Scaling Out Link Prediction with SNAPLE: 1 Billion Edges and Beyond

Anne-Marie Kermarrec, François Taïani, Juan M. Tirado
Netflix never used its $1 million algorithm due to engineering costs

By Casey Johnston | Published April 13, 2012 4:25 PM

Netflix awarded a $1 million prize to a developer team in 2009 for an algorithm that increased the accuracy of the company's recommendation engine by 10 percent. But today it doesn't use the million-dollar code, and has no plans to implement it in the future, Netflix announced on its blog Friday. The post goes on to explain why: a combination of too much engineering effort for the results, and a shift from movie recommendations to the "next level" of personalization caused by the transition of the business from mailed DVDs to video streaming.
Netflix never used its $1 million algorithm due to engineering costs

By Casey Johnston | Published April 13, 2012 4:25 PM

Netflix awarded a $1 million prize to a developer team in 2009 for an algorithm that increased the accuracy of the company's movie recommendation engine by 10 percent. But today it doesn't use the million-dollar code, and has no plans to implement it in the future, Netflix announced on its blog Friday. The post goes on to explain why a combination of too much engineering effort to the "next level" of personalization caused by the transition of the business from mailed DVDs to video streaming.
Netflix never used its $1 million algorithm due to engineering costs

By Casey Johnston | Published April 13, 2012 4:25 PM

Netflix awarded its the developer team in 2009 for an algorithm that increased the accuracy of the company’s movie recommendations by 10 percent. But today it doesn't use the million-dollar code, and has no plans to implement it in the future, Netflix announced on its blog Friday. The post goes on to explain why a combination of too much engineering effort and the transition of the business from mailed DVDs to video streaming.
Message and Plan

Tools impose what tasks you can solve

This talk

- The Tool: Gather Apply Scatter Graph Engines
- The Task: Link Prediction (LP) in Large Graphs
Message and Plan

Tools impose what tasks you can solve

This talk

- The Tool: **Gather Apply Scatter Graph Engines**
- The Task: **Link Prediction (LP) in Large Graphs**

F. Taiani
The GA(S) Model

- Gather Apply Scatter
  ➔ Adaptation of Bulk Synchronous Parallel (BSP) to graphs

Computations

Communication

Synchronization Barrier

Time
The GA(S) Model

Think like a vertex (Pregel, Giraph, GraphLab, ...)

Example: "Top net worth in my 1-hop network."

Simple

Local

Highly Parallelizable & Scalable

(scatter not shown)
The GA(S) Model

Limitations

- Not good at **non-local** operations
Non Local Computations

- GAS engines provide **horizontal scalability**

Non-locality: Network and memory costs

Too much non-locality: scalability hit

1000€
This talk

- The Tool: Gather Apply Scatter Graph Engines
- The Task: Link Prediction (LP) in Large Graphs
Link prediction
Link prediction is everywhere
Datasets sizes keep growing
Can GAS Graph Engine help scale LP?
Link Prediction

- Twin Goals
  - Good predictions (✔)
  - Fast (⌚)

Observed graph → "True" graph
Link Prediction

- Twin Goals
  - Good predictions (✔)
  - Fast (⌚)

Observed graph

Predicted graph

“True” graph

Predict

1/3
A classification problem

Does $e = (\text{Penguin}, \text{Girl})$ belong to True Graph?

Typical: Scoring exploiting topological information

$0.5$
Link Prediction

- A classification problem
  - Does $e = (\text{企鹅}, \text{人})$ belong to True Graph?
  - Typical: Scoring exploiting topological information

- $(\text{企鹅}, \text{人}) \rightarrow 0.5 \checkmark$
- $(\text{企鹅}, \text{机器人}) \rightarrow 0.0 \times$

F. Taiani
Porting Link Prediction to GAS

- **Challenge 1:** Random graph access impossible
  - Strategy: reduce search space to Friends-of-Friends
  - Avoid $\sim N^2$ score computation

- **Challenge 2:** Obtaining topological information is costly
  - It must be propagated.
The Propagation Problem
The Propagation Problem

3 Supersteps

Lots of data passed around

Poor performance & Scalability

1

2

3
Snaple

Intuition

→ Propagate similarities, not neighborhoods

\[(\text{Penguin} \ , \ \text{Woman}) = (\text{Penguin} \ , \ \text{Man}) \otimes (\text{Man} \ , \ \text{Woman})\]

\[\{ (\text{Man} \ , \ \text{Woman}) : 0.5 \}\]

(\text{Man} \ , \ \text{Woman}) \rightarrow 0.5

F. Taiani
Additional Complexities

- Which combination operator $\otimes$?
  - Linear combination is best

- Multiple paths
  - Path aggregation. Sum is best.

- Very large neighborhoods (e.g. up to $10^4$ neighbors)
  - Sampling above threshold size
  - Selection of best neighbors

- Details are in the paper
Evaluation

- **GraphLab Implementation**

- **5 datasets:**
  - 1 outgoing edge removed per vertex

| dataset       | $|V|$   | $|E|$    | domain              |
|---------------|-------|---------|---------------------|
| gowalla [8]   | 196,591 | 0.95M   | social network      |
| pokec [40]    | 1.6M   | 30.6M   | social network      |
| orkut [28]    | 3M     | 223M    | social network      |
| livejournal [2] | 4.8M   | 68.9M   | co-authorship       |
| twitter-rv [18] | 41M    | 1.4B    | microblogging       |

- **Similarity: Jaccard (+ linear comb. and sum)**

- **Baseline: Direct implementation in GraphLab**
Computation Time

4 nodes, 80 cores per node, 128 GBytes of memory, 10-Gigabit ethernet

Average speedup: $\times 34.3$
Recall

![Bar chart showing recall for Gowalla, Pokec, and Livejournal]

**Average recall gain: \( \times 2.17 \)**
Scalability

1.4 billion edges processed in 2m57s on 160 cores

Scales linearly w.r.t. graph size

4 or 8 nodes, 2 Intel Xeon E5-2660v2 (2.2 Ghz) per node (20 cores per node)
Also in the paper

- More combinator / aggregator combinations
- Effect of sampling and selection
- Recall – speed tradeoff analysis
- Single-machine comparison with Cassovary
Conclusion

- Graph engines not well adapted to all algorithms
  - Substantial adaptation might be required
- Snaple: a framework for GAS-based link-prediction
- Large design space
  - We have only explored a small part
- Future work
  - Supervised similarities
  - Supervised combinator / aggregator design
- Ports to other GAS engines
  - Apache Giraph, Spark GraphX
Thank you
Questions?
\[ \hat{\Gamma} (a) = \{b, c, d\} \]
\[ b : f \left( \hat{\Gamma}(a), \hat{\Gamma}(b) \right) \]
\[ c : f \left( \hat{\Gamma}(a), \hat{\Gamma}(c) \right) \]
\[ d : f \left( \hat{\Gamma}(a), \hat{\Gamma}(d) \right) \]
\begin{align*}
\text{a.sims}[b] \otimes \text{b.sims}[e] \\
\text{a.sims}[b] \otimes \text{b.sims}[f] \\
\text{a.sims}[c] \otimes \text{c.sims}[g] \ldots
\end{align*}