WHITE PAPER

Mobile Linkのパフォーマンス

概要

Mobile Linkは、リモート・クライアントのデータベースと中央のデータベース(統合データベースと呼ばれる)との間で、スケーラブルでハイ・パフォーマンスなデータ同期を確立するものです。Mobile Linkの同期パフォーマンスの主要な要素は次のとおりです。

- 同期スクリプトに対する統合データベースのパフォーマンス
- リモート・クライアントの処理速度とネットワークの帯域幅
- Mobile Linkを実行するコンピュータの速度
- Mobile Linkのワーカー・スレッド(それぞれが統合データベースに接続される)の数

Mobile Linkの同期のパフォーマンスについては、次のとおりです。

- Mobile Linkのハードウェア要件は比較的少なく、複数のプロセッサをうまく利用しています。たとえば、Adaptive Server Anywhere(ASA)統合データベースを用いた当社のテストでは、Mobile Linkが必要とする処理能力は、ASA以外の統合データベースの半分以下です。ASAとともに使用すると、プロセッサ数の増加とともに着実にパフォーマンスが向上します。
- Mobile Linkのスループットは、リモート・クライアントの追加に伴って直線的に拡張されるため、少数のクライアントでのテストから、クライアント数が非常に多い場合のスループットを予測できます。
- Ultra Lightのダウンロードでは、リモート・クライアント側で相当の処理が必要となる一方で、Mobile Link側はほとんど処理を必要としないため、ダウンロードのスループットは、アップロードに比べて、クライアントの速度に大幅に依存することになります。
- 最高のスループットを得るには、統合データベースへの最適同時接続数より多いMobile Linkワーカー・スレッド(それぞれがデータベースに接続)は使用しないでください。たとえば、高速なクライアントによるテストでは、ワーカー・スレッド数が5の場合にMobile Linkで最高のスループットが得られました。特にダウンロードの場合には、クライアントが低速になるほど、ワーカー・スレッドの最適数が大きくなりました。
- 同期ごとにオーバヘッドが存在するため、同期サイズが小さくなるほどスループット・レートが低下します。
- Mobile Linkは、接続と接続の間の状態を管理していないため、複数のMobile Linkサーバを使用できます。たとえば、高可用性とスケーラビリティを得るために、サード・パーティ製のロードバランシング・システムを備えた複数のMobile Linkサーバを使用して、これらを単一のサーバのように見せることができます。

パフォーマンスの例として、1,000のクライアントが92バイトのローを1,000個同期した結果を次に示します。それぞれ、Pentium III 550MHzプロセッサを4基搭載したDell PowerEdgeサーバ上でMobile LinkとAdaptive Server Anywhere統合データベースを実行しています。

統合データベースからのダウンロード：77秒または13069ロー／秒
統合データベースへの挿入のアップロード：450秒または2223ロー／秒
統合データベースへの削除のアップロード：570秒または1755ロー／秒
統合データベースへの更新のアップロード（競合の自動検出を含む）：1044秒または958ロー／秒

本書の後半で、上記の情報を補足する詳細な内容について説明します。

目次

はじめに

私どもは、Mobile Link 同期技術を開発することにより、多数のリモート・データベースと単一の中央データベース (統合データベースと呼ばれる) との間で、効率的でスケーラブルなデータ同期を可能にしました。Mobile Link は、SQL Anywhere Studio に含まれています。この Mobile Link によって、リモートの Ultra Light や Adaptive Server Anywhere データベースと、Adaptive Server Anywhere、Adaptive Server Enterprise、Microsoft SQL Server、Oracle、IBM DB2 などの ODBC アクセスをサポートする統合データベースとの間でデータ同期が可能になりました。

本書は、ユーザが Mobile Link のパフォーマンスについて理解し、データ同期のニーズに対する Mobile Link の適用性を評価できるようになることを目的としています。本書では、最初に Mobile Link のデータ同期において時間のかかる手順について説明し、様々な条件で実施した Mobile Link 同期技術のパフォーマンステストの結果を紹介します。

本書では、まずテスト方法について簡単に説明し、その後で次の条件を変化させた場合のパフォーマンス結果と分析について記述します。

- Mobile Link のワーカー・スレッドの数
- 同時に同期を行うリモート・データベースの数
- 各同期で転送されるデータ量
- 中央サーバで使用されるプロセッサの数

次に、Mobile Link から最高のパフォーマンスを得る方法と、実際の状況においてパフォーマンスを効率よく予測できるようにするための小規模テストの実施方法を示します。

Mobile Link の同期で時間のかかる手順とは？

Mobile Link のパフォーマンスを理解しやすくするため、まず、1 つの Mobile Link 同期での手順を調べ、各手順がどの程度、同期のために時間を費やしているのかに注目します。その後で、クライアントを増やした場合について同様に調べ、予測されるボトルネックについても調べます。

開始

Mobile Link の同期は、リモート・クライアントによって開始されます。クライアントが Ultra Light データベースであるか、Adaptive Server Anywhere(ASA) データベースであるかによって、イベントの順序はわずかに異なります。Ultra Light のクライアントは、最初に Mobile Link との接続を確立してからアップロード・ストリームを構築して送信しますが、ASA のクライアントは、アップロード・ストリームを構築してから接続を確立します (アップロード・ストリームの構成方法が異なるため、Ultra Light のクライアントは、極めて少量のメモリで作業できるようになります)。

接続

Mobile Link の同期には、2 種類の接続があります。クライアントと Mobile Link 間の同期接続と、Mobile Link と統合データベース間のデータベース接続です。

クライアントは、Mobile Link 同期サーバと同期接続を行います。クライアントは、Mobile Link がワーカー・スレッドのプールからワーカー・スレッドを割り当てるのを待ちます。このため、クライアントと Mobile Link 間で少量のデータ転送が必要となり、接続を確立するためのオーバヘッドが生じます (たとえば、TCP/IP ストリームを使用している場合、TCP/IP ソケットを作成するためにオペレーティング・システムが必要とする時間など)。このオーバヘッドは小さなものですが、クライアントから見れば、Mobile Link のすべてのワーカー・スレッドが他のクライアントを処理中の場合、そのワーカー・スレッドが利用可能になるまでキューで長時間を費やす可能性があります。

Mobile Link は、ワーカー・スレッド待ちのクライアントをキューに入れるため、実際には、同時に同期を試みることのできるクライアント数には制限がありません。Mobile Link は、待ち状態のクライアント 1 つについてほとんどメモリを使用しないため (32 バイト未満)、データ構造体は無数のクライアントをキューに入れることができます。クライアントは、Mobile Link への接続を作成している間、一時的にトランスポート層でキューイングする場合もあります。たとえば、TCP/IP または HTTP を使用するときに、オペレーティング・システムのポート・リスナは、待ち状態のポート接続をキューに入れます。この場合 Mobile Link は、そのポート上で最大 32,000 件の要求をキューに入れるようオペレーティング・システムに指定します。

Mobile Link は、統合データベースへの一連の接続と同時に、Mobile Link のシステム・テーブルにアクセスするための接続も管理します。同期の最初に、ワーカー・スレッドはクライアント同期情報を得るためと同期スクリプトを取り込むために、このテーブルを使用します。その他のデータベース接続は、必要に応じてワーカー・スレッドによって作成されます。接続は、同期の期間中、あるワーカー・スレッド専用となり、その後でプールに戻されます。すべての同期データはこの接続を介して送信されます。ワーカー・スレッド数を超える数のデータベース接続を指定することもできますが、一度に使用できる最大数は、1 つのワーカー・スレッドについて 1 つと、Mobile Link システム・テーブル用の 1 つを加えた数となります。

Mobile Link がデータベース接続をクローズして新しい接続をオープンするケースが 2 つあります。1 つ目のケースは、エラーが発生するときです (ただし、このデフォルトの動作は無効にできます)。2 つ目のケースは、クライアントが同期バージョンを要求したにもかかわらず、利用可能な接続の中にこの同期バージョンを使用したものがないという場合です。複数の同期バージョンがある場合、プール内の接続最大数をワーカー・スレッド数 (デフォルト数) よりも大きく設定します。その結果、Mobile Link は、別の同期バージョンが要求されるたびに接続をクローズして新しいデータベース接続をオープンする必要がなくなります。

次の項目を実行すると、新しいデータベース接続の作成に要する時間を最小限にすることができます。

- ワーカー・スレッドの使用数を減らす (Mobile Link の -w コマンドライン・スイッチを使用)。統合データベースへの最適同時接続数を超える数のワーカー・スレッドを使用しないようにしてください (スループットに合わせてワーカー・スレッドの最適数を選択する方法については、後で詳しく説明します)。

- プールされる Mobile Link データベース接続の最大数を、同期バージョンの標準的な数に Mobile Link ワーカー・スレッド数を乗算した数に設定 (Mobile Link の -cn コマンドライン・スイッチを使用)。

アップロード

クライアントがワーカー・スレッドに接続されると、クライアントは、アップロード・ストリームをワーカー・スレッドに転送します。クライアントと Mobile Link 間の接続の帯域幅とアップロード・ストリームのサイズは主に、アップロード・ストリームをワーカー・スレッドに転送するのに要する時間に影響します。

Ultra Light クライアントの場合、アップロード段階でも、アップロードが必要なデータ (つまり新しいデータ、または最後に同期してから変更が加えられたデータ) を決定するための処理時間が必要となります。これは通常、わずかな処理時間であり、ダウンロードの段階で必須とされるクライアントの処理時間をはるかに下回るものです。

ASA クライアントの場合、クライアントが Mobile Link に接続する前にアップロード・ストリームが作成されるため、アップロード・ストリームの作成に要する時間は、Mobile Link への接続前に発生します。ASA クライアントは、Ultra Light と Mobile Link が使用するフォーマットに数値データをパックするという追加のタスクがあります (数値データをリモート・クライアントと Mobile Link 間で転送するとき、あるいは Ultra Light に保存するときは、数値データのサイズを小さくするために単純なパッキング・スキームが使用されます)。

アップロード・ストリームの転送は、完全に Mobile Link のメモリ内で行う必要があります。Mobile Link ワーカー・スレッドはメモリ内のアップロード・キャッシュを共有します。したがって、ワーカー・スレッドのアップロード・ストリームの累積サイズがアップロード・キャッシュよりも大きい場合、オーバーフローがディスクに書き込まれます。アップロードとダウンロードの両方で、BLOB カラム (大きい文字型カラムを含む) 用の追加キャッシュが共有され、このキャッシュもまたオーバーフローとしてディスクに書き込まれます。Mobile Link が完全にメモリ内で実行されるようにするため、またディスク・アクセスに伴うボトルネックを回避するためには、次の項目を実行する必要があります。

- 最大アップロード・ストリームのサイズにワーカー・スレッド数を乗算した数よりも大きいアップロード・キャッシュを使用します (Mobile Link の -u コマンドライン・スイッチを使用)。

- ロー内の最大 BLOB データの 2 倍 (競合検出を使用する場合) に Mobile Link ワーカー・スレッド数を乗算した数値よりも大きい BLOB キャッシュを使用します (Mobile Link の -bc コマンドライン・スイッチを使用)。

- 他のメモリ要件に加えて、アップロードキャッシュと BLOB キャッシュに十分に対応できる物理メモリが、Mobile Link を実行するコンピュータに搭載されていることを確認します。

- Mobile Link のディスク・アクセスを完全に回避するためには、Mobile Link のログ情報をディスクに書き込まないようにします。ただし、詳細ログ記録をオンにしない限り、これが Mobile Link のパフォーマンスに影響することはありません。詳細ログ記録がオフの場合、Mobile Link は、起動情報、警告、およびエラーだけをログ記録するので、Mobile

Link を使ってログ情報をファイルに書き込むときに、進行中のディスク・アクセスはなくなります。

Mobile Link ワーカー・スレッドは、アップロード・ストリームの全体を取得した後、ロー単位でストリームを処理します。最初に、各ローの文字セットをテキスト文字列の Unicode に変換します (Windows CE のクライアントは例外で、文字列がすでに Unicode になっています。また、Windows 版以外の Mobile Link も例外です。Windows 以外のプラットフォーム上で動作する Mobile Link に Windows CE のクライアントが接続されると、逆の変換が行われます)。この変換には、ほとんど時間はかかりません。この時間は文字データの量に比例します。

次に Mobile Link ワーカー・スレッドは、データベース接続を使用して、アップロードしたローを統合データベースに適用します (定義済みのアップロード・スクリプトを使用)。このため、Mobile Link ワーカー・スレッドは、統合データベースを取り扱う他のクライアントと同じパフォーマンス上の問題に直面します。この問題としては、同時接続、トランザクションのサイズ、同時実行性の問題、およびネットワークの帯域幅 (統合データベースが Mobile Link とは別個のコンピュータにある場合) などがあります。さらに、アップロード・スクリプトに競合検出が組み込まれている場合、Mobile Link は更新を適用する前に、更新すべき各ローを統合データベースからフェッチし、競合の有無をチェックします。何らかの競合が検出された場合には、追加の処理も必要となります。したがって、競合検出を備えた更新の同期は、競合検出のない更新の同期よりも時間が長くなります。

統合データベース内でのアップロード・トランザクションが完了すると、トランザクションはコミットされ、Mobile Link ワーカー・スレッドは、受信確認をクライアントに送信します。この時点でクライアントは、修正されたデータのレコードのリセットを開始し、以前にアップロードされた変更内容が次回の同期時に再度アップロードされないようにします。同時に Mobile Link ワーカー・スレッドは、ダウンロード段階の準備を開始します。

ダウンロード

Mobile Link ワーカー・スレッドは、ダウンロードに備えて、統合データベース内でダウンロードスクリプトを実行することによって、Mobile Link クライアントで挿入、更新、または削除されるローのリストを決定します。次に、ローがデータベースからフェッチされます。アップロードの場合と同様、Mobile Link ワーカー・スレッドは、統合データベースを取り扱う他のクライアントと同じパフォーマンス上の問題に直面します。フェッチされた BLOB 値は、アップロードで使用されたものと同じ BLOB キャッシュに保持されます。文字データについては、アップロード時と逆の文字セット変換が実行されます。これらのローは、ダウンロード・ストリームの形態で Mobile Link クライアントに返送されます。フェッチ、処理、転送の処理はロー単位で行われるので、クライアントへの転送は、最初のローが統合データベースからフェッチされてから開始されます。すべてのローがダウンロードされるまで、これらの処理が交互に行われます。通常、ダウンロード段階で必要な処理は、Mobile Link ワーカー・スレッドおよび統合データベースのどちらについても、アップロード段階と比較してはるかに少なくなります。

これとは対照的に、クライアントはダウンロード段階でアップロード段階よりも多くの処理を行います。ダウンロード・ストリームの各部分が受信されるごとに、古いローの削除、新しいローの挿入、更新されたローの変更、インデックスの更新、および参照整合性のチェックが行われます。プロセッサの速度が遅いリモート・クライアントでは、この処理に相当の時間を要する場合があります。Ultra Light クライアントの場合、データベースが大

きいほど、新しいローの挿入に要するクライアント処理時間が長くなります。クライアントへのダウンロード帯域幅も、この段階で要する時間に影響を与えます。

クライアントがダウンロード・ストリームを受信、処理してコミットすれば、最終の受信確認を Mobile Link ワーカー・スレッドに送信します。Mobile Link ワーカー・スレッドは、この受信確認を受け取ると、統合データベース内のダウンロード・トランザクションをコミットします。これで、ワーカー・スレッドは別の同期で利用できるようになります。

ダウンロードでは、リモート・クライアント側で相当の処理が必要となる一方で、Mobile Link 側での処理はほとんど必要としないため、ダウンロード時間は、アップロードに比べて、クライアントの処理速度に大きく依存することになります。Mobile Link ワーカー・スレッドがダウンロード・トランザクションをコミットできるようになるには、ダウンロード・ストリームの処理が完了したというクライアントの受信確認を受け取る必要があります。

クライアントと Mobile Link 間の接続の帯域幅、ダウンロード・ストリームのサイズ、およびクライアントの処理速度は主に、ダウンロード・ストリームをクライアントに転送するのに要する時間に影響します。

より多くのクライアントに拡張

以上で、１つ同期で時間のかかる手順についての検証は終了しました。次に、複数クライアントが同時に同期を行うケースについて考えていきます。

前述のように、同期を試みるクライアントの数が Mobile Link ワーカー・スレッドの数を超えると、余分なクライアントは、ワーカー・スレッドが利用できるようになるまでキューの中で待機します。したがって、ワーカー・スレッド数が多いほど、より多くの同期を同時にアクティブにできることになります。少なくともワーカー・スレッドと同数のクライアントがあれば、同期時間の合計値は、次に示すように、各同期時間 (ワーカー・スレッドを待つ時間を除く) の合計をワーカー・スレッド数で除算した値になると推定できます。

$$\text{合計時間} = \frac{\sum_{i=1}^{\text{クライアント数}} \text{同期 i の時間}}{\text{ワーカー数}}$$

または、

$$\text{合計時間} = \frac{\text{クライアント数} \times \text{平均同期時間}}{\text{ワーカー数}}$$

これは理想化した公式であり、ワーカー・スレッド間の競合 (Mobile Link 内、および統合データベースを用いたトランザクション内の両方) やオペレーティング・システムのマルチタスキング・オーバヘッドについては無視しています。この式は、Mobile Link サーバと統合データベースが、未使用の処理機能とディスク・アクセス機能を持つ場合にのみ適用されます。この機能が追加されていなければ、ワーカー・スレッド数を増やすと (したがってデータベース接続を増やすと)、パフォーマンスが低下することになります。この問題については、以降で詳しく説明します。

潜在的なボトルネック

Mobile Link 同期のスループットをはじめとして、システムの全体パフォーマンスは通常、システム内のある一点におけるボトルネックによる制限を受けます。Mobile Link 同期については、次の項目が同期スループットを制限するボトルネックとなる可能性があります。

- 統合データベースのパフォーマンス

  Mobile Link について特に重要なのは、選択した同時にアクティブになる接続数 (すなわち Mobile Link ワーカー・スレッド数に他にアクティブな接続があればその数を加えたもの) に対して Mobile Link スクリプトを実行できる速度です。最高のスループットを得るには、同期スクリプト内での競合を避ける必要があります。

- Mobile Link と統合データベース間との通信の帯域幅

  Mobile Link と統合データベースが同じコンピュータ上で実行されている場合、あるいは別々のコンピュータ上にあっても高速のネットワークで接続されている場合には、ボトルネックの可能性はほとんどありません。

- Mobile Link を実行するコンピュータの速度

- Mobile Link ワーカー・スレッド数 (すなわち同時にアクティブになる、統合データベースへの接続数)

  スレッド数が少なくなるとデータベース接続の数も減少し、統合データベース内での競合の可能性やオペレーティング・システムのオーバヘッドも少なくなりますが、極端に少なくなると、クライアントがワーカー・スレッドの空きを待っている状態になったり、統合データベースを十分に活用できなくなる場合があります。

- クライアントと Mobile Link 間との通信の帯域幅

  広域の無線ネットワーク上の場合のように接続が遅い場合は、そのネットワークのためにクライアントと Mobile Link ワーカー・スレッドがデータの伝送を待たなければならないことがあります。

- クライアントの処理速度

  これは傾向として、アップロードよりもダウンロードのときにボトルネックとなります。ダウンロード時には、クライアント側での処理がより多く必要となるからです。

Mobile Link に固有でない潜在的ボトルネックは、このホワイト・ペーパーの適用外です。たとえば、ネットワーク帯域幅の影響や、統合データベースに関する一般的なクライアント／サーバのパフォーマンスは対象外とします。

パフォーマンステスト

このテストは、Mobile Link のパフォーマンス特性を評価することによって、状況に応じて Mobile Link がどのように動作するかをユーザが考察できるようになることを目的としています。特に次の項目について決定してください。

- 多数のクライアントが同時に同期する場合の Mobile Link のパフォーマンス

- スループットの最大化に最適な Mobile Link ワーカー・スレッド数

- 同時に同期するクライアント数の変化による影響
- 各同期で転送されるデータ量の変化による影響
- 複数プロセッサを利用する Mobile Link の能力
- Mobile Link サーバのハードウェア要件 (統合データベースの要件との比較)

結果を理解しやすくするために、テスト方法として次の原則を採用しました。

- 一度に 1 つのパラメータだけを変更する

  Mobile Link 同期のスループットに影響する変動要因は多く存在するため、できるだけ一度に 1 つのパラメータだけを変更するようにしました。ほとんどの場合、最適なスループットが達成されるまで 1 つのパラメータだけを変更し、達成されるとそのパラメータを固定して、次に別のパラメータに変更を加えています。

- Mobile Link か統合データベースまたはその両方にストレスを加える

  Mobile Link のパフォーマンスに集中するため、Mobile Link に確実にストレスが加わるようにしました。通常、Mobile Link と統合データベースを同じサーバ・コンピュータ上で実行し、クライアントを別のコンピュータ上に配置し、また、同時に行う同期の数はサーバの CPU 使用率を 100% に維持できる数としました。Mobile Link と統合データベースは同じコンピュータ上で実行したので、それぞれの相対的な CPU 使用率を用いて関連するハードウェア要件を推測できるようになりました。

- 単純さを保つ

  Mobile Link 固有のパフォーマンスに集中するため、同期スクリプト、クライアント・アプリケーション、およびデータベース・スキーマはすべて極めて単純なものとしました。

次の項で、パフォーマンス・テストの特定の側面について説明します。

データベース・スキーマ

すべてのテストは、単一テーブル上のデータを同期することを基本にしています。カラム型を選択し、いくつかの一般的なデータ型を使用できるようにしました。次の SQL 文でテーブル定義を示します。

```
CREATE TABLE Purchase (
    emp_id       INT           NOT NULL,
    purch_id     INT           NOT NULL,
    cust_id      INT           NOT NULL,
    cost         NUMERIC(30,6) NOT NULL,
    order_date   TIMESTAMP NOT NULL,
    notes        VARCHAR(64),
    PRIMARY KEY ( emp_id, purch_id )
)
```

2 カラムのプライマリ・キーを使用しているのは、リモート・クライアント間でデータを分割しやすくするためです。最初のカラムは、どのリモート・クライアントにローが関連付けされているかを示します。統合データベース内の競合が複雑になるのを防ぐために、クライアント間でのデータの共有はありません。

値

Purchase テーブルの記入に使用するデータ値は、クライアントまたは統合データベースのいずれかで生成されます。値の生成アルゴリズムでは、大きな値を整数データに使用するようにしているため、クライアントとの間でデータを転送するときに、パッキング・スキームはデータを縮小しません。cost の開始値は、すべてのローについて 123456789.12 であり、更新するごとに cost を 1 ずつ増分します。文字列カラムに使用するすべての値は、正確には 64 文字の長さです。これは、各ローに一定のバイト量が転送されるようにするためです。このケースでは、転送される各ローは 92 バイトになりました。

タイミング・フレームワーク

パフォーマンス・テストを実行するため、Mobile Link 同期のタイミング設定のためのフレームワークを開発しました。フレームワークのコンポーネントは次のとおりです。

- 統合データベース内の特別テーブル

  実行回数、実行ごとのクライアント数、各クライアントが実行すべき内容など、実行するテストに関する情報を保持するテーブルがあります。これとは別に、同期ごとに 4 つの時刻を保持するタイミング情報のテーブルがあります。これは、クライアントと統合データベースの両方について記録される開始時刻と終了時刻の 4 つです。また、別のテーブルを使用して、各クライアントにダウンロードされる次のデータを記録しています。さらに、実行情報とタイミングに関連するいくつかのテンポラリ・テーブルとビューもあります。

- Mobile Link 同期スクリプト

  複数の同期バージョン、またはスクリプトのセットを採用しています。2 つのバージョンによってセットアップを制御し、また別の 2 つはタイミング設定の同期用です (更新のアップロード時に競合検出がある場合とない場合)。さらに、クライアントが実行終了時にタイミングを統合データベースに送信するためのバージョンが 1 つあります。タイミング設定の同期には、次のスクリプトが定義されています。

  begin_synchronization - サーバの開始時刻を記録します。
  end_synchronization - サーバの終了時刻を記録します。
  download_cursor - ダウンロードするローの数を制御します。
  upload_cursor - アップロードを適用する場所を定義します。
  old_row_cursor - 更新を同期するときの競合を検出します。
  new_row_cursor - 更新を同期するときの競合を検出します。
  resolve_conflict - 競合を検出した場合に呼び出されます (今回のテストでは使用していません)。

- クライアントのアプリケーション

  多数のクライアントが同時に同期を実行する Mobile Link にストレスを加えるためには、複数のインスタンスを 1 つまたは複数のコンピュータ上で実行できる、小さな効率のよいクライアント・プログラムが必要となります。クライアントの各インスタンスの占有するメモリ領域を少なくするために、Ultra Light クライアントの使用を選択しました。また、効率を高めるために、Ultra Light の ESQL インタフェースを使用し、C でクライアントを実装することを選択しました。ディスク・アクセスによってクライアントの速度が遅くならないようにするために、特別バージョンの Ultra Light ランタイム・ライブラリを使用しました。これは、Ultra Light データベース用にファイル・ベースの永続的な

記憶領域を使用していませんでした。マルチプロセッシングを容易にするために、自身のインスタンスを複数生成できる Windows NT コンソール・プログラムとなるようにクライアントを選択しました。最初のインスタンスはマスタ・プロセスとして使用され、これが、クライアントとして動作する子プロセスを生成します。

- スーパーバイザのアプリケーション
  これは、クライアントが複数のコンピュータ上で実行している場合にのみ使用されます。つまり、各コンピュータ上で動作するクライアントを調整します。

すべてのクライアントは、「ゲート」を使用することによって同期されます。ゲートでは、レースのスターティングゲートのように、各クライアントは、他のすべてのクライアントが同じ地点に到達するのを待ってから先に進みます。効率を向上するために、ゲートの実装はオペレーティング・システムのプリミティブを使用しています (同一のコンピュータ上のクライアントを待つ場合には、Win32 イベント・オブジェクトを使用し、複数のコンピュータにまたがって待つ場合には、スーパーバイザに対して Win 32 の名前付きパイプを使用)。ゲートを実装することにより大幅に効率が向上します。たとえば、10 または 15 台のコンピュータに 1,000 のクライアントが存在する場合、ゲートをくぐった後、1 ～ 2 秒の間にすべてのクライアントが開始されます。

クライアントは、タイミング設定の同期の前後にゲートを使用します。同期前のゲートにより、すべてのクライアントが同時に同期の開始を試みるようにしています。また同期後のゲートにより、すべてのクライアントの同期が完了するまで他のクライアントの処理が行われないようにしています。

タイミングを設定した同期について、次の手順が実行されます。

同期のタイミング設定

	クライアント	Mobile Link
	1. 同期の準備をする。	
⌛	2. 他のすべてのクライアントが揃うまで待つ (「ゲート」を利用)。	
🕓	3. クライアントの開始時刻を記録する。	
	4. ULSynchronize() を介して同期を開始する。	
🕓	5.	サーバの開始時刻を記録する (begin_synchronization スクリプトを使用)。
	6. 同期を実行する。	同期を実行する。
🕓	7.	サーバ終了時刻を記録する (end_synchronization スクリプトを使用)。
🕓	8. クライアント終了時刻を記録する。	
⌛	9. 他のすべてのクライアントが揃うまで待つ (「ゲート」を利用)。	

表 1 : 同期のタイミング設定の手順

上の表のアイコンは、タイミングに関連付けられた追加手順を表しています。これらは、タイムスタンプの記録 (で表示)、または他のすべてのクライアントをゲートで待機する (で表示) 手順を伴います。

クライアント時間はクライアントのメモリに記録され、タイミング設定の同期がすべて終了してから統合データベースに送信されます。サーバ時間は統合データベースに直接記録されます。

タイミング結果を述べる場合には、タイムスタンプ値の記録から得られる次の量を引用します。

クライアント同期時間 = クライアント終了時刻 - クライアント開始時刻

サーバ同期時間 = サーバ終了時刻 - サーバ開始時刻

総サーバ時間 = 最終サーバ終了時刻 - 初回サーバ開始時刻

$$\text{スループット} = \frac{\text{同期されたローの総数}}{\text{総サーバ時間}}$$

各クライアントが複数の同期を実行する場合、それらは同期のセットとしてグループ化されます。セット内のすべての同期は同時に開始され、それらがすべて完了するまで次のセットは開始されません。この場合には、次の式を使用して総サーバ時間を計算します。

$$\text{総サーバ時間} = \sum_{i=1}^{\text{セット数}} \text{セット} i \text{の総サーバ時間}$$

すなわち、複数セットの同時同期が存在する場合でも、ゲート間で要した時間だけを計算するということです。

テスト環境

タイミング・テストは、次のソフトウェアとハードウェア環境で実行しました。

テストしたソフトウェアは、Sybase SQL Anywhere Studio 7.0.1 です。統合データベースには Adaptive Server Anywhere(ASA) ネットワーク・データベース・サーバ、同期サーバには Mobile Link、クライアントには (前述したように)Ultra Light ESQL インタフェースを使用しました。Ultra Light ランタイム・ソフトウェアは出荷された 7.0.1 バージョンとは少し異なり、ファイル・ベースの永続的な記憶領域を使用していませんでした。つまり、クライアントの Ultra Light データベースはメモリだけを使用しているということです。特に記載しない限り、Mobile Link と ASA についてはデフォルトのオプションを使用しています。テストには TCP/IP 通信プロトコルを暗号化なしで使用しました。

Mobile Link と ASA は、Dell PowerEdge 6300/550 サーバ・コンピュータ上で実行しました。このコンピュータは、550MHz の Pentium Ⅲ Xeon プロセッサ 4 基と 512MB の RAM を搭載しています。ソフトウェアは 1 台の物理ドライブに収録され、データベースのログファイルは別の物理ドライブに収録されています。さらに、データベース・ファイルは別のドライブ・アレイに収録されています。オペレーティング・システムは Windows NT Server バージョン 4.0(Service Pack 5) です。

クライアントは、Windows NT Workstation バージョン 4.0(Service Pack 3) が動作する、10 台の Dell Optiplex Gxa コンピュータを装着したラック上で稼働しています。各コンピュータは 266MHz の Pentium Ⅱプロセッサ (PS-266) と 64MB の RAM を搭載しています。これらは、使用率インジケータ付きの Fast Ethernet(100Mbps) ハブを使用して、サーバにネットワーク接続しました。タイミングの実行中は、ネットワークを切断ました。1 つのテストについて、2 台の Dell Optiplex Gxa コンピュータを追加で使用し、クライアントを実行しました。

速度の遅いクライアントをシミュレートするために、一部のテストは、Windows NT Workstation バージョン 4.0(Service Pack 3) が動作する、15 台の Hewlett-Packard Vectra VL シリーズ 3 5/75 コンピュータを装備したラック上で稼働するクライアントで実行しました。これらはそれぞれ、75MHz Pentium プロセッサ (P-75) と 64MB の RAM を搭載しています。これらは、Ethernet(10Mbps) ハブを使用して、サーバにネットワーク接続し、タイミングの実行中は、ネットワークを切断しました。

実行したテスト

Mobile Link のパフォーマンスとスケーラビリティの特性を評価するために、タイミング・フレームワークを使用して次のテストを実行しました。

- ワーカー・スレッド数を変化させて、高速なクライアントと低速なクライアントの両方についてスループットを最大化する最適なワーカー・スレッド数を求める。
- 同時に同期するクライアント数を変化させて、クライアントの増加につれてスループットが低下するかどうかを調べる。
- 同期のサイズを変化させて、スループットへの影響を調べる。
- サーバ上で利用可能なプロセッサ数を変化させて、Mobile Link と ASA が複数プロセッサを効率よく利用できるかどうかを調べる。

それぞれのケースについて、次の 4 種類の同期を測定しました。

- 新しいデータ (挿入) を中央データベースからダウンロードする
- 更新をリモート・データベースからアップロードする (競合検出を有効にする)
- 削除をリモート・データベースからアップロードする
- 挿入をリモート・データベースからアップロードする

タイミングの実行はすべて同じ手順に従います。まず、新規の統合データベースを作成し、ダウンロードに十分なデータを記入します。次に、各クライアントが空の同期を実行し、Mobile Link ユーザ・テーブルにクライアント名を設定し、Mobile Link が正しい同期バージョンとのすべてのデータベース接続を確立できるようにします。その後、タイミング設定の同期が、ダウンロード、更新、削除、挿入という順序で実行されます。これは、4 種類の同期を実行することになります。この順序で実行すれば、統合データベースは必ず開始時と同じローの数で終了するので、Ultra Light データベースにダウンロードした数以上のローは存在しないことになります。

結果と分析など、このテストの詳細については以降の項で説明します。

テスト 1 : Mobile Link ワーカー・スレッドの最適な数

このテストでは、同時に同期を実行する 1,000 のクライアントを選択し、Mobile Link ワーカー・スレッド数 (すなわち同時にアクティブになる、統合データベースへの接続数) に変更を加えました。このように多数のクライアントを選択したのは、Mobile Link に確実にストレスを加え、クライアントを使い切る前にスループットが減少する点を見つけるようにするためです。次の条件を一定に保ちました。

- 10 台の P2-266 コンピュータ上で 1,000 のクライアントが稼働。
- クライアントの 1 回の同期についてローの数が 1,000(92 バイト／ロー)。
- 合計で 1,000 の同期。

結果を次のグラフと表に示します。

スループット対ワーカー・スレッド数 (高速なクライアントの場合)

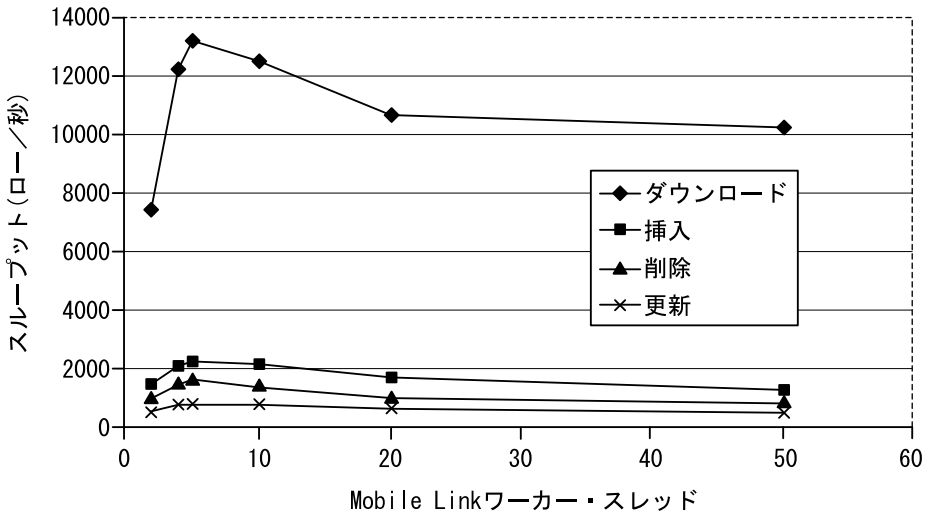

図 1 : スループット対ワーカー・スレッド数 (1,000 の高速なクライアントがそれぞれ 92 バイトの 1,000 のローを同期する場合)

ワーカー	ダウンロード	挿入	削除	更新
2	7442	1579	1196	650
4	12064	2135	1651	911
5	13069	2223	1755	958
10	12355	2176	1653	953
20	10563	1745	1065	854
50	10112	1291	979	691

表 2 : スループット (ロー／秒) 対 Mobile Link ワーカー・スレッド数 (1,000 の高速なクライアントがそれぞれ 92 バイトの 1,000 ローを同期する場合)

このテストでは、すべての種類の同期について、ワーカー・スレッド数が 5 の場合に最高のスループットが得られました。1,000 の高速なクライアントが同時に同期を試みる場合、10 以上のワーカー・スレッドを使用しても利点はありません。ワーカー・スレッド数が増加するにつれてパフォーマンスは低下します。ワーカー・スレッド数の増加に伴ってパフォーマンスが低下する原因として、最も可能性の高い 2 つの原因を次に示します。

- サーバ・プロセッサまたはディスク・スペースの飽和

  処理能力が限界でこれ以上利用できない場合に、ワーカー・スレッドを追加すると、オペレーティング・システムのマルチタスキングのためのオーバヘッドが増加し、またハードウェアの競合が増大することになります。このテストでは、4 基のプロセッサすべての CPU 使用率が、基本的にワーカー・スレッド数が 5 以上について 100% となるので、これ以上ワーカー・スレッドを増やしても、使用できる余分なリソースがありませんでした。

- 統合データベース内の競合

  ある接続が別の接続によってブロックされる可能性がワーカー・スレッド数を増加させ、このブロックされた接続によってパフォーマンスが低下します。このテストでは、各クライアントが別々のデータにアクセスできるように設定したため、データベースの競合がパフォーマンスを制限することはありませんでした。ただし、ほとんどのアプリケーションでは、このように設定することは不可能であるため、統合データベース内の競合がパフォーマンスのボトルネックとなる可能性があります。

ワーカー・スレッド数が 5 の場合、同期の 0.5% だけが同時にアクティブになります。残りのクライアントはキュー内でワーカー・スレッドを待っているか、すでに同期を終了したかのいずれかです。クライアントから見れば、クライアントがキューの最後に近いほど待ち時間が長くなるため、同期に要する時間が長くなります。ただし、スループットを最大化すれば、キューの移動も最速となるため、平均クライアント時間は最短になります。

同期のいろいろな種類についての相対的なスループットにも留意してください。ダウンロードはアップロードに比べて大幅に速く、またアップロードの中では挿入が最も速く、更新が最も遅くなります。その他のテストで、競合検出を無効にした更新の場合、更新は挿入とほぼ同じ速度で同期されることがわかりました。これは、競合を検出するには、更新する各ローを統合データベースからフェッチし、最後の同期から値が変更されているかどうかを調べる必要があるためです。

10 台の P2-266 コンピュータ上で動作するクライアントは、Mobile Link と ASA を飽和させるのに十分であったと考えられます。これを確認するために、クライアントとして 12 台と 8 台の P2-266 コンピュータを使用していくつかのテストを繰り返したところ、スループットが大きく異なることはありませんでした。

したがって、クライアントが十分に高速で、各 Mobile Link ワーカー・スレッドをビジー状態に維持できる場合は、少数のスレッドだけを使用するようにしてください。この場合には、デフォルトの 5 を選択してください。

ただし、クライアントが十分に高速ではなく、各 Mobile Link ワーカー・スレッドをビジー状態に維持できない場合は、ワーカー・スレッドの数を増やすとスループットが向上する場合があります。これをテストするために、前述のテストのいくつかを速度の遅いコンピュータ (15 台の P-75 コンピュータ) で繰り返しました。スループットの結果を次のグラフと表に示します。

スループット対ワーカー・スレッド数 (低速なクライアントの場合)

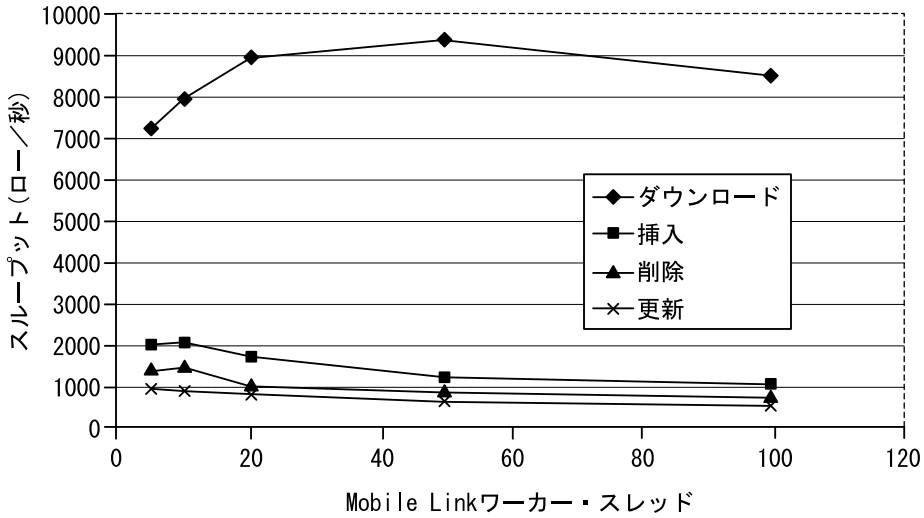

図 2 : スループット対ワーカー・スレッド数 (1,000 の低速なクライアントがそれぞれ 92 バイトの 1,000 のローを同期する場合)

ワーカー	ダウンロード	挿入	削除	更新
5	7109	1989	1447	861
10	7810	2040	1541	931
20	8813	1687	1016	820
50	9178	1196	942	629
100	8334	1034	835	542

表 3 : スループット (ロー／秒) 対 Mobile Link ワーカー・スレッド数 (1,000 の低速なクライアントがそれぞれ 92 バイトの 1,000 のローを同期する場合)

最初に、高速なクライアントの場合に得られた速度と比較してみます。予測どおり、ダウンロードの速度がクライアントの速度に最も大きく依存しています。ワーカー・スレッド数が 5 の場合の結果を比較すると、ダウンロードはほぼ半減 (-46%) しましたが、更新と挿入の速度は約 10% の低下となりました。また、削除の同期速度は約 18% 低下しました。

ワーカー・スレッド数を増加すれば、不足分の一部を補うことができます。低速なクライアントでワーカー・スレッド数が 10 の場合と、高速なクライアントでワーカー・スレッド数が 5 の場合を比較すると、アップロードの速度はほぼ同じ (-3%)、挿入の速度は 8% の低下、削除の速度は 12% の低下となりました。低速クライアントの場合のダウンロードの最高速度は、ワーカー・スレッド数が 50 の時に記録されました。これは、高速クライアントの最高速度よりも 35% 遅いものの、同じワーカー・スレッド数では、高速クライアントの速度より 10% 遅いだけでした。

したがって、低速クライアントについては、アップロードの場合に最適なワーカー・スレッド数は、ダウンロードの場合よりもはるかに少なくなります。アップロードが最速になるのは、ワーカー・スレッド数が 10 前後のときですが、ダウンロードが最速になるのは、ワーカー・スレッド数が 50 前後のときです。どちらを最適化すればいいのでしょうか ? これを

判断できるようにするために、低速クライアントが 10 および 50 のワーカー・スレッドを使用したときの経過時間を調べてみます。

時間対ワーカー・スレッド数 (低速なクライアントの場合)

動作	ワーカー・スレッド数 10 の時間	ワーカー・スレッド数 50 の時間	差	差 (%)
ダウンロード	128	109	-19	-15%
挿入	490	836	346	71%
削除	649	1061	412	64%
更新	1074	1589	515	48%

表 4 : 時間 (秒) と Mobile Link ワーカー・スレッド数の比較 (1,000 の低速なクライアントがそれぞれ 92 バイトの 1,000 のローを同期する場合)

このケースでは、ダウンロードを 19 秒早めるために 50 のワーカー・スレッドを使用すると、アップロードの各種類は 300 秒以上も低速になります。ユーザの同期のすべてまたはほとんどがダウンロードの同期でない限り、アップロードのスループットを最適化する方がよいことになります。

低速なクライアントの方が、ダウンロード速度がはるかに変動しやすいということも判明しました。通常、算出したスループットには、± 1 ～ 4% のばらつき (95% の信頼区間) があります。低速なクライアントでは、ダウンロードのばらつきは± 25% にも達することがありました。これはクライアントがダウンロードのボトルネックであるという人為的な原因によって生じたものと思われます。キューの順序によって、複数のクライアントが同一のクライアント・コンピュータからアクティブな同期を同時に実行するようになった場合、これらのクライアントは速度が遅くなり、これがダウンロードの速度に影響を及ぼします。すなわち、クライアントのキューのランダムな配置によってクライアント・コンピュータの使用率が不均一になるということです。

使用するワーカー・スレッド数を選択するときは、次の点に注意してください。

- 最高のスループットは、比較的少数のワーカー・スレッドで達成されます。サーバの CPU またはディスクがすでに飽和状態の場合、または統合データベース内の競合がボトルネックである場合、ワーカー・スレッド数の増加は逆効果となります。

- 高速なクライアントの場合、今回のテストでは、ワーカー・スレッド数がデフォルトの 5 において最高のスループットが得られました。

- 低速なクライアントについては、ワーカー・スレッド数を増加する場合があります。このケースでは、アップロードはワーカー・スレッド数が 10 で最速となりましたが、ダウンロードは 50 前後で最速でした。ユーザの同期がダウンロード中心でない限り、少ないワーカー・スレッド数を選択し、アップロードのスループットを最適化するようにしてください。

テスト 2 : クライアント数を基準にしたスケーラビリティ

次のテストでは、クライアント数を変更することにより (ただしワーカー・スレッド数は前回のテストで得られた最適値を保持する)、Mobile Link がクライアント数とともにどのように拡張されるかについて調べてみました。全体にわたって同じ量のデータが転送されるように、また各同期のサイズが一定となるように、クライアント当たりの同期の数を調整しました。たとえば、1,000 のクライアントでは、クライアント当たり 1 つの同期 (各種類について) を使用し、500 のクライアントでは、クライアント当たり 2 つの同期 (各種類について) を使用し、また 200 のクライアントでは、クライアント当たり 5 つの同期 (各種類について) を使用するというように、以下同様に使用しました。テーブル当たり 64K ローという Ultra Light の制限を超えてしまうため、20 未満のクライアント数は使用できませんでした。次の条件を一定に保ちました。

- Mobile Link ワーカー・スレッド数が 5。
- クライアントは 10 台の PS-266 コンピュータ上で稼働。
- クライアントの 1 回の同期についてローの数が 1,000(92 バイト／ロー)。
- 合計で 1,000 の同期。

結果を次のグラフと表に示します。

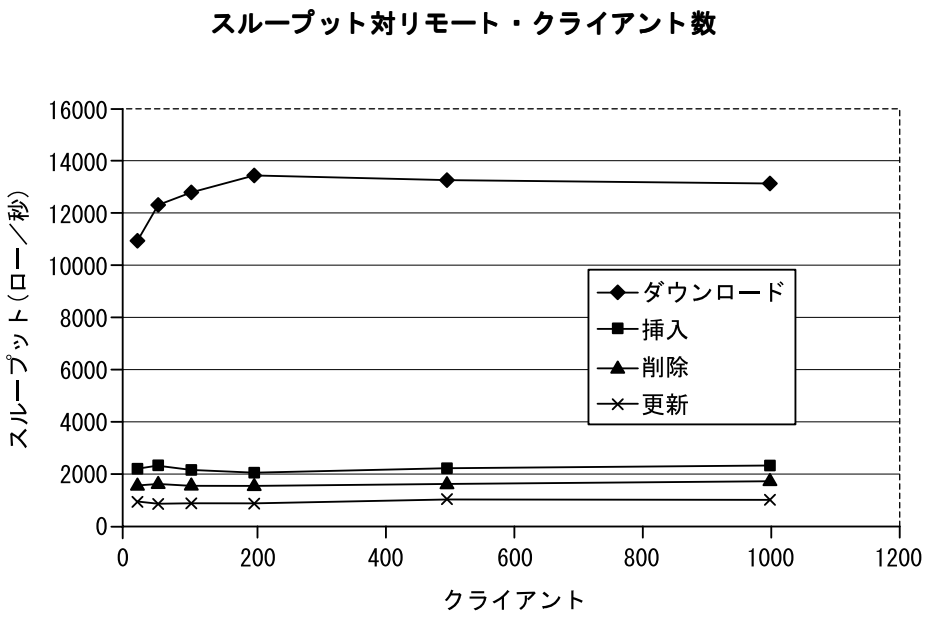

図 3 : スループット対 Ultra Light リモート・クライアント数 (Ultra Light クライアントがそれぞれ同時に 92 バイトの 1,000 のローを同期する場合)。各点は、1,000 の同期の合計を表す (すなわちクライアント数が少ないほど実行する同期セット数が多い)。

クライアント	ダウンロード	挿入	削除	更新
20	10998	2197	1623	968
50	12343	2241	1682	977
100	12802	2191	1706	977
200	13392	2167	1716	978
500	13226	2188	1731	983
1000	13069	2223	1755	958

表 5 : スループット (ロー／秒) 対 Ultra Light リモート・クライアント数 (Ultra Light クライアントがそれぞれ同時に 92 バイトの 1,000 のローを同期する場合)。各値は、1,000 の同期の合計を表す (すなわちクライアント数が少ないほど実行する同期セット数が多い)。

Mobile Link はリモート・クライアントの追加に伴って極めて良好に拡張されます。ダウンロードについて、スループットは、クライアント数 200 までは増大し、クライアント数 200 以上では一定になります (実験のばらつき範囲内)。アップロードについては、少ないクライアント数でも最大スループットが得られます。すなわち、アップロードのスループット・レートは、クライアント数 20 以上で一定になります。

この理想的なスケーラビリティから推察できることは、少数のクライアントでテストを実行することで、多数のクライアントでの結果を予測できるということです。理想的なスケーラビリティの公式を思い出してください。

$$\text{合計時間} = \frac{\sum_{i=1}^{\text{クライアント数}} \text{同期 i の時間}}{\text{ワーカー数}}$$

この方程式は、実際にはどのように機能するのでしょうか ? 前回のテストで、ワーカー・スレッド数を少数から増やすときに、クライアント数が非常に多い場合であっても合計時間は減少しないことがわかりました。ただし、ワーカー・スレッド数が一定の場合については、個々の同期時間から、すべての同期の合計時間をどの程度まで予測できるか、上記の公式を使用して調べることができます。

次の表は、クライアント数 1,000 での実際の同期時間を、クライアント数 1,000 での 1 回の同期時間、クライアント数 20 での 50 回の同期時間、およびクライアント数 20 での 1 回の同期時間という 3 つの予測時間とそれぞれ比較したものです。

合計同期時間の予測と実際

	動作	サーバ時間の合計	予測合計時間	実際の合計時間	差
クライアント数 1,000 × 同期 1 回	削除	2732	546	570	4%
	ダウンロード	371	74	77	3%
	挿入	2192	438	450	3%
	更新	4637	927	1044	13%
クライアント数 20 × 同期 50 回	削除	2843	569	570	0%
	ダウンロード	422	84	77	-9%
	挿入	2053	411	450	10%
	更新	4893	979	1044	7%
クライアント数 20 × 同期 1 回	削除	57	569	570	0%
	ダウンロード	9	90	77	-15%
	挿入	40	396	450	14%
	更新	98	981	1044	6%

表 6 : クライアント数 1,000 での実際の同期時間と、個々の同期のサーバ時間の合計から予測した合計時間をワーカー・スレッド数で除算した値との比較

クライアント数 1,000 の予測については、実際の時間との差は、オーバヘッドの量になります。これは、ワーカー・スレッドが複数の同期を切り替える時間を表すと思われます。この時間は、タイミング・フレームワークを使用した個々の同期についてサーバ上に記録されないからです。

クライアント数 20 での結果は、Mobile Link のほぼ理想的なスケーラビリティを示しています。つまり、クライアント数 1,000 での合計同期時間をクライアント数 20 でのテストから効果的に予測できます。クライアント同期の繰り返し回数によって多少のばらつきはありますが、予測は実際の値の 20% 以内に納まっています。このケースでは、クライアント数はワーカー・スレッド数の 4 倍であるため、各 Mobile Link ワーカー・スレッドは、おそらく個々の同期を 4 回実行したと考えられます。テストするクライアント数がワーカー・スレッド数の倍数でない場合、各ワーカー・スレッドの処理量が等しくならないため、予測の精度が低下する可能性があります。

少数のクライアントでのテストから、クライアント時間を推定することもできます。すべてのクライアントが同時に同期を開始した場合、最初の同期は、サーバで測定した単独の同期よりもわずかに時間が長くなります (クライアントが Mobile Link に接続するために余分の時間が必要であるため)。最後の同期は合計同期時間 (前述のように予測可能) とほぼ同じ時間がかかります。今回のテストでは、クライアント時間はこれら 2 つの極値の間で均一に配分されるので、クライアントの平均同期時間は、合計時間に単独の同期時間を加えた合計の半分であると推定できます。これは、実際の多くの環境における最悪の場合の推定値であり、クライアントがまったく同じ時刻に同期を開始するのでなければ、平均値と最大値は小さくなります。

Mobile Link を多数のクライアントで使用するときは、次の点に注意してください。

- Mobile Link は、クライアント数に伴って直線的に拡張されます。

- 少数のクライアントでのテストを使用して、多数のクライアントでの結果を予測できます。このようなテストでは、Mobile Link ワーカー・スレッド数の倍数となるクライアント数を使用するようにしてください。

テスト 3 : 各同期サイズ

このテストでは、各同期で転送するロー数を変更し、同期のサイズがパフォーマンスに与える影響について調べました。クライアント数は 200 を選択しました。これは、前回のテストで、クライアント数をこれ以上増やしてもスループット・レートが変化しないことがわかっているからです。また、200 のクライアントを使用することで、同期当たりのロー数を変更しました。ただし転送データの総量は変更のないようにしました。そのため、クライアント当たりの同期数が変更されました。たとえば、各クライアントは、5,000 ローを 1 回同期、2,500 ローを 2 回同期、1,000 ローを 5 回同期 (前回のテストと同様) というように以下同様に実行しました。次の条件を一定に保ちました。

- Mobile Link ワーカー・スレッド数が 5。
- 10 台の P2-266 コンピュータ上で 200 のクライアントが稼働。
- クライアント当たり合計 5,000 の 92 バイトのローを同期。全体で 1,000,000 のロー。

結果を次のグラフと表に示します。

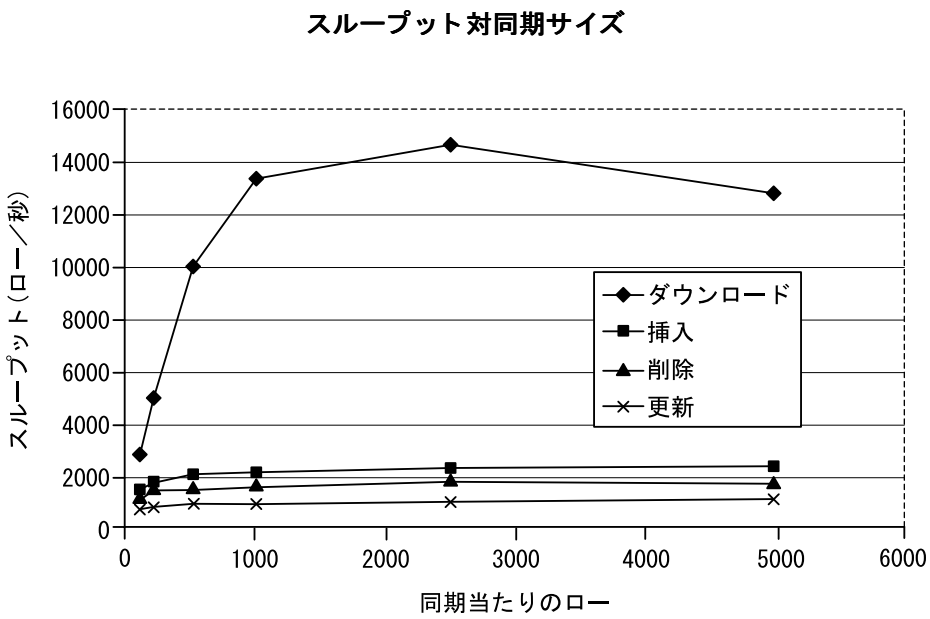

図 4 : スループット対同期当たりのロー数 (Ultra Light リモート・クライアント数 200 の場合)。各点は、92 バイトの 1,000,000 のローの同期合計を表す (同期当たりのロー数が少ないほど実行する同期セット数が多い)。

ロー／同期	ダウンロード	挿入	削除	更新
100	2890	1479	1264	828
200	5054	1846	1512	860
500	9978	2043	1595	956
1000	13392	2167	1716	978
2500	14605	2310	1741	1000
5000	12813	2262	1703	984

表 7：スループット (ロー／秒) 対同期当たりのロー数 (Ultra Light リモート・クライアント数 200 の場合)。各値は、92 バイトの 1,000,000 のローの同期合計を表す (すなわち同期当たりのロー数が少ないほど実行する同期セット数が多い)。

同期のサイズはスループットに影響します。最もサイズの小さい同期が最も遅くなりますが、速度の変化は、同期サイズが増加して 2,500 に達するにつれて徐々に減少します。これは、各同期に伴うオーバヘッドを示していると思われますが、同期サイズがさらに大きくなると次第に減少します。ダウンロードで最も影響が大きくなりますが、更新の同期についてはほとんど影響を受けません。同期当たりのロー数が 5,000 のとき、同期当たりのロー数が 2,500 のときと比較して、ダウンロードのスループットは 12% 減少しますが、アップロードについては約 2% の減少です。理由の 1 つとして、100 万のローが 5,000 ロー (1 メガバイトのおよそ半分) のアクセスを同時に受けるために、統合データベース内の競合が増加するためと考えられます。CPU の使用率が低いことが判明しており、これは統合データベースへのディスク・アクセスがボトルネックであったことを示しています。

同期サイズを検討するときは、次の点に注意してください。

- 同期ごとにオーバヘッドが存在するため、同期サイズが小さくなるほどスループット・レートが低下します。
- 大きな同期の場合、特にダウンロードのときにスループットが減少する場合があります (使用する統合データベースに依存します)。

テスト 4：並列パフォーマンス

このテストでは、Mobile Link と ASA を実行するサーバ・コンピュータのアクティブなプロセッサ数を変更して、複数プロセッサをどの程度効率よく利用できるかについて調べました。タイミングの動作と動作の間に、Dell PowerEdge 6300/550 サーバ・コンピュータ (4 基の 550MHz の Pentium Ⅲ Xeon プロセッサを搭載) をリブートし、異った数のプロセッサをオペレーティング・システムで利用できるようにしました。次の条件を一定に保ちました。

- Mobile Link ワーカー・スレッド数が 5。
- 10 台の P2-266 コンピュータ上で 200 のクライアントが稼働。
- クライアントの同期 1 回につきロー数 1,000(92 バイト／ロー)。
- 同期セット数 5(合計同期回数 1,000 回)。

結果を次のグラフと表に示します。

スループット対サーバ・プロセッサ数

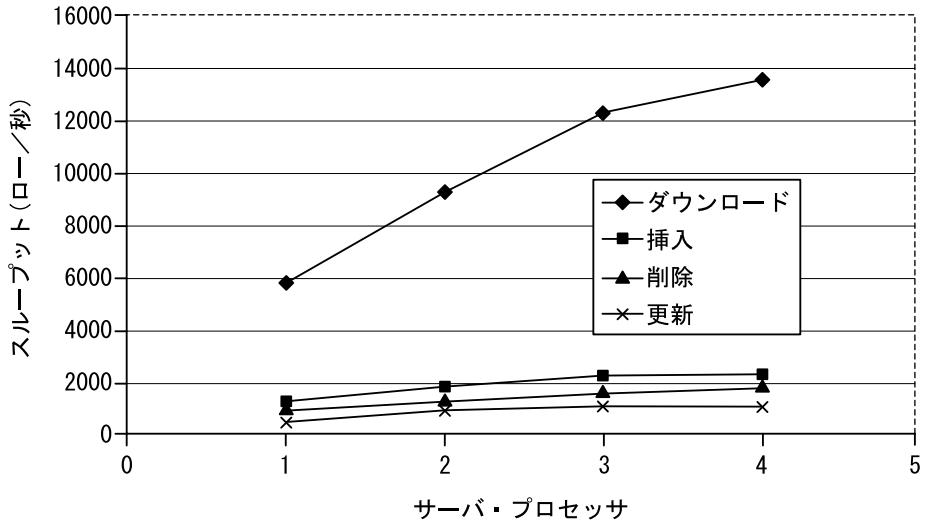

図 5 : スループット対サーバ・プロセッサ数 (Ultra Light リモート・クライアント数 200 で同期回数 5 回の場合)。各点は 1,000 回の同期合計を表す。

CPU	ダウンロード	挿入	削除	更新
1	5733	1123	852	476
2	9130	1672	1277	716
3	12078	2044	1614	928
4	13392	2167	1716	978

表 8 : スループット (ロー／秒) 対サーバ・プロセッサ数 (Ultra Light リモート・クライアント数 200 で同期回数 5 回の場合)。各点は 1,000 回の同期合計を表す。

並列パフォーマンスを評価するために、プロセッサ数が 1 の場合に得られたスループットを基準としたときの高速化の度合いを調べることにします。これを次の表に示します。

高速化対サーバ・プロセッサ数

CPU	1	2	3	4
ダウンロード	1.00	1.59	2.11	2.34
挿入	1.00	1.49	1.82	1.93
削除	1.00	1.50	1.90	2.01
更新	1.00	1.50	1.95	2.05

表 9 : 相対的な高速化対サーバ・プロセッサ数 (Mobile Link と ASA をサーバで実行)

プロセッサ数を増やすと、相対的にスループットが増加します。高速化の度合いが理想的にならない (すなわちプロセッサ数について同等になる) のは、特にプロセッサ数を 3 から 4 に増やしたときです。ダウンロードの同期は最大の高速化を示し、また更新と削除のアップロードの高速化は挿入をわずかに上回っています。

Mobile Link と ASA の両方をサーバ・コンピュータで実行したため、Mobile Link そのものの並列効率は求めることはできませんでした。同期を実行中、Mobile Link のワーカー・スレッド間には最低限の競合しか起こらないため、Mobile Link の並列パフォーマンスは非常に優れているものと考えられます。Mobile Link ワーカー・スレッドが呼び出したスクリプトを実行しているときに、Mobile Link 内の競合は、統合データベース内の競合と比較して、パフォーマンスにはほとんど影響を与えません。

Mobile Link サーバ上で複数のプロセッサを使用する場合は、次の点に注意してください。

- Mobile Link 同期と ASA がともに動作することにより、Mobile Link 同期のパフォーマンスは、プロセッサ数の増加に伴って着実に向上します。

- 同期スクリプトを実行するときに、Mobile Link 内よりも統合データベース内ではるかに競合が発生しやすくなります。

ハードウェア要件

Mobile Link の相対的なハードウェア要件を評価するために、統合データベースと同一コンピュータ上で Mobile Link を実行してストレスを加え、各プロセスにおける CPU 使用率を観察しました。Windows NT4 上で実行したため、タスク・マネージャを使用して全体の CPU 使用率を調べ、次に Mobile Link と ASA の使用率を調べました。

タイミング設定の同期中、全体の CPU 使用率は通常 98 ～ 100% でした。ただし、低速 (特にダウンロード時) のクライアントで実行した、少数クライアントによるダウンロードのテストと最小ワーカー・スレッド数の場合のテストは例外でした。また、統合データベースが定期的なチェックポイントを実行するためにトランザクションを一時停止するときに、CPU 使用率が急激に低下しました。CPU 使用率がすべてのプロセッサについて 100% 前後となったときに、Mobile Link と ASA にはストレスが加わっているということ、またボトルネックがサーバ上での処理速度であるということを確信しました。

タイミング設定の同期の実行中に全体の CPU 使用率がおよそ 100% になったときに、Mobile Link の CPU 使用率は 25 ～ 35% まで変化し、ASA の使用率は 65 ～ 75% まで変化しました。この比率は、ダウンロード、更新、削除、挿入のすべての種類の同期について観察されました。このことは、Mobile Link と ASA が別個のコンピュータ上にある場合に、Mobile Link を実行するコンピュータが、ASA を実行するコンピュータに比べて半分以下の処理能力で ASA を飽和状態にできるということを示しています。統合データベースと異なり、Mobile Link はディスク・アクセスをほとんど使用しないため (十分な RAM があり、キャッシュサイズの設定が適切な場合)、高速なディスク・アクセスについて、統合データベースと同じ要件は必要ありません。

Mobile Link のハードウェア要件を考慮する時は、次の点に注意してください。

- Mobile Link は、統合データベースと比べて、必要な処理能力は少なくて済み、またディスクの容量やパフォーマンスも大幅に少なくて済みます。

- Adaptive Server Anywhere(ASA) 統合データベースを飽和させるために Mobile Link が必要とする処理能力は、統合データベースの半分以下です。

推奨

この項では、前述の Mobile Link のパフォーマンスに関するヒントを要約し、小規模なテストを実行することによって大規模に展開したときの Mobile Link のパフォーマンスを予測する方法について助言し、また、Mobile Link に割り当てるべきハードウェア・リソースを予測します。

パフォーマンスに関するヒント

ここでは、前述のパフォーマンスに関するヒントを思い出してみましょう。次に挙げるヒントが、Mobile Link の最大パフォーマンスを引き出すのに役立ちます。

- 同期スクリプト内の競合を避けるように注意します。
- 最高のスループットが得られる、最小の Mobile Link ワーカー・スレッド数を使用します (Mobile Link の -w コマンドライン・スイッチを使用)。たとえば、高速のクライアントを使用した今回のテストでは、ワーカー・スレッド数が 5 で、最高のスループットが得られています。低速のクライアントでは、これ以上のワーカー・スレッド数が必要になります。ワーカー・スレッド数をできるだけ少なくすることで、統合データベース内の競合が起こりにくくなり、統合データベースへの接続数 (接続の確立に時間がかかる) や、キャッシュに使用するメモリも少なくて済みます。
- Mobile Link データベース接続の最大数を、標準の同期バージョン数に Mobile Link ワーカー・スレッド数を乗算した数値に設定します (Mobile Link の -cn コマンドライン・スイッチを使用)。これによって、Mobile Link のデータベース接続の終了や確立の必要性が減少します。
- アップロード・キャッシュは、アップロード・ストリームの最大サイズにワーカー・スレッド数を乗算した数値よりも大きい値を使用します (Mobile Link の -u コマンドライン・スイッチを使用)。これによって、ディスクに対するアップロード・キャッシュのオーバフローを防ぎます。
- ロー内に BLOB データがある場合、ロー内の最大 BLOB データの 2 倍にワーカー・スレッド数を乗算した数値よりも大きい BLOB キャッシュを使用すれば (Mobile Link の -bc コマンドライン・スイッチを使用)、BLOB キャッシュのディスク・アクセスを防止できます。
- Mobile Link を実行するコンピュータに十分な物理メモリがあって、アップロードと BLOB キャッシュ、およびその他のメモリ要件に対応可能であることを確認します。
- Mobile Link に十分な処理能力を確保して、必要な場合に統合データベースを飽和できるようにします。ASA 統合データベースを使用した今回のテストでは、両方にストレスを加えたときに、Mobile Link が必要とする処理能力は、ASA が必要とする処理能力の 3 分の 1 から 2 分の 1 でした。
- ビジネス・ニーズに適合した、最小限の詳細ログ記録を使用します。デフォルトでは、詳細ログ記録はオフになっており、Mobile Link はログをディスクに書き込みません。

応用範囲

本書で報告したテストで、定量的な Mobile Link のパフォーマンスをある程度は理解できたと思います。ただし、ユーザ自身の Mobile Link 設定のパフォーマンスを評価するためには、独自のスキーマ、データ、統合データベース、同期スクリプト、クライアントを使用してテストを実行する必要があります。たとえば今回のテストでは、ASA 以外の統合データベースの使用については取り扱っていません。

すでに説明したように、少数のクライアントを用いたテストによって、多数のクライアントの場合のパフォーマンスを効果的に予測することができ、これは一般的にも応用できると考えられます。この種のテストを実行する場合には、次の手順を推奨します。

1. 同期のニーズを明らかにする

   対象期間の間にデータを同期するユーザ数を把握または推定し、同時に同期を実行すると思われるユーザ数を推定する必要があります。また、アップロードおよびダウンロードされるデータの種類やそのサイズ、またデータに対して挿入／更新／削除のいずれを実行するのかなど、代表的な同期の特性を認識する必要があります。

2. 少数のクライアントによる試験的な実装を設定する

   できる限り実際の同期スクリプト、統合データベース、およびクライアント／サーバのハードウェアを使用してください。目的のクライアント・ハードウェアやネットワーク上で実際のクライアントを使用するまたはシミュレートして、代表的なデータを用いて代表的な同期を実行するクライアントを構成します。クライアント数については、試用したい Mobile Link ワーカー・スレッド数の倍数となる数を選択します。たとえば、20 のクライアント数を使用して、ワーカー・スレッド数が 5、10、20 の場合でテストを実行してください。

3. CPU の相対使用率を利用する

   試験実装で、Mobile Link と統合データベースを同一のサーバ・コンピュータ上で実行する場合があります。その場合、CPU の相対使用率を利用すれば、Mobile Link と統合データベース間でハードウェア・リソースの割り当てを決定するのに役立ちます。

4. ディスクに対して最小限の詳細ログ記録を有効にして Mobile Link を実行する

   これにより、各同期のサーバ・ベースの開始時刻と終了時刻を記録します。

5. テスト同期を実行する

   たとえば、クライアント数 20 のテストを実行するには、20 のクライアント希望者を集め、20 のデバイス上で同時に同期を開始します。

6. Mobile Link ログを使用して各同期のサーバ時間を計算する

   これにより、理想化した公式を使用して、クライアント数の多い合計時間、および平均と最大のクライアント時間を予測できます。

大規模な展開

単一の Mobile Link サーバで、数万あるいは数十万のリモート・データベースを処理できます。今回のテストでは、最大 1,000 のクライアントが同時に同期を実行するのを確認しました。同期の種類にもよりますが、これには 1 ～ 18 分の時間を要します。これは、1 時間で 3,000 ～ 60,000 のリモート・データベースが同期を実行することになり、1 日では 80,000 ～ 1,400,000 の同期となります。このために必要なハードウェア要件は厳しいものではありません。P3-550 プロセッサを 4 基搭載したコンピュータ上で、Mobile Link は CPU 時間の 3 分の 1 未満しか使用しませんでした。つまり、Mobile Link が単一の場合でも、非常に多数のリモート・データベースを処理できるということであり、また接続された統合データベースに比べてはるかに少ないハードウェアしか必要としないということです。

専用のサーバ上で動作する単一の Mobile Link では、ユーザのパフォーマンスや可用性の要件を満たさないと判断した場合は、複数の Mobile Link サーバを使用することが可能です。Mobile Link は、接続と接続の間の状態を管理していないため、複数の Mobile Link サーバを使用することができ、高可用性とスケーラビリティを実現できます。たとえば、複数の Mobile Link サーバ・コンピュータを使用して、サード・パーティ製のロードバランシング・システム (Cisco の LocalDirector や F5 の BIG-IP など) を用いることにより、クライアントに対して 1 台のサーバ・コンピュータのように見せることができます。このような設定では、ロード・バランサが定期的に各 Mobile Link サーバをポーリングし、応答性の程度から負荷を評価して、まだ動作中であるかどうかを調べます。Mobile Link の 1 つが利用不能となった場合、まだ動作中のサーバに対して新しい接続を確立するだけでフェールオーバ機能を実現できます。以上が、単一の Mobile Link を拡張する、最も簡単で効率的な方法です。多くの顧客が、スケーラブルで高可用性のデータ同期ソリューションにおいてこのアーキテクチャを利用し、成功を収めています。

複数の Mobile Link サーバを利用するもう 1 つのアーキテクチャが同期階層です。同期階層では、クライアントはセカンダリ統合データベースとの同期を実行します。このセカンダリ統合データベースは、定期的にプライマリ統合データベースと同期されています。セカンダリ統合データベースが ASA データベースであれば、両方の同期レイヤに対して Mobile Link を使用できます。同期階層は、ロード・バランサを使用することに比べてはるかに複雑です。また、おそらく単一 Mobile Link サーバ (Mobile Link を両方のレイヤに使用する場合) を介して、データの総量をプライマリ統合データベースに同期するという問題が依然として存在するため、直接スケーラビリティに対処することはできません。ただし、同期階層がユーザのインフラストラクチャやビジネス・ニーズにふさわしい場合があります。たとえば、Mobile Link サーバを地理的に分散しなければならない場合、あるいは異常に長い同期 (クライアントのプロセッサやネットワーク接続の速度が遅いことによる) にプライマリ統合データベースが拘束されることのないようにする場合です。

法的注意

Adaptive Server、iAnywhere、iAnywhere Solutions、SQL Anywhere、SQL Anywhere、Sybaseは、米国法人iAnywhere Solutions, Inc.または米国法人Sybase,Inc.とその系列会社の米国または日本における登録商標または商標です。その他の商標はすべて各社に帰属します。

Mobile Linkの技術には、Certicom,Inc.より供給を受けたコンポーネントが含まれています。これらのコンポーネントは特許によって保護されています。

本書に記載された情報、助言、推奨、ソフトウェア、文書、データ、サービス、ロゴ、商標、図版、テキスト、写真、およびその他の資料(これらすべてを"資料"と総称する)は、iAnywhere Solutions, Inc.とその供給元に帰属し、著作権や商標の法律および国際条約によって保護されています。また、これらの資料はいずれも、iAnywhere Solutions, Inc./Sybとその供給元の知的所有権の対象となるものであり、iAnywhere Solutions, Inc./Sybase,Inc.とその供給元がこれらの権利のすべてを保有するものとします。

資料のいかなる部分も、iAnywhere Solutionsの知的所有権のライセンスを付与したり、既存のライセンス契約に修正を加えることを認めるものではないものとします。

資料は無保証で提供されるものであり、いかなる保証も行われません。iAnywhere Solutionsは、資料に関するすべての陳述と保証を明示的に拒否します。これには、商業性、特定の目的への整合性、非侵害性の黙示的な保証を無制限に含みます。

iAnywhere Solutionsは、資料自体の、または資料が依拠していると思われる内容、結果、正確性、適時性、完全性に関して、いかなる理由であろうと保証や陳述を行いません。iAnywhere Solutionsは、資料が途切れていないこと、誤りがないこと、いかなる欠陥も修正されていることに関して保証や陳述を行いません。ここでは、「iAnywhere Solutions」とは、iAnywhere Solutions,Inc.とその部門、子会社、継承者、および親会社と、その従業員、パートナー、社長、代理人、および代表者と、さらに資料を提供した第三者の情報元や提供者を表します。

＊　本書は、米国iAnywhere Solutions, Inc.が作成・テストしたものを日本語に翻訳したものです。

アイエニウェア・ソリューションズ株式会社
http://www.ianywhere.jp

1009664