CMSC 691
High Performance Distributed Systems

Apache Hadoop: Apriori Algorithm

Dr. Alberto Cano
Assistant Professor
Department of Computer Science
acano@vcu.edu

Frequent itemset

- Given a set of transactions, find combinations of items that occur frequently in a database
- **K-itemset**: a set of k items
- **Support:** frequency of occurrence of an itemset (absolute, relative)
- Frequent itemset when its support is greater than a threshold

Market-Basket transactions

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Examples of frequent itemsets

{Diaper, Beer} support = 3/5
{Milk, Bread} support = 3/5
{Milk, Beer, Bread} support = 2/5

Importance of frequent itemsets

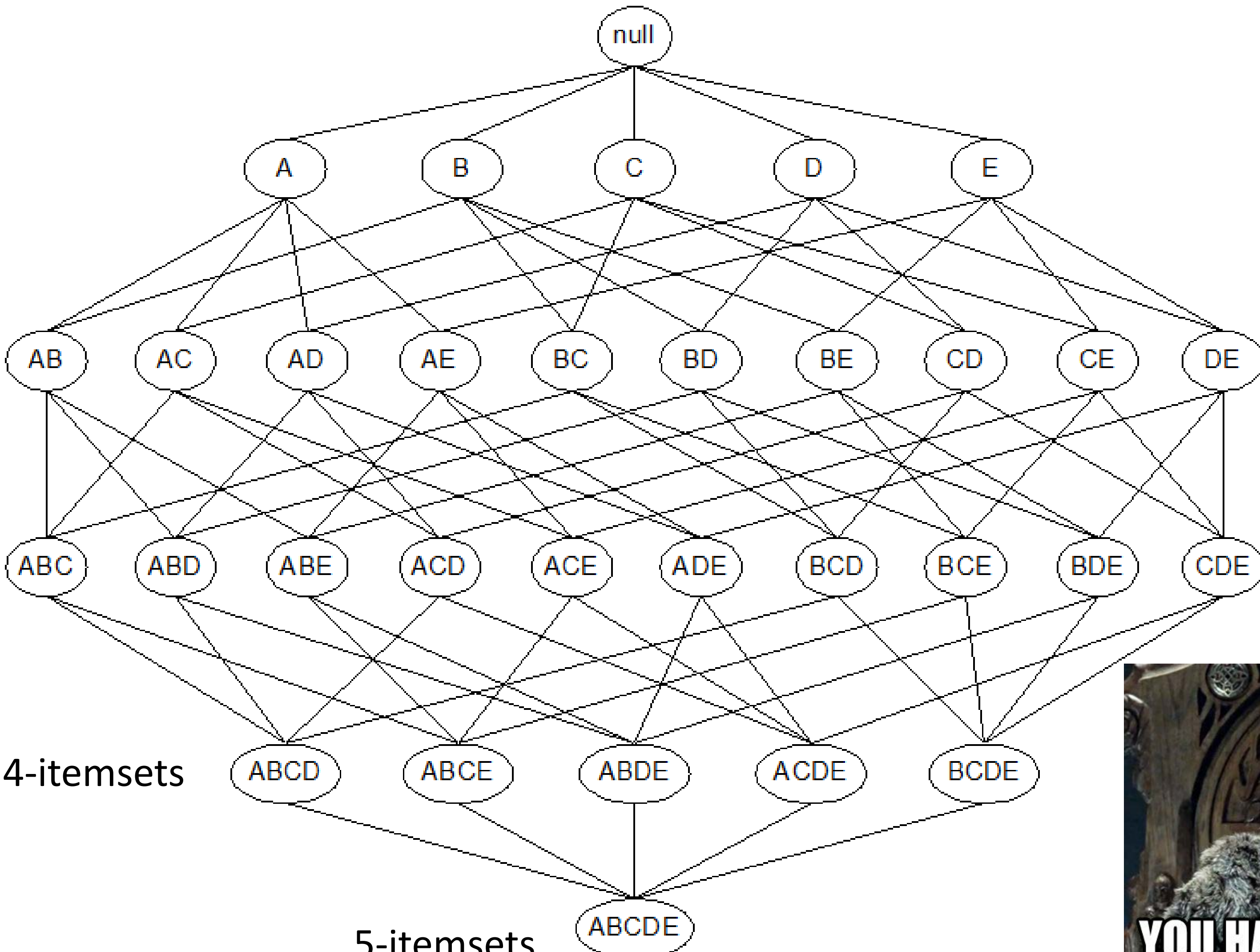
- Find all combinations of items that occur together
- $ tons of $ in business analytics, e.g. placement of items in a store
- Frequent itemsets are only positive combinations, we do not report combinations that do not occur frequently together
- Frequent itemsets provide a summary of the data
- **Task**: Given a transactions database and a minimum support threshold find all frequent itemsets
- **Issue**: how many itemsets may happen in a database? Given d items, there are 2^d possible itemsets. How to scale to big data? Considering *only* 100 features, 2^{100} is … infinite. How many items in Walmart transactions database?

Brute-force algorithm for finding all frequent itemsets

1. Generate all possible itemsets from 1-itemsets to d-itemsets
2. Compute the support of each itemset
3. If the support is greater than the minimum support then add it to the collection of frequent itemsets

- Computational complexity: 2^d
- Good luck!
- Definitely we need faster ways

Do not underestimate the exponential complexity



Reducing the number of candidates

- If an itemset is frequent **then** all of its subsets are also frequent
- The support of an itemset never exceeds the support of its subsets
- This is known as the anti-monotone property of support

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

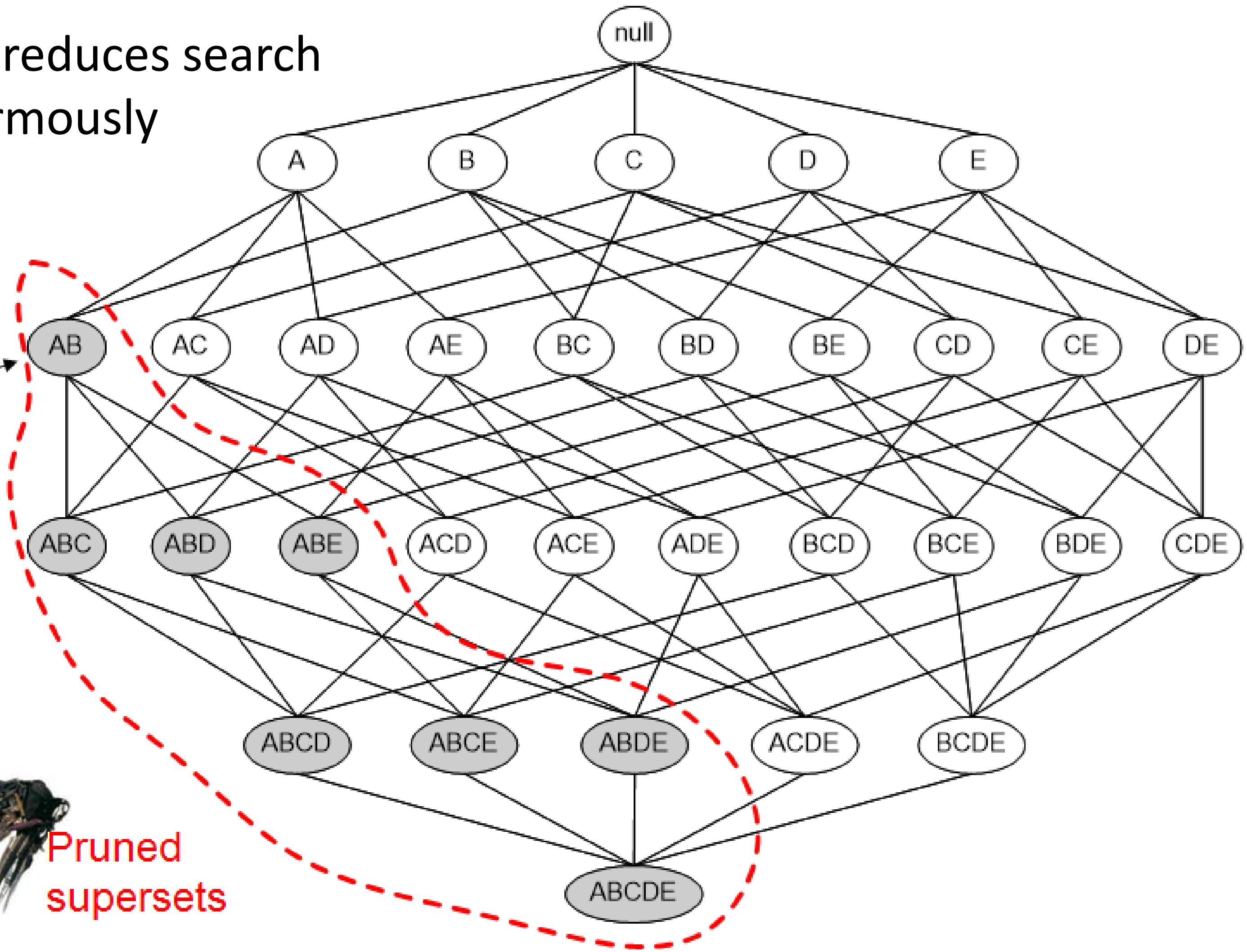
TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

s(Bread) > s(Bread, Beer)
s(Milk) > s(Bread, Milk)
s(Diaper, Beer) > s(Diaper, Beer, Coke)

Also:
s(A,B) <= MIN(s(A), s(B)) // Intersec

Pruning supersets

- Still 2^d but reduces search space enormously



Apriori algorithm

1. Find frequent 1-items and put them to $\mathbf{L_k}$ (**k=1**)
2. Use $\mathbf{L_k}$ to generate a collection of *candidate* itemsets $\mathbf{C_{k+1}}$ with size (**k+1**)
3. Scan the database to find which itemsets in $\mathbf{C_{k+1}}$ are frequent and put them into $\mathbf{L_{k+1}}$
4. If $\mathbf{L_{k+1}}$ is not empty

   k=k+1

   Go to step 2
5. All frequent k-itemsets are in $\mathbf{L_k}$ sets

- It makes multiple passes over the dataset, one pass for every level k
- Multiple passes over the dataset is inefficient when we have thousands of candidates and millions of transactions

Apriori algorithm

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)

minsup = 3/5

Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

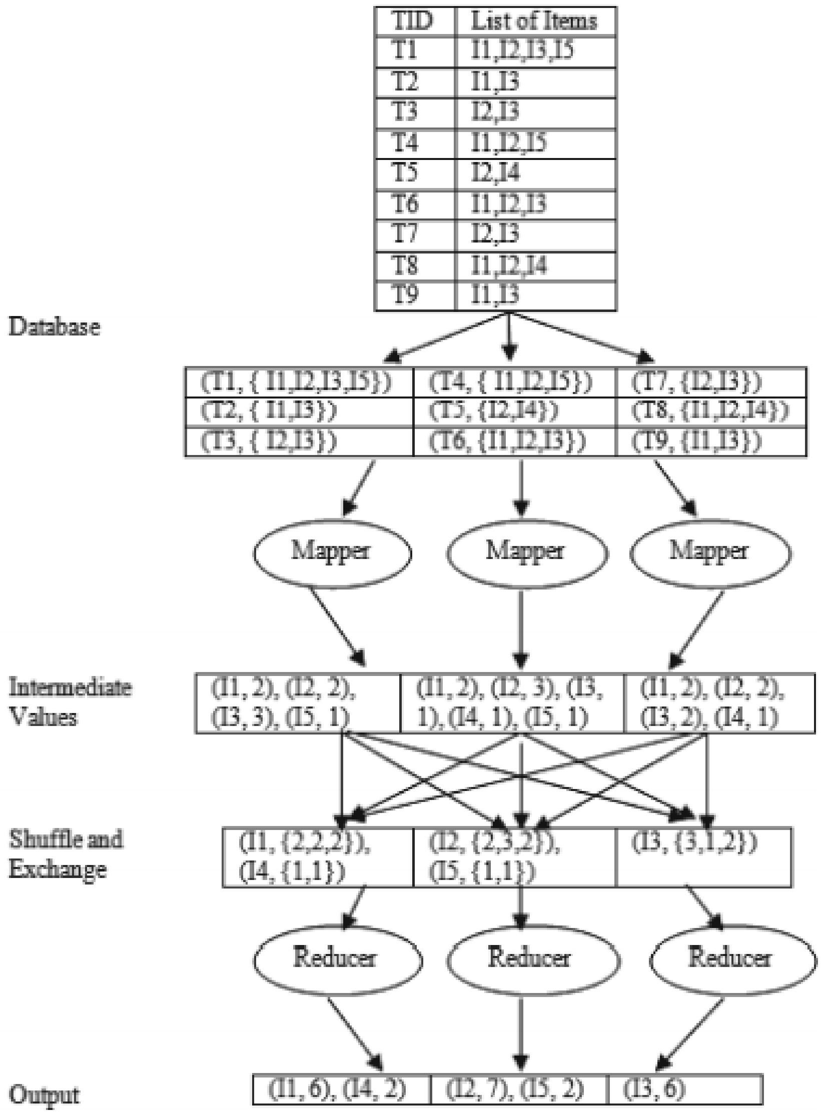
Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

Apache Hadoop Apriori



Generation of frequent 1-itemsets.

Association rules

- Given a set of transactions, find rules that will ***imply*** the occurrence of an item (or a set of items) based on the occurrences of other items in the transaction
- Implication: IF antecedent THEN consequent
- CANNOT be reversed

Market-Basket transactions

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Examples of association rules

{Diaper} -> {Beer}
{Milk, Bread} -> {Diaper, Coke}
{Beer, Bread} -> {Milk}

Association rules

- **Support**: fraction of transactions that contain both the antecedent and consequent

  support (A -> C) = support (A U C)

- **Confidence**: determines how frequently items in the consequent appear in transactions that contain the antecedent

  confidence (A -> C) = support (A U C) / support (A)

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

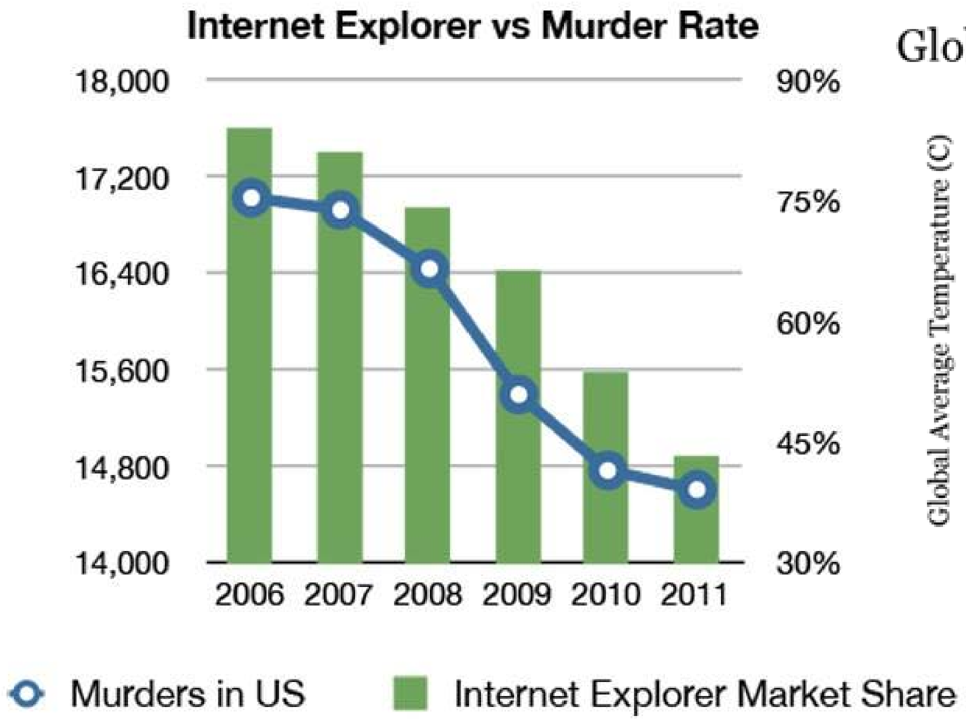
Rule: {Milk, Diaper} -> {Beer}

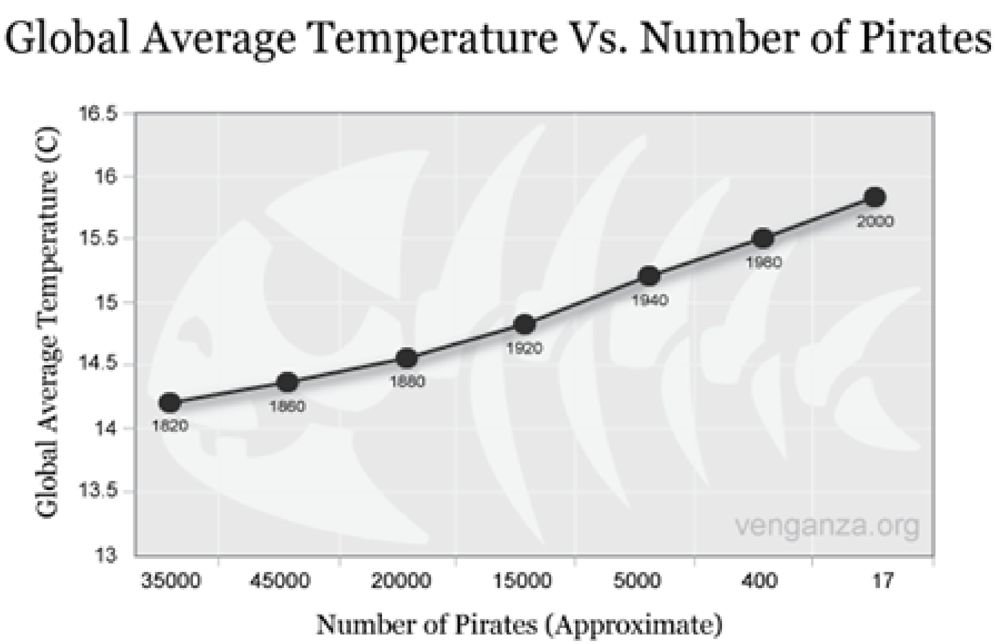
s(A) = 3/5
s(C) = 4/5

s(A -> C) = 2/5 c(A -> C) = 2/3

But remember!

- **Correlation does not imply causation**
- We find implications due to data correlations, subject to the interpretation of an expert in the area to verify the causation



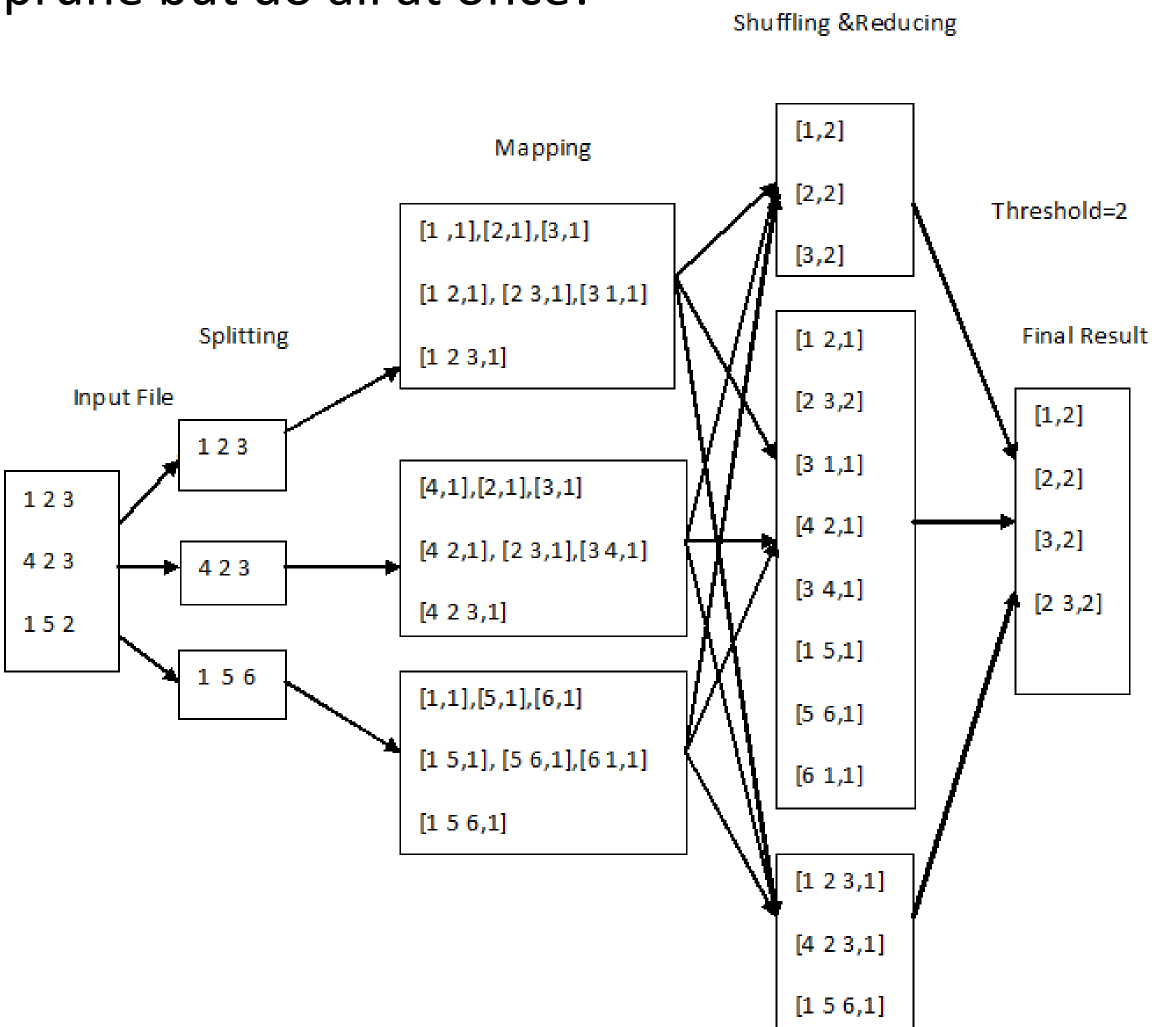


Extracting association rules (have a look at my friend's papers!)

- Given a set of transactions, the goal of association rule mining is to find all rules having:
  - support ≥ minsup threshold
  - confidence ≥ minconf threshold
- Brute-force algorithm: list all possible association rules, compute the support and confidence for each rule, prune rules that fail the minsup and minconf thresholds.
- Total number of possible association rules: $3^d - 2^{d+1} + 1$
- Generate high confidence rules from the frequent itemsets, where each rule is a binary partition of a frequent itemset

Educative implementation of Apriori and Association rules

- Doesn't prune but do all at once!



CMSC 691
High Performance Distributed Systems

Apache Hadoop: Apriori Algorithm

Dr. Alberto Cano
Assistant Professor
Department of Computer Science
acano@vcu.edu