Improving Large Graph Processing on Partitioned Graphs in the Cloud

Rishen Chen
Xuetian Weng
Binsheng He
Mao Yang
Byron Choi
Xiaoming Li

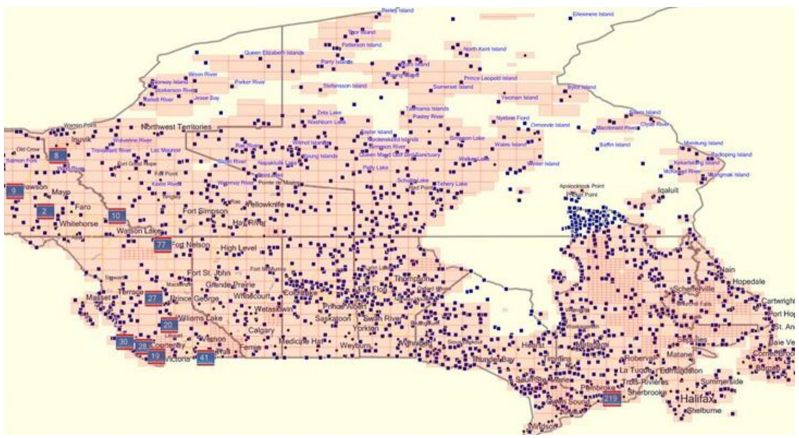
Presented by Prateek Goel
November 19, 2014

CS 755 – Fall 2014
David R. Cheriton School of Computer Science
University of Waterloo

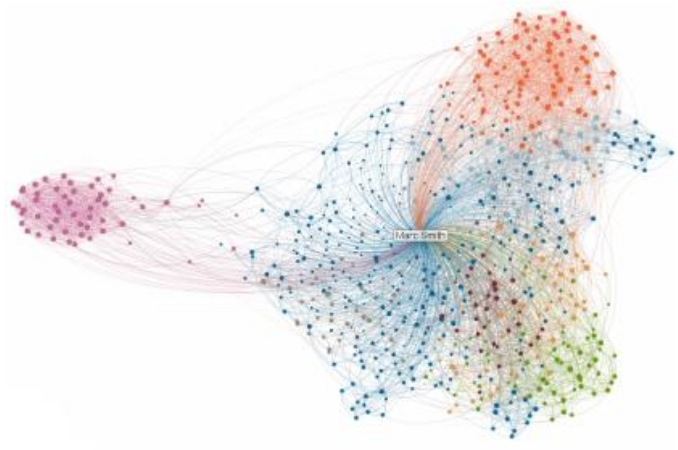
All images are sourced from the paper "Improving Large Graph Processing on Partitioned Graphs in the Cloud" unless specified otherwise.

Importance of Graphs

- Graphs abstract application-specific algorithms into generic problems represented as interactions using vertices and edges



Graph in road network



Graph in social networking

Image source: google.ca/images

Large Graph Processing Systems

- Study of large graphs of $\geq$ 100GB
- Hot and fruitful research area
- Existing Systems:
  - Vertex-oriented execution model
  - Significant amount of network traffic

Aim of the Paper

- Graph partitioning framework to improve network performance of graph partitioning itself.
- Partitioned graph storage.
- Vertex-oriented graph processing.

Result

- *Surfer*.

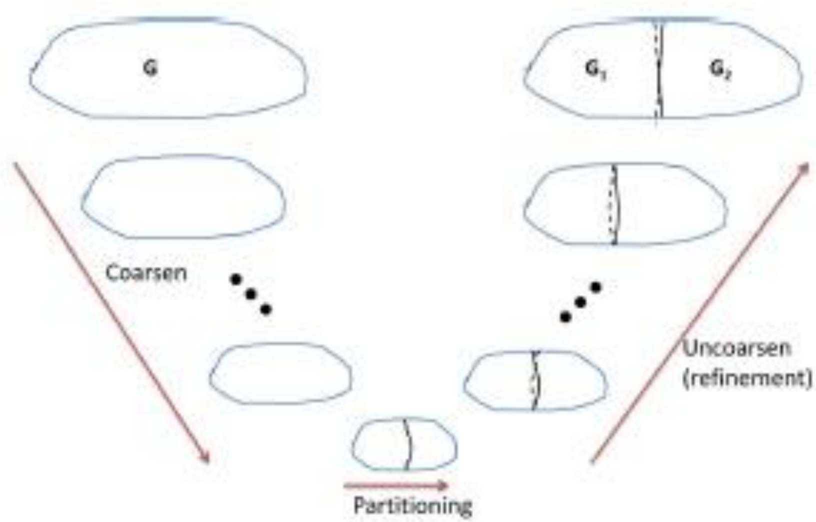
- A bandwidth aware graph partitioning framework to minimize network traffic in partitioning and processing.

- *Surfer - Pregel* (latest vertex-oriented graph engine by Google) extended with graph partitioning framework.

Processing and Partitioning

- Large graph processing:
  - Batched processing on large graphs
  - Network traffic – bottleneck for vertex-oriented computation

- Graph partitioning:
  - Graph bisection
    - Coarsening
    - Partitioning
    - Uncoarsening



Graph Partitioning Framework Models

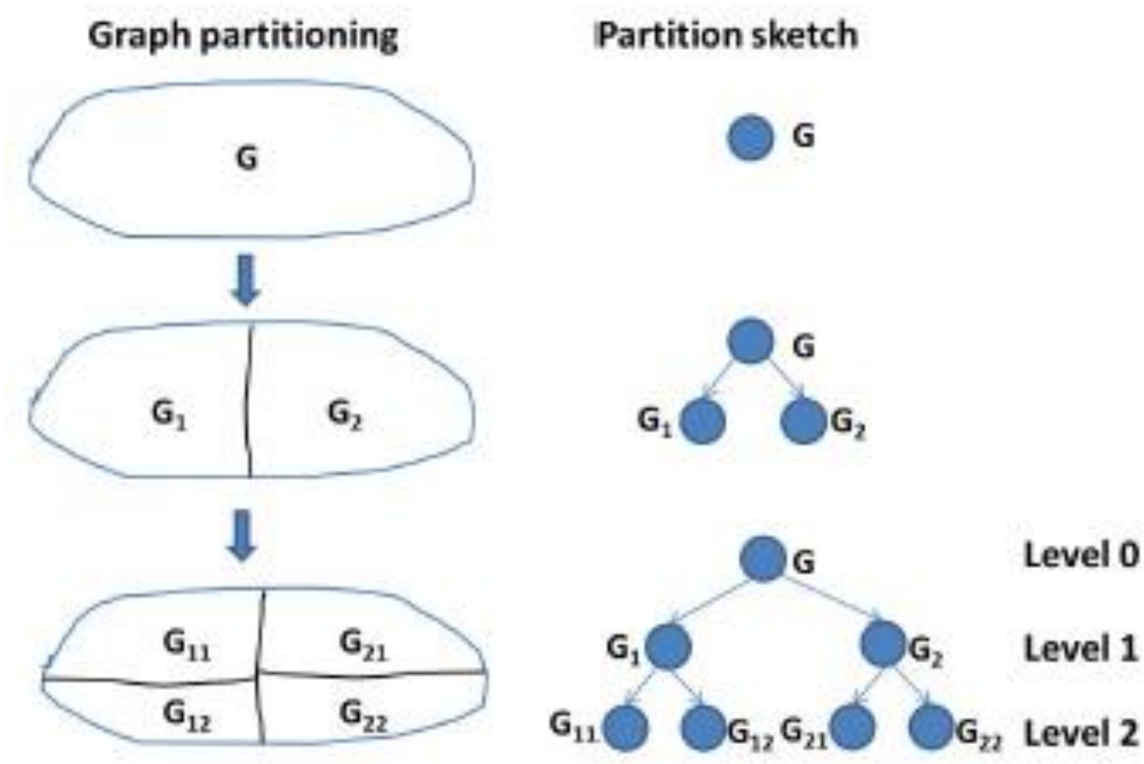
- Partition Sketch
  - Graph partitioning process
- Machine Graph Building
  - Network performance

Partition Sketch

- Basic idea is to partition and store graph partitions according to their # of cross-partition edges.

- Partitions with large # of cross-partition edges are stored in machines with high network bandwidth.

Partition Sketch



Partitioning the graph into four partitions

Partition Sketch

- Properties of ideal partition sketch:
  - Local optimality
    - $C(n_1, n_2)$ – # of cross-partition edges between two nodes n_1 and n_2, $C(n_1, n_2)$ is minimum on all possible bisections of common parent node *p*
  - Monotonicity
    - At the same level $T_i \leq T_j$, if $i \leq j$;

      T is total number of cross-partition edges at the same level.

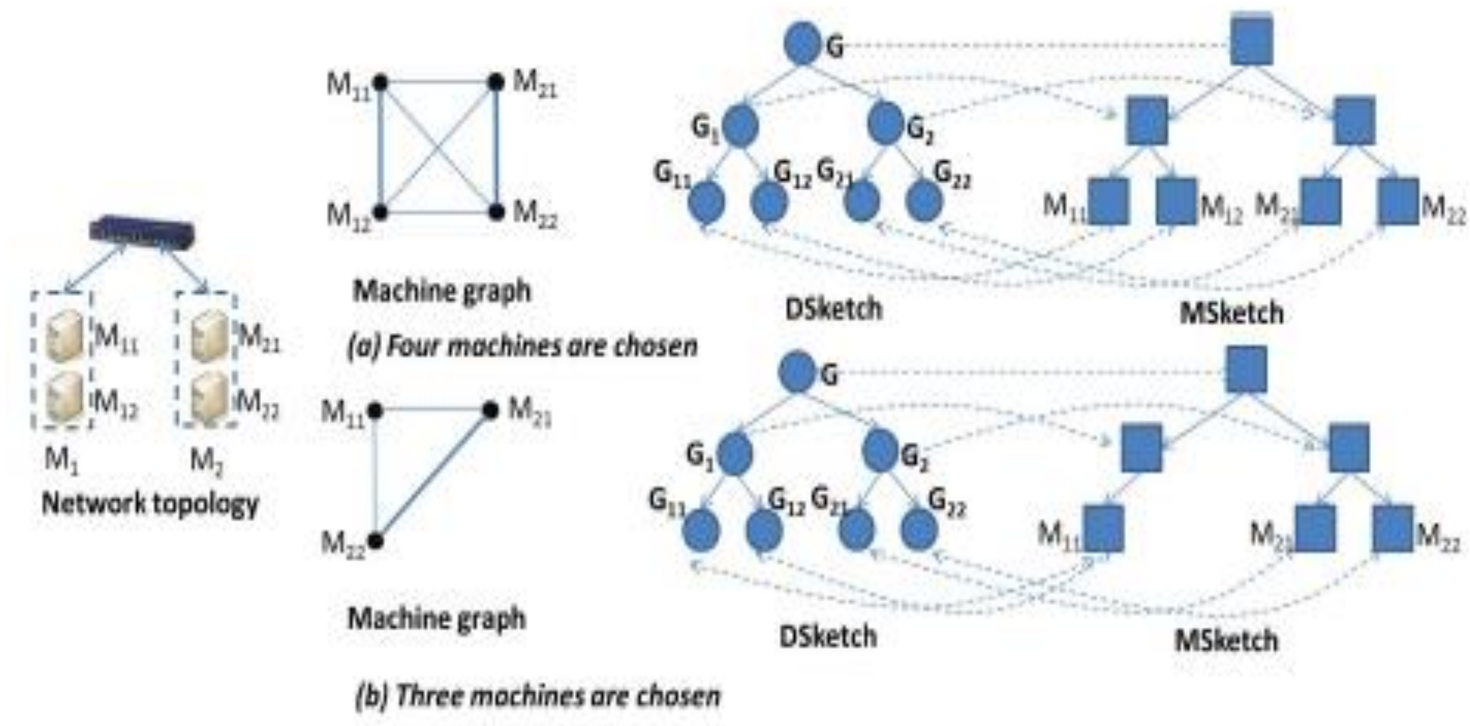
Partition Sketch

- Properties of ideal partition sketch:
  - Proximity
    - Nodes n_1 and n_2 with common parent p
    - Nodes n_3 and n_4 with common parent p'
    - p and p' with same parent
      - $C(n_1, n_2) + C(n_3, n_4) \geq C(n_{\pi(1)}, n_{\pi(2)}) + C(n_{\pi(3)}, n_{\pi(4)})$
      - Π is any permutation on (1, 2, 3, 4)

Machine Graph Building

- Modelled using machine graph
  - Each vertex – machine
  - Each edge – connectivity between machines
  - Weight of the edge is network bandwidth

Machine Graph Building



Mapping on partition sketches between machine graph and data graph

Bandwidth Aware Graph Partitioning

Algorithm 1 Bandwidth aware graph partitioning

Input: A set of machines S in the cloud, the data graph G, the number of partitions P ($L = \log_2 P$)

Description: Partition G into P partitions with S

1: Construct the machine graph M from S;
2: *BAPart*(M, G, 1);//the first level of recursive calls.

Procedures: BAPart(M, G, l)

1: Divide G into two partitions (G_1 and G_2) with the machines in M;
2: **if** M consists of a single machine **then**
3: Let the machine in M be m.
4: Divide G into 2^{L-l} partitions using m with the local partitioning algorithm;
5: Store the result partitions in m;
6: **else**
7: Divide M into two partitions M_1 and M_2;
8: Divide G into two partitions G_1 and G_2 with the machines in M with distributed algorithm [22];
9: *BAPart*(M_1, G_1, l+1);
10: *BAPart*(M_2, G_2, l+1);

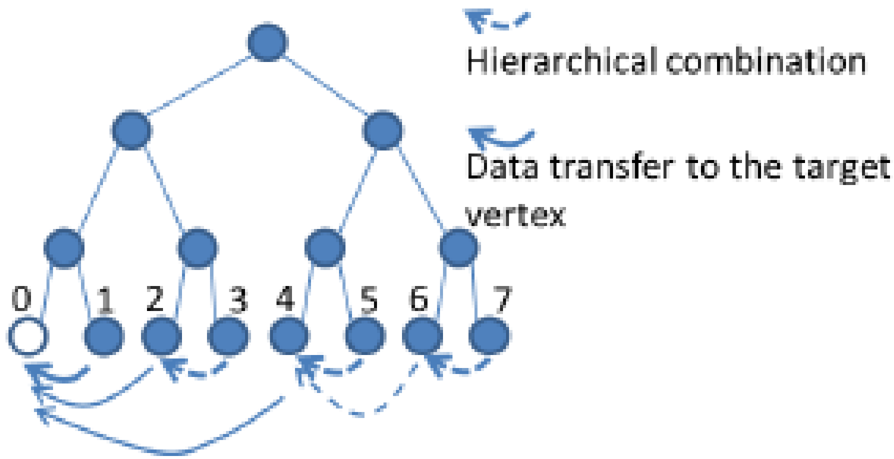
Bandwidth Aware Graph Partitioning

- Partitioning algorithm satisfies the three design principles of:
  - Local optimality
  - Monotonicity
  - Proximity

Hierarchical Combination

- *Local Combination* is a commonly used approach to reduce network traffic.
- *Local Combination* is not aware of the network unevenness in the cloud
- Solution is *Hierarchical Combination*.

Hierarchical Combination



Hierarchical combination of machine graph of eight machines

Experimental Setup

- Conducted experiments on a local cluster (with 32 machines) and Amazon EC2.
- System prototype called *Surfer* implemented in C++, compiled in Visual Studio 9.
- Tree-structured network topology
  - T_2(#pods, #level)

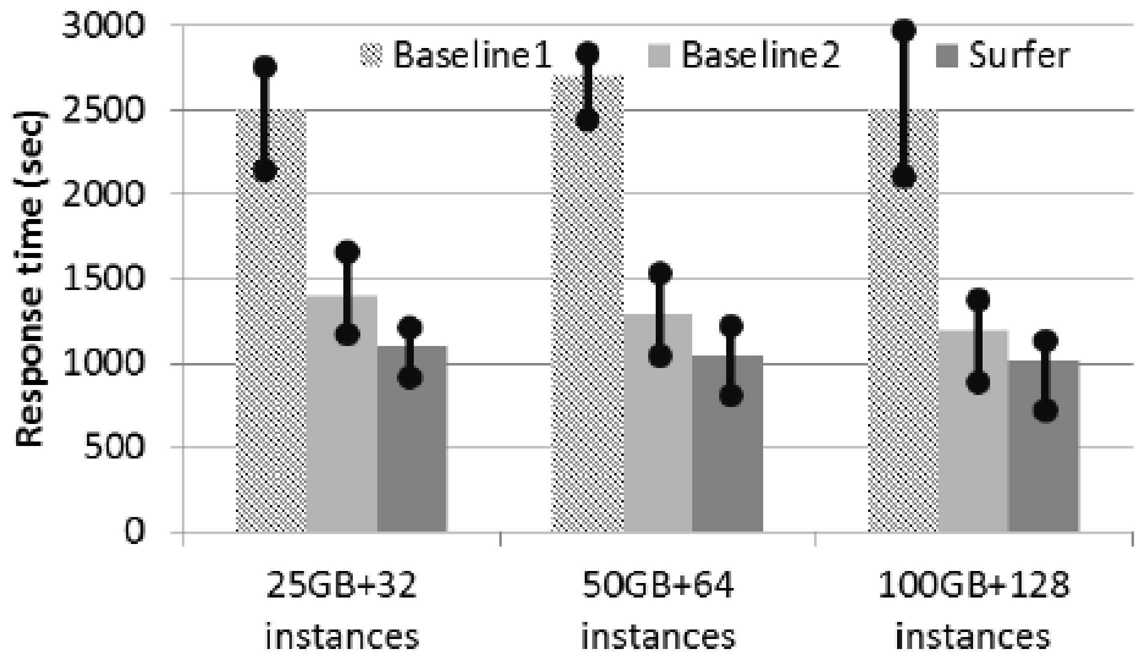
Results on Partitioning

Number of partitions	128	64	32	16
Partition granularity (GB)	1	2	4	8
ier of our partitioning(%)	50.3	57.7	65.5	72.7

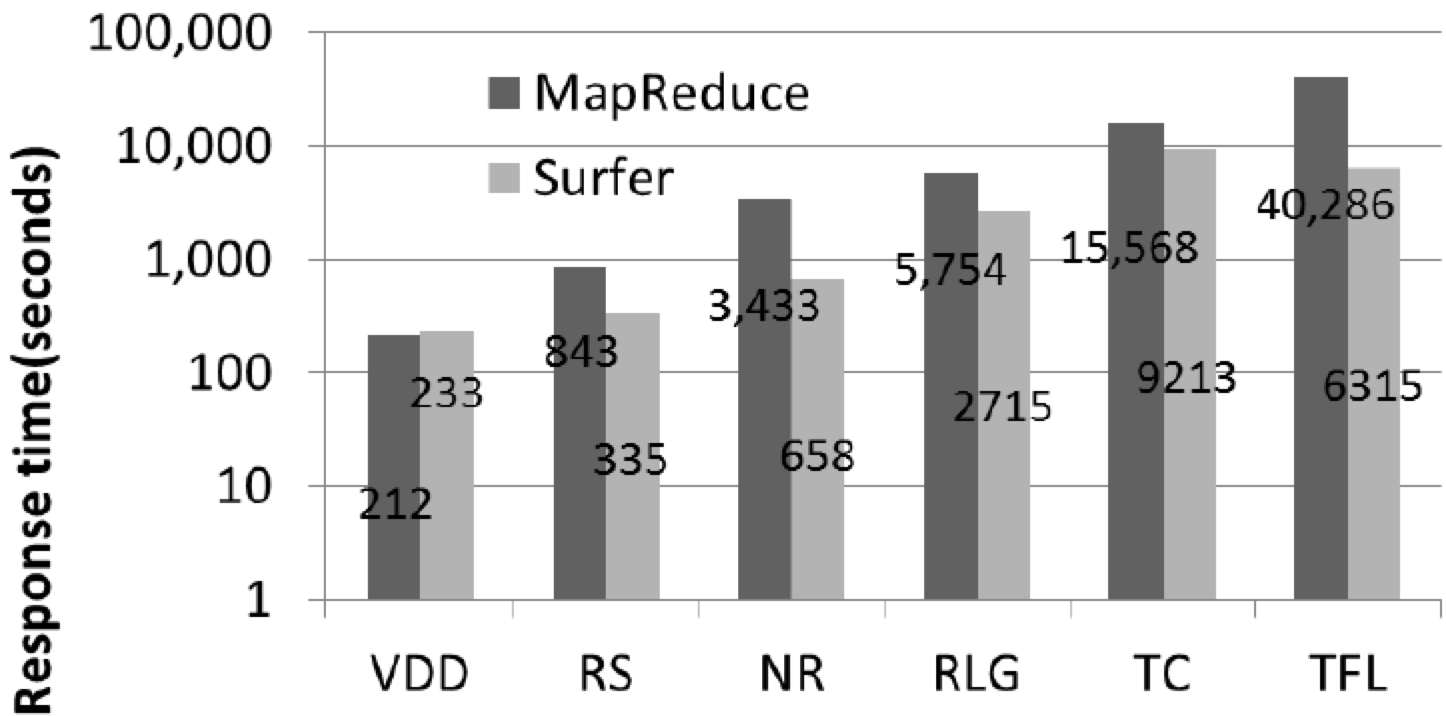
ier: Inner edge ratio

Validates Monotonicity: as depth of partition sketch increases, # of cross-partition edges increases.

Results on Amazon EC2



Comparisons with MapReduce



References

- [1] Improving Large Graph Processing on Partitioned Graphs in the Cloud, R Chen, X Weng, B He, M Yang, B Choi, X Li, 2012
- [2] On the Efficiency and Programmability of Large Graph Processing in the Cloud, R Chen, X Weng, B He, M Yang, B Choi, X Li, 2010

Thank you!