

Presentation to
WWISA

Architecting for Performance:
A Collection of War Stories

Richard Leeke
Senior Consultant
Equinox Limited

Agenda

- ❑ Architectural Requirements – Balancing the Trade-offs
- ❑ The Importance of Workload Modelling
- ❑ Allow For (Performance) Testability
- ❑ Deploying Data Access Components – Which Tier?
- ❑ Performance Management - an Architectural Issue
- ❑ Summary
- ❑ Questions and Discussion



Architectural Requirements
—
Balancing the Trade-offs

Be Clear about the Priorities

Various System Qualities count against Performance

- Flexibility – e.g. *everything* table-driven
- Portability – e.g. no use of DBMS features
- Ease of development (versus runtime efficiency)
- Scalability – generally comes at a cost
- Etc...

Portability

Explore how important portability *really* is

- For a package vendor it's probably crucial
- For a one-off system it may just be hedging the bets

Example - database independence

- Architecture may ban DBMS specific features ...
- ... but stored procedures and triggers are much faster than client-side or middle-tier logic
- Consider encapsulating DBMS specific features
- *True* DBMS independence is *hard* – means developing and testing against multiple platforms

Scalability

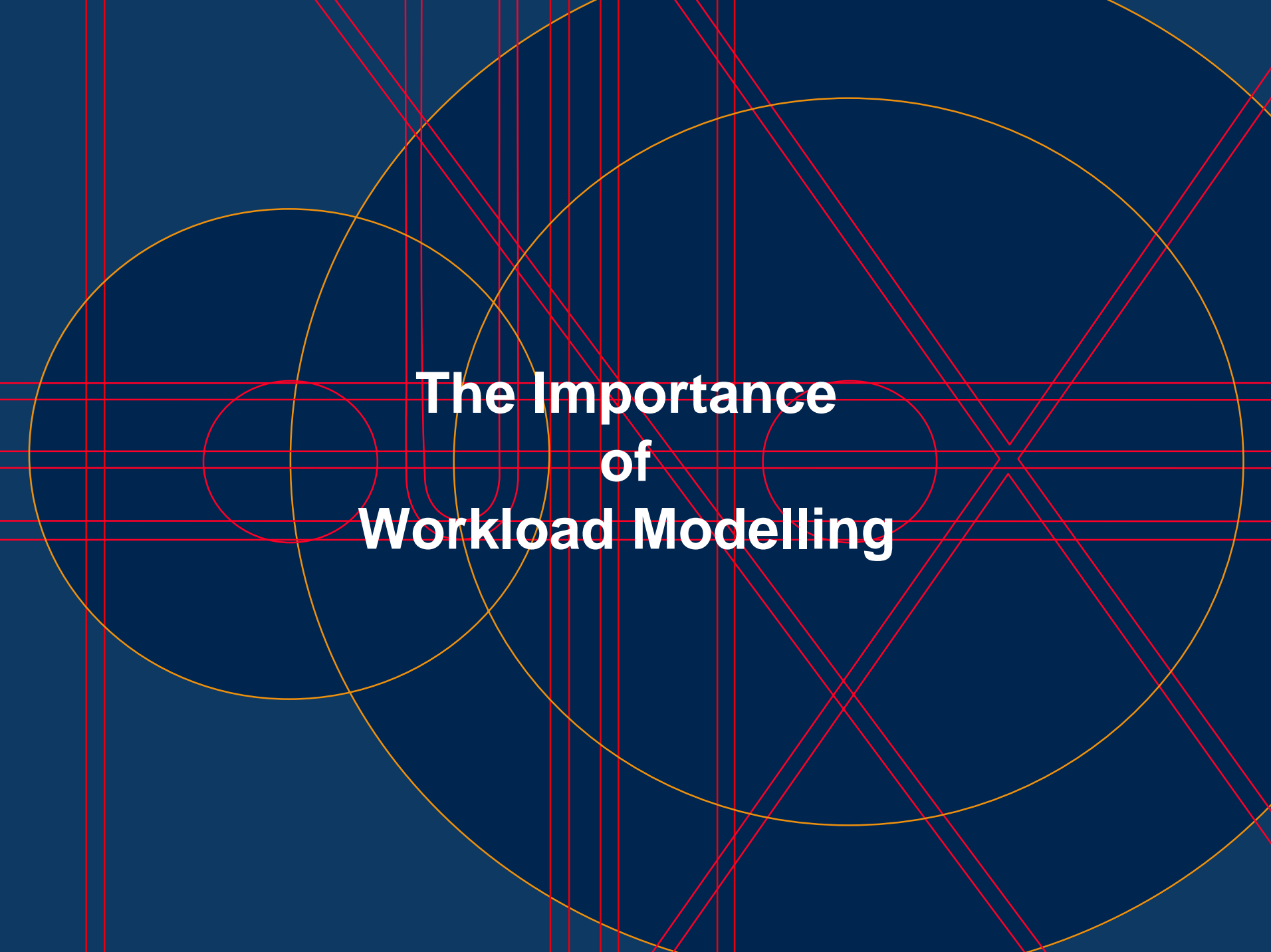
Some Definitions

=> Scalable

“You’ll need to spend more money on the hardware”

=> Highly Scalable

“You’ll need to spend MUCH more money on the hardware”



The Importance of Workload Modelling

Understanding the Workload

Get the basic business numbers (often not easy)

- Data volumes
- Transaction volumes
- Total and concurrently *active* users
- etc

Derive *estimates* of system metrics to predict behaviour

- Rate of database operations
- Network messages & data volumes
- Object creation/destruction events
- Or whatever is relevant...

Workload model is vital for performance testing

Case Study 1)

Insurance Data Capture System

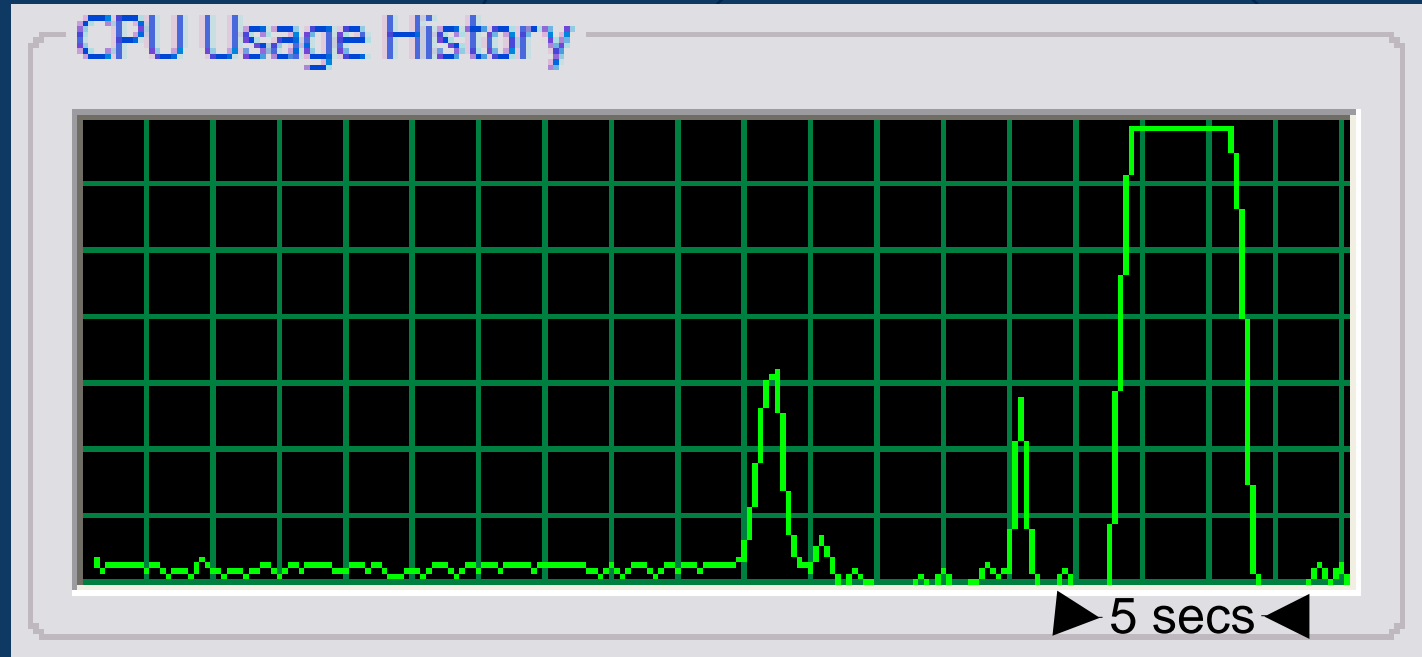
Requirements

- Capture of insurance details
- 200 concurrent users
- 3 minutes per business transaction per user
- => ~ 1 transaction per second

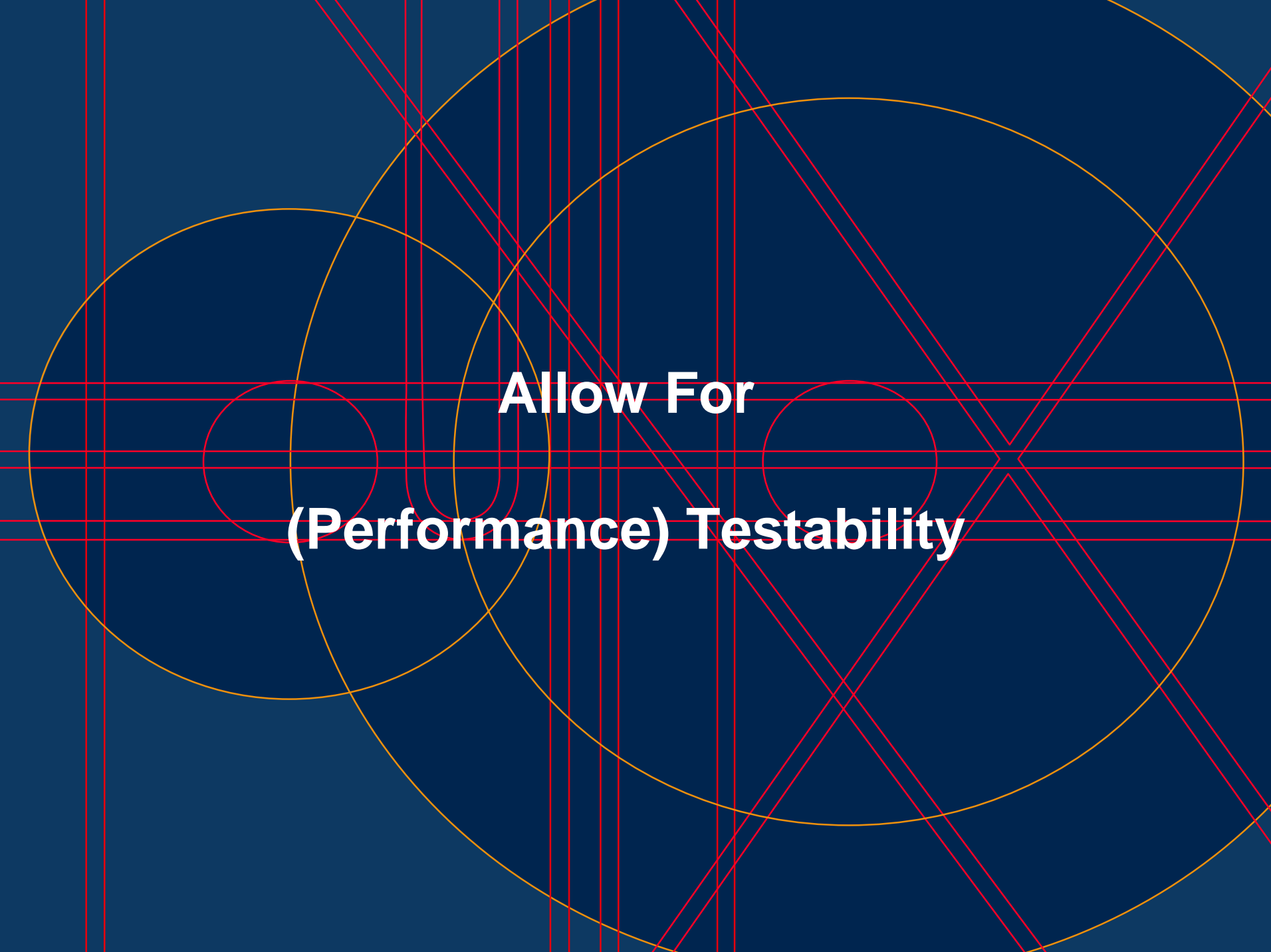
Solution

- Data entry via browser - HTML forms
- Apache web server and Java based application server
- Sybase database server

Application Server CPU Usage



- 1 transaction requires 5 CPU seconds
- Dual CPU server
- => Maximum throughput $\leq 2/5 = 0.4$ txn/sec
- Requirement was 1 txn/sec



**Allow For
(Performance) Testability**

What Constitutes Testability?

It Depends HOW Performance Testing Will Be Done

- Meaningful performance tests need automated tools (mostly)
- Understand architectural issues that affect chosen test tools
- “The devil is in the detail”

Factors that affect “shelf-life” of test scripts

- “Under the covers” changes
- E.g. stability of (hidden) attributes

Case Study 2) Complex Commercial Package

Java Based Package - Tested for Major NZ and Australian Corporates

- Package architecture allows definition of products via user defined tables.
- UI uses dynamically generated HTML driven from these tables
- Multiple, load balanced instances of middle-tier application server
- HTML field names generated dynamically at run-time - first time each function runs after server startup
- Architecture gives *flexibility* at the expense of *testability*

Sample HTML

A radio button...

```
<INPUT TYPE="RADIO"  
onClick="updateTabPosition(this,");submit();"   
NAME="FieldID54"  
VALUE="0">
```

Recorded Test Scripts break because...

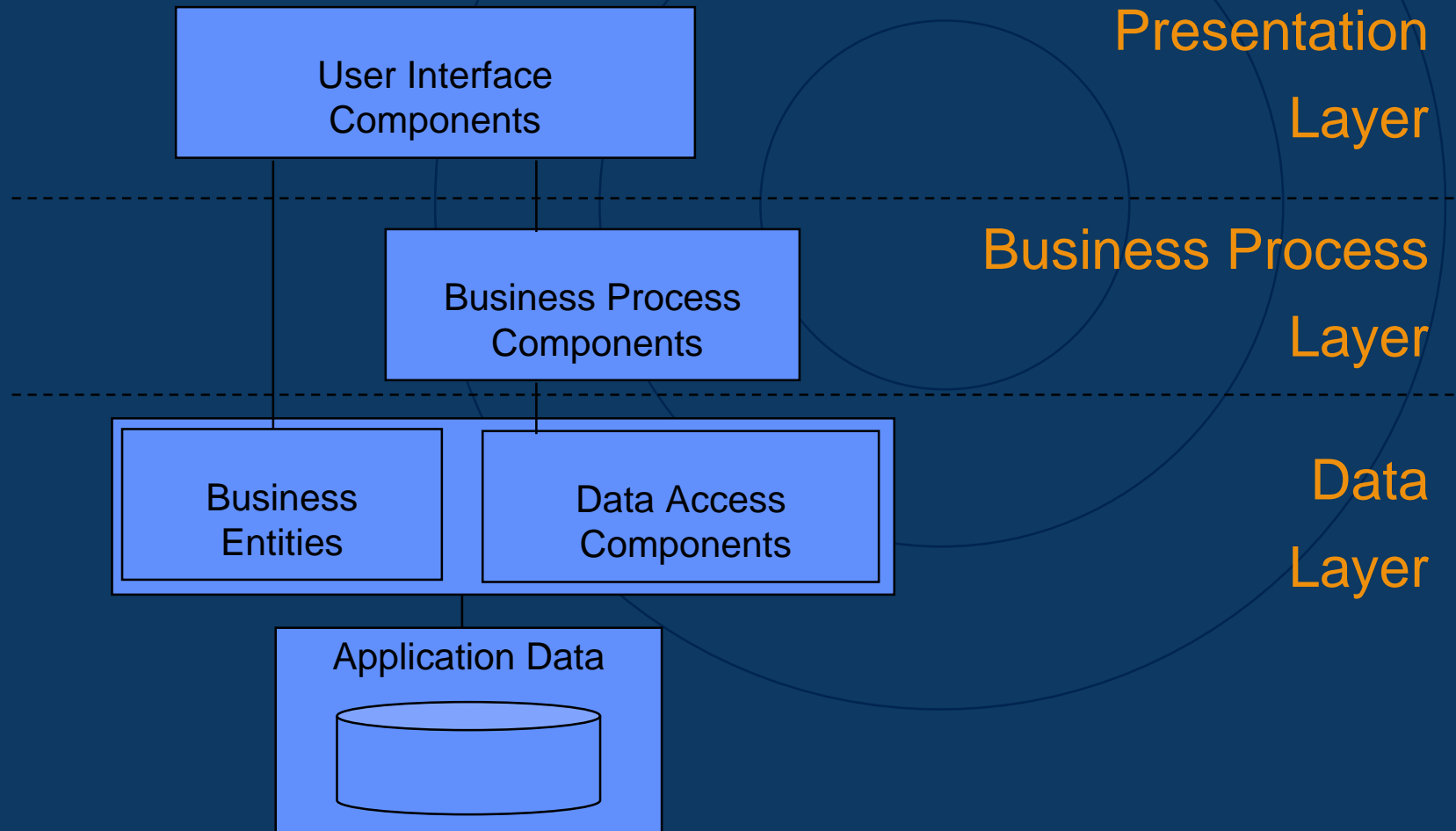
- Recorded test scripts say “FieldID54”, but next time the server starts it may be “FieldID99”
- Load-balanced server instances may use different numbers
- Minor changes in table-driven rules change the IDs

Deploying Data Access Components

–

Which Tier?

Layered Logical Architecture



Case Study 3)

Data Conversion System

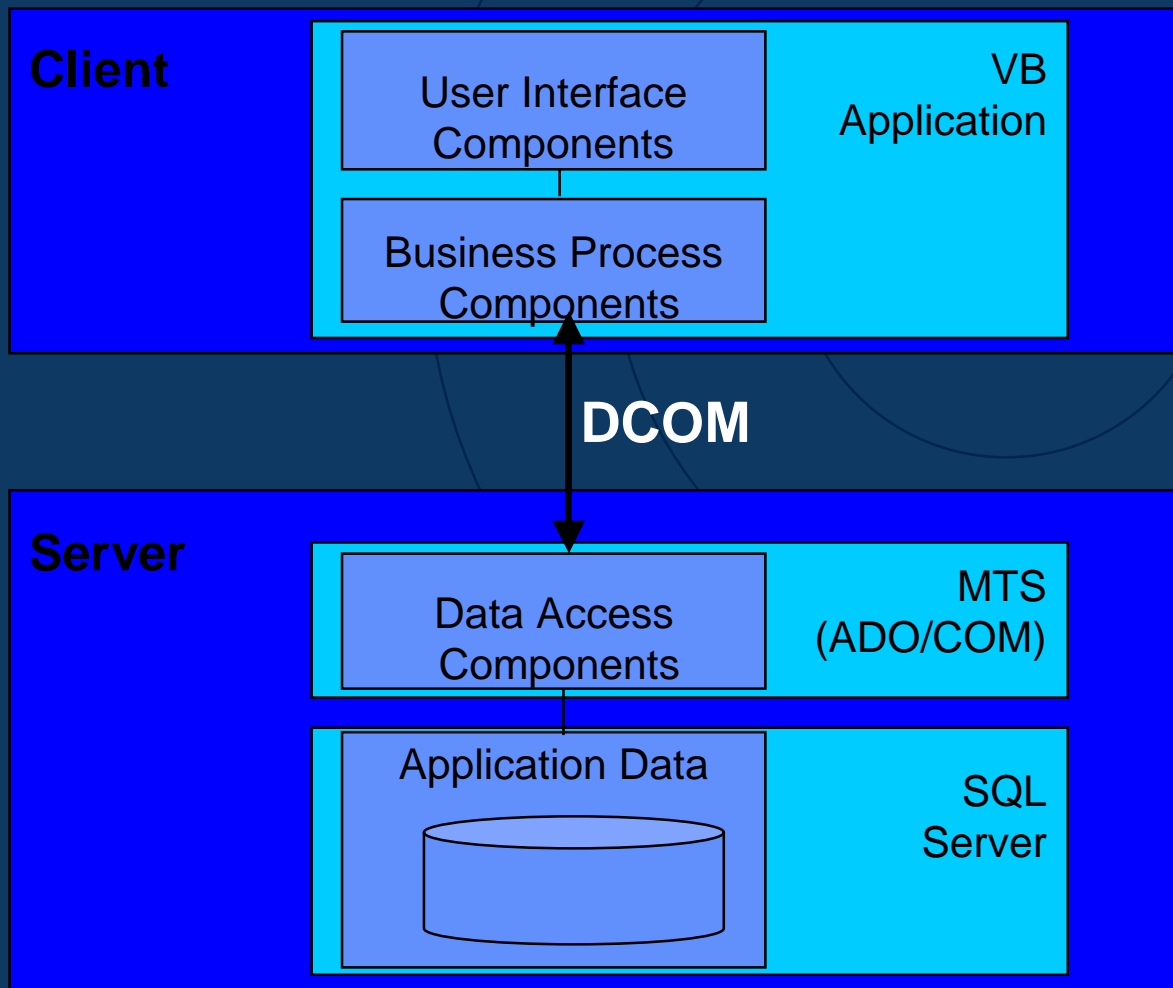
Requirements

- Intensive data capture application
- Complex data structure – required rich UI
- 1 business transaction => 50 D/B rows in 20 tables
- 150 concurrent users

Solution

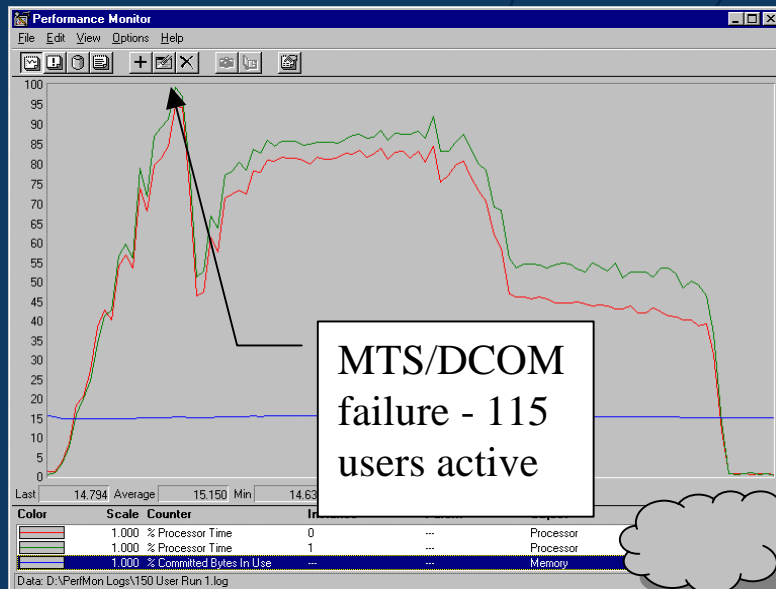
- VB, MTS/DCOM, SQL Server
- All users at a single site with switched 100 Mbit LAN

Physical Deployment of Data Access Components

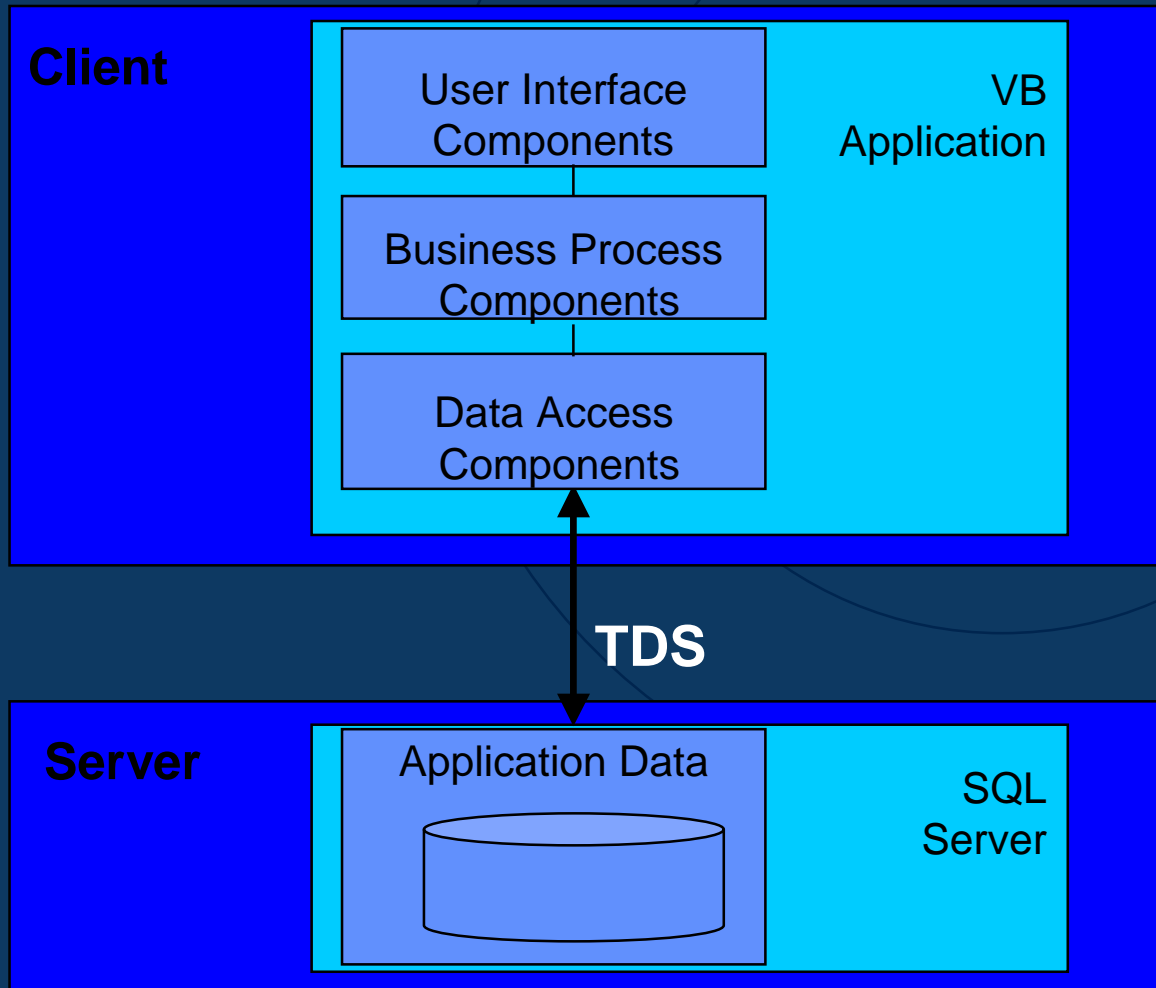


150 User Load Test – Server Utilisation

Data Access via MTS

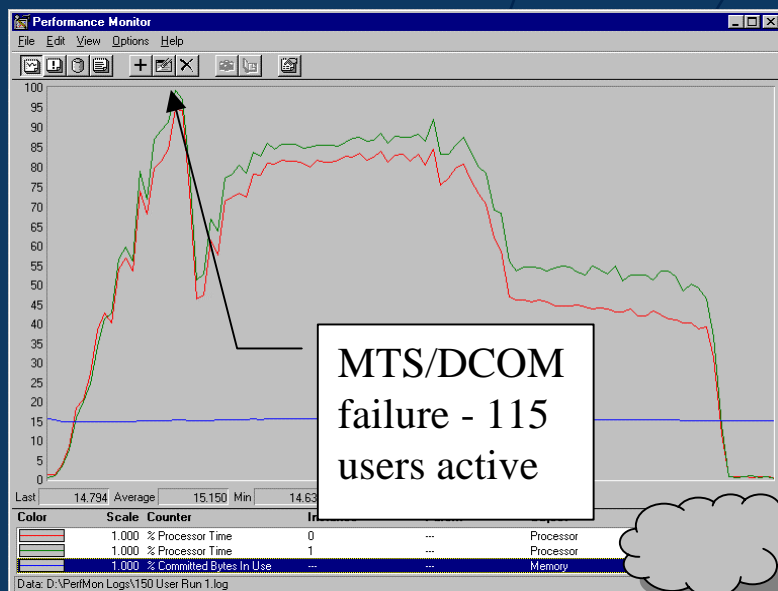


Physical Deployment of Data Access Components – 2-Tier

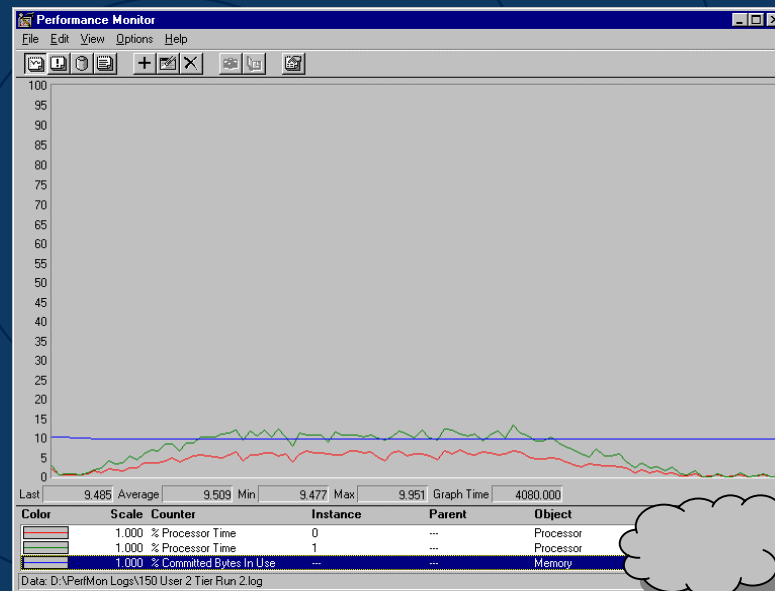


150 User Load Test – Server Utilisation

Data Access via MTS



Data Access from Client



- “3-tier” - server maxed out at 115 users
- 2-tier - server lightly loaded at 150 users

Impact on Response Times

Response times for most intensive transaction

	Minimum	Average	Maximum
3-Tier	1.92 secs	9.20 secs	35.78 secs
2-Tier	0.39 secs	0.63 secs	< 1 secs*

Notes:

- The 3-tier/MTS test never reached 150 users before it failed – results shown only go up to 115 users
- Minimum times represent single-user performance (i.e. at start of “ramp-up”)
- *A few 2-tier times over 1 second due to test tool start-up issue – steady state was all sub-second

Case Study Lessons

Scalability != Performance

- Rationale for MTS was “Scalability” (D/B connection pooling => less D/B connections), but...
- Much higher CPU load on central server (object creation)

Granularity of Cross-Tier I/F

- DCOM marshalling much chattier than SQL (TDS)
- Consider deploying Data Access and Business Process components to the same physical tier
- For remote object access, keep granularity low (“big chunks”)

Validate the Architecture Early

- In this case the performance test at end of 1st construction iteration gave time to fix it



Performance Management
-
an Architectural Issue

Why Performance Management?

Performance Degrades Over Time

- Application changes
- Infrastructure changes
- New applications sharing common infrastructure
- Database growth
- New versions of O/S
- New versions of DBMS (optimiser changes!)
- New versions of JVM
- Etc, etc

Typical Performance Issues

Problem	Resolution
<p>1500 User Insurance System</p> <p>Slow-down every afternoon 100% CPU, long run queue</p>	<p>Rewrite “branch banking”, denormalise D/B Result: < 50% CPU, no queues</p>
<p>Large Internet News Site</p> <p>Slow and variable page loading</p>	<p>Ethernet switch set-up – subtle (“incorrect”) change gave 10-fold improvement</p>
<p>Complex Java Application</p> <p>Excessive object serialisation time from app server to client</p>	<p>Required re-write of significant parts of Apache SOAP for streaming and sessions</p>
<p>Content Management System</p> <p>Stock-price component – repeated regeneration swamped server</p>	<p>Smarter management of cached components</p>

Plan for Ongoing Manageability

Good Metrics are Essential

- Consider metrics for ongoing tracking and diagnosis...
 - Business transaction volumes and user response times
 - Middle-tier component metrics
 - SQL statement performance and resource usage
 - Network and Server Utilisation
 - Etc...
- Ideally 1 repository for all statistics - Enterprise Architecture issue
- Time-series – all at same granularity for correlation

The Architect's Role

Expectation Management

- Set expectation that ongoing management will be needed...
- ... and has a cost

Application Manageability

- Ensure collection of relevant metrics is *possible* (instrumentation)

Ensure Focus on the Big Picture

- Facilitate cross-discipline co-operation
- Assist with interpretation of the data (maybe)



Summary

Some Thoughts to Take Away

Establish the Priorities

Do *Realistic* Performance Testing

Pay Attention to Data Access Architecture

Plan for Ongoing Performance Management

=> No Architecture is Immune to Performance Issues

=> Cost of Under or Over Sizing Hardware can be Huge



Questions and Discussion