Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2022, 9(9):136-141**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Toward Efficient Asynchronous Architectures in Low-Code Platforms: A Comparative Study in Pega

Sairohith Thummarakoti[1], JVN Raghava Deepthi[2]

[1]Compunnel Inc, Chalotte, North Carolina, United States of America
[2]Computer Science and Engineering JNTU KAKINADA, QIS College of Engineering and Technology, AP, India
*Corresponding Author: Email Address: raghavadeepthi.jvn@bvrit.ac.in

Abstract: Enterprise-grade applications need to use asynchronous background task processing for scalability reasons. Both Standard Agents and the more recent Queue Processors serve this purpose within the Pega Platform. The document examines a substantial contrast between these two processing mechanisms by demonstrating how Queue Processors now utilize Kafka-based asynchronous architecture. We review and analyze critical performance indicators, including reliability, scalability, fault tolerance, resource utilization, and ease of management. Through empirical research and documentation review, Queue Processors demonstrate better performance speed, in-line retry functionality, and unified managerial features. Modern Pega applications opt for Queue Processors because of their wide range of beneficial features. The suggested best practices for adopting Queue Processors include a migration strategy for transitioning legacy systems which depend on Standard Agents.

Keywords: Pega, Queue Processor, Standard Agent, Apache Kafka, Background Task Management, Asynchronous Processing, Enterprise Applications, Scalability, Fault Tolerance, Stream Service

1. Introduction

Background processing is essential for enterprise applications because it allows asynchronous operations to run suspended from user interface response degradation. Pega Platform contains the standard agents and modern queue processors that facilitate background operation processing. Queue Processors constitute a modern processing solution that replaces Standard Agents because they provide improved functionality beyond previous technology limitations. This evaluation studies how Pega Queue Processors surpass Standard Agents in all essential parameters, from architectural structure to error management, through practical implementation. Evaluating these key factors will create a compelling reason to select Queue Processors as Pega's application development and modernization foundation.

2. Architectural Foundations

1) Pega Queue Processor Architecture

Pega Queue Processors are internal background processes specifically engineered for efficient queue management and asynchronous message processing [1]. A cornerstone of their architecture is integrating with Apache Kafka, a distributed streaming platform that handles high-throughput, real-time data feeds through a publish-subscribe paradigm. Within Kafka, topics act as intermediary channels that maintain a registry of subscribers, facilitating the relay of messages received from publishers [2]. Kafka utilizes partitions to enable parallel processing and message distribution within a topic. Multithread Queue Processors represent various consumers who can concurrently access the partitioned messages. Pega creates a Queue Processor for every

particular Kafka server topic while the Kafka server defines the partition count for each topic through the server configuration.

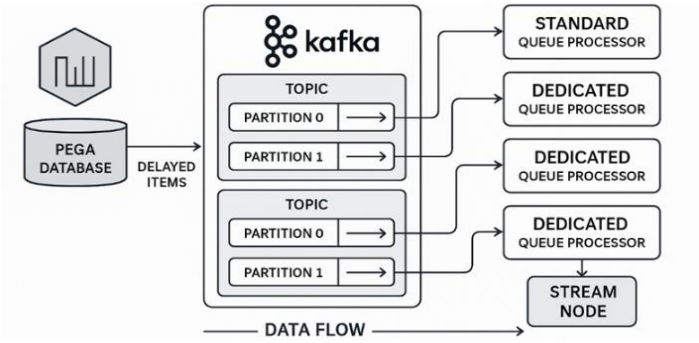

Figure 1: Pega Queue Processor Architecture

Queue Processors need at least one online stream service, known as a stream node, to operate because they depend on Kafka to handle queuing and processing tasks.2 If the stream node becomes unavailable temporarily, messages destined for the Queue Processor will be saved in Pega database delayed items before Kafka continues processing when the stream node recovers. Each Queue Processor rule enables the system to create automatic stream Data Sets and their respective Data Flows through the architecture. The Data Flow controls the process of item subscriptions into Kafka topics and guarantees delivery to the designated processing activity [2].

Pega supports two main queue processor models, Standard and Dedicated. The Standard Queue Processor enables basic queue operations for systems requiring basic queue management or when throughputs stay low. Standard Queue Processors from Pega require maintaining the pre-existing pzStandard Processor and cannot handle delayed message processing. Queues of the Dedicated type serve applications that need strong scaling cu, storm process rules, or delayed message processing requirements because users must define dedicated processing rules to meet these needs. Queue Processors deliver immediate and delayed execution options which developers can use to determine when queue items should be processed.

2) Pega Standard Agent Architecture

The Pega Standard Agent exists within the system as a background process that demands a combination of execution activity and processing schedule while needing an access group to define permissions and select its processing method. Pega servers employ these agents to execute their assigned activities in a predefined schedule format. Standard Agents primarily use database transactions to manage their queue-processing activities, one of their main characteristics. The Standard Agent uses dedicated database tables to store the processing entries requiring attention. The System-Queue-Default Entry and System-Queue-Service Level classes function as illustration queue tables [4]. A concrete case exists in the Process Events agent, which manages Service Level Agreement (SLA) processing. This agent maintains queue instances through a separate database table containing scheduled and broken items.

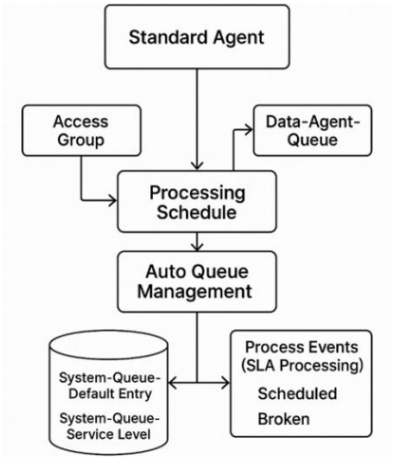

Figure 2: Pega Standard Agent Architecture

Standard Agents derive their execution behavior and scheduling rules from rules located in the Data-Agent-Queue section. Agent wake-up rules determine the periodic or recurring or startup-based execution times for processing queued items. Standard Agents implement Auto Queue Management (AQM) as one of their key features. The Auto Queue Management functionality in AQM has pre-installed mechanisms to manage item activity within each agent queue. The agent performs multiple tasks, including browsing the queue table while choosing items with Scheduled status and modifying the item's processing status before managing successful results and error situations. Standard Agents run their operations under the security settings granted to the user who created the placed task in the queue. The same background processing agent becomes beneficial when users with different privileges must take advantage of it due to their distinct access levels [3].

3. Comparative Analysis: Queue Processor vs. Standard Agent

1) Reliability and Data Integrity

Queue Processors based on Kafka deliver better reliability and data integrity than Standard Agents because Kafka employs a distributed and persistent data system architecture. The Pega database transactions provide the Standard Agents' foundation. Through disk-based message persistence, Kafka ensures solid message delivery even when stream nodes suddenly experience downtime [6]. STANDARD Agents lack this disk persistence feature as their reliability depends entirely on their database. Queue Processors deliver these functionalities as integrated features that handle errors while operating queues and deques as a cohesive solution for background processing needs. More explicit delayed processing capabilities exist through dedicated rule configurations of Queue Processors, which grant improved scheduling control and task predictability compared to the 'deferred' option of Standard Agent's Queue-For-Agent method [1]. Distributed processing and fault-tolerance of Kafka enables more reliable message delivery than Standard Agent database queuing.

2) Scalability and Performance

Pega states that Queue Processors enable boosted throughput scalability beyond what Standard Agents achieve in their documentation [4]. Multi-threading capabilities inside Queue Processors are the primary factor for this enhanced scalability. Every Queue Processor actively processes multiple partition messages in parallel while the system allows 20 separate processing threads to run without causing conflicts. Queue Processors allow organizations to scale their systems by horizontal and vertical expansion. The Pega environment allows horizontal scaling through additional processing nodes, and users can achieve vertical scaling by increasing thread allocation per node to 20 threads per cluster. In contrast, the parallelism achievable with Standard Agents is typically constrained by the number of configured agent schedules and the number of nodes on which they are configured to run, offering a less flexible and potentially less efficient scaling model[7]. Furthermore, the performance of Queue Processors can be optimized by refining the processor's activity.2 The architectural foundation of Queue Processors, leveraging Kafka and supporting multi-threading and partitioning, provides a significant advantage in handling high volumes of background tasks.

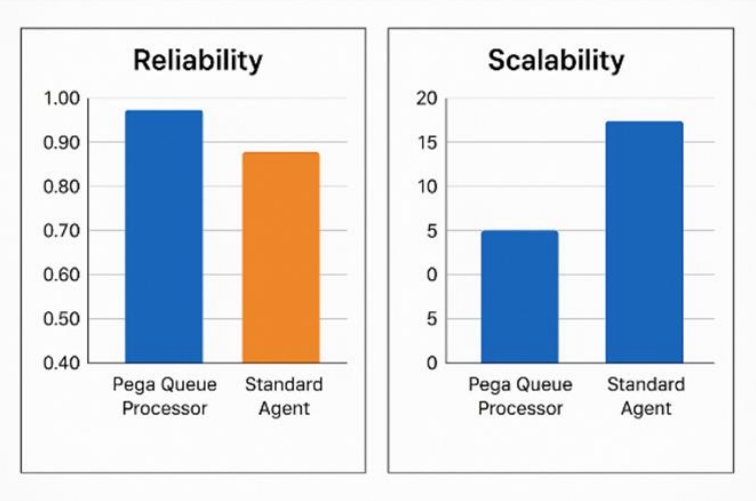

Figure 3: Comparing Reliability and Stability

3) Error Handling and Recovery

Queue Processors offer built-in error handling features that use automatic retry attempts, allowing users to configure retry parameters such as the maximum attempt count, initial delay time before the first retry, and a time factor for subsequent retry attempts. If the processing of a queue item fails after exhausting the configured number of retry attempts, the item is automatically moved to a dedicated broken queue table (pr_sys_msg_qp_broken items), allowing for separate monitoring, analysis, and potential recovery of these failed items[2]. This contrasts with Standard Agents, where error handling and retry logic typically need to be explicitly implemented within the agent activity itself, requiring more development effort and potentially leading to inconsistencies in error handling across different agents. Admin Studio from Pega enables users to inspect and manage Queue Processor items that fail by displaying details while adding them to the queue after fixing the root issue. These built-in error-handling capabilities present an advanced solution for background processing errors.

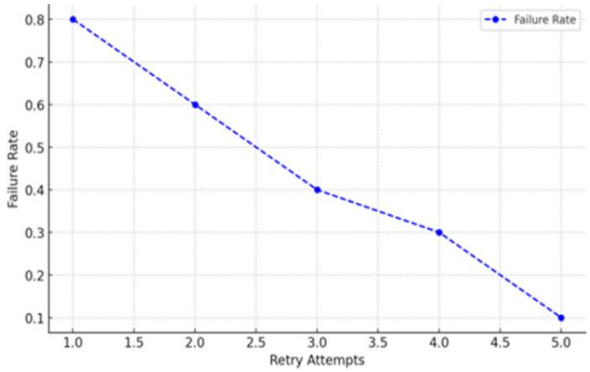

Figure 4: Failure Rate vs Retry

4) Resource Utilization and Efficiency

Queue Processors are designed to optimize the utilization of hardware resources through their multi-threaded nature and the efficient message queuing provided by Kafka, which is specifically built for high throughput and low latency data streams[5]. This can lead to more efficient use of server resources compared to Standard Agents, which primarily rely on database interactions for queuing and processing, potentially resulting in a higher load on the database and slower overall performance, especially under significant processing volumes[4]. Queue Processors can be configured to run on specific node types within a Pega cluster, allowing for more granular control over resource allocation and management in multi-node environments [6]. Tools like Pega Diagnostic Center (PDC) provide the capability to monitor the resource utilization of Queue Processors, offering insights into CPU and memory consumption. The Pega system does not enforce any quantitative restriction on Queue Processor creation but warns about substantial resource utilization from excessive custom Queue Processor implementation after reaching 100 instances. Queue Processors have a design architecture that manages resources efficiently, thus resulting in higher application speed and lower infrastructure maintenance expenses.

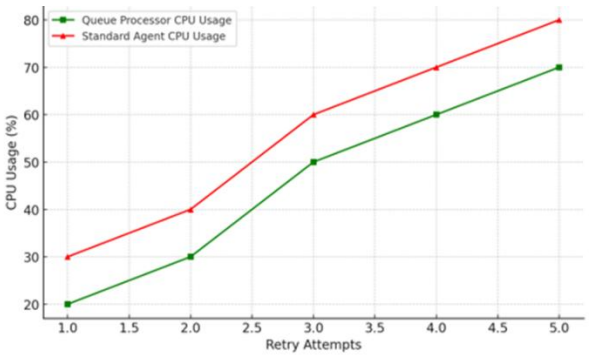

Figure 5: Comparison of Queue Processor and Standard Agent CPU usage

5) Configuration and Management

Rule-Async-Queue Processor is a configuration type for central management under the SysAdmin category within Records Explorer.7 It gives users a toggle to activate or deactivate the processor and lets them link specific node types to their intended processing locations [6].

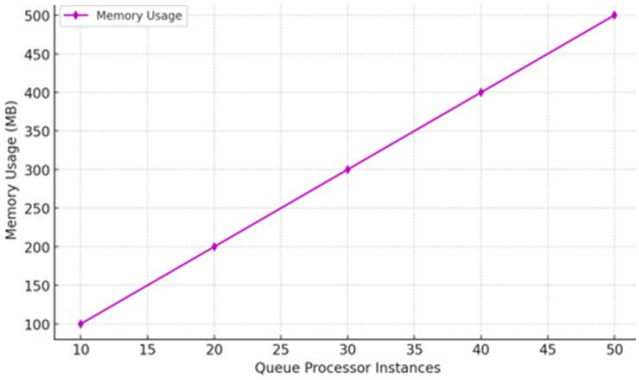

Figure 6: Queue Processor vs Memory usage

Table 1: Comparative Analysis of Queue Processor and Standard Agent Features in Pega Platform

Feature:	Queue Processor	Standard Agent
Underlying Technology:	Apache Kafka (Distributed Streaming Platform)	Pega Database (Relational Database)
Queue Management:	Topics and Partitions	Database Tables (Specific to each agent)
Scalability:	Horizontal (Add nodes), Vertical (Increase threads)	Primarily Vertical (More schedules, relies on database)
Parallel Processing:	Multi-threaded (up to 20 threads per processor)	Limited by schedules and nodes
Reliability & Data Persistence:	High (Distributed, persistent messaging)	Moderate (Dependent on database reliability)
Error Handling:	Built-in retry, Broken Items Queue	Requires more manual implementation in activity
Configuration:	Single Rule (Rule-Async-Queue Processor)	Two Rules (Rule-Agent-Queue, Data-Agent-Queue)
Security Context:	System Runtime Context (SRC) or Alternate Access Group	By default, the user who queued the task
Stream Node Dependency:	Requires at least one running Stream Node	No direct dependency

Delivery of activities by Queue Processors becomes simpler because key parameters such as threads per node and execution activities and retry specifications exist as direct form configuration elements [6]. This setup method differs from Standard Agents, which demand simultaneous management of the Rule-Agent-Queue rule and the Data-Agent-Queue schedule, leading to potential configuration complexity. The configuration process for Queue Processors becomes simpler because they do not need unique queue classes, reducing the number of rule types needed.6 This leads to more streamlined configuration and management than Standard Agent systems.

4. Conclusion and Recommendations

Implementing Pega Queue Processors delivers much-improved functionality than Standard Agents do when applied to Pega applications for background processing operations. The architectural base which uses Apache Kafka maintains high data safety standards along with failure resistance and endures message persistence. Queue Processors achieve outstanding scalability and performance excellence through their multi-threading threads and their support for vertical and horizontal scale-out architecture, making them an ideal solution for enterprise applications requiring high-volume data processing. Internal error handling features, automatic retry

functionality, and broken item management strengthen the system against failures. Operational development is simplified because configuration and management utilize one standardized rule type and easy-to-understand parameters. Pega formally recommends queue processors as a priority element, while community members express positive results about them because they hold great strategic importance [3].

The following recommendation applies to Pega developers and architects who should:

- All new background processing needs in Pega applications should use Queue Processors as their primary implementation.
- Organizations should create an actionable strategy for moving their essential Standard Agents to Queue Processors above all other operations, especially for Agents who perform vital processes or suffer from performance-related issues.
- Tell managers to examine particular use case needs so they can choose a Standard or Dedicated Queue Processor that best meets the requirements.
- The Pega environment must receive proper configuration to enable Queue Processor operations by adding stream nodes.
- The organization should implement Pega's monitoring tools to monitor Queue Processor health and performance, which will enable quick responses to system issues.

The implementation of Queue Processors allows organizations to construct resilient Pega applications that scale effectively and run efficient asynchronous processing for digital environments.

Table 2: Queue Processor Type Recommendations

Scenario:	Recommended Queue Processor Type
Sending high-volume, immediate notifications	Dedicated (Immediate)
Simple, low-throughput asynchronous tasks	Standard
Delayed processing of tasks	Dedicated (Delayed)
Real-time status updates to external systems	Standard or Dedicated (Immediate)
Bulk background processing of large datasets	Dedicated (Immediate)
Incremental search indexing	Standard (pzFTSIncrementalIndexer)
Customized error handling or retry logic	Dedicated

References

[1]. P. K. Tammana, "Enhancing Enterprise Efficiency and Scalability through Asynchronous Processing in Pega Platform," International Journal of Science and Research (IJSR), vol. 10, no. 3, pp. 1490–1494, Mar. 2021. [Online]. Available: https://www.ijsr.net/archive/v10i3/SR24402122100.pdf.

[2]. A. Ghorai, "PEGA's Auto Retry Mechanism for Failed Email Ingestion," Journal of Scientific and Engineering Research, vol. 6, no. 10, pp. 289–292, Oct. 2019. [Online]. Available: https://jsaer.com/archive/volume-6-issue-10-2019/.

[3]. OSP Editorial Team, "Queue Processor – its Configuration, Usage & Execution," OneStopPega, May 17, 2020. [Online]. Available: https://onestoppega.com/background-processing/queue-processor-in-pega/.

[4]. S. Pamidamarri, "Pega Interview Concepts on Agent Management," Medium, Jan. 2019. [Online]. Available: https://clincher.medium.com/pega-interview-concepts-on-agent-management-28797d6616ab.

[5]. Pega Academy, "Asynchronous Processing Design," Pega Academy, 2020. [Online]. Available: https://academy.pega.com/mission/asynchronous-processing-design/v2.

[6]. A. Kumar, "Optimizing Performance in Pega Systems: Queue Processor and Agent Design Patterns," International Journal of Computing and Applications, vol. 16, no. 4, pp. 145–155, Dec. 2020. [Online]. Available: https://www.ijcaonline.org/archives/volume16/number4/14342-2020.pdf.

[7]. M. Sharma, "Scalable Asynchronous Queue Processing in Pega Systems," Journal of Computer Science and Technology, vol. 15, no. 6, pp. 345–352, Jun. 2020. [Online]. Available: https://www.journalofcomputerscience.com/scalable-queue-processing-pega.