Computing the Closeness Centrality of Evolving Networks on Clusters

Ahmet Erdem Sarıyüce, Erik Saule, Kamer Kaya, Ümit V. Çatalyürek
The Ohio State University. University of North Carolina at Charlotte. Sabancı University.

Introduction

Centrality metrics answer questions such as:

- Who controls the flow in a network?
- Who is more important? Who has more influence?
- Whose contribution is significant for connections?

They have various applications:

- Covert network (e.g., terrorist identification).
- Contingency analysis (e.g., weakness/robustness of networks).
- Viral marketing. Traffic analysis. Store locations.

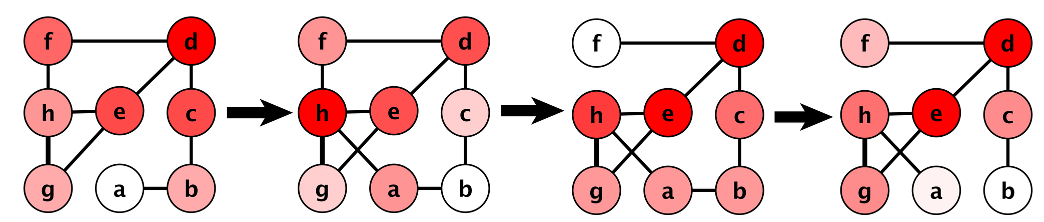
How to maintain an accurate analysis upon graph evolutions?

Closeness Centrality

Let $G = (V, E)$ be an unweighted graph with the vertex set V and edge set E.
The farness of a vertex is $far[u] = \sum_v d(u, v)$.
The closeness of a vertex is $cc[u] = \frac{1}{far[u]}$.
$d(u, v)$ is the length of the shortest path $u \rightsquigarrow v$.

The best known algorithm computes the shortest path graph rooted in each vertex of the graph. The complexity is $O(E)$ per source, $O(VE)$ in total, which makes its computationally expensive.

Variation of Closeness Centrality



The redness of a vertex represents its Closeness.

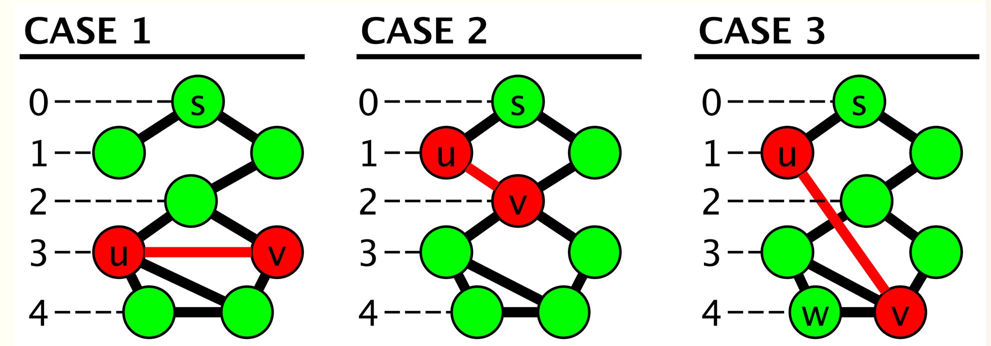
Conclusion

Structural properties of Closeness Centrality lead to an efficient incremental algorithm. The algorithm scales on clusters and enables real-time Centrality analysis.

References

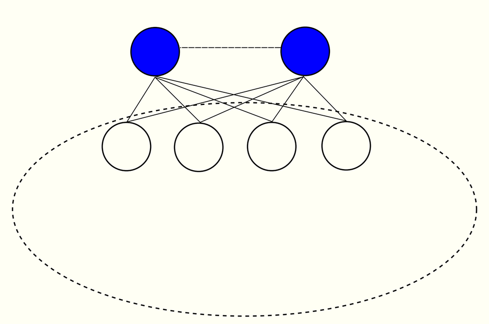
[1] A. E. Sarıyüce, E. Saule, K. Kaya, and Ü. V. Çatalyürek. Hardware/software vectorization for closeness centrality on multi-/many-core architectures. In *Workshop on Multithreaded Architectures and Applications (MTAAP)*, 2014.
[2] A. E. Sarıyüce, E. Saule, K. Kaya, and Ü. V. Çatalyürek. Streamer: a distributed framework for incremental closeness centrality computation. In *IEEE Cluster*, 2013.
[3] A. E. Sarıyüce, E. Saule, K. Kaya, and Ü. V. Çatalyürek. Incremental algorithms for closeness centrality. In *IEEE BigData*, 2013.
[4] A. E. Sarıyüce, E. Saule, K. Kaya, and Ü. V. Çatalyürek. Shattering and compressing networks for betweenness centrality. In *SIAM International Conference on Data Mining, SDM*, 2013.

Level Filtering



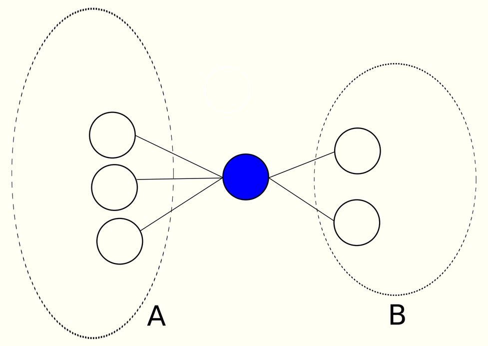
Case 1: No change.
Case 2: Only the structure changes.
Case 3: Some distances change.

Identical Vertices



Vertices identical to each other have the same farness.

Articulation point



When an edge changes in A, all the vertices in B are impacted in the same way as the articulation point.

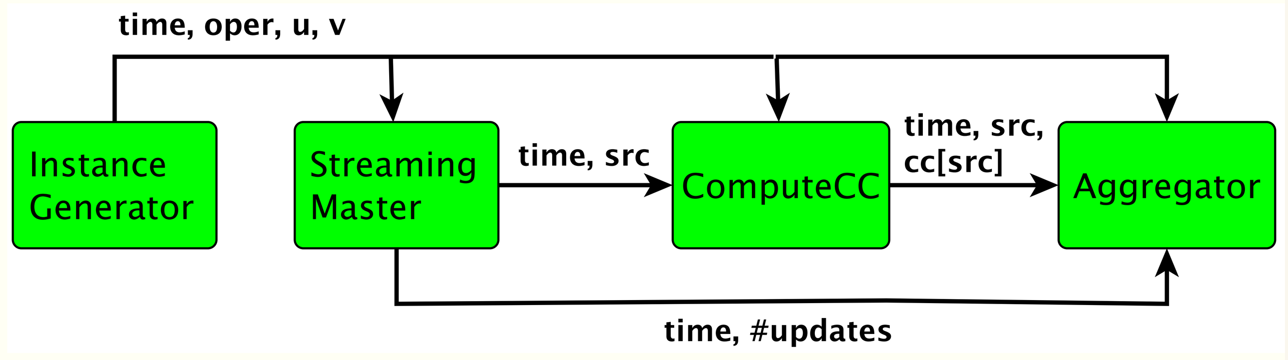
Incremental Algorithm

- Upon insertion of (u, v)
- Ignore temporarily identical vertices
- Consider (u, v)'s biconnected component
- Compute BFS from u and v (before edge insertion)
- Flag all $s \neq u, v$, s.t. $|d(u, s) - d(v, s)| > 1$,
- Add (u, v) to the graph
- Compute $cc[s]$ for all flagged s
- Change $cc[s]$ in other biconnected components
- Copy $cc[s]$ for identical vertices

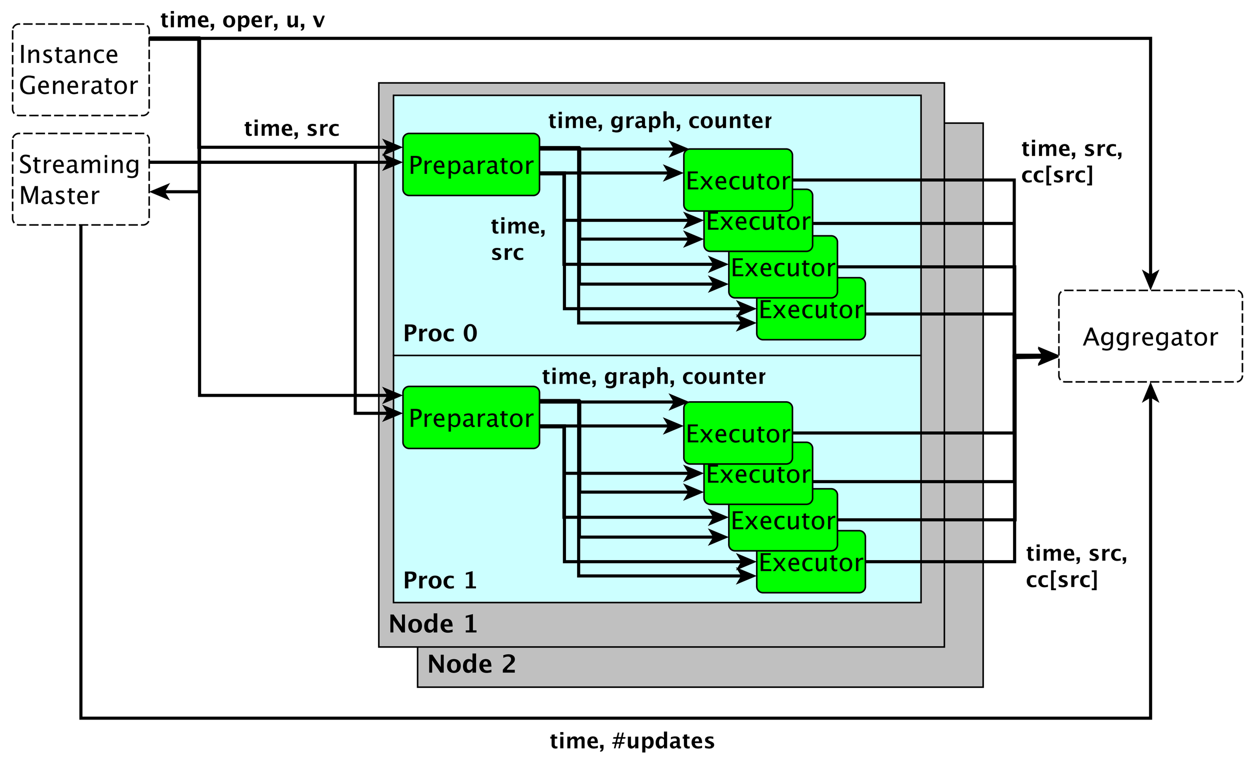
Acknowledgment

This work was partially supported by the Defense Threat Reduction Agency grant HDTRA1-14-C-0007.

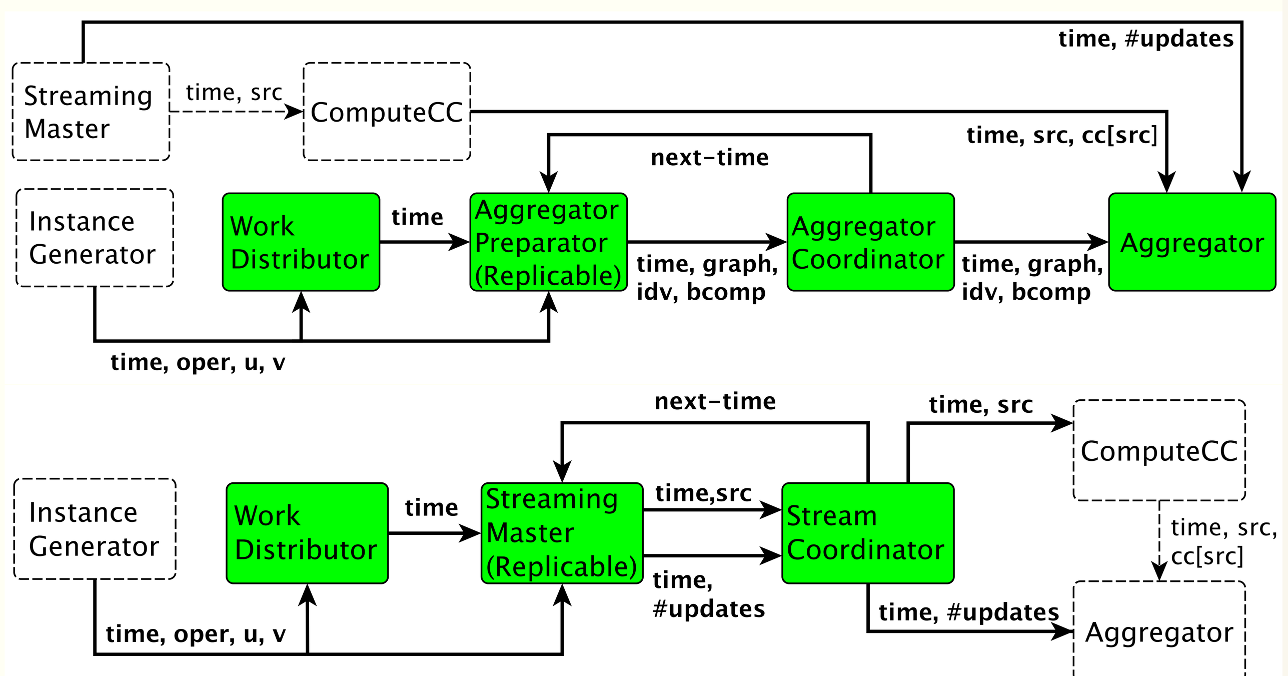
Streamer Layout



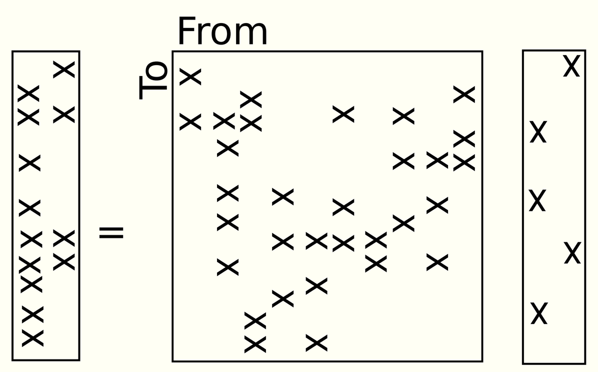
Leveraging the NUMA Hierarchy



Making Aggregator and StreamingMaster Replicable



Vectorized Bottom-Up Traversal



Multiple traversals can be executed simultaneously to leverage the large vector registers of modern processors.

Test Machine

64 nodes computational cluster:

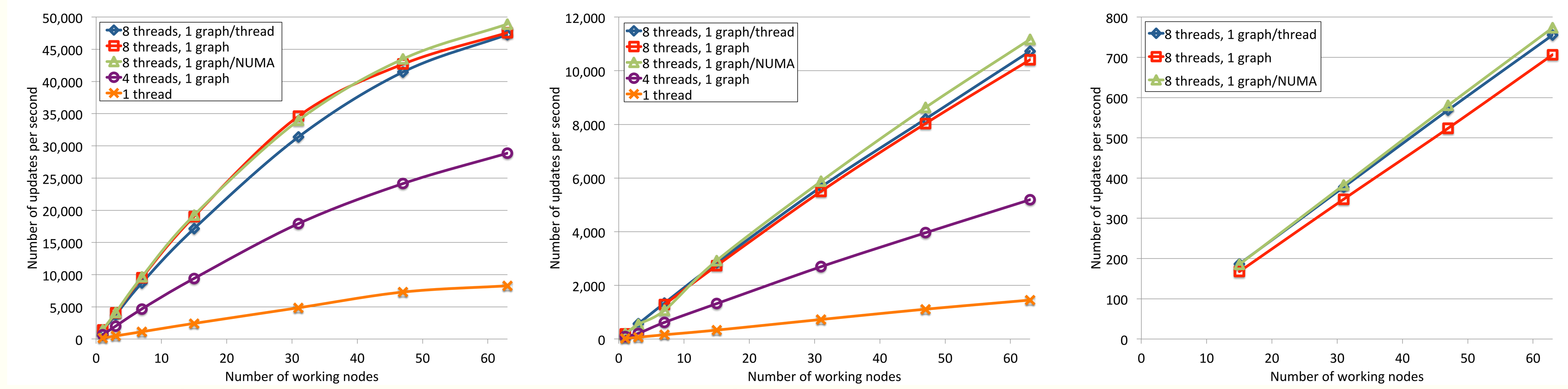
- 2 Intel Xeon E5520 @2.27Ghz
  - 4 cores (8 cores per node)
  - 2-way SMT (16 threads per node)
  - 8MB L3 cache (shared per proc)
- 48 GB of main memory
- 20 Gbps InfiniBand

Test Instances

Incrementally inserting 50 initially removed edges on different graphs. It leads to different number of updates in centrality.

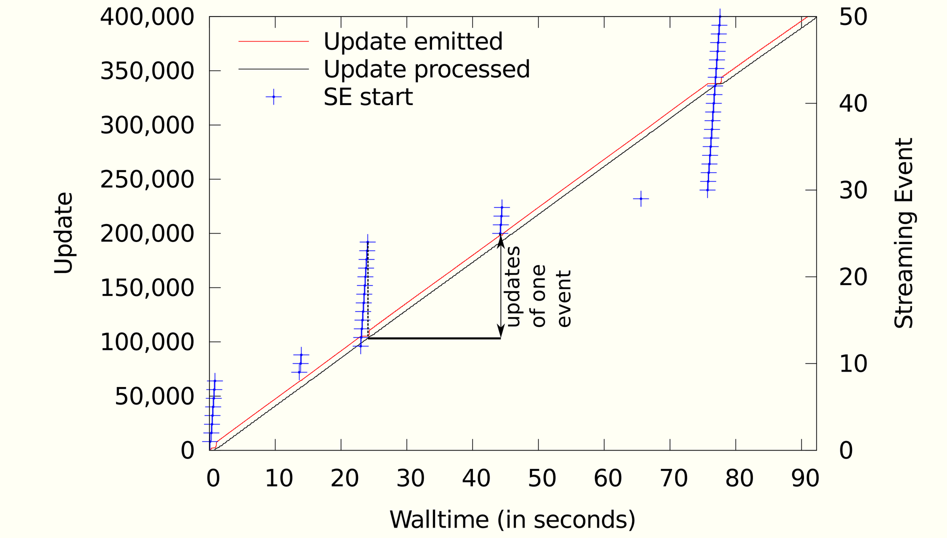
Name	$\|V\|$	$\|E\|$	# updates
`web-NotreDame`	325,729	1,090,008	399,420
`web-Google`	916,428	4,321,958	2,527,088
`soc-pokec`	1,632,804	30,622,464	4,924,759

Experimental Results

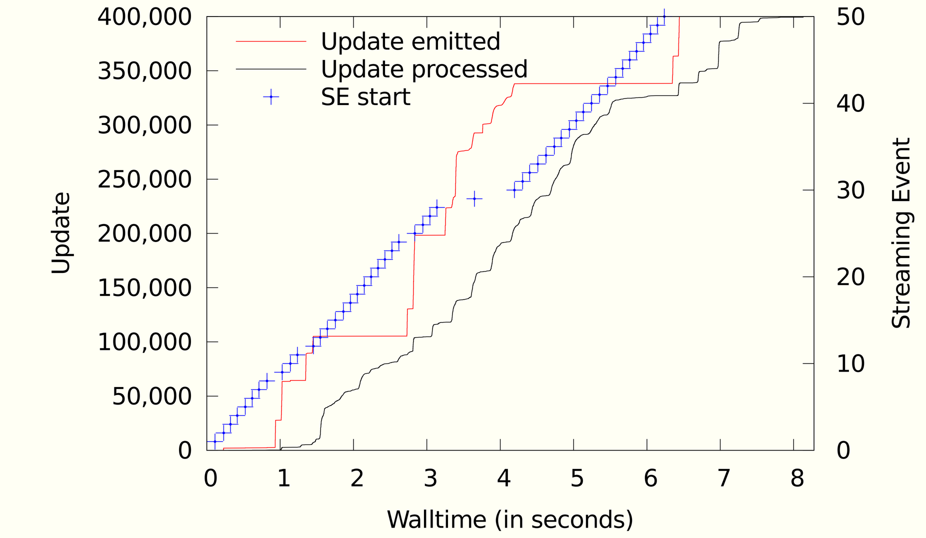


`web-NotreDame` `web-Google` `soc-pokec`

Log Analysis of `web-NotreDame`

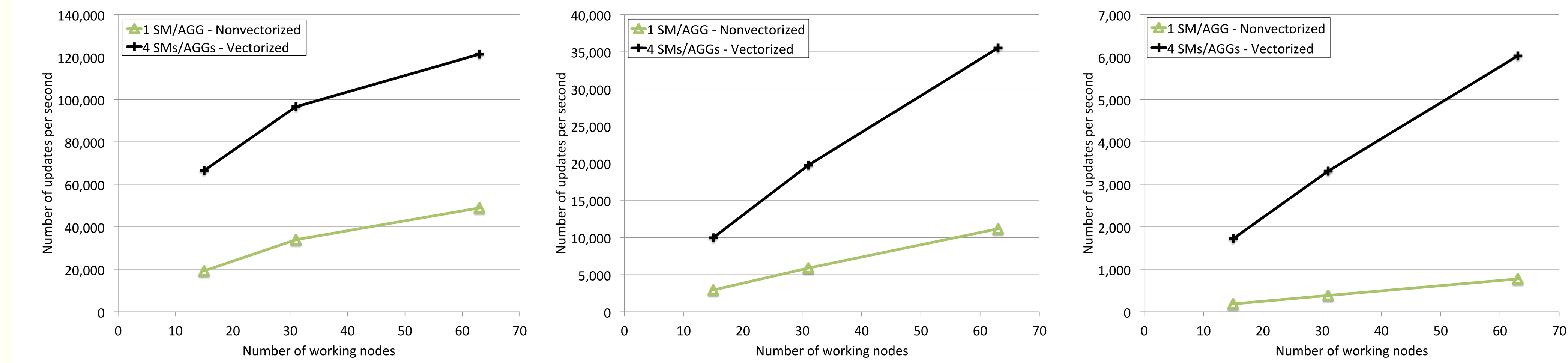


3 worker nodes



63 worker nodes

Using Replicable StreamingMaster and Aggregator (and Vectorization)



`web-NotreDame` `web-Google` `soc-pokec`