PiCloud:
A simple approach to cloud computing

Ken Younge
Krannert School of Management
Purdue University

2013 New Directions in Text as Data Workshop
London School of Economics, September 27-29, 2013

Overview

- Amazon AWS + PiCloud.com

- Example projects
  - Patent-to-patent similarity — 7 million patents
  - USPTO office actions — 5 million rulings

- Example code
  - 1 job — One VM, 64GB RAM, a lot of storage
  - 1,000 jobs — A repetitive task, split up into 1,000 blocks
  - 100 jobs x 7 threads x multiple steps — 'Queues' to implement a pipeline

Amazon AWS

- Amazon sells access to virtual machines (VMs) running on the cloud
- The scale and rate of expansion of AWS drive down costs
- AWS requires considerable setup, configuration, and system admin
- AWS offers high-performance-computing (HPC) clusters, but you need HPC-coded solutions (e.g., molecular modeling, genome analysis, ...)
- AWS doesn’t provide an easy way to parallelize the kinds of custom, ad-hoc, cobbled-together programs we often run in text analysis

PiCloud

- PiCloud manages AWS for you
  - they provide a unified control panel to monitor execution across multiple computers
  - you can remotely monitor stdout, stderr, CPU, RAM, disk, swap, memalloc, etc... (but if you don't know how to do any of that - that's OK too).
- PiCloud provides a simple API to transfer data, execute code, and collect your results
  - Your core functions can be written in Python, R, C, C++, Java, MATLAB, Fortran, etc...
  - You can customize Linux environments to run anything that runs in Linux x86-64
  - You can call the API from Python or from a command line interface
  - There are many types of VMs (large/small RAM, fast/slow CPU, large/small disk, ...)

Capacity

- You can scale up to thousands of machines and unlimited storage
- You can reserve capacity when you know you need it
  ... but excess capacity is usually available

Cost

- AWS + PiCloud is cheaper than all of the small-, mid-, and super-computing-sized clusters I've reviewed
- You pay for what you use
- You can enter a hard-stop dollar limit to cover your downside

Example Project: Patent-to-patent vector space model

- Scrape content from the US Patent and Trademark Office
- Build a vocabulary space
- Vectorize every patent
- Save a sparse vector for each patent (small file)
- Calculate patent-to-patent cosine similarities as needed

Example Project: USPTO ‘office actions’

- Remotely mount 5 million ZIP archives
- OCR (tesseract) particular files from the archive
- Python-nltk the text
- Identify events in the text that we care about
- Build a directed graph of events

Example Code: Run one job

- Maybe you need more RAM
- Maybe you need your computer for something else
- Run it on the cloud:

```
import cloud
import nltk

def calculate():

    # insert code here that you want to run on the cloud
    # you can save results to cloud storage for download
    # or return the result(s) directly from the function

    return "All Done"

job_id = cloud.call(calculate, _type='m1', _cores=8)  # m1 with 64 GB RAM
```

Example Code: Run 1,000 jobs

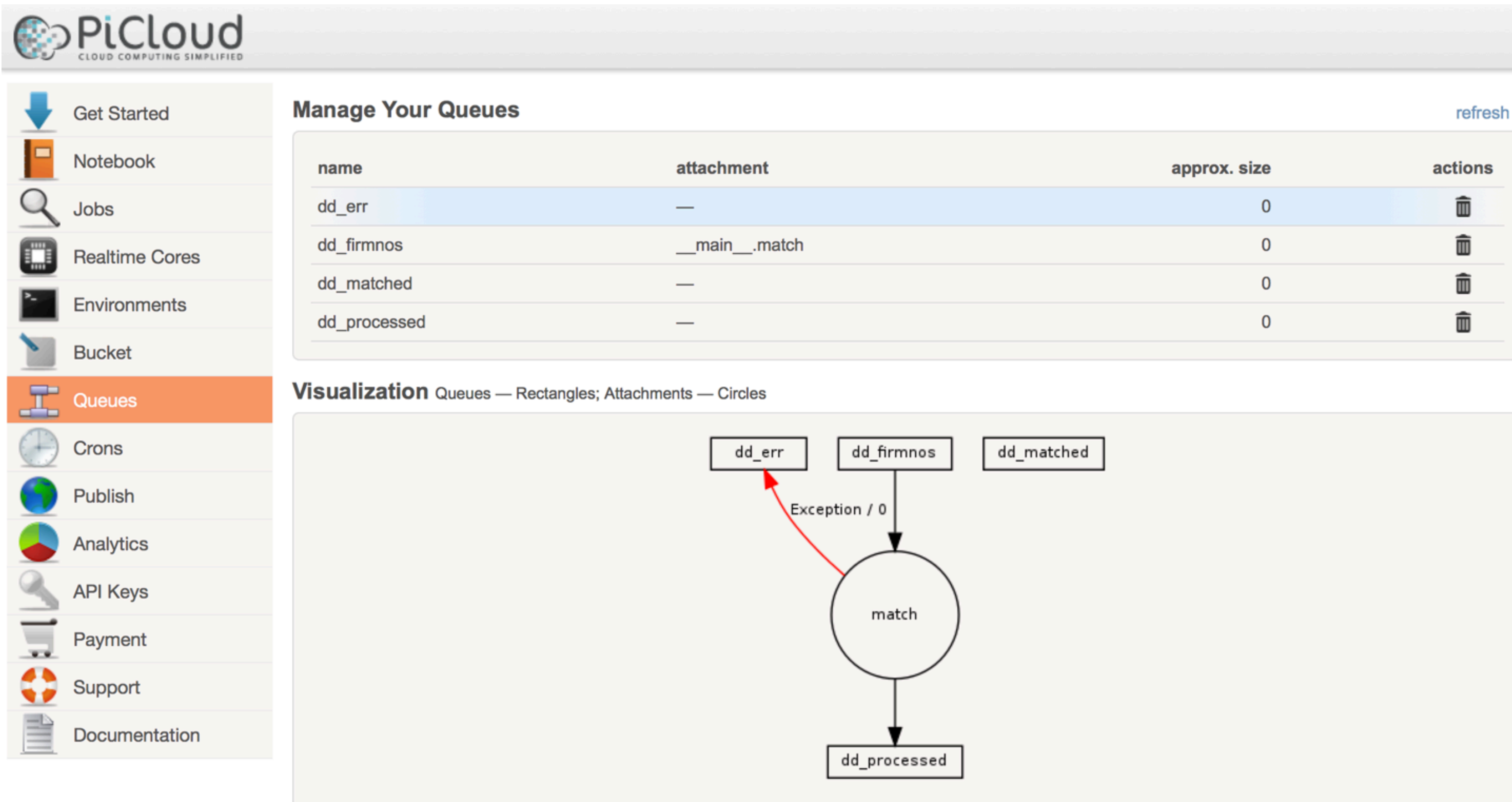
- Maybe you have a simple job but you're in a hurry
- Chop the problem into 1,000 batches and run 1,000 jobs

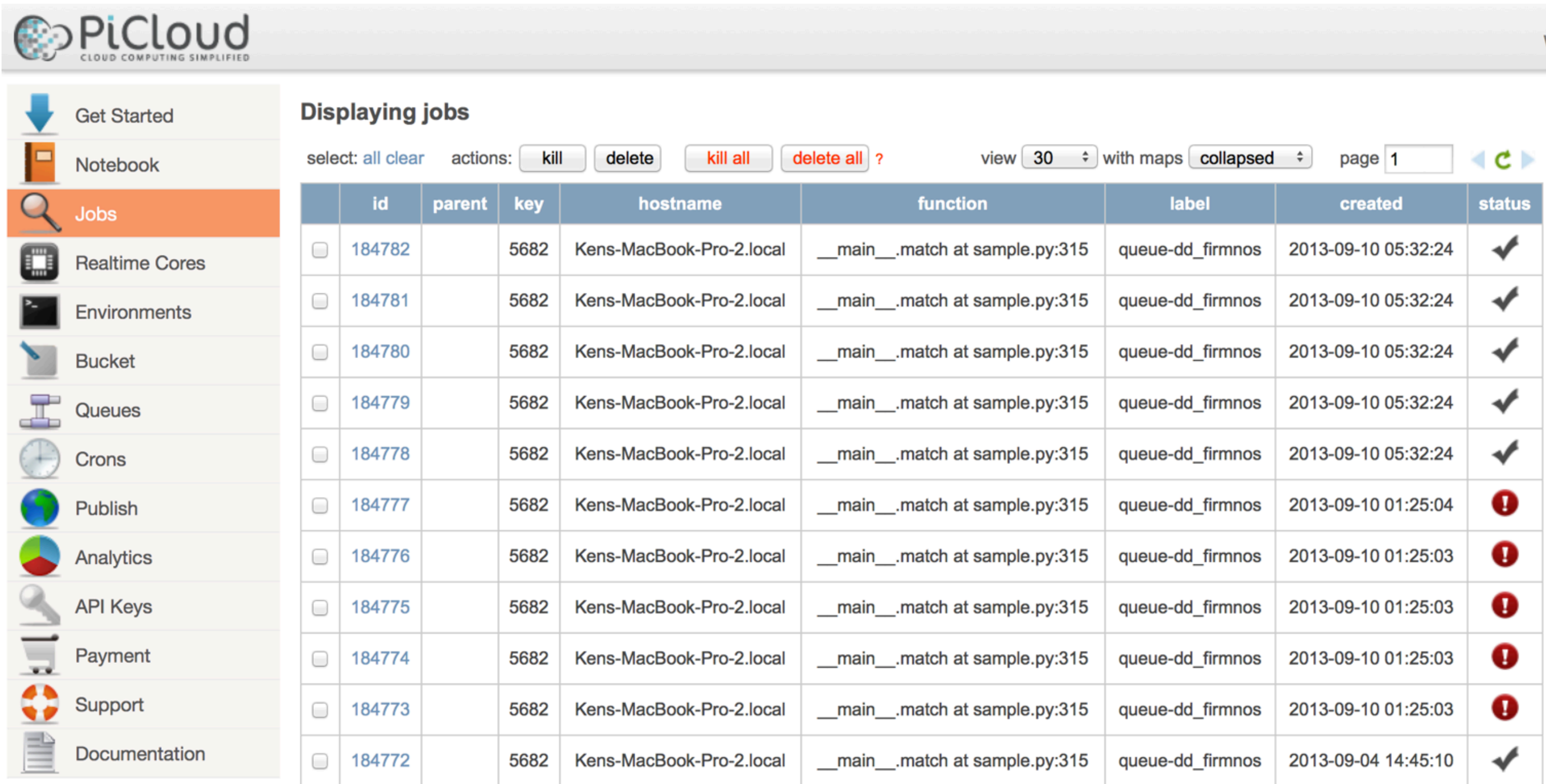
```
import cloud
import Levenshtein

def calculate(batch_no):

    # insert code here that you want to run on the cloud

    # limit execution to the "batch" of operations represented
    # by the batch_no parameter passed to the funtion

    # you can save results to cloud storage for download
    # or return the result(s) directly from the function

    return "Results from batch no " + str(batch_no)

batch_nos = [i for i in range(1000)]
job_ids = cloud.map(calculate, batch_nos, _type='c1', _env='younge')
```

Example Code: Scrape website from unique IP addresses

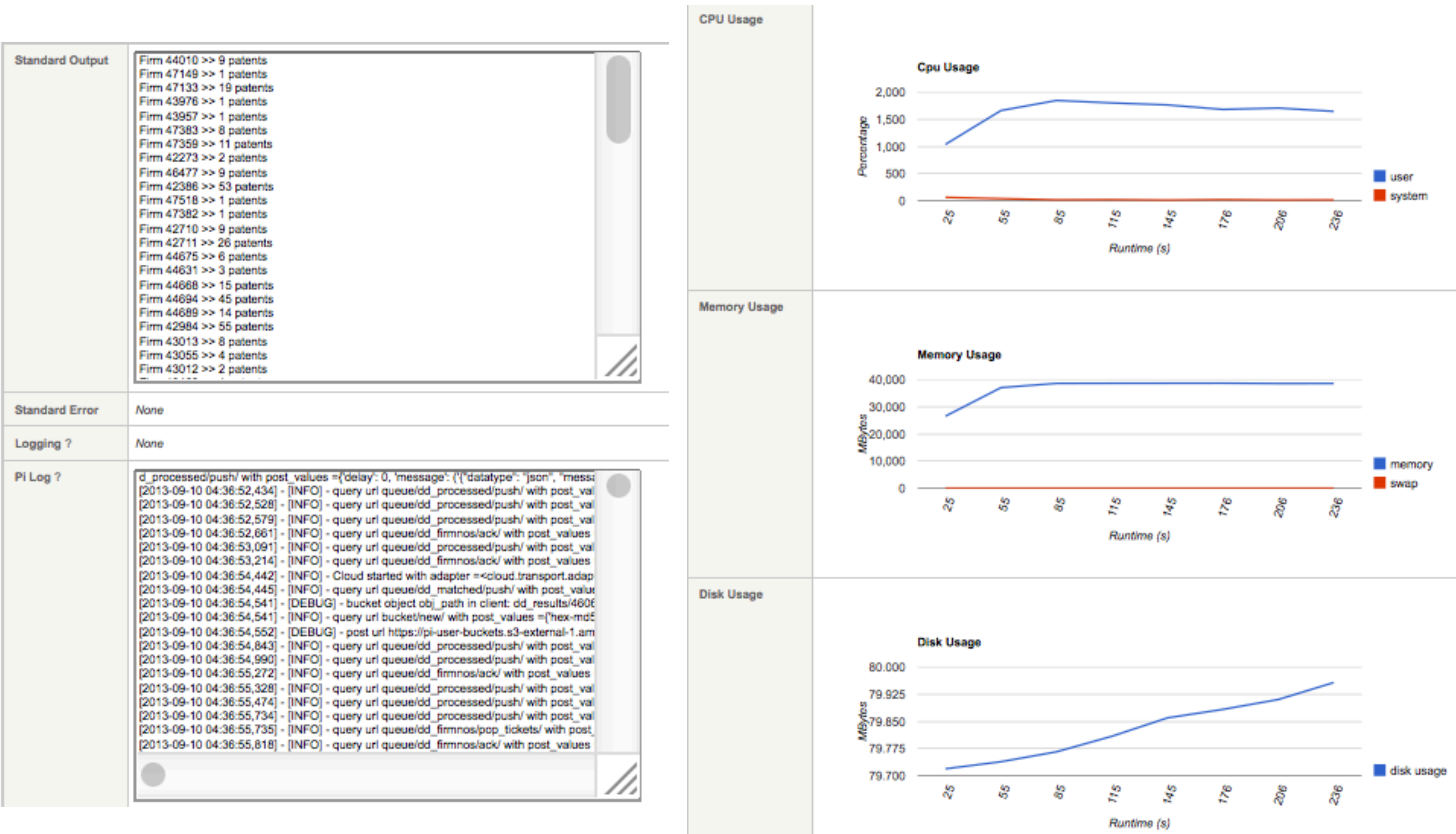
```
import cloud
import urllib2
import myfuncs

def scrape(patno):

    # scrape
    url = "http://patft.uspto.gov/netacgi/nph-Parser?&s1=" + str(patno) + ".PN."
    cnn = urllib2.urlopen(url)
    content = cnn.read()
    cnn.close()

    # save
    fname = str(patno) + ".html"
    cloud.bucket.putf(content, fname)
    cloud.bucket.make_public(fname)

    # move patno to the 'DONE' queue
    return patno

def main():

    # initialize queues
    q_todo = cloud.queue.get('ToDo')
    q_done = cloud.queue.get('Done')
    q_err  = cloud.queue.get('Error')

    # push list of patent numbers onto the starting queue
    patnos = load_list("patnos.txt")
    q_todo.push(patnos)

    # start execution
    q_todo.attach(scrape, q_done,
                  on_error={Exception: {'queue': q_err, 'delay': 0}},
                  retry_on=[urllib2.HTTPError, urllib2.URLError],
                  retry_delay=10, max_retries=3, max_parallel_jobs=1000,
                  readers_per_job=7, _type="s1")

if __name__ == "__main__": exit(main())
```

Example: Control Panel showing the queuing system





	id	parent	key	hostname	function	label	created	status
	184782		5682	Kens-MacBook-Pro-2.local	__main__.match at sample.py:315	queue-dd_firmnos	2013-09-10 05:32:24	✓
	184781		5682	Kens-MacBook-Pro-2.local	__main__.match at sample.py:315	queue-dd_firmnos	2013-09-10 05:32:24	✓
	184780		5682	Kens-MacBook-Pro-2.local	__main__.match at sample.py:315	queue-dd_firmnos	2013-09-10 05:32:24	✓
	184779		5682	Kens-MacBook-Pro-2.local	__main__.match at sample.py:315	queue-dd_firmnos	2013-09-10 05:32:24	✓
	184778		5682	Kens-MacBook-Pro-2.local	__main__.match at sample.py:315	queue-dd_firmnos	2013-09-10 05:32:24	✓
	184777		5682	Kens-MacBook-Pro-2.local	__main__.match at sample.py:315	queue-dd_firmnos	2013-09-10 01:25:04	!
	184776		5682	Kens-MacBook-Pro-2.local	__main__.match at sample.py:315	queue-dd_firmnos	2013-09-10 01:25:03	!
	184775		5682	Kens-MacBook-Pro-2.local	__main__.match at sample.py:315	queue-dd_firmnos	2013-09-10 01:25:03	!
	184774		5682	Kens-MacBook-Pro-2.local	__main__.match at sample.py:315	queue-dd_firmnos	2013-09-10 01:25:03	!
	184773		5682	Kens-MacBook-Pro-2.local	__main__.match at sample.py:315	queue-dd_firmnos	2013-09-10 01:25:03	!
	184772		5682	Kens-MacBook-Pro-2.local	__main__.match at sample.py:315	queue-dd_firmnos	2013-09-04 14:45:10	✓


Standard Output
Firm 44010 >> 9 patents
Firm 47149 >> 1 patents
Firm 47133 >> 19 patents
Firm 43976 >> 1 patents
Firm 43957 >> 1 patents
Firm 47383 >> 8 patents
Firm 47359 >> 11 patents
Firm 42273 >> 2 patents
Firm 46477 >> 9 patents
Firm 42386 >> 53 patents
Firm 47518 >> 1 patents
Firm 47382 >> 1 patents
Firm 42710 >> 9 patents
Firm 42711 >> 26 patents
Firm 44675 >> 6 patents
Firm 44631 >> 3 patents
Firm 44668 >> 15 patents
Firm 44694 >> 45 patents
Firm 44689 >> 14 patents
Firm 42984 >> 55 patents
Firm 43013 >> 8 patents
Firm 43055 >> 4 patents
Firm 43012 >> 2 patents
Standard Error
None
Logging ?
None
Pi Log ?
[2013-09-10 04:36:52,434] - [INFO] - query url queue/dd_processed/push/ with post_val
[2013-09-10 04:36:52,528] - [INFO] - query url queue/dd_processed/push/ with post_val
[2013-09-10 04:36:52,579] - [INFO] - query url queue/dd_processed/push/ with post_val
[2013-09-10 04:36:52,661] - [INFO] - query url queue/dd_firmnos/ack/ with post_values
[2013-09-10 04:36:53,091] - [INFO] - query url queue/dd_processed/push/ with post_val
[2013-09-10 04:36:53,214] - [INFO] - query url queue/dd_firmnos/ack/ with post_values
[2013-09-10 04:36:54,442] - [INFO] - Cloud started with adapter =<cloud.transport.adap
[2013-09-10 04:36:54,445] - [INFO] - query url queue/dd_matched/push/ with post_value
[2013-09-10 04:36:54,541] - [DEBUG] - bucket object obj_path in client: dd_results/4606
[2013-09-10 04:36:54,541] - [INFO] - query url bucket/new/ with post_values ={'hex-md5
[2013-09-10 04:36:54,552] - [DEBUG] - post url https://pi-user-buckets.s3-external-1.am
[2013-09-10 04:36:54,843] - [INFO] - query url queue/dd_processed/push/ with post_val
[2013-09-10 04:36:54,990] - [INFO] - query url queue/dd_processed/push/ with post_val
[2013-09-10 04:36:55,272] - [INFO] - query url queue/dd_firmnos/ack/ with post_values
[2013-09-10 04:36:55,328] - [INFO] - query url queue/dd_processed/push/ with post_val
[2013-09-10 04:36:55,474] - [INFO] - query url queue/dd_processed/push/ with post_val
[2013-09-10 04:36:55,734] - [INFO] - query url queue/dd_processed/push/ with post_val
[2013-09-10 04:36:55,735] - [INFO] - query url queue/dd_firmnos/pop_tickets/ with post
[2013-09-10 04:36:55,818] - [INFO] - query url queue/dd_firmnos/ack/ with post_values
CPU Usage
Cpu Usage
Percentage
2,000
1,500
1,000
500
0
25
55
85
115
145
176
206
236
Runtime (s)
user
system
Memory Usage
Memory Usage
MBytes
40,000
30,000
20,000
10,000
0
Runtime (s)
memory
swap
Disk Usage
Disk Usage
MBytes
80.000
79.925
79.850
79.775
79.700
Runtime (s)
disk usage

Summary

- Cloud computing can be easy and cheap
- I'm happy to chat more down at the pub!

Thank You

Machine Types

Core Type	Compute Units [1]	Memory	Disk	Max Multicore [2]	Price/Hour
c1 (default)	1	300 MB	15 GB	1	$0.05
c2	2.5	800 MB	30 GB	8	$0.13
f2	5.5 w/ HT	3.7 GB	100 GB	16	$0.22
m1	3.25	8 GB	140 GB	8	$0.30
s1 [3]	0.5 to 2	300 MB	4 GB	1	$0.04