new relic

2024年 Javaエコシステムの現状

最も普及しているプログラミング言語の最新状況を解説

レポート / 2024年4月

目次

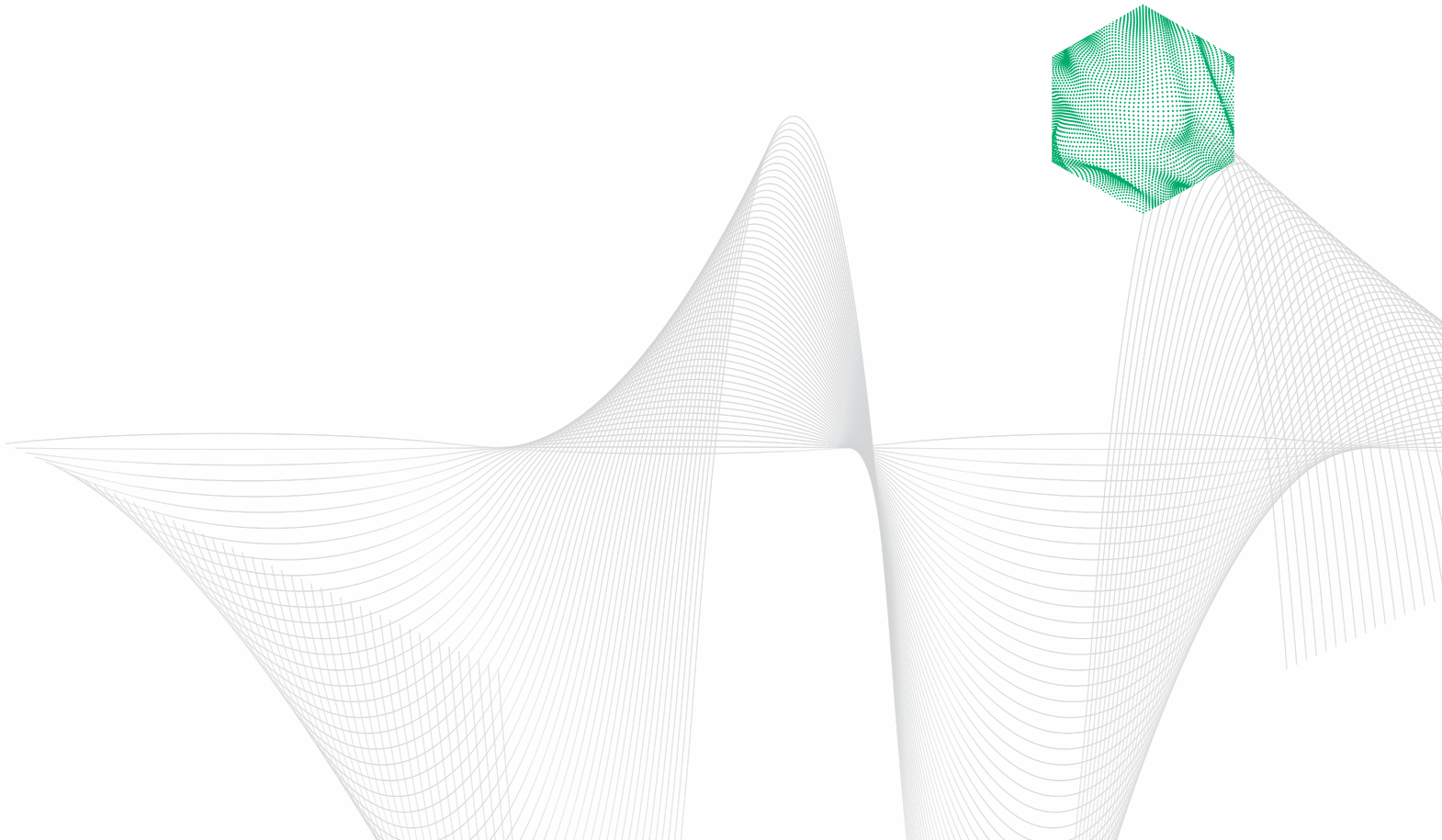
概要

New Relicは過去数年間、Javaエコシステムをチェックし続け、開発者の使用方法の変化を明らかにしてきました。例えば、Javaのバージョン毎の普及状況や、ベンダーとコミュニティがサポートするJavaDeveloper Kit（JDK）の傾向、New Relicを使用している開発者から寄せられるJava関連の質問の傾向などが含まれます。

本年次レポートは、New Relicに毎月報告される数十万のアプリケーションから得たデータに基づき、Javaエコシステムの現状についての洞察を提供します。

2024年のレポートでは、次のトピックについて検証します。

- 実運用で最も使用されているJavaのバージョン
- 最も人気のあるJDKベンダー
- JavaアプリケーションでのCPUとメモリの使用
- ロギング、暗号化、データベースに関して最も人気のあるJavaフレームワークとライブラリ
- 開発者から多く寄せられるJava関連の質問や要望



新しいJavaバージョンほど採用のスピードが速い

まず、実運用で最も使用されているJavaのバージョンを見てみましょう。

かつて、OracleはJDKのメジャーアップデートや変更に伴ってのみ新しいバージョンをリリースしていました。現在、Oracleは6カ月ごと（通常は3月と9月）に新しいJavaバージョンをリリースしており、各リリースにはいくつかの新機能とバグ修正が含まれています。また、Oracleは2年ごとに、新しいJavaの長期サポート（LTS）バージョンを発表しています。安定性、セキュリティ、パフォーマンスを改善するアップデートを含まれており、開発者がJavaバージョンをアップグレードする最も重要な要因のひとつとしてよく挙げられています。

Java 21のリリース後の6カ月間で、New Relicが監視しているアプリケーションの1.4%がJava 21を使用していました。Java 17は導入されてから6カ月間で、Java 17を使用していたアプリケーションはわずか0.37%でした。

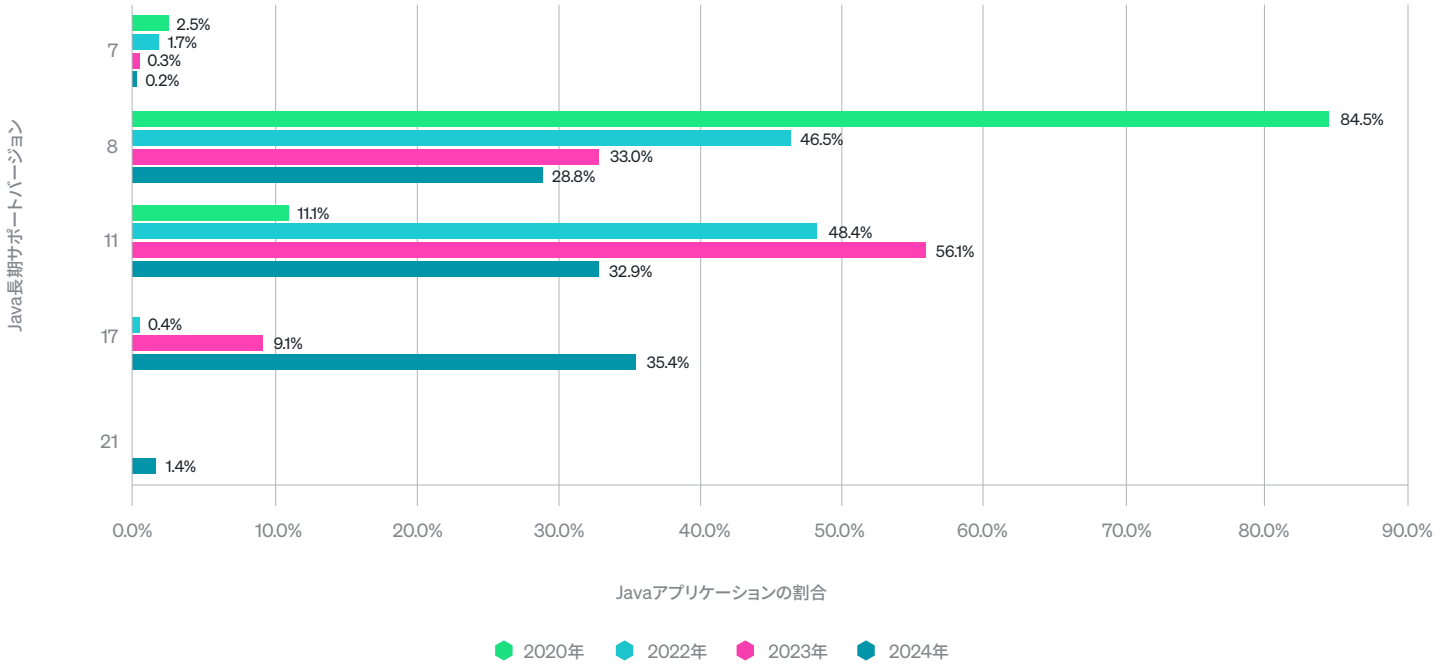
Java 21のリリース後の6カ月間で、New Relicが監視しているアプリケーションの1.4%がJava 21を使用していました。これを大局的に見ると、Java 17が導入されてから6カ月間で、Java 17を使用していたアプリケーションはわずか0.37%で、287%減少したことになります。

Java 17の採用率は、Java 11導入時の数字をはるかに上回っています。2023年にはアプリケーションの約10分の1 (9%) が実運用でJava 17を使用していましたが、現在ではアプリケーションの35%がJava 17を使用しており、1年間でほぼ300%の成長率を示してます。Java 11がそのレベルに近づくには何年もかかりました。

Java非LTSバージョンを使用しているアプリケーションは2%未満でした。通常、実運用では使用されないため、これは当然の結果です。

35%

Java 17を使用しているアプリ



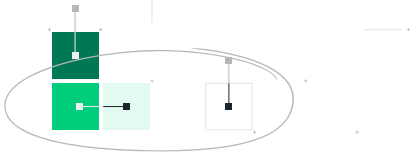
Java長期サポートバージョンの年度別導入状況

人気が高まるJDKベンダー：Eclipse Adoptium

次に、最も人気のあるJDKベンダーを見てみましょう。

かつてJavaはクローズドソースまたはプロプライエタリであったため、開発者はSun MicrosystemsからJDKを直接ダウンロードすることしかできませんでした。しかし、Sun MicrosystemsがJavaの最初のバージョンをリリースしてから約10年後、Oracleが最初のオープンソースのJavaバージョンをリリースしました。OpenJDKはこれを維持し、開発者コミュニティや、MicrosoftやAmazonなどのベンダーが他のバージョンのJDKを維持できるようにしています。

2020年には、Oracleが最も人気のあるJDKベンダーで、Java市場のおよそ75%を占めていました。JDK 11ディストリビューションのライセンスがより制限されるようになり（Java 17でよりオープンなスタンスに戻る前）、Oracleのバイナリから離れる顕著な動きがありました。しかし、それ以来、毎年（前年比）着実に減少しています。Oracleは首位を維持しているものの、2022年にはトップの座（34%）でしたが、2023年には29%に下がり、現在では21%に落ち込んでいます。一つまり、1年間で28%減少しました。

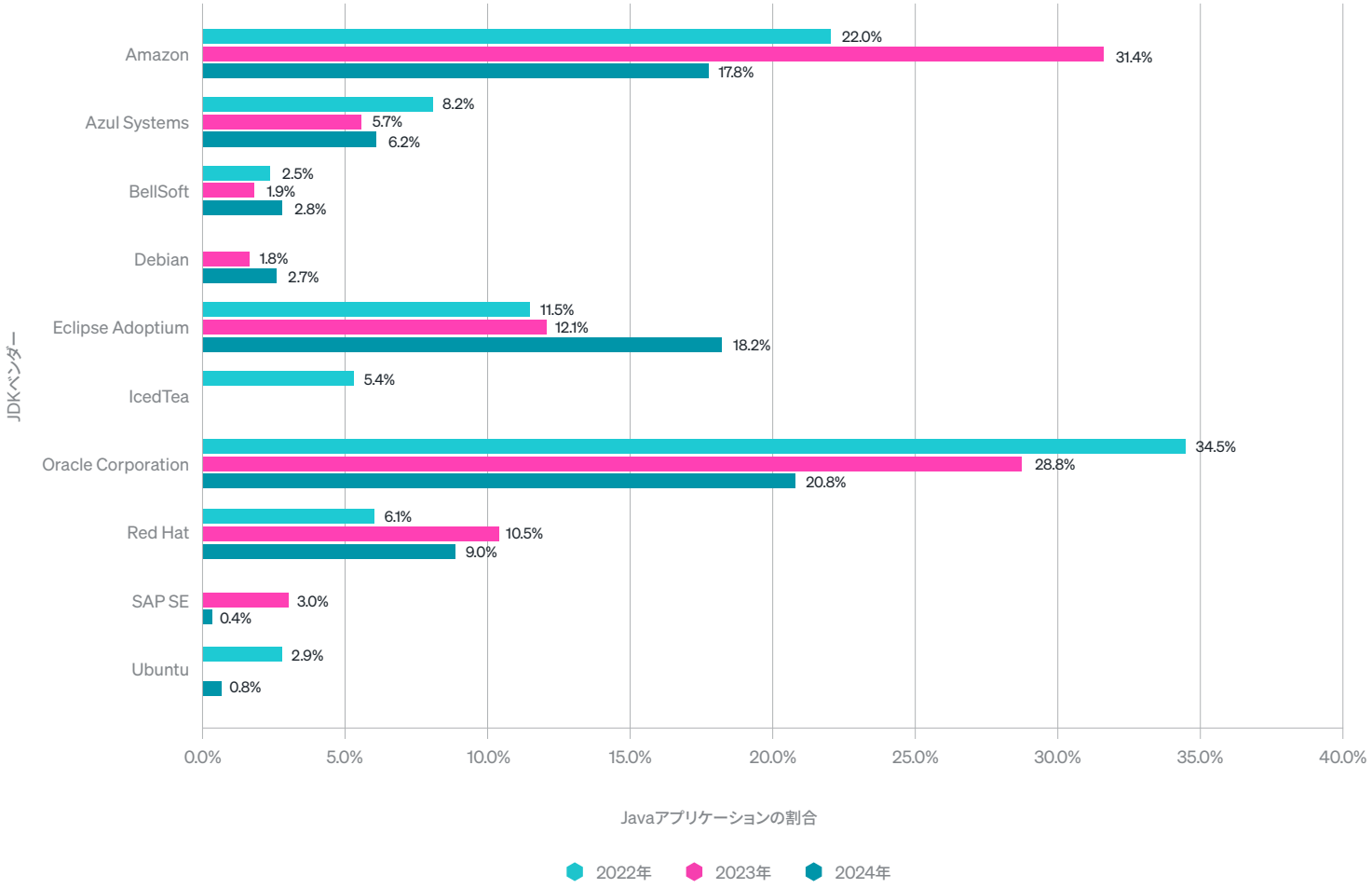


Amazonの利用は、2023年には市場の31%まで増加しました（2020年の2.2%、2022年の22%から増加）が、2024年には18%に低下しました。これは前年比43%の減少となります。

今年人気が高まっているのはEclipse Adoptiumで、その導入率は前年比12%から18%に、50%増加しました。Eclipse Adoptiumはコミュニティが管理しているため、このJDKはOracleやAmazon JDKよりも頻繁に更新される傾向にあります。

18%

Eclipse Adoptium JDKを使用している



年別の最も人気のあるJDKベンダー

最も一般的なJavaアプリケーション設定

次に、Javaアプリケーションでのガベージコレクター、CPUとメモリの使用について見ていきます

43%

G1ガベージコレクターを使用している

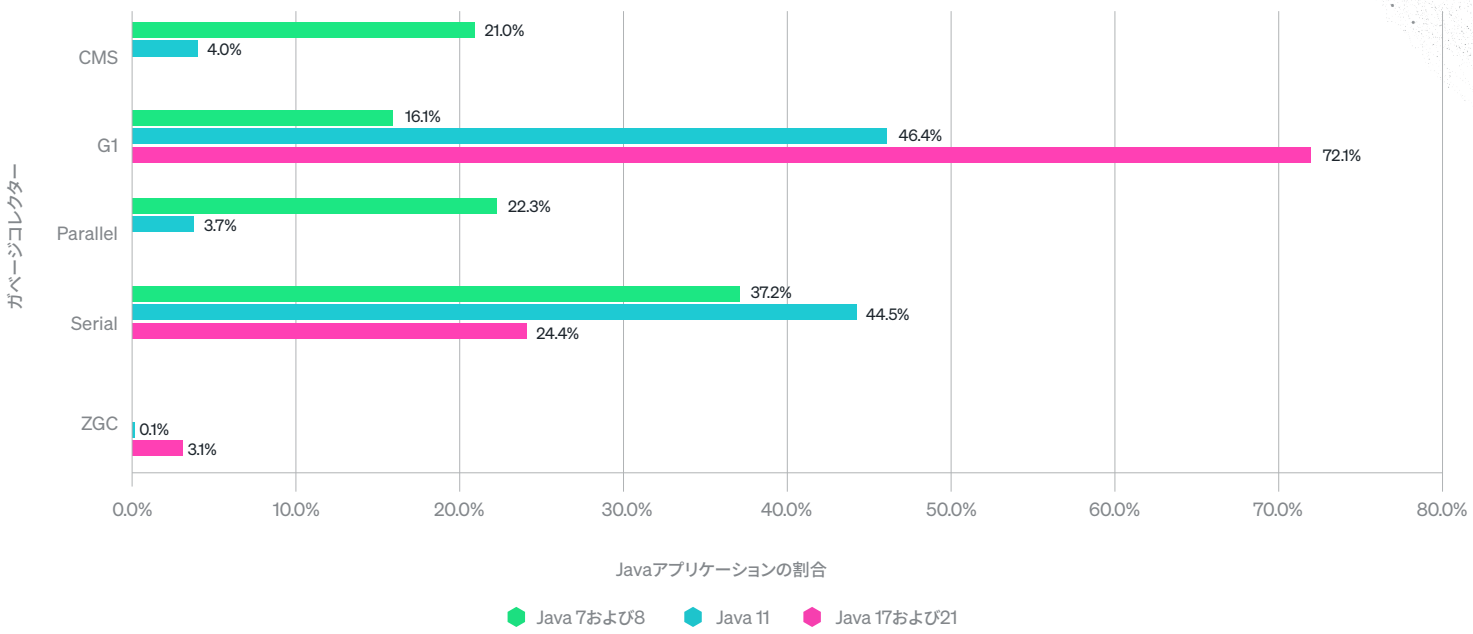
ガベージイン・ガベージアウト

Javaガベージコレクター（GC）はメモリ管理コンポーネントです。メモリリークを防止し、メモリ使用量を最適化し、Javaアプリケーションの全体的なパフォーマンスと安定性を確保するために使用します。

一部のJavaバージョンは特定のGCのみをサポートするため、開発者が使用するGCは部分的にJavaバージョンに依存します。

Java 11以降、Garbage-First（G1）GCがデフォルトになっています。デフォルトのコレクターであるということが、顧客の43%がこれを使用している要因であり、そしてJava 7や8と比較してJava 11、17、21における使用が大幅に増加した要因でもあります。さらに、G1の主な利点の1つは、大きな領域を一度にクリアするのではなく、小さな領域をクリアすることで、これによって収集プロセスが最適化されます。また、GCの実行がフリーズすることはめったになく、若い世代と古い世代の両方を同時に収集できるため、開発者にとって優れたデフォルト設定となっています。

2番目に普及しているGCはシリアル（37%）で、単一プロセッサ上で実行されるアプリケーションやシステム、または同じマシン上で多数のJava仮想マシン（JVM）を実行している場合に最適です。また、より複雑なGCに比べてCPUとメモリのオーバーヘッドが低いため、リソースに制約のある環境に適しています。



Java LTSバージョンで使用されるガベージコレクター

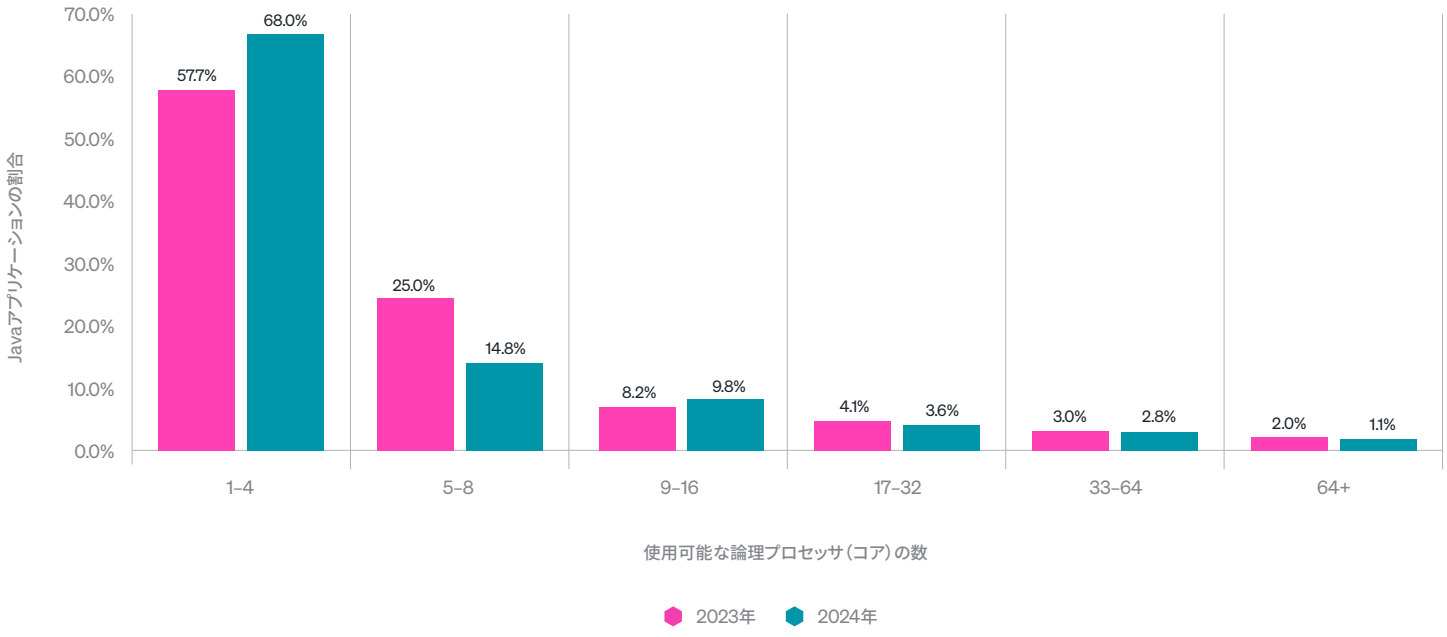
CPUとメモリの設定

68%

1～4コアを使用しています

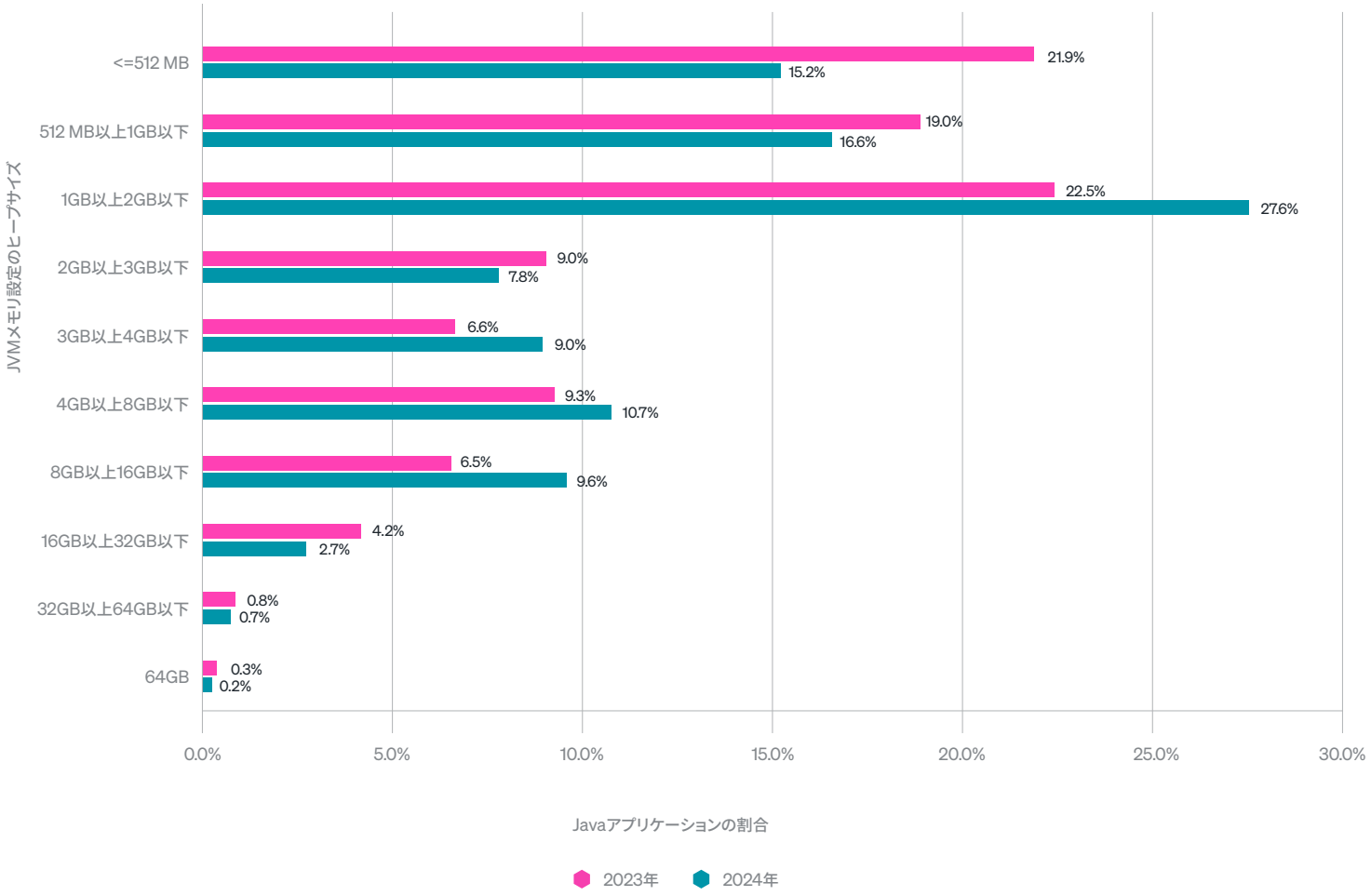
New Relicのデータによると、4コア以下で実行されているアプリケーションは前年比18%増加しています。

コンテナをデプロイすることが多いクラウド環境では、より小さいサイズで実行させることは非常に理にかなっています。しかし、この傾向は、アプリケーションによっては思わぬ問題を引き起こす可能性があります。特に、最近のJVMデフォルトのG1 GCによる同時実行のメリットの多くは、2コア未満で実行すると消滅してしまいます。これらのシングルコアのインスタンスはすべて、シリアルコレクターを使用しており、そのパフォーマンスコストが発生している可能性があります。



2023年と2024年にJavaアプリケーションで使用できる論理プロセッサ（コア）

JVMメモリ設定を見ると、Javaアプリケーションの32%が1GB 以下を使用し、68%が1GB 以上を使用しています。これは、1GBを超えるメモリを使用するアプリケーションが前年比で15%増加したことになります。



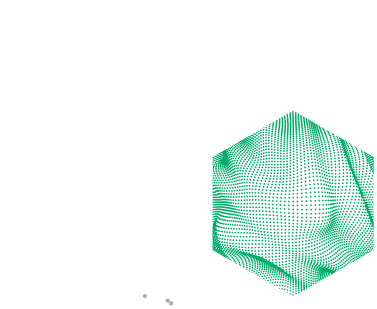
年別のJVMメモリ設定のヒープサイズ

ロギング、暗号化、データベースに関して一般的なJavaフレームワークとライブラリ

すべてのJava開発者は、ウェブ開発やデータベース接続などの一般的なタスクを実現するために、フレームワークとライブラリを使用することでアプリケーション開発プロセスを効率化します。ライブラリは、アプリケーションの機能を強化するために使用されます。ここでは、ロギング、暗号化、データベース接続で最も一般的なフレームワークとライブラリを確認しましょう。

最も普及しているロギングフレームワーク：Log4j

テスト環境や運用環境では、どのアプリケーションやシステムでも、バグや問題が発生する可能性があり、開発者はログツールを使用して問題のトラブルシューティングや修正を行います。ただし、ログの記録はアプリケーションのパフォーマンスに悪影響を与えることなく、ログメッセージから必要な情報を提供する場合にのみ役立ちます。ソフトウェア開発者は、これらの問題を解決するためにさまざまなロギングフレームワークを使用します。実際、New RelicにレポートするJavaアプリケーションの91%がロギングフレームワークを使用しています。

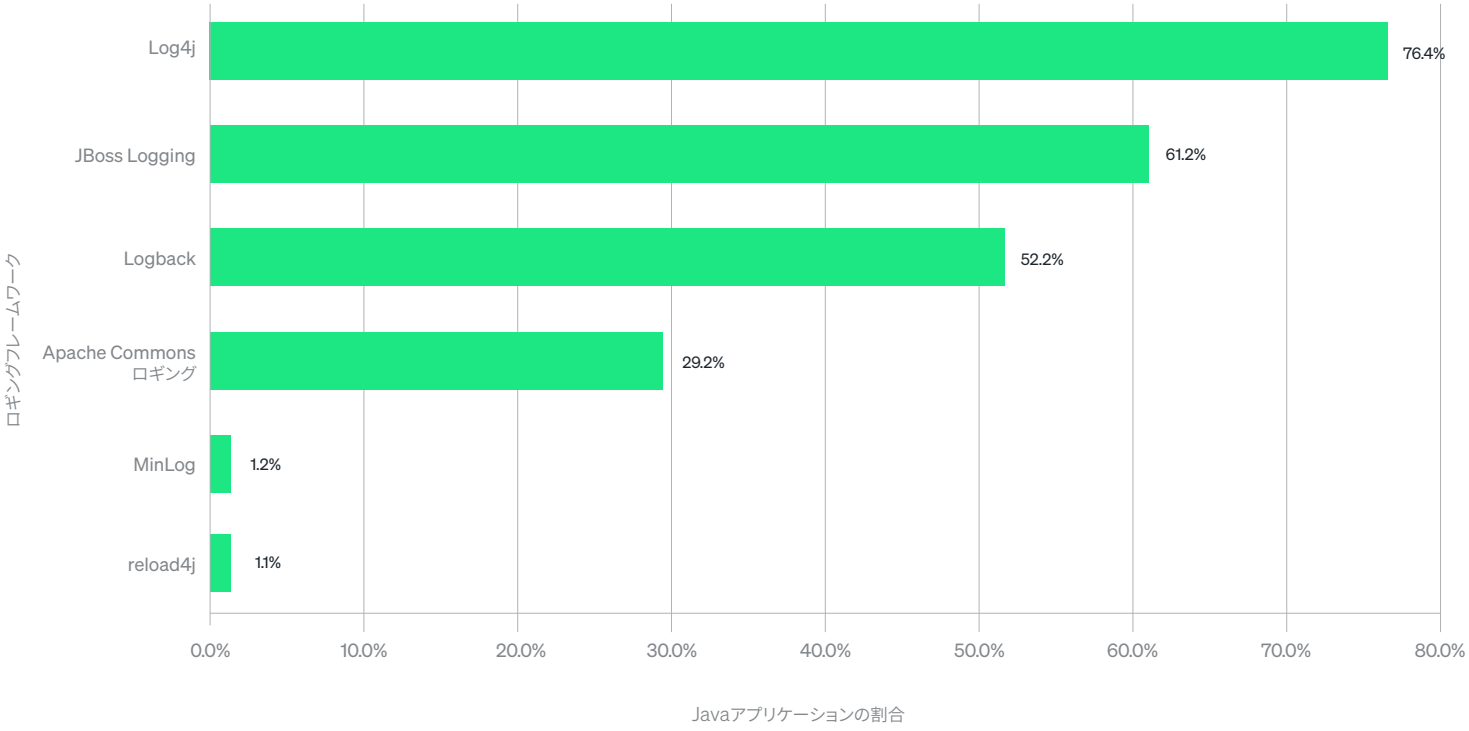


最も使用されているロギングフレームワークはLog4jで、Javaアプリケーションの76%がLog4jを使用しており、次にJBoss Logging (61%)、Logback (52%) が続きます。

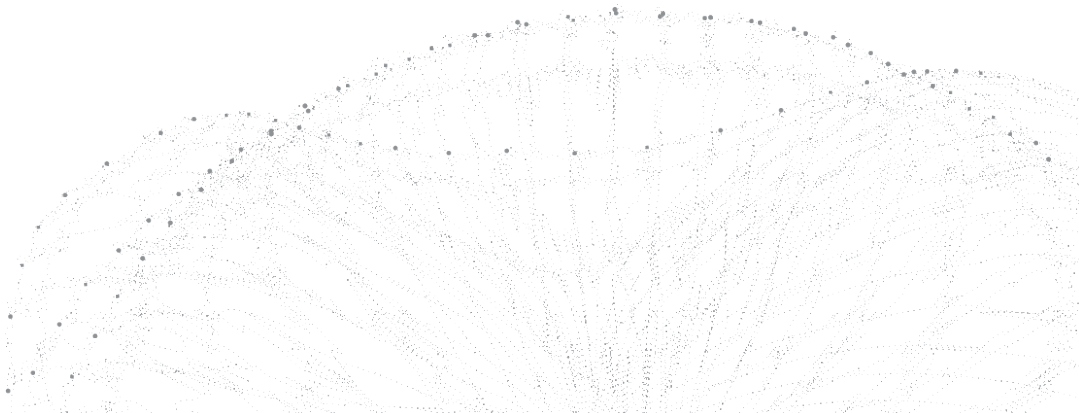
また、ほとんど (83%) のJava開発者は、他のタイプのJavaロギングフレームワークの抽象化として機能するフレームワークであるSLF4jに依存しています。SLF4jを使用すると、ソフトウェア開発者は任意のロギングフレームワークを使用できるようになり、アプリケーションは実装に影響を与えたり変更を加えたりすることなく、任意のJavaロギングフレームワークに互換的に切り替えることができます。この機能により、SLF4jはアプリケーションをロギングフレームワークから独立させ、システムのあらゆる部分にわたりロギングの柔軟性と移植性を高めます。つまり、Javaアプリケーションが複数のロギングフレームワークを使用できるということです。

76%

Log4jロギングフレームワークを使用しています



Javaアプリケーションで最も普及しているロギングフレームワーク



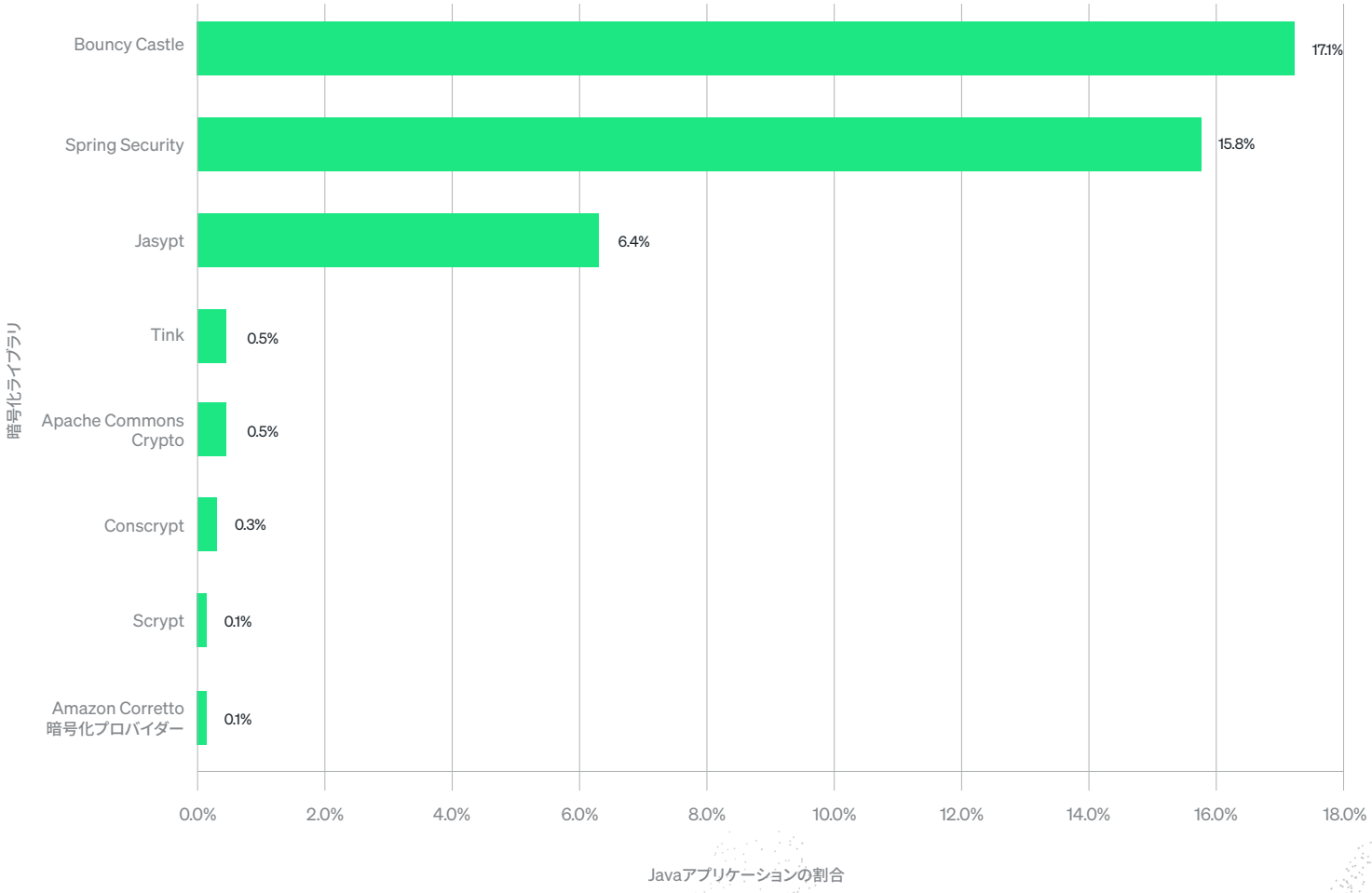
最も普及している暗号化用ライブラリ：Bouncy Castle

17%

Bouncy Castle暗号化ライブラリを使用しています

New RelicにレポートするJavaアプリケーションの3分の1以上 (41%) が暗号化ライブラリを使用しています。その割合は、Bouncy Castleが17%、Spring Securityが16%、Jasyptが6%です。Bouncy Castleは、暗号スイートとユーティリティのセットにより、長年にわたって最も普及している暗号化用ライブラリです。

Amazon Corretto Crypto Provider (ACCP) ライブラリを使用している開発者はわずか0.09%ですが、一般にベンダーの統合によりパフォーマンスが向上するため、企業や開発者がベンダー統合を検討しており、近い将来さらに多くのアプリケーションがこれを使用することが予想されます。



Javaアプリケーションで最も普及している暗号化ライブラリ

最も普及しているデータベースシステム：Oracle

17%

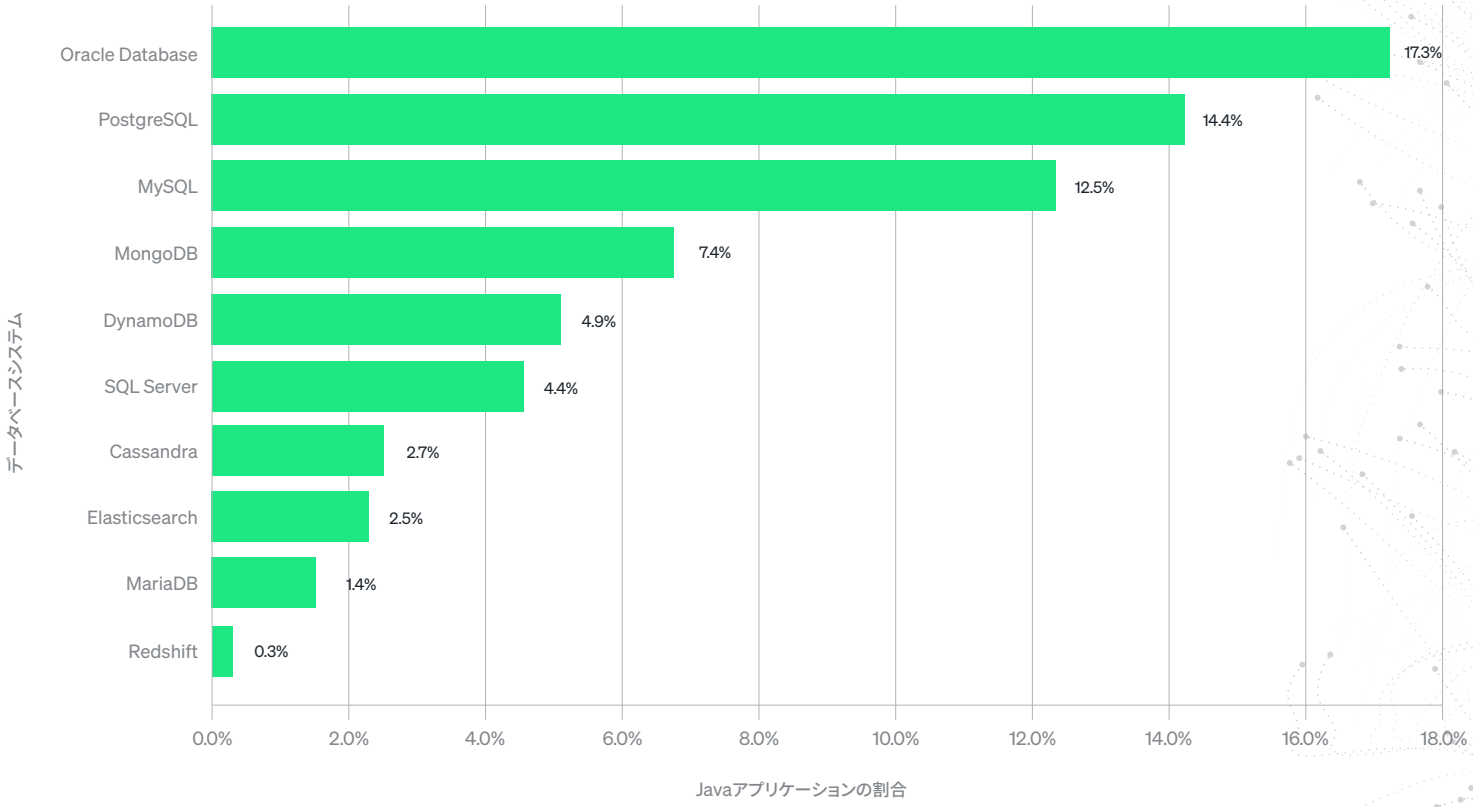
Oracle Databaseを使用しています

データベースに注目すると、Oracle Databaseが最も広く使用されており、New Relicにレポートするアプリケーションの17%がOracleデータベースを使用しています。Oracle Databaseは、その拡張性と、大量のデータを迅速かつ効率的に管理できることで知られています。そのため、企業が好むデータベースシステムです。さらに、カスタマーサポートと堅牢なツールセットも提供します。

2番目に人気のあるデータベースシステムはPostgreSQLで、New Relicにレポートするアプリケーションの14%がPostgreSQLを使用しています。Oracle DatabaseはOracleが直接管理し、ライセンスを通じて利用できますが、PostgreSQLは無料で使用できるオープンソースデータベースであり、読み取り/書き込み操作や複雑なクエリの管理に適しています。

3位はMySQLで、アプリケーションの13%がMySQLを使用しています。MySQLもオープンソースデータベースです。Oracle DatabaseやPostgreSQLよりも提供される機能が少ないため、特に読み取り専用クエリを処理する場合に、処理がより安定し、高速になります。

PostgresSQLはSQL仕様により準拠しており、MySQLよりも多くの機能をサポートしているため、昨年注目度が高まりました。MySQLは依然として上位のデータベースシステムですが、PostgresSQLの人気が高まっている一方で、その人気が低下していることがわかります。



Javaアプリケーションで最も普及しているデータベースシステム

開発者から寄せられるJava関連の最もよくある質問と要望

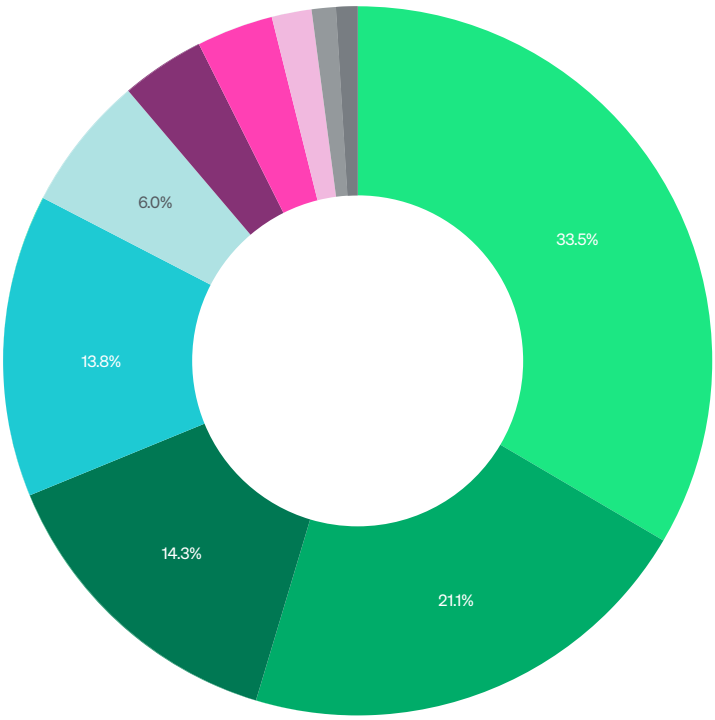
34%

Java関連の質問やリクエストは、どうすればよいかという方法について

また、開発者がNew Relic経由で、どのような種類のJava関連の質問や要望を寄せているかについても調査しました。2024年1月以降のデータによると、Java関連の483件のうち34%が方法（学習）についての質問で、21%は特定のメトリクスのクエリ関連、14%は設定に関するもの、14%はトラブルシューティングに関するものでした。

質問と要望の例は次のとおりです。

- Java Spring BootでAPIコールをログに記録するにはどうすればよいか？
- JavaフレームワークがTomcatからJBossに変更された場合、エージェントを再設定する必要があるか？
- コンテナのメモリ使用率とJVMヒープ使用率のダッシュボードを作成するには？
- Vaadinフレームワークを使用するJavaアプリケーションのメモリ消費量が多い原因は？
- JVMでは、G1 Eden Spaceのヒープ使用量は何を意味するか？



New Relic AIを使用する開発者から寄せられるJava関連の最もよくある質問とリクエストの種類

調査方法

本レポートは、パフォーマンス情報を提供するNew Relicにレポートする数十万のアプリケーションから収集されたデータに基づいています。したがって、Javaの使用状況の全体像を提供するものではありません。すべてのデータは2024年に収集されました。

New Relicは、適切なデータを匿名化し、意図的に粗視化することで、Javaエコシステムの一般的な概要を提供します。攻撃者やその他の悪意のある者の役に立つような詳細情報は、意図的に本レポートから除外されています。

さっそくNew RelicでJavaデータのモニターを始めましょう。

Java Quickstartをインストール

New Relicについて

New Relicは、オブザーバビリティのリーダーとして、優れたソフトウェアの計画、構築、導入、運用に対するデータドリブンなアプローチによりエンジニアを支援しています。New Relicは、メトリクス、イベント、ログ、トレースからなる全テレメトリーが集約された唯一の統合データプラットフォームを、強力なフルスタック分析ツールと組み合わせて提供し、意見ではなくデータにもとづくエンジニアのベストパフォーマンスを可能にします。

New Relicは、直感的で予測可能な業界初の使用量ベースの価格体系によって提供され、計画サイクルタイム、変更失敗率、リリース頻度、平均解決時間（MTTR）の改善を支援することにより、エンジニアに高い費用対効果をもたらします。これにより、世界をリードする大企業や成長著しいスタートアップ企業のアップタイムと信頼性、運用効率の向上を助け、イノベーションと成長を加速させる優れたカスタマーエクスペリエンスの創出を支援します。

new relic.