At the Fun House: Experiences Tuning Oracle, Linux and z/VM for a Large ERP

Brian Gormanly
Marist College

August 9th, 2011
Session: 9367

Trademarks

- z/VM® is a registered trademark of IBM Inc.
- Performance Toolkit for VM is a registered trademark of IBM Inc.
- System z9® is a registered trademark of IBM Inc.
- System z10® is a registered trademark of IBM Inc.
- System z196® is a registered trademark of IBM Inc.
- SunGard Higher Education and Banner are registered trademarks of SunGard Data Systems Inc.
- Intel™ is a *registered trademark* of Intel Corporation
- Xeon™ is a trademark of Intel Corporation
- Oracle Database is a registered trademark of Oracle Corporation
- Oracle Enterprise Manager is a trademark of Oracle Corporation
- Spotlight for Oracle is a trademark of Quest Software
- HyperSockets/is a registered trademark of IBM Inc.

Agenda

- Where we started
- Background on Sungard Higher Education's Banner ERP
- Making it run on System z
- Initial load testing
- Registration
- Detailed testing
- Conclusions

Where we Started

- For 20 years we ran on a CICS on z/OS based ERP called IA Plus (a SunGard HE product).
- Controlled all business and academic processes for the College, including billing, course registration and payroll.
- IA Plus product has been sunset, so move to “new” ERP was necessary.
- Almost countless customizations to the legacy system over the years.
- The system had been stable for 2 decades – it was expected.
- Now, many business processes needed to be completely re-done.

Where we Started (continued)

- The Master Project Plan.
  - Executive Sponsors were fully supportive
  - 3 year complete implementation and conversion
  - Experimental Platform
    - Help IBM and SunGard make new markets
  - The only client
  - A previous attempt in the industry failed
  - Budgeting for the project was below average – on purpose
  - Bailout would cost additional millions

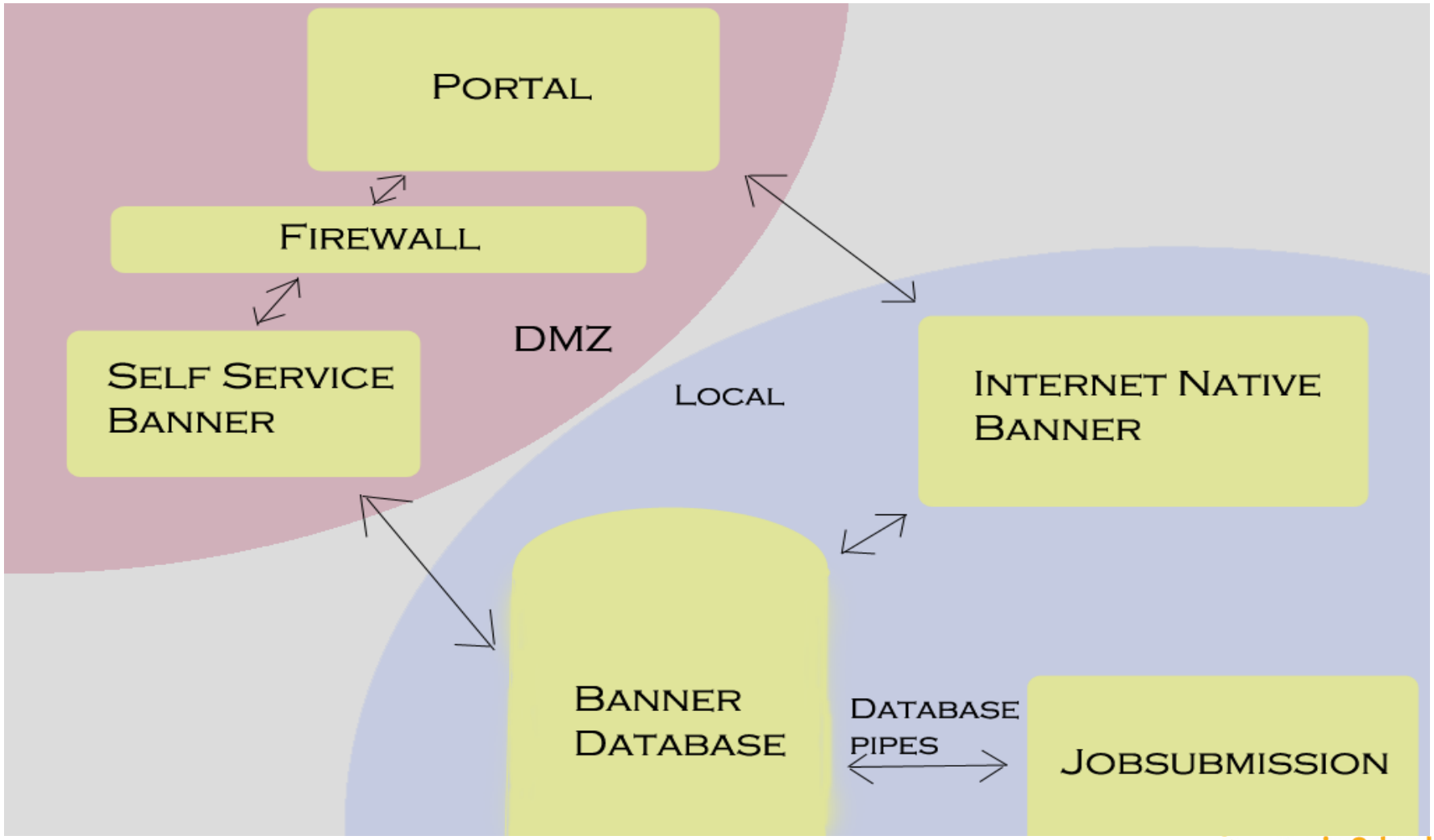
Welcome to Banner

- Banner from Sungard HE is a commonly used ERP at colleges and universities around the world.
- System z is not a supported platform. Marist arranged with Sungard HE and IBM to be test case for porting Banner.
- Implementation at Marist began fall 2007, the first module went live in 2008. The student system was the last to go live in September of 2010.
- Initial roll out consisted of database running on SLES 10 and Oracle 10g R2 (10.2.0.4) on a System z9.
- Oracle App Server Forms/Reports 10.1.2.0.2 for application server on a P550.
- While we would have to reassess all of our custom code, there was a ‘community source’ initiative in which we could submit changes for baseline.

Sungard Higher Education Banner

- Four Parts
  - Database
  - INB – Internet Native Banner: Used by administrative offices
  - SSB – Self Service Banner: Used mostly by student, faculty and advisors. This is the part of Banner that our performance tuning that is in use during our peak usage periods.
  - Job Submission: Handles mainly reporting functionality, not relevant here.

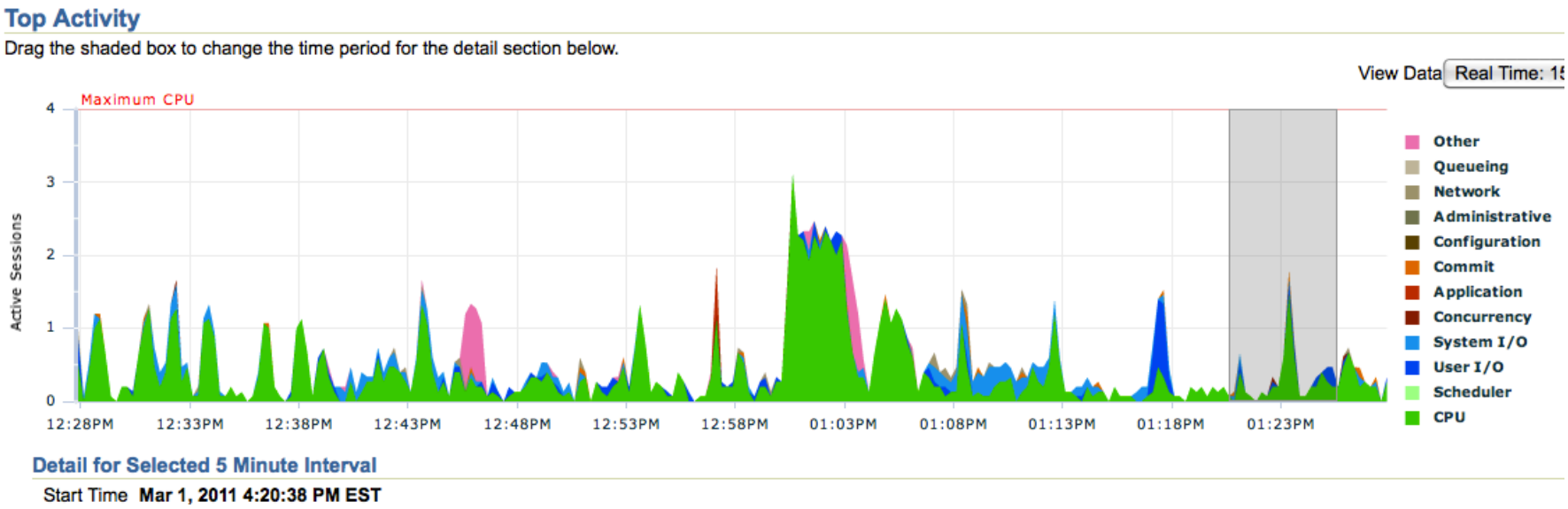
Visual Banner



First on the System z

- The database successfully ran on System z from the beginning.
  - 10.2.0.3 later upgraded to 10.2.0.4 on SLES 10
- Application software was first deployed and running on P550 hardware.
  - Migration to System z happened as data conversion took place
- Running on System z allowed us to run many instances of the database for development easily cloning and creating databases.
- Many Pro* C programs on the job submission server would not compile with certain libraries.
  - Modifications and testing required
- Implementation went very well.
  - Under normal daily administrative use the system preformed admirably.

Green is Good..



Unique Application Challenges

- Application self service web software written entirely in PL/SQL that is stored in the database, making the database server do the work of both the database and the application server, application middle tier provides graphics and manages Apache connections only.
- Sungard HE recommends high CPU count (16 Intel Xeon cores) in a non virtual, dedicated environment. We would be running in an LPAR that has around 60 guests on very few IFL's.
  - We started with 3 IFL's on System z9 in the LPAR

Student Registration

- Student registration is by far our highest load event. And it is extremely important that everything runs as smoothly as possible. All eyes are on IT. Getting students in classes is how we pay the bills.
  - Normal use will see:
    - 100 – 200 current office users during the work day (connections persist for long periods of time).
    - 10 - 30 web users at any one slice in time (connections only persist for as long as required to return information to client or are artificially extended only if keep-alive settings are used to have Apache maintain connection with client.

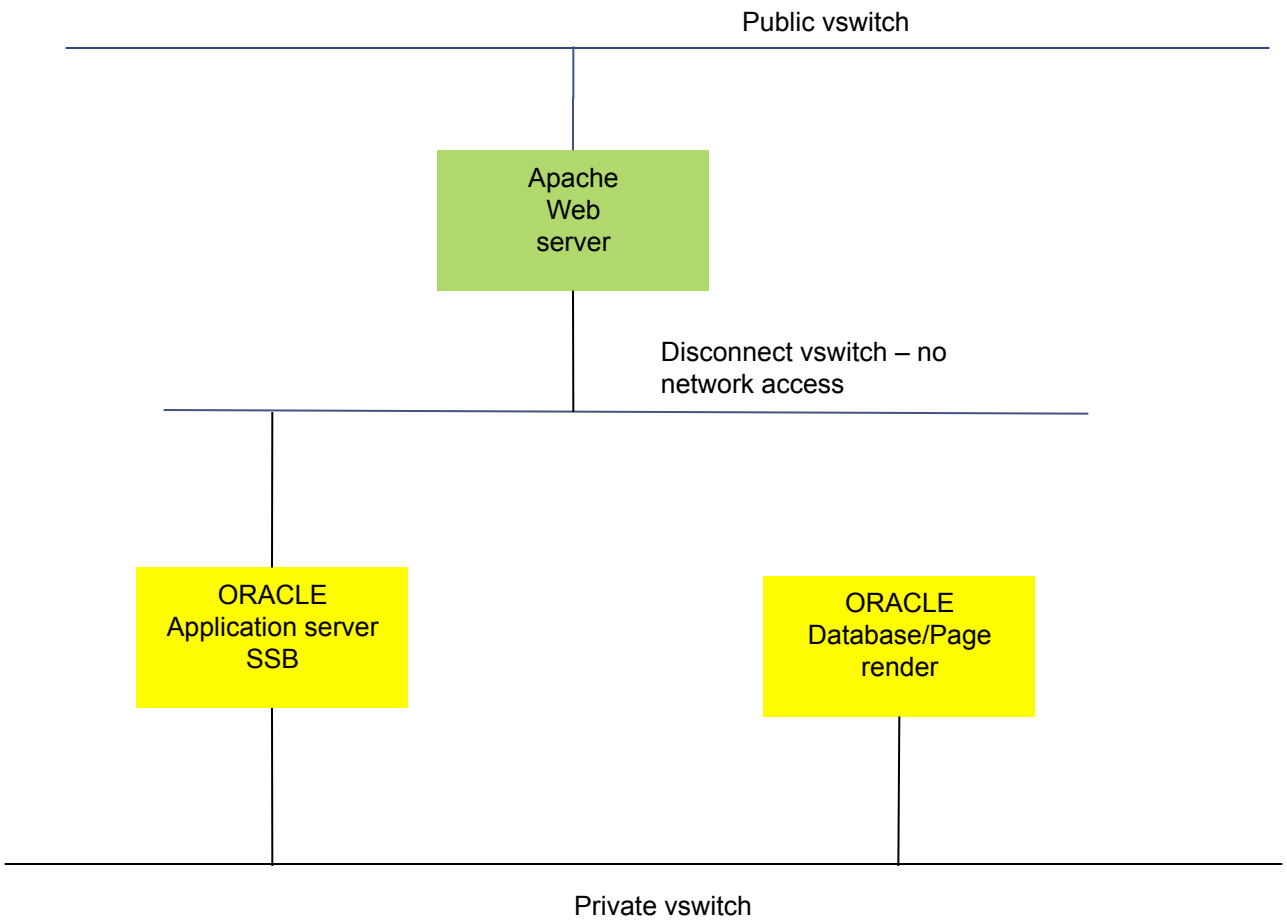
Student Registration (continued)

- Registration peak usage:
  - Office usage not a factor, registration starts early in the morning.
  - There are over 1000 students in each class all of whom are under intense pressure to register early to get in desired classes.
    - *Each class is given an open registration time that they can register.*
    - *High percentage (from 50% to 75%) of students in each class try to login and use the system at the exact same moment!*
    - *The only way to deal with the storm is to group the students into manageable sizes.*
  - Being the first to use the application on the platform we had to test to determine the maximum amount of students the system could handle at once.

Configuration as of First Registration

- The big day approached….we faced it with apprehension.
- Configuration consisted of:
  - Database running on SLES 10 SP2, 10G memory, 4 virtual IFLs
  - Application server, SSB (Self-Service Banner), 2G memory, 2 virtual IFLs
  - Apache web front-end, 512M memory, 1 virtual IFL
  - All running on same z/VM 5.4.0 system with 23G of main memory, 5G of expanded and 4 real IFLs on a z9
  - Servers connected to each other and local network via vswitch. Apache web server is the only one with public address for security.
  - LPAR shared with approx. 60 other Linux servers

Server Configuration at Registration



Initial Load Testing

- Mindful of the looming registration storm on the horizon we set about creating a test to determine capacity.
- Load testing tool “The Grinder” was chosen:
  - Open Source
  - Scalable – multiple workers can generate load and report back to central console
  - Portable – Java based, you can zip up the application folder and move to other systems and platforms quickly
- We discovered that testing the act of course searching simulated the heaviest part of the registration process load and did not require inserting bogus data into the database and setting up logins for many users.

Initial Load Testing (continued)

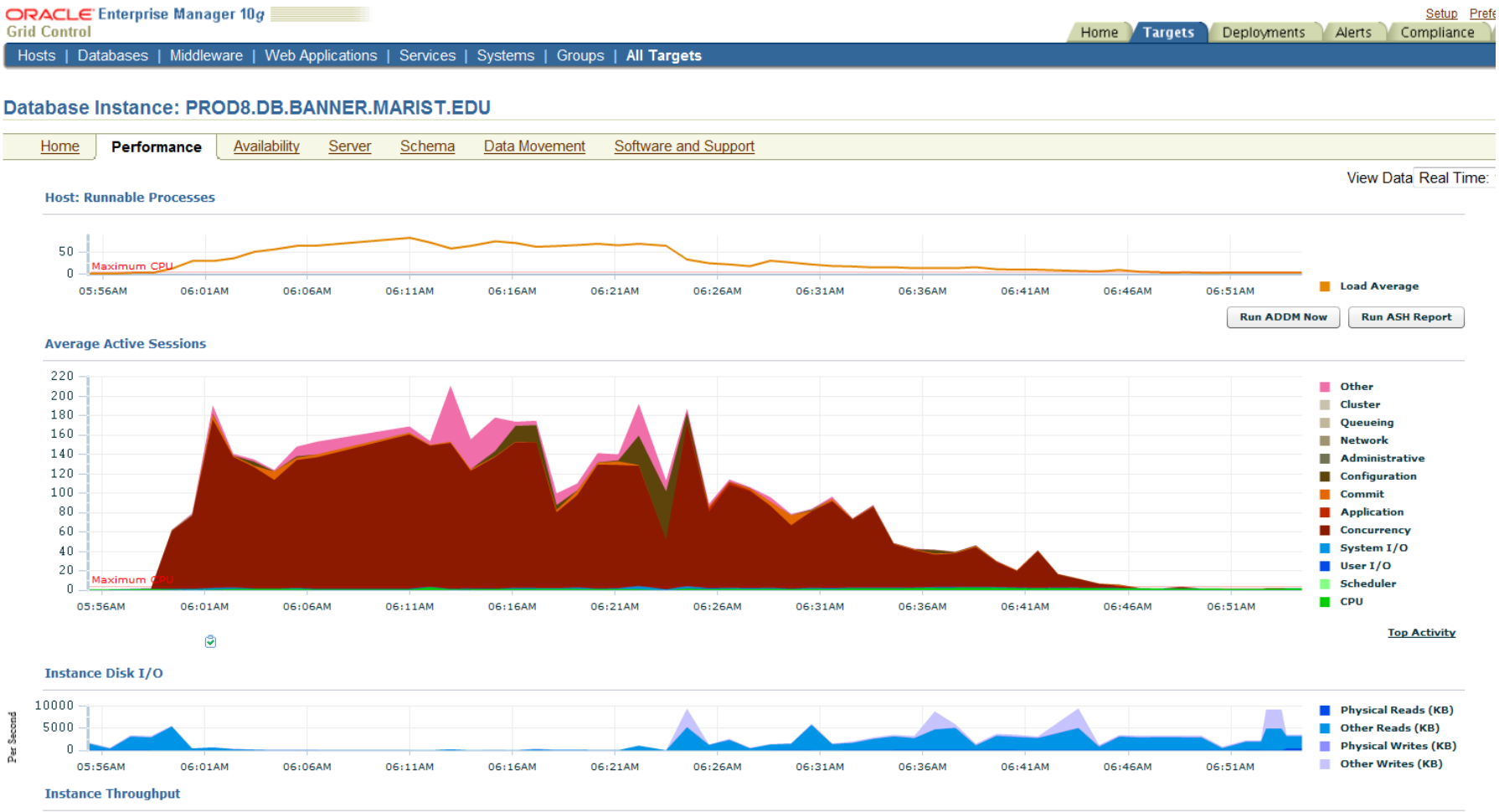
- When doing our initial runs we incorporated a 4 second wait time after each page load before another page was requested.
- Test runs through searching multiple majors and then drilling down into a few course details for each search. Sampling included both large and smaller majors.
- Initial testing on our System z9 indicated that we should be able to handle surges of around 400 simultaneous users.
- Oracle's Enterprise Manager along with Linux system tools were used to oversee database and system load.
- Classes were divided up into appropriately sized groups. And we were ready to go…. So we thought.

Registration – Day 1 – A Little Rough

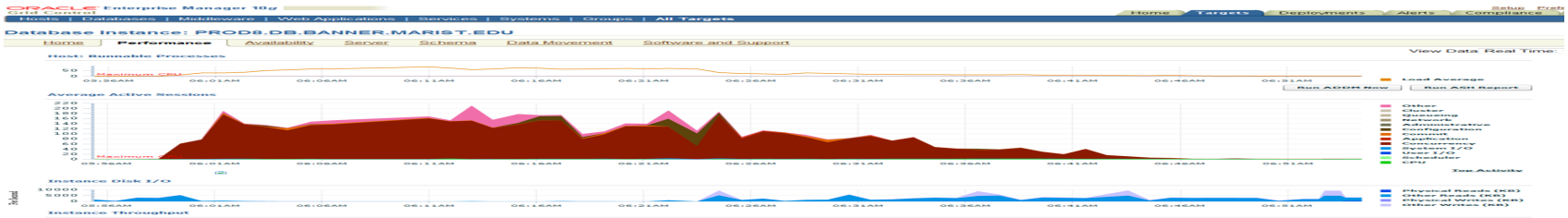
- Day 1, was not a large event. Classes were split in pieces and sense of urgency was not as strong for the lucky ones who were in the first group.
- There were very few complaints.
- We felt a certain sense of accomplishment

Until….

Day 2


ORACLE Enterprise Manager 10g
Grid Control
Setup Prefs
Home Targets Deployments Alerts Compliance
Hosts | Databases | Middleware | Web Applications | Services | Systems | Groups | All Targets
Database Instance: PROD8.DB.BANNER.MARIST.EDU
Home Performance Availability Server Schema Data Movement Software and Support
View Data Real Time:
Host: Runnable Processes
Maximum CPU
Load Average
Run ADDM Now
Run ASH Report
Average Active Sessions
Other
Cluster
Queueing
Network
Administrative
Configuration
Commit
Application
Concurrency
System I/O
User I/O
Scheduler
CPU
Top Activity
Instance Disk I/O
Per Second
Physical Reads (KB)
Other Reads (KB)
Physical Writes (KB)
Other Writes (KB)
Instance Throughput

Day 2 – Oh, the Horror!



- Day 2, 6 am arrived and the flood gates were opened. The Seniors that were left desperately try to register as classes fill up.
- Overall load on z/VM rose quickly to 100% on all 4 IFLs with 98% of it going to the database machine.
- All productive work on the database stopped. The database and console both hung and became unresponsive. The clock ticked off the minutes while nothing happened. We had to forcefully shutdown the database to recover.

The Day After That…

- Load dropped substantially and was handled with no problem….until next class started to register.
- The Problem is handling peak demand at start of registration. The “Login Storm” occurred because users were waiting for a specific point in time to try to access the system simultaneously.
- Students, administrators and management were NOT HAPPY.
- Our first reaction was to work with Velocity to improve tuning of z/VM.
  - Reset relative shares back to default of 100, no upper limit
  - Increased relative shares on major servers that had more than 1 virtual IFL (100 x number of vIFLs)
  - Added 100 to servers that should get more priority
  - Overall performance improved, but problem did not go away

Damage Control – Oracle and Application Tuning

- Since the system became unresponsive, we also looked at the ways in which we were handling the swarm of database connections being established and ways to limit them.
- Initial actions included using Apache to limit the MaxClients directive to 40 changed the KeepAlive directive to off.
  - We found that setting MaxClients to 40 substantially lowered the amount of load on the Linux guest and helped prevent the guest from taking all of the resources of the LPAR
  - Setting KeepAlive to off freed up connections immediately after they were complete, so the 40 available connections could be used most efficiently
- These changes, along with breaking the classes up into smaller groups helped us survive the remaining registration.

MORE POWER!?!?!

- At least, that is what we hoped. A new z10-BC was purchased in December 2010. 2 GPs to support legacy LPAR and 4 IFLs for Linux servers LPAR (MARLINUX).
- After installation and configuration, test Oracle servers moved there to evaluate.
- Tests showed that overall number of users serviced went up, but problem of the load reaching incredible levels still occurred, even with no other workload on the LPAR. We could manage load only by limiting users with max clients.
- Before going live on the z10 we were able to test with no other workload and found 15% increase in the amount of users handled compared to a fully loaded LPAR in production.

But Wait! There's More! – Load Testing

- We decided to do some testing on other platforms including Power and X. The results indicated that we might be experiencing a platform specific problem.
- Working with a consulting company, Mainline, a taskforce was formed to evaluate the problem. Group members included individuals from IBM, Oracle, Mainline and Marist.
- Previous load tests were refined and standardized for testing at the IBM Poughkeepsie Mainframe Benchmark Center. This would provide a consistent test environment. Without fear of affecting any production systems.

Testing – Platform Detail

- Cross platform testing
  - Test systems included:
    - At Marist:
      - *System z9*
      - *System z10 BC*
      - *P550*
      - *System X*
    - At IBM Poughkeepsie Mainframe Benchmark Center:
      - *System z10 EC*
      - *System z196*
      - *System X*
- Testing at the Benchmark center removed other variables from testing. Results were constant and comparable.

Platform Detail (continued)

- All testing was performed from a X server on site using The Grinder.
- Network used was entirely gigabit Ethernet.
- Tools used for monitoring the system during test analysis included:
  - Oracle Enterprise Manager
  - Spotlight for Oracle

Testing Overview – Where's the Beef?

- The biggest and most immediate finding was that the application design was inefficient from a peak performance perspective. Queries returned far more data than necessary and the same data was queried from the database multiple times without need.
- Concurrency was at the center of the problem. Chicken and egg situation, was the applications design causing extra spinning because resources could not be accessed when needed? Or were limited cpu resources creating a situation of contention for resources?
  - Pin S Wait on X : waiting on child cursor
  - Library cache waits
  - Cache buffer chains latch

Outcomes of Testing

- With the problem identified we were able to set about formulation a plan for improvement.
- Testing concentrated on 4 areas where performance could be measured and might be able to be improved.
  - Oracle database tuning
  - Application tuning
  - Hardware performance
  - Connection allocation and management
- We set to work attacking each area separately, one at a time, to ensure the results were accurate and easily measurable.

Oracle Database Testing Detail

- To determine what gains could be made in the database, we:
- Tried many documented and un-documented Oracle parameters and settings, we also tried different SGA and PGA configurations.
  - _kks_use_mutex_pin, _cursor_features_enabled, cursor_sharing, cursor_space_for_time
  - We did not realize any appreciable gains changing the parameters
- Tested on multiple Oracle versions including 10.2.0.4, 10.2.0.4.6, 10.2.0.5, 10.2.0.5.2, 11.2.0.1 and 11.2.0.2 (11g was tested on the X server as we wait patently for it on System z).
  - We found almost no real difference in using Oracle versions 10.2.0.4, 10.2.0.4.6, 10.2.0.5, 10.2.0.5.2, 11.2.0.1
  - However, once we applied the 11.2.0.2 patch, magic happened!

Oracle 11gR2

31% Performance increase! (11.2.0.2)

Application Testing Detail

- To try to discover how much of a role the application played in the problem, we:
  - Made database and application modifications, including:
    - Identifying “Hot” blocks in memory where concurrency was occurring. The following methods were tried to alleviate
      - *Some tables rebuilt with 4K and 32K datablocks and re-indexed*
    - Created in-memory arrays to store information, requiring less cursor creation.
- Still more to explore, we have been working with Sungard HE to formulate a plan to help improve performance.

Application Gains (so far)…

Changes so far have yielded a 12% performance increase! More gains will be realized in this area.

- Adding in program memory one value alone that originally was being retrieved from database cache on each call provided a 5.8% gain! (the term)

Hardware Testing Detail

- Test results showed that problem was produced on all platforms, differences included the amount of users the system could handle before system load increased dramatically and how bad the system reacted. (console locking up, etc)
- We tried many combinations of memory and number of IFL's to see if there was an amount of hardware that would affectively make the problem vanish.
- Number of cores was king. And was more important than any other factor, on any platform.

Hardware Testing Detail (continued)

- We tested 4, 8, 9 and 16 z10 IFL's and on an X's with 4 and 8 cores and saw nothing but a linear increase in work getting done.
- We looked into whether or not OOE (out of order execution) might be playing a role.
  - The thought was that concurrency might be less pronounced if the processor can "put aside" a spinning thread, letting more work get done.
  - z10 and P6 does not support OOE, but X, P5, P7 and Z196 does.
  - Tests were inconclusive, OOE not fully supported in SLES 11 SP2.
- The largest gain was seen when going from the z10 to the z196…

System z196 Gain

51% gain! It is believed the caching benefits of the z196 is the main reason..

Shared vs Dedicated Connections

- To survive the first registration we used Apache to limit the number of database connections that were allowed and how long they remained active.
- In testing we discovered that managing the connections using shared oracle connections was much better.
  - Users received far less errors when the server was heavily loaded. Max Clients caused many immediate errors as Apache turned them away at the gate
  - More productive work was getting done
  - Server load can be managed much more efficiently. Shared connections allowed us to handle many more users while cutting server load more than half!
    - from above 70 to in the 30's (load could be further decreased by reducing the number of shared connections, balance tuning)

Shared Connections Gain

10% Gain in the number of users that could be handled without a drop in response time.

- More importantly, shared connections allowed us to handle 100's more uses, while cutting server load. The only by-product of the increased user load was increased response time, saving us from another failure of the system

Conclusions

- Our work is not done, but a lot of progress has been made in understanding the problem.
- Our largest gain was seen in testing on System z196, 51%.
- Second largest gain was moving to Oracle 11.2.0.2, 31%! Come on, 11gR2 for System z!
- Application tuning has already yielded a 12% gain with just a few quick changes. We want to explore working with Sungard HE to rewrite troubled parts.
- Shared connections provides catastrophe avoidance! (and a 10% gain).

Conclusions (continued)

- We managed to invent a new solution and optimize the application.
- Our work is not done, but a lot of progress has been made in understanding the problem, and implementing fixes.
- We believe we have helped make new markets.
- In the process we not only improved our systems, but all current and future users of the system.

SHARE
Technology · Connections · Results

Questions?????

SHARE in Orlando 2011