Comp567 Case Study Proposal

Optimal Purchasing on Amazon Elastic Compute Cloud

Consultants:

Yijia Xu, Weizhong Li, Zhe Chen, Yiwei Shi

McGill

Outline

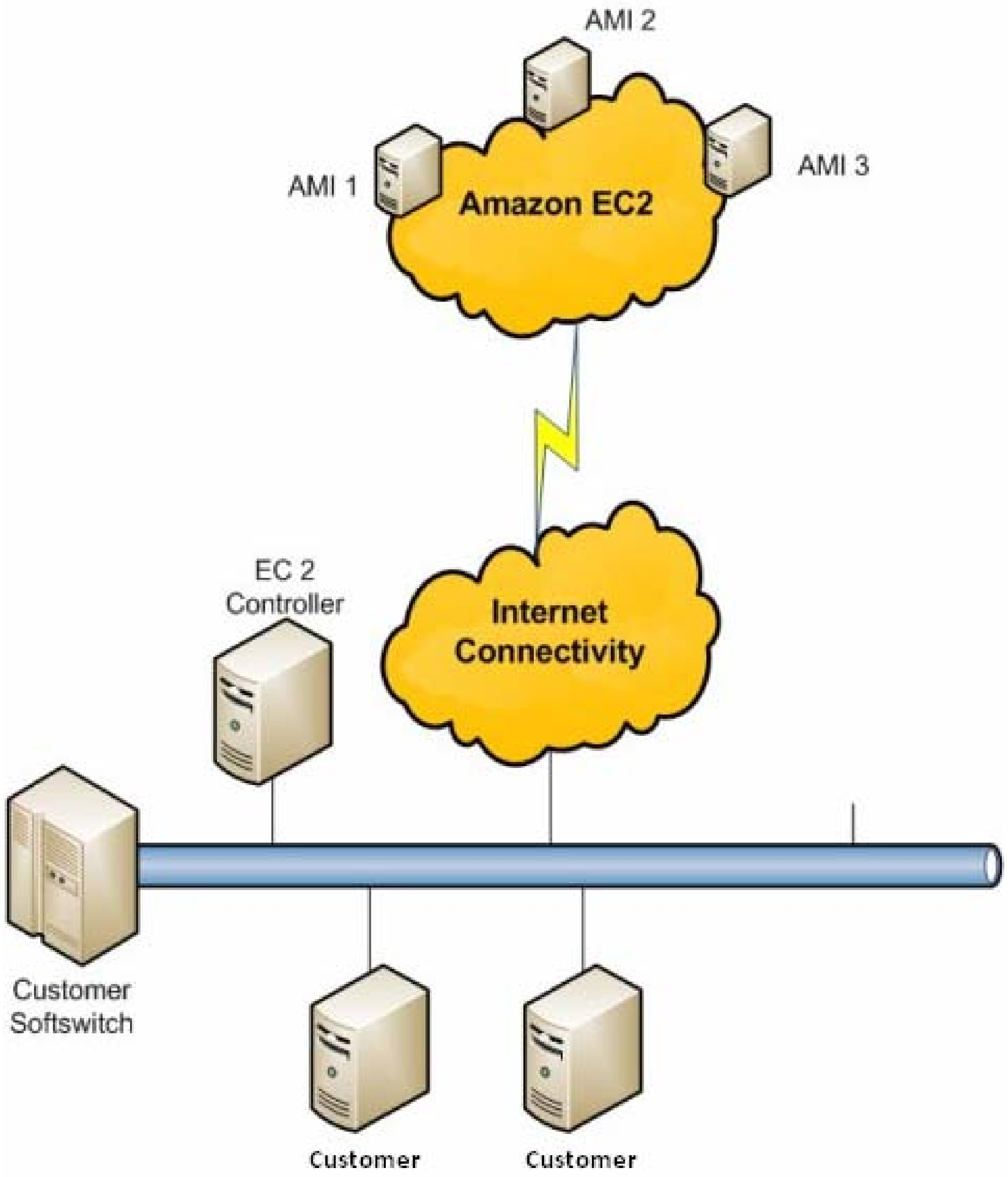
- Problem Overview
- Problem formulation
- Simplified prototype
- Extension and future work

Problem Overview

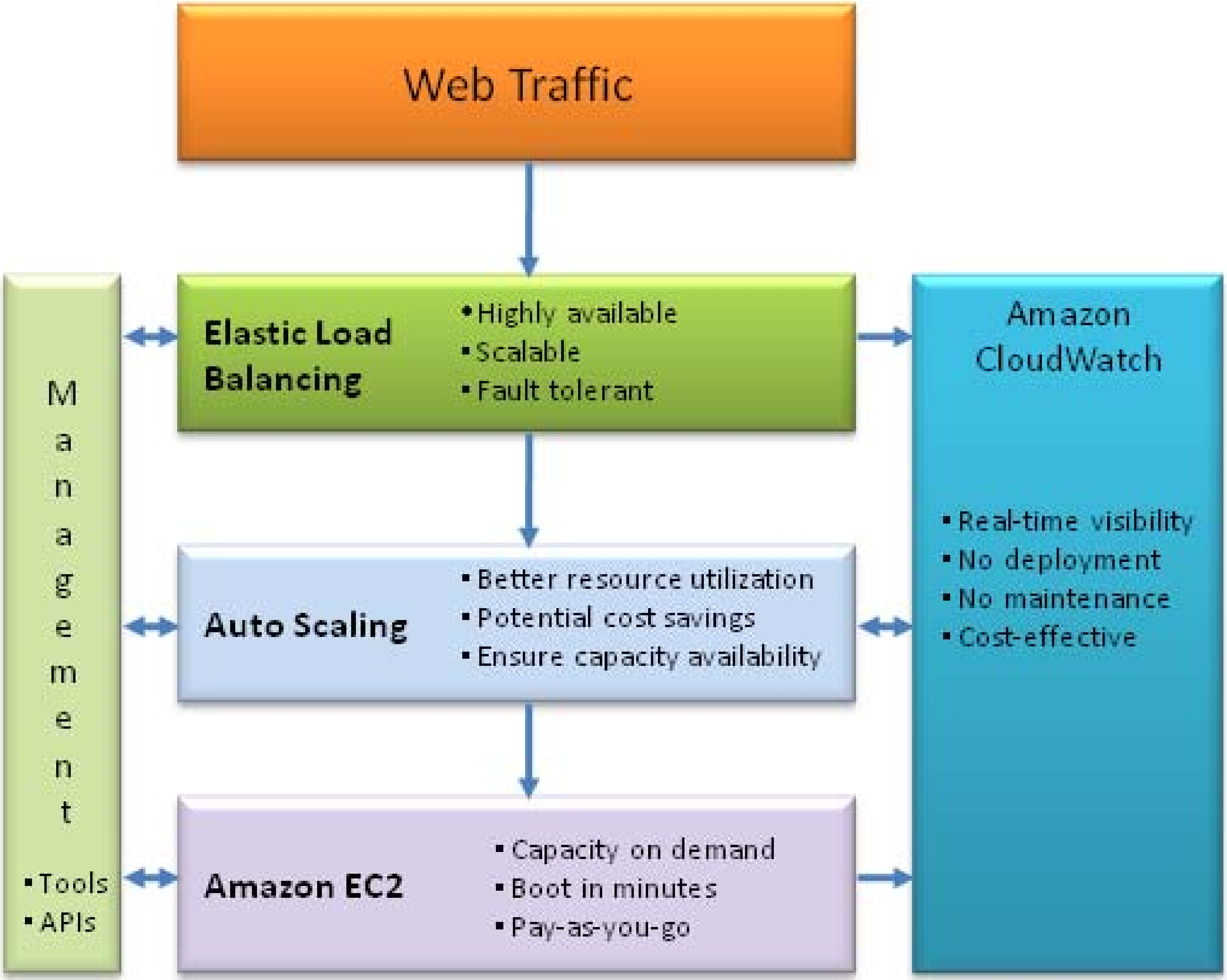
What is Amazon EC2

- A web service that provides resizable compute capacity in the cloud.
- Allows users to rent computers on which to run their own computer applications.
- Allows scalable deployment of applications by providing a web service through which a user can boot an Amazon virtual machine ---“instance”.

EC2 General View



EC2 Service Highlights



Web Traffic

Elastic Load Balancing
- Highly available
- Scalable
- Fault tolerant

Auto Scaling
- Better resource utilization
- Potential cost savings
- Ensure capacity availability

Amazon EC2
- Capacity on demand
- Boot in minutes
- Pay-as-you-go

Management
- Tools
- APIs

Amazon CloudWatch
- Real-time visibility
- No deployment
- No maintenance
- Cost-effective

EC2 Service Highlights

- create, launch, and terminate server instances as needed
- paying by hour for resources that you actually consume
- multiple pricing ways
- multiple instance types
- multiple instance locations
- multiple operating systems, and software packages

Problem Description

- We run a company which have some online business service deployed on EC2.
- The workload of this service varies from hour by hour every day and would possibly increases at a constant rate every 3 months.

How to find an optimal choice on instance types, locations, pricing ways etc. such that our expenditure in one year is minimum?

Input Data Specification

- q: Query types and data needed for each type of query. Daily changes of the number of each type of query, specified up to each period of the day. How does the traffic grow over seasons in the year.

- c(capacity): For each type of instance, how much traffic is supported in an hour.

- p(price): Price of each instance

Input Data Specification

- λ: percentage of the data happened beyond service capacity.

- FT(fault-tolerance): each site can handle at least certain portion of all the data queried.

Decision Variables

Amazon Elastic Compute Cloud pricing

- Three locations: Virginia, California, Ireland.---l

  Three instances: Small, Large, Extra Large (Linux usage)---s

  Two pricing schemes: On-Demand {x}, Reserved {y}

  Now we have 18 different instances{ {x_{sl}}, {y_{sl}} } (s=1,2,3; l=1,2,3)

- m periods each day

  n seasons a year, we consider 30 days each month

  Add up to mn different time periods for each instance---t

x_{slt} (t=1,2,3...mn) for On-Demand

y_{sl} for Reserved

Objective

$$\min \sum_{s=1}^{3} \sum_{l=1}^{3} \left(\sum_{t=1}^{mn} 90 p_{xsl} 6 x_{slt} + p_{ysl} y_{sl} \right)$$

minimize cost for the whole year

Constraints

- Each time period, we can only fail at most λ portion of queried data.
- At each location, we allocate at least FT×100% data service
- Integer solution

OBJECTIVE FUNCTION

$$\min \sum_{s=1}^{3} \sum_{l=1}^{3} (540 \cdot x_{sl} \, p_{x_{sl}} + y_{sl} \, p_{y_{sl}})$$

$$(x_{sl} = \sum_{t=1}^{16} x_{slt})$$

x_{sl}: # of ***On-Demand*** instances;

y_{sl}:# of ***Reserved*** instances;

p_{*sl}:price of instances;

CONSTRAINT FUNCTIONS

1) DATA TRANSFER CONSTRAINT

For each period t (1~16):

$$\sum_{s=1}^{3}\sum_{l=1}^{3}(x_{slt}c_{x_{sl}} + y_{sl}c_{y_{sl}}) \geq {}^{q_t}\!\big/_{(1+\lambda)}$$

$$(0.1 \geq \lambda \geq 0)$$

CONTINUE

2) FAULT-TOLERANCE CONSTRAINT

For each period t (1~16)

& a certain location

(1~3):

$$\sum_{s=1}^{3} x_{slt} + y_{sl} \geq 0.3 \cdot q_t$$

$$\sum_{s=1}^{3} x_{slt} + y_{sl} \leq 0.7 \cdot q_t$$

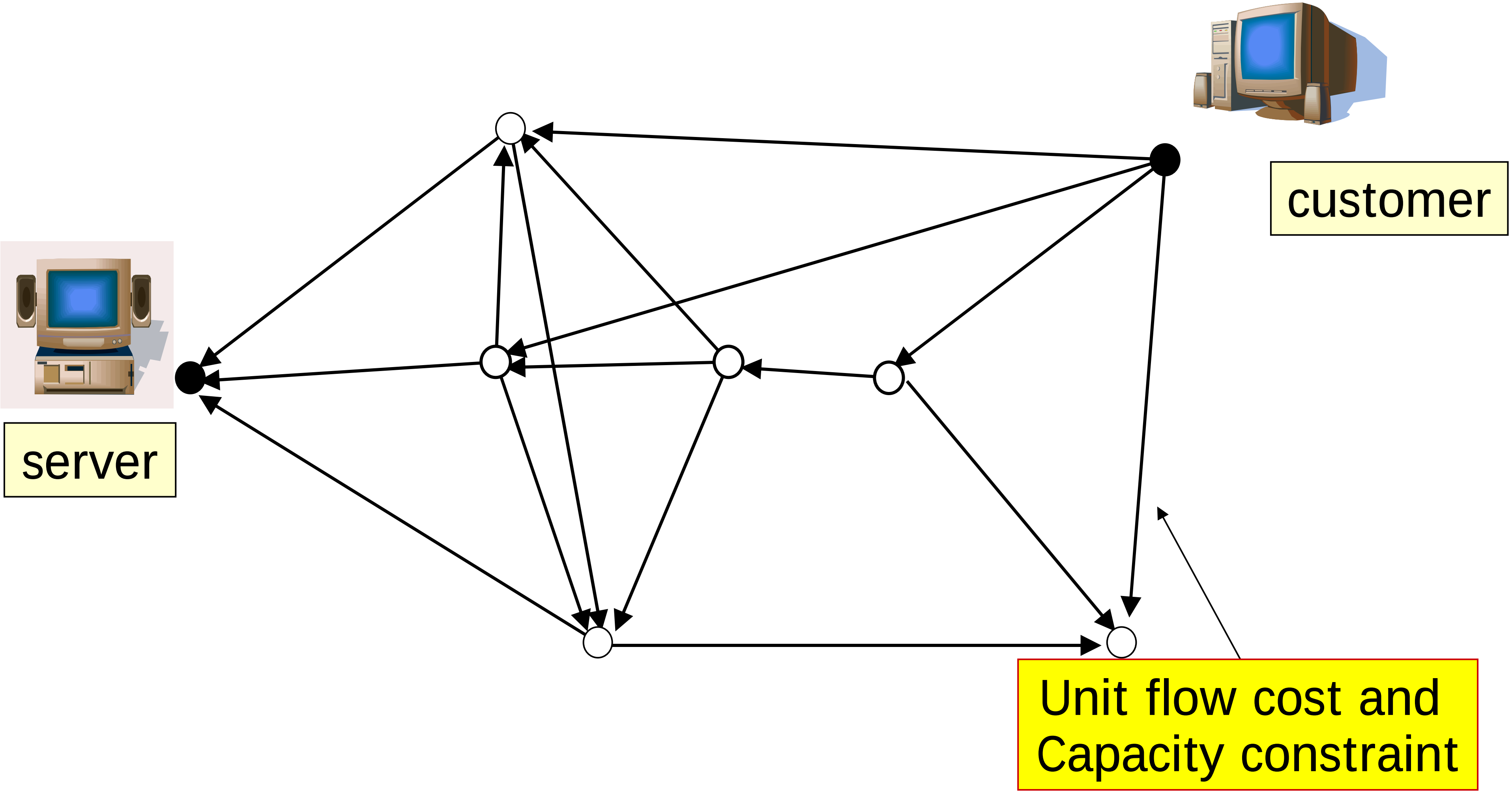
CONTINUE

3) CUSTOMER SATISFIED CONSTRAINT

$$\sum_{t=1}^{16}\sum_{s=1}^{3}\sum_{l=1}^{3}(x_{slt}c_{x_{sl}} + y_{sl}c_{y_{sl}}) \Big/ \sum_{t=1}^{16} q_t \geq S$$

S is a limit for measuring stratification

Question: minimize the cost, and try to get the extreme flow



Network transmission

Given a directed graph *G (network model)*

A source node *s*

A sink node *t*

Goal: To send as much information from *s* to *t, meanwhile try to maintain the minimal cost.*

Flows

An *s-t* flow is a function *f* on the edges which satisfies:

(capacity constraint) $0 \leq f(e) \leq c(e)$

(conservation of flows)

$$\sum_{e \in \delta^{in}(v)} f(e) = \sum_{e \in \delta^{out}(v)} f(e)$$

Value of the flow $= \sum_{e \in \delta^{out}(s)} f(e)$

Minimum Cost Flows

Goal: Build a cheap network to satisfy the flow requirement.

Input:

- A directed graph *G*
- A source vertex *s*
- A sink vertex *t*
- A capacity function *c* on the edges, i.e. *c:E->R*
- A cost function *w* on the edges, i.e. *w:E->R*
- A flow requirement *k*

Output: an *s-t* flow *f* of value k which **minimizes** *Σf(e) w(e)*