EDGE EMEA 2009
October 11-13, 2009
Amsterdam

Optimising CA Gen applications

Tim Dargavel, Facet Consulting

Session Track 3
Tuesday 13th October, 2009 - 9:50-10:30

Biography

Tim Dargavel
Business Development Manager at Facet

Tim has worked with CA Gen for 15 years now, and over 25 years on Enterprise-scale applications. He has worked throughout Australia, Asia and also in Europe and has experience with the mainframe, distributed, and web platforms as well as enterprise, applications and technical architecture, and lots of development and applications tuning.

EDGE EMEA 2009
October 11-13, 2009
Amsterdam

Overview

Optimising covers performance tuning and a few other things
Why bother with performance tuning?
What is involved with performance tuning your applications?
Common areas to look at during performance tuning
Other areas of application optimisation
A couple of case studies.

EDGE EMEA 2009
October 11-13, 2009
Amsterdam

Why bother?

Cost – savings can be delivered by optimal platform use
- MIPS on the mainframe – delivers recurrent savings
- Deferral of capital expenditure on hardware upgrades

Risk - to ongoing application availability – service continuity
Compliance - service level agreements need to be met
User experience – may be maintained or improved.

EDGE EMEA 2009
October 11-13, 2009
Amsterdam

What triggers the need for tuning?

Sudden problems vs: Gradual worsening

The risk to ongoing application availability – service continuity

Changing externalities usually degrade performance over time

Continual degradation

- Slower response times for end-users
- Increased batch processing

Continual degradation eventually results in failure

- Transactions begin to fail due to timeouts
- Batch processing starts to creep into the start of the online day

So the need to do performance tuning is usually inevitable

- Proactively – as part of the SDLC or special projects or
- Reactively – when application failure occurs.

Environmental triggers

Sometimes your application is unchanged, but still needs to be tuned

The application usage changes over time:

- Transaction volumes increase
- Transaction processing profiles change
- Data volumes increase as the application ages
- Network traffic increases as transaction volumes increase

Other tiers of the application architecture may degrade

Inter-connected applications no longer perform (as) efficiently

Infrastructure capacity has not scaled with increased application usage.

Understand the need

There are a variety of drivers that may trigger performance tuning

Be clear about which drivers are the ones you are addressing

The technical ones, and

The political/perception ones…

The Performance Tuning process

Ideally forms part of the standard Testing process

- Usually occurs after the application functional testing is close to completion
- Often undertaken with, or as part of, technical stress (load) testing

Alternatively: Undertaken in response to major production slow-down

- This often results in considerable pressure for rapid resolution
- In pressure situations, emotion and opinion are loud, but data will prevail

The underlying process is the same however:

- A four step process
- Incrementally repeated until "acceptable" results achieved.

The Performance Tuning process

Baseline: Measure how bad the problem is now

Identify: Understand where the problems lie

- Mean Time Between Resolution (MTBR) – 75% of time is in identifying the problem
- Use data from reporting software (CICS logs/Run stats)
- Determine which transactions are the ones to pay attention to
- Utilise performance monitoring tools (Strobe, Sysview, CA Wily)
- Produce a "Top-10" list of areas that need resolution

Resolve: Incrementally resolve each bottleneck

Prove: Compare new results against the original baseline.

Process: Baseline your application

Baselining requires collection of metrics on the application as it is now

It is crucial to have the data that defines performance

Look at the following

- Transaction volumes
- Transaction execution times
- Transaction load profiles – peak processing periods especially
- Machine performance – CPU, I/O capacity, Database load, Network bandwidth

And…

- Ensure you understand why people believe there is a performance problem
- What is the user experience – what is the end-customer impact?

Process: Identify the problems

Sudden problems usually have a change-related cause

- The change may be from modification to the application software
- The change may be from patching/upgrades to the infrastructure software
- The change may be from hardware modifications
- The change may be from network configuration changes
- All of these should be identified through your Change management process!

Gradual problems may have any number of causes – usually many

- Poor application coding practices introduced over time
- Database tuning required for increased transaction/data volumes
- Environmental factors outside your application – be aware of these.

Process: Common types of problems

View handling

- High-performance view-passing should always be set
- Perfect view matching should be used wherever possible
- Dramatically affects deep calling chains – especially in CBD environments

Data base access

- Read properties should be appropriate
- Ensure table joins are made using indexed columns
- Ensure there are no function calls in the qualifying of READ EACH statements
- For very large cursors (typically batch), consider sorting the index for the READ EACH

Inbuilt functions

- Performance improved significantly with CA Gen R7.6
- Provide CA Gen's technical abstraction – but can be very processing intensive.

Process: Resolve identified issues

Prioritise the issues to resolve and be practical

Resolution of some problems may require significant design changes

Group the changes into a number of categories

Action now:

- Changes to the application code (Cluster into functional areas)
- Changes to the database organisation (Indexes, Tablespaces, Paging, Sorting)

Action later:

- Set aside design changes to be done with Business-initiated activity
- Set aside less-urgent application changes to be clustered with other change

Particularly important if you are tuning as a result of a crisis!

EDGE EMEA 2009 October 11-13, 2009 Amsterdam

Process: Prove the results

Having made the changes – check the revised application

Ensure that the environmental baseline is the same

Collect the same metrics as the agreed set from the baseline

Compare the new results to the baseline

- What has worked – has sufficient improvement been made?
- What has not worked – Why? What else can be tried?

Always come back to check if the original problem has been resolved

- Has the broken SLA now been met?
- Has the customer/end-user experience been restored?
- Have the perceptions of the performance problems been addressed?

EDGE EMEA 2009 October 11-13, 2009 Amsterdam

Performance Tuning summary

Know when to start doing this:

- [Proactive] Ideally forms part of the standard Testing process
- [Reactive] Serious performance problems will trigger this activity

The underlying process is the same however:

- Baseline to collect the metrics on how your application performs now
- Identify where the problems worth solving are
- Resolve the "Top-10" actions that can be resolved – some may be for later…
- Prove the effectiveness by comparing to the original baseline

Know when to stop!

Remember that because our applications become more complex

- Additional tiers, platforms and connected applications make tuning more difficult
- Better to be pro-active than reactive
- Tuning your applications can save real money

EDGE EMEA 2009 October 11-13, 2009 Amsterdam

Performance Tuning recommendations

Save your organisation some money by doing this now!

Plan and execute it periodically rather than in an emergency

Ensure your coding standards incorporate performance considerations

Ensure you have related specialists to work together

- Expert developers with experience in the applications
- Senior DBAs with experience in performance tuning
- Network specialists who can analyse the relevant traffic
- Systems Engineers for the Operating System environment

Find out what tools you have now that can help you do this

EDGE EMEA 2009 October 11-13, 2009 Amsterdam

Example scenario - Tuning

Australian Public Sector (Z/OS, DB2, custom Java)

- Massively degraded customer experience

Some challenges

- Lots of political "noise" on the need for action and the "causes"
- New application system release deployed in the same timeframe
- Lots of new functionality compounded with training issues

Resolved with…

- Review of metrics, and actual performance, and…. no application change
- But did trigger a subsequent (planned) performance tuning project.

EDGE EMEA 2009 October 11-13, 2009 Amsterdam

Opimisation and tuning

Optimising is about getting your applications working as well as possible

Performance tuning is part of this

Performance tuning gets each application as efficient as possible

But periodically look at the bigger picture

- What problems were your applications originally designed to solve?
- How have they evolved since then?
- Can the current solution be delivered more efficiently?

Upgrades are a good time to look at this

- Significant testing is often involved
- Additional business benefit can be delivered.

EDGE EMEA 2009 October 11-13, 2009 Amsterdam

Opimisation opportunities

Industry continues to deliver better packaged software

Many industry-wide functions are now best within external packages

- CRM solutions for Customer management
- General Ledger packages for financial management
- Inventory management and a wealth of other functions

CA are part of this industry that continues to expand CA Gen capability

- What technical capability was built in-house, that is now delivered from CA Gen?

No-one can afford to re-invent the wheel any more

Review what could be better delivered with external software for

- Reducing risk to your organisation – the vendor accepts risk for functional delivery
- Reducing cost to your organisation – products cost less than specialised services.

EDGE EMEA 2009 October 11-13, 2009 Amsterdam

Example scenario - Optimisation

European Bank (Z/OS, DB2, HPUX, Solaris, Oracle, custom Java)

- Application success story (30% year on year transaction growth)
- Hardware struggling to deliver on SLAs

Some challenges

- "Lost" the source code for the in-house proxy generator
- Web and Application servers dramatically under-utilised
- Both server sets require constant monitoring and manual rebooting

As part of a CA Gen upgrade to R7.6

- Replaced the hand-coded .COM proxies and plumbing with CA Gen proxies
- Performance tuning on application after the upgrade – metrics before
- Massively successful – significant performance gains and SLA improvements.

EDGE EMEA 2009 October 11-13, 2009 Amsterdam

Summary

Performance tuning is the bread and butter of optimisation
Understand what the drivers for performance tuning are
Always use objective metrics to analyse the problem and prove the results
Know when to stop tuning
Optimisation takes a wider application portfolio view
Look for opportunities to replace custom software with product

Questions?