Institute of Technology
University of Washington – Tacoma
Instructor: Wes Lloyd

Term Project Proposal

Version 0.1

Due Date: Thursday April 12th, 2018 @ 11:59 pm

Objective

For the TCSS562 term project, select from one of the listed projects, or propose your own cloud computing project. For original projects, please arrange time to discuss details of the project proposal before submitting the final proposal either after class, during office hours, via email, or by appointment.

Option #1 – Cloud Services Evaluation

Cloud providers offer a plethora of competing cloud services available and ready to be integrated into the architecture of a cloud application. As a system architect/designer, it is important to understand the tradeoffs provided by competing cloud services.

For the cloud services evaluation project option, your group will prototype the use of a minimum of two or three competing cloud services. Option #1 focuses on standalone cloud services evaluation and is less concerned about the evaluation of cloud applications that integrate multiple cloud services. The goal will be to assess alternatives using at least three metrics. The more, the better. At least one metric must be a non-generic metric, unique to the service.

Generic metrics:

- Average service turnaround time for queries (single client x 100 runs)

- Average service turnaround time for concurrent queries (single client, from 10 to 100, step by 10)

  Testing concurrent query performance provides stress testing to find the processing limits of cloud services.

- Average service turnaround time for concurrent queries (2 to 10 parallel clients x 100 runs)

  Due to networking limitations, a single client can only supply a finite stress workload against backend cloud services. In practice, cloud services must support ***many*** concurrent distributed clients.

- Diurnal performance variation of average service turnaround time by hour for 24 hours (single client x 100 runs)

  Measuring diurnal performance variation will help characterize the performance behavior across a 24-hour day. We are interested in learning if there are patterns to performance behavior

throughout the day. For example, are their busy periods where performance degrades? What is the best time of day to accomplish work in the cloud? Diurnal performance experiments can be extended to a week, or month.

- Comparison of observed / measured hosting cost for experiment(s)
  Host costs include: data transfer, data hosting, and transaction costs…

  We would like to "reconcile" the bill for cloud services. By measuring resource utilization for the user's point-of-view, we would like to certify if indeed the cloud provider's bill is accurate. It should be. But who is actually checking?

Cloud services evaluation projects should help answer the question, "Why should I choose technology A over technology B for my cloud implementation?". This project will allow groups to explore alternative cloud service offerings, learn about their respective features, and share these findings with the entire class.

The following are potential comparison projects. This is not an exhaustive list of potential services comparison projects. Groups are encouraged to perform initial exploratory research to determine the cloud services they would like to explore.

The specific project recommendation assume a group size of 4 students. These project recommendations will scale with the number of students. Evaluation will take into account the "body of work" produced by the group over the course of the quarter. For example, if a given project has strengths in some areas, these can outweigh weaknesses in others.

1. Object/blob storage services comparison

Services: Amazon S3, Google blobstore, Azure blobstore, self-hosted key-value service on a single VM (minio, ceph, LeoFS, riak)

Project recommendations:

1. At least one self-hosted open source storage service should be considered.
2. At least three total services should be compared.
3. Testing should include concurrent read transactions from at least 5 client VMs in parallel.
4. Transactions must involve at least 10 different sizes of objects.

Questions to Explore:
Which blob storage service is most performant?
Which blob storage service is most cost effective?
How does the type and size of data impact performance and cost for different blobstores?

Non-generic metrics:
1. Eventual consistency (time): when an object is updated, how long does it take before all replicas are updated?
2. Eventual consistency (failures): when an object is updated, what % of reads immediately after the update reflect the new vs. old value?

3. Read data transfer rate: object storage typically involves transfer of large data objects. The transfer rate can be measured in KB/s, MB/s
4. Write data transfer rate in KB/s, MB/s

2. **Cloud relational database services comparison**

Amazon RDS, Amazon Aurora, Azure SQL Database, Google Cloud SQL, Heroku, Self-hosted RDBMS (e.g. PostgreSQL, MySQL, others)

Project recommendations:

1. At least one self-hosted open source RDBMS should be considered.
2. At least three services should be compared.
3. Queries performance must be tested against tables with at least 1,000,000 rows of data
4. *Testing of one system that scale horizontally with read replicas is excellent.
5. Several non-generic metrics should be measured

Questions to Explore:
What are the performance and cost tradeoffs for adopting different relational database cloud services?

For this project the team would locate a sample relational database online or construct an artificial database with synthetic data and conduct database performance experiments to evaluate the systems.

Non-generic metrics:
1. Update query average turnaround time for updating 1 to many rows
2. Read query with join average turnaround time
3. Row count query average turnaround time
4. Delete query average turnaround time

3. **NoSQL database services comparison**

DynamoDB, Azure Table, Google BigTable, Google NoSQL, Self-hosted NoSQL DB (e.g. MongoDB, Cassandra)

Specific Project recommendations:

1. At least one self-hosted open source NoSQL DB should be considered
2. At least three services should be compared.
3. Queries performance must be tested against data sets of at least 100,000 key/value pairs
4. Having at least one system scale horizontally with a read replica is excellent.
5. At least two non-generic metrics should be measured

Questions to Explore:
What are the performance and cost tradeoffs for adopting different NoSQL database cloud services?

For this project the team would create a sample dataset of key/value pairs. This dataset could be found on line or an artificial database with synthetic data could be created to conduct NoSQL DB performance experiments.

Non-generic metrics:
1. Time to create 1, 10, 100, 1,000, and 10,000 key/value pairs
2. Time to update 1, 10, 100, 1000 key/value pairs
3. Time to query key/value pairs
4. Time to delete 1, 10, 100, 1000 key/value pairs
5. Time for eventual consistency with replicated nodes

4. **In Memory Key-Value store services comparison**

ElastiCache, Azure redis, Google redis, Self-hosted NoSQL DB (e.g. redis, memcached)

Specific Project recommendations:

1. At least one self-hosted open source NoSQL DB should be considered
2. At least three services should be compared.
3. Queries performance must be tested against data sets of at least 100,000 key/value pairs
4. Having at least one system scale horizontally with a read replica is excellent.
5. At least two non-generic metrics should be measured

Questions to Explore:
What are the performance and cost tradeoffs for adopting different in memory key-value store cloud services?

For this project the team would create a sample dataset of key/value pairs. This dataset could be found on line or an artificial database with synthetic data could be created to conduct NoSQL DB performance experiments.

Non-generic metrics:
1. Time to create 1, 10, 100, 1,000, and 10,000 key/value pairs
2. Time to update 1, 10, 100, 1000 key/value pairs
3. Time to query key/value pairs
4. Time to delete 1, 10, 100, 1000 key/value pairs
5. Time for eventual consistency with replicated nodes

5. **Cloud application containers comparison for web application hosting**

Amazon Elastic Beanstalk, Heroku, Google App Engine, vs. self-hosted

This comparison will involve developing a web services application and deploying the application to a variety of PaaS services. (web containers)

Project recommendations:

1. At least one self-hosted application container such as Apache Tomcat deployed to a VM must be considered.
2. At least three services must be compared.
3. A minimum of two webservices must be built with different resource requirements (CPU, memory, disk I/O, network I/O) For example, if one service is CPU-bound, then another must be memory or disk I/O bound.
4. Having at least one system scale horizontally and leverage a load balancer to distribute requests to multiple compute nodes is excellent.

   Questions to Explore:
   What are the performance and cost tradeoffs for adopting different PaaS web container cloud services?
   How do the hosting costs compare?
   How does the characteristics of the application (e.g. memory bound, CPU bound) impact performance and cost?
   What is the maximum number of client requests that can be processed for each host VM before there is a performance dip? (consider how many CPU cores the VM has) In other words, when does the first scaling bottleneck occur when hosting using only a single VM? How many concurrent requests result in the first performance dip?

   Non-generic metrics:
   1. What is the average service throughput (requests/sec)?

6. **Serverless computing comparison**

   AWS Lambda, Azure Functions, Google Cloud Functions, IBM OpenWhisk, Iron.io

   Project recommendations:

   1. At least two serverless computing platforms must be compared.
   2. The language used for two serverless computing platforms must be the same.
   3. A minimum of three microservices should be built with different resource requirements (CPU, memory, disk I/O, network I/O) For example, if one service is CPU-bound, then another must be memory or disk I/O bound.
   4. One or more of the following should be considered:
      a. Performance difference of asynchronous vs synchronous calls
      b. Performance difference of HTTP/REST vs Command-Line-Interface
      c. Performance of a chained call, where a microservice calls itself or another microservice hosted by the same cloud provider before returning a response to the client

   Questions to explore:
   How responsive are different serverless computing platforms to increasing demand? How quickly can they scale?
   Does one particular serverless computing platform appear better for CPU-bound, memory-bound, network I/O-bound, or disk I/O-bound activity?

For self-hosted services, the intent is to compare a cloud-hosted variant with a self-hosted software application running on a virtual machine on the cloud to provide an “equivalent service”. For example,

redis and mongodb can provide key-value storage as an application installed onto a virtual machine. From a functional point-of-view this is equivalent to S3, but with a slightly different API for access.

Other cloud services comparison projects:

1. Cloud Container Services comparison: Amazon Elastic Container Service (ECS), Azure Container Service (ACS), Azure Kubernetes Service (AKS), Google Kubernetes Service
2. Cloud Container Service vs. self-hosted container orchestration on VMs (Docker SWARM or Kubernetes)
3. Amazon Elastic Load Balancer vs self-hosted load balancer (haproxy or nginx)
4. Cloud virtual machine performance comparions: AWS, Azure, Google
5. Storage system comparison: Amazon EBS, Amazon EFS, Amazon Ephemeral Storage
6. Message queueing services comparison

Option #2 – Integrated Cloud Services Evaluation

In contrast to option #1, option #2 focuses on performance and cost comparison of a system that integrating two or more cloud services. Ideally this would be a web services project, perhaps using Serverless Computing microservices where multiple competing cloud data services are compared.

The result of this evaluation will consider the overall performance of the system while integrating competing components internally. Option #2 can be thought of as simply an extension of option #1 where cloud microservices are the cloud service clients instead of a personal client such as a bash script using curl, or the use of JMeter.

Project recommendations:

1. The project must integrate at least two cloud services. Integrating three or more would be excellent.
2. At least one of the cloud services must be pluggable, and the evaluation must consider overall performance with at least two competing cloud services that are “plugged in”. For example, a set of microservices can be built which exercises competing cloud data services. Integrated performance and cost will be measured for the alternate system architectures.
3. The comparison must consider at least three metrics. One should be considered non-generic or specific to the system. More metrics is better.

Option #3 – Serverless Computing Microservices Composition

Serverless computing platforms allow up to 256MB of code to be deployed as a single microservice. This enables a fairly large amount of code for simple microservices. The idea behind option #3 is to experiment with composition of microservices on serverless computing platforms such as AWS Lambda. Groups will locate, or develop a set of related microservices that perform sequential processing on images (i.e. image filters) or on data (i.e. data transformation). For example, imagine that a given data file was to be processed with 3 sequential steps each producing a new transformation of the data. If each transformation was supported by an independent microservice, then the data payload must be exchange over the network between intermediate stages. Alternatively, if the code to perform the transformations is simple, all steps could be hosted by a single service eliminating the need for intermediate transformation steps.

For option #3, groups will deploy alternate service compositions for sequential data and image transformation tasks to compare the performance and cost tradeoffs. Ideally there would be no more than three or four transformation steps so that brute force comparisons are feasible. At least three metrics such as average service execution time, average service throughput, and hosting costs should be quantified.

The project will quantify the gap between the best and worst case service performance for alternate compositions. We are interested in knowing to what degree composition matters for microservices deployment to serverless computing platforms.

Option #4 – Propose your own Cloud Project

For TCSS 562, groups are free to propose their own project by consulting with the instructor. Projects must use the cloud and include an evaluation typically of performance and/or cost. Projects may compare alternate architectures though this may not necessarily be required. The complexity should and effort should be similar to the other potential projects.

1 Proposal Requirements

[5% of course grade]

The following are key requirements of the project proposal:

Each team will submit a 1 to 2 page short project proposal description.

The proposal must identify:

1. The member names of the project group.
2. The name of the group project coordinator. The project coordinator will be responsible for scheduling and arranging group meetings and work sessions, creating agendas for project check-ins for TCSS 562, and ensuring that tasks are assigned to group members.
3. The topic. If option #1, #2, or #3, then the proposal should suggest more specifically what systems will be investigated and which metrics will be evaluated. It is ok to make changes as needed later on in the project as needed.
4. For option #4, at least 2 cloud systems research or evaluation questions that will be considered by the project. It is ok to make changes to the questions as needed later on.
5. For option #4, at least 2-3 metrics that will be evaluated based on the project that is implemented. It is ok to make changes as needed later on throughout the project.
6. At least 3 references for each project. These could include research papers related to the clod technologies being used, white papers, and/or extensive documentation or blog websites concerning the use of the cloud systems. The references, for example, could discuss the capabilities (features) of the systems, and may remark on details regarding their associated performance and cost.

If available, good proposals will:

1. Identify prior performance comparison studies which relate to the proposed project. These comparisons could be research papers or blog articles. Use scholar.google.com to search for

research papers. Research papers published at IEEE, ACM, or USENIX conferences or journals are rigorously peer-reviewed and are generally considered of higher quality.

2 Future Deliverables

The final project will involve a group project presentation during the final exam session on Wednesday June 6th. Requirements of the final project presentation will be provided later on.

The final project will also involve a written report in IEEE conference format. In the project report, groups should be prepared to describe their project, identify the evaluation questions, describe any prototyping, and benchmark tests, and create tables and graphs to describe results of the evaluation. Project reports will also describe any related work or comparison studies, why the evaluation questions are relevant and interesting, and reflect on the results of the study. Additional details and requirements for the final project report will be provided later on.

3 Project Check-ins (20% of the TCSS 562 course grade)

There will be three "written" project check-ins throughout the quarter roughly at two-week intervals. The project-checkins are grouped in the same category as tutorials, class activities, and quizzes for TCSS 562. Groups are encouraged to meet with the instructor before/after class, during office hours, or by scheduling an appointment to seek clarification and for assistance.

4 Submission Deadline

Project proposals should be submitted in PDF format on Canvas no later than 11:59pm on Thursday April 12th. Projects proposals will be approved or revisions requested by Sunday April 15th. Groups will have ~51 days, or just over 7 weeks to complete projects by Wednesday June 6th.

Change History

Version	Date	Change
0.1	04/01/2018	Original Version