Mirroring with 4D 2004.4, PART II

By Kent Wilbur

TN 06-37

PartnerWare

4D は、4D アプリケーションを開発するという目的に限り、このテクニカルノートに収録されているソフトウェアの使用をパートナーに許諾します(パートナーウェア)。パートナーウェアの使用権は、4D で動作するアプリケーションに組み込まれた場合に限られており、単体での使用、あるいは 4D アプリケーション以外に組み込まれた形では認められません。ライセンスは無期限ではなく、開発者が何らかの理由でパートナーではなくなった時点で失効します。パートナーウェアを使用して開発されたアプリケーションの運用は続けられますが、開発者がパートナーではなくなった場合、パートナーウェアによる新たな開発は認められません。4D は、派生する権利を含め、パートナーウェアの全権を留保します。

The Mirroring process – Main Server side

ミラーリングを実際に制御するのはメインサーバです。メインサーバは、定期的に新しいログファイルを作成し、そのファイルをミラーサーバに転送します。Mirror_P_MirrorProcess メソッドは、ほとんどの時間、延期されているスケジュール管理メソッドです。メモリに常駐しているとはいっても、メソッド自体はサイズが小さく、必要に応じてロードされる他のメソッドに依存しています。メソッドの主要な役目は自身を延期させること、そして延期している間に環境設定が変更された場合のスケジュール再設定です。設定が変更されるとメソッドが覚醒し、次にミラーリングを開始すべき時刻を再計算するようになっています。ミラーリングのタイミングはタイムスタンプに基づいており、何らかの理由で失敗した場合には環境設定の指示に従い、一定の回数ループしながら再実行を試み、最終的に失敗が確定するとエラーメッセージをメールで送信します。

エラーの性質により、以後のミラーリングが不可能になる場合もある点に留意してください。問題を避けるため、サンプルデータベースではそのような場合、ミラーリングプロセス自体を停止するようにしました。エラーメールの送信を設定していないと、知らない間にミラーリングが中止されるかもしれないということです。

Mirror_SOAP_LHandleEvents は実際の処理が集約されているメソッドですが、それを制御しているのは Mirror_LSendLogFile メソッドです。このメソッドは最初に Mirror_SOAP_LHandleEvents をコールしてメソッドが少なくとも 1 台は稼働中であることを確認します。ミラーがまったく存在しない場合、新しいログファイルは作成されません。ログファイルが作成され

たならば、統合のみ、あるいは統合とミラーサーバのバックアップの両方を実施します。

Mirror_LHandleEvents は、ミラーサーバに対する接続を管理するメソッドです。今回のソリューションでは複数のミラーサーバも想定しているため、一部のミラーがダウンし、残りのミラーが健在である状況にも対応する必要があります。その状態は恒久的に続くのかもしれませんが、一時的なものである可能性もあり、少なくとも 1 台のミラーが稼働している限りはミラーリングを継続しなければなりません。VerifyPresent はアクティブなミラーを発見するまでミラーを探すメソッドです。ミラーが確認できた場合、前述の Mirror_LSendLogFile がコールされ、新しいログファイルが作成されます。

ログファイルが転送された後、SOAP リクエストでそのステータスが調べられます。ステータスが 2 であれば、ログファイルは統合中であり、エラーまたはステータスに 1 が返されるまでプロセスが延期されます。このような仕組みにより、SOAP 接続のタイムアウトエラーを起こさずに大きなログファイルを扱うことができます。新しいログファイルが作成されたのであれば、メソッドには閉じられたばかりのログファイルに対するパスが渡されます。ログファイルをログの途中から作成する手段はないため、複数のミラーサーバを個別に扱うことはできません。そうであっても、一時的にオフラインとなったミラーも正しい順序でログファイルセグメントを統合する必要があります。

この問題を克服するため、それぞれのミラーサーバに XML 設定ファイルを持たせることにしました。ファイルには、そのミラーが統合しなければならないログファイルセグメントのリストが記録されています。そのようなわけで、統合は閉じられたばかりのログファイルセグメントに対するフルパスを XML 設定ファイルに追加するところから始まります。

次にそれぞれのミラーマシンに置かれている XML 設定ファイルが解析され、必要なファイルが順番どおりに送信されます。統合されたファイルの名前は XML から取り除かれ、最終的には最新のログセグメントだけが残ることになります。ミラーが一時的にオフラインになった場合、複数のログファイルが統合待ちとなっているかもしれません。復帰後、統合を見送られたファイルは順番どおりに送信され、ミラーはやがて最新の状態まで更新されます。注記：複数のログファイルを連続して統合する必要があり、統合とミラーサーバのバックアップを実行する設定になっている場合、統合がすべて終わってからバックアップを開始するようになっています。

Mirror_HandleMultipleMirrorsXML は、このテクニカルノートですでに取り上げた他のメソッドに似た XML 解析メソッドです。XML の解析に関心があれば、興味深いかもしれません。

それぞれのログファイルセグメントは BLOB に収納され、SOAP 経由でミラーサーバに送信さ

れます。(注記：今回は SOAP を選択しましたが、他の方法を採用することもできるでしょう) メソッド Mirror_proxy_SOAPHandleEvents はミラーサーバの場所を特定するためにパラメータを使用している点を除けば、ほとんど Web サービスウィザードが生成したものをそのまま使用しています。

ミラーサーバが 1 台の場合(ほとんどのケースがそうだと思います)、ログセグメントがひとつ送信されるだけであり、ミラーサーバ別の XML 設定ファイルは必要ありません。

The Mirroring process – Mirrored Server side

ミラーサーバは基本的に黙って待機するだけの存在であり、これといった処理はありません。実際、プロセスを起動する必要もなく、仮にデータベースがストアドプロシージャを使用するような設計になっているのであれば、ミラーサーバはその処理を無効にしておくべきです。

メインサーバからの SOAP コールがエントリーメソッドの MIRROR_SOAP_MirrorHandleEvents に到達するとミラーサーバが仕事を開始し、処理が成功すれば 0、そうでなければエラーコードが返されます。

メソッドの基本的な内容は、データベースの名前を確認し、ログファイルを統合し、オプションでバックアップを実行するというものです。

サンプルデータベースは 4D の標準バックアップ設定ファイルの項目をいくつか利用しています。たとえばログファイルセグメントやバックアップに含めるファイルなどの保存先フォルダを特定した場合、その情報はバックアップ設定ファイルに記録されています。(注記：バックアップファイルのフォルダの場所を使用したほうが、ファイルを指定しなくてもよいため、ログファイルの場所を使用するよりも都合が良いと思います。)

Mirror_HandleBackupPreferences は必要なバックアップ設定情報を読み込むメソッドです。メソッドは書き込み禁止モードでファイルにアクセスし、4D のバックアップ設定を更新することはありません。

ログファイルセグメントには、バックアップ番号およびセグメント番号を示す合計 8 桁の ID が付与されています。メインサーバはバックアップを実行することがなく、ミラーサーバが任意でバックアップが実行できるため、ファイル名のバックアップ番号とミラーマシンにおける実際のバックアップ回数は一致しないかもしれません。メソッドは、ミラーマシンにおけるバックアップ番号と対応するようにログファイルセグメントのファイル名を変更するようになっています。

統合中に発生したエラーはエラー処理メソッドにトラップされ、エラーメッセージとともにSOAP FAULT が Mirror_SOAP_ErrorHandling によりメインサーバへ返されます。1403 から1420 までのエラー番号は 4D のバックアップモジュールにより予約されています。エラー番号-17050 から-17054 はサンプルデータベース独自のエラーコードです。それぞれのエラーメッセージの意味についてはコードを参照してください。

Performing backups on the main server

メインサーバのバックアップは実行しないでください。そのひとことに尽きます。万一バックアップを実行した場合、ミラーリングは最初からやり直しが必要です。サンプルの場合、メインサーバでバックアップを試みるとアラートが表示されるようになっています。実際にバックアップを実行するべきなのは、新しいミラーを設置するときだけです。

Performing backups on the mirroring server

4D 2004 の大きな改善点のひとつにミラーサーバでバックアップを実行してもミラーリングが壊れなくなった点が挙げられます。このメカニズムにより、サーバを停止しなくてもバックアップを他の場所に移動するなど、いろいろなセキュリティ対策が実施できるようになりました。

とはいえ使用にあたっては注意も必要です。現実的な問題として、データベースが大きければバックアップには相当な時間がかかります。バックアップ中は、ミラーサーバにログファイルを転送することは避けたいはずです。バックアップ開始時にセマフォを作成し、バックアップ終了時にそのセマフォをクリアすることにより、ログファイルの転送を控えるべきことをメインサーバに伝えることができます。前述のエラー-17053 がこのシナリオに相当します。バックアップをミラーマシン本体ではなく、メインサーバ側で制御するよう提案したのは、このようにセマフォが使用できるからです。

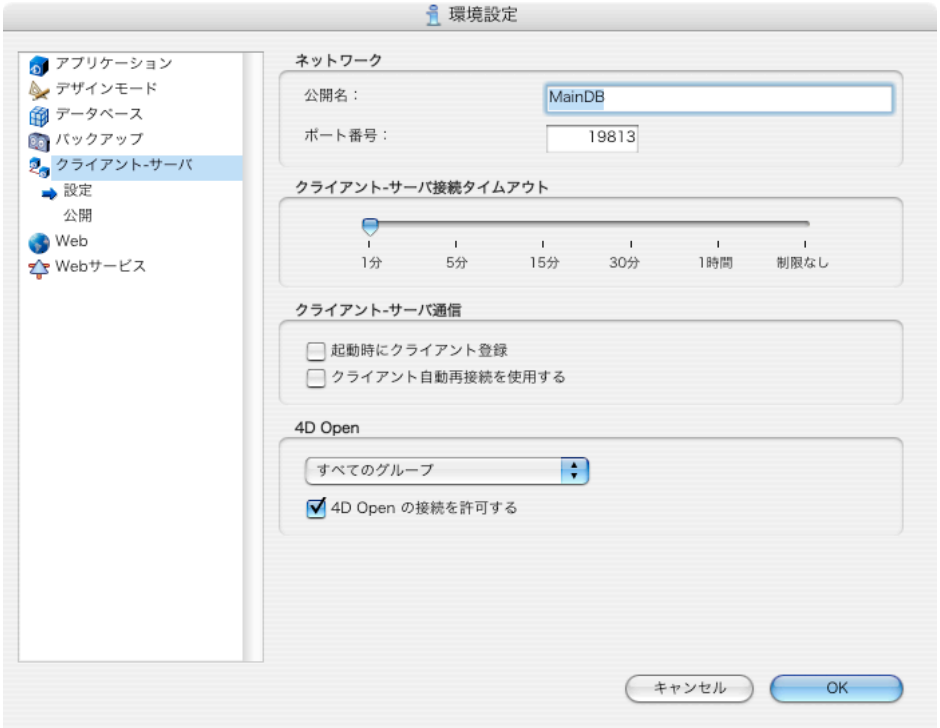
Mirror_OnBackupStartup メソッドは On Backup Startup データベースメソッドからコールされることを想定しています。

4D のバックアップモジュールは最新数世代のバックファイルだけを保持し、古いものを削除することはできますが、ログファイルセグメントを削除するという概念には対応していません。サンプルのメソッドがミラーマシン側でログファイルセグメントをリネームし、古いものは削除するようになっているのはそのためです。ログファイルセグメントは、バックアップよりも新しいものでない限り、復元を実行することもできません。

Keeping users out of the mirrored database

メインサーバおよびミラーサーバは、いずれもオンラインであり、ユーザが誤ってミラーサーバにログインする可能性がまったくないとは言い切れません。そうした事態を防ぐためにできることはいくつかあります。

サーバアプリケーションをビルドしている場合、メインとミラーで異なる公開名を設定する必要があります。そうでないとユーザは容易に混乱してしまうことでしょう。しかし非常時にはミラーマシンがメインとなることを考えると、ミラー側にメイン用のストラクチャも用意しなければなりません。



それよりも簡単な解決策は、メソッドでミラーに対するアクセスを禁止することです。外部からの接続は On Server Open Connection、On Web Authenticaion いずれかのデータベースメソッドを通過するので、それぞれのメソッドでミラーサーバに対するアクセスを識別し、拒否するようにすれば大丈夫です。

On Server Open Connection で接続を拒否する方法で唯一の欠点は、クライアントマシン側に表示されるエラーメッセージがコントロールできないことです。このレベルでの接続エラーは、インターセプトすることができません。

すべてを拒否すればよい On Server Open Connection とは異なり、SOAP による一部のメソッドコールは許可する必要があるため、On Web Authenticaion は多少注意が必要です。

さらにWeb接続も識別し、ユーザフレンドリなエラーページを送信すると良いかもしれません。

What to do about those pesky transactions

トランザクションの対応に関し、環境設定で待機時間を変更するべきことはすでに述べました。ここではクライアントマシンやストアドプロシージャで実行するコードの内容に注意を向けることにしましょう。まず第一に、ミラーリングを実行しているサーバでは、自動トランザクションを使用しないでください。次に、トランザクションを開始する前にはNew log fileが実行中ではないことを確かめてください。新しいログファイルセグメントの作成をセマフォで保護するようにしたのはそのためです。

START TRANSACTIONコマンドを使用する箇所は、すべてそのセマフォを確認するようにしてください。Customersのフォームメソッドはそのような構成にしてあります。おおまかなコードですが、大体の感じが理解できると思います。

Recovering from a disaster - small

ミラーリングシステムは、多少の軽微な問題であれば、短時間で復旧できるのが特徴です。ここでいる軽微な問題とは、一般的な基準はかなり深刻な部類に属するトラブルであり、具体的にはメインサーバが停止するような事態を指しています。データにアクセスできなくなり、バックアップを復元すれば復旧できるものの、それには何時間も必要です。

メインサーバのハードディスクから使用中で統合前のログファイルが回復できれば、復旧は簡単です。そのファイルをミラーサーバに統合するだけで済むからです。以下はサンプルデータベースの例ですが、どんなシステムでも基本的な手順は一緒です。

1) 回復したログファイルを4D Clientのインストールされたマシンにコピーします。ミラーサーバをメインサーバとして再起動する前にそのログファイルをミラーマシンで統合する必要がありますが、INTEGRATE LOG FILEコマンドはサーバのプロセスでしか実行できません。そのようなわけで、最新のログファイルはクライアントマシンにコピーし、フォルダにいれておきます。その後、ログファイルを転送また統合するメソッドを実行します。
2) 4D Clientを起動し、ミラーマシンに接続します。セキュリィ対策としてポート番号が変えられているのであれば、クライアントのカスタム接続ダイアログでポート番号を指定、あるいはサーバのポート番号を変更して再起動する必要があるかもしれません。
3) ユーザモードに切り替え、メソッドE_RestoreDataFromMirrorLogsを実行します。
4) 統合するログファイルを選択します。
5) 必要であればサーバのポート番号をメイン用のものに変更します。
6) ミラーサーバを終了します。

7) mirror の設定フォルダを削除します。
8) サーバを起動し、メイン用で設定します。これで復旧完了です。

最新のログファイルが回復できなかった場合、その分のデータはあきらめるしかありません。その代わり上記 1 から 4 の手順は飛ばしてステップ 5 から始めることができます。

Recovering from a disaster - major

ミラーリングシステムで考えられる最悪のシナリオは、メインサーバが再起不能な上、ミラーサーバのデータファイルも破損してしまっているケースです。ミラーサーバのバックアップを実行していれば、最新のバックアップから復元することができますが、それもないならば一番最初のバックアップに戻って無数のログファイルセグメントを統合してゆくことが必要です。(空き領域を空けるためにログファイルセグメントを削除したりしていなければの話です。)いずれにしても、バックアップから復元されたデータベースには、バックアップ実行後に作成されたログファイルセグメントの内容は反映されていません。次の手順で復旧することが必要です。

1) バックアップからデータベースを復元します。(最初のバックアップ、あれば最新のもの)
2) 上記で選択したバックアップよりも新しいログファイルをすべて適当なクライアントマシンにコピーし、ひとつのフォルダにまとめておきます。
3) 4D Client を起動し、復元したデータベースに接続します。
4) ユーザモードに切り替え、E_RestoreDataFromMirrorLogs メソッドを実行します。
5) 統合するログファイルを選択します。
6) 統合が完了したらバックアップを実行し、最初にミラーをセットアップしたときと同じ作業をすればシステム復旧完了です。

注記 : E_RestoreDataFromMirrorLogs は特定フォルダにあるログファイルを統合するメソッドですが、閉じられたログファイルセグメントと使用中だったログファイルセグメントを区別しません。使用中だったログファイルセグメントは個別に統合する必要があるかもしれません。

4D Client で実行中ですが、ログファイルの統合にはサーバ同士で使用されていた SOAP メソッドがそのまま使用することができます。この SOAP メソッドは、本来、ミラーサーバのデータベース名と IP アドレスを受け取るものなので、プロセス間通信を実行し、ミラーサーバに自身の情報を取得させます。変数の値は GET PROCESS VARIABLE コマンドで読み取れます。

ログファイルセグメントは、[0000-0000]という特殊なフォーマットのファイル名を有しています。複数のログファイルを連続して統合する場合、このファイル名をみて同一のバックアップ番号に対するセグメントを正しい順序で統合する必要があります。

[0000-0000]フォーマットのセグメントではない単体のログファイルの統合は、メソッド内の別の部分で処理します。

4D Server はここで自身の情報を取得し、いくつかのプロセス変数にその値を代入します。プロセスは数秒間待機した後消滅しますが、待機している間にプロセス変数の内容が前述のコードにより読み取られるようになっています。

Summary

ミラーリングシステムの設置は、いまやデータベース管理者の仕事ではなく、デベロッパにより実装が必要な項目となりました。とはいえ、以前から 4D データベースのミラー管理は管理者の手に負えないことが多く、デベロッパが関与していたことを思えば、ことさら新しいことでもないのかもしれません。

その点、新しいミラーリングコマンドは以前のバージョンにおけるミラーリングよりも柔軟性があり、管理用のインタフェースを作成することも難しくはありません。

ミラーリングができるようになったのは 2004.2 からですが、最低でも 2004.4 の使用を勧めたいと思います。いくつかのマイナーフィックスが施され、動作上の制限もいくつか取り払われているからです。

ほとんどの 4D データベースはミラーリングを実施しなくても大丈夫です。ミラーリングが必要な場合、このテクニカルノートに収録されているミラーリングコンポーネントの利用を検討してみてください。もしかしたらそれで問題が解決できるかもしれません。