CS 433 (2) HW3 Report

Boris Nikulin

2019-03-29

Contents

1 Data Analysis

1.1 Data Import

First we load the simulation data, filter out processes that never finished, and give the data a once over.

```
library(readr)
library(dplyr)
library(magrittr)

col_spec <- cols(
    scheduler = col_character(),
    pid = col_integer(),
    arrival = col_integer(),
    finish = col_integer(),
    cpu = col_integer(),
    io = col_integer(),
    wait = col_integer()
)

data10 <- read_csv('./os_sim_10.csv', col_types = col_spec)
data20 <- read_csv('./os_sim_20.csv', col_types = col_spec)
data100 <- read_csv('./os_sim_100.csv', col_types = col_spec)
data1000 <- read_csv('./os_sim_1000.csv', col_types = col_spec)
```

```r
data10$num_procs <- 10
data20$num_procs <- 20
data100$num_procs <- 100
data1000$num_procs <- 1000

glimpse(data10, width = 60)
```

```
## Observations: 20
## Variables: 8
## $ scheduler <chr> "FCFS", "FCFS", "FCFS", "FCFS", "FCFS...
## $ pid       <int> 9, 8, 7, 6, 1, 0, 2, 3, 4, 5, 9, 8, 7...
## $ arrival   <int> 290671, 274023, 271748, 250512, 40644...
## $ finish    <int> 0, 0, 0, 0, 46512, 53631, 224614, 0, ...
## $ cpu       <int> 1971, 9584, 7586, 17198, 3024, 6755, ...
## $ io        <int> 1951, 6210, 7253, 16848, 2581, 7474, ...
## $ wait      <int> 5326, 10167, 13339, 15441, 263, 1305,...
## $ num_procs <dbl> 10, 10, 10, 10, 10, 10, 10, 10, 10, 1...
```

```r
glimpse(data20, width = 60)
```

```
## Observations: 40
## Variables: 8
## $ scheduler <chr> "FCFS", "FCFS", "FCFS", "FCFS", "FCFS...
## $ pid       <int> 19, 18, 17, 16, 5, 4, 3, 2, 0, 1, 6, ...
## $ arrival   <int> 298950, 297875, 290671, 290319, 83552...
## $ finish    <int> 0, 0, 0, 0, 0, 196672, 249049, 209215...
## $ cpu       <int> 15, 102, 1263, 575, 11303, 24683, 493...
## $ io        <int> 96, 211, 1227, 1252, 61354, 29269, 48...
## $ wait      <int> 532, 1376, 6820, 7703, 143596, 76408,...
## $ num_procs <dbl> 20, 20, 20, 20, 20, 20, 20, 20, 20, 2...
```

```r
glimpse(data100, width = 60)
```

```
## Observations: 200
## Variables: 8
## $ scheduler <chr> "FCFS", "FCFS", "FCFS", "FCFS", "FCFS...
## $ pid       <int> 99, 98, 97, 96, 95, 94, 93, 92, 91, 9...
## $ arrival   <int> 298950, 298232, 297875, 294344, 29118...
## $ finish    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ cpu       <int> 0, 0, 0, 58, 42, 7, 42, 167, 162, 121...
## $ io        <int> 0, 0, 0, 31, 52, 42, 99, 95, 144, 127...
## $ wait      <int> 0, 0, 0, 4791, 4688, 4605, 9516, 9371...
## $ num_procs <dbl> 100, 100, 100, 100, 100, 100, 100, 10...
```

```r
glimpse(data1000, width = 60)
```

```
## Observations: 2,000
## Variables: 8
```

```
## $ scheduler <chr> "FCFS", "FCFS", "FCFS", "FCFS", "FCFS...
## $ pid       <int> 999, 998, 997, 996, 995, 994, 993, 99...
## $ arrival   <int> 299735, 298954, 298950, 298852, 29836...
## $ finish    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ cpu       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ io        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ wait      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ num_procs <dbl> 1000, 1000, 1000, 1000, 1000, 1000, 1...
```

```
data <- rbind(data10, data20, data100, data1000) %>%
    # filter out processes that never finished
    filter(finish != 0) %>%
    # convert ms to s
    mutate_at(.funs = list(~./1000), .vars = 3:7) %>%
    # col spec of col_factor() doesnt seem to work
    mutate(scheduler = as.factor(scheduler),
        turnaround = finish - arrival)

glimpse(data, width = 60)
```

```
## Observations: 34
## Variables: 9
## $ scheduler  <fct> FCFS, FCFS, FCFS, FCFS, FCFS, SJF, S...
## $ pid        <int> 1, 0, 2, 4, 5, 1, 0, 2, 4, 4, 3, 2, ...
## $ arrival    <dbl> 40.644, 38.097, 66.312, 189.715, 244...
## $ finish     <dbl> 46.512, 53.631, 224.614, 236.296, 26...
## $ cpu        <dbl> 3.024, 6.755, 58.732, 24.784, 7.550,...
## $ io         <dbl> 2.581, 7.474, 89.009, 19.978, 11.004...
## $ wait       <dbl> 0.263, 1.305, 10.561, 1.819, 2.479, ...
## $ num_procs  <dbl> 10, 10, 10, 10, 10, 10, 10, 10, 10, ...
## $ turnaround <dbl> 5.868, 15.534, 158.302, 46.581, 21.0...
```

The times are in seconds.

```
library(xtable)

data_summary <- data %>%
    group_by(scheduler, num_procs) %>%
    summarise(
        mean_arrival = mean(arrival),
        sd_arrival = sd(arrival),
        mean_finish = mean(finish),
        sd_finish = sd(finish),
        mean_cpu = mean(cpu),
        sd_cpu = sd(cpu),
        mean_io = mean(io),
        sd_io = sd(io),
```

```
        mean_wait = mean(wait),
        sd_wait = sd(wait),
        mean_turnaround = mean(turnaround),
        sd_turnaround = sd(turnaround),
        throughput = n() / (max(finish) - min(arrival))
    )

data_summary %>%
    select(scheduler, num_procs, mean_arrival:sd_arrival) %>%
    xtable()
```

	scheduler	num_procs	mean_arrival	sd_arrival
1	FCFS	10.00	115.84	95.06
2	FCFS	20.00	84.63	60.60
3	FCFS	100.00	31.97	44.92
4	SJF	10.00	83.69	71.82
5	SJF	20.00	60.49	27.10
6	SJF	100.00	87.34	52.02
7	SJF	1000.00	71.57	49.53

```
data_summary %>%
    select(scheduler, num_procs, mean_finish:sd_finish) %>%
    xtable()
```

	scheduler	num_procs	mean_finish	sd_finish
1	FCFS	10.00	165.30	106.27
2	FCFS	20.00	178.39	63.24
3	FCFS	100.00	238.52	40.24
4	SJF	10.00	140.31	104.36
5	SJF	20.00	132.29	65.43
6	SJF	100.00	156.94	48.93
7	SJF	1000.00	161.99	49.81

```
data_summary %>%
    select(scheduler, num_procs, mean_cpu:sd_io) %>%
    xtable()
```

```
data_summary %>%
    select(scheduler, num_procs, mean_wait:sd_wait) %>%
    xtable()
```

```
data_summary %>%
    select(scheduler, num_procs, mean_turnaround:throughput) %>%
    xtable()
```

	scheduler	num_procs	mean_cpu	sd_cpu	mean_io	sd_io
1	FCFS	10.00	20.17	23.14	26.01	35.79
2	FCFS	20.00	22.79	16.09	22.54	15.81
3	FCFS	100.00	12.11	5.30	10.18	5.49
4	SJF	10.00	23.32	25.45	29.66	40.00
5	SJF	20.00	21.44	19.55	26.69	19.34
6	SJF	100.00	8.64	4.90	40.26	26.67
7	SJF	1000.00	6.45	4.44	71.21	49.11

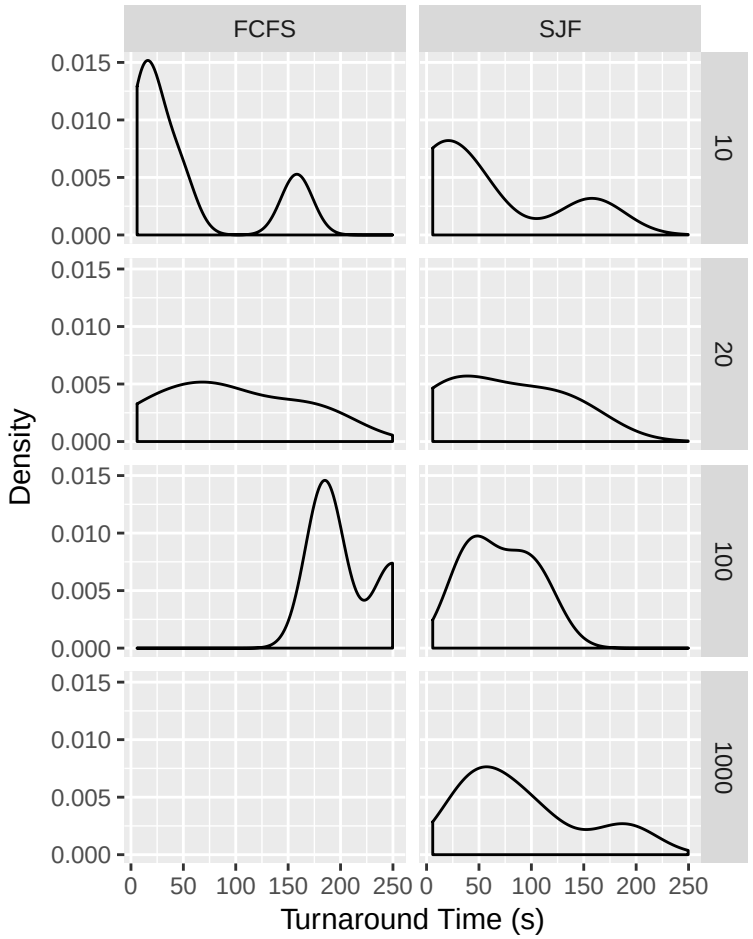
	scheduler	num_procs	mean_wait	sd_wait
1	FCFS	10.00	3.29	4.15
2	FCFS	20.00	48.43	35.67
3	FCFS	100.00	184.26	29.23
4	SJF	10.00	3.63	4.71
5	SJF	20.00	23.67	20.33
6	SJF	100.00	20.70	11.80
7	SJF	1000.00	12.76	8.37

	scheduler	num_procs	mean_turnaround	sd_turnaround	throughput
1	FCFS	10.00	49.46	62.68	0.02
2	FCFS	20.00	93.76	65.91	0.04
3	FCFS	100.00	206.54	37.41	0.01
4	SJF	10.00	56.61	69.82	0.02
5	SJF	20.00	71.81	57.08	0.02
6	SJF	100.00	69.60	33.02	0.03
7	SJF	1000.00	90.43	61.40	0.03

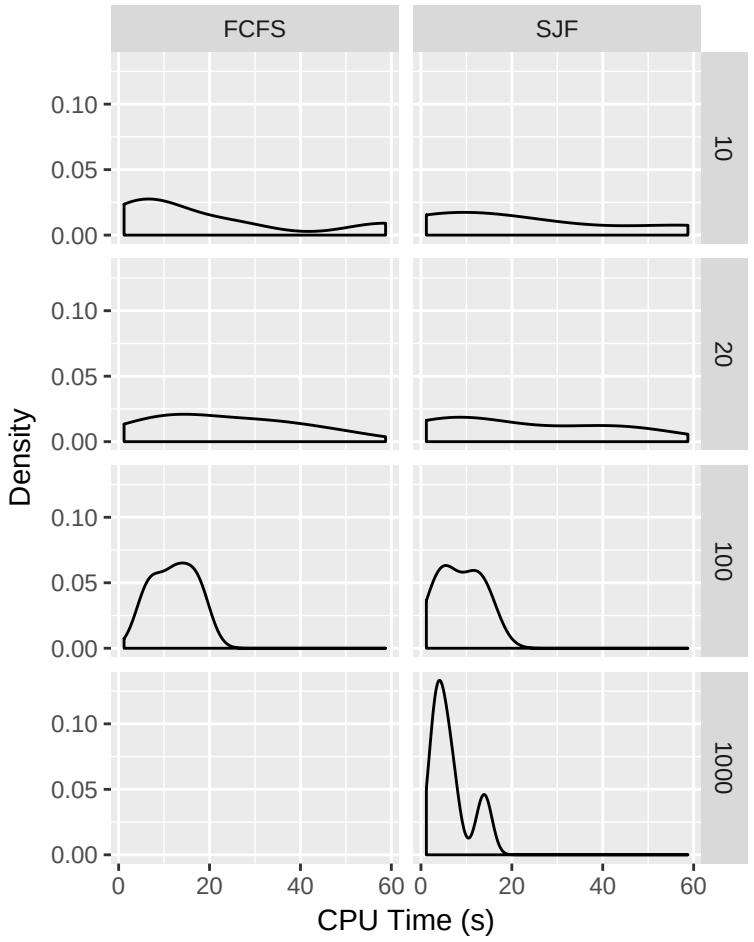
1.2 Data Visualization

Next we plot the data using density plots and line plots.

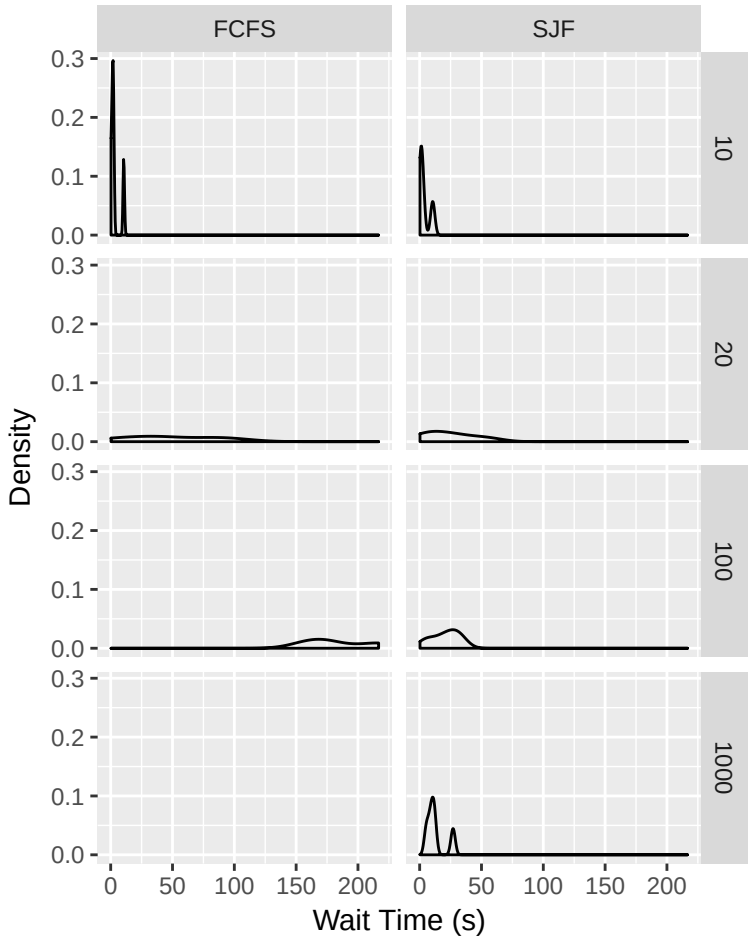
```
library(ggplot2)

ggplot(data, aes(turnaround)) +
    geom_density() +
    labs(x = 'Turnaround Time (s)', y = 'Density') +
    facet_grid(num_procs~scheduler)
```



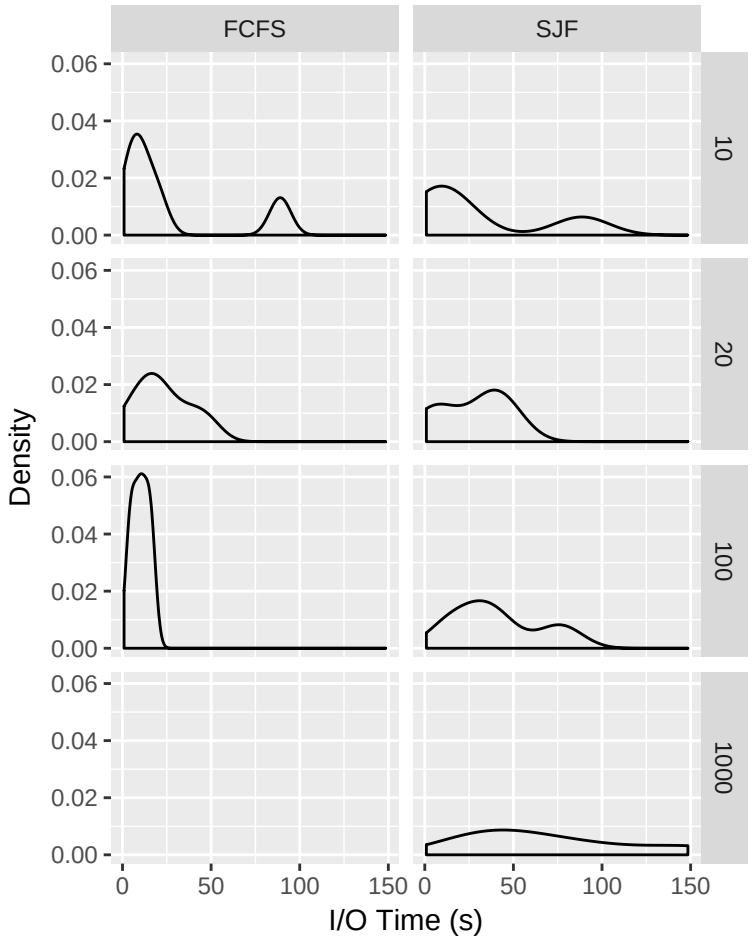

```
ggplot(data, aes(cpu)) +
    geom_density() +
    labs(x = 'CPU Time (s)', y = 'Density') +
    facet_grid(num_procs~scheduler)
```




```
ggplot(data, aes(wait)) +
    geom_density() +
    labs(x = 'Wait Time (s)', y = 'Density') +
    facet_grid(num_procs~scheduler)
```



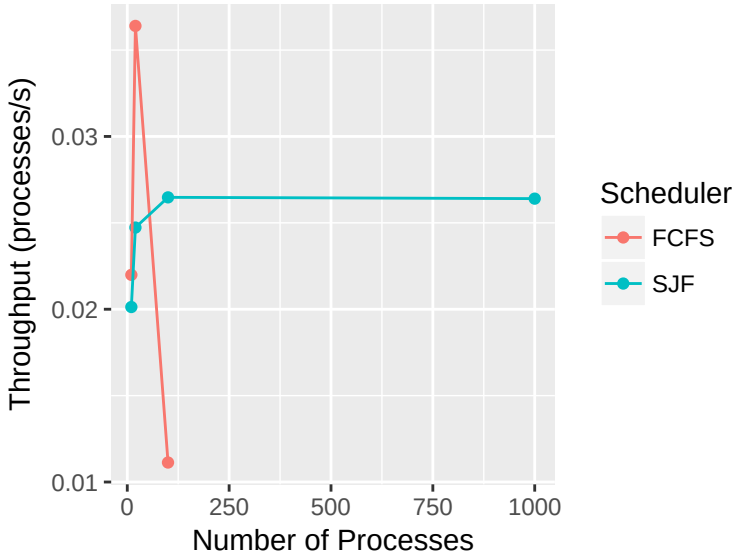

```
ggplot(data, aes(io)) +
    geom_density() +
    labs(x = 'I/O Time (s)', y = 'Density') +
    facet_grid(num_procs~scheduler)
```



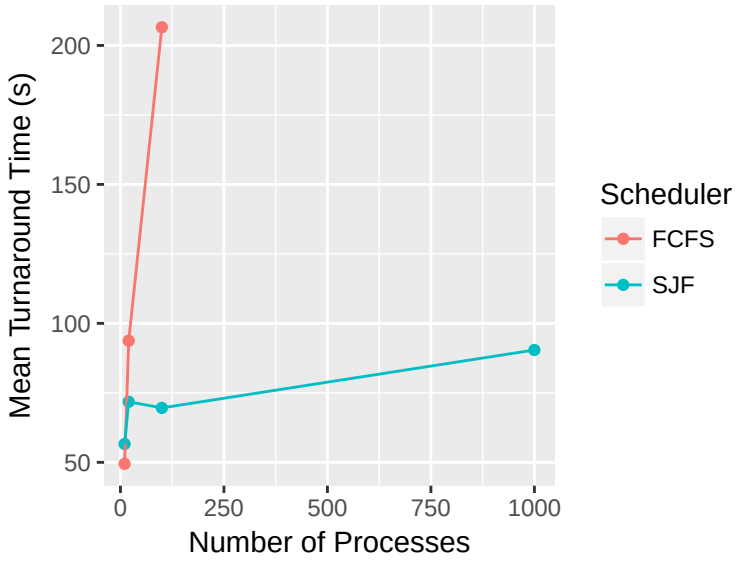
The density plots for FCFS at 1000 processes does not contain anything because when the small CPU burst finishes, the processes has to wait for IO and all the processes already in the ready queue which is a staggering number. Thus, every processes blocks each other for a very long time — longer than the simulation duration of 5 min — and no process can complete.

```
ggplot(data_summary,
        aes(
            num_procs,
            throughput,
            color = scheduler)) +
    geom_point() +
    geom_line() +
    labs(
        x = 'Number of Processes',
```

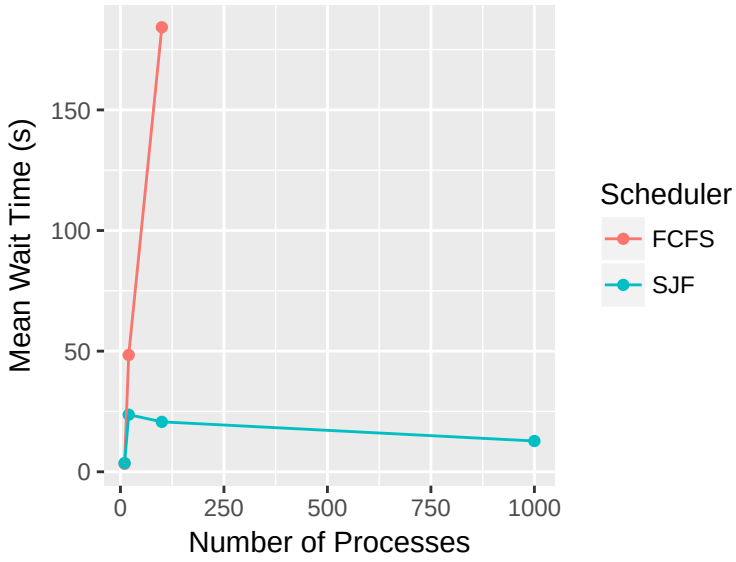
```
        y = 'Throughput (processes/s)',
        color = 'Scheduler')
```



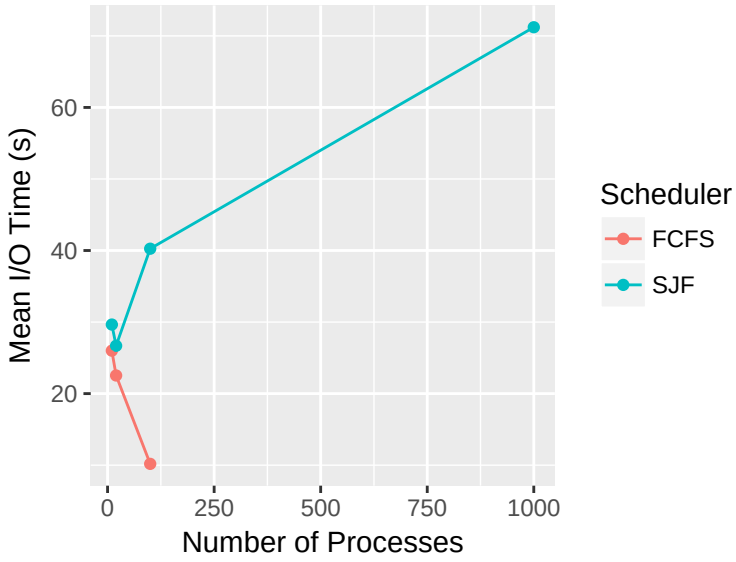

```
ggplot(data_summary,
        aes(
            num_procs,
            mean_turnaround,
            color = scheduler)) +
    geom_point() +
    geom_line() +
    labs(
        x = 'Number of Processes',
        y = 'Mean Turnaround Time (s)',
        color = 'Scheduler')
```




```
ggplot(data_summary,
        aes(num_procs,
            mean_wait,
            color = scheduler)) +
    geom_point() +
    geom_line() +
    labs(
        x = 'Number of Processes',
        y = 'Mean Wait Time (s)',
        color = 'Scheduler')
```




```
ggplot(data_summary,
        aes(num_procs,
            mean_io,
            color = scheduler)) +
    geom_point() +
    geom_line() +
    labs(
        x = 'Number of Processes',
        y = 'Mean I/O Time (s)',
        color = 'Scheduler')
```



2 Scheduling Algorithm Analysis

From the throughput vs. number of processes graph, one can see that at a sufficiently small number of processes, FCFS does a better job at maximizing throughput. However, when the number of processes is too high, the time to traverse the ready queue in FCFS is incredibly high. This leads to FCFS failing drastically. Conversely, SJF doesn't beat FCFS on throughput, but is stable under a large amount of processes.

From the mean wait time vs. number of processes, one can see that SJF is clearly better at minimizing wait times. Also, as noted before, SJF does not choke on a large amount of processes.

In conclusion, FCFS maximizes throughput and SJF minimizes wait times. However, with limited resources and a sufficiently large amount of processes, FCFS grinds to a halt.