CS553 – Cloud Computing

Programming Assignment 4

Performance Evaluation

Karan Jeet Singh; Kanwal Preet Singh; Alaa Ayach

12-1-2014

1 INTRO

In this assignment we have implemented dynamic provisioning for a task execution framework on Amazon EC2 using the SQS.

2 DESIGN

We have implemented our dynamic provisioning framework on Amazon EC2. The framework is separated into three different components

1) Client
2) Scheduler
3) Workers

2.1 ABSTRACT

The overall functionality of the components is as below

- Client

The tasks are submitted to the scheduler using the Client. The client the sends the list of tasks that needs to be executed. The tasks are executed in a batch i.e. all the tasks in a given file are sent over the network to the scheduler in order to be processed by the workers.

- Scheduler

The scheduler receives the file which in turn consists of the lists of tasks to be executed. The scheduler is responsible for distribution of tasks to the workers. The received file is processed and tasks are then put in a queue. The queue acts as the repository of all the tasks those need to be processed. The tasks are picked by the workers for processing, when a worker successfully executes a given task then it puts the same task in a separate response queue with a status "true" indicating the task has been executed without any exception or errors, if the task fails then the status of the respective task is set as false. The response queue is then used to check the status of the individual tasks.

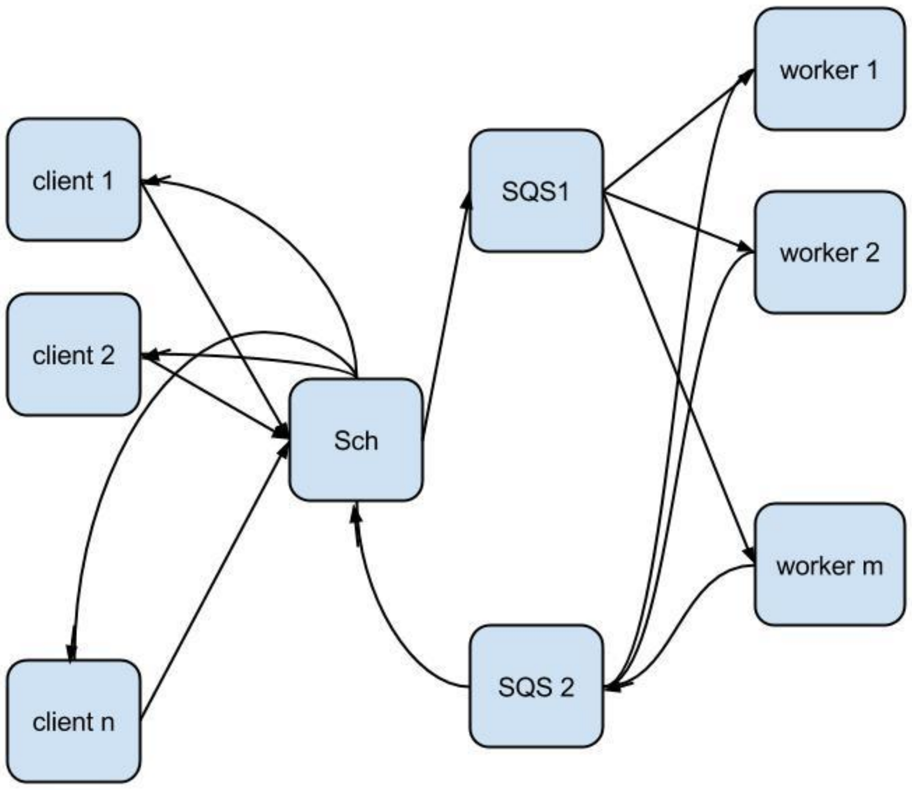
Workers

The interaction between the scheduler and the worker is done using the SQS. The workers pick the task from the SQS which needs to be executed.

2.2 ASSUMPTIONS WHILE DESIGN

We assumed that the scheduler will be responsible for ***distributing and bashing*** all the messages to the queues and the clients according to the messages that it gets.

The overall design is shown in the figure below (DynamoDB isn't considered in the figure), where the curve arrows showed the responses from workers through SQS 2.



2.3 DESIGN TRADEOFFS

The assignment handout puts the scheduler as the bottleneck of this system, because now we can't be able to use the full scalability of SQS because here only one instance is pushing messages to the queue.

2.4 Improvement and Extensions to the current system

The system can be improved by deleting the scheduler part and making the clients pulling the responses by themselves.

3 Design Details

3.1 Design Details

- Client

Client <server ip address> <port number> <workload>

Here the server IP address is known beforehand so that the client can request the connection to the server on the specified port. The client also sends the workload file which consist of all the tasks to be executed.

- Scheduler

Scheduler <port number> <no. of local workers> <worker switch>

The scheduler opens the connection for the given port number. The number of threads is the no. of local workers and the worker switch specifies whether to execute the tasks on local workers or on the remote workers.

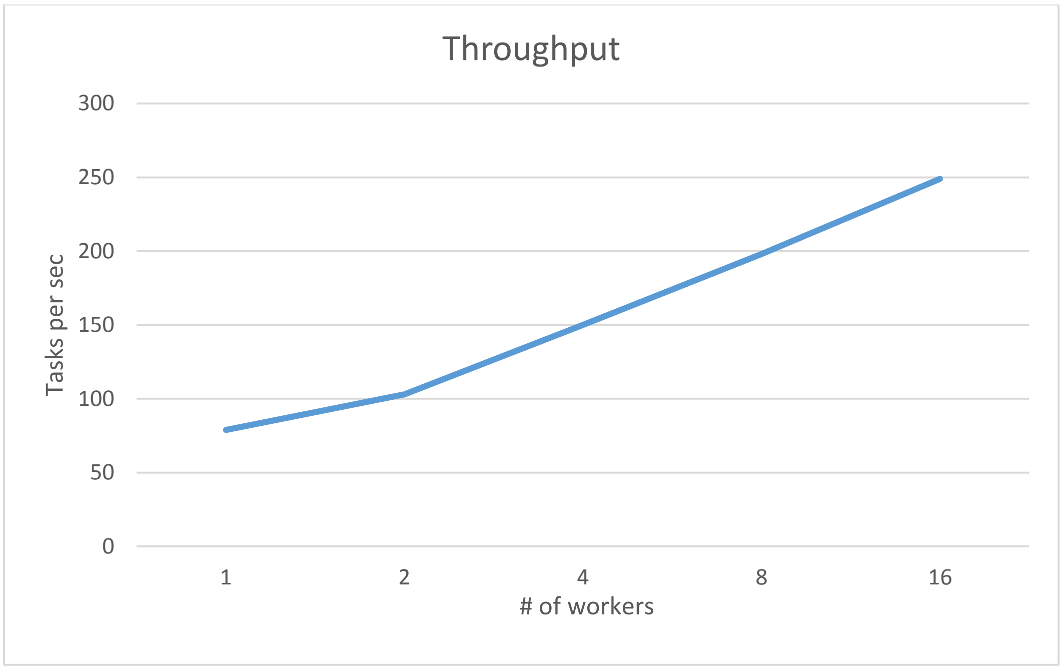
- Local workers

The tasks needs to be executed are there in the queue. The Local worker then picks up the tasks from the queue, executes them and puts back the result on the response queue with the status of each task.

4 Performance Evaluation

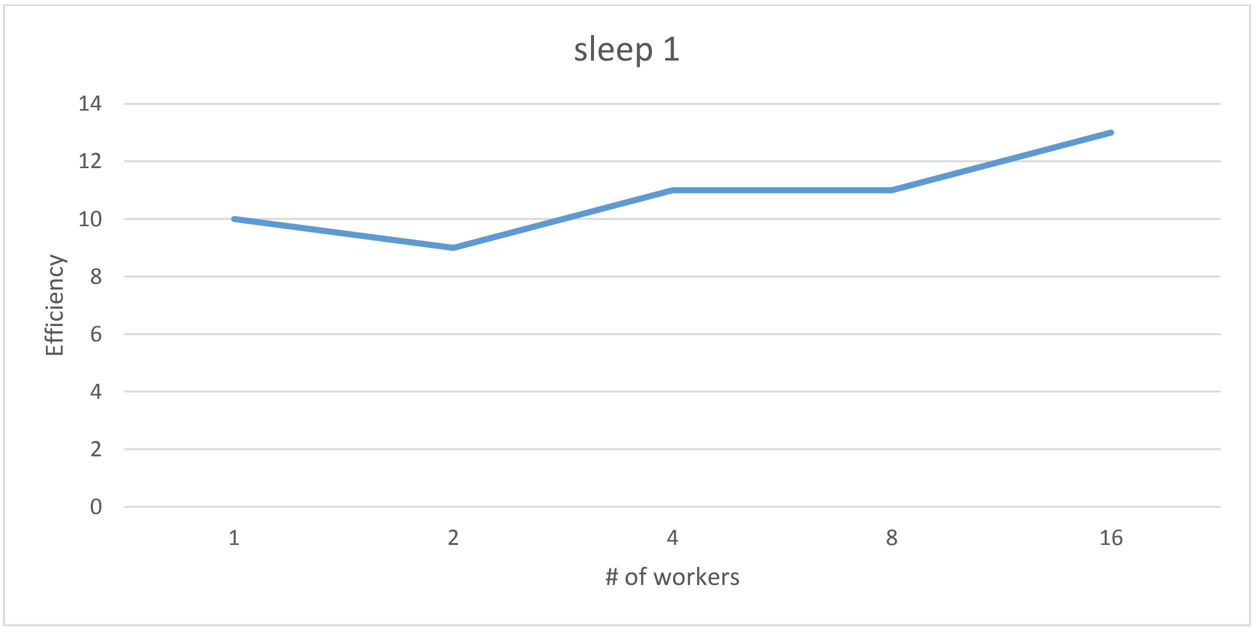
4.1 Throughput and Efficiency

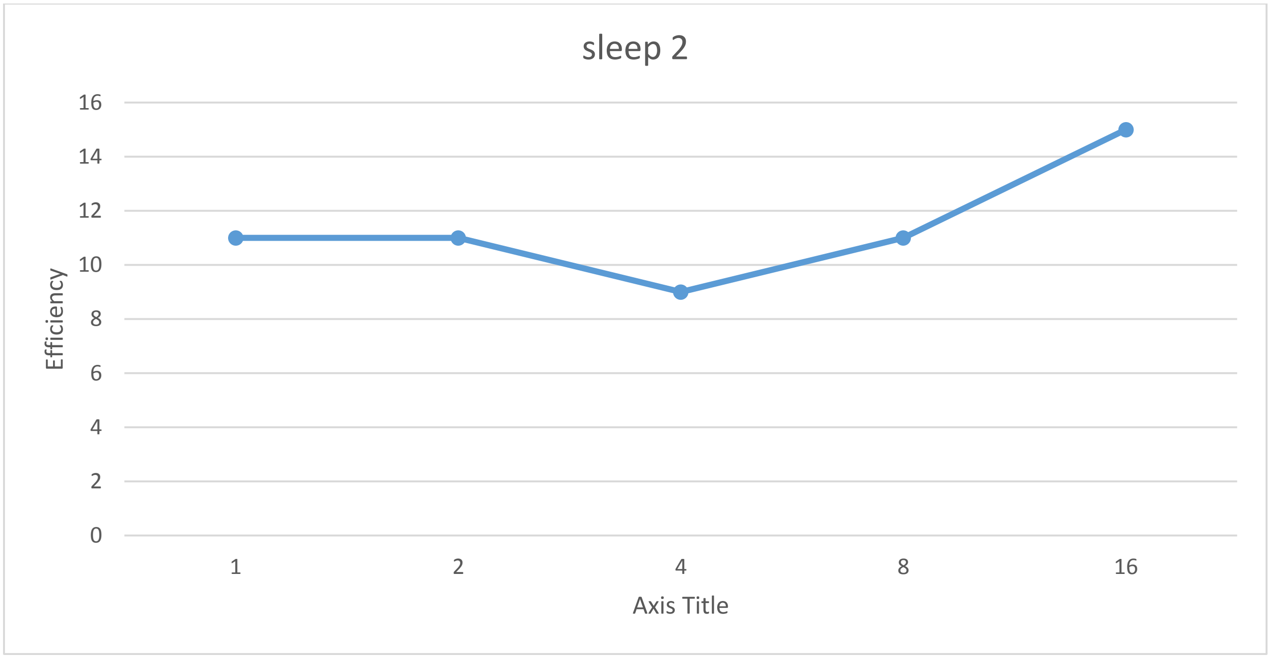
- Throughput

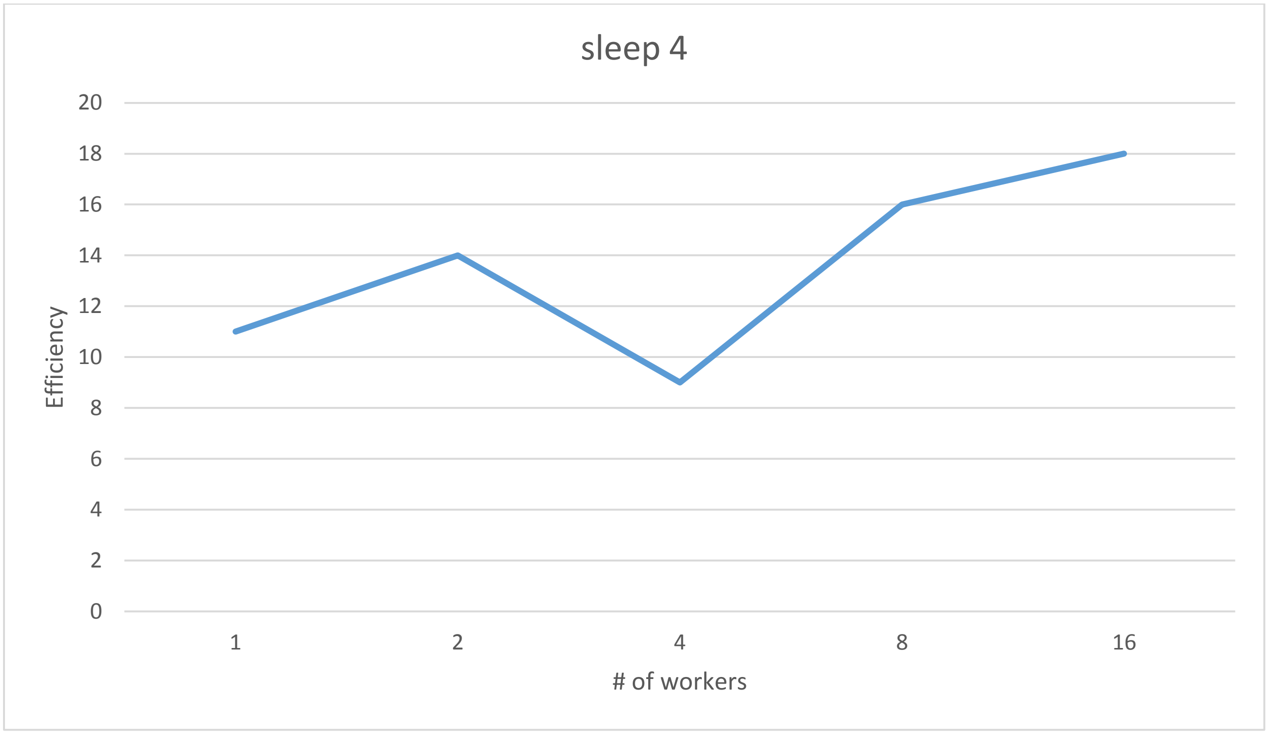


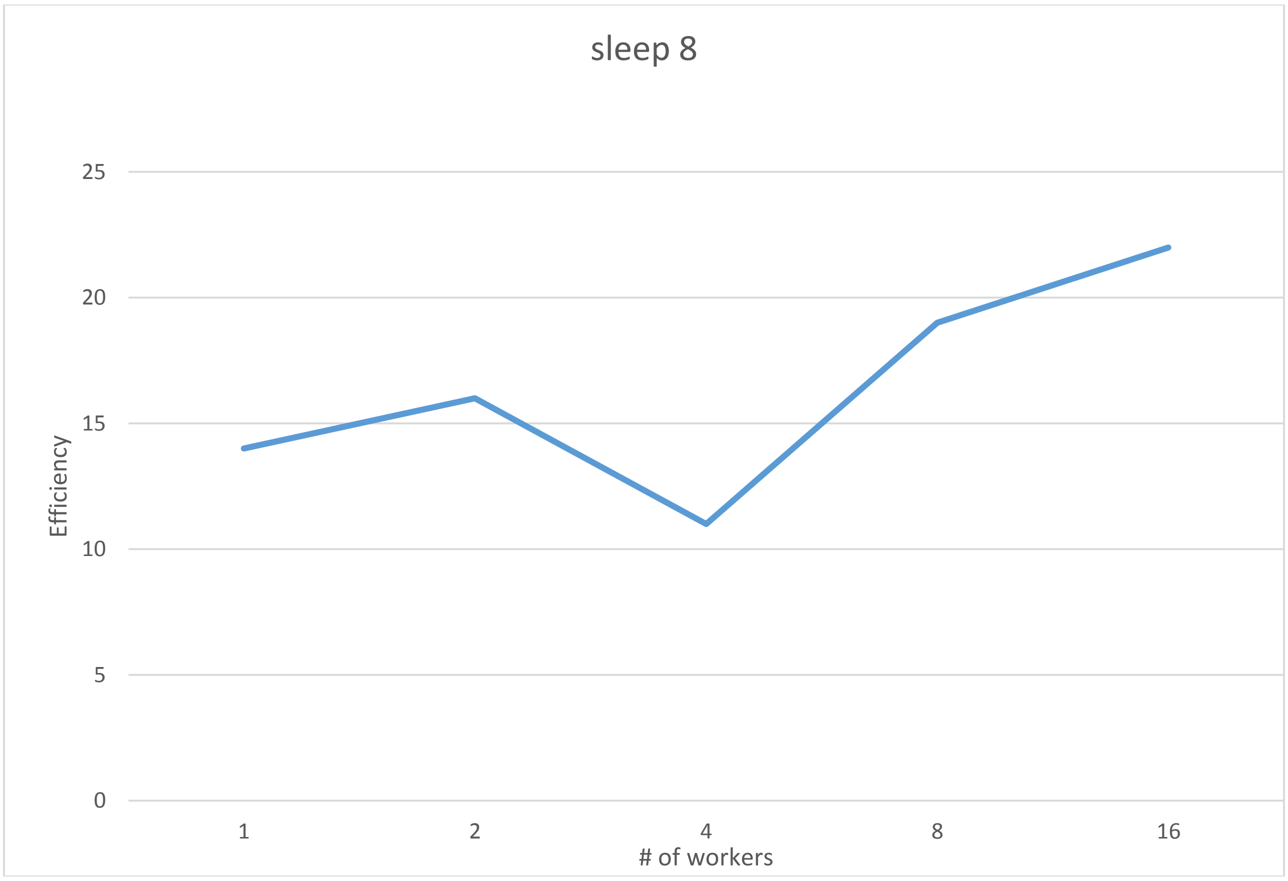
<u>We notice that we didn't get a double increase in throughput by doubling the number of workers, that's because of the bottleneck of the scheduler.</u> The throughput does increase with the increase in number of workers.

- Latency




sleep 2
Efficiency
0
2
4
6
8
10
12
14
16
1
2
4
8
16
Axis Title


sleep 4
Efficiency
0
2
4
6
8
10
12
14
16
18
20
1
2
4
8
16
# of workers



The efficiency is increasing when sleep time increasing, as the overheads involved remain the same for all the tasks. The overheads cause the variations when the tasks are shorter.

5 CONCLUSION

The throughput does increase with the increased workers but the increase is not linear as the scheduler is the bottleneck with is responsible for putting the tasks in SQS.

We can see that the efficiency increases with the increase in the sleep time as the overhead value remains same and gets small relative to the increase sleep time.