

Load Testing Neo4j

With JMeter & Gatling

David Pecollet
Customer Success Architect

neo4j

Table of content

- Load testing intro
- Apache JMeter 

- Gatling 

neo4j

Load testing introduction



Why?

- Ensure application/service SLA
- Find bottlenecks, limits
- Estimate system sizing



What?

- Simulate users/clients concurrently querying the system
- Realistic/peak/stress workload
- Measure response times

How?

- Tools : JMeter, Gatling

Load testing Neo4j

Testing scope

~~Application E2E~~

Direct database testing ⇐ our focus

Access / Protocol

Cypher over BOLT ⇐

~~Cypher over HTTP API~~

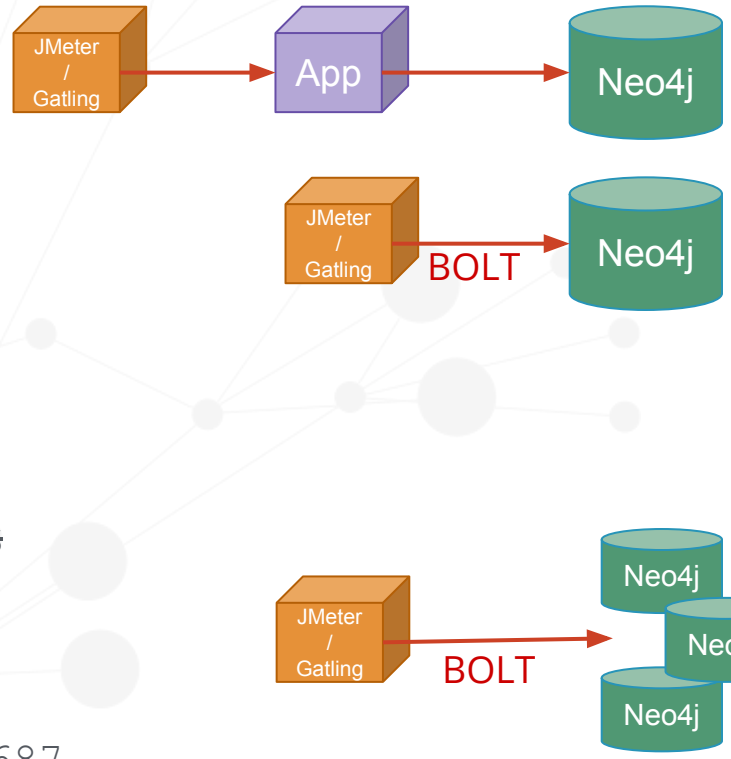
~~Unmanaged extensions HTTP endpoints~~

~~Embedded / java API~~

Standalone instance / cluster

Direct protocol `bolt://host:7687`

Routing protocol `neo4j://cluster:7687`





Apache JMeter

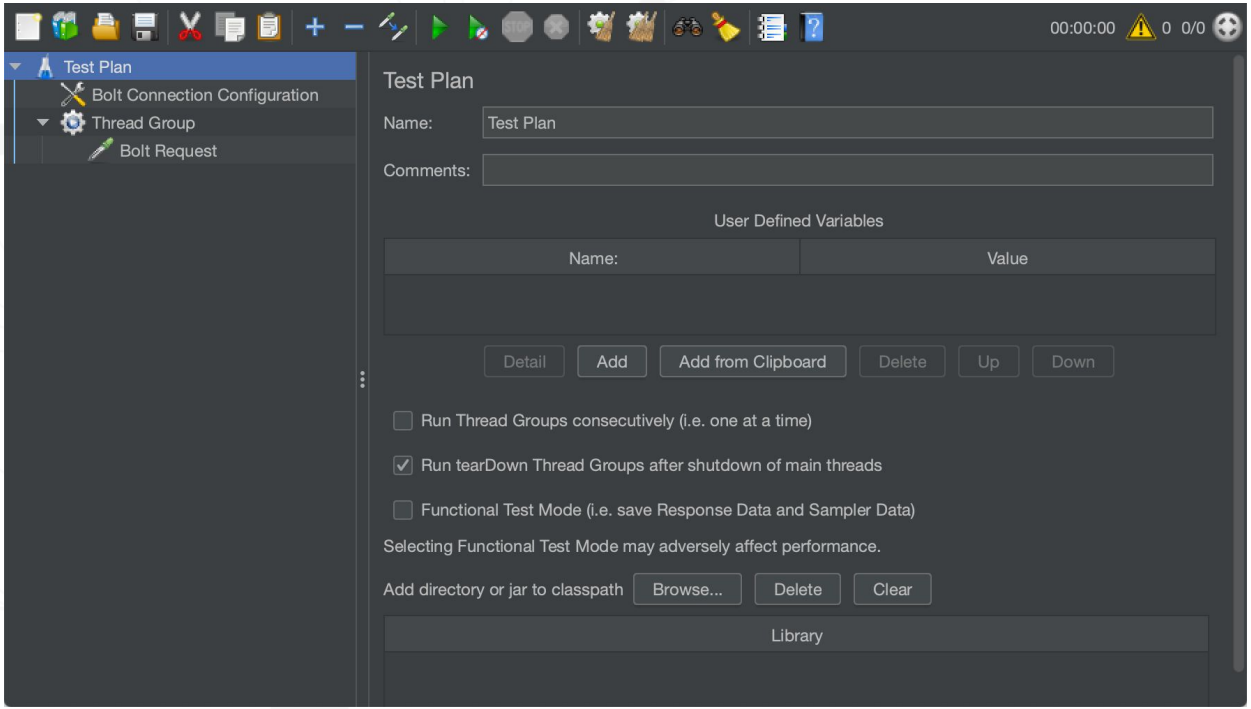


JMeter versions

JMeter 5.2+ comes with Bolt support out of the box.

Upcoming version

- 4.x multi-database
- Cluster access mode

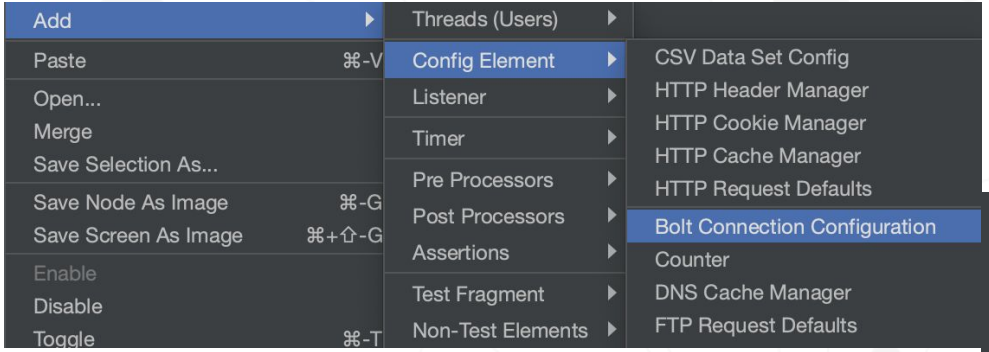


neo4j

Config Element

Name : “Bolt Connection Configuration”

Add it to the Test Plan.



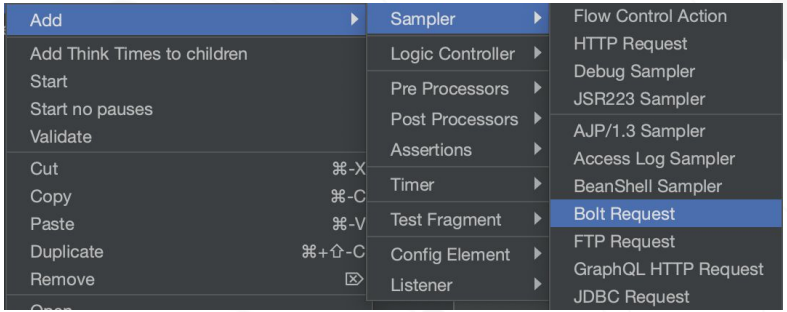


Fill in URL (using neo4j:// scheme for a cluster),
…user, password.

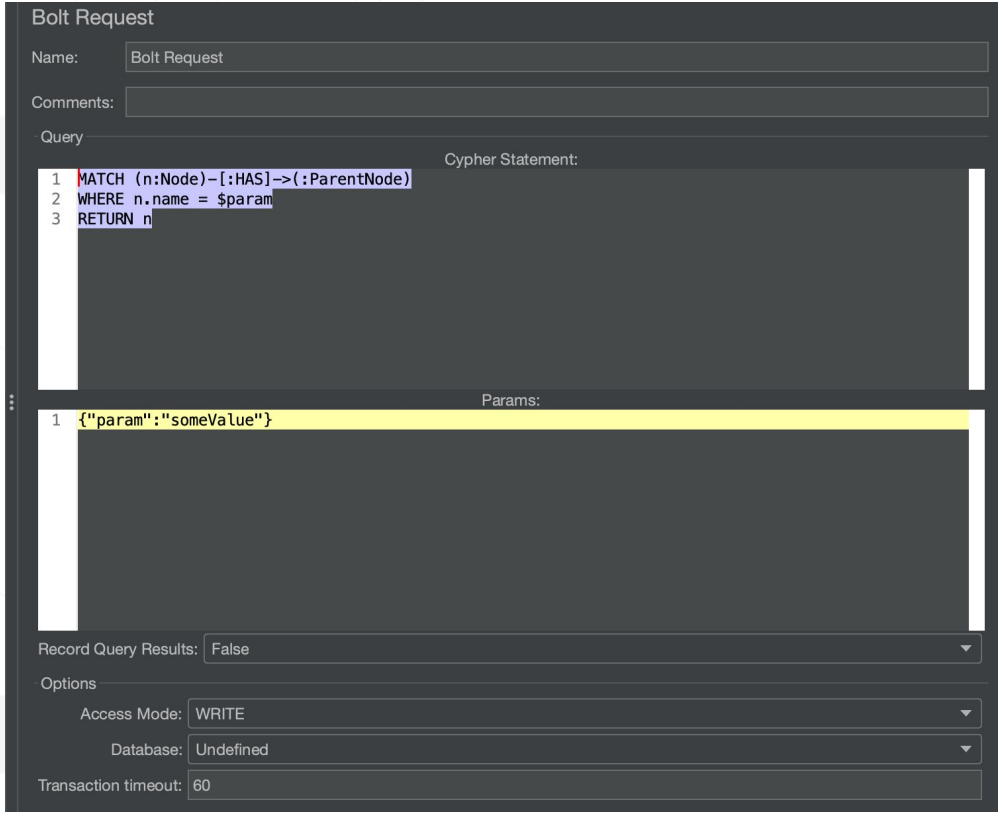
Sampler

Name : “Bolt Request”

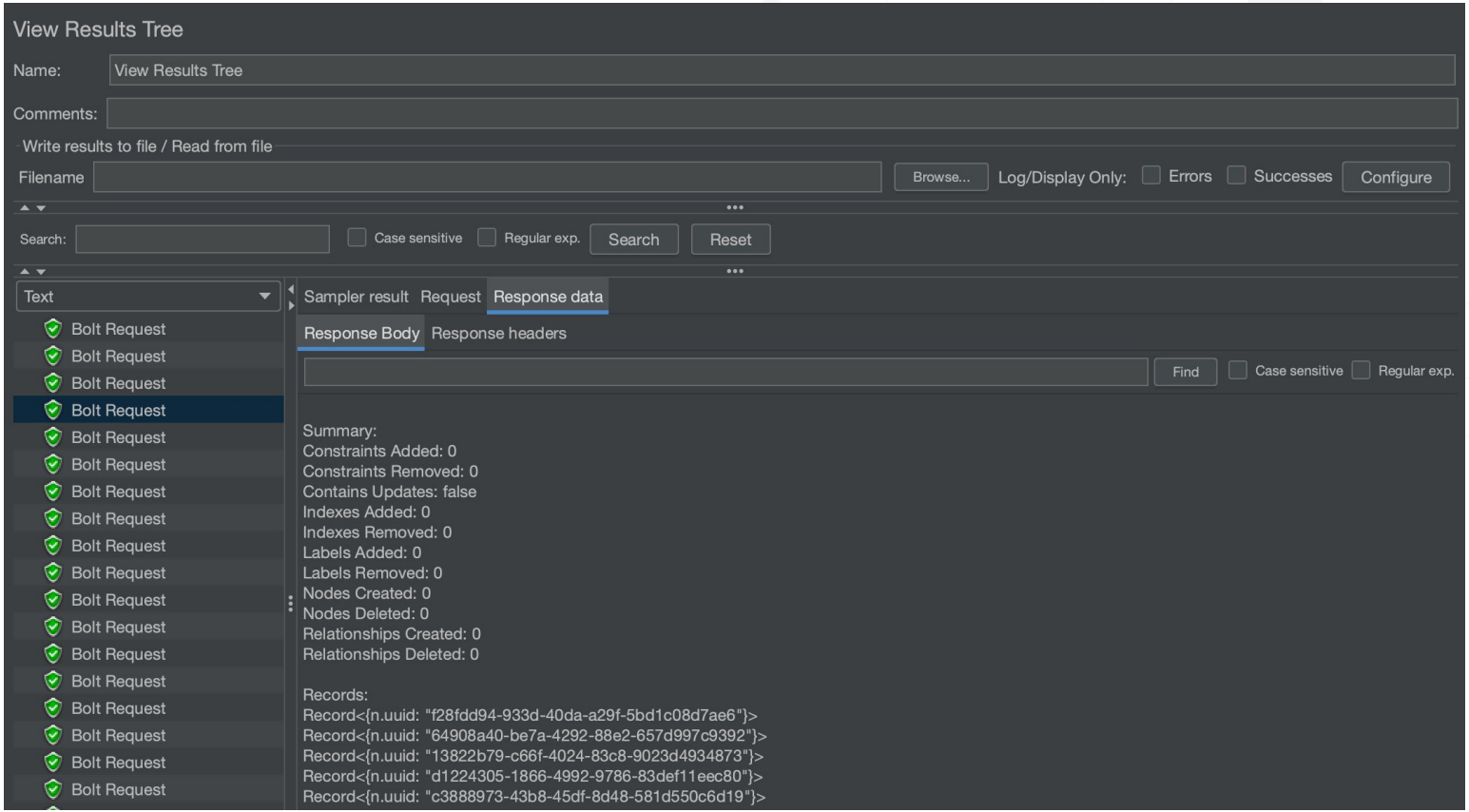
Add it to a Thread group.



Fill in cypher query,
params
database (for 4.0+ instances),
access mode (for proper cluster routing).



View Results Tree





Gatling



Gatling intro

- Tests as code (in scala)
  - Using Gatling DSL
- Simulation
  - protocol config
  - scenario
  - setup
- Drop scala file in

  `$GATLING_HOME/user-files/simulations`

```
class HttpApiSim extends Simulation {
  //protocol
  val httpProtocol = http
    .baseUrl("http://localhost:7474")
    .acceptHeader("application/json")
    .basicAuth("neo4j", "changeme")

  //scenario
  val payload = """{"statements":[{"statement":
      "MATCH (n) WHERE ID(n) = $nodeId RETURN n",
      "parameters":{"nodeId":3}}]}"""
  val testScenario = scenario("10 Cypher queries")
    .repeat(10) {
          exec(
            http("match node")
              .post("/db/neo4j/tx/commit")
              .body(StringBody(payload))
              .asJson
              .check(status.is(200))
          )
          .pause(0 milliseconds, 1 milliseconds)
    }

  //simulation setup
  setUp(
    testScenario.inject(rampUsers(10) during (10 seconds)))
      .protocols(httpProtocol)
}
```

Gatling run simulation

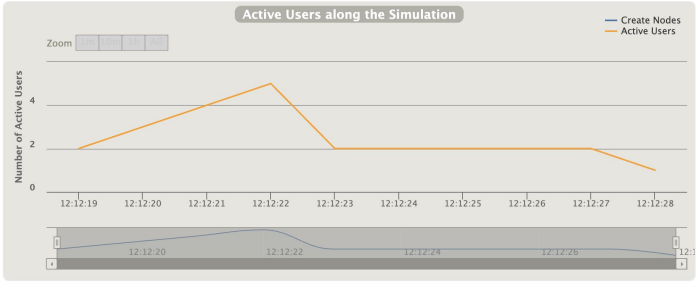
```
> ./gatling.sh
GATLING_HOME is set to /.../gatling-charts-highcharts-bundle-3.3.1
Choose a simulation number:
     [0] HttpApiSim
     [1] computerdatabase.BasicSimulation
       ...
0
Select run description (optional)

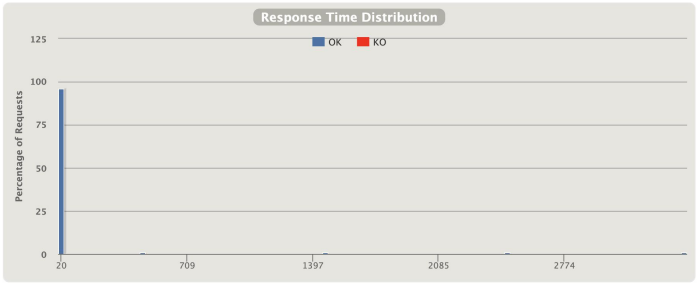
Simulation HttpApiSim started...
Simulation HttpApiSim completed in 9 seconds
Parsing log file(s)...
Parsing log file(s) done
Generating reports...
================================================================================
---- Global Information --------------------------------------------------------
> request count                                        100 (OK=100    KO=0     )
> min response time                                      3 (OK=3      KO=-     )
> max response time                                   3445 (OK=3445   KO=-     )
...
> mean requests/sec                                   10 (OK=10     KO=-     )
---- Response Time Distribution ------------------------------------------------
> t < 800 ms                                          97 ( 97%)
> 800 ms < t < 1200 ms                                 0 (  0%)
> t > 1200 ms                                          3 (  3%)
> failed                                               0 (  0%)
================================================================================
Reports generated in 0s.
Please open the following file:
/.../gatling-charts-highcharts-bundle-3.5.0/results/httpapisim-20210107121216192/index.html
>
```


neo4j

Gatling reports







gatling-bolt plugin

https://github.com/sarmbruster/gatling-bolt (PoC)
Compatible with Gatling 3.3.
Compile. Put `gatling-bolt.jar` in $GATLING_HOME/lib/

```
import org.neo4j.driver.GraphDatabase.driver
import org.neo4j.gatling.bolt.Predef._
import org.neo4j.gatling.bolt.builder.SessionHelper._
import org.neo4j.driver.{AccessMode, AuthTokens}

class BoltSim extends Simulation {
  //protocol
  val boltProtocol = bolt(driver("bolt://localhost:7687",
            AuthTokens.basic("neo4j", "changeme")))

  //scenario
  val testScenario = scenario("Single Cypher query")
    .repeat(10) {
      exec(cypher("MATCH (n) WHERE ID(n) = $nodeId RETURN n", Map("nodeId" -> 3), "mydb")
      .pause(0 milliseconds, 1 milliseconds)
    }
  //simulation setup
  setUp(testScenario.inject(rampUsers(10) during (10.seconds)))
      .protocols(boltProtocol)
}
```


neo4j

Gatling-bolt DSL

Protocol :

```
bolt(driver)
```

Actions :

```
cypher(query_string, params_map, db)
transaction(
    cypher(...),
    cypher(...), ...
 )
session(accessMode,
    transaction(...) // or cypher(...)
 )
```


neo4j

JMeter vs Gatling, for BOLT



+ Works out of the box
+ UI-based, simple to set up

− Little control on Driver params & Sessions/Transactions

  Basic auth only. No SSL.

  No bookmarks, no multi-statements.



+ Code-based => powerful, flexible

  Complete control on Driver params

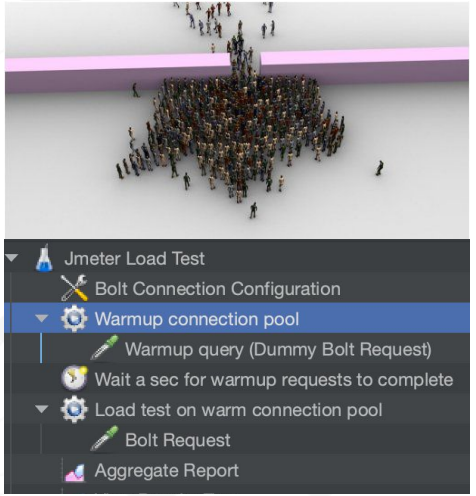
+ Nice reports

− requires some skills in Scala & the Gatling DSL

− limited DSL coverage for now

  No assertions, no bookmarks

Best practices (client side)



Driver connection pool

Default size = 100 → bottleneck for high concurrency tests?

Connection establishment is on-demand & expensive!!

→ Warm up the pool

Other considerations

cluster? → use **neo4j://...** and the correct READ/WRITE **AccessMode**

→ use cypher parameters (to benefit from the query plan cache)

→ consuming results causes slow down

→ https://jmeter.apache.org/usermanual/best-practices.html

Best practices (server side)

Server side

→ Make sure caches are warm : page cache, query plan cache.

Ex : run the test once for warm up, then measure the second run.



→ query.log, metrics, Fabric routing, RBAC each have a (small) cost

```
dbms.logs.query.enabled=OFF
metrics.csv.enabled=false
fabric.enabled_by_default=false
dbms.security.auth_enabled=false
```