Predicting Real-time Service-level Metrics from Device Statistics

Rerngvit Yanggratoke (1), Jawwad Ahmed (2),
John Ardelius (3), Christofer Flinta (2),
Andreas Johnsson(2), Daniel Gillblad (3), Rolf Stadler (1)

(1) KTH Royal Institute of Technology, Sweden
(2) Ericsson Research, Sweden
(3) Swedish Institute of Computer Science (SICS), Sweden

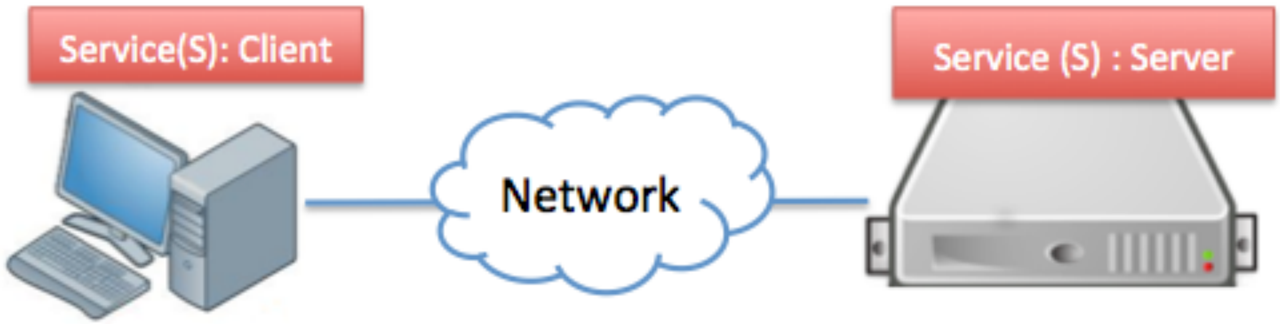
14th IFIP/IEEE Symposium on
Integrated Network and Service Management 2015 (IM 2015)

May 14, 2015

Outline

- Problem / motivation
- Design goal / approach
- Testbed for producing traces
- X-Y traces for model evaluation
- Evaluation method
- Selected evaluation results
- Conclusions / ongoing work

Problem / motivation



Y: service-level metrics

- **Video streaming:** video frame rate, audio buffer rate, RTP packet rate
- We select Video streaming (VLC) as an example service

X: device statistics

- CPU load, memory load, #network active sockets, #context switching, #processes, etc..
- We read raw data from /proc provided by Linux kernel

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

Motivation :

- Building block for real-time service assurance for a telecom cloud

Design goal / approach

Existing works

1. Apply formal models, e.g., queuing models, to model and analyze the system and the service.
2. Statistical learning on few service-specific features (<= 10) (e.g., service queue length).

Design goal ➔ "Service-agnostic prediction"

Approach

1. Take as many features as we can (>= 4000 features)
2. Statistical learning on low-level (OS-level) metrics
   - CPU load, memory load, #network active sockets, #context switching, #processes, disk statistics, etc

Note

We do not consider network statistics and client low-level metrics.
Network and client machine are lightly loaded.

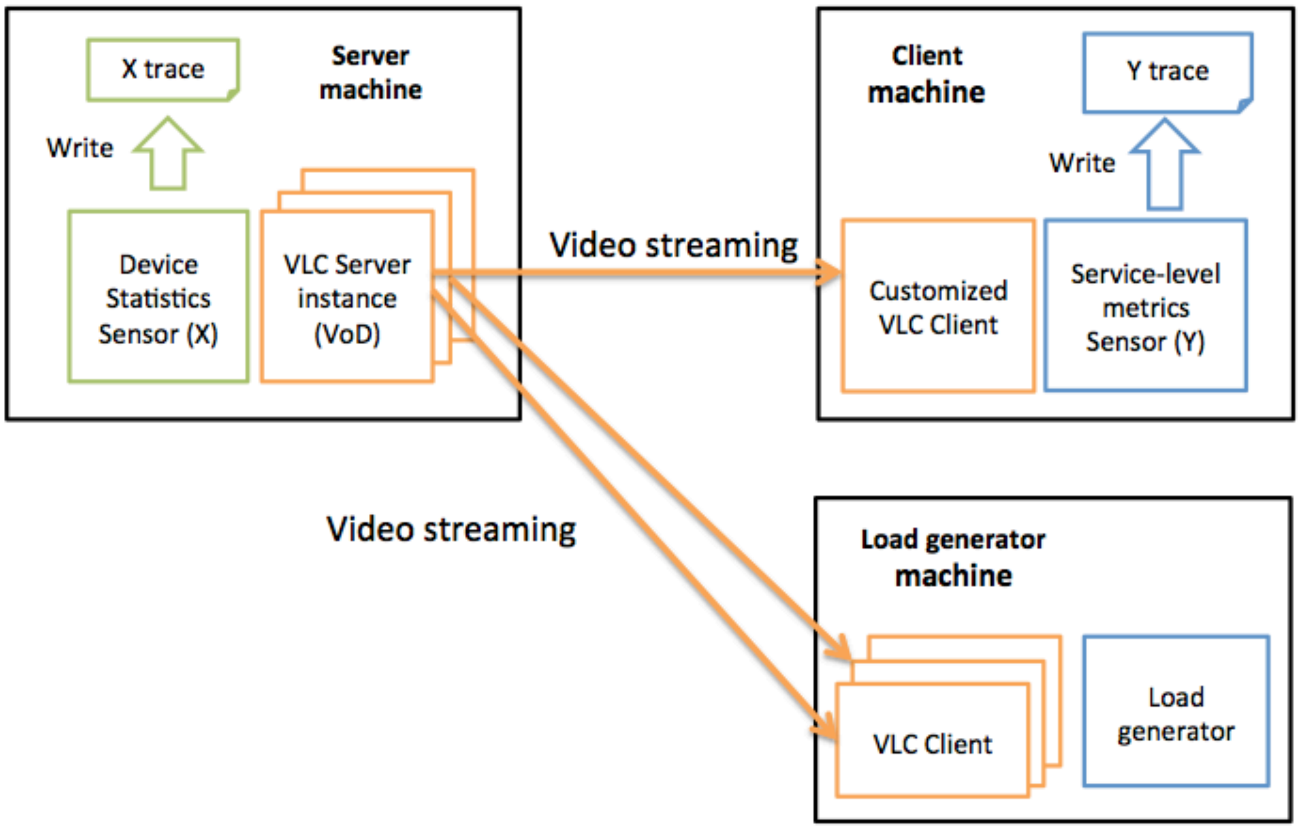
Device statistics X_{proc} and X_{sar}

- Linux kernel statistics X_{proc}
  - Features extracted from /proc directory
  - CPU core jiffies, current memory usage, virtual memory statistics, #processes, #blocked processes, …
  - About 4000 metrics
- System Activity Report (SAR) X_{sar}
  - SAR computes metrics from /proc over time interval
  - CPU core utilization, memory and swap space utilization, disk I/O statistics, …
  - About 840 metrics
- X_{proc} contains many OS counters, while X_{sar} does not
- For model predictions, include numerical features

Service-level metrics Y

- Video streaming service based on VLC media player
- Measured metrics
  - Video frame rate (frames/sec)
  - Audio buffer rate (buffers/sec)
  - RTP packet rate (packets/sec)
  - …
- We instrumented the VLC software to capture underlying events to compute the metrics.

Testbed for producing traces

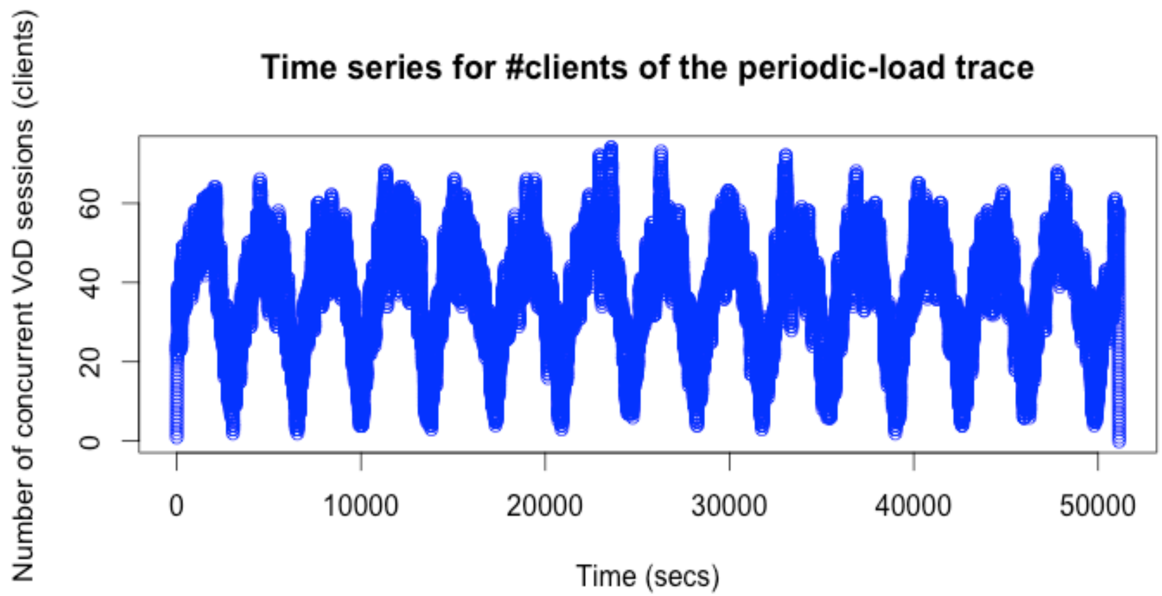


Dell PowerEdge R715 2U rack servers, 64 GB RAM, two 12-core AMD Opteron processors, a 500 GB hard disk, and a 1 Gb network controller

X-Y traces for evaluation

We collect the following traces

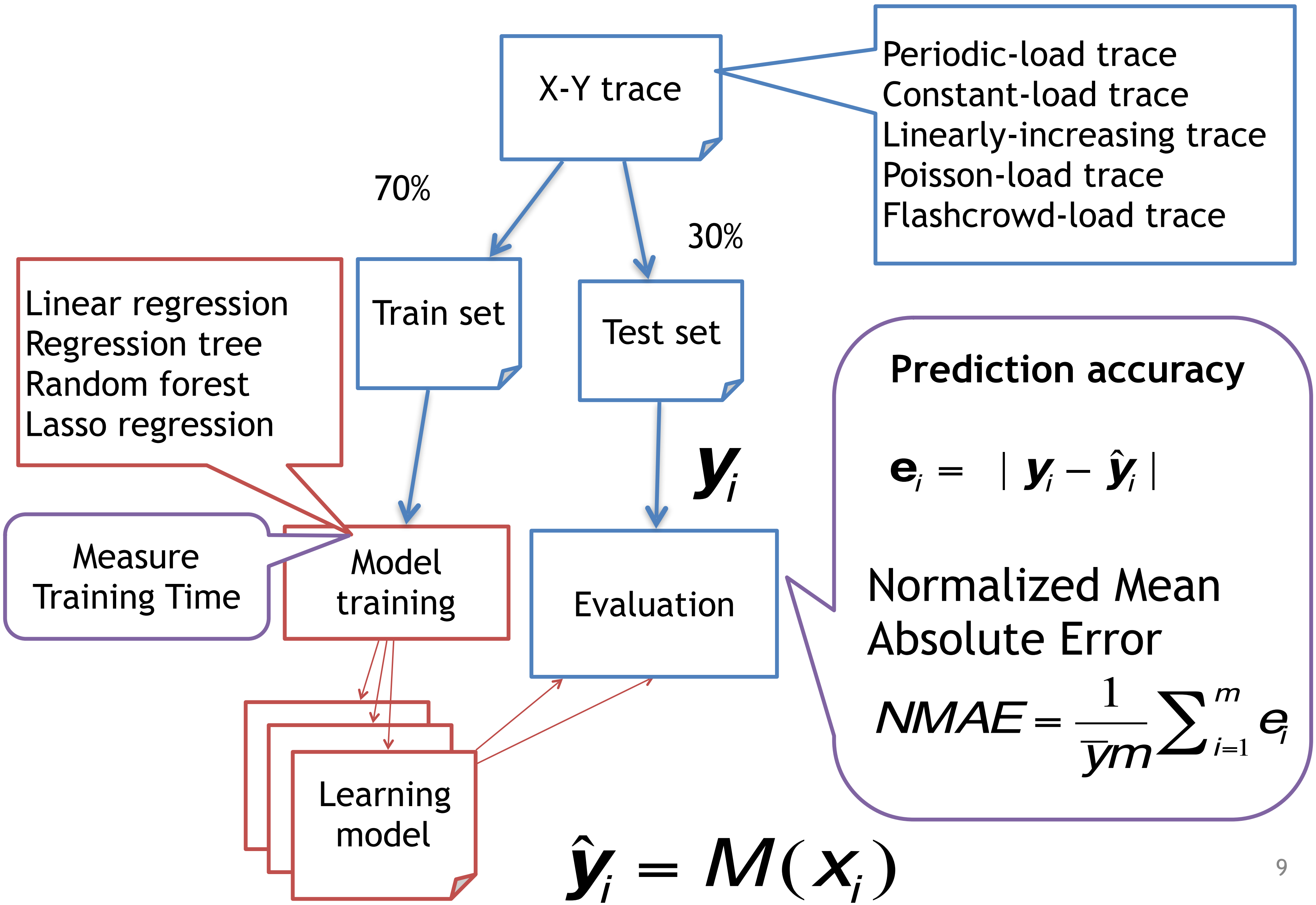
– Periodic-load trace, flashcrowd-load trace, constant-load trace, poisson-load trace, linearly-increasing-load trace



We published the traces used in this work

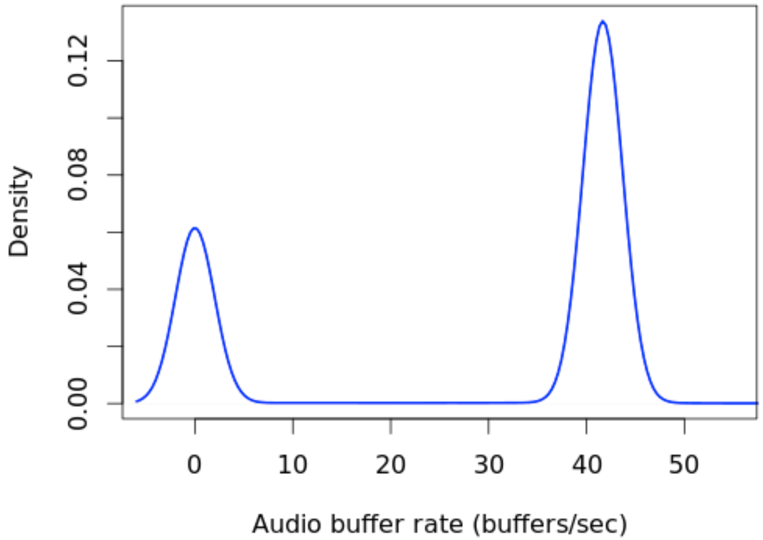
http://mldata.org/repository/data/viewslug/realm-im2015-vod-traces/

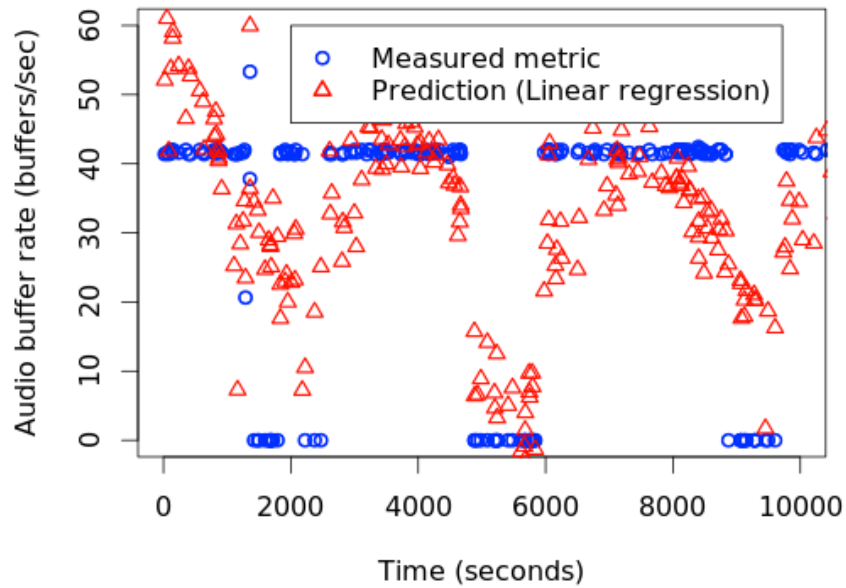
Model training and evaluation



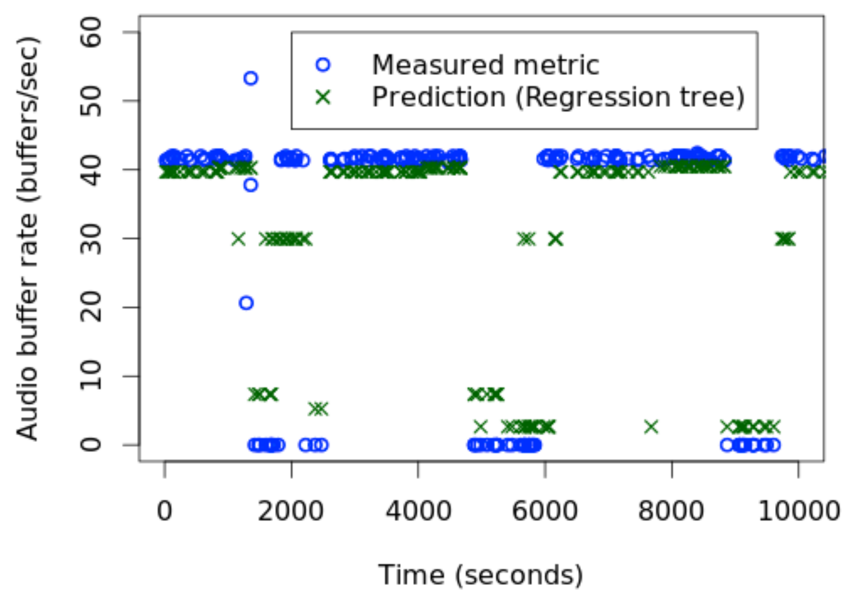
Selected evaluation results

Audio buffer rate





Method	NMAE (%)
Linear regression	41
Regression tree	19



- Y – bimodal distribution
- Regression tree outperforms least-square linear regression

Evaluation results – periodic-load trace

Device statistics	Regression method	NMAE (%)		
		Video	Audio	RTP
X_sar	Linear regression	12	41	15
	Lasso regression	16	51	17
	Regression tree	11	19	19
	Random forest	**6**	**0.94**	**15**
X_proc	Linear regression	26	59	39
	Lasso regression	23	63	35
	Regression tree	23	61	36
	Random forest	22	60	34

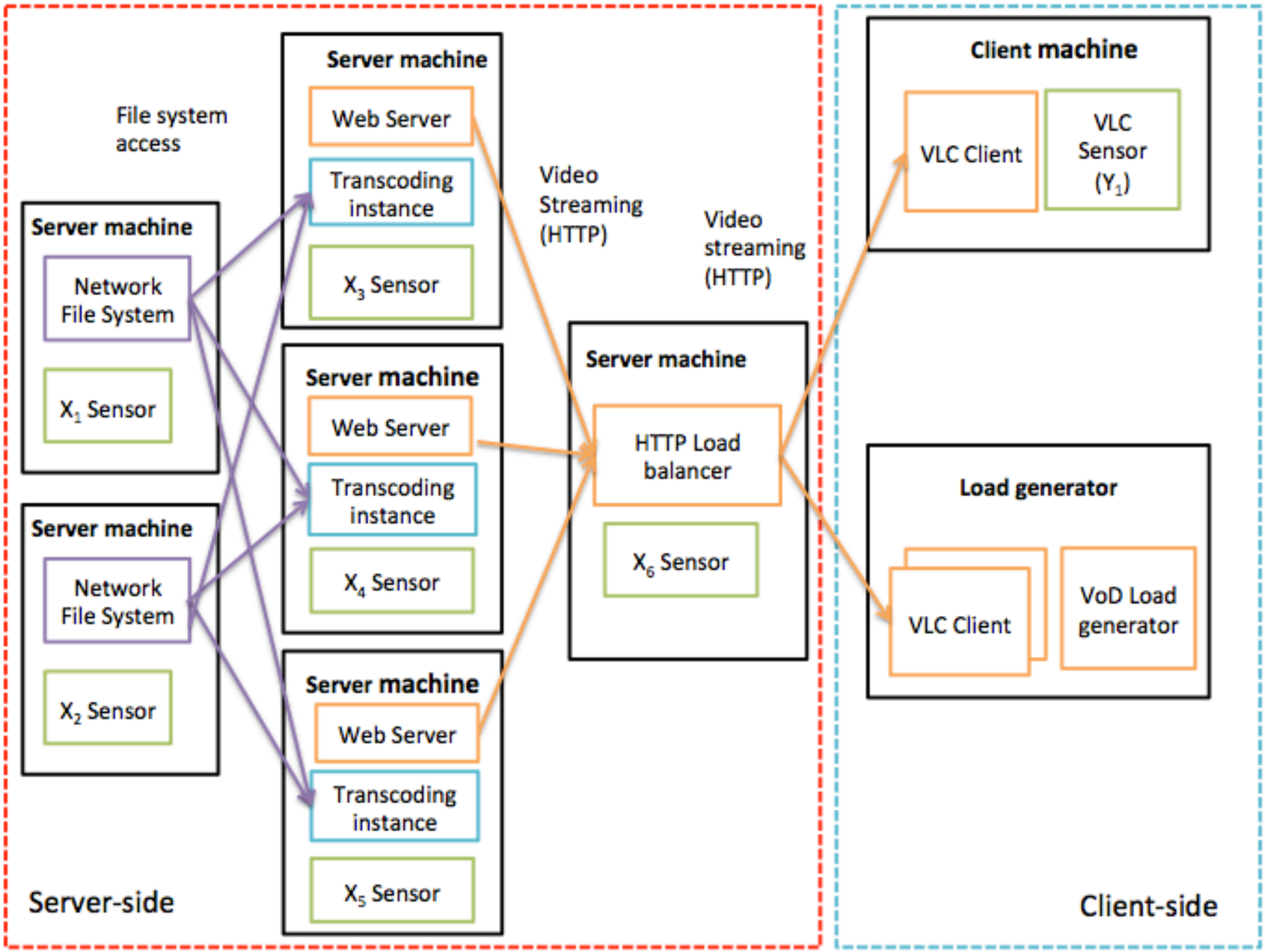
Evaluation results – other traces

Regression method	Trace	NMAE (%)		
		Video	Audio	RTP
Linear regression	Constant-load trace	0.47	0.62	12
	Poisson-load trace	3	3.6	12
	Linearly-increasing trace	6.1	7.0	13
	Flashcrowd-load trace	9	28	14
Random forest	Constant-load trace	0.34	0.57	10
	Poisson-load trace	2.0	1.3	11
	Linearly-increasing trace	3.4	0.69	11
	Flashcrowd-load trace	**6.0**	**4.4**	**11**

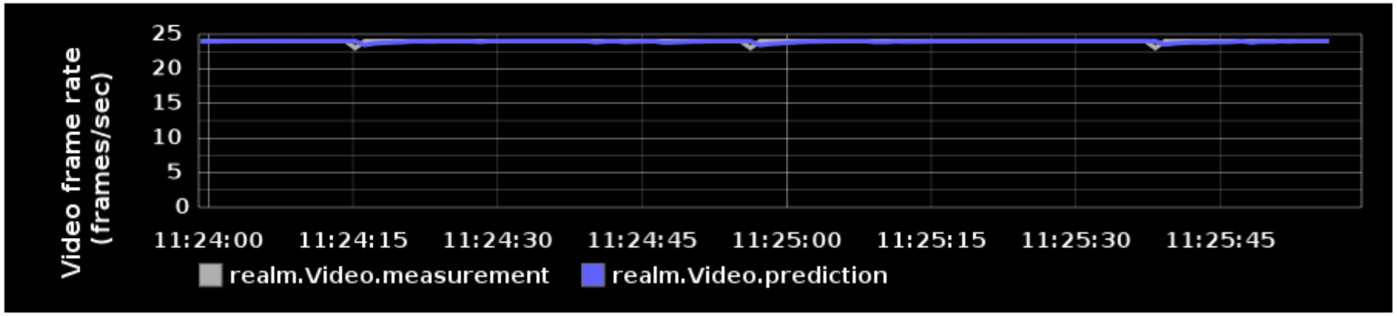
Conclusions

- It is feasible to accurately predict client-side metrics based on low-level device statistics
  - NMAE below 15% across service-level metrics and traces
- Preprocessing of X is critical
  - Significant improvement of prediction accuracy
- There is a trade-off between computational resources vs. prediction accuracy
  - Random forest vs. linear regression

Extended test bed

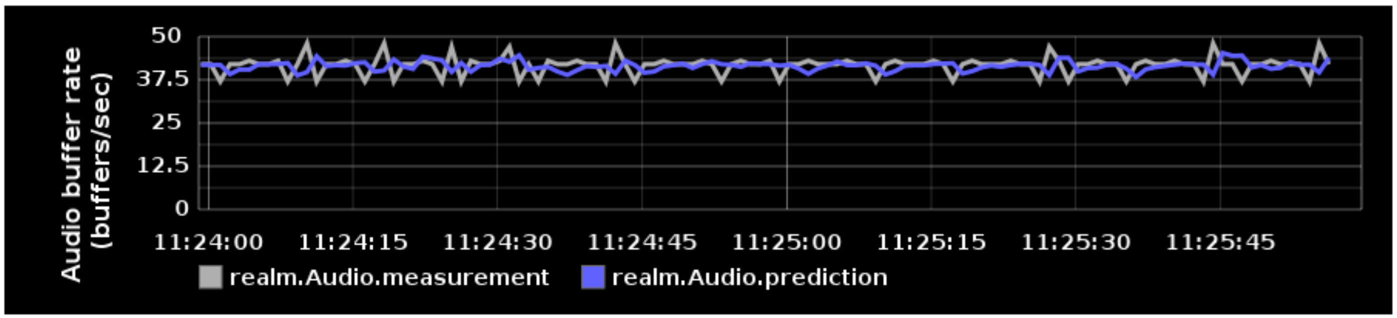


Real-time Predictions of Service Metrics from Device Statistics



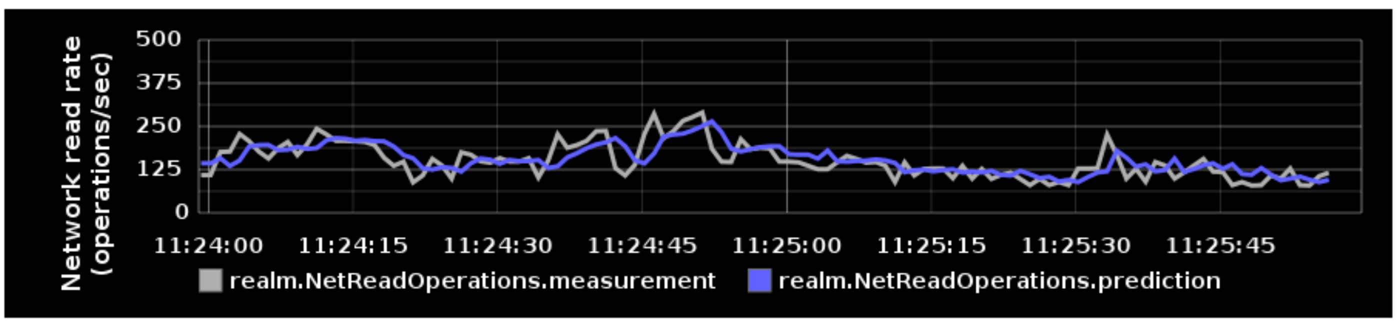
Normalized Mean Absolute Error (last 5 minutes)

1.40 %



Normalized Mean Absolute Error (last 5 minutes)

7.23 %

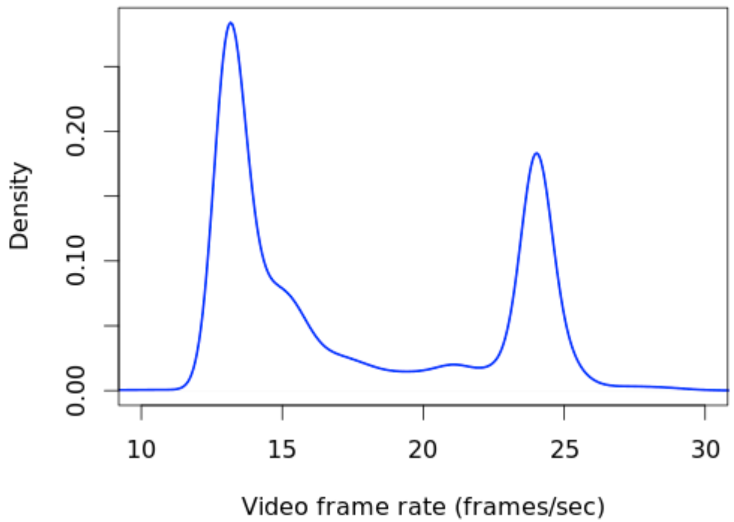


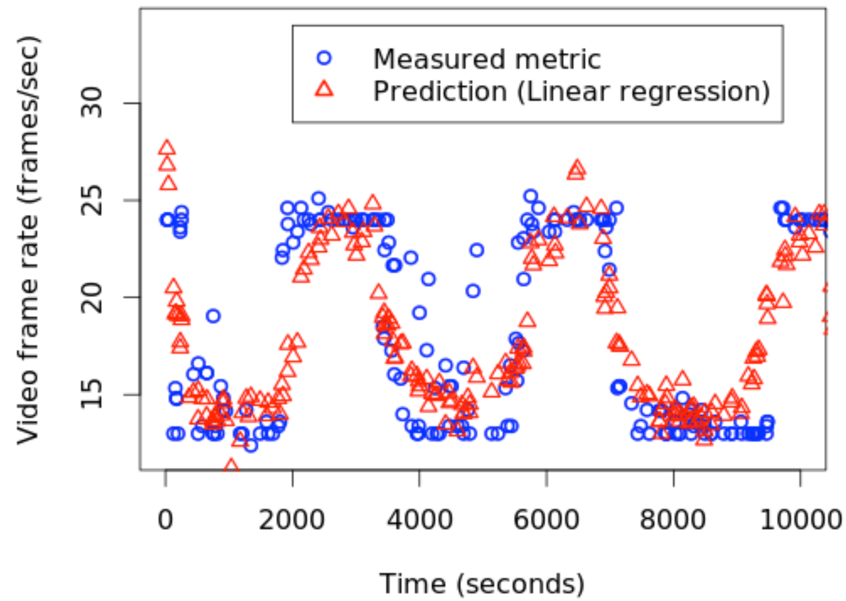
Normalized Mean Absolute Error (last 5 minutes)

24.53 %

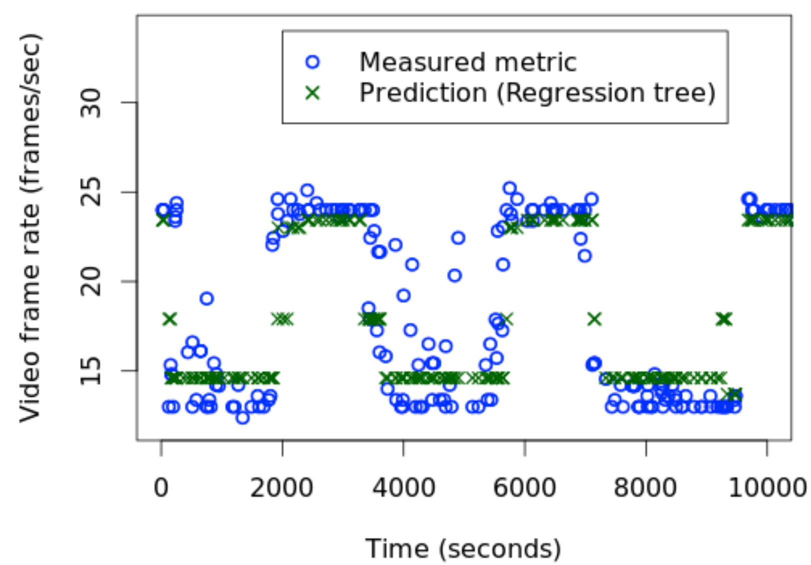
We compute predictive models on kernel statistics collected from each machine in a cluster. Examples: the rate of context switches and the number of active TCP connections.

Video frame rate



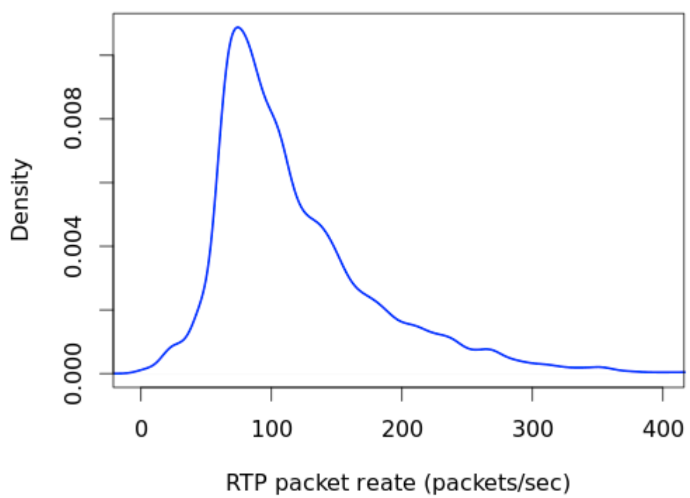


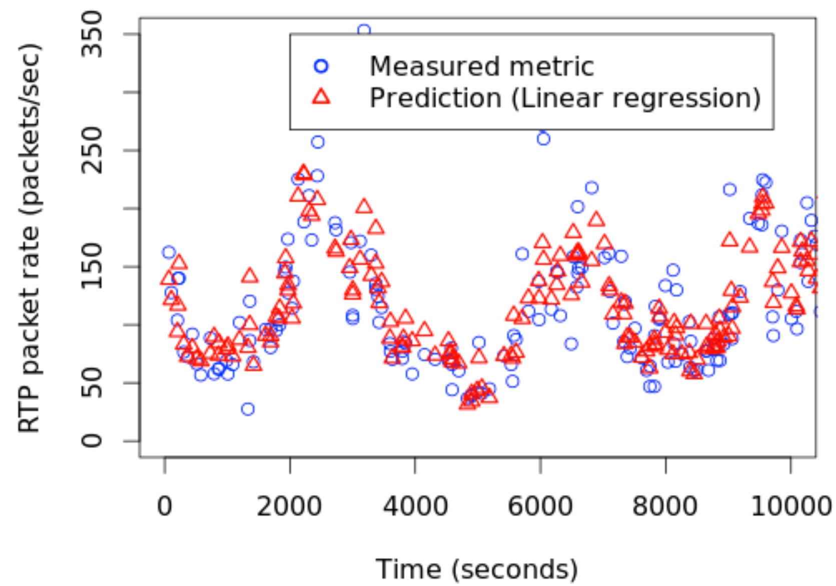
Method	NMAE (%)
Linear regression	12
Regression tree	11



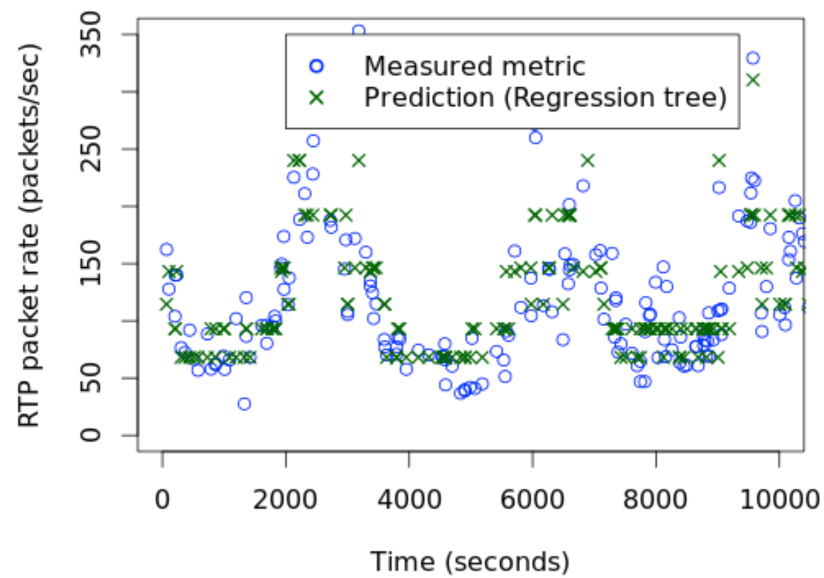
- Y – bimodal distribution
- Both methods have similar prediction accuracy

RTP packet rate





Method	NMAE (%)
Linear regression	15
Regression tree	19



- Y – wider spread distribution
- Least-square linear regression outperforms regression tree

Periodic load trace