ORACLE
FUSION MIDDLEWARE

Oracleホワイト・ペーパー
2014年5月

Oracleエンジニアド・システムとTuxedoで処理するミッション・クリティカルなアプリケーション

ORACLE

免責事項

下記事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル社の製品に関して記載されている機能の開発、リリース、および時期については、弊社の裁量により決定されます。

図一覧

はじめに

Oracleエンジニアド・システムには、エンタープライズ・アプリケーションをサポートするための、ハードウェアおよびソフトウェア・プラットフォームを作成する新しい方法が導入されています。これらのシステムには、物理環境と仮想環境の両方において、優れたパフォーマンス、簡素化された管理、および高い柔軟性を備えた統合プラットフォームを提供するために、連携して動作するように設計されたハードウェアとソフトウェアが組み込まれています。これまでは、アプリケーション・プラットフォームの各層が、独立して設計および構築されていましたが、Oracleエンジニアド・システムは、最初からエンタープライズ・アプリケーションのパフォーマンスを最適化するように設計されています。Oracle Exalogic、Oracle SuperCluster、およびOracle Exadataは、これらのシステムのうちの3つであり、以下の機能を備えています。

InfiniBandネットワーク

InfiniBandは、長い間、高パフォーマンスのコンピューティングと関連付けられてきました。また、完全にインターコネクトされたメッシュ・ネットワークを提供していますが、これは、高度にインテリジェントなホスト・チャネル・アダプタ（HCA）に対するネットワーク処理の大部分をオフロードしています。InfiniBand HCAは、障害が発生した場合でも、保証された手法でメッセージを順次配信することにより、ネットワーク・プロトコル・スタックを処理します。これにより、オペレーティング・システムおよびアプリケーションは、TCP/IP層をスキップできます。通常、これらの層が提供している機能を、ハードウェアが提供しているためです。アプリケーションは、ソケット・ダイレクト・プロトコルを使用して、カーネルTCP/IP層をバイパスできるため、CPUの消費量が低減され、待機時間も最小化されます。

InfiniBand HCAが提供するおもな機能の1つに、リモート・マシンのメモリに直接アクセスする機能があります。この機能は、リモート・ダイレクト・メモリ・アクセス（RDMA）と呼ばれ、あるマシン上のアプリケーションによる、別のマシン上のメモリとの直接読取りや書込みを可能にします。この処理はすべてユーザー・モードで実行できるため、カーネルを含める必要はありません。また、送信の際にもカーネル・モードに切り替える必要がないため、ネットワークの待機時間が1マイクロ秒未満に低減され、ホストCPUの消費量も大幅に低減されます。

統合ストレージ

各Oracle ExalogicおよびSuperClusterシステムには、高可用性を備えた統合ストレージ・アプライアンスが組み込まれています。このストレージ・アプライアンスは、エンタープライズ・ストレージを提供できるように最適化された、Oracle ZFSファイル・システムとハードウェアを使用して構築されます。また、NASシステムには、組込みトランザクション・ファイル・システムの更新、スナップショット機能、およびデータの整合性を向上させる機能が備えられており、格納されているデータに対して、常にエラーなしでアクセスできます。各アプライアンスにはデュアル・ヘッドが備えられており、ファイル・システムの可用性を保証すると同時に、InfiniBand経由でシステム内の他のホストに接続されます。InfiniBandコントローラは、冗長性のあるネットワーク・パスを提供しているため、あらゆるストレージに対して2つのパスが存在することになります。ドライブは、2つのヘッド間でデュアル・ポート構成であり、2つのヘッドは、お互いを監視しており、いずれかのヘッドに障害が発生した場合でも、お互いを引き継ぐことができます。

Oracle Exadataシステムは、統合Oracle Exadata Storage Serverまたはストレージ・セルに付属しており、データベース・サーバーのストレージに対して、高パフォーマンスのアクセスを実行できます。

これらのサーバー内のソフトウェアは、Smart Scan、Hybrid Columnar Compression、その他の高度なストレージ機能といった操作を実行して、データベース・サーバーからこれらの操作をオフロードし、ストレージ・サーバーとストレージ・セル間で送信する必要のあるデータ量を最小化します。このセルは、InfiniBand経由で接続されており、データは複数のセル間で自動的にミラー化されます。

大容量物理メモリ

Oracle ExalogicおよびSuperClusterシステムは、大容量の物理メモリで構成されています。システムの世代とタイプに応じて、各コンピュート・ノードでのDRAMの容量範囲は、96GB～1TBにまで拡張されています。この大容量メモリは、大容量のバッファ・キャッシュを提供するために利用でき、多数の仮想マシンをサポートします。また、極めてパフォーマンスの高いプロセス間メッセージにも利用できます。

ExalogicおよびSuperClusterシステムでのTuxedoの拡張機能

Tuxedo 11.1.1.3.0リリース以降の時点では、Tuxedoには、ExalogicおよびSuperClusterシステム・ハードウェア上でTuxedoアプリケーションのパフォーマンスを最大化するための組込み機能が備えられています。この機能は、エンジニアド・システム上でTuxedo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Tuxedoに追加された、Exalogic Elastic Cloudソフトウェアの拡張機能の一部です。これには、クラスタ環境でサービスへのリクエストまたは応答を送信するためのInfiniBand RDMAの使用、共有メモリ・メッセージを提供する大容量物理メモリの使用、およびエンジニアド・システムに関連のある大きいコア数による、ロック・メカニズムのパフォーマンスの最適化が含まれています。以下に、これらの機能の詳細について説明します。

BRIDGEバイパス

Tuxedoでは、クラスタ環境がネイティブにサポートされています。Tuxedoクラスタは、2つ以上のマシンから構成されており、これらのマシンは連携して動作することにより、ネイティブ言語のアプリケーションを実行するための、高度にスケーラブルで可用性の高いプラットフォームを提供します。多数のリソースが必要な場合は、クラスタにマシンを追加して、Tuxedoアプリケーションで使用可能なハードウェア・リソースを増やすことができます。クラスタ内のマシンは、BRIDGEと呼ばれるシステム・サーバーを使用して、TCP/IP経由でお互いと通信します。BRIDGEは、クラスタ内のあるマシンから別のマシンへのメッセージを中継します。
Tuxedoは、クラスタ環境内で動作している場合、単一のエンティティとして管理されており、クラスタ内のマシンで使用可能なすべてのサービスは、いずれかのマシンで実行されているクライアントか別のサーバーで利用できます。マシンAでDEPOSITサービスが公開された場合、そのサービスは、クラスタ内のすべてのクライアントおよびサーバーに表示されてアクセス可能になります。Tuxedoのリクエスト・ルーティング・メカニズムは、サーバーが実行されているマシンには関係なく、サービスを提供しているサーバー間で、リクエストをロードバランシングできます。

サーバーは、複数のマシン上で実行でき、マシンでサーバーの最低1つのコピーが実行されている限り、提供するサービスはすべてのマシンに対して使用可能になるため、クラスタ環境内の可用性が向上します。たとえば、A、B、およびCという名前の3台のマシンによるクラスタ内で、DEPOSITサービスを提供しているサーバーが、マシンAとBでは実行されており、Cでは実行されていないとします。障害または保守のため、マシンAが停止した場合でも、DEPOSITサービスは、マシンBとC上のすべてのクライアントとサーバーで引き続き使用可能です。マシンAが再稼働され、DEPOSITサービスを提供しているサーバーが、マシンAでブートされた場合、Tuxedoは、再度そのサーバーへのリクエストのルーティングを開始します。これにより、サービスに割り込むことなく、24時間365日の動作が

可能になります。マシンを追加すると、アプリケーションの可用性がさらに向上します。
非エンジニアド・システム内では、BRIDGEサーバーを使用して、クラスタ内の別のマシンで実行されているサーバーとの間で、リクエストを中継して、メッセージを返信できます。BRIDGEは、プロキシ・サーバーとして効率的に動作しており、別のマシン上のプロセスは、透過的にお互いと通信できます。ローカル・リクエストの標準的なフローは、以下のとおりです。

1. リクエスト・メッセージが、サーバーのSystem V IPCキュー上に配置されます
2. サーバーは、System V IPCキューからメッセージを取得し、リクエストを処理して、応答メッセージを作成します
3. サーバーからの応答メッセージは、クライアントの応答キューに配置されます

別のマシン上に配置されているサーバーへのリクエストの場合、フローは以下のとおりです

1. リクエスト・メッセージが、BRIDGEのSystem V IPCキュー上に配置されます
2. 次に、BRIDGEは、リクエスト・メッセージをTCP/IP経由で、リモート・マシン上のBRIDGEに送信します
3. リモート・マシン上のBRIDGEは、ネットワークからリクエスト・メッセージを取得し、リモート・マシン上にサーバーのSystem V IPCキューを配置します
4. リモート・サーバーは、System V IPCキューからメッセージを取得し、リクエストを処理して、応答メッセージを作成します。その後、リモートBRIDGEのSystem V IPCキュー上に応答メッセージを配置します。
5. 次に、リモートBRIDGEは、応答メッセージをTCP/IP経由で、ローカル・マシン上のBRIDGEに送信します
6. ローカル・マシン上のBRIDGEは、ネットワークからメッセージを取得し、ローカル・クライアントの応答キュー上に応答メッセージを配置します

この図は、非エンジニアド・システム上でのメッセージ・フローを示しています。

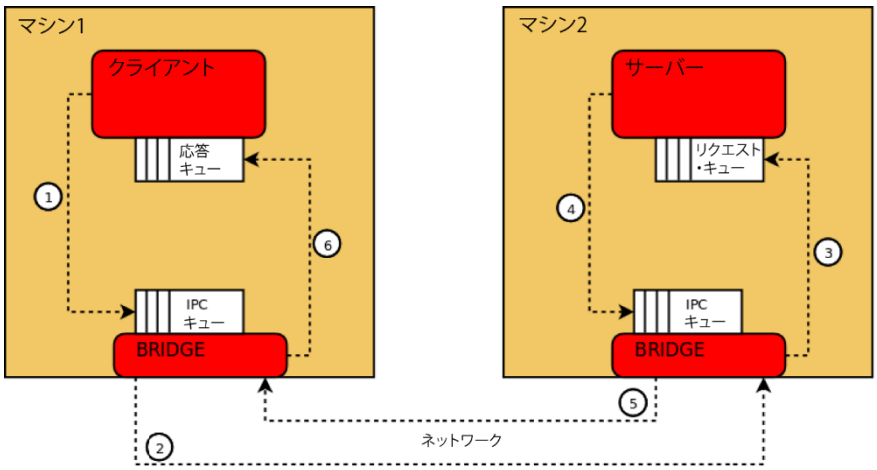

図1：BRIDGEによる通信

全プロセスは35マイクロ秒未満で実行できるため、リクエストの実行により、明らかに、ローカルで良好なパフォーマンスを実現できます。リモート・マシン上でリクエストを実行する場合、プロセスの実行には1ミリ秒以上を要します。ファイングレイン・サービスの場合、これはかなり高い値ですが、標準的なコースグレイン・サービスの場合、リモート・リクエストを作成することによる損失は、クラスタが提供する別のスケーラビリティと可用性により、相殺されます。

エンジニアド・システムでは、このパスはさらに直接的になります。Tuxedoは、InfiniBandのRDMA機能を利用して、別のマシン上のサービスを使用する影響を最小限にできます。Tuxedo11.1.1.3では、クライアントがリモート・サービスに直接アクセスする機能が導入されています。Tuxedoは、InfiniBandを利用することにより、リモート・サービスにアクセスするプロセスが、直接このプロセスと通信できるため、BRIDGEプロセスをバイパスできます。このようにエンジニアド・システムでは、TuxedoはRDMAを使用して、リクエストと応答を関連するキュー上に直接配置できます。これにより、リモート・サービスにアクセスするパスは、以下のようになります。

1. リクエスト・メッセージは、InfiniBand RDMAを使用して、リモート・サーバーのキューに送信されます
2. リモート・サーバーは、キューからメッセージを取得し、リクエストを処理して、応答メッセージを作成します。このメッセージは、RDMA経由で直接クライアントの応答キューに送信されます

この図は、エンジニアド・システム上でのメッセージ・パスを示しています。

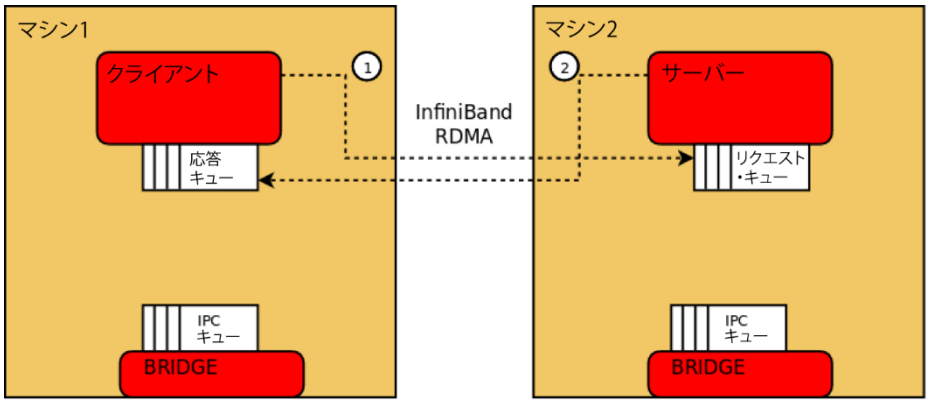

図2：エンジニアド・システム上でのBRIDGEバイパス

このパスは、ローカル・サーバーに対して実行されたリクエストにより取得したパスに非常に類似しており、BRIDGEを介して取得したパスより適切に実行されます。次の図は、エンジニアド・システム上でInfiniBandのRDMA機能を利用して取得できる、メッセージ・パフォーマンスの向上率を示しています。

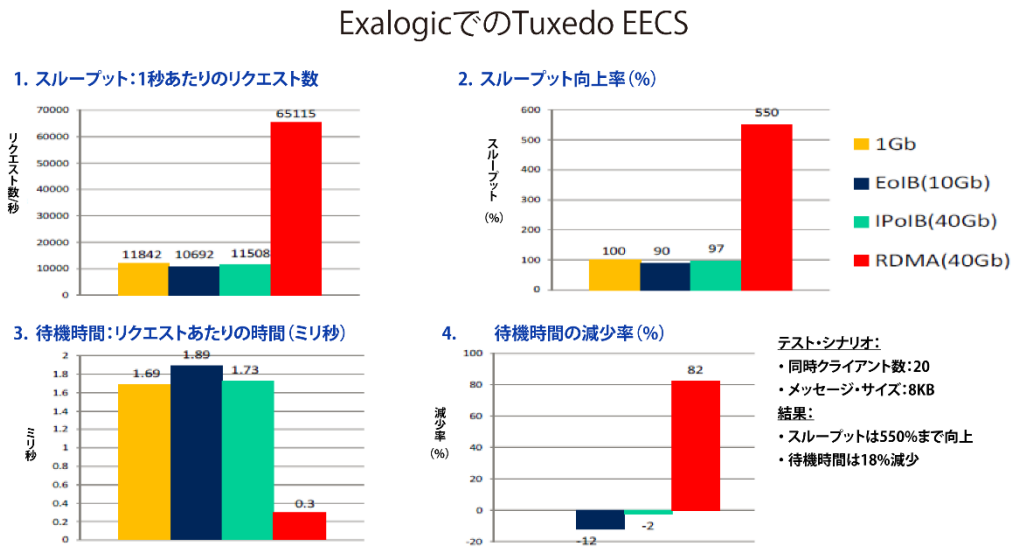

図3：BRIDGEバイパスのパフォーマンス

図に示されているように、スループットは最大550%向上し、待機時間は82%にまで低減されています。軽量なサービスの場合、この結果はパフォーマンスに大幅な影響を与えます。リモート・サービスを起動する際の損失はかなり減少しているため、1クラスタのマシン間でリクエストをロードバランシングする、Tuxedoの機能は大幅に向上しています。

ドメイン・ゲートウェイ・バイパス

大規模なTuxedoアプリケーションは、多くの場合、複数のTuxedoドメインから構成されています。これらのドメインは、サービスを共有している場合でも、独立して動作します。また、アプリケーション境界、管理境界、または地理的境界に基づいて構成されている場合もあります。Tuxedoの標準的な特徴は、ドメイン・ゲートウェイです。Tuxedoドメインは、これを使用して他のTuxedoドメインとサービスを共有できます。非エンジニアド・システムの場合、ドメイン・ゲートウェイは、BRIDGEと非常に類似した動作をします。つまり、別のドメイン内のサービスのプロキシとして機能します。おもな相違点は、BRIDGEは、クラスタ内の他のノードに対して、マシンとクラスタの状態情報を伝送するのにも使用されるのに対して、ドメイン・ゲートウェイでは、基本的に状態情報は共有されないという点です。

エンジニアド・システムに接続されているInfiniBand内で実行されているドメインの場合、Tuxedo 12.1.3リリースでは、ドメイン境界間のクライアントとサーバー間の通信にRDMAを利用するオプションが用意されており、ドメイン・ゲートウェイをバイパスすることにより、パフォーマンスが向上しています。非エンジニアド・システムの場合、リモート・ドメイン内のサービスに対するリクエストは、まずローカルのドメイン・ゲートウェイに送信されます。次に、ローカルのドメイン・ゲートウェイは、メッセージをTCP/IP経由でリモート・ドメイン・ゲートウェイに送信します。その後、サービスを提供しているリモート・ドメイン内のサーバーに、このメッセージが送信されます。サービスに対する応答は、同じルートを逆に使用します。ドメイン・ゲートウェイは、1秒間に数千のリクエストを処理できますが、追加の手順とバッファ・コピーにより、達成可能なスループットが制限されます。ドメイン・ゲートウェイをバイパスすることにより、エンジニアド・システム内

のドメイン間の通信のパフォーマンスを大幅に向上できます。

BRIDGEバイパス機能の場合と同様に、リモート・ドメイン内のクライアントとサーバー間の通信は、ドメイン・ゲートウェイによって調整および設定されます。この場合、必要なRDMAキーを交換することにより、クライアントはメッセージを直接リモート・サーバーに送信できるようになり、通信のエンド・ポイントが設定されると、ドメイン・ゲートウェイはバイパスされます。

Oracle RACの拡張サポート

エンジニアド・システム上のTuxedoは、多数の拡張機能を提供することにより、Oracle Database Real Application Clusters（Oracle RAC）をサポートしています。これらの拡張機能により、Oracle RACを利用しているTuxedoアプリケーションのパフォーマンスと可用性が向上します。

Oracle Database FANのサポート

Oracle RACには、高速アプリケーション通知（FAN）と呼ばれる機能が備えられています。エンジニアド・システム上で、Tuxedoはこの機能を使用してOracle RACデータベースのトポロジを追跡し、各インスタンスで使用可能なデータベース・サービスおよび各インスタンスのステータスを認識します。Tuxedoシステム・サーバーをOracle RACで登録して、FAN通知を受信します。この通知は、Tuxedoに対して、インスタンスが停止するか、または中止されるタイミング、稼働中のインスタンス、および各インスタンスが処理する必要のある負荷について知らせます。Tuxedoは、この情報を使用して、データベース・サービス用のデータベース接続を、FANが生成したリアルタイム・ロードバランシング（RLB）の量に比例して、使用可能なすべてのインスタンスに分散します。インスタンスが中止された場合、Tuxedoはそのインスタンスへの接続を、データベース・サービスを提供している、別のインスタンスに自動的にリダイレクトします。これにより、Oracle RACインスタンスは、そのインスタンスを使用し、Tuxedoアプリケーションに対する割込みが発生しないようにして、中止させることができます。新しいインスタンスが稼働すると、TuxedoはFANが提供する情報を使用し、受信したRLB情報に基づいて、一部のサーバーの接続を新しいインスタンスに移行します。同時に、稼働および停止したインスタンスに対して、シームレスなサポートを実行します。

トランザクションベースのアフィニティ

現在、Tuxedoは、グローバル・トランザクションに参加している、Oracleデータベース・インスタンスを追跡しています。この情報は、トランザクションの範囲が複数のドメインにまたがっている場合、他のトランザクション・コンテキストとともに伝送されます。Tuxedoは、この情報を使用して、トランザクション・アフィニティを提供します。つまり、Tuxedoは、すでにトランザクションの一部であるインスタンスに接続されている、Tuxedoサーバーにリクエストをルーティングすることを試みます。これは、グローバル・トランザクション内でのコミット処理を高速化するために、参加者の人数を減らす場合に有効です。同様に、トランザクションとその関連ロックおよびデータがすでにメモリ内に格納されているインスタンスでは、リクエストが終了するため、データベースのパフォーマンスが向上します。

密結合トランザクション

Oracleエンジニアド・システムでは、Tuxedoは、複数のドメインにまたがるトランザクションに対して、グローバル・トランザクション識別子（GTRID）の保存を試みます。以前は従属していたか、または割込みの対象であったトランザクションは、新しいGTRIDにより、リモート・ドメインで作成されます。つまり、両方のドメインが、共通のリソース・マネージャにより相互作用している場合、

リソース・マネージャは、2つのトランザクションが完全に独立していると見なします。各ドメインのトランザクションは、同じGTRIDを共有することにより、単一のグローバル分散トランザクションの一部であると見なされます。

共通のXID

Tuxedoは、エンジニアド・システム上で密結合トランザクションの範囲を越えると、共通のGTRIDを使用するだけではなく、可能な場合は、共通のブランチ修飾子を使用することを試みます。これは、ドメイン内の複数のグループ間で動作するのに対して、非エンジニアド・システム上では、トランザクションに参加している各グループは、一意のブランチ修飾子を使用します。この機能により、同一のOracleデータベース・インスタンスに接続されているトランザクション内のグループはすべて、同一のブランチ修飾子を使用します。同様に、トランザクションの範囲が複数のドメインにまたがる場合、同一のGTRIDが使用されるだけではなく、トランザクションにすでに参加しているインスタンスに対して、リクエストがルーティングできるときには、Tuxedoは以前使用されていたのと同じブランチ修飾子を使用します。

1フェーズ・コミット

トランザクション・アフィニティおよび共通のXID機能により、実際にトランザクションに参加しているリソース・マネージャが、Oracleデータベースのみであるような多数のシナリオにおいては、トランザクションが複数のグループまたはドメインにまたがる場合でも、1フェーズ・コミットが実行できます。1フェーズ・コミットを使用すると、トランザクション・ログを書き込む必要がなくなり、分散トランザクションを使用したときのアプリケーションのパフォーマンスが大幅に向上します。

共有メモリ・メッセージ・キュー

エンジニアド・システムでの別の機能として、共有メモリを利用し、リクエストおよび応答メッセージを交換するという機能があります。リクエスト・メッセージは、共有メモリ・セグメントを使用して、共有メモリ内のバッファ・プールから割り当てられます。これは、リクエストされたサービスを提供しているサーバーにメッセージを渡すために使用されたり、応答をクライアントに戻すために使用されたりします。これにより、コードを変更せずに、クライアントとサーバー間でメッセージを渡すのに必要なバッファ・コピーの数が最小化されます。クライアントは、コードを最小限変更して、リクエストが発生した際に、リクエスト・バッファを使用せずに、Tuxedoがサーバーに対して、バッファ・コピーをゼロにして、メッセージを渡すように指定できます。サーバーの応答メッセージは、常時コピーをゼロにして送信されます。通常、この処理により、Tuxedoで使用されている標準のSystem V IPCキュー・メカニズムがバイパスされるため、パフォーマンスが大幅に向上します。ただし、軽負荷の下では、System V IPCメッセージを使用して、受信プロセスを起動し、共有メモリ内に処理するメッセージがあることを通知します。この図は、標準のSystem V IPCベースのメッセージを示しています。

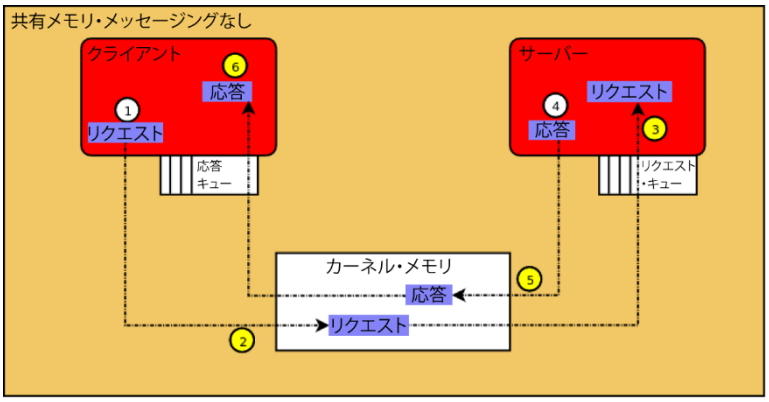

図4：IPCメッセージ・フロー

黄色で示されている手順は、メモリ内にメッセージをコピーする必要があることを示しています。小規模なメッセージの場合、このオーバーヘッドが重要になることはほとんどありませんが、大規模なメッセージの場合、リクエストをコピーしてメッセージに応答する処理がそれぞれ2回実行されるため、かなりのオーバーヘッドが発生することがあります。この図は、共有メモリを使用したメッセージ・パスを示しています。

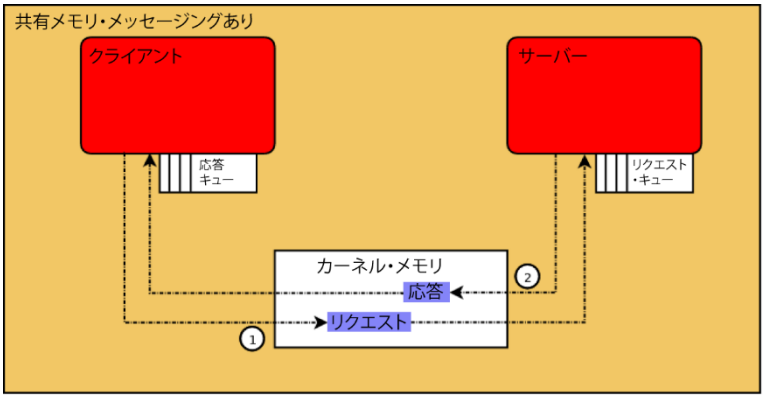

図5：共有メモリ・メッセージ・フロー

このメッセージ・パスでは、バッファ・コピーは作成されません。大規模なメッセージの場合、メッセージのパフォーマンスは、次の図に示すように、800%以上向上することがあります。

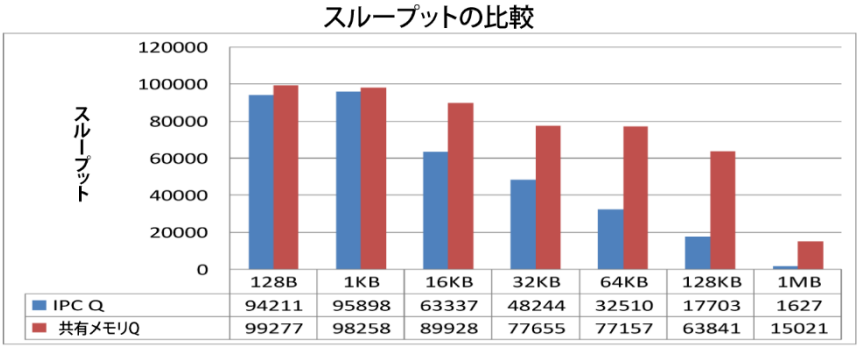

図6：共有メモリ・メッセージのスループット

セルフチューニング・ロック・メカニズム

Tuxedoは、System V共有メモリ・セグメントを大量に使用します。これには、Tuxedoの構成とアプリケーションの状態を保持している掲示板が含まれています。Tuxedoは、共有メモリ・セグメントへのアクセスを調整するために、ユーザー・モードのセマフォとカーネル・モードのSystem Vセマフォを組み合わせて使用します。ユーザー・モードのセマフォは、テストおよび設定命令を使用して実装され、オーバーヘッドは極めて低くなります。Tuxedoは、セマフォが即座に取得できない場合、SPINCOUNTで指定した一定期間、ループしてセマフォの取得を試みます。非エンジニアド・システムでは、この値はTuxedoアプリケーションの管理者が設定する必要があります。ユーザー・モードのセマフォの取得試行プロセスにおいて、SPINCOUNTで指定したループ回数の間にセマフォを取得できない場合、Tuxedoはカーネル・モードのセマフォの使用に戻り、セマフォが取得できるまでプロセスをブロックします。通常、SPINCOUNTの最適値は、共有メモリへのアプリケーションのアクセス、アプリケーションが生成するセマフォの競合の程度、およびマシン上のコアまたはプロセッサの数に基づいて、試行錯誤により決定されます。

エンジニアド・システムでは、Tuxedoは、ユーザー・モードのセマフォの取得を試行するループで消費される時間を変更できます。この場合、基準となるのは、ループ回数を超えたことが原因で、Tuxedoがカーネル・モードのセマフォを使用することが必要になった回数、および現在のCPU利用率です。これは、2つのパラメータによって制御されます。1つ目は、SPINTUNING_FACTORで、共有メモリへのアクセスを調整するために、Tuxedoがカーネル・モードのセマフォを使用する頻度の目標値を設定します。この値を100に設定した場合、100回の間に1回だけカーネル・モードのセマフォを使用するように、Tuxedoは十分な時間だけループする必要があります。もう1つのパラメータは、SPINTUNING_MINIDLECPUで、CPUのアイドル時間の最小目標値を設定します。CPUのアイドル時間が、SPINTUNING_MINIDLECPUよりも低い場合、TuxedoはSPINCOUNTの値を減少させ、アイドル時間をSPINTUNING_MINIDLECPUに戻すように試行します。SPINTUNING_MINIDLECPUが10に設定されている場合、Tuxedoは、SPINTUNING_FACTORの値に達するまで、またはCPUのアイドル時間が10%より低くなるまで、引き続きSPINCOUNTの値を増加させます。

SDPのサポート

InfiniBandベースのネットワーク・ハードウェアを使用するメリットの1つは、ソケット・ダイレクト・プロトコル（SDP）を利用することです。アプリケーションは、このプロトコルを使用して、標準のソケット・インタフェース経由でお互いと通信できるようになります。ただし、順序設定、断片化、タイムアウト、再試行などの処理が含まれる、TCP/IP関連のネットワーク処理はバイパスされます。これは、InfiniBandハードウェアが、これらの問題を処理しているためです。また、InfiniBandハードウェアは、コール元のアドレス空間から直接バッファを送信できるため、SDPはゼロ・コピー送信をサポートできます。Tuxedoアプリケーションは、SDPを利用することにより、ネットワーク操作で消費されるCPU量を低減できると同時に、ネットワーク操作全体のスループットを向上できます。SDPは、Tuxedoのすべてのネットワーク接続で使用できます。これには、BRIDGE間の通信、およびドメイン・ゲートウェイGWTDOMAINが含まれています。このゲートウェイは、他のTuxedoドメインとの通信、ワークステーションおよびJoltクライアント用、さらにはWebLogic Tuxedo Connector経由でWebLogic Serverとの通信に使用されます。

デプロイメント・トポロジ

クラスタ（複数マシン）ドメイン

エンジニアド・システムでのTuxedoで最も一般的なデプロイメント・トポロジは、クラスタとも呼ばれる、複数マシン・ドメインを利用することです。このモデルでは、アプリケーションは複数のホスト・ノード間にデプロイされ、単一のドメインとして管理されます。この明確なメリットとして、高可用性と簡素化された管理があります。アプリケーション・サーバーは、クラスタ内の各ノードで実行され、クラスタ内のすべてのクライアントまたはサーバーに対して、透過的にサービスを提供します。サービス・リクエストの応答時間を最適化するように、クラスタ内の負荷が分散されます。あるマシン上のサーバーの応答が遅く、別のマシン上のサーバーの応答は速い場合、Tuxedoは応答が速いマシンに負荷をシフトします。これは、TSAM Plusと組み合わせて、事実上リアルタイムで実行されます。この場合、リクエストをリモート・マシンにルーティングするためのネットワーク時間、サーバーにキューイングされている現在の処理、および応答をリクエスト元に返すためのネットワーク時間など、応答時間のあらゆる要素が考慮されます。

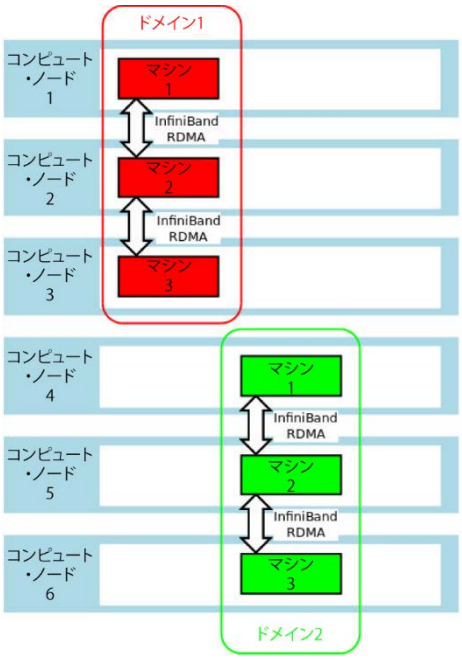

図7：クラスタ・デプロイメント

複数の単一マシン・ドメイン

Tuxedoの別の一般的なデプロイメント戦略は、複数の単一マシン・ドメインを使用することです。このシナリオでは、各ドメインは単一マシン上に存在しており、他のドメインとは独立して管理されます。単一マシンが提供できるより多い数のリソースが必要なアプリケーションの場合は、Tuxedoドメイン・ゲートウェイを使用して、別のドメインに接続できます。これはクラスタ・デプロイメントに類似していますが、いくつかの重要な相違点が存在します。最初に、各ドメインは、他のすべてのドメインから分離され、他のドメインで使用できるのは、そのドメインからエクスポートされたサービスのみです。次に、サービスの可用性は、ローカル・ドメインのみに通知されるため、

あるドメインが別のドメインのサービスを使用している際に、これらのサービスが何らかの理由で現在使用できなくなった場合、これらのサービスへのリクエストは失敗します。このデプロイメント・モデルのおもなメリットは、あるドメインから提供されたサービスにアクセスするマシンまたはドメインを制御できることです。これにより、事実上、ドメインは非共有アーキテクチャで動作しますが、構成には制限されません。

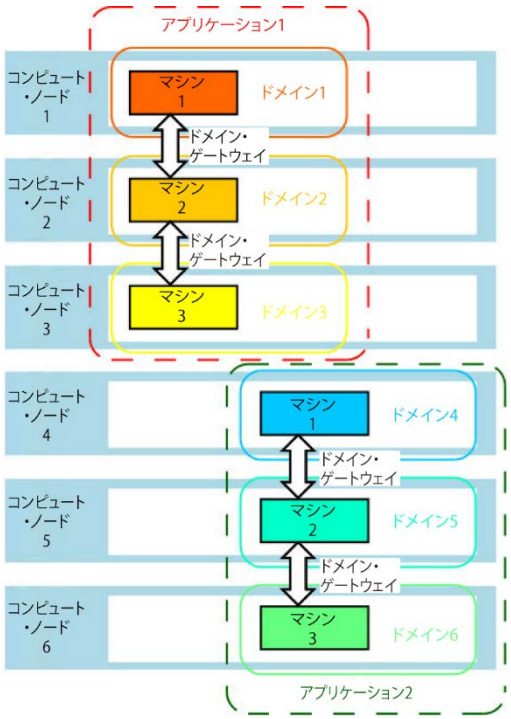

図8：複数の単一マシン・ドメイン

マシンを共有する複数クラスタ

Tuxedoアプリケーションでは、多くの場合、クラスタ構成内で使用可能なすべてのリソースが必要になることはありませんが、クラスタの高可用性は必要になることがあります。このモデルでは、複数のクラスタが使用され、マシンの一般的なセットを共有します。Tuxedoは、複数のドメインを単一マシンにデプロイできますが、いくつかの独立したアプリケーションでは、複数マシンのリソースを共有できるため、それぞれが、複数マシンでの実行により達成された高可用性のメリットを享受します。

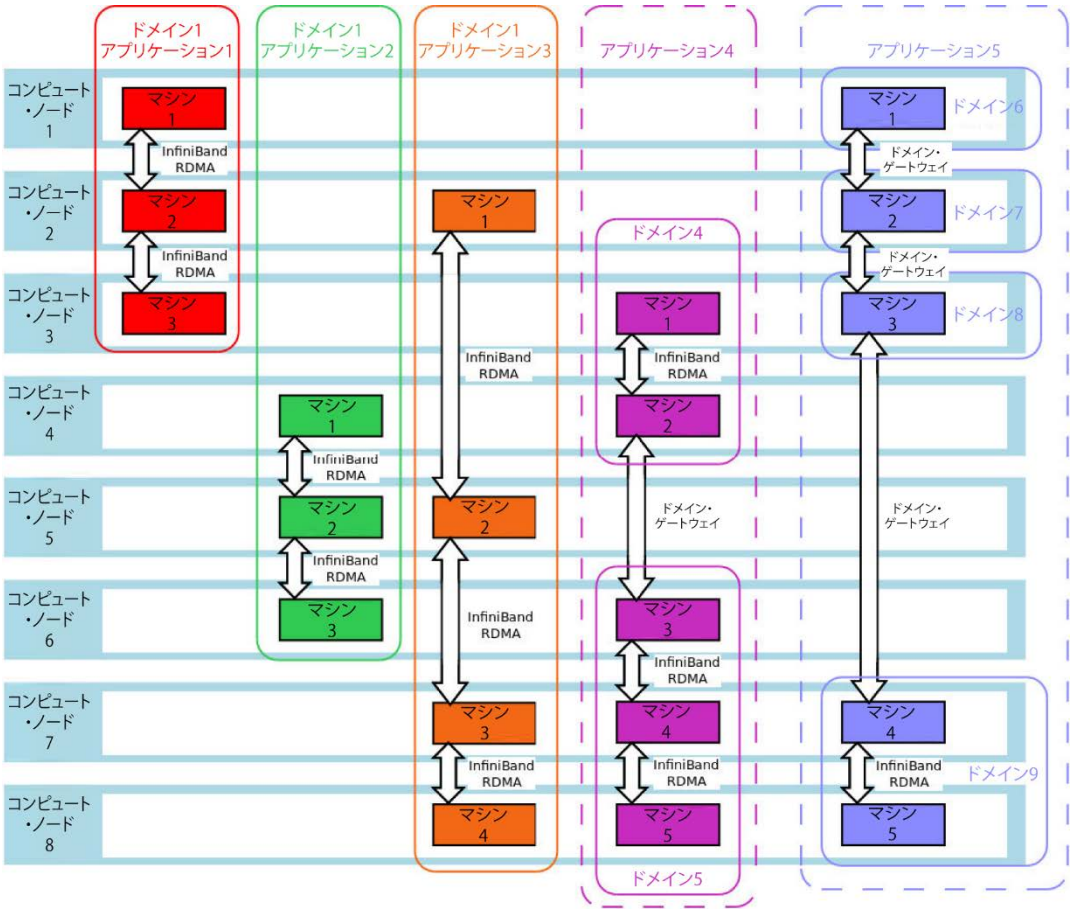

図9：ノードを共有する混合デプロイメント

上の図では、5つの異なるアプリケーションが存在しており、クラスタ・ドメイン、単一マシン・ドメイン、およびクラスタと単一マシン・ドメインの混合としてデプロイされ、すべてがExalogicのクォーターラックを共有しています。アプリケーション・サーバーは、アプリケーションに関連するドメイン内の複数のマシンにデプロイされていると仮定しており、高可用性が達成されています。

パフォーマンスと可用性を最適化するためのベスト・プラクティス

ソフトウェア冗長性

システム内で高可用性を達成するためのおもな要件の1つは、冗長性です。エンジニアド・システム・プラットフォームでは、ハードウェア冗長性が組み込まれており、シングル・ポイント障害は発生しません。アプリケーション全体で可用性を保証するには、ソフトウェアにおいても、シングル・ポイント障害が発生しないよう構成する必要があります。

サーバーの複数のコピー

Tuxedoでは、ドメイン内にアプリケーション・サーバーの複数のコピーを構成できます。この複数コピーは、パフォーマンスのスケーラビリティとアプリケーションの可用性の両方で構成できます。

Tuxedoグループ内でこれを簡単に達成するには、サーバーのMINとMAX値を、デフォルトの1以外の値に設定します。MINを5に設定すると、Tuxedoはアプリケーションの起動時に、サーバーの5つのコピーを自動的にブートします。これにより、アプリケーション・サーバーに障害が発生した場合でも、残りのサーバーは、引き続き動作してリクエストを処理できます。MAXを10に設定すると、構成に変更を加えなくても、必要に応じて追加のサーバーをブートできます。

クラスタ構成

Tuxedoのもっとも強力な機能の1つに、MPモードとも呼ばれることのある、クラスタ化機能があります。この機能では、単一のTuxedoアプリケーションを複数のマシンにまたがって配置することにより、アプリケーションに変更を加えなくても、パフォーマンスおよびアプリケーションの可用性が向上します。Tuxedoクラスタは、単一のエンティティとして管理され、すべてのクライアントとサーバーが、場所には関係なく、クラスタ内のすべてのサービスを使用できます。マシンに障害が発生した場合でも、アプリケーションのサービスを引き続き使用できます。クラスタ内に3台以上のマシンを構成すると、複数のマシンに同時に障害が発生した場合でも対応が可能になります。

推奨する最小のクラスタ構成は、3台のマシンまたはノードによる構成です。これにより、あるマシンが保守のために停止している場合でも、アプリケーション・サービスの冗長性は維持されたままです。負荷を処理するためにマシンを追加する必要がある場合、N+M台を超える構成を作成できます。ここで、Nは負荷を処理する必要があるマシンの数、Mはアプリケーションの可用性を向上させるために、冗長性が構成されるマシンの数です。この手法を使用すると、99.999%以上の可用性を比較的簡単に達成できます。

対称性

クラスタ内のマシンは、対称的に構成できます。つまり、すべてのアプリケーション・サーバーは、すべてのマシン上で実行されるように構成され、すべてのマシンは、同一バージョンのソフトウェアを実行します。指定したサーバーの2台以上のマシン上で複数のコピーが実行されている限り、アプリケーションは高可用性の状態が維持されます。対称性のある構成では、マシンは大部分が交換可能で、クラスタでのマシンの追加または削除も簡素化されているため、比較的管理しやすい構成です。

非対称性

クラスタは、非対称的にも構成できます。つまり、複数のアプリケーション・コンポーネントが、複数のマシン上で実行されているか、または複数のバージョンのソフトウェアが、クラスタ内のマシン上で実行されています。これには、複数のバージョンのオペレーティング・システム、複数のバージョンのTuxedo、および複数のバージョンのアプリケーションも含まれます。1週間に24時間、365日の可用性を必要とする高可用性アプリケーションでは、クラスタ内のマシンは、ローリング更新の手法により、ソフトウェアをアップグレードできます。あるマシンが停止すると、そのソフトウェアが更新されてから、ブートされてクラスタに戻ります。Tuxedoでは、アプリケーションのサービス・バージョンと呼ばれる機能を介して、複数バージョンのアプリケーション・コンポーネントを並行して実行できます。これにより、アプリケーションは、全体を停止させなくても、増分アップグレードを実施することが可能になります。

自動操作

アプリケーションの可用性を向上させる手法の1つは、障害からのリカバリ時間を低減させることです。この場合の障害としては、保守などの計画的な停止、またはサーバーが停止寸前であるなどの予期しない停止のいずれかです。

Tuxedo自動操作の有効化

Tuxedoには、サーバーの再起動、ネットワーク接続のフェイルオーバーなどの、さまざまな自動操作が用意されています。これらの操作は、管理者がある程度までは制御できますが、元々UBBCONFIGファイルで定義されている、Tuxedo構成の一部です。以下に、自動操作を達成するために設定する、一般的なパラメータのいくつかを示します。

サーバー再起動

アプリケーションの可用性を向上させる単純な方法として、予期せずに障害が発生したサーバーの自動再起動を有効にすることがあります。これを実行するには、UBBCONFIGファイルのserverセクションで、パラメータRESTART=YおよびMAXGEN>1を設定します。デフォルトはRESTART=Nであるため、自動再起動は実行されません。RESTART=Yを設定することに加えて、再起動がループするのを防止するために、GRACEとMAXGENパラメータを適切に設定することを推奨します。再起動がループすると、サーバーは再起動されますが、ほとんどすぐに停止してしまいます。MAXGENは、Tuxedoがアプリケーションを起動する回数を制御しており、値に3を設定すると、GRACE秒以内にアプリケーションが最大2回再起動されます。

サーバーの自動再起動を設定する別のメリットとして、リクエスト・キュー内にすでに存在するすべてのリクエストが、サーバーの再起動と同時に保守されて処理されるということがあります。RESTART=Nの場合、Tuxedoはサーバーの IPCキューを削除するため、キュー内のすべてのリクエストが最終的にはタイムアウトしてしまいます。

自動移行

場合によっては、サーバーのグループをあるマシンから別のマシンに移行するか、またはDistinguished Bulletin Board Liaison（DBBL）を別のマシンに移行することが必要になります。このプロセスは、Tuxedoの用語では、移行と呼ばれます。Tuxedo 12cでは、この移行プロセスを自動化する機能が追加されました（有効化されている場合）。DBBLの自動移行を有効にするには、UBBCONFIGファイルの*RESOURCESセクションを変更して、MASTERエントリ用のバックアップ・マシンを定義し、DBBLFAILOVERオプションをDBBLの移行が発生するまでの時間に設定します。サーバー・グループの自動移行を有効にするには、*GROUPSセクションのLMIDパラメータで、フェイルオーバーを実施するマシンを定義します。MIGRATEおよびLANオプションを設定し、UBBCONFIGファイルの*RESOURCESセクションのSGRPFAILOVERパラメータを使用して、サーバー・グループの移行が発生するまでの時間を定義します。

アプリケーションのパフォーマンスと可用性の監視

エンタープライズ・アプリケーションを管理する際のおもな問題点は、すべてのコンポーネントを監視して、品質保証契約（SLA）に適合していることを確認する必要があるということです。一般的なエンタープライズ・アプリケーションは、静的コンテンツを処理するフロントエンドWeb層（Oracle Traffic Director、Oracle HTTP Serverなど）、おもにその後面に配置されて、動的ページの作成を実施するユーザー・インタフェース層（Oracle WebLogic Serverなど）、さらにその後面に配置されて、動的ページを作成するのに必要なエンタープライズ・サービスにアクセスするサービス層（Tuxedoなど）、およびサービス層の後面に配置されて、永続データにアクセスするデータベース層（Oracle Databaseなど）から構成されます。このすべての下には、さまざまなハードウェアおよびオペレーティング・システム・プラットフォームが存在しており、Oracle Linux、およびExalogic、SuperCluster、ExadataなどのOracleエンジニアド・システムのようなすべての層をサポートしています。

TSAM Plus

Tuxedo System and Application Monitor Plus（TSAM Plus）製品は、上述の多くの問題、特にTuxedoアプリケーションのパフォーマンスに関連する問題について対処を試みることを基本としています。TSAM Plusは、CPU消費量および応答時間（Tuxedo内のアプリケーションのエンド・ツー・エンドの応答時間を含む）などの詳細なパフォーマンス・メトリックを提供します。データは、監視対象のTuxedoシステム上でリアルタイムに収集され、TSAM Plusの1つのコンポーネントであるTSAM Plus Managerにレポートされます。このコンポーネントは、データを格納し、収集されたデータにアクセスしてレポートするためのユーザー・インタフェースを提供します。データが収集されるのは、アプリケーションの実行時であるため、大容量アプリケーションでは、膨大な量のデータが生成されることがあります。データ量を制限するために、TSAM Plusには、収集されたデータの内容と収集先のコンポーネント、およびデータが収集される頻度を制御する柔軟な手段が用意されています。たとえば、サービス・データの収集時には、収集先を、単一のサーバー、いくつかのグループ・リスト内のすべてのサーバー、またはアプリケーション内のすべてのサーバーから選択できます。データは固定間隔で収集できるため、1秒に1回、または5つのデータ・ポイントのうちの1つから、実際にデータが収集されて格納される比率に設定します。通常、各TSAM Plus Managerインスタンスに対して、1秒あたり1000～5000データ・ポイントの間で収集することを推奨します。

TSAM Plusは、パフォーマンス・メトリックを収集してレポートする他に、アラート生成に使用する品質保証契約を設定できます。これらの契約は、監視データが収集されていない場合でも、チェックできます。パフォーマンス・データを収集できるのが、100のサービス呼出しのうちの1つのみの場合もありますが、それぞれがSLAに適合していることをチェックされます。SLAに適合していない場合は、運用スタッフに対して、SLAに適合していないことを通知するアラートが生成されます。そして、これ以降、SLAに適合していない状態になることを回避するために、自動修正措置をトリガーするTuxedoイベントまたはEnterprise Managerインシデントが生成されます。

Oracle Enterprise Manager

TSAM Plusには、Enterprise ManagerがTuxedoアプリケーションを構成、操作、および管理できるEnterprise Manager Cloud Controlオプションが用意されています。エージェントが、監視対象のTuxedoシステムにインストールされると、Enterprise Managerは、定期的にTuxedoアプリケーションのパフォーマンス・メトリック、アプリケーションの状態を収集して、アプリケーションの構成を管理します。収集されたメトリックは、アプリケーション全体のパフォーマンスの分析、アプリケーションとそのさまざまなコンポーネントの可用性の監視、および動的スケールアップまたはスケールダウン機能を提供するために、追加のサーバーまたはマシンを起動するなどの、自動管理操作を開始するインシデントのトリガーに使用できます。

アプリケーションの可用性を向上させる確実な方法の1つは、可能な限り操作を自動化することです。高可用性システムでは、停止時間の70%以上が人為的エラーで説明できます。Enterprise ManagerをTSAM Plus Cloud Controlとともに使用し、標準的な多数の操作手順をスクリプト化して自動化することにより、このタイプのエラーの大部分が除去できます。このスクリプトは、計画保守操作などのように定期的に実行するようスケジューリングするか、またはスケジューリングされていない定期保守の場合は手動で開始できます。さらに、SLAのパフォーマンスや可用性に適合していないなどの問題が発見された場合には自動的にトリガーできます。

Enterprise Managerは、ハードウェア、オペレーティング・システム、ネットワーク、データベース、アプリケーン・サーバーからWeb層にわたるスタック全体で、すべての監視および管理を一元的に実行しています。これにより、運用スタッフは、その全体について把握でき、運用上の状況に対処

する際には、情報に基づいた決定ができるようになります。これは、エンタープライズ・アプリケーションの可用性をさらに向上させるために有効です。

ストレージの可用性

ストレージは、大部分のエンタープライズ・アプリケーションにおいて、重要なリソースです。ストレージの使用範囲は、ローカルに接続されたハード・ドライブまたはSSDから、SANやデータベースにまで及んでいます。ストレージ・システムを設計する際に重要な考慮事項の1つは、重要なデータを保持する際のストレージの可用性です。高可用性を達成するためには、最低2つのデバイス上にデータを格納し、その両方のコピーに影響を与えるシングル・ポイント障害が発生しないようにする必要があります。つまり、コントローラ・レベル、ネットワーク・レベル、およびソフトウェア・レベルにおいて、ストレージへの冗長性のあるパスが存在する必要があります。

ファイル・システム・ストレージの場合、Oracleエンジニアド・システムでは、ZFS Storage Appliance（ZFSSA）が使用されます。エンジニアド・システム内の各コンピュート・ノードには、ローカルのSSDが備えられていますが、これらのドライブはシングル・ポート構成であるため、その目的はおもに、オペレーティング・システム・ファイル、その他の非アプリケーション固有データ、または重要なデータを保持することです。ZFSSAは、デュアル・ポート構成のJBODを共有する2つのヘッド・ノードから構成されています。アプライアンスで使用されるZFSファイル・システムでは、すべてのデータとメタデータのチェックサムを計算することにより、高レベルのデータ整合性が維持されています。また、RAID-Zのような技術を使用して、データの複数のコピーを保守することにより、高レベルの可用性も維持されています。2つのヘッド・ノードは、お互いを監視しており、障害が発生した場合には、パッシブ・ヘッドが、障害の発生したアクティブ・ヘッドから操作を引き継ぐことができます。各ヘッドは、InfiniBand経由でエンジニアド・システムの他の部分に接続されており、パスで障害が発生した場合にアクセスが欠落しないように、デュアル・ネットワーク・パスを提供しています。このすべてにより、高可用性を備えたファイル・ストアが実現されます。

エンジニアド・システム内のデータに対する別のおもなストアとして、Oracle Databaseがありますが、これは多くの場合Exadataシステムにデプロイされて、優れたパフォーマンスと高可用性を実現しています。さらに、Exadataシステムは、データベースをExalogicおよびSuperClusterシステムに接続するのに使用できる、InfiniBandネットワークを採用しています。Tuxedoは、ソケット・ダイレクト・プロトコル（SDP）を使用して、エンジニアド・システム上のOracle Databaseにアクセスすることにより、データベース・アクセスのパフォーマンスを大幅に向上しています。

トランザクション・ログ

Tuxedoアプリケーションでもっとも重要なファイルの1つに、トランザクション・ログがあります。このファイルは、部分的にコミットされたトランザクションのレコードを保守するのに使用されます。トランザクションがコミットされることが決定されると、この決定を記録するトランザクション・ログ内にレコードが配置されます。分散トランザクションに含まれているすべてのリソース・マネージャが、それぞれのブランチをコミットする前に障害が発生した場合、Tuxedoはリカバリ時に、トランザクションのコミットを確実に終了させることができます。トランザクション・ログが、高可用性を備えていないストレージに配置されている場合、トランザクションのACIDプロパティが違反状態になり、一貫性のない状態に移行してしまうことがあります。Tuxedoノードでシステム障害が発生した場合、Tuxedoを再起動すると、トランザクション・ログの内容を読み取って、障害発生時にコミットの最中であったトランザクションを判別します。再起動時にトランザクション・ログが使用できない場合、トランザクションが部分的にのみコミットされた状態で終了してしまうこ

とがあります。この場合、アプリケーションは一貫性のない状態に置かれたままになり、リソース・マネージャ内のリソースはロックされます。

トランザクション・ログの1つのオプションとして、そのログをOracle Databaseに配置することがあります。この場合、ログが高可用性を備えていることが保証されます。トランザクション・ログのOracle Databaseへの格納を有効にするには、UBBCONFIG内のTLOGDEVパラメータを、データベースをポイントするように設定し、TLOGNAMEパラメータを、トランザクション・ログの一意の表名に設定します。

キュー領域

Tuxedo内のキュー領域には、永続キューの内容が保持されています。永続キューの状態は、アプリケーションの一貫性に対して重要になることが多いため、キュー領域は、トランザクション・ログの場合と同様に、高可用性を備えたストレージに配置する必要があります。エンジニアド・システムでは、これらのファイルをZFSSA上に配置して、キュー領域が他のノードにアクセスできるようにする必要があります。これにより、Tuxedoキュー・サーバーを保持しているマシンまたはグループを、障害発生時に別のマシンに移行することができますが、キュー領域を保持しているファイルには引き続きアクセスできます。

ディザスタ・リカバリ・トポロジ

業務が全体的にエンタープライズ・アプリケーションに依存するようになったため、情報システムでは、ディザスタ・リカバリがますます重要になりつつあります。ディザスタ・リカバリ環境の計画と実装は複雑なタスクですが、この項では、ディザスタ・リカバリ計画を成功させるために役に立つ、重要な考慮事項およびソリューションに注目して説明します。

アクティブ/パッシブ

もっとも一般的なディザスタ・リカバリ・トポロジは、アクティブ/パッシブ・トポロジです。つまり、あるサイトが現在の負荷全体を処理しており、別のリモート・サイトはアイドル状態ですが、アクティブ・サイトに障害が発生した場合には、負荷を受け入れる準備が完了しています。アプリケーション・コンポーネントにはすべて、フロントエンドWeb層、JEEアプリケーション・サーバー、Tuxedo、およびデータベースが含まれており、各サイトですべて同一の場所に配置されています。このトポロジでは、指定した時間に監視する必要があるのは単一サイトのみであり、比較的管理しやすいトポロジですが、最高のパフォーマンスを実現することもあります。パフォーマンス上のメリットの理由は、おもにすべてのコンポーネント間でローカル・ネットワーク接続を提供していることです。Tuxedoアプリケーションの場合、通常Local Area Network（LAN）での待機時間が短いことが要求される、もっとも重要なネットワーク接続は、データベース・アクセスに使用されます。この重要性は、データベースへのアクセス時に、サービスが呼び出される頻度にかなり依存しています。通常、各データベース・リクエストは、データベースに対してラウンドトリップであることが要求されるため、ネットワークで発生する余計な待機時間は、サービスが実行したデータベース・リクエストの数だけ増大します。アクティブ/パッシブ環境では、データベースがExadataシステムによって提供されている場合、そのデータベースは同一の場所に配置され、InfiniBand経由で接続されています。

アクティブ・サイトからパッシブ・サイトに切り替えるのに要する時間を最小化するために、通常、何らかの形式のレプリケーションを使用して、ファイルやデータベースなどの永続状態をレプリケートすることを推奨します。これにより、バックアップからパッシブ・システムをリストアする

のに要する時間が取り除かれます。ExalogicまたはSuperCluster上のファイルをレプリケートするために、ZFSSAには、パッシブ・サイトでファイルをZFSSAにレプリケートするのに使用できる、組込みのファイル・レプリケーション機能が用意されています。データベースの場合、Oracle Active Data GuardまたはOracle GoldenGateのいずれかを使用して、データベースの変更内容をアクティブ・サイトからパッシブ・サイトに伝播できます。レプリケーションの影響を最小限にするために、通常はデータを非同期でレプリケートします。つまり、データが正常にレプリケートされる直前に、リクエストが確認されます。一部のアプリケーション・シナリオでは、これにより非一貫性が引き起こされることがあり、その場合は、同期レプリケーションを実行できます。

アクティブ・サイトからパッシブ・サイトにデータをレプリケートするタイミングについて検討する際の1つの項目として、トランザクション一貫性があります。アプリケーションが、Tuxedoの分散トランザクション管理機能を使用している場合、重要になるのは、トランザクション・ログとデータベース・レプリケーションが同期しているという点です。これを達成するためには、同期レプリケーションを使用しますが、データがリモート・サイトに安全にレプリケートされるまで、更新が確認されないため、パフォーマンスに大きな影響を与える場合があります。Tuxedoトランザクション・ログがファイル・システム内で保守されている場合、この同期レプリケーションが、一貫性を保証する唯一のオプションです。適切なパフォーマンス・ソリューションは、Tuxedoトランザクション・ログをデータベース内に配置することです。状態情報はすべてデータベース内に含まれており、上述のレプリケーション技術により一貫性が保証されているため、非同期レプリケーションを使用できるようになります。

アクティブ/パッシブのディザスタ・リカバリ・モデルのおもなデメリットとして、パッシブ・サイトでは実際の負荷が処理されないことがあります。ユーザーの実際の負荷ではなく、最大限でも、レプリケーション更新を受け入れるだけです。パッシブ・サイトにおいて、構成、ソフトウェア、ハードウェアなどに関する問題が存在する場合、フェイルオーバーを試みるまで表示されないことがあります。アクティブ/パッシブ・デプロイメントの別のデメリットとして、大量のハードウェアがアイドル状態のままになることがあり、その状態はハードウェア全体の半分にも及ぶ可能性もあります。この場合に使用可能な1つのソリューションとして、パッシブ・サイトにおいて、タスクのレポートなどの読取り専用ワークロードを実行することがあります。

アクティブ/アクティブ

アクティブ/アクティブのディザスタ・リカバリ・デプロイメント・モデルでは、アクティブな負荷が両方のサイトで処理されます。各サイトにトラフィックを送信するには、外部のロードバランサを使用します。このモデルのおもなメリットは、常にすべてのハードウェアを利用して、両方のサイトが正常に運用されているのを確認できることです。アクティブ/アクティブ・デプロイメントの達成における最大の問題は、ファイルやデータベースなどの永続データの処理に関する問題です。理想的な状況では、これらのデータは、論理的にパーティション化されます。パーティションはローカルでのみアクセスされ、一部のレプリケーションでは、他のサイトが更新されます。また、障害が発生した場合、障害が発生したパーティションの負荷を、他のサイトが引き継いで処理することもできます。この手法のメリットは、データに対してローカル・ネットワーク・アクセスを利用するということですが、障害が存在するパーティションでも、引き続き高可用性が達成されます。データのパーティション化は理想的な手法ですが、一部のデータが適切にパーティション化できない可能性があります。ただし、このパーティション化されていないデータへのアクセスが頻繁に発生しない限り、アクティブ/アクティブ・モデルを引き続きデプロイできます。

Tuxedoのアクティブ/アクティブ・デプロイメントでの一般的な手法は、各サイトで、同じセットの

サービスをサポートするように構成されたTuxedoクラスタを所有することです。これにより、各サイトで高可用性が達成され、リモート・サイトへのフェイルオーバーが必要となる回数が最小化されます。パーティション化が使用されており、ロードバランサが、必要なパーティションの正しいサイトに対して、一部またはすべてのリクエストを適切にルーティングできない場合、サイトはTuxedoドメイン・ゲートウェイを使用してリンクされ、トラフィックは、Tuxedoのデータ依存ルーティング機能を使用して、正しいサイトにルーティングされます。この機能により、リクエストの内容に応じて、サーバーの特定のドメインまたはグループに対して、リクエストをルーティングできます。また、この機能は、リクエストおよび応答において、余計なネットワーク・ホップを発生させてしまうこともありますが、パーティションが存在するサーバー上でサービスを実行できます。Tuxedoサービスでは、複数のデータベース・リクエストを実行することがあるため、ローカル・データベースに接続されているサーバーが、リクエストを処理できます。

各サイトにクラスタをデプロイする際の別のオプションとして、両方のサイトにまたがる分割クラスタを作成することがあります。この機能はサポートされており、必要に応じて、パーティションにローカル接続されているサーバー上でリクエストを実行するために、データ依存ルーティングも使用できますが、通常は推奨されません。クラスタ内のマシンが地理的に分散されている場合、パーティション化クラスタが存在する可能性が大幅に増加します。パーティション化クラスタ内では、引き続きリクエストが処理されますが、クラスタのパーティション化された部分は、クラスタが再構築されてはじめて、管理できるようになります。

アクティブ/アクティブ・デプロイメントに関する別の考慮事項として、フェイルオーバーの手順があります。この手順は、対象がトランザクション・ログのみの場合でも必要です。ただし、各サイトでトランザクション・ログのリモート・サイトへのレプリケートが開始されており、障害が発生した場合は、トランザクション・ログをリカバリして、実行中のトランザクションを完了する必要があると仮定しています。

まとめ

Tuxedoは、ミッション・クリティカルなエンタープライズ・アプリケーションをデプロイするための、もっともスケーラブルで、可用性が高く、信頼性の高いプラットフォームを提供しています。Tuxedoでは、C、C++、COBOL、Python、Ruby、PHP、およびJavaがサポートされているため、アプリケーション開発者は、特定の要件に最適な言語を選択できます。この機能を、Oracleエンジニアド・システムで実現されたInfiniBandなどの最先端のハードウェアと組み合わせて、アプリケーションをExadataで実現されたもっともスケーラブルなデータベース・アプライアンスに接続します。これにより、アプリケーションは、単一ラックでは単一コア未満から720コアまで、多重接続されたラックでは5,760コアまでに対応できるようになります。このように、Tuxedoに組み込まれているパフォーマンス最適化機能をOracleエンジニアド・システムに追加することにより、ミッション・クリティカルなアプリケーション向けの他に類を見ないプラットフォームを手に入れたことになります。

ホワイト・ペーパー：
Oracleエンジニアド・システムとTuxedoで処理するミッション・クリティカルなアプリケーション
2014年5月
著者：**Todd Little**

共著者：

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

海外からのお問い合わせ窓口：
電話：**+1.650.506.7000**
ファクシミリ：**+1.650.506.7200**

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together