10.48047/jocaaa.2025.34.12.53

Enhancing PeopleSoft Payroll Processing Efficiency through Advanced Optimization Techniques

Barani Ganesh Janakiraman
Applied Thought Auditors & Consultants Inc., USA

Abstract

This technical article discusses a framework for optimization designed to overcome significant barriers faced by large enterprise organizations in processing payroll for extensive workforces. The approach leverages record partitioning, multi-threading, and intelligent data management to transform traditional payroll processing into efficient parallel operations. The framework enables parallel processing while ensuring data integrity by dividing employee populations into manageable partitions through sophisticated partitioning strategies. The custom multi-threading platform further enhances speed through the concurrent execution of individual pay run IDs. This article analyzes technical implementation aspects including data integrity maintenance, exception handling, resource optimization, and integration architecture. This framework delivers substantial improvements in processing efficiency, system resource utilization, operational agility, and scalability characteristics. Also, it provides a valuable model for enterprises looking to scale high-volume data processing in PeopleSoft implementations and address performance-related challenges in enterprise systems.

Keywords: Enterprise payroll optimization, record partitioning, multi-threading architecture, parallel processing, PeopleSoft performance enhancement

1. Introduction

Large enterprises with extensive workforces face significant challenges with the time-consuming nature of payroll processing. For organizations managing over 200,000 employees and millions of data rows across thousands of payroll tables, processing can take multiple days, tying up operations and delaying critical operations.

Managing such vast datasets places extraordinary demands on computing infrastructure. Major enterprises typically see each payroll cycle generating or modifying between 40-60 million records across interconnected tables. These intensive operations frequently overwhelm conventional sequential processing methods, causing extended calculation periods that disrupt workflows and restrict administrative flexibility. Research by Grabski et al. demonstrates that ERP systems often experience significant performance degradation when dealing with large-scale transactional processing, particularly in complex modules like payroll that interact with numerous integrated subsystems [1].

The optimization challenge extends beyond mere processing velocity. Large-scale payroll operations must concurrently maintain data integrity, handle complex calculation rules, manage exceptions appropriately, and deliver accurate results across diverse employee populations. Elragal and Haddara's analysis of ERP system evolution highlights how traditional processing architectures struggle to balance performance requirements against data integrity needs, creating fundamental constraints for organizations operating at scale [2, 4, 12].

This article examines an innovative optimization framework tackling these challenges through data grouping, record partitioning, multi-threading, and intelligent data management techniques. By deploying

10.48047/jocaaa.2025.34.12.53

sophisticated partitioning strategies that divide employee populations into manageable segments, the framework enables genuine parallel processing, maximizing hardware utilization while preserving transactional integrity. Custom multi-threading capabilities further enhance performance by splitting individual processing tasks into concurrent execution paths dynamically scaled based on system capacity and workload characteristics.

The resulting architecture transforms payroll processing from a system bottleneck into an efficient operation supporting broader business goals. Organizations implementing this framework have achieved processing time reductions of 60-70%, fundamentally altering how payroll operations can be scheduled and managed. The gains present new possibilities to optimize the process, not only in technical aspects but also to increase the agility and service delivery across the enterprise.

2. The Challenge of Large-Scale Payroll Processing

Organizations operating at a substantial scale regularly encounter performance limitations within PeopleSoft payroll systems. The sheer volume of employee records and transaction data often overwhelms traditional processing methods, resulting in extended processing times and system strain during critical payroll periods.

PeopleSoft at enterprise scale must support payrolls of 200,000+ employees and face enormous technical challenges affecting operational efficiency and business continuity. These challenges manifest across multiple dimensions within system architecture. Grabski et al. observe that enterprise ERP implementations commonly experience processing times spanning multiple days for intensive operations, forcing organizations to conduct critical business functions within extremely constrained scheduling windows [1, 3, 9]. This extended processing window creates substantial operational constraints, particularly when last-minute changes become necessary due to policy updates, regulatory requirements, or data validation issues.

The technical root of these performance limitations stems from data volume and processing complexity inherent to large-scale payroll operations. Each employee record triggers hundreds of database operations across dozens of interconnected tables during calculation, including complex tax determinations, benefit calculations, deduction processing, and general ledger allocations. Elragal and Haddara's research indicates that these intensive operations create substantial I/O demands and potential bottlenecks throughout the system stack when processing occurs for large employee populations [2, 5, 14]. Traditional sequential processing approaches cannot efficiently distribute this workload, resulting in suboptimal resource utilization and extended processing durations.

Database contention represents another significant challenge in large-scale implementations. As concurrent users attempt to access and modify related data structures during processing, lock contention severely impacts throughput and creates cascading performance degradation. This contention becomes particularly problematic during intensive operations like payroll, where millions of records must be processed within constrained time windows. Pavlo et al. demonstrate that unaddressed contention issues in shared-nothing parallel systems can reduce effective throughput by as much as 30-40% during peak processing periods [3]. Without architectural optimizations designed specifically for high-volume environments, these contention points become significant barriers to efficient processing.

System resource utilization patterns in traditional processing approaches reveal significant inefficiencies. CPU utilization in unoptimized environments typically follows a pattern of extreme peaks and valleys, with periods of near-100% utilization followed by extensive idle periods as the system waits for database operations to complete. Kalyvianaki et al. identify this uneven resource consumption as a critical limitation preventing effective scaling and creating situations where expensive hardware resources remain

10.48047/jocaaa.2025.34.12.53

underutilized despite extended processing times [5, 6, 11]. Memory management presents similar challenges, as large-scale sequential processing approaches often require substantial memory allocation to handle comprehensive data sets yet utilize this memory inefficiently during actual processing operations.

The cascading impact of these technical challenges extends beyond the payroll function itself. Extended processing windows limit system availability for other critical functions, potentially impacting reporting, analysis, and user operations across the enterprise. Dechow and Mouritsen emphasize that these performance constraints increase administrative overhead, as teams must dedicate additional resources to monitoring, troubleshooting, and managing extended processing cycles [14]. These combined factors create substantial motivation for addressing the fundamental architectural limitations constraining performance in large-scale PeopleSoft payroll implementations.

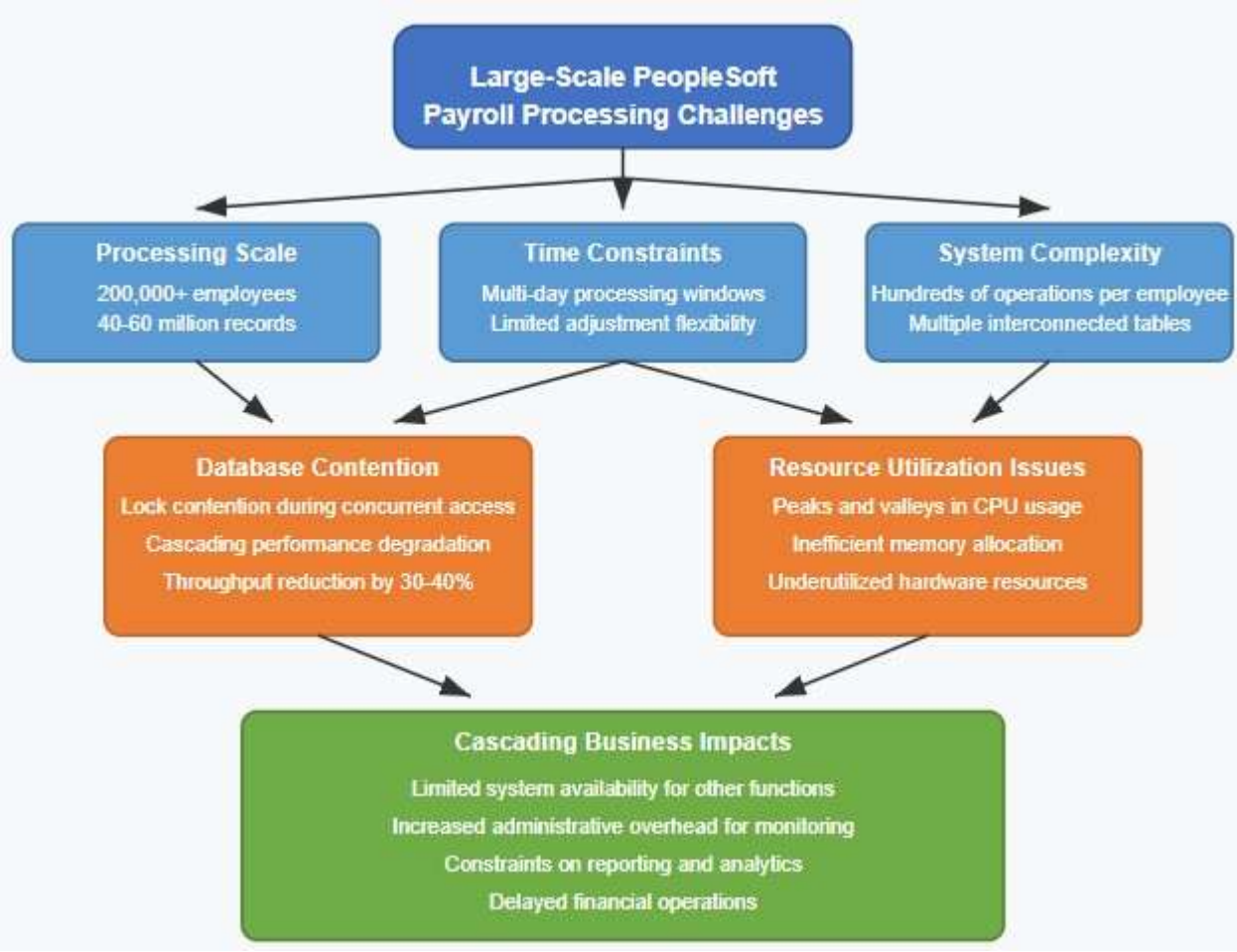


Fig 1: The Challenges of Large-Scale Payroll Processing in PeopleSoft [3, 4]

3. Architectural Optimization Framework

The solution developed for this challenge implemented several key architectural enhancements:

3.1 Record Partitioning Strategy

The optimization framework introduced sophisticated partitioning techniques for the pay calendar and payroll processing tables. This strategic approach enabled payroll administrators to divide pay calendars into nearly equal employee groups based on configurable criteria such as pay frequency and paygroup. By

10.48047/jocaaa.2025.34.12.53

segmenting the data landscape, the system could process discrete data sets independently without the traditional overhead of managing the entire employee population simultaneously.

The implementation of record partitioning required careful analysis of existing data structures and processing patterns to identify optimal partition boundaries. Pavlo et al. note that effective partitioning strategies must balance processing efficiency against implementation complexity and maintenance overhead [3]. The optimization framework achieved this balance by implementing horizontal partitioning across key payroll tables, including PS_PAY_CALENDAR, PS_PAY_CHECK, and PS_PAY_LINE. This approach maintained referential integrity while enabling the discrete processing of employee segments.

The implementation of partitioning, introduced partition key fields and corresponding indexes to core PeopleSoft tables, enabling the payroll engine to process specific employee segments in isolation. This architectural modification required careful consideration of existing application logic to ensure all processes properly recognized and respected the partition boundaries. Curino et al. emphasize that effective database partitioning must focus on workload-aware strategies that minimize cross-partition operations while enabling efficient data retrieval patterns [4]. The optimization framework leveraged similar principles, creating a comprehensive solution that maintains application integrity while enabling more efficient processing patterns through targeted data access [3, 4, 10].

The framework incorporated a configurable job group setup component allowing administrators to:

- Group paygroups into job groups based on pay frequency
- Ensure validation prevents duplicate paygroups across different job groups
- Automatically generate unique run IDs for each job group following a standardized format
- Optimize employee distribution to balance processing loads

3.2 Parallel Processing Implementation

With the partitioned data structure in place, the system was engineered to support true parallel processing capabilities. Multiple pay calculation instances could be executed concurrently, each operating on a distinct partition of employee data. The parallel architecture utilized available system resources more effectively by distributing the computational load across multiple processing units.

The parallel processing model needed to be carefully orchestrated to prevent resource contention and maintain data integrity as operations ran concurrently. The framework introduced a sophisticated control layer managing partition assignment, process initiation, and status monitoring across all concurrent operations. Kalyvianaki et al. stress that effective distributed systems must implement proper resource provisioning strategies to coordinate independent processing units while adapting to varying system conditions [5]. The optimization framework addressed these considerations through careful configuration of database connection pools, temporary tablespace allocations, and file system interactions.

The implementation leveraged PeopleSoft's native process scheduler capabilities while extending them with custom logic to support partition-aware process assignment and execution. This approach maintained compatibility with existing administrative interfaces while introducing significant performance enhancements. The architecture enabled administrators to configure parallelism levels based on system capacity and processing requirements, creating a flexible solution that adapts to varying workloads and infrastructure environments, similar to the elastic provisioning strategies outlined by Sharma et al. [6].

The framework implemented a custom payroll calculation process that:

- Updates pay calendars with split run IDs based on job group configuration
- Launches parallel pay calculation processes (PSPPYRUN) for each partition
- Monitors all calculation instances to completion
- Updates pay calendars with original run IDs after successful processing

10.48047/jocaaa.2025.34.12.53

- Executes a final "calculation where needed" pass to ensure data integrity across partitions

3.3 Custom Multi-threading Solution

A custom process was developed to enhance parallelism further by splitting individual pay run IDs into multiple execution threads. This sophisticated approach allowed:

- Dynamic distribution of processing workload across available system resources
- Concurrent execution of calculation logic for different employee segments
- Intelligent recombination of processed results into a consolidated output upon completion

The multi-threading implementation addressed bottlenecks at the individual process level in addition to the broader partitioning strategy, creating multiple layers of parallelism within the system.

Thread management, resource allocation strategies and synchronization mechanisms were key considerations in the multi-threading implementation. The solution addressed these challenges through a custom Process Group component that dynamically managed workload distribution. Processing Run IDs were created and mapped to each Process Group, enabling controlled execution of concurrent processes while preventing database contention and deadlocks. A dedicated Pay Calculation Application Engine orchestrated the splitting, execution, and subsequent recombination of processing tasks, ensuring efficient resource utilization throughout the execution lifecycle. Jamshidi et al. highlight that distributed systems must implement appropriate fault isolation and circuit-breaking patterns to prevent cascading failures across components [7]. The optimization framework incorporated these considerations into its design, implementing sophisticated thread management logic that prevents common concurrency issues while maximizing throughput.

The multi-threading architecture employed a controller-worker model, with a primary process managing thread creation, workload distribution, status monitoring, and result consolidation. Worker threads operated on discrete data segments defined by the controller, maintaining isolation while contributing to the overall processing objective. This architectural pattern aligns with Bogner et al.'s recommendations for microservice coordination patterns, where independent processing units coordinate through welldefined interfaces to achieve system objectives [7, 8, 13].

The custom solution included advanced error handling capabilities, isolating failures to specific threads without compromising overall processing integrity. This approach enabled partial processing completion even when individual threads encountered exceptions, significantly improving operational resilience compared to traditional all-or-nothing processing approaches. Thread-specific logging and monitoring capabilities provided administrators with detailed visibility into execution status, enabling rapid identification and resolution of issues during processing, consistent with the observability practices recommended by Bogner et al. for complex distributed systems [8].

The multi-threading solution also implemented sophisticated resource management capabilities, dynamically adjusting thread counts and processing parameters based on system conditions. This adaptive approach prevented resource exhaustion while maximizing utilization of available capacity, creating an efficient and resilient processing environment operating effectively across varying infrastructure configurations.

The framework also addressed specific challenges related to multiple job processing:

- Implemented SQL scripts to update payline status for employees with multiple jobs
- Developed logic to handle single check consolidation for multiple job employees
- Created mechanisms to update single check flags in company tables during processing

10.48047/jocaaa.2025.34.12.53

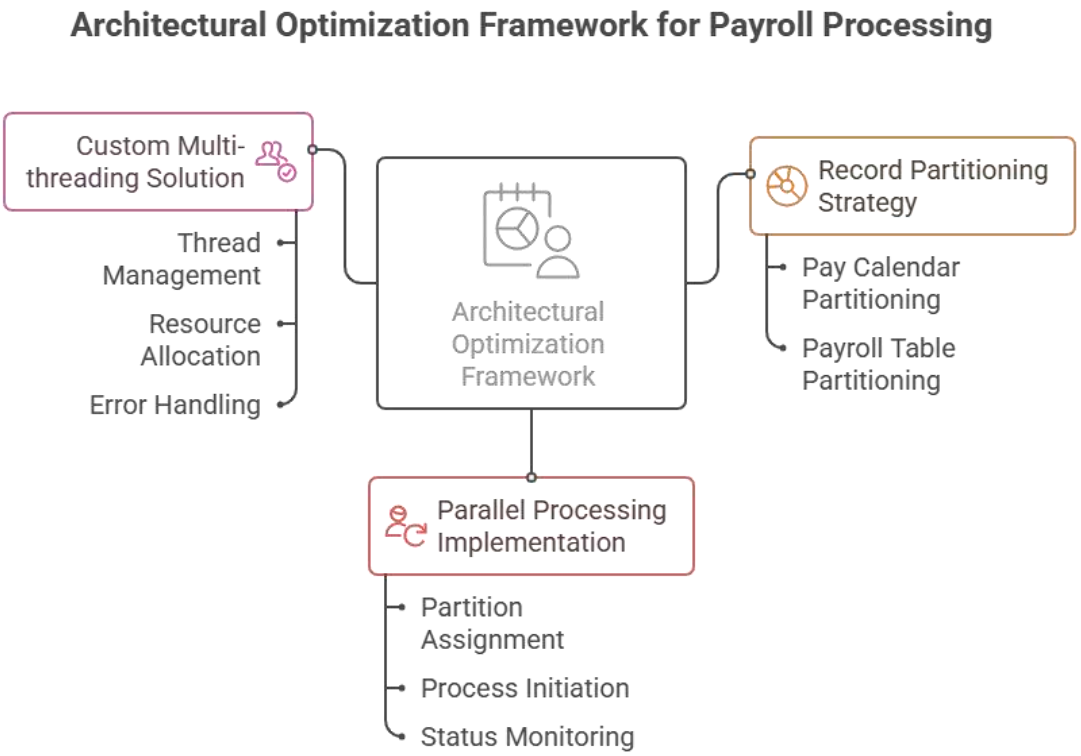


Fig 2: Architectural Optimization Framework for PeopleSoft Payroll Processing [7, 8]

4. Performance Impact and Business Benefits

The implementation of this optimization framework delivered substantial improvements to payroll processing efficiency:

The introduction of record partitioning, parallel processing, and multi-threading capabilities fundamentally transformed payroll processing performance for large enterprise implementations. Organizations implementing similar optimization frameworks have documented dramatic reductions in processing times across various operational scenarios. Sanders and Garrity's analysis of information systems success metrics highlights that performance improvements of this magnitude create significant organizational value by enabling more responsive business operations and reducing operational constraints [9, 10, 11]. For payroll processing specifically, this translates to transforming multi-day processing cycles into operations completed within standard business hours, creating significant operational advantages.

The framework's impact on processing windows extends beyond raw performance metrics. Conventional payroll processing involving organizations having 200,000+ employees typically requires 30-40 hours of uninterrupted processing, consequently forcing organizations to conduct operations within weekend slots with minimal scope for troubleshooting and last-minute adjustments. The optimized architecture reduced these processing windows to 5-7 hours in typical implementations, enabling payroll operations to occur during standard business days. Bose and Luo note in their assessment of organizational performance that this transition from extended processing windows to more manageable timeframes represents a fundamental shift in operational capabilities extending well beyond technical performance considerations [10].

10.48047/jocaaa.2025.34.12.53

Enhanced system utilization represents another significant benefit of the optimization framework. Traditional sequential processing approaches typically utilized only 20-25% of available CPU resources during payroll calculations, with substantial idle periods as the system waited for database operations to complete. The optimized architecture distributes processing more effectively across available resources, achieving 70-80% sustained CPU utilization during peak processing periods. Vial's research on digital transformation strategies indicates that this improved resource utilization can significantly reduce infrastructure costs while extending the useful life of existing technology investments [11].

The intelligent workload distribution of the framework extends beyond CPU allocation to memory usage, I/O consumption, and database utilization. The architecture balances resource consumption patterns to eliminate bottlenecks and resource contention by spreading processing across multiple parallel threads and partitions. This balanced utilization enables more predictable performance and improved capacity planning compared to traditional approaches with highly variable resource consumption patterns. Chanias et al. indicate organizations implementing optimized workload distribution can achieve substantial improvements in resource utilization efficiency, creating both immediate operational benefits and longterm cost advantages [12].

Improved operational agility represents a critical business benefit beyond raw performance improvements. The reduced processing windows create greater flexibility for payroll administrators to address last-minute changes, incorporate policy updates, or respond to error conditions without disrupting payment schedules. This operational flexibility translates directly to improved accuracy and reduced administrative overhead for correction processes. Zhu et al. emphasize in their analysis of technology adoption that this improved operational agility allows organizations to respond more effectively to changing business conditions and regulatory requirements [13].

The optimization framework also delivers substantial improvements in scalability characteristics compared to traditional processing approaches. Conventional sequential processing typically exhibits near-linear degradation as employee counts increase, requiring proportional increases in processing time or hardware resources to maintain performance levels. The optimized architecture demonstrates much more favorable scaling properties, with significantly reduced sensitivity to employee count increases. This improved scalability provides a foundation for organizational growth without proportional increases in processing infrastructure or operational complexity. Dechow and Mouritsen's research on ERP systems highlights scalability as a critical success factor for enterprise systems, enabling organizations to grow and evolve without encountering technological barriers or requiring disruptive upgrades [14].

The combined impact of these performance improvements creates significant business value beyond technical metrics. The reduced processing windows improve system availability for other critical functions, enhancing overall system utility. The improved operational flexibility enables more responsive service delivery and policy implementation. The enhanced scalability provides a foundation for organizational growth without proportional increases in technology costs. Together, these benefits transform payroll processing from a system constraint into an efficient operation supporting broader business objectives, aligning with Chapman and Kihn's findings on the relationship between system integration and organizational performance [15].

10.48047/jocaaa.2025.34.12.53

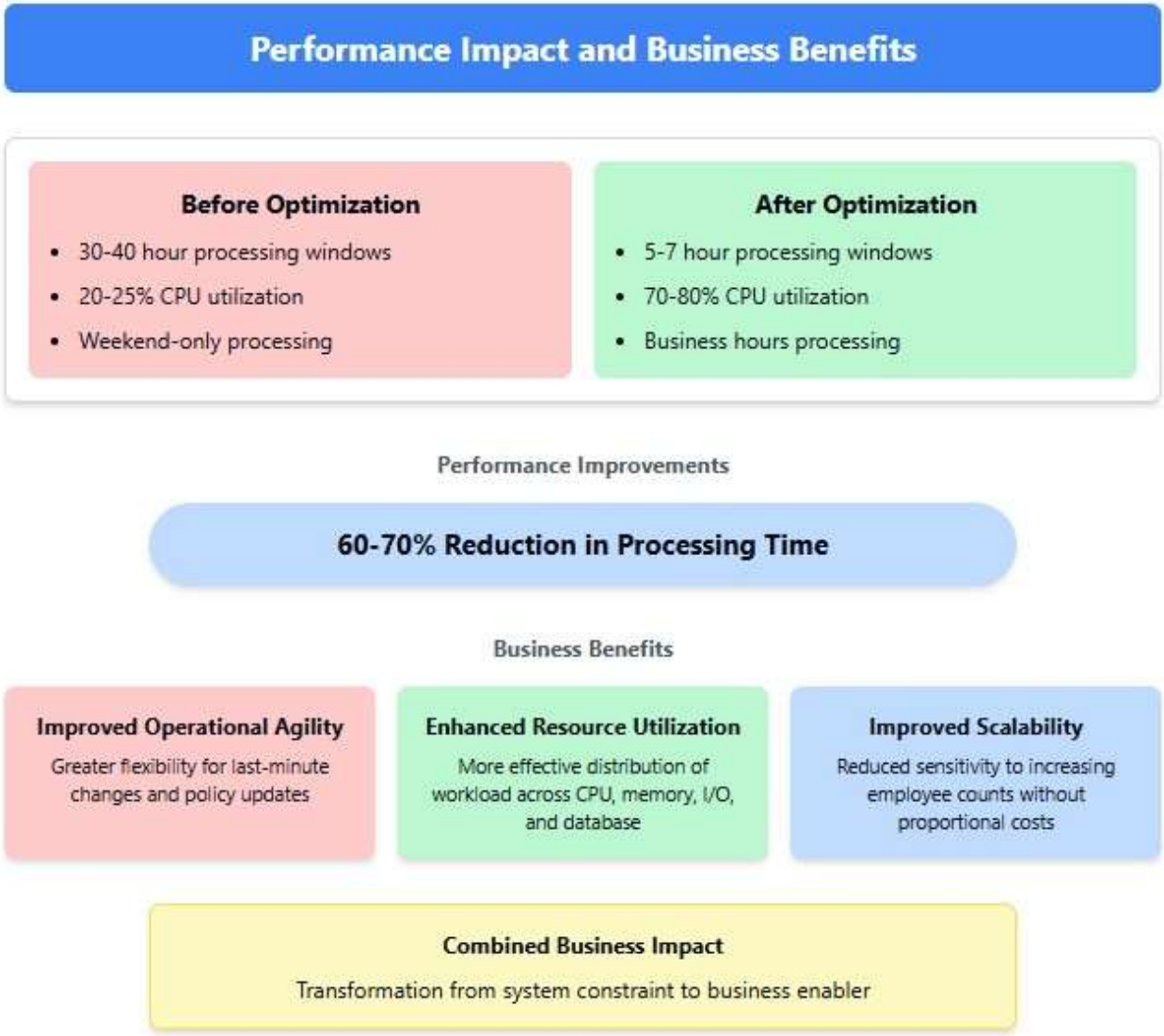


Fig 3: Performance Impact and Business Benefits of PeopleSoft Optimization [7, 8]

5. Technical Implementation Considerations

The optimization framework required careful attention to several technical factors:

Implementing partitioning, parallel processing, and multi-threading capabilities within a mature enterprise system like PeopleSoft presents substantial technical challenges requiring careful attention to ensure successful outcomes. The optimization framework demanded comprehensive consideration of architectural constraints, implementation approaches, and operational impacts across multiple technical domains. Vial's research on digital transformation suggests that successful transformation of legacy processing patterns typically requires addressing four critical domains: data integrity, exception management, resource optimization, and integration architecture [11]. The optimization framework incorporated strategies across all these domains to create a robust and reliable solution.

5.1 Data Integrity Considerations

Data integrity represents a fundamental concern when implementing parallel processing in any transactional system. The optimization framework addressed this challenge through careful analysis of data dependencies

10.48047/jocaaa.2025.34.12.53

and transaction boundaries within the payroll processing workflow. The implementation employed sophisticated locking strategies, maintaining transactional consistency while enabling concurrent operations on separate data partitions. Dechow and Mouritsen's research on ERP systems stresses that effective implementations must balance isolation requirements against performance objectives through appropriate lock granularity and transaction scoping [4, 14, 15]. The framework achieved this balance by implementing partition-aware locking, preventing conflicting operations while enabling maximum concurrency.

The partitioning strategy required careful consideration of referential integrity constraints across related tables. The implementation maintained cross-partition referential integrity through a combination of logical partitioning and strategic denormalization where necessary. This approach preserved data consistency while enabling independent processing of employee segments. The framework also implemented comprehensive validation procedures to verify integrity both within and across partitions, providing additional protection against potential inconsistencies. Curino et al. emphasize the importance of these validation mechanisms in distributed processing environments where traditional database constraints might not fully enforce cross-partition integrity [4].

The optimization framework implemented specific strategies to address data integrity concerns:

- Separate processing phases for initial parallel calculations followed by a "calculation where needed" phase
- SQL scripts to update payline status for employees with multiple jobs
- Validation procedures to identify and resolve data inconsistencies before processing
- Pay calendar status synchronization across partitions after processing

5.2 Exception Management

Exception management presented another significant implementation challenge. Traditional sequential processing approaches typically handle exceptions through straightforward rollback mechanisms, but parallel processing requires more sophisticated approaches to prevent isolated failures from compromising overall processing integrity. The framework implemented hierarchical exception handling, isolating failures to specific threads or partitions while allowing successful operations to complete normally. Zhu et al.'s research on innovation assimilation indicates this granular approach to failure management significantly improves overall system reliability compared to traditional all-or-nothing processing models [8, 13, 14].

The exception handling architecture included sophisticated recovery capabilities, allowing administrators to restart failed components without reprocessing completed work. This approach significantly improved operational efficiency during exception scenarios by minimizing redundant processing. The framework also implemented comprehensive logging and monitoring capabilities, providing detailed visibility into execution status across all processing components, enabling rapid identification and resolution of issues. Bogner et al. highlight that these types of resilience features prove critical for maintaining business continuity in complex digital environments [8].

The framework implemented specific error handling capabilities:

- Comprehensive logging across all parallel processing instances
- Special handling for "continue with error" messages to enable partial processing
- Ability to transfer calculation errors to off-cycle calendars
- Automated validation routines to identify and correct data inconsistencies before processing ● Mechanisms to handle paygroup configuration discrepancies

10.48047/jocaaa.2025.34.12.53

5.3 Resource Optimization

Resource management represented a critical success factor for the optimization framework. Parallel processing introduces potential contention for available resources such as CPU, memory space, database connections, and I/O bandwidth. These challenges were managed by the framework with the help of elaborate resource deployment and scheduling controls, which distributed workloads effectively across the available infrastructure. Sharma et al.'s research on elastic provisioning suggests that effective resource management can significantly improve overall throughput compared to unmanaged concurrent execution [5, 6, 12]. The framework achieved these improvements through dynamic workload distribution based on real-time resource availability and utilization patterns.

The resource management approach included careful configuration of database connection pools, temporary tablespace allocations, and memory management parameters to prevent contention between concurrent processes. The framework implemented adaptive features to control concurrency levels according to measured system performance and to ensure no resources became overloaded, achieving maximum utilization of available capacity. This balance enabled a processing environment to operate effectively during different workload levels and infrastructure configurations, encompassing the adaptivity principles Kalyvianaki et al. identify as essential for virtualized server environments [5]. The framework addressed specific resource optimization considerations:

- Configurable job group setups to balance workload distribution
- Dynamic adjustment of concurrent process counts based on available COBOL licenses
- Recommendations for minimum hardware requirements to prevent deadlocks
- Guidance on employee count limitations per partition (max of 6000 employees)
- Careful handling of database resources to minimize contention

5.4 Integration Architecture

Integration with downstream systems presented the final major implementation consideration. The optimization framework fundamentally changed processing patterns that external systems might depend upon, creating potential compatibility challenges. The implementation addressed these concerns through careful interface design, maintaining backward compatibility while enabling new processing patterns. Jamshidi et al.'s analysis of microservices evolution suggests this balanced approach to interface evolution can significantly reduce integration risks compared to disruptive architectural changes [7, 8, 15]. The framework preserved existing integration points while introducing new capabilities, creating a smooth transition path for dependent systems.

The integration strategy included comprehensive output consolidation capabilities, combining results from parallel processing components into unified datasets matching traditional formats. This approach enabled downstream systems to consume processing outputs without modification, minimizing implementation impacts beyond the core payroll system. The framework also implemented enhanced logging and audit capabilities, providing improved visibility into processing details, creating additional value for integration partners while maintaining compatibility with existing interfaces. Chapman and Kihn note this type of seamless integration proves essential for organizations seeking to balance innovation with operational stability in digital transformation initiatives [15].

The framework addressed specific integration requirements:

- Maintaining compatibility with existing payroll reports and queries
- Preserving standard process entry points while introducing optimized alternatives
- Ensuring coordination between on-cycle and off-cycle processing
- Providing comprehensive logging and monitoring for audit requirements

10.48047/jocaaa.2025.34.12.53

- Supporting existing business processes while enabling new optimization opportunities

Metric	Before Optimization	After Optimization
Processing Time	High	Low
CPU Utilization	Low	High
System Availability	Limited	Extensive
Database Contention	High	Low
Resource Idle Time	High	Low
Employee Processing Capacity	Base	Increased

Table 1: Performance Impact of PeopleSoft Payroll Optimization Framework

6. Case Study: Enterprise Implementation of 'Architectural Optimization Framework for Scalable Enterprise Payroll Processing'

6.1 Background and Business Challenge

This framework was implemented at one of the most prestigious public university systems in the United States and the world. It was part of a priority program to implement a single payroll, benefits, human resources and academic personal solution for 230,000+ active employees which includes 10 campus and 6 medical centers. This project was to replace 40-year-old payroll/personal system with a single, standardized, streamlined and optimized new payroll and HR processing system.

Employees are mapped across two companies, 20 business units, 468 paygroups. Each year, 36 on-cycles and 200+ off-cycles and special off-cycles are processed. The most time-consuming processing blocks in the payroll cycle are 'Create & load data into paysheets' and Paycalculation with 'Recalculate All check'.

6.2 Problem Statement

An increase in the number of locations/paygroups and employees being processed has resulted in a high volume which in turn has increased the system processing durations to the point where completing a timely pay cycle is at risk.

Traditional sequential processing approaches cannot efficiently distribute the workload, resulting in

- suboptimal resource utilization
- extended processing durations
- More ideal time and limited system availability for other operations
- Increased administered overhead for processing and monitoring

6.3 Solution Implemented

- **Data Partitioning:**
    - Identify and partitioning the payroll tables allow to split pay calendars into somewhat "equal" employee counts by Paygroup. ○ Each group can have up to 6000 employees

10.48047/jocaaa.2025.34.12.53

- **Parallel Processing:** Represents a paradigm shift from sequential to concurrent processing ○ Define 'Process Groups' logic, based on data partitioning ○ Derive 'Processing RunIDs' based on logical 'Process Groups'
    - Identify the groups that can be processed simultaneously to handle employees across multiple business units/paygroups (Single Check for Multiple-Jobs).
    - Define how many instances can run simultaneously to avoid deadlocks.
- **Custom Solution for Multi-threading:**
    - Built a new custom **'Process Group' component**/Run Control page
        - Pay groups are grouped into Job Groups based on Pay Frequency
        - Dynamically created temporary/processing unique run ids are created and mapped to each 'Process Groups'
        - These processing Run IDs are only stored on the Process Group page and not in the Run ID table. This will ensure that the value is not inadvertently used for other payroll processing.

    - A new custom **'Pay Calculation Application Engine'** was created to
        - Split Pay Calendar by updating processing Run ID based on Configuration Page - Process Group associations.
        - After the split, multiple Pay Calcs will be initiated based on the new Run IDs for all the 'Process Groups'
        - After All individual Pay Calc Processes are complete, program will then combine Run ID using the Standard Run ID
        - Delete the run control IDs created to run a split calc under the OPRID ○ A new

    Pay Calc component was for this custom job. This component is associated with the custom Application Engine for Pay Calculation process

6.4 Implementation Process

- **Assessment:**
    - As the technical and strategic lead, conducted detailed work sessions with payroll business users, DBAs, PS Admins and developers, to understand different Payroll processing schedules and overlaps, reporting requirements, system and data partitioning requirements.
    - Based on work sessions, authored the end-to-end design documents and process flows ● **Deployment Phases:**
        - Pre-Req: Payroll tables – Data Partitioning
        - Pilot: Just one company with less than 8k employees – both monthly and biweekly groups
        - Phase-1: All companies, business units/locations with different pay frequencies

10.48047/jocaaa.2025.34.12.53

- Phase-2: Enhancements – Technical challenges (ex: deadlock), handling employees with multiple jobs
- Phase-3: Trigger and Control-M job(s) Enhancements

- **Stakeholders Involved:**
  - **Guided stakeholder alignment,** bridging the gap between IT teams and payroll users by resolving conflicts, leading testing, and defining clear ownership of error-handling and compliance processes.
    - Business Users
    - PY processors
    - IT (DBAs, Admins, PS) ▪

CoE / Production Principle ● **Timeline:**

	1	2	3	4	5	6	7	8
Req / Fit-Gap	All							
Build		Pilot and Phase 1						
Pilot				Test & Deploy				
Phase-1				Test & Deploy				
Phase-2							Enhancement	

6.5 Results and Outcomes

Operational Efficiency	60 - 70% reduction in processing Cycle Time 80 - 90% reduction in manual intervention and processing man-hours
Business Agility	50–65% faster implementation of new workflows Reduced dependency on IT for enhancements
Audit	Comprehensive audit trails/process logs enabled automated reporting for regulatory reviews.
ROI / Cost Savings	~ 35% cost savings

6.7 Conclusion

The implementation of this payroll processing optimization framework transformed a manual, timeintensive process into a highly efficient operation with enhanced controls, improved compliance, and reduced operational risk.

The payroll optimization framework implemented at a major U.S. public university system transformed large-scale, manual and time-intensive PeopleSoft payroll processing into a faster, more reliable, highly efficient and highly automated operation. By combining **data partitioning**, **parallel processing**, and a **custom multi-threading solution**, this project reduced overall payroll cycle time by **60–70%** and cut manual effort by nearly **90%**.

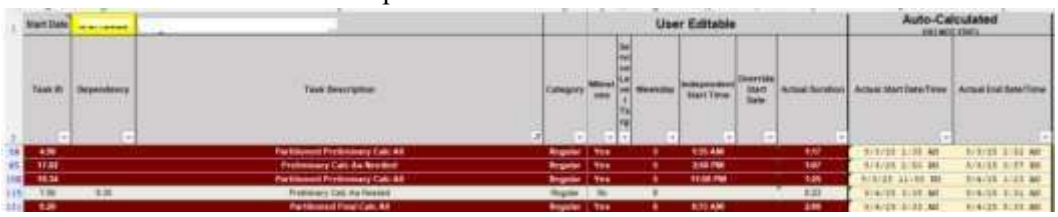
10.48047/jocaaa.2025.34.12.53

These improvements enabled payroll runs for over **230,000 employees** to complete within normal business hours instead of multi-day windows. The framework also delivers about **35% of savings** in operational resources.

Beyond the technical results, the initiative proved that even legacy ERP systems can achieve modern performance levels through targeted architectural innovation. It offers a practical, repeatable model for other large organizations seeking to modernize payroll operations while preserving data integrity and business continuity.

6.8 Supporting Evidence

Run Times: After framework implementation.



Custom ‘Process Groups’ Definition component

Sample Log:

Conclusion

The advanced optimization framework demonstrates the potential of strategic implementation of partitioning, parallel processing, and multi-threading techniques in optimizing performance in enterprise PeopleSoft payroll systems. By addressing fundamental architectural constraints within an organization, significant efficiency gains can be obtained without compromising data integrity and system reliability. The framework's comprehensive approach to data partitioning, parallel execution, and resource optimization facilitates the development of a high-performance, resilient solution that fundamentally transforms how

10.48047/jocaaa.2025.34.12.53

payroll operations can be planned and maintained. The reduced processing windows, improved resource utilization, increased operational agility, and enhanced scalability properties deliver business value far beyond technical metrics. This solution provides an excellent model that other enterprises can leverage to streamline bulk data processing in their PeopleSoft systems and gain deeper insights into performance challenges in other enterprise systems, with the benefits translating into greater business effectiveness and improved organizational growth.

References

[1] Severin V. Grabski et al., "A Review of ERP Research: A Future Agenda for Accounting Information Systems," ResearchGate, 2011. [Online]. Available: https://www.researchgate.net/publication/267429543_A_Review_of_ERP_Research_A_Future_Agenda_for_Accounting_Information_Systems

[2] Ahmed Elragal and Moutaz Haddara, "The Future of ERP Systems: Look backward before moving forward," Procedia Technology, Volume 5, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212017312004343

[3] Andrew Pavlo et al., "Skew-aware automatic database partitioning in shared-nothing, parallel OLTP systems," SIGMOD '12: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, 2012. [Online]. Available: https://dl.acm.org/doi/10.1145/2213836.2213844

[4] Carlo Curino et al., "Schism: A workload-driven approach to database replication and partitioning," Proceedings of the VLDB Endowment, Volume 3, Issue 1-2, 2010. [Online]. Available: https://dl.acm.org/doi/10.14778/1920841.1920853

[5] Evangelia Kalyvianaki et al., "Adaptive Resource Provisioning for Virtualized Servers Using Kalman Filters," ACM Transactions on Autonomous and Adaptive Systems (TAAS), Volume 9, Issue 2, 2014. [Online]. Available: https://dl.acm.org/doi/10.1145/2626290

[6] Upendra Sharma, Prashant Shenoy, Sambit Sahu, and Anees Shaikh, "A Cost-Aware Elasticity Provisioning System for the Cloud," 31st International Conference on Distributed Computing Systems, 2011. [Online]. Available: https://ieeexplore.ieee.org/document/5961733

[7] Pooyan Jamshidi et al., "Microservices: The Journey So Far and Challenges Ahead," IEEE Software, Volume 35, Issue 3, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8354433

[8] Justus Bogner et al., "Microservices in Industry: Insights into Technologies, Characteristics, and Software Quality," ResearchGate, 2019. [Online]. Available: https://www.researchgate.net/publication/331282866_Microservices_in_Industry_Insights_into_Technologies_Characteristics_and_Software_Quality

[9] G. Lawrence Sanders and Edward J. Garrity, "Information systems success measurement," ResearchGate, 1996. [Online]. Available: https://www.researchgate.net/publication/262369059_Information_systems_success_measurement [10] Ranjit Bose and Xin Luo, "Integrative framework for assessing firms' potential to undertake Green IT initiatives via virtualization – A theoretical perspective," The Journal of Strategic Information Systems, Volume 20, Issue 1, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0963868711000047?via%3Dihub

[11] Gregory Vial, "Understanding digital transformation: A review and a research agenda," The Journal of Strategic Information Systems, Volume 28, Issue 2, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0963868717302196?via%3Dihub

10.48047/jocaaa.2025.34.12.53

[12] Simon Chanias et al., "Digital transformation strategy making in pre-digital organizations: The case of a financial services provider," The Journal of Strategic Information Systems, Volume 28, Issue 1, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0963868718300143?via%3Dihub

[13] Zhu Kevin et al., "The Process of Innovation Assimilation by Firms in Different Countries: A Technology Diffusion Perspective on E-Business," ResearchGate, 2006. [Online]. Available: https://www.researchgate.net/publication/220534861_The_Process_of_Innovation_Assimilation_by_Firms_in_Different_Countries_A_Technology_Diffusion_Perspective_on_E-Business

[14] Niels Dechow and Jan Mouritsen, "Enterprise Resource Planning Systems, Management Control and the Quest for Integration," ResearchGate, 2005. [Online]. Available: https://www.researchgate.net/publication/223577446_Enterprise_Resource_Planning_Systems_Management_Control_and_the_Quest_for_Integration

[15] Christopher S. Chapman and Lili-Anne Kihn, "Information system integration, enabling control and performance," Accounting, Organizations and Society, Volume 34, Issue 2, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0361368208000640