Tableau Server の高可用性

規模に応じたミッションクリティカルな分析を実行する

著者: Fatima Calcuttawala (プロダクトマネージャー)
Kitty Chou (プロダクトマネージャー)

目次

セルフサービス分析はミッションクリティカル

今日、セルフサービス分析およびデータドリブンな意思決定は、世界中の大手の組織で標準となっています。ユーザーと意思決定者は、質問に対する答えをリアルタイムで得るために、データへの即時アクセスとセルフサービスツールを必要とするようになりました。また、企業の経営陣もデータドリブンな意思決定の重要性を理解しており、毎日これらのシステムを活用しています。データを活用するには、基盤となるシステムに高い可用性が必要です。現在のチームと企業向けツールにとって、よりアクセスしやすく、より簡単に構成できるプラットフォームの機能が必要になっています。

Tableau Server は、ミッションクリティカルなセルフサービス分析の未来を実現するプラットフォームです。セルフサービスのデータ探索を迅速に行え、堅牢なガバナンスによってコンテンツとデータの信頼性が向上します。また、導入と管理が簡単で、あらゆる企業で全社に拡張できます。このホワイトペーパーでは、Tableau サービスマネージャー (TSM) を実行している Tableau Server が、高可用性 (HA) を維持しながら規模に応じたセルフサービス分析を実現する方法をご紹介します。

高可用性を理解する

高可用性システムの目標は、システムのダウンタイムを最小限にすることです。ほとんどのシステム管理者が、保守、アップグレード、パッチ適用のためのダウンタイムを計画します。さらに、計画外ダウンタイムと呼ばれる、予期しない障害が発生する可能性もあります。管理者は当然、ハードウェアまたはソフトウェアのアップデートのために計画的な保守を実施する必要があります。目標は、計画外ダウンタイムを最小限にすることです。

高可用性の実現には一般的な方法が 2 つあります。まず、想定外の障害に対してシステムが堅牢になるように、単一障害点を排除する方法です。障害は現実に発生するものであり、障害から保護するための最善の方法はシステムに冗長性を持たせることなのはよく知られています。そしてもう 1 つは、障害の発生を検出して、信頼できるフェールオーバー機構を必要に応じて作動させる方法です。Tableau Server では、この 2 つの方法により高可用性を実現できます。

ユーザーがデータをすぐに表示して理解できることは非常に重要です。一方で、ビジネスインテリジェンスシステムの可用性を脅かす出来事はいつでも発生する可能性があります。それは、ハードウェア、ソフトウェア、ネットワークに関連するエラーの場合もあれば、ヒューマンエラーの場合もあります。そのため Tableau は、高可用性をすぐに実現できる Tableau Server を開発し、構成と設定を簡単に行えるようにしました。コンポーネント障害の発生時には、少なくとも Tableau Server のプロセスは自動的に再起動され、システムの稼働状態が保たれます。また、複数ノードを導入して適切に構成すれば、冗長プロセスを使用したサーバーの高可用性も実現できます。

最小限の高可用性の導入

高可用性を実現するにはクラスタで冗長性が必要になるため、最初のステップとして Tableau Server の分散インストールを行います。ノードの 1 つがダウンしても、クラスタにある別のいずれかのノードに構成された Tableau Server を実行できるようにするには、どのコアサービスも 1 つ以上が実行可能な状態を保たなければなりません。しかし、Tableau Server の導入環境で高可用性を実現する場合、2 ノードでは不十分です。高可用性の実現には、Tableau Server を 3 ノード以上にインストールする必要があります。その最大の理由として挙げられるのは、クラスタがネットワーク分断の問題から保護された一貫性のある状態にあるかどうかを判断する際に、Tableau Server が利用するクォーラムという概念です(詳しくは「CAP 定理」をご覧ください)。クォーラムは、絶対多数と同じ意味です。クラスタにあるノードのクォーラム (絶対多数) がシステムの状態に合致した場合、システムには一貫性があり、そのためネットワークパーティションの問題から影響を受けにくいと判断できます。2 ノードのクラスタでは、1 ノードがダウンするとクォーラムが確立できなくなり、Tableau Server は障害モードに入らざるを得なくなります。しかし 3 ～ 4 ノードのクラスタは、最大で 1 ノードの障害に耐性があります。また 5 ノード以上のクラスタは、最大で 2 ノードの障害に耐性があります(詳しくは「コーディネーションサービス」をご覧ください)。Tableau サービスマネージャーの Web UI やコマンドラインインターフェイス (CLI) を使うと、簡単にノードを追加して、その追加ノードに冗長サービスを構成できます。

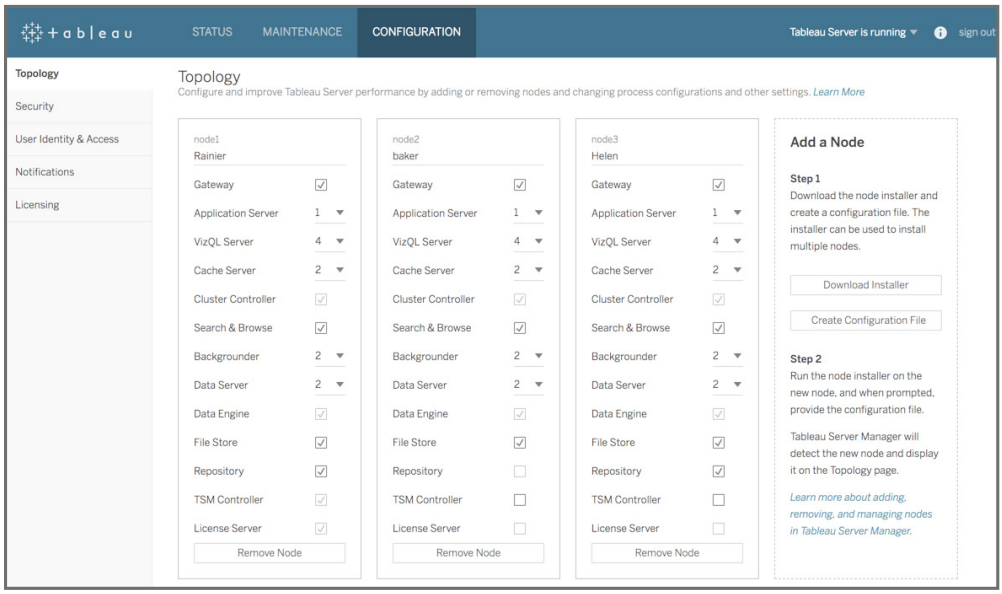

図 1 Tableau サービスマネージャーの Web UI を使うと、ノードや、クラスタのノードごとのプロセスを、簡単に追加/削除できます。

Tableau サービスマネージャーを実行する Tableau Server は、プライマリ/ワーカーサーバーというコンセプトから、クラスタの全ノードをピアとして扱うように変更されました。Tableau サービスマネージャーを利用すると、ライセンス発行サービスの障害に備えるフェールオーバーで、専用のバックアッププライマリマシンを用意する必要はなくなります。Tableau の高可用性の実現方法について、詳しくは「ライセンス発行サービス」をご覧ください。

外部ロードバランサー (ELB)

高可用性システムの導入では、エンドユーザーが Tableau Server クラスタに接続する方法も考慮する必要があります。ほとんどの場合、ユーザーは Tableau Server に割り当てられた DNS ホスト名を使って接続します。これは、ゲートウェイプロセスを持つクラスタのどのノードのホスト名でもかまいません。しかし、ゲートウェイプロセスがダウンした場合、そのホストに接続しているユーザーは Tableau Server にアクセスできなくなります。Tableau Server クラスタの信頼性を高めるには、クラスタに複数のゲートウェイを構成して、Tableau Server クラスタの手前に外部ロードバランサーを追加することをお勧めします。その場合、エンドユーザーは外部ロードバランサーの DNS に接続するだけで、ロードバランサーがクラスタの使用可能な各ゲートウェイにリクエストを分配することができます。そして、クラスタにあるゲートウェイの 1 つが使用できなくなっても、ロードバランサーはその障害を検出して、問題のゲートウェイへのリクエスト送信を停止します。Tableau Server で外部ロードバランサーを適切に設定する方法の詳細は、「ロードバランサーの追加」をご覧ください。

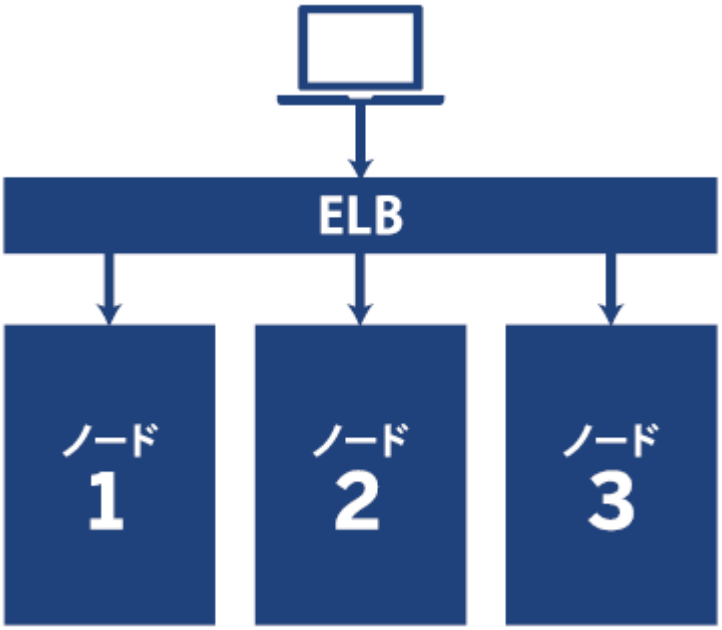

図 2 外部ロードバランサーの後ろにノードが 3 つある Tableau クラスタ

各サーバープロセスに対する Tableau Server の高可用性を理解する

Tableau Server には、エンドユーザーがシステム全体に確実にアクセスできるようにするためのプロセスが複数あります。このセクションは、Tableau Server のコンポーネントとその機能について精通していることを前提としています。まだよく知らない場合は、先に「Tableau Server プロセス」をお読みになることをお勧めします。

Tableau Server で高可用性を実現する方法を理解するには、Tableau Server の各コンポーネントで高可用性を実現する方法を理解することが何より重要です。言い換えれば、Tableau Server クラスタ全体で高可用性を実現して、単一障害点を回避する冗長性を持たせるには、すべてのコンポーネントでも高可用性を実現する必要があります。それでは、コンポーネントを 1 つずつ確認していきましょう。

全般

Tableau Server には、すべてのサーバープロセスを再起動する自動化機能が組み込まれています。この自動化機能によって、障害が発生したサーバープロセスは自動的に再起動されるため、高可用性を維持できます。この自動化が実行されるには、Tableau Server を構築しているハードウェアまたは仮想マシンが正常状態である必要があります。

ノード全体の障害から保護するには、固有のサーバープロセスを構成して、それらがクラスタ内の異なるノード間で冗長性を持つようにすることが重要です。この冗長性は、クラスタ全体でインスタンスを 1 つしか持てないライセンス発行サービスと管理コントローラサービスを除く、全サーバープロセスに持たせることが可能です。この 2 つのプロセスにある制約への対応方法は、第 1 ノードのフェールオーバーについて説明したセクションで詳しく取り上げます。

Tableau サービスマネージャーサービス

Linux 版 Tableau 10.5 と Windows 版 Tableau 2018.2 のリリースで、Tableau サービスマネージャー (TSM) による一連の新しいサービスが登場しました。この新しいサービスは Tableau Server 導入環境の管理を行います。Tableau Server の停止時でも、5 つの Tableau サービスマネージャーサービスが常に実行されています。また、必要な場合以外は停止している Tableau サービスマネージャーメンテナンスサービスも 3 つあります。Tableau サービスマネージャーサービスは、停止した場合でもホスト PC が正常であれば自動的に再起動されます。

コーディネーションサービス

コーディネーションサービスはオープンソースプロジェクトの Apache ZooKeeper 上に構築されており、サーバーのアクティビティを調整します。障害時のクォーラムを保証するとともに、サーバーのトポロジや構成、状態を把握する「情報源」の役割を持っています。Tableau Server の第 1 ノードに自動的にインストールされますが、新しいノードを追加しても別のインスタンスはインストールされません。Tableau Server が正常に動作するにはコーディネーションサービスも適切に動作していなければならないため、3 ノード以上でサーバーをインストールする場合は、新しいコーディネーションサービスアンサンブルを導入して、コーディネーションサービスの新しいインスタンスを追加するようお勧めします。これにより、コーディネーションサービスの 1 インスタンスで問題が発生した場合に備える、冗長性とより高い可用性が得られます。

コーディネーションサービスをインストールするノードの推奨数は、下の表のようにクラスタのノード数に応じて異なります。

クラスタのノード数	コーディネーションサービスがあるノードの数
1 ～ 2	1
3 ～ 4	3
5 以上	5

コーディネーションサービスアンサンブルを導入するノードは選べるほか、1 ノード、3 ノード、5 ノードのいずれかでコーディネーションサービスアンサンブルの導入が可能です。

実行中のコーディネーションサービスプロセスの数が、構成されているコーディネーションサービスプロセスの合計数に基づくクォーラムに達していない場合、Tableau Server は完全に機能停止します。2 つのノードのみのクラスタでは、1 つのコーディネーションサービスプロセスの障害も許容できないことに注意してください。自動フェールオーバーを含む完全な高可用性に、最低でも 3 つのノードが必要なのはこのためです。

調整サービスプロセスが失敗するとどうなりますか。残りの調整サービスプロセスがクォーラムに達している場合は何も起こりません。機能しているコーディネーションサービスプロセスの数がクォーラムに達していない場合、基盤となる Postgres データベースの参照整合性とサービス状態を保護するために、Tableau Server クラスタ全体が利用できなくなります。また、PC 自体に問題がない場合、障害が発生したコーディネーションサービスプロセスは自動的に再起動されます。

エージェント

管理エージェントは、コーディネーションサービスをモニタリングして構成やトポロジの変更を検出します。構成が変更された場合は各サービスに新しい構成を送信し、トポロジが変更された場合は新しいサービスを導入し古いサービスを削除します。また、各サービスの状態を確認して、その結果をコーディネーションサービスに送信する役割も担っています。エージェントプロセスは、インストール時にクラスタの各ノードで自動的に構成されるので、手動で構成する必要はありません。

エージェントプロセスに障害が発生した場合、Tableau サービスマネージャーのステータスページには、同じノードで実行されている他のすべての Tableau Server プロセスに「使用不可」と表示されます。Tableau Server は期待通りに動作し続けますが、クラスタでは構成/トポロジの変更を行えなくなります。また、PC 自体に問題がない場合、障害が発生したエージェントプロセスは自動的に再起動されます。エージェントがノードで起動しない場合は、次のコマンドを実行して手動でサービスを起動してみてください。

- Windows: `sc start tabadminagent_0`
- Linux: `systemctl start tabadminagent_0`

サービスマネージャー

サービスマネージャーは、次のセクションで説明されている Tableau Server ビジネスプロセスのライフサイクルを管理します。このプロセスはエージェントと同様に、インストール時にクラスタの各ノードで自動的に構成されるので、手動で構成する必要はありません。

サービスマネージャープロセスに障害が発生した場合、そのノードの全ビジネスサービスにも障害が発生します。ノードで実行されているサービスによっては、Tableau Server が縮退状態で実行されることもあります。マシン自体が正常な場合、サービスマネージャーは自ら自動的に再起動します。サービスマネージャーがノードで起動しない場合は、次のコマンドを実行して手動でサービスを起動してみてください。

- Windows: `sc start tabsvc_0`
- Linux: `systemctl start tabsvc_0`

クライアントファイルサービス (CFS)

クライアントファイルサービスは、Tableau サービスマネージャーに必要なファイル (SSL 証明書やカスタマイゼーションファイルなど) の保存と配布を行います。これは、ビジネスサービスに必要なファイル (抽出やサムネイルなど) に対するファイルストアの機能に似ています。既定で、クライアントファイルサービスは Tableau Server 導入環境の第 1 ノードにのみインストールされます。高可用性を実現するようにクライアントファイルサービスを構成するには、コーディネーションサービスを導入するそれぞれのノードで、クライアントファイルサービスのインスタンスを構成することをお勧めします。

クライアントファイルサービスプロセスに障害が発生した場合でも、少なくとも動作しているクライアントファイルサービスプロセスがまだクラスタにある限り、何も起こりません。コントローラは、動作している他のクライアントファイルサービスプロセスに、ファイル転送リクエストをリダイレクトします。

また、PC 自体に問題がない場合、障害が発生したクライアントファイルサービスプロセスは自動的に再起動されます。

Tableau サービスマネージャーメンテナンスサービス

クラスタのどのノードにもインストールされる Tableau サービスマネージャーメンテナンスサービスには、データベースのメンテナンス、バックアップ/復元、サイトのインポート/エクスポートの 3 つがあります。この 3 つのサービスは、それを必要とする何かのメンテナンスタスクをサーバー管理者が開始するまで停止されたままです。高可用性の構成を別途行う必要はありません。バックアップや復元などのメンテナンスタスクでのみ使用されるサービスであり、エンドユーザーから見た Tableau Server の動作に影響を与えることもありません。

コントローラ

管理コントローラプロセスは、Tableau Server 導入環境の構成や管理を行うための Tableau サービスマネージャー REST API をホスティングします。管理コントローラのインスタンスは、クラスタ全体で 1 つしかありません。管理コントローラに障害が発生した場合でも、Tableau Server クラスタは動作し続けます。ただし、コントローラが復旧するまで、構成/トポロジの変更や更新を行うことはできません。他の Tableau サービスマネージャーと同様に、コントローラは停止/障害発生の場合に自動的に再起動されます。

それでも問題が解決しない場合は、クラスタの他のノードにコントローラプロセスを移動して問題の軽減を図る必要があります。詳しくは、「第 1 ノードの障害復旧」セクションをご覧ください。

ライセンス発行サービス

ライセンス発行サービスは、サーバー全体のライセンスが適切に設定されているようにする役割を担っています。コントローラと同様に、クラスタ全体で実行されるライセンス発行サービスのインスタンスは 1 つのみです。ライセンス発行サービスは、コントローラと同じ場所にある必要があります。ライセンス発行サービスに障害が発生した場合、スケジュールされた次回のライセンス確認時か、ライセンスの必要なサービスの再起動時まで、Tableau Server は最大 72 時間実行され続けます。ライセンス発行サービスに障害が発生した場合は、ライセンスのない状態にならないように、ライセンス発行サービスを別のノードに移動して、プロダクトキーの再アクティブ化のステップに従う必要があります。詳しくは、「第 1 ノードの障害復旧」セクションをご覧ください。

ビジネスサービス

ゲートウェイ

ゲートウェイプロセスは、Tableau Server のどのノードでも実行できます。ゲートウェイは Web サーバーであり、ブラウザや Tableau Desktop などのクライアントから Tableau Server への全リクエストを処理します。各ノードで実行できるゲートウェイプロセスは 1 つのみなので、システムの冗長性のために、クラスタの複数のノードでゲートウェイのインスタンスを構成することをお勧めします。実際には、各ノードでゲートウェイプロセスを構成すると良いでしょう。こうすることで、ゲートウェイプロセスが単一障害点となってサービスが利用できなくなるリスクを軽減できます。実行されておりトラフィックもあるゲートウェイプロセスが 1 つ以上あれば、クラスタはユーザーリクエストを処理し続けられます。

ゲートウェイは、トラフィックの負荷分散や、Tableau Server にある特定のビジネスサービスへのリダイレクトを行うコンポーネントなので、Tableau Server の高可用性を実現するうえでも重要な役割を果たします。ゲートウェイは、同じ種類のアクティブなサービス全体でトラフィックの負荷分散を行います。何かのプロセスが使用不可になるとそれを検出して、しばらくの間、使用できなくなったプロセスにユーザートラフィックを送信しなくなります。ゲートウェイは、それらのプロセスのステータスを定期的に確認し、復旧したら負荷分散のローテーションに戻します。手動で操作しなくても、このようにトラフィックは自動的に負荷分散され、クラスタの正常なサービスにのみルーティングされます。

ゲートウェイプロセスに障害が発生した場合、前述のように、ゲートウェイプロセスが 1 つも実行されていなければ、Tableau Server クラスタ全体が利用できなくなります。他のゲートウェイプロセスが実行されていれば、機能しているゲートウェイに対するリクエストは正常に処理されます。ただし、他に機能しているゲートウェイがある場合でも、障害のあるゲートウェイによって受け取られたリクエストはリダイレクトされず、失敗したままになります。失敗したゲートウェイプロセスは自動的に再起動されるため、コンピューター自体が機能している場合は、失敗したゲートウェイプロセスが再度実行されてリクエストの処理が再開されます。

ゲートウェイの障害に対してさらに堅牢になるように構成するには、Tableau Server クラスタを外部ロードバランサーの後ろに配置して、機能しているゲートウェイプロセスだけにリクエストがルーティングされるようにすることをお勧めします。設定方法に関する詳しい説明は、ヘルプ「ロードバランサーの追加」をご覧ください。

リポジトリとクラスタコントローラ

リポジトリは PostgreSQL データベースであり、ユーザー情報、パーミッション、ワークブック、データソース、スケジュールなどが保存されている、Tableau Server のメタデータの中央リポジトリです。完全に機能しているリポジトリがない場合、Tableau Server クラスタ全体が利用できなくなります。完全に機能している「アクティブ」リポジトリは、常時 1 つなければなりません。リポジトリが関わるあらゆる処理は、このアクティブリポジトリを使用します。

可用性を向上させるために、クラスタ内の異なるノードに追加の「パッシブ」リポジトリを作成して Tableau Server を構成できます。アクティブリポジトリのコンテンツは、常にパッシブリポジトリに送られます。高可用性を考慮して構成されたクラスタでは、アクティブリポジトリに障害が発生した場合、サーバーの可用性を保つためにパッシブリポジトリのステータスが自動的にアクティブに変わります。高可用性を必要としているお客様には、パッシブリポジトリを構成することを強くお勧めします。クラスタ全体における最大のリポジトリ数は 2 つのみ (アクティブ 1 つとパッシブ 1 つ) であり、その 2 つを同じノードに配置することはできないため、別々のノードに配置する必要があります。

クラスタコントローラは、クラスタのどのノードにも必要なサーバーコンポーネントです。このプロセスは、インストール時にクラスタの各ノードで自動的に構成されるので、手動で構成する必要はありません。クラスタコントローラの主な役割は、リポジトリの起動やシャットダウン、アクティブからパッシブへのフェールオーバーで、モニタリングと管理を行うことです。クラスタコントローラプロセスに障害が発生した場合、そのノードで実行されている全リポジトリプロセスも使用できなくなります。他のサーバープロセスと同様に、PC 自体に問題がない限り、障害が発生したクラスタコントローラプロセスは自動的に再起動されます。クラスタコントローラが再起動されると、そのノードに構成されている全リポジトリプロセスも再起動されます。クラスタコントローラプロセスで障害が発生すると、リポジトリにも問題が発生することがあるのはこのためです。

リポジトリプロセスが失敗するとどうなりますか。それは状況によって異なります。
以下のリストはさまざまなケースをまとめたものです。

- パッシブリポジトリに障害が発生した場合、ユーザーに影響はありません。アクティブリポジトリが機能しているため、すべてが引き続き動作します。バックグラウンドでパッシブリポジトリが再起動され、データ複製が再開されます。ただし、パッシブリポジトリとアクティブリポジトリが完全にもう 1 度同期されるまで少し時間がかかる場合があります。

- アクティブリポジトリに障害が発生したときに、完全に同期されたパッシブリポジトリがあり、クラスタを高可用性の構成にしている場合は、5 分後にパッシブリポジトリへのフェールオーバーが自動的に始まります。フェールオーバー後は、それまでのパッシブリポジトリが新しいアクティブリポジトリとなります。システムによって、障害のあったアクティブリポジトリは新しいパッシブリポジトリとして再起動され、同期が開始されます。また、その他の関連するプロセスも自動的に再起動され、新しく昇格したアクティブリポジトリを認識して再接続します。この短い再起動期間は、サービスが一時的に中断されます。ただし、これは自動化された手順であるため、可用性を維持するために管理者が介入する必要はありません。パッシブリポジトリを手動でアクティブリポジトリに昇格させる場合は、次のコマンドを使用してください。

  ```
  tsm topology failover-repository
  ```

- アクティブリポジトリに障害が発生し、完全に同期されたパッシブリポジトリがない場合は、アクティブリポジトリが再起動されるまで Tableau Server は利用できなくなります。システムが自動的に再起動を試みますが、障害の原因によっては再起動できないこともあります。つまり、現在使えるのがアクティブリポジトリのみの場合は、Tableau Server に高可用性があると見なすことはできません。同期されたパッシブリポジトリがなければ、アクティブリポジトリがシステム全体の単一障害点となります。

注: 2 つのリポジトリ間の同期は、PostgreSQL が実行し制御します。
Tableau Server は一切管理しません。

ファイルストア

ファイルストアプロセスは、ノード間で抽出などのファイルの保存とレプリケーションを管理します。ファイルストアの高可用性は、クラスタの複数のノードにファイルストアプロセスを構成するだけで実現できます。

ファイルストアの仕組み

ユーザーが初めて Tableau Server に抽出ファイルをパブリッシュするか、抽出の更新が行われると、システムに抽出ファイルが作成されます。このようなイベントの直後に、単一のファイルストアに抽出が保存されます。特定の抽出ファイルには冗長性がなく、単一障害点となるため、そのままでは高可用性があるとはいえません。ファイルストアプロセスは相互に通信し、ローカルの抽出をクラスタ内の他のすべてのファイルストアノードにすばやく複製します。ファイルストアプロセスは、クラスタのネットワークリソースが許す限り短い時間でファイルをコピーするように設計されていますが、抽出のサイズやシステムに要求する他のリソースに応じて必要な時間は変わります。コピーされたファイルがクラスタ内の複数のノードで利用可能になれば、その抽出ファイルのレプリケーションは終わり、冗長性を持つようになります。

ファイルストアプロセスが失敗するとどうなりますか。2 種類の結果があります。

• 関連するノード間の抽出ファイルのコピーが停止します。

• 関連するノードで不要になった抽出ファイルの削除が一時停止します(通常、この削除プロセスは「抽出の刈り取り」と呼ばれます)。

抽出の刈り取りが一時停止しても、すぐに影響はありません。不要な抽出ファイルが蓄積することにより、そのノードのディスク領域が消費されるだけです。最終的にはこれが問題につながることもありますが、適切なサイズのノードは十分なディスク空き容量を持っています。

ファイルのレプリケーションが行われないということは、動作しているファイルストアノードに追加された新しい抽出ファイルが、クラスタ内の障害があるファイルストアノードでは使用できないことを意味します。ファイルストアプロセスが再起動されると、すべてのノードにあるすべてのファイルストアを同期することでシステムが自動的に修正されます。

コンピューター自体に問題がない場合、失敗したファイルストアプロセスは自動的に再起動されます。ファイルストアプロセスはすぐに機能を再開し、障害発生中に追加されたファイルも再起動後に追加されたファイルもすべて同期されます。

アプリケーションサーバー

アプリケーションサーバー (VizPortal) は、Web アプリケーションや REST API の呼び出しを処理し、参照と検索をサポートします。アプリケーションサーバーで高可用性を確保することは簡単です。Tableau Server クラスタの各ノードにアプリケーションサーバーインスタンスを構成するだけです。

アプリケーションサーバープロセスが失敗するとどうなりますか。そのインスタンスによって処理されていたリクエストは失敗しますが、その後のリクエストは実行中の他のアプリケーションサーバープロセスにルーティングされます。障害のあるアプリケーションサーバーを含むノードがまだ実行中である場合、失敗したプロセスは数秒で再起動されます。

SAML サービス

サイト固有の SAML を有効にした Tableau Server 導入環境では、アプリケーションサーバーを構成した各ノードで、SAML サービスのインスタンスも実行されます。この構成は、サイト固有の SAML がサーバーで有効にされたときに自動的に行われます。また、サイト SAML が有効になっていない場合、Tableau Server の SAML サービスは停止した状態になります。このプロセスに障害が発生し、ユーザーのリクエストがそのノードのアプリケーションサーバーにルーティングされている場合、ユーザーは Tableau Server にログインできなくなります。他のプロセスと同様に、SAML サービスを持つノードで障害が発生した場合は、障害の発生したプロセスは数秒で自動的に再起動されます。

バックグラウンダー

バックグラウンダーは、サーバーのタスク (抽出更新、サブスクリプション、「今すぐ実行」のタスク、tabcmd から開始されたタスクなど) を実行します。バックグラウンダーサービスの高可用性を実現するには、1 つ以上のインスタンスをクラスタ内の複数のノードで実行するように構成します。バックグラウンダーを実行する場所と実行数を決める際は、各マシンで使用できるキャパシティに対し、他のサーバープロセスが与える影響を考慮してください。

バックグラウンダープロセスがダウンした場合、障害の発生したバックグラウンダープロセスの更新ジョブとサブスクリプションジョブは、バックグラウンダープロセスが障害から復旧し次第、再び実行されます。ほとんどのバックグラウンドジョブは定期的に実行されるようにスケジュール設定されているため、機能しているバックグラウンダープロセスが次回のスケジュール時間に同じバックグラウンドタスクを開始し、通常通りに実行します。

障害が発生したバックグラウンダープロセスは、PC 自体に問題がない限り自動的に再起動され、障害が発生したジョブも再び実行されます。

Data Server

Data Server は、Tableau Server データソースへの接続を管理します。Data Server の高可用性を実現するには、1 つまたは複数の Data Server プロセスをクラスタ内の複数のノードで実行するように構成します。

Data Server プロセスが失敗するとどうなるでしょうか。Data Server プロセスで実行されているクエリにも障害が発生し、ビューのレンダリング、抽出更新、アラートの障害を引き起こします。機能している Data Server があり、再ルーティングされたリクエストを受け取れる場合、障害の発生した操作の再試行など、その後のリクエストは成功します。

Tableau Server は Data Server にかかわらず動作します。しかし、実行中の Data Server がない場合、サーバー上のワークブックはパブリッシュされたデータソースに対して、接続もクエリの実行もできなくなります。なお、データソースに対して Data Server を使っていないビューは、引き続き正常に機能します。

VizQL Server

VizQL Server は、ビューの読み込みとレンダリング、クエリの計算と実行を行います。VizQL Server プロセスの高可用性は、1 つ以上のインスタンスを複数のノードで実行するように構成するだけで実現できます。

VizQL Server プロセスで障害が発生した場合、VizQL Server プロセスが 1 つしかないのであれば、Tableau Server はビューをレンダリングできなくなります。高可用性を達成するには、冗長性のある VizQL プロセスを構成する必要があります。ほとんどの一般的な構成では、各ノードで 2 つから 4 つの VizQL Server プロセスを実行します。これにより、高可用性とスケーラビリティのニーズを同時に満たすことができます。また、複数の VizQL Server プロセスを実行していて 1 つのプロセスで障害が発生した場合、障害発生時のリクエストにも障害が発生してセッションデータは失われます。その後のリクエストは、Tableau Server クラスタで機能している他の VizQL Server プロセスにルーティングされます。

Hyper

Hyper プロセスは、インメモリ分析時に抽出の読み込みとクエリ実行を行います。Hyper プロセスは、ファイルストアや VizQL Server、バックグラウンダー、Data Server、アプリケーションサーバーのプロセスを 1 つ以上実行するノードに自動的に構成されます。

Hyper プロセスがダウンした場合、進行中の抽出更新に障害が発生します。それ以降の .tde 抽出の更新は影響を受けませんが、そのノードの .hyper 抽出の更新には、Hyper が復旧するまで障害が発生し続けます。また、障害の発生したノードで Viz のレンダリングに伴うクエリが実行されている場合は、レンダリングにも障害が発生します。同じ操作を再び実行すると、機能している他の Hyper に自動的に再割り当てされます。さらに、Viz にシャドー抽出があり、障害の発生した Hyper のノードにリクエストがルーティングされた場合、そのノードの Hyper が復旧するまでビューのレンダリングには障害が発生し続けます。

PC 自体に問題がない限り、障害の発生した Hyper プロセスは自動的に再起動され、再開されます。

キャッシュサーバー

キャッシュサーバーは、共有の外部クエリキャッシュとして機能します。これはキーと値のペアのキャッシュであり、以前のクエリの情報を保持して、以降のリクエストの速度を向上させる役割を持っています。キャッシュサーバーの高可用性を実現するには、1 つ以上のキャッシュサーバープロセスをクラスタ内の複数のノードに構成します。

キャッシュサーバープロセスがダウンした場合でも、影響は比較的少なく済みます。Tableau Server は引き続き機能しますが、すでにキャッシュされた結果が利用できないため、アクションには比較的長い時間がかかる可能性があります。クエリが再度実行されると、再起動されたキャッシュサーバーに再び保存されて、エンドユーザーの操作速度が向上します。事実上、キャッシュサーバーは可用性に影響しませんが、エンドユーザーパフォーマンスのさまざまなシナリオに影響を及ぼします。ユーザーパフォーマンスへの影響を抑えるには、クラスタ全体でこのタイプのプロセスを複数実行してください。

PC 自体に問題がない限り、障害の発生したキャッシュサーバープロセスは自動的に再起動され、再開されます。

検索と参照

検索サービスは、サーバーのコンテンツメタデータの高速検索、フィルタリング、取得、表示を処理します。検索と参照プロセスの高可用性は簡単に実現でき、検索と参照プロセスを複数のノードで実行するようにシステムを構成するだけです。

検索と参照プロセスで障害が発生した場合、Tableau Server の機能の多くは使用可能のままで、ユーザーもシステムにログインできますが、ワークブックコンテンツが不足しているように見えます。しかし、実際はコンテンツは不足しておらず、検索結果に返されなくなるだけです。検索と参照プロセスが再起動されると、コンテンツは再び表示されるようになります。複数の検索 & 参照プロセスが複数のノードで構成および実行されている状態で障害が発生した場合、失敗した検索 & 参照プロセスに対するリクエストも失敗しますが、その後のリクエストは機能している検索 & 参照プロセスにルーティングされます。各検索 & 参照プロセスで、クラスタ内のすべてのノードにわたるインデックスが作成されるため、1 つを除いてすべての検索 & 参照プロセスが失敗した場合でもすべてのノードに結果が返されます。

クラスタの状態をモニタリングする

ここまで、各サーバープロセスの障害時の動作と、障害のリスクを抑えて Tableau Server クラスタ全体の高可用性を維持する方法を説明しました。しかし、障害の各シナリオに対して計画を立てるだけではなく、過去に起こった障害を知っておくためにクラスタを積極的にモニタリングする必要もあります。

システム管理者は、Tableau サービスマネージャーのステータスページで Tableau Server の状態をモニタリングできます。このページには全ノードのサーバープロセスが表示されており、クラスタ全体の健全性を理解することができます。

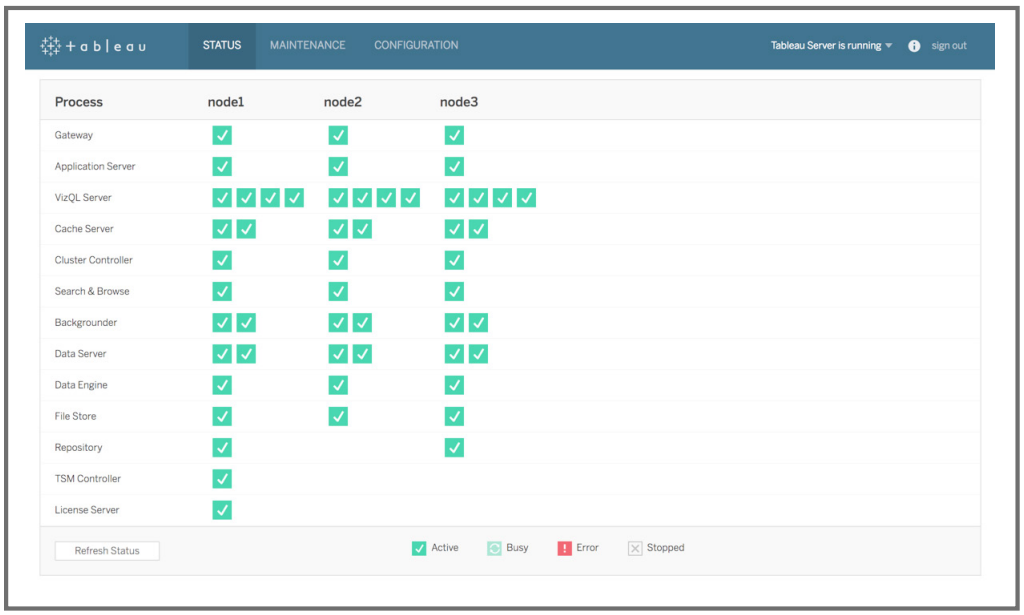

図 3 Tableau サービスマネージャーのステータスページには、各ノードのプロセスごとにステータスが表示されます

Tableau Server では、システムに障害が発生した場合にシステム管理者にメールアラートを送信するよう設定できます。また、ディスク空き容量の問題について事前に警告を送信することも可能です。

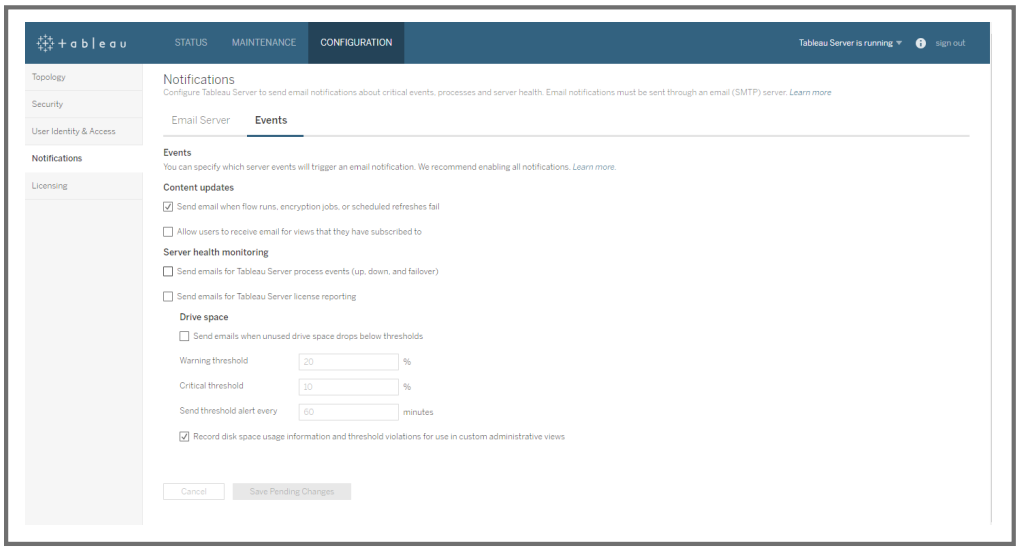

図 4 Tableau サービスマネージャーの Web UI では、システム障害時にアラートを送信するように設定できます

サードパーティーのモニタリングツールを統合する

Tableau Server に組み込まれている機能を使ってシステムの健全性をモニタリングするだけではなく、各プロセスのステータスをコンピューターに読み込める JSON 形式でリモートで受け取ることもできます。それには、各ステータスに対して Tableau サービスマネージャーの REST API を使う必要があります。例を示します。

```
GET /status
Response status code
200 OK
Response body example
{
  "clusterStatus": {
    "nodes": [{
      "services": [{
        "serviceName": "filestore",
        "instances": [{
          "code": "ACTIVE",
          "processStatus": "Active",
          "instanceId": "0",
          "timestampUtc": 1497060680268,
          "currentDeploymentState": "Enabled",
          "binaryVersion": "<build>"
        }],
        "rollupStatus": "Running",
        "rollupRequestedDeploymentState": "Enabled"
      }, {
        "..."
      }],
      "nodeId": "node1",
      "rollupStatus": "Running",
      "rollupRequestedDeploymentState": "Enabled"
    }],
    "href": "/api/0.5/status",
    "rollupStatus": "Running",
    "rollupRequestedDeploymentState": "Enabled"
  }
}
```

第 1 ノードの障害復旧

Tableau Server のインストールでは、ライセンス発行サービスとコントローラのプロセスが、クラスタの第 1 ノードに既定でインストールされます。この 2 つのサーバープロセスはクラスタ全体でそれぞれ 1 インスタンスのみなので、クラスタの他のノードにはなく、したがってシステムの単一障害点になります。つまり、この 2 つのプロセスで障害が発生した場合、サーバーはライセンスのない状態になり使用できなくなる可能性があります。また、復旧するまでサーバークラスタの管理も変更もできなくなります。そのため、上記のモニタリング手段を使って障害を検出し対処することが極めて重要です。

第 1 ノード (ライセンス発行とコントローラを実行しているノード) で障害が検出された場合でも、Tableau Server は次回のライセンス確認時か、ライセンスの必要なサービスの再起動時まで動作し続けます。問題の原因を突き止めて、問題があるノードやサービスを修復してください。ノードやサービスを正常な状態に戻せない場合は、ヘルプトピックの「初期ノードの障害からの回復」に記載されている手順に従って、この 2 つのサービスをクラスタの他のノードに移動することをお勧めします。なおその手順は、簡単に何度でも行えるようにして実行時に入力ミスを防ぐために、スクリプトで自動化することもできます。

アーキテクチャの考慮事項

高可用性を適切に実装するには、組織のアップタイムの目標と、必要としているサービスレベルを理解し、それに沿って冗長性を計画する必要があります。最適なクラスタ構成は、ビジネスのニーズと、組織で利用できるリソースを考えてあわせて決める必要があります。高可用性を実現する構成はいくつか考えられますが、さまざまなサイズのクラスタのトレードオフを理解して、現在の環境に最適な構成を選択することが重要です。

さらに、対称性のある PC 構成 (各ノードに同じコンポーネントをすべて置く) と、非対称の PC 構成 (各ノードにあるコンポーネントの数と種類が異なる) のいずれかを選択できます。通常、ワーカーノードを対称性のある構成にすると、ノードを複製してクラスタに追加することが簡単になります。ただし、構成を計画する際は、1 つのアクティブリポジトリと 1 つのパッシブリポジトリという制限があることに留意してください。

基本的な 3 ノードによる高可用性の導入

前述したように、完全な高可用性モードで実行するには、クラスタに最低 3 ノードが必要です。3 ノードのクラスタは、大規模な導入の第一歩としても最適です。1 つのノードに障害が発生した場合でも、残りの 2 つのアクティブなノードによってクォーラムを達成できます。下の図 5 に、基本的な 3 ノードによる高可用性の導入を示します。すでに述べたように、第 1 ノード (ノード 1) はクラスタ全体の管理機能とライセンス発行機能をホスティングしますが、その点を除けばクラスタの他のノードと何の違いもありません。この構成では、アクティブリポジトリを持つノード 1 に障害が発生した場合、パッシブリポジトリを持つノード 3 が自動的にアクティブになります。クラスタの 3 つのノードはいずれも、Tableau Server のデータとビジュアライゼーションの機能を処理するように構成され、クラスタ全体と各ノード内の両方に冗長性があります。また、コーディネーションサービスは 3 ノードすべてに導入されています。いずれか 1 つのノードで障害が発生した場合でも、クォーラムを達成できます (3 ノード中 2 つが動作し続けます)。なお、コンポーネントの実際の組み合わせは、高可用性のニーズのほか、スケーラビリティのニーズによっても異なることがあります。Tableau Server のスケーラビリティに関する詳細は、ホワイトペーパーをご覧ください。

図 5 基本的な 3 ノードによる高可用性の導入例

さらに、クラスタは外部ロードバランサーの後ろに導入して、エンドユーザーに対する可用性を高めてください。高可用性の導入はこの構成から始めるのが良いでしょう。

4 ノード以上による導入

偶数個のノード (4 ノード、6 ノードなど) でクラスタを導入する場合、そのクラスタのクォーラム能力は 1 つノードが少ないクラスタと同じです。コーディネーションサービスは奇数個のノードにのみ導入できるからです。4 ノードのクラスタを例に取って見てみましょう。

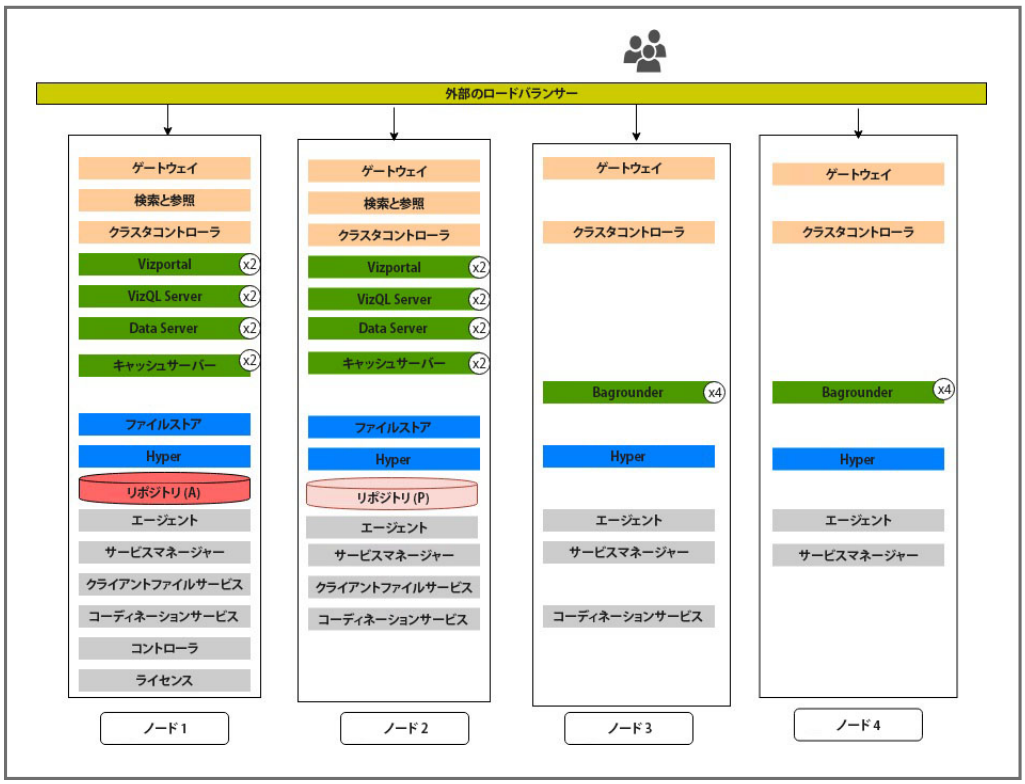

図 6 基本的な 4 ノードによる高可用性の導入例

第 1 ノード (ノード 1) は、クラスタ全体の管理機能とライセンス発行機能をホスティングします。この構成では、アクティブリポジトリを持つノード 1 に障害が発生した場合、パッシブリポジトリを持つノード 3 が自動的にアクティブになります。クラスタのノード 1 とノード 2 は、Tableau Server のデータビジュアライゼーション機能を処理するように構成され、クラスタ全体には冗長性があります。また両ノードには、VizPortal、VizQL Server、Data Server、キャッシュサーバーの複数のインスタンスがあります。一方で、ノード 3 とノード 4 は、バックグラウンダー専用のノードとして構成されています。これにより、抽出更新の作業負荷がデータビジュアライゼーションに支障を及ぼさないようになります。また、コーディネーションサービスは 3 つのノードに導入されています。コーディネーションサービスがあるノードのいずれか 1 つで障害が発生した場合でも、クォーラムを達成できます (3 ノード中 2 つが動作し続けます)。クラスタのノード 4 ともう 1 つのノードで障害が発生した場合も、コーディネーションサービスはクォーラムを達成できるのでクラスタは動作し続けます。しかし、ノード 1、ノード 2、ノード 3 のうち 2 つのノードで障害が発生した場合、3 つのうち 2 つのコーディネーションサービスにも障害が発生するため、クラスタはクォーラムを達成できなくなります。このような場合には、システムのディザスタリカバリ計画が必要になります。

他の考慮事項は関係なくダウンタイムを最小限にすることだけを考えているなら、5 つ以上のノードの導入アーキテクチャをお勧めします。このホワイトペーパーでは、初めての導入に適している 3 ノードのアーキテクチャをお勧めしていますが、ミッションクリティカルな高可用性要件がある全社規模での導入の場合は、基準として 5 つ以上のノードを使用することを検討してください。

抽出と抽出の更新を頻繁に行う組織では、一般的に、専用のワーカーノードにバックグラウンダープロセスを取り入れた導入アーキテクチャを採用します。抽出の更新のワークロードは VizQL Server プロセスがサポートするビジュアライゼーションのワークロードと干渉することがあるため、バックグラウンダープロセスを専用のワーカーマシンに割り当てることで、これらの異なる 2 つのワークロード間におけるリソースの競合を避けることができます。クラスタの複数のノードにわたりプロセスの冗長性を確保するために、バックグラウンダーのみのノードは一般的にペアで導入します。

外部のコーディネーションサービスアンサンブル

コーディネーションサービスは、サーバーの他のコンポーネントと通信するため大量の I/O を生成することがあります。したがって、最低ハードウェア要件を満たす程度のコンピューターで Tableau Server を実行している場合や、さらに堅牢な Tableau Server 導入環境を必要としている場合は、コーディネーションサービスのみのノードを使用する構成で Tableau Server をインストールする必要があるかもしれません。これはつまり、他のサーバープロセスを実行しないノードにコーディネーションサービスをインストールし、他のサーバープロセスを実行しているどのノードからもコーディネーションサービスを削除するということです。専用ハードウェアにコーディネーションサービスアンサンブルを設定する方法について、詳しくは「コーディネーションサービス専用ノードの構成」をご覧ください。下の図に、外部コーディネーションサービスアンサンブルを使ったクラスタ設定例を示します。

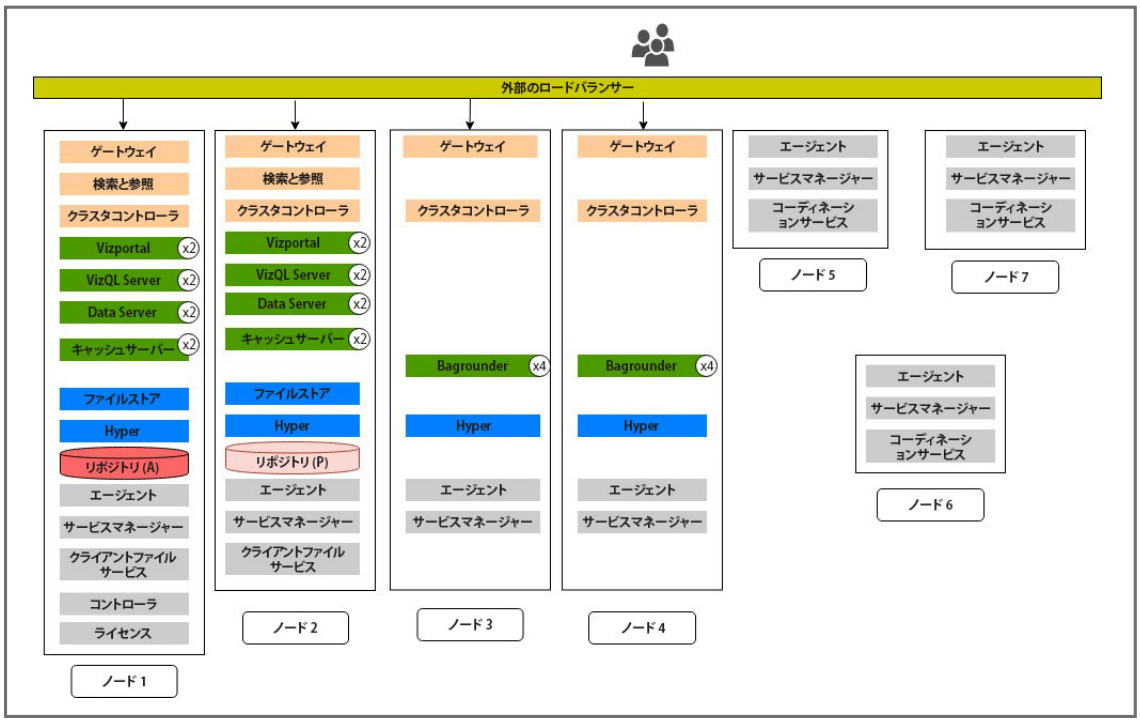

図 7 外部コーディネーションサービスを使った高可用性の導入例

高可用性のその先に

高可用性を実現するには、Tableau Server に組み込まれている機能以外にも必要なものがあります。ハードウェアやネットワークなどの障害は、Tableau Server ソフトウェアが管理できる範囲の外で発生する可能性もあります。パブリッククラウドやプライベートクラウドの仮想ハードウェアなど、そうした障害に対してより復旧容易性の高いインフラストラクチャに Tableau Server を導入すると、高可用性を強化できます。

真の高可用性は、ユーザーのニーズを理解し、そのニーズを満たすためのベストプラクティスとプロセスに従うことによってもたらされます。Tableau の堅牢な機能を使うと高可用性は簡単に達成できますが、定期的な保守や計画的バックアップなどのベストプラクティスに代わるものではありません。ビジネスで意思決定を行うには BI アプリケーションが重要です。組織が抱える分析上の問題を解決するために、Tableau Server をご利用ください。

Tableau について

Tableau は、規模に応じた超高速セルフサービス分析を通じてお客様がデータを見て理解できるように支援する、全てがそろった使いやすいエンタープライズ対応のビジュアル BI プラットフォームです。オンプレミスでもクラウドでも、また Windows でも Linux でも、Tableau はテクノロジーへの既存の投資を生かし、お客様のデータ環境の変化と成長に合わせた規模の拡大が可能です。最も貴重なアセットであるデータと人の力を解き放ちます。

その他のリソース

Tableau サービスマネージャーの概要

分散型/高可用性 Tableau Server インストール環境

Tableau Server における高可用性とディザスタリカバリ

Tableau Server のスケーラビリティ

tableau