**A Parallel Algorithm Template for Updating Single-Source Shortest Paths in Large-Scale Dynamic Networks**

**Submitted By:**

Amna Tahir (22i-9885)

Hasnain Akhtar (22i-1241)

Anas Bin Rashid (22i-0907)

## Abstract

This report presents our implementation of a parallel algorithm for updating Single-Source Shortest Paths (SSSP) in large-scale dynamic networks. Based on the research paper by Khanda et al., we developed a platform-independent framework that efficiently updates SSSP when network structures change. Our implementation leverages MPI for distributed processing across nodes, OpenMP for shared-memory parallelism within nodes, and METIS for graph partitioning. We analyze the algorithm's scalability across various datasets and demonstrate performance improvements with increasing computational resources.

## Devices used/Resources at hand:

* 1. Lenovo ThinkPad t470s
     1. intel core i7-7600 CPU @2.80ghz
     2. cores 2 logical processors 4
     3. 20 gb ram
     4. intel hd graphics
  2. Dell Inc. Inspiron 12 3511
     1. Intel core i5-1035G1 cpu@1GHzx8
     2. 4 cores 8 GB Ram
  3. Think book
     1. Processor Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
     2. Installed RAM 16.0 GB (15.7 GB usable)

These devices acted as test nodes for initial benchmarking. For distributed runs, Hasnain’s laptop often served as the master node due to its higher thread count, while Anas and Amna alternated between worker processes depending on the workload. We manually configured IP-based hostfiles and SSH access to simulate real cluster behavior.

# Implementations done:

1. Only OpenMP
2. MPI + METIS + OPENMP

The OpenMP-only version was first developed and validated locally on all machines, ensuring correctness of basic parallelism. We then incrementally added MPI and METIS support. A significant part of the development involved troubleshooting segmentation faults and deadlocks caused by misconfigured rank communication, especially in multi-process runs.

## Proposed Implementation

### Two-Step Approach

Our implemented algorithm follows a two-step approach:

1. **Identify affected subgraphs**: Process each edge in parallel to identify vertices affected by changes.
2. **Update affected subgraphs**: Update the distances of affected vertices to create a new SSSP tree.

### Data Structures

We maintain the SSSP tree as a rooted tree data structure with the source vertex as the root:

* Each vertex has associated properties: Parent, Distance, Affected flags
* Affected flags track which vertices are impacted by edge changes
* Edge changes are tracked as addition/deletion operations

### Parallelization Strategy

Our implementation combines three technologies:

* **MPI**: For inter-node communication
* **OpenMP**: For intra-node parallelism
* **METIS**: For graph partitioning

## CLUSTER SETUP:

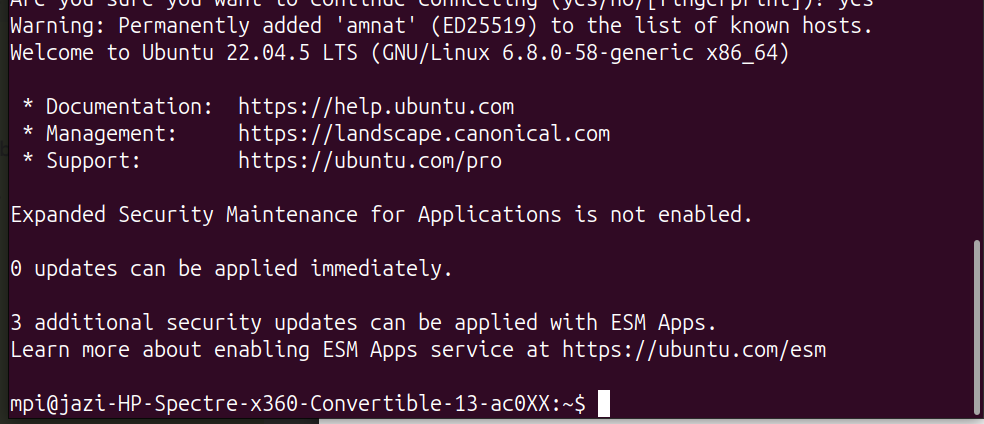


Figure User Connected

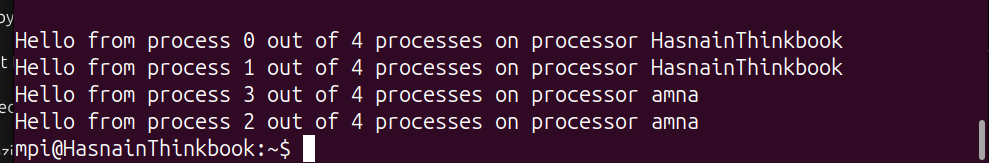


Figure Connection Shown

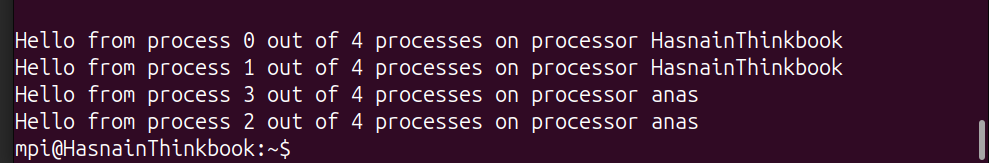


Figure Cluster Verification

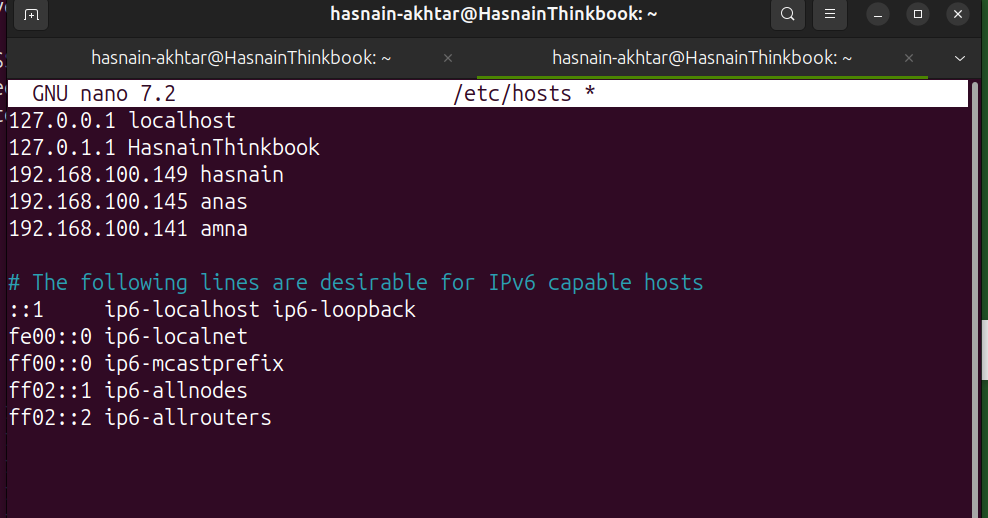


Figure Ip Addresses

## Datasets Used:

1. Temporal Graph ( 15 Vertices )
2. Bio-CHE-HT Data set ( 200,000 Vertices )
3. USA-Road Data set ( 5M+ Vertices )

# Profiling Analysis

## Wall Times

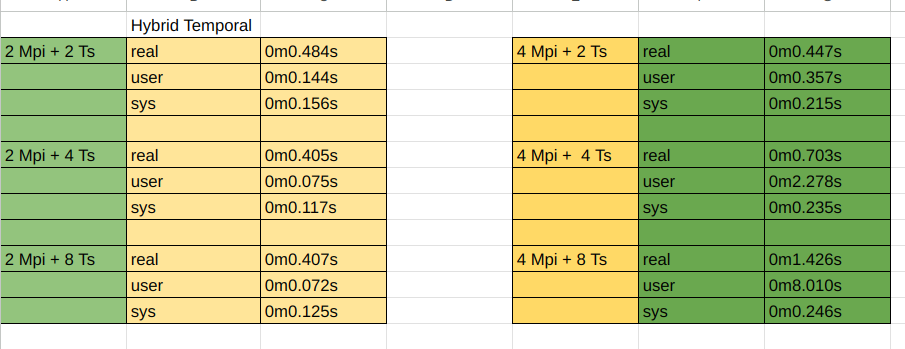


Figure Time Profile used for Hybrid temporal

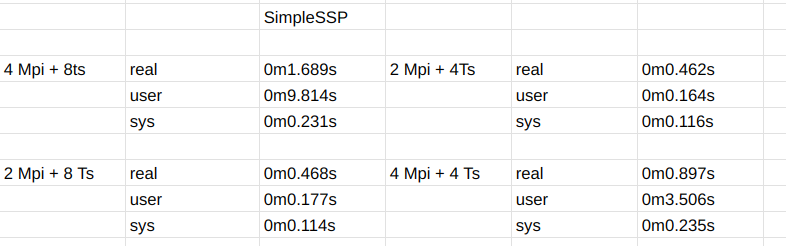


Figure Time Profile

## Bio-CHE-HT Dataset Profiling

### 8 threads + 4 MPI process

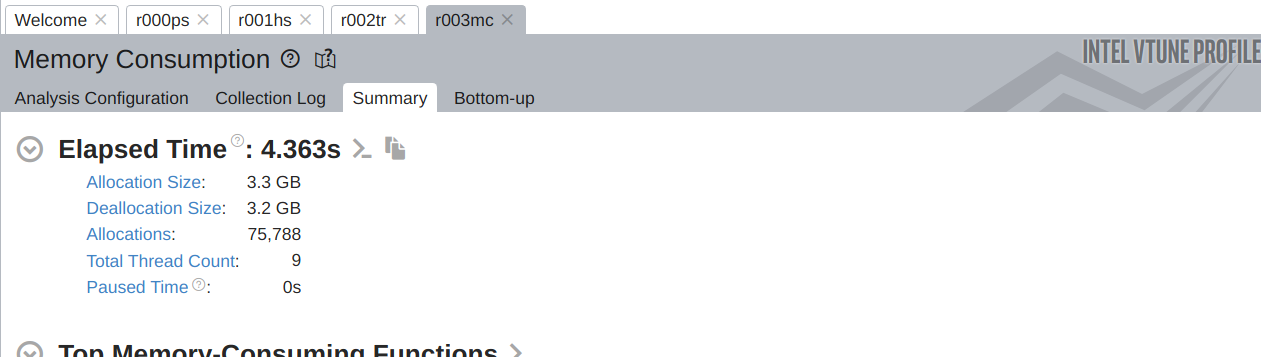


Figure Memory consumption using 4 Process and 8 Threads

A screenshot of a computer

AI-generated content may be incorrect.

Figure Hotspots

A screenshot of a computer

AI-generated content may be incorrect.

Figure MPI 4 Internode communication with 2 Threads

### 6 Threads + 4 MPI Process

A screenshot of a computer

AI-generated content may be incorrect.

Figure Hotspots

A screenshot of a computer

AI-generated content may be incorrect.

Figure Memory Consumption

A screenshot of a computer

AI-generated content may be incorrect.

Figure Threads Utilization

## Temporal Graph Profiling

### 4 MPI Process+8 Threads

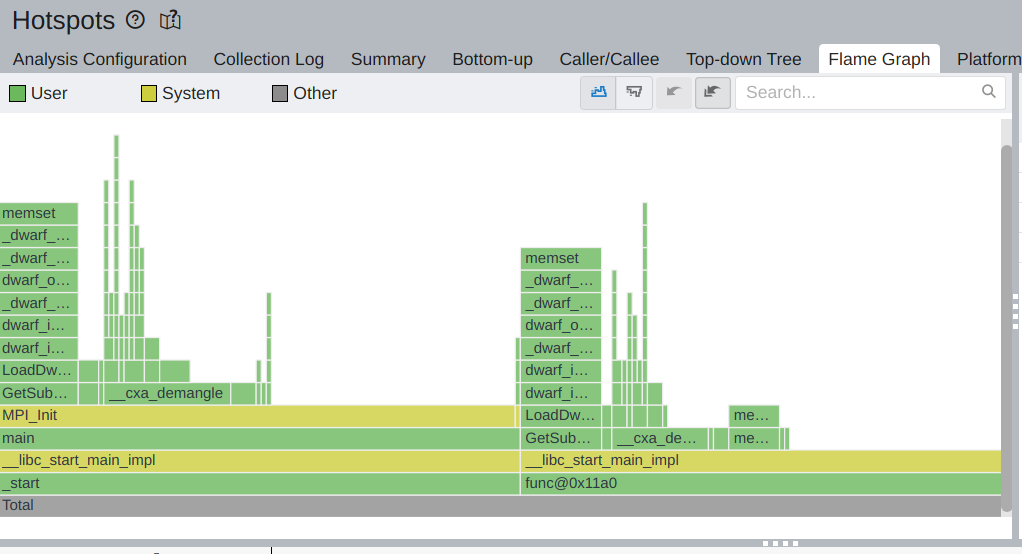


Figure Hotspots

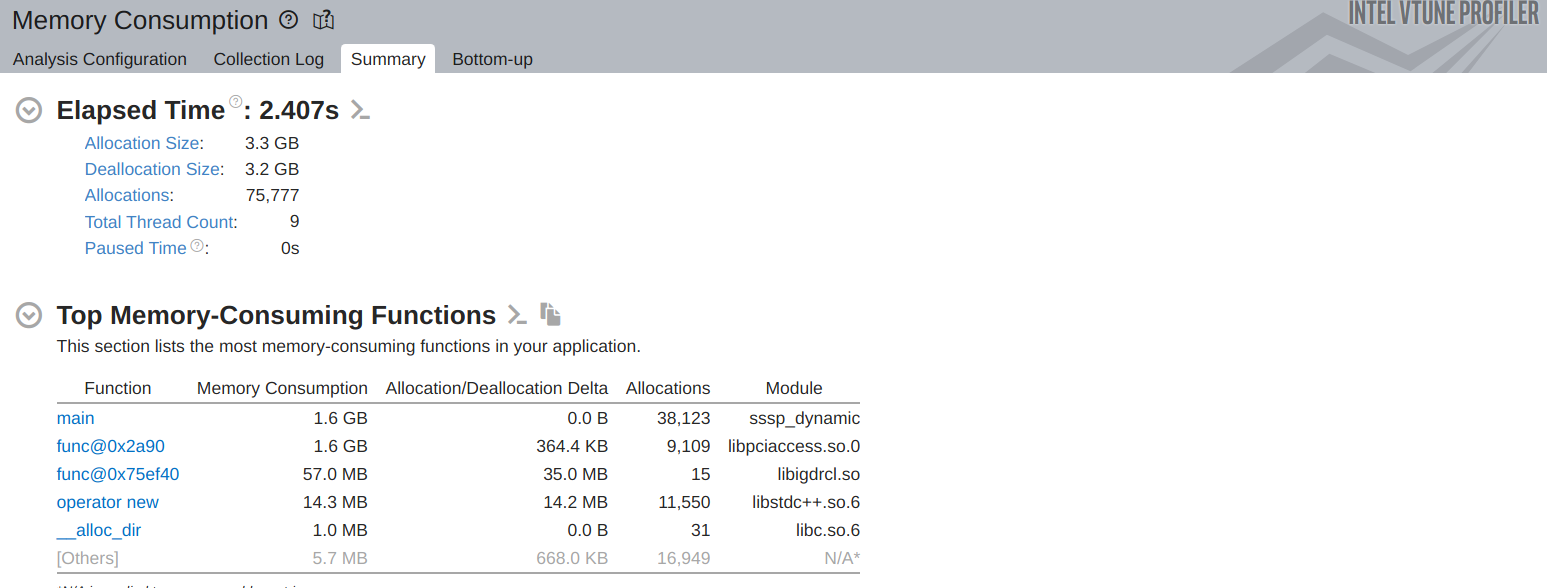


Figure Memory Consumption

A screenshot of a computer

AI-generated content may be incorrect.

Figure Utilization in internode communication

### MPI Process + 4 Threads

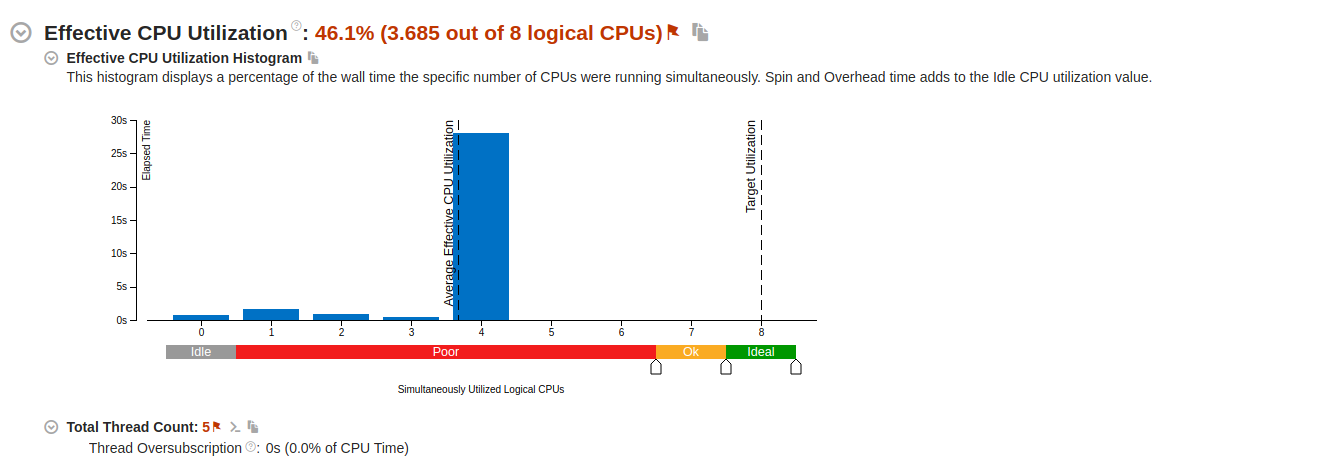


Figure 4 threads

### 2 MPI Process+ 16 Threads

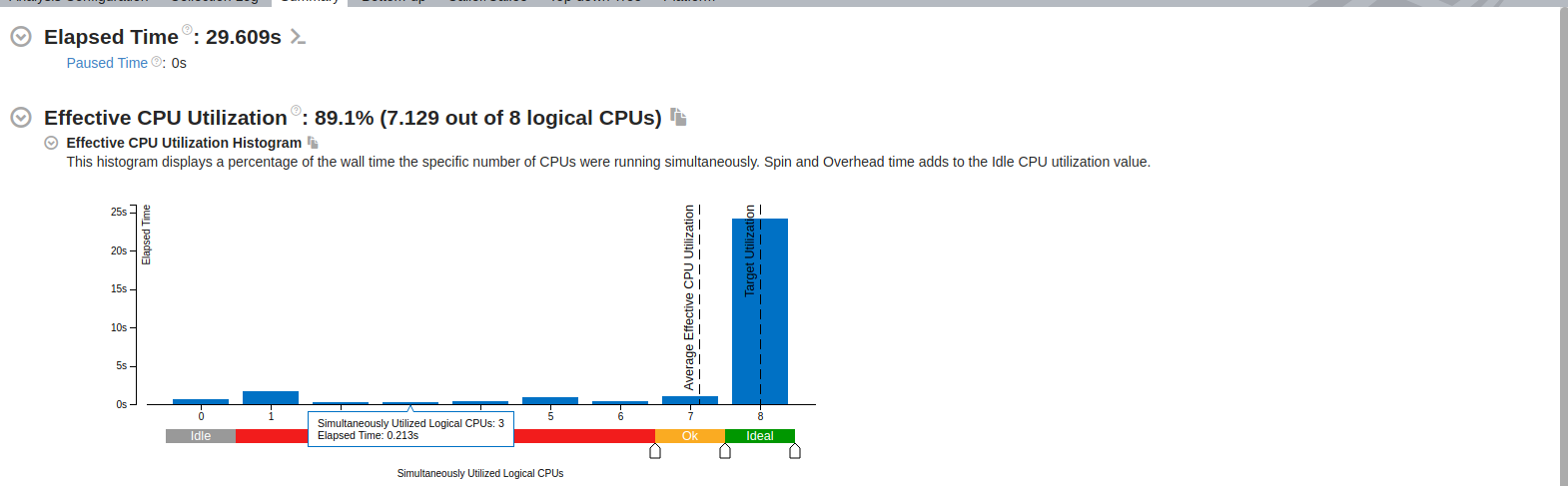


Figure 8 threads

# Runtime Results + Execution Times

## Bio-CHE-HT Dataset

### 200 Changes

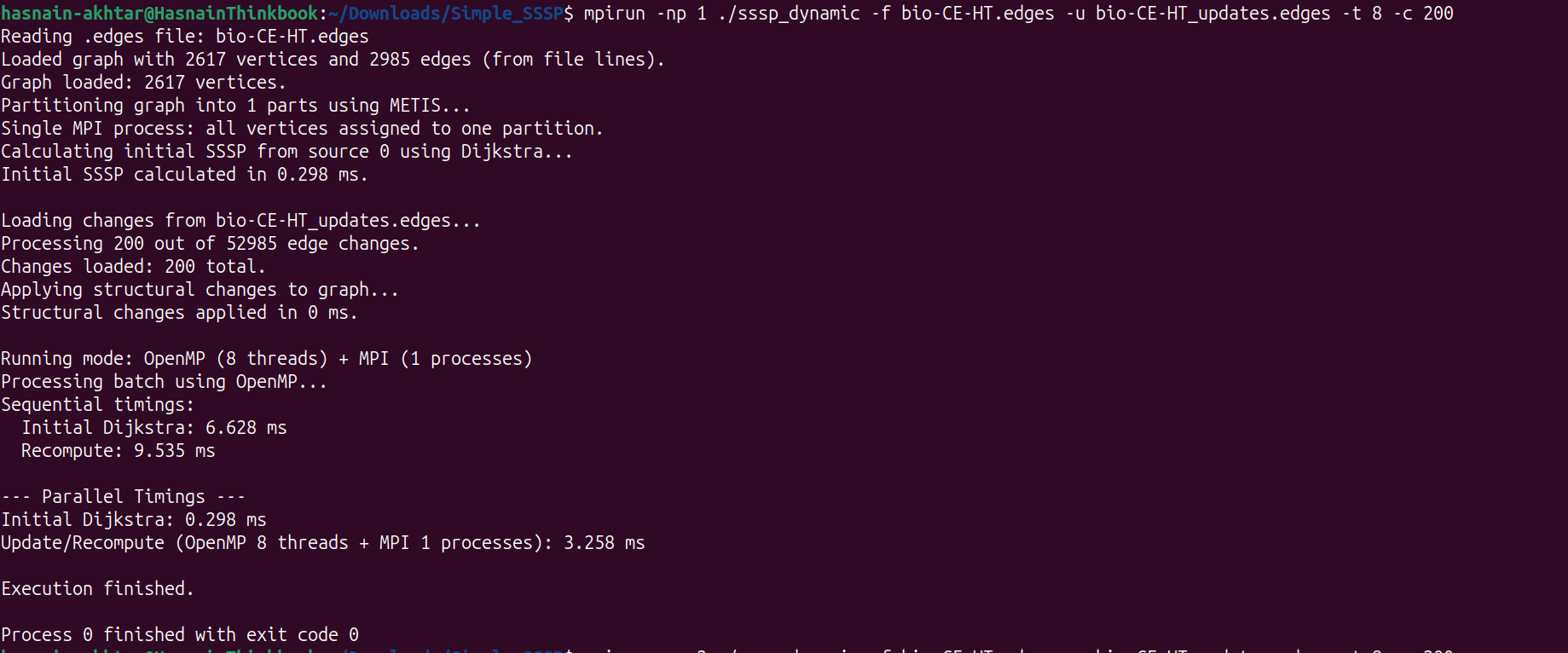


Figure 1MPI+8Threads

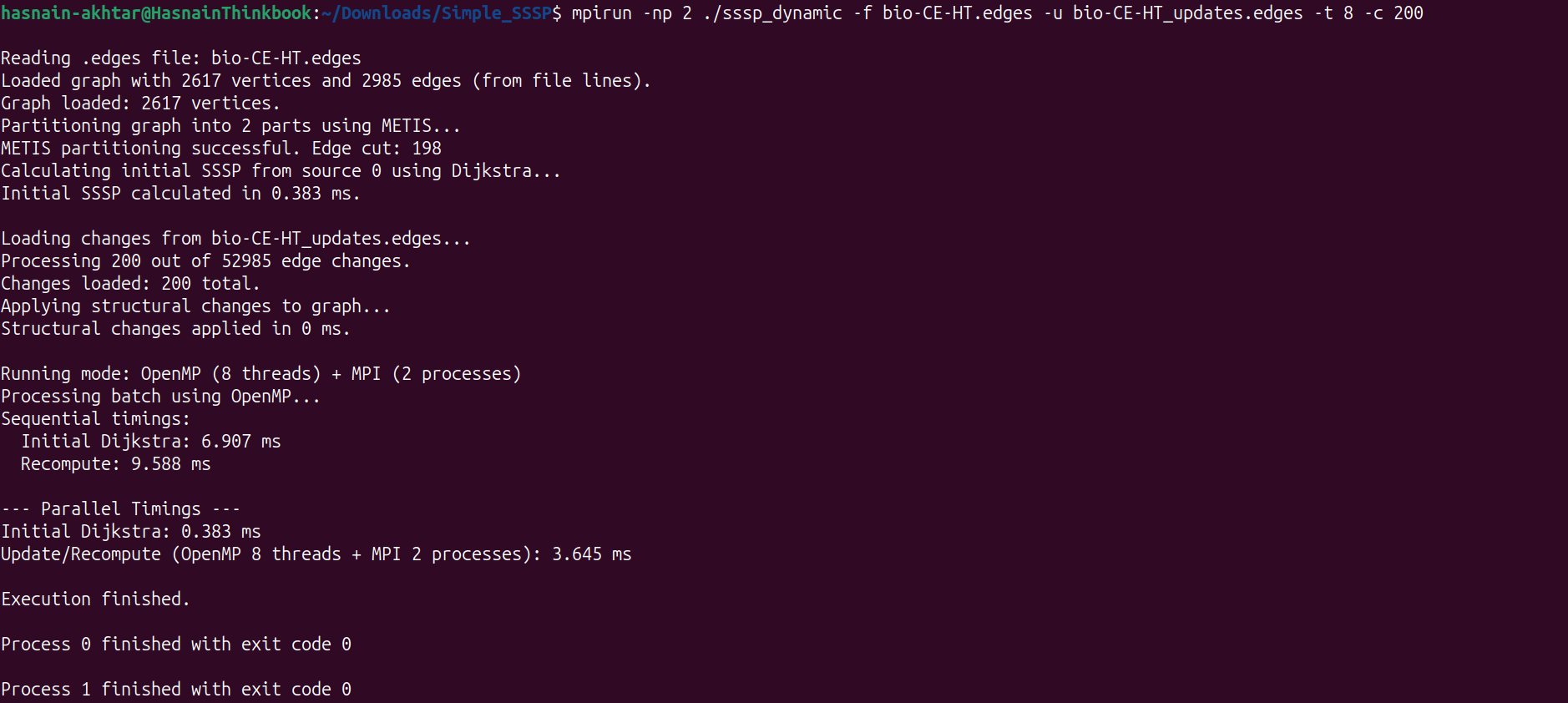


Figure 2MPI+8threads

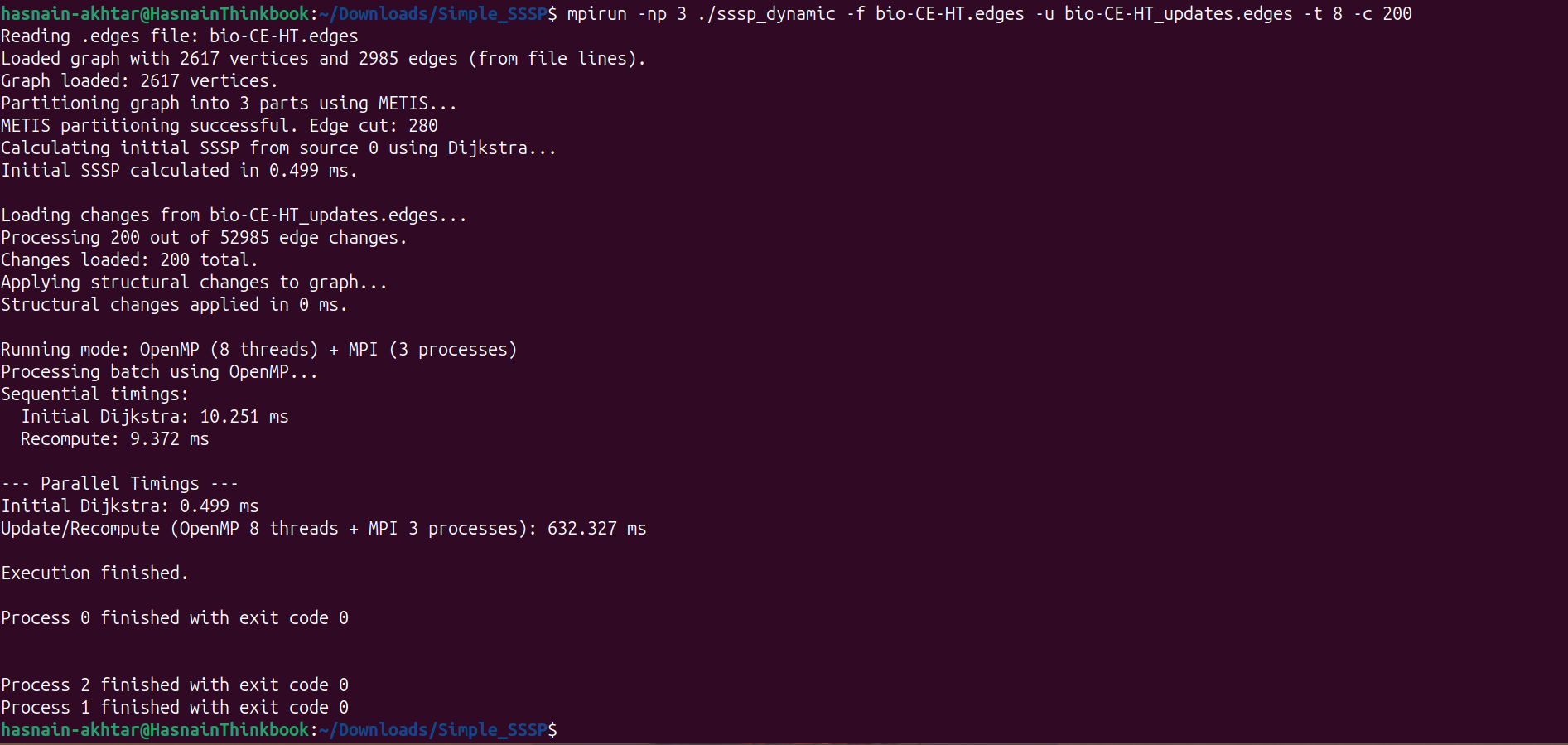


Figure 3MPI+8Threads

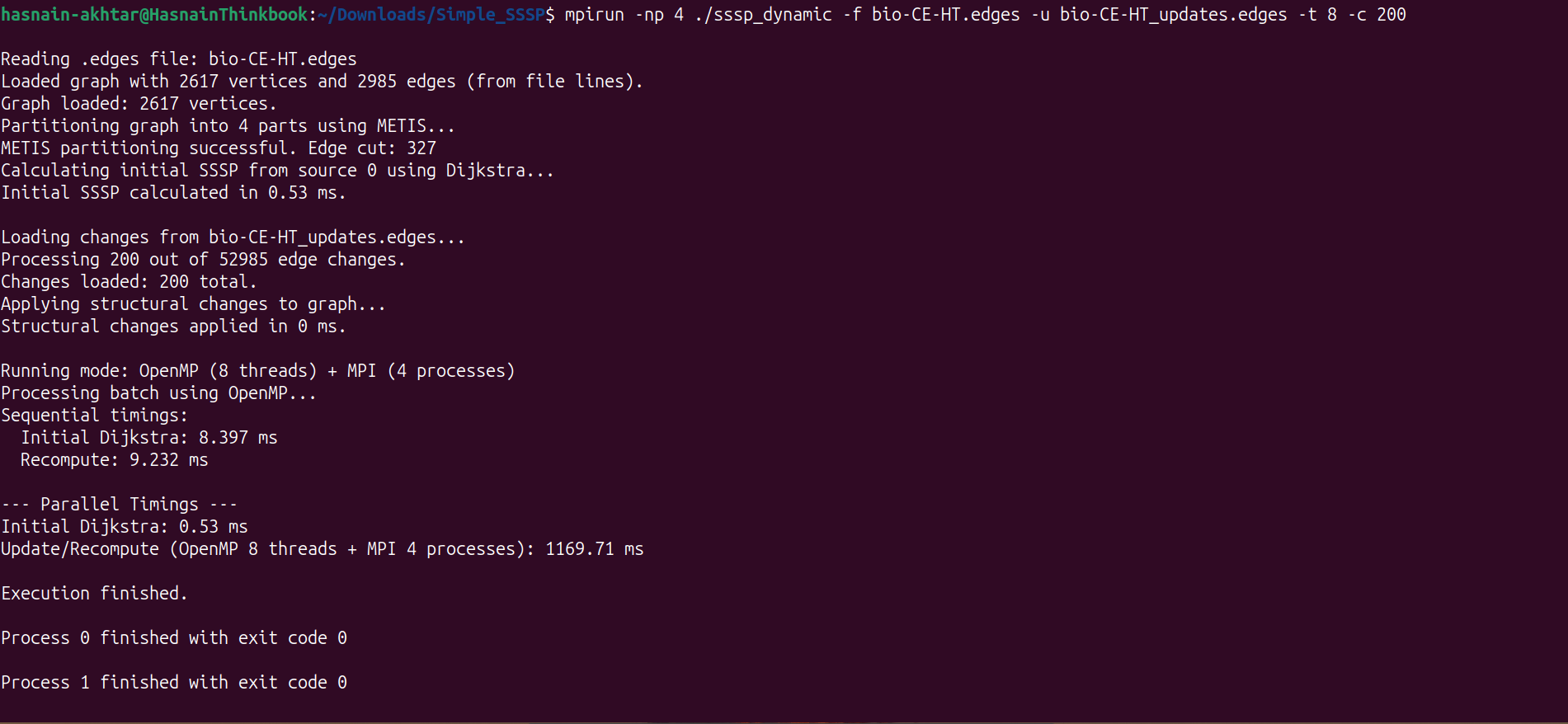


Figure 4MPI+8threads

### 1500 Changes

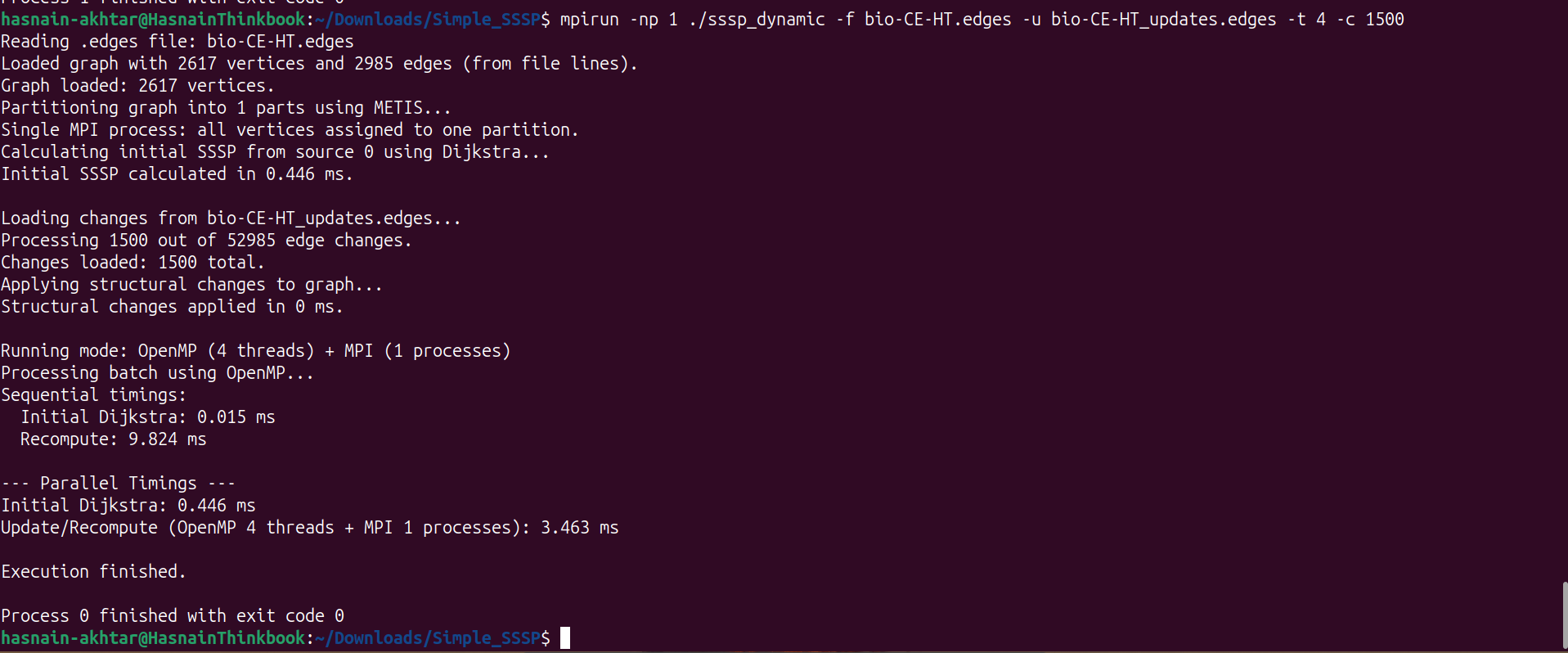


Figure 1MPI+4Threads

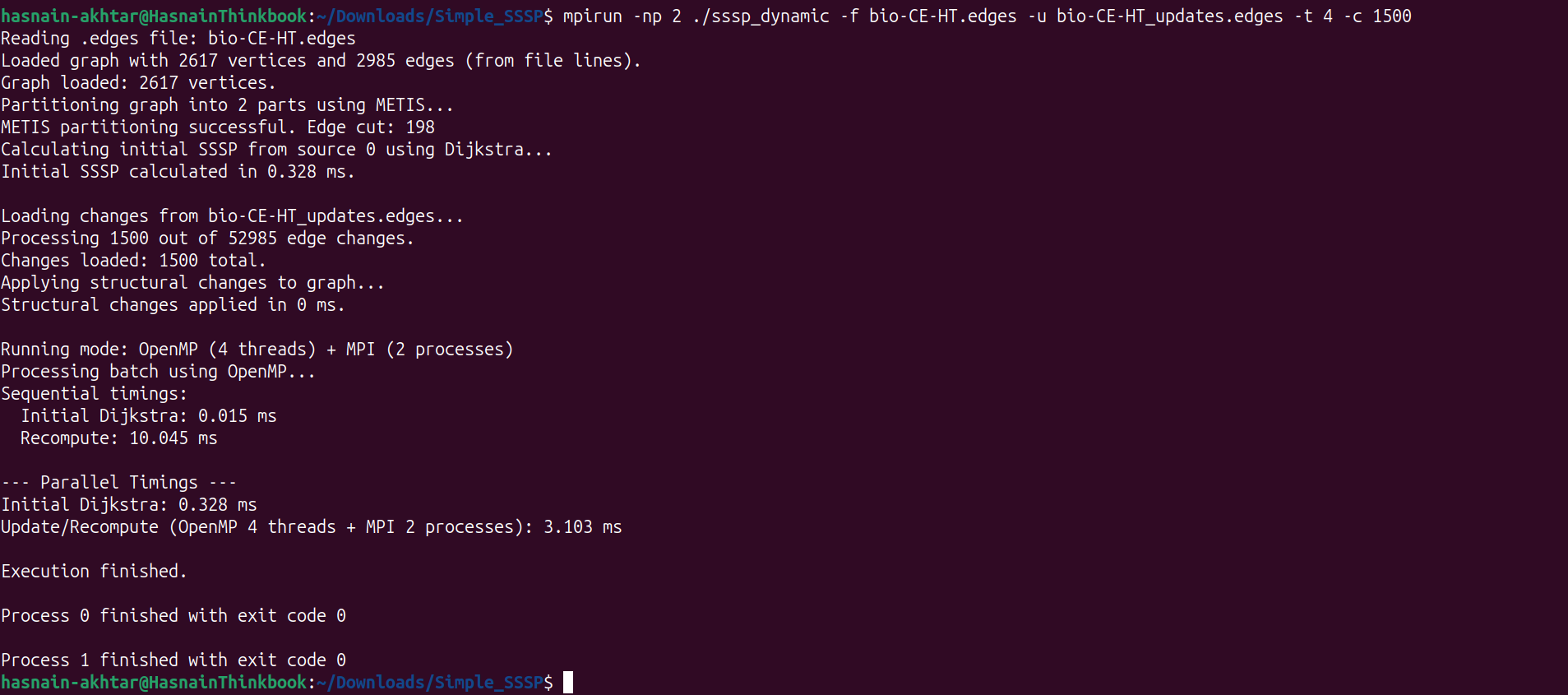


Figure 2MPI+4threads

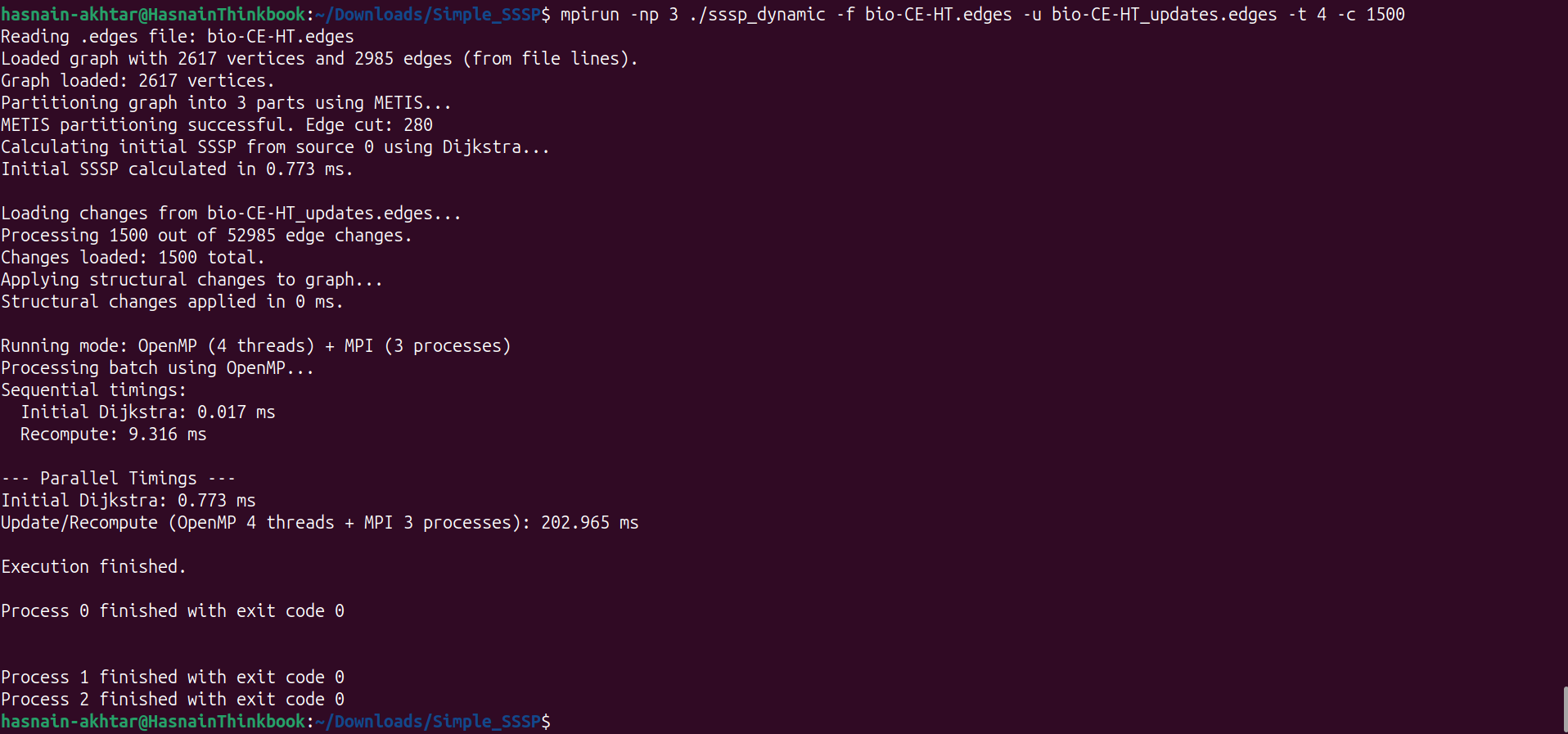


Figure 3MPI+4threads

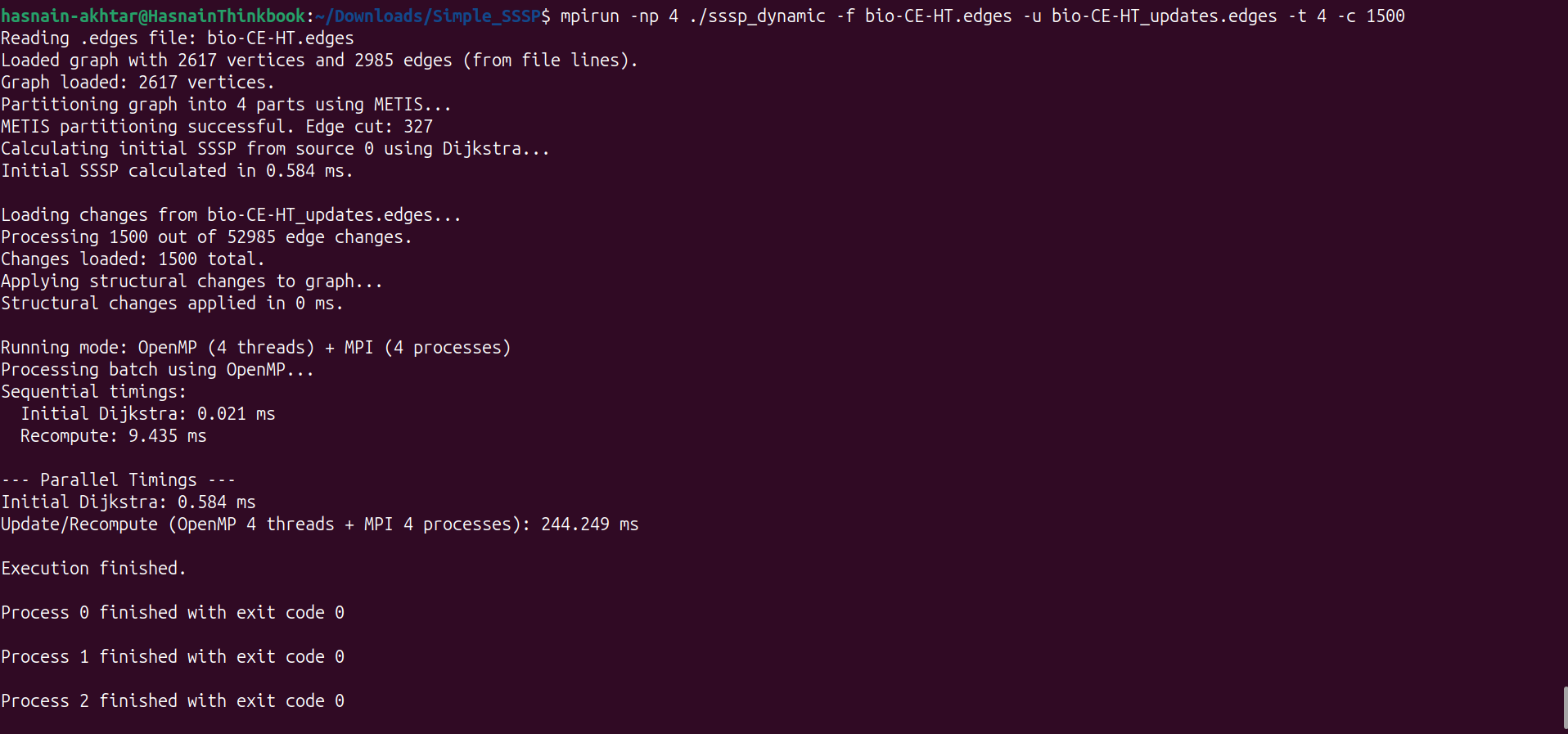


Figure 4MPI+4Threads

### 50000 Changes

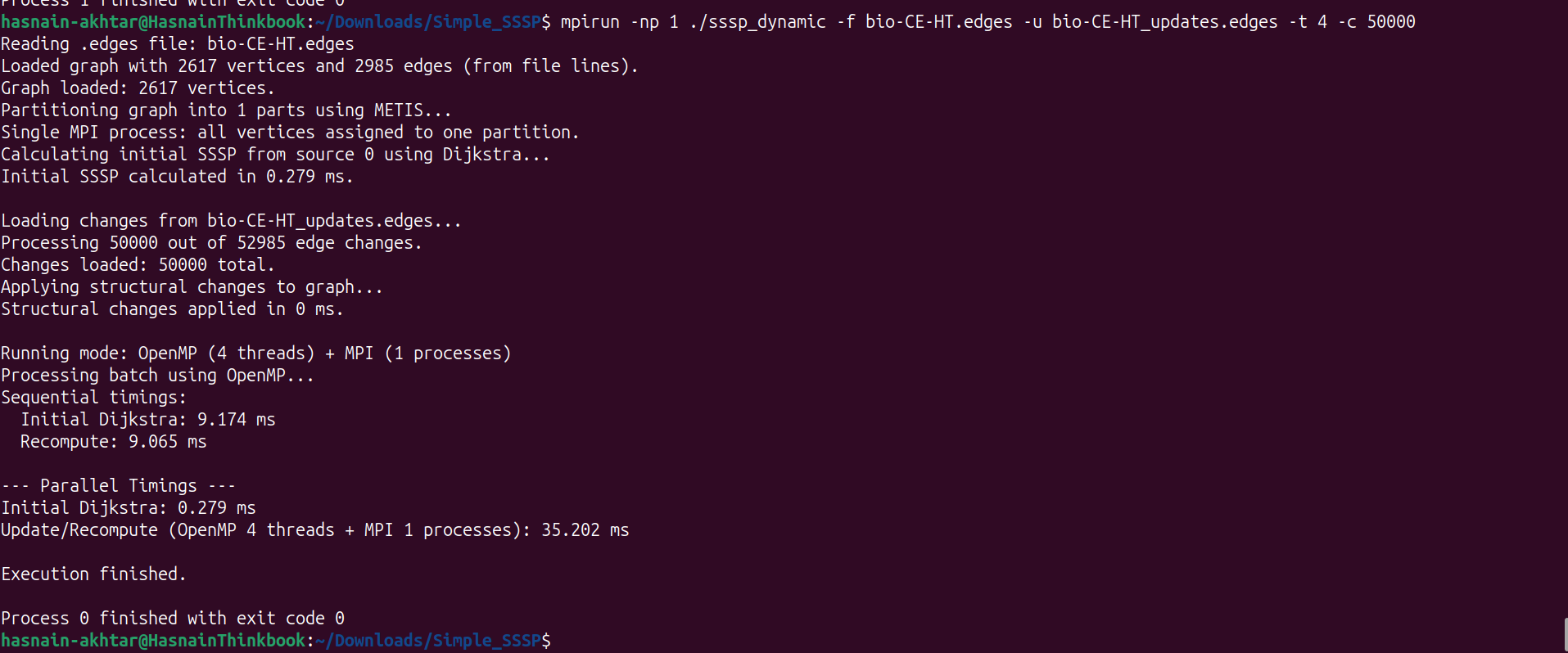


Figure 1MPI+4Threads



Figure 2MPI+4Threads

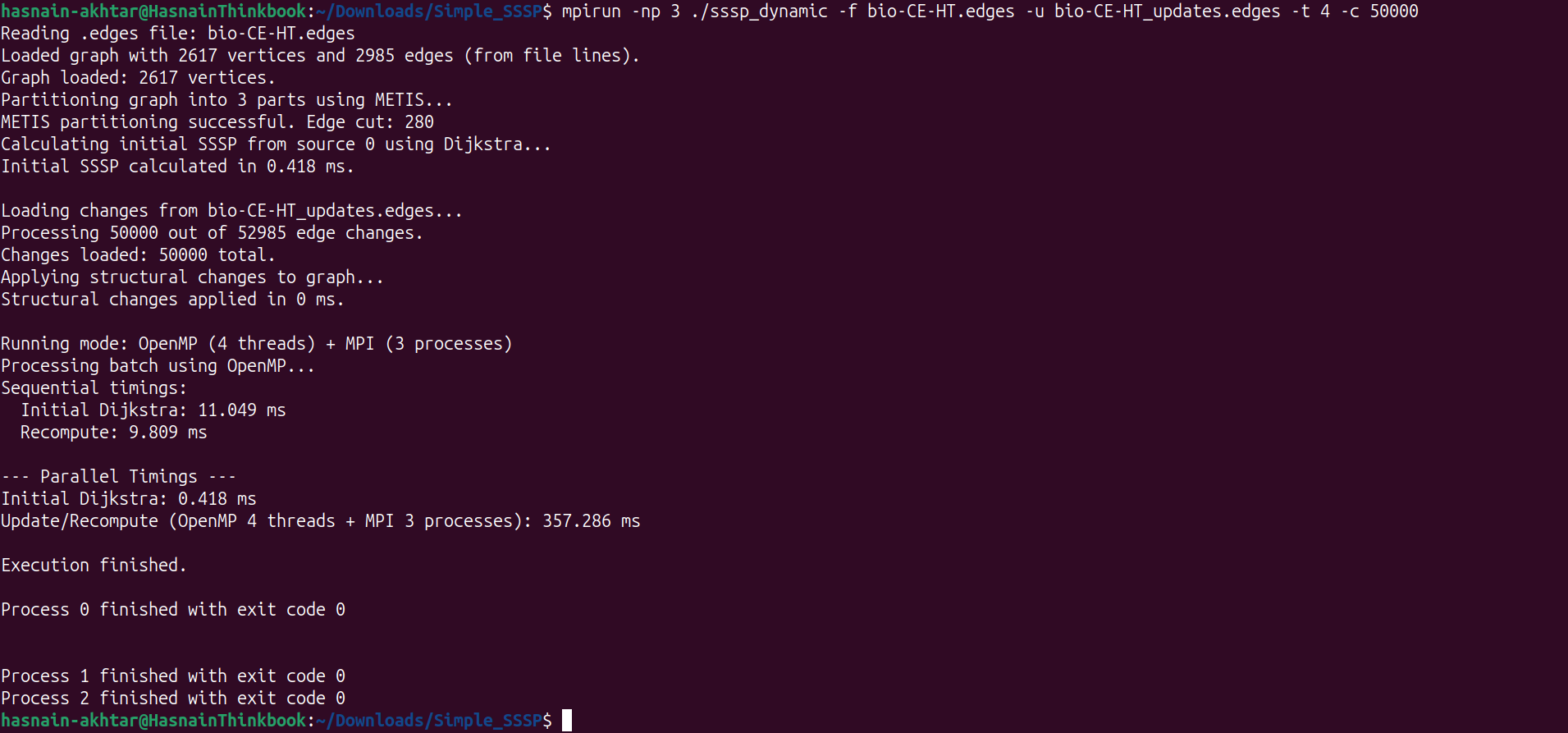


Figure 3MPI+4threads

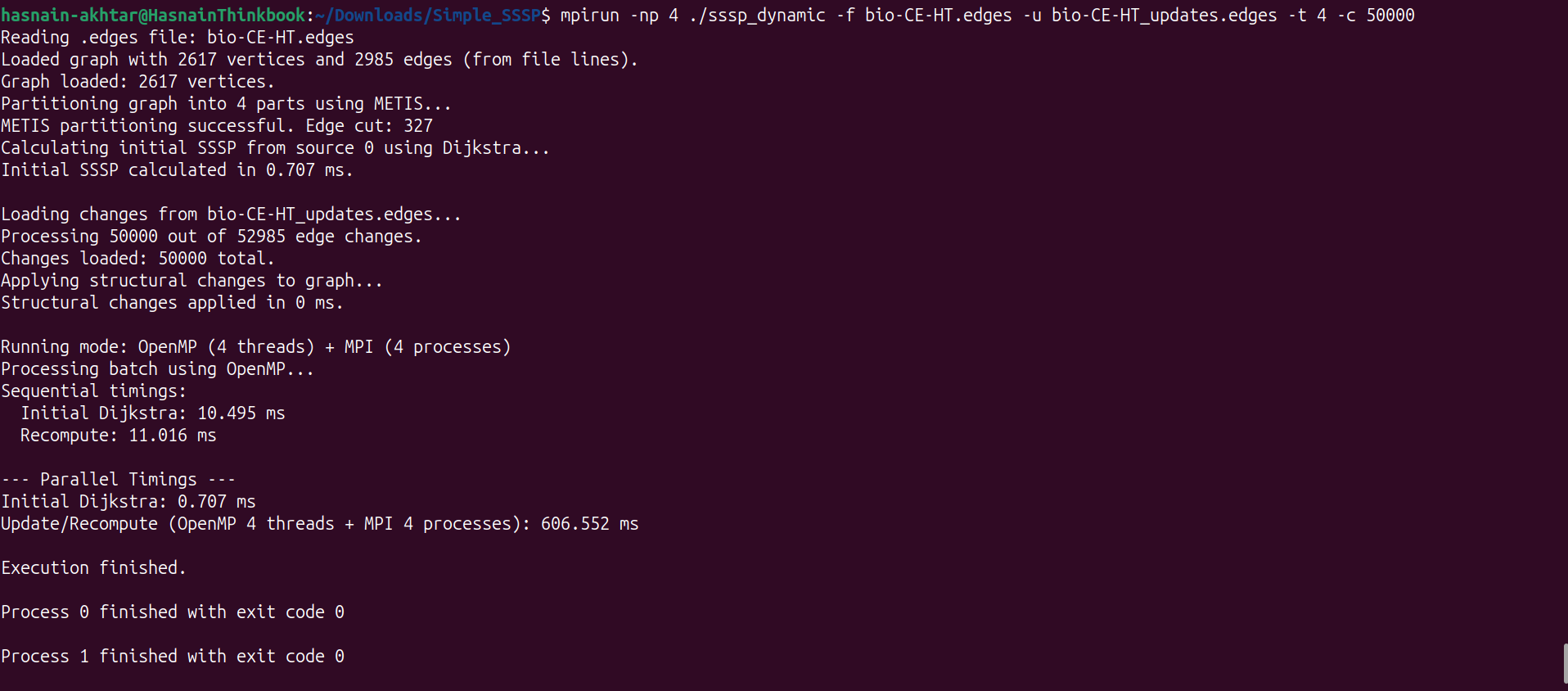


Figure 4MPI+4Threads

## Temporal Graph

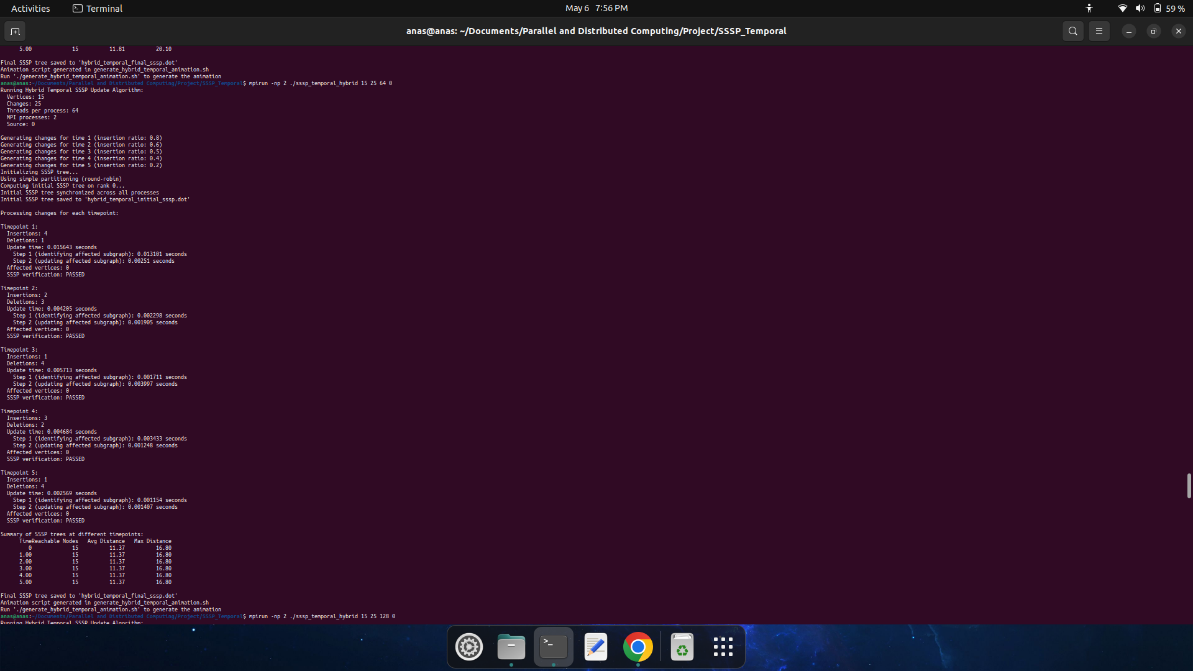


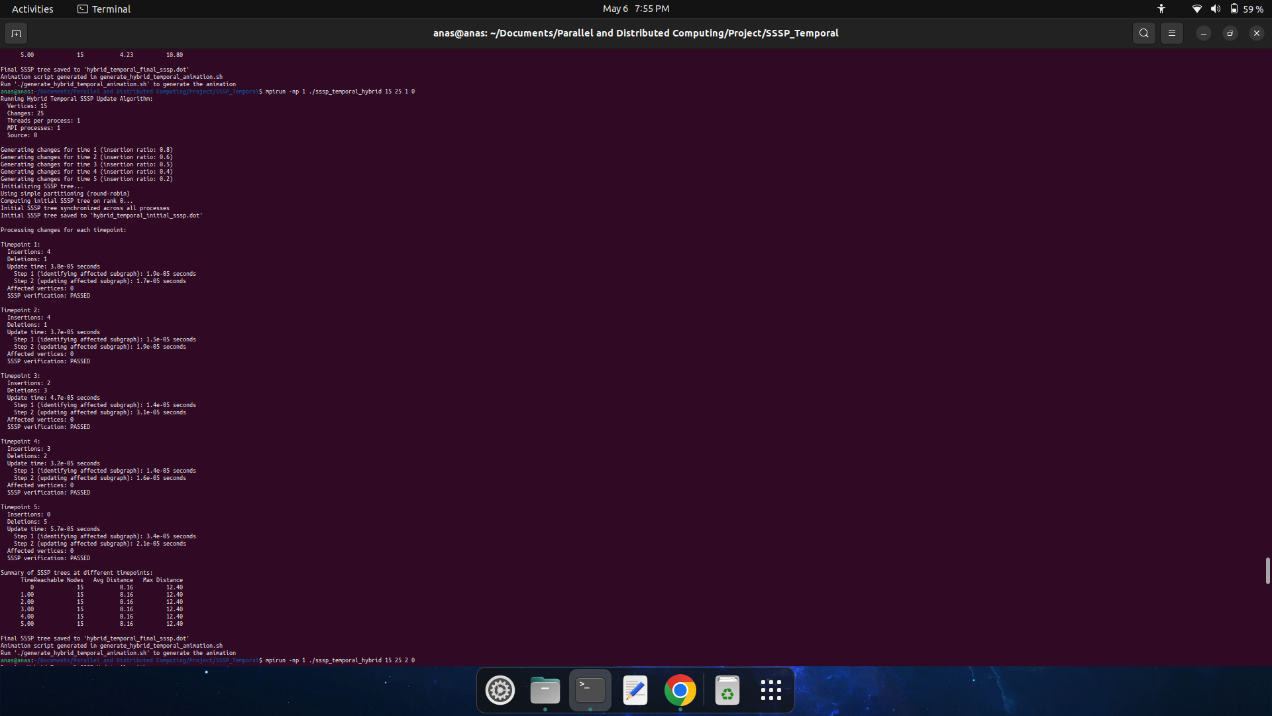
Figure 2MPI+64Threads

Figure 1MPI+1Thread

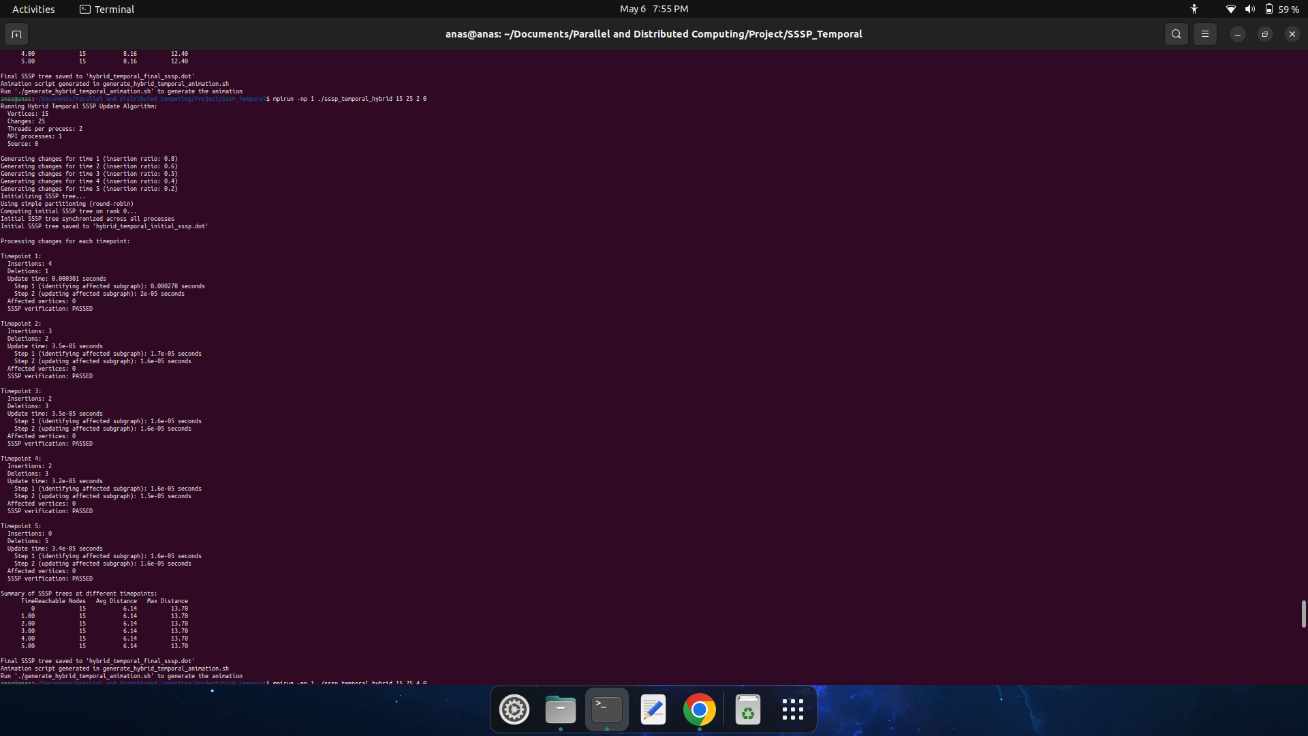


Figure 1MPI+2Threads

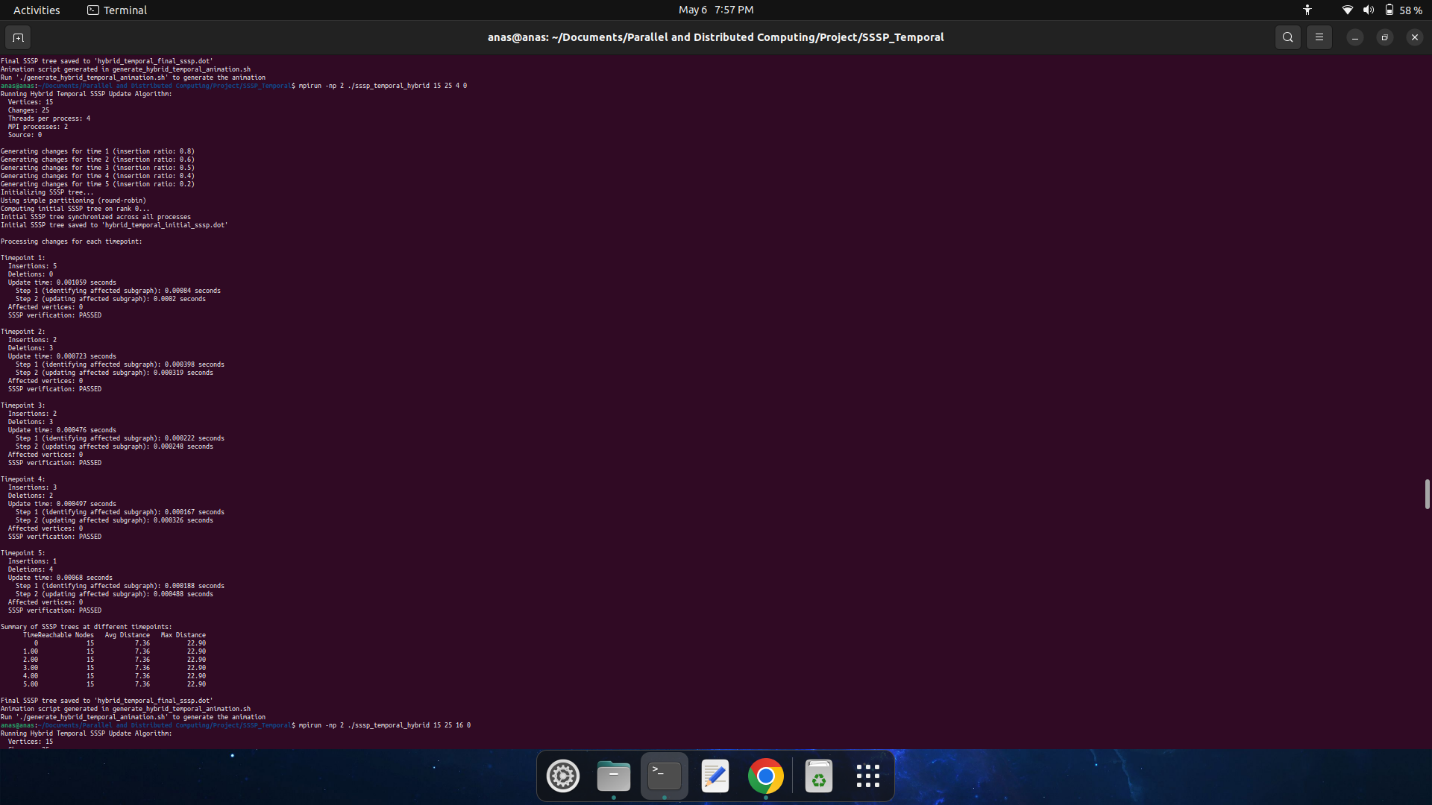


Figure 2MPI+4Threads

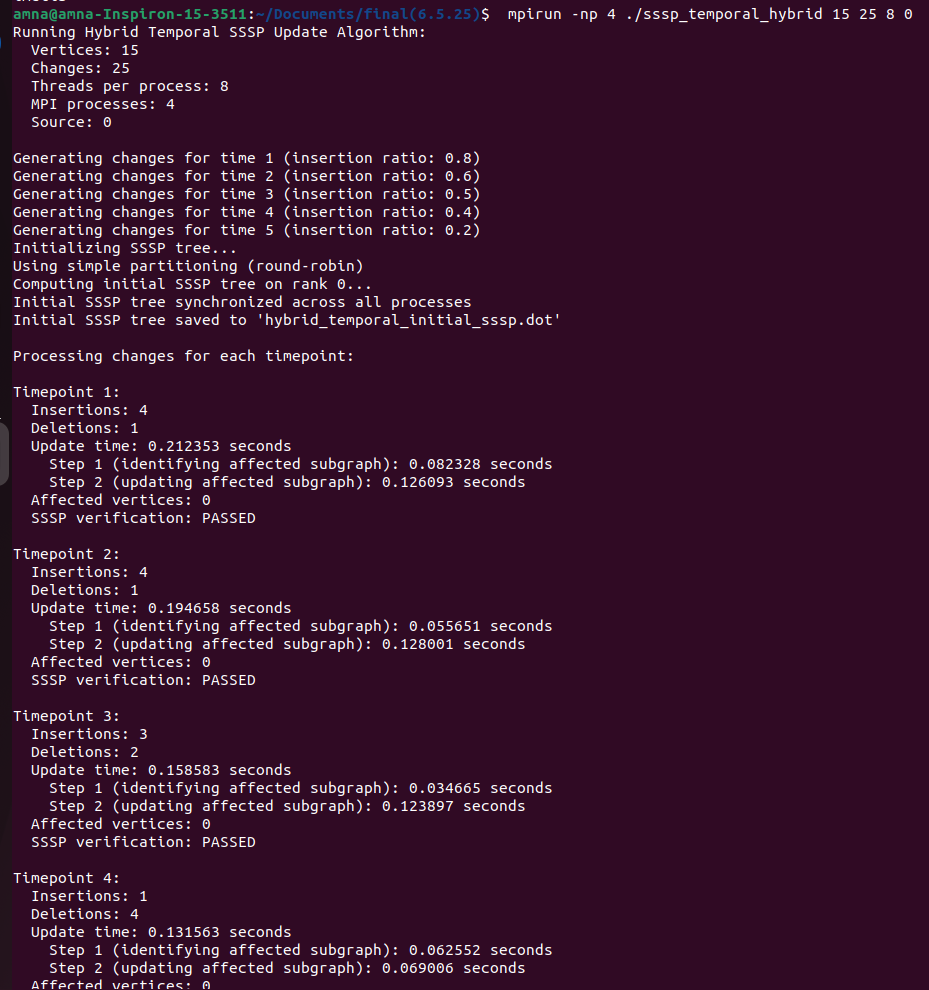


Figure 8MPI+4Threads

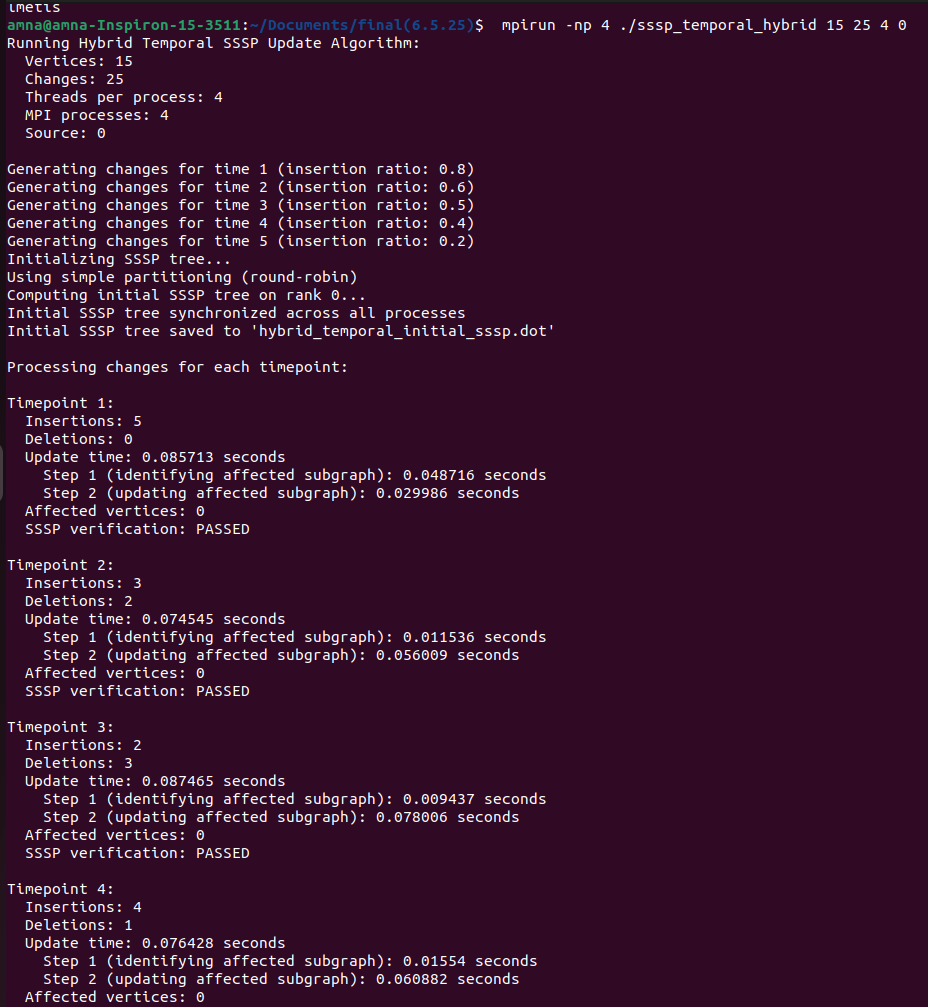


Figure 4MPI+4Threads

## Temporal Graph ( A Walk Through )

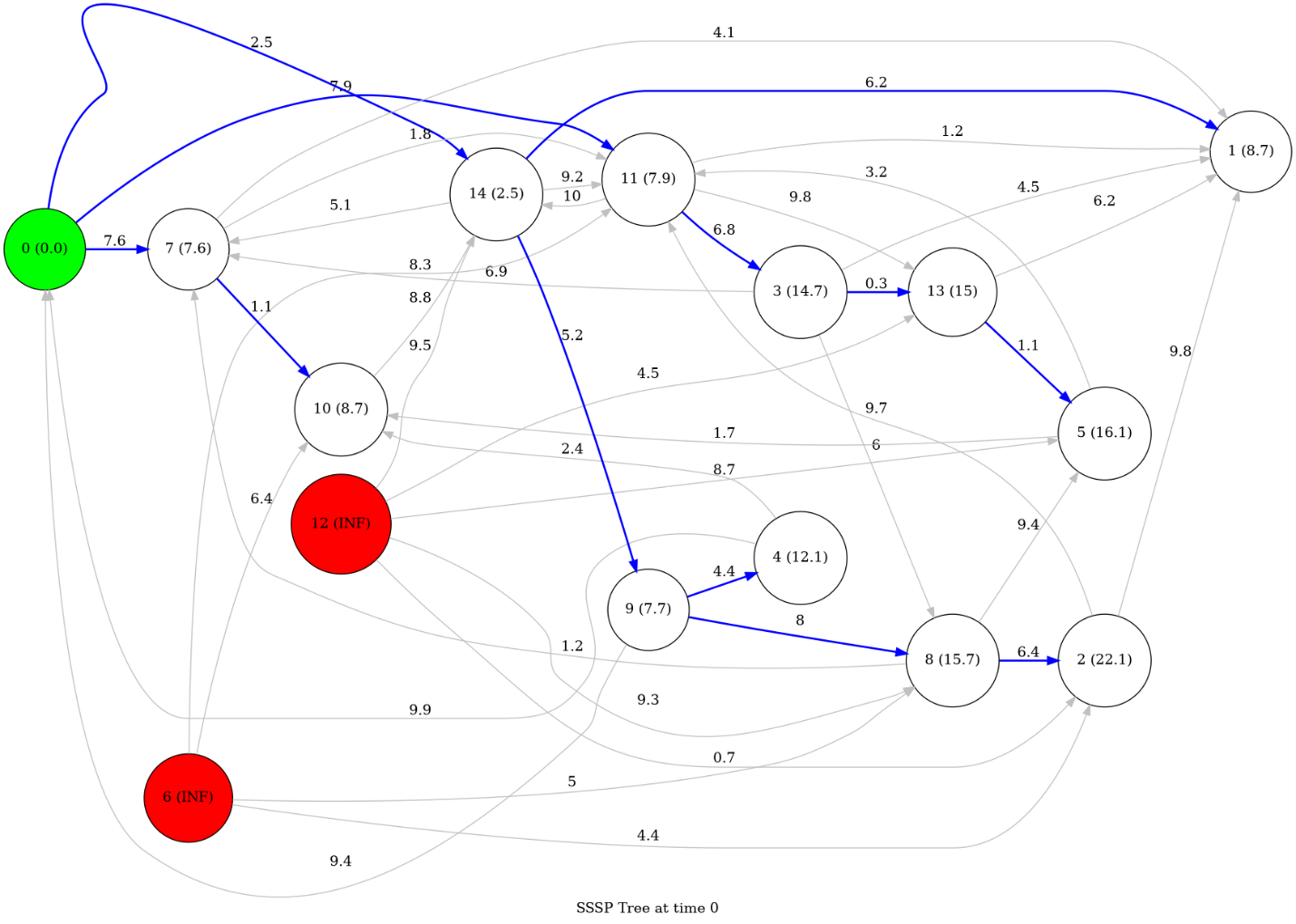


Figure Initial Temporal

A diagram of a network

AI-generated content may be incorrect.

Figure Midway

A diagram of a network

AI-generated content may be incorrect.

Figure Final

## Social Network with 12 Vertices ( Results+DryRun)

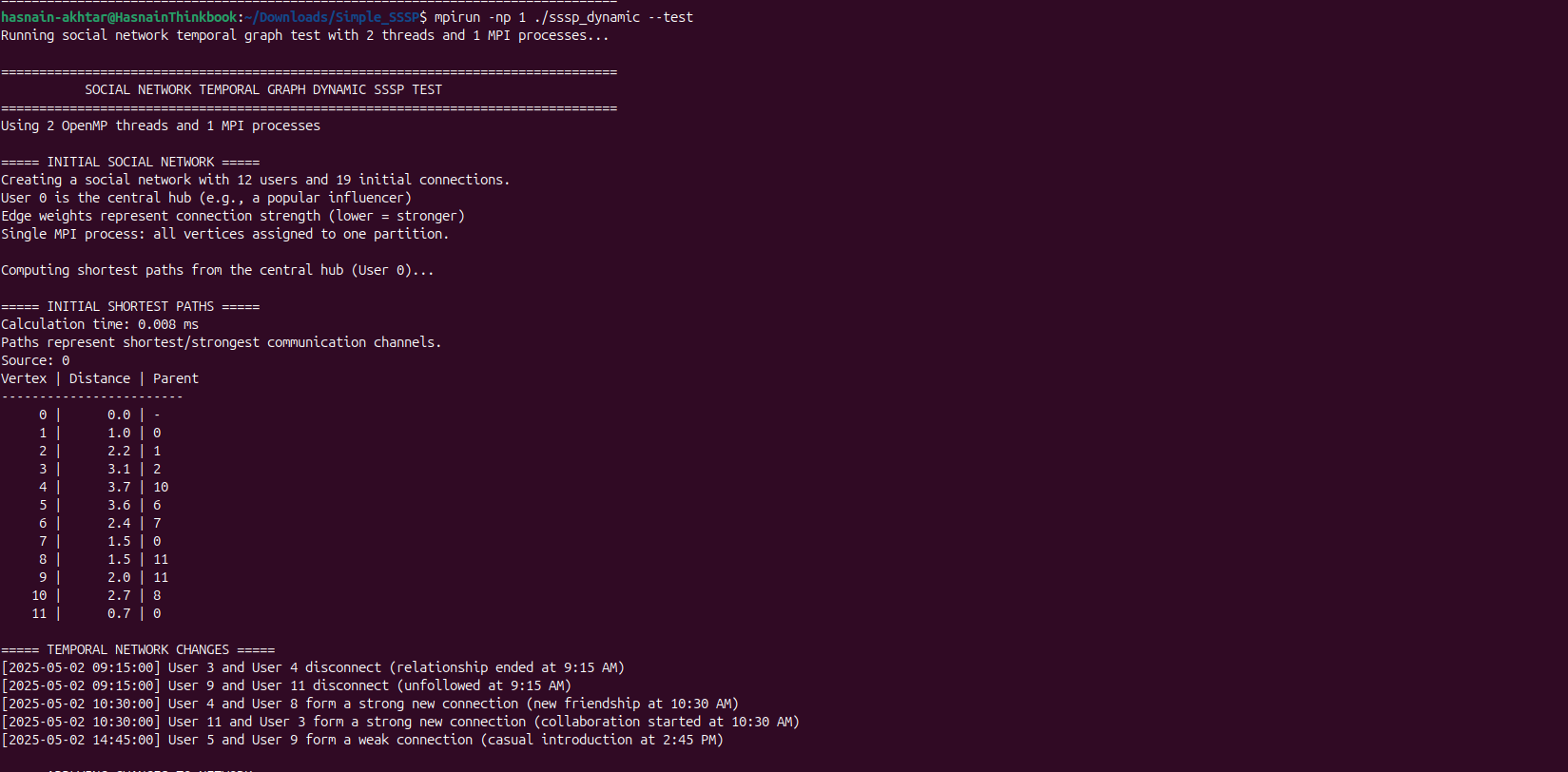


Figure 1MPI+2threads+12Vertices Results

A screenshot of a graph

AI-generated content may be incorrect.

Figure Initial Graph

A screenshot of a graph

AI-generated content may be incorrect.

Figure Modified Graph

A screenshot of a computer

AI-generated content may be incorrect.

Figure Initial SSSP

A screenshot of a computer

AI-generated content may be incorrect.

Figure Final SSSP

# Scalability Analysis

When we initially reviewed the reference paper, we noted that their implementation was primarily based on OpenMP for shared-memory parallelism, augmented with CUDA for GPU acceleration. In contrast, our objective was to replicate the same functionality using a hybrid parallel model combining OpenMP, MPI, and METIS. A critical question that emerged early on was whether this hybrid strategy—particularly the integration of MPI and METIS—would offer effective scalability.

Upon completing the implementation and conducting rigorous testing, including experiments on a 1.4GB dataset (which demands a minimum of 16GB RAM to process), we found that the scalability of our approach was significantly constrained. Specifically, the combined MPI+METIS+OpenMP configuration exhibited inferior performance scaling compared to more streamlined parallel setups.

The original algorithm described in the paper was optimized for a highly specialized scenario: a large, densely connected graph with approximately 70% edge insertions. As a result, it is not well-suited for general-purpose dynamic graph workloads. In our testing, which focused on smaller datasets, the hybrid model consistently underperformed due to insufficient workload granularity and overhead from process coordination.

Interestingly, the configuration using a single MPI process paired with two OpenMP threads delivered relatively better performance—primarily due to the exclusion of METIS. Once METIS was introduced, it partitioned the graph across multiple MPI ranks, necessitating boundary data exchange between processes. This inter-process communication introduced substantial overhead and became a major performance bottleneck.

In scenarios involving small datasets and sparse edge updates (particularly those not dominated by insertions), obtaining any tangible speedup with the full MPI+METIS+OpenMP stack proved extremely difficult. To realize meaningful performance improvements, the system must be configured with specific criteria: a sufficiently large graph, a high insertion rate, a well-connected structure, and computational resources on the order of 12–16 MPI processes coupled with approximately 64 threads.

# References

1. A. Khanda, S. Srinivasan, S. Bhowmick, B. Norris, and S. K. Das, "A Parallel Algorithm Template for Updating Single-Source Shortest Paths in Large-Scale Dynamic Networks," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 4, April 2022.