LDBC

The graph & RDF benchmark reference

Full Disclosure Report of the LDBC Social Network Benchmark

Audit of the LDBC Social Network Benchmark's Business Intelligence Workload over TigerGraph

April 6, 2023

General Terms

Executive Summary

This report documents an audited execution of the LDBC SNB BI (Social Network Benchmark Business Intelligence) workload for TigerGraph.

TigerGraph is a massively parallel processing (MPP) graph database management system, designed for handling hybrid transaction/analytical processing (HTAP) query workloads. It is a distributed platform using a native graph storage format with an edge cut partitioning strategy. Within this, each segment (partition) of the graph holds a similar amount of vertices and processes requests in parallel. TigerGraph offers GSQL, a Turing-complete query language which provides both declarative features (e.g. graph patterns) as well as imperative ones (e.g. for expressing iterative graph algorithms with loops and accumulator primitives). TigerGraph previously passed the LDBC SNB BI benchmark with a single-machine on-premise setup using the SF1 000 dataset.

Declaration of Audit Success

This report contains details of a successful execution of the LDBC SNB BI benchmark. The results have been gathered by an independent auditor who has validated the implementation of the queries and verified the system's configuration conforms to the description of the benchmark and its strict requirements.

Sponsorship and Funding Disclaimer

TigerGraph, as an LDBC member, are the Test Sponsor of this audit. The audit and its associated execution costs (compute, storage) were funded by AMD. This arrangement was deemed acceptable by both parties, the LDBC Steering Committee and the Auditor.

DocuSigned by:
David Püroja
C7FE4F71E2564AE...

4/6/2023

Mr. David Püroja
(Auditor)

Date

Pometry
(Audit Company)

DocuSigned by:
Gabor Szarnyas
FA33C2C8CB944A5...

4/6/2023

Dr. Gábor Szárnyas
(Leader of LDBC SNB Task Force)

Date

DocuSigned by:
Songting Chen
BD88B2D9F5AA4C8...

4/6/2023

Dr. Songting Chen
(Test Sponsor Representative)

Date

Table of Contents

1 Benchmark Description

The audit was conducted in compliance with the Social Network Benchmark Business Intelligence workload's specification.

Table 1.1: Benchmark Overview

Artifact	**Version**	**URL**
Specification	2.2.0	`https://arxiv.org/pdf/2001.02299v7.pdf`
Data generator	0.5.0	`https://github.com/ldbc/ldbc_snb_datagen_spark/releases/tag/v0.5.0`
Driver and implementations	1.0.3	`https://github.com/ldbc/ldbc_snb_bi/releases/tag/v1.0.3`

2 System Description and Pricing Summary

2.1 Details of machines driving and running the workload

2.1.1 Machine overview

The audit was conducted on 48 virtual machine instances in the Amazon Web Service (AWS) cloud, placed in an AWS Placement group with the cluster strategy to reduce network latency[1]. The details below where obtained from the AWS console.

Table 2.1: Machine Type and Location

Cloud provider	Amazon Web Services
Machine region	N. Virginia (us-east-1)
Common name of the item	r6a.8xlarge
Operating system	Amazon Linux 2 (5.10.167-147.601.amzn2.x86_64)

2.1.2 CPU details

The details below were obtained using the command `cat /proc/cpuinfo` (Listing A.1) and `dmidecode -t processor` (Listing A.3) issued from the machine instance.

Table 2.2: CPU details summary

Type	AMD® AMD EPYC® 7R13 CPU
Total number	1
Cores per CPU	16
Threads per CPU core	2
CPU clock frequency	3.725 MHz
Total cache size per CPU	L1i cache: 32KiB L1d cache: 32KiB L2 cache: 512KiB L3 cache: 33MiB

2.1.3 Memory details

The total size of the memory installed is 256GiB and the type of memory is DDR4 with frequency 3200MHz. This information was obtained using the `sudo lshw -c memory` command (Listing A.5) issued from the virtual machine instance.

2.1.4 Disk and storage details

The virtual machine instance used a Amazon EBS General Purpose (GP) 3 storage, formatted in `xfs`, configured with properties shown in Table 2.3.

Disk controller or motherboard type was not obtainable from the virtual machine instance. Information on AWS General Storage can be found on the website `https://aws.amazon.com/ebs/general-purpose/` (accessed: March 16, 2023).

The 4KB QD1 write performance on the data disk was measured with the `fio` command and the output (Listing B.1) showed an average of 1 574 IOPS.

[1]https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html#placement-groups-cluster

Table 2.3: Amazon EBS GP3 Configuration

Type	GP3
Size	1000GB
IOPS	16000
Throughput	1000

2.1.5 Network details

The benchmark was run using 48 `r6a.8xlarge` instances, both deployed in the same availability zone. All instances where assigned security groups to open the ports required for communication between the nodes[2]. The `r6a.8xlarge` instances use the Elastic Network Adapter provided by Amazon. This information was obtained using the `lshw -class network` command (Listing B.2).

2.1.6 Machine pricing

The system pricing summary in US dollars (USD) is included in the table below. The pricing of the AWS machine instance is the price for a 3-year reserved dedicated instance machine without upfront payment using the EC2 Instance savings plan [3].

Table 2.4: Pricing summary

Item	**Price**
r6a.8xlarge reserved instance machine in AWS (standard 3-year term)	1 141 994 USD
Software license (3 years)	6 117 600 USD
Maintenance fee (3 years)	611 760 USD
Total cost	7 871 354 USD

2.1.7 System version and availability

Table 2.5: System versions

System	Version	License
TigerGraph	3.7.0	Enterprise Licence provided by TigerGraph

[2]`https://docs.tigergraph.com/tigergraph-server/current/reference/ports`

[3]`https://aws.amazon.com/savingsplans/compute-pricing/`

3 Dataset Generation

3.1 General information

The data generation settings of the LDBC Datagen are described below.

Table 3.1: Datagen settings summary

Data format for TigerGraph	composite-projected-fk layout, compressed CSV files
Scale factors for TigerGraph	10 (validation), 10000 (benchmark)
Data format for Umbra	composite-merged-fk layout, compressed CSV files
Scale factors for Umbra	10 (validation)

3.2 Datagen configurations

The datasets and query substitution parameters used for the benchmark and the cross-validation runs were retrieved from the following URLs and copied to an AWS S3 Bucket, with the `tar.zst` files uncompressed. The URLs are served by LDBC's official data repository, available as a public bucket in the Cloudflare R2 object storage.[1]

3.2.1 SF10

- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/parameters-2022-10-01.zip
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10-composite-merged-fk.tar.zst
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10-composite-projected-fk.tar.zst

3.2.2 SF10000

- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/parameters-2022-10-01.zip
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.000
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.001
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.002
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.003
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.004
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.005
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.006
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.007
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.008
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.009
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.010
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.011
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.012
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.013
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.014
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.015
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.016
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.017
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.018
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.019
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.020
- https://pub-383410a98aef4cb686f0c7601eddd25f.r2.dev/bi-pre-audit/bi-sf10000-composite-projected-fk.tar.zst.021

To re-generate these datasets from scratch, use the instructions provided in Appendix C.

[1] https://www.cloudflare.com/products/r2/

3.3 Data loading and data schema

For SF10000, the data is partitioned evenly accross each instance using a partitioning script. The partitioning is executed by calling the following command:

```
cd /data/ldbc_snb_bi/tigergraph
# change the following lines in k8s/vars.sh
# export NUM_NODES=48 # number of pods or nodes
# export SF=10000
# export TG_DATA_DIR=/home/tigergraph/sf${SF}
# export TG_PARAMETER=/home/tigergraph/parameters-sf${SF}

python3 download_all.py 10000 ./ip_list -t 20 --access_key_id $ACCESS_KEY_ID --secret_access_key
    $SECRET_ACCESS_KEY --region $BUCKET_REGION --bucket_name $BUCKET_NAME --provider $CLOUD_PROVIDER
tail -f ~/log.download
```

Listing 3.1: Script to execute the benchmark on TigerGraph for SF10000

The data preprocessing and loading times are reported below. Values were measured using the GNU Time tool (`/usr/bin/time`) with the `-v` flag, reading the *Elapsed (wall clock) time* from the output. The column **Data preprocessing time** shows how much time it took to preprocess the CSV files. For this benchmark execution, the preprocessing only consisted of decompressing the `.csv.gz` files, for which the timing of the first node is reported. The column **Data loading time** shows how long it took to create a graph from the input CSV files and perform the initial indexing, compilation of the queries and precomputation. The column **Total time** contains the sum of the data preprocessing and loading times.

The TigerGraph data schema is shown in Listing D.1.

Table 3.2: Data preprocessing and loading times for TigerGraph on scale factor 10000

Scale factor	Data preprocessing time (s)	Data loading time (s)	Total time (s)
10000	409	5 226	5 635

The decomposition of the **Data loading times** are shown in Table 3.3.

Table 3.3: Decomposition of data loading time

	Time (s)
Schema setup	43
Load data	4 289
Query install	79
Precompute	629
Rebuild	186
Total	5 226

4 Implementation Details

4.1 Execution mode

Section 7.5.2.2 of the SNB specification defines two execution modes for the *throughput batches*. In *disjoint read-write mode*, the updates for each day of the benchmark's simulation are applied in bulk, separately from the read queries (i.e. there are no overlapping read and write operations). In *concurrent read-write mode*, the updates are applied concurrently with the reads. Systems opting for concurrent read-write mode are subject to the LDBC ACID test[1].

In the current audited run, TigerGraph was executed using the *disjoint read-write mode*. Therefore, no ACID tests were conducted.

4.2 Use of auxiliary data structures

The TigerGraph implementation precomputes the following auxiliary data structures. These are executed in each batch after the writes have been applied.

- **Root Post:** For each Message node (Comments and Posts), an edge to the corresponding Message thread's root Post is inserted. These derived edges are maintained incrementally, i.e. root Post edges are inserted for newly inserted Messages and removed for deleted Messages.
- **Q4:** For each Forum, the maximum number of members (for number of members per country) is precomputed.
- **Q6:** For each Message, the popularityScore defined in the query is precomputed.
- **Q14:** The weight attributed on the knows edges are precomputed based on the number of interactions between the two Person nodes.
- **Q19:** The weight attributes on the knows edges are precomputed based on the number of interactions between the two Person nodes.
- **Q20:** The weight attributes on the knows edges are precomputed based on the classYear attributes on the studyAt edges that point to the same University from the endpoint Person nodes.

The precomputations for Q14 and Q19 are performed together using different scoring methods for establishing the edge weights. We display the runtime of this operation as "precomputation for Q14 and Q19" in Table 5.3.

[1] `https://github.com/ldbc/ldbc_acid`

4.3 Benchmark execution

The benchmark is executed using the following commands.
Note: despite what the script's name suggests, this is benchmark was executed on a AWS EC2 instance without Kubernetes running, with the TigerGraph instance running on a single server machine and not using any containerization technology.

```
cd /data/ldbc_snb_bi/tigergraph
# change the following lines in k8s/vars.sh
# export NUM_NODES=48 # number of pods or nodes
# export SF=10000
# export TG_DATA_DIR=/home/tigergraph/sf${SF}
# export TG_PARAMETER=/home/tigergraph/parameters-sf${SF}

nohup ./k8s/benchmark.sh > log.benchmark 2>&1 < /dev/null &
tail -f log.benchmark
```

Listing 4.1: Script to execute the benchmark on TigerGraph for SF10000

5 Performance Results

5.1 TigerGraph performance results

Table 5.1: Summary of results for TigerGraph on scale factor 10000

Benchmark duration	Power@SF	Power@SF/$	Throughput@SF	Throughput@SF/$
643.50 minutes	89 444.50	11.36	41 025.76	5.21

Table 5.2: Detailed **power test results** for TigerGraph on scale factor 10000. Execution times are reported in seconds.

Query	Count	Min.	Max.	Mean	P_{50}	P_{90}	P_{95}	P_{99}
1	30	5.414	6.616	6.015	5.975	6.463	6.591	6.616
2a	30	11.182	33.855	19.399	15.087	31.684	33.716	33.855
2b	30	7.209	12.893	9.883	9.811	11.507	12.742	12.893
3	30	9.592	48.787	16.580	13.841	21.916	30.870	48.787
4	30	4.784	5.161	4.914	4.870	5.065	5.085	5.161
5	30	6.088	7.129	6.574	6.549	6.872	6.981	7.129
6	30	6.593	8.787	7.298	7.305	7.564	7.725	8.787
7	30	21.499	24.015	22.876	22.776	23.827	23.997	24.015
8a	30	6.969	9.422	7.714	7.645	8.049	8.750	9.422
8b	30	3.299	3.904	3.538	3.494	3.750	3.859	3.904
9	30	13.915	15.858	15.162	15.358	15.609	15.699	15.858
10a	30	19.861	43.339	24.830	24.424	27.223	30.291	43.339
10b	30	4.048	14.621	8.849	8.835	12.837	14.352	14.621
11	30	6.843	9.592	7.747	7.759	8.351	8.524	9.592
12	30	12.349	18.878	15.513	17.222	18.507	18.557	18.878
13	30	48.326	50.221	49.365	49.226	50.124	50.201	50.221
14a	30	26.595	29.811	27.684	27.596	28.041	29.156	29.811
14b	30	12.824	15.163	13.519	13.475	13.914	14.283	15.163
15a	30	37.833	41.233	39.067	39.007	39.898	41.231	41.233
15b	30	38.020	180.580	110.354	115.688	147.722	168.314	180.580
16a	30	21.448	55.294	30.765	29.120	36.243	46.640	55.294
16b	30	4.300	10.035	6.814	6.315	9.138	9.539	10.035
17	30	17.715	21.714	19.929	20.119	21.401	21.535	21.714
18	30	64.729	74.026	67.959	67.318	70.291	71.913	74.026
19a	30	6.512	10.137	9.105	9.770	10.033	10.080	10.137
19b	30	6.931	10.296	9.732	9.914	10.179	10.238	10.296
20a	30	0.895	6.063	1.818	0.925	3.472	4.958	6.063
20b	30	1.288	3.901	1.959	1.932	2.191	2.500	3.901

Table 5.3: Operations in the **power test** for TigerGraph on scale factor 10000. Execution times are reported in seconds. Root Post precomputations are performed for each Comment insertion and deletion operation, therefore, they are reported as part of the writes.

Operation	Time
reads	16 949.254
writes	1 874.439
q4precomputation	147.878
q6precomputation	250.388
q19precomputation	914.878
q20precomputation	48.462

6 Validation of the Results

The results were cross-validated against the Umbra reference implementation[1] on scale factor 10, using Umbra version `664890d7f`. Umbra is an in-memory relational database management system developed at the Technische Universität München. The queries of the BI workload are implemented using PostgreSQL-compatible SQL queries that use the `WITH RECURSIVE` clause to implement graph traversal operations.

Listing 6.1: Output of the Umbra–TigerGraph cross-validation command

```
$ export SF=10
$ scripts/cross-validate.sh umbra tigergraph

+++  Files "umbra/output/output-sf10/results.csv" and "tigergraph/output/output-sf10/results.csv" are equal
```

[1] https://github.com/ldbc/ldbc_snb_bi/tree/a2d3ac18c946d6a698c6aa5e6cf5d8954296be63/umbra

7 Supplementary Materials

Table 7.1: Supplementary materials

File or Directory	Purpose
`umbra/output/output-sf10`	Output of the Umbra reference implementation
`parameters/parameters-sf{10,10000}.tar`	Query substitution parameters
`scoring/calculate-scores.py`	Python script to calculate the scores of the benchmark run
`tigergraph/output/output-sf{10,10000}`	Benchmark logs and outputs
`ldbc_snb_bi-1.0.3.tar.gz`	Benchmark driver and reference implementations
`ldbc_snb_datagen_spark-0.5.1.tar.gz`	Data generator
`ldbc_snb_specification-2.2.0.pdf`	Benchmark specification
`tigergraph-3.7.0-offline.tar.gz`	Installer of Tigergraph

The `ldbc_snb_bi_tigergraph_sf10000_attachments` folder's directory structure is as follows:

```
ldbc_snb_bi_tigergraph_sf10000_attachments
├── umbra
│   └── output
│       └── output-sf10
│           └── results.csv
├── parameters
│   ├── parameters-sf10.tar
│   └── parameters-sf10000.tar
├── scoring
│   └── calculate-scores.py
├── tigergraph
│   └── output
│       ├── output-sf10
│       │   ├── benchmark.csv
│       │   ├── load.csv
│       │   ├── results.csv
│       │   └── timings.csv
│       └── output-sf10000
│           ├── benchmark.csv
│           ├── load.csv
│           ├── log.benchmark
│           ├── results.csv
│           └── timings.csv
├── ldbc_snb_bi-1.0.3.tar.gz
├── ldbc_snb_datagen_spark-0.5.1.tar.gz
├── ldbc_snb_specification-2.2.0.pdf
└── tigergraph-3.7.0-offline.tar.gz
```

A CPU and Memory details

Listing A.1: Output of the `cat /proc/cpuinfo` command for a single CPU core

```
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 25
model           : 1
model name      : AMD EPYC 7R13 Processor
stepping        : 1
microcode       : 0xa001173
cpu MHz         : 1803.263
cache size      : 512 KB
physical id     : 0
siblings        : 32
core id         : 0
cpu cores       : 16
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 16
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr
 sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm constant_tsc rep_good nopl nonstop_tsc cpuid
 extd_apicid aperfmperf tsc_known_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes
  xsave avx f16c rdrand hypervisor lahf_lm cmp_legacy cr8_legacy abm sse4a misalignsse 3dnowprefetch topoext
 invpcid_single ssbd ibrs ibpb stibp vmmcall fsgsbase bmi1 avx2 smep bmi2 erms invpcid rdseed adx smap
 clflushopt clwb sha_ni xsaveopt xsavec xgetbv1 clzero xsaveerptr rdpru wbnoinvd arat npt nrip_save vaes
 vpclmulqdq rdpid
bugs            : sysret_ss_attrs null_seg spectre_v1 spectre_v2 spec_store_bypass
bogomips        : 5299.97
TLB size        : 2560 4K pages
clflush size    : 64
cache_alignment : 64
address sizes   : 48 bits physical, 48 bits virtual
power management:
```

Listing A.2: Output of the `lscpu` command

```
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              32
On-line CPU(s) list: 0-31
Thread(s) per core:  2
Core(s) per socket:  16
Socket(s):           1
NUMA node(s):        2
Vendor ID:           AuthenticAMD
CPU family:          25
Model:               1
Model name:          AMD EPYC 7R13 Processor
Stepping:            1
CPU MHz:             1502.144
BogoMIPS:            5299.97
Hypervisor vendor:   KVM
Virtualization type: full
```

```
  L1d cache:           32K
  L1i cache:           32K
  L2 cache:            512K
  L3 cache:            32768K
  NUMA node0 CPU(s):   0-7,16-23
  NUMA node1 CPU(s):   8-15,24-31
  Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr
     sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm constant_tsc rep_good nopl nonstop_tsc cpuid
    extd_apicid aperfmperf tsc_known_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes
     xsave avx f16c rdrand hypervisor lahf_lm cmp_legacy cr8_legacy abm sse4a misalignsse 3dnowprefetch topoext
    invpcid_single ssbd ibrs ibpb stibp vmmcall fsgsbase bmi1 avx2 smep bmi2 erms invpcid rdseed adx smap
    clflushopt clwb sha_ni xsaveopt xsavec xgetbv1 clzero xsaveerptr rdpru wbnoinvd arat npt nrip_save vaes
    vpclmulqdq rdpid
```

Listing A.3: Output of the `dmidecode -t processor` command

```
# dmidecode 3.2
Getting SMBIOS data from sysfs.
SMBIOS 2.7 present.

Handle 0x0004, DMI type 4, 42 bytes
Processor Information
  Socket Designation: CPU 0
  Type: Central Processor
  Family: Zen
  Manufacturer: Advanced Micro Devices, Inc.
  ID: 11 0F A0 00 FF FB 8B 17
  Signature: Family 25, Model 1, Stepping 1
  Flags:
    FPU (Floating-point unit on-chip)
    VME (Virtual mode extension)
    DE (Debugging extension)
    PSE (Page size extension)
    TSC (Time stamp counter)
    MSR (Model specific registers)
    PAE (Physical address extension)
    MCE (Machine check exception)
    CX8 (CMPXCHG8 instruction supported)
    APIC (On-chip APIC hardware supported)
    SEP (Fast system call)
    MTRR (Memory type range registers)
    PGE (Page global enable)
    MCA (Machine check architecture)
    CMOV (Conditional move instruction supported)
    PAT (Page attribute table)
    PSE-36 (36-bit page size extension)
    CLFSH (CLFLUSH instruction supported)
    MMX (MMX technology supported)
    FXSR (FXSAVE and FXSTOR instructions supported)
    SSE (Streaming SIMD extensions)
    SSE2 (Streaming SIMD extensions 2)
    HTT (Multi-threading)
  Version: AMD EPYC 7R13 Processor
  Voltage: 1.1 V
  External Clock: 100 MHz
  Max Speed: 3725 MHz
  Current Speed: 2650 MHz
  Status: Populated, Enabled
```

```
Upgrade: Socket SP3
L1 Cache Handle: 0x0005
L2 Cache Handle: 0x0006
L3 Cache Handle: 0x0007
Serial Number: Not Specified
Asset Tag: Not Specified
Part Number: Not Specified
Core Count: 16
Core Enabled: 16
Thread Count: 32
Characteristics:
  64-bit capable
  Multi-Core
  Hardware Thread
  Execute Protection
```

Listing A.4: Output of the `cat /proc/meminfo` command

```
  MemTotal:       258586064 kB
  MemFree:        256575764 kB
  MemAvailable:   256563876 kB
  Buffers:             2704 kB
  Cached:           1407712 kB
  SwapCached:             0 kB
  Active:            269588 kB
  Inactive:         1176892 kB
  Active(anon):         256 kB
  Inactive(anon):     36360 kB
  Active(file):      269332 kB
  Inactive(file):   1140532 kB
  Unevictable:            0 kB
  Mlocked:                0 kB
  SwapTotal:              0 kB
  SwapFree:               0 kB
  Dirty:                628 kB
  Writeback:              0 kB
  AnonPages:          36372 kB
  Mapped:             50336 kB
  Shmem:                524 kB
  KReclaimable:       85604 kB
  Slab:              200248 kB
  SReclaimable:       85604 kB
  SUnreclaim:        114644 kB
  KernelStack:         6400 kB
  PageTables:          4276 kB
  NFS_Unstable:           0 kB
  Bounce:                 0 kB
  WritebackTmp:           0 kB
  CommitLimit:    129293032 kB
  Committed_AS:      282916 kB
  VmallocTotal:   34359738367 kB
  VmallocUsed:       228772 kB
  VmallocChunk:           0 kB
  Percpu:              9216 kB
  HardwareCorrupted:      0 kB
  AnonHugePages:          0 kB
  ShmemHugePages:         0 kB
  ShmemPmdMapped:         0 kB
```

```
    FileHugePages:         0 kB
    FilePmdMapped:         0 kB
    HugePages_Total:       0
    HugePages_Free:        0
    HugePages_Rsvd:        0
    HugePages_Surp:        0
    Hugepagesize:       2048 kB
    Hugetlb:               0 kB
    DirectMap4k:      145316 kB
    DirectMap2M:    11003904 kB
    DirectMap1G:    252706816 kB
```

Listing A.5: Output of the `lshw -C memory` command

```
*-memory
    description: System Memory
    physical id: 8
    slot: System board or motherboard
    size: 256GiB
  *-bank:0
       description: DIMM DDR4 Static column Pseudo-static Synchronous Window DRAM 3200 MHz (0.3 ns)
       physical id: 0
       size: 128GiB
       width: 64 bits
       clock: 3200MHz (0.3ns)
  *-bank:1
       description: DIMM DDR4 Static column Pseudo-static Synchronous Window DRAM 3200 MHz (0.3 ns)
       physical id: 1
       size: 128GiB
       width: 64 bits
       clock: 3200MHz (0.3ns)
```

B IO PERFORMANCE

Listing B.1: Output of the `fio` command

```
iotest: (g=0): rw=write, bs=4K-4K/4K-4K/4K-4K, ioengine=sync, iodepth=1
fio-2.14
Starting 1 process
Jobs: 1 (f=1): [W(1)] [100.0% done] [0KB/6056KB/0KB /s] [0/1514/0 iops] [eta 00m:00s]
iotest: (groupid=0, jobs=1): err= 0: pid=11001: Sun Mar 19 10:08:27 2023
  write: io=2048.0MB, bw=6298.6KB/s, iops=1574, runt=332957msec
    clat (usec): min=322, max=14022, avg=632.71, stdev=238.08
     lat (usec): min=322, max=14022, avg=632.83, stdev=238.08
    clat percentiles (usec):
     |  1.00th=[  386],  5.00th=[  418], 10.00th=[  442], 20.00th=[  474],
     | 30.00th=[  510], 40.00th=[  540], 50.00th=[  580], 60.00th=[  628],
     | 70.00th=[  668], 80.00th=[  732], 90.00th=[  852], 95.00th=[ 1020],
     | 99.00th=[ 1576], 99.50th=[ 1816], 99.90th=[ 2640], 99.95th=[ 3184],
     | 99.99th=[ 4640]
    lat (usec) : 500=27.35%, 750=54.92%, 1000=12.41%
    lat (msec) : 2=5.00%, 4=0.29%, 10=0.02%, 20=0.01%
  cpu          : usr=0.68%, sys=2.01%, ctx=524289, majf=0, minf=12
  IO depths    : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     issued    : total=r=0/w=524288/d=0, short=r=0/w=0/d=0, drop=r=0/w=0/d=0
     latency   : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
  WRITE: io=2048.0MB, aggrb=6298KB/s, minb=6298KB/s, maxb=6298KB/s, mint=332957msec, maxt=332957msec

Disk stats (read/write):
  nvme0n1: ios=0/524319, merge=0/20, ticks=0/326632, in_queue=326632, util=100.00%
```

Listing B.2: Output of the `lshw -class network` command

```
  *-network
       description: Ethernet interface
       product: Elastic Network Adapter (ENA)
       vendor: Amazon.com, Inc.
       physical id: 5
       bus info: pci@0000:00:05.0
       logical name: eth0
       version: 00
       serial: 0e:ea:ef:75:27:3f
       width: 32 bits
       clock: 33MHz
       capabilities: pciexpress msix bus_master cap_list ethernet physical
       configuration: broadcast=yes driver=ena driverversion=2.8.3g ip=10.0.1.17 latency=0 link=yes multicast=yes
       resources: irq:0 memory:febf4000-febf5fff memory:febf6000-febf7fff memory:fe800000-fe8fffff
```

C Dataset generation instructions

The datasets can be generated using the LDBC SNB Datagen. To regenerate the data sets used in this benchmark, build the Datagen JAR as described in the project's README, configure the AWS EMR environment, upload the JAR to the S3 bucket (denoted as `${BUCKET_NAME}`) and run the following commands to generate the datasets used in this audit.
Note that while the datasets for TigerGraph were generated as gzip-compressed archives, they are decompressed during preprocessing. Decompressing the SF10000 data set took 409 seconds (wall clock) on the first node (48 nodes total), when performed by the following command on each node: `time find /home/tigergraph/sf10000 -name "*.csv.gz" -print0 | parallel -q0 gunzip`

Listing C.1: Script to generate the SF10 dataset for TigerGraph in AWS EMR. This dataset is only used for cross-validation

```
export SCALE_FACTOR=10
export JOB_NAME=sf${SCALE_FACTOR}-projected-csv-gz

./tools/emr/submit_datagen_job.py \
    --use-spot \
    --bucket ${BUCKET_NAME} \
    --copy-all \
    --az us-east-2c \
    ${JOB_NAME} \
    ${SCALE_FACTOR} \
    csv \
    bi \
    -- \
    --explode-edges \
    --format-options compression=gzip \
    --generate-factors
```

Listing C.2: Script to generate the SF10000 dataset for TigerGraph in AWS EMR. This dataset is used for the benchmark run

```
export SCALE_FACTOR=10000
export JOB_NAME=sf${SCALE_FACTOR}-projected-csv-gz

./tools/emr/submit_datagen_job.py \
    --use-spot \
    --bucket ${BUCKET_NAME} \
    --copy-all \
    --az us-east-2c \
    ${JOB_NAME} \
    ${SCALE_FACTOR} \
    csv \
    bi \
    -- \
    --explode-edges \
    --format-options compression=gzip \
    --generate-factors
```

Listing C.3: Script to generate the SF10 dataset for Umbra locally. This dataset is only used for cross-validation

```
export SCALE_FACTOR=10
export LDBC_SNB_DATAGEN_MAX_MEM=60G
export LDBC_SNB_DATAGEN_JAR=$(sbt -batch -error 'print assembly / assemblyOutputPath')

```

```
tools/run.py \
    --cores $(nproc) \
    --memory ${LDBC_SNB_DATAGEN_MAX_MEM} \
    -- \
    --format csv \
    --scale-factor ${SCALE_FACTOR} \
    --explode-edges \
    --mode bi \
    --output-dir out-sf${SCALE_FACTOR}/ \
    --generate-factors \
    --format-options header=false,quoteAll=true,compression=gzip
```

D Data schema

Listing D.1: Content of the GSQL schema used by TigerGraph

```
## Message
CREATE VERTEX Comment (PRIMARY_ID id UINT, creationDate INT, locationIP STRING, browserUsed STRING, content STRING, length UINT) WITH primary_id_as_attribute="TRUE"
CREATE VERTEX Post (PRIMARY_ID id UINT, imageFile STRING, creationDate INT, locationIP STRING, browserUsed STRING, language STRING, content STRING, length UINT) WITH primary_id_as_attribute="TRUE"
## organisation
CREATE VERTEX Company (PRIMARY_ID id UINT, name STRING, url STRING) WITH primary_id_as_attribute="TRUE"
CREATE VERTEX University (PRIMARY_ID id UINT, name STRING, url STRING) WITH primary_id_as_attribute="TRUE"
## place
CREATE VERTEX City (PRIMARY_ID id UINT, name STRING, url STRING) WITH primary_id_as_attribute="TRUE"
CREATE VERTEX Country (PRIMARY_ID id UINT, name STRING, url STRING) WITH primary_id_as_attribute="TRUE"
CREATE VERTEX Continent (PRIMARY_ID id UINT, name STRING, url STRING) WITH primary_id_as_attribute="TRUE"
## etc
CREATE VERTEX Forum (PRIMARY_ID id UINT, title STRING, creationDate INT,
    maxMember UINT) WITH primary_id_as_attribute="TRUE" // maxMember is for precompute in BI-4
CREATE VERTEX Person (PRIMARY_ID id UINT, firstName STRING, lastName STRING, gender STRING, birthday INT, creationDate INT, locationIP STRING, browserUsed STRING, speaks SET<STRING>, email SET<STRING>,
    popularityScore UINT) WITH primary_id_as_attribute="TRUE" // popularityScore is for precompute in BI-6
CREATE VERTEX Tag (PRIMARY_ID id UINT, name STRING, url STRING) WITH primary_id_as_attribute="TRUE"
CREATE VERTEX TagClass (PRIMARY_ID id UINT, name STRING, url STRING) WITH primary_id_as_attribute="TRUE"


# create edge
CREATE DIRECTED EDGE CONTAINER_OF (FROM Forum, TO Post) WITH REVERSE_EDGE="CONTAINER_OF_REVERSE"
CREATE DIRECTED EDGE HAS_CREATOR (FROM Comment|Post, TO Person) WITH REVERSE_EDGE="HAS_CREATOR_REVERSE"
CREATE DIRECTED EDGE HAS_INTEREST (FROM Person, TO Tag) WITH REVERSE_EDGE="HAS_INTEREST_REVERSE"
CREATE DIRECTED EDGE HAS_MEMBER (FROM Forum, TO Person, creationDate INT) WITH REVERSE_EDGE="HAS_MEMBER_REVERSE"
CREATE DIRECTED EDGE HAS_MODERATOR (FROM Forum, TO Person) WITH REVERSE_EDGE="HAS_MODERATOR_REVERSE"
CREATE DIRECTED EDGE HAS_TAG (FROM Comment|Post|Forum, TO Tag) WITH REVERSE_EDGE="HAS_TAG_REVERSE"
CREATE DIRECTED EDGE HAS_TYPE (FROM Tag, TO TagClass) WITH REVERSE_EDGE="HAS_TYPE_REVERSE"
CREATE DIRECTED EDGE IS_LOCATED_IN (FROM Company, TO Country | FROM Person, TO City | FROM University, TO City) WITH REVERSE_EDGE="IS_LOCATED_IN_REVERSE"
CREATE DIRECTED EDGE MESG_LOCATED_IN (FROM Comment, TO Country | FROM Post, TO Country) // Reverse edge of Comment/Post -IS_Located_IN-> Country will cause Country connected by too many edges, which makes loading slow
CREATE DIRECTED EDGE IS_PART_OF (FROM City, TO Country | FROM Country, TO Continent) WITH REVERSE_EDGE="IS_PART_OF_REVERSE"
CREATE DIRECTED EDGE IS_SUBCLASS_OF (FROM TagClass, TO TagClass) WITH REVERSE_EDGE="IS_SUBCLASS_OF_REVERSE"
CREATE UNDIRECTED EDGE KNOWS (FROM Person, TO Person, creationDate INT, weight19 UINT, weight20 UINT DEFAULT 10000)
CREATE DIRECTED EDGE LIKES (FROM Person, TO Comment|Post, creationDate INT) WITH REVERSE_EDGE="LIKES_REVERSE"
CREATE DIRECTED EDGE REPLY_OF (FROM Comment, TO Comment|Post) WITH REVERSE_EDGE="REPLY_OF_REVERSE"
CREATE DIRECTED EDGE STUDY_AT (FROM Person, TO University, classYear INT) WITH REVERSE_EDGE="STUDY_AT_REVERSE"
CREATE DIRECTED EDGE WORK_AT (FROM Person, TO Company, workFrom INT) WITH REVERSE_EDGE="WORK_AT_REVERSE"

CREATE DIRECTED EDGE ROOT_POST (FROM Comment, TO Post) WITH REVERSE_EDGE="ROOT_POST_REVERSE" //FOR BI-3,9,17
CREATE DIRECTED EDGE REPLY_COUNT (FROM Person, TO Person, cnt UINT)

CREATE GLOBAL SCHEMA_CHANGE JOB addIndex {
  ALTER VERTEX Country ADD INDEX country_name ON (name);
  ALTER VERTEX Company ADD INDEX company_name ON (name);
  ALTER VERTEX University ADD INDEX university_name ON (name);
  ALTER VERTEX Tag ADD INDEX tag_name ON (name);
  ALTER VERTEX TagClass ADD INDEX tagclass_name ON (name);
```

```
}

RUN GLOBAL SCHEMA_CHANGE JOB addIndex
CREATE GRAPH ldbc_snb (*)
```