Challenges of Parallelizing Graph Algorithms for Network Science

Boleslaw K. Szymanski[a]
Konstantin Kuzmin[a,] Mingming Chen[b], Chris Gaiteri[c] Xiaoayn Lu[a]

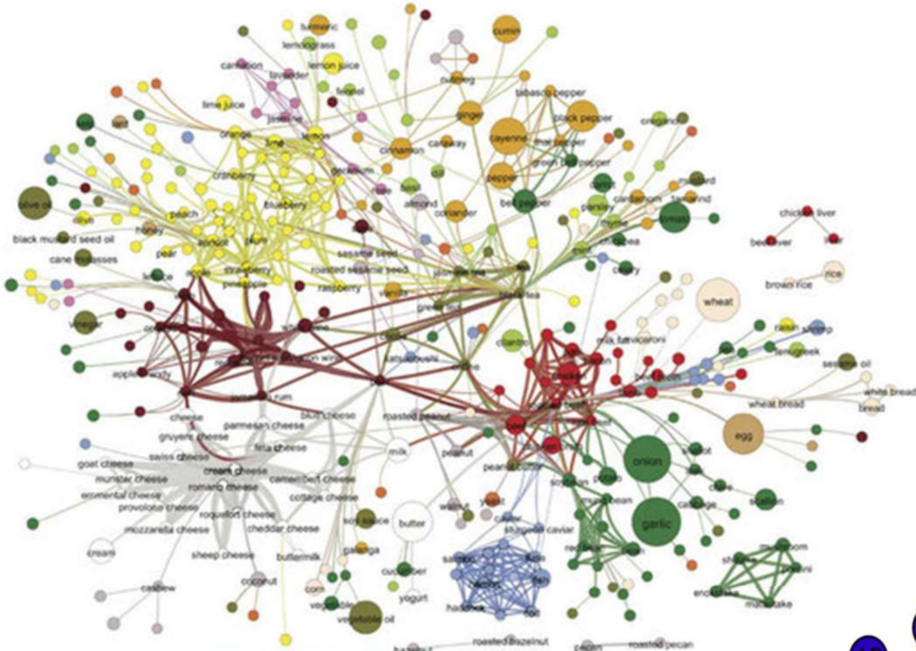
[a]NeST Center & SCNARC, RPI, Troy, NY
[b]Google, Inc., Mountain View, CA
[c]Rush Medical College, Rush University, Chicago, IL

PPAM, Lublin, Poland, September 12, 2017

Outline

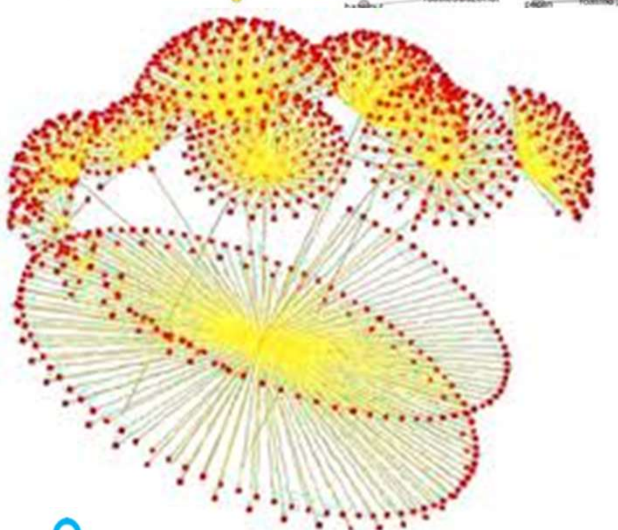
- **Motivation for this research**
- Top-down bottom-up based community detection algorithm SpeakEasy
- Parallelization of SpeakEasy
- Algorithm for Prediction of Viral News Cascades and its parallelization
- Conclusions

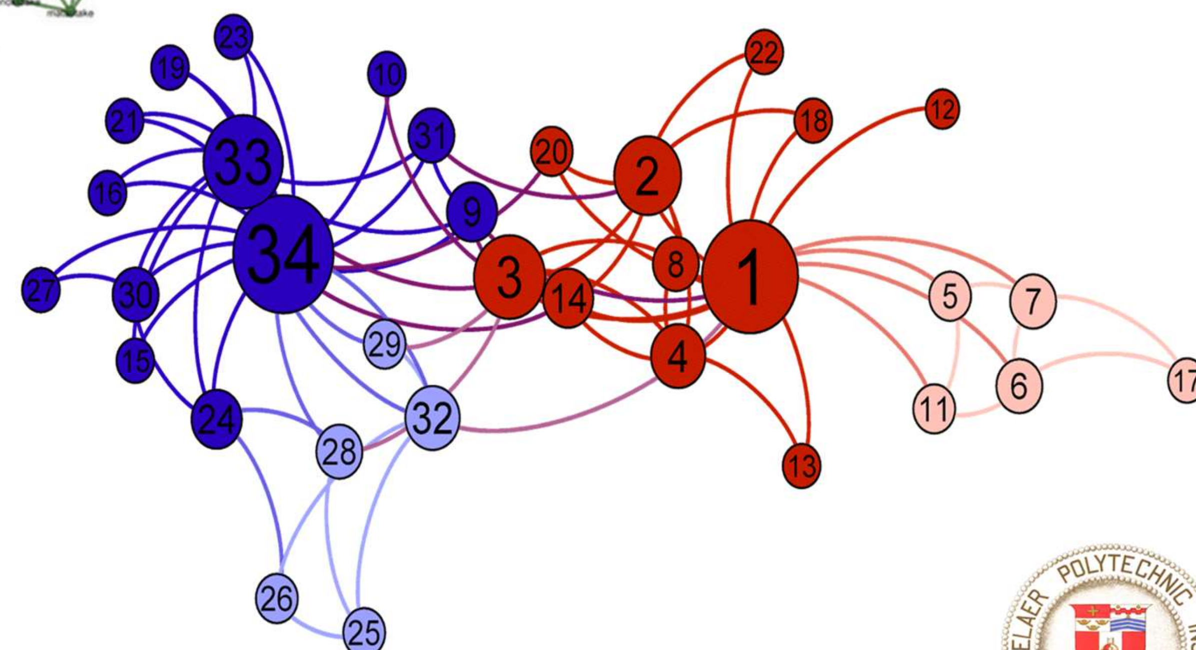
Discovering Communities in Social & Bio-networks



Clustering implies modularity
Functional modularity imposes natural boundary lines between communities.

Discovering community structure uncovers functionality





Bio (left) and social (right) networks are driven by functionality

Using Community Detection for Studying Alzheimer's Disease

Why take a new approach to so well studied subject?

Because we barely understand it at all

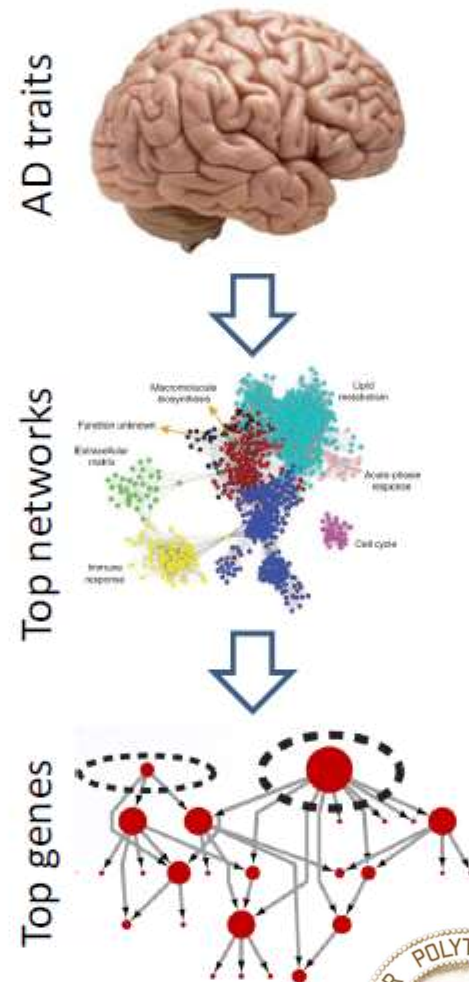
- 400+ clinical trials
- 200+ compounds
- One with slight reduction of symptoms (Memantine) and no preventative drugs
- Genetic linkage studies indicate multiple molecular systems involved in pathology
- For most cases small contributions from many molecules
- What is perceived as AD is clouded by other age-related pathologies

Overview of datasets and approach

Primary Datasets

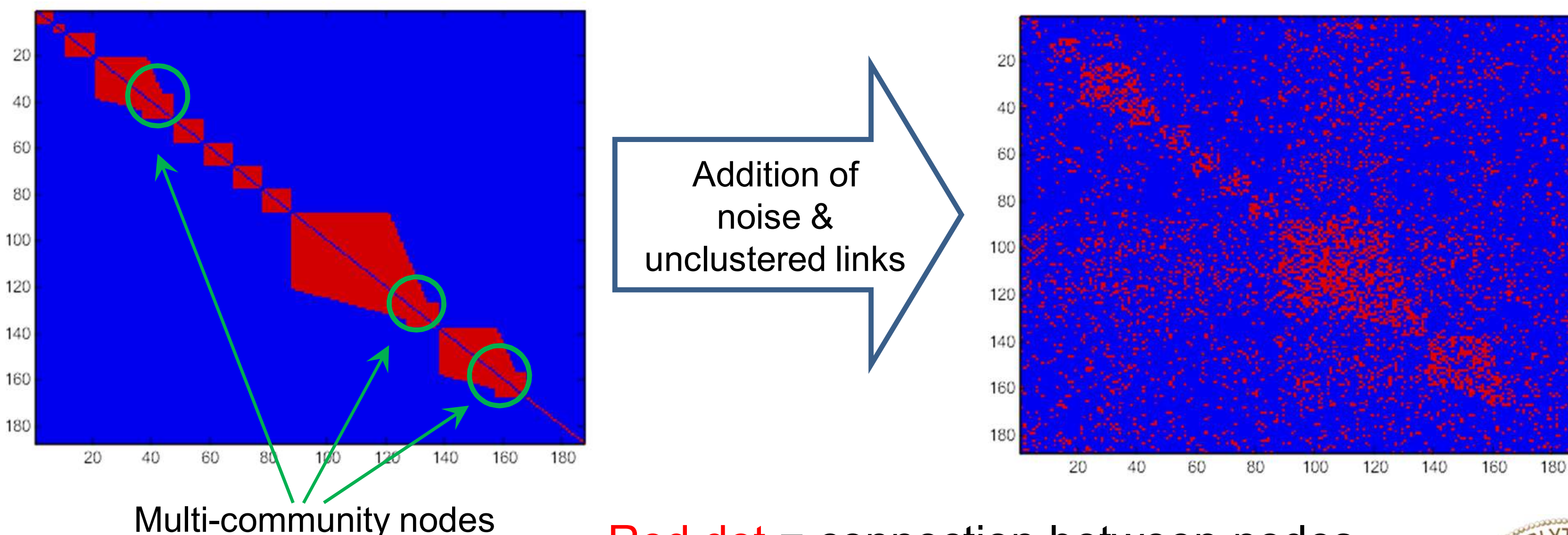
		Harvard Brain Tissue Resource	ROSMAP
Pre Frontal Cortex		SNPs, gene expression, phenotypes	SNPs, gene expression (RNAseq), more phenotypes
	AD (*n*)	284	~300
	Control (*n*)	153	~200
Visual Cortex	AD (*n*)	168	
	Control (*n*)	116	
Cerebellum	AD (*n*)	220	
	Control (*n*)	122	

Approach



Motivation for Specialized Algorithms

- Biological and social networks have high level of noise and therefore have incorrect or missing links
- Biological or social functions are accomplished by communities of interacting molecules/cells or people
- Membership in these communities may overlap when humans or biological components are involved in multiple functions



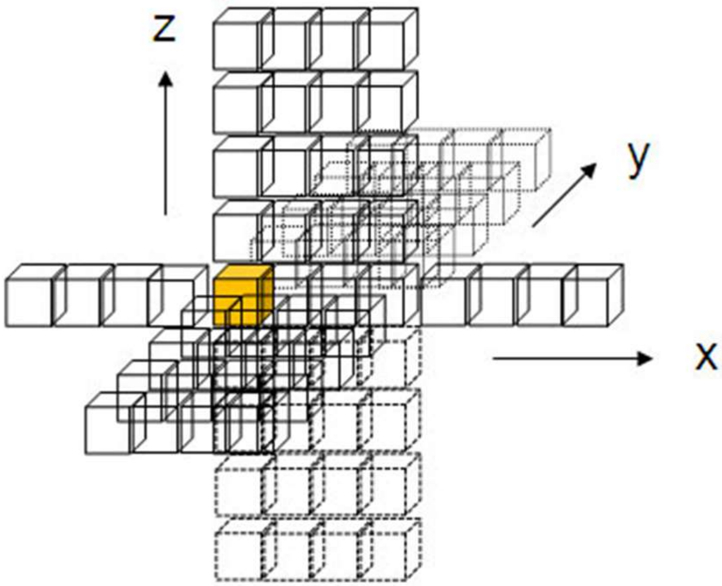
Red dot = connection between nodes

Computational Patterns in Network Science

Physics Interactions

Regular grid embedded in space, all interaction are local.

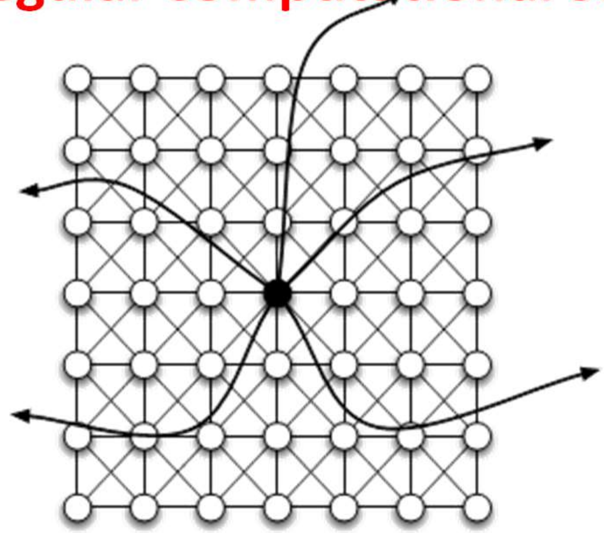
Regular computational stencils.



Network interactions

Links are independent of nodes' locations, interactions are global.

Irregular computational stencils.



E. David, J. Kleinberg, *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. CUP (2010).

Outline

- Motivation for this research
- **Top-down bottom-up based community detection algorithm SpeakEasy**
- Parallelization of SpeakEasy
- Algorithm for Prediction of Viral News Cascades and its parallelization
- Conclusions

SpeakEasy Algorithm

- **Novelty**: Identifies communities using top-down and bottom-up approaches simultaneously. Specifically, nodes join communities based on their local connections and global information about the network structure.

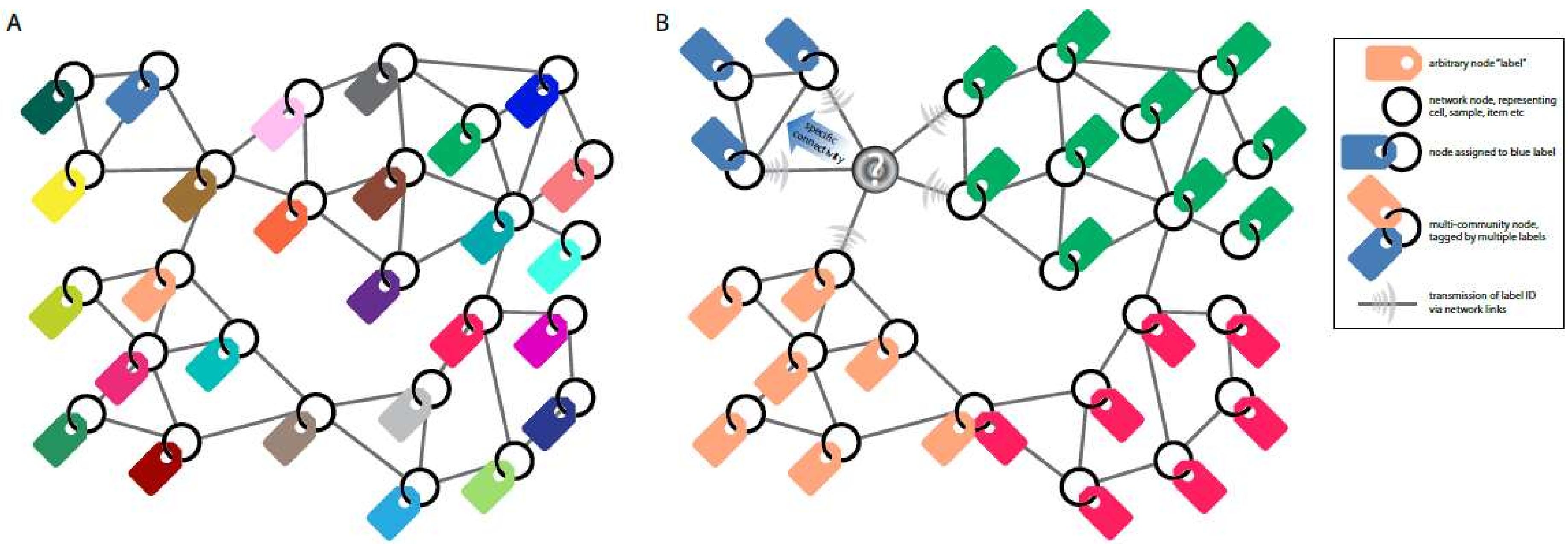
- **Label propagation algorithm**: each node updates its status to the label found among nodes connected to it which has the greatest specificity, i.e., the actual number of times this label is present in neighboring nodes minus its expected number based on its global frequency.

- **Consensus clustering**: the partition with the highest average adjusted Rand Index among all other partitions is selected as the representative partition to get robust community structure.

- **Overlapping communities**: overlapping communities can be obtained with co-occurrence matrix. Multi-community nodes are selected as nodes which co-occur with more than one of the final clusters with greater than a user-selected threshold.

Visual Example of SpeakEasy Clustering

- Labels are represented by color tags
- Multi-community nodes are tagged with multiple colors



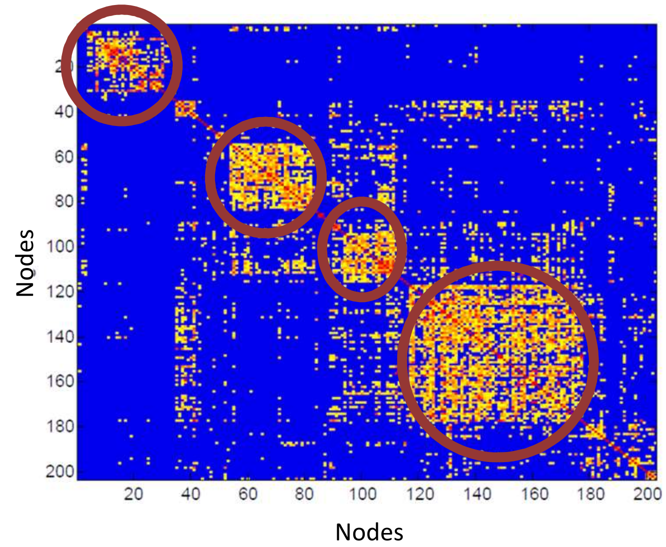
A. Each node is assigned with random ue label (before clustering)

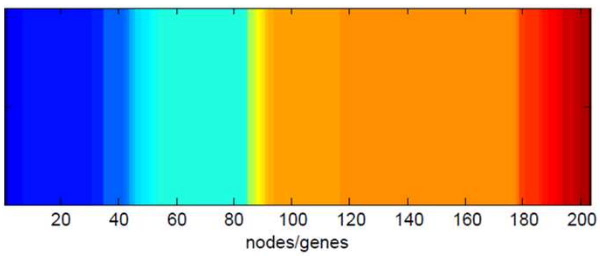
B. Nodes with the same labels belong to the same community (after clustering)

Clustering Workflow

- Algorithm identifies communities though evolution of common labels.
- After a certain number of iterations of label propagation or if none of the nodes updates its labels in the given iteration, nodes with the same label will be clustered into the same community.
- However, because the clustering is fast and parameter-free, running the algorithm multiple times, we get an assessment of the robustness of the clusters and the identity of multi-community nodes.

Correlation matrix after clustering





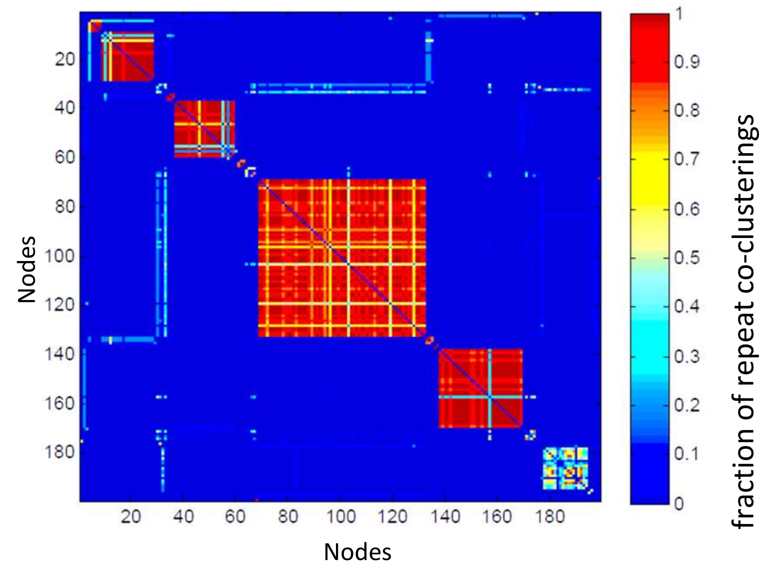
Color-coded community ID

C. Gaiteri, M. Chen, B.K. Szymanski, et al. *Scientific Reports* **5**:16361 (2015)

Identifying Multi-community Nodes

- ➢ Run SpeakEasy multiple times (e.g. 100x).
- ➢ For all pairs of nodes (*i*, *j*) the “co-occurrence” matrix records number of times they land in same cluster.
- ➢ This is useful for both identifying robust clusters and for finding nodes that link multiple communities together.

Co-occurrence matrix

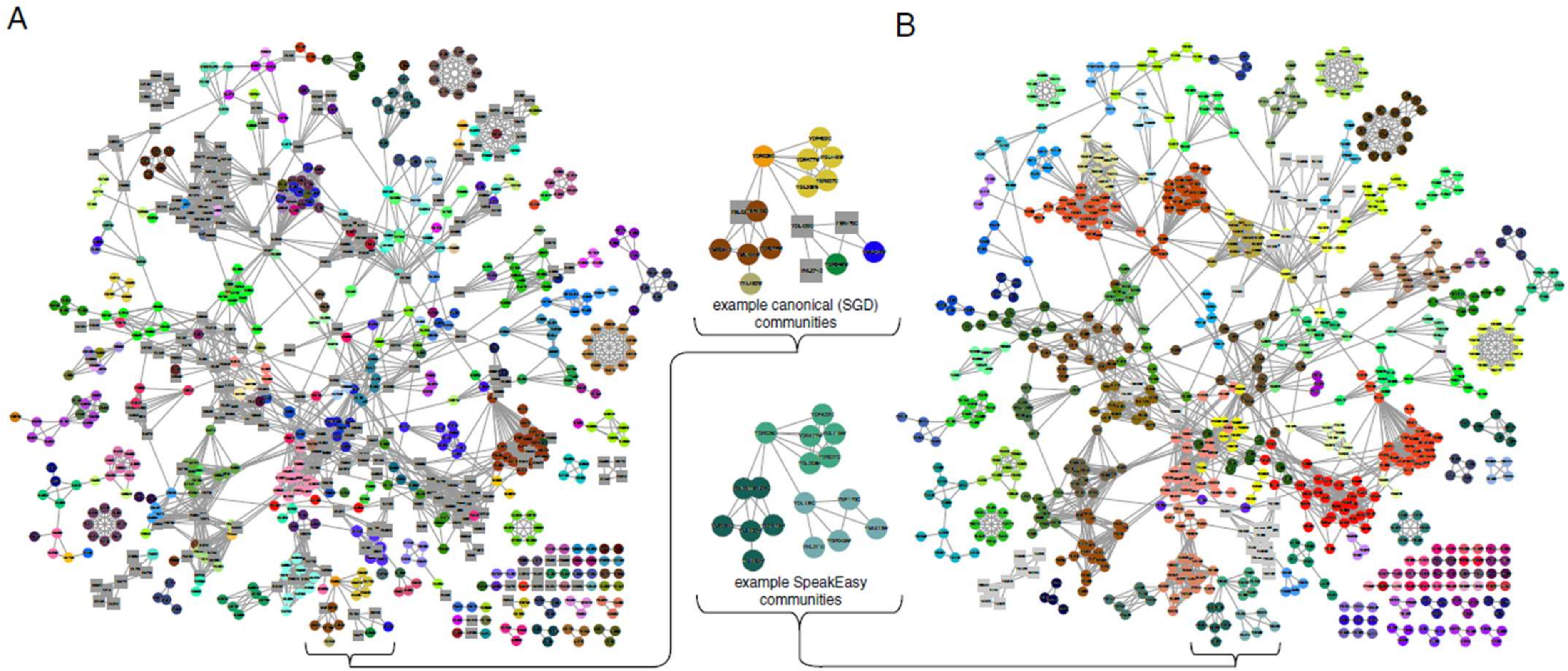


Clusters in this matrix show nodes that cluster across many initial conditions

Strong non-clustered/ off-diagonal elements show multi-community nodes

C. Gaiteri, M. Chen, B.K. Szymanski, et al. *Scientific Reports* ***5***:16361 (2015)

Application to Protein-protein Interaction Datasets



A. The high throughput interaction dataset from Gavin et al. has nodes colored according to protein complexes found in the Saccharomyces Genome Database (SGD).

B. The communities identified with SpeakEasy on the high throughput interaction dataset from Gavin et al.

Outline

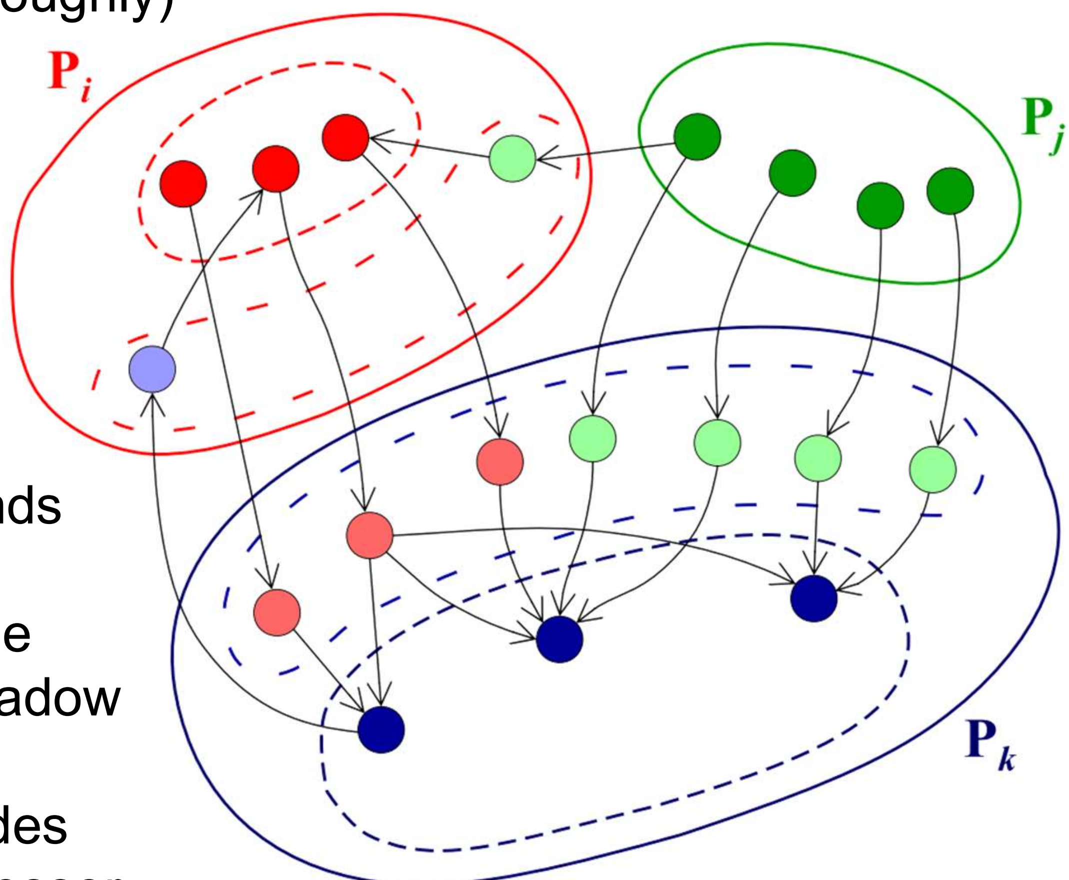
- Motivation for this research
- Top-down bottom-up based community detection algorithm SpeakEasy
- **Parallelization of SpeakEasy**
- Algorithm for Prediction of Viral News Cascades and its parallelization
- Conclusions

High-Level Approach to Parallelizing SpeakEasy

- Partition the data (nodes) between processors
- Perform label propagation on each partition in parallel
- Synchronize at the end of each label propagation iteration
- Exchange the global label frequencies information among the processors
- Extract community data from label histories (also in parallel)

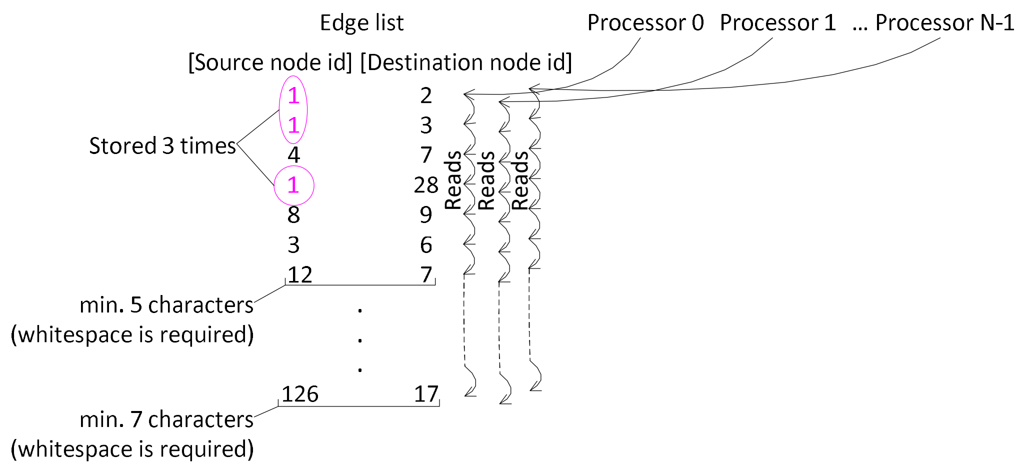
Partitioning nodes for parallel processing

- Each processor gets a (roughly) the same number of nodes
- Colors correspond to processors
- A lighter shade is used for external shadow nodes
- Fine dashed line surrounds internal nodes
- Wider spaced dashed line encloses the external shadow nodes
- Solid line denotes all nodes associated with the processor internal and external shadow nodes



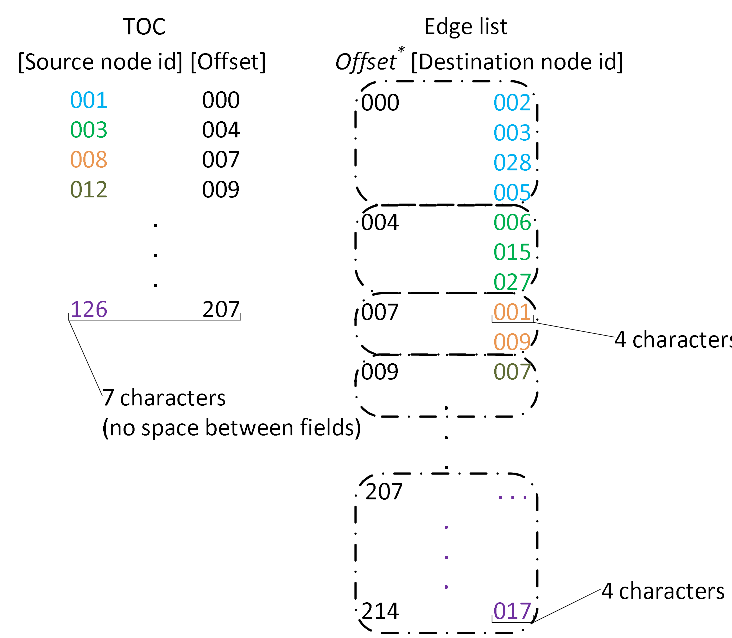
A Traditional Edge List File Is Not the Best Format for Parallel I/O

- An edge list file is a text file with a variable line length
  - cannot be efficiently partitioned for parallel I/O since file offsets for each processor are not known before reading the entire file
  - redundant due to storing multiple copies of the source node id
- Edges in the file are not guaranteed to be arranged in any particular order (e.g., not grouped by the source node id)
  - each processor needs to read the entire edge list file sequentially, filtering out the nodes which belong to other processors
  - I/O requests for the same data from different processors are not guaranteed to be grouped together resulting in poor scalability



Designing a Parallel Efficient Edge List Format

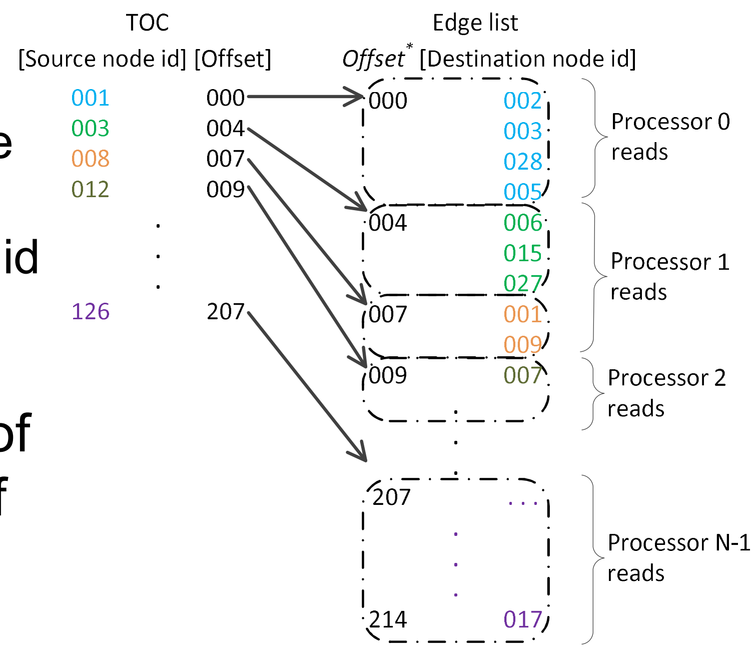
- Store edges in a format with a fixed field size. All values are either fixed size text (e.g., 10 characters per value) or fixed size binary integer (e.g., 4 bytes per value)
- Group all edges with the same source node id together
- Only store the destination node id in the edge list file
- Store the source node id (only once!) in a supplemental TOC file along with an offset to the first edge of this source node id in the edge list file
- The TOC file also has a fixed field size ([Source node id] [Edge list file offset])



* not a stored field, shows the value of the offset in the file

Efficient Parallel I/O and Scalability

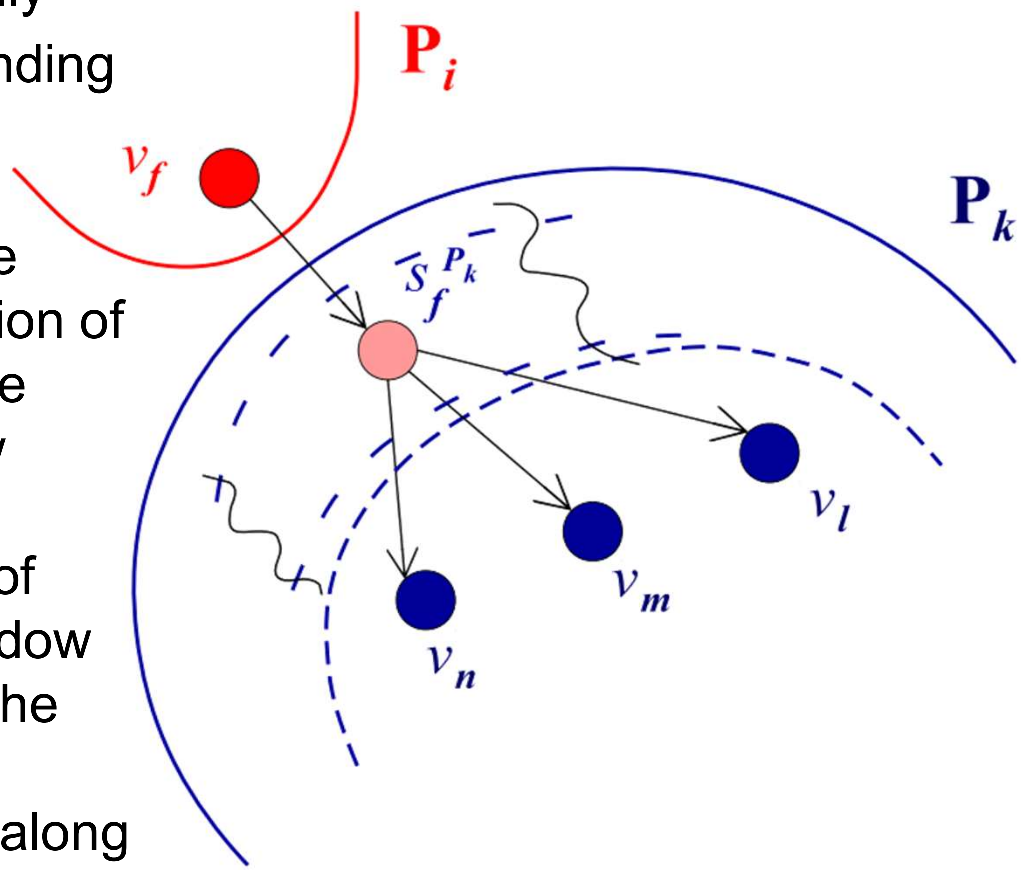
- Each processor:
  - computes its preliminary edge list file offsets based on the file size (field size is fixed!)
  - adjusts the offsets to the source node id boundaries by reading the TOC file
  - reads the edges
- Each processor reads only a section of the edge list file (an integer number of source node groups)
- Each edge from the edge list file is read only by one processor
- Each processor gets the same number of edges (up to the adjustment to the source node id group boundaries) which provides a balanced load for the O(m) label propagation algorithm



* not a stored field, shows the value of the offset in the file

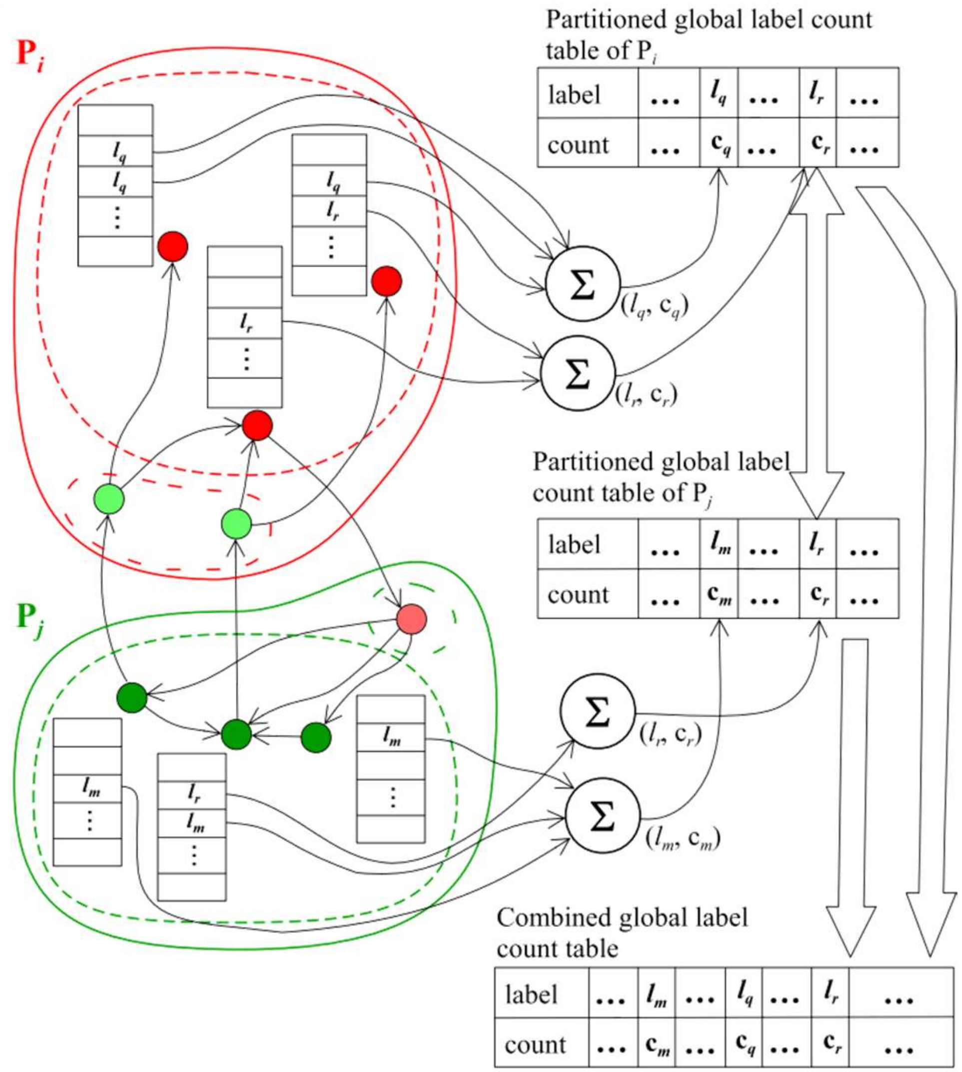
Propagation of Label Updates

- A shadow node ($s_f^{P_k}$) fully represents its corresponding original node (v_f) in the external partition
- Multiple connections are replaced by a combination of a single link between the original and the shadow nodes and the corresponding number of edges between the shadow node and the nodes in the related partition
- Label updates are sent along the virtual edge (v_f, $s_f^{P_k}$)



Computing Label Counts

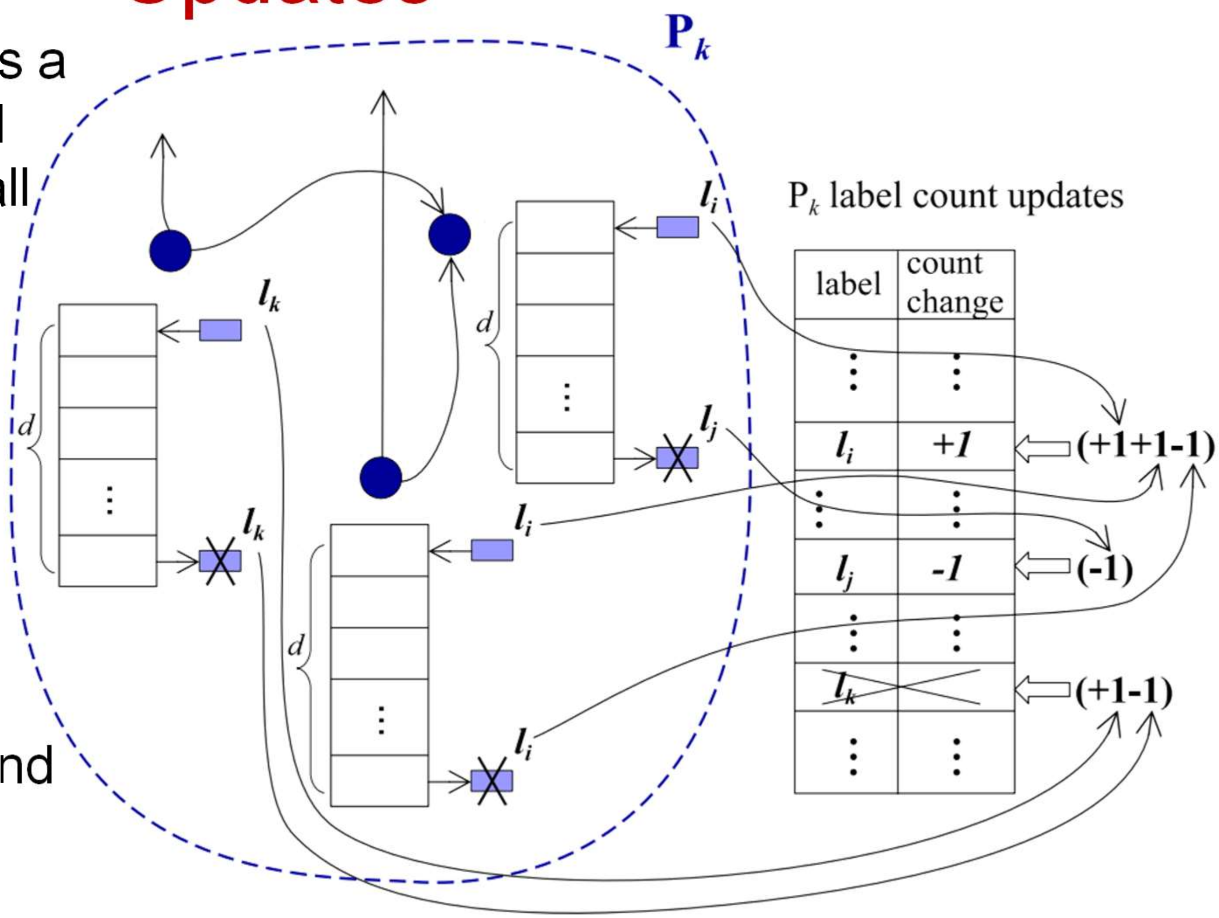
- Label history list for each node has a fixed configurable size
- Label counts are computed from individual label history lists
- The same label can occur more than once in the history of a node
- Label count values are stored in partitioned global label count tables (one table per processor)
- A combination of all individual partitioned count tables can be thought of as the global label count table



SCNARC

Collecting and Maintaining Label Updates

- Each processor has a list that tracks label count updates for all labels found in history lists of nodes which belong to internal and external partitions
- At each iteration a new label is added to the history list, and the oldest label is deleted
- If the accumulated change in label count across all nodes during a certain iteration is zero, there is no need to keep this update



Test Networks

- YouTube – a platform for sharing video with certain features (e.g., the ability to like certain videos or subscribe to certain channels) characteristic of social networks. Edges correspond to friendship relations which users can establish.
- LiveJournal – on-line blog. The network of users is connected to each other through a self-declared friendship relationship.
- Methylation – describes the distribution of methyl groups on DNA (which helps to control which genes are transcribed). Arranged linearly along DNA, so in the same chromosome all sites should be placed into the same initial partition, as they are more likely to show interactions.

Network	# of nodes	# of edges
com-Youtube	$\approx 1.13\times10^6$	$\approx 2.99\times10^6$
com-LiveJournal	$\approx 3.99\times10^6$	$\approx 34.68\times10^6$
methylation	5,000	25×10^6

SCNARC

RENSSELAER POLYTECHNIC INSTITUTE 1824

Testing Environment

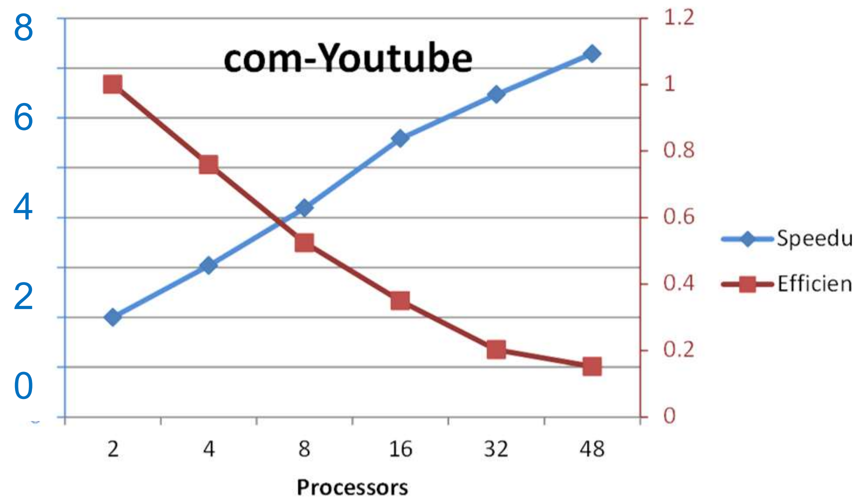
- High-performance shared-memory machine (off-the-shelf Silicon Mechanics[2] Rackform iServ R420.v4)
- 32 cores organized as four Intel[3] Xeon™ E5-4620v2 (2.6 GHz, 8-core, 20 MB Cache)
- Shared 1 TB of Random Access Memory (RAM) (32 x 32 GB DDR3-1600 ECC Registered 4R DIMMs) running at 1600 MT/s Max

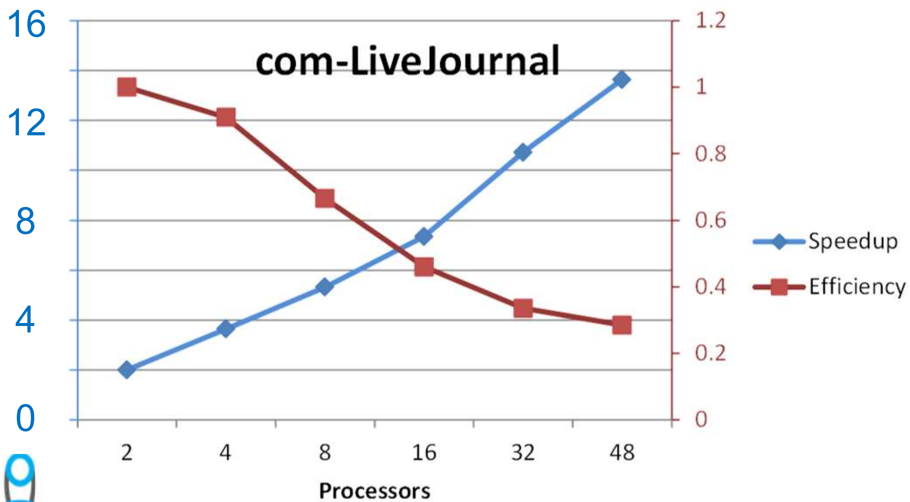
- Standard hyper-threaded Ubuntu[1] Linux operating system
- OpenMPI[4] high performance computing library

1 Ubuntu and Canonical are registered trademarks of Canonical Ltd.
2 Silicon Mechanics and the Silicon Mechanics logo are registered trademarks of Silicon Mechanics, Inc.
3 Intel, the Intel logo, the Intel Inside logo and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.
4 The Open MPI Project.

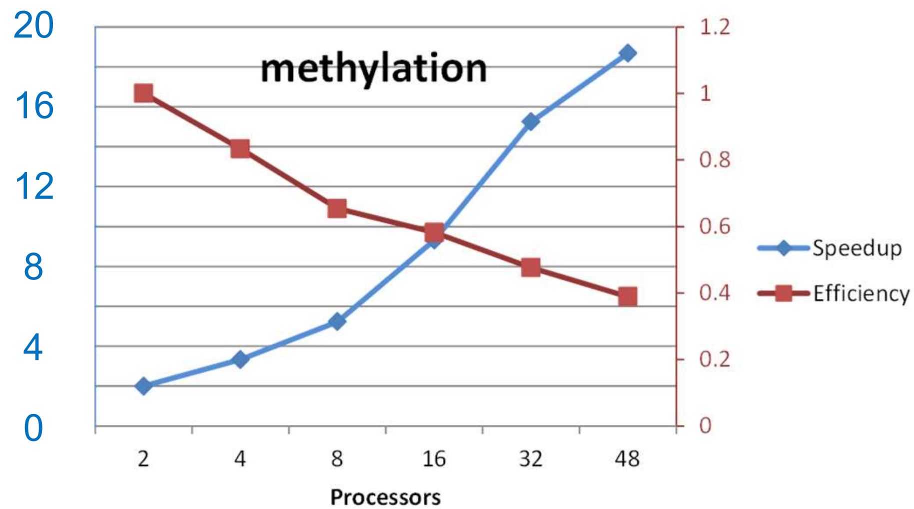
Performance Results

Social networks





A bio-medical network



Work in Progress

- More advanced partitioning methods to minimize the number of edges going across the processor boundaries
- A wider selection of networks to test the performance of the algorithm
- OpenMPI + OpenMP[1] for even greater parallel efficiency
- Update the code for distributed memory supercomputers
- Tackle billion-scale networks by conducting additional experiments on the IBM® System Blue Gene®/Q

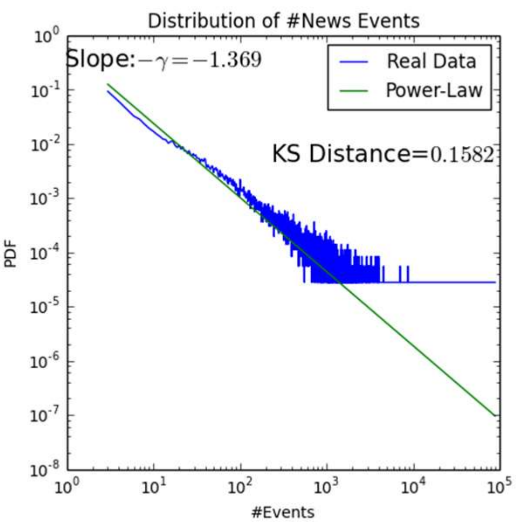
1 OpenMP ARB (Architecture Review Boards)
Image from http://www.timesunion.com/business/article/RPI-s-Amos-a-fast-study-4867074.php

Outline

- Motivation for this research
- Top-down bottom-up based community detection algorithm SpeakEasy
- Parallelization of SpeakEasy
- **Algorithm for Prediction of Viral News Cascades and its parallelization**
- Conclusions

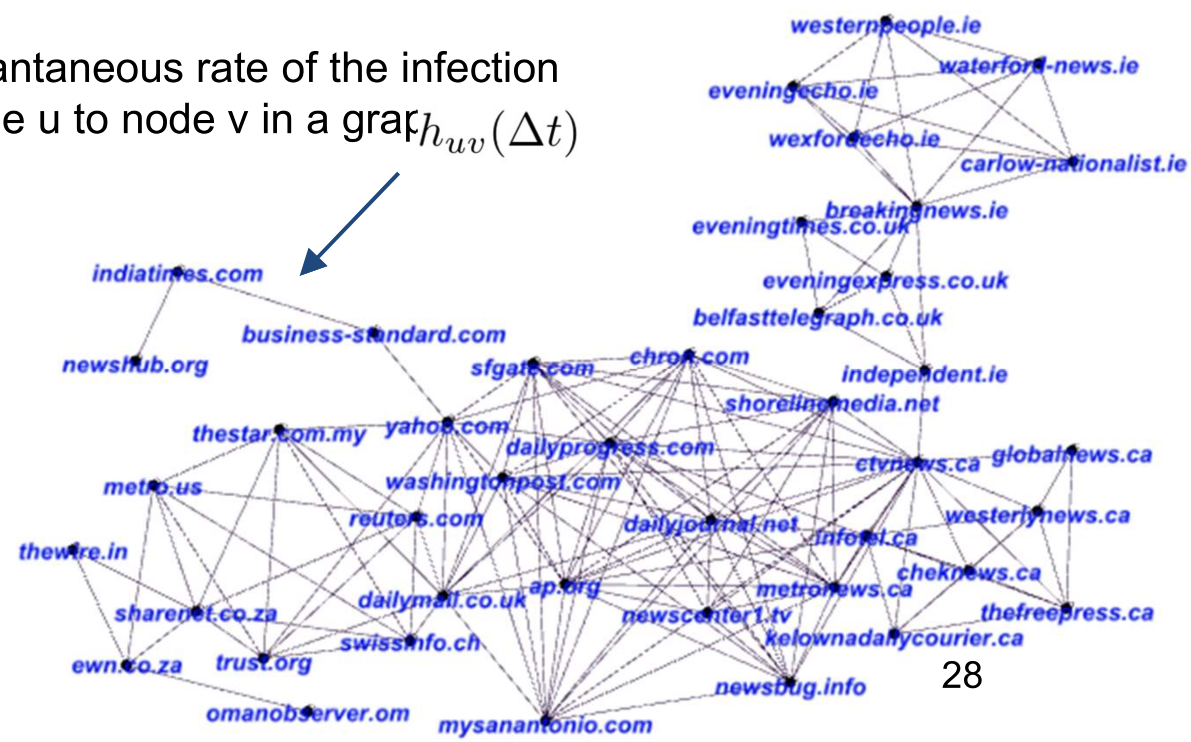
News Events in Online Media

- The spread of the news stories exhibits an emergent pattern in online media. **Can we predict viral news?**
- We use **survival analysis** to model the spread of news events from one online media site to its neighbors.



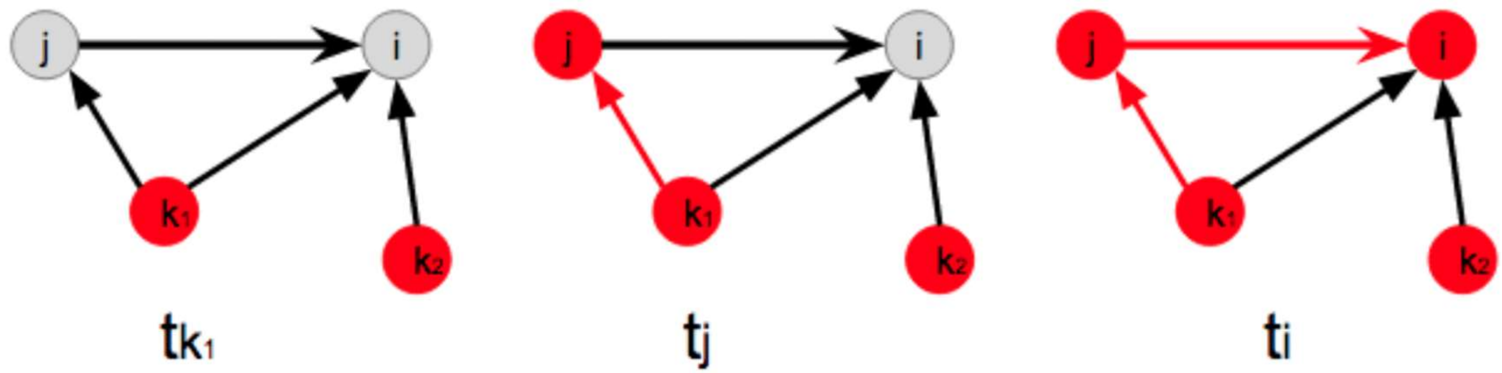
#news events per site follows the power-law

The instantaneous rate of the infection from node u to node v in a graph $h_{uv}(\Delta t)$



Survival Analysis

- **The stochastic propagation model** [Kempe 2003]:
  Infection delay through every link is independent.
  Once a node has been infected, it won't be infected again.



- According to the survival analysis [Infopath 2013]

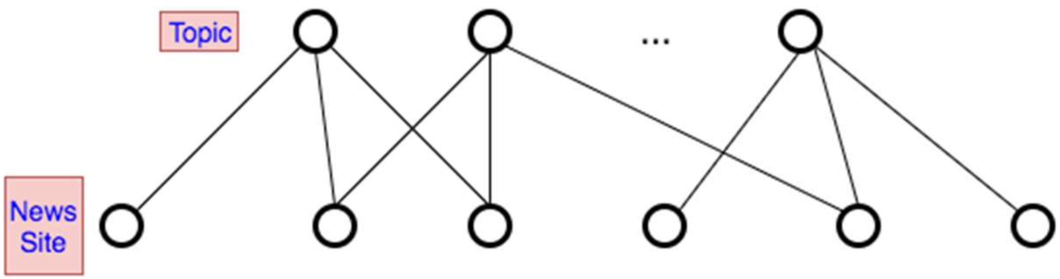
$$\mathcal{P}[j \text{ infects } i \text{ at time } t_i] = f_{j\to i}(t_i - t_j) \times S_{k_1\to i}(t_i - t_{k_1}) S_{k_2\to i}(t_i - t_{k_2})$$

Hazard Function
$$= \frac{f_{j\to i}(t_i - t_j)}{S_{j\to i}(t_i - t_j)} \times \prod_{l} S_{l\to i}(t_i - t_l)$$

where the survival function S(τ) denotes the probability that NO infection happens within the period of time τ.

Nodes vs. Edges

- Instead of modeling the **links**, we focus on the **nodes**. The number of latent variable becomes linear in the number of nodes.
- Topic Model:



where **A_{uk}** is the influence of node u on topic kl; ***B_{vk}*** is the selectivity of node v on topic k.
A common choice for the survival time distribution ***S_{uv}*** is the exponentially decaying. The minimum infection delay across the K topics follows the exponential distribution with intensity ***h_{uv}***

Influence Vector

$$h_{uv}(\Delta t) = \sum_{k=1}^{K} A_{u,k} B_{v,k} = A_u B_v^T$$

Selectivity Vector

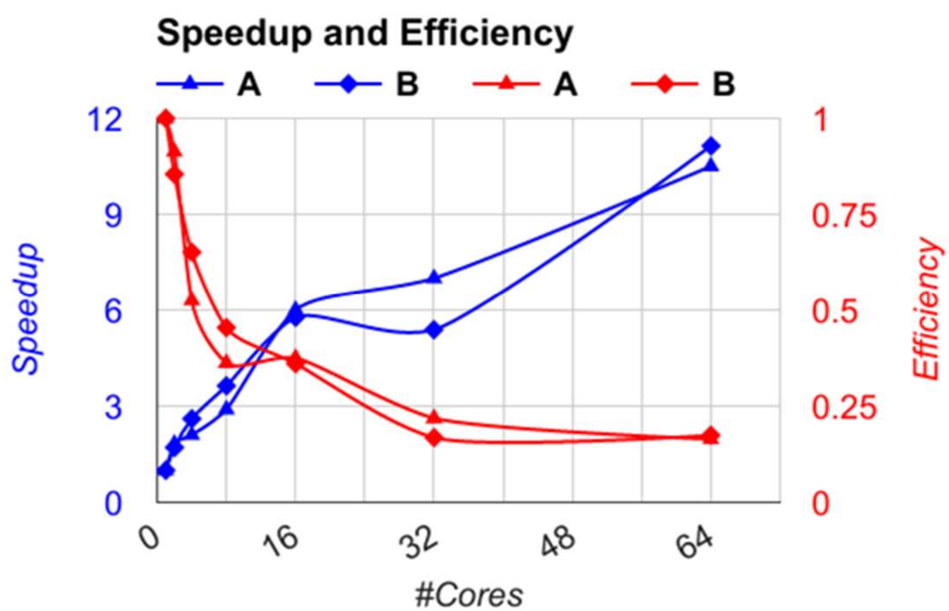
$$S_{uv}(\Delta t) = \prod_{k=1}^{K} e^{-A_{u,k} B_{v,k} \Delta t} = e^{-A_u B_v^T \Delta t}$$

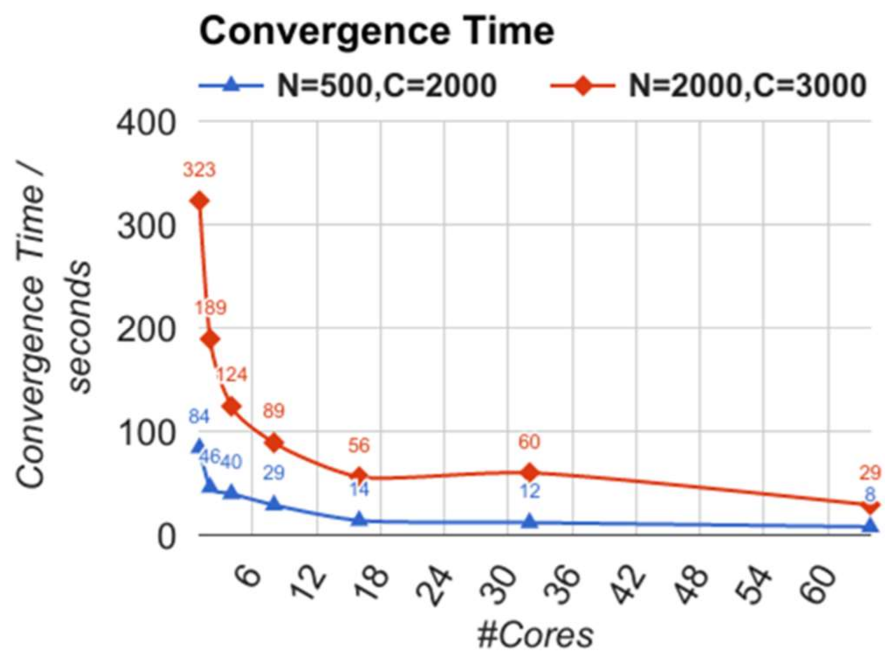
SCNARC

Parallelized Model Training on Shared Memory Machines

- Every process accepts an individual cascade and does gradient descent in parallel.
- Atomic Compare-And-Swap (CAS) operations to update the components of the influence and selectivity vector of the same node.

$$\nabla_{B_v}\mathcal{L}_c(A,B) = \sum_{l \prec_c v}(t_l - t_v)A_l + \frac{\sum_{u \prec_c v} A_u}{\sum_{u \prec_c v} A_u B_v^T}$$





Parallelized Model Training on Distributed Memory Machines

- On distributed memory machines, a **cascade layer** is proposed to reduce the inter-core communication caused by node-node connection in the survival analysis.
- The response time of a node to a cascade follows exponential distribution with rate parameter **A_u** ***M_c*** where ***M_c*** is the influence vector of a cascade.

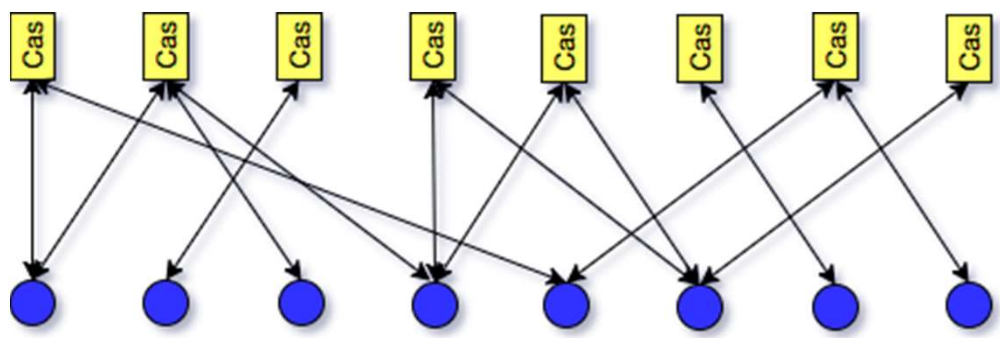
$$\mathcal{L}_c = \log L_c \approx \sum_{u \in V_c} \log \sigma(\langle A_u, M_c \rangle) - \sigma(\langle A_u, M_c \rangle) t_u^c$$

← **Positive** Samples

$$+ \frac{|V|}{|D_c|} \sum_{u \in D_c} \log \sigma(\langle A_u, M_c \rangle) - \sigma(\langle A_u, M_c \rangle) T$$

← **Negative** Samples

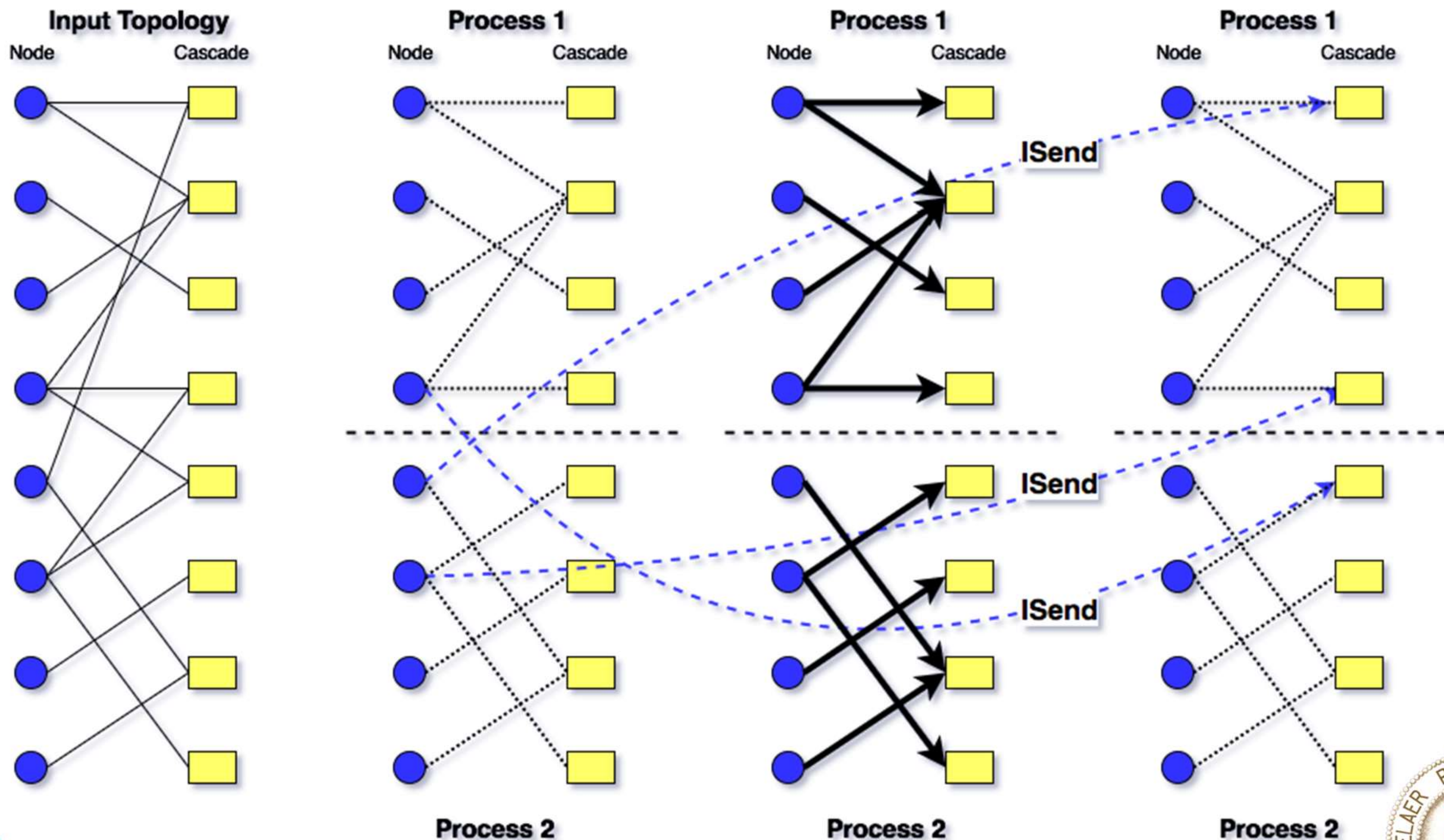
- The training algorithm propagates parameters between the cascade layer and node layer. A node (blue) is connected to all the cascades (yellow) in which it involves.



SCNARC

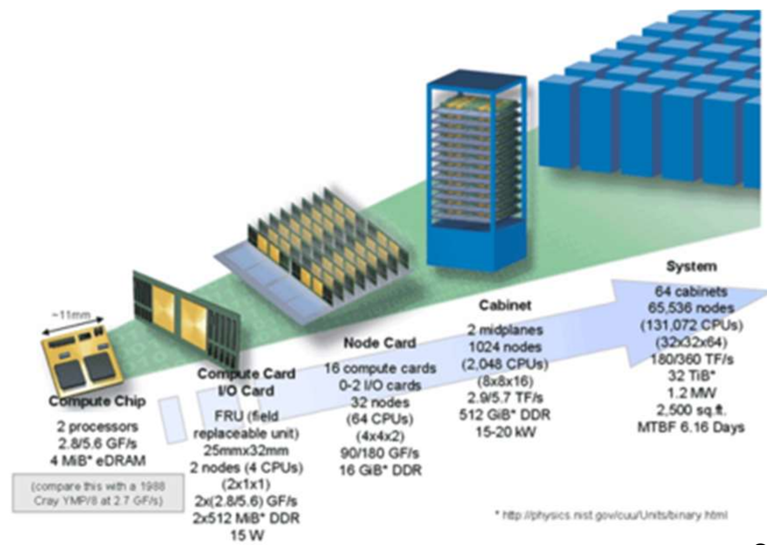
Parallelization Scheme for Distributed Memory Machines

- ➢ Asynchronous communication occurs between different processes while each process does internal computations.



AMOS Supercomputer @ Rensselaer

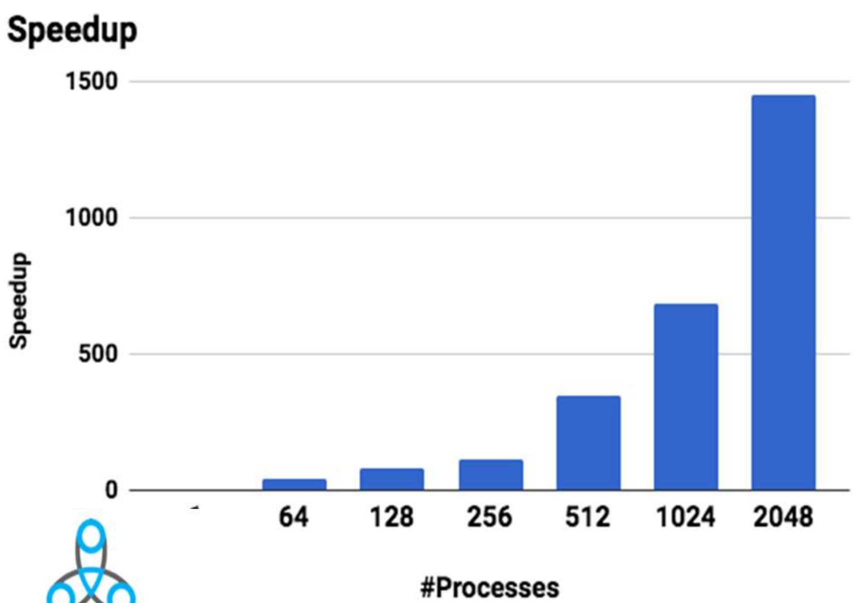
- **Advanced Multiprocessing Optimized System (AMOS)** is named after Amos Eaton, natural scientist, educator, and co-founder of the Rensselaer school.
- Ranked **No. 1** among supercomputers at private American academic institutions and **No. 3** among supercomputers at American academic institutions.
- The system is 5-rack, **5K nodes, 80K cores** IBM Blue Gene/Q with additional equipment.
- Each node consists of a **16-core**, **1.6 GHz A2 processor**, with 16 GB of DDR3 memory.

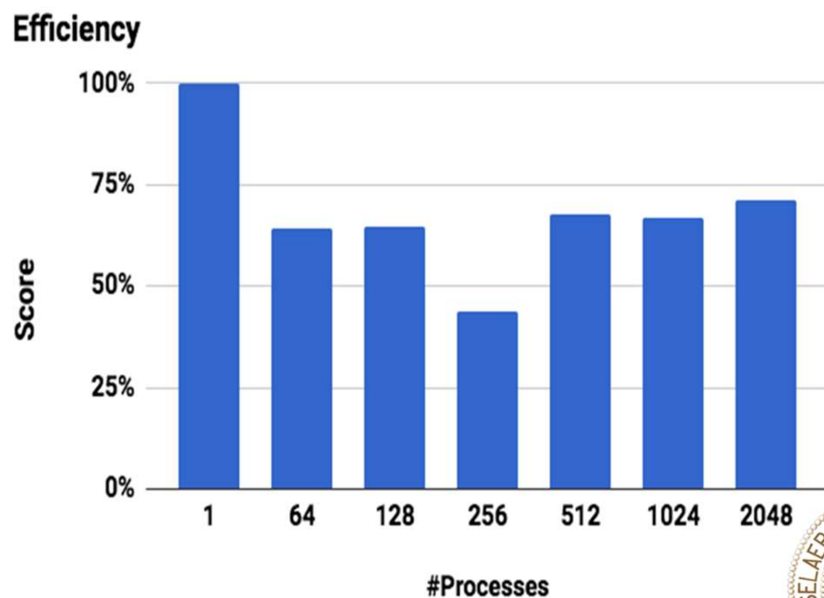


SCNARC

Speedup and Efficiency on AMOS Supercomputer

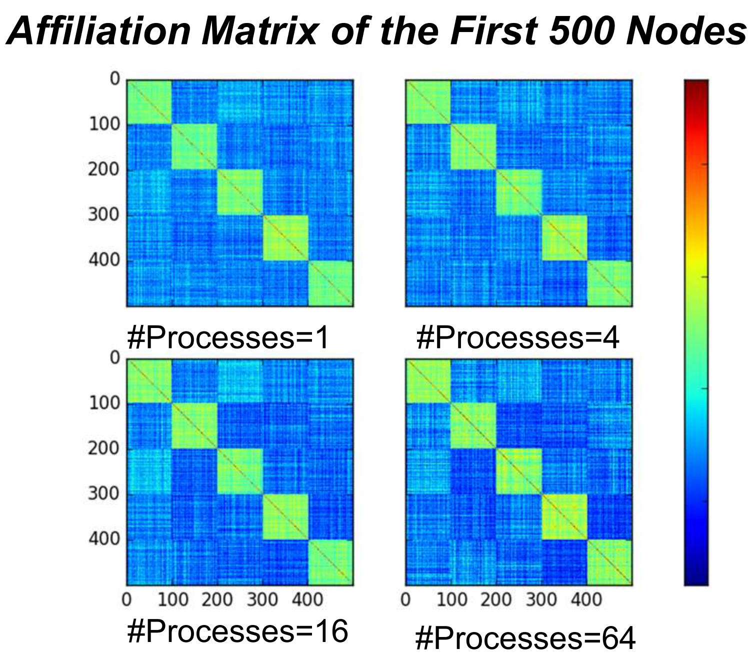
- **Input**: **1 million** cascades in a network with **2 million** nodes.
- Every node of the AMOS system uses 4 cores. Each core has an independent local memory for the embeddings associated with its own nodes/cascades and a ghost memory for the embeddings associated with remote nodes/cascades.

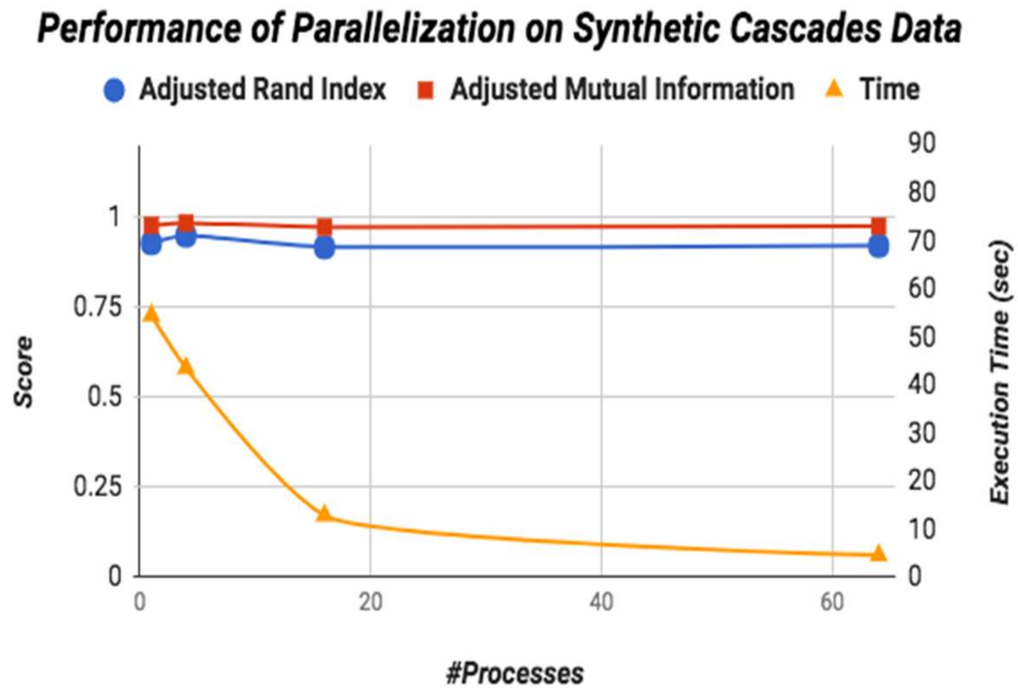




Parallelization Performance on Community Detection

➢ The parallelization scheme preserves the quality of the resulting node embeddings.

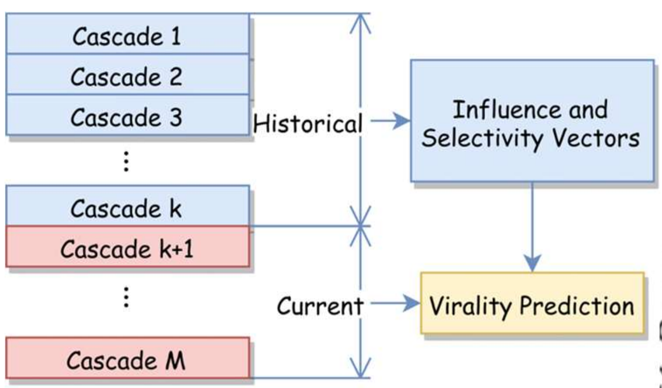




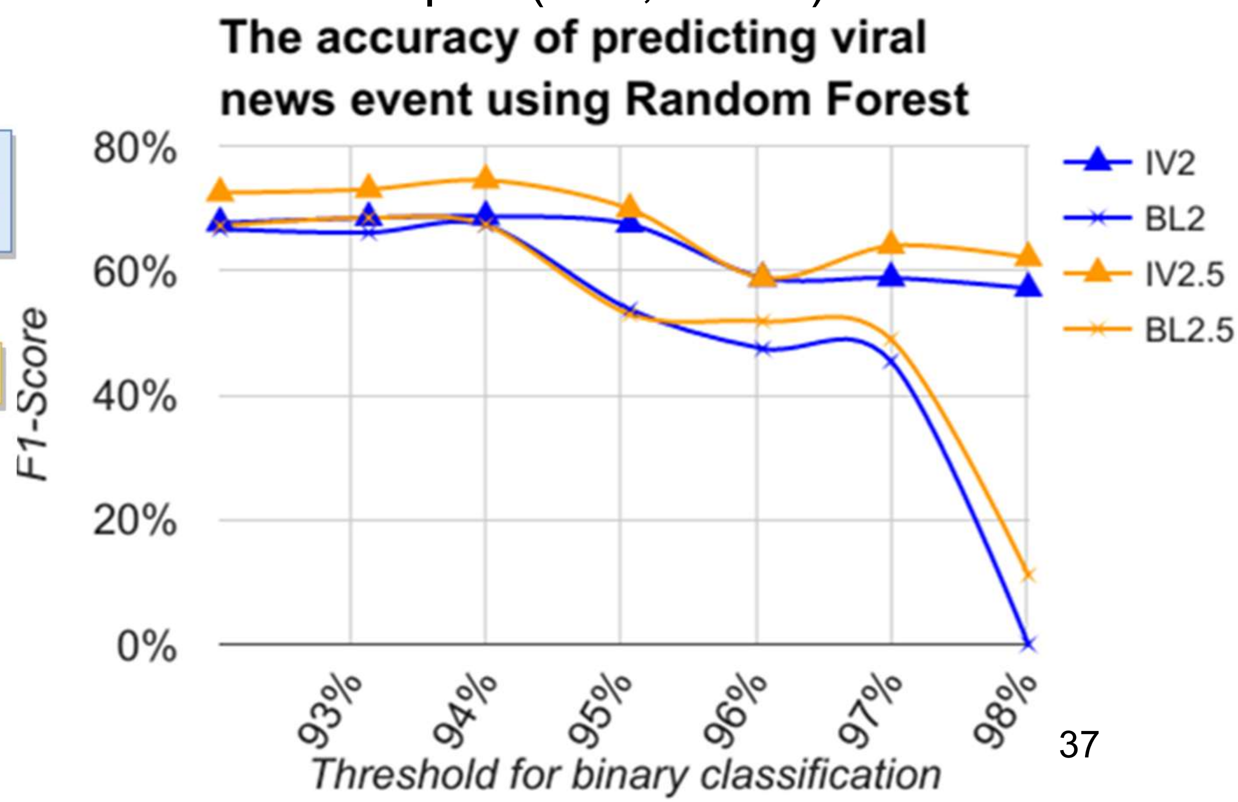
5K cascades simulated on a Stochastic Blockmodel (SBM) network with **10K** nodes. We evaluate the quality of the community discovered by ***K-means algorithm*** based on the vector representation of nodes.

Virality Prediction of Online News Cascades

- **Task: Predict the final number of news sites reporting an emergent news event.**
  - The summation of the influence vectors of the early adopters in the first 2 or 2.5 hours is used as the input. (IV2, IV2.5)
  - A baseline model uses features including number of early adopters, time intervals etc. as input. (BL2, BL2.5)



#News Sites = 5634
#News Events = 41452
(K=35000)



Outline

- Motivation for this research
- Top-down bottom-up based community detection algorithm SpeakEasy
- Parallelization of SpeakEasy
- Algorithm for Prediction of Viral News Cascades and its parallelization
- **Conclusions**

Conclusions

- **Social networks, and complex networks in general create new challenges for parallel computing**
  - ➢ Node degrees may vary widely in networks (e.g. scale-free networks), unlike spatial locality of cause and effect in physical world
  - ➢ Size of the networks could be enormous creating challenged for memory
- **The benefits and shortcomings of parallel multi-core shared memory machine and supercomputers are different than for large scale numerical computations**
  - ➢ Share memory simplifies parallelization, and is affordable, but limited in terms of final speedup
  - ➢ Supercomputers are expensive, difficult to program, but can achieve higher speedup even though it comes with lower efficiency.

Thank You

Questions

?