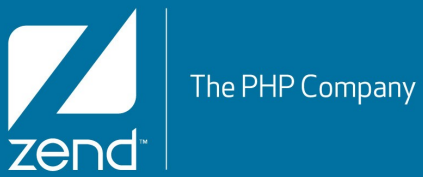


Job Queue in Zend Server 5.0

Shahar Evron
Technical Product Manager for Zend Server

Who am I?

- I am:
  - A PHP programmer since 2002
  - At Zend since 2005
  - Technical Product Manager for Zend Server
- Yes, I have a difficult name (at least for English speakers)
  - Shachar (German, Dutch)
  - Shajar (Spanish)
  - Шахар (Russian)
  - شخر (Arabic)
  - שחר (Hebrew)

Agenda

- What is Job Queue and what is it good for?
- A bit of how it works
- Using the API
  - Creating jobs, passing parameters
  - Accepting parameters
  - Schedule and recurrence
  - Reporting failure and success
- A quick tour of the administration interface
- For dessert: some benchmarks
- For those who want more...

What is Job Queue?

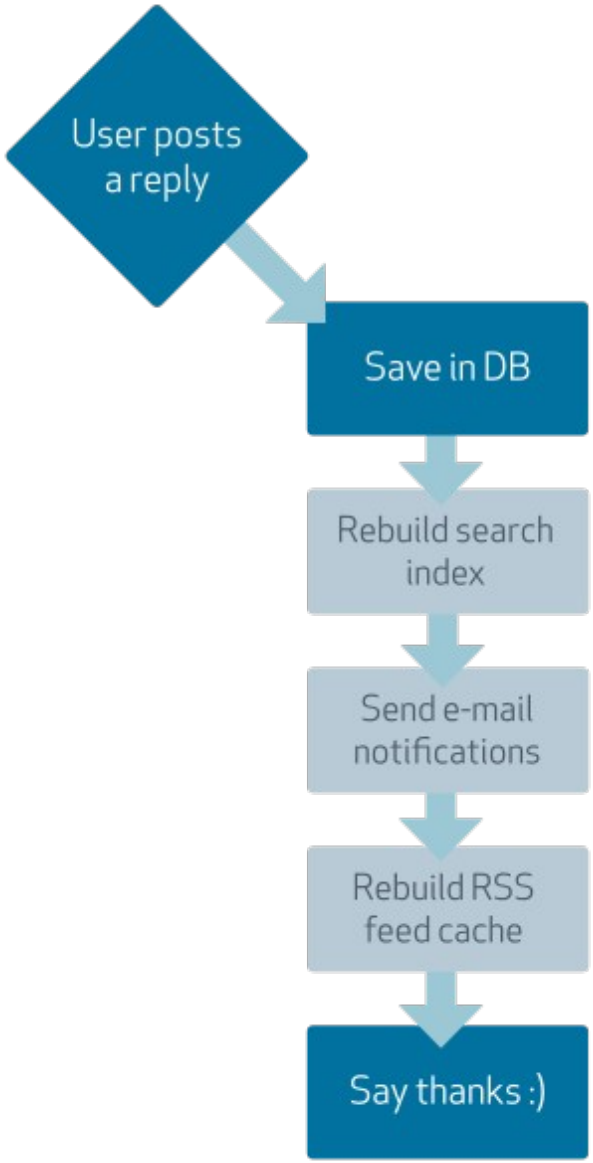
...and what is it good for?

zend

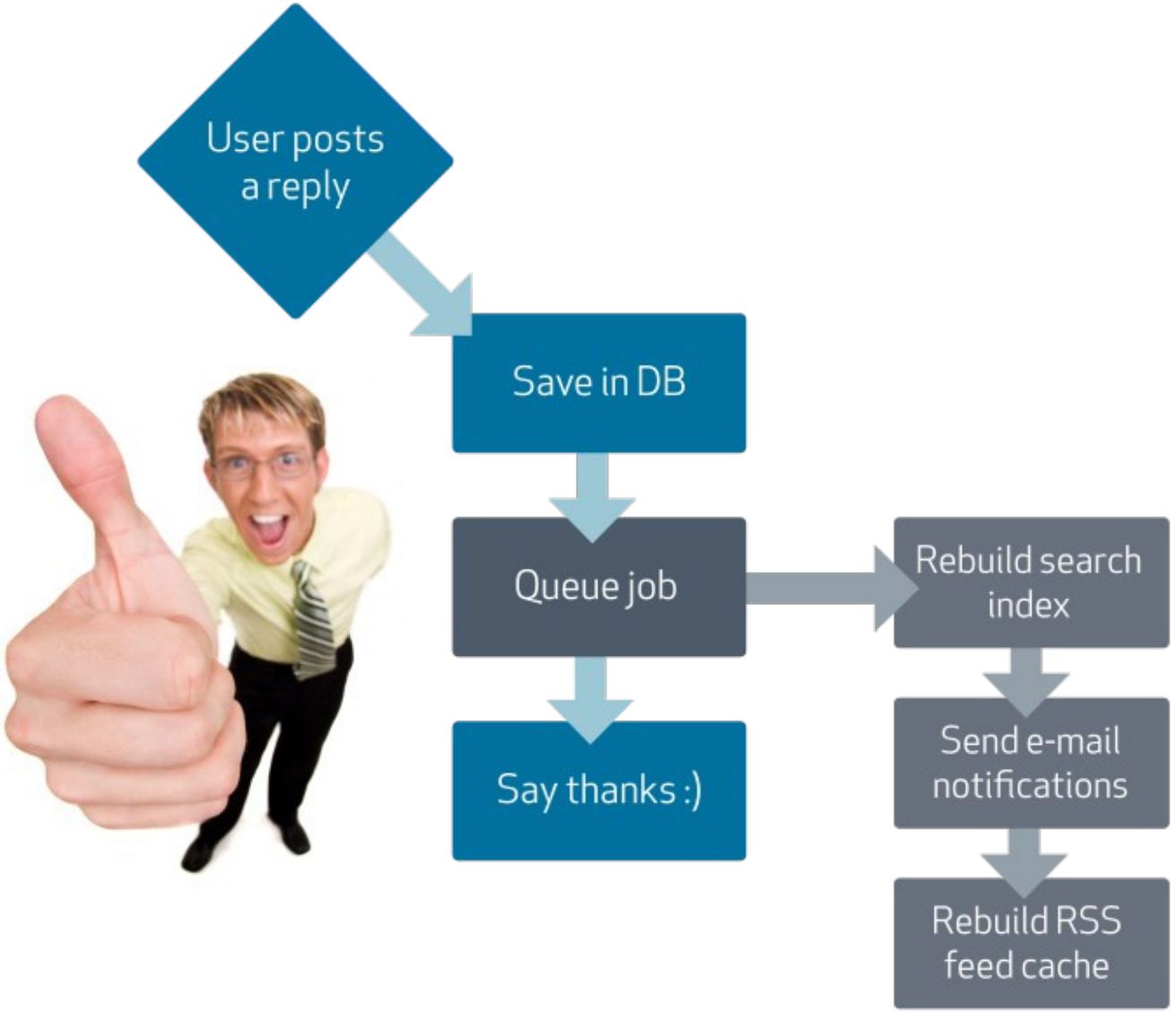
What are we trying to solve?

- Off-line processing?
  - Web applications tend to “live” in HTTP request/response cycles
  - What do you do when you need to take something off-line?
  - What do you do when you need periodical execution?
- It's also a matter of user experience:
  - Sometimes, it's just silly to let the user wait
- Zend Server Job Queue allows you to take it off-line!
  - Run things asynchronously, later, on a different server
  - Run things periodically

Example: an on-line forum



And now, with Job Queue!



Job Queue allows you to...

- Put certain tasks into a separate execution queue
  - Off-load to a later time (or even run in parallel)
  - Off-load to a different server
- Execute certain tasks at a specified time
  - Distribute processing load to off-hours
- Execute certain tasks periodically

- While..
  - Maximizing reuse of existing infrastructure & code
  - Making sure nothing falls between the cracks
  - Doing it all from a PHP API

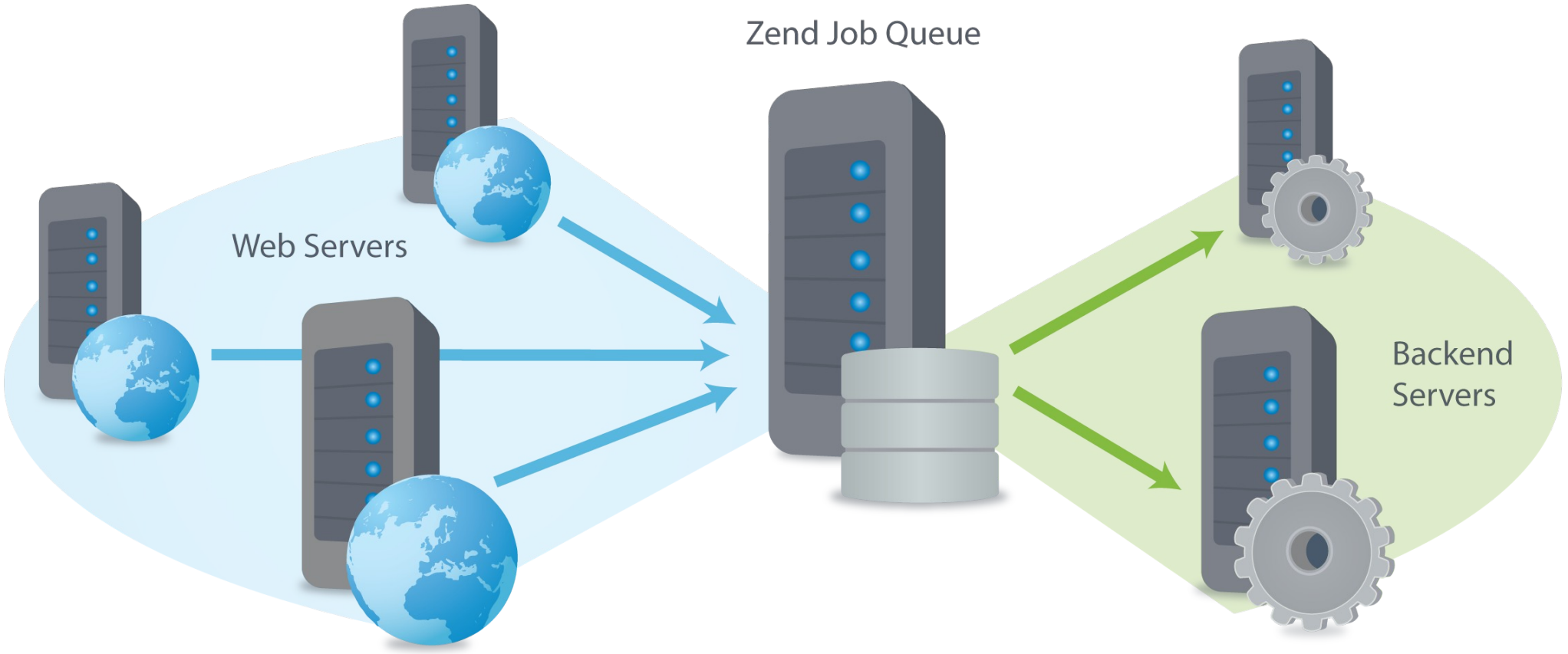
So, is it some glorified cron?

- No!
  - You can run things **now**, but without waiting for them to finish
  - You can run things **once**, but not right now
  - You can run things **periodically** (like cron)
    - But have full control over them – start, stop, suspend, resume from PHP API
  - Job Queue gives you full visibility into what's going on
    - Get alerts on failed jobs, analyze errors and re-queue
    - Keep track of past, current and pending jobs from the GUI
    - API for querying job status and handling failures
  - You don't need to hack it all to work for you

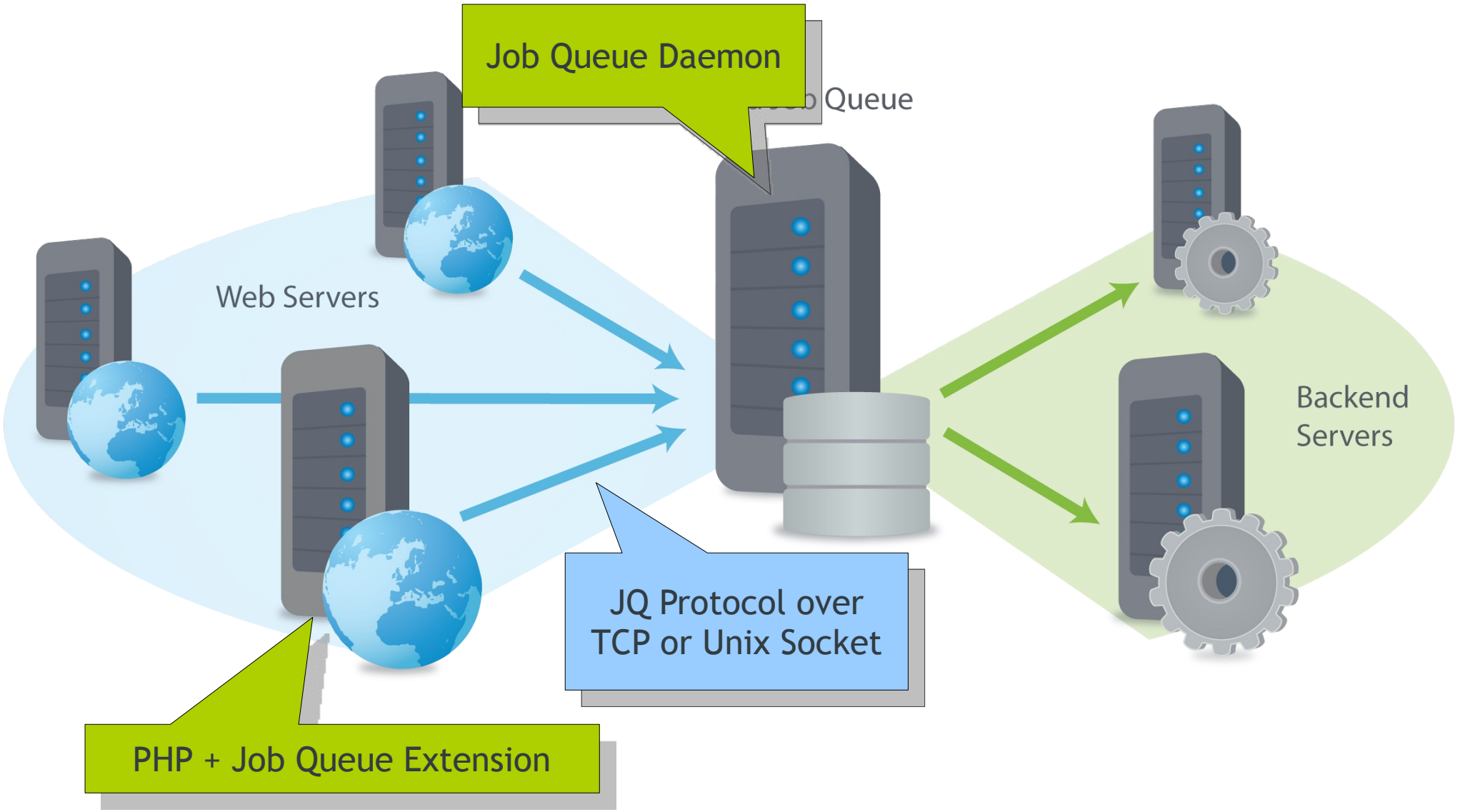
So how does it work?

Architecture and a bit of internals

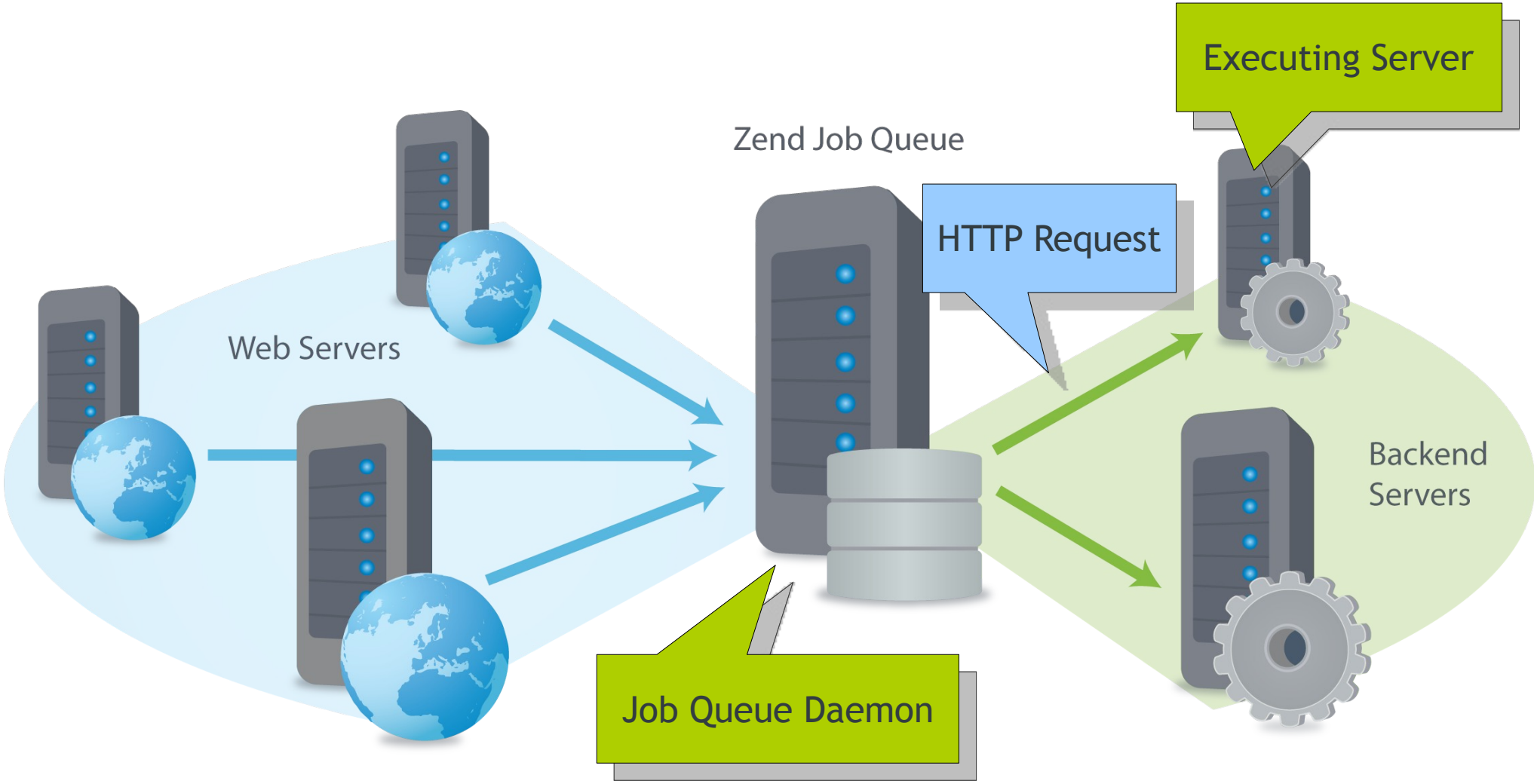
Job Queue 4.x - Architectural Overview



Job Queue 4.x - Architectural Overview



Job Queue 4.x - Architectural Overview



Using the Job Queue API

Creating & executing jobs

The *ZendJobQueue* class

- The *ZendJobQueue* class contains almost all the PHP API for Job Queue

- To preform most tasks, you will need to connect to a Job Queue server by instantiating a *ZendJobQueue* object:

```
// Connect to the default JQ server
$queue = new ZendJobQueue();

// Connect to any other JQ server
$queue = new ZendJobQueue("tcp://1.2.3.4:5678");
```

Creating Jobs

- Jobs are created using the *createHttpJob()* method

```
$queue = new ZendJobQueue();
$queue->createHttpJob(
    'http://backend.local/jobs/somejob.php');
```

- Passing a path instead of a full URL will create the job with *$_SERVER['HTTP_HOST']* in the host name

```
$jobPath = '/jobs/otherjob.php';

$queue->createHttpJob($jobPath);
// This is equivalent to:
$queue->createHttpJob('http://' .
    $_SERVER['HTTP_HOST'] . $jobPath);
```

Passing Parameters

- Simple parameters can be passed as part of the query string
  - These will be available inside the job in *$_GET*
- Complex parameters can be passed in the 2nd parameter of *createHttpJob()*
  - Pass an associative array of key => value pairs
  - Value can be any data representable by JSON
    - Null, booleans, strings, integers, floating point numbers
    - Indexed arrays (including nested arrays)
    - Objects and associative arrays have the same representation

Passing Parameters (example)

```
$params = array(
  'cart' => array(
    'items' => array(
      array('id' => 324, 'qty' => 1, 'price' => 19.95),
      array('id' => 75, 'qty' => 2, 'price' => 14.95,
        'size' => 'XL')
    ),
    'total'    => 49.85,
    'coupon'   => null,
    'giftwrap' => true
  ),
  'user' => $user
);

$queue->createHttpJob(
  'http://backend/jobs/checkout.php', $params);
```

Accessing Parameters

- Inside the Job code, use the *ZendJobQueue::getCurrentJobParams()* static method:

```
$params = ZendJobQueue::getCurrentJobParams();
var_export($params);
/* Output will be:
 array (
  'cart' => array (
    'items' => array (
      0 => array (
        'id' => 324,
        'qty' => 1,
        'price' => 19.95,
      ),
      ...
*/
```

- You can also *json_decode()* the raw POST body:

```
$params = json_decode(file_get_contents('php://input'));
```

Additional Job Options

- The 3rd parameter of *createHttpJob* is an associative array of options:
  - ***name***
  - ***priority***
  - ***persistent***
  - ***predecessor***
  - ***http_headers***
  - ***schedule***
  - ***schedule_time***

Creating Deferred Jobs

- You can set a job's (estimated) execution time by passing the *schedule_time* option:

```
// Process the form at 3:00 am
$runAt   = date('Y-m-d h:i:s', strtotime('+1 day 3:00am'));
$options = array(
  'schedule_time'  => $runAt
);

$queue->createHttpJob('http://backend/jobs/process.php',
  $_POST, $options);
```

- The job will not run **before** the specified time
  - Depending on queue load, it might run **after** it

Creating Recurring Jobs

- You can create a recurring job from API using the *schedule* option.
- This option takes a cron-like expression that specifies scheduling

```
// Run on Sunday, Monday, and Tuesday at midnight
$jq->createHttpJob('http://localhost/jobs/feed/405',
    null, array('schedule' => '0 0 * * 0,1,2'));

// Run every other day of the month at 2:30pm
$jq->createHttpJob('http://localhost/jobs/feed/405',
    null, array('schedule' => '30 14 */2 * *'));
```

Reporting Logical Failures

- A logical failure happens when something has programatically failed
  - e.g. failure to send an e-mail or connect to a SOAP web service
  - Different from an execution failure which is a technical failure (e.g. TCP error or HTTP 500 error from server)

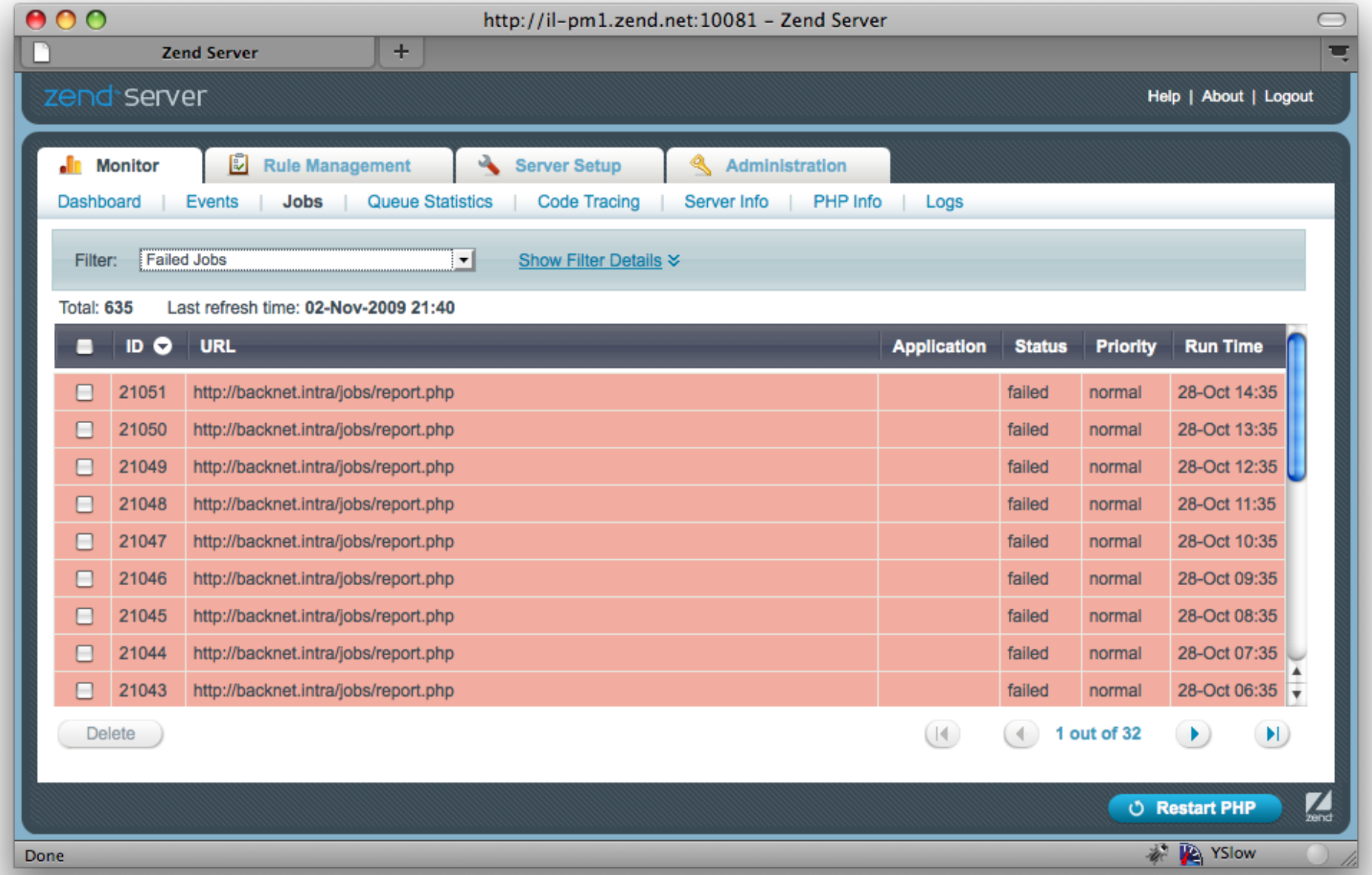
- Logical failures need to be reported programatically:

```
if (! $pp->sendPaymentRequest()) {
  // Report failure
  ZendJobQueue::setCurrentJobStatus(ZendJobQueue::FAILED,
    "Error from payment service: " . $pp->getError());
}

// Everything went well
ZendJobQueue::setCurrentJobStatus(ZendJobQueue::OK);
```

A Quick Tour of the UI

Demo Time!



Some Benchmarks

...so seriously, what can it do for me?

Sending Emails

- Requests/second on a simple e-mail sending script:

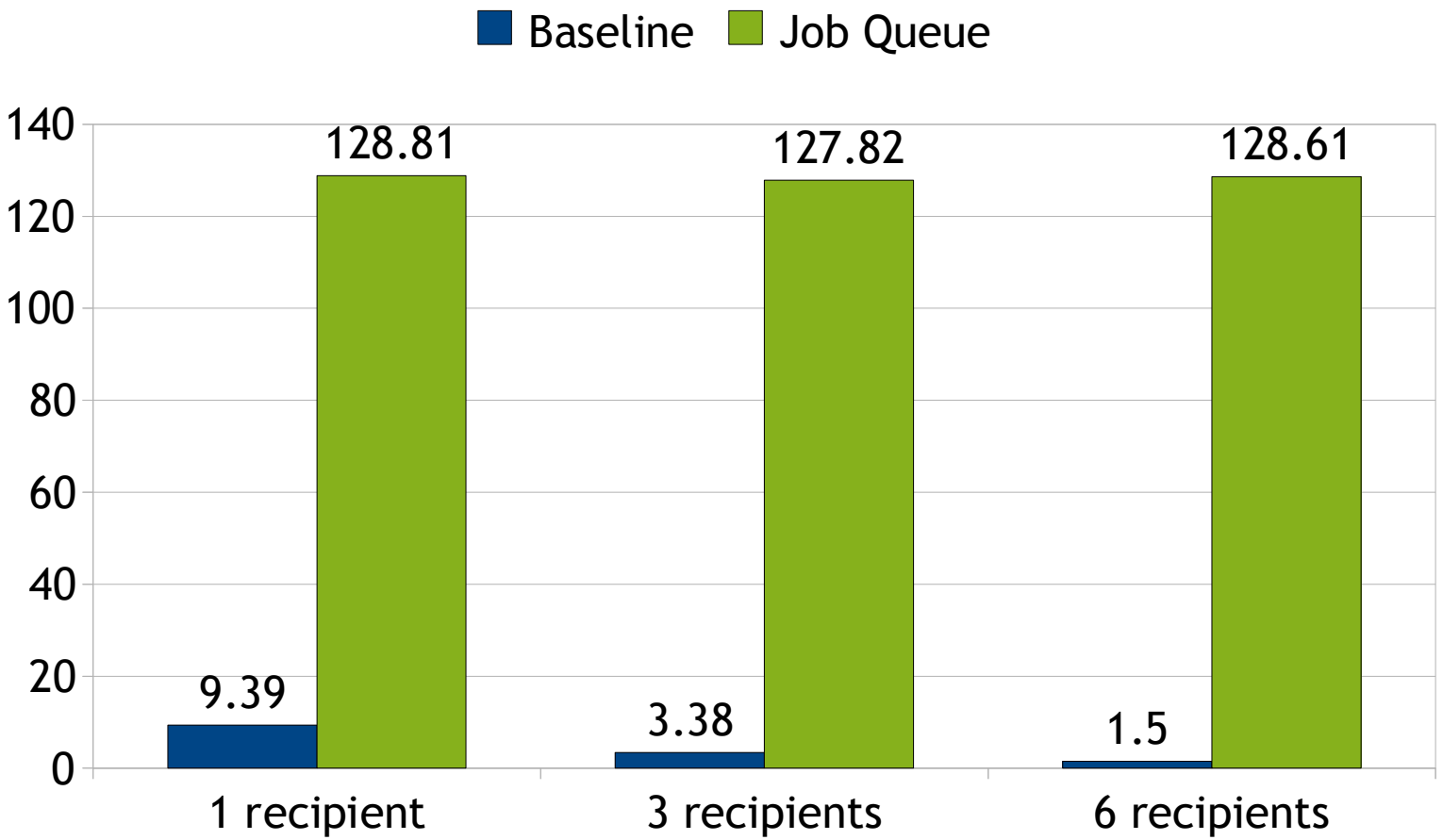
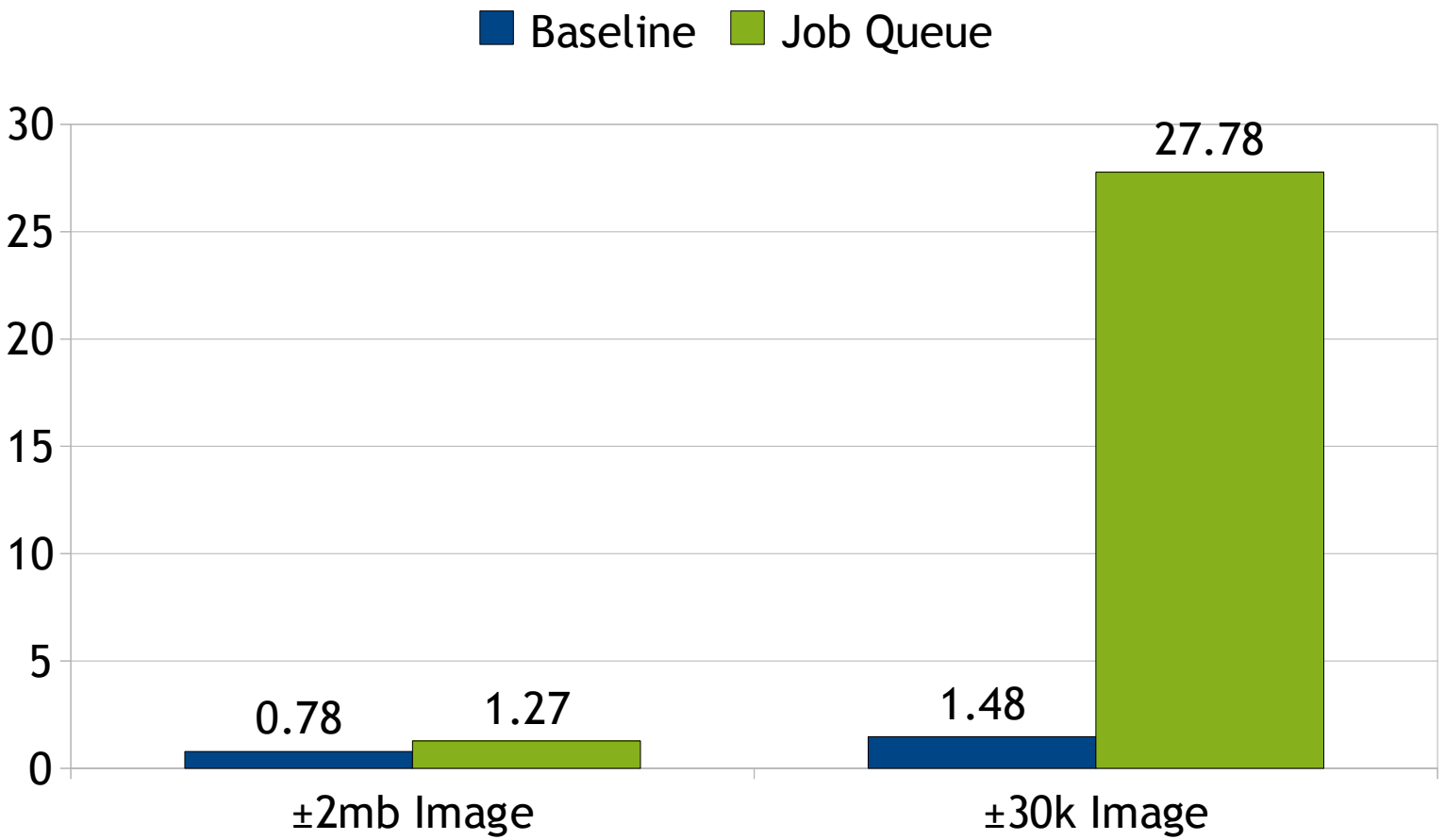


Image Conversion

- Image conversion using ImageMagick



Final Words

Where to go from here?

There's more!

- Managing priorities
- Job dependencies
- Querying for jobs
- Checking job status and queue statistics
- Suspending and resuming recurring jobs & queues
- Passing custom HTTP headers
- Failure handling and retry control
- Load management and balancing

You can try Job Queue today!

- Zend Server 5.0 Beta is available now
  - http://www.zend.com/server
- Documentation is available in the Zend Server 5.0 beta forum
  - http://forums.zend.com/
  - Got questions? The beta forum is exactly for that!
- Find me: shahar.e@zend.com,
  - @shevron on Twitter
  - #zendserver on FreeNode IRC

Thank You!

Using the Job Queue API

Querying for jobs and checking status

Querying for job status

- If you know the job ID, you can query for its status:

```
// When creating the job:
$jobId = $jq->createHttpJob($url, $params, $options);

// To check for the job's status (must be connected to the same queue)
$status = $jq->getJobStatus($jobId);
var_export($status);

// Output is:
array (
  'id'            => 126,
  'type'          => 1,
  'status'        => 4,
  'priority'      => 1,
  'persistent'    => false,
  'script'        => 'http://localhost/mail/job.php',
  'vars'          => '{"email":"shahar.e@zend.com"}',
  'output'        => 'HTTP/1.1 200 OK ...<snip>',
  'creation_time' => '2009-10-21 19:40:01',
  'start_time'    => '2009-10-21 19:41:03',
  'end_time'      => '2009-10-21 19:41:05',
);
```

Querying for jobs

- You can search the job list for jobs according to different parameters:

```
// Query for all failed "Sales Report" jobs
$jobs = $jq->getJobsList(array(
    'status' => ZendJobQueue::STATUS_FAILED,
    'name'   => 'Sales Report'
));

// Try to re-run failed jobs
foreach($jobs as $job) {
    echo "Attempting to re-queue failed job #{$job['id']}...\n";
    $jq->restartJob($job['id']);
}
```