Dylan Prosser
2/17/25
Stat129

1)
dylan@nsm-stats:~$ parallel 'zcat {} | grep ",TAVG,"' ::: /stat129/*.csv.gz > TAVG.csv
dylan@nsm-stats:~$ ./count_tavg.sh
dylan@nsm-stats:~$

parallel: Runs the specified command across multiple CPUs, reducing load time

'zcat {} | grep ",TAVG,"': Decompresses each file (zcat {}) and searches for lines containing ",TAVG," (grep ",TAVG,").

::: /stat129/*.csv.gz: Supplies each CSV.gz file in /stat129 as an argument to the {} placeholder in parallel.

> TAVG.csv: Redirects all extracted TAVG lines from every parallel job into the file TAVG.csv.

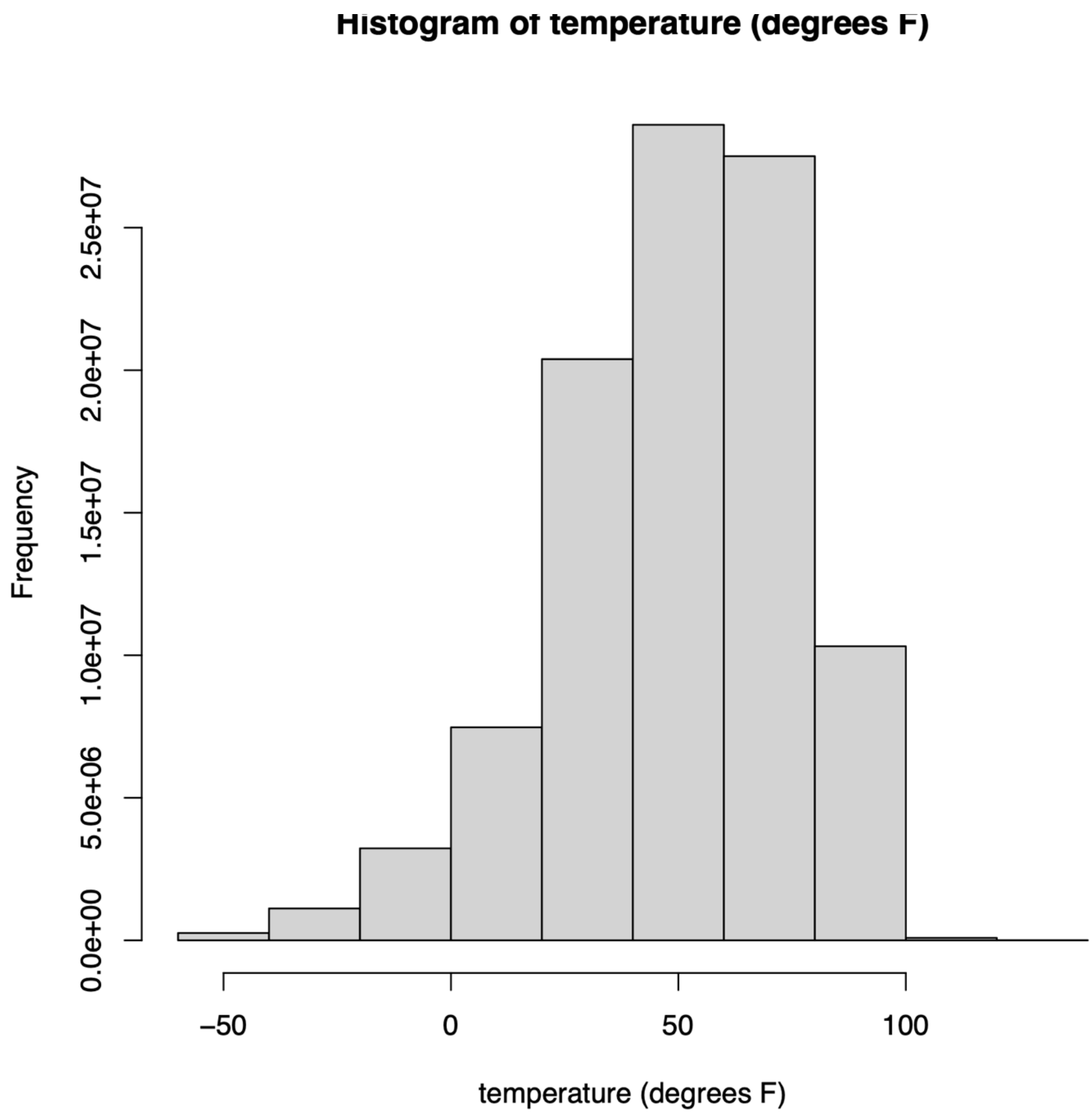
I modified count_months.sh to suit the needs of count TAVG

./count_tavg.sh: I executed my script which counted all temperatures and made a count for how often it occurred

2)
dylan@nsm-stats:~$ wc -l TAVG_counts.txt
2489 TAVG_counts.txt

2489 different temperatures were recorded, this was calculated by counting unique numbers after all temperature recordings were sorted numerically. Then we counted the lines in the file and that's how many different recordings there were.

3)
dylan@nsm-stats:~$ exit
logout
Connection to nsm-stats closed.
dylan@Dylans-Laptop ~ % sftp dylan@nsm-stats
dylan@nsm-stats's password:
Connected to nsm-stats.
sftp> get temp_histogram.pdf
Fetching /home/dylan/temp_histogram.pdf to temp_histogram.pdf
temp_histogram.pdf 100% 4640 83.2KB/s 00:00
sftp> bye



This Histogram represents the count of Temperatures recorded across all years in the stat129 database. These kinds of numbers were honestly expected, the frequency is so high due to the fact that these recordings are from over 100 (estimate) different stations across almost 200 years.

4) I originally ran it in parallel, so I will do the reverse.

dylan@nsm-stats:~$ time zcat /stat129/*.csv.gz | grep ",TAVG," > TAVG2.csv

real 4m54.199s
user 5m33.647s
sys 0m22.969s
dylan@nsm-stats:~$

```
   PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
110632 dylan     20   0    3560   1536   1152 R  99.7   0.0   0:08.47 gzip
110633 dylan     20   0    6544   2304   2112 S  22.9   0.0   0:02.12 grep
 95255 saqif     20   0   15128   6900   4992 S   0.7   0.0   0:05.25 sshd
 41063 spencer   20   0   10448   6528   3648 S   0.3   0.0   1:39.56 htop
 96458 root      20   0       0      0      0 I   0.3   0.0   0:09.74 kworker/u48+
     1 root      20   0   22536  12888   9240 S   0.0   0.0   0:02.59 systemd
     2 root      20   0       0      0      0 S   0.0   0.0   0:00.02 kthreadd
     3 root      20   0       0      0      0 S   0.0   0.0   0:00.00 pool_workqu+
     4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-r+
     5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-r+
     6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-s+
     7 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-n+
     9 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0+
    12 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-m+
    13 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_k+
    14 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_r+
```

The Result of the parallel processing reduced load times significantly, by putting more CPUs to work and reducing the overall bottle necks. I believe there is a bottle neck with gzip, due to high CPU usage compared to grep. And this is the main hold for the long load times.

5)
dylan@nsm-stats:~$ time parallel 'zcat {} | grep ",TAVG,"' ::: /stat129/*.csv.gz > TAVG2.csv

real	0m20.688s
user	6m20.932s
sys	0m27.772s
dylan@nsm-stats:~$

htop:

```
    PID USER       PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 107938 alexand+   20   0    3560   1536   1152 R  83.7   0.0   4:25.04 gzip
 110400 dylan      20   0    3556   1536   1152 R  81.4   0.0   0:02.56 gzip
 110394 dylan      20   0    3556   1536   1152 R  80.7   0.0   0:02.88 gzip
 110388 dylan      20   0    3556   1536   1152 R  80.1   0.0   0:02.92 gzip
 110403 dylan      20   0    3556   1536   1152 R  79.1   0.0   0:02.44 gzip
 110379 dylan      20   0    3556   1536   1152 R  78.1   0.0   0:02.94 gzip
 110385 dylan      20   0    3556   1536   1152 S  76.7   0.0   0:02.84 gzip
 110391 dylan      20   0    3556   1536   1152 R  75.7   0.0   0:02.70 gzip
 110382 dylan      20   0    3556   1536   1152 R  72.8   0.0   0:02.85 gzip
 107933 faisal     20   0    3560   1536   1152 R  70.8   0.0   4:21.18 gzip
 110406 dylan      20   0    3556   1536   1152 R  68.4   0.0   0:02.06 gzip
 110409 dylan      20   0    3556   1536   1152 R  63.8   0.0   0:01.92 gzip
 110364 dylan      20   0    3556   1536   1152 R  61.5   0.0   0:02.67 gzip
 110415 dylan      20   0    3556   1536   1152 R  59.5   0.0   0:01.79 gzip
 108601 hadia      20   0    3560   1536   1152 R  58.5   0.0   3:29.19 gzip
 110412 dylan      20   0    3556   1536   1152 R  56.8   0.0   0:01.71 gzip
 110397 dylan      20   0    3556   1536   1152 R  52.2   0.0   0:01.64 gzip
 110373 dylan      20   0    3556   1536   1152 R  51.2   0.0   0:02.60 gzip
 110370 dylan      20   0    3556   1536   1152 R  48.2   0.0   0:02.39 gzip
 110418 dylan      20   0    3556   1536   1152 R  40.9   0.0   0:01.23 gzip
 110421 dylan      20   0    3556   1536   1152 R  35.9   0.0   0:01.08 gzip
 108604 hadia      20   0   27588   9984   3072 S  32.9   0.0   1:23.05 sort
 110424 dylan      20   0    3556   1536   1152 R  29.9   0.0   0:00.90 gzip
 107936 faisal     20   0   27464   9792   3072 S  29.6   0.0   0:39.91 sort
 110427 dylan      20   0    3556   1536   1152 S  28.6   0.0   0:00.86 gzip
 110430 dylan      20   0    3556   1536   1152 R  17.9   0.0   0:00.54 gzip
 110395 dylan      20   0    6544   2112   2112 S  17.6   0.0   0:00.61 grep
 110380 dylan      20   0    6544   2304   2112 R  16.9   0.0   0:00.63 grep
 110389 dylan      20   0    6544   2112   2112 S  16.9   0.0   0:00.61 grep
 110401 dylan      20   0    6544   2112   2112 R  16.9   0.0   0:00.53 grep
 110404 dylan      20   0    6544   2112   2112 S  16.9   0.0   0:00.52 grep
 107941 alexand+   20   0   27464   9408   2880 S  16.6   0.0   0:21.26 sort
 108602 hadia      20   0    6676   2112   2112 S  16.6   0.0   0:57.02 grep
 110386 dylan      20   0    6544   2112   2112 R  16.6   0.0   0:00.61 grep
 110392 dylan      20   0    6544   2112   2112 S  16.3   0.0   0:00.57 grep
 110383 dylan      20   0    6544   2112   2112 S  15.3   0.0   0:00.60 grep
```

The program extracts all TAVG (average daily temperature) observations from multiple compressed CSV files and saves them to a single file. First, we decompress each .csv.gz and filter lines for ,TAVG,. Normally, this would be done serially by piping zcat outputs into grep. To run in parallel, we used GNU Parallel and replaced the single zcat /stat129/*.csv.gz with a parallel command template, such as parallel 'zcat {} | grep ",TAVG,"' ::: /stat129/*.csv.gz > TAVG2.csv. This way, each .csv.gz file is processed by a different CPU core or process simultaneously, combining all results into one output file and reducing overall execution time. The Result of the parallel processing reduced load times significantly, by putting more CPUs to work and reducing the overall bottle necks.