[[Marco : <title>]]

Thank you, Klaas. So I am Marco, and this is Frans.

I trust everyone here has at least heard of Cloud Computing

You don't build and maintain your own computing hardware,
but use the existing infrastructure of a cloud provider.

[Cloud Computing]

It has many advantages, like reduced cost, and ease of use.

Cloud providers operate on a pay-as-you-go basis. You only pay for the resources your application used.

There are drawbacks as well. It is difficult to guarantee security and privacy when you don't have full control over the platform.

Nevertheless, cloud computing has gotten very popular over the past decade.

With the rising demand for cloud computing, the supply of cloud providers has increased as well.

Amazon, Microsoft and Google are just a few of the many companies offering public cloud services.

[Choices, choices, choices]

The different cloud providers have different advantages and drawbacks.

Some, for example, offer high performance, at a high cost. While others give more security, but are also more difficult to use.

On the other hand, applications can differ a lot in what they require too.

Maybe your application needs a lot of CPU performance, but not much storage.

Or you need support for a specific database system, but network performance is not as important.

[Comparison Techniques]

There are many choices of public cloud providers. Too many, you might say.

And you want the one that suits your requirements at the most attractive price.

Fortunately, there are tools and techniques to help with that.

We will discuss three of these techniques today:
CloudCmp,
AHP,
and TOPSIS

[CloudCmp]

First up is CloudCmp.

CloudCmp was developed and first tested by Li and others in two thousand ten.

Their goal was to create a cloud comparison tool that can be applied as broadly as possible.

To this end they have identified three common services:
the elastic compute cluster,
persistent storage,
and the intra-cloud and wide-area networks.

For each of these services they measure a number of performance metrics.

They ran CloudCmp on Amazon Web Services, Microsoft's Azure,
the Google App Engine and CloudServers.

For legal reasons the results are anonymized, so the cloud providers are identified as C-one through C-four.

Additionally, three of the providers offer different tiers.

Tier one is the cheapest and higher tiers are more expensive, but offer more performance.

Provider C-three is the exception to this and offers only one performance tier.

In fact, we will see C-three being the exception to quite a few tests of CloudCmp.

[Compute cost per task]

To measure the elastic compute cluster, they ran three benchmark tasks and measured how fast they finish.

One task measures the CPU, the other is more memory-intensive and the third focuses on local disk input/output.

In the graphs here we can see the finishing time per task.

Multi-threading and local disk performance are absent for C-three because they do not support it.

As you can see, there is quite some difference between the providers, but not a lot between the different tiers of the same provider.

This is different when we look at the cost per benchmark. It turns out that, at least for these providers, low-tier providers are more cost-effective.

So it's more beneficial to buy many low-tier instances, than a few high-tier ones.

// The authors of CloudCmp explain this by saying the physical instances they
// were running on were likely not used by anyone else. So the lower tier
// instances were not being throttled.

The authors of CloudCmp also measured the scaling latency. That is the time it takes to create more instances when requested.

[Storage]

For persistent storage, the authors of CloudCmp look at three storage services: table, blob and queue.

The table service offers a way to store structured data, like a traditional relational database.

CloudCmp compares it by measuring the times for get, put and query operations.

Except on C-two, which has no table service.

They also looked at the time to consistency: how long it takes to read back data that was just put there after a put operation.

While none of the providers guarantee strong consistency in their table service, when the authors of CloudCmp measured it they only found C-one to exhibit a noticable delay in consistency.

Blob storage allows for large amounts of unstructured data, like images, audio or other binary data.

It was measured by up- and downloading blobs of 1 kilobyte and 10 megabytes.

Except on C-three, for the usual reason.

And, finally, a queue can be used to pass small messages between different instances in the same cloud.

Only C-one and C-four offer a queue service.

They measured the time it takes to send a message to the queue, how long it takes
to retrieve a message and the propagation delay:

the time between sending and retrieving one message from one instance to another.

// They found that it takes about the same amount of time as up- and downloading a
// small blob.

[Network]

The last service is network.

CloudCmp looks at intra-cloud and wide area traffic: with the rest of the internet.

Intra-cloud traffic was not measured on C-three as they do not allow direct communication between instances.

They tested the intra-cloud network with standard tools, like ping and iperf.

C-two turned out significantly slower than C-one and C-four.

To test the wide area network, the CloudCmp authors used the PlanetLab network to connect to the cloud instances from locations all over the world.

C-three was the fastest in the wide-area network tests.

[Case studies]

These benchmarks are all well and good, but as we all know, real-world applications are more complex.

In order to verify that the benchmarks really are indicative of a cloud provider's performance, Li et al also ran a few case studies.

In these case studies they deployed more complex applications to the cloud providers that are more representative of real-world applications.

The three cases they studied are an E-commerce website, a scientific computation application and a latency-senstive website.

[TPC-W]

The E-commerce website is implemented with TPC-W: a widely used benchmark for e-commerce websites.

It mainly uses table storage, and therefore could not be deployed on C-two as they do not offer that.

To measure the performance they look at the page generation time.

The results are in line with CloudCmp's measurements of the table performance.

C-one is fastest overall and C-three is faster with query operations, whereas C-four is faster with gets and puts.

[Blast]

The second case study is performed with Blast. A parallel computing application for DNA alignment.

The main objective is to measure computing performance. However, as Blast requires a queue service to distribute jobs, it could only be deployed on C-two and C-four.

In line with the elastic compute measurements, C-four is in all cases faster than C-one.

[Latency]

The last case study is the latency sensitive website.

The authors of CloudCmp created a simple web server that serves static pages of two sizes: a small one of one kilobyte and a large page of 100 kilobytes.

The goal is to download the pages as fast as possible.

They download the pages from various locations around the world, using PlanetLab.

C-three is the fastest for both page sizes.

As expected, since C-three was also the fastest in the CloudCmp network tests.

[<concluding>]

CloudCmp is a very broad tool.

Although, as we have seen in the examples, there is still quite a bit of disparity between the features that providers offer.

Even in such a seemingly broad set of functionality, some providers have either significantly different behaviour or lack the features altogether.

Nevertheless, CloudCmp gives detailed insight into raw performance metrics of cloud providers.

And these measurements also correspond to the performance a real-world application may expect. As the case studies have shown.

In addition to providing insight to prospective cloud users, CloudCmp also helps the cloud providers.

They can use CloudCmp to look at bottlenecks or weak points in their services.

And to see how they can improve their pricing models to generate more income, or make their service more attractive to users.

However, CloudCmp does not help much when an application needs specific functionality.

And this is where the other two techniques come in.

[[Frans : AHP]]

Another technique that we can use to select an ideal cloud provider is Analytical Hierarchy Process or AHP.

This technique was first introduced by Saaty while directing research projects in the US government.

This is really a popular technique for dealing with complex decision making since it can be applied to many different cases.

You can use it to choose a house to rent, a university to study, or even for choosing a wife or husband!

So basically, this technique applies a mathematic approach to any decision that needs to be made.

Here are the steps of AHP:

For the purpose of the analysis with AHP model, we followed the experiment conducted by Mingrui.Sun et.al from the Harbin Institute of Technology in China.

The goal of the experiment was to select a cloud service for a health medical rehabilitation system.

First, you define the objective.

Your goal must be clear in order to avoid any ambiquity.
In our case, the goal is to select an ideal cloud provider.

After that, you define he criteria corresponding to the goal.

In cloud selection case, the criteria could be the response time, availability or cost.

Or if your criteria are based on the attributes of the cloud providers, you can define the criteria such as the price/hour, virtual core or memory.

Now that we have defined the criteria, we can perform the pair comparison.

You can rate the comparison from equal to extremely strong, as you can see on the screen.

Figure → rating

For example: *explain the image*

Figure → pairwise comparison

After you make a pair comparison for each criteria, you transform the results into a square matrix.

Figure → square matrix of pairwise comparison
explain the image

After we transform the results into a square matrix, we can obtain the weight of each criteria by applying the principal eigen vector.

So, how do we do that?

Well, first we must sum the value of each column in the matrix.

Figure → pairwise comparison

Then, divide the value of each matrix with the sum of its column.

Figure → pairwise comparison

Then, compute the normalized principal eigen vector by obtaining the average of each row multiply by fraction of the total number of criteria.

Figure → pairwise comparison

Finally we have obtained the weight of each criteria. Now let's verify it by checking the consistency of the judgments before we apply it on each alternative.

The first thing to check the consistency is calculating the lambda max. You can do it by multiplying the weight of each criteria with the sum of each column of reprocical matrix first and after that get the sum of the results.

Second, we must compute theconsistency ratio by subtracting the value of lambda max with the total number of criteria divided by n-1.

And finally we can obtain the consistency ratio by dividing the degree of consistency with the consistency ration. We can refer to the table for the consistency ratio.

Figure → consistency ratio

The rules for the consistency ratio is

"If the value of consistency is smaller than or equal to 10 %, the subjective judgement is consistent".

We see that the value is below 10 %, therefore the judgement that we made

consistent.

Now we can apply the value same pattern for the alternatives layer.

Figure → alternative layers

Finally, after obtaining the the weight for each alternatives layer, we can get the final results as shown on the screen.

Figure → results

[TOPSIS]

Another technique for multicriteria decision making is "Technique for Order Preference by Similarity to Ideal Solution" or TOPSIS.

This technique was introduced by Hwang and Yoon in 1981.

The basic principle is that the selected alternative should have the shortest distance from the ideal solution and the farthest distance from the negative ideal solution in a geometrical sense.

So basically, this technique is a bit similar to AHP. It applies a mathematic approach to any decision that needs to be made. In order to perform TOPSIS, we conducted an experiment based on the case study of AHP.

These the steps of TOPSIS:

First, determine the decision problem and identify the relevant evaluation criteria as we did with AHP.

Then, we develop the preference for the criteria by assigning weights. In this case, we can use the weights from AHP.

The next step is construct the Decision Matrix for the alternatives based on the criteria. We must convert all the values into numbers ranging from 1 to 9.

Figure → Decision Matrix

Then Calculate a normalized decision matrix

Figure → Normalized Decision Matrix

Compute the weighted normalized decision matrix (NDM) by multiplying weights of criteria with the corresponding alternatives value.

Figure → NDM

Identify the Positive Ideal Solution

Figure → PIS

Identify the Negative Ideal Solution

Figure → NIS

Then, Compute separation of each criteria value for each alternative from both ideal and negative ideal solution.

Figure → Sep

Finally, measure the relative closeness of each location to the ideal solution.

Figure → Rel closeness

[[Marco : Conclusions]]

We have looked at three techniques for comparing cloud providers: CloudCmp, AHP and TOPSIS.

CloudCmp is a very broad tool.

It gives detailed insight into raw performance metrics of cloud providers.

But it does not help much when an application needs specific functionality.

AHP and TOPSIS are more complex and will actually help you make the decision, rather than only providing a comparison.

However, because of their complexity they are also far more difficult and time-consuming to use.

Our final recommendation is that there is no one solution that fits all.

The best way to compare cloud providers is to use a combination of techniques.

One to pick a providers that offer the functionality you need and the other to find the best deal among those.

Thank you very much for your attention.