QlikView

QlikView スケーラビリティ概要

QlikView テクノロジー ホワイトペーパー シリーズ

April 2011

qlikview.com

目次

はじめに

膨大なユーザー数、データ量、およびアプリケーション数を含むシステムをさまざまな場所で導入するニーズが高まるにつれ、スケーラビリティはますます重要になっています。

QlikView の導入を初めて計画する場合も、または既存の導入を拡張する場合も、「どのようなマシンをいつ購入したらいいのか」という基本的な疑問が常に生じます。
この疑問に対する回答は、検討すべき事項によって、非常に簡単にも、かなり複雑にもなる可能性もあります。主な検討事項は以下の通りです。

QlikView のパフォーマンスは、データとユーザーとともに均等にスケーリングします。

QlikView は、アプリケーションに追加されるデータや、アプリケーションにアクセスするユーザーが増加するのに伴い、均等にスケーリングします。QlikView では非常に大量のデータが処理されており、数千人の同時ユーザーが、サービスを中断することなく繰り返しアクセスしています。

- ソース データのサイズと性質
- 同時ユーザー数
- QlikView 導入におけるデータとアプリケーションの構築方法
- GUI と全体的なアプリケーションの設計

以下に挙げる 4 つの一般的な領域は、

- データ サイズ
- ユーザー数
- アプリケーション数
- アプリケーション設計

QlikView のスケーラビリティ能力に関する議論を進める上で、推進的な役割を果たす考慮すべき側面です。

ここでは、上記の考慮点についての課題、QlikView が推奨するアプローチ、顧客コミュニティにおける実例について説明します。

本ペーパーでは、スケーラビリティに関連する課題に対する、予測可能で相応な QlikView 導入の対応を理解する上で必要な技術力、実例、トレードオフについての概要を説明します。

ここではスケーラビリティを、「データ、ユーザー、アプリケーションを追加する場合、およびアプリケーション設計を変更する場合において、パフォーマンス レベルを持続する能力」と定義します。

本ペーパーでは、サーバー側でのデータのリロードという視点からではなく、エンド ユーザーの視点（つまり、エンド ユーザーのパフォーマンス）からスケーラビリティを調査します。

また、QlikView が典型的な使用シナリオで予想通りにスケーリングするかどうかを調査する Scalability Center の結果を確認するとともに、一般的な導入におけるスケーラビリティ要因に対する QlikView の対応を調査します。

『QlikViewアーキテクチャおよびシステムリソースの利用について』にもう一度目を通し、さまざまなQlikViewコンポーネントと、それらのコンポーネントがRAMやCPUなどの各ハードウェア リソースをどのように利用しているかについて、基本的に理解することをお勧めします。

QlikView スケーラビリティの 4 つの考慮側面

データ サイズ

組織が生成するデータ量が急増していることは周知の事実です。これはビジネス インテリジェンス（BI）においても同様です。容量が増えても、引き続きエンド ユーザーに高パフォーマンスで予測可能な体験を提供するという BI システムへの要求は増加しています。そのため、従来のクエリ ベースのソリューションでは、より高速で高価なデータベース クエリを促進する技術が奨励されることになり、さまざまな結果をもたらしました。QlikView のようなインメモリ ソリューションを使用する新しい技術ソリューションでは、RAM（Random Access Memory）リソースに対するシステムの信頼度を理解することが特に重要です。

QlikView のアプローチ：

基本的には、QlikView のパフォーマンスは、データのロードとともに均等にスケーリングすることを知っておくことが重要です。より多くのデータが QlikView アプリケーションに追加されると、それに対応して RAM と CPU 容量をともに追加することで、エンド ユーザーのパフォーマンスを予測可能な形で維持できます。

QlikView は、そのインメモリ アーキテクチャにより、RAM に大きく依存しています。QlikView 導入における RAM の役割についての詳細を説明している『QlikView Architecture and System Resource Usage Tech Brief』をご参照ください。

データ サイズが増大する中でエンド ユーザーのパフォーマンスを維持するためには、RAM と CPU 容量、両方の追加を検討することの重要性を見ていきましょう。

インメモリ ベースの技術においては、より多くのデータがアプリケーションに追加されると、パフォーマンス レベルを維持するためにはさらなる RAM 容量が必要になることは明白です。QlikView はオペレーティング システムの仮想メモリ機能（ハード ドライブを RAM として使用する機能）を利用しますが、この利用に伴いパフォーマンスが低下し、ユーザー体験が許容できないレベルになる可能性があります。QlikView は、RAM コストが継続的に低下していることを利用しています。従って、データ量の増加に伴い RAM 容量を追加しても、設備投資は比較的少なくて済みます。QlikView はまた、最新のデータ圧縮アルゴリズム（『QlikView Architecture and System Resource Usage Tech Brief』にて概要を説明）を採用しており、使用可能な RAM リソースを極めて効率的に使用してデータ分析を行っています。圧縮率は、圧縮するデータの性質に応じて、20～90%の範囲で可能です。

データをスケールアップする際に、CPU（Central Processing Unit）容量の検討が必要である理由はあまり知られていません。『QlikView Architecture and System Resource Usage Tech Brief』で述べているように、ユーザーが QlikView アプリケーションと対話する場合は、常に CPU サイクルが使用されます。たとえば、チャートが再計算される場合や新規の集約が要求される場合は、常に CPU サイクルが消費されます。どのアプリケーションにおいても、エンド ユーザーの要求に対する応答に要する時間は、要求を処理し、データの再計算を実行し、UI を再表示する CPU 能力に依存します。従って、アプリケーションに多くのデータが追加され、新規の集計を実行する必要がある場合は、CPU は相当量のデータを再計算しなければならないことから、必要とされる CPU サイクルは増加します。

以下の図 1 は、アプリケーションにデータが追加される際、CPU と RAM 容量の両方を均等に増加させることで、許容範囲内のパフォーマンス レベルを維持できることを示したものです。

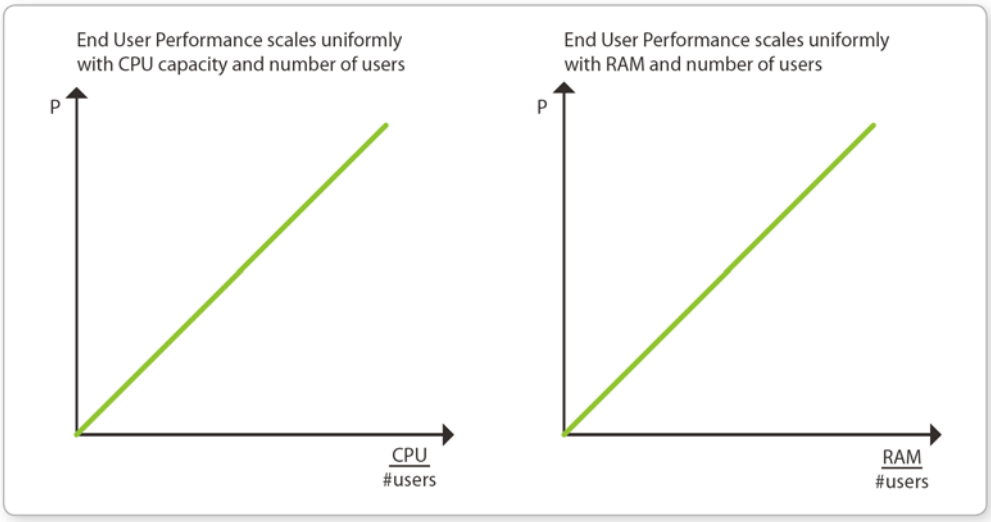

図 1

アプリケーション設計のセクションで説明するように、一般的な QlikView アプリケーションのパフォーマンス特性は、単純な型にはまったアプローチでは理解できないため、QlikView が処理可能なデータ量の「上限」を単純に述べることはできません。

ただし、多くの導入体験から、ソース データ サイズと予想される使用量に必要な RAM を大まかに計算することは可能です。この計算は、実例のための一般的かつ平均的な見積もりであることに留意してください。

$RAM = (RAM_{user} \times \text{ユーザー数}) + RAM_{initial}$

ここで、

$RAM_{initial} = QVWsize_{disk} \times FileSizeMultiplier$ これは、任意のアプリケーションの初期 RAM 消費量です

$RAM_{user} = RAM_{initial} \times userRAMratio$ これは、各追加ユーザーが消費する RAM です

$QVWsize_{disk} = SourceData \times (1 - CompressionRatio)$ これは、ディスク上の QlikView ファイルのサイズです

前提：

userRAMratio：1〜10%

FileSizeMultiplier：2〜10

CompressionRatio：20〜90%

ユーザー数とは、サポートしている合計ユーザー数ではなく、システムにアクセスしている同時ユーザー数です。例：

SourceData	50GB
CompressionRatio	90%
FileSizeMultiplier	4
userRAMratio	5%
同時ユーザー数	30

$QVWsize_{disk}$ = 50GB × (1 - 0.9) = 5GB

$RAM_{initial}$ = 5GB × 4 = 20GB

RAM_{user} = 20GB × 5% = 1GB

従って、この導入で 30 人の同時ユーザーをサポートする RAM 消費量は、以下のようになります。

RAM = (1GB × 30) + 20GB = 50GB

より実践的なアプローチは、さまざまな QlikView プラットフォーム コンポーネントを使用して、極めて高速なユーザー応答の特性を維持しながら、非常に大きなデータ サイズを提供するベストプラクティス技法を理解することです。これらの技法については、以下で詳しく説明します。

既知のデータ モデルおよび UI 設計に基づくリニアなスケーリング：

ある導入において、新たなデータ（およびユーザー数）がアプリケーションに追加される際、追加のハードウェア リソースがどの程度必要であるかを理解するためのベストプラクティス アプローチは、まず、既知の安定したデータ モデルと本稼働レベルのユーザー インターフェースを使用した小規模の導入から、パフォーマンス特性を測定することです。膨大なデータ セットやユーザー数（またはその両方）に対応できるよう QlikView アプリケーションが拡張された導入のほとんどで、当初の小規模導入のパフォーマンス特性が「ベンチマーク」として調査されました。そのベンチマークに基づき、大規模導入のハードウェア要件を決定するためのデータが推定されました。この方法は非常に有効かつ正確であることが証明されています。すべての場合において、これらの導入は、追加データ（およびユーザー）がロードされると、リニアにスケーリングすることが示されています。非常に簡略化すると、以下のようなステップになります。

1. 小規模導入のパフォーマンス特性を測定します（新規の集約集計要求または新規の検索パスに対するユーザー応答時間の測定）。
2. この環境での CPU と RAM の使用量を記録します。
3. アプリケーションへの予想追加データやユーザーのロード（またはその両方）を用いて線形外挿法を実行し、必要な追加 RAM および CPU 容量を決定します。

インテリジェントなデータ分散：水平スケーリング：

増大するデータ量を処理するためには、単一の QlikView アプリケーションを使用してすべて

のデータを格納および分析しようと試みるよりも、データを分割して分散させるベストプラクティス アプローチを使用することが望まれます。世界中の販売データを例に挙げます。小規模企業の場合は、1 つの QlikView アプリケーションを使用してビジネスを行っているすべての国のデータを分析し、提示することは理にかなっています。しかしながら、大規模企業の場合は、データ量とユーザー数が多いことからこのアプローチは非実用的です。

このような環境では、QlikView Publisher 製品を使用して、ソース アプリケーション（世界中の販売アプリケーションなど）を、データ 消費量の小さい小規模のアプリケーション（国固有のアプリケーションなど）に分割します。これらの「縮小された」アプリケーションへは、QlikView Server 経由のクライアントを使用してアクセスします。

水平スケーリングとは、サーバー リソースを仮想的または物理的に導入へ追加することです。QlikView Server は、新規のサーバー マシンを追加し、クラスタリングおよびロード バランシング戦略を実装するという簡単な方法によって、水平にスケーリングします。以下の図 2 で説明するシナリオでは、各部門専用の QlikView Server が 1 台あり、それぞれの QlikView Server は、その部門内の全グループに関連するビジネス アプリケーションを搭載しています。

このように水平スケーリングの戦略を使用すると、分析に必要な大量のデータに合わせてスケールアップしながら、エンド ユーザーのパフォーマンスを維持もしくは改善できます。

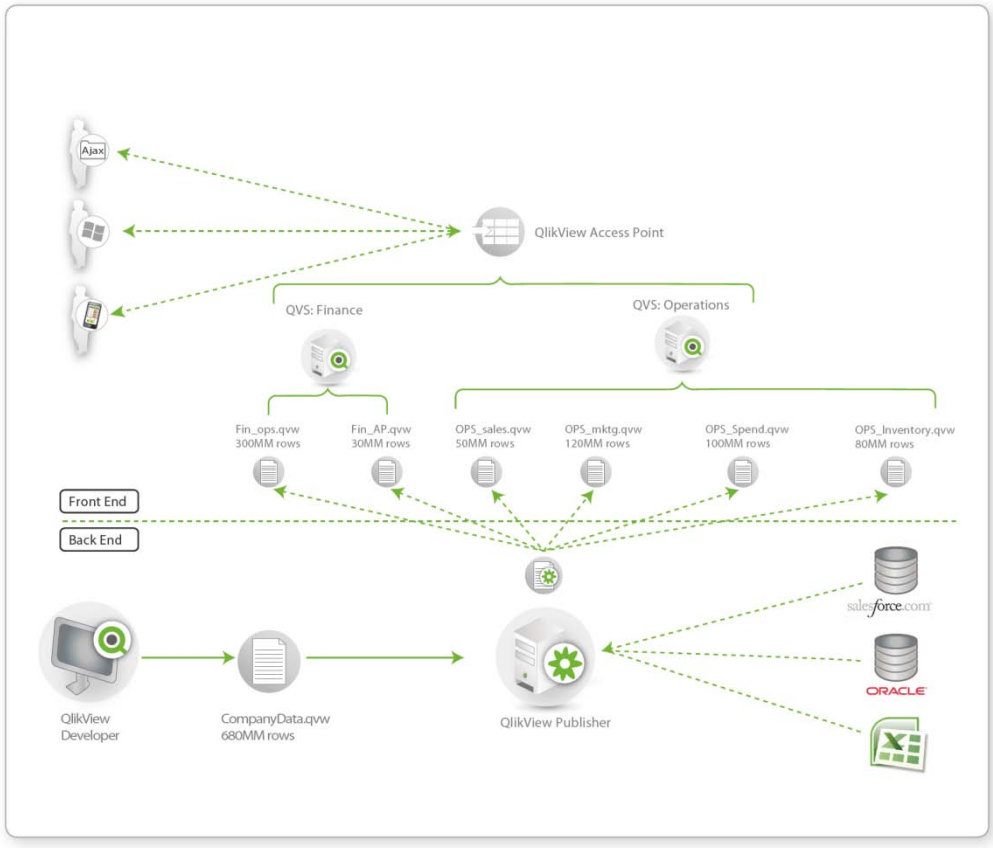

図 2

さらに、QlikView 導入は、マルチステージのデータ アーキテクチャ アプローチの概念をサポートしています。このアプローチは、水平スケーリングの戦略と組み合わせることで、エンド ユーザーのパフォーマンス特性を許容範囲内で維持しながら、数十億行を超えるデータにまでスケールアップする効果的な方法になります。

図 3 は、QlikView におけるステージ化されたデータ アーキテクチャの例を示したものです。このシナリオで QlikView が処理するデータをすべて合計すると、数十億行に上りますが、「ステージ化」アプローチを採用してデータを処理し、水平スケーリング アプローチを組み入れることで、信頼性のあるエンド ユーザー パフォーマンスを維持できます。

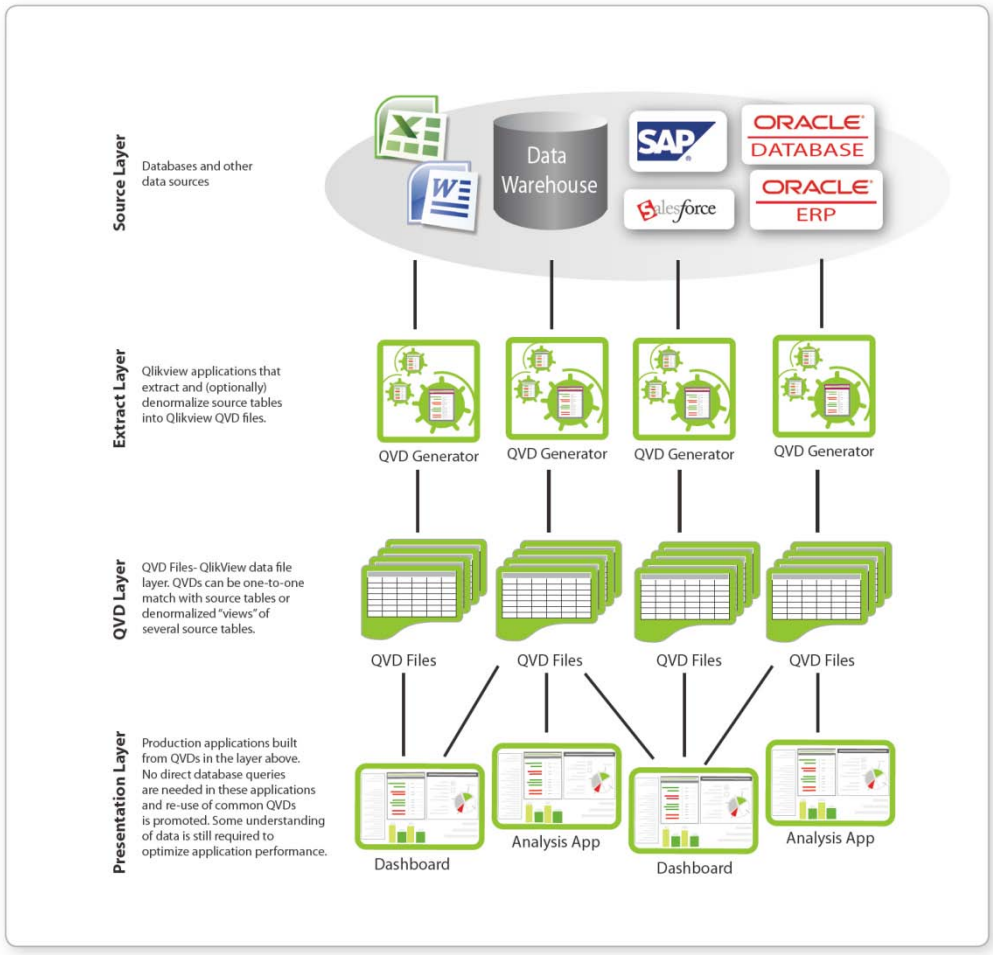

図 3

スケール「アップ」: 垂直スケーリング :

垂直スケーリングは、ハードウェア リソース（CPU と RAM）を同じマシンに追加して、データおよびユーザーの増加に対応するアプローチです。

垂直スケーリングは、水平スケーリングよりも比較的簡単な作業ですが（本ドキュメントは、両者のメリットを議論することを目的とするものではありません）、QlikView 導入は単一マシンにおける RAM と CPU 容量の垂直スケーリングを活用して、大量のデータに対応できます。これは線形で予測可能な関係です。データがアプリケーションに追加されると、RAM および CPU 容量をマシンへリニアに追加することで、エンド ユーザーのパフォーマンスが維持されます。

インテリジェントなデータのロード：増分リロード：

増大するデータを処理するための効果的な戦略は、増分リロードの採用です。このトピックは、エンド ユーザーのスケーラビリティよりも、データ モデリング戦略における考察に関連していますが、実際に導入された QlikView アプリケーションの重要な特性を強調しているため、ここで簡単に取り上げます。

増分ロードとは、QlikView がソース データから最新データまたは最新トランザクションのみをロードすることです。増分ロードは、定期的にスケジュールされたリロード プロセスの一部として実行でき、また QVD（QlikView Data）ファイルを使用するマルチステージのデータ環境の一部に使用できます。増分リロードを使用する利点は、データを指定したアプリケーションに極めて高速に追加できるため、データのリフレッシュに必要な時間を制限でき、エンド ユーザーは最新データへより迅速にアクセスできる点です。

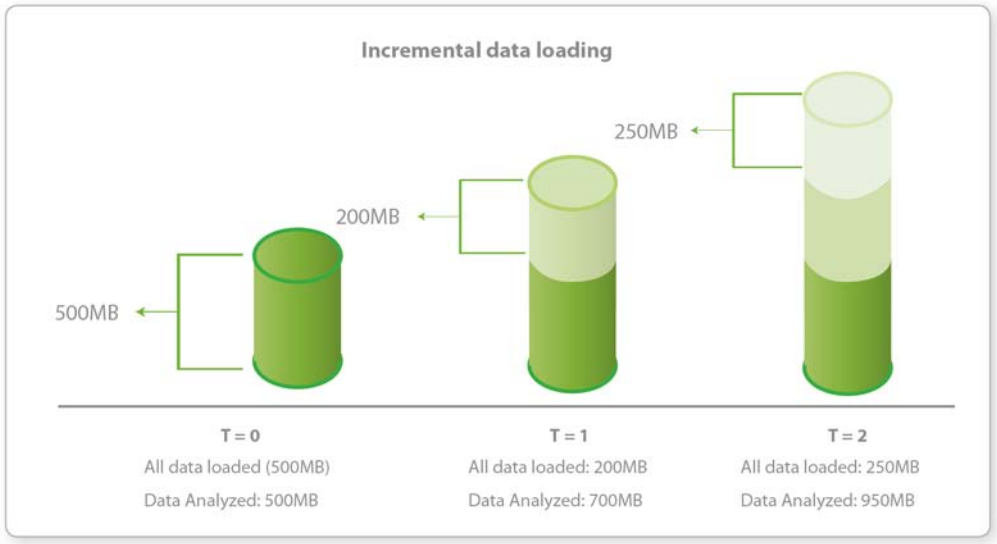

図 4

技術面の顧客事例：大手アパレル小売業者、70GB のデータ：

米国中西部の大手アパレル小売業者は、高パフォーマンスでスケーラブルなビジネス ディスカバリ プラットフォームを必要としていたため、QlikView を選択しました。複数事業を展開するこの小売業者は、男性用および女性用アパレルのデザインと販売の両方を手掛けており、世界各国の市場で数十億ドルの売上を上げています。同社は、多くの類似企業と同様に、既存の従来型ビジネス インテリジェンス ツールの課題に直面していました。新規のレポートや最新の分析に関するビジネス ユーザーの要求に対して応答時間が長いこと、柔軟性に欠けることなどから、既存ツールへのフラストレーションは増加していました。

QlikView を採用して、在庫計画と需要予測を行う 200 人以上の同時ユーザーに権限を与え、5 億行を超えるデータにアクセスできるようにすることで、意思決定プロセスを簡素化しました。

QlikView Publisher は、導入アーキテクチャに多層アプローチを使用しています。安全なバックエンド内にあり、15 のソース qvw ファイルを用いて夜間にデータ リロードを実行します。ソース データベースは、DB/2、Microsoft Excel、Oracle など多岐にわたり、データの合計サイズは 70GB、平均の夜間リロード時間は 4 時間です。これは、「ステージ化データ」環境の始まりで、250MB から 60GB までのサイズの qvd ファイルが 5 つ、中間データ ファイル ストアとして作成されます。これらのファイルは、機能領域別に分割されるデータを保持するとともに、安全なバイナリ リポジトリとして機能します。プレゼンテーション レイヤ内のエンド ユーザー ドキュメントは、このバイナリ リポジトリからデータを抽出します。プレゼンテーション レイヤとは、QlikView Server がある場所です。この導入では、4 つの QVS がマシン 4 台のクラスター環境に導入され、3 台は 64 コアの CPU と 256GB の RAM、1 台は 32 コアの CPU と 128GB の RAM を搭載し、4 台すべてが Windows Server 2003 を実行しています。

同社は、メインフレーム スケジューラを QlikView Publisher に連結し、メインフレーム上で使用可能なファイルに基づき、メインフレーム スケジューラが実際に QlikView タスクを起動する機能も設計しました。

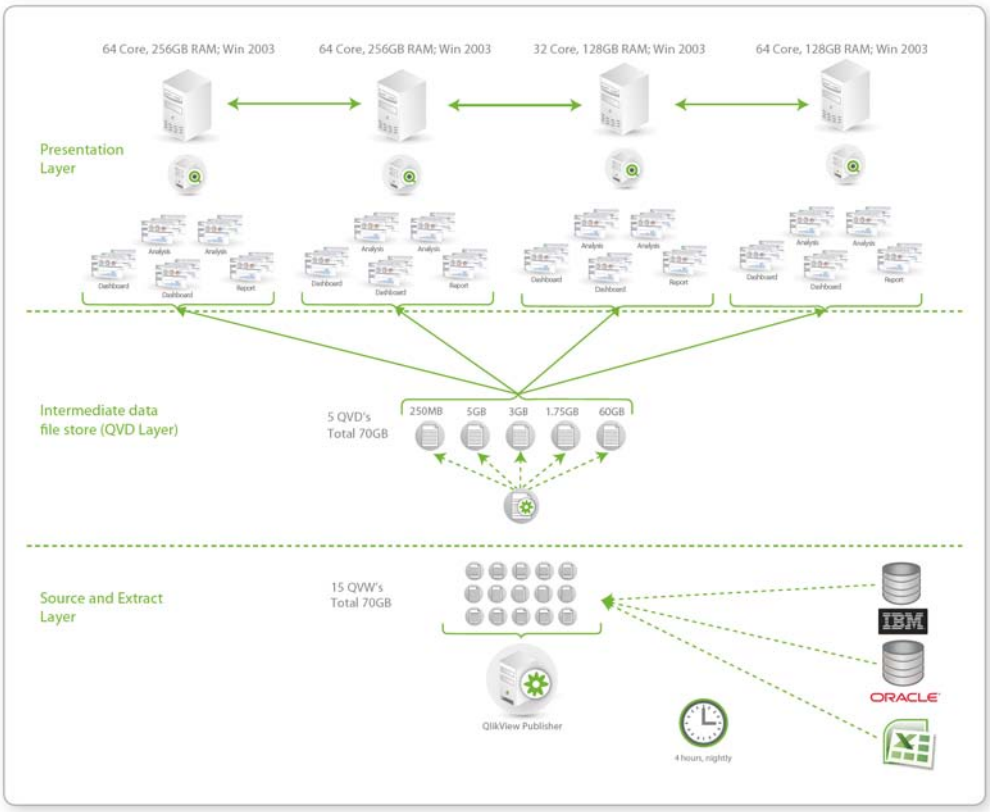

図 5

ユーザー数

企業が成長し、より多くの人がアプリケーションとリソースにアクセスする必要が出てくると、すべての IT インフラ資産に対する要求は増加します。QlikView アプリケーションについても同様です。会社の成長によって、導入したアプリケーションに対するユーザーの要求が増えるだけでなく、QlikView ソリューションの採用が組織内に根付くにつれて、要求が大幅に増加するのを目の当たりにしてきました。

IT 部門は、アプリケーションのスケーリング特性を把握する役割を担っており、新たな要求に対応するために、採用しているベンダーのソリューションが、ハードウェア リソースの増加に対して予想通りかつ比例的に対応できることを知っている必要があります。

QlikView のアプローチ：

基本的には、QlikView のパフォーマンスはユーザー負荷に合わせて均等にスケーリングすることを知っておくことが重要です。

すべての導入に見られるように、既存のシステムに新規のユーザーが追加された場合、パフォーマンスに対する影響は予測可能かつ比例的であるため、CPU や RAM 容量などのリソースを追加することで対応できます。

データ スケーリングに関する前のセクションで説明したように、より多くの同時ユーザーがシステムにストレスを加える中で、エンド ユーザーのパフォーマンスを最適に維持するためには、CPU と RAM 容量の両方を追加することで得られる効果を理解することが重要です。

ここでも QlikView は、指定の QlikView セッションで分析されるすべてのデータが格納されるインメモリ データ ストアを利用します。『QlikView Architecture and System Resource Usage Tech Brief』、および本ペーパーの「データ サイズ」セクションにおける「RAM の計算」の部分で述べているように、最初のエンド ユーザーが QlikView ドキュメントを開くときに必要な RAM 量は、通常、ディスク上のアプリケーション サイズの 2～10 倍です。これは、インデックスやデータの関連付けなどのオーバーヘッドに対応するためです。同時ユーザーが追加されると、さらなる RAM が割り当てられます。コア集約は中央キャッシュによってすべてのユーザー間で共有されますが、各ユーザーには独自のセッション状態が必要であるため、追加の RAM が割り当てられます。

『QlikView Architecture and System Resource Usage Tech Brief』、および本ペーパーの「データ サイズ」セクションにおける「RAM の計算」の部分で述べているように、大まかな方法を用いて、新規の同時ユーザーに関連するユーザーごとの追加オーバーヘッドを見積もることができます（つまり、最初のユーザーが使用する RAM の 1～10%を追加）。

> 例：1GB の.qvw ドキュメントでは、最初のユーザーは約 4GB の RAM を使用します（乗数が 4 の場合）。ユーザー数が 2 になると、計算がキャッシュされるため、必要な RAM 消費量は約 10%増加して 4.4GB となります。ユーザー数が 3 になると、消費量はさらに 10%増え 4.8GB になります。

『QlikView Architecture and System Resource Usage Tech Brief』、および本ペーパーの「アプリケーション設計」セクションの後半でも説明しているように、データおよびデータ モデル構造の密度は、UI 設計とともに、全体の RAM 消費量を決定する際に大きな影響力を持つことに注意することが重要です。

つまり、適切に設計された QlikView アプリケーションでは、2 人目以降の各追加ユーザーに対する RAM 使用量が 1～10%を超えることはありません。

ロードする同時ユーザーが増加する可能性がある場合は、CPU 容量も考慮する必要があります。『QlikView Architecture and System Resource Usage Tech Brief』で説明しているように、ユーザーがアプリケーションで新規チャートの対話型操作に基づき新規集計、新規ドリルダウン パス、UI の再表示などを要求する場合は、常に CPU サイクルが使用されます。従って、同時ユーザーの増加が予想される場合は、CPU 容量を追加する必要があります。Scalability Center の結果に関するセクションでも説明しますが、エンド ユーザーのパフォーマンスは処理能力の追加（および、それに対応した RAM の追加）によって維持できます。

図 6 は、より多くの同時ユーザーがアプリケーションへのアクセスを要求した場合、CPU と RAM 容量の両方を均等に増加させることで、許容範囲内のパフォーマンス レベルを維持できることを示したものです。

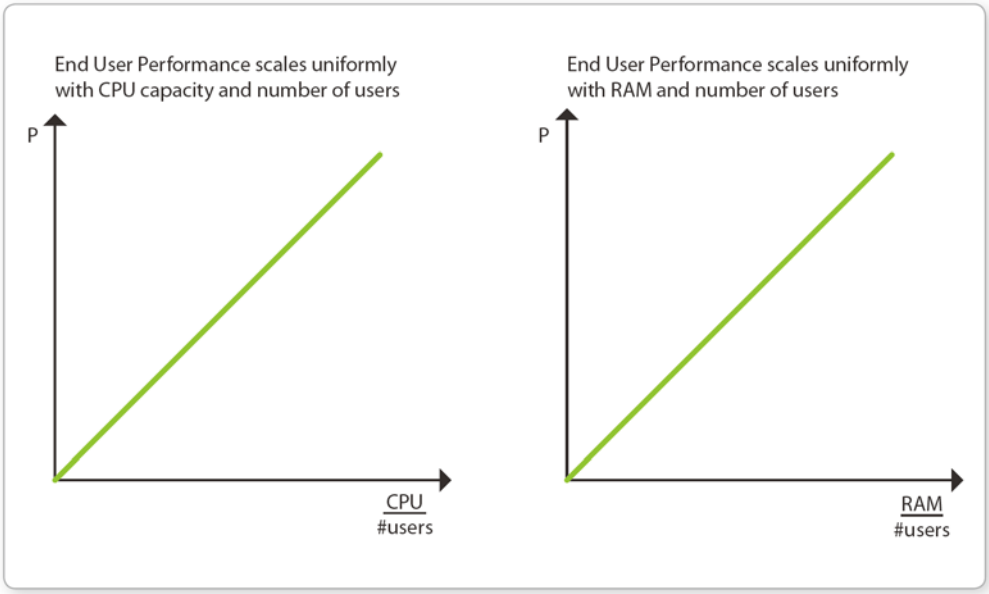

図 6

使用できるコア数や CPU が限定されているアプリケーションに対して要求を行うユーザー数が増加すると、当然ながらパフォーマンスが低下します。この場合は、クラスタリングおよびロード バランシング技術を使用した水平スケーリングによって処理能力を追加するのが最も一般的な対処法です。

クラスタリングおよびロード バランシング：

クラスタリングとは、「クラスター」という単一エンティティ内で複数のサーバーを結合する機能です。ロード バランシングとは、クラスター全体でリソース（プロセッサ容量やメモリなど）を均等に、かつ定義した方法で分散して共有する機能です。

クラスタリングおよびロード バランシングは、エンド ユーザーのパフォーマンスを低下さ

せることなく、QlikView アプリケーションを使用するユーザー数をスケールアップするための効果的な技法です。QVS（QlikView Server）クラスター（ロード バランサーとして QlikView AccessPoint を使用）は、現在のリソース要求が低い（つまり、プロセッサ使用量またはメモリ使用量が低い）サーバーに新規のユーザー要求を自動的に割り当てることで、ユーザー体験を最適化します。これは、サーバーを QVS クラスターに追加するだけで実現されます。自動フェイルオーバーはクラスター化された QlikView 環境のもう 1 つのコンポーネントであるため、ユーザーの高可用性も確保されます。以下の図 7 は、代表的な QVS クラスター環境の例を示したものです。

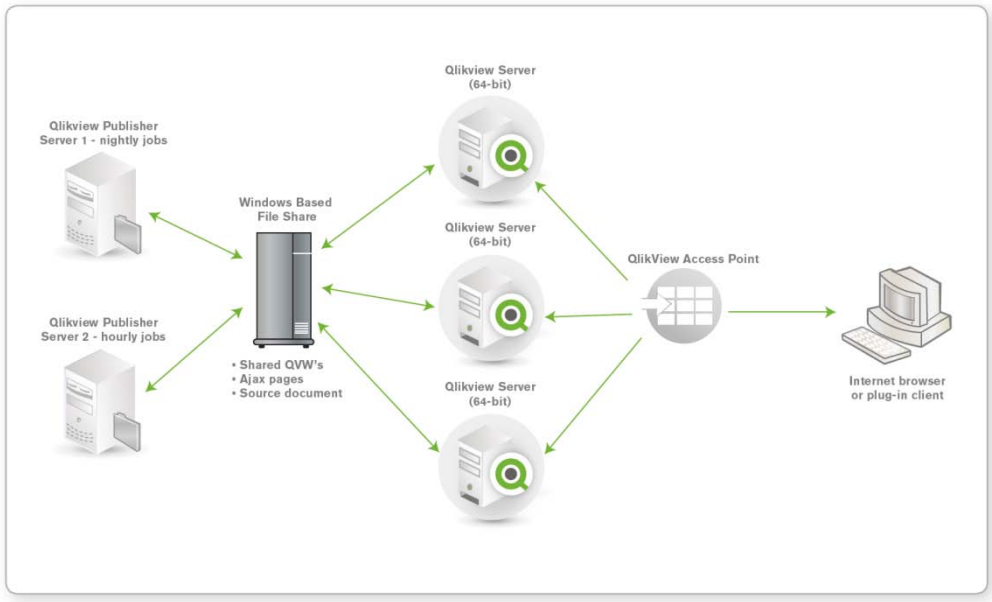

図 7

技術面での顧客事例：米国中西部の大手製造会社：8,000 人超のユーザー：

米国中西部の大手製造会社は、高パフォーマンスでスケーラブルなビジネス インテリジェンス プラットフォームを必要としていたため、QlikView を選択しました。全世界における売上が数十億ドルに上り、数千もの革新的な製品が世界中の多種多様な市場で使用され、数千人の従業員を擁する同社は、現在使用しているビジネス インテリジェンス ツール、柔軟性のないシックス シグマ プロジェクトの要求、製品の差益分析の提供ニーズなどの課題に直面していました。ビジネスの要求に対応するニーズが高まる中で、同社はさまざまな職務に就く 8,000 人以上のユーザーに QlikView を導入しました。

同社は Teradata、SQL Server、DB2 および Sybase からデータを取り込み、販売分析、サプライ チェーン分析、IT 分析、シックス シグマ分析に QlikView を使用しています。

15 台の本稼働用 QlikView Server と 4 台の開発用サーバーが使用され、11 の QlikView Publisher インスタンスがすべてのデータ リロードと配信タスクを処理しています。

本稼働中の QVW は数百に及びます。最も広く使用されているのは、販売分析、および現場担当者向けのレポート作成用のアプリケーションです。このアプリケーションは、毎週月曜日の朝に 8,000 を超える代理店にループおよび配信されます。また、Active Directory データベースと結合されており、400,000 人を超える従業員が使用しています。最大 QVW のディスク上のサイズは 600MB です。

同社は販売分析とレポート作成のために、週末に 500 MB 規模の 2 つのマスターQlikView ドキュメント（高圧縮）を処理し、5 億行、300 列のデータをロードし、QlikView Publisher で 1,800 のユーザー ドキュメントを作成します。毎日 400 のドキュメントを実行し、最大のドキュメントはディスク上で 1.1GB、約 4000 万行を保有しています。

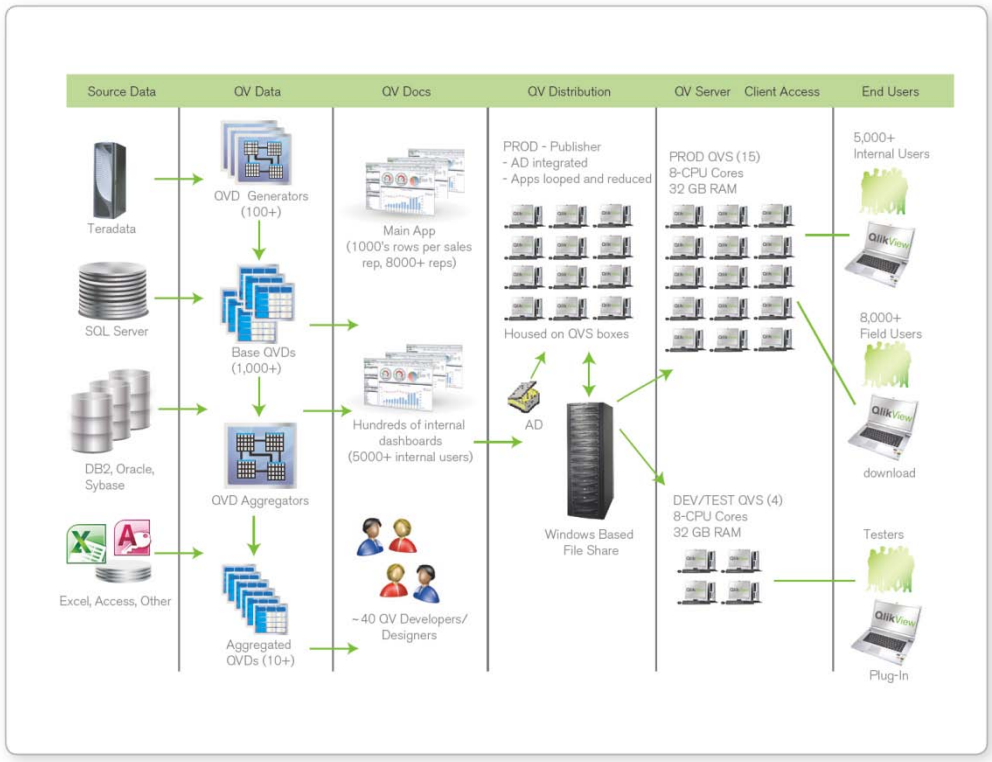

図 8

アプリケーション数

多くの場合、QlikView の導入は小規模（典型的な例としては 1 部門、1 アプリケーション）で開始されますが、各企業は QlikView の BI に対するアプローチが素早い投資利益率を実現することを知り、すぐに大規模なものへと拡張します。その他の部門は独自のアプリケーションを作成し始め、瞬く間に数百のアプリケーションが多くの場所の多くの部門で使用されるようになります。さらに、単一部門内でも使用率が増えると、さまざまな人がデータにアクセスする必要性が生じ、その結果、多様なアプリケーションが必要になります。これにより、さまざまな面において IT 要求が増加します。中でも最も重要なものは、（ソース データベースからの）データ ロード要求の処理方法、データ管理、ユーザー管理などです。

QlikView のアプローチ：

このセクションで使用する「アプリケーション」の定義を把握しておくことが重要です。通常、アプリケーションは下記を指します。a)エンド ユーザーがデータの分析、可視化、表示に使用する QlikView ファイル。.qvw ファイルとして知られています。b)データを保有し、「ヘッダーのない」（つまり、UI のない）最適化されたアプリケーションとして使用される QlikView ファイル。.qvd ファイルとして知られています。いずれの場合も、アプリケーションにはデータベースやフラット ファイルなど、オリジナルのデータ ソースから抽出したデータが含まれます。

QlikView アプリケーションの消費量が拡大した場合は、各企業の導入担当者は多層データおよび多層アプリケーション アーキテクチャのアプローチを採用することが重要です。

例を見ていきましょう。

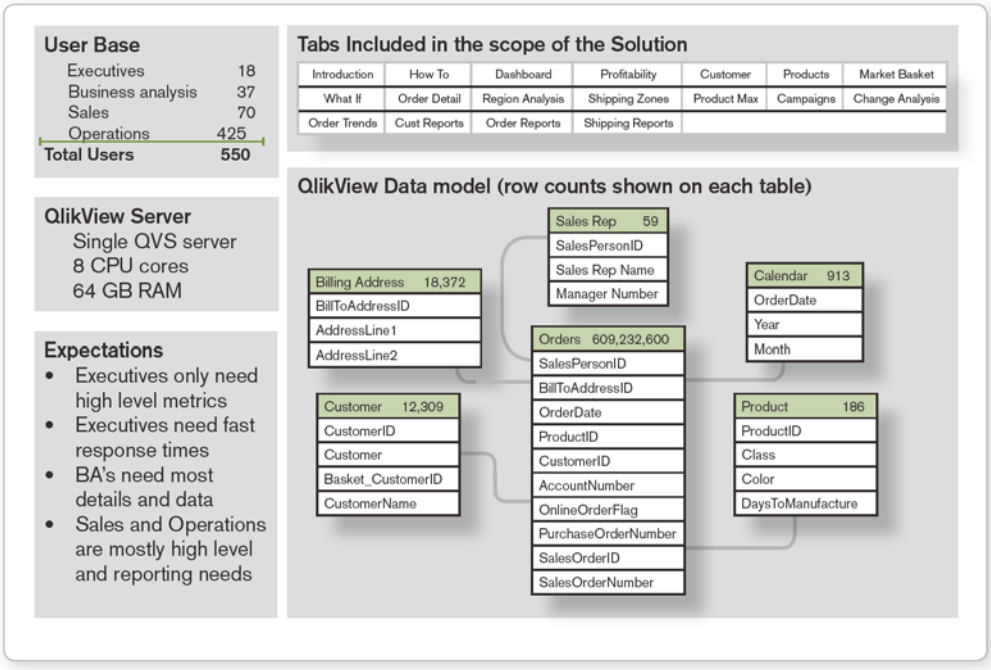

図 9

上記で概要を説明したシナリオは、会社の販売データを調査するためのものです。組織は、データへのアクセスに関するさまざまな要求を抱えており、要求の精度や専門性のレベルもまたさまざまです。以下は、導入を視覚的に表したものです。

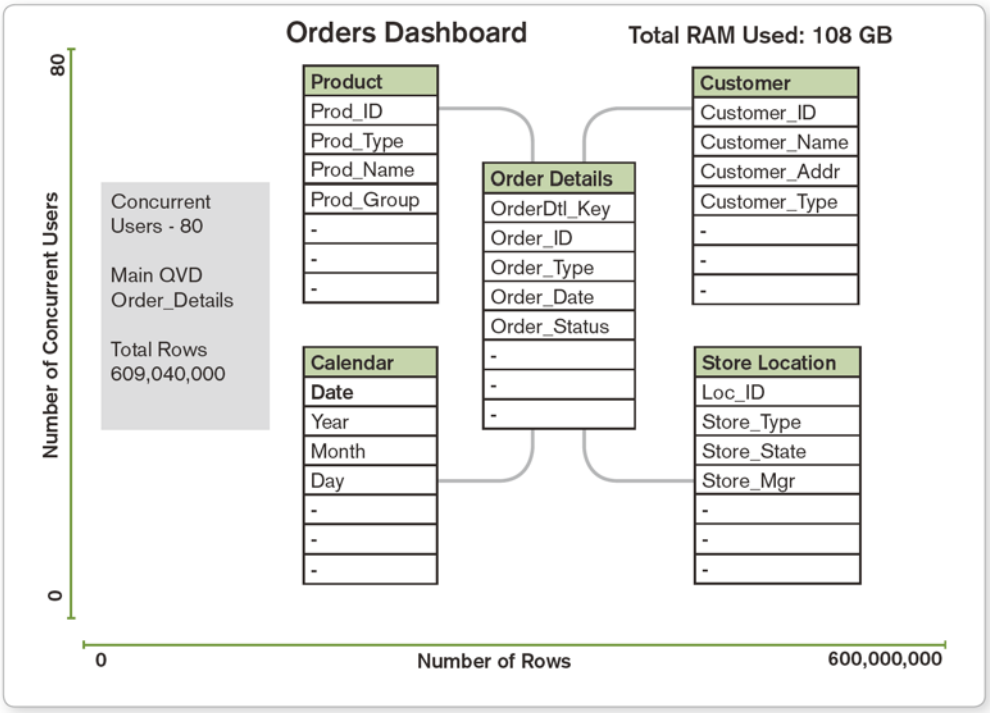

図 10

この導入では、会社の合計ユーザーは 550 人で、ビジネス上の職務と役割の両方に分散しています。データには 6 億を超える受注明細と顧客、店舗、製品、販売員、日付データが含まれています。会社は、エグゼクティブ、販売部門、および業務部門に対して、データの集約ダッシュボード ビューを提供するとともに、ビジネス アナリストに対して、詳細な分析のためにデータのより細部へのアクセスを提供したいと考えています。

シナリオ 1：すべての使用例に対応した単一アプリケーション：

このシナリオでは、受注ダッシュボードという QlikView ドキュメントを 1 つ使用します。この QlikView ドキュメントには、6 億 900 万行のデータ全体におけるデータ モデルの全関連事項が含まれています。これは非常に大きなドキュメントであり、ここで紹介するシナリオの中ではパフォーマンスは最も悪くなります。

このアプローチの利点は、開発が容易であるという点です。しかしながら、エグゼクティブのニーズを満たすため、およびビジネス アナリストが時折必要とする取引詳細にドリルダウンするために、パフォーマンスを調整することは最も困難です。このアプリケーションの消費量は大きいため、後に説明するシナリオと比較して、同時ユーザーのための容量は小さくなります。

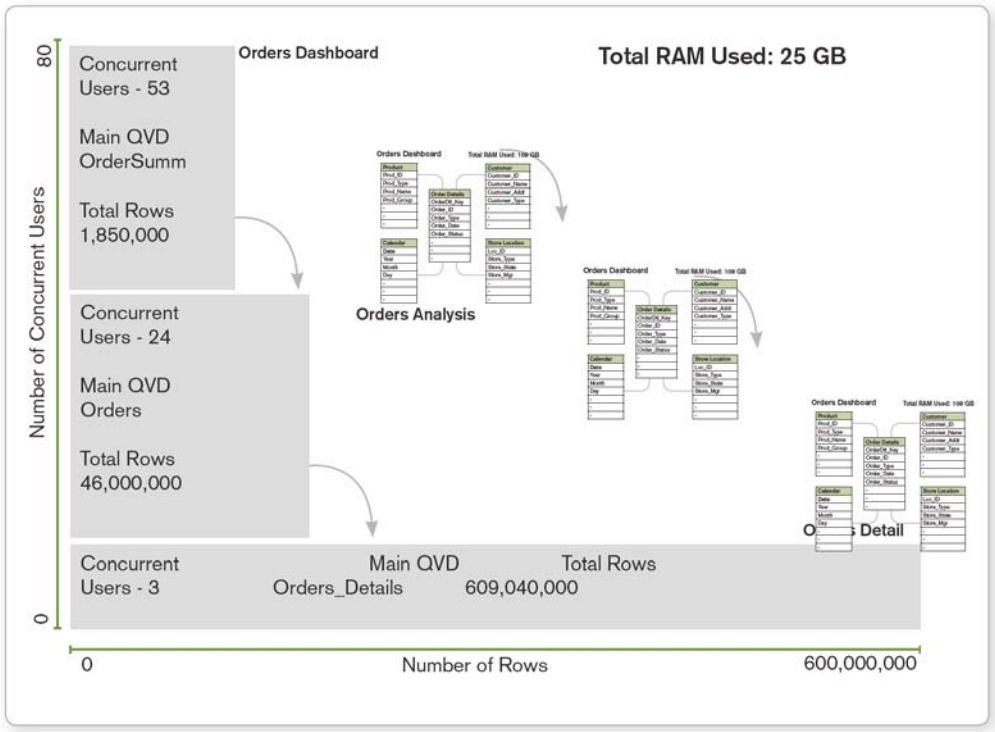

図 11

シナリオ 2：詳細別にセグメント化された 3 つのアプリケーション：

図 11 に示すこのシナリオでは、受注ダッシュボード（180 万行）、受注分析（4,600 万行 ）、受注詳細（6 億 900 万行）という 3 つの QlikView ドキュメントが使用されます。これらの QlikView ドキュメントのデータ モデルはほとんど同じですが、データの精度は非常に異なっています。ユーザーがダッシュボードから分析アプリにドリルダウンし、続けて詳細アプリにドリルダウンすると、ユーザーの同時ボリュームは減少します。これが、セグメント化されたドキュメントの通常の配信パターンです。

このアプローチの利点は、最小アプリがほとんどのインスタンスとともにメモリ内にあるため、より多くの並行処理がサポートされる点です。最小アプリケーションは最速であるため、高いパフォーマンスも提供されます。

シナリオ 3：テーマ別にセグメント化された 5 つのアプリケーション：

このシナリオでは、5 つの QlikView ドキュメントが使用されます。ダッシュボード（500 万行）が 1 つと、5,000 万行あるテーマ単位の分析アプリケーションが 4 つです。各分析アプリケーションのデータ モデルは非常に類似していますが、ベース データ モデルとは異なる考慮側面を使用して集約されます。4 つの分析アプリケーションの集約は、QVD の詳細セットから行われるか、もしくは QVD を事前集約してこれらの分析アプリケーションに提供できます。

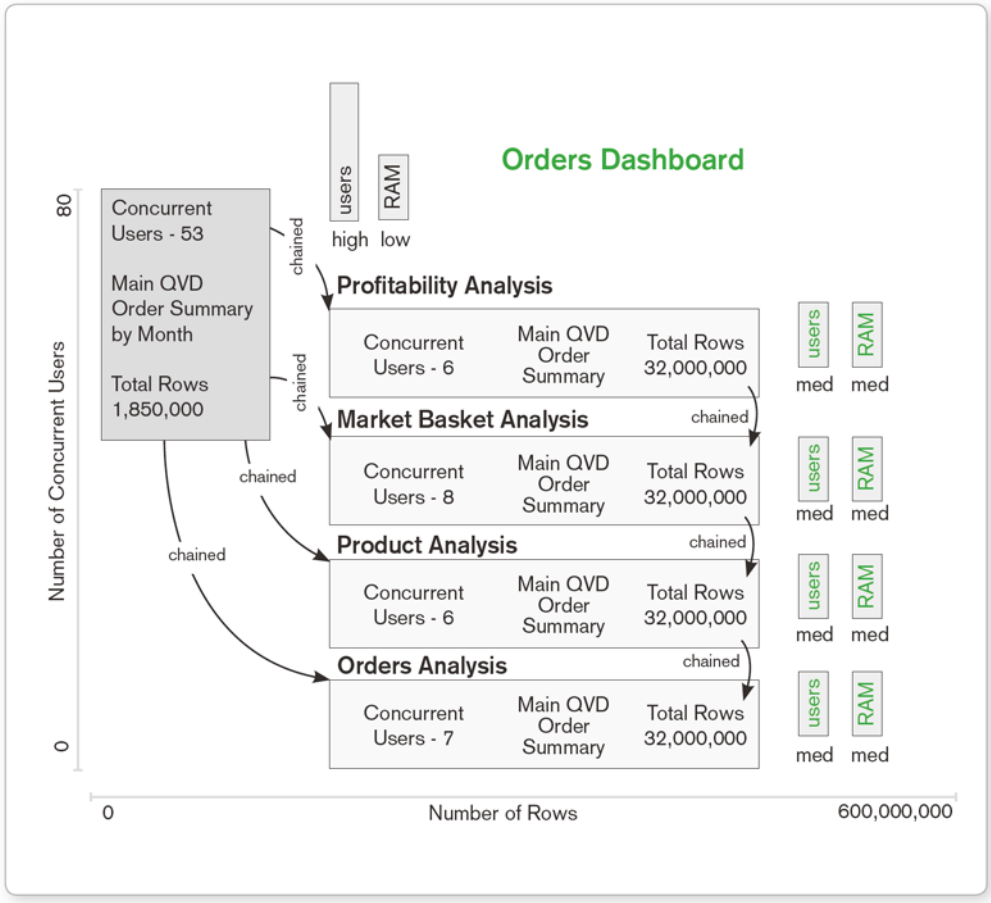

図 12

このアプローチの利点は、最小アプリケーションがほとんどのインスタンスとともにメモリ内にあるため、より多くの並行処理がサポートされる点です。

先の例からも分かるように、分析が必要なデータ タイプや分析を行うユーザー コミュニティなどに応じて、QlikView のアプリケーション数は瞬く間に増加します。そのため、複数層のデータおよびアプリケーション アーキテクチャを構築することが重要です。以下の図 13 は、増加し続けるアプリケーション数をサポートする 3 層式混合アーキテクチャの例を示したものです。

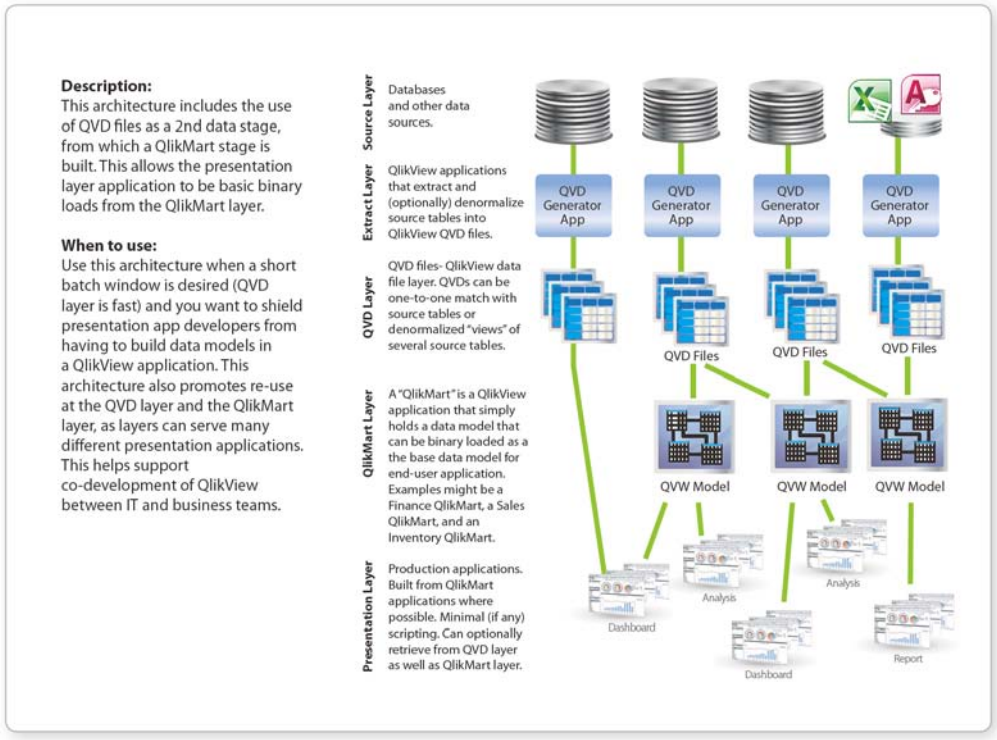

図 13

テクニカル ケーススタディ：Sandvik：5,000 万行、100 超のデータ ソース、3,000 人のユーザー

超硬工具、特殊鋼製品、鉱山・建設用機械の大手製造会社である Sandvik は、販売、価格設定、購買、人事、サービスなど、さまざまなビジネス職務を担う 3,000 人以上の QlikView ユーザーを抱えています。同社は最初に、複数のシステムからのデータを使用して顧客別の収益性を分析するアプリケーションを即座に稼動させました。現在は、400 を超える QlikView アプリケーションを搭載しており、中でも最大のアプリケーションは、数テラバイトのデータから成るデータ ソースから抽出した 5,000 万行以上を含んでいます。QlikView は、DB2、SAP BW、iSeries、Microsoft SQL Server、Oracle、テキスト ファイル、メインフレームなど、合計で 100 以上のソースからのデータを利用しています。

QlikView アプリケーションの設計

人生のあらゆる側面と同じく、悪い設計は本稼働環境で問題（または障害）を引き起こす可能性があります。QlikView アプリケーションの場合も同様です。スケーラビリティの観点から非常に重要であるのは、プレゼンテーション レイヤ（つまり、UI）の設計です。多くの類似アプリケーションと同じく、QlikView は UI がどのように設計されているか（つまり、どのようなオブジェクトを、どのように使用し、期待する対話型操作のモデルはどのようなものか）に対して敏感です。期待通りのユーザー体験を提供するためには、UI 設計のベストプラクティスを検討する必要があります。

QlikView のアプローチ：

本ペーパーは、UI 設計のベストプラクティスを包括的に調査したものではありませんが、アプリケーションの実行を非効率的にする関連要因に留意しておく価値はあります。アプリケーションの設計は、スケーリングが必要であることから、いかなるシステムのパフォーマンスにおいても重要な要素です。そのため、すべての QlikView 導入に対して、包括的なスケーラビリティの「公式」を提供する上での関連課題を強調することは重要です。

アプリケーションはその目的のために設計する必要があります。アプリケーションの設計には広範囲な意志決定が関わっています。またパフォーマンスはデータ モデル、数式、一意の値、オブジェクト数、レコード数、変数の数などに依存するため、新規プロジェクトを立ち上げる場合は、できるだけ早い段階でパフォーマンスを検討することが重要です。大規模実装の設計特性を検討する際は、実装プロセスの初期段階でアプリケーションのパフォーマンスを考慮し、監視する必要があります。

実際、同じドリル パスを使用してエンド ユーザーに同じ値を提供する 2 つの異なるアプリケーション設計では、処理能力に対する要求が異なる場合があります。たとえば、統計ボックスではなくピボット テーブルを使用する場合、またはサーバー側の「loop and reduce」ではなく、セクション アクセスを使用してセキュリティを導入する場合などです。このような違いがあることから、ベンチマーク アプリケーション（予想されるユーザー インターフェース設計と、一般的に期待されるユーザー対話型モデルを使用）を作成し、エンド ユーザーの応答特性を測定することが重要です。このベンチマークから、より多くのユーザー、データ、および／またはアプリケーションに対応できる予想ハードウェア要件を非常に簡単に推測できます。

QlikView はハードウェアを追加することで均等にスケーリングされますが、アプリケーションの設計に問題があると、ハードウェア要求が最適でなくなる可能性があります。アプリケーションの設計を徹底的に行うことで、初期の要求を大幅に減らし、ハードウェアの消費量に対してより優れたパフォーマンスを実現できます。

以下は、アプリケーションのパフォーマンスを最適化するために検討すべき設計要素の例です。

- タブ上でリストボックスを使用し過ぎない。
- テーブル ボックスとピボット テーブルを使用し過ぎない。
- ビジュアリゼーションで大きな数式を繰り返し使用しない。
- マクロを使用しない。
- UI 数式に変数を使用し過ぎない。各変数は、どれを最初に計算するかが分かっている必要がある。リロード時に、変数の一部を「固定」値としてスクリプトに移動するのがベストプラクティスである。
- 複雑な計算を使用する多数のテキスト オブジェクトを使用しない。チャートを使用して、類似のメトリクス（ミニチャートなど）を表示する。
- 今日の日付が必要な場合は、関数 date(Now)の使用を避け、today()を使用。Now()は、その日のすべてを再計算する。
- 項目で異なるテキスト値を使用し過ぎない。ファイル サイズが大きくなり、パフォーマンスが低下する。例 : 電話番号と市外局番を別の項目に分ける（(2125555555)ではなく、(212)と(555-5555)にする）。住所は 1 つにまとめずに、別の項目に分ける。
- 使用していない、もしくは今後使用しない不要項目をメモリに保持しない。select *の使用を避け、必要な項目を選択する。
- テーブルに大量のレコードが含まれる場合、可能であれば、計算条件を指定して選択を絞り込んでから表示する。
- （データ サイズのスケーリングのセクションを参照）: アプリケーション全体を配信する必要がない場合は、QlikView Publisher を使用してループおよび配信を実行し（地域別または部門別など）、各アプリケーションが小さくなるようにする。

UI 設計の決定が RAM 使用量に与える影響：

作成したユーザー インターフェース オブジェクトの特性

販売員	売上合計	コスト	利益	利益率
Tina	$900	$600	$300	33%
Tom	$600	$200	$400	66%
Teresa	$1000	$500	$500	50%

図 14

オブジェクトの定義によって、オブジェクトの描画に必要な RAM 量は大幅に異なります。たとえば、考慮すべき側面として「販売員」を表示している上記のチャートの描画に必要な RAM 量は、販売員が 3 人のみであったと仮定するとわずかです。ただし、アプリケーション デザイナーが考慮側面を「製品」に変更し、データベースに 1,300 万の異なる製品 SKU（Stock-Keeping Unit: 最小在庫単位）が保存されている場合、そのオブジェクトの描画には相当の RAM 量が必要になります。

非集計データの表示にも RAM は使用されます。たとえば、5,000 万レコードのリストボックスを表示するには、非常に高額な費用がかかるでしょう。これは、特にテーブル ボックスやその他詳細レベルのオブジェクトの描画についても言えることです。たとえば、販売員（3 人）、地域（3 箇所）、顧客（200 人）、日付（3,652 日）、オーダーID（5,000 万件）という 5 つの項目をテーブル ボックスに使用した場合、メモリ内に 2 億 5,000 万のセル（最も細かい項目のレコード数（5,000 万）に列数（5）を掛けたもの）が格納される構造を作成したことになります。このオブジェクトの表示は非常に高コストになります。ベストプラクティスは、このオブジェクトを使用しない、または、条件を付けて返されるレコード数が限定されるまで選択を行い、このオブジェクトを抑制することです。

表示するユーザー インターフェース オブジェクト数

QlikView アプリケーションを開くと、すべての表示されている（つまり、抑制、非表示または最小化されていない）チャートとオブジェクトが計算および描画され、それに対応する RAM 量が使用されます。表示されるシート オブジェクトが多いほど、それらすべてを表示するためにより多くの RAM が使用され、計算により多くの CPU が必要となります。表示されたことがなく、誰の目にも触れたことのないオブジェクトも、QlikView ドキュメントで定義されているために、多少の RAM を使用します。たとえば、非表示シート上のすべてのオブジェクトも、RAM をいくらか使用します。この場合、オブジェクトが描画され、ドキュメント内で多少の RAM が使用されている場合に必要な RAM の総量よりは少なくなります。

集計のキャッシュ

オブジェクトが一度表示されると、再度表示が必要となる場合に備えて、集計データはキャッシュされます。RAM にこの情報をキャッシュする代償として、チャート計算時間のパフォーマンスは改善されます。データのキャッシュは選択状態別にも行われます。一連の異なる選択を用いて同じチャートを表示する場合、同じドキュメント内で異なる選択がされたということで、そのチャートの表示はキャッシュされます。ユーザー インターフェース オブジェクトの描画に使用される RAM とは異なり、キャッシュされた集計の保存に使用される RAM はユーザー間で共有できるため、同じ選択状態で同じチャートを表示する複数のユーザーが同じキャッシュを利用できます。

アーキテクチャがスケーラビリティにとって重要な理由

本ペーパーを通して述べているように、QlikView のスケーラビリティに関する説明の多くは、アーキテクチャの設計と導入に対するベストプラクティスの理解と採用を中心に行われています。たとえば、IT の専門家は、QlikView が推奨する大規模データ処理のアプローチをしっかりと理解することが重要です（ステージ化されたデータ環境の使用、ドキュメントをより小さく、管理しやすく、意味のある単位に分割、マルチ サーバー環境の導入、など）。
以下の実例は、事前対応の最新のアーキテクチャ設計が、優れたパフォーマンスを実現する導入と優れたエンド ユーザー体験を生み出す理由を示しています。

シナリオ 1：

導入におけるデータは 8 億行、合計ユーザー数は 500 人です。約 10%の最大同時ユーザーにより、任意の時間における最大同時ユーザー数は 50 人です。当初は（オプション 1 にて ）、組織のニーズを満たすために QlikView アプリケーションが 1 つのみ作成され、パフォーマンス テストが実施されました。

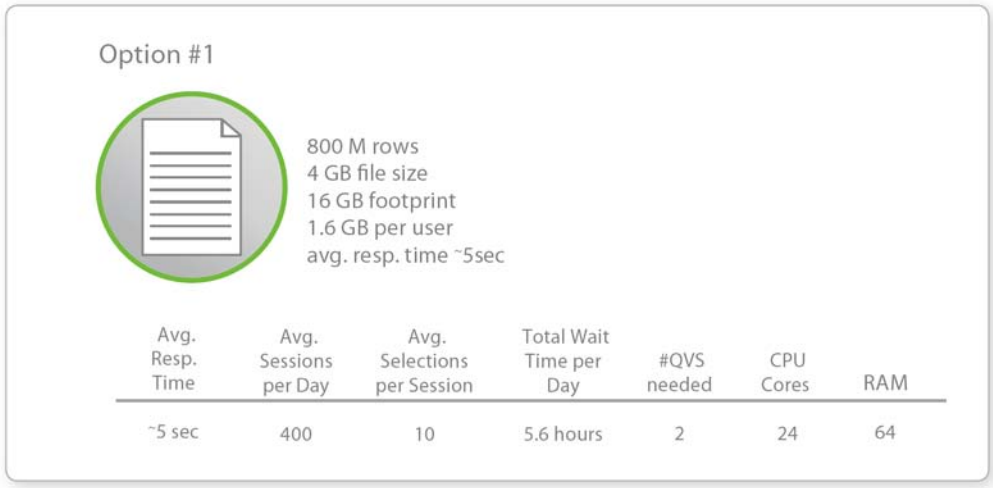

図 15

集計すると、オプション 1 では同時ユーザー50 人の 1 日あたりの合計待機時間は 5.6 時間、平均応答時間は約 5 秒です。システムで使用できる RAM 量を考慮すると、エンド ユーザーのパフォーマンスは許容できないレベルであると見なされました。

同じデータへアクセスする 2 つ目のオプション（オプション 2）が作成されましたが、関係ユーザーのために、小規模で集約されたレベルのアプリケーションも用意されました。（組織は、すべてのユーザーが下層レベルの詳細情報を必要としているわけではないことを認識していました）。結果を、以下の図 16 に示します。

QlikView 導入を構築する際に非常に簡潔なアプローチを採用することで（ドキュメントは 1 つではなく 2 つ用意し、適切な人に適切なレベルの詳細データを提供）、提供されるソリューションとデータ アクセスが同じであっても、エンド ユーザーのパフォーマンスは改善され（この場合は、最大 300%）、必要なハードウェアは少なくて済むことが分かります。

	Avg. Resp. Time	Avg. Sessions per Day	Avg. Selections per Session	Total Wait Time per Day	#QVS needed	CPU Cores	RAM
App 1	<1 sec	350	10	.95 hours	-	-	-
App 2	~5 sec	50	10	.7 hours	-	-	-
				1.7 hours	**1**	**24**	**64**

図 16

シナリオ 2：

2 社が QlikView を導入し、両社とも同量のデータ（最大 8 億行）を扱い、ハードウェア構成（16 コアと 64 GB の RAM）も同じでした。
導入管理に使用できる IT 管理リソースに関しても、両社は同様のオーバーヘッドを抱えていました。ただし、1 社は事前対応の周到な QlikView 導入アーキテクチャのアプローチを採用することで、もう一方の会社と比較して 5 倍近いユーザー数を提供できます。

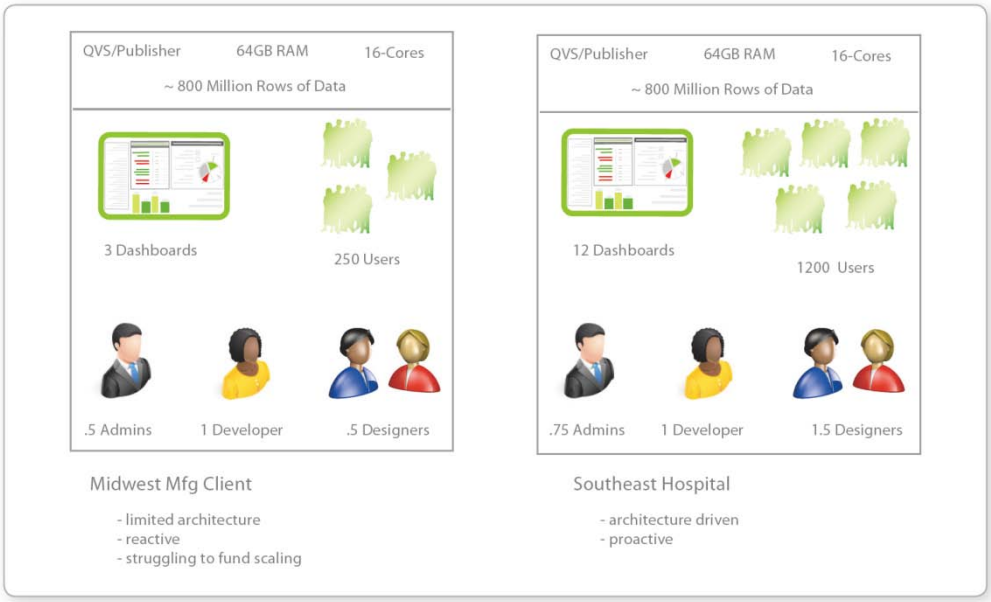

図 17

どちらの例も、大規模な QlikView 導入を実施する際に事前対応のアーキテクチャ設計のアプローチを採用することの有効性と、エンド ユーザーのパフォーマンス、ハードウェア要件、および QlikView 導入の全体的な有効性の観点から得られる利点を示しています。

QlikTech Scalability Center とベストプラクティス

QlikTech は、Scalability Center という専用の拠点でパフォーマンスとスケーラビリティに関する問題に取り組んでいます。この拠点では、大量の同時ユーザーおよび大量のデータを扱うアプリケーションの処理に重点を置いています。Scalability Center の業務の一環として、Scalability Center が提供するラボに導入された顧客固有のアプリケーションの現実的な負荷シナリオをシミュレーションするサービスを行っています。この種のサービスには、3 つの利点があります。まず、広範囲なアプリケーションのパフォーマンス測定を検証でき、優れた実績のあるカスタマイズされたアプリケーションに基づき結論を導くことで、リスクが軽減されます。次に、QlikView ドキュメントの設計および利用方法に透明性があり、製品改善の意見として、その設計方法や利用方法が R&D に直接送信される場合もあります。最後に、センター内で顧客セッションを提供することで、顧客は当然ながら、大規模導入において自社のアプリケーションがどのようにスケーリングするかについて役立つ情報を得ることができます。

方法論：

Scalability Center では、サーバー パフォーマンスのテスト ツールである Jmeter が、ユーザー シナリオの立案とデータ負荷シナリオの作成に使用されます。実際のシナリオを再現する現実的な負荷を複製するためには、顧客シナリオを徹底的に分析する必要があります。ただし、実際のシナリオの複製を作成するために使用される前提条件と指針があり、それらは以下の通りです。

- 使用量の多い時間帯は、たいていの場合 QlikView アプリケーションが新規のデータで更新されたときです
- テストは、QVS が再起動され、キャッシュが空であるときを考慮して行う必要があります
- 典型的なユーザー シナリオは、ドリルダウンに使用される可能性の高いオブジェクト、および一般的なユーザーが関心を示す可能性が高いシートを事前に定義することで、シミュレーションできます
- テストには、平均的なユーザーが実行すると思われるアクション／クリック タイプのパターンがあります
- 各々のユーザーは、オブジェクト内の異なる選択に関心を示す可能性があります
- テスト スクリプトは、特定のオブジェクト内の選択をランダムに選びます（たとえば、ユーザーが「国」オブジェクトをクリックすると仮定すると、ユーザーはさまざまな国に関心があるものと思われます）
- 一般的なユーザーは、クリックとクリックの間に思考時間があります
- テストは、クリック間の思考時間を補います。通常、この思考時間はシミュレーションするアプリケーションやユーザーのタイプによって異なります
- 平均的なユーザーは、事前に定義されたクリック数を実行するものと仮定できます。アクティブな同時ユーザー数が一定時間以上継続する場合は、新規セッションを開始する必要があります
- シミュレーションされるユーザーが最後のアクションを実行すると、新規ユーザー用に新規セッションの作成が開始されます

上記で定義されたスクリプト作成の方法論を使用して、さまざまな視点からテストの実行と調査を行い、特定の状況下において特定の QlikView アプリケーションがどのように機能するかをまとめます。QlikView がどのように機能するかについての調査結果は、アプリケーションの設計や想定されるユーザーの行動様式に依存することを理解しておくことが重要です。多くの計算処理が必要なオブジェクトもあれば、そうでないオブジェクトもあります。従って、ユーザー シナリオが実施される方法（つまり、どのオブジェクトがどのような頻度で開始されるか）が、結果に影響を及ぼします。各 VU（仮想ユーザー）は、セッション中に特定のクリック回数を生成します。クリック間での思考時間が短いと、同時 VU が少ない場合でも、時間単位ごとのシミュレーションされるクリック数は多くなります。逆の場合も同様です。そのため、顧客のベンチマーキング セッションを実行する場合は、以下の変数に実際的な値を設定することが重要です。

- 同時仮想ユーザー
- 要求間での思考時間
- ユーザー セッションあたりのクリック回数とクリック タイプ

Scalability Center の一部の調査結果の紹介：

あらゆるパフォーマンス結果および測定は、テスト セッションの対象となっている特定のアプリケーションに大きく依存します。従って、結果を解釈する際は、特定の値ではなく傾向に目を向けることが重要です。パフォーマンスの改善においては、アプリケーション設計が 1 つの重要な要素です。その他には、ある時点で必要な共通のソリューションとして、ハードウェアのアップグレードがあります。本ペーパーで先に説明したように、ハードウェアに対する要求を増大させる可能性がある状況はいくつかあります。この点を考慮する場合、ハードウェアのアップグレードによって、パフォーマンスの改善という観点から何が実現されるかを認識することが重要です。
以下の前提条件を区別してみましょう。

- 同時ユーザー数に対する要求の増加
- 応答時間改善に対する要求
- データ量の増加

ハードウェアは水平または垂直にスケーリングできます。水平スケーリングとは、クラスターへの新規サーバーの追加、またはロード バランサーによるものです。垂直スケーリングとは、既存マシンへのハードウェアの追加です。どちらが適切な選択肢であるかは状況によって異なります。最初に確認する必要があるのは、特定のアプリケーションが要求される同時セッション数を処理するのに十分な RAM があるかどうかです。先の説明から推測できるように、単一アプリケーションが RAM に収まらない場合は、サーバー クラスターに追加される小さなマシン数は問題ではありません。アプリケーションに十分な RAM があることが確認されたら、水平または垂直スケーリングを行い、同時セッション数を増やすことができます。このセクションでは、さまざまな視点から QlikView がどのようにスケーリングするかについての調査を紹介します。

優れたアプリケーション設計とは別に、実現可能な優れたパフォーマンスを提供するために必要な前提条件が QlikView Server に備わっていることも確認する必要があります。サーバー構成はそのような前提条件の一例です。次のセクションでは、大規模導入におけるサーバー設定に関するベストプラクティスを紹介します。

サーバー構成：

QlikView Server を実行するハードウェアに対して、より重要なサーバー設定の調整方法を検

証するための広範囲なテストが実施されました。概して、ベストパフォーマンスを実現するためには、表 1 に示した BIOS 設定を使用するべきであるとの結論に至りました。QlikView は NUMA（Non-Uniform Memory Access）を認識していないため、最適なパフォーマンスを得るためには NUMA を無効にする必要があります。ノード インターリーブを有効にすると、NUMA は無効になります。

変数	設定	構成方法
ハイパー スレッディング	無効	システム BIOS 設定
ノード インターリーブ	有効	詳細 BIOS 設定
ハードウェア プリフェッチ	無効	詳細 BIOS 設定

表 1：QVS のパフォーマンスを最適化する推奨 BIOS 構成

省電力設定はパフォーマンスに対しては悪影響を及ぼすため、電源管理に関する設定も、ベストパフォーマンスを得るために調整すべきものです。
電源および電力設定は高パフォーマンスに設定する必要があります。

本稼働サーバーでは、パフォーマンスのログ記録が不要な場合はオフにすることをお勧めします。パフォーマンスのログ記録は、Enterprise Management Console から停止できます。

Scalability Center の結果を述べる次のセクションにおいて、ベンチマーキングに使用した QlikView アプリケーションは小売りアプリケーションです。表 2 は、アプリケーションに関する情報を示したものです。

データ モデル	レコード数	ディスク上のアプリケーションサイズ	メモリ内のアプリケーション サイズ
Star	68 M	438 M	1 GB

表 2：このテスト セッションで使用したアプリケーションに関する統計。メモリ内のアプリケーション サイズの測定は、アイドル状態で（つまり、ロードが生成される前に）実施されました

結果 1：同時ユーザー - QVS が多数のユーザーを処理できることを示す：

先に説明した通り、QlikView サーバーが理論上処理できる同時ユーザー数は、使用可能な CPU 容量と、特定のアプリケーションの特定のセッションが使用するメモリ容量に依存します。Scalability Center で使用する通常の値は、キャッシュを除外した場合、メモリ内のアプリケーション サイズの 1～10%です。多くの同時ユーザーを処理する場合は、CPU 容量の利用度が高くなることが予想されます。使用できる処理能力が十分にないと、要求がキューに入りサービスを待機することになるため、応答時間は長くなります。

使用できる処理能力と RAM が十分な場合、QlikView が多くの同時ユーザーを処理できることを示すために、12 コアと 96 GB の RAM を搭載したマシンでテストが実施されました。

図 18 は、新規に VU（仮想ユーザー）を追加すると、どのようにアクティブ セッション数が時間とともに増加するかを示したものです。同時 VU が 1400 人に達すると、セッションは要求の生成のみを繰り返し行いながら存続します。図 19 は、選択に対する平均応答時間をプロットしたものですが、テスト セッションを通して応答時間が非常に安定していたことが分かります。応答時間のピークは、予想通り平均 CPU と相関しています（図 20）。すべての VU が起動されると、1 分あたり約 350 の選択のロードが生成されます（図 21 を参照）。サーバーの使用が最も多いときの CPU の平均使用率は約 65%になります。これらを総合すると、このアプリケーションの調査シナリオは、このテスト セッションで使用した 12 コアのマシンで安定的に実施されることが分かります。

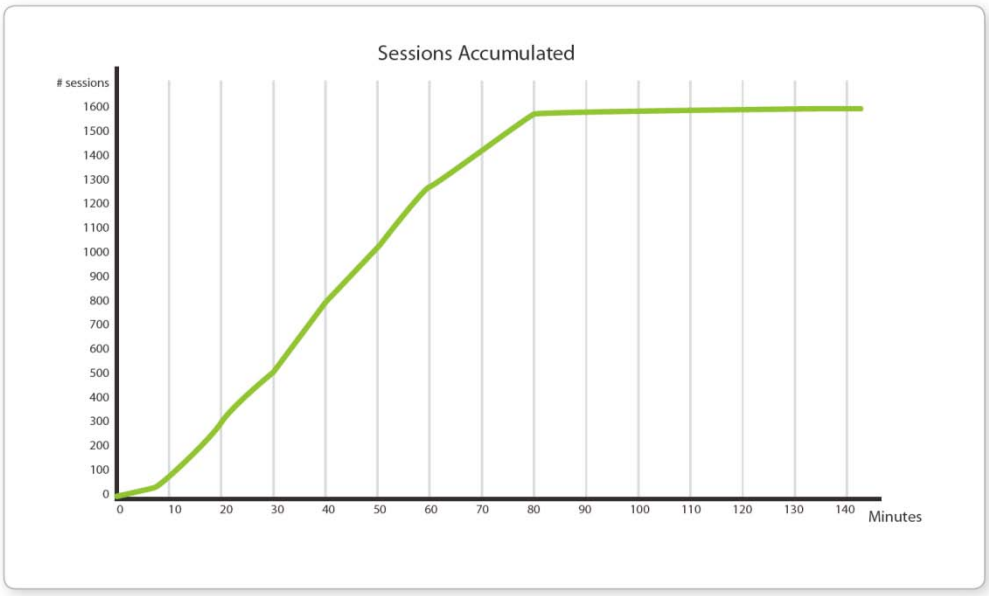

図 18：緑の線は、新規セッションが開始されたときの累積セッション数（つまり、起動中の同時 VU 数）を示しています。

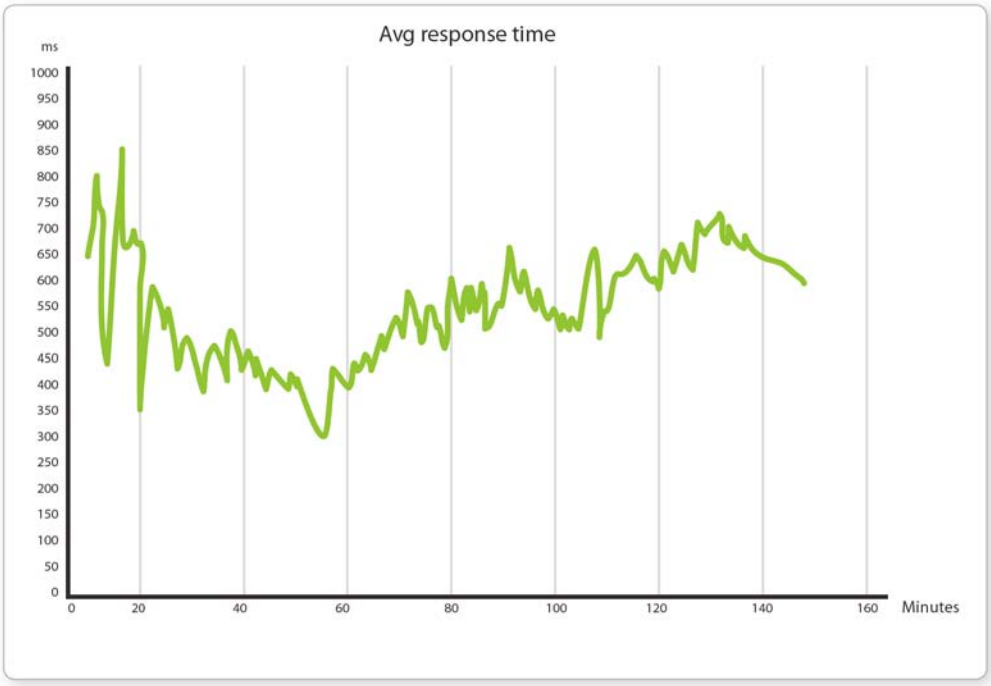

図 19：クリックあたりの平均応答時間（レンダリングは含まず）

図 20：パフォーマンスで測定したテスト セッション中の CPU ロード

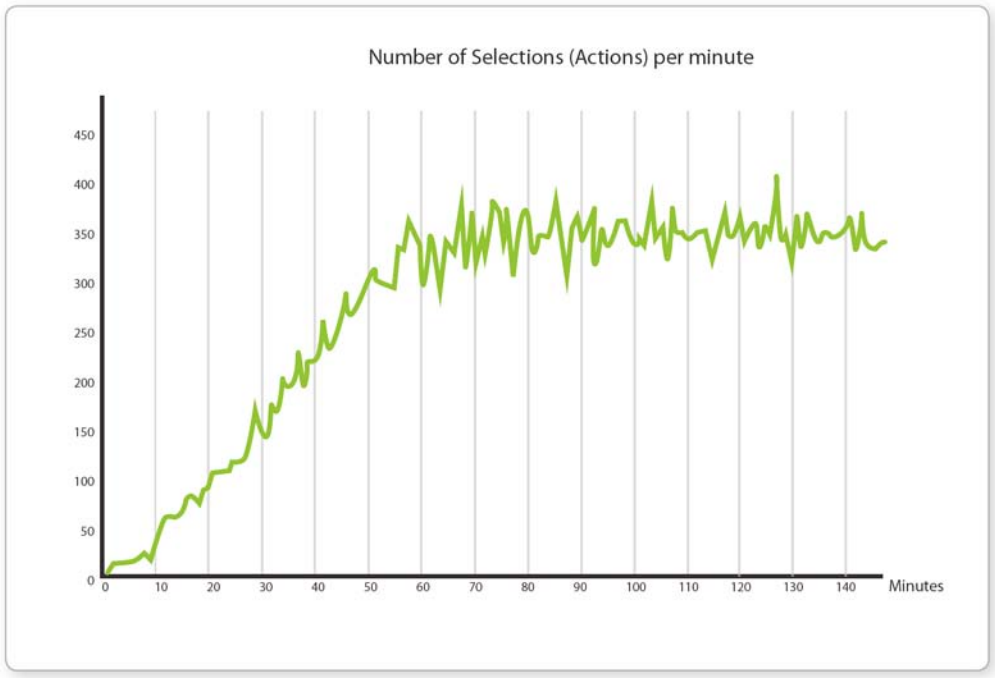

図 21：時間とともにプロットした 1 分あたりの選択数

結果 2：コア数に対するスケーリング - QlikView がプロセッシング コアの追加に伴いどのように均等にスケーリングするかを示す：

QlikView は、何らかの計算対象がある場合は、その処理のためにすべての使用可能容量を利用しようとします。処理の同時要求が複数ある場合は、使用可能な CPU 容量を共有する必要があります。つまり、応答時間は長くなります。処理要求またはデータ量が増加している場合にパフォーマンスを維持するには、追加容量が必要になります。処理能力を追加することで、QlikView のパフォーマンスがどのようにスケーリングするかを示すために、アイドル状態および高負荷時にテストが実施されました。

サーバーの使用可能な処理能力によって、パフォーマンスがどのようにスケーリングするかを検証するテストが実施されました。単一ユーザーのアイドル状態を測定することで、単一の選択がコアに対してどのようにスケーリングするか、および QlikView Server にロードを生成する多数の同時ユーザーをシミュレーションすることで、要求がより少ない操作がコアに対してどのようにスケーリングするかを確認するテストが実施されました。テスト中にさまざまなコア数を使用可能にすることで、処理能力の追加が QlikView にいかに有効であるかを調査できます。

負荷テストに使用した環境はこの目的専用のもので、外部要因（ネットワーク変数など）は無視できるものと考えられます。ベンチマーキング セッション中の環境設定は、単一 QlikView Server（QVS）と同じ専用ハードウェアを実行する IIS Web サーバーと通信する別のマシンにあるロード クライアントです。このテスト セッション中に使用されたサーバーは、6 コアの CPU を 2 基搭載し、3.33 GHz で動作します。

アイドル状態テスト：

アイドル状態のテスト中は、機能豊富な QlikView アプリケーションに対して、単一選択を手動で実行しました。実行された選択は、大量の計算に相当し、測定は非キャッシュ選択に対

して実施されました。サーバー パフォーマンスの監視は、主にタスク マネージャーの統計を収集して行われました。これらのテストで使用されたアフィニティ構成は、6 コアと 12 コアでした。

多数の同時ユーザーをシミュレーションする負荷テスト：

テスト セッション中は常に同じテスト スクリプトが使用されました。表 3 は、各 VU（仮想ユーザー）がシミュレーションした選択／クリックについての説明です。このスクリプトでは、100 人の同時ユーザーを、クリック間の思考時間 4 秒でシミュレーションします。サーバーの処理能力は、QlikView Enterprise Management Console でコア数のアフィニティを変更して調整されました。この調査で使用したアフィニティ構成は、3、6、9 および 12 コアでした。

ステップ	説明
1	ドキュメントを開き、新規セッションを作成
ループ開始	各セッションはその選択をこのステップから 5 回ループします。5 回繰り返すと、新規仮想ユーザーが起動され、ステップ 1 から再び開始されます。
2	シート 1 に変更
3	年度リスト ボックスからランダムに選択
4	月リスト ボックスからランダムに選択
5	マルチ ボックス「upc」からランダムに選択
6	チャート 1 のランダムな領域を拡大・縮小
ループ終了	上記のステップを実行したら、セッションをステップ 2 から繰り返します。

表 3：各仮想ユーザーに対する実装済みのシナリオ

結果：

コアの増加により QVS がスケーリングすると、ユーザー応答時間がコアの追加に対して極めて比例的に短縮されることが分かりました。

多数の同時ユーザーをシミュレーションする負荷テスト：

さまざまな利用可能コア数で構成された同一サーバーでテスト スクリプトが実行されました。このスクリプトでは、4 秒ごとにクリックを実行する 100 人の同時ユーザーがシミュレーションされます。図 22 は、異なる構成のサーバー要求に対する平均応答時間をプロットしたものです。サーバー要求は、QVS からアクションを要求するものとして定義されます（静的コンテンツに対する要求は除外）。利用可能な処理能力が増加するのに伴い、応答時間が短縮されることが分かります。また図 23 では、使用可能なコアの CPU 使用率が減少していることが分かります。

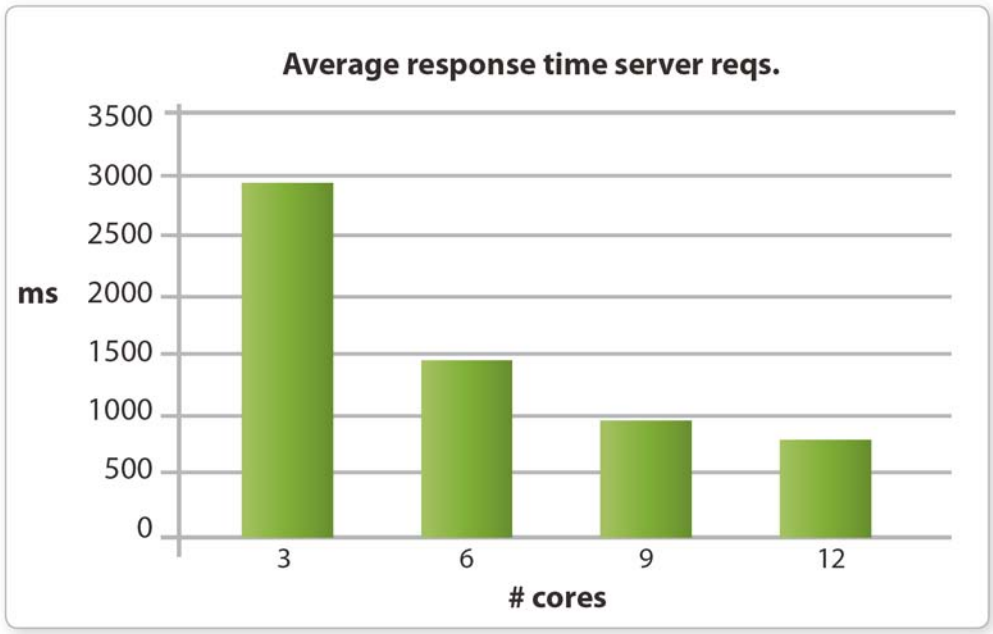

図 22：サーバー要求に対する平均応答時間。

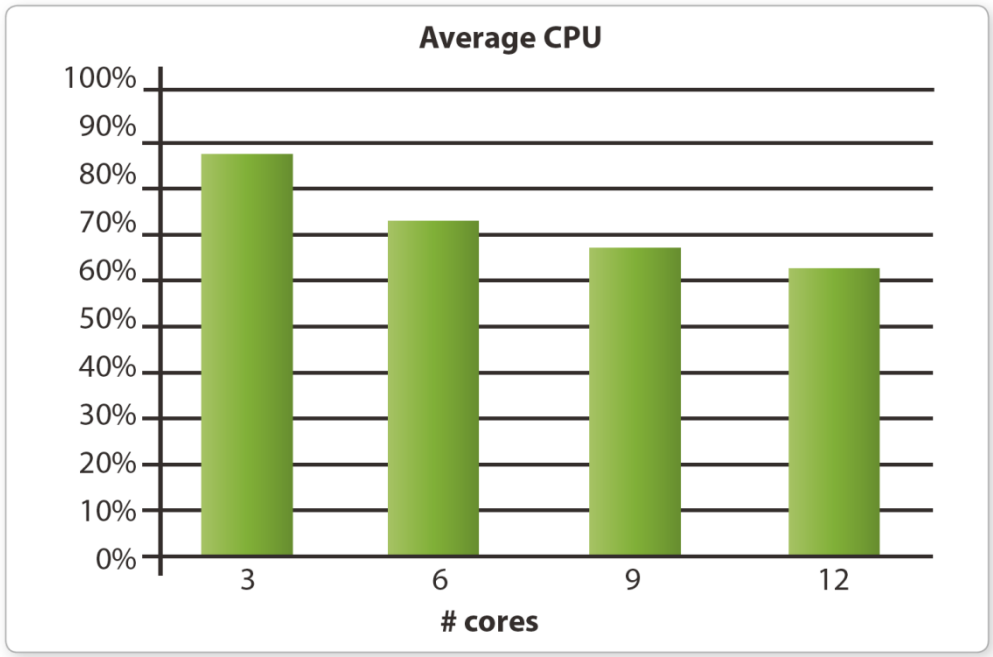

図 23：異なるシナリオの平均 CPU 使用率をプロットしたもので、構成済みコア数内の利用可能処理能力に 100%一致しています。

アプリケーションに使用できる処理能力の増加のみを考慮した場合、CPU 使用率の減少は予想されるほど顕著ではありません。ただし、これはスクリプトの実装／設計によるものです。VU の応答時間が短い場合でも、思考時間は 4 秒のままです。つまり、応答時間が短い VU は、同じ時間内により多くの要求を生成するということです。図 24 は、100 人の VU に対して 1 分あたりにシミュレーションされたクリック数として、スループットをプロットしたものです。

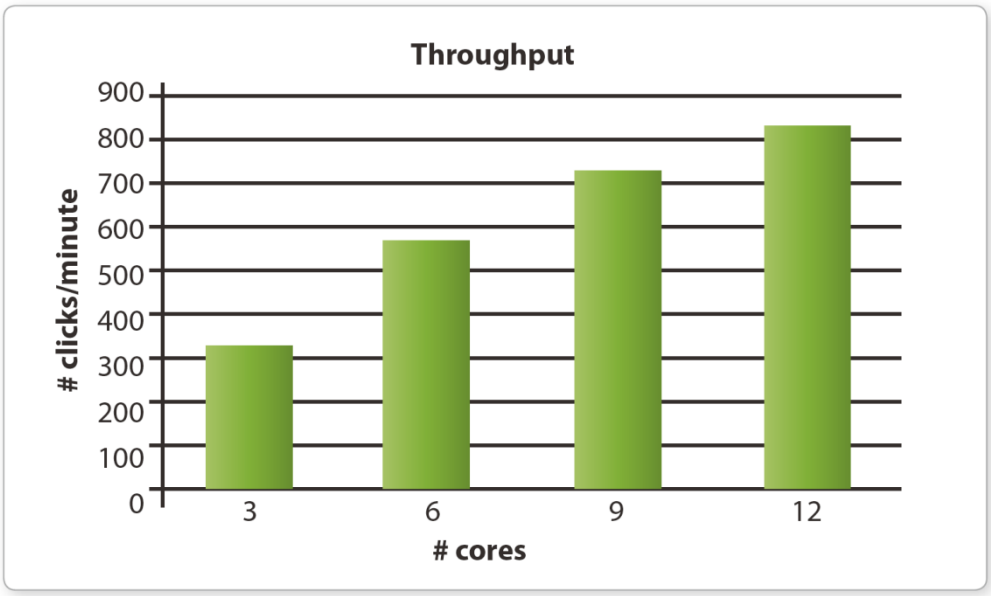

図 24：異なるテストの実行でシミュレーションされた 1 分あたりのクリック数

シミュレーションされたユーザー シナリオでテストされたアプリケーションのパフォーマンスが、処理能力の追加に比例してスケーリングしていることが分かります。特定の数のユーザーが利用すると仮定されるアプリケーション数に必要なハードウェアの大きさを示す場合は、許容できる応答時間のために十分な処理能力があることを確認することが重要です。短い時間、CPU を 100%使用することは適切なことです。応答時間が短くなり、ユーザーの体験が向上します。ただし、CPU 使用率が概して飽和状態になっていないことを確認する必要があります。そのような状態は、処理要求がキューに入り、結果的に応答時間が長くなることを意味しています。

結果 3：レコード数に応じたパフォーマンス：

テストに関する技術情報：

ベンチマーキングに使用された、さまざまな大量のデータが入力された QlikView アプリケーションは、典型的な小売りアプリケーションです。オリジナル ドキュメントは 100 店舗からのデータを提供します。アプリケーションのデータ モデルはスター形状です。この調査で使用するアプリケーションのサイズの差異は、アプリケーションに含まれている店舗数と相関性があります（表 2 を参照）。レコード変数はソース データに含まれるトランザクション数に対応しています。

アプリケーション	レコード数（M）	ディスク上のアプリケーション サイズ	メモリ内のアプリケーション サイズ
10 店舗	68	438 MB	1 GB
30 店舗	168	980 MB	2.5 GB
50 店舗	271	1.52 GB	4 GB
70 店舗	315	1.77 GB	4.7 GB
90 店舗	400	2.33 GB	6.4 GB

表 4：このテスト セッションで使用された異なるアプリケーションへのレコード数のマッピング。メモリ内のアプリケーション サイズの測定は、アイドル状態で（つまり、ロードが生成される前に）実施されました

負荷テストに使用した環境はこの目的専用のもので、外部要因は無視できるものと考えられます。テスト環境に関する詳細は、以下の表をご参照ください。

環境	属性	値
QVS	バージョン	9 SR7
	RAM	96 GB
	コア数	12
Web ブラウザ	バージョン	Firefox 3.6.15

表 5：テスト セッションで使用した環境

アイドル状態分析：

このセクションでは、CPU および RAM に対する要求がレコード数にどのように依存しているかを調査します。表 2 で示されたアプリケーションが、基準測定として使用されています。特定のオブジェクトの計算に必要な CPU 秒数は、要求されている計算の種類と、計算に関与するレコード数によって異なります。オブジェクトがデータ テーブル内のすべてのレコードが関与する何らかの計算を要求している場合、使用されるレコード数が多いことから、必要な CPU 秒数は増加します。100 個の値を合計するには、10 個の値を合計するよりも多くの労力を要するわけで、これは非常に論理的なことです。これを可視化するために、ベンチマーク テストが実施されました。いくつかの異なる選択の計算に費やされた CPU 秒数を記録するだけで、これが比例的にスケーリングしている様子が図 25 に示されています。すべての測定は、非キャッシュ選択に対して行われました。しかしながら、この例においては、異なる種類の選択に対する曲線のカーブは同じではありません。その理由は、特定の選択にはさまざまな計算が使用されているためです。以下は、例における選択が何の計算に対応しているかを簡単に説明したものです。

期間別売上：トランザクション数と同じレコード数（つまり、表 2 のレコード数）に対して、3 つの合計数式を用いて 1 つのオブジェクトを主に起動するシートを開きます

項目別売上：トランザクション数と同じレコード数（つまり、表 2 のレコード数）すべてに対して、6 つの多様で複雑な計算を用いて 1 つのオブジェクトを主に起動するシートを開きます

年度選択：項目別売上シートでの選択。項目別売上を開くときに起動されるものと同じ 6 つの計算を起動します

年度選択は、オリジナルのデータ量の 3 分の 1 をわずかに超える選択に相当します。項目別売上選択と年度選択の経過 CPU 秒数の比率を比較する場合、この比率は選択したデータ量に比例した関係になります（つまり、必要な CPU 秒数の 3 分の 1 をわずかに超過）。これは、データ量と関連した CPU に対する要求に比例して、QlikView がスケーリングすることを示しています。

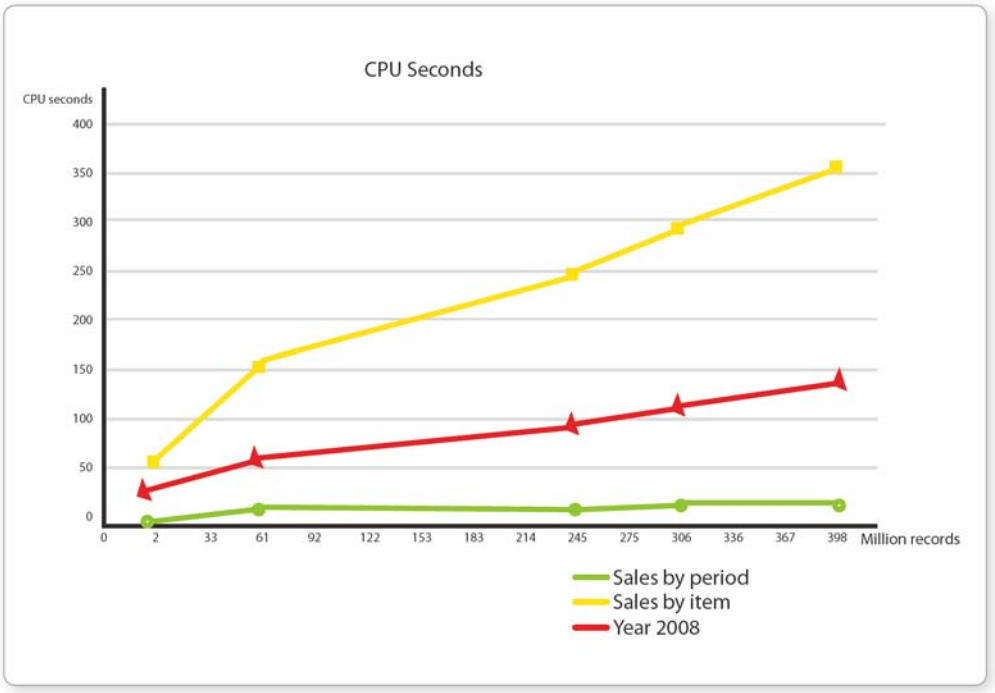

図 25：レコードに応じた CPU 秒数

まとめ - QlikView は均等かつ予想通りにスケーリングし、証明された実績がある

本ペーパーでは、分析に必要なデータ量、またはシステムへのアクセスを要求するユーザー数をスケールアップする必要性が生じた際、QlikView が採用するアプローチで、どのようにして一貫性があり、かつ予測可能なエンド ユーザーのパフォーマンスを提供できるかについて要約しました。

QlikView の製品コンポーネントおよびアーキテクチャは、水平および垂直スケーリングを実現でき、システムの要求が増加した際に活用されます。QlikView は均等にスケーリングすることから、IT の専門家は、エンド ユーザーのパフォーマンスを期待される通りの高いレベルで維持するために、将来予想されるシステム使用に対して効果的なキャパシティー計画を立てることが可能です。

QlikView 導入では、数千人の同時ユーザーと極めて大量のデータを処理できることが一貫して証明されており、エンド ユーザーのパフォーマンスも優れています。IT の専門家は、クラスタリング、パフォーマンス監視、水平および垂直スケーリングなどの標準的なアプローチに従って、システムを管理できます。