TigerGraph evaluation (vs Janusgraph)

Test environment (single node)

CPU	Intel(R) Xeon(R) E5-2682, 64 cores
Memory	512GB
Disk	6TB SSD
Network	25Gb/s

Datasets

Dataset	# vertices (million)	# edges (million)	Raw size (GB)
Twitter_rv	41.6	1,470	26.0
LDBC(10W)	43.0	98	6.8

1: Twitter_rv : http://an.kaist.ac.kr/traces/WWW2010.html

2: LDBC(10W): https://github.com/ldbc/ldbc_snb_implementations

Commands to generate the datasets (the schema can be found in the Appendix):

```
git clone https://github.com/ldbc/ldbc_snb_datagen.git
mvn clean package -DskipTests
# Edit the configuration file "params.ini" to control the amount of the data
vi params.ini
hadoop jar target/ldbc_snb_datagen-0.2.7-jar-with-dependencies.jar ./params.ini
```

Important parameters used to generate our test datasets:

- Configure the number of Persons to 100,000
- Use the following 12 tables:

1) comment,
2) person,
3) post,
4) place,
5) tag,
6) comment_replyOf_comment,
7) comment_hasCreator_person,
8) person_knows_person,
9) comment_replyOf_post,

10) post_hastag_tag,

11) post_isLocatedIn_place,

12) post_hasCreator_person.

Data loading

Dataset	TigerGraph (s)	Janusgraph (s)
Twitter_rv	652.09	7,982
LDBC(10W)	197.22	1,731

Size of loaded data

Dataset	Raw size (GB)	TigerGraph (GB)	Janusgraph (GB)
Twitter_rv	26.0	11	47
LDBC(10W)	6.8	13	19.2

Query latency

k-neighbor query using Twitter_rv

k-neighbor	# results (AVG)	TigerGraph (ms)	Janusgraph (ms)
1	1,611	5.60	12.50
2	6,765,143	351.00	66,700.00

LDBC Social Network Benchmark (SNB interactive)

case	TigerGraph (ms)	Janusgraph (ms)
1	9.56	3.63
2	15.99	829.63
3	12.48	301.70
4	56.13	13,816.72
5	13.33	273.58
6	54.84	6,422.00
7	10.84	195,440.00

Concluding remarks

- +3 TigerGraph loads data very fast, with low storage cost.
- +2 TigerGraph's query performance is much better than Janusgraph.

- -0 While TigerGraph's G-SQL looks somewhat like SQL, it is not. For simple/common cases, the benefit comparing to using Gremlin isn't that clear, and may make applications less portable.
- -1 TigerGraph's performance could become (much) worse when the query involves reading and/or filtering on attribute values (in addition to graph topology).
- -2 For 6 out of the 7 LDBC SNB cases, TigerGraph's performance is worse than using an open-sourced dataflow system (with our home-built Gremlin compiler); following company policy, we didn't include the actual numbers in the report.

Appendix

Janusgraph queries for LDBC SNB

1	g.V().hasLabel("comment").has("id",%sL).as("x").in("comment_replyof_comment").as("y").valueMap().as("reply").select("y").out("comment_hascreator_person").valueMap().as("replycreator").select("reply","replycreator").count()
2	g.V().hasLabel("person").has("id",%sL).repeat(__.out("person_knows_person")).times(3).emit().dedup().has("firstname",eq("John")).count()
3	g.V().hasLabel("person").out("person_knows_person").as("friend").in("post_hascreator_person").has("creationdate",lt("2011-03-27")).as("post").order().by("creationdate",decr).by(id(),incr).count()
4	g.V().hasLabel("person").has("id",%sL).as("0").out("person_knows_person").as("x").out("person_knows_person").as("x").select(all,"x").unfold().dedup().where(neq("0")).as("friend").in("post_hascreator_person").has("creationdate",inside("2011-03-27","2011-04-27")).where(__.out("post_islocatedin_place").has("name",eq("Vietnam"))).as("post").count()
5	g.V().hasLabel("person").has("id",%sL).as("root").out("person_knows_person").inE("post_hascreator_person").outV().has("creationdate",inside("2011-03-27","2011-04-27")).as("post").outE("post_hastag_tag").inV().as("tag").select("tag").by("id").groupCount().order(local).by(values,decr).unfold().count()

6	g.V().hasLabel("person").has("id",%sL).repeat(out("person_knows_person").simplePath()).emit().times(2).dedup().in("post_hascreator_person").as("post").out("post_hastag_tag").groupCount().by("name").order(local).by(values,decr)
7	g.V().hasLabel("person").has("id",%sL).in("post_hascreator_person").as("post").in("comment_replyof_post").as("comment").out("comment_hascreator_person").as("commenter").order().by(select("comment").by("creationdate")).select("commenter","comment").count()

TigerGraph queries for LDBC SNB

Case 1:

```
    CREATE QUERY query1(VERTEX<comments> p) FOR GRAPH ldbc_10w{
       SumAccum<INT> @@cnt = 0;
        start = {p};
        reply = SELECT tgt FROM start:s-(comment_replyed_comment:e)->comments:tgt;
        replyCreator = SELECT tgt FROM reply:s-
    (comment_hascreator_person:e)->person:tgt ACCUM @@cnt += 1;
        PRINT @@cnt;
    }
```

Case2:

```
CREATE QUERY query2(VERTEX<person> p) FOR GRAPH ldbc_10w{
   SumAccum<INT> @@cnt = 0;
    OrAccum @visited = false;
    start = {p};
    friend = {};
    friend01 = SELECT tgt FROM start:s-(person_knows_person:e)->person:tgt ACCUM
tgt.@visited += true, s.@visited += true;
    friend02 = SELECT tgt FROM friend01:s-(person_knows_person:e)->person:tgt WHERE
tgt.@visited == false ACCUM tgt.@visited += true;
    friend03 = SELECT tgt FROM friend02:s-(person_knows_person:e)->person:tgt WHERE
tgt.@visited == false ACCUM s.@visited += true;
    result = SELECT tgt FROM friend01:tgt WHERE tgt.firstName == "John" ACCUM @@cnt +=
1;
```

```
    result = SELECT tgt FROM friend02:tgt WHERE tgt.firstName == "John" ACCUM @@cnt +=
1;
    result = SELECT tgt FROM friend03:tgt WHERE tgt.firstName == "John" ACCUM @@cnt +=
1;
    PRINT @@cnt;
}
```

Case3:

```
CREATE QUERY query3(VERTEX<person> p) FOR GRAPH ldbc_10w{
    SumAccum<INT> @@cnt = 0;
    start = {p};
    friend = SELECT tgt FROM start:s-(person_knows_person:e)->person:tgt;
    Post = SELECT tgt FROM friend:s-(person_create_post:e)->post:tgt WHERE
datetime_diff(tgt.creationDate, to_datetime("2011-03-27")) < 0 ORDER BY
tgt.creationDate DESC, tgt.id ASC;
    result = select tgt from Post:tgt ACCUM @@cnt += 1;
    PRINT @@cnt;
}
```

Case4:

```
CREATE QUERY query4(VERTEX<person> p) FOR GRAPH ldbc_10w{
    SumAccum<INT> @@cnt = 0;
    OrAccum @visited = False;
    start = {p};
    friends={};
    friend = SELECT tgt FROM start:s-(person_knows_person:e)->person:tgt ACCUM
tgt.@visited += true, s.@visited += true;
    friendFriend = SELECT tgt FROM friend:s-(person_knows_person:e)->person:tgt WHERE
tgt.@visited == false ACCUM tgt.@visited += true;
    friends = friend UNION friendFriend;
    result = SELECT tgt FROM friends:s-(person_create_post:e)->post:tgt WHERE
datetime_diff(tgt.creationDate, to_datetime ("2011-03-27")) > 0 AND
datetime_diff(tgt.creationDate, to_datetime ("2011-04-27") )< 0;
    finalResult = SELECT s FROM result:s-(post_islocatedin_place:e)->place:pl WHERE
pl.name == "Vietnam" ACCUM @@cnt += 1;
    PRINT @@cnt;
}
```

Case5:

```
CREATE QUERY query5(VERTEX<person> p) FOR GRAPH ldbc_10w{
    SumAccum<INT> @@cnt = 0;
    MapAccum< INT,INT> @@tagPostCounts;
    start = {p};
    friend = SELECT tgt FROM start:s-(person_knows_person:e)->person:tgt;
    Post = SELECT tgt FROM friend:s-(person_create_post:e)->post:tgt WHERE
datetime_diff(tgt.creationDate, to_datetime ("2011-03-27")) > 0 AND
datetime_diff(tgt.creationDate, to_datetime ("2011-04-27")) < 0;
    Tags = SELECT tgt FROM Post:s-(post_hastag_tag:e)->tag:tgt ACCUM @@tagPostCounts +=
(tgt.id -> 1);
    results = SELECT tgt FROM Tags:tgt ORDER BY @@tagPostCounts.get(tgt.id) DESC;
    finalresult = select tgt from results:tgt ACCUM @@cnt += 1;
    PRINT @@cnt;
}
```

Case6:

```
CREATE QUERY query6(VERTEX<person> p) FOR GRAPH ldbc_10w{
    SumAccum<INT> @@cnt = 0;
    OrAccum @visited = false;
    MapAccum< STRING,INT> @@tagPostCounts;
    start = {p};
```

```
   friendSet = {};
   friend = SELECT tgt FROM start:s-(person_knows_person:e)->person:tgt ACCUM
tgt.@visited += true, s.@visited += true;
   friendFriend = SELECT tgt FROM friend:s-(person_knows_person:e)->person:tgt WHERE
tgt.@visited == false;
   friendSet = friend UNION friendFriend;
   Post = SELECT tgt FROM friendSet:s-(person_create_post:e)->post:tgt;
   Tags = SELECT tgt FROM Post:s-(post_hastag_tag:e)->tag:tgt ACCUM @@tagPostCounts +=
(tgt.name -> 1);
   results = SELECT tgt FROM Tags:tgt ORDER BY @@tagPostCounts.get(tgt.name) DESC LIMIT
10;
   PRINT results.name, @@tagPostCounts.get(results.name);
}
```

Case7:

```
CREATE QUERY query7(VERTEX<person> p) FOR GRAPH ldbc_10w{
   SumAccum<INT> @@cnt = 0;
   start = {p};
   posts = SELECT tgt FROM start:s-(person_create_post:e)->post:tgt;
   comment = SELECT tgt FROM posts:s-(post_replyed_comment:e)->comments:tgt ORDER BY
tgt.creationDate ASC;
   commenter = SELECT tgt FROM comment:s-(comment_hascreator_person:e)->person:tgt
ACCUM @@cnt += 1;
   PRINT @@cnt;
}
```

Schema of the LDBC-10W

```
CREATE VERTEX person (PRIMARY_ID id UINT, id UINT, firstName STRING, lastName STRING,
gender STRING, birthday STRING, creationDate STRING, locationIP STRING, browserUsed
STRING)
CREATE VERTEX post (PRIMARY_ID id STRING, id STRING, imageFile STRING, creationDate
STRING, locationIP STRING, browserUsed STRING, lang STRING, content STRING, length UINT)
CREATE VERTEX tag (PRIMARY_ID id UINT, id UINT, name STRING, url STRING)
CREATE VERTEX place (PRIMARY_ID id UINT, id UINT, name STRING, url STRING, tp STRING)
CREATE VERTEX comments (PRIMARY_ID id UINT, id UINT, creationDate STRING, locationIP
STRING, browserUsed STRING, content STRING, length UINT)
CREATE DIRECTED EDGE person_knows_person (FROM person, TO person, creationDate STRING)
WITH REVERSE_EDGE="person_known_person"
CREATE DIRECTED EDGE comment_replyof_post (FROM comments, TO post) WITH
REVERSE_EDGE="post_replyed_comment"
CREATE DIRECTED EDGE comment_replyof_comment (FROM comments, TO comments) WITH
REVERSE_EDGE="comment_replyed_comment"
CREATE DIRECTED EDGE post_hascreator_person (FROM post, TO person) WITH
REVERSE_EDGE="person_create_post"
CREATE DIRECTED EDGE post_hastag_tag (FROM post, TO tag) WITH
REVERSE_EDGE="tag_hadtag_post"
CREATE DIRECTED EDGE comment_hascreator_person (FROM comments, TO person) WITH
REVERSE_EDGE="person_hadcreator_comment"
CREATE DIRECTED EDGE post_islocatedin_place (FROM post, TO place) WITH
REVERSE_EDGE="place_waslocatedin_post"
CREATE GRAPH ldbc_10w (person, post, tag, place, comments, person_knows_person,
comment_replyof_post, comment_replyof_comment, post_hascreator_person, post_hastag_tag,
comment_hascreator_person, post_islocatedin_place)
```