> By Jiahui Tang

Spec:

- GPU: AWS GPU g3.4xlarge, Tesla M60 (More details attached)
- OS: Ubuntu 18.04
- Kernel: Linux ip-172-31-36-197 5.4.0-1039-aws #41~18.04.1-Ubuntu SMP Fri Feb 26 11:20:14 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
- EBS: Resize EBS Volume to 128GB
- Compiler: gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0 (no flags)

```
ubuntu@ip-172-31-36-197:~$ uname -a
Linux ip-172-31-36-197 5.4.0-1039-aws #41~18.04.1-Ubuntu SMP Fri Feb 26 11:20:14 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
```

```
ubuntu@ip-172-31-36-197:~$ lspci | grep -i nvidia
00:1e.0 VGA compatible controller: NVIDIA Corporation GM204GL [Tesla M60] (rev a1)
ubuntu@ip-172-31-36-197:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev             60G     0   60G   0% /dev
tmpfs            12G   844K  12G   1% /run
/dev/xvda1      7.7G  1.2G  6.6G  15% /
tmpfs            60G     0   60G   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs            60G     0   60G   0% /sys/fs/cgroup
/dev/loop0       32M    32M     0 100% /snap/snapd/11036
/dev/loop1       56M    56M     0 100% /snap/core18/1988
/dev/loop2       34M    34M     0 100% /snap/amazon-ssm-agent/3552
tmpfs            12G     0   12G   0% /run/user/1000
ubuntu@ip-172-31-36-197:~$
```

```
ubuntu@ip-172-31-36-197:~$ pgaccelinfo

CUDA Driver Version:           11020
NVRM version:                  NVIDIA UNIX x86_64 Kernel Module  460.32.03  Sun Dec 27 19:00:34 UTC 2020

Device Number:                 0
Device Name:                   Tesla M60
Device Revision Number:        5.2
Global Memory Size:            7988903936
Number of Multiprocessors:     16
Concurrent Copy and Execution: Yes
Total Constant Memory:         65536
Total Shared Memory per Block: 49152
Registers per Block:           65536
Warp Size:                     32
Maximum Threads per Block:     1024
Maximum Block Dimensions:      1024, 1024, 64
Maximum Grid Dimensions:       2147483647 x 65535 x 65535
Maximum Memory Pitch:          2147483647B
Texture Alignment:             512B
Clock Rate:                    873 MHz
Execution Timeout:             No
Integrated Device:             No
Can Map Host Memory:           Yes
Compute Mode:                  default
Concurrent Kernels:            Yes
ECC Enabled:                   Yes
Memory Clock Rate:             2505 MHz
Memory Bus Width:              256 bits
L2 Cache Size:                 2097152 bytes
Max Threads Per SMP:           2048
Async Engines:                 2
Unified Addressing:            Yes
Managed Memory:                Yes
Concurrent Managed Memory:     No
Default Target:                cc50
```

```
ubuntu@ip-172-31-36-197:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/7/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 7.5.0-3ubuntu1~18.04' --with-bugurl=fi
only --program-suffix=-7 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id
e=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-
ib --enable-objc-gc=auto --enable-multiarch --disable-werror --with-arch-32=i686 --with-abi=m64 --wit
cking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)
ubuntu@ip-172-31-36-197:~$
```

```
ubuntu@ip-172-31-36-197:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 79
model name      : Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz
stepping        : 1
microcode       : 0xb000038
cpu MHz         : 1794.171
cache size      : 46080 KB
physical id     : 0
siblings        : 16
core id         : 0
cpu cores       : 8
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge m
cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l
bogomips        : 4600.14
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

3. Accelerated Computing (35 points)

3.1. Acceleration of Basic Code (20 points)

The goal of this exercise is to familiarize you with OpenACC and make your first accelerated code. Develop an OpenACC version of the jacobi code used in section 2.1. Your task is to parallelize the `for` loops in the function `jacobi` using OpenACC. You should do this by copying over the corresponding code from `jacobi1d.c` and putting OpenACC `#pragma`s in appropriate places. You should develop two versions:

A. One with a copy of data in each iteration and
B. One with a single copy of data.

1. Run both codes with `ncells` equal to 10^k, for k = 6, 7, and 8, and 100 steps. Plot the execution times of both codes and the sequential execution (with no acceleration) as a function of k. Consider the elapsed time provided by the application and the GPU runtime provided by the driver.
2. Run the second, enhanced version of the code, with `ncells` of 10^8, 100 steps and multiple values (32, 64, 128, 256, 512, 1024) for the number of threads per block (vector length). Which value for the number of threads is faster and why?

Submission

- `P31.pdf`: Answer the questions and discuss the implementations
- `P31a.c`: First version source code
- `P31b.c`: Second version source code, with vector lengths specified

Dicuss the implementation:

```
for (sweep = 0; sweep < nsweeps; sweep += 2) {
#pragma acc data copy(u[0:n]) create(utmp[0:n]) copyin(f[0:n])

    /* Old data in u; new data in utmp */
    //#pragma acc parallel loop vector_length(32)
    #pragma acc kernels loop independent
    for (i = 1; i < n; ++i){
        utmp[i] = (u[i-1] + u[i+1] + h2*f[i])/2;

    }
    /* Old data in utmp; new data in u */
    // #pragma acc parallel loop vector_length(32)
    #pragma acc kernels loop independent
    for (i = 1; i < n; ++i){
        u[i] = (utmp[i-1] + utmp[i+1] + h2*f[i])/2;
    }
}

free(utmp);
```

```
    #pragma acc data copy(u[0:n]) create(utmp[0:n]) copyin(f[0:n])
    for (sweep = 0; sweep < nsweeps; sweep += 2) {

        /* Old data in u; new data in utmp */
        //#pragma acc parallel loop vector_length(32)
        #pragma acc kernels loop independent vector_length(32)
        for (i = 1; i < n; ++i){
            utmp[i] = (u[i-1] + u[i+1] + h2*f[i])/2;

        }
        /* Old data in utmp; new data in u */
        // #pragma acc parallel loop vector_length(32)
        #pragma acc kernels loop independent vector_length(32)
        for (i = 1; i < n; ++i){
            u[i] = (utmp[i-1] + utmp[i+1] + h2*f[i])/2;
        }
    }

    free(utmp);
}
```

For implementation,

1. `#pragma acc data copy(u[0:n]) create(utmp[0:n]) copyin(f[0:n])`

For `#pragma acc data` , in order to optimize jacobi() function and copying data for iterations, we put the data optimziation inside/outside the outer for loop for version A and B.

- matrix u: It is defined before the for loop, and will be changed inside the inner two for loops, and will also be an output after executing for loops, so we use copy clause for both copy in and export.
- utmp matrix: It is a temporary matrix that only storage intermediate value inside two for loops. By using create clause, we could pre-allocate space for him, but it's a temporary variable and we don't need output.
- f matrix: the f matrix information is used when running two inner for loops, so we need copyin to send its data into the parallel processing. But there's no output for f matrix needed, as its value is not changed. So we use only `copyin` instead of `copy`
- [0:N]: since we want to make sure that length of space is appropriately allocated for the elements in matrix, we explicitly specify the index and length.

1. `#pragma acc kernels loop independent`

I tried to distinguish the difference between two different clauses

```
* #pragma acc kernels loop
* #pragma acc parallel loop
```

They could all be compiled but the main difference is that kernels construct may be thought of as a hint to the compiler of where it should look for parallelism, while the parallel directive is an assertion to the compiler of where there is parallelism. Thus, I put it as `#pragma acc kernels loop` .

As for `independent`, by using this construct in conjunction with the loop construct, we can tell the compiler that loop iterations are data-independent and can be executed in parallel, overriding compiler dependency analysis.

Version A requires one copy of data of each iteration, thus the `#pragma acc data copy(u[0:n]) create(utmp[0:n]) copyin(f[0:n])` is put inside the outer most loop; while version B require only one with a single copy of data, thus the `#pragma acc data copy(u[0:n]) create(utmp[0:n]) copyin(f[0:n])` is put outside the outer most loop to only keep one copy.

Reference:

- https://www.cs.ucdavis.edu/~matloff/OpenACCDir/Intros158/FuLinMiller.pdf (P7)
- https://www.openacc.org/sites/default/files/inline-files/OpenACC_Programming_Guide_0.pdf (p17)

(Q1). Table for improvements in elapsed execution time

(*unit: seconds*)

With `ncells = 10^k`, `step=100`, For P31b.c Version B, we used `vector_size = 32` explicitly.

	k=6 Elpased Time	k=6 GPU Kernal Timing Data	k=7 Elpased Time	k=7 GPU Kernal Timing Data	k=8 Elpased Time	k=8 GPU Kernal Timing Data
version A: P31a.c	0.775286	0.239208	5.86755	2.344471	53.7508	23.567850
version B: P31b.c	0.310221	0.021626	0.543963	0.224418	3.0322	2.401439
Sequential Execution: jacob1d.c	0.107952	N/A	3.26706	N/A	32.8349	N/A

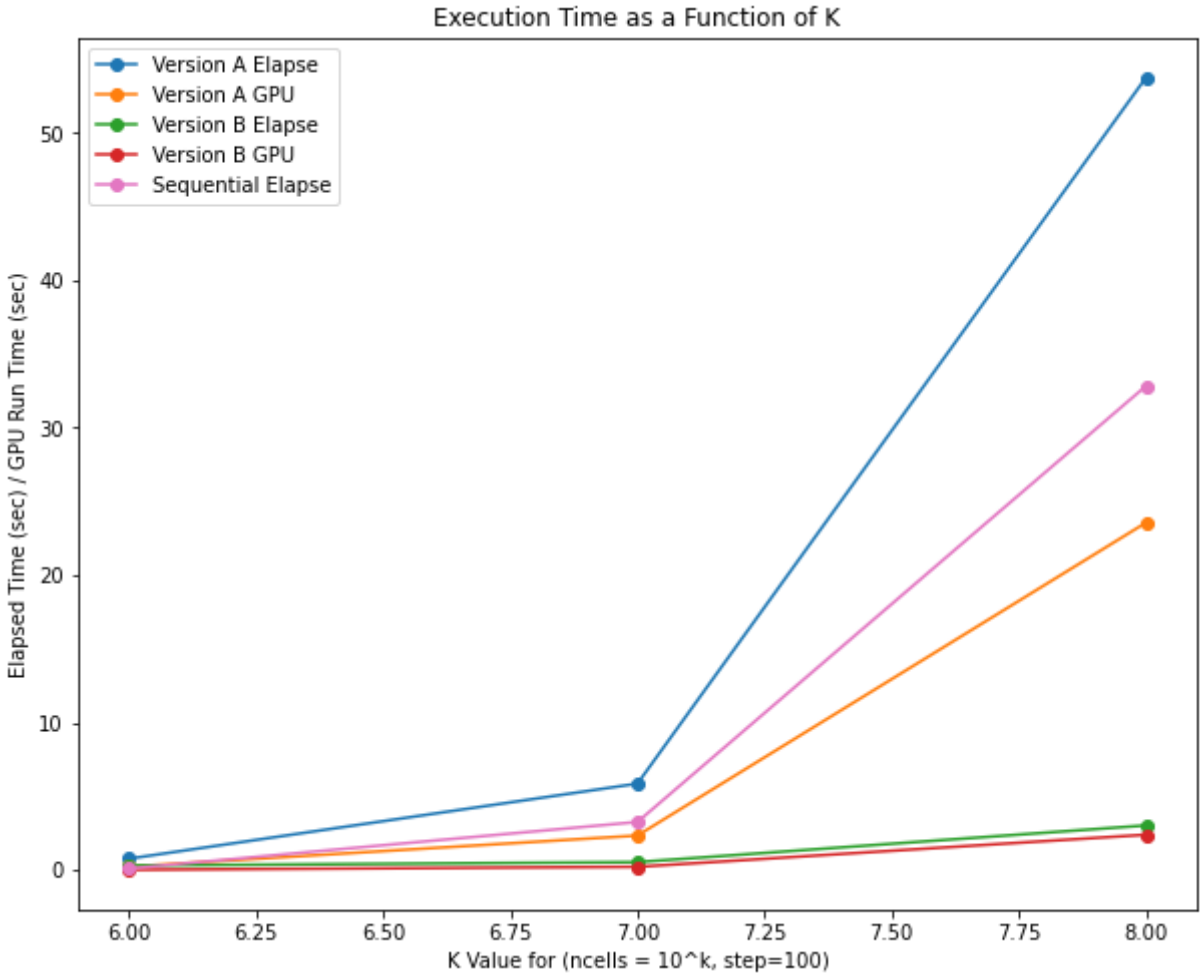
In [19]:

```
import matplotlib.pyplot as plt

k = [6,7,8]
ver_A_elapse = [0.775286, 5.86755, 53.7508]
ver_A_GPU = [0.239208,2.344471,23.567850]
ver_B_elapse = [0.310221, 0.543963, 3.0322]
ver_B_GPU = [0.021626,0.224418,2.401439]
sequential_elapse = [0.107952, 3.26706, 32.8349]


plt.figure(figsize = (10,8))
plt.plot(k, ver_A_elapse, '-ob', label = "Version A Elapse", color="tab:blue")
plt.plot(k, ver_A_GPU, '-ob', label = "Version A GPU",color="tab:orange")
plt.plot(k, ver_B_elapse, '-ob', label = "Version B Elapse",color = "tab:green")
plt.plot(k, ver_B_GPU, '-ob', label = "Version B GPU",color = "tab:red")
plt.plot(k, sequential_elapse, '-ob', label = "Sequential Elapse",color = "tab:pir


plt.xlabel('K Value for (ncells = 10^k, step=100)')
plt.ylabel('Elapsed Time (sec) / GPU Run Time (sec)')
plt.title('Execution Time as a Function of K')
plt.legend()
plt.show()
```



We could observe that `Version A` takes the longest elapsed time and is the slowest for all level of K, even if it uses openAcc parallelizatoin to do accelerate computing. As it copies data at each iteration, copy data creates too much overhead, thus the performance and efficiency is even slower than pure `sequential` execution.

Next to Version A elapsed time, it followed by sequential code with no acceleration, then version A GPU time; The version B elapsed time and GPU time is the most effective one.

We can also observe that GPU runtime provided by the driver is always less than elapsed time.

(Q2) With `ncells = 10^8 , step=100`

Unit: Seconds

Number of Threads	n=32	n=64	n=128	n=256	n=512	n=1024
Elapsed Time **version B**: P31b.c	3.0744	3.01042	2.93694	2.86104	2.81429	2.7706
GPU Driver Time **version B**: P31b.c	2.453468	2.390645	2.330780	2.251912	2.203696	2.161270

> We could see that as number of threads increases, the elapse time and GPU time also decrease and goes faster.
>
> It looks value `n=1024` for thread number is faster. When it divides matrix into the chunk, it divides 65535 grids into blocks of 1024 vector length. Vector length is number of threads in each gang/block. As worker is architecture specific, related to thread scheduling group, and we have 65535 grids, as number of threads in each

gang/block increase, the num_gang(number of blocks in the grid) decreases. When `vector_size=1024, grid=65535` , `num_gang=64` .

Looking at the device specific information for our NVIDIA Tesla M60, it has `32 warps (workers)` and `1024 maximum threads per block` . Thus we could see that fully utilize `32 warps` by setting `vector_size = 2048` might leads to even faster performance, as it uses all workers and didn't have reuse/communication/latency overheads. However, as the hard limit of max threads is 1024 here, the performance reaches best speed when thread number is 1024. It uses each of 32 workers twice. If we set thread number smaller than 1024, it will have to reuse 32 workers for multiple times and there might be some overheads in data transfer or communciation, making it slightly slower.

Thus the best thread here is `1024`

```
Warp Size:                    32
Maximum Threads per Block:    1024
```

Reference:

```
ubuntu@ip-172-31-36-197:~$ pgcc -acc -Minfo -DUSE_CLOCK P31b.c timing.c -o p31b_1024
P31b.c:
jacobi:
     29, Generating copyin(f[:n]) [if not already present]
         Generating create(utmp[:n]) [if not already present]
         Generating copy(u[:n]) [if not already present]
     33, Loop is parallelizable
         Generating Tesla code
         33, #pragma acc loop gang, vector(1024) /* blockIdx.x threadIdx.x */
     34, FMA (fused multiply-add) instruction(s) generated
     39, Loop is parallelizable
         Generating Tesla code
         39, #pragma acc loop gang, vector(1024) /* blockIdx.x threadIdx.x */
     40, FMA (fused multiply-add) instruction(s) generated
timing.c:
```

```
ubuntu@ip-172-31-36-197:~$ ./p31b_32 100000000 100
n: 100000000
nsteps: 100
Elapsed time: 3.0744 s

Accelerator Kernel Timing data
/home/ubuntu/P31b.c
  jacobi  NVIDIA  devicenum=0
    time(us): 2,453,468
    29: data region reached 2 times
        29: data copyin transfers: 96
             device time(us): total=204,333 max=2,209 min=1,515 avg=2,128
        42: data copyout transfers: 48
             device time(us): total=61,199 max=1,287 min=898 avg=1,274
    33: compute region reached 50 times
        33: kernel launched 50 times
            grid: [65535]  block: [32]
             device time(us): total=1,095,424 max=22,770 min=21,751 avg=21,908
            elapsed time(us): total=1,096,649 max=22,792 min=21,772 avg=21,932
    39: compute region reached 50 times
        39: kernel launched 50 times
            grid: [65535]  block: [32]
             device time(us): total=1,092,512 max=22,657 min=21,702 avg=21,850
            elapsed time(us): total=1,093,625 max=22,679 min=21,722 avg=21,872
```

```
ubuntu@ip-172-31-36-197:~$ ./p31b_64 100000000 100
n: 100000000
nsteps: 100
Elapsed time: 3.01042 s

Accelerator Kernel Timing data
/home/ubuntu/P31b.c
  jacobi  NVIDIA  devicenum=0
    time(us): 2,390,645
    29: data region reached 2 times
        29: data copyin transfers: 96
             device time(us): total=135,637 max=1,429 min=980 avg=1,412
        42: data copyout transfers: 48
             device time(us): total=61,205 max=1,288 min=898 avg=1,275
    33: compute region reached 50 times
        33: kernel launched 50 times
            grid: [65535]  block: [64]
             device time(us): total=1,097,169 max=21,974 min=21,908 avg=21,943
            elapsed time(us): total=1,098,327 max=21,998 min=21,927 avg=21,966
    39: compute region reached 50 times
        39: kernel launched 50 times
            grid: [65535]  block: [64]
             device time(us): total=1,096,634 max=21,970 min=21,884 avg=21,932
            elapsed time(us): total=1,097,705 max=21,991 min=21,904 avg=21,954
```

```
ubuntu@ip-172-31-36-197:~$ ./p31b_128 100000000 100
n: 100000000
nsteps: 100
Elapsed time: 2.93694 s

Accelerator Kernel Timing data
/home/ubuntu/P31b.c
  jacobi  NVIDIA  devicenum=0
    time(us): 2,330,780
    29: data region reached 2 times
        29: data copyin transfers: 96
             device time(us): total=135,621 max=1,499 min=978 avg=1,412
        42: data copyout transfers: 48
             device time(us): total=61,208 max=1,287 min=900 avg=1,275
    33: compute region reached 50 times
        33: kernel launched 50 times
            grid: [65535]  block: [128]
             device time(us): total=1,067,054 max=21,375 min=21,309 avg=21,341
            elapsed time(us): total=1,068,194 max=21,417 min=21,329 avg=21,363
    39: compute region reached 50 times
        39: kernel launched 50 times
            grid: [65535]  block: [128]
             device time(us): total=1,066,897 max=21,369 min=21,310 avg=21,337
            elapsed time(us): total=1,068,016 max=21,419 min=21,331 avg=21,360
```

```
ubuntu@ip-172-31-36-197:~$ ./p31b_256 100000000 100
n: 100000000
nsteps: 100
Elapsed time: 2.86104 s

Accelerator Kernel Timing data
/home/ubuntu/P31b.c
  jacobi  NVIDIA  devicenum=0
    time(us): 2,251,912
    29: data region reached 2 times
        29: data copyin transfers: 96
             device time(us): total=135,558 max=1,427 min=978 avg=1,412
        42: data copyout transfers: 48
             device time(us): total=61,192 max=1,288 min=898 avg=1,274
    33: compute region reached 50 times
        33: kernel launched 50 times
            grid: [65535]  block: [256]
             device time(us): total=1,027,967 max=20,577 min=20,542 avg=20,559
            elapsed time(us): total=1,029,074 max=20,618 min=20,563 avg=20,581
    39: compute region reached 50 times
        39: kernel launched 50 times
            grid: [65535]  block: [256]
             device time(us): total=1,027,195 max=20,565 min=20,528 avg=20,543
            elapsed time(us): total=1,028,270 max=20,606 min=20,547 avg=20,565
```

```
ubuntu@ip-172-31-36-197:~$ ./p31b_512 100000000 100
n: 100000000
nsteps: 100
Elapsed time: 2.8149 s

Accelerator Kernel Timing data
/home/ubuntu/P31b.c
  jacobi  NVIDIA  devicenum=0
    time(us): 2,203,775
    29: data region reached 2 times
        29: data copyin transfers: 96
             device time(us): total=135,581 max=1,426 min=979 avg=1,412
        42: data copyout transfers: 48
             device time(us): total=61,195 max=1,287 min=899 avg=1,274
    33: compute region reached 50 times
        33: kernel launched 50 times
            grid: [65535]  block: [512]
             device time(us): total=1,002,253 max=20,061 min=20,030 avg=20,045
            elapsed time(us): total=1,003,365 max=20,104 min=20,051 avg=20,067
    39: compute region reached 50 times
        39: kernel launched 50 times
            grid: [65535]  block: [512]
             device time(us): total=1,004,746 max=20,115 min=20,081 avg=20,094
            elapsed time(us): total=1,005,824 max=20,147 min=20,100 avg=20,116
```

```
ubuntu@ip-172-31-36-197:~$ ./p31b_1024 100000000 100
n: 100000000
nsteps: 100
Elapsed time: 2.7706 s

Accelerator Kernel Timing data
/home/ubuntu/P31b.c
  jacobi  NVIDIA  devicenum=0
    time(us): 2,161,270
    29: data region reached 2 times
        29: data copyin transfers: 96
             device time(us): total=135,567 max=1,425 min=978 avg=1,412
        42: data copyout transfers: 48
             device time(us): total=61,201 max=1,286 min=897 avg=1,275
    33: compute region reached 50 times
        33: kernel launched 50 times
            grid: [65535]  block: [1024]
             device time(us): total=982,706 max=19,667 min=19,641 avg=19,654
            elapsed time(us): total=983,766 max=19,707 min=19,661 avg=19,675
    39: compute region reached 50 times
        39: kernel launched 50 times
            grid: [65535]  block: [1024]
             device time(us): total=981,796 max=19,645 min=19,626 avg=19,635
            elapsed time(us): total=982,836 max=19,687 min=19,646 avg=19,656
ubuntu@ip-172-31-36-197:~$
```