# A System For Navigational Queries

Guided by Dr. Gunturi

Presented By
Yugandhar Bandi
Sainath Thota

# Basic Concepts:

- Spatial Networks
- Spatio Temporal Networks
- Navigational Queries
- WCRR - The chance that a pair of connected nodes that are more likely to be accessed together are allocated to a common page of the file

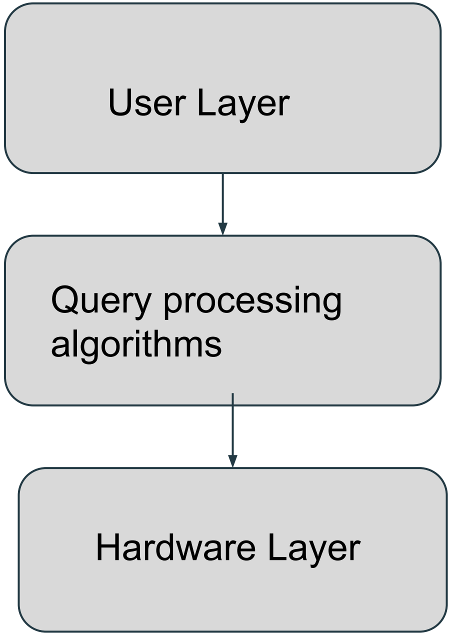
# Problem Statement :

- Given a Real time Spatio Temporal Network ⇨ A System which efficiently implements frequent Navigational Queries

# Motivation

- Google and Bing Maps use a lot of heuristics and never publish any sort of correctness and completeness guarantees for their systems.
- Exploring a system which can guarantee some level of correctness and completeness.
- Google maps only provides APIs for others to use but not its implementation.
- Open sourcing an efficient implementation.

# Rough Design :



# Challenges :

- Efficient implementation of algorithms to compute frequent navigational queries
- Need an efficient design of storage and access methods for network data.
- It is not clear if existing spatial access methods can efficiently support network computations in Spatio Temporal Networks

# Possible Solution for Challenge 1

CCAM : A Connectivity - Clustered Access Method for Networks and Network Computations

# What is CCAM ?

- It is a new access method to efficiently support aggregate queries over general networks.
- It supports the operations like Create, Insert, Delete, Find, Get-A-Successor, Get-successors.
- CCAM clusters the nodes of the network via graph partitioning using ratio-cut heuristic.
- The crux of CCAM is to maintain high WCRR.

# Operations Required:

1) Create: <list of node records> - Network

2) Find: <node-id, Network>  - node properties

3) Insert: <node-id, node-properties, Network> - Network

   Insert: <edge, edge-properties, Network> - Network

4) Delete: <node-id, Network> - Network

Delete: <edge, edge-properties, Network> - Network

5) Get-successors: <node-id, Network> - list of <node-id, node-properties> of successors

6) Get-A-successor: <node-id, successor-id, Network> - node-properties of the successor

## CCAM: A Connectivity-Clustered Access Method for Networks and Network Computations [Shekhar et al, IEEE TKDE'97]

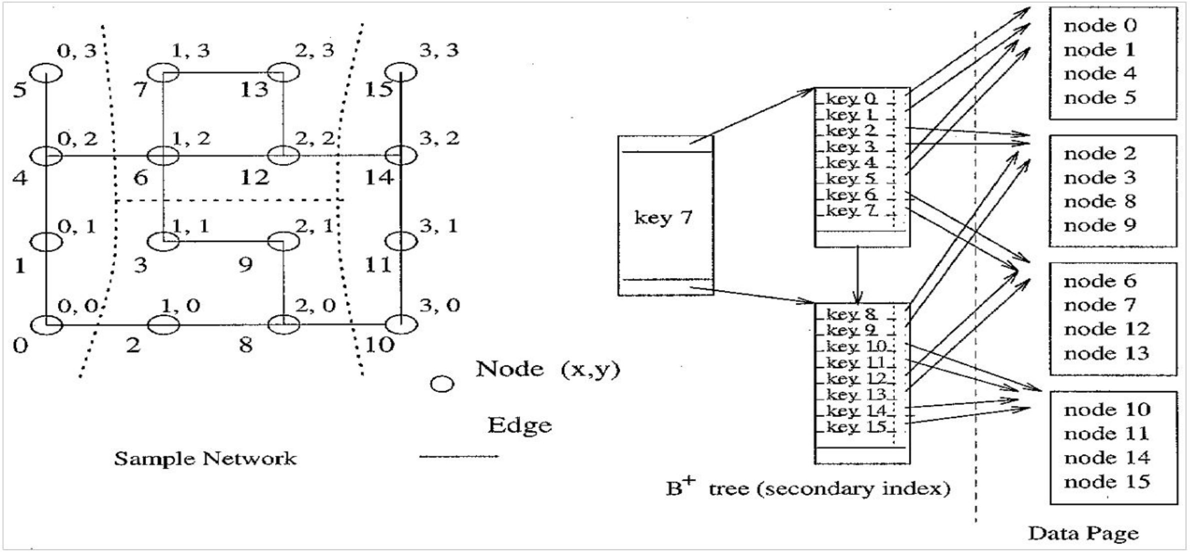
- The expected I/O cost for many network operations can be reduced by maximizing the Weighted Connectivity Residue Ratio (WCRR).

$$\text{WCRR} = \frac{\text{Sum of } w(u, v) \text{ such that } Page(u) = Page(v)}{\text{Total sum of weights over all edges}}$$

The weight w(u, v) associated with edge(u, v) represents the absolute frequency of a query accessing nodes u and v together

# Continued..

- Node-id is an integer representing the Z-order of (x,y) coordinates
- Since it is not practical to store the data in sorted format, Secondary index is created to quickly locate data without having to search for every row.



# Create(): Creation of CCAM :

- The nodes of the network are clustered via the cluster-nodes-into-pages() algorithm, which returns a set of pages.
- The nodes (records) which belong to the same subset are stored on the same data page, and an index entry for each node is created and inserted into the B+tree.
- Each subset contains at least min-page-size bytes.
- two-way-partition() is repeatedly applied to cluster the graph until all subset sizes are less than the page-size.

# Algorithm :

```
Procedure: cluster-nodes-into-pages
        (V: set of nodes, E: set of edges, page-size): return set of pages
        F, P: set of page(partition) of nodes;
        V', A, A': set of nodes;
        begin
                Initialise F={V}; P={}; MinPgSize=ceil(page-size/2);
                while F is not empty do
                        Choose a V' from F, E'={(u, v)|(u, v) belong to E, u belongs to V' and v belongs to V'};
                        Remove V' from F;
                        <A,A'>=2-way-partition(V',E',MinPgSize);
                        if sizeof(A)>page-size then add A to F;
                        else add A to P;
                        if sizeof(A')>page-size then add A' to F;
                        else add A' to P;
                endwhile
                return P;
        end;
```

# The Incremental Create() Operation :

- The Static-Create() operation is not efficient when the entire network does not fit inside main memory.
- The incremental Create() operation is implemented as a sequence of Add-node() operations, which are similar to the Insert() operations
- The Add-node() operation does not need to update the successor and predecessor lists, since the node records initially presented to create a file can be preprocessed to have the proper values for the predecessor-list and successor-list.

# Efficient Support of Search Operations

- Find(record x): Retrieve the Record of a Given Node-id
- Get-A-successor(node n1, node n2): Retrieve a Specified Successor of a Given Node
- Get-successors(node n1): Retrieve Records for the Successor Nodes of a Given Node

# Maintenance and Dynamic Reclustering Strategies

DEFINITION 1 (Page Access Graph). *Let $G = (V, E)$ be the given network. P is called a page of G if and only if P is a set of records, such that for each record(x) $\in P$, $x \in V$, and all records $\in P$ are stored in the same disk data page, i.e.,, the total size of the records included in P is at most full disk page size. Let each of $P_1, P_2, \cdots, P_n$ be a page of G. Then the page access graph (PAG) $G_p = (V_p, E_p)$, where $V_p$ is a set of pages and $E_p$ is a set of edges, defined as follows:*

$$V_p = \{P_1, P_2, \cdots, P_n\},$$
$$E_p = \{(P_i, P_j) \mid \exists\, x, y$$

*such that $x \in V$, $y \in V$, $(x, y) \in E$, record$(x) \in P_i$, and record$(y) \in P_j\}$.*

DEFINITION 2.

- *Is-Neighbor-Page*$(P, Q)$ = *true iff either* $(P, Q) \in E_p$ (
$(Q, P) \in E_p$.
- *NbrPages*$(P \in V_p) = \{Q \mid Q \in V_p$ *and Is-Neighbo*
*Page*$(P, Q)\}$.
- *Page*$(x \in V) = Q$, *where* $Q \in V_p$ *and record*$(x) \in Q$.
- *PagesOfNbrs*$(x \in V) = \{Page(u) | u \in succ(x) \cup pred(x)$

# Re-Organization Policy :

SET OF PAGES REORGANIZED BY DIFFERENT POLICIES FOR MAINTENANCE

| Reorganization Policy | Set of Pages to be Reorganized | | |
|---|---|---|---|
| | argument = edge(u,v) | argument = node x | Guiding Principle |
| **First order** | none<br>handle underflow/overflow | none<br>handle underflow/overflow | avoid or delay reorganization |
| **Second order** | {Page(u), Page(v)} | {Page(x)} ∪ PagesOfNbrs(x) | reorganize pages which must be updated anyhow |
| **Higher order** | 1. NbrPages(Page(u))∪{Page(u)} ∪NbrPages(Page(v))∪{Page(v)}<br>or<br>2. {Page(u)}∪PagesOfNbrs(u)∪ {Page(v)}∪PagesOfNbrs(v)<br>or<br>3. all pages in data file | 1. {Page(x)} ∪ PagesOfNbrs(x) ∪ NbrPages(Page(x))<br>or<br>2. all pages in data file | reorganize more pages than second order policy |

Page(x) = page selected to place x in Insert() or page containing x in Delete()

# Insert(): Insert a New Node or Edge

- During the insertion of a new node x, a data page must be selected in which to store the new node
- Page selection may be accomplished by ranking the pages by the total weight on the edges to the neighbors of x located in the page
- If there is an overflow in the neighbouring pages, cluster them using the previous algorithm

# Algorithm :

```
Procedure: Insert(x: node-id; record(x): node-properties;
           policy: reorganization-policy)
begin
 retrieve PagesOfNbrs(x);
 if PagesOfNbrs(x) is empty then
    insert record(x) into an available disk page P;
    insert index entry (node-id x, disk address of P);
 Otherwise,
    update succ-list and pred-list of neighbors(x);
    select a page P from PagesOfNbrs(x) to put record(x);
    if (policy == first-order policy) then
       for each page Q in PagesOfNbrs(x) do
          if Q overflow then split Q into two pages
          else if Q has been modified then Write Q;
    else
       Reorganize(x, policy);
end;
```

# Delete(): Delete a Node or Edge

- The data page P that stores the record(x) to be deleted can be retrieved by using the node-id value of node x.
- If the deletion makes the page underflow, two data pages might be merged to increase data-page utilization.
- We can simply choose a neighboring page Q of P from PagesOfNbrs(x) to be merged with P
- If Q and P cannot be merged into one page, they are distributed between the two pages, using the cluster-nodes-into-pages() procedure.
- The selection of page Q may be accomplished by ranking the pages by the total weight on the edges that cross page P and the number of data-bytes in the page.

# Algorithm :

```
Procedure: Delete(x: node-id; policy: reorganize-policy)
begin
  retrieve P = Page(x); retrieve PagesOfNbrs(x);
  update succ-list and pred-list of neighbors(x);
  delete x from P; delete index entry of x;
  if (policy == first-order) then
     if page P underflow then
        select a page Q from PagesOfNbrs(x);
        perform data page merging on {P, Q};
     for each page Q in {PagesOfNbrs(x), P} do
        if Q has been modified then Write Q;
  else
     Reorganize(x, policy);
end;
```

# Is the WCRR the Right Metric?

THEOREM 1. *The expected cost of network operations (e.g., Get-A-successor()) is minimized by maximizing the Weighted Connectivity Residue Ratio (WCRR).*

PROOF. Given a graph $G = (N, E)$ and the edge cut-set $E_C$, let an *unsplit* edge (u, v) be characterized by page(u) = page(v). Let the unsplit edge set denoted by $E_R$ be $E - E_C$. The cost of accessing the pair of nodes connected by edge $(u, v) \in E$, $c(u, v)$, is defined by

$$c(u, v) = \begin{cases} \sigma & \text{if } page(u) = page(v), \\ \tau & \text{if } page(u) \neq page(v),\ \tau \geq \sigma. \end{cases} \quad (1)$$

# Proof Contd :

- Let the weight on edge(u,v), denoted by w(u,v), represent the absolute frequency of network operations that access the pair of nodes connected by edge(u,v).

Let $g(u, v)$ be equal to $\frac{w(u,v)}{\sum_{(u,v)\in E} w(u,v)}$

Then g(u,v) is the Probability[pair of nodes connected by edge(u,v) used in network operations|(u,v) $\in$ E]

$$\Theta = \sum_{(u,v)\in E_C} c(u, v) \cdot g(u\ v) + \sum_{(u,v)\in E_R} c(u, v) \cdot g(u, v)$$
$$= \tau \cdot \sum_{(u,v)\in E} g(u, v) - (\tau - \sigma) \cdot \sum_{(u,v)\in E_R} g(u, v)$$

From the fact that $\sum_{(u,v)\in E} g(u, v) = 1$, we can further derive the following equation:

$$\Theta = \tau - (\tau - \sigma) \cdot \sum_{(u,v)\in E_R} g(u, v) \qquad (2)$$

$$\Theta = \tau - (\tau - \sigma) \cdot WCRR,$$

$$where\ WCRR = \sum_{(u,v)\in E_R} g(u,\ v) = \frac{\sum_{(u,v)\in E_R} w(u,v)}{\sum_{(u,v)\in E} w(u,v)}$$

Thus, maximizing the WCRR minimizes θ, the expected cost of accessing an edge.

# Application of CCAM to Spatio-Temporal Networks

- We can incorporate Temporal Data as node properties
- This may decrease the number of records in each page

# References :

S. Shekhar and Duen-Ren Liu, "CCAM: a connectivity-clustered access method for networks and network computations," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, no. 1, pp. 102-119, Jan.-Feb. 1997

C. -. Cheng and Y. -. A. Wei, "An improved two-way partitioning algorithm with stable performance (VLSI)," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 12, pp. 1502-1511, Dec. 1991

C. -. Cheng and Chung-Kaun Cheng,"Ratio cut partitioning for hierarchical designs" in *IEEE Transactions on Knowledge and Data Engineering*, vol.10, no. 7, July 1991

# Thank you