SCAN-XP:

Parallel Structural Graph Clustering Algorithm on Intel Xeon Phi Coprocessors

Hiroaki Shiokawa

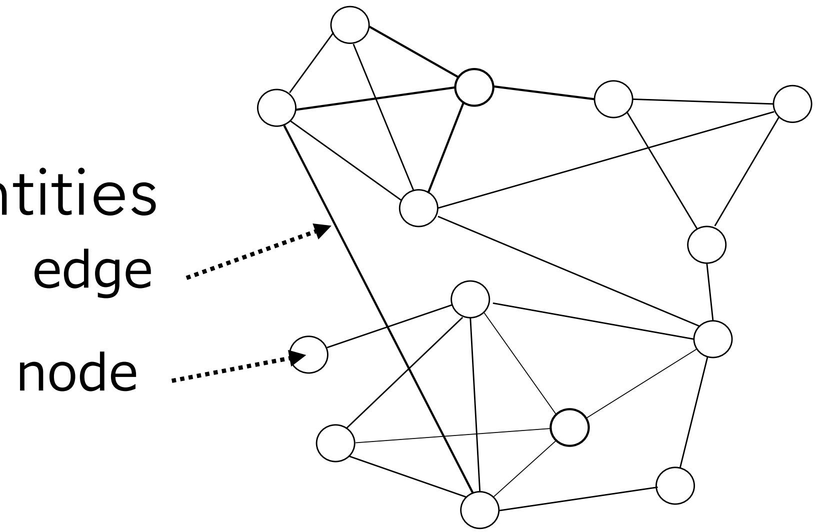
Database Group, Division of Computational Informatics,

CCS, University of Tsukuba

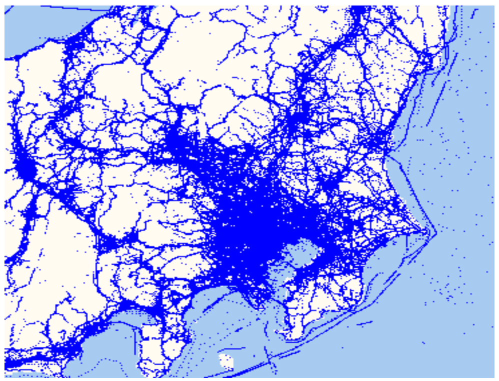
2017 CCS-EPCC Workshop

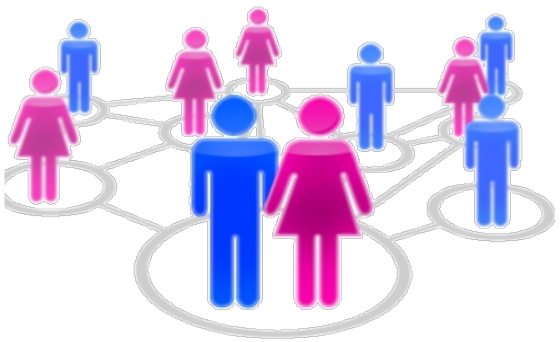
Graph and its applications

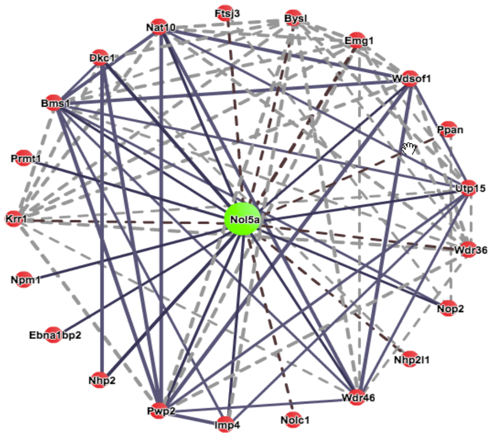
- Graph
  - Node: data entities
  - Edge: relationships among entities



- Key applications



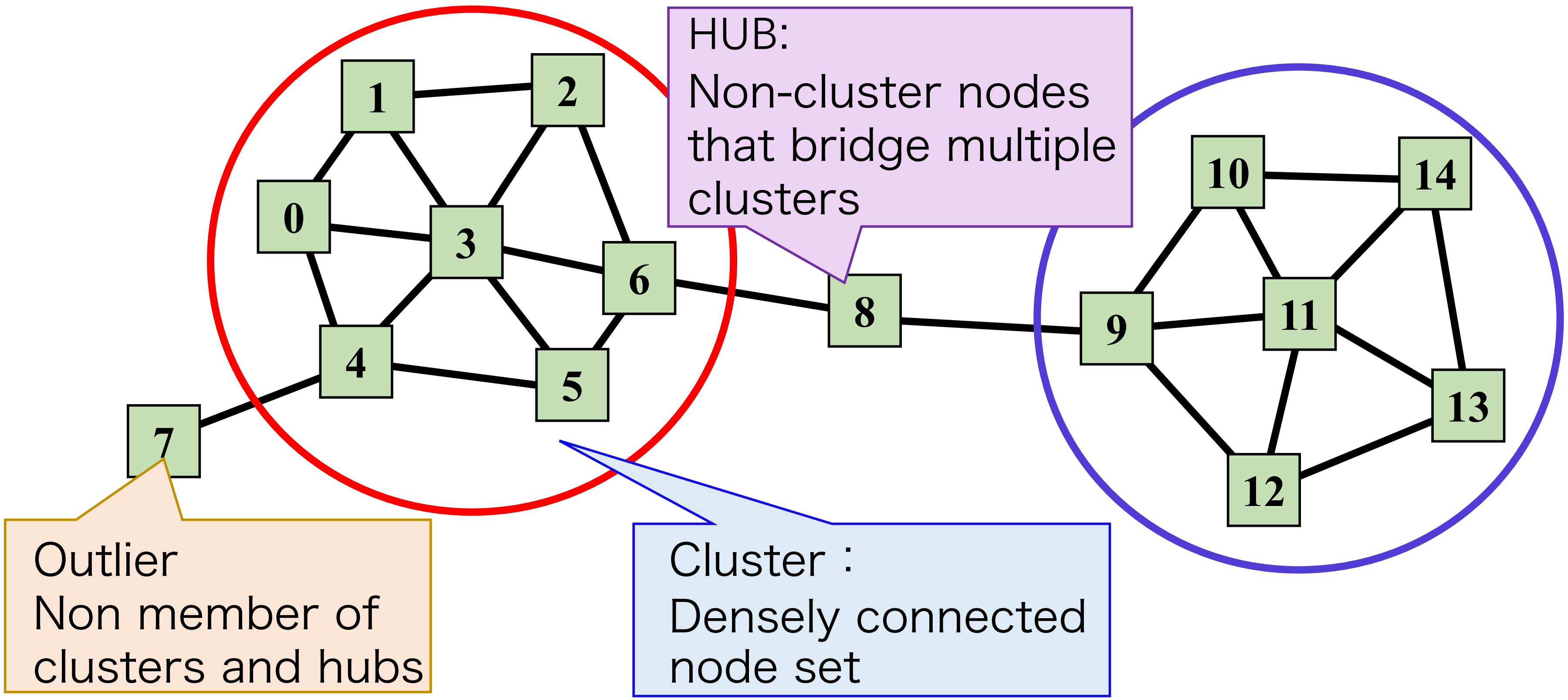




Roads and Trajectories **Web and SNS** **Protein-protein interactions**

Structural Graph Clustering: SCAN

- **SCAN** [Xu+,2007]
  - SCAN identifies clusters, hubs and outliers based on density between two nodes



Large-scale Graphs are now available

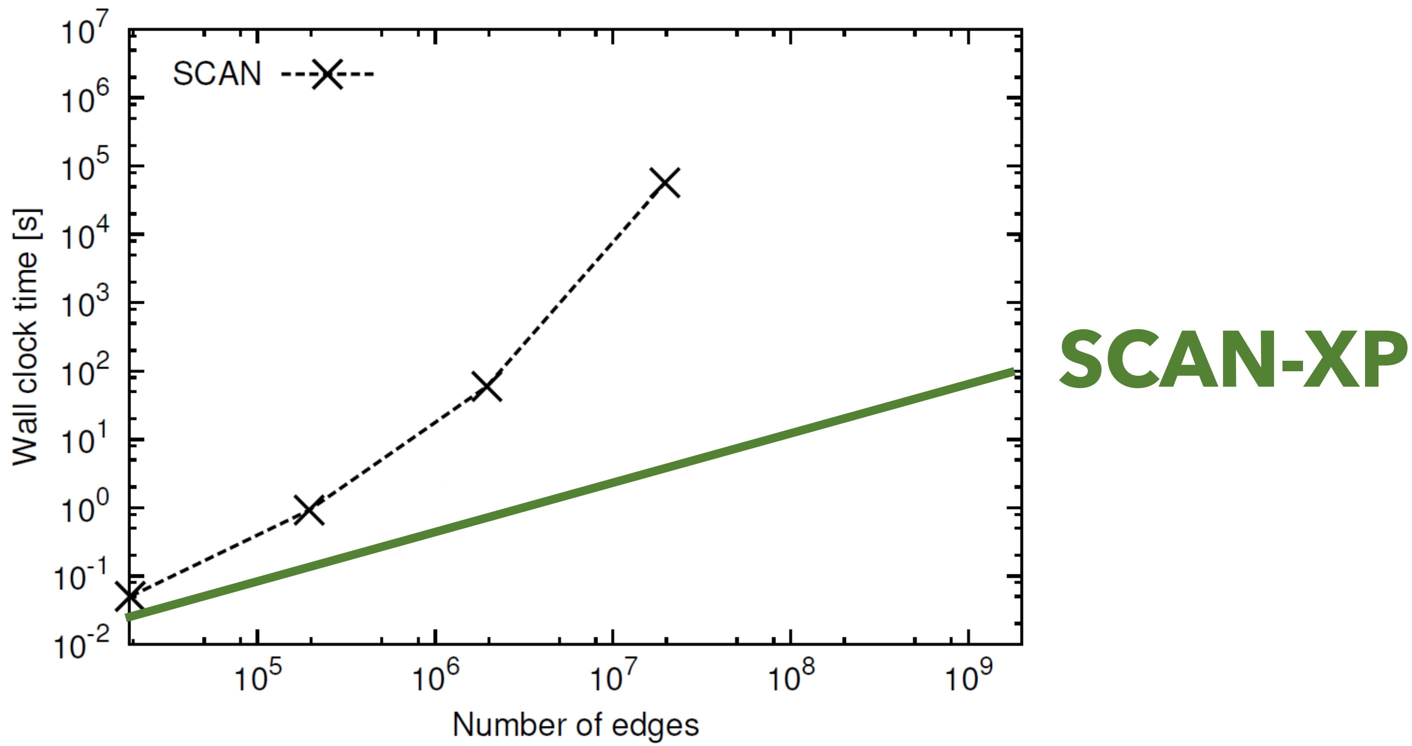
1.49 Billion Users/Month

500 Million Tweets/ Day
320 Million Users/Month

How we can efficiently find clusters on a Large-scale Graphs?

Our Contributions

- Proposed method **SCAN-XP**
  - Scaling SCAN using Intel Xeon Phi Coprocessor
  - We examined its efficiency on COMA and Oakforest-PACS



Baseline: SCAN

Clustering procedure of SCAN

- **Cluster** = **Cores** and its **densely connected neighbors**
  - **Core:** Nodes that have enough neighbors with dense connections

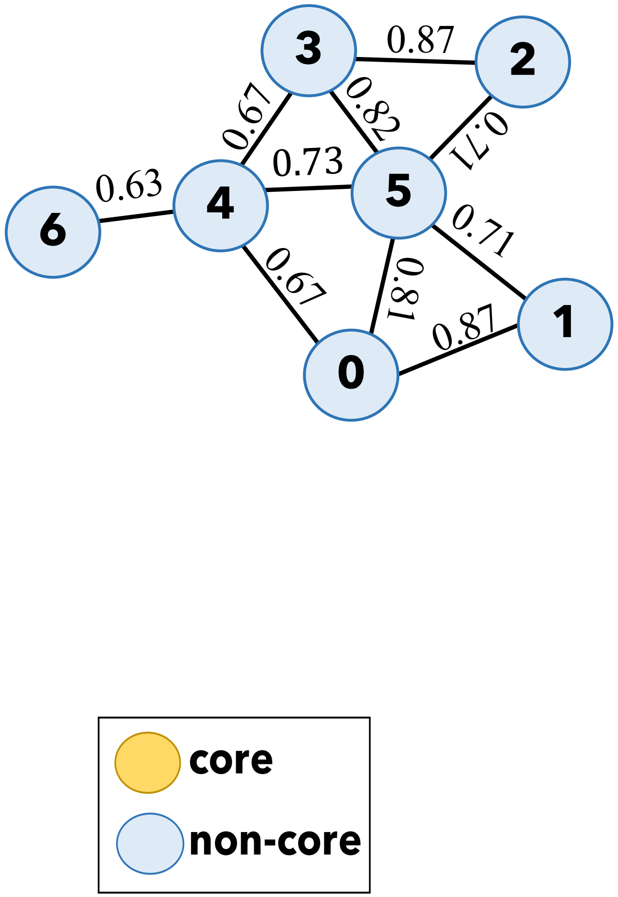
    Structural similarity $\sigma(v,w)$

    $$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

  - By setting **density threshold ϵ** and **minimum cluster size μ**, SCAN specifies the clusters.

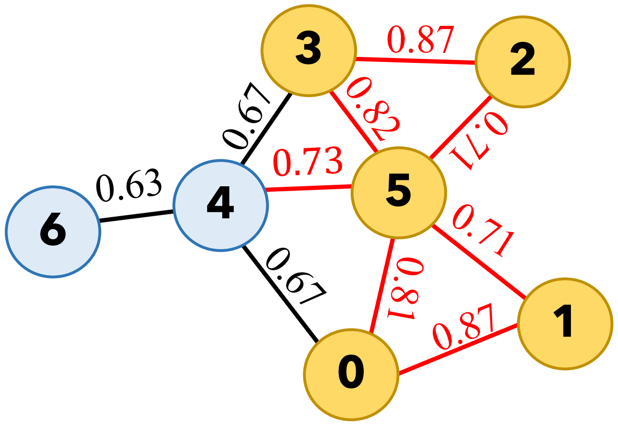
Example of SCAN ($\varepsilon = 0.7, \mu = 2$)

Step 1
Core detection

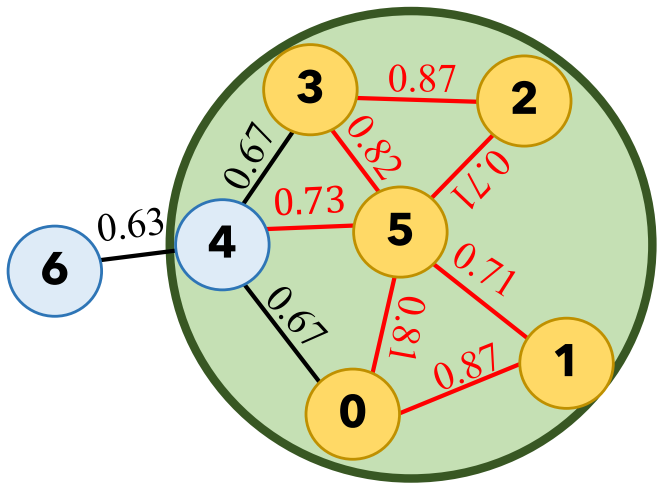


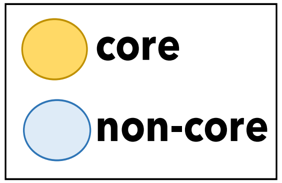
Example of SCAN ($\varepsilon = 0.7, \mu = 2$)

Step 1
Core detection



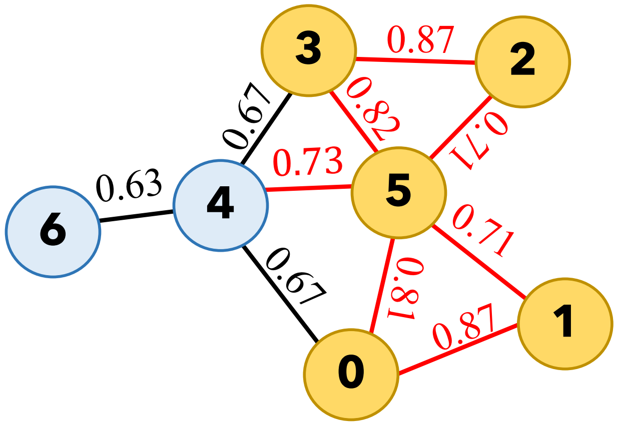
Step 2
Cluster construction

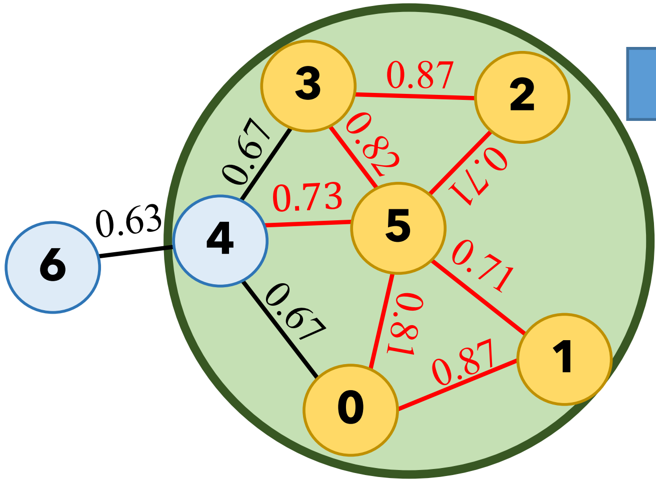


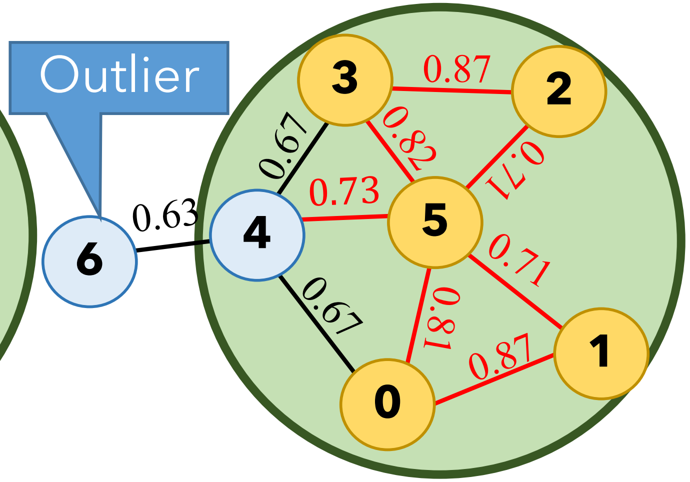


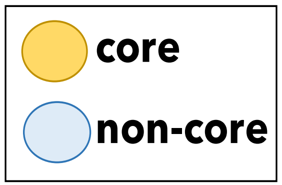
Example of SCAN ($\varepsilon = 0.7, \mu = 2$)

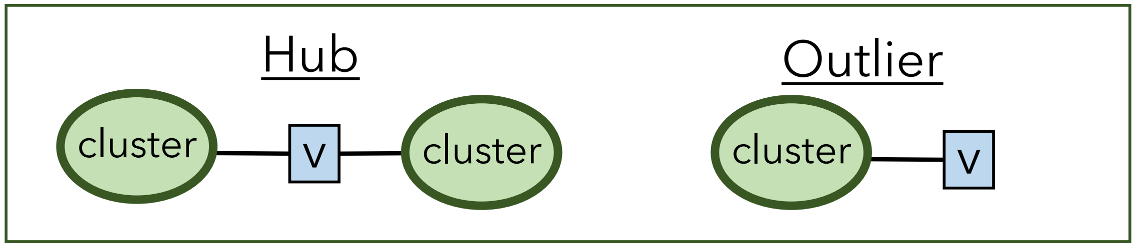
Step 1 Core detection	Step 2 Cluster construction	Step 3 Hub・Outlier detection









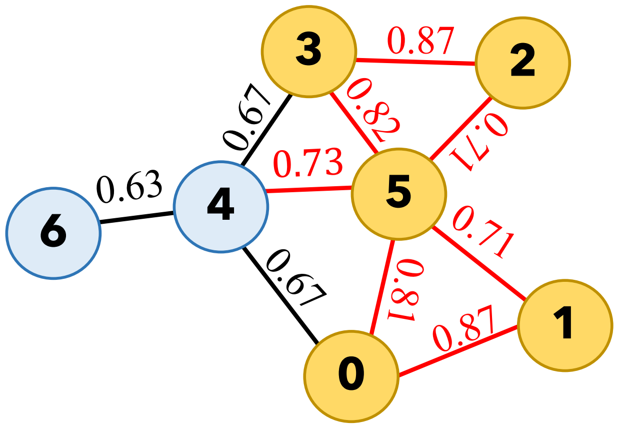


Proposed Method: SCAN-XP

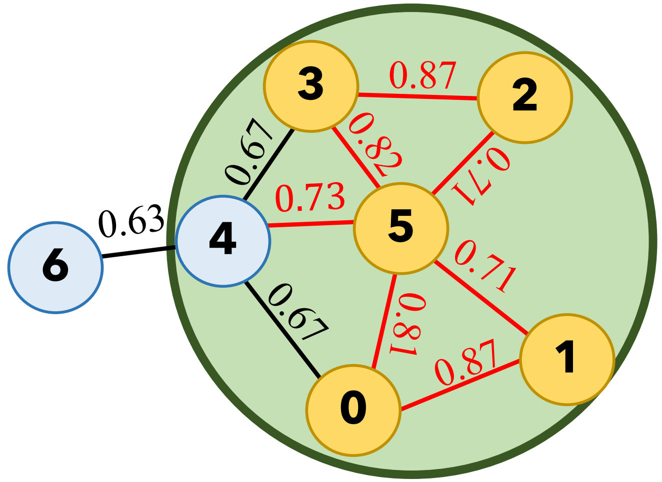
Proposed method: SCAN-XP

- **Core detection** and **Cluster construction** are bottlenecks
  - They require exhaustive computations…

Step 1: Core detection



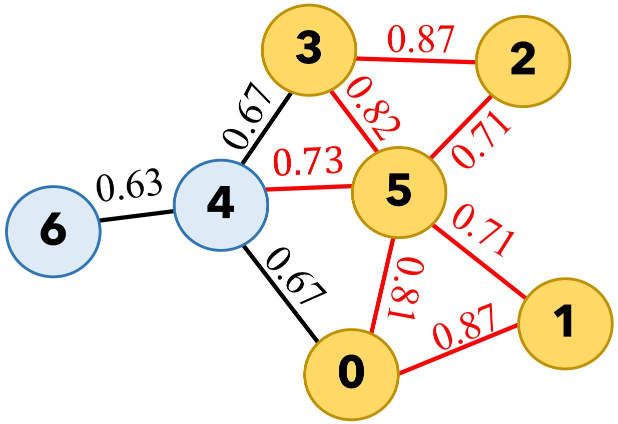
Step 2: Cluster construction



- Proposed method: SCAN-XP
  - Step1: Thread-based & SIMD-based parallelization
  - Step2: Thread-based parallelization using Union-Find Tree

Step1: Parallel Core Detection

- Thread-based parallelization is trivial
  - The structural similarity computation $\sigma(u,w)$ is independent among edges ☺



- Set Intersection in $\sigma(u,w)$

$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

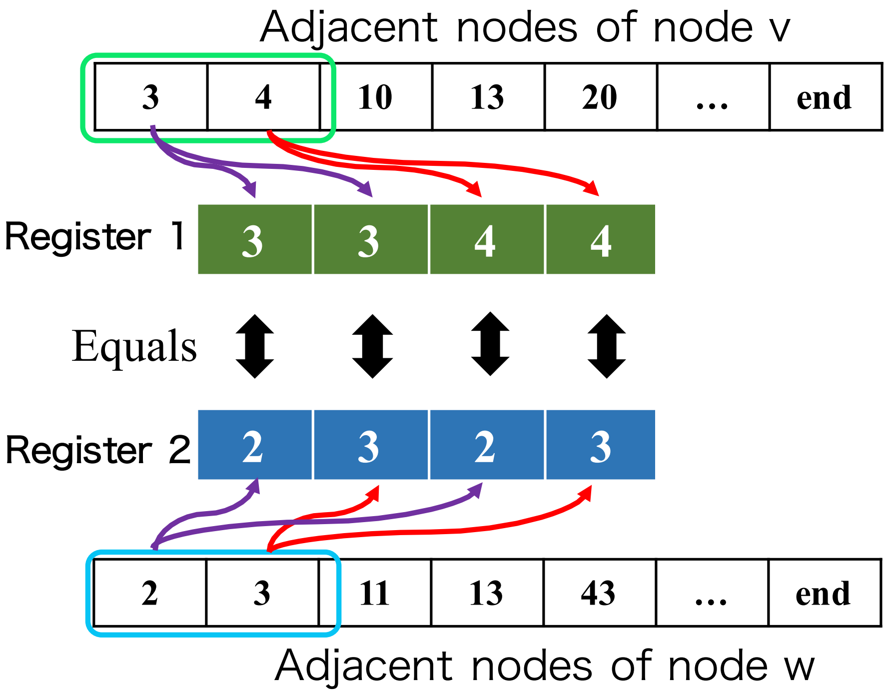
Parallelized Set Intersection is not trivial ☹

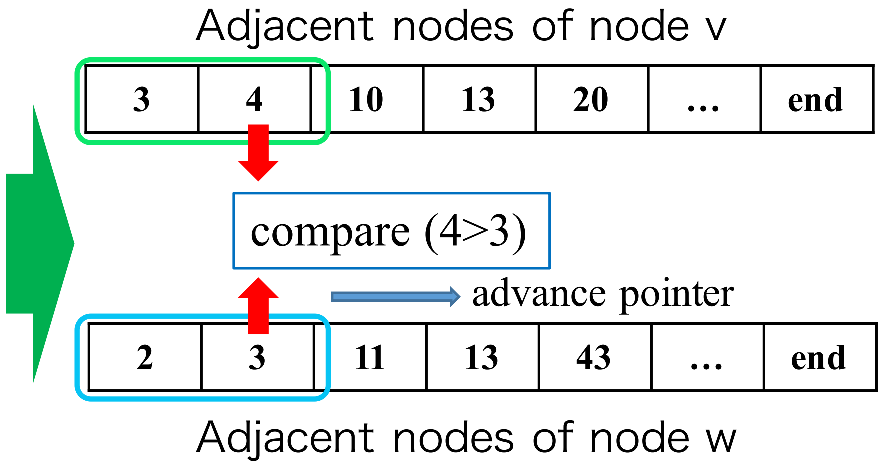
SIMD-based Sort Merge Join (SMJ)

- SMJ is a lightweight set intersection algorithm

$$\Gamma(v) = \{3, 4, 10, 13, 20, \ldots\}, \Gamma(w) = \{2, 3, 11, 13, 43, \ldots\}$$

$$|\Gamma(v) \cap \Gamma(w)| = ?$$



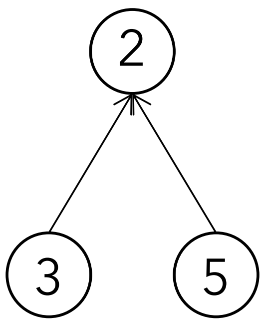


Step2: Parallel Cluster Construction

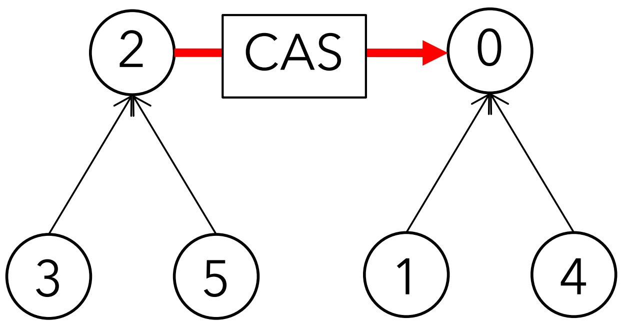
- SCAN needs to expand a cluster from a set of cores step by step ☹
  - Parallel cluster construction is not trivial

- Parallel Union-Find Tree (UFT) construction
  - Assign threads to nodes, and construct UFT in parallel

Cluster = {2, 3, 5}



Union(3,4)



Evaluations

Experimental settings

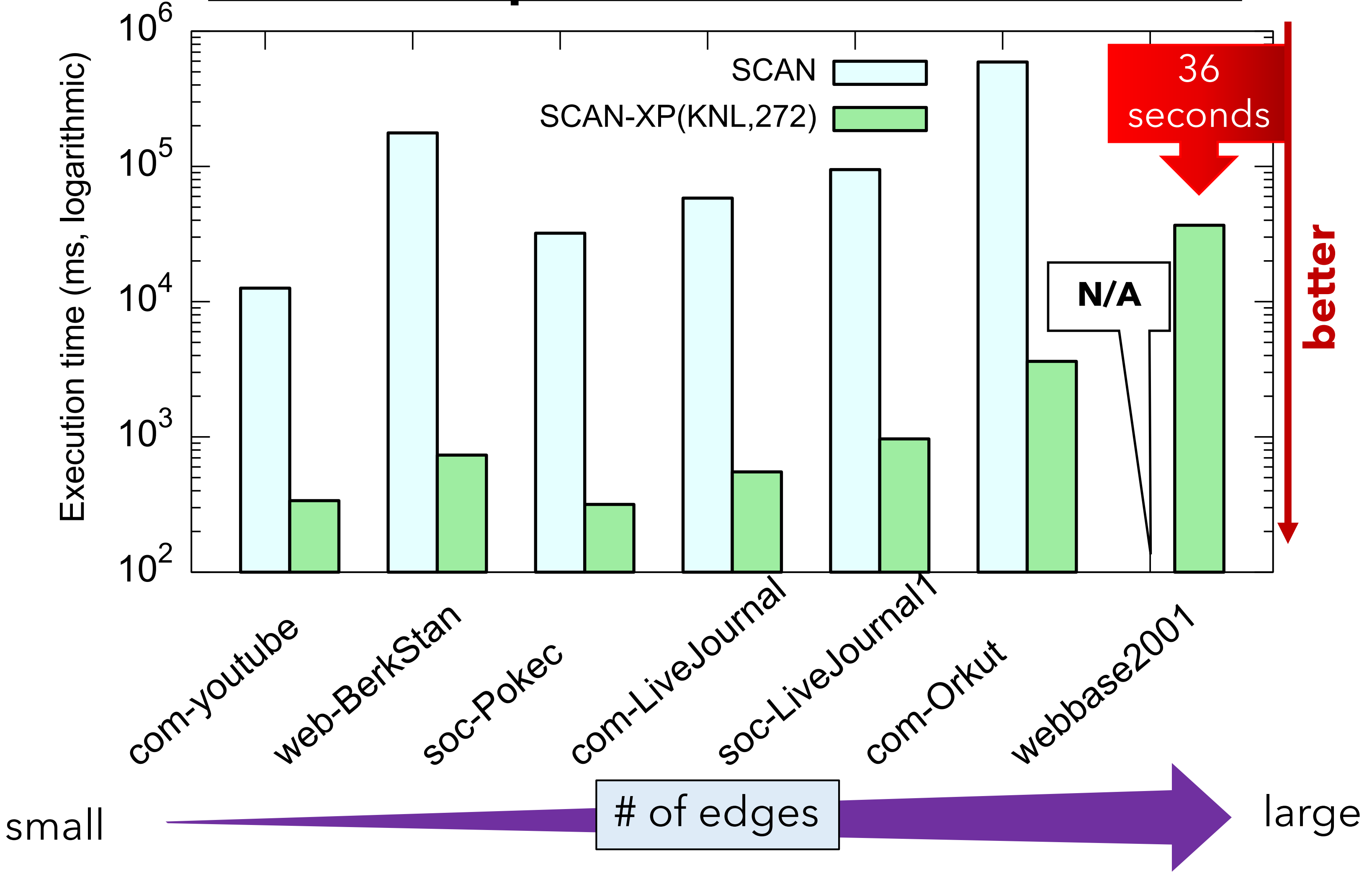
- Real-world Datasets

Dataset	# of nodes	# of edges
com-youtube	1,134,890	2,987,624
web-BerkStan	685,230	6,649,470
soc-Pokec	1,632,803	22,301,964
com-LiveJournal	3,997,962	34,681,189
soc-LiveJournal1	4,846,609	42,851,237
com-Orkut	3,072,441	117,185,083
webbase2001	115,554,441	854,809,761

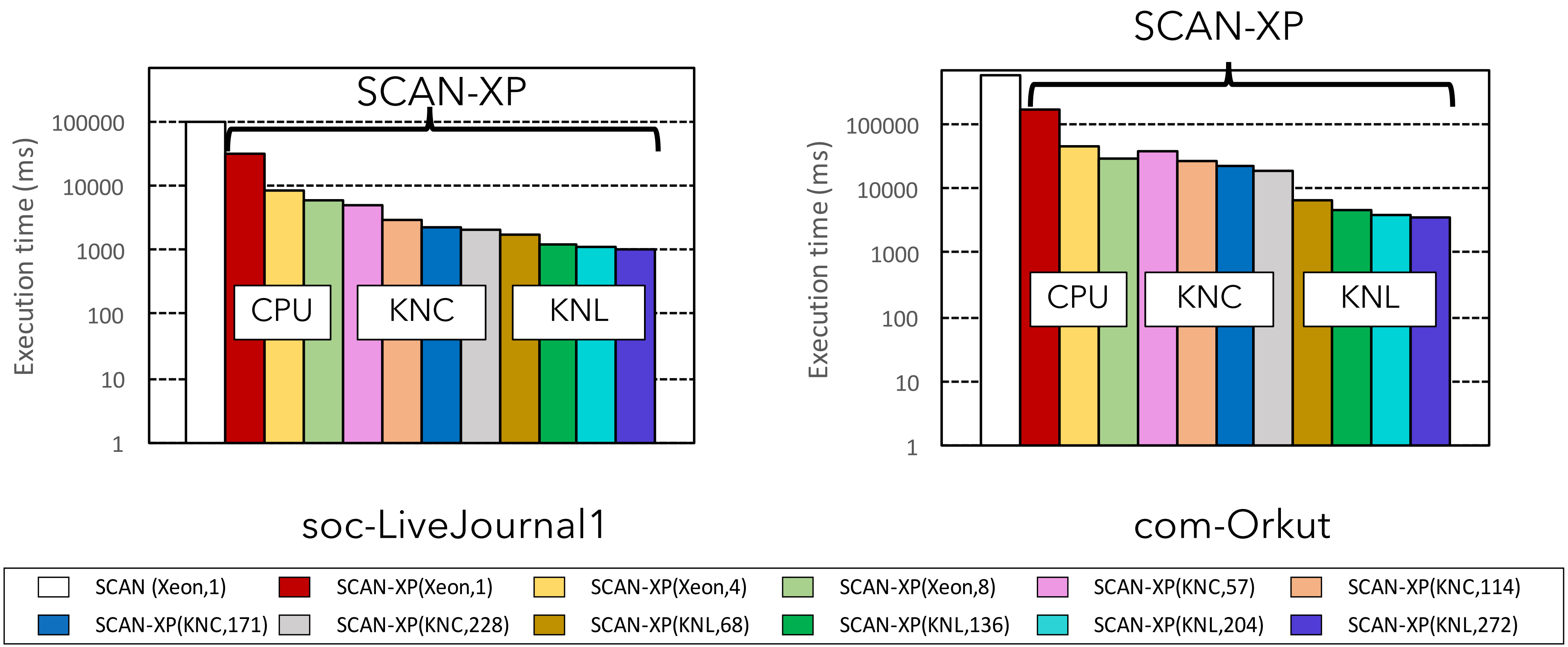
- Experimental environment
  - CPU ：Processor Xeon E5 1620
  - KNC ：Intel Xeon Phi 7110P
  - KNL ：Processor Xeon Phi 7250

Execution time

SCAN-XP is up to 100 times faster than SCAN



Performance comparison (CPU,KNC,KNL)



Conclusion

- Summary
  - We proposed **SCAN-XP**
  - SCAN-XP is 100 times faster than SCAN

T. Takahashi, H. Shiokawa, H. Kitagawa,
"SCAN-XP: Parallel Structural Graph Clustering on Intel Xeon Phi Coprocessors,"
In *Proc. SIGMOD 2017 Workshops on Network Data Analysis*, 2017

- Future works
  - Employ pruning approaches into SCAN-XP
  - Exploit multiple Xeon Phi for significantly large graphs