

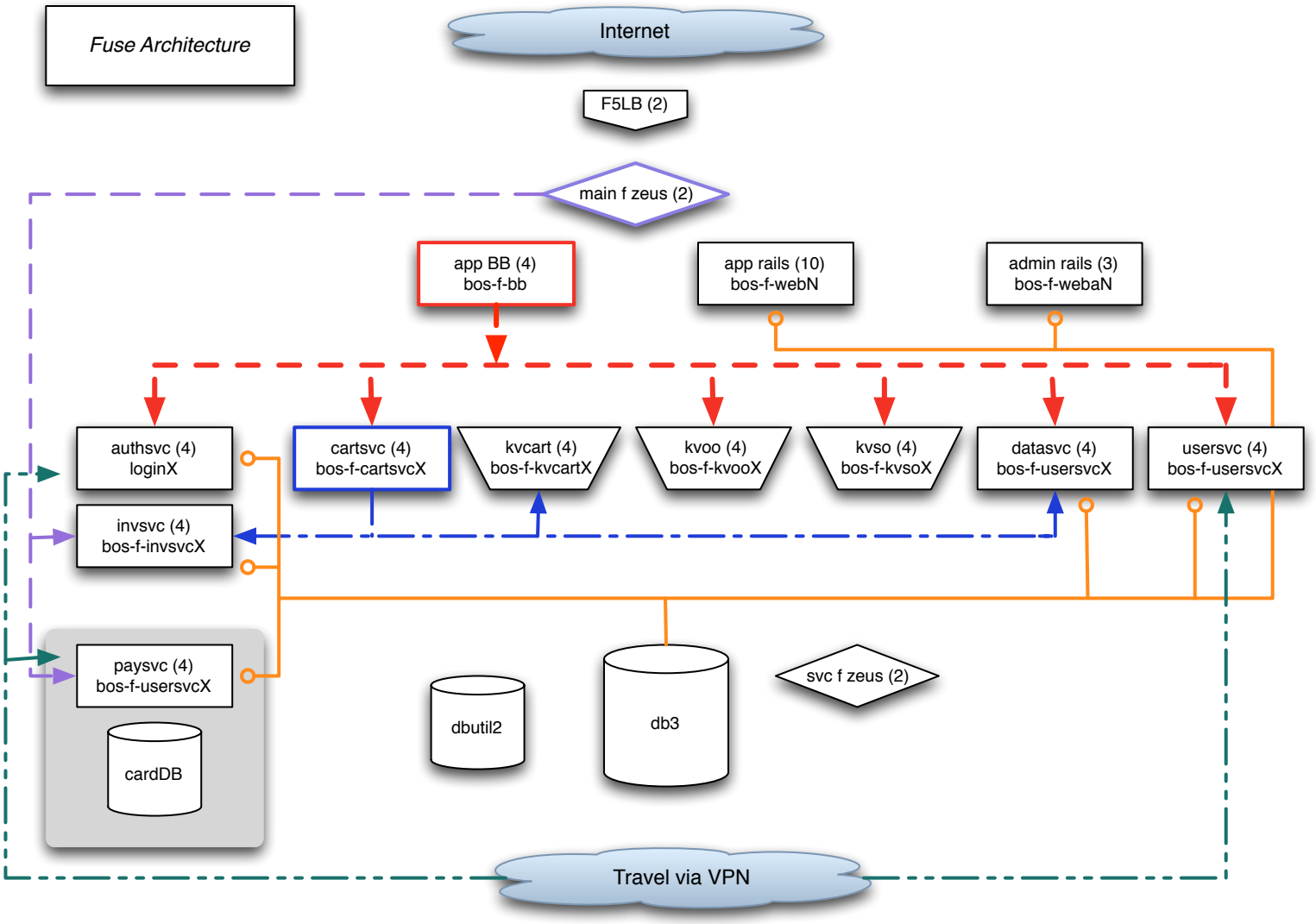
2009-10-22 15:56:17,456 [1986749137@qtp-1533779554-0] INFO
com.gilt.svc.framework.servlet.ServiceServlet - 43 ms : POST : 127.0.0.1 : - : /kvstore/put : {
  "status" : 0,
  "msg" : "ok",
  "request" : "/kvstore/put",
  "timestamp" : "Thu, 22 Oct 2009 19:56:17 UTC",
  "nodename" : "pthbbb-2",
  "nodeID" : 0,
  "data" : {
    "version_string" : "version(0:2)",
    "store" : "cart",
    "success" : true,
    "key" : "1",
    "version" : "000101000001000001247dd16ca6"
  }
}


Zeus LB
JSON Service Interface
V Client
V Server
BDB
JSON Service Interface
V Client
V Server
BDB
JSON Service Interface
V Client
V Server
BDB
JSON Service Interface
V Client
V Server
BDB

It works

- In production since August 2009 with Gilt Fuse
- Full Gilt production load Sept 2009
- Uptime measured in months

Gilt Service Architecture



Summary

- The RDBMS is great - it's served us well for almost 40 years.
- We're in a "renaissance" for databases
  - New problems challenge status quo architectures
  - Advances in distributed computing gives us powerful alternatives
- This is changing how we approach data in our apps
  - Different APIs
  - Different responsibilities
- Expand your professional toolbox - go play and learn

Thanks!

geir@pobox.com

GILT GROUPE

QCon