Parallel Graph Mining with GPUs

Robert Kessl¹, Nilothpal Talukder², Pranay Anchuri², Mohammed J. Zaki²,³
¹Czech Technical University, Prague, Czech Republic ² Rensselaer Polytechnic Institute, Troy, NY, USA ³Qatar Computing Research Institute, Doha, Qatar

Frequent Subgraph Mining

- Discovering frequent subgraph patterns from a set of graphs (multi-graph) or a single graph.
- Practical applications in biology (PPI network), chemistry (similar compounds), or social networks (similar communities), etc.
- The problem is computationally hard as it requires:
  1. Enumeration of an exponential number of candidate subgraph patterns
  2. Checking their presence in a set of graphs (subgraph isomorphism)
- We address multi-graph problem with Graphics Processing Units (GPUs)

Graph Mining with GPUs

Algorithm:GRAPHMINING(D, minsup, P, Embeddings Σn(P))

/\Initial Call: P = ∅, Σn(P) = \∅
1:\hspace{1ex}(GPU step) Get all possible edges extensions (exts) \epsilon_r(P) of P
2: (GPU step) Compute support for exts in \epsilon_r(P) and remove infrequent exts
3: for each e = (v_i, v_j, l_i, l_j, b) ∈ \epsilon_r(P) do
4: if P' ⊆ P extended by e then
5: if P' = minDFS(P') then
6: output P'
7:\hspace{1ex}( GPU step) Create Σn(P')
8: GRAPHMINING(D, minsup, P', Σn(P'))
9: end if
10: end for

Data Structures

- Graph database D = {G_1, G_2, G_3} and pattern P
- Support: The number of graphs in D = {G_1, ..., G_n} that contain P, i.e., \sigma(P, D) = \{G : G ∈ D and P subgraph isomorphic to G\}.
- Frequent Subgraph Mining: Given D and minsup ∈ \mathbb{R}, find all frequent subgraph patterns, P, s.t. \sigma(P, D) ≥ minsup.

Graphics Processing Units (GPUs)

- GPUs have SIMT (single instruction multi-thread) architecture
- Energy efficient/less expensive choice for general purpose HPC
- Available programming platforms, such as NVIDIA CUDA[1]

Main challenges for graph mining:
- Serialized data for GPU kernels (no linked list or hashmap)
- Global memory coalescing as memory read in thread-groups
- Expensive GPU-CPU I/O transfer and dynamic allocation
- Algorithm steps are transformed into a set of parallel primitives (Prefix sum and Reduction etc.)

DFS code and Right-most Extension

- Graph mining algorithms [2,3] systematically generate all candidate subgraph patterns starting from a single edge
- gSpan algorithm [2] uses minimal DFS code [2] to uniquely represent patterns and avoid duplicates in candidate generation
- Performs right-most path extension from a minmal and frequent DFS code (subgraph pattern) by extending an edge
- Our variant keeps all occurences/embeddings [3] of the patterns (original gSpan does not) that grow with patterns

Major GPU Steps

- Support Computation: Compute the support of P from the extensions
- First, extract unique DFS code extensions from \epsilon_r(P)
- Support computation variants are based on the number of extenstions processed in parallel: (a) single-ext, (b) single-seg, and (c) multi-seg
- Number of threads is a bottleneck; single-seg which processes EXT_v demonstrates the best performance

Growing the Embeddings:
- Forward extensions from EXT_v introduce a new embeddings column
- For the backward extensions the last embeddings column is filtered

Results

- Setup: Sequential - Opteron 2.1GHz (256GB memory SMP 16 cores)
- GPU - Tesla C2075 (448 cores and 6GB memory)
- Table: Properties of the datasets
- GPU representation (partial) of graph database
- Figure: Graph database with global vertex ids and its storage in GPU memory.

Conclusions and Future Directions

- GPU memory is constrained for storing embeddings
- Investigate large graph mining in distributed setting
- Explore other accelerators, e.g. Intel Xeon Phi coprocessor

References

2. X. Yin, J. Han: gspan: Graph-Based Substructure Pattern Mining. International Conference on Data Mining, 2002
3. S. Nikolov, J. Han, A quickstart to frequent structure mining can make a difference. ACM International Conference on Knowledge discovery and data mining, 2004

Acknowledgements: This work was supported in part by NSF Awards CCF-1240646 and IIS-1302231.