グリッドコンピューティング

2012/11/05

小島諒介(12M38195)

論文

“gpuDCI: Exploiting GPUs in Frequent Itemset Mining”

Claudio Silvestri, Salvatore Orlando

Universit`a Ca’ Foscari Venezia

2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing

Outline

- Frequent Itemset Mining(FIM)
- Challenges
- DCI algorithm
- The CUDA framework
- gpuDCI algorithm
- Experiments
- Conclusion

Outline

- Frequent Itemset Mining(FIM)
- Challenges
- DCI algorithm
- The CUDA framework
- gpuDCI algorithm
- Experiments
- Conclusion

Frequent Itemset Mining(FIM)

- Goal: Find the sets of items that are bought together in not less than *minimum support.*

Ex) Market Basket Analysis

Customers	products
A	Milk, Bread
B	Milk, Bread, Butter
C	Bread

Minimum support=50%

{ Milk}, { Bread }, { Milk, Bread }

{ Milk}, { Bread }, { Milk, Bread } are Frequent patterns in this example,
Because they are occurred at more than 50%

The minimum support is decided by user.

Term / Notation

- $I = \{i_1, i_2, \ldots, i_m\}$: a set of items.
- Transaction t : a set of items.
- Transaction Database T : a set of transactions
  $T = \{t_1, t_2, \ldots, t_n\}$.

Customers	products
A	Milk, Bread
B	Milk, Bread, Butter
C	Bread

The rows are transaction.
The columns are a set of items.

Outline

- Frequent Itemset Mining(FIM)
- Challenges
- DCI algorithm
- The CUDA framework
- gpuDCI algorithm
- Experiments
- Conclusion/future work

The challenges in FIM

- Large size of the search space
  - Power set of the items
    the search space is
    exponential size

candidate

{Milk}
{Bread}
{Butter}
{Milk, Bread}
{Bread, Butter}
{ Milk, Butter}
{Milk, Bread}
{Milk, Bread, Butter}

- Large size of transaction/items
- Small minimum support

The Common techniques

- Apriori principle:

  "all the subsets of a frequent set must be frequent"

  Ex)

  { Milk, Bread } is frequent →

  {Milk} is frequent and *{Bread}* is frequent

- restricting as much as possible search space

  (candidate generation / pruning)

  – starting with on short pattern, and increasing the size of pattern
  – If *{Butter}* is not frequent, frequent sets don't has a subset *{Butter}* .

DCI Algorithm

- Iterative algorithm
  - search k-size itemsets in k-step
- Multi-strategy algorithm
  - Direct Count Phase (DC phase)
    - Count how many times candidate itemsets occur in Dataset.
    - Horizontal layout
  - Intersection Phase (I phase)
    - And operation + count
    - Vertical layout

Customers	products
A	Milk, Bread
B	Milk, Bread, Butter
C	Bread

	Milk	Bread	Butter
A	○	○	×
B	○	○	○
C	×	○	×

DCI is a multi-strategy algorithm for Frequent Itemset Mining (FIM), characterized by several phases, each exploiting a different strategy.

Candidates item set is pruned by Apriori principle.

k is incremented at each iteration.
When k is small, this algorithm use DC phase.
When k is large, this algorithm use I phase.

Features of DCI Algorithm

- Simple static data structure
- Permits a lot of data parallelization
- Bitwise operations(I phase)

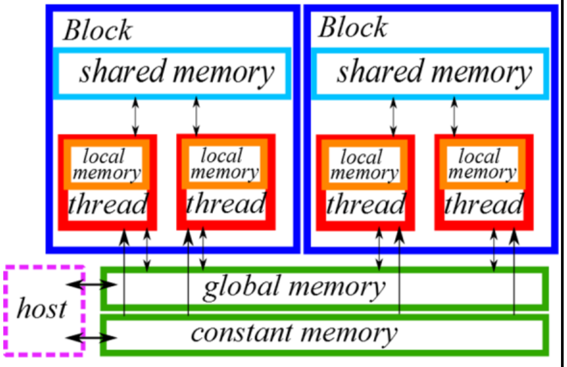
The bitwise operations are And operation and popcount operation

Outline

- Frequent Itemset Mining(FIM)
- Challenges
- DCI algorithm
- The CUDA framework
- gpuDCI algorithm
- Experiments
- Conclusion

The CUDA framework

- Optimal GPU usage
  - Processor utilization
    - Resources
  - Blocking operations
    - Kernel launches
    - Memory transfers
  - Memory access patterns
    - Coalesced access to global memory



Outline

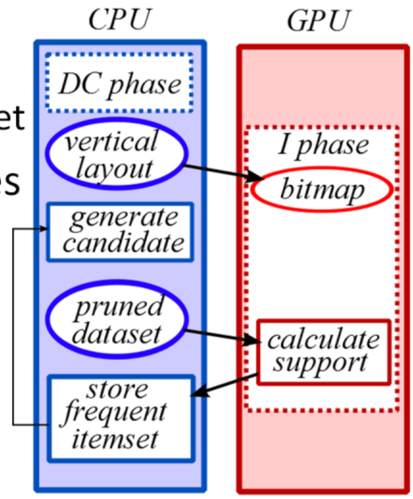
- Frequent Itemset Mining(FIM)
- Challenges
- DCI algorithm
- The CUDA framework
- gpuDCI algorithm
- Experiments
- Conclusion/future work

DCI on GPUs

- The parallelizing strategies
- The data access patterns
  - Global memory: Coalescing
  - Shared memory: Bank conflict
- The careful management of the GPU memory hierarchy

gpuDCI

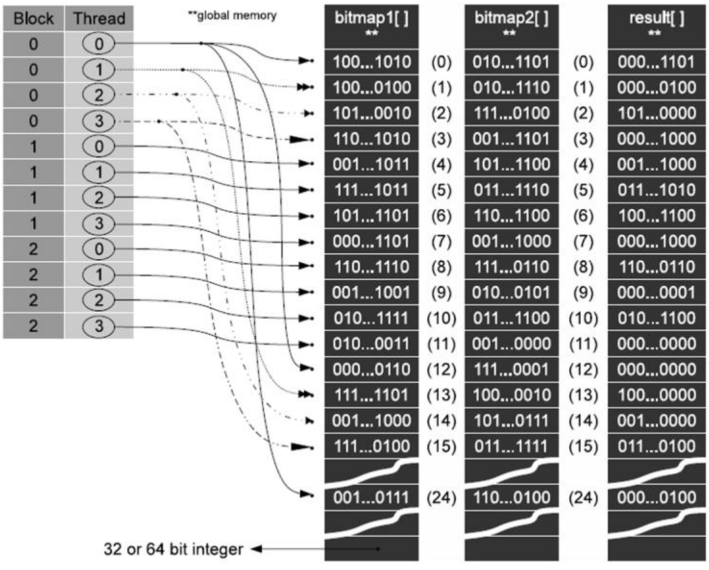
- DC phase and candidate generation don't use GPU.
  - IO bound
  - Transferring unpruned dataset
- two parallelization strategies
  - Transaction-wise
  - Candidate-wise



When k is small, CPU calculate support with DC

When k is large, CPU generates candidates, and then sends them to GPU.
The generated candidates are pruned by Apriori principle.

Transaction-wise intersection



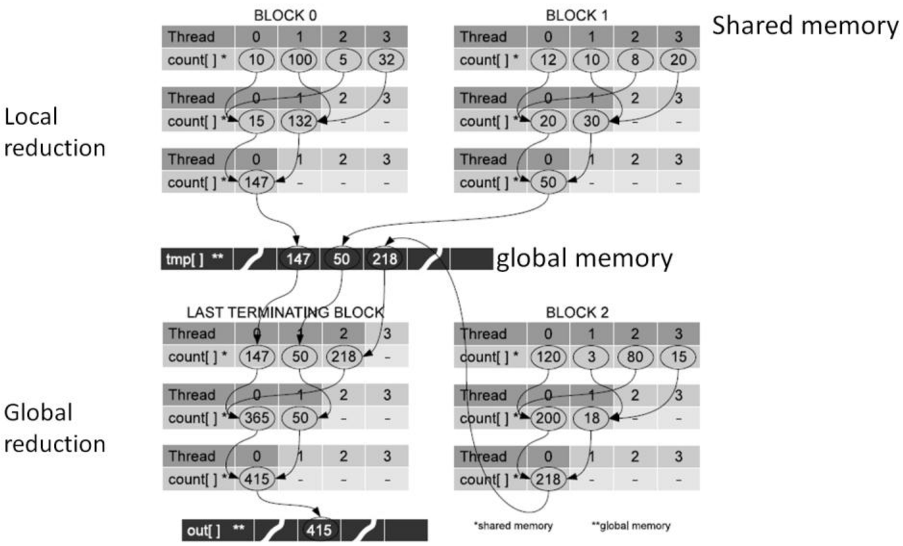
(a) Intersection

global memory

Each thread is in charge of an interleaved portion of the bitmap.

DCI uses a bitwise
data structure: each retained frequent item is associated
with a *bitmap*, where the bit in the *nth* position is equal
to 1 iif the *nth* transaction contains the item.(vertical layout)

Transaction-wise count



(b) Count

To avoid bank conflicts, both reduction are performed by using a pair-wise, tree based, approach make use
of the fast shared memory that is present on each multiprocessor.

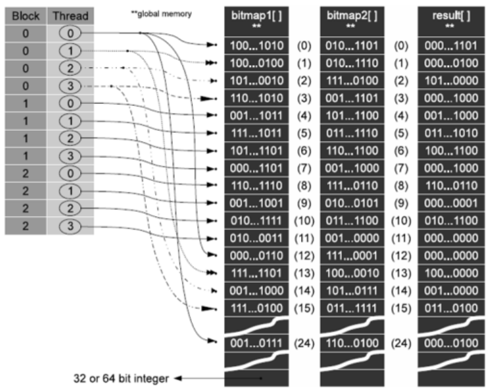
Transaction-wise features

- fixed stride of *blocks* $\times$ *threads*
- thread blocks $\leqq$ GPU multiprocessors
  - To ensure that all cores are involved in the computation
- global memory access should be overlapped with computation
  - to ensure that the cores of each multiprocessor are active

Transaction-wise problems

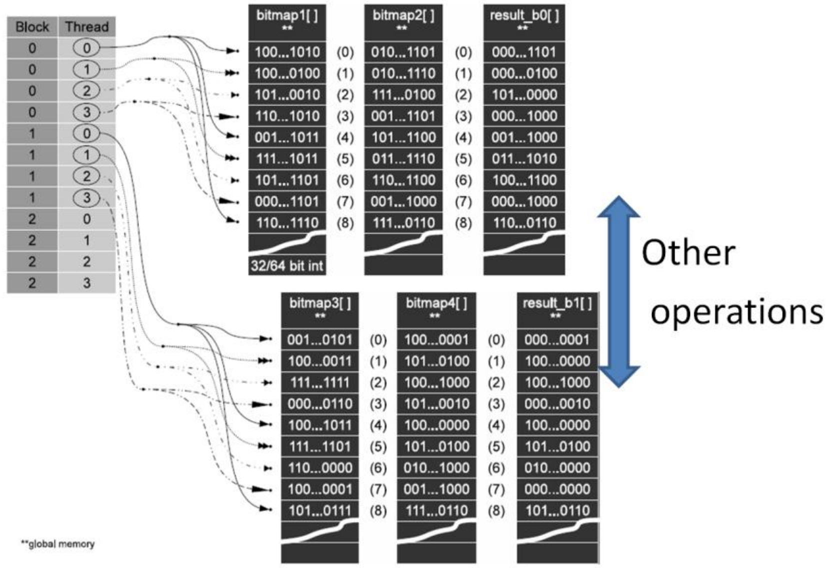
The size of transaction is small

- Leave some multiprocessor idle
- Latency for the global memory access



(a) Intersection

Candidate-wise intersection

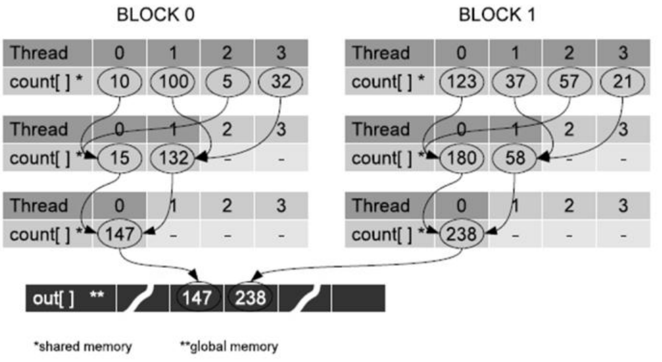


(a) Intersection

In this approach each GPU multiprocessor works on the intersection and count operations related to a different candidate.

The amount of GPU memory required is larger than the one required by the previous strategy.

Candidate-wise count



(b) Count

Some results about the different items is calculated at the same time.

Implementation

- Batches of operation
  - Send sequances
    - Transaction-wise: kernel launch
    - Candidate-wise : constant memory
- Basic operation on GPU
  - operation 32/64bit logical and
  - Popcount: An hardware implementation
  - globalReduce / localReduce: http://www.nvidia.com/object/cuda_sample_data-parallel.html(Broken link)

Outline

- Frequent Itemset Mining(FIM)
- Challenges
- DCI algorithm
- The CUDA framework
- gpuDCI algorithm
- Experiments
- Conclusion/future work

Experiment: Dataset

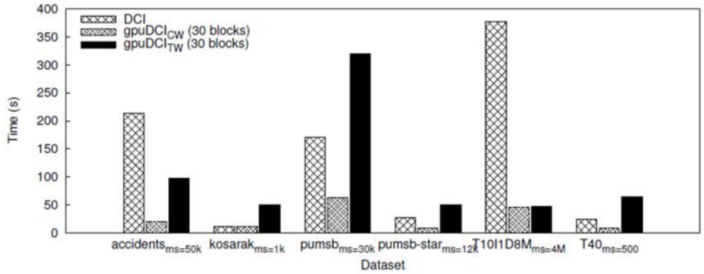


Figure 6. Running time for different datasets

Machine
Intel Core2 Quad CPU @ 2.66GHz + 8 GB of RAM

NVIDIA GTX275 GPU +30 MP(240 cores) @1.4 GHz + 896MB device memory +Cuda device capability 1.3.

- CPU version use more aggressive pruning algorithm
- gpuDCI$_{CW}$ is faster than gpuDCI$_{TW}$

Experiment: Pattern length

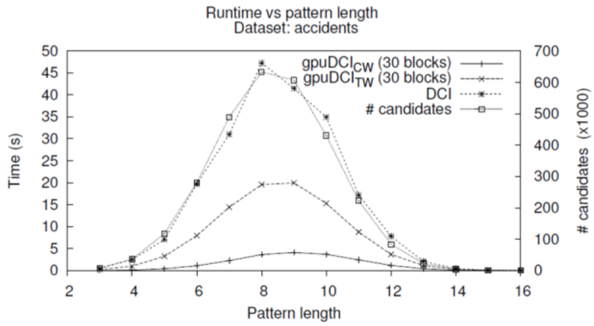
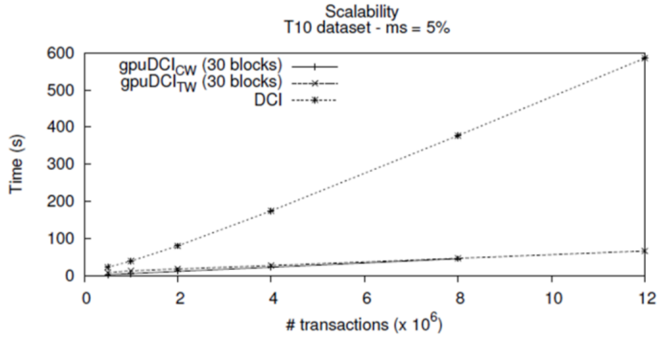


Figure 7. Running time for different pattern length on two different datasets

- gpuDCI has advantage of nearly one order
- Running time is roughly proportional to the number of candidates

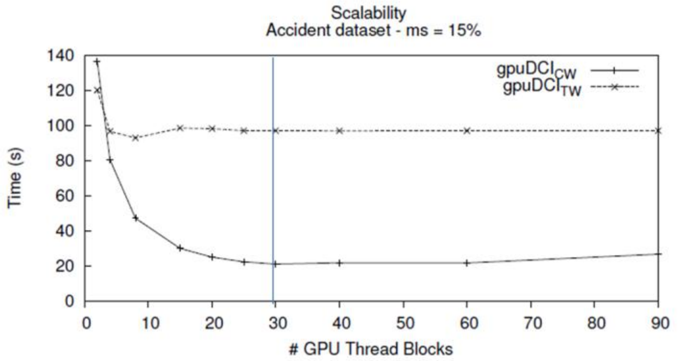
Experiment: Dataset size



(a) Increasing dataset size.

- linear with respect to the dataset sizes
- gpuDCI$_{CW}$ needed to exploit many caches.

Experiment: Multiprocessors



(b) Increasing number of thread blocks.

- Multiprocessors are not under scheduled due to memory access latency (gpuDCI$_{CW}$)
- The number of transactions is not sufficient. (gpuDCI$_{TW}$)

The number of multiprocessors is 30.

Outline

- Frequent Itemset Mining(FIM)
- Challenges
- DCI algorithm
- The CUDA framework
- gpuDCI algorithm
- Experiments
- Conclusion/future work

Conclusion

- gpuDCI : a parallel algorithm, which exploits GPUs to compute frequent itemset.
- Two parallelization strategies
  - The candidate-wise approach is faster but uses more memory.
- The experiments showed that using the GPU gives clear advantages.

Future work

- Some other technique for DCI
- Expansion to the frequent closed itemset

frequent closed itemsets are a condensed representation of frequent itemsets that can be directly computed from the data.