ORACLE | TANGOSOL

JavaOne

The Top 10 Ways to Botch Enterprise Java Technology-Based Application Scalability and Reliability

Cameron Purdy
CEO

Tangosol
http://www.tangosol.com/

TS-4249

Finding Meaning in This Presentation

Laugh, cry, or go to sleep… but please turn off your phones

It's never been that funny to watch your app go down, and thanks to all your users it's the slowest site in town

Disclaimer

Common Sense Trumps Dogma

Take everything with a grain of salt

- There will be real world situations in which the principles from this presentation will be wrong; Always use common sense
- Any similarity of material in this presentation to disasters that you have witnessed, either real or imagined, is purely coincidental

#10

"The Internet is the most important single development in the history of human communication since the invention of call waiting."

—Dave Barry

Optimize performance assuming that it will translate to scalability

Performance and Scalability

Never as simple as it seems…

- Quick definition
  - Performance: “Wall Clock” time to completion
  - Scalability: The ability to add resources to increase the capacity of a system
- Goals: Why care?
  - Performance: Make the single-user scenario faster
    - Eliminate algorithmic inefficiency
    - Focus on expensive queries and high-latency operations
  - Scalability: Support many users
    - Quantify resources required for growth

Raw Performance vs. Scalability

Are you optimizing for two servers, or two hundred?

- Optimize performance by 10%
  - 10% improvement regardless of the number of servers
- Increase the scaling factor by 10% (0.90 to 0.99)
  - 2 servers: 5% improvement
  - 4 servers: 15% improvement
  - 8 server: 36% improvement
  - 16 servers: 82% improvement
- This doesn't even take system bottlenecks into consideration
  - Potential orders of magnitude improvement!

#1

"Technological progress has merely provided us with more efficient means for going backwards."

—Aldous Huxley

Ignore the potential impact of performance on scalability (and vice-versa)

Skewed Access Patterns

Or, “How to rig a benchmark…”

- Access patterns are not uniform!
  - Most benchmarks tend towards uniform access
  - …so products will show linear scalability
  - …that your applications will never see
- Skewing creates “Hot Spots”
  - Great for read-mostly cache-intensive applications
  - Horrible for write-intensive applications

Impact of Performance on Scalability

It's all about the Hot Spots!

- In write-intensive applications, data consistency requirements result in serial execution
- Serial execution creates data "Hot Spots"
- Due to Hot Spots, high-end scale is limited by serial performance

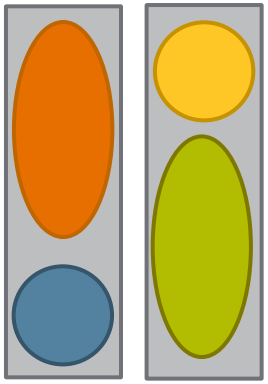
Example: Foreign Exchange

For those of you who have abandoned the gold standard

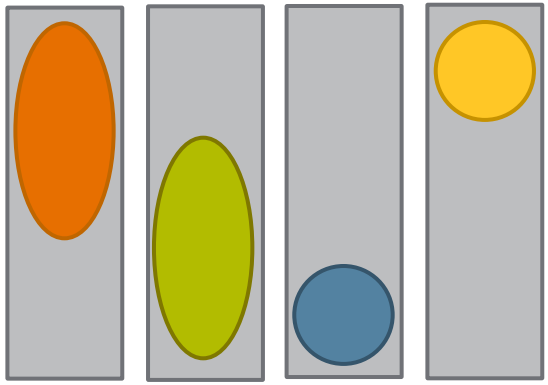
- USD-EUR trading is 40% of the market
- Updates to that instrument are inherently serial
- Dedicate an entire server to USD-EUR
  - If one server can handle 40% of the market
  - …then another 1.5 servers can handle the other 60%
- No theoretical increase in matching capacity is possible beyond three servers

Example: Serial Bottlenecks

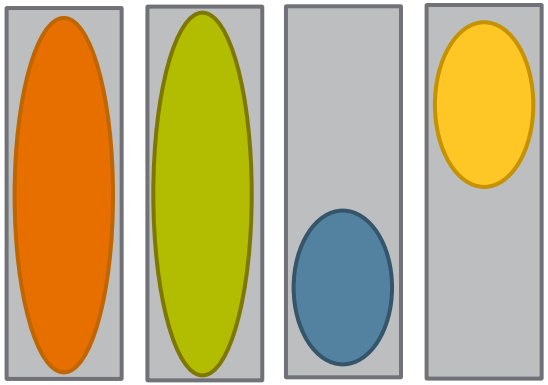
Linear scale to two servers



Same load on four servers



Load can only grow by 60% before bottlenecking



Beyond three, additional servers add **no** capacity!

Impact of Scalability on Performance

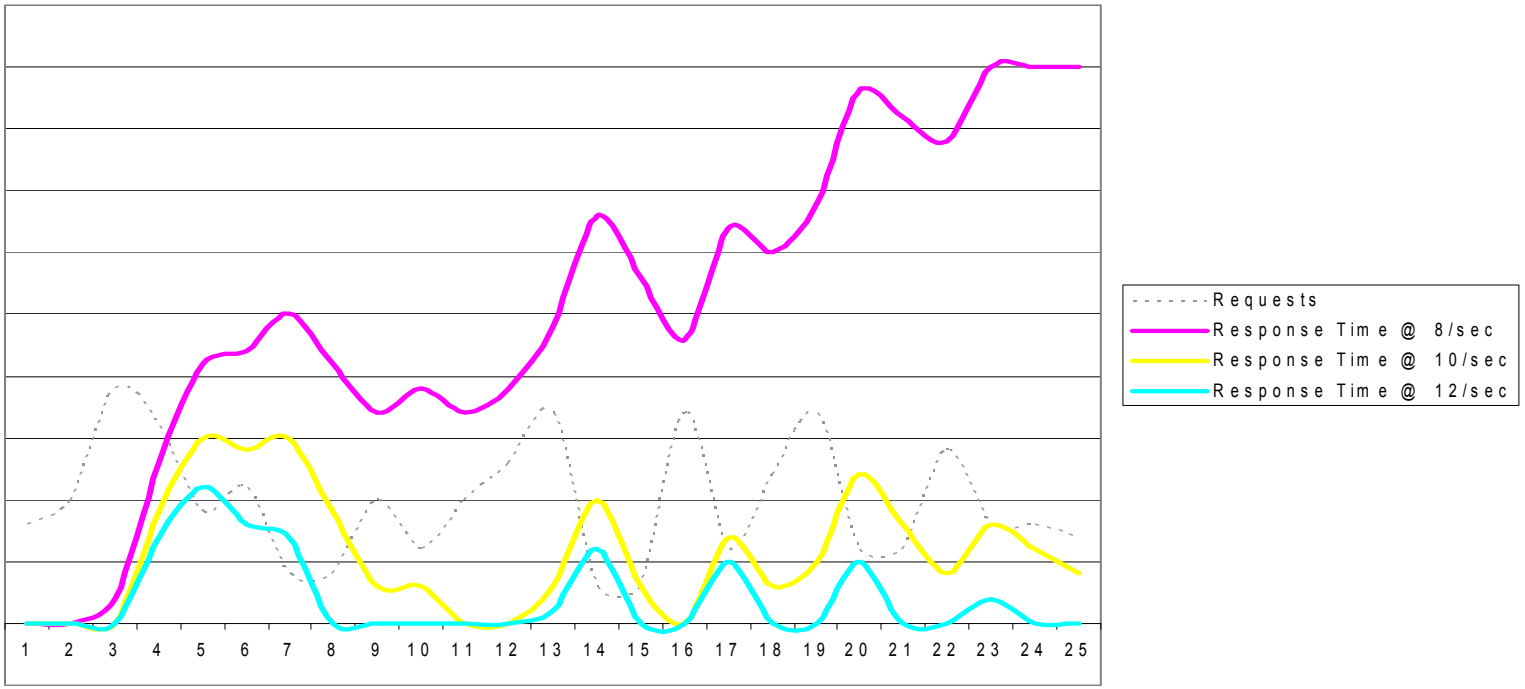
Queue Theory, or: “When it rains, it pours…”

- Inability to scale sufficiently will kill performance
  - Service overloads increase request queue depth
  - Even minor overloads can bring massive wait times
- Service queues are just like credit cards
  - Paid off regularly: Empty or near empty queues
    - When a service keeps pace, the queue never backs up
  - In debt: Deep and getting deeper
    - If the service loses pace, the queue can expand indefinitely

Example: Response Times

Does it remind you of online shopping at Christmas?

- Client averages 10 requests per second
- Service supports 8, 10 or 12 requests per second



"I think computer viruses should count as life. I think it says something about human nature that the only form of life we have created so far is purely destructive. We've created life in our own image."

—Stephen Hawking

Assume you are smarter than the infrastructure

Pet Shop Inventory

Fish are friends, not food

```
public class PetShopInventory {
    public synchronized boolean trade(Pet oldPet, Pet newPet) {
        if (m_inv.contains(newPet)) {
            m_inv.remove(newPet);
            m_inv.add(oldPet);
            return true;
        }
        return false;
    }

    public boolean buy(Pet pet) {
        return m_inv.remove(pet);
    }

    private final List m_inv =
        Collections.synchronizedList(new ArrayList());
}
```

Inspired by http://www.javaconcurrencyinpractice.com/listings/ListHelpers.java (Brian Goetz and Tim Peierls)

Threading Violations

Shared Mutable State

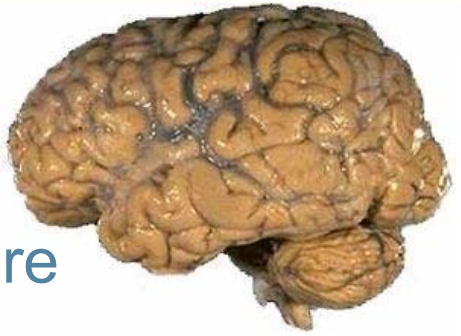
- Who **hasn't** done this?
  - Singletons—e.g. product inventory
  - Hashtable caches
- Potential Problems
  - Multi-threaded programming is hard… to do well
  - Both data corruption and visibility problems can occur
  - Thread pool depletion from blocking operations
  - Opportunity for thread-level deadlock
- The container may be able to help
  - Java™ Specification Request (JSR) 236/237: Timer and Work Manager APIs

Resource Violations

Keep your hands where I can see them

- File systems
  - Putting a Java Cryptography Architecture (JCA) adapter on a File[1] does not make it a transactional resource!
- XA “impostors”
  - Implement the interface… but not the behaviour!
- Managing your own sockets

1. Inspired by http://www.theserverside.com/news/thread.tss?thread_id=33341



Reinvent the Wheel

Other ways to be smarter than the infrastructure

- Replace built-in functionality
  - "We had to build our own connection pool…"
  - "Java Message Service (JMS) is too heavy-weight, so we…"
- Reinvent available frameworks and libraries
  - "Anyone can build a better web framework than Struts"
  - "Hibernate just didn't support our domain model…"
  - "I built my own Dependency Injection framework because Spring is too bloated"

#9

"Everybody gets so much information all day long that they lose their common sense."

—Gertrude Stein

Follow the rules blindly

An Application Server Is Not Magic

When it's your application running in your environment

- The implications of breaking the rules
  - Working against the application server, e.g. security
  - Additional responsibilities may not be obvious
- The benefits of being able to break the rules
  - Some impossible things become possible
  - Some difficult things become simple
- Example: FTP
  - FTP is still in common use for application integration
  - Sending files via FTP can require socket I/O
  - Receiving FTP'd files can require file I/O

#8

"Men have become the tools of their tools."

—Henry David Thoreau

Abuse the database

Abuse the Database

When the only tool you have is a hammer…

- Pro: Information consistency and durability
- Con: Scale-out is limited and costly
- Avoid high volume, low value tasks
  - Request State, Logging
  - Conversational State (HTTP sessions, etc)
  - “File system” responsibilities, non-transactional state
- Rule of Thumb: Persistent **and** Transactional

#8

Avoid the database

"Getting information off the Internet is like taking a drink from a fire hydrant."

—Mitchell Kapor

Avoiding the Database Altogether

Darling, I don't know why I go to extremes

- Sessions, Stateful Session Beans are insufficient
  - Can provide scalability and availability
  - Limited options for durability
  - Lacks information management and reporting
- Java object model
  - Does not support composable transactions
  - Persistence tends to be inefficient and brittle
  - Lacks information management and reporting
- Persistent transactional state: Use a database!

#7

"I must have a prodigious quantity of mind; it takes me as much as a week sometimes to make it up."

—Mark Twain

Introduce Single Points of Bottleneck

Single Points of Bottleneck

- A Single Points Of Bottleneck (SPOB) is any server, service, etc. that all (or many) requests have to go through, and that has any load-associated latency
- The simplest way to create a SPOB is to depend on a shared resource, such as a database

Single Points of Bottleneck

- Internal bottlenecks include connection pools, singletons, mutable state shared across threads
- External bottlenecks include databases, message queues, web services, mainframes, and other enterprise applications

#7

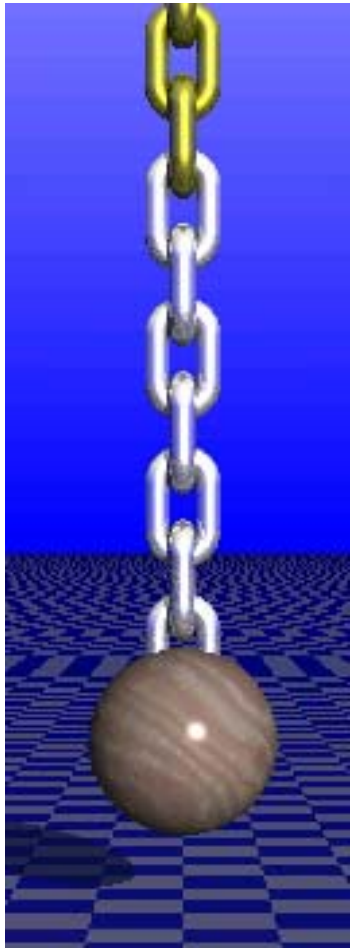
"If builders built buildings the way programmers wrote programs, the first woodpecker that came along would destroy civilization."

—Anonymous

Introduce Single Points of Failure

Single Points of Failure

- A Single Point Of Failure (SPOF) is any single device, server, process, etc. that must be available in order for the application to operate
- Designing for high availability involves eliminating (by redundancy, etc.) as many Single Points Of Failure as possible



Single Points of Failure

- Enterprise Java Technology-Based Applications can eliminate most middle-tier SPOFs with a combination of application server features and careful design
  - Use the “cookie cutter” responsibility model

#6

Abuse abstractions

"There's no problem that can't be solved with yet another level of indirection."

—Unknown

Not Everything Needs to Be Abstracted

Blinded by elegance

- Abstraction can hide real costs
  - I/O: ORMs and the N+1 problem
  - Memory: ORMs and large sets
  - UI: Component rendering
  - Conversational State: Sessions
- Abstraction can make simple things impossible
  - ORMs that don't support stored procedures
  - Web frameworks that don't support all HTML options
  - Try building a legal Enterprise Java technology-based "singleton"

#6

"Real programmers can write assembly code in any language."

—Larry Wall, inventor of Perl

Avoid abstractions

Abstractions Are a Valuable Tool

Wisely chosen, most abstractions cost almost nothing

- Runtime cost of programming abstractions is close to zero
  - Inheritance and method invocation
- Abstractions layer applications, allowing for orthogonal concerns to be introduced
  - Intelligent caching
  - Security, auditing
- Simplify both programming and maintenance

#5

"It is only when they go wrong that machines remind you how powerful they are."

—Clive James

Assume DR can be added when it becomes necessary

Why WAN?

Because having to manage one site isn't enough

- Usually for Disaster Recovery (DR)
  - Inherent compromises
- Globally distributed applications
  - Multiple Data Centers, one global application
  - Generally regional division of information
- Multi-site access to "real time" data
  - Pull (on demand)
  - Push (on subscription)
  - Hierarchical distributed caching
  - Continuous Query

Disaster Recovery

Job security is # 1

- Requirements
  - No measurable impact on performance
  - All data centers should be active (“hot-hot”)
  - Processing continues uninterrupted if the WAN fails
  - Consistent information without any update conflicts
- Reality
  - MAN connections can be high bandwidth and fast
  - Legal requirements push DR out further
  - WAN roundtrips can be up to 1 second
  - Interruption of connectivity is terribly disruptive

Information Availability and Consistency

How the backup is maintained

- Synchronous updates are considered “safe”
  - May add seconds per transaction
- Asynchronous updates are fast
  - No real latency penalty
  - May allow update conflicts in hot/hot scenarios
  - Window for data loss from site failure
    - Actual loss may not be measurable or determinable
- Log shipping
  - Transactional logs for recoverability on DR site

Common DR for HA Databases, Queues

How the backup is maintained

- Log synchronization
  - Network bandwidth must be greater than the log generation rate
- Log shipping
  - Transactional logs are maintained on a shared file system for recoverability on DR site
- Database or Storage Replication

#4

"One machine can do the work of fifty ordinary men. No machine can do the work of one extraordinary man."

—Elbert Hubbard

Use a one-size-fits-all architecture

One Size Fits All

When dogma defeats logic

- Availability menu
  - Business hours
  - 24x7 “five nines”
  - Continuous Availability
- Reliability menu
  - Yesterday’s data
  - Best effort
  - Real Time
- Scalability menu
  - One server is plenty
  - Small cluster
  - Make it like Google
- Performance menu
  - Whenever
  - 7 seconds (or they leave)
  - 99% under 1 second

"Computers shouldn't be unusable. You don't need to know how to work a telephone switch to make a phone call, or how to use the Hoover Dam to take a shower, or how to work a nuclear-power plant to turn on the lights."

—Scott McNealy

Use big JVM™ machine heaps

The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java™ platform.

Big JVM Machine Heap = Big Pause

In theory, there is no difference between theory and practice

- In the lab
  - Full GC of **10GB** JVM machine heap takes a few seconds
- In production
  - Full GC of **1GB** JVM machine heap takes 30 seconds
- Why?
  - Not all customers are running on JDK™ 1.6
  - Differing complexities of object graphs
  - Use a finalizer? It can't be young-generation collected
  - To be honest, we don't always know

"If builders built buildings the way programmers wrote programs, the first woodpecker that came along would destroy civilization."

—Anonymous

Assume the network works

Network Infrastructure

The failure modes are unpredictable

- First fallacy of distributed computing

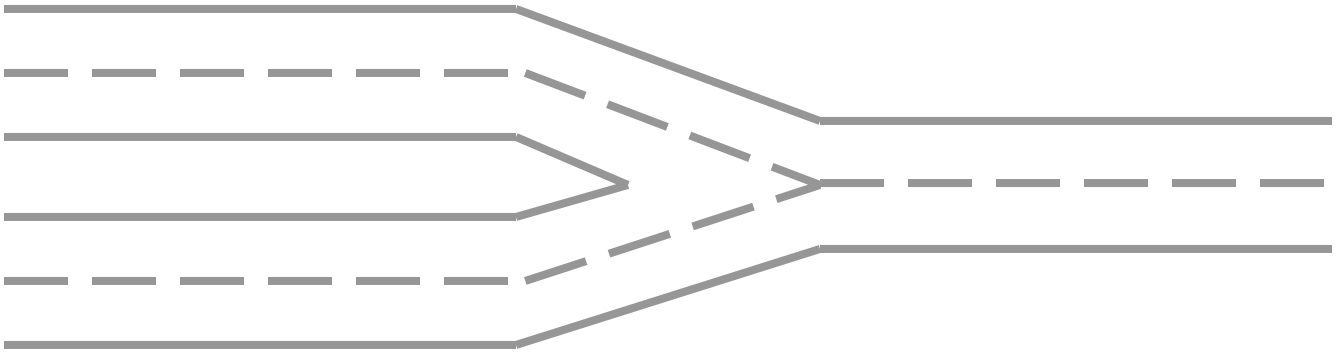
  "The network is reliable"

- Distributed environments are no more reliable than their weakest links
  - Redundant switches, cabling and NICs
  - HA load-balancers, DNS, databases, message servers

The Bottleneck Isn't Where You Think

When gigabit isn't gigabit…

- When two two-lane highways merge into one, the net result (with heavy traffic) is that the two-lane highways are no better than one

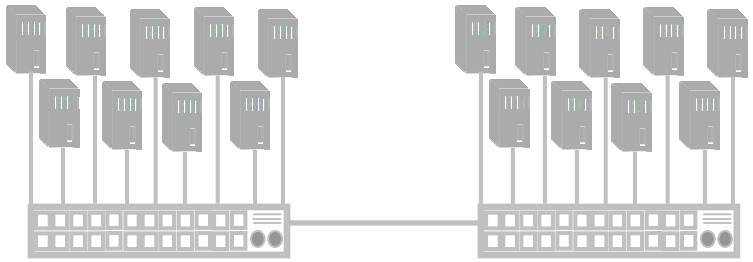


- Servers using a shared bus face an equivalent challenge: Each server can only use a portion of its full theoretical bandwidth

The Bottleneck Isn't Where You Think

When gigabit isn't gigabit…

- If a highway connects two cities, one hopes that the residents don't commute each to the other

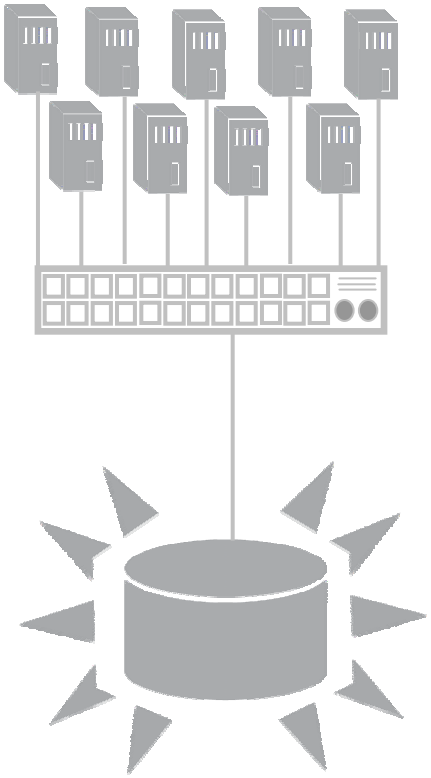


- Blade frames face a similar challenge
  - Fully switched-fabric gigabit backplanes
  - Higher bandwidth frame-to-frame, e.g. 4-gigabit
  - Effective max bandwidth across frames: 0.25 gigabit

The Bottleneck Isn't Where You Think

When gigabit isn't gigabit…

- Everyone gets off at the shopping mall exit
- Shared resource connected by a single pipe
- Database connections
  - Server farms
  - Enterprise Java Technology-Based Application clusters
  - Compute Grids



"I think complexity is mostly sort of crummy stuff that is there because it's too expensive to change the interface."

—Jaron Lanier

Avoid proprietary features

Proprietary Features

Competitive Advantage or Lock-In?

- Database
- “Non-Standard” Frameworks
- Standards
  - Choice
  - Portability
- Proprietary
  - Best of Breed
  - “Get it Done”

#1

"The most important and urgent problems of the technology of today are [..] the reparations of the evils and damages by the technology of yesterday."

—Dennis Gabor

Believe product claims

What Products Don't Do Everything?

- Linearly Scalable
  - …if you re-write your application from scratch
- Fault Tolerant
  - …minus a timeout period and a bit of data corruption
- Transactional
  - …as long as nothing fails
- Test **anything** that really matters to you!

Summary

Conclusions

I sat through the whole thing and all I got was this stupid slide

- Achieving success is often the result of making informed engineering trade-offs
  - Understand the **actual** requirements early
- Use common sense
  - Don’t do good things just because you think they are good things
  - Don’t do bad things unless they happen to be good things in your environment

Q&A

More Information

www.tangosol.com

ORACLE | TANGOSOL

JavaOne

The Top 10 Ways to Botch Enterprise Java Technology-Based Application Scalability and Reliability

Cameron Purdy
CEO

Tangosol
http://www.tangosol.com/

TS-4249