# Web ResponseTime エージェントのスケーラビリティを予測するためのベストプラクティス

**ホワイトペーパー**
2005 年 5 月

## 目次


目次..... 1

Web ResponseTime エージェントのスケーラビリティを予測するためのベストプラクティス.. 1

World Wide Web..... 1

URLCheck ガイドライン..... 1

スケーラビリティに関する問題点 ..... 2

Web トランザクションガイドライン..... 7

実行時間の計画..... 7

結論..... 12



Web サイトと Web ベースアプリケーションの高度な多様性は、その他の要因と相まって、単一の AppManager エージェント上で実行できる Web-RT Knowledge Script ジョブ数の判断を非常に困難にしています。この文書では、判明している制限と、単一エージェント上で実行可能なジョブ数を予測するためのガイドラインを示します。この文書は特に AppManager ResponseTime for Web バージョン 6.2 について記載しています。

## ご注意（重要）

本ドキュメントおよび本ドキュメントに記載されているソフトウェアは、「ライセンス契約」または「機密保持契約」に基づいて提供されるものであり、「ライセンス契約」または「機密保持契約」の条項に服するものです。「ライセンス（使用許諾）契約」または「機密保持契約」で別途明示的に規定されている場合を除き、ネットアイキュー株式会社（以下「当社」といいます。）および名国**NetIQ Corporation**は、本ドキュメントおよび本ドキュメントに記載されているソフトウェアをいかなる種類の保証もなしに、「現状有姿**(as is)**」の状態でお客様に提供するものであり、本ソフトウェアに関して、明示および黙示を問わず、商品適合性又は特定目的に対する適合性に関する黙示の保証を含む、いかなる保証を行うものではありません。

本ドキュメントおよび本ドキュメントに記載されているソフトウェアは、当社による事前の書面承諾なしに、賃貸、売買または贈与してはならないものとします。ただし、法により別途許諾される場合はのぞきます。「ライセンス（使用許諾）契約」または「機密保持契約」で別途規定されない限り、本ドキュメントおよび本ドキュメントに記載されているソフトウェアのいかなる部分についても、当社による事前の書面承諾なしに、複製したり、情報検索方式に保存したり、また、その形式および方法（電子的方法、機械的な方法、その他の方法）を問わず、送信伝達してはならないものとします。本ドキュメントに含まれる会社名、個人名およびデータの中には、説明のために使用されているものもあり、必ずしも実際の会社、個人またはデータを表しているわけではありません。

本ドキュメントには、技術的な不正確性、誤字が含まれている可能性があります。本ドキュメントに含まれている情報には、定期的に変更が加えられます。これらの変更は、本ドキュメントの新版に組み込まれる可能性があります。当社は、いつでも、本ドキュメントに記載されているソフトウェアにつき、改良、変更を加えることができるものとします。

**© 1995-2005 NetIQ Corporation, all rights reserved.**

Check Point、FireWall-1、Provider-1、SiteManager-1、およびVPN-1は、Check Point Software Technologies Ltd.の商標または登録商標です。

ActiveAgent、ActiveAnalytics、ActiveAudit、ActiveKnowledge、ActiveReporting、ADcheck、AppAnalyzer、 Application Scanner 、 AppManager 、 AuditTrack 、 Chariot 、 ClusterTrends 、 CommerceTrends 、 Configuration Assessor 、 ConfigurationManager、the cube logo design、 DBTrends、DiagnosticManager、Directory and Resource Administrator、Directory Security Administrator、Domain Migration Administrator、End2End、Exchange Administrator、Exchange Migrator、Extended Management Pack、FastTrends、File Security Administrator、Firewall Appliance Analyzer、Firewall Reporting Center、Firewall Suite、Ganymede、the Ganymede logo、Ganymede Software、Group Policy Administrator、imMarshal、Intergreat、Knowledge Scripts、MailMarshal、Marshal、Migrate.Monitor.Manage、Mission Critical Software、Mission Critical Software for E-Business、the Mission Critical Software logo、MP3check、NetIQ、the NetIQ logo、the NetIQ Partner Network design 、 NetWare Migrator 、 OnePoint 、 the OnePoint logo, Operations Manager 、 PentaSafe 、 PSAudit 、 PSDetect 、 PSPasswordManager、PSSecure、 Qcheck、RecoveryManager、Security Analyzer、Security Manager、Security Reporting Center、 Server Consolidator、SQLcheck、VigilEnt、Visitor Mean Business、Vivinet、W logo、WebMarshal、WebTrends、WebTrends Analysis Suite、WebTrends for Content Management Systems、WebTrends Intelligence Suite、WebTrends Live、WebTrends Log Analyzer、WebTrends Network、WebTrends OLAP Manager、WebTrends Report Designer、WebTrends Reporting Center、WebTrends Warehouse、Work Smarter、WWWorld、およびXMPは、米国NetIQ Corporationまたは当社を含むNetIQ Corporationの関連会社（アメリカ合衆国内の関連会社であるか否かを問いません。）の米国およびその他の国における商標または登録商標です。

言及されている他の会社名および商品名は、全て、識別目的のためにのみ使用されているもので、他社の商標または登録商標である可能性があります。

# World Wide Web

World Wide Web は素晴らしい、有機的な発明です。多くの組織や人々の創造力と工夫によってある環境が生まれ、その環境では選択肢となる数々の標準、導入できる多くの製品、そして、これらの標準を緩やかに順守する幾多の製品が存在します。

注目すべきことに、多くのユーザーは、過剰な広告用ポップアップなど余計な邪魔物に悩まされることなく、必要なものを閲覧できます。ただし、AppManager ResponseTime for Web をサポートする Web-RT 管理オブジェクトなどのロボットにとって、Web にはさまざまな難題が存在します。

この環境の相違から、NetIQ は Web サイトと Web ベースアプリケーションのレスポンスタイムを監視するために全く異なる 2 つの方法を使用します。1 つはシンプルな URL Check Recorder です。基本的なアベイラビリティとシンプルな個別 Web ページのパフォーマンスを対象としています。もう 1 つは Web Recorder で、Web ベースアプリケーションに見られるような複雑なシーケンスを扱います。それぞれのスケーラビリティとパフォーマンスの問題点は明確に異なります。この文書ではこの両方について説明し、これらの問題点を解明するための手掛かりを提供します。Web および Web ベースソフトウェアは本質的に予測不可能であり、常に単純で固定的な制限を提供することは困難です。その代わり、この文書はユーザーが自身の管理要件に関する制限を予測することができるように制限の本質を理解する手助けとなります。

# URLCheck ガイドライン

最初のセクションでは、AppManager ResponseTime for Web のアドイン URL Check Recorder に関するパフォーマンスの制約に焦点を置きます。URL Check Recorder バージョン 2.2 の機能と制限を説明します。2005 年 5 月のリリースでは、わずかな変更が加えられました（6.2 とバージョン付けられました）が、この情報は実質的に URL Check Recorder の前のバージョンから変更されていません。

## テクノロジとアーキテクチャ

URL Check Recorder は、チェックされる一連の URL から構成される URLCheck Knowledge Script を作成します。最も簡易なフォームでは、(URL ごとに)実行される操作は Web サーバに接続し、HTTP 戻りコードを取得するだけです。

### WinINET の制限

WinHTTP がコンピュータ上に存在する場合は、URL Check Recorder は WinHTTP インターフェイスを使用して再生シーケンスを駆動し、コンピュータ上に存在しない場合は、堅牢さが劣る古い WinINET インターフェイスを使用することになります。

WinHTTP インターフェイスは Windows Server 2003 では標準ですが、Windows NT では全くサポートされていません。WinHTTP をサポートするには、Windows 2000 では SP3 を Windows XP では SP1 を適用する必要があります。

URL Check Recorder が WinINET に対して実行されている場合は、大幅に速度が遅くなります。そのため、URL Check Recorder から最高のパフォーマンスを得るための最初のガイドラインとなるのは、必ず、WinHTTP をサポートするシステムに URL Check Recorder をインストールするということです。私たちのベンチマークはすべて、WinHTTP を使用して実行されています。

## 個々の URL ごとに可能な操作

URLCheck Knowledge Script では、次の操作を任意に組み合わせて試みることができます。

1. URL への接続と HTTP コードに基づく成功または失敗のレポート。

2. ページの全コンポーネント(HTML ファイル、GIF など)のダウンロードとダウンロード時間のレポート。この操作では、ページのレンダリング時間も要求します。
3. テキスト文字列の検索、特定のオブジェクトとリンクの存在チェック、さらには、これらのリンクが有効なページをポイントしているかどうかのチェック。これらの付加的な任意の操作はスクリプトステップ上に積み重なり、Knowledge Script の 1 回の実行間隔を完了するために要する合計時間が大幅に増加します。

## URLCheck Knowledge Script のサイズ

URL Check Recorder は、最大で 999 の URL から構成される Knowledge Script を作成することができます。ただし、この制限に近づくことは好ましくありません。非常に長いスクリプトではジョブの実行に時間が掛かり、リソースがロックされ、他のプロセスの実行が完全にブロックされることがあります。目安として、100 を超える URL ステップを持つ URLCheck スクリプトを実行することはお勧めしません。

## 警告とデータに関する考慮事項

複数の警告をともなう長いスクリプトでは、AppManager ResponseTime for Web の旧バージョンで作成された URLCheck Knowledge Script は、ジョブの実行の最後に最高レベルの警告のみを発行しました。デフォルトでは、「アベイラビリティのパーセント」データポイントが常に生成されたので、それが唯一の警告になる可能性がありました。他のすべてのデータ収集は、そのデータに関連する警告と同様に、デフォルトで無効とされました(現在も無効です)。警告には通常はジョブの各ステップに関する詳細が含まれていますが、スケーラビリティを計画する上で、単一の警告のみが生成されたことに留意する必要があります。

URL ステップごとに 1 つのデータ系列というように追加のデータ系列を生成すると、ベースの AppManager エージェント(MC)の負荷が明らかに増加します。

バージョン 6.2 での URLCheck の主な変更は、スクリプト内の各 URL ごとに 1 つのイベントを生成できるようになったことです。このオプションを選択し、必要なイベントの状態が真になると、(返される任意のデータ系列に加えて)それぞれの個別イベントが生成されるため、ジョブの実行の最後に負荷が顕著に変化する可能性があります。

# スケーラビリティに関する問題点

このセクションでは、スケーラビリティに関連する既知の問題について説明し、ベンチマークデータを示してそれについて説明します。ただし、スケーラビリティは単純なオン/オフ機能ではないことを理解する必要があります。どのような実装でもスケーラビリティの限界に達すると、さまざまな機能低下が発生します。たとえば、ベースエージェントがデータを AppManager リポジトリに送り返す場合、それがリソースに対する URLCheck プロセスと競合すると、データの損失やレスポンスのスローダウンが発生する可能性があります。

WinHTTP インターフェイスが可能なシステムの使用に加え、実績とテストにもとづくガイドラインを以下に示します。

## スクリプトの簡潔さ

簡潔なスクリプトは複雑なスクリプトよりも高速に実行されます。集計データをレポートするスクリプトは、個々の URL データごとにレポートするスクリプトよりも高速に実行されます。

たとえば、25 の異なる Web サイト(25 の URL)のアベイラビリティのみをチェックし、単一のアベイラビリティデータポイントを返す単純なテスト URLCheck Knowledge Script は、約 12 秒で完了しました。個々のアベイラビリティデータポイント(各 URL に 1 つ)を Knowledge Script に追加しても、認識できるほど遅くはなりません。ただし、ダウンロード時間の測定を追加すると、スクリプトの実行時間が約 12 秒から 45 秒へと増加します。

# ベンチマークの理解

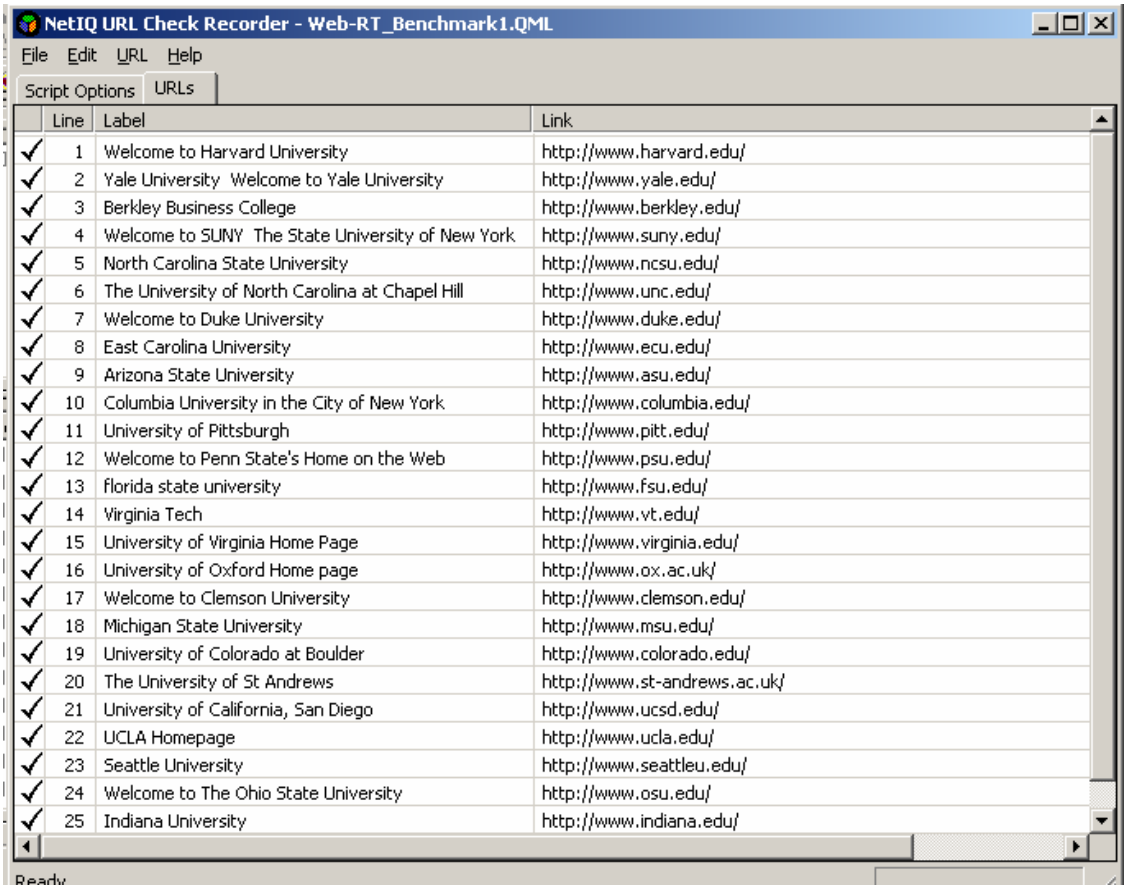
2004 年半ば、スケーラビリティの限界を調べるため、URL Check Recorder で一連のベンチマークテストを実行しました。もちろん利点はさまざまです。実装計画を実施する前に、必ず多数のジョブを実行してスケーラビリティテストを行ってください。

## ベンチマークの設定

25 のパブリック URL(以下の図に表示)をチェックする簡単なスクリプトを使用しました。

全テストをシングルプロセッサ 1.2GHz CPU、512MB RAM で実行しました。テストは、SP3 が適用された Windows 2000(WinHTTP をサポート)が稼動するコンピュータ上で AppManager 5.0.1 と AppManager ResponseTime for Web バージョン 2.2(AppManager 6.0 と共に出荷)を使用して実行されました。

次の画像は、この Knowledge Script の URLs タブが URL Check Recorder でどのように表示されるかを示したものです。




## ベンチマークスクリプト

同じ URL セットを使用して、4 つの異なるバージョンの Knowledge Script を構築しました。

1. 全体的アベイラビリティ

   この Knowledge Script によって、全体の URL アベイラビリティのパーセントに対する単一データポイントが生成されました。

2. 個別データのある全体のアベイラビリティ

この Knowledge Script によって単純な URL 確認が実行されましたが、全体のアベイラビリティに加えて個々の URL アベイラビリティ(合計 26 データポイント)が生成されました。

3. 全体のダウンロード

   このスクリプトによって、リンクチェックとダウンロード時間が測定されました。2 つのデータポイントが生成されました。1 つはアベイラビリティのパーセント、もう 1 つは全体のダウンロード時間です。

4. 個別データのある全体のダウンロード

   この Knowledge Script によって、リンクチェックとダウンロード時間の測定が実行されました。全体のダウンロード時間、全体のアベイラビリティパーセントに加えて、個々の URL ダウンロード時間の合計 27 データ系列が生成されました。

アベイラビリティベンチマーク Knowledge Script にはおよそ 12 秒を要し、ダウンロードベンチマークには通常約 45 秒を要しました。追加データポイントのレポートでは、この時間は根本的には変わりませんでしたが、ベースエージェントの負荷はいくらか異なりました。

## ベンチマークテストのパターン

上述のオプションを使用して、さまざまなバージョンの URLCheck Knowledge Script を実行しました。同じ Knowledge Script を使用して、同時に複数のジョブの実行を開始しました。過剰負荷や不安定度の限界が見つかるまで、同時実行のジョブ数を変更して、すべてのジョブを 1 分間隔で実行しました。

## ベンチマークマトリクス

メモリ使用量と CPU 負荷を確認しました。実際には、URLCheck が高負荷の場合、エージェントコンピュータは CPU **の制限**でありメモリの制限ではないことがわかりました。2 秒間隔でさまざまな Perfmon カウンタを監視しました。コンピュータの全 CPU 負荷、シングル `WEBEX21SM` プロセス(URLCheck ジョブを実行しているプロセスに相当)の CPU(およびメモリ)使用量、および AppManager ベースエージェントプロセス `NetIQMC` の CPU(およびメモリ)使用量を計測しました。メモリの空き容量は十分に利用されていましたが、メモリデータは、メモリが足りているということしか示さず、ここでは再生されません。

## アベイラビリティベンチマークの結果

最初に、簡単なリンクチェックのみを実行したアベイラビリティベンチマーク Knowledge Script の結果を確認します。次の頻度で実行され、次の CPU 負荷結果が得られました。

| 全体のアベイラビリティ | 全体の CPU | WEBEX21SM CPU | NetIQMC CPU |
|---|---|---|---|
| 1 分ごとに 10 ジョブ | 19% | 1.46% | 0.46% |
| 1 分ごとに 30 ジョブ | 27.8% | 1.19% | 1.58% |
| 1 分ごとに 50 ジョブ | 28.5% | 1.4% | 1.52% |
| 1 分ごとに 100 ジョブ | 52.8% | 1.57% | 4.11% |

| アベイラビリティ+データ | 全体の CPU | WEBEX21SM CPU | NetIQMC CPU |
|---|---|---|---|
| 1 分ごとに 10 ジョブ | 21% | 1.07% | 4.3% |
| 1 分ごとに 30 ジョブ | 38.6% | 1.05% | 11.63% |
| 1 分ごとに 50 ジョブ | 50% | 1.4% | 18.6% |
| 1 分ごとに 100 ジョブ | -- | -- | -- |

これらの違いは、アベイラビリティ + データベンチマークが、ベースアベイラビリティベンチマークと比較して、26 のデータ系列を送り返すことです。これは、リポジトリ/管理サーバコンピュータへのデータ転送を実行したベースエージェントプロセス NetIQMC の負荷が増える原因となります。

アベイラビリティ + データベンチマークは、100 ジョブ/分の負荷で十分に動作できませんでした。CPU 使用量は実質的に 100%に達し、他のシステムが不安定になりました。私たちのテストベッドでは、実際の限度は 50 から 100 ジョブ/分の負荷内にあります。結果として、そのようなコンピュータ上の URLCheck は、ざっと見て 1 分ごとに 1250 より多い URL へのリンクチェックを実行することができます。

もちろん、それぞれに 625 の URL を持つ 2 つのジョブは、完了するのに非常に長時間を要する可能性があり、1 分ごとに実行できない可能性があります。WebTransaction セクションで説明した"タイムバジェット"に関する考慮事項(以下参照)は、最終的にはここでも適用されます。これらのジョブは完了するまでにおよそ 45 秒を要するため、1 分ごとに 1 度試みることができました。

OS、ハードウェアやアプリケーションなど、URLCheck 外の別の監視については、このガイドラインでは触れていません。このエージェントコンピュータは、この負荷で URLCheck ジョブを実行するためだけに動作します。

## ダウンロードベンチマークの結果

ダウンロードベンチマークスクリプトはどちらも、単純なリンクチェックと完全なダウンロード測定を実行しました。次の頻度で実行され、次の CPU 負荷結果が得られました。

| ダウンロード | 全体の CPU | WEBEX21SM CPU | NetIQMC CPU |
|---|---|---|---|
| 1 分ごとに 10 ジョブ | 67.6% | 4.69% | 0.88% |
| 1 分ごとに 30 ジョブ | 91.3% | 2.51% | 1.58% |
| 1 分ごとに 50 ジョブ | 96.59% | 2.07% | 2.49% |
| 2 分ごとに 50 ジョブ | 84.81% | 2.25% | 1.92% |

| ダウンロード+データ | 全体の CPU | WEBEX21SM CPU | NetIQMC CPU |
|---|---|---|---|
| 1 分ごとに 10 ジョブ | 63.27% | 3.70% | 3.88% |
| 1 分ごとに 30 ジョブ | 93.63% | 2.19% | 6.50% |
| 1 分ごとに 50 ジョブ | 95.66% | 1.77% | 7.75% |
| 2 分ごとに 50 ジョブ | 91.15% | 1.96% | 6.89% |

1 分ごとに 100 ジョブという頻度の負荷では、ダウンロードベンチマークをうまく実行できませんでした。ジョブの実行でより消費される CPU によって、データを生成された速度で返すために必要なサイクルがベースエージェントから奪われます。アベイラビリティ+データベンチマーク(27 データ系列を返します)では、ベースエージェントを維持するために CPU 負荷のほぼ 20%を必要としたことを思い出してください。

そのため、ダウンロードの測定を実行するジョブについては、最大 1 分ごとに 50 ジョブ(それぞれ 25 URL)の制限をガイドラインとします。これによって、1 分ごとの URL は合計 1250 になります。実際のダウンロード操作は、1 ページにあるオブジェクトの数とサイズに依存します。このテストで使用した教育ホームページは「一般的なページ」と言えるかもしれませんが、この目的のために、本当に意味のある「通常」の定義がないことは明らかです。グラフィックが密集したページはこのジョブの実行時間を 50%増加し、タイムバジェットの制約によってサイクル時間を効果的に制限する 25 の異なる URL/ページを見つけることは、おそらく難しくありません。

OS、ハードウェアやアプリケーションなど、URLCheck 外の別の監視については、このガイドラインでは触れていません。このエージェントコンピュータは、この負荷で URLCheck ジョブを実行するためだけに動作しています。

# Web トランザクションガイドライン

AppManager ResponseTime for Web の Web Recorder アドインは、Web サイトまたは Web ベースのアプリケーションにおけるユーザーのブラウズ操作のパスを記録して、そのパスの AppManager Knowledge Script 版（WebTransaction Knowledge Script）を作成することにより、複雑なトランザクションスクリプトを作成します。このモジュールの最新バージョン（6.2）では、Web Recorder アドインの機能が大幅に改良され、さまざまな Web ベースアプリケーションと複雑な Web サイトの記録と再生を行えるようになりました。もちろん、再生や記録を難しくする特定の Web ページ設計や技術に関連した制限はまだ残っています。

しかし、本書の主題はパフォーマンスとスケーラビリティなので、記録した Web トランザクションを配備する前にその再生をテストしてあるものと想定しています。本書の前バージョン（2004 年 9 月版）で論じた概念は今でも有効ですが、製品がいくつかの点で変化したため、スケーラビリティの評価は少し異なります。

## 新旧スクリプト/エージェントの同時実行

新しいモジュール（バージョン 6.2）は、古い再生エンジン（バージョン 2.2）と、新機能をフルに備えた新しい再生エンジン（バージョン 6.2）を、両方ともインストールします。その目的は、記録された既存スクリプトの実行という面で製品のアップグレードを容易にすることだけです。言い換えると、新バージョンは古い（バージョン 2.2 の）スクリプトも実行できます。もちろん、古いスクリプトは新しい Web Recorder で記録し直さない限り新しい機能を使用できません。記録し直したスクリプトは、新しい再生エンジンで再生されることになります。

バージョン 2.2 の再生エンジンは、バージョン 2.1 のスクリプトを実行できません（記録し直さない限り）。現在バージョン 2.1 を使用している場合は、アップグレードの一環としてすべてのスクリプトを作成し直すことを考慮するべきでしょう。バージョン 6.2 の新機能と明確なアーキテクチャを考えれば、その努力をする価値は十分にあります。

NetIQ はベストプラクティスとして、古い Knowledge Script を使った WebTransaction ジョブと、新しく記録した Knowledge Script を実行するジョブを、単一のエージェント上で一緒に実行することをお勧めしません。両方のジョブを同じ環境内で実行する必要がある場合は、それぞれを別々のコンピュータで実行することを強くお勧めします。

スケーラビリティについては、古いスクリプトと新しいスクリプトを混ぜて実行する場合のスケーラビリティの限界は、**より古い**バージョンのものになるものと想定してください。

## 「有用性はさまざま」

負荷とスケーラビリティに関するあらゆる疑問に対する最良の解答（唯一の決定的な解答）は、適切に設計されたテストを実行することです。目的の実現可能性を検証するには、監視のために実行する必要のあるトランザクションの数とタイプをテストするしかありません。しかし、このセクションの残りの部分では、この機能のパフォーマンスとスケーラビリティの見積もりと理解に関するガイドラインを提示しています。

# 実行時間の計画

WebTransaction のスケーラビリティの計算において必要不可欠な要因は「実行時間の計画」です。この概念を理解して、使用するスクリプトとジョブについて実行時間の計画を計算する必要があります。

AppManager Suite の Knowledge Script の多くは、PerfMon カウンタの読み取りや JMX インタフェイスからの値の取得など、比較的単純な操作を実行します。このような操作のほとんどは、完了するために大きな計算能力やクロック時間を必要としません。たとえば、カウンタを読み取るジョブは実行開始から数ミリ秒で完了します。データポイントまたはイベントを返すためのオーバーヘッドを含めても、このジョブ間隔は数百ミリ秒以内に完了します。

一方、WebTransaction Knowledge Script ジョブは、これに比べると実行に非常に長い時間がかかります。通常観察される特定のマルチステップ Web トランザクションの自然な所要時間は（ブラウザを自分で使用する場合）、10～60 秒ですが、この時間は多くの要因によって決まります。これらの要因には、アプリケーションまたはサイトの設計自体、Web サイトにかかる負荷、ネットワークの接続性と帯域幅、バックエンドシステム（ページまたは操作にデータを提供するアプリケーション、データベース、コンテンツサーバなど）のパフォーマンスおよび負荷と応答性などが含まれます。

AppManager ResponseTime for Web はこのトランザクションを処理するときに、トランザクションの「自然な」所要時間に独自のオーバーヘッドを付け加えます（このオーバーヘッドについては以下で詳しく説明します）。たとえば、あるマルチステップトランザクションで、このオーバーヘッドが 30 秒であるとします。

ジョブ間隔が 90 秒（1 分半）の場合は当然、そのジョブを 1 時間に 40 回を超える頻度で実行することはできません。もちろんほかにも関係する要因はありますが、最も決定的な要因は結局、処理能力ではなく実行時間の計画です。

実行時間の計画を算出する助けとして、新バージョンではオプションのデータ系列としてスクリプトの総処理時間が提供されます。必要な時間をよく理解するために、スクリプト自体の実行にかかる時間を観察して報告することができます。

各スクリプト/ジョブの総処理時間を評価し、このデータを使って頻度と負荷を計算する必要があります。WebTransaction ジョブは実行に時間がかかるため、オーバーランを発生させるスケジュールを設定するのは非常に簡単です。このオーバーランは、ジョブが 1 つも実行されなくなるまで悪化し続けます。

## 再生のしくみ

WebTransaction Knowledge Script の再生では、実行される各ジョブごとに Internet Explorer のインスタンスがメモリにロードされます。これ自体、各ジョブ間隔ごとに大きなオーバーヘッドを生成するとともに、少なくとも各同時ジョブにつき 15MB のメモリを必要とします。したがって、可能な範囲内で最高のパフォーマンスを達成するために重要な最初のパフォーマンスパラメータは、十分なメモリです。

第 2 に、最新バージョンの Web Recorder では、同時に実行できるジョブの数が 3（旧バージョン）から 5 に増えました。この数はレジストリ値ですが、製品は 5 より大きい値を無視します。

従来どおり、5 個より多いジョブを同時に実行しようとすると、超えた分のジョブは必要なリソースが使用可能になるまでキュー内で待機することになります。キュータイムアウトが時間切れになるまでにジョブが実際に開始されなかった場合は、ジョブがタイムアウト前に Internet Explorer リソースを獲得できなかったため強制終了されたということを知らせるイベントが発生します。これを「キュータイムアウト」と呼びます。

短いスケジュール間隔で多すぎるジョブを実行しようとすると、キュータイムアウトやデータポイントの欠落が発生します。

以前のバージョンでは、1 つのタイムアウト値が 2 つの異なる目的に使用されていました。1 回はジョブが実行まで待機する時間（キュータイムアウト）を測定するために使用され、もう 1 回は実行開始から終了またはタイマの期限切れまでの時間（ジョブタイムアウト）を測定するために使用されていたのです。バージョン 6.2（およびそれ以降）の Web Recorder では、この 2 つのタイムアウトが別々に分けられました。ジョブがキュー内でタイムアウトまで待機する時間を指定し、それとは異なる値をジョブ全体の完了を制御する（ジョブの実行が開始された後で）タイムアウト値として指定できるようになったのです。

## ジョブのオーバーヘッド

AppManager ResponseTime for Web のバージョン 6.2 では、ジョブのオーバーヘッドの長さを見積もる必要がなくなりました。Web Recorder と再生エンジン（必要な場合）は、レコーダーで記録されたスクリプトが対話モードで再生されると、ジョブの総処理時間を値として返します。この値は、実行中のジョブのパフォーマンスとスケーラビリティの限界を理解しやすいように、データ系列として返すことができます。この処理時間の値が大きくなりすぎたときに通知するイベントを有効にすることができます。

また、「再生速度」を全体的に制御することもできます。すべてのステップには、ページを確実に完了させられるように最小限の遅延が設けられています。再生速度として「高速」を選択するとこの待機時間が 1 秒になり、「中速」を選択すると 3 秒、「低速」を選択すると 5 秒になります。この待機時間は、返されるレスポンスタイムの値には含まれないので、ジョブの総処理時間がそれによって長くなることはありません。いずれかの再生速度設定でスクリプトを記録してテストした後で、ジョブを設定するときに再生速度を変更することができます。

ジョブの総処理時間は、実行時間の計画を算出するために必要不可欠な値です。

## ジョブの頻度

明らかに、ジョブの頻度は実行時間の計画の主要な出力です。デフォルトで、WebTransaction Knowledge Script ジョブを 15 分ごとに実行することが推奨されています。しかし、実行時間の計画の結果すべての見積もりが常にこの推奨値をサポートするとは限りません。

## 実行時間の計画例

この例では、次のことを前提としています。

1. レスポンスタイムが約 40 秒の一群のトランザクションがあります。
2. これらのトランザクションは平均して、総処理時間を 80 秒と報告しています。

したがって、80 秒が 1 つのトランザクション（その Knowledge Script を使用したジョブの 1 ジョブ間隔）にとって最低限必要な実行時間です。

ジョブの総処理時間と等しい間隔でジョブを実行するのは賢明とは言えません。この例のジョブを 80 秒ごとに実行した場合、タイミングに少しでもずれが生じると次のジョブの実行が遅れることになり、最終的にオーバーランやデータポイントの欠落（実際には遅れ）を招く可能性があります。

実証されたアプローチは現実のテストだけですが、経験則として、総レスポンスタイム（レスポンスタイムが通常の 2 倍まで遅くなる状況に対応）と 30 秒のうち長い方を総処理時間に付け加えるという方法が有効です。

レスポンスタイムが 40 秒で処理時間が 80 秒というこの例では、少なくとも 40 秒を処理時間に付け加えて、最小限の時間的余裕をもたせたジョブ実行時間を 120 秒にする必要があります。これが、実行時間の計画を算出するために使用する数字です。

1 つの管理クライアントに対して同時に実行できる WebTransaction ジョブは 5 個までなので、5 個のジョブを 2 分ごとに実行できることになります。2 分の実行時間は 1 時間に 30 回あります。この実行時間の計画では、1 時間あたり 5 x 30、つまり 150 のジョブ反復タイムスロットを使用できます。希望するサンプリング頻度は 1 時間に 4 回なので、使用可能なタイムスロット数 150 を 4 で割った 37 が実行可能なスクリプトの数になり、各スクリプトが 15 分間隔で実行されることになります。

それ以上の数のジョブを実行するには、追加の Web-RT 管理オブジェクトを配備する必要があります。あるいは、Knowledge Script の実行頻度を 1 時間あたり 4 回より減らします。

この制限を考慮して、90 秒ごとにジョブを反復実行できるという仮説を立てることもできます（上記と同じデータを使って）。結局のところ、算出した実行時間は 80 秒だったのです。ジョブ間隔を 90 秒にすれば、1 つのジョブにつき 1 時間あたり 40 のタイムスロットを確保できます。同時に実行できるジョブは 5 個なので、ジョブの反復実行に使用できるタイムスロットは 200 になります。

個々の Knowledge Script ジョブを 1 時間に 4 回実行すれば、50 の異なる Knowledge Script を 15 分間隔で実行できる計算になります。これは、十分な時間的余裕を持たせて計算した 37 という Knowledge Script 数よりはるかに改善された数のように思えます。

この 2 番目のより積極的な計算は、紙の上では適切なように見えます。しかし（そしてこれが現実のテストが必要な理由なのですが）、50 個の Knowledge Script のうち 1 つ以上の実行がときどき（または頻繁に）完了までに 80 秒以上かかることはほぼ確実です。「120 秒」の計算では、1 回のジョブ実行ごとに 40 秒の余裕があります。より積極的な「90 秒」の計算では、この余裕が 10 秒しかありません。

ジョブの実行時間の変動を予測するのは困難です（もし不可能でないとしても）。しかし、AppManager ResponseTime for Web のバージョン 6.2 では、実際の処理時間をデータ系列として記録できるので、この値の変動と傾向を示すデータを提供できます。処理時間は、1 日のうちでも、より長期的にも変動することがあります。監視がオーバーラン状態になったときに警告を受けられるように、総処理時間のしきい値を設定することもできます。

問題の Web アプリケーションが大量のバックエンドロジックを使用したりファイル転送を実行する場合や、複数の地理的位置を表している場合には、大きな変動の可能性は高くなります。誰もが昼休みのネットサーフィンと夜中のネットサーフィンの違いから直観的にわかるように、単純なユーザー負荷によってジョブの実行時間が大きく変化することがあります。負荷バランス機能を使用しているサイトは 1 台以上のサーバがオフラインになっても通常どおりに振る舞いますが、必ずしも通常どおりのパフォーマンスで動作し続けるとは限りません。

これに関するもう 1 つの考え方は、システム管理の目標を考慮することです。サービスレベルレポートを開発するためにデータを使用する場合は、そのデータが特にパフォーマンスが悪いときに収集されるようにする必要があります。管理対象のシステムに何の問題も遅延もないときにだけ実行されるように配備を設計すると、最も役に立つときのデータが欠けてしまう可能性が大きくなります。つまり、個々のスクリプトを実行できる頻度については保守的な見積もりを使用する方が、最も信頼性の高い最良の結果が得られます。

## ジョブの実行と CPU

Web ブラウザのやりとりの総レスポンスタイムには常にクライアント側での処理も含まれるという事実を理解することが重要です。通常は、クエリや入力に対するネットワークのダウンロードおよびバックエンド応答によってレスポンスタイムが決まります。しかし、ブラウザは Web ページを表示するためにローカルリソースを（ときには大量に）使用します。

同時ジョブの最大数が増えたため、CPU の使用状況が制限要因になるケースが見られるようになっています。Internet Explorer は、ブラウジングのうち多くの計算処理を必要とする部分（レンダリングなど）のために使用可能な CPU サイクルをすべて使用するように設計されています。これは、Internet Explorer がある時点で対話的に実行されている唯一のものであると常に想定されるためです。もちろん、IE のインスタンスを 5 つ実行すると、特定の Web ページ要素の組み合わせ（ブラウザがスケーリングする必要のある多数のグラフィックなど）が生じたときに CPU に大きな負荷がかかる可能性があります。

各 ResponseTime for Web エージェント上で、AppManager を使ってオペレーティングシステム CPU を監視することをお勧めします。これにより、ジョブの実行中に CPU の制限値に達したかどうかを理解できます。一般に、頻繁に 98%以上に急上昇する傾向のある負荷や、92%以上が持続する負荷は避けるべきです。最悪のケースのシナリオでは、多くの計算処理を必要とするジョブの影響により、報告されるレスポンスタイムが遅くなります。

CPU のパワーが上がっても、実行時間の計画の制限は変わりません。しかし、パワー不足のコンピュータでは、ブラウザによる多くの計算処理を必要とするページが取得されるときに動作が遅くなることがあります。

## ステップ再生の失敗

通常、WebTransaction Knowledge Script の実行中に起きるステップ再生の失敗は、エージェント/クライアントコンピュータ上でのページ固有のレンダリングのオーバーヘッドか、高すぎる CPU 使用率を示しています。多くの場合、この種の失敗をよく起こすステップの直前に短い遅延ステップを追加すれば、この問題を排除できます。ただし、Web トランザクション内の個々のステップがハングするのは、ネットワークの問題からサーバの遅さまで、さまざまな原因があります。

## ジョブタイムアウト

通常、ジョブタイムアウトは監視対象 Web サイト自体の問題（保守ウィンドウなど）を示しています。時間的余裕が不可欠である理由の 1 つは、ジョブタイムアウトが発生しても回復不可能なジョブオーバーランが引き起こされないように余裕をもたせる必要性です。「再生のしくみ」の項で説明したように、AppManager ResponseTime for Web v6.2 では、専用のジョブタイムアウトタイマと、ジョブ完了の測定だけに使用される値が別々に用意されています。

## データポイントの欠落

データには、スクリプト実行の終了時にベース AppManager エージェントが Web-RT モジュールからそのデータを受け取ったときに、タイムスタンプが付けられます。オーバーラン状態でも、ジョブは完了するとすぐにデータを返します。しかし、15 分ごとに完了するように計画されているジョブが 20 分後まで完了しなかった場合、データの間隔が狂ってデータポイントが欠落する結果となります。「欠落した」データポイントは、そのデータが予定より遅れていることをタイムスタンプが示している場合、ジョブのスケジュールがその環境のスケーラビリティを超えていることを示します。

## データ系列とイベントの負荷

Web Recorder Knowledge Script のすべてのステップでデータまたはイベントが返されることもあり得ます。前出の実行に 80 秒かかる Knowledge Script に 15 個のステップが含まれていて、[...場合にイベントを発生]パラメータと[...のデータを収集]パラメータがすべて有効になっている場合、各ジョブにつき 17 のイベントと 32 のデータポイントが生成される可能性があります。その内訳は次のとおりです。

- 15 個の Step Response Time しきい値超過イベント
- 15 個の Step Response Time データポイント
- 15 個の Step Availability データポイント
- 1 個の Overall Response Time しきい値超過イベント
- 1 個の Overall Response Time データポイント
- 1 個の Overall Processing Time しきい値超過イベント
- 1 個の Overall Processing Time データポイント
- 1 個の Overall Availability データポイント

これにジョブ数の 30 または 50 を掛けると、大量のデータが生成されることがわかります。したがって、Web 環境を適切に管理するために必要な情報だけを選択する必要があります。

## 冗長モニタ

これは単純なスケーラビリティに関する考察を超えた記述になりますが、ResponseTime for Web を SLA モニタとして使用するには複数のエージェントの使用が必要になる場合があります。ミッションクリティカルな監視にとっては、冗長モニタを設定し、複数のアベイラビリティ系列を手動で統合して総アベイラビリティが報告されるようにするのは良い手法です。これにより、起こり得るハードウェアの問題、オペレーティングシステムの不安定さ、ジョブ実行の過負荷状態などから測定値を隔離できます。

# 結論

Web-RT Knowledge Script のスケーラビリティの限界を見積もるには、ある程度の技術が必要です。本書のガイドラインに従えば、妥当な配備に行き着くことができるはずです。特定環境のパフォーマンスの限界を調べるのに有効な方法は、テストだけです。そのテストでは、現実世界でのアクセス帯域幅と負荷を計算に入れる必要があります。配備はテスト結果に基づいて行い、Web 環境の変化しやすさを十分に考慮して調整する必要があります。URLCheck Knowledge Script は、単純なアベイラビリティ測定値と単純なパフォーマンス測定値を数多く提供します。WebTransaction Knowledge Script は特定のトランザクションとユーザーによる操作をより綿密に模倣しますが、URLCheck Knowledge Script と同じ規模では実行できません。代わりに、一群の WebTransaction Knowledge Script を実行するために必要な「実行時間の計画」を理解する必要があります。