Serverless Supercomputing Demo 3

From last time….

Release 3 (Due 02/23/2018)

- Write the pseudo code of the algorithm.
- Identify the parts of algorithm that can be parallelized/
- Build and Parallelize the algorithm to run on OpenWhisk on OpenShift in JavaScript.
- Parallelized implementation of the algorithm in JavaScript.

Working of code

```
var piFunctionString = "function main(params) { " +
        "var numPoints = parseInt(params.numPoints); " +
        "var inCircle = 0; " +
        "for(var i=0; i<numPoints; i++){ " +
                "randX = (Math.random() * 2) - 1; " +
                "randY = (Math.random() * 2) - 1; " +
                "distFromCenter = Math.sqrt(randX * randX
                "if (distFromCenter <= 1){ " +
                        "inCircle = inCircle + 1; " +
                "} " +
        "} " +
        "return {inCircle: inCircle}; " +
"}";
```

```
const registerActionPromise = new Promise((resolve, reject) => {
        var headers = {
        'Content-Type': 'application/json'
        };

        var dataString = JSON.stringify({
                "namespace":"_",
                "name":"testPoints",
                "exec": {
                        "kind": "nodejs:6",
                        "code": piFunctionString
                }
        });

        var options = {
            url: 'https://' + AUTH + '@' + APIHOST + '/api/v1/namespaces/_
            method: 'PUT',
            headers: headers,
            body: dataString
        };

        function callback(error, response, body) {
            if (!error && response.statusCode == 200) {
                console.log(body);
                console.log("Action Registered");
                resolve(body);
            } else {
                console.log(error);
                console.log(response);
                resolve(false);
            }
        }

        console.log("Registering action on OpenWhisk");
        request(options, callback);
});
```

```
const triggerActionPromise = function(){
        return new Promise((resolve, reject) => {
                var headers = {
                'Content-Type': 'application/json'
                };

                var dataString = JSON.stringify({
                    "numPoints": POINTS_PER_ACTION
                });

                var options = {
                    url: 'https://' + AUTH + '@' + APIHOST + '/api/v1/namespaces/_/action
                    method: 'POST',
                    headers: headers,
                    body: dataString
                };

                function callback(error, response, body) {
                    if (!error && response.statusCode == 200) {
                        resolve(JSON.parse(body));
                    } else {
                        console.log(error);
                        console.log(response);
                        resolve(false);
                    }
                }

                request(options, callback);
        });
}
```

```
registerActionPromise.then(triggerActionPromises).then((resp) => {
        if (resp){
                var totalInCircle = 0;
                const totalPoints = NUM_ACTIONS * POINTS_PER_ACTION;
                for (var i = 0; i< resp.length; i++){
                        var actionResult = resp[i]["response"]["result"]
                        console.log("Action " + (i + 1) + " finished with " + JSON.stringify(actionResult));
                        totalInCircle += actionResult['inCircle'];
                }
                console.log(totalInCircle + " out of " + totalPoints + " were in the circle.");
                console.log("Computed Value of Pi: " + 4 * (totalInCircle / totalPoints));
                console.log("Finished in " + ((Date.now() - startingMilliseconds) / 1000) + " seconds");
        }
});
```

Results

```
Action 84 finished with {"inCircle":788}
Action 85 finished with {"inCircle":773}
Action 86 finished with {"inCircle":787}
Action 87 finished with {"inCircle":783}
Action 88 finished with {"inCircle":792}
Action 89 finished with {"inCircle":765}
Action 90 finished with {"inCircle":779}
Action 91 finished with {"inCircle":771}
Action 92 finished with {"inCircle":762}
Action 93 finished with {"inCircle":772}
Action 94 finished with {"inCircle":815}
Action 95 finished with {"inCircle":763}
Action 96 finished with {"inCircle":808}
Action 97 finished with {"inCircle":771}
Action 98 finished with {"inCircle":800}
Action 99 finished with {"inCircle":809}
Action 100 finished with {"inCircle":773}
78387 out of 100000 were in the circle.
Computed Value of Pi: 3.13548
Finished in 22.339 seconds
```

Challenges and Fixes

- Running out of space on the VM filesystem
  - log files on VM kept on growing.
  - /var/log/journal/<>
  - Deleting these log files freed the space

- Not able to invoke actions more than 60 times in a min
  - default action invoked per min was set to 60
  - `actions_invokes_perMinute: "60"`
  - Changing it to 1200

Problems we are facing

I. Request Time Out

- In our algorithm to compute the value of “pi”, we keep track of the total number of points, and the number of points that are inside the circle. If we divide the number of points within the circle, N_{inner} by the total number of points, N_{total}, we should get a value that is an approximation of the ratio of the areas we calculated, $\pi/4$.
- In other words,

$$\pi \approx 4 * N_{inner} / N_{total}$$

- The estimation is accurate only when we have hundreds of thousands of points.
- Right now we are estimating “pi” with 100000 points.
- For more than 100,000 points we are getting 504 Error: Request Timeouts

Our Speculation

- We are invoking actions with a *blocking* invocation (i.e., request/response style)
- A blocking invocation request will *wait* for the activation result to be available
- Per action timeout (ms) (Default: 60s)
- A container that runs longer than 60s is terminated.

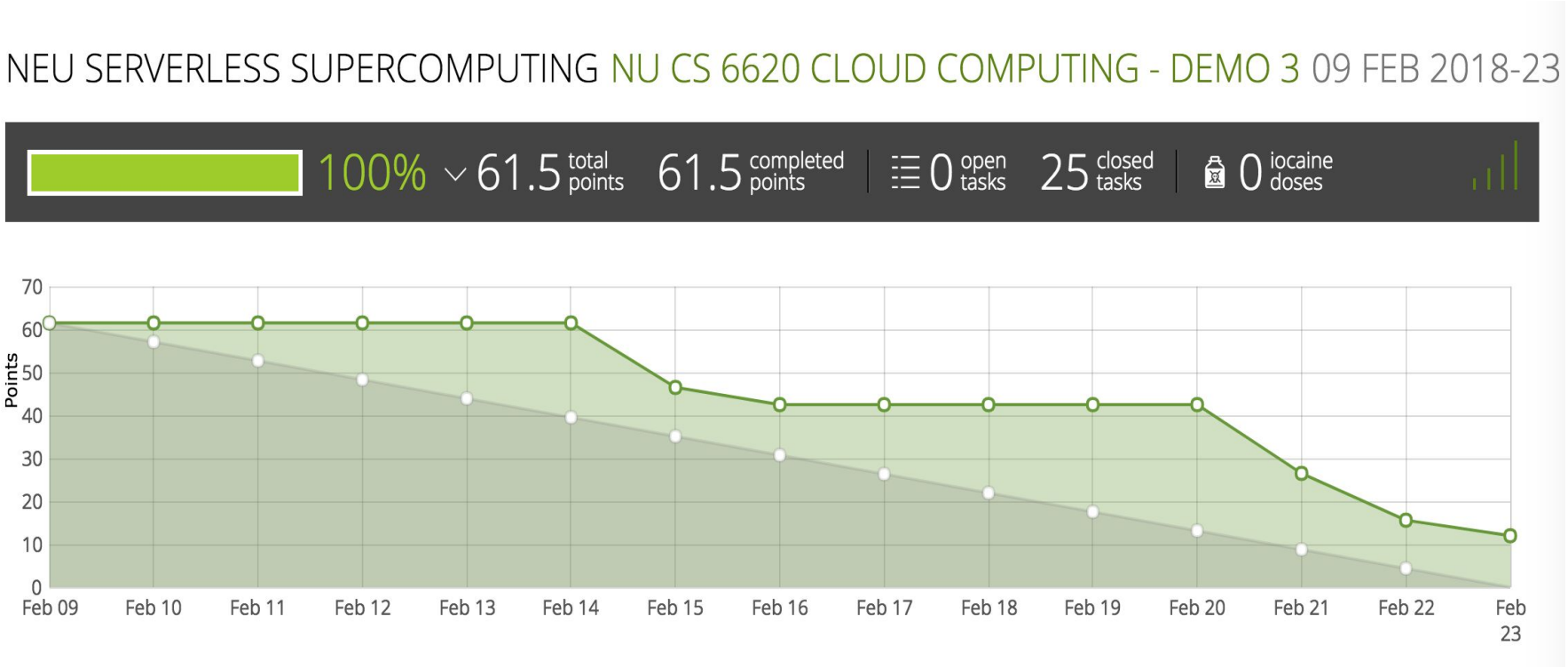
Fix:

- We don't need the action result right away
- Omit the --blocking flag to make a non-blocking invocation.
- Fetch the result later by using the activation ID.

II. Increasing resource quota on OpenStack

- Currently only 2 floating IPs are allocated
- We need 4 for each team member to work on their own VM on OpenStack
- Waiting for the request to be fulfilled.

Burndown


NEU SERVERLESS SUPERCOMPUTING NU CS 6620 CLOUD COMPUTING - DEMO 3 09 FEB 2018-23
100%
61.5 total points
61.5 completed points
0 open tasks
25 closed tasks
0 iocaine doses
Points
70
60
50
40
30
20
10
0
Feb 09
Feb 10
Feb 11
Feb 12
Feb 13
Feb 14
Feb 15
Feb 16
Feb 17
Feb 18
Feb 19
Feb 20
Feb 21
Feb 22
Feb 23

Release 4 (Due 03/16/2018)

- Fix implementation to scale efficiently
- Performance Evaluation
- Error Handling
- Finalize Application
- Run application on MoC OS cluster
- Documentation

Release 5 (Due 03/30/2018)

- Test the application on OpenShift cluster on MoC
- Analyse if the computation time decreases linearly and relative to the number of additional pods added
- Show that the algorithm should scale linearly
- Test the algorithm as scale and provide performance data of the results