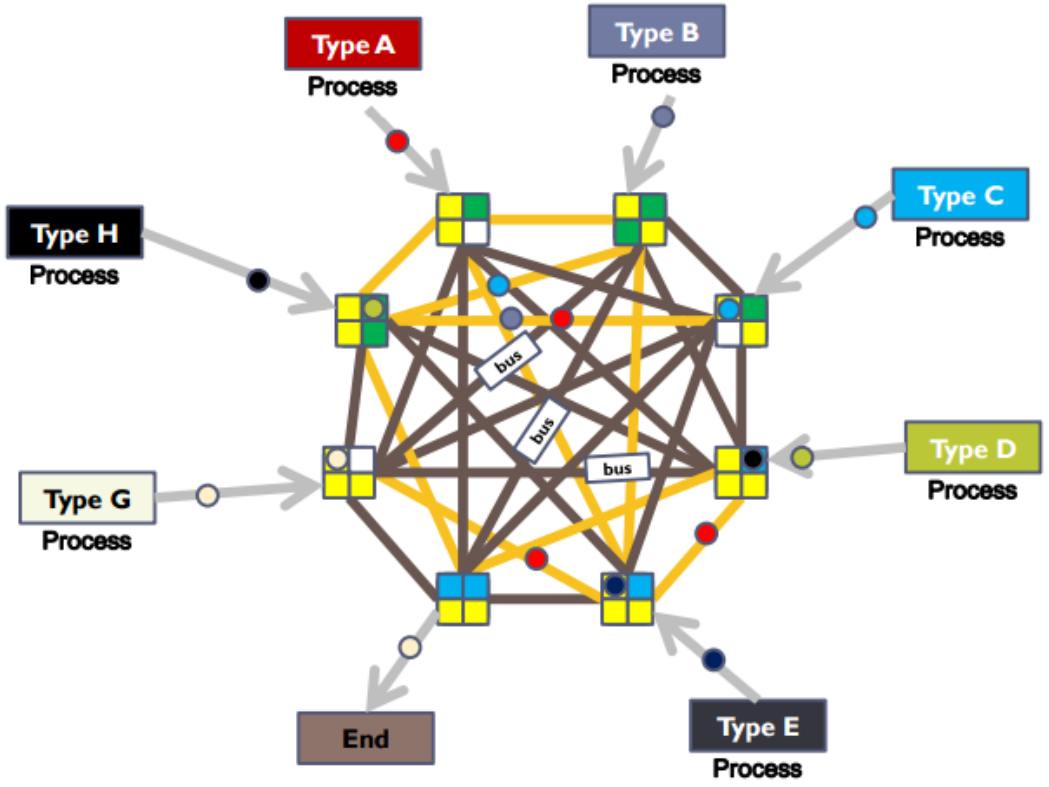
Mini Factory with
High throughput & Low cost

Semiconductor Systems Engineering
SKKU
Young Dae KWON

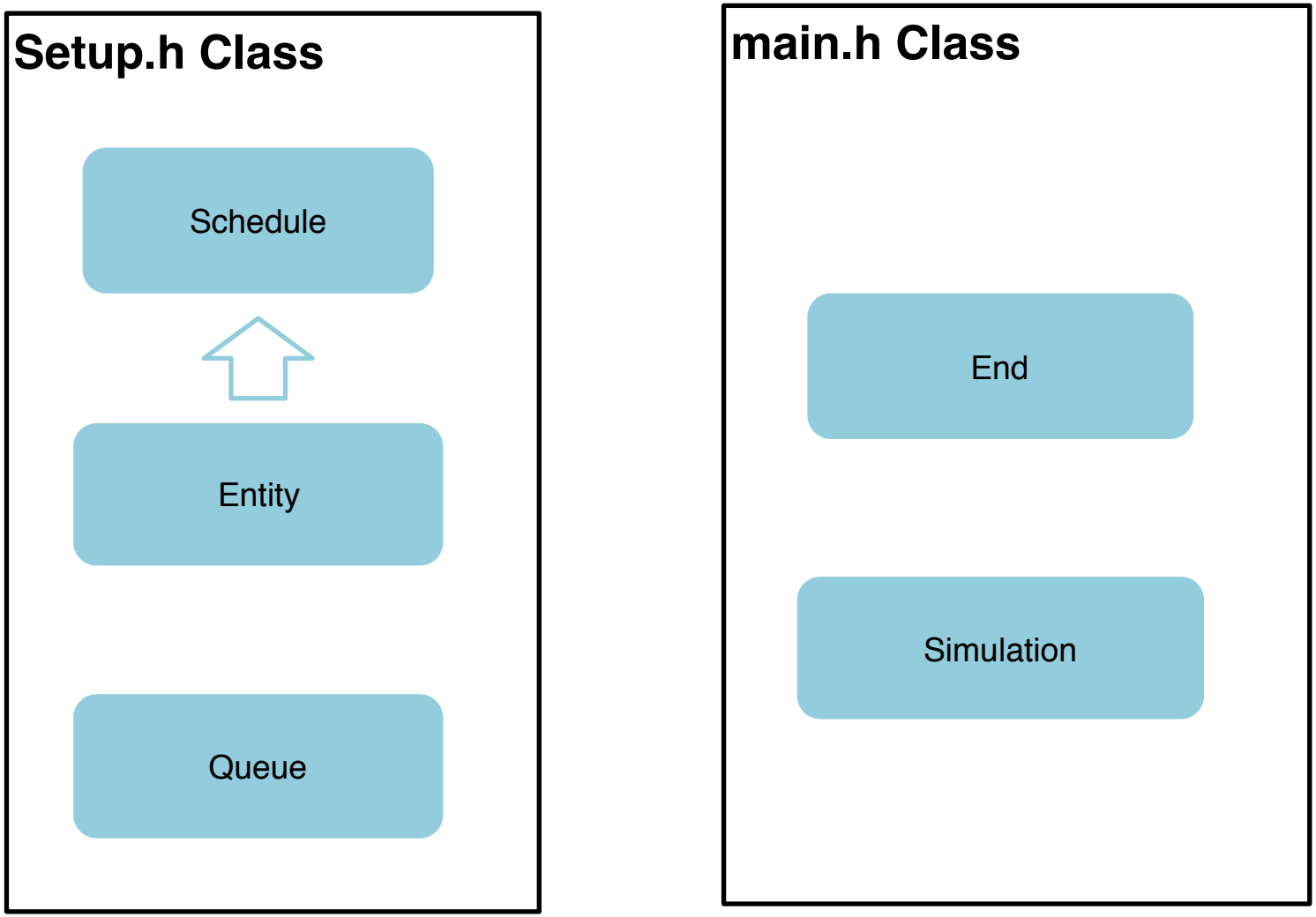
1 Introduction

A. Mini Factory (Server Queue Scheduling) Simulation

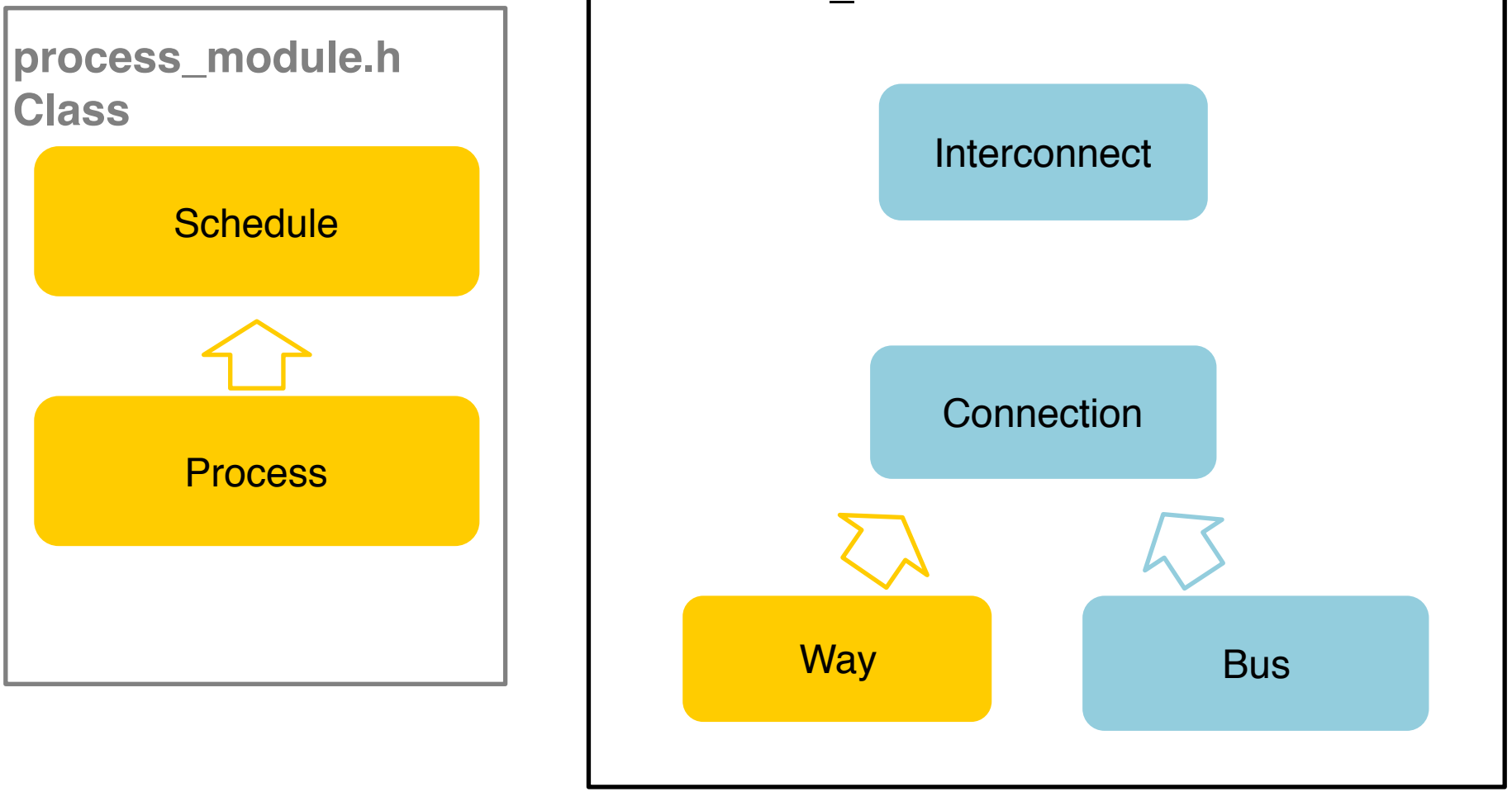
- Analyzed conditions causing bottle neck in server queue scheduling
- Improved the efficiency of server queue with limited resources



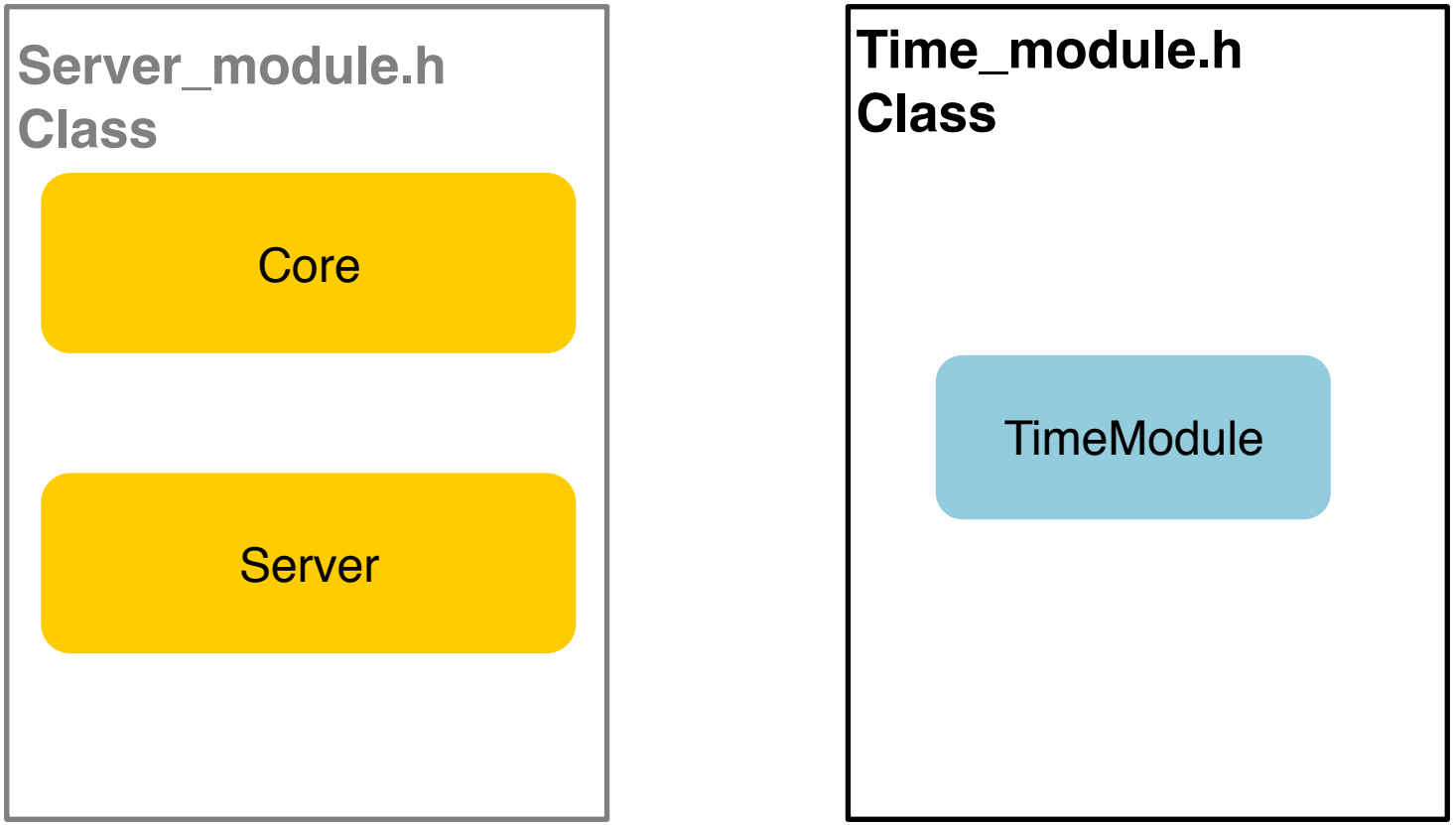
2 Model


Setup.h Class
Schedule
Entity
Queue
main.h Class
End
Simulation

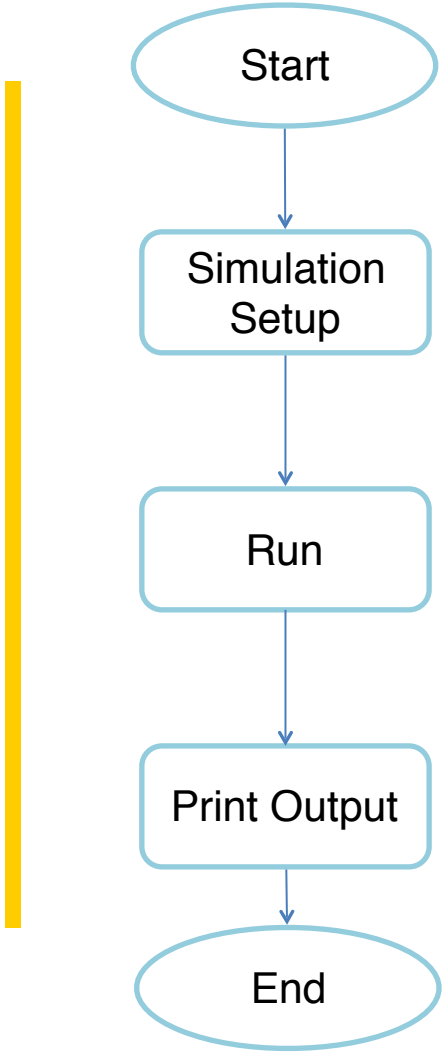
2 Model

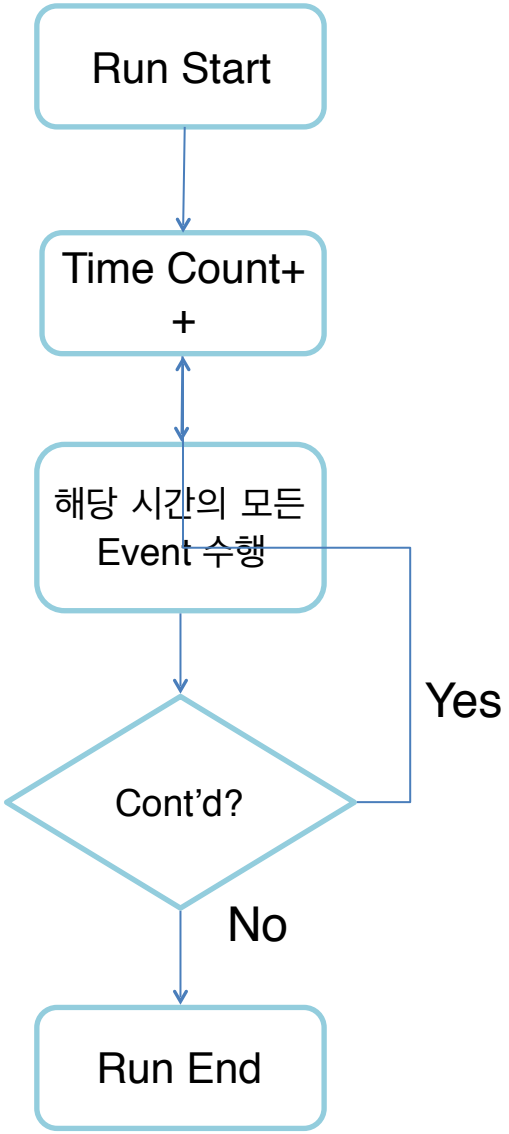

process_module.h Class
Schedule
Process
connection_module.h Class
Interconnect
Connection
Way
Bus

2 Model


Server_module.h
Class
Core
Server
Time_module.h
Class
TimeModule

3 Algorithm (Flow Chart)





3 Algorithm (Event Driven)

Map< int , vector<event> > EventTime;

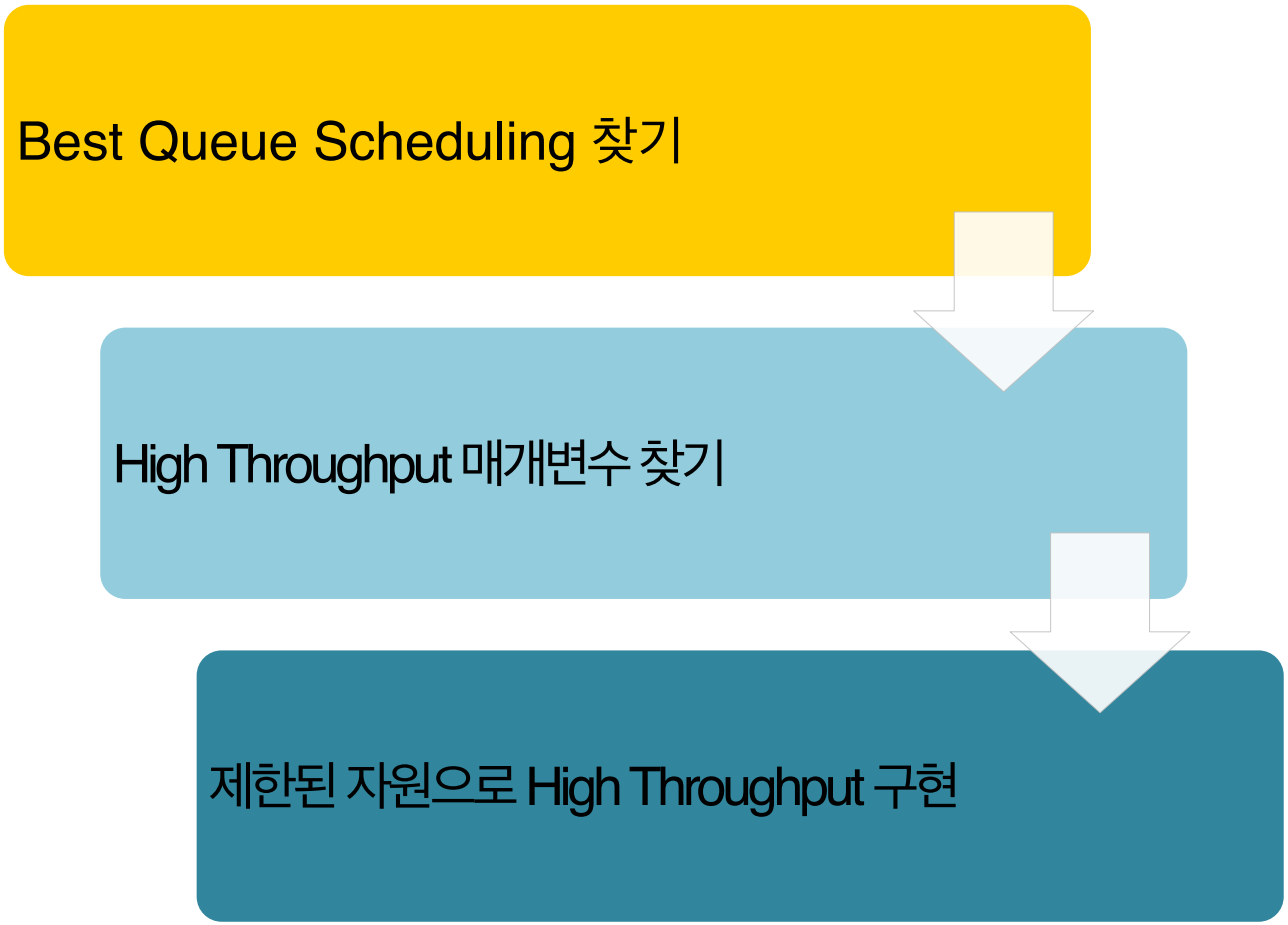
1. Event Driven의 자료구조로 C++ STL인 Map 사용

   - **Time Tick 을 Map의 key로 활용한다.**
   - **Key에 대응하는 Map의 value로는 event구조체를 가지는 벡터로 구성하여 Event Driven으로 구현하였다.**

```
==== event 구조체 구성 ====
processID
serverID
coreID
busID
wayID
```

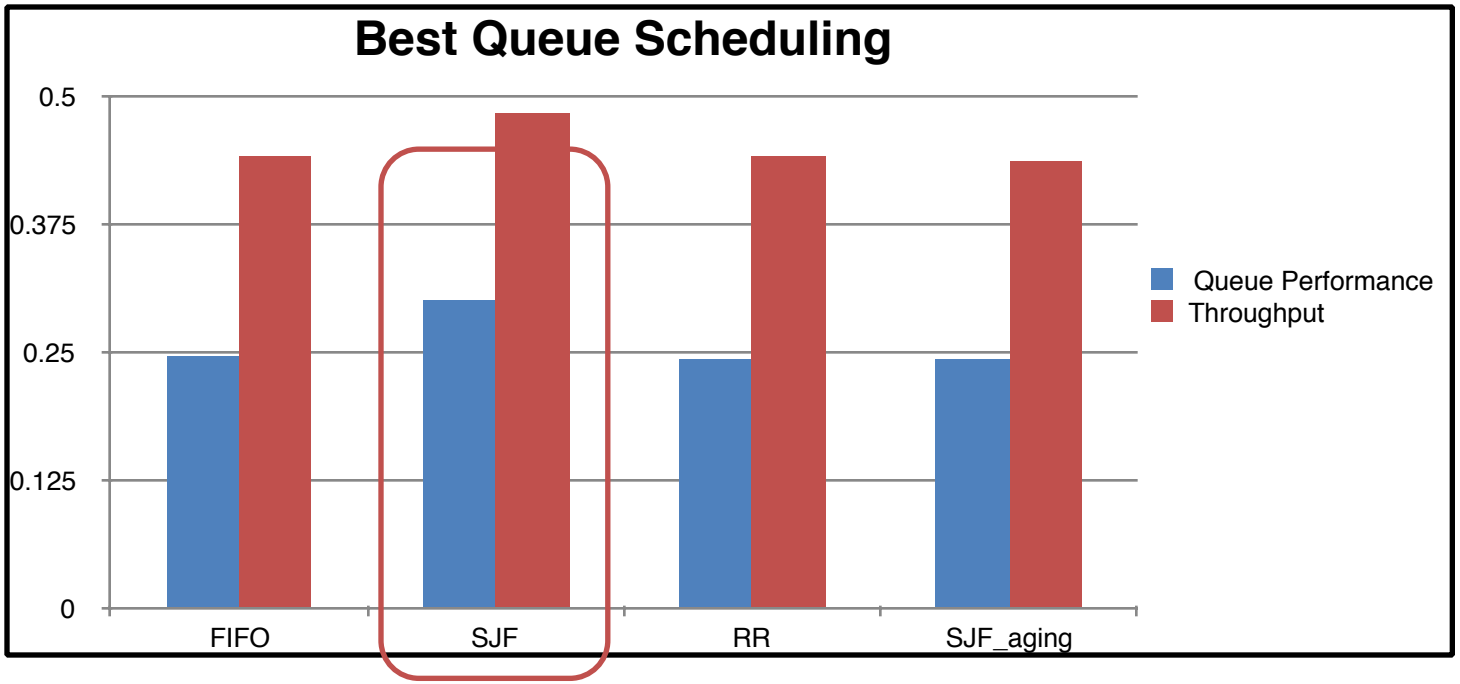
3 Algorithm

전개 과정



4 Experiment (Best Queue Scheduling 찾기)

==== 매개변수 Setting ====
커넥션 당 Way 수 : 2
Way Capacity : 50
Simulation Time : 10,000



Throughput = 서버에서 끝난 Entity 수 / Simulation Time
Queue Performance = Queue 에서 기다린 시간/ 서버에서 끝난 Entity 수

4 Experiment (High Throughput 매개변수 찾기)

1. 매개변수 세팅 시 고려한 점

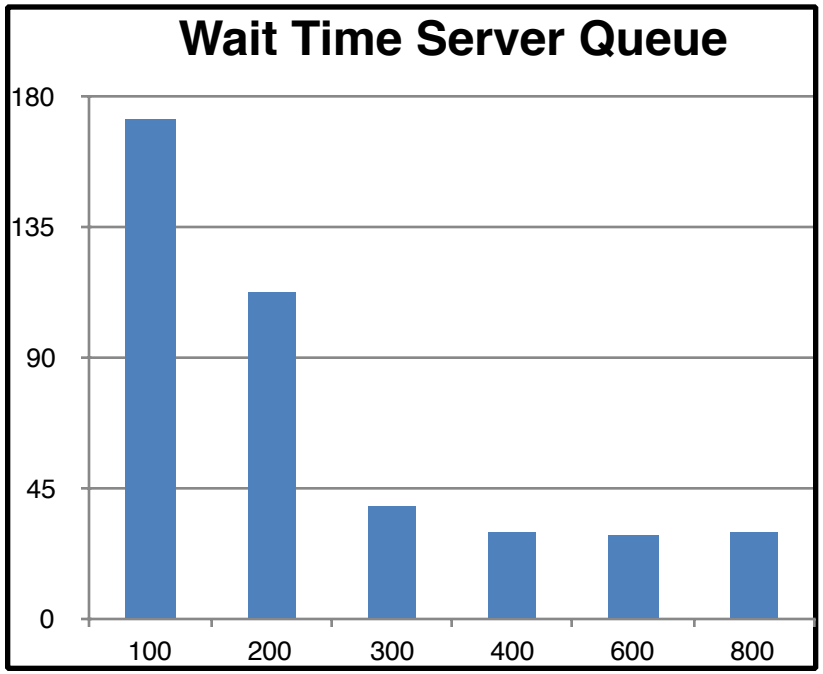
- 정해진 Speed가 있는 Way보다 서버 사이를 time tick 한 번에 이동할 수 있는 Bus가 이득이다.
- Entity가 이동하기 위해 Connection Queue에서 기다리는 것을 최소한으로 할 수 있다.
- 한 서버가 가질 수 있는 Core는 무제한이므로 Level 1 코어를 모든 서버에 대해 Uniform하게 가지도록 셋팅한다.
- Entity가 Server Queue 기다리는 것을 최소한으로 한다.

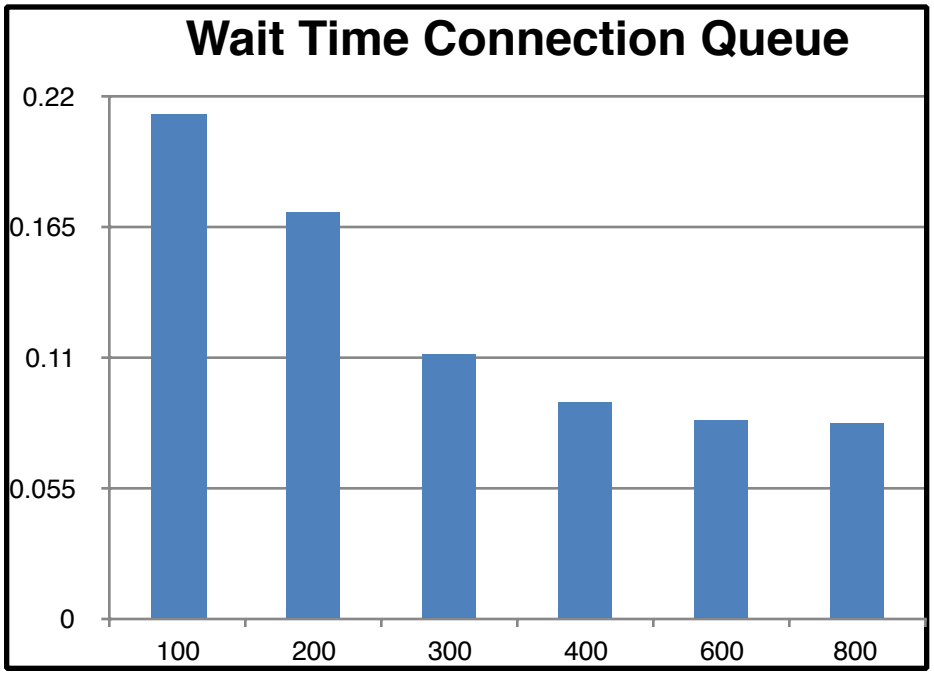
2. Cost를 고려하지 않은 High Throughput 을 위한 매개변수 셋팅

- Number of Ways : 0
- Capacity of Ways : 0
- Number of Buses : 1
- Capacity of Buses : 1000
- Speed of Buses : 500
- Simulation Time : 10,000

4 Experiment (High Throughput 매개변수 찾기)

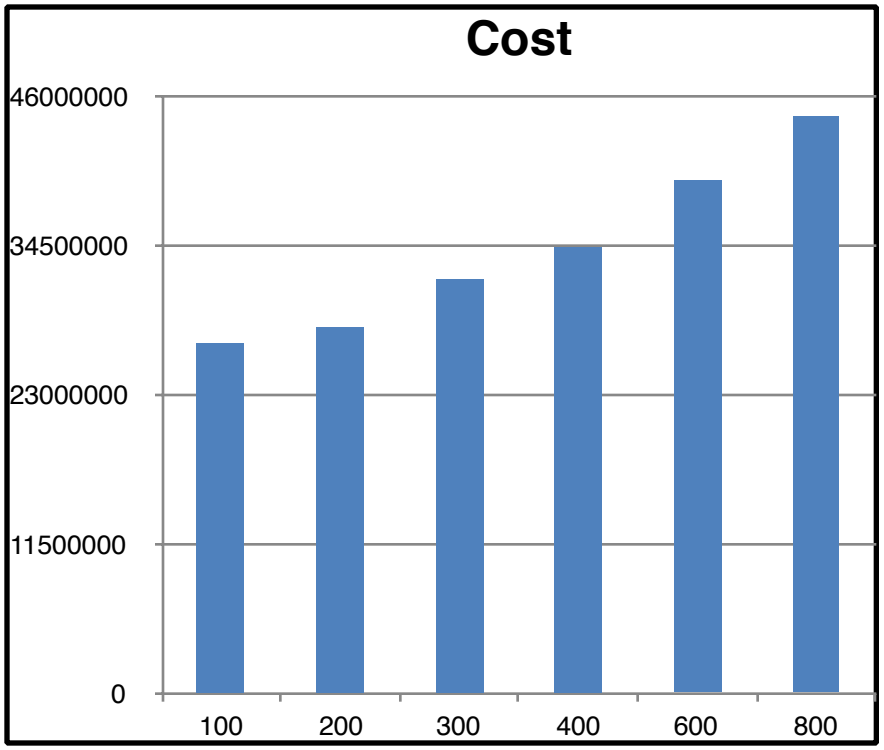
==== Setting ====
Produced Entity : 130,665
X축 : 각 서버에서 사용된 Core개수
Simulation Time : 10,000

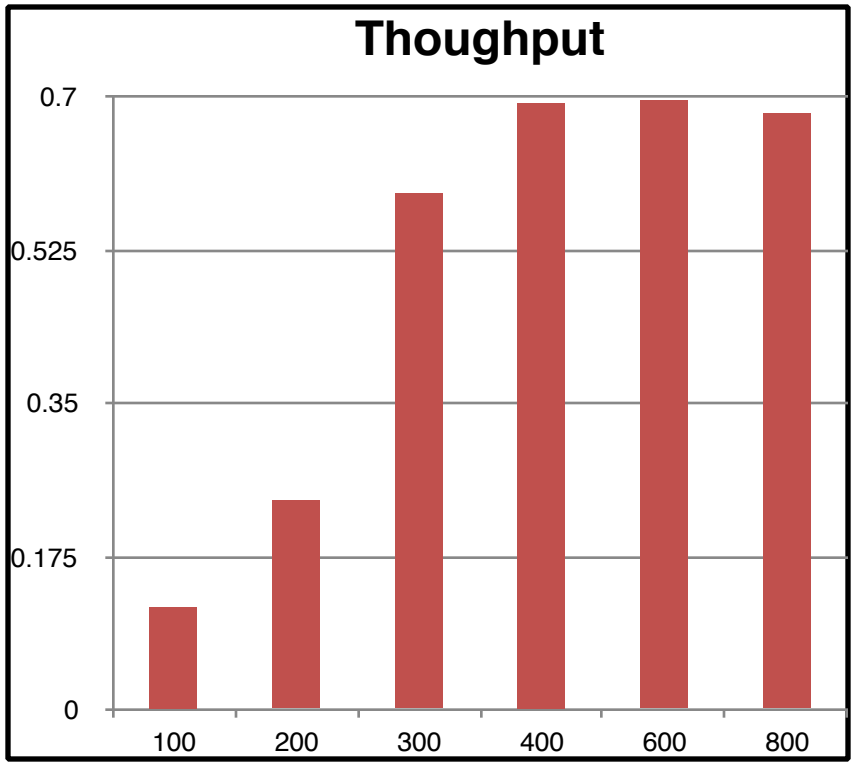




4 Experiment (High Throughput 매개변수 찾기)

==== Setting ====
Produced Entity : 130,665
X축 : 각 서버에서 사용된 Core개수
Simulation Time : 10,000

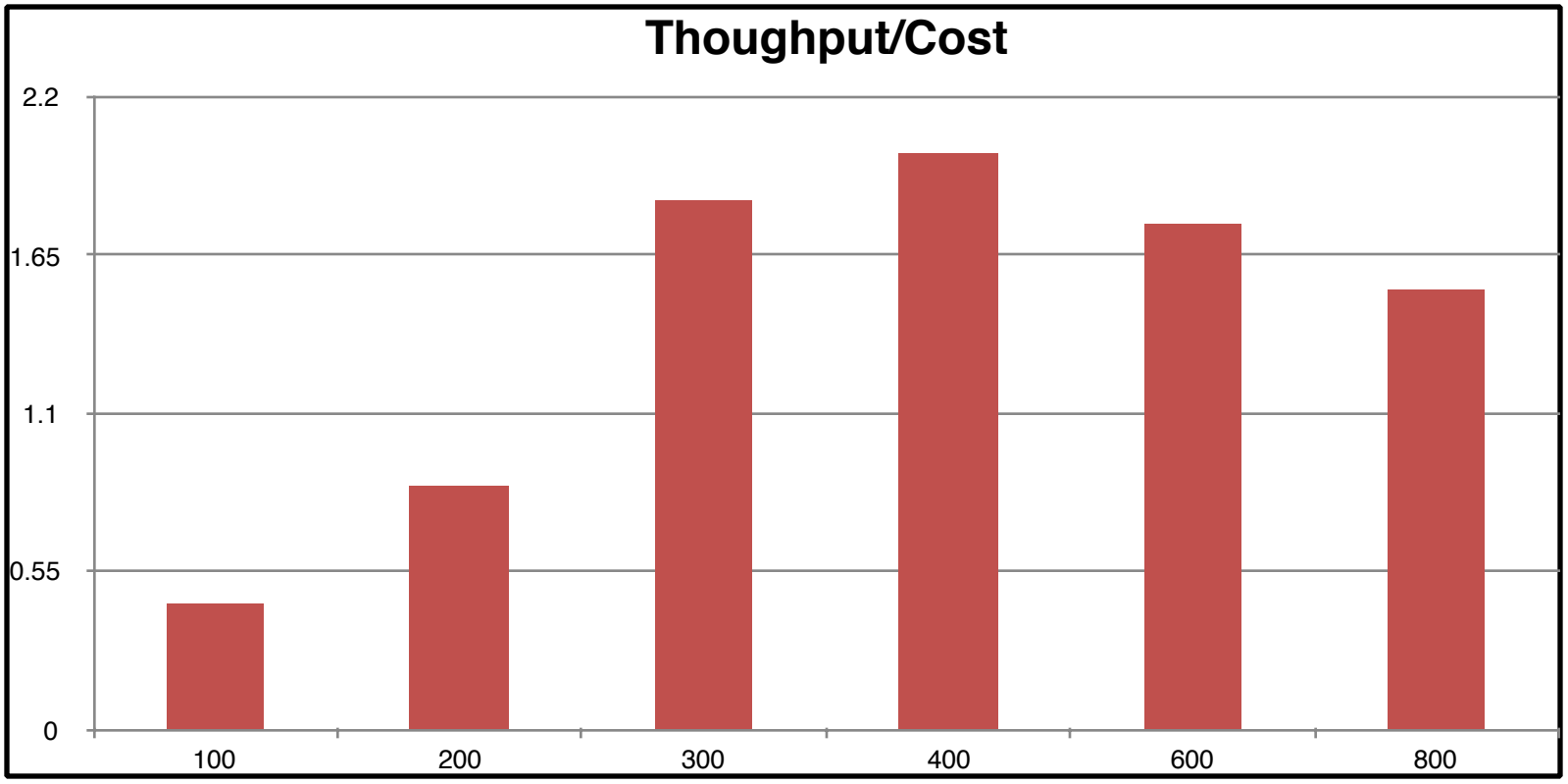




Throughput = Finished Entity / Produced Entity

4 Experiment (High Throughput 매개변수 찾기)

==== Setting ====
Produced Entity : 130,665
X축 : 각 서버에서 사용된 Core개수
Simulation Time : 10,000



Throughput = Finished Entity / Produced Entity

4 Experiment (High Throughput 매개변수 찾기)

실험 결과 정리

1. 각 서버당 사용하는 코어의 개수를 증가시킴에 따라 Cost가 Linear하게 증가함.

2. Mini Factory의 성능을 결정하는 다른 요소들 Server Queue Time, Connection Queue Time, Throughput은 일정한 값에 수렴하는 것을 확인할 수 있음.

3. 즉, Throughput 대비 Cost는 단순히 특정 값에서 최대값을 갖는 것을 알 수 있음. (위 실험에서는 코어 개수 400개)

4 Experiment (제한된 자원으로 High Throughput 구현)

1. Server중 Bottle Neck이 되는 지점 찾기

- 평균적으로 많은 Entity를 처리하는 Server가 Bottle Neck이 되는 지점이라 가정한다.
- 각각의 서버가 일정한 비율로 Entity들을 처리하고 있다면 그 비율에 비례하는 만큼의 코어를 각 서버에 배분하는 것이 가장 이상적이라 할 수 있다.

2. 실험 진행 방법

- Time Count 1000, 2000, 3000, 4000, 6000, 8000, 10,000 에서 서버별 Entity처리량의 시간에 대한 기울기를 구한다.
- 위에서 구한 기울기들을 서버 별로 평균을 구한다. (Warm up period 제거 => Time Count 1000은 제외함)
- 위에서 구한 평균에 일정한 값을 곱하여 내림한 다음 그 값을 코어의 개수로 사용한다.

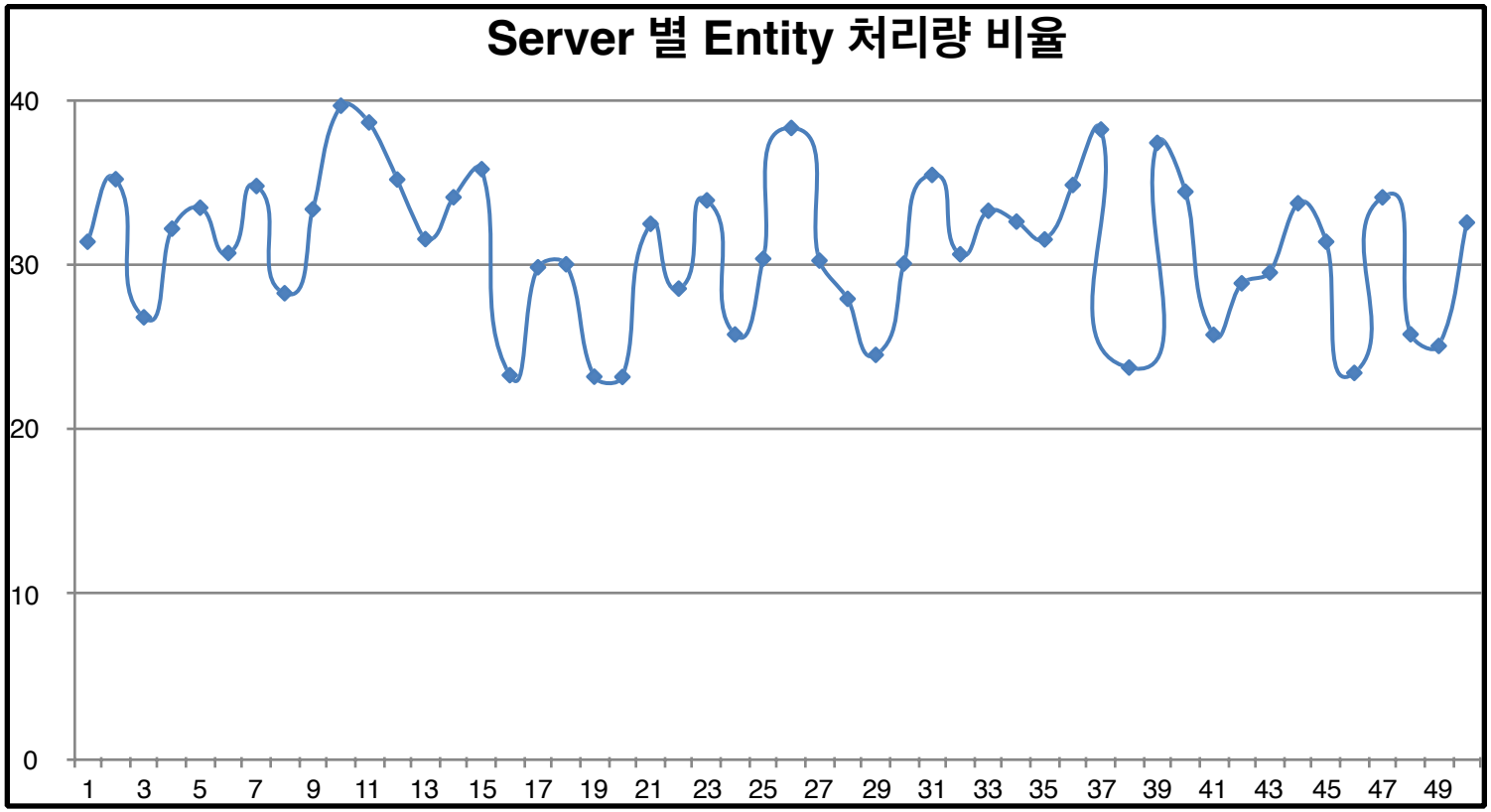
• 자세한 자료는 hw2_stats 파일 참고

4 Experiment (제한된 자원으로 High Throughput 구현)

==== Setting ====
Y축 : Entity 처리량
X축 : Time Count

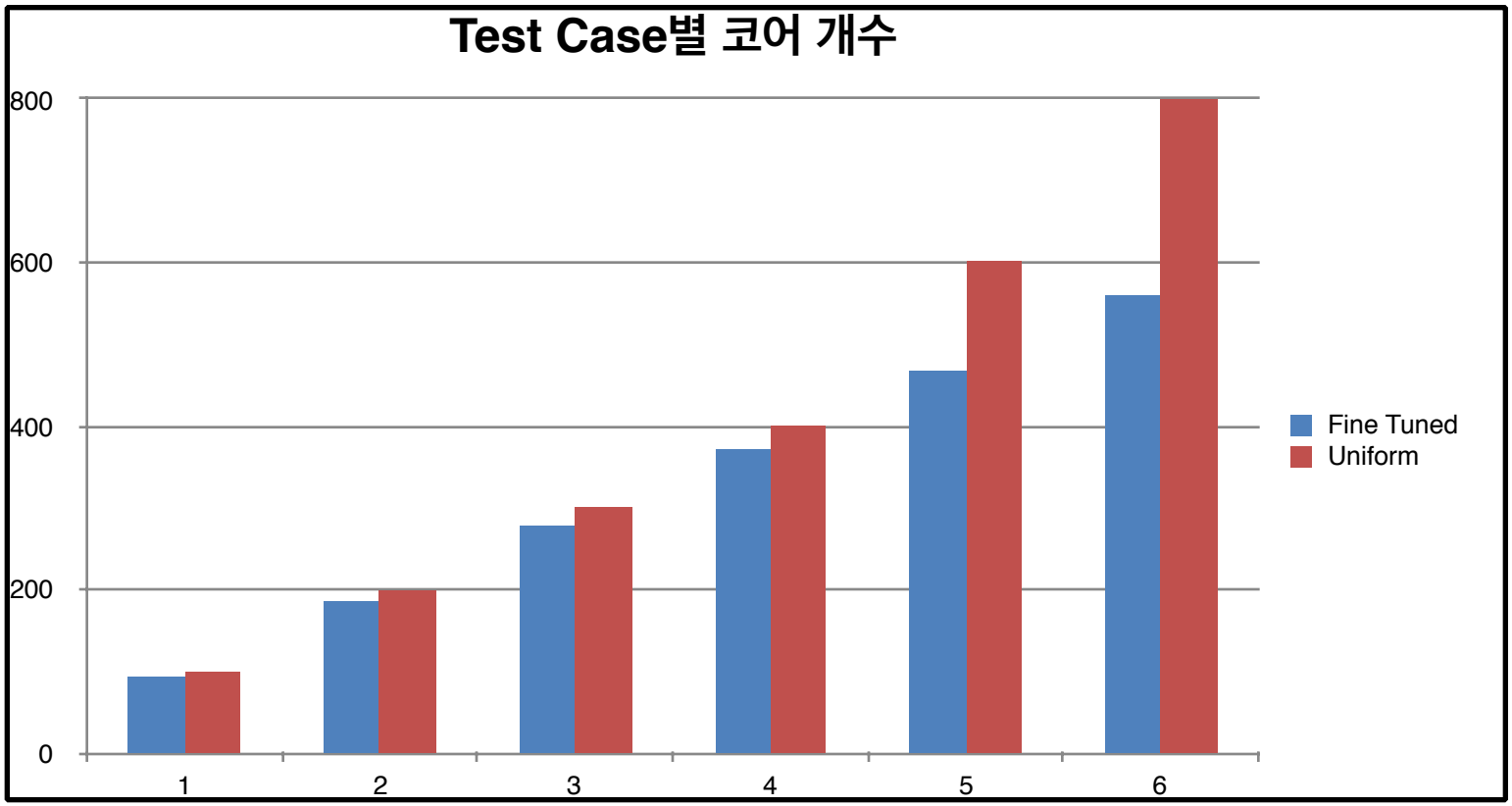
4 Experiment (제한된 자원으로 High Throughput 구현)

==== Setting ====
Y축 : Entity 처리량 비
X축 : Server ID



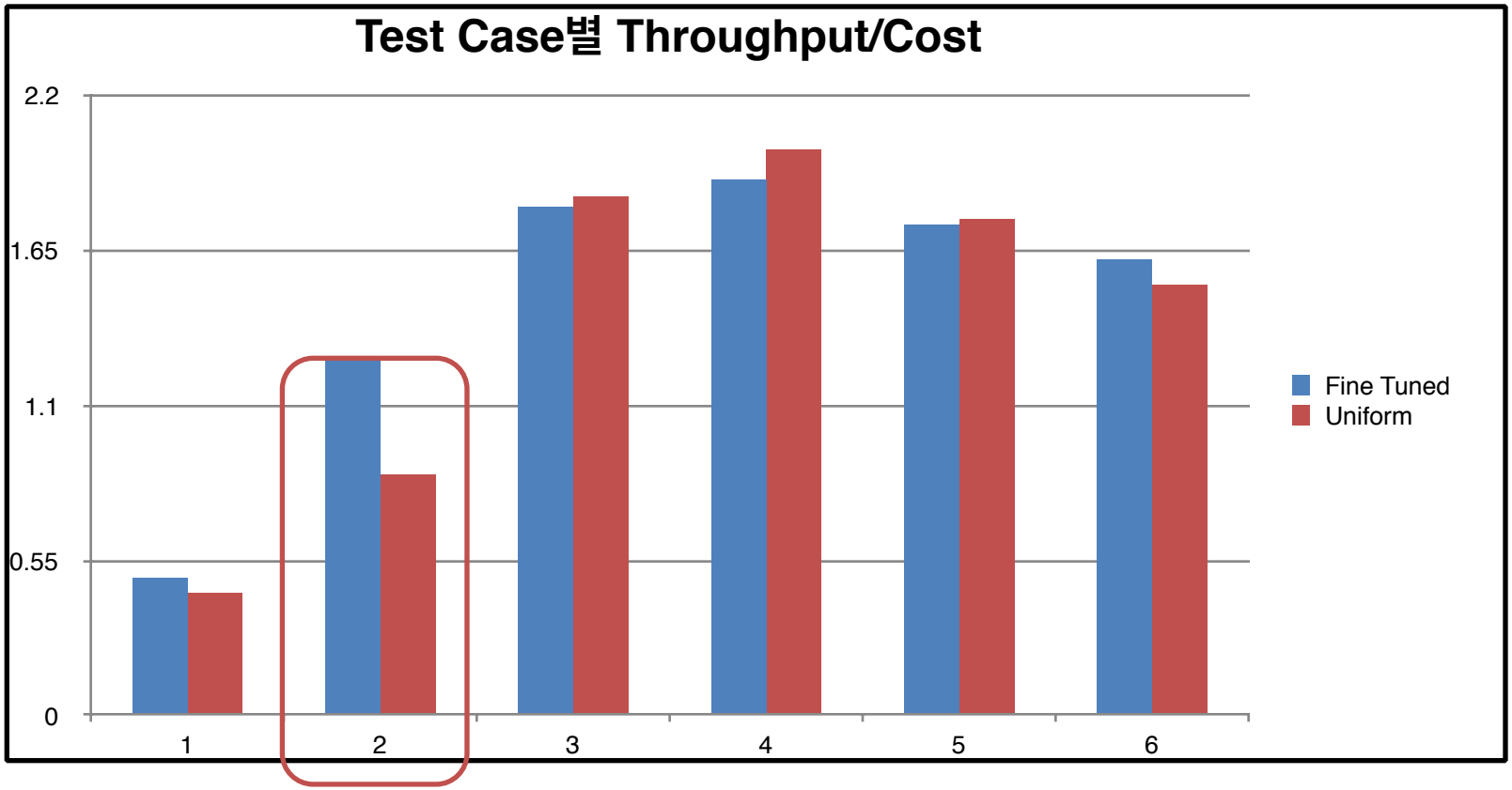
4 Experiment (제한된 자원으로 High Throughput 구현)

==== Setting ====
Fine Tuned : 각 서버는 Entity처리량 비율에 따라 코어를 가짐
Uniform : 각 서버는 동일한 개수의 코어를 가짐



4 Experiment (제한된 자원으로 High Throughput 구현)

==== Setting ====
Fine Tuned : 각 서버는 Entity처리량 비율에 따라 코어를 가짐
Uniform : 각 서버는 동일한 개수의 코어를 가짐



4 Experiment (제한된 자원으로 High Throughput 구현)

실험 결과 정리

1. 각 서버의 Entity처리량을 확인하여 Bottle Neck이 되는 서버를 알아낼 수 있었다.

2. 서버가 가지는 평균적인 코어 개수가 약 400개 이상이 되면 Throughput/Cost가 오히려 내려가는 것을 확인할 수 있었다.

3. 우리가 Fine Tuning하여 각 서버의 코어의 개수를 정한 모델은 자원이 한정된 환경에서 매우 좋은 성능을 보이는 것을 확인할 수 있었다. (각 서버당 평균 코어 개수 200개 이하)

5 Conclusion

1. 컴퓨터 시스템을 간략화한 Mini Factory 시뮬레이션 프로젝트를 통해서 시뮬레이션과 컴퓨터 시스템에 대해 더 잘 이해하게 되었다.

2. 블랙잭 프로젝트에 이어서 OOP (Object Oriented Programming) 방식으로 구현해 봄으로써 C++에 대해 친숙해질 수 있었다.