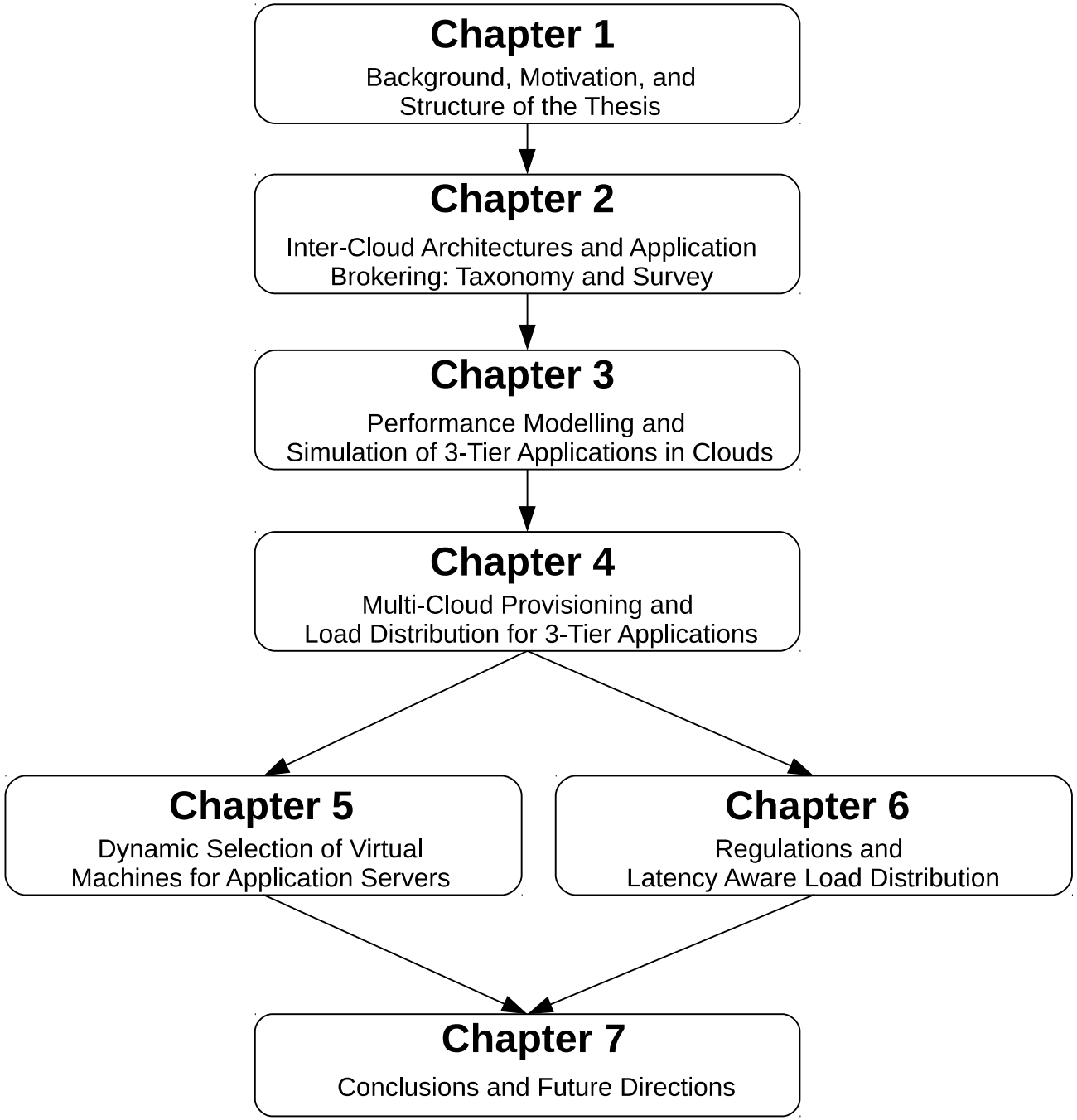
Brokering Techniques for Managing Three-Tier Applications in Distributed Cloud Computing Environments

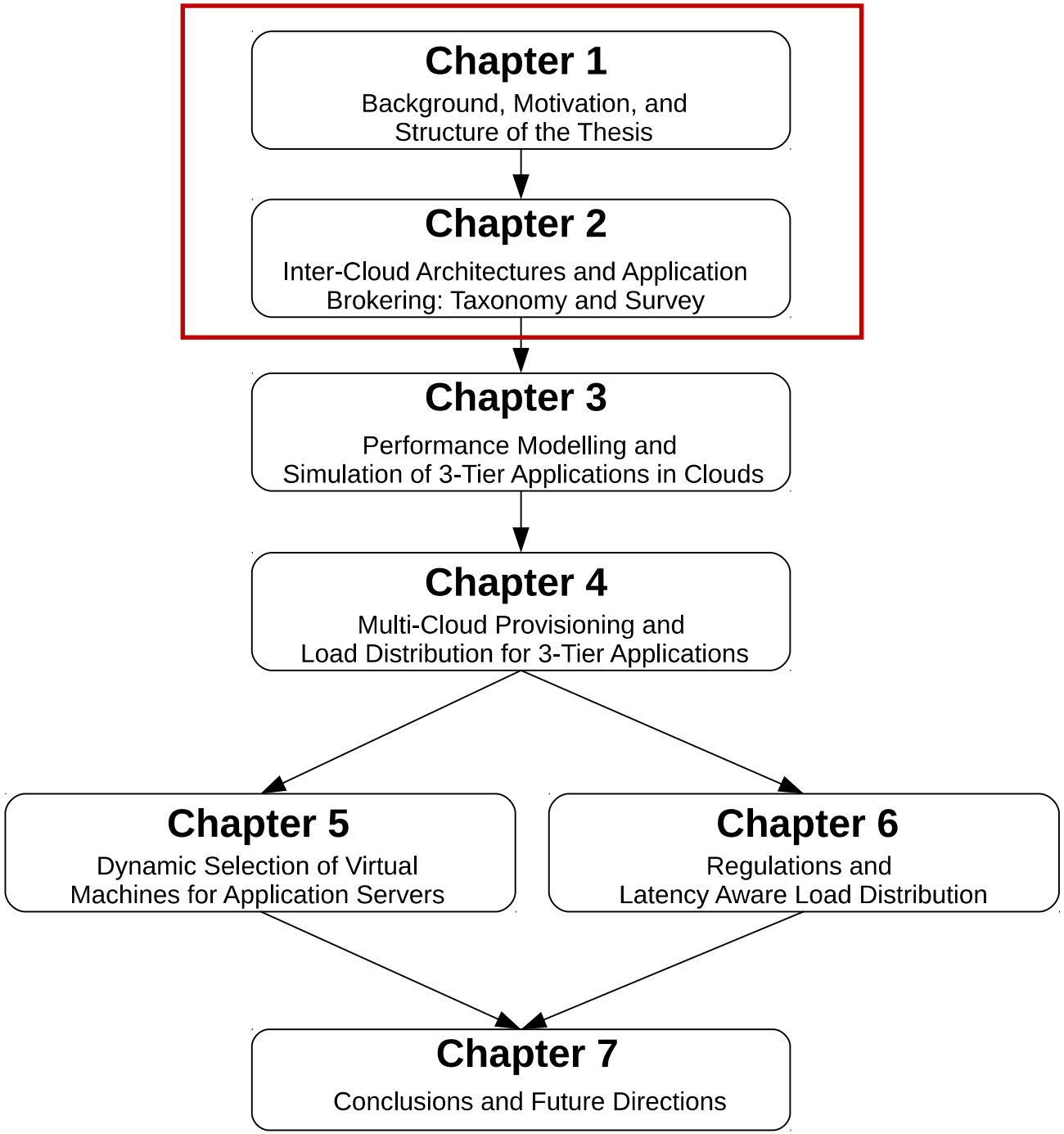
Nikolay Grozev

Supervisor: Prof. Rajkumar Buyya

7th October 2015

PhD Completion Seminar


Chapter 1
Background, Motivation, and Structure of the Thesis
Chapter 2
Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey
Chapter 3
Performance Modelling and Simulation of 3-Tier Applications in Clouds
Chapter 4
Multi-Cloud Provisioning and Load Distribution for 3-Tier Applications
Chapter 5
Dynamic Selection of Virtual Machines for Application Servers
Chapter 6
Regulations and Latency Aware Load Distribution
Chapter 7
Conclusions and Future Directions


Chapter 1
Background, Motivation, and Structure of the Thesis
Chapter 2
Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey
Chapter 3
Performance Modelling and Simulation of 3-Tier Applications in Clouds
Chapter 4
Multi-Cloud Provisioning and Load Distribution for 3-Tier Applications
Chapter 5
Dynamic Selection of Virtual Machines for Application Servers
Chapter 6
Regulations and Latency Aware Load Distribution
Chapter 7
Conclusions and Future Directions

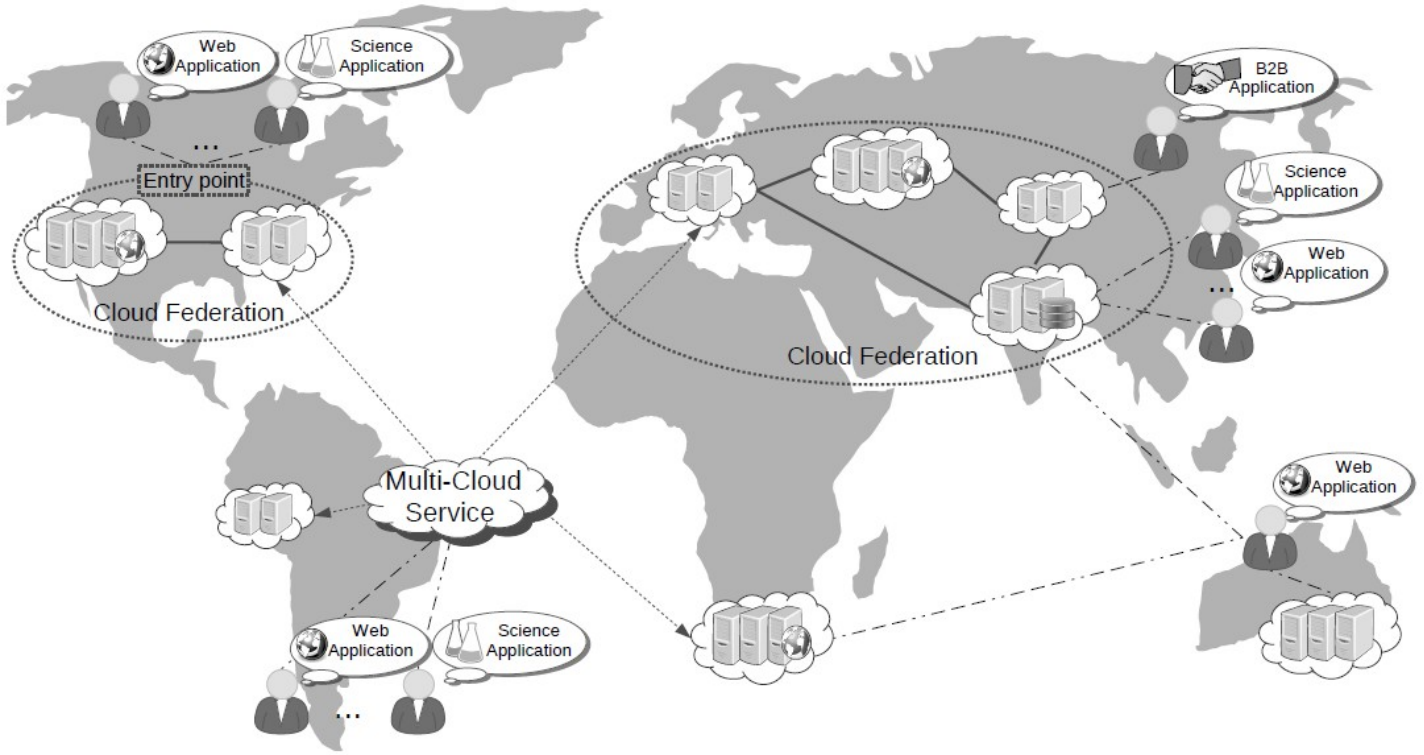
Cloud Computing

- Cloud computing ...
  - is a model for delivering virtualized computing resources over the Internet;
  - is supported by large scale data centres aggregating commodity hardware;
  - is subscription based (pay-as-you-go);
- Challenges - outages, security, etc.

www.google.com/datacenters/

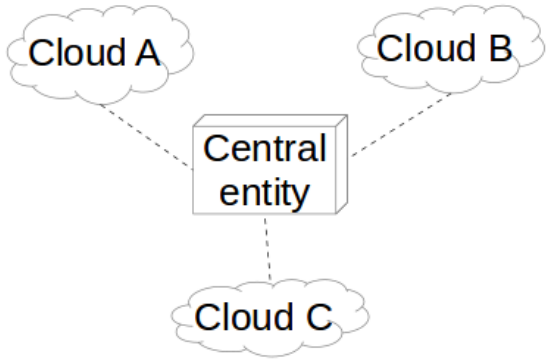
Inter-Cloud Computing

- Motivation:
  - Mitigate effects of cloud outage;
  - Diversify geographical locations;
  - Avoid vendor lock-in;
  - Latency.
- Solution - use multiple clouds

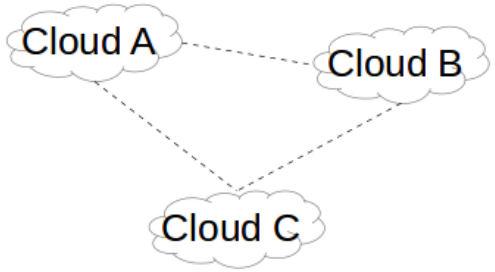


Inter-Cloud Computing: Architectures

Federations

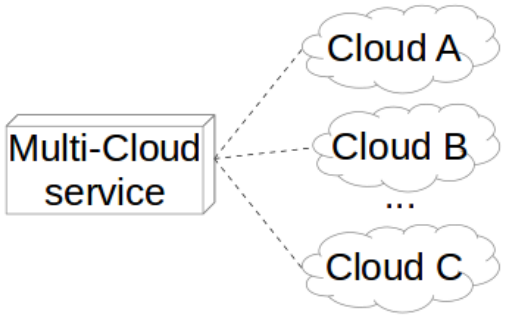


a) Centralised Inter-Cloud Federation. Clouds use a central entity to facilitate resource sharing.

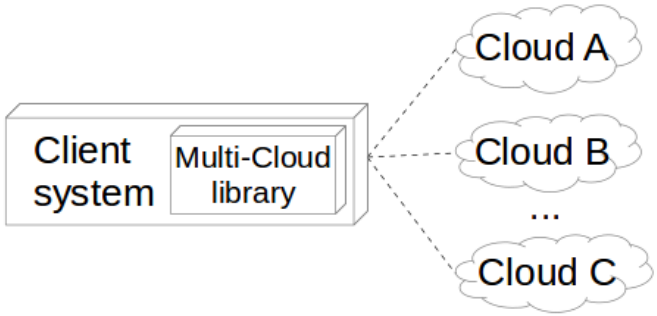


b) Peer-to-Peer Inter-Cloud Federation. Clouds collaborate directly with each other.

Multi-Clouds



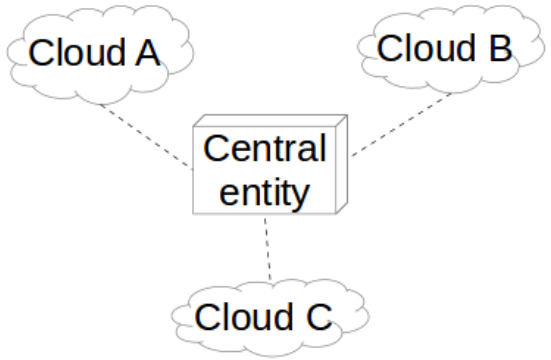
c) Multi-Cloud Service. Clients access multiple clouds through a service.



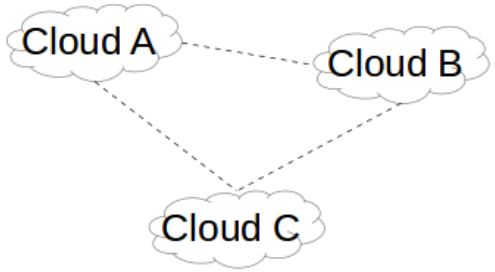
d) Multi-Cloud Library. Clients develop their own brokers by using a unified cloud API in the form of a library.

Inter-Cloud Computing: Architectures

Federations

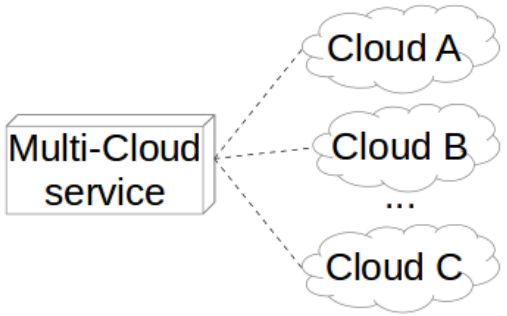


a) Centralised Inter-Cloud Federation. Clouds use a central entity to facilitate resource sharing.

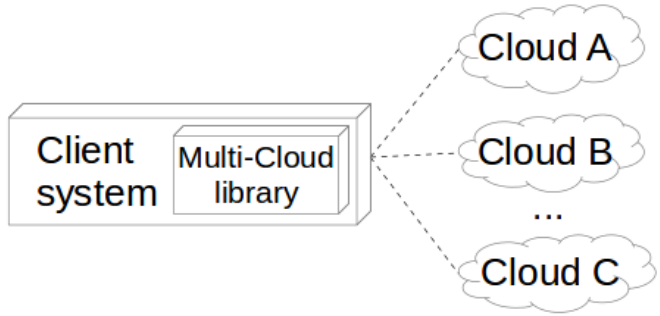


b) Peer-to-Peer Inter-Cloud Federation. Clouds collaborate directly with each other.

Multi-Clouds

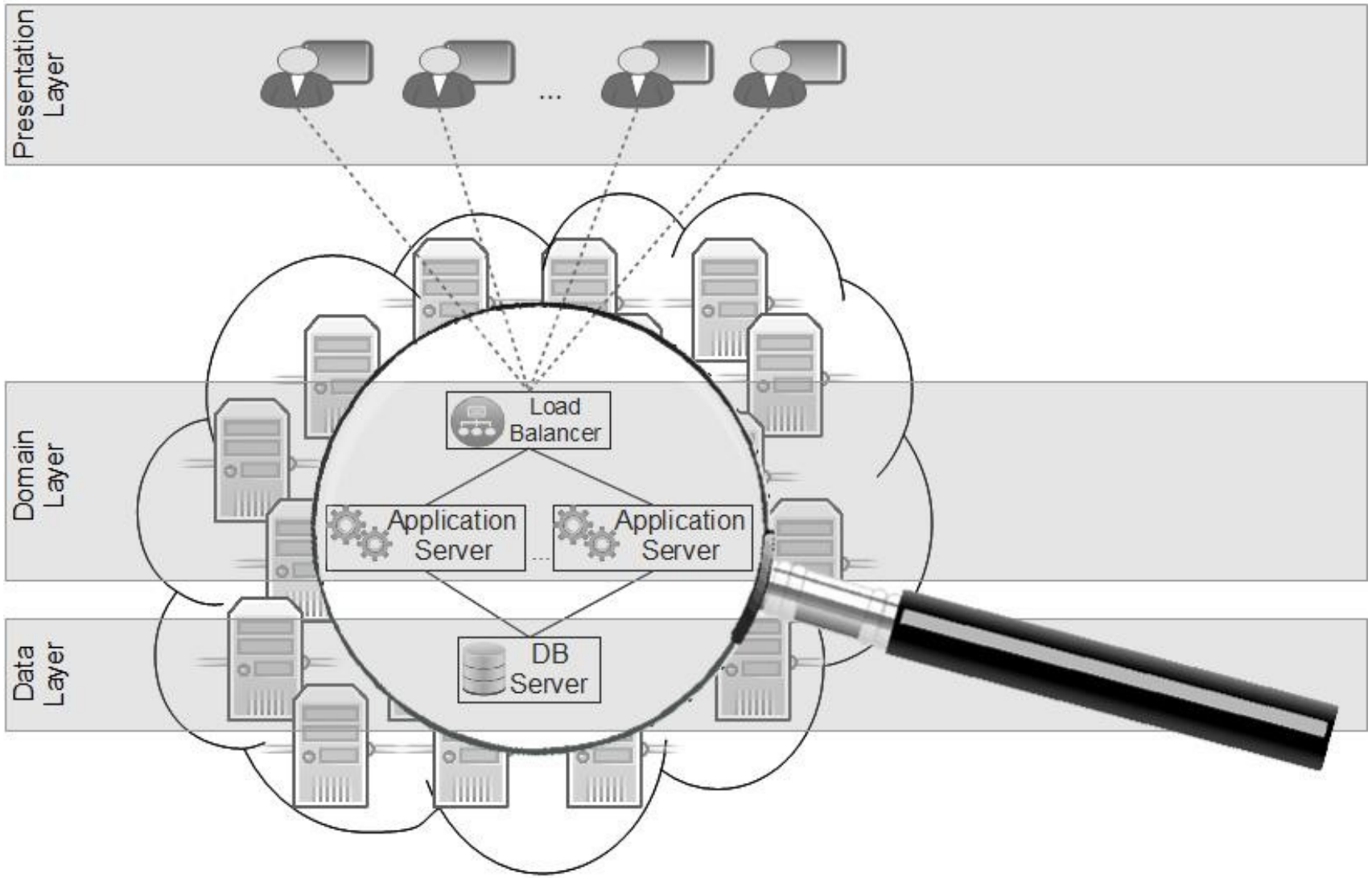


c) Multi-Cloud Service. Clients access multiple clouds through a service.

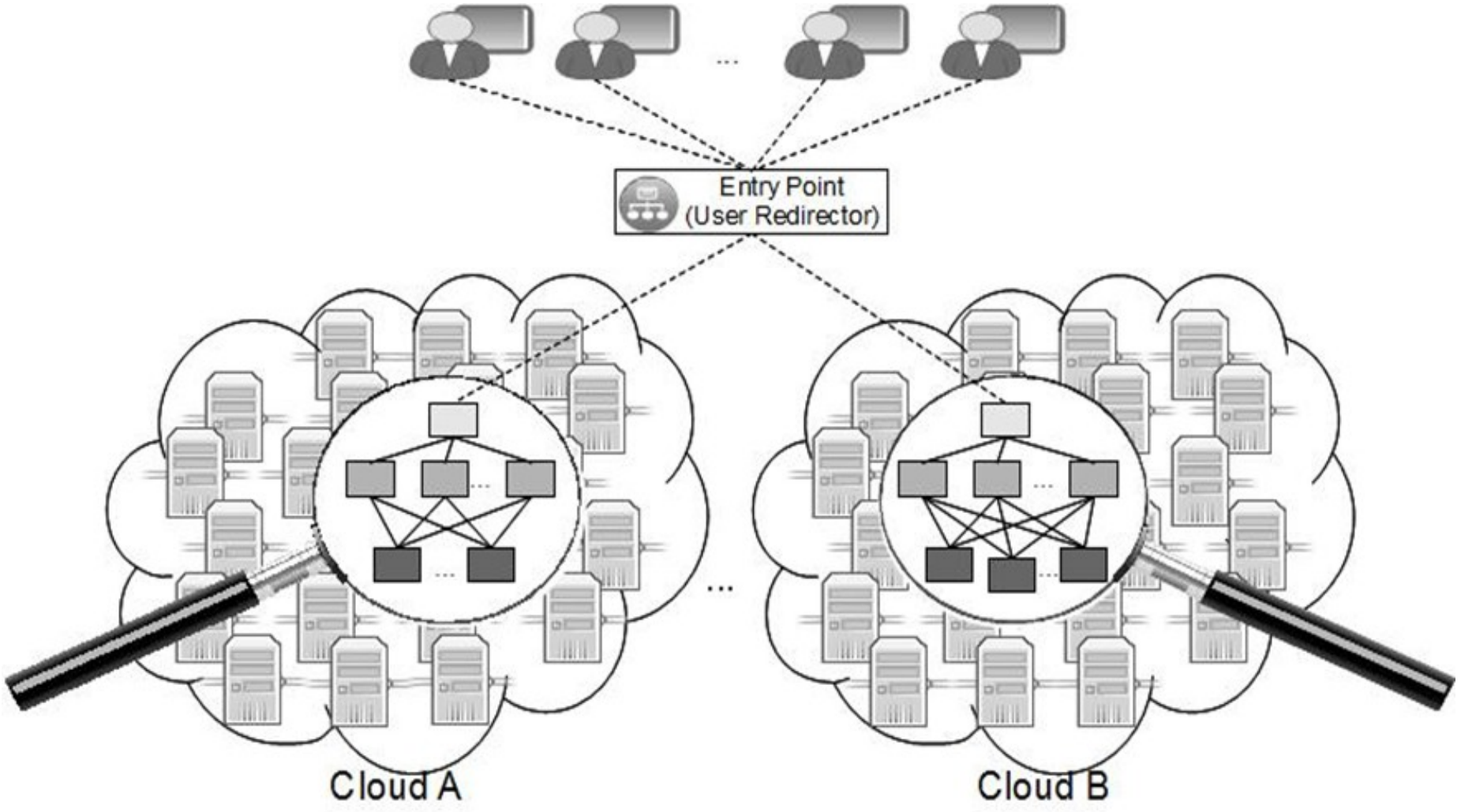


d) Multi-Cloud Library. Clients develop their own brokers by using a unified cloud API in the form of a library.

3-Tier applications in cloud



3-Tier applications in a Multi-Cloud



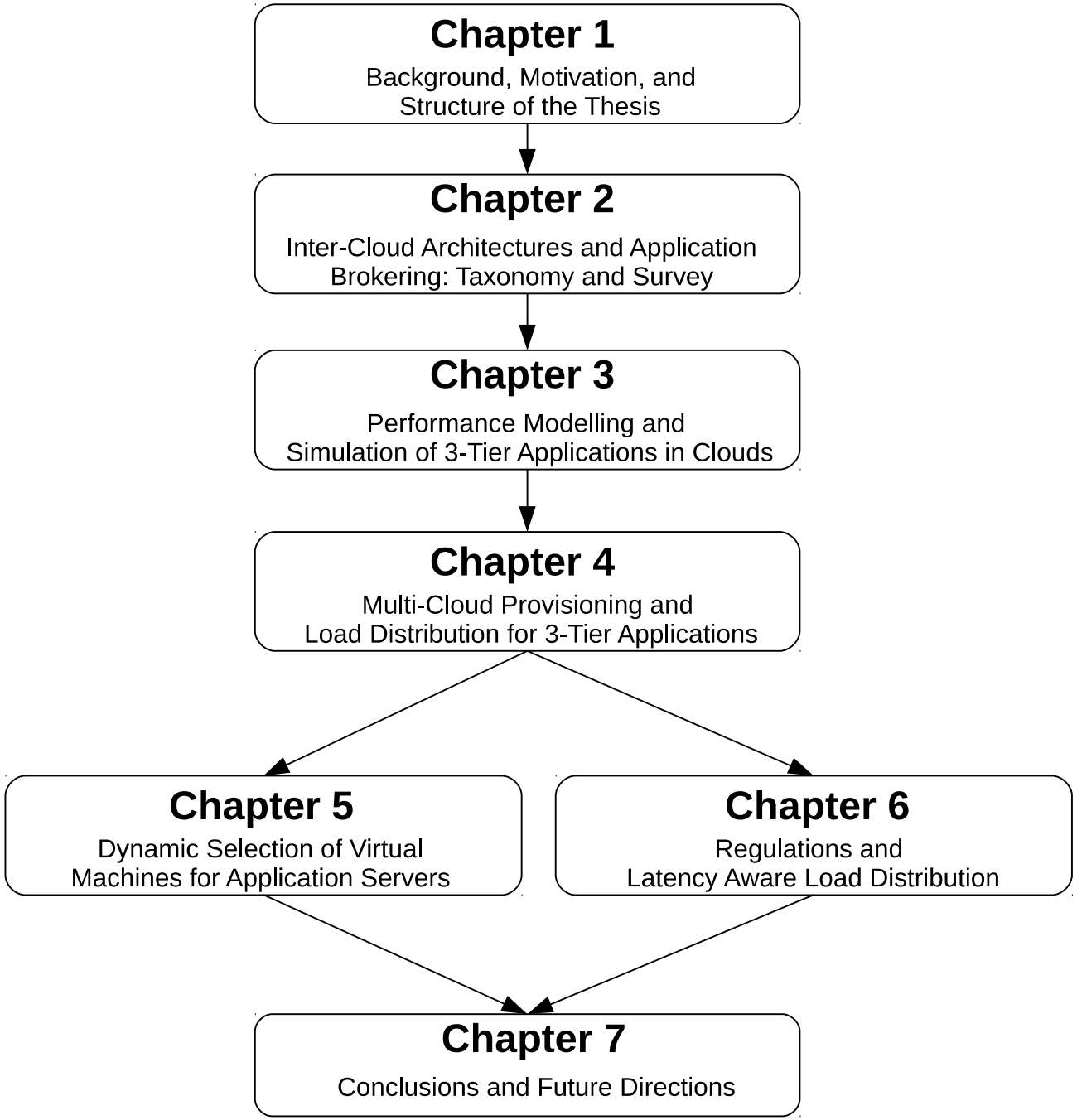
Research Question

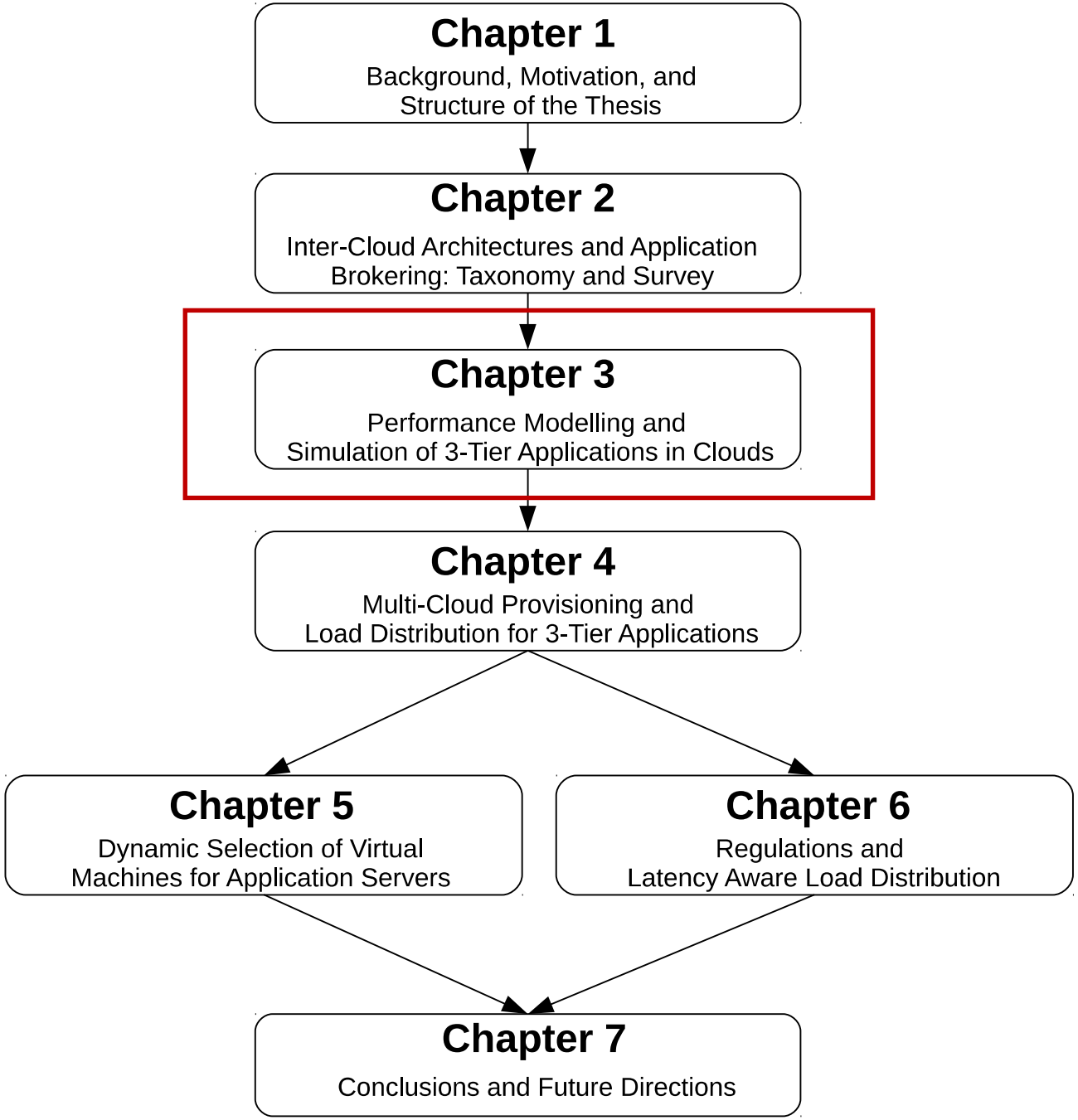
- How to broker 3-Tier applications in a Multi-Cloud environment, considering Quality of Service (QoS) requirements in terms of:
  - **Network Latency Awareness** — end users should be served near their geographical location to experience better responsiveness;
  - **Pricing Awareness**— the overall costs for hosting should be minimized;
  - **Legislation/Policy Awareness** — legal and political considerations about where individual users are served should be honoured;
  - **Code Re-usability** — few changes to existing 3-Tier applications should be made. The technical overhead of moving an existing 3-Tier system to a Multi-Cloud should be minimal.

Research Question

?

- How to broker 3-Tier applications in a Multi-Cloud environment, considering Quality of Service (QoS) requirements in terms of:
  - **Network Latency Awareness** — end users should be served near their geographical location to experience better responsiveness;
  - **Pricing Awareness**— the overall costs for hosting should be minimized;
  - **Legislation/Policy Awareness** — legal and political considerations about where individual users are served should be honoured;
  - **Code Re-usability** — few changes to existing 3-Tier applications should be made. The technical overhead of moving an existing 3-Tier system to a Multi-Cloud should be minimal.

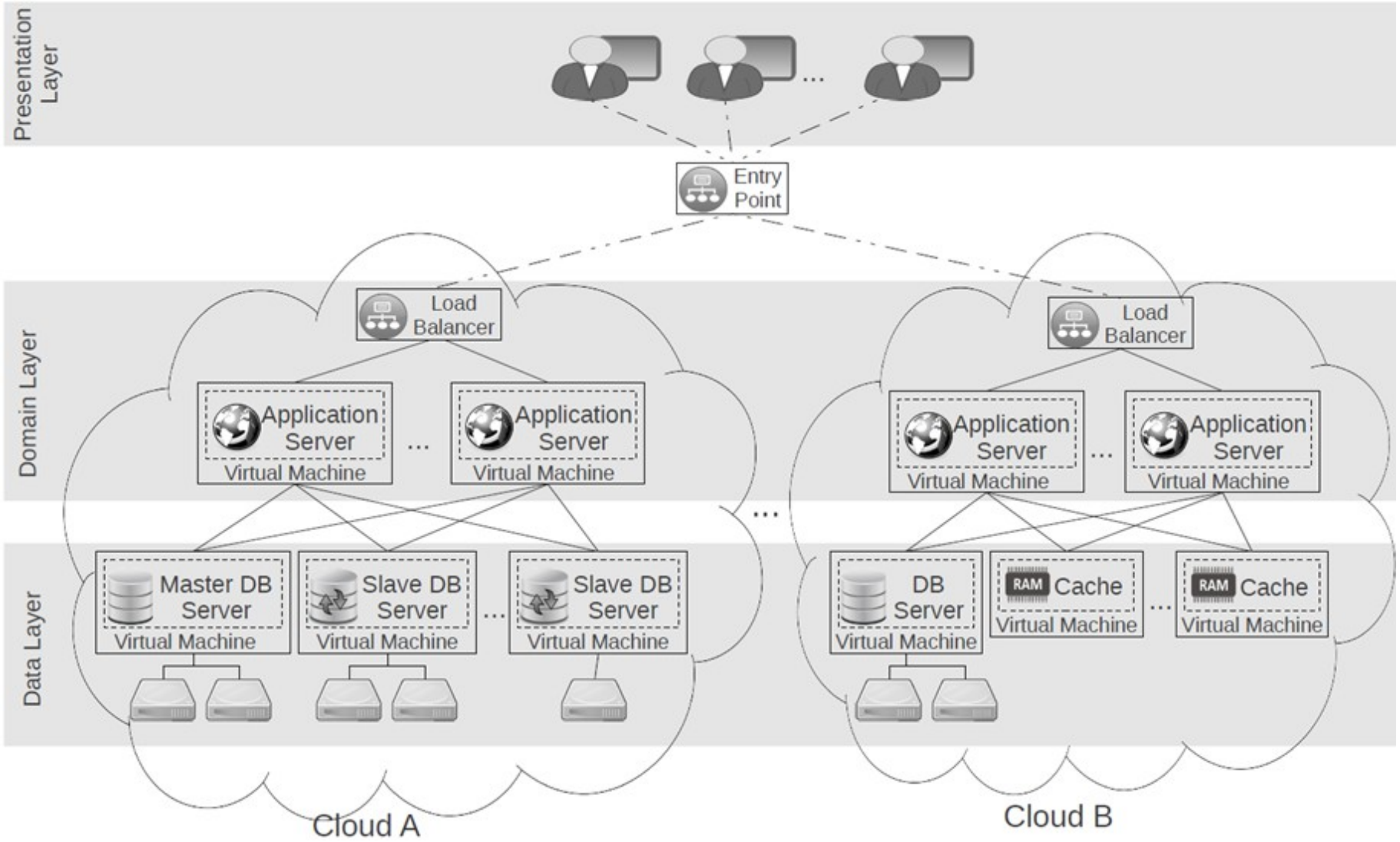

Chapter 1
Background, Motivation, and Structure of the Thesis
Chapter 2
Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey
Chapter 3
Performance Modelling and Simulation of 3-Tier Applications in Clouds
Chapter 4
Multi-Cloud Provisioning and Load Distribution for 3-Tier Applications
Chapter 5
Dynamic Selection of Virtual Machines for Application Servers
Chapter 6
Regulations and Latency Aware Load Distribution
Chapter 7
Conclusions and Future Directions


Chapter 1
Background, Motivation, and Structure of the Thesis
Chapter 2
Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey
Chapter 3
Performance Modelling and Simulation of 3-Tier Applications in Clouds
Chapter 4
Multi-Cloud Provisioning and Load Distribution for 3-Tier Applications
Chapter 5
Dynamic Selection of Virtual Machines for Application Servers
Chapter 6
Regulations and Latency Aware Load Distribution
Chapter 7
Conclusions and Future Directions

Background and Objectives

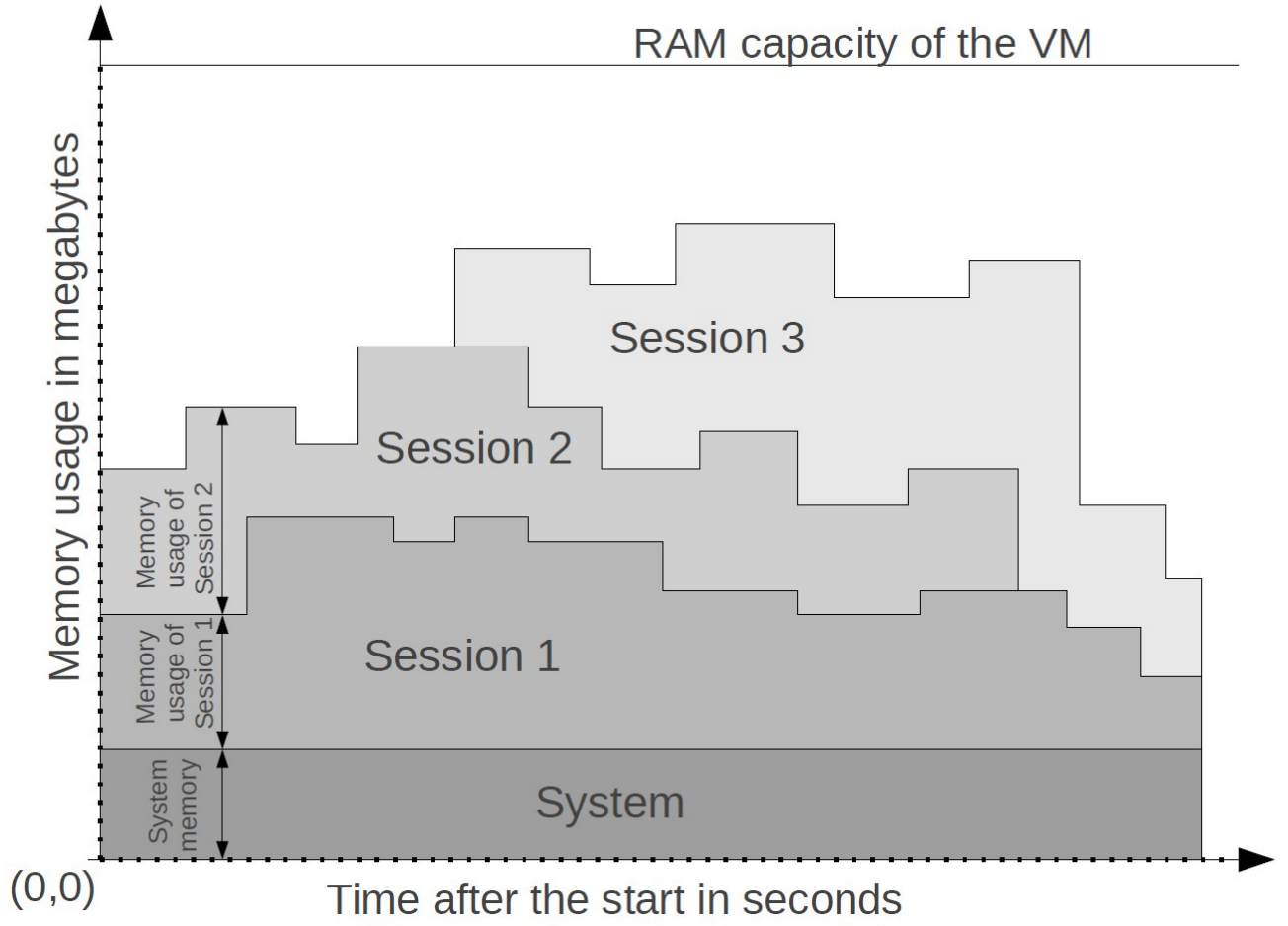
- Distributed systems simulation has already fostered the research efforts;
- Existing simulators can be used to simulate batch processing and infrastructure utilisation workloads only;
- Previous works on multi-tier application modelling have series of shortcomings;
- **Goal** – define a flexible and coarse grained model and simulator for 3-Tier applications in one and multiple clouds.

Target Scenario

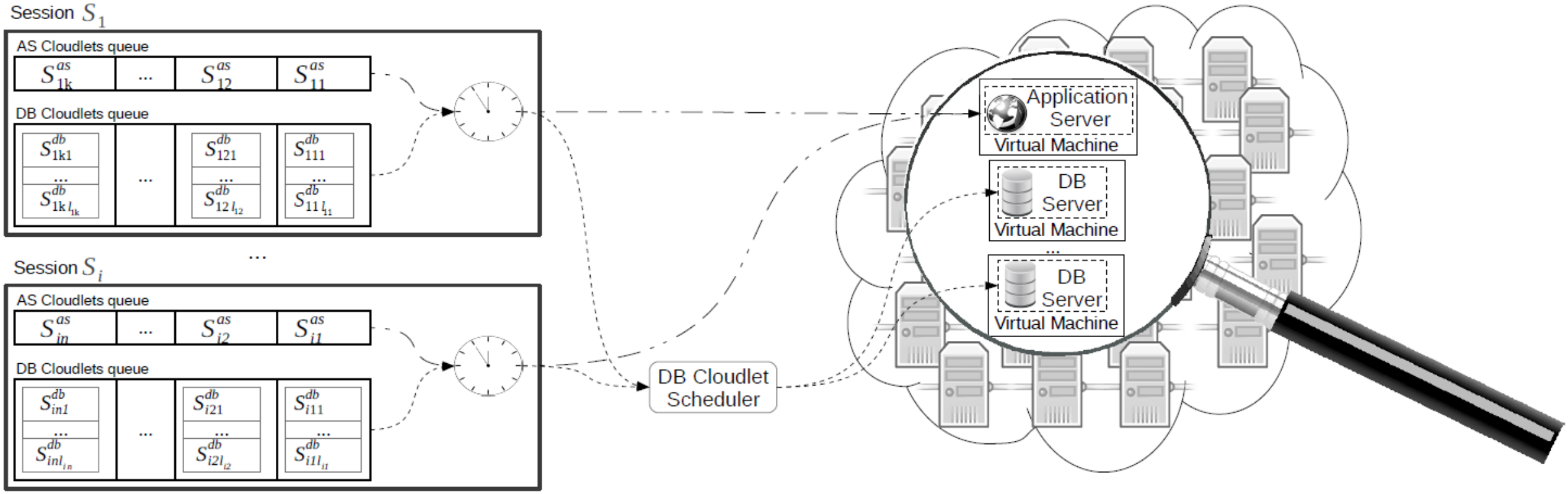


Session Performance Model

- *AS Memory Load -* $\phi_{as}(t)$
- *AS CPU Load -* $\nu_{as}(t)$
- DB Memory Load - $\phi_{db}(t, d_i)$
- DB CPU Load - $\nu_{db}(t, d_i)$
- DB Disk I/O Load - $\sigma_{db}(t, d_i)$
- Step Size – δ
- *Session arrival model:*
  - *Model each session type separately*
  - *Poison distribution of a frequency function -* $Po(\lambda(t))$



Simulator Implementation

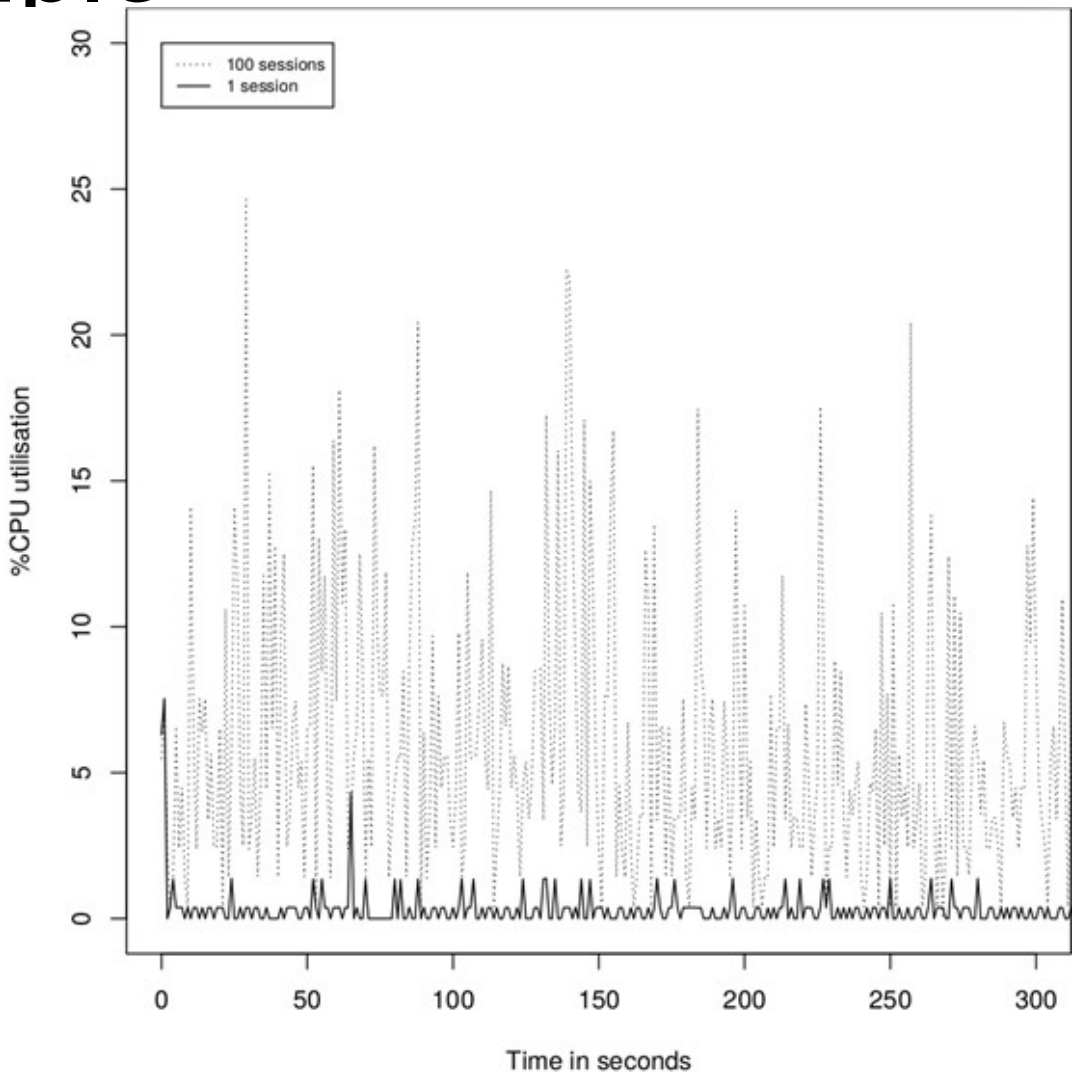


Validation Environment

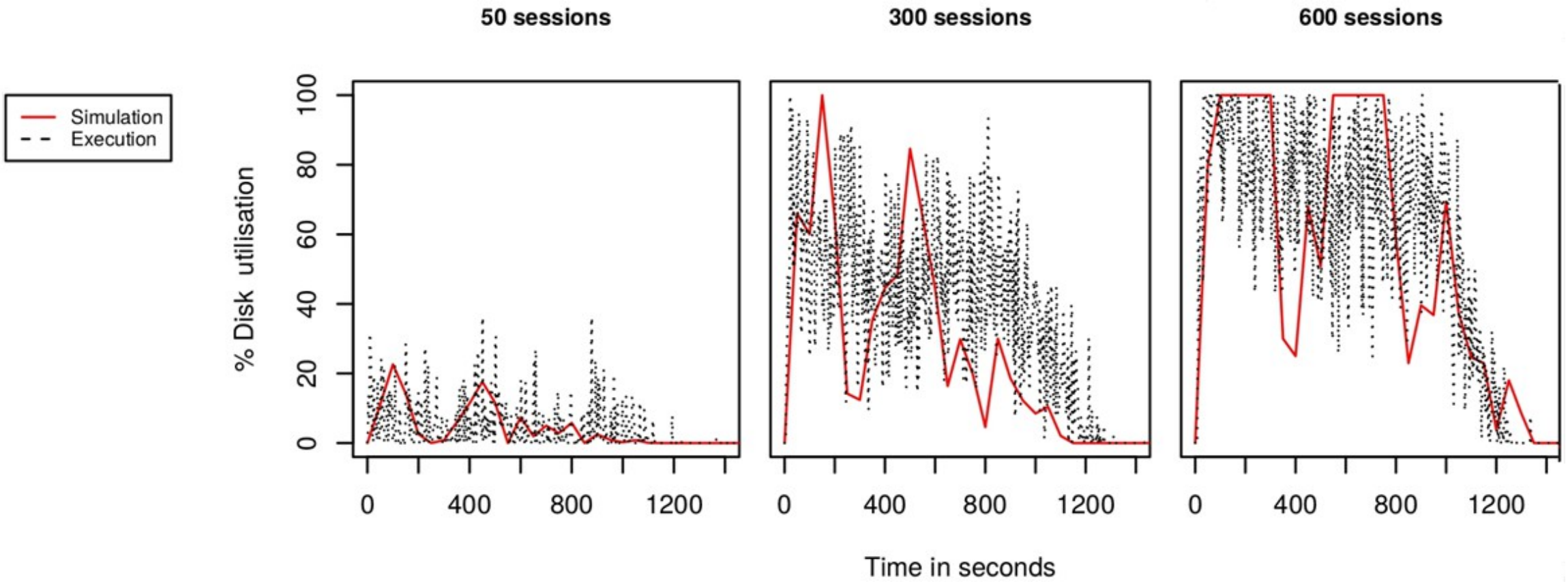
- *3-Tier app. designed after ebay;*
- *Client application, generating requests;*
- *Transition table;*
- *"Think times";*
- *Experiments;*
  - *Benchmarking;*
  - *Experiment 1 - static workload on local infrastructure;*
  - *Experiment 2 - dynamic workload on local infrastructure (DC1) and EC2(DC2);*

Model Extraction - Example

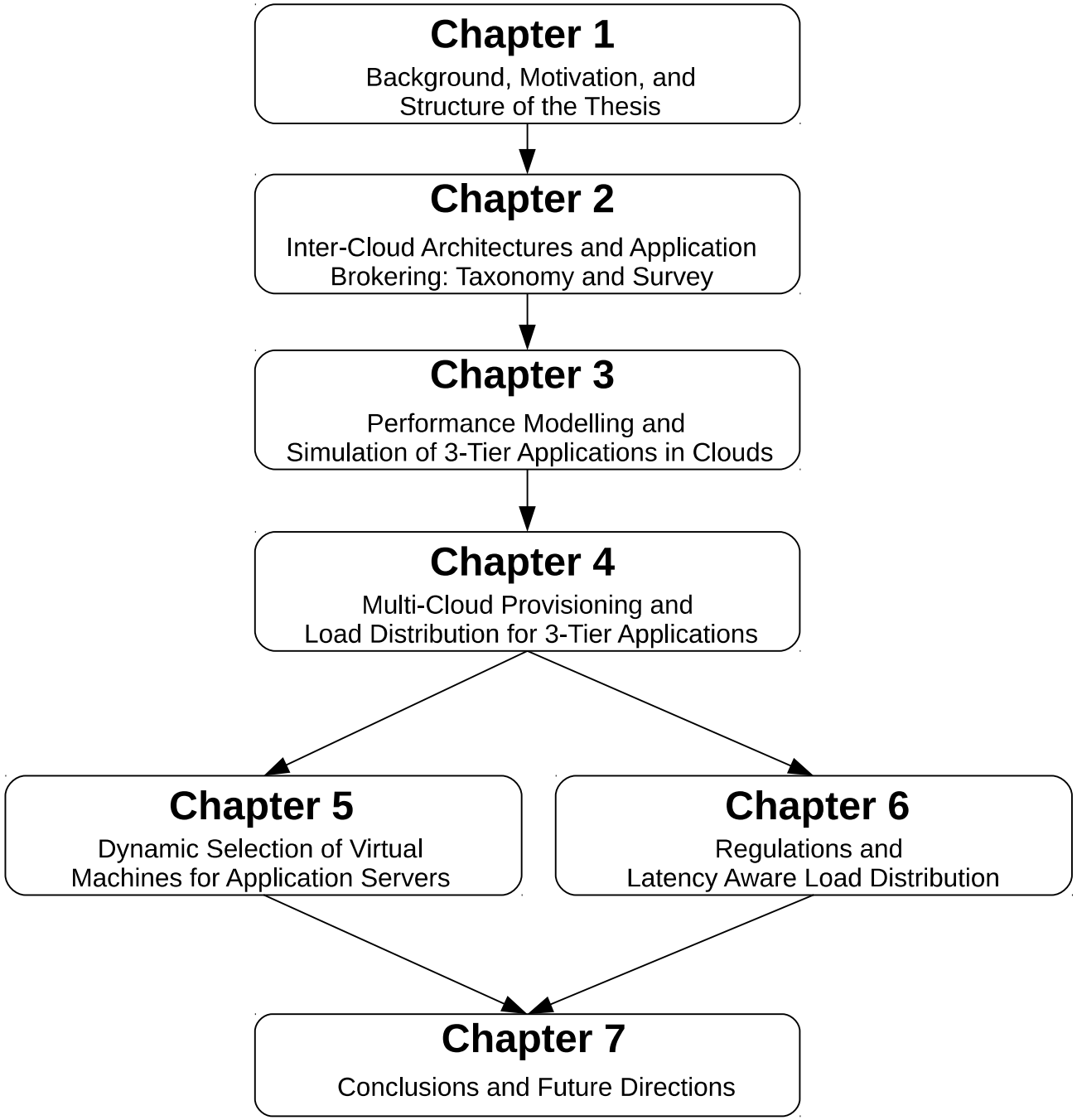
- Execute 2 Experiments:
  - With 1 user;
  - With 100 users;
- Compute the “average” session behavior;
- Standard Linux utilisation measurement tools.

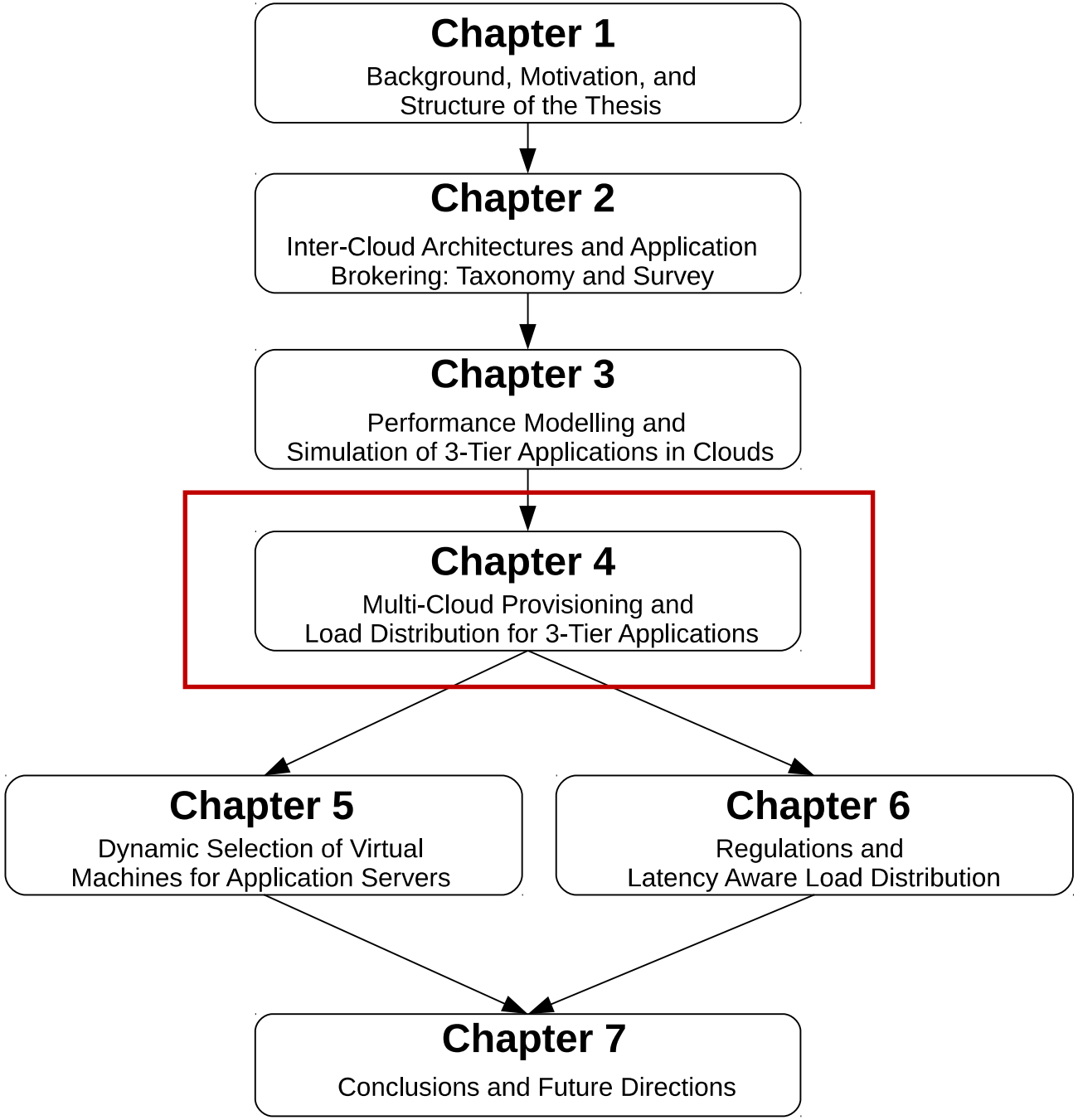


Experiment 1: Static Workload in 1 cloud



Predicted and actual disk I/O utilisation of the DB server with 50, 300, and 600 simultaneous sessions in Experiment 1.

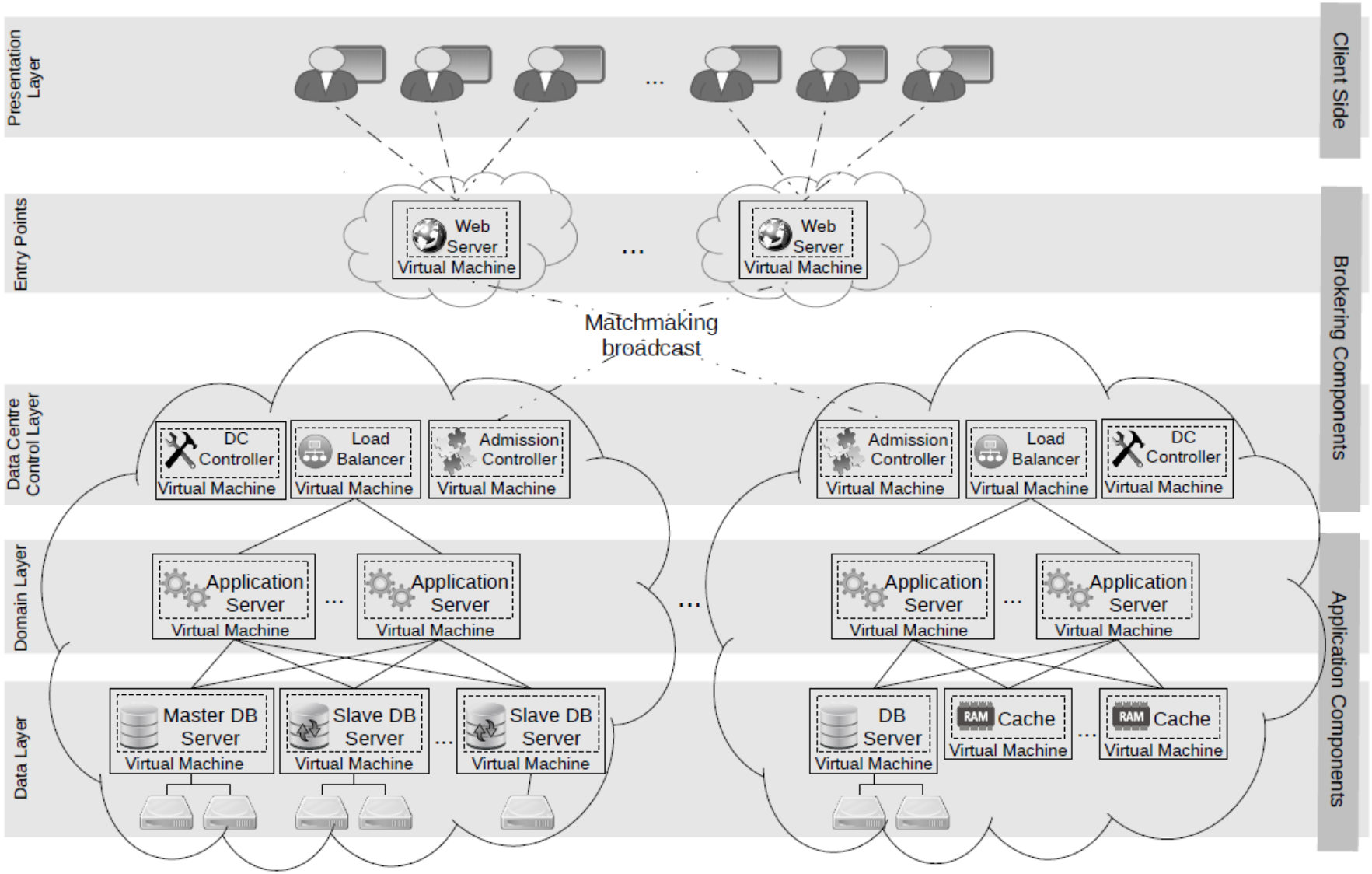

Chapter 1
Background, Motivation, and Structure of the Thesis
Chapter 2
Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey
Chapter 3
Performance Modelling and Simulation of 3-Tier Applications in Clouds
Chapter 4
Multi-Cloud Provisioning and Load Distribution for 3-Tier Applications
Chapter 5
Dynamic Selection of Virtual Machines for Application Servers
Chapter 6
Regulations and Latency Aware Load Distribution
Chapter 7
Conclusions and Future Directions


Chapter 1
Background, Motivation, and Structure of the Thesis
Chapter 2
Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey
Chapter 3
Performance Modelling and Simulation of 3-Tier Applications in Clouds
Chapter 4
Multi-Cloud Provisioning and Load Distribution for 3-Tier Applications
Chapter 5
Dynamic Selection of Virtual Machines for Application Servers
Chapter 6
Regulations and Latency Aware Load Distribution
Chapter 7
Conclusions and Future Directions

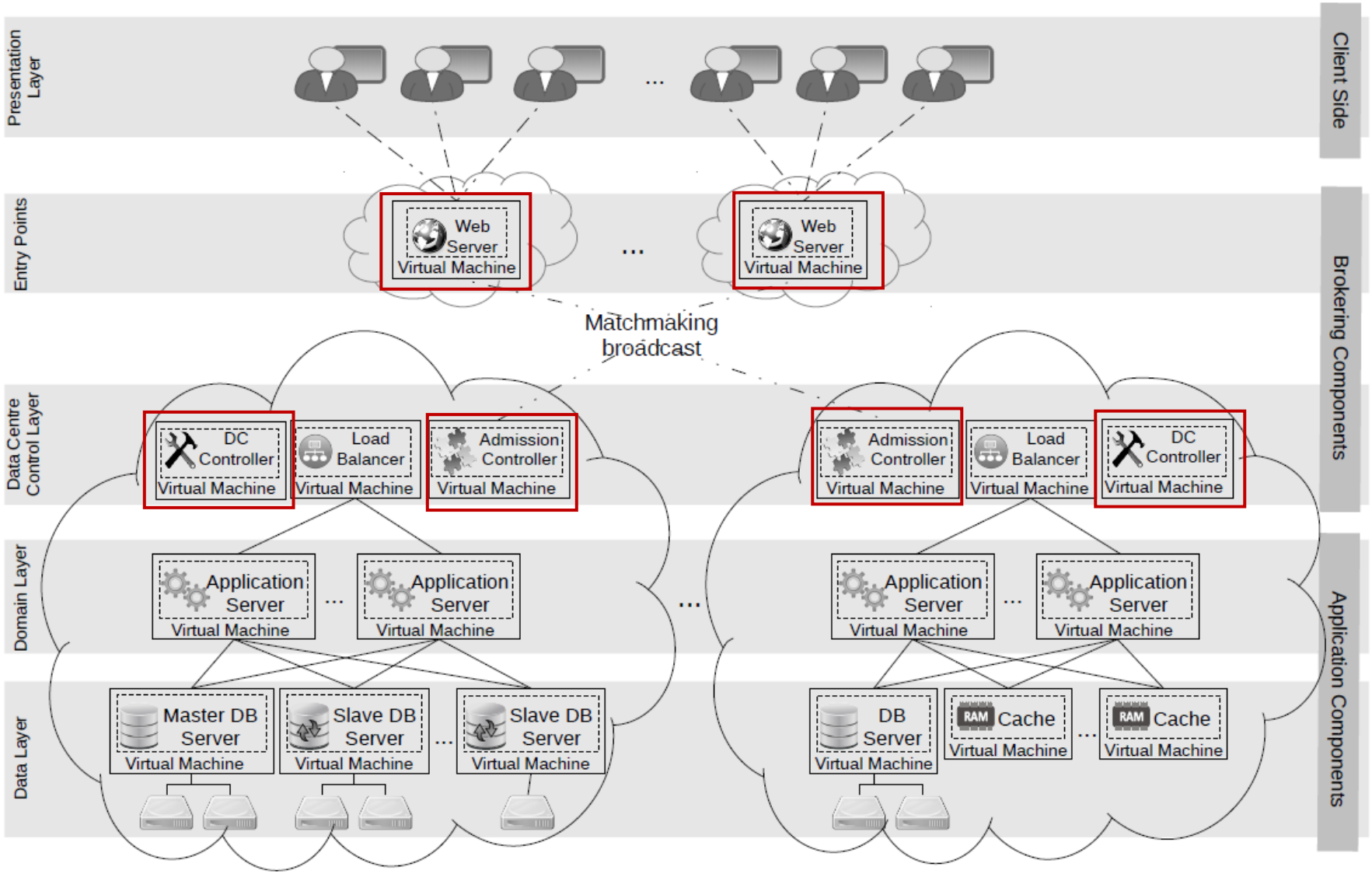
Background and Objectives

- Current Multi-Cloud 3-Tier have limitations manage resources and workload suboptimally;
- They do not consider essential regulatory requirements;
- **Goal**: propose a general and flexible architecture that honours key non-functional requirements and optimises cost and latency.

Overall Architecture



Overall Architecture



Load Balancing and Autoscaling

- Load balancing algorithm – sticky or not?
  - Monitor VM utilization;
  - Free underutilized VMs.
- Autoscaling algorithm:
  - Repeated periodically;
  - Number of pre-provisioned instances;
  - Do not terminate before billing time is over;

Load Balancing and Autoscaling

- Load balancing algorithm – sticky or not?
  - Monitor VM utilization;
  - Free underutilized VMs.
- Autoscaling algorithm:
  - Repeated periodically;
  - Number of pre-provisioned instances;
  - Do not terminate before billing time is over;

ALGORITHM 3: Load Balancing Algorithm.

input : s_i, th_{cpu}, th_{ram}, VM_{as}

```
1  sortDescendinglyByCPUUtilisation(VM_as);
2  hostVM ⟵ last element of VM_as;
3  for vm_i ∈ VM_as do
4      vm_cpu ⟵ CPU utilisation of vm_i;
5      vm_ram ⟵ RAM utilisation of vm_i;
6      if vm_cpu < th_cpu and vm_ram < th_ram and !networkBuffersOverloaded() then
7          hostVM ⟵ vm_i;
8          break;
9      end
10 end
11 assignSessionTo(s, hostVM)
```

Load Balancing and Autoscaling

- Load balancing algorithm – sticky or not?
  - Monitor VM utilization;
  - Free underutilized VMs.
- Autoscaling algorithm:
  - Repeated periodically;
  - Number of pre-provisioned instances;
  - Do not terminate before billing time is over;

Load Balancing and Autosc

- Load balancing algorithm – sticky or not?
  - Monitor VM utilization;
  - Free underutilized VMs.
- Autoscaling algorithm:
  - Repeated periodically;
  - Number of pre-provisioned instances;
  - Do not terminate before billing time is over;

ALGORITHM 4: Scale Up/Down Algorithm.

```
input : t_cur, tgr_cpu, tgr_ram, VM_as, n, Δ
1  nOverloaded ⟵ 0;
2  listFreeVms ⟵ empty list;
3  for vm ∈ VM_as ;                                   // Inspect the status of all AS VMs
4  do
5      vm_cpu ⟵ CPU utilisation of vm;
6      vm_ram ⟵ RAM utilisation of vm;
7      if vm_cpu ≥ tgr_cpu or vm_ram ≥ tgr_ram or networkBuffersOverloaded() then
8          nOverloaded ⟵ nOverloaded + 1;
9      else if vm_i serves no sessions then
10         listFreeVms.add(vm);
11     end
12 end
13 nFree ⟵ length of listFreeVms;
14 nAS ⟵ length of VM_as;
15 allOverloaded ⟵ nOverloaded + nFree = nAS and nOverloaded > 0;
16 if nFree ≤ n ;                                     // Provision more VMs
17 then
18     nVmsToStart ⟵ 0;
19     if allOverloaded then
20         nVmsToStart ⟵ n − nFree + 1;
21     else
22         nVmsToStart ⟵ n − nFree
23     end
24     launch nVmsToStart AS VMs
25 else
26     nVmsToStop ⟵ 0 ;                                // Release VMs
27     if allOverloaded then
28         nVmsToStop ⟵ nFree − n;
29     else
30         nVmsToStop ⟵ nFree − n + 1
31     end
32     sortAscendinglyByBillingTime(listFreeVms);
33     for i = 1 to nVmsToStop do
34         billTime ⟵ billing time of listFreeVms[i];
35         if billTime − t_cur < Δ then
36             terminate listFreeVms[i];
37         else
38             break
39         end
40     end
41 end
```

Load Balancing and Autoscaling

- Load balancing algorithm – sticky or not?
  - Monitor VM utilization;
  - Free underutilized VMs.
- Autoscaling algorithm:
  - Repeated periodically;
  - Number of pre-provisioned instances;
  - Do not terminate before billing time is over;

Cloud Selection Algorithm

- Ensure users are served in eligible clouds;
- Timeout;
- Estimate network latency;
- Estimate potential cost;
- Overloaded infrastructure;
- Optimise latency and cost.

ALGORITHM 5: Cloud Site Selection Algorithm.

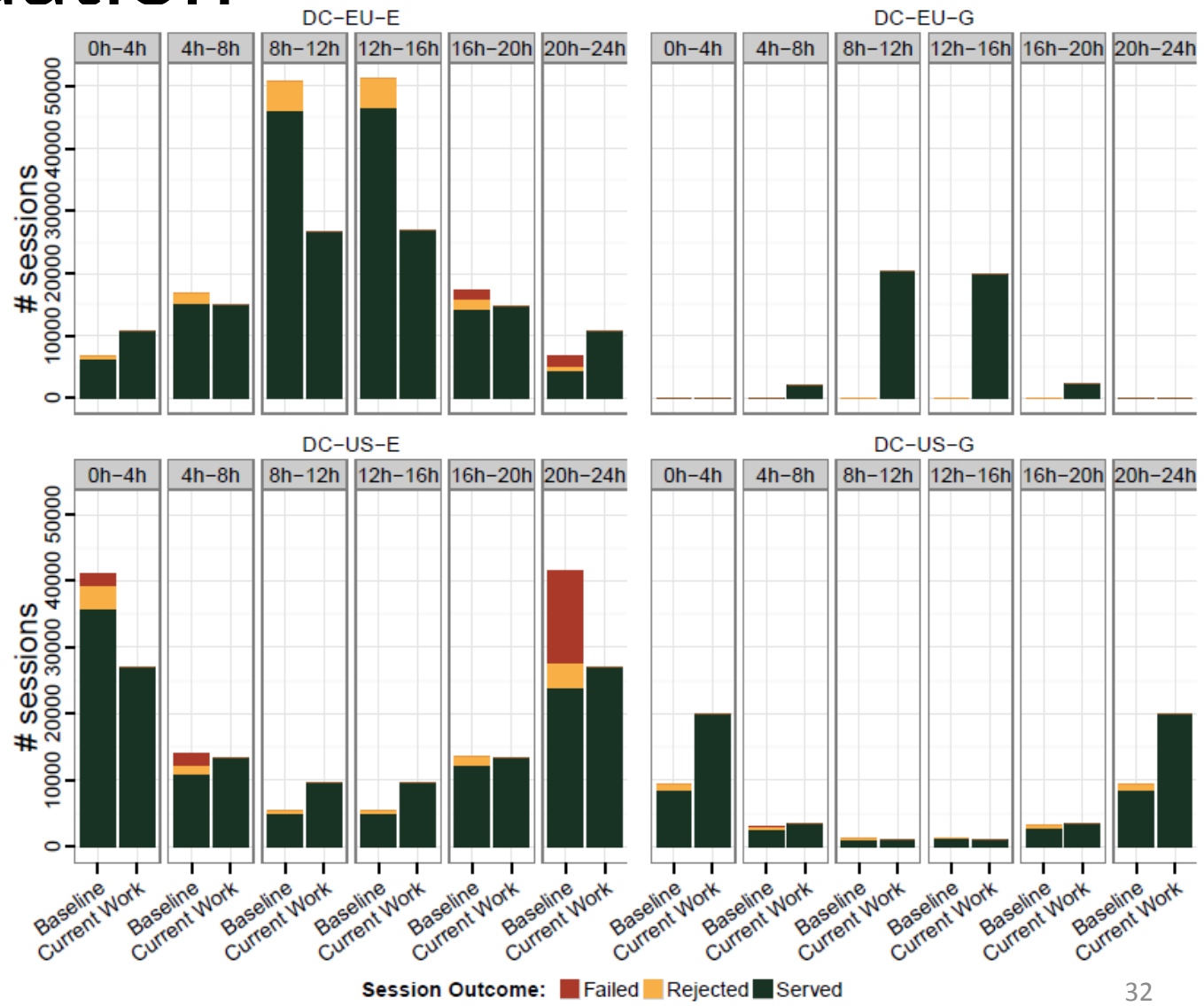
input : $users, timeout, clouds, latency_{SLA}$

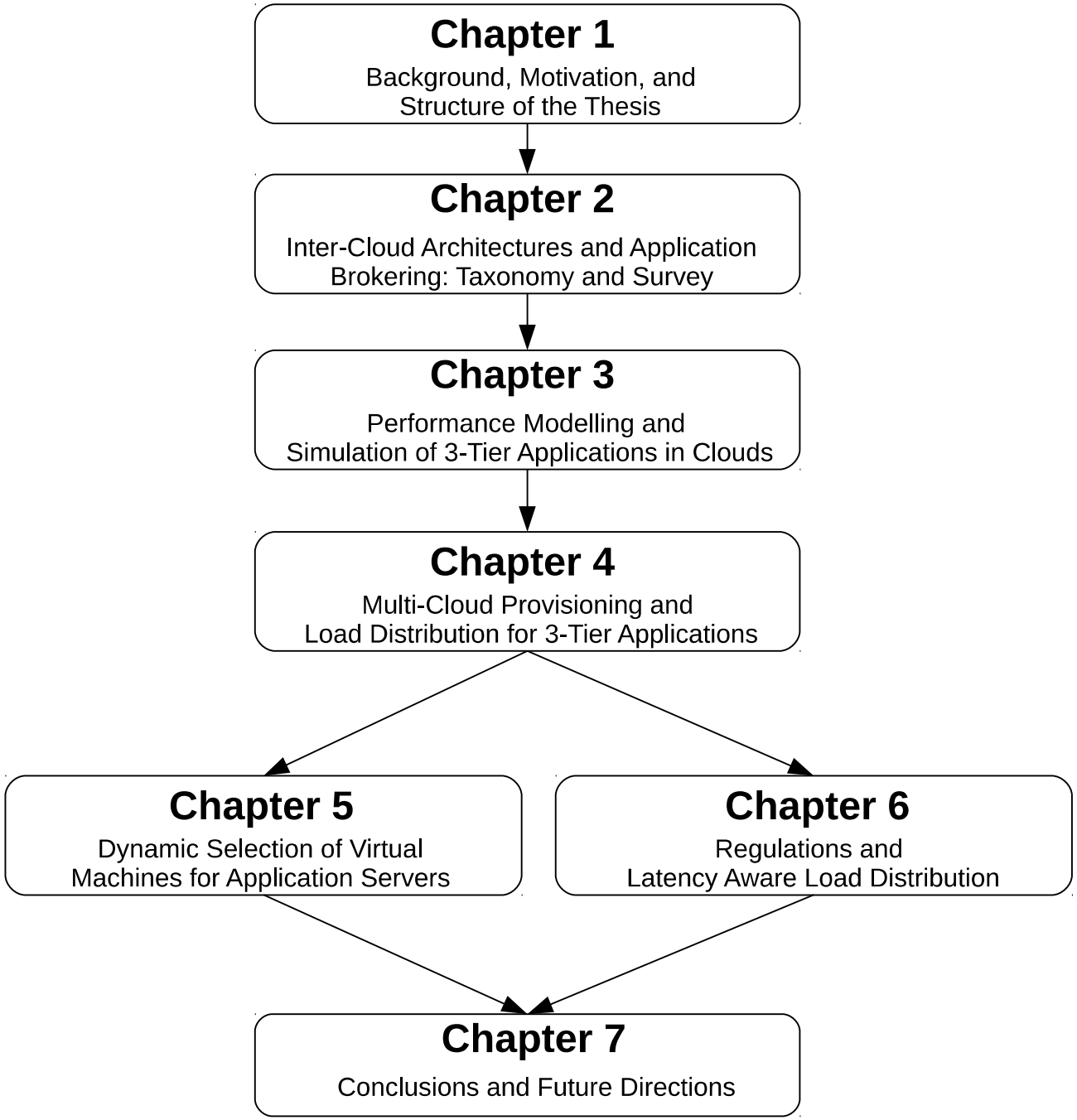
// Broadcast users' data to admission controllers

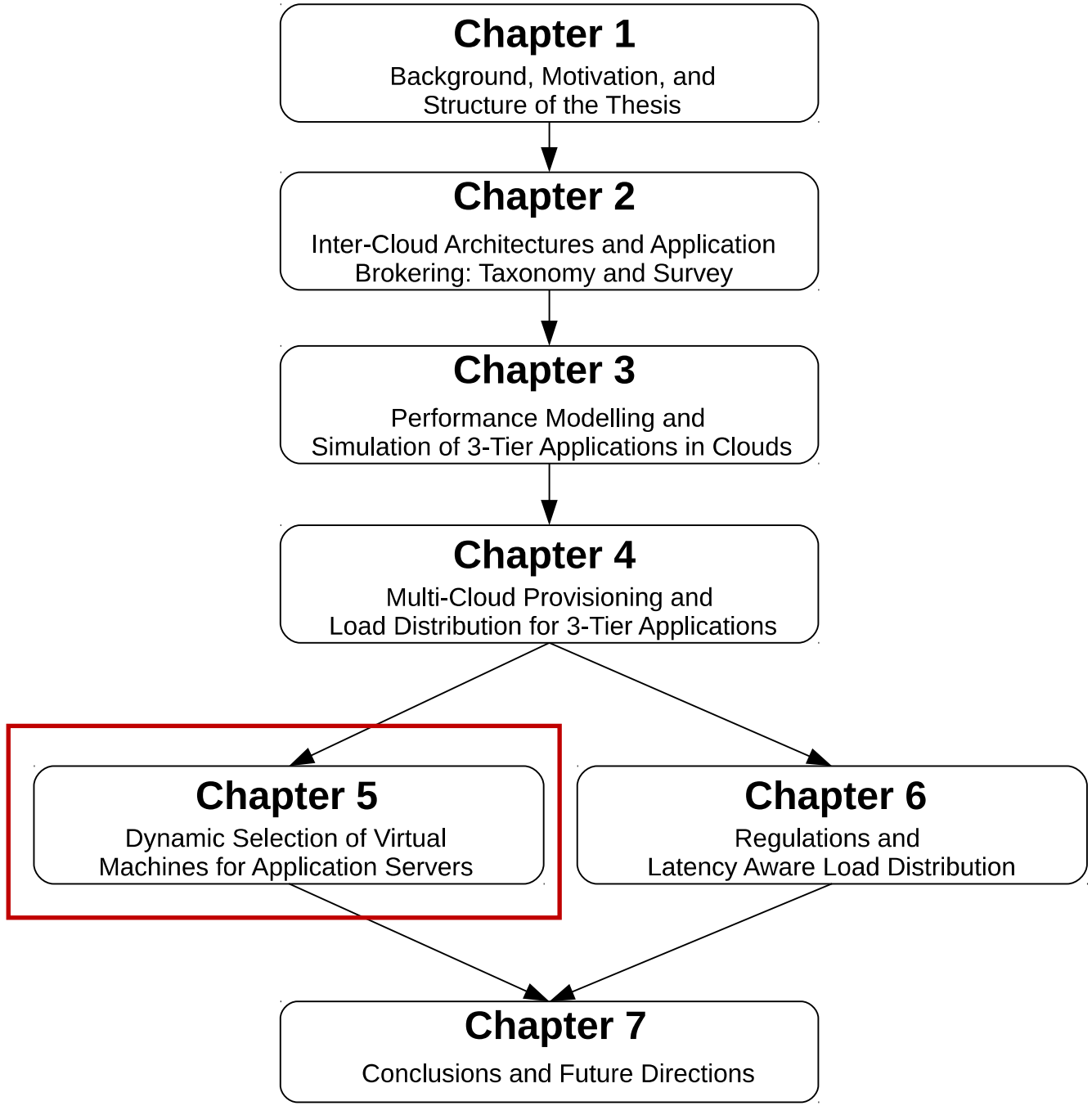
1 **for** $c_i \in clouds$ **do**
2 | $ac_i \longleftarrow$ IP address of c_i's *admission controller*;
3 | send to ac_i users' identifier;
4 **end**
5 wait *timeout* seconds or until all clouds respond;
6 **for** $u_i \in users$ **do**
7 | $clouds_{accept} \longleftarrow$ clouds eligible to serve u_i;
8 | sortAscendinglyByPrice($clouds_{accept}$);
9 | $selectedCloud \longleftarrow$ null;
10 | $selectedLatency \longleftarrow +\infty$;
11 | **for** $c_i \in clouds_{accept}$ **do**
12 | | $latency \longleftarrow$ latency between u and c_i;
13 | | **if** $latency < latency_{SLA}$ **then**
14 | | | $selectedCloud \longleftarrow c_i$;
15 | | | break;
16 | | **else if** $selectedLatency > latency$ **then**
17 | | | $selectedCloud \longleftarrow c_i$;
18 | | | $selectedLatency \longleftarrow latency$;
19 | | **end**
20 | **end**
21 | **if** $selectedCloud = null$ **then**
22 | | Deny Service;
23 | **else**
24 | | $lb \longleftarrow$ IP of *load balancer* in $selectedCloud$;
25 | | redirect u to lb;
26 | **end**
27 **end**

Performance Evaluation

- Previous simulation env.;
- Clouds of AWS and Google in the US and Europe;
- Baseline:
  - AWS Route 53;
  - AWS Elastic LB;
  - AWS Autoscaling;

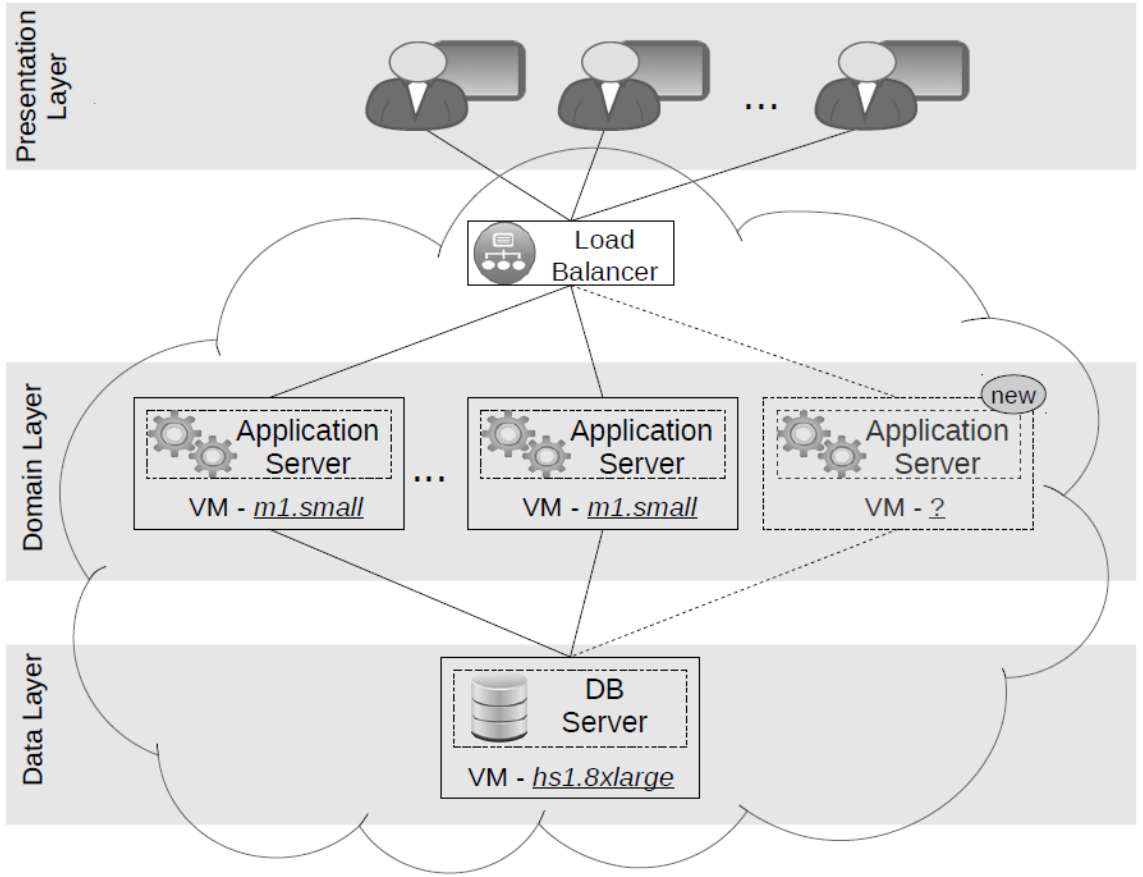



Chapter 1
Background, Motivation, and Structure of the Thesis
Chapter 2
Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey
Chapter 3
Performance Modelling and Simulation of 3-Tier Applications in Clouds
Chapter 4
Multi-Cloud Provisioning and Load Distribution for 3-Tier Applications
Chapter 5
Dynamic Selection of Virtual Machines for Application Servers
Chapter 6
Regulations and Latency Aware Load Distribution
Chapter 7
Conclusions and Future Directions


Chapter 1
Background, Motivation, and Structure of the Thesis
Chapter 2
Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey
Chapter 3
Performance Modelling and Simulation of 3-Tier Applications in Clouds
Chapter 4
Multi-Cloud Provisioning and Load Distribution for 3-Tier Applications
Chapter 5
Dynamic Selection of Virtual Machines for Application Servers
Chapter 6
Regulations and Latency Aware Load Distribution
Chapter 7
Conclusions and Future Directions

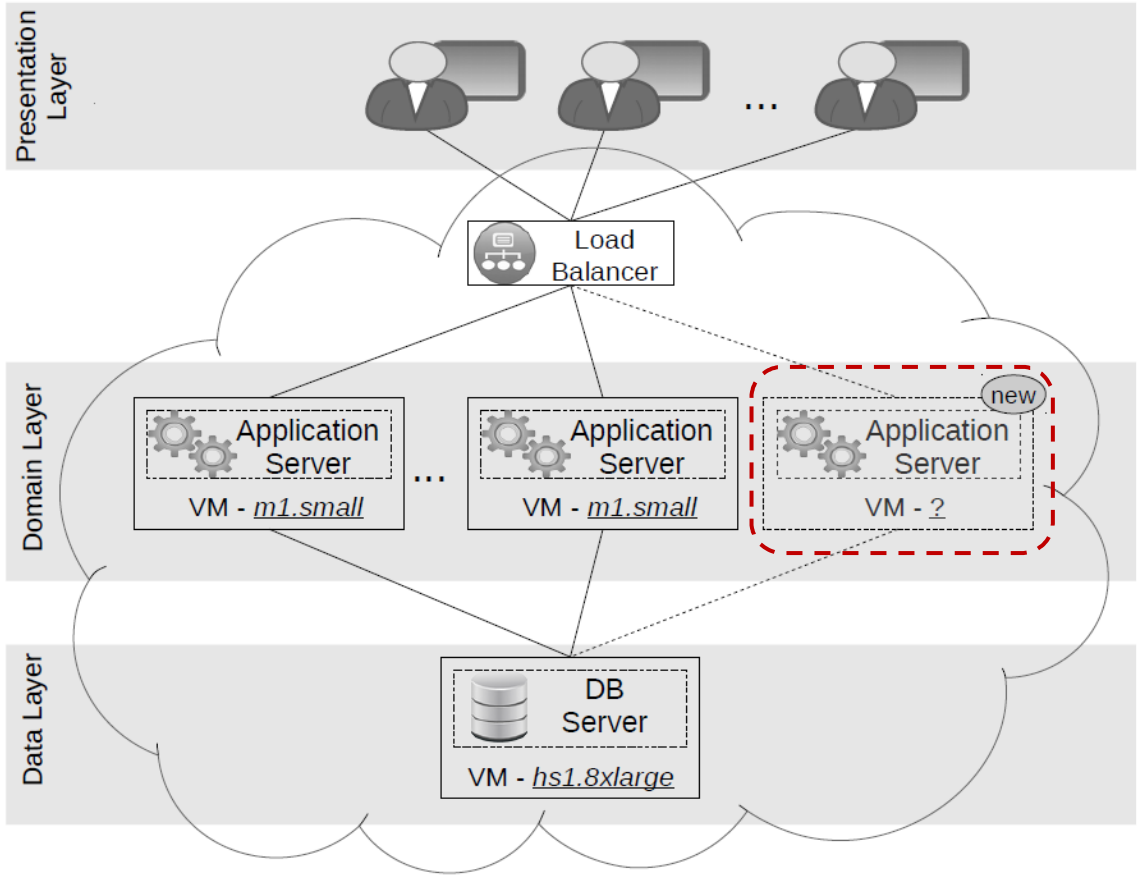
Background and Objectives

- Current autoscaling approaches select VMs statically:
  - Applications change over time;
  - Workload changes over time;
  - Infrastructure capacity changes over time.
- **Goal**: propose a flexible approach to VM selection that adapts to such changes.

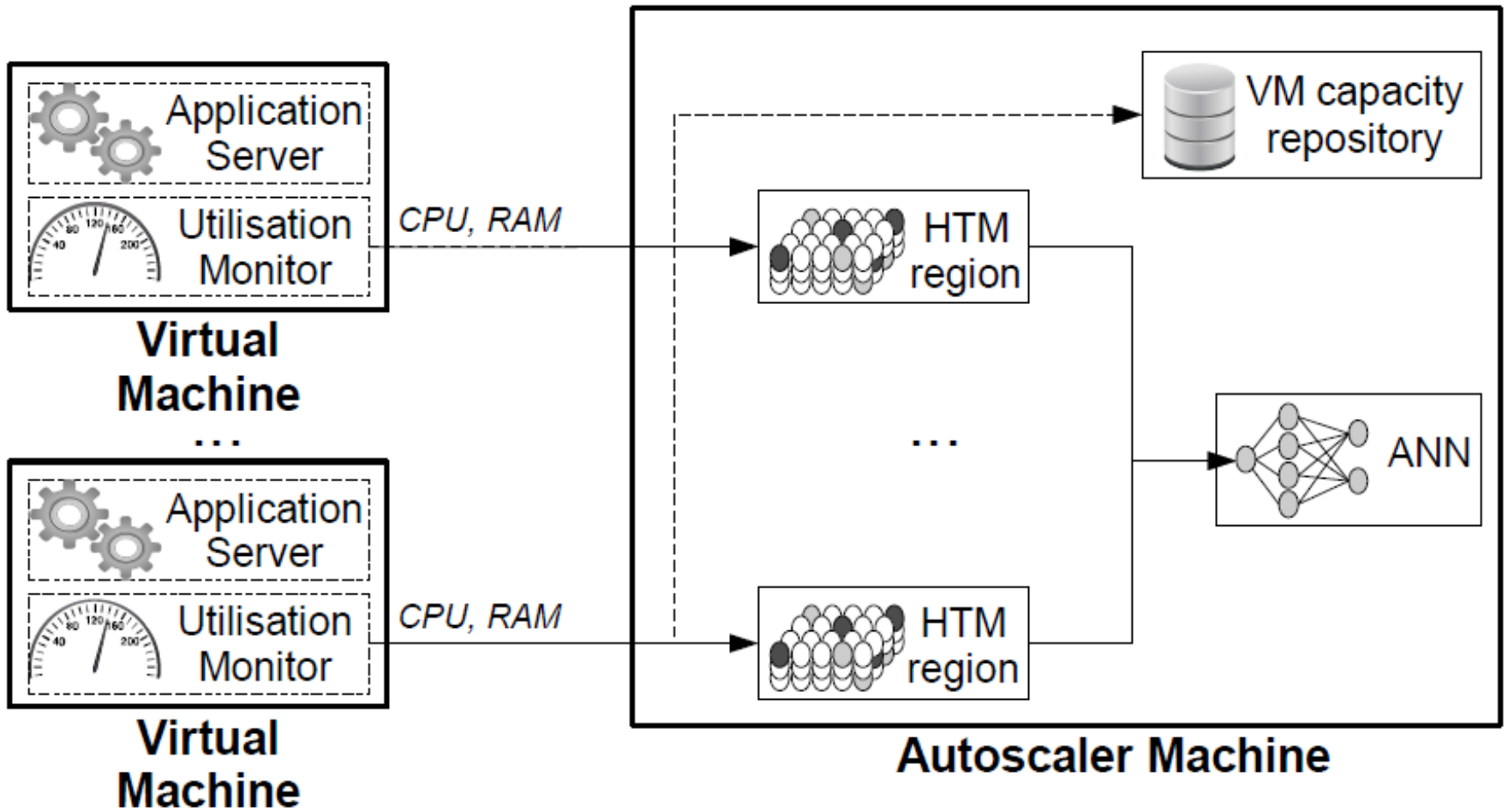


Background and Objectives

- Current autoscaling approaches select VMs statically:
  - Applications change over time;
  - Workload changes over time;
  - Infrastructure capacity changes over time.
- **Goal**: propose a flexible approach to VM selection that adapts to such changes.



Approach Overview



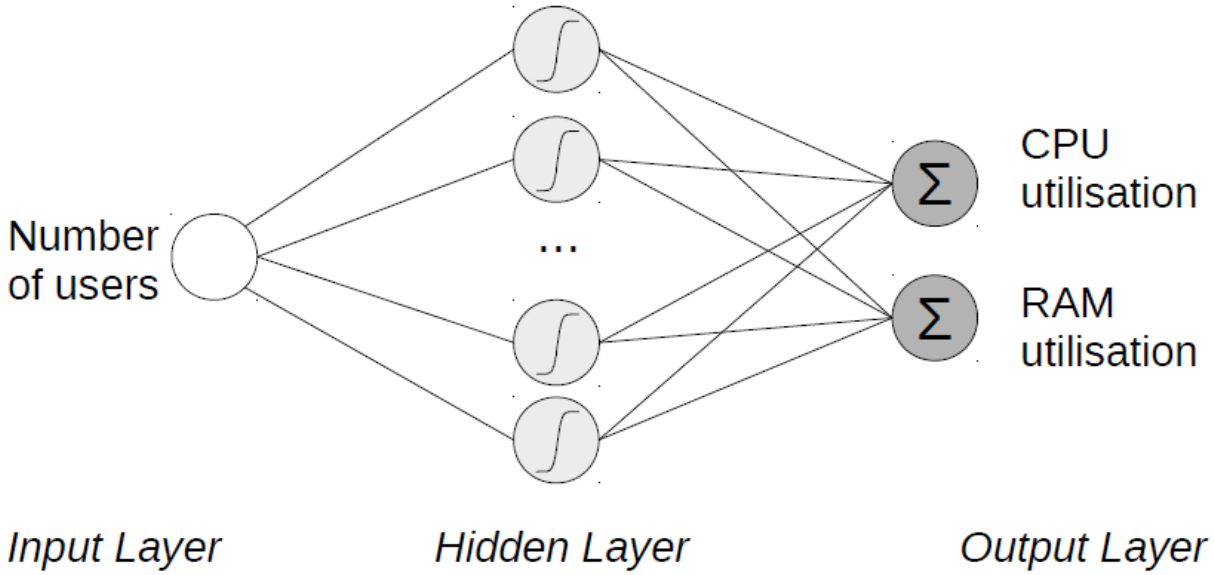
Capacity Estimation and Normalisation

- Linux kernel file: */proc/cpuinfo*;
- Mpstat: *%steal, %idle, active_memory*;
- Frequencies: $fr_1, \ldots fr_n$;
- n_{max_cores}, f_{rmax}, RAM_{max};

- $cpuCapacityNorm = \frac{(100 - \%steal) \sum_{i=0}^{n} fr_i}{100\ n_{max\_cores}\ fr_{max}}$
- $cpuLoadNorm = \frac{(100 - \%idle - \%steal) \sum_{i=0}^{n} fr_i}{100\ n_{max\_cores}\ fr_{max}}$
- $cpuCapacity(vmt) = \frac{1}{|V|} \sum_{vmt_i \in V} \frac{cpuCapacity(vmt_i) \times cpuSpec(vmt_i)}{cpuSpec(vmt)}$
- $ramLoadNorm = \frac{\text{active\_memory}}{\text{RAM}_{max}}$

ANN based online regression

- Learning rate and Momentum;
  - Increase learning rate in the beginning and when anomaly is detected;
  - Increase momentum at later stages and when no anomaly is detected;
- Online training and filtering;



VM type selection algorithm

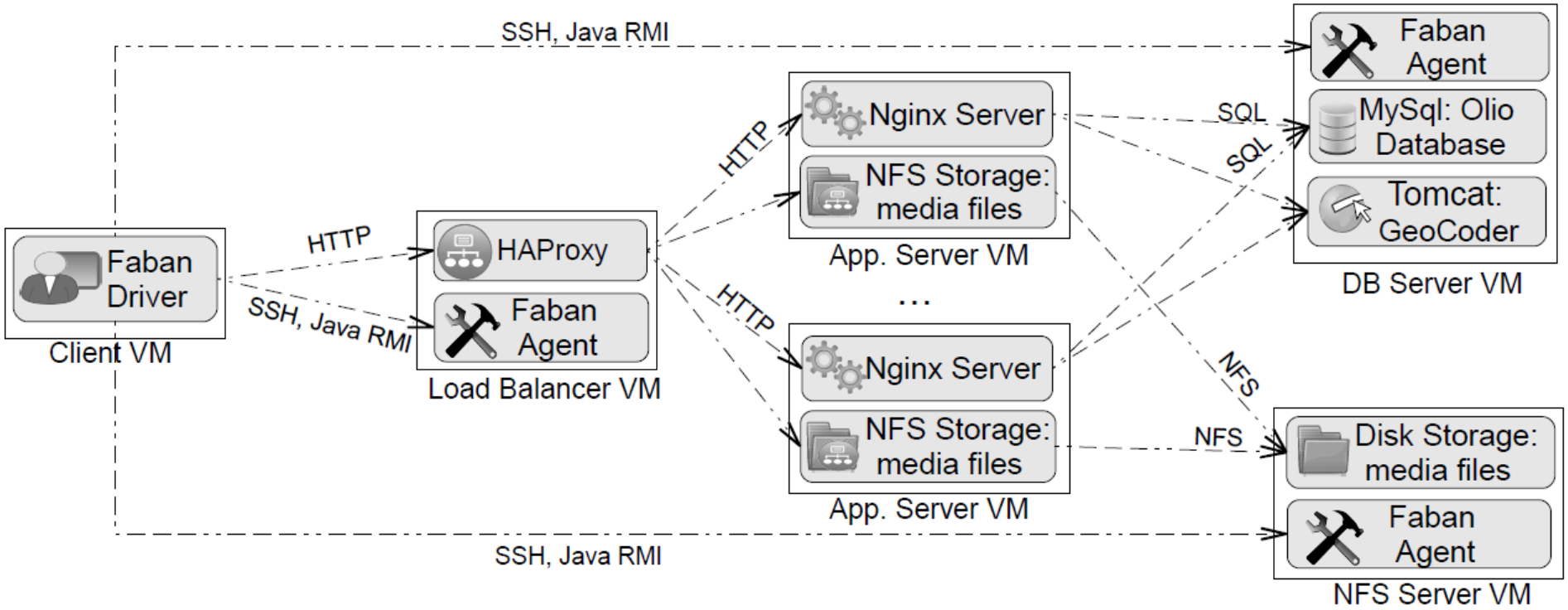
- For each VM type:
  - Estimate its capacity;
  - Estimate how many users it can serve;
- Choose best VM type in terms of cost per user;

ALGORITHM 6: Dynamic VM Type Selection (DVTS).

input : $VT, ann, \Delta, minU, maxU$

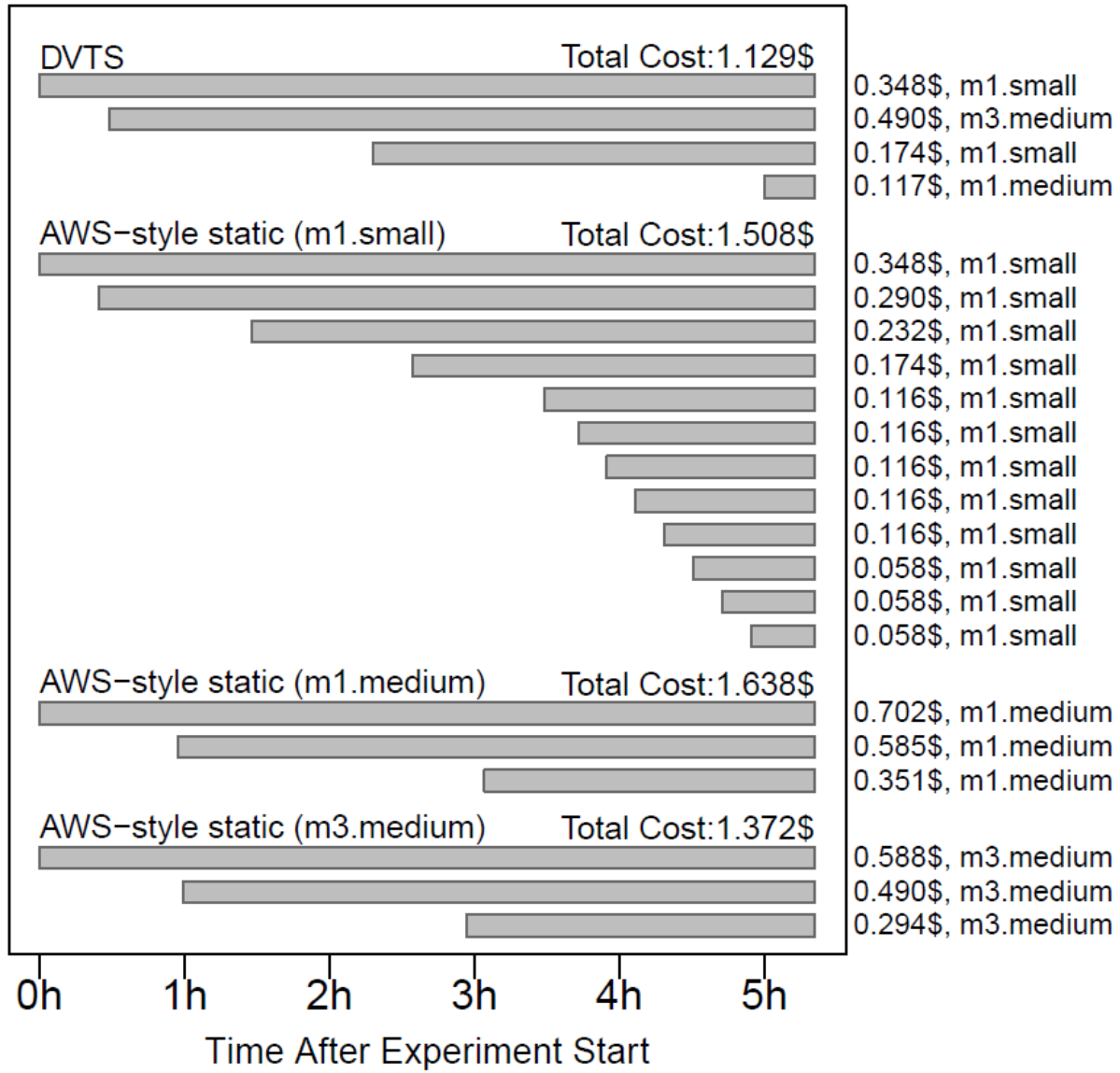
```
1  bestVmt ⟵ null;
2  bestCost ⟵ 0;
3  for vmt ∈ VT ;                                   // Inspect all VM types
4  do
5      cpuCapacity ⟵ vmt's norm. CPU capacity ;
6      ramCapacity ⟵ vmt's norm. RAM capacity;
7      vmtCost ⟵ vmt's cost per time unit;
8      userCapacity ⟵ 0;
9      n ⟵ minU;
10     while True ;                                 // Find how many users it can take
11     do
12         cpu, ram ⟵ predict(ann, n, minU, maxU);
13         if cpu < cpuCapacity and ram < ramCapacity then
14             userCapacity ⟵ n;
15         else
16             break;
17         end
18         n ⟵ n + Δ;
19     end
       // Approximate the cost for a user per time unit
20     userCost ⟵ vmtCost / userCapacity;
       // Find the cheapest VM type
21     if userCost < bestCost then
22         bestCost ⟵ userCost;
23         bestVmt ⟵ vmt;
24     end
25 end
26 return bestVmt;
```

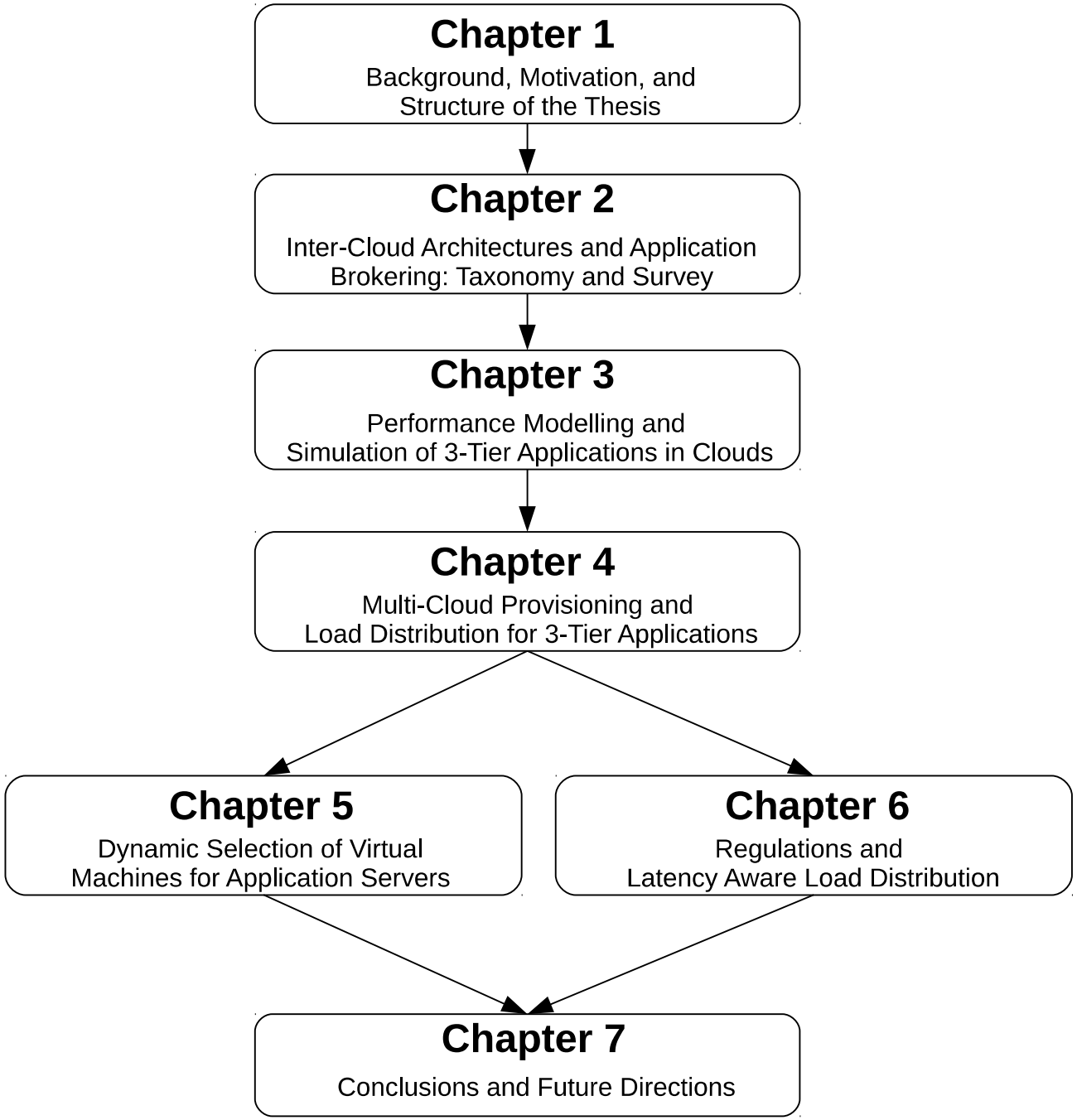
Experimental setup and workload

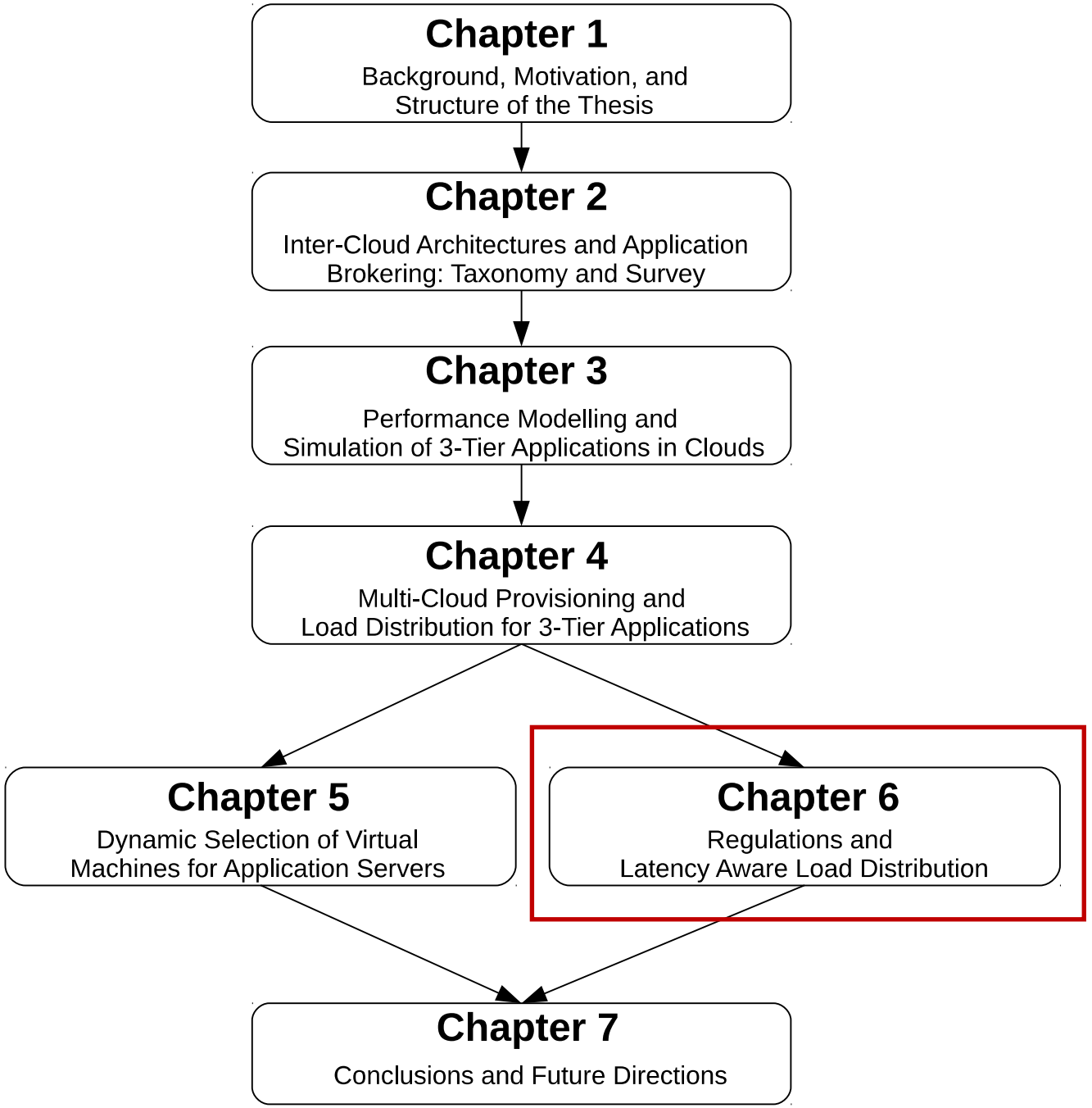


- CloudStone in AWS EC2;
- Choose best VM type in terms of cost per user;
- Increasing workload for 5 hours;
- Workload change after 3.5 hours;
- Baseline – AWS-like autoscaling;

Experimental Results



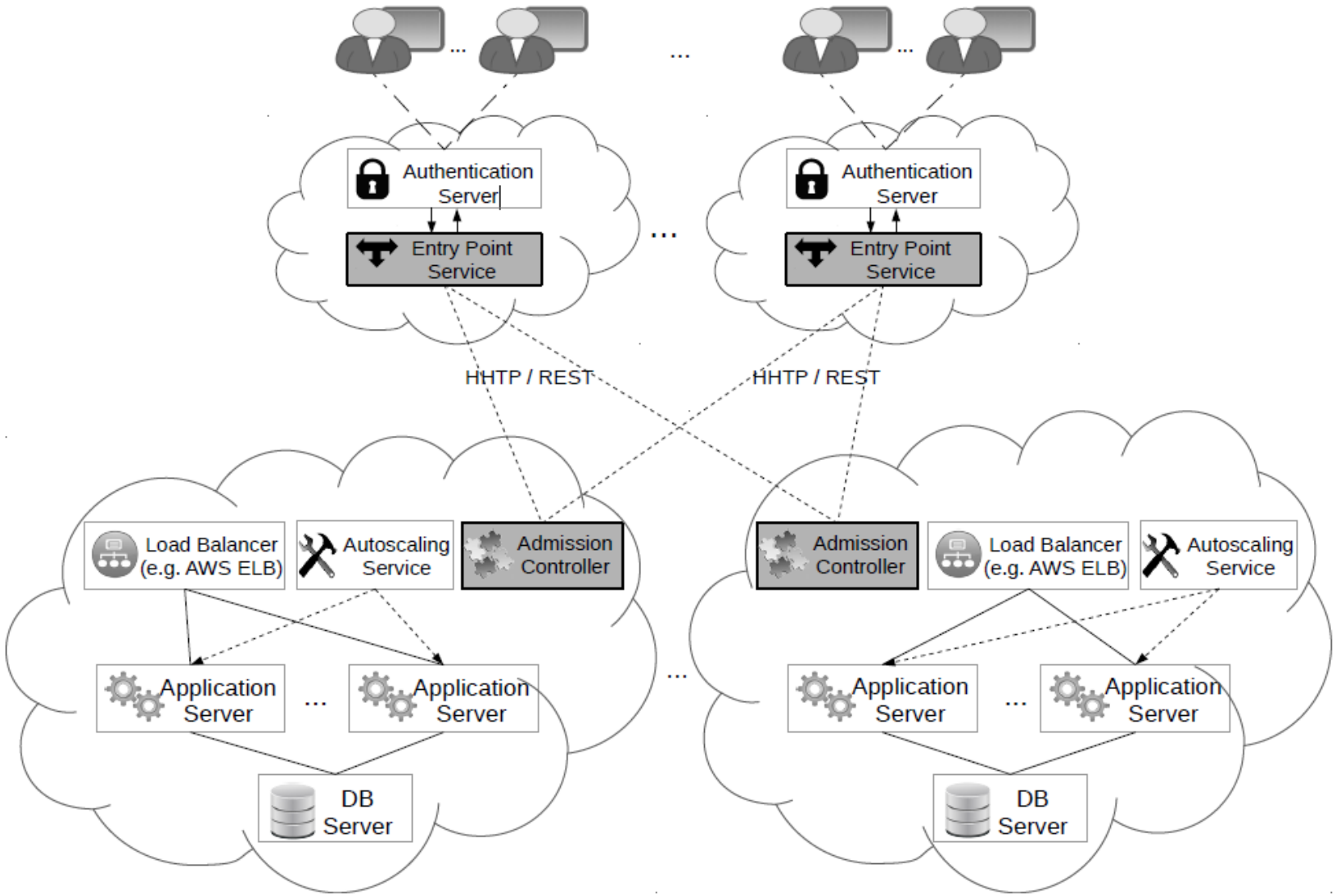

Chapter 1
Background, Motivation, and Structure of the Thesis
Chapter 2
Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey
Chapter 3
Performance Modelling and Simulation of 3-Tier Applications in Clouds
Chapter 4
Multi-Cloud Provisioning and Load Distribution for 3-Tier Applications
Chapter 5
Dynamic Selection of Virtual Machines for Application Servers
Chapter 6
Regulations and Latency Aware Load Distribution
Chapter 7
Conclusions and Future Directions


Chapter 1
Background, Motivation, and Structure of the Thesis
Chapter 2
Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey
Chapter 3
Performance Modelling and Simulation of 3-Tier Applications in Clouds
Chapter 4
Multi-Cloud Provisioning and Load Distribution for 3-Tier Applications
Chapter 5
Dynamic Selection of Virtual Machines for Application Servers
Chapter 6
Regulations and Latency Aware Load Distribution
Chapter 7
Conclusions and Future Directions

Background and Objectives

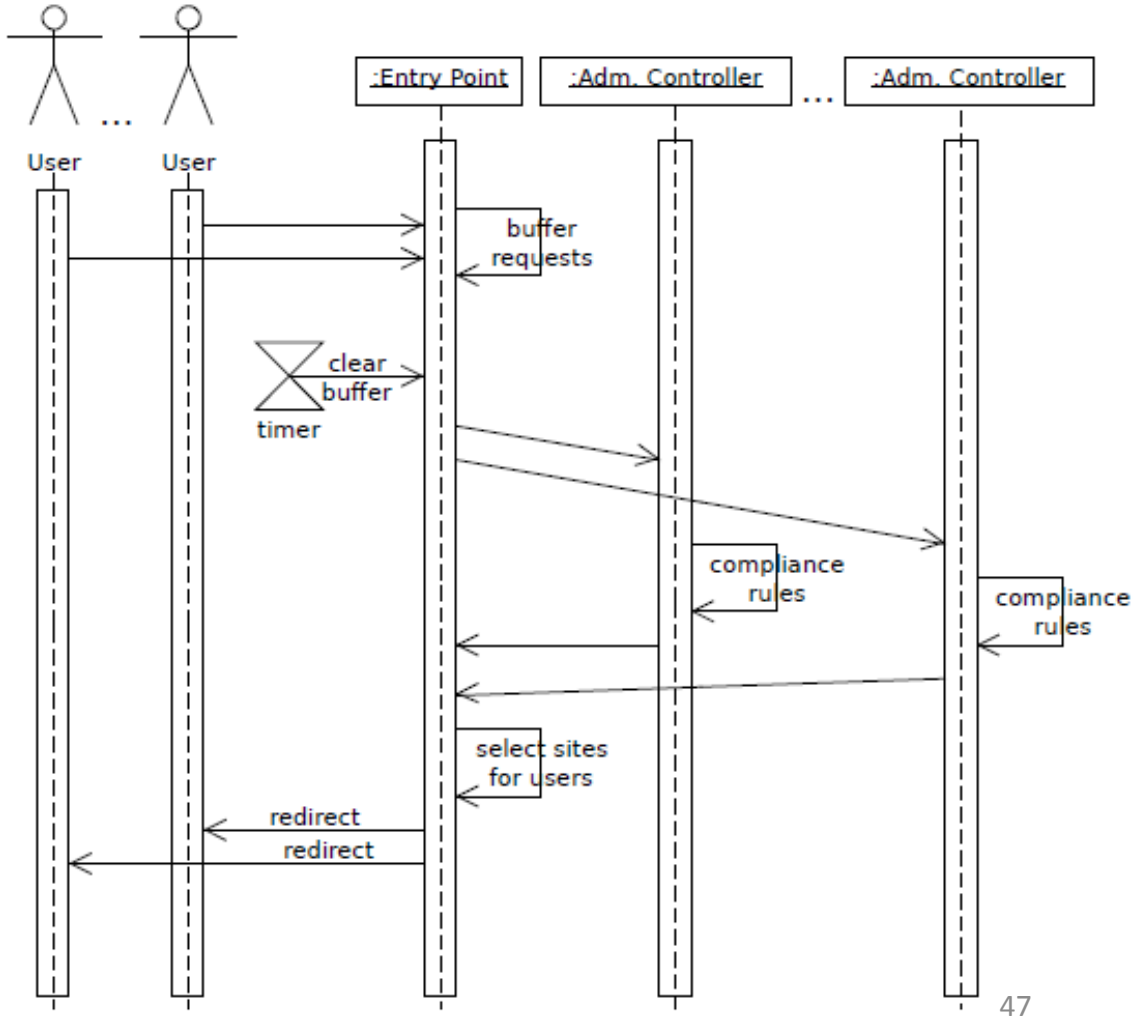
- How to implement the system from Chapter 4 with modern software technologies;
- How to easily model user redirection requirements;

Scope


Authentication Server
Entry Point Service
Authentication Server
Entry Point Service
HHTP / REST
HHTP / REST
Load Balancer (e.g. AWS ELB)
Autoscaling Service
Admission Controller
Admission Controller
Load Balancer (e.g. AWS ELB)
Autoscaling Service
Application Server
Application Server
Application Server
Application Server
DB Server
DB Server

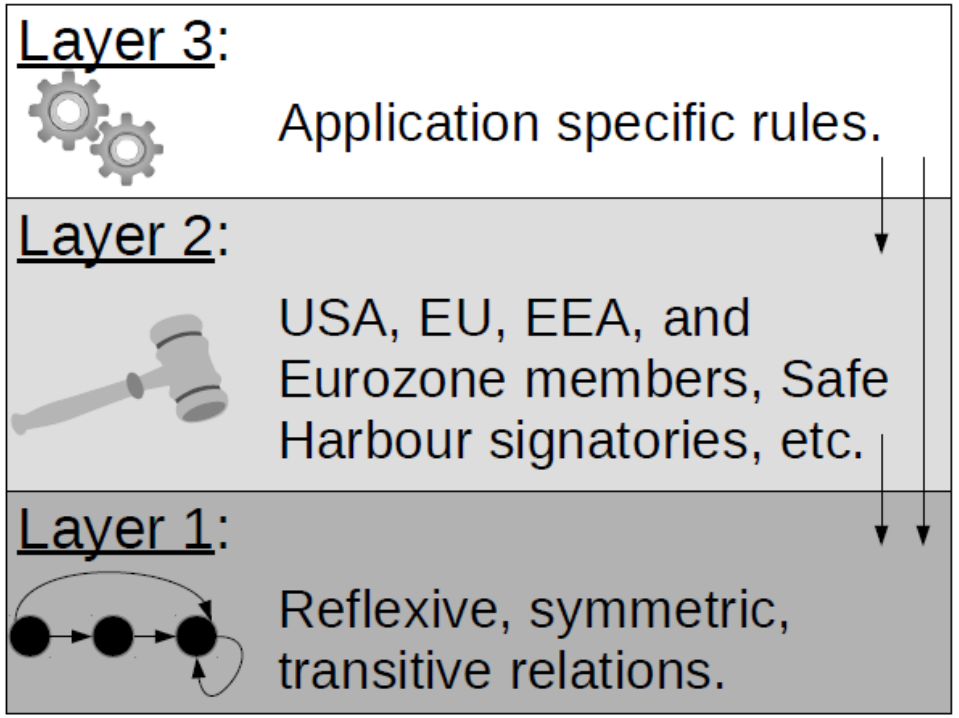
Entry Point – Admission Controller interaction

- Restful web servers;
- Entry Point buffers and sends requests in batch;
- Admission Controller uses a rule inference engine;
- Entry Point choses optimal cloud site.



Admission rules

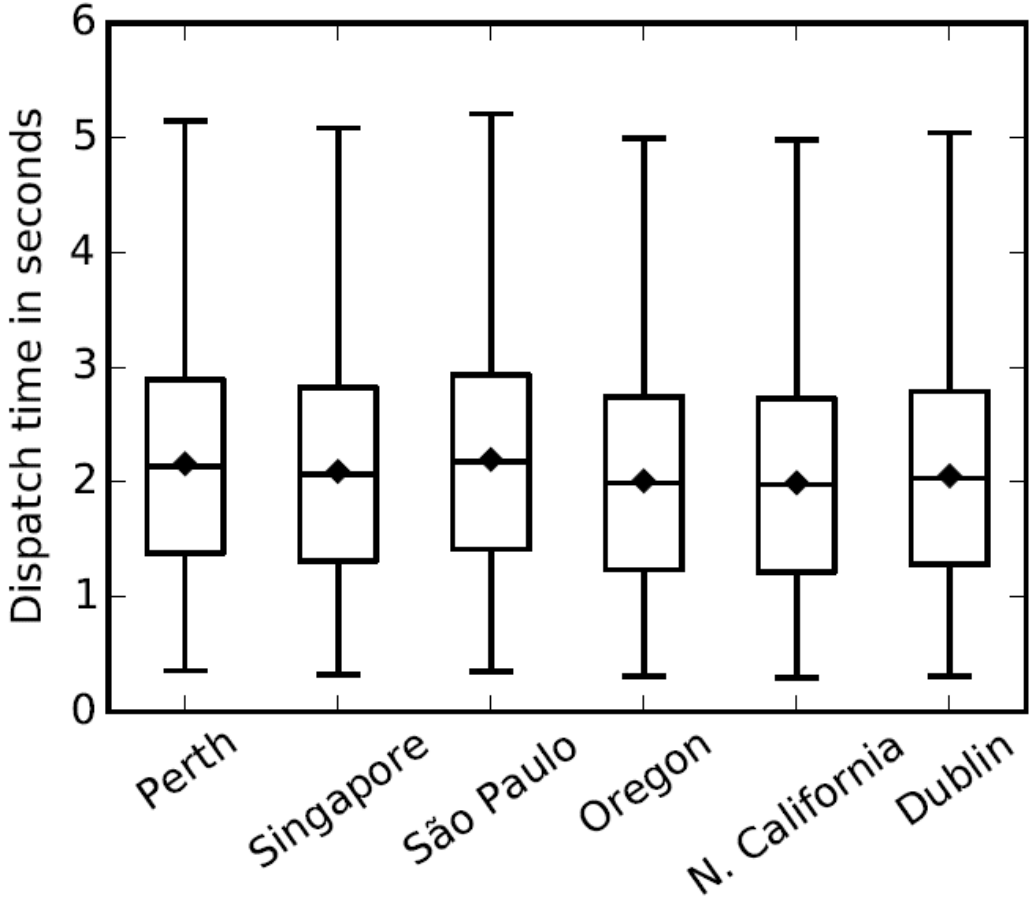
- Drools rule inference engine;
- 3 layers of rules;
- Polymorphism and rules;
- Admission through contradiction.



Experimental setup and workload

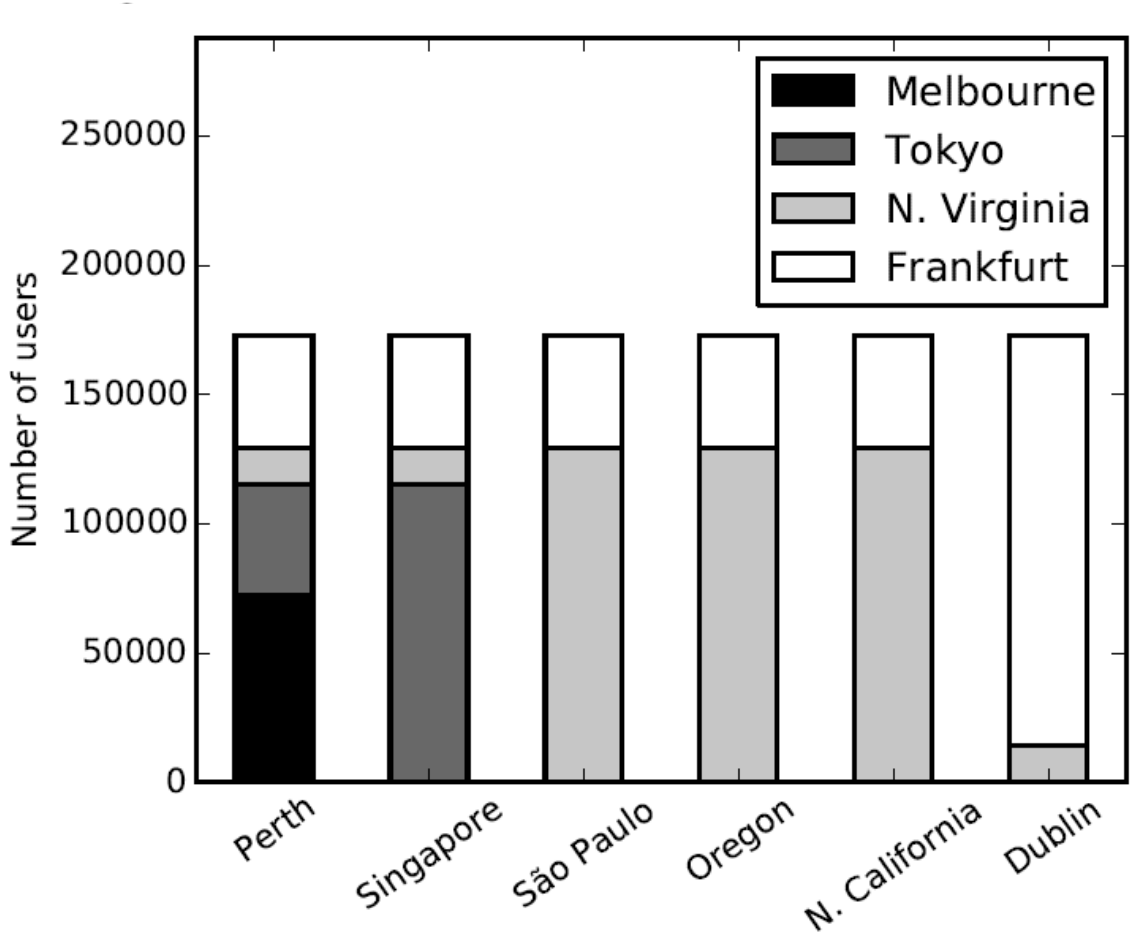
- 24 hours, 2 users per second;
- 50% of users require PCI-DSS compliant clouds;
- Random citizenship: Germany, USA, Australia, or Canada;
- 50% of US citizens are government officials.

Results: dispatch times and destinations

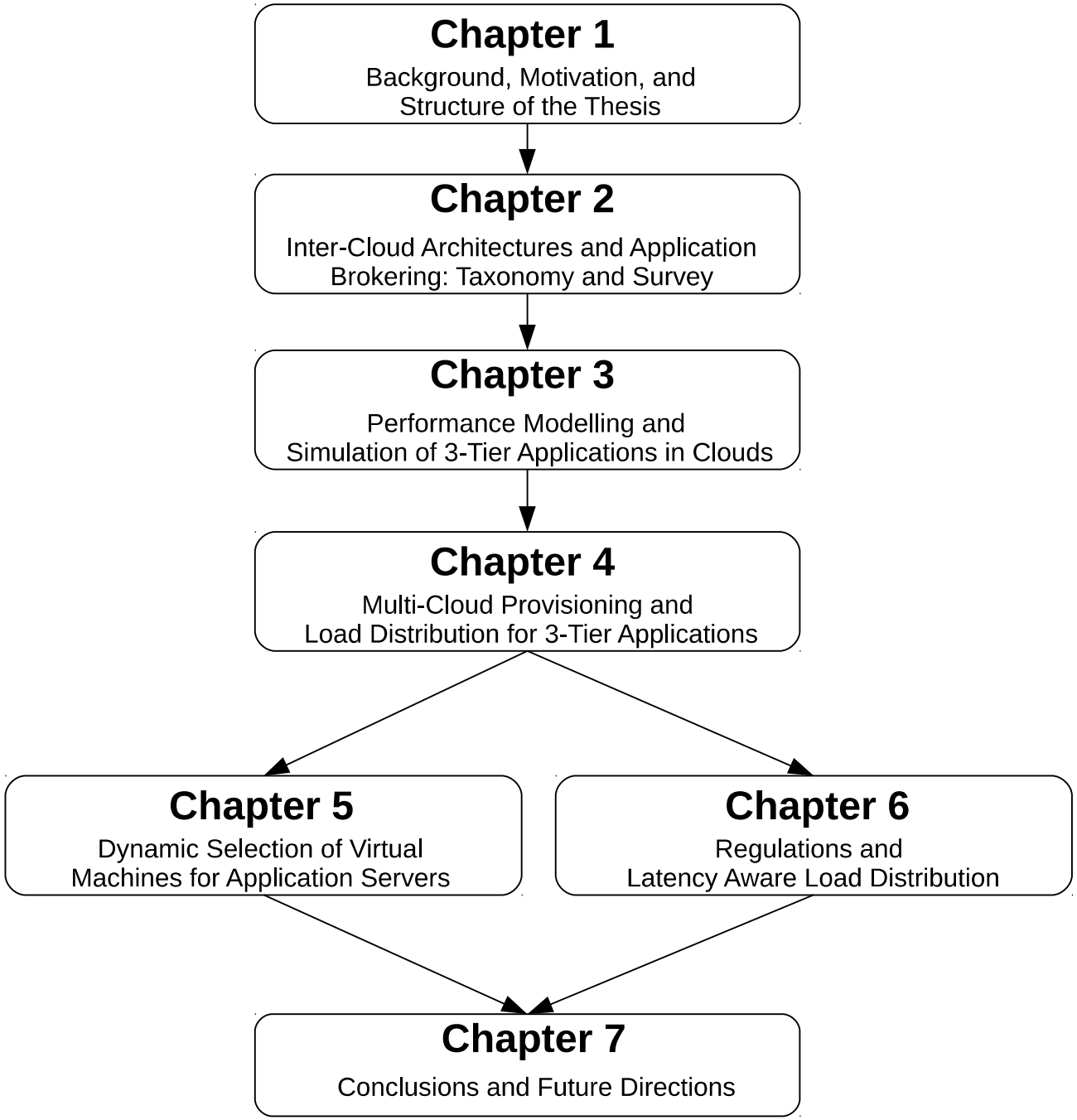


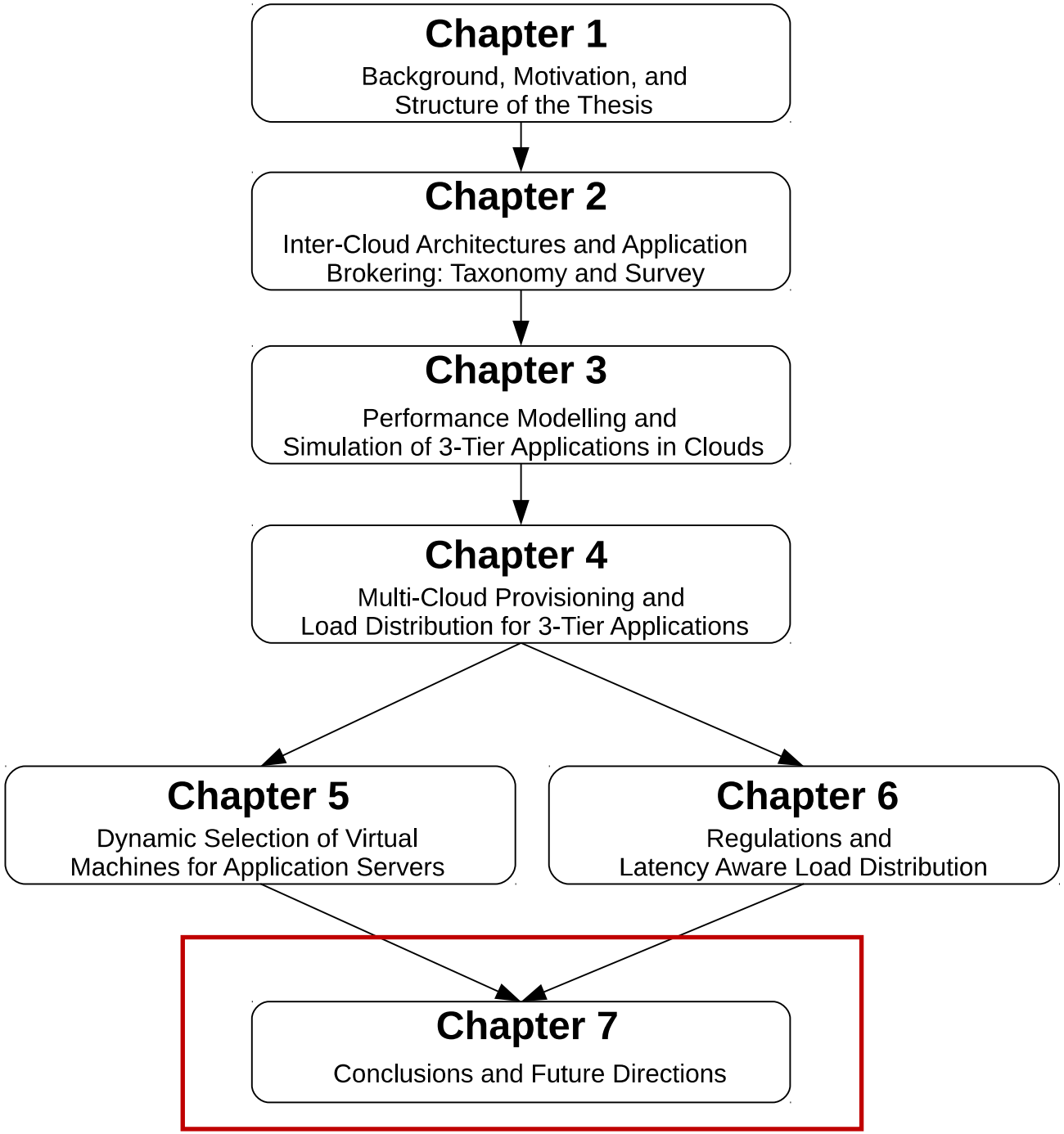
	N.Virginia	Frankfurt	Melbourne	Tokyo
Perth	137.0	180.0	**20.0**	124.0
Singapore	121.0	131.0	69.0	**44.0**
São Paulo	**61.0**	106.0	224.0	140.0
Oregon	**38.0**	73.0	88.0	45.0
N.Cal.	**41.0**	84.0	84.0	52.0
Dublin	39.0	**11.0**	159.0	108.0

Results: dispatch times and destinations



	N.Virginia	Frankfurt	Melbourne	Tokyo
Perth	137.0	180.0	**20.0**	124.0
Singapore	121.0	131.0	69.0	**44.0**
São Paulo	**61.0**	106.0	224.0	140.0
Oregon	**38.0**	73.0	88.0	45.0
N.Cal.	**41.0**	84.0	84.0	52.0
Dublin	39.0	**11.0**	159.0	108.0


Chapter 1
Background, Motivation, and Structure of the Thesis
Chapter 2
Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey
Chapter 3
Performance Modelling and Simulation of 3-Tier Applications in Clouds
Chapter 4
Multi-Cloud Provisioning and Load Distribution for 3-Tier Applications
Chapter 5
Dynamic Selection of Virtual Machines for Application Servers
Chapter 6
Regulations and Latency Aware Load Distribution
Chapter 7
Conclusions and Future Directions


Chapter 1
Background, Motivation, and Structure of the Thesis
Chapter 2
Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey
Chapter 3
Performance Modelling and Simulation of 3-Tier Applications in Clouds
Chapter 4
Multi-Cloud Provisioning and Load Distribution for 3-Tier Applications
Chapter 5
Dynamic Selection of Virtual Machines for Application Servers
Chapter 6
Regulations and Latency Aware Load Distribution
Chapter 7
Conclusions and Future Directions

Summary

- Proposed a performance model and a simulator for 3-Tier apps in clouds;
- Defined a generic architecture for such applications that honors the key functional and non-functional requirements;
- Proposed a method for VM type selection during autoscaling;
- Proposed and implemented a user redirection approach in Multi-Clouds.

Future Directions

- Provisioning Techniques Using A Mixture of VM Pricing Models;
- Dynamic Replacement of Application Server VMs;
- VM Type Selection In Private Clouds;
- Regulatory Requirements Specification Using Industry Standards;
- Generalisation to Multi-Tier Applications.

List of publications

- **Nikolay Grozev** and Rajkumar Buyya, “Inter-cloud Architectures and Application Brokering: Taxonomy and Survey”, *Software: Practice and Experience*, John Wiley & Sons, Ltd, vol. 44, no. 3, pp. 369–390, 2014;
- **Nikolay Grozev** and Rajkumar Buyya, “Performance Modelling and Simulation of Three-Tier Applications in Cloud and Multi-Cloud Environments”, *The Computer Journal*, Oxford University Press, vol. 58, no. 1, pp. 1–22, 2015;
  - Nitisha Jain, **Nikolay Grozev**, J. Lakshmi, Rajkumar Buyya , “*PriDynSim: A Simulator for Dynamic Priority Based I/O Scheduling for Cloud Applications*”, Proceedings of the IEEE International Conference on Cloud Computing for Emerging Markets, 2015 (In Press);
- **Nikolay Grozev** and Rajkumar Buyya, “Multi-Cloud Provisioning and Load Distribution for Three-tier Applications”, *ACM Transactions on Autonomous and Adaptive Systems*, vol. 9, no. 3, pp. 13:1–13:21, 2014;
- **Nikolay Grozev** and Rajkumar Buyya, “Dynamic Selection of Virtual Machines for Application Servers in Cloud Environments”, *Technical Report, CLOUDS Laboratory, The University of Melbourne*, CLOUDS-TR-2016-1
- **Nikolay Grozev** and Rajkumar Buyya, “Regulations and Latency Aware Load Distribution of Web Applications in Multi-Clouds”, *Journal of Supercomputing* (Under Review), 2015;

Acknowledgements

- **Supervisor**: Professor Rajkumar Buyya;
- **Committee**: Professor James Bailey, Dr. Rodrigo Calheiros;
- Dr. Amir Vahid and Dr. Anton Beloglazov;
- Past and Present CLOUDS Lab members and CIS Department;
- Microsoft;
- Amazon Inc;
- Family and Friends.

Q&A

Thank you!