Performing *embarrassingly* parallel data analysis in Python using **ipyparallel** and **pandas**

Marcos Oliveira

Complexity72h @ IMT School for Advanced Studies Lucca

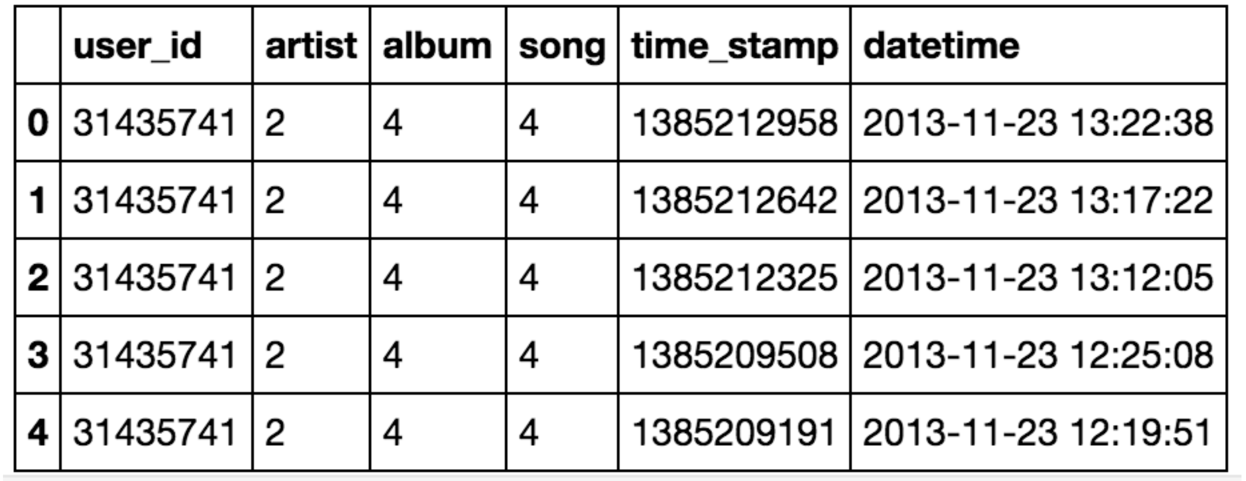
Why?

- An embarrassingly parallel problem

  A problem that may be solved in parallel by easily splitting it into separate problems.

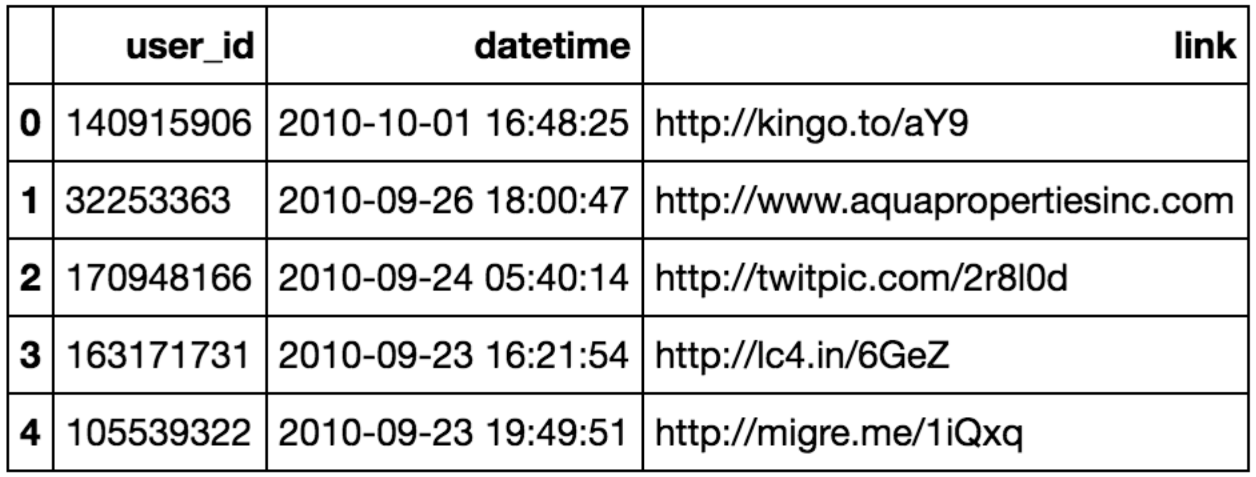
- In data analysis, quite often:
  - Same procedure for different parts of the data:

Example: Last.fm data set



	user_id	artist	album	song	time_stamp	datetime
0	31435741	2	4	4	1385212958	2013-11-23 13:22:38
1	31435741	2	4	4	1385212642	2013-11-23 13:17:22
2	31435741	2	4	4	1385212325	2013-11-23 13:12:05
3	31435741	2	4	4	1385209508	2013-11-23 12:25:08
4	31435741	2	4	4	1385209191	2013-11-23 12:19:51

Example: a Twitter data set

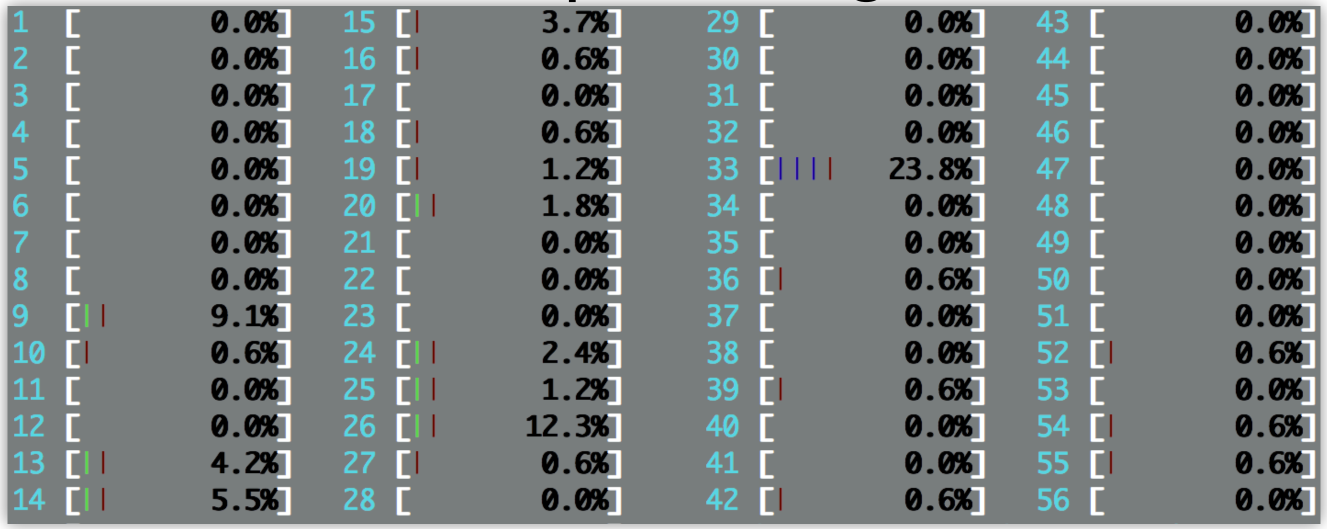


	user_id	datetime	link
0	140915906	2010-10-01 16:48:25	http://kingo.to/aY9
1	32253363	2010-09-26 18:00:47	http://www.aquapropertiesinc.com
2	170948166	2010-09-24 05:40:14	http://twitpic.com/2r8l0d
3	163171731	2010-09-23 16:21:54	http://lc4.in/6GeZ
4	105539322	2010-09-23 19:49:51	http://migre.me/1iQxq

Why?

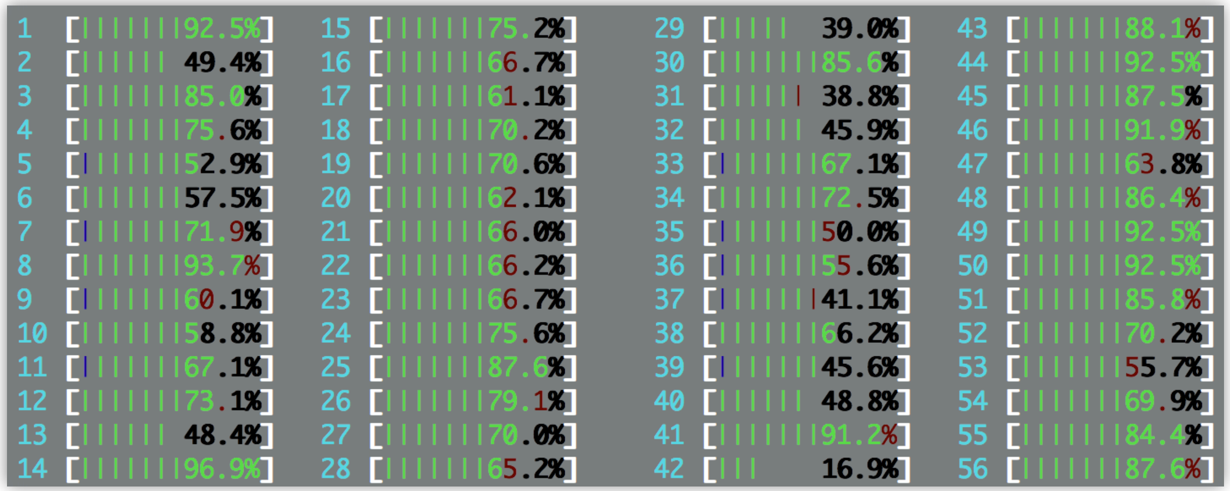
- Embarrassingly parallel data analysis

Available processing units



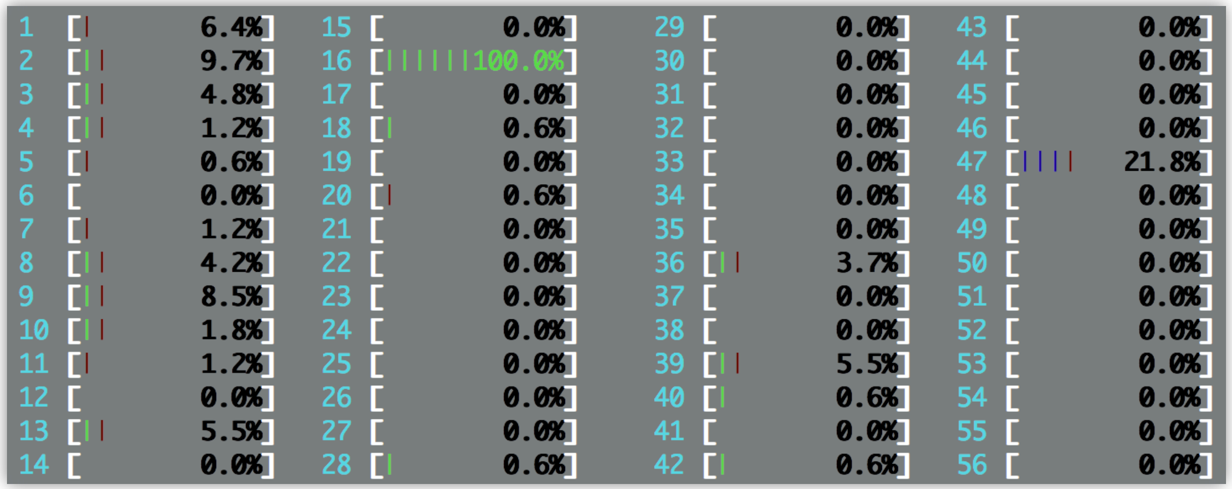
```
1  [        0.0%]  15 [|       3.7%]  29 [        0.0%]  43 [        0.0%]
2  [        0.0%]  16 [|       0.6%]  30 [        0.0%]  44 [        0.0%]
3  [        0.0%]  17 [        0.0%]  31 [        0.0%]  45 [        0.0%]
4  [        0.0%]  18 [|       0.6%]  32 [        0.0%]  46 [        0.0%]
5  [        0.0%]  19 [|       1.2%]  33 [||||   23.8%]  47 [        0.0%]
6  [        0.0%]  20 [||      1.8%]  34 [        0.0%]  48 [        0.0%]
7  [        0.0%]  21 [        0.0%]  35 [        0.0%]  49 [        0.0%]
8  [        0.0%]  22 [        0.0%]  36 [|       0.6%]  50 [        0.0%]
9  [||      9.1%]  23 [        0.0%]  37 [        0.0%]  51 [        0.0%]
10 [|       0.6%]  24 [||      2.4%]  38 [        0.0%]  52 [|       0.6%]
11 [        0.0%]  25 [||      1.2%]  39 [|       0.6%]  53 [        0.0%]
12 [        0.0%]  26 [||     12.3%]  40 [        0.0%]  54 [|       0.6%]
13 [||      4.2%]  27 [|       0.6%]  41 [        0.0%]  55 [|       0.6%]
14 [||      5.5%]  28 [        0.0%]  42 [|       0.6%]  56 [        0.0%]
```

<u>Usage</u> - what we want:



```
1  [|||||||92.5%]  15 [|||||||75.2%]  29 [|||||  39.0%]  43 [|||||||88.1%]
2  [|||||| 49.4%]  16 [|||||||66.7%]  30 [|||||||85.6%]  44 [|||||||92.5%]
3  [|||||||85.0%]  17 [|||||||61.1%]  31 [|||||| 38.8%]  45 [|||||||87.5%]
4  [|||||||75.6%]  18 [|||||||70.2%]  32 [|||||| 45.9%]  46 [|||||||91.9%]
5  [|||||||52.9%]  19 [|||||||70.6%]  33 [|||||||67.1%]  47 [|||||||63.8%]
6  [|||||||57.5%]  20 [|||||||62.1%]  34 [|||||||72.5%]  48 [|||||||86.4%]
7  [|||||||71.9%]  21 [|||||||66.0%]  35 [|||||||50.0%]  49 [|||||||92.5%]
8  [|||||||93.7%]  22 [|||||||66.2%]  36 [|||||||55.6%]  50 [|||||||92.5%]
9  [|||||||60.1%]  23 [|||||||66.7%]  37 [|||||||41.1%]  51 [|||||||85.8%]
10 [|||||||58.8%]  24 [|||||||75.6%]  38 [|||||||66.2%]  52 [|||||||70.2%]
11 [|||||||67.1%]  25 [|||||||87.6%]  39 [|||||||45.6%]  53 [|||||||55.7%]
12 [|||||||73.1%]  26 [|||||||79.1%]  40 [|||||| 48.8%]  54 [|||||||69.9%]
13 [|||||| 48.4%]  27 [|||||||70.0%]  41 [|||||||91.2%]  55 [|||||||84.4%]
14 [|||||||96.9%]  28 [|||||||65.2%]  42 [|||    16.9%]  56 [|||||||87.6%]
```

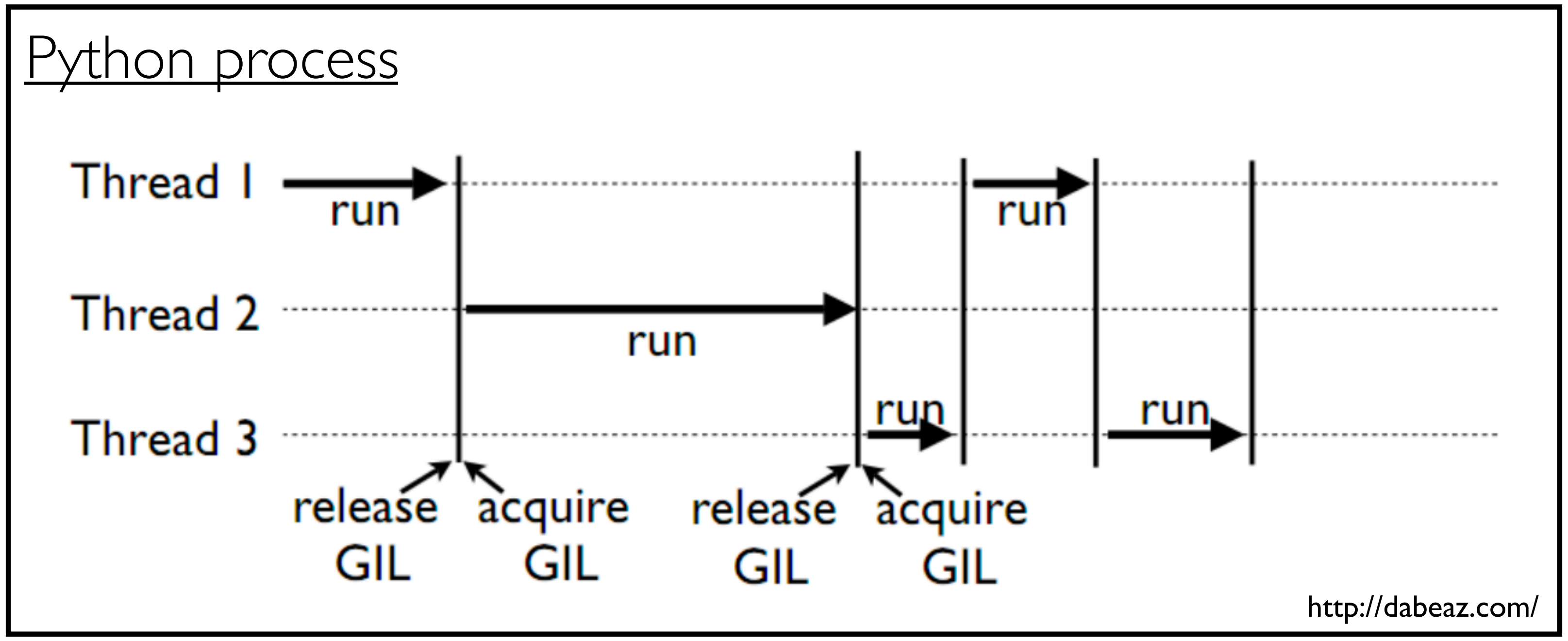
<u>Usage</u> - what we sometimes have:



```
1  [|       6.4%]  15 [        0.0%]  29 [        0.0%]  43 [        0.0%]
2  [||      9.7%]  16 [||||||100.0%]  30 [        0.0%]  44 [        0.0%]
3  [||      4.8%]  17 [        0.0%]  31 [        0.0%]  45 [        0.0%]
4  [||      1.2%]  18 [|       0.6%]  32 [        0.0%]  46 [        0.0%]
5  [|       0.6%]  19 [        0.0%]  33 [        0.0%]  47 [||||   21.8%]
6  [        0.0%]  20 [|       0.6%]  34 [        0.0%]  48 [        0.0%]
7  [|       1.2%]  21 [        0.0%]  35 [        0.0%]  49 [        0.0%]
8  [||      4.2%]  22 [        0.0%]  36 [||      3.7%]  50 [        0.0%]
9  [||      8.5%]  23 [        0.0%]  37 [        0.0%]  51 [        0.0%]
10 [||      1.8%]  24 [        0.0%]  38 [        0.0%]  52 [        0.0%]
11 [|       1.2%]  25 [        0.0%]  39 [||      5.5%]  53 [        0.0%]
12 [        0.0%]  26 [        0.0%]  40 [|       0.6%]  54 [        0.0%]
13 [||      5.5%]  27 [        0.0%]  41 [        0.0%]  55 [        0.0%]
14 [        0.0%]  28 [|       0.6%]  42 [|       0.6%]  56 [        0.0%]
```

Python

- Python allows multi-thread *programming*… `import thread`
- but GIL — the global interpreter lock



- GIL prevents multi-thread *execution*.

- "*Solution*": instead of multi-threads, multi-processes.
  - Reminder: CPU bounded vs. I/O bounded

The good the bad the ugly

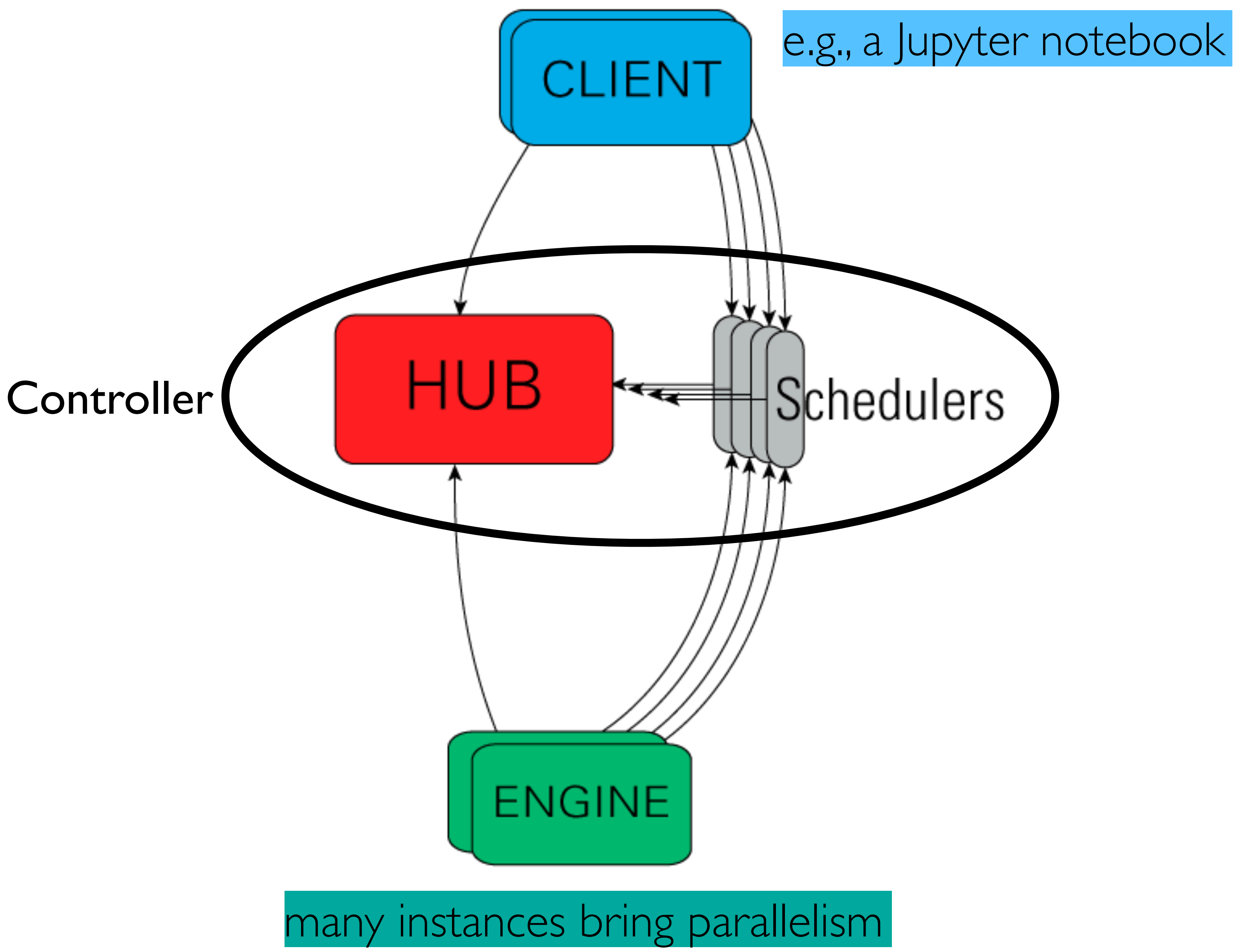
- Multiprocessing native in Python

```
from multiprocessing import Pool
from multiprocessing import Process
```

- IPyparallel
  - Jupyter notebook ~ quick-dirty coding, preliminary analyses
  - With pandas, we have parallel quick data analysis in few code lines.
  - Interactive

IPyparallel

- In a nutshell:



IPyparallel

- Installing via pip:

```
pip install ipyparallel
```

- Creating controller + 4 engines:

```
ipcluster start -n 4
```

- By the way, ipcluster is a shortcut that creates both controller and engine in the localhost.
    - You can have more control of configuration using profiles:

      https://ipyparallel.readthedocs.io/en/latest/process.html

IPyparallel

- Coding time