FASTER PHP APPS USING

QUEUES & WORKERS

PHP DORSET 6/6/16

RICHARD BAKER


twitter.com/r_bake_r


github.com/rjbaker

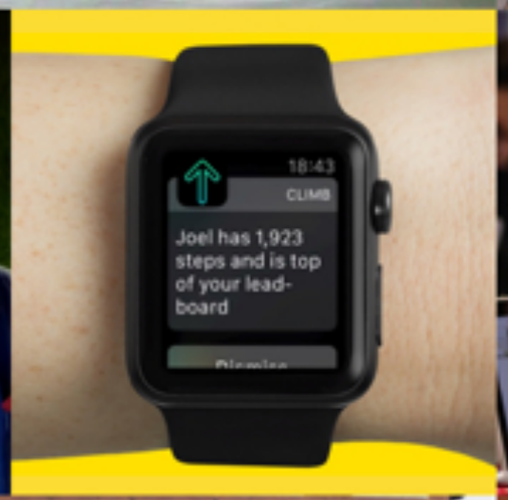

uk.linkedin.com/in/richjbaker

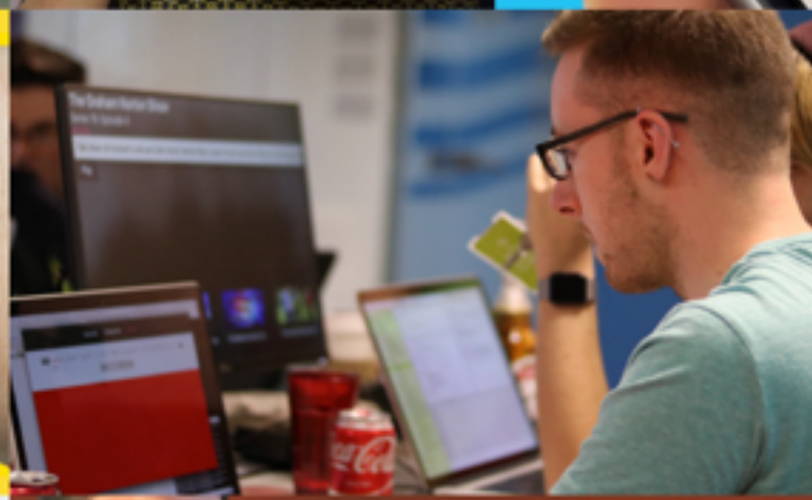
3 SIDED CUBE

We're hiring!



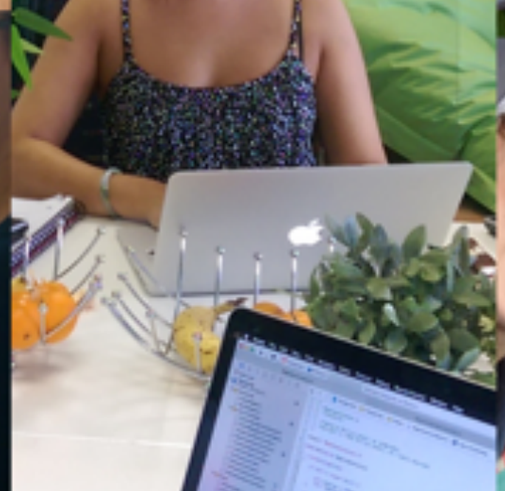






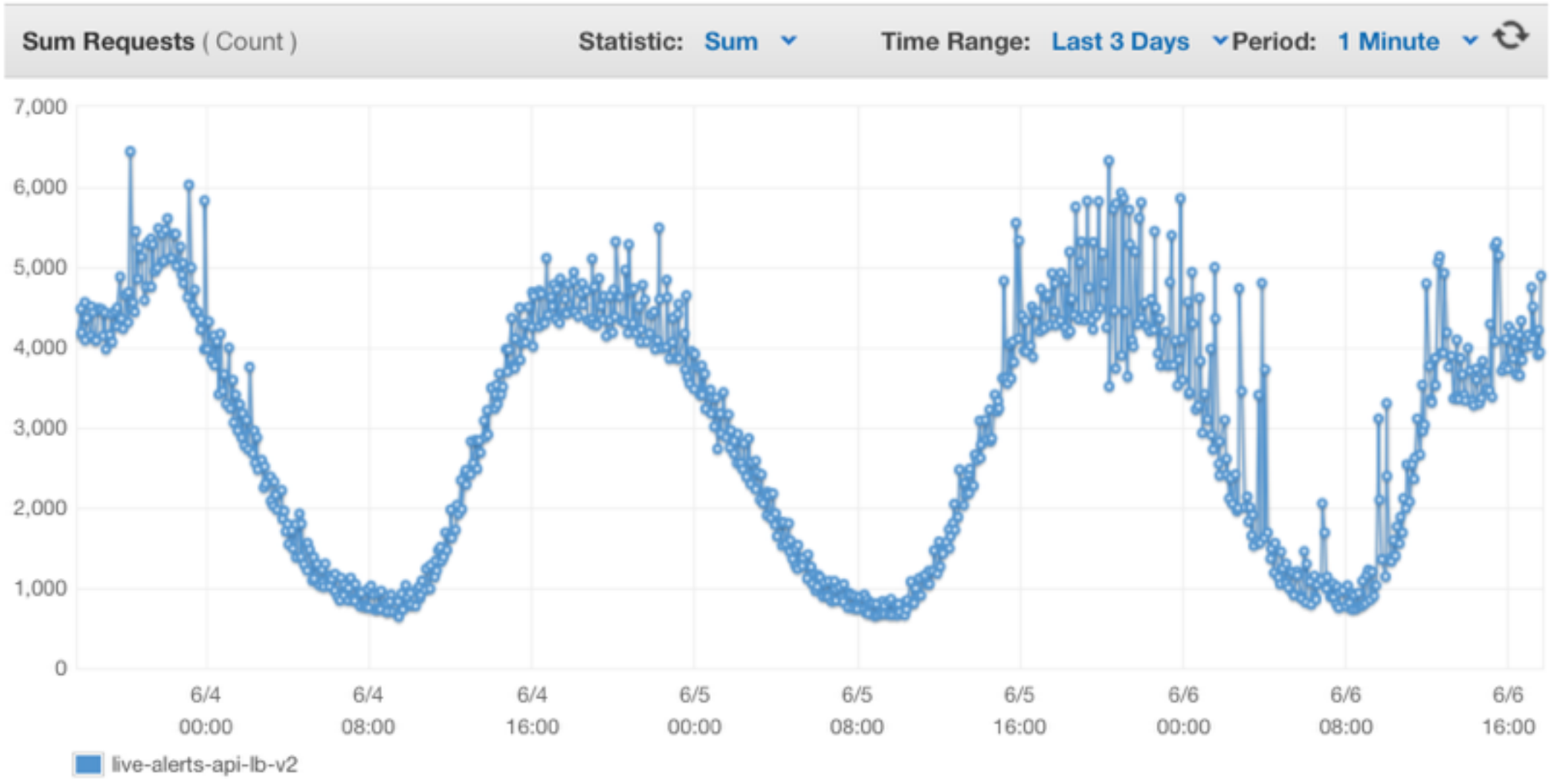




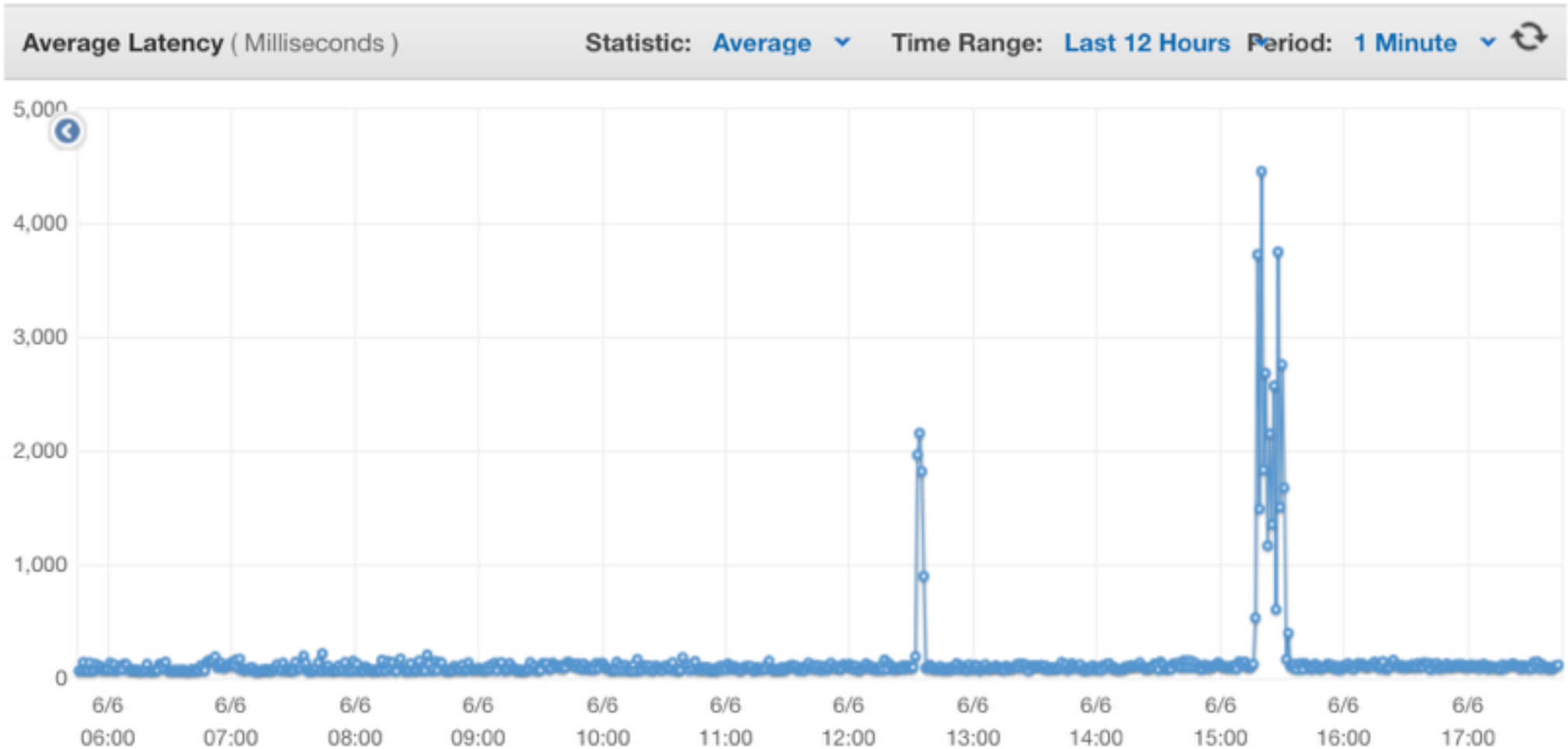

MOBILE API

- 140+ apps
- ~4.5m users
- REST/JSON
- Apache + PHP + ElasticSearch
- 20+ servers / various tasks
- Hosted on AWS

TRAFFIC


Sum Requests (Count)
Statistic: Sum
Time Range: Last 3 Days
Period: 1 Minute
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
6/4 00:00
6/4 08:00
6/4 16:00
6/5 00:00
6/5 08:00
6/5 16:00
6/6 00:00
6/6 08:00
6/6 16:00
live-alerts-api-lb-v2

LATENCY GROWS



ELIMINATE EXPENSIVE, TIME CONSUMING OPERATIONS AND CALLS TO EXTERNAL SERVICES

USE MYSQL

MYSQL AS A QUEUE

job_id	job_name	data	status
1	sendEmail	{payload}	completed
2	sendPush	{payload}	processing
3	requestThing	{payload}	waiting
4	resizeSelfie	{payload}	waiting

WHY NOT USE TRANSACTIONS TO OBTAIN A LOCK?

THE DATABASE IS NOT A QUEUE. THE DATABASE IS NOT A QUEUE.

Stephen Corona

USE A QUEUE!

MESSAGE QUEUES VS JOB QUEUES

- Kind of similar
- Most job queues built upon some kind of message queue
- Broker messages between systems
- Provide transport, storage and protocol
- Job queues abstract the lower level message component
- Integrate with most applications fairly easily

JOBS VS SCHEDULED TASKS

- Run at a predefined point in time ⏰
- May repeat at regular intervals or according to calendar 📆
- Typically triggered by cron or other scheduler 🔫
- Scheduled tasks can trigger jobs! 💥

CHOOSE A QUEUE WITH FEATURES BEST SUITED TO YOUR APPLICATION

CONSIDERATIONS

- Job priority & time sensitivity
- Job ordering and consistency (FIFO)
- Payload size limits
- Message data type & protocol
- Support for other languages / client libraries
- Failure management / retry policy
- Fault tolerance & redundancy
- One-time delivery guarantee
- Monitoring & statistics
- Distribution by task to specific workers e.g. Video encoding

BEANSTALKD

- Protocol similar to Memcached
- Clients need to know about all Beanstalkd servers (like memcached!)
- Beanstalkd servers can persist jobs, handle restarts without losing jobs
- Uses “tubes” to differentiate different queues
- Supports job TTR. Failed/hung jobs get put back into queue.
- Supports blocking. Client connects and waits for a new job.
- Requires setup and maintenance
- Loads of client libraries

http://kr.github.io/beanstalkd/

AMAZON SQS

- SAAS - Already using AWS, literally no setup
- Massively redundant, cheap, maintenance free
- HTTP/JSON under the hood, simple
- Supports long-polling.
- Best effort FIFO (no guarantees)
- No concept of job priority. Use different queues.
- Retry policy allows jobs to reappear in queue if not completed in specified time
- Configurable number of retries
- Queue stats and alarms integrate with autoscaling
- Scale worker instances based on queue length/backlog/rate

https://aws.amazon.com/sqs/

OTHER POPULAR QUEUES

- Celery (backed by RabbitMQ) - http://www.celeryproject.org
- php-resque (backed by Redis) -https://github.com/chrisboulton/php-resque
- Kafka - http://kafka.apache.org
- Gearman - http://gearman.org
- Iron.io (SAAS) - https://www.iron.io
- Loads more - www.queues.io

PROCESSING JOBS

WORKER PROCESS

- Essentially an infinite loop
- Executed on command line
- Asks queue for new job
- Resolves job method
- Execute with payload
- Delete job from queue
- Repeat

PROCESSING JOBS

```php
<?php

$queue = new Queue();

while(true) {

    $job = $queue->pop('queue-name');

    try {
        if ($job->execute()) {
            $job->delete();
        } else {
            $job->release();
        }
    } catch (\Exception $e) {
        $job->release();
    }
}
```

IMPROVING THE WORKER

```
pcntl_signal_dispatch();
```

PROCESS CONTROL EXTENSIONS (PCNTL)

- Respond to unix process signals
- Gracefully stop worker processes
- Complete current job before exiting
- Careful if using Apache mod_php on same server
- http://php.net/manual/en/book.pcntl.php

IMPROVED WORKER

```php
<?php

namespace Demo;

use Demo\Queue\QueueInterface;

class Worker
{
  protected $shouldRun = true;

  protected $queue;

  public function __construct(QueueInterface $queue)
  {
    declare(ticks = 1);

    $this->queue = $queue;

    pcntl_signal(SIGTERM, [$this, 'signalHandler']);
    pcntl_signal(SIGINT,  [$this, 'signalHandler']);
    pcntl_signal(SIGQUIT, [$this, 'signalHandler']);
  }
```

```php
public function run($queueName)
  {
    echo "Starting worker on queue '{$queueName}' \n";

    while ($this->shouldRun) {

      $job = $this->queue->pop($queueName);

      try {
        if ($job->execute()) {
          $job->delete();
        } else {
          $job->release();
        }
      } catch (\Exception $e) {
          $job->release();
          error_log($e->getTraceAsString());
      }
        pcntl_signal_dispatch();
    }
  }

  public function signalHandler($signal)
  {
    switch ($signal) {
        case SIGTERM:
        case SIGINT:
        case SIGQUIT:
            echo "Job completed. Exiting... \n";
            $this->shouldRun = false;
          break;
    }
  }
}
```

WTF IS DECLARE(TICKS = 1);?

- Officially deprecated
- Triggered after php has executed a certain number of statements
- Interacts with pcntl_signal_dispatch()
- I admit i've not fully tested this with PHP7.0

KEEPING WORKERS RUNNING

SUPERVISOR

- Process manager in similar vein to forever, pm2, php-fpm
- Runs as service
- Starts and restarts php worker processes
- Has CLI client (supervisorctl)
- Web interface
- Easy to install and configure

http://supervisord.org

SUPERVISOR CONFIG

```
[program:alertworker]
command = /usr/bin/php /path/to/queueRunner.php -q=prod-alerts
autorestart = true
autostart = true
directory = /path/to/scripts
environment = DEPLOYMENT='production'
exitcodes = 0,2
numprocs = 1
numprocs_start = 0
priority = 999
startretries = 3
startsecs = 4
stderr_capture_maxbytes = 1MB
stderr_events_enabled = false
stderr_logfile = AUTO
stderr_logfile_backups = 10
stderr_logfile_maxbytes = 50MB
stderr_syslog = false
stdout_capture_maxbytes = 1MB
stdout_events_enabled = true
stdout_logfile = AUTO
stdout_logfile_backups = 10
stdout_logfile_maxbytes = 40MB
stdout_syslog = false
stopsignal = TERM
stopwaitsecs = 10
umask = 022
user = worker
```

DEMO TIME

THANKS FOR LISTENING!

twitter.com/r_bake_r

github.com/rjbaker

uk.linkedin.com/in/richjbaker