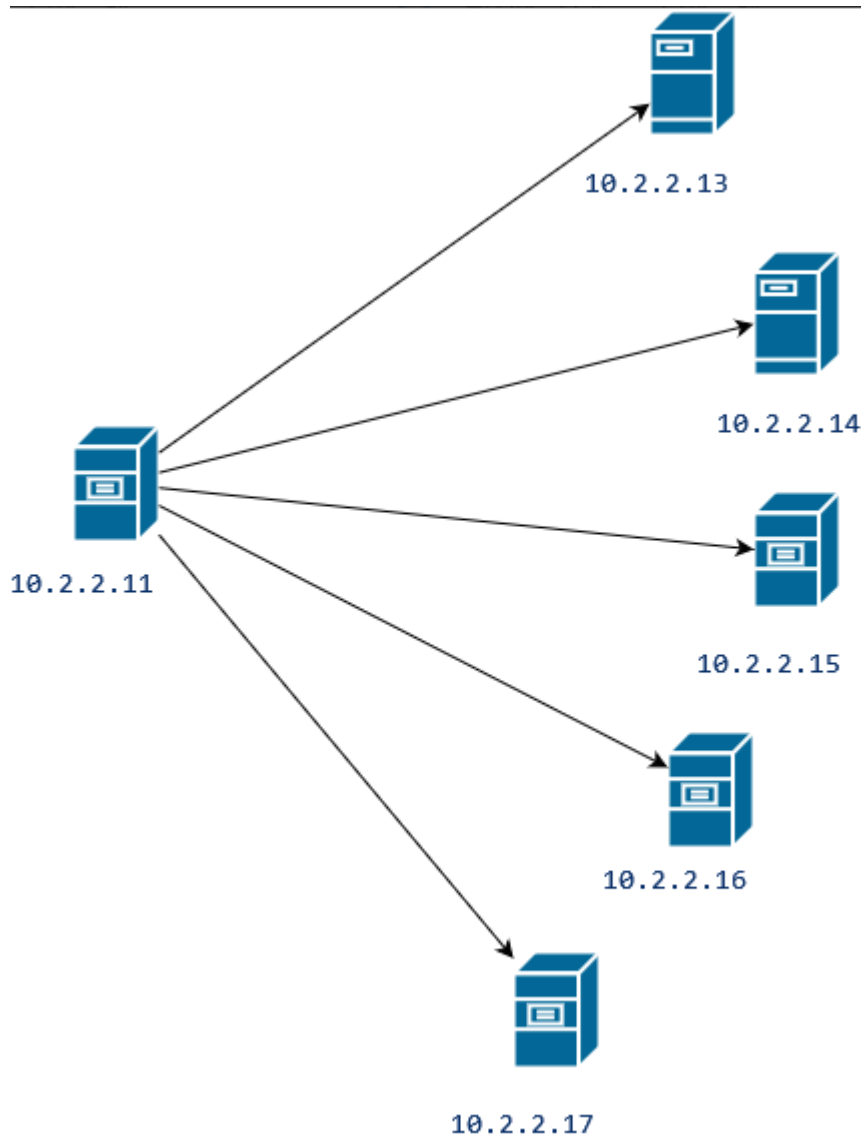


Distributed system report,

1. System architecture.



- The server 10.2.2.11 will be the server that send queries to 5 other servers in the network.
- Servers that have IP address 10.2.2.13-17 store the same database.

2. The guideline to build and run the test for sql servers.

Node.js environment setup.

Download Node.js on Ubuntu 22.04.4 LTS:

```
#Installs nvm (node version manager)
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash

#Download and install node.js
nvm install 21
```

```
#verifies the right node.js version is in the environment
node -v # should print "v21.7.3"


#verifies the right npm version is in the environment
npm -v # should print "10.5.0"
```

Guideline to run script.

1. Create a folder named testQueries.

```
mkdir testqueries
cd testqueries
```

2. Clone this Github repository: .
3. Stand at the current folder run `npm i` it will install all necessary package for the project .

A screenshot of a code editor window titled 'root@api: ~/testQueries/nodejs'. The editor shows a 'package.json' file with the following content:

```
{
  "type": "module",
  "main": "test.js",
  "dependencies": {
    "dotenv": "^16.4.5",
    "mysql2": "^3.9.4",
    "p-map": "^7.0.2",
    "uuid": "^9.0.1"
  }
}
```

 The file is 10 lines long and 162 bytes. The editor interface includes a dark theme, a cursor at line 1, and a status bar at the bottom showing 'package.json' 10L, 162B, 4,19, and All.

4. Run command `node --max-old-space-size=4096 test1.js` to test the performance of the program in multithreading way without any load-balancer.
5. Run command `node --max-old-space-size=4096 test2.js` to test the performance of the program in multithreading way with the load-balancer using Round Robin algorithm.
6. Run command `node --max-old-space-size=4096 test3.js` to test the performance of the program in multithreading way without the load-balancer using Least Connection algorithm.

3. Explain about the scripts.

- This script will run the queries in a queries.txt file concurrently and then log out how long it takes to run all the queries.

```

root@api: ~/testQueries/nodejs
emp_no: 10009,
salary: 78335,
from_date: 1994-02-16T00:00:00.000Z,
to_date: 1995-02-16T00:00:00.000Z
},
[Object: null prototype] {
  emp_no: 10009,
  salary: 80944,
  from_date: 1995-02-16T00:00:00.000Z,
  to_date: 1996-02-16T00:00:00.000Z
},
[Object: null prototype] {
  emp_no: 10009,
  salary: 82507,
  from_date: 1996-02-16T00:00:00.000Z,
  to_date: 1997-02-15T00:00:00.000Z
},
... 2843947 more items
]
Load test completed
Total execution time: 17.246s
root@api:~/testQueries/nodejs# ls
node_modules package-lock.json package.json queries.txt test.js test2.js
root@api:~/testQueries/nodejs#

```

- Here I let the script run all the queries one time and there are 2 queries will be started at the same time

```

async function loadTest(queries) {
  const queriesArray = new Array(1).fill(queries).flat(); // Flatten the array
  await pMap(queriesArray, query => executeQuery(query), { concurrency: 2 }); //
  Reduce concurrency
  console.log('Load test completed');
}

```

- Here is the queries in the queries.txt file:

```

-- Query 1: Get all employees with their department numbers from the
current_dept_emp table
SELECT emp_no, dept_no FROM current_dept_emp;

```

- The function that implement load balancer, the mechanic here is that the function will return the pool that has the least connections:

```

function getLeastConnectionPool() {
  // Sort the pools by the number of active connections and select the one with
  the least connections
  return pools.sort((a, b) => a.activeConnections - b.activeConnections)[0];
}

```

4. The scenario.


```
root@api: ~/testQueries/nodejs
[Object: null prototype] { emp_no: 10082, dept_no: 'd008' },
[Object: null prototype] { emp_no: 10083, dept_no: 'd004' },
[Object: null prototype] { emp_no: 10084, dept_no: 'd004' },
[Object: null prototype] { emp_no: 10085, dept_no: 'd004' },
[Object: null prototype] { emp_no: 10086, dept_no: 'd003' },
[Object: null prototype] { emp_no: 10087, dept_no: 'd007' },
[Object: null prototype] { emp_no: 10088, dept_no: 'd009' },
[Object: null prototype] { emp_no: 10089, dept_no: 'd007' },
[Object: null prototype] { emp_no: 10090, dept_no: 'd005' },
[Object: null prototype] { emp_no: 10091, dept_no: 'd005' },
[Object: null prototype] { emp_no: 10092, dept_no: 'd005' },
[Object: null prototype] { emp_no: 10093, dept_no: 'd007' },
[Object: null prototype] { emp_no: 10094, dept_no: 'd008' },
[Object: null prototype] { emp_no: 10095, dept_no: 'd007' },
[Object: null prototype] { emp_no: 10096, dept_no: 'd004' },
[Object: null prototype] { emp_no: 10097, dept_no: 'd008' },
[Object: null prototype] { emp_no: 10098, dept_no: 'd009' },
[Object: null prototype] { emp_no: 10099, dept_no: 'd007' },
[Object: null prototype] { emp_no: 10100, dept_no: 'd003' },
... 299924 more items
]
Load test completed
Total execution time: 3:14.806 (m:ss.mmm)
root@api:~/testQueries/nodejs#
```

==> The result show that the time it do the same amount of queries.