Predicting response times for the Spotify backend

Rerngvit Yanggratoke[1], Gunnar Kreitz [2],
Mikael Goldmann [2], Rolf Stadler [1]

[1]ACCESS Linnaeus Center, KTH Royal Institute of Technology, Sweden

[2]Spotify, Sweden

8th International Conference on
Network and Service Management (CNSM 2012), Las Vegas
24 October, 2012

What is Spotify?

- On-demand music streaming service, similar to MOG or Rhapsody.
- Large catalogue, over 15 million tracks.
- Over 15M active users and 4M subscribers around the world.

"Spotify: large-scale distributed system with real users."

Overview

- Spotify is a peer-assisted system.
- Response time and latency.

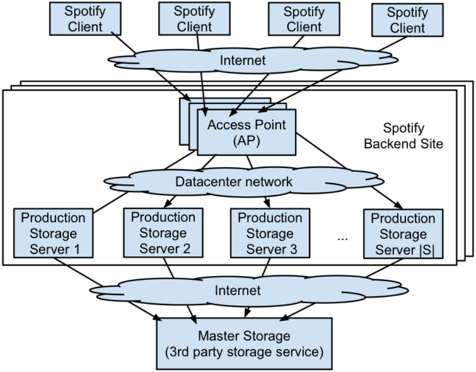
Goal

- An analytical model for distribution of the response time for the Spotify backend.
- The model should be tractable and accurate.

- Motivation
  - Low latency is key to the Spotify service.
  - Related works do not have a performance model or model only the average response time.
- Steps
  - Study Spotify backend and find the simpler system model.
  - Replicate Spotify backend implementations at KTH testbed.
  - Develop and validate the model for
    - KTH testbed with small and large servers.
    - Spotify operational infrastructure.

Outline

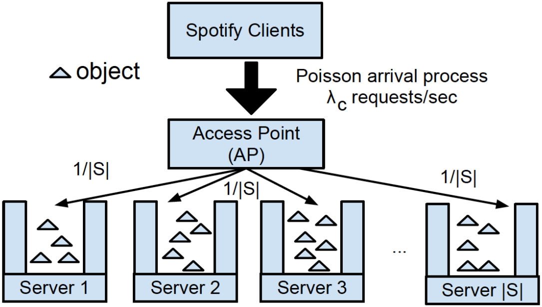
The Spotify backend architecture



- Backend sites: Stockholm, London, Ashburn.
- Master Storage stores all songs.
- Production Storage acts as a caching layers.

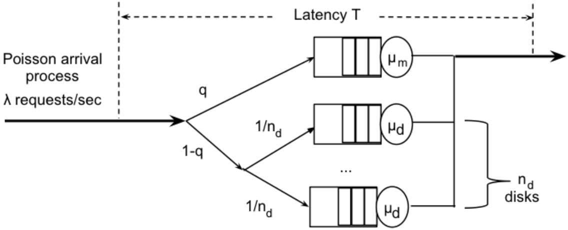
Outline

Simplified architecture for a particular site



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

Model of a single storage server



- A request is served from
  - memory with probability q,
  - one of the disks with probability $(1-q)/n_d$ (n_d: number of identical disks).
- Model memory or a disk as an $M/M/1$ queue.
  - μ_m: service rate of memory and μ_d: disk. $\mu_m \gg \mu_d$ holds.

Model for a single storage server

probability that a request to the server is served below a latency t is

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

Model of a cluster of storage servers

- Set of storage servers S.
- Arrival process: Poisson process with rate λ_c.
- A request forwarded to a server $s \in S$ independently and uniformly at random.
- For server s
  - $\mu_{d,s}$: the service rate of a disk.
  - $n_{d,s}$: number of identical disks.
  - q_s: probability that the request is served from memory.
- $f(t, n_d, \mu_d, \lambda, q) = Pr(T \leq t)$.
- T_c: latency of a request for the cluster.

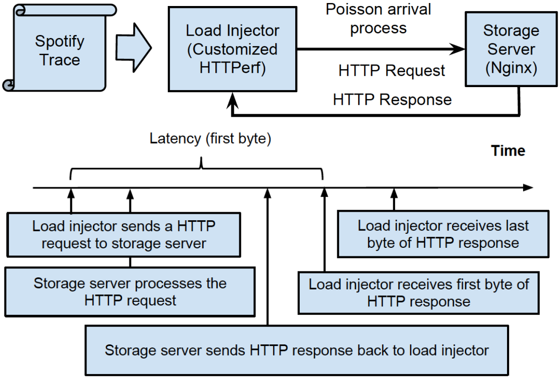
Model of a cluster of storage servers

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

$$Pr(T_c \leq t) = \frac{1}{|S|} \sum_{s \in S} f(t, n_{d,s}, \mu_{d,s}, \frac{\lambda_c}{|S|}, q_s). \tag{2}$$

Outline

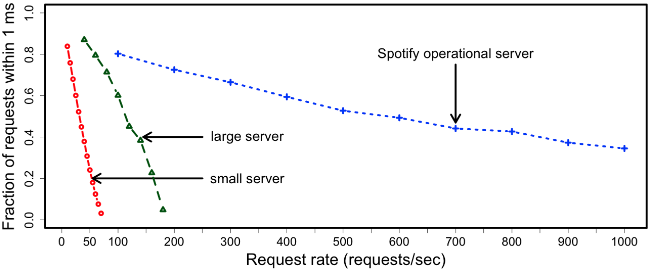
Setup and measuring of request latency for a server



Validation compares measurements at load injector with model (equation 1).

Estimating model parameter for a storage server

- q - benchmark the server with a range of request rates.



Approximate q through least-square regression.

- μ_d - run **iostat** when the server is serving a Spotify load.

Parameter	Small server	Large server	Spotify server
μ_d	93	120	150
n_d	1	1	6
α	0.0137	0.00580	0.000501
q_0	0.946	1.15	0.815

Model confidence limit for a single storage server

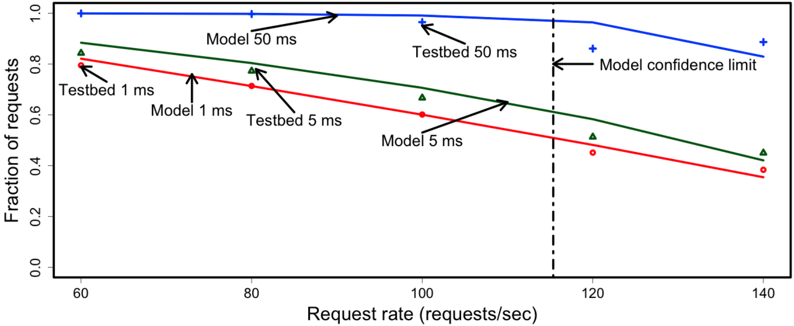
- Extensive testing reveals that model predictions are close to the measurements when the average length of disk queue is at most one.
- The model confidence limit is the maximum request rate below which above condition holds.

Model confidence limit for a single storage server

The limit λ_L is the positive root of

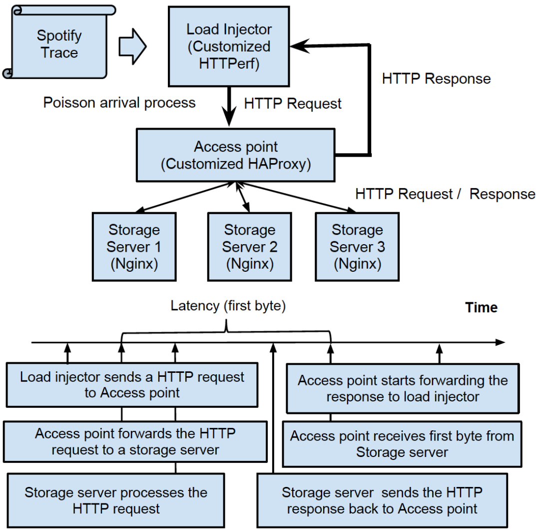
$$\alpha\lambda_L^2 + (1 - q_0)\lambda_L - \frac{1}{2}\mu_d n_d = 0.$$

Validation for a single large server



The prediction accuracy decreases with increasing request rate.
The maximum error is within 5%.

Setup and measuring of request latency for a cluster



Validation compares measurements at the access point with model (equation 2).

Model confidence limit for a storage cluster

- The *confidence limit* is the max request rate to the cluster, such that the rate to any server does not exceed the confidence limit for a single server with high probability.
- To compute the limit, we must know the load of the highest loaded server with high probability by applying the balls-and-bins model.

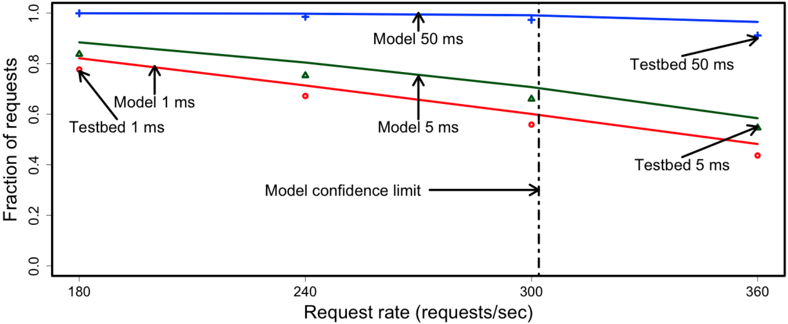
Model confidence limit for a storage cluster

The limit is the smaller root of

$$\frac{1}{|S|^2}\lambda_{L,c}^2 + (\frac{2\lambda_L}{|S|} - \frac{2\log|S|K_{\beta,|S|}}{|S|})\lambda_{L,c} + \lambda_L^2 = 0,$$

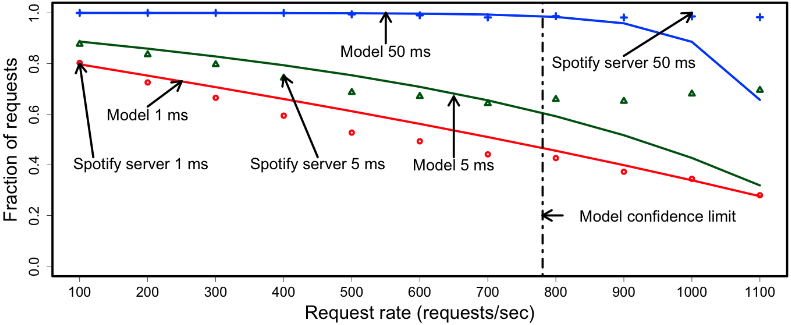
whereby $\beta = 2$ and $K_{\beta,|S|} = 1 - \frac{1}{\beta}\frac{\log\log|S|}{2\log|S|}$.

Result for a cluster of three large servers



The prediction accuracy decreases with increasing request rate.
The maximum error is within 4.6%.

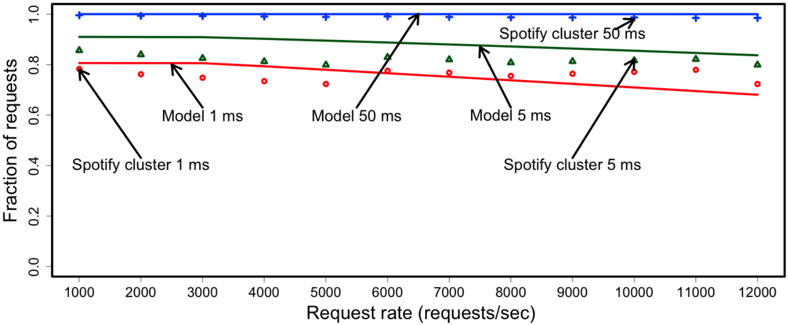
Result for a Spotify operational server - single server



The prediction accuracy decreases with increasing request rate. The maximum error is within 8.5%.

Result for Spotify operational service - Stockholm site

- 31 operational Spotify storage servers.
- 24 hours measurements data.
- Arrival rate and response time distribution for the requests (five-minutes averages).

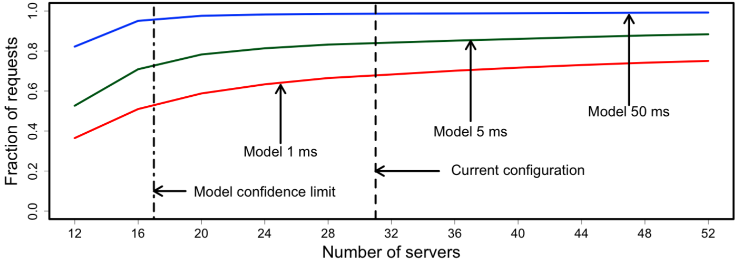


- The maximum error is within 9.7%.

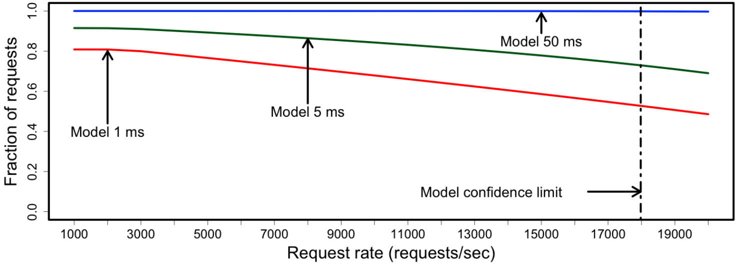
Outline

Applications of the model

Varying the number of servers for the load of 12,000 requests/sec (peak load).



Varying the load for 25 storage servers.



Outline

Conclusions

- Develop the model for distribution of the response time for the Spotify backend that is tractable.
- Extensive validation of the model, on (1) KTH testbed (2) the Spotify operational infrastructure.
  - The model predictions are accurate for the lightly-loaded storage system, with error up to 11%.
  - The confidence range of our model covers the entire operational range of the load to the Spotify storage system.
- For instance, the model confirms that the Stockholm site could handle significantly higher peak load, or handle the load with fewer servers.

Current / Future work

- An online performance management system for a distributed key-value store like the Spotify storage system.
- Evaluation of object allocation policies.