Dynamic Graph Query Primitives for SDN-based Cloud Network Management

Ramyaa Raghavendra, Jorge Lobo, Kang-Won Lee
IBM T. J. Watson Research Center
Network as a Service Model

- Provide rich set of services
  - isolation, custom addressing, service differentiation etc.

- Interact with variety of network devices
  - support multiple cloud platforms, device management protocols

- Introduces a network controller
  - network-aware VM placement, QoS support, real-time monitoring, diagnostics, management, security etc.
Graph Queries for Network Management

- Support for efficient queries on network graph, e.g., shortest path between two nodes
- Utilized in various network management operations

Network graphs can represent:
- physical network elements such as routers, switches, and servers
- virtual elements such as VMs and virtual switches
- logical elements such as people, processes, web pages, etc.
- and links between them

Algorithm support:
- shortest path
- spanning tree
- min flow
- …

NetGraph: shared library for graph primitives
Cloud DCNs may experience frequent changes
- Deployment of new VMs, removal of old ones
- Migration of VMs
- Layer 2/3 network reconfiguration

Graph updates
- edge/vertex insertion & deletion
- edge weight change (e.g. congestion level)

Customer: this is what my workloads look like

High-level VM & network description

Cloud controller

Network controller

OpenFlow

Monitoring, notifications

High-level network description & commands
Interaction of NetGraph with Network Controller

Cloud Controller

SDN-enabled Network Controller

Service Modules
- n/w aware placement
- Load balancing
- Broadcast
- Isolation
- Routing
- Path computation
- QoS
- Monitoring
- Topology

Physical and virtual network infrastructure

NetGraph
- shortest path
- spanning tree
- reachability
- bipartite matching

Graph queries

Physical & virtual network graph, updates
# All Pair Shortest Path as Graph Query Example

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Technique</th>
<th>Running time</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dijsktra’s algorithm</td>
<td>Static algorithm. Shortest paths are recomputed on update</td>
<td>HIGH</td>
<td>HIGH</td>
</tr>
<tr>
<td>Dynamic shortest path algorithms*</td>
<td>Maintain subgraphs and recompute queries on affected subgraph</td>
<td>LOW</td>
<td>HIGH</td>
</tr>
<tr>
<td>Index-based shortest path algorithm</td>
<td>Maintain tree-decomposition based index</td>
<td>LOW</td>
<td>LOW</td>
</tr>
<tr>
<td>(our approach§)</td>
<td>Update tree nodes that are affected by update</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* G. Ramalingam and T. Reps. An incremental algorithm for a generalization of the shortest-path problem

§ Extensions to F. Wei. TEDI: efficient shortest path query answering on graphs published in SIGMOD 2010
Index-based shortest path computation (our approach)

Utilize tree decomposition-based shortest paths method:

• Shortest path search can be executed in a bottom-up manner
• Query time is decided by the height and the bag cardinality of the tree, instead of the size of the graph
• Updates are restricted to the bags containing the edge and bags affected by edge update
## Interfaces

<table>
<thead>
<tr>
<th>Maintain network graph</th>
<th>Compute graph query</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Network topology: nodes and links</td>
<td></td>
</tr>
<tr>
<td>• Graph class with Vertices and Edges</td>
<td></td>
</tr>
<tr>
<td>• Topology updates</td>
<td></td>
</tr>
<tr>
<td>(node1, node2, edgeweight)</td>
<td></td>
</tr>
<tr>
<td>• Node and edge computations</td>
<td></td>
</tr>
<tr>
<td>CountDegree, CountNeighbors, etc.</td>
<td></td>
</tr>
<tr>
<td>• Algorithmic functionalities</td>
<td></td>
</tr>
<tr>
<td>ComputeMST(S), IsSubPath(P1,P2)</td>
<td></td>
</tr>
<tr>
<td>• Shortest paths</td>
<td></td>
</tr>
<tr>
<td>• ComputeAPSP</td>
<td></td>
</tr>
<tr>
<td>• ComputeSSSP(S)</td>
<td></td>
</tr>
<tr>
<td>• ...</td>
<td></td>
</tr>
</tbody>
</table>
Experimental Evaluation

- Query update time
- Memory requirement
Data Center Network

Synthetic DCN topologies with 1024 nodes, 16 first level, 48 second level switch ports

Different topologies considered

Tree
VL2
Fat-tree
BCube
• Graphs represent connections in Gnutella P2P network during August of 2002
• The graphs vary from 8,111 to 22,686 nodes and 20,781 to 65,373 edges.
Snapshots of the connections between Autonomous Systems (AS) taken from the University of Oregon Route Views Archive Project

The resulting graphs (AS 500, . . . , AS 3000) have 500 to 3,000 nodes and 2,406 to 13,734 edges, with edge weights as high as 20,000
Discussion

- Networking tasks rely on **graph abstractions**
- Cloud DCNs experience **network dynamics**

- **NetGraph**: shared graph library for underlying graph operations
  - Receive topology updates and recompute graph queries
  - Implemented shortest path queries as a specific example
  - Scalable due to tree-decomposition based index maintenance
  - Architecture extensible to other graph queries

Questions?

Ramya Raghavendra
rraghav@us.ibm.com