Data-driven Performance Prediction and Resource Allocation for Cloud Services

Rerngvit Yanggratoke

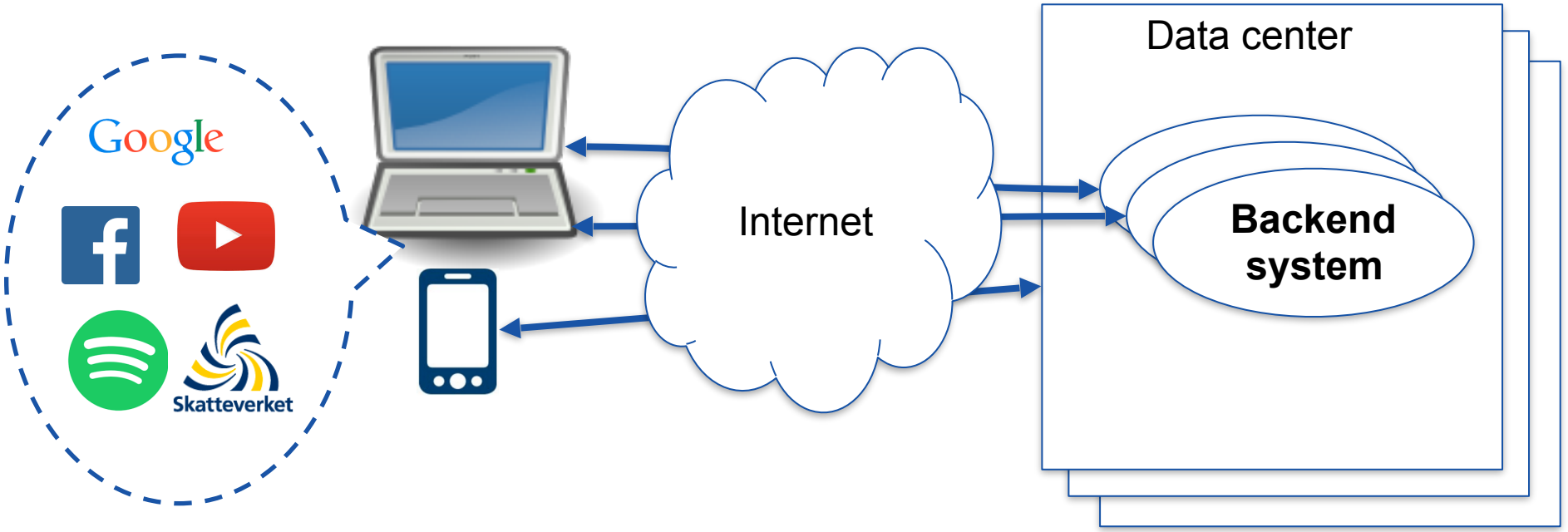
Doctoral Thesis
May 3, 2016

Advisor: Prof. Rolf Stadler
Opponent:
Prof. Filip De Turck, Ghent University, Ghent, Belgium
Grading Committee:
Prof. Raouf Boutaba, University of Waterloo, Canada
Dr. Giovanni Pacific, IBM Research, USA
Prof. Lena Wosinska, KTH Royal Institute of Technology, Sweden

Introduction

Cloud services – search, tax filing, video-on-demand, and social-network services

Performance of such services is important



This thesis focuses on performance management of backend systems in a data center

Problem and Approach

Three **fundamental problems** for performance management of backend systems in a data center

1. Resource allocation for a large-scale cloud environment
2. Performance modeling of a distributed key-value store
3. Real-time prediction of service metrics

Data-driven approach
Estimate model parameters from measurements

Outline

1. **Resource allocation for a large-scale cloud environment**
2. Performance modeling of a distributed key-value store
3. Real-time prediction of service metrics
4. Contributions and open questions

Motivation – Large-scale Cloud

"Big gets Bigger: the Rise of the Mega Data Centre"

DimensionData Research, 2014

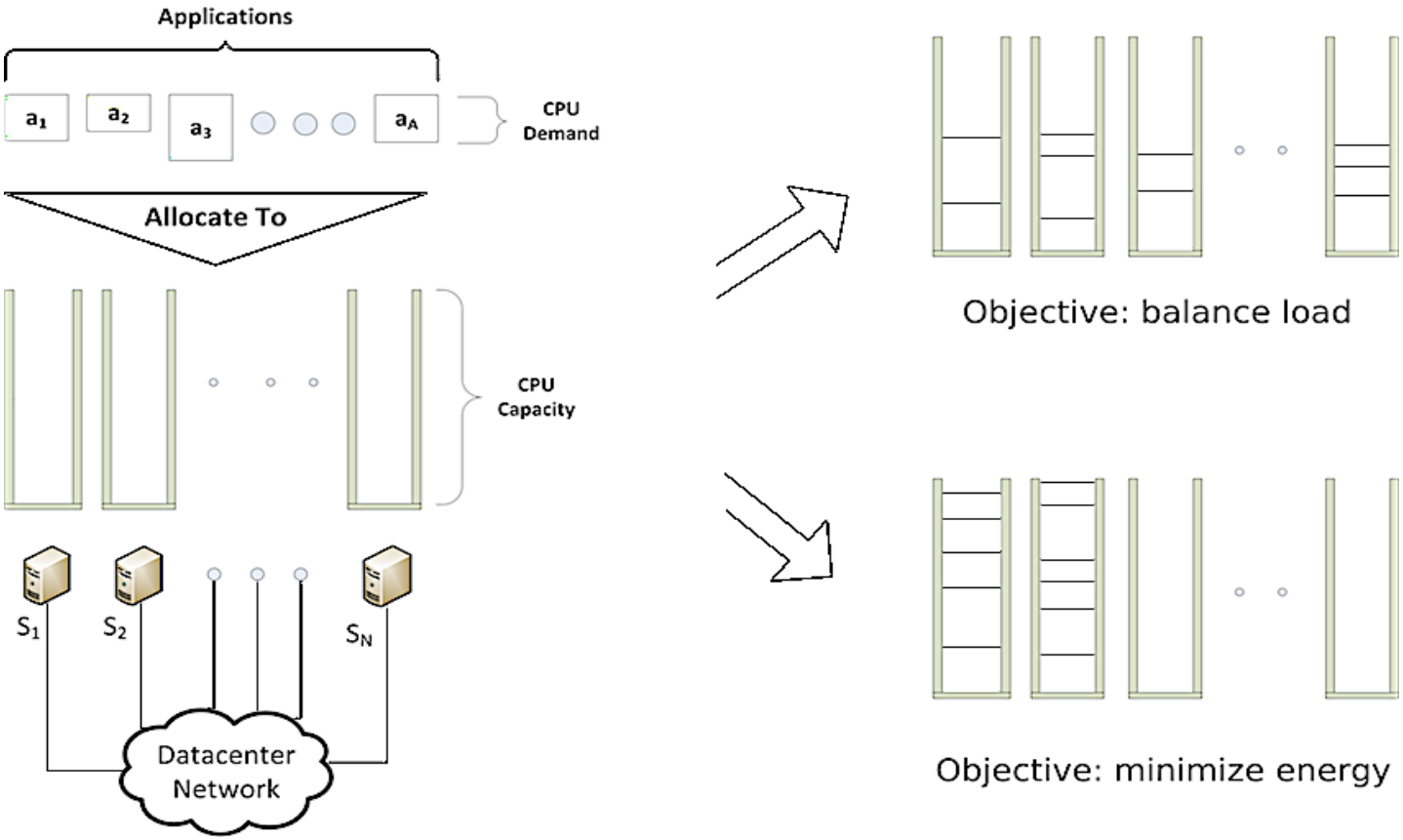
- Apple's data center in North Carolina, USA (about 500'000 feet^2)
- Expected to host 150'000+ machines

- Amazon data center in Virginia, USA (about 300'000 machines)
- Microsoft's mega data center in Dublin, Ireland (300'000 feet^2)
- Facebook planned for a massive data center in Iowa, USA (1.4M feet^2)
- eBay's data center in Utah, USA (at least 240'000 feet^2)

Resource Allocation

Select the machine to run an application that satisfies:

- Resource demands of all applications in the cloud
- Management objectives of the cloud provider



Resource allocation system computes the solution

Requirement and Approach

Resource allocation system that supports

- Joint allocation of compute and network resources
- Generic and extensible for management objectives
- Scalable operation (> 100’000 machines)
- Dynamic adaptation to changes in load patterns

Approach

- Formulate the problem as an optimization problem
- Use distributed protocols - gossip-based algorithms
- Assume the full-bisection-bandwidth network

The Objective Function f

The objective function f expresses a management objective

Balance load objective

$$f = \sum_{\theta \in \{\omega,\gamma,\lambda\}} k^{\theta} \sum_{n \in N} \theta_n^c (\theta_n^u)^2$$

Energy efficiency objective

$$f = \sum_{n \in N} p_n$$

P_n = energy of machine n

Fairness objective

$$f = -\left(\sum_{\theta \in \{\omega,\gamma\}} k^{\theta} \sum_{m \in M} \theta_m^d \sqrt{\theta_m^v} \right)$$

Service differentiation objective

$$f = -\left(\sum_{\theta \in \{\omega,\gamma\}} k^{\theta} \sum_{m \text{ of } \mathcal{B}} \theta_m^d \sqrt{\theta_m^v} \right)$$

$\theta \in \{\omega, \gamma, \lambda\} = \{CPU, Memory, Network\}$

N = set of machines, M = set of applications

The Objective Function $f(t)$

The objective function $f(t)$ expresses a management objective

Balance load objective

$$f(t) = \sum_{\theta \in \{\omega,\gamma,\lambda\}} k^{\theta} \sum_{n \in N} \theta_n^c \theta_n^u(t)^2$$

Energy efficiency objective

$$f(t) = \sum_{n \in N} p_n(t)$$

$P_n(t)$ = energy of machine n at time t

Fairness objective

$$f(t) = -\left(\sum_{\theta \in \{\omega,\gamma\}} k^{\theta} \sum_{m \in M(t)} \theta_m^d(t) \sqrt{\theta_m^v(t)} \right)$$

Service differentiation objective

$$f(t) = -\left(\sum_{\theta \in \{\omega,\gamma\}} k^{\theta} \sum_{m \text{ of } \mathcal{B}(t)} \theta_m^d(t) \sqrt{\theta_m^v(t)} \right)$$

$\theta \in \{\omega, \gamma, \lambda\} = \{CPU, Memory, Network\}$

N = set of machines, $M(t)$ = set of applications

Optimization Problem

Minimize f

Subject to

(1) $\theta_n^{net} + \sum_{m \in A_n} \theta_m^r \leq \theta_n^c \text{ for } \theta \in \{\omega, \gamma, \lambda\}, n \in N$

(2) $\theta_m^r \geq \theta_m^d \text{ for } \theta \in \{\omega, \gamma, \lambda\}$

(1) are capacity constraints
(2) are demand constraints

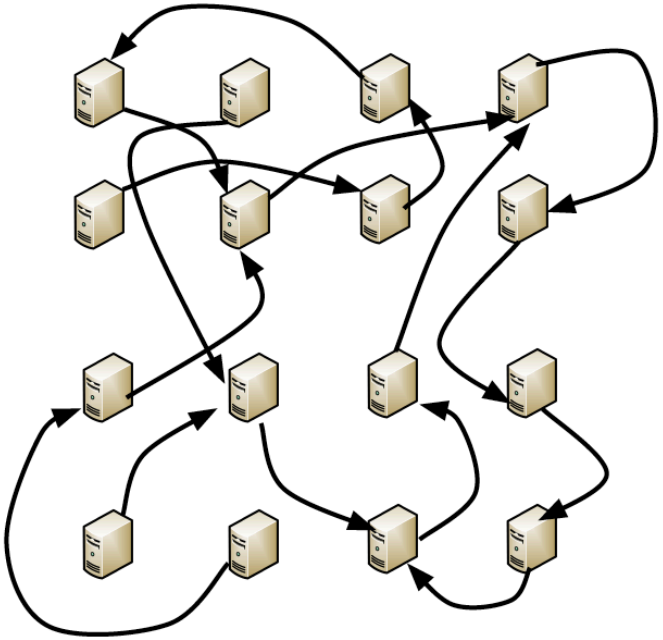
This problem is NP-hard

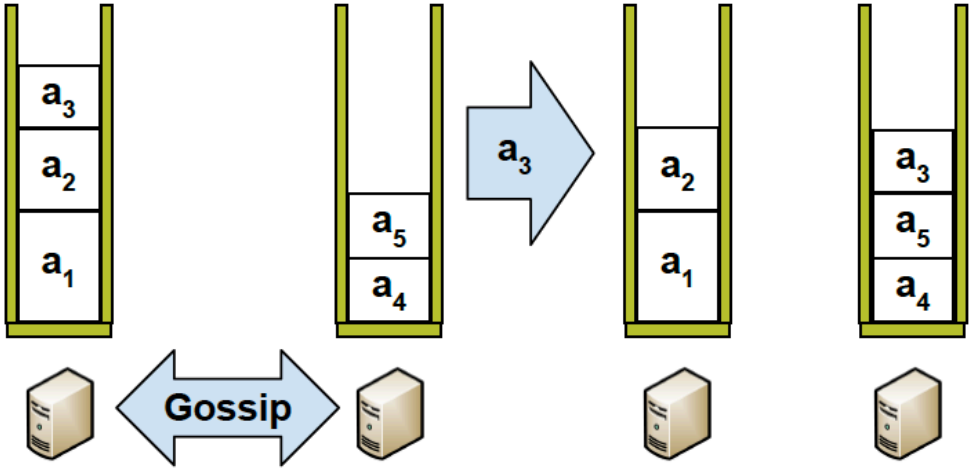
- We apply a heuristic solution
- How to distribute this computation?

Gossip Protocol

- A round-based protocol that relies on pairwise interactions to accomplish a global objective
- Each node executes the same code
- The size of the exchanged message is limited

"During a gossip interaction, move an application that minimizes $f(t)$"





Balanced load objective

Generic Resource Management Protocol (GRMP)

A generic and scalable gossip protocol for resource allocation

Algorithm 1 Protocol GRMP runs on each machine i

```
initialization
1: initInstance()
2: start passive and active threads;
```

```
passive thread
1: while true do
2:    read current state s_i
3:    s_j =receive(j); send(j, s_i)
4:    updateState(s_i, s_j)
5:    realize s_i through live-migration and
      updating resources reserved to VMs
6: end while
```

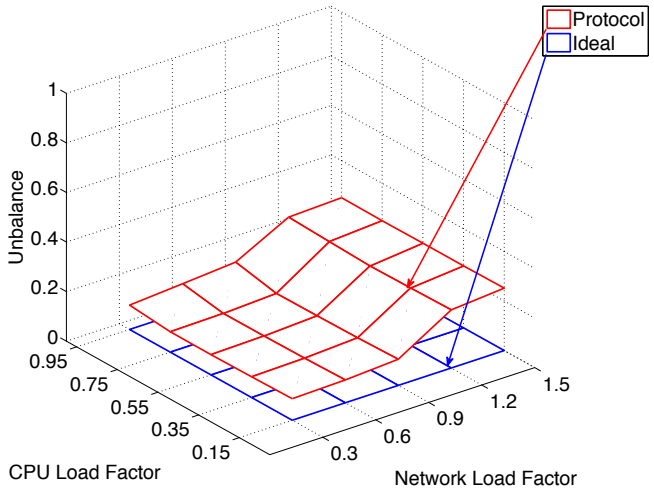
```
active thread
1: while true do
2:    read current state s_i
3:    j =choosePeer()
4:    send(j, s_i); s_j =receive(j)
5:    updateState(s_i, s_j)
6:    realize s_i through live-migration and
      updating resources reserved to VMs
7:    sleep until end of round
8: end while
```

$\texttt{updateState}(s_i, s_j)$ <- $f(t)$ is minimized locally here

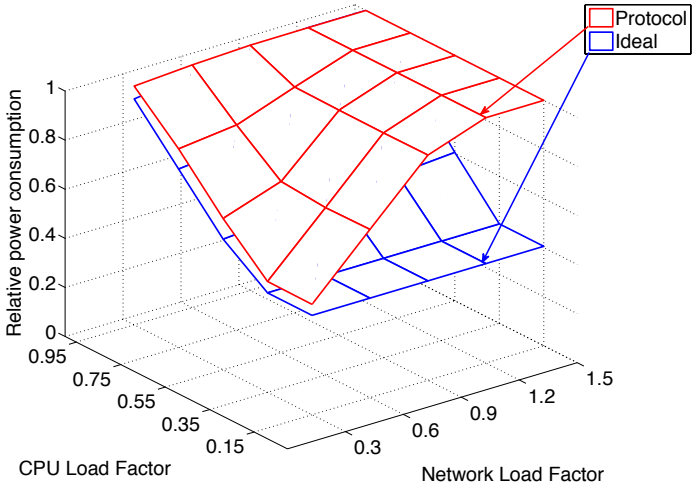
The protocol implements an iterative descent method

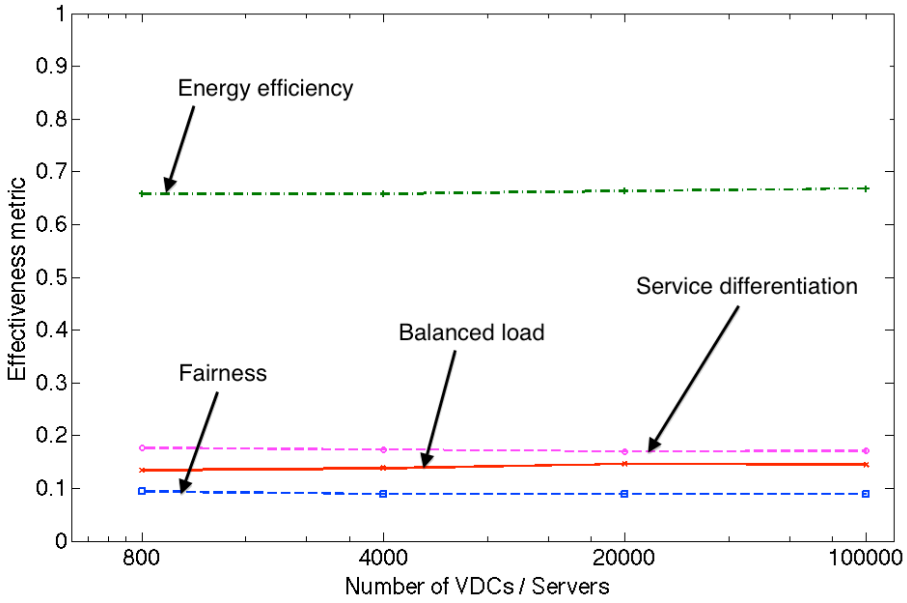
Evaluation Results from Simulation

Balanced load objective



Energy efficiency objective





Scalability

- System size up to 100’000
- Evaluation metrics do not change

Publications

R. Yanggratoke, F. Wuhib and R. Stadler, “Gossip-based resource allocation for green computing in large clouds,” In Proc. *7th International Conference on Network and Service Management (CNSM)*, Paris, France, October 24-28, 2011

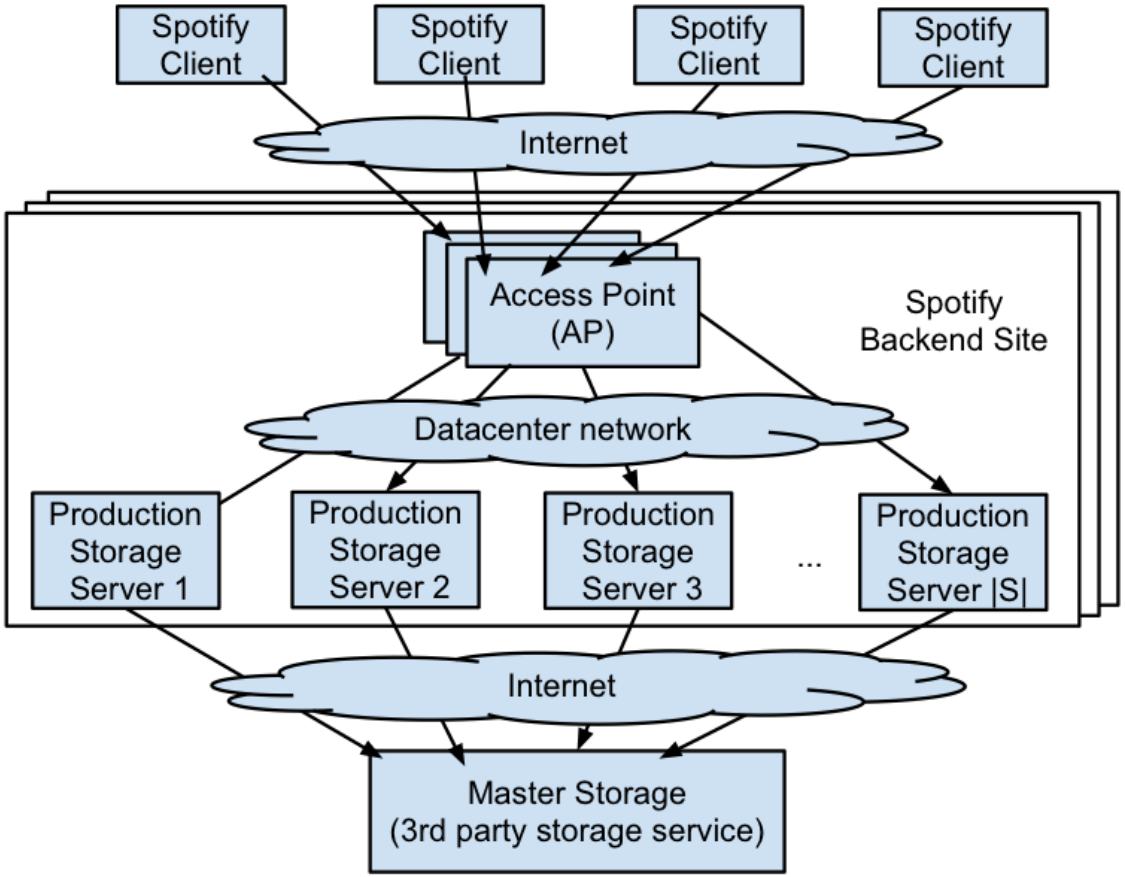
F. Wuhib, R. Yanggratoke, and R. Stadler, “Allocating Compute and Network Resources under Management Objectives in Large-Scale Clouds,” *Journal of Network and Systems Management (JNSM)*, Vol. 23, No. 1, pp. 111-136, January 2015

Outline

1. Resource allocation for a large-scale cloud environment
2. **Performance modeling of a distributed key-value store**
3. Real-time prediction of service metrics
4. Contributions and open questions

Spotify Backend for Music Streaming

A distributed key-value store



Low latency is key to the Spotify service

Problem and Approach

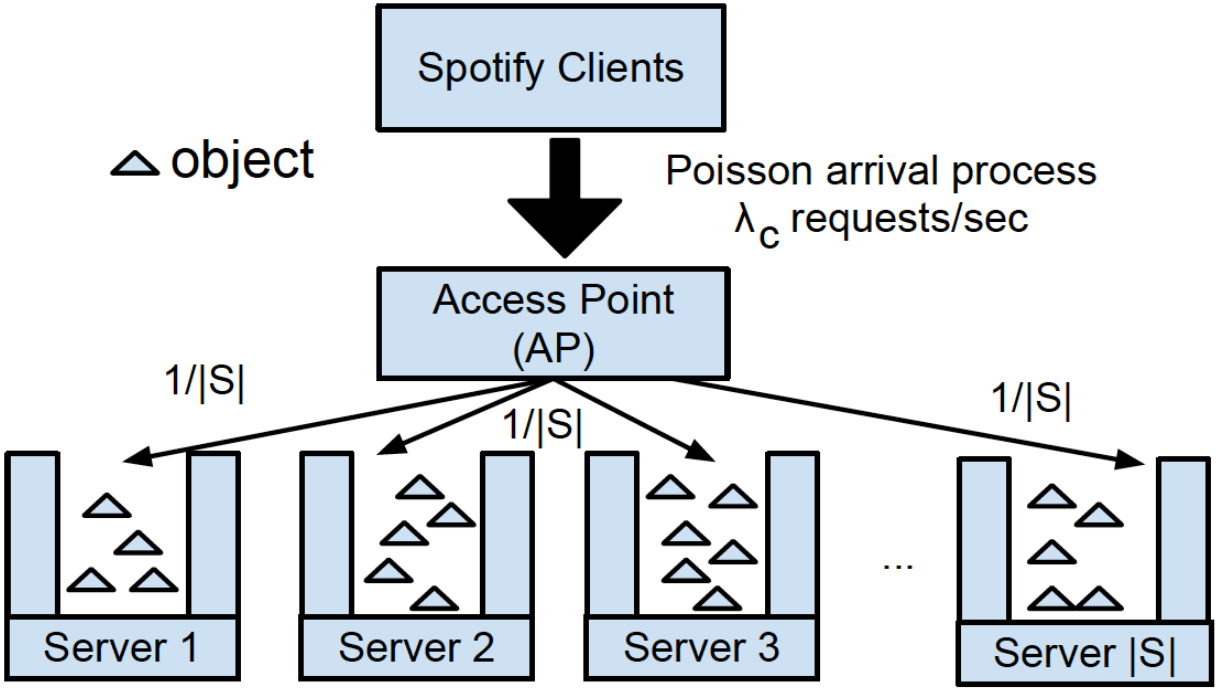
Development of performance models
for a Spotify backend site

1. Predicting response time distribution
2. Estimating capacity under different object allocation policies
   - Random policy
   - Popularity-aware policy

Approach

- Simplified architecture
- Probabilistic and stochastic modeling techniques

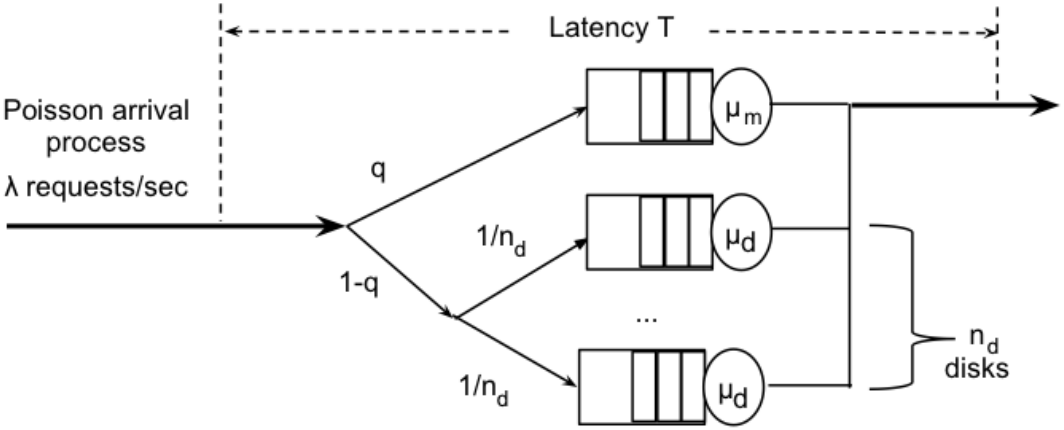
Simplified Architecture for a Spotify backend site



- Model only Production Storage
- AP selects a storage server uniformly at random
- Ignore network and access-point processing delays
- Consider steady-state conditions and Poisson arrivals

Model for Response Time Distribution

Model for a single storage server



Probability that a request to a server is served below a latency t is

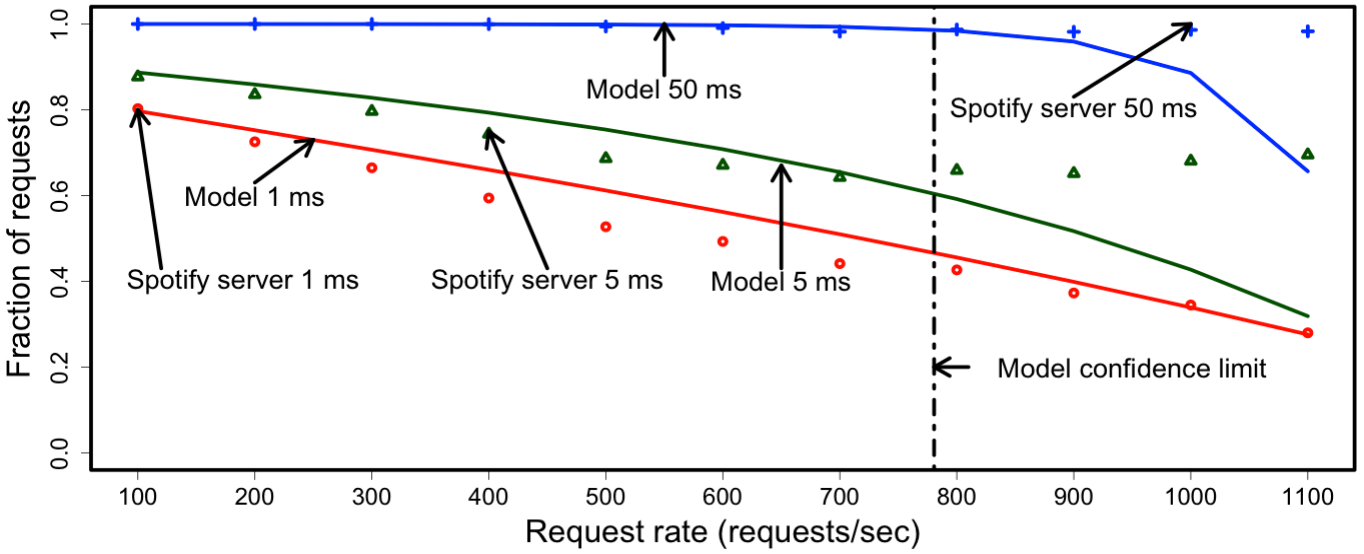
$$Pr(T \leq t) = q + (1-q)(1 - e^{-\mu_d(1-(1-q)\lambda/\mu_d n_d)t})$$

Model for a cluster of storage servers

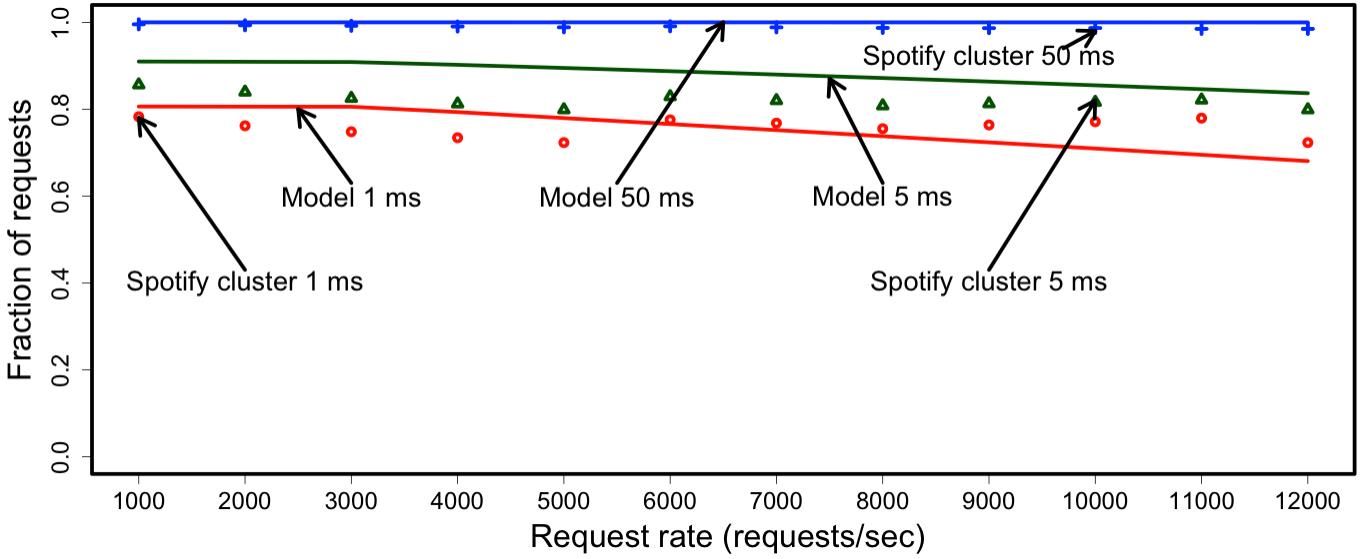
Probability that a request to the cluster is served below a latency t is

$$Pr(T_c \leq t) = \frac{1}{|S|} \sum_{s \in S} f(t, n_{d,s}, \mu_{d,s}, \frac{\lambda_c}{|S|}, q_s)$$

Model Predictions vs. Measurements from Spotify Backend



Spotify Server



Spotify Cluster

Model for Estimating Capacity under Different Object Allocation Policies

Capacity

Ω : Max request rate to a server so that a QoS is satisfied

Ω_c: Max request rate to the cluster so that the request rate to each server is at most Ω

Capacity of a cluster under the popularity-aware policy

$$\Omega_c(popularity) = \Omega \cdot |S|$$

Capacity of a cluster under the random policy

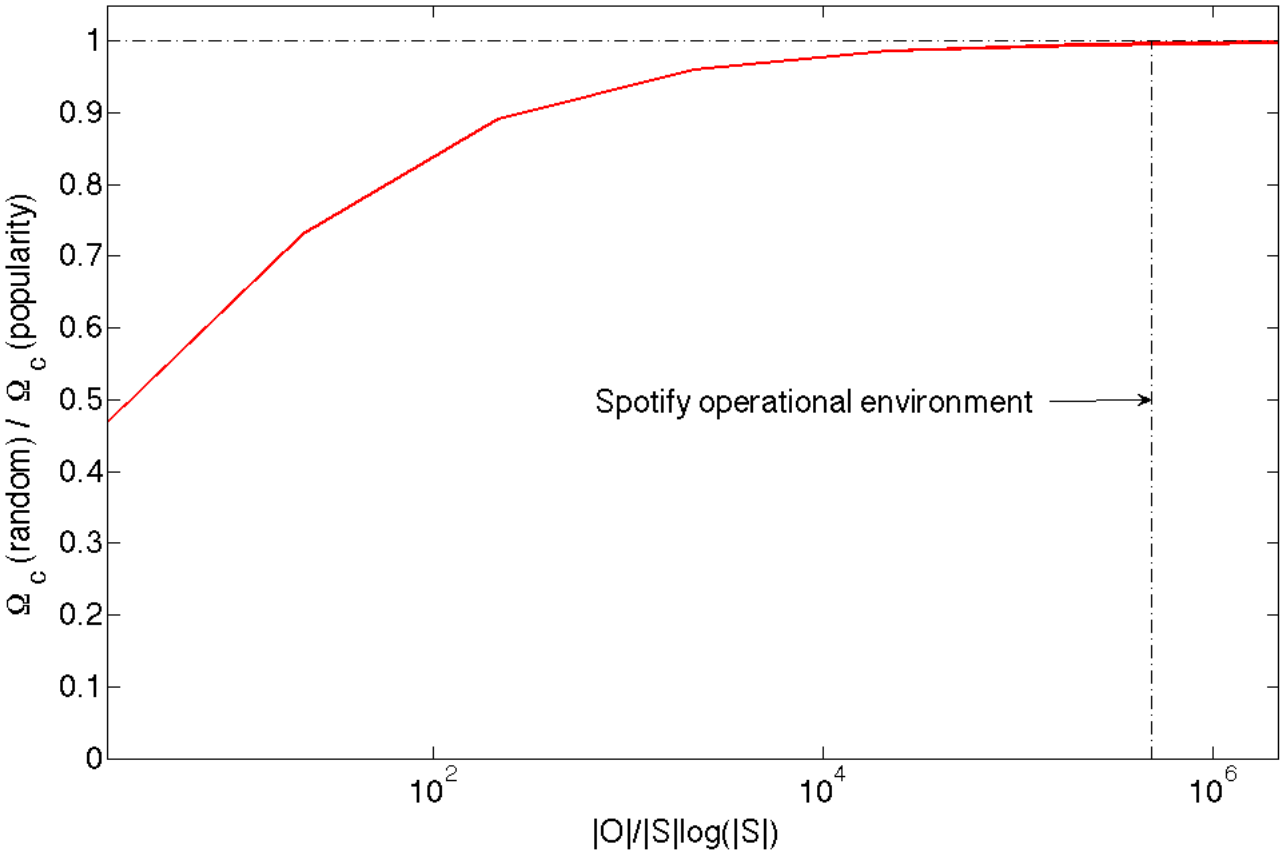
$$\Omega_c(random) \approx \Omega \cdot \frac{|O|^{1/\alpha} + \frac{|O|\alpha}{\alpha-1}}{|O_m|^{1/\alpha} + \frac{|O_m|\alpha}{\alpha-1}}$$

When $|O_m| \approx \frac{|O|}{|S|} + \sqrt{\frac{2|O|\log|S|}{|S|}}$ assuming $|O| \gg |S|(\log|S|)$

Model Predictions vs. Measurements from Testbed

Number of objects	Random policy			Popularity-aware policy		
	Measurement	Model	Error (%)	Measurement	Model	Error (%)
1250	166.50	176.21	5.83	190.10	200	5.21
1750	180.30	180.92	0.35	191.00	200	4.71
2500	189.70	182.30	3.90	190.80	200	4.82
5000	188.40	186.92	0.78	191.00	200	4.71
10000	188.60	190.39	0.95	192.40	200	3.95

Cluster Capacity for the Random Policy vs. Popularity-Aware policy



Spotify backend does not need the popularity-aware policy

Publications

R. Yanggratoke, G. Kreitz, M. Goldmann and R. Stadler, “Predicting response times for the Spotify backend,” In Proc. 8th *International conference on Network and service management (CNSM)*, Las Vegas, NV, USA, October 22-26, 2012

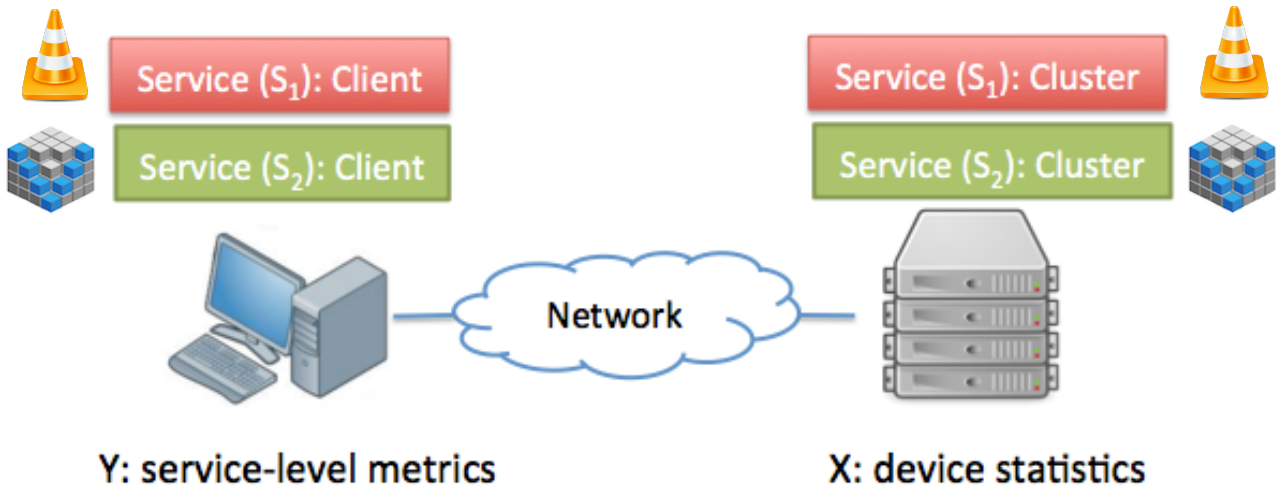
Best Paper Award

R. Yanggratoke, G. Kreitz, M. Goldmann, R. Stadler and V. Fodor, “On the performance of the Spotify backend,” accepted to *Journal of Network and Systems Management (JNSM)*, Vol. 23, No. 1, pp. 111-136, January 2015

Outline

1. Resource allocation for a large-scale cloud environment
2. Performance modeling of a distributed key-value store
3. **Real-time prediction of service metrics**
4. Contributions and open questions

Real-time Prediction Problem



Video-on-demand (VOD):
video frame rate, audio buffer rate

Key-value store (KV):
response time

- X : CPU load, memory load, #network active sockets, #processes, etc..

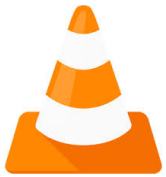
Problem $M : X \rightarrow \hat{Y}$ predicts Y in real time

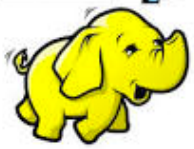
Motivation :
Key building block for real-time service assurance system

Service-agnostic Approach

Existing works

- Apply analytical models to model the service
- Statistical learning on engineered service-specific features



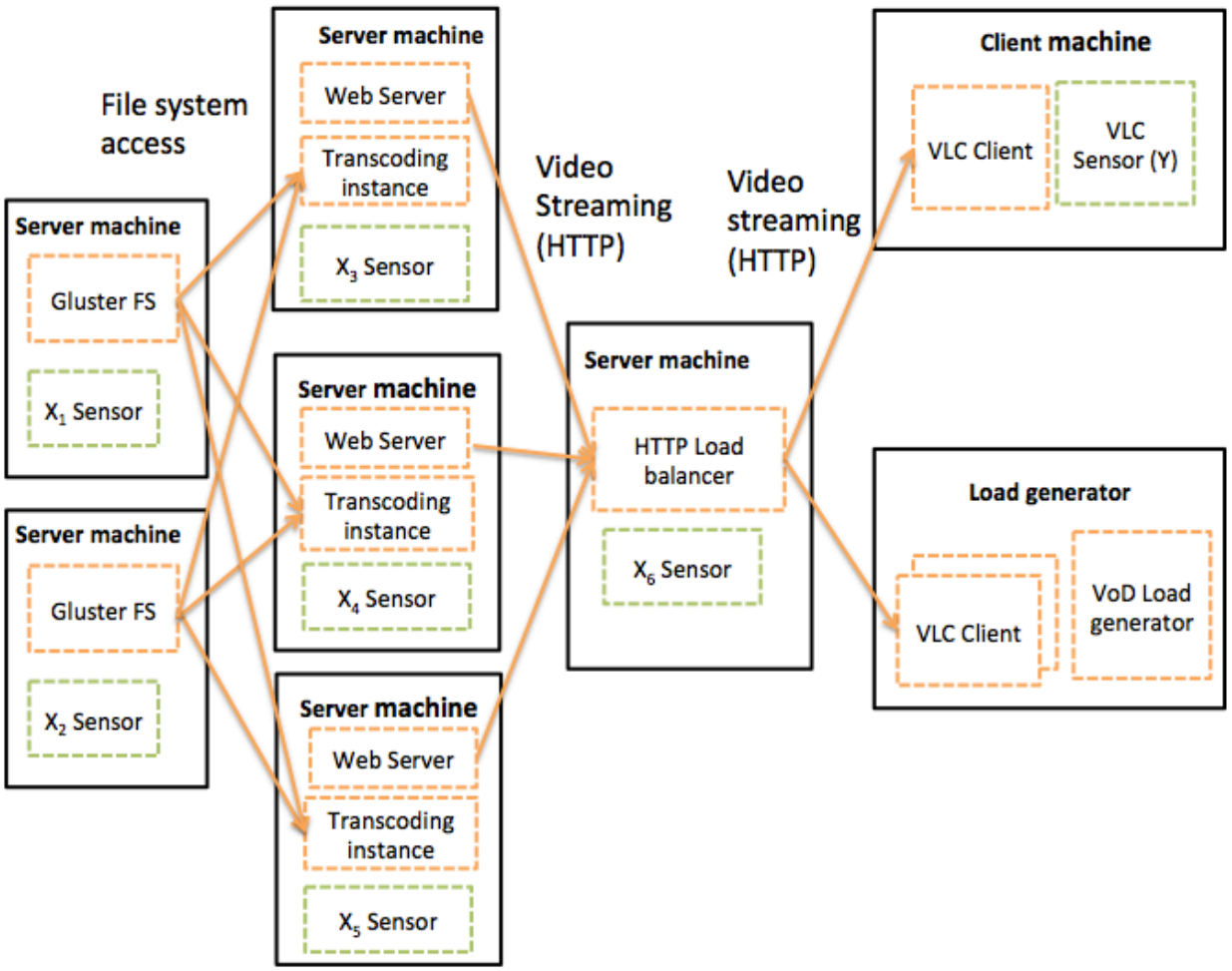




Design goal ➔ Service-agnostic prediction

Our approach

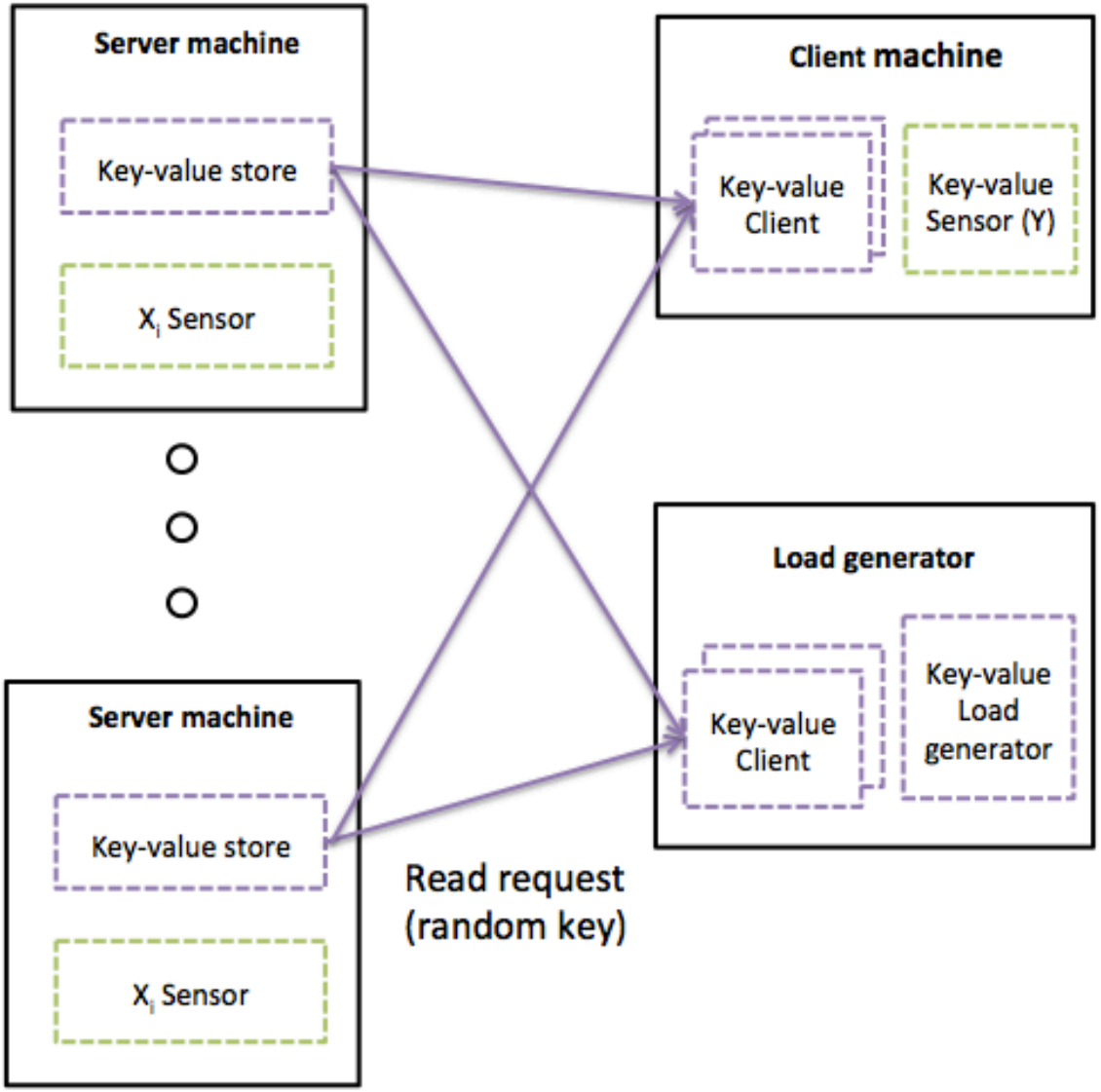
- Take “all” available statistics (> 4000 features)
- Learn using low-level (OS-level) metrics

Testbed – Video Streaming



Dell PowerEdge R715 2U rack servers, 64 GB RAM, two 12-core AMD Opteron processors, a 500 GB hard disk, and 1 Gb network controller

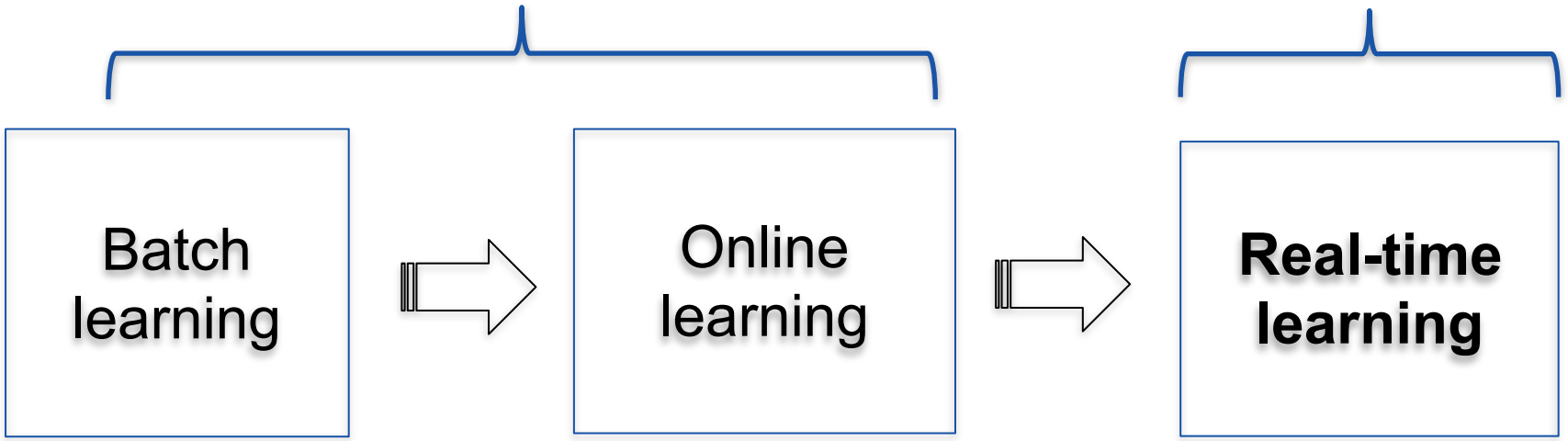
Testbed – Key-value Store



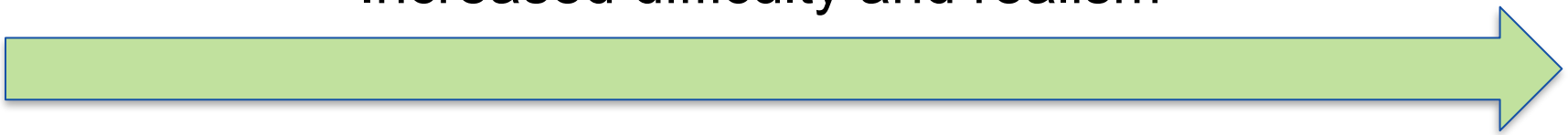
Prediction Methods

Using traces

Using live statistics



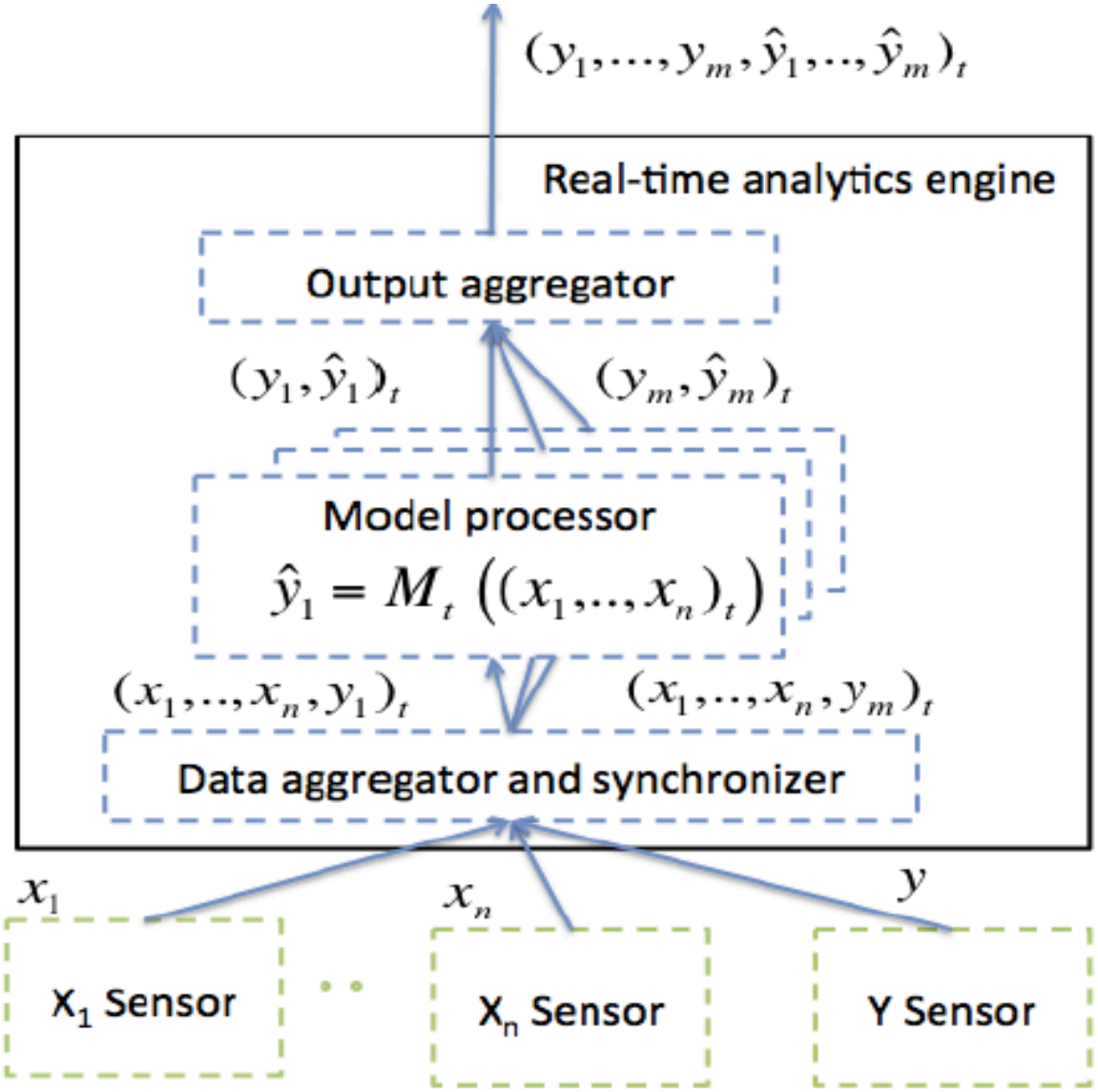
Increased difficulty and realism



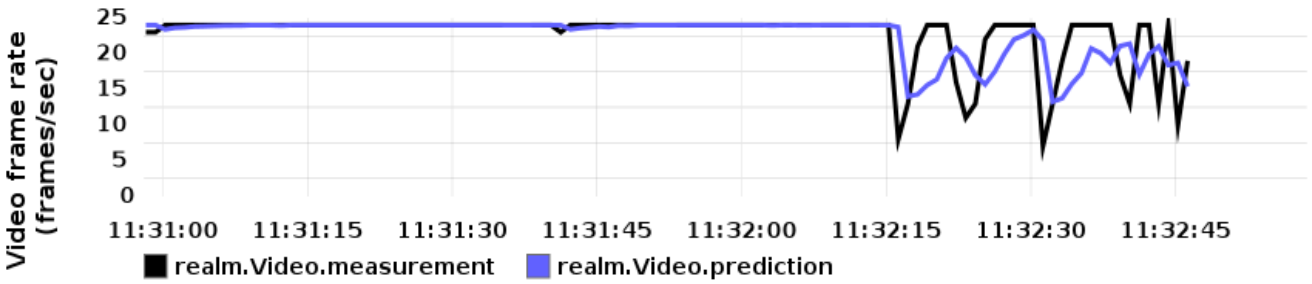
Evaluation metric: Normalized mean absolute error (NMAE)

$$NMAE = \frac{1}{\bar{y}}\left(\frac{1}{m}\sum_{i=1}^{m}\left|y_i - \hat{y}_i\right|\right)$$

Real-time Analytics Engine

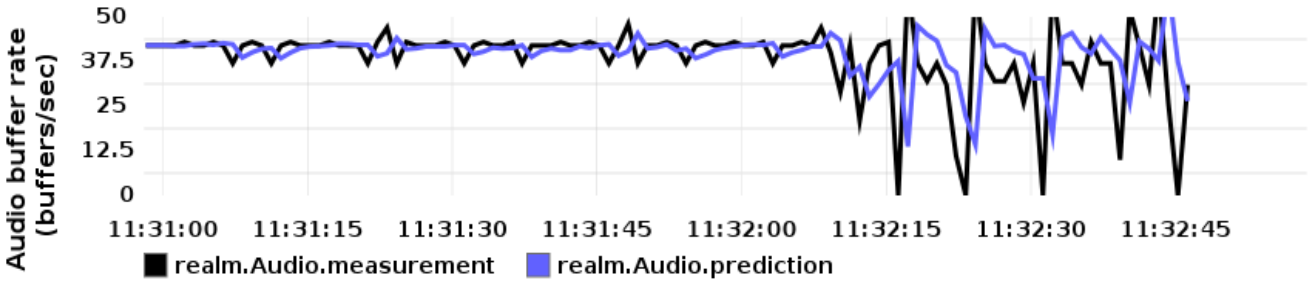


Real-time Analytics Demonstrator



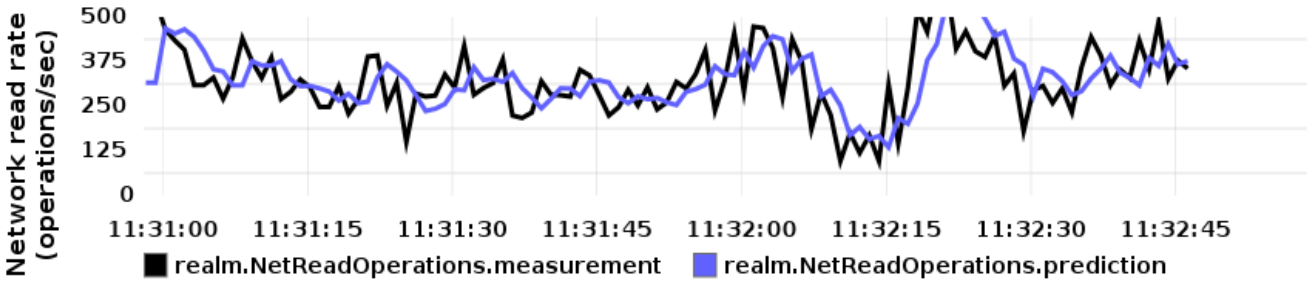
Normalized Mean Absolute Error (last 5 minutes)

2.95 %



Normalized Mean Absolute Error (last 5 minutes)

10.46 %



Normalized Mean Absolute Error (last 5 minutes)

22.31 %

Evaluation for Real-time Learning

Load pattern	Video-on-demand (VoD)		Key-value store (KV)
	Video frame rate	Audio buffer rate	Response time
Periodic-load	3.6%	14%	7%
Flashcrowd-load	5.6%	11%	6%
Periodic-load (VoD) + Flashcrowd-load (KV)	8%	29%	11%

Publications

R. Yanggratoke, J. Ahmed, J. Ardelius, C. Flinta, A. Johnsson, D. Gillblad, R. Stadler, “Predicting real-time service-level metrics from device statistics,”, *IM 2015*

R. Yanggratoke, J. Ahmed, J. Ardelius, C. Flinta, A. Johnsson, D. Gillblad, R. Stadler, “A platform for predicting real-time service-level metrics from device statistics,” *IM 2015, demo session*

R. Yanggratoke, J. Ahmed, J. Ardelius, C. Flinta, A. Johnsson, D. Gillblad, R. Stadler, “Predicting service metrics for cluster-based services using real-time analytics,” *CNSM 2015*

R. Yanggratoke, J. Ahmed, J. Ardelius, C. Flinta, A. Johnsson, D. Gillblad, R. Stadler, “A service-agnostic method for predicting service metrics in real-time,” submitted to *JNSM*

J. Ahmed, A. Johnsson, R. Yanggratoke, J. Ardelius, C. Flinta, R. Stadler, “Predicting SLA Conformance for Cluster-Based Services Using Distributed Analytics,” *NOMS 2016*

Outline

1. Resource allocation for a large-scale cloud environment
2. Performance modeling of a distributed key-value store
3. Real-time prediction of service metrics
4. **Contributions and open questions**

Key Contributions of the Thesis

We designed, developed, and evaluated a
generic protocol for resource allocation that

- supports joint allocation of compute and network resources
- enables scalable operation (> 100’000 machines)
- supports dynamic adaptation to changes in load patterns

We designed, developed, and evaluated performance ***models for response time distribution and capacity*** of a distributed key-value store that is

- Simple yet accurate for Spotify’s operational range
- Obtainable and efficient

We designed, developed, and evaluated a
solution for predicting service metrics in real-time that

- Service-agnostic
- Accurate and efficient

Open Questions for Future Research

Resource allocation for a large-scale cloud environment

- *Centralized vs. decentralized resource allocation*
- *Multiple data centers and telecom clouds*

Performance modeling of a distributed key-value store

- *Black-box models for performance predictions*
- *Online performance management using analytical models*

Analytics-based prediction of service metrics

- *Prediction in large systems*
- *Analytics-based performance management*
- *Forecasting of service metrics*
- *Prediction of end-to-end service metrics*

PhD Defense