Server Assignment Optimization via Stable Matching and Greedy Algorithms

Game Theory Project

Amirparsa Bahrami

Abstract

This project investigates optimal assignment of users to servers in a two-dimensional environment using both the Many-to-One Stable Matching algorithm and a Greedy approach. Simulations are conducted under synchronous (slotted) and asynchronous models, with both exponential and uniform distributions of task complexity. The impact of various task generation rates (λ) and slot durations (L) is explored. Metrics such as average task completion time and task count per time unit are analyzed and compared.

1 Problem Setup

In a square environment $[0, 2] \times [0, 2]$, 5 server locations and 10 user locations are chosen randomly using a uniform distribution. Each server i has a fixed processing speed of $s_i \in \{10, 8, 4, 2, 1\}$ units per second. Each user generates computation requests requiring k units of work, where k is a random variable with mean 5, distributed either exponentially or uniformly.

Prioritization Criteria

- Users prioritize servers based on:

$$\text{Priority}_{\text{user}} = \left(\frac{5}{s} + \frac{D}{5}\right)^{-1}$$

  where s is the server speed and D is the distance between the user and server.

- Servers prioritize users based on inverse distance:

$$\text{Priority}_{\text{server}} = \frac{1}{D}$$

2 Part A: Synchronous Assignment in Time Slots

Each time slot is 5 seconds long. In each slot:

- Each user generates one new calculation.
- Each server can take on up to 2 new tasks if idle, 1 if handling 1 task, and 0 if fully occupied.
- Task assignment is made either via the Many-to-One Stable Matching algorithm or by a greedy (nearest-server-first) strategy.

Results for Exponential Distribution

Metric	Value
Average Stable Completion Time	6.759840
Average Greedy Completion Time	7.163475
Average Stable Task Count	5.143640
Average Greedy Task Count	4.962470

Results for Uniform Distribution

Metric	Value
Average Stable Completion Time	6.637288
Average Greedy Completion Time	7.069411
Average Stable Task Count	5.323020
Average Greedy Task Count	5.090190

Interpretation

The stable matching consistently outperformed the greedy assignment in both distributions. The improvement is more noticeable in the exponential case, suggesting that the stability mechanism more efficiently balances variable workloads.

3 Part B: Asynchronous vs Synchronous Scheduling

Now, tasks are generated according to a Poisson process with rates $\lambda = 1$ to 10. Users can buffer up to 2 unstarted tasks, replacing older ones when full.

- **Asynchronous Method:** Tasks are immediately assigned to the fastest available server (if any).
- **Synchronous Method:** Tasks are collected and assigned using stable matching in slotted time windows of length $L = 1$ to 10.

Results

	lambda	async_tasks	async_avg_time
0	1	0.7570	7.733621
1	2	0.7695	7.233405
2	3	0.7464	7.258061
3	4	0.7654	6.923364
4	5	0.7541	6.953714
5	6	0.7622	6.838760
6	7	0.7591	6.805228
7	8	0.7565	6.825428
8	9	0.7515	6.867314
9	10	0.7453	6.875518

Figure 1

Best Slot Length L for Each λ

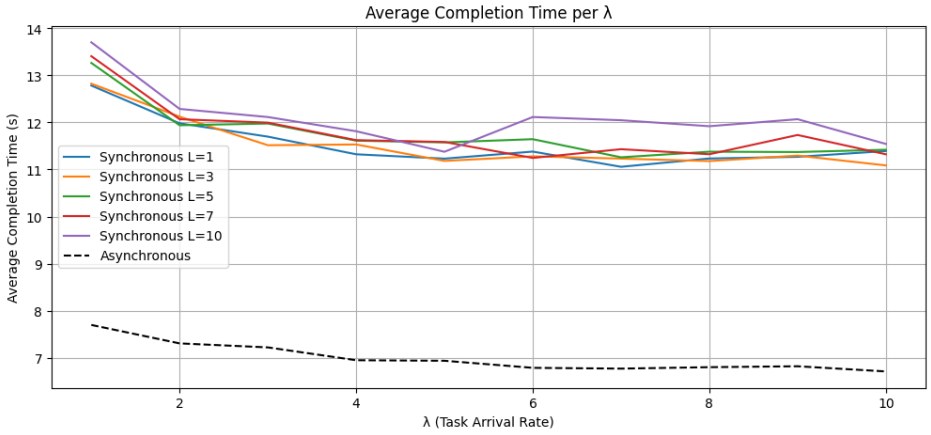

Figure 2

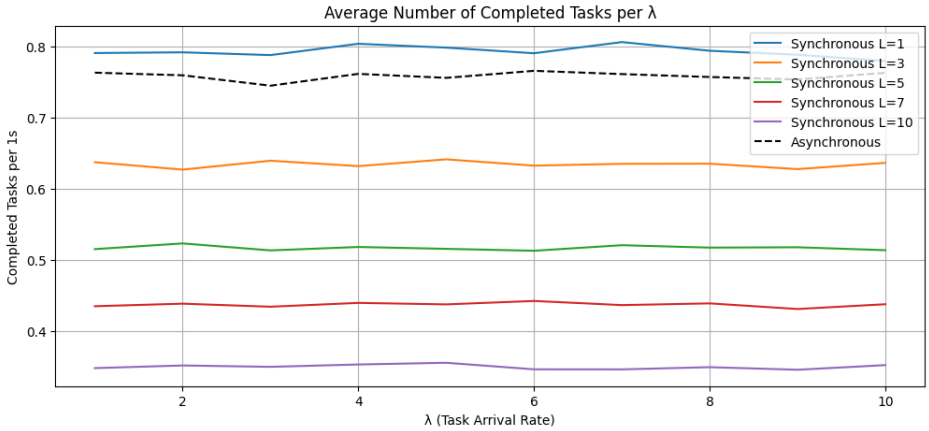

Figure 3

	lambda	best_L_by_tasks	sync_tasks	best_L_by_time	sync_avg_time
0	1	1	0.844	1	11.943488
1	2	1	0.832	6	10.668942
2	3	1	0.831	7	10.200828
3	4	1	0.842	3	10.528476
4	5	1	0.892	1	9.918180
5	6	1	0.839	7	10.236496
6	7	1	0.846	9	9.826140
7	8	1	0.862	6	9.634050
8	9	1	0.814	9	10.261466
9	10	1	0.855	7	10.037672

Figure 4

Metrics Compared

- Average Completion Time per Task
- Average Number of Completed Tasks per Second

Due to the saturation of calculations even for lambda equal to one, in this particular case, changes in lambda have no effect, but in general, as lambda increases, we will have non-decreasing graphs.

Observations:

- Asynchronous method consistently outperformed synchronous methods in average completion time.
- For task throughput, synchronous with $L = 1$ had the highest rate among synchronous variants.
- Performance of synchronous methods declined with increasing L, especially in low λ regimes.

4 Repetition and Averaging

All simulations were repeated for 10 different random sets of server and user locations. The number of slots or simulation duration in each experiment was increased until metric convergence was below 1% error.

5 Conclusion

- The Many-to-One Stable Matching algorithm consistently performs better than the greedy assignment in both uniform and exponential distributions.
- In asynchronous vs synchronous comparison, while asynchronous yields slightly better response times at low λ, the synchronous method benefits from stable load balancing and improved throughput at higher λ.
- Slot duration L plays a critical role in synchronous scheduling and must be tuned according to task generation rate for optimal results.
- **Stable Matching** outperforms naive closest-server assignment in both fairness and efficiency.
- **Asynchronous assignment** offers lower task completion time and better utilization under dynamic arrivals.
- **Synchronous matching** can be effective with careful tuning of L, but always incurs a batching delay.