Spectral clustering for billion-node graphs

Da Zheng, Disa Mhembere, Youngser Park,
Joshua Vogelstein, Carey E. Priebe, Randal Burns

Johns Hopkins University

We build a framework that scales spectral clustering to massive graphs using SSDs in a single machine.

Spectral clustering

- Three steps
  - Get the largest connected component (optional)
  - Spectral embedding
  - K-means

Challenges

- Many real-world graphs are massive but sparse.
- Seemingly random vertex connection.
- Power-law distribution in vertex degree.

Step 1: get the largest connected component

- Two steps (FlashGraph): *fg.get.lcc*
  - Compute connected components.
  - Extract the subgraph ~ the same size as the original graph.

Step 2: spectral embedding

- eigen(A)
- eigen(D-A)
- eigen($I-D^{-1/2}AD^{-1/2}$)

FlashEigen

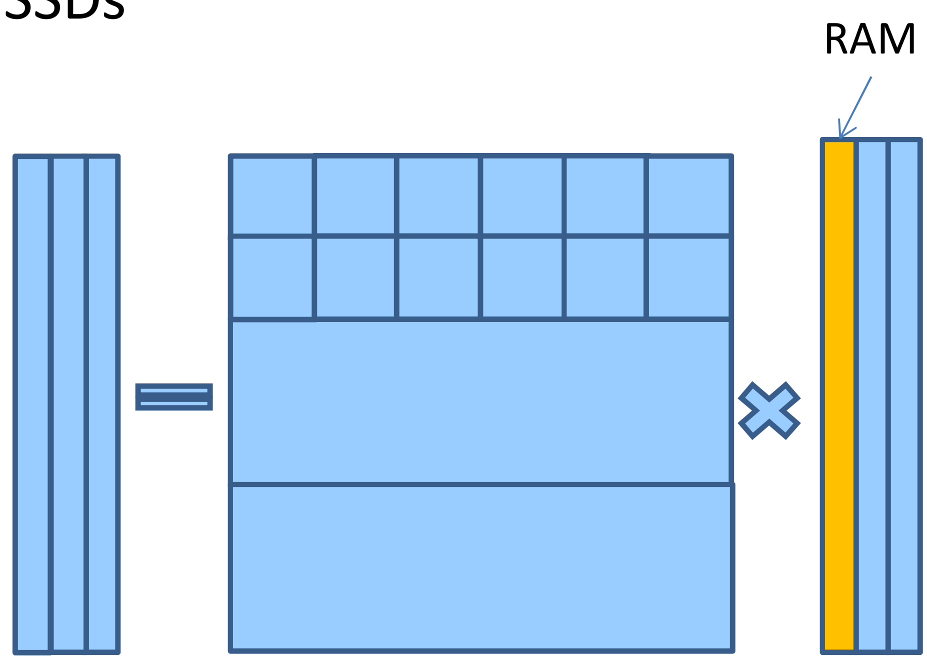
- An SSD-based eigensolver integrated with Trilinos Anasazi framework.
  - *fm.eigen(fun, options)*: *fun* defines sparse matrix multiplication.

Why Anasazi?

- Trilinos Anasazi framework
  - Customize sparse matrix multiplication.
    - The sparse matrix is large.
  - Customize the operations on the vector subspace.
    - Vector subspace is large (~ the size of the sparse matrix).

Sparse matrix multiplication

- Key operation of computing eigenvalues.
- Semi-external memory:
  - The input dense matrix in memory
  - The sparse matrix on SSDs



Operations on the vector subspace

- Reimplement Anasazi::MultiVec to operate on the vector subspace on SSDs.

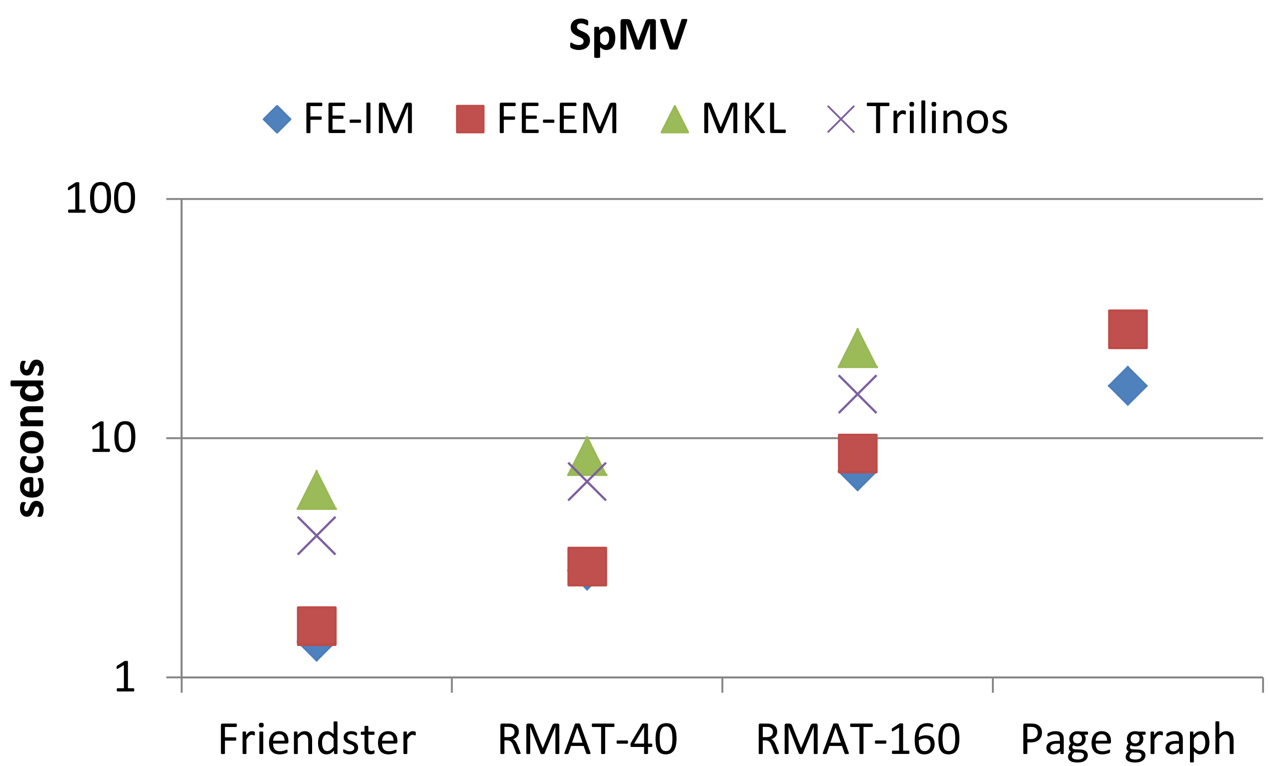
MvTimesMatAddMv	MvAddMv	MvScale
MvTransMv	MvDot	MvNorm
SetBlock	MvRandom	MvInit

Graphs for performance evaluation

Graphs	# Vertices	# Edges
Friendster	65M	1.7B
RMAT-40	100M	3.7B
RMAT-160	100M	14B
Page graph	3.4B	129B

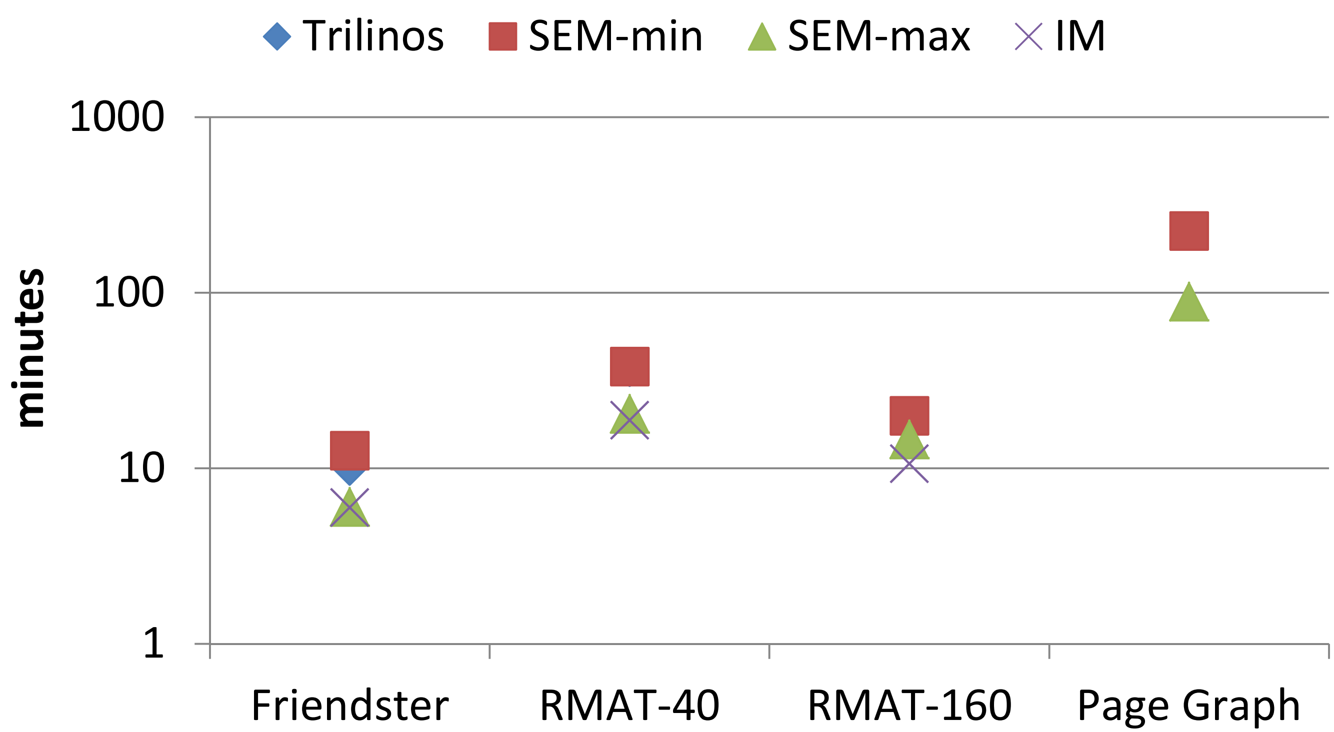
SpMV in FlashEigen vs. others

- FM-SpMV significantly outperforms MKL and Trilinos.
- FM-SpMV can scale to the Page graph.

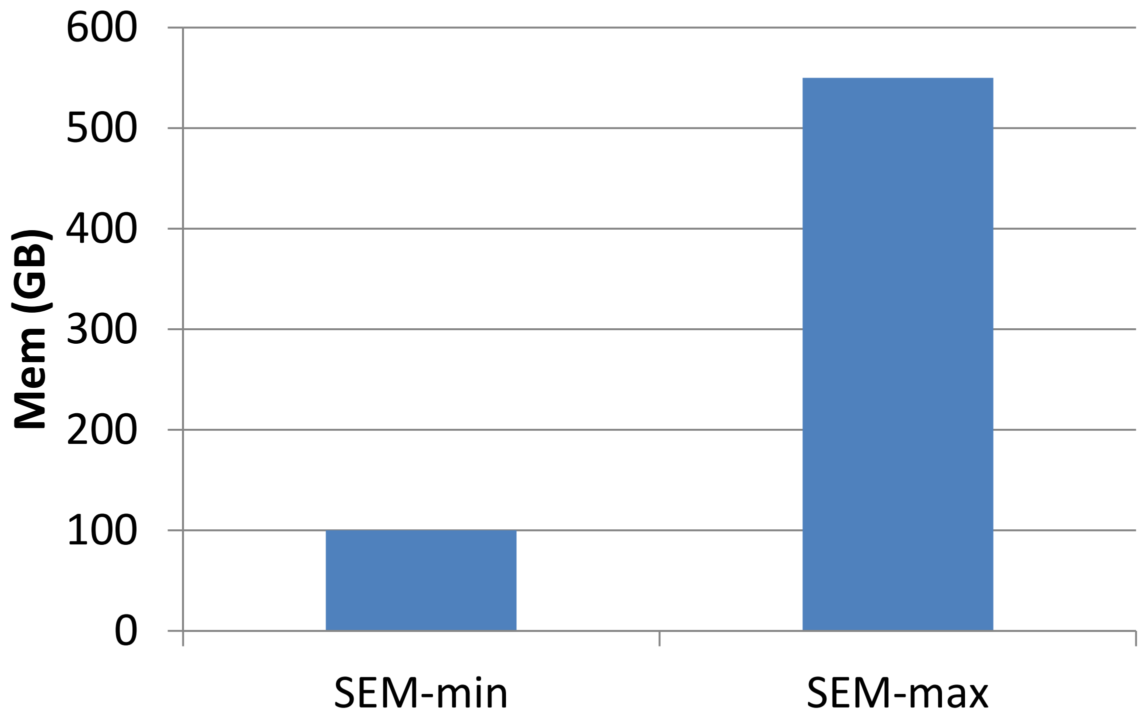


FlashEigen vs. Trilinos

- FlashEigen outperforms Trilinos.
- FlashEigen computes eigenvalues of the Page graph.



Memory consumption of computing 8 eigenvalues on the Page graph



Step 3: k-means

- Importance of accelerating k-means:
  - Run k-means for different k and d (embedded dimension size) to search for right k and d.
  - K-means can be the bottleneck in spectral clustering.

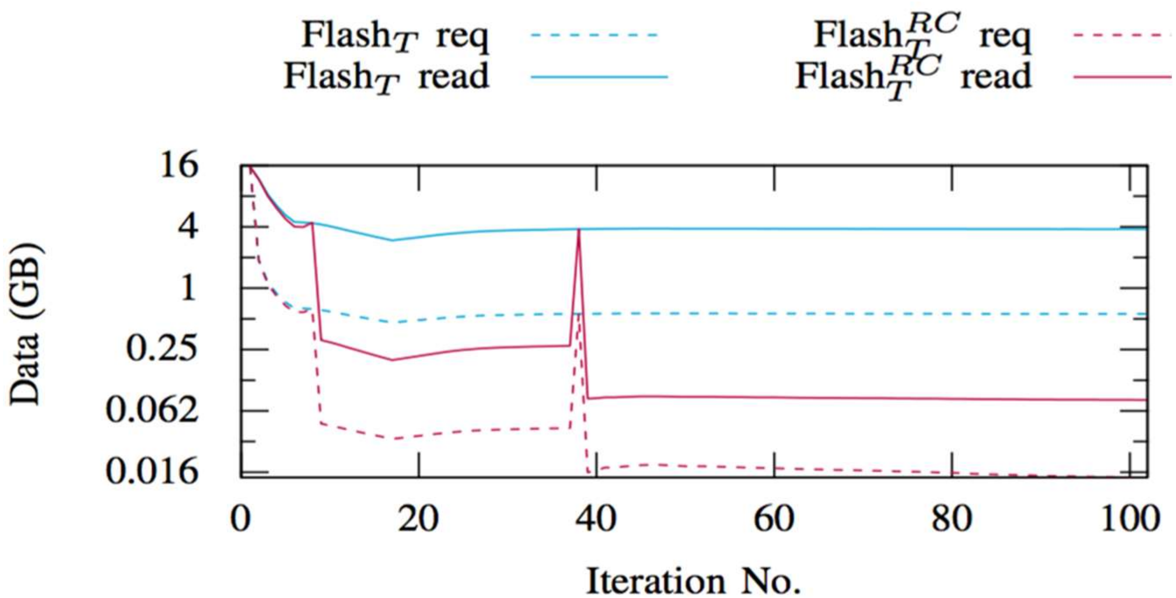
Accelerate k-means with triangle inequality

- Avoid unnecessary computation with the two lemmas (x is a point and b and c are centers)[1]:
  - If $d(b,c) \geq 2d(x,b)$, then $d(x,c) \geq d(x,b)$.
  - $d(x,c) \geq \max\{0,\ d(x,b) - d(b,c)\}$.
- This optimization skips computation on most of data points (idle).

1. ELKAN, C. Using the triangle inequality to accelerate k-means. In ICML (2003).

Implementation of k-means with triangle inequality

- Keep the entire datasets on SSDs.
- Keep most recently active data points in memory -- row caching (RC).



Perforamnce of k-means

- Often an order of magnitude faster than MLlib, H20 and Dato!

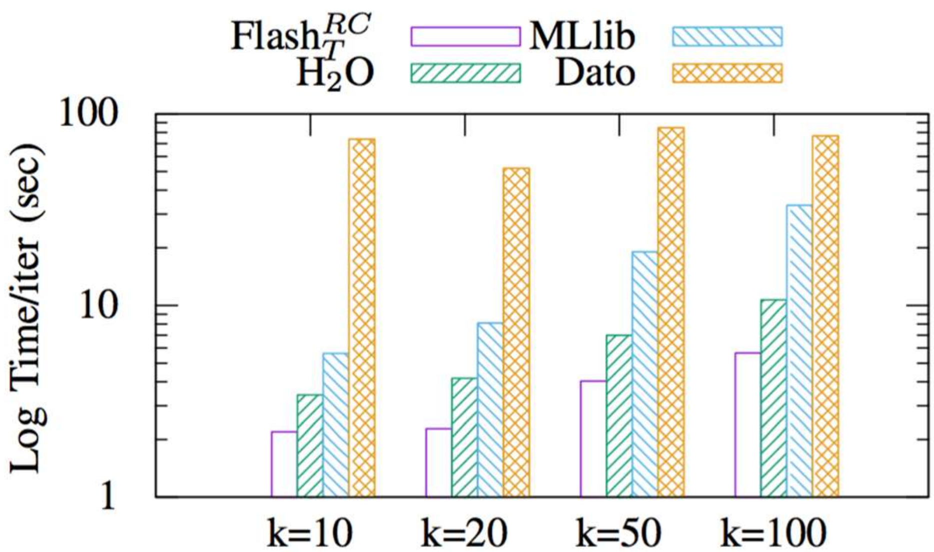


Fig. 9: The Friendster graph top-32 eigenvector dataset.

FlashX

- FlashX contains all of the functions for spectral clustering.
  - FlashX: http://flashx.io/

Thank you

- Da Zheng: dzheng5@jhu.edu