How Fujitsu Racing is super-charging its OpenCMS performance

Author: Dammian Miller
About me

- From an island called Australia
- Web developer for 10+ years - Java focus, government, corporate, private, consulting
- OpenCMS since 2005
- Implemented a number of small and large projects using OpenCMS, notably:
  - Australia's leading loyalty program website
  - Australia's leading department store members site
  - Fujitsu Racing V8 Supercars team website
  - And a blatant product placement - my company's fully managed multi-site OpenCMS offering
- OpenCMS evangelist
Here to introduce the open-source OpenCMS HighPerf module

But before I do, a brief history leading to its creation.

Once upon a time...
Business Case: Fujitsu Racing V8 Supercars Team Website

Complex site comprising many components:

- News
- Events
- Merchandise
- Image Gallery
- Video Gallery
- Form
- Edit areas
- RSS feeds
The client - delivery requirements

- Trackside update of news, image and video gallery
- Fine-grained user permissions
- Sub-2 second initial browser display - HTML delivery
- Sub 8 second until completion time
- Performance to be delivered at up to 100 concurrent users
- Delivered yesterday

Heard that before - OpenCMS is my choice, again, however...
OpenCMS - highly dynamic software

*With great power comes great responsibility...and system requirements*

- A rule of thumb - the more flexible the software, the slower it will be
  - More calls to data sources
    - Local - datastore - low latency
    - Remote - RSS feeds, web services - high latency
  - Data aggregation and formatting requires additional steps
- The slower a system, the less it will scale as existing resources are unavailable for longer
How we made OpenCMS lightning fast

Decision criteria:

• Local audience - Content Delivery Network (CDN) == overkill
• Time constraint - due in 2 weeks
• Limited budget - more hardware not an option

We chose to implement a reverse proxy cache/HTTP cache in front of our Tomcat + Apache web stack.
What is a reverse proxy cache/HTTP Cache?

- HTTP Cache sits in front of traditional HTTPD server or the application server taking over as the client connection point
- Populates cache from backend HTTPD or application server directly
- Serves cached pages from memory - optimised
- more efficient with system resource use for each connection - more connections for the same resource usage
Typical OpenCMS Stack

- Customer Web Browser
- Apache HTTPD
- Apache Tomcat
- Database

Typical OpenCMS Stack with Varnish HTTP Cache

- Customer Web Browser
- Varnish HTTP Cache
- Apache HTTPD
- Apache Tomcat
- Database
Why choose a HTTP Cache?

- Non-invasive - no code changes
- Field tested - proven technology
- Works with static and dynamic content
- Quick to implement - 30 minutes setup time
- FAST - It is much more efficient to serve pages from optimised cache memory without web or application server overheads
- Efficient - increased system capacity due to efficiencies gained from bypassing web and application server overheads
- Potential for reduced hardware requirements - $ savings
Varnish Cache - HTTP cache extraordinaire

- Commercially-backed Free Open-Source Software (FOSS)
- Optimised HTTP cache, not jack of all trades
- Highly configurable via internal domain configuration language - VCL
- VCL supports RegEx., inline C
- Multiple backend support with programmable routing using VCL
- No SSL support
- Supports Edge-Side Includes (ESI) - more on this later
Varnished OpenCMS Fujitsu Racing - the numbers

- Simple indicative test - not real-world
- UAT system
- LAN-based test - not real-world
- Synthetic benchmarks using Apache Bench
- `ab -c 100 -n 1000`
- Test scenario - 100 concurrent users repeated 1000 times
Supercharging...start your engines!

- **Varnish**:
  - Pages/sec: 4,026
  - Delivery Time (ms): 60

- **Apache**:
  - Pages/sec: 10,063
  - Delivery Time (ms): 23

- **Tomcat**:
  - Delivery Time (ms): 14

**Graph Legend**
- Blue: pages/sec
- Green: Delivery Time (ms)
So we have our speed, but now we need to pay the price - content freshness - dynamism

- HTTP cache is unaware of when backend content is updated
- Content refresh is time-based
- Different content types or paths can have different expiry times. Vendor-dependent implementation and capabilities
How we made OpenCMS lightning fast - *without* sacrificing dynamism

**OpenCMS HighPerf - PURGE on publish**

- Vendor independent HTTP PURGE request type. Tells HTTP cache to purge requested resource from cache

- Triggered on publish event

- Next resource request results in a cache miss - HTTP cache will request and return fresh resource from web/application server

- Enables HTTP cache benefits without sacrificing content freshness with time-based purge/refresh policies.
  
  - Memory-based content lookup and delivery - fast
  
  - Optimised system resource use
  
  - Increased system capacity
OpenCMS HighPerf information

- OpenCMS self-contained module including all dependencies
- LGPL license
- Source code contained within module including all dependency modifications
- Self-signed certificates need to be added to local JSSE certificates store
- Extra parameter to tell OpenCMS HighPerf to allow self-signed certificates - not recommended for production environments
  - Дau.com.melbournebusinessonline.opencms.publish.allowSelfSigned
Cluster awareness

• Enable purging to multiple backends behind a single domain
  • We can achieve this using the OpenCMS site aliases. Limitations?

• Purge piping via a single HTTP cache in front of a cluster of HTTP caches?
Future Directions - Edge Side Includes (ESI)

ESI - Edge Side Includes

- Ratified HTTP cache specification - vendor independent
- Enables HTTP cache to cache individual page components
- assemble page from components
- Only purge components that have changed
- Fine granularity compared to page-level caching
- Supports compression
Future Directions - Edge Side Includes (ESI) - page components

- Header
- Navigation
- Race schedule
- Advertising
- Video Gallery
- News
- Merchandise
- Driver Info
- Series Standings
- Newsletter Form
- Events
- Image Gallery
- Footer
Future Directions - Edge Side Includes (ESI) - Implementation

Initial investigation has revealed the following implementation methods:

- Implement via template mechanism
  - Potentially no OpenCMS changes required
  - Requires explicit re-coding of site template to utilise ESI
  - Relies on ESI support or fails?

- Implement via OpenCMS FlexCache
  - Requires OpenCMS modifications
  - Does not rely on ESI enablement - only enables if ESI support present
  - Transparent enabling for all existing site templates

- Other methods - ADE?
Get OpenCMS HighPerf from the following URL. Be sure to check back for updates and future modules!

Questions & Answers, Feedback & Suggestions

• Dammian Miller

• Melbourne Business Online

Thank you for attending