Running tasks in parallel with Celery

Irina Colgiu
Human Genetics Informatics
Wellcome Trust Sanger Institute

What is Celery?

Celery is a task management and queueing system

Celery vocabulary

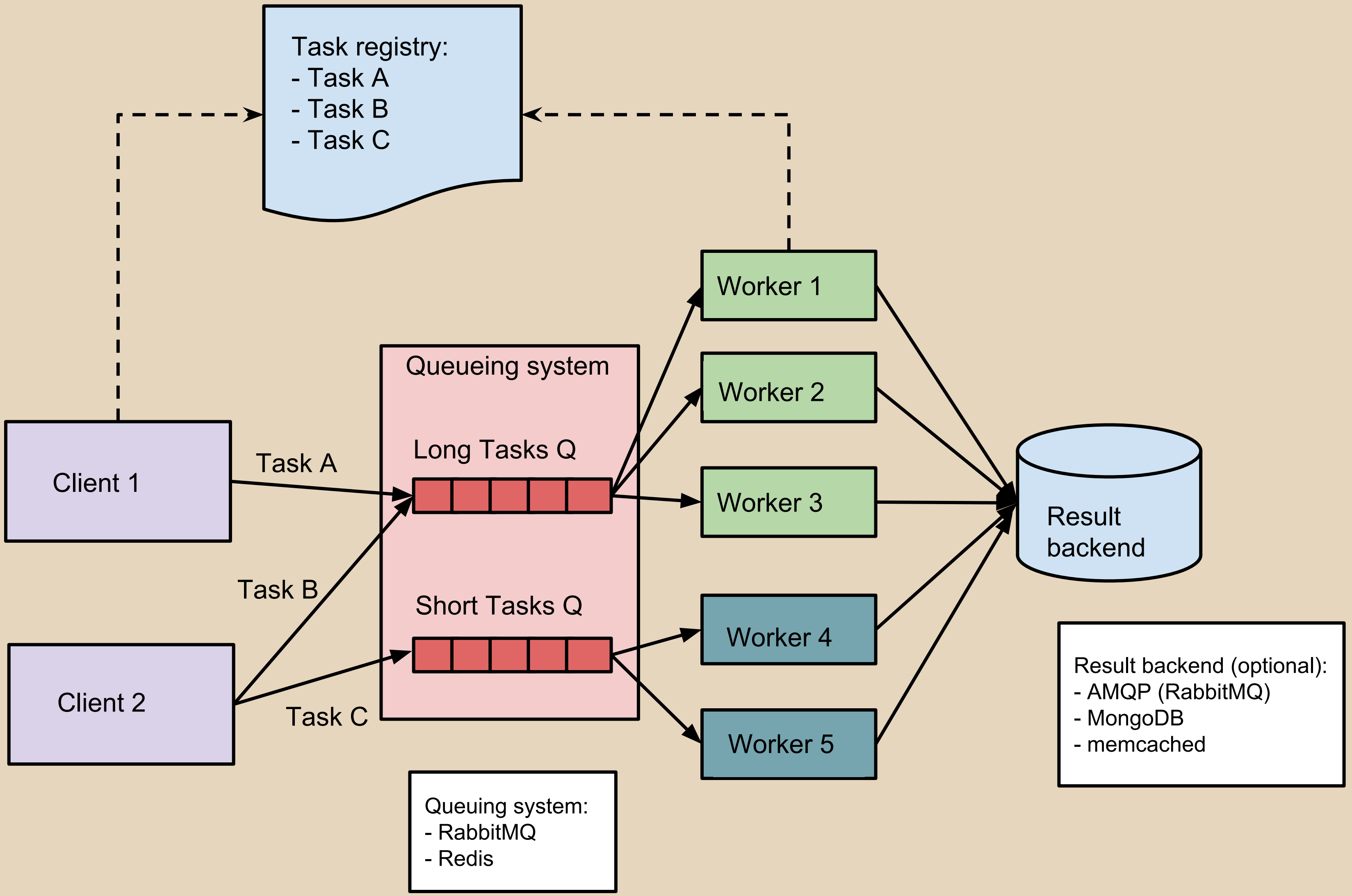
Task = a unit of work (a function defined by the user)

Worker = a process that executes tasks

What Celery offers

- A solution for executing work **async** (or sync) on **one/more machines, as one/more processes/threads remotely**
- Managing **task dependencies**
- **Dealing with task failure**
- **Handling machine failure**
- **Task execution updates** (useful for long-running tasks)
- **Collect results** from tasks
- **Monitoring** the tasks

Celery functionality



How to execute a task?

Defining a task:

```
broker = 'amqp://guest@machineA:5672'
celery = Celery('tasks',broker=broker)

@celery.task
def add(x, y):
    return x + y
```

Defining a results backend:

```
...
celery = Celery('tasks',
                broker=broker,
                backend='amqp')
...
```

Calling a task:

```
# ASYNC - the task executed in a worker
# process

async_result = add.apply_async((2, 2))
```

Collecting the results:

```
async_result.ready()
# True/False

async_result.state
# PENDING/STARTED/RETRY/SUCCESS/..

async_result.result
# Contains the return value of your task
```

How to start a worker?

```
$ celery worker -Q LongTasksQ
```

Useful options

- **Task routing** – one/more queues, the workers can listen to one/more queues

- **Task timeout:**
    - **soft limit** – the task is considered as failed if it doesn't finish before the timeout => requeued
    - **hard limit** – restart the worker

Useful options

- **Task retry** – configure the max retries, delay between retries, delay before the first retry

- **Rate limit** – limiting the nr of tasks to be executed in a given time frame

- **Autoscaling** – resize the worker pool depending on the load

```
$ celery worker --autoscale=3,10 -Q LongTasksQ
```

Limitations

- The **tasks MUST be predefined** - in the *tasks registry* by the time you start the worker processes

- You need to start and kill the workers “by hand” (unless you work in the daemon mode)

- Each type of results backend has its own strengths and weaknesses - e.g.: amqp (queues) - each task result is reported on its own queue

So what do I need to install?

```
$ pip install Celery
```

Either of:

- RabbitMQ
- Reddis
- Database (MongoDB)

as message broker & results backed.

Where to read more

http://www.celeryproject.org/

Thank you for your attention!

Questions?