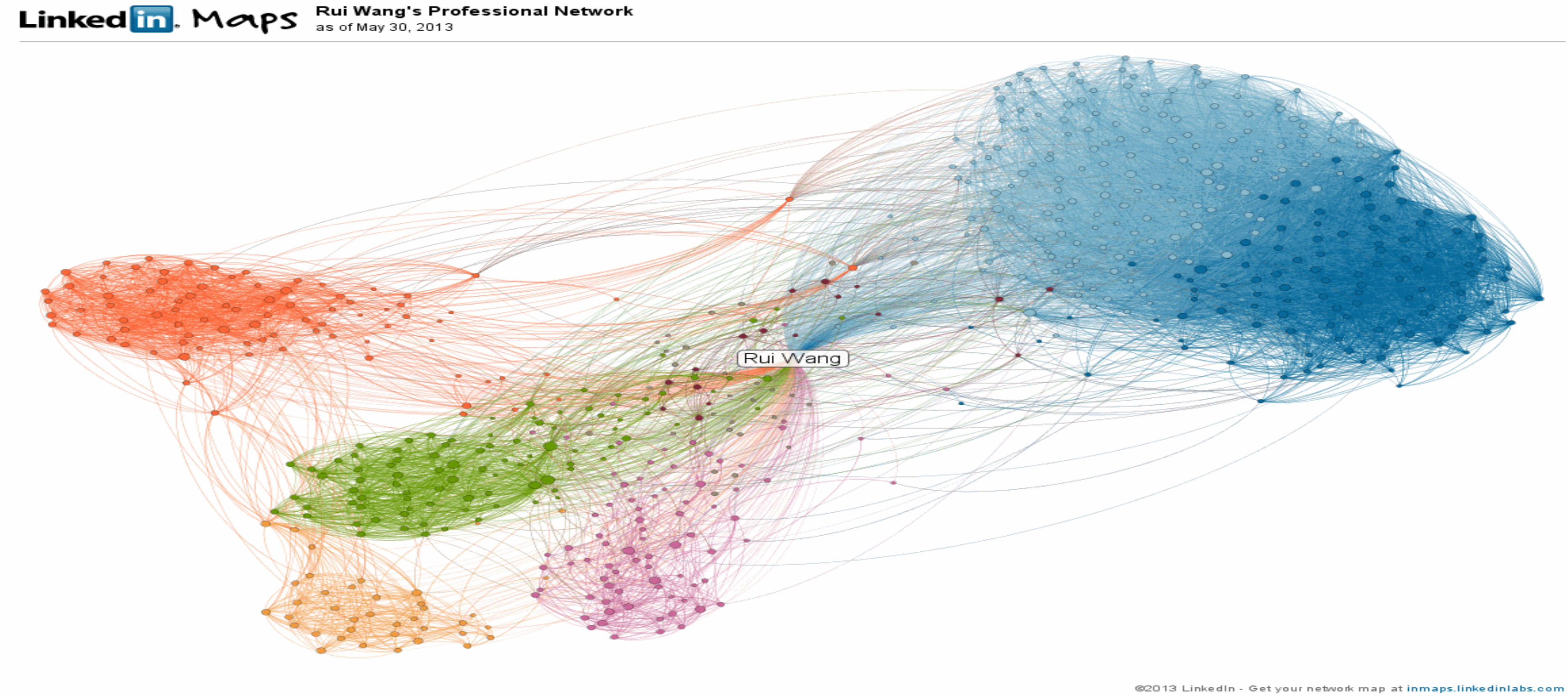


Using Set Cover to Optimize a Large-Scale Low Latency Distributed Graph

Rui Wang
Staff Software Engineer

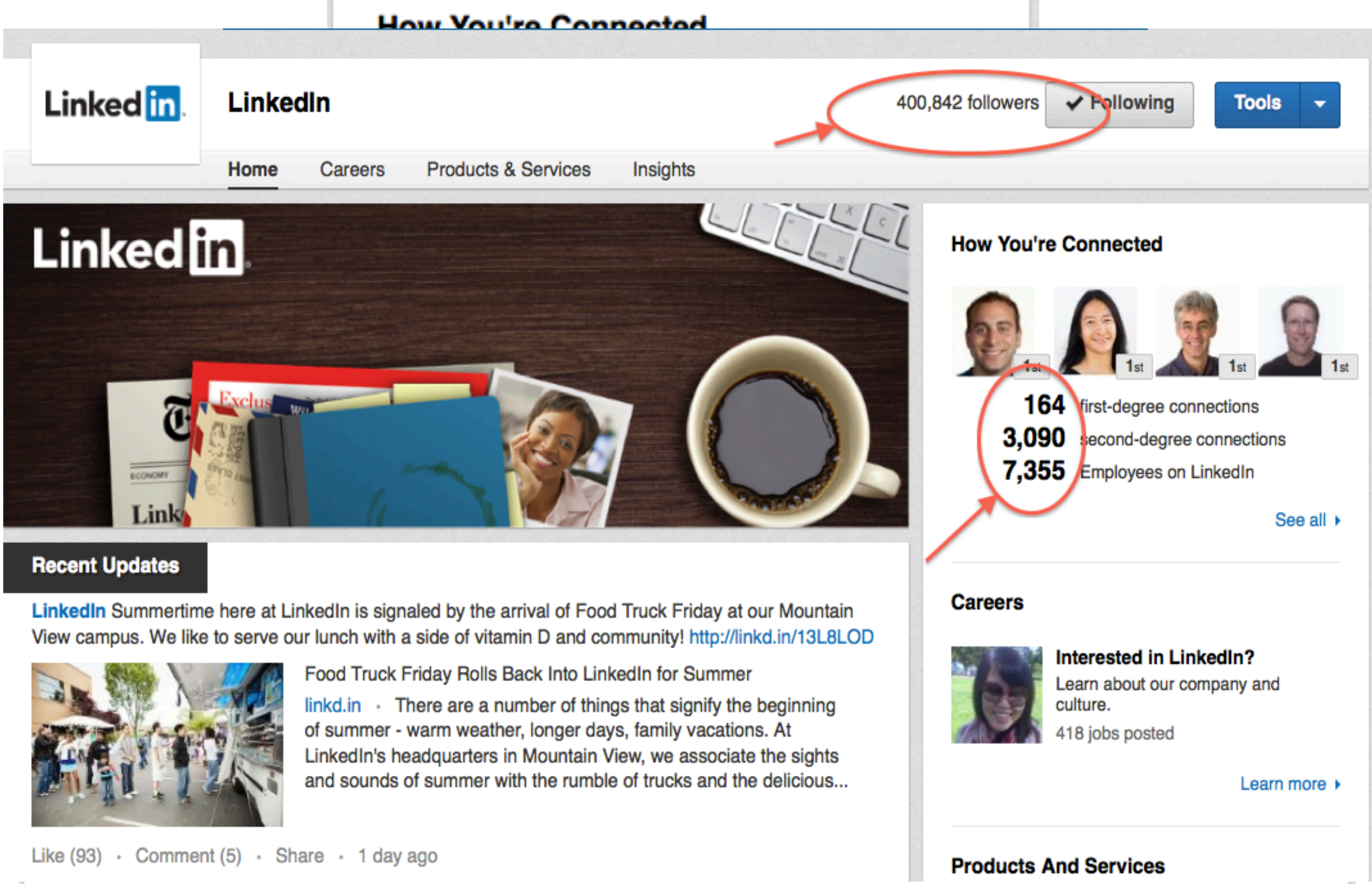
Outline

- LinkedIn’s Distributed Graph Infrastructure
- The Scaling Problem
- Set Cover by Example
- Evaluation
- Related Work
- Conclusions and Future Work

Outline

- LinkedIn's Distributed Graph Infrastructure
- The Scaling Problem
- Set Cover by Example
- Evaluation
- Related Work
- Conclusions and Future Work

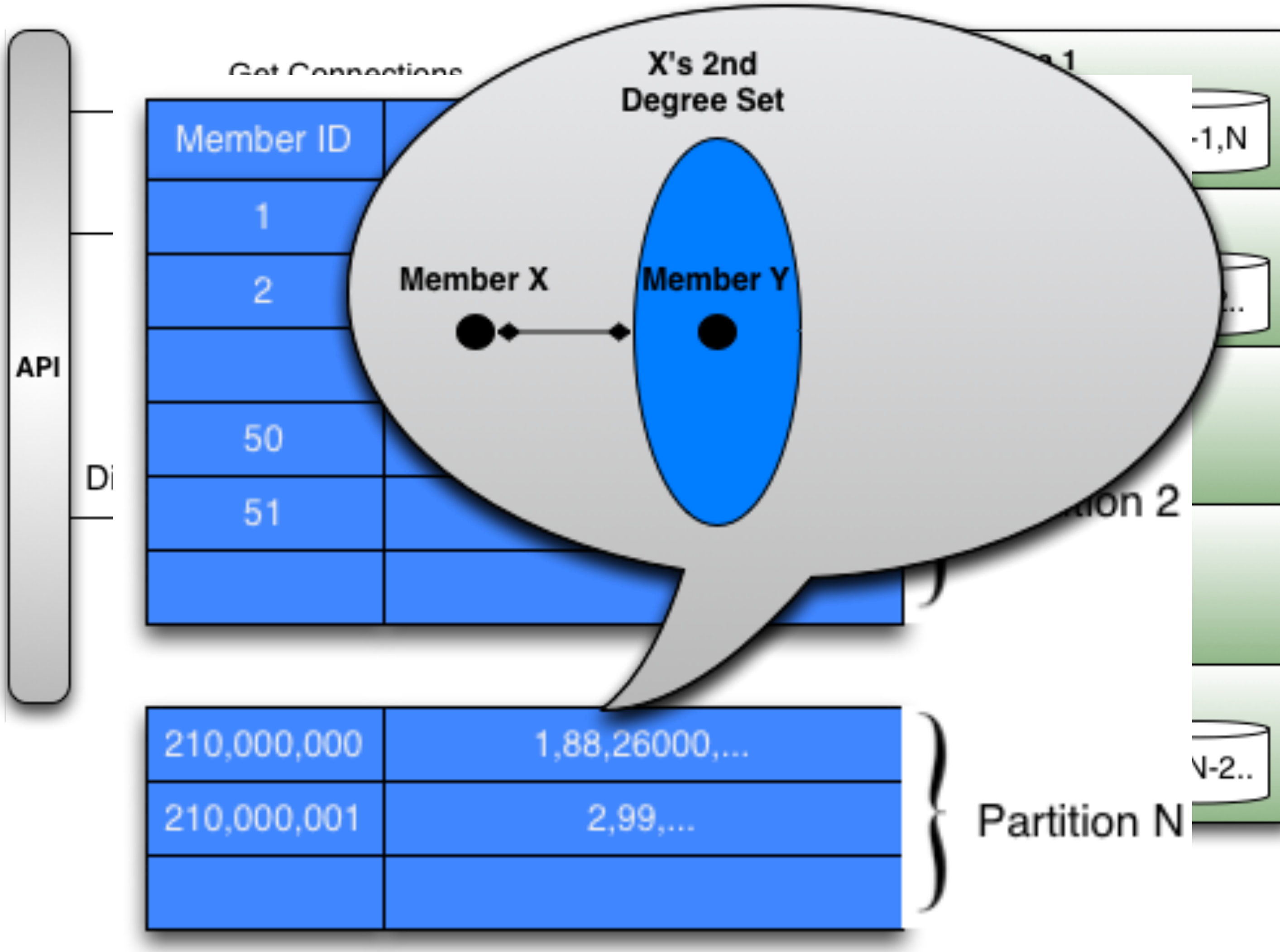
LinkedIn Products Backed By Social Graph



LinkedIn’s Distributed Graph APIs

- Graph APIs
  - Get Connections
    - “Who does member X know?”
  - Get Shared Connections
    - “Who do I know in common with member Y”?
  - Get Graph Distances
    - “What’s the graph distances for each member returned from a search query?”
    - “Is member Z within my second degree network so that I can view her profile?”
- Over 50% queries is to get graph distances

Distributed Graph Architecture Diagram



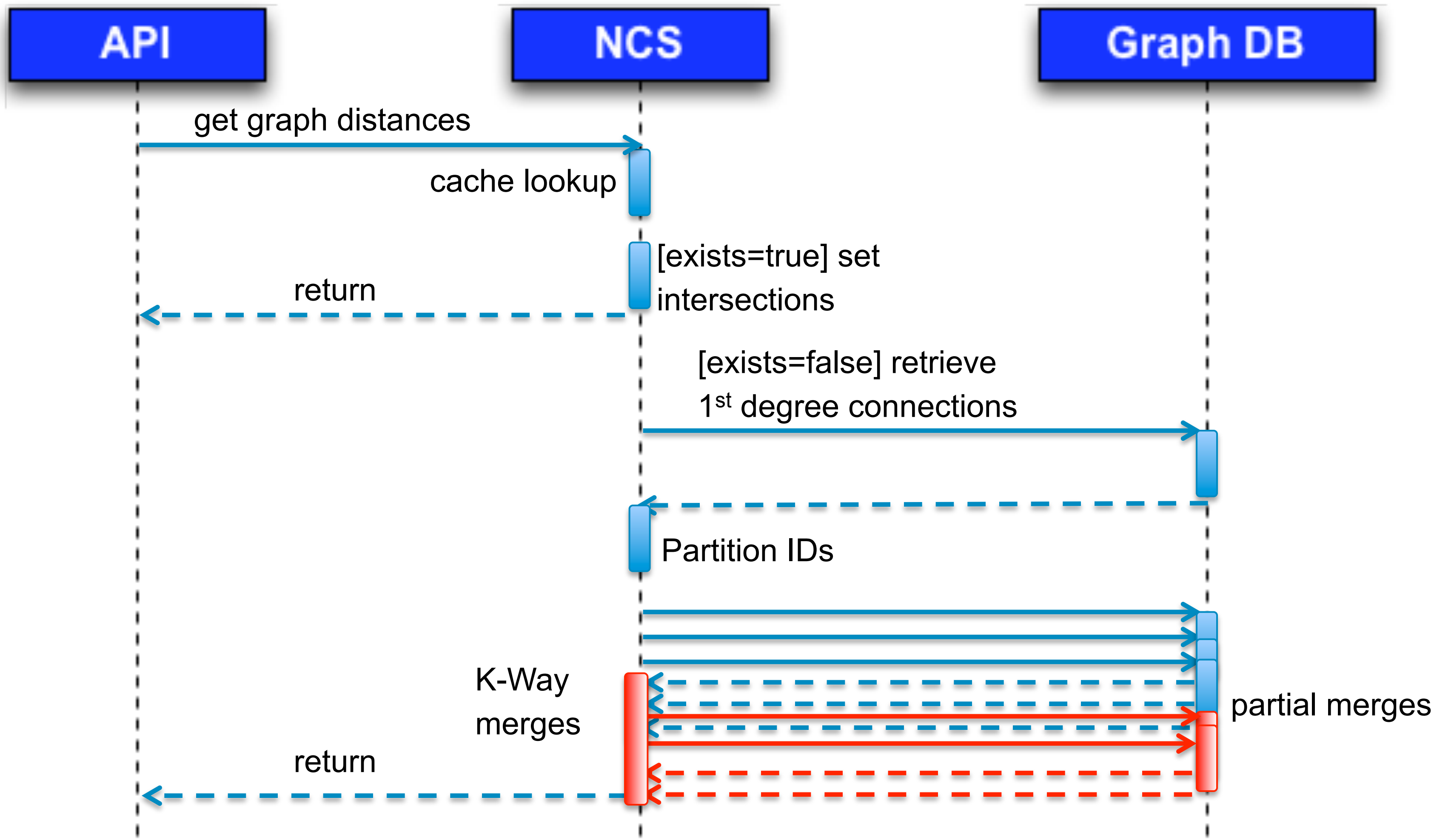
LinkedIn’s Distributed Graph Infrastructure

- Graph Database (Graph DB)
  - Partitioned and replicated graph database
  - Distributed adjacency list
- Distributed Network Cache (NCS)
  - LRU cache stores second degree network for active members
  - Graph traversals are converted to set intersections
- Graph APIs
  - Get Connections
  - Get Shared Connections
  - Get Graph Distances

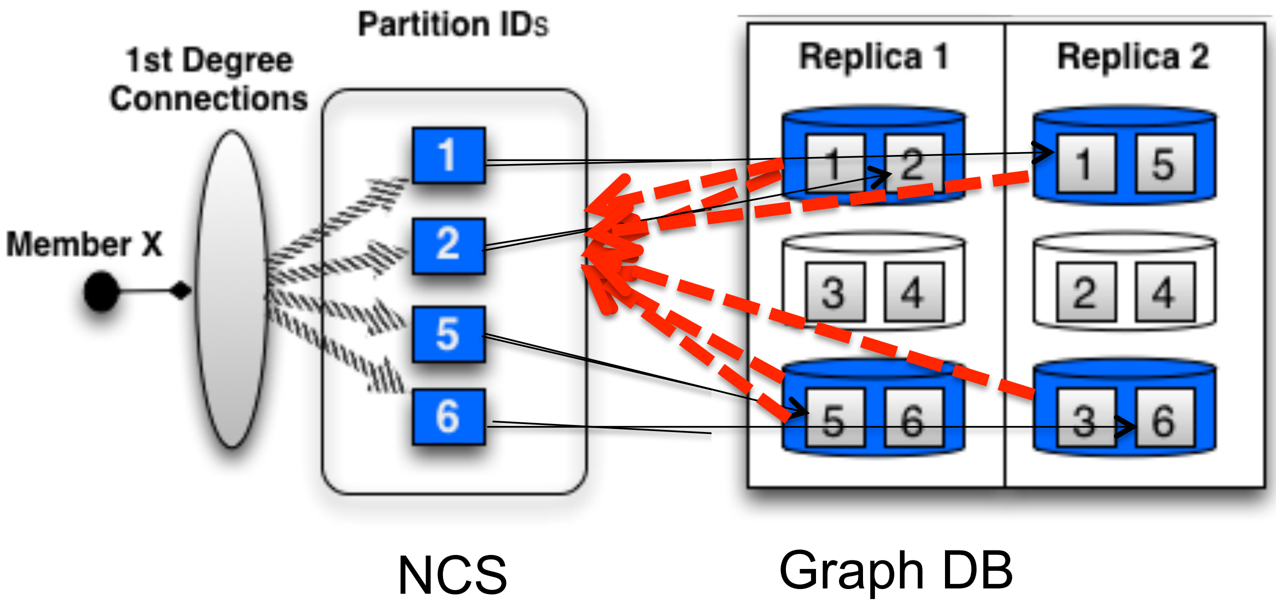
Outline

- LinkedIn’s Distributed Graph Infrastructure
- The Scaling Problem
- Set Cover by Example
- Evaluation
- Related Work
- Conclusions and Future Work

Graph Distance Queries and Second Degree Creation



The Scaling Problem Illustrated



- Increased Query Distribution
- Merging and De-duping shift to single NCS node

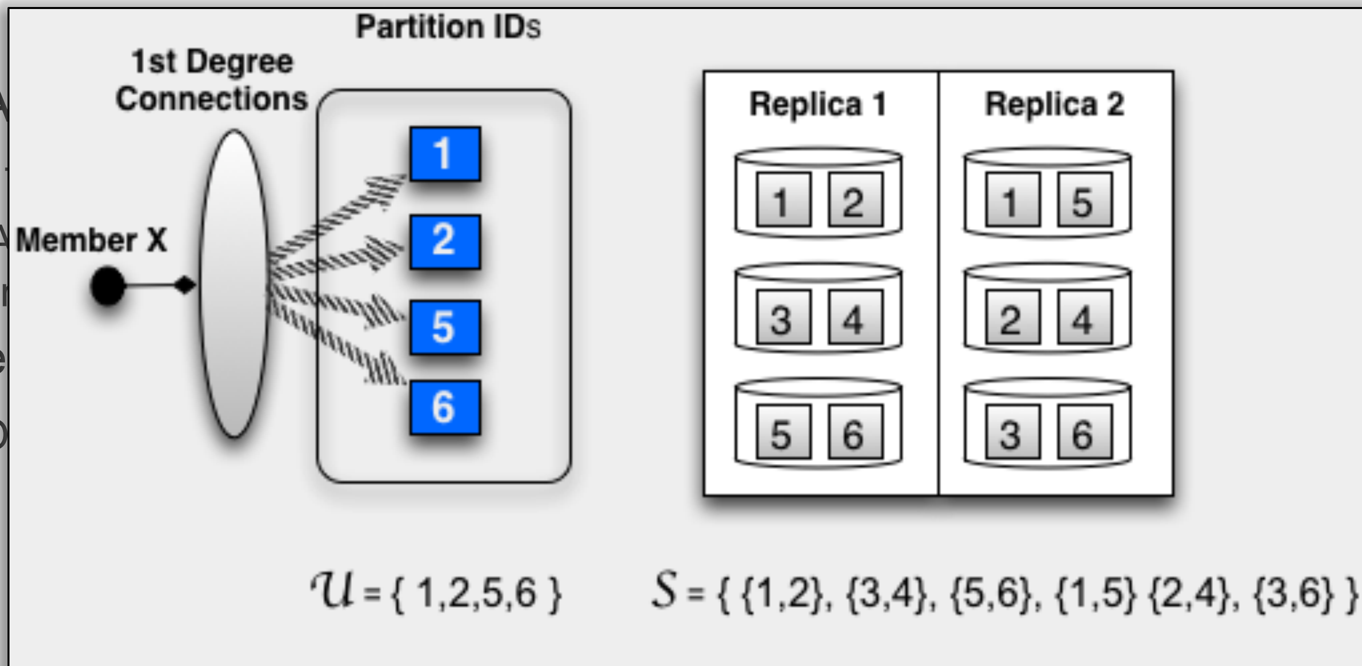
Outline

- LinkedIn's Distributed Graph Infrastructure
- The Scaling Problem
- Set Cover and Example
- Evaluation
- Related Work
- Conclusions and Future Work

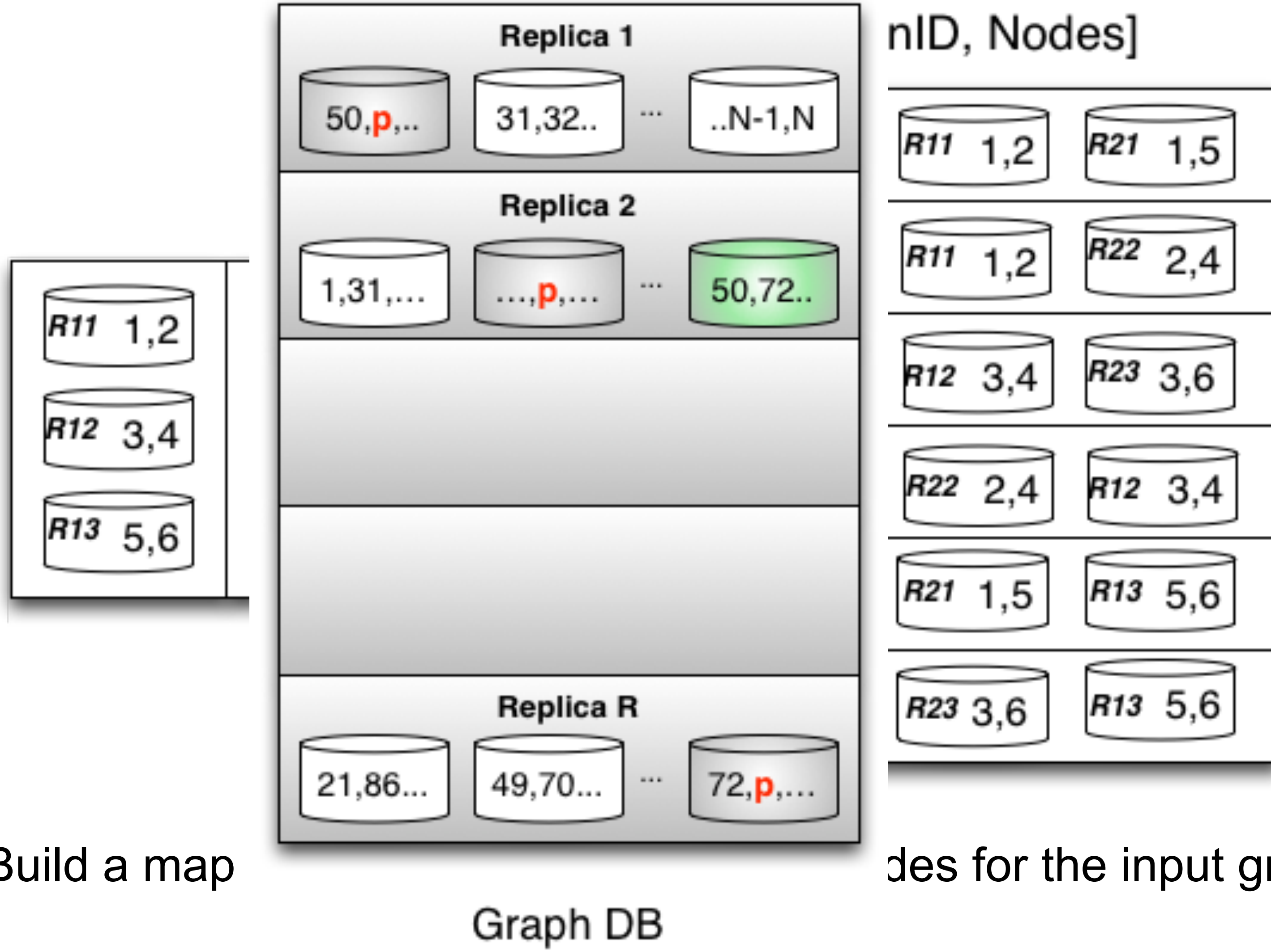
Set Cover Problem

- Definition
  - Given a set of elements $\mathcal{U} = \{1, 2, \ldots, m\}$ (called the universe) and a family $\mathcal{S}$ of n sets whose union equals the universe, the set cover problem is to identify the smallest subset of $\mathcal{S}$ the union of which contains all elements in the universe.

- Greedy A
  - Rule
    - A
      u
  - Uppe
    - O



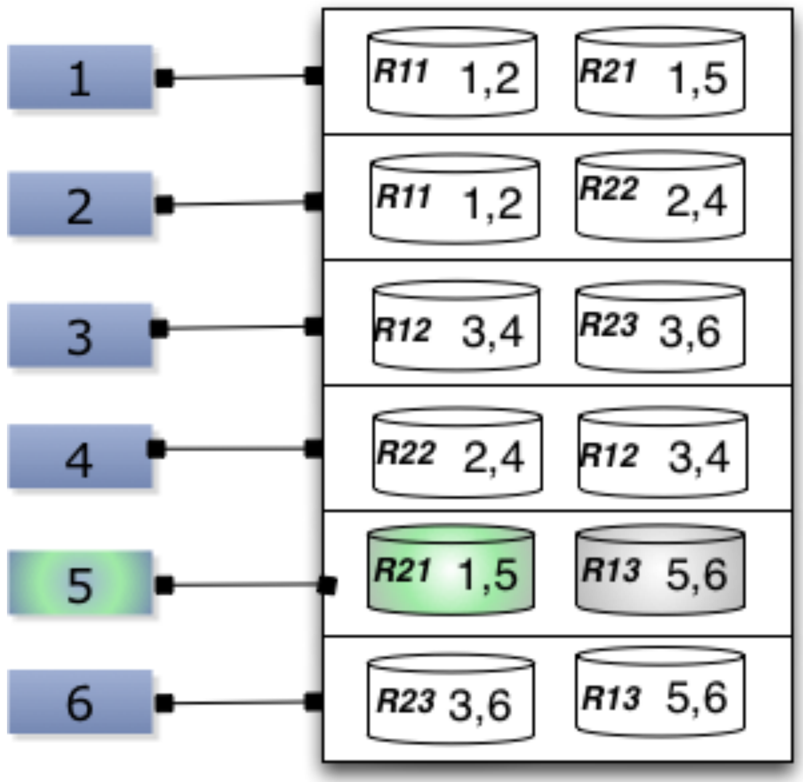
Modified Set Cover Algorithm Example Cont.



nID, Nodes]

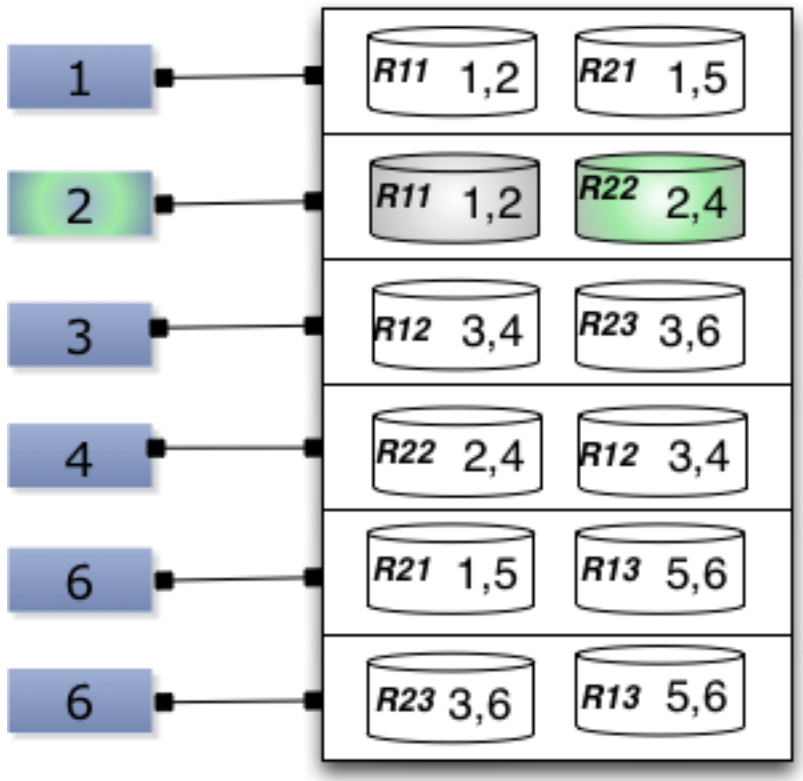
- Build a map des for the input graph

Modified Set Cover Algorithm Example Cont.



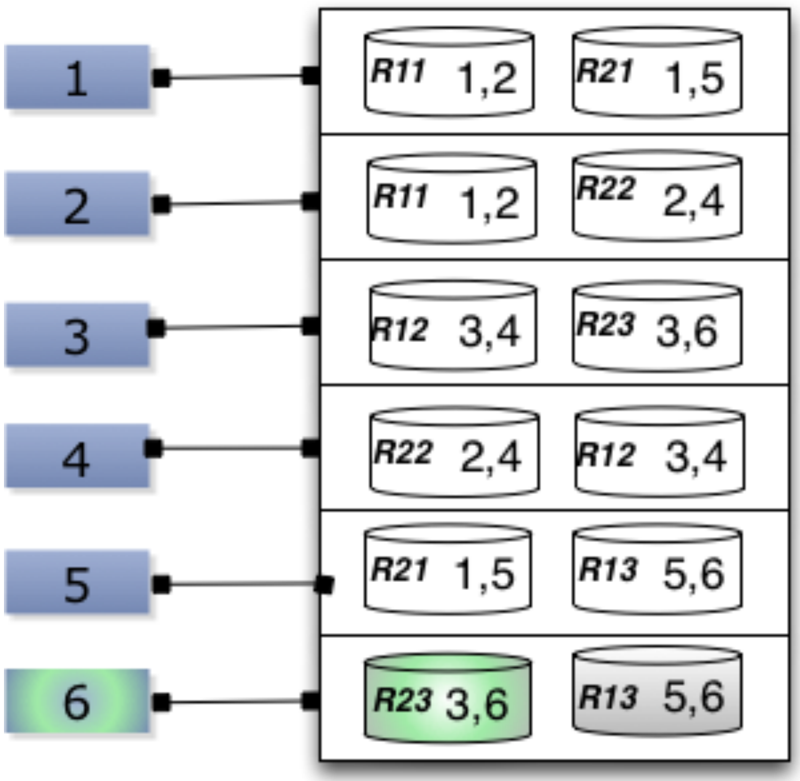
- Randomly pick an element from $\mathcal{U}$ = { 1,2,5,6 }
  - e = 5
- Retrieve from map
  - nodes = { R21, R13 }
- Intersect
  - R21 = {1,5 } with $\mathcal{U}$,
  - R13 = { 5,6 } with $\mathcal{U}$
- Select R21
  - $\mathcal{U}$ = {2,6}, $\mathcal{C}$ = { R21}

Modified Set Cover Algorithm Example Cont.



- Randomly pick an element from $\mathcal{U}$ = { 2,6 }
  - e = 2
- Retrieve from map
  - nodes = { R11, R22 }
- Intersect
  - R11 = {1,2 } with $\mathcal{U}$,
  - R22 = { 2,4 } with $\mathcal{U}$
- Select R22
  - $\mathcal{U}$ = {6}, $\mathcal{C}$ = { R21, R22 }

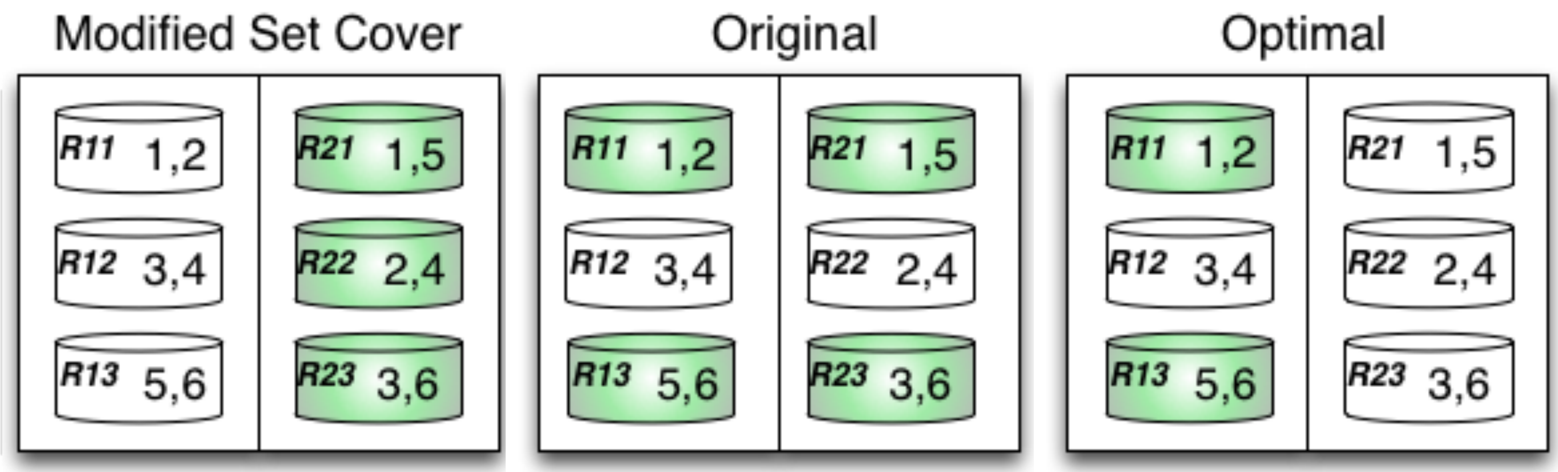
Modified Set Cover Algorithm Example Cont.



- Pick the final element from $\mathcal{U}$ = { 6 }
  - e = 6
- Retrieve from map
  - nodes = { R23, R13 }
- Intersect
  - R23 = { 3,6 } with $\mathcal{U}$,
  - R13 = { 5,6 } with $\mathcal{U}$
- Select R23
  - $\mathcal{U}$ = {}, $\mathcal{C}$ = {R21, R22, R23 }

Modified Set Cover Algorithm Example Solution

- Solution Compared to Optimal Result for $\mathcal{U}$, where $\mathcal{U} = \{1,2,5,6\}$

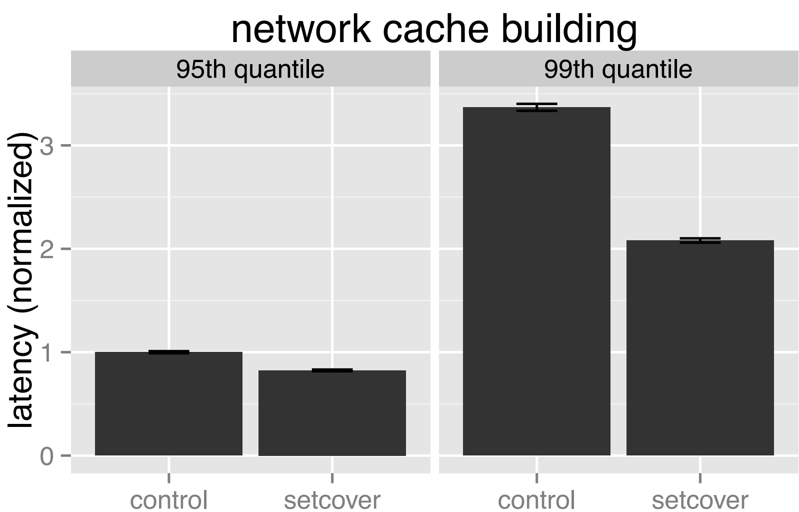


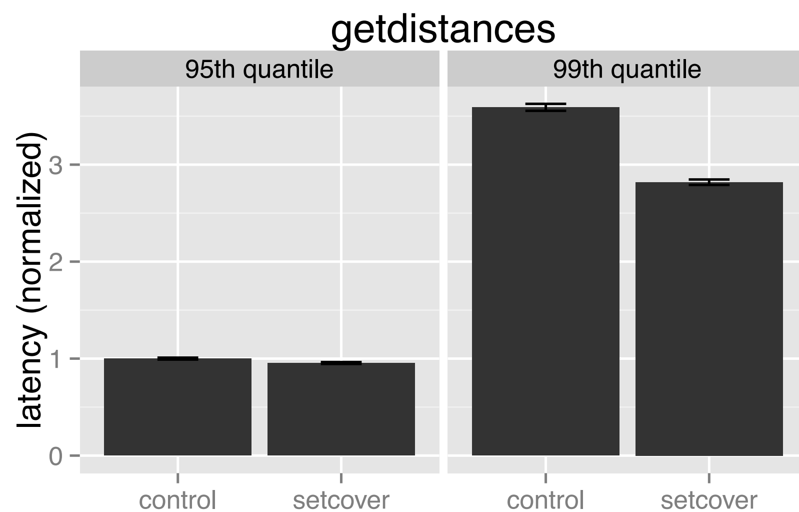
Outline

- LinkedIn's Distributed Graph Infrastructure
- The Scaling Problem
- Set Cover by Example
- Evaluation
- Related Work
- Conclusions and Future Work

Evaluation

- Production Results
  - Second degree cache creation drops 38% on 99th percentile
  - Graph distance calculation drops 25% on 99th percentile
  - Outbound traffic drops 40%; Inbound traffic drops 10%





Outline

- LinkedIn's Distributed Graph Infrastructure
- The Scaling Problem
- Set Cover by Example
- Evaluation
- Related Work
- Conclusions and Future Work

Related Work

- Scaling through Replications
  - Collocating neighbors [Pujol2010]
  - Replication based on read/write frequencies [Mondal2012]
  - Replication based on locality [Carrasco2011]
- Multi-Core Implementations
  - Parallel graph exploration [Hong2011]
- Offline Graph Systems
  - Google's Pregel [Malewicz2010]
  - Distributed GraphLab [Low2012]

Outline

- LinkedIn's Distributed Graph Infrastructure
- The Scaling Problem
- Set Cover by Example
- Evaluation
- Related Work
- Conclusions and Future Work

Conclusions and Future Work

- Future Work
  - Incremental cache updates
  - Replication on GraphDB nodes through LRU caching
  - New graph traversal algorithms
- Conclusions
  - Key challenges tackled
    - Work distribution balancing
    - Communication Bandwidth
  - Set cover optimized latency for distributed query by
    - Identifying a much smaller set of GraphDB nodes serving queries
    - Shifting workload to GraphDB to utilize parallel powers
    - Alleviating the K-way merging costs on NCS by reducing K
  - Available at

    https://github.com/linkedin-sna/norbert/blob/master/network/src/main/scala/com/linkedin/norbert/network/partitioned/loadbalancer/DefaultPartitionedLoadBalancerFactory.scala