Improving the Efficiency of Cloud Computing

Matthew Perrault

University of Minnesota, Morris

April 28, 2014

Outline

- Why cloud computing?
- Efficiency through indexing and queries
- Overbooking
- Conclusions

Importance of cloud computing

- Cloud computing provides users with a large amount of powerful and reliable resources.
- Storage and computation are the two main areas of cloud computing.
- Cloud services are rented out based on specific service-level agreements (SLAs).

Databases

- A database (DB) is a large collection of data organized for easy lookup on specific data.
- A query is the process of 'asking' a DB for information.
- Indexes help label and organize data so that a query does not have to search through all information in a DB.

Virtual Machines

- A virtual machine (VM) is a software based emulation of a physical computer.
- VMs can provide a complete system platform supporting execution of an operating system.
- Several VMs can be run from a single physical computer.

Extensible Markup Language (XML)

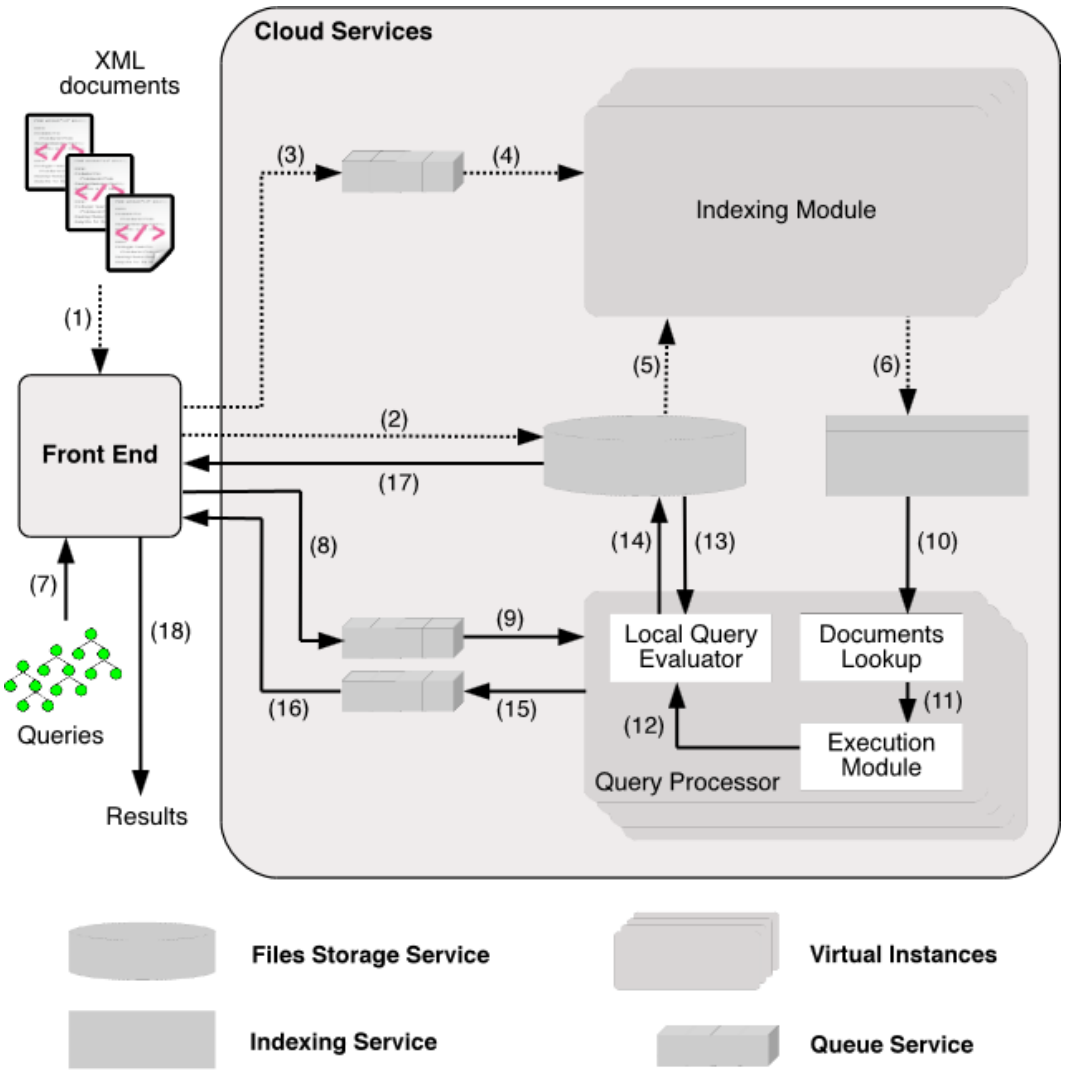
- XML is a language that is designed to transport and keep data organized.
- XML can structure and organize data with tags that are not predefined.

```
<painting id="1854-1">
  <name>"Olympia"</name>
  <painter>
    <name>
      <first>"Edouard"</first>
      <last>"Manet"</last>
    </name>
  </painter>
</painting>
```

Efficiency through indexing and queries

- The efficiency of a cloud system refers to the response time it takes for the user to receive requested data.
- Manipulating how indexes are stored through indexing strategies can result in faster and more accurate queries.

Cloud Architecture used by Amazon Web Services (AWS)

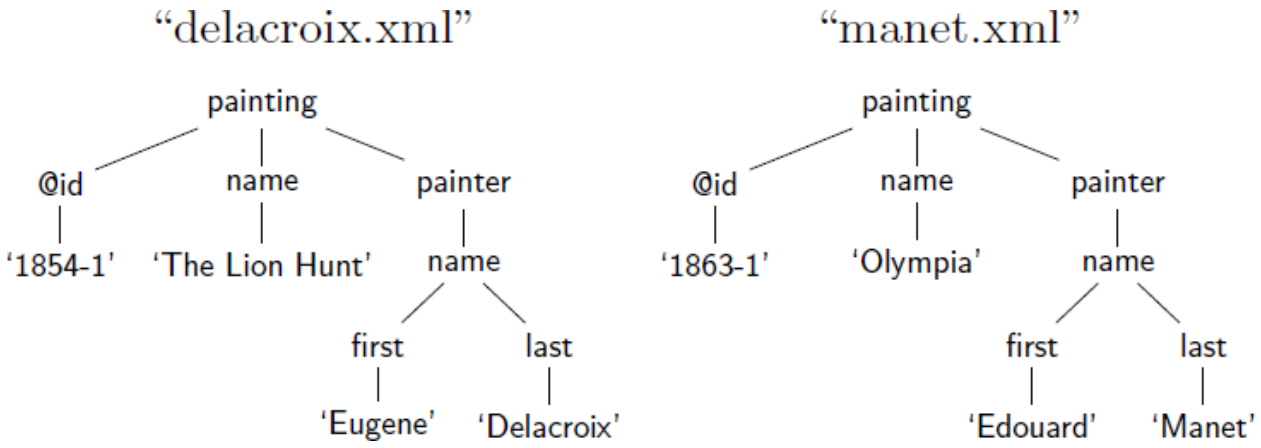


Indexing Strategies

- Uniform Resource Identifier (URI) is a string of characters that are used to identify or locate a name or ID.
- For a given node n in a document, the function *key(n)* computes a string key based on which n's information is indexed.
- Let $\underline{e}$, $\underline{a}$ and $\underline{w}$ be three constant string tokens, and $\|$ denote string concatenation.
- *key(n)* is defined as:

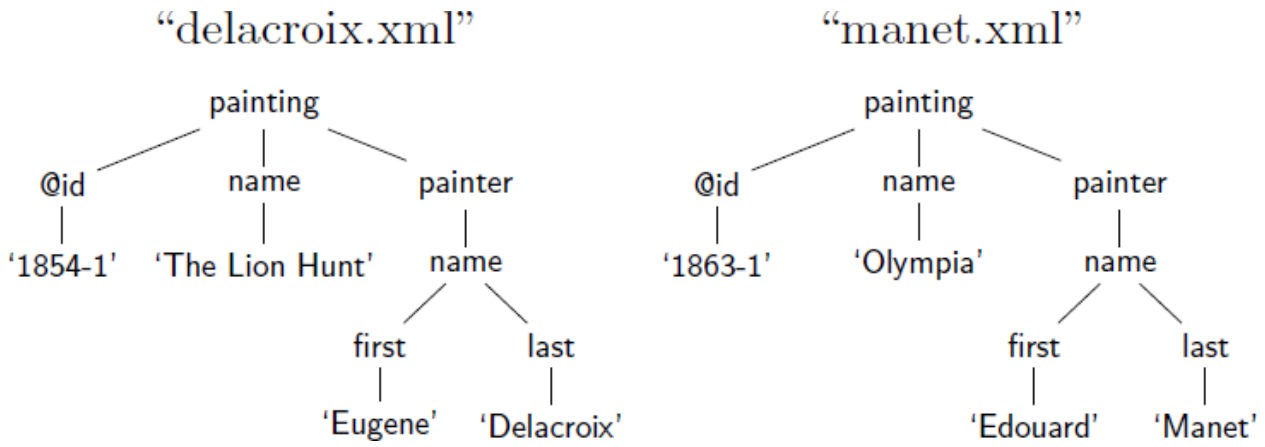
$$key(n) = \begin{cases} \underline{e}\|n.label & \text{if } n \text{ is an XML element} \\ \underline{a}\|n.name & \\ \underline{a}\|n.name\ n.val & \text{if } n \text{ is an XML attribute} \\ \underline{w}\|n.val & \text{if } n \text{ is a word} \end{cases}$$

Strategy LU (Label-URI)



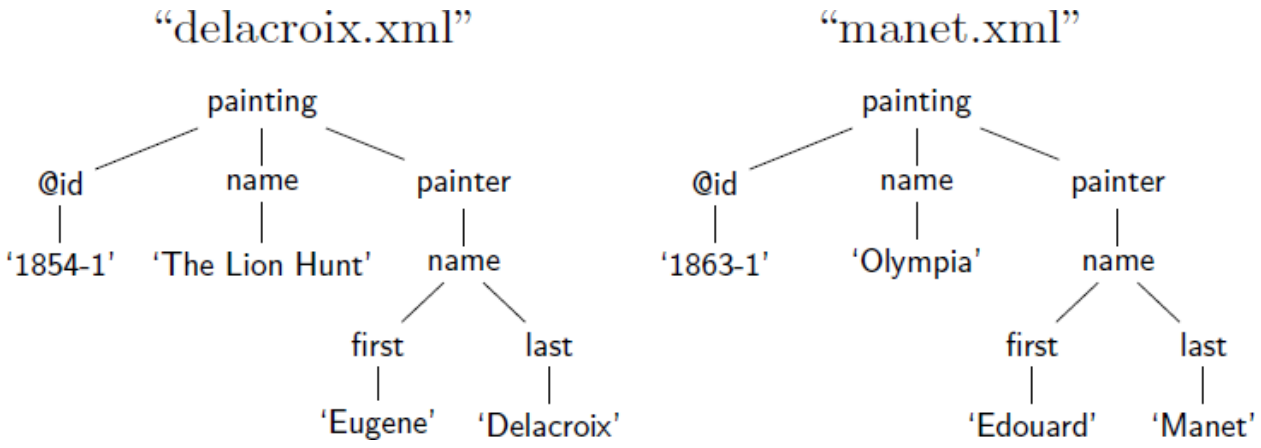
key	attribute name	attribute values
$\underline{e}$name	"delacroix.xml"	ϵ
	"manet.xml"	ϵ
$\underline{a}$id	"delacroix.xml"	ϵ
	"manet.xml"	ϵ
$\underline{a}$id 1863-1	"manet.xml"	ϵ
$\underline{w}$Olympia	"manet.xml"	ϵ

Strategy LUP (Label-URI-Path)



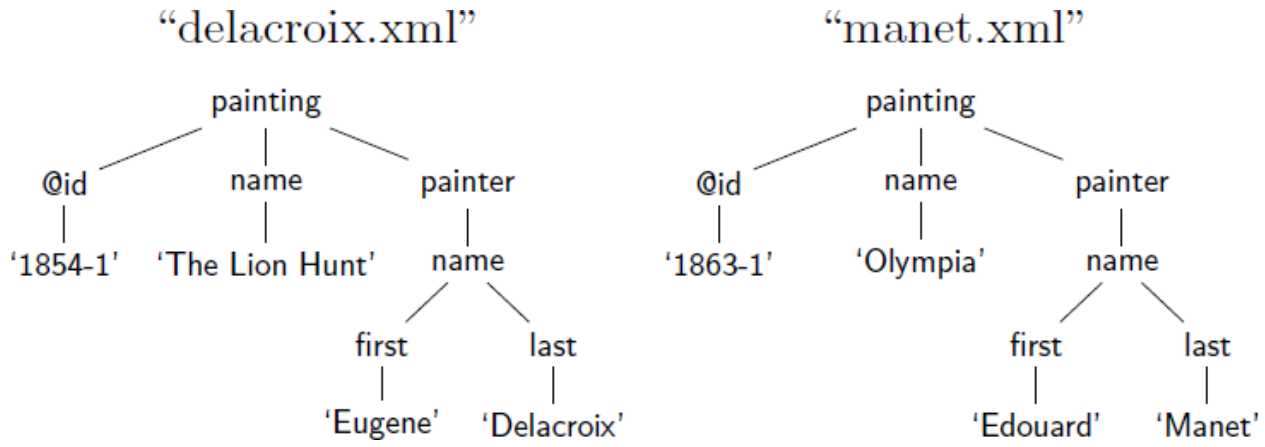
key	attribute name	attribute values
$\underline{e}$name	"delacroix.xml"	/$\underline{e}$painting/$\underline{e}$name, /$\underline{e}$painting/$\underline{e}$painter/$\underline{e}$name
	"manet.xml"	/$\underline{e}$painting/$\underline{e}$name, /$\underline{e}$painting/$\underline{e}$painter/$\underline{e}$name
$\underline{a}$id	"delacroix.xml"	/$\underline{e}$painting/$\underline{a}$id
	"manet.xml"	/$\underline{e}$painting/$\underline{a}$id
$\underline{a}$id 1863-1	"manet.xml"	/$\underline{e}$painting/$\underline{a}$id 1863-1
$\underline{w}$Olympia	"manet.xml"	/$\underline{e}$painting/$\underline{e}$name/$\underline{w}$Olympia

Strategy LUI (Label-URI-ID)



key	attribute name	attribute values
$\underline{e}$name	"delacroix.xml"	$(3,3,2)(6,8,3)$
	"manet.xml"	$(3,3,2)(6,8,3)$
$\underline{a}$id	"delacroix.xml"	$(2,1,2)$
	"manet.xml"	$(2,1,2)$
$\underline{a}$id 1863-1	"manet.xml"	$(2,1,2)$
$\underline{w}$Olympia	"manet.xml"	$(4,2,3)$

Strategy 2LUPI (Label-URI-Path, Label-URI-ID)

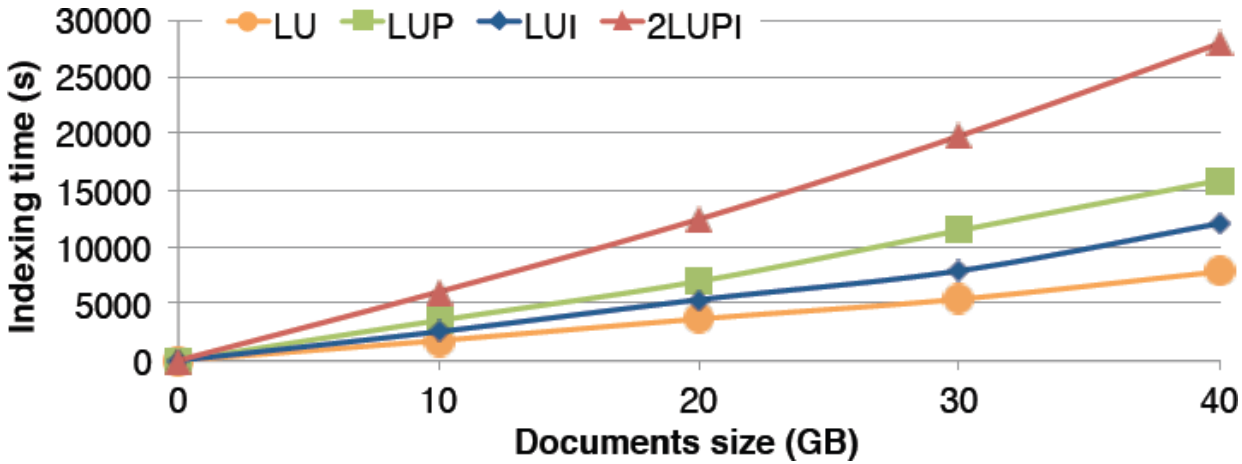


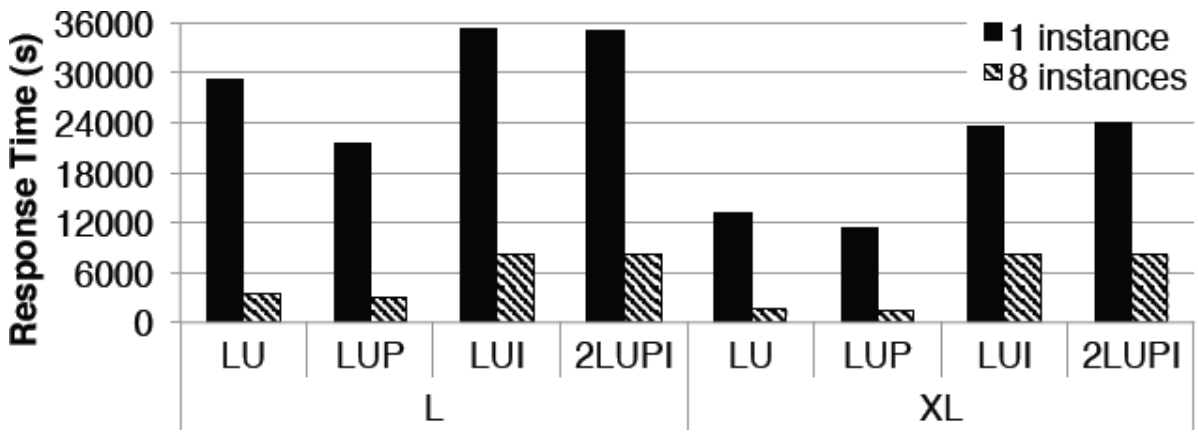
key	attribute name	1st attribute values	2nd attribute values
$\underline{e}$name	"delacroix.xml"	/$\underline{e}$painting/$\underline{e}$name, /$\underline{e}$painting/$\underline{e}$painter/$\underline{e}$name	$(3,3,2)(6,8,3)$
	"manet.xml"	/$\underline{e}$painting/$\underline{e}$name, /$\underline{e}$painting/$\underline{e}$painter/$\underline{e}$name	$(3,3,2)(6,8,3)$
$\underline{a}$id	"delacroix.xml"	/$\underline{e}$painting/$\underline{a}$id	$(2,1,2)$
	"manet.xml"	/$\underline{e}$painting/$\underline{a}$id	$(2,1,2)$
$\underline{a}$id 1863-1	"manet.xml"	/$\underline{e}$painting/$\underline{a}$id 1863-1	$(2,1,2)$
$\underline{w}$Olympia	"manet.xml"	/$\underline{e}$painting/$\underline{e}$name/$\underline{w}$Olympia	$(4,2,3)$

Testing Environment for Indexing Strategies

- Researchers used the Amazon Elastic Compute Cloud (Amazon EC2) within the Amazon Web Services framework to test the indexing strategies.
- Amazon EC2 provides virtual computing environments, which are referred to as instances.
- There are two types of EC2 instances used:
  - Large (L) with 7.5 GB of RAM and 4 CPUs.
  - Extra Large (XL) with 15 GB of RAM and 8 CPUs.
- 20000 XML documents were generated to test the indexing strategies.

Indexing Strategies Results



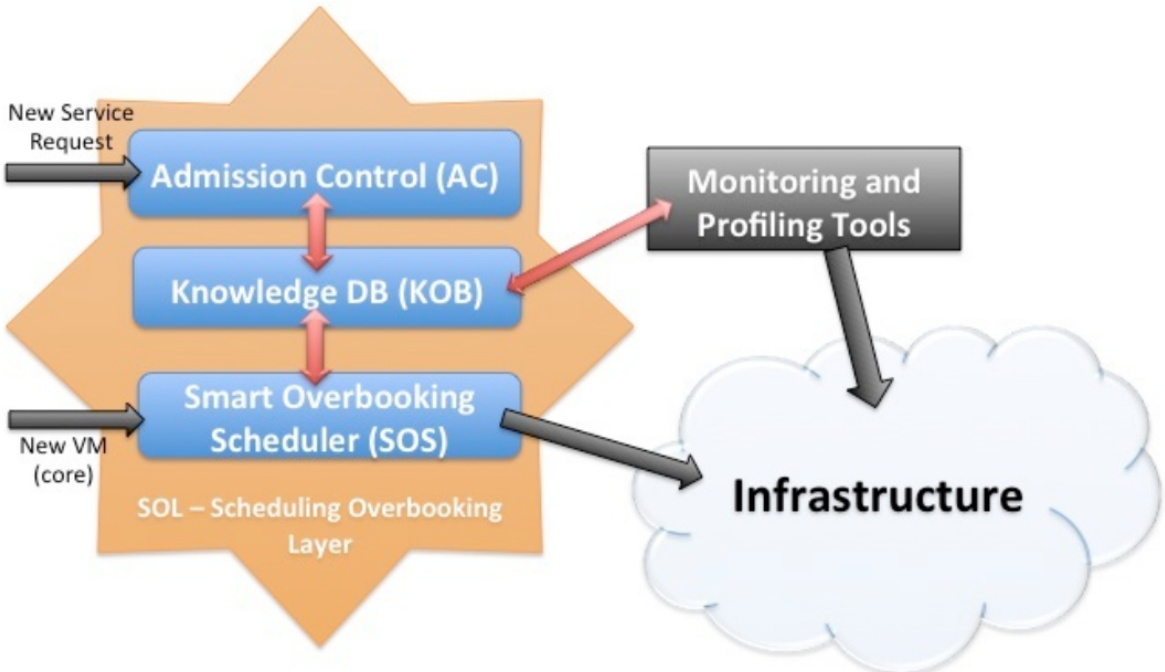


Overbooking

- Overbooking is a common method used by hotels and airlines to minimize resource waste.
- It allows the requested number of resources to exceed the actual amount of resources available.
- Users of cloud systems often overestimate the amount of resources they will need, meaning there are unused resources that could be applied elsewhere.

Overbooking

The overbooking method for cloud computing is comprised of three main parts that interact together to minimize resource waste.



The Knowledge DB (KOB) is a DB that stores information about past interactions with new applications and virtual machines.

Admission Control

- Admission Control (AC) is given the current number of VM's available through the KOB.
- AC ensures that overbooking does not guarantee incoming applications resources that are not available.
- If AC determines that it has the available resources, then it allocates the requested resources and accepts the incoming application.

Smart Overbooking Scheduler (SOS)

- Once an application is accepted, SOS is in charge of allocating the most suitable VM for that application.
- Based on the real resource usage and the demands of the application, SOS predicts the future expected usage of resources.

Smart Overbooking Scheduler (SOS)

- The overbooking factor (OBF) is a variable that allows us to measure how overbook-able the resources are:

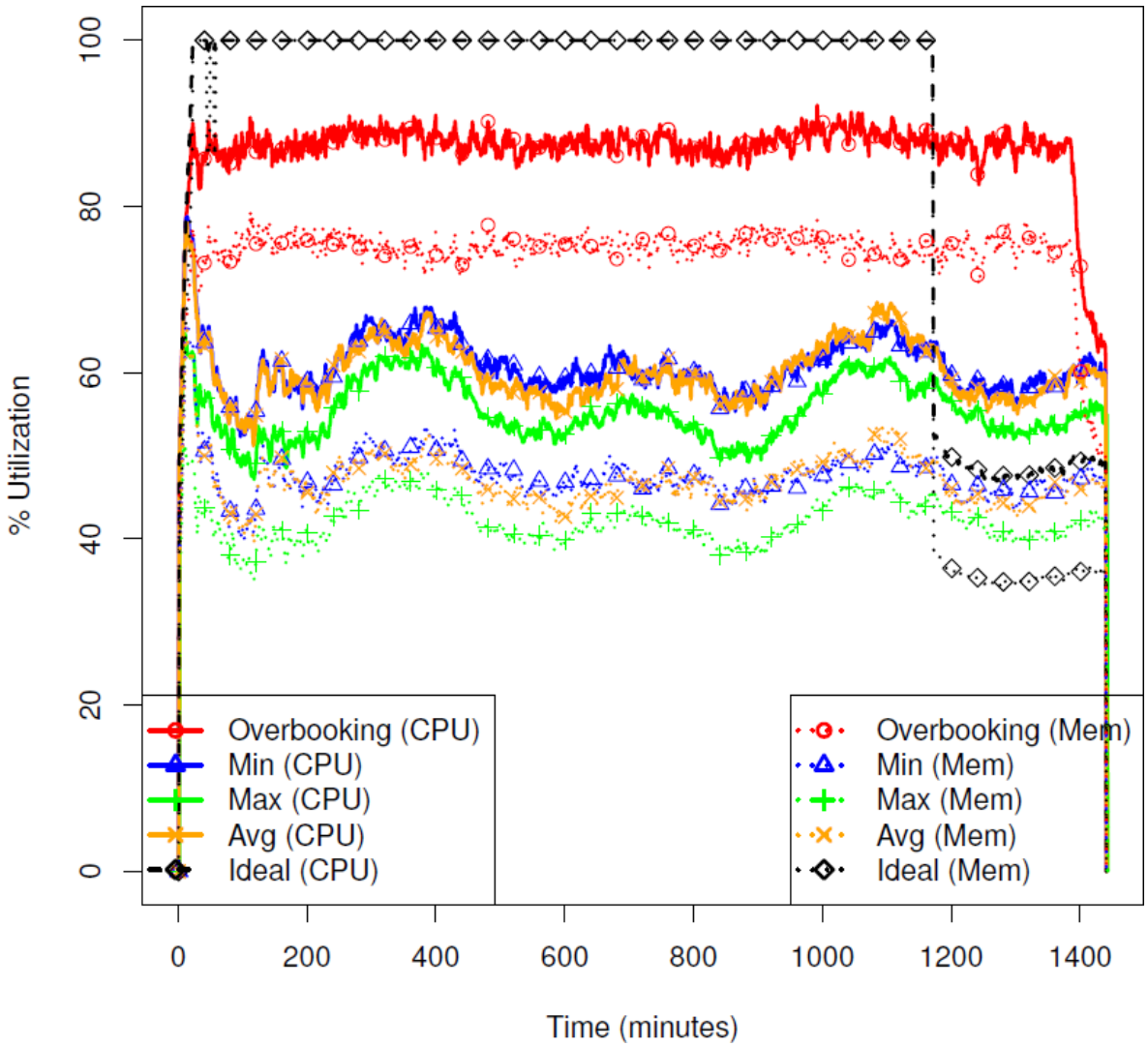
$$OBF_X = \frac{(UsageRequested_X - RealUsage_X)}{min(UsageRequested_X, RealCapacity_X)}$$

- The range of *OBFx* is (0,1), where the larger values indicate that there are resources that need to be overbooked.
- SOS chooses the largest OBF value and then allocates the appropriate number of VMs to the application.
- The x value refers to either CPU, memory, or I/O.

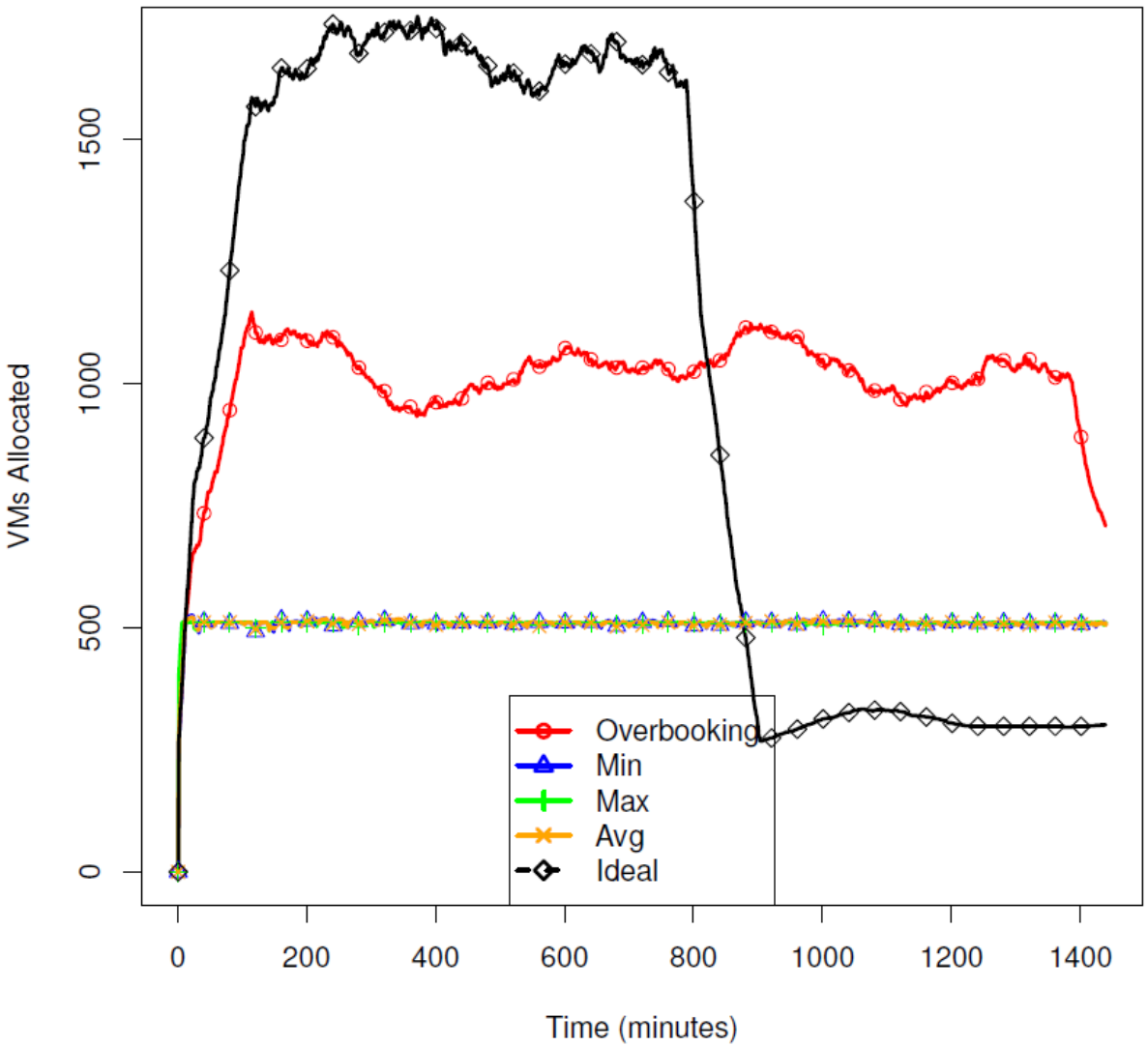
Testing Environment for Overbooking

- To simulate an actual cloud environment, the researchers used a cloud framework that is sent a large amount of applications.
- The incoming applications consist of two different types:
  - Bursty applications - vary in their resource demands over time.
  - Steady applications - have constant resource demands.
- The researchers also used 3 different sizes of virtual machines:
  - Small - 2 GB of RAM with 1 CPU.
  - Large - 4 GB of RAM with 4 CPUs.
  - Extra Large - 8 GB of RAM with 8 CPUs.

Memory and CPU usage



Allocated Virtual Machines



Conclusion

- Overall, the results showed improvements in both areas of storage and computation.
- Indexing strategies improved both query response times, as well as the accuracy of query results.
- Overbooking was shown to vastly reduce resources that are wasted when non-overbooking methods are used.
- The efficiency and demand of cloud computing will continue to grow in the future.