1. (Parallel and distributed systems) Suppose you want to parallelize a program on an 8-node cluster. Assume 20% of the program is purely **serial code** that cannot be parallelized, what is the **speedup** you can achieve by perfectly parallelizing this program on the 8-node cluster compare to run the same code on one node? (Hint: Use the Amdahl’s law.)
2. Discuss **different storage options** provided by AWS in terms of their advantage and disadvantage.

|  |  |  |
| --- | --- | --- |
| Storage option | Advantage | Disadvantage |
| ***Simple Storage Service (S3)*** | + provide persistent storage  +Independent of EC2 instances | + EC2 instances need to “download” data from S3 in order to access it (cannot issue read/write to S3) |
| ***Amazon Glacier*** | + low-cost storage service that provides secure and durable storage for data archiving and backup  + offload the administrative burdens of operating and scaling storage + cost | + slower than S3 |
| ***Storage Gateway*** | + securely store data to the AWS cloud for scalable and cost-effective storage  + All data is securely transferred to AWS over SSL and stored encrypted in Amazon S3 using AES 256-bit encryption  + Gateway-Stored volumes store your primary data locally, while asynchronously backing up that data to AWS. | + virtual tapes had issues retrieving data virtual tapes - we were only able to restore 1 file at a time when attempting batch restores. |
| ***Elastic Block Store (EBS)*** | *+*provide block level storage volumes (virtual disk, i.e., disk-like) to EC2 instances  + Persistent even after instances are terminated | + Instances have to mount EBSs (EFS) |

1. Compare the auto-scaling scheme in Amazon web services (AWS) and Google AppEngine (GAE)

**Auto-scaling:** the capability in cloud computing infrastructures that allows dynamic provisioning of virtualized resources. This is important in maintaining *elasticity*, avoiding resource under-utilization and over-utilization.

**AWS Auto Scaling** combines dynamic scaling and predictive scaling.

* *Dynamic scaling* is executed in response to real-time changes in resource utilization, provide enough capacity to maintain utilization at the target value specified by the scaling strategy.
* *Predictive scaling* uses machine learning to analyze each resource's historical workload and regularly forecasts the future load for the next two days.

**Google AppEngine:** App engine scales based of number of requests it receives.

* Every request your application receives is added into a instance queue. AppEngine monitors this queue, if the queue gets too long, it will start a new instance to serve your request.
* When a instance is idle, app engine turns off that instance.
* Instances are either *dynamic* or *resident*. A dynamic instance starts up and shuts down automatically based on current needs. A resident instance runs all the time, which can improve your application's performance.

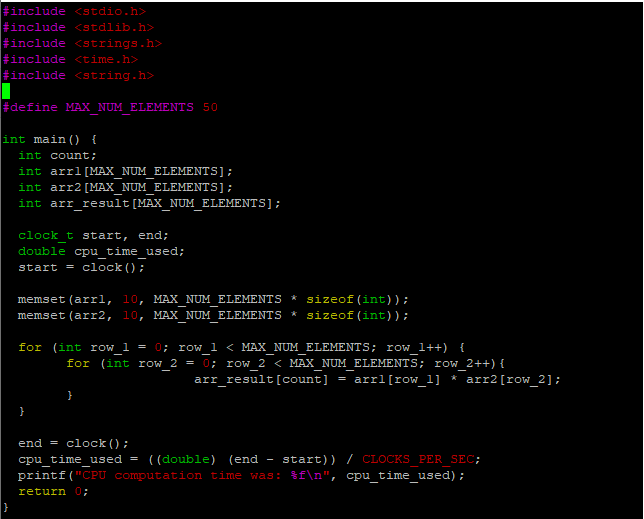
Links:

AWS: <https://docs.aws.amazon.com/autoscaling/plans/userguide/how-it-works.html>

AppEngine: <https://cloud.google.com/appengine/docs/standard/python/how-instances-are-managed>

1. Create an AWS EC2 console with Linux OS, and write a code to multiply two integer arrays (each with N numbers). Monitor the time to run that code remotely on the cloud with the following N: 50, 100, 200, and 400. Also report which zone your instance is. Submit screenshots as well.

**Availability zone:** us-east-1c



N = 50



N = 100



N = 200



N = 400

