PEGASUS: A peta-scale graph mining system - Implementation and observations

U. Kang, C. E. Tsourakakis, C. Faloutsos
What is Pegasus?

- Open source Peta Graph Mining Library
- Can deal with very large
  - Giga-, Tera-, Peta-byte
- Implemented on top of Hadoop
- Several graph mining operations:
  - PageRank, Random Walk with Restart, Diameter estimation, Connected components
- Uses GIM-V (Generalized Iterated Matrix-Vector multiplication)
Three Primitives (xG):

1) combine2($m_{i,j}$, $v_j$) : combine $m_{i,j}$ and $v_j$.
2) combineAll$_i(x_1, ..., x_n)$ : combine all the results from combine2() for node $i$.
3) assign($v_i$, $v_{new}$) : decide how to update $v_i$ with $v_{new}$.

Iterative:

- Operation applied till algorithm-specific convergence criterion is met.
PageRank $p$ of $n$ web pages given by:

$$p = (cE^T + (1 - c)U)p$$

$c = \text{Damping Factor (0.85)}$

$E = \text{row-normalised adjacency matrix (src, dest)}$
GIM-V - PageRank (cont)

- Direct application of GIM-V
- Construct matrix $M$ by column-normalise $E^T$
  - each column of $M$ sums to 1
- $p$ calculated by $M \times G \times p^{cur}$

1) $\text{combine2}(m_{i,j}, v_j) = c \times m_{i,j} \times v_j$
2) $\text{combineAll}_i(x_1, ..., x_n) = \frac{(1-c)}{n} + \sum_{j=1}^{n} x_j$
3) $\text{assign}(v_i, v_{new}) = v_{new}$
GIM-V BASE

- 2-stage algorithm with 2 Map-Reduce in each stage
- Input: Edge and Vector file
  - Edge line: \((id_{src}, id_{dst}, mval)\) -> cell adjacency Matrix M
  - Vector line: \((id, vval)\) -> element in Vector V

1. Stage1 performs combine2() on columns of \(id_{dst}\) of M with rows of id of V
2. Stage2 combines all partial results and assigns new vector -> old vector
3. The combineAlli() and assign() operations are done later in Stage2
4. Run iteratively until application-specific convergence criterion is met
GIM-V Block Multiplication (BL)

- Group elements of input matrix in submatrices of size $b \times b$
- Group elements of vectors in length $b$
- Make them fit into 1 line of input file
- Only non-zero elements
- Forces nearby edges to be closely stored
- 5 times faster
  - Sorting time
  - Compression
Block Multiplication allows use of Cluster Edges
Smaller number of blocks for input (if clustered)
Preprocessing done only once, used in all further iterations
GIM-V Diagonal Block Iter (DI)

- Reduces runtime by reducing iterations -> less disk IO
- Multiplies diagonal matrix blocks and corresponding vector blocks
  - As much as possible in one iteration -> till content not change
- Pass id to neighbours located more steps away
Performance and Scalability

- Run Pegasus on M45 cluster by Yahoo!
  - In top 50 supercomputers
  - 1.5 Pb Storage
  - 3.5 Tb Memory
  - Used synthetic graphs (Kronecker)
Results - PageRank

- Running time decreases with more machines
- Clustering edges does not performed if not combined with Block Encoding
- Relative performance decreases with BASE as machines increase
  - (fixed costs) 3 machines 5.27x, 90 machines 2.93
- All scale linearly with size of input
GIM-V DI vs BL-CL

- Used Connected Components
- Diameter 17 with 282M edges
- 6 Iterations vs 18
Real Graph Analysis

- Power law tails in connected components
- Stable connected components after gelling point
- Absorbed connected components and Dunbar's number
Anomalous connected components:
- First Spike: Domain selling company -> sites replicated from same template
- Second Spike: Porn sites disconnected from giant connected components (80%)
  - This are special purpose communities disconnected from rest of Internet
- PageRank of YahooWeb follows a power law distribution with exponent 1.97, close to exponent 1.98 (from previous research in smaller networks).
- Observation holds true for 10,000 times larger network with 1.4 billion pages snapshot of the Internet.
Diameter - Real Networks

- LinkedIn 2003: Avg Diameter: 5.89
- LinkedIn 2004: Avg Diameter: 6.09
- LinkedIn 2005: Avg Diameter: 5.28
- LinkedIn 2006: Avg Diameter: 5.56


- DBLP doc-doc: Avg Diameter: 2.77
- Flickr: Avg Diameter: 3.72
- Epinions: Avg Diameter: 3.82
Contributions

- Authors present new primitives to allow analysis of graphs
- Give various algorithms that operate with those primitives
- Several optimisations for the algorithms
- New results about very large networks
• Examples for the algorithms could have been more step-by-step
• The paper has a lot of information for its size (bit terse)
• Largest performance claim is based on using 3 machines?