Faster processing of Multi-Tenant Distributed application and log data from heterogeneous Data-Sources

Sripada H Ravindranath[1] and Dr S Saravana Kumar[2]

[1]Department of Computer and Engineering, CMR University, Bangalore.
[2]Professor, Department of Computer and Engineering, CMR University, Bangalore.
[1]hrsripad@gmail.com [2]saravanakumarmithun@gmail.com

Abstract - With the introduction of cloud-based computation, many like-minded enterprise organizations are migrating applications to cloud and cloud oriented infrastructure platform. In any given system Transactions-Logs, Application-Logs, System/Machine-Logs are one of the important key factors in identifying and managing the customers/business related understanding and the overall-status of the applications running on the cloud/cloud/on-premises hybrid platform environment. They are most vital for numerous scenario-based situation, such as SLA agreements of the business, service-oriented stability assessment of the business, A proper root cause analysis of the issues/impediments etc and user-based activity and its profiling. Therefore, it is essential and important to manage the massive amount of numerous types of logs which are collected on the cloud/Hybrid premises and get insights from their value. They are most vital for numerous scenario-based situation, such as SLA agreements of the business, service-oriented stability assessment of the business, A proper root cause analysis of the issues/impediments etc and user-based activity and it's profiling.

Key Words: Multi-Tenant, Latency, Tagging, Message-queue, Classification, Categorization

1. Introduction

This paper focus on handling this varied data sources from different environment into a parallel processing module/ methodology which can do the justification of putting them into a varied mechanism to analyse over a time for near-real/ real-time needs. Often it happens that in a multi-tenant architecture the tenant refers to the Actual-Owners of a data segment which is the original source of respective client/ System/Operating-System/ Source-Data-Systems etc. The whole idea is to provide a varied mechanism into a granular way to handle the aspects, processing mechanism, distributed log and provide a caching service in the end to faster retrieval and making things easier and relevant for the downstream- ML/AI engine to use the content relatives information with respect to performing LLAP/Low-Latency-Analytical- Processing as needed for faster processing on relevant tenet data on-demand basis.

This paper will also discuss about different possibilities, mechanism related storing, processing, retrieving, under the hood architecture, hand-off between 2 connected processing unites etc to make things clearer from upstream to- downstream consumptions. Based on the broader areas of focus mentioned above we can classify the symbolic processing units to below categories

1. Capturing events/source with respective tagged values
2. Tenant Separation at source and processing
3. Classification of heterogenous data sources and its characteristics
4. Building Unified processing capability
5. No feed forwarding for data
6. Segregated Analysis and pipelined processing of interim results
7. Collating results in streams
8. Publishing the results for respective tenant

2. Literature survey

Since the inception of BigData Technologies and Cloud- oriented platform. Its has become the most prominent feature and necessity for enterprise-organization to handle the massive amount of data which needs to be accumulated from multiple sources. These data sources may not only be limited to the transaction data related to a specific-transaction performed by the vendor/client/customer etc. It goes beyond transaction, and it is subjected to the items like code layout transactions, application logs, functional logs, server logs, VM-performance logs, Api/client/server logs, load-balancer information, metadata management layer performance stack logs etc.

The need for storing, processing and using those app/ machine logs are need of the hour due to its inference towards the metrics it provides of the tenet data processing which might be done using homomorphic way or in a discrete way. These specified mechanisms are carried out in current industry standards but not in unified way. There are

tons of tools, processing utilities etc are being used in current day Software enterprises to make these work which is not feasible in nature for long run, not a cost effective way and also requires numerous skilled professionals for each minuscule part of the processing component being used for this purpose. Often this leads to delayed handover, struck in between component situation and migration challenges on regular intervals on demand changes as needed for modern requirements

In this paper we will focus us on using the platform and data generation using Cloud technologies like Microsoft-Azure/Amazon-Web-Services/Google-Cloud-Platform etc. to mimic the same situation across the three platforms as and when needed to eradicate the challenges of different offerings and also to keep the solution platform agonistic. Data Collected, processed will be adhering to the HIPPA, SOC-2 compliance.

3. Methodology

A. **Capturing events/source with respective tagged value:** The Source system can be a cloud/on-Premise/Database/ logger system which can be running on any given base OS etc with a Logging or may not be with a logging system. This is identified with a Unique Identifier-Value associated with it.

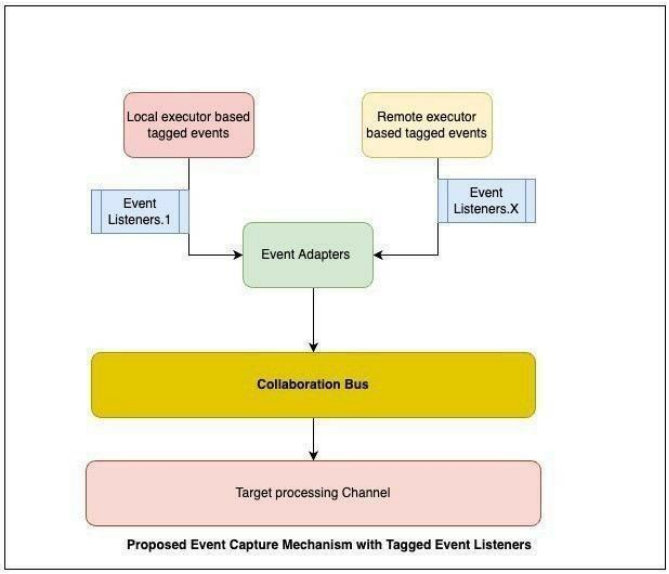


Fig1. Event Capture mechanism from Source

As Showcased in Fig1. Each source systems are tagged with unique executor based event tagging mechanism and those Executors can be Local or Remote in nature. The benefit of this executor is that is can be any source catering mechanism which has capability to generate the Unique- Identity for every event under each and every tent it is being used for. The unique identity is based on OS, Type of system, Type of data, Type of processing, Type of executor etc.

A Event Adapters are collaborator of these such events and then unifies them and process them to collaboration Bus, which is used for adding refined values depending of the nature of data capturing done at the executor level. Target processing is the channel bus which gets the data from the Collaboration bus which is then fed into next downstream consumption for processing layer of the events/action streams. Tagged values referred to the values which comes associated with the event in the collaboration bus. This helps in capturing/managing the unique of the event/data across the system from upstream till downstream mechanism.

B. **Tenant Separation at source and processing:** Tenant separation is the critical and very important piece of processing that is done from the help of the Event listeners and Event adapters. Tenant Separation is either done at the event capturing level or at event adapter processing level.

Event capturing level: The Event capturing is the process of capturing the events/attributions of the respective event which comes with predefined attribution values like eventtype, event-attribute name, time- interval definitions of the attributes etc. These values are captured from the event listens which carefully process these information into possible unique identifiers of the system. The collation of the different attributes into non- negated unique identifiers

which acts single source of truth for the underlying the consistent information system must be handled by message-queue/event-queue bus architecture which has highest-mode of fault-tolerant-high-availability.

Event adapter processing level: The event which is captured in tagged values without any discrepancy in the unique identifier's values in the Adapter level. This is usually done for the events which are not captured as needed in Event-Capturing level. This process involves the mechanism of event-message queue and collaboration bus integration which is needed for the downstream needs.

The below table TABLE1 provides the systematic evaluation of event captured in both ways. It describes the functionalities supported by the both the mechanisms.

TABLE I
COMPARISON METRICS OF EVENT CAPTURING MECHANISM

Sl. No.	**Capturing Mechanism of Data-Events of Sources**		
	Mechanism	**Event-Capturing level**	**Event-Adapter level**
1	Tagging mechanism Support	Supported	Partially Supported
2	Message Queue Aggregation	Not Supported	Fully Supported
3	Efficiency	Efficient	Highly Efficient
4	Supported in Cloud systems	Yes	Yes
5	Support for heterogenous events	Partially Yes	Yes

C. **Classification of heterogenous data sources and its characteristics:** The Classification of the heterogenous data is needed for the LLAP processing depending on the mulct tenant architecture. This is high likely to be used in cross processing of the API layer in the enterprise use cases of the ML/AI applications. The Classification of the heterogenous data sources can be defined based on the below inputs

1. Mode of Event Data:

Stream data or Batch or Near Real time data mode to be applicable for this requirement.

2. Type of Data in arrival:

Formatted, Non-Formatted, information which comes with supervised values embedded into it, Interdependency of the values within it

3.Meta-Information separated at arrival:

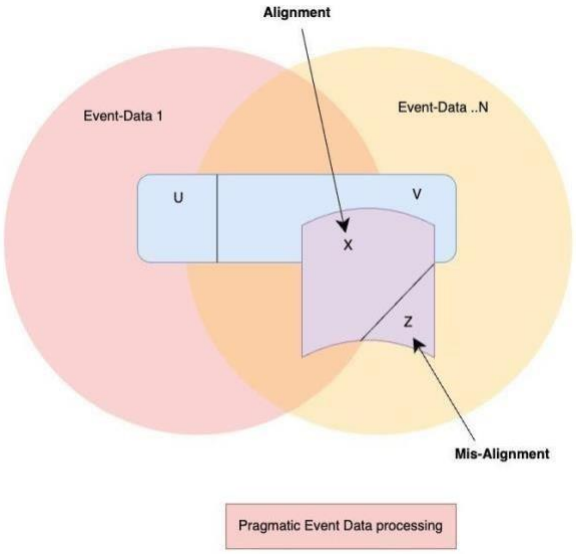
In some cases the values like Metadata and Attributes and Entity Dictionary of the Data might be shared or procured separately from the actual Data-Event

Characteristics of the heterogenous data to be determined with respect to the type of data being consumed and the mode of the data being processed from the system at the time of capturing the event-data from the sources which are discussed below

1. **Semantic-Correspondences:** The data being captured will be evaluated for Instance level correspondences and Multi-variate Hierarchical Correspondences. The semantics observed out of these values are ideally captured along with semantic values and relevant retrospective measures to cumulatively address the metric generated over a time form the same sources.
2. **Terminological Heterogeneity:** This is part of the data conceptualization process in which the Named-Entities of data-event are evaluated for being same the event type from different

source or not. This is done to abstract the semantic feature which is going to be injected into the mechanism of the Target processing system.

3. **Pragmatic Conclusion:** The Pragmatic Conclusion is a mode of approach in which how and when exactly the representation of the expected variation of the data is seen. Usually observed in the multi-tenant and complex event processing like Homomorphic encrypted value processing and Attribute-entity oriented Dictionary generation for the underlying process



Pragmatic Processing

D. **Building Unified processing capability:** The need of building semantic and unified layer is to have a dedicated approach of processing the data streams/batches and its events in to a single multivariate layer instead of breaking it into numerous nuclear processing events which is not efficient and const effective in nature.

The idea or rationale building the unified process is explained below.

1. To have unified capability for all tenants to view, manage the inference
2. Dedicated bandwidth as and when needed for Processing layer to cater the needs
3. Centralized mechanism to handle any processing events
4. No feed-forwarding mechanism for multivariate data
5. Reduced License cost and managed view of under the hood processing
6. All processing under one umbrella of Security and Data protection compliance

Some of the gains and Trade-offs of having unified processing layer is provided below in TABLE2

TABLE 2
COMPARISON MATRIX

Sl. No	Feature Comparison		
	Methods	**Traditional Approach**	**Proposed Approach**
1	Storage	Fixed, Doesn't Change	Dynamic, Changes as per Needs like cost, Type, Urgency etc.

2	Upper Threshold	Upper cap limited, can't change beyond a limit	Upper cap limited only by infrastructure size and not by processing capacity
3	Efficiency	Efficient for limited and known bandwidth processing	Highly efficient for more complex and heterogenous processing. Less efficient for limited and mundane processing.
4	Cost	Less Cost	High Cost for traditional processing. Less costly for heterogenous processing
5	Code Complexity	Less Complex	Modular Complex
6	Automated Management	Complex for handling	Very easy and Cruise control management
7	Preferable for Super heavy processing	Not a right fit	Great Fit for such processing
8	Centralized Governance	Difficult to build but doable	Easy to govern the entire system up to thread level
9	Built in Services	Depends on vendor	No dependency on vendor to build

The main purpose of handling all the events processing is to have dedicated CPU, Memory, Storage, Network under the single hood for the structure which is going to be used for the larger purpose. The below diagram Fig2 explains the stack visually

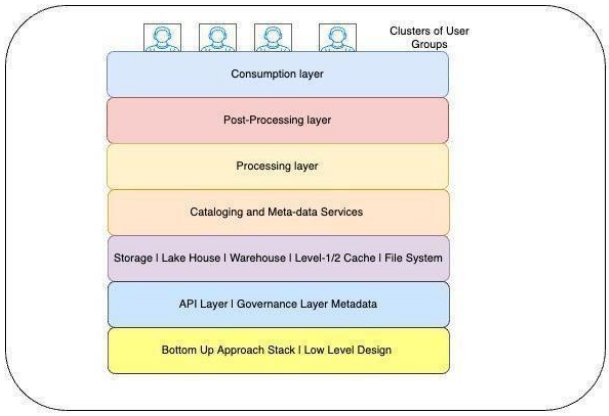


Fig. 2 Stack of proposed Architecture

E. **No feed-forwarding mechanism for multivariate data:** The idea of Multilayer-feedforward neural- network to perform the well-known Sammon nonlinear projection. The learning algorithm is an extension of the backpropagation algorithm. A purpose of the network-based conjunctions over the traditional Sammon algorithm is that the trained network is able to perform different type of patterns as and when needed based on the variation on the inputs parameters provided towards the desired modelled approach. Experimental results indicate that the projection network has good generalization but this is not necessarily an improvement for this kind of stack as it involves many such proportions from particular evolution values. Lower The bound, together with the dedicated-generalization capable values, it provides overall improvement to the system under study/event-under study to make things better and capable than the traditional methods.

F. **Segregated Analysis and pipelined processing of interim results:** The results obtained from the collated processing needs to be always combined to avoid many such iteration for intermediate-key value-pair to keep

and feed-forward mechanism. This helps in many such iteration which might occur due to normalized values of multi meant values. The overall segregated abacuses is shown in below diagram Fig3

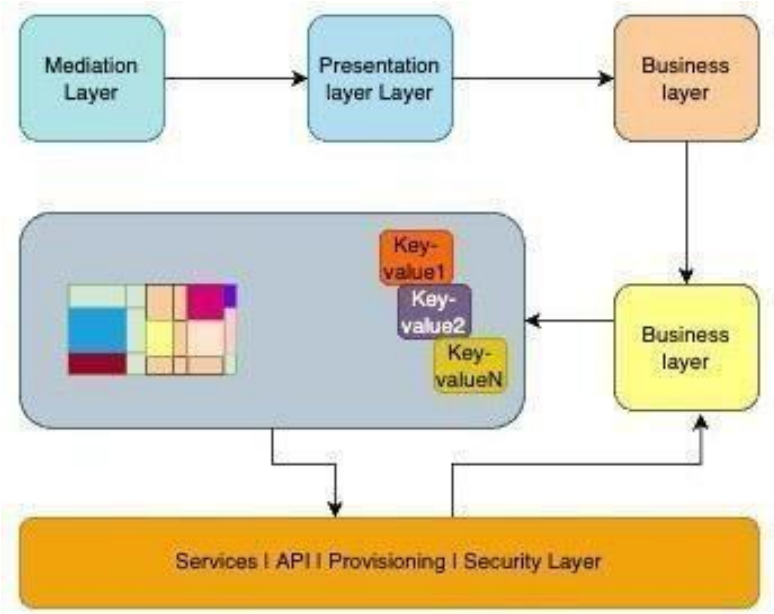


Fig. 3 Segregated Analysis

The interim results are such results which are obtained from the handoff from one thread/process in the entire system which comprises of numerous process events which are interrelated with each other. The process of handoff between the two or more different process of the interim results are called ideally as handover data process. This is needed in orchestrated manner. This is essentially a pipelined process.

The pipelined process can be designed in Dag or a workflow in nature. A sample code module is provided below

```
""" try:
        queue_runs = await
          self.client.get_runs_queue( id=work_queue.i d,
          limit=5)

        )
        submittable_runs.extend(queue_runs)
    except ObjectNotFound:
self.logger.error(“error”) """
```

The larger implementation of the above code is provided in the below snapshot of the python code

```python
async def make_run_manage_job_submit_flows(self) -> List[FlowRun]:
    """
    The principled method on agents. parallelQueries are supported
    The way the job gets submitted to be idenfitied by the WorkFlow Module
    params:
        UIV: Unique_Identifier_value
        Time: AllTime in UST converted in Mins/Hours
        tenantName: Name of the tenant

    """
    if not self.started:
        raise RuntimeError("Agent not turned on. check the agent status")

    self.logger.debug("Checking for agentFlow Runs..")

    before = Act.now("utc").add(
        seconds=self.prefetch_seconds or SRI_AGENT_SECONDS.value()
    )

    submittable_runs = []


    async for sri_work_queue in self.ideal_running_queues():


        if sri_work_queue.is_paused:
            self.logger.info(
                f"Work queue {work_queue.name!r}) is paused."
            )

        else:
            try:
                queue_runs = await self.client.get_runs_queue(
                    id=work_queue.id, limit=5, scheduled_before=before
                )
                submittable_runs.extend(queue_runs)
            except ObjectNotFound:
                self.logger.error(
                    f"Work queue {work_queue.name!r} ({work_queue.id}) is nowheret to be found"
                )
            except Exception as exc:
                self.logger.exception(exc)

    for flow_run in submittable_runs:
        self.logger.info(f"Submitting flow run '{flow_run.id}'")

    return submittable_runs
```

Fig. 4 Code Snippet

G. **Collating the results in Stream:** The collation of the Data from the systems is a unique method in which the overall data from the multiple tenant process threads are collated. The best part is that all the data are proceed from same worker and event thread, but they are bifurcated at the event data level. This is possible because of homomorphic encryptions.

H.

The collective usage if collation in stream to reduced time to gather the aggregated results needed for waiting threads in ML/AI engines which are LLAP in nature and strongly Low-latency-native computation which are occasionally resource hungry in nature due to the enormity of complexity associated with the data.

I. **Publishing the results for respective tenants**:

The last part of the proposed systems is to take a collated results and push them back to consumption layer of the system. The publishing these results are taken care by 3 step process.

1. **Update API Specs and Update the API cache**

The API specs are always updated once the underlying processing engine updates the status of requested compute ID. The Cache of the respective last call gets invalidated, and it updates the new cache with the newest values

2. **Update the client Sensor:**

The Client sensor is the listener from client/Requester which had requested for the data/submitted the job or waiting for the results. This is the listen which is checking for the cache/requested job id on its status if the status is supposed to be updated based on previous step it is gets updated asynchronously.

3. **Update the Job Status Pool:**

The Job pool is the largest pool of current and ongoing jobs pool of information which has the highest lookup from all the relevant and dedicated services in the system. As part of the process, The JOB ID and queue and Specs/Cache status gets updated and thus marking the end of the processing.

4. Conclusions:

The Overall processing capacity increases with increase in the best practices mentioned in this paper. Tighter the module processing better will be the throughput and latency of the downstream layer. Few of the key observation made in this analysis is provided below

1. Complexity remains constant after 70%-80% processing is complete
2. Initial Warm-up time decreases with increase in similar use case scenarios
3. Latency gets better with component evaluation fine tuning parameters in the processing and orchestration-layer.

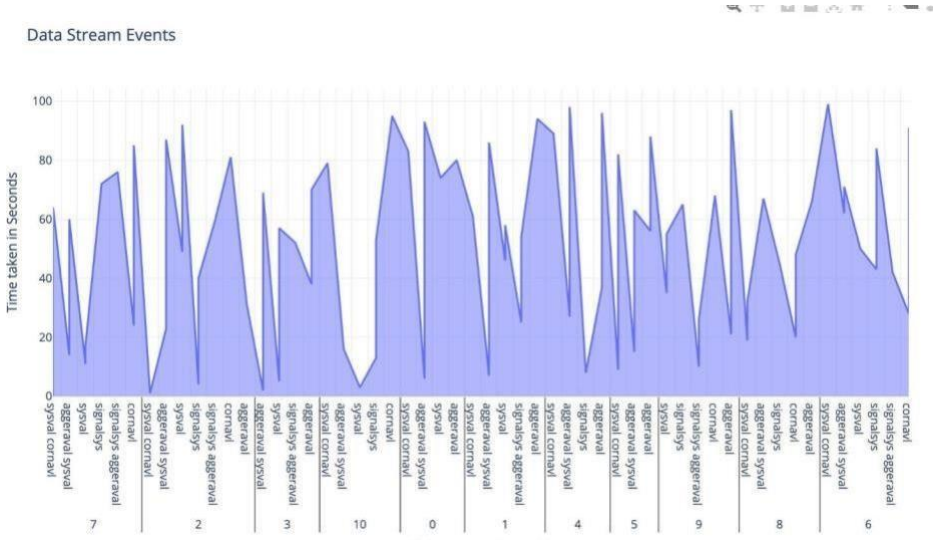


Fig. 5 Performance Plot

The plot obtained from the data after same events getting processed is shown above in Fig5.

Future Improvements

1. There is a huge scope to improve the performance in preprocessing stage by aggregating the typesource information
2. Better results on metadata management can be achieved by connecting the semi-structured API layer with collated metadata fields in processing layer
3. Fine Tuning can be done in Collation of result to reduce time by adopting the much optimized algorithm 4. Improvement in Space Complexity can be achieved by pre analyzing the latency matrix of the process.

REFERENCES

1. M. K. Aguilera, A. Merchant, M. Shah, A. Veitch, and C. Karamanolis. Sinfonia: a new paradigm for building scalable distributed systems. In SOSP '07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, pages 159–174, New York, NY, USA, 2007. ACM 2. https://www.akamai.com Akamai Org

3. S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature polySi TFT," *IEEE Electron Device Lett.*, vol. 20, pp. 569–571, Nov. 1999.
4. J. MacCormick, N. Murphy, M. Najork, C. A. Thekkath, and L. Zhou. Boxwood: abstractions as the foundation for storage infrastructure. In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, pages 8–8, Berkeley, CA, USA, 2004. USENIX Association
5. M. Dahlin, R. Wang, T. Anderson, and D. Patterson. Cooperative Caching: Using Remote Client Memory to Improve File System Performance. In Proceedings of the First Symposium on Operating Systems Design and Implementation, pages 267–280, Nov 1994 6.

(2002) The IEEE website. [Online]. Available: http://www.ieee.org/

7. Posix. http://standards.ieee.org/regauth/posix/.
8. *FLEXChip Signal Processor (MC68175/D)*, Motorola, 1996.
9. B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni. Pnuts: Yahoo!'s hosted data serving platform. Proc. VLDB Endow., 1(2):1277–1288, 2008
10. Javaid, A. Q. Niyaz, W. S., & Alam, M. A.: Deep learning approach for network intrusion detection system, In Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol. (BIONETICS), 21-26 (2016).
11. Kim, J. J., Kim, H. L., Thu, T., & Kim, H.: Long short term memory recurrent neural network classifier for intrusion detection, In Proc. Int. Conf. Platform Technol. Service (PlatCon), 1-5, (2016).
12. Ali, F. A. B. H., & Len, Y. Y.: Development of host based intrusion detection system for log files, In Proc. IEEE Symp. Bus., Eng. Ind. Appl. (ISBEIA), 281-285 (
13. Topallar, M. M., Depren, O., Anarim, E., & Ciliz, K.: Host-based intrusion detection by monitoring Windows registry accesses,'' in Proc. IEEE 12th Signal Process. Commun. Appl. Conf., Apr. 728–731 (2004)
14. Pise, N.: Application of machine learning for intrusion detection system. Information Technology in Industry, 9(1), 314-323 (2021)
15. Hsu, Chih-Yu, Shuai Wang, & Yu Qiao.: Intrusion detection by machine learning for multimedia platform." Multimedia Tools and Applications 80(19), 29643-29656 (2021).
16. Jayasri, P., Atchaya, A. M. Sanfeeya P., & Ramprasath, J.: Intrusion detection system in software defined networks using machine learning approach." International Journal of Advanced Engineering Research and Science 8(4), (2021).
17. Bozcan, I., Oymak, Y., Alemdar, I. Z., & Kalkan, S.: What is (missing or wrong) in the scene? A Hybrid Deep Boltzmann Machine for Contextualized Scene Modeling, in 2018 IEEE International Conference on Robotics and Automation (ICRA), 1-6 (2018).
18. Deng, L.: Deep learning: Methods and applications, Found. Trends Signal Process. 7(3/4), 197-387 (2014).
19. Aghaei, E., & Serpen, G.: Ensemble classifier for misuse detection using N-gram feature vectors through operating system call traces,'' Int. J. Hybrid Intell. Syst., 14(3), 141–154 (2017).
20. Borisaniya, B., & Patel, D.: Evaluation of modified vector space representation using ADFA-LD and ADFAWD datasets, J. Inf. Secur., 6(3), 250, 2015.