



Oracle and .NET at Allstate

A case study on building a highly scalable system

Agenda



- The Problem
- Choosing Oracle and .NET
- Performance Testing Early and Often
- Mid Tier Tuning
- Oracle/Unix Tuning
- Where do we go from here?
- Where do YOU go from here?



Business Problem That We Are Trying to Solve For...

- **Simplify Technology & Process** allowing the business to:
 - **Improve** Loss Adjusted Expense (LAE)
 - Bring the right claim **skills** to help when and how they are needed
 - **Transform** the way that we interact with customers and prospects
 - Reduce Call Time through a more **intuitive & integrated UI**
 - **Rapidly evolve**, try new things, learn from them, and adapt quickly
 - Provide a foundation to **Grow & Scale**



Life Was Good, Opportunities to Game Change

● Our claim app portfolio of apps was >90+

- Smalltalk, VB 6, C++ and mainframe apps all doing parts of the process

● Data Topology

- 9 Regional Copies of Info – All on DB2
- Consolidated Reporting was required to bring the 9 copies together

● People / Process

- Our Information files were paper based – longer cycle times to work across organization (out of the office)
- Assignment of work to right people with the right skills and optimizing work distribution would improve by moving to an electronic file



The Constraints

- **“Our new system should work like the old system” - fast**
 - Response Time – sub-second page response time to a web browser like the green screen apps we will be replacing
- **Move into a two datacenter model**
 - Reduction of the # of databases, high availability
- **On Demand Growth**
 - The system that needed to handle 2x time average daily workload
- **Ability to change quickly –**
 - A component architecture that would create a logical separation of application code – easing changes
 - Implement Business Rules that are modifiable by the business



The Solution

We looked at vendors that could jump start our initiative

● The App:

- We chose a “starter kit” designed for our industry (.NET)
- Classic N-Tier Architecture – with Web/App and Database

● The Database:

- Had to pick a DB – DB2 was our standard – we chose Oracle 10g for scalability and disaster recovery
- RAC – not ERAC due to distance
- Designed a single central db for data with 2 data guard copies - logical (LDG) and physical (PDG)



Performance Testing Early and Often

Our Approach

- **Started with a “Prove It” approach to validate DB / system before development started.**
- Built out a 6 Million Claims DB with a mixed set of mock data to test against the “out of the box system”
- Validated our Hardware Platform – 4 way web boxes, Sun E25ks, SAN, and 32-bit Windows mid-tiers
- **Needed to test 2 major volumes:**
 - Release 1.0 – Property – 2 Node RAC
 - Release 2.0 – Auto (with Property and CAT) – 4 Node RAC
 - Note: We ruled out CAT in year 1 as our implementation was in July.



Performance Testing Early and Often

What we looked at

3 app layers

- Application SQL
- Mid Tier Server Performance
- Back End – Oracle DB tier (Unix/San ETC)

Scenarios

• Day in the Life, Firehose (Read, Write, Mixed)

SQL improvements in the app had a targeted benefit – no silver bullet

• Needed to focus on the mid and back end tiers

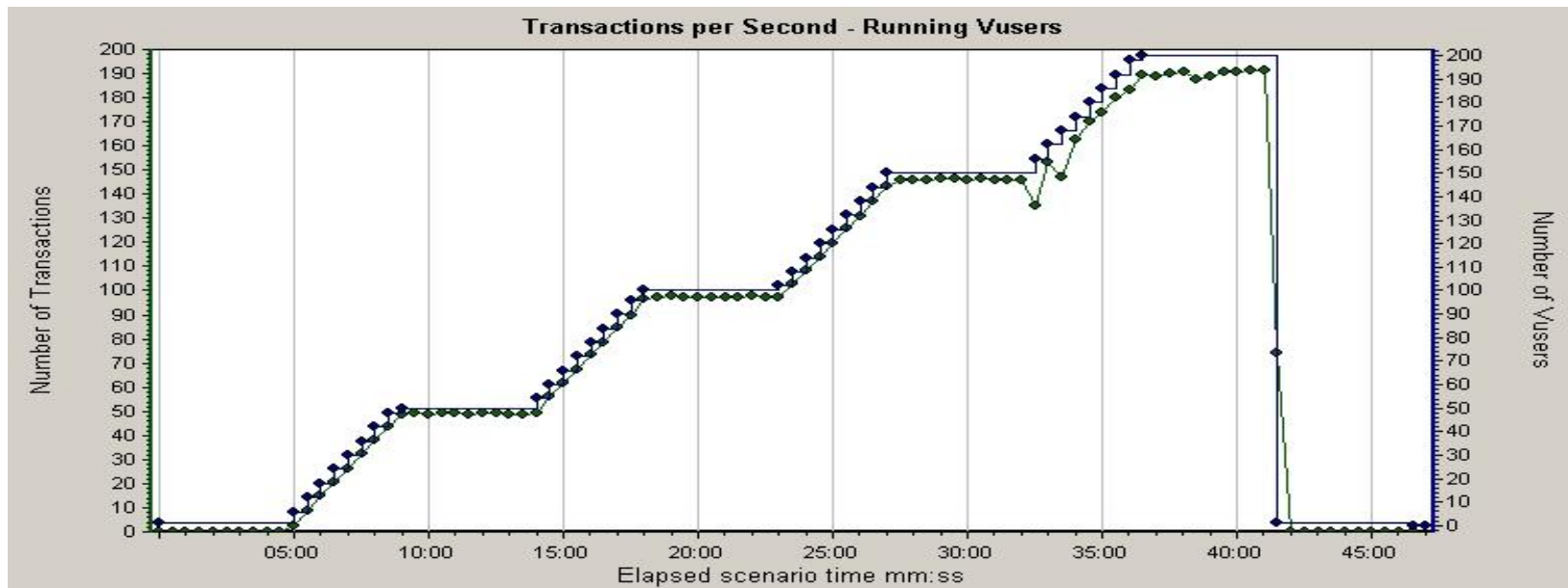
• This is where the “big levers” are

Our testing showed us very interesting results – it was Peeling an Onion



Mid Tier Scaling - Profile

- Linear Scalability from ODP.NET
- Windows CPU fully utilized through 90%
- Needed to focus on Tunable Config setting on the Mid-Tier



We wanted 2000 pages per second – We got to 200pps



Mid Tier Tuning – Fetch Size

Testing / Tuning Work on Connection Strings for R2

Tuning Fetch Size

- Fetch Size controls the amount of data fetched per server roundtrip
- Reduced Fetch Size to 32K from 64K
 - Most of the app result sets do not exceed 32K
- Results
 - Reduced Unix (server) CPU
 - Required less memory usage on middle-tier

Fetch Size (FS) can provide mid tier memory reduction – use carefully



Mid Tier Tuning – Statement Cache

Testing / Tuning Work on Connection Strings for R2

Tuning Statement Cache Size

- **Controls numbers of statement information cached per connection**
- **We set it to 200**
 - Lower than the number of unique app SQLs used
 - Started to crank it up and hit a bump (more on this later)
- **Results**
 - Reduced Windows (mid-tier) and Unix (server) CPU utilization
 - Improved application speed and response time

(FS) *Statement Cache size * Total number of Oracle Connections =
Windows Memory Required



We hit a bump at the 2GB Limit

● Limiting Memory Usage/Avoid Out of Memory Errors

- Limit number connection creations – set "max pool size"
- Reduce Statement Cache Size and Fetch Size if possible
- Try turning on 3GB switch on Windows
 - Can make Windows unstable under load
 - Currently turned off for production

● Use Web Gardening – implemented on Core Application Pool

- Optimal configuration was 1 Garden per Physical CPU core
- Each Garden is a W3WP IIS process so Memory dropped below 1GB per process
- No more OUT OF MEMORY
- More Processes == More listeners == 3 x the previous throughput
- More load on the Oracle/Unix Tier because each processes has a Connection Pool

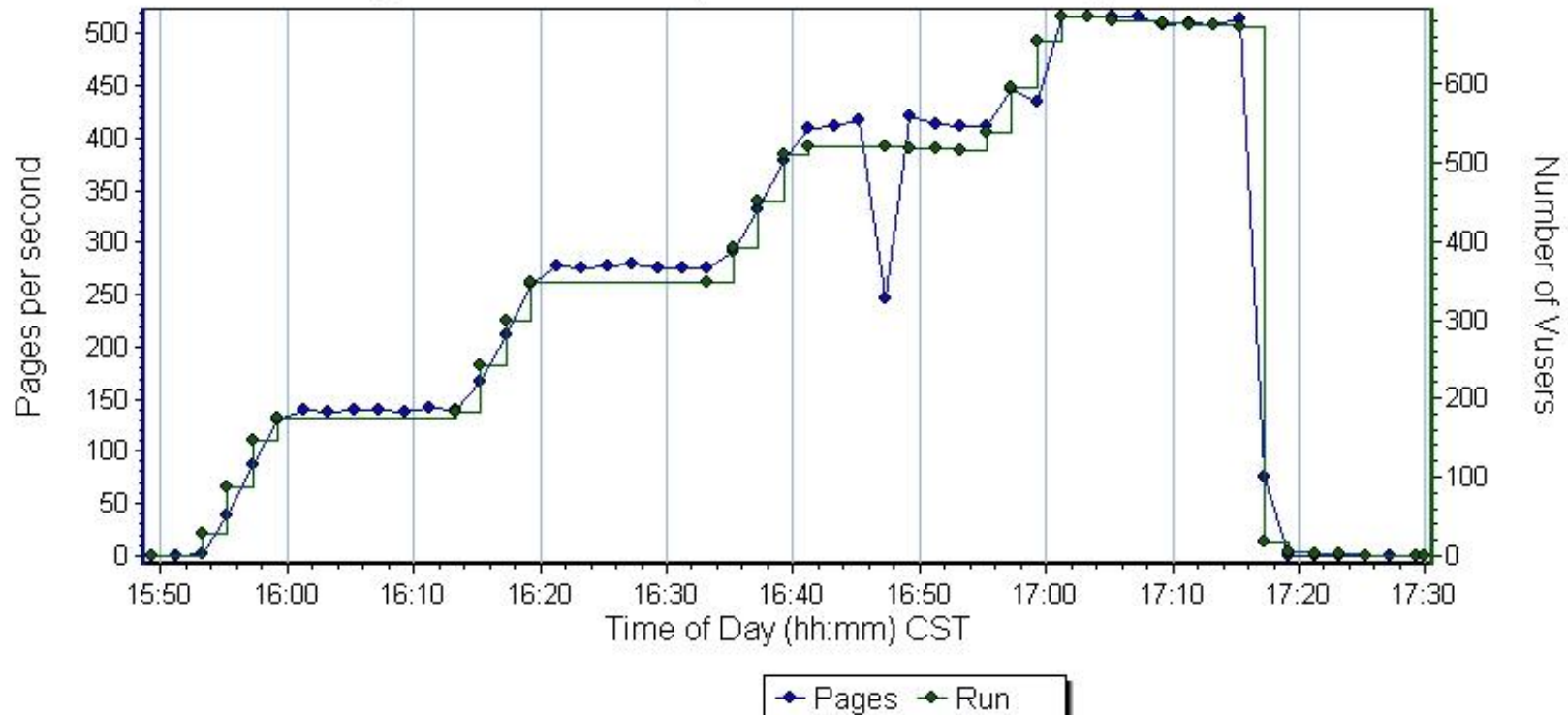


RAC Works

● All the tuning on the Mid Tier Pushed the Bottleneck to the Oracle

- Just add more CPUs to Production Unix
- SAN response times / CPU usage were increasing

Pages Downloaded per Second - Running Vusers

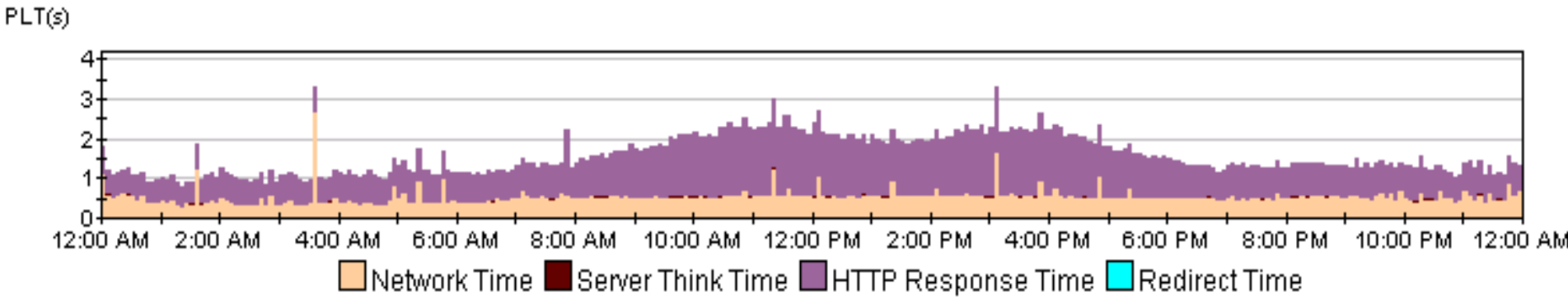




Oracle/Unix Tuning - The 2 Sec Bottleneck

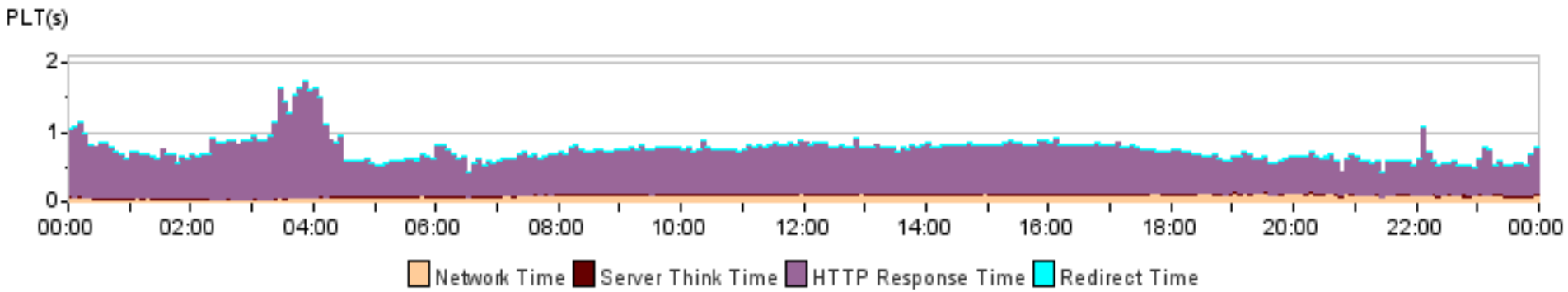
Before SGA Change (2 seconds)

Page Load Time



After SGA to uncage memory - we got to $\leq \sim 1$ second

Page Load Time





Current State

Current State

● RAC Works

- Scaled up to 192 (dbserver) processors to support CAT
- Now at 100 processors daily after SGA of 50GB.

● Mid Tier - Can process at very fast pace now

- 21 (mid-tier – quad core) servers 82k pages per sec 40-50% CPU
- 2000 pps per windows server

● Happy Users – Very Happy Customers



Would have hit a wall on SQL Server - Shared Nothing Architecture

Where do we go from here?



On the Horizon

- Move to Windows 2008 x64 with current ODP.NET driver
- Upgrade to Oracle Database 11g
- Tune Memory and Driver Parameters on Windows 2008
- Upgrade to .NET Framework 3.5 in 2010
- Read Only Physical Data Guard
- We want to jointly explore Data Entity/LINQ with oracle as a way to simplify SQL statement management in a share component architecture

We expect 11g and Win 2008 64-bit to be the next big breakthrough

Where do YOU go from here?



Recommendations to Audience

- **Oracle Database works great with .NET**
 - Initially skeptical, but performance and features convinced us
- **RAC (and ODP.NET) scale linearly as load and servers are added**
 - Great combination for Allstate and our Customers
- **For high throughput applications, performance considerations need to be top of mind**
 - Tune up and down the stack – ODP.NET, RAC, network settings, operating system, etc.
- **Leverage experience that already exists – Oracle and .NET**