Aditya Kale – A20355807

Cloud Computing

Programming Assignment – 3

- **Experimental Purpose**
  - In this experiment we have created distributed task execution framework on Amazon EC2 using the SQS.
  - The assignment is to implement a task execution framework (similar to CloudKon architecture).
  - I have created two frame works of which one runs on local system and other run on remote systems.
  - I have used EC2 to run my framework experiments.
  - In this frame work there are two components Client and worker which works individually.
  - The SQS service is used to handle the queue of requests to load balance across multiple workers
  - Results of all data are compared among themselves and graph are plotted.
  - DynamoDB is used to maintain and execute unique task id by a worker. Multiple worker gets data from the queue and executes these tasks.
  - Throughput of the system is calculated by changing the number of workers in remote worker execution.
  - While number of threads are changed while evaluating the framework on local machine.
  - Graphs of throughput and efficiency is plotted based on the results obtain by changing these threads and remote workers.

- **Methodology:**
  - Amazon SQS service is used to put the task in a Queue and get responses from a Queue.
  - While Amazon's DynamoDB is used to check whether an atomic operation is performed on a task.

- **Operating System Used:**
  - Linux ubuntu 14.04
  - Linux kernel version – 3.19.0-25-generic

- **Versions:**
  - Ant-1.9.6
  - Java- 1.8

- **Results And Graphs**

For Local System

Evaluation of Throughput

- For Local system distributed task execution framework I have run a 10,000 tasks of sleep time zero seconds.
- This program is executed by changing the number of worker threads in power of 2. like 1,2,4,8,16.
- All the readings are taken by keeping no of client as 1.
- Only one client submits the task to a queue while there can be multiple threads of a worker which will work on that tasks concurency.
- Following are the readings observed after evaluating the above experiment.
- Throughput of a system is evaluated as
  Throughput = Total Number of Operations / Total time taken in seconds

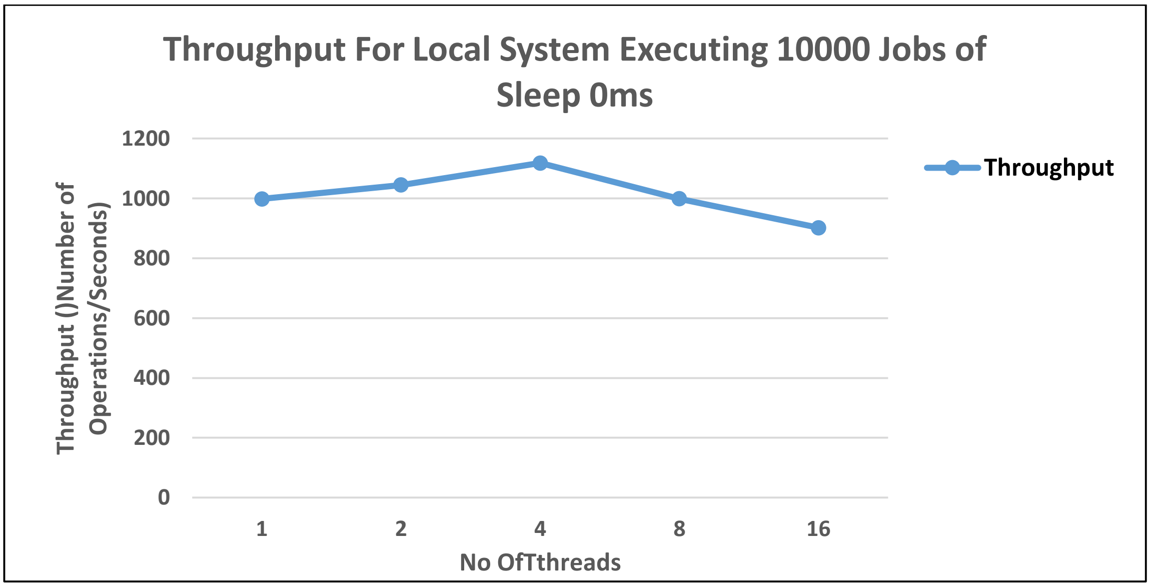
Time taken To execute 10,000 task of sleep time Zero milliseconds

Threads	1	2	4	8	16
Time in milliseconds	100189	95728	89414	100124	110875

Throughput results

Number of Threads	1	2	4	8	16
Throughput	998.1136	1044.626	1118.393	998.7615	901.9166

Throughput Graph:



Conclusion:

- From the above graph we can see that throughput of a system increases as the number of thread increases.
- The maximum throughput for the system is obtained for 4 threads.
- Further increasing the number of threads throughput of a system also decreases.
- Throughput decreases because as number of threads increases threads will require more time in context switching as t2.micro is a single core system.

For Evaluation of Efficiency

- The Ideal time to run these experiments are assumed to have zero cost of communication and distribute the tasks.
- Efficiency of a system is measured as ideal time divided by the actual time.
- The efficiency of a system is measured by varying the number of remote workers and the sleep time.
- As the workers are increased the total number of task submitted by a client is also increased in the same manner.
  Like if number of task submitted by a client 1000 for one worker, then the number of task submitted for 16 workers is 16*1000.
- The efficiency is measured by varying the sleep time to 10ms, 1000ms, and 10000ms.
- Following are the readings observed after evaluating the above experiment.

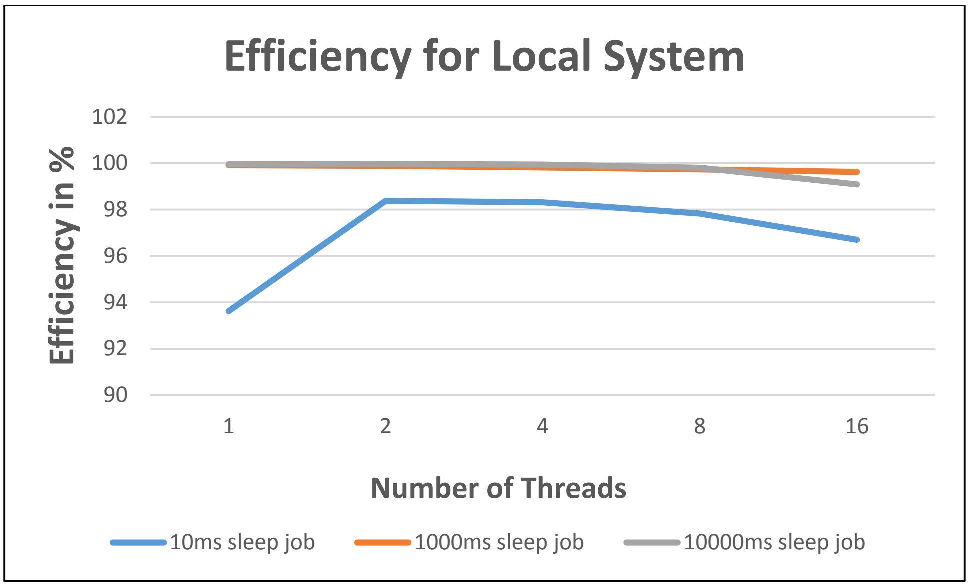
Total Task Per Worker	Sleep Time	1-Thread	2-Thread	4-Thread	8-Thread	16-Thread
1000	10ms	10682	10165	10172	10222	10342
100	1000ms	100091	100114	100190	100270	100385
10	10000ms	100039	100029	100062	100201	100920

Efficiency calculated from above table

Total Task Per Worker	Sleep Time	1-Thread	2-Thread	4-Thread	8-Thread	16-Thread
1000	10ms	93.61542782	98.376783	98.309084	97.828214	96.693096
100	1000ms	99.90908273	99.88613	99.81036	99.730727	99.616477
10	10000ms	99.9610152	99.971008	99.938038	99.799403	99.088387

Efficiency % = (Ideal time / Actual Time) * 100

Efficiency Graph:



Conclusion:

- From the above graph we can see that as similar to the behaviour of throughput, efficiency increases for the 2 threads and then the efficiency of the system decreases as the number of threads are increased further.
- Efficiency for 1000ms job and 10000ms job is very less affected. For these task it gives up to 98% efficiency.
- Efficiency of the 10ms job is less as compared to efficiency of 1000ms job and 10000ms job.

For Remote System

Evaluation of Throughput

- For Remote system distributed task execution framework I have run a 10,000 tasks of sleep time zero seconds.
- This program is executed by changing the number of worker having a single thread in power of 2. Like 1,2,4,8,16 workers.
- All the readings are taken by keeping no of client as 1.
- Only one client submits the task to a queue while there were multiple workers who are working on that tasks concurrently.
- Following are the readings observed after evaluating the above experiment.
- Throughput of a system is evaluated as
  Throughput = Total Number of Operations / Total time taken in seconds

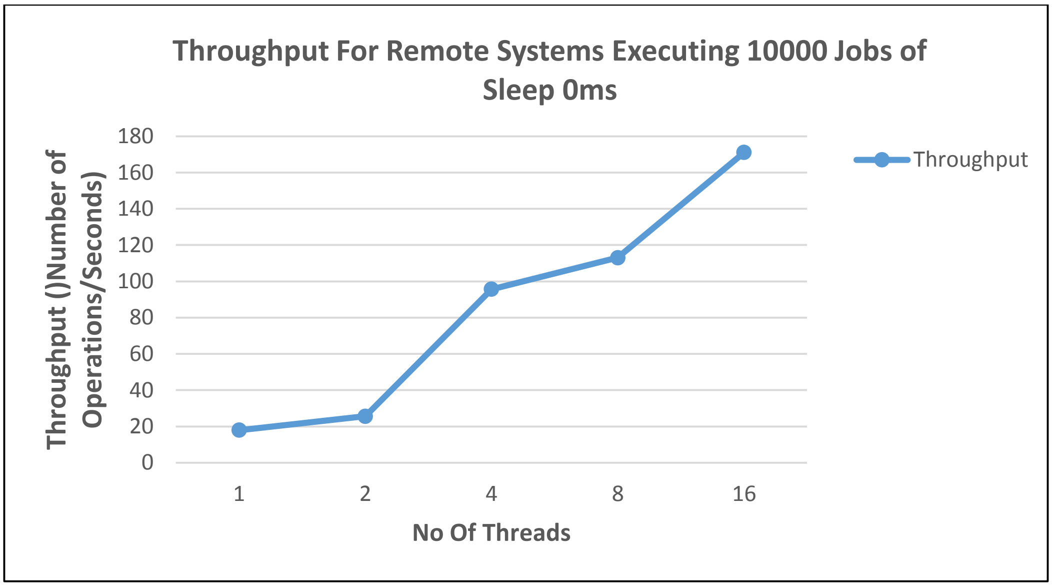
Time taken to execute 10,000 task of sleep time Zero milliseconds

Number of concurrent Workers	1-Worker	2-Worker	4-Worker	8-Worker	16-Worker
Time in milliseconds	557295	389912	104587	88492	58465

Throughput results

Number of concurrent Workers	1-Worker	2-Worker	4-Worker	8-Worker	16-Worker
Throughput	17.94379	25.64679	95.61418	113.0039	171.0425

Throughput Graph:



Conclusion:

- From the above graph we can see that as the number of worker are increases in power of 2 the throughput of the system also increases.
- As workers are increased more number of tasks are process at the same time, thus the number of operations performed per second increases, increasing the overall throughput of the system.
- Here in the independent backend worker all workers work together having their own resources. The throughput of the system can also be increased by performing the multithreading operations on each worker.

- In local system throughput decreases after certain number of threads as there is only one resource, this is not the case with independent worker and thus are more beneficial in this system.

For Evaluation of Efficiency

- The Ideal time to run these experiments are assumed to have zero cost of communication and distribute the tasks.
- Efficiency of a system is measured as ideal time divided by the actual time.
- The efficiency of a system is measured by varying the number of remote workers and the sleep time.
- As the workers are increased the total number of task submitted by a client is also increased in the same manner.
- The efficiency is measured by varying the sleep time to 10ms, 1000ms, and 10000ms.
- Following are the readings observed after evaluating the above experiment.

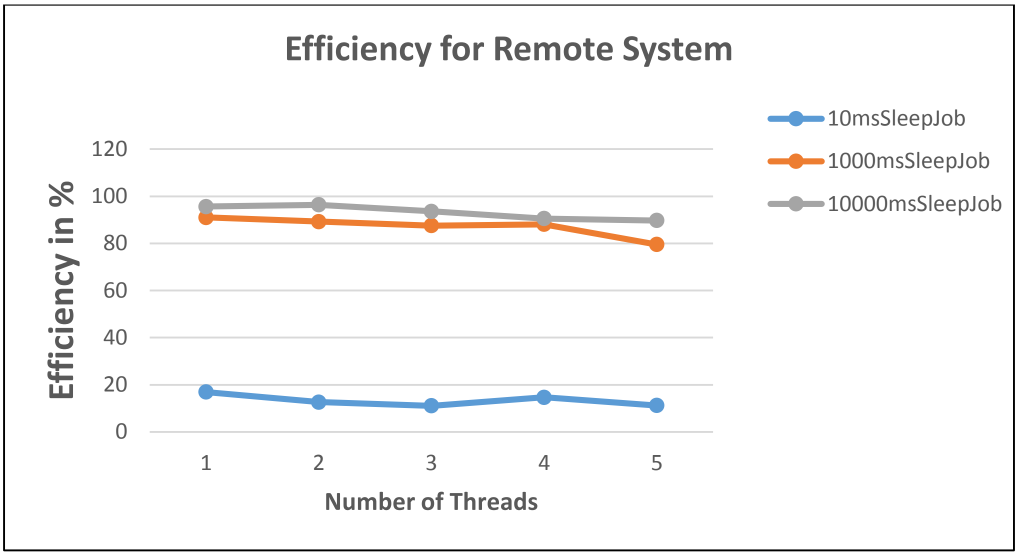
Total Task Per Worker	Sleep Time	1-Worker	2-Worker	4-Worker	8-Worker	16- Worker
1000	10ms	59080	79195	90330	68123	89130
100	1000ms	104578	103742	106809	110485	111486
10	10000ms	109842	112036	114215	113548	125745

Efficiency calculated from above table

Total Task Per Client	Sleep Time	1- Worker	2-Worker	4-Worker	8-Worker	16-Worker
1000	10ms	16.926	12.62693	11.07049	14.67917	11.21949
100	1000ms	91.03986	89.25702	87.55417	88.06848	79.52602
10	10000ms	95.62241	96.39219	93.62454	90.51002	89.69736

Efficiency % = (Ideal time / Actual Time) * 100

Efficiency Graph:



Conclusion:

- The above graph display the efficiency of 10ms sleep job, 1000ms sleep job and 10000 ms sleep job.
- The number of task vary according to the number of workers. It is clearly seen that for large number of task in a system i.e 1000 task of 10ms, the efficiency is less as compared to the less task system.
- If the number of task in a system decreases then scheduler has to schedule less no of task, so the efficiency of the distributed task execution framework increases.
- In these experiments the execution time of task is constant, so this system only measures the time of a putting a task in a SQS and a Getting a task from SQS. It also measures the time of putting data in a DynamoDB. Thus this efficiency values gives us the pure efficiency of a system handling different number of task having different execution time.