LDBC Benchmark on JanusGraph

https://github.com/Junyangz/ldbc_snb_janusgraph

0：IO | 张俊阳 张小洋 王传仁 刘志磊 冀海川 | 2018.6.20

LDBC Benchmark

LDBC社交网络图数据Benchmark是一项针对专业图数据库的基准测试工具

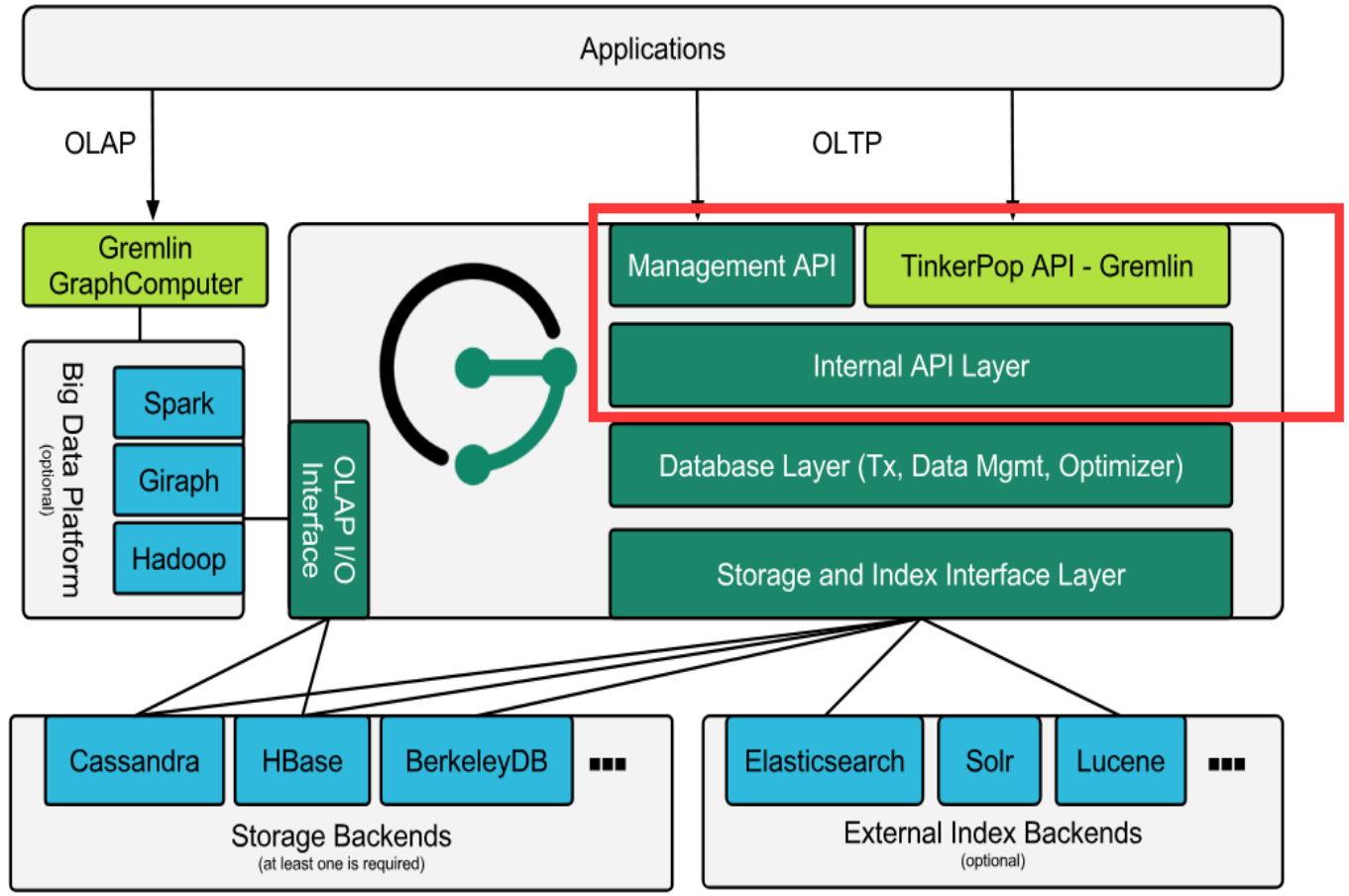
SNB主要部件有下列四项

- SNB benchmark 规范文档
- SNB数据生成器
- LDBC驱动（实现查询的驱动）
- 交互式工作负载实现

LDBC The graph & RDF benchmark reference

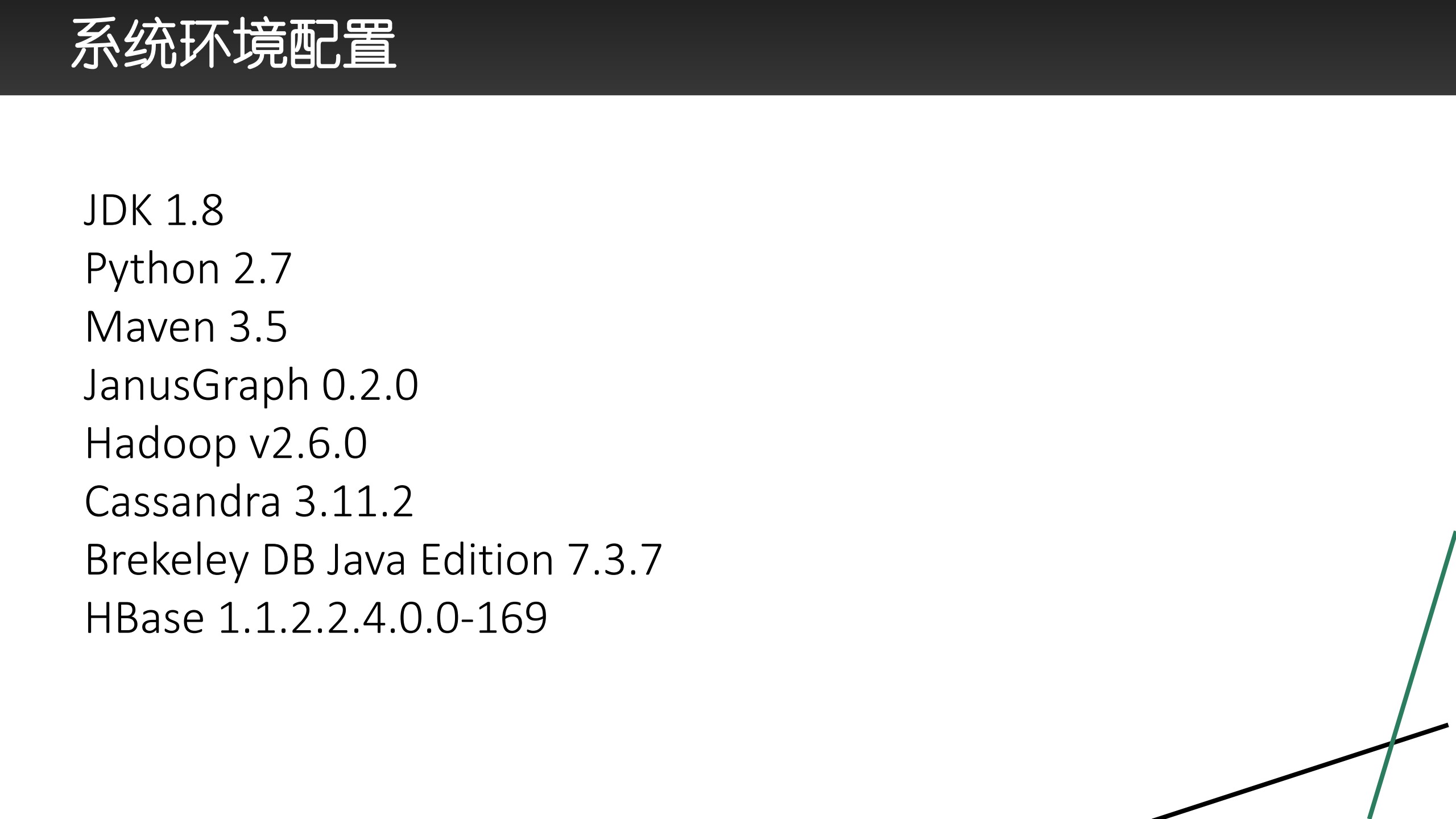
JanusGraph

JanusGraph是一个可扩展的图数据库，可以把包含数千亿个顶点和边的图存储在多机集群上。它支持事务，支持数千用户实时、并发访问存储在其中的图



系统环境配置

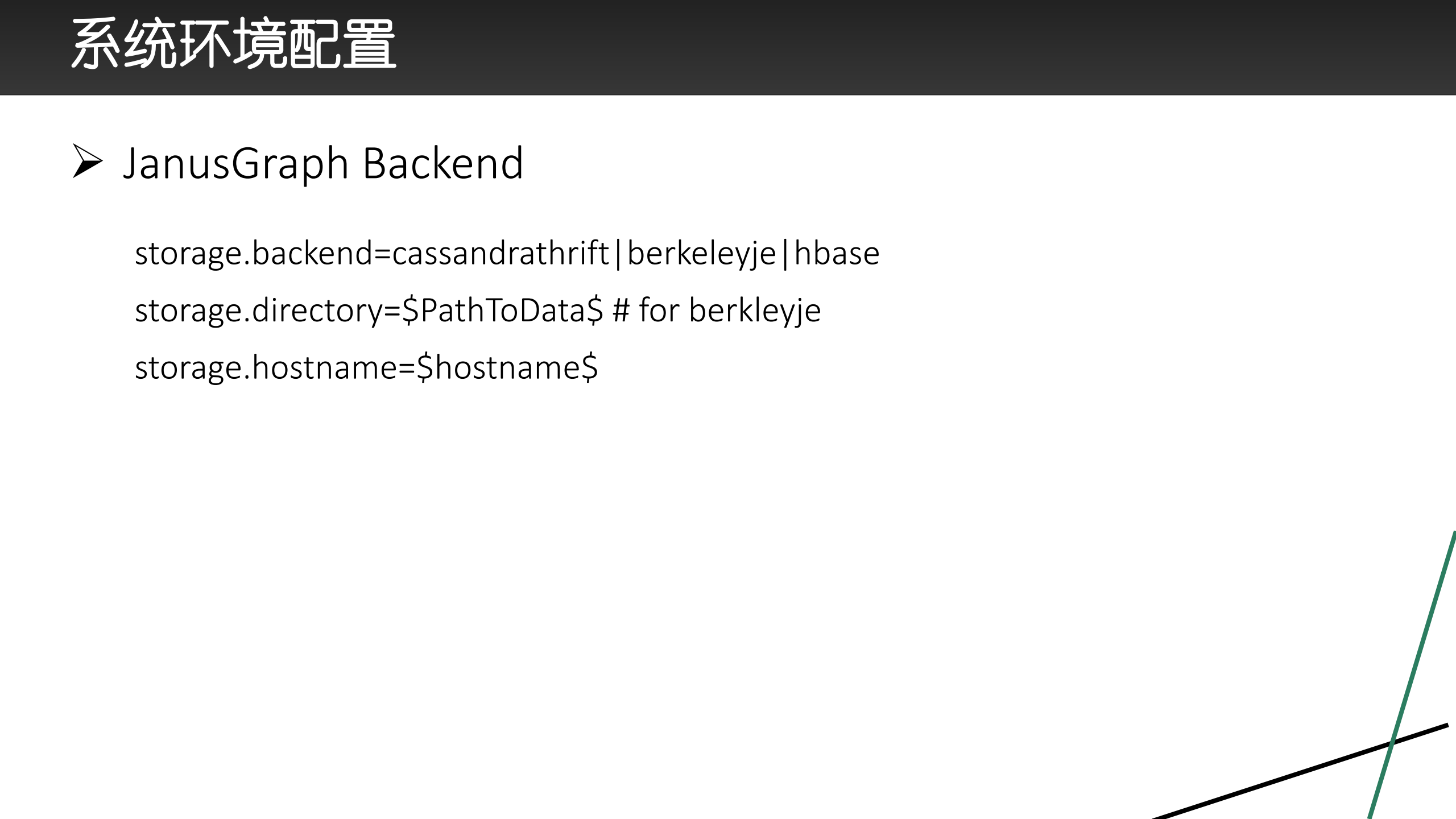
JDK 1.8
Python 2.7
Maven 3.5
JanusGraph 0.2.0
Hadoop v2.6.0
Cassandra 3.11.2
Brekeley DB Java Edition 7.3.7
HBase 1.1.2.2.4.0.0-169



系统环境配置

- JanusGraph Backend

```
storage.backend=cassandrathrift|berkeleyje|hbase
storage.directory=$PathToData$ # for berkleyje
storage.hostname=$hostname$
```



系统环境配置

- JanusGraph Backend: Cassandra

单节点：

Host9 192.168.5.34

两节点：

seed Host9 192.168.5.34 | node: Host4 192.168.5.29

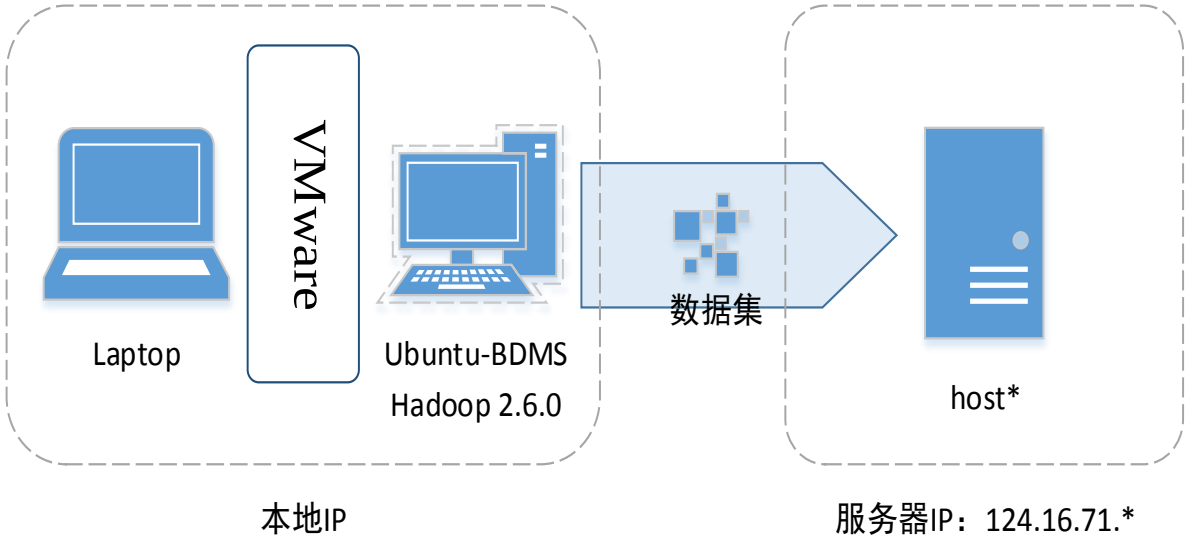
```
[user26@host9 ~]$ .opt/cassandra/bin/nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address       Load       Tokens       Owns (effective)  Host ID                               Rack
UN  192.168.5.29  61.83 MiB  256          48.3%             978c82e3-1562-4745-b825-e4d882e897c8  rack1
UN  192.168.5.34  46.61 MiB  256          51.7%             64ea6135-a87d-4d90-af8e-12c1be7e05f3  rack1

[user26@host9 ~]$
```

四节点：

```
[user26@host5 ~]$ .opt/cassandra/bin/nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address       Load       Tokens       Owns (effective)  Host ID                               Rack
UN  192.168.5.30  19.28 MiB  256          24.1%             cfbb1ad7-7322-44df-bed7-0fe0a8a343f6  rack1
UN  192.168.5.32  22.11 MiB  256          26.2%             b3a9df85-790d-4c64-b497-569911eb15b3  rack1
UN  192.168.5.33  21.83 MiB  256          25.6%             3073e423-53e7-4a40-ad8a-71111825dd7f  rack1
UN  192.168.5.35  25.96 MiB  256          24.1%             3f99fb7d-05d6-4d3a-9892-1f9adcd24610  rack1
```

数据生成

- ldbc_snb_datagen (以下简称datagen)使用Apache Hadoop v2.6.0 进行数据生成
- 使用Maven 进行ldbc_snb项目的构建
- 使用python脚本对生成数据属性自定义设置



数据生成

- Datagen 参数 （图属性、数据规模、数据项格式）

- **ldbc.snb.datagen.serializer.personSerializer**
  ldbc.snb.datagen.serializer.snb.interactive.CSVPersonSerializer
- **ldbc.snb.datagen.serializer.invariantSerializer**
  ldbc.snb.datagen.serializer.snb.interactive.CSVInvariantSerializer
- **ldbc.snb.datagen.serializer.personActivitySerializer**
  ldbc.snb.datagen.serializer.snb.interactive.CSVPersonActivitySerializer

数据生成

- Datagen 参数 （图属性、数据规模、数据项格式）

- **ldbc.snb.datagen.generator.scaleFactor**
  snb.interactive.0.1

Scale Factor	1	3	10	30	100	300	1000
# of Persons	11K	27K	73K	182K	499K	1.25M	3.6M
# of Years	3	3	3	3	3	3	3
Start Year	2010	2010	2010	2010	2010	2010	2010

生成标准的大小为100MB左右的测试数据，基本上每个不同entity的数据内容分别存储在不同的csv文件中。

数据生成

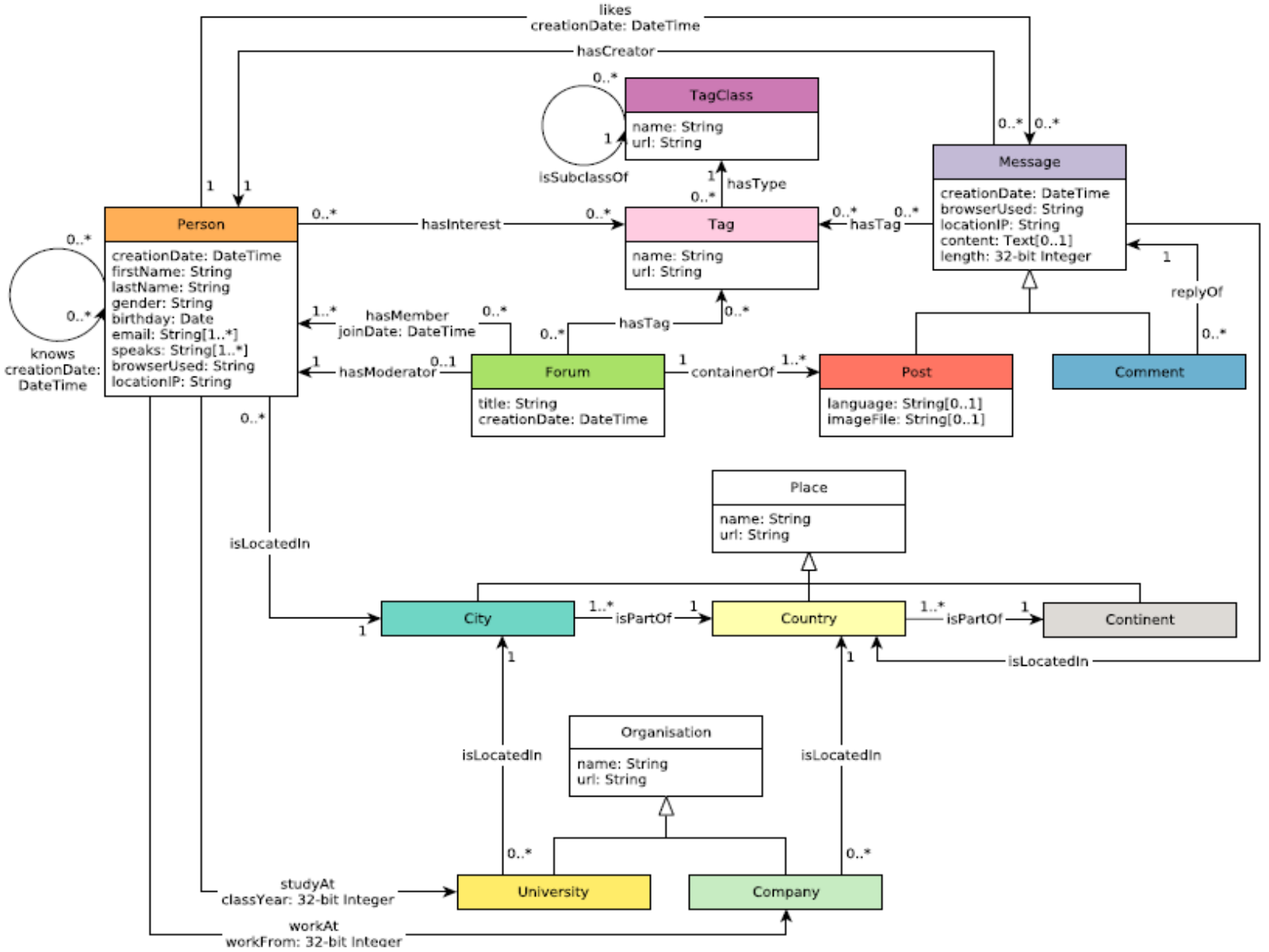
- Datagen 将生成三类文件

- **数据集**
  用于benchmark测试的主要数据集，大约占生成数据的90%

- **更新流（upadtae stream）**
  用于update query的数据，大约占生成数据的10%

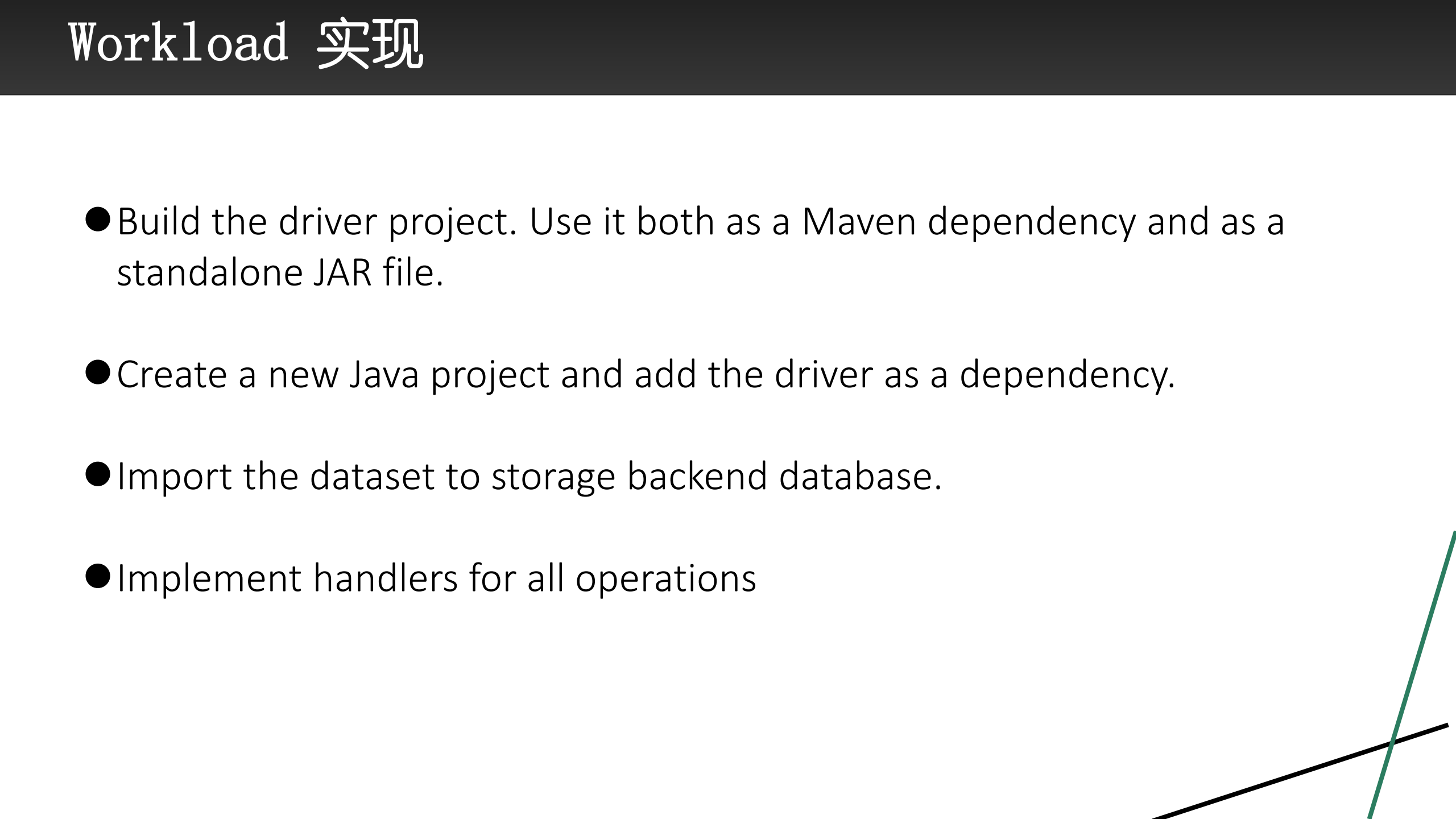
- **参数（substitution parameters）**
  用于BI query和IC query的参数

数据生成


likes
creationDate: DateTime
hasCreator
TagClass
name: String
url: String
isSubclassOf
hasType
Message
creationDate: DateTime
browserUsed: String
locationIP: String
content: Text[0..1]
length: 32-bit Integer
Person
creationDate: DateTime
firstName: String
lastName: String
gender: String
birthday: Date
email: String[1..*]
speaks: String[1..*]
browserUsed: String
locationIP: String
knows
creationDate: DateTime
hasInterest
Tag
name: String
url: String
hasTag
replyOf
hasMember
joinDate: DateTime
hasModerator
Forum
title: String
creationDate: DateTime
containerOf
Post
language: String[0..1]
imageFile: String[0..1]
Comment
Place
name: String
url: String
isLocatedIn
City
isPartOf
Country
isPartOf
Continent
Organisation
name: String
url: String
studyAt
classYear: 32-bit Integer
University
Company
workAt
workFrom: 32-bit Integer

Workload 实现

- Build the driver project. Use it both as a Maven dependency and as a standalone JAR file.
- Create a new Java project and add the driver as a dependency.
- Import the dataset to storage backend database.
- Implement handlers for all operations



数据导入

- 使用importer工具进行数据导入

  将生成的social_network数据导入到JanusGraph指定的存储后端中

  通过执行com.ldbc.snb.janusgraph.importer.Main类来导入

  基于LDBC ldbc_snb_implementations开源代码修改实现

数据导入

- 导入过程

启动JansusGraph server（gremlin-server）

java -cp target/janusgraphSNBInteractive-0.1-SNAPSHOT-jar-with-dependencies.jar com.ldbc.snb.janusgraph.importer.Main [-n 2] [-s 2000] [-d test-data-100m/social_network] [-c bdb.conf]

--numThreads/-n　　#加载过程中线程数

--transactionSize/-s　#读取文件事务的大小（每个读取任务读取的行数）

--dataset/-d　　#要导入数据集的文件夹路径

--backend-config/-c　#配置后端存储的文件路径

数据导入

- 导入过程　　以Cassandra作为存储后端

```
[user26@host9 janusgraph]$ ./import-to-janusgraph.sh
17:23:21.540 [main] INFO  org.janusgraph - entered init
17:23:23.511 [main] INFO  o.j.g.c.GraphDatabaseConfiguration - Set default timestamp
provider MICRO
17:23:23.525 [main] INFO  o.j.d.c.t.CassandraThriftStoreManager - Closed Thrift conne
ction pooler.
17:23:23.530 [main] INFO  o.j.g.c.GraphDatabaseConfiguration - Generated unique-insta
nce-id=c0a8052223854-host91
17:23:23.558 [main] INFO  org.janusgraph.diskstorage.Backend - Initiated backend oper
ations thread pool of size 48
17:23:28.128 [main] INFO  o.j.diskstorage.log.kcvs.KCVSLog - Loaded unidentified Read
Marker start time 2018-06-20T09:23:28.128Z into org.janusgraph.diskstorage.log.kcvs.K
CVSLog$MessagePuller@4d15107f
Connected
17:23:28.146 [main] INFO  org.janusgraph - Building Schema
17:23:28.147 [main] INFO  org.janusgraph - Creating Vertex Labels
17:23:30.316 [main] INFO  org.janusgraph - Creating edge labels
17:23:32.023 [main] INFO  org.janusgraph - Creating edge labels
17:23:32.139 [main] INFO  org.janusgraph - Created Property Key browserUsed
17:23:32.251 [main] INFO  org.janusgraph - Created Property Key length
17:23:32.365 [main] INFO  org.janusgraph - Created Property Key locationIP
17:23:32.496 [main] INFO  org.janusgraph - Created Property Key Comment.id
17:23:32.622 [main] INFO  org.janusgraph - Created Property Key creationDate
17:23:32.735 [main] INFO  org.janusgraph - Created Property Key content
```

数据导入

- 导入过程　　以Cassandra作为存储后端

```
17:32:19.819 [main] INFO  org.janusgraph - completed loading of Vertex Properties
17:32:19.820 [main] INFO  org.janusgraph - completed import data
17:32:19.820 [Thread-4] INFO  org.janusgraph - Stats reporting thread interrupted
17:32:19.820 [main] INFO  org.janusgraph - Number of vertices loaded: 327588, Number
of edges loaded 1477965
17:32:19.820 [Thread-4] INFO  org.janusgraph - Vertices Loaded 327588, Edges Loaded 1
477965, Properties Loaded 2389451, Current vertices loaded/s 0, Current edges loaded/
s 1972, Current properties loaded/s 1750
17:32:19.904 [main] INFO  o.j.d.c.t.CassandraThriftStoreManager - Closed Thrift conne
ction pooler.
```

约100M的图数据包含327588个点，1477965条边，
2389451条Properties

Workload 实现

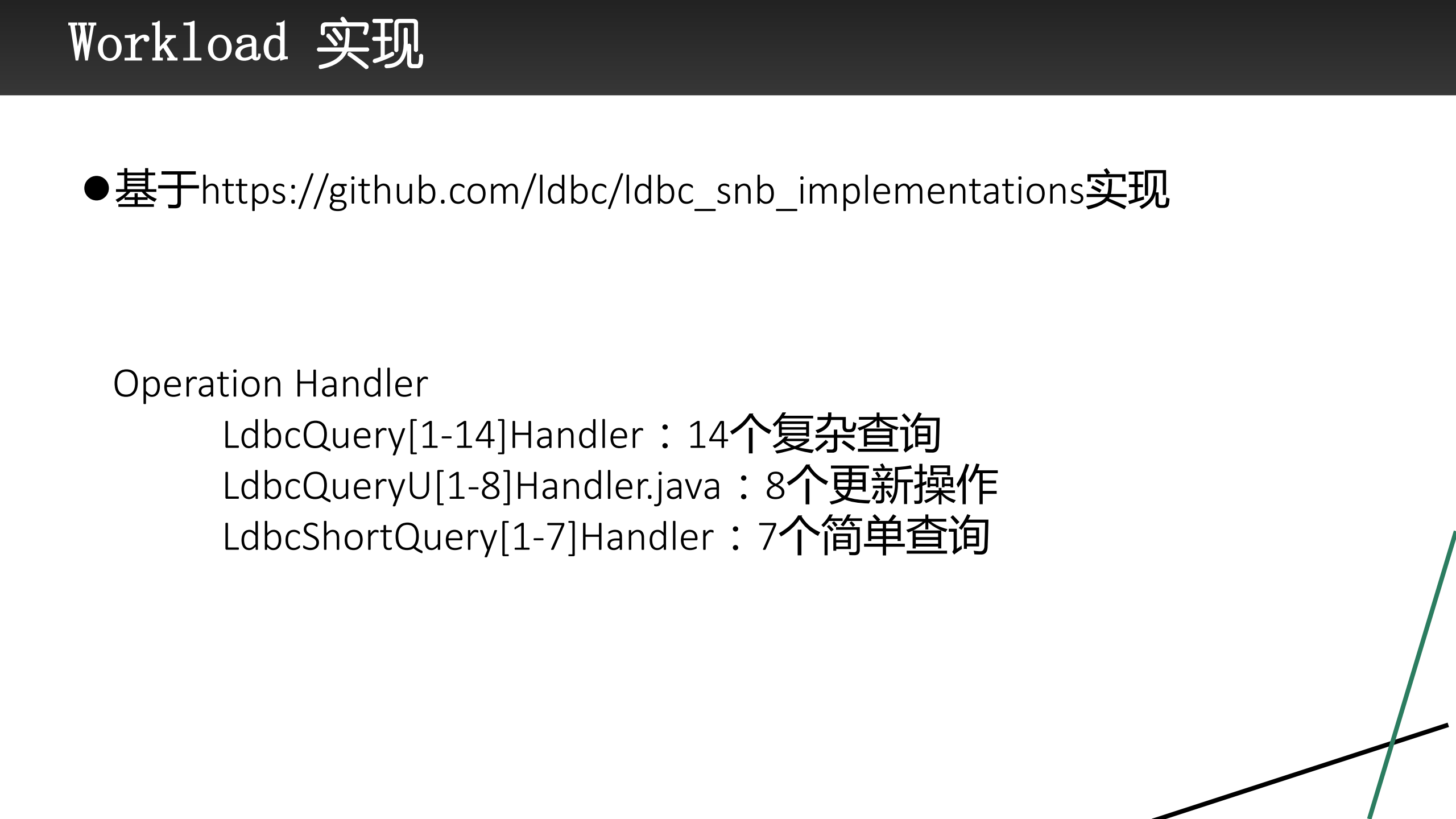
- 基于https://github.com/ldbc/ldbc_snb_implementations实现

Operation Handler

LdbcQuery[1-14]Handler：14个复杂查询

LdbcQueryU[1-8]Handler.java：8个更新操作

LdbcShortQuery[1-7]Handler：7个简单查询



Workload 实现

- 以LdbcQuery2为例

```
query="g.V().has('Person.id', $id)."+
        "out('knows').as('friend').valueMap().as('x').in('hasCreator').has('creationDa
      te',P.lte($maxDate))."+
      "order().by('creationDate',decr).by('messageId',incr)."+
      "limit($limit).as('post').valueMap().as('y')"+
      ".select('x','y')\n";

ResultSet resultSet = dbConnectionState.runQuery(query, parameters);
```

Workload 实现

- 以LdbcQuery2为例


```
"unit" : "MILLISECONDS",
"throughput" : 13.820335636722607,
"all_metrics" : [ {
  "name" : "LdbcQuery1",
  "count" : 31,
  "unit" : "MILLISECONDS",
  "run_time" : {
    "name" : "Runtime",
    "unit" : "MILLISECONDS",
    "count" : 31,
    "mean" : 4.67741935483871,
    "min" : 3,
    "max" : 8,
    "25th_percentile" : 0,
    "50th_percentile" : 4,
    "75th_percentile" : 0,
    "90th_percentile" : 6,
    "95th_percentile" : 6,
    "99th_percentile" : 8,
    "99.9th_percentile" : 8,
    "std_dev" : 1.0282179000328533
  }
}, {
  "name" : "LdbcQuery2",
  "count" : 17,
  "unit" : "MILLISECONDS",
  "run_time" : {
    "name" : "Runtime",
    "unit" : "MILLISECONDS",
    "count" : 17,
    "mean" : 5.235294117647059,
    "min" : 3,
    "max" : 7,
    "25th_percentile" : 0,
    "50th_percentile" : 5,
    "75th_percentile" : 0,
    "90th_percentile" : 6,
    "95th_percentile" : 6,
    "99th_percentile" : 7,
    "99.9th_percentile" : 7,
    "std_dev" : 1.0017286097603766
  }
```

图分割

- 默认策略

随机划分策略。随机安排顶点到所有机器上。

缺点：查询效率慢，存在大量的跨节点的通信。

- 显式划分

```
cluster.partition = true        //开启集群自定义分区策略
cluster.max-partitions = 32       //最大的虚拟分区数
ids.flush = false
```

图分割

- 显式划分

Edge Cut (默认)

- 对于频繁遍历的边，应该减少cut edge的存在，从而减少跨设备间的通信，提高查询效率。即把进行遍历的相邻顶点放在相同的分区，降低通信消耗。

Vertex Cut

- 目的：一个拥有大量边的顶点，在加载或者访问时会造成热点问题。Vertex Cut 通过分散压力到集群中所有实例从而缓解单顶点负载。

测试结果

```
erations [47], Last [00:00.028 (m:s.ms)], Throughput (Total) [13.74] (Last 3s) [13.74]
Shutting down status thread...
03:37:55,200  INFO ExecuteWorkloadMode:40 - Shutting down workload...
03:37:55,200  INFO ExecuteWorkloadMode:40 - Shutting down completion time service...
03:37:55,301  INFO ExecuteWorkloadMode:40 - Shutting down metrics collection service...
03:38:00,407  INFO ExecuteWorkloadMode:78 -
-------------------------------------------------------------------------------
Operation Count:                        56
Duration:                               00:04.052.000 (m:s.ms.us)
Throughput:                             13.82 (op/s)
Start Time (China Standard Time):       2018-06-20 - 03:37:50.974
Finish Time (China Standard Time):      2018-06-20 - 03:37:55.026
-------------------------------------------------------------------------------
    LdbcQuery1
        Units:              MILLISECONDS
        Count:              31
        Min:                3
        Max:                8
        Mean:               4.68
        50th Percentile:    4
        90th Percentile:    6
        95th Percentile:    6
        99th Percentile:    8
    LdbcQuery2
        Units:              MILLISECONDS
        Count:              17
        Min:                3
        Max:                7
        Mean:               5.24
        50th Percentile:    5
        90th Percentile:    6
        95th Percentile:    6
        99th Percentile:    7
    LdbcQuery3
        Units:              MILLISECONDS
        Count:              8
        Min:                7
        Max:                15
        Mean:               9.88
        50th Percentile:    8
        90th Percentile:    14
        95th Percentile:    15
```

Thanks