Performance Evaluation of Cloud Services

Jamal Zarak Khan

Introduction

- Cloud computing is everywhere.
- Project aims to model cloud computing services using queuing theory.
- Look at the performance and cost implications of different cloud service providers using these models.

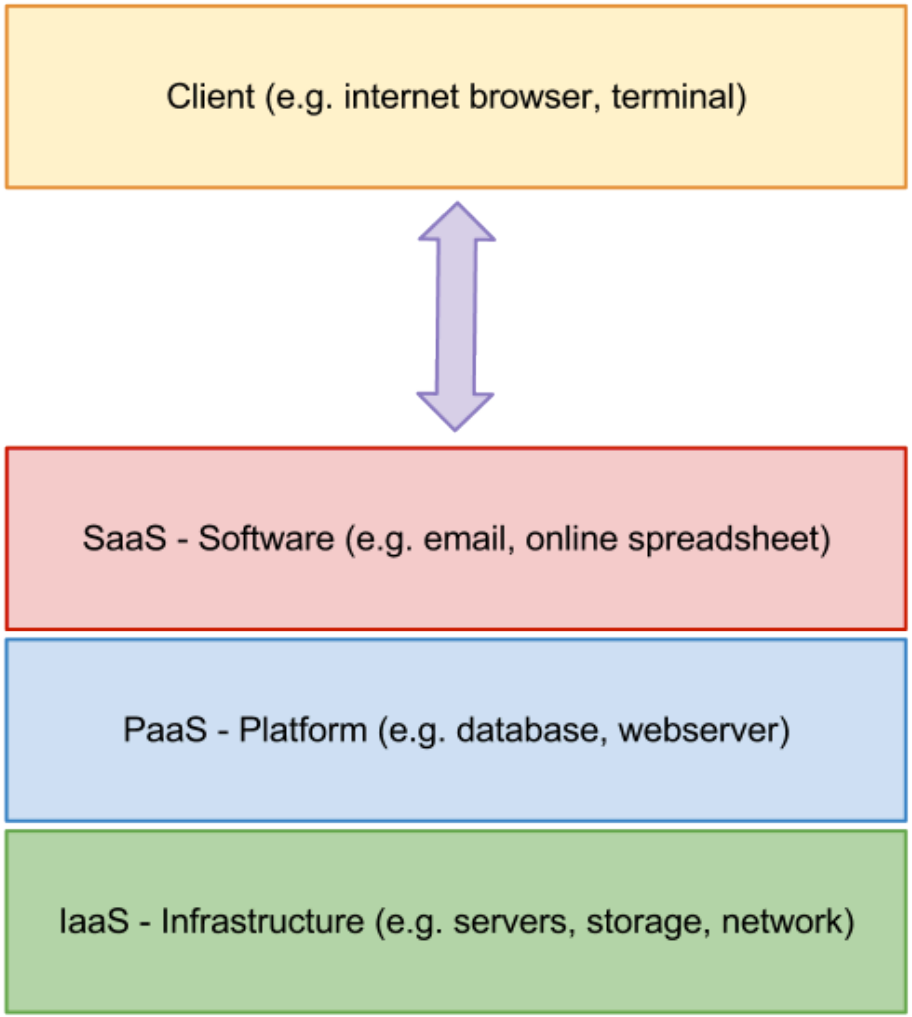
Cloud computing, probability distributions & queuing theory

BACKGROUND

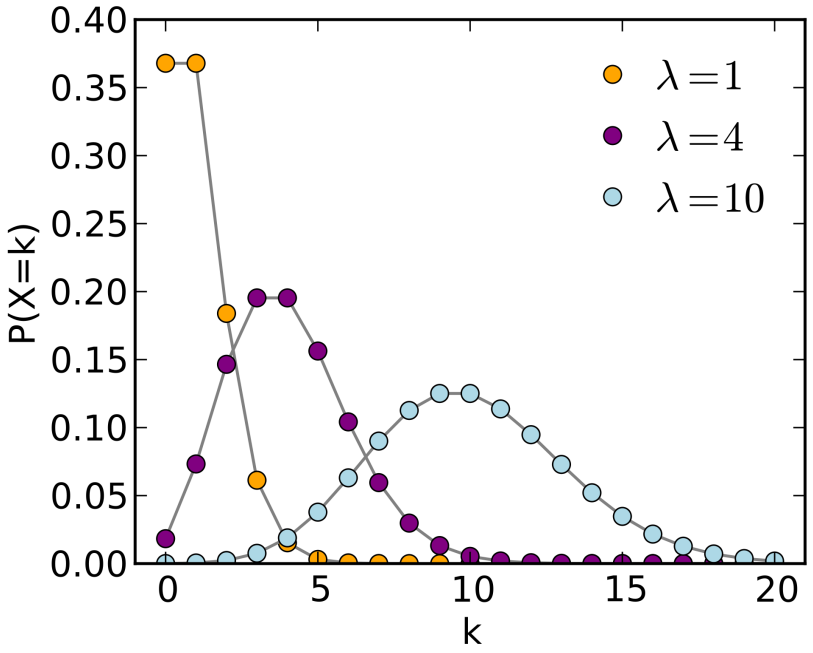
What is cloud computing?

- Provision of computing capabilities as a use-on-demand service
- Outsourcing of computing power (storage, computation and other services)

Layers of Cloud Services



Poisson Distribution



- Given the average rate of an event, conveys the number of occurrences in a given time span.
- PDF is given by:

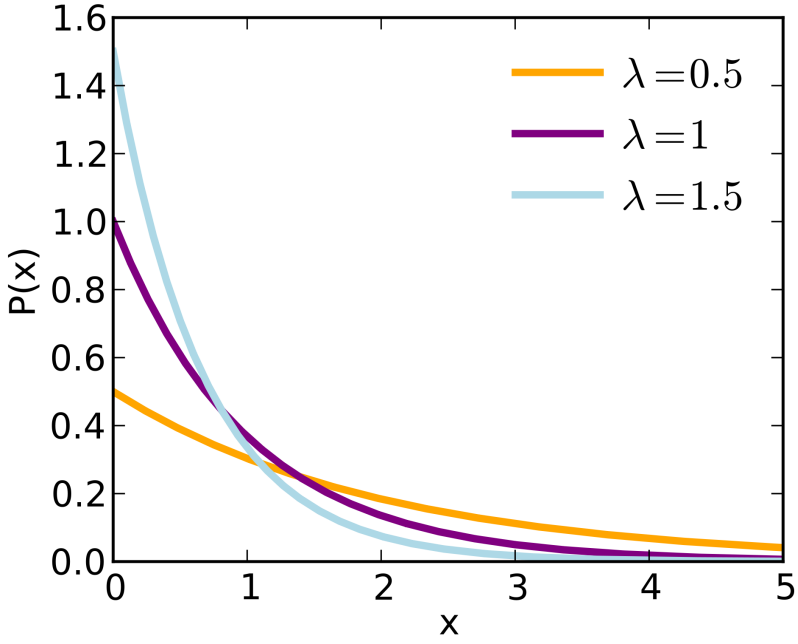
$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Exponential Distribution

- Used for measuring waiting times between successive events in a Poisson distribution.
- Cumulative and probability distribution function are:

$$F(x) = P(X \leq x) = 1 - e^{-\lambda x}$$

$$f(x) = F'(x) = \lambda e^{-\lambda x}$$



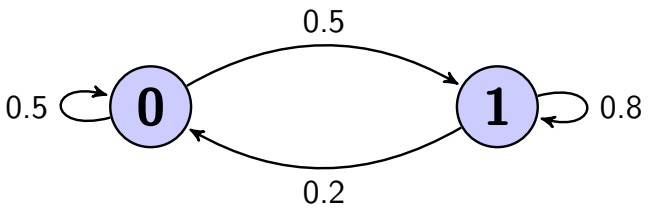
Markov Property

$$P(S \leq t + s | S > t) = P(S \leq s) \quad \forall t, s \geq 0$$

Markov Chains

- A process that moves from state to state with a certain probability.
- Satisfies the Markov property.
- Matrix Q representing the transition probabilities.

$$Q = \begin{pmatrix} 0.5 & 0.5 \\ 0.2 & 0.8 \end{pmatrix}$$



Continuous Time Markov Chain (CTMC)

- Discrete time Markov chains where transitions can happen at any time.
- Time spent in each state is non-negative and has exponential distribution.

Markov Processes

- Continuous time parameter space and discrete event space.
- An example is a birth-death process.

$$a_{ij} = \mu_i q_{ij} \quad \text{where } i \neq j$$

μ_i is the rate out of state i and q_{ij} is the probability of selecting j next.

Kendall's Notation (Queuing node)

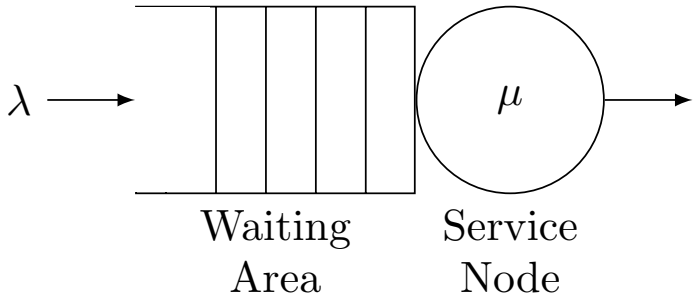
A / S / c

- 'A' - the arrival process
- 'S' - the service time distribution
- 'c' - the number of servers

- M/M/1 (Arrival Markov process, Service time Markov process, single server)

Single Server Queues (SSQs)

- Poisson arrival process (rate parameter λ)
- Queue that arriving tasks join
- Server processing tasks with exponentially distributed service time (parameter μ)



M/M/1 queue

Some definitions

- “Service time” – time taken to serve a customer at a service node.

- “Response time” – total time customer spends in a system.

Some equations

$$\rho = \frac{\lambda}{\mu}$$

Server utilisation

Average response time

$$W = \frac{1}{\mu - \lambda}$$

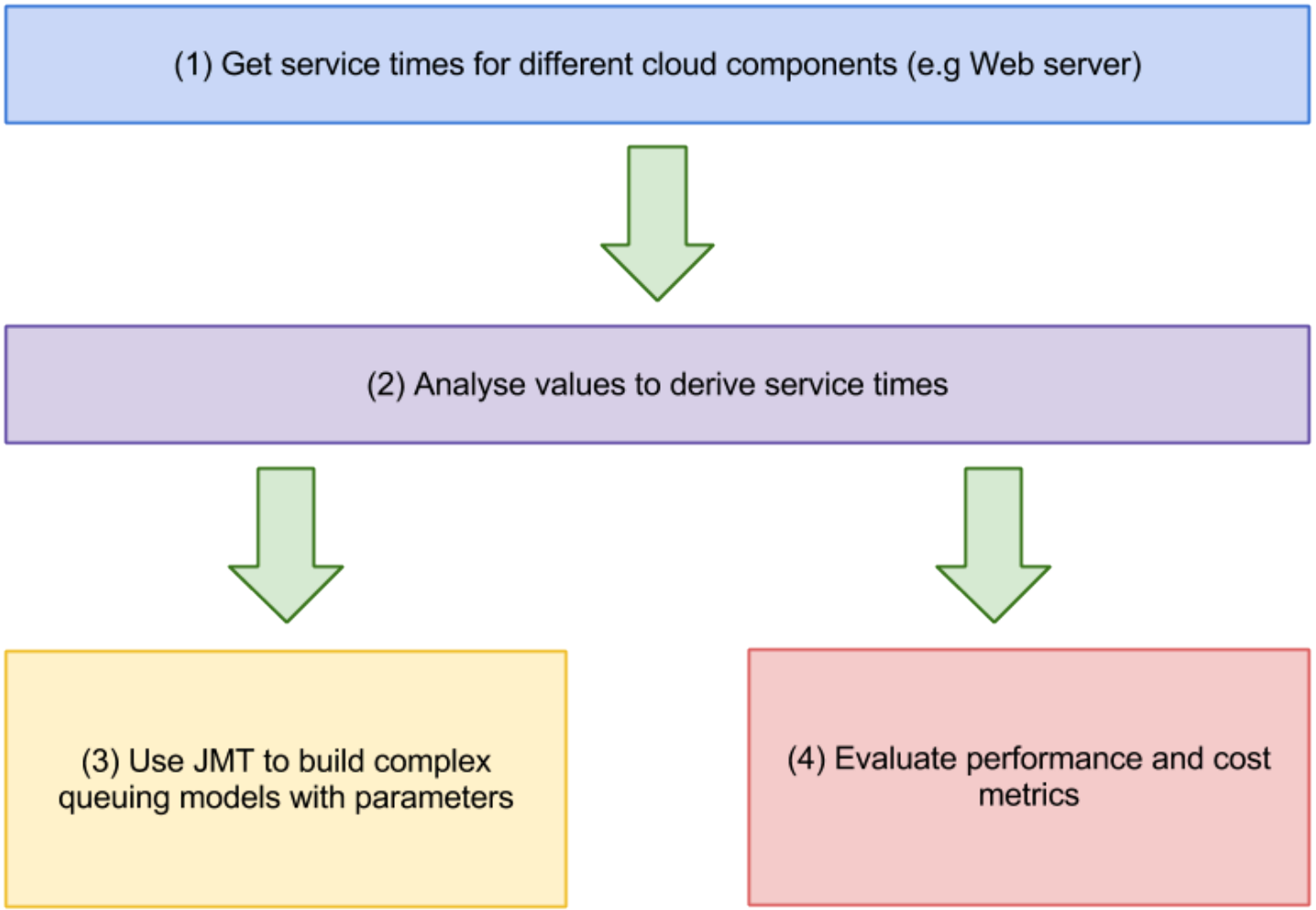
$$\rho < 1$$

Stability condition

Open Queuing Networks

“A network where customers can arrive into the network from an external source and there can also be external departures from the queuing network”

Workflow


(1) Get service times for different cloud components (e.g Web server)
(2) Analyse values to derive service times
(3) Use JMT to build complex queuing models with parameters
(4) Evaluate performance and cost metrics

Service Time Evaluation Tool

IMPLEMENTATION

Service Time Evaluation Tool (STET)

- Command line tool to get the service time distributions of different components.
- Focus on two major components:
  - HTTP web servers
  - Databases

Random data generation

- Typical applications store data with certain properties:
  - Structured format (as records)
  - Record has a tuple (option, value) – with type
  - Constraints on size of value (can also be null)

Twitter to the rescue

- Store tweets and their meta data as test data for the tool.
- Use Twitter Streaming API, providing certain fraction of all tweets (in JSON format).

Jamal Khan @jamalzkhan 17 Jun

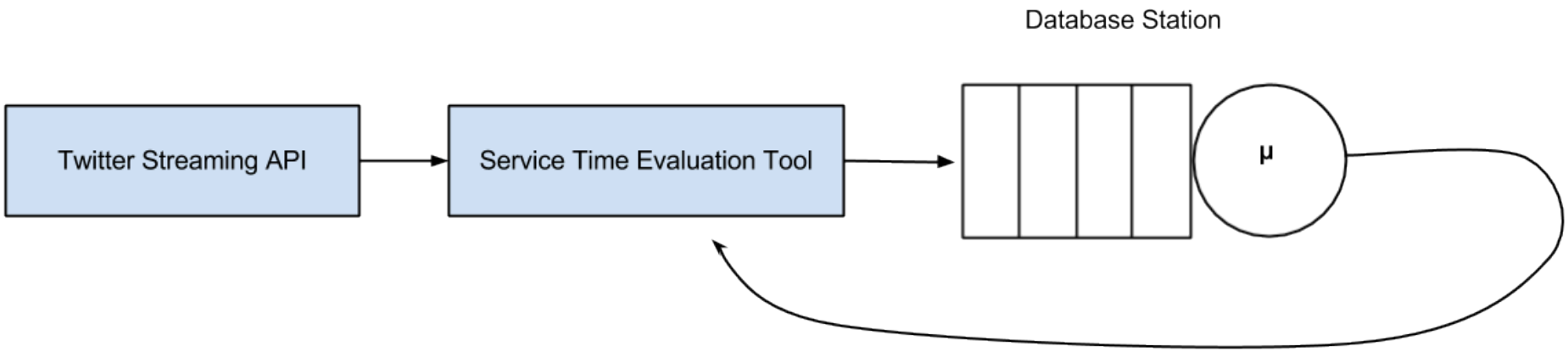
"It is not the mountain we conquer but ourselves." - Edmund Hillary #inspiration

Collapse Reply Delete Favorite More

3:25 AM - 17 Jun 13 from Westminster, London · Details

Modeling database service times

- Focused on two database types:
  - PostgreSQL (SQL database)
  - MongoDB (noSQL JSON oriented datastore)



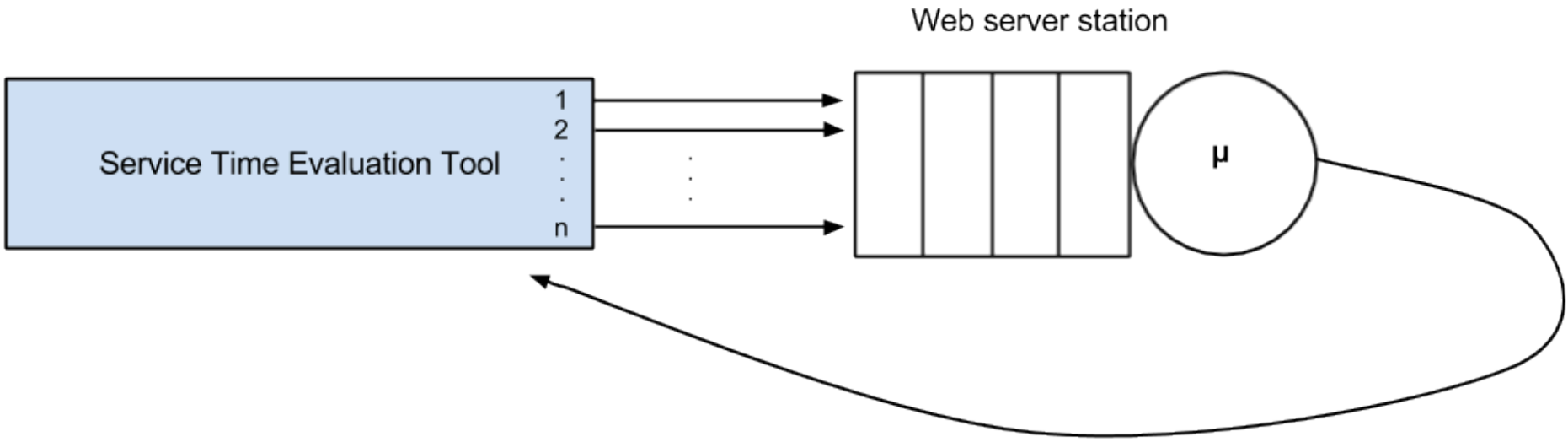
Usage: Database service times

```
...
before = DateTime.now()
# Perform some database action
Database.write(TweetData)
after = DateTime.now()
response_time = after - before
```

```
> python Performance.py -d mongo -i 2000 -s 1000
```

Modeling web server service times

- Created a webserver in Node.js (server side JavaScript).
- Issued GET and POST requests to calculate service times.

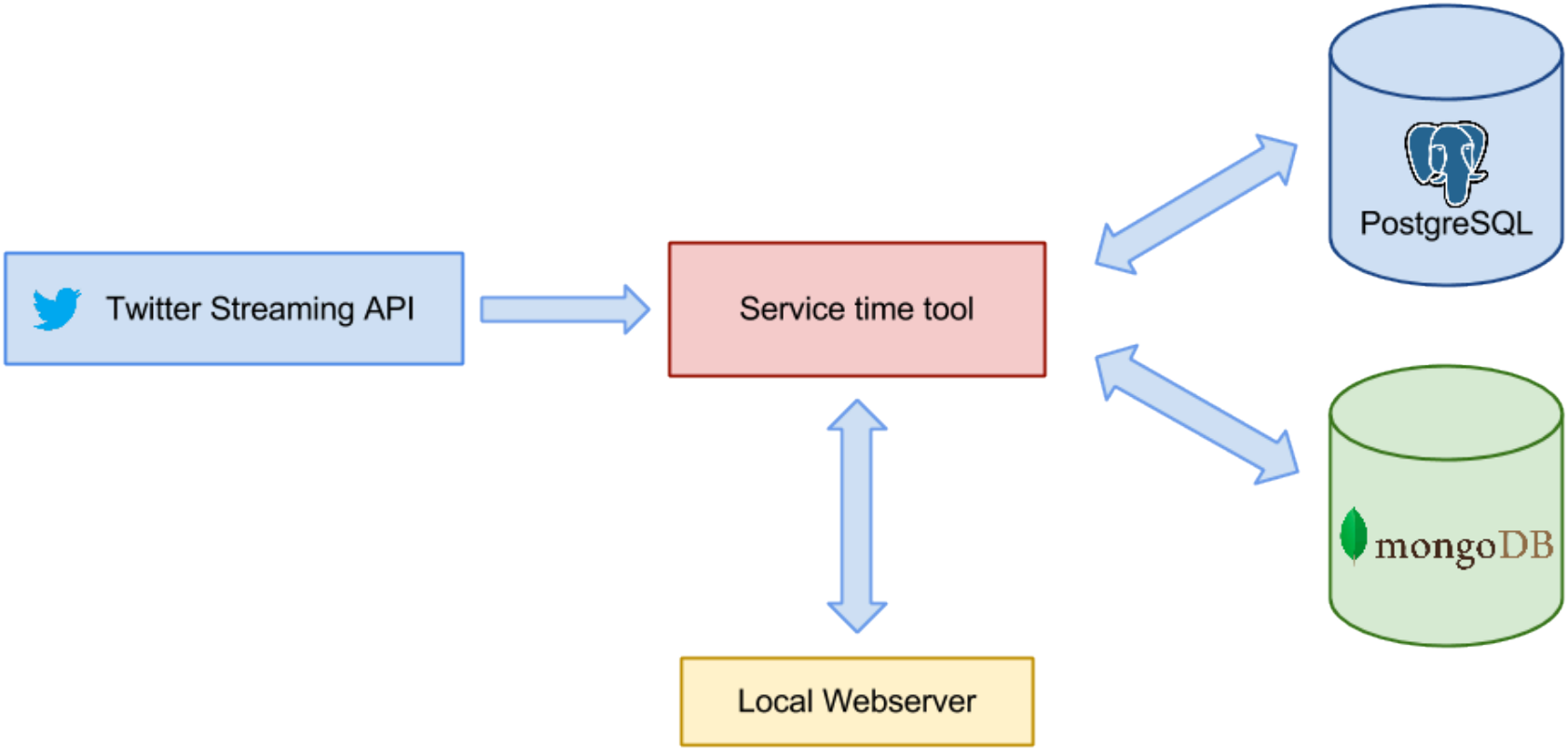


Usage: Web server service times

```
...
before = DateTime.now()
# Perform a web server query
HTTP_connection.request("GET", url)
request = HTTP_connection.get_response()
after = DateTime.now()
response_time = after - before
```

```
> python Performance.py -u localhost -p 80 -g 3000 -x 2000 -n 30
```

Dataflow in Tool

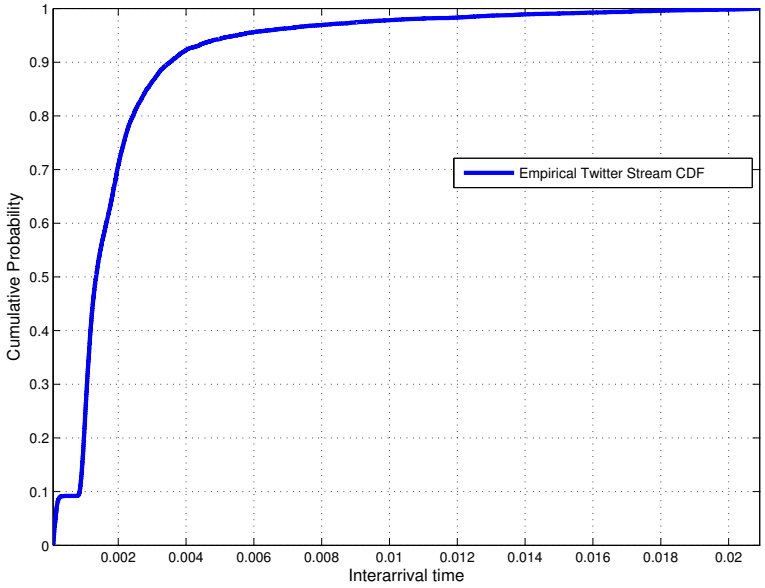


Twitter Stream, service time tool & using JMT

ANALYSIS

Analysis of Twitter Stream

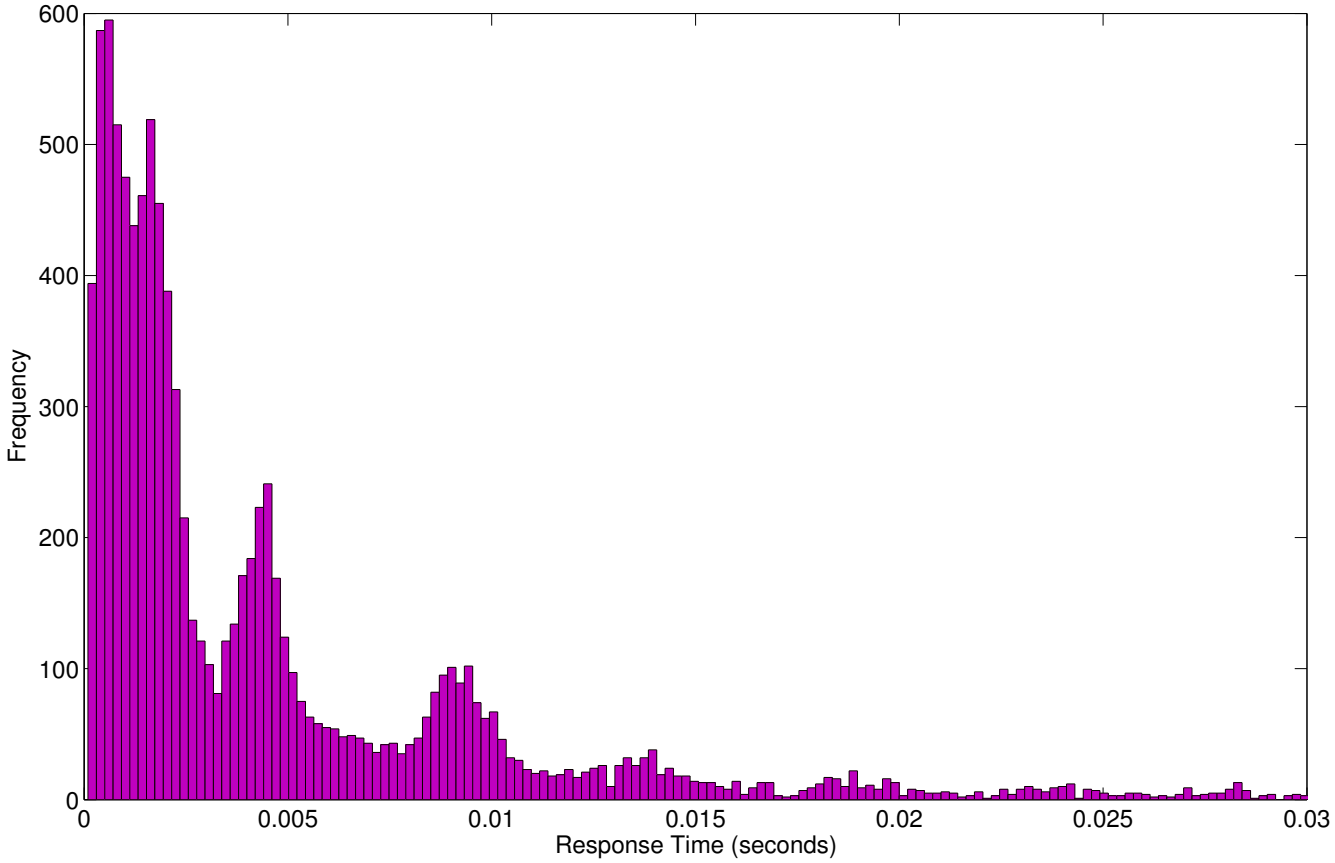
- 8000 tweet arrival events, inter-arrival times logged
- Plotted as a CDF plot
- Model this as an exponential distribution, MLE = 492 tweets per second.



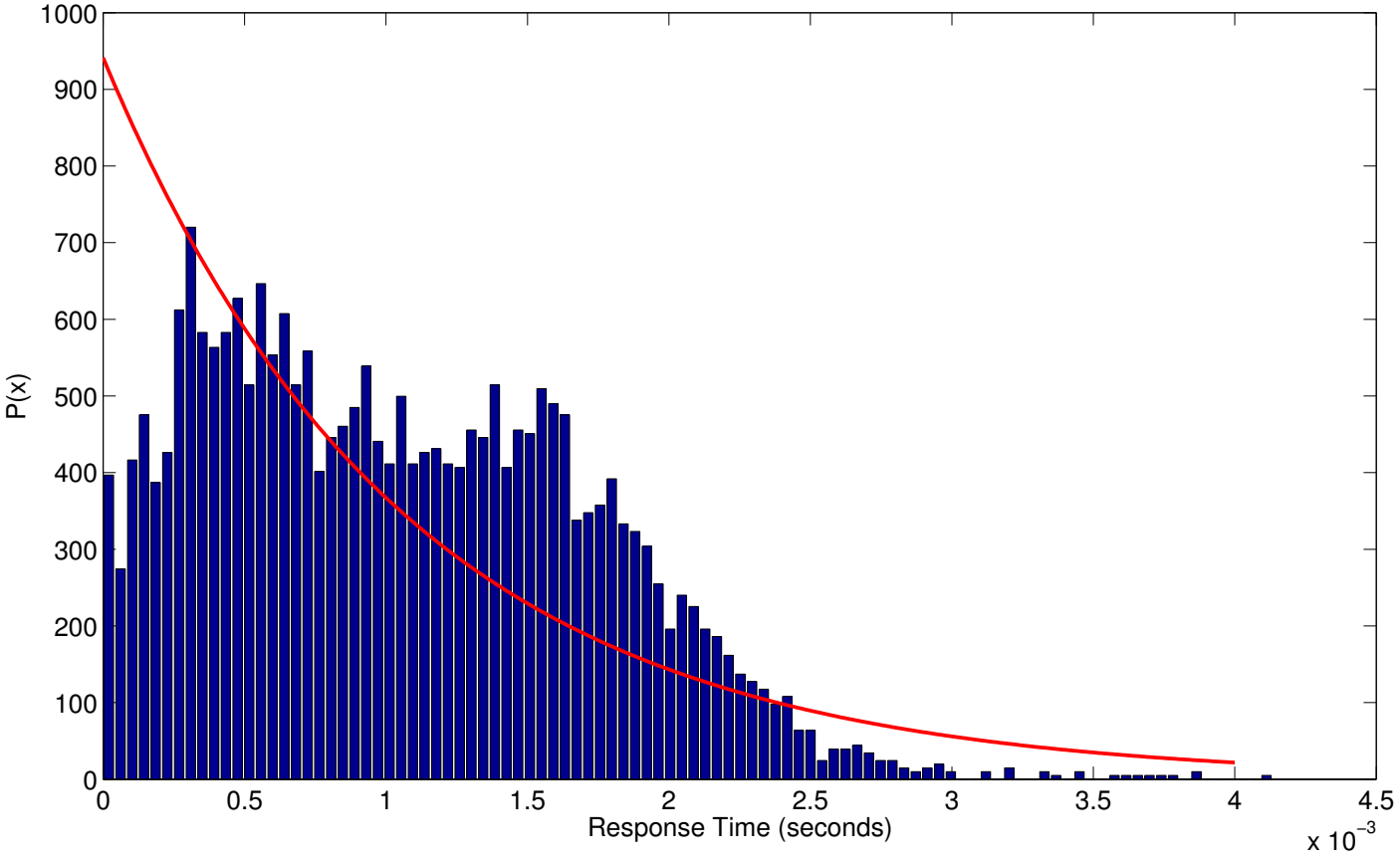
Elastic Compute Cloud (EC2)

- "Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud"
- Used service time tool on two types of instances:
  - Micro instance ($0.025 per instance-hour)
  - Extra-large instance ($0.520 per instance-hour)

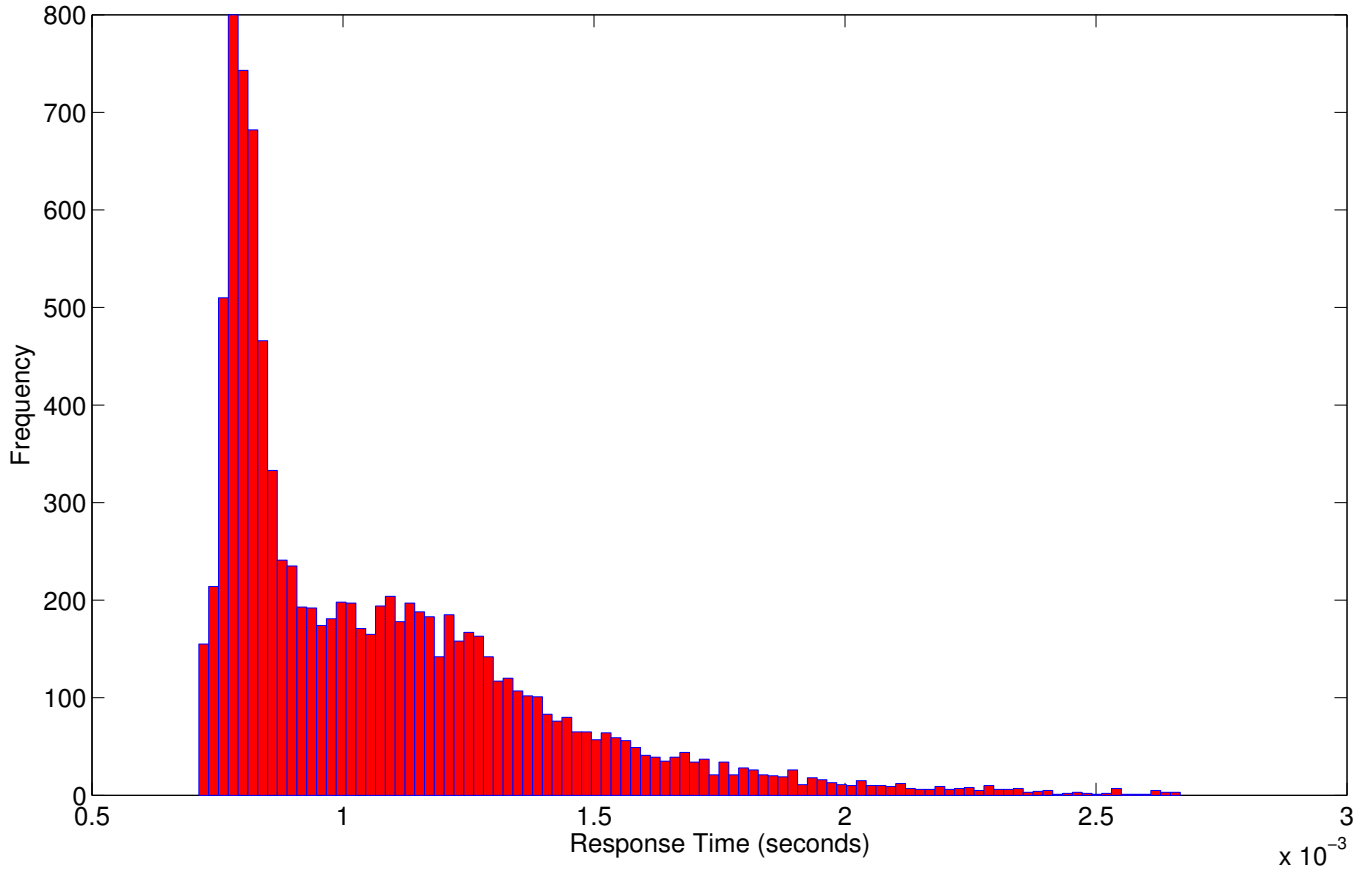
PostgreSQL 5000 reads, 5000 writes



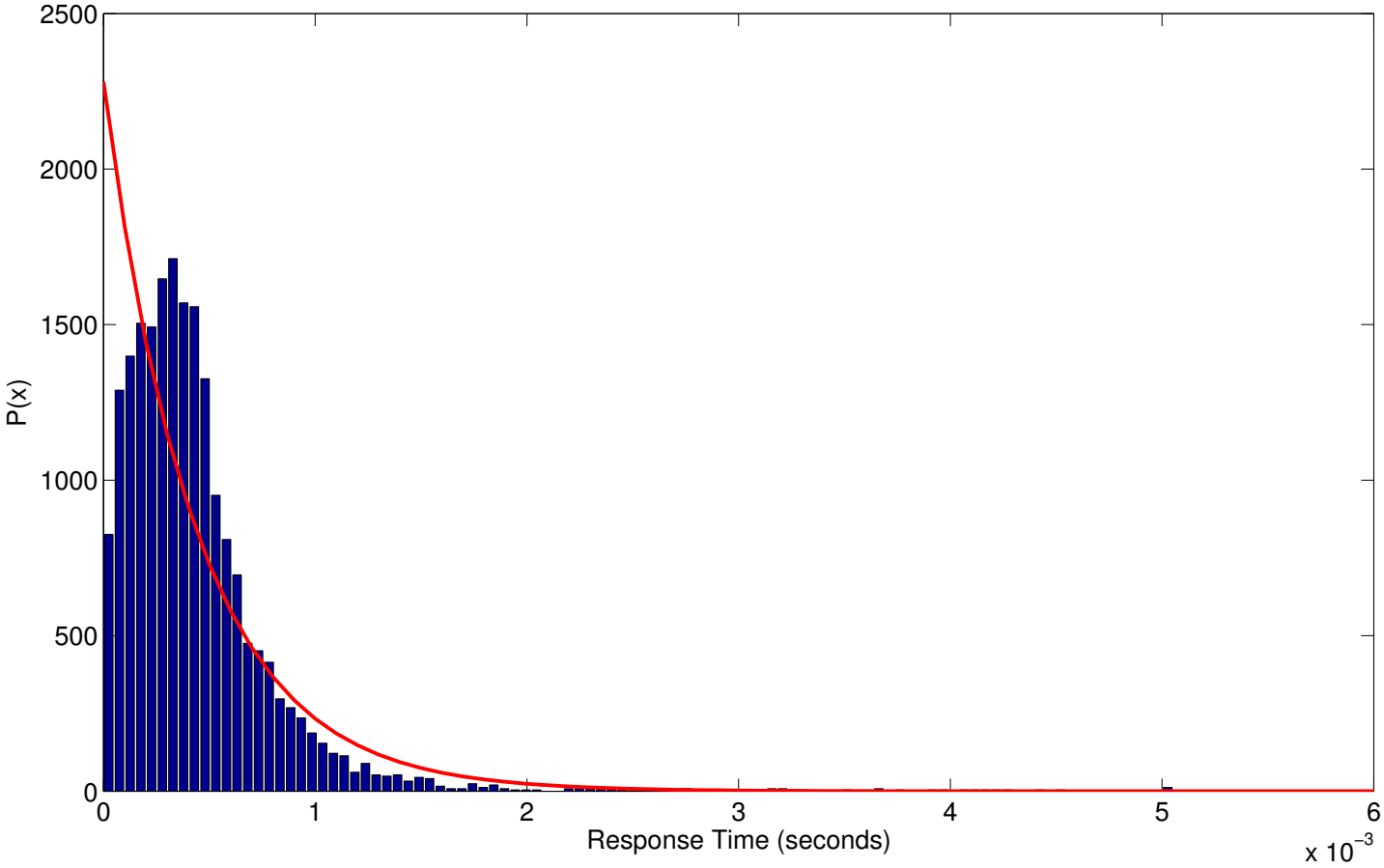
PostgreSQL PDF for reads (micro instance)



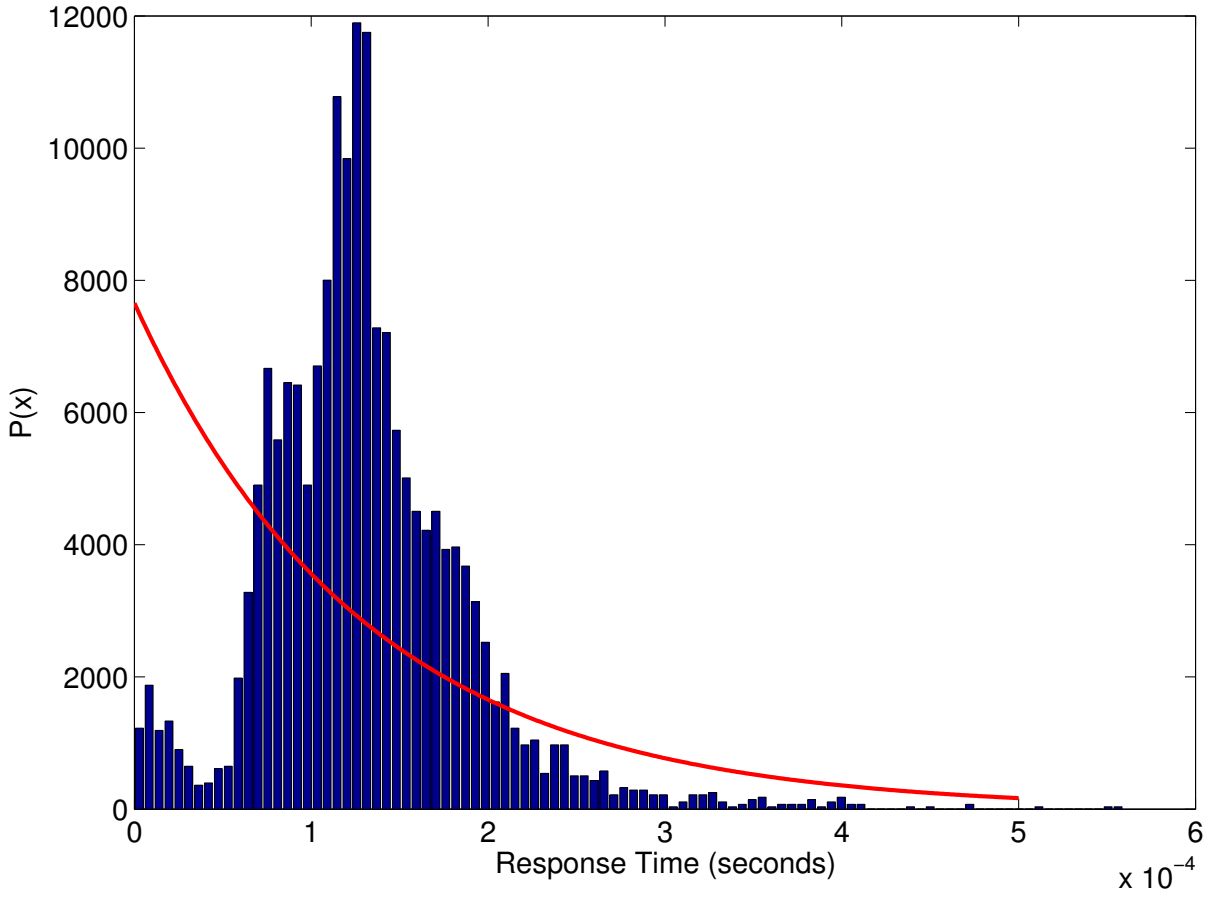
Node.js web server 5000 GET, 5000 POST requests



Node.js PDF for POST requests (micro instance)



MongoDB PDF for writes

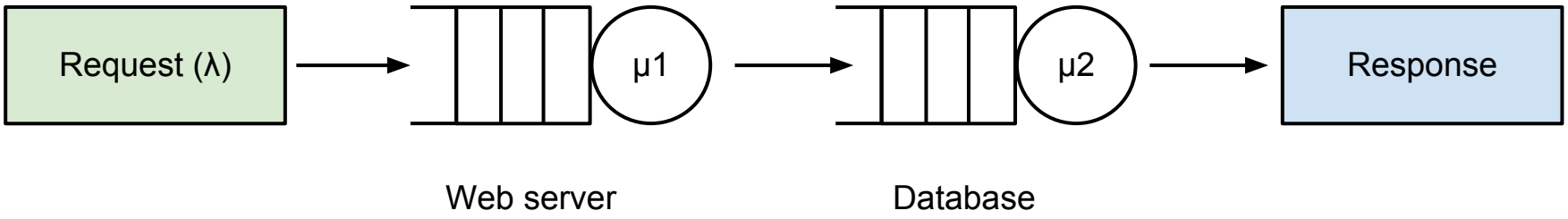


Estimation using MLE

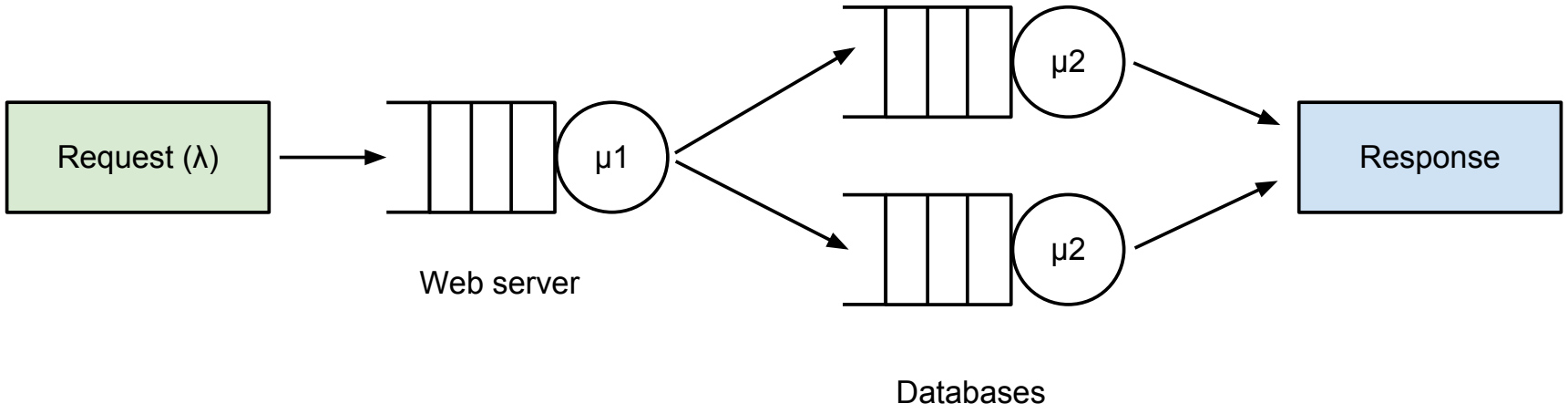
- Experiments carried out for Micro and Extra-large instances.
- Calculated the Maximum-Likelihood estimate for an exponential distribution for:
  - PostgreSQL reads and writes
  - MongoDB reads and writes
  - Node.js GET and POST requests

Java Modeling Tools

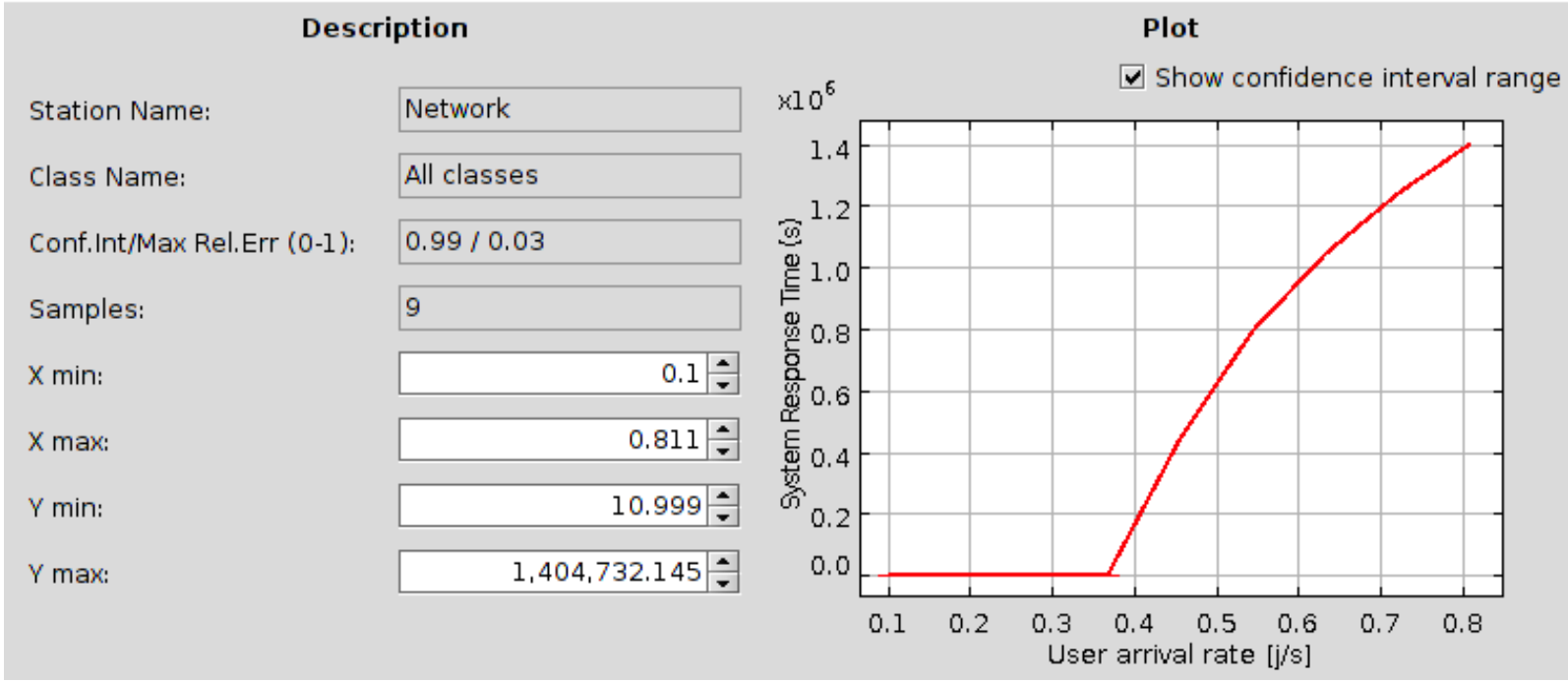
- Open source suite used for performance evaluation, capacity planning and modeling of computer systems.
- JSIM: discrete event simulator for analysis of queuing networks



More complex configuration



Node.js web server + 4 PostgreSQL databases



JMT Findings

- Threshold value, where the system response time increases by a much larger factor than before.
- Building of complex cloud systems much easier theoretically than practically.

Performance + Cost implications

RESULTS

Performance of different instance types

- Performance is calculated as the maximum-likelihood estimate of the response time.
- Extra-large instance type had a quicker mean response rate

Component	Instance type	Operations	μ_{MLE}
MongoDB	Micro		
		Read	43.300
		Write	0.745
	Extra-large		
		Read	67.902
		Write	6.4354
Postgres	Micro		
		Read	0.9091
		Write	0.1266
	Extra-large		
		Read	1.5684
		Write	1.2624
Node.js web server	Micro		
		GET	4.3693
		POST	1.9773
	Extra-large		
		GET	21.6828
		POST	11.2618

Performance vs Cost

- After calculating time per response, we calculated the cost per response.
- Extra-large instances cost **more** per response.
- Increase in cost is not proportionally worth the cost!

Instance Type	Instance-hour cost ($)
Micro	$0.025
Small	$0.065
Medium	$0.130
Large	$0.260
Extra-large	$0.520

Response time description	Ratio Extra-Large vs Micro
Postgres	
Read	12.0627
Write	2.0855
MongoDB	
Read	13.2626
Write	2.4096
Node.js web server	
GET	4.1911
POST	3.6523

JMT Findings

- To get the similar performance using HTTP POST requests to Node.js and MongoDB writes we need 10 micro instance databases to match the extra-large instance.

Web server config	Database config	Threshold
1 × micro	1 × micro	116
1 × micro	1 × extra-large	1160
1 × micro	2 × micro	235
1 × micro	4 × micro	380
1 × micro	10 × micro	1200

Conclusion

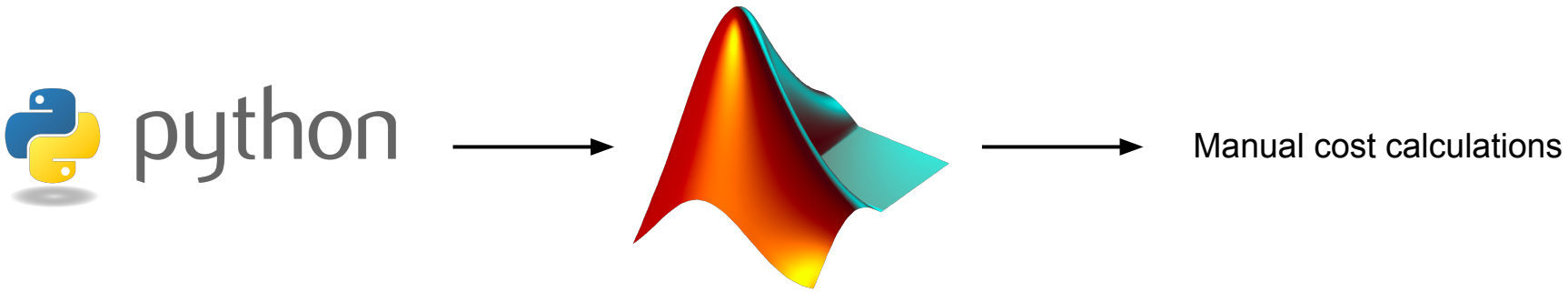
- Built a tool to receive response times for database and web server components of a cloud service.
- Fitted distributions to response time data.
- Used JMT and component analysis to build complex networks.
- Evaluated performance and costs of different EC2 instances

Improvements and Future work

- Model Content Delivery Networks (CDNs) and Load balancing components of cloud computing.
- Increase the test coverage to small, medium, and large EC2 instances (also to other cloud service providers)

Future work

- Build a tool that encompasses the service time retrieval, service time distribution fitting, performance and cost calculation.



Thank you for listening

QUESTIONS?