Foster City, Silicon Valley
6th-10th October 2014

GASGD: Stochastic Gradient Descent for Distributed Asynchronous Matrix Completion via Graph Partitioning

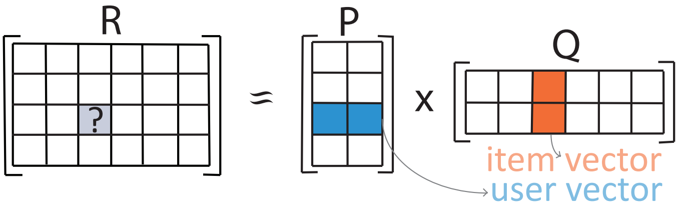
Fabio Petroni *and Leonardo Querzoni*

Sapienza Università di Roma

CIS Sapienza Cyber Intelligence and Information Security

Department of Computer Control and Management Engineering Antonio Ruberti, Sapienza University of Rome - petroni|querzoni@dis.uniroma1.it

Matrix Completion & SGD



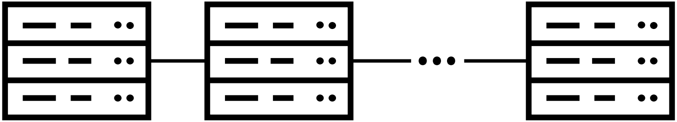
$$LOSS(P, Q) = \sum (R_{ij} - P_i Q_j)^2 + \ldots$$

- **Stochastic Gradient Descent** works by taking steps proportional to the negative of the gradient of the LOSS.
- *stochastic* = P and Q are updated for each given training case by a small step, toward the average gradient descent.

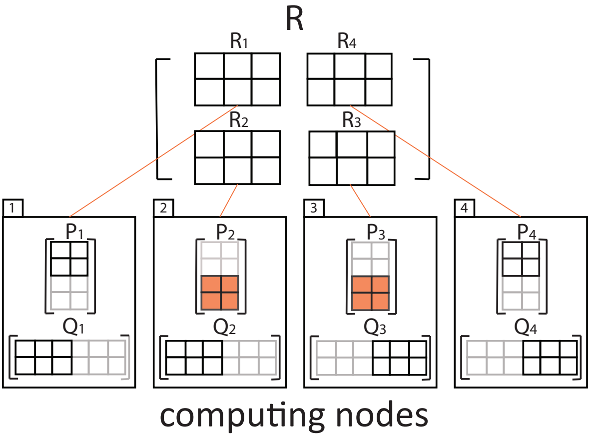
Scalability

- ✗ Lengthy training stages;
- ✗ high computational costs;
- ✗ especially on large data sets;
- ✗ input data may not fit in main memory.

- *goal* = efficiently exploit computer cluster architectures.



Distributed Asynchronous SGD



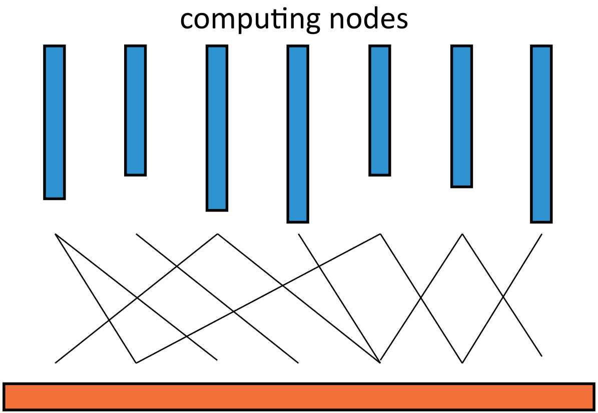
- R is splitted;
- vectors are replicated;
- replicas concurrently updated;
- replicas deviate inconsistently;
- synchronization.

Bulk Synchronous Processing Model

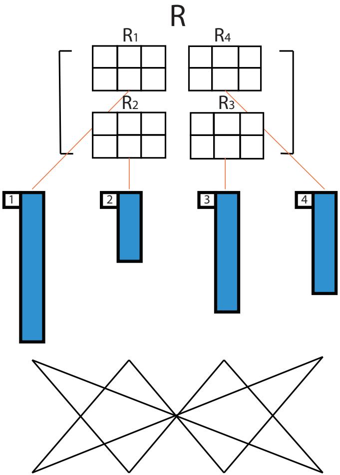
1. local computation

2. communication

3. barrier synchronization

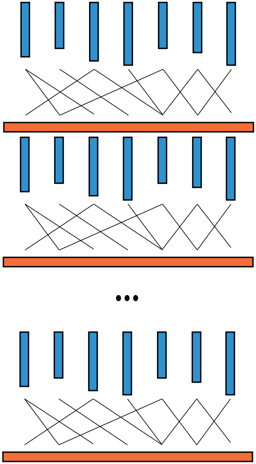


Challenges 1/2



- **1. Load balance**
  - ensure that computing nodes are fed with the same load.
- **2. Minimize communication**
  - minimize vector replicas.

Challenges 2/2



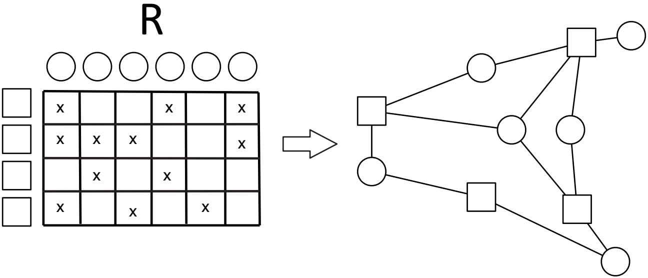
- **3. Tune synchronization frequency** among computing nodes.
- Current implementations synchronize vector copies:
  - continuously during the epoch (*waste of resources*);
  - once after every epoch (*slow convergence*).
- *epoch* = a single iteration over the ratings.

Contributions

- ✓ We **mitigate the load imbalance** by proposing an input slicing solution based on graph partitioning algorithms;
- ✓ we show how to **reduce the number of shared data** by properly leveraging known characteristics of the input dataset (bipartite power-law nature);
- ✓ we show how to leverage the tradeoff between communication cost and algorithm convergence rate by **tuning the frequency** of the bulk synchronization phase during the computation.

Graph representation

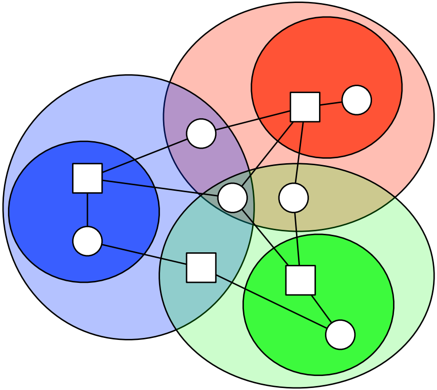
- The rating matrix describes a bipartite graph.



- Real data: skewed power-law degree distribution.

Input partitioner

- *vertex-cut* performs better than *edge-cut* in power-law graphs.



- Assumption: the input data doesn't fit in main memory.
- Streaming algorithm.
- Balanced k-way vertex-cut graph partitioning:
  - minimize replicas;
  - balance edge load.

Balanced Vertex-Cut Streaming Algorithms

- **Hashing**: pseudo-random edge assignment;
- **Grid**: shuffle and split the rating matrix in identical blocks;
- **Greedy**: *[Gonzalez et al. 2012]* and *[Ahmed et al. 2013]*.

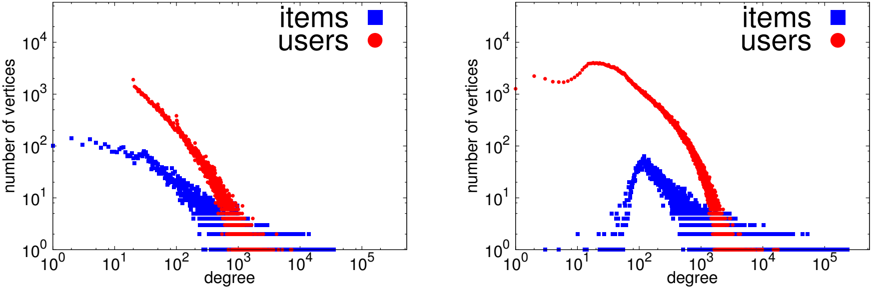
Bipartite Aware Greedy Algorithm

- Real word bipartite graphs are often significantly skewed: one of the two sets is much bigger than the other.
- By perfectly splitting the bigger set it is possible to achieve a smaller replication factor.

- **GIP** (Greedy - Item Partitioned)
- **GUP** (Greedy - User Partitioned)

Evaluation: The Data Sets

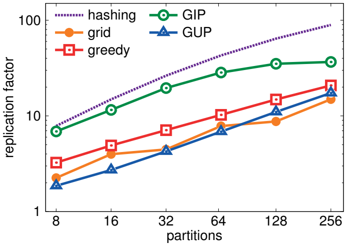
Degree distributions:

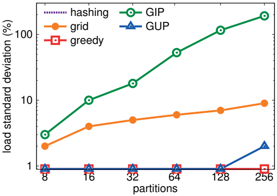


MovieLens 10M　　　　Netflix

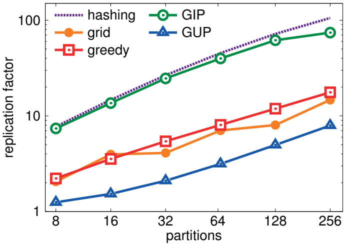
Experiments: Partitioning quality

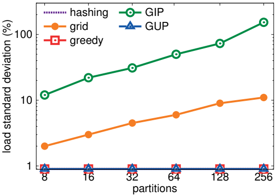
MovieLens





Netflix



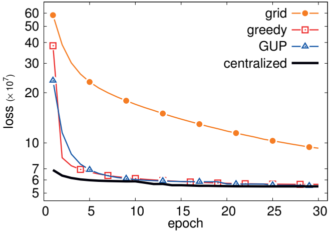


- RF
  Replication Factor
- RSD
  Relative Standard Deviation

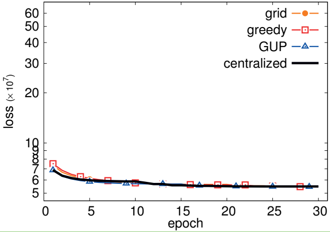
Synchronization frequency

Netflix

$f = 1$



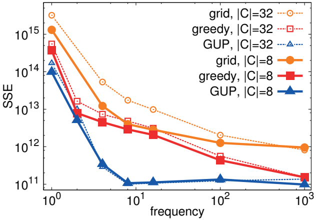
$f = 100$



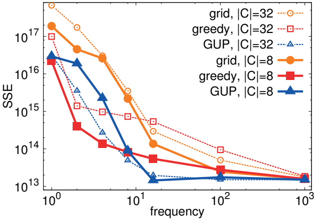
- **f** = synchronization frequency parameter
  - number of synchronization steps during an epoch.
- tradeoff between communication cost and convergence rate.

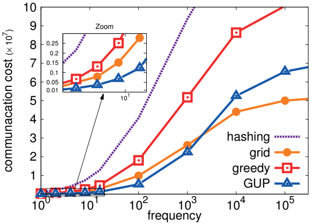
Evaluation: SSE and Communication cost

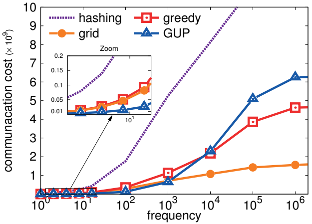
MovieLens



Netflix







- SSE

  between ASGD variants and SGD curves

- CC

Communication cost

- T = the training set
- U = users set
- I = items set
- $V = U \cup I$
- C = processing nodes
- RF = replication factor
- RF_U = users' RF
- RF_I = items' RF

$$RF = \frac{|U|RF_U + |I|RF_I}{|V|}$$

$$f = 1 \rightarrow \quad CC \approx 2(|U|RF_U + |I|RF_I) = 2|V|RF$$

$$f = \frac{|T|}{|C|} \rightarrow \quad CC \approx |T|(RF_U + RF_I)$$

Conclusions

- three distinct contributions aimed at improving the efficiency and scalability of ASGD:

1. we proposed an input slicing solution based on graph partitioning approach that mitigates the load imbalance among SGD instances (i.e. better scalability);
2. we further reduce the amount of shared data by exploiting specific characteristics of the training dataset. This provides lower communication costs during the algorithm execution (i.e. better efficiency);
3. we introduced a synchronization frequency parameter driving a tradeoff that can be accurately leveraged to further improve the algorithm efficiency.

Thank you!

Questions?

Fabio Petroni

Rome, Italy

Current position:

PhD Student in Computer Engineering, Sapienza University of Rome

Research Areas:

Recommendation Systems, Collaborative Filtering, Distributed Systems

petroni@dis.uniroma1.it