Bringing Introspection into BlobSeer

Towards Self-Adaptative Cloud Storage

Alexandru Costan

DataCloud@work Associate Team, UPB / KerData

30 November 2010

Outline

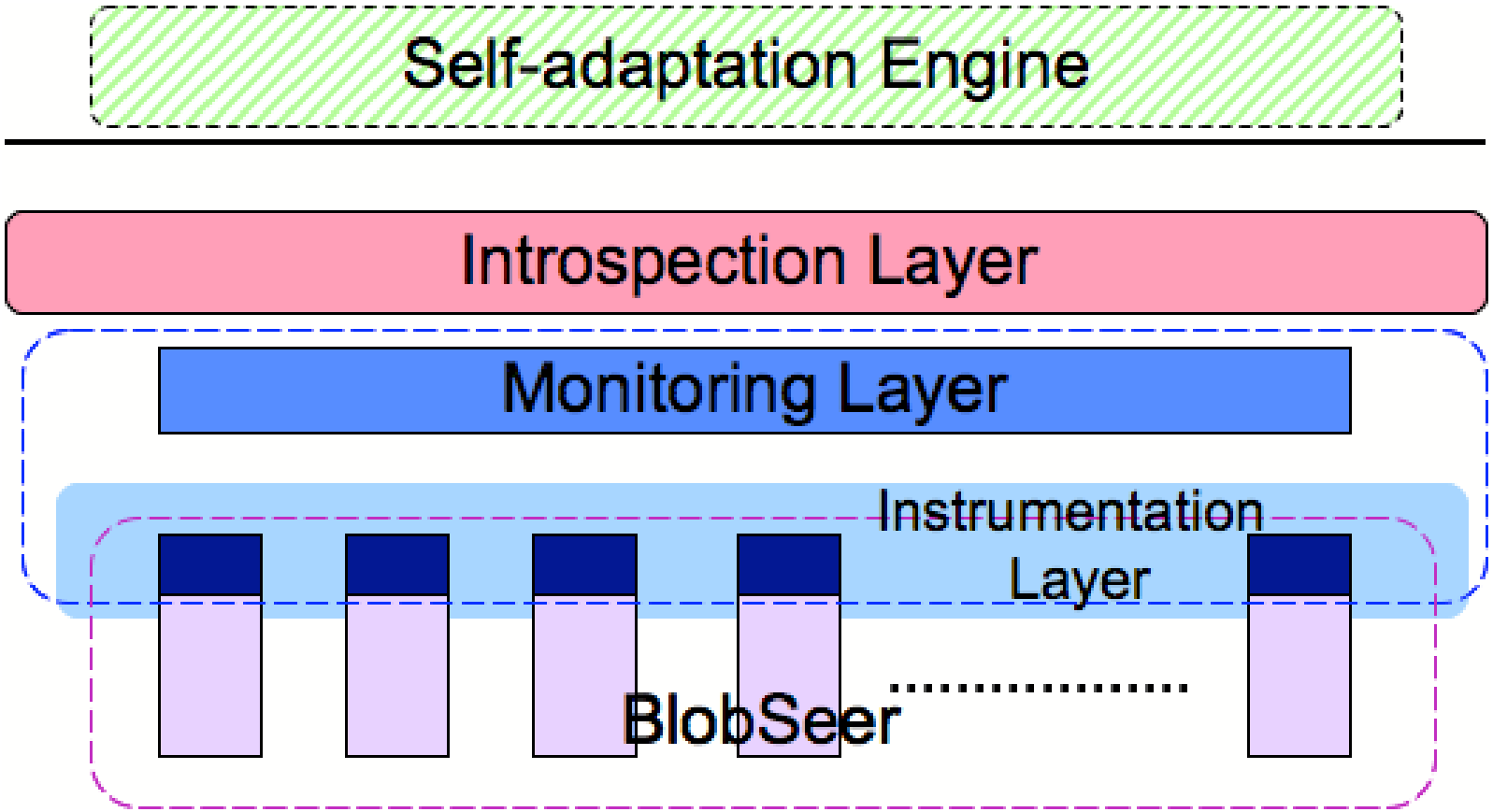
Introspection in large-scale distributed systems

- Exposes the system behavior accurately and in real time
- Relies on monitoring tools:
  - Provide global and specific knowledge of the state of the system and the underlying infrastructure
- Enables the autonomic behavior

Challenges

- Identifying the relevant data
- Adapting a general-purpose monitoring system

Towards an introspective BlobSeer



Relevant monitoring data

General data:

- CPU load, used/available memory, storage space, network traffic, disk usage
- *Aggregated data:* total storage space occupied / available

Individual BLOB-related data:

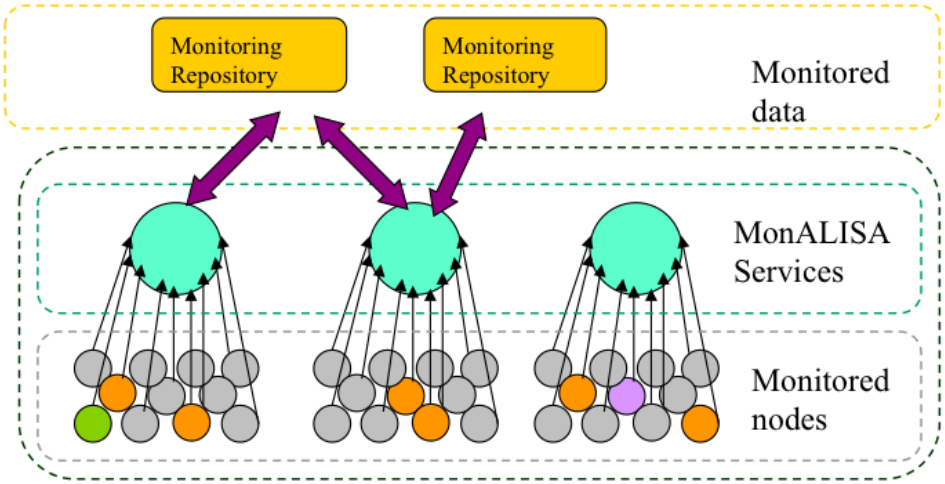
- Access patterns: number of READ/WRITEs from/to the BLOBs

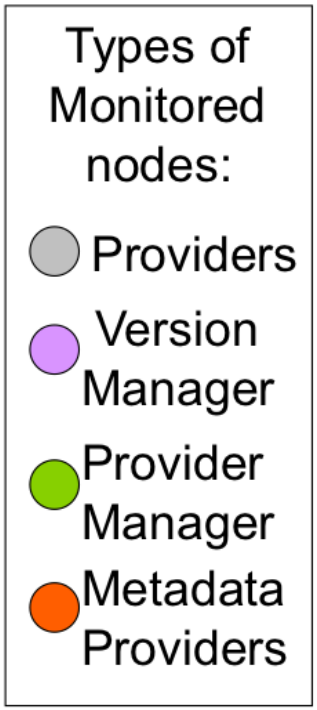
Global state:

- Total number of accesses on providers
- Distribution of the BLOBs across providers

Monitoring instrumentation

- Based on the **MonALISA** monitoring framework and the **ApMon** instrumentation library
- Custom *monitoring modules*, *data filters*, *aggregators*
- Distributed monitoring repositories

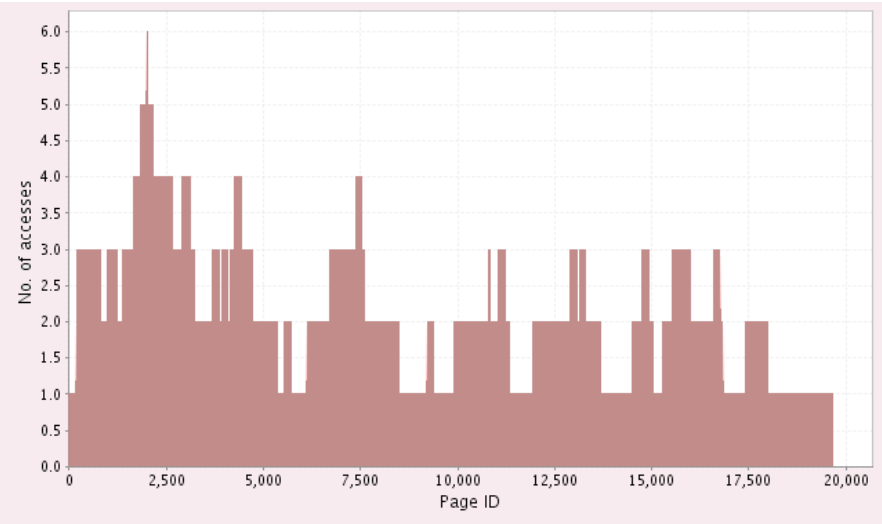




Experimental evaluation on Grid'5000

Goals:

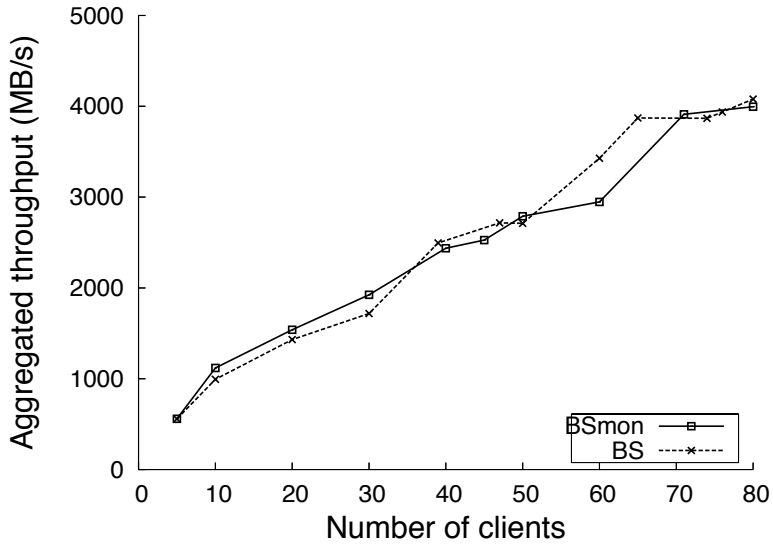
- Monitoring as a visualization tool for BlobSeer-specific data
- Impact of the introspection architecture on the Blobseer data-access performance



Experimental evaluation on Grid'5000

Goals:

- Monitoring as a visualization tool for BlobSeer-specific data
- Impact of the introspection architecture on the Blobseer data-access performance



Autonomic properties of BlobSeer using introspection

- **Self-configuration** through dynamic dimensioning
- **Self-protection** through malicious clients detection
- **Self-healing** through adaptive data-replication strategies
- **Self-optimization** through state-dependent allocation algorithms for storage providers

Autonomic properties of BlobSeer using introspection

- **Self-configuration** through dynamic dimensioning
- **Self-protection** through malicious clients detection
- **Self-healing** through adaptive data-replication strategies
- **Self-optimization** through state-dependent allocation algorithms for storage providers

Dynamic providers deployment

Goal:

- Enable BlobSeer to **scale up** and **down** automatically

Motivation:

- Cloud Computing - pay-per-use model
- Optimize resource consumption

Challenges

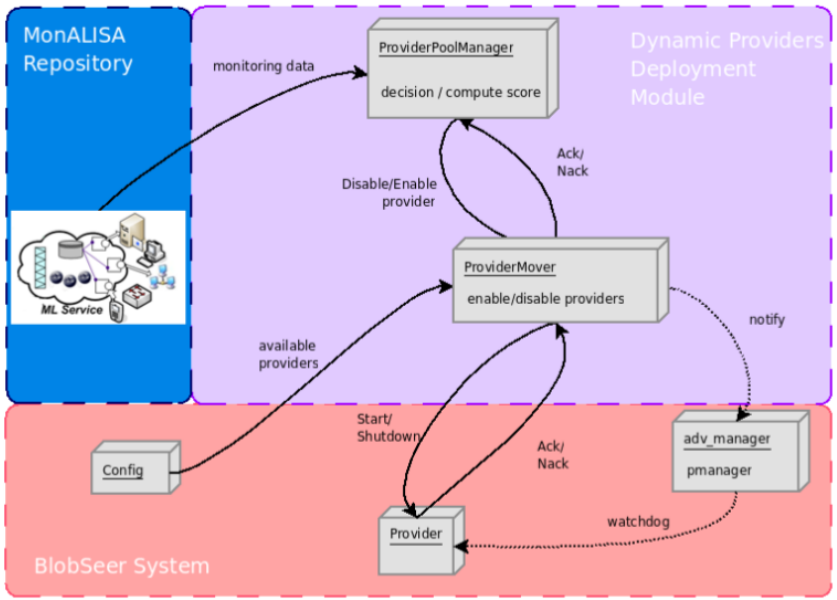
- Finding the optimal number of resources
- Maintaining data integrity when scaling down

Architectural Overview

Maintains two pools of providers: **active** and **backup**.

Two steps:

- Decision taking: *compute a score* for each provider based on the *introspection layer*
- Decision enforcement: *enable* or *disable* providers



Providers evaluation

Heuristics for computing the providers' score

- Factors: **general data** (storage space, bandwidth usage) and **global state** (number of accesses)
- Weights associated with factors
- Decision based on thresholds

Framework for specifying the scenarios defining the scoring algorithm

- **Flexible**: select factors, define conditions for the factors' values, time intervals
- **Extensible**: define new scenarios

Example of a scenario:

- free disk space is above the 70% threshold
- read access rate per time unit is small
- write access rate per time unit is small
- the provider can be shut down

Providers evaluation

Heuristics for computing the providers' score

- Factors: **general data** (storage space, bandwidth usage) and **global state** (number of accesses)
- Weights associated with factors
- Decision based on thresholds

Framework for specifying the scenarios defining the scoring algorithm

- **Flexible**: select factors, define conditions for the factors' values, time intervals
- **Extensible**: define new scenarios

Example of a scenario:

- free disk space is above the 70% threshold
- read access rate per time unit is small
- write access rate per time unit is small
- the provider can be shut down

Malicious clients detection

Goals:

- Enable the detection of malicious clients for large scale data management systems
- Develop a complete security solution for BlobSeer

Motivation:

- Exposing BlobSeer as a Cloud service

Challenges

- Maintaining a high throughput rate relies on direct connections between clients and data providers

Types of malicious activities

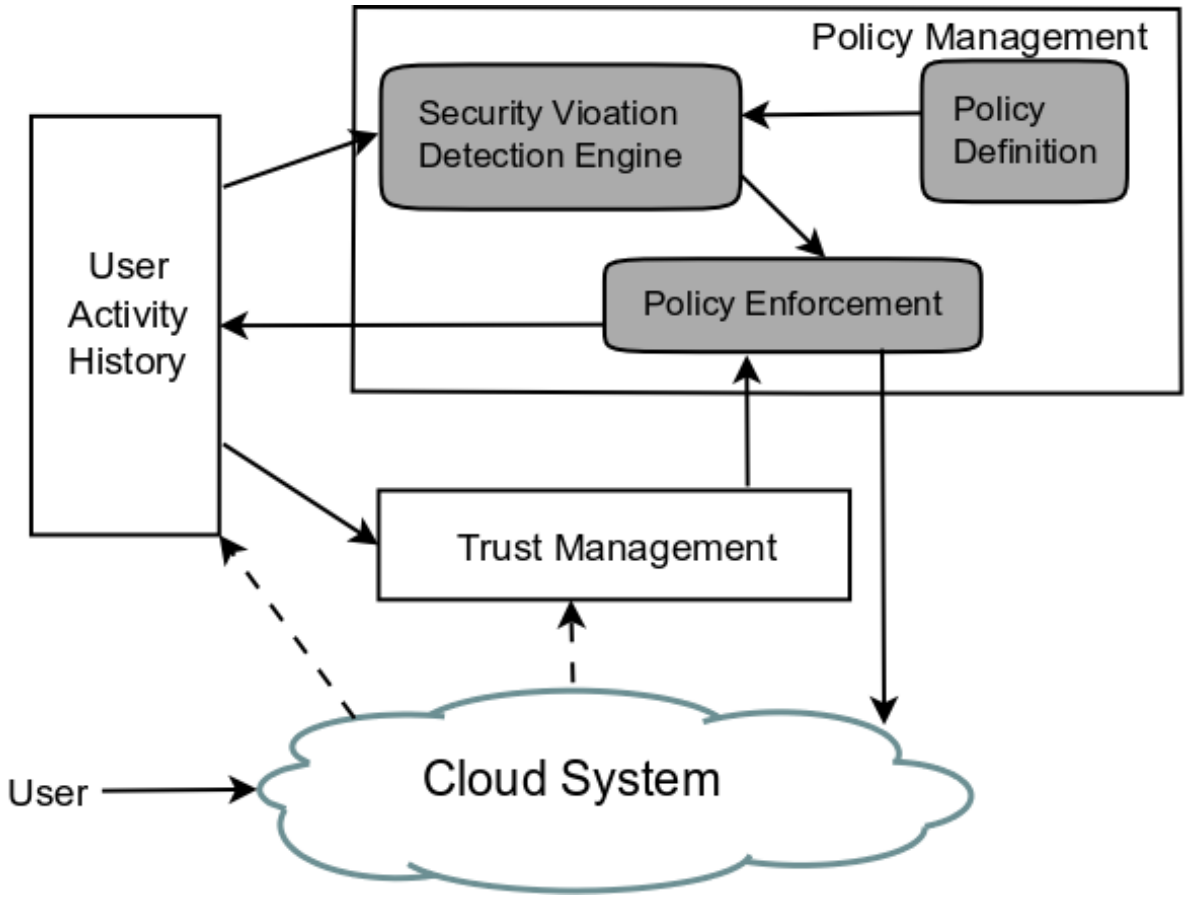
Protocol breach

- Heavy writing without the creation of a new version *(WriteNoPublish)*
- Publish the version and upate the metadata tree without writing to data providers *(PublishNoWrite)*

Matching of predefined attacks (scenarios)

- Denial of Service
- Detection of suspicious activity
- Crawling
- Repeated reading of some data
- Abnormal client activity

Architectural overview



Uniform representation for various types of security rules

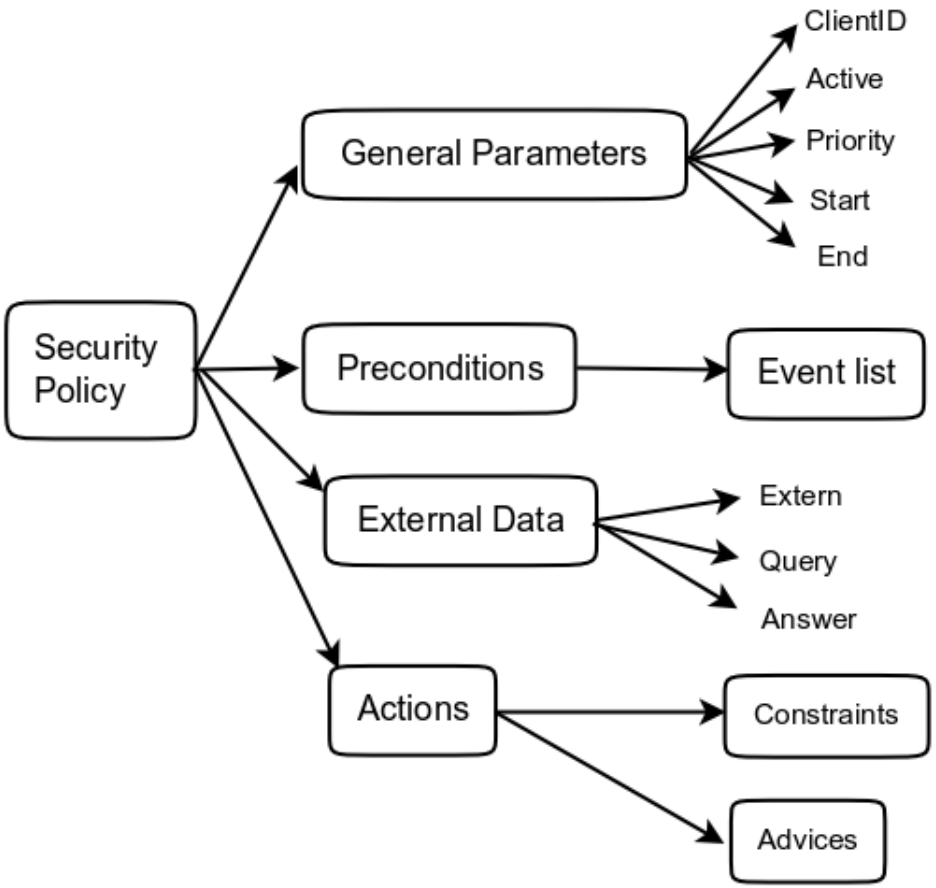
Security policies enforcement

Policies specification

- XML based descriptions for the targeted malicious activities
- *Extensible, customizable*
- **Events**: simple, composite

Policies enforcement

- Automatically execute **predefined actions** in case of policy violations:
  - Directly
  - Using the **Trust Level**



Trust management

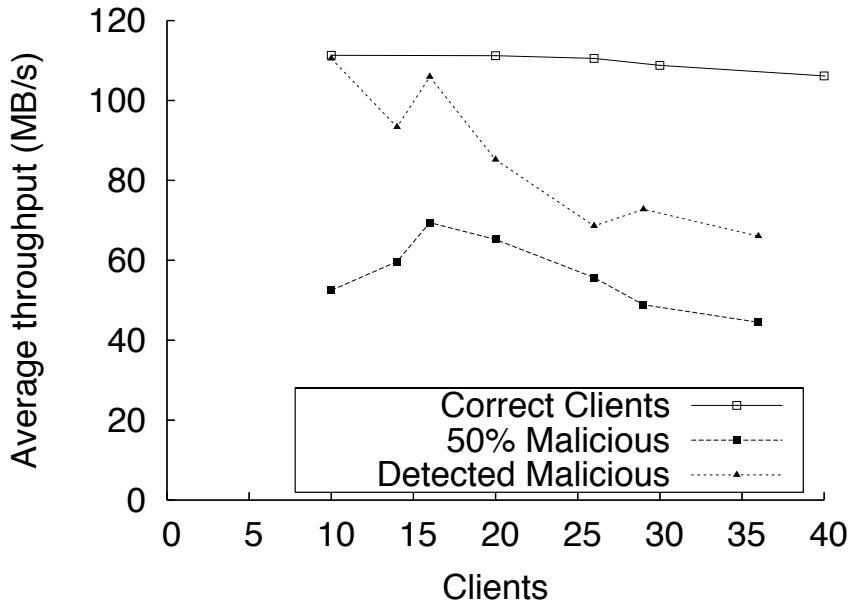
Trust Level

- Provides a trust value for each user based on his past actions
- Identifies a user as a **fair** or a **malicious** one
- Weighted using **age** and **system state**
- Dynamic adaptation:
  - *High values*: relaxed security policies for a period of time
  - *Low values*: restrictive policies

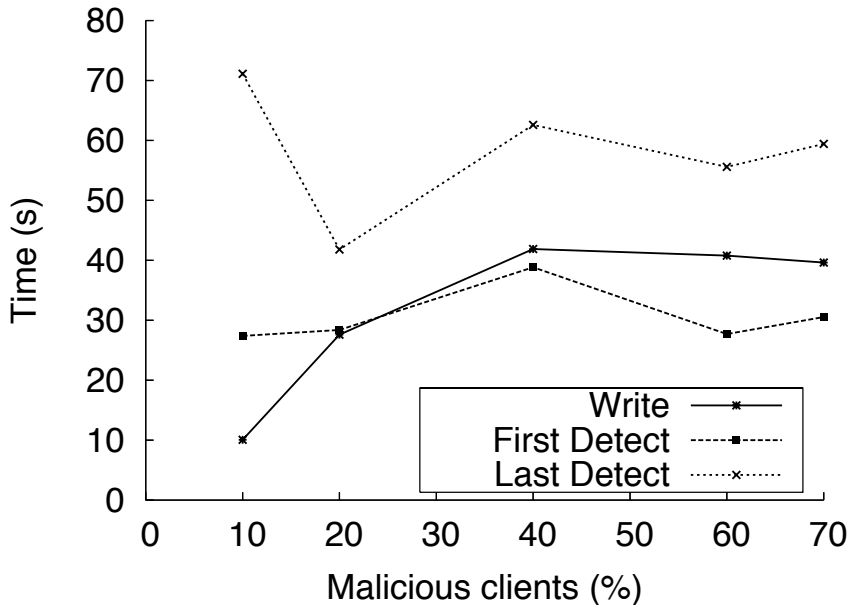
Policies adaptation

- Customization through a set of predefined rules
- Rules specify which parameters of the policies can be modified and in what extent

Experimental evaluation on Grid'5000



Impact of concurrent DoS attacks



Detection delay

Summary

- Introspection on top of a data-management system
- 3-layered architecture:
  - **Instrumentation** layer
  - Monitoring layer
  - Introspective layer
- Instrumentation:
  - Dynamic dimensioning of data providers
  - Malicious clients detection

Outlook

- Autonomic behavior in BlobSeer
  - New allocation schemes for the data providers
  - Support for data deleting from BlobSeer: enables the adaptive replication and the protection from malicious clients
- Adaptive security management
  - Authentication and authorization mechanisms for BlobSeer users
  - Anonymization for privacy preservation
  - Adaptive security policies that take into account the past actions of each user

Consistent with the Associate Team work programme for 2011.

Contributions

- People involved:
  - Master students: *Mihaela Vlad, Cristina Basescu (UPB)*
  - PhD Students: *Alexandra Carpen-Amarie (KerData), Alexandru Costan, Catalin Leordeanu (UPB)*
  - Supervisors: *Gabriel Antoniu, Luc Bougé (KerData), Valentin Cristea (UPB)*
- Papers published at *CISIS2010, AINA2011*, in *IJAMCS* (under review).