Monitoring and Troubleshooting at Scale with Advanced Analytics

A Technical Paper prepared for SCTE by

Nitin Kumar
Principal Architect
Harmonic, Inc
2590 Orchard Parkway, San Jose, CA 95131
+1 408 542 2559
Nitin.Kumar@harmonicinc.com

Amir Leventer
Senior Director, System Architecture and Networking
Harmonic, Inc
2590 Orchard Parkway, San Jose, CA 95131
+1 408 542 2559
Amir.Leventer@harmonicinc.com

Asaf Matatyaou
Vice President, Solutions and Product Management, Cable Access
Harmonic, Inc
2590 Orchard Parkway, San Jose, CA 95131
+1 408 542 2559
Asaf.Matatyaou@harmonicinc.com

Table of Contents

Title **Page Number**

List of Figures

Title **Page Number**

1. Introduction

Access systems such as Cable Modem Termination Systems (CMTS) have traditionally been monitored using tools that leverage CLI and SNMP interfaces. These same interfaces are also used to gather live information while troubleshooting issues in the field. The data exposed by these interfaces are limited by the standard set of commands and MIBs supported by the system. With the move to a distributed access architecture (DAA), deployments can support much higher scale, and their software-centric designs make available richer data useful for vendors in operating, debugging, and optimizing the systems, but access to this increased amount of data is bottlenecked by the limited performance and limited extensibility of CLI and SNMP interfaces. Newer system designs support streaming logs and telemetry to overcome these limitations, and this paper looks at how these features not only overcome the limitations of traditional interfaces but also enable more efficient monitoring and troubleshooting workflows. We will show the building blocks of a system that implements streaming logs and telemetry. The performance and security improvements offered by such a system will be noted. We will illustrate some monitoring features and describe something we call “time-travel debugging” — using the streamed data for easier troubleshooting. Finally, we will give an overview of using the data for advanced analytics and machine learning, specifically for faster root cause analysis of field issues.

2. Monitoring and Troubleshooting at Scale with Advanced Analytics

Monitoring and troubleshooting traditional CMTS deployments can be a labor-intensive activity. Deployment health is monitored via SNMP or CLI scripts using metrics such as modem registration status or call volumes to customer support. Basic troubleshooting of common issues can be done by operations or vendor support personnel logging into each platform remotely to run monitoring commands and look at locally stored logs. Complicated issues may require direct involvement of development engineers to analyze logs and perform further troubleshooting steps. Some issues with significant service impact may occur only intermittently and may not be easily reproducible. Fault monitoring and resolution is mostly a manual process.

Compared with traditional architectures, the DAA allows deployments to be launched quickly, and operators can provide increased bandwidth to more subscribers. The troubleshooting load will grow with the scale of deployment, with a corresponding increase in operational overhead.

Monitoring and troubleshooting workflows can be augmented with tools that enable operators to handle the increased scale while keeping operational overhead under control. There are non-obvious but nevertheless important implications to vendors and operators, which will be explored in subsequent sections.

3. Troubleshooting

Let’s look at some of the challenges with troubleshooting production deployments at scale and explore some of the solutions possible with new tools and workflows.

3.1. Challenges

As deployments scale to handle increases in bandwidth and number of subscribers, there’s a corresponding increase in issues that occur and need to be debugged. Issues having significant operational impact can be quite complex to root-cause. Here is a list of potential issues:

- Transient cable plant problems

- Vendor-specific interpretation of DOCSIS specifications
- Non-compliant cable modems (sometimes with different firmware versions exhibiting different behavior on the same model)
- Incorrect handling of new DOCSIS features by older equipment
- Intermittently malfunctioning equipment
- Complicated interactions between specific configurations and plant population
- Combination of multiple issues

3.1.1. Limited Storage

With limited onboard storage on the CMTS, only a reduced number of data samples are stored for a short time span, severely hampering analytics and troubleshooting efforts that require a longer history of data. The logs pertaining to the issue may no longer be available, and troubleshooting efforts can be delayed until the issue occurs again, or effort is made to actively reproduce the issue. Because storage capacity is limited, only limited logging is enabled by default, and this means that logs are not as information-rich as they could be for some root-cause analysis.

3.1.2. Availability of Expertise

When logs are available, there may be only a limited number of engineers with the knowledge required to interpret the logs and correlate them with the observed issues. Thus, troubleshooting can be limited by availability of skilled personnel to handle the increased workload.

3.1.3. Security Considerations

Scale operations involve software running across many IP hosts, and this necessitates maintaining a list of IPs for support personnel to log in to. Login accounts need to be set up and/or passwords shared for debugging efforts. Managing access to these accounts and hosts need to be done carefully to avoid security issues. For example, the integrity of a deployment can be compromised by a single compromised account, or by a compromised host accessing a jump host.

3.1.4. Complexity of Issues

A DAA deployment has many hardware and software components. Problems can occur in any of these components, and some issues will be due to complex interactions between these components.

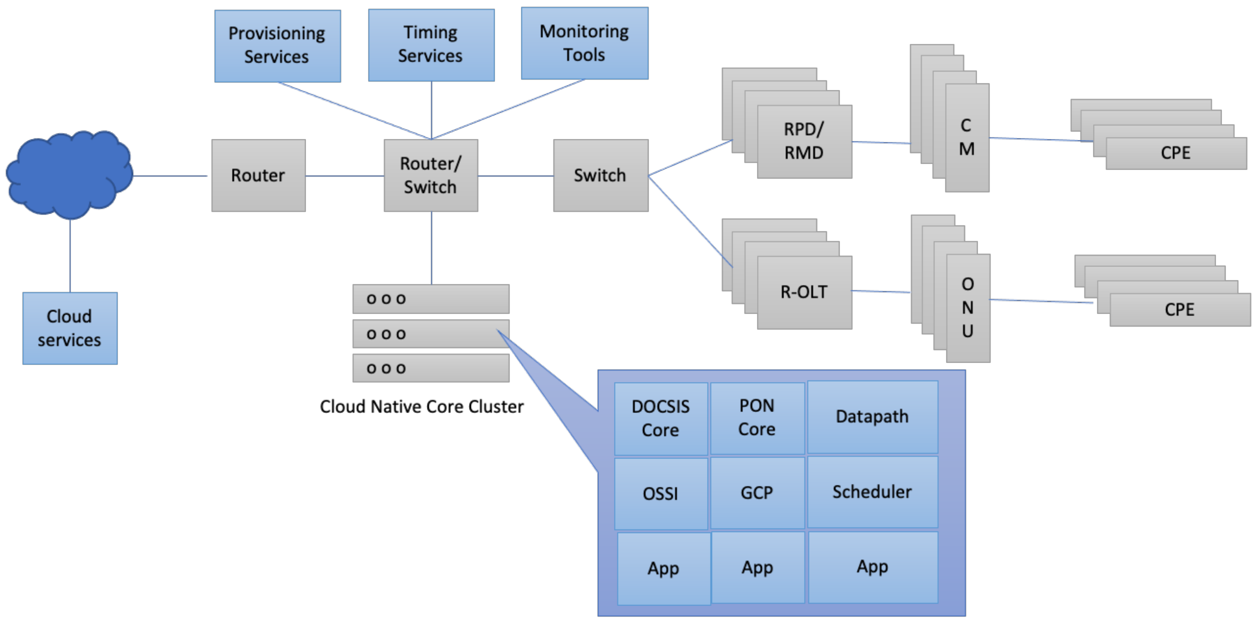


Figure 1 - A deployment has many possible fault sources

Logs and monitoring data from multiple sources must be analyzed to pinpoint the cause of a failure. Traditional troubleshooting processes can be overwhelmed by the volume of logs and metrics generated by the many software components of a modern cloud native solution. For instance, logs from multiple sources might need to be examined to troubleshoot a modem bandwidth problem. Generating a timeline of events across multiple system components will be challenging without new tools.

3.1.5. Legacy Monitoring Tools

Traditionally, the monitoring interface of a CMTS is defined by the DOCSIS OSSI standard, and tools such as CLI and SNMP are used to query the system to gather performance monitoring metrics. The OSSI standard does not specify data objects for all metrics that can be exposed by modern software designs, and CLI and SNMP tools do not scale well to handle vast amounts of fast-updating data. With increasing scale, using CLI for manual real-time analysis becomes impractical. A data pipeline constrained by these legacy tools limits the richness of data available for analysis and increases latency in detecting critical changes in key metrics.

3.2. Cloud Native Solutions

Cloud native, software-based solutions are now the popular field-proven choice for cost-effective, high-scale CMTS deployments. Service is provided by loosely coupled microservices deployed in containers running in a compute cluster. Each microservice exposes its own metrics and logs that provide highly granular insights into their operation to aid with advanced monitoring and troubleshooting. A vast amount of data is generated by the software components, and new tools are needed to process and consume the data.

3.2.1. Streaming Logs

Logs are streamed out to an external (cloud-based) service such as Elasticsearch. Analysis tools automatically monitor logs for known patterns and can generate alerts on critical events. Dashboards summarize log statistics and visualize events in time-series graphs, making it easier for operations personnel to monitor and troubleshoot systems. Search capabilities allow complex log patterns to be easily searched across multiple log sources.

3.2.2. Streaming Telemetry

Each software component exposes performance and state metrics that are continuously streamed to external services for monitoring, analysis, and visualization. Threshold-based alerting indicate when metric values are outside expected bounds, allowing operations personnel to investigate potential issues before they lead to failure. Dashboards help visualize the state of the system, and visual indicators draw attention to potentially problematic states.

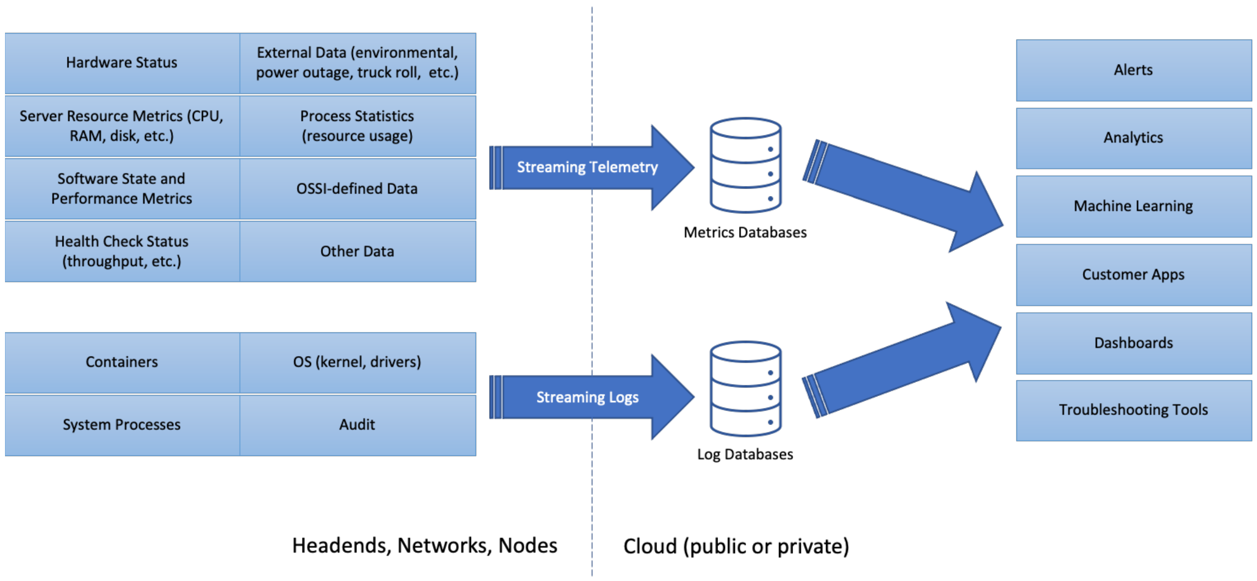

Hardware Status
External Data (environmental, power outage, truck roll, etc.)
Server Resource Metrics (CPU, RAM, disk, etc.)
Process Statistics (resource usage)
Software State and Performance Metrics
OSSI-defined Data
Health Check Status (throughput, etc.)
Other Data
Containers
OS (kernel, drivers)
System Processes
Audit
Streaming Telemetry
Streaming Logs
Metrics Databases
Log Databases
Alerts
Analytics
Machine Learning
Customer Apps
Dashboards
Troubleshooting Tools
Headends, Networks, Nodes
Cloud (public or private)

Figure 2 - Logs and metrics are streamed out and used for various applications

Streaming logs and telemetry mitigate the challenges with troubleshooting mentioned previously. The next sections will discuss how.

3.2.3. Scalability

The scalability of compute and storage resources in the cloud allows vast amounts of logs and metrics to be stored and analyzed. Since the data is continuously streamed off the system, local storage no longer limits availability of the data. Also, DAA deployments consist of many software components running across many servers, and tracking which server runs what software component while also figuring out which components are contributing to a field issue quickly becomes a problem if troubleshooting requires direct access to the server. With logs and metrics streamed out to accessible and external cloud-based services, such considerations are no longer an issue.

3.2.4. Automation and Analysis Tools

Logs and metrics are automatically analyzed for indicators of failure or violations of SLA guarantees. Alerts are automatically generated to draw attention of operations personnel to problematic areas. Data is visualized in dashboards and reports generated to ease understanding of system state without manually digging through logs or running debug commands. Search interfaces and interactive data exploration tools allow filtering out data irrelevant to a troubleshooting activity and drilling down to additional details in areas of interest. The data pipeline is continuously augmented with new analysis algorithms, dashboards, alerts, and other features based on experience and input from operations and domain experts so that troubleshooting activities are continuously being simplified. Since most of this runs in the cloud, adding new features to the pipeline do not generally require changes on core servers that might introduce risks such as increased CPU usage and core software bugs.

3.2.5. Security

Access to logs and metrics no longer require logging into a deployment instance since all debug and system state information are streamed out and made available in secure and globally accessible external systems. There may be many deployments streaming data to a few external entities. Managing access to these external entities is much simpler than managing direct access to deployments. Since logs and metrics are pushed out to the external systems, there are no incoming connections into the deployments, further enhancing security and simplifying firewall rules.

3.3. "Time Travel" Debugging

"Time Travel" here refers to being able to look at the state of the system at a point in time in the past while troubleshooting a problem.

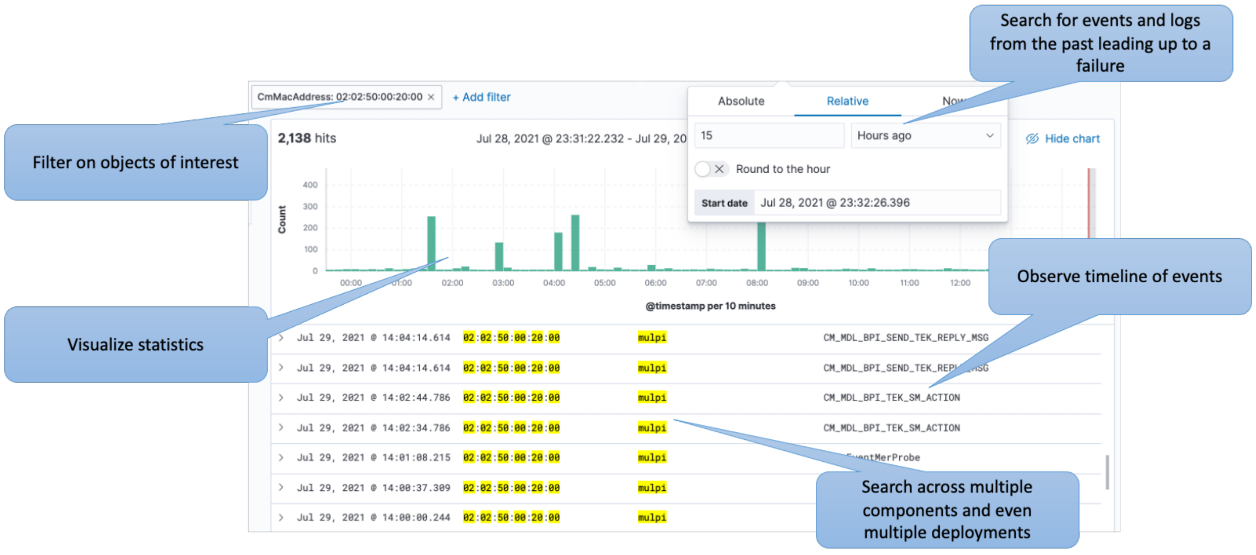
In many cases, the failure that needs to be diagnosed occurred in the recent past, and operations personnel might have already executed recovery procedures to restore service before support engineers have had a chance to access the system to gather information needed to root cause the failure. Without streaming logs and telemetry, most of the information needed to diagnose problems would be stored on the system itself. As previously mentioned, the amount of data stored is limited because of onboard storage capacity limitations. By the time someone is able to access the system, the logs relevant to the failure might have already been flushed from the system. Further, default logging levels might not have all the information needed to debug the problem. Since recovery procedures were executed, the system is no longer in a problematic state, and state information critical to determining the root cause of the failure has been lost.

Such situations lead to prolonged troubleshooting and root cause analysis efforts. Lack of system state and logs relevant to the failure would require waiting for the failure to occur again and being able to capture logs and system state leading up to the failure. Recording system state leading up to the failure can be challenging if that state is not captured in logs or by OSSI tools such as SNMP. Initial analysis might determine that additional logs or debug features need to be turned on, requiring further reproduction of the failure. Additionally, if live debugging is needed (requiring access to the system in the failed state), there could be prolonged service outages while the failure is being analyzed. Debugging under time pressures can result in erroneous analysis.

Streaming logs and telemetry mitigate these issues. Onboard capacity constraints are no longer a limiting factor, so logs are available in an external log database for extended time periods, and richer information can be logged by default (although there are cost implications; see later section). This means that the logs pertaining to a failure can be looked at, long after the failure occurred. Streaming telemetry makes periodic (many times a minute) state snapshots available in external databases, enabling the state of the system at any point in time to be analyzed. Operations personnel can execute recovery procedures without worrying about losing critical historical debug information.

In a sense, operations and support personnel and development engineers can "travel through time" to observe the state of the system at any point in the past. This can be done simply by using database query interfaces to search for information pertaining to the system at any time. Tools can show how various state metrics change over time leading up to a failure. This means that failures can be analyzed without having to wait for multiple reproductions of the issue.

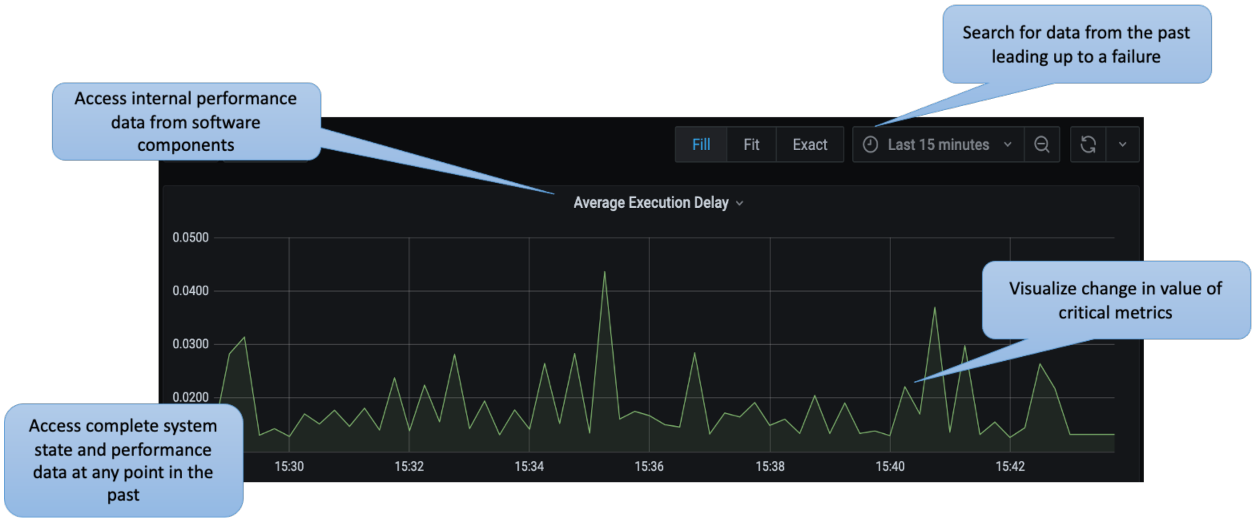
The state information streamed out is comprehensive and not constrained by the limited time resolution and information richness of CLI or SNMP data models, so almost any information needed to analyze the system at any point in time is always available. During root-cause analysis, various subsystems might need to be analyzed as the analysis progresses. Various domain experts will need to be brought in at different points of time. Since the streamed-out state is rich with information across all subsystems, there is seldom a situation where additional reproduction efforts are needed because information in some specific domains are lacking. Separate subsystems can be analyzed independently — each stage of the analysis can use the tools to explore system state across various subsystems over various periods of time. The result is faster root-cause analysis and minimal downtime.



Search for events and logs from the past leading up to a failure
Filter on objects of interest
Observe timeline of events
Visualize statistics
Search across multiple components and even multiple deployments
CmMacAddress: 02:02:50:00:20:00
+ Add filter
2,138 hits
Jul 28, 2021 @ 23:31:22.232 - Jul 29, 20
Absolute
Relative
Now
15
Hours ago
Round to the hour
Start date
Jul 28, 2021 @ 23:32:26.396
Hide chart
Count
@timestamp per 10 minutes

```
> Jul 29, 2021 @ 14:04:14.614  02:02:50:00:20:00  mulpi  CM_MDL_BPI_SEND_TEK_REPLY_MSG
> Jul 29, 2021 @ 14:04:14.614  02:02:50:00:20:00  mulpi  CM_MDL_BPI_SEND_TEK_REP  SG
> Jul 29, 2021 @ 14:02:44.786  02:02:50:00:20:00  mulpi  CM_MDL_BPI_TEK_SM_ACTION
> Jul 29, 2021 @ 14:02:34.786  02:02:50:00:20:00  mulpi  CM_MDL_BPI_TEK_SM_ACTION
> Jul 29, 2021 @ 14:01:08.215  02:02:50:00:20:00  mulpi  EventMerProbe
> Jul 29, 2021 @ 14:00:37.309  02:02:50:00:20:00  mulpi
> Jul 29, 2021 @ 14:00:00.244  02:02:50:00:20:00  mulpi
```

Figure 3 – Search and analysis of logs using tools such as Elasticsearch and Kibana



Search for data from the past leading up to a failure
Access internal performance data from software components
Visualize change in value of critical metrics
Access complete system state and performance data at any point in the past
Fill
Fit
Exact
Last 15 minutes
Average Execution Delay
0.0500
0.0400
0.0300
0.0200
15:30
15:32
15:34
15:36
15:38
15:40
15:42

Figure 4 - Data visualized and analyzed using tools such as Prometheus and Grafana

Looking into data from the past can help determine the reason of a current system fault. Current symptoms and logs and telemetry data can be compared with analysis from previous investigations and corresponding logs and telemetry signatures. For help with future investigations, a database of failure events and their corresponding logs and telemetry snapshots may be created. Frequency of each event can be tracked to generate correlations between the events and certain logs and metrics, and this can be used to isolate root causes to specific system components.

4. Advanced Analytics

The data available in the cloud can be used for other data-driven apps such as billing and analytics. Data from the past can be used to train predictive algorithms and fed into machine learning pipelines. Closed-loop controls can be implemented where external applications analyze the streamed data and call back into system configuration interfaces to make performance-optimizing adjustments. In this section, we will take a brief look at some of these applications.

4.1. Auto Observability

Observability of systems must scale with the number of deployments. Manually monitoring systems and taking manual corrective actions impede the ability to expand service footprint and operate at scale while maintaining SLAs. A robust data analytics pipeline can help with this.

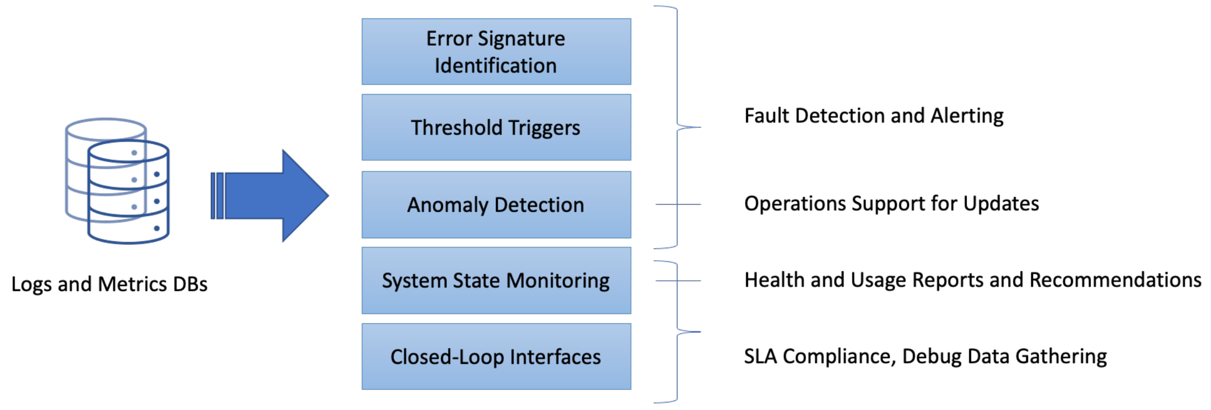

Logs and Metrics DBs
Error Signature Identification
Threshold Triggers
Anomaly Detection
System State Monitoring
Closed-Loop Interfaces
Fault Detection and Alerting
Operations Support for Updates
Health and Usage Reports and Recommendations
SLA Compliance, Debug Data Gathering

Figure 5 - Observability tasks are automated using data processing pipelines

4.1.1. Identify Failures

Comprehensive system state information is available in streamed telemetry and logs. Failures can be identified using basic data analytics (such as by identifying log signatures and metrics thresholds corresponding to failures) or by advanced approaches such as machine learning.

4.1.2. Rapidly Root-Cause Issues

At scale, failures must be analyzed quickly before they propagate and affect a large population of users. Failures detected during upgrades or operational updates must be diagnosed immediately to prevent stalling of planned operational procedures and rollouts. Potential issues introduced by an update can be identified by anomaly detection algorithms analyzing streamed logs and telemetry.

4.1.3. Proactive Debug Data Collection

Some investigations require more debug data than what's provided by logs and telemetry (such data might incur too much overhead to be streamed in logs and telemetry). For instance, an issue with specific kinds of data traffic might require analyzing a packet capture. Such data may be collected automatically when a potential issue is identified. This can be done by triggering advanced data collection when certain thresholds are exceeded. For example, a set threshold of subscriber-reported traffic problems can automatically trigger a traffic recording process for more advanced investigation.

4.1.4. Maintain SLAs

Performance metrics must be monitored across all deployments to ensure service quality. Monitoring tools must analyze streamed data to identify, report, and potentially correct problems as soon as they occur. Reports can identify subscribers who may benefit from moving to a different service tier. Closed-loop pipelines can observe streamed data and call into system configuration interfaces to take corrective action or adjust configuration to maintain SLA targets. For example, OFDM RxMER data from cable modems can be monitored to dynamically create and apply OFDM profiles optimized for current plant conditions. Having such closed-loop pipelines run in the cloud enables the associated algorithms to be easily updated and the performance of the algorithms themselves be monitored.

4.1.5. Characterize System Health

Data provided by streaming telemetry and logs must be analyzed to characterize system state and usage over time. The knowledge of system behavior provided by such analysis will aid in optimizing existing deployments and forecasting and planning for future needs. Automated reports can give an overview of system usage, and analysis pipelines can provide recommendations on managing deployments (such as recommending node splits based on bandwidth data or enabling features such as dynamic load balancing and equalization).

4.2. Multi-Variable KPI

Key Performance Indicators (KPIs) tracking customer satisfaction can be optimized for using data analysis and machine learning. As an example, bandwidth utilization could be a KPI, and optimizing it can involve tuning multiple parameters. RF parameters (modulation profile), subscriber density, cable modem configuration, and connectivity bandwidth, are just some of the things that contribute to optimal bandwidth usage. However, these parameters can also have adverse influence on some other metrics. Let's look at a couple of these parameters.

4.2.1. Modulation Profile

More robust modulation profiles may be configured when RF plant conditions deteriorate. Streaming telemetry includes RF metrics such as counts of uncorrectable and corrected FEC codewords, and this can be monitored to determine an appropriate modulation profile. A single-variable optimization would be to maintain the counts below a certain threshold. However, the more robust modulation profiles usually reduce available bandwidth as well. If the optimization is based on just optimizing the FEC-related counts, bandwidth utilization is adversely affected.

4.2.2. Subscriber Density

More per-user bandwidth can be made available by reducing the number of users sharing the RF spectrum. This involves a node-split. Bandwidth utilization is reported via telemetry, and peak bandwidth can be used as a threshold to decide on splitting a node. Additional nodes imply additional cost; therefore, optimizing on just the peak bandwidth can adversely affect cost.

Thus, monitoring and optimizing a single parameter can adversely affect other KPIs. What's needed is optimization across multiple variables: configuration must be dynamically adjusted to maximize utility over multiple metrics such as FEC, bandwidth utilization, and cost. One method of achieving this is by using a machine learning pipeline based on reinforcement learning.

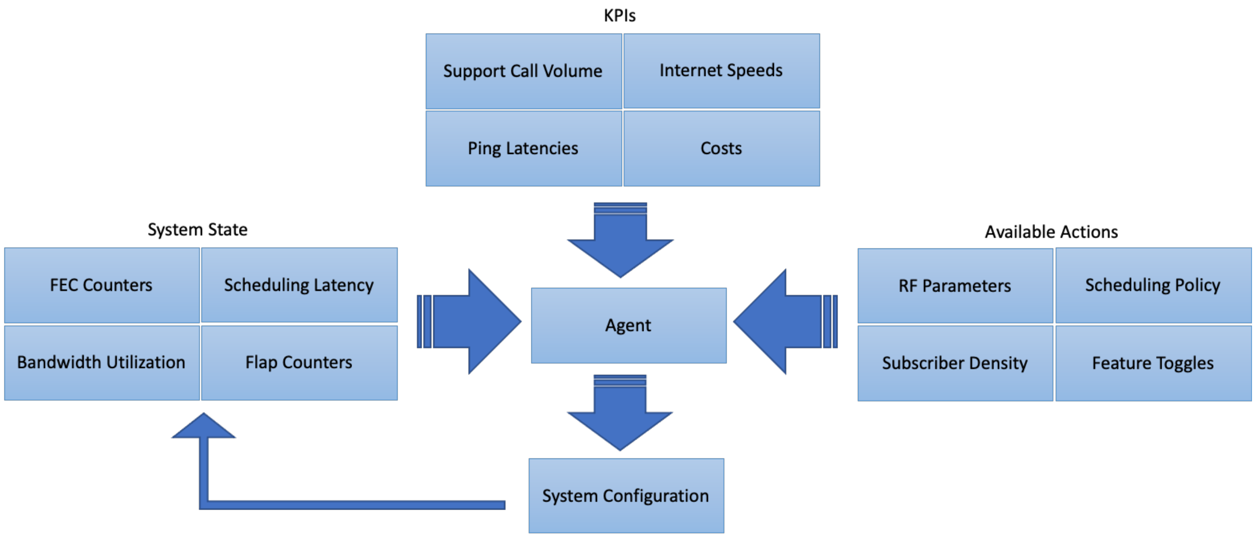


Figure 6 - Machine learning pipelines optimize key performance indicators

4.3. Fault Isolation

A DAA deployment contains many hardware elements (RPD, core servers, RF plant components, network switches and routers, etc.) and software components (RPD software, core software, software on network elements, OSSI tools, etc.). Each of these components are further composed of many subsystems, and many components are duplicated for redundancy. The overall system can be quite complex, with many potential points of failure. When a problem occurs, it can be challenging to pinpoint the source.

As an example, consider a case where some cable modems go offline. The root cause could be plant RF issues, a power outage event, a transient failure on a node or RPD or RF chassis, a network issue, a software crash, a hardware resource issue, or any number of other possibilities. A first step in analyzing the failure would be to identify the point in the topology where the failure originated. If the root cause is a power failure, then the metrics for multiple components would show a problem depending on the affected outage area. Identifying the cause of failure is not easily done just by looking at system dashboards. However, an analytics pipeline that is fed all the relevant data, including data from power supplies and third-party data sources such as utility companies, can correlate failures to the observed events, and combined with topology information, indicate the origin of the failure.

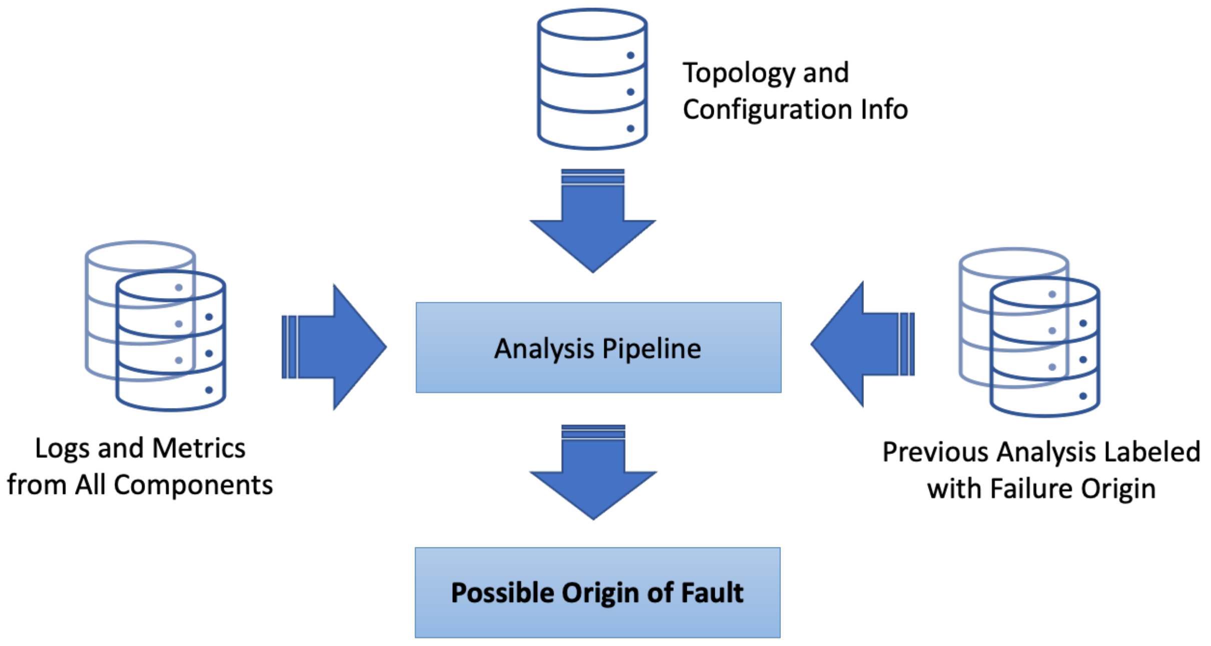

Topology and Configuration Info
Logs and Metrics from All Components
Analysis Pipeline
Previous Analysis Labeled with Failure Origin
Possible Origin of Fault

Figure 7 - Logs and metrics signatures indicate fault origin

Past analysis efforts can be used to automate future identification of root causes. Streamed data is continuously analyzed for patterns and anomalies, and problems identified in the past can be used to label the data and used to train machine learning algorithms. The algorithms can then automatically identify failures and indicate the root cause in future occurrences.

5. Other Considerations

Data analysis and machine learning enable tools that provide powerful new capabilities. There are some potential issues that need to be considered when deploying these tools at scale. This section will examine a couple of them.

5.1. Data Privacy

Logs and telemetry data can contain privacy-sensitive data. Information such as cable modem registration attempts, subscriber data usage, subscriber data activity, IP and MAC addresses, and other identifiers can potentially be combined with external data sources to gather information about subscribers. It is best to not collect such data in logs and telemetry. Identifiers must be sanitized, and any data collected must be anonymous. Data collection practices must be guided by both moral and legal (such as GDPR) considerations.

5.2. Infrastructure Cost

The most convenient and scalable method for running the components of the analytics pipeline discussed in this paper involves use of public cloud infrastructure. The associated cost can be significant if not managed well. It can be challenging to assign a dollar amount to the value provided by the pipeline so that a budget can be allocated for the public cloud services. For instance, the value is close to nil for the troubleshooting use case when everything works and there are no problems to be debugged. However, the services provided by the pipeline can be invaluable when a complex issue occurs that would have otherwise required the operator's or vendor's most experienced engineers to spend significant chunks of time analyzing and root causing the issue. The cost analysis is further complicated by the fact that this time spent troubleshooting takes away valuable time from those engineers.

The features provided by the pipeline, such as smart alerting and usage reports with data analysis, can significantly improve the operator's customer satisfaction metrics. The value provided by such features also needs to be considered when assessing the cost.

Regardless of the challenges assigning a value to the services, there are some knobs that may be used to control the costs.

5.2.1. Compute vs. Storage

The most expensive cost component is cloud compute resources. The data streamed to the cloud needs to be processed for analysis and display. For instance, compute resources are needed to index logs to enable fast searches. On the other hand, storing the data is relatively inexpensive. The pipeline can be tweaked to store raw data and use compute resources only on demand. The main drawback of this strategy is increased latency in accessing the data.

5.2.2. On-Prem vs. Public Cloud

Running the services on-premises instead of using a public cloud provider can reduce operational expenditure (depending on how the cost associated with operating the on-prem infrastructure is accounted for), with the caveat that additional overhead is incurred maintaining the on-prem hardware and software. Service expense is traded for capital expenditure. Drawbacks of this approach include additional difficulty in aggregating and analyzing data across multiple deployments, and more challenging upgrade processes and infrastructure scalability. In general, on-prem infrastructure might be a good option if all use cases are well-defined and already implemented and scale of operations is known in advance. Machine learning and advanced data applications are still evolving and use of public cloud infrastructure can provide flexibility in accommodating the changing use cases.

5.2.3. Data Retention

Storage costs can be managed by limiting the amount of data that is retained. For example, data older than a certain age can be discarded. Some architectures support a warm or cold storage option where older data is stored in cheaper (but higher latency) storage mediums. Discarding older data has the disadvantage that longer-term seasonal patterns may be missed by data analysis tools; this may be mitigated by feature generation capturing traits of the data and storing that longer term.

6. Conclusion

In this paper, we looked at how modern data analysis and machine learning tools help with operating critical infrastructure at scale. We examined the challenges posed by legacy tools and systems and how those challenges are mitigated by software-centric solutions that support streaming logs and telemetry. We explored how data and tools can be used as a framework for powerful troubleshooting capabilities that simplify analysis of failures. We looked at some of the applications made possible by using automated data analysis and machine learning algorithms. We also looked at some potential issues that need to be considered when deploying these applications at scale.

The applications and use cases presented in this paper are just a sample of what's made possible by rich data sets and advanced data analytics. As deployments scale to support more subscribers and bandwidth, such applications will be an essential component of successful operations and critical to ensuring customer satisfaction.

Abbreviations

CLI	command line interface
CM	cable modem
CMTS	cable modem termination system
DAA	distributed access architecture
DOCSIS	Data-Over-Cable Service Interface Specifications
FEC	forward error correction
GDPR	General Data Protection Regulation
IP	Internet Protocol
KPI	key performance indicator
MAC	media access control
MIB	management information base
OSSI	operations support system interface
OFDM	orthogonal frequency division multiplexing
RF	radio frequency
RPD	remote PHY device
RxMER	receive modulation error ratio
SLA	service-level agreement
SNMP	simple network management protocol

Bibliography & References

Deep Learning: A Visual Approach, Andrew Glassner; No Starch Press

Elasticsearch: The Definitive Guide, Clinton Gormley & Zachary Tong; O'Reilly Media

Practical Time Series Analysis, Aileen Nielsen; O'Reilly Media

Cloud Native Definition, Cloud Native Computing Foundation

DOCSIS Specifications, CableLabs