Python multiprocessing

Практикум на ЭВМ

Полыковский Даниил

ВМК МГУ

26 октября 2015 г.

Мотивация

Процессор не всегда загружен на 100%

- Использование только одного ядра
- Ожидание доступа к ресурсам (нпр. памяти)
- Неравномерная нагрузка запросов

Параллелизм

- Незавимые части программы (master/worker)
- Последовательность выполнения не важна

Построение RandomForrest[1]: **14 минут 50 секунд**

[1] 100 деревьев, 100000 объектов, 29095 признаков

Два вида параллелизма

- Процессы

```
1  from multiprocessing import Process
2  p = Process(target=worker_func, args=(...))
3  p.start()
4  p.join()
```

- Threads

```
1  from multiprocessing.dummy import Pool
2  pool = Pool(number_of_workers)
3  res = pool.map(func, ar) #in parallel
```

Разделение ресурсов

- Блокировка

```
1  from multiprocessing import Lock
2  lock = Lock()
3  lock.acquire()
4  lock.release()
```

- Потокобезопасные структуры

```
1  from multiprocessing import Manager
2  manager = Manager()
3  shared_list = manager.list()
4  shared_dict = manager.dict()
```

Многопоточный RandomForest

```
1  def train_in_parallel(model, tree_args, X, y):
2      n = tree_args['n_estimators']
3      pool = Pool()
4      m = model(**tree_args)
5      tree_args['n_estimators'] = 1
6      m.estimators_ = pool.map(
7          lambda x: model(**tree_args).fit(X, y)[0],
8          xrange(n_estimators))
9      return m
```

Многопоточный RandomForest

```
1  def train_in_parallel(model, tree_args, X, y):
2      n = tree_args['n_estimators']
3      pool = Pool()
4      m = model(**tree_args)
5      tree_args['n_estimators'] = 1
6      m.estimators_ = pool.map(
7          lambda x: model(**tree_args).fit(X, y)[0],
8          xrange(n_estimators))
9      return m
```

Строим RandomForrest: **8 минут 4 секунды**. Выигрыш почти в 2 раза!

Литература

- Python documentation
  docs.python.org/2/library/multiprocessing.html
- Параллелизм в одну строчку
  toly.github.io/blog/2014/02/13/parallelism-in-one-line