Parallelize FP-growth Algorithm

張軒 310551115
王柏偉 310551166
蔡秉達 310554034

Introduction

- **Frequent Pattern Growth Algorithm**

  1. This algorithm is an improvement to the Apriori method.

  2. FP-Growth is an frequent pattern mining algorithm that does not require candidate generation.

- **Two part of FP-growth**

  1. Build FP-Tree

  2. Find Frequent Pattern

Introduction

Build FP-Tree

TID	Items
1	Milk, Bread, Beer
2	Bread, Coffee
3	Bread, Egg
4	Milk, Bread, Coffee
5	Milk, Egg
6	Bread, Egg
7	Milk, Egg
8	Milk, Bread, Egg, Beer
9	Milk, Bread, Egg

Find each item with its corresponding frequency

FIRST SCAN!

Sort the list in decreasing order!

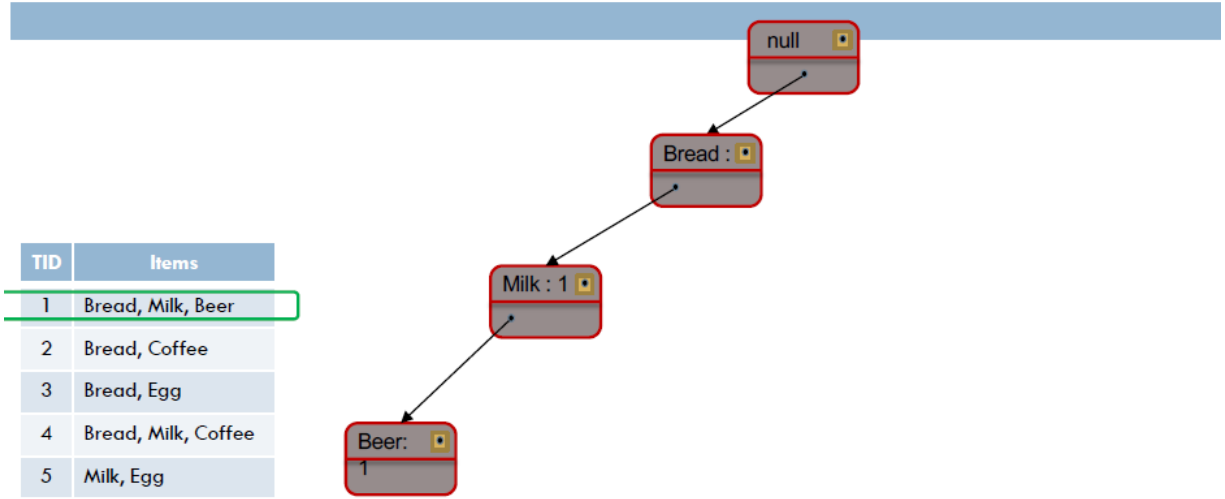
Milk : 6	Bread : 7	Beer : 2	Coffee : 2	Egg : 6

minsup = 2

Introduction

Build FP-Tree

TID	Items
1	Bread, Milk, Beer
2	Bread, Coffee
3	Bread, Egg
4	Bread, Milk, Coffee
5	Milk, Egg
6	Bread, Egg
7	Milk, Egg
8	Bread, Milk, Egg, Beer
9	Bread, Milk, Egg

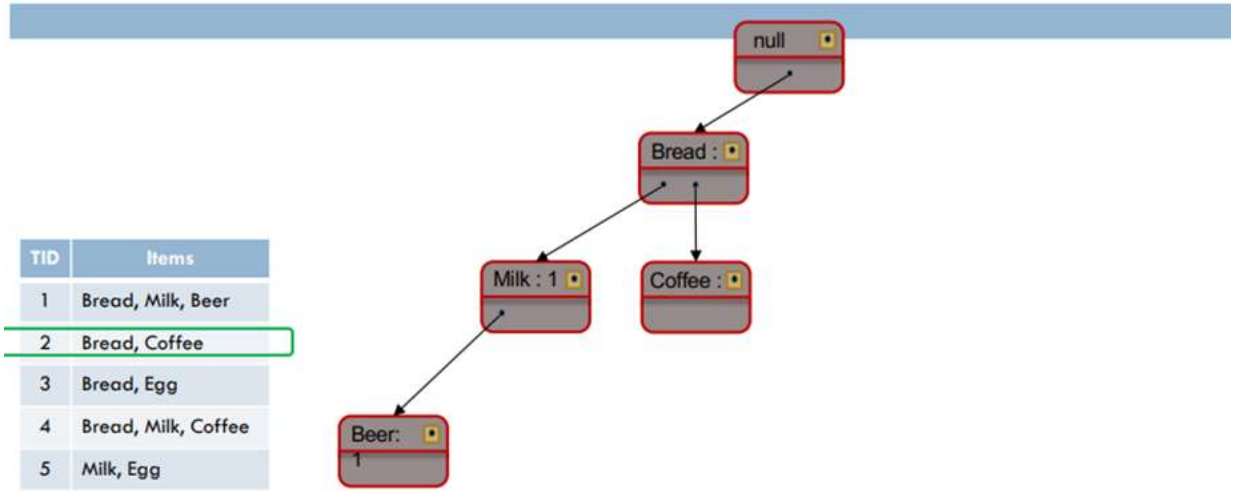


minsup = 2

Introduction

Build FP-Tree

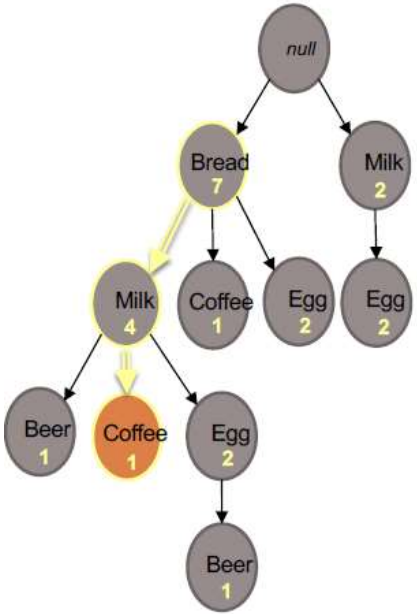
TID	Items
1	Bread, Milk, Beer
2	Bread, Coffee
3	Bread, Egg
4	Bread, Milk, Coffee
5	Milk, Egg
6	Bread, Egg
7	Milk, Egg
8	Bread, Milk, Egg, Beer
9	Bread, Milk, Egg

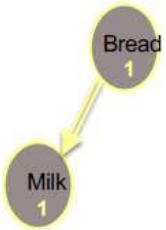


minsup = 2

Introduction

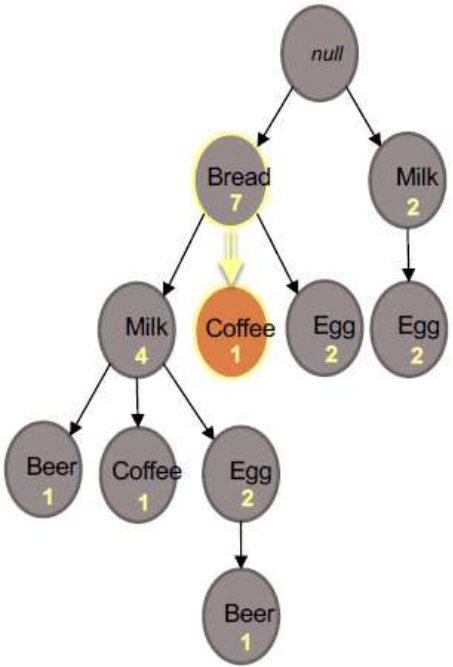
Find Frequent Pattern

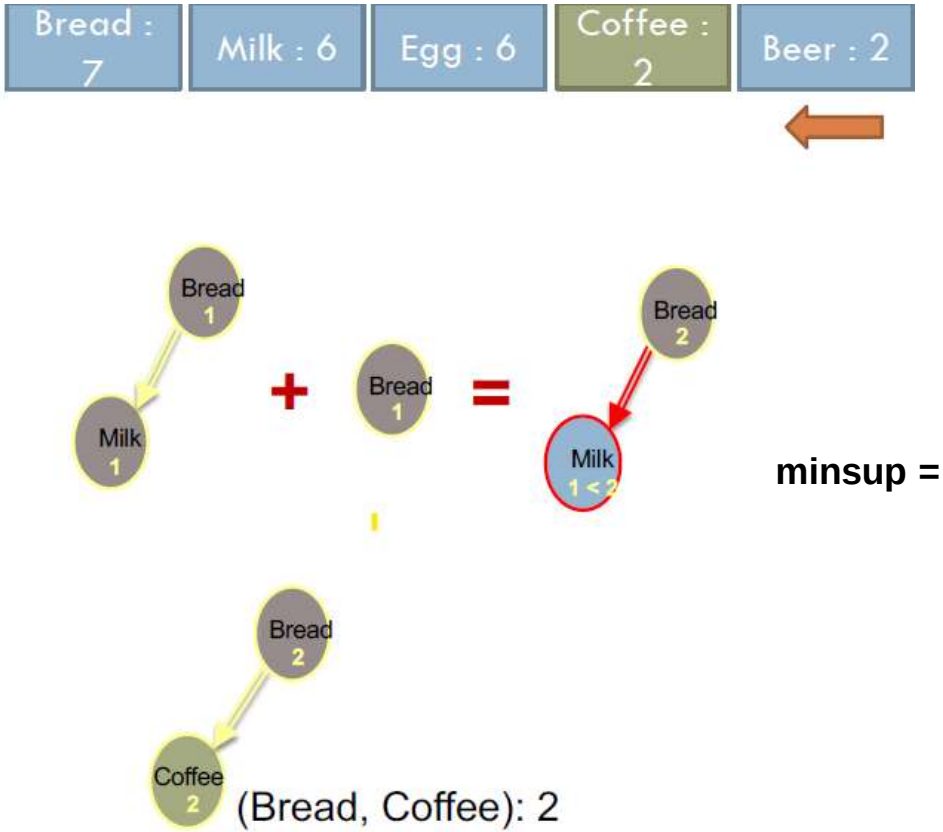




Introduction

Find Frequent Pattern





Method

- **Parallelization**

  Q: How to parallelize tree construction?

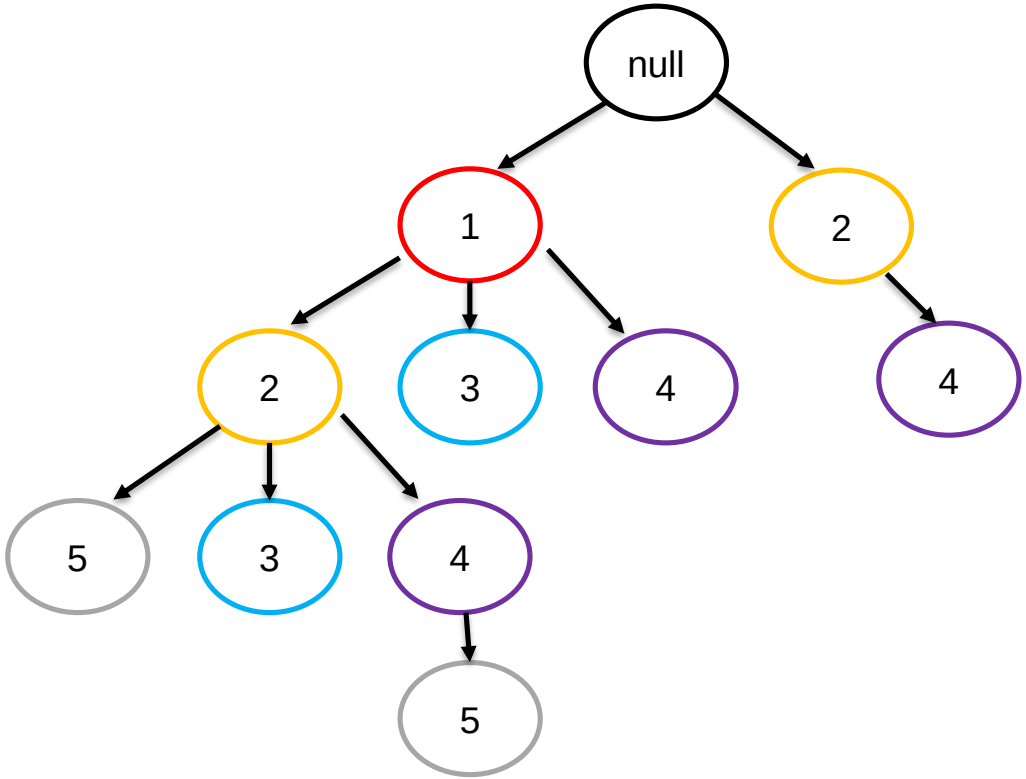
  A: Every header constructs their own tree recursively.

- **Load balancing**

  Q: It's impossible to determine tree size statically,
  how to balance the load among threads?

  A: Generate tasks dynamically and assign to idle threads.
  (OpenMP)

Method (cont'd)



main

↓

createFpTree

↓

findPatterns_parallel

(Build trees concurrently)

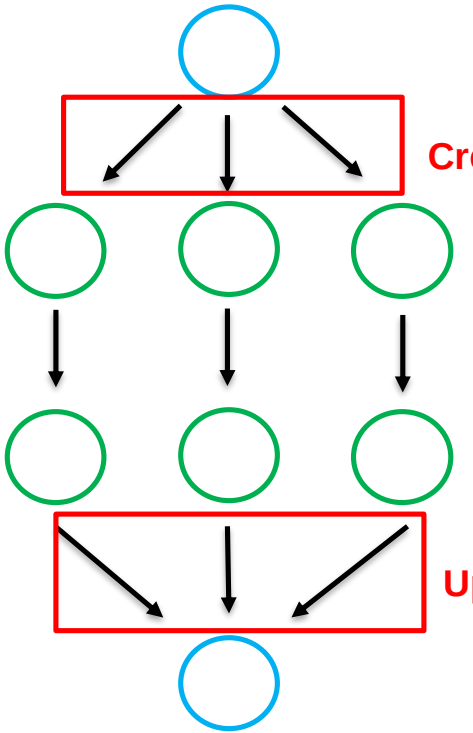
↓

findPatterns_seria

(Build trees recursively and serially)

Method (cont'd)

findPatterns_parallel findPatterns_serial



Create local_patterns

Update local_patterns

```
while(__sync_val_compare_and_swap(&pattern_lock, 1, 0) == 0);
patterns.push_back(local_patterns_ptr);
pattern_lock++;
```

- Update pointers only.
- It takes lots of time to merge map with many elements.
- The patterns are not required to be ordered.

Evaluation

Hardware

- **CPU:** intel i5-11400, 6 core 12 hardware thread, 2.6Ghz.
- **Memory :** DDR4, 32GB, 3200Mhz.

Software

- Ubuntu
- **Compilation:** g++ -std=c++17 –O3 -m64 -fopenmp

Evaluation

Correctness

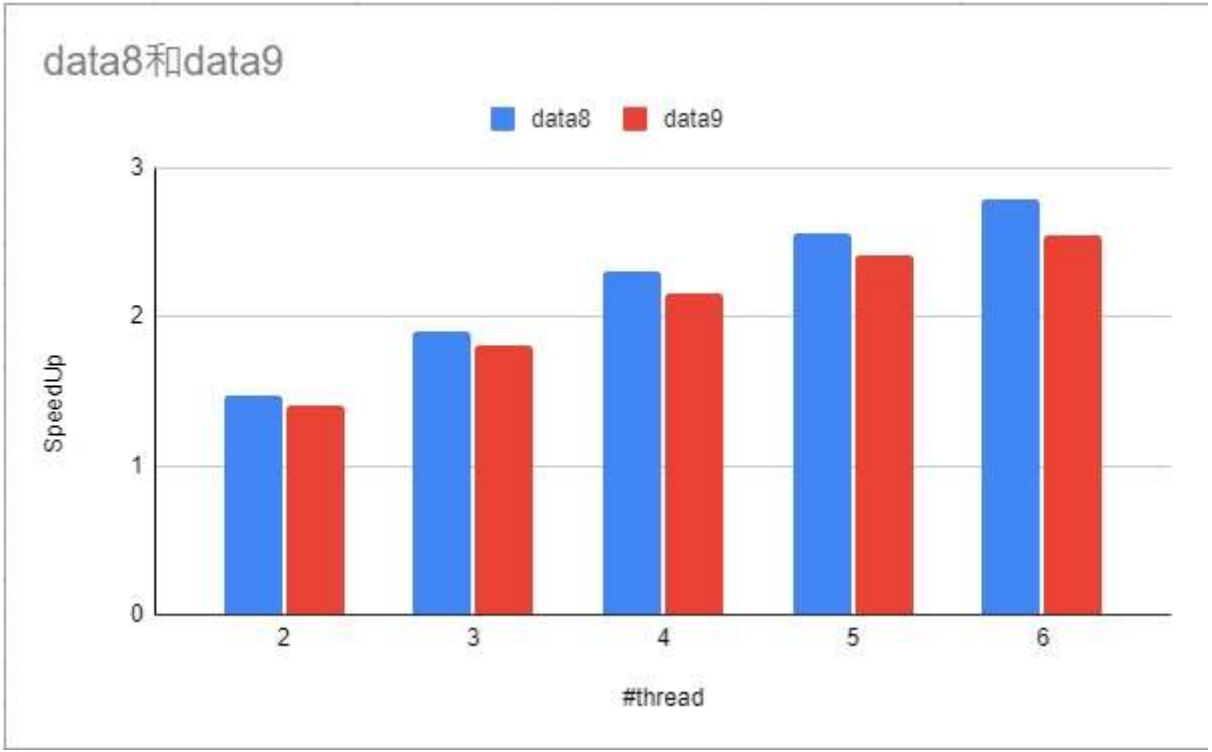
- **Serial :** We find a python version, and we compare our serial result to this version by using diff the output.
- **Parallel :** We merge the result each thread create and compare it to serial version by using diff the output.

Evaluation

Performance

- **Serial :** createFpTree + findPatterns
- **Parallel :** createFpTree + findPatterns
- **SpeedUp :** $\frac{\textbf{Serial}}{\textbf{Parallel}}$
- **Efficiency :** $\frac{\left(\frac{T_{\text{serial}}}{T_{\text{parallel}}}\right)}{p} = \frac{T_{\text{serial}}}{p \cdot T_{\text{parallel}}}$

Evaluation

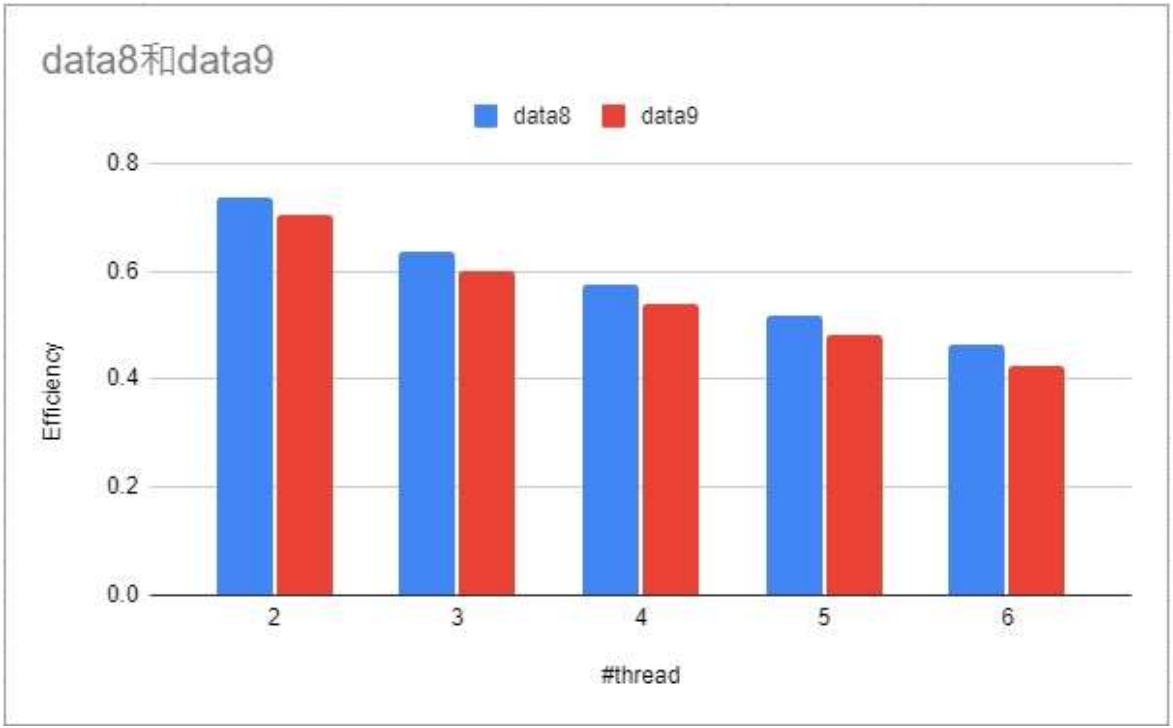


SpeedUp : $\frac{\text{Serial}}{\text{Parallel}}$

Data8 size : 5000

Data9 size : 50000

Evaluation



Efficiency : $\frac{\left(\frac{T_{\text{serial}}}{T_{\text{parallel}}}\right)}{p} = \frac{T_{\text{serial}}}{p \cdot T_{\text{parallel}}}$

Data8 size : 5000

Data9 size : 50000

Thank you