Pietro Liguori

Tutor: Domenico Cotroneo – co-Tutor: Roberto Natella

XXXIV Cycle - III year presentation

Fault Injection for Cloud Computing Systems

From Failure Mode Analysis to Runtime Failure Detection

Università degli Studi di Napoli
Federico II

Background

- I received in year 2018 the Master Science degree cum laude in Computer Engineering from the University of Napoli Federico II
- I attended a curriculum in Computer Engineering within the PhD programme in Information Technology and Electrical Engineering at the University of Napoli Federico II
- I received a grant from Ateneo Federico II
- I am part of the DESSERT (Dependable and Secure Software Engineering and Real-Time Systems) research group, DIETI Department
- I collaborated with a research group in the College of Computing and Informatics at the University of North Carolina - Charlotte (UNCC), North Carolina, United States

Credits Table

	Credits year 1								Credits year 2								Credits year 3											
		1	2	3	4	5	6			1	2	3	4	5	6			1	2	3	4	5	6	7	8			
	Estimated	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	Summary	Estimated	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	Summary	Estimated	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	Summary	Total	Check
Modules	25	0	2,2	6	9	3,6	4,8	**26**	10	4,3	0	0	6	0	0	**10**	5	0	0	4	2	0	0	0	0	**6**	**42**	30-70
Seminars	5	0,8	0	0,5	3,8	0,8	0	**5,9**	5	0	3,2	0	0	0	0	**3,2**	5	1,6	0	0	0	1,6	3,2	0	0	**6,4**	**16**	10-30
Research	30	9,2	7,8	3,5	0	5,6	5,2	**31**	45	5,7	6,8	10	4	10	10	**47**	50	6,4	8	4	6	6,4	4,8	8	4	**48**	**125**	80-140
	60	10	10	10	13	10	10	**63**	60	10	10	10	10	10	10	**60**	60	8	8	8	8	8	8	8	4	**60**	**183**	180

Main Research Activity Overview

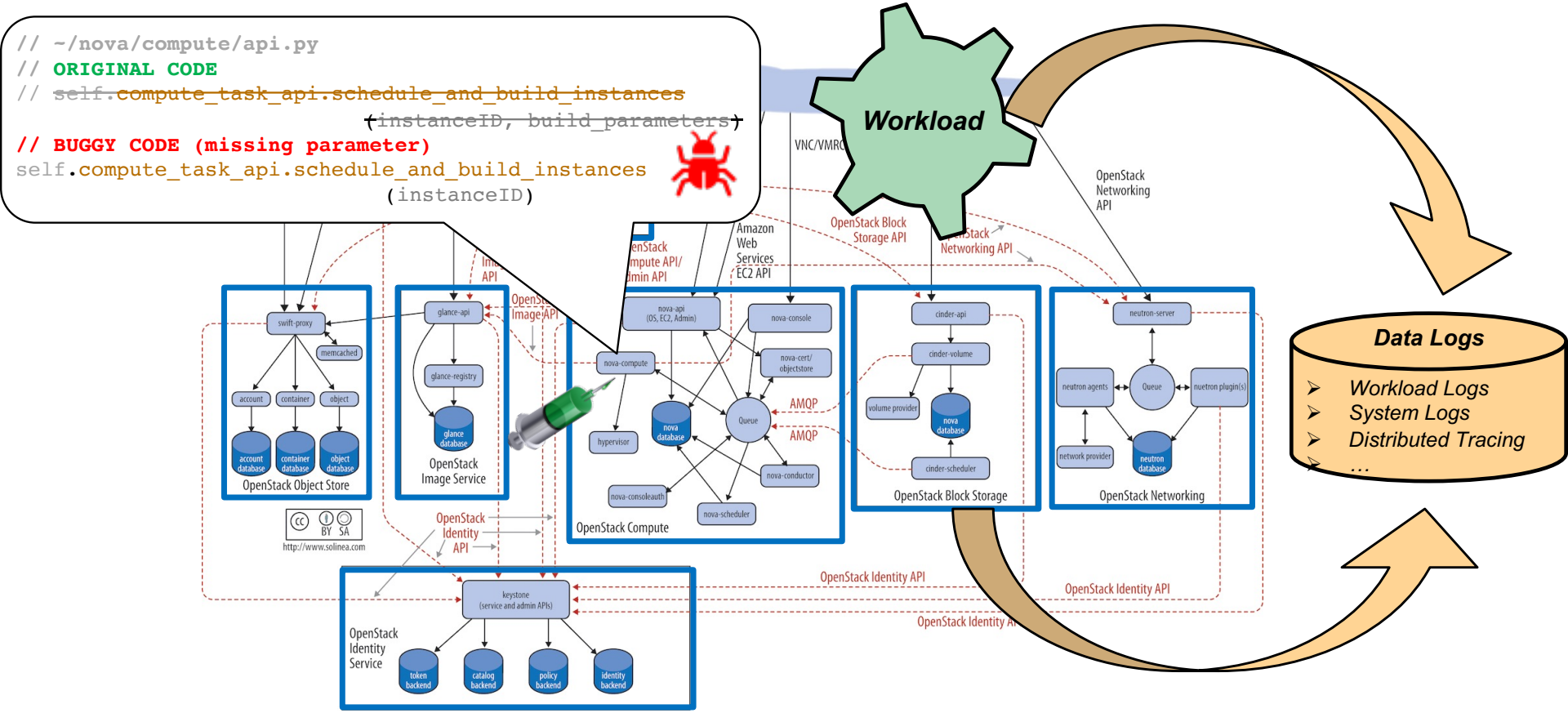
- **Research Problem**: Fault-injection in cloud computing infrastructures for reliability issues
- **Proposed Solutions**:
  - Fault-injection tool-suite 
  - Anomaly Detection Approach to Identify Failure Symptoms 
  - Machine Learning Approaches to Failure Mode Analysis 
  - A Monitoring Strategy to Runtime Failure Detection 

Further Research Activities

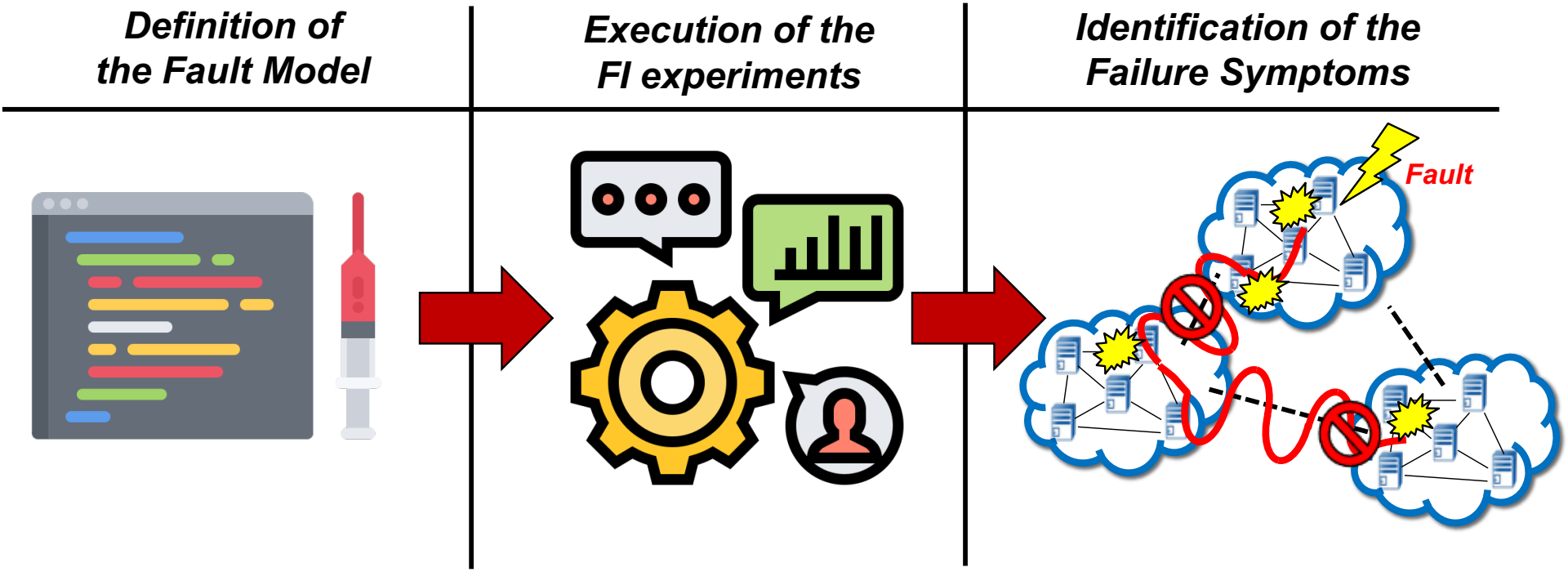
- Automatic Generation of Software Exploits starting from Natural Language description
  - Starting from description in English language, I developed an approach to automatically generate shellcodes in assembly and their encoding/decoding structures in Python/assembly
  - The approach leverages Neural Machine Translation (NMT) techniques
- Research activity in collaboration with UNCC

Fault-injection in Cloud Infrastructures

```
// ~/nova/compute/api.py
// ORIGINAL CODE
// self.compute_task_api.schedule_and_build_instances
                        (instanceID, build_parameters)
// BUGGY CODE (missing parameter)
self.compute_task_api.schedule_and_build_instances
                        (instanceID)
```



Open Issues

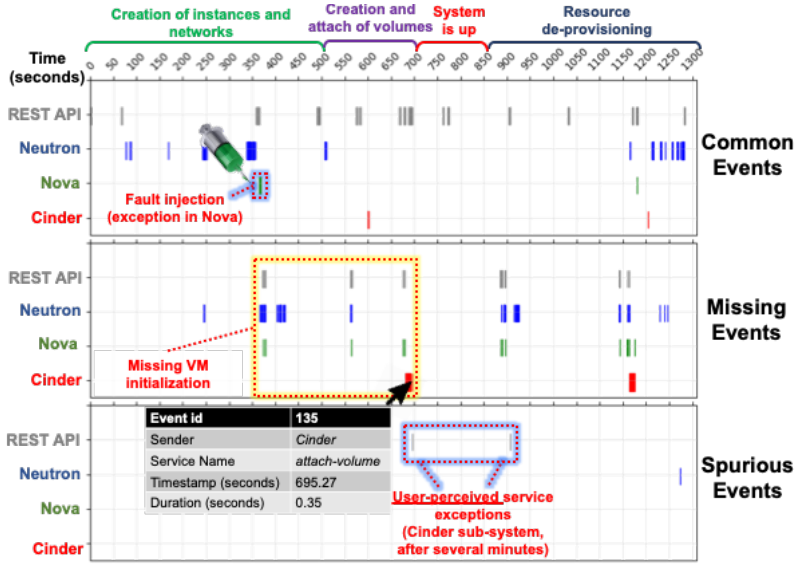

Definition of the Fault Model
Execution of the FI experiments
Identification of the Failure Symptoms
Fault

Fault-Injection Tool-suite

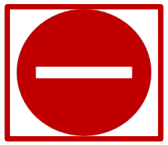
- The tool provides a ***fault injection as a service*** solution to automate and accelerate the tests on the cloud
  - It allows the user to customize faults and introduce new faults
  - Fault models validated in cooperation with *Huawei Technologies Co. Ltd.*
  - Use a mix of *Python language* elements and/or *DSL* directives to describe code pattern and replacement

```
change {
    // the code pattern
}
into {
    // the code replacement
}
```

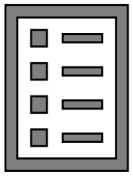
- Provides ***advanced features***
  - *Coverage analysis, Failure logging, Failure propagation, Failure visualization*



Empirical Analysis of Cloud Systems



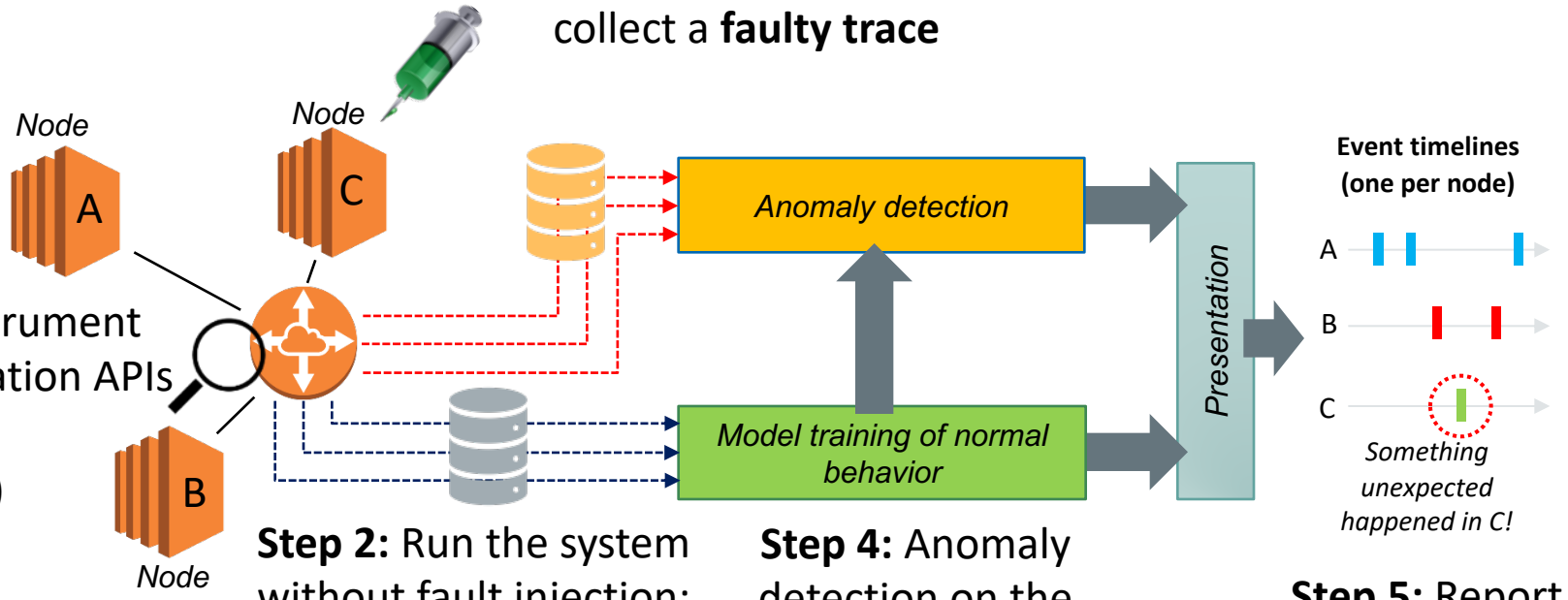
- **RQ1:** *Are failures actually "fail-stop"?*
  - ❖ *Answer:* In **the majority** of the cases, OpenStack does not behave in a **«fail-stop»** way (late or no API error)
  - ✓ *Suggestions:* Mitigate failures by actively checking the **status of virtual resources** as in our assertion checks (e.g., checks incorporated in a monitoring solution)



- **RQ2:** *Are failures logged?*
  - ❖ *Answer:* In a **small** fraction of the experiments, there was **no indication of the failure in the logs**
  - ✓ *Suggestions*: Improve logging in the source code (e.g., by checking for errors returned by the faulty function calls)

- **RQ3:** *Are failures propagated across sub-systems?*
  - ❖ *Answer:* In **most** of the failures, the injected bugs **propagated** across several OpenStack sub-systems. There were also **relevant cases** of failures that caused **subtle residual effects** on OpenStack
  - ✓ *Suggestions*: Improve **resource clean-up** on errors, to prevent propagation across service API calls and across subsystems.

Identification of the Failure Symptoms



Anomaly Detection Algorithm

- Compare faulty-traces with fault-free traces using **Longest Common Subsequence** (LCS)

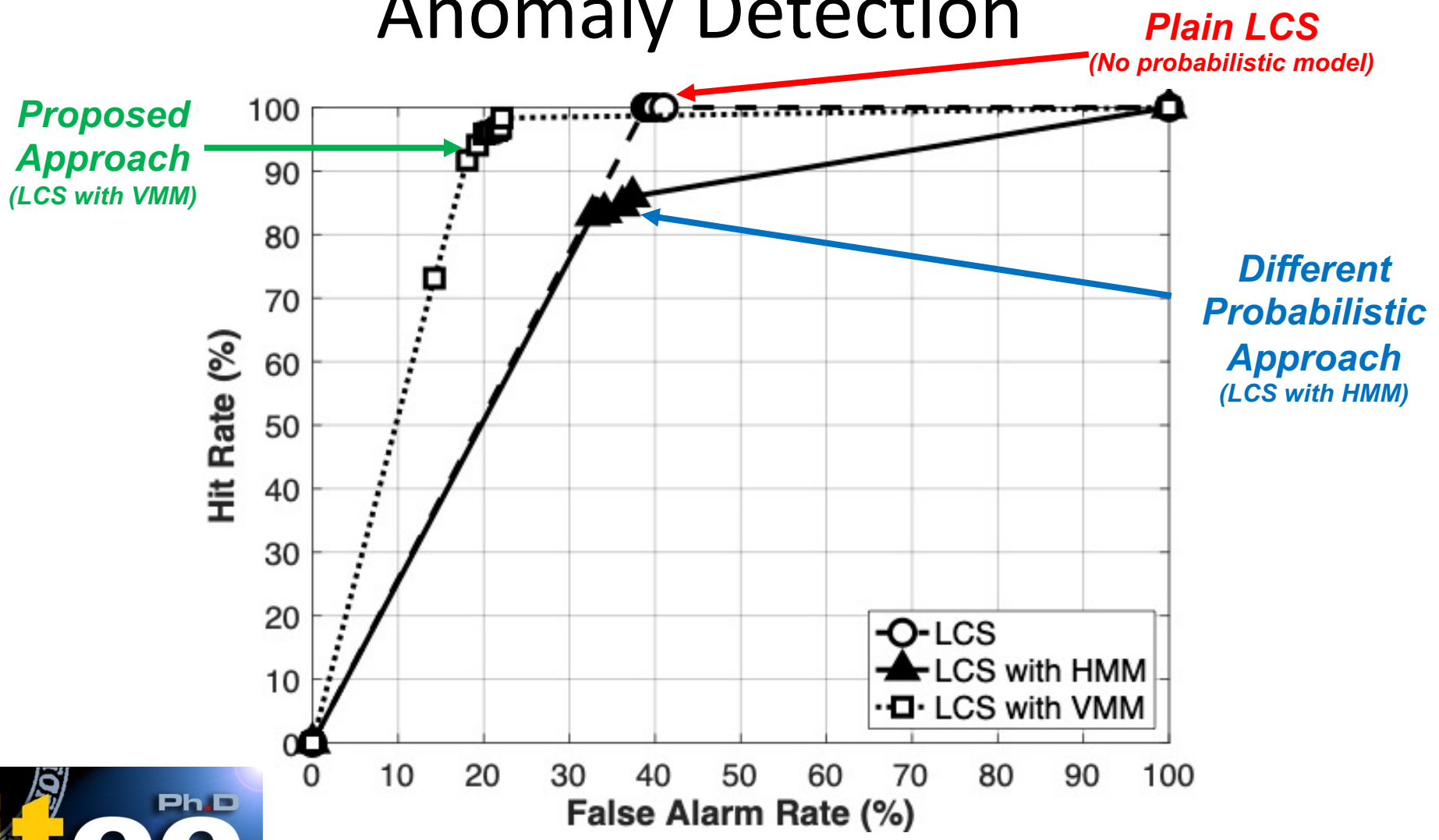
Non-determinism of cloud systems

Not all the deviations are true anomalies

Probabilistic Model

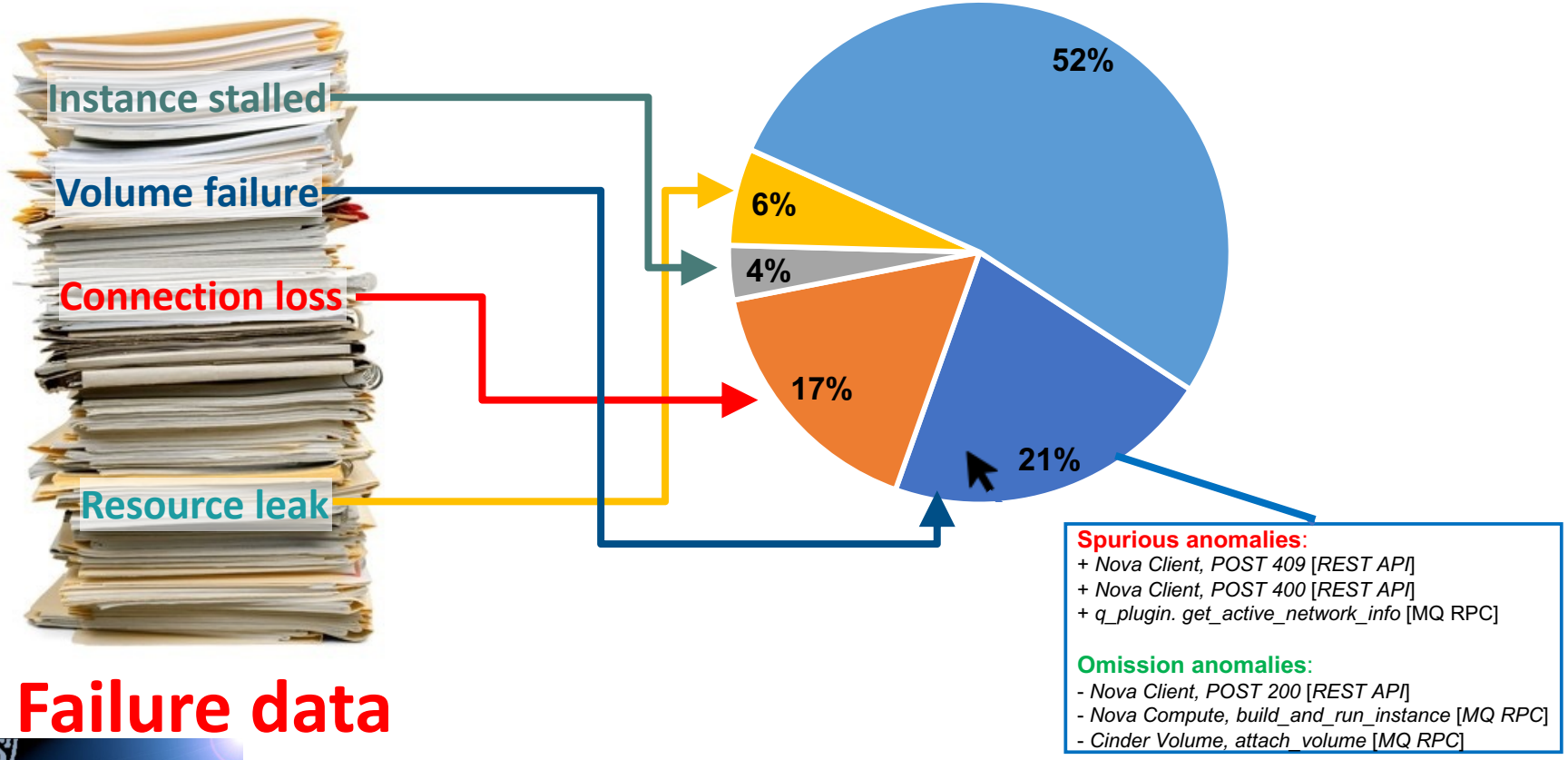
*Application of **Variable order Markov Model** on the deviations to discard benign variations*

Experimental Evaluation of the Anomaly Detection



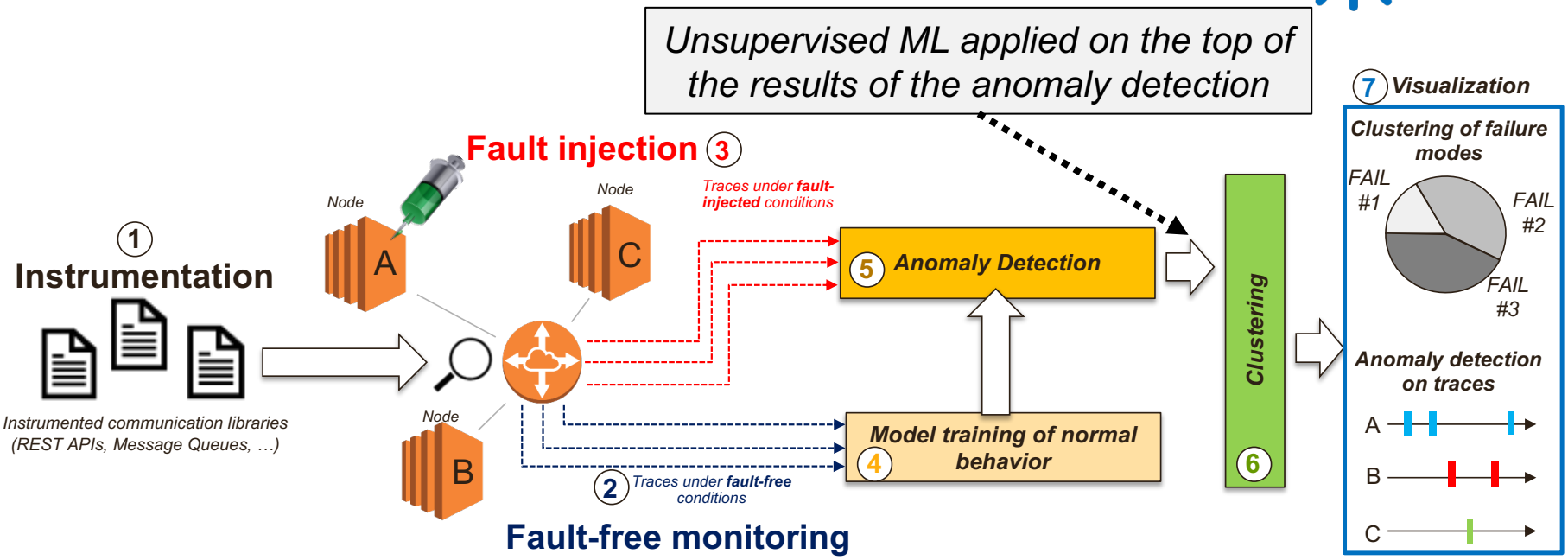
Failure Mode Analysis

Failure mode clustering



Failure data

Failure Mode Analysis



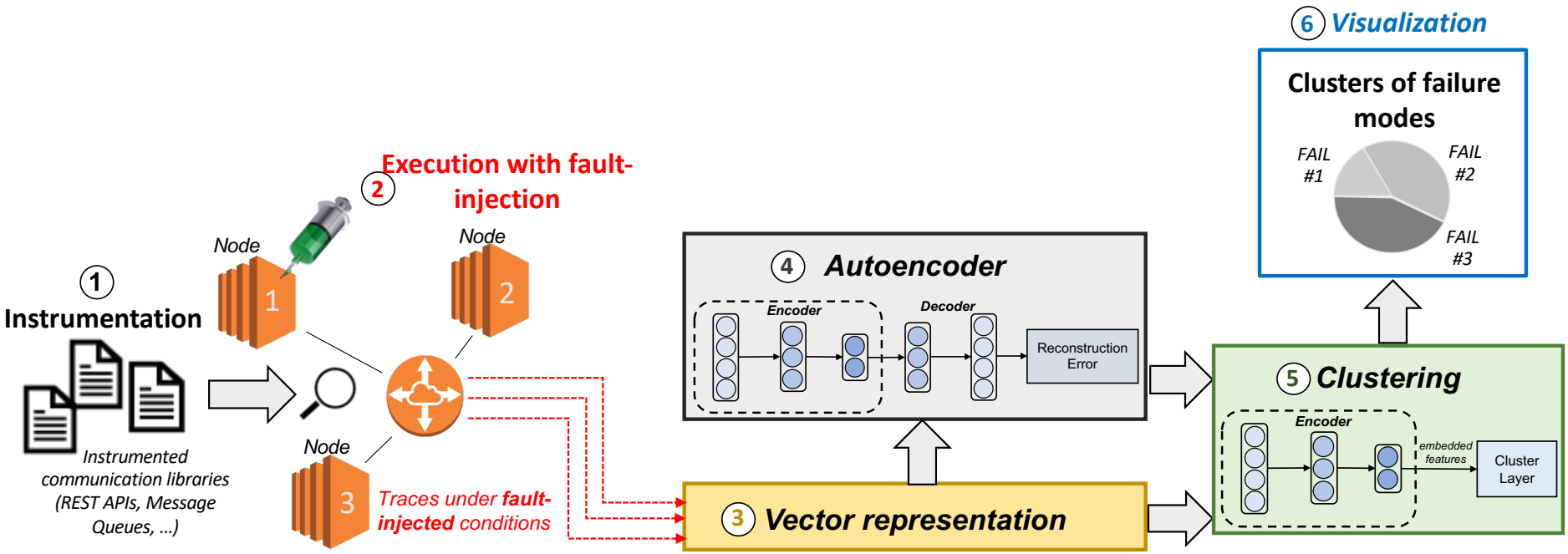
Event Type

$$[1, 0, 9, \ldots, 0, 3]$$

i-th faulty trace

# of Anomalies

DL Approach to Failure Mode Analysis



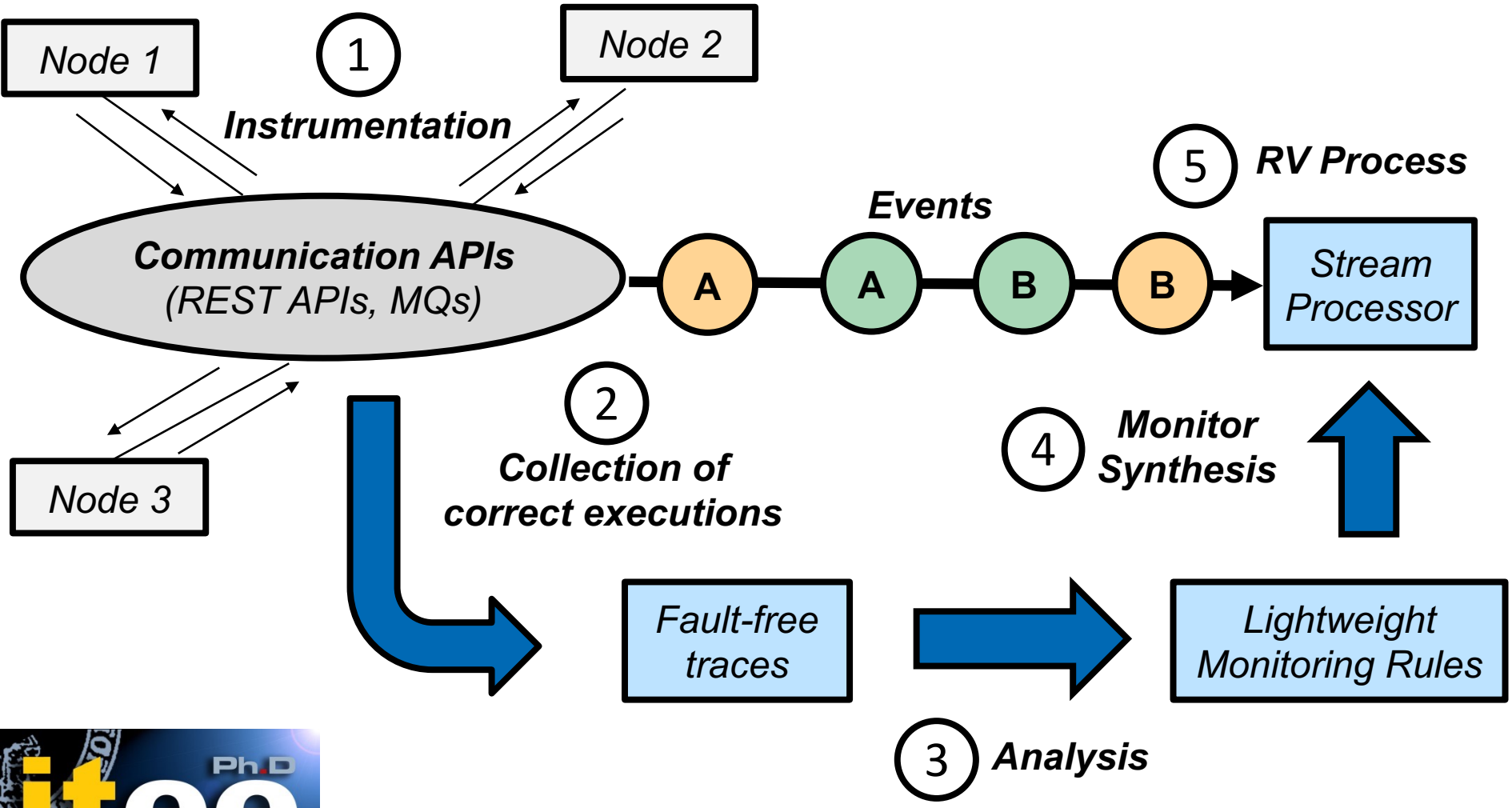
The approach uses **Deep Embedded Clustering (DEC)** and does not require a manual effort by the human analyst for feature engineering

Clustering Evaluation

	Workload		
Clustering Approach	**DEPL**	**NET**	**STO**
Clustering w/o fine-tuning	0.80	0.78	0.87
Clustering with fine-tuning	**0.94**	**0.86**	**0.90**
Deep Embedded Clustering	0.84	0.83	0.89

***DEC* approaches the performance of manually-tuned clustering with anomaly detection**

Runtime Failure Detection



Rule Types

- **Ordered-Events Rules** (ORD): Events following always the same order and occurrence

$$a \rightarrow b \rightarrow c$$

- **Occurred-Events Rules** (OCC): Events occurring without following any specific order and/or occurrence

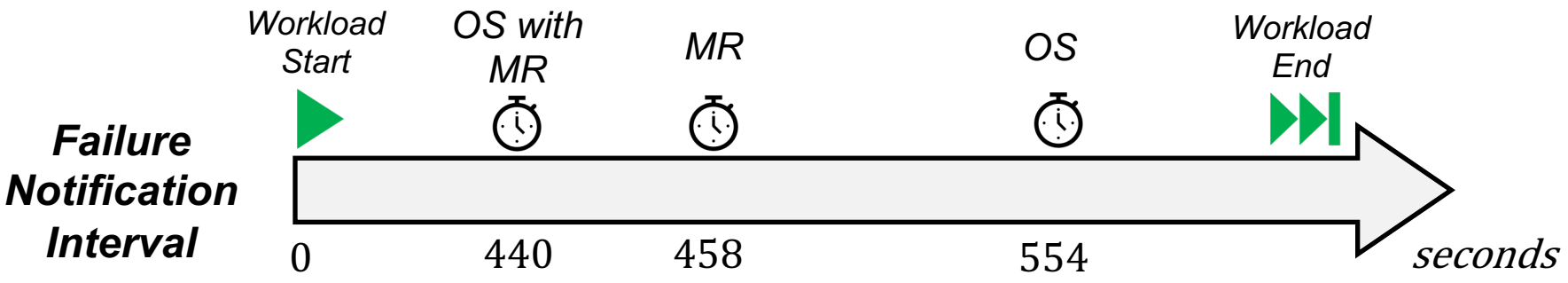
$$a \rightarrow b \rightarrow b \rightarrow c \quad \text{or} \quad a \rightarrow c \rightarrow b$$

- **Counted-Events Rules** (COUNT): Event repeated several times varying in a range of value

$$min < |a| < max$$

Experimental Evaluation of the Runtime Monitoring Approach

Approach	Precision	Recall	F1 Score
OpenStack (OS)	1.00	0.36	0.53
Monitoring Rules (MR)	0.89	0.81	0.85
OS with MR	0.89	0.92	0.91



Research Products (1/2)

International Journals Q1 (SCIMAGOJR)

1. D. Cotroneo, L. De Simone, P. Liguori and R. Natella, "Fault Injection Analytics: A Novel Approach to Discover Failure Modes in Cloud-Computing Systems," in *IEEE Transactions on Dependable and Secure Computing*, September 2020. DOI: 10.1109/TDSC.2020.3025289
2. D. Cotroneo, L. De Simone, P. Liguori, and R. Natella, "Enhancing the analysis of software failures in cloud computing systems with deep learning," in *Journal of Systems and Software*, Volume 181, 2021, 111043, ISSN 0164-1212. DOI: 10.1016/j.jss.2021.111043

International Journals Q2 (SCIMAGOJR)

3. P. Liguori, E. Al-Hossami, D. Cotroneo, R. Natella, B. Cukic, and S. Shaikh, "Can We Generate Shellcodes via Natural Language? An Empirical Study", in *Automated Software Engineering*, 2022, Accepted for Publication

International Conference A+ (GGS Rating)

4. D. Cotroneo, L. De Simone, P. Liguori, R. Natella, and N. Bidokhti, "How bad can a bug get? an empirical analysis of software failures in the OpenStack cloud computing platform," In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (ESEC/FSE 2019), 2019, Association for Computing Machinery, New York, NY, USA, pp. 200–211. DOI: 10.1145/3338906.3338916

International Conference A (GGS Rating)

5. D. Cotroneo, L. De Simone, P. Liguori and R. Natella, "ProFIPy: Programmable Software Fault Injection as-a-Service," *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (DSN), 2020, pp. 364-372. DOI: 10.1109/DSN48063.2020.00052

Research Products (2/2)

International Conference A-, B (GGS Rating)

6. D. Cotroneo, L. De Simone, P. Liguori, R. Natella and N. Bidokhti, "Enhancing Failure Propagation Analysis in Cloud Computing Systems," *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 2019, pp. 139-150. DOI: 10.1109/ISSRE.2019.00023.
7. P. Liguori, E. Al-Hossami, V. Orbinato, R. Natella, S. Shaikh, D. Cotroneo and B. Cukic "EVIL: Exploiting Software via Natural Language," *2021 IEEE 32nd International Symposium on Software Reliability Engineering* (ISSRE), 2021. DOI: 10.1109/ISSRE52982.2021.00042

Other Conferences and International Workshops

8. D. Cotroneo, L. De Simone, P. Liguori, R. Natella and N. Bidokhti, "FailViz: A Tool for Visualizing Fault Injection Experiments in Distributed Systems," *2019 15th European Dependable Computing Conference* (EDCC), 2019, pp. 145-148. DOI: 10.1109/EDCC.2019.00036.
9. D. Cotroneo, L. De Simone, A. Di Martino, P. Liguori and R. Natella, "Enhancing the Analysis of Error Propagation and Failure Modes in Cloud Systems," *2018 IEEE International Symposium on Software Reliability Engineering Workshops* (ISSREW), 2018, pp. 140-141, DOI: 10.1109/ISSREW.2018.00-13
10. D. Cotroneo, L. De Simone, P. Liguori, R. Natella, and A. Scibelli, "Towards Runtime Verification via Event Stream Processing in Cloud Computing Infrastructures," In: *Service-Oriented Computing – ICSOC 2020 Workshops* (ICSOC 2020). Lecture Notes in Computer Science, vol 12632. Springer, Cham. DOI: 10.1007/978-3-030-76352-7_19
11. P. Liguori, E. Al-Hossami, D. Cotroneo, R. Natella, B. Cukic, and S. Shaikh, "Shellcode_IA32: A Dataset for Automatic Shellcode Generation," in *Proceedings of the 1st Workshop on Natural Language Processing for Programming* (NLP4Prog 2021), 2021, pp. 58-54. DOI: 10.18653/v1/2021.nlp4prog-1.7
12. P. Liguori, C. Improta, S. De Vivo, R. Natella, B. Cukic and D. Cotroneo, "Can NMT Understand Me? Towards Perturbation-based Evaluation of NMT Models for Code Generation", in *The 1st Intl. Workshop on Natural Language-based Software Engineering* (NLBSE 2022), Submitted for First Review