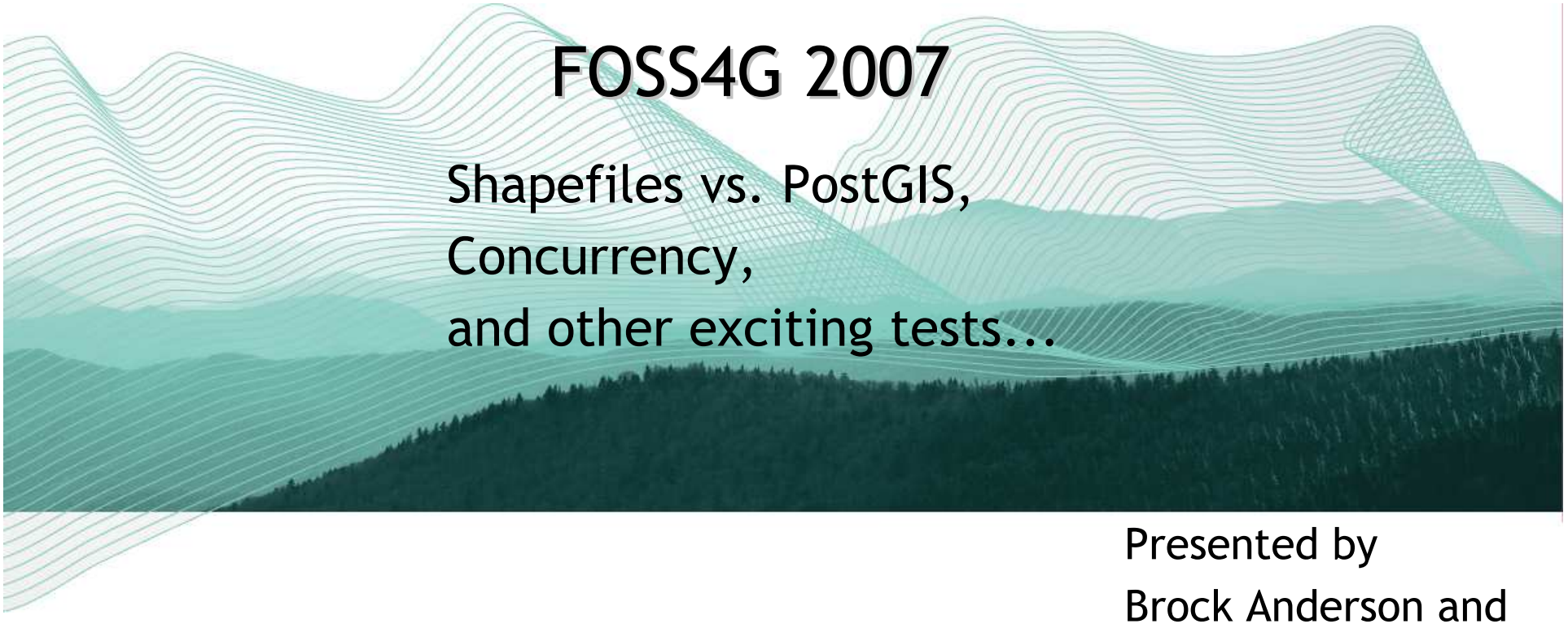
WMS Performance Tests!
Mapserver & Geoserver



FOSS4G 2007

Shapefiles vs. PostGIS,
Concurrency,
and other exciting tests...

Presented by
Brock Anderson and
Justin Deoliveira

Refractions RESEARCH

THE GEOSPATIAL EXPERTS

Presentation Outline

- Goals of testing.
- Quick review of WMS.
- Description of the test environment.
- Discussion of performance tests and results.
- Questions.

Goals

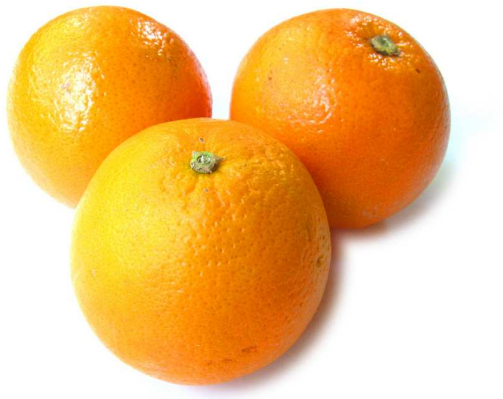
1. Compare performance of WMS GetMap requests in Mapserver and Geoserver.

2. Identify configuration settings that will improve performance.

3. Identify and fix inefficiencies in Geoserver.

* We do not test stability, usability, etc.,
We do not test styling or labelling.
We focus on vector input.

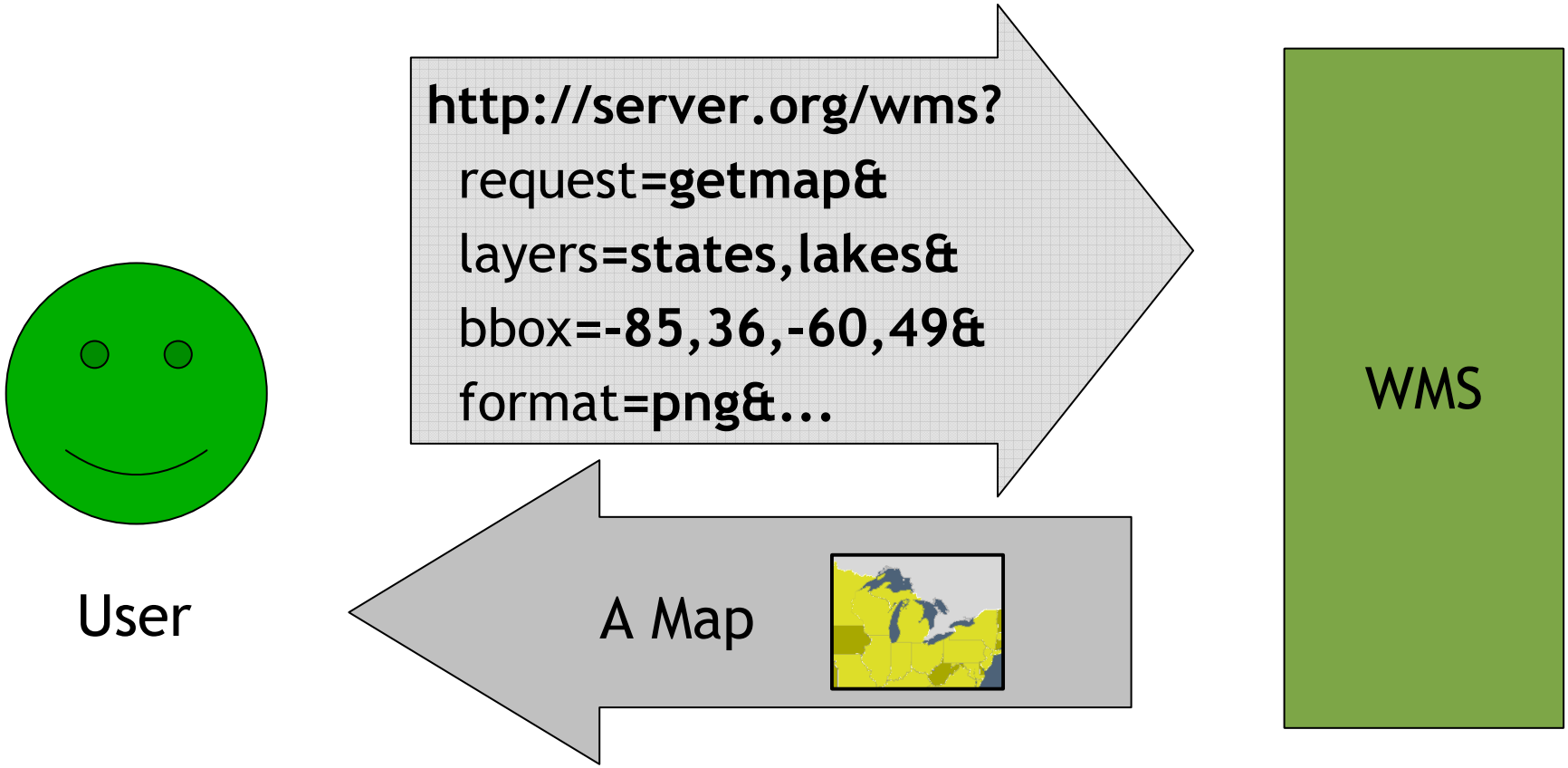
Keeping the tests fair



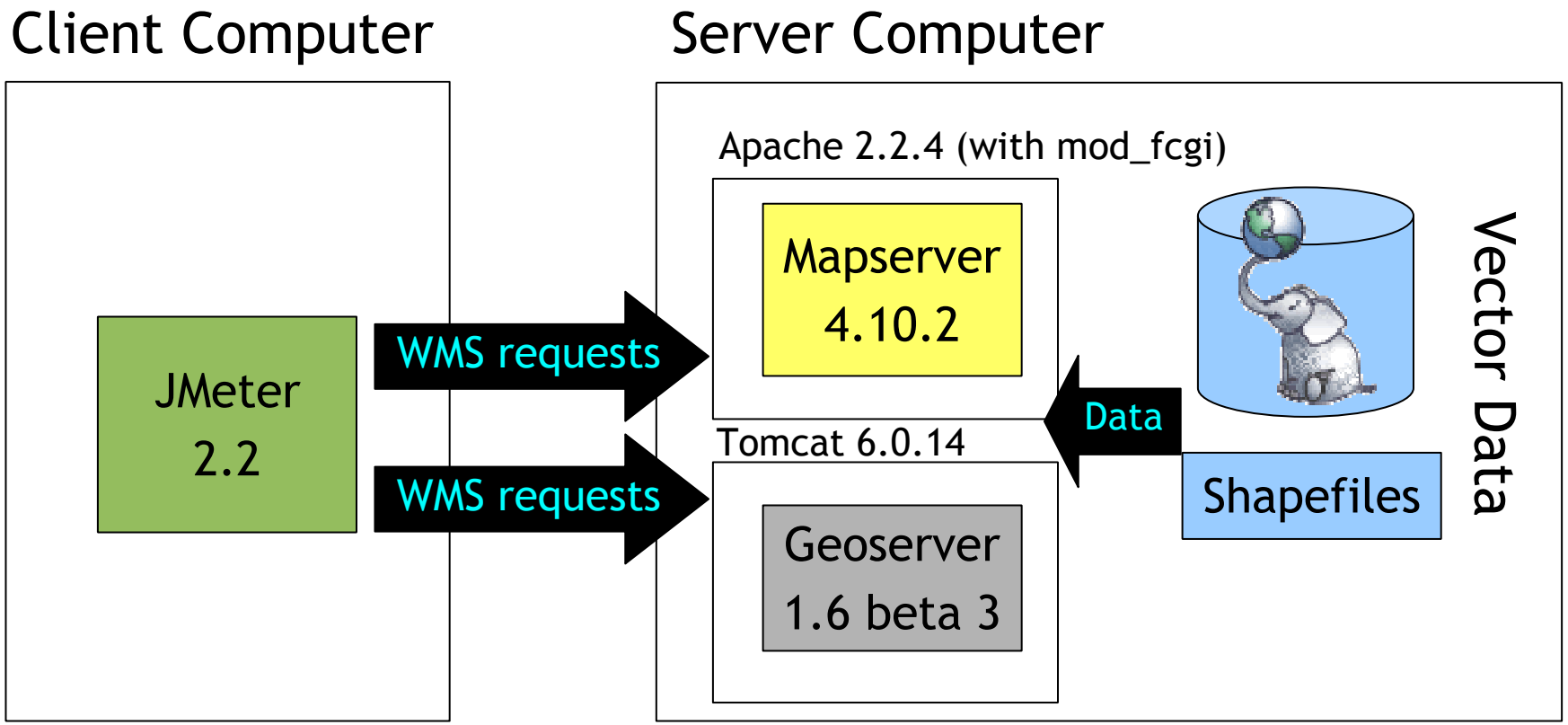
- **Not an easy job!**

- **We tried to understand what each server does** under the hood **to ensure we're not accidentally performing unnecessary processing on either server.**

Web Map Service (WMS)



Test Environment



Additional Server Specs: **Dual core (1.8Ghz per core). 2GB RAM. 7200RPM disk. Linux. PostgreSQL 8.2.4. PostGIS 1.2.**

Test #1: PostGIS vs. Shapefiles

- Two Data Sets:

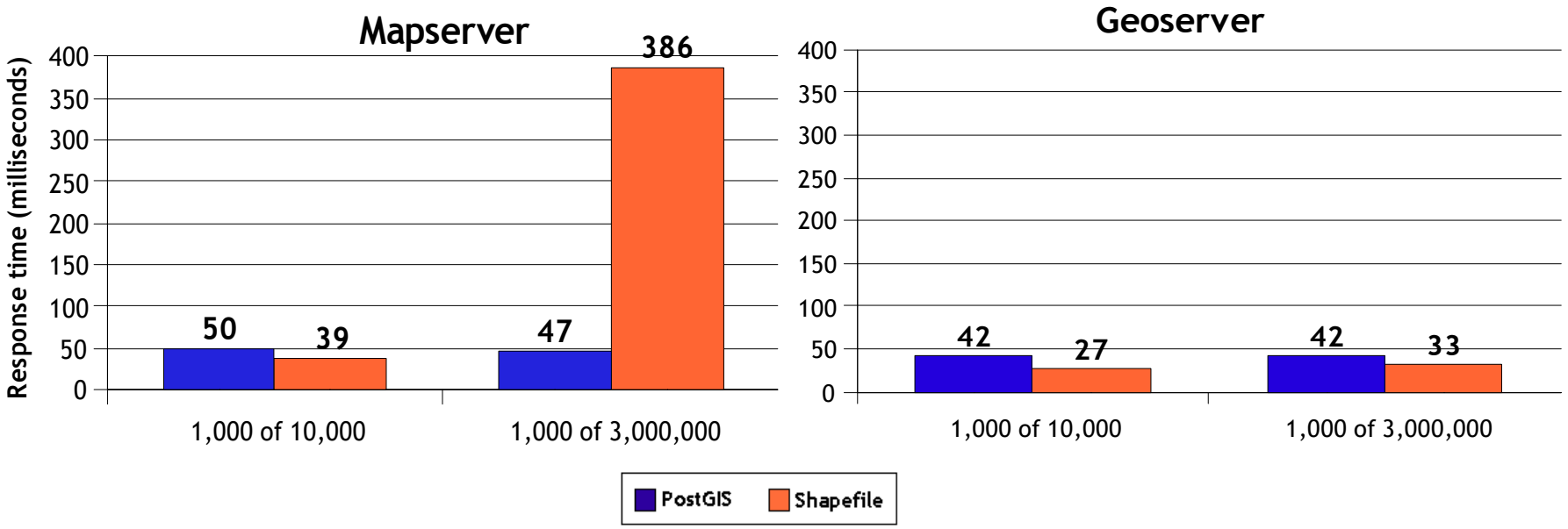
  3,000,000 Tiger roads in Texas

  10,000 Tiger roads in Dallas, Texas

- **Both data sets are in PostGIS and shapefile format.**
- **Spatial indexes on both data sets.**
- **Mapserver and Geoserver layers point at the data.**
- **Minimal styling.**
- **JMeter issues WMS requests to fetch ~1,000 features, limited by the 'bbox' parameter.**

And the results are...

Test #1: PostGIS vs. Shapefiles



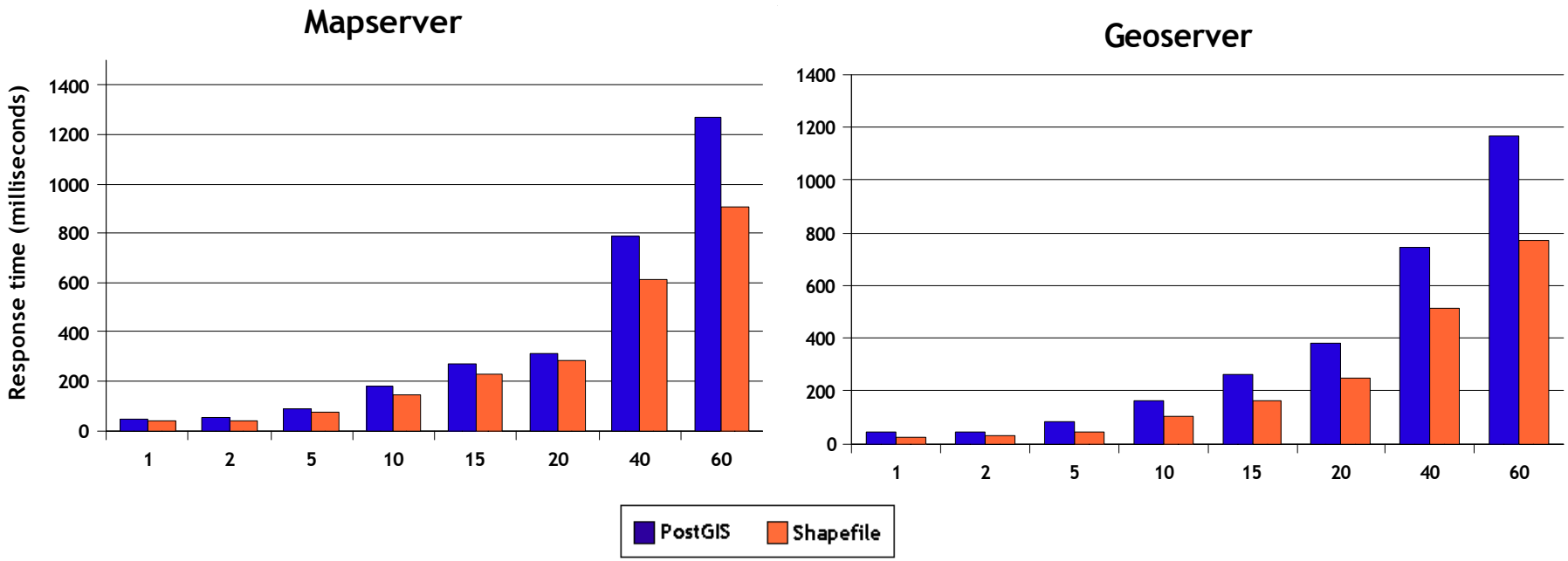
Notes: **This test uses two different data sets: one with 3 million features, the other with 10,000. Each bar is an average of 30 sample WMS requests, each using a different bounding box to fetch and draw appx. 1000 features (+/- 15%). The same 30 requests are executed for each scenario. One request at a time (no concurrency). Mapserver and Geoserver use the same data. Mapserver is using FastCGI via Apache/mod_fcgi. Spatial indexes on both data sets. Quadtree indexes generated by 'shptree'. No reprojection required. Minimal styling. Responses are 1-bit PNG images.**

Test #2: Concurrent Requests

- **Using the same tiger roads data set with 10,000 records.**
- **We issue multiple requests with pseudo-random BBOXes that fetch approximately 1,000 features.**
- **The main difference is that now we're issuing multiple concurrent requests.**

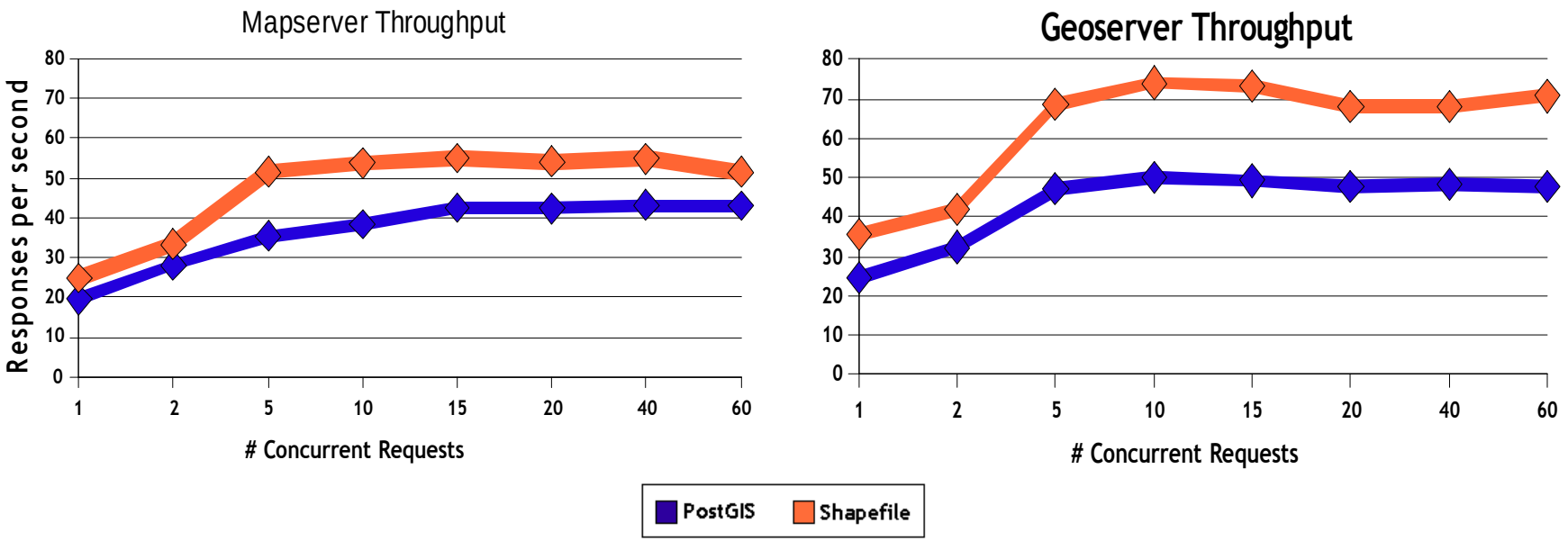
Let's see what happened...

Test #2: Concurrent Requests



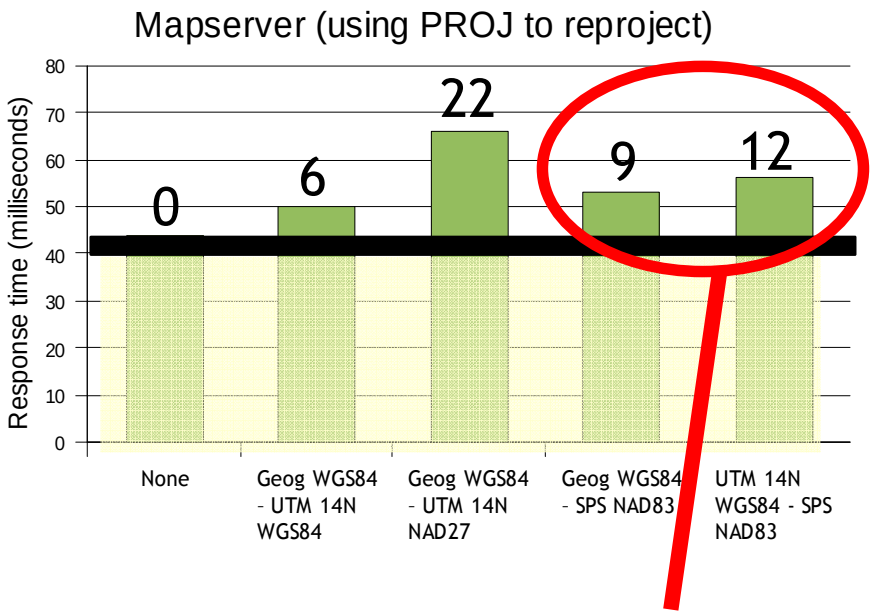
Notes: **Data in PostGIS and shapefile formats. Mapserver and Geoserver use the same data. Mapserver is using FastCGI via Apache/mod_fcgi. 20 FastCGI mapserv processes. Geoserver uses connection pooling with 20 connections. Spatial indexes on both data sets. No reprojection required. Minimal styling. Responses are 2-color PNG images. More details in the appendix.**

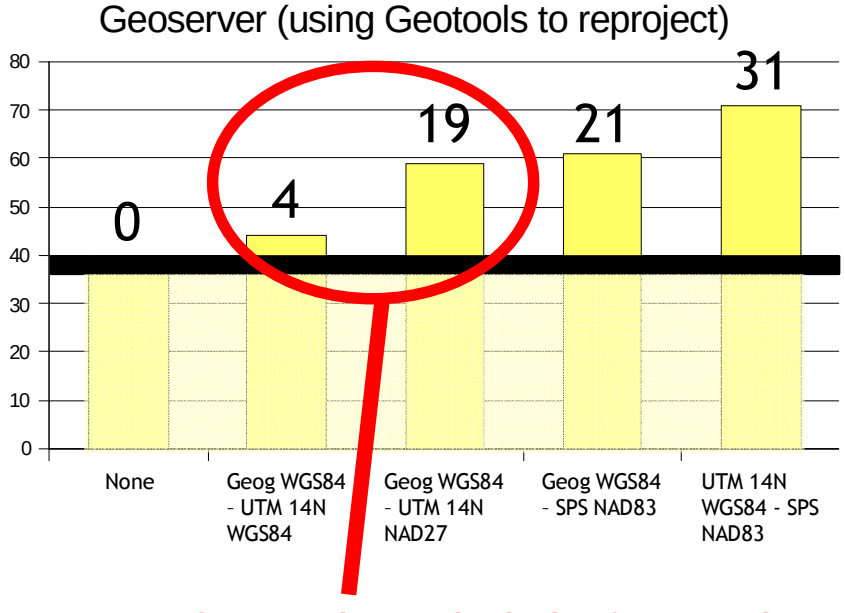
... or Throughput, if you prefer



An alternative way to summarize the data collected for the concurrency test. (Higher lines are better here.)

Test #3: Reprojection





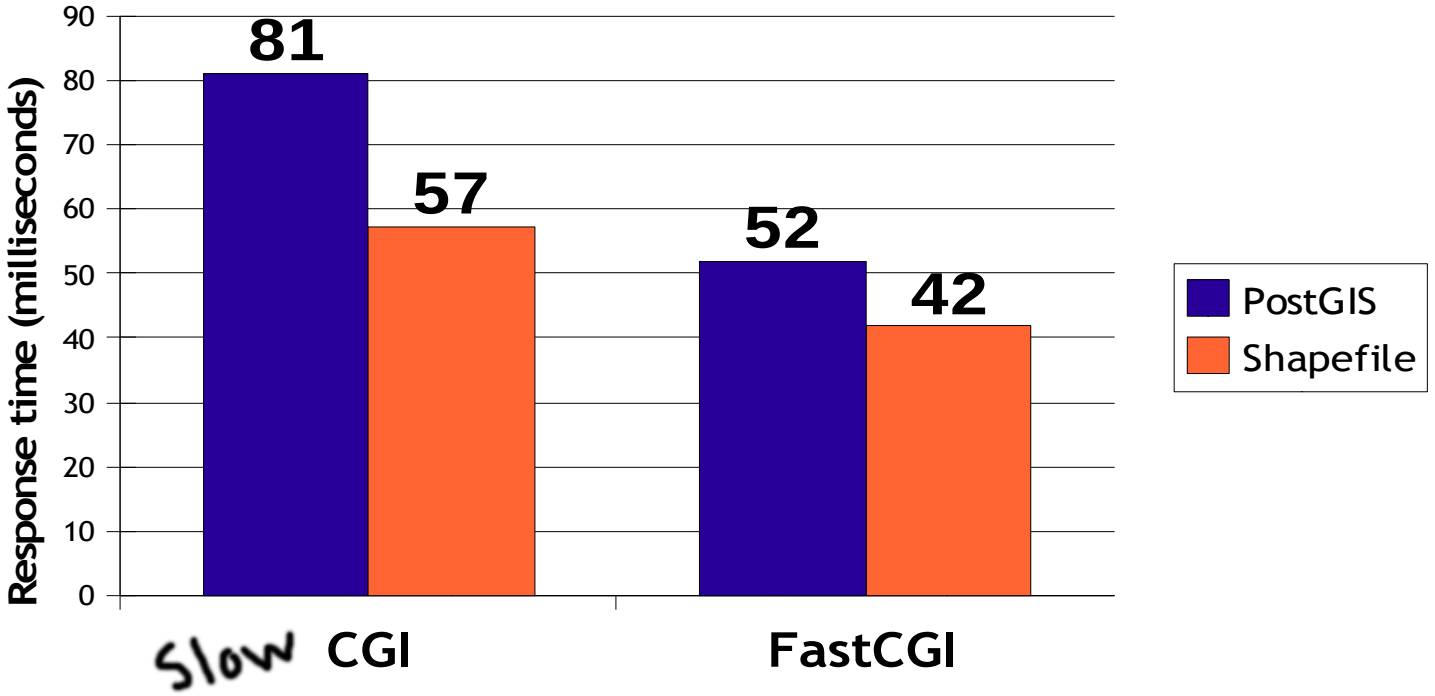
PROJ optimizes by assuming these source and target datums are equivalent.

Currently Mapserver calls PROJ for every vertex, but it could improve by batching those into a single call.

Geotools is slightly faster than PROJ *for these cases*.

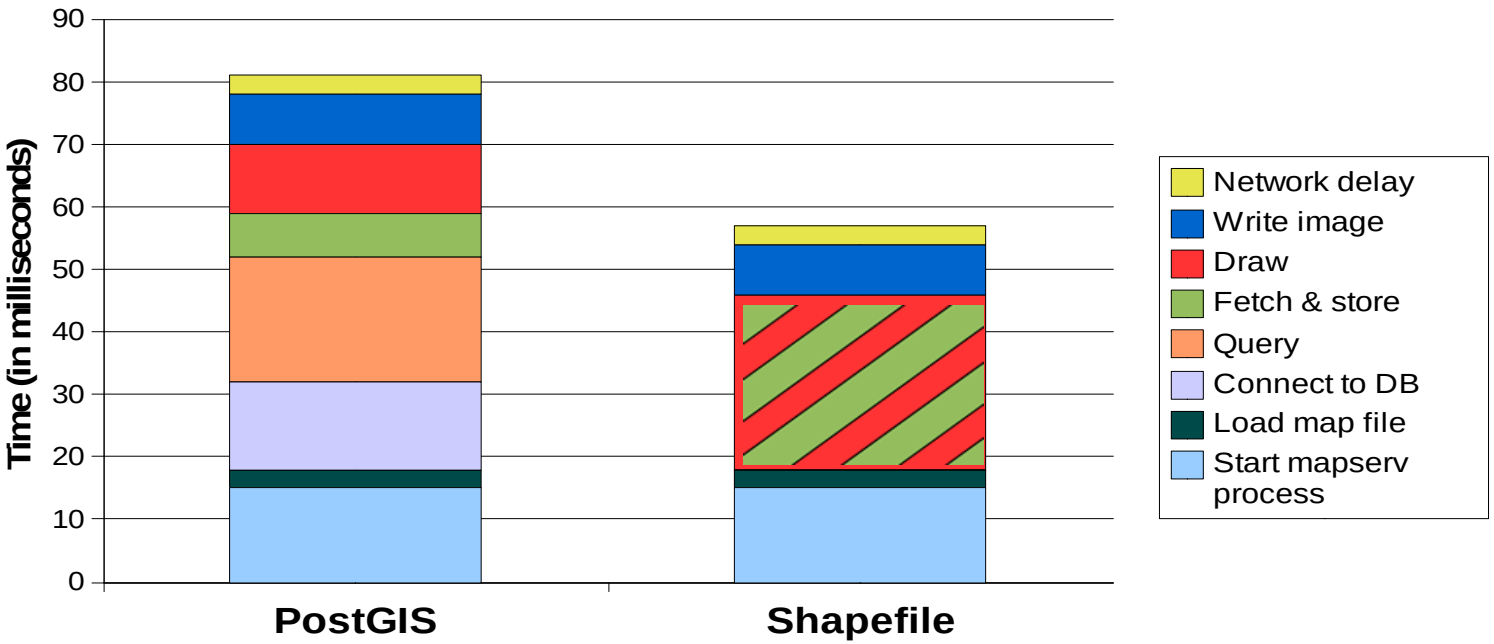
Geoserver simplifies geometry before reprojecting.

CGI vs. FastCGI (Mapserver only)



Notes: **Average of 30 samples. One request at a time (no concurrency). Each request fetches one layer with 1000 features from a data set of 10,000. Spatial indices on both data sets. No reprojection required. Minimal styling. Responses are 1-bit PNG images. The same binary file was used for both CGI and FastCGI. FastCGI through Apache and mod_fcgi.**

Breakdown of Mapserver Response Time

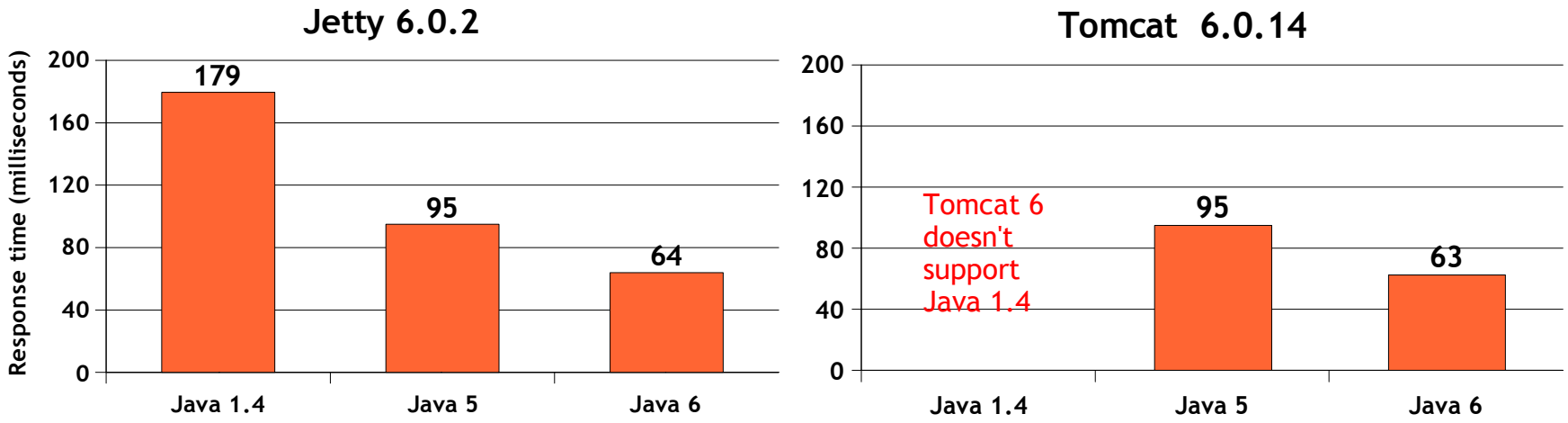


- FastCGI eliminates Start mapserv process and Connect to DB costs.
- The Write image step is dependant on output format.

Breakdown of Geoserver Response Time

404: Document not found

Servlet Container and Java (Geoserver only)



- **These results show average response times for** the same WMS request **when Geoserver is backed by different Servlet containers and Java versions.**
- **Using shapefile backend.**
- **Conclusion:** Use Java 6!

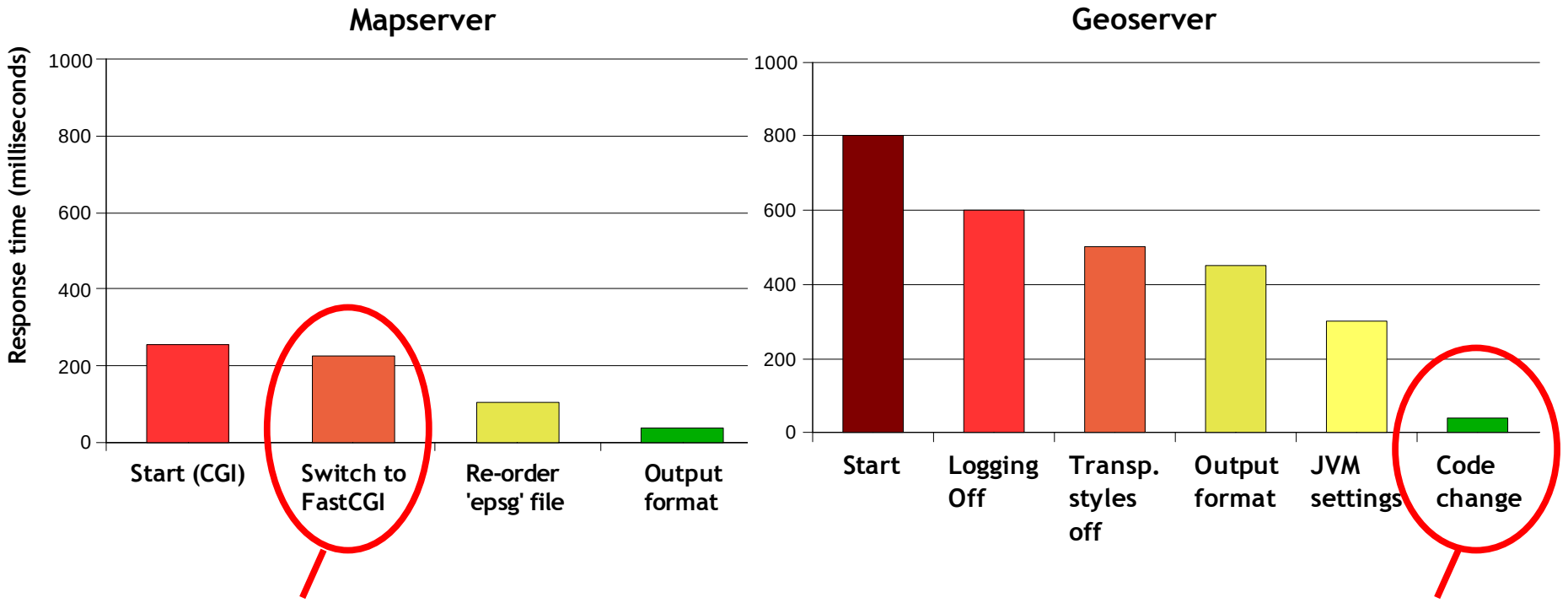
Outcome of the tests

- Lots of performance optimizations to Geoserver which will be available in version 1.6.

- Identified a few places where Mapserver can improve too. (These will be reported as "bugs" as time permits.)

- Both servers can be FAST, but require some special configuration.

The Road to Speed



Data sources with high connection overhead will benefit much more from FastCGI.

All will be in Geoserver 1.6

Performance Tips (Mapserver)

- Beware of

```
PROJECTION
  'init=epsg:4326'
END
```

  The “init=” syntax causes one lookup in the PROJ4 'epsg' file for every occurrence in the map file. (Move your most-used EPSG codes to the top of the 'epsg' file.)

- Use FastCGI instead of ordinary CGI. Instruction here: http://mapserver.gis.umn.edu/docs/howto/fastcgi
- Ensure you have enough FastCGI processes.

Performance Tips (Geoserver)

- Geoserver has many features enabled by default. Gain performance by disabling features you don't need.
  - Transparent styles double draw time. Use opacity=1 in your SLD to disable.
  - Antialiasing linework is costly. Try '&format_options=antialias:none' to disable.
  - Experiment with disabling “PNG native acceleration”
- Favour Java 6 over Java 5 over Java 1.4.
- JVM Settings: Increase heap size. Use -server switch.
- Experiment with different shapefile index depths.
- Turn off logging

How can the servers improve?

Mapserver

- **More efficient scanning of shapefile quadtree indexes.** [Bug Reported]
- **Batch PROJ calls when doing on-the-fly reprojection.**
- **Reduce number of 'epsg' lookups on map files.**

Geoserver

- **Various optimizations to the renderer.**
  [Fixes Committed]
- **More efficient scanning of shapefile quadtree index.** [Fixes Committed]

Questions? Contact Us.

Brock Anderson:

banders@refractions.net

Justin Deolivera:

jdeolive@openplans.org

General WMS Performance Tips

- Only fetch from your data source the features that will be drawn, otherwise the servers have to spend time scanning and discarding the unused ones.

- Output format affects response time. 256 color PNG is faster to create than PNG24 on both servers.

- On-the-fly reprojection has a price. Store data in the same projection it's most commonly requested in.

Appendix

Breakdown of Mapserver Response Time

The graph represents mapserv running in CGI mode to show all startup costs. Metrics for "Load map file", "Connect to DB", "Fetch & store", "Draw" and "Write image" were collected by modifying source code to capture and log durarions of those operations. "Query" time measured with PostgreSQL's *explain analyze* command. "Start mapserv process" + "Network delay" = difference between response times recorded by JMeter and my custom mapserv logging which recorded the total time servicing a request.

	PostGIS	Shapefile
Start mapserv process	15ms	15ms
Load map file	3ms	3ms
Connect to DB	14ms	n/a
Query	20ms	n/a
Fetch	7ms	n/a
Draw	11ms	28ms
Write image	8ms	8ms
Network delay	3ms	3ms

PostGIS vs Shapefiles

This test uses two different data sets: one with 3,000,000 features, the other with 10,000. Each request fetches 1000 features by limiting with a 'bbox' WMS parameter. Each bar is an average of 30 samples. One request at a time (no concurrency). Mapserver and Geoserver use the same data. Mapserver is using FastCGI via Apache/mod_fcgi. Spatial indices on both data sets. The shapefile indices were generated with 'shptree'. No reprojection required. Minimal styling. Responses are 2-color PNG images (indexed color).

The unusual Mapserver result for the case of a 3 million record shapefile has been reported to the Mapserver bug tracker: http://trac.osgeo.org/mapserver/ticket/2282

Appendix

Concurrency and Throughput

Notes: **Data in PostGIS and shapefile formats. Mapserver and Geoserver use the same data. Mapserver is using FastCGI via Apache/mod_fcgi. 20 FastCGI mapserv processes. Geoserver uses connection pooling with 20 connections. Spatial indexes on both data sets. No reprojection required. Minimal styling. Responses are 2-color PNG images (indexed color). "Concurrent" requests were fired in bursts with zero ramp up (as near to simultaneously as possible). I.e. For the test of 10 concurrent requests, all ten requests were fired at the same time. Once all the responses came back then the next burst of requests went out. Requests use random bboxes which fetch ~1000 features. The same random bboxes are used against both servers.**

Mapserver (Response times)

	PostGIS	Shapefile
1	50	39
2	51	40
5	91	75
10	182	147
15	269	229
20	315	283
40	784	612
60	1269	905

Geoserver (Response times)

	PostGIS	Shapefile
1	42	27
2	43	30
5	81	47
10	166	103
15	261	162
20	378	252
40	747	514
60	1170	773

Response times are measured in milliseconds. Throughput times represent responses per second.

The concurrency level is the left-most column in each table (1, 2, 5, 10, ...).

Mapserver (Throughput times)

	PostGIS	Shapefile
1	19.6	24.9
2	28.2	33.4
5	35.4	51.6
10	38.4	53.8
15	42.5	55
20	42.4	54.1
40	43.2	54.9
60	43.1	51.5

Geoserver (Throughput times)

	PostGIS	Shapefile
1	24.6	35.6
2	32.3	41.8
5	47.1	68.6
10	49.9	74.1
15	49.2	73.3
20	47.7	68
40	48.3	68
60	47.8	70.7

Appendix

Summary of Geoserver code changes made to improve performance:

• optimized access to the shapefile spatial index (it was reading tiny sections of the file instead of doing some buffered access)

• figure out the optmimal palette out of the SLD style (when possible, that is, when antialiasing is off)

*** don't access the dbf file when not necessary**

*** avoid unecessary operations, like duplicating over and over the same coordinate[] during rendering (loading it, generalize, reproject, copy back in the geometry and so on, now the array it's copied just once)**

Raw list of changes here:

http://jira.codehaus.org/secure/ManageLinks.jspa?id=55176