The Parallel Apriori Algorithm [*]

Hongfei Chen (hc3222)

November 20, 2021

1 Overview

The project will implement a Haskell program for the Apriori Algorithm introduced in the paper "Fast algorithms for mining association rules" (Agrawal & Srikant, 1994). I will apply a parallel implementation to improve the performance of the Apriori Algorithm on a large dataset.

2 Background

The Apriori Algorithm is an algorithm for data mining, in particular, association rule mining. It searches for boolean association rule of the frequent itemsets in a dataset, which is useful for discovering the items that tend to appear together in a transaction. The main idea of the algorithm is that for every possible size of itemsets, generate the candidate frequent itemsets from the smaller-sized frequent itemsets and then filter the candidates based on the required minimum support value (Figure 1). The candidate generation consists of the join step (Figure 2) and the prune step (Figure 3), in which the algorithm finds the candidate size-k itemsets by self-joining the size-$(k-1)$ itemsets, and then prune those who have a subset which is not a size-$(k-1)$ itemset. Finally, the association rules which satisfy the minimum confidence value will be output.

```
1)  $L_1$ = {large 1-itemsets};
2)  **for** ( $k = 2$; $L_{k-1} \neq \emptyset$; $k$++ ) **do begin**
3)      $C_k$ = apriori-gen($L_{k-1}$); // New candidates
4)      **forall** transactions $t \in \mathcal{D}$ **do begin**
5)          $C_t$ = subset($C_k$, $t$); // Candidates contained in $t$
6)          **forall** candidates $c \in C_t$ **do**
7)              $c$.count++;
8)      **end**
9)      $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$
10) **end**
11) Answer = $\bigcup_k L_k$;
```

Figure 1: The Apriori Algorithm

[*]This is a final project proposal for COMS 4995 Parallel Functional Programming, Fall 2021.

```
**insert into** $C_k$
**select** $p.\text{item}_1$, $p.\text{item}_2$, ..., $p.\text{item}_{k-1}$, $q.\text{item}_{k-1}$
**from** $L_{k-1}$ $p$, $L_{k-1}$ $q$
**where** $p.\text{item}_1 = q.\text{item}_1, \ldots, p.\text{item}_{k-2} = q.\text{item}_{k-2}$,
    $p.\text{item}_{k-1} < q.\text{item}_{k-1}$;
```

Figure 2: `apriori-gen` Join Step

```
**forall** itemsets $c \in C_k$ **do**
  **forall** $(k-1)$-subsets $s$ of $c$ **do**
    **if** $(s \notin L_{k-1})$ **then**
      **delete** $c$ from $C_k$;
```

Figure 3: `apriori-gen` Prune Step

3 Objectives

The project will apply the Apriori algorithm on a $10,000$-row dataset and find the frequent itemsets based on some fixed minimum support and confidence value. I will first implement the non-parallel Haskell program based on the Python implementation written by me in the past. Then, I will make the implementation parallel and therefore achieve a better performance. My plan for now is to introduce parallelism in both the outer for loop in the Apriori algorithm and the candidate generation process. There are also a few other steps in the algorithm where filtering is required, which can be done in parallel as well.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In *Proc. of 20th intl. conf. on vldb* (pp. 487–499).