

Leveraging **System Performance Metrics** and **Execution Logs** to Proactively **Diagnose** System of Systems **Performance Issues**

SOFTCOM-INT

26/01/2016

Issue/Revision: .

Reference:

Status:

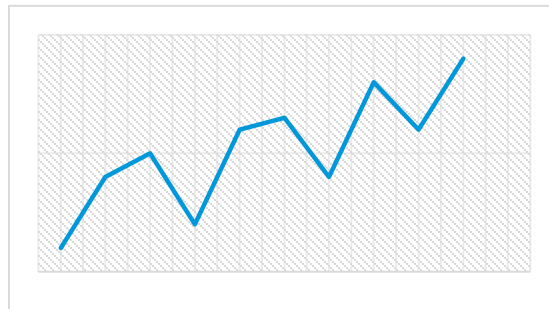
ESA UNCLASSIFIED - For Official Use

The problem

Software systems might misuse **resources** (e.g. memory)

Performance analysts may run **load tests** in order to **expose** the problem

They may plot **resource usage** over time in order to **detect** the problem



They need to manually **review logs** and **resource usage** in order to **report** the problem to **developers**

Leveraging Performance Counters and Execution Logs to Diagnose Memory-Related Performance Issues

Published in 2013 by M. Syer, Z. Jiang, M. Nagappan, A. Hassan, M. Nasser and P. Flora at the International Conference on Software Maintenance

Proposes an automated approach that combines **performance counters** and **execution logs** to **diagnose** memory-related performance **issues** that appear in **load tests**

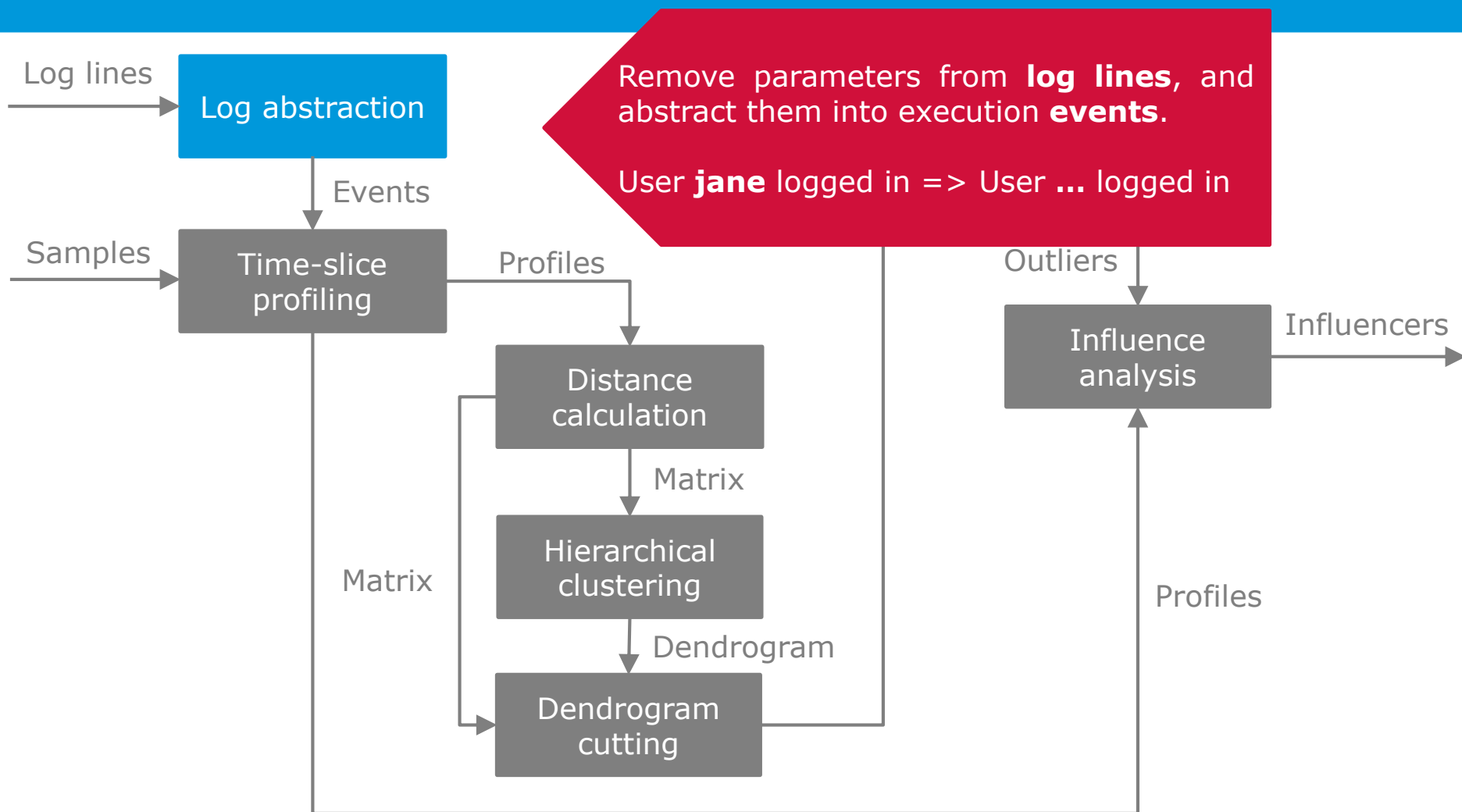


Identical system activity has **identical impact** on system resource usage.

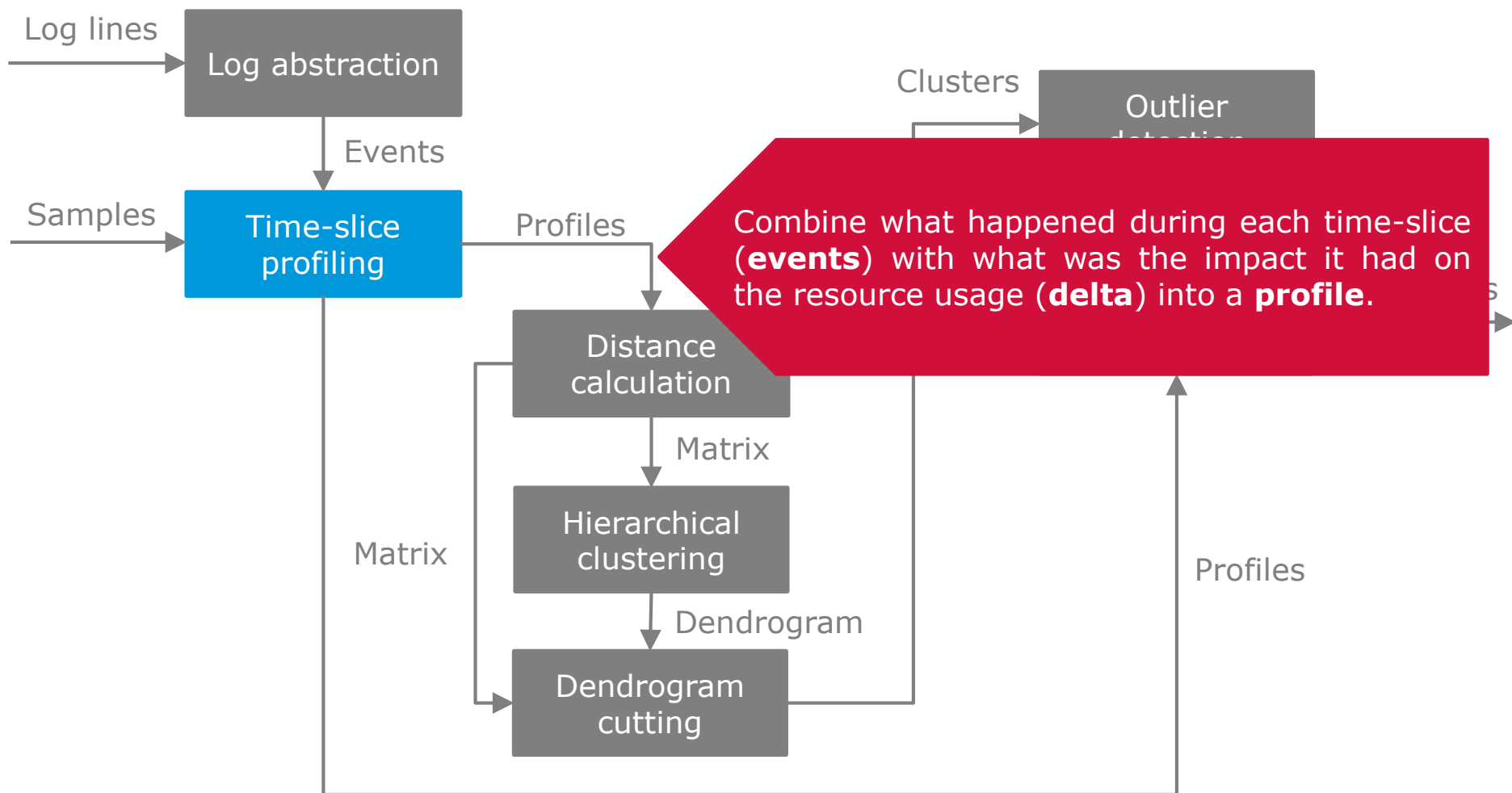
Similar system activity should have **similar impact** on the system resource usage.

Whatever **invalidates** that, might be the **source** of **performance issues**.

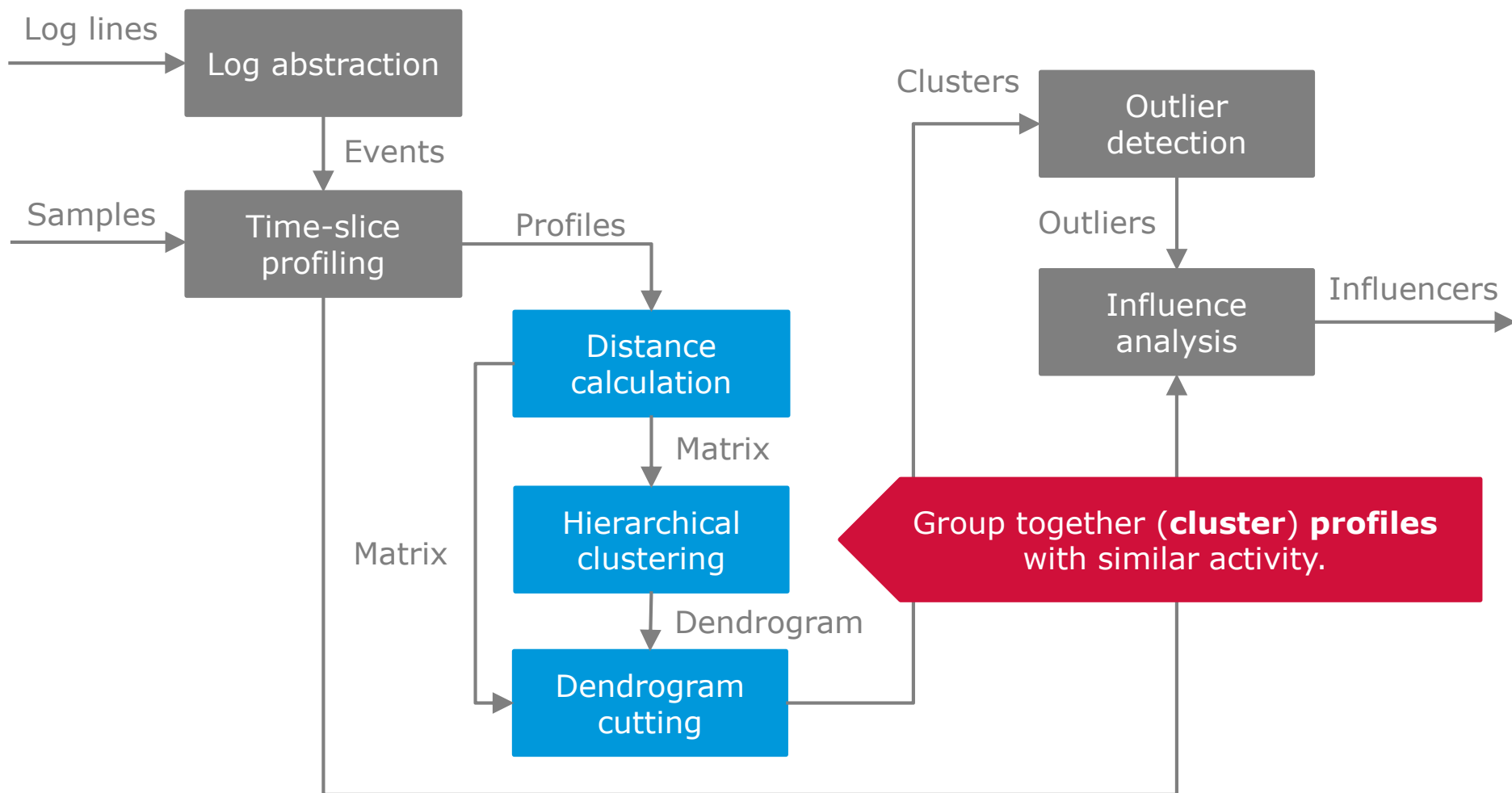
The approach



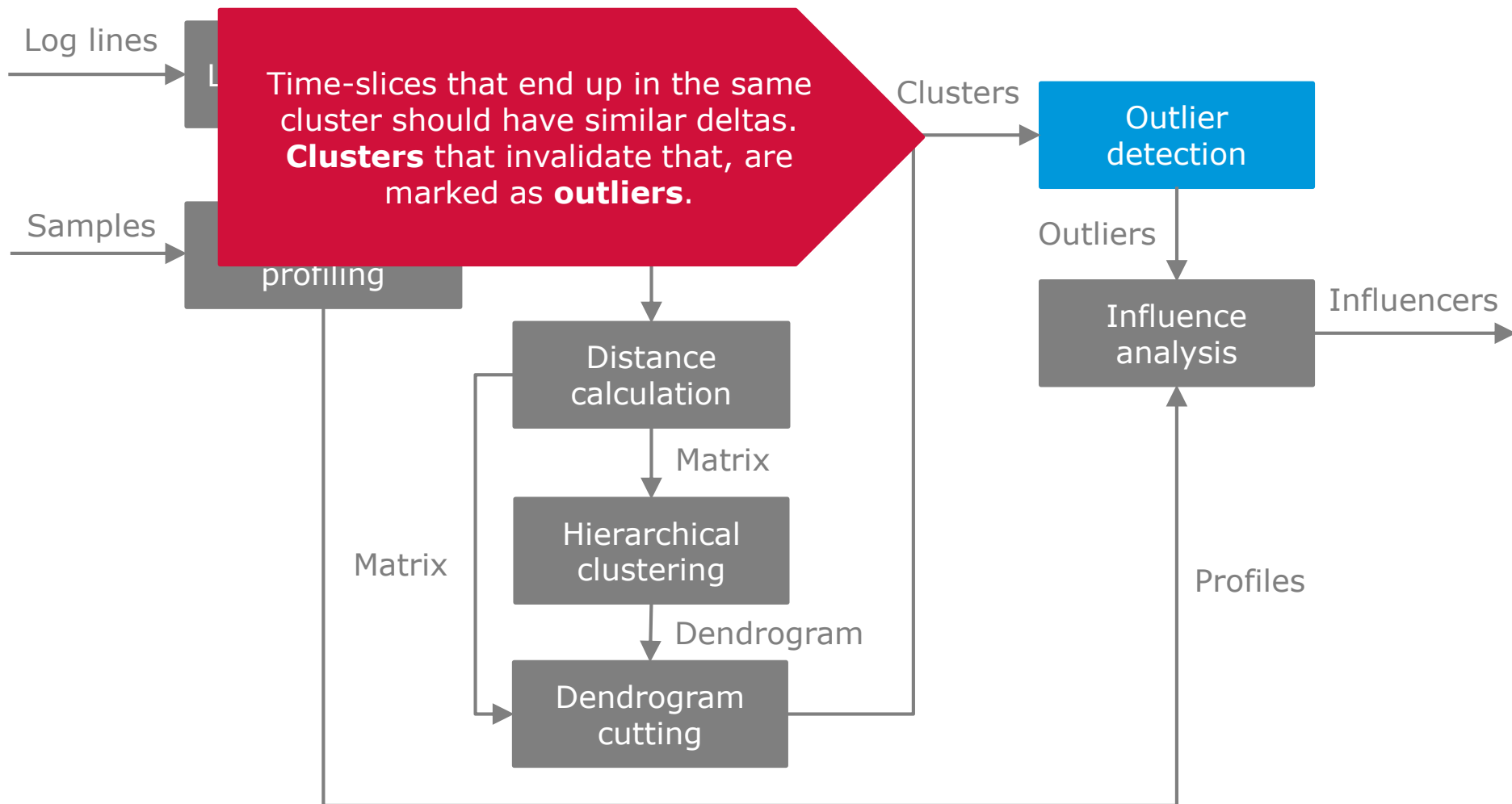
The approach



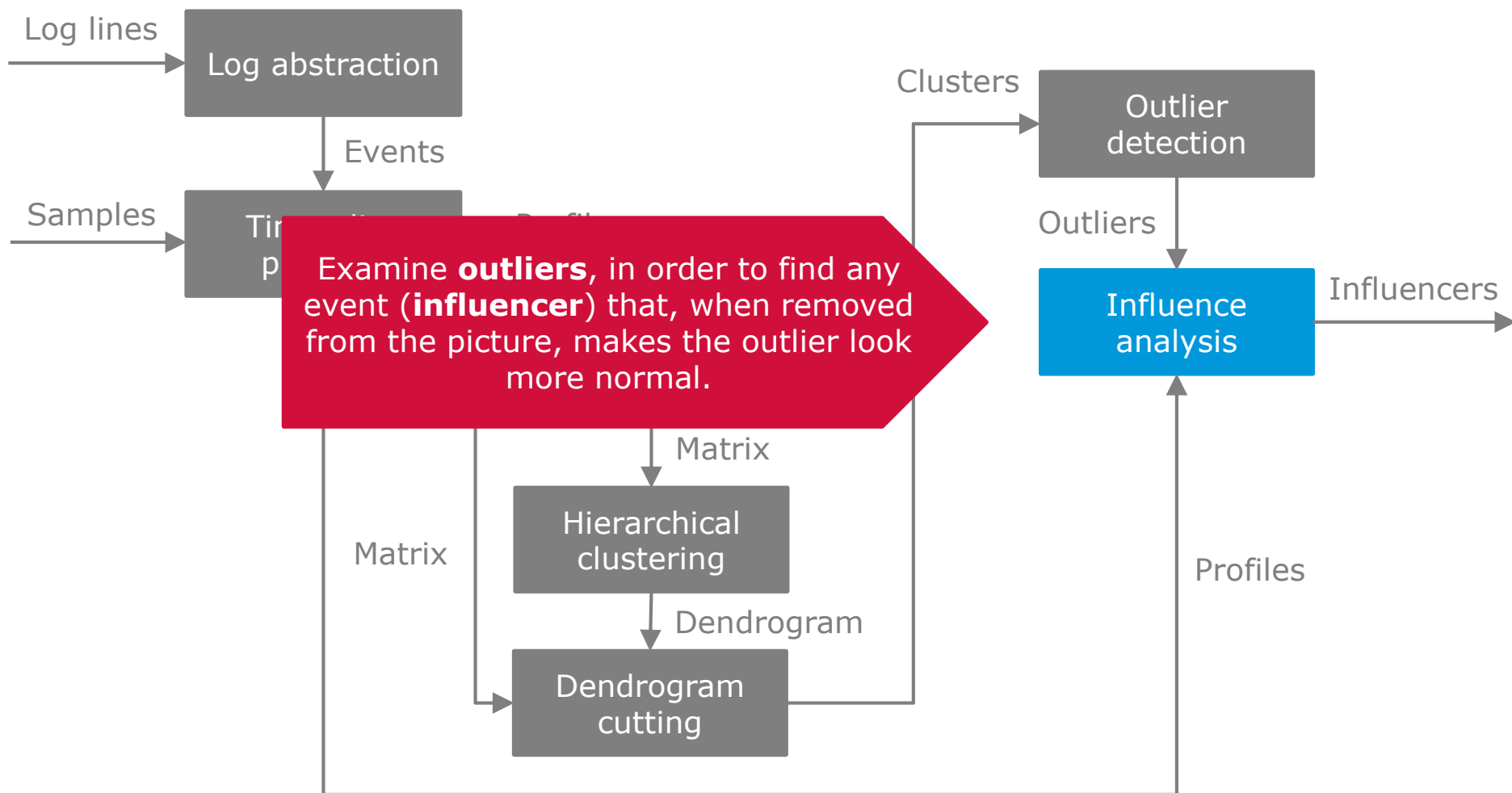
The approach



The approach



The approach



Leveraging System Performance Metrics and Execution Logs to Proactively Diagnose System of Systems Performance Issues

Analyse the approach proposed by Syer et al.

Build a **prototype correlation engine** based on that approach

Analyse the **suitability** of **logs** produced by HSO-GI software

Apply the correlation engine on **simulation applications**

Apply the correlation engine **HSO-GI software**

Apply the correlation engine on other **real-world applications**

Build a **proactive error detection system prototype** based on that approach

The performance counter monitor

An **extensible** tool that monitors **performance counters** using other external tools

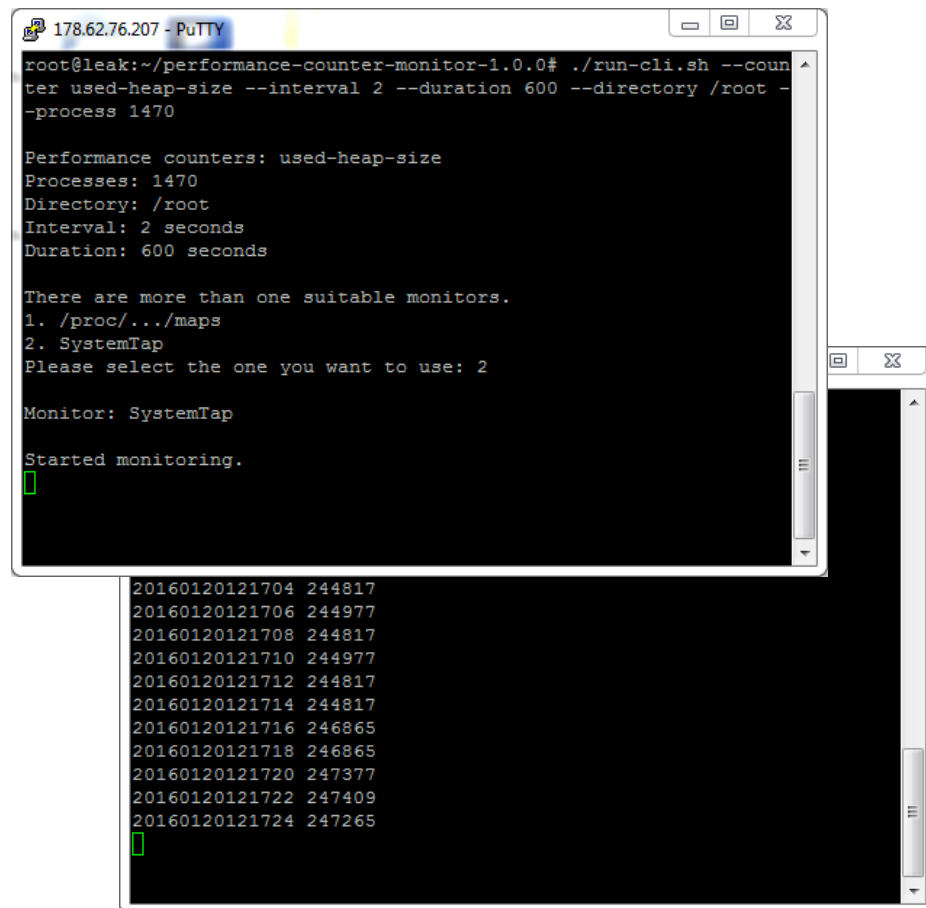
Selects automatically **the most suitable** external tool to use

Can be extended to support **more performance counters**

Now: Used heap size

Can be extended to use **more external tools**

Now: DTrace, JMX, /proc/id/maps, SystemTap



```
178.62.76.207 - PuTTY
root@leak:~/performance-counter-monitor-1.0.0# ./run-cli.sh --counter used-heap-size --interval 2 --duration 600 --directory /root --process 1470

Performance counters: used-heap-size
Processes: 1470
Directory: /root
Interval: 2 seconds
Duration: 600 seconds

There are more than one suitable monitors.
1. /proc/.../maps
2. SystemTap
Please select the one you want to use: 2

Monitor: SystemTap

Started monitoring.
[ ]

20160120121704 244817
20160120121706 244977
20160120121708 244817
20160120121710 244977
20160120121712 244817
20160120121714 244817
20160120121716 246865
20160120121718 246865
20160120121720 247377
20160120121722 247409
20160120121724 247265
[ ]
```

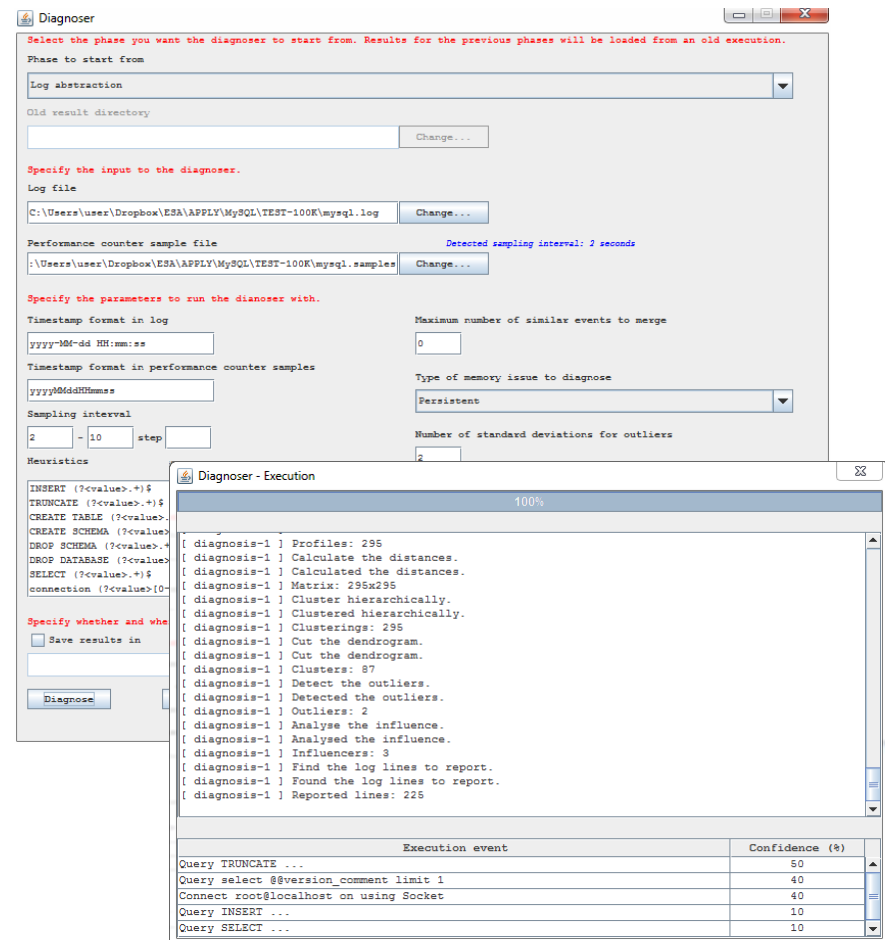
The prototype correlation engine



A tool that receives **execution logs** and **performance counter samples**, and diagnoses performance issues that are present in them based on the approach proposed by Syer et al.

Automatically tries **different parameter sets**

Calculates **confidence values**



The spiker and the leaker



The **spiker** is written in **Java** and causes **memory spikes**

The **leaker** is written in **C++** and causes **memory leaks**

They both produce **execution logs**

They are both **parameterisable** for exploration purposes

The demo



We ran **~7,000** load tests on the **spiker**

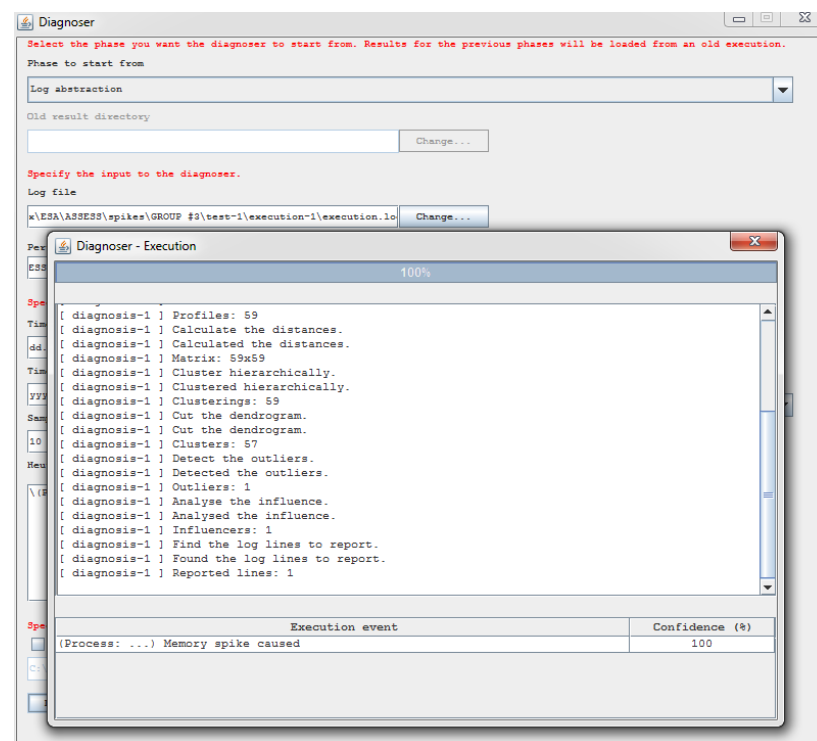
We ran **~3,000** load tests on the **leaker**

We ran **~10** load tests on **MySQL server**

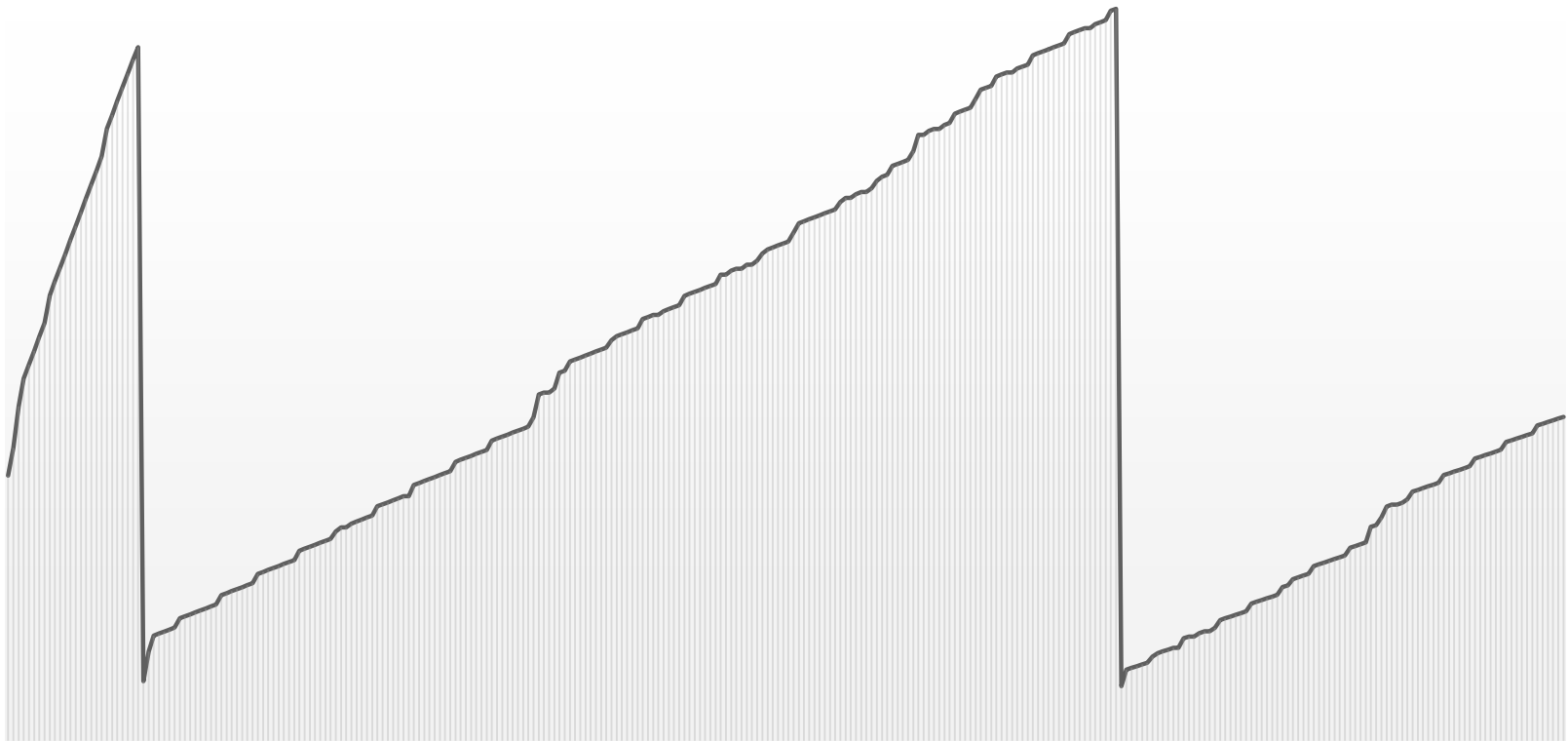
We ran **~10** load tests on **Apache Tomcat**

The engine and the spiker

Duration	10 minutes
Sampling interval	10 seconds
Spike size	1KB
Number of log lines	5,262
Number of samples	60
Flagged events	1
Reported lines	1
Precision (%)	100%

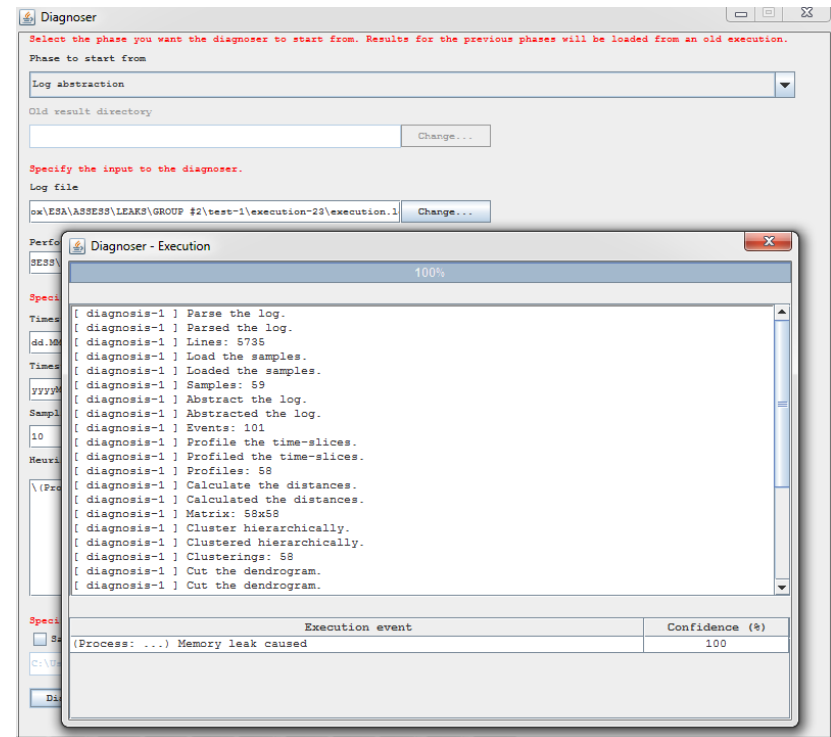


The engine and the spiker

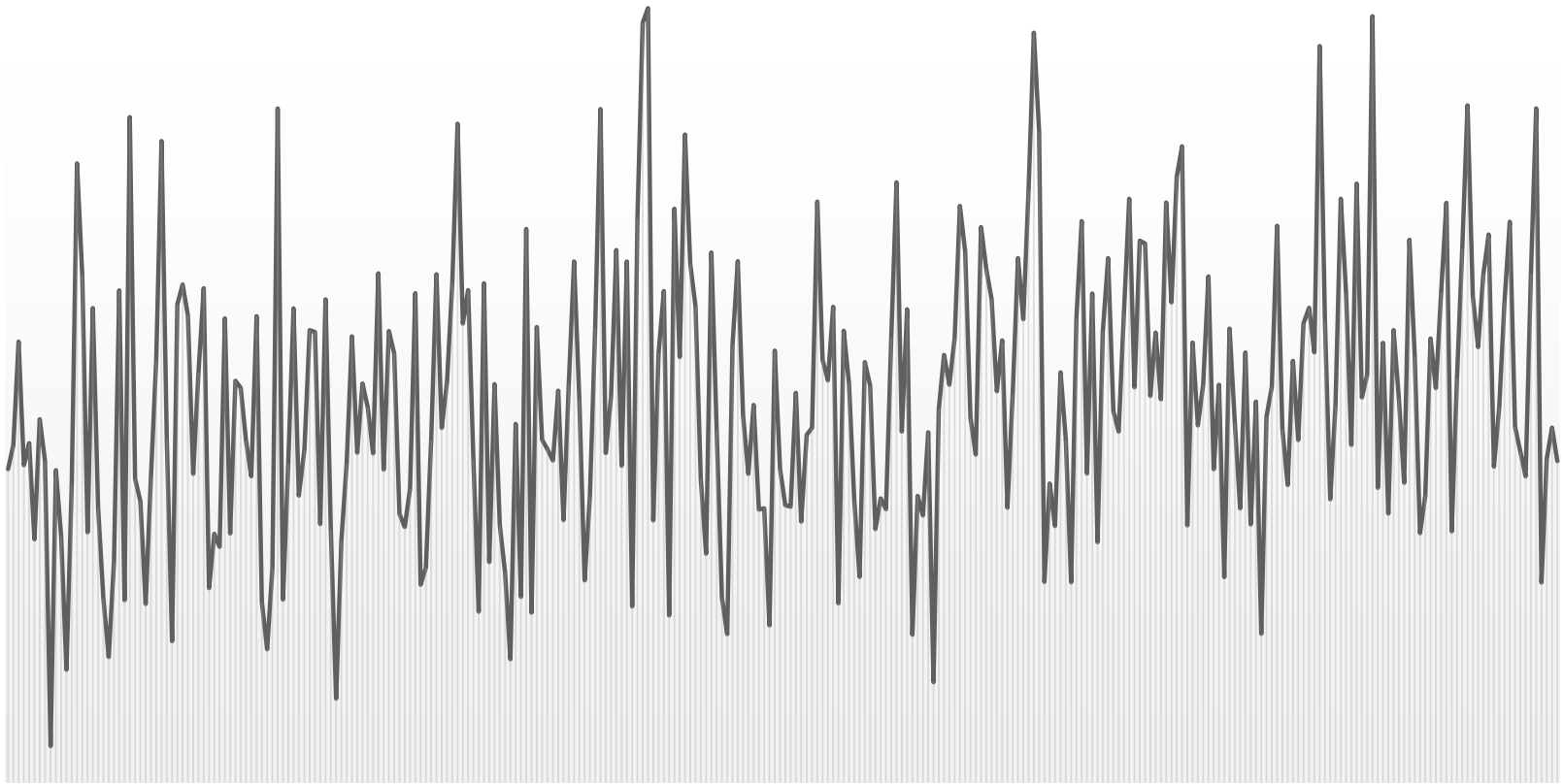


The engine and the leaker

Duration	10 minutes
Sampling interval	10 seconds
Leak size	100B
Number of log lines	5,735
Number of samples	59
Flagged events	1
Reported lines	1
Precision (%)	100%



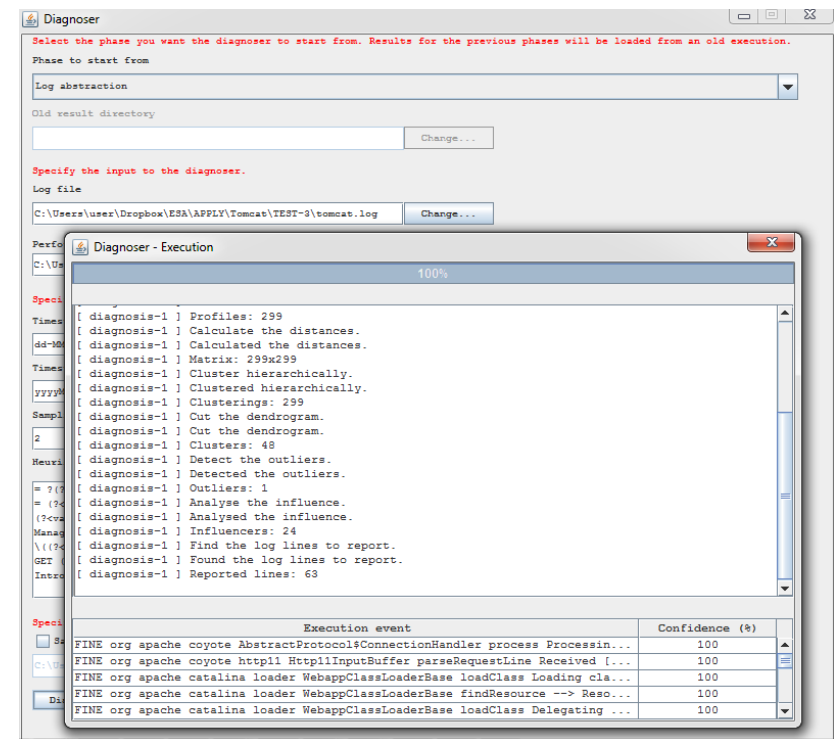
The engine and the leaker



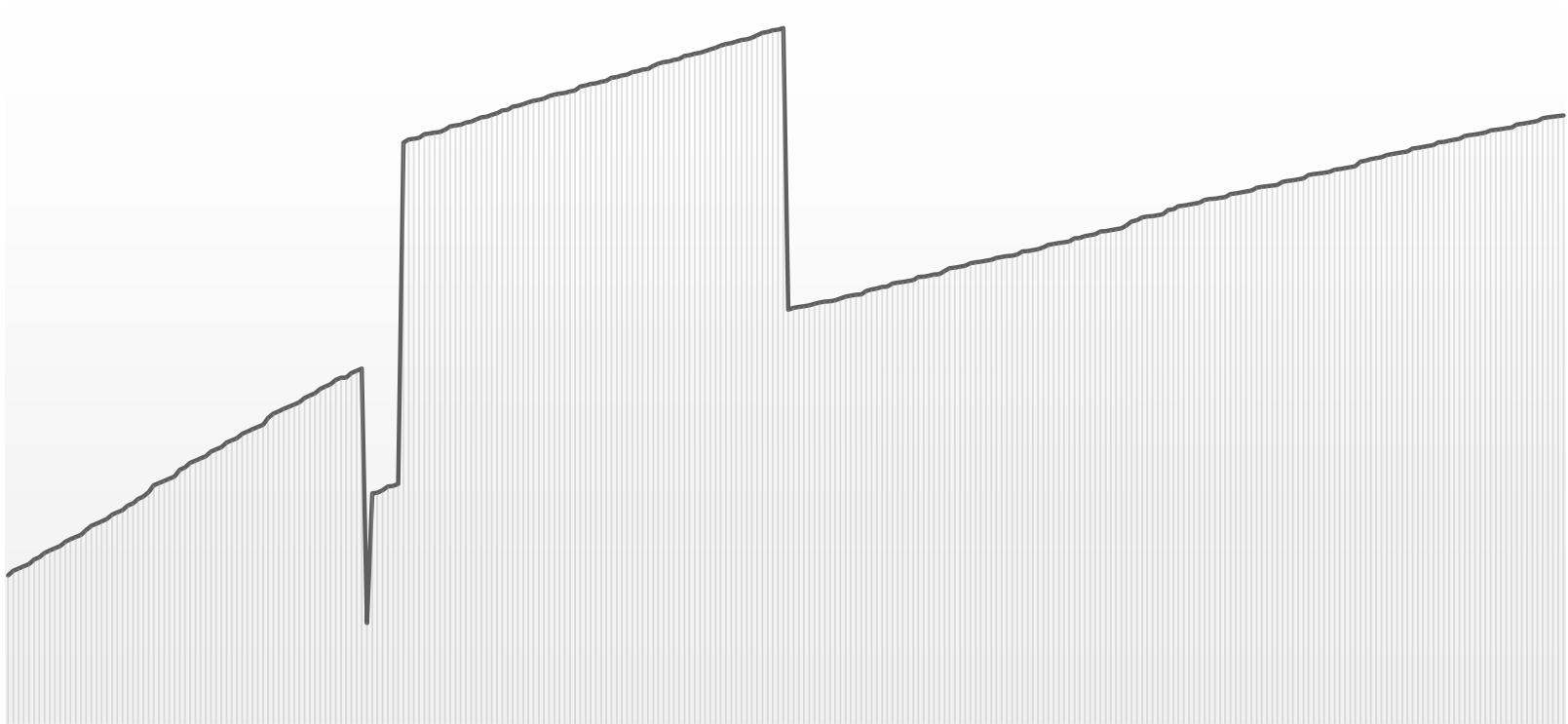
The engine and Apache Tomcat



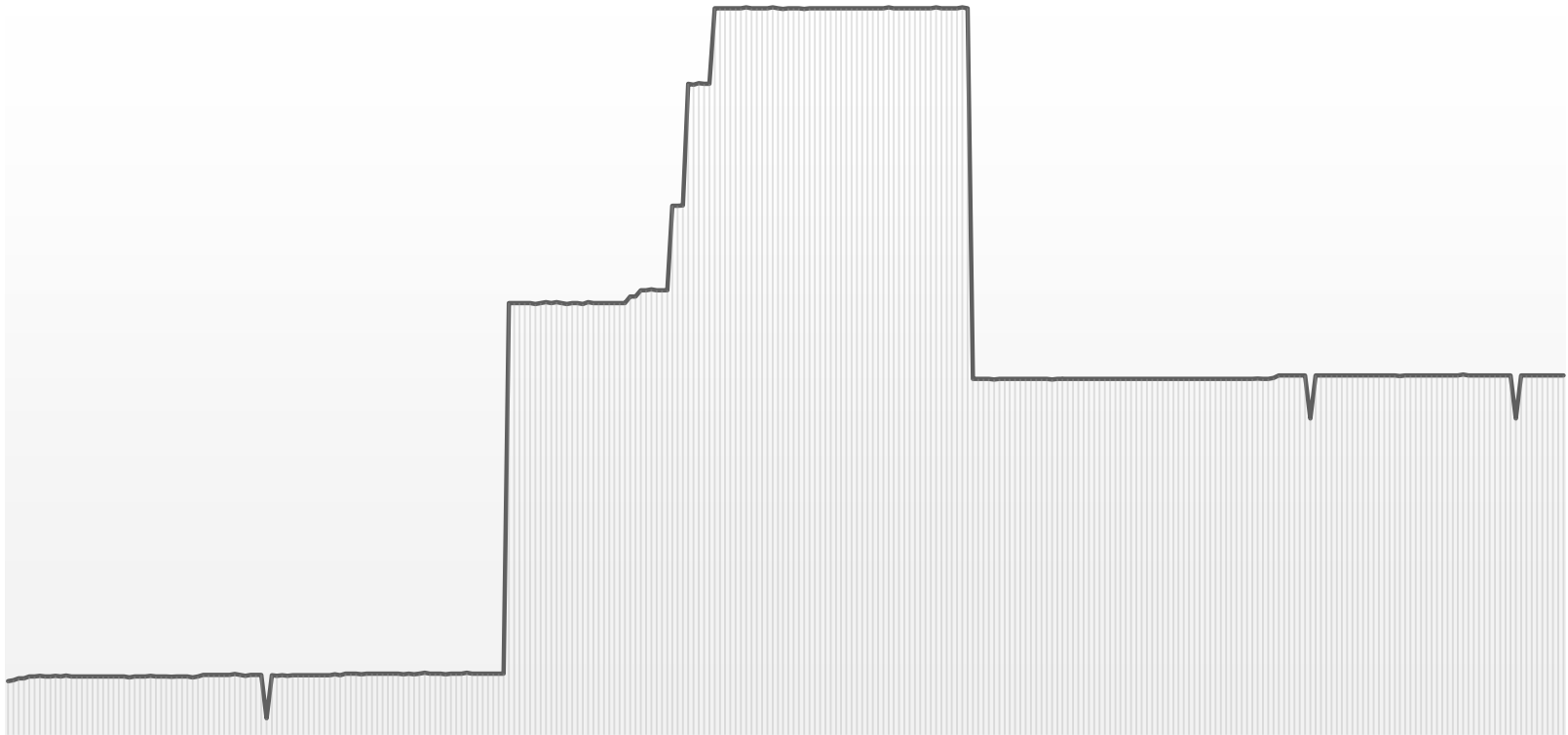
Duration	10 minutes
Sampling interval	2 seconds
Spike size	100MB
Number of log lines	8,911
Number of samples	300
Flagged events	24
Reported lines	63
Precision (%)	4.2%



The engine and Apache Tomcat

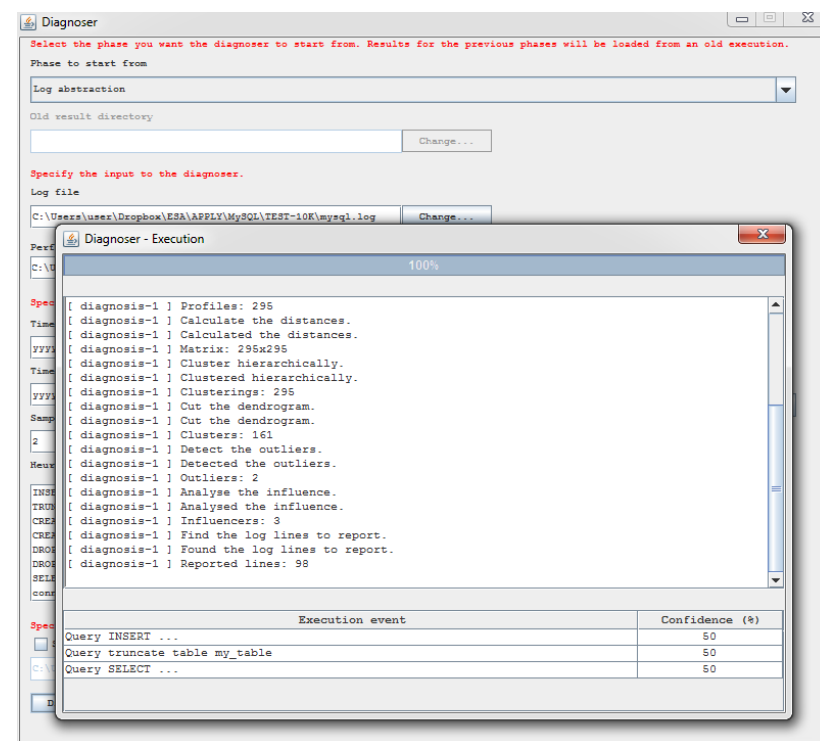


The engine and MySQL server



The engine and MySQL server

Duration	10 minutes
Sampling interval	2 seconds
Leak size	10KB
Number of log lines	36,097
Number of samples	296
Flagged events	3
Reported lines	98
Precision (%)	33.33%



We injected leaks and spikes into **4 different applications**

We ran over **10,000 load tests**

We ran tests that lasted between **10** and **60 minutes**

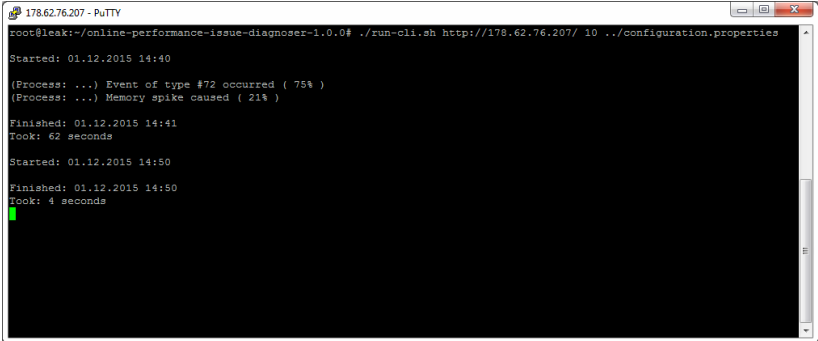
We correctly diagnosed **leaks** between **100B** and **10MB**

We correctly diagnosed **spikes** between **1KB** and **100MB**

The proactive error detection system prototype



Both execution logs and performance counters from a **running application** are **periodically copied** into a **web accessible** directory under **well-known names**

A screenshot of a terminal window titled '178.62.76.207 - PuTTY'. The terminal shows the output of a script. The first run starts at 01.12.2015 14:40 and finishes at 01.12.2015 14:41, taking 62 seconds. It reports an event of type #72 occurred (75%) and a memory spike caused (21%). The second run starts at 01.12.2015 14:50 and finishes at 01.12.2015 14:50, taking 4 seconds. A green cursor is visible at the end of the second run's output.

```
root@leak:~/online-performance-issue-diagnoser-1.0.0$ ./run-cli.sh http://178.62.76.207/ 10 ../configuration.properties
Started: 01.12.2015 14:40
(Process: ...) Event of type #72 occurred ( 75% )
(Process: ...) Memory spike caused ( 21% )
Finished: 01.12.2015 14:41
Took: 62 seconds

Started: 01.12.2015 14:50
Finished: 01.12.2015 14:50
Took: 4 seconds
```

The files are periodically **downloaded**, and the approach is **applied** on them

The results are **reported** and **archived** for future reference

When there are no **business processes** (i.e. sequences of events that repeat over time), time-slice profiles look random.

When too many **threads** write in the same log, time-slice profiles become **too** diverse.

When the **frequency** of a **normal event** is **too high**, it is incorrectly considered as overly influential.

When the **sampling interval** is **too long**, time-slice profiles look too similar. When the sampling interval is **too short**, time-slice profiles look too different.

- The approach will most likely work.
- The approach will most likely flag an event that is relevant to the problem.
- The approach will most likely also flag several events that have nothing to do with the problem.

Leveraging System Performance Metrics and Execution Logs to Proactively Diagnose System of Systems Performance Issues

- ✓ **Analyse** the approach proposed by Syer et al.
- ✓ Build a **prototype correlation engine** based on that approach
- ✓ Analyse the **suitability** of **logs** produced by HSO-GI software
- ✓ Apply the correlation engine on **simulation applications**
- ✗ Apply the correlation engine **HSO-GI software**
- ✓ Apply the correlation engine on other **real-world applications**
- ✓ Build a **proactive error detection system prototype** based on that approach

- Make it work on **large data sets**
 - **Now:** single-threaded (almost), all data in memory
- Implement a better alternative for the **influence analysis** phase
 - **Now:** blame too rare events
- Allow the user to provide **feedback**
 - **Now:** the time that user spends to further analyse the flagged events is wasted