DBSlayer: A Simpler Way To Proxy MySQL

Derek Gottfrid
derek@nytimes.com

Motivation

- Reduce configuration
- Reduce dependencies
- Handle failovers
- Simple load balancing
- Easy to monitor

Other Motivation

Requirements

Fast

minimal performance overhead

Flexible

works in different configuration scenarios

support different programming lanuages,
different db setups - sharding, master-write/slave-reads

Simple

trivial barrier to entry for new programmers

central configuration, debug, logging, docless

Awesome Name

(thanks anjali)

Solution

HTTP + JSON + MySQL

HTTP - Protocol Transport

Libraries in every language worth programming in.

Lots of tools - load balancers, load testers, browser testing.

REST is great but I am not religious about it.

JSON - Data Exchange Format

Maps to SQL types - exceptions for data/time, blobs

A true data definition language - not a markup language

Libraries available in numerous languages - json.org

Other DB Proxies

SQLRelay

- per process model
- multiple databases
- own protocol

MySQL Proxy

- powerful
- MySQL protocol / libraries
- 1 day tutorial != simple
- internal versions of dbslayer pre-date MySQL Proxy

DBSlayer API:

Database URI:

http://machine:port/db?URLENCODED(JSON OBJECT)
http://machine:port/dbform?URLENCODED(HTML FORM)

Parameters:

SQL - SQL to execute
... others ...

Example Request

```
http://localhost:5621/dbform?
 SQL=select+city,state,latitude,longitude+
  from+zipcode+where+zipcode=90210
```

Example Response

```
{'RESULT': {'HEADER': ['city', 'state', 'latitude',
'longitude'],
            'ROWS': [['Beverly Hills',
                      'CA',
                      34.096600000000002,
                      -118.41200000000001]],
            'TYPES': ['MYSQL_TYPE_VAR_STRING',
                      'MYSQL_TYPE_VAR_STRING',
                      'MYSQL_TYPE_DOUBLE',
                      'MYSQL_TYPE_DOUBLE']},
 'SERVER': 'movies-slave'}
```

Sample Python Client

```
# usually i write this on 1 line w/ a lambda

import urllib2,urllib,cjson

def dbexec(query):
    uri = 'http://localhost:9090/dbform?SQL=%s'
    d = urllib2.urlopen(uri%urllib.quote(query)).read()
    return cjson.decode(d)
```

Other URI / API Endpoints

http://machine:port/stats [# of queries per second]

http://machine:port/stats/log [last 100 requests]

http://machine:port/stats/errors [last 100 errors]

http://machine:port/shutdown [only from localhost]

Feature Review

Connect to multiple DB from one instance

Automatic fail over

Round-robin load balancing

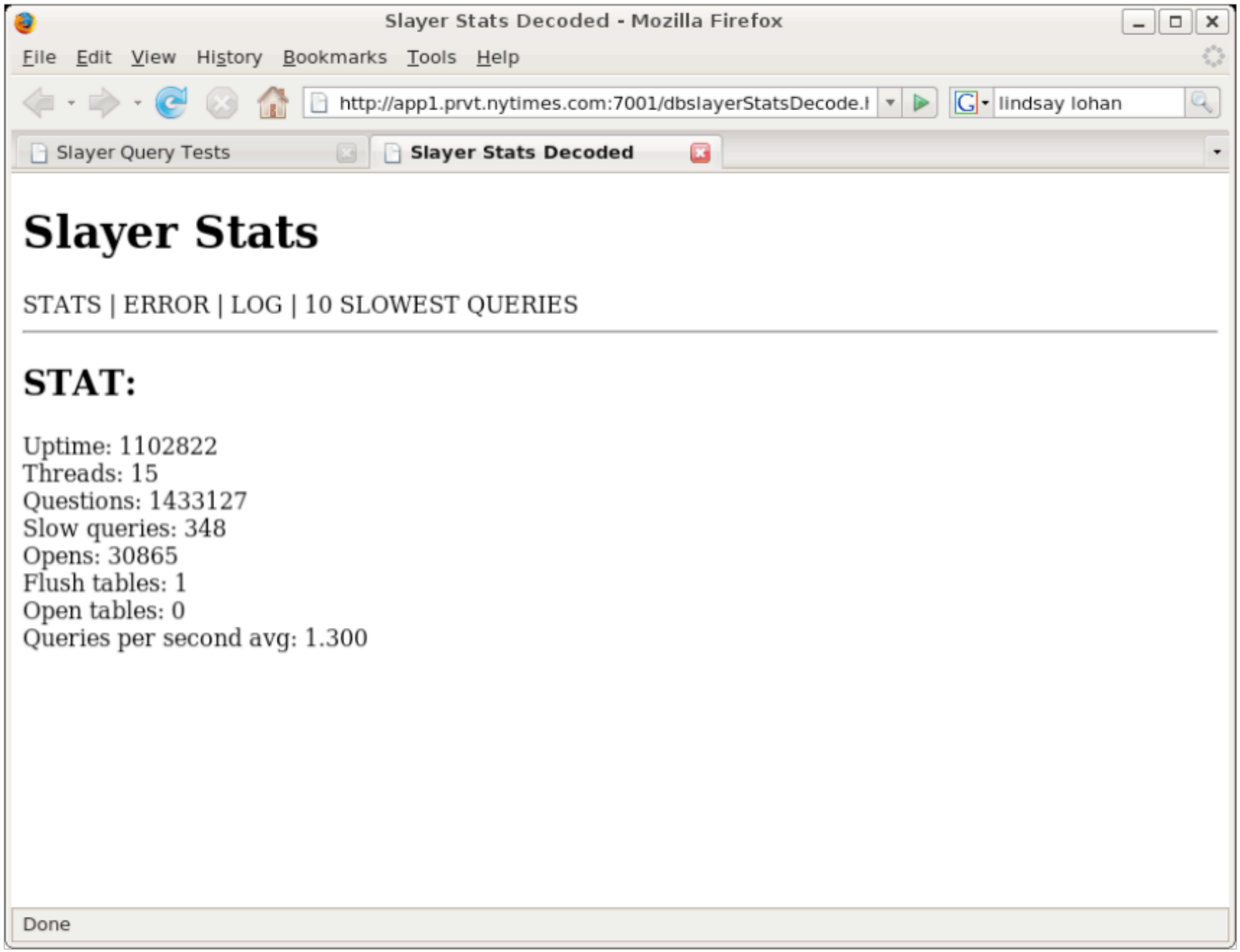
Configurable logging

Statistics gathering

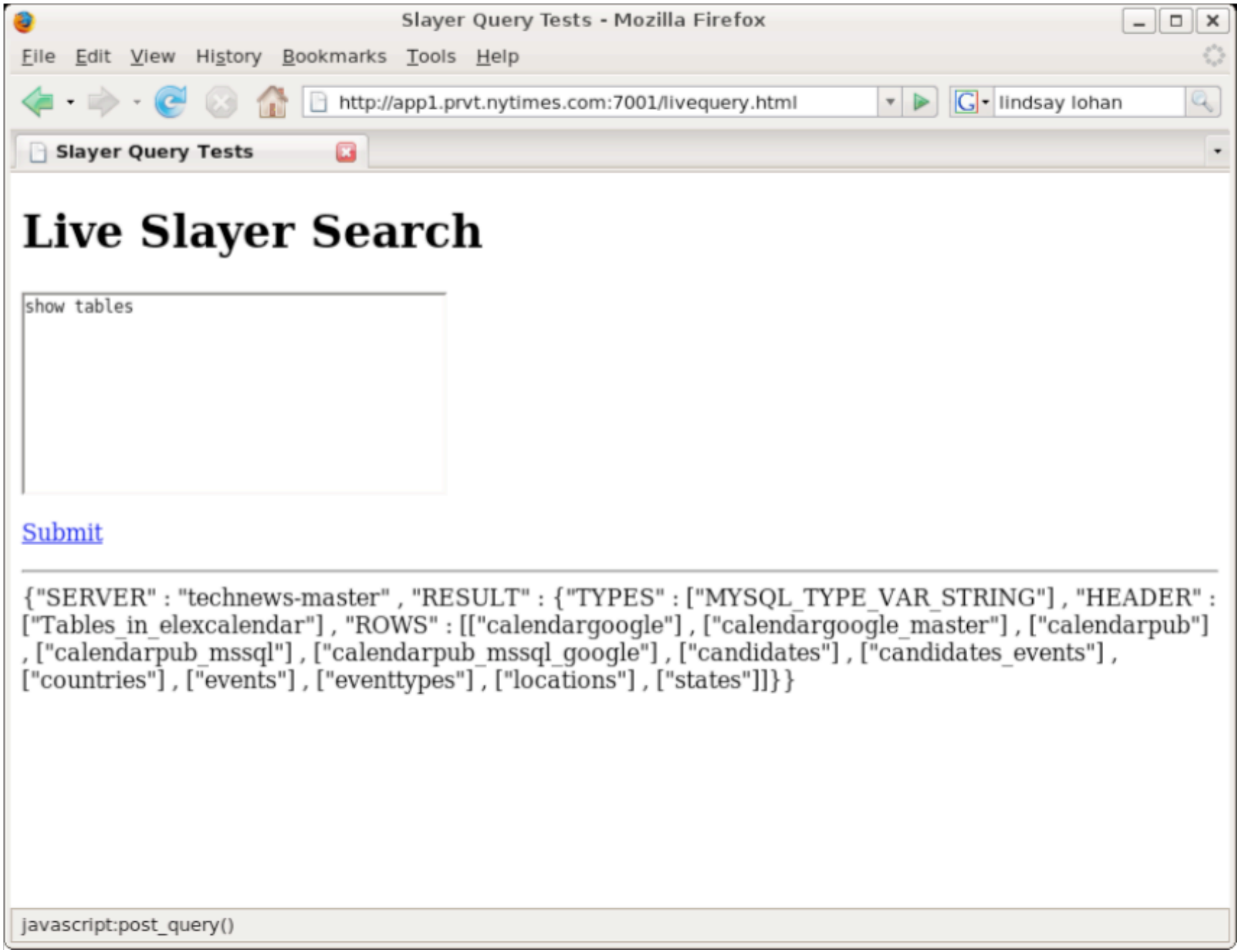
Dead simple API

High Performance - 1k of req/sec - db remains the bottleneck

Stats Monitoring



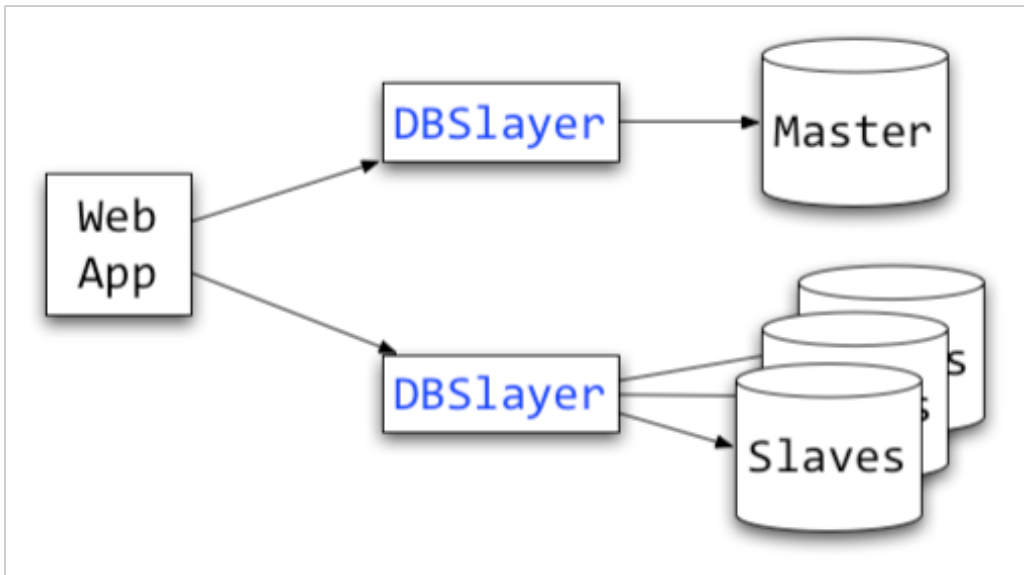
Browser Testing



Usage Scenarios

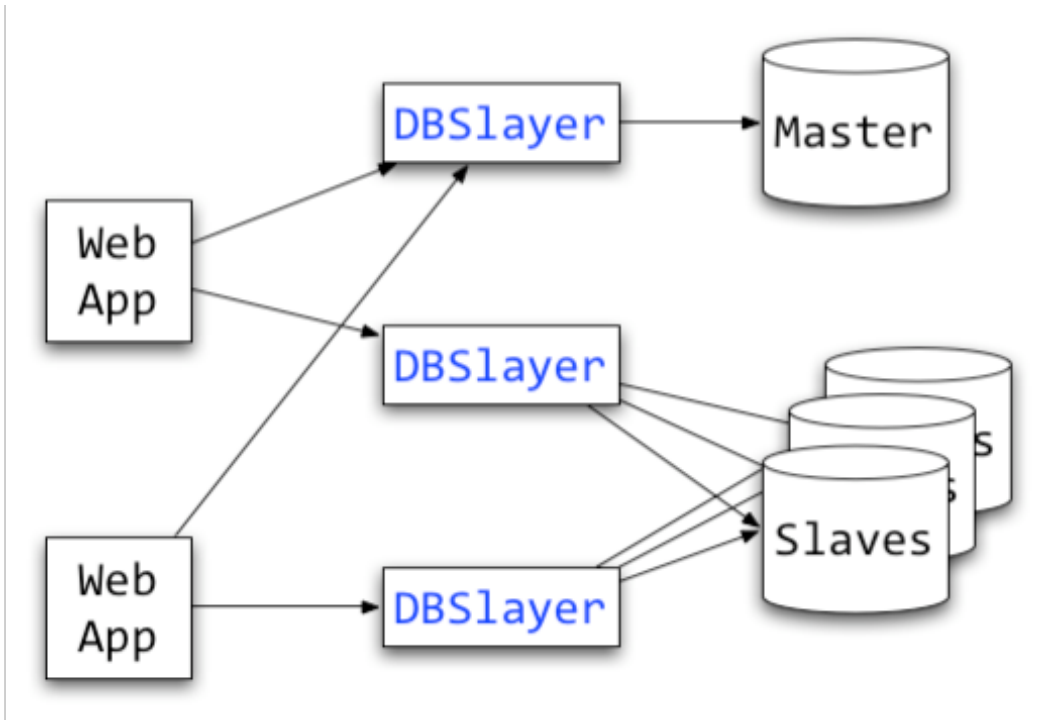
Simple Version

Common Version

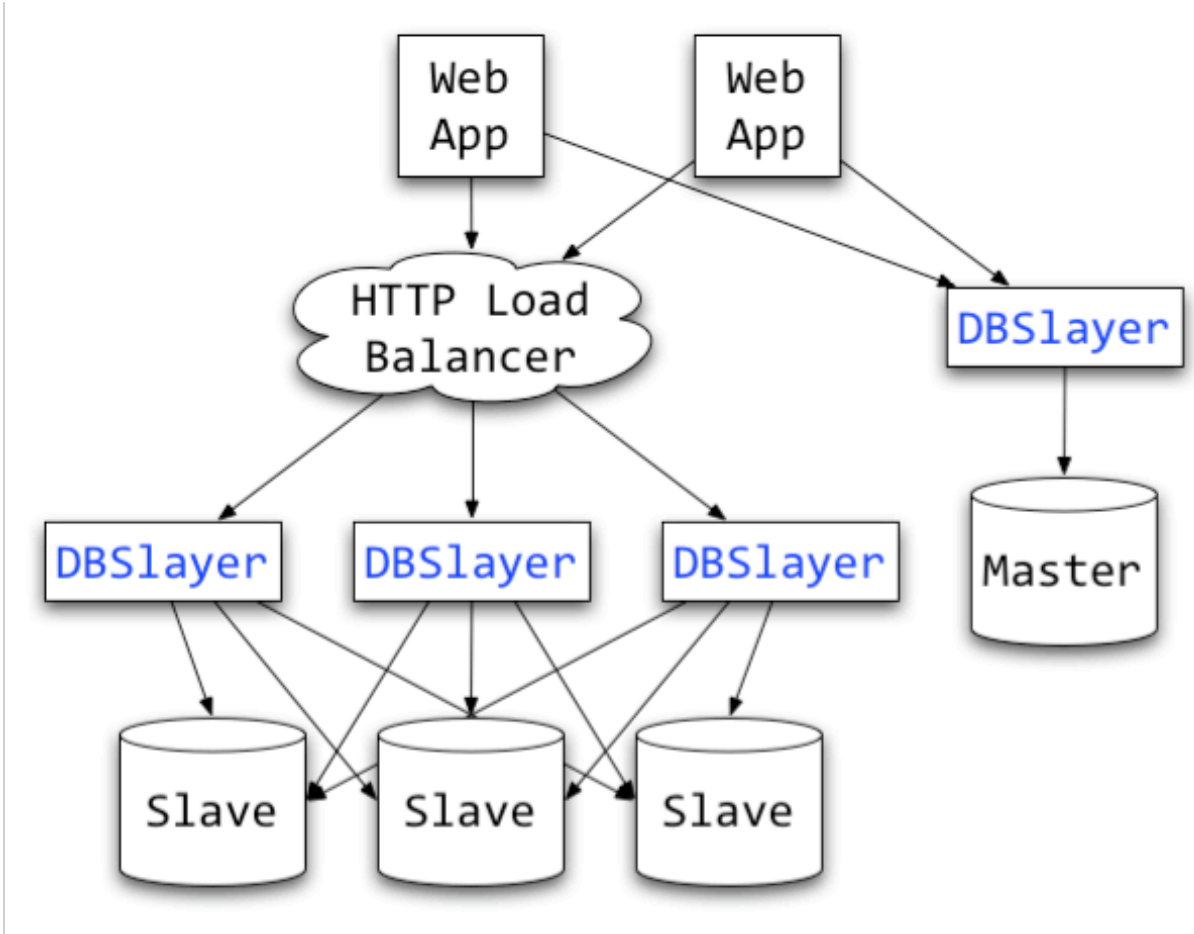


More Scenarios

Standard Version



Uh now we are talkin krazy ...



Notes From Real World Production

It works - We really use it. We actually see considerable traffic.

Deployed on Solaris x86, Linux x86-32 and x86-64, (also builds on Mac I am told)

Easier debugging

It wasn't always %100 bug free

Future of the DBSlayer

Improve HTTP implementation

Alternate DB back ends - Oracle is the most likely
Scripting / Filter Option - branch
Security - basic digest
Enhanced Stats and Reporting

Integrated Result Caching

View Source

Multi-threaded server written in C built with

- Apache Runtime (apr) library
- MySQL Client Libs
- Glue Code

Small Code base ~1.5k loc

Very Hackable

DBSlayer Resources

DBSlayer Site
http://code.nytimes.com/dbslayer - svn, tarballs, documentation

NYTimes Blog w/ DBSlayer Info
http://open.nytimes.com/ - developer blog

The End