Contributions to Performance Modeling and Management of Data Centers

Rerngvit Yanggratoke

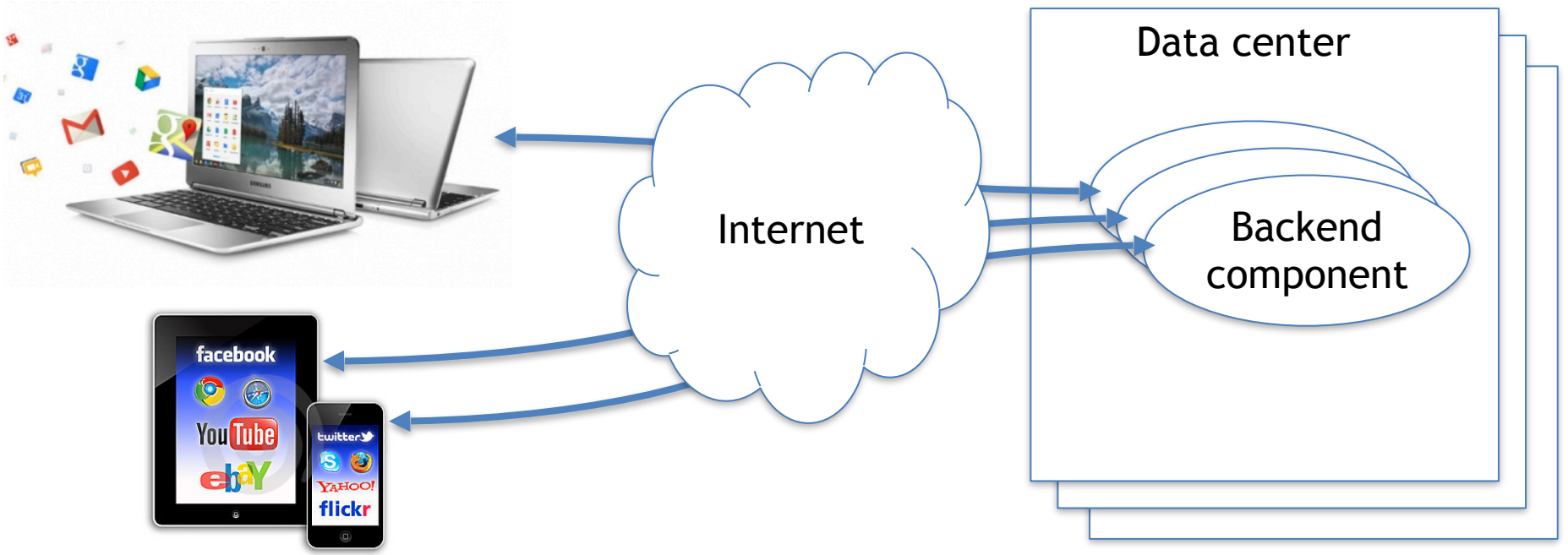
Advisor: Prof. Rolf Stadler
Opponent: Prof. Filip De Turck

Licentiate Seminar
October 25, 2013

Introduction

Networked services – search and social-network services

Performance of such services is important



Performance management of backend components in a data cen

1. Resource allocation for a large-scale cloud environment
2. Performance modeling of a distributed key-value store

Outline

1. **Resource allocation for a large-scale cloud environment**
   - Motivation
   - Problem
   - Approach
   - Evaluation results
   - Related research / publications
2. Performance modeling of a distributed key-value store
   - Motivation
   - Problem
   - Approach
   - Evaluation results
   - Related research / publications

– Contributions

– Open questions

Motivation – Large-scale Cloud

"U.S. Data centers Growing in Size But Declining in Numbe

IDC Research, 2012.

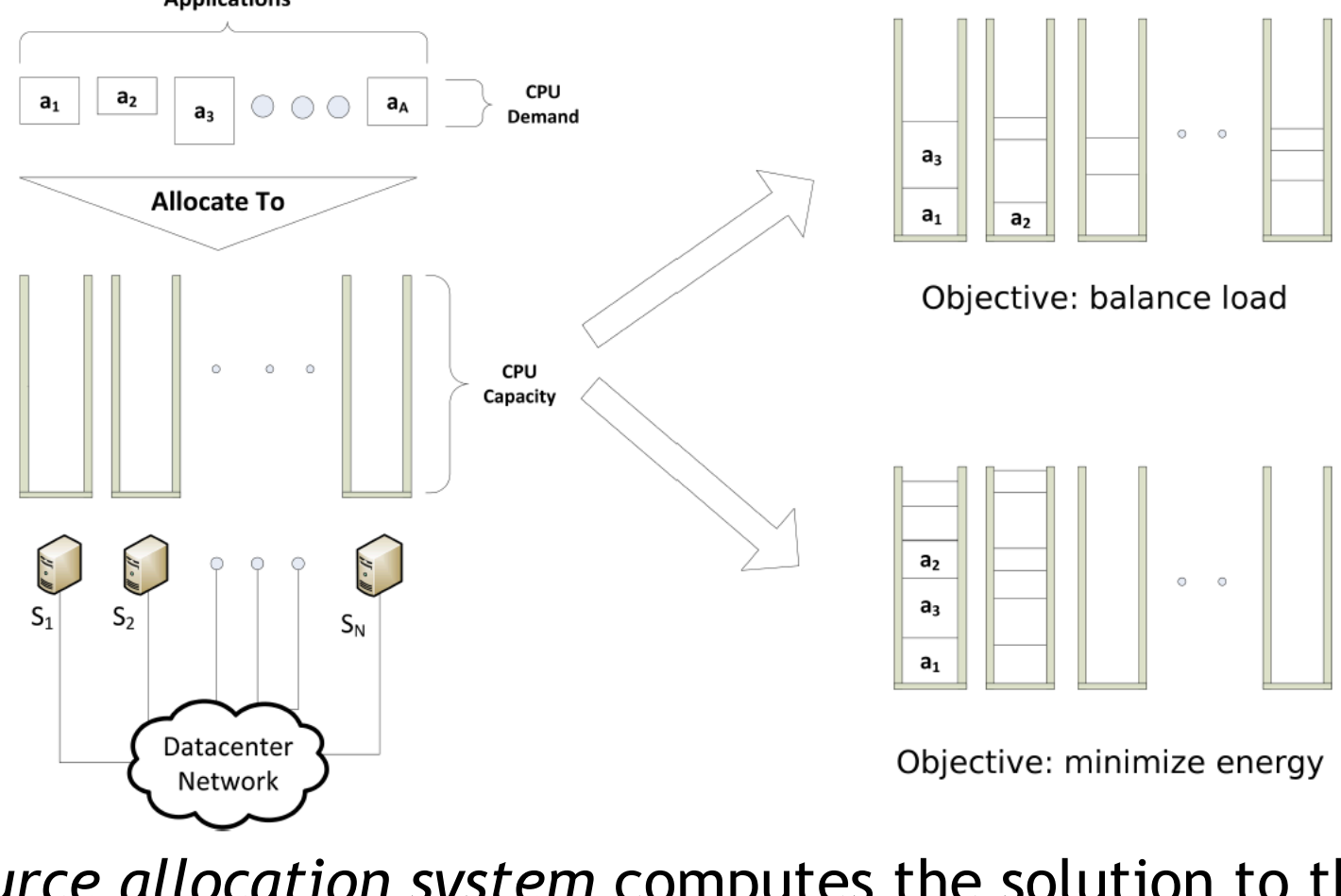
- Apple's data center in North Carolina, USA (500'000 feet^2)
- Expected to host 150'000+ machines

- 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 for executing an application that satisfies:

- Resource demands of all applications in the cloud



Resource allocation system computes the solution to the problem

Problem & Approach

Resource allocation system that supports

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

Approach

- We assume the **full-bisection-bandwidth network**
- Formulate the problem as an **optimization problem**
- Distributed protocols - **gossip-based algorithms**

An Objective Function f(t)

An objective function f(t) expresses a management objective

"Smaller f(t) moves the system closer to achieve the objective"

Balance load objective

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

Energy efficiency objective

$$f(t) = \sum p_n(t)$$

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

Fairness objective

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

Service differentiation objective

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

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

N = set of machines, M = set of

Optimization Problem

Minimize f(t)

Subject to:

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

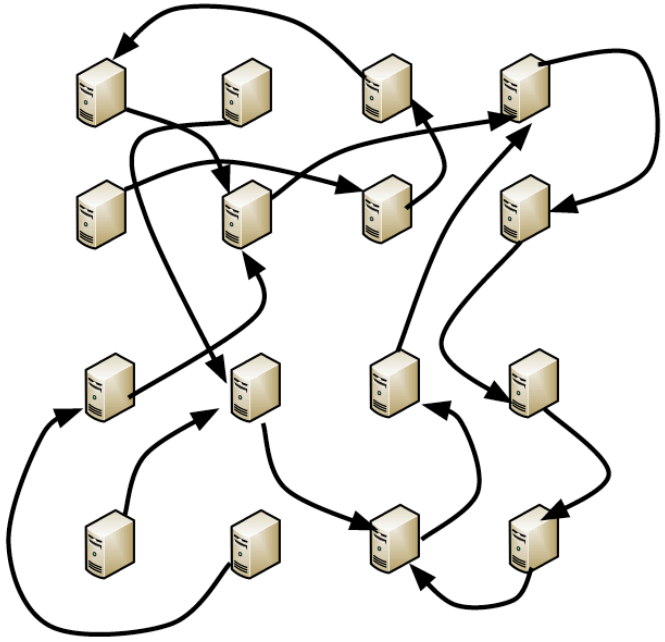
$$(2)\ \theta_m^r(t) \ge \theta_m^d \ \ for\ \theta \in \{\omega, \gamma, \lambda\}\ and\ m \in M$$

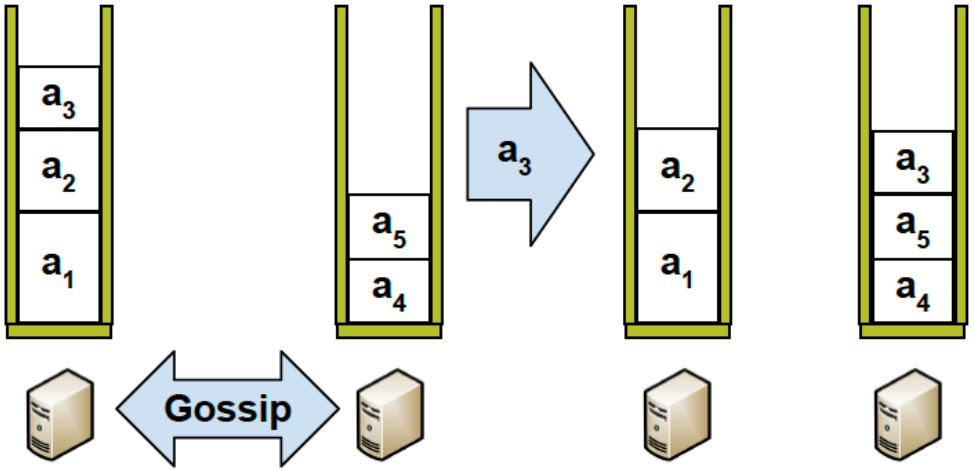
- (1) are capacity constraints
- (2) are demand constraints
- Choose a specific f(t) states the problem for a specific management objective
- This problem is NP-hard
  - We apply a heuristic solution
  - **How to distribute the computation?**

Gossip Protocol

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

"During a gossip interaction, move an application that minimizes





Balanced load objective

Generic Resource Management Protocol (GRMP)

A generic and scalable gossip protocol for resource allocation

Algorithm 3 Protocol GRMP runs on each machine i

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

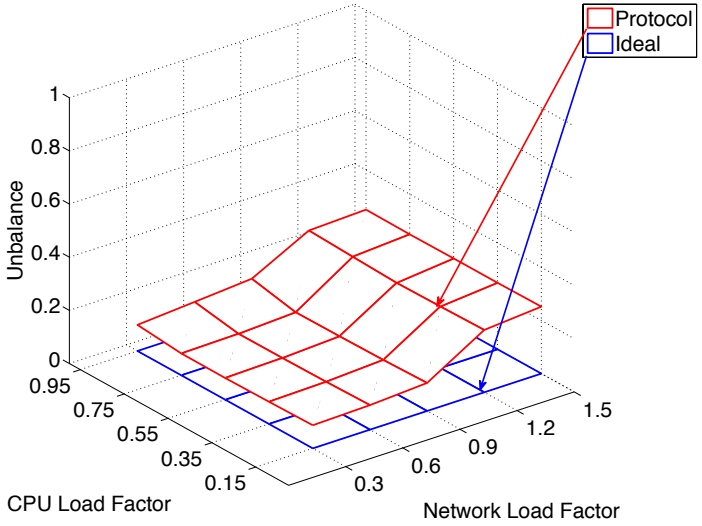
```
passive thread
 1: while true do
 2:    read current state s_i
 3:    s_j =receive(j); send(j, s_i)
 4:    updateState(s_i, s_j)
 5:    realize s_i through
       live-migration and updating
       resources reserved to VMs
```

```
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
```

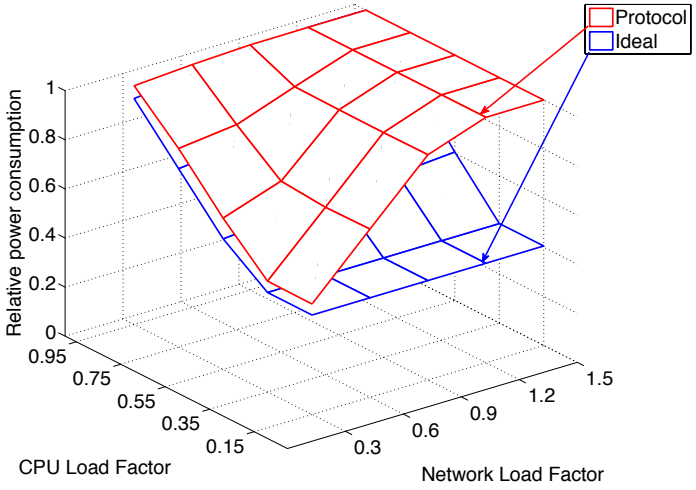
$\mathbf{updateState}(s_i, s_j)$ <- f(t) is applied here

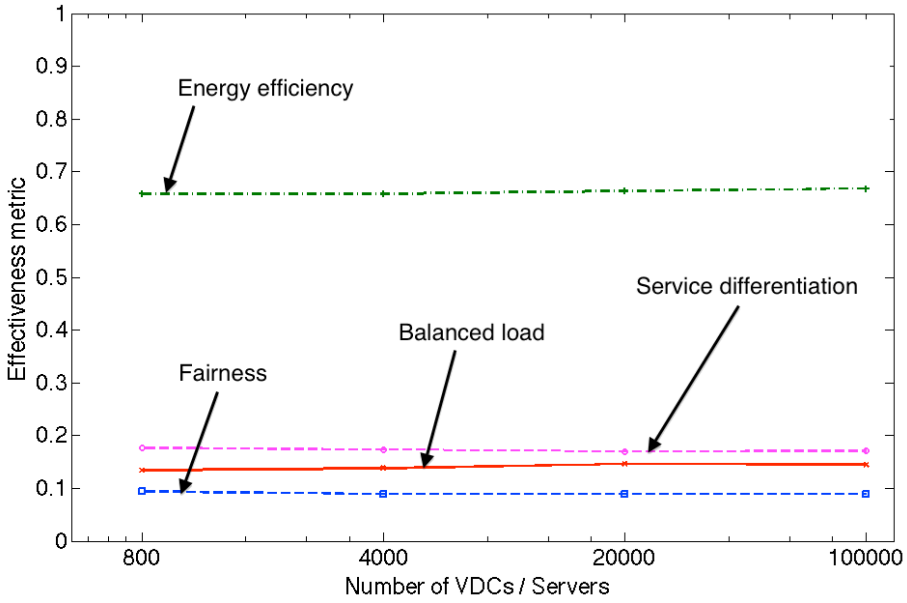
Evaluation Results

Balanced load objective



Energy efficiency objective





Scalability:

- System size up to 100’000
- Other parameters do not change

Related Research

- Centralized resource allocation system
  - (D. Gmach et al., 2008), (A. Verma et al., 2009), (C. Guo et al., 2010), (H. Hitesh et al., 2011), (V. Shrivastava et al., 2011), (D. Breitgand and A.Epstein, 2012), (J.W. Jiang et al., 2012), (O. Biran et al. 2012), …
- Distributed resource allocation system
  - (G. Jung et al., 2010), (Yazir et al., 2010), (D. Jayasinghe et al., 2011), (E. Feller et al., 2012), **this thesis**
- Initial placement
  - (A. Verma et al., 2009), (M. Wang et al., 2011), (D. Jayasinghe et al., 2011), (X. Zhang et al., 2012)
- Dynamic placement
  - (F. Wuhib et al., 2012), (D. Gmach et al., 2008), (Yazir et al., 2010), (J. Xu and Jose Fortes, 2011) , **this thesis**

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)*, pages 126, 2013.

Outline

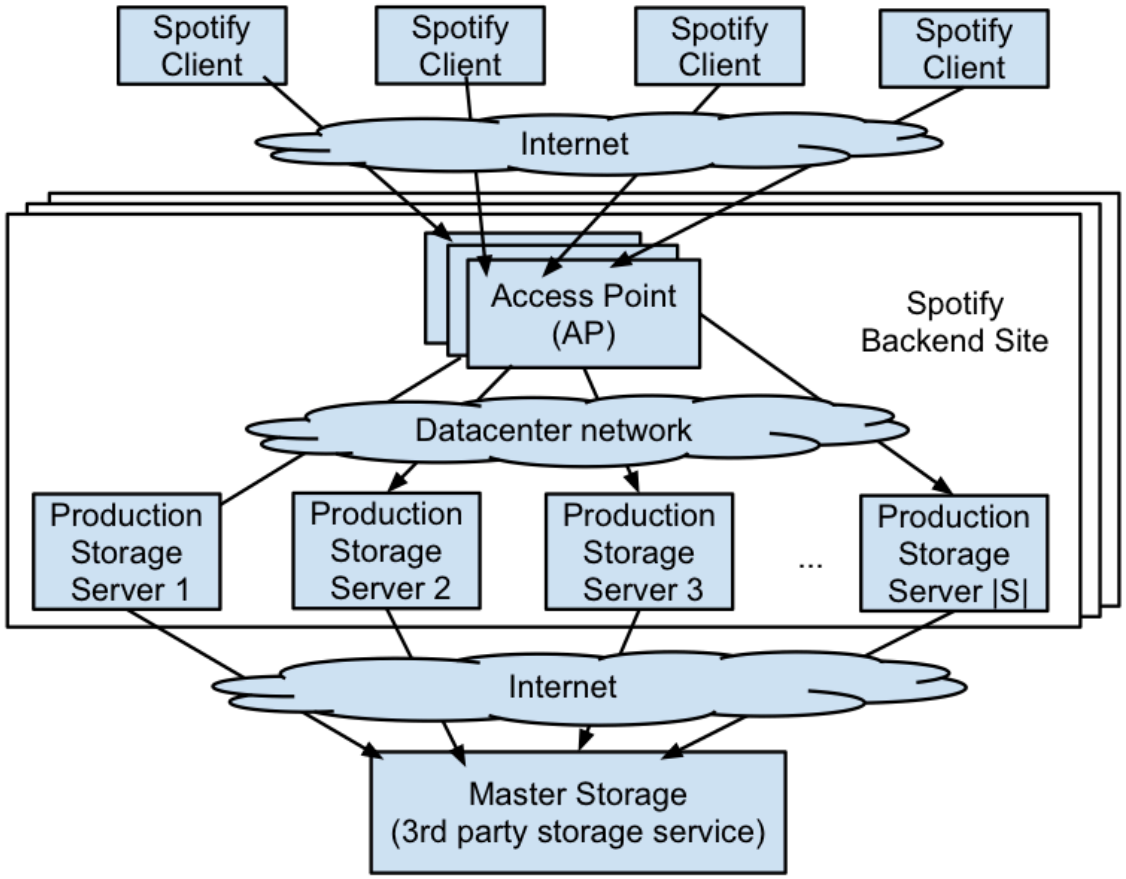
1. Resource allocation for a large-scale cloud environment
   - Motivation
   - Problem
   - Approach
   - Evaluation results
   - Related research / publications
2. **Performance modeling of a distributed key-value store**
   - Motivation
   - Problem
   - Approach
   - Evaluation results
   - Related research / publications

– Contributions

– Open questions

Spotify Backend

A distributed key-value store

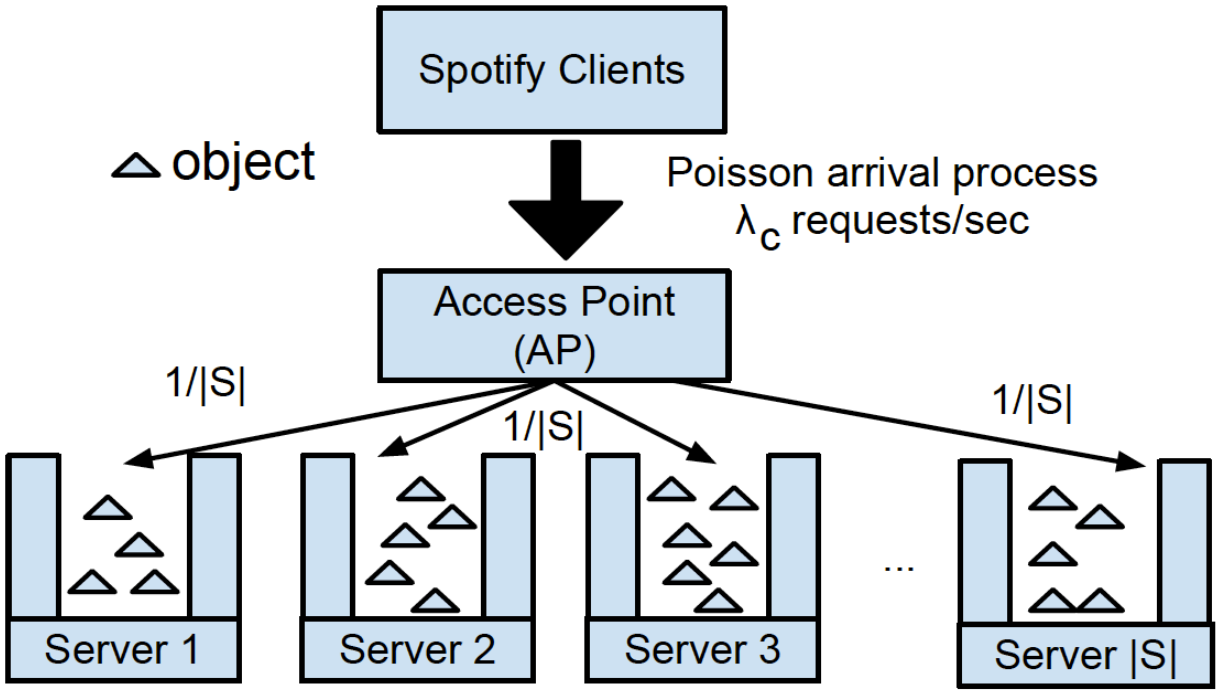


- Motivation: low latency is key to the Spotify service

Problem & Approach

- Development of analytical models for
  - Predicting response time distribution
  - Estimating capacity under different object allocation policies
    - Random policy
    - Popularity-aware policy
- The models should be
  - Accurate for Spotify's operational range
  - Obtainable
  - Simple
  - Efficient
- Approach
  - Simplified architecture
  - Probabilistic and stochastic modeling techniques

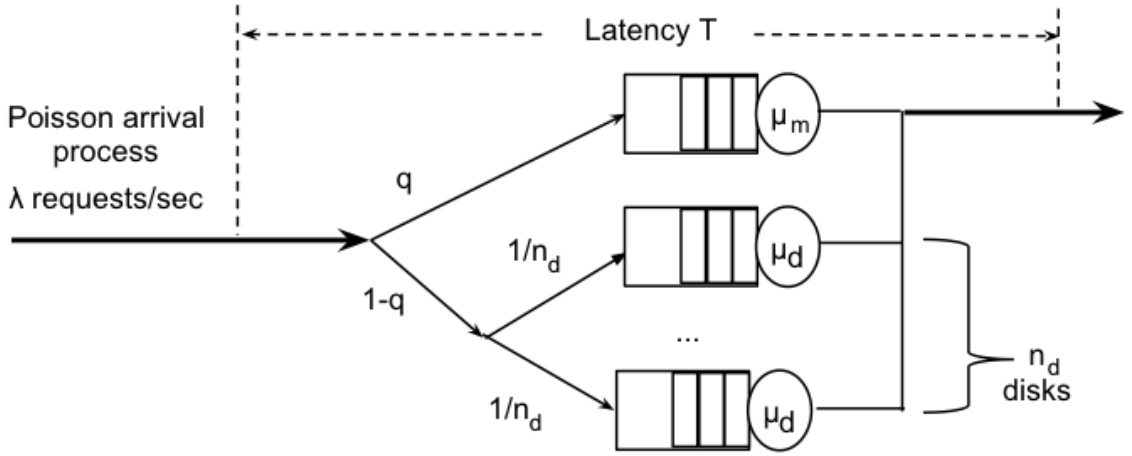
Simplified Architecture for a Particular Site



- Model only Production Storage
- AP selects a storage server uniformly at random
- Ignore network and access point 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 is

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

Model for a cluster of storage servers

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

$$\Pr(T_c \le t) = \frac{1}{|S|}\sum_{s \in S} f\left(t, n_d, \mu_{d,s}, \frac{\lambda_c}{|S|}, q_s\right)$$

Model for Estimating Capacity under Different Object Allocation Policies

Capacity

Ω : Max request rate to a server before a QoS is not satisfied

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

Capacity of a cluster under the popularity-aware policy

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

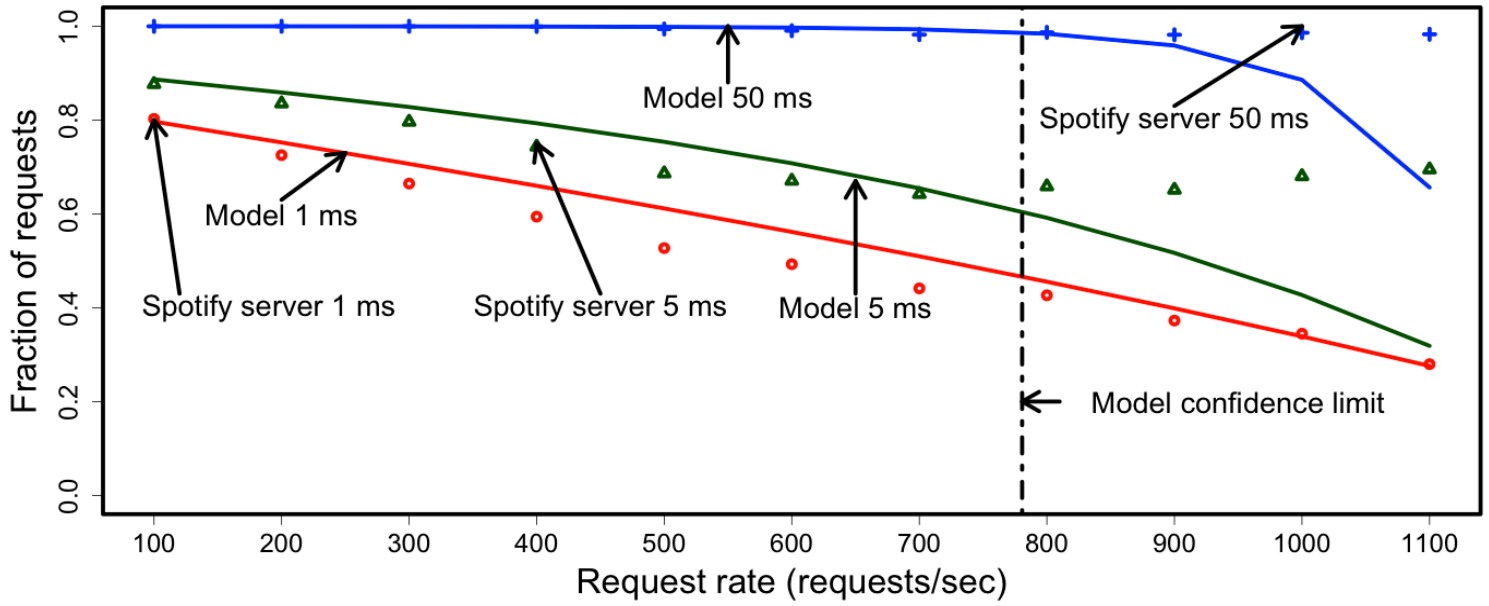
Capacity of a cluster under the random policy

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

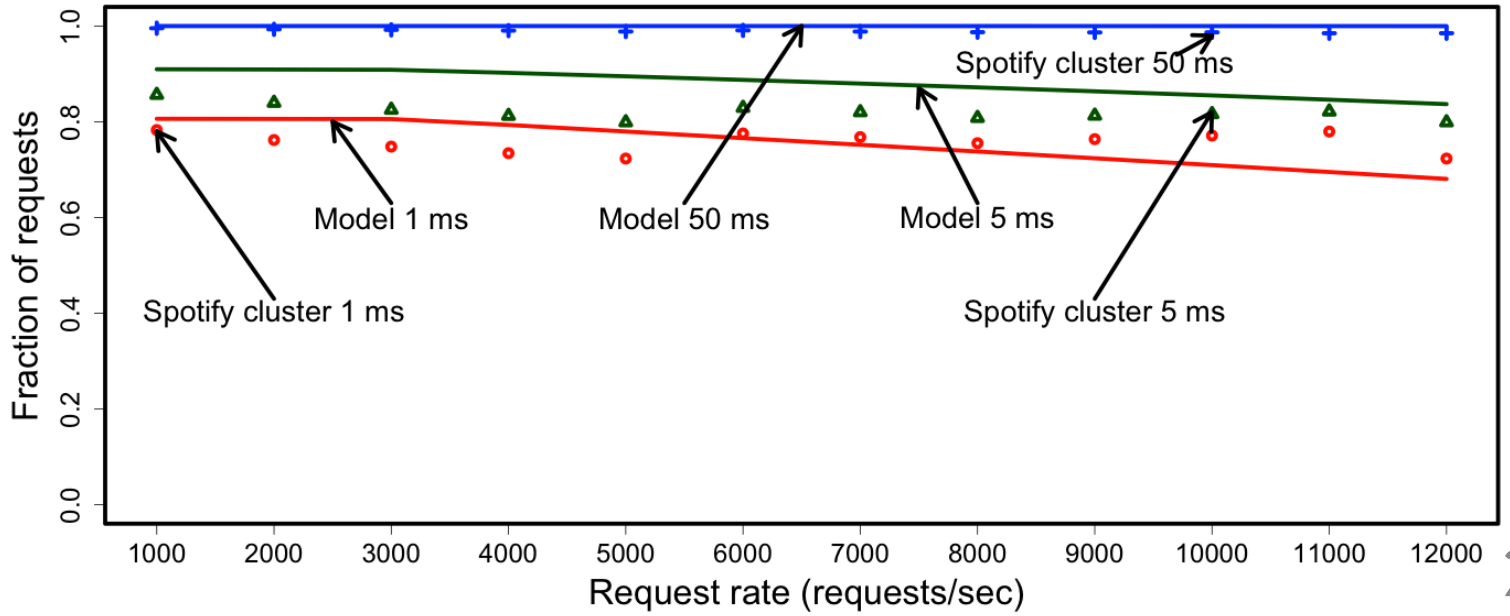
When $|O_m| \approx \frac{|O|}{|S|} + \sqrt{\frac{2|O|\ln|S|}{|S|}\left(1 - \frac{\log\log|S|}{2\log|S|}\right)}$, Assuming $|O| >> |S|(\log|S|)^3$

Evaluation Results – Response Time Dist.

Spotify Server



Spotify Cluster



Evaluation Results – Estimating Capacity

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

Related Research

- Distributed key-value store
  - Amazon Dynamo (G. Decandia et al., 2007), Cassandra (A. Lakshman, 2010), Riak, HyperDec, Scalaris
- Performance modeling of a storage device
  - (J.D. Garcia et al., 2008), (A.S. Lebrecht., 2009), (F. Cady et al., 2011), (S. Boboila and P. Desnoyers, 2011), (P. Desnoyers, 2012), (S. Li and H.H. Huang, 2010)
- White-box vs. black-box performance model for a storage system
  - Whitebox: (A.S. Lebrecht., 2009), (F. Cady et al., 2011), (S. Boboila and P. Desnoyers, 2011), (P. Desnoyers, 2012), **this thesis**
  - Blackbox: (J.D. Garcia et al., 2008), (S. Li and H.H. Huang, 2010), (A. Gulati et al., 2011)

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), 2013*.

Contributions

- We designed, developed, and evaluated a generic protocol for resource allocation that
  - supports joint allocation of compute and network resources
  - Is generic and extensible for different management objectives
  - enables scalable operation (> 100’000 machines)
  - supports dynamic adaptation to changes in load patterns
- We designed, developed, and evaluated performance models for predicting response time distribution and capacity of a distributed key-value store
  - Simple yet accurate for Spotify’s operational range
  - Obtainable
  - Efficient

Open Questions for Future Research

- Resource allocation for a large-scale cloud environment
  - *Centralized vs. decentralized resource allocation*
  - *Multiple data centers & telecom clouds*
- Performance modeling of a distributed key-value store
  - *Black-box models for performance predictions*
  - *Online performance management using analytical models*