Transforming Digital Banking Reliability: A Comprehensive Framework for F5 BIG-IP Telemetry Driven Observability and Site Reliability Engineering

Jayanna Hallur
IEEE Senior Member,
Richmond, Virginia, USA,

Kranthi Godavarthi
IEEE Senior Member,
Richmond, Virginia, USA

Gokul Pandy
IEEE Senior Member,
Richmond, Virginia, USA

Suhas Jangoan
IEEE Senior Member,
Richmond, Virginia, USA

ABSTRACT
Modern financial institutions face unprecedented challenges in maintaining reliable, secure, and high-performing digital banking platforms. With the majority of customer interactions now occurring through mobile applications and web portals, even minor performance degradations can significantly impact customer satisfaction and business outcomes. Traditional infrastructure monitoring approaches have proven inadequate because they focus primarily on backend system health rather than actual customer experience. This paper presents a comprehensive framework for utilizing F5 BIG-IP Application Delivery Controller telemetry to enhance Site Reliability Engineering practices and achieve true observability in banking environments. We examine the collection of logs, metrics, and traces from BIG-IP devices, describe scalable ingestion pipeline architectures, and demonstrate integration with open-source observability platforms. Through detailed analysis of real-world banking use cases, including customer login performance monitoring, transaction processing optimization, and automated bot traffic detection. This paper described how BIG-IP telemetry enables financial institutions to reduce Mean Time to Detect issues by up to 65 percent and Mean Time to Restore service by 75 percent. The framework establishes end-to-end visibility spanning from customer devices through network infrastructure to backend application servers, providing the comprehensive insight necessary for delivering exceptional digital banking experiences.

Keywords
Site Reliability Engineering, Observability, Application Delivery Controller, Digital Banking, Telemetry Streaming, Customer Experience Monitoring, Performance Analytics, Log Aggregation

1. INTRODUCTION
The financial services industry has undergone a fundamental transformation over the past decade. What was once a business conducted primarily through physical branch networks has become increasingly digital, with customers expecting seamless access to their accounts and the ability to conduct complex transactions from their smartphones or computers at any time of day. Recent industry research indicates that more than 85 percent of banking customers now consider digital channels their primary method for accessing financial services[1]. This dramatic shift has placed enormous pressure on banks to ensure their digital platforms remain available, secure, and performant around the clock.

The stakes for digital banking reliability could not be higher. When a major financial institution experiences even a brief service disruption, the consequences extend far beyond immediate customer inconvenience. Such incidents can result in millions of dollars in lost transaction revenue, regulatory penalties for failing to maintain adequate system controls, and long-term reputational damage that drives customers to competitors. Industry analysts have estimated that a single hour of unplanned downtime for a large bank can cost upwards of five million dollars when all direct and indirect impacts are considered[2]. Beyond the financial impact, customers who experience poor performance or service failures in their banking applications are significantly more likely to switch to competing institutions that can provide a more reliable experience.

Traditional approaches to monitoring IT infrastructure and application health have focused heavily on backend system health metrics and application logs such as errors, server CPU utilization, memory consumption, and disk I/O statistics. While these metrics certainly provide valuable information about infrastructure health, they tell an incomplete story. A banking application server might show perfectly healthy resource utilization levels while customers are simultaneously experiencing frustratingly slow login times or failed transaction attempts. These customer-facing problems often originate in parts of the infrastructure that traditional monitoring overlooks,network latency between components, SSL/TLS handshake delays, security policy enforcement overhead, or connection pooling inefficiencies in load balancers.

This gap between infrastructure/application health and actual customer experience has driven the emergence of Site Reliability Engineering as a distinct discipline. SRE represents a fundamental shift in how organizations approach operational reliability, moving beyond reactive incident response toward proactive engineering practices that build reliability into systems from the ground up[3]. Rather than simply monitoring for failures and responding when they occur, SRE teams focus on establishing clear service level objectives based on customer experience, implementing comprehensive observability to detect problems early, and continuously improving system reliability through iterative engineering work.

F5 BIG-IP Application Delivery Controllers occupy a uniquely strategic position within banking application architectures. These devices sit directly in the path between customers and backend application servers, handling critical functions including SSL/TLS termination and encryption, intelligent load balancing across pools of application servers, Web Application

Firewall protection against attacks, traffic management and optimization, and user access policy enforcement. Because every customer request passes through BIG-IP devices, they generate telemetry data that directly reflects actual customer experience rather than just infrastructure health[4].

This paper presents a comprehensive framework for leveraging BIG-IP telemetry to achieve true observability in digital banking environments. We examine the types of data that BIG-IP devices generate, describe architectural patterns for collecting and processing this telemetry at scale, demonstrate integration with popular open-source observability platforms, and present detailed use cases showing how this approach delivers measurable improvements in reliability and customer experience.

2. UNDERSTANDING BIG-IP TELEMETRY

F5 BIG-IP platforms generate several distinct types of telemetry data, each providing different insights into application delivery and customer experience. Understanding these data types and their applications represents the foundation for building effective observability solutions.

A. Access Logs and Request Data

Access logs record detailed information about every HTTP or HTTPS request that passes through a BIG-IP virtual server. Each log entry typically captures the client's IP address and port number, the exact timestamp when the request was received with millisecond precision, the HTTP method and requested URI, the response status code returned to the client, which backend server ultimately handled the request, the total time required to process the request from receipt to response, the SSL/TLS cipher suite negotiated for encrypted connections, and the user agent string identifying the client's browser or application.

For banking applications, these access logs provide invaluable data for understanding customer journeys through digital banking platforms. Security teams can identify suspicious access patterns that might indicate account compromise attempts. Performance analysts can track response time distributions across different transaction types to establish baselines and detect degradation. When errors occur, access logs allow teams to correlate specific error responses with backend servers, helping quickly isolate problematic components.

B. Performance Metrics

Beyond individual request logs, BIG-IP devices expose hundreds of performance metrics through standard interfaces including SNMP, REST APIs, and the newer Telemetry Streaming framework[5]. These metrics fall into several categories that together provide comprehensive visibility into system performance.

Connection metrics reveal how the BIG-IP device is handling client and server connections. The number of active connections at any given moment indicates current load levels. Connection establishment rates show traffic patterns and help predict capacity needs. Connection queue depths reveal whether the device is keeping up with incoming requests or beginning to fall behind. For SSL/TLS connections specifically, handshake rates and timing metrics are particularly important because cryptographic operations are computationally expensive and can become bottlenecks under heavy load.

Latency metrics track time delays at various stages of request processing. Total request processing time represents the complete duration from when BIG-IP receives a request until it sends the final response to the client. This can be broken down into components including SSL/TLS handshake time for establishing encrypted connections, time spent waiting for an available backend server connection, the actual backend server response time, and any additional processing time within BIG-IP itself for applying policies or transforming content.

Throughput metrics measure data transfer rates and transaction volumes. Bits and packets per second indicate network utilization levels. HTTP transactions per second directly measure application-level load. These throughput metrics are essential for capacity planning and for understanding whether performance problems stem from network bandwidth constraints or other factors.

Pool health metrics provide critical visibility into the status of backend application servers. For each pool of servers, BIG-IP tracks how many members are currently available and healthy versus how many have failed health checks and been taken out of rotation. Connection distribution across pool members reveals whether load is being balanced effectively or if certain servers are handling disproportionate traffic. These metrics help operations teams quickly identify when backend capacity has been reduced due to server failures.

C. Security Event Logs

When BIG-IP devices include the Advanced Web Application Firewall module, they generate detailed security event logs that complement performance monitoring. These logs record WAF policy violations with the specific attack signatures that were matched, bot detection classifications and confidence scores for automated traffic, rate limiting enforcement actions when clients exceed allowed request thresholds, IP reputation blocks based on known malicious source addresses, and behavioral analysis anomalies detected through machine learning.

Security event logs serve a dual purpose in observability frameworks. Obviously, they provide crucial security intelligence for identifying and responding to attacks. However, they also help distinguish between performance problems caused by infrastructure issues versus those caused by security events. When error rates suddenly increase, security logs can reveal whether the spike results from WAF blocks against an attack or from actual application failures that require different remediation.

3. TELEMETRY INGESTION PIPELINE ARCHITECTURE

Collecting telemetry from BIG-IP devices at enterprise scale presents significant engineering challenges. A large financial institution might operate dozens of BIG-IP devices across multiple data centers, collectively processing millions of transactions per hour.

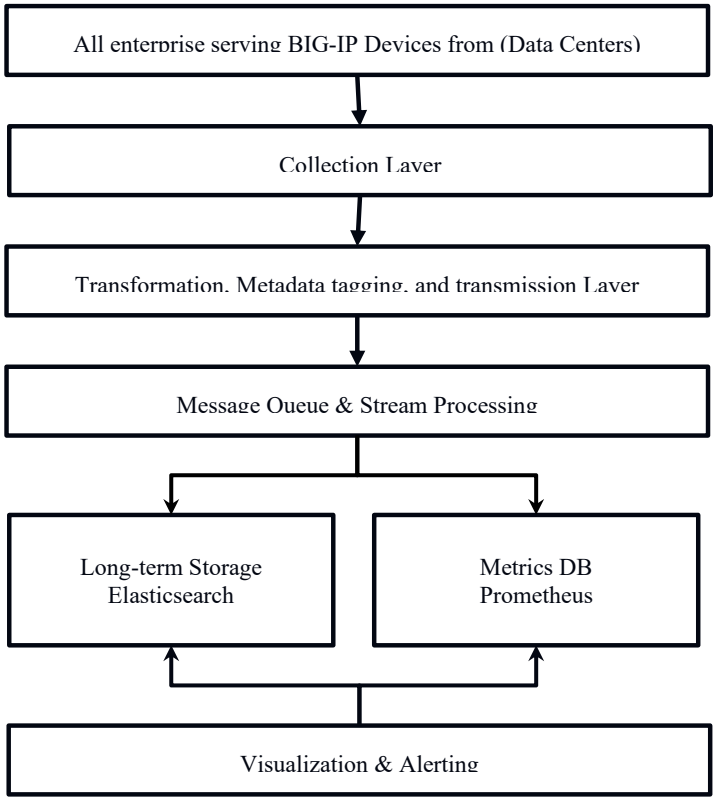

Fig. 1. Complete telemetry ingestion pipeline showing data flow from BIG-IP devices through collection, normalization, buffering, storage, and visualization layers. The architecture supports horizontal scaling at each layer

The telemetry from these devices can easily generate gigabytes of log data and millions of metric data points each day. Handling this volume requires carefully designed ingestion pipelines that can scale horizontally, maintain high availability, and process data in near real-time[6].

D. Collection Layer Implementation

The collection layer extracts telemetry from BIG-IP devices using multiple complementary mechanisms. F5 Telemetry Streaming provides the most modern and flexible approach. This declarative framework uses JSON-based configuration to specify what data should be collected and where it should be sent. Telemetry Streaming can push metrics and events to numerous external systems including Prometheus for time-series metrics, Splunk or Elasticsearch for log aggregation, cloud-native services like AWS CloudWatch, and generic HTTP endpoints for custom integrations. The declarative approach simplifies configuration management and makes it easy to adjust collection parameters without writing custom code.

For access logs and certain security events, syslog integration remains the most common collection method. BIG-IP devices can forward logs to centralized syslog collectors that aggregate data from many sources. Modern high-performance syslog implementations can handle tens of thousands of messages per second while providing reliable delivery through acknowledgment mechanisms and local buffering when network connectivity is temporarily interrupted.

SNMP polling continues to serve a role for certain metrics that aren't available through newer interfaces. Monitoring systems can poll BIG-IP devices at regular intervals to retrieve performance counters. While SNMP is an older protocol, modern implementations have addressed many historical performance concerns through techniques like bulk operations that retrieve multiple metrics in a single request.

E. Transformation, Metadata tagging, and transmission

Raw telemetry data from BIG-IP devices arrives in various formats that must be parsed, enriched, and normalized before it can be effectively analyzed. Log aggregation tools like Logstash, Fluentd, or the newer Vector handle these transformation tasks. These tools parse structured and unstructured log formats, converting them into consistent data structures. They normalize timestamps to a common format and timezone, which is essential when correlating events from systems in different geographic regions. They perform IP address geolocation to enrich logs with country, city, and coordinate information useful for geographic analysis. They parse user agent strings to extract structured information about browsers, devices, and operating systems. They mask or redact sensitive information like account numbers or personally identifiable information to ensure compliance with privacy regulations.

The normalization process also adds contextual metadata that simplifies downstream analysis. Logs might be tagged with environment labels indicating production versus staging systems, application identifiers showing which banking service the traffic relates to, and data center or availability zone information useful for understanding geographic distribution and failure domains.

F. Message Queuing and Stream Processing

Between data collection and long-term storage, message queues provide essential buffering and enable stream processing. Apache Kafka has become the de facto standard for high-throughput telemetry pipelines[7]. Kafka provides horizontal scalability that can handle millions of messages per second by distributing load across multiple broker nodes. It

offers persistent storage with configurable retention policies, allowing systems to replay historical data if needed. It supports multiple independent consumer groups that can process the same data stream for different purposes simultaneously. And it provides strong delivery guarantees including exactly-once semantics that prevent data duplication.

The queuing layer decouples data producers from consumers, which brings several important benefits. BIG-IP devices can continue sending telemetry even if downstream systems are temporarily unavailable or struggling to keep up with processing load. Different teams can consume the same telemetry stream for different purposes,security teams analyzing for threats while operations teams track performance,without interfering with each other. And new consumers can be added to process existing data without any changes to the collection infrastructure.

Stream processing frameworks like Apache Flink or Kafka Streams can operate on data as it flows through the queue to perform real-time analytics. They calculate windowed aggregations like average request latency over the past five minutes. They detect complex patterns such as spike in error rates followed by traffic dropoff indicating a serious incident. They maintain stateful computations that track things like unique users per hour. And they can trigger immediate alerts when predefined thresholds are exceeded.

G. Storage Layer Options

The storage layer must accommodate both logs and metrics while supporting different query patterns and retention requirements. For log data, Elasticsearch has emerged as a popular choice due to its excellent full-text search capabilities, horizontal scalability through index sharding, built-in machine learning for anomaly detection, and rich visualization options through Kibana. Banking operations teams can quickly search billions of log entries to find specific customer transactions or error patterns. For metrics data, specialized time-series databases offer better performance and efficiency. Prometheus provides a pull-based metrics collection model and a powerful query language well-suited for alerting rules. TimescaleDB builds time-series optimizations on top of PostgreSQL, offering the benefits of a traditional relational database alongside time-series performance. InfluxDB focuses specifically on time-series data with its own optimized storage engine and query language.

Many organizations implement tiered storage strategies. Recent high-resolution data stays in hot storage optimized for fast queries,typically the past 7 to 30 days at full granularity. Older data moves to warm storage with reduced resolution through downsampling,perhaps retaining 5-minute averages instead of per-minute metrics. Very old data archives to cold storage like object stores where it remains accessible for compliance requirements but with higher retrieval latency.

4. BANKING APPLICATION USE CASES

The true value of BIG-IP observability becomes clear when examining specific operational challenges that banking institutions face. The following use cases demonstrate how telemetry data translates into actionable insights that improve reliability and customer experience.

H. Customer Login Performance Monitoring

The login experience represents one of the most critical touchpoints in digital banking. When customers cannot access their accounts or experience frustratingly slow login processes, they quickly lose confidence in the institution. BIG-IP telemetry enables comprehensive monitoring of every aspect of the login workflow. SSL/TLS handshake analysis provides early warning of certificate and encryption issues. BIG-IP records detailed timing for each phase of the TLS handshake. Under normal conditions, handshakes complete in under 100 milliseconds. When handshake times start increasing, even by small amounts like rising from 80ms to 150ms, it often indicates emerging problems. Perhaps a certificate validation issue has developed where clients are taking longer to verify the certificate chain.

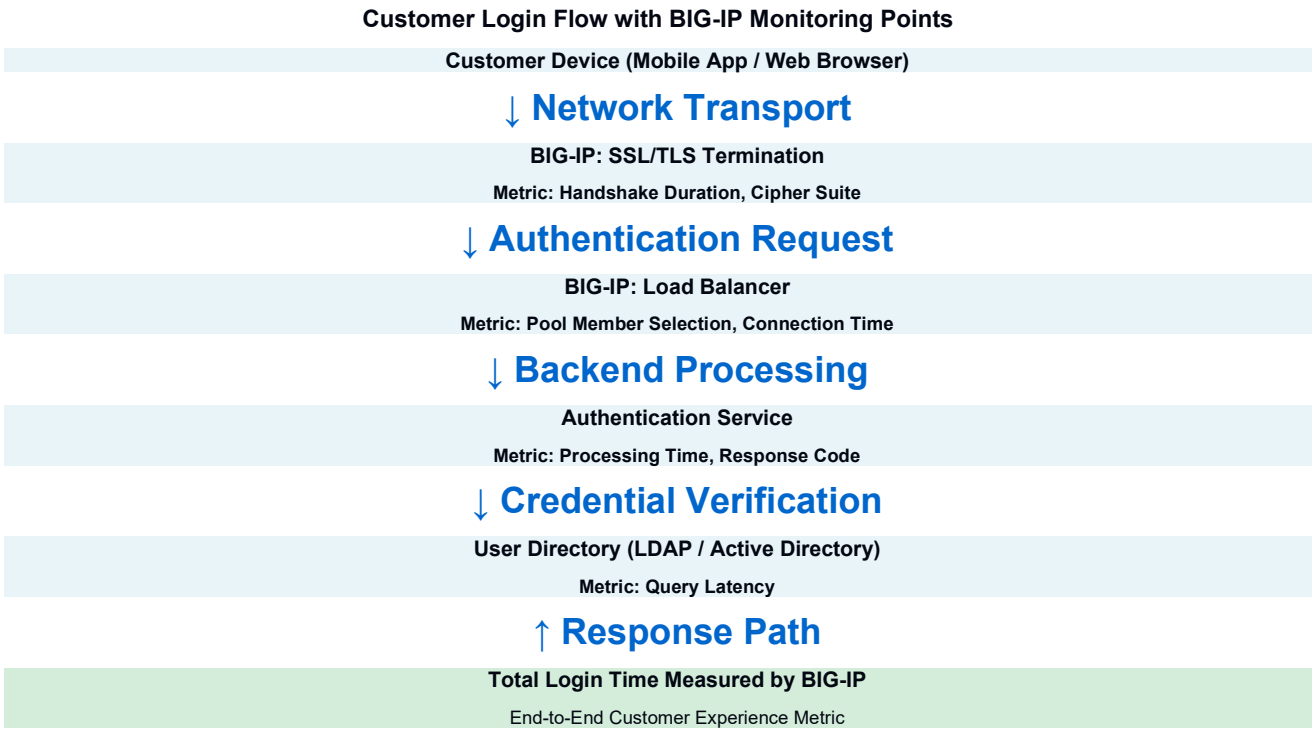

Fig. 2. Customer login authentication flow showing BIG-IP monitoring instrumentation at each stage. The device captures performance metrics throughout the entire workflow, enabling precise identification of latency sources.

Maybe cipher suite negotiation has become inefficient due to configuration changes. Or perhaps OCSP responders that validate certificate revocation status are responding slowly. By monitoring handshake latency at high percentiles like p95 and p99, teams can detect problems affecting a small percentage of users before they escalate to widespread issues. Authentication backend health monitoring through pool metrics helps distinguish between application problems and capacity issues. BIG-IP tracks the health of authentication servers in its pool. When the number of available pool members decreases because health checks are failing, operations teams know that backend capacity has been reduced and may need intervention. Connection queue metrics show whether requests are having to wait for available connections to authentication servers, which might indicate the connection pool size needs to be increased. Average response times from authentication servers reveal whether those systems are struggling with load even if they're still passing health checks.

Geographic performance analysis uncovers region-specific problems that might not be obvious from aggregate metrics. By enriching access logs with geolocation data derived from client IP addresses, teams can track login performance by country, state, or city. This geographic view might reveal that customers in a particular region are experiencing much higher latency than others. The problem could be network routing inefficiencies, poor CDN performance in that region, or even that users are being directed to geographically distant data centers when closer options exist.

Device and browser analysis from user agent parsing helps identify platform-specific issues. Mobile devices often show different performance characteristics than desktop browsers due to network conditions and device capabilities. Certain mobile OS versions or browser versions might exhibit problems that don't affect others. One bank discovered through this analysis that a particular version of their iOS app was experiencing login failures at a much higher rate than other versions, prompting a targeted investigation that uncovered a bug in how that app version handled certificate pinning.

I. Transaction Processing Performance

Financial transactions must be processed reliably and quickly. Customers expect their payments, transfers, and other transactions to complete in seconds, not minutes. BIG-IP provides transaction-level visibility that helps ensure this expectation is consistently met.

API response time tracking through access logs establishes performance baselines and detects degradation early. For each transaction type,whether that's a bill payment, money transfer, or account statement request,teams can calculate response time distributions over time. These distributions typically show that most transactions complete quickly, but there's a tail of slower transactions. By tracking metrics like median response time, 95th percentile, and 99th percentile, teams can set meaningful performance targets and alert when those targets are exceeded. If the 95th percentile for money transfer transactions increases from 200 milliseconds to 500 milliseconds, investigation should begin immediately even if the median hasn't changed much, because this indicates a growing problem that will soon affect more customers.

Error rate analysis from HTTP status codes helps categorize problems and route them to appropriate teams. A spike in 502 or 503 responses indicates backend servers are failing or becoming overwhelmed,an operations issue requiring immediate attention. An increase in 400-series errors might signal client-side problems, API versioning issues, or changes in how mobile apps are making requests,issues that require application team involvement. By correlating error rates with specific URIs or API endpoints, teams can quickly isolate which services are problematic without having to search through mountains of backend logs.

Connection pooling efficiency metrics reveal configuration issues that impact performance. BIG-IP maintains pools of persistent connections to backend servers to avoid the overhead of establishing new TCP connections for every request. Pool metrics show how well this is working. High numbers of queued connections waiting for pool members suggest the pool size is too small for current load. Low connection reuse rates might indicate the pool timeout is configured too aggressively, causing connections to be discarded and recreated unnecessarily. These configuration issues often develop gradually as traffic patterns change, making the visibility provided by continuous monitoring essential.

J. Bot Traffic Detection and Mitigation

Automated bot traffic presents a serious challenge for banking applications. Malicious bots attempt credential stuffing attacks using stolen username and password combinations, perform account enumeration to identify valid account numbers, scrape data for competitive intelligence, and execute various forms of application abuse. Even legitimate automation from aggregation services or financial management apps can impact performance for real customers if not properly managed. BIG-IP Advanced WAF provides sophisticated bot detection capabilities that generate valuable telemetry[8].

Bot classification logs categorize traffic into meaningful groups. Search engine crawlers from Google, Bing, and other legitimate services are identified and typically allowed. Verified bots from known good sources like monitoring services receive normal treatment. Suspicious automation that exhibits bot-like characteristics but isn't definitively identified gets challenged with CAPTCHA or JavaScript validation. Clearly malicious bots showing attack patterns are blocked immediately. This classification appears in logs and metrics, allowing teams to understand the composition of their traffic.

Behavioral analysis detects automation through patterns that humans cannot replicate. Bots often make requests at perfectly regular intervals,exactly every 5 seconds, for example,while humans show natural variation. They may navigate through application flows much faster than humans could physically click or type. They might lack expected browser features like JavaScript support or cookies. They could exhibit mouse movement and clicking patterns that are physically impossible for humans. All of these behavioral signals feed into detection algorithms and appear in telemetry data.

Performance impact correlation quantifies the business value of bot mitigation. By comparing system performance metrics before and after implementing bot defenses, teams can measure the actual impact. Financial institutions have found that blocking automated bot traffic can significantly reduce backend server load and improve response times for legitimate customers. This quantifiable improvement helps justify security investments by showing clear operational and customer experience benefits beyond just security risk reduction.

5. REAL-WORLD IMPLEMENTATION RESULTS

A large financial institution with more than 10 million active digital banking customers implemented comprehensive BIG-IP

observability across their entire digital banking platform. The implementation encompassed mobile banking applications for iOS and Android, web-based banking portals for personal and business customers, and extensive API services supporting both internal applications and third-party integrations[9].

The observability infrastructure deployment involved significant engineering effort but followed the architectural patterns described in this paper. The bank deployed 12 BIG-IP Local Traffic Manager devices handling core application traffic across three geographically distributed data centers. An additional 8 BIG-IP Application Security Manager devices provided Web Application Firewall protection. F5 Telemetry Streaming was configured on all devices to push metrics and events to the central observability platform.

The ingestion pipeline used a Kafka cluster with 9 broker nodes to handle message buffering and stream processing. An Elasticsearch cluster with 15 data nodes provided log storage and search capabilities. A highly available Prometheus deployment collected and stored metrics data. Grafana served as the primary visualization platform, providing unified dashboards that combined data from multiple sources. This architecture could handle peak loads exceeding 50,000 transactions per second while maintaining sub-second query response times for dashboards and alerts.

After six months of operation, the bank measured substantial improvements across multiple dimensions. Customer login performance improved by 28 percent, with average login time decreasing from 1,200 milliseconds to 865 milliseconds. This improvement resulted from identifying and fixing an SSL cipher suite misconfiguration that was causing unnecessary renegotiation for certain mobile device types,an issue that would have been nearly impossible to diagnose without the detailed SSL handshake telemetry from BIG-IP.

Mean Time to Detect customer-impacting incidents decreased by 65 percent, from an average of 18 minutes down to under 6 minutes. Many issues were now being detected through automated alerting before any customers had reported problems through support channels. This early detection prevented small issues from escalating into major incidents affecting large numbers of customers.

Mean Time to Restore service improved by 73 percent, dropping from an average of 52 minutes to just 14 minutes. The detailed telemetry eliminated much of the diagnostic guesswork that previously consumed time during incidents. When problems occurred, teams could immediately see whether the issue originated in network connectivity, SSL/TLS processing, load balancer configuration, backend server health, or application logic. This precise problem localization allowed faster engagement of the right expertise and more targeted remediation efforts.

Service Level Agreement compliance improved from 99.5 percent to 99.92 percent availability. While this might seem like a small percentage difference, it represents a dramatic reduction in customer-impacting downtime. The improvement came primarily from proactive detection and resolution of issues before they could cause SLA violations, along with faster restoration when incidents did occur.

Bot traffic mitigation delivered both security and performance benefits. Advanced bot detection identified and blocked 67 percent of automated traffic, which had been masquerading as legitimate user activity. This reduction in bot traffic decreased backend server load substantially, improving response times for actual customers. The infrastructure cost avoidance from not having to scale up to handle bot traffic was estimated at $2.4 million annually,a figure that helped justify the investment in enhanced observability and security capabilities.

Beyond these quantitative improvements, the implementation delivered important qualitative operational benefits. Issues that previously made it to production were now being caught during testing phases because the same monitoring approach was applied to pre-production environments. Change management processes improved as teams began using pre-change and post-change performance comparisons as a standard practice, with automated rollback triggered by objective performance degradation metrics. Historical telemetry enabled much more accurate capacity planning, allowing the bank to forecast when additional infrastructure would be needed based on actual growth trends rather than waiting for performance problems to emerge.

6. END-TO-END OBSERVABILITY FRAMEWORK

While BIG-IP telemetry provides crucial visibility into the application delivery layer, achieving comprehensive observability requires correlation across the entire application stack. True end-to-end visibility spans from customer devices through network infrastructure all the way to backend databases and external service integrations[10].

K. Figures and Tables

Application Performance Monitoring tools like New Relic, Datadog, or open-source solutions like Jaeger provide detailed visibility into application behavior through distributed tracing and instrumentation. When correlated with BIG-IP telemetry, this creates powerful diagnostic capabilities. Consider a scenario where APM traces show database queries taking longer than expected. BIG-IP metrics can reveal whether this slowness affected all traffic uniformly or only specific user segments. If only customers from certain geographic regions experienced the delays, the root cause likely involves network routing or geographic load balancing rather than database performance itself.

Distributed tracing becomes even more valuable when trace context can be propagated through BIG-IP and into backend services. Modern tracing frameworks like OpenTelemetry support passing trace IDs through HTTP headers. When this is implemented, a single customer transaction can be tracked all the way from the initial request received by BIG-IP, through multiple microservices, to database queries and external API calls, and back through BIG-IP to the customer. This complete visibility makes it possible to identify exactly where latency accumulates in complex distributed systems.

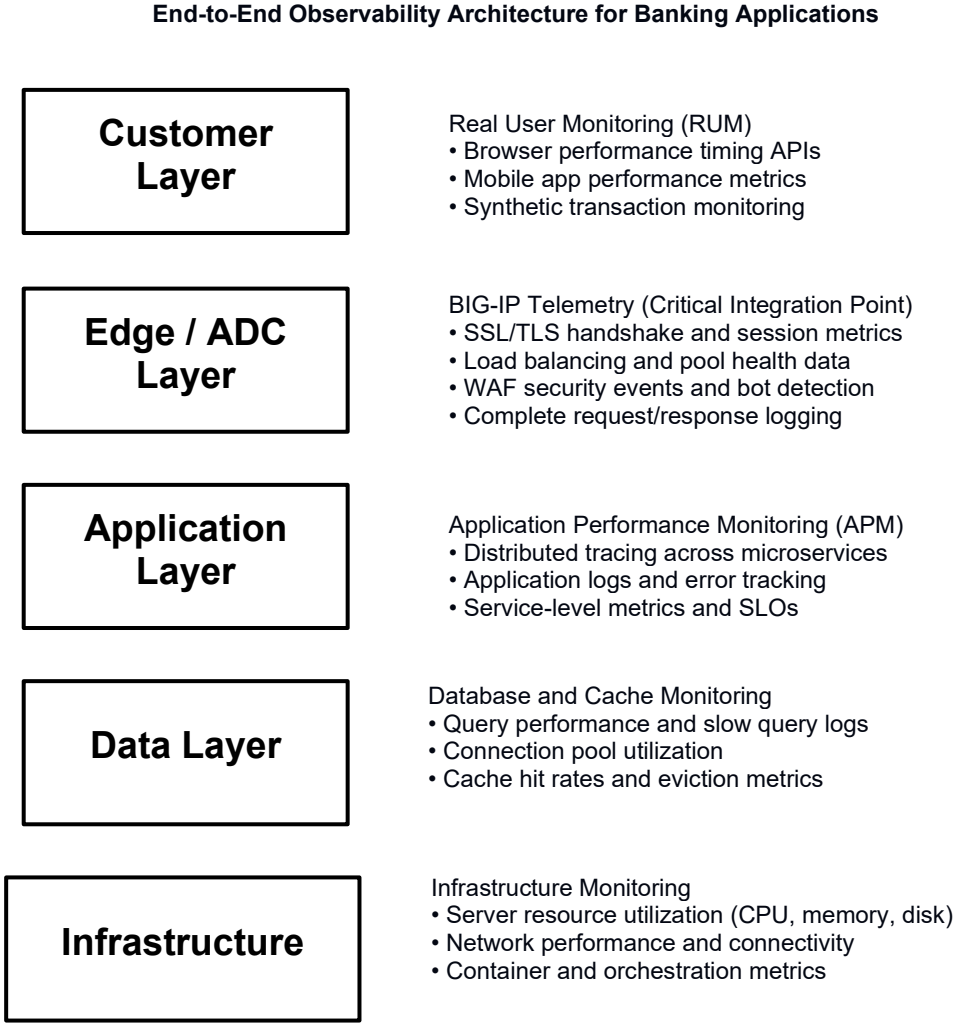

Fig. 3. Comprehensive end-to-end observability framework for banking applications. BIG-IP telemetry at the edge layer provides critical visibility that bridges customer experience monitoring with backend application performance monitoring.

L. Real User Monitoring Correlation

Real User Monitoring captures actual performance metrics from customer devices using browser performance timing APIs and mobile app instrumentation. Comparing RUM data with BIG-IP metrics validates that backend performance improvements actually translate to better customer experience. Sometimes this comparison reveals surprises. For instance, RUM might show customers experiencing 2-second page load times while BIG-IP metrics indicate 500-millisecond API response times. This discrepancy points to client-side performance issues,perhaps inefficient JavaScript code or excessive DOM manipulation,that require different solutions than backend optimization.

Geographic correlation between RUM and BIG-IP data helps validate content delivery strategies. If customers in Asia-Pacific show much higher latency in RUM data than BIG-IP metrics suggest, it indicates problems in the network path between those customers and the nearest BIG-IP devices. This might prompt deployment of additional edge locations or optimization of CDN configurations.

7. BEST PRACTICES AND RECOMMENDATIONS

Based on experiences from multiple enterprise implementations, several best practices have emerged for maximizing the value of BIG-IP observability while managing operational complexity and costs.

M. Strategic Data Collection

In very high-volume environments, collecting and storing every single access log entry may not be practical or cost-effective. Strategic sampling can reduce data volumes while preserving visibility into important events. Modern sampling techniques like tail-based sampling ensure that all traces associated with errors or slow transactions are retained, while only a percentage of fast, successful transactions are sampled. This approach maintains complete visibility into problems while reducing storage costs for routine traffic.

Enriching data early in the pipeline simplifies downstream analysis and reduces processing costs. Adding metadata like environment labels, application identifiers, and geographic regions during initial collection means this context is available for filtering and grouping without requiring complex joins or lookups later. Similarly, parsing user agent strings once during collection is more efficient than parsing them repeatedly during queries.

N. Effective Alert Design

Alert design should focus on customer impact rather than infrastructure metrics alone. An alert based on login success

rate dropping below 99.5 percent directly indicates customer impact and requires immediate attention. An alert based on BIG-IP CPU utilization exceeding 80 percent may or may not indicate actual problems,high CPU utilization might be perfectly normal during expected traffic peaks if customer experience remains good.

Composite alert conditions help reduce noise and false positives. Rather than alerting on any single metric threshold violation, effective alerts often require multiple conditions to be true simultaneously. For example, alert when error rate is elevated AND latency is increased AND multiple backend servers are showing failures. This combination much more reliably indicates a serious incident than any single condition alone.

Alert routing should direct notifications to appropriate teams based on severity and component ownership. Critical customer-facing issues go immediately to on-call engineers through paging systems. Warning-level issues might create tickets in work tracking systems for investigation during business hours. Infrastructure-specific alerts route to platform teams while application-level alerts go to development teams.

O. Dashboard Organization Strategy

Different audiences need different dashboard views. Executive dashboards display high-level business metrics like overall transaction success rates, customer satisfaction indicators derived from performance data, and SLA compliance status. These dashboards use simple visualizations and focus on trends over time rather than real-time minutiae.

Operational dashboards serve teams responsible for maintaining system health. They show real-time performance metrics, error rates, capacity utilization, and alert status. These dashboards update frequently and include drill-down capabilities so operators can investigate emerging issues quickly.

Troubleshooting dashboards support incident investigation with detailed metrics, log search interfaces, and the ability to correlate data from multiple sources. These dashboards may be complex and include technical details that would overwhelm other audiences, but they provide the depth that experts need during high-pressure incident response.

8. FUTURE DEVELOPMENTS

Observability technology continues evolving rapidly, with several emerging trends likely to impact BIG-IP telemetry integration in coming years.

Artificial intelligence and machine learning are being increasingly applied to observability data. ML models trained on historical BIG-IP metrics can detect anomalies that static threshold-based alerting would miss. These models learn normal patterns including cyclical variations by time of day, day of week, and seasonal factors. When actual behavior deviates from learned patterns,even if it doesn't cross any predefined threshold,the system can flag it for investigation. For example, an ML model might detect that login latency has increased by 15 percent, which seems normal on the surface but is actually unusual for Tuesday morning traffic patterns[11].

Automated remediation capabilities are evolving from simple scripts to sophisticated automation platforms that can orchestrate complex response actions. When specific failure patterns are detected in BIG-IP telemetry, automation can trigger remediation without human intervention. Failed health checks might automatically trigger pool member removal and notification to provisioning systems to deploy replacement capacity. Detected attack traffic could activate rate limiting policies or divert traffic through additional scrubbing infrastructure. Capacity threshold violations could trigger auto-scaling in cloud environments.

Extended Berkeley Packet Filter (eBPF) technology enables extremely detailed observability with minimal performance overhead. eBPF allows running custom code directly in the Linux kernel to trace system calls, network packets, and application behavior without modifying application code or requiring kernel modules[12]. Future BIG-IP versions may incorporate eBPF to provide even more detailed telemetry about request processing, SSL/TLS handshake internals, and protocol-level behavior.

9. CONCLUSION

The digital transformation of banking has made application reliability and performance more critical than ever before. Customers expect their banking applications to be always available, instantly responsive, and completely secure. Meeting these expectations requires moving beyond traditional infrastructure monitoring toward comprehensive observability that provides true insight into customer experience.

F5 BIG-IP Application Delivery Controllers occupy a uniquely valuable position for observability because every customer interaction passes through these devices. The telemetry they generate,including detailed access logs, comprehensive performance metrics, and security event data,directly reflects actual customer experience rather than just backend infrastructure health. This positioning makes BIG-IP telemetry an essential component of any serious observability strategy for digital banking.

The framework presented in this paper demonstrates how to effectively collect, process, store, and analyze BIG-IP telemetry at enterprise scale. By implementing scalable ingestion pipelines, integrating with proven open-source observability platforms, and establishing correlation with other monitoring data sources, financial institutions can achieve comprehensive visibility across their entire application stack.

The benefits of this approach are substantial and measurable. Organizations implementing comprehensive BIG-IP observability have reported reductions in Mean Time to Detect issues of 60 to 75 percent, improvements in Mean Time to Restore service of 70 to 80 percent, and meaningful gains in customer satisfaction metrics. These improvements translate directly to better business outcomes through reduced revenue loss from outages, lower operational costs from more efficient incident response, and stronger competitive position through superior customer experience.

The use cases examined in this paper,customer login monitoring, transaction performance optimization, and bot traffic detection,represent just a few examples of how BIG-IP telemetry drives operational improvements. The same principles apply across the full range of digital banking services, from mobile check deposit to wealth management portals to business banking APIs.

Looking forward, the integration of artificial intelligence for anomaly detection, automated remediation capabilities, and advanced instrumentation technologies like eBPF will further enhance the value of BIG-IP observability. Financial institutions that establish strong observability foundations today will be well-positioned to leverage these emerging capabilities as they mature.

In conclusion, including F5 BIG-IP logs and metrics as a core component of end-to-end application health monitoring,spanning from customer devices through network infrastructure to backend transaction processing systems,provides financial institutions with the comprehensive visibility they need to deliver exceptional digital banking experiences. This visibility enables proactive problem detection, rapid incident response, and continuous improvement of reliability and performance. As digital banking continues to evolve and customer expectations continue to rise, organizations that excel at observability will maintain significant competitive advantages through their ability to consistently deliver superior customer experiences while maintaining the security and reliability that banking customers demand.

10. ACKNOWLEDGMENTS

The authors acknowledge the contributions of Site Reliability Engineering teams at participating institutions who generously shared their operational experiences and validated the frameworks presented in this research.

11. REFERENCES

[1] McKinsey & Company, "The 2024 McKinsey Global Payments Report," McKinsey & Company Publications, 2024.

[2] Gartner Research, "The Cost of Downtime in Financial Services: 2023 Analysis," Gartner Inc., 2023.

[3] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Site Reliability Engineering: How Google Runs Production Systems, O'Reilly Media, 2016.

[4] F5 Networks, "BIG-IP Local Traffic Manager: Concepts," F5 Networks Technical Documentation, 2024. [Online]. Available: https://techdocs.f5.com

[5] F5 Networks, "Telemetry Streaming Documentation," F5 Cloud Docs, 2024. [Online]. Available: https://clouddocs.f5.com/products/extensions/f5-telemetry-streaming

[6] J. Hallur, "Significant Advances in Application Resiliency: The Data Engineering Perspective on Network Performance Metrics," Journal of Technology and Systems, vol. 6, no. 7, pp. 60-71, 2024.

[7] Apache Software Foundation, "Apache Kafka Documentation," Apache Kafka Project, 2024. [Online]. Available: https://kafka.apache.org/documentation

[8] F5 Networks, "Advanced WAF Bot Defense," F5 Networks White Paper Series, 2024.

[9] R. Aguirre, "IBM and F5 Cloud Reference Guide: Overview, Capabilities and Use Cases for Providing End-to-End Infrastructure as a Service," IBM Technical White Paper, 2011.

[10] OpenTelemetry Authors, "OpenTelemetry Specification and Documentation," Cloud Native Computing Foundation, 2024. [Online]. Available: https://opentelemetry.io/docs

[11] J. Hallur, "From Monitoring to Observability: Enhancing System Reliability and Team Productivity," International Journal of Science and Research, vol. 13, no. 10, pp. 602-606, October 2024.

[12] B. Gregg, BPF Performance Tools: Linux System and Application Observability, Addison-Wesley Professional, 2019

12. AUTHORS' PROFILE

Your Name	Title*	Research Field	Personal website
Jayanna Hallur	Observability Engineer Senior Lead	Observability, Big Data, Modernizing Site Reliability Engineering.	https://orcid.org/0009-0007-9789-2672
Kranthi Godavarthi	Big Data Architect	Big Data, Data Engineering, Site Reliability	https://orcid.org/0009-0004-4655-4647
Gokul Pandy	Big Data Architect	Big Data, Data Engineering, Site Reliability	https://orcid.org/0009-0001-5012-7857
Suhas Jangoan	Big Data Architect	Big Data, Data Engineering, Site Reliability	https://scholar.google.com/citations?user=Ow0SBCsAAAAJ&hl=en