Kestrel

An XMPP-Based Framework for Many Task Computing Applications

Lance Stout
Mike Murphy
Sebastien Goasguen

HISTORY/PURPOSE

Kestrel's Goals

Lightweight / Easy to set up
- Run cross-platform without re-compiling
- No extensive, manual configuration
- Minimal dependencies

Detect Irregular Resource Outages
- Know quickly if a worker process terminates with kill -9

Traverse NAT

High Availability / Reliabilty

EXTENSIBLE MESSAGING AND PRESENCE PROTOCOL (XMPP)

XMPP Benefits

Presence notifications
- Always aware of the status of the worker pool
- Always receive unavailable status updates

Indirect Communication
- All messages sent through server
- NAT and subnet traversal

Identifiers
- Address workers without knowing IP addresses
- Workers can be grouped using JIDs

Jabber IDs username@server/resource

worker@kestrel_pool/42
(Only use to group small numbers of workers)

worker42@kestrel_pool/
(One username per worker is best for large pools)

machine27@kestrel_pool/core2

Messages

Kestrel uses JSON for message contents

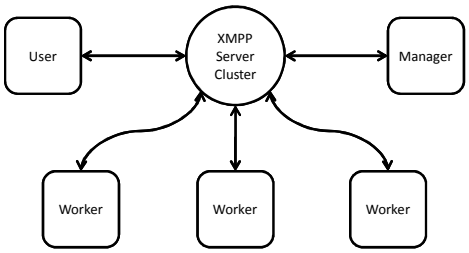
- Differentiated by "type" attribute
- Can be sent directly from an instant messaging client (GoogleTalk/Pidgin)

Example:

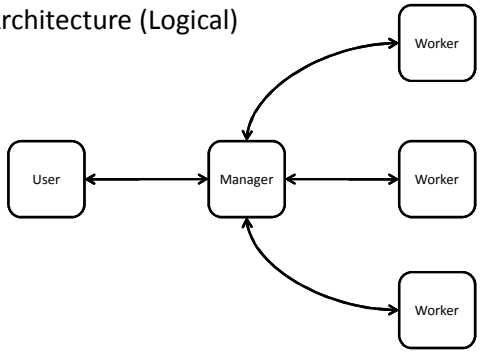
```
{"type": "profile",
 "os": "Linux",
 "ram": 4096,
 "cores": 4,
 "provides": ["FOO", "BAR"]}
```

ARCHITECTURE

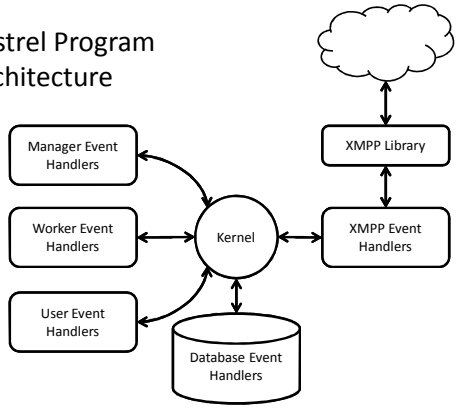
Kestrel Network Architecture (Actual)



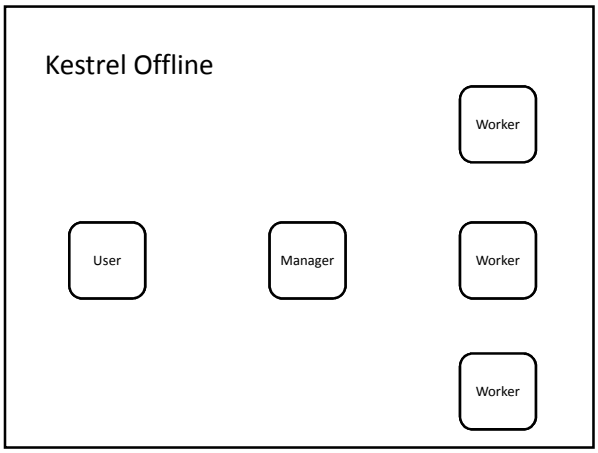
Kestrel Network Architecture (Logical)

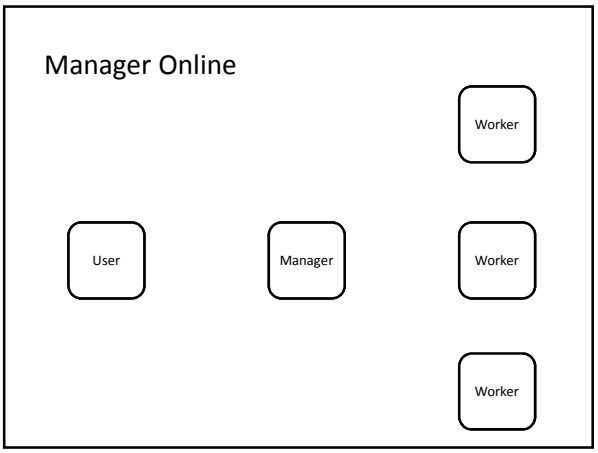


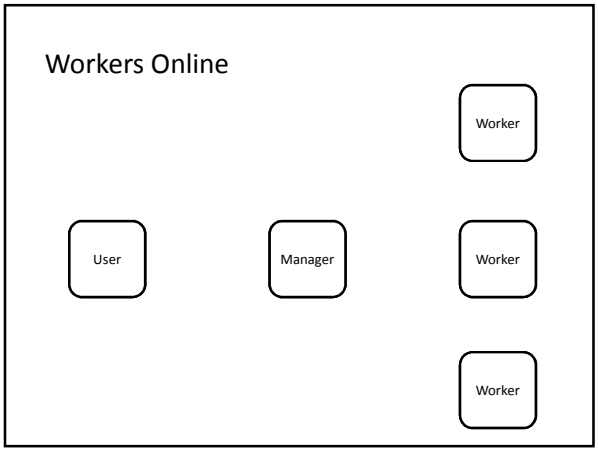
Kestrel Program Architecture

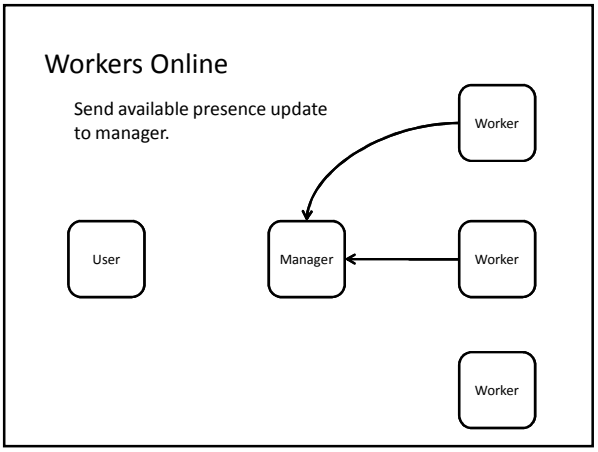


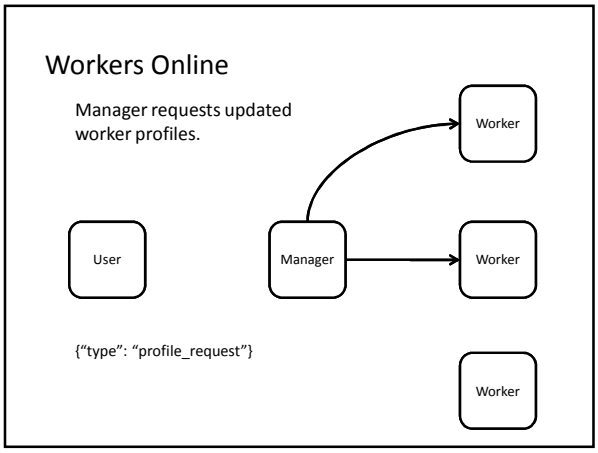
PSEUDO-DEMO

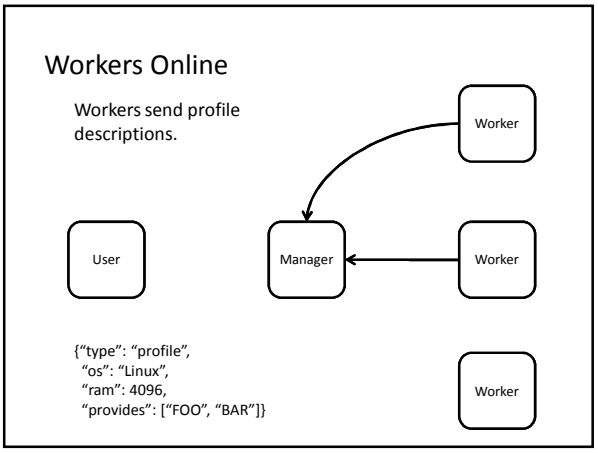

Kestrel Offline
Worker
User
Manager
Worker
Worker


Manager Online
Worker
User
Manager
Worker
Worker

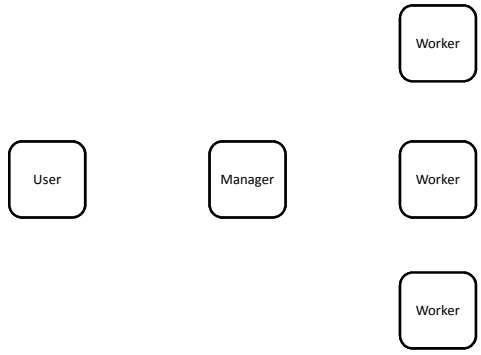

Workers Online
Worker
User
Manager
Worker
Worker


Workers Online
Send available presence update to manager.
Worker
User
Manager
Worker
Worker


Workers Online
Manager requests updated worker profiles.
Worker
User
Manager
Worker
{"type": "profile_request"}
Worker

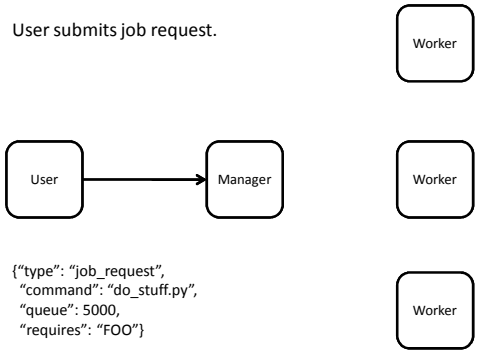

Workers Online
Workers send profile descriptions.
Worker
User
Manager
Worker
{"type": "profile",
"os": "Linux",
"ram": 4096,
"provides": ["FOO", "BAR"]}
Worker

User Online



Job Submission

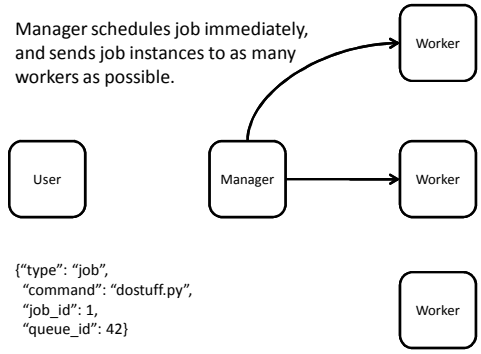
User submits job request.




```
{"type": "job_request",
 "command": "do_stuff.py",
 "queue": 5000,
 "requires": "FOO"}
```

Job Distributed

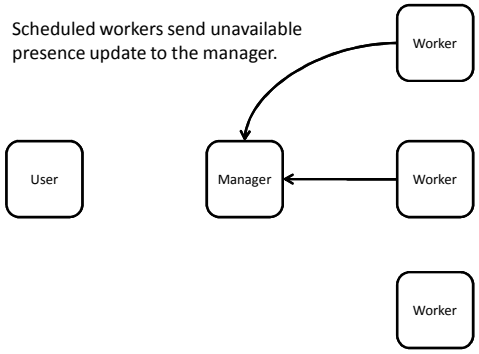
Manager schedules job immediately, and sends job instances to as many workers as possible.




```
{"type": "job",
 "command": "dostuff.py",
 "job_id": 1,
 "queue_id": 42}
```

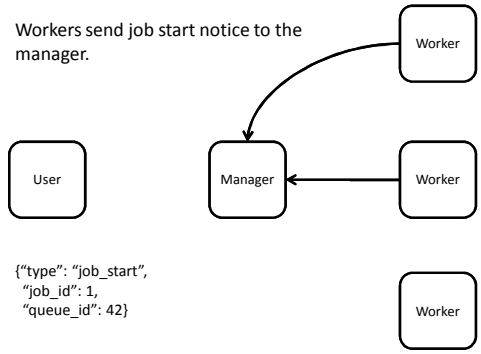
Job Execution

Scheduled workers send unavailable presence update to the manager.



Job Execution

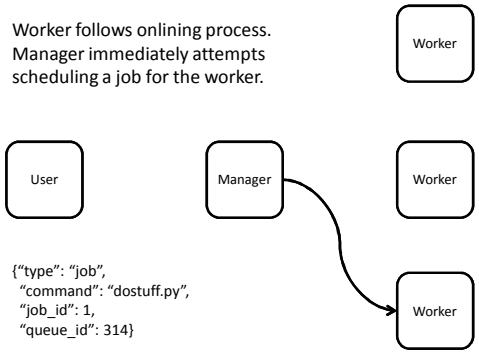
Workers send job start notice to the manager.




```
{"type": "job_start",
 "job_id": 1,
 "queue_id": 42}
```

Worker Added

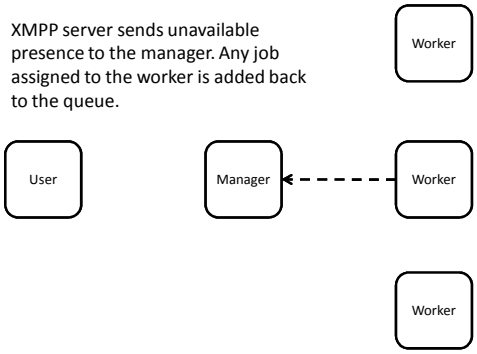
Worker follows onlining process. Manager immediately attempts scheduling a job for the worker.




```
{"type": "job",
 "command": "dostuff.py",
 "job_id": 1,
 "queue_id": 314}
```

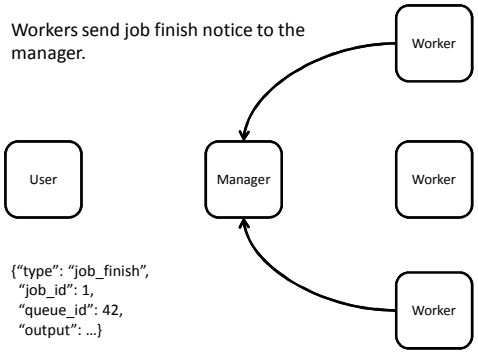
Worker Dropped

XMPP server sends unavailable presence to the manager. Any job assigned to the worker is added back to the queue.



Job Instance Finished

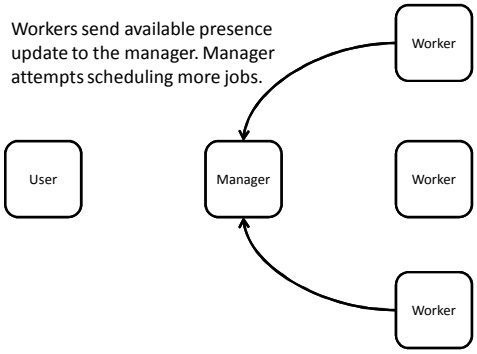
Workers send job finish notice to the manager.



{"type": "job_finish",
"job_id": 1,
"queue_id": 42,
"output": …}

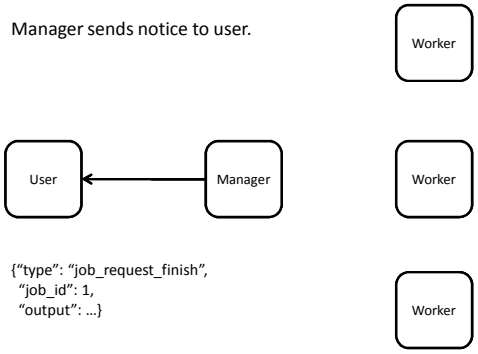
Job Instance Finished

Workers send available presence update to the manager. Manager attempts scheduling more jobs.



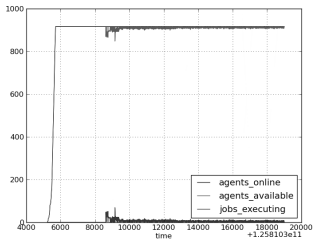
Job Finished

Manager sends notice to user.



{"type": "job_request_finish",
"job_id": 1,
"output": …}

RESULTS

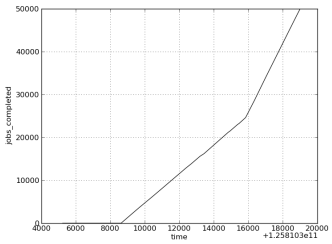


50,000 Jobs, 917 Workers, sleep 0

Finished in 103 seconds.

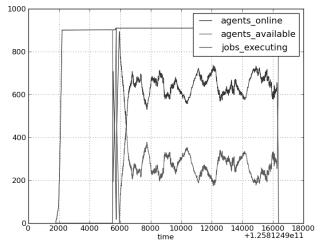
Dispatched 480 jobs per second.

Very few instances running concurrently due to short execution times.



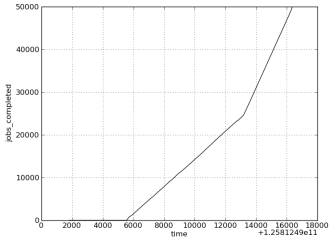
50,000 Jobs, 917 Workers, sleep 0

Why the uptick? We're still working on that one. Probably due to server processing faster as resources are freed.



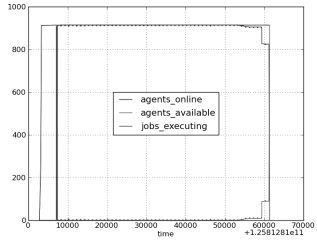
50,000 Jobs, 910 Workers, sleep 1

Finished in 108 seconds



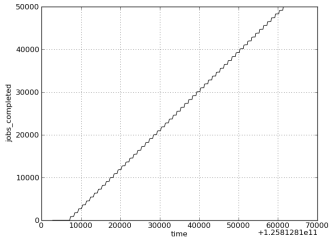
50,000 Jobs, 910 Workers, sleep 1

About the same as last time, including uptick.



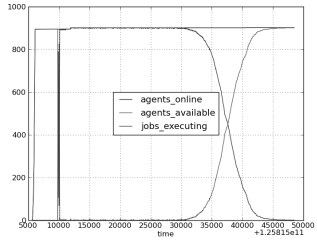
50,000 Jobs, 914 Workers, sleep 10

Finished in 543 seconds, or 9 minutes 3 seconds.



50,000 Jobs, 914 Workers, sleep 10

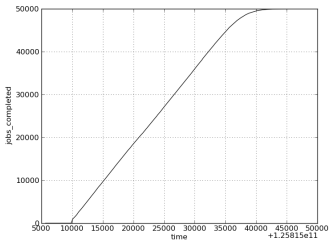
Longer, uniform execution time creates stair step pattern and no uptick.



50,000 Jobs, 902 Workers, sleep random

Jobs lasted between 0 and 10 seconds.

Finished in 387 seconds, or 6 minutes 27 seconds.



50,000 Jobs, 902 Workers, sleep random

No stair step pattern or uptick this time.