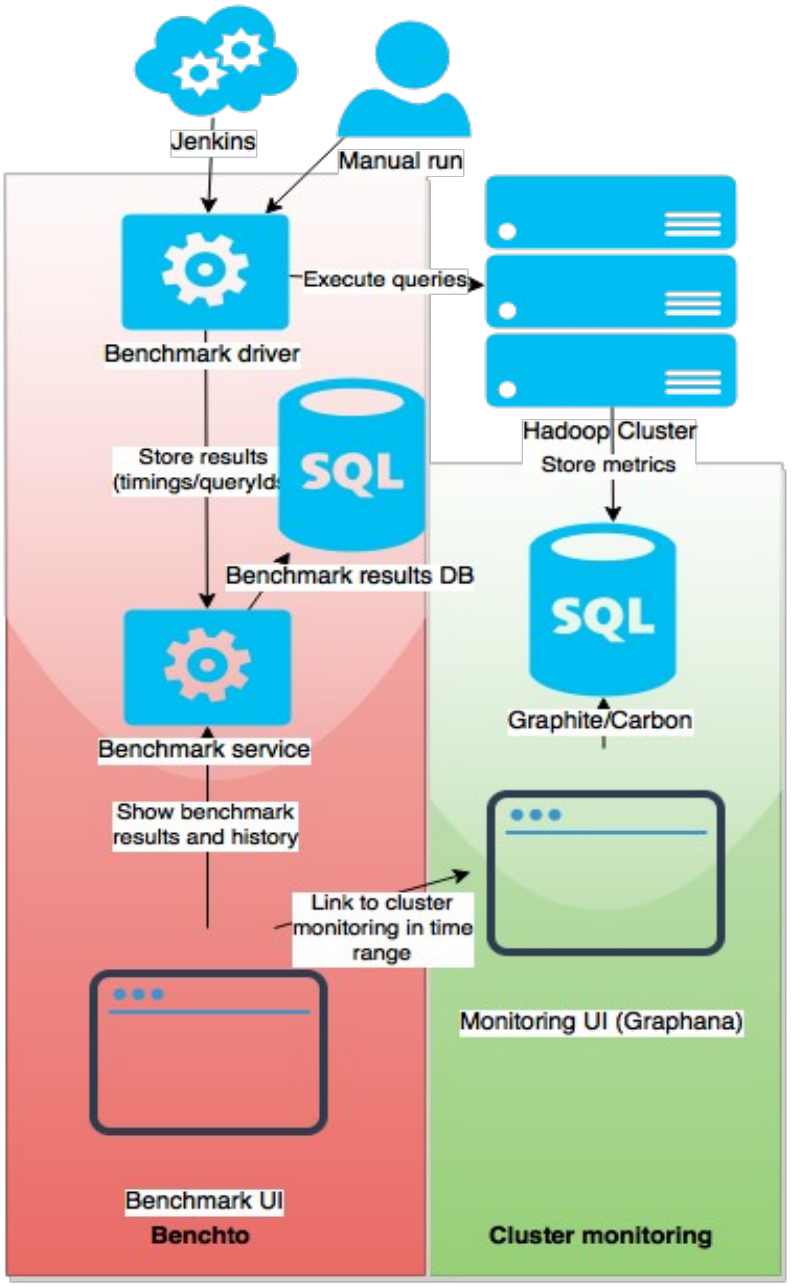
Benchto

macro benchmarking framework

TERADATA

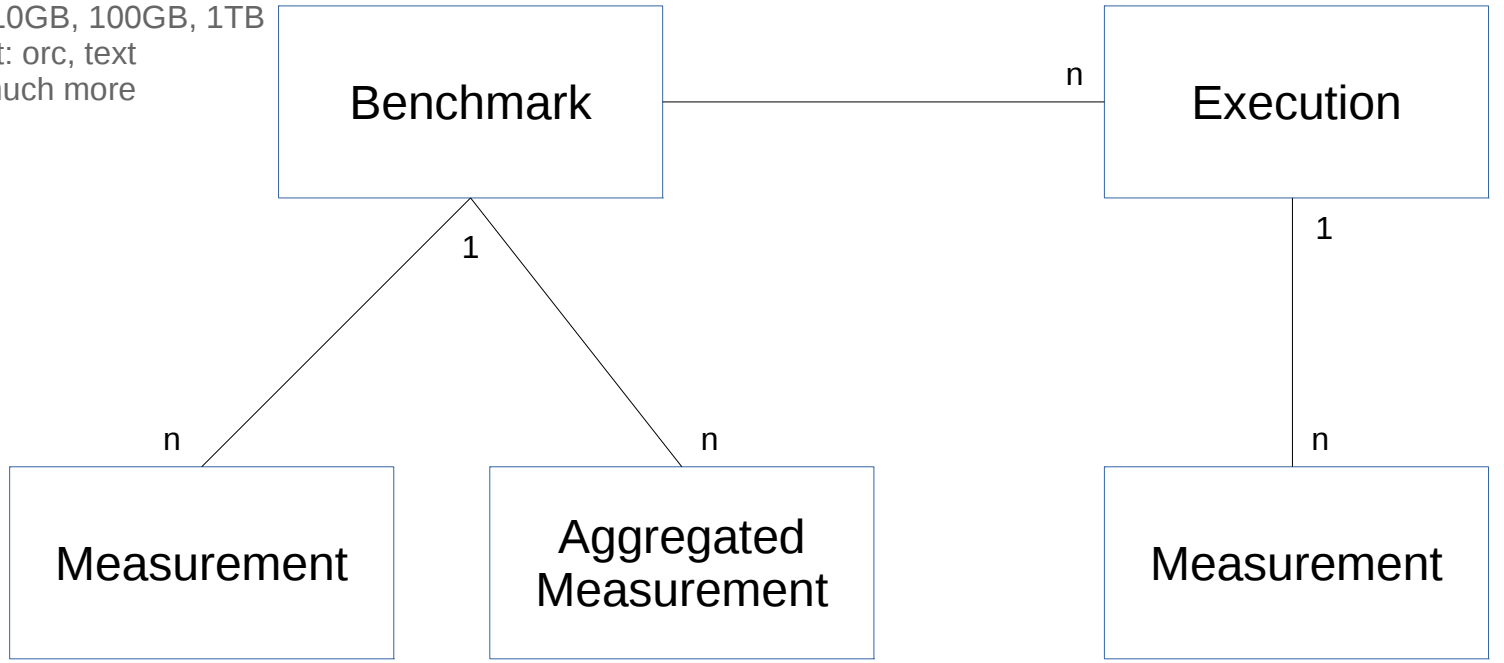
High level architecture



TERADATA

Benchmarks – model

Benchmark, e.g.: query 1 from TPCH
Multiple variables sets:
- size: 10GB, 100GB, 1TB
- format: orc, text
- and much more



Benchmark consists of multiple execution (queries)
Executions are identified by Sequence number: 0, 1, ...

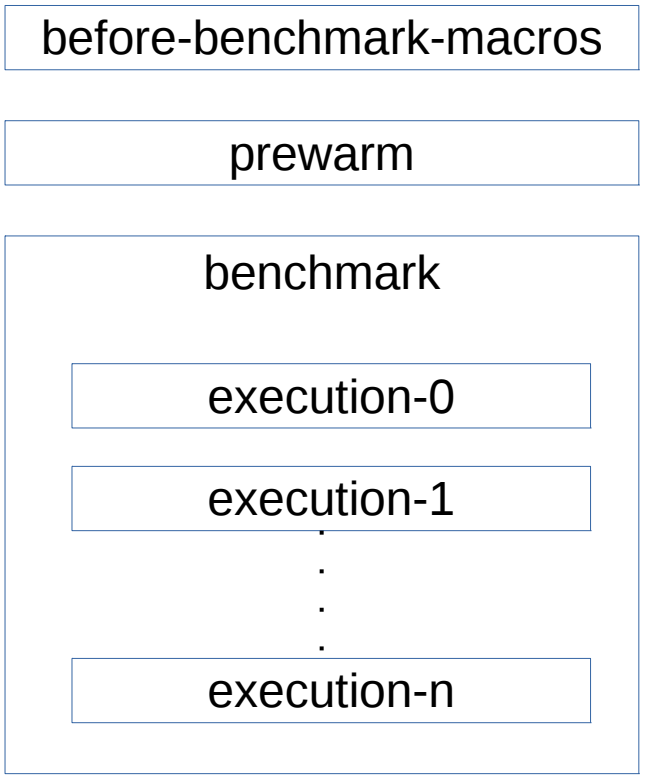
Measurements per execution:
- duration
- CPU/memory/network usage

Benchmark level measurements – especially important for concurrency benchmarks, e.g.: throughput.

Aggregated measurements for all executions of benchmark. For each measurement min, max, mean and stdDev are calculated.

TERADATA

Benchmarks – execution



Execution of custom scripts before benchmark execution, e.g. dropping caches

If needed, prewarm queries can be run several times before benchmark run

Benchmark consists of multiple executions, which are run by specified number of workers (1 for sequential run and >1 for concurrency tests).

Life-cycle of benchmark/execution is reflected immediately in *benchto-service* and can be easily tracked during execution. After execution of benchmark or event, all measurements are gathered and stored in service.

It takes some time for cluster measurements to be stored in Graphite, so extra stops between benchmarks and queries are needed.

after-benchmark-macros

Execution of custom scripts after benchmark execution, e.g. removing created tables

TERADATA

Defining benchmarks - structure

- Convention based defining of benchmark through descriptors (YAML format) and query SQL files

```
$ tree .
.
├── application-cdh.yaml
├── application-td-hdp.yaml
├── benchmarks
│   ├── presto
│   │   ├── concurrency-insert.yaml
│   │   ├── concurrency.yaml
│   │   ├── linear-scan.yaml
│   │   └── tpch.yaml
│   └── querygrid-presto-ansi
│       └── concurrency.yaml
└── sql
    ├── presto
    │   ├── linear-scan
    │   │   ├── selectivity-0.sql
    │   │   └── selectivity-50.sql
    │   ├── linear-scan-insert
    │   │   ├── create-insert-table.sql
...
```

TERADATA

Defining benchmarks - descriptor

- Descriptor is YAML configuration file with various properties and user defined variables

```
$ cat benchmarks/presto/concurrency.yaml
datasource: presto
query-names: presto/linear-scan/selectivity-${selectivity}.sql
schema: tpch_100gb_orc
database: hive
concurrency: ${concurrency_level}
runs: ${concurrency_level}
prewarm-runs: 3
variables:
  1:
    selectivity: 10, 100
    concurrency_level: 10
  2:
    selectivity: 10, 100
    concurrency_level: 20
  3:
    selectivity: 10, 100
    concurrency_level: 50
```

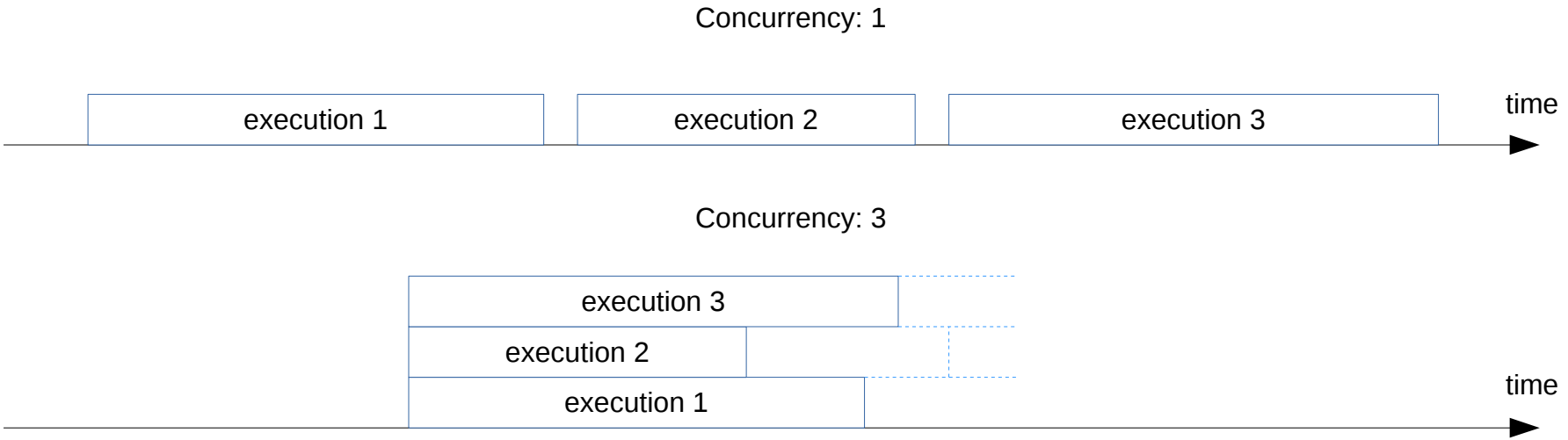
Defining benchmarks – SQL file templating

- SQL files can use keys defined in YAML configuration file – templates are based on FreeMarker

```
$ cat sql/presto/tpch/q1.sql
SELECT
  l.orderkey,
  sum(l.extendedprice * (1 - l.discount)) AS revenue,
  o.orderdate,
  o.shippriority
FROM
  "${database}"."${schema}"."customer" AS c,
  "${database}"."${schema}"."orders" AS o,
  "${database}"."${schema}"."lineitem" AS l
WHERE
  c.mktsegment = 'BUILDING'
  AND c.custkey = o.custkey
  AND l.orderkey = o.orderkey
  AND o.orderdate < DATE '1995-03-15'
  AND l.shipdate > DATE '1995-03-15'
GROUP BY
  l.orderkey,
  o.orderdate,
  o.shippriority
ORDER BY
  revenue DESC,
  o.orderdate
```

Defining benchmarks – sequential vs. concurrent

- There are two main benchmark types: sequential and concurrent



- *concurrency* parameter determines number of workers
- Sequential benchmarks should be used to accurately benchmark execution of particular query
- Concurrent benchmarks can be used to measure throughput and behavior under load from multiple workers

TERADATA

Running benchmarks – maven configuration

- It is most convenient to use maven exec plugin to run driver (JDBC dependencies resolution)

```
    <dependencies>
        <dependency>
            <groupId>com.teradata.benchmark</groupId>
            <artifactId>benchmark-driver</artifactId>
            <version>1.0.0-SNAPSHOT</version>
        </dependency>
        <dependency>
            <groupId>com.facebook.presto</groupId>
            <artifactId>presto-jdbc</artifactId>
        </dependency>
…
           <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
...
                <configuration>
                    <mainClass>com.teradata.benchmark.driver.DriverApp</mainClass>
                    <arguments>
                        <argument>--spring.config.location=classpath:/application-$
{active.spring.environment}.yaml</argument>
                        <argument>--activeBenchmarks=${activeBenchmarks}</argument>
                        <argument>--activeVariables=${activeVariables}</argument>
                    </arguments>
                </configuration>
            </plugin>
```

TERADATA

Running benchmarks – filtering

- You can filter benchmarks by name (contains match)

```
$ mvn -Pbenchmark-hdp package exec:java -DactiveBenchmarks=concurrent_benchmark
09:40:30.373 INFO  [main] c.t.b.driver.loader.BenchmarkLoader - Excluded Benchmarks:
09:40:30.373 INFO  [main] c.t.b.driver.loader.BenchmarkLoader -  | Benchmark Name
09:40:30.373 INFO  [main] c.t.b.driver.loader.BenchmarkLoader -  | simple_select_benchmark
09:40:30.374 INFO  [main] c.t.b.driver.loader.BenchmarkLoader -  | test_benchmark
09:40:30.375 INFO  [main] c.t.b.driver.loader.BenchmarkLoader - Selected Benchmarks:
09:40:30.375 INFO  [main] c.t.b.driver.loader.BenchmarkLoader -  | Benchmark Name
09:40:30.375 INFO  [main] c.t.b.driver.loader.BenchmarkLoader -  | test_concurrent_benchmark
```

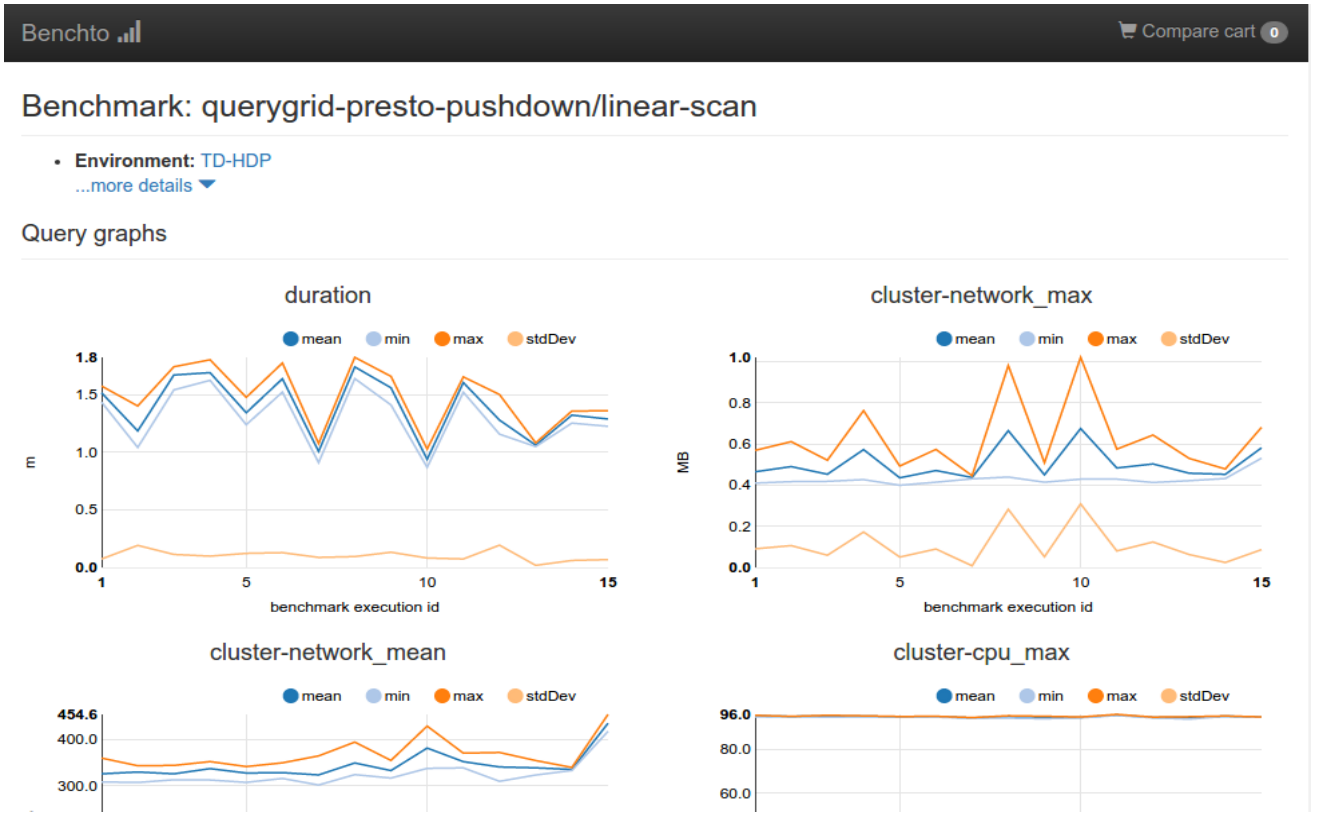
- You can filter benchmarks by variable value

```
$ mvn -Pbenchmark-hdp package exec:java -DactiveVariables=concurrency=2
09:40:30.373 INFO  [main] c.t.b.driver.loader.BenchmarkLoader - Excluded Benchmarks:
09:40:30.373 INFO  [main] c.t.b.driver.loader.BenchmarkLoader -  | Benchmark Name                |
Data Source         | Runs | Prewarms | Concurrency |
09:40:30.373 INFO  [main] c.t.b.driver.loader.BenchmarkLoader -  | simple_select_benchmark      |
test_datasource     | 1    | 1        | 1           |
09:40:30.374 INFO  [main] c.t.b.driver.loader.BenchmarkLoader -  | test_benchmark               |
test_datasource     | 2    | 0        | 1           |
09:40:30.375 INFO  [main] c.t.b.driver.loader.BenchmarkLoader - Selected Benchmarks:
09:40:30.375 INFO  [main] c.t.b.driver.loader.BenchmarkLoader -  | Benchmark Name                |
Data Source         | Runs | Prewarms | Concurrency |
09:40:30.375 INFO  [main] c.t.b.driver.loader.BenchmarkLoader -  | test_concurrent_benchmark    |
test_datasource     | 2    | 0        | 2           |
```

TERADATA

Benchto UI

- Benchto UI is used to visualize benchmarks results
- Linking between tools (Grafana, Presto UI)
- Ability to compare multiple benchmarks



Grafana monitoring

- We use Grafana dashboard with Graphite
- Benchmark/executions life-cycle events are showed on dashboards
- Provides good visibility into state of the cluster