Drupal Performance for the Rest of Us

Eric Peterson (iamEAP)

Goals

1. Understand caching and how Drupal does it.

2. Get comfortable configuring Drupal for performance.

3. Learn a subset of performance modules:
   - What they do and when to use them
   - How to configure them
   - *See* the impact

Why trust this guy?

- My day job is to work on a site that supports over 1.5 million pageviews per month.
- My company's IT department has enough on its plate besides this "Drupal" thing.
- As a result, we've been forced to make our Drupal installation scale and perform without the help of Varnish or Memcache.
- You can always trust a handsome man.

So what exactly is caching?

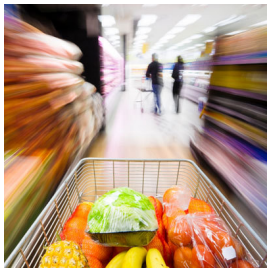
Caching is when you save off data that was expensive to obtain so that subsequent calls for that data are less expensive.

There are essentially three ways to configure:

1. What are you caching?
2. Where are you storing it?
3. How long are you storing it?

The obligatory contrived example

Your in-laws are staying for the month and you're out of groceries. They're also on different and incredibly stupid diets.



Everything is a cache within a cache

- DNS cache
  - Browser cache
    - Page cache
      - Block cache
      - Views cache
        - Query result cache
        - Output cache
          - MySQL query cache
          - PHP static caching
            - Opcode caching

What's your Drupal cache strategy?

Primarily Anonymous Traffic

Focus on ways to cache each page as a whole:

- Page Cache
- External Cache (for browsers)
- Boost

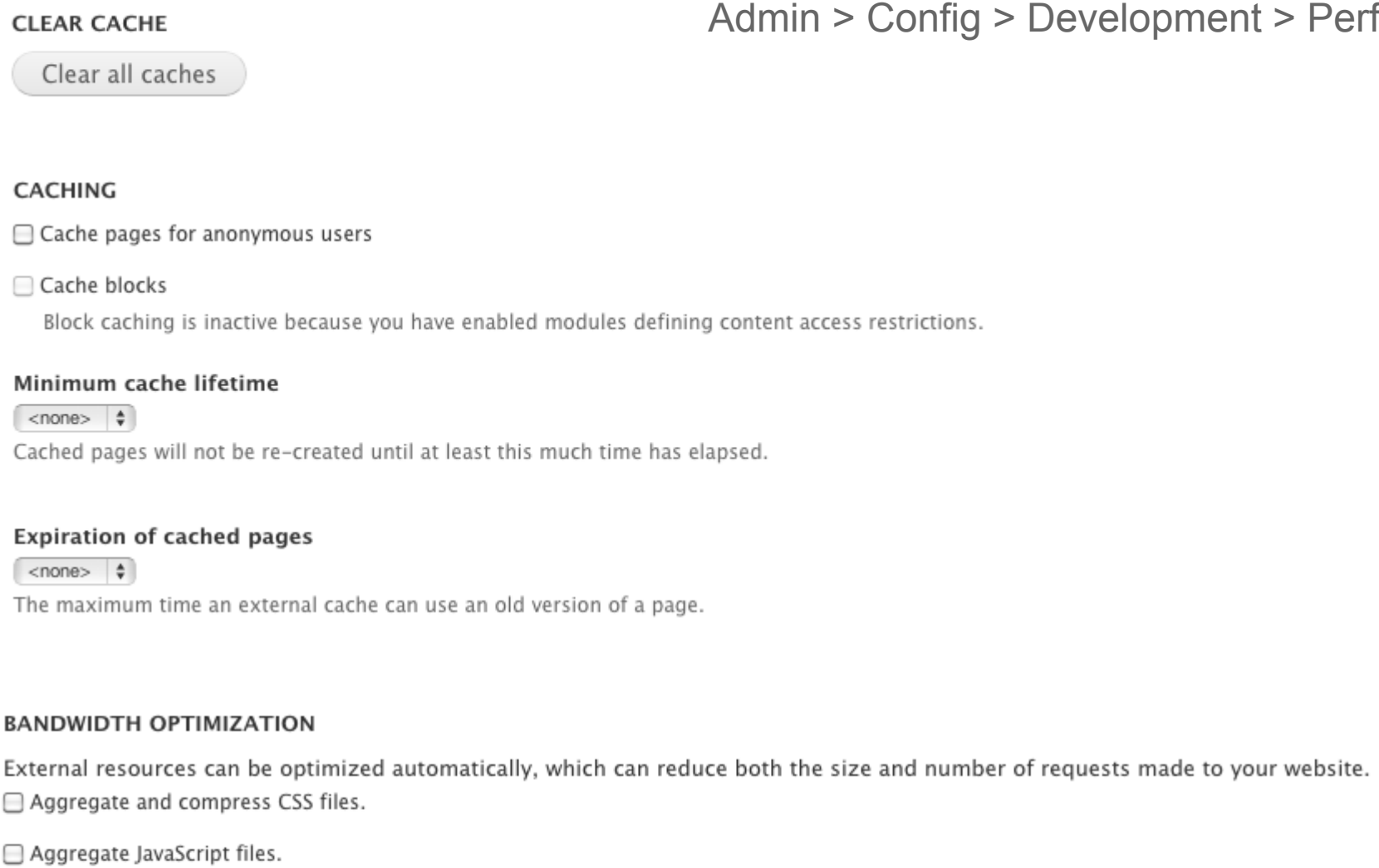
Primarily Authenticated Traffic

Focus on caching individual components of a page:

- Blocks
- Views
- Panels
- Entities

Drupal performance configuration

Admin > Config > Development > Performance



Cache pages for anonymous users

This checkbox does exactly what it claims with only a couple of sensible caveats:

- If a user just POSTed form data, the page is not cached.
- If a module stores data for anonymous users in the session, pages will not be cached.

Bonus: you can disable page cache hook invocations by setting the following in your settings.php file.

```
$conf['page_cache_invoke_hooks'] = FALSE;
```

Caution: Some modules depend on this functionality. Configure with care. Those familiar with D6: this is the same as "aggresive" caching.

Minimum cache lifetime

The most misunderstood configuration, maybe in all of Drupal.

> *"Cached pages will not be re-created until at least this much time has elapsed."*

This is something of a half truth:

- True, this is not an explicit expiration time; it is a *minimum* time period during which the cache will not be deleted.
- However, it does not guarantee that any individual cache entry lives as long as you select. We'll explain this in a bit.

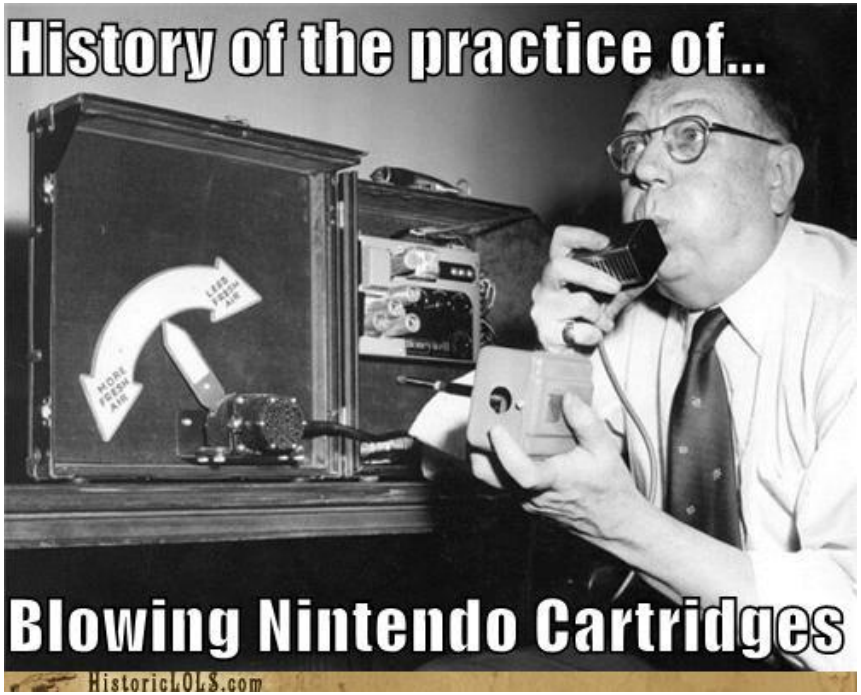
Expiration of cached pages

This setting relates only to how systems *external* to Drupal may cache your page. Don't confuse this with minimum cache lifetime.

Think of this as the length of time a user's browser is allowed to hang on to a copy of your page before requesting another copy from your server.

As your scalability needs increase, this setting is also what you use to let Varnish and other reverse proxies know how long it should hold on to the same page.

Clear all caches



This button does exactly what it claims. Use with caution!

We'll discuss alternatives and augmentations later.

Cache garbage collection

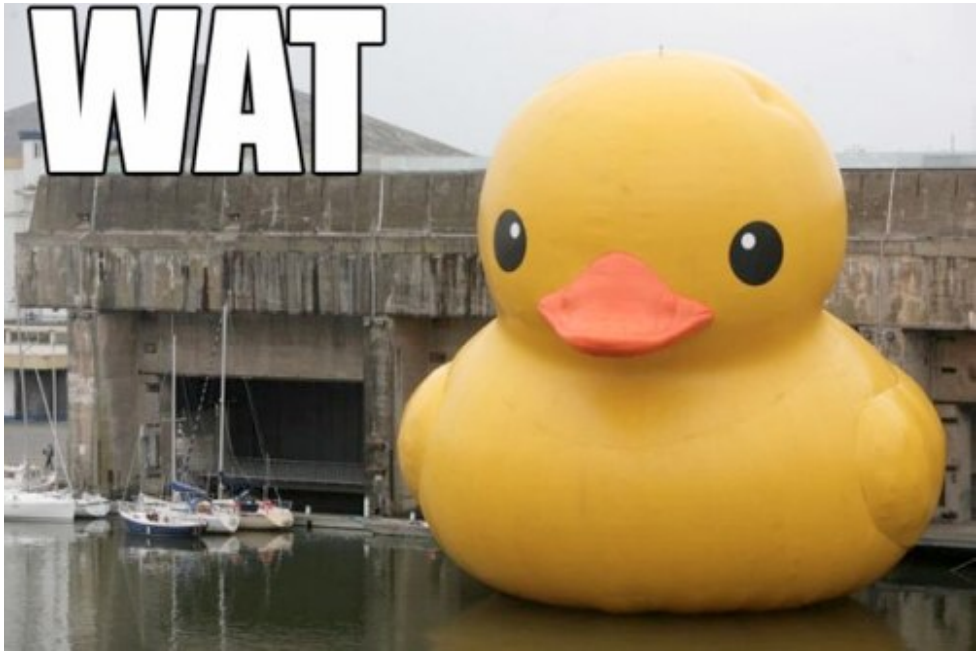
If you specify a *minimum* cache lifetime instead of an explicit expiration time, how do those cache entries ever expire?

Answer: Cron!

- Cache entries that adhere to your minimum lifetime configuration are always marked *temporary*.
- Cron clears all *temporary* cache entries no more than once every minimum cache lifetime interval.

Caveat emptor: saving content

Anytime you save a node, Drupal performs the same garbage collection procedure that cron does (except just on page and block caches).



Unexpected consequences

When minimum cache lifetime = none

- Every time cron is run, your entire cache is emptied.
- Anytime anyone adds or updates any piece of content, your entire page and block caches are emptied.

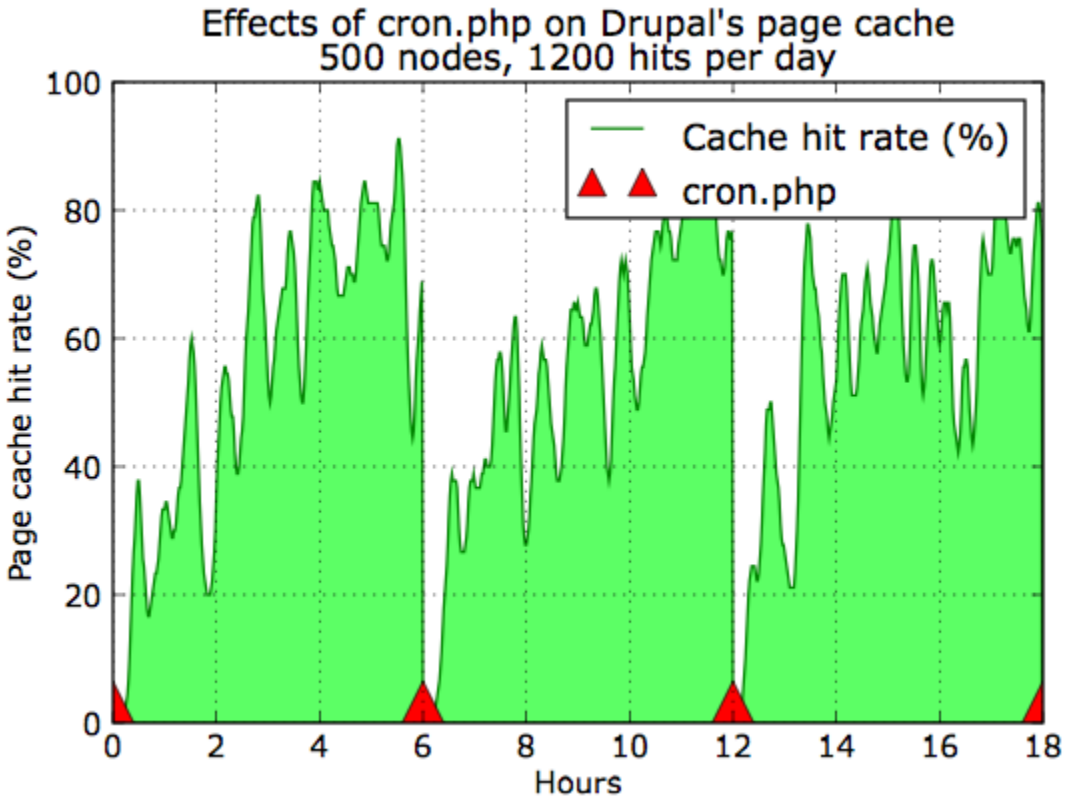
For the love of god, never under any circumstances keep your minimum cache lifetime set to "none."

When minimum cache lifetime > cron interval

- In practice, your minimum cache lifetime is a maximum.

Unexpected consequences (cont.)

Because cache garbage collection is bin-based rather than entry-based, garbage collection is expensive.



Via Dylan Tack at Metaltoad Media

Recommendations

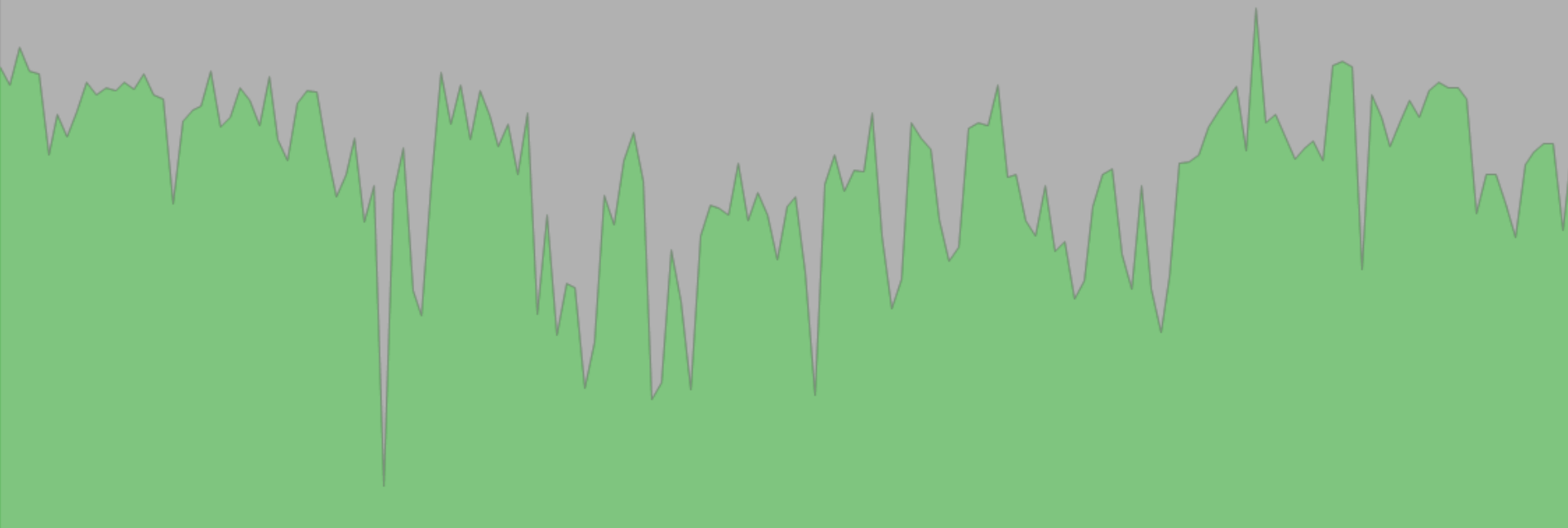
- If at all possible, run cron less often.
- If that's not possible, increase minimum cache lifetime as much as is reasonable.
- What's "reasonable" depends on your site.
  - The more often content is updated or added, the smaller you want your minimum lifetime (for editor UX).
  - The same applies for comments if you're using the core commenting system.

- The above not good enough?
  - Alternative Database Cache
  - Finer control over cache expiration

Alternative Database Cache (ADBC)

http://www.drupal.org/project/adbc

The ADBC module attempts to make garbage collection less traumatic by clearing cache bins on an entry-by-entry basis, rather than all at once.



Installing & configuring ADBC

1. Enable the module like any other module.
2. Add the following code to settings.php:

```
$conf['cache_backends'][] = 'sites/all/modules/adbc/adbc.cache.inc';
$conf['cache_default_class'] = 'AlternativeDrupalDatabaseCache';
```

No additional configuration is necessary beyond how you would normally configure Drupal for performance.

The result will be increased overall cache hit rate.

Finer control over cache expiration

Another strategy is to increase cache minimum lifetime abnormally high (12-24 hours), and create a smarter cache invalidation system.

Two modules are well suited for this particular use-case in different ways:

- Cache Actions (http://www.drupal.org/project/cache_actions)
- Flush Page Cache (http://www.drupal.org/project/flush_page_cache)

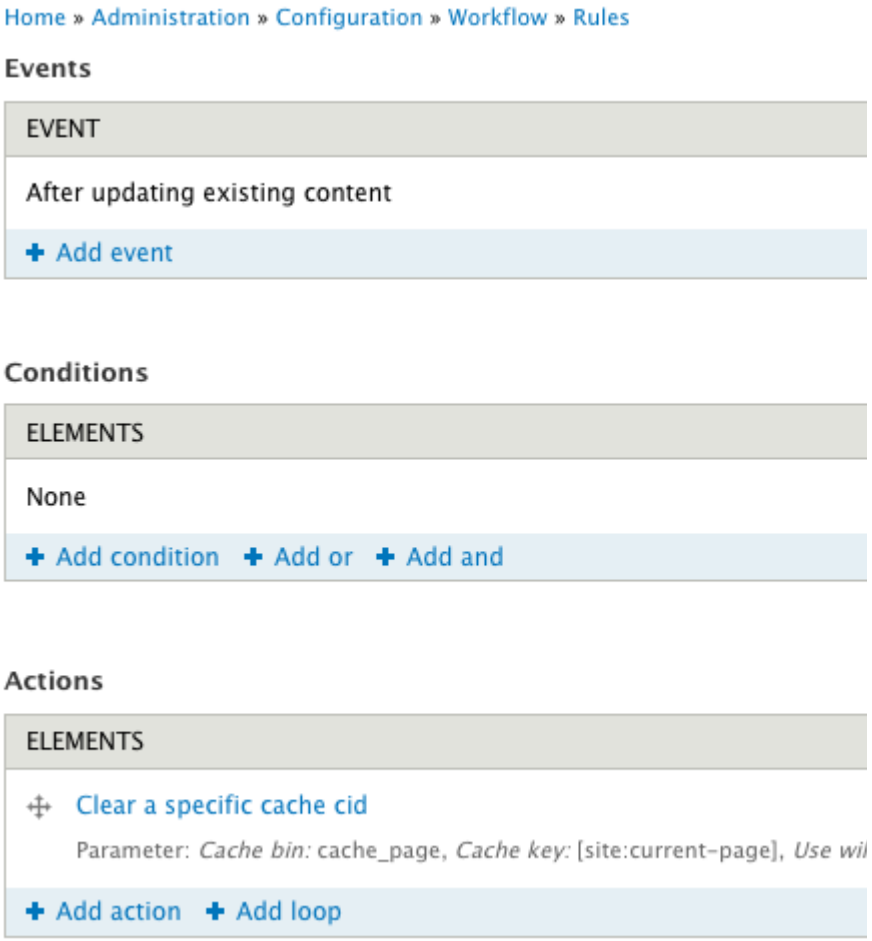
Cache Actions

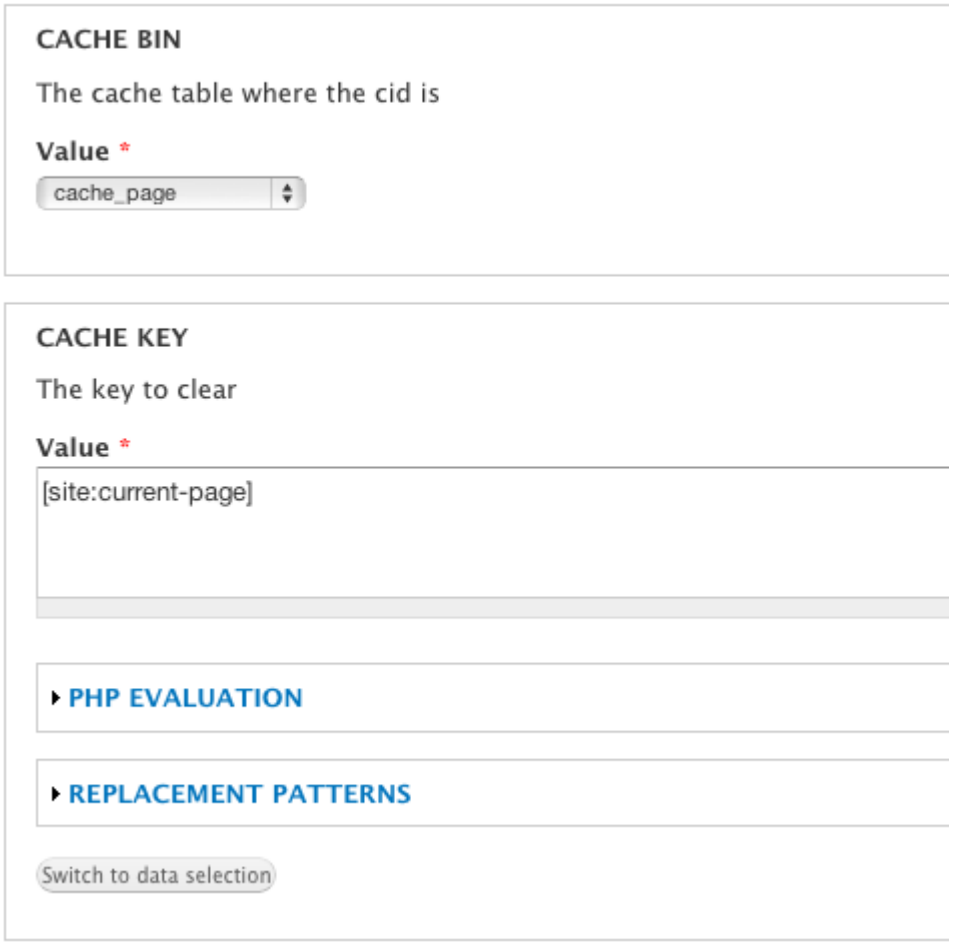
Cache Actions integrates with the Rules module by introducing cache-related actions that can be triggered on any regular Rules event.

Some common use-cases:

- Clear the page cache for the current page when an editor makes content modifications.
- Clear the page cache for the current blog post when a user adds a comment.
- Clear the views cache for a view after re-ordered by an editor.

Configuring Cache Actions





On-demand cache expiration

While Cache Actions is great for general cache clear rules, it quickly becomes cumbersome and difficult as events become more obscure and cache items become more specific.

One way to address these situations is with the Flush Page Cache module.

Flush Page Cache gives you and your editors a button on each page that invalidates every cache loaded on the page where it was clicked.

Configuring Flush Page Cache

1. Once the module is installed, add the following to your settings.php:

```
$conf['cache_backends'][] = 'sites/all/modules/flush_page_cache/flush_page_cache.cache.inc';
$conf['cache_default_class'] = FlushPageCacheDrupalDatabaseCache;
```

2. You'll need to place the button the module generates on every page and limit it to trusted roles. This can be done by placing a block or enabling the footer link the module optionally provides.

3. You may also want to style the button to a fixed position on the page.

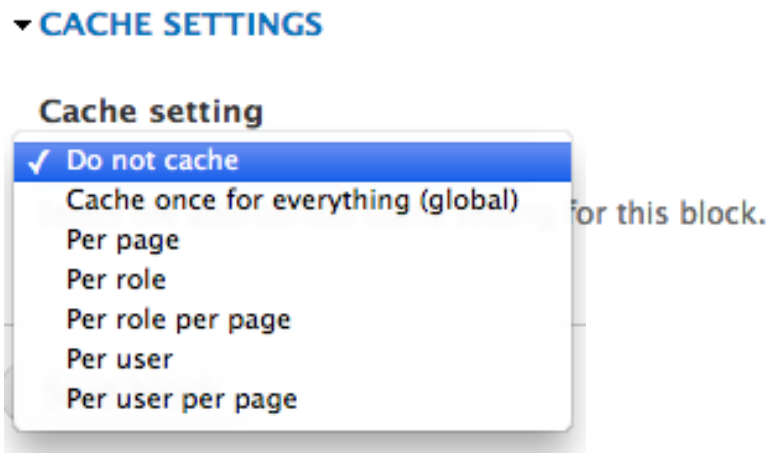
Caching more

Now that we've covered Drupal's cache idiosyncrasies and smarter expiration techniques, let's dive into specifics...

Module: Block Cache Alter

Drupal 7 introduced an API for different cache levels.

Block Cache alter provides a GUI to manage those cache levels on a block-by-block basis.



Caveat emptor: Blocks via Context

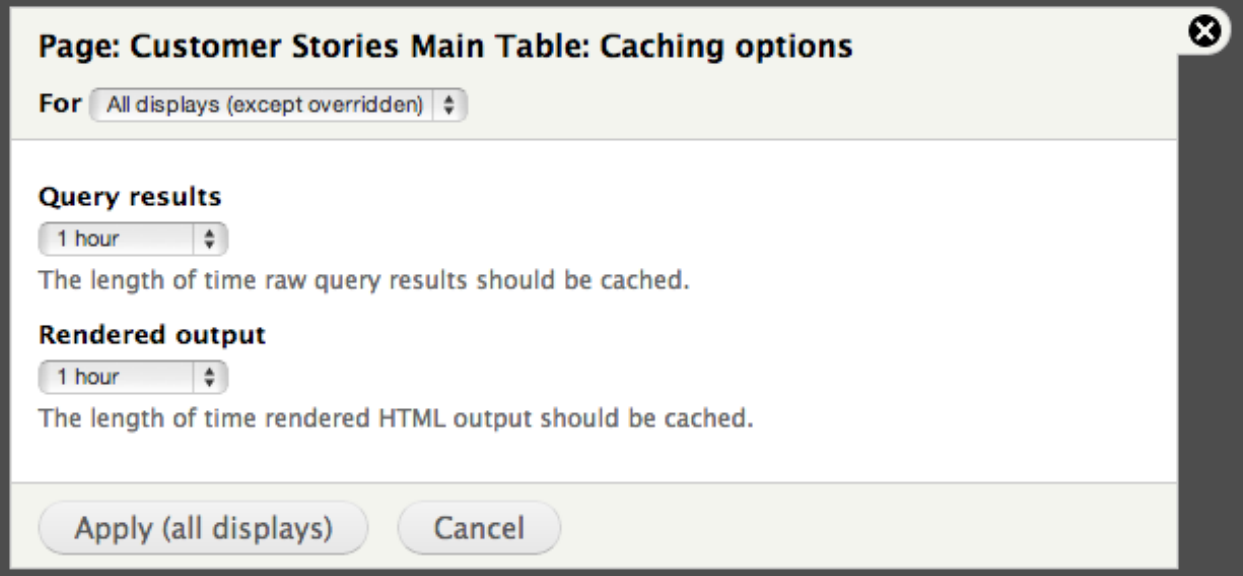
If you use context to place your blocks, they're not necessarily run through the normal Block API.

There's a module for D6, but the D7 port is still in dev.

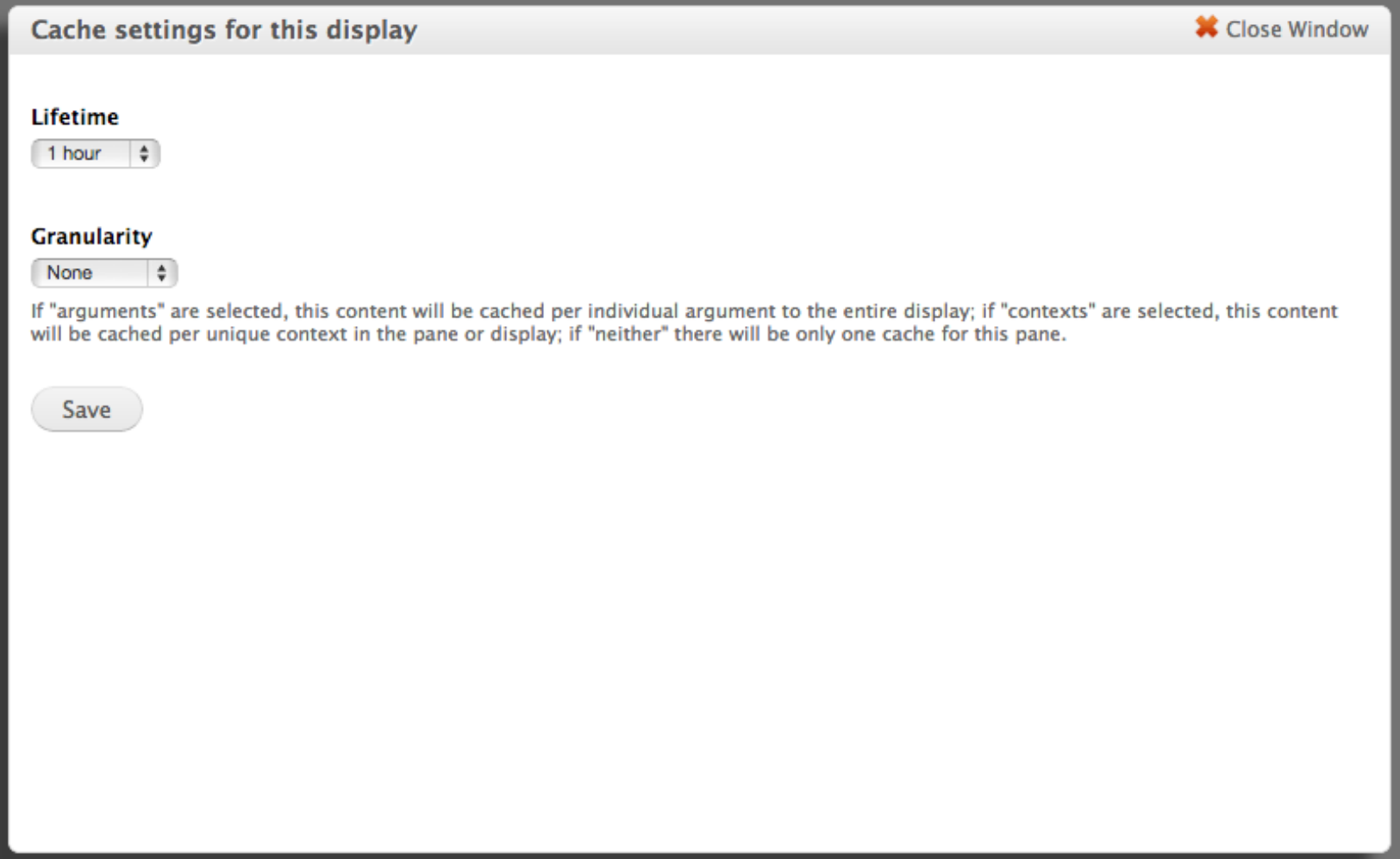
Context Block Cache Alter

http://drupal.org/project/context_blockcache_alter

Caching Views



Caching Panels



Caching Entities

Nodes! Comments! Taxonomies! Files! Users! Oh my!

Occasionally, we need to load these things.

The **Entity Cache** module provides caching for the above core entities.

Installation is extremely simple. Just download and install.

In practice, only really beneficial on frequently accessed pages that load large numbers of entities.

Storing cache entries as files

Performing a database query requires a certain amount of overhead.

Reading a file is an extremely fast operation. Why not store cache entries as files rather than in the database?

The **File Cache** module opens up this possibility.

Install the module, then add the following to settings.php:

```
$conf['cache_backends'][] = 'sites/all/modules/filecache/filecache.inc';
$conf['cache_default_class'] = DrupalFileCache;
```

Caching pages to HTML (Boost)

How is this different from File Cache?

File cache switches all (or select) cache storage to be file-backed. Boost saves the final rendered page markup to an HTML file, then subsequently serves the HTML file.

File cache will still boot PHP and Drupal on each request. Boost avoids PHP completely once the page is saved off.

Installing & configuring Boost

Somewhat more involved than most module installations.

1. Install and enable the module as usual.
2. Disable core page cache.
3. Ensure the cache directory is writeable by the web server.
4. Configure Apache with rules generated by Boost (.htaccess).

Details available at http://drupal.org/node/1459690

Interactive Performance Comparisons

Jump over to Tableau Public!

http://public.tableausoftware.com/shared/G76NWKNK7

Questions


We don't have the money, what
I have a couple features u
Should I do my taxes before, during or after pro
Will my production insurance
a hard time understanding thi
flipping. We don't h
how do I navigate the film festiva
Should I try to

Thanks!

Eric Peterson

E-mail	epeterson@tableausoftware.com
Handle	iamEAP
WWW	http://www.asmallwebfirm.net

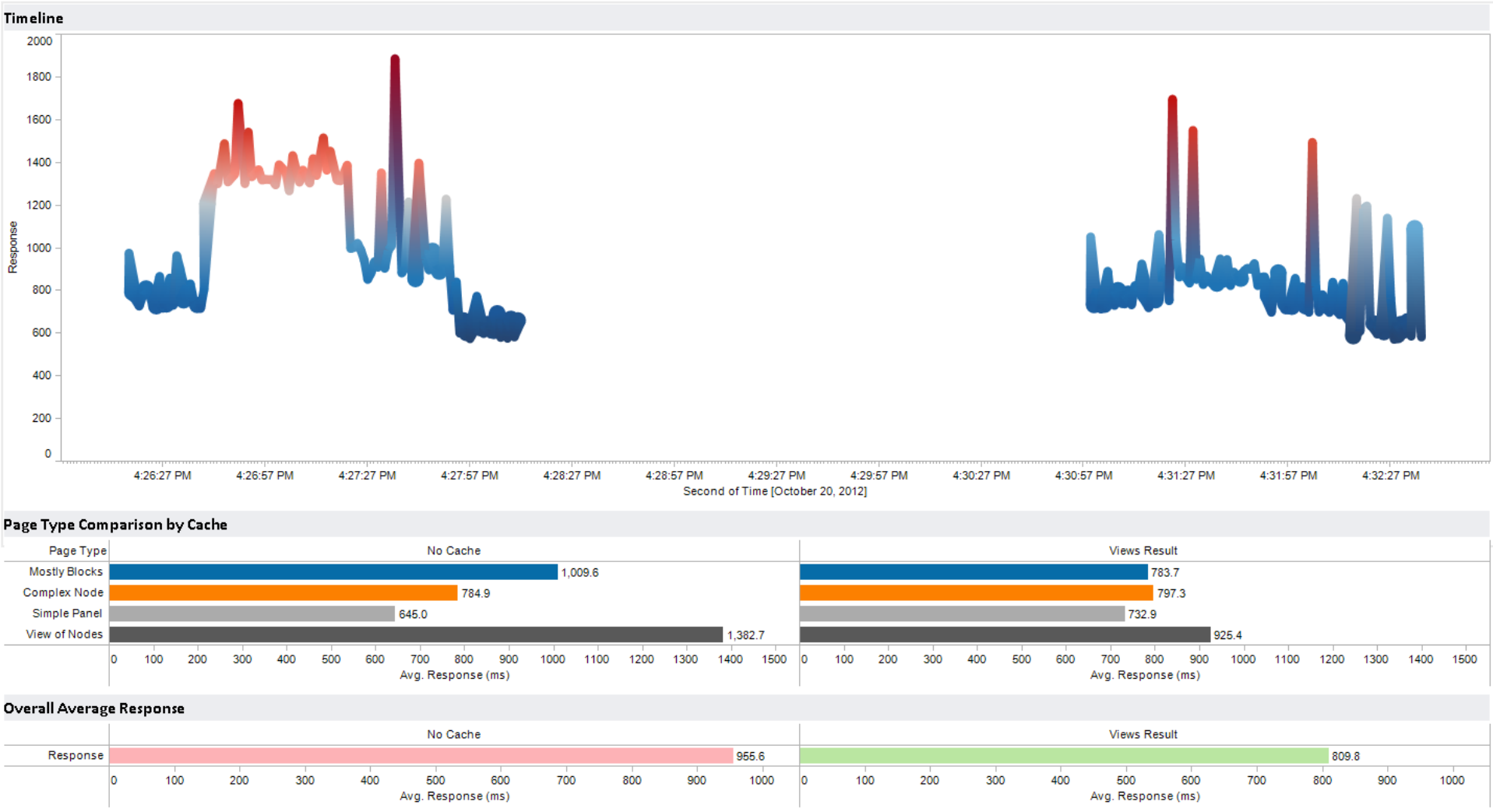
Benchmarking Core block cache

http://public.tableausoftware.com/shared/67HGQMF3B

Benchmarking Views cache (output)

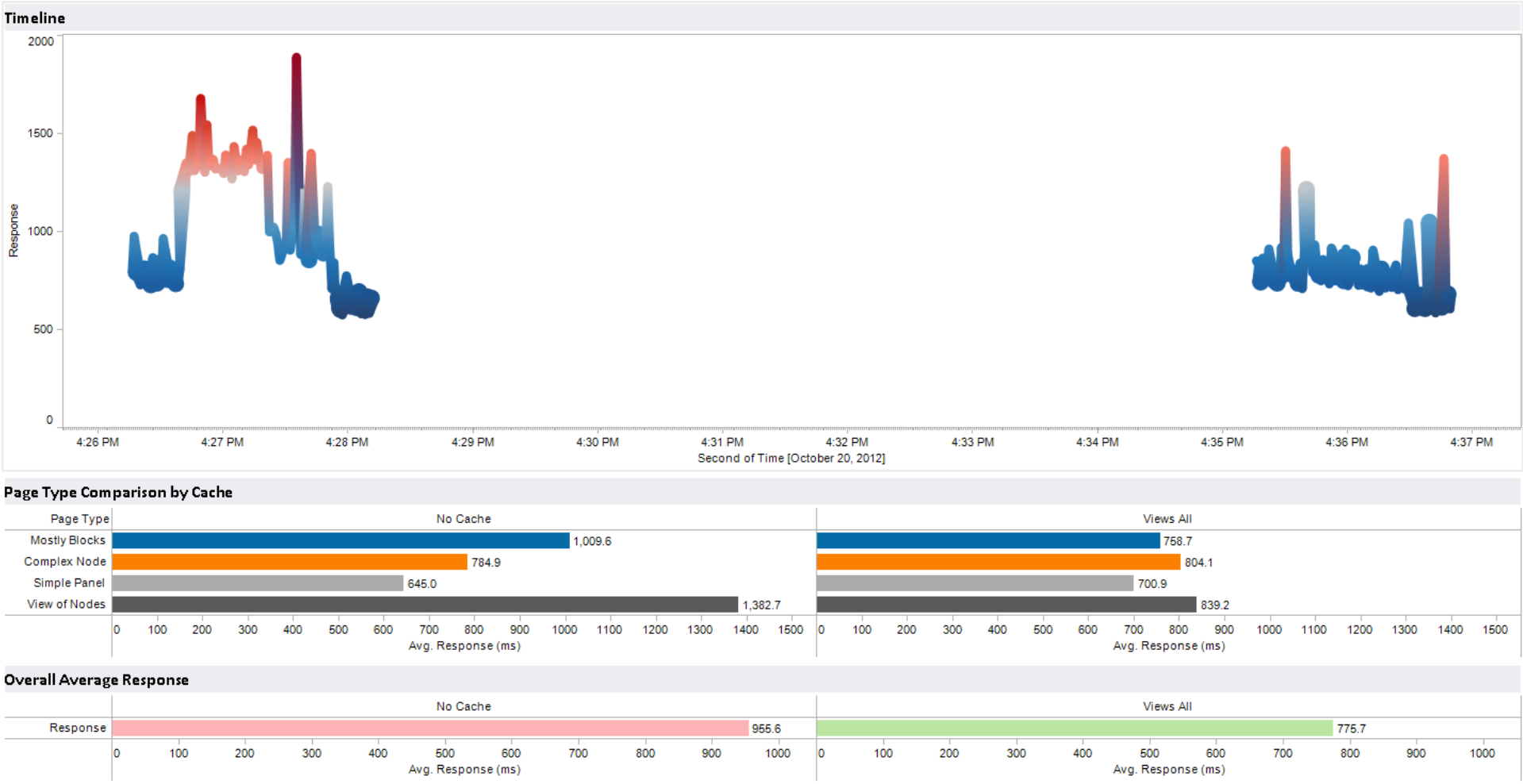
http://public.tableausoftware.com/shared/6N8J54XR9

Benchmarking Views cache (results)



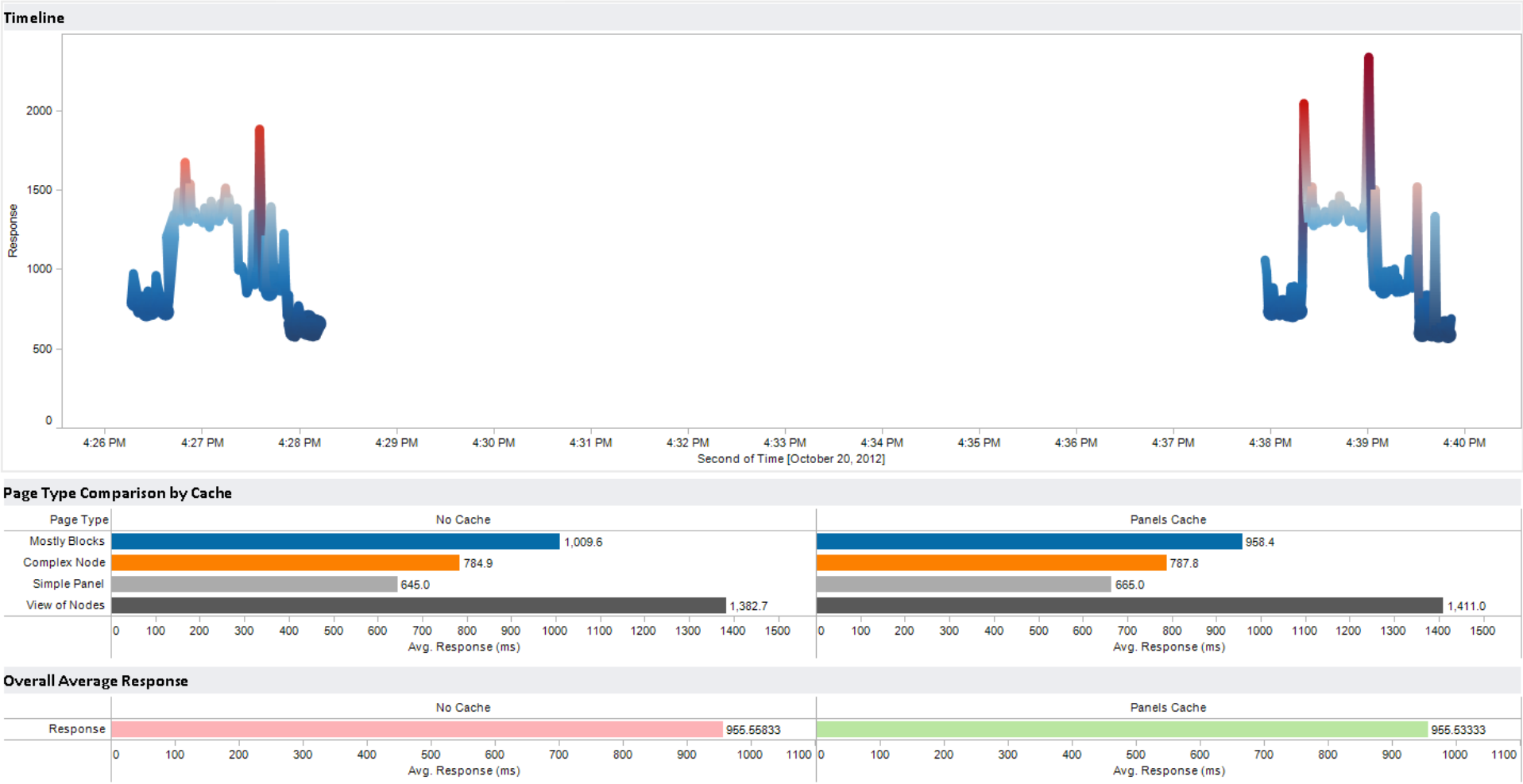
http://public.tableausoftware.com/shared/2YPB56GRM

Benchmarking Views cache (both)



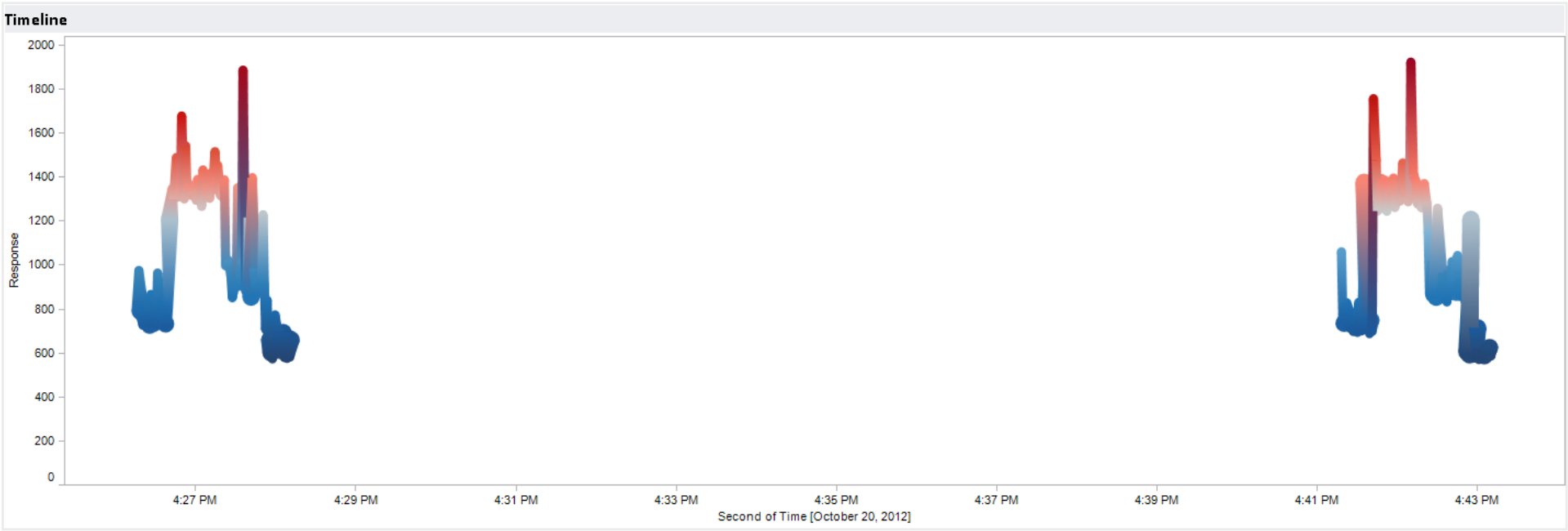
http://public.tableausoftware.com/shared/GBXZ8XDGH

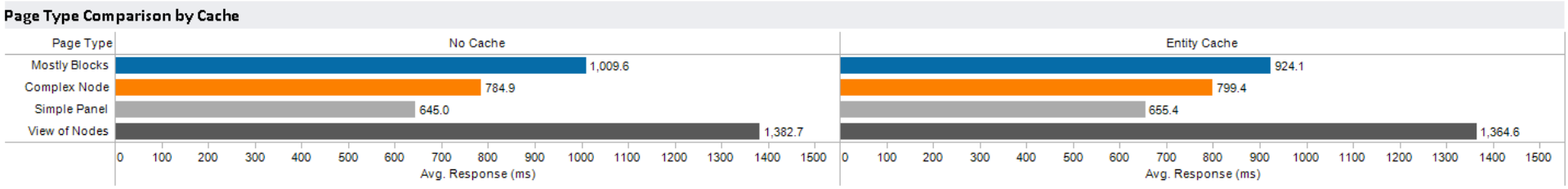
Benchmarking Panels cache

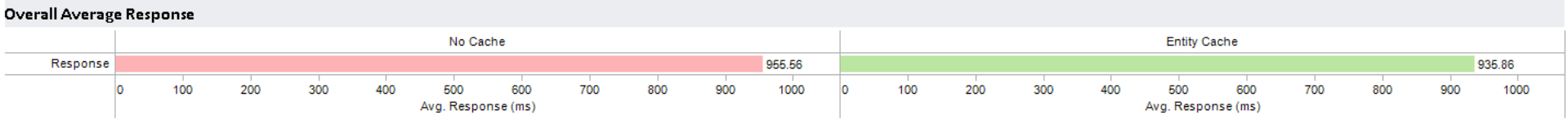


http://public.tableausoftware.com/shared/4SY5PYQJD

Benchmarking Entity Cache

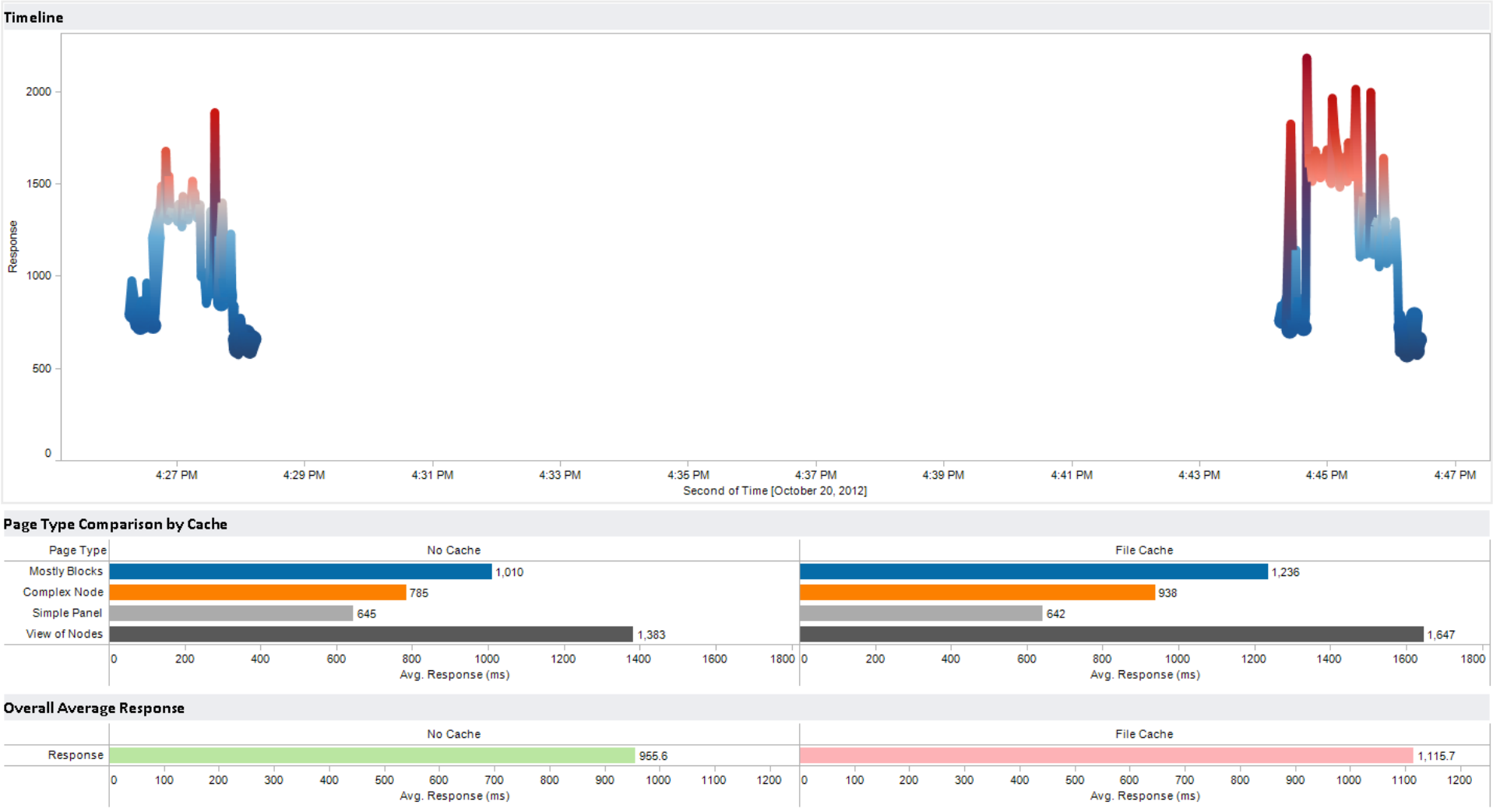






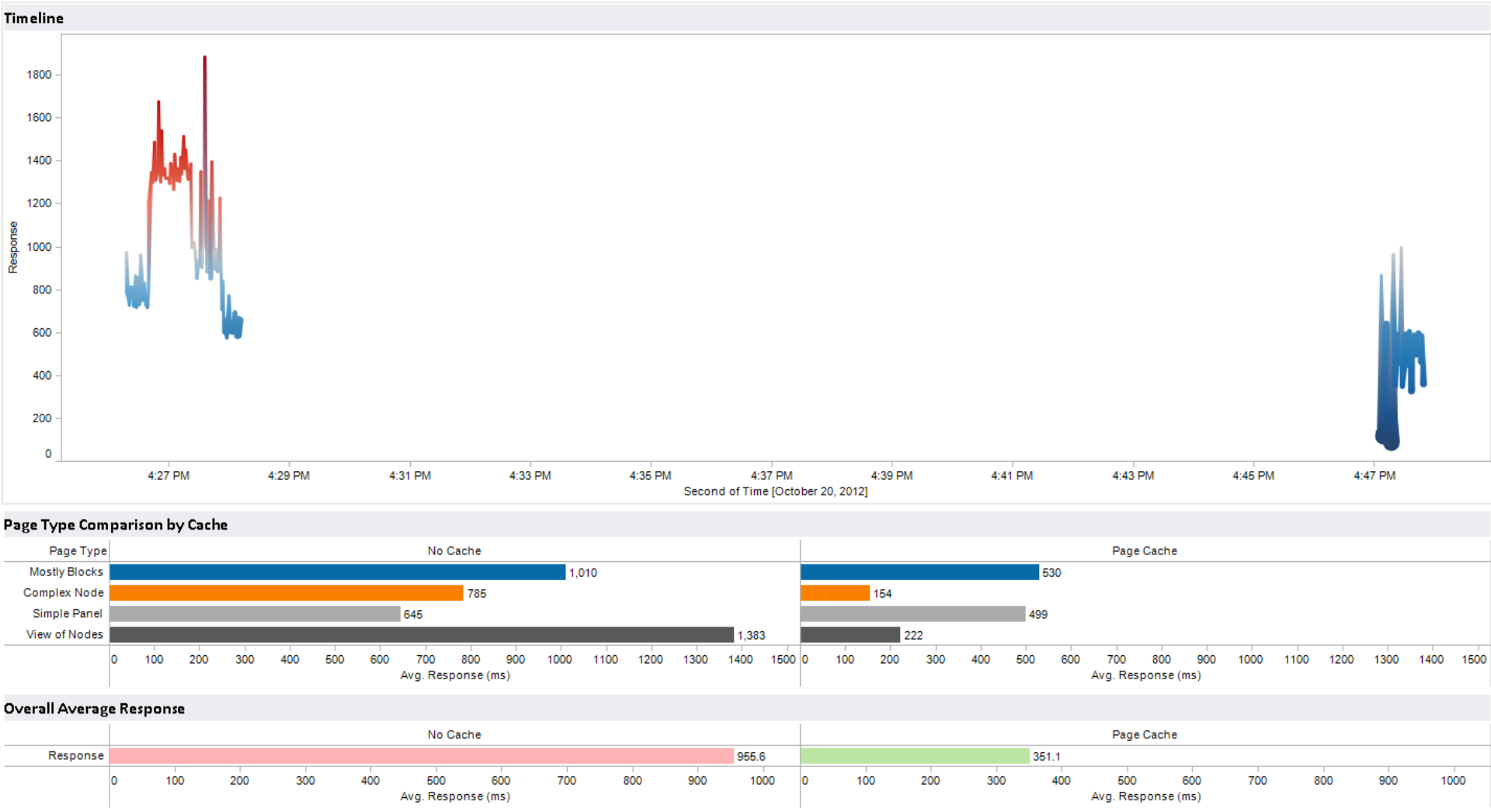
http://public.tableausoftware.com/shared/KWXTS49F5

Benchmarking File Cache



http://public.tableausoftware.com/shared/KZC37FY4R

Benchmarking Core page cache

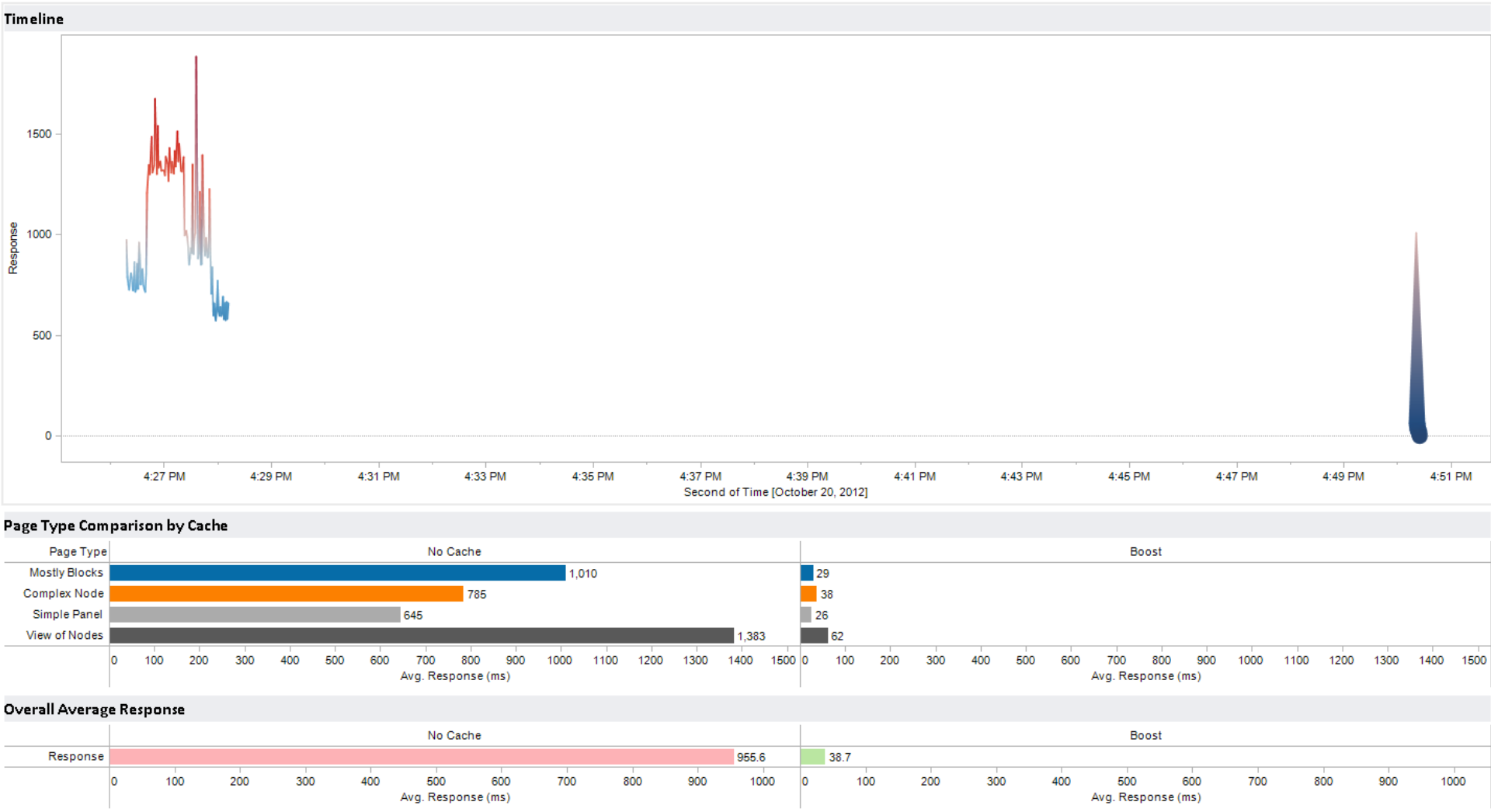


http://public.tableausoftware.com/shared/8YN72F52S

Page cache with File Cache enabled

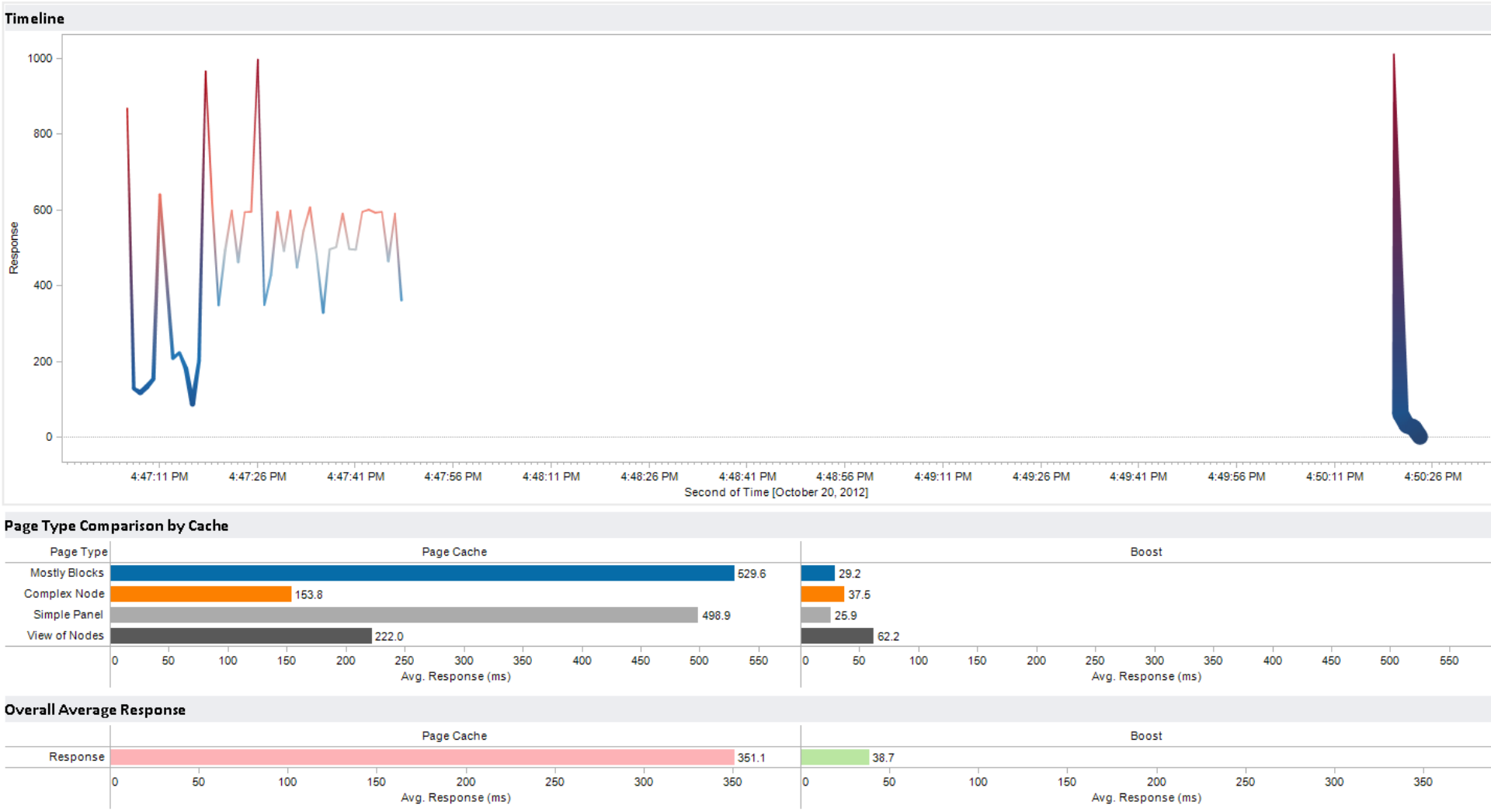
http://public.tableausoftware.com/shared/8WH7ZTFCW

Benchmarking Boost



http://public.tableausoftware.com/shared/S4F6FDQX4

Boost vs. Page Cache



http://public.tableausoftware.com/shared/97Y4D7TSP