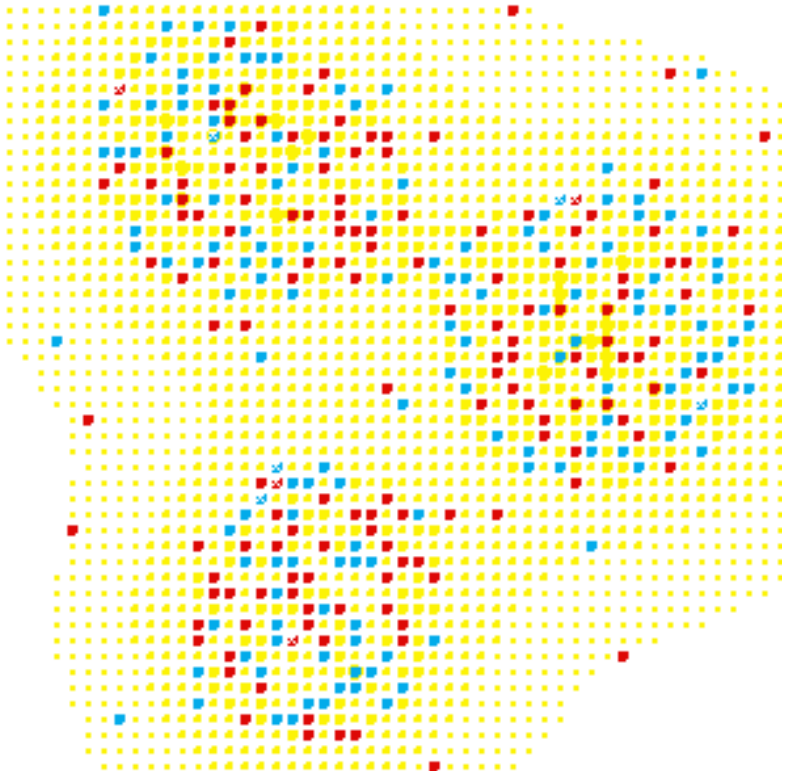
Sugar Scape Simulation using Pthread API

Semiconductor Systems Engineering
SKKU
Young Dae KWON

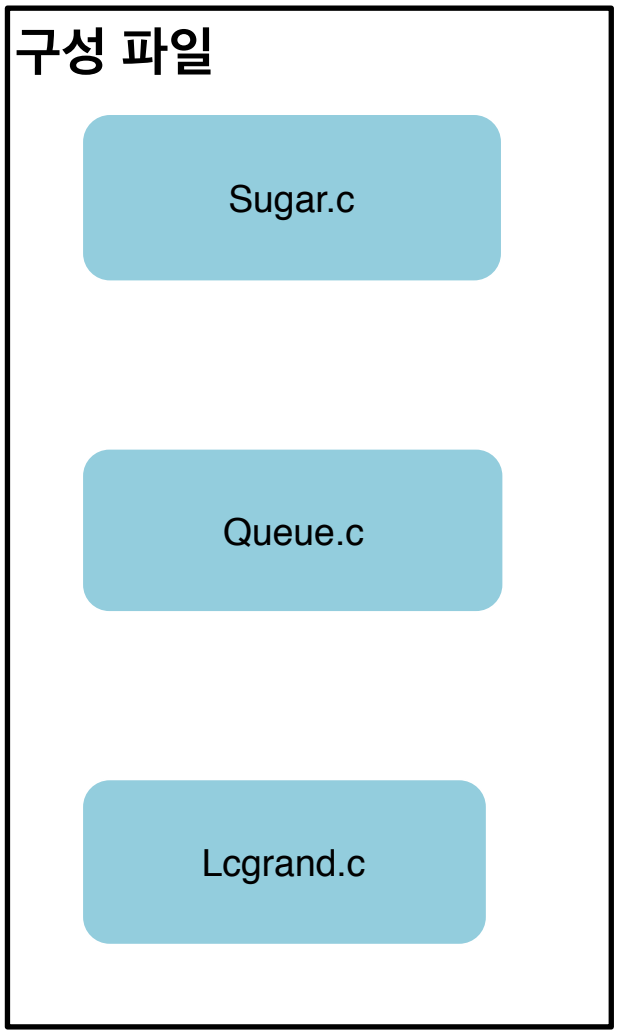
1 Introduction

1. Sugar Scape

- **Simulated behaviors of the agents in an environment of limited resources**
- **Improved the simulation speed using Pthread API and Parallel Programming**



2 Building Blocks

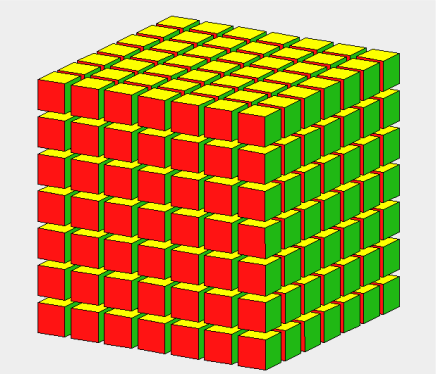


1. Sugar : Sugar Scape 모델을 시뮬레이션하는 메인 파일
2. Queue : 큐 기능을 구현하고 있는 파일
3. Lcgrand : 랜덤 숫자를 생성해주는 파일

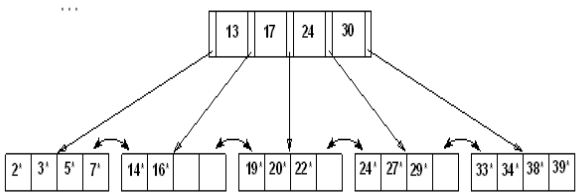
3 Algorithm (B+ Tree & People Array)

문제점

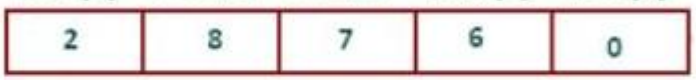
3D Map Array



B+ Tree for Move



People Array for Merge & Eat



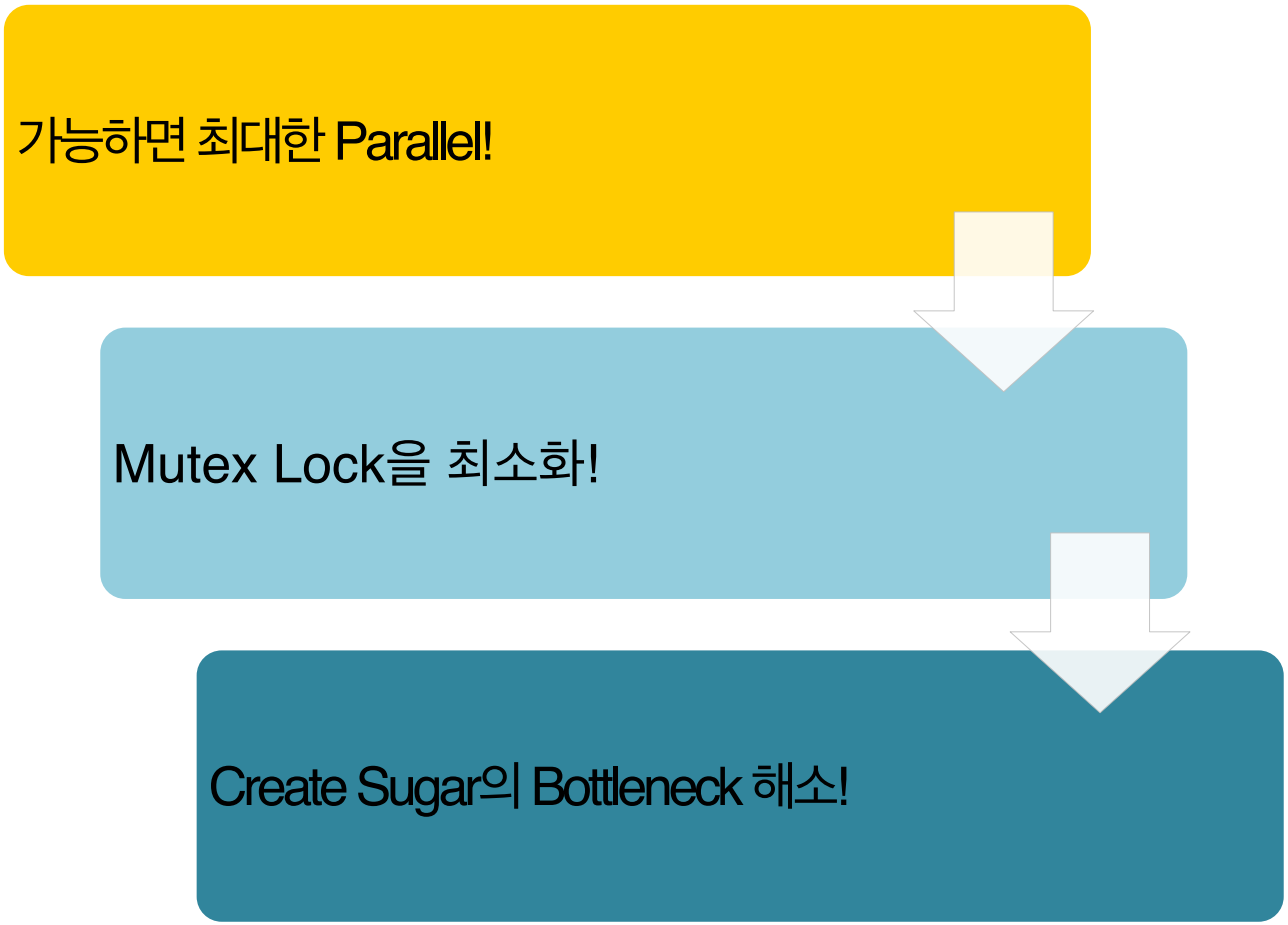
1. Mutex Lock Overhead : 하나의 B+ Tree에 여러 개의 스레드가 동시에 접근하면서 Mutex Lock을 잡으며 오버헤드가 발생

2. Coding Difficulty : Tree의 일을 적절히 나누기 위해서는 Dynamic Partitioning 필요(어렵다!)

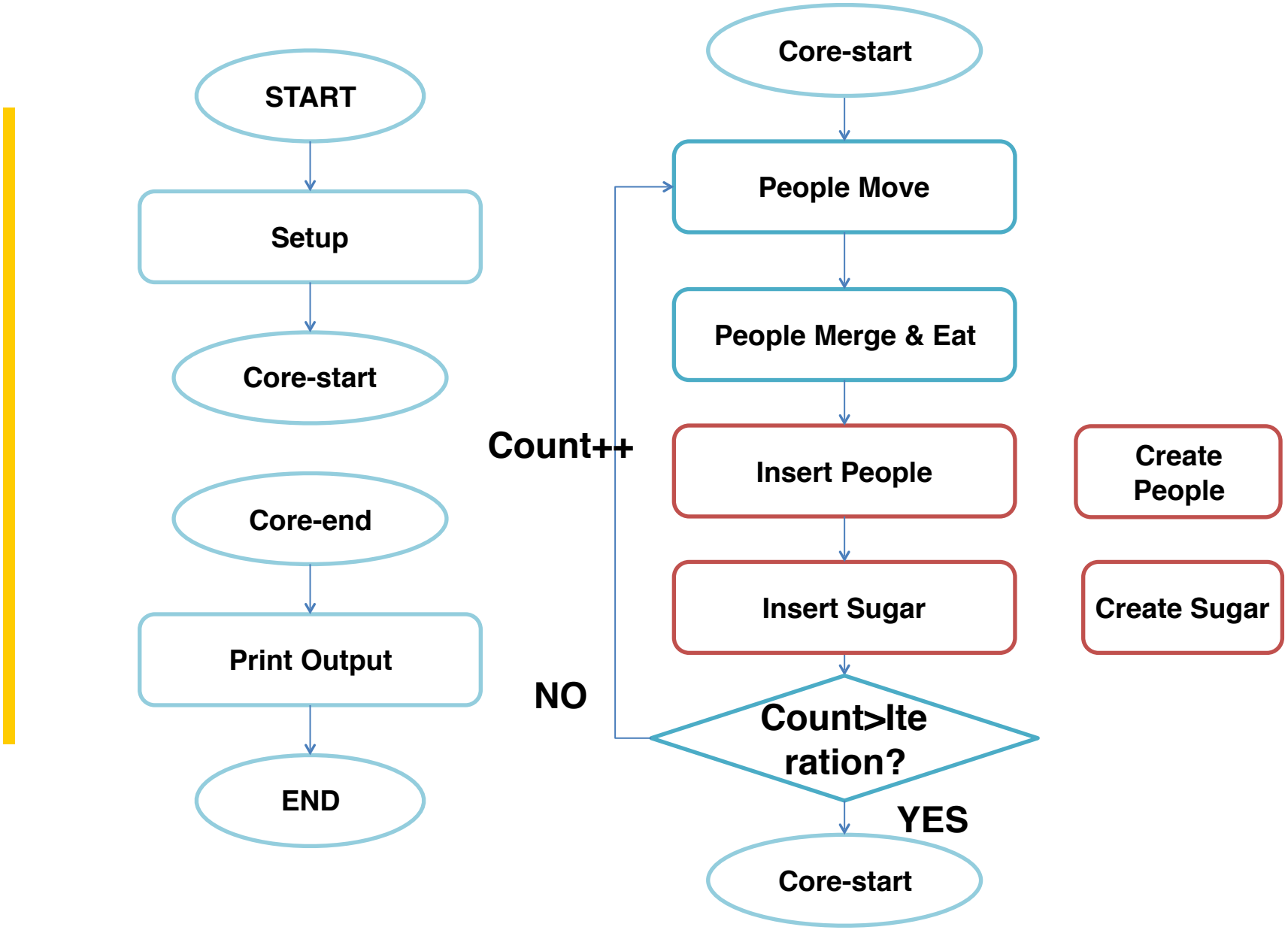
3. 단순히 병렬화만 시켜서는 Create Sugar 함수의 Bottleneck을 피할 수 없음

3 Algorithm

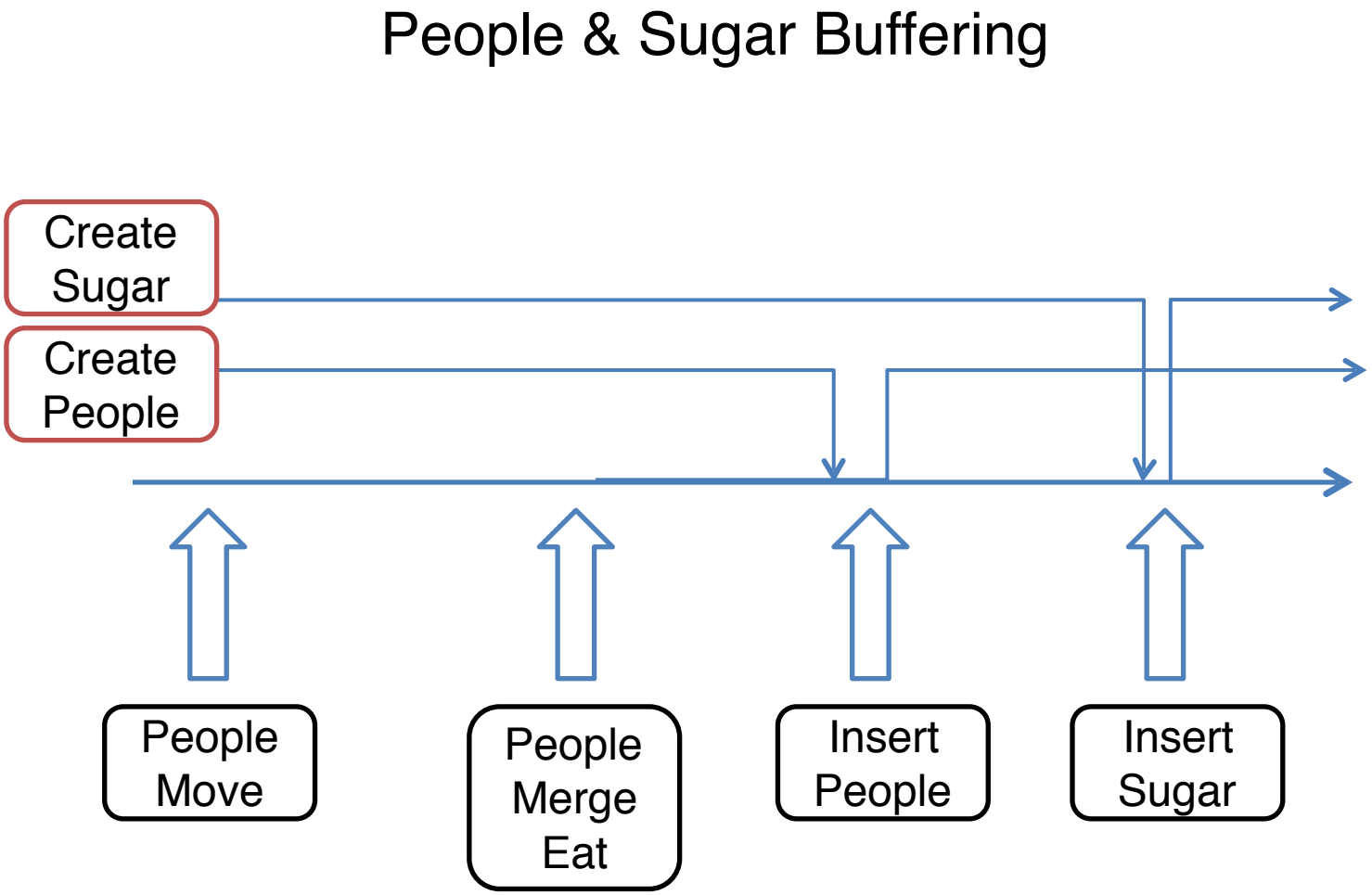
Parallel Programming 아이디어



3 Algorithm (Flow Chart)

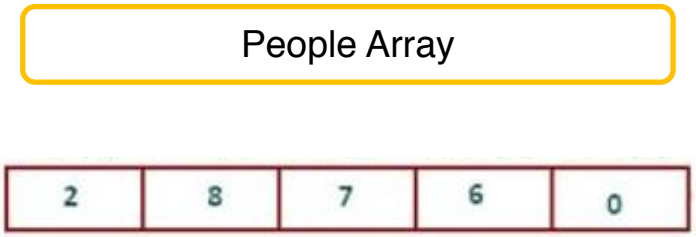


3 Algorithm (Flow Chart)



3 Algorithm (Map with Multiple Arrays)

People Move



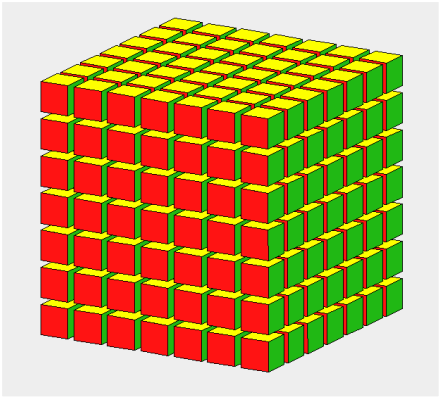
1. People Array : 전체 사람들에 대한 정보를 담고 있는 Array

2. Parallel Algorithm : Block Distribution으로 각 스레드에 작업 할당

3 Algorithm (Map with Multiple Arrays)

People Merge & Eat

3D Map Array



People Point Array

2	8	7	6	0

People Array 2D

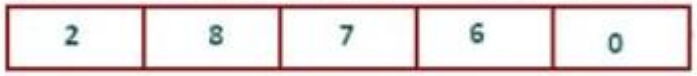
2	8	7	6	0
2	8	7	6	0
2	8	7	6	0

1. Map Array : 각 차원마다 Map size의 크기를 가지는 3D Map Array
2. People Point Array : 사람들의 위치 정보를 중복되지 않게 담고 있는 Array
3. People Array 2D : 각 스레드 별로 지니고 있는 People Array (Mutex Lock 불필요!)

3 Algorithm (Map with Multiple Arrays)

People & Sugar Buffering

People Array Buffer



1. Dead People Array : 미리 만들어질 사람들의 정보를 담고 있는 Array

Sugar Buffer

1. Sugar Buffer X : 미리 만들어질 Sugar의 X 정보를 담고 있는 Array
2. Sugar Buffer Y
3. Sugar Buffer Z

3 Algorithm (Map with Multiple Arrays)

Buffering Policy

1. 약 3Cycle 정도의 만들어질 사람과 Sugar를 미리 만들어서 Buffer에 넣어둔다.

2. **Insert People & Insert Sugar 함수에서 Buffer를 채우는 스레드를 Join 하고 곧바로 Create 해준다.**

3. **1st Cycle에서는 Buffering 스레드 Create 하지 않고 2nd Cycle때부터 Buffering 스레드 Create 한다.**

3번 Policy로 인해 Buffer를 채우는 스레드와 Buffer에서 값을 읽어들이는 스레드가 다른 주소를 Access하기 때문에 두 개의 작업을 동시에 실행 가능

3 Algorithm

Summary

1. **병렬로 돌릴 수 있는 부분은 최대한 병렬화**
   - Move, Merge & Eat, Insert 함수를 병렬로 구현

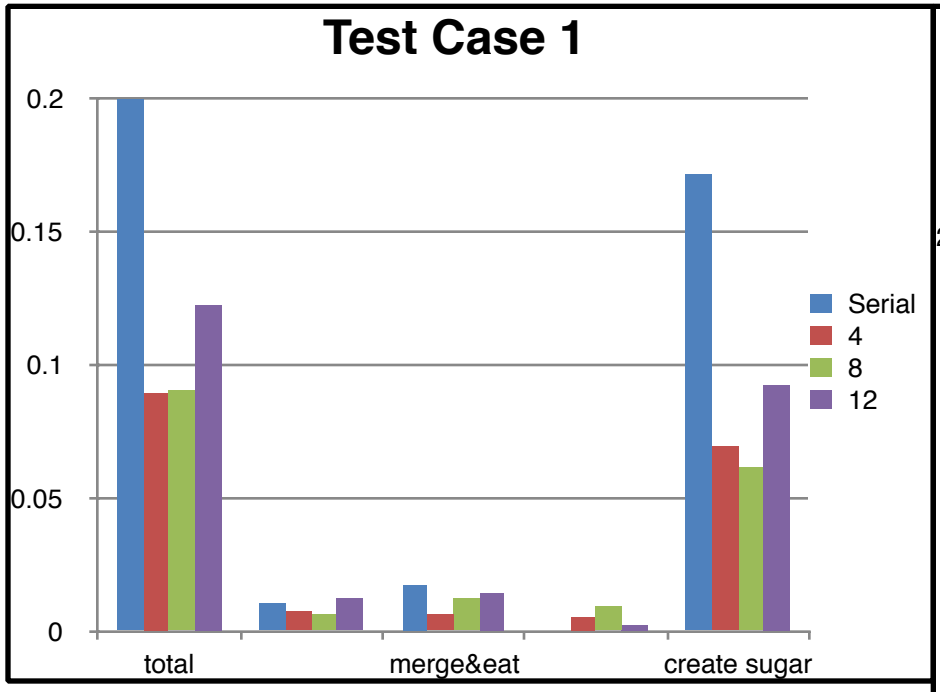
2. **Map의 각 Point별로 Mutex Lock선언 & People Array 2D**
   - Merge와 Insert Sugar에서 Map접근할 때만 Lock사용 => Reduce Overhead!
   - 모든 스레드에 대하여 공통된 변수인 People Array 에 대한 접근 최소화

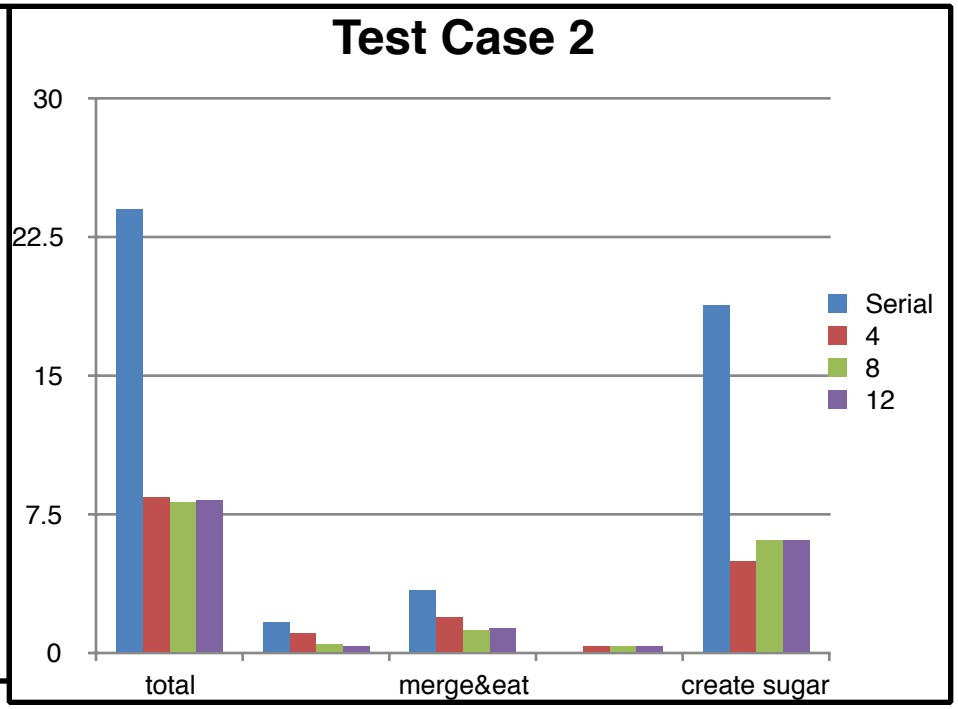
3. **People & Sugar Buffering**
   - Program 성능 저하의 가장 큰 문제였던 Create Sugar 부분의 Bottleneck 해소

4 Experiment

==== Test Case 2 ====
total_loop : 100
map_size : 200
total_people : 500

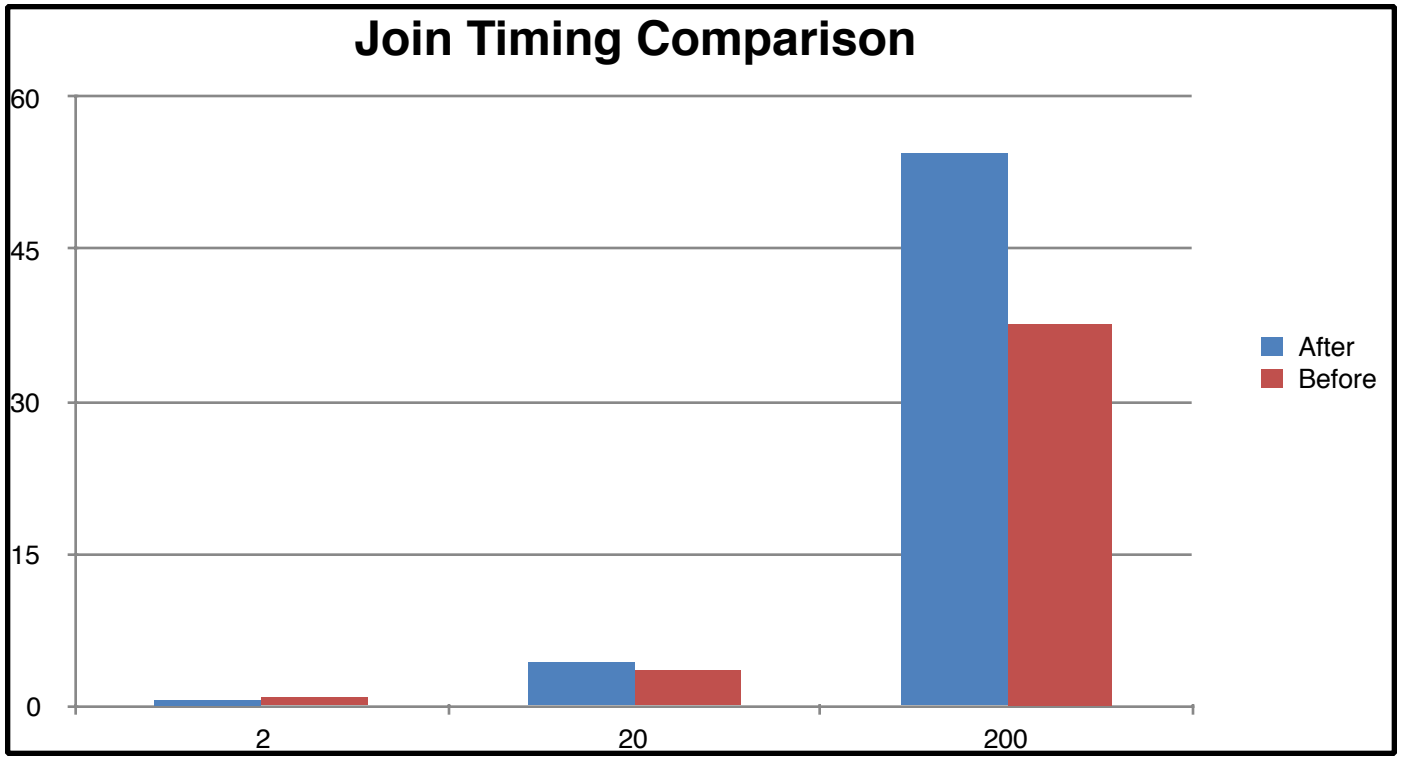
==== Test Case 3 ====
total_loop : 100
map_size : 500
total_people : 50,000





4 Experiment

==== Join Timing Comparison ====
map_size : 200
total_people : 100,000
X축 : total loop
Y축 : 시간
After : Insert 끝난 후 Create 스레드 호출
Before : Insert 시작 전 Create 스레드 호출



5 Tricks for Performance Boost

1. 1D Long Array 사용
   - Reduce memory access!!

2. **People & Sugar Buffering**
   - Reduce Dependency Problem !!

6 Conclusion

1. 빠른 알고리즘을 개발하기 위해 여러가지 방법들을 시도해 보았다.

2. 가능한 많은 부분에서 **Parallel**하게 동작하면서도 하나의 **Global** 변수에 여러 스레드가 접근하여 오히려 **Sequential** 코드보다 느려지는 것을 방지하는 코드를 구현했다.

3. **Sugar** 를 만들어낼 때 발생하는 **Bottleneck**을 해결하기 위해 여러 가지 방법들을 생각해 보았고 실험을 통해서 검증했다.