A Characterization of Data Mining Algorithms on a Modern Processor

Amol Ghoting, Gregory Buehrer, and Srinivasan Parthasarathy

Data Mining Research Laboratory,

The Ohio State University

Daehyun Kim, Anthony Nguyen, Yen-Kuang Chen,

and Pradeep Dubey

Architecture Research Laboratory,

Intel Corporation

Roadmap

- *Motivation and Contributions*
- Algorithms under study
- Performance characterization
- Case study:
  - Improving performance of FP-Growth
- Related work
- Conclusions

Motivation

- KDD applications constitute a rapidly growing segment of the commercial and scientific computing domains
    - Interactive process → response times
    - Memory and compute intensive
- Modern architectures
    - – Memory wall issues
        - Latency tolerating mechanisms – prefetching, SMT
- Objective here is to characterize such applications on a modern architecture
    - – Can we leverage above mechanisms effectively?

Contributions

- Specifically, we study
  - Performance and memory access behavior of eight data mining algorithms
  - Impact of processor technologies such as hardware pre-fetching and simultaneous multithreading (SMT)
  - How to leverage latency-tolerating mechanisms to improve performance of frequent pattern mining

Roadmap

- Motivation and Contributions
- *Algorithms under study*
- Performance characterization
- Case study:
  - Improving performance of FP-Growth
- Related work
- Conclusions

Algorithms under study (1)

- Frequent itemset mining
  - Finds groups of items that co-occur frequently in a transactional data set
  - Example: “Item A and Item B are purchased together 90% of the time”
    - FPGrowth (FP-tree)
    - MAFIA (Tid-list as a bit vector)
- Sequence mining
  - Discovers sets of items that are shared across time
  - Example: “70% of the customers who buy item A also buy item B within 1 month”
    - SPADE (Tid-list)

Algorithms under study (2)

- Graph mining
  - Finds frequent sub-graphs in a graph data set
    - FSG
      - Tid-list
- Clustering
  - Partitions data points into groups or clusters such that intra-cluster distance in minimized and inter-cluster distance in maximized
    - kMeans and vCluster

Algorithms under study (3)

- Outlier detection
  - Finds the top *k* points in a data set that are most different from the remaining points
    - ORCA
- Decision tree induction
  - Learns a decision tree from a data set
    - C4.5

Roadmap

- Motivation and Contributions
- Algorithms under study
- *Performance characterization*
- Case study:
  - Improving performance of FP-Growth
- Related work
- Conclusions

Performance characterization

- Setup
  - Intel P4 at 2.8GHz with HT technology
  - 1.5GB of main memory
  - 8KB L1 d-cache and 512 KB L2 u-cache
  - Intel VTune Performance Analyzers to collect performance characteristics of execution
  - Synthetic/Real datasets
  - All codes were obtained from the authors

Operation mix

	FPGrowth	MAFIA	SPADE	FSG	kMeans	vCluster	C4.5	ORCA
Integer ALU operations / instruction	0.56	0.822	0.636	0.625	0.688	0.620	0.769	0.607
Floating point operations / instruction	0.001	0.001	0.004	0.015	0.252	0.207	0.087	0.273
Memory operations / instruction	**0.65**	**0.433**	**0.593**	**0.692**	**0.267**	**0.353**	**0.517**	**0.392**
Branch operations / instruction	0.136	0.074	0.177	0.166	0.057	0.154	0.163	0.156

Cache and CPU performance

- FPGrowth
  - Poor cache hit rates
  - Large number of DTLB misses per instruction
  - Poor data locality
  - Low ILP

L1 LD hit rate	0.891
L2 LD hit rate	0.430
L2 LD misses / instruction	0.03
LD operations / instruction	0.515
ST operations / instruction	0.135
DTLB misses / instruction	0.024
ITLB misses / instruction	0.000
CPU utilization	0.119

Cache and CPU performance

- MAFIA
  - Has the highest CPU utilization of the considered workloads
    - Counting using bit-vectors is very efficient
  - Temporal locality can be improved
- Note:
  - The search is not as efficient as FPGrowth

L1 LD hit rate	0.953
L2 LD hit rate	0.997
L2 LD misses / instruction	0.001
LD operations / instruction	0.391
ST operations / instruction	0.042
DTLB misses / instruction	0.000
ITLB misses / instruction	0.000
CPU utilization	0.446

Cache and CPU performance

- SPADE
  - Temporal locality can be improved
  - Very poor CPU utilization
    - Tidlist joins are expensive

L1 LD hit rate	0.954
L2 LD hit rate	0.992
L2 LD misses / instruction	0.001
LD operations / instruction	0.538
ST operations / instruction	0.116
DTLB misses / instruction	0.012
ITLB misses / instruction	0.000
CPU utilization	0.146

Cache and CPU performance

- FSG
  - Temporal locality can be improved
  - Very poor CPU utilization
    - Tidlist joins are expensive

L1 LD hit rate	0.963
L2 LD hit rate	0.985
L2 LD misses / instruction	0.002
LD operations / instruction	0.532
ST operations / instruction	0.160
DTLB misses / instruction	0.007
ITLB misses / instruction	0.000
CPU utilization	0.152

Cache and CPU performance

- kMeans
  - Poor CPU utilization
    - FPU intensive

L1 LD hit rate	0.979
L2 LD hit rate	0.989
L2 LD misses / instruction	0.000
LD operations / instruction	0.254
ST operations / instruction	0.013
DTLB misses / instruction	0.001
ITLB misses / instruction	0.001
CPU utilization	0.244

Cache and CPU performance

- vCluster
  - Poor data locality
    - Graph partitioning

L1 LD hit rate	0.882
L2 LD hit rate	0.987
L2 LD misses / instruction	0.000
LD operations / instruction	0.279
ST operations / instruction	0.083
DTLB misses / instruction	0.001
ITLB misses / instruction	0.000
CPU utilization	0.322

Cache and CPU performance

- C4.5
  - The sort routine has poor data locality
    - Cache-conscious sort?

L1 LD hit rate	0.60
L2 LD hit rate	0.969
L2 LD misses / instruction	0.031
LD operations / instruction	0.385
ST operations / instruction	0.131
DTLB misses / instruction	0.005
ITLB misses / instruction	0.000
CPU utilization	0.049

Cache and CPU performance

- ORCA
  - Similar trends

L1 LD hit rate	0.970
L2 LD hit rate	0.993
L2 LD misses / instruction	0.000
LD operations / instruction	0.335
ST operations / instruction	0.057
DTLB misses / instruction	0.003
ITLB misses / instruction	0.000
CPU utilization	0.316

Impact of hardware prefetching and SMT

	FPGrowth	MAFIA	SPADE	FSG	kMeans	vCluster	C4.5	ORCA
Speedup due to hardware pre-fetching	1.11	1.65	1.02	1.15	1.02	1.19	1.06	1.01
Speedup due to SMT	1.02	1.06	1.05	1.26	1.30	1.26	1.18	1.03

- Prefetching improves performance for MAFIA, significantly
  - AND operation on bit-vectors
  - Working set is larger than other frequent pattern mining workloads
- SMT helps the FPU intensive workloads, as it is able to mask FPU latency
  - Not easy to hide memory latency

Characterization summary

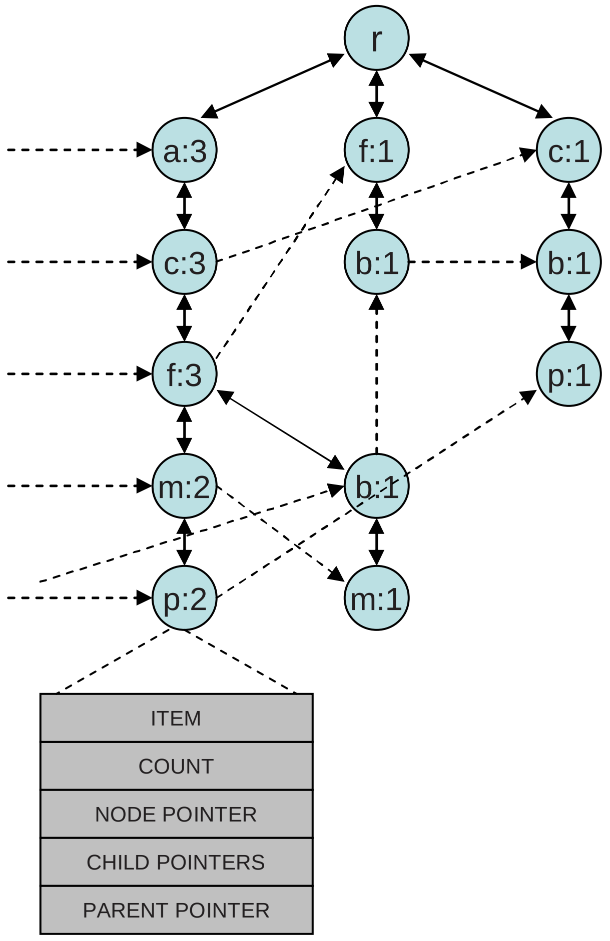
- Compute intensive
  - Integer ALU and FPU intensive
- Memory intensive
  - Limits CPU utilization
- Good spatial locality
  - Temporal locality can be improved in most cases
- SMT improves performance for FPU intensive workloads

Roadmap

- Motivation and Contributions
- Algorithms under study
- Performance characterization
- *Case study:*
  - *Improving performance of FP-Growth*
- Related work
- Conclusions

Improving performance of FPGrowth (1)

- FP-tree as an intermediate data set representation
- Pointer-based structure
- Tree traversals are bottom-up accesses
  – We only need *item* and *parent pointer*!

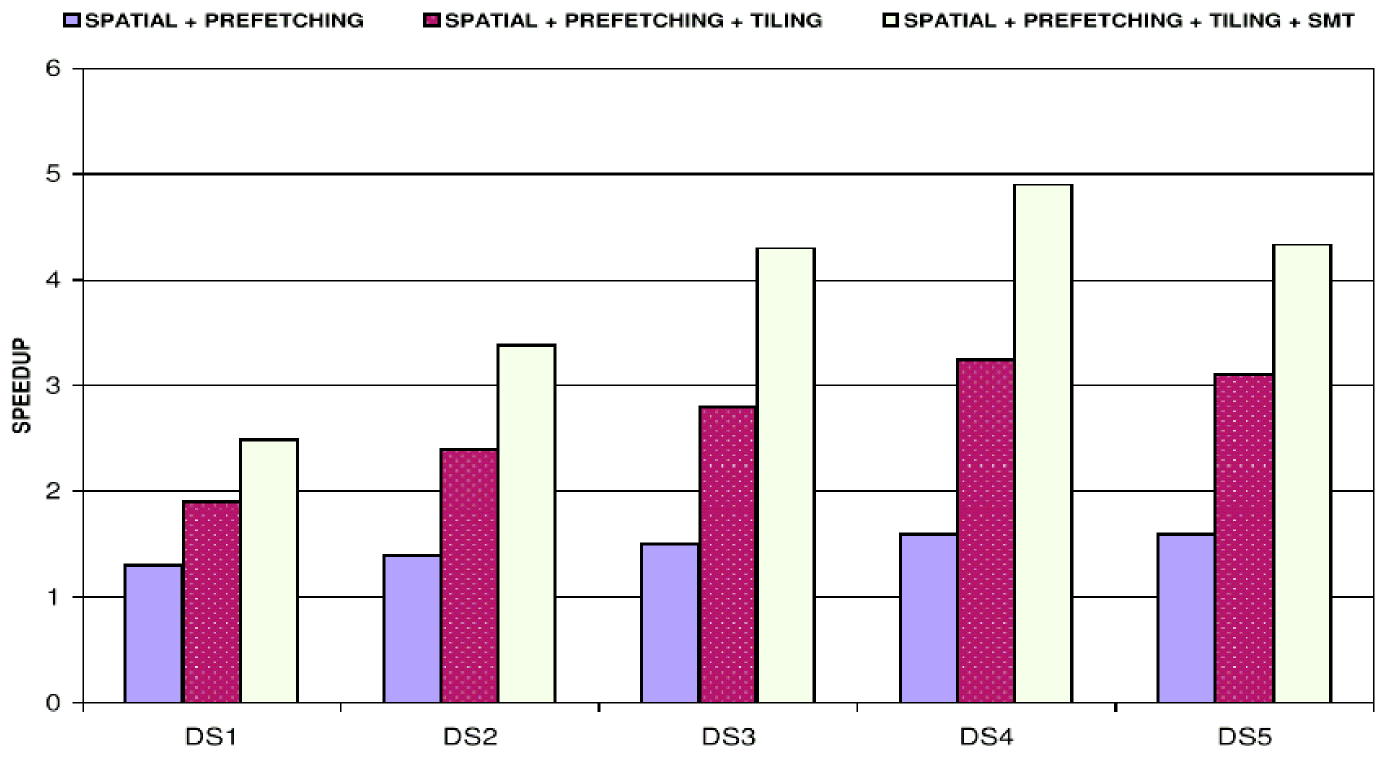


Improving performance of FPGrowth (2)

- Improve spatial locality
  - Node size reduction
  - Depth-first tree reordering
- Improve temporal locality
  - Path tiling
- Improve ILP
  - Thread co-scheduling on an SMT for improved cache-reuse

Speedup

- DS1 to DS4 - synthetic datasets (increasing size)
- DS5 – real dataset



Roadmap

- Motivation and Contributions
- Algorithms under study
- Performance characterization
- Case study:
  - Improving performance of FP-Growth
- *Related work*
- Conclusions

Related work

- Cache-conscious data base algorithms
  - DBMS on modern hardware
    - Ailamaki *et al.* [VLDB99]
  - Cache sensitive search trees and B+ trees
    - Rao and Ross [VLDB99,SIGMOD00]
  - Prefetching for B+ trees and Hash-Join
    - Chen *et al.* [SIGMOD01,ICDE04]
- Cache performance of data mining algorithms
  - SOM
    - Kim *et al.* [WWC99]
  - C4.5
    - Bradford and Fortes [WWC98]
  - Apriori
    - Parthasarathy *et al.* [KAIS01]
- Parallel scalability and I/O performance of data mining algorithms
  - Y. Liu *et al.* [PDCS04]

Conclusions

- We presented a characterization of 8 data mining algorithms
  - Compute and memory intensive
  - Temporal locality can be improved in most cases
  - Prefetching helps workloads with good spatial locality
  - SMT helps FPU intensive workloads
  - Memory intensive nature of these algorithms limits performance
- Improved performance of FPGrowth

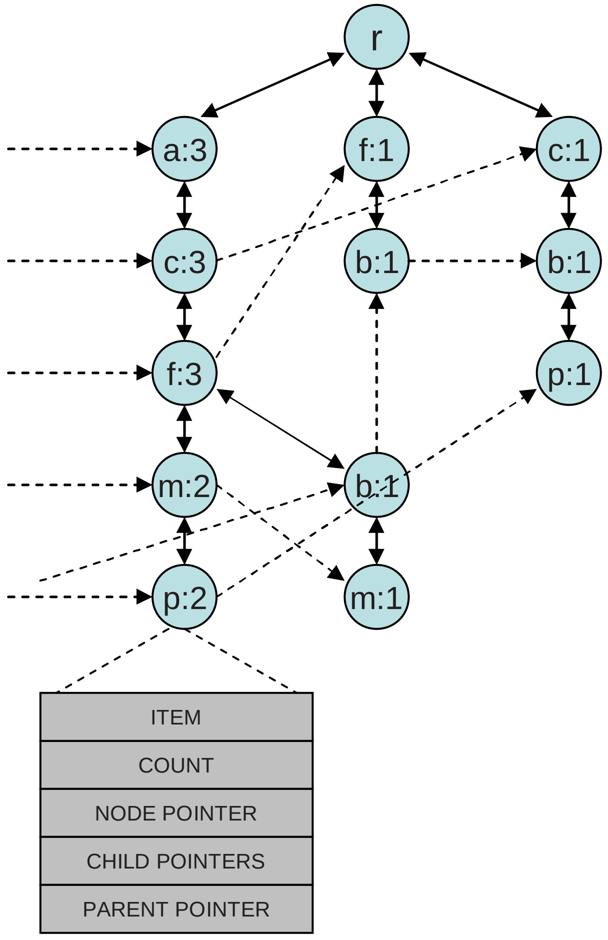
- *Effective algorithm design needs to take account both traditional complexity issues* ***<u>and</u>*** *modern architectural designs.*

Questions?

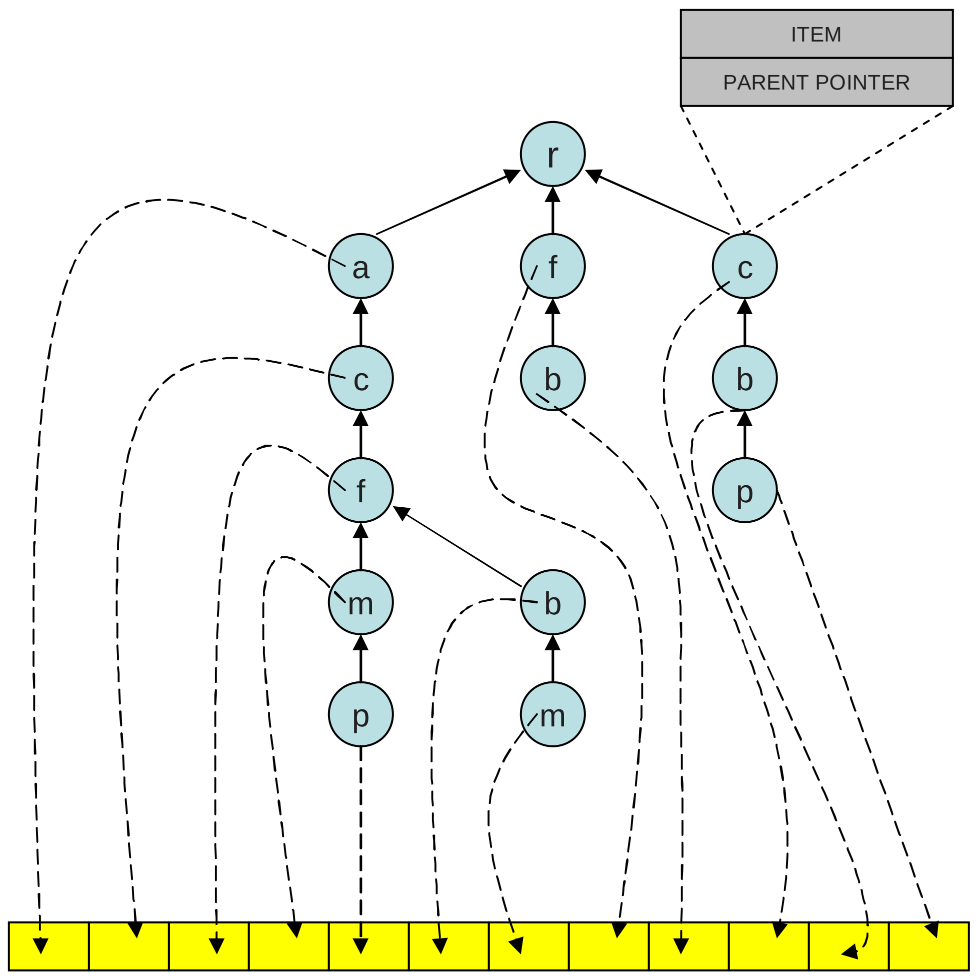
- Thanks
  - NSF CAREER IIS-0347662
  - NSF NGS CNS-0406386
  - DOE ECPI DE-FE02475

Improving performance of FPGrowth

- FP-tree as an intermediate data set representation
- Pointer-based structure
- Tree accesses are bottom-up and are repeated for each item
  - We only need *item* and *parent pointer*!



Cache-conscious prefix tree



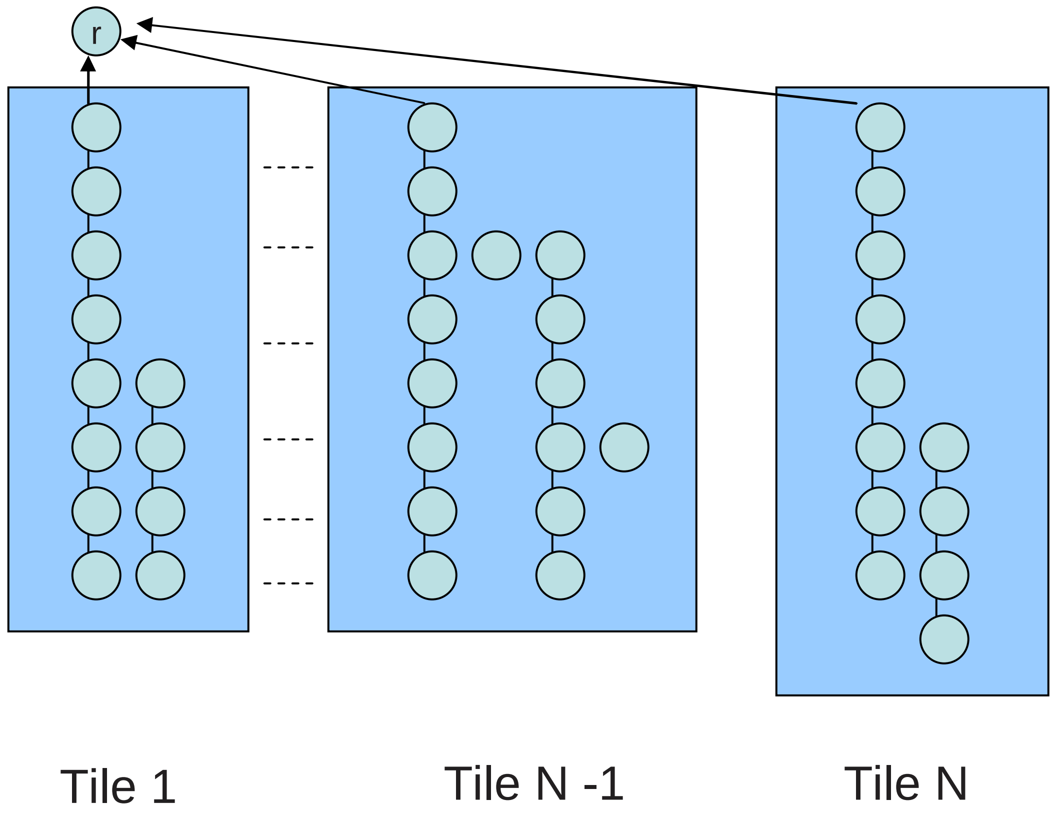
Header lists (for Node pointers)

o		
o	o	o
o	o	
o	o	
o	o	
o	o	

Count lists (for Node counts)

o		
o	o	o
o	o	
o	o	
o	o	
o	o	

Path tiling & Co-scheduling for cache-reuse SMT


r
Tile 1
Tile N -1
Tile N