An AWS Experiment

- On a new `t2.micro`:
- What Size Data Set Will Cause a Memory Exception?
- What Size Dataset Is Too Large to Be Used to Fit Different Kinds of Simple Models?

An AWS Experiment

What Size Data Set Will Cause a Memory Exception?

- Run the `jupyter/scipy-notebook` image
- Monitor memory usage using `docker stats`.
- Use the `sklearn.datasets.make_classification` function to create datasets of arbitrary sizes
- Restart the Python kernel after each test.
- Record resource consumption for each dataset and take note of any memory exceptions.

Create a New `t2.micro`

- Computing resources:
  - 1 CPU, 1 GB Ram, 8GB of Hard Drive space
  - Make sure that ports 22, 2376, and 8888 are open.
- Install `docker`

  ```
  $ curl -sSL https://get.docker.com/ | sh
  $ sudo usermod -aG docker ubuntu
  ```

  Then disconnect and reconnect.

```
sudo curl -L https://github.com/docker/compose/releases/download/1.15.0/docker-compose-
`uname -s`-`uname -m` > docker-compose
sudo mv docker-compose /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

`docker stats`

- Display a live stream of container(s) resource usage statistics.

- Run using
  `$ docker stats`

- to stop
  `ctrl-c`

CONTAINER	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
72038ae46932	0.02%	157.1MiB / 7.787GiB	1.97%	1.01MB / 1.49MB	114MB / 0B	20

tmux

- We will use `tmux` to design an environment for our tests
- We will use two windows:
  - One for managing our Docker processes
  - One for monitoring system usage
- This tmux cheatsheet may be helpful.

ITerm

ssh connected to AWS - worker

tmux session named `worker`

tmux window named `controller`

tmux window named `docker stats`

ssh connected to AWS - mongo

tmux session named `mongo`

tmux window named `controller`

tmux window named `docker stats`

tmux session named `new_session`

tmux window named `0`

tmux window named `1`

tmux


/Users/joshuacook
joshuacook@LOCAL:~
$
Green bar signifies that tmux is running.
tmux windows
[0] 0:bash- 1:bash*
"Joshuas-MacBook-Pro.l" 13:20 27-Aug-17

tmux

- Important commands:
  - Use `ctrl-b` to issue commands to tmux
  - *create a new window*:
    - `ctrl-b`, then `c`¶
  - *previous window*:
    - `ctrl-b`, then `p`
  - *next window*:
    - `ctrl-b`, then `n`

Run the jupyter/scipy-notebook image and `docker stats`

- Run `tmux`.
- Pull the `jupyter/scipy-notebook` image
- While the pull is running, create a new `tmux` window.
- Start a `docker stats` monitor in this window
- Check the pull status in the previous window
- When the pull completes, run jupyter and obtain the token

Record Baseline System Usage

- Examine the `docker stats` window with jupyter running and a single notebook open with no code executed.

- In a text document on your computer, record this as

  `{'name':'baseline','cpu_perc':#val1#,'mem_usage': #val2#, 'rows': 0, 'columns': 0},`

- Note the comma at the end of the line.

Load a Dataset

- In this testing notebook:

- Restart the kernel

- use `make_classification` to load a dataset of 100 rows and 20 columns.

```
from sklearn.datasets import make_classification
X, y = make_classification(100, 20)
```

- Check the resource usage in `docker stats` and update the text file.

```
{'name':'default','cpu_perc':#val1#,'mem_usage':#val2#,
'rows': 100, 'columns': 20},
```

Load a Dataset

- Repeat (restart the kernel each time) with
  - 10x - dataset of 1000 rows and 20 columns.
  - 100x - dataset of 1000 rows and 200 columns.
  - 1000x - dataset of 10000 rows and 200 columns.
  - 10000x - dataset of 10000 rows and 2000 columns.
  - ...
- Until you hit a memory exception

An AWS Experiment

What Size Dataset Is Too Large to Be Used to Fit Different Kinds of Simple Models?

- Run the `jupyter/scipy-notebook` image
- Monitor memory usage using `docker stats`.
- Use the `sklearn.datasets.make_classification` function to create datasets of arbitrary sizes. Use `sklearn.svm.SVC` to fit a classification model.
- Restart the Python kernel after each test.
- Record resource consumption for each dataset and take note of any memory exceptions.

Record Baseline System Usage

- Examine the `docker stats` window with jupyter running and a single notebook open with no code executed.

- In a text document on your computer, record this as

  ```
  {'name':'baseline','cpu_perc':#val1#,'mem_usage':
  #val2#, 'rows': 0, 'columns': 0, 'model': 'SVC'},
  ```

- Note the comma at the end of the line.

Load a Dataset and Fit a Model

- In the testing notebook, restart the kernel, then

- use `make_classification` to load a dataset of 100 rows and 20 columns.

```
from sklearn.datasets import make_classification
X, y = make_classification(100, 20)
```

- use `sklearn.SVM.SVC` to fit a classification model

```
from sklearn.svm import SVC
this_svc = SVC()
this_svc.fit(X, y)
this_svc.score(X, y)
```

- Check the resource usage in `docker stats` and update the text file.

```
{'name':'default','cpu_perc':#val1#,'mem_usage':#val2#,
'rows': 100, 'columns': 20},
```

- Make sure to monitor **peak cpu percentage.**

Load a Dataset and Fit a Model

- Repeat (restart the kernel each time) with
    - 10x - dataset of 1000 rows and 20 columns.
    - 100x - dataset of 1000 rows and 200 columns.
    - 1000x - dataset of 10000 rows and 200 columns.
    - 10000x - dataset of 100000 rows and 200 columns.
    - ...
- Until you hit a memory exception