
Firefox
Crash
Reporting
laura@ mozilla.com
@lxt

Webtools @ Mozilla

- Crash reporting
- Performance measurement
- Localization
- Code search and static analysis
- Other stuff: product delivery and updates, plugins management, infrastructure dashboards, authentication, Etherpad Lite, Air Mozilla...

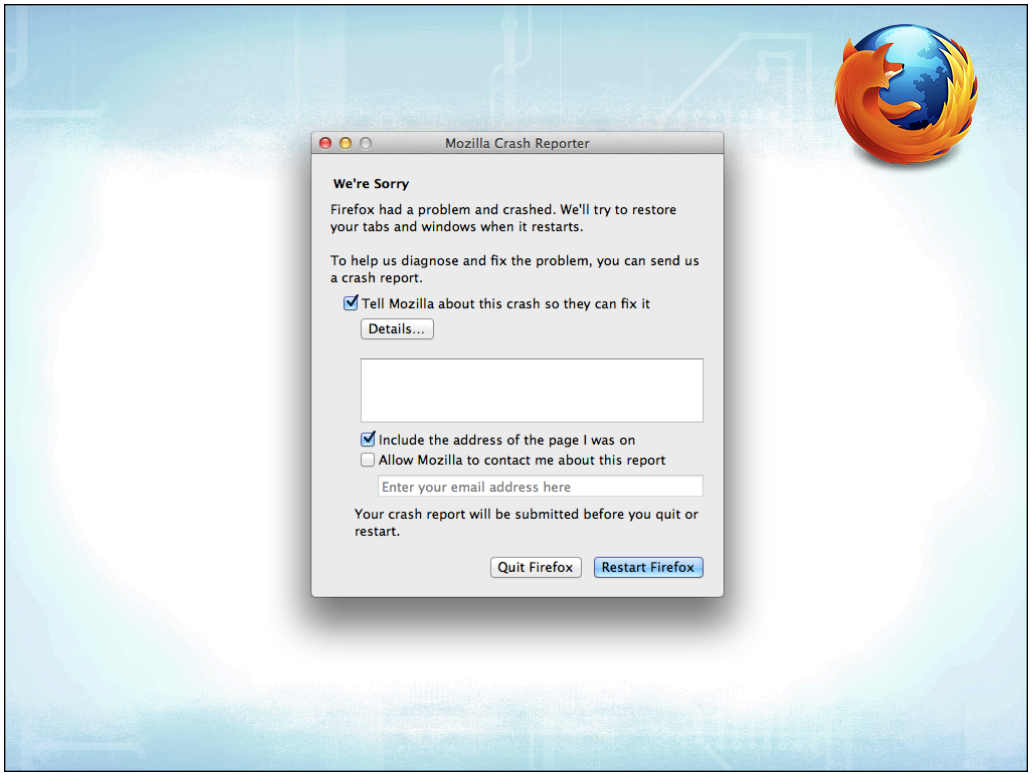
Overview

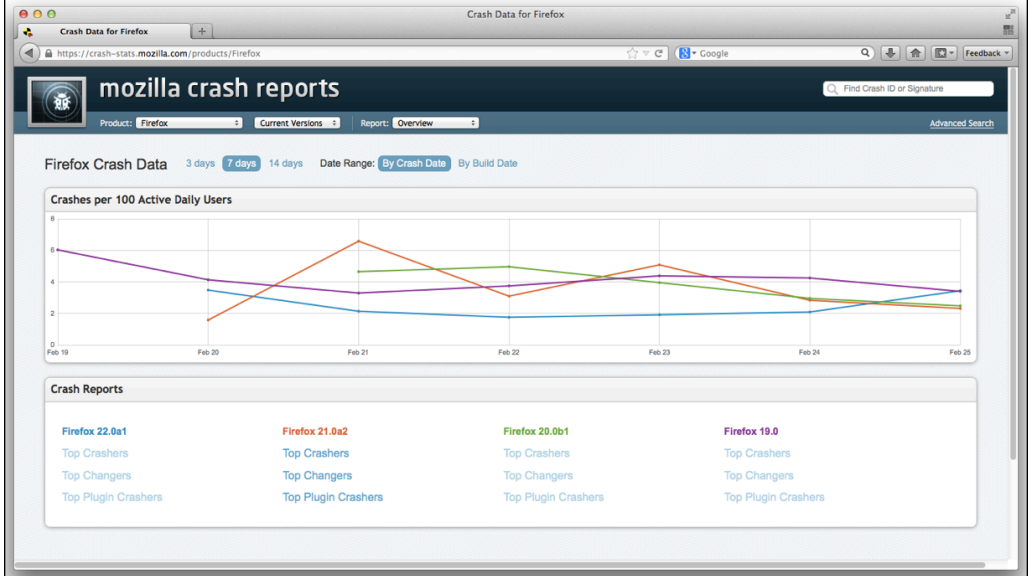
- What is Socorro?
- Some numbers
- Architecture
- Work process and tools
- Future, questions

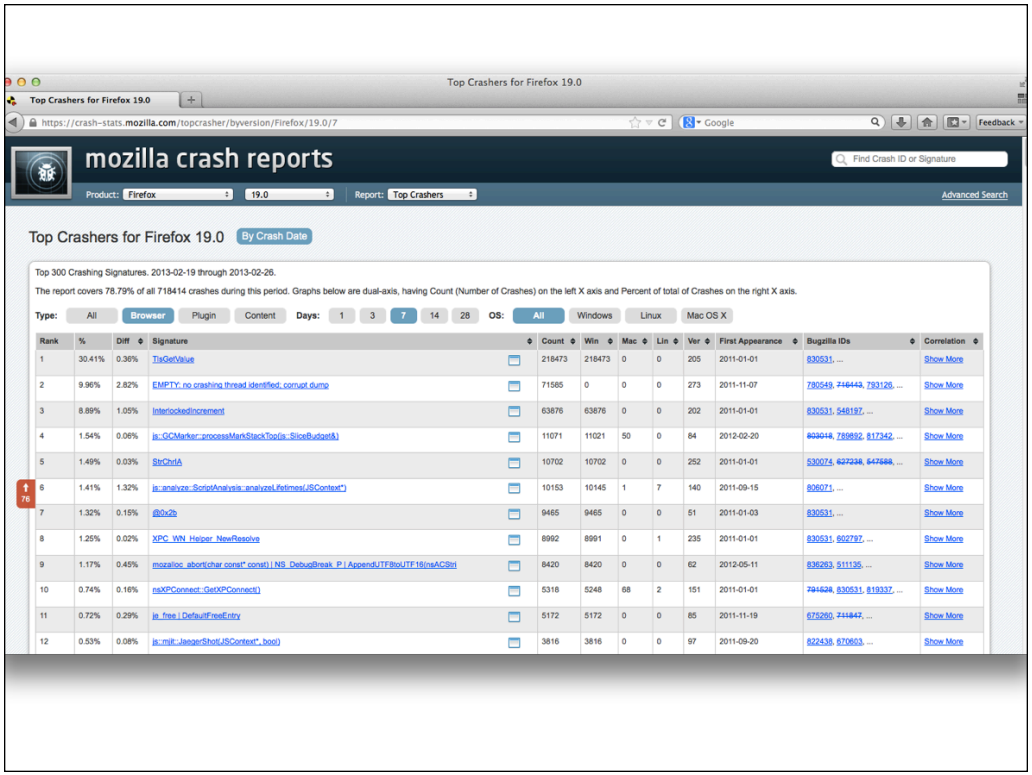
What is Socorro?

Socorro

Very Large Array at Socorro, New Mexico, USA. Photo taken by Hajor, 08.Aug.2004. Released under cc:by.sa and/or GFDL. Source: http://en.wikipedia.org/wiki/File:USA.NM.VeryLargeArray.02.jpg


Mozilla Crash Reporter
We're Sorry
Firefox had a problem and crashed. We'll try to restore your tabs and windows when it restarts.
To help us diagnose and fix the problem, you can send us a crash report.
Tell Mozilla about this crash so they can fix it
Details...
Include the address of the page I was on
Allow Mozilla to contact me about this report
Enter your email address here
Your crash report will be submitted before you quit or restart.
Quit Firefox
Restart Firefox


Crash Data for Firefox
Crash Data for Firefox
https://crash-stats.mozilla.com/products/Firefox
Google
Feedback
mozilla crash reports
Find Crash ID or Signature
Product: Firefox
Current Versions
Report: Overview
Advanced Search
Firefox Crash Data
3 days
7 days
14 days
Date Range:
By Crash Date
By Build Date
Crashes per 100 Active Daily Users
8
6
4
2
0
Feb 19
Feb 20
Feb 21
Feb 22
Feb 23
Feb 24
Feb 25
Crash Reports
Firefox 22.0a1
Top Crashers
Top Changers
Top Plugin Crashers
Firefox 21.0a2
Top Crashers
Top Changers
Top Plugin Crashers
Firefox 20.0b1
Top Crashers
Top Changers
Top Plugin Crashers
Firefox 19.0
Top Crashers
Top Changers
Top Plugin Crashers

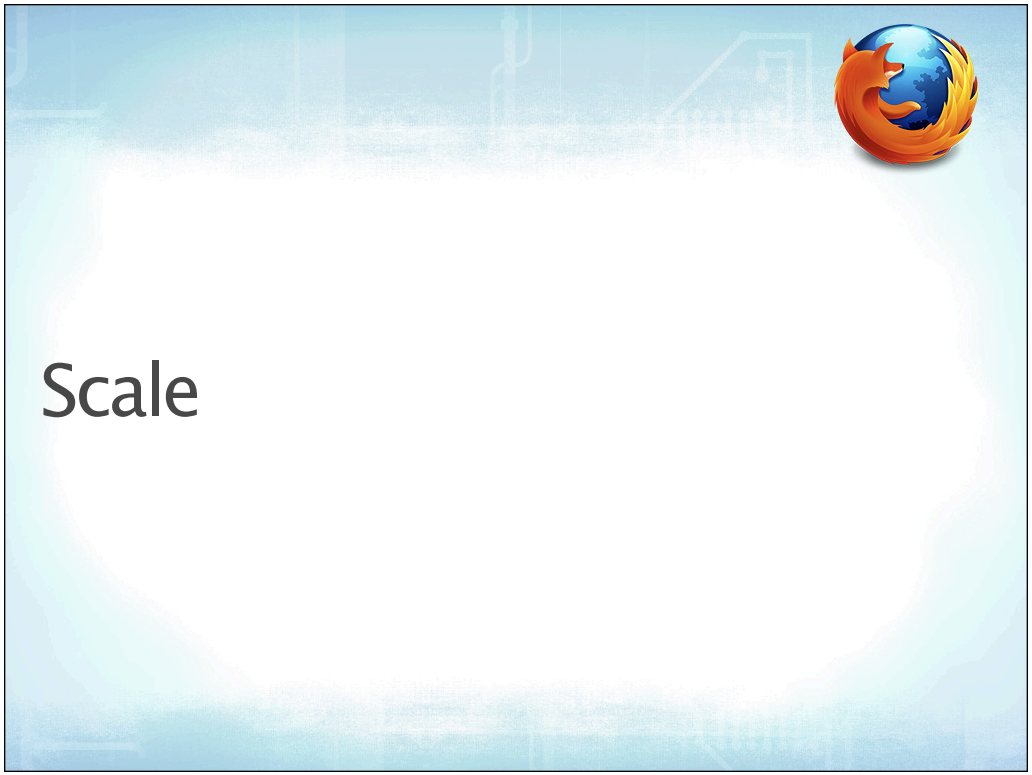

Top Crashers for Firefox 19.0
Top Crashers for Firefox 19.0
https://crash-stats.mozilla.com/topcrasher/byversion/Firefox/19.0/7
Google
Feedback
mozilla crash reports
Find Crash ID or Signature
Product: Firefox
19.0
Report: Top Crashers
Advanced Search
Top Crashers for Firefox 19.0
By Crash Date
Top 300 Crashing Signatures. 2013-02-19 through 2013-02-26.
The report covers 78.79% of all 718414 crashes during this period. Graphs below are dual-axis, having Count (Number of Crashes) on the left X axis and Percent of total of Crashes on the right X axis.
Type: All Browser Plugin Content
Days: 1 3 7 14 28
OS: All Windows Linux Mac OS X
Rank % Diff Signature Count Win Mac Lin Ver First Appearance Bugzilla IDs Correlation
1 30.41% 0.36% TlsGetValue 218473 218473 0 0 205 2011-01-01 830531, ... Show More
2 9.96% 2.82% EMPTY: no crashing thread identified; corrupt dump 71585 0 0 0 273 2011-11-07 780549, 716443, 793126, ... Show More
3 8.89% 1.05% InterlockedIncrement 63876 63876 0 0 202 2011-01-01 830531, 548197, ... Show More
4 1.54% 0.06% js::GCMarker::processMarkStackTop(js::SliceBudget&) 11071 11021 50 0 84 2012-02-20 803018, 789892, 817342, ... Show More
5 1.49% 0.03% StrChrIA 10702 10702 0 0 252 2011-01-01 530074, 627238, 647588, ... Show More
6 1.41% 1.32% js::analyze::ScriptAnalysis::analyzeLifetimes(JSContext*) 10153 10145 1 7 140 2011-09-15 806071, ... Show More
7 1.32% 0.15% @0x2b 9465 9465 0 0 51 2011-01-03 830531, ... Show More
8 1.25% 0.02% XPC_WN_Helper_NewResolve 8992 8991 0 1 235 2011-01-01 830531, 602797, ... Show More
9 1.17% 0.45% mozalloc_abort(char const* const) | NS_DebugBreak_P | AppendUTF8toUTF16(nsACStri 8420 8420 0 0 62 2012-05-11 836263, 511135, ... Show More
10 0.74% 0.16% nsXPConnect::GetXPConnect() 5318 5248 68 2 151 2011-01-01 791528, 830531, 819337, ... Show More
11 0.72% 0.29% je_free | DefaultFreeEntry 5172 5172 0 0 85 2011-11-19 675280, 711847, ... Show More
12 0.53% 0.08% js::mjit::JaegerShot(JSContext*, bool) 3816 3816 0 0 97 2011-09-20 822438, 670603, ... Show More
76

Typical use cases

- What are the most common crashes for a product/version/channel?
- What new crashes / regressions do we see emerging? What's the cause of an emergent crash?
- How crashy is one build compared to another?
- What correlations do we see with a particular crash?

What else can we do?

- Does one build have more (null signature) crashes than other builds?
- Analyze differences between Flash versions x and y crashes
- Detect duplicate crashes
- Detect explosive crashes
- Find “frankeninstalls”
- Analyze exploitable crashes
- Ad hoc reporting for tracking down chemspill bugs


Scale

A different type of scaling:

- Typical webapp: scale to millions of users without degradation of response time
- Socorro: less than a hundred users, terabytes of data

Basic law of scale still applies:

The bigger you get, the more spectacularly you fail

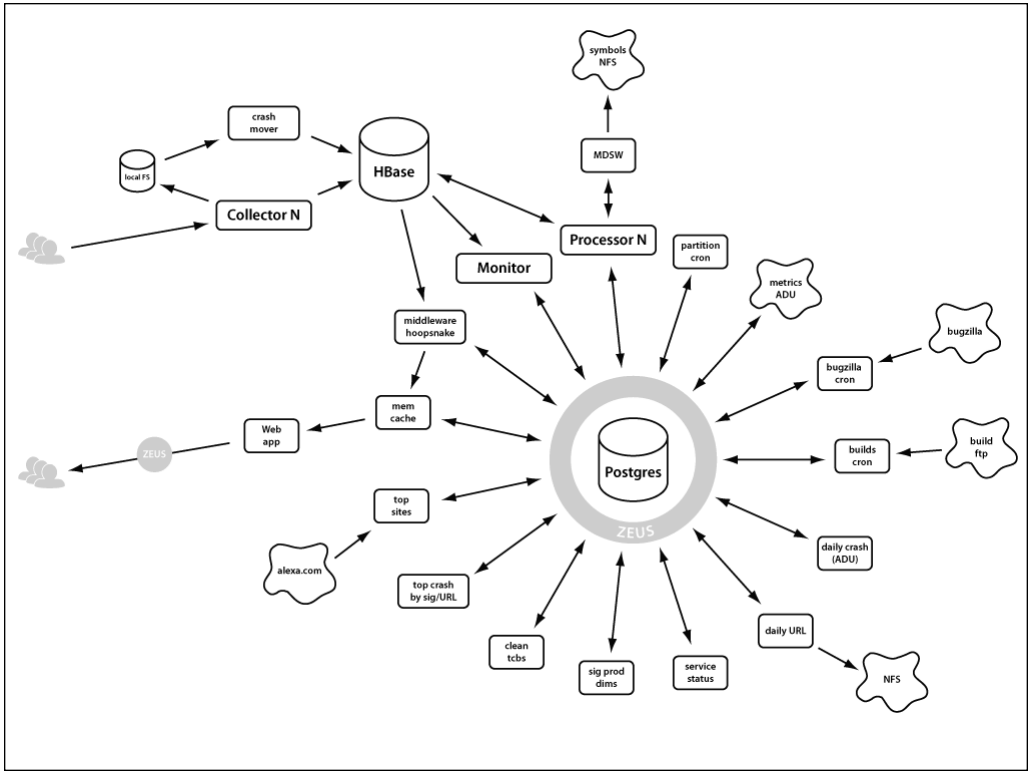
Firehose engineering

- At peak we receive up to 3000 crashes per minute
- 3 million per day
- Median crash size 150k, max size 20MB (reject bigger)
  - Android/FirefoxOS crashes a bit bigger (200k/250k)
- ~800GB stored in PostgreSQL - metadata + generated reports
- ~110TB stored in HDFS (3x replication, ~40TB of HBase data) - raw reports + processed reports

Implementation scale

- > 120 physical boxes (not cloud)
- ~10 developers + DBAs + sysadmin team + QA + Hadoop ops
- Deploy weekly+ but will move to CD within next month or so

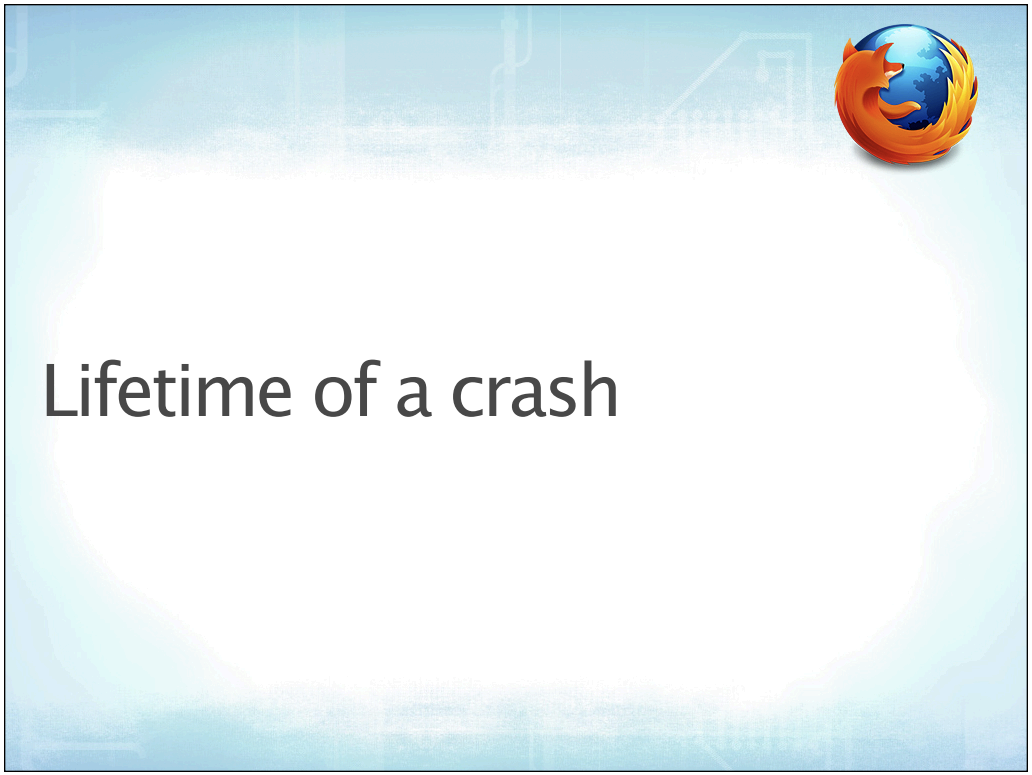
Architecture


symbols
NFS
crash
mover
local FS
HBase
MDSW
Collector N
Processor N
partition
cron
Monitor
metrics
ADU
middleware
hoopsnake
bugzilla
bugzilla
cron
mem
cache
Web
app
ZEUS
Postgres
builds
cron
build
ftp
top
sites
alexa.com
daily crash
(ADU)
top crash
by sig/URL
daily URL
clean
tcbs
sig prod
dims
service
status
NFS
ZEUS

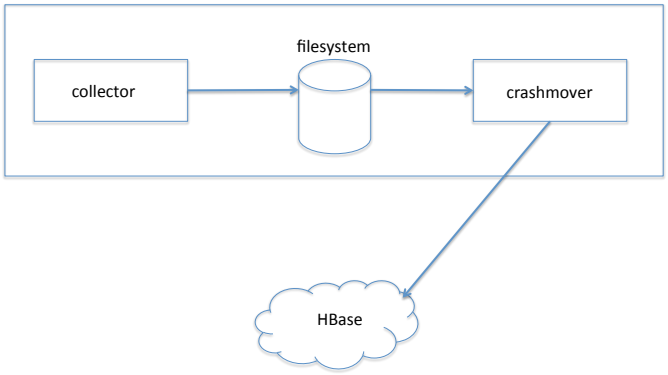
"Socorro has a lot of moving parts"

...

"I prefer to think of them as dancing parts"


Lifetime of a crash

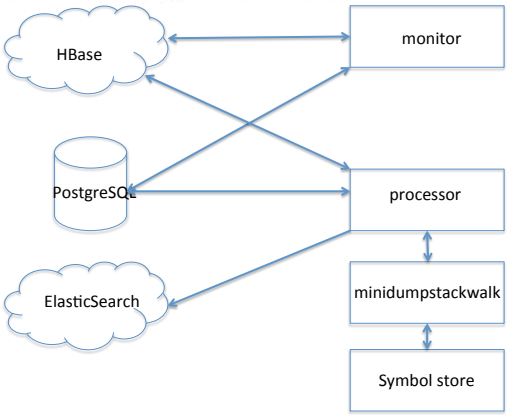
Collection



Collection

- Breakpad submits raw crash via POST (metadata json + minidump)
- Collected to disk by collector (web.py WSGI app)
- Moved to HBase by crashmover
- Noticed in queue by monitor and assigned for processing

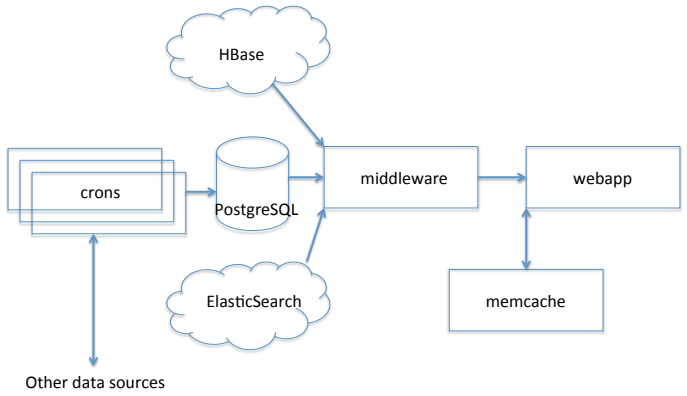
Processing


HBase
monitor
PostgreSQL
processor
ElasticSearch
minidumpstackwalk
Symbol store

Processing

- Noticed in queue by monitor and assigned for processing
- Processor spins off minidumpstackwalk (MDSW)
- MDSW re-unites raw crash with symbols to generate a stack
- Processor generates a signature and pulls out other data
- Processor writes processed crash back to HBase and metadata to PostgreSQL and ElasticSearch

Reporting


HBase
crons
PostgreSQL
middleware
webapp
ElasticSearch
memcache
Other data sources

Back end processing

Large number of cron jobs, e.g.:

- Copy clean data into fact tables
- Update ADU from Vertica
- Calculate aggregates: Top crashers by signature, crashes/100ADU/build
- Process incoming builds from ftp server
- Match known crashes to bugzilla bugs
- Duplicate detection
- Generate extracts (CSV) for further analysis (in CouchDB, f.e.)

Middleware

- All data access to through REST API
- Enable other apps against the data platform (and allow the core team to rewrite webapp more easily)
- Experiments with giving each component its own API for status and health checks

Webapp

- Hardest part is sometimes how to visualize the data
- Example: reporting in build time as well as clock time
- Just rewritten in Python/Django; running in parallel with older PHP app

Pluggable architecture

- Goal is to have components be pluggable and easy to switch out
- Back end components have a simple fetch-transform-save architecture
- Storage systems pluggable, e.g. for low volume installation use filesystem instead of HBase
- Middleware isolates data storage from the webapp

Implementation details

- Python 2.6
- PHP 5.3
- PostgreSQL 9.2
- memcache for the webapp
- Thrift for HBase access
- HBase (CDH3, 4 sometime soon)
- Some bits of C++, Java, perl, Pig


Managing complexity:
work process and tools

Development process

- Fork
- Hard to install (you can use a VM)
- Pull request with bugfix/feature
- Code review
- Lands on master

Development process –2

- Jenkins polls github master, picks up changes
- Jenkins runs tests, builds a “package”
- Build automatically picked up and pushed to dev
- Wanted changes merged to release branch
- Jenkins builds release branch, pushes to stage
- QA runs acceptance on stage (Selenium/Jenkins + manual)
- Push same build to production

Deployment =

- Run a single script with the build as parameter
- Pushes it out to all machines and restarts where needed
- About to automate this further

As an aside:

Build all the machinery for continuous deployment even if you don't want to deploy continuously

Configuration management

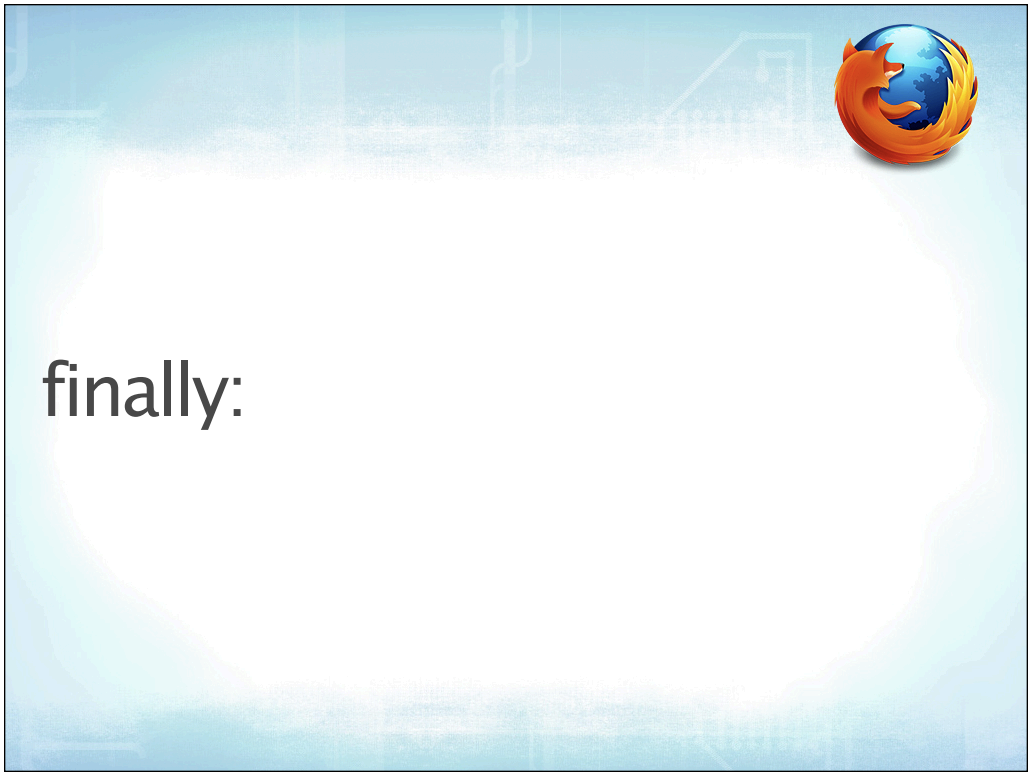
- Some releases involve a configuration change
- These are controlled and managed through Puppet
- Again, a single line to change config the same way every time
- Config controlled the same way on dev and stage; tested the same way; deployed the same way

Virtualization

- Front end devs don’t want to install HBase
- Use Vagrant to set up a virtual machine
- Use Jenkins to build a new Vagrant VM with each code build
- Use Puppet to configure the VM the same way as production
- Hard part is keeping Vagrant instances up to date and finding the right packages (Ubuntu vs RHEL in prod) - failing right now
- Second hard part is having a useful amount of data for development - fakedata instance for testing

New and Upcoming

- crontabber: manage cron dependencies and auto-recover on failure
- More use of statsd/graphite for perf measurement and monitoring
- chief for deployment via IRCbot
- Try servers: stage different branches in parallel


finally:

New and upcoming

- More reports and visualizations: gc crashes, crash trends (done), Flash version reporting, better signature summaries (done), better correlation reports
- ElasticSearch: better search including faceting (shipping in 2 weeks)
- Dragnet: using crash data to populate a database of DLLs (staged)
- More analytics: exploitability, etc
- More ways to query data: API, reporting replica of PostgreSQL, Pig (done), ES
- Better (real) queueing (massive bikeshedding effort)
- Grand Unified Configuration System (done, shipping piecewise)
- crontabber (cronjob co-ordination and management. done)
- SaaS

Everything is open (source)

Site: https://crash-stats.mozilla.com

Fork: https://github.com/mozilla/socorro

Read/file/fix bugs: https://bugzilla.mozilla.org/

Docs: http://www.readthedocs.org/docs/socorro

Mailing list: https://lists.mozilla.org/listinfo/tools-socorro

Join us in IRC: irc.mozilla.org #breakpad

Questions?

- Ask me, now or later
- laura@mozilla.com