

# Proactive Tuning vs. Reactive Resolution

- Know how your site/database runs
  - Baseline measurement
  - Continual review
  - Effect tuning recommendations
- Repeat measurement and tuning cycles
  - The frequency will depend on change rate

Measure → Review → Tune

But...

...there comes a time when tuning the database is not enough

Normally around the same time you start hearing...

- “The website is slow”
- “The database is slow”
- “The server is slow”
- “We need a new server”

IMHO a new server is not necessarily the panacea, just an expensive stop-gap.

And...

...a highly tuned database can be a time-bomb waiting to happen

- Impact of problem code or specific events can bring a site to it's knees very quickly...  
e.g. rugby (players), handbags and broken cellphones
- Just how can this risk be mitigated?
- When should you do it?

## It's all about what and where you "cache"

cache ([IPA](#):/kæʃ/, like "cash" [\[1\]](#)) is a collection of data duplicating original values stored elsewhere or computed earlier, where the original data is expensive to fetch (due to slow [access time](#)) or to compute, relative to the cost of reading the cache. In other words, a cache is a temporary storage area where frequently accessed data can be stored for rapid access. Once the data is stored in the cache, future use can be made by accessing the cached copy rather than re-fetching or recomputing the original data, so that the average access time is lower.

But... Isn't this automatic? Yes and No!!!

# What “cache”?

1. Client Browser cache
2. ISP content cache
3. Hosted content cache
4. Local web server cache
5. Distributed application cache
6. Persistent database cache
7. Database buffer cache
8. Physical disk cache
9. ... and not to be forgotten

So, how do you manage them?

Where do you get the best return?

Where's the best bang for the buck?

## Denormalisation

which is just another cache internal to the database/application design

## Physical Disk?

- Can only service a limited number of Input/Output operations per second (IO/s).
  - A larger 'controller' cache will enable you to clear more data in write-back mode to disk. For writes.
- Controller read cache is not important! You will have problems getting this through to your storage/server tech!
- Database "buffer cache" increases this by buffering both reads and writes (dirty pages waiting checkpoint).
  - A 99.6% buffer cache efficiency will reduce reads by 250x.

## Database Buffer Cache?

- Now, instead of 99.5%, if your buffer cache is at 98% efficiency, how much more physical IO are you actually doing?
- Answer: four times  
[ only 50x saving (1/2%), instead of 200x (1/.5%) ]
- Quadrupling the memory for your buffer cache will NOT ever (read, NEVER) achieve the same results vis. 1GB -> 4GB
- No magic ratios... of cache size to database (can be as low as 1:50 and still as optimal as possible).
- What's currently in your buffer cache? Why? How? When?

## Persistent Database Cache?

- What do you need to do to scale a query out that takes 100ms and currently runs 10 times per second, to 30 times per second?
  - A bigger/ faster server?
  - Another server?
- How about cache the results of the query for reuse?
- Recode the query to generate the result and persist the key details. Subsequent queries run significantly faster based on these persisted details.
  - $1 \times 130\text{ms} + 9 \times 30\text{ms} = 400\text{ms} (+20 \times 30\text{ms})$

Example; *Trade Me search results and category listings*

# Application Cache? Distributed?

- Use the webserver memory, its \$cheap
- Data persisted in structured form for reuse/lookup
- Content Expiry? Forced?
- Distributed cache?
  - Useful to persist slow-changing results
  - Useful when running multiple webs to reduce requirement for multiple hits to/from database
  - Pros and cons when compared to individual webserver caching

Example: *Trade Me category index and parent category pages*

## Browser/ Content Cache?

- Big savings for traffic
  - Fetch the content once per (expiry) period
- Improved user experience
  - Graphical content significantly more snappy
- Some ISPs do this upstream from the browser as well to reduce transit costs.

# What's next?

- How to get started?
  - With a site review...
  - Then, to  
Measure → Review → Tune
- When to stop?
  - When you cannot see any measurable and positive results with your tuning actions...
- Or, do you need any help?

[John@projectx.co.nz](mailto:John@projectx.co.nz) or [Paul@projectx.co.nz](mailto:Paul@projectx.co.nz)  
[Auric.PG@gmail.com](mailto:Auric.PG@gmail.com)