A Very Brief Overview of

Gearman

Presented By
Jamie Pitts

What is Gearman?

A protocol and server for distributing work across multiple servers.

- ☐ Job Server implemented in pure perl and in C
- ☐ Client and Workers can be implemented in any language
- ☐ Not tied to any particular processing model
- ☐ No single point of failure

Current Maintainers: Brian Aker and Eric Day

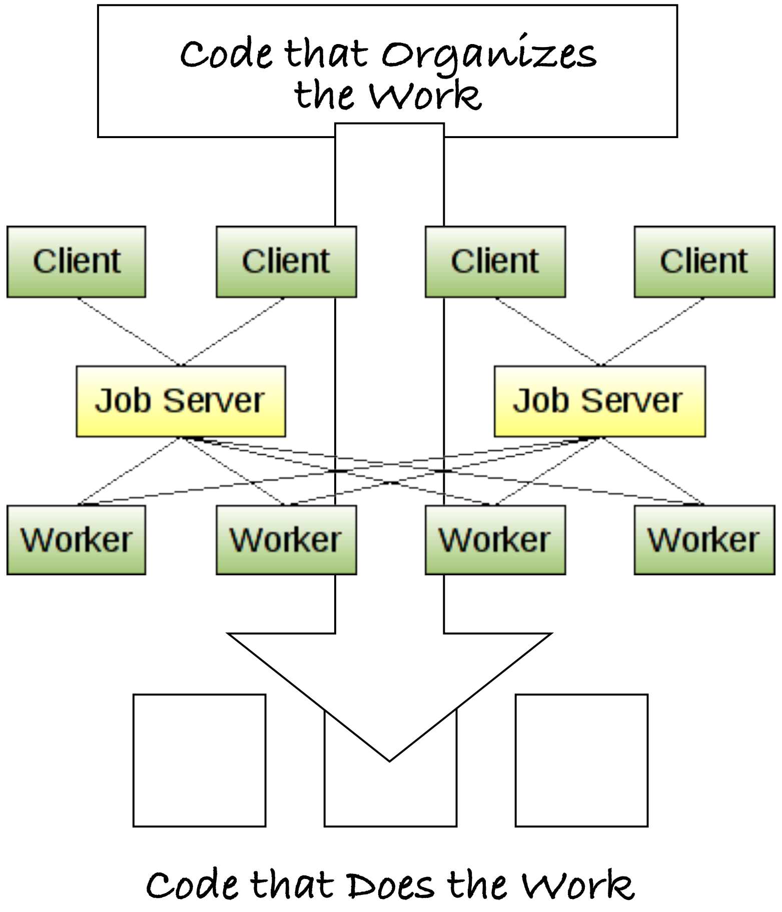
Facts About Gearman

- Created by Brad Fitzpatrick, who also created Memcached.
- The server and protocol were first implemented in perl, and then later re-implemented in C.
- First developed and used at LiveJournal.
- Deployed widely at Yahoo (before they created Hadoop).
- “Gearman” is an anagram for “Manager”

What Can You Do with Gearman

- ☐ Job Queue: from a web request, pass image, email sending, etc. to be processed the background.
- ☐ Pipeline Process: organize work in steps, monitor progress, notify when work is done or when something breaks.
- ☐ Multi-Language Environment: use Gearman as an intermediary between disparate systems.
- ☐ Map/Reduce: Break a dataset into parts, distribute the work, re-integrate the results.
- ☐ Combine any of the above...

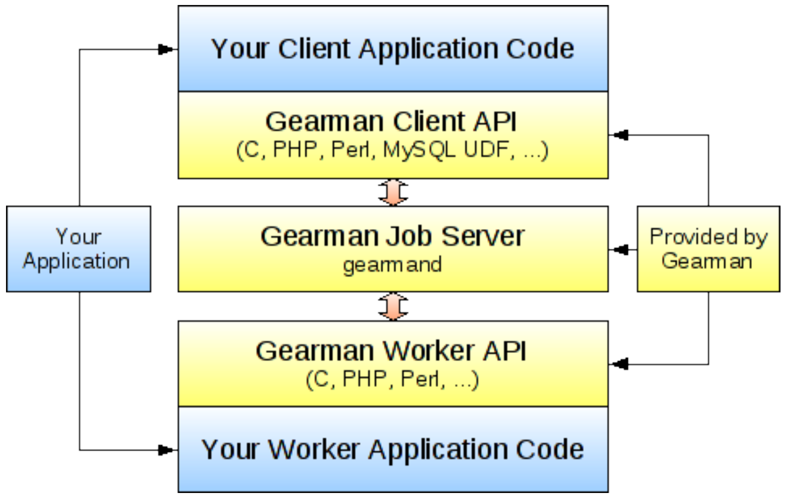
Gearman Architecture



- ☐ Job Server is the intermediary between Clients and Workers
- ☐ Clients organize the work
- ☐ Workers define and register work functions with Job Servers.
- ☐ Servers are aware of all available Workers.
- ☐ Clients and Workers are aware of all available Servers.
- ☐ Multiple Servers can be run on one machine.

Step by Step

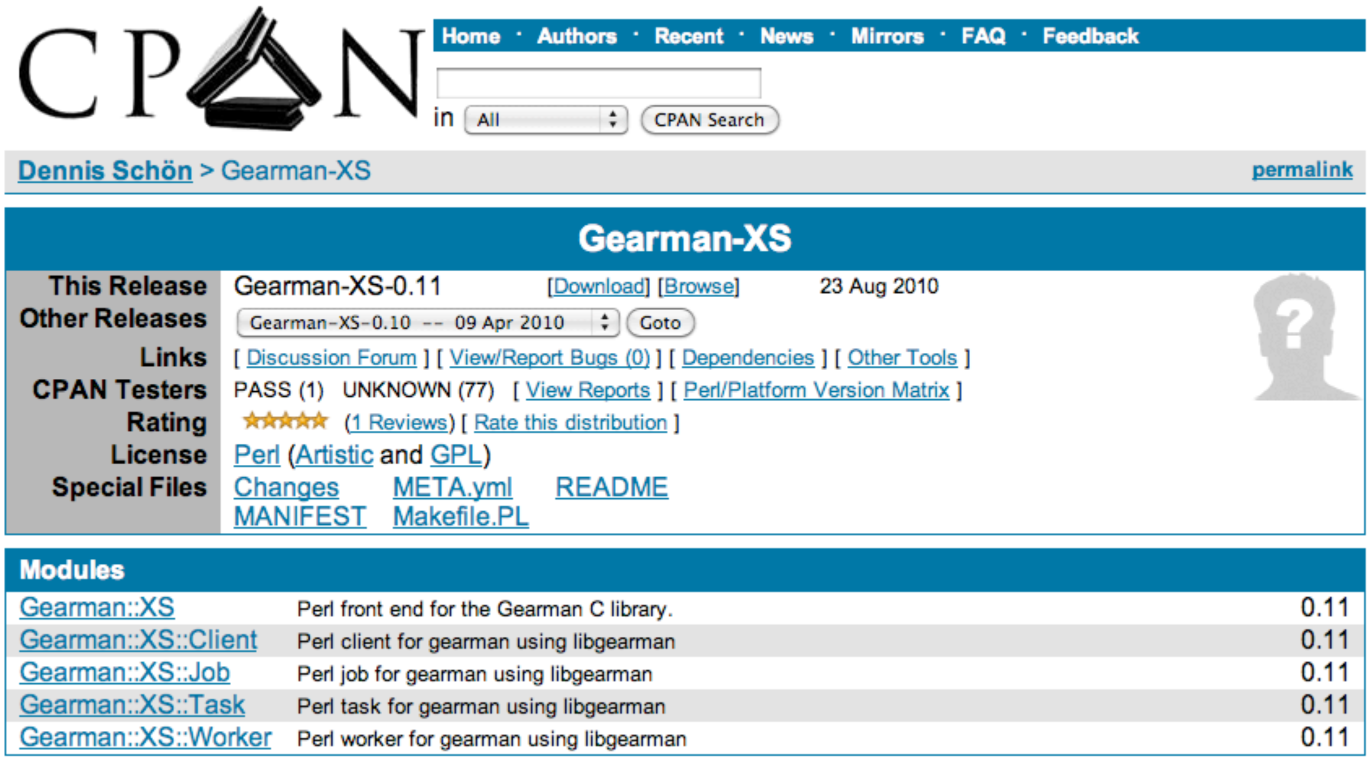
How Work is Completed



1. Client creates a job with a payload and submits it to a Job Server.
2. The Server finds an open Worker and forwards the job.
3. The Worker processes the payload, and can update the Server with progress. The Client can query the Server about progress.
4. When the job is finished, the Server may forward a new job to the Worker.

Perl and Gearman

Use Gearman::XS to Create a Pipeline Process



Dennis Schön > Gearman-XS permalink

Gearman-XS

This Release	Gearman-XS-0.11 [Download] [Browse] 23 Aug 2010
Other Releases	Gearman-XS-0.10 -- 09 Apr 2010 Goto
Links	[Discussion Forum] [View/Report Bugs (0)] [Dependencies] [Other Tools]
CPAN Testers	PASS (1) UNKNOWN (77) [View Reports] [Perl/Platform Version Matrix]
Rating	★★★★★ (1 Reviews) [Rate this distribution]
License	Perl (Artistic and GPL)
Special Files	Changes META.yml README MANIFEST Makefile.PL

Modules

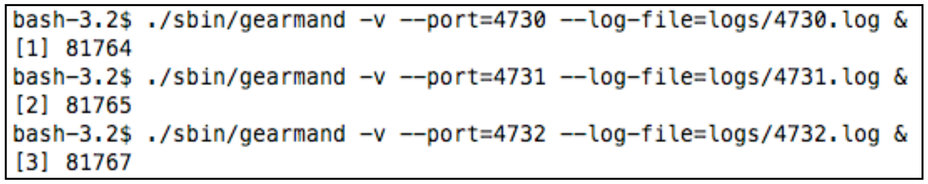
Gearman::XS	Perl front end for the Gearman C library.	0.11
Gearman::XS::Client	Perl client for gearman using libgearman	0.11
Gearman::XS::Job	Perl job for gearman using libgearman	0.11
Gearman::XS::Task	Perl task for gearman using libgearman	0.11
Gearman::XS::Worker	Perl worker for gearman using libgearman	0.11

http://search.cpan.org/~dschoen/Gearman-XS/

Perl and Gearman

Client-Server-Worker Architecture

Gearman Job Servers



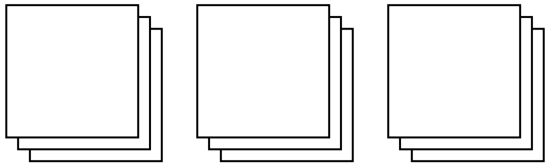
```
bash-3.2$ ./sbin/gearmand -v --port=4730 --log-file=logs/4730.log &
[1] 81764
bash-3.2$ ./sbin/gearmand -v --port=4731 --log-file=logs/4731.log &
[2] 81765
bash-3.2$ ./sbin/gearmand -v --port=4732 --log-file=logs/4732.log &
[3] 81767
```

Run one Job Server process per machine
(this setup is for experimentation)

Gearman::XS::Client

Run one Client process.
This will manage the Pipeline.

Gearman::XS::Worker



Each worker can perform any step.
Run many Workers per machine

Perl and Gearman

Pipeline Worker Implementation

```perl
use Gearman::XS qw(:constants);
use Gearman::XS::Worker;
use FreezeThaw qw(freeze thaw cmpStr safeFreeze cmpStrHard);

# set up the worker
my $worker = new Gearman::XS::Worker;
$worker->add_server('localhost', 4730);
$worker->add_server('localhost', 4731);
$worker->add_server('localhost', 4732);
```

Worker

```perl
# add the functions to the worker
$worker->add_function("step_1", 0, \&step_1, {});
$worker->add_function("step_2", 0, \&step_2, {});

# process jobs
while (1) {
    $worker->work();
}

# worker function: step 1
sub step_1 {
    my $job = shift;
    my ($workload) = thaw( $job->workload() );

    # ... do some heavy lifting ...

    return $result;
}

# worker function: step 2
sub step_2 {
    my $job = shift;
    my ($workload) = thaw( $job->workload() );

    # ... do some more heavy lifting ...

    return $result;
}
```

- ☐ Step 1 and Step 2 functions are added (registered).
- ☐ A Job is processed as it comes in from the Servers. Any function may be called.
- ☐ A Job with the workload is passed to the worker function.
- ☐ Result scalar is returned when done.

Perl and Gearman

Pipeline Client Implementation

```perl
use Gearman::XS qw(:constants);
use Gearman::XS::Client;
use FreezeThaw qw(freeze thaw cmpStr safeFreeze cmpStrHard);

# set up the client
my $client = new Gearman::XS::Client;
$client->add_server('localhost', 4730);
$client->add_server('localhost', 4731);
$client->add_server('localhost', 4732);
```

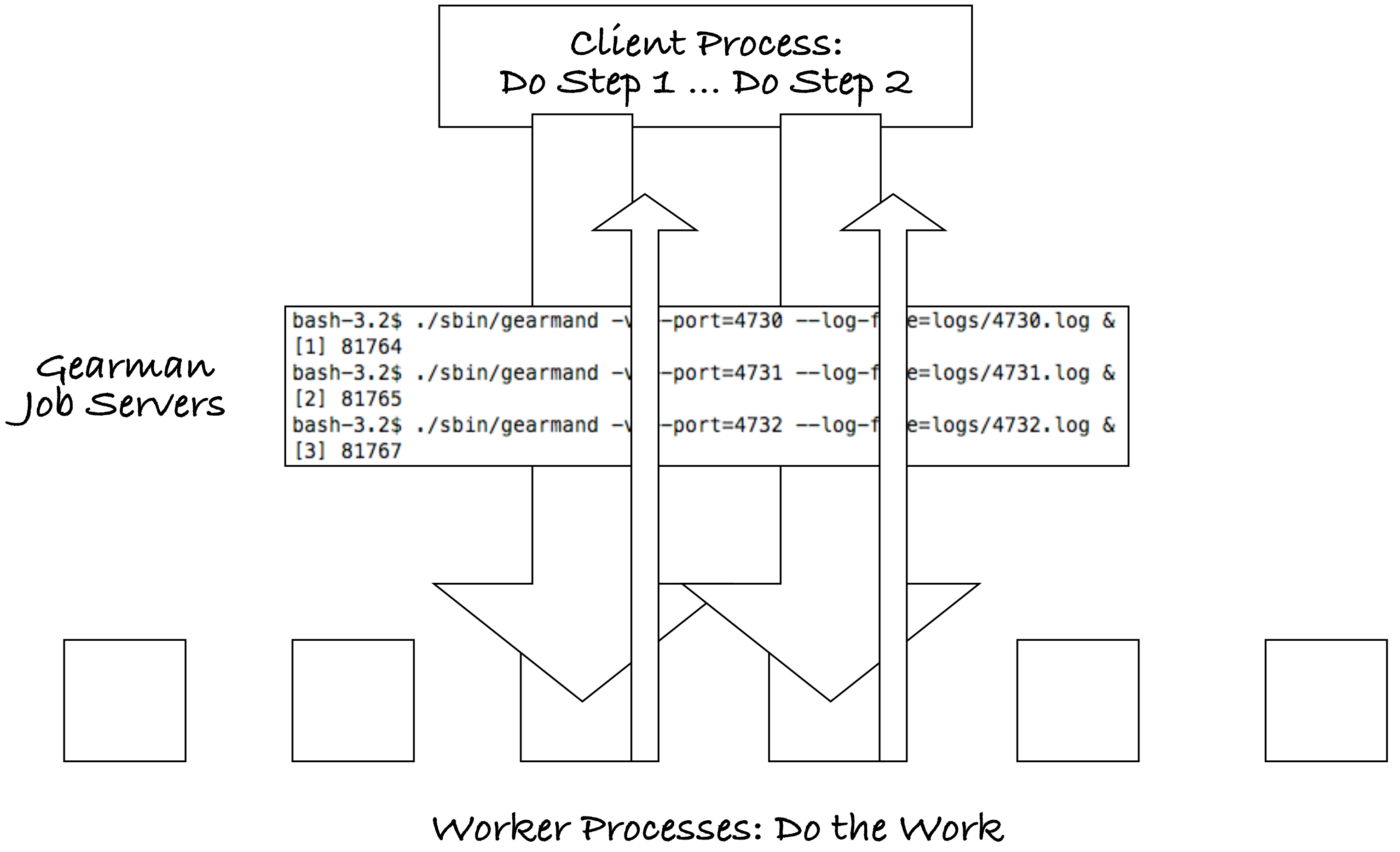
Client

```perl
# loop over the functions in the processing pipeline
foreach my $worker_function (qw/ step_1 step_2  /) {

    # submit a job to be performed by gearman workers
    my ($ret, $job_handle) = $client->do_background(
        $worker_function,   # 1. name of the worker function
        freeze( {           # 2. the workload, can be binary
            some_useful_id => $self->some_useful_id
        })
    );

    # monitor the progress of the job
    while (1) {

        # get the status of the job
        my ($return_value, $is_status_known, $status, $numerator,
            $denominator) = $client->job_status($job_handle);

        # check to see if this job is done
        last unless ($running_status);

        # sleep for 0.10 seconds
        select(undef, undef, undef, 0.10);

    }
}
```

- ☐ Outer loop: Names of Step 1 and Step 2 worker functions.
- ☐ A Job with a frozen payload is added to the Job Server queue.
- ☐ Inner loop: the Job is monitored using the unique $job_handle.
- ☐ The loop ends when the status is null, allowing the next stop of the Pipeline to begin.

Perl and Gearman

Processing Steps 1 and 2



```
bash-3.2$ ./sbin/gearmand -v   -port=4730 --log-f   e=logs/4730.log &
[1] 81764
bash-3.2$ ./sbin/gearmand -v   -port=4731 --log-f   e=logs/4731.log &
[2] 81765
bash-3.2$ ./sbin/gearmand -v   -port=4732 --log-f   e=logs/4732.log &
[3] 81767
```

That's It For Now

To Be Continued...

Presented By
Jamie Pitts