Concurrent Web Server

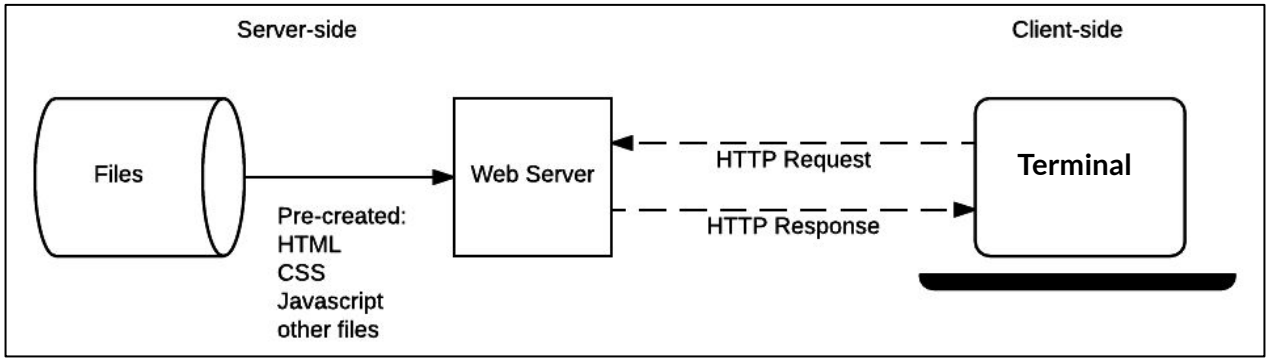
Contributors:
Kanishk Kalra (17110067)
Kavita Vaishnaw (17110073)
Mohit Mina (17110078)
Prakash R (17110109)
Shaurya Agarwal (17110145)

Project Description

Make a concurrent web server with scheduling policies and security considerations.

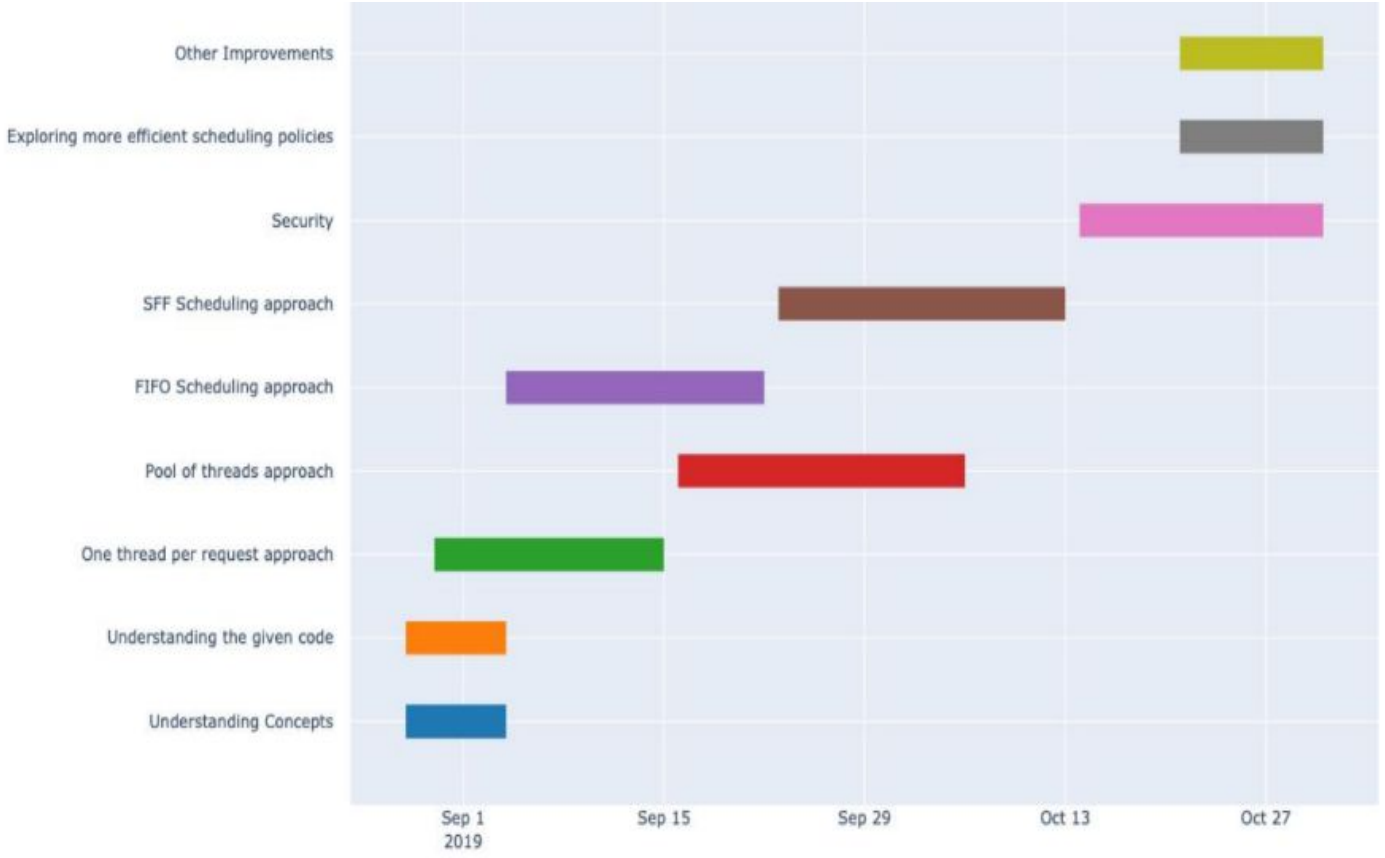
Aim:

- Webserver should handle multiple requests simultaneously.
- It should have 2 scheduling policies (FCFS/FIFO and SFF).
- Web server should have some security considerations..

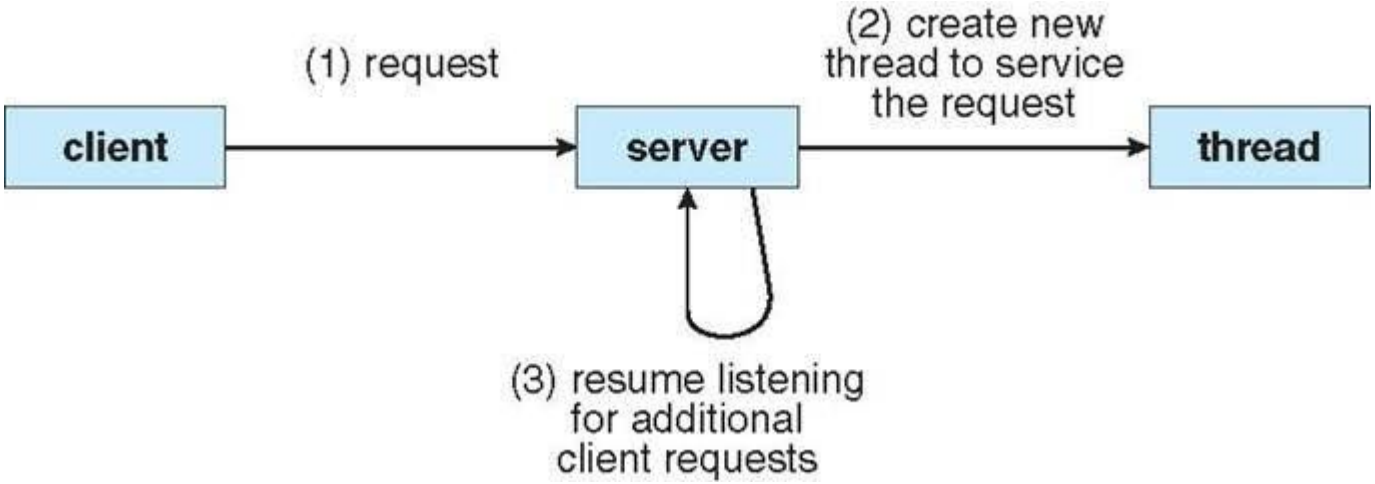


Source: https://mdn.mozillademos.org/files/13841/Basic%20Static%20App%20Server.png

Gantt Chart

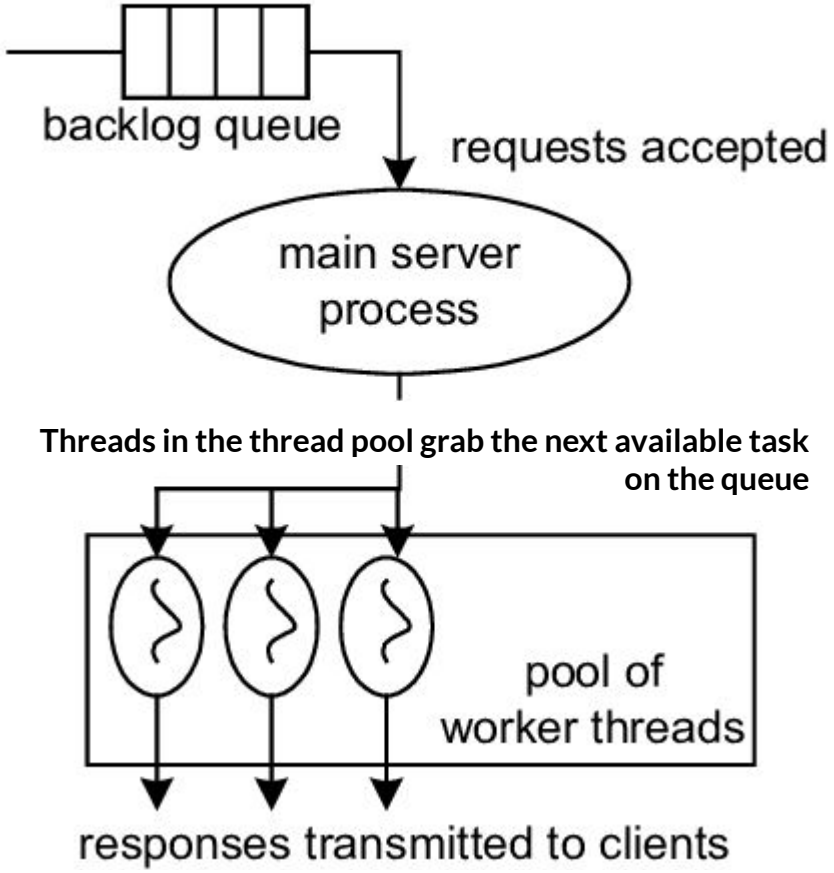

Other Improvements
Exploring more efficient scheduling policies
Security
SFF Scheduling approach
FIFO Scheduling approach
Pool of threads approach
One thread per request approach
Understanding the given code
Understanding Concepts
Sep 1
2019
Sep 15
Sep 29
Oct 13
Oct 27

One Thread per Request Approach



Source:
http://www.shubhsblog.com/programming/multithreaded-webserver-java.html

Pool of Threads with FCFS Scheduling



Source: https://www.classes.cs.uchicago.edu/archive/2016/spring/12300-1/threadpool.png

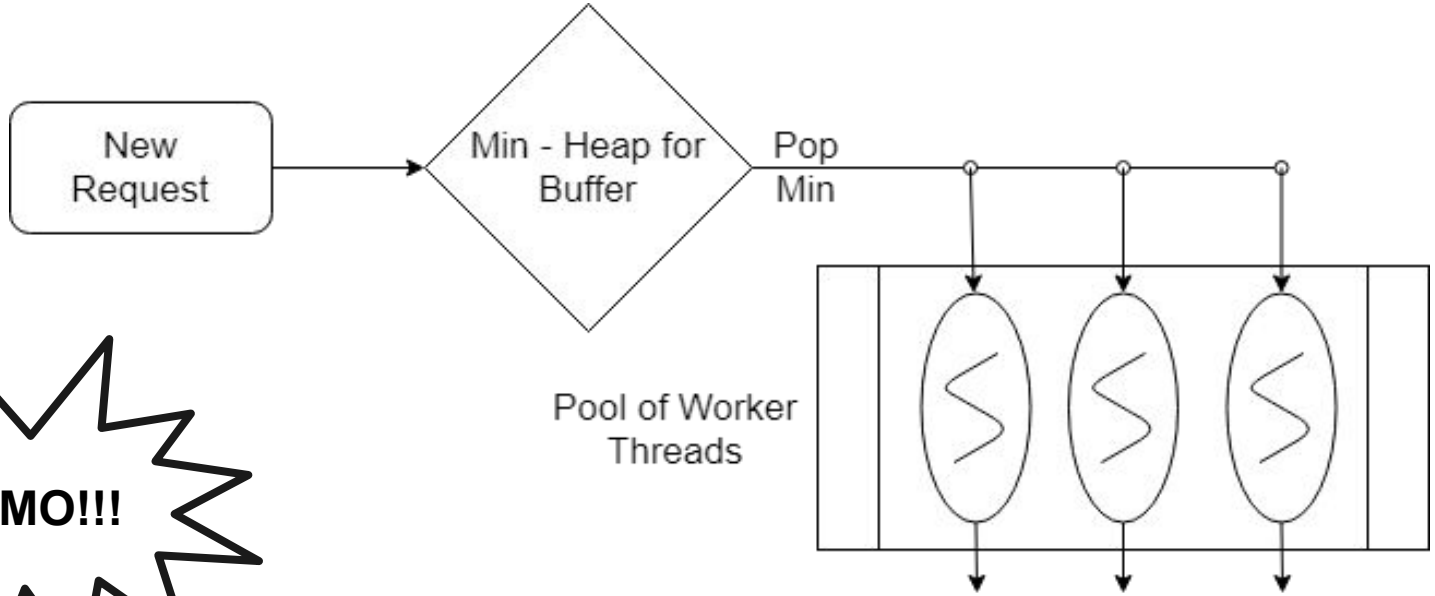
Structure of Thread Pool

```
18  struct thread_pool{
19      pthread_t *workers; //worker threads
20      struct q_node* queue; //queue that holds tasks waiting to be executed
21      int q_head, q_tail;
22      int max_workers;
23      int TASK_Q_MAX;
24      int scheduled; //keeps track of number of tasks scheduled
25      pthread_mutex_t mutex;
26      pthread_cond_t task_available; //signaled on when we go from no task to task available
27      pthread_cond_t no_task_left; //signaled on when no more tasks scheduled
28
29  };
```

```c
void *worker_func(void *pool_arg){
    struct thread_pool *pool = (struct thread_pool *)pool_arg;
    while(1){
        struct q_node task_picked;
        pthread_mutex_lock(&pool->mutex);

        while (pool->q_head == pool->q_tail){
            //empty queue
            pthread_cond_wait(&pool->task_available, &pool->mutex);
        }

        //assert(pool->q_head != pool->q_tail);
        task_picked = pool->queue[pool->q_head % pool->TASK_Q_MAX];
        pool->q_head++;
        pool->scheduled++; //task scheduled

        pthread_mutex_unlock(&pool->mutex);

        task_picked.routine(task_picked.arg);

        pthread_mutex_lock(&pool->mutex);



        pool->scheduled--;

        if (pool->scheduled == 0){
            pthread_cond_signal(&pool->no_task_left);
        }

        pthread_mutex_unlock(&pool->mutex);
    }

    return NULL;
}
```

Worker Function

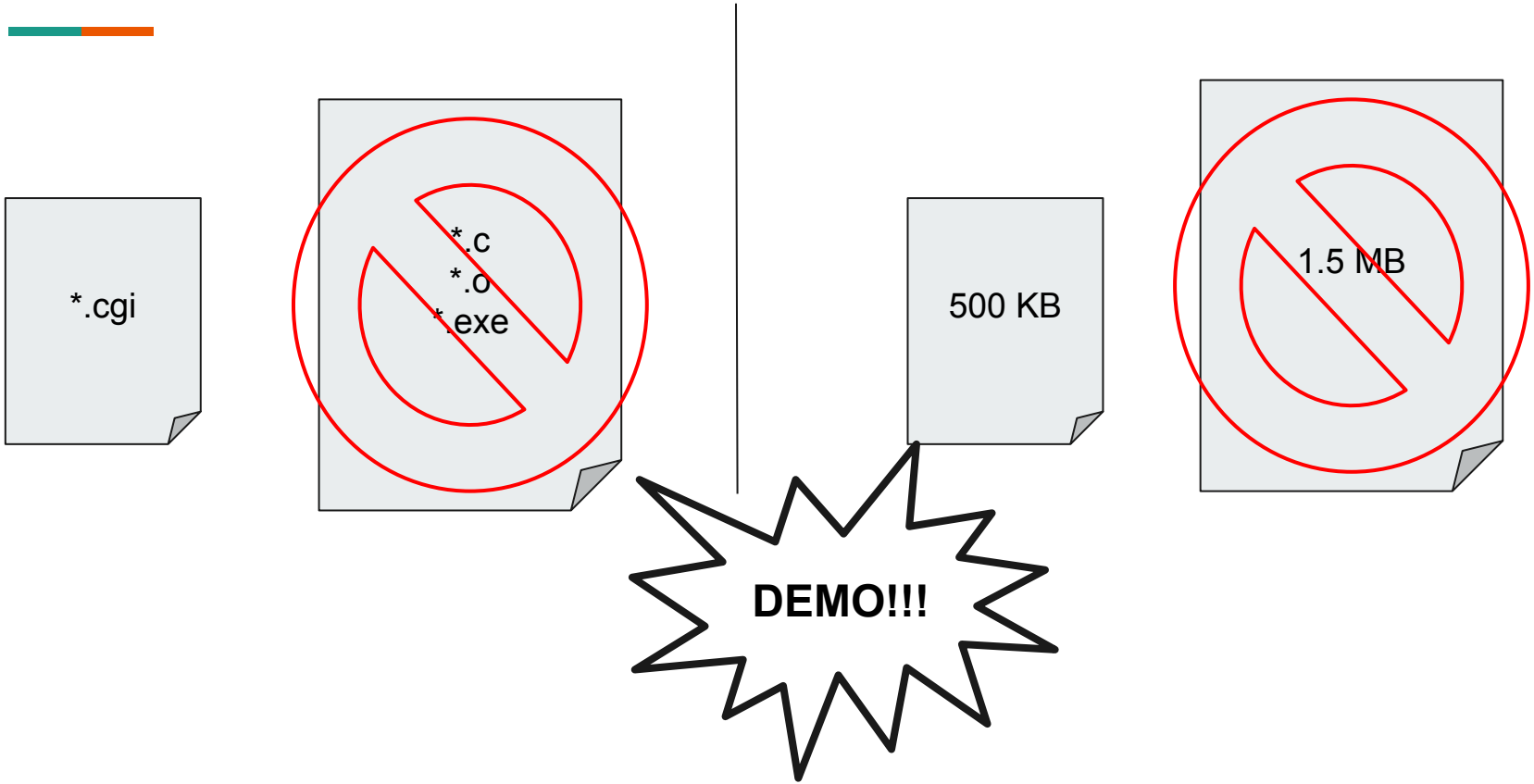
Shortest File First (SFF) Scheduling



```c
void pool_add_task(struct thread_pool *pool, void *(*routine)(void*), void *arg,void *tem){
    struct info* temp = (struct info*) tem;
    pthread_mutex_lock(&pool->mutex);
    if ((pool->heap)->count==0){
        pthread_cond_broadcast(&pool->task_available);
    }
    struct q_node task;
    task.routine = routine;
    // task.arg = arg;
    task.size = temp->size;
    task.req = *temp;
    insert(pool->heap,task);
    print_heap(pool);
    pthread_mutex_unlock(&pool->mutex);
}
```

```c
void insert(Heap *h, struct q_node key){
    if( h->count < h->capacity){
        h->arr[h->count] = key;
        heapify_bottom_top(h, h->count);
        h->count++;
    }
}
```

Security

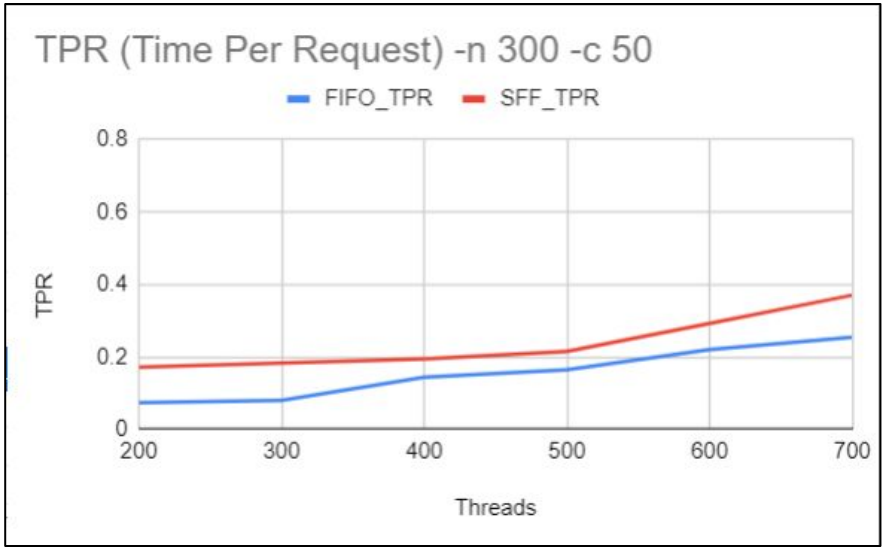


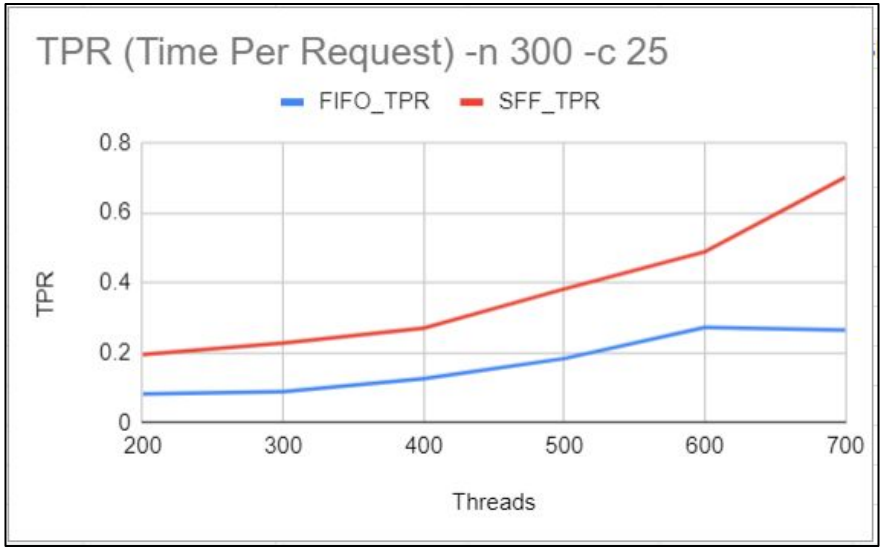
Apache Benchmark Testing

Apache Bench (ab) is a load testing and benchmarking tool for Hypertext Transfer Protocol (HTTP) server.

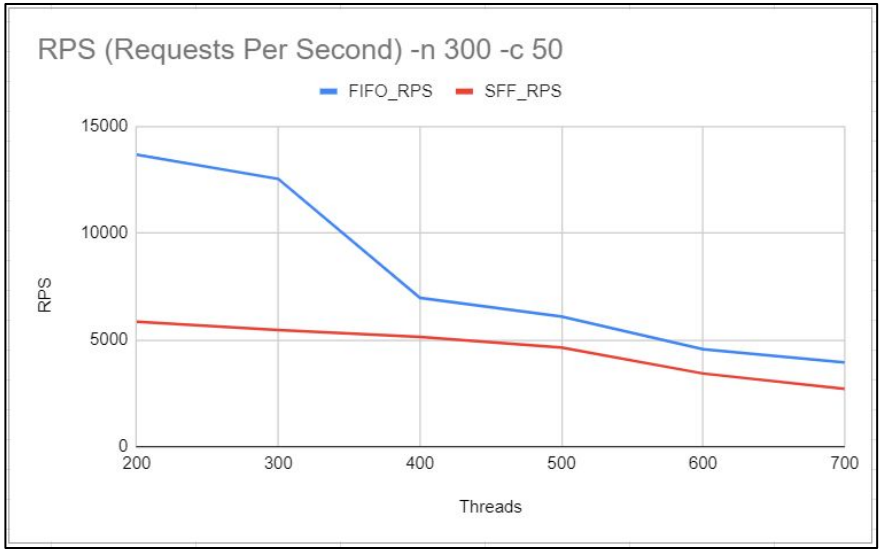
```
Server Software:        OSTEP
Server Hostname:        localhost
Server Port:            10000

Document Path:          /
Document Length:        3271 bytes

Concurrency Level:      100
Time taken for tests:   0.007 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      336200 bytes
HTML transferred:       327100 bytes
Requests per second:    13548.30 [#/sec] (mean)
Time per request:       7.381 [ms] (mean)
Time per request:       0.074 [ms] (mean, across all concurrent requests)
Transfer rate:          44481.82 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        1    2   0.3      2       2
Processing:     1    2   1.3      2       4
Waiting:        1    2   1.2      1       4
Total:          3    4   1.0      3       5

Percentage of the requests served within a certain time (ms)
  50%      3
  66%      4
  75%      5
  80%      5
  90%      5
  95%      5
  98%      5
  99%      5
 100%      5 (longest request)
```

TPR Comparison





RPS Comparison



Contributions of Members

- ★ One Thread Per Request Approach: Kanishk, Shaurya
- ★ Web Client: Mohit, Kanishk, Shaurya
- ★ Pool of Threads with FCFS Scheduling: Prakash, Kavita
- ★ SFF Scheduling: Prakash, Kavita
- ★ Security: Kanishk, Shaurya
- ★ Apache Benchmarking: Mohit

References

- https://github.com/delta-ng/Concurrent-Webserver
- https://github.com/remzi-arpacidusseau/ostep-projects/tree/master/concurrency-webserver
- https://www.petefreitag.com/item/689.cfm
- https://www.tutorialspoint.com/apache_bench/index.htm