CS-848 - February 26, 2019

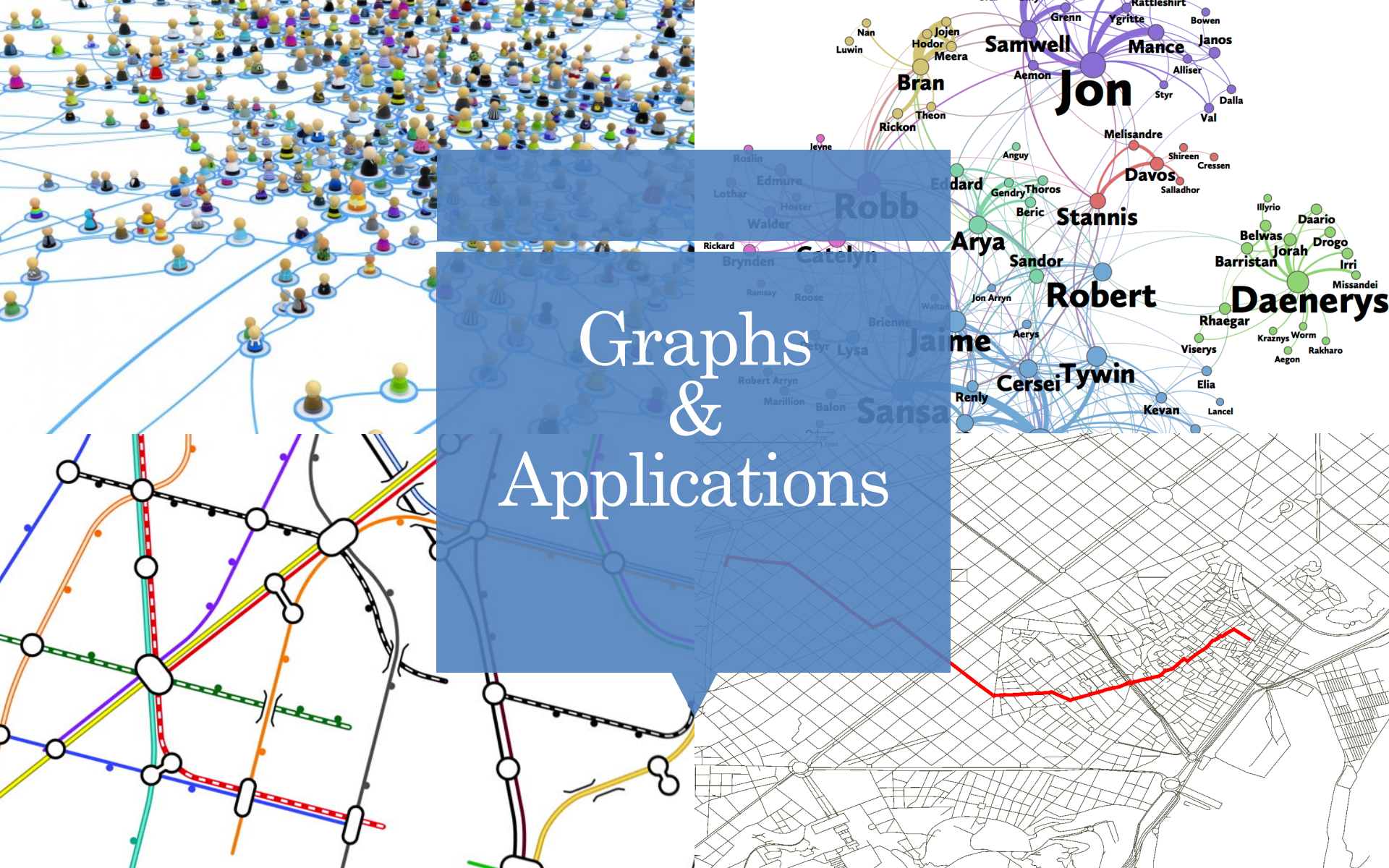
Trinity: A Distributed Graph Engine on a Memory Cloud

By Shao et al.

Presented by Ruoxi Zhang

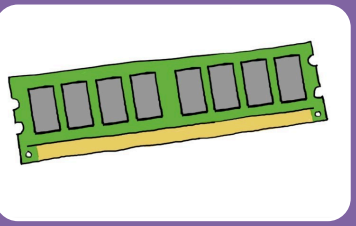
Reference

- B. Shao, H. Wang, Y. Li. Trinity: a distributed graph engine on a memory cloud, Proc. ACM SIGMOD International Conference on Management of Data, pages 505-516, 2013.


Graphs
&
Applications
Samwell
Mance
Jon
Bran
Robb
Arya
Sandor
Robert
Stannis
Davos
Melisandre
Daenerys
Cersei
Tywin
Kevan
Nan
Luwin
Jojen
Hodor
Meera
Grenn
Ygritte
Rattleshirt
Bowen
Janos
Alliser
Aemon
Styr
Dalla
Val
Theon
Rickon
Shireen
Cressen
Salladhor
Anguy
Gendry
Thoros
Beric
Illyrio
Daario
Belwas
Jorah
Drogo
Barristan
Irri
Missandei
Rhaegar
Kraznys
Worm
Rakharo
Aegon
Viserys
Elia
Lancel
Renly
Aerys
Jon Arryn
Rickard
Roslin
Edmure
Lothar
Hoster
Walder
Brynden
Catelyn
Ramsay
Roose
Brienne
Lysa
Robert Arryn
Marillion
Balon
Sansa

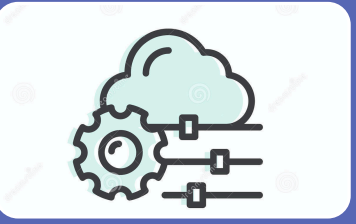
Online Query Processing	Offline Graph Analytics
Requires low latency	Requires high throughput
Graph exploration Random access	Iterative, batch processing Parallel computing
e.g., BFS, subgraph matching, etc.	e.g., PageRank, betweenness computation, etc.

Why Trinity?



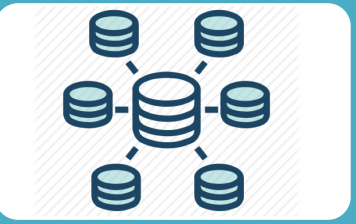
Characteristics of graph computation

- Fast random data access



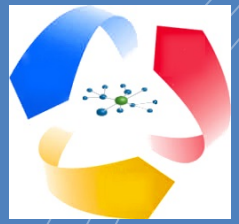
Improve performance

- Keeping data in main memory



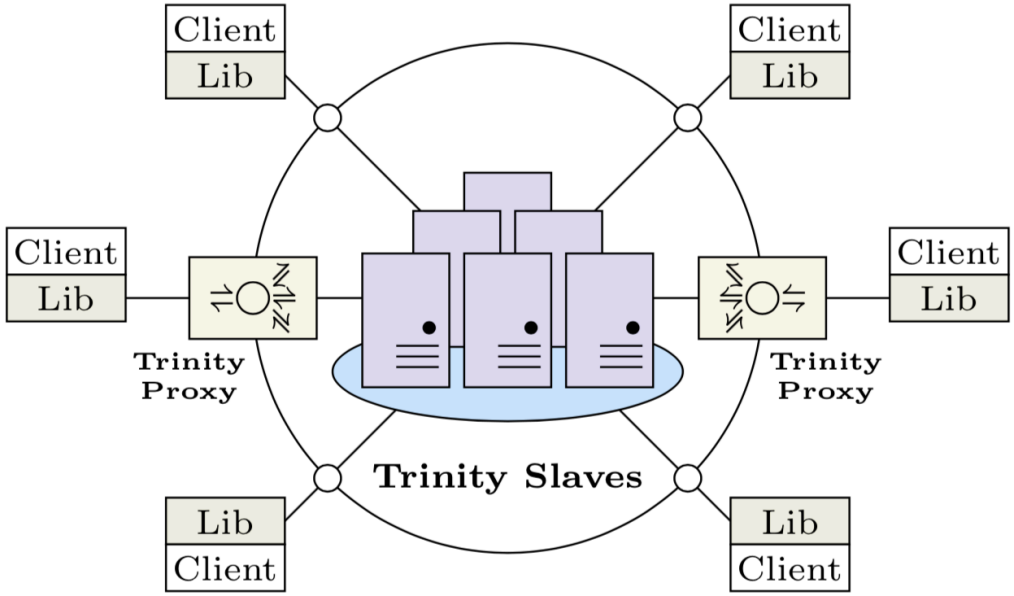
The scale of data

- Distributed parallel computation



Trinity

- Trinity is a general purpose graph engine over a distributed memory cloud.
- **Online** query processing + **offline** graph analytics
- The belief of **all-in-memory** solutions
  - High-speed network
  - Low price DRAM
- No comprehensive built-in graph computation modules
  - Flexible data
  - Computation modeling capability

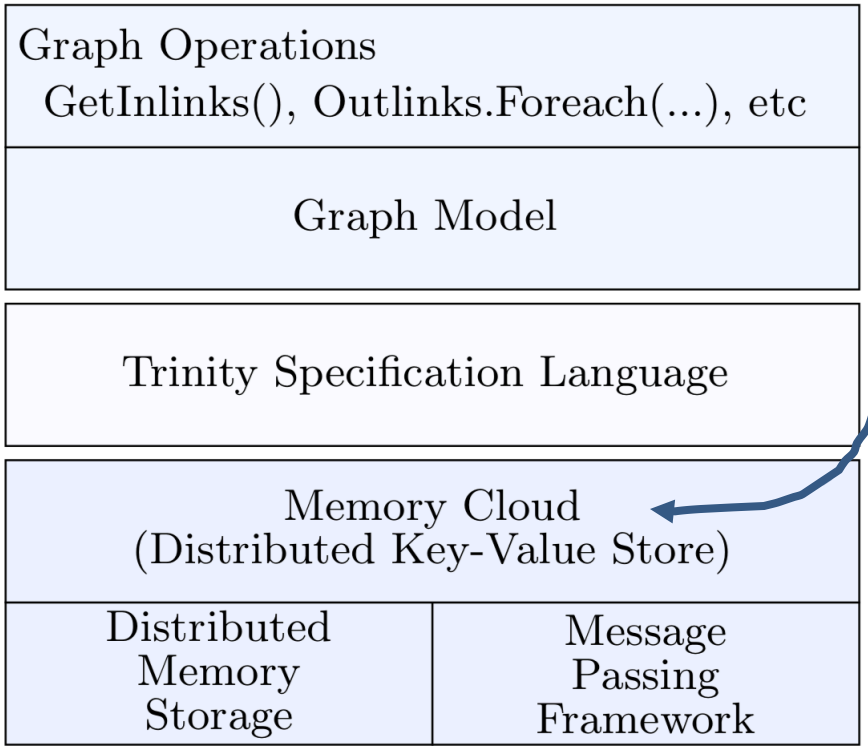


Trinity Cluster

Slaves **Proxies** **Clients**

Distributed Memory Cloud

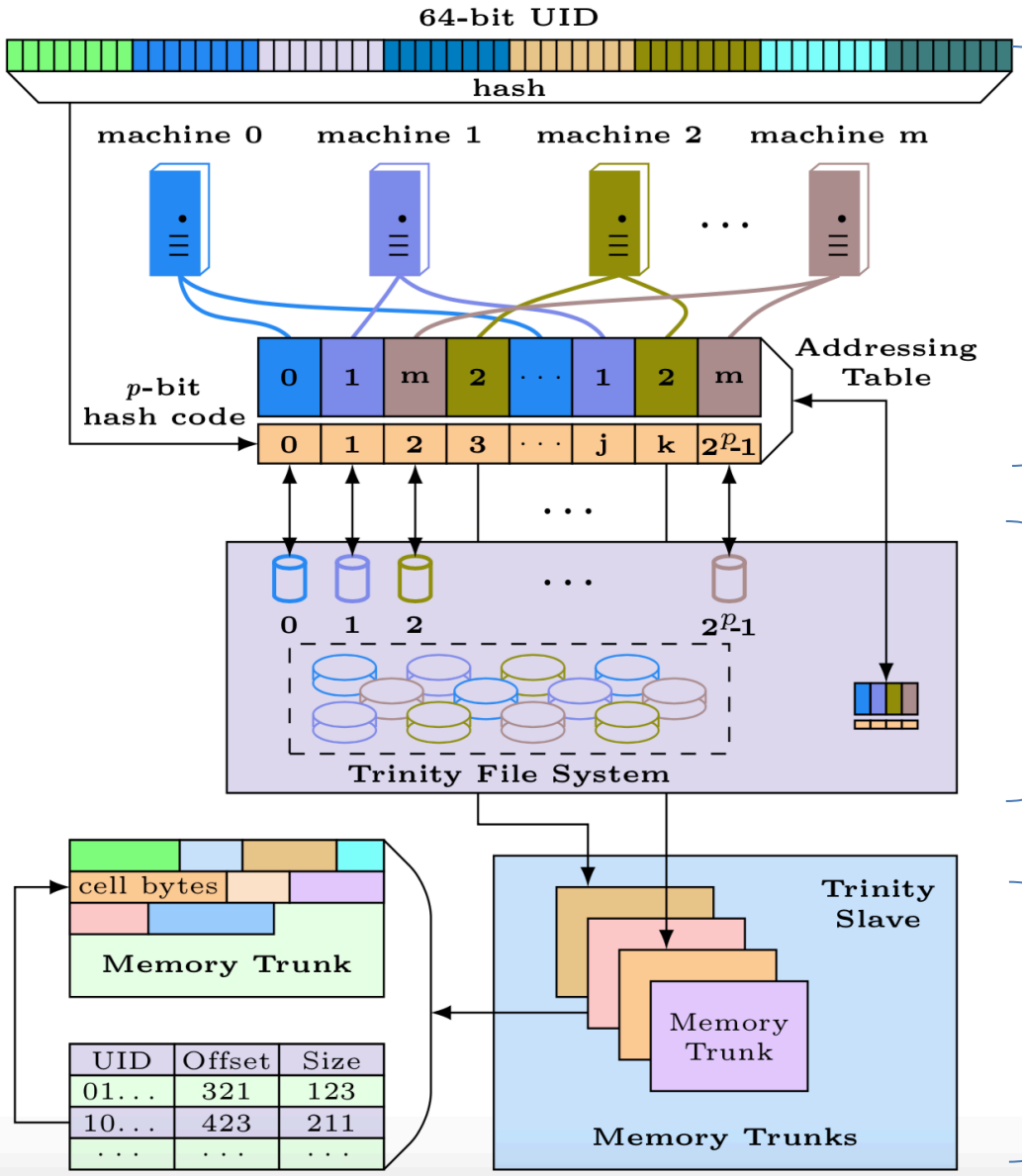
Trinity System Layers



- Partition memory space into 2^p (> m) trunks.
  - Trunk level parallelism
  - Efficient hashing
- Basic data structure: key-value pairs
  - **Key**: 64-bit globally unique identifiers (UID)
  - **Value**: blobs of arbitrary length
  - **Metadata**: spinlock

memory cloud → fast random access → fast graph exploration

Partitioning & Addressing

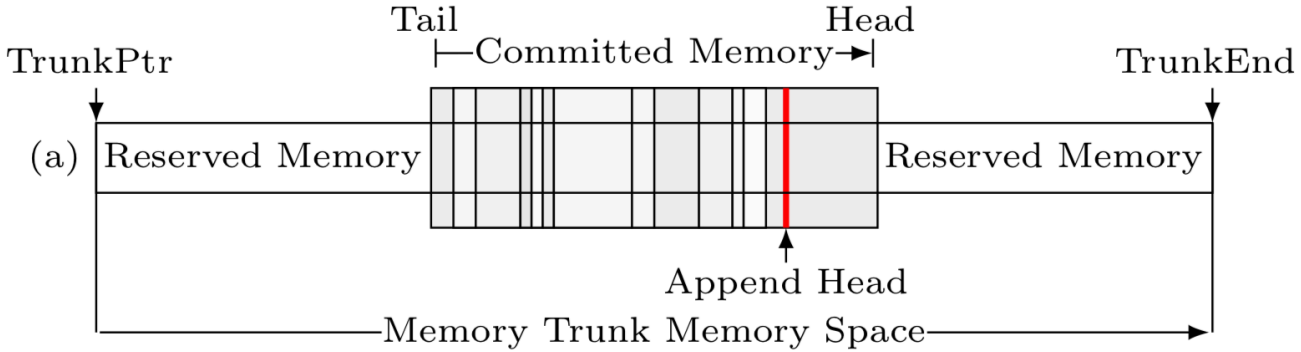


- [Scalable] Hashing mechanism
- Trunk number to machine ID
  - UID → p bit $i \in [0, 2^p - 1]$
  - Addressing table

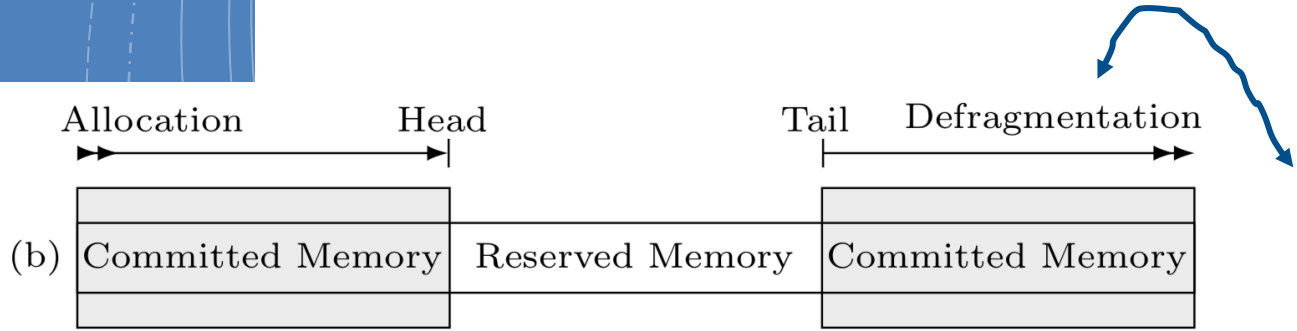
- [Fault-tolerant] memory trunks are backed up in TFS (Trinity File System).

- Hash again to find the value
  - Use the hash table in trunk
  - UID → offset & size

Circular Memory Management

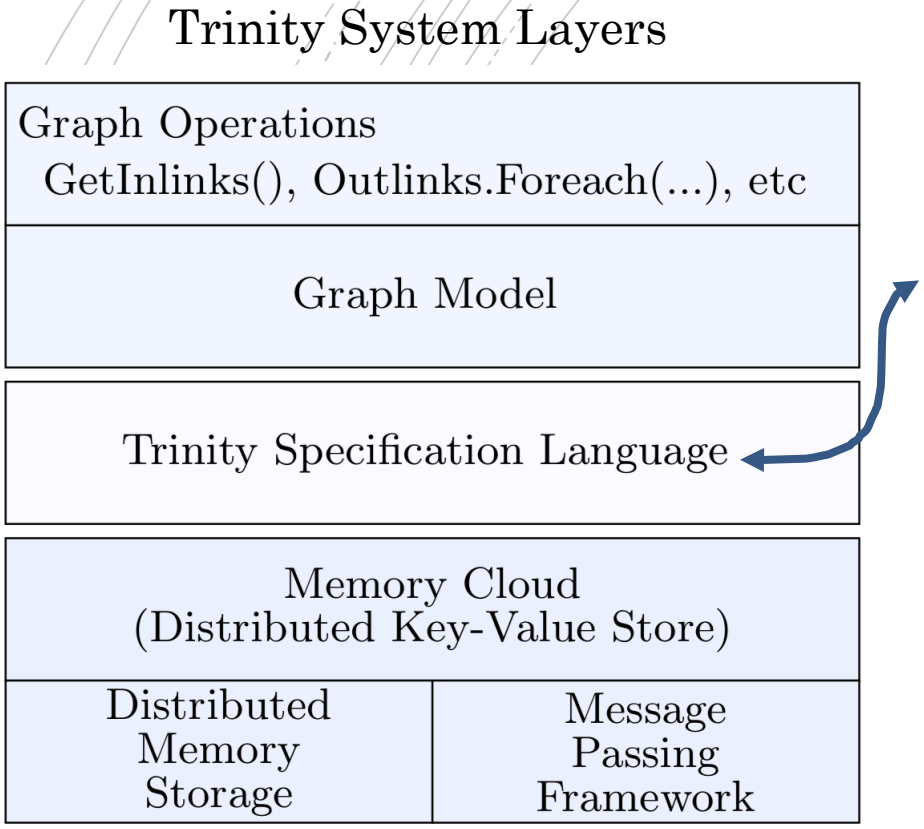


- Given: a large number of key-value pairs; size may increase or decrease
- Goal: avoid memory gaps



- Append head
- **Defragmentation** daemon

Data Model



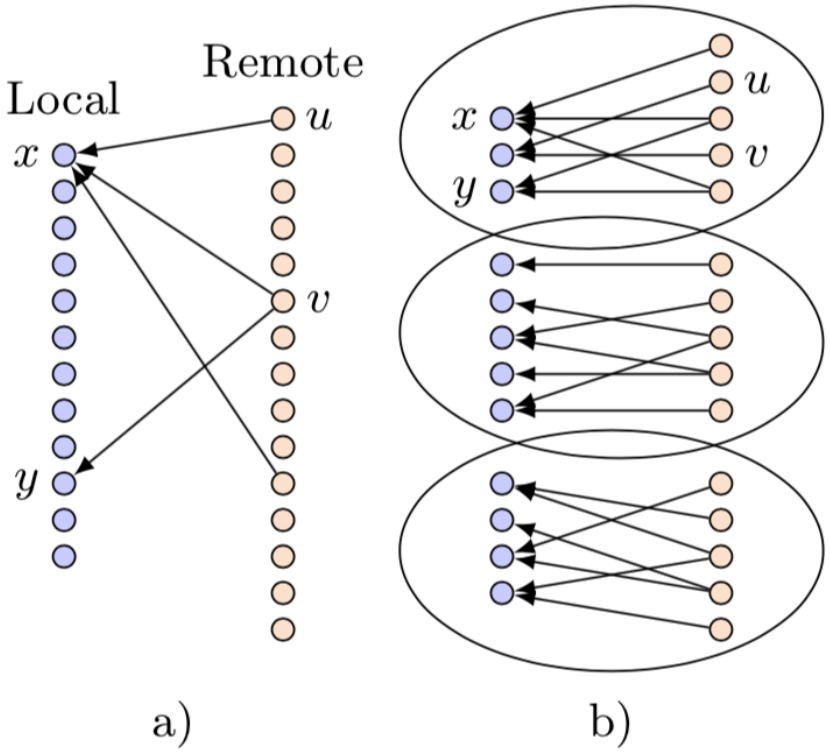
- **Cell** = value + schema
  - a node (or a rich edge)
- TSL
  - **Object-oriented** cell manipulation
    - Cell accessor
  - Data integration
    - Transparent query processing
    - Automatic data conversion
  - System extension
    - Models cell schema
    - Models message passing

Computation Paradigms (1)

- **Online** queries processing (e.g., traversal & subgraph matching)
  - Fast random access & parallel computing, no index
- **Offline** graph analytics (e.g., PageRank)
  - Restricted vertex-centric computation model
  - Aggregate local answers & probabilistic inference
- Higher performance and lower memory usage

PBGL, a parallel graph processing library	BFS
Giraph, offline system	PageRank

Computation Paradigms (2)



- Message passing optimization
  - Create a bipartite partition of the local graph
  - Buffer messages from **hub vertices**
  - Obtain messages from vertices in other partitions on demand
  - Given a data access pattern:

Vertices in a partition currently scheduled to run

UID
neighbors
attributes
local variables
message

Other vertices

UID
message

Fault Tolerance

- Maintain a primary replica of the shared addressing table on a leader machine
- Heartbeat messages to detect machine failures
- Fault recovery varies by computation models
  - BSP (batch synchronous processing): checkpoints
  - Asynchronous: periodical interruption

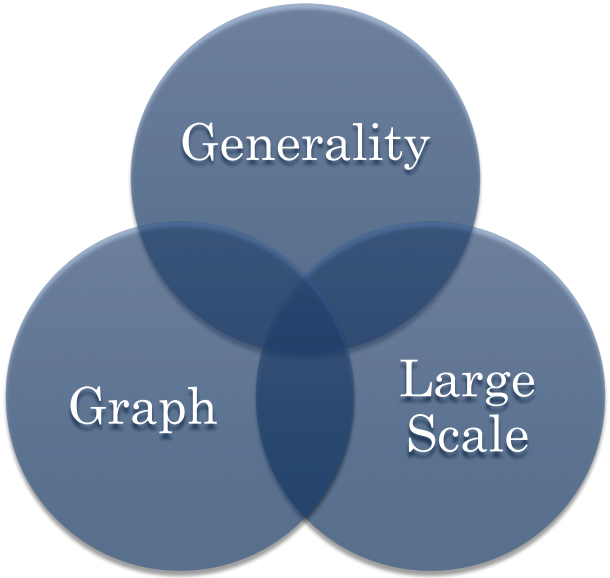
Partition

- Divide a graph into many equal size parts, such that the number of edges among them is minimized.
- Goal: load balance + reduce communication overhead

- Billion-node graph partitioning is an unsolved problem on general-purpose graph platforms.

- Multi-level partitioning algorithm

Summary



- **General purpose** – algorithms & graphs & computation models
- **Large scale** – billions of nodes
- **Distributed** – instead of storing it centrally on a single machine
- **Memory-based** – keep the graph in memory, at least the topology

Trinity is a … graph engine.

Related Publications

- Zhao Sun, Hongzhi Wang, Bin Shao, Haixun Wang, and Jianzhong Li, **Efficient Subgraph Matching** on Billion Node Graphs, VLDB 2012.

- Kai Zeng, Jiacheng Yang, Haixun Wang, Bin Shao, and Zhongyuan Wang, A Distributed Graph Engine for **Web Scale RDF Data**, VLDB 2013.

- Wanyun Cui, Yanghua Xiao, Haixun Wang, Ji Hong, and Wei Wang, **Local Search of Communities** in Large Graphs, SIGMOD 2013.

Microsoft® Research Asia

Discussion

- What is unique about in-memory engine design compared to disk-based engines?
- Why wouldn't regular disk-based databases adopt the same techniques?
- In reality, Trinity does not provide ACID transaction support. If it is added as a feature, what is the trade-off?
- What are the advantages and disadvantages of a general-purpose system?