John Davies
CTO
Incept5

Caching, NOSQL & Grids
What the banks can teach us

QCon
London
9th March 2012

John Davies

- An ageing "Über-geek"
  - Hardware, Assembler, C, Objective-C, C++, OCCAM, SmallTalk, Java
  - Worked mostly in trading systems, FX & Derivatives
  - Head of trading systems at Paribas, head of architecture at BNP Paribas, global head of architecture at JP Morgan
  - Author of Learning Trees Enterprise Java courses & co-author of several Java & architecture books
- Co-founder of C24 Solution in 2000
  - Sold to Nasdaq's Iona Technologies in 2007, Iona sold to Progress Software in 2008, Technical Director of both companies
- Co-founded Incept5 in 2008, re-acquired C24 from Progress in April 2011
  - CTO of both Incept5 & C24
  - Original technical architect behind Visa's V.me (pre-public release)

Incept5

What we're going to look at today...

- SQL or NoSQL - Good or Bad?
  - If Java better than SQL?
  - What can't SQL do?
- The need to scale
  - Too much data to store on spinning disks of iron
  - Too many calculations
- Virtualisation
  - Distributed computing, grids, private and public cloud
- Look at a few grid technologies

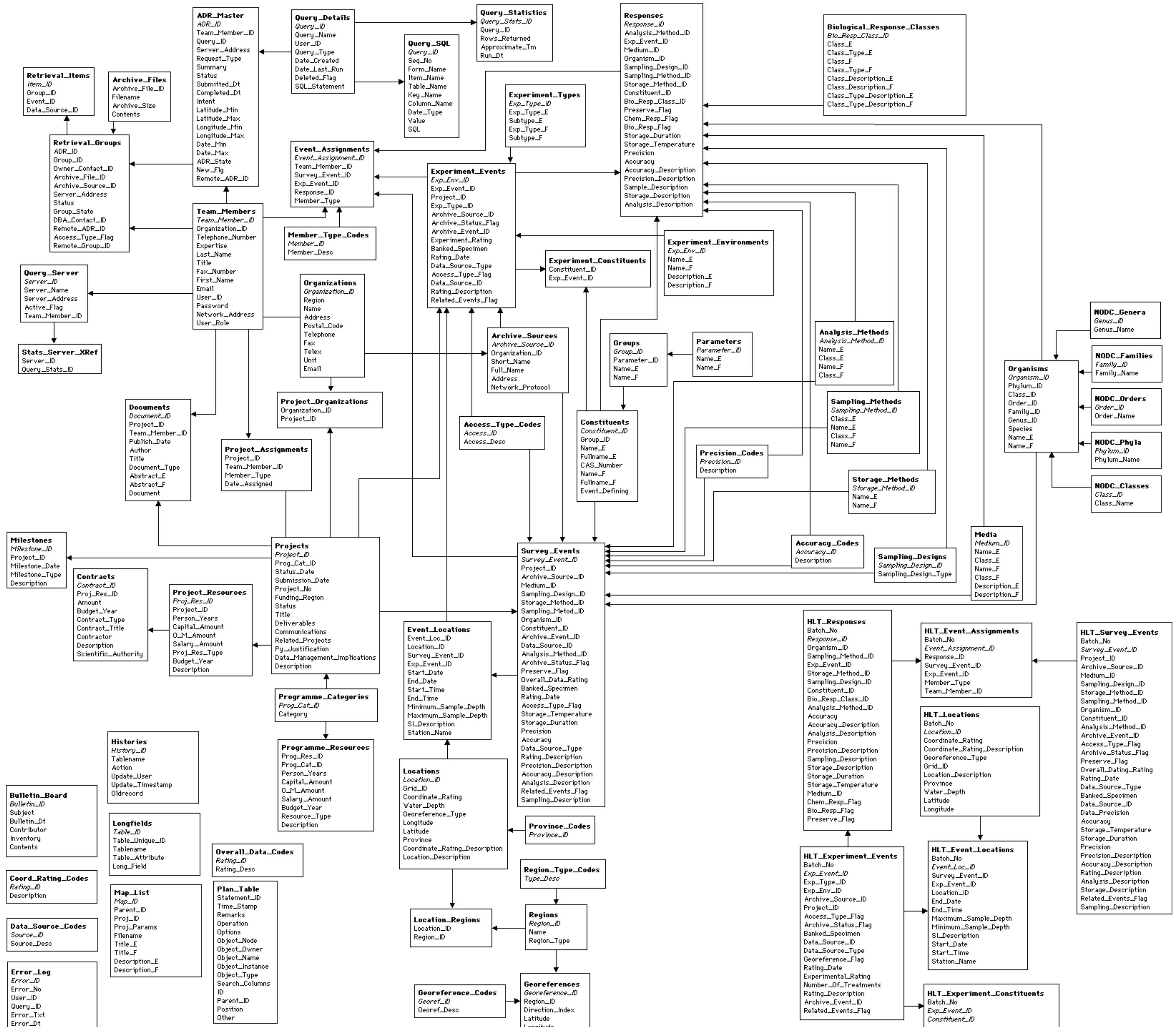
Incept5

SQL is just a language

- SQL is a language that allows you to manipulate relational databases - RDBs
- If the data you're storing is relational then SQL is a pretty good fit
- If fact, for dealing with lists (as tables) it's a great language, dynamic and relatively fast
  - Sure it has a few problems but give me a language that doesn't

Incept5

How complex can it be?



Incept5

NoSQL - No What?

- Did we really need a name for it?
  - We'd worked for years without needing a name for it

- Later No SQL became Not Only SQL
  - The vendors chickened out :-)
  - NoSQL vendors support NoSQL because they can't support SQL

- NOJ - No Java
- NOSS - No Shell Scripts
- NOW! - No Windows!

Incept5

SQL works

- SQL might be the wrong language for hierarchical data but it's superb for tables
- Imagine the telephone directory - in a single table
  - Tell me how many "Davies's"
  - select count(*) from directory where surname="Davies"
- Now in XPath (for example)
  - count(/directory[surname='Davies'])

Incept5

SQL is useful - when things are flat

- Easy, so now I'd like to see the 10 most popular names and order them by popularity...
  - Select surname,count(*) from directory group by surname order by count(*) desc limit 10

- If you had the list as a CSV then you could do the same with a little shell magic - assuming you're on a decent OS of course
  - cut -d "," -f 1 | sort | uniq -c | sort -r | head -10

- Both will execute, even on a million rows in around a second

Incept5

Java instead of SQL?

- A few years back we needed a cache for 1TB of data
  - Coherence was perfect so I called Cameron Purdy for a price
  - It would have been over $500k!
- So we used MySQL distributed over 20 machines with a query aggregator
  - Data was stored in a columnar format with everything indexed in memory
- It worked incredibly well but...
- Joins and aggregate functions took another year to write
  - avg, max, min, count etc.

Incept5

A question for you...

- I have 1 million rows of CSV (Comma-Separated-Values), each row roughly 1k in size and some 50 columns
  - So 1GB of data, 50 million fields

- I now want to count the number of rows where the 7th column contains the word "Think"

- How long SHOULD it take to return the answer?
  - A - Around 30 seconds to a minute
  - B - Around 20-30 seconds
  - C - Around 10-15 seconds
  - D - Sub 10 seconds

Java NIO

- Using ~~Sun's~~ Oracle's NIO Grep example it threw OOM with -Xms4G -Xmx4G
  - The 100m version took 1.2 seconds so let's estimate 1GB will take 12 seconds as it's a linear search

- The following however took under a second...

```
grep -c Think BigFile.csv | cut -d "," -f 7
```

- Java is not the silver bullet

ORM - OMG!

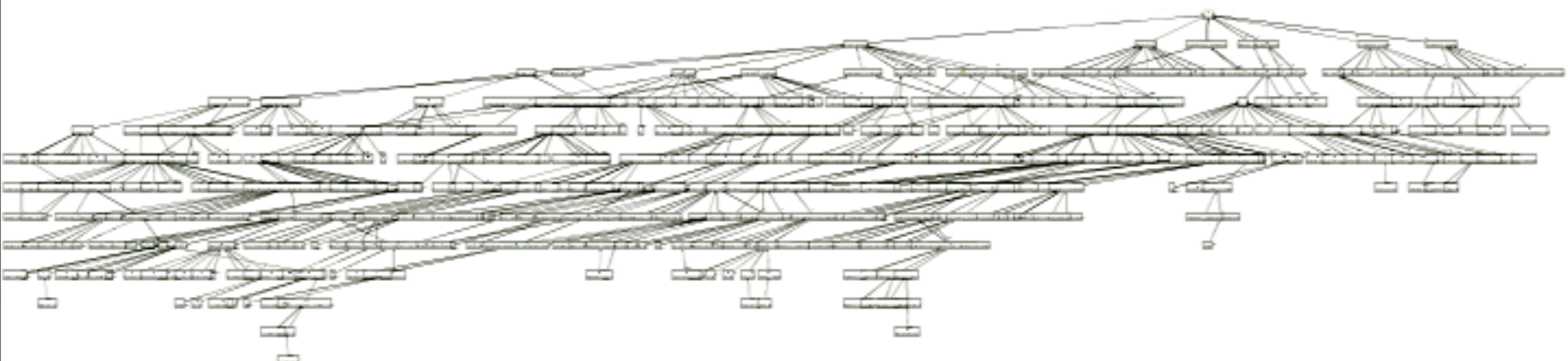
- Probably the biggest waste of programming time, lines of code and source of bugs and latency is ORM
- The effort of trying to convert something inherently hierarchical into something relational
- It's probably the main source of complaints about SQL
  - People start with a nice XML document, map it with ORM into a RDBMS and then end up writing custom SQL because the ORM layer is too slow
- GOOD LUCK!

Incept5

FpML into a classic database?

- The fuzzy patch below is the complete model of and FpML Swap from the IRD (Interest Rate Derivative) schema

- It's one of several dozen financial models in FpML



Incept5

FpML has to be persisted

- **This still needs to be stored...**

```xml
<?xml version="1.0" encoding="UTF-8"?><!--
  == Copyright (c) 2002-2007. All rights reserved.
  == Financial Products Markup Language is subject to the FpML public license.
  == A copy of this license is available at http://www.fpml.org/license/license.html
  -->
<FpML xmlns="http://www.fpml.org/2007/FpML-4-4" xmlns:fpml="http://www.fpml.org/2007/FpML-4-4" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" version="4-4" xsi:schemaLocation="http://www.fpml.org/2007/FpML-4-4 ../fpml-main-4-4.xsd http://www.w3.org/
2000/09/xmldsig# ../xmldsig-core-schema.xsd" xsi:type="DataDocument">
   <trade>
      <tradeHeader>
         <partyTradeIdentifier>
            <partyReference href="party1"/>
            <tradeId tradeIdScheme="http://www.chase.com/swaps/trade-id">TW9235</tradeId>
         </partyTradeIdentifier>
         <partyTradeIdentifier>
            <partyReference href="party2"/>
            <tradeId tradeIdScheme="http://www.barclays.com/swaps/trade-id">SW2000</tradeId>
         </partyTradeIdentifier>
         <tradeDate>1994-12-12</tradeDate>
      </tradeHeader>
      <swap><!-- Chase pays the floating rate every 6 months, based on 6M USD-LIBOR-BBA,
            on an ACT/360 basis -->
         <swapStream>
            <payerPartyReference href="party1"/>
            <receiverPartyReference href="party2"/>
            <calculationPeriodDates id="floatingCalcPeriodDates">
               <effectiveDate>
                  <unadjustedDate>1994-12-14Z</unadjustedDate>
                  <dateAdjustments>
                     <businessDayConvention>NONE</businessDayConvention>
                  </dateAdjustments>
               </effectiveDate>
               <terminationDate>
                  <unadjustedDate>1999-12-14Z</unadjustedDate>
                  <dateAdjustments>
                     <businessDayConvention>MODFOLLOWING</businessDayConvention>
                     <businessCenters id="primaryBusinessCenters">
                        <businessCenter>GBLO</businessCenter>
                        <businessCenter>JPTO</businessCenter>
                        <businessCenter>USNY</businessCenter>
                     </businessCenters>
                  </dateAdjustments>
               </terminationDate>
```

NoSQL?

- No RDBMS or NoSQL?
- We can still store the XML or hierarchical data in a RDBMS but SQL is pretty useless
- Now we need something like XQuery
- We have however found a good reason to move away from SQL

Incept5

Enough of SQL

OK, so SQL has its uses

Hierarchical data is not one of them

Let’s look at another problem...

Incept5

Lots of data

- In the banking world we have a lot of data
- Today 50-100,000 quotes a second isn't unusual
  - We recently hit 350,000/sec from just one source (CME)
- Writing this to a database is possible but not usually practical or necessary
  - Storing 100,000 objects is pretty simple, the data is relatively flat
  - But the problem is rarely so simple
- It gets more complex...

Incept5

Adding complexity

- 10,000 portfolios, each with 1,000 buy/sell orders at specific prices
  - For example one portfolio might contain someone's investment, partly held in Hong Kong equity, one of those equities might have a sell order (for the 500 shares) at HKD $85
- We now have 100,000 prices coming in every second and 10 million orders to watch
  - Technically 1 trillion (10^{12}) but there are optimisations that can be made
- Then repeat this across 20 exchanges
  - Oh yes and if you get a match we need sub milli-second triggers

Incept5

Time is critical

- I've simplified the use-case but you should get the idea
- In the world of trading only the first one gets the deal, there is no second place
  - This drives performance - mainly around latency
- While being first to have the order is what makes the money banks now have a "new" problem
  - To be honest it's not new, they're finally being forced to manage it
- Risk

Incept5

Risk management

- Especially after recent events it's critical that financial institutions monitor their exposure to risk
  - How much they owe or how much they are owed
  - Or how much they **might** owe etc.

- A 100 years ago teams, often hundreds of people, would calculate the figures every day
  - It could take days or weeks to know if you were bankrupt or rich

- Today we need to know by the second
  - Every single trade has a risk associated with it

Incept5

Risk

- Everything presents a risk...
  - The equity / asset could de-value
  - The seller could go under (bankrupt)
  - The currency of the asset could de-value
  - The political regime of the seller could change
  - The political regime or currency of the parent bank could change
  - The broker or counter-party who brokered the deal could go under
- All these may appear small risks but they are very real
  - Remember Sub-prime, Enron, Northern Rock?
- It's like a Tokyo or San Francisco earthquake, a very small chance it will happen tomorrow but it will happen one day

Incept5

Lots of data, lots of calculations

- There are two main flavours of distributed computing
  - Data
  - Computation

- Often they are closely related but not always

- To achieve either we usually need lots of memory and CPUs

- We don’t stack them or put them in clusters these days, we distribute them
  - Usually in a rack - but you don’t need to know that

We need to scale

Huge amounts of data
Vast amounts of computations

We need scaleability

Incept5

Distributed Computing - Ideal case

- In an ideal world we code without having to know about the deployment architecture
  - We assume a machine powerful enough and with enough memory to perform our task(s)
  - For a long time this was the case
- Sadly this doesn't usually work so we need to code with a view to scaling, scaling both memory and CPU power
- We code for a distributed environment, we hand out tasks to be distributed and data through and API to be stored

Change the programming model

- Probably the hardest thing to drill into programmers and worse, their managers, is to program for a distributed architecture
  - It all seems like an overhead at first
  - Abstraction of location
  - Storage though APIs
- The EJB model was a start but scalability was limited to the server or cluster of servers
  - JNDI lookup
  - Object life-cycle manage by the container
  - Spring extended this

Virtualisation

- If we just had 4 machines we might be mistaken into thinking it's a cluster
- Ideally we need to imagine an infinite amount
  - We code it, someone else pays for and adds computing resources
- Today's machines are a little large, we need finer granularity
  - This is where Virtualisation comes in
- We can split a large server into half a dozen smaller processing units (sometimes more)

Virtualisation is good

- Each virtual machine is independent from the other
  - Almost - VMs on the same physical machine rely on the same hardware
  - This needs to be understood by the provisioning software
- In many cases the Java VM is good enough but today's OS VMs usually leave us with more control
  - The Java VM runs on top of the OS Virtual Machine - usually Linux
  - We still rely on a lot of the OS - logging, network etc.
- Usually the same VM is as easy to run locally as remotely
  - This makes it easy to configure, code, test and deploy

Incept5

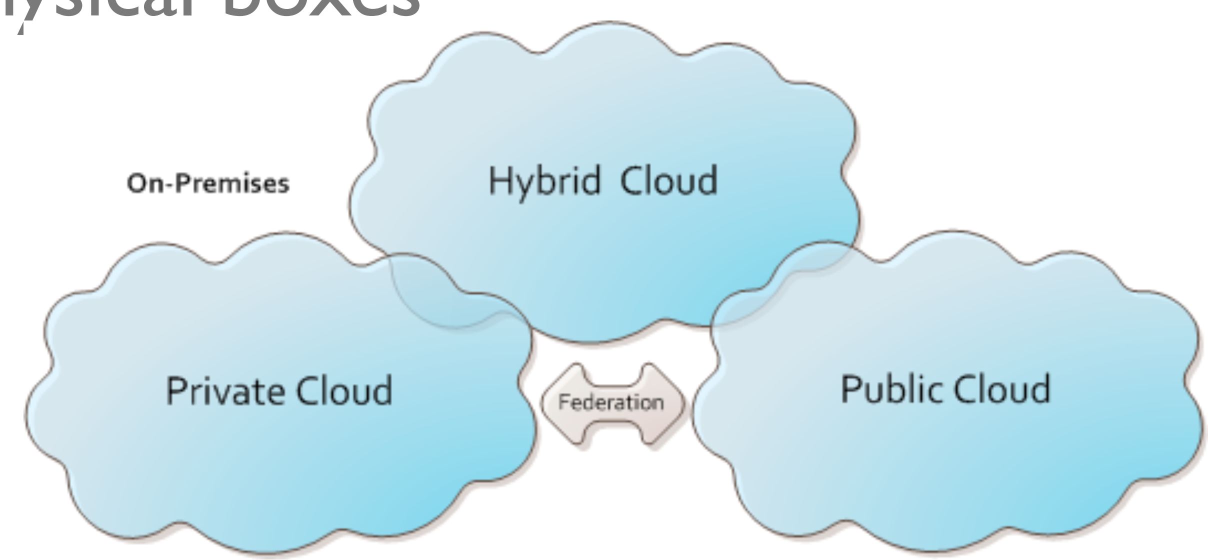
Local vs Grid vs Cloud

- If we can distribute to local VMs we're most of the way there
- Move the VMs to other machines on the network and we have "Grid"
  - Also known today as "private cloud", these can be physically local or remote but usually on your network or VPN
  - Today's investment banks and hedge-funds have anything from 200 to 20,000 CPUs in their grids
- Use someone else's hardware and you have "Cloud"
  - Amazon's EC2 is a perfect example

Incept5

Local vs Grid vs Cloud

- Local
  - Very fast but limited by the physical size/power of your box
  - Perfect for developing and testing
- Private Cloud / Grid
  - Very secure - perfect for banks & governments
  - Very scalable but slight latency
  - Costly - as you have to buy the physical boxes
- Cloud
  - Pay for what you eat
  - Extremely scalable
  - Latency and security can be an issue



Incept5

Grid technologies

- Let's look at some of the Grid technologies, there are dozens but we'll take a slightly closer look at a few...
  - GemFire
  - Terracotta (BigMemory)
  - GigaSpaces
  - Coherence
  - Neo4J

- Many other technologies overlap in areas, predominately the caching side, these too are viable alternatives
  - EHCache, Memcached, JCache (JSR-107) etc.

- It would be wrong not to mention NoSQL DBs...

The list goes on...

- Many of the following are appearing on the scenes
  - MongoDB
  - HBase
  - Cassandra
  - Riak
  - CouchDB
  - Redis

- MongoDB is pretty popular
  - HBase with Hadoop and Cassandra occasionally too
  - Others I've not seen but that doesn't mean they're not being used, many of them have extremely powerful features with considering

How to compare?

- I could write a book and talk to you for days
  - But I still can not tell you which is best
- I've seen a lot of money spent on comparing them
  - Each one of them will give you examples where they've excelled
- Most of them will do the job and most of them will tell you bad things about the others
- All I can do is point out a few major differences
  - Anything claim I make would be disproven as things evolve

Incept5

Grid technologies

- GemFire
  - Originally an OOD, now has a pure Java implementation
  - Recently acquired by VMWare
- Terracotta
  - Uses Java VM replication,
  - Recently acquired by Software AG
- GigaSpaces
  - Originally the only viable implementation of Sun's JavaSpaces
- Coherene
  - Formally "Tangosol", now owned by Oracle
- Neo4J
  - The wild-card, a graph database

Incept5

GemFire

- Connect to the distributed system and get the Cache ref

```
// Create / Find a cache (using the map interface)
DistributedSystem ds = DistributedSystem.connect();

// Get the Singleton instance of the cache
Cache cache =  CacheFactory.create(system);
```

- Instantiate your object and simply put into a Map

```
// Create / Find Data Region "Prices" in the cache
Map prices = (Map ) cache.getRegion("Prices");

// Write the Price to the cache...
prices.put( price.getKey(), price );
```

- As you can see this couldn't be easier

Incept5

Reading a Price from GemFire

```
// Get Access to the Data Region "Prices" and cast as a java.util.Map
Map map = (Map)cache.getRegion("Prices");

// Retrieve the latest spot price for GBP/NOK
Price myPrice = (Price ) map.get("GBP/NOK-SPOT" );
```

- All GemFire Data Regions are indexes on the key used in Put

```
// Get Access to the Data Region "Prices"
Region prices = cache.getRegion("Prices");

// If the retrieval is not based on primary key, you can use OQL
// Retrieve the latest spot price for GBP/NOK
SelectResults results = prices.query("getKey() = 'GBP/NOK-SPOT'");
for (Iterator iter = results.iterator(); iter.hasNext(); ){
    Price myPrice = (Price) iter.next();
}
```

- All GemFire Data Regions can be indexed on fields or and/or methods

Incept5

Spring work equally well

- With most of these you can use Spring Integration to insert complex data into our Grids...

```
< gfe:cache/>
 <gfe:replicated-region id="swift-etc">
        <gfe:cache-listener>
            <bean class="biz.c24.io.swift.etc.data.CacheListener"/>
        </gfe:cache-listener>
 </gfe:replicated-region>

 <file:inbound-channel-adapter
        id="filesIn" directory="file:/Users/jdavies/dev/Spring_C24/spring-integration-samples/input"
        filename-pattern="*.txt">
      <int:poller id="poller" fixed-delay="0"/>
  </file:inbound-channel-adapter>

 <int-gfe:outbound-channel-adapter id="channel2" region="swift-etc">
     <int-gfe:cache-entries>
        <entry key="payload.Block4.SeqA.Field20aReference.C.Reference" value="payload"/>
     </int-gfe:cache-entries>
 </int-gfe:outbound-channel-adapter>
```

Incept5

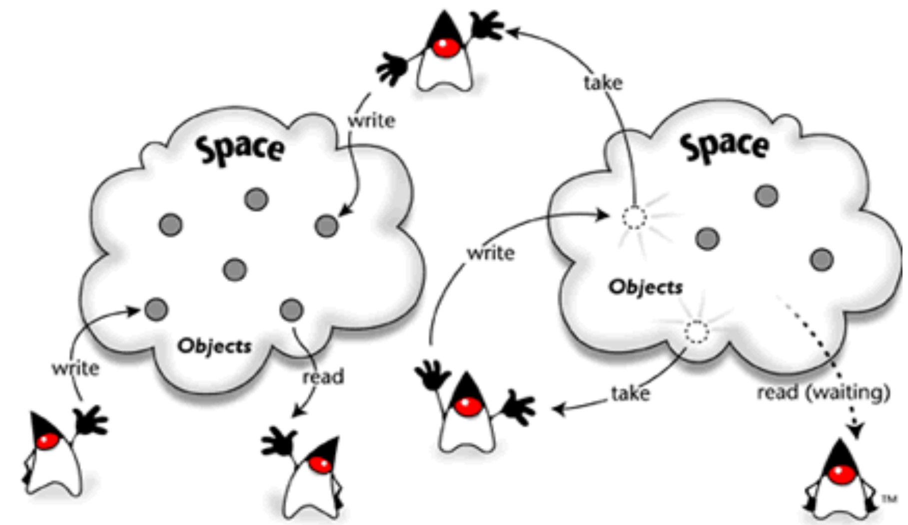
GigaSpaces

- GigaSpaces is the perfect implementation of a Master/Worker pattern
  - But they don't really push this sadly

```
public String url = "jini://*/*/C24Space";
space = new GigaSpaceConfigurer(new UrlSpaceConfigurer(url)).clustered(true).gigaSpace();

// CacheItem is our own POJO to host an ID & SWIFT POJO from C24's Integration Objects
ci = new CacheItem();
MT513Element mt513Element = new MT513Element();
MT513Message mt513 = ci.parseMT513FromString(rawSwiftMT513);
id = extractID(mt513);

ci.setId(id);
ci.setMessageData(mt513);
space.write(ci);
```



- Once again really easy to use
  - Notice the interface is not a Map here
  - In "classic" JavaSpaces we use a template to retrieve data
  - But GigaSpaces have added new search APIs now

Incept5

Finally my "wild-card"

- Neo4J is interesting but I've yet to find a problem where it's the obvious solution
  - If I had to re-implement Twitter or Facebook I'd use Neo4J
- I've come across a few trading systems that would benefit from the graph-database traversal
  - However the graphs were not deep so the use-case was not obvious
- It is however pretty cool and very fast
  - I encourage you to take a look as it just need someone with a fresh mind to come up with the "killer use-case" other than social groups

Incept5

Neo4J

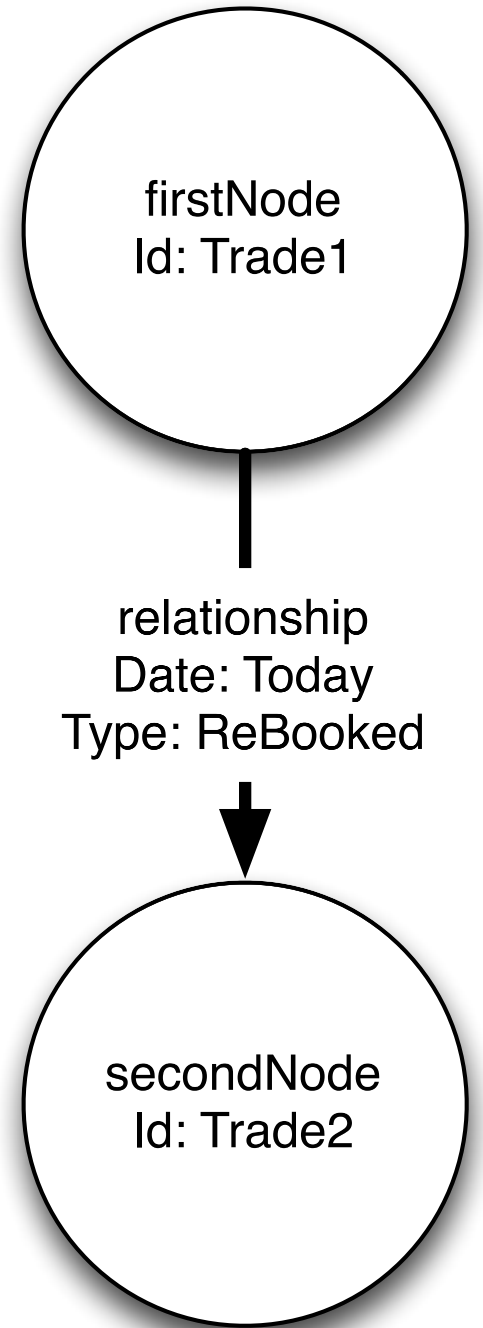
- Some code snippet for Neo4J...

```
private static enum RelType implements RelationshipType {
    ReBooked
}

GraphDatabaseService graphDb = new EmbeddedGraphDatabase( "./" );
registerShutdownHook( graphDb );

Node firstNode;
Node secondNode;
Relationship relationship;

Transaction tx = graphDb.beginTx();
try {
    firstNode = graphDb.createNode();
    firstNode.setProperty( Trade1.getId(), Trade1 );
    secondNode = graphDb.createNode();
    secondNode.setProperty( Trade2.getId(), Trade2 );
    relationship = firstNode.createRelationshipTo( secondNode, RelType.ReBooked );
    relationship.setProperty( “Date”, new java.lang.Date() );
    tx.success();
}
finally {
    tx.finish();
}
```



Grids & Complex Data

- Going back to the FpML
  - And adding ISO-20022, Fix, SWIFT, etc. into the mix
- We're extracting data from the model (usually with XPath) and using it as an index (or indices) into the POJO in the grid
- At one large broker (for example) we're storing Fix messages from dozens of exchanges into distributed memory
  - Over 10,000 per node per second
- Parse the message, read the key data and insert into the grid as key-value pair

It's question time...

Incept5

Thank you

- We are looking for talented Spring/Java programmers and architects in London, Chicago, New York & San Francisco

- John.Davies@Incept5.com
- Twitter: @jtdavies