Distributed Novelty Detection at the Edge for IoT Network Security

Luís Puhl, Guilherme Weigert Cassales, Helio Crestana Guardia, Hermes Senger

Luís Puhl

Universidade Federal de São Carlos, Brasil

luispuhl@gmail.com

September 13-16, 2021

Introduction

Context

- Growth of IoT devices and associated risks;
  - Heterogeneous devices;
  - Less frequent software updates;
  - Example: Mirai Botnet, infecting IP cameras and routers, generating 620 Gb/s [1].
- Network Intrusion Detection:
  - Detection by signature versus anomaly;
  - Fog and IoT network environment.

Proposal

- A system for IoT network intrusion detection implemented on the fog;
- The hypothesis of this work is: The MINAS algorithm can be run distributed in fog, reducing latency without classification quality reduction.

Introduction - Scenario

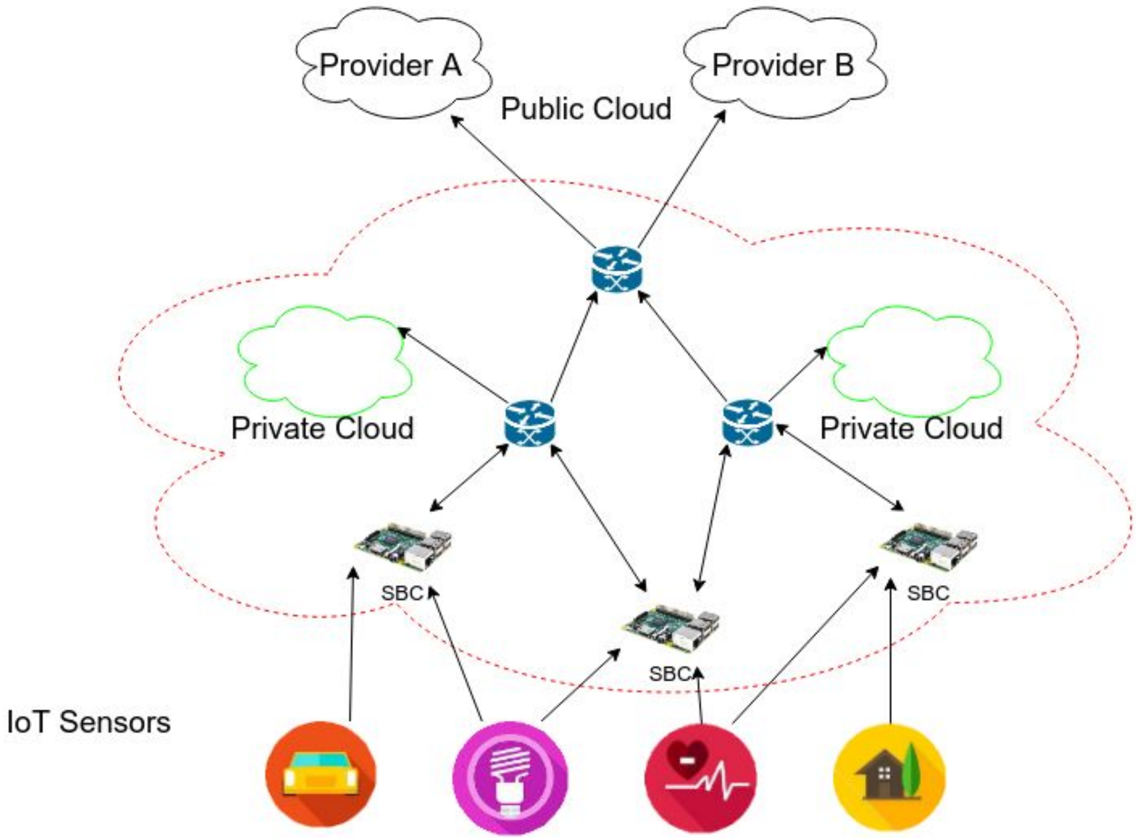

Fig. 1: IDSA-IoT [2] physical architecture and deployment scenario overview.

Related Work

BigFlow [5]:

- ● Intrusion via anomaly detection system capable of handling high speed networks;
- \+ Complete integration from flow descriptor extraction to alarms;
- \+ Capable of handling 10 Gbps with 40-core cluster;
- \- Weekly update with human specialist intervention;
- \- Cloud only.

Catraca [6]:

- ● Monitoring and threat detection system with stream computing and NFV;
- \+ Layered architecture allocated in cloud and fog;
- \+ Decision model based on decision tree;
- \- Flow descriptor extraction is done in fog, classification and detection on the cloud.

Related Work

IDSA-IoT Architecture [2]:

+ Evaluation of MINAS, ECSMiner and AnyNovel algorithms;
+ Task distribution on fog and cloud, focused on IoT;
- Implementation and evaluation in distributed scenario left open.

MINAS Algorithm [7]

- Algorithm for Novelty Detection in Data Streams;
- Analysis on space $\mathbb{R}^d$;
- Offline-Online learning;
- Classification in *known*, extension, and *novel* patterns or *unknown* labels;
- Decision model using spherical clusters and Euclidean distance;
- Clustering (k-means, CluStream) is used to find new patterns;
- Source available at http://www.facom.ufu.br/~elaine/MINAS.

Proposal

- Employ MINAS in a IoT-IDS on a fog environment, implementing the IDSA-IoT architecture;
- Observe effects on classification quality due to distribution for scalability;
- Implement and evaluate viability and quality;

Method:

- Choice of technique and platform for implementation;
- Implementation of the IDSA-IoT architecture:
    - Extend fog usage to minimize latency;
- Experimentation with a suitable environment and dataset:
    - Classification quality metrics for validation;
    - Scalability metrics.

Implementation with MPI

- C, OpenMPI 4.0.4, compiled on Raspberry Pi;
- 2 modules: Root (single node) and Leaf (remainder nodes);
- Root: Sampler and Detector tasks;
- Leaf: Classifier (paralel) and Model Update tasks;
- Available at https://github.com/luis-puhl/minas-flink.

Implementation with MPI

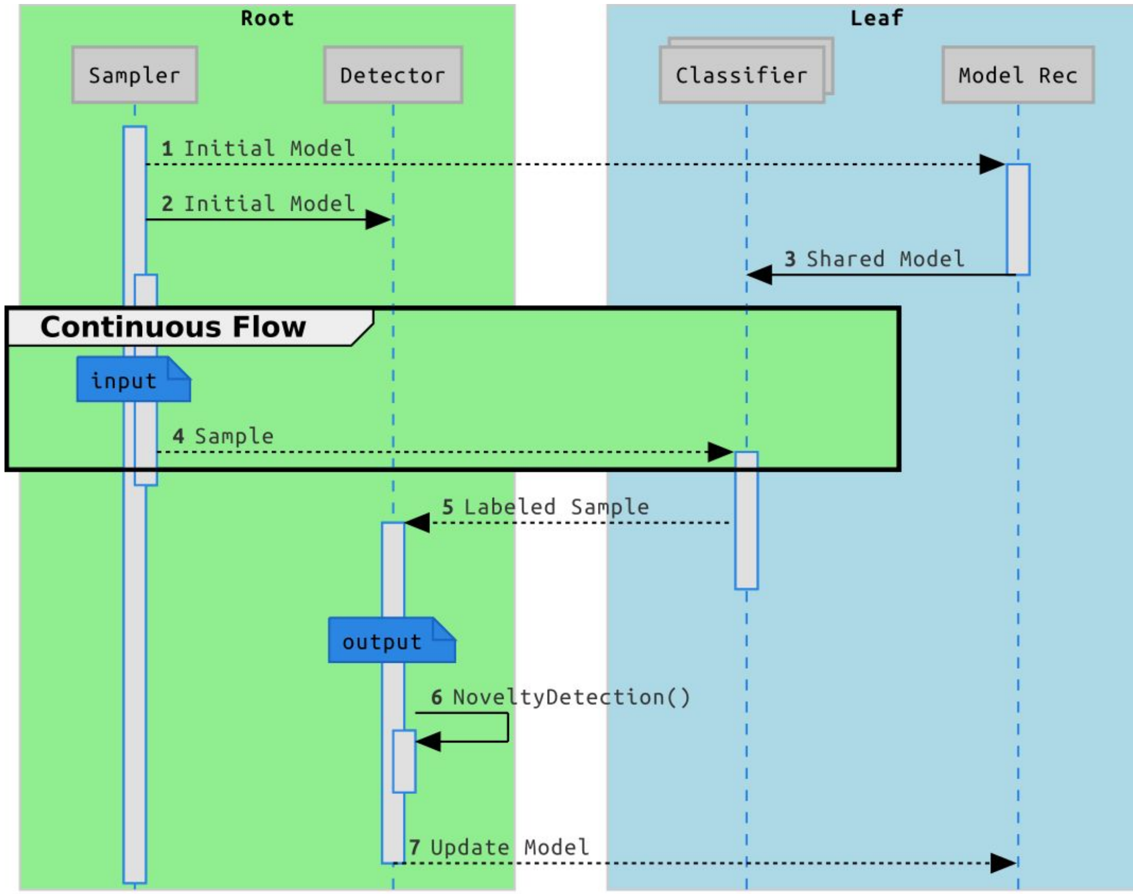

Fig. 2: MFOG Sequence diagram.

Experiments and Results

Experimental Setup:

- Executed in a 3 Raspberry Pi 3B and Ethernet environment;
- December 2015 segment of Kyoto 2006+ data set [8]:
    - Available at http://www.takakura.com/Kyoto_data/.
    - 72 000 samples for training (offline) and 653 457 test (online) samples;
    - “N” (normal, 206 278 instances) known class;
    - “A” (attack, 447 179 instances) class to be detected as novelty.

Measurements:

- Multiclass confusion matrix with novelty label assignment;
    - Summary metrics (*Hits, Unknowns, Misses*);
    - Stream visualization of summary metrics;
- GNU Time: *Time*, *System* and *Elapsed* in seconds.

Experiments and Results

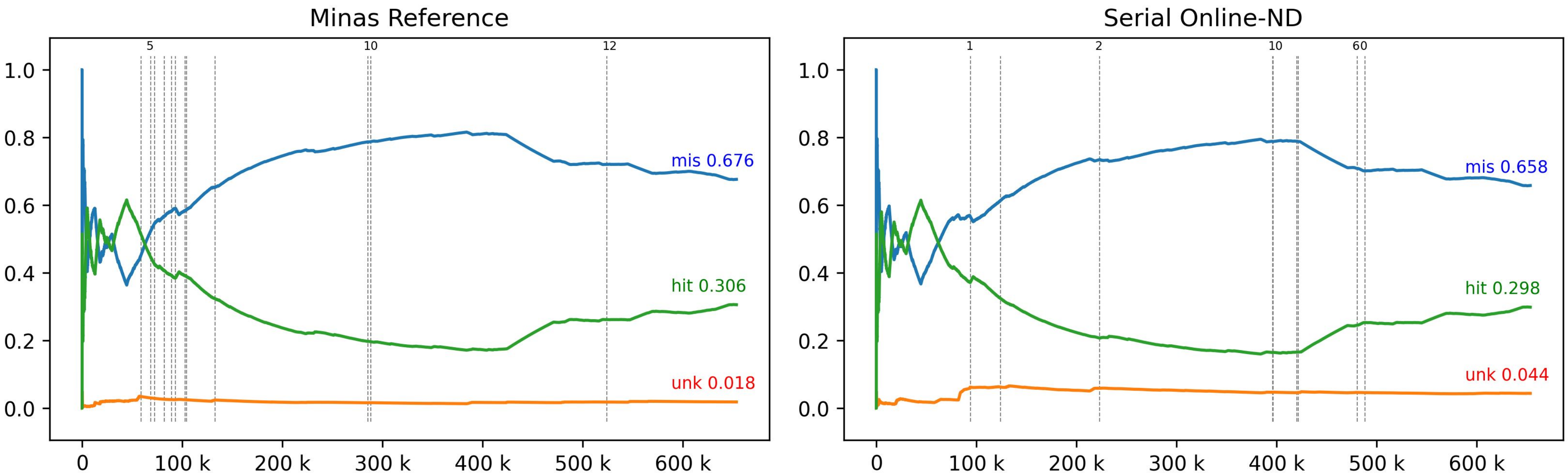

Fig. 3: Stream hits and novelties visualization.

Experiments and Results

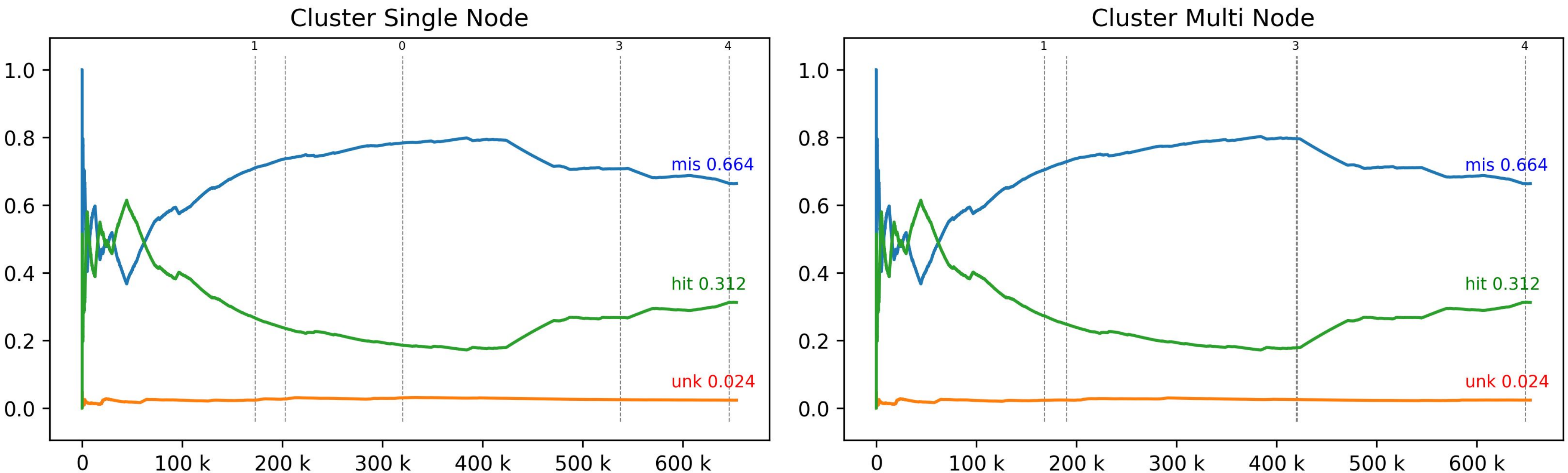

Fig. 3: Stream hits and novelties visualization.

Experiments and Results

Experiment Metric	*Ref* (a)	Offline	Sequential (b)	Single Node (c)	Multi Node (d)
unk	11980 0.018333		28567 0.043717	15370 0.023521	15499 0.023718
hit	199708 0.305618		195017 0.298438	204151 0.312416	204191 0.312478
err	441769 0.676049		429873 0.657843	433936 0.664061	433767 0.663802
Time (s)	2761.83	194.12	80.79	522.10	207.14
System (s)	7.15	0.075	11.51	47.77	157.61
Elapsed (s)	2772.07	194.27	93.03	145.04	95.38
Latency (s)	$4.24 \cdot 10^{-3}$		$1.42 \cdot 10^{-4}$	$2.22 \cdot 10^{-4}$	$1.46 \cdot 10^{-4}$
Processors	1	1	1	4	12
Speedup				0.6414092	0.9753617
Efficiency				0.1603523	0.0812801

Table 2: Collected Measures Summary.

Experiments and Results

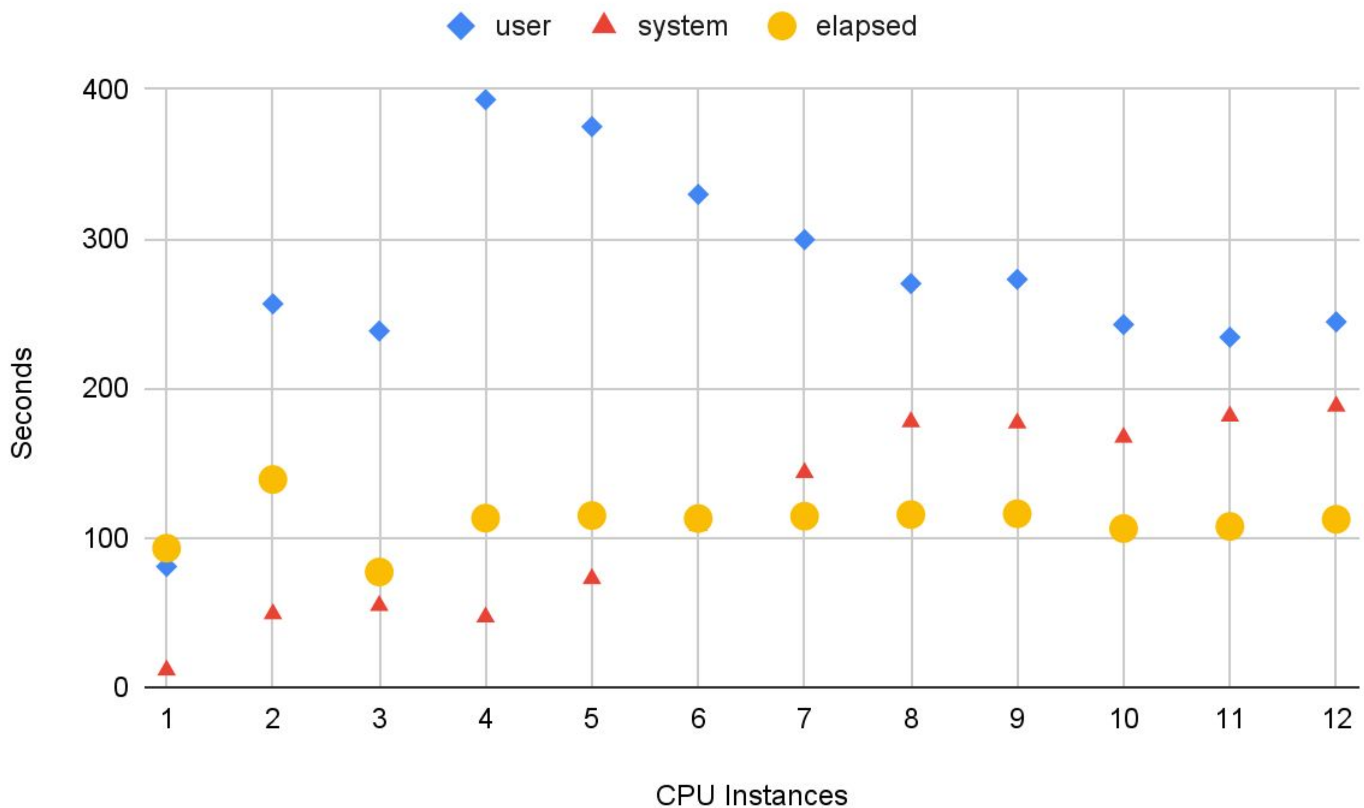

Fig. 4: Time measurements per added instance.

Conclusion

- Data Stream Novelty Detection as in Network Intrusion Detection or system behavior monitoring and analysis on IoT environments is still challenging due to data volume, latency and small devices constraints;
- Distributed data processing is a valid approach for novelty detection in this scenario;
- Our proposal, MFOG, a distributed architecture appling the Novelty Detection algorithm MINAS, was able to serve as an IoT Intrusion Detection system;
- Distribution causes an impact on the predictive metrics however, but a negligible loss of overall accuracy;
- The distributed model was faster than reference implementation, however parallel speedup was lower than expected.
- Further work:
  - Evaluate other Novelty Detection algorithms;
  - Change MINAS internal clustering;
  - Better load balancing strategies.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and Programa Institucional de Internacionalização – CAPES-PrInt UFSCar (Contract 88887.373234/2019-00). Authors also thank Stic AMSUD (project 20-STIC-09), FAPESP (contract numbers 2018/22979-2, and 2015/24461-2) and CNPq (Contract 167345/2018-4) for their support.

References

1. Kolias, C., Kambourakis, G., Stavrou, A., Voas, J.: DDoS in the IoT: Mirai and Other Botnets. Computer 50(7), 80–84 (2017). https://doi.org/10.1109/MC.2017.201.
2. Cassales, G.W., Senger, H., de Faria, E.R., Bifet, A.: Idsa-iot: An intrusion detection system architecture for iot networks. In: 2019 IEEE Symposium on Computers and Communications (ISCC). pp. 1–7 (June 2019). https://doi.org/10.1109/ISCC47284.2019.8969609.
3. AGGARWAL, C. C. et al. A framework for clustering evolving data streams. Proceedings - 29th International Conference on Very Large Data Bases,VLDB 2003, p. 81–92, 2003.

References

4. GAMA, J.; RODRIGUES, P. P. Knowledge Discovery from Data Streams. [S.l.]: Chapman and Hall/CRC, 2010. ISBN 9781439826119.
5. VIEGAS, E. et al. Bigflow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. Future Generation Computer Systems, Elsevier, v. 93, p. 473 – 485, 2019. ISSN 0167-739X.
6. LOPEZ, M. E. A. A monitoring and threat detection system using stream processing as a virtual function for Big Data. Tese (Theses) — Sorbonne Université ; Universidade federal do Rio de Janeiro, Jun 2018. hhttps://tel.archives-ouvertes.fr/tel-02111017i.

References

7. FARIA, E. R. de; CARVALHO, A. C. Ponce de L. F.; GAMA, J. Minas: multiclass learning algorithm for novelty detection in data streams. Data Mining and Knowledge Discovery, v. 30, n. 3, p. 640–680, May 2016. ISSN 1573-756X.
8. Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D., Nakao, K.: Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation. Proceedings of the 1st Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS 2011 pp. 29–36 (2011). https://doi.org/10.1145/1978672.1978676

Fundamentals

Data Stream: a massive sequence, possibly unlimited, of multi-dimensional examples x1, x2, … , xn, … received on the instants t1, t2, …., tn, … [3]

Novelty Detection Techniques: Handle the classification of examples in patterns and the detection of new patterns [4].

- **Concept Evolution**: new concept appearing in the data stream;
- **Concept Drift**: change in a known concept, being sutten, incremental or recurrence;
- **Noise and Outliers**: examples not included in the distribution of a known concept or not in a known concept.

Preliminary Attempts

Python and Apache Kafka on a cluster of 3 constrained computers:

- Hypothesis: Kafka can handle load distribution among nodes via “partitions”;
- Test: 1 producer (the probe), 8 partitions and 8 consumers (the classifiers);
- Result: 1 consumer got the majority while the majority of consumers got none.

Apache Flink on a cluster of 3 small computers:

- Hypothesis: Flink will handle model state and load distribution;
- Test: 1 load producer, 12 classifiers and 1 sink to file;
- Result: On the first run, results equivalent to the original implementation. Subsequent runs exhausted the 1GB memory limit, deemed not reliable.

Experiments and Results

(a) Reference implementation

Labels	-	N	1	2	3	4	5	6	7	8	9	10	11	12
Classes														
A	3774	438750	123	145	368	8	52	165	1	1046	161	2489	71	26
N	8206	193030	0	79	44	0	0	0	229	181	154	4066	289	0
Assigned	-	N	A	A	A	A	A	A	N	A	A	N	N	A
Hits	0	193030	123	145	368	8	52	165	229	1046	161	4066	289	26

(b) Sequential implementation

Labels	-	N	0	1	2	4	5	6	7	8	10
Classes											
A	16086	429765	94	995	104	0	23	3	29	46	34
N	12481	193642	3	94	0	47	0	0	0	11	0
Assigned	-	N	A	A	A	N	A	A	A	A	A
Hits	0	193642	94	995	104	47	23	3	29	46	34

Table 1: Confusion Matrices and Qualitative measurements

Experiments and Results

(c) Parallel single-node

Labels	-	N	0	1	2	3	4
Classes							
A	12282	433797	147	952	0	0	1
N	3088	203019	40	99	27	5	0
Assigned	-	N	A	A	N	N	A
Hits	0	203019	147	952	27	5	1

(d) Parallel multi-node

Labels	-	N	0	1	2	3	4
Classes							
A	12378	433631	117	886	0	162	5
N	3121	202916	40	96	105	0	0
Assigned	-	N	A	A	N	A	A
Hits	0	202916	117	886	105	162	5

Table 1: Confusion Matrices and Qualitative measurements