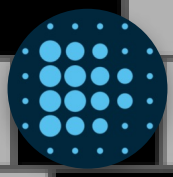
A Peek into GNU Parallel

4th Aug, 2015



BY DATAWEAVE

GNU Parallel is a tool that can be deployed from a shell to parallelize job execution. A job can be anything from simple shell scripts to complex interdependent Python/Ruby/Perl scripts. The simplicity of 'Parallel' tool lies in it usage. A modern day computer with multicore processors should be enough to run your jobs in parallel. A single core computer can also run the tool, but the user won't be able to see any difference as the jobs will be context switched by the underlying OS.

At **DataWeave**, we use Parallel for automating and parallelizing a number of resource extensive processes ranging from crawling to data extraction. All our servers have 8 cores with capability of executing 4 threads in each. So, we experienced huge performance gain after deploying Parallel. Our in-house image processing algorithms used to take more than a day to process 200,000 high resolution images. After using Parallel, we have brought the time down to a little over 40 minutes!

GNU Parallel can be installed on any Linux box and does not require sudo access. The following command will install the tool:

```
(wget -O - pi.dk/3 || curl pi.dk/3/) | bash
```

GNU Parallel can read inputs from a number of sources—a file or command line or stdin. The following simple example takes the input from the command

line and executes in parallel:

```
parallel echo ::: A B C
```

The following takes the input from a file:

```
parallel -a somefile.txt echo

Or STDIN:


cat somefile.txt | parallel echo
```

The inputs can be from multiple files too:

```
parallel -a somefile.txt -a anotherfile.txt echo
```

The number of simultaneous jobs can be controlled using the—jobs or -j switch. The following command will run 5 jobs at once:

```
parallel --jobs 5 echo ::: A B C D E F G H I J
```

By default, the number of jobs will be equal to the number of CPU cores. However, this can be overridden using percentages. The following will run 2 jobs per CPU core:

```
parallel --jobs 200% echo ::: A B C D
```

If you do not want to set any limit, then the following will use all the available CPU cores in the machine. However, this is NOT recommended in production environment as other jobs running on the machine will be vastly slowed down.

```
parallel --jobs 0 echo ::: A B C
```

Enough with the toy examples. The following will show you how to bulk insert JSON documents in parallel in a MongoDB cluster. Almost always we need to insert millions of document quickly in our MongoDB cluster and inserting documents serially doesn't cut it. Moreover, MongoDB can handle parallel inserts.

The following is a snippet of a file with JSON document. Let's assume that there are a million similar records in the file with one JSON document per line.

```
{“name”: “John”, “city”: “Boston”, “age”: 23} {“name”: “Alice”, “city”: “Seattle”, “ag
e”: 31} {“name”: “Patrick”, “city”: “LA”, “age”: 27} ... ...
```

The following Python script will get each JSON document and insert into “people” collection under “dw” database.

```
import json

import pymongo

import sys

document = json.loads(sys.argv[1])

client = pymongo.MongoClient()

db = client[“dw”]

collection = db[“people”]

try:

    collection.insert(document)

except Exception as e:

    print “Could not insert document in db”, repr(e)
```

Now to run this parallely, the following command should do the magic:

```
cat people.json | parallel ‘python insertDB.py {}’
```

That’s it! There are many switches and options available for advanced processing. They can be accessed by doing a man parallel on the shell. Also the following page has a set of tutorials: **GNU Parallel Tutorials**.

Originally published at **blog.dataweave.in.**

- DataWeave Marketing

4th Aug, 2015

DATA ENGINEERING