Mor Consulting

Applied Parallel Computing with Python – Map/Reduce

PyCon 2013

Ian@MorConsulting.com @IanOzsvald - PyCon 2013

Goal

- Introduce map/reduce using Disco
- Count words and filter for interesting things
- Counting social interactions
- Practical configuration

Overview (pre-requisites)

- Disco (and erlang)
- Matplotlib (for visualisations)
- Cython+PIL/pillow+scikit-learn (for visualisations)
- NetworkX (for the social network visualisation)

Disco

- Disco – Python + Erlang
- http://discoproject.org/
- Install needs some experience
- Small but friendly community
- DDFS Filesystem
- Web management view
- Assumes node failures will occur

Disco

- Assumes no communication between nodes
- Can chain multiple map/reduce processes
- Very good for line processing on big, growing data sets

What is map/partition/reduce?

Mor Consulting

- Take paper
- Count frequency of “the”, “of”, “oochy”
- Partition using a hash function, send data
- Reduce partial counts to one count
- Return the complete results to master
- What if a node had died?

Web interface

- Let's run disco:
- `DISCO_HOME $ bin/disco nodaemon`
- Check it in the browser:
- http://localhost:8989/
- 1 node with 1 worker
- Be *aware* of localhost vs hostnames

Our data

- Count words in 357 tweets on 1 machine
- `./2_MapReduceDisco/tweet_data/`
- Lines of JSON-encoded data
- Approximately 12 words per line * 357 lines
- 4,500 words to count, lots are repeated

Running tiny example

- Count words in 357 tweets on 1 machine
- `./2_MapReduceDisco`
- This is a *generator* function
- Returns `(key,value)` pair
- Do->Split the line into words
- Do->Yield a count of 1 per word
- `$ python count_tweet_words.py`
- `->mapreduceout_wordcount.json`

What's going on?

- Check localhost:8989 → job (right)
- Takes a list of files (we have 1)
- Utilises local filesystem
- `import count_tweet_words #why?`

What's the output?

- `mapreduceout_wordcount.json`
- 1968 lines of counted words
- `$ python word_count_cloud/plot_from_mapreduce.py mapreduceout_wordcount.json`

Running larger example

- Count words in 859,157 tweets on 1 machine
- 12*859157==10,309,884 rows to count
- Same code, different input
- `$ python count_tweet_words.py`
- `->mapreduceout_wordcount.json`
- Check localhost:8989 → job (right)
- Maybe you run out of RAM? 1.9GB...

Use a combiner

- `from disco.func import sum_combiner`
- `Job()(..,combiner=sum_combiner)`
- Run it again – 100MBs only
- It does the counting after mapping

Using DDFS

- `./tweet_data`
- `$ split -l 100000 tweets_859157.json # xaa..xai`
- Run it again – 100MBs only
- `$ ddfs chunk data:tweets859157xa ./xa?`
- We've created 9 input files
- Lives in `DISCO_HOME/root/ddfs`

Run with DDFS

- `from disco.func import chain_reader, sum_combiner`
- `input = ["tag://data:tweets859157xa"]`
- `job=...map_reader=chain_reader,`
- Run it again – takes 1 minute
- Configure 4 workers in web interface
- Now it takes about 30 seconds

Reduction?

- Reduction occurs on each machine, hashed to a machine (data shuffled, can move) – counts for keyX → same machine
- This shuffling means that reduction occurs evenly over machines
- Sort pairs
- Reduce same keys to 1 value
- Combine results back on master

Now visualise again

- `$ python word_count_cloud/plot_from_mapreduce.py mapreduceout_wordcount.json`
- What about word frequencies – Zipf distribution?
- `$ python check_word_frequencies.py`

Your task

- You need to filter for “samsung” tweets (or “olympics” or “london”)
- `“filter_word in tweet.lower()”`
- `“yield "", 0” # means ignore me`
- How does the visualisation change?

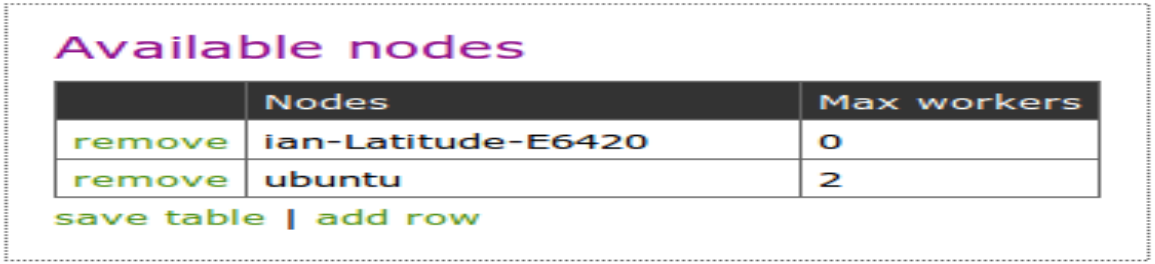
Now we'll count interactions

- Run my example:
- `count_tweet_words_6.py`
- `$ python draw_interactions_graph.py`
- Who is talked at a lot?

Multi-machine configuration

- /etc/hosts
  - 127.0.0.1 localhost
  - 127.0.1.1 ian-Latitude-E6420
  - 192.168.0.32 ubuntu

system configuration



Feedback

- Write-up: http://ianozsvald.com
- I want feedback (and a testimonial please)
- “High Performance Python” book/site?
- ian@ianozsvald.com
- Thank you :-)