OnlineElastMan:
Self-Trained Proactive Elasticity Manager for Cloud-Based Storage Services

Ying Liu, Daharewa Gureya, Ahmad Al-Shishtawy, Vladimir Vlassov

2016 IEEE International Conference on Cloud and Autonomic Computing (ICCAC)

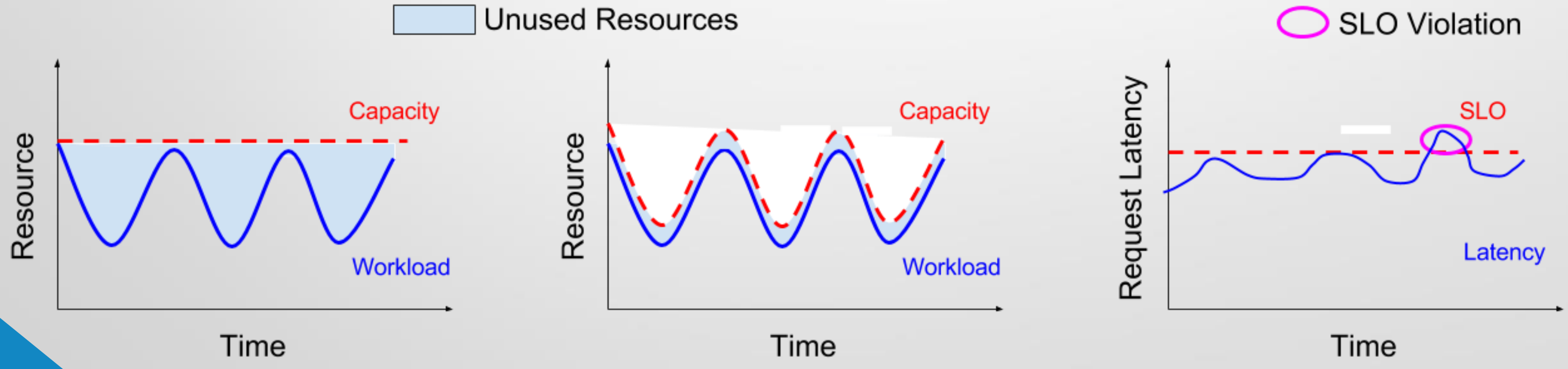
Augsburg, Germany, September 13, 2016

Outline

- Motivation and Background
- Online ElastMan Design
- Evaluation Results
- Conclusions

Elasticity Control (Auto-Scaling)

- Elastic Provisioning: allocate resources dynamically in response to the changes of workload
- Goal: minimize cost while maintaining the desired Service Level Objectives (SLOs), e.g., latency



Traditional Provisioning
(a)

Elastic Provisioning
(b)

Service Level Objective
(c)

Cloud Storage Services

- Put-Get operations (key-value stores)
- Horizontal scalability
- Replicated
- Load-balancing
- Apache Cassandra

Existing Approaches for Elasticity Control

- Too Simple: Threshold based rules
  - Easy to implement for small scale systems
  - Reduced accuracy and adaptability
- Too Complex: Control theory, Machine learning, …
  - Requires manual training and tuning of the controller
  - Targeting specific services and use cases

Some Challenges

- Nonlinear & Discrete
    - 1 VM + 1 VM = Double capacity
    - 100VM + 1VM = 1% increase
- Startup Delay
    - Stateful services such as storage need to be initialized with data
- Workload Prediction

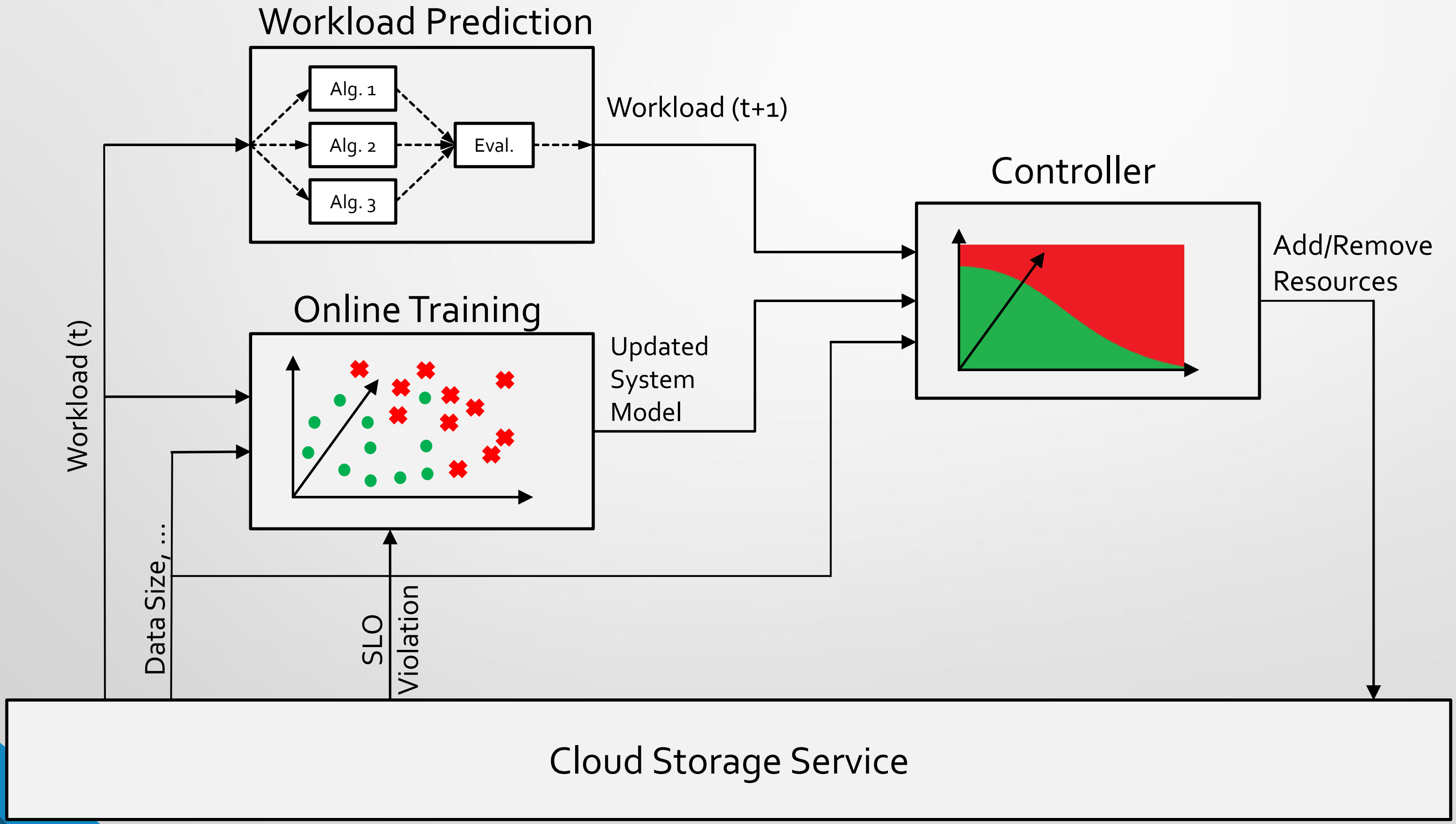
Working “Out-of-the-Box” Vision

- Generic
- Easy to integrate into your service
- Self-training
- Adapts to unexpected changes
- Pluggable architecture

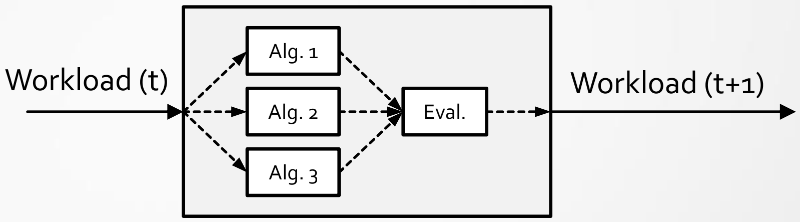
Monitored parameters

- Workload
  - read/write operations
  - data size
- SLO: operation latency
- Other parameters
  - Instance size
  - Hardware (processor, disks, ...)
  - Software & OS version

Overall Controller Architecture



Workload Prediction

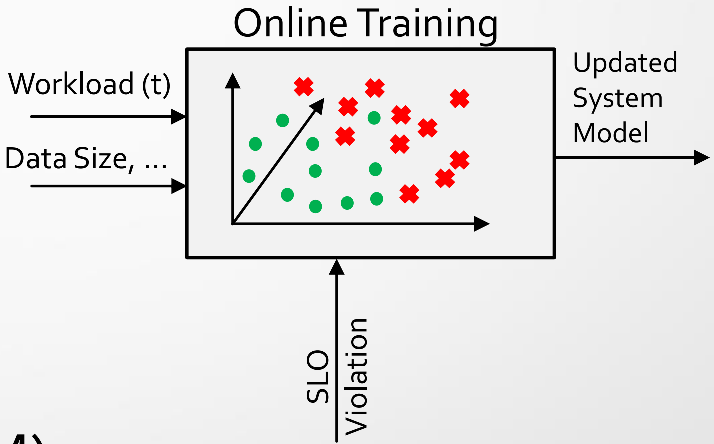


- Depends on the workload patterns
- Provide several generic workload prediction algorithms
- Use a "weighted majority algorithm" to evaluate and select best algorithm for the current workload
  - Construct a compound algorithm from a pool of prediction algorithms

Workload Prediction

- ARIMA: Autoregressive Integrated Moving Average model
  - Popular approach to time series forecasting
  - AR, I, MA Components
  - ARIMA(p,d,q)
    - ARIMA(0,1,1) is a simple exponential smoothing.
    - ARIMA(2,0,0) is a second-order autoregressive model

Multidimensional Performance model



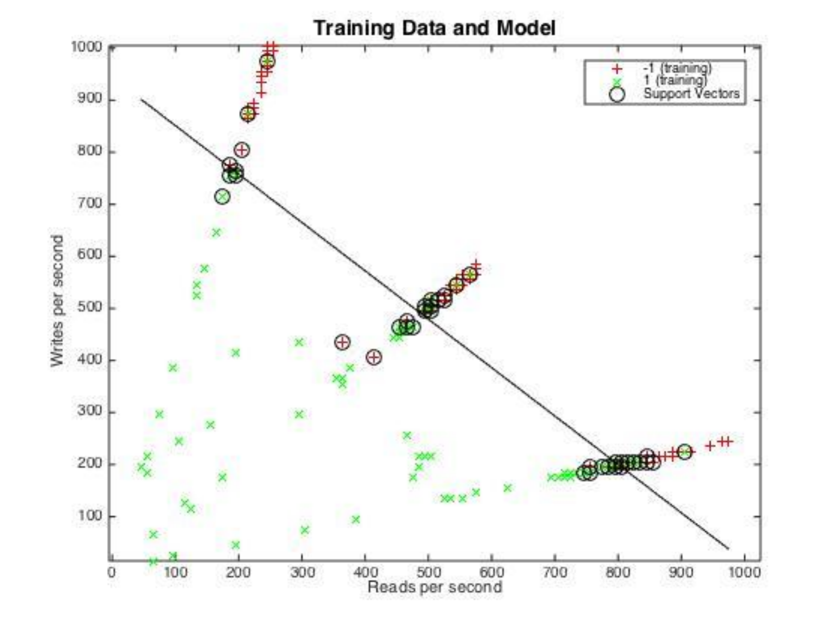
- Find the relation between the workload and the SLO
- Use Support Vector Machine (SVM)
- 3 dimensions (read throughput, write throughput, data size)

Linear SVM -- the labeled data set

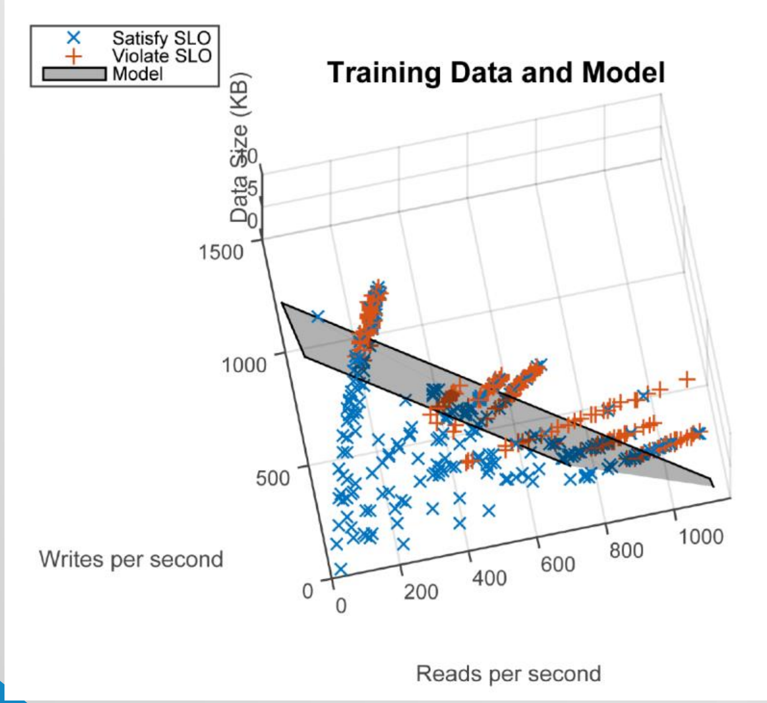
- Granularity of the model
  - Each request is mapped to a training case with data format x ϵ Rn, where n is the training features, e.g., read, write, data_size, etc.
  - Then, it is labelled y ϵ {1, i.e., SLO_commitment, -1, i.e., SLO_violation} from the collected service latency
  - Training cases are mapped to discretized data plane
- Historical data buffer
  - The n most recent training cases are stored in each cell of the discretized data plane
- Confidence level
  - Training cases in each cell make a consensus for a global label
- Update frequency
  - The global label for each cell is updated with a configurable rate

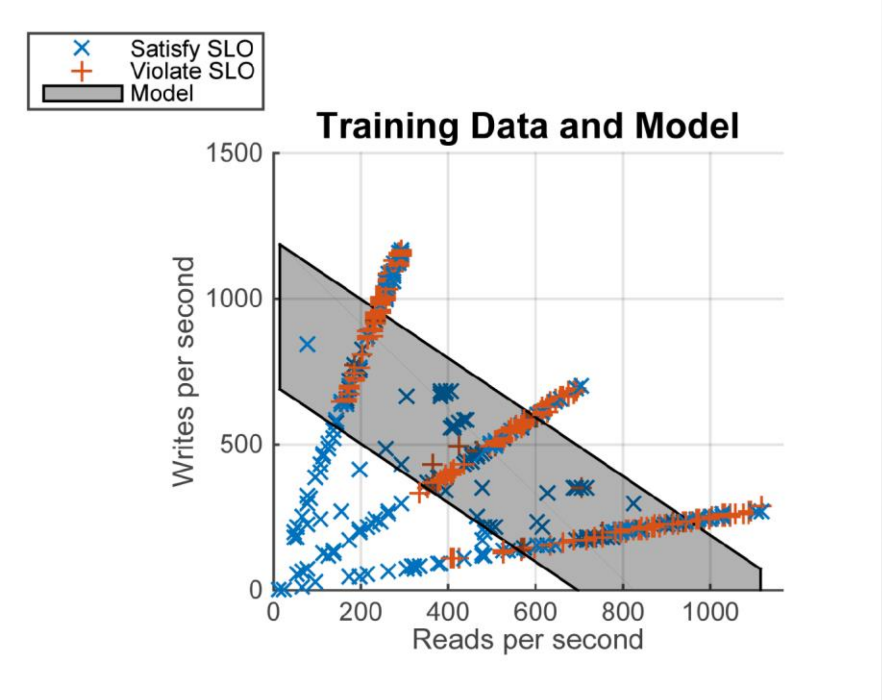
Linear SVM – the model

- Globally labeled cells are the input for the linear SVM
- $w^Tx + b = 0$ is the linear separator (plane), given that $y_i \in \{1, -1\}$

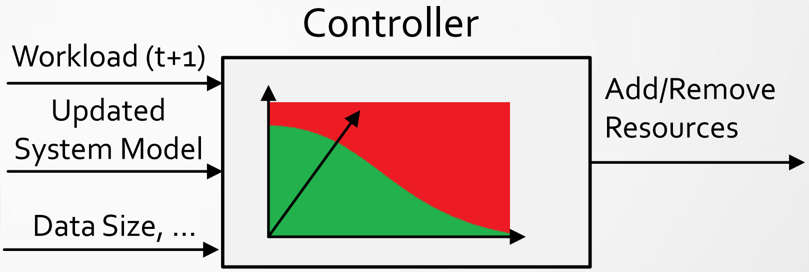


3D Model


Satisfy SLO
Violate SLO
Model
Training Data and Model
Data Size (KB)
10
5
0
1500
1000
500
0
Writes per second
0
200
400
600
800
1000
Reads per second

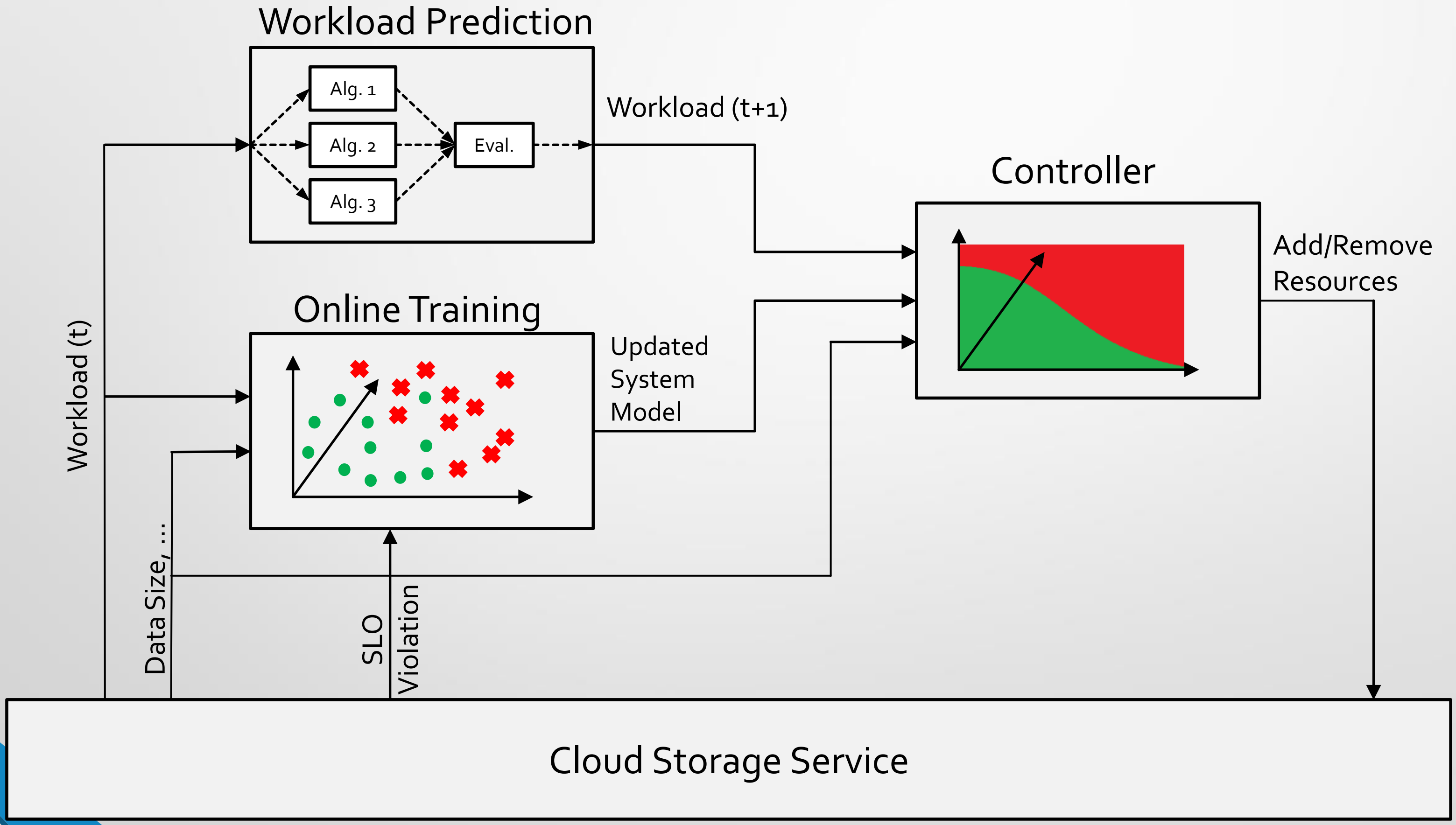

Satisfy SLO
Violate SLO
Model
Training Data and Model
1500
1000
500
0
Writes per second
0
200
400
600
800
1000
Reads per second

Elasticity Controller



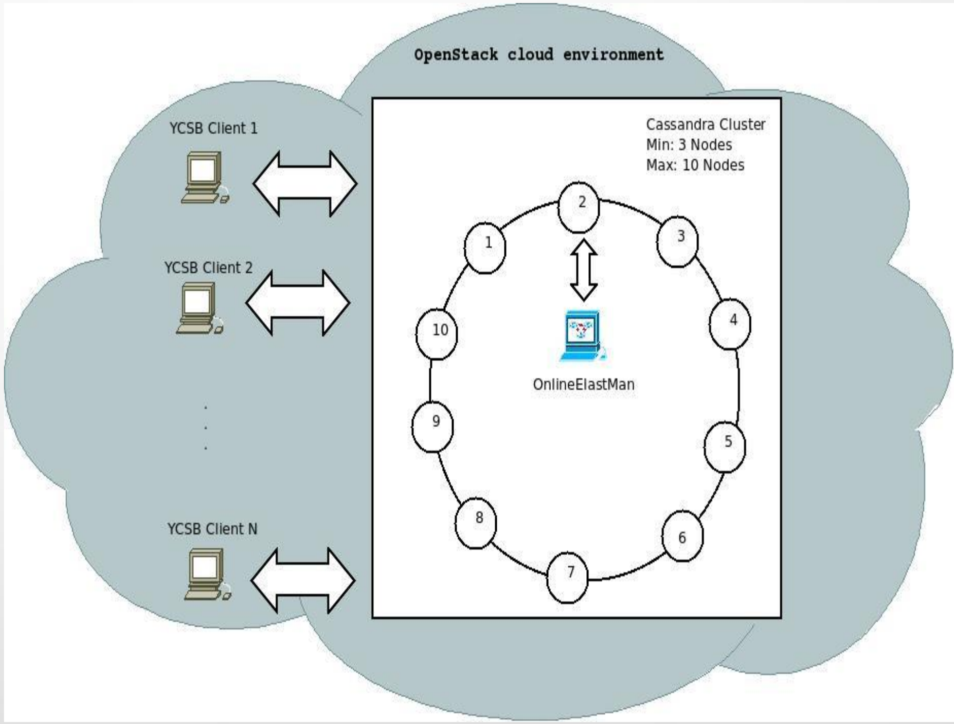
- Reads the predicted workload and other system parameters
- Use the system model to make scaling decisions (add/remove resources)
  - Calculate available capacity for VMs
- The system model is continuously updated to adapt to changes
- Keep SLO at the desired level

Overall Controller Architecture (Revisisted)

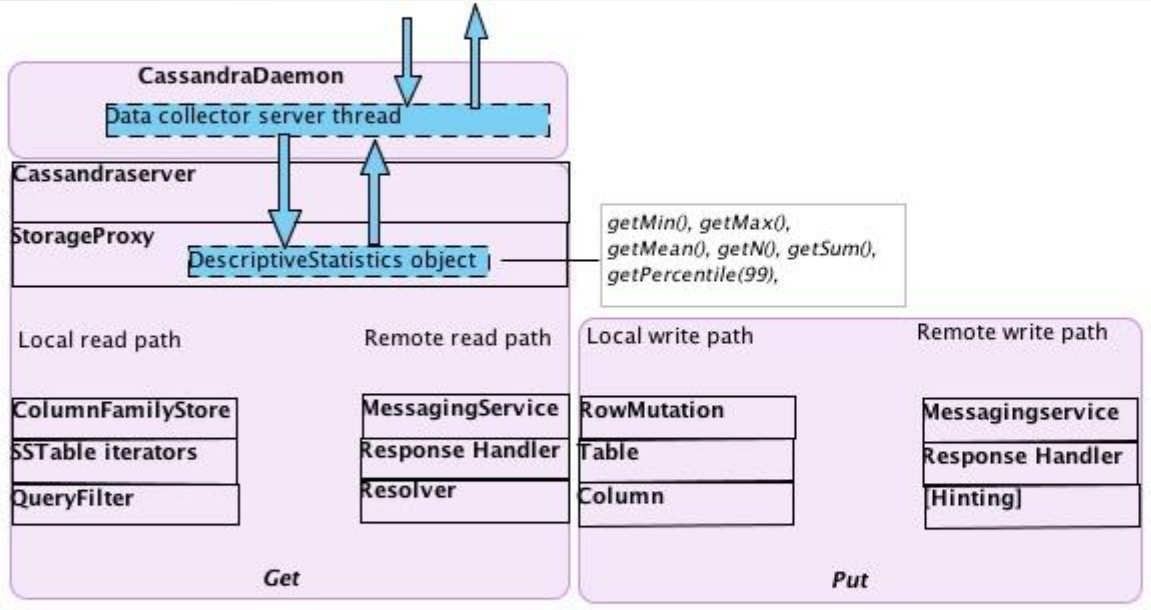


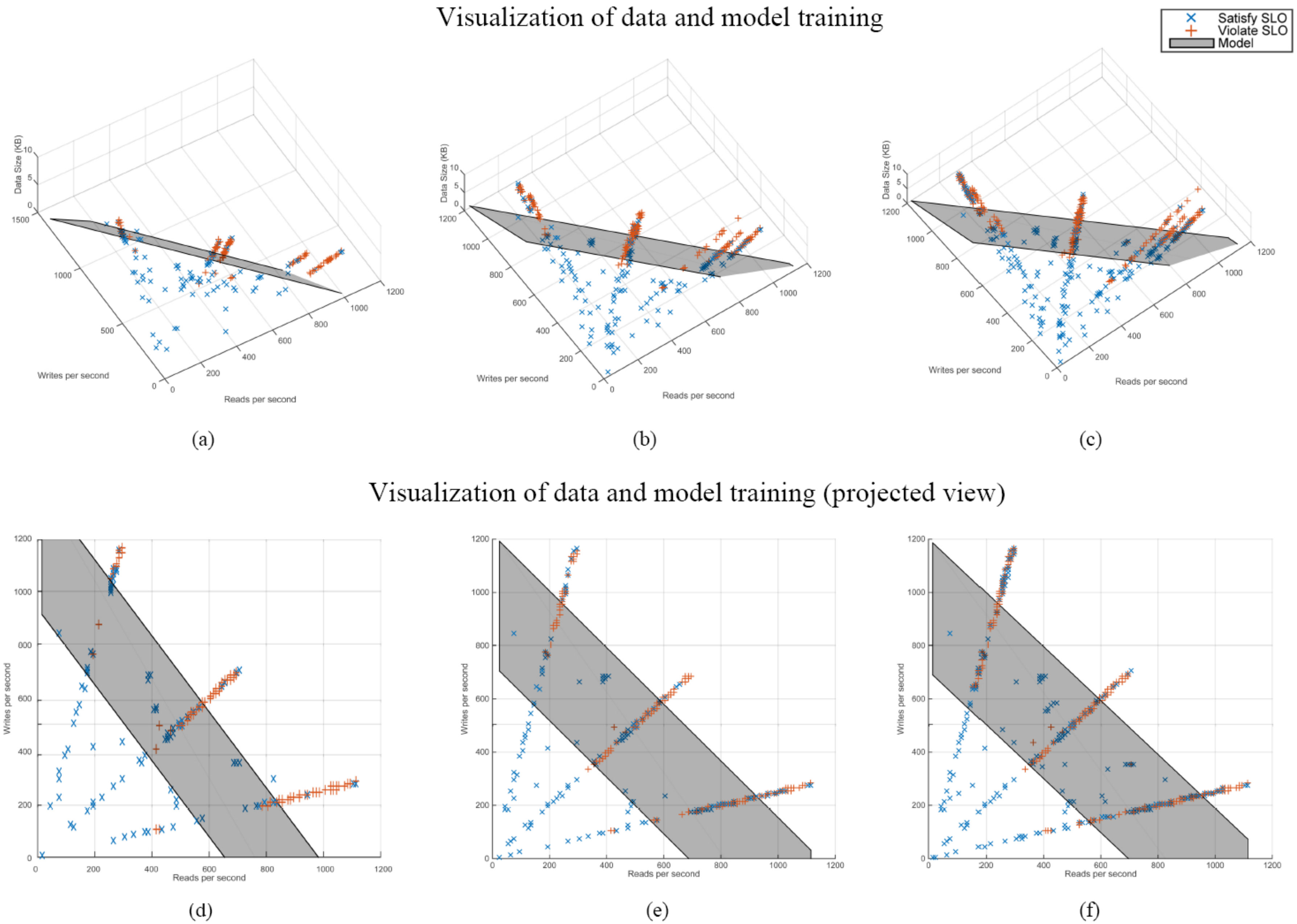
Evaluation

- Private OpenStack Cloud
  - VMs with 2 cores, 4GB ram, 40 GB disk
- Cassandra key-value store
- Workload generated using YCSB



Instrumentation in Cassandra

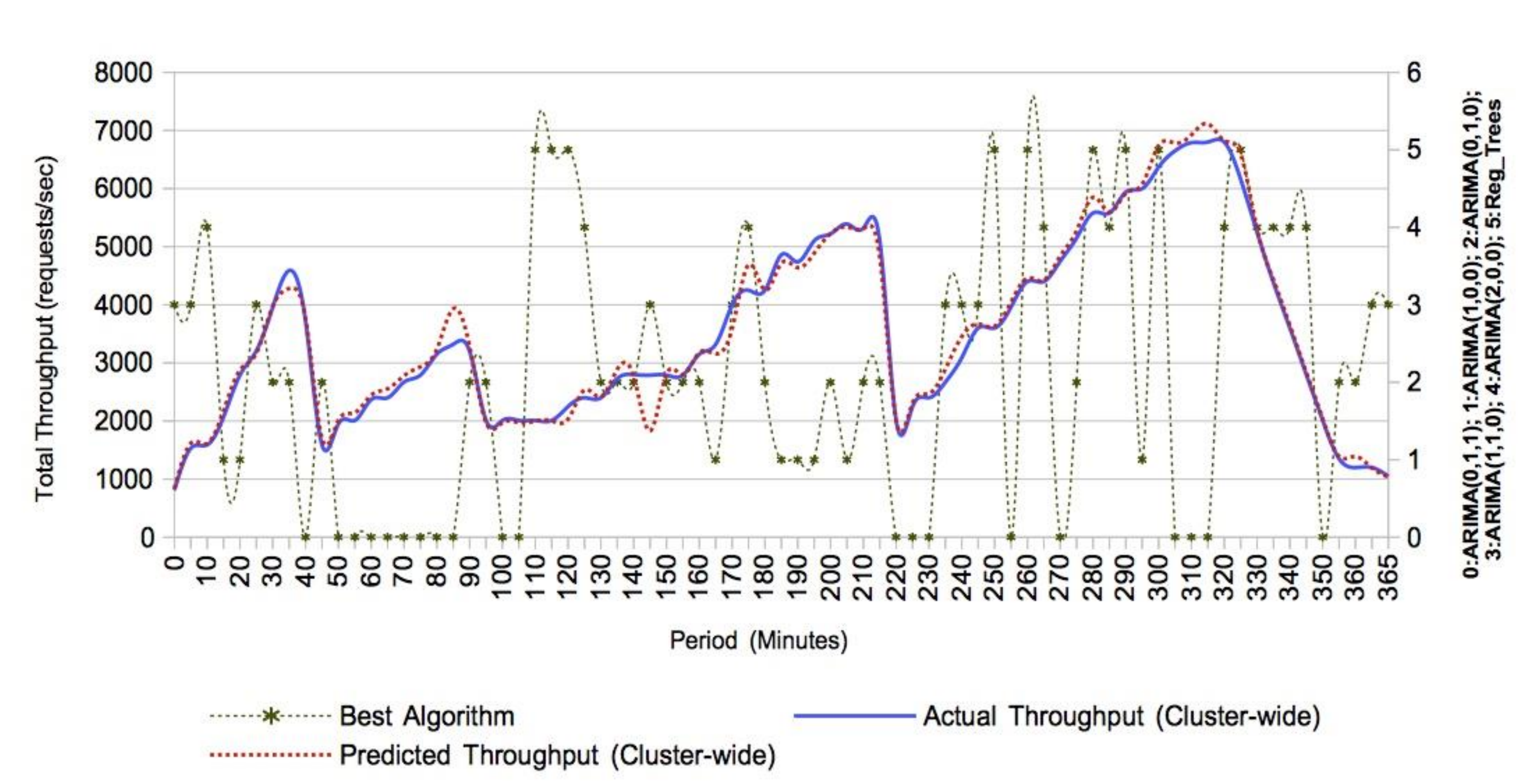



Visualization of data and model training
Satisfy SLO
Violate SLO
Model
Data Size (KB)
10
5
0
1500
1000
500
0
Writes per second
0
200
400
600
800
1000
1200
Reads per second
Data Size (KB)
10
5
0
1200
1000
800
600
400
200
0
Writes per second
0
200
400
600
800
1000
1200
Reads per second
Data Size (KB)
10
5
0
1200
1000
800
600
400
200
0
Writes per second
0
200
400
600
800
1000
1200
Reads per second
Visualization of data and model training (projected view)
Writes per second
1200
1000
000
600
400
200
0
0
200
400
600
800
1000
1200
Reads per second
Writes per second
1200
1000
800
600
400
200
0
0
200
400
600
800
1000
1200
Reads per second
Writes per second
1200
1000
800
600
400
200
0
0
200
400
600
800
1000
1200
Reads per second

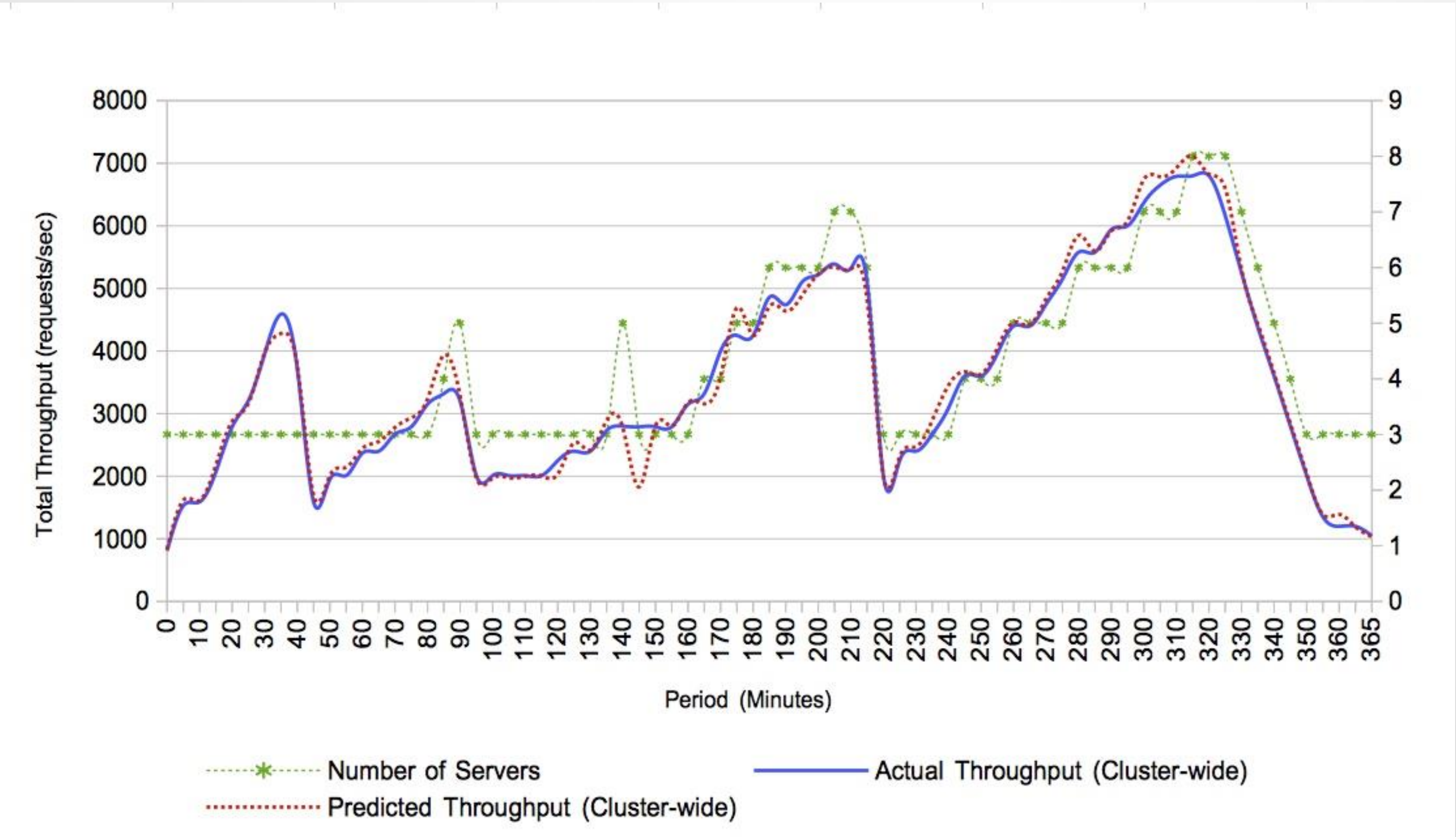
(a) (b) (c)

(d) (e) (f)

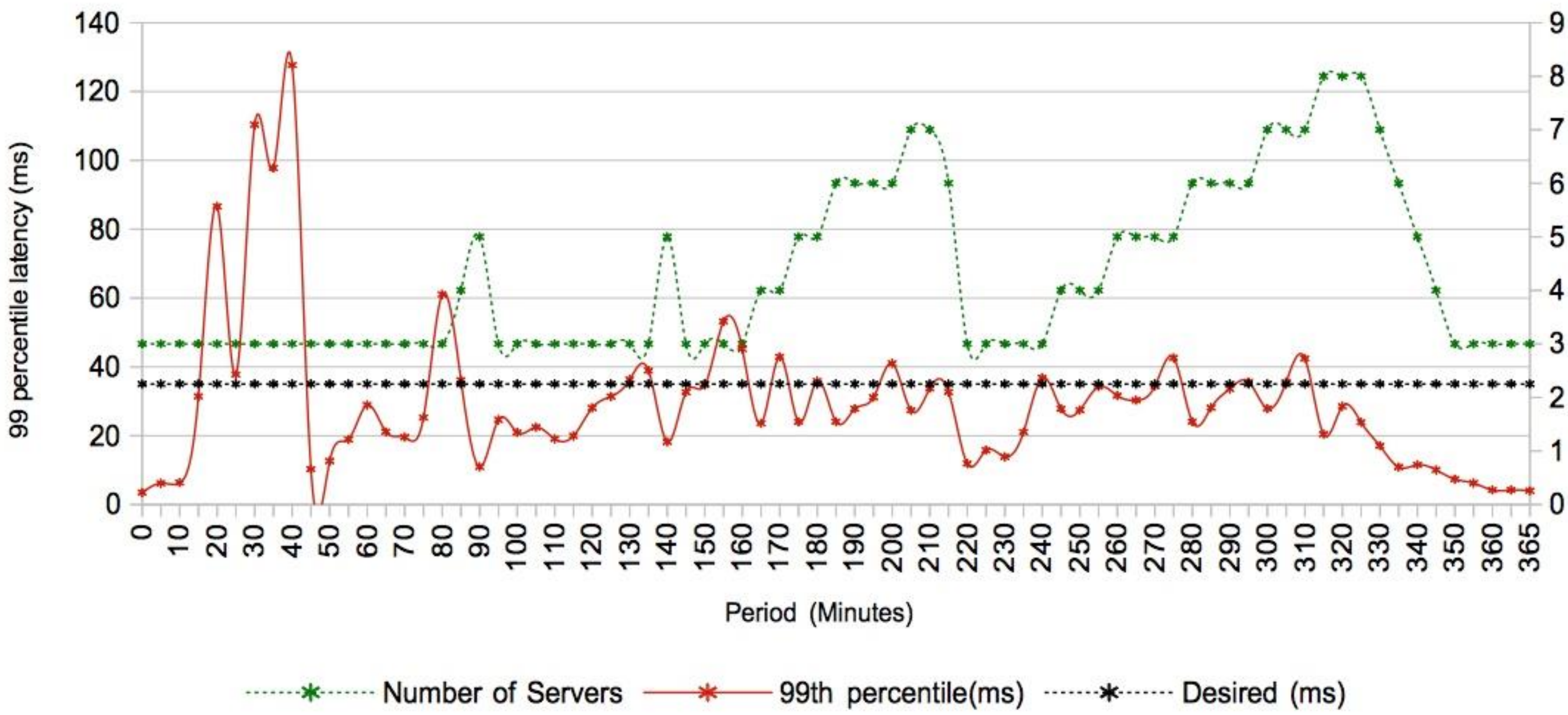
Workload Prediction and Weighted Majority Algorithm



Automatic Resource Provisioning



Performance Evaluation



Conclusions

- Elasticity controller for Cloud storage services
- Self-trained multidimensional performance model
- Self-tuning workload prediction module
- Pluggable modular architecture
- Prototype evaluated on Apache Cassandra