TigerGraph

Conquering LDBC SNB BI At SF-10K

15th LDBC TUC Meeting

Mingxi Wu, Yuchen Zhang

June 2022

Disclaimer

- This is not an LDBC benchmark result
- The result presented is ready for audit
- It should not be used as a baseline to be compared against

LDBC Social Network Benchmark

	Interactive	Business Intelligence
Focus	OLTP	OLAP
Typical query	2-3 hop neighborhood queries with filtering	Multi-hop/path/subgraph queries with filtering and aggregation
Data size	30G, 100G, 300G	100G, 300G, 1T, 3T, 10T, 36T
Refresh operations	Single-point inserts	Micro-batch inserts/deletes
Target metric	Throughput (ops/s)	Mean latency

BI Workload

- **Target for OLAP**
- **Read Queries** (touching a large portion of the data)
- **Micro-batches of inserts and deletes**

Setup - Hardware

- **Software:** TigerGraph 3.6.0 RC
- **CPU type:** GCP N2D CPU AMD Milan EPYC 7B12 2.25GHz
- **Operating System:** CentOS 7 v20220406
- **Disk type:** SSD persistent disk
- **Machine Configuration:**
  - SF-1K: **4** x n2d-highmem-64 (64 vCPU, 512 GB RAM, 2000 GB disk)
  - SF-10K: **30** x n2d-highmem-96 (96 vCPU, 768 GB RAM, 3200 GB disk)

Setup - Data

- **LDBC-SNB SF-10K**
- **Raw Initial data:** 11.64T
- **batch update data:**
  - **Insert files:** 1.59T
  - **Delete files**: 24G
- **Loading performance:**
  - **Time:** 5439s ~ 1.5h
  - **Loading speed:** ~256.8 GB/hr/machine
  - **Disk size:** 6.06T

Vertices	**SF-1k**	**SF-10k**
Comment	1,876,785,283	18,880,439,325
Post	519,738,978	4,461,342,990
Forum	33,168,124	257,338,738
Person	3,399,580	26,384,952
Subtotal	2,433,117,531	23,625,531,571
Edges	**SF-1k**	**SF-10k**
CONTAINER_OF	519,738,978	4,461,342,990
HAS_CREATOR	2,396,524,261	23,341,782,315
HAS_INTEREST	79,573,188	617,405,426
HAS_MEMBER	3,004,740,356	29,398,116,490
HAS_MODERATOR	33,168,124	257,338,738
HAS_TAG	3,305,380,125	32,807,993,807
KNOWS	187,247,788	1,854,528,925
LIKES	3,390,033,467	38,373,535,968
REPLY_OF	1,876,785,283	18,880,439,325
STUDY_AT	2,719,877	21,108,848
WORK_AT	7,398,286	57,416,114
Subtotal	17,203,259,133	173,439,201,772

Results

SF	load	w	r	q1	q2a	q2b	q3	q4	q5	q6	q7	q8a	q8b	q9	q10a
1,000	4,295	1,758	6,199	136	127	28	315	51	18	18	42	36	16	522	293
10,000	5,439	4,165	16,954	190	268	114	974	88	65	71	213	99	39	1,112	684

SF	q10b	q11	q12	q13	q14a	q14b	q15a	q15b	q16a	q16b	q17	q18	q19a	q19b	q20
1,000	70	89	182	755	484	70	762	1,422	101	31	117	333	81	78	22
10,000	119	181	442	2,266	1,506	324	2,094	3,195	433	78	319	1,602	217	223	41

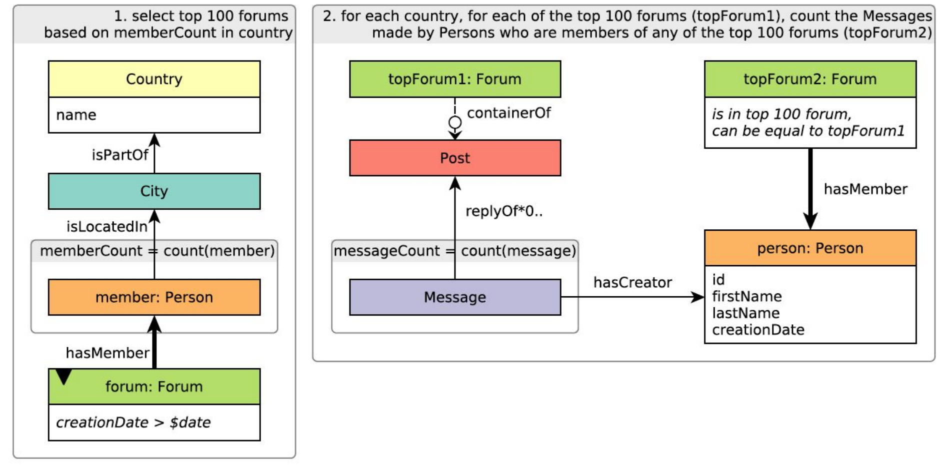
- w (write): sum of insert, delete and precompute time
- r (read): sum of read queries q1 ~ q20 time (in total 27 variations)
- qi - sum of 10 runs time of different input parameters
- Overall: batch = w + r
  - SF-1K: 1 batch takes 7,957s= 2.21h, (23s/query, total 27 query)
  - SF-10K: 1 batch takes 21,119s= 5.8h, (63s/query, total 27 query)

Implementation - Precomputation

Precomputation is used in BI 4, 6, 19 and 20

For example, query 4:

1. Precomputes: for a given forum, calculate the max MemberCount of all countries and store it as an auxiliary attribute of the forum
2. When querying, get the top 100 forums using the auxiliary attribute

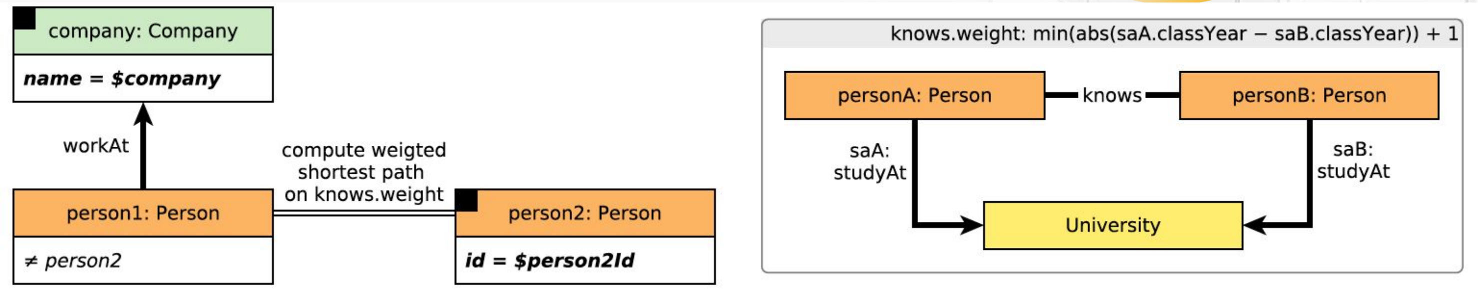

1. select top 100 forums based on memberCount in country
Country
name
isPartOf
City
isLocatedIn
memberCount = count(member)
member: Person
hasMember
forum: Forum
creationDate > $date
2. for each country, for each of the top 100 forums (topForum1), count the Messages made by Persons who are members of any of the top 100 forums (topForum2)
topForum1: Forum
containerOf
Post
replyOf*0..
messageCount = count(message)
Message
hasCreator
topForum2: Forum
is in top 100 forum, can be equal to topForum1
hasMember
person: Person
id
firstName
lastName
creationDate

Implementation - shortest paths

Query 15, 19, 20 is related to shortest path searching

1. Precomputes: the edge weight is calculated and inserted as weighted edge
2. When querying, we used GSQL query to search the shortest paths on the added edges.

Query 20



Implementation - insert

Insert operations are performed using GSQL loading scripts

- Files are distributed on cluster and is loaded concurrently
- Same speed as the intial loading job

Implementation - delete

Delete operations are cascade. Deletes are done by GSQL DML queries.

- The vertices/edges are selected in the query
- The selected vertices and edges are deleted