High Performance Data Mining

Chapter 4: Association Rules

Vipin Kumar

Army High Performance Computing Research Center
Department of Computer Science
University of Minnesota

http://www.cs.umn.edu/~kumar

Chapter 4: Algorithms for Association Rules Discovery

Outline

- **Serial Association Rule Discovery**
  - Definition and Complexity.
  - Apriori Algorithm.
- **Parallel Algorithms**
  - Need
  - Count Distribution, Data Distribution
  - Intelligent Data Distribution, Hybrid Distribution
  - Experimental Results

Association Rule Discovery: Support and Confidence

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

Association Rule: $X \Rightarrow_{s,\alpha} y$

Support: $s = \frac{\sigma(X \cup y)}{|T|} \quad (s = P(X, y))$

Confidence: $\alpha = \frac{\sigma(X \cup y)}{\sigma(X)|} \quad (\alpha = P(y \mid X))$

Example:

$\{\text{Diaper}, \text{Milk}\} \Rightarrow_{s,\alpha} \text{Beer}$

$$s = \frac{\sigma(\text{Diaper}, \text{Milk}, \text{Beer})}{\text{Total Number of Transactions}} = \frac{2}{5} = 0.4$$

$$\alpha = \frac{\sigma(\text{Diaper}, \text{Milk}, \text{Beer})}{\sigma(\text{Diaper}, \text{Milk})|} = 0.66$$

Handling Exponential Complexity

- Given *n* transactions and *m* different items:
  - number of possible association rules: $O(m2^{m-1})$
  - computation complexity: $O(nm2^m)$
- Systematic search for all patterns, based on support constraint [Agarwal & Srikant]:
  - If {A,B} has support at least α, then both A and B have support at least α.
  - If either A or B has support less than α, then {A,B} has support less than α.
  - Use patterns of n-1 items to find patterns of n items.

Apriori Principle

- Collect single item counts. Find large items.
- Find candidate pairs, count them => large *pairs* of items.
- Find candidate triplets, count them => large *triplets* of items, And so on...
- Guiding Principle: *Every subset of a frequent itemset has to be frequent.*
  - **Used for pruning many candidates.**

Illustrating Apriori Principle

Items (1-itemsets)

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

Pairs (2-itemsets)

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

Minimum Support = 3

Triplets (3-itemsets)

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

If every subset is considered,

$$^6C_1 + {}^6C_2 + {}^6C_3 = 41$$

With support-based pruning,

$$6 + 6 + 2 = 14$$

Apriori Algorithm

```
F_1 = {frequent 1-item sets};
k = 2;
while( F_k-1 is not empty ) {
        C_k = Apriori_generate( F_k-1 );
        for all transactions t in T {
                Subset( C_k, t );
        }
        F_k = { c in C_k s.t. c.count >= minimum_support};
}
Answer = union of all sets F_k;
```

Association Rule Discovery: Apriori_generate

Apriori_generate(F(k-1)) {

join F_{k-1} with F_{k-1} such that,

$c_1 = (i_1, i_2, .., i_{k-1})$ and $c_2 = (j_1, j_2, .., j_{k-1})$ join together if

$i_p = j_p$ for $1 <= p <= k-1$,

and then new candidate, c, has a form

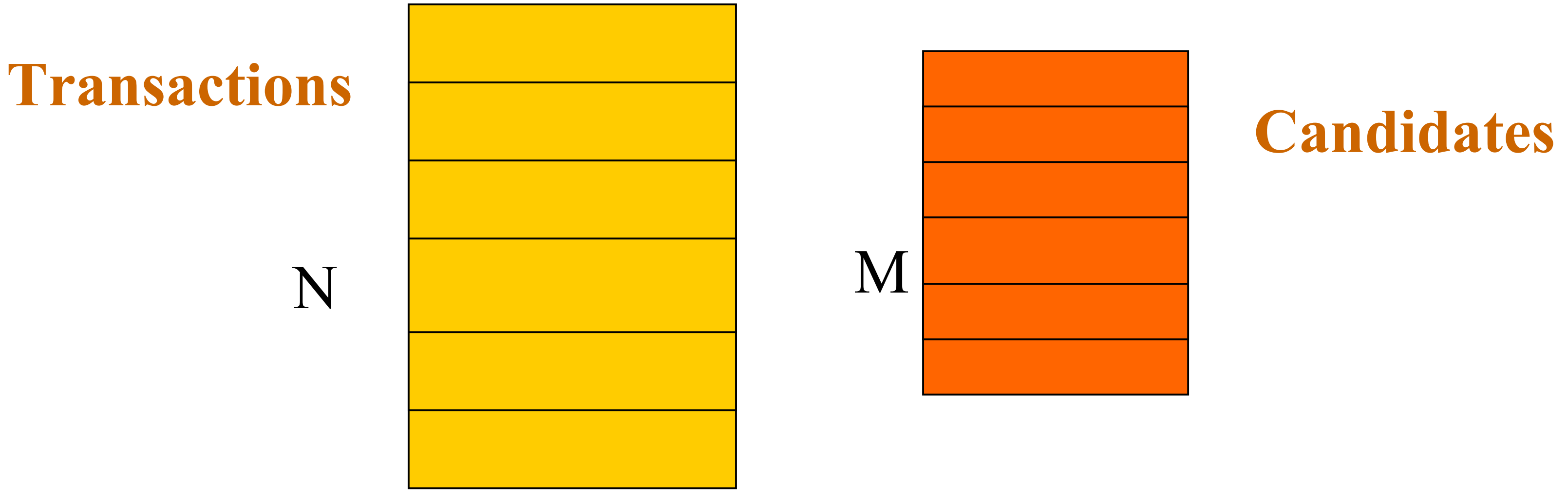
$c = (i_1, i_2, .., i_{k-1}, j_{k-1})$.

c is then added to a *hash-tree* structure.

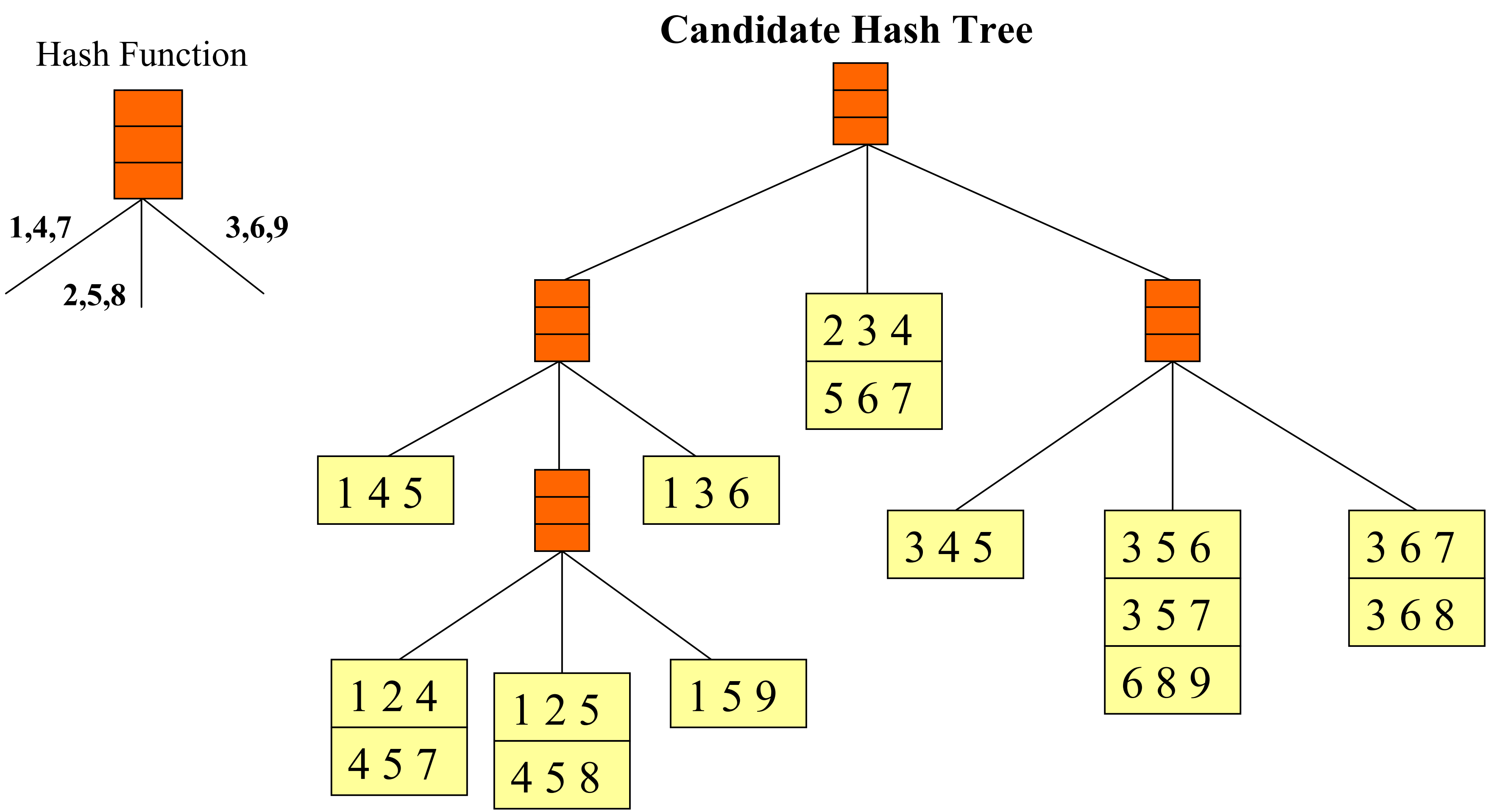
}

Counting Candidates

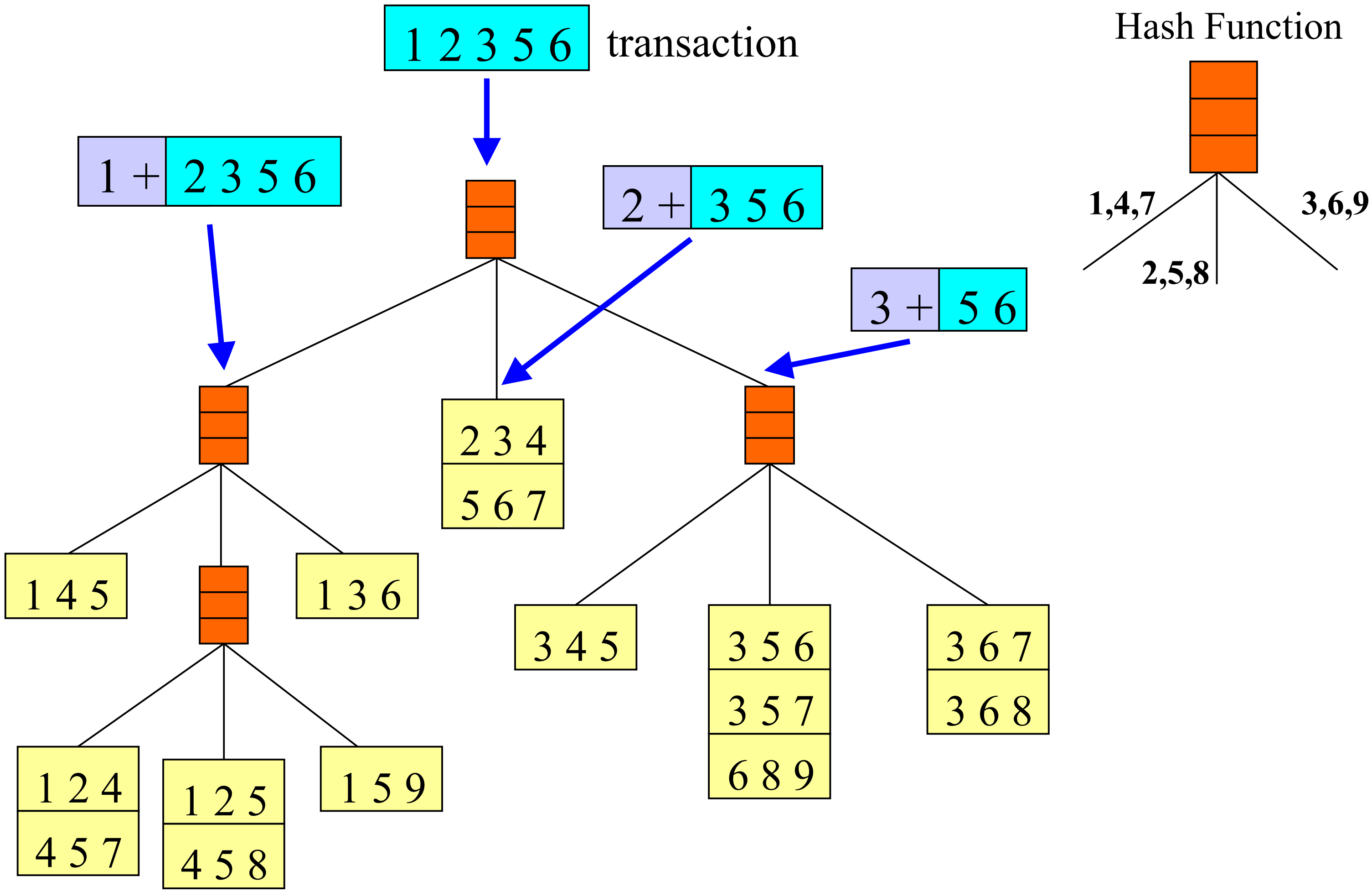
- Frequent Itemsets are found by counting candidates.
- Simple way:
  - Search for each candidate in each transaction. Expensive!!!



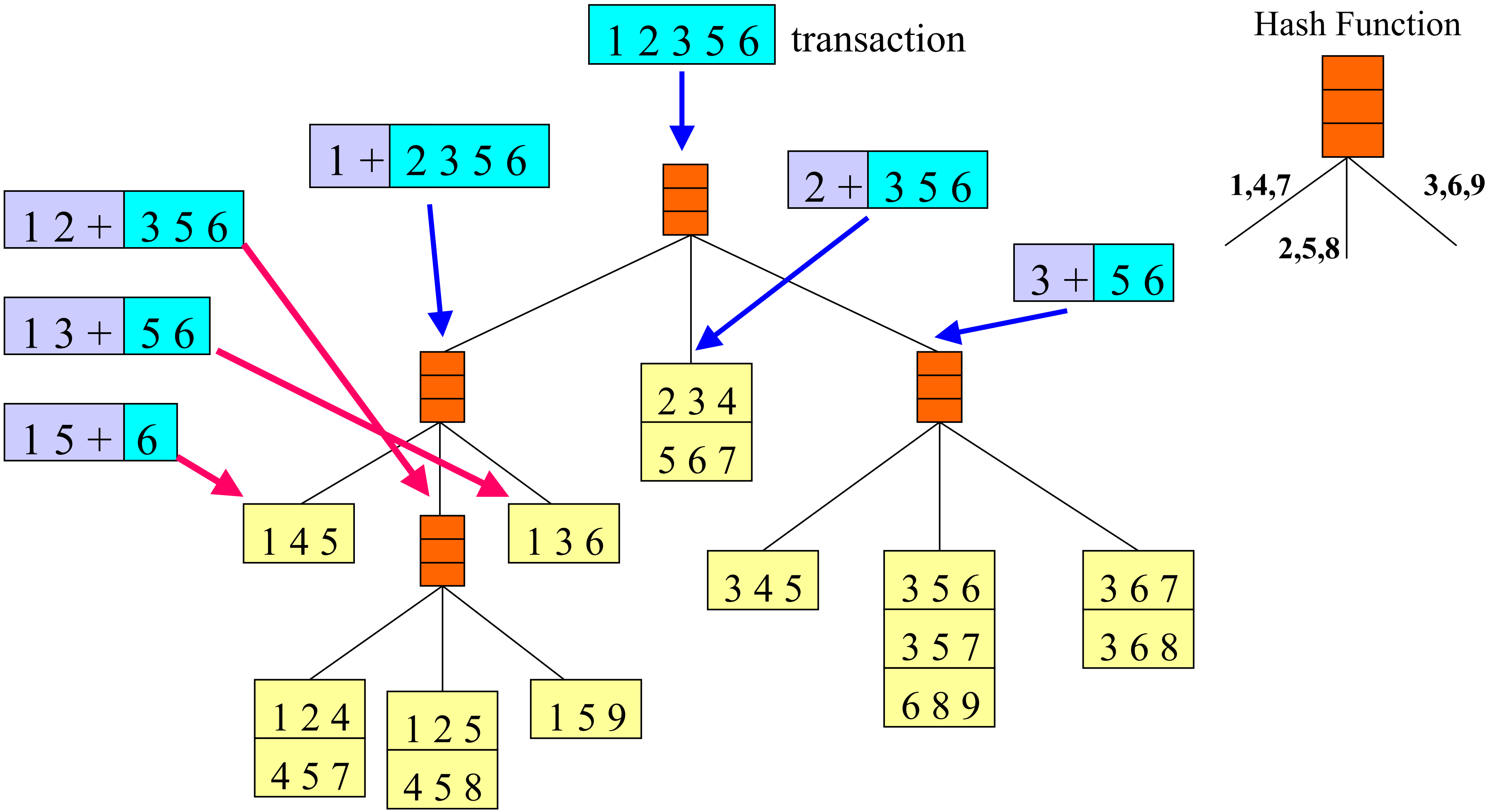
Association Rule Discovery: Hash tree for fast access.



Association Rule Discovery: Subset Operation



Association Rule Discovery: Subset Operation (contd.)



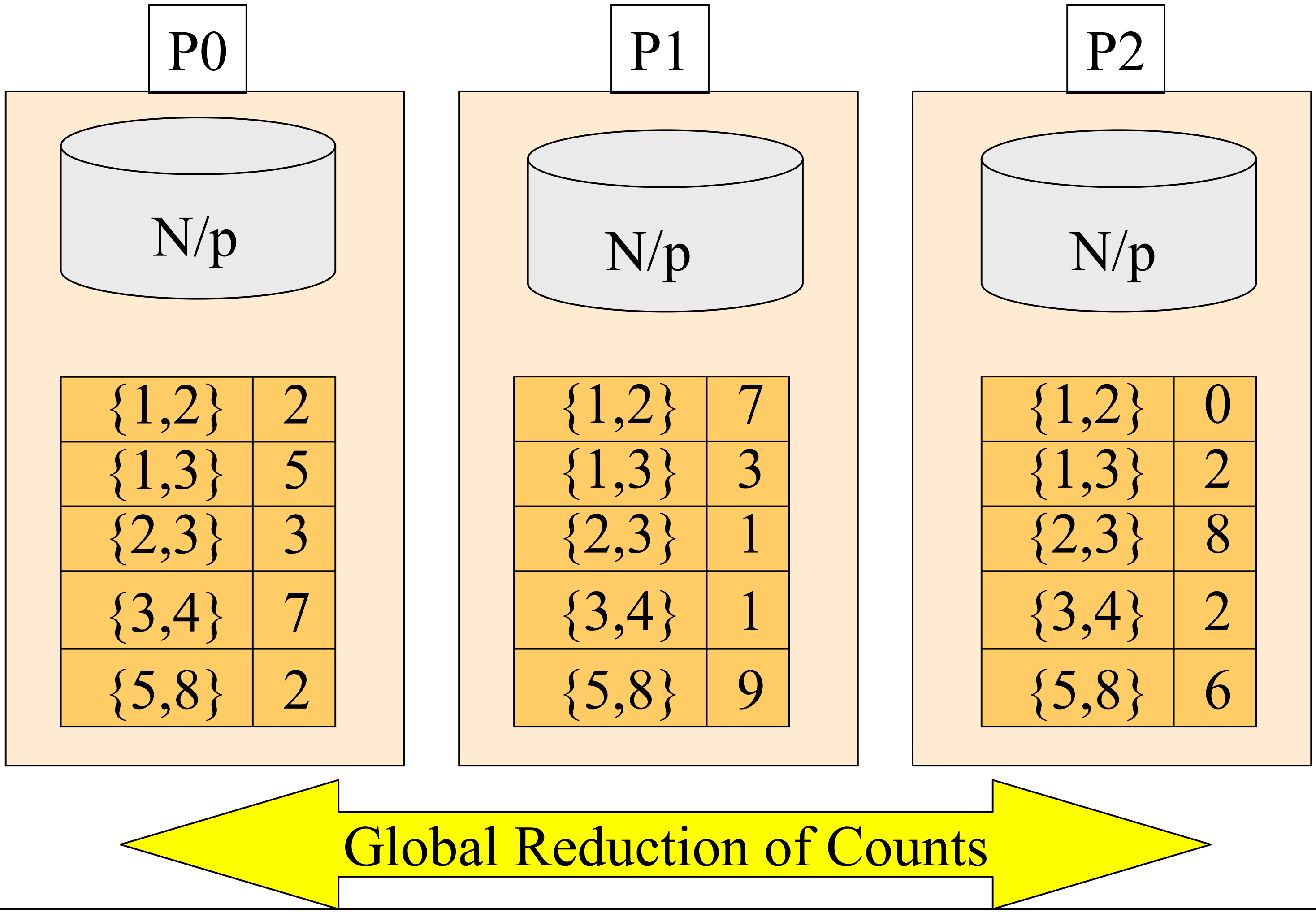
Parallel Formulation of Association Rules

- Need:
  - Huge Transaction Datasets (10s of TB)
  - Large Number of Candidates.
- Data Distribution:
  - Partition the Transaction Database, or
  - Partition the Candidates, or
  - Both

Parallel Association Rules: Count Distribution (CD)

- Each Processor has complete candidate hash tree.
- Each Processor updates its hash tree with local data.
- Each Processor participates in global reduction to get global counts of candidates in the hash tree.
- Multiple database scans are required if the hash tree is too big to fit in the memory.

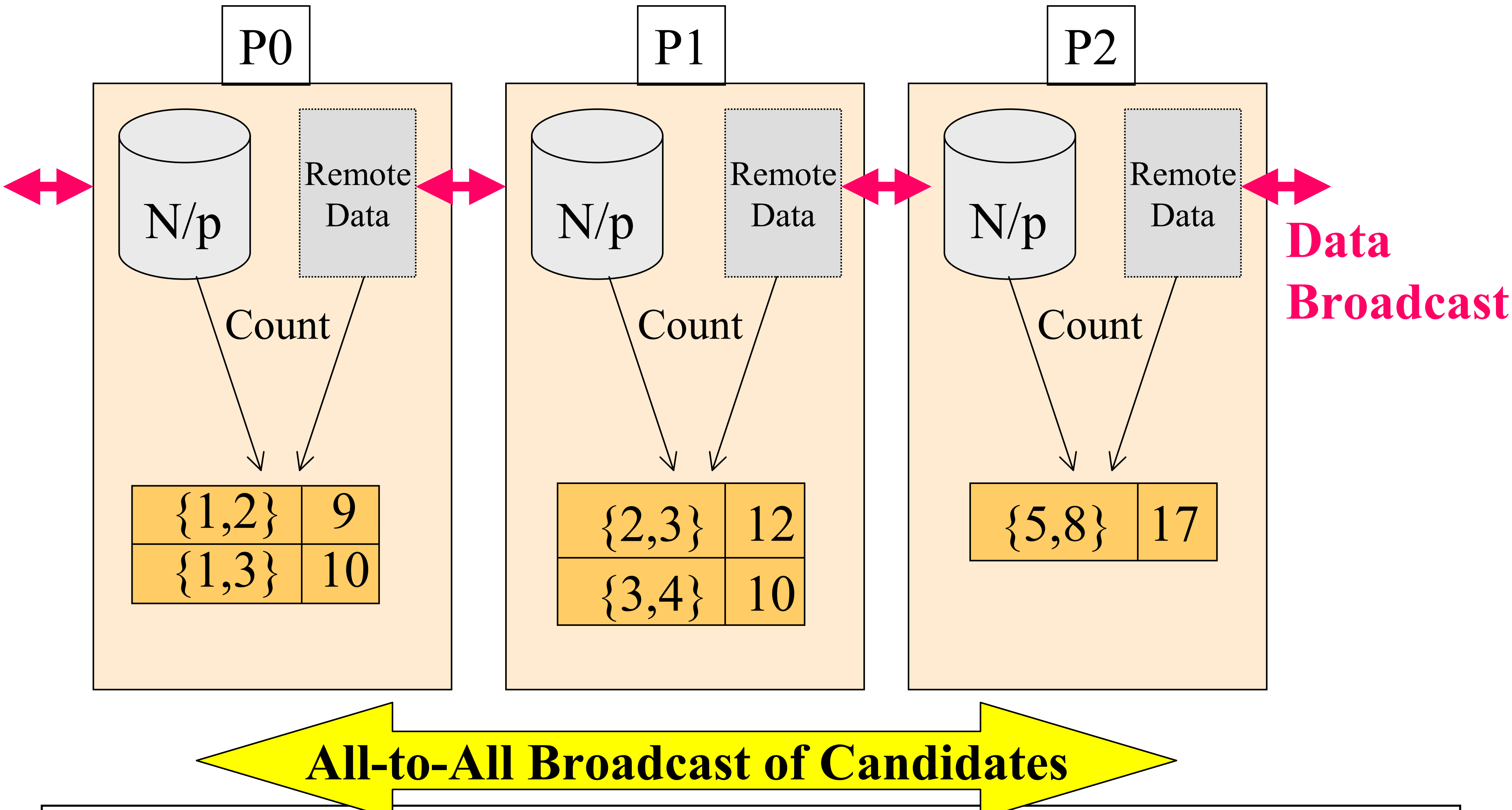
CD: Illustration



Parallel Association Rules: Data Distribution (DD)

- Candidate set is partitioned among the processors.
- Once local data has been partitioned, it is broadcast to all other processors.
- High Communication Cost due to data movement.
- Redundant work due to multiple traversals of the hash trees.

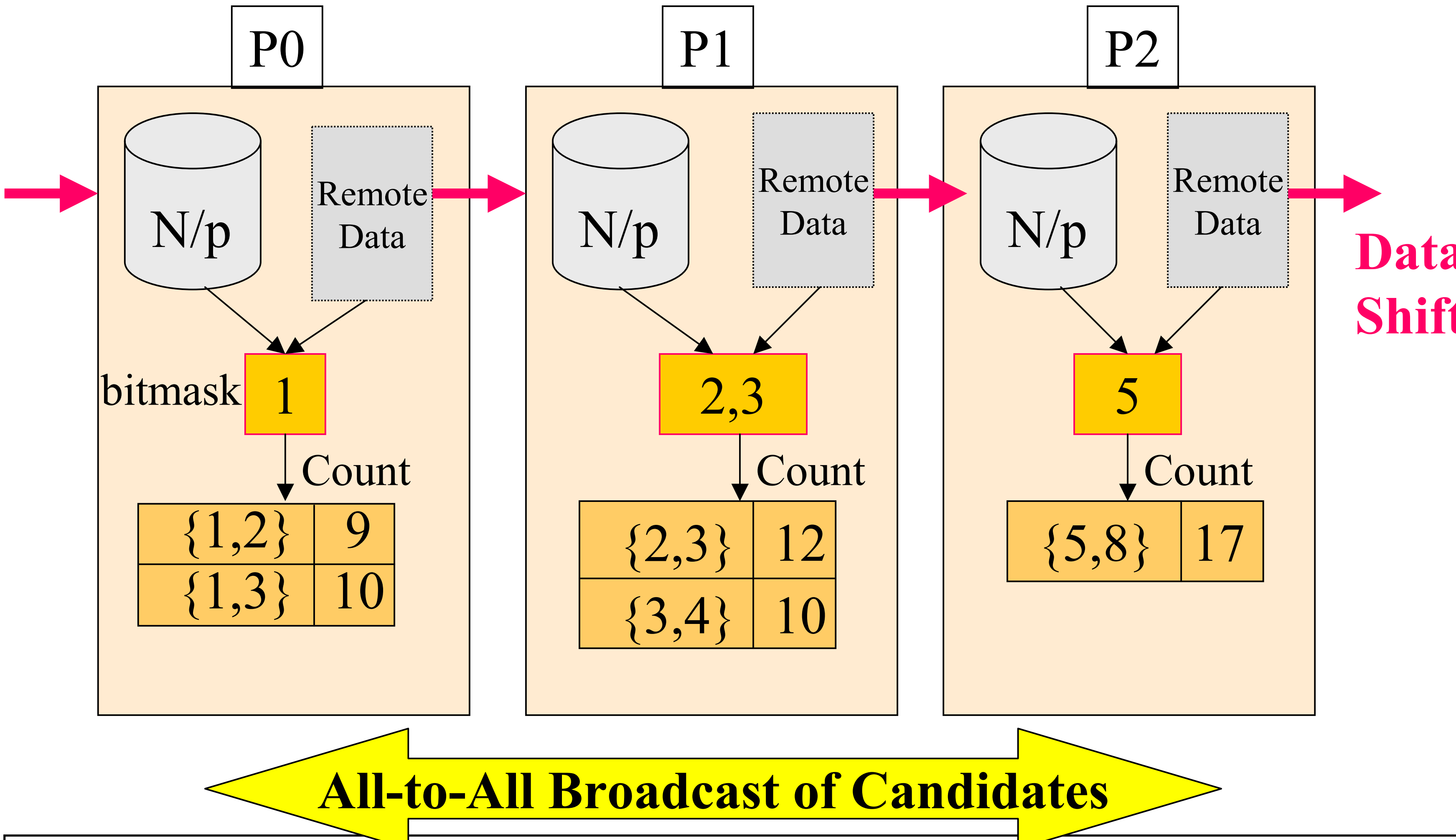
DD: Illustration



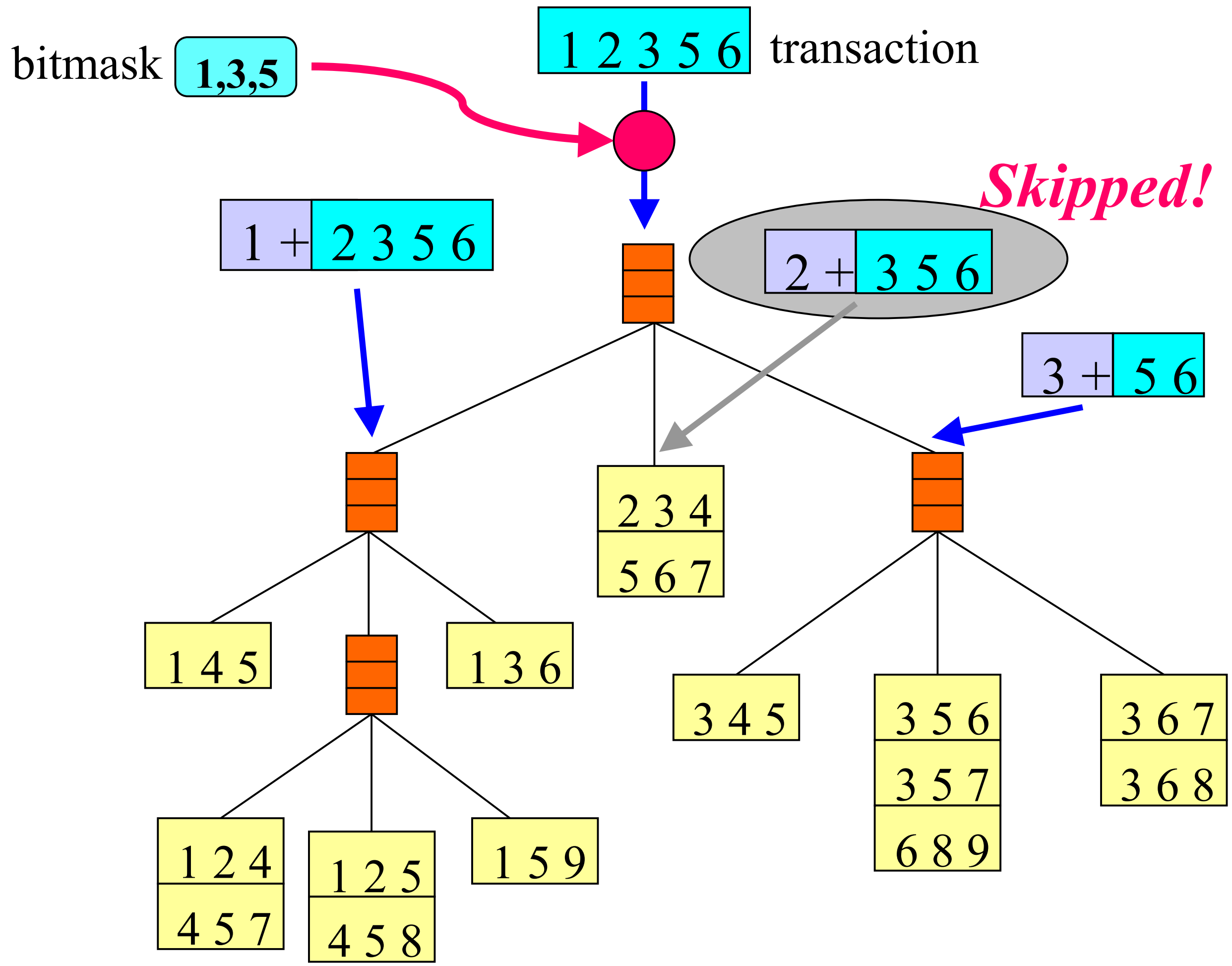
Parallel Association Rules: Intelligent Data Distribution (IDD)

- Data Distribution using point-to-point communication.
- Intelligent partitioning of candidate sets.
  - Partitioning based on the first item of candidates.
  - Bitmap to keep track of local candidate items.
- Pruning at the root of candidate hash tree using the bitmap.
- Suitable for single data source such as database server.
- With smaller candidate set, load balancing is difficult.

IDD: Illustration



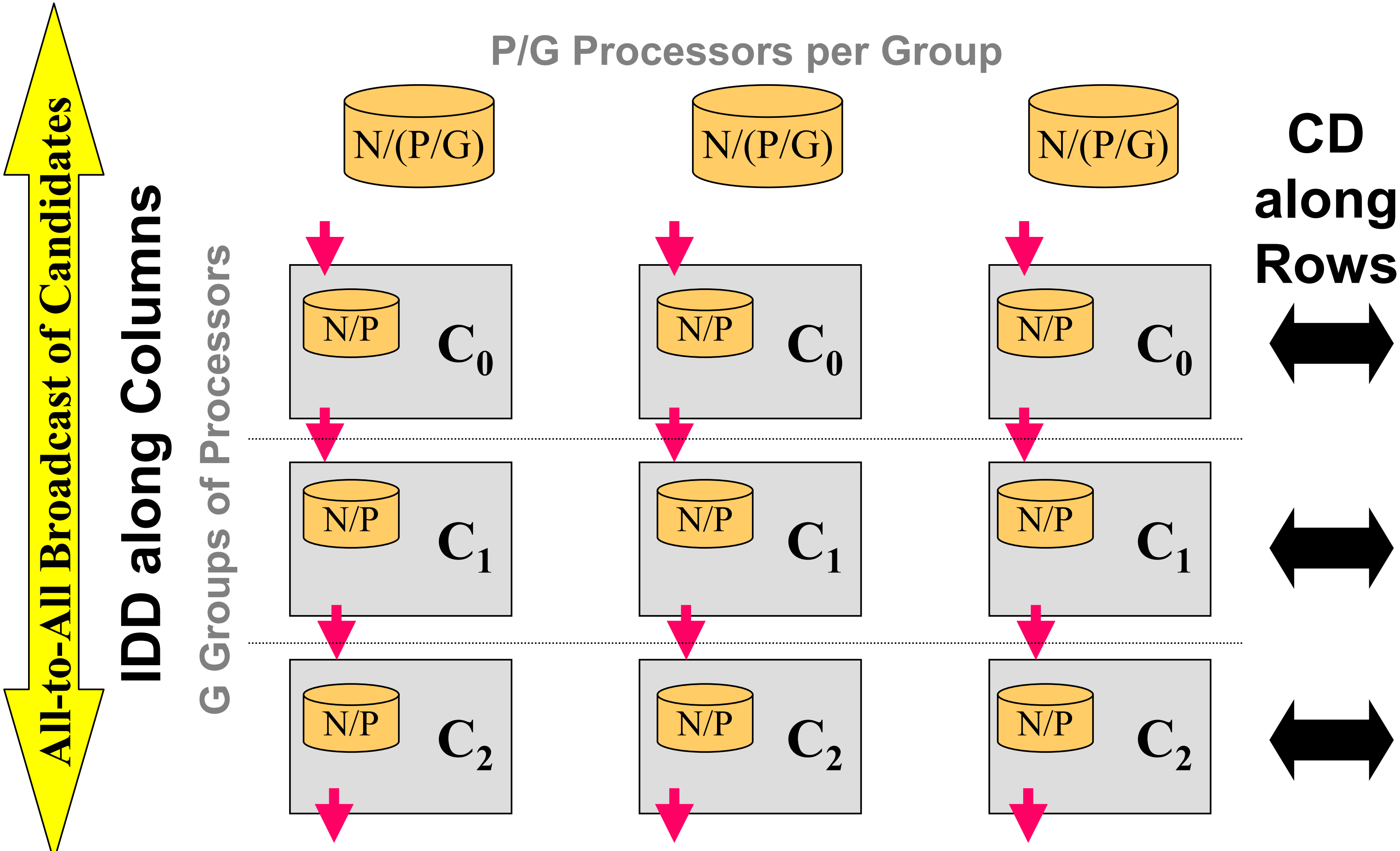
Filtering Transactions in IDD



Parallel Association Rules: Hybrid Distribution (HD)

- Candidate set is partitioned into G groups to just fit in main memory
  - ***Ensures Good load balance with smaller candidate set.***
- Logical processor mesh G x P/G is formed.
- Perform IDD along the column processors
  - ***Data movement among processors is minimized.***
- Perform CD along the row processors
  - ***Smaller number of processors is global reduction operation.***

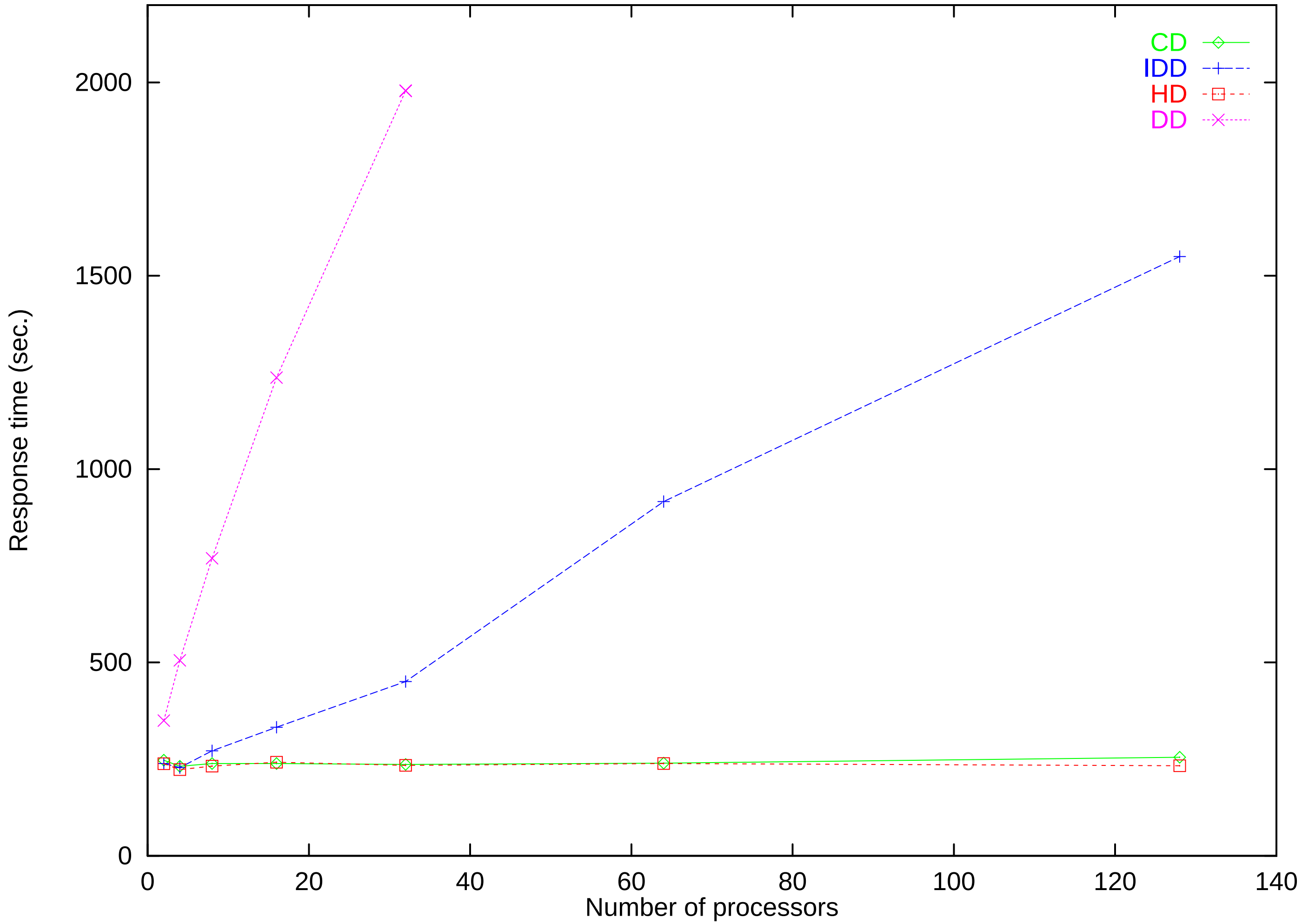
HD: Illustration



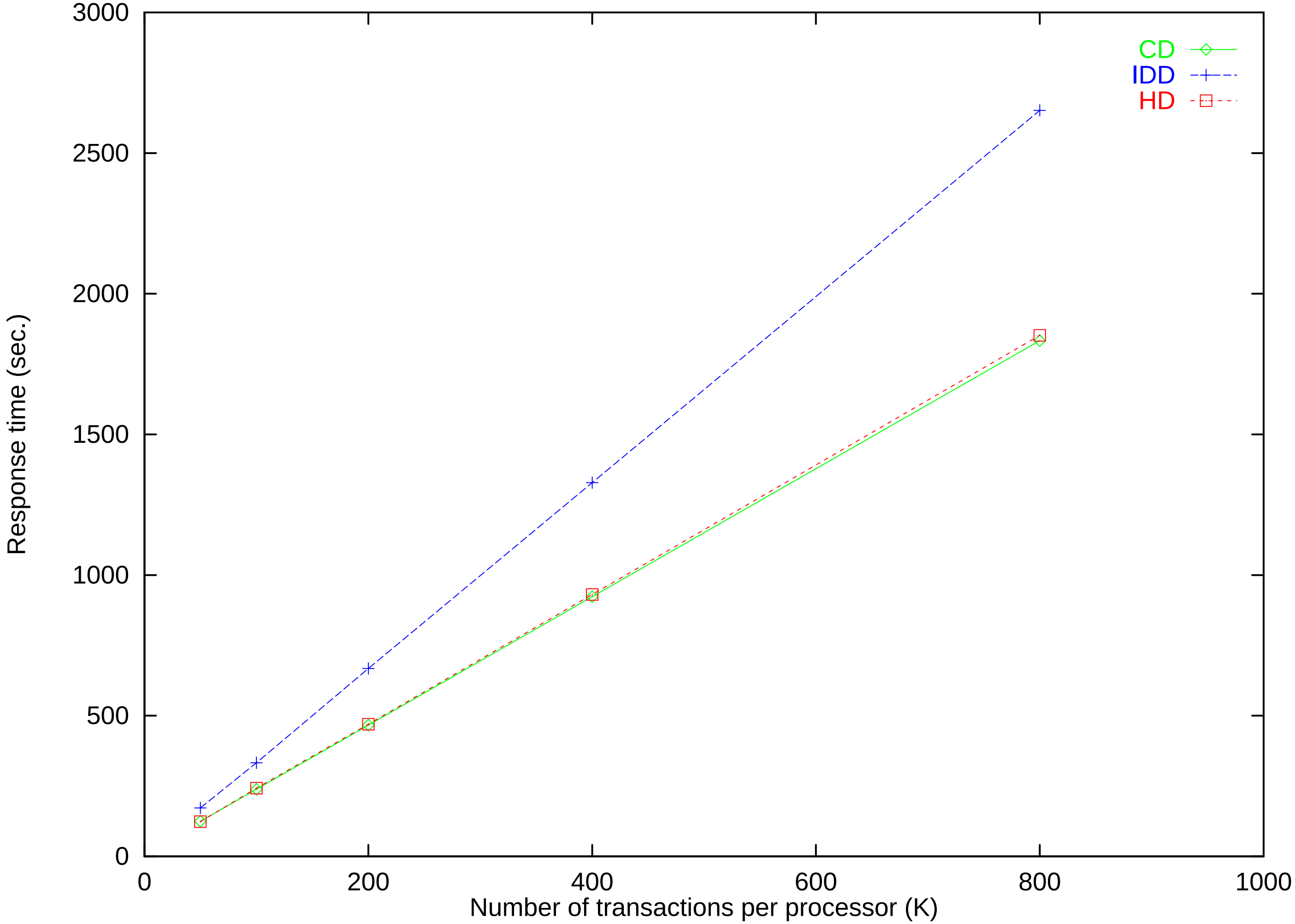
Parallel Association Rules: Experimental Setup

- 128-processor Cray T3D
  - 150 MHz DEC Alpha (EV4)
  - 64 MB of main memory per processor
  - 3-D torus interconnection network with peak unidirectional bandwidth of 150 MB/sec.
- MPI used for communications.
- Synthetic data set: avg transaction size 15 and 1000 distinct items.
- For larger data sets, multiple read of transactions in blocks of 1000.
- HD switch to CD after 90.7% of the total computation is done.

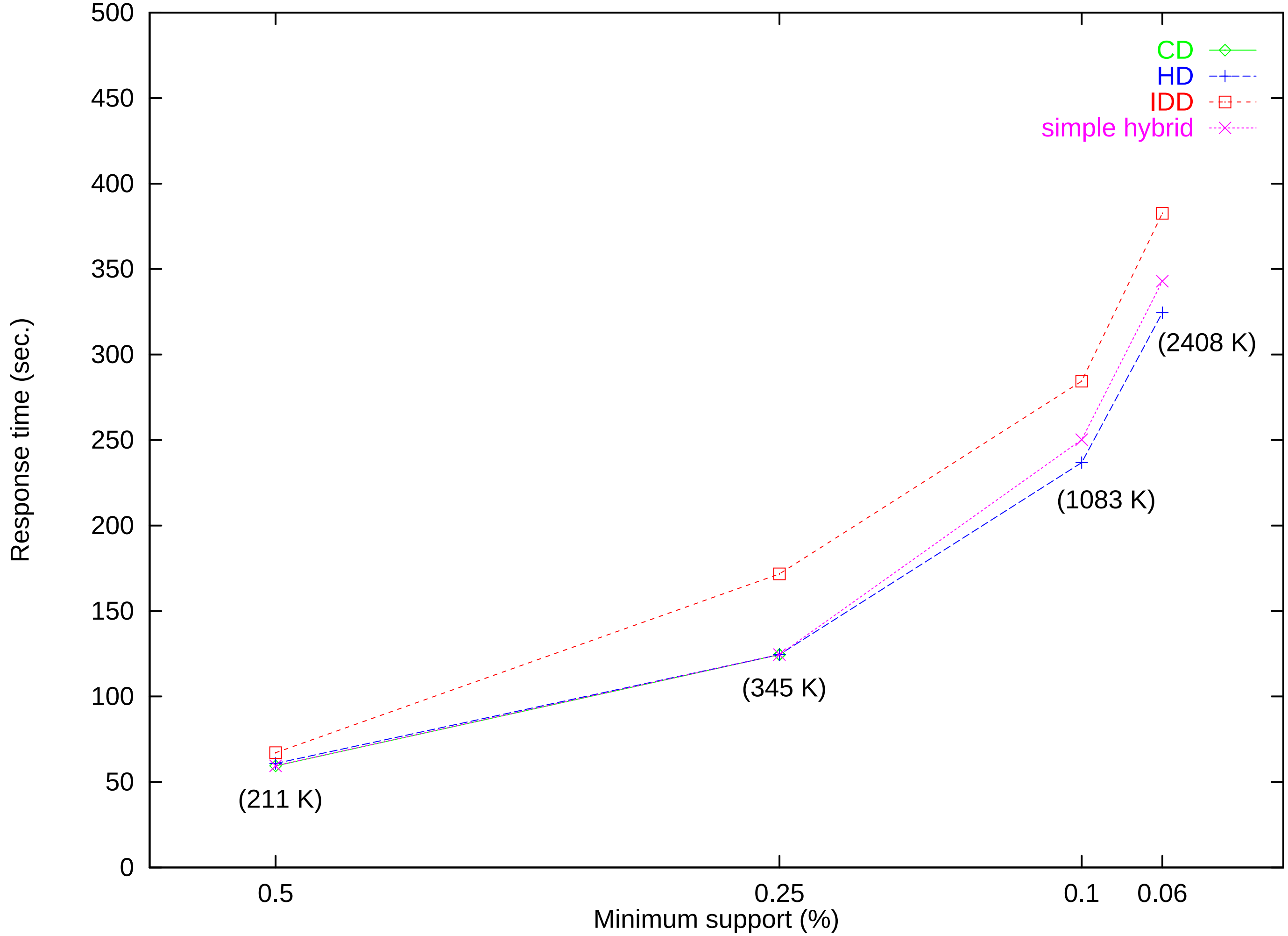
Parallel Association Rules: Scaleup Results (100K,0.25%)



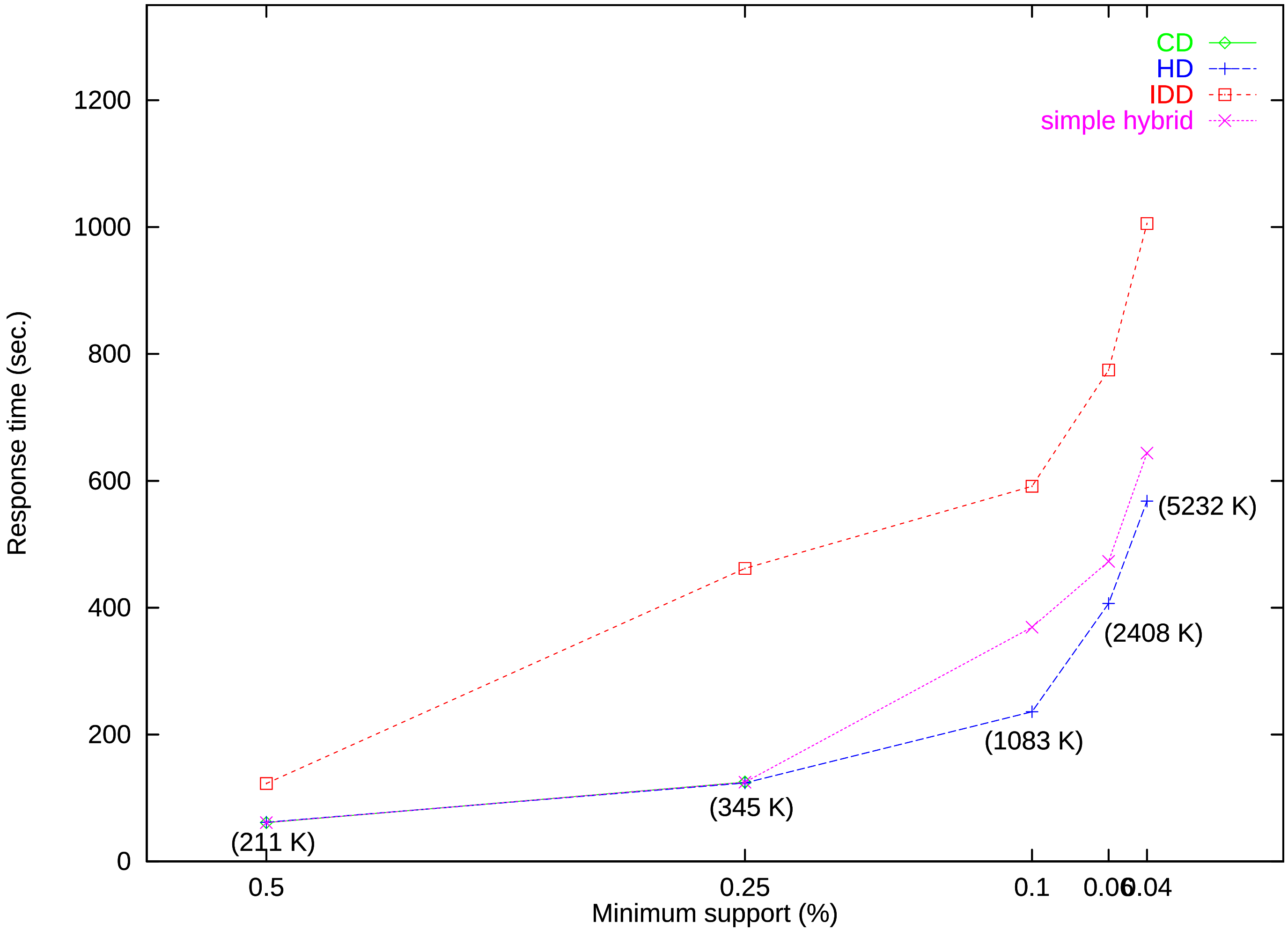
Parallel Association Rules: Sizeup Results (np=16,0.25%)



Parallel Association Rules: Response Time (np=16,50K)



Parallel Association Rules: Response Time (np=64,50K)



Parallel Association Rules: Minimum Support Reachable

Number of Processors	**1**	**2**	**4**	**8**	**16**	**32**	**64**
Successful Down to	**0.25**	**0.2**	**0.15**	**0.1**	**0.06**	**0.04**	**0.03**
Ran out of memory at	**0.2**	**0.15**	**0.1**	**0.06**	**0.04**	**0.03**	**0.02**

Parallel Association Rules: Processor Configuration in HD

64 Processors and 0.04 minimum support

Pass	2	3	4	5	6	7	8
Configuration	8 x 8	64 x 1	4 x 16	2 x 32	2 x 32	2 x 32	2 x 32
# of Candidates	351 K	4348 K	115 K	76 K	56 K	34 K	16 K

Parallel Association Rules: Summary of Experiments

- HD shows the same linear speedup and sizeup behavior as that of CD.
- HD Exploits Total Aggregate Main Memory, while CD does not.
- IDD has much better scaleup behavior than DD