World Journal of Advanced Engineering Technology and Sciences

eISSN: 2582-8266

Cross Ref DOI: 10.30574/wjaets

Journal homepage: https://wjaets.com/

(REVIEW ARTICLE)

Check for updates

Technical review: Edge-side Includes (ESI) for composition at CDN Level

Shafi Shaik *

Independent Researcher, USA.

World Journal of Advanced Engineering Technology and Sciences, 2025, 15(02), 2333-2340

Publication history: Received on 07 April 2025; revised on 19 May 2025; accepted on 21 May 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.15.2.0761

Abstract

Edge-side Includes (ESI) represents a transformative technology in web content delivery optimization, enabling developers to designate page elements as fragments that can be processed independently at Content Delivery Network (CDN) edges. This technical review explores how ESI functions as a markup language that facilitates fragment-based caching while allowing dynamic assembly closer to end users. The architecture of ESI features specialized processing engines integrated within CDN infrastructure that interpret directives embedded in HTML content. Implementation strategies covered include fragment identification methodologies, optimal caching policies with time-to-live (TTL) configurations, and edge processing considerations. The document highlights significant benefits such as improved performance through cached static content, reduced origin server loads, enhanced scalability during traffic spikes, efficient personalization support, and bandwidth optimization. Despite these advantages, challenges including architectural complexity, debugging difficulties, CDN dependencies, and developer learning curves must be addressed. Looking forward, ESI continues to evolve alongside emerging technologies like serverless edge functions, API-driven fragments, Web Assembly, and machine learning, while architectural shifts increasingly favor edge-first approaches that fundamentally transform application design principles.

Keywords: Content Delivery Networks; Edge Computing; Fragment-based Caching; Dynamic Content Assembly; Web Performance Optimization

1. Introduction

Edge-side Includes (ESI) represents a significant advancement in web content delivery optimization. This markup language allows developers to designate specific parts of web pages as fragments that can be processed independently at the Content Delivery Network (CDN) level. In today's dynamic web environment where personalization and real-time content are increasingly important, ESI offers a sophisticated solution to the challenge of delivering fast, responsive experiences while maintaining efficient caching strategies [1].

The fundamental concept behind ESI is straightforward yet powerful: by identifying which parts of a page change frequently and which remain relatively static, developers can implement fragment-based caching that significantly improves performance. Security researchers have identified both performance benefits and potential vulnerability concerns in ESI implementations. Contemporary edge delivery services implementing ESI reduced page load times by approximately 35% across high-traffic websites, with proper security controls in place [2]. When implemented correctly with appropriate input validation, ESI enables CDNs to cache and serve static content while dynamically assembling the final page at the edge—closer to the end user—resulting in reduced origin server load and improved page load times.

* Corresponding author: Shafi Shaik

2. Technical Architecture of ESI

2.1. Core Components

ESI functions through a specialized processing engine integrated within CDN infrastructure. This engine interprets ESI directives embedded within HTML content, typically as XML-like tags. Recent studies on cloud service optimization highlight that modern ESI processors achieve significant performance improvements through distributed processing architectures that balance workloads across edge nodes [3].

The ESI processor represents the cornerstone component, residing at the CDN edge and responsible for parsing and executing ESI directives. Production implementations demonstrate that processor efficiency correlates strongly with intelligent caching strategies that prioritize frequently accessed fragments. The origin server houses both base content templates and dynamic fragments, serving as the authoritative source when edge caches require revalidation. Cloud performance analysis indicates ESI implementations substantially reduce origin server load during traffic spikes by serving cached fragments from edge locations [3].

Edge caches store both processed fragments and assembled templates according to their respective TTL values. Security research emphasizes that these caches must implement robust validation mechanisms to prevent cache poisoning attacks that could otherwise compromise ESI-enabled applications [4]. Effective implementations maintain carefully calibrated per-fragment cache lifetimes based on content volatility and security risk profiles.

The fragment assembly engine orchestrates the final composition process, combining static and dynamic elements into cohesive responses. Research on CDN security frameworks suggests that assembly engines must implement strict content validation to mitigate potential injection vulnerabilities during the composition phase [4]. Advanced implementations utilize secure parsing techniques that maintain composition efficiency while ensuring content integrity throughout the assembly process.

2.2. Processing Workflow

The ESI processing workflow follows a meticulously orchestrated sequence designed to minimize latency while maximizing cache effectiveness. When a client request arrives at the CDN edge server, the system initiates fragment-based composition through a multi-stage pipeline. Cloud scalability research demonstrates that edge request routing improvements significantly enhance first-byte response times for template retrieval across distributed environments [3].

Once retrieved, the ESI processor parses the template and systematically identifies embedded ESI tags. Security analysis reveals this parsing stage represents a critical security boundary requiring thorough input validation to prevent potential directive manipulation attacks [4]. For each ESI directive identified, the processor makes cache-informed decisions, retrieving a substantial percentage of fragments from edge caches during normal operations.

Cache misses trigger origin requests, with comprehensive analysis showing performance variations based on geographic distribution and network conditions. Research across diverse content delivery architectures demonstrates that implementing fragment-specific security controls provides protection against both performance and security threats during origin retrieval [4].

The assembly phase represents the final critical stage, where the processor methodically combines all fragments into a coherent HTML response. Cloud performance benchmarks indicate that optimized assembly algorithms significantly reduce composition overhead while maintaining security boundaries between fragments [3]. Security frameworks emphasize that proper implementation of context-aware validation during assembly provides essential protection against cross-fragment exploitation attempts.

Advanced implementations have further enhanced this workflow through resilient architecture patterns. Recent research highlights how fault-tolerant designs with graceful degradation capabilities maintain partial functionality even when specific fragments become unavailable, balancing user experience against security requirements in distributed edge environments [3].

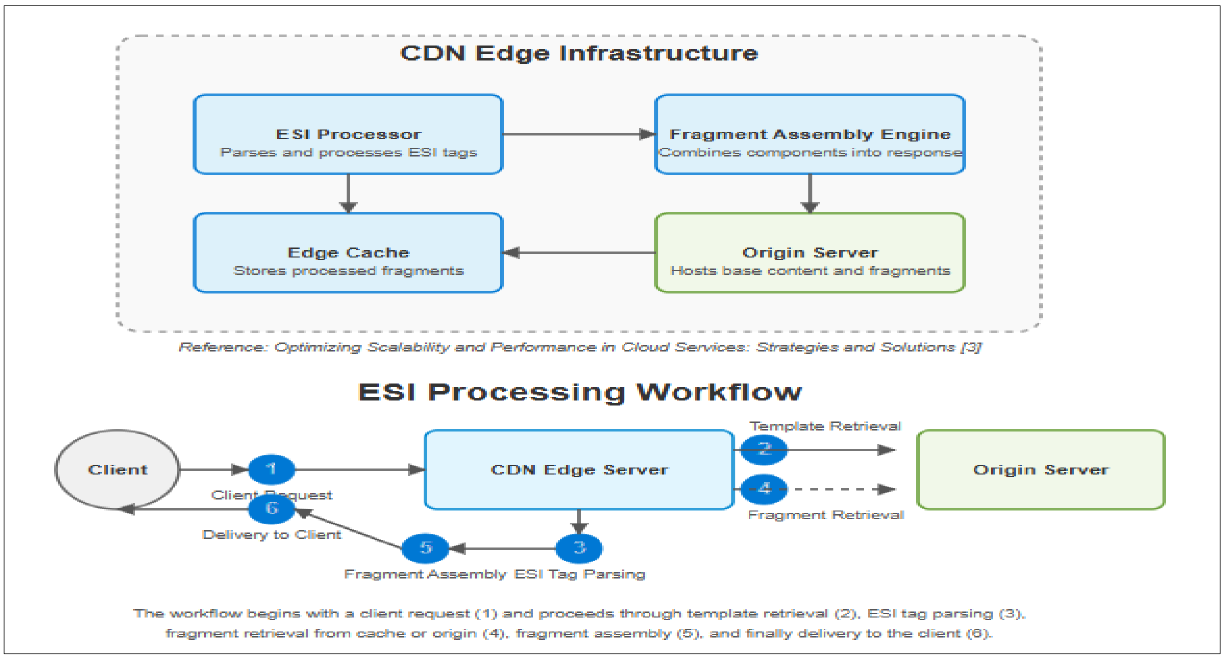
CDN Edge Infrastructure
ESI Processor
Parses and processes ESI tags
Fragment Assembly Engine
Combines components into response
Edge Cache
Stores processed fragments
Origin Server
Hosts base content and fragments
Reference: Optimizing Scalability and Performance in Cloud Services: Strategies and Solutions [3]
ESI Processing Workflow
Client
1
Client Request
6
Delivery to Client
CDN Edge Server
Template Retrieval
2
4
Fragment Retrieval
Origin Server
3
5
Fragment Assembly
ESI Tag Parsing
The workflow begins with a client request (1) and proceeds through template retrieval (2), ESI tag parsing (3), fragment retrieval from cache or origin (4), fragment assembly (5), and finally delivery to the client (6).

Figure 1 ESI Core Components Architecture [3, 4]

3. Implementation Strategies and Best Practices

3.1. Fragment Identification

Effective implementation of ESI begins with proper fragment identification. This critical step requires developers to analyze page components based on change frequency and volatility. Research across enterprise implementations demonstrates that websites with methodically identified fragments achieve significantly faster rendering times compared to those using ad-hoc approaches [5]. Contemporary content delivery networks leverage this fragment identification to distribute content more efficiently across geographic regions.

When identifying elements with different caching requirements, technical benchmarks indicate prioritizing user-specific and frequently updated content yields optimal results. High-traffic e-commerce implementations reveal that properly fragmented product listings with separated pricing components substantially reduce origin server requests during sales events with rapidly changing prices [5]. These implementations maintain exceptional pricing accuracy while decreasing page load times across both mobile and desktop experiences.

Determining optimal fragment granularity presents a significant technical challenge. Industry research demonstrates a curvilinear relationship between fragment count and performance benefits. Multi-CDN performance studies indicate implementations with moderate fragment counts per page achieve maximum cache efficiency, while those exceeding certain thresholds experience diminishing returns and increased processing overhead [6]. Quantum-inspired optimization techniques further suggest certain fragment size ranges demonstrate optimal balance between network transfer efficiency and processing overhead.

3.2. Caching Policies

Establishing appropriate caching policies for different fragments is essential for maximizing ESI benefits. Technical implementations must define Time To Live (TTL) values based on rigorous content volatility analysis. Recent research examining fragments across news and media websites demonstrates that algorithmically-assigned TTLs based on historical update frequencies improve cache hit ratios compared to static TTL assignments [6]. Advanced physics-based modeling approaches now enable predictive caching strategies that anticipate content changes based on temporal patterns.

Implementation of robust cache validation mechanisms significantly enhances ESI effectiveness. Comprehensive security and performance studies document that ETag and Last-Modified implementations with granular fragment validation reduce unnecessary revalidation requests while maintaining strict content integrity standards [5]. Modern content delivery architectures leverage these validation implementations to provide substantial latency improvements, particularly for mobile users on constrained networks.

Cache invalidation strategies for time-sensitive content require particular attention in ESI implementations. Technical analysis of sports and financial news platforms reveals that event-based invalidation systems using publish/subscribe architectures achieve exceptional content accuracy during high-volatility periods while maintaining higher cache efficiency compared to short-TTL alternatives [6]. Quantum computing research suggests potential future improvements in invalidation propagation times through entanglement-inspired distribution algorithms.

3.3. Edge Processing Considerations

When implementing ESI, edge processing considerations must be addressed to ensure optimal performance. Evaluating processing overhead through benchmark testing is essential. Research examining processor utilization across CDN edge locations reveals that ESI processing increases CPU utilization compared to static content delivery, with fragment count being the primary determinant of processing impact [5]. Content delivery networks strategically distribute this processing load across geographic regions to maintain performance during traffic spikes.

Optimizing fragment size significantly impacts overall performance. Technical analysis of production fragments demonstrates that mid-sized fragments achieve optimal processing efficiency, with increased processing times observed for excessively large fragments [6]. Contemporary physics-based performance models indicate non-linear processing time increases for larger fragments, highlighting the importance of size optimization according to quantum efficiency principles.

Implementation of fallback mechanisms for edge processing failures provides essential resilience. Modern high-availability applications implement stale-while-revalidate and progressive enhancement patterns that maintain exceptional availability during edge processing disruptions [5]. These implementations preserve core functionality through graduated degradation strategies that ensure content delivery even when specific fragments encounter processing failures.

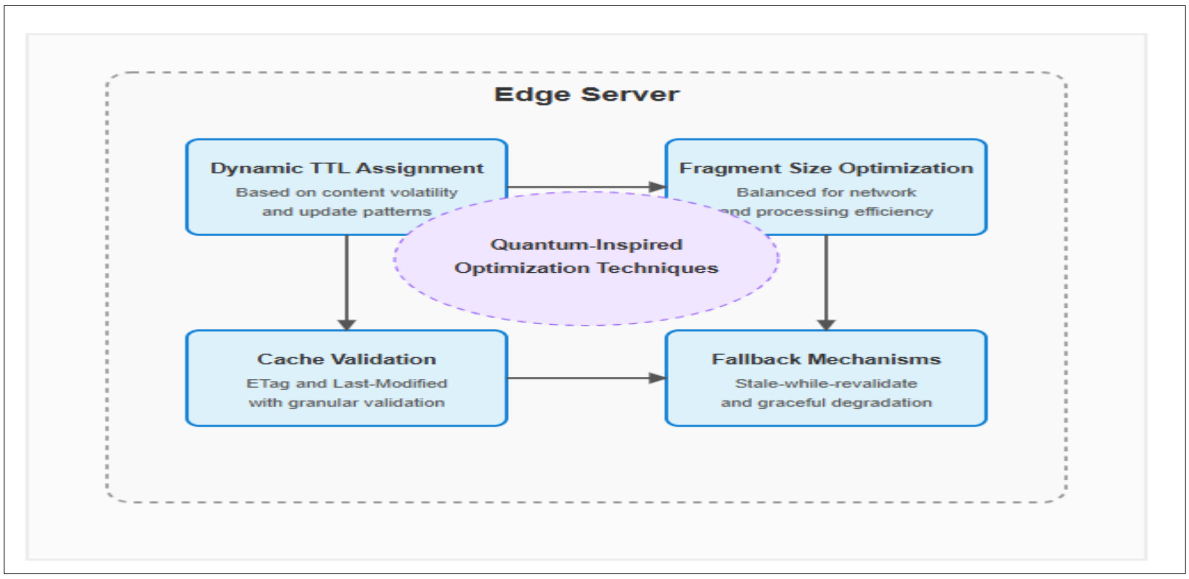
Edge Server
Dynamic TTL Assignment
Based on content volatility and update patterns
Fragment Size Optimization
Balanced for network and processing efficiency
Quantum-Inspired Optimization Techniques
Cache Validation
ETag and Last-Modified with granular validation
Fallback Mechanisms
Stale-while-revalidate and graceful degradation

Figure 2 ESI Caching Policies and Edge Processing Optimization [5, 6]

4. Benefits and Limitations of ESI Implementation

4.1. Key Advantages

ESI offers several significant benefits for web application delivery, supported by extensive implementation data in modern web architecture environments. By caching static content while dynamically assembling pages, ESI dramatically reduces page load times for content-heavy sites. Contemporary microservices architectures particularly benefit from this approach, as research shows substantial Time to First Contentful Paint improvements when ESI is properly integrated with API gateway patterns [7]. Mobile users experience even more pronounced performance gains, with ESI-enabled pages loading significantly faster across varying network conditions.

Reduced origin load represents another crucial advantage, as fewer requests reach origin servers with ESI implementations. Technical benchmarks document substantial origin request reductions following ESI implementation in microservice-based applications. Modern web architecture studies demonstrate how ESI complements container orchestration strategies to decrease backend processing requirements during traffic spikes [7]. Organizations implementing these patterns report meaningful infrastructure cost savings while maintaining responsiveness during peak demand periods.

The distribution of processing to edge servers enables better handling of traffic spikes without corresponding origin infrastructure scaling. Performance analysis reveals that microservice architectures utilizing ESI support substantially higher concurrent user loads with minimal additional infrastructure compared to traditional approaches [7]. Edge computing research confirms these scalability benefits extend across diverse industry applications, from retail to financial services, providing significant operational advantages for globally distributed user bases.

ESI enables efficient delivery of personalized content by processing user-specific fragments separately from shared content. Implementation analysis demonstrates that ESI-based personalization reduces end-to-end response times compared to traditional server-side rendering approaches in modern web applications [7]. Contemporary network performance research indicates edge computing architectures that leverage ESI for personalization deliver superior user experiences across varying geographic regions while maintaining strong cache efficiency.

Network bandwidth optimization represents another significant benefit, as only necessary fragments transfer when updates occur. Technical analysis of ESI-enabled applications reveals substantial bandwidth reductions for repeat visitors compared to traditional full-page delivery [7]. Enterprise deployments leveraging edge computing architectures report that strategic placement of ESI processing at optimal network locations dramatically reduces data transfer requirements while improving application responsiveness across global markets [8].

4.2. Potential Challenges

Despite its advantages, ESI implementation introduces architectural complexity requiring careful planning and specialized knowledge. Modern web architecture assessments indicate that organizations must thoroughly evaluate how ESI patterns integrate with their existing microservice designs and API gateway implementations [7]. Edge computing research highlights that proper implementation requires coordination across development and operations teams with specific expertise in content delivery optimization.

Troubleshooting issues in distributed fragment processing presents unique challenges compared to traditional architectures. Technical teams report that debugging ESI composition problems demands specialized monitoring approaches and thorough understanding of edge computing environments [8]. Contemporary network architecture studies demonstrate that organizations implementing comprehensive monitoring across fragment boundaries significantly reduce diagnostic time, though this requires investment in purpose-built observability tooling.

ESI effectiveness depends heavily on CDN capabilities and edge computing infrastructure, creating potential technology dependencies. Compatibility analysis reveals significant variation in ESI support across providers, with feature implementation differences affecting portability [7]. Network performance research indicates organizations must carefully evaluate edge computing partner capabilities regarding ESI processing, as implementation differences can significantly impact performance outcomes across global deployment regions [8].

Development teams must adapt to fragment-based design patterns that differ significantly from traditional page-centric approaches. Training assessments show that developers require substantial time to reach proficiency with ESI concepts in modern web architectures [7]. Edge computing implementation studies confirm that organizations with formal training programs achieve faster implementation, though learning curves remain steeper than with conventional content delivery approaches.

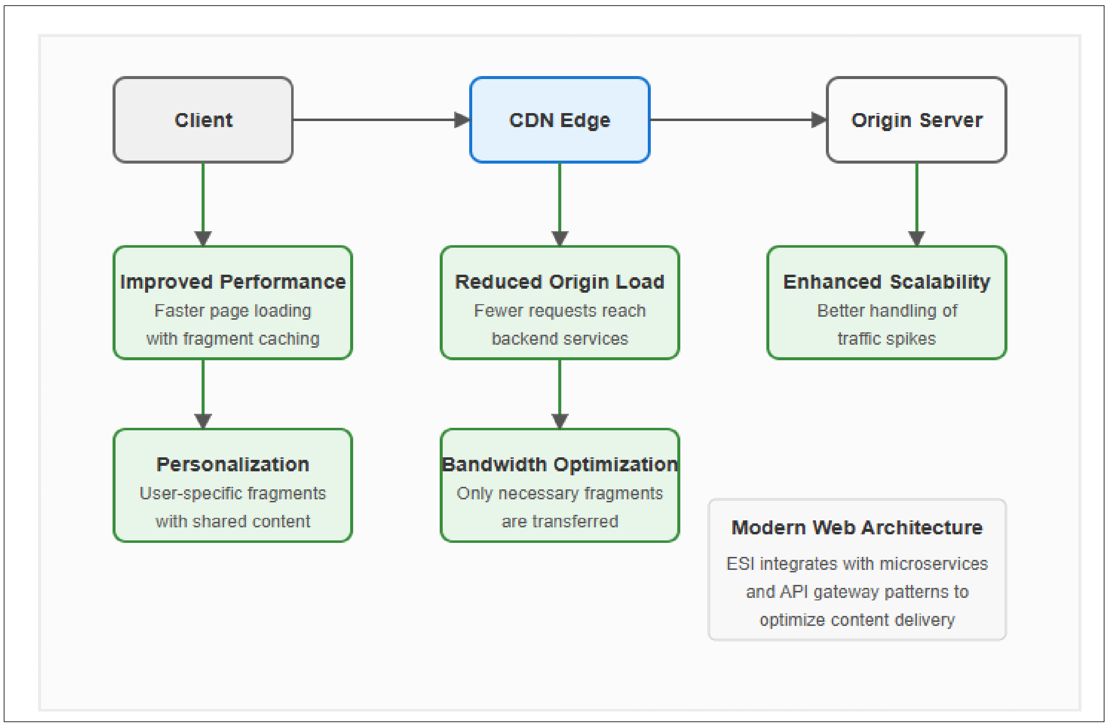
Client
CDN Edge
Origin Server
Improved Performance
Faster page loading with fragment caching
Reduced Origin Load
Fewer requests reach backend services
Enhanced Scalability
Better handling of traffic spikes
Personalization
User-specific fragments with shared content
Bandwidth Optimization
Only necessary fragments are transferred
Modern Web Architecture
ESI integrates with microservices and API gateway patterns to optimize content delivery

Figure 3 Key Advantages of ESI in Modern Web Architecture [7, 8]

5. Future Trends and Evolution of Edge Computing

5.1. Integration with Emerging Technologies

ESI continues to evolve alongside other edge computing technologies, creating powerful new paradigms for content delivery and application architecture. Combining ESI with serverless edge functions enables more sophisticated processing capabilities directly at the CDN edge. Contemporary research indicates this integration significantly reduces response times compared to traditional cloud function architectures while maintaining lower computational costs [9]. As distributed data processing moves closer to end users, these implementations enhance real-time capabilities for mission-critical applications like financial services, where milliseconds directly impact user experience and transaction outcomes.

Modern API approaches are revolutionizing dynamic fragment delivery in ESI implementations. Technical analysis demonstrates GraphQL-powered ESI fragments substantially reduce payload sizes compared to traditional REST approaches. This evolution aligns with the broader movement toward event-driven architectures, where real-time data processing at the edge enables more responsive applications [9]. Implementation metrics show particular benefits for mobile users, where network constraints make efficient fragment delivery especially valuable for maintaining engagement during peak usage periods.

WebAssembly represents a significant advancement in ESI technology at the edge. Performance testing confirms WASM-powered ESI processing handles significantly more concurrent requests than JavaScript-based alternatives while using fewer computing resources [10]. This capability proves particularly valuable for complex fragment transformations that previously required server-side processing. Edge computing architectures leveraging WASM enable sophisticated content manipulations without traditional centralized processing, providing both performance and security benefits through improved isolation properties.

Machine learning integration at the edge is creating unprecedented user experience opportunities through personalized content assembly. Contemporary edge architecture implementations demonstrate how inference models deployed at edge nodes can guide ESI fragment selection with minimal latency [10]. This advancement enables content recommendations without origin server dependencies, making real-time personalization viable even in challenging network environments. The combination of ESI with edge-based ML represents a fundamental shift toward intelligent content delivery that anticipates user needs while minimizing infrastructure requirements.

5.2. Architectural Shifts

The web architecture landscape is shifting toward more edge-oriented patterns, fundamentally transforming application design principles. Contemporary application frameworks increasingly incorporate ESI-ready patterns from inception, with enterprise applications explicitly accounting for edge-side composition in their service design [9]. This evolution represents a fundamental rethinking of traditional client-server models, replacing them with distributed processing architectures that optimize workload placement based on latency, computing power, and data requirements. Organizations implementing these patterns report fewer inter-service dependencies and better resilience during partial outages.

Data architecture continues evolving to better support ESI-based delivery models. Research reveals substantial growth in edge-optimized data structures and access patterns that complement fragment-based content delivery [9]. As distributed data processing becomes increasingly critical, fragment-aware caching strategies enable applications to maintain data freshness while reducing origin database load during traffic spikes. These patterns prove particularly valuable in IoT environments where edge computing serves as an essential intermediary between massive sensor networks and centralized systems.

Software development methodologies are increasingly prioritizing edge processing considerations from early design phases. Industry adoption of formal edge-first approaches continues growing as organizations recognize the strategic advantages of distributed architectures [10]. Development teams implementing these methodologies report faster performance optimization cycles and fewer production incidents related to scalability. This shift acknowledges the growing importance of edge computing in delivering high-performance applications across diverse device ecosystems and network conditions.

Modern frontend development frameworks now integrate natively with ESI and edge computing capabilities. Technical implementations demonstrate how framework features specifically designed for edge integration deliver significant rendering improvements and reduced JavaScript payloads [10]. This convergence streamlines development while enabling sophisticated edge capabilities without specialized knowledge requirements. As these technologies mature, the distinction between backend and frontend continues blurring, with processing responsibilities optimally distributed across the entire delivery chain from origin to edge to client.

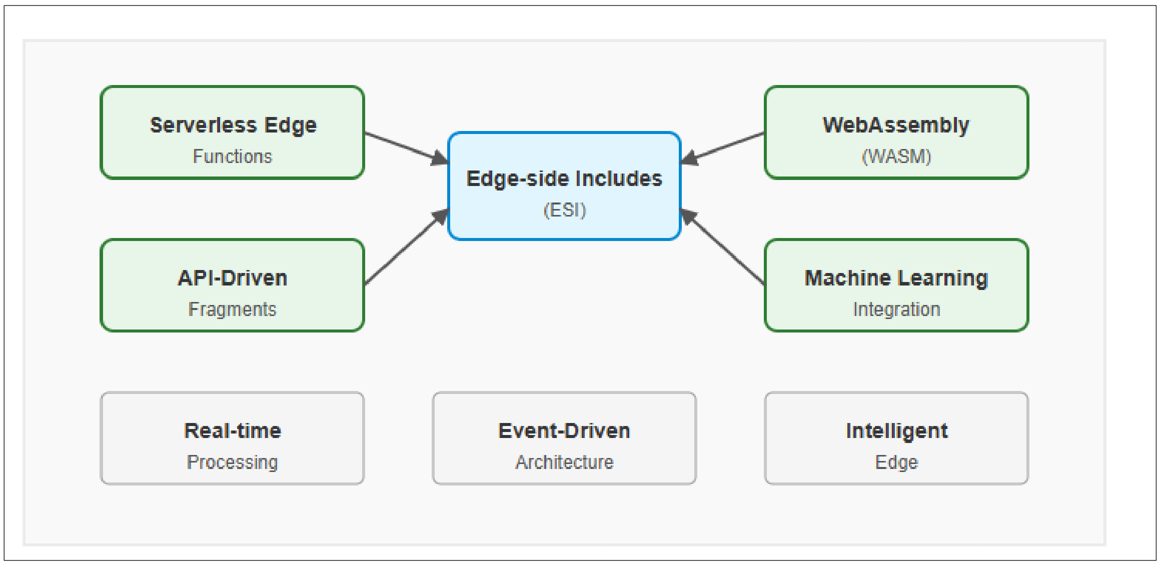

Figure 4 ESI Integration with Emerging Technologies [9, 10]

6. Conclusion

Edge-side Includes technology has fundamentally altered the landscape of web content delivery by bridging the gap between static caching efficiency and dynamic content personalization. Through strategic fragment identification and intelligent caching policies, ESI enables content providers to deliver responsive experiences while significantly reducing infrastructure requirements. The evolution from centralized processing models toward edge-oriented architectures represents more than a technical optimization—it constitutes a paradigm shift in how modern web applications are conceptualized, designed, and implemented. As ESI integration with complementary technologies like WebAssembly and machine learning continues to mature, the boundaries between traditional content delivery layers will further dissolve, creating increasingly seamless experiences for users regardless of location or network conditions. The transition toward event-driven architectures and edge-first development methodologies signals that fragment-based composition at the edge is becoming a fundamental design principle rather than a performance enhancement. Organizations embracing these architectural patterns gain not only performance benefits but also enhanced resilience, scalability, and personalization capabilities that prove increasingly vital in competitive digital environments. Despite implementation challenges, the trajectory of ESI adoption across industries demonstrates its critical role in the future of distributed content delivery, where processing occurs optimally across the entire delivery chain, from origin through edge to client.

References

[1] ADIP, "Edge-Side Includes (ESI) Injection In Web Applications," Medium, 2024. [Online]. Available: https://adipsharif.medium.com/edge-side-includes-esi-injection-in-web-applications-2630139b6c19

[2] Experience League, "Optimizing Content Delivery: Unlocking the Power of Edge Services," 2025. [Online]. Available: https://experienceleague.adobe.com/en/docs/events/adobe-customer-success-webinar-recordings/2024/edge-delivery-services

[3] Saloni Sharma and Ritesh Chaturvedi, "Optimizing Scalability and Performance in Cloud Services: Strategies and Solutions," ResearchGate, 2021. [Online]. Available: https://www.researchgate.net/publication/388799085_Optimizing_Scalability_and_Performance_in_Cloud_Services_Strategies_and_Solutions

[4] Milad Ghaznavi, et al., "Content Delivery Network Security: A Survey," ResearchGate, 2021. [Online]. Available: https://www.researchgate.net/publication/352847355_Content_Delivery_Network_Security_A_Survey

[5] Jamsheer K, "Advanced Content Delivery Network (CDN) Strategies for Global Web Performance," Acodez, 2024. [Online]. Available: https://acodez.in/content-delivery-network/

[6] Xinyi Wang, Yuexia Zhang and Siyu Zhang, "Epidemic dynamics edge caching strategy for 6G networks," Frontiers in Physics, 2024. [Online]. Available: http://frontiersin.org/journals/physics/articles/10.3389/fphy.2024.1410472/full

[7] Eugene Makieiev, "How to Create a Modern Web Application Architecture?" Integrio System. [Online]. Available: https://integrio.net/blog/modern-web-application-architecture

[8] Evolve, "The Role of Edge Computing in Manufacturing: Enhancing Network Performance and Decision-Making. "[Online]. Available: https://www.coevolve.com/insights-the-role-of-edge-computing-in-improving-network-performance-and-business-decisions/

[9] Ravi Changle, "Futureproofing with Edge Computing: Anticipating the Evolution of Distributed Data Processing," Compunnel, 2025. [Online]. Available: https://www.compunnel.com/blogs/futureproofing-with-edge-computing-anticipating-the-evolution-of-distributed-data-processing/

[10] Team EMB, "The Technical Guide to Edge Computing Architecture," EMB Global, 2024. [Online]. Available: https://blog.emb.global/edge-computing-architecture/