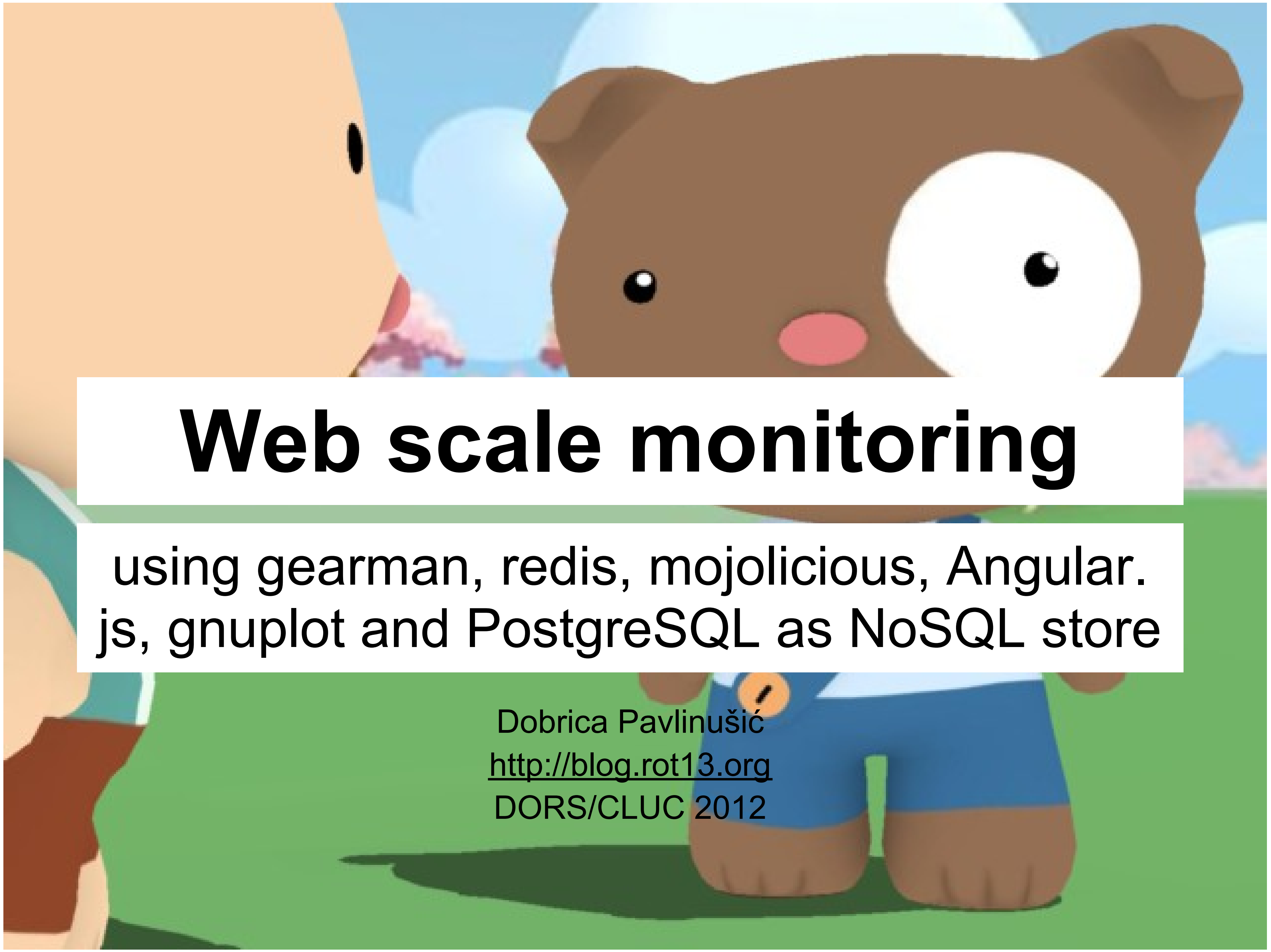


Web scale monitoring

using gearman, redis, mojolicious, Angular.js, gnuplot and PostgreSQL as NoSQL store

Dobrica Pavlinušić
http://blog.rot13.org
DORS/CLUC 2012

Goals

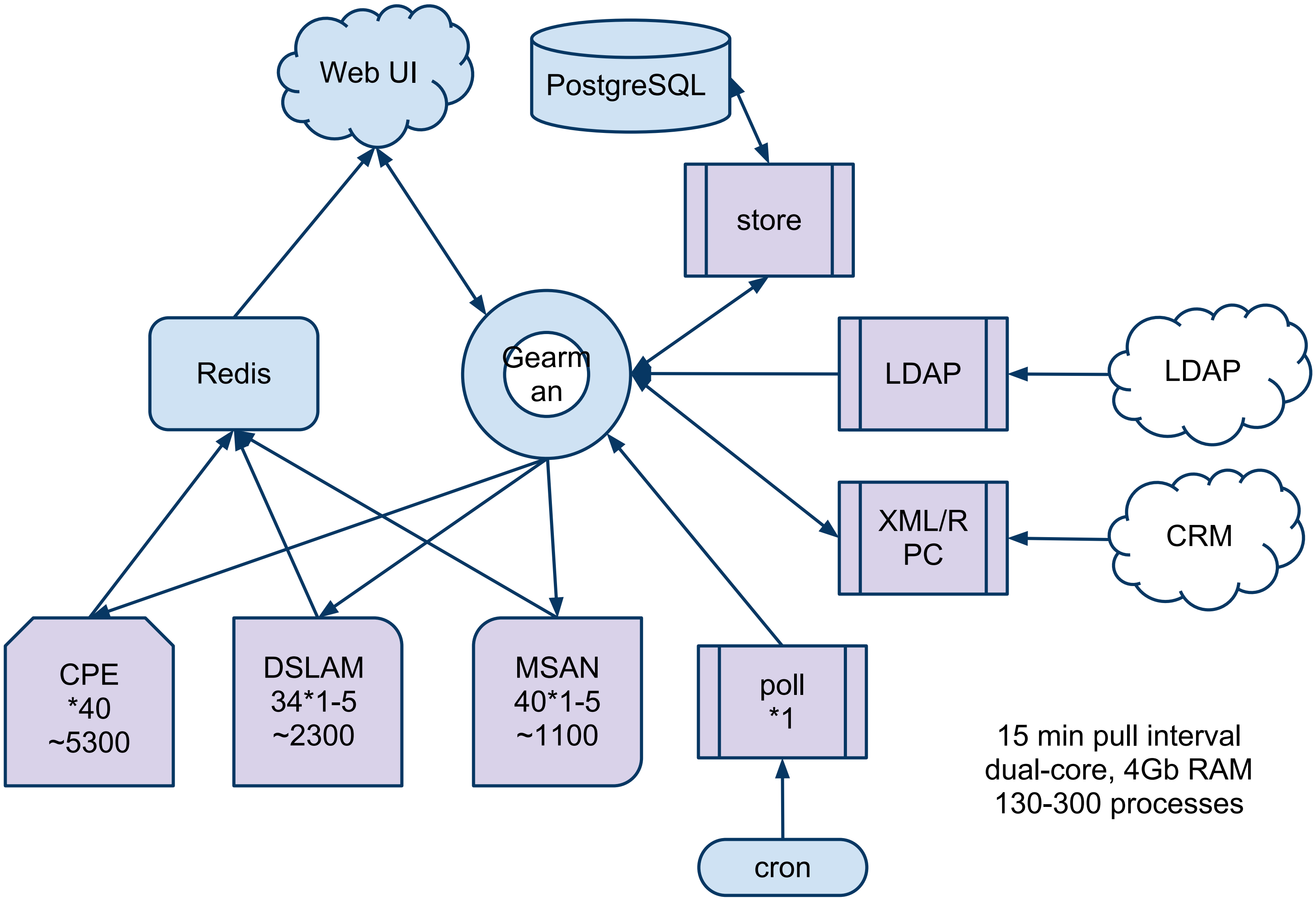
- define problem in terms of scaling
  - Gearman as distributed fork
- don't lock yourself into technological choice
  - relational data database, so what?
- don't mungle and rename data
  - preserve naming through whole stack
- test driven development
  - small iterations, easy deployment
- is your cache really useful?
  - can you make web interface out of it?
- why are web interfaces hard?
  - Angular.js comes to rescue!

Project specification

- Existing perl scripts parse telnet output
  - end-users (CPE)
  - equipment in-between (MSAN, DSLAM)
- Create monitoring system!
- Users data split between LDAP and CRM
- Horizontal scalability (on single box!)
  - number of users grow
- Store data in relational database for reporting
  - All collected data is interesting
- Web interface to inspect data
  - prototype http://youtu.be/Cp31xUdyZBQ

Proposed architecture

- Gearman as queue server
    - workers collect, process and store data
    - Gearman::Driver fork workers on-demand
- PostgreSQL with hstore
    - don't mungle data - not normalized
    - use views for reporting
    - table inheritance for easy expiry of data
- Redis rich structures for data caching
    - provide "warm" data for Web interface
- Web: mojolicious, Angular.js, gnuplot
    - gearman calls and SQL queries to JSONP



15 min pull interval
dual-core, 4Gb RAM
130-300 processes

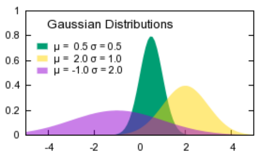
More information

http://gearman.org/

http://redis.io/

http://www.postgresql.org/

http://mojolicio.us/



http://angularjs.org/

Queue

Gearman

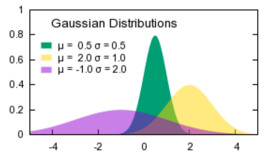
- distributed (across cores) on-demand fork
- German::Driver manages workers
  - min, max process limits
  - copy-on-write fork
  - three master processes (services, MSAN, DSLAM)
  - modify process name for status info (ps ax)
- german workers
  - pollers generate timestamp for data (inserts are queued!)
  - one per work (CPE pollers)
  - persistent workers (TCP connection to MSAN/DSLAM is re-used for all work)

Cache with structures

- store
  - all data from gearman calls (which are slow)
  - statistics from poll workers
- expire data after poll interval
  - fresh data for web interface
- name your keys in sane way!
  - CPE.*, ZTEMSAN.*, ZTEDSLAM.* (poll stats)
  - CRM.login, LDAP.login
  - table.dslam.login (last row inserted)
  - columns.dslam (existence, needed for Web)

hstore

- store key-value pairs (single-level) in single column
- additional columns to support indexes
  - GiST and GIN indexes on hstore are not enough
- table inheritance
  - partitioning of tables by date
  - DELETE and VACUUM can take a long time
  - set sql_inheritance = false
- using PostgreSQL 8.4 (nothing new!)
- PostgreSQL 9.2 will have JSON type support and v8!



Web interface

- mojolicious
  - web server and JSON provider
  - MojoX::Gearman
- gnuplot graphs from **huge** amount of data
  - JavaScript doesn't cut it!
  - get textual data from gearman
  - generate graphs on-the-fly
- Angular.js as nice way to generate HTML from JSON $resources