WHITE PAPER

The Third Path:

Why Native Linux Recompilation is the

Overlooked Alternative to IBM i Refactoring and Hosting

An examination of three modernization pathways available to the global IBM i installed base, the economic and operational trade-offs of each, and the case for in-place recompilation to native Linux on hyperscale cloud infrastructure.

Dr. Bruce Acacio
Chief Executive Officer
Infinite Corporation

May 2026

Abstract

Approximately 120,000 organizations worldwide continue to operate business-critical applications on IBM i (formerly AS/400) systems, representing one of the largest concentrations of mission-critical legacy code in commercial computing[1]. These applications — written predominantly in RPG, COBOL, Control Language (CL), and Data Description Specifications (DDS) — process transactions for global manufacturers, regional financial institutions, insurance carriers, and government agencies. Yet despite consistent industry surveys indicating strong intent to modernize, the actual migration rate remains low. This paper argues that the persistence of IBM i workloads is not principally a matter of customer reluctance, but a function of the two dominant modernization strategies marketed to this segment being structurally inadequate for the median customer. Refactoring to Java imposes multi-year timelines, costs frequently exceeding ten million dollars, and a complete workforce transition. Lift-and-shift to hosted IBM i environments — Skytap on Microsoft Azure, IBM Power Virtual Server — preserves the application stack but leaves customers indefinitely dependent on IBM Power hardware, IBM licensing, and the same skills crisis they sought to escape. A third pathway, native Linux recompilation, has received markedly less industry attention despite addressing the limitations of both incumbent approaches. This paper examines the comparative economics, operational profiles, and risk characteristics of all three strategies, draws on documented enterprise case studies, and concludes that recompilation is the appropriate default strategy for organizations whose objective is platform exit without code rewrite.

Keywords: *IBM i, AS/400, legacy modernization, RPG, COBOL, cloud migration, application refactoring, technology debt, native recompilation*

1. Introduction

In conversations with information technology leaders responsible for IBM i estates, one observation recurs with striking frequency: the intention to migrate off the platform has been on the strategic roadmap for five years or more. Successive iterations of Fortra's annual IBM i Marketplace Study confirm this in aggregate. The mainstream IT analyst community — IDC, Gartner, Forrester — has effectively stopped tracking the IBM i installed base in any granular way[2], which has left the customer base reliant on community-led research for visibility into its own market dynamics. What that research consistently shows is a customer base aware that its platform is aging, possessing senior leadership endorsement for modernization, and yet largely unable to execute on that intent.

The conventional explanation attributes this inertia to customer-side factors: organizational risk aversion, the complexity of legacy codebases, budget constraints, or the unique difficulty of the IBM i environment. This paper proposes a different explanation. The principal reason customers have not migrated is that the two pathways the industry has aggressively marketed to them are, on close examination, poor fits for the majority of the installed base. The first — wholesale refactoring of RPG and COBOL applications into Java — imposes timelines and costs that compound the original technical debt rather than retiring it. The second — lift-and-shift to a hosted IBM i environment in a third-party cloud — relocates the workload without addressing any of its structural liabilities.

A third pathway exists and is implemented at production scale by a small number of vendors, but it has received markedly less industry coverage. This approach involves recompiling RPG, COBOL, CL, and DDS source code, without modification, into native binaries that execute directly on Linux running on commodity x86 or ARM cloud infrastructure. The application's business logic is preserved in its original language. The development team retains its skills and continues to maintain the codebase using the same source. The underlying platform, however, becomes Amazon Web Services, Microsoft Azure, or Google Cloud Platform — with all the cost, scalability, and integration advantages those platforms confer.

This paper contributes three things to the modernization discourse. First, it characterizes the structural mismatch between incumbent modernization strategies and the operational reality of the median IBM i customer. Second, it presents recompilation as a distinct architectural pathway and analyzes its trade-offs honestly, including the scenarios in which it is not the right answer. Third, it draws on enterprise case data to estimate realistic timelines and economics for the recompilation approach. The intended audience is enterprise IT leadership, system integration partners, and hyperscale cloud account teams whose customers run IBM i workloads.

> *The path exists. The industry has not yet told the IBM i customer base about it at the scale or with the clarity that the choice deserves.*

— Author observation, based on customer engagements 2023–2026

2. The IBM i Installed Base and the Persistence Problem

2.1 Scale and Composition

Estimates of the global IBM i installed base have remained remarkably stable, with the most current community-validated figure placing it at approximately 120,000 organizations operating one or more IBM i systems as a primary platform[1]. The platform is heavily concentrated in industries with long-lived transactional workloads: discrete manufacturing, distribution and wholesale, regional and community banking, insurance underwriting and claims, healthcare administration, retail, and certain segments of state and local government. Customers range from mid-market enterprises with single-LPAR Power Systems installations to large multinationals — Honda, Toyota, the Walt Disney Company, and segments of the United States Air Force have been publicly identified as continuing IBM i users[3].

Within this installed base, a clear bifurcation exists between customers maintaining relatively current hardware and operating system levels — typically estimated at 25,000 to 30,000 organizations — and a substantially larger group of laggards running IBM i on older Power generations and on operating system releases that are either approaching or have already passed end-of-standard-support. The 2025 Fortra Marketplace Survey indicates that IBM i 7.4, an operating system release dating to 2019, remains the primary release at approximately fifty percent of surveyed installations, with the more current IBM i 7.5 deployed at only 8.1 percent of sites. IBM announced in September 2025 that it would withdraw IBM i 7.4 from marketing on April 30, 2026, which will accelerate end-of-support pressure across roughly half the installed base.

2.2 The Application Layer

The strategic difficulty of IBM i modernization is not the operating system. It is the applications. Approximately seventy percent of IBM i installations operate at least one substantial homegrown application written in RPG, with most environments also containing CL procedures, DDS-described database structures, and significant volumes of COBOL. These applications are not peripheral. They are typically the core systems of record for the businesses that run them: order entry and fulfillment, general ledger, claims processing, manufacturing resource planning, dealer management, member administration. They embed decades of accumulated business rules — many of which exist nowhere else, including not in documentation.

This combination of business criticality and tribal knowledge produces what might be termed the persistence problem. A customer cannot simply replace these applications with a packaged ERP or SaaS equivalent without a multi-year functional re-implementation project, and most customers know it. Yet the applications themselves, being written in languages with declining commercial vitality, increasingly resemble a strategic liability. The question is not whether to modernize. It is which pathway to choose.

2.3 The Skills Crisis as a Forcing Function

Compounding the strategic question is a demographic one. The RPG and COBOL workforce that maintains these applications is aging rapidly. Multiple industry analyses converge on the following figures: the average COBOL developer is approximately 55 to 58 years old, with roughly ten percent of the workforce retiring annually[4,5]. The RPG workforce is older still. ASNA, a vendor focused on the IBM i installed base, models the typical RPG programmer as being approximately seventy years old in 2025[6], and notes the obvious corollary: very few professionals work past seventy regardless of intention. Sixty percent of organizations operating COBOL infrastructure report that finding skilled developers is their single largest operational challenge[7].

Universities largely stopped teaching either language two decades ago. The pipeline of new entrants is effectively non-existent except through specialized retraining programs operated by a handful of vendors and consultancies. This means that for any IBM i customer with a five-year planning horizon, the question is no longer whether the workforce will be available to maintain the existing platform. It is when, precisely, the inability to maintain it will become a business-continuity issue. The skills crisis is therefore not merely a backdrop to the modernization decision. For many customers, it is the forcing function.

3. Path A: Refactoring to Java

3.1 Definition and Approach

The first modernization pathway, and the one most aggressively marketed by large system integrators and hyperscale cloud providers, is automated or semi-automated refactoring of RPG, COBOL, and ancillary code into Java, with the application's data layer migrated to a relational database such as PostgreSQL, Oracle, or, in some deployments, the cloud-native equivalents (Amazon Aurora, Azure Database). Representative offerings in this category include AWS Mainframe Modernization with the Blu Age conversion engine, several offerings from Fresche Solutions (including the X-Modernize and X-Analysis toolchains, marketed together as a Modernization-as-a-Service subscription), and the modernization practices of major consulting firms.

The technical premise of refactoring is that the existing application's business logic can be preserved through automated translation, with manual remediation handling the cases that the translator cannot process correctly. Vendors in this space frequently cite automation rates of 70 to 90 percent, with the residual fraction requiring human intervention. The resulting application is, in form, an entirely new system: Java services, typically Spring-based, with modern web user interfaces (Angular or React) and integrations with the broader cloud-native ecosystem.

3.2 The Economics and Timeline

The honest economic profile of refactoring is significantly less favorable than is typically presented at the point of sale. Gartner has estimated that the average COBOL application refactoring engagement requires approximately 600 person-years of effort across analysis, conversion, remediation, and testing[8]. Industry timelines for a single substantial application range from twelve months for the simplest cases to thirty-six months for representative enterprise estates. Costs scale accordingly. For a mid-market customer with a typical IBM i application portfolio, total program cost frequently exceeds ten million United States dollars when system integration fees, internal labor, parallel operations during cutover, and post-cutover remediation are all included.

These costs are not anomalous outcomes. They are the central tendency of the approach. The Standish Group, which has tracked information technology project outcomes for over two decades across more than 80,000 documented engagements, has consistently reported that 60 to 70 percent of large IT projects are either outright failures or substantially over budget, behind schedule, or below specification[9]. Mainframe and IBM i refactoring projects, which sit toward the high-complexity end of this distribution, fail at rates that several industry observers have placed at approximately fifty percent — a figure cited by Fresche Solutions itself in defense of its incremental Modernization-as-a-Service approach[10].

3.3 The Skills Substitution Problem

Beyond the direct economics, refactoring imposes a less-discussed but equally significant cost: the wholesale replacement or retraining of the development workforce. An RPG team that has maintained a particular application for fifteen years and possesses irreplaceable knowledge of its quirks and undocumented business rules is, after refactoring, suddenly responsible for a Java/Spring/Angular stack in which most of them have no expertise. The institutional knowledge embodied in the original codebase persists, in compiled form, but the team's ability to navigate that knowledge in the new language is significantly diminished.

Customers undertaking refactoring projects therefore face one of two costly options. They can retrain the existing team, which slows the program and creates a multi-year period of reduced productivity. Alternatively, they can hire Java developers and gradually transition the old team off the application, accepting attrition of the institutional knowledge that the refactoring was supposed to preserve. Neither option is consistent with the timelines or costs typically presented at the start of the engagement.

3.4 When Refactoring Is the Right Answer

Refactoring is not, on these grounds, an illegitimate strategy. There are scenarios in which it is the correct choice. The most defensible cases are: (1) when the existing application is genuinely scheduled for substantial functional re-architecture in any case, such that the language is being changed at the same time the business logic is being rewritten; (2) when the customer has strategic reasons to standardize entirely on a Java or .NET stack across all systems of record; or (3) when the application is sufficiently small that the refactoring effort is bounded and the cost is recoverable within a defensible payback period. The error this paper identifies is not in refactoring per se but in its presentation as the default modernization pathway for the IBM i installed base, when for the median customer it is neither the lowest-cost nor the lowest-risk approach.

4. Path B: Hosted IBM i

4.1 Definition and Approach

The second pathway is, in a sense, the inverse of refactoring. Rather than transforming the application to fit a new platform, hosted IBM i preserves the application and the operating environment entirely and relocates the underlying Power Systems hardware to a third-party data center. The two principal offerings in this category are Skytap, which runs IBM Power workloads — IBM i, AIX, and Linux on Power — on virtualized Power infrastructure inside Microsoft Azure or IBM Cloud data centers, and IBM Power Virtual Server (PowerVS), IBM's own hosted Power Systems service. Both deliver an experience in which the customer's IBM i applications run substantively unchanged, with the customer paying a recurring fee for the hosted environment rather than capital expense for on-premises hardware.

This approach has legitimate appeal. It eliminates the need to operate Power Systems hardware in a corporate data center, reduces the burden on internal infrastructure teams, and provides a path out of legacy facilities. For organizations whose principal modernization driver is data center exit — rather than platform exit — it is a serviceable strategy and the implementation timeline is short, often a matter of weeks rather than months.

4.2 The Economic and Strategic Limits

The strategic question with hosted IBM i is what, precisely, has been modernized. The answer, on close examination, is the location of the hardware. Everything else is unchanged. The customer continues to pay IBM software licensing on the same terms as before — and IBM has raised those terms repeatedly in recent years, with multiple successive price increases on Power Systems software and PowerVS announced through 2025. The customer continues to depend on the IBM Power processor roadmap, including its capacity and pricing dynamics, both of which are controlled by a single vendor. The application continues to require RPG, COBOL, and CL maintenance skills, meaning the skills crisis described in Section 2.3 is not addressed in any way.

The economic profile reflects these limits. Hosted IBM i is not, in practice, materially cheaper than well-managed on-premises Power Systems infrastructure, and is in many configurations more expensive. The cost savings that customers expect from cloud migration — driven by hyperscaler economies of scale, elastic capacity, and competitive pricing pressure — apply principally to x86 and ARM compute on AWS, Azure, and GCP. They do not apply, or apply only weakly, to IBM Power workloads hosted in a cloud environment. The customer has moved from one IBM-priced environment to a different IBM-priced (or IBM-priced-adjacent) environment.

4.3 The Compounding Skills Problem

Hosted IBM i has a further characteristic that is rarely discussed: it preserves and arguably extends the customer's dependency on a labor market that is rapidly disappearing. An organization that lifts-and-shifts its IBM i estate to Skytap or PowerVS has secured itself a multi-year contract for hosted Power capacity. To extract value from that contract, it must continue to operate RPG and COBOL applications on the hosted environment, which means continuing to employ — or contract for — RPG and COBOL skills. The decision therefore extends the customer's exposure to the skills crisis by precisely the length of the hosting contract, typically three to five years.

> *Hosted IBM i is a data-center exit, not a platform modernization. It addresses the question of where the application runs without addressing the more consequential question of how, and by whom, the application is maintained.*
>
> *— Author analysis*

5. Path C: Native Linux Recompilation

5.1 The Technical Proposition

The third pathway, and the one this paper argues is the appropriate default for the median IBM i customer, is native recompilation of RPG, COBOL, CL, and DDS source code to binaries that execute directly on Linux operating on commodity cloud compute infrastructure. Architecturally, the approach treats RPG, COBOL, CL, and DDS as what they technically are — programming languages whose compilers can, in principle, target any sufficiently capable runtime. The difficulty of executing them outside of IBM i is not in the languages but in the runtime environment that surrounds them: the Integrated Language Environment (ILE), the DB2 for i database semantics, the EBCDIC character encoding, the job queue and subsystem model, data areas, data queues, and the broader set of operating system services on which IBM i applications implicitly depend.

A native recompilation toolset, of which Infinite i is one example, addresses this by implementing the IBM i runtime services on Linux as a compatibility layer beneath the recompiled binaries. The application is not aware that it has moved. Its calls to system services are answered by Linux-resident equivalents that preserve the original semantics. DB2 for i tables are presented to the application through interfaces that match DB2/400 syntax and behavior, even though the underlying storage may be PostgreSQL, MySQL, or a managed cloud database. EBCDIC data is handled transparently. ILE program activation, service program binding, and modular invocation operate as the application expects.

5.2 Operational Profile

The principal operational advantage of this approach is that it preserves the customer's existing development team and the existing source code unchanged. RPG and COBOL developers continue to maintain RPG and COBOL applications, using the same source files, with the same business logic, against semantically identical database structures. The migration is, from the development team's perspective, primarily a build-system change. The deployment target shifts from IBM i to Linux, but the source code is the source code.

This in turn produces a markedly different timeline profile than refactoring. Where Java refactoring engagements run twelve to thirty-six months, recompilation projects typically complete in 45 to 60 days for a representative enterprise application estate. The shorter timeline is not the result of cutting scope. It is the result of having vastly less work to do: no language translation, no business logic verification at the statement level, no user interface rewrite, no parallel team training, and no extended bug-triage period for translation defects that did not exist in the original code.

5.3 Published Case Outcomes

Two enterprise outcomes illustrate the practical results of this approach. AXA Insurance, a global insurance carrier, migrated a COBOL, CL, and DDS application estate from IBM i to Linux on a hyperscale cloud platform in approximately two months, with zero source code modification, and reported infrastructure cost reductions exceeding fifty percent versus the equivalent on-premises Power Systems footprint. Fidelity Investments completed a comparable migration of RPG applications in approximately four months, again with no source modification, and reported operating cost reductions of approximately thirty-five percent. Both organizations retained their existing development teams. Neither retrained those teams on a new language.

5.4 The Trade-offs

Recompilation is not the appropriate strategy in every scenario. Honest analysis requires acknowledging the cases in which it is not. First, if a customer's strategic objective is to retire the RPG and COBOL workforce — perhaps because the customer is consolidating its application portfolio on a single modern stack such as Java or .NET — recompilation does not advance that objective. The same source code persists on the new platform, and so the same skill requirements persist with it. Second, if the customer has determined that its applications must be substantially functionally re-architected — for example, decomposed into microservices, exposed through REST APIs, or rewritten to support a fundamentally new user interaction model — recompilation alone will not accomplish that. It will, however, position the customer to undertake such re-architecture on a more modern platform, at the customer's preferred pace, and without the parallel cost of operating Power Systems infrastructure. Third, for very small IBM i environments where the total cost of the existing platform is already low, the economic case for any migration — including recompilation — may be weak.

The customer profile for whom recompilation is most clearly correct is one whose principal modernization objective is platform exit while preserving the existing application portfolio and development team, with cost reduction and elimination of IBM hardware and licensing dependency as the secondary objectives. This profile describes a substantial fraction — plausibly the majority — of the 120,000-organization installed base.

6. Comparative Analysis

The three pathways differ along multiple dimensions that are not always commensurable, but the principal characteristics can be summarized in tabular form. Table 1 presents the comparison.

Table 1. Comparative characteristics of three IBM i modernization pathways.

Dimension	Refactor to Java	Hosted IBM i	Native Recompile
Code modification	Full conversion to Java	None	None
Target runtime	Java / Spring on hyperscaler	IBM i on virtualized Power	Native Linux on x86 / ARM
Cloud platform	AWS, Azure, GCP	Azure (Skytap) or IBM Cloud	AWS, Azure, GCP
IBM licensing after migration	Eliminated	Continues	Eliminated
Development team retained	Requires Java retraining	Retained	Retained
Typical timeline	12–36 months	Weeks (hosting only)	45–60 days
Cost vs. on-premises	Capital project + ongoing	Comparable or higher	30–60% reduction
Business-logic risk	High (translation defects)	None (unchanged)	None (unchanged)
Skills crisis exposure	Resolved post-cutover	Extended by contract term	Resolved at platform exit
Best fit	Strategic Java migration	Data-center exit only	Platform exit, code preserved

Several observations from this comparison merit emphasis. First, the timeline differential between refactoring and recompilation — typically an order of magnitude — is not a marginal variable. It is the difference between a strategic initiative that delivers within a budgetary planning cycle and one that consumes multiple cycles before producing operational value. Second, the elimination of IBM licensing exposure, which obtains under both refactoring and recompilation but not under hosted IBM i, is a substantial and often understated benefit. Third, the preservation of the development team under both hosted IBM i and recompilation, but not under refactoring, addresses a real institutional risk that customers consistently underweight at the planning stage.

The hosted IBM i pathway should be understood for what it is: a useful tactical solution for customers whose principal objective is to exit a physical data center while preserving the existing IBM i environment, with explicit acceptance of continued IBM dependency. It is not a strategic modernization. Recompilation, by contrast, achieves data-center exit, platform exit, and licensing exit simultaneously, in a timeline competitive with hosted IBM i for most application portfolios.

7. Economic Considerations

7.1 Total Cost of Ownership Framework

The economic case for any modernization pathway should be evaluated on a multi-year total cost of ownership basis, encompassing platform hardware or hosting fees, software licensing, application maintenance and development labor, infrastructure operations labor, and the amortized cost of the modernization program itself. Single-line comparisons of monthly hosting fees, which are common in vendor presentations, obscure the relevant decision.

Consider, as a representative example, a mid-market manufacturer or distributor with a single Power Systems S1024 or comparable installation, an IBM i software stack including the operating system, Db2 for i, and a small number of licensed program products, an application portfolio of approximately 1.5 million lines of RPG and COBOL, and an internal team of four to six application developers and one to two systems administrators. The annual run rate for such an environment, in 2026 conditions, typically falls in the range of three hundred and fifty thousand to five hundred thousand United States dollars in direct platform costs, exclusive of labor. The labor component, at fully loaded cost, adds another seven hundred thousand to one million dollars annually.

7.2 Comparative Economics by Pathway

Under a refactoring scenario, the customer faces a one-time program cost typically in the range of five to fifteen million dollars over twenty-four to thirty-six months, with significant variance depending on application complexity. During the program, the legacy platform must continue to operate, so platform costs are not reduced — and may actually increase, due to parallel-run requirements. Post-cutover, the platform cost falls substantially, but the labor profile shifts from RPG and COBOL maintenance to Java and front-end development, which is typically more expensive on a per-developer basis in current labor markets, even if the headcount is similar.

Under a hosted IBM i scenario, the customer typically achieves modest reductions in direct platform cost — perhaps fifteen to twenty-five percent — through the avoidance of on-premises facility costs and certain hardware refresh cycles. The labor profile is unchanged. IBM licensing continues at substantively the same rates, subject to periodic vendor-initiated increases. The cumulative five-year TCO is generally similar to or slightly above on-premises operation.

Under a native recompilation scenario, the customer pays a one-time fixed-fee migration program cost typically measured in low hundreds of thousands of dollars and completed in 45 to 60 days. Post-migration platform costs drop substantially — case studies consistently report thirty to sixty percent reductions, driven by the underlying economics of x86 and ARM cloud infrastructure relative to Power hardware. The labor profile is preserved at its prior cost. The cumulative five-year TCO is therefore meaningfully lower than either of the other two pathways.

7.3 Payback Period

The payback period is the metric where recompilation differentiates itself most sharply. Refactoring engagements, with multi-million-dollar program costs and multi-year timelines, frequently exhibit payback periods exceeding the useful life of the resulting application — meaning the modernization is, in economic terms, never fully recovered. Hosted IBM i, lacking material run-rate savings, has an indeterminate payback. Recompilation, with low program costs and substantial run-rate savings, typically achieves payback inside twelve months of go-live. This is not a marginal economic distinction. It is a categorical one.

8. Strategic Implications and Conclusion

8.1 Implications for IBM i Customers

The principal implication for organizations operating IBM i estates is that the modernization decision should not be framed as a binary choice between refactoring and continued operation. The three-pathway framework presented here is more accurate to the available options. For customers whose principal objective is platform exit while preserving the existing application portfolio — which describes a large fraction of the installed base — recompilation should be the default starting hypothesis, with refactoring reserved for cases where a parallel functional re-architecture is genuinely warranted, and hosted IBM i reserved for cases where data-center exit, not platform modernization, is the actual driver.

8.2 Implications for Hyperscale Cloud Providers

For Amazon Web Services, Microsoft Azure, and Google Cloud Platform, the IBM i installed base represents approximately 120,000 enterprise workloads that none of their native services can capture. The Power architecture cannot be virtualized on x86 or ARM compute, and the IBM i operating system is not portable in any conventional sense. The hyperscale providers' current offerings address this gap only partially: AWS Mainframe Modernization with Blu Age addresses some IBM i workloads through refactoring, with the timeline and cost limitations described above. Microsoft's relationship with Skytap addresses the gap through hosted IBM i. Neither approach captures the workload onto native hyperscaler compute. Recompilation does. A hyperscaler-aligned recompilation strategy would convert a substantial portion of the 120,000-organization installed base into consumers of native EC2, Azure VM, or Compute Engine capacity — workloads that are otherwise structurally unavailable to these providers.

8.3 Implications for System Integrators

For Kyndryl, IBM Consulting, Accenture, Wipro, and other major system integrators with IBM i modernization practices, recompilation does not displace the services opportunity. It changes its character. The customer still requires program management, data migration, integration testing, change management, and post-cutover support. What changes is the duration and complexity of the language conversion phase, which collapses from years to weeks. This is, on balance, an improvement in the partner's economic position: shorter engagements with higher confidence of completion, greater customer satisfaction, and a foundation for follow-on modernization work (functional re-architecture, integration, analytics) on the recompiled application portfolio.

8.4 Conclusion

The persistence of IBM i workloads is not a customer problem. It is a market-failure problem. The two pathways the industry has marketed at scale are not, for the median customer, well matched to the customer's actual objectives. The result has been a fifteen-year stall in modernization at the application layer, during which the underlying skills crisis has worsened, the IBM hardware and licensing roadmap has continued to consume customer budget, and the strategic value of the IBM i installed base to its operators has gradually eroded.

A third pathway exists. It has been implemented at production scale at enterprise customers including AXA Insurance and Fidelity Investments. It preserves the application portfolio, preserves the development team, exits the platform in a single quarter rather than over multiple years, and delivers a cost structure consistent with the underlying economics of hyperscale cloud computing. It is not the right answer in every scenario, and this paper has been deliberate in identifying the cases where it is not. But for the substantial majority of IBM i customers whose actual objective is to exit IBM Power while keeping their code and their team, it is the appropriate default. The industry has not yet told the customer base about it at the scale, or with the clarity, that the choice deserves. This paper has attempted to begin that correction.

References

This paper draws on community-tracked industry research, vendor case publications, and analyst commentary current through May 2026. Where IBM i installed base figures are cited, they reflect the community-validated estimates from the annual Fortra IBM i Marketplace Study and IT Jungle's installed-base modeling, which together constitute the most reliable public data on the platform now that mainstream analyst coverage has lapsed.

1. Prickett Morgan, T. (2025, February 10). State of the Power Systems base 2025: The operating systems. IT Jungle. Retrieved from https://www.itjungle.com/2025/02/10/state-of-the-power-systems-base-2025-the-operating-systems/
2. Prickett Morgan, T. (2025, October 27). Please take the IBM i Marketplace Survey. IT Jungle. Retrieved from https://www.itjungle.com/2025/10/27/please-take-the-ibm-i-marketplace-survey-2/
3. Software Engineering of America. (2023). What will the future of IBM i look like? Seasoft. Retrieved from https://seasoft.com/blog/ibm-i/what-will-the-future-of-ibm-i-look-like/
4. Pragmatic Coders. (2025, September 2). 2025 legacy code stats: Costs, risks & modernization. Retrieved from https://www.pragmaticcoders.com/resources/legacy-code-stats
5. Fazio, C. (2025, December 12). Why COBOL still matters. DistantJob. Retrieved from https://distantjob.com/blog/why-cobol-still-matters/
6. ASNA. (n.d.). The IBM i RPG crisis decade. ASNA White Papers. Retrieved from https://www.asna.com/en/white-paper/ibm-i-crisis-decade
7. MoldStud. (2024, August 27). What are the biggest challenges faced by COBOL developers? Retrieved from https://moldstud.com/articles/p-what-are-the-biggest-challenges-faced-by-cobol-developers
8. Planet Mainframe. (2024, October 2). Mainframe modernization isn't about mainframe migration. Retrieved from https://planetmainframe.com/2024/10/mainframe-modernization-isnt-about-mainframe-migration/
9. The Standish Group International. (Various years). CHAOS reports on information technology project outcomes. As summarized in Astadia (2023, June 19). Why IT projects fail (most managers don't know). Retrieved from https://www.astadia.com/blog/why-it-projects-fail
10. Fresche Solutions. (n.d.). Modernization-as-a-Service: Reducing the risk of legacy modernization. (Fresche cites a 49% failure rate for full code rewrites in its X-Modernize and X-Analysis marketing materials.)
11. Prickett Morgan, T. (2025, September 22). And then there were two: Big Blue withdraws IBM i 7.4. IT Jungle. Retrieved from https://www.itjungle.com/2025/09/22/and-then-there-were-two-big-blue-withdraws-ibm-i-7-4/
12. Prickett Morgan, T. (2026, January 12). 2025: An IBM i year in review. IT Jungle. Retrieved from https://www.itjungle.com/2026/01/12/2025-an-ibm-i-year-in-review/
13. LANSA. (2025, April 18). RPG programming language: All you need to know. Retrieved from https://lansa.com/blog/application-modernization/what-is-rpg-programming-language-used-for/

14. Quinnox. (2025, May 30). Legacy mainframe modernization: A complete guide for 2025. Retrieved from https://www.quinnox.com/blogs/legacy-mainframe-modernization/

15. AveriSource. (n.d.). What is legacy modernization? Retrieved from https://www.averisource.com/topics/what-is-legacy-modernization

About the Author

Dr. Bruce Acacio is the Chief Executive Officer of Infinite Corporation, the company that develops Infinite i, a toolset that recompiles RPG, COBOL, CL, and DDS applications natively to Linux on hyperscale cloud infrastructure. He has worked in the IBM i ecosystem for over three decades and writes regularly on application modernization strategy.

Infinite Corporation does not endorse or disparage any specific vendor mentioned in this paper. Where named vendors are cited (Skytap, IBM, Fresche Solutions, AXA, Fidelity, Kyndryl, AWS, Microsoft, Google), citations are drawn from publicly available sources, and the analysis represents the author's professional assessment, not a statement on behalf of those organizations.

Comments and correspondence: infinitecorporation.com