Orchestrator 管理者のための
ミドルウェア（IIS と SQL Server）運用設定ガイド

免責事項

- 本ガイドの内容は 2019 年 6 月現在の情報であり、下記の製品リリースに基づいております。
  UiPath Orchestrator v2018.4.6
  製品の新しいリリース、修正プログラム などによって、動作・仕様が変わる可能性がありますので、予めご留意ください。
- 本ガイドに含まれる情報は可能な限り正確を期しておりますが、UiPath 株式会社の正式なレビューを受けておらず、本ガイドに記載された内容に関して UiPath 株式会社は何ら保証するものではございません。従って、本ガイドに含まれる情報の利用またはこれらの技法の実施は使用者の責任においてなされるものであり、ガイドの内容によって受けたいかなる被害に関して一切の補償をするものではございません。
- 本ガイドをコピー等で複製する場合は、UiPath 株式会社および執筆者の承諾なしではできません。

商標について

- UiPath、UiPath Orchestrator、UiPath Robot、UiPath Studio および UiPath ロゴは、世界の多くの国で登録された UiPath の米国およびその他の国における商標です。
- Microsoft、Windows、Windows Server、SQL Server、Active Directory および Windows ロゴ は Microsoft Corporation の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその他の関連会社の米国およびその他の国における商標または登録商標です。
- Elastic、および関連するロゴとマークは、Elastic N.V.及びその関連会社の商標または登録商標です。
- Redis は、Redis Labs Ltd の商標です。
- その他、記載されている製品名、会社名およびサービス名はそれぞれの各社の商標または登録商標です。

Revision History

Date	Version	Author	Description
2019/06/25	1.0	Hiroaki Mishima	第 1.0 版

目次

Revision History 2

1. はじめに 6

1.1. 概要と注意事項 6

1.2. UiPath の用語 7

2. Windows Server の設定 8

2.1. 電源プラン 8

2.2. Windows Server のウイルススキャン除外設定 9

3. IIS の設定 11

3.1. IIS インスタンスの設定 11

3.1.1. アプリケーション プールの設定 11

3.1.2. ラピッドフェール保護の設定 14

3.1.3. SQL Connection Pool の設定 17

3.1.4. SignalR の設定 20

3.1.5. Timeout 値の設定 21

3.2. IIS の監視 25

3.2.1. パフォーマンスカウンターによる監視 25

3.2.2. 既定ではオフになっているカウンターの有効化 31

3.3. IIS のウイルススキャン除外設定 33

4. SQL Server の設定 35

4.1. SQL Server OS の設定 35

4.1.1. OS 特権の付与 35

4.1.2. ストレージ構成 37
4.2. SQL Server インスタンスの設定 40
4.2.1. メモリの設定 40
4.2.2. クエリ並列処理度数の設定 42
4.3. データベースの設定 45
4.3.1. データファイルの分散配置 45
4.3.2. ファイルグループの分割 46
4.3.3. データファイルを複数の I/O デバイスに配置 47
4.3.4. トランザクション ログファイルの構成 48
4.3.5. データベースファイルの初期サイズ 49
4.3.6. データファイルの自動拡張サイズ 49
4.3.7. 複数のデータファイルの同時拡張 52
4.3.8. データファイルの分割 54
4.3.9. エクステントの設定 55
4.4. SQL Server のメンテナンス 58
4.4.1. データベースの定常メンテナンス 58
4.4.2. 統計の更新設定 61
4.4.3. データベース内の整合性の確認 63
4.4.4. SQL Server のバックアップ 65
4.4.5. データベースの空き容量の管理 68
4.4.6. メンテナンスプランの利用 70
4.5. SQL Server の監視 74

4.5.1. 稼働監視 74
4.5.2. リソース監視 75
4.5.3. アプリケーション監視 79
4.5.4. SQL Server のボトルネックの探索 79
4.6. SQL Server のウイルススキャン除外設定 83
5. 付録 85
5.1. サーバー障害時のトラブルシューティングに関する Tips 85
5.1.1 解析対象のログ 85
5.1.2 Log Parser 86
5.2. WCF 通信チャネルの設定 88
5.3. 動的管理ビューによる監視 89
5.4. サーバーの監視および性能測定ツールの例 93
5.4.1. SQL Server の固有機能としての監視ツール 93
5.5. PaaS 環境における SQL Server のバックアップ 96
5.5.1. Hyper-V ホストで推奨されるウイルススキャン除外項目 96
5.5.2. Azure SQL Database 自動バックアップについての詳細情報 97
5.5.3. Amazon RDS for SQL Server 97

1. はじめに

1.1. 概要と注意事項

本ガイドは「UiPath Orchestrator（以下、OC と略記）の運用・保守に従事する管理者」を読者として想定しており、OC 稼働に必須の構成要素（ソフトウェア）であるミドルウェア（Microsoft Internet Information Service〈以下、IIS と略記〉および Microsoft SQL Server〈以下、SQL Server と略記〉）に対して推奨される設定項目、監視項目などのベストプラクティスを解説しています。一部に Load Balancer（以下、LB と略記）を利用する場合に特有の推奨設定項目が含まれていますが、LB そのものの設定については本ガイドの範囲を超えるため触れません。

OC に関する日々の運用・保守作業において、監視するべき要素は多岐に亘りますが、中でも SQL Server は注意すべき事項が多く、本ガイドでも多くのスペースを割いて解説をしています。また、個々の要素（ソフトウェア）だけを注視していても、日々の運用で発生する事象要因の解明に繋がらない場合も多く、ある程度横断的かつ網羅的な Tips が必要であることから、本ガイドでは IIS や SQL Server のみならず、オペレーティングシステム（以下、OS と略記）である Microsoft Windows Server（以下、Windows Server）の関連項目についても可能な限り解説しています。

なお、本ガイドに示すウイルス対策ソフトに対するスキャン除外設定については日本マイクロソフト株式会社および Microsoft Corporation の公式記事（公式ブログ記事を含む）を参考にしています。弊社の公式見解ではございません。予めご留意の上、本ガイドをご活用ください。本ガイドで提示されている設定は、記載されている回避策を管理者が自己の判断で使用することを前提に提供されているものであり、自己の責任においてご使用ください。

本ガイドの Web リソースは下記のリンクにございます。

https://www.uipath.com/ja/resources/knowledge-base/middleware-config-maintenance-guide

1.2. UiPath の用語

本ガイドで利用されている UiPath 製品に関わる用語とその定義を次の表に示します。UiPath 製品以外の用語については、必要に応じて説明を与えています。

用語	略語	定義
ユーザー	(N/A)	UiPath RPA Platform へのアクセスが認められている各個人を表します。
Orchestrator	OC	一連の Robot にライセンスを付与し、配置、管理及び監視を行うことができる Web アプリケーションです。Robot が参照するリソースを管理したり、動作状況を監視したり、Robot を実行環境に展開したりできます。
Studio	ST	Robot に実行させるための自動化処理（Workflow）を作成するための統合開発環境です。
Robot	(N/A)	Studio で作成された Workflow に従って、コンピュータ上でプロセス（実際の自動化処理）を実行するソフトウェアです。
Attended Robot	AR	Robot の種類の一つです。ユーザーの監視下において、直接操作をした結果として起動し、Workflow 等のデベロップメント・アウトプットを実行します。
Unattended Robot	UR	Robot の種類の一つです。ユーザーの監視が無い状態で Workflow 等のデベロップメント・アウトプットを実行します。
Queue	(N/A)	Robot が Orchestrator に接続されている場合に利用できます。無制限に項目を保持できる Orchrstrator の収納機能であり、Queue へのアイテムの設定やステータスの変更、処理を行う場合は、専用のアクティビティを使用します。
Asset	(N/A)	Robot が Orchestrator に接続されている場合に利用できます。Orchestrator で管理される各自動化プロジェクトで使用可能な共有変数または認証情報を表します。Asset を作成し特定の情報を格納することで、Robot は容易にその情報にアクセスすることができます。
Process	(N/A)	自動化処理実行時の呼び方です。
Workflow	WF	自動化処理の流れをグラフィカルに表現した Robot への指示内容を表します。
Heart Beat	HB	Robot (/Studio) から Orchestrator に 30 秒間隔で送信されます。Robot (/Studio) のステータス確認のための HTTP リクエストです。
Node Locked	NL	ライセンス体系の一つです。

2. Windows Server の設定

2.1. 電源プラン

Windows OS では、消費電力とパフォーマンスのバランスを取るように、事前に幾つかの電源プランが用意されています。既定では [バランス] と呼ばれる電源プランが設定されています。[バランス] プランは適度な性能を維持しながら消費電力を少なく抑える仕様になっているため、サーバーの能力を 100%発揮する想定になっていません。

優先度：必須

検討タイミング：運用後

推奨

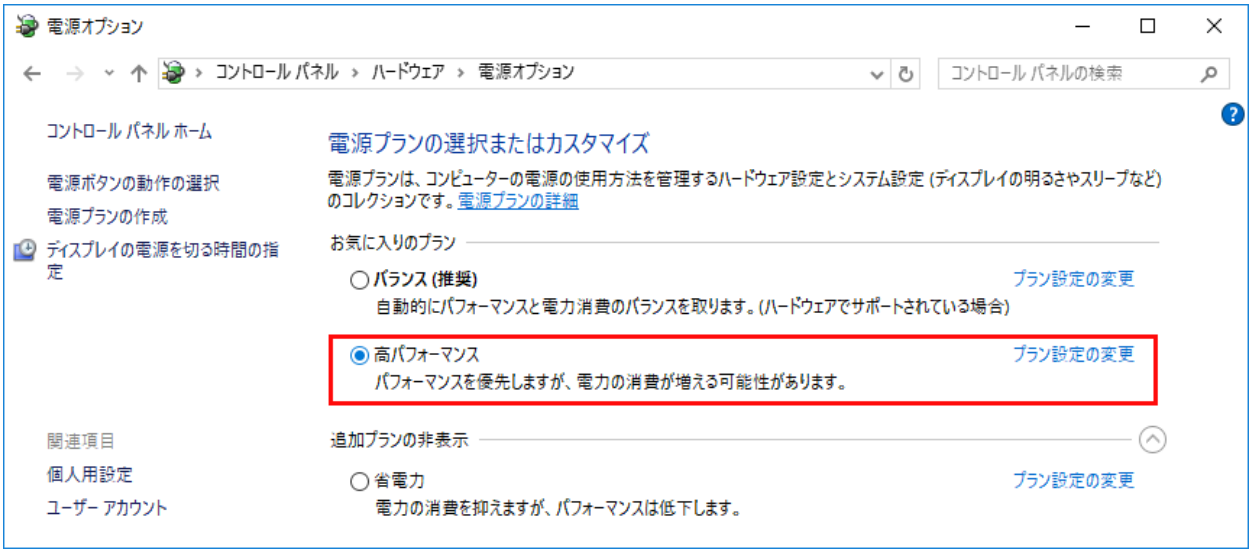
高い処理性能が求められる環境では、電源プランを [高パフォーマンス] に設定します。

理由

電源プランを [バランス] にした状態で CPU に負荷が掛かると、通常は電源プランが CPU のクロックを向上させるような動作をしますが、例えばシングルスレッドのアプリケーションを実行すると CPU クロックがほとんど向上しない場合があります。この理由は、複数の CPU コアを搭載したシステムではシングルスレッドのアプリケーションを実行しても電源プランで定義された CPU クロックを向上させるための閾値に達しないためです。このように [バランス] の設定の場合、CPU に余裕があっても電力消費を抑えるために OS レベルでパフォーマンスを出し惜しみする可能性があります。

方法

[コントロールパネル] > [ハードウェア] > [電源オプション] からシステムの電力消費量を調整できます。



備考

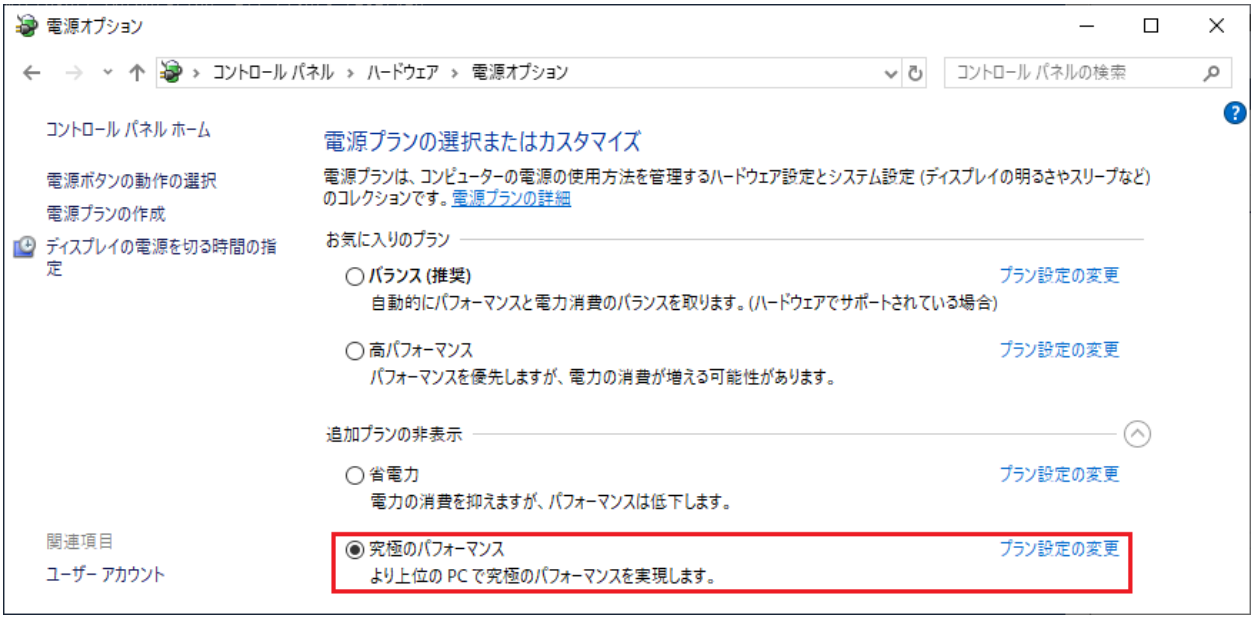
<Microsoft Azure における電源プラン>

Azure での Windows Server 仮想マシン (IaaS) の電源プランは、既定で [高パフォーマンス] プランに設定されています。

<[究極のパフォーマンス] プラン>

[高パフォーマンス] プランの設定を基により詳細な電源管理を行うなどして遅延を少なくした [究極のパフォーマンス] プランも選択できる場合があります。Orchstrator の利用状況やサーバー構成に依存して、[高パフォーマンス] プランとの性能差が表れないこともあるため、ベンチマークテストを実施し、[高パフォーマンス] プラン選択時と比較して適用を検討します。コマンドプロンプトまたは Powwershell を管理者権限で開き、下記のコマンドを実行することで選択できます。

```
※ PowerShell（管理者）で実行
PS %userprofile%> powercfg -duplicatescheme e9a42b02-d5df-448d-aa00-03f14749eb61
電源設定の GUID: ace2b271-2c52-48eb-8eac-b5d82a184aaf  (究極のパフォーマンス)
```



参考

[1] Slow Performance on Windows Server when using the “Balanced” Power Plan
[2] Announcing Windows 10 Insider Preview Build 17101 for Fast & Build 17604 for Skip Ahead

2.2. Windows Server のウイルススキャン除外設定

Orchstrator を動作させるためのミドルウェアが Windows Server 上で安定的に振る舞うために最低限の検討が必要なウイルススキャン除外設定を下記のように提案します。

優先度：必須

検討タイミング：運用後

推奨

除外対象	既定のディレクトリのパス	説明
Windows Update または自動更新データベース ファイル (Datastore.edb)	%windir%\SoftwareDistribution\Datastore	自動更新関連のファイルがウイルススキャンされる際にファイルがロックされ、パフォーマンスに悪影響を与える場合があります。 また、Windows Update を実行するとそのログが Datastore フォルダの Datastore.edb ファイルに記録されていきます。Datastore.edb は Windows セキュリティパッチを適用するたびに肥大化していきますので、ディスクを逼迫させている場合は削除を検討します。
イベント ログ (Edb*.jrs / Edb.chk / Tmp.edb)	%windir%\SoftwareDistribution\Datastore\Logs	
Windows セキュリティ ファイル (*.edb / *.sdb / *.log / *.chk / *.jrs / *.xml / *.csv / *.cmtx)	%windir%\Security\Database	これらのファイルが除外されない場合、ウイルス対策ソフトウェアによりこれらのファイルへの適切なアクセスが妨害され、セキュリティ データベースが破損する可能性があります。これらのファイルをスキャンすると、ファイルが使用できなくなったり、ファイルにセキュリティ ポリシーが適用されなくなる可能性があります。ウイルス対策ソフトウェアはこれらのファイルを独自のデータベース ファイルとして正しく扱わない場合があるため、これらのファイルはスキャンしないでください。

参考

[3] 現在サポートされているバージョンの Windows を搭載しているエンタープライズ コンピューターでウイルス スキャンを行う場合の推奨事項を参照してください。

3. IIS の設定

IIS は、Windows Server 等のオペレーティングシステムに標準コンポーネントとして含まれる Web サーバー機能を提供するソフトウェアです。

Web サーバーは、ロードバランサーやルーター等を除けば、ユーザー（あるいは他システム）から最初の処理要求を受け付けるシステムの入口であるため、セキュリティ対策／脆弱性対策として、Windows Update によるセキュリティパッチの適用に加え、標準設定の見直しや不要な設定の削除を行うことは運用・保守の観点から重要になります。

しかし一方で、サーバー用のウイルス対策ソフトウェアが、メモリリークでシステムに影響を与える事例も報告されています。ウイルススキャンは、フィルタドライバ（カーネルモード）の仕組みを利用しているため、ここにバグが含まれていた場合は、ブルースクリーンで即システム停止となる場合も想定されますし、停止による影響が大きいサーバーと共に運用にした場合は、システムの可用性に悪影響を与える可能性もあります。全てのファイルをウイルススキャン対象にすればウイルス対策の観点からは安全ですが、サーバーの性能／機能に悪影響を及ぼす可能性を回避するためには、一部のファイルについてはウイルススキャン対象から除外するといった措置が必要な場合もあります。

以下では、IIS の設定にフォーカスをし、セキュリティ対策の観点以外で、日常のシステム運用に有益と思われる幾つかの対策について示します。

3.1. IIS インスタンスの設定

IIS マネージャや構成エディター（Web.config 等の設定ファイル）から設定します。

3.1.1. アプリケーション プールの設定

アプリケーションプールとは、Web アプリケーションを動かすための「ワーカープロセスの論理的な集合」を指します。IIS 上で Web アプリケーションが実行され、ウェブサイトで初めてリクエストを受け取ったタイミングで、アプリケーション プールの設定に基づき、Web アプリケーションに対して「ワーカープロセス」がアサインされ、起動します。このプロセスは、[タスクマネージャー] > [プロセス] タブから「W3WP.EXE」という名前で確認することができます。

アプリケーションプールは既定で 1740 分（=29 時間）毎に再起動されます。この再起動処理は「リサイクル」と呼ばれ、メモリリーク対策等で実装されており、プールがリサイクルされるたびにプロセス ID は変化します。IIS マネージャーから、リクエスト数（例えば 1000 万リクエスト毎）、特定の時間（例えば毎朝 03:00）、メモリの使用量等に応じてリサイクルイベントを発生させるように設定することが可能です。

優先度：必須

検討タイミング：運用後

推奨

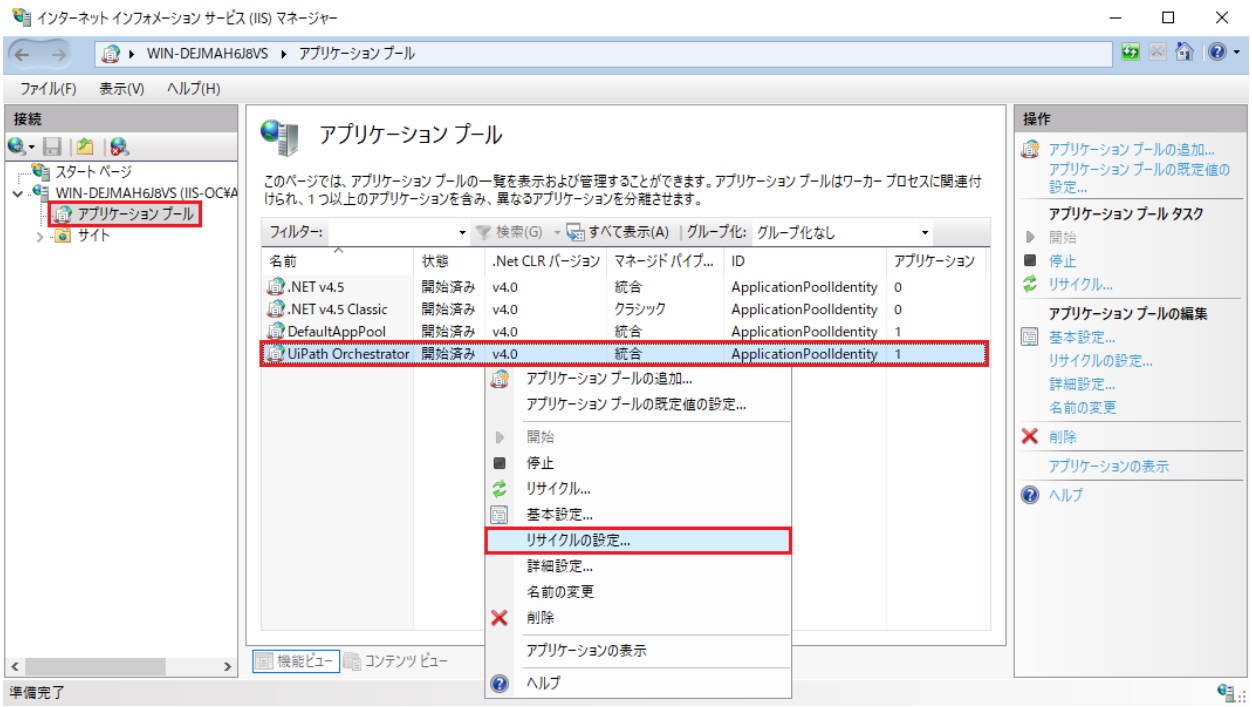
UiPath Studio（以下、Studio と略記）や Robot が OC に接続されている環境では、このリサイクルイベントをメンテナンス時間に限定させます。

理由

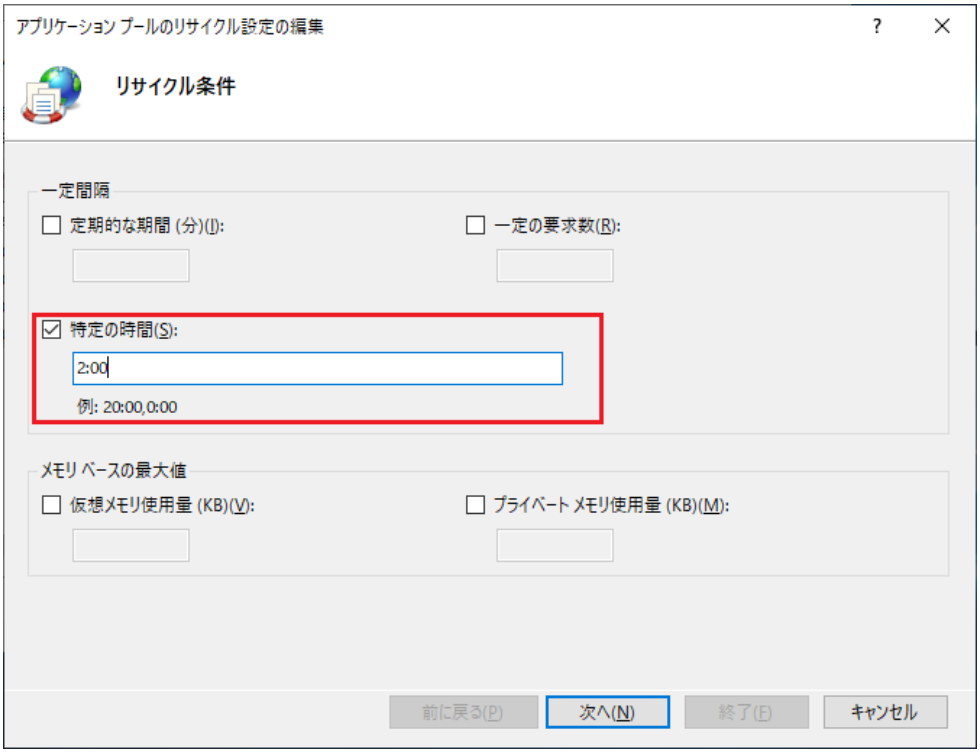
OC で利用される Asset や Queue を使用するロボットの実行中にリサイクルが実行されると、ジョブが失敗する可能性があります。また§3.1.3 で言及するように、不意のリサイクル実行は突発的な SQL Connection の急増を誘発するため、Connection Pool の枯渇によって OC が一時的にスタックする可能性があります。さらに、Studio や Robot のライセンスを OC から配信している場合は、Unlicensed となって開発やジョブ実行が一時的に不能となる可能性があります。

方法

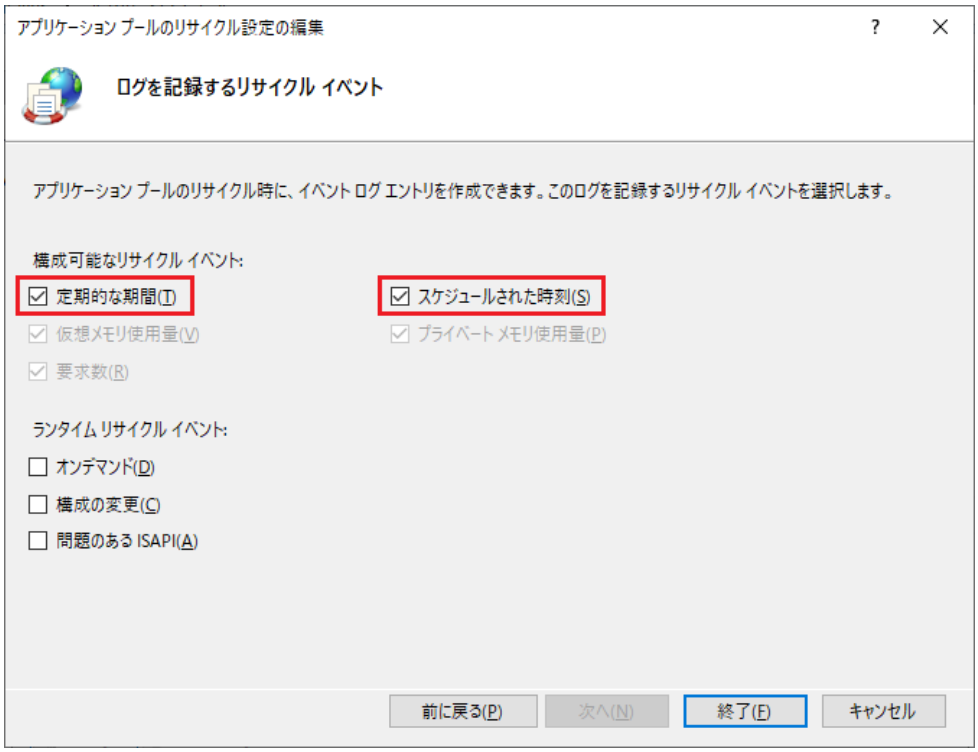
1) IIS マネージャー > [アプリケーション プール] 画面 > UiPath Orchestrator を右クリックして、[リサイクルの設定] 項目を選択します。



2) リサイクル条件として、「**特定の時間(S)**」を設定し、[**次へ(N)**] をクリックします。



3) 例えば、「**定期的な期間(T)**」と「**スケジュールされた時刻(S)**」にチェックを入れて [**終了(F)**]をクリックします。



事象例

IIS のリサイクルが走ると、例えば次のような IIS アクセスログが記録されます。IIS アクセスログは下記のディレクトリに格納されています：

C:\inetpub\logs\LogFiles\W3SVC<*website ID*>\ u_exyyMMdd.log

IIS アクセスログはデフォルトのリサイクル設定では、GET SignalR の reconnect リクエストが記録され、time-taken 値がおよそ 29 時間となります。リサイクルに要する時間は約 15 秒のため、IIS ログを見ると、リサイクルの直後に 15 秒程度のギャップが発生していることが分かります。リサイクル後は HeartBeat の time-taken 値が急増し（通常は 1 秒未満）、HW スペックに依存して IIS Server が不安定になる可能性があります。

time (JST)	cs-method	cs-uri-stem	sc-status	time-taken (ms)
14:41:30	POST	/api/robotsservice/SubmitJobState	200	93
14:41:30	POST	/api/robotsservice/SubmitHeartbeat	200	78
14:41:30	POST	/signalr/hubs/signalr/poll	200	18687
14:41:31	GET	/signalr/hubs/signalr/reconnect	200	104404745 (≃29h)
14:41:31	GET	/signalr/hubs/signalr/connect	200	4109508
14:41:31	GET	/signalr/hubs/signalr/reconnect	200	104404761 (≃29h)
14:41:31	GET	/signalr/hubs/signalr/connect	200	18275413
14:41:31	GET	/signalr/hubs/signalr/connect	200	21897435
14:41:31	POST	/api/robotsservice/SubmitHeartbeat	200	343
14:41:46	POST	/api/robotsservice/SubmitHeartbeat	200	15172
14:41:46	POST	/api/robotsservice/SubmitHeartbeat	200	164659
14:41:46	POST	/signalr/reconnect	101	108189

備考

アプリケーションの再起動は、以下の場合にも発生するため、業務時間内のデータ更新には注意する必要があります。

- 大量のファイルが更新された場合
- 仮想ディレクトリの接続パスが変更された場合
- web.config ファイルが変更された場合
- global.asax ファイルが変更された場合
- bin ディレクトリ内のファイルが変更された場合

3.1.2. ラピッドフェール保護の設定

アプリケーション プールには、割り当てられたワーカープロセスが特定の条件を満たしたタイミングでプールを強制的にシャットダウンさせる機能があり、**ラピッドフェール保護**（英: Rapid-Fail Protection）と呼ばれています。プールの挙動を制御したい場合は、2.2.1 に加えて、この機能に対する設定にも留意する必要があります。

ラピッドフェール保護は、ワーカープロセスに根本的な問題が存在する場合に、処理を継続させないようにする非常措置です。**既定値は True（有効）**に設定されており、IIS Server が下記の状態に陥ると、アプリケーション プール内のすべてのアプリケーションが WWW サービスから削除されます：

- ワーカー プロセスのクラッシュ数が、rapidFailProtectionMaxCrashes 属性で指定された最大値（既定値は 5）に達した場合、且つ

- そのクラッシュが、rapidFailProtectionInterval 属性で指定された時間内（既定では 5 分） に発生した場合。

既定値は「5 分間に 5 回」が閾値のため、5 分間にワーカープロセスの異常終了が 5 回発生すると異常事態とみなされ、ワーカープロセスの再起動は停止し、サービスが停止します。この時、クライアントには HTTP ステータスコード ステータス ID:503（「サービス利用不可」「サービスが利用できません」の意）が返されます。

優先度：任意

検討タイミング：運用後

推奨

許容可能なエラー数上限を緩和する設定を行います。例えば、「1 分間に 6 回」に設定し、約 10 秒毎に 1 回エラーが発生する状態に陥った場合に保護を実行させるようにします。

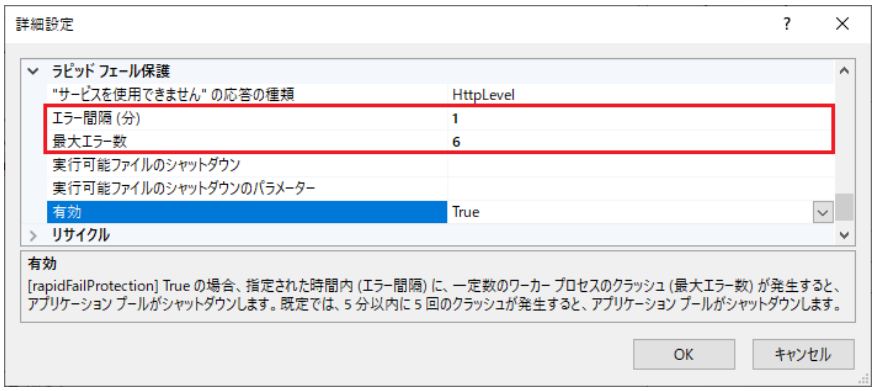
理由

エラーは不定期に発生するが、定期メンテナンス外での OC のサービス停止（ラピッドフェール保護の実行）を極力避けたい場合には、上記設定の適用を推奨します。ただし、断続的にエラーが発生する場合は、メンテナンス時間（本番運用に影響を与えない時間帯）にワーカープロセスのトラブルシューティングを実施してください。

方法

<GUI から設定>

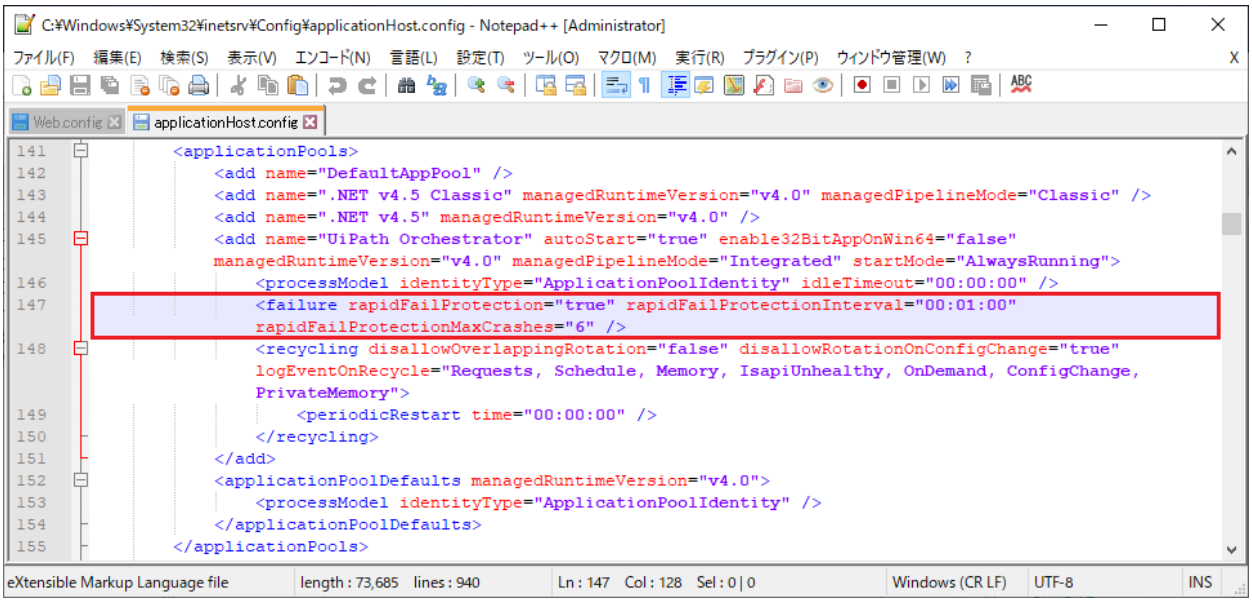
[**IIS マネージャー**] > [**アプリケーション プール**] > [**アプリケーション プールの編集**] > [**詳細設定...**] から「エラー間隔（分）」と「最大エラー数」の値を変更します。



<config ファイルから設定>

%WinDir%\System32\Inetsrv\Config 配下の **applicationHost.config** ファイルを編集します。

```
<applicationPools>
    <add name="DefaultAppPool" />
    <add name=".NET v4.5 Classic" managedRuntimeVersion="v4.0" managedPipelineMode="Classic" />
    <add name=".NET v4.5" managedRuntimeVersion="v4.0" />
    <add name="UiPath Orchestrator" autoStart="true" enable32BitAppOnWin64="false"
managedRuntimeVersion="v4.0" managedPipelineMode="Integrated" startMode="AlwaysRunning">
        <processModel identityType="ApplicationPoolIdentity" idleTimeout="00:00:00" />
        <failure rapidFailProtection="true" rapidFailProtectionInterval="00:01:00"
rapidFailProtectionMaxCrashes="6" />
        <recycling disallowOverlappingRotation="false" disallowRotationOnConfigChange="true"
logEventOnRecycle="Requests, Schedule, Memory, IsapiUnhealthy, OnDemand, ConfigChange,
PrivateMemory">
            <periodicRestart time="00:00:00" />
        </recycling>
     </add>
    <applicationPoolDefaults managedRuntimeVersion="v4.0">
        <processModel identityType="ApplicationPoolIdentity" />
    </applicationPoolDefaults>
</applicationPools>
```



事象例

ラピッドフェール保護が走ると、次のようなエラーが計画外に発生する可能性があります。この場合、対応するワーカープロセスを擁する WWW 接続が切断され、クライアントからのリクエストに対し、「503」エラーが返されます。

- 「アプリケーション プールを提供しているプロセス内での一連のエラーのため、アプリケーション プール 'DefaultAppPool' は自動的に無効になっています。」
- 「アプリケーション プール 'DefaultAppPool' に使われているプロセスで Windows プロセス アクティブ化サービスとの通信に重大なエラーが発生しました。」

備考

<ラピッドフェール保護の無効化>

ラピッドフェール保護の無効化は、IIS 起因のインシデントへの対応が遅れる可能性があるため**非推奨**です。無効化する場合は、定期メンテナンス時に IIS の手動再起動が必須となります。

< applicationHost.config ファイル>

applicationHost.config ファイルの設定の一部は Web.config ファイルに委任され、applicationHost.config ファイル内で指定されている設定は上書きされます。委任許可がされていない設定は Web.config ファイルに追加することはできません。

< DisallowRotationOnConfigChange プロパティ>

false（既定では true）の場合、構成ファイル変更時に強制的にリサイクルが実行されてしまいます。

参考

[4] Configuring Application Pool Settings

3.1.3. SQL Connection Pool の設定

Robot (/Studio) は Orchestrator に 30 秒間隔で Heart Beat が送信し続けています。Orchestrator が Heart Beat をトリガーにして SQL Server の[UiPath] データベース内の情報を参照するために、IIS は SQL Server に対して SQL connection を張ります。

Orchestrator に同時に接続され得る Studio と Robot の合計台数が 100 を超える場合、Web.config で SQL Connection Pool の **Max Pool Size**（既定では 100）を適切に設定する必要があります。

優先度：必須

検討タイミング：運用後

推奨

Max Pool Size の推奨設定値 > Orchestrator に同時接続され得る Studio または Robot 数 ／ Orchestrator のノード数 x 安全係数（例えば、Studio または Robot 1 台当たりの平均スレッド消費数）

HW スペックやネットワークなどの要因で Pool の解放漏れ（Pool の断片化）が起きる場合もあるため、設定値には、ある程度の余裕を持たせる必要があります。

理由

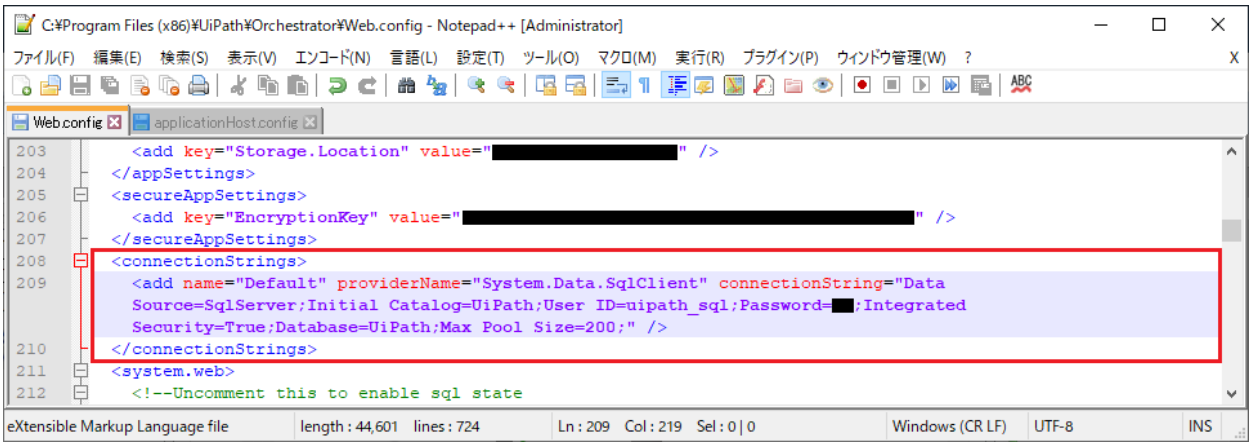
Connection Pool の枯渇防止

方法

例えば Max Pool Size = 200 を設定したい場合、下記の XML コードを Web.config の接続文字列に追加しま

す。

```
<connectionStrings>
      <add name="Default" providerName="System.Data.SqlClient" connectionString="Data
      Source=SqlServer;Initial Catalog=UiPath;User ID=uipath_sql;Password=<password>;Integrated
      Security=True;Database=UiPath;Max Pool Size=200;" />
</connectionStrings>
```



事象例

Application イベントログに「**The operation was canceled**」というエラーメッセージが散見されるようになった場合は、一因として Connection Pool の枯渇が疑われるため、推奨設定の適用を検討する必要があります。

< Application イベントログのエラーメッセージ例>

- Error updating job state:System.Data.Entity.Core.EntityException: The underlying provider failed on Open.
- The underlying provider failed on Open.System.Data.Entity.Core.EntityException: The underlying provider failed on Open.
- Could not authenticate robotSystem.Data.Entity.Core.EntityException: The underlying provider failed on Open.

これらのエラーは総じて次の例外（メッセージ）を伴う：

- System.InvalidOperationException: タイムアウトに達しました。プールから接続を取得する前にタイムアウト期間が過ぎました。プールされた接続がすべて使用中で、プールサイズの制限値に達した可能性があります。

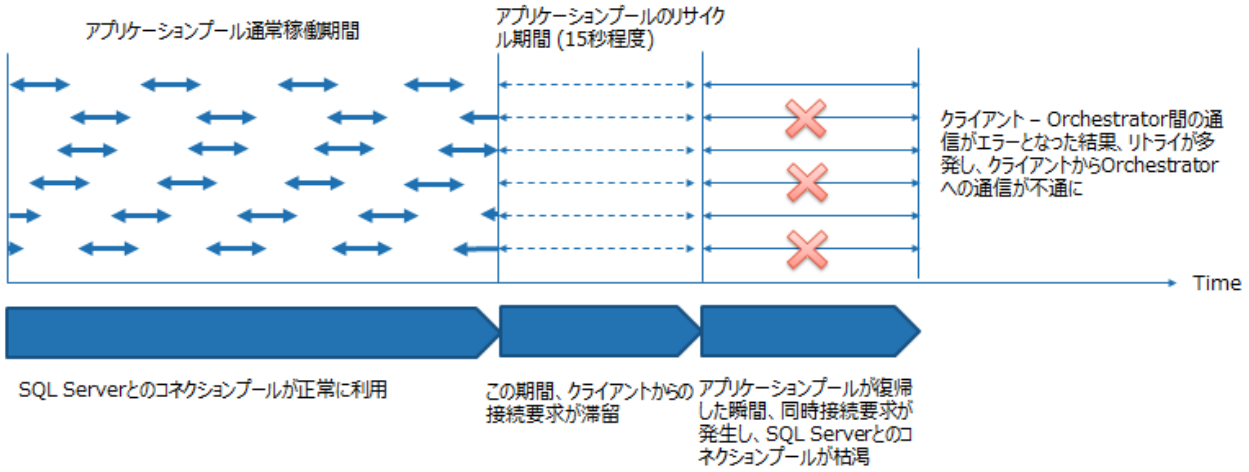
< Connection Pool の枯渇による OC との疎通遮断>

上記のようなエラーが発生するほとんどの要因は、接続文字列キーワード Max Pool Size 値を超えた接続試行が行われた場合です。Max Pool Size は、接続プール機能を有効にしている場合に接続プールで管理可能な接続数を示しています。Max Pool Size の既定値は 100 であり、この既定値を適用した場合は 101 個目の接続を Open しようとした場合に、このエラーが発生します。

Max Pool Size を超える要因は次の 2 つに大別されます：

A) アプリケーションで Max Pool Size 以上の接続が必要となることがある。
B) Connection Pool の Close 漏れ（プールの断片化）があり、Max Pool Size を超える状況が生じる。

プール済みの接続は、Close メソッドまたは Dispose メソッドを実行することで、明示的に再利用可能となります。使用済みと考えている接続でも Close 漏れの場合は再利用できず、接続の Open 要求時に意図せずに新規接続が生成されるような動作では、Max Pool Size を超えてしまう可能性があります。処理の重いプロセスを実行した場合、1 台の Studio または Robot が 2 つ以上のスレッドを消費した結果、コネクションを複数消費する場合も考えられます。



備考

<Connection Pool 消費数の概算>

Orchestrator に接続されている Robot は、既定では 30 秒間隔で Heart Beat を Orchestrator に投げています。正常系では、一体の Robot が 1 つの Heart Beat を投げる毎にクエリが一つ流れます。Node Locked ライセンスのように、1 つの端末に複数のユーザーが登録されている場合においても、1 つの端末からは必ず 1 つの Heart Beat だけが定期的に投げられる。この Heart Beat 1 つにつき 1 つの Connetion が張られます。

- Pool の定常消費数 ≃ Robot 数 ／ HeartBeat 送信間隔（既定では 30 秒） x HeartBeat 平均オーバーラップ間隔（秒） ／ Orchestrator の Node 数

アプリケーションプールのリサイクルが走ると、接続が一時的に切れ、リサイクル終了後にクライアントからの同時接続要求が発生します。例えば、Orchestrator に同時に接続される Studio と Robot の台数が 200 台の場合、

- 定常消費数 ≃ 200 台／30 秒 x 3 秒／1 ノード =20
- 推奨設定値 > 200 台 x 1.5（安全係数の例） = 300

<過剰な Max Pool Size の弊害>

愚直に Max Pool Size を増やしても、Server のスペックに起因して、逆に Orchestrator のトラブルを誘発したり、

パフォーマンスを劣化させる場合があります。少々極端な例ではありますが、**SQL Server の HW スペックに不安がある環境で、**Orchestrator に接続を試行する Robot が 1000 体存在する場合の接続プロセスを考えてみましょう。

A) Max Pool Size = 500 の場合、半分の Robot 500 体の接続は待ち行列に入り、例えば、「接続成功が 500 回 → 接続失敗で 500 回の再試行 → 残り 500 体が接続成功」のように振る舞う可能性があります。
B) Max Pool Size = 1000 の場合、すべての Robot を同時に処理しようとして、個別の接続によるパフォーマンスは落ちてしまうため、最悪の場合、「接続成功 0 回 → 接続失敗ですべて再試行 → 接続成功 0 回 → （接続失敗の無限ループ）」のように振る舞う可能性があります。

参考

[5] 「タイムアウトに達しました。プールから接続を取得する前にタイムアウト期間が過ぎました。プールされた接続がすべて使用中で、プール サイズの制限値に達した可能性があります。」エラーの対処方法

3.1.4. SignalR の設定

SignalR は ASP.NET 上に構築されたライブラリであり、OC のような双方向のリアルタイム通信を必要とする Web アプリケーション開発に用いられます。

優先度：任意

検討タイミング：運用後

推奨

Orchestrator の SignalR チャネルに使用するトランスポートプロトコルを Websockets 通信に指定します。

理由

LongPolling 通信による SQL connection の過剰消費を回避します。Robot または Web ブラウザが Orchestrator へ接続を試みるとき、次の 3 つの方法を順に試行します：① WebSockets → ② Server-Sent Events → ③ LongPolling。まず①で接続を試み、①で失敗したら②で、②で失敗したら③で接続を試行し、その後は延々と③の LongPolling 通信が続きます。LongPolling は Robot およびブラウザが Orchestrator に接続するための最終手段なので、正常系では通常使用されません。

方法

Web.config を下記のように設定します。

```
<!-- Scalability -->
<add key="Scalability.SignalR.Enabled" value="true" />
<add key="Scalability.SignalR.Transport" value="1" /> <!-- Use WebSockets only -->
```

備考

<Robot と Orchestrator 間のコネクション（正常系）>

製品の仕様上、Robot は再接続されるまで Orchestrator への WebSocket 接続を維持しようとする一方、別の接続チャネルとして HTTP リクエストも利用されます。

接続タイプ		リソース利用率	データベースへのアクセス
WebSockets		小さい	なし
HTTP リクエスト	HeartBeat	比較的大きい	あり
	Logs		
	JobState		
	Queues		
	Assets		
	NuGet Pkg		なし

<ブラウザから Orchstrator サイトへの通信>

複数のブラウザから Orchestrator の Web サイトにアクセスして情報を参照／更新すると、Robot 単位の死活監視毎に Post リクエストが飛ぶことで、SQL connection を過剰に消費する可能性があります。パフォーマンスカウンターの NumberOfActiveConnection（§3.2.1 と§3.2.2 を参照）の値が正常時と比較して異常に大きい場合は不必要なブラウザを閉じることで Open 状態の Active な Connection が解放され、プール逼迫状態が改善される場合があります。

<SignalR と IIS アクセスログ>

IIS アクセスログに記録される cs-uri-stem = “/signalr/hubs/signalr/poll” は「Robot が Orchestrator に WebSockets と Server-Sent Events のいずれでも接続が出来なかった際に、最終手段として LongPolling で接続を確立しようとしている」という意味です。そもそも LongPolling が発生すること自体が正常ではないため、“/signalr/hubs/signalr/poll” が散見され始めたら、IIS Server のメモリや CPU のリソース状態について確認する必要があります。ブラウザから Orchestrator サイトへの接続で LongPolling が発生した場合は ”/signalr/poll” として記録されます。

参考

[6] https://orchestrator.uipath.com/lang-ja/v2018.4/docs/app-settings#section-scalability

3.1.5. Timeout 値の設定

Orchestrator に関わる [Timeout] プロパティは複数ありますが、ここでは Orchestrator の環境に応じて設定の変更が推奨される項目を紹介します。

優先度：必須

検討タイミング：運用後

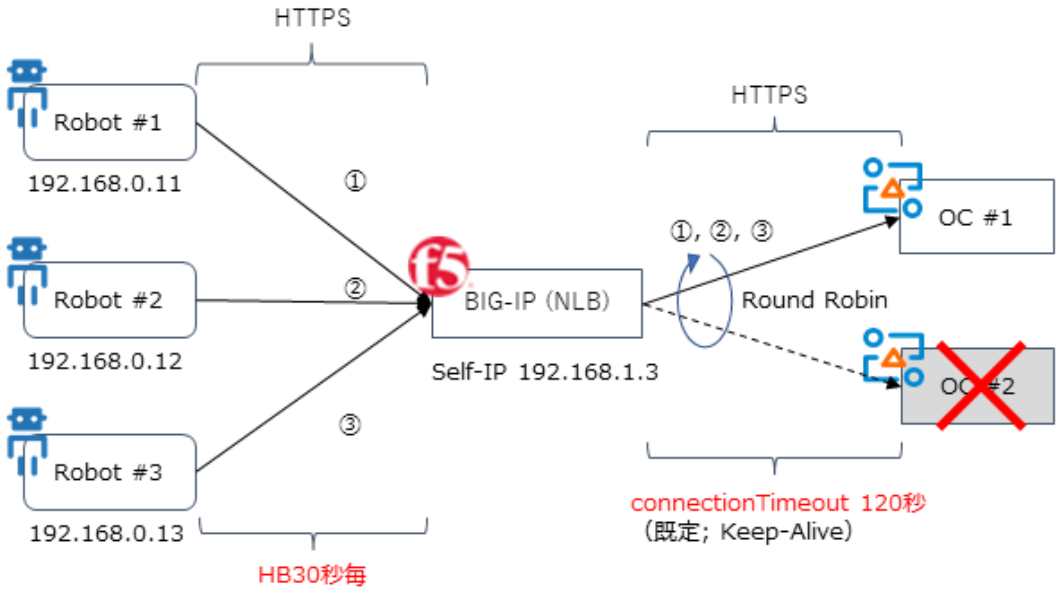
推奨

複数の Orchestrator ノードを Load Balancer で Round Robin にてバランスさせる構成の場合、クライアントと Orchestrator 間の [接続のタイムアウト(秒)] プロパティを 120 秒（既定）から **20 秒**に変更します。

理由

Load Balancer 利用環境において、複数の Ochestrator へ張られるセッションのストライピング平準化のために設定することが推奨されます。

Robot から Orchestrator への HeartBeat は既定で 30 秒毎に送信されます。一方、HTTP キープアライブにより既定値のタイムアウト 120 秒が発生する前に次の HeartBeat が送信されるため、Robot からの TCP セッションは張られ続けることになります。その結果、メンテナンス時などに一方の Orchestrator を停止したときには他方の Orchestrator に接続が偏ってしまいます。その後 Orchestrator (OC) #1 を起動しても、TCP セッションが OC #2 に張りっぱなしとなるため、平準化されないという現象が発生します。



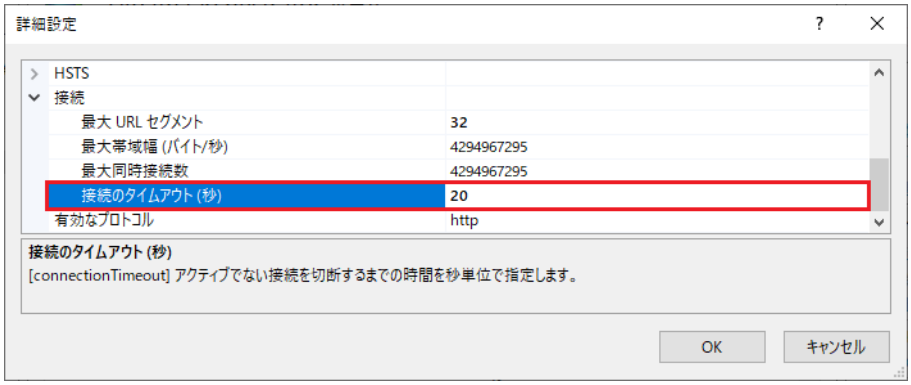
接続タイムアウトを 20 秒に変更すると、次の Heart Beat (HB) が来る前にセッションが一旦終了し、再度張り直すため、この偏りを解消することができます。

方法

以下のいずれかの方法で設定することができます。

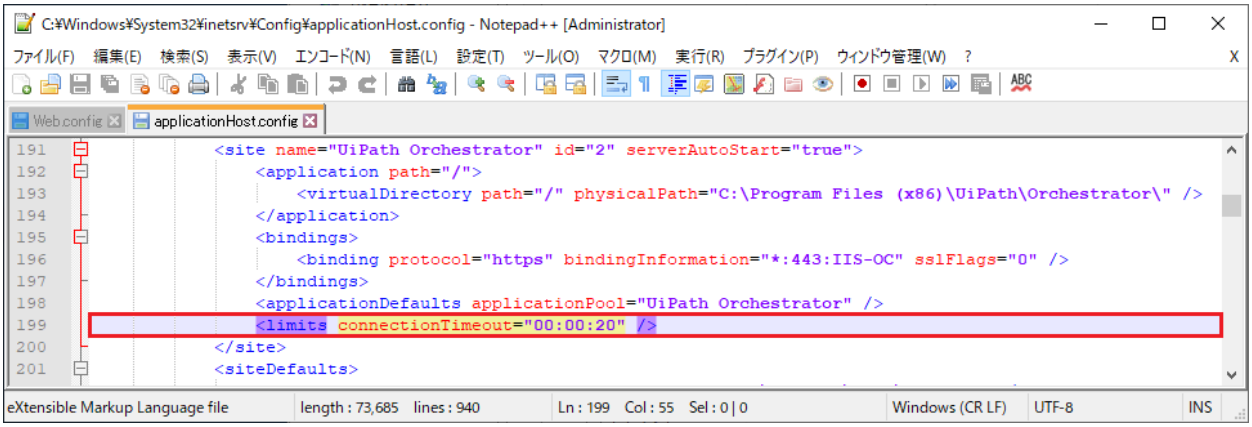
<GUI から設定>

[IIS マネージャー] > [サイト] > [UiPath Orchestrator] > [詳細設定] > [接続] > [接続のタイムアウト (秒)] から、設定値を「120（既定）」から「20」に変更します。



<applicationHost.config ファイルから設定>

このファイルは %Windir%\System32\inetsrv\Config\applicationHost.config にあります。赤枠の箇所のように変更します。




```
<limits connectionTimeout="00:00:20" />
```

事象例

ASP.NET で SQL Server に重いクエリコマンドを投げると、例えば「Timeout に達しました。操作が完了する前にタイムアウト期間が過ぎたか、またはサーバーが応答していません。」といったエラーメッセージがログに記録される場合があります。

備考

<ConnectionTimeout プロパティの存在意義>

このプロパティはネットワーク トラフィックの停滞やサーバーの過負荷などが原因で接続の確立が遅れた場合に、接続の試行を中断するために使われます。接続が開くまでに ConnectionTimeout プロパティに設定した時間が経過すると、エラーが発生し、ADO はその試行を中断する。このプロパティを 0 に設定した場合は、ADO は接続が開くまで無期限に待機します。

SQL Connection の使用／解放の繰り返しのサイクルで、Connection の割当頻度が解放頻度を下回っている状態は正常な均衡を保っています。処理の遅延によりこの均衡状態が狂い、Connection の使用時間が伸びると、割当速度が解放速度を上回ってしまい、タイムアウトによる ConnectionPool の解法漏れ（Pool の断片化）が発生し、プールの枯渇を誘引する可能性があります。この場合はリソース監視結果から HW スペックの見直しが推奨されます。

3.2. IIS の監視

3.2.1. パフォーマンスカウンターによる監視

Windows OS には、システムの状態を監視するために、様々なツールが用意されているが、「パフォーマンスモニター」もその 1 つです。その最も基本的な機能として、収集したカウンター値をリアルタイムに表示し、グラフ化する機能があります。収集できるカウンター値は多様であるが、例えば CPU の利用率のカウンター値を表示させれば、タスク マネージャーの CPU 使用率グラフと同等の結果が得られます。ただしタスク マネージャーではローカルのコンピュータの状態しか取得できませんが、パフォーマンスモニターでは適切な権限を持ったアカウントがネットワーク経由で他のコンピュータのデータも収集できるので、複数のシステムの状態を一括で監視できます。

優先度：必須

検討タイミング：運用後

推奨

パフォーマンスモニターで IIS 由来の特に下記のパフォーマンスカウンターを定期的に監視します。

➢ IIS + SQL Server 共通 resource	#) \object(instance)\counter	説明	正常系条件	対処例
RAM	1) \Memory\Available Bytes	実行中のプロセスに利用可能な物理メモリのサイズをバイト数。このサイズが大きければ、アプリケーションに大きくメモリを割り当てることができます。逆に、メモリがボトルネックになっている場合は小さな値になり、アプリケーションに割り当てるメモリの余力が少なくなります。	4MByte 以上、または 5%以上。	
RAM	2) \Memory\Committed Bytes	仮想メモリーの使用量。継続してしきい値（PI レコード Total Physical Mem Mbytes フィールド）以上の場合，より多くの物理メモリーが必要な可能性があります。		
RAM	3) \Memory\Pages/sec	ハードページフォルトを回避するためにディスクから読み書きされるページング数/秒。この値が高い場合、ページングが過剰であると考えられます。	継続的に 5 以下。瞬間的には 20 まで許容される場合もあります。	メモリ増設
RAM	4) \Memory\Page Faults/sec	このカウンターは Memory\Pages Input/sec と Memory\Pages Output/sec の合計値。システム全体の遅延を引き起こすような事象検知のための主な指標となります。ディスク利用状況がページングによるものかどうかを確認できます。場合はメモリがボトルネックになる可能性があります。	継続的に 5 以下。5 より大きく、且つ使用可能な Bytes も少ない場合はメモリがボトルネックになります。	メモリ増設
RAM	5) \Paging File\%Usage	仮想メモリ（ページングファイル）の使用率。仮想メモリの容量は [コントロールパネル] > [システム] であらかじめ決められており、その容量の範囲内で仮想メモリのサイズが決まります。%Usage カウンタは最大容量の内、どの程度の仮想メモリサイズを使っているかを確認します。	仮想メモリの最大容量の 90%未満。	• 仮想メモリサイズ調整 • メモリ増設
CPU	6) \Processor Information(_Total)\% Processor Time	プロセッサが非アイドルスレッドに費やした経過時間の割合(%) (= % User Time + % Priviledged Time)。	80-85%以下。高い値は CPU がボトルネックになる。	CPU 強化
CPU	7) \Processor Information(_Total)\% User Time	SQL Server 等のユーザープロセスの実行にプロセッサが費やす時間比率。ユーザーモードで経過した非アイドル状態のプロセッサ時間。		

CPU	8) \Processor Information(_Total)\% Priviledged Time	プロセスのスレッドが特権モードでコード実行に費やした経過時間。		
CPU	9) \System\Processor Queue Length	サーバーのプロセッサが他のスレッドの処理でビジー状態になっている際に、%Processor Time を待機しているスレッド数。	2 以下。2 以上かつ % Processor Time が高い状態はプロセッサがボトルネックになります。	CPU 強化
Disk I/O	10) \Physical Disk\% Disk Time	選択したディスクドライブが読み取りまたは書き込み要求の処理でビジーだった経過時間の割合(%)。通常、このカウンターは低い水準で保たれます。	70%以上。	ハードウェアの I/O 効率の向上
Disk I/O	11) \Physical Disk\Avg. Disk Bytes/Transfer	書き込み操作または読み取り操作中にディスクとの間で転送された平均バイト数。このカウンターの値が大きいほど効率的なディスクであるとされます。	20KByte 以上。低い値はディスク I/O がボトルネックになる。	ハードウェアの I/O 効率の向上
Disk I/O	12) \Physical Disk\Avg. Disk Queue Length	サンプル間隔中に選択されたディスクに対して、キューに入れられた（待機している）I/O 要求数の平均値。	スピンドル数＋2 以下。大きな値ではディスク I/O 処理要求で待ちが発生します。	
Network	13) \Network Interface\Bytes Total/sec	各ネットワークアダプタ上で送受信される（フレーミング文字を含めた）バイト数/秒。イーサネットの場合はセグメントの利用帯域幅を計測するために使用することができます。	100Mbps/秒の NIC の場合は 6~7MB/秒未満、1000Mbps/秒の NIC の場合は 60~70MB/秒未満。	
Network	14) \Network Interface\Packets Outbound Errors	ネットワークのエラーで送信できなかったパケット数。	常にゼロ。	
File Cash	15) \Web Service Cache\File Cache Hits %	キャッシュ要求の全体に対してファイル キャッシュがヒットした割合(%)。低い値は、ファイルがキャッシュ機能に何らかの異常があることを意味します。		
File Cash	16) \Web Service Cache\File Cache Misses	サービス起動以降にユーザーモードのファイルキャッシュで失敗した検索の総数。		
File Cash	17) \Web Service Cache\File Cache Flushes	サービスの開始以降にユーザーモードのキャッシュから削除されたファイルの数。		
File Cash	18) \Web Service Cache\URI Cache Hits %	サービス開始以降のキャッシュ要求全体に対する URI キャッシュヒット数の比率(%)。このカウンター値が低い場合、要求がキャッシュされた応答を見つけられていないことが懸念されます。		
File Cash	19) \Web Service Cache\URI Cache Misses	サービス起動以降にユーザーモードの URI キャッシュで失敗した検索の総数。この値が高い場合は、低い URI Cache Hits 値と組み合わせて、問題を調査することが推奨されます。		
File Cash	20) \Web Service Cache\URI Cache Flushes	サービス起動以降にユーザーモードの URI キャッシュで失敗した検索の総数。この値が高い場合は、低い URI Cache Hits 値と組み合わせて、問題を調査することが推奨されます。		

➢ IIS resource	#) \object(instance)\counter	説明	正常系条件	対処例
RAM	21) \Memory\Page Reads/sec	ハード ページ フォルトを解決するためにディスクが読み取られた回数。		
RAM	22) \Memory\Page Writes/sec	ハード ページ フォルトが発生した時にページアウトしたページ数/秒。		

CPU	23) \Process(InetMgr)\% Processor Time	該当プロセスのスレッドすべてが、命令を実行するためにプロセッサを使用した経過時間の割合(%)。	
CPU	24) \Process(InetMgr)\Working Set	プロセスで使用しているメモリサイズ（仮想メモリ含む）の内、実際のメモリ上で確保されているメモリ量。	
CPU	25) \Process(InetMgr)\IO Read Bytes/sec	プロセスが I/O 操作からバイトを読み取っている率。	
CPU	26) \Process(InetMgr)\IO Write Bytes/sec	プロセスが I/O 操作にバイトを書き込んでいる率	
CPU	27) \Process(InetMgr)\Page Faults/sec	ページフォルトの発生回数/秒。この値はハード ページ フォルト（ディスクアクセスを伴う処理）とソフト ページ フォルト（ページ フォルトが物理メモリの他の場所に検出される）の両方を含みます。	
Web Service	28) \Web Service(_Total)\Bytes Received/Sec	Web サービスによって受信されたバイト/秒。	
Web Service	29) \Web Service(_Total)\Bytes Sent/Sec	Web サービスによってデータが送信されたバイト/秒。	
Web Service	30) \Web Service(_Total)\Current Connections	Web サービスまたは FTP サービスで現在確立されている接続数。インスタンスの選択に依らず、すべての Web サイトと FTP サイトの現在の合計値。特に Robot 端末と Orchestrator 間の接続の合計。	正常時と同程度の値。
Web Service	31) \Web Service(_Total)\Get Requests/Sec	GET メソッドを使用する HTTP リクエスト数/秒。高い値は Web サーバーが過負荷であることを表します。	正常時と同程度の値。
Web Service	32) \Web Service(_Total)\Post Requests/Sec	POST メソッドを使用する HTTP リクエスト数/秒。高い値は対策を講じる必要があることを表します。	正常時と同程度の値。
Web Service	33) \Web Service(_Total)\Put Requests/sec	PUT メソッドを使用する HTTP リクエスト数/秒。	正常時と同程度の値。
Web Service	34) \Web Service(_Total)\Delete Requests/sec	DELETE メソッドを使用する HTTP リクエスト数/秒。	正常時と同程度の値。
Web Service	35) \Web Service(_Total)\Connection Attempts/Sec	Web サービスへの接続試行回数/秒。	正常時と同程度の値。
SignalR	36) \SignalR\Connection Messages Received/Sec	接続によって受信された（サーバーからクライアントに送信された）メッセージ数。	
SignalR	37) \SignalR\Connection Messages Sent/Sec	接続によって（クライアントからサーバーに）送信されたメッセージ数。	
SignalR	38) \SignalR\Errors: All/Sec	生成されたエラー数/秒。	
SignalR	39) \SignalR\Connections Connections Current	現在アクティブな接続数。WebSockets (WS)、ForeverFrame (FF)、ServerSentEvents (SSE)、LongPolling (LP) の接続数の合計値。	
SignalR	40) \SignalR\Connections Current WebSockets	現在アクティブな WebSocket による接続数。	
SignalR	41) \SignalR\Connections Current ServerSentEvents	現在アクティブな ServerSentEvents による接続数。	WebSocket より十分少ない値。

SignalR	42) \SignalR\Connections Current LongPolling	現在アクティブな LongPolling による接続数。LongPolling は Robot が Orchestrator に接続するための最終手段でのため、正常系では使われるべきではありません。	基本的にゼロ。	
.NET Framework	43) \.NET CLR Memory(_Global_)\# Total committed Bytes	Garbage Collector によって現在コミットされている仮想メモリのバイト。このカウンターを使用すると、アプリケーションが使用しているメモリの内、Garbage Collector によって消費されているメモリの過多を確認できます。		
.NET Framework	44) \.NET CLR Memory(_Global_)\# Bytes in all Heaps	GC ヒープに割り当てられたコミットと予約の双方の含む仮想メモリのバイト数。このカウンタの値が増え続けている場合、マネージド メモリ リークが発生しています。		
.NET Framework	45) \.NET CLR Memory(_Global_)\# Total committed Bytes	GC によって現在予約されている仮想メモリのバイト。		
.NET Framework	46) \.NET CLR Memory(_Global_)\% Time in GC	前回の Garbage Collection (GC) サイクルの後に GC の実行に費やされた経過時間の割合(%)。このカウンターを使用すると、Garbage Collector がマネージド ヒープの領域を確保するために費やしている時間が長過ぎないかどうかを確認できます。GC に費やされている時間が比較的短い場合、マネージド ヒープ以外のリソースに問題がある可能性があります。	5%以下が正常値。	
.NET Framework	47) \.NET CLR Exceptions(_Global_)\# of Exceps Thrown/Sec	スローされたすべての.NET の例外数/秒。この数には .NET 例外と .NET 例外に変換されるアンマネージ例外の両方が含まれます。 ※ 例えば、アンマネージド コードから返された HRESULT は、マネージド コードの例外に変換されます。このカウンターには、処理済みの例外と未処理の例外の両方が含まれます。このカウンターは、全時間を通しての平均値ではなく、直近 2 回の収集で計測された値の差を収集間隔で割った値が表示されます。例外数が過多でないことを確認する際に利用します。		
ASP.NET	48) \ASP.NET\Requests Rejected	処理不能件数。処理に必要なサーバーリソースが不足しているために実行できなかった要求（503 エラーを返した要求）の合計数を取得できます。		サーバーリソースの増強
ASP.NET	49) \ASP.NET Applications\Requests/Sec	IIS の ASP.NET アプリケーションへの 1 秒間のリクエスト数。		
ASP.NET	50) \ASP.NET Applications\Requests In Application Queue	IIS と ASP.NET の両方によって処理された要求数。この値が小さいほどサーバーによるリクエスト処理能力が高いことを意味します。		
ASP.NET	51) \ASP.NET Applications(_ToTal)\Request Execute Time	IIS Server で稼働しているアプリケーションの実行時間。特にサーバー全体の情報を取得したいので、「__Total」を選択します。		
ASP.NET	52) \ASP.NET Applications\Request Wait Time	処理待ち件数。キュー内で処理を待機していた時間をミリ秒単位で取得できます。		
ASP.NET	53) \ASP.NET Applications\Errors Unhandled During Execution/Sec	未対処の例外数/秒。		
ASP.NET	54) \ASP.NET Applications\Errors Total/Sec	ASP.NET アプリケーション内で発生したコンパイル、前処理、実行エラーの総数/秒。		
ASP.NET	55) \.NET CLR Memory(_Global_)\Allocated Bytes/sec"	配置されたメモリのバイト/秒。		
ASP.NET	56) \.Net Data Provider for SqlServer\NumberOfActiveConnectionPoolGroups	アクティブな一意の接続文字列の数。接続プール グループは接続文字列単位でマップされます。		
ADO.NET	57) \.Net Data Provider for SqlServer\NumberOfActiveConnectionPools	アクティブな接続プールの合計数。		

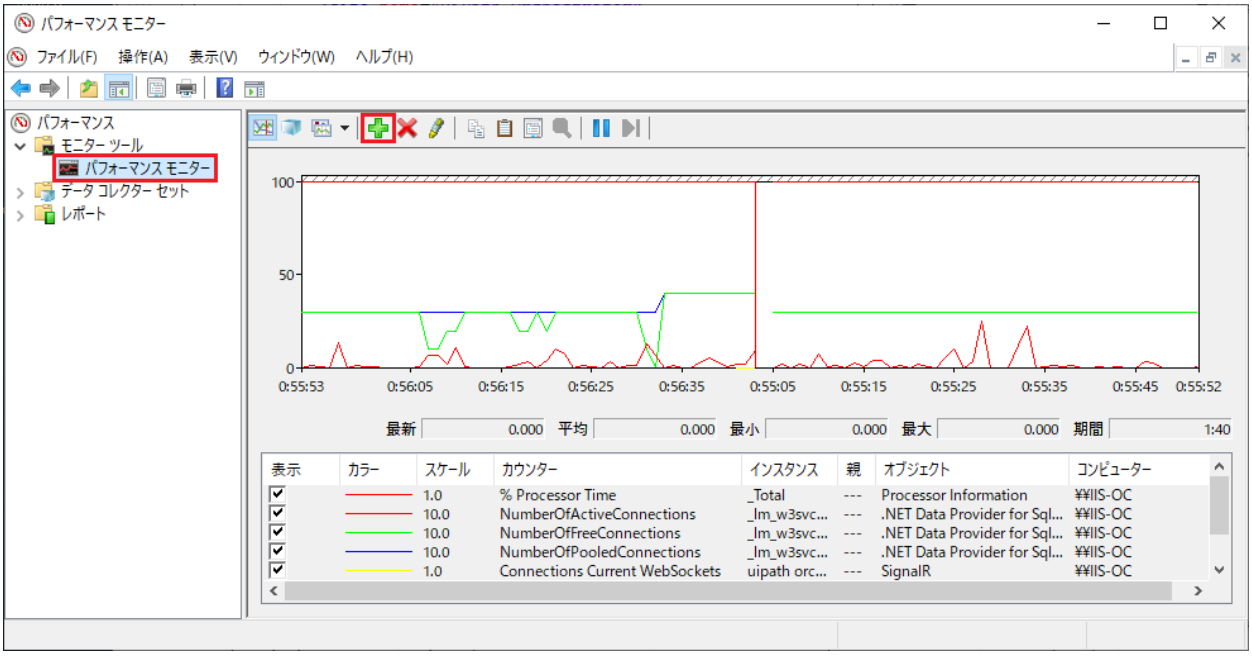
ADO.NET	58) \.Net Data Provider for SqlServer\NumberOfActiveConnections	現在使用中のアクティブな接続数。IIS 側ではクライアント側からの、SQL Server 側ではサーバー側から観た接続数。 ※ このカウンターは既定でオフになっているため、§2.3.2 の方法で有効化する必要があります。	常に Max Pool Size の 25%以下。スパイク（最大値）は 50%以下。	Max Pool Size の変更
ADO.NET	59) \.Net Data Provider for SqlServer\NumberOfFreeConnections	接続プールで Close または Dispose されて使用可能な接続数。 ※ このカウンターは既定でオフになっているため、§2.3.2 の方法で有効化する必要があります。	常に Max Pool Size の 75%以上。	
ADO.NET	60) \.Net Data Provider for SqlServer\NumberOfPooledConnections	接続プールが管理している接続数。再利用可能な接続も含まれる。このカウンターが返す値はすべてのプールを包括します。 ※ アプリケーションに大量のトラフィックを流している状態でこの値が少ない場合、このアプリケーションは少ない接続プールで接続を共有しなければならないため、接続によってボトルネックになっている可能性があります。 ※ NumberOFreeConnections との差が広がり続ける場合、コネクションリーク（プールの断片化）が疑われます。	値がトラフィックに伴って柔軟に変動する。	最小プール接続数 (Min Pool Size)の変更（方法は§3.2.3 を参照）
ADO.NET	61) \.Net Data Provider for SqlServer\NumberOfReclaimedConnections	アプリケーションによって Close も Dispose も呼び出されなかった際に GC によって回収された接続数。接続を明示的に閉じるか破棄しない限り、パフォーマンスが低下する。この値が安定しない場合、コネクションリークが疑われます。	値が安定している。	

理由

事前のトラブル発生の検知、トラブル時の原因の切り分け、原因特定に活用します。

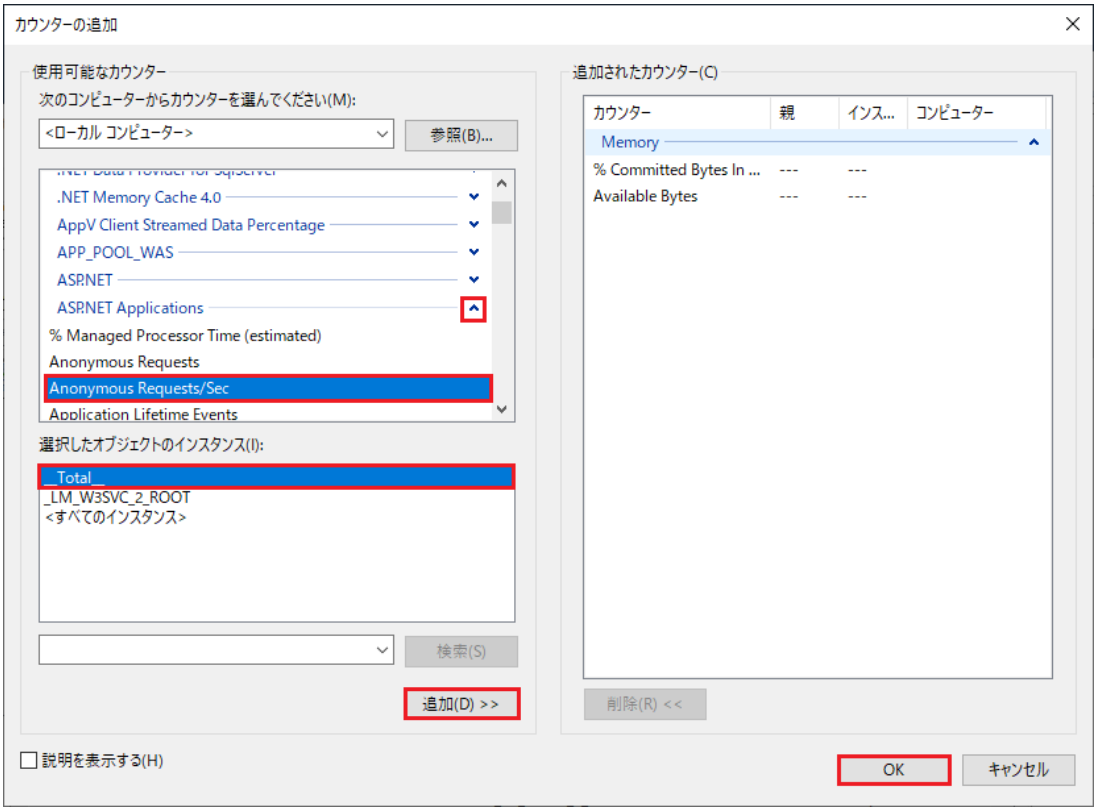
方法

1)　パフォーマンスモニターの [+ (プラス)] タブをクリックします。



2)　該当するカウンターを[使用可能なカウンター]から選択し、[選択したオブジェクトのインスタンス(I)]からインスタン

スを選択し、[追加(D)]ボタンを押下します。



備考

<パフォーマンスカウンターの有効化による影響>

パフォーマンスカウンターの追加による Orchestrator（IIS）へのパフォーマンスへの影響は無視できるレベルです。Orchestrator の Web サイトの描画が遅れる等の影響が出てしまう場合は、HW スペックが適切ではない可能性があります。

<ADO.NET カウンターの振る舞い例>

接続文字列が 3 種類あり、且つ接続自体は 4 つある場合を考えます。1 番目から 3 番目の接続はそれぞれ独立だが、4 番目の接続は 3 番目と同じ接続文字列を利用するとします。Min Pool Size（最小接続数）と Max Pool Size（最大接続数）は既定でそれぞれ 0 と 100 である。NumberOfActiveConnectionPools, NumberOfActiveConnections, NumberOfFreeConnections の数の変化に着目します。それらのカウンター値は最初はゼロで、1 番目 → 2 番目 → 3 番目の順に Open していくと、NumberOfActiveConnectionPools と NumberOfActiveConnections が 1 つずつ増え、3 番目を Open した段階でそれぞれのカウンター値は 3 になる。4 番目の接続文字列は 3 番目と同じなので、4 番目を Open すると、NumberOfActiveConnections は 1 つ増えて 4 になるが、NumberOfActiveConnectionPools は 3 のまま変わりません。

<プールの断片化>

一般に、Web アプリケーションではプロセスが終了するまで解放されないプールを多数作成できる一方で、**コネクション リーク**が問題となる可能性があります。これにより、多数の接続が open のままメモリを消費し、パフォーマンス低下を引き起こす場合があります。接続は、接続文字列 (connectionString) とユーザーID に基づいてプールされます。したがって、Orcherstrator の Web サイトでローカル認証または Windows 認証を使用していて、統合セキュリティ ログインを使用している場合は、1 ユーザー毎に 1 つのプールが作成されます。これによって、単一のユーザーによる後続のデータベース要求のパフォーマンスが向上しますが、他のユーザーによって作成された接続は使用できません。また、1 ユーザー毎に少なくとも 1 つの接続がデータベースサーバーに存在することになります。

参考

[7] Web Service Cache Counters for the WWW Service
[8] Performance Counters for ASP.NET
[9] SignalR Performance
[10] Performance Counters in ADO.NET
[11] SQL Server Connection Pooling (ADO.NET)

3.2.2. 既定ではオフになっているカウンターの有効化

次のパフォーマンス カウンター (ADO.NET) は既定ではオフになっており、追加してもそのままではデータは反映されません。以下に示す手順で有効化する必要があります。

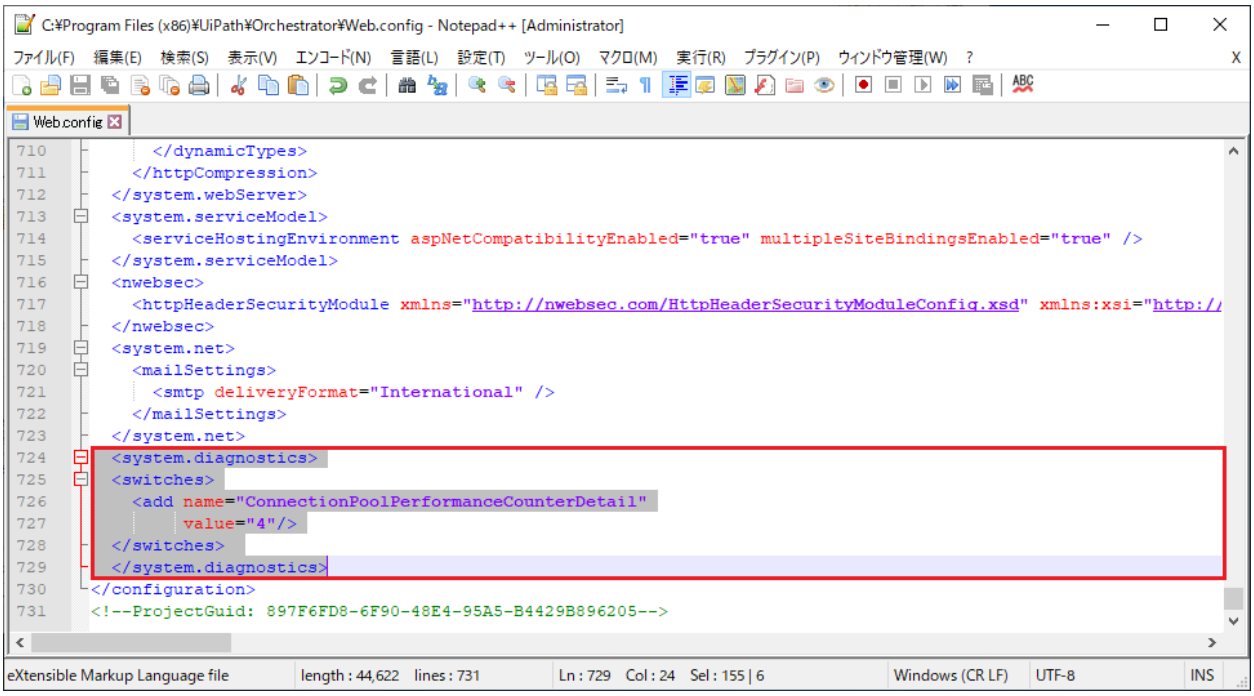
パフォーマンス カウンター	説明
NumberOfActiveConnections	現在使用中のアクティブな接続数
NumberOfFreeConnections	接続プール内の利用可能な接続数
SoftDisconnectsPerSecond	接続プールからプルされている 1 秒当たりのアクティブな接続数
SoftConnectsPerSecond	接続プールに戻されている 1 秒当たりのアクティブな接続数

方法

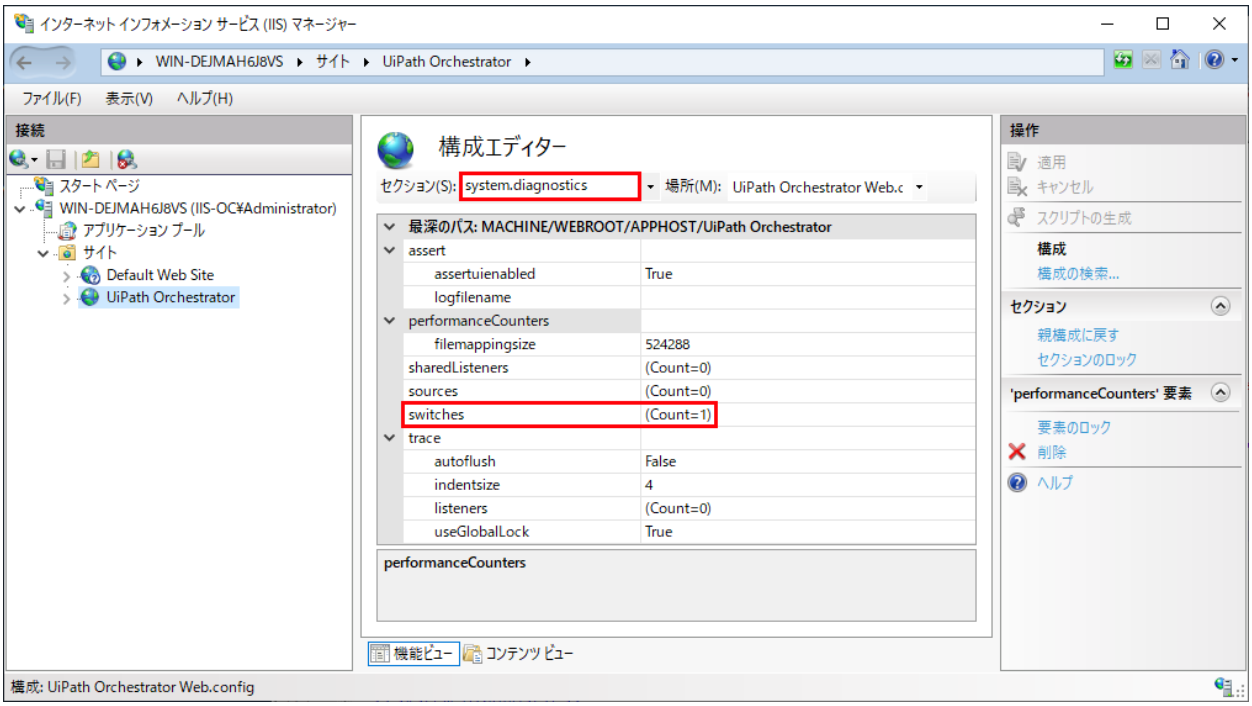
Orchestrator ディレクトリ配下の Web.config を下記のように編集することで機能を有効化できます。

(1) Web.config に次の XML スクリプトを追加します。

```
<system.diagnostics>
  <switches>
    <add name="ConnectionPoolPerformanceCounterDetail"
      value="4"/>
  </switches>
</system.diagnostics>
```



(2) [IIS マネージャー] > [構成エディター]画面で、[UiPath Orchestrator]サイト > [system.diagnostics]セクション > [performanceCounters] > [switches]が「(Count=1)」になっていることを確認し、IIS のサービスを再起動します。



(3) [パフォーマンスモニター] > [カウンターの追加] 画面から、例えば [.Net Data Provider for SqlServer] オブジェクト配下の[NumberOfActiveConnectionPools]カウンター等を追加します。

参考

[12] Activating Off-By-Default Counters

3.3. IIS のウイルススキャン除外設定

ウィルス対策ソフトやバックアップソフト等の予期しない動作によって、IIS が動作するために必要なモジュールやファイルの処理時に影響が発生し、後述の事例のような現象が発生する場合があります。IIS Server の安定稼働のため、IIS および Orchestrator に対して除外設定を検討することが推奨されます。

優先度：必須

検討タイミング：運用後

推奨

ウイルススキャンとよって特定のファイルが握られると TimeOut エラー等の原因と成り得るため、IIS の観点から下記のファイルをスキャン対象から除外することが推奨されます。

除外対象	既定のディレクトリのパス	説明
IIS のインストールフォルダ	%SystemRoot%\System32\inetsrv およびその配下	IIS が動作するために必要なモジュールや、構成ファイルなどの情報が当該フォルダに含まれます。必要なモジュールや構成ファイルへのアクセスが阻害され得る場合、IIS が正常に動作しない恐れがあるため、除外が推奨されます。
IIS 圧縮ファイル キャッシュ	%SystemDrive%\inetpub\temp\IIS Temporary Compressed Files\UiPath Orchestrator	IIS が実行されているサーバーで圧縮を有効にすると、IIS 圧縮ディレクトリへの HTTP 要求を行ったときに、要求したファイルではなく 0 バイトのファイルが返される場合があります。この現象は、HTTP 静的圧縮が有効になっている場合にのみ発生します。
IIS のログ、構成ファイル格納フォルダ	%SystemDrive%\inetpub およびその配下	IIS が処理を実行する際に利用される一時的なファイルの保管場所となるフォルダである。当該フォルダへのアクセスが阻害され得る場合、同様に IIS が正常に動作しない可能性があるため、除外が推奨される。
HTTPERR ログフォルダ	%SystemRoot%\System32\LogFiles\HTTPERR およびその配下	HTTP の要求を受け付ける、http.sys がエラーを出力するフォルダである。エラー内容がトラブルシューティングに活用され得るため、アクセスが阻害されてエラーが記録されないことを防ぐために、除外が推奨されます。
.NET Framework フォルダ	%SystemRoot%\Microsoft.NET\Framework{|64}\<.NET Framework のバージョン> およびその配下 ※ {x86|x64}で異なります。	ASP.NET がアプリケーションをコンパイルして生成したアセンブリ (dll) を保持するためのフォルダであります。該当フォルダ内のファイルがウイルススキャン等によって何らかの変更が加えられた場合、アプリケーションの再起動が発生するため、除外が推奨されます。
Orchestrator 関連ファイルが配置されているフォルダ	%ProgramFiles(x86)%\UiPath\Orchestrator およびその配下	Orchestrator の格納フォルダ。

事象例

下記以外にも、セキュリティ関連ソフトの影響により、予期しない挙動をする場合があります。ウイルス対策ソフトによっては、スキャンの除外対象のフォルダやファイルに対してもアクセスが発生してしまう場合があり、アンインストールしない限り、影響を抑えられなかったという事例もあります。除外や停止のみでは改善されない場合は、アンインストールにより事象が改善されるか否かも確認する必要があります。

- アプリケーションプールの強制再起動
  アプリケーションドメインの再起動は、Web.config などの構成ファイルの変更や、bin フォルダ配下のファイルの編集をトリガーとして行われ、再起動されると次回アクセス時にアセンブリのロード等が再度行われるため、他のタイミングよりリクエストの処理に時間を要します。セキュリティ関連ソフトの影響により、構成ファイルの変更が誤検知され、アプリケーションドメインが意図せず再起動してしまう場合もあり、アプリケーションに何も変更を加えていないにも関わらず、突発的にアクセスに時間を要する現象として見えることがあります。ASP.NET にてヘルスモニタリングのイベントを有効にしている場合、**イベント ID：1305** が記録されます。

- 構成ファイル作成時のアクセス拒否
  IIS や OS を再起動した後の初回アクセス等のタイミングで、普段は問題ないものの、低頻度で**イベント ID：5189** が記録され、構成ファイルを作成しようとした際にアクセス拒否が発生する場合があります。アプリケーション プール用の構成ファイル、既定では

  %SystemDrive%\inetpub\temp\appPools

  を作成しようとしたものの、アクセスが拒否されたために失敗したことで、該当のイベントが記録されます。アクセス拒否は一般的に当該ファイルにアクセス権がない場合に発生し、その場合は常時エラーとなります。しかし、本エラーが単発で記録される場合には一時的にアクセスが阻害された可能性があり、そのような場合にはセキュリティ関連ソフトの影響が考えられる。

- アプリケーション プールのプロセスの突然終了
  普段は問題なく動作しているものの、低頻度で**イベント ID：5009** が記録され、プロセスが予期せず終了する場合があります。イベントに記録されている終了コード 0xfffffffe は "CLEAN_WORKER_PROCESS_EXIT_CODE" を示し、異常終了を示すものではなく、正常にワーカープロセスが終了された場合に記録されます。Application Error イベント等も特に記録されていない場合、IIS にロードされているモジュールが何らかの要因により自発的に停止したと考えられます。このような場合、セキュリティ関連ソフトの動作が、IIS が動作するために必要なモジュールやファイルの処理に影響を及ぼし、プロセスの突然終了に一因となる可能性があります。

参考

[13] IIS 観点でアンチウイルス スキャン対象から除外したいフォルダ

4. SQL Server の設定

本章では、SQL Server 2016 環境構築時に、処理性能の最大化と突発的な性能問題の予防を想定として、考慮すべき事項を説明します。一部で SQL Server 2017 および SQL Server 2014 以前の環境における設定方法等の情報も記載しています。

4.1. SQL Server OS の設定

4.1.1. OS 特権の付与

SQL Server インスタンスの起動アカウントに付与する権限によって、SQL Server の挙動が変わります。

優先度：必須

検討タイミング：運用後

推奨

SQL Server インスタンスの起動アカウントに次の権限を付与します。

A) **ボリューム保守タスクを実行**（予約名：SeManageVolumePrivilege）
B) **メモリ内ページロック**（予約名：SeLockMemoryPrivilege）

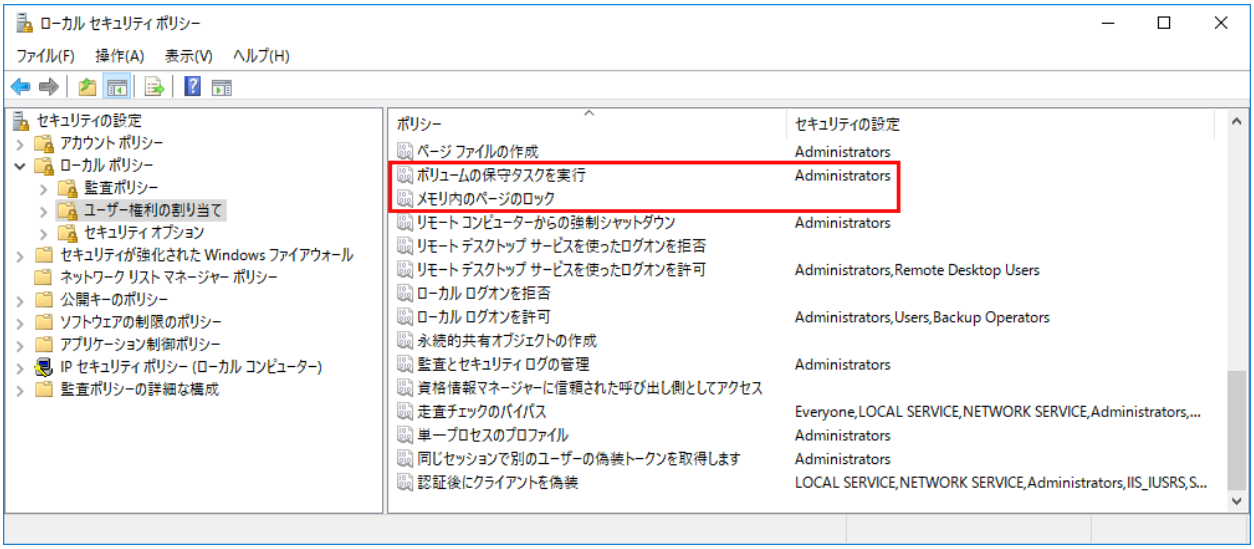
理由

A) SQL Server では、データファイルが瞬時に初期化され、ゼロイング（ゼロを書き込む処理）行われます。SQL Server インスタンスの起動アカウントに SeManageVolumePrivilege 権限を付与し、ファイルの瞬時初期化を有効化することで、追加／拡張／復元等のファイル操作の実行時に必要な領域への**ゼロイングを回避して**、使用中のディスク領域の返還要求のみを行うため、ディスク領域の確保が高速化されます。代わりに、新しいデータがファイルに書き込まれる際に、ディスクの内容が上書きされます。
   ※ データファイルが追加、拡張、復元されたり、ディスクに以前削除したファイルの残存データがある場合に、新規領域のゼロ埋め処理（ゼロイング）によるファイル初期化が実行されます。この初期化は処理コストが高く、処理中は更新処理が排除されます。ただし、データを初めてファイルに書き込む際には、オペレーティングシステムはファイルをゼロイングしません。この初期化を瞬時化するためのオプションがファイルの瞬時初期化です。セキュリティに起因して既定ではゼロイングされるが、有効にするとゼロイングがスキップされ、ファイルの初期化が瞬時に達成されます。
B) 連続的なメモリ領域を確保し（メモリの断片化を防ぎ）、ディスク上の仮想メモリへの**ページングを防止**すること (ページロック) で、不用意な性能劣化を防ぐことができます。

方法

1) [SQL Server 構成マネージャー]で、SQL Server サービスを実行しているアカウントを確認します。

2) secpol.msc（ローカルセキュリティポリシー）を起動します。
3) 左ペインから [セキュリティの設定] > [ローカル ポリシー] > [ユーザー権利の割り当て] を選択します。
4) A) については「ボリュームの保守タスク実行」ポリシーを、B) については「メモリ内ページロック」ポリシーを付与します。実行権限は Windows のユーザーに対して付与されるので、反映には SQL Server の再起動が必要になります。一度 SQL Server のサービスを停止することになりますので、メンテナンス時間帯に実施します。



事象例

SQL Server ログに「Cannot use Large Page Extensions: lock memory privilege was not granted.」と表示されている場合、メモリの連続領域が確保されていない状態で動作しているため、ローカルポリシーを見直す必要があります。正しく設定されていれば、ログには「Large Page Granularity: 2097152」(2MB) 及び「Large Page Extensions enabled.」と表示され、プロセスは 2MB 毎の物理的に連続したメモリ領域を確保していることが確認できます。

<SQL Server ログ>
C:\Program Files\Microsoft SQL Server\導入済製品／バージョン\MSSQL\Log
SQL Sever 2016 Express の場合、導入済製品／バージョンの部分は、MSSQL12.MSSQLSERVER2014 です。

備考

<ファイルの瞬時初期化について>

トランザクション ログファイルは、シーケンシャルで循環的に利用される特性がありますが、これに起因して、データベースのクラッシュリカバリ時にログの終端位置を識別するために常にゼロイングが必須です。トランザクション ログファイルに関してはファイルの瞬時初期化ができません。

SQL Server 2016 以降では、Azure BLOB ストレージにデータベースファイルを配置できます。この場合、Azure BLOB ストレージで瞬時初期化が利用でき、クラッシュリカバリ時のログの終端位置を見失うことなく、ログファイルについても瞬時初期化が可能です。

参考

[14] SQL Server 2016 環境構築時のパフォーマンスに関するベストプラクティス
[15] Database File Initialization

4.1.2. ストレージ構成

ディスクパフォーマンス性能をチューニングする際にアロケーションユニットサイズを変更する場合があります。特に RAID5 を使用しているシステムはこの変更が有効です。

コマンドプロンプトを「管理者として実行」で開き、「fsutil fsinfo ntfsinfo <Drive Dame>」を実行することで、現在のフォーマット形式やアロケーションユニットサイズを確認できます（下図を参照）。

```
※ PowerShell（管理者）で実行
PS C:\WINDOWS\system32> fsutil fsinfo ntfsinfo <Drive Name>
NTFS ボリューム シリアル番号          : 0xd20a8e4d0a8e2f13 ← NTFS 固有の情報
NTFS バージョン                      : 3.1 ← NTFS 固有の情報
LFS バージョン                       : 2.0
セクター数                           : 0x000000003b7627ff
総クラスター数                       : 0x00000000076ec4ff
空きクラスター数                     : 0x0000000001cc909d
総予約数                             : 0x0000000000001803
セクターあたりのバイト数             : 512
物理セクターあたりのバイト数         : 4096
クラスターあたりのバイト数           : 4096 ← アロケーションユニットサイズ(GB)
ファイル セグメントあたりのバイト数  : 1024
ファイル セグメントあたりのクラスター数 : 0
MFT の有効なデータ長                 : 0x000000003b4c0000
MFT 開始 LCN                         : 0x00000000000c0000
MFT2 開始 LCN                        : 0x0000000000000002
MFT ゾーン開始                       : 0x0000000001df8b00
MFT ゾーン終了                       : 0x0000000001e01360
デバイスの最大トリム エクステント数  : 256
デバイスの最大トリム バイト数        : 0xffffffff
ボリュームの最大トリム エクステント数 : 62
```

```
ボリュームの最大トリム バイト数        : 0x40000000
リソース マネージャー識別子:      22F9B61B-2539-11E8-AD9F-8C16453DD20B
```

優先度：必須

検討タイミング：運用前

推奨

データファイル、ログファイルを配置するディスクは、以下の設定にてフォーマットします。

設定項目	既定値	推奨設定値
フォーマット形式	NTFS	NTFS
アロケーション ユニット サイズ	ディスク サイズにより規定値が異なります。	64KB（= 1 エクステント）またはこの整数倍

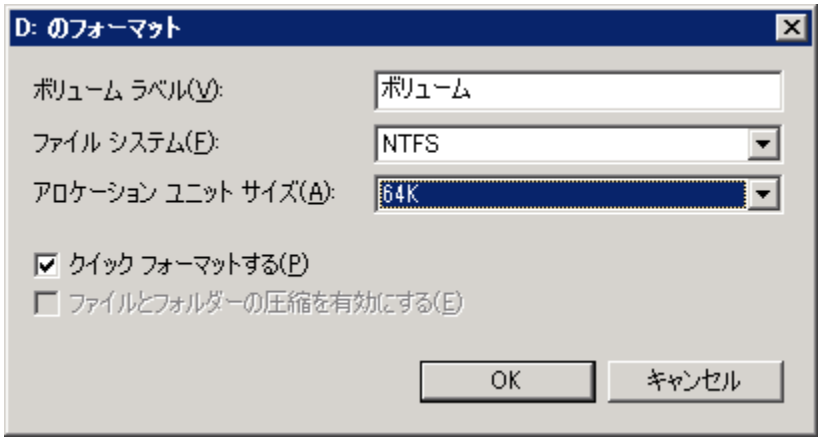
SQL Server はディスク I/O がボトルネックになる場合が多いため、HDD よりも SSD の方が推奨されます。

理由

SQLServer のデータファイルは、1 ページ=8KB であり、8 ページ（64KB）が 1 エクステントです。データはこのエクステントの単位で読み込まれるため、NTFS のアロケーションユニットサイズを 64KB（x 整数倍）に設定することで、アクセス効率が向上します。

方法

パーティション内のデータはすべて消えるため、バックアップする必要があります。「コンピュータの管理」から「ディスクの管理」を選択し、フォーマットしたいパーティションを右クリックメニューで「フォーマット」を選択し、設定します。



備考

Windows で使用されるすべてのファイルシステムは、アロケーションユニットサイズ（クラスターサイズ）に基づいてハー

ドディスクに整理されます。アロケーションユニットサイズはファイルを保持するために使用できるディスク領域の最小量を表します。アロケーションユニットサイズの倍数にファイルのサイズが達していると場合、ファイルを保持するために追加の空き領域を（次のクラスターサイズの倍数まで）使用しなければなりません。アロケーションユニットサイズが指定されていないパーティションをフォーマットする場合、パーティションのサイズに基づいて既定値が選択されます。これらの既定値は、失われる領域を減らし、パーティションで発生する断片化を抑制します。

参考

[16] Fsutil fsinfo
[17] SQL Server 2016 環境構築時のパフォーマンスに関するベストプラクティス

4.2. SQL Server インスタンスの設定

4.2.1. メモリの設定

SQL Server のメモリ設定は、基本的に「**最小サーバーメモリ**」と「**最大サーバーメモリ**」の 2 つしかありませんが、各々の意味を正しく理解しないと、不用意な問題が発生する可能性があるため、注意をする必要があります。
（「最小サーバーメモリ」、「最大サーバーメモリ」の意味については、後述の [備考] セクションを参照。）

優先度：必須

検討タイミング：運用後

推奨

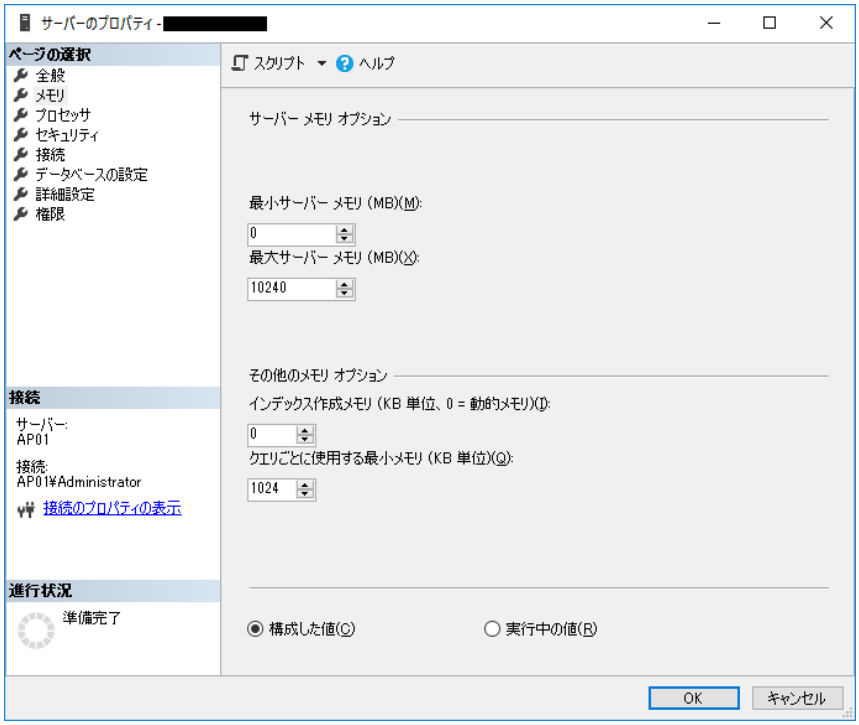
A) 最小サーバーメモリについて、SQL Server 単体かつインスタンスが一つの物理サーバーでの運用では既定値の 0MB とします。
B) 最大サーバーメモリについて、OS 用に 2GB 程度確保し、他にメモリを大きく使うアプリケーションとの共存がなければ、残りを SQL Server に充てます。

理由

B) SQL Server が稼働しているとメモリ使用が増え続けます。その際に他のプログラム・サービスとメモリ使用の競合や取り合いを起こさせないために設定する必要があります。

方法

1) SSMS を起動して該当サーバー名を右クリックします。
2) [プロパティ]を選択し、[サーバーのプロパティ]ウィザードを開きます。
3) [メモリ]ページから最小／最大サーバーメモリ（単位は MB）を設定し、[OK]ボタンを押下します。



設定後のサーバーへの反映にはサービスの再起動を必要としません。また、例えば次のクエリを実行することで、最大サーバーメモリを 4GB に設定することができます。

<SQL>

```
-- SQL 文
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'max server memory', 4096;
GO
RECONFIGURE;
GO
```

備考

<最小サーバーメモリ (min server memory)>

この値を下回る場合、SQL Server はメモリを解放しません。他のアプリケーションの利用によって OS からメモリ開放要求があったとしても、この値を下回って解放することはありません。ただし、SQL Server の起動時にすべてのメモリ領域を確保する訳ではなく、この値よりも小さいメモリサイズで稼働している場合もあります。

<最大サーバーメモリ (max server memory)>

SQL Server が確保する最大メモリサイズ。既定値 (2,147,483,647MB) の設定では、物理的な空きメモリが残り 4MB から 10MB の間になるまでメモリを確保しようとします。他のアプリケーションとの相乗り環境では、SQL Server に割り当てるメモリサイズを明示的に指定する必要があります。**規定値では OS とメモリ競合を起こす可能性があ**

るため非推奨です。

参考

[18] SQL Server 2016 環境構築時のパフォーマンスに関するベストプラクティス

4.2.2. クエリ並列処理度数の設定

クエリ並列処理度数は **MAXDOP (MAX Degree Of Parallelism)** で設定します。MAXDOP は、1 ユーザーコネクションで複数のスレッドを用いた並行処理を許すかどうかを制御するパラメータです。

優先度：任意

検討タイミング：運用後

推奨

処理特性によって必要となる並列処理度数（規定値は 0 ）が異なるため、パフォーマンステストによるチューニングの検討を要します。下記の推奨設定はあくまで目安であり、パフォーマンステスト等を実施の上で変更をご検討ください。

A) OLTP 処理のトランザクションの同時実行処理を重視する場合
   MAXDOP = 物理 CPU コア数 x (1/4~1/2)
B) バッチ処理を重視する場合
   MAXDOP ≠ 1
C) CPU のマルチコア環境による NUMA 構成の場合
   MAXDOP ≤ 各 NUMA ノードを構成している物理 CPU コア数以下の値

理由

既定値の「0」は全ての CPU を利用可能という意味です。例えば、あるクエリを処理する際に、「A、B という処理を並列で行って、その結果を C で処理し、次に D、E で処理し、…（後続処理に続く）」という実行計画に最適化されたとします。「MAXDOP＝0」の場合、親スレッドを「01」として、ほかのスレッドを「02」「03」という「子スレッド」として使えるため、この並列処理は効率的に見えますが、並列処理特有のデータ交換、同期によるオーバーヘッドも生じるために、サーバー全体のパフォーマンスが劣ってしまう場合があります。この場合、既定値の「0」では SQL Server のパフォーマンスを十分に発揮できない可能があります。

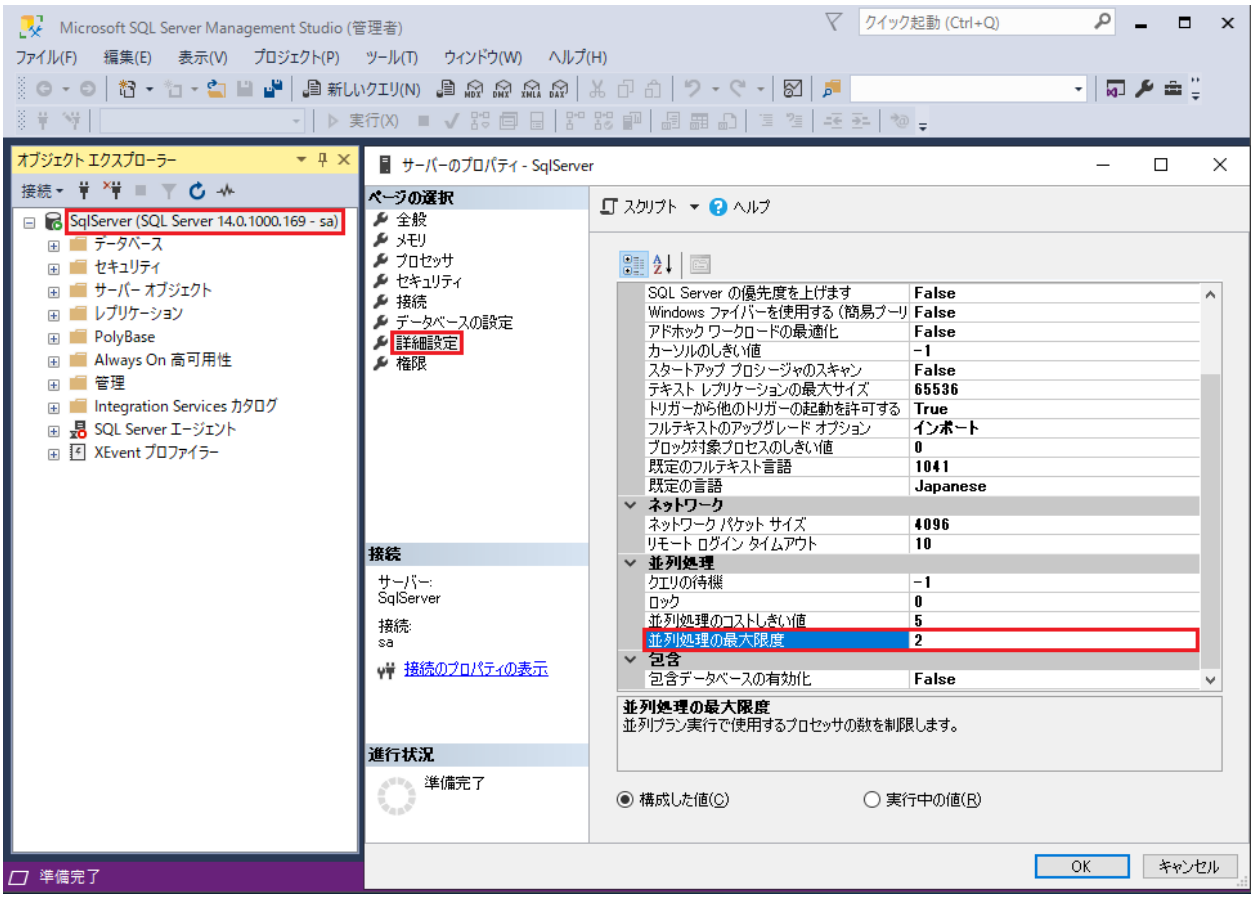
並列プランの場合、基本的には使用するプロセッサ数が多いほど CPU 負荷は上昇するが、実行速度が速くなる傾向があるため、MAXDOP の設定値はサーバー負荷と実行速度とのトレードオフとなっています。

方法

次の例では、**sp_configure ストアドプロシージャ** を使用して、max degree of parallelism オプションを 2 に設定する方法を示します。MAXDOP の設定変更では SQL Server の再起動は不要なため、オンライン処理の時間帯とバッチ処理の時間帯で設定を変更できます。

<GUI>

[SSMS] > [サーバーのプロパティ] > [詳細設定] > [並列処理] > [並列処理の最大限度] から設定します。



<SQL>

```
-- SQL文
USE <DATABASE> ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE WITH OVERRIDE;
GO
EXEC sp_configure 'max degree of parallelism', 2;
GO
RECONFIGURE WITH OVERRIDE;
GO
```

備考

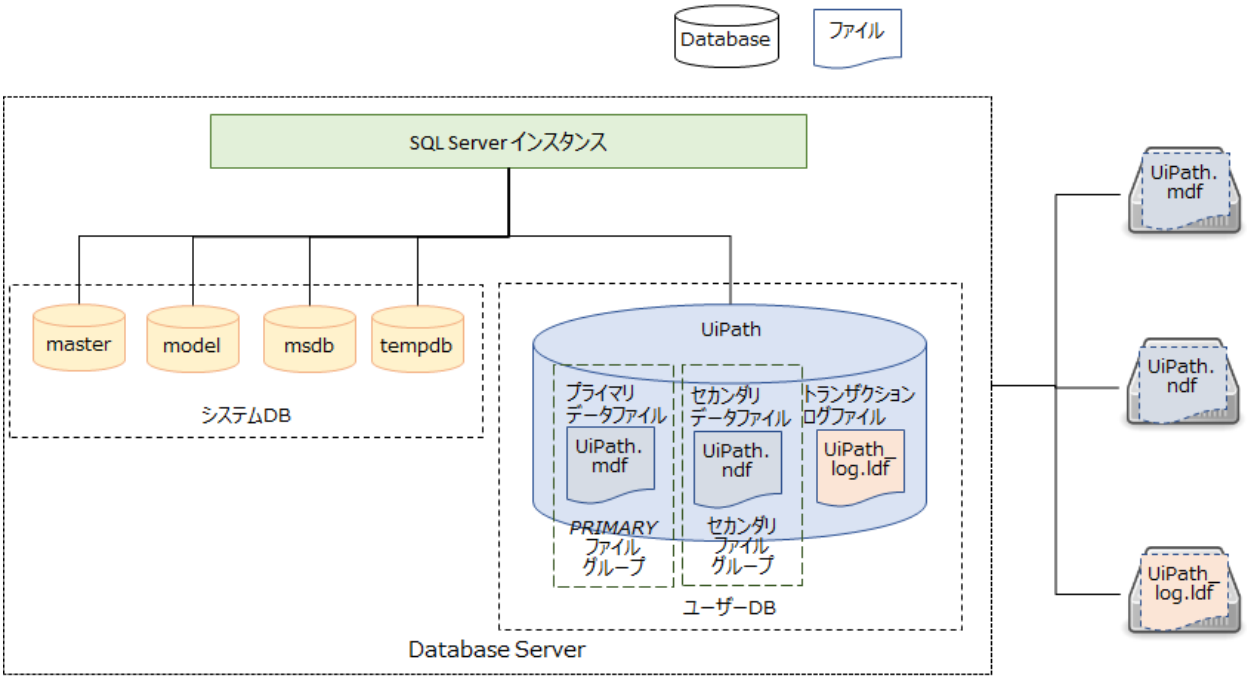
MAXDOP	効果	説明

0（規定値）	並列処理が有効なクエリにおいて、サーバーに搭載されている全ての論理 CPU コアを用いてクエリを並列実行します。	この設定はバッチ処理には適していますが、OLTP 処理のようにトランザクション処理の同時実行性を重視する環境には適しません。
1	並列処理を行いません。	少なくともクラウド環境であれば MAXDOP≠0 とします。
2 以上	最大並列度を指定します。	N/A

参考

[19] SQL Server 2016 環境構築時のパフォーマンスに関するベストプラクティス
[20] Configure the max degree of parallelism Server Configuration Option

4.3. データベースの設定



4.3.1. データファイルの分散配置

ストレージへのデータベース配置を I/O 特性や性能要件を踏まえて分散させます。UiPath データベース（既定名：UiPath）は製品仕様上一つです。

優先度：任意

検討タイミング：運用後

推奨

A) トランザクション ログファイルをデータファイルと異なる物理デバイスのストレージに配置します。
B) tempdb データベースを可能な限り高速な I/O デバイスに配置します。
C) I/O デバイスが同じでもデータファイルを論理的に分割します。データファイル数は 8 または CPU コア数 の小さい方の値を設定します（SQL Server 2016 以降は自動で設定されます)。

理由

[UiPath] データベースを使用していることを前提とします。

A) SQL Server のデータベースを構成するデータファイルとトランザクション ログファイルは、前者がランダムアクセス、後者がシーケンシャルアクセスという異なる I/O 特性を持つため、それらが同じ物理デバイスに配置されると、磁気ヘッドの移動効率が劣化します。また、ログファイルの I/O 性能はデータベース性能に大きく影響するため、性

能の観点でも独立配置が推奨されます。SSD 等のフラッシュデバイスの場合、磁気ヘッドを持たないため、異なる I/O 特性の混在による I/O 効率劣化は大きく低減されています。

ファイル タイプ	I/O 特性	推奨デバイス
データファイル	ランダム アクセス	SSD
トランザクション ログファイル	シーケンシャル アクセス	SSD

B) データベースにはユーザーが作成するデータベースとは別に SQL Server が内部的に利用するシステムデータベース (tempdb, msdb, model, master) があります。tempdb システムデータベースは、SQL Server のインスタンスまたは SQL Database に接続しているすべてのユーザーが使用できるグローバルリソースです。一時テーブルやテーブル変数等のユーザーオブジェクトの格納、作業（結合、集計、ソート、インデックス再構築等）のための内部オブジェクト格納、バージョンストアの格納等のグローバルな共有リソースとして用途が多岐に亘ることから、パフォーマンスのボトルネックとなる場合があるため、より高速な I/O デバイスに配置することが推奨されます。

C) データファイルの管理領域へのアクセスの競合がパフォーマンスボトルネックになることを回避することを目的とした分割であるため、ディスク（I/O デバイス）が同じでも効果があります。当然、異なるディスクにすれば負荷分散も図れるため最良です。

方法

A) データファイルとトランザクション ログファイルを異なるボリュームへ配置するクエリ例を次に示します。

```
-- SQL 文
CREATE DATABASE [UserDB]
  ON PRIMARY
    (NAME = N'data', FILENAME = N'Y:\datafile.mdf')
  LOG ON
    (NAME = N'log',  FILENAME = N'Z:\xlogfile.ldf')
GO
```

B) tempdb データベースを既定の場所から異なるボリュームへ移動させるクエリ例を次に示します。

```
-- SQL 文
ALTER DATABASE tempdb MODIFY FILE
  (NAME = N'tempdev' ,FILENAME = N'T:\tempdb.mdf')
ALTER DATABASE tempdb MODIFY FILE
  (NAME = N'templog' ,FILENAME = N'L:\templog.ldf')
GO
```

SQL Server を再起動し、元の場所にあったデータファイルとログファイルをファイルシステムから手動で削除する必要があります。

4.3.2. ファイルグループの分割

ファイルグループは複数のデータファイルを束ねる論理的な管理上の境界です。Oracle で云うところの「表領域」に近い概念です。データベース毎に 1 つの PRIMARY ファイルグループが存在し、必要に応じてユーザー定義のファイルグループを追加できます。データベース配下に作成するテーブルやインデックス等のオブジェクトは、CREATE 時に自身

の配置場所を指定しますが、そこで指定するのはデータファイルではなく、ファイルグループです。

優先度：任意

検討タイミング：運用後

推奨

オブジェクト単位で I/O を制御したい場合は、インデックスを実データ（ヒープもしくはクラスタ化インデックス）と分けて配置します。

理由

表データの更新に伴って索引の更新も必要となることから、それらの配置を分割することで I/O 効率の改善を図ります。バックアップについても、バックアップ対象データの読み込み (Read) とバックアップファイルへの書き込み (Write) が同時に処理されるため、それらの配置を分割することで I/O 効率の改善が期待されます。

方法

1) インデックス用のファイルグループを持つデータベースを作成します。

```
-- SQL 文
CREATE DATABASE [UiPath]
  ON PRIMARY
    (NAME = N'data',  FILENAME = N'D:\datafile.mdf'),
  FILEGROUP [FG_INDEX]
    (NAME = N'index', FILENAME = N'I:\indexfile.ndf')
  LOG ON
    (NAME = N'log',   FILENAME = N'L:\xlogfile.ldf')
GO
```

2) テーブルを PRIMARY ファイルグループに配置します。

```
-- SQL 文
USE [UiPath]
CREATE TABLE [TABLE1] (C1 INT) ON [PRIMARY]
GO
```

3) インデックスは FG_INDEX ファイルグループに配置します。

```
-- SQL 文
CREATE INDEX [INDEX1] ON [TABLE1](C1) ON [FG_INDEX]
GO
```

参考

[21] Database Files and Filegroups

4.3.3. データファイルを複数の I/O デバイスに配置

ストレージの性能を余すことなく極力均一に利用するためにファイルを分割し、外部要因の性能改善策とします。

優先度：任意

検討タイミング：運用後

推奨

複数の I/O デバイスを利用できる場合、ファイルを分割配置することで負荷を平準化します。既にストライピングが十分に機能している場合、必ずしも本推奨設定を適用する必要はありません。

理由

ファイルグループを複数のデータファイルで構成すると、SQL Server はその各ファイルの空き領域に応じて均等にエクステントを割り当てようとします。このとき、各データファイルが異なるドライブに配置されていると、ストライピングが機能し、I/O が平準化されてパフォーマンスの向上が見込まれます。

SQL Server のパフォーマンスが大量のログ等により低下すると、送信ログの重複等の Orchestrator の予期せぬの不具合を招く可能性があります。

備考

<ストライピング（英: striping）>
複数台のハードディスクにデータを分散して書き込み、読み書きを高速化する方法です。ただし、1 つのハードディスクが故障するとすべてのデータが使えなくなるため、耐障害性が低いことから、RAID には含まれず、RAID レベルの RAID 0 と呼ばれます。

4.3.4. トランザクション ログファイルの構成

トランザクション ログファイルは SQL Server のようなリレーショナルデータベースの原子性と永続性を保証するための根幹となる最も重要な要素の一つです。

優先度：任意

検討タイミング：運用前

推奨

トランザクション ログファイルを**単一ファイルで構成**する。ログファイルの性能を向上させたい場合には、単一ファイルをより高速で高可用な I/O ドライブに配置します。

理由

トランザクション ログファイルはシーケンシャルにアクセスするファイルであるため、複数ファイルで構成してもラウンドロビンで直列利用されるだけで性能向上は見込めないどころか、悪影響を及ぼす場合もあります。複数で構成されたログファイルが自動拡張した場合、各々のログファイルの拡張領域は交互に埋まるため断片化を引き起こします。結果として、ログの読み取り操作でファイル間の往来が必要になり、性能劣化を誘発します。さらに、実際にはログの切り捨てによる再利用も行われるため、より複雑な断片化が引き起こされます。単一ファイルで自動拡張が発生しないように制御できれば、ログファイルの再利用が行われても断片化することはありません。

4.3.5. データベースファイルの初期サイズ

Robot や Orchestrator の運用方法によって初期サイズの適正値は異なるが、モニターの上で可能な限り余裕を持たせた設定値を検討します。

優先度：必須

検討タイミング：運用後

推奨

A) すべてのデータベースファイル（データファイルとトランザクション ログファイル）の初期サイズを運用期間中に拡張が必要とならない十分な大きさで設定します。
B) データファイルを分割する場合、分割された各ファイルの初期サイズを統一させます。

理由

A) データファイルの領域が不足した場合に自動拡張の設定に従ってファイルのサイズ拡張（ファイル拡張）が行われます。この拡張はオーバーヘッドを発生させ、突発的かつ全体的な性能劣化を招くため、極力避けるべきです。
B) 各ファイルの空き領域に応じて順次割り当てが行われるため、各ファイルの初期サイズが統一されていないと、各ファイル毎のデータ量の均衡が崩れ、サイズの大きいファイルに I/O が偏ってしまいます。初期サイズが統一されていれば、均等にデータが分散され、ストライピングが効果的に機能します。

4.3.6. データファイルの自動拡張サイズ

前提として、データファイルの（自動）拡張は何か想定外の状況が発生した場合の非常対策であり、極力回避すべきである。UiPath Orchestrator の運用面・性能面でも悪影響を与える恐れがあるため、万一のためにも適切な自動拡張サイズの設定が重要となります。自動圧縮は既定で無効です。

ファイル拡張処理中はクエリ実行が待機状態となって突発的なレスポンス低下を招きます。例えばトランザクション ログファイルの拡張が発生すると、そのログファイルに書き込む必要がある他のトランザクションも拡張処理が完了するまで待機状態を強いられます。

優先度：必須

検討タイミング：運用後

推奨

自動拡張を有効にして、割合（パーセント）ではなくサイズで指定します。

<自動拡張サイズの見積もり>

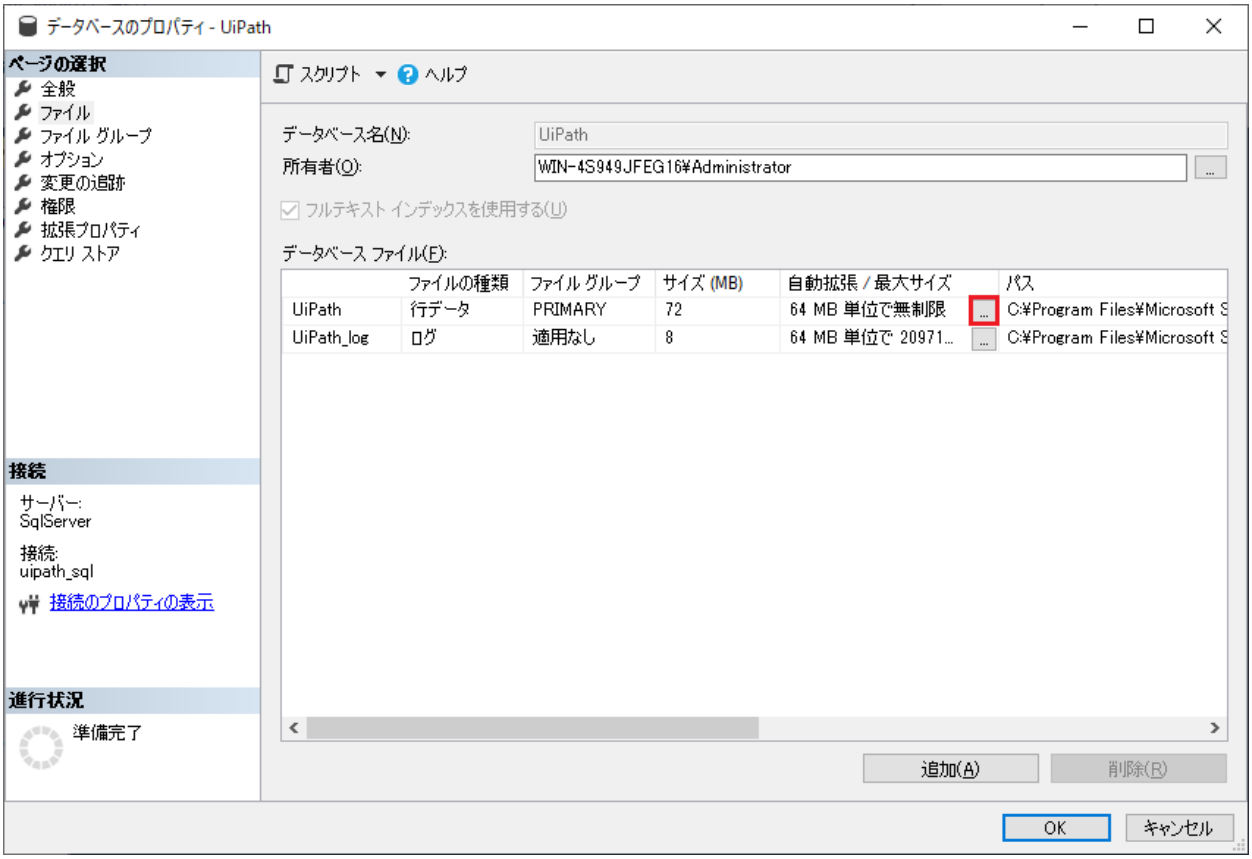
「ファイル拡張に要する時間 ＋ クエリ実行時間 < クエリタイムアウト時間」が成り立つように設定します。一先ずの目安は 1GB 程度です。

理由

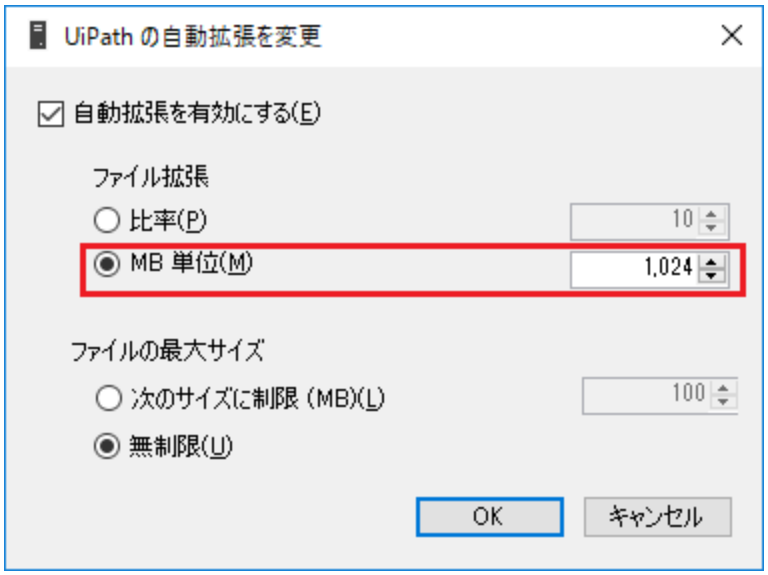
自動拡張が無効の状態では、データファイルがフルになって SQL Server が起動不可となったケースが報告されています。

方法

i.) [データベースのプロパティ] > [データベース ファイル (E)] （ここでは UiPath） > [自動拡張 / 最大サイズ]列から編集します。



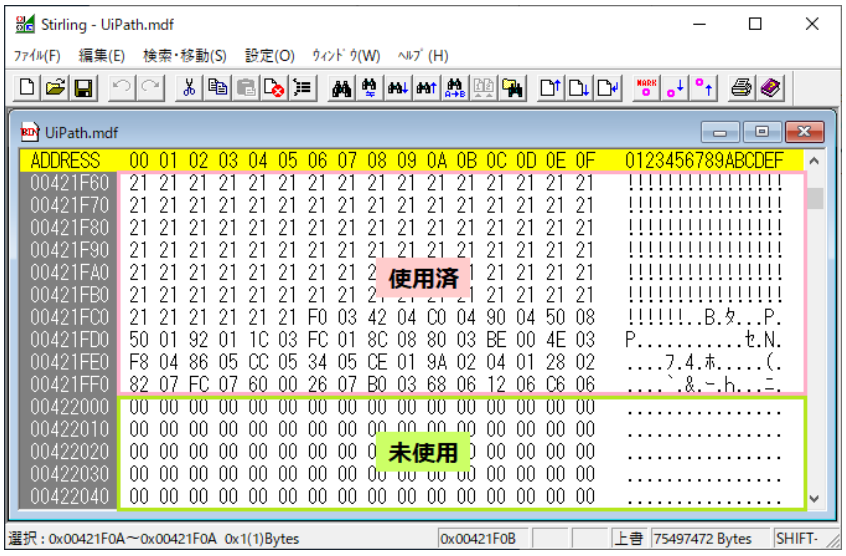
ii.) 該当箇所を設定し、[OK]ボタンを押下します。



備考

SQL Server 2014 以前では、SQL Server のインストール後に、tempdb のベストプラクティスに則した構成変更を個別に行わなければならなかったが、SQL Server 2016 では、インストール時にハードウェア構成を自動的に判断して、最適な構成で tempdb を構成できるようになりました。tempdb は再起動しない限り膨らみ続けます。

データベースの「大きさ」を測る指標として、ファイルサイズ、実データサイズ、レコード件数があります。データベースを構成するファイル（データファイルとトランザクション ログファイル）のファイルサイズは、ファイルのプロパティから確認できる”サイズ”に等しくなります。実データサイズは前以て大きく確保されたファイルの中でデータベースが実際に使用している領域のサイズです。次の図はデータファイルのイメージである。未使用部分はゼロイングされています。



ファイル拡張の逆の振る舞いになるファイル圧縮については、手動（DBCC SHRINKDATABASE や DBCC SHRINKFILE）、自動（AUTO_SHRINK）ともに非推奨です。ここで云う「圧縮」とは ZIP 圧縮のような符号

化ではなく、ファイルの不要領域（ゼロイングされた領域）を OS に返却することを指します。細々とした拡張、拡張と圧縮の繰り返しは断片化の原因となるため、性能面の観点から非推奨です。

自動拡張の上限に到達した、もしくはファイルを配置したディスク領域の上限に到達し、領域不足となるとデータベースへの書き込みエラーが発生します。

参考

[22] SQL サーバーの「自動拡張」および「自動圧縮」の設定に関する考慮事項

4.3.7. 複数のデータファイルの同時拡張

優先度：必須

検討タイミング：運用後

推奨

複数データファイル構成において、データファイルの自動拡張の**同時拡張**を有効にします。各ファイルのデータ量の偏りを抑制し、ファイルへの I/O の均衡を保つことで、ストライピング機能 (RAID 0) を向上させます。

理由

複数データファイル構成において、すべてのファイルがフルになると自動拡張が走りますが、既定ではファイル単位で順次拡張されます。時として、拡張されたファイルに I/O が偏り、ストライピングが十分に機能しなくなる恐れがあります。

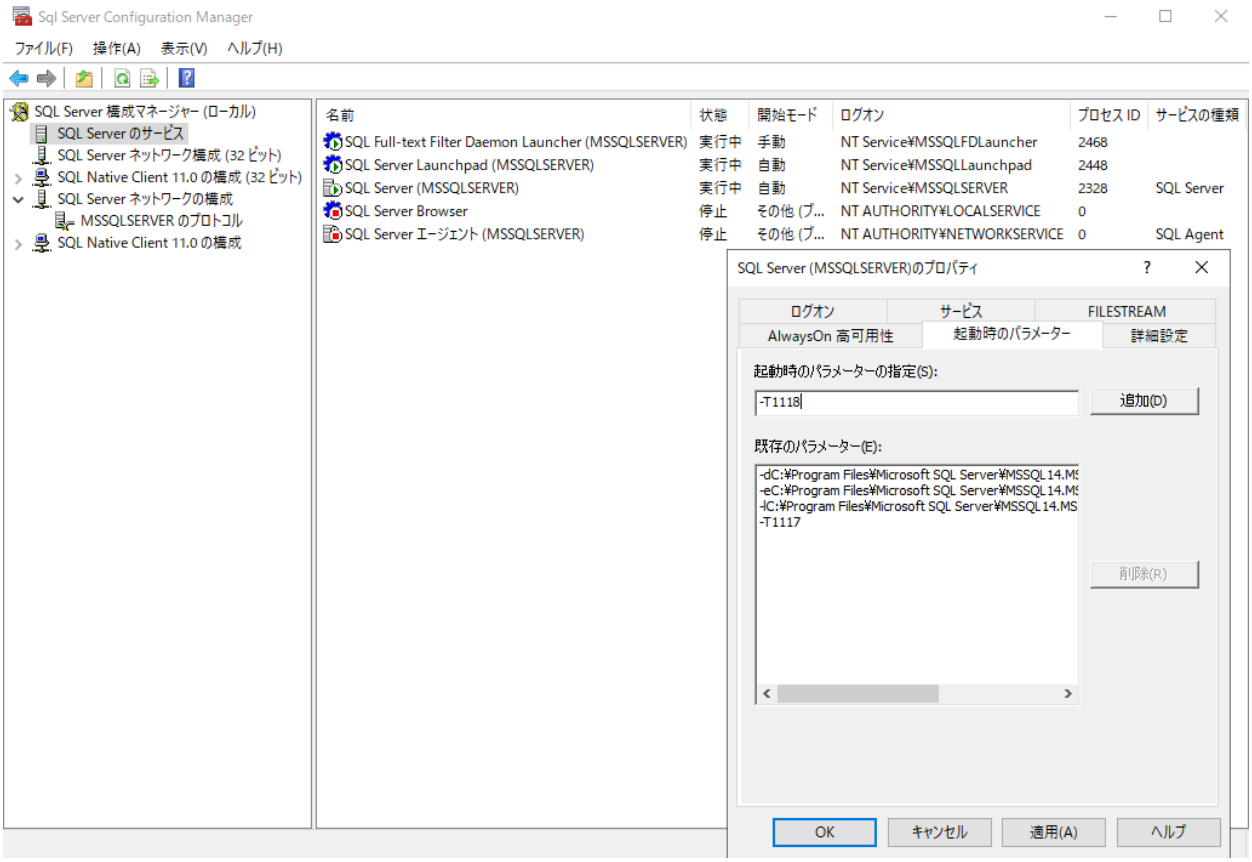
方法

<SQL Server 2016 以降>

```
-- SQL 文
USE <dbname>
GO
-- 同時自動拡張: ON
ALTER DATABASE <dbname> MODIFY FILEGROUP <filegroup> AUTOGROW_ALL_FILES
GO
-- 同時自動拡張: OFF
-- ALTER DATABASE UserDB MODIFY FILEGROUP <filegroup> AUTOGROW_SINGLE_FILE
-- GO
-- 設定確認
SELECT
  name, is_autogrow_all_files
FROM
  sys.filegroups WHERE name = '<filegroup>'
GO
```

<SQL Server 2014 以前>

トレースフラグ 1117 を有効化します。SQL Server 構成マネージャを開き、SQL Server サービスの [プロパティ] ウィザードの [起動時パラメーター] タブから設定します。[起動時のパラメーターの指定(S)] 欄に次の文字列「-T1117」を追加します。



備考

ログファイルについては、ファイルへの I/O がシーケンシャルアクセスでストライピング無しのため、同時自動拡張の意味はなく、オプションの ON/OFF に関わらず制御対象外です。

AUTOGROW_ALL_FILES データベースオプションを有効に設定することで、すべてのファイルが同時に拡張されるようになり、引き続きストライピングが機能します。このオプションは SQL Server 2016 以降でサポートされたデータベースオプションであり、tempdb では既定で有効、ユーザーDB では既定で無効になっています。この制御は SQL Server 2014 以前のトレースフラグ 1117 と等しくなります。ただし前者はファイルグループ単位で、後者はインスタンス単位での設定となります。また、SQL Server 2016 以降ではトレースフラグ 1117 は無力化され、オプションによる制御が必要です。

トレースフラグは SQL Server のデータベースエンジンに備わる『内部動作を変更する機能』です。トラブルシューティングにおいて、パフォーマンス問題の解決や SQL Server 自身の動作のデバッグで有効な手段となります。

4.3.8. データファイルの分割

データベース内にテーブル等のオブジェクトを新規作成／削除すると、管理情報の更新が必要となります。この処理の多重度が増すと管理情報へのアクセスに対するラッチ競合によって性能劣化を起こします。この管理情報は各ファイルで保有されています。

ラッチ（英: Latche）はページのデータ更新時にページヘッダに対して瞬間的に排他制御をかけます。一方、ロックはコミット命令によってトランザクションが完了するまで保持される排他制御です。

SQL Server のアーキテクチャに関係するファイルを分割し、内部要因の性能改善策とします。特に tempdb のデータファイルは SQL Server に割り当てられた論理 CPU 数に応じて、以下のように分割することが一般的な推奨事項になっています。論理 CPU 数は [タスクマネージャ] > [パフォーマンス] タブ > [CPU] から「論理プロセッサ数」で確認できます。

優先度：任意

検討タイミング：運用後

推奨

tempdb のデータファイルを分割することで管理情報を分割し、管理情報のラッチ競合を緩和させます。

A) 論理 CPU 数 < 8
論理 CPU 数で分割します。例えば、論理 CPU が 4 つの場合、tempdb のデータファイルを 4 つに分割します。

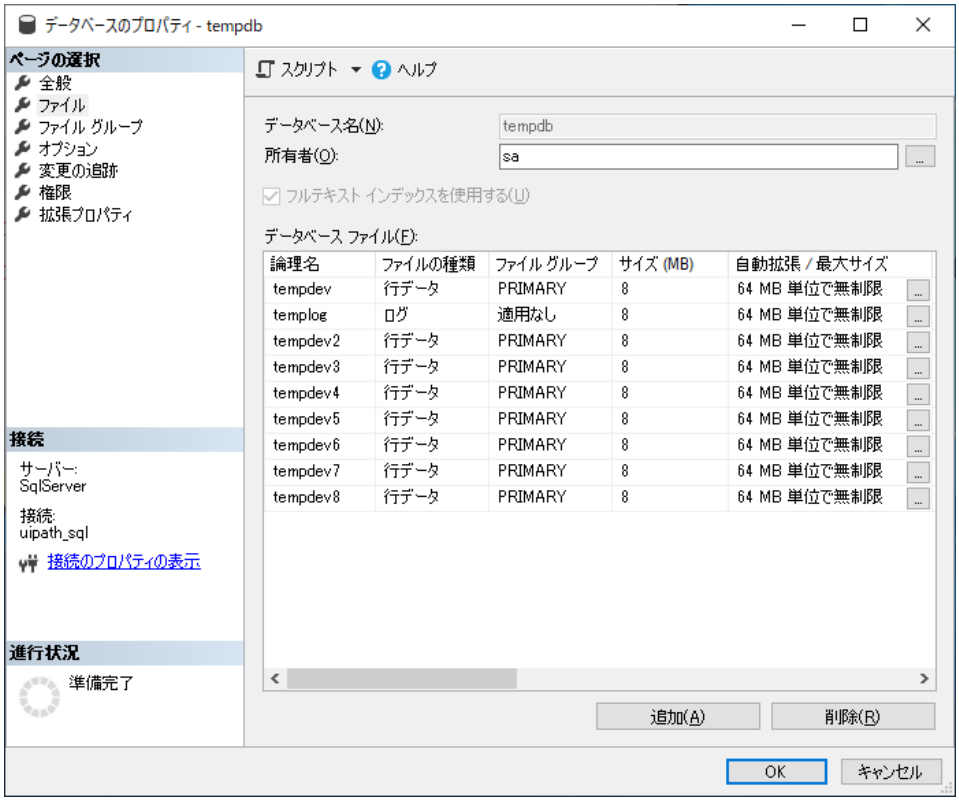
B) 論理 CPU 数 ≧ 8
一先ず 8 つで分割します。ラッチ競合の問題が検出された場合は、4 つずつファイルの追加を検討します。

理由

論理 CPU 数を基にファイル分割数を調整する理由は、論理 CPU 数がラッチ獲得の最大多重度となるためです。ファイルを過剰に分割しても意味を成さないばかりか、1 つのファイルへの I/O サイズが小さくなるため、I/O 効率の観点では非推奨です。故に分割上限の目安として、一先ず「8」としています。

方法

下図の例では [tempdev] データファイルを 8 分割しています。



備考

UiPath Orchestrator では、ユーザーデータベースについてオブジェクトの新規作成／削除を頻発させることは多くないと思われます。しかし、システム内部で利用される tempdb についてはこれが頻発する傾向にあるため、特に tempdb についてはラッチ競合緩和の観点でファイル分割が推奨されます。

参考

[23] DO's&DONT's #17: やっておいた方がいいこと – tempdb データファイル数を CPU 数に一致させる

4.3.9. エクステントの設定

SQL Server のデータは、「ページ」という 8KB の単位で管理され、ページの中には 1 種類のオブジェクトだけを格納します。また、8 ページ分 (64KB) を一纏まりとした「**エクステント**」という単位でも管理しており、同じオブジェクトのページだけで構成されたエクステントを「**単一エクステント**」、オブジェクトが異なる複数のページで構成されたエクステントを「**混合エクステント**」と呼ばれます。

優先度：任意

設定検討タイミング：運用後

推奨

混合エクステントを無効化して単一エクステントを利用することで、ラッチ競合を緩和させます。

理由

新しいオブジェクトを作成した場合、どの程度ページを使用するかは不定なため、まず空きのある混合エクステントを探し出してページを割り当てるが、状況によってはこの探索処理でボトルネックが発生する場合があります。推奨設定を適用することで、混合エクステントからではなく、可能な限り単一エクステントから割り当てようとするため、ボトルネックが発生しにくくなります。

方法

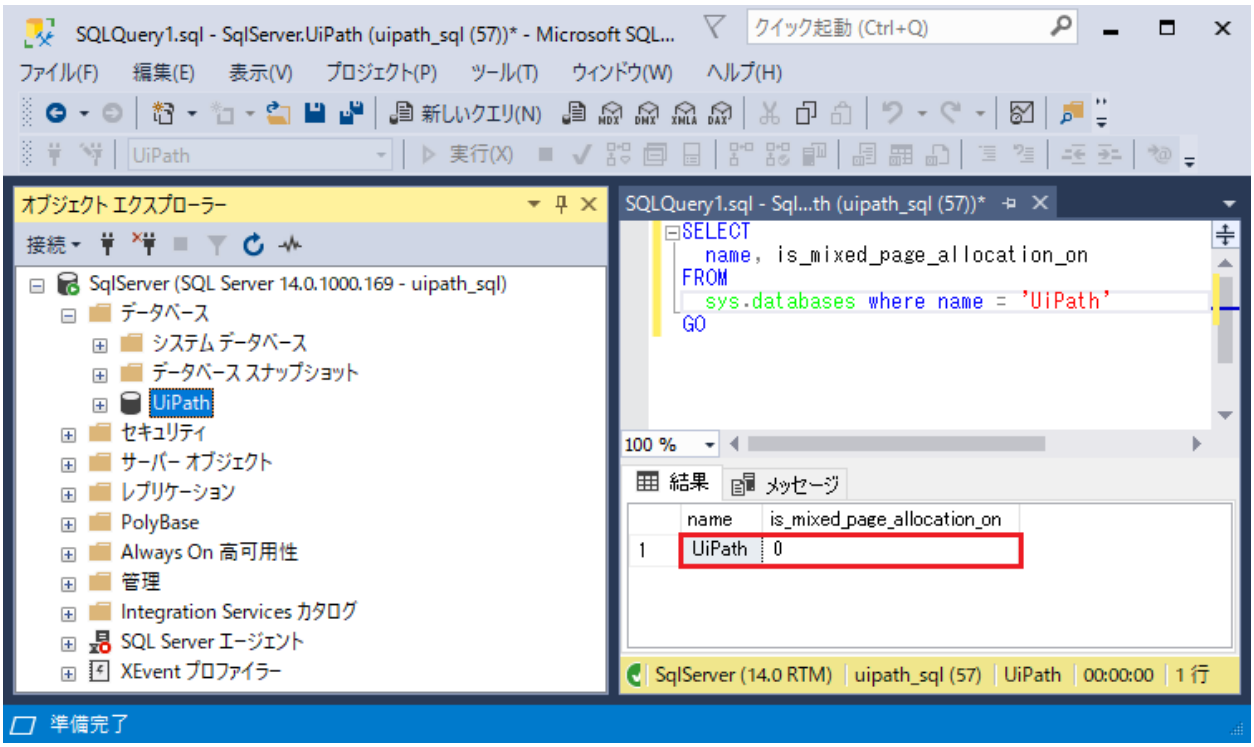
<SQL Server 2016 以降の設定方法>
既定で混合エクステントの利用は無効化されていますので、特別な設定は不要です。

```
-- SQL 文
ALTER DATABASE <dbname> SET MIXED_PAGE_ALLOCATION OFF
```

<SQL Server 2014 以前の設定方法>
トレースフラグ 1118 を有効化します。SQL Server 構成マネージャを開き、サービスの[プロパティ]ウィザードの[起動時パラメーター]タブから設定します。[起動時のパラメーターの指定(S)] 欄に次の文字列「-T1118」を追加します（§4.3.8 を参照）。

<エクステントの設定確認>
次のクエリ例を実行することで、現在の [UiPath] データーベースのエクステントの設定を確認することができます。

```
-- SQL 文
SELECT
  name, is_mixed_page_allocation_on
FROM
  sys.databases where name = 'UiPath'
GO
```



備考

<混合エクステント利用の問題点>

例えば、一時データファイルが作成／削除されると tempdb 内の混合エクステントから単一ページの割当要求／割当解除要求が発生します。この要求は管理情報（GAM、SGAM、PFS、IAM）にアクセスする必要があり、この要求の多重度が上がることでラッチ競合が発生します。ただし、混合エクステントの利用を無効化することで領域使用効率がわずかに低下する可能性がある点は留意する必要があります。

参考

[24] Pages and Extents Architecture Guide

4.4. SQL Server のメンテナンス

4.4.1. データベースの定常メンテナンス

インデックス (index) はデータベースのデータの検索を高速に行うための索引です。これにより、データベース内の全てを検索することなく目的のデータを特定することができます。SQL Server データベースエンジンでは、基になるデータに対して挿入 (INSERT)、更新 (UPDATE)、削除 (DELETE) のデータ更新操作の度にインデックスが自動的に変更されます。長期的な変更により、インデックス内の情報がデータベース内に散在（断片化）します。インデックスにキー値に基づく論理順序とデータファイル内の物理順序が一致しないページが存在すると断片化が発生します。インデックスが大量に断片化されると、クエリのパフォーマンスが低下し、Orchestrator の Web ページの応答が遅くなる場合があります。特にスキャン操作が遅くなります。

インデックスの再構成 (Reorganize) または再構築 (Rebuild) を行うと、断片化が解消されるため、定期的な再構成／再構築が推奨されますが、やみくもにデータベース内の全てのインデックスを再構成／再構築すると、データベースが大きくなれば実行に膨大な時間を要したり、ログファイルが大きくなり過ぎてしまう恐れがあります。

優先度：必須

検討タイミング：運用後

推奨

動的管理システム関数 **sys.dm_db_index_physical_stats** を用いて、その出力項目の一つである **avg_fragmentation_in_percent 値**から論理的な断片化 (インデックス内で順序が乱れたページ) の割合を確認できます。この値はインデックスを再構成または再構築するべきかどうかの判断指標となります。

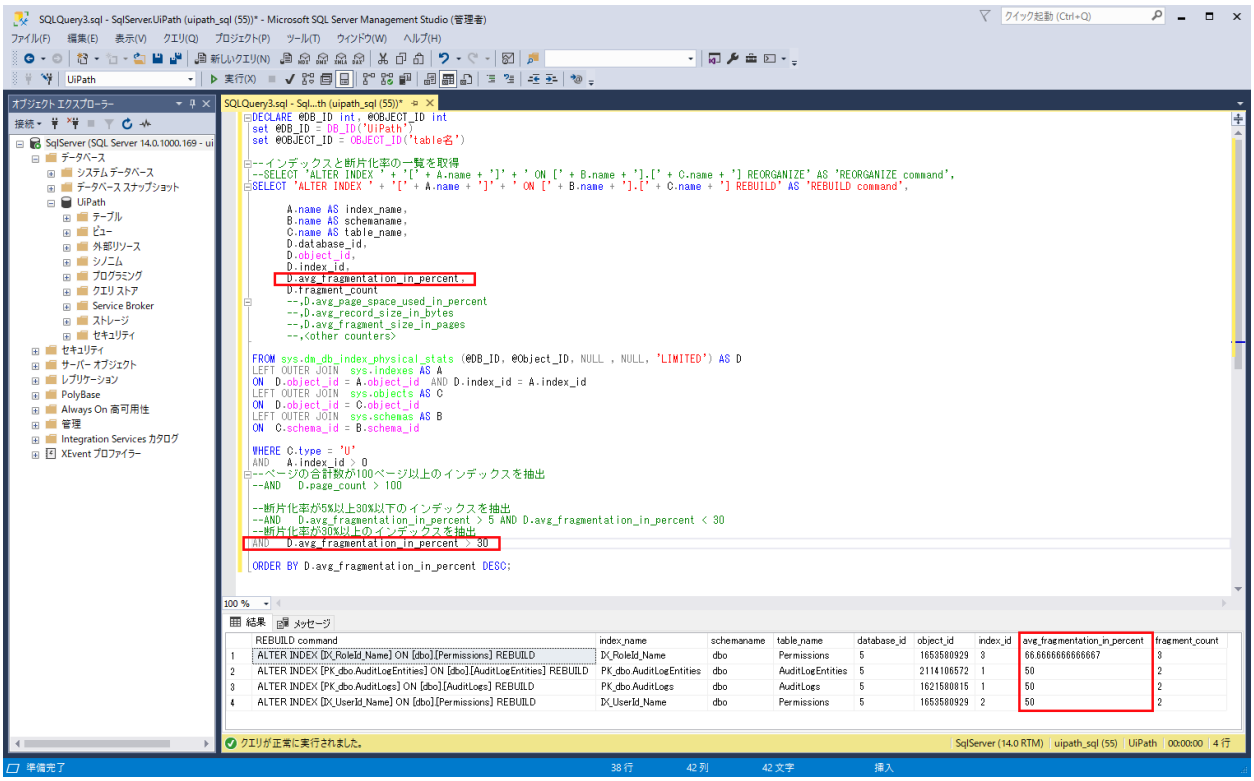
avg_fragmentation_in_percent 値	断片化解消ステートメント（SQL）
5 ～ 30%	-- SQL 文 USE <DATABASE NAME>; GO ALTER INDEX <INDEX NAME> ON <TABLE NAME> REORGANIZE; GO 断片化の度合いが酷い場合、インデックスの再構築（REBUILD）よりも実行時間がかかる場合があります。
30%以上	-- SQL 文 USE <DATABASE NAME>; GO ALTER INDEX <INDEX NAME> ON <TABLE NAME> REBUILD (WITH ONLINE = ON); GO オンライン再構築は、Enterprise Edition でのみ選択可能です。

方法

sys.dm_db_index_physical_stats へのアクセスは一般に負荷が高いため、夜間や非営業日等に実行します。

<インデックスの断片化をチェックし、断片化解消ステートメントを生成するクエリの例>

```
-- SQL 文
DECLARE @DB_ID int, @OBJECT_ID int
set @DB_ID = DB_ID('UiPath')
set @OBJECT_ID = OBJECT_ID('table 名')

--インデックスと断片化率の一覧を取得
--SELECT 'ALTER INDEX ' + '[' + A.name + ']' + ' ON [' + B.name + '].[' + C.name + '] REORGANIZE' AS
'REORGANIZE command',
SELECT 'ALTER INDEX ' + '[' + A.name + ']' + ' ON [' + B.name + '].[' + C.name + '] REBUILD' AS 'REBUILD
command',

A.name AS index_name,
B.name AS schemaname,
C.name AS table_name,
D.database_id,
D.object_id,
D.index_id,
D.avg_fragmentation_in_percent,
D.fragment_count
--,D.avg_page_space_used_in_percent
--,D.avg_record_size_in_bytes
--,D.avg_fragment_size_in_pages
--,<other counters>

FROM sys.dm_db_index_physical_stats (@DB_ID, @Object_ID, NULL , NULL, 'LIMITED') AS D
LEFT OUTER JOIN  sys.indexes AS A
ON  D.object_id = A.object_id  AND D.index_id = A.index_id
LEFT OUTER JOIN  sys.objects AS C
ON  D.object_id = C.object_id
LEFT OUTER JOIN  sys.schemas AS B
ON  C.schema_id = B.schema_id

WHERE C.type = 'U'
AND   A.index_id > 0
--ページの合計数が 100 ページ以上のインデックスを抽出
--AND   D.page_count > 100

--断片化率が 5%以上 30%以下のインデックスを抽出
--AND   D.avg_fragmentation_in_percent > 5 AND D.avg_fragmentation_in_percent < 30
--断片化率が 30%以上のインデックスを抽出
AND   D.avg_fragmentation_in_percent > 30

ORDER BY D.avg_fragmentation_in_percent DESC;
```



備考

<sys.dm_db_index_physical_stats 関数の構文>

```
-- SQL 文
SELECT * FROM sys.dm_db_index_physical_stats (DB_ID, Table_ID, index_ID, partition_No, 'ScanMode')
```

<sys.dm_db_index_physical_stats 関数の主な出力項目例>

出力項目	説明	Limited モード
avg_fragmentation_in_percent	断片化の割合（%）	✔
fragment_count	インデックス内の断片化 (物理的に連続したリーフ ページ) の数	✔
avg_page_space_used_in_percent	ページ内データ占有率（%）	(Null)
avg_record_size_in_bytes	平均レコードサイズ（byte）	(Null)
avg_fragment_size_in_pages	インデックス内の 1 つの断片化内の平均ページ数	(Null)

sys.dm_db_index_physical_stats 関数は特定のインデックス、テーブルやインデックス付きビュー上のすべてのインデックス、データベース内のすべてのインデックス、またはすべてのデータベース内のすべてのインデックスの断片化を検出できます。再構築にはオンライン方式もありますが、オンライン再構築は SQL Server の Enterprise Edition でしか使用できません。

<スキャンモード（ScanMode オプション）>

dm_db_index_physical_stats 関数の第 5 引数で指定するスキャンモードは、主に次の 3 種類です。

モード	速度	説明
LIMITED	最速	断片化の割合のみを判別する。基本的には[LIMITED]モードで十分です。
SAMPLED	高速	インデックスまたはヒープの全ページの 1% のサンプルに基づく統計情報を取得する為、この結果は近似と見なす必要があります。インデックスまたはヒープのページが 10,000 ページに満たない場合、SAMPLED モードの代わりに DETAILED モードが使用されます。
DETAILED	低速	すべてのデータ（リーフ）を対象にすべての情報を取得します。

参考

[25] SQL query performance might decrease when the SQL Server Database instance has high index fragmentation
[26] sys.dm_db_index_physical_stats (TRANSACT-SQL) > Scanning Modes

4.4.2. 統計の更新設定

アプリケーションの応答の遅延やバッチ処理／クエリのパフォーマンスが低下した場合に確認すべき設定項目です。

優先度：必須

検討タイミング：運用後

推奨

統計情報を最新の状態に保持し、現在のデータ分布に最適な実行プランが選択されるようにします。同時に、統計情報の自動更新の閾値をテーブルデータ件数により変動させます。

方法

A) 統計情報の自動作成を{ON（小規模環境、開発環境） | OFF（大規模環境）} に設定します。

```
-- SQL 文
ALTER DATABASE <DATABASE>
SET AUTO_CREATE_STATISTICS {ON | OFF} (INCREMENTAL=ON);
```

B) 統計情報の自動更新を ON に設定します。

```
-- SQL 文
ALTER DATABASE <DATABASE NAME>
SET AUTO_UPDATE_STATISTICS {ON | OFF};
```

C) 統計情報の非同期自動更新を ON に設定します。

```
-- SQL 文
ALTER DATABASE <DATABASE NAME>
SET AUTO_UPDATE_STATISTICS_ASYNC {ON | OFF};
```

D) データベース単位で統計情報を手動更新します。あらゆるクエリが遅く、どのオブジェクトの統計情報を更新すべきが不明点な場合に実施します。

```
-- SQL 文
USE <DATABASE NAME>;
GO
EXEC sp_updatestats;
```

E) テーブル／インデックス単位で統計情報を手動更新します。特定のクエリが遅い場合に実施します。

```
-- SQL 文
USE <DATABASE NAME>;
GO
UPDATE STATISTICS <TABLE NAME or INDEXED VIEW NAME> <INDEX NAME or STATISTICS
NAME> (WITH FULLSCAN);
GO
```

※ WITH FULLSCAN オプション
すべての行をスキャンして統計を計算することにより、高品質のクエリプランを作成できる場合があります。

F) トレースフラグ 2371 の有効化
SQL Server 構成マネージャを開き、サービスの[プロパティ]ウィザードの[起動時パラメータ]タブから設定します。
起動時パラメータに次の文字列「-T2371」を追加します。

備考

SQL Server の統計情報について、AUTO_CREATE_STATISTICS と AUTO_UPDATE_STATISTICS は **既定で ON** になっていますが、自動更新に関しては SQL Server 2008 R2 SP1 で追加された**トレースフラグ 2371** を使用しない場合、テーブルの母数の **20% 程度**が更新されない限り自動更新が行われないため、データが増加すると実データと統計情報に乖離が発生し、想定していたインデックスが使われない場合があります。したがって、**データが多い**（規模の大きい）**テーブルに関しては統計情報を手動更新する**べき場合があります。

SQL Server 2016 では巨大テーブルに対しても、より頻繁に統計情報が更新されるようになりました。SQL Server 2014 以前は、統計情報は変更行数の閾値 20%をトリガーとして自動更新されていました。SQL Server 2016 (互換性レベル 130) からは、この閾値はテーブル内の行数と連動します。テーブル内の行数が増加すると、統計情報変更のトリガーとなる閾値は下がります。この振る舞いは、トレースフラグ 2371 より利用できます。例えば、テーブルの行数が 10 億行の時、SQL Server 2016 以前の振る舞いでは、統計情報の自動更新が走るには 2 億行に達しなければなりませんでした。SQL Server2016 は、100 万行の更新で統計情報の自動更新が走ります。

統計の更新は同期更新 (既定) と非同期更新に大別されます。

- 同期更新
  クエリは常に最新の統計を使用してコンパイルおよび実行されます。統計が古い場合、クエリ最適化（クエリオプティマイザー）では、統計が更新されるまでクエリのコンパイルおよび実行を待機します。
- 非同期更新
  クエリは、既存の統計が古い場合でもその統計を使用してコンパイルされます。クエリのコンパイル時に古い統計が使用された場合、クエリ最適化で最適なクエリ プランが選択されない可能性があります。非同期更新の完

了後にコンパイルされるクエリには、更新された統計を使用できます。

参考

[27] Update Statistics
[28] どうする？ SQL Server のクエリ パフォーマンスが低下した！

4.4.3. データベース内の整合性の確認

データベースの破損が生じると、安定稼働に影響を与えます。破損が発生したタイミングではなく、その後アクセスしたタイミングでエラーに見舞われることになります。また、バックアップから復旧が必要な場合、どのバックアップが安全であるか確認が必要です。SQL Server では、**DBCC CHECKDB** コマンドを使用することで、指定されたデータベース内のすべてのデータベース オブジェクトの割り当てと構造的整合性をチェック、修復できます。

優先度：必須

検討タイミング：運用後

推奨

- 定期的にデータベース内の整合性、内部一貫性を確認する。DBCC CHECKDB コマンドは CPU とディスクに大きな負荷がかかるため、サーバーの負荷が軽い時間帯（夜間、非営業日）に実施します。
- バックアップが問題ないことを保証するために、完全バックアップのタイミングと併せて実行します。

方法

下記の DBCC コマンド群を実行して、データベースに物理的／論理的破損が無いか確認します。

- DBCC CHECKDB
- DBCC CHECKFILEGROUP
- DBCC CHECKTABLE

オンライン処理の性能に影響を与えないためにも、下記の事項に注意します。

- システムの使用率が低いときに実行します。
- 他のディスク I/O 操作を実行していないことを確認します。
- tempdb を別のディスク システム、または高速なディスク サブシステムに配置します。
- tempdb が拡張できるように、ドライブに十分な空き領域を用意します。
- オプションの利用
  - ESTIMATE ONLY オプション
    tempdb に必要な空き領域のサイズを見積もります。
  - NO_INFOMSGS オプション
    処理量と tempdb の使用率を減らします。

- PHYSICAL_ONLY オプション
  チェック内容を DB の物理的一貫性のみの低オーバー ヘッド チェックに変更します。

例えば DBCC コマンドは次のように実行します。

```
-- SQL 文
DBCC CHECKDB <DATABASE NAME> WITH NO_INFOMSGS,
ALL_ERRORMSGS;
```

備考

<エラー824>

読み取りまたはデータベース ページの書き込みの後、論理的な整合性チェックが失敗した場合、SQL Server エラー ログまたは Windows アプリケーション イベント ログに**エラー824** が記録されることがある。このエラーは「SQL Server のデータが破損している」または「データに不整合がある」場合にみられます。SQL Server は 8KB の「ページ」という単位でデータを保存していますが、このページ内のチェックサムと実際のデータに不整合があるとエラー824 が出力されます。

<トレースフラグ 2528>

既定動作では、DBCC CHECKDB が必要に応じて自動的に並列実行されるが、並列実行された場合、想定以上にスレッドや CPU リソースを消費する可能性があります。**トレースフラグ 2528** は DBCC CHECKDB における処理をシングルスレッドで動くように動作変更します。サーバオプション **MAXDOP = 1** にすることでシングルスレッド動作に変更可能だが、DBCC CHECKDB 以外のユーザー処理も全てシングルスレッドの動作になってしまいます。また、DBCC CHECKDB のサブセットでもある DBCC CHECKFILEGROUP、DBCC CHECKTABLE でも同様にシングルスレッド動作に変わります。

SQL Server 2014 SP2 以降では DBCC CHECKDB の実行オプションに MAXDOP が実装されたため、トレースフラグ 2528 を使わなくても制御可能となった。

<トレースフラグ 2549>

DBCC CHECKDB はデータベースのファイルが複数ドライブに分かれていた場合、ドライブ毎に纏めて処理します。通常、異なるドライブにあるか否かを物理ファイルパスのドライブレターから判断しますが、**トレースフラグ 2549** を有効にすることで、各ファイルが全て異なるドライブに配置されていると決め打ちできます。したがって、各ドライブレター配下にマウントポイントを使って実際には複数ドライブが配置されているような場合に効果が期待されます。

<トレースフラグ 2562>

DBCC CHECKDB コマンド実行時には一時領域として TEMPDB が使われます。コマンド実行中の他処理へのパフォーマンスを考慮し、既定動作では TEMPDB の使用領域に対して格納されているインデックス数などに基づいて内部的に制限が掛けられています。**トレースフラグ 2562** は DBCC CHECKDB における TEMPDB の利用制限を解除することで、コマンドの性能向上を期待できます。一方で、TEMPDB の構成や配置されているドライブの性

能などに依存するため、必ずしも性能が向上する訳ではありません。トレースフラグ 2562 を有効にすることで TEMPDB の使用量が少なくとも 5%増加するため、TEMPDB のサイズ管理や同時に流れている処理へのパフォーマンスの影響などに注意しなければなりません。

参考

[29] Check Database Integrity Task (Maintenance Plan)
[30] DBCC CHECKDB コマンドの機能強化、可能性がありますパフォーマンスを向上させる PHYSICAL_ONLY オプションを使用する場合

4.4.4. SQL Server のバックアップ

バックアップの運用では次の点に留意します。

- 復旧に必要なバックアップを正しく理解します。
- 復旧手順書を作成します。
- 復旧のリハーサルを定期的に行います。

優先度：必須

検討タイミング：運用後

推奨

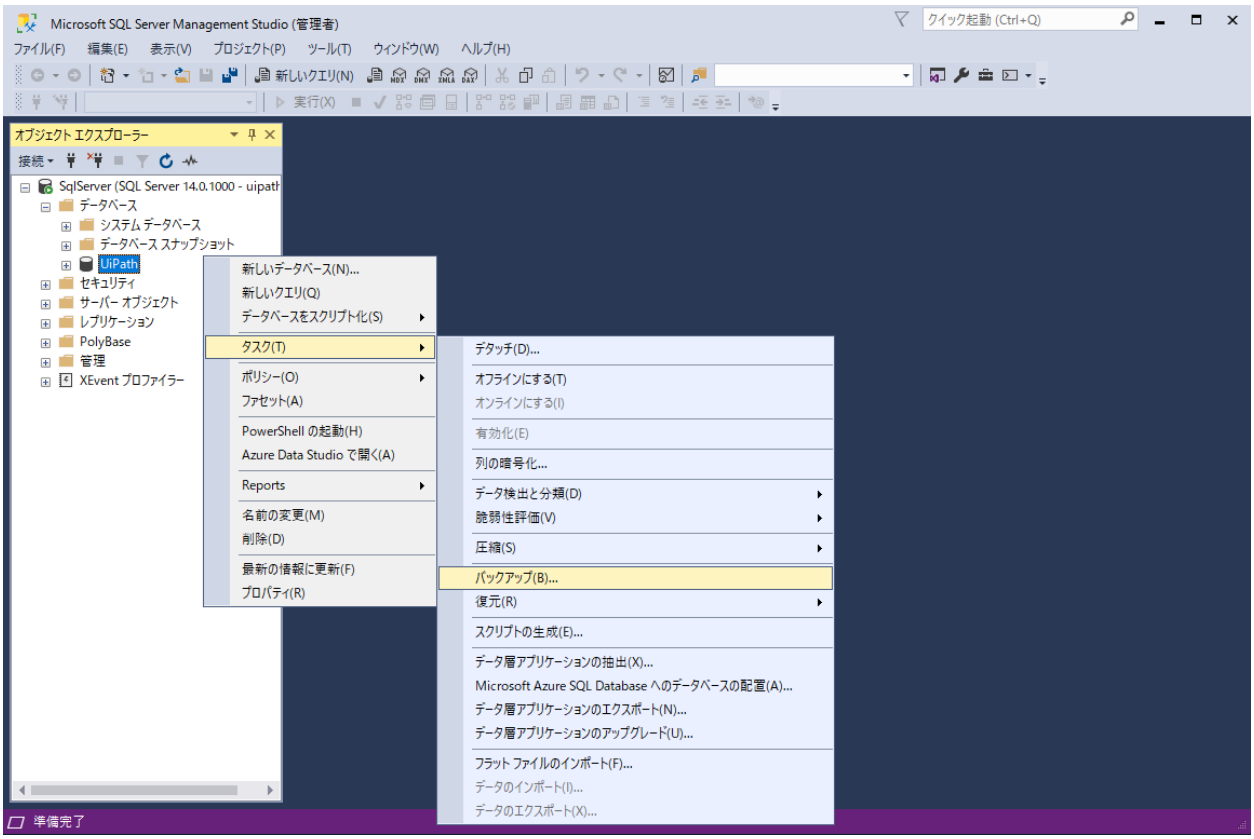
定期的にバックアップを取得します。サーバーの負荷が軽い時間帯（深夜帯、非営業日）に実施することが推奨されます。

<バックアップの基本処置>

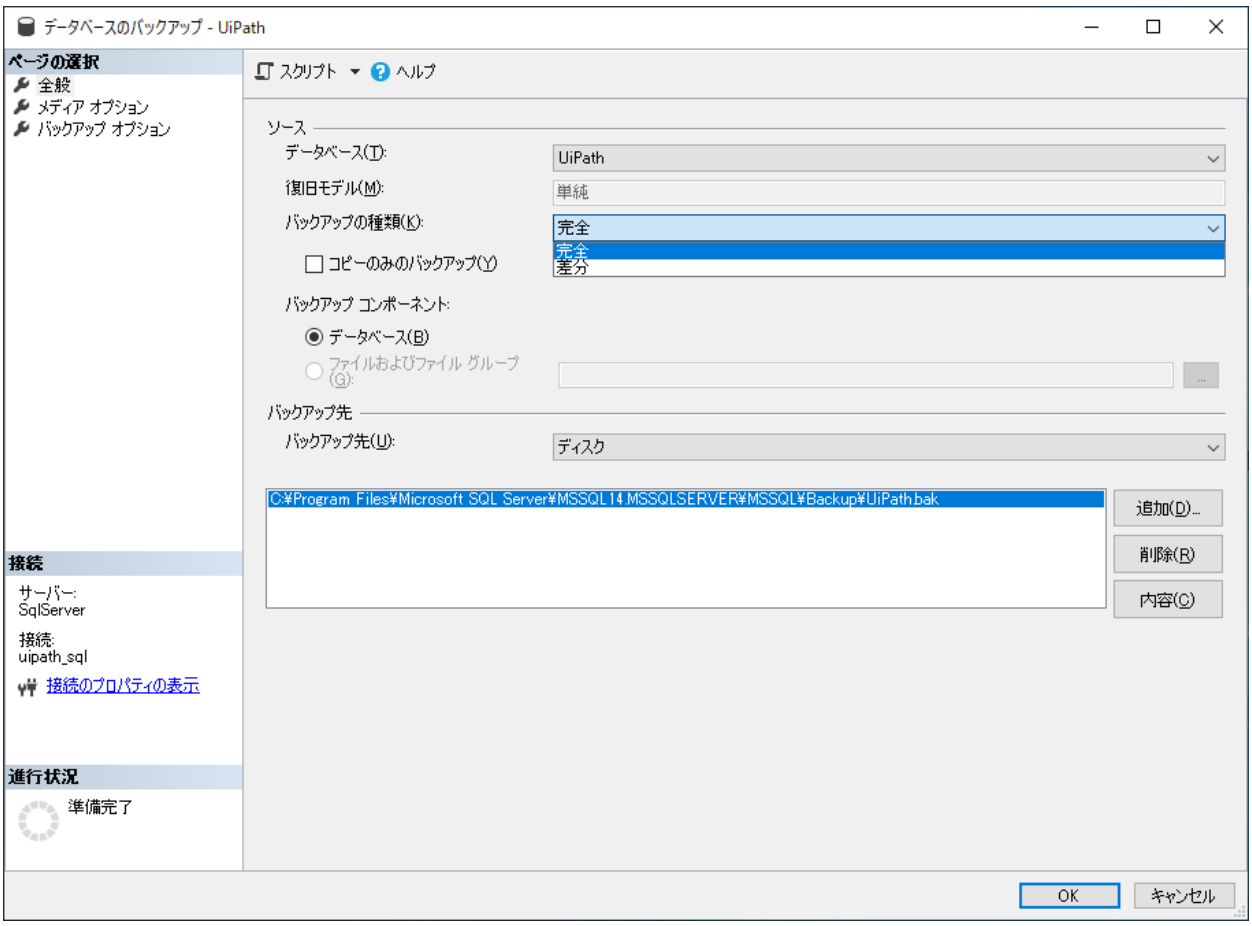
- ユーザーデータベースを週 1 で完全バックアップを取得します。
- ユーザーデータベースを完全バックアップ以外の曜日で差分バックアップを取得します。
- システムデータベースの完全バックアップを毎日取得します。
- 一日に定期的な間隔でトランザクションログをバックアップします。
- 過去のバックアップファイル (bak/trn) を削除します。
- 過去のメンテナンスプラン テキストレポート (txt) を削除します。
- 過去のバックアップログ (msdb のレコード) を削除します。

方法

1) バックアップを取得する際は 3.4.3 に従い、データベースが破損していないことを前以て確認することが推奨されます。破損しているデータベースのバックアップもまた破損しているため、正常な復旧ができません。
2) SSMS より、バックアップするデータベースを右クリックして、[タスク(T)] > [バックアップ(B)...] を選択します。



3) [データベースのバックアップ]ウィザード > [全般]ページからバックアップの種類（既定は完全バックアップ）を選択する。必要に応じてバックアップ先を設定します。



備考

３つの復旧モデルの概要を示します。

A) 単純復旧モデル：
特定の時点への復旧が必要ない場合に使用します。可用性グループやログ配布では使用できません。処理性能に優れる一括コピーではありますが、「チェック ポイント」が発生するたびに「トランザクション ログ」が切り捨てられます。このため、必要なスペースを抑制できますが、最新の「完全バックアップ」または「差分バックアップ」の時点にしか復旧できません。

B) 一括復旧モデル：
一括コピー操作のパフォーマンスに優れています。バルク挿入などの一括コピー操作を実行するときに一時的に使用することが想定されます。このモデルは、特定の大規模な操作を除いた全てのトランザクションが「トランザクション ログ」に記録されるため、ほぼ完全な復旧が可能です。特定の大規模操作の際に、「トランザクション ログ」には、エクステントのビットだけが記録されます。このため高いパフォーマンスを実現し、「トランザクション ログ」

のスペースを抑制できます。大規模操作を実行した後は、「トランザクション ログ バックアップ」を利用したデータの復旧ができなくなるが、大規模操作後、直ちに「トランザクション ログ」をバックアップすれば、その時点までの復旧が可能になる。

C) 完全復旧モデル：
ログの末尾が損傷している場合を除いて、データが損失しません。データが最大限に保護され、「トランザクション ログ」からデータを完全に復旧することができます。この「トランザクション ログ」には、全ての操作が記録されるため、大規模な操作の場合は、パフォーマンスが問題となる。「トランザクション ログ」を保持するための潤沢なログ領域が必要になる。復元の手順には、①「完全バックアップ」のリストア、②「差分バックアップ」のリストア、③「トランザクション ログ バックアップ」のリストアを実施します。

次に各復旧モデルの比較表を示します。

復旧ﾓﾃﾞﾙ	ﾊﾟﾌｫｰﾏﾝｽ	ﾃﾞｰﾀ消失の影響度	運用手順の難易度	必要なﾛｸﾞ領域の容量	特定の時点に復旧
単純	高	大	易しい	小	不可
一括	中	小	難しい	中	不可
完全	低	極小	普通	大	可

<完全復旧モデルのバックアップ運用例>

参考

[31] Recovery Models (SQL Server)

4.4.5. データベースの空き容量の管理

トランザクション ログファイル (.ldf) は、SQL Server が稼働中に実施するデータベース毎に発生したトランザクションと、加えられた変更が全て記録されるログ ファイルです。ロールバック処理や障害発生時にバックアップから復旧を行う際に利用され、データの一貫性を保つ役割を担います。完全復旧モデルや一括ログ復旧モデルならば、トランザクション ログが必須ですが、単純復旧モデルもトランザクションログに履歴を記録します。単純復旧モデルでは、トランザクションログのバックアップは任意だが、トランザクション ログ自体は必要です。

トランザクション ログファイルには全ての更新履歴が書き込まれます。既定ではログの書き込み容量上限が無いため、**放置するとサイズは増え続けます**。

優先度：必須

検討タイミング：運用後

推奨

トランザクション ログを**定期的にバックアップする**ことで領域のクリアをしておく運用が必要です。より正確には、消す（クリアする）のではなく、クリアされたバックアップ部分へ新規ログを書き込ませることで、容量的には増えないようにします。

方法

データベースに単純復旧モデルが使用されている場合は、トランザクション ログの削除（切り捨て）が自動的に実施されます。それ以外の場合は、例えば下記のクエリを実行してトランザクション ログを削除します。

1) 次のクエリを実行して、トランザクションログのサイズと使用率を確認します。

```
-- SQL 文
DBCC SQLPERF('LOGSPACE')
```

2) 次のクエリの実行結果の name フィールド値（トランザクションログの論理名）を確認します。

```
-- SQL 文
SELECT * FROM sys.database_files
```

3) 次のクエリを実行して、NULL デバイスへのバックアップを取得します。

```
-- SQL 文
BACKUP DATABASE [db_name] TO DISK = N'NUL'
BACKUP LOG [db_name] TO DISK = N'NUL'
```

「NULL」としてしまうと既定のバックアップディレクトリに「NULL」というファイル名でバックアップが取得されてしまいます。NULL デバイスという出力先へ取得されるため実態は無いが、バックアップとしては取得されています。

単純復旧モデルの場合はエラーとなる。BACKUP LOG ステートメントは、復旧モデルが「単純」の場合は許可されないため、BACKUP DATABASE または ALTER DATABASE を使用して復旧モデルを変更します。また、データベースの完全バックアップを取得していない場合、エラーメッセージとして「現在のデータベースのバックアップが存在しないために、ログのバックアップを実行できません。」が発生し、ログのバックアップを取得できません。

4) 次のクエリを実行して、取得した論理名 (db_name_Log) とファイルサイズ[MB]を指定することでトランザクション ログファイルを（この例では 1024MB に）圧縮し、ログを削減します。

```
-- SQL 文
USE [db_name]
GO
DBCC SHRINKFILE (N'db_name_Log', 1024)
GO
```

備考

データファイルは既定では

%programfiles%\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA 配下

にあり、名前は下記の通りです。

- UiPath.mdf
- UiPath_log.ldf

データベースの空き容量が逼迫しているからと云って、Logs テーブルの安易な削除はトランザクション ログの大量生成やインデックスの断片化に寄与します。また、トランザクション ログの削除によるパフォーマンスへの影響は少ないが、データベースの圧縮はパフォーマンスへの影響が大きくなります。データベース内のデータサイズを考慮せずに圧縮し過ぎると、データが増えた際に自動拡張が走り、パフォーマンスに影響を与えます。

参考

[32] SQL Server を実行しているコンピュータでトランザクション ログのサイズが予期せず増大する、または、ログがいっぱいになる
[33] Back Up a Transaction Log (SQL Server)

4.4.6. メンテナンスプランの利用

SQL Server メンテナンス プラン ウィザード を使用すると、Microsoft SQL Server エージェントが定期的に実行するメンテナンス プランを作成できます。次のようなデータベース管理タスクを定期的に実行できます。

- インデックス メンテナンスの実行（§4.4.1 を参照）
- データベース統計の更新（§4.4.2 を参照）
- データベースの整合性のチェック（§4.4.3 を参照）
- データベースのバックアップの実行（§4.4.4、§4.4.5 を参照）

このウィザードにより、SSMS で編集できるメンテナンス プランが作成されます。メンテナンス プランを編集して新しいタスクを追加したり、タスク間でワークフローを定義します。

優先度：**任意**

検討タイミング：**運用後**

推奨

設定内容については§3.4.1~§3.4.5 に準拠する。

方法

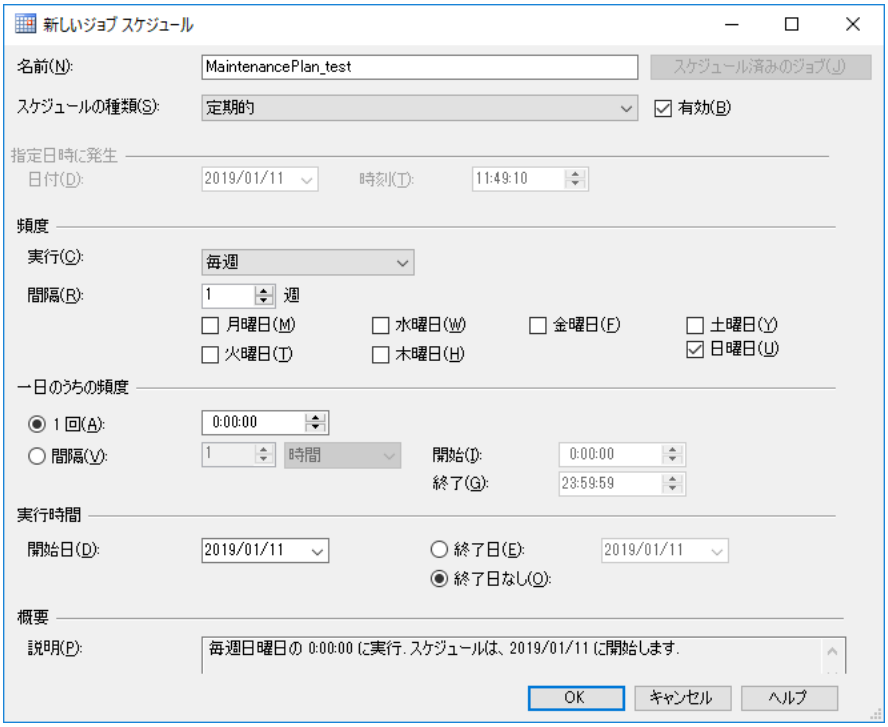
1) ウイザードの起動

i) SSMS を起動します。

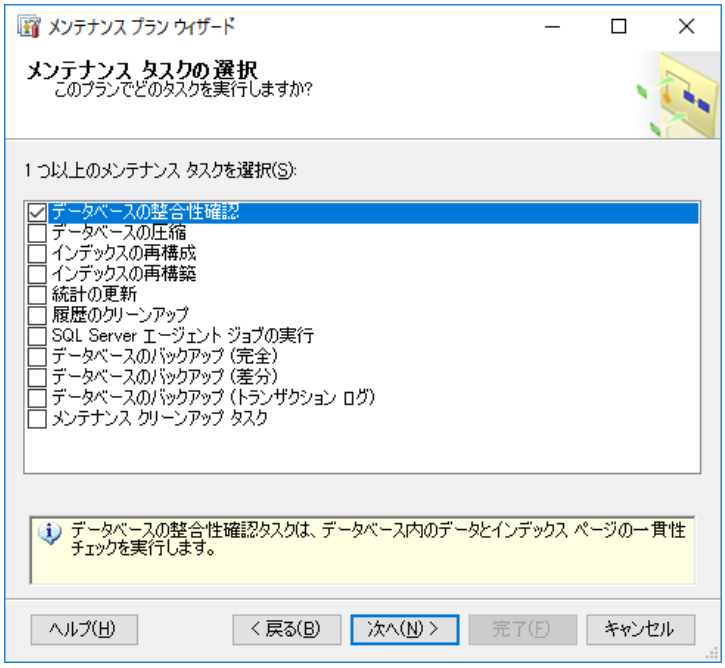
ii) メンテナンスを実行するサーバー配下の [管理] フォルダを展開します。

iii) [メンテナンス プラン] フォルダを右クリックし、[メンテナンス プラン ウィザード(W)] をクリックします。

2) ウィザード画面から、各々のメンテナンスプラン毎に詳細なスケジューリング設定ができます。



3) 複数のメンテナンス タスクから適切な項目を選択します。

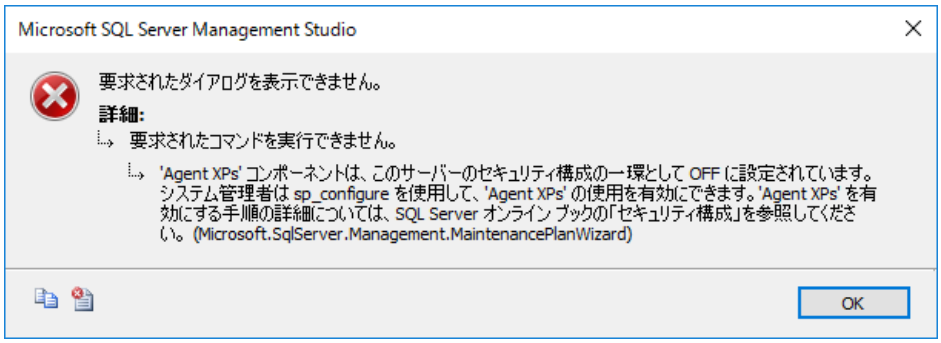


※ **データベースの圧縮** (英: Shrink Database) は空き領域にデータを無理やり詰めてインデックスの断片化を進める要因となり、パフォーマンス劣化の恐れがあるため**非推奨**です。ログが入っているデータテーブルならばページ圧縮します。一般に行圧縮は効果が薄いとされています。

備考

- Agent XPs サーバー構成オプションの有効化

メンテナンス プラン ウイザードの起動には **Agent XPs サーバー構成オプション** を有効 (Agent XPs = 1) にする必要があります。既定では無効 (Agent XPs = 0) となっているため、メンテナンス プランを参照しようとすると下記のようなエラーメッセージが表示されます。



Agent XPs オプションは、このサーバーで SQL Server エージェントの**拡張ストアド プロシージャ**を有効にする場合に使用します。このオプションを有効にしない限り、SQL Server のオブジェクト エクスプローラに [SQL Server Management Studio] エージェントのノードが表示されない。

<有効化手順>

1) ローカル グループ ポリシー エディター (gpedit.msc) >コンピュータの構成 > Windows の設定 > セキュリティの設定 > ローカルポリシー > ユーザー権利の割り当て > メモリ内のページロックから SQL Server を稼動させているユーザー（例えば Administrator）を設定します。

2) 以下のクエリを実行して、SQL Server エージェントの拡張ストアド プロシージャを有効にします。この設定の有効化にはサーバーの再起動を必要としません。

```
-- SQL 文
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'Agent XPs', 1;
GO
RECONFIGURE
GO
```

- UiPath Orchestrator のメンテナンス参考事例

ログが入っているデータテーブルならばページ圧縮を実行します。一般に行圧縮は効果が薄いとされています。

定期的に SQL Server に貯められたログのメンテナンス実行を仕込む必要があります。

<Resolution Summary>
Issue Description: How to delete Orchestrator logs
Resolution:

- If you want to delete Orchestrator logs you have to do it from the SQL database and delete the old data periodically.
- Please check the below link for the maintenance considerations.
  https://orchestrator.uipath.com/docs/maintenance-considerations

---

下記のスクリプトを参考に、SQL Server のジョブとして削除スクリプトを登録し、例えば毎週日曜 xx 時で定期的に自動で実行させます。

https://orchestrator.uipath.com/docs/maintenance-considerations#section-deleting-old-data-periodically

- Creating an Archive Database
- Deleting Old Data Periodically

参考

[34] Agent XPs Server Configuration Option

4.5. SQL Server の監視

システム全体の稼働状況をリアルタイムに把握することはシステムの効率的な安定稼働にとって重要です。

4.5.1. 稼働監視

稼働監視は最も基本的で容易に行える監視です。ネットワーク的に外部から監視でき、Windows OS に付随するコマンドで実現されます。

優先度：任意

検討タイミング：運用後

推奨

A) Orchestrator API を送信して、HTTP が正常処理されていること確認します。
B) ICMP 受信 (ping) できることを確認し、ポートチェックによるサービスの死活監視を実施します。

方法

A) ヘルスチェック用の Orchestrator API (GET /api/Status) を送信して、HTTP ステータスコード 200（正常処理）が受信できることを確認します。

```
GET /api/Status HTTP/1.1\r\nHost: <SQL Server のホスト名>\r\nConnection: Close \r\n
```

B) アプリケーションが使う Port に接続して応答を観ることでサーバーやネットワーク機器、アプリケーションが正常に動作しているかを監視します。
<ping コマンドの例>

```
C:\>ping www.uipath.com
接続中: server...
```

ping コマンドは、デフォルトでは 32bytes のデータ部（データ内容は単なる ASCII 文字列）を持つ ICMP echo パケットを 4 つ生成して、1 秒間隔で指定された宛先（通信相手の IP アドレスやホスト名）に対して送信します。

<telnet コマンドの例>
Web サーバとの接続テストを行うためには、次のようなコマンドを実行します。telnet に続けて、接続先のサーバ名と TCP のポート番号（標準的な HTTP プロトコルなら 80）を指定します。

```
C:\>telnet server 80
接続中: server...
```

< netstat コマンド>

このコマンドは TCP および UDP プロトコルを対象に統計情報を表示します。

< netsh diag コマンドの例>

```
C:\>netsh diag connect ieproxy

Internet Explorer Web プロキシ (192.168.0.xx)
   IEProxyPort = 8080
   IEProxy = 192.168.0.xx
   サーバは次のポートで実行中と思われます [8080]
```

Proxy サーバ（192.168.0.13 の 8080 番ポートでリッスンしている）と正常に通信されると、末行のように「サーバは次のポートで実行中と思われます [8080]」という結果が帰ってきます。Proxy サーバとの通信が失敗すると、最後の [ ] の内部は、ポート番号ではなく、[（なし）] となります。

4.5.2. リソース監視

CPU、メモリ、ディスク I/O、プロセスなど、OS やサーバーのハードウェア、ネットワーク機器等のリソース使用状況を確認することで、システムやサービスが正常に稼働しているかを監視します。

リソース監視を行うためには各サーバーやネットワーク機器にログインしてコマンドを実行して情報を収集する必要があります。ネットワーク機器や NAS 等の専用ハードウェア機器では SNMP エージェントが動作している場合が多いため、外部から SNMP コマンドを利用して情報を収集できます。また、OS の機能であるリソースモニターを利用して、例えば下記の項目を各カウンターから確認できます。

優先度：必須

検討タイミング：運用後

推奨

パフォーマンスモニターから、§2.3.1 の IIS との共通カウンターと下記のカウンターを監視します。

➢ SQL Server				
resource	**#) \object(instance) \counter**	**説明**	**正常系条件**	**対処例**
Buffer Cash (RAM)	62) \SQL Server:Buffer Manager\Free pages	バッファ プールの未使用ページリスト（Free Page List） にあるページの総数。メモリ不足の場合、レイジー ライター（実行プランのキャッシュに必要なメモリを管理する）あるいはチェック ポイントなどの処理を保持できなくなる可能性があります。	4 ページ以下	メモリ増設
Buffer Cash (RAM)	63) \SQL Server:Buffer Manager\Buffer cache hit ratio	バッファーキャッシュ内における、ディスクから読み取る必要が無いページの比率。長い時間が経過すると、この比率はほとんど変化が見られなくなります。キャッシュから読み取る方が、ディスクから読み取るよりもコストが低いので、この比率が高くなるようにします。 一般に、SQL Server が使用できるメモリの量を増やすか、バッファープール拡張機能を使用することで、バッファーキャッシュヒット率を増加させることができます。	90%以上を維持（100%に近いほど正常です。）	メモリ増設

		※ この比率は、直近の数千のページアクセスでのキャッシュヒットの総数をキャッシュ参照の総数で割って算出されます。Buffer cache hit ratio base カウンターと併せて利用される場合があります。		
Buffer Cash (RAM)	64) \SQL Server:Buffer Manager\Checkpoint pages/sec	チェックポイントにより、またはすべてのダーティページをフラッシュする必要があるその他の操作により、ディスクにフラッシュされた 1 秒あたりのページ数。 ※ ダーティバッファーは、異なるページのためにバッファーを再利用する前にディスクに書き戻す必要がある変更を含んでいるバッファーです。		
Buffer Cash (RAM)	65) \SQL Server:Buffer Manager\Lazy writes/sec	バッファーマネージャーのレイジーライターにより書き込まれたバッファーの 1 秒あたりの数。 ※ レイジーライターは、古いダーティバッファーをまとめてフラッシュし、ユーザープロセスで使用できるようにするシステムプロセスです。レイジーライターを使用することで、使用可能なバッファーを作成するために頻繁にチェックポイントを実行する必要がなくなります。		
Buffer Cash (RAM)	66) \SQL Server:Buffer Manager\Page Life Expectancy	ページが参照されないままバッファープールに存在する秒数。		
RAM (Buffer Cash)	67) \SQL Server:Buffer Manager \Page Life expectancy	バッファ プール内で、ページが参照されなくても保持される秒数。	300 秒以下	メモリ増設
CPU	68) \Process(sqlservr)\% Processor Time	該当プロセスのスレッドすべてが、命令を実行するためにプロセッサを使用した経過時間の割合(%)。		
CPU	69) \Process(sqlservr)\Working Set	プロセスで使用しているメモリサイズ（仮想メモリ含む）の内、実際のメモリ上で確保されているメモリ量。		
CPU	70) \Process(sqlservr)\IO Read Bytes/sec	プロセスが I/O 操作からバイトを読み取っている率。		
CPU	71) \Process(sqlservr)\IO Write Bytes/sec	プロセスが I/O 操作にバイトを書き込んでいる率。		
CPU	72) \Process(sqlservr)\Page Faults/sec	ページフォルトの発生回数/秒。この値はハード ページ フォルト（ディスクアクセスを伴う処理）とソフト ページ フォルト（ページ フォルトが物理メモリの他の場所に検出される）の両方を含みます。		
Disk I/O	73) \LogicalDisk(_Total)\Avg. Disk sec/Read	論理ディスクに対する平均読み取り時間。必要に応じて [PhysicalDisk] オブジェクトでも確認します。 ※ 例えば、1 本の論理ディスクが 2 本の物理ディスクで構成されている場合、LogicalDisk のカウンターから得られるデータと PhysicalDisk のカウンターから得られる値は異なります。LogicalDisk のデータを参照した場合、2 本の物理ディスクを 1 つの論理ディスクとして束ねた結果のパフォーマンス データが得られます。PhysicalDisk を参照する場合、個別の物理ディスクのパフォーマンス データが得られます。アプリケーションからは論理ディスクの単位でディスクを参照しますので、ディスクのボトルネックを調査する時には最初に LogicalDisk で論理ディスクの応答時間を測定し、後に PhysicalDisk で物理ディスクを個別に調査すると良いでしょう。	常に 20 ミリ秒未満。スパイク（最大値）は 50 ミリ秒以下。	

Disk I/O	74) \LogicalDisk(_Total)\Avg. Disk sec/Write	論理ディスクに対する平均書き込み時間。必要に応じて [PhysicalDisk] オブジェクトでも確認します。	常に 20 ミリ秒未満。スパイク（最大値）は 50 ミリ秒以下。	
Disk I/O	75) \Logical(_Total)\Avg. Disk sec/Transfer	1 回の操作中に論理ディスク間で転送されたバイト数の平均値。 ※ 必要に応じて [PhysicalDisk] オブジェクトでも確認します。0.3 秒より大きい値が確認された場合、エラーのためにディスク コントローラがディスクを再試行している可能性がある。	常に 15 ミリ秒未満。	
Disk I/O	76) \PhysicalDisk(_Total)\Avg. Disk Read Queue Length	ディスクからの読み取りを待つ物理ドライブ毎のカウンター採取間隔の平均リクエスト数。		
Disk I/O	77) \PhysicalDisk(_Total)\Avg. Disk Write Queue Length	ディスクへの書き込みを待つ物理ドライブ毎のカウンター採取間隔の平均リクエスト数。		
Disk I/O	78) \PhysicalDisk(_Total)\Current Queue Length	ディスク I/O 待ちキュー長の平均。2 よりおおきいあたいがかくにんされたばあい、大きな値はディスク I/O 処理要求で待ち発生している可能性があります。	2 未満	
Disk I/O	79) \PhysicalDisk(_Total)\% Disk Time	読み込みまたは書き込み要求の処理でディスクがビジー状態だった経過時間の割合（%）。通常，継続的に 100%に近い状態の場合，ディスクが激しく使用されています。ただし、この値が高い場合でも必ずしもディスクがボトルネックになる訳ではありません。例えば、低速なネットワークを利用している場合、ディスクから効率よくデータを読み書きできなくなるため、ディスク自体の不調という原因も考えられます。	常に 80%以下。常に数値が 80%以上の場合、メモリリークの可能性があります。	
Disk I/O	80) \PhysicalDisk(_Total)\Disk Reads/sec	読み取り操作/秒。	一般に Ultra Wide SCSI ディスクは 1 秒当たり 50~70 の I/O 処理が可能です。	
Disk I/O	81) \PhysicalDisk(_Total)\Disk Writes/sec	書き込み操作/秒		
Disk I/O	82) \SQL Server:Buffer Manager \Page Reads/Sec	SQL Server のバッファ マネージャによる、ディスク上の読み取り回数/秒。	ディスクの仕様で規定されている値の最大値未満。	• メモリ増設 • I/O 効率改善 • 非正規化 • インデックス使用方法の調整
Disk I/O	83) \SQL Server:Buffer Manager \Page Writes/Sec	SQL Server のバッファ マネージャによるディスク上の書き込み回数/秒。	ディスクの仕様で規定されている値の最大値に近い。	I/O 効率改善
Transaction	84) \SQL Server:Databases \Transactions/sec	DB で開始されたトランザクション数/秒。この数値はシステム規模の参考になります。また、トランザクション負荷の増減をチェックするのにも役立ちます。		
Transaction	85) \SQL Server:Databases(_Total) \Active Transactions	現在の DB でアクティブなトランザクション数。[SQL Server:Databases \Transactions/sec] が、[SQL Server:Databases(_Total) \Active Transactions] を超える場合、サーバーの負荷が超過状態です。		
Transaction	86) \SQL Server:General Statistics \User Connections	SQL Server に現在接続しているユーザー数。この値の大幅な変動には注意を払いましょう。		
Index	87) \SQL Server:Access Methods \Full Scans/Sec	インデックスのフルスキャン回数/秒。この値が大きくなる場合、アプリケーションがインデックスを効率的に使用していないことが懸念される。原因となっているコード（クエリ）を調べ、必要に応じてインデックスを作成します。また [tempdb] は殆どインデックス付けされないた		

		め、[tempdb] データベースから情報を返す際にこの値が高くなる場合があります。		
Index	88) \SQL Server:Access Methods \Index Searches/Sec	インデックス検索数/秒。このカウンターでシステムのデータアクセスのパターンをチェックできます。		
Index	89) \SQL Server:Access Methods \Page Splits/Sec	ページ分割回数/秒。性能上問題となるページ分割の回数をチェックできます。		
Lock	90) \SQL Server:Locks \Average Wait Time (ms)	待ち状態の原因となる各ロック要求の平均待ち時間（ミリ秒）。		
Lock	91) \SQL Server:Locks \Lock Waits/Sec	ロック取得のために、待機しなければならない要求の数/秒。		
Lock	92) \SQL Server:Locks \Lock Timeouts/sec	タイムアウトしたロック要求の数/秒。NOWAIT ロックの要求を除きます。		
Lock	93) \SQL Server:Locks \Number of Deadlocks/sec	デッドロックに至るロック要求の数/秒。		
Latch	94) \SQL Server:Latches \Average Latch Wait Time	ラッチ要求の平均待ち時間 （ミリ秒）。この数値が大きくなると、サーバーがリソースを求めて競合に巻き込まれる恐れがあります。ラッチは、負荷の軽い短期の同期化オブジェクトで、トランザクション全体にわたってロックする必要がない動作を保護します。主に、接続に対して行が読み取られている間、行を保護するために使用します。		
SQL Statistics	95) \SQL Server:SQL Statistics\Batch Requests/sec	1 秒あたりに受信した Transact-SQL コマンドのバッチの数。バッチ要求の数が多さは、スループットが優れていることの目安です。		
SQL Statistics	96) \SQL Server:SQL Statistics\SQL Compilations/Sec	SQL コンパイルの回数/秒。クエリの再コンパイルの回数も含まれる。SQL Server のユーザー利用状況が安定している場合、通常この値は安定した状態になります。この値が定常的に高い場合は調査が必要です。アドホック クエリの問題の監視に使用します。		
SQL Statistics	97) \SQL Server:SQL Statistics\SQL Re-Compilations/Sec	クエリの再コンパイルの回数/秒。この値が定常的に高い場合は調査が必要です。アドホック クエリ、結合方法の問題を監視するのに使用します。クエリの再コンパイル処理は「スキーマの変更」「テーブルに多くの行をインサートする」「テーブルから多くの行をデリートする」等の操作によって発生します。		
SQL Statistics	98) \SQL Server:SQL Statistics\Buffer cache hit ratio	バッファーキャッシュ内における、ディスクから読み取る必要が無いページの比率。長い時間が経過すると、この比率はほとんど変化が見られなくなります。キャッシュから読み取る方が、ディスクから読み取るよりもコストが低いので、この比率が高くなるようにします。 一般に、SQL Server が使用できるメモリの量を増やすか、バッファープール拡張機能を使用することで、バッファーキャッシュヒット率を増加させることができます。 ※ この比率は、直近の数千のページアクセスでのキャッシュヒットの総数をキャッシュ参照の総数で割って算出されます。 Buffer cache hit ratio base と併せて使う必要があります。		
others	99) \SQL Server:Buffer Manager\Memory Grants Pending	作業領域メモリの使用許可を待っている処理の数。アドホック クエリ、結合方法の問題を監視するのに使用します。		
Others	100) \SQL Server:Buffer Manager\Stolen Page Count	他のサーバー メモリ要求によって奪われたバッファ キャッシュのページの数。アドホック クエリ、結合方法の問題を監視するのに使用します。		

方法

§3.2.1 と同様。

参考

[35] Performance Counters (SSAS)

4.5.3. アプリケーション監視

アプリケーション監視とは、サーバー上で稼働しているアプリケーションやミドルウェアの内部ステータスやファイル、ログを監視することです。

優先度：任意

検討タイミング：運用後

備考

<SQL Server エラーログ>

```
2018-09-10 00:35:55.73 Logon       Error: 18456, Severity: 14, State: 8.
2018-09-10 00:35:55.73 Logon       Login failed for user 'kenats'. Reason: Password did not match that for the login provided. [CLIENT: <local machine>]
```

<イベントログ (アプリケーション／システム)>

```
Information,9/10/2018 12:33:11 AM,MSSQLSERVER,17811,Server,The maximum number of dedicated administrator connections for this instance is '1'
Error,9/10/2018 12:33:08 AM,Microsoft-Windows-Perflib,1008,None,"The Open Procedure for service ""BITS"" in DLL ""C:\Windows\System32\bitsperf.dll"" failed. Performance data for this service will not be available. The first four bytes (DWORD) of the Data section contains the error code."
Information,9/10/2018 12:33:07 AM,MSSQLSERVER,873,Server,Buffer pool extension is already disabled. No action is necessary.
```

4.5.4. SQL Server のボトルネックの探索

クエリの実行から完了までの間には、ブロックの獲得待ち、タスクを処理するためのスレッド割り当て待ち、タスクに必要なメモリの割り当て待ちなど、様々なリソースを獲得するための待機状態が存在します。クエリのパフォーマンスを解析、あるいはチューニングする場合、どのような種類のリソースをどれだけの期間待機したかを把握することは、クエリの実行時間の短縮を考える上でそれ非常に重要となります。

ある種のクエリは、ヒープテーブルに ORDER BY 句を指定する場合等のように、メモリ内でデータの並べ替えを行う必要があります。その際、SQL Server が使用できるメモリリソースに余裕がないと、必要なサイズのメモリを獲得できずにクエリは待機状態となります。そのような場合には SQL Server の内部で、クエリは「RESOURCE_SEMAPHORE」というリソースを待機した状態になります。

優先度：任意

検討タイミング：運用後

推奨

A) **sys.dm_os_wait_stats 動的管理ビュー**を利用して、SQL Server インスタンス全体の様々な待機状態や「どのリソースの待機が多いか」を確認します。

B) **sys.dm_os_latch_stats 動的管理ビュー**を利用して、LATCH%系の詳細情報を確認します。

方法

A) 次のクエリを実行して、SQL Server インスタンス全体で参照したい待機状態 (wait_type) の情報を確認します。

```
-- SQL 文
SELECT
    CASE
        WHEN [wait_type] LIKE N'SOS_SCHEDULER_YIELD'        THEN N'CPU'
        WHEN [wait_type] LIKE N'RESOURCE_SEMAPHORE'        THEN N'Memory'
        WHEN [wait_type] LIKE N'LCK_M%'                    THEN N'Lock'
        WHEN [wait_type] LIKE N'LATCH%'                    THEN N'SqlServer Process Memory'
        WHEN [wait_type] LIKE N'PAGELATCH%'                THEN N'SqlServer Management Page'
        WHEN [wait_type] LIKE N'PAGEIOLATCH%'              THEN N'Disc I/O'
        WHEN [wait_type] LIKE N'WRITELOG'                  THEN N'Transaction Log File'
        WHEN [wait_type] LIKE N'ASYNC_NETWORK_IO'          THEN N'Network'
    END AS [Bottleneck],
*
FROM sys.dm_os_wait_stats
WHERE       [wait_type] LIKE N'SOS_SCHEDULER_YIELD'
        OR  [wait_type] LIKE N'RESOURCE_SEMAPHORE'
        OR  [wait_type] LIKE N'LCK_M%'
        OR  [wait_type] LIKE N'LATCH%'
        OR  [wait_type] LIKE N'PAGELATCH%'
        OR  [wait_type] LIKE N'PAGEIOLATCH%'
        OR  [wait_type] LIKE N'WRITELOG'
        OR  [wait_type] LIKE N'ASYNC_NETWORK_IO'
```

%部分には「_SH（参照）」、「_UP（更新）」、「_EX（排他）」などが入ります。出力される情報は SQL Server が起動してからクエリが実行された時間までの累積値であるため、時系列として動きは見えません。ただし、SQL Server インスタンス全体の値が確認できるだけで、個々のクエリと関連付けることはできないため、特定のクエリのチューニングを目的として待機時間を正確に把握するための用途には適しません。

B) 次のクエリを実行して、SQL Server インスタンス全体で参照したいノンバッファーラッチの情報を確認します。

```
-- SQL 文
SELECT * FROM sys.dm_os_latch_stats
WHERE [waiting_requests_count] > 0
ORDER BY [wait_time_ms] DESC;
GO
```

待機状態	説明	ボトルネック
SOS_SCHEDULER_YIELD	あるタスクが他のタスクの実行のためにスケジューラを自主的に解放したときに発生する。この待機中、タスクは Quantum の更新を待機しています。 ※ Windows ではスレッドに対して、タイムスライスと呼ばれる一定の実行時間が割り当てられ、すべてのスレッドはこのタイムスライスの間だけ実行され、経過すると次のスレッドに切り替わる。Quantum とは Windows におけるタイムスライスの最小単位である。基本的にタイムスライスの割り当て量は、Quantum の整数倍になります。	CPU
RESOURCE_SEMAPHORE	他の同時実行クエリがあるため、クエリ メモリの要求がすぐに許可されない場合に発生する。待機および待機時間が高い値を示している場合、同時実行クエリの数が多すぎるか、またはメモリ要求の数が多すぎる可能性があります。	RAM
LCK_M%	タスクがロックの取得を待機しているときに発生します。 ※ ロックとは、コミット命令によってトランザクションが完了するまで保持される排他制御を掛ける仕組み。ロックはトランザクション全体を保護する際に使用されます。	ロック
PAGELATCH%	バッファーラッチ（英: Buffer Latch）。タスクが I/O 要求内に無いバッファーラッチで待機しているときに発生します。	SQL Server 管理ページ
PAGEIOLATCH%	バッファーラッチ。タスクが、I/O 要求内のバッファーラッチで待機しているときに発生します。待機時間が長い場合、ディスク サブシステムに問題がある可能性があります。 ※ バッファーラッチとは、メモリ上のデータを操作する場合などに発生するラッチです。	ディスク I/O (データファイル)
LATCH%	ノンバッファーラッチ（英: Non-Buffer Latch）。ラッチを待機しているときに発生します。バッファーラッチまたはトランザクション マーク ラッチは含みません。 ※ ラッチとは、ページデータ更新時に瞬間的な排他制御を掛ける仕組みです。ラッチはトランザクション内部の短い時間でオブジェクトを保護する際に使用されます。	SQL Server プロセスのメモリ
WRITELOG	ログ フラッシュの完了を待機しているときに発生します。ログ フラッシュの原因となる主な操作としては、チェックポイントとトランザクションのコミットがあります。	ディスク書き込み (トランザクション ログ ファイル)
ASYNC_NETWORK_IO	ネットワークの書き込みでタスクがブロックされているときに発生します。クライアントがサーバーからのデータを処理しているかどうかを確認すると良いでしょう。	ネットワーク

クエリ実行中に発生している待機時間の種類によって、クエリにボトルネックや機能停止ポイントがあるかどうかを判断できます。サーバー全体の待機時間や待機カウントが高い値を示している場合は、サーバーインスタンス内の対話型クエリの対話にボトルネックまたはホットスポットが存在していることを表しています。

備考

累積値をクリアするには次の DBCC コマンドを実行します。

```
-- SQL 文
DBCC SQLPERF ('sys.dm_os_wait_stats', CLEAR);
GO
DBCC SQLPERF ('sys.dm_os_latch_stats', CLEAR);
GO
```

4.6. SQL Server のウイルススキャン除外設定

ウイルス対策ソフトウェアを実行するには、いくつかのシステムリソースが必要となります。SQL Server を実行しているコンピュータにパフォーマンスの影響があるか否かを判断するために、ウイルス対策ソフトウェアをインストールする前後でテストを実行する必要があります。

優先度：必須

検討タイミング：運用後

推奨

除外対象	既定のディレクトリのパス	説明
SQL Server Database Primary Data File (.mdf)	%ProgramFiles%\Microsoft SQL Server\<Instance_ID>.<Instance_Name>\MSSQL およびその配下	プライマリ データ ファイルにはデータベースの起動情報（ブートページ）が含まれており、データベース内の他のファイルを指定します。各データベースには 1 つのプライマリ データ ファイルがあります。プライマリ データ ファイルにはブートページ ファイルが格納されており、これが破損するとデータベースは起動できません。
SQL Server Database Secondary Data File (.ndf)	（同上）	ユーザー データおよびオブジェクトは、プライマリ データ ファイルまたはセカンダリ データ ファイルに格納される。セカンダリ データ ファイルにはブートページが格納されていません。
SQL Server Database Transaction Log File (.ldf)	（同上）	トランザクションログは、SQL Server が稼働中に行ったデータベース毎に発生したトランザクションと、加えられた変更が全て記録される重要なログファイルです。ロールバック処理や障害発生時にバックアップからの復旧に利用され、データの一貫性を保つために利用されます。
SQL Server のバックアップファイル (.bak / .trn)	%ProgramFiles%\Microsoft SQL Server\<Instance_ID>.<Instance_Name>\MSSQL\Backup	
フルテキスト カタログ ファイル（英: Full-Text catalog files）	• 既定のインスタンス %ProgramFiles%\Microsoft SQL Server\MSSQL\FTDATA • 名前付きインスタンス %ProgramFiles%\Microsoft SQL Server\MSSQL$<instancename>\FTDATA	
SQL Server Profiler - trace data file (.trc)		profiler tracing を手動で設定したとき、またはサーバーに対して C2 監査を有効にしたときに生成されます。
SQL 監査ファイル (.sqlaudit)		
SQL クエリファイル (.sql)		Transact-SQL ステートメントを含みます。
SQL Server Analysis Services (SSAS) の{データ|バックアップファイル|ログファイル}	%ProgramFiles%¥Microsoft SQL Server¥MSSQL.X¥OLAP¥{Data|Backup|Log}	SSAS は SQL Server の標準機能として提供される分析専用（特に現状分析）のサービスです。ディレクトリのパスは{DataDir|BackupDir|LogDir} プロパティで指定されます。
SSAS 処理中に使用する一時ファイル	%ProgramFiles%¥Microsoft SQL Server¥MSSQL.X¥OLAP¥Data	ディレクトリのパスは TempDir プロパティで指定され、既定では空となっています。

既定のデータディレクトリに格納されていない SSAS のパーティション		パーティションを作成する際、これらの場所がパーティションウィザードの Processing and Storage Locations ページの Storage location セクションで定義されています。
FILESTREAM データ ファイル		特有のファイル拡張子はありません。ファイルは、sys.database_files の FILE_STREAM 型のコンテナによって識別されるフォルダ構造の下にあります。
Reporting Services の一時ファイル (RSTempFiles) とログファイル (LogFiles)		
Reporting Services の		{RSTempFiles|LogFiles}
拡張イベントファイルのターゲット (.xel / .xem)		システムによって生成されるファイルは、そのインスタンスの LOG フォルダに保存されます。
例外ダンプファイル (.mdmp)		（同上）
インメモリ OLTP ファイル（SQL Server 2014 以降のバージョン）		• ネイティブ手続きとインメモリテーブル定義関連ファイルである。インスタンスの DATA ディレクトリの下の xtp サブフォルダに存在します。ファイル形式は次の通り： ┗ xtp_<t/p>_<dbid>_<objid>.c ┗ xtp_<t/p>_<dbid>_<objid>.dll ┗ xtp_<t/p>_<dbid>_<objid>.obj ┗ xtp_<t/p>_<dbid>_<objid>.out ┗ xtp_<t/p>_<dbid>_<objid>.pdb ┗ xtp_<t/p>_<dbid>_<objid>.xml • チェックポイントファイルとデルタファイル 特有のファイル拡張子は無い。ファイルは、sys.database_files の FILE_STREAM 型のコンテナによって識別されるフォルダ構造の下にあります。
DBCC CHECKDB ファイル		これらは一時ファイルであり、ファイル形式は次の通り： <Database_data_filename.extension> _MSSQL_DBCC <database_id_of_snapshot>
sqlservr.exe プロセス	%ProgramFiles%\Microsoft SQL Server\<*Instance_ID*>.<*Instance_Name*>\MSSQL\Binn\sqlservr.exe	<Instance_ID>=MSSQL??（既定） <Instance_Name>=MSSQLSERVER（既定）
ReportingServicesService.exe プロセス	%ProgramFiles%\Microsoft SQL Server\<*Instance_ID*>.<*Instance_Name*>\Reporting Services\ReportServer\Bin\ReportingServicesService.exe	
MSMDSrv.exe プロセス	%ProgramFiles%\Microsoft SQL Server\<*Instance_ID*>.<*Instance_Name*>\OLAP\Bin\MSMDSrv.exe	

参考

[36] SQL Server を実行しているコンピューター上で実行するウイルス対策ソフトウェアを選択する方法
[37] How to choose antivirus software to run on computers that are running SQL Server
[38] File Locations for Default and Named Instances of SQL Server
[39] c2 audit mode Server Configuration Option

5. 付録

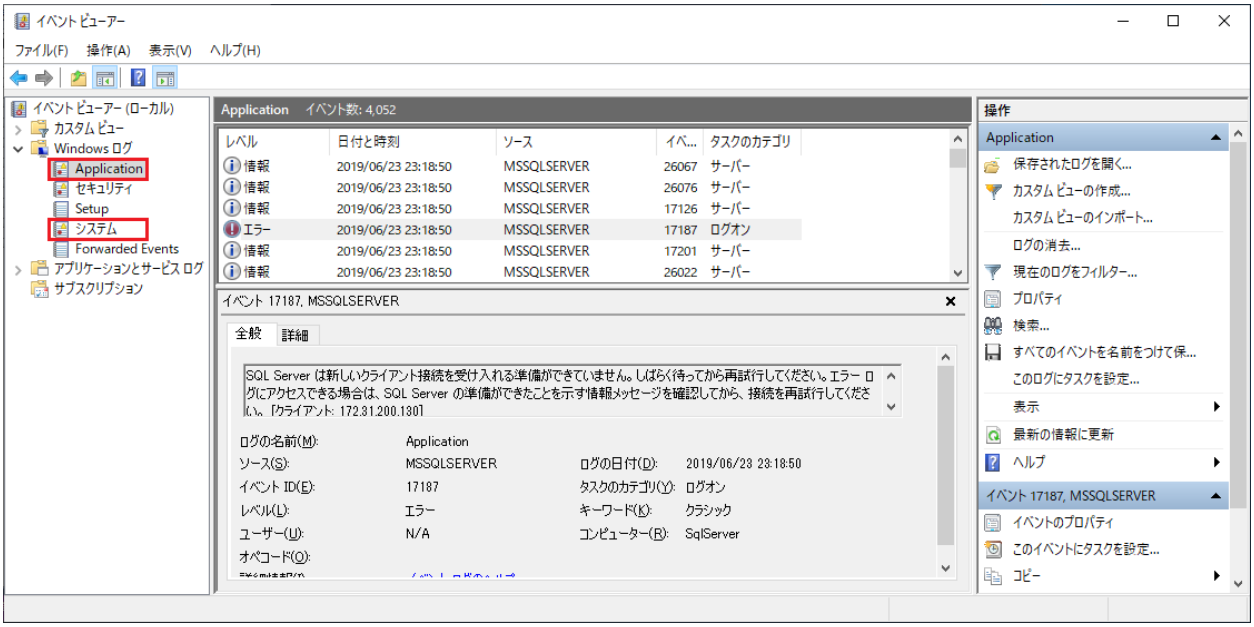
5.1. サーバー障害時のトラブルシューティングに関する Tips

Orchestrator の障害時は、必然的にログを解析することになります。

5.1.1 解析対象のログ

イベントログ

[イベントビューアー] > [Windows ログ] から確認できます。トラブルシュート時に頻繁に利用されるログは **Application ログ** と **システムログ** です。



IIS アクセスログ

IIS では自らのサービスの状態と挙動を監視するための機能を提供しています。

<保存場所>

C:\inetpub\logs\LogFiles\W3SVC<サイト ID>\u_ex<yyMMdd>.log

<既定のアクセスログのフィールド>

フィールド名	説明
date time	リクエストを受信した日時（UTC 形式）
s-ip	サーバーの IP アドレス
cs-method	使用された HTTP メソッド
cs-uri-stem	操作のターゲット

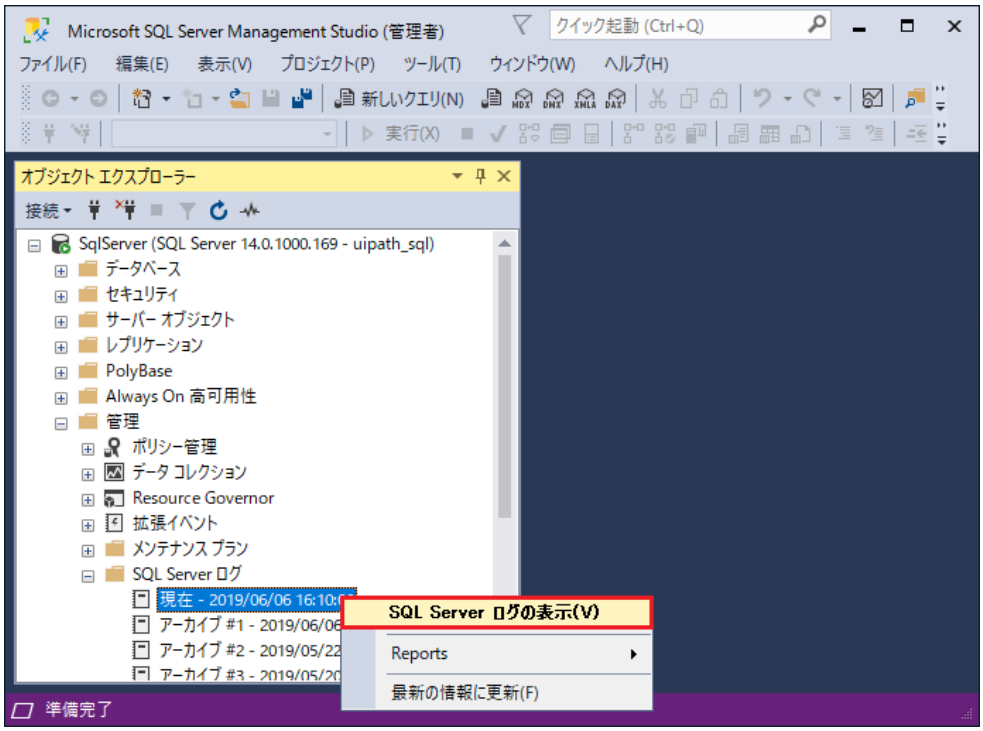
cs-uri-query	ターゲットのクエリ情報
s-port	サーバーのポート番号
cs-username	ユーザー名
c-ip	クライアントの IP アドレス
cs(User-Agent)	リクエストに使用されたクライアントの情報
cs(Referer)	ユーザーが利用した直前の参照元サイトのこと。このサイトが、現在のサイトへのリンクを提供したことになります。
sc-status	HTTP 状態コード
sc-substatus	HTTP 副状態コード
sc-win32-status	Windows 状態コード
time-taken	リクエストに処理した時間

IIS エラーログ (HTTPErr)

<保存場所>
%SystemRoot%\system32\System32\LogFiles\HTTPERR\httperr*.log

SQL サーバーログ

SSMS > [オブジェクト エクスプローラー] > [管理] > [SQL Server ログ]を右クリックして、[SQL Server ログの表示(V)]をクリックすることで表示できます。



5.1.2 Log Parser

Log Parser はさまざまなログを入力として受け付け、それらの中から必要な情報を素早く検索したり、特定の情報

を抜き出して並べ直したり、Excel 用のデータに加工したり等、多様なログの分析を支援する Microsoft 製の無料ツールです。大きな特徴の 1 つは、これらのデータ操作にデータベース問い合わせ言語の SQL 文を使えることです。

Log Parser は、マイクロソフトのテスターが自身の必要に迫られて開発したもので、マイクロソフトから無償で提供されています。以下特に断りなく “Log Parser” と表記した場合は、Log Parser 2.2 を指すものとします。

使用例

1)　Log Parser の実行ファイルが格納されているディレクトリに移動します。

```
※ PowerShell で実行
PS C:\Users\<User Name>> cd "C:\PROGRA~2\Log Parser 2.2"
```

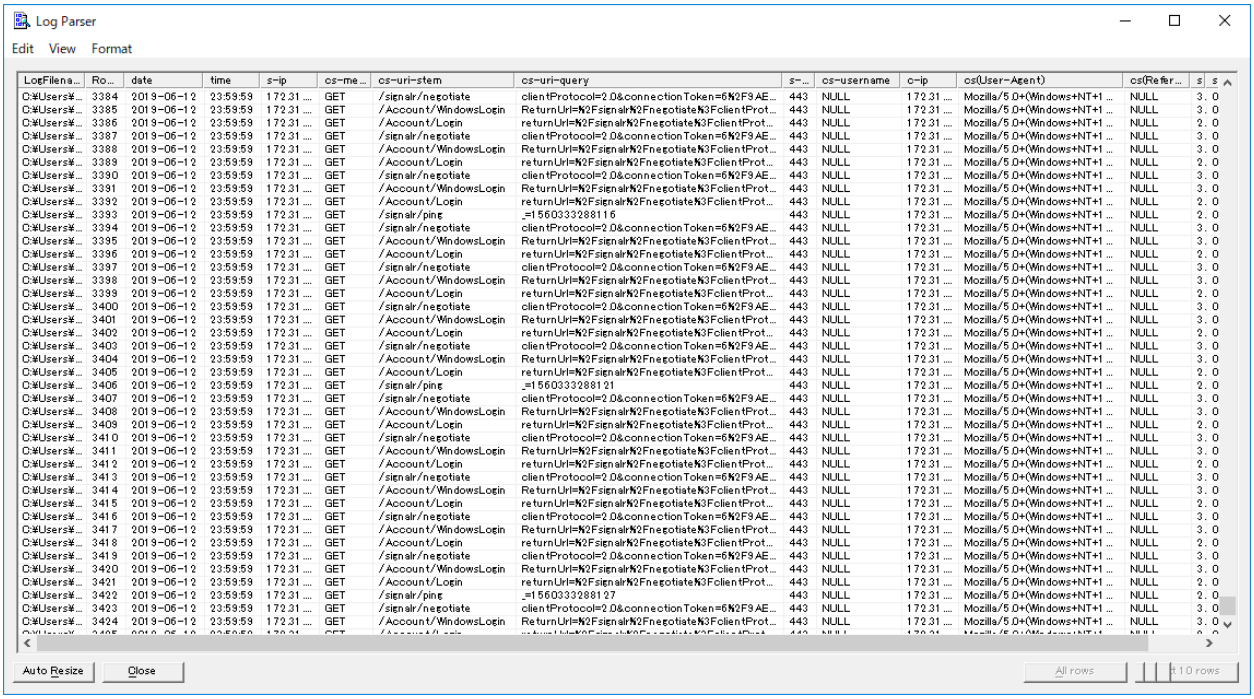
2-a)　イベントログを CSV ファイルに出力する例です。

```
※ PowerShell で実行
PS C:\Program Files (x86)\Log Parser 2.2> .\logparser -i:EVT -o:csv "select * INTO '<filePath>\iis.csv' from '<filePath>\iis.log'"
```

2-b)　IIS アクセスログを DataGrid に表示する例です。

```
※ PowerShell で実行
PS C:\Program Files (x86)\Log Parser 2.2> .\logparser -i:W3C -o:datagrid "select * INTO '<filePath>\iis.csv' from '<filePath>\iis.log'"

統計情報:
---------
処理された要素: 3421
出力された要素: 3421
実行時間:       284.91 秒 (00:04:44.90)
```

<Datagrid>
Datagrid のデータは範囲選択して Excel にコピーすることもできます。



5.2. WCF 通信チャネルの設定

Robot Executor と Robot Service 間で大容量データを送受信する際、Robot からの応答時間が遅れたり、ワークフローが突然終了することがあります。これは例えば「... stopped due to unexpected process termination」といったエラーの原因となります。

優先度：任意

検討タイミング：運用後

推奨

maxReceivedMessageSize プロパティで Robot Executor と Robot Service 間の WCF 通信チャネルサイズを既定値の 1MB 以上に増やします。設定値を高くすることで、ワークフロー実行の遅延、または中断の解決が期待されます。

方法

1) 以下のディレクトリに移動します。

   C:\Program Files (x86)\UiPath\Studio

2) UiPath.Service.Host.exe.config を開く。

3) system.serviceModel タグ内の下記の項目を変更します。ここでは maxReceivedMessageSize を 10MB に設定しています。設定箇所が 2 つあるが、双方を一致させるのが無難です。

```
<!-- Web.config -->
<netNamedPipeBinding>
        <binding name="UiPathRemoteBinding" receiveTimeout="00:10:00"
        sendTimeout="00:10:00" maxReceivedMessageSize="10485760" />
        <binding name="ExecutorBinding" receiveTimeout="00:30:00" sendTimeout="00:20:00"
        maxReceivedMessageSize="10485760" />
</netNamedPipeBinding>
```

備考

各 Binding 名と WCF 通信路の関係を下記のテーブルに示します。

Binding 名	通信路
UiPathRemoteBinding	Executor (Robot) と Service 間
ExecutorBinding	Studio と Service 間

参考

[40] WCF メッセージのサイズ構成

5.3. 動的管理ビューによる監視

動的管理ビュー (英: Dynamic Management View; DMV) をクエリ実行により利用することで、サーバー全体のパフォーマンス情報を数値で確認できます。

優先度：任意

検討タイミング：運用後

推奨

次の項目を定期的に監視する。

A) クエリ処理による CPU 使用率
B) サーバーのメモリ負荷
C) ディスク I/O

方法

A) CPU 負荷の高いクエリを特定してチューニング
sys.dm_exec_query_stats 動的管理ビューを利用します。下記クエリを実行して、プランの実行で使用された CPU 時間の合計 (total_worker_time) をプランが実行された回数 (execution_count) で割った値を平均 CPU 負荷として、ワースト 100 件を表示します。ただし、実行頻度（execution_count）が少なけれ

ば、それほど重要ではありません。

```
-- SQL 文
SELECT TOP 100
    total_worker_time/execution_count AS avg_cpu_cost,
    execution_count,
    SUBSTRING(st.text, (qs.statement_start_offset/2)+1,
    ((CASE qs.statement_end_offset WHEN -1 THEN DATALENGTH(st.text)
ELSE qs.statement_end_offset
    END - qs.statement_start_offset)/2) + 1) AS query_text, query_plan
FROM sys.dm_exec_query_stats AS qs
    CROSS APPLY sys.dm_exec_sql_text(sql_handle) AS st
    CROSS APPLY sys.dm_exec_query_plan(plan_handle) AS qp
ORDER BY avg_cpu_cost DESC;
```

現在キャッシュされているクエリに対する集計情報のみが表示されるため、メモリ負荷が高くなれば、過去に実行された高コストのクエリがキャッシュから削除されている可能性もあります。よって、定期的な抽出により、高コストのクエリの特定に寄与すると期待されます。

下記のクエリを実行することで、SQL Server のインスタンスを定期的に監視し、メモリ使用率が通常の範囲内であることを確認します。

```
-- SQL 文
SELECT
(physical_memory_in_use_kb/1024) AS Memory_usedby_Sqlserver_MB,
(locked_page_allocations_kb/1024) AS Locked_pages_used_Sqlserver_MB,
(total_virtual_address_space_kb/1024) AS Total_VAS_in_MB,
process_physical_memory_low,
process_virtual_memory_low
FROM sys.dm_os_process_memory;
```

B) SQL Server のバッファーキャッシュの確認

sys.dm_os_buffer_descriptors 動的管理ビューを利用して、データベース毎のバッファープールを調べます。下記のクエリを実行して、SQL Server のバッファーキャッシュを消費しているテーブルまたはインデックスのリストを出力します。

```
-- SQL 文
SELECT
    DB_NAME(database_id) AS [Database Name]
    ,CAST(COUNT(*) * 8/1024.0 AS DECIMAL (10,2))  AS [Cached Size (MB)]
FROM sys.dm_os_buffer_descriptors WITH (NOLOCK)
WHERE database_id not in (1,3,4) -- system databases
    AND database_id <> 32767 -- ResourceDB
GROUP BY DB_NAME(database_id)
ORDER BY [Cached Size (MB)] DESC OPTION (RECOMPILE);
```

この動的管理ビューは、SQL Server 2017 ではバッファー プール拡張ファイルのデータ ページに関する情報も返します。

C) ディスク I/O の負荷の確認

A) と同様に、sys.dm_exec_query_stats 動的管理ビューを利用します。下記クエリを実行して、論理読み取りの合計数 (total_logical_reads) と論理書き込みの合計数 (total_logical_writes) の合計の多いものから順に上位 100 件を表示させます。

```
-- SQL 文
SELECT TOP 100
(qs.total_logical_reads/qs.execution_count) AS avg_logical_reads,
(qs.total_logical_writes/qs.execution_count) AS avg_logical_writes,
(qs.total_physical_reads/qs.execution_count) AS avg_phys_reads,
execution_count,
SUBSTRING(st.text, (qs.statement_start_offset/2)+1,
((CASE qs.statement_end_offset WHEN -1 THEN DATALENGTH(st.text)
ELSE qs.statement_end_offset
END - qs.statement_start_offset)/2) + 1) AS query_text, query_plan
FROM sys.dm_exec_query_stats AS qs
CROSS APPLY sys.dm_exec_sql_text(sql_handle) AS st
CROSS APPLY sys.dm_exec_query_plan(plan_handle) AS qp
ORDER BY (total_logical_reads + total_logical_writes) DESC;
```

備考

A)　CPU 負荷が高い場合のチェック項目

パフォーマンスカウンターの **SQL Server: SQL Statistics オブジェクト**には、コンパイルの動作や、SQL Server のインスタンスに対する要求の種類を監視するカウンターが用意されている。クエリのコンパイルと再コンパイルの回数、および SQL Server のインスタンスが受信するバッチの数を監視することで、SQL Server がユーザークエリを処理する速度や、クエリ最適化 (クエリオプティマイザー) によるクエリ処理の効果が評価できます。例えば次のカウンターを確認しましょう。

92) \SQL Server:SQL Statistics\Batch Requests/sec
93) \SQL Server:SQL Statistics\SQL Compilations/sec
94) \SQL Server:SQL Statistics\SQL Re-Compilations/sec

既に実行しているプランを再生成する必要が発生した場合、クエリのリコンパイルが発生します。リコンパイル中は COMPILE ロックが発生するため同時実行性の低下に繋がります。リコンパイルが発生するタイミングとしては、例えば下記の場合があります：

- スキーマの変更
- 実行プランで使用しているインデックスの変更
- sp_recompile による手動リコンパイル
- 参照されるテーブルに大量の変更があった場合
- 統計情報の更新
- 遅延コンパイル
- SET オプションの変更
- 一時テーブルの変更
- ストアドプロシージャ作成時に WITH RECOMPLIE オプションを指定

- クエリヒントとして、RECOMPILE オプションを指定

B) メモリ負荷が高い場合のチェック項目

メモリはトランザクション構造、データ ファイルのバッファー、ガーベジコレクション構造等のシステム オブジェクトによっても消費されます。パフォーマンスカウンターの **SQL Server: Buffer Manager オブジェクト**には、次に示す項目の SQL Server による使用状況を監視するためのカウンターが用意されています：

- データページを保存するメモリ
- SQL Server によるデータベースページの読み書きに伴う物理 I/O を監視するカウンター
- ソリッドステート ドライブ (SSD) などの高速不揮発性記憶域を使用してバッファーキャッシュを拡張するバッファープール拡張

SQL Server で使用されるメモリやカウンターを監視すると、次のことを確認できます：

- 物理メモリが適切でないことによるボトルネックが存在するか否か。頻繁にアクセスするデータをキャッシュに格納できない場合、SQL Server はデータをディスクから取得する必要があります。
- メモリを増設したり、データキャッシュや SQL Server の内部構造により多くのメモリを使用できるように設定することで、クエリのパフォーマンスを向上できるか否か。
- SQL Server がディスクからデータをどの程度の頻度で読み取る必要があるか否か。メモリアクセスなどの他の操作に比べ、物理 I/O は多くの時間を要します。物理 I/O を少なくできれば、クエリのパフォーマンスの向上が期待されます。

SQL Server: Buffer Manager オブジェクトにおける重要なカウンター例を下記に示します。

63) \SQL Server:Buffer Manager\Buffer cache hit ratio
64) \SQL Server:Buffer Manager\Checkpoint pages/sec
65) \SQL Server:Buffer Manager\Lazy writes/sec
66) \SQL Server:Buffer Manager\Page Life Expectancy

```
-- SQL 文
SELECT [バッファキャッシュヒット率] = ( opc.cntr_value * 1.0 / bcr.cntr_value ) * 100.0
FROM sys.dm_os_performance_counters opc
JOIN (SELECT cntr_value,
            object_name
     FROM sys.dm_os_performance_counters
     WHERE counter_name = 'Buffer cache hit ratio base'
     AND object_name LIKE '%Buffer Manager%') bcr
ON  opc.OBJECT_NAME = bcr.OBJECT_NAME
WHERE opc.counter_name = 'Buffer cache hit ratio'
AND opc.OBJECT_NAME LIKE '%Buffer Manager%'
```

C) 不要に多くのページを読み込んでいるクエリがないかを確認

パフォーマンスカウンターの **LogicalDisk オブジェクト**および **PhysicalDisk オブジェクト**も重要です。SQL Server がディスクからの読み取り要求やディスクへの書き込み要求を発行してからその要求が完了するまでに 15 秒を超えると、エラーメッセージ 833 がイベントログに記録されます。SQL Server からこのエラーが報告された場合は、ディスク I/O サブシステムに問題があると推察されます。この長い遅延が SQL Server 環境のパフォ

ーマンスに深刻な悪影響を及ぼす可能性があります。

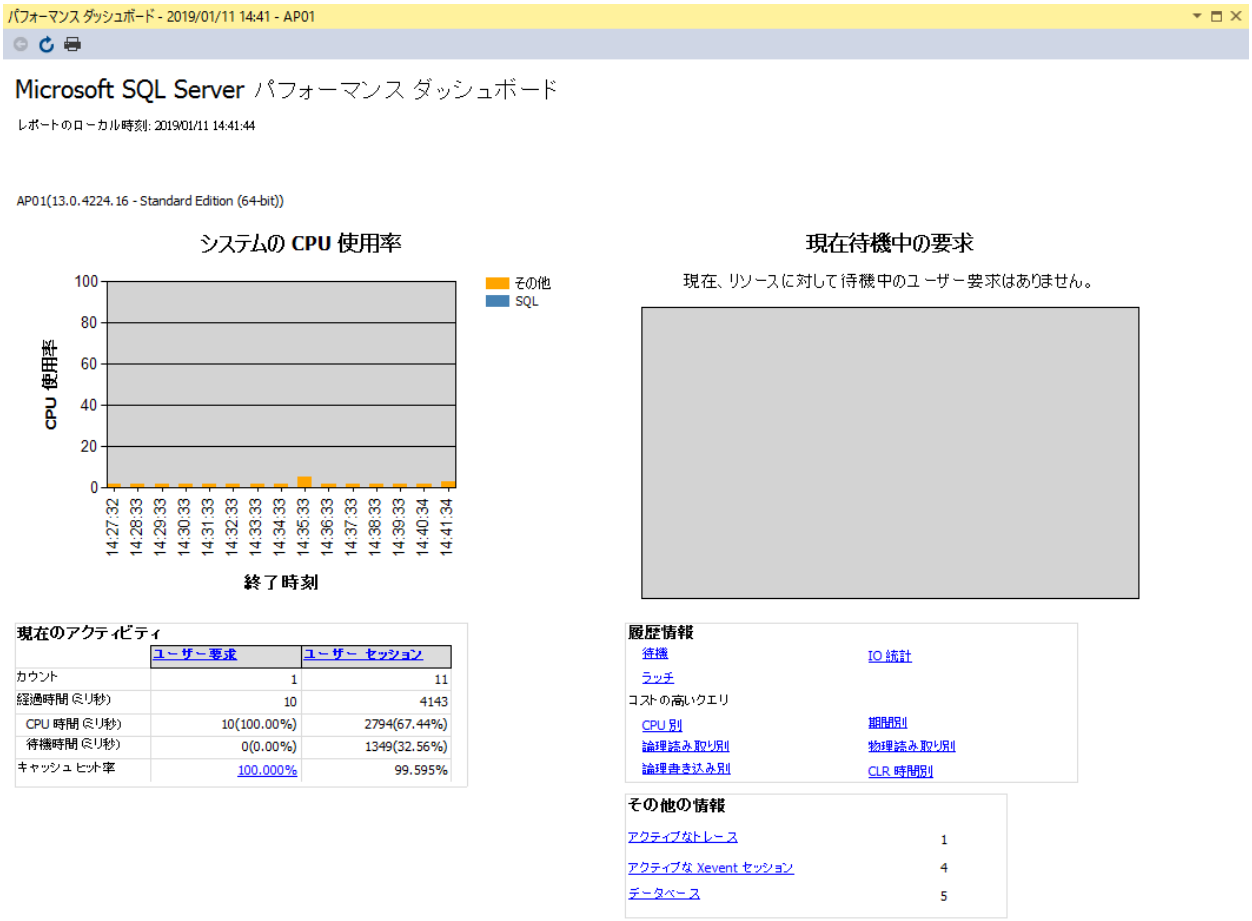
75) \PhysicalDisk(_Total)¥Avg. Disk sec/Transfer
76) \PhysicalDisk(_Total)\Avg. Disk Read Queue Length
77) \PhysicalDisk(_Total)\Avg. Disk Write Queue Length
78) \PhysicalDisk(_Total)\Current Disk Queue Length

5.4. サーバーの監視および性能測定ツールの例

5.4.1. SQL Server の固有機能としての監視ツール

パフォーマンス ダッシュボード

SQL Server Management Studio (SSMS) を開き、[**サーバー名**] を右クリックします。[**レポート(P)**] > [**標準レポート**] > [**パフォーマンス ダッシュボード**]から、パフォーマンス ダッシュボードを起動できます (SSMS 17.2 以降)。



データコレクター

データコレクターは、様々なデータのセットを収集する SQL Server **2017** のコンポーネントです。継続的に、またはユ

ーザー定義のスケジュールに基づいて実行され、収集したデータを管理データウェアハウスというリレーショナルデータベースに格納します。

- サーバーの利用状況
- クエリ統計
- ディスク使用量

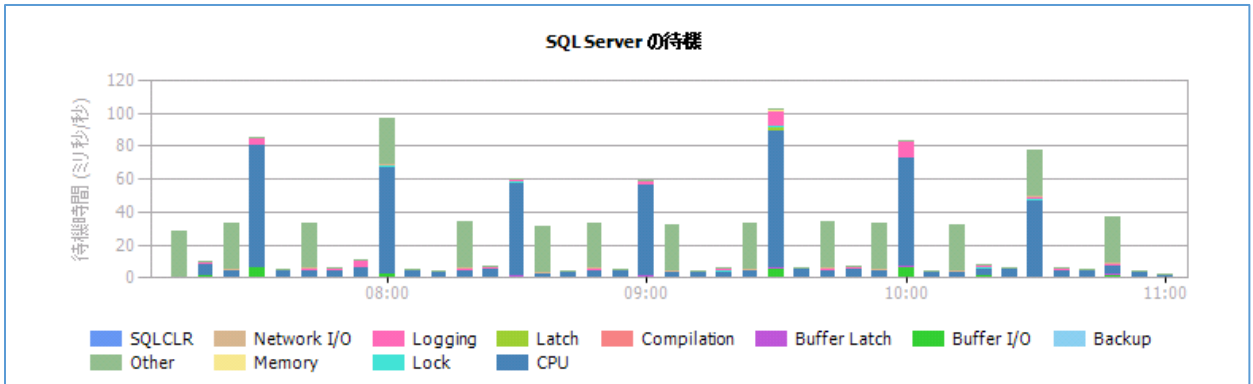
<データ コレクターの有効化>
ストアド プロシージャ **dbo.sp_syscollector_enable_collector** を使用した次のクエリを実行して、データ コレクターを有効にします。

```
-- SQL 文
USE [DBNAME];
GO
EXEC dbo.sp_syscollector_enable_collector;
```

<データ コレクターの無効化>
ストアド プロシージャ **dbo.sp_syscollector_disable_collector** を使用した次のクエリを実行して、データ コレクターを無効にします。

```
-- SQL 文
USE [DBNAME];
GO
EXEC dbo.sp_syscollector_disable_collector;
```

このプロシージャの実行には、EXECUTE 権限を持つ **dc_admin** または **dc_operator** 固定データベース ロール メンバーシップが必要です。



待機のカテゴリ	完了した待機	待機時間 (ミリ秒/秒)	% (合計待機時間に対する割合)
⊟ CPU	19,460	14.777	54.88%
CPU (Consumed)		14.311	53.15%
CPU (Signal Wait)		0.463	1.72%
SOS_SCHEDULER_YIELD	19,460	0.002	0.01%
⊞ Other	289	9.744	36.19%
⊞ Logging	11,247	1.316	4.89%
⊟ Buffer I/O	68	0.693	2.57%
ASYNC_IO_COMPLETION	9	0.637	2.37%
PAGEIOLATCH_EX	27	0.033	0.12%
PAGEIOLATCH_UP	32	0.023	0.08%
IO_COMPLETION	0	0.000	0.00%
PAGEIOLATCH_SH	0	0.000	0.00%
⊞ Network I/O	53,103	0.323	1.20%

クエリストア

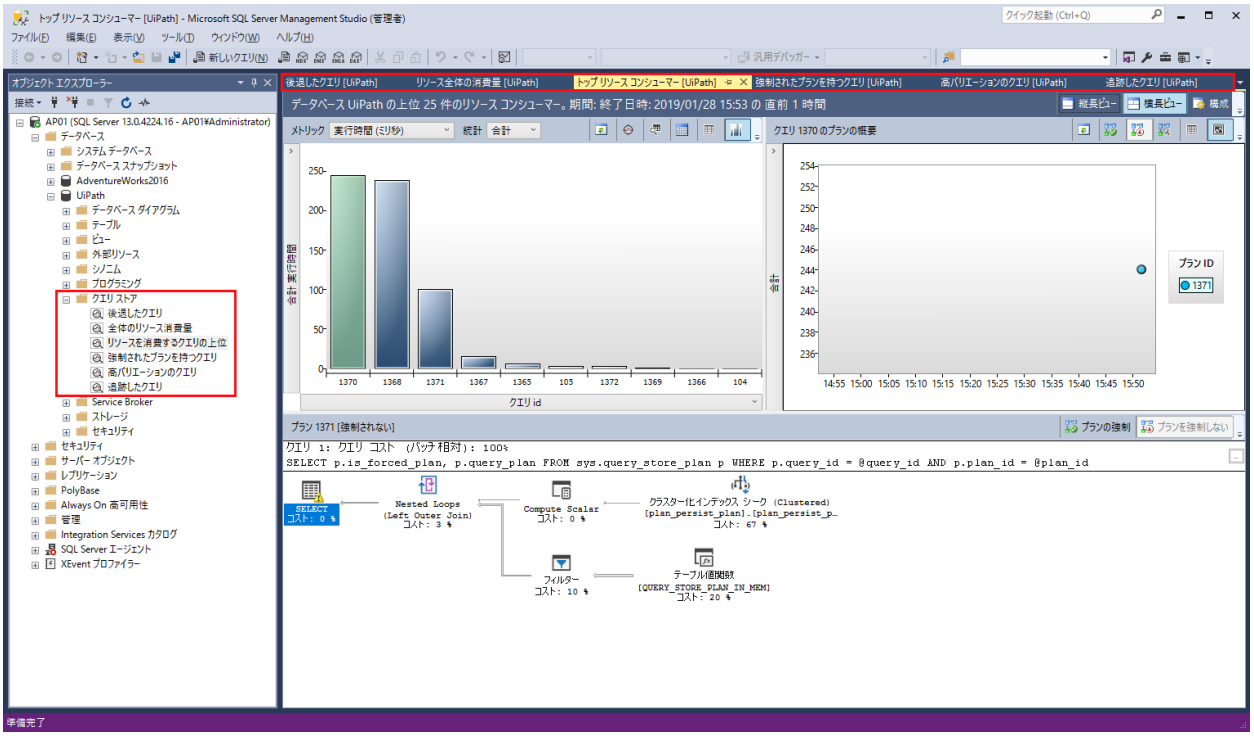
SQL Server のクエリストア機能により、クエリプランの選択やパフォーマンスを把握できる。これにより、クエリプランの変更によって生じるパフォーマンスの違いを早急に検知でき、パフォーマンス上のトラブルシューティングを簡略化させます。クエリストアは、自動的にクエリ、プラン、および実行時統計の履歴をキャプチャし、確認用に保持します。データは時間枠で区分されるため、データベースの使用パターンを表示して、サーバー上でクエリプランが変わった時点を確認することができます。

- クエリの実行統計
- データベースのリソース消費

GUI または **ALTER DATABASE SET** オプションを使用してクエリストアを構成することができます。

```
-- SQL 文
ALTER DATABASE <dbname> SET QUERY_STORE = ON
```

注) master データベースまたは tempdb データベースに対しては、クエリストアを有効にできません。



5.5. PaaS 環境における SQL Server のバックアップ

5.5.1. Hyper-V ホストで推奨されるウイルススキャン除外項目

推奨

ディレクトリ、プロセス、およびファイルを除外するようにウイルス対策ソフトウェアのリアルタイム スキャン コンポーネントを設定します。詳細は [**参考**] セクションに示す資料を参照してください。

事象例

Hyper-V 仮想マシンを作成または開始しようとすると、次のいずれかの問題が発生します。

① 次のいずれかのエラー メッセージが表示されます。

- エラー メッセージ 1

要求された操作はユーザー マップ セクションで開いたファイルでは実行できません。(0x800704C8)

- エラー メッセージ 2

VMName' Microsoft 代理イーサネット ポート (インスタンス ID{7E0DA81A-A7B4-4DFD-869F-37002C36D816}): 電源をオンにできませんでした。エラー: '指定されたネットワーク リソースまたはデバイスは利用できません。' (0x80070037)。

- エラー メッセージ 3

スレッドの終了またはアプリケーションの要求によって、I/O の処理は中止されました。(0x800703E3)

② 仮想マシンが Hyper-V 管理コンソールで表示されなくなります。

参考

[41] 仮想マシンを起動または作成しようとすると、仮想マシンが見つからないか、エラー 0x800704C8、0x80070037、または 0x800703E3 が表示される

5.5.2. Azure SQL Database 自動バックアップについての詳細情報

- バックアップは自動で作成され、ユーザーの操作は必要ありません。
- 既定のバックアップ保持期間はサービス レベルにより異なります。
  - Basic: 1 週間
  - Standard: 5 週間
  - Premium: 5 週間

  最大 10 年間まで設定可能です。

参考

[42] Automated backups

5.5.3. Amazon RDS for SQL Server

- バックアップは自動で作成され、ユーザーの操作は必要ありません。
- 自動バックアップの保持期間は、最大 35 日間まで設定可能です。

参考

[43] https://aws.amazon.com/jp/rds/sqlserver/