Heterogeneous Parallel Programming COMP4901D

Frequent Itemset Mining on the GPU

Overview

- Frequent item set mining
- GPU-based frequent item set mining

Frequent Itemset Mining (FIM)

Find groups of items, or *itemsets* where all items in a group frequently co-occur in a transaction.

Transaction ID	Items
1	*A, B, C, D*
2	*A, B, D*
3	*A, C, D*
4	*C, D*

Minimum support: 2

1-itemsets (frequent items):
A, B, C, D

2-itemsets:
AB, AC, AD, BD, CD

3-itemsets:
ABD, ACD

The Apriori Algorithm

Input: 1) Transaction Database 2) Minimum support
Output: All frequent itemsets

L_1 = {All frequent 1-itemsets}
k = 2
While (L_{k-1} != empty) {
  //Generate candidate k-itemsets
  Self-join //e.g. ***AB***C JOIN ***AB***D => ***AB***CD
  Subset test //e.g., Test all 3-subsets of ABCD
  //Generate frequent k-itemsets
  Support Counting
  k += 1
}

Frequent 1-itemsets

Candidate 2-itemsets

Frequent 2-itemsets

Candidate 3-itemsets

Frequent 3-itemsets

...

Candidate (K-1)-itemsets

Frequent (K-1)-itemsets

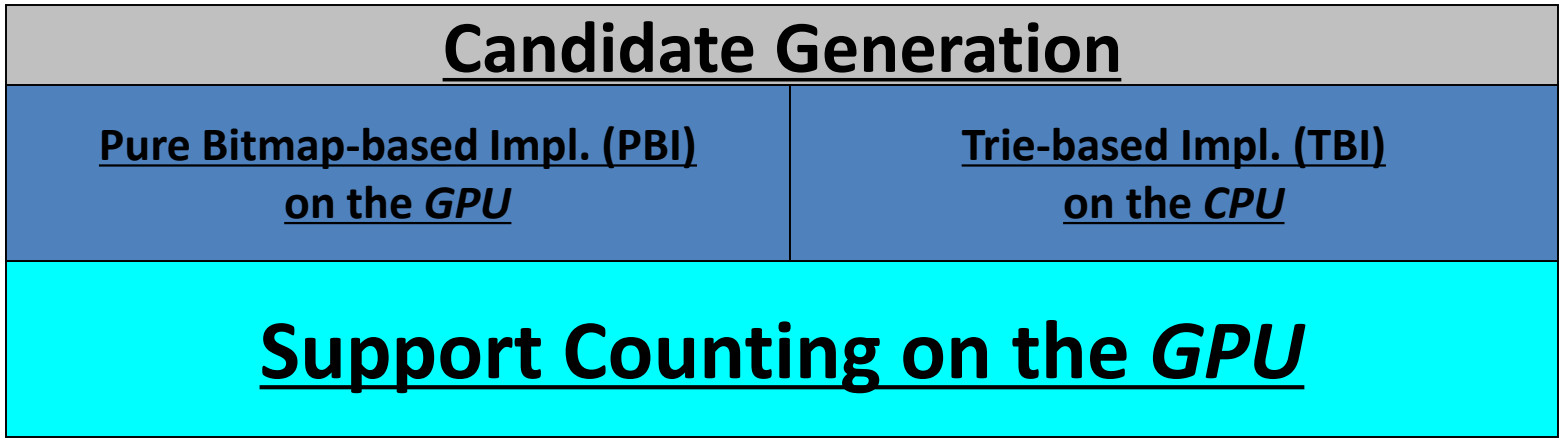
Candidate K-itemsets

Frequent K-itemsets

GPU Acceleration for APriori

Input: 1) Transaction Database 2) Minimum support
Output: All frequent itemsets

L_1 = {All frequent 1-itemsets}
k = 2
While (L_{k-1} != empty) {



    k += 1
}

Data Layout

Horizontal data layout

TID	Items
1	*A, B, C, D*
2	*A, B, D*
3	*A, C, D*
4	*C, D*

Vertical data layout

Item	Transactions
A	1,2,3
B	1,2
C	1,3,4
D	1,2,3,4

Horizontal layout requires scanning all transactions.

Pure Bitmap-based Implementation (PBI)

Bitmap representation for itemsets

	T1	T2	T3	T4
AB	1	1	0	0
AC	1	1	0	1
AD	1	0	0	1
BD	0	1	0	1
CD	0	0	1	1

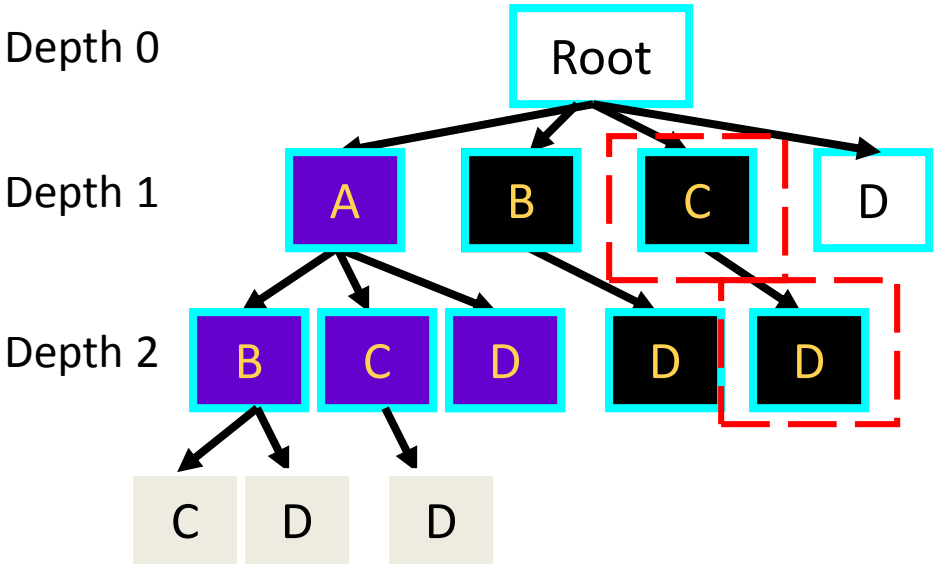
Bitwise OR In Join
(e.g., AB JOIN AD = ABD)

	T1	T2	T3	T4
ABD	1	1	0	1

Binary search
In Subset test
(e.g., 2-subsets {AB, AD, BD})

One GPU thread generates one candidate itemset.

Trie-based Impl. (TBI)



Candidate 3-itemsets: { ABD, ACD}

On CPU
Irregular memory access

1-itemsets: {A, B, C, D}

2-itemsets: {AB, AC, AD, BD, CD}

AB JOIN AC = ABC

{AB, AC, ~~BC~~}

AB JOIN AD = ABD

{AB, AD, BD}

AC JOIN AD = ACD

{AC, AD, CD}

Support Counting on the GPU

	T1	T2	T3	T4
AB	1	1	0	0
AC	1	0	1	0
AD	1	1	1	0
BD	1	1	1	0
CD	1	0	1	1

	T1	T2	T3	T4
ABD	1	1	0	0
ACD	1	0	1	0

Intersect = bitwise AND operation

1. Intersect
2. Count

Support Counting on the GPU (Cont.)

Bitmap representation for transactions

	T1	T2	T3	T4
ABD	1	1	0	0
ACD	1	0	1	0

Lookup table

Index	Count
0000	*0*
0001	*1*
…	
1000	*1*
1010	*2*
1011	*3*
1100	*2*

Constant memory

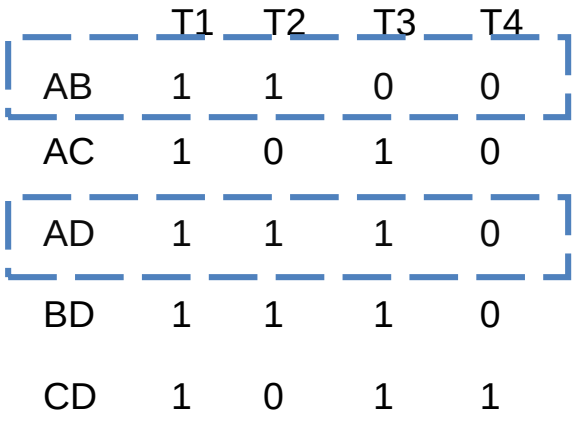
1. Cacheable
2. 64 KB
3. Shared by all GPU threads

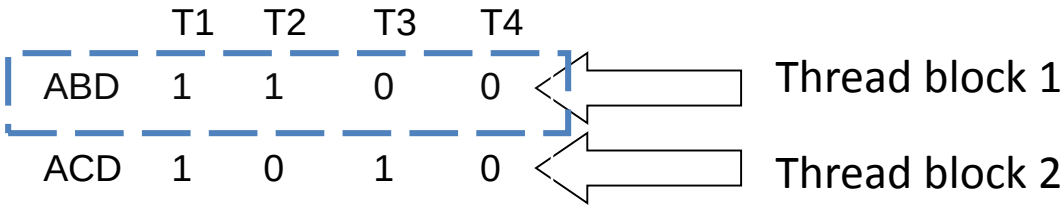
Index	Count
0000 0000 0000 0000 (0)	*0*
0000 0000 0000 0001 (1)	*1*
…	
1111 1111 1111 1110 (65534)	*15*
1111 1111 1111 1111 (65535)	*16*

1 byte

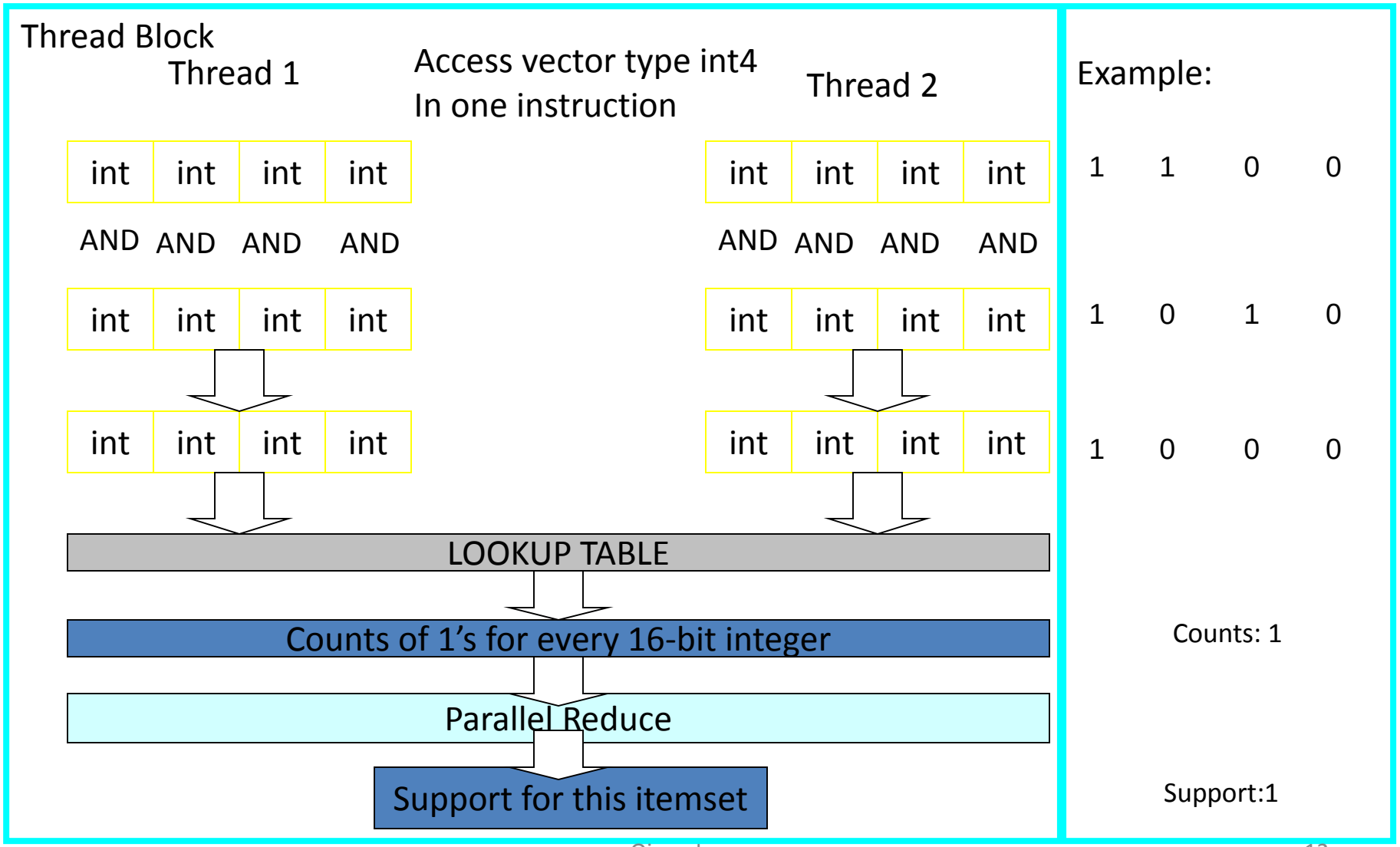
$2^{16} = 65536$

Support Counting on the GPU (Cont.)





Support Counting on the GPU (Cont.)



Experimental setup

Platform	Intel Core2 quad-core CPU	NV GTX 280 GPU
Processors	2.66 GHz * 4	**1.35 GHz * 30 * 8**
Memory Bandwidth (GB/sec)	10.4	**141.7**
Development Env.	Windows XP + Visual Studio 2005 + CUDA	

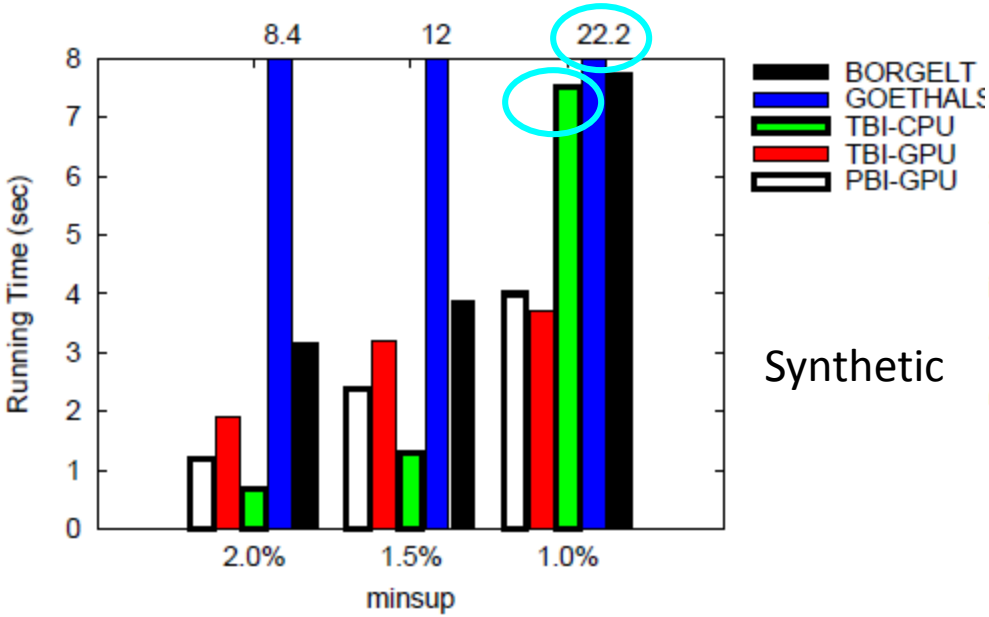
Dataset	#Item	Avg. Length	#Tran	Density	Original data size	Bitmap size
T40I10D100K (synthetic)	1,000	40	100,000	4%	~15MB	12MB
Retail	16,469	10.3	88,162	0.6%	~4MB	180MB
Chess	75	37	3196	49%	~335KB	30KB

Apriori Implementations

Impl.	Candidate Generation	Support Countnig	Itemset	Transactions
PBI-GPU	Multi-threaded on the GPU	Multi-threaded on the GPU	Bitmap	Bitmap
TBI-GPU	Single-threaded on the CPU	Multi-threaded on the GPU	Trie	Bitmap
TBI-CPU	Single-threaded on the CPU	Multi-threaded on the CPU	Trie	Bitmap
GOETHALS	Single-threaded on the CPU	Multi-threaded on the CPU	Trie	Horizontal layout
BORGELT	Single-threaded on the CPU	Single-threaded on the CPU	Trie	Trie

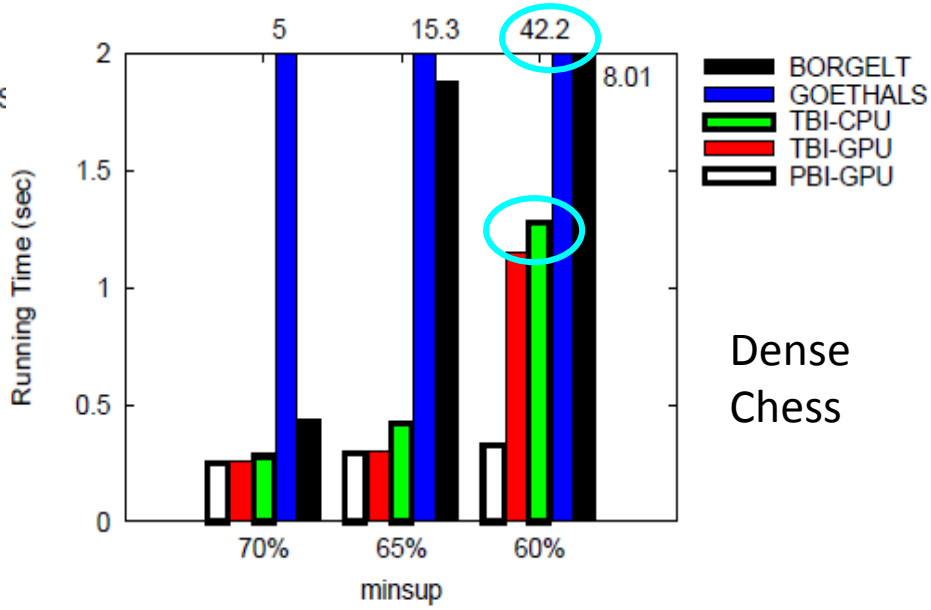
Best Apriori implementation in FIMI repository.
(Frequent Itemset Mining Implementations Repository)

TBI-CPU vs GOETHALS



Synthetic

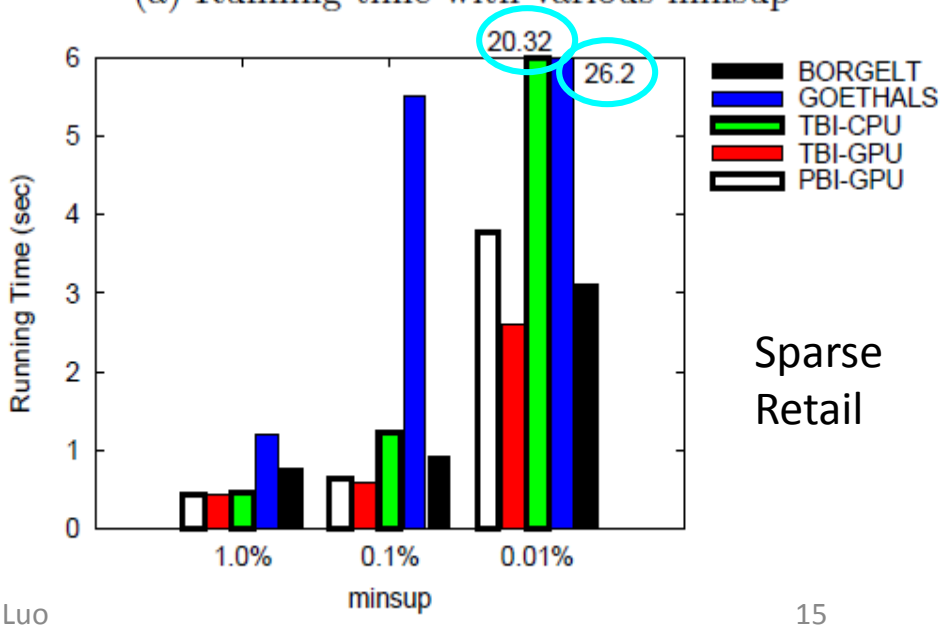
(a) Running time with various minsup



Dense
Chess

(a) Running time with various minsup

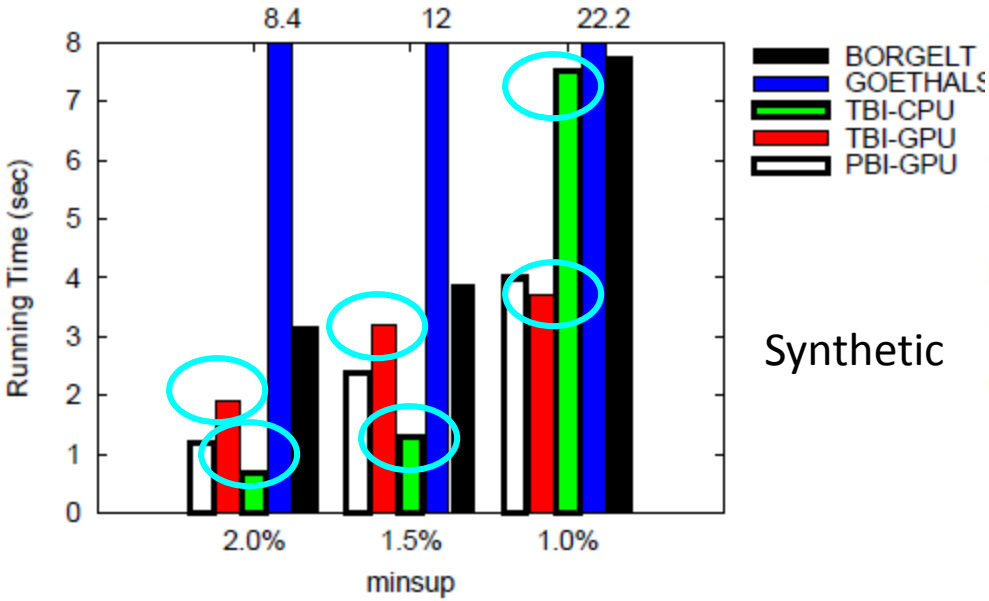
Impl.	Itemset/Candidate Generation	Transactions / Suport Counting
TBI-CPU	Trie / CPU	Bitmap / CPU
GOETHALS	Trie / CPU	Horizontal layout / CPU



Sparse
Retail

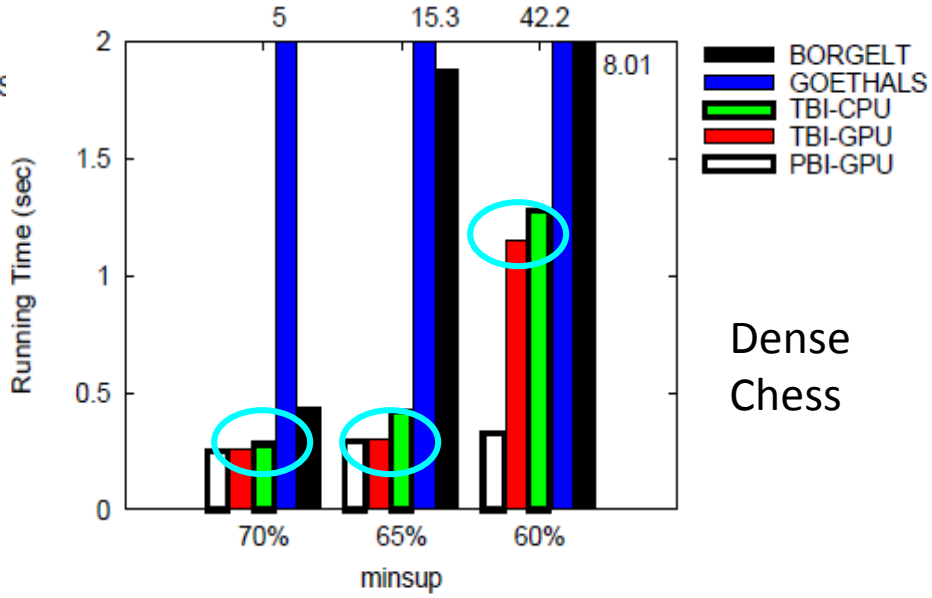
(a) Running time with various minsup

TBI-GPU vs TBI-CPU



Synthetic

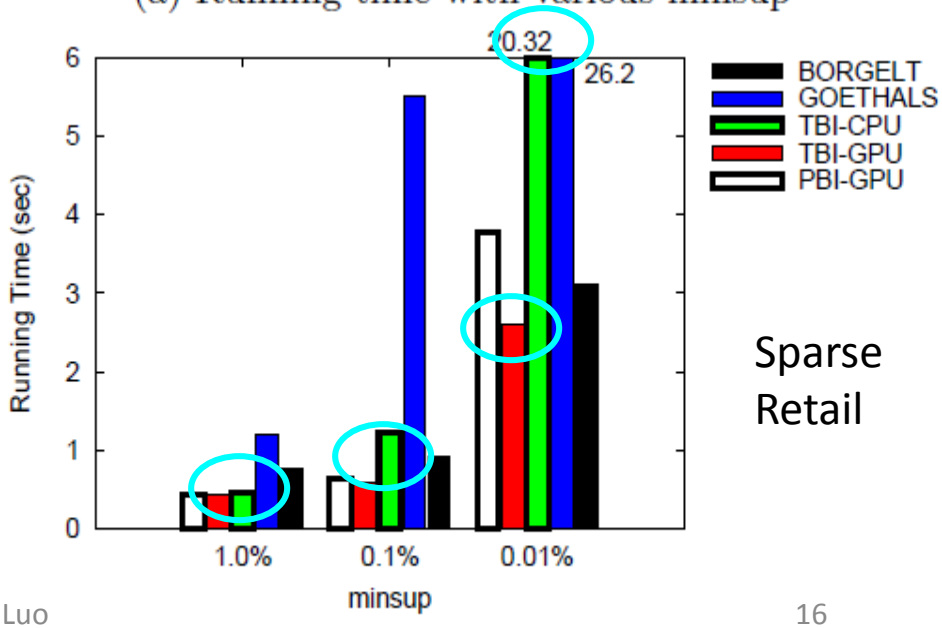
(a) Running time with various minsup



Dense
Chess

(a) Running time with various minsup

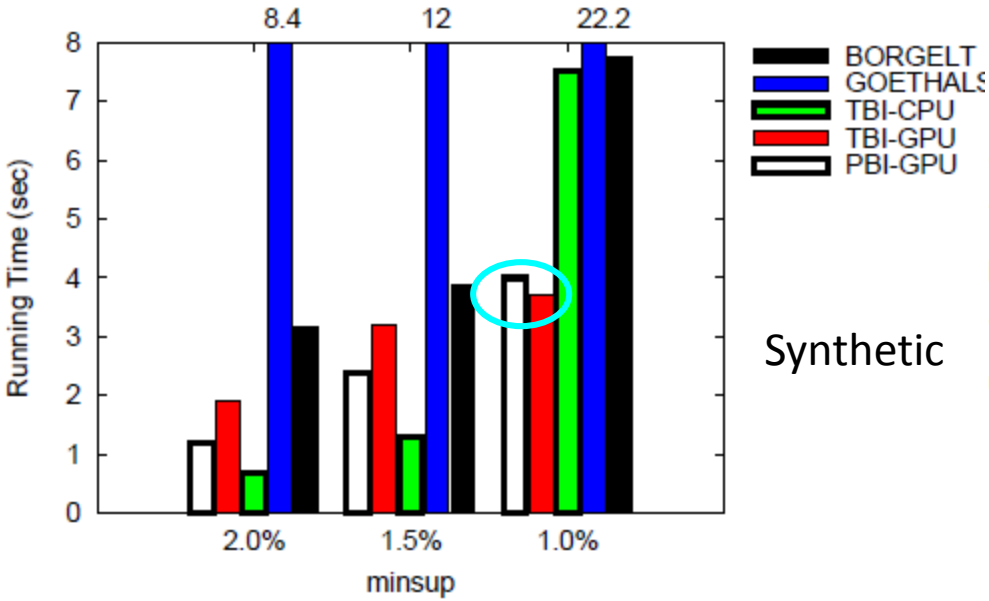
Impl.	Itemset/Candidate Generation	Transactions / Suport Counting
TBI-GPU	Trie / CPU	Bitmap / GPU
TBI-CPU	Trie / CPU	Bitmap / CPU



Sparse
Retail

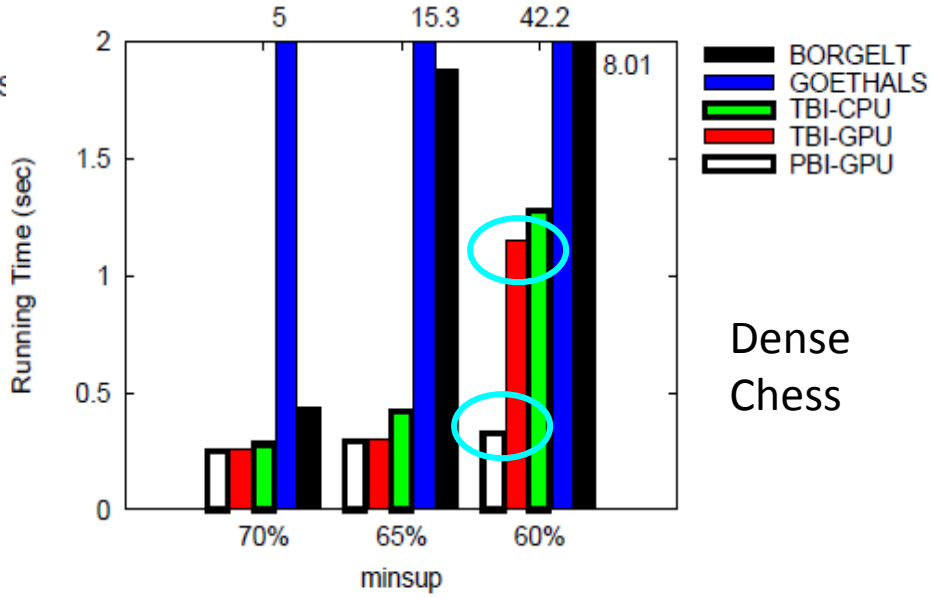
(a) Running time with various minsup

PBI-GPU vs TBI-GPU



Synthetic

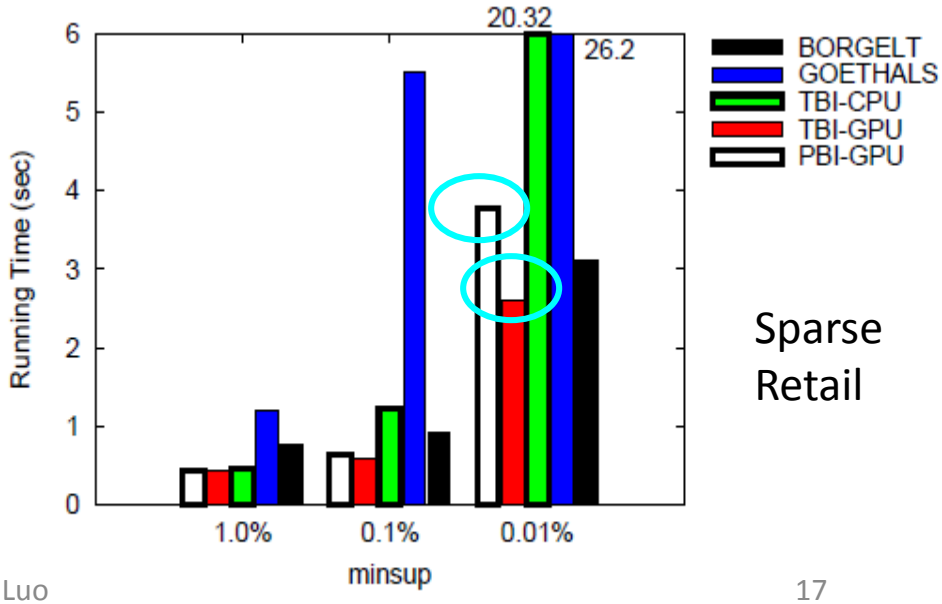
(a) Running time with various minsup



Dense
Chess

(a) Running time with various minsup

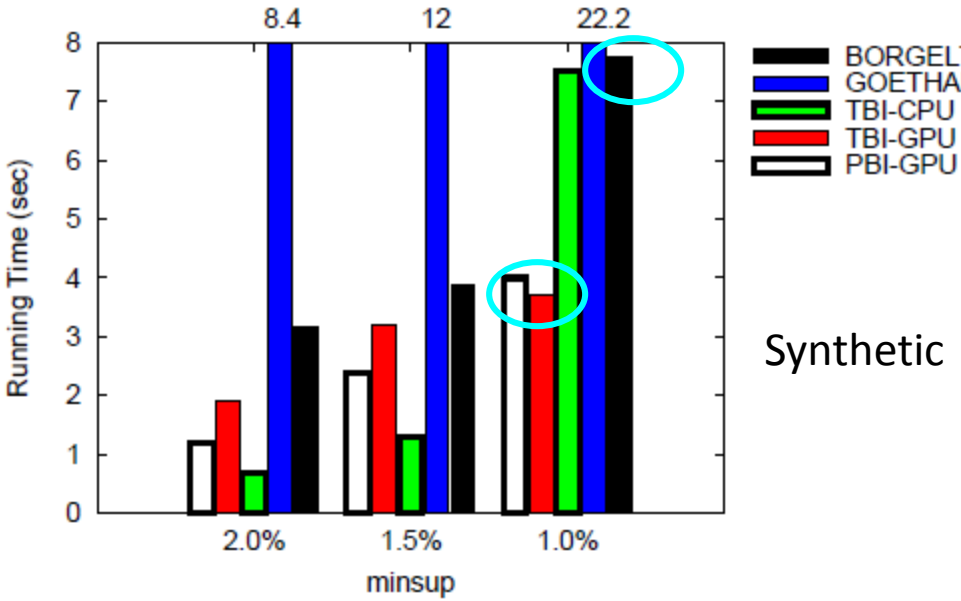
Impl.	Itemset/Candidate Generation	Transactions / Suport Counting
PBI-GPU	Bitmap / GPU	Bitmap / GPU
TBI-GPU	Trie / CPU	Bitmap / GPU



Sparse
Retail

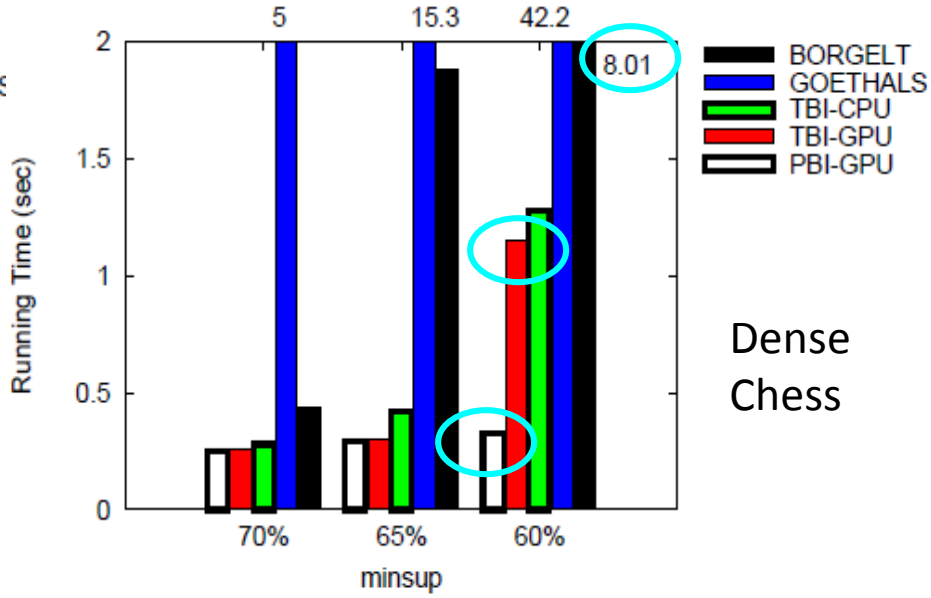
(a) Running time with various minsup

PBI-GPU/TBI-CPU vs BORGELT



Synthetic

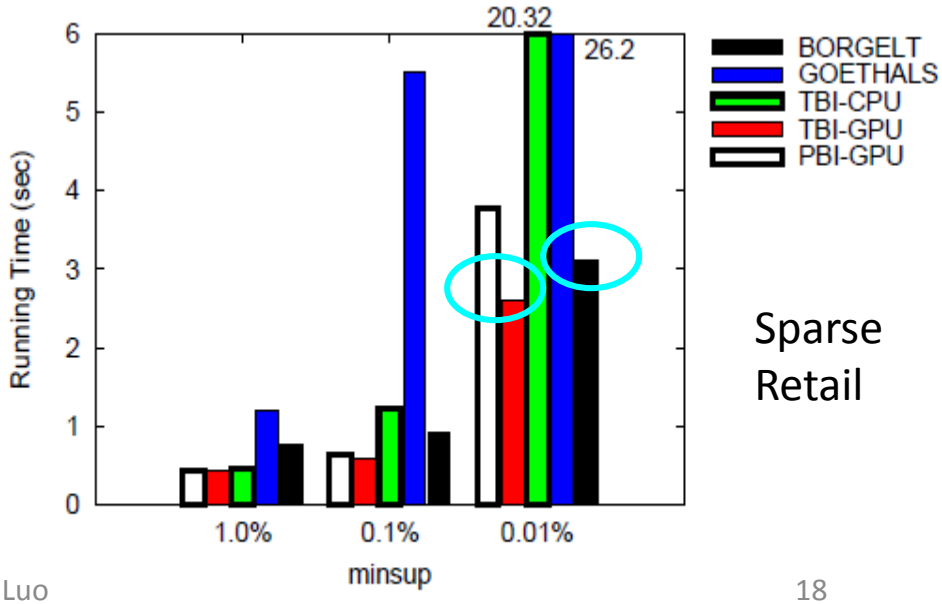
(a) Running time with various minsup



Dense
Chess

(a) Running time with various minsup

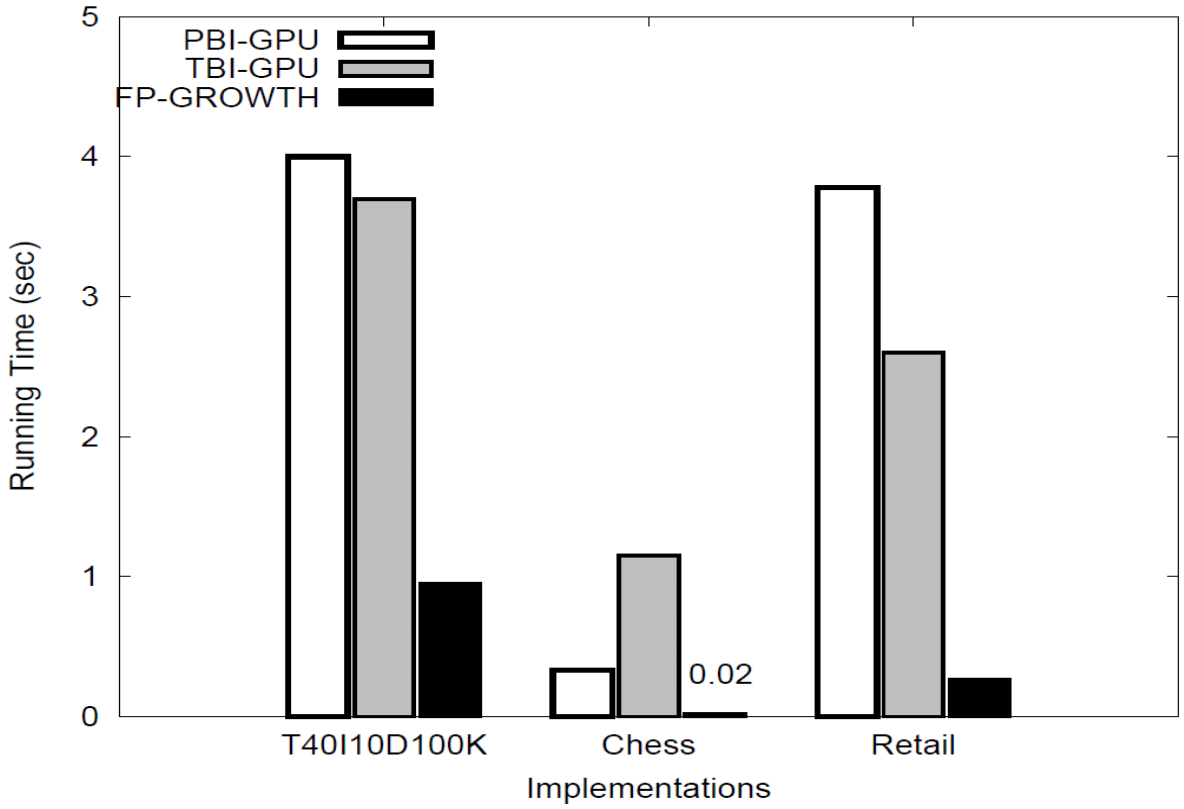
Impl.	Itemset/Candidate Generation	Transactions / Suport Counting
PBI-GPU	Bitmap / GPU	Bitmap / GPU
TBI-GPU	Trie / CPU	Bitmap / GPU
BORGELT	Trie /CPU	Trie /CPU



Sparse
Retail

(a) Running time with various minsup

Comparison with FP-growth



Summary

- GPU-based Apriori
  - Pure bitmap-based impl.
    - Bitmap representation for itemsets
    - Bitmap representation for transactions
    - GPU processing
  - Trie-based impl.
    - Trie Representation for itemsets
    - Bitmap Representation for transactions
    - GPU + CPU co-processing
  - Better than CPU-based Apriori
  - Still worse than CPU-based FP-growth