New Performance Report Preview

NEXMark

NEXMark is a popular benchmark for stream processing. A few systems have tested against it and have public numbers such as https://github.com/nexmark/nexmark, so we together with our audience have more references. In the near future, we will have `TPC-H` and also modified `TPC-H` queries to make the workload even more intensive to test RisingWave and other stream processing systems, stay tuned.

We built our own data generator: https://github.com/risingwavelabs/nexmark-bench. Details can be found in the README.md.

Hardware Setting:

RisingWave affinity:

```
Compute Node and Compactor Node shared the same c5.2xlarge EC2 instance(8vCPUs 16 GB memory).
Meta Node and Front Node has no impact on the performance. They share another instance.
Storage: S3 only

RisingWave will support file cache on EBS in the future. This is NOT enabled now.
```

Flink:

```
Task Manager is on a c5.2xlarge EC2 instance.
Job Manager has no impact on the performance. It occupies another instance.

Storage:
RocksDB backend with EBS gp3 of size 100GiB.
We use the better configuration gp3(12000 IOPS, 500 MB/s Bandwidth)
```

System Configuration:

1. RisingWave has no special configuration. All use the default ones: https://github.com/risingwavelabs/risingwave/blob/main/src/common/src/config.rs
2. Flink(v1.16.0):

```
flink-conf.yaml: |+
    execution:
```

```
  planner: blink
  type: streaming
  time-characteristic: event-time
  periodic-watermarks-interval: 200
  result-mode: table
  max-table-result-rows: 1000000
  parallelism: 8
  max-parallelism: 128
  min-idle-state-retention: 0
  max-idle-state-retention: 0
  current-catalog: default_catalog
  current-database: default_database
  restart-strategy:
    type: fallback

# JVM options for GC
env.java.opts: -verbose:gc -XX:NewRatio=3 -XX:+PrintGCDetails -Xlog:gc* -XX:ParallelGCThreads=4

# Restart strategy related configuration
restart-strategy: fixed-delay
restart-strategy.fixed-delay.attempts: 2147483647
restart-strategy.fixed-delay.delay: 10s

# Max task attempts to retain in JM
jobmanager.execution.attempts-history-size: 100

# Maximum backoff time (ms) for partition requests of input channels.
taskmanager.network.request-backoff.max: 30000
jobmanager.rpc.address: flink-jobmanager
blob.server.port: 6124
jobmanager.rpc.port: 6123
taskmanager.rpc.port: 6122
queryable-state.proxy.ports: 6125
parallelism.default: 8
taskmanager.numberOfTaskSlots: 8
jobmanager.memory.process.size: 15G
taskmanager.memory.process.size: 15G
taskmanager.memory.managed.fraction: 0.4
taskmanager.network.memory.floating-buffers-per-gate: 256

# The number of buffers available for each external blocking channel.
# Will change it to be the default value later.
taskmanager.network.memory.buffers-per-external-blocking-channel: 16
# The maximum number of concurrent requests in the reduce-side tasks.
# Will change it to be the default value later.
task.external.shuffle.max-concurrent-requests: 512
# Whether to enable compress shuffle data when using external shuffle.
# Will change it to be the default value later.
task.external.shuffle.compression.enable: true
table.exec.mini-batch.enabled: true
table.exec.mini-batch.allow-latency: 2s
table.exec.mini-batch.size: 50000
table.optimizer.distinct-agg.split.enabled: true
pipeline.object-reuse: false
execution.checkpointing.mode: EXACTLY_ONCE
execution.checkpointing.interval: 60000
execution.checkpointing.max-concurrent-checkpoints: 1
# disable final checkpoint to avoid test waiting for the last checkpoint complete
execution.checkpointing.checkpoints-after-tasks-finish.enabled: true
```

```
    io.tmp.dirs: /opt/flink/tmp
    state.backend: rocksdb
    state.checkpoints.dir: s3://flink-bench-checkpoints/flink-nexmark-0-15-pc-test-better-2811
    state.backend.incremental: true
    state.backend.local-recovery: true
    state.backend.rocksdb.block.blocksize: 4KB
    state.backend.rocksdb.thread.num: 2
    state.backend.rocksdb.writebuffer.count: 2
    state.backend.rocksdb.writebuffer.number-to-merge: 1
    state.backend.rocksdb.compaction.level.use-dynamic-size: false
    state.backend.rocksdb.compaction.level.target-file-size-base: 64MB
    state.backend.rocksdb.use-bloom-filter: false

    # akka configs
    akka.ask.timeout: 120s
    akka.watch.heartbeat.interval: 10s
    akka.framesize: 102400kB
    fs.s3a.endpoint: s3.us-east-1.amazonaws.com
    metrics.reporter.prom.factory.class: org.apache.flink.metrics.prometheus.PrometheusReporterFactory
    metrics.reporter.prom.port: 9249
```

Workload Setting:

1. Generate 2B Nexmark events by https://github.com/risingwavelabs/nexmark-bench into a single Kafka topic with 8 partitions before starting RisingWave or Flink. The events are in JSON data format.
2. Start processing each query one after another.
    a. Start time: when the throughput becomes non-zero.
    b. End time: when the throughput becomes zero again.
    c. If the query cannot finish within 1 hour, we kill it and calculate average throughput based on the period of 1 hour. If the query finish within 1 hour, then we calculate average throughput based on the period from start time to end time.
3. After each query, we restart the systems and clean all the legacy data on S3 to make sure the previous one cannot interfere with the next query in any way.
4. Both RisingWave and Flink use `kafka` connector as the source and `blackhole` connector as the sink. `blackhole` sink means that all the results output by the streaming queries are simply discarded. They are neither output to a downstream system nor stored in the materialized view of RisingWave.

Test number

Aa Nexmark Query	# Throughput(kr/s)	# CPU(compute+compactor,%)	# Duration(mins)	# Kafka events(million)	≡ S3 Size
q0	1147.67	674.35	29	2000	NA
q1	1186.08	707.56	28	2000	NA
q2	1186.77	691.79	28	2000	NA
q3	1007.86	740.77	33	2000	2.5 GiB
q4	188.91	754.68	60	680.08	21.6 GiB
q5	88.57	783.82	60	318.85	12.9 GiB
q5-rewrite	106.77	760.73	60	384.37	33.3 GiB
q6	117.37	736.4	60	422.53	20.0 GiB
q7	296.28	790.48	60	1066.61	47.0 GiB
q7-rewrite	1038.02	716.46	32	2000	NA
q8	627.02	744.08	53	2000	5.9 GiB
q9	154.88	730.52	60	557.57	31.3 GiB
q10-blackhole	1072.71	716.08	31	2000	NA
q12	722.49	787.59	46	2000	3.1 GiB
q13	875.2	745.75	38	2000	NA
q14	1105.88	717.94	30	2000	NA
q15	402.85	310.56	60	1450.26	7.4 GiB
q16	85.2	717.83	60	306.72	20.7 GiB
q17	329.52	772.17	60	1186.27	8.9 GiB
q18	188.22	729.4	60	677.59	21.9 GiB
q19	116.01	690.56	60	417.64	110.7 GiB

Aa Nexmark Query	# Throughput(kr/s)	# CPU(compute+compactor,%)	# Duration(mins)	# Kafka events(million)	≡ S3 Size
q20	142.23	783.73	60	512.03	66.0 GiB
q21	898.53	770.59	37	2000	NA
q22	1070.55	714.03	31	2000	NA

Aa Nexmark Query	# Throughput(kr/s)	# CPU(taskmanager, %)	# Duration(mins)	# Kafka events(million)	≡ S3 Size	≡ EBS Size
q0	856.48	677.98	39	2000	NA	NA
q1	836.7	696.97	40	2000	NA	NA
q2	900.84	677.41	37	2000	NA	NA
q3	776.79	700.55	43	2000	260.6 MiB	424M
q4	116.22	673.92	60	418.39	5.1 GiB	7.1G
q5	147.51	670.06	60	531.04	1.6 GiB	2.1G
q5-rewrite	150.16	662.2	60	540.58	1.6 GiB	2.6G
q7	35.82	254.31	60	128.95	10.7 GiB	15G
q8	724.6	712.42	46	2000	1.7 GiB	2.7G
q9	48.77	437.16	60	175.57	24.7 GiB	27G
q10-blackhole	813.06	689.04	41	2000	NA	NA
q11	301.13	767.3	60	1084.07	734.6 MiB	1.7G
q12	370.92	770.86	60	1335.31	32.1 MiB	218M
q13	654.67	756.81	51	2000	NA	NA
q14	759.02	701.04	44	2000	NA	NA

Aa Nexmark Query	# Throughput(kr/s)	# CPU(taskmanager, %)	# Duration(mins)	# Kafka events(million)	≡ S3 Size	≡ EBS Size
q15	339.56	769.47	60	1222.42	487.2 MiB	777M
q16	52.78	472.05	60	190.01	1.8 GiB	2.5G
q17	355.36	695.22	60	1279.3	3.5 GiB	4.3G
q18	44.78	135.8	60	161.21	6.4 GiB	7.0G
q19	89.05	419.76	60	320.58	18.8 GiB	24G
q20	58.62	510.2	60	211.03	27.9 GiB	27G
q21	515.5	783.85	60	1855.8	NA	NA
q22	667.77	753.41	50	2000	NA	NA