RINGO: A System for Interactive Graph Analytics

Jure Leskovec (@jure)

Including joint work with Y. Perez, R. Sosič, A. Banarjee, M. Raison, R. Puttagunta , P. Shah

Background

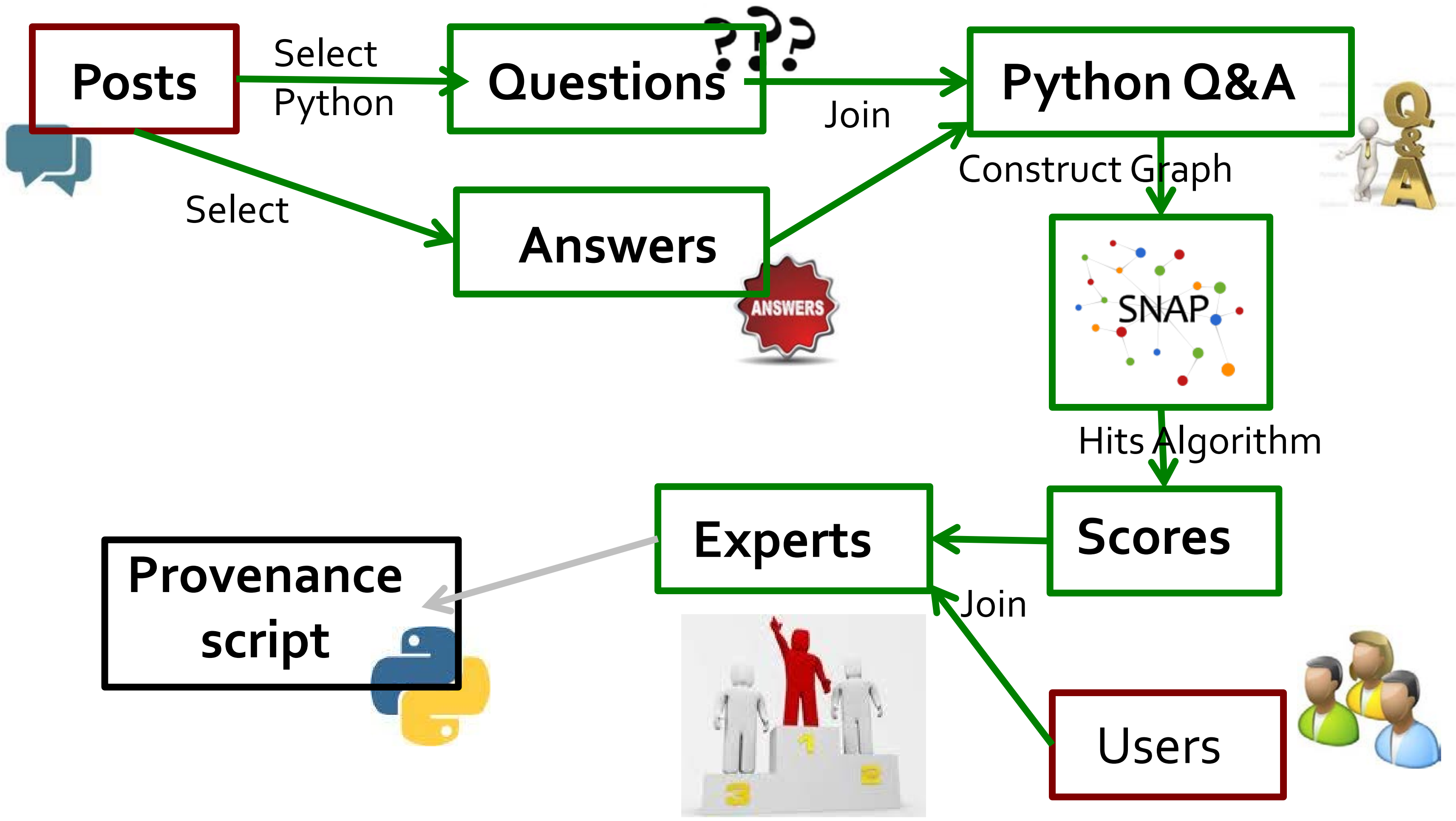
My research group at Stanford:

- Mining and modeling large social and information networks
- Problems motivated by the Web and on-line media, large scale data
- We work with networks from FB, Yahoo, Twitter, LinkedIn, Wikimedia, StackOverflow

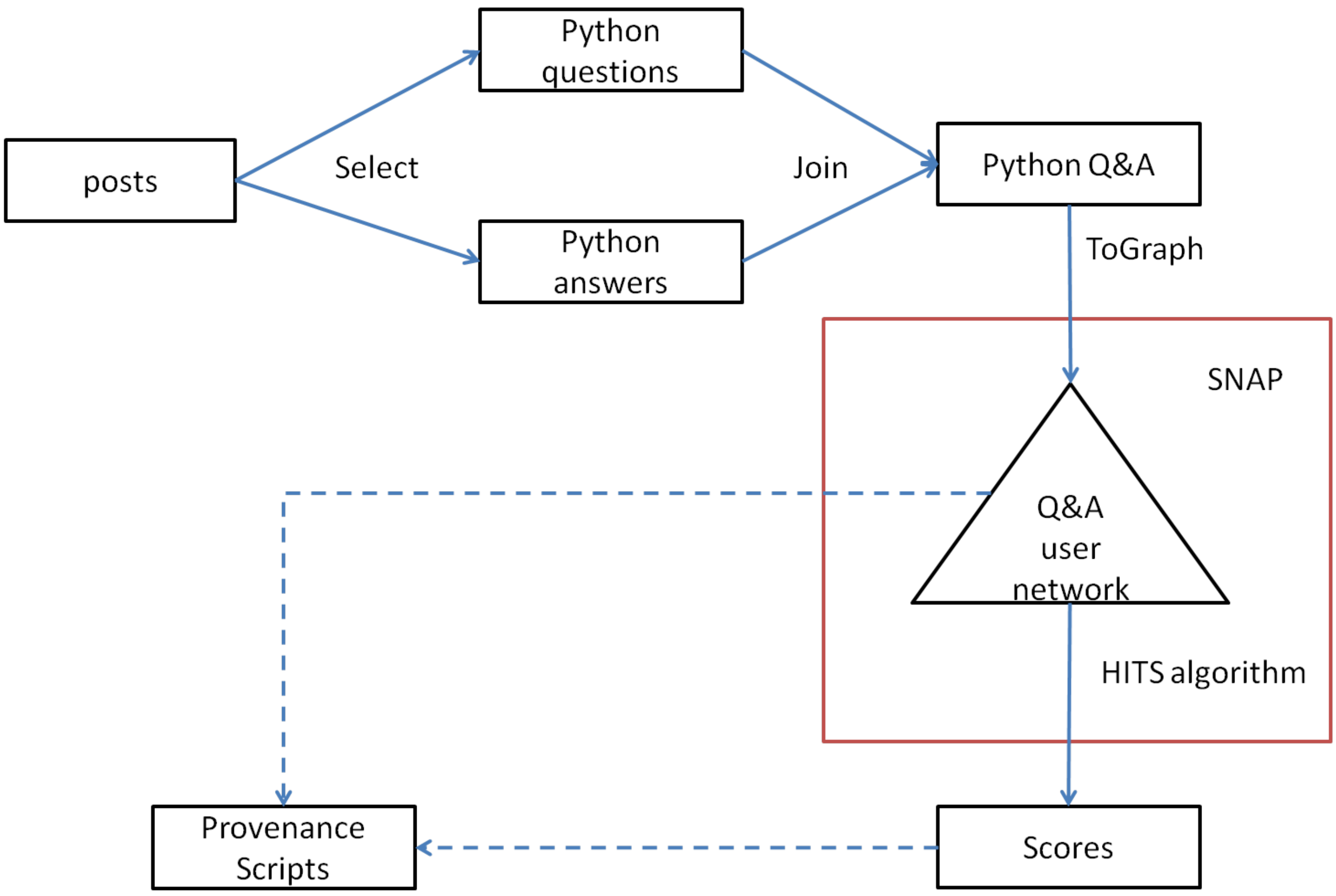
This talk:

Feedback appreciated!

Experts on StackOverflow



Experts on StackOverflow



Observation

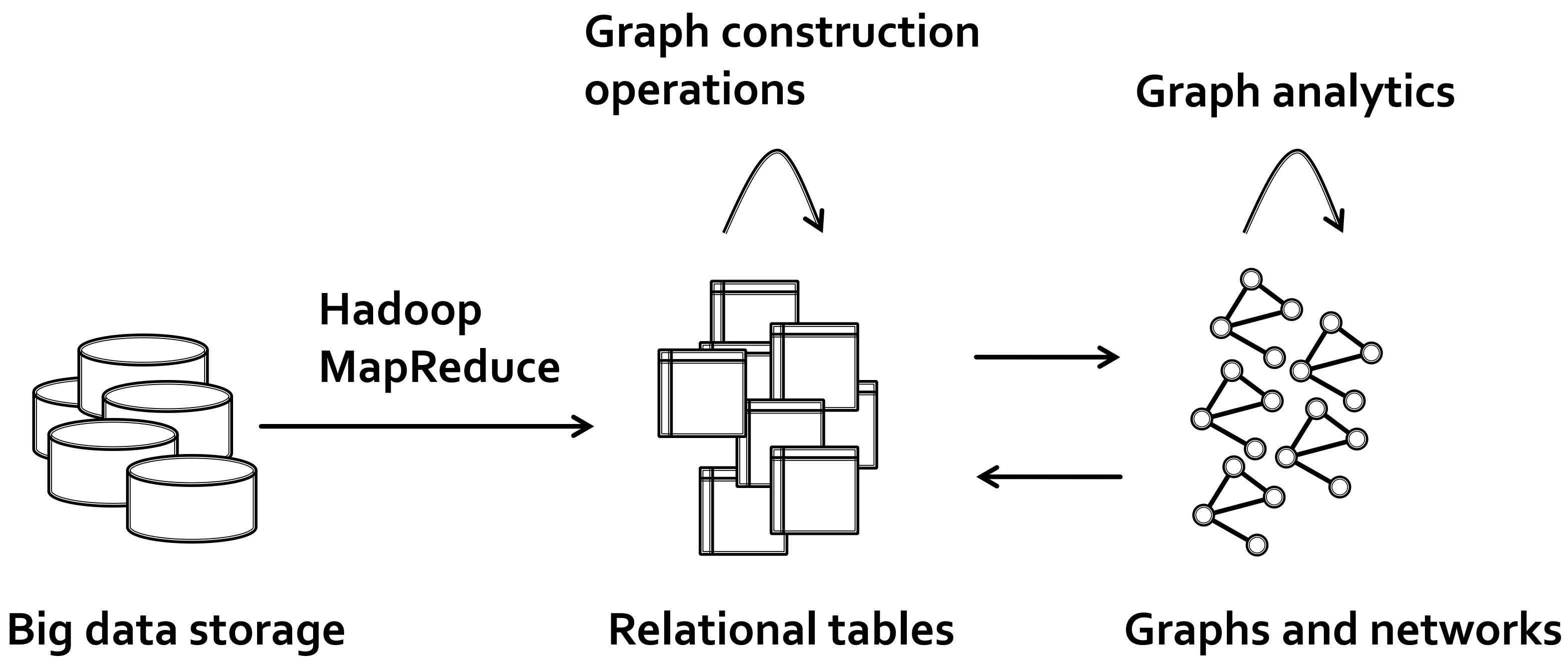
Graphs are almost never given but they have to be constructed from input data!

(graph constructions is a part of modeling/discovery process)

Examples:

- **Facebook graphs:** Friend, Communication, Poke, Co-Tag, Co-location, Co-Event
- **Cellphone/Email graphs:** How many calls?
- **Biology:** P2P, Gene interaction networks

Graph Analytics Workflow



Desiderata for Graph Analytics

Easy to use front-end

- Common high-level programming language

Fast execution times

- Interactive use (as opposed to batch use)

Ability to process large networks

- Tens of billions of edges

Support for several representations

- Transformations between tables and graphs

Large number of graph algorithms

- Straightforward to use

Workflow management and reproducibility

- Provenance

Machines and Graph Sizes

Two observations:

(1) Most graphs are not that large

(2) Big-memory machines are here: 4x Intel CPU, 64 cores, 1TB RAM, **$35K**

Number of Edges	Number of Graphs
<0.1M	16
0.1M – 1M	25
1M – 10M	17
10M – 100M	7
100M – 1B	5
> 1B	1

Stanford Large Network Collection
71 graphs

Trade-offs

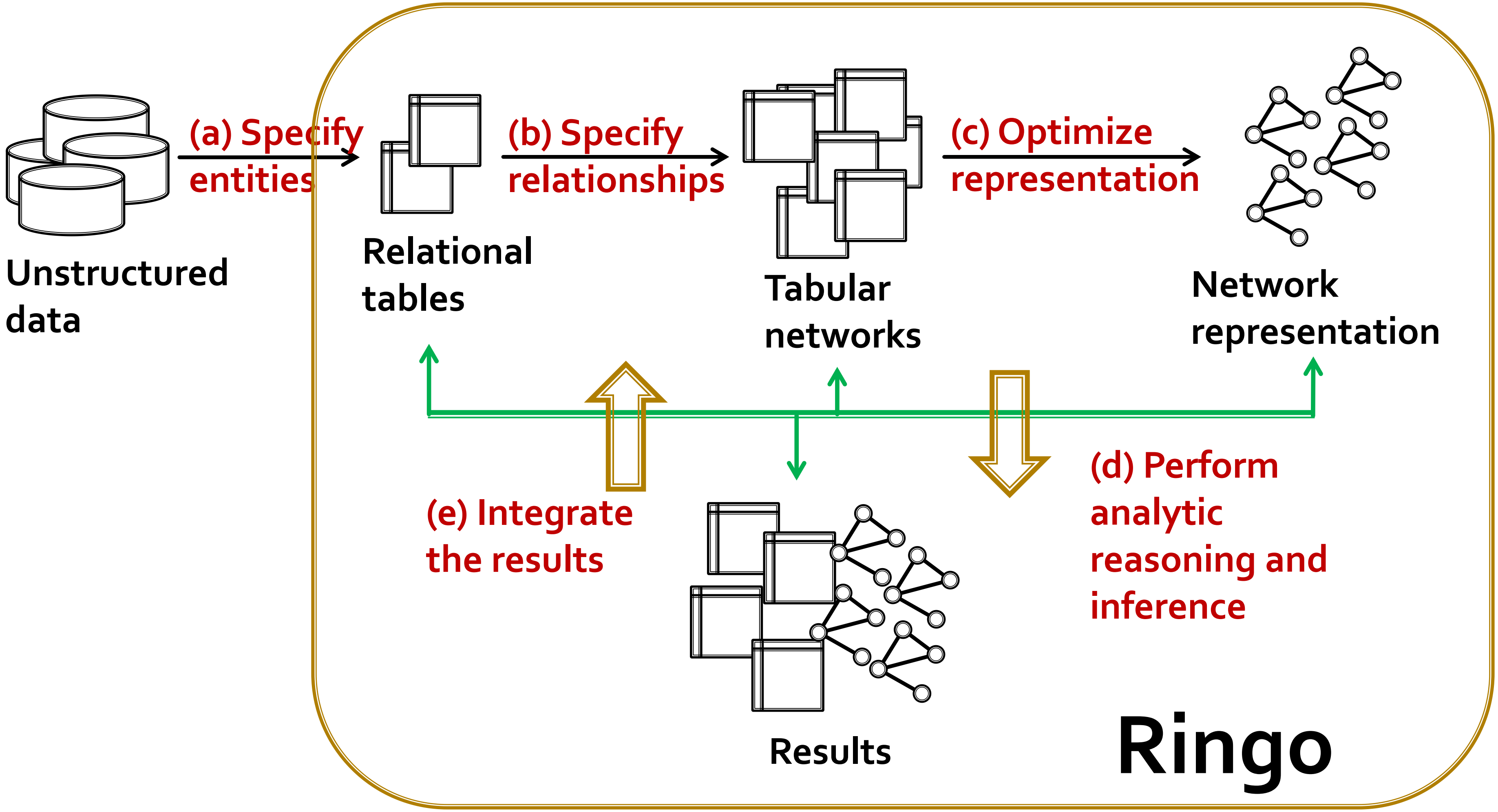
Option 1	Option 2
Standard SQL database	Custom representations
Separate systems for tables and graphs	Integrated system for tables and graphs
Single representation for tables and graphs	Separate table and graph representations
Distributed system	Single machine system
Disk based structures	In-memory structures

Trade-offs

Option 1	Option 2
Standard SQL database	**Custom representations**
Separate systems for tables and graphs	**Integrated system for tables and graphs**
Single representation for tables and graphs	**Separate table and graph representations**
Distributed system	**Single machine system**
Disk based structures	**In-memory structures**

Ringo

Graph Analytics: Ringo

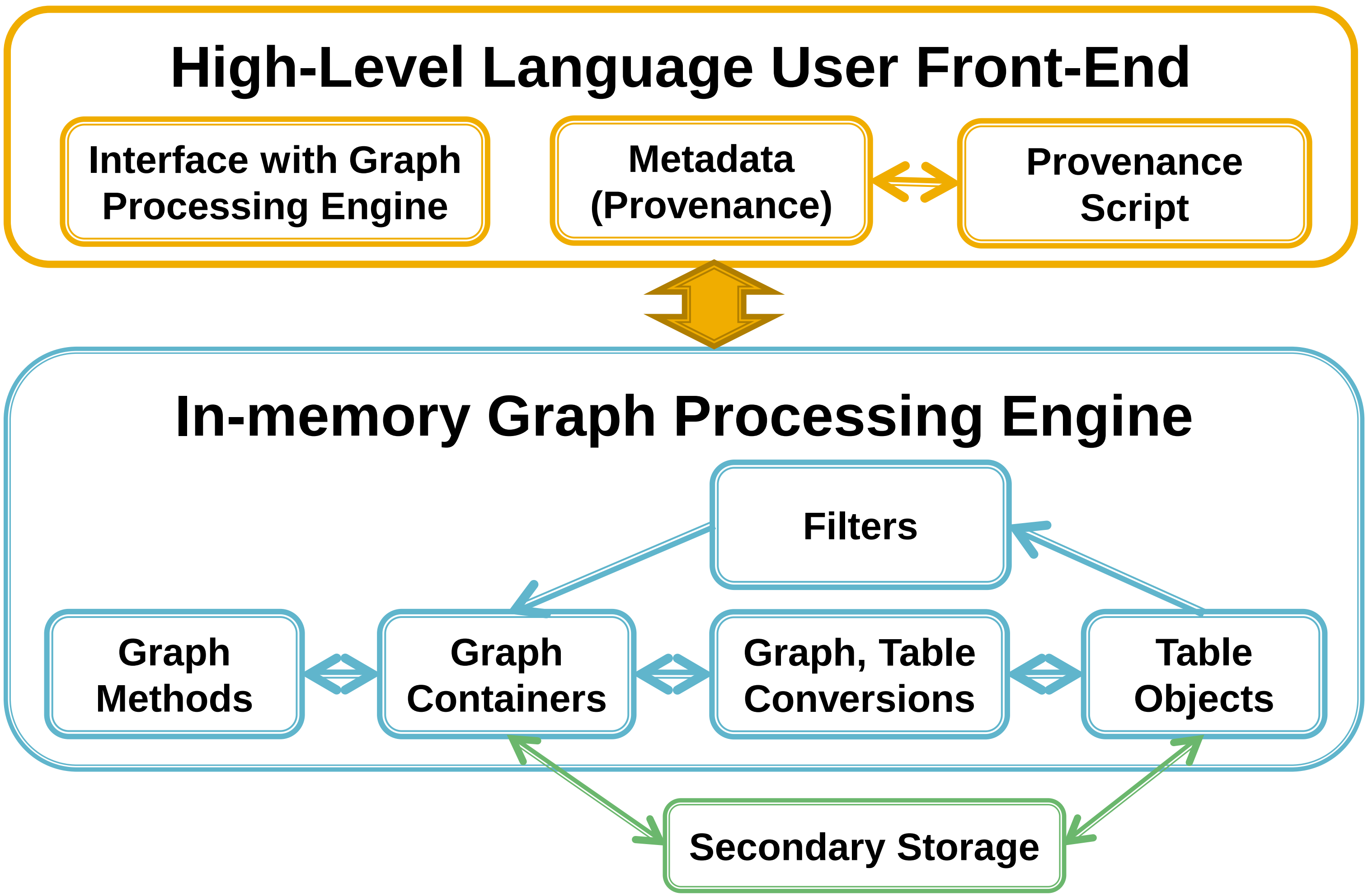


Ringo!

- **Ringo (Python) code for executing finding the StackOverflow example**

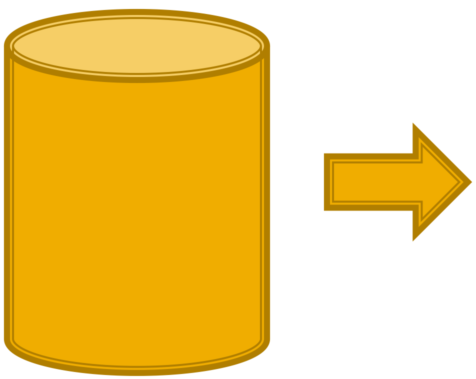
```
P = ringo.LoadTable(schema,'posts.tsv')
JP = ringo.Select(P,'Tag=Java')
Q = ringo.Select(JP,'Type=question')
A = ringo.Select(JP,'Type=answer')
QA = ringoJoin(Q,A,'AnswerId','PostId')
G = ringo.ToGraph(QA,'UserId.1','UserId.2')
PR = ringo.GetPageRank(G)
S = ringo.ToTable(PR,'UserId','Score')
ringo.Save(S,'scores.bin')
```

Ringo Overview



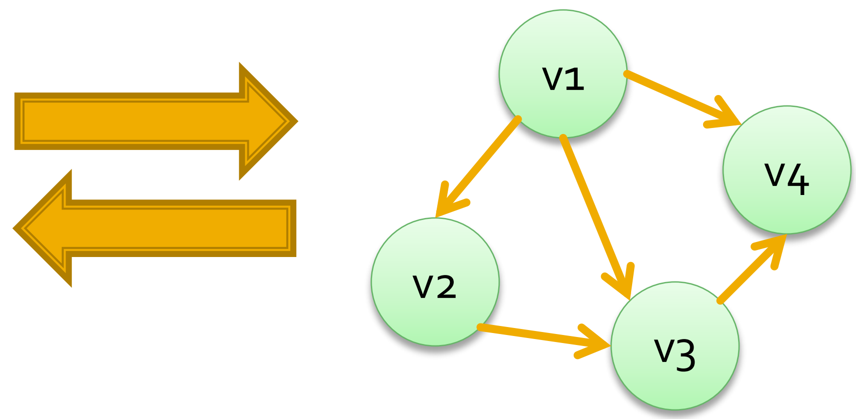
Graph Construction

- **Input data must be manipulated and transformed into graphs**



Src	Dst	...
v1	v2	...
v2	v3	...
v3	v4	...
v1	v3	...
v1	v4	...

Table data structure



Graph data structure

Creating a Graph in Ringo

- **Four ways to create a graph:**
  - The data already contains edges as source and destination pairs
  - Nodes are connected based on their similarity
  - Nodes are connected based on a temporal order
  - Nodes are connected based on grouping and aggregation

Edge Filters in Ringo

- **Use case: In a forum, connect users that post to similar topics:**
  - Distance metrics
    - Euclidean, Haversine, Jaccard distance
  - Connect similar nodes
    - *SimJoin*, connect if closer than the threshold
  - Quadratic complexity
    - Locality sensitive hashing

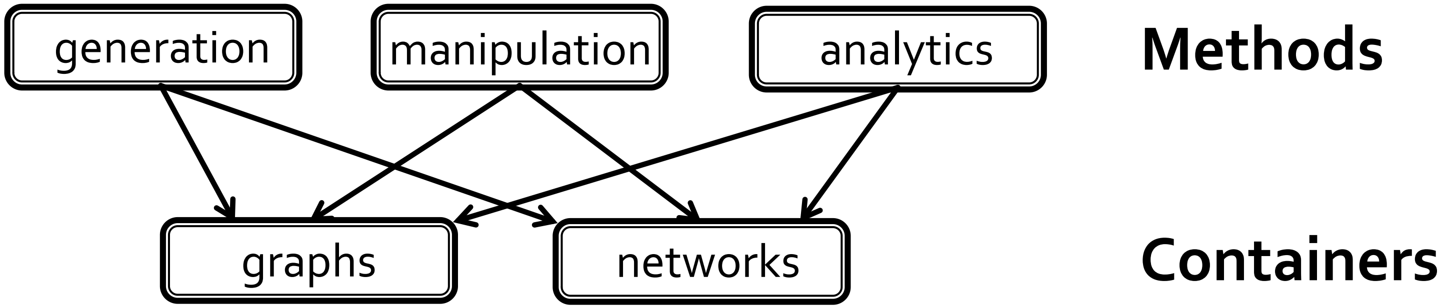
Edge Filters in Ringo

- **Use case: In a Web log, connect pages in a temporal order as clicked by the users**
  - Connect a node with its successors
    - Events selected per user, ordered by timestamps
    - *NextK*, connect K successors

Edge Filters in Ringo

- **Use case: In a Web log, measure the activity level of different user groups**
  - Edge creation
    - Partition users to groups
    - Identify interactions within each group
    - Compute a score for each group based on interactions
  - Treat groups as super-nodes in a graph

Graph Containers & Methods



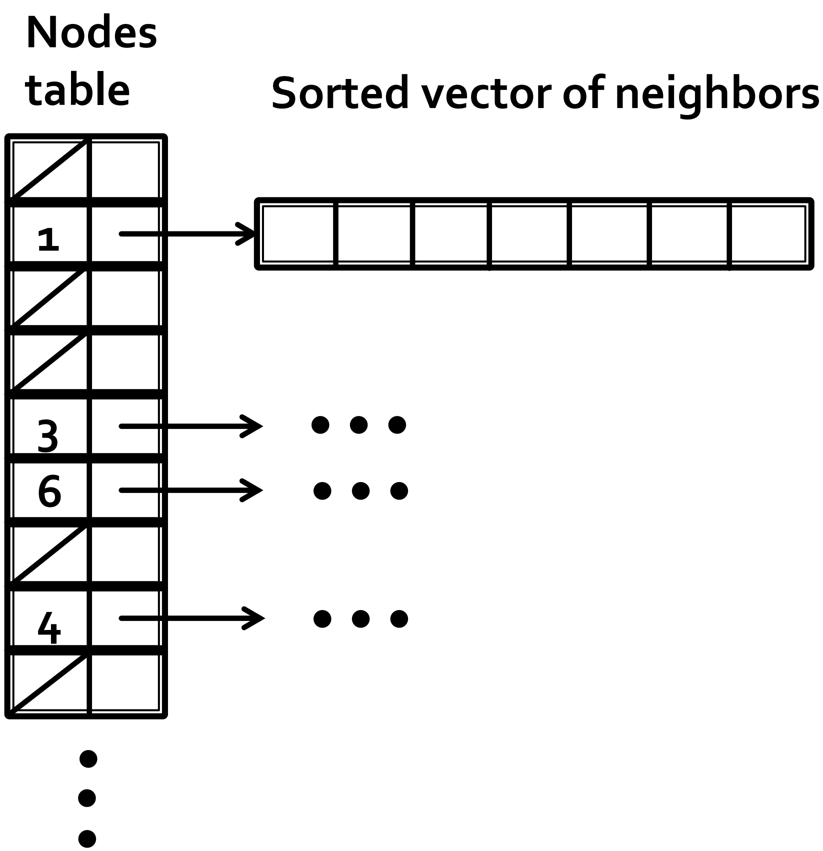
- **Several graph containers are supported**
- **Over 200 graph algorithms (provided via SNAP)**

Graph Representation

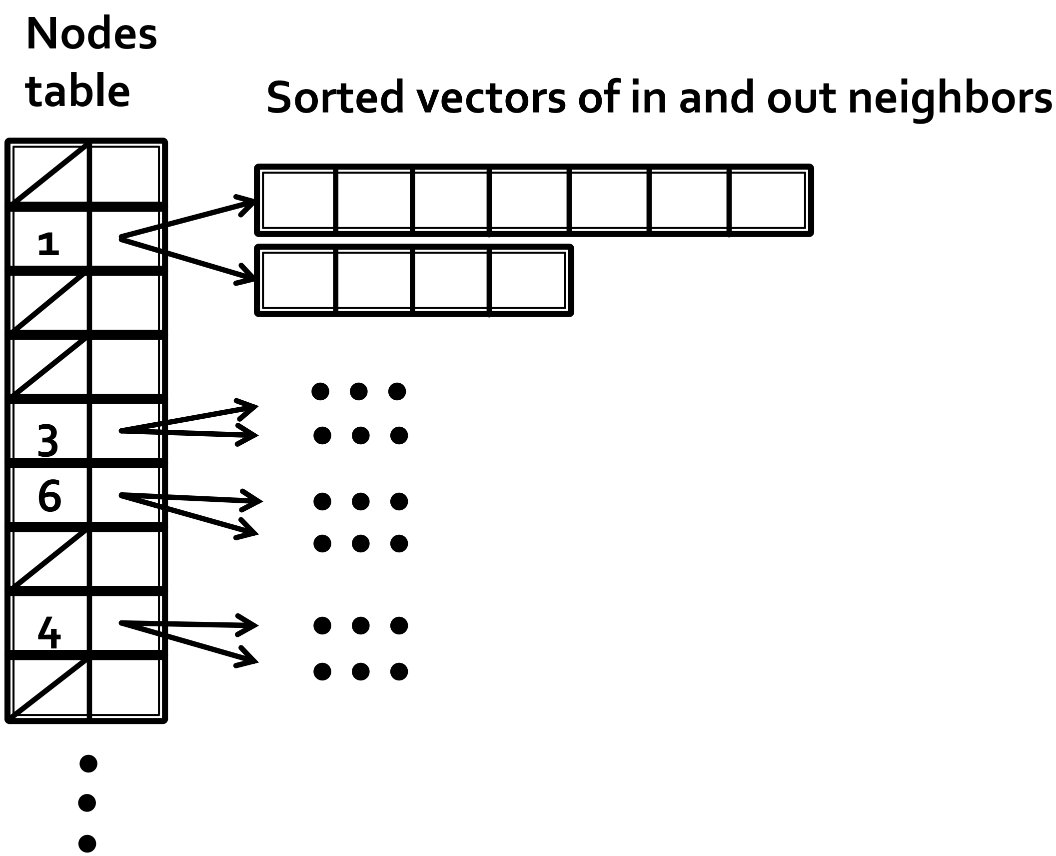
Requirements:

- Fast processing
  - Efficient traversal of nodes and edges
- Dynamic structure
  - Quickly add/remove nodes and edges
    - Create subgraphs, dynamic graphs, …
- **How to achieve good balance?**

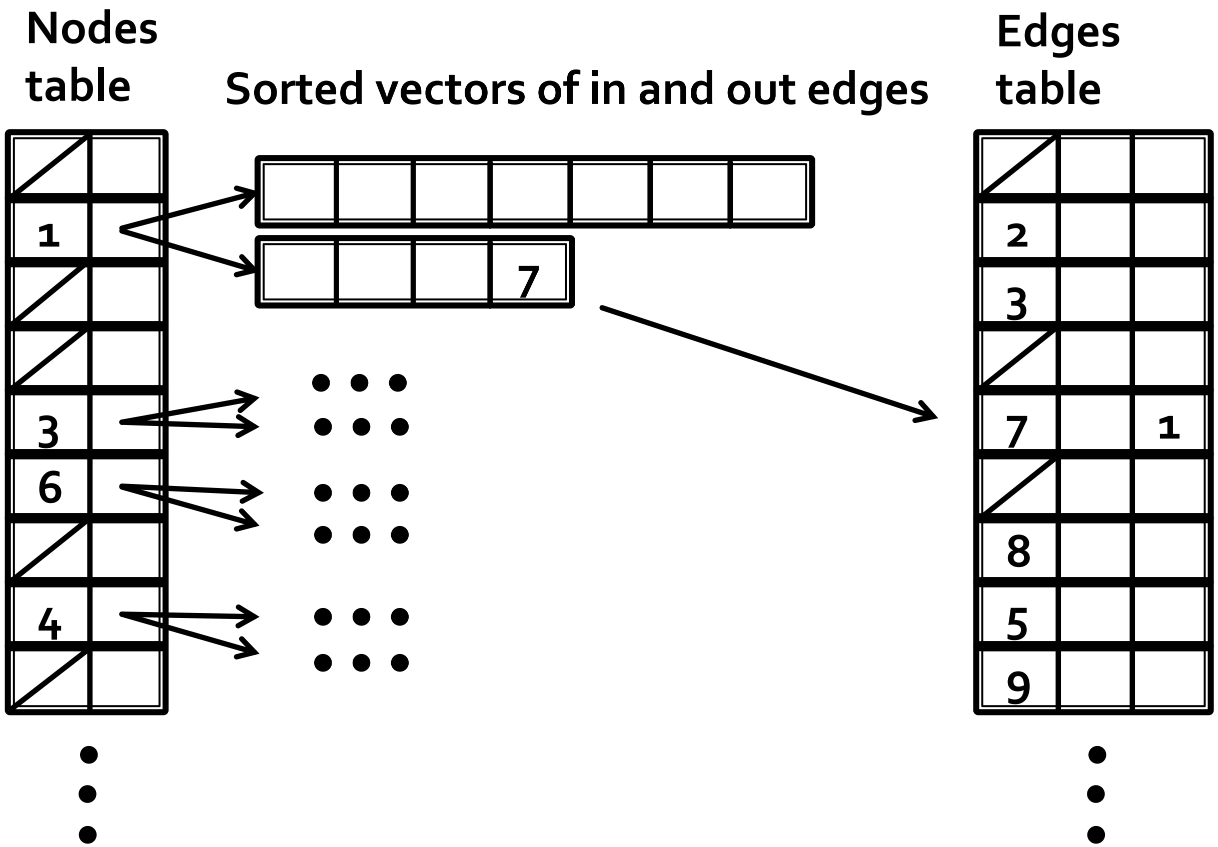
Undirected Graph in Ringo



Directed Graph in Ringo



Multigraph in Ringo

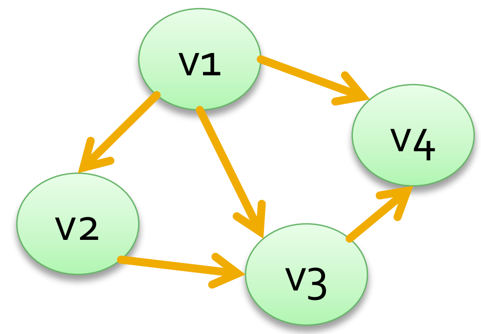


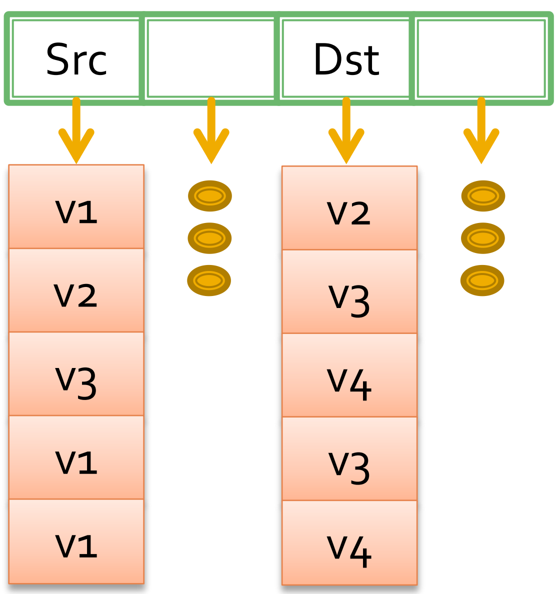
Tables in Ringo

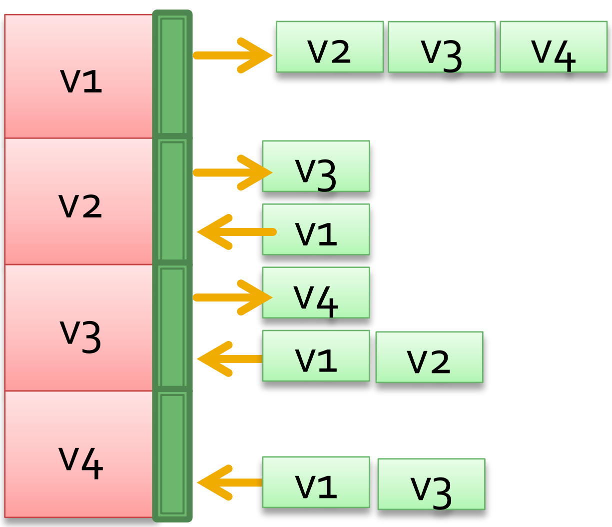
- **Column based store**
  - Persistent row identifier
  - **Lots of tricks:** Implicit select, String pools
- **Standard table operations**
  - Select, join, project, group, aggregate
- **Graph construction operations**
  - Explicit pairs
  - Distance based edges, *SimJoin*
  - Time based edges, *NextK*

Convert Tables to Graphs

Src	…	Dst	…
v1	…	v2	…
v2	…	v3	…
v3	…	v4	…
v1	…	v3	…
v1	…	v4	…







Convert Tables to Graphs

- Make two copies of pairs of (*Src,Dst*) columns (**Paralle**
- Sort first copy by *Src*, second copy by *Dst* (**Parallel**)
- Merge column *Src* from the first copy and *Dst* from the second copy (**S**equential)
- Count the number of unique values in the merged column to get the number of nodes in the graph (**S**)
- Allocate the node hash table
- For each node, build vectors of neighbors (**Parallel**)

Ringo Implementation

- **High-level front end**
  - Python module
  - Based on Snap.py, uses SWIG for C++ interface
- **High-performance graph engine**
  - C++ based on SNAP
- **Multi-core support**
  - OpenMP to parallelize loops
  - Fast, concurrent hash table, vector operations

Experiments: Datasets

Dataset	LiveJournal	Twitter2010
Nodes	4.8M	42M
Edges	69M	1.5B
Text Size	1.1GB	26.2GB
Graph Size	0.7GB	13.2GB
Table Size	1.1GB	23.5GB

Benchmarks, One Computer

Algorithm Graph	PageRank LiveJournal	PageRank Twitter2010	Triangles LiveJournal	Triangles Twitter2010
Giraph	45.6s	439.3s	N/A	N/A
GraphX	56.0s	-	67.6s	-
GraphChi	54.0s	595.3s	66.5s	-
PowerGraph	27.5s	251.7s	5.4s	706.8s
Ringo	2.6s	72.0s	13.7s	284.1s

Hardware: 4x Intel CPU, 64 cores, 1TB RAM, $35K

Benchmarks, Published

System	Hosts	CPUs host	Host Configuration	Time
GraphChi	1	4	8x core AMD, 64GB RAM	158s
TurboGraph	1	1	6x core Intel, 12GB RAM	30s
Spark	50	2		97s
GraphX	16	1	8X core Intel, 68GB RAM	15s
PowerGraph	64	2	8x hyper Intel, 23GB RAM	3.6s
Ringo	1	4	20x hyper Intel, 1TB RAM	6.0s

Twitter2010, one iteration of PageRank

Tables and Graphs

Dataset	LiveJournal	Twitter2010
Table to graph	8.5s 13.0 MEdges/s	81.0s 18.0 MEdges/s
Graph to table	1.5s 46.0 MEdges/s	29.2s 50.4 MEdges/s

Hardware: 4x Intel CPU, 80 cores, 1TB RAM, $35K

Table Operations

Dataset	LiveJournal	Twitter2010
Select 10K	<0.2s 405.9 MRows/s	1.6s 935.3 MRows/s
Select All-10K	<0.1s 575.0 MRows/s	1.6s 917.7 MRows/s
Join 10K	0.6s 109.5 MRows/s	4.2s 348.8 MRows/s
Join All-10K	3.1s 44.5 MRows/s	29.7s 98.8 MRows/s

Sequential Algorithms

Algorithm	Runtime
3-core	31.0s
Single source shortest path	7.4s
Strongly connected component	18.0s

LiveJournal, 1 core

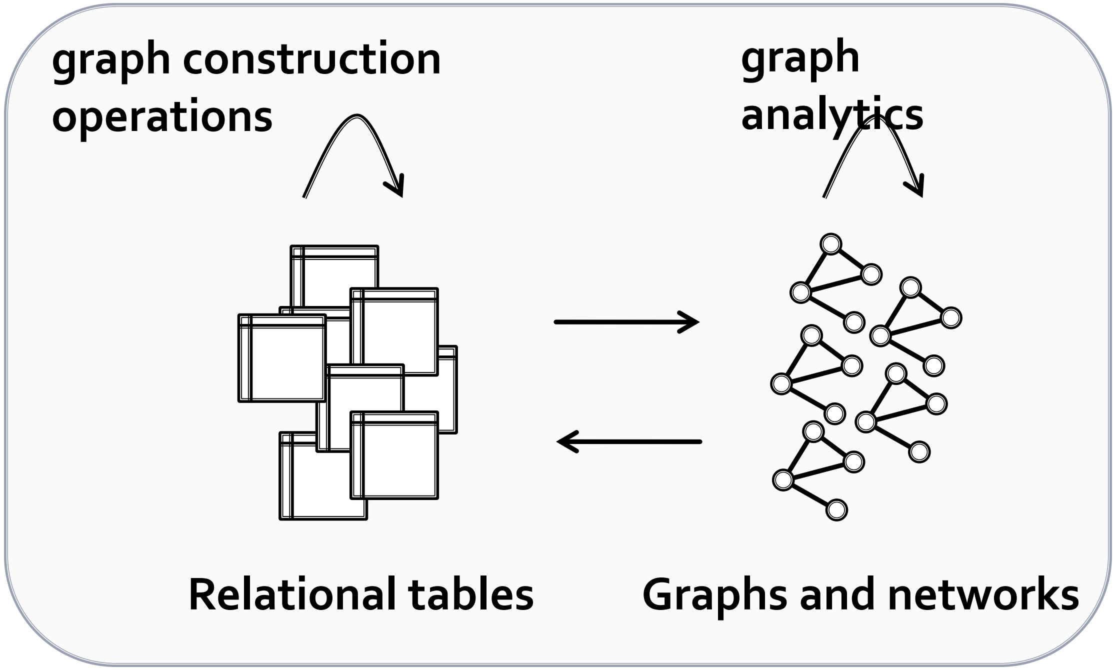
Load and Save Performance

Dataset	LiveJournal	Twitter2010
Load graph	5.2s	76.6s
Save graph	3.5s	69.0s
Load table	6.0s	114.3s
Save table	4.3s	100.2s

Conclusion

- **Big-memory machines are here:**
  - 1TB RAM, 100 Cores ≈ a small cluster
  - No overheads of distributed systems
  - Easy to program
- **Most "useful" datasets fit in memory**
- **Big-memory machines present a viable solution for analysis of all-but-the-largest networks**

Conclusion: Ringo



Ringo: Network science & exploration

- In-memory graph analytics
- Processing of tables and graphs
- Fast and scalable

Bottom line…

Get your own 1TB RAM server!

And download RINGO/SNAP
http://snap.stanford.edu/snap

☺

THANKS!

@jure

http://snap.stanford.edu