IBM SPSS Analytic Server
バージョン 3.2.2

管理者ガイド

注記

本書および本書で紹介する製品をご使用になる前に、35 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM® SPSS® Analytic Server バージョン 3、リリース 2、モディフィケーション 2、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに 適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典：
IBM SPSS Analytic Server
Version 3.2.2
Administrator's Guide

発行：
日本アイ・ビー・エム株式会社

担当：
トランスレーション・サービス・センター

目次

第 1 章 テナント管理

テナントにより、オブジェクトをテナント間で共有できないように、ユーザー、プロジェクト、およびデータ・ソースの大まかな分類が提供されます。各ユーザーは、割り当てられているテナントに照らしてシステムにアクセスします。

Analytic Server コンソールでテナントを管理して、ユーザーをテナントに割り当てます。「テナント」ページのビューは、コンソールにログオンしているユーザーの役割によって異なります。

- インストール時にセットアップされる「スーパーユーザー」管理者は、テナント管理者です。このユーザーのみが新規テナントを作成して、すべてのテナントのプロパティーを編集することができます。
- 管理者役割のユーザーは、ログインしているテナントのプロパティーを編集できます。
- ユーザー役割のユーザーは、テナントのプロパティーを編集できません。「テナント」ページは、これらのユーザーに対して非表示になります。
- 読者役割のユーザーは、データ・ソースを編集できません。また、Analytic Server コンソールにログインすることもできません。

管理者は、「プロジェクト」ページと「データ・ソース」ページにアクセスして、クリーンアップと管理のためにプロジェクトやデータ・ソースを管理できます。詳しくは、「*IBM SPSS Analytic Server* ユーザーズ・ガイド」を参照してください。

テナントのリスト表示

メインの「テナント」ページには、既存のテナントが表に表示されます。「スーパーユーザー」管理者のみがこのページで編集を行うことができます。

- テナントの名前をクリックして、その詳細を表示し、プロパティーを編集します。
- テナントの URL をクリックして、そのテナントのコンテキストでコンソールを開きます。

  注 : コンソールからログアウトされ、そのテナントに有効な資格情報を使用してログインする必要があります。
- 名前に検索文字列が含まれるテナントのみを表示するようにリスト表示をフィルタリングするには、検索域に入力します。
- 「**新規**」をクリックして、「**新規テナントの追加 (Add new tenant)**」ダイアログで指定した名前で新規テナントを作成します。テナントに付ける名前に関する制約事項については、2 ページの『命名規則』を参照してください。
- 選択したテナントを削除するには、「**削除**」をクリックします。
- リスト表示を更新するには、「**更新**」をクリックします。

個々のテナントの詳細

コンテンツ領域は、いくつかの縮小可能なセクションに分かれています。

詳細

名前
テナントの名前を表示する編集可能なテキスト・フィールド。

説明
テナントに関する説明テキストを指定できる編集可能なテキスト・フィールド。

URL
これは、Analytic Server コンソールを使用してテナントにログインし、SPSS Modeler サーバーの構成に使用するためにユーザーに指定する URL です。SPSS Modeler の構成について詳しくは、「*IBM SPSS Analytic Server* インストールと構成のガイド」を参照してください。

状態
「**アクティブ (Active)**」なテナントは現在使用されています。テナントを「**非アクティブ (Inactive)**」にすると、ユーザーがそのテナントにログインすることを防ぎますが、基本情報は削除されません。

プリンシパル

プリンシパルは、インストール時にセットアップされるセキュリティー・プロバイダーから引き出されるユーザーとグループです。管理者、ユーザー、または読者は、プリンシパルをテナントに 追加できます。

- テキスト・ボックスに入力すると、検索文字列が名前に含まれるユーザーとグループがフィルタリングされます。ドロップダウン・リストから「**管理者**」、「**ユーザー**」、または「**読者 (Reader)**」を選択して、テナント内の役割を割り当てます。「**参加者の追加 (Add participant)**」をクリックして、作成者のリストに追加します。
- 参加者を削除するには、メンバー・リストからユーザーまたはグループを 選択して、「**参加者を削除 (Remove participant)**」をクリックします。

メトリック
テナントのリソースの制限を構成できます。テナントによって現在使用されているディスク・スペースを報告します。

- テナントの最大ディスク・スペース割り当て量を設定できます。この制限に達すると、テナントのディスク・スペース使用量が割り当て量を下回るように十分なディスク・スペースが消去されるまで、このテナントでディスクにこれ以上のデータを書き込むことはできません。
- テナントのディスク・スペース警告レベルを設定できます。この割り当て量を超えると、テナントのディスク・スペース使用量が割り当て量を下回るように十分なディスク・スペースが消去されるまで、このテナントでプリンシパルが 分析ジョブを実行依頼することはできません。
- このテナントで一度に実行できる並行ジョブの最大数を設定できます。この割り当て量を超えると、現在実行中のジョブが完了するまで、このテナントでプリンシパルが 分析ジョブを実行依頼することはできません。
- 1 つのデータ・ソースが保持できるフィールドの最大数を設定できます。データ・ソースが作成または更新されるたびに、この制限が確認されます。
- ファイルの最大サイズをメガバイトで設定できます。ファイルがアップロードされる 際にこの制限が確認されます。

セキュリティー・プロバイダー構成
ユーザー認証プロバイダーを指定できます。「**デフォルト**」では、インストールおよび構成時にセットアップされたデフォルトのテナントのプロバイダーが 使用されます。「**LDAP**」では、Active Directory や OpenLDAP などの外部 LDAP サーバーを使用してユーザーを認証できます。プロバイダーの設定を指定して、オプションで「プリンシパル」セクションで選択可能なユーザーとグループを制御するためのフィルター設定を指定します。

命名規則

データ・ソースやプロジェクトなど、Analytic Server で固有の名前を付けることができるすべてのものの名前には、以下の規則が適用されます。

- 1 つのテナント内では、同じタイプのオブジェクト内で名前が固有でなければなりません。例えば、2 つのデータ・ソースの両方に insuranceClaims という名前を付けることはできませんが、データ・ソースとプロジェクトのそれぞれに insuranceClaims という名前を付けることはできます。
- 名前では大文字と小文字が区別されます。例えば、insuranceClaims と InsuranceClaims は固有の名前と見なされます。
- 名前では、先頭と末尾の空白文字は無視されます。
- 以下の文字は、名前では無効です。

```
~, #, %, &, *, {, }, ¥¥, :, <, >, ?, /, |, ", ¥t, ¥r, ¥n
```

第 2 章 ユーザーの作業開始

`http://<host>:<port>/<context-root>/admin/<tenant>` にナビゲートし、ユーザー名およびパスワードを入力して Analytic Server コンソールにログオンするようにユーザーに通知します。

注 : Analytic Server コンソールのログイン・プロンプトに入力されるユーザー名は、レルム名の接尾辞なしで入力されます。このため、複数のレルムが定義されている場合は、ユーザーに「**レルム (Realms)**」ドロップダウン・リストが表示され、該当するレルムを選択できます。レルムが 1 つしか定義されていない場合は、Analytic Server へのサインイン時に「**レルム (Realms)**」ドロップダウン・リストは表示されません。

<host>
Analytic Server ホストのアドレス。

<port>
Analytic Server が listen するポート。デフォルトは 9080 です。

<context-root>
Analytic Server のコンテキスト・ルート。デフォルトは analyticserver です。

<tenant>
複数テナント環境では、ユーザーが所属するテナント。単一テナント環境の場合、デフォルトのテナントは **ibm** です。

例えば、ホスト・マシンの IP アドレスが 9.86.44.232 であり、"mycompany" テナントを作成してユーザーを追加済みで、その他の設定はデフォルトのままである場合、ユーザーは `http://9.86.44.232:9080/analyticserver/admin/mycompany` にナビゲートして Analytic Server コンソールにアクセスします。

第 3 章 Analytic Server ジョブ名

Analytic Server は、map-reduce ジョブおよび Spark ジョブを作成します。これらのジョブは、Hadoop クラスターの Resource Manager のユーザー・インターフェースを 使用してモニターできます。

map-reduce ジョブ名は以下の構造になっています。

```
AS/{tenant name}/{user name}/{algorithm name}
```

{tenant name}
これは、ジョブが実行されるテナントの名前です。

{user name}
これは、ジョブを要求したユーザーです。

{algorithm name}
これは、ジョブの 1 次アルゴリズムです。単一のストリームが複数の map-reduce ジョブを生成する可能性があることに注意してください。同様に、1 つのストリーム内の複数の操作が単一の map-reduce ジョブに含まれる可能性があります。

Resource Manager のユーザー・インターフェースにはすべての map-reduce ジョブが表示されます。Analytic Server ごとに Spark アプリケーションが 1 つ開始されます。Spark ジョブをモニターするには、Spark アプリケーションのユーザー・インターフェースを 開きます (ジョブ名は「**Description**」列に表示されます)。

第 4 章 Analytic Server カスタム・プロパティー

以下のカスタム・プロパティーは、Analytic Server `analytic.cfg` ファイルに定義済みであるもの、あるいは構成できるものです。

表 *1. Analytic Server* カスタム・プロパティー

プロパティー名	型	デフォルト値	使用可能な値	説明
admin.username	ストリング			Analytic Server の管理ユーザー名を定義します。
ae.cluster.ha.cascade.failure.protection	ブール	TRUE		有効 (true) にすると、クラスター作業マネージャーは、複数のクラスター・メンバーで失敗しているジョブによってすべてのクラスター・サーバーがクラッシュするのを防ぎます。これは、問題のジョブを永続的に中断させるか、指定の検疫サーバーでのみ実行可能にすることで行います。

表 *1. Analytic Server* カスタム・プロパティー *(*続き*)*

プロパティー名	型	デフォルト値	使用可能な値	説明
ae.cluster.heapdump_onexit.filename	ストリング			ファイル名を指定すると、複数の **CPU_Starvation** イベントが発生した場合に、Analytic Server JVM が heap-dump を (指定したファイルに) 書き込みます。
ae.cluster.job.artifacts.cleanup.delay.minutes	整数	5		ジョブが完了した後、zookeeper のジョブ関連の成果物をクリーンアップするまで Analytic Server が待機する時間。
ae.cluster.quarantine.server.name	ストリング			クラスター環境内の、検疫対象ジョブ (失敗回数のしきい値を超えたジョブ) の実行に使用する Analytic Server サーバーを指定します。

表 1. *Analytic Server* カスタム・プロパティー (続き)

プロパティー名	型	デフォルト値	使用可能な値	説明
ae.cluster.queue.callback.threadpool.size	整数	20		zookeeper 経由で送信されたプロセス間クラスター・メッセージを消費する場合にクラスター・サービスが使用する **ThreadPool** サイズ。
ae.cluster.thread.scheduler.delay.detector	整数	30		ローカルのパフォーマンス上の問題を検出します (例えば、CPU 不足のメッセージをログに記録)。
ae.cluster.thread.scheduler.detect.error	整数	10		ローカルのパフォーマンス上の問題を検出します (例えば、CPU 不足のメッセージをログに記録)。
as.db.connect.method	ストリング	Basic	Kerberos|Basic	Analytic Server データベースのデータ・ソース接続方式を指定します。
as.spark.driver.cleanup.delay	整数	2		ログアウト後に Spark クライアント JVM が強制終了するまでの時間 (分単位)。

表 *1. Analytic Server* カスタム・プロパティー *(*続き*)*

プロパティー名	型	デフォルト値	使用可能な値	説明
cleanup.delay	整数	20		各バックグランド・クリーンアップ (プロジェクト・ファイルなど) が実行される間隔の遅延分数。
default.project.versions.tokeep	整数	25		古いプロジェクト・バージョンをクリーンアップする際に残しておくプロジェクト・バージョンの番号。
distrib.fs.root	ストリング	/user/as_user/analytic-root		Analytic Server の分散ファイル・システム用のベース・フォルダー。
hive.precheckPermission	ブール	TRUE		**TRUE** に設定すると、Analytic Server は HDFS ファイルのアクセス権を確認して、データベース表のデータ・ロケーションに対するユーザー・アクセス権を検証します。
hive.sql.check	ブール	FALSE		生成された SQL ステートメントに EXPLAIN 接頭辞を追加します。

表 1. *Analytic Server* カスタム・プロパティー (続き)

プロパティー名	型	デフォルト値	使用可能な値	説明
io.sort.mb	整数	10		ファイルのソート時に使用するバッファー・メモリーの総量 (メガバイト単位)。デフォルトでは、マージ・ストリームごとに 1MB が割り当てられ、これによってシークが最小限に抑えられます。詳細については、http://hadooptutorial.info/hadoop-performance-tuning/ を参照してください。
java.security.krb5.conf	ストリング			Kerberos の `krb5.conf` ファイルの場所。
jndi.aedb.driver	ストリング			Analytic Server Metastore のドライバー・クラス。
jndi.aedb.password	ストリング			Analytic Server Metastore のパスワード。
jndi.aedb.url	ストリング			Analytic Server Metastore のリポジトリー JDBC 接続ストリング。

表 *1. Analytic Server* カスタム・プロパティー *(*続き*)*

プロパティー名	型	デフォルト値	使用可能な値	説明
jndi.aedb.username	ストリング			Analytic Server Metastore のユーザ名。
join.small.data.size	整数	1048576		分析エンジンがマップ側のアルゴリズムで結合を試みるデータの最大量 (バイト単位)。
mapred.child.java.opts	ストリング	"-server"		マップ用の JVM ヒープ・サイズを制御し、Hadoop で実行されるタスクを削減します。この値は、クラスター内のノードが処理できる最大限の値に設定します。
max.asl.size	整数	20971520		ASL プログラムの最大許容サイズ (バイト単位)。
max.datamodel.size	整数	20971520		データ・モデル XML ストリングの最大許容サイズ (バイト単位)。
mmr.taskparallel.targets.threshold	整数	100		ターゲットとコアの比率がこのしきい値より低い場合、ジョブは M3R によって処理されます。 .

表 *1. Analytic Server* カスタム・プロパティー *(*続き*)*

プロパティー名	型	デフォルト値	使用可能な値	説明
mmr.threads	整数	4		メモリー内での MapReduce (M3R) ジョブに使用するしきい値の数。(*2)
mmr.upper.bound.threshold	数値	100		M3R によって処理されるデータの最大量。大容量のデータは Hadoop または Spark によって処理されます。
nested.groups	ストリング	enabled	enabled| disabled| null (未定義)	LDAP でネストされたグループを使用するかどうかを指定します。
node.max.jobs	整数	50		Analytic Server クラスター・メンバーで同時実行できるジョブの最大数。
orchestrator.thread.pool	整数	30		分析エンジンの作業 (**EngineCommands** など) を送信する際に使用される、オーケストレーターのスレッド・プール・サイズ。

表 1. *Analytic Server* カスタム・プロパティー (続き)

プロパティー名	型	デフォルト値	使用可能な値	説明
orchestrator.thread.pool.fixed	ブール	TRUE		オーケストレーターのスレッド・プールが、弾力性があるか固定であるかを指定します。
preferred.mapreduce	ストリング	spark	m3r|hadoop|spark	使用する優先 MapReduce エンジンを定義します。
resource.pool.default(*1)	ストリング			**resource.pool.mapping** にコンシューマー・マッピングが見つからない場合に **spark.scheduler.pool** 値を設定します。詳しくは、https://spark.apache.org/docs/latest/job-scheduling.html を参照してください。
resource.pool.enabled	ブール	FALSE		カスタム YARN キュー・マッピング定義 (**yarn.queue.mapping**) の使用を有効にします。

表 *1. Analytic Server* カスタム・プロパティー *(*続き*)*

プロパティー名	型	デフォルト値	使用可能な値	説明
resource.pool.mapping(*1)	マップ		(a:b,c:d....) ここで、a と c は AS コンシューマー名、b と d は YARN リソース・プール名です。	Analytic Server コンシューマー名を YARN リソース・プール名にマップします。
session.max.inactivity.time	整数	14400		HTTP セッション・タイムアウト値 (秒単位)。デフォルトは 14400 秒 (4 時間) です。
spark.cache	ブール	TRUE		Spark クライアント JVM でキャッシュに入れた Spark RDD を活用するかどうかを決定します。パフォーマンス上の理由から、デフォルト値の **TRUE** のままにしておくことをお勧めします。
spark.dependency.exclude.regex	ストリング		Spark クライアント JVM クラスパスから *.jar ファイルを除外するための有効な正規表現	問題のある *.jar ファイルを Spark クライアント JVM クラスパスから除外します。

表 1. *Analytic Server* カスタム・プロパティー (続き)

プロパティー名	型	デフォルト値	使用可能な値	説明
spark.version	ストリング	2.x		Analytic Server が Spark ジョブの実行に使用する HDP/CDH スタックでの Spark のバージョン
split.sort.mb	整数	100		**io.sort.mb** 値を設定します。詳細については、https://hadoop.apache.org/docs/r2.4.1/hadoop-mapreduce-client/hadoop-mapreduce-client-core/mapred-default.xml を参照してください。
yarn.queue.default	ストリング	default		**yarn.queue.mapping** で有効なマッピングが見つからない場合に使用するデフォルトの YARN キュー名。

表 1. *Analytic Server* カスタム・プロパティー (続き)

プロパティー名	型	デフォルト値	使用可能な値	説明
yarn.queue.mapping	マップ		(a:b,c:d....) ここで、a と c は Analytic Server のコンシューマー名かユーザー名 (**yarn.queue.mode** で決定した方)、b と d は YARN キュー名です。	名前 (ユーザーまたはコンシューマー) を YARN キューにマップします。
yarn.queue.mode	ストリング		user|tenant	**mapreduce.job.queuename** に **userName** を使用するか **consumerName** を使用するかを決定します。詳細については、https://hadoop.apache.org/docs/r2.4.1/hadoop-mapreduce-client/hadoop-mapreduce-client-core/mapred-default.xml を参照してください。

第 5 章 IBM SPSS Analytic Server ベスト・プラクティスと推奨事項

以下のセクションは、データ・ソース、クラスター構成、および IBM SPSS Modeler ストリームに関する Analytic Server のベスト・プラクティスと推奨事項について説明します。

データ・ソース

Analytic Server では、 次のデータ・ソース・タイプがサポートされます。

- ファイル・ベースのデータ・ソース (区切りテキスト、固定テキスト、Microsoft Excel ファイルなど)。
- リレーショナル・データベース (Db2、Oracle、Microsoft SQL Server、Teradata、Postgres、Netezza、MySQL、Amazon Redshift など)。
- Hive/HCatalog データ・ソース。これには、すべての組み込みデータ型 (例えば ORC や Parquet) と、適切な Hive Serializer-Deserializer 実装が使用できるカスタム・データ型が含まれます。さらに Analytic Server は、NoSQL データベース (HBase、MongoDB、Accumulo、Cassandra、Oracle NoSQL など、適切な Hive ストレージ・ハンドラー実装が使用できるデータベース) にアクセスするように構成できます。

  注 : Parquet のサポートは Hive 表の読み取りと付加に制限されています。表の情報を上書きする必要がある場合、上書き処理はデータ値、ならびにデータ・モデルが変更される原因となる可能性があるため、新規の表が作成されます。
- 地理空間タイプのデータ・ソース (形状ファイル・ベースおよびマップ・サービス・ベース)。

Analytic Server の Hive/HCatalog データ・ソースに関する制限

- SPSS Modeler 条件抽出ノードで Hive プッシュバックが必要な場合、フィルター式は、パーティション化された STRING 型の列のみを参照できます。Analytic Server 3.0 以降では、パーティション化された列 TINYINT、SMALLINT、INT、BIGINT のデータ型サポートが追加されました。 Hive データ・ソースに指定される静的フィルター式には、パーティション化されたすべてのデータ型の列に関してフィルター式を設定できます。
- Analytic Server では、Hive ビューに基づく HCatalog データ・ソースはサポートされません。Hive ビューに基づくその他のデータ・ソース・タイプ (Hive SQL など) はすべてサポートされます。

Cloudera 上の大容量データを持つ Hive データ・ソース

Cloudera 上の大容量データを持つ Hive データ・ソースの処理をスピードアップするには、Apache Spark を有効にすることをお勧めします。

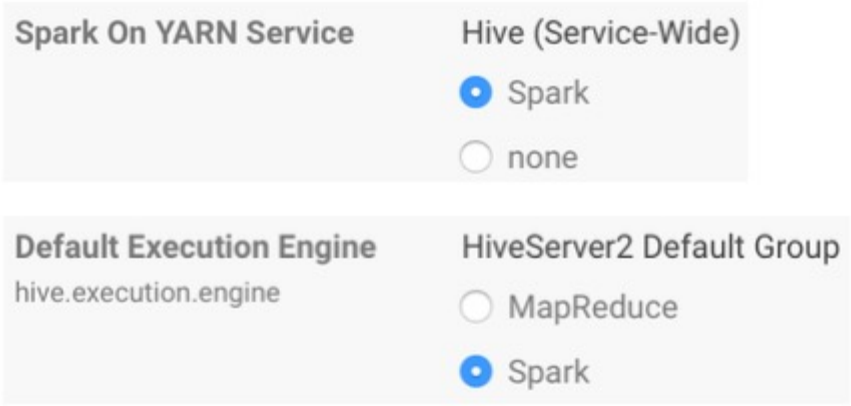

図 *1. Spark* 設定

クラスター構成 - セキュリティー

Kerberos の偽名

バージョン 3.0.1 より前では、Analytic Server インスタンスは、Kerberos セキュリティーが有効な場合は Analytic Server キータブのユーザー・プリンシパル名を HDFS 操作の認証に使用していました。バージョン 3.0.1 以降では、Analytic Server は Kerberos の偽名を使用する HDFS 操作を認証するために、Analytic Server キータブのサービス・プリンシパル名を要求側 (REST 要求を行うユーザーの) ユーザー名と共に使用します。Analytic Server 3.0.1 以上では、Kerberos が有効になっているクラスターで稼働する場合、偽名の構成属性を HDFS (または Hive サービス構成) に追加する必要があります。HDFS の場合は、以下のプロパティーを HDFS `core-site.xml` ファイルに追加します。

```
hadoop.proxyuser.<analytic_server_service_principal_name> .hosts = *
hadoop.proxyuser.<analytic_server_service_principal_name> .groups = *
```

ここで、`<analytic_server_service_principal_name>` は Analytic Server 構成の `Analytic_Server_User` フィールドに指定されたデフォルトの `as_user` 値です。

HDFS から Hive/HCatalog 経由でデータにアクセスする場合は、以下のプロパティーも HDFS `core-site.xml` ファイルに追加する必要があります。

```
hadoop.proxyuser.hive.hosts = *
hadoop.proxyuser.hive.groups = *
```

Kerberos のクロスレルム認証

Analytic Server では、Kerberos のクロスレルム認証がサポートされます。この機能を有効にするには、まず KDC クロスレルム認証が有効になっていることを確認し、次に、以下の設定を Analytic Server Ambari 構成の **Custom analytics.cfg** セクションに追加します。

```
kerberos.user.realm.trim = true
```

クラスター構成 - パフォーマンス調整設定および結果

Spark 構成

Analytic Server は、`yarn-client` モードを使用して YARN と対話し、Hadoop クラスターで Spark ジョブを実行します。

Analytic Server カスタム構成:

- Ambari 設定は、Analytic Server Ambari 構成の **Custom analytics.cfg** セクションで定義されています。
- Cloudera 設定は、Cloudera Manager の **Analytic Server Advanced Configuration Snippet (Safety Valve) for analyticserver-conf/config.properties** セクションにあります。

1. Analytic Server カスタム構成で構成項目を追加して、**spark.driver.memory** 構成設定の値を大きくすること検討してください (明示的に設定されていない場合、デフォルト値は 1g)。例:

   ```
   spark.driver.memory=2g
   ```

2. 以下のいずれかの Analytic Server （Spark リソースの使用オプションを指定) を選択してください。

   - **オプション A: 静的リソース割り振り構成**

     Analytic Server カスタム構成で、以下の 3 つのパラメーターを構成する必要があります。

     ```
     spark.executor.instances
     spark.executor.cores
     spark.executor.memory
     ```

     以下の手順は、パラメーター値の決定方法を示しています。

     a. CPU およびメモリーに関して、Analytic Server が Spark に永続的に割り振ることができるパーセンテージを設定します。これは結果として、各マシンで使用できる特定のコア数 (C) および一定量のメモリー (M) になります。

b. 各マシンが実行できる executor の数 (E) を設定します。これらの executor は各クラスター・ノードで別個の Hadoop コンテナー (プロセス) として実行されます。通常は、2 より大きい値が適切ですが、コアの総数よりも小さい値でなければなりません。Spark に割り振られるメモリーは、これらの executor 間で分割されるため、このパラメーターに対して高い値を選択すると、各コンテナーに割り振られるメモリー量は減少します。

c. executor 当たりの使用コア数 (CE) を設定します。通常、この値は C/E (Spark アプリケーションに割り振られた各マシンのコア数を executor の総数で割った値) です。

d. 各 executor で使用されるメモリー量 (ME) を設定します。これは通常、M/E です。

注 : 使用される executor 数およびコア数は、各 executor のメモリー量が 3G * CE より大きくなるようにバランスをとる必要があります。各 executor の各コアには少なくとも、ストレージまたは計算メモリーとして使用される 3G のメモリー割り当てが必要です。

```
spark.executor.instances = <E>*N /<E> // value established in step b where N is the number of compute nodes
spark.executor.cores = <CE> // value established in step c
spark.executor.memory = <ME> // value established in step d
```

図 *2. Custom analytics.cfg Spark* 設定

- **オプション B: 動的リソース割り振り構成**

  このオプションを使用する場合、YARN によって割り振られたすべての executor が、クラスター全体の実際の使用可能なリソースに応じて、動的に増減されます。

  最小構成は次のとおりです。

```
spark.dynamicAllocation.enabled = true
spark.shuffle.service.enabled = true
```

  標準的な構成は次のとおりです。

```
spark.default.emitter.class = com.spss.ae.spark.ListEmitter
spark.default.emitter.compressed = false
spark.dynamicAllocation.enabled = true
spark.executor.cores = 4
spark.executor.memory = 16g
spark.io.compression.codec = snappy
spark.rdd.compress = true
spark.shuffle.service.enabled = true
```

  注 :

  – `spark.executor.instances = <E>` は使用しないでください。これを使用すると、静的リソース割り振りが使用されます。

  – executor のコアとメモリーの値に関する考慮事項は、オプション A と同じです。

3. 以下の設定を使用すると、Analytic Server カスタム構成で Spark キャッシュを無効にできます。

```
spark.cache=false
spark.storage.memoryFraction = 0.3
```

図 *3. Custom analytics.cfg Spark* キャッシュ設定

大規模な IBM SPSS Modeler ストリームを使用する場合は、Spark キャッシュを無効にするべきではありません。このインスタンスで Spark キャッシュを無効にすると、ストリームの実行速度は低下します

が、指定された executor 当たりのメモリー量が小さい場合に発生する可能性のあるメモリー不足状態を回避できます。

JVM 構成

Ambari 設定:

1. Analytic Server Ambari 構成で、サーバーがローカル処理に使用できるメモリーの量を設定します。小規模から中規模のストリームでは、デフォルト値 (2 GB) は問題なく使用できますが、大規模なストリームの場合は、より高い値のヒープ・サイズ (例えば、10 GB) を使用する必要があります。

   「**Analytic Server**」 > 「**構成**」 > 「**Advanced analytic-jvm-options**」
2. `-Xmx2048M` を `-Xmx10G` に置き換えて、構成を保存し、Analytic Server を再起動します。

図 *4. Advanced analytic-jvm-options* 設定

Cloudera 設定:

1. Cloudera Manager で Analytic Server サービスの「**構成 (Configuration)**」タブに移動し、「`jvm-options`」 コントロールを更新してサーバーがローカル処理に使用できるメモリーの量を設定します。小規模から中規模のストリームでは、デフォルト値 (2 GB) は問題なく使用できますが、大規模なストリームの場合は、より高い値のヒープ・サイズ (例えば、10 GB) を使用する必要があります。

   「**Analytic Server サービス (Analytic Server service)**」 > 「**構成 (Configuration)**」 > 「**jvm-options**」
2. `-Xmx2048M` を `-Xmx10G` に置き換えて、構成を保存し、Analytic Server を再起動します。

YARN MapReduce2 構成:

- Analytic Server の実行で MapReduce ジョブを Spark ジョブと並行して実行しなければならない場合は、YARN コンテナー当たりのメモリーが少なくとも 4 GB となるように YARN クラスターを構成する必要があります。

Zookeeper 構成:

- Cloudera では、Zookeeper 構成を手動で更新する必要があります。詳細については、https://www.cloudera.com/documentation/enterprise/5-4-x/topics/cdh_ig_zookeeper_server_maintain.html を参照してください。
- 複雑な SPSS Modeler ストリームまたはワイド・データ (多数のフィールド) を使用する場合は、Analytic Server–Zookeeper 接続が切断されたためにジョブが失敗するという問題に直面する可能性があります。この問題は、SPSS Modeler Server が Analytic Server に送信する大きなプログラム・サイズがもたらした結果です。この問題は、Analytic Server 3.0 (またはそれ以上) で発生する可能性は低いです。以下のステップを使用して、この問題を解決します。

  1. Ambari コンソールで、Zookeeper サービスの「**Configs**」タブにナビゲートし、「**Advanced zookeeper-env**」の下の `zookeeper-env` テンプレートに以下の行を追加して、Zookeeper サービスを再始動します。

```
export JVMFLAGS="-Xmx2048m -Djute.maxbuffer=2097152"
```

図 *5. zookeeper-env* テンプレートの設定

2. Ambari コンソールで、Analytic Server サービスの「**Configs**」タブにナビゲートし、以下を **Advanced analytics-jvm-options** に追加して、Analytic Server サービスを再始動します。

```
-Djute.maxbuffer=2097152
```

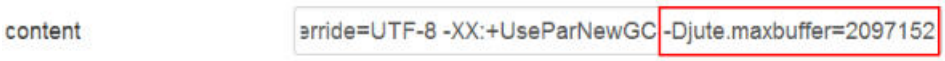

図 6. *Advanced analytics-jvm-options* 設定

注 : 問題が解決しない場合は、両方の場所で `-Djute.maxbuffer` 値を `2097152` から `4194304` に増やしてください。

IBM SPSS Modeler ストリームの推奨事項

注 : 以下の推奨事項のほとんどは、スモールデータにも適用されます。

スモールデータでのプロトタイプ化

ストリームを試行する場合、たいていはいくつかのノードを追加し、その時点までのストリームをテストし、おそらく表出力またはグラフィカル出力を確認するためのノードを追加して、ストリームの作成を続けます。通常は、ストリームをテストするたびに、ビッグデータのデータ・パスを実行するわけにはいきません。

ビッグデータの適切なデータ・サンプルを作成すると、完全なデータ・パスを実行する場合に必要となる時間をかけることなく、実際のデータに対してストリームをテストできます。データ・サンプルには、ストリームを正常に実行するための十分なデータが含まれていなければなりません。例えば、ミネソタ州の店舗での取引を分析する場合、データ・サンプルには、ミネソタ州の店舗からの取引が含まれていなければなりません。

サンプリング後は、次のいずれかを実行できます。

- ビッグデータが存在するクラスター上にデータ・サンプルのキャッシュを作成します。

  長所 - 単純であり、ソース・ノードの切り替えは不要です。

  短所 - セッションが終了するとキャッシュは消えます。
- データ・サンプルを含む新規 Analytic Server データ・ソースを作成します。

  長所 - 永続データ・ソースです。

  短所 - ソース・ノードの編集/切り替えが必要です。
- データ・サンプルをローカル・システムにダウンロードして、ローカル・データ・ソースを作成します。

  長所 - プロトタイピングのときにクラスター・リソースを消費しません。スモールデータを処理する場合は、SPSS Modeler クライアントのほうが Analytic Server よりも効率的です。

  短所 - ソース・ノードを切り替える必要があります。

ソース・ノードとは別のデータ型ノードとフィルター・ノードを作成する

各 SPSS Modeler ソース・ノードには、フィルター・ノードとデータ型ノードを組み合わせた機能もあります。これは、キャンバスを合理化された状態にしておくのに役立ちますが、異なるソース・ノード・タイプに切り替える場合は困難になります。さらに、データ型やフィルターの操作が発生しているという事実が分かりにくくなります。

フィルター・ノードと条件抽出ノードを可能な限りソース・ノードの近くに配置する

これにより、ダウンストリームの操作でのレコード数が削減されます。

可能な限りソート・ノードを避ける

Analytic Server では、ソート対象のデータに依存するノード (結合ノードなど) での最適化はサポートされません。 そのため、中間ストリームのソート・ノードはあまり実用的ではありません。ソート・ノードは、上位 N 件 (または下位 N 件) のレコードを取得する目的でその直後にサンプル・ノードが続く場合に、価値があります。

使用されるフィールドのみを計算する

フィールドを計算せず、その後すぐにそのフィールドをフィルタリングします 。

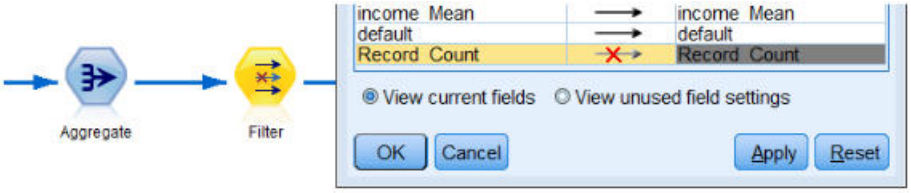

図 *7. Modeler* フィールドのオプション

可能な限り、分かりにくい式にすることなく、多数の一時フィールドを作成しないようにします。例えば、以下の例のように定義するとします。

```
now = datetime_now()
birthdate = datetime_date(bYear, bMonth, bDay)
age = date_years_difference(birthdate, now)
```

この場合、代わりに以下の例のように定義してください。

```
age = date_years_difference(datetime_date(bYear, bMonth, bDay), datetime_now())
```

このようにして一時的データをインライン式に組み込むと、多数のフィールドが変換される場合にパフォーマンスが向上する可能性があります。

データ・ソースでのストレージの設定

中間ストリームでフィールドのストレージ・タイプを (例えば、文字列から整数に) 変更する操作は、パフォーマンス全体に悪影響をもたらす可能性があります。Analytic Server コンソールでデータ・ソースを定義するときに、フィールドのストレージを設定すると、これらの変換の繰り返しを避けることできます。

スモールデータを処理するときは SPSS Modeler を使用する

Analytic Server でビッグデータを操作し、その後 SPSS Modeler を使用してスモールデータの計算を完了げます。

適切な Analytic Server 関連ストリーム・プロパティーを選択する

関連するストリーム・プロパティー (「**ツール**」 > 「**オプション**」 > 「**ストリームのプロパティー**」 > 「**Analytic Server**」) を構成して、(Analytic Server でノードを実行できない場合に) データ処理が Analytic Server からドロップアウトして SPSS Modeler で続行できるようにするかどうかを決定します。

デフォルトでは、SPSS Modeler はこの状況でエラーを報告し、実行を停止するように構成されます。エラーをバイパスするには、設定を「エラー」から「警告」に変更し、SPSS Modeler で処理できるデータ量の制限を調整します。例えば、(必要に応じて) データ転送速度をデフォルトの 10000 レコード値から更新できます。この制限は、SPSS Modeler テーブル・ノードを使用した結果を表示する場合にも適用されるので、注意してください。この制限を超えると、SPSS Modeler は、データ・フェッチがストリーム・プロパティーに設定された制限を超過したことを報告します。

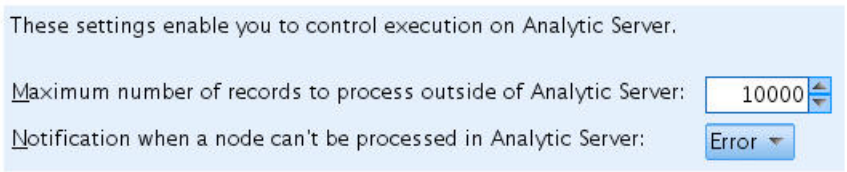

図 *8. Analytic Server* 設定

Analytic Server ソース・ノードの使用

Analytic Server は、さまざまなデータベース・データ・ソースに接続できますが、SPSS Modeler では、(ストリーム全体を 1 つの Analytic Server ジョブとして実行するためには) すべてのソース・ノードが Analytic Server ソース・ノードであることが求められます。 ストリーム全体を Analytic Server で実行するために

は、データベース・ソース・ノードを Analytic Server ソース・ノードに変更し、Analytic Server コンソールで Analytic Server データベース・データ・ソースを作成する必要があります。

サポートされないノードの使用方法の検討

Analytic Server では、すべてのノードがサポートされるわけではありません (行列入替ノードが良い例です)。行列入替操作の結果をストリームの残りの部分と結合し、それを Analytic Server で実行させるには、Analytic Server エクスポート・ノードを使用する Analytic Server データ・ソースに、行列入替ノードが含まれたサブストリームを書き出す必要があります。これにより、Analytic Server に書き込むためにストリームが中断された Analytic Server ソース・ノードを接続できます。

注 : 行列入替操作は、一回限りの操作やほとんど実行しない操作に適していますが、日常的なストリーム操作には使用するべきではありません。

ストリームが Analytic Server で動作するかを実行前に確認する

ストリームを Analytic Server で実行する準備ができたら、ターミナル・ノードを選択し、SPSS Modeler プレビュー機能 (ツールバーの「**プレビュー実行**」コントロール) を使用して、ターミナル・ノードの実行に関連するすべてのノードが Analytic Server で動作することを (ストリームを実行せずに) 確認します。問題はメッセージ・ウィンドウで報告されます。

連続する結合操作をまとめる

一連の結合ノードがキーも結合の種類も同じである場合は、それらの結合ノードを 1 つのノードにまとめます。

同じサブストリームをまとめる

同じサブストリームは、可能な限り (特に、結合やソートなど、コストのかかる操作が含まれている場合は)、まとめるようにしてください。SPSS Modeler はこれらの操作を 1 回実行し、キャッシュを使用してパフォーマンスを改善します。次の例で、ストリームは **newField** ノードまで同一です。

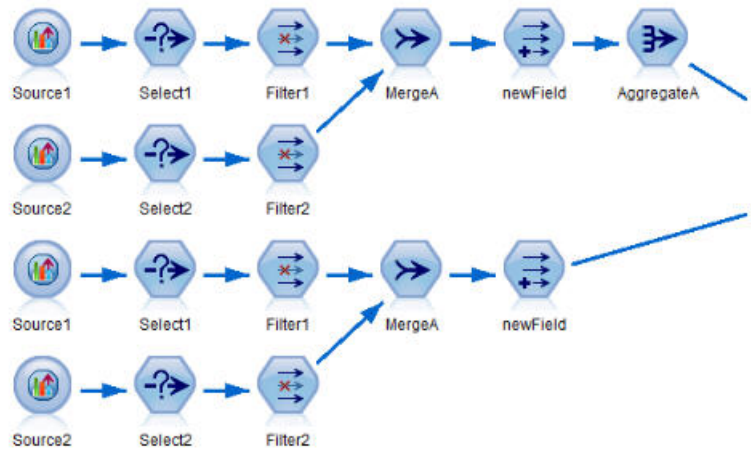

図 *9.* ストリームの例

このサブストリームを次のように構成すると、より効率的に (かつ保守しやすく) なります。

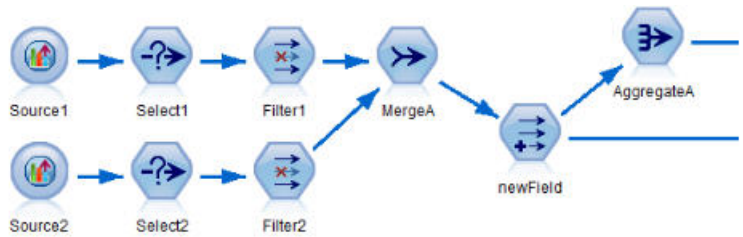

図 *10.* ストリームの例

余分なデータ型ノードの削除

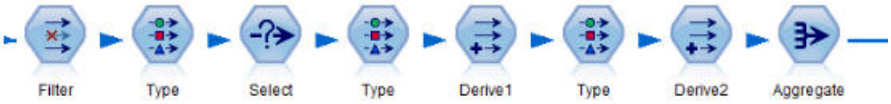

図 *11.* ストリームの例

Analytic Server に対して実行するときは、不要なデータ型ノードを使用しないようにします。データ型ノードの「値の読み込み」操作によって MapReduce ジョブが開始されます。データ型ノード値を消去しない限り、これは通常は一回限りに節約します。

各ストリームの完全な文書化

次の例は、多数のサブストリームが含まれる複雑なストリームを示します。

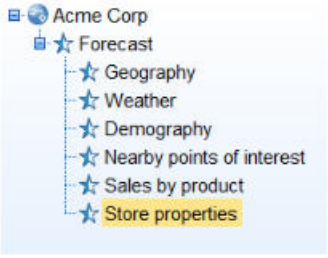

図 *12.* サブストリームの例

このような場合は、スーパーノードに適切な名前を付けて、(コードを文書化するのと同じように) ストリームを文書化することが重要です。明確なコメントを入れることで、ストリームを読んだり保守したりする他の分析者に対して貴重な情報を提供できます。例:

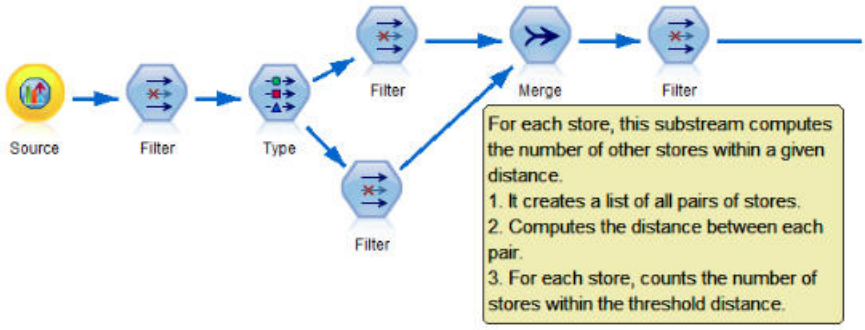

図 *13.* コメント付きストリームの例

ストリーム開発時に SPSS Modeler キャッシュを使用して中間結果を素早く保管する

Analytic Server に対して実行するストリームでは、ノード・キャッシングは、(SPSS Modeler Server への保管とは対照的に) ストリームの特定の部分のデータを HDFS 上の一時ファイルに保管することによって機能します。キャッシュはビッグデータに適しており、 Analytic Server で実行されるストリームで安全に使用できます。

第 6 章 トラブルシューティング

Analytic Server では、問題の判別に役立つ複数のツールを提供しています。

ロギング

Analytic Server は、問題の診断に役立つカスタマー・ログ・ファイルおよびトレース・ファイルを作成します。 デフォルトの Liberty インストール済み環境では、これらのログ・ファイルは {AS_ROOT}/ae_wlpserver/usr/servers/aeserver/logs ディレクトリーにあります。

デフォルトのロギング構成では、2 つのログ・ファイルが作成されます。これらのログ・ファイルは、毎日ロールオーバーします。

as.log
このファイルには、通知目的の警告およびエラー・メッセージの大まかな要約が含まれます。 ユーザー・インターフェースに 表示されるエラー・メッセージでは解決できないサーバー・エラーが発生した場合は、最初にこのファイルを確認してください。

as_trace.log
このファイルには、ae.log 内のすべてのエントリーが含まれますが、これに加え、主に IBM サポートおよび開発を対象としたデバッグ用の情報が追加されます。

Analytic Server は、基礎となるロギング機構として Apache LOG4J を使用します。LOG4J を使用することで、{AS_SERVER_ROOT}/configuration/log4j.xml 構成ファイルを編集してロギングを動的に調整できます。 ロギングの変更は、問題の診断に役立てる目的でサポートから要請される場合も、保持されるログ・ファイルの数を制限するためにユーザーが行う場合もあります。このファイルに対する変更は数秒以内に自動的に検出されるため、Analytic Server を再始動する必要はありません。

log4j およびこの構成ファイルについて詳しくは、Apache 公式 Web サイトの資料 (http://logging.apache.org/log4j/) を参照してください。

バージョン情報

{AS_ROOT}/properties/version フォルダーを調べると、どのバージョンの Analytic Server がインストールされているかが分かります。 バージョン情報は、以下のファイルに含まれています。

IBM_SPSS_Analytic_Server-*.swtag
詳細な製品情報が記載されています。

version.txt
インストールされている製品のバージョンおよびビルド番号。

ログ・コレクター

ログ・ファイルを直接検討しても問題を解決できない場合は、すべてのログをバンドルして IBM サポートに送信することができます。 必要なすべてのデータを容易に収集するためのユーティリティーが 提供されています。

コマンド・シェルを使用して、以下のコマンドを実行します。

```
cd  {AS_ROOT}/bin
   run  >sh ./logcollector.sh
```

これらのコマンドにより、{AS_ROOT}/bin に圧縮ファイルが作成されます。圧縮ファイルには、すべてのログ・ファイルと製品のバージョン情報が含まれています。

一般的な問題

このセクションでは、いくつかの一般的な管理上の問題と、それらの修正方法について説明します。

ストリームの実行

R ジョブによる英語以外の単語の Unicode への変換

Cloudera クラスターで、Hadoop サーバーのシステム・エンコードが UTF-8 ではない場合、R によって英語以外の単語を Unicode に変換します。

1. Cloudera Manager コンソールで YARN 構成タブにナビゲートします。
2. 以下の設定を「NodeManager Environment Advanced Configuration Snippet (Safety Valve)」フィールドに追加します。

```
LC_ALL=""
LANG=en_US.utf8
```

PySpark ジョブを実行できない

Spark サービスがすべての Analytic Server ノードおよびすべてのノード・マネージャーにデプロイされていることを確認します。

Kerberos が有効になっている環境で PySpark ジョブを実行できない

PySpark テストが正常に実行されるためには、事前に `kinit` コマンドを実行してから Analytic Server を再始動する必要があります。例:

HDP Kerberos

```
cd /etc/security/keytabs/
sudo -u as_user kinit -k -t as_user.headless.keytab as_user/lyrh1.fyre.ibm.com@IBM.COM
```

CDH Kerberos

```
cd /run/cloudera-scm-agent/process/387-analyticserver-ANALYTIC_SERVER
sudo -u as_user kinit -k -t analyticserver.keytab as_user/cdh12-1.fyre.ibm.com@IBM.COM
```

XGBoost-AS ノードを実行できない

XGBoost-AS ノードを実行しようとすると、以下のエラーが発生することがあります。

```
ASL の実行中にエラーが発生しました: 実行が失敗しました。理由: ml.dmlc.xgboost4j.java.XGBoostError:
XGBoostModel のトレーニングが失敗しました (Reason: ml.dmlc.xgboost4j.java.XGBoostError:
XGBoostModel training failed)
```

Analytic Server カスタム構成

1. この問題を解決するには、以下の Analytic Server カスタム構成設定を適用します。

```
spark.executor.memory=12g (or above)
```

   - Ambari の設定は、Ambari 構成の **Custom analytics.cfg** セクションに定義されています。
   - Cloudera 設定は、Cloudera Manager の **Analytic Server Advanced Configuration Snippet (Safety Valve) for analyticserver-conf/config.properties** セクションにあります。

2. Analytic Server カスタム構成を更新したら、Analytic Server サービスを再始動します。

IBM SPSS Modeler 構成

IBM SPSS Modeler XGBoost-AS ノードの「**作成オプション**」 > 「**一般**」 > 「**外部メモリーの使用**」設定を有効にします。

メモリー・エラー

executor のメモリー・エラー発生後の YARN の構成

executor に必要なメモリーが最大しきい値を超える場合、以下のエラーが発生することがあります。

```
Caused by: com.spss.mapreduce.exceptions.JobException:
    java.lang.IllegalArgumentException: Required executor memory (1024+384  MB) is above the max
    threshold (1024 MB) of this cluster! Please  increase the value of
    'yarn.scheduler.maximum-allocation-mb'.
```

この問題の解決に必要な YARN 構成設定のステップを以下に示します。

Ambari の場合

1. Ambari のユーザー・インターフェースで 「**YARN**」 > 「**Configs**」 > 「**Settings**」に移動します。
2. メモリー **node (the memory that is allocated for all YARN containers)** を 8192MB に増やします。
3. コンテナーの値を以下の値に増やします。
   - **Minimum Container Size (Memory)** を 682MB に増加
   - **Maximum Container Size (Memory)** を 8192MB に増加
4. **Maximum Container Size (VCores)** を 3 に増やします。
5. YARN、Spark、および Analytic Server サービスを再始動します。

Cloudera の場合

1. `yarn.nodemanager.resource.memory-mb` を 8GB に増やします。
   - Cloudera Manager のユーザー・インターフェースで 「**YARN service**」 > 「**Configurations**」 > 「**Search Container Memory**」に移動し、値を 8GB に増やします。
2. Cloudera Manager のユーザー・インターフェースで「**YARN service**」 > 「**Quick Links**」に移動し、「**Dynamic Resource Pools**」を選択します。
3. 「**Configuration**」にある使用可能なプールの「**edit**」をそれぞれクリックし、「**YARN**」で「**Max Running Apps**」の値を 4 に設定します。
4. YARN、Spark、および Analytic Server サービスを再始動します。

executor に必要なメモリー

executor に必要なメモリーが現在の設定値を超える場合、以下のエラーが発生することがあります。

```
AS Executor が削除されました :SparkListenerExecutorRemoved(1557300399468,716,メモリー制限を超えたため、コンテナーは YARN によって強制終了されました。(AS Executor
Removed:SparkListenerExecutorRemoved(1557300399468,716,Container killed by YARN for exceeding memory limits.)
2 GB の物理メモリー中 2.0 GB が使用されました。(2.0 GB of 2 GB physical memory used.)
spark.yarn.executor.memoryOverhead を増やすことを検討してください。(Consider boosting spark.yarn.executor.memoryOverhead.))
```

1. この問題を解決するには、以下の Analytic Server カスタム構成設定を適用します。

   ```
   spark.executor.memory=4g (or above)
   ```

   - Ambari の設定は、Ambari 構成の **Custom analytics.cfg** セクションに定義されています。
   - Cloudera 設定は、Cloudera Manager の **Analytic Server Advanced Configuration Snippet (Safety Valve) for analyticserver-conf/config.properties** セクションにあります。
2. Analytic Server カスタム構成を更新したら、Analytic Server サービスを再始動します。

Hadoop と Apache Spark 2.2

- Hadoop と Apache Spark 2.2 が同じ環境に存在すると、ほとんどの `forcespark` ジョブと `forcehadoop` ジョブが失敗します。このエラーは、YARN アプリケーション・ログに `java.lang.NoClassDefFoundError: org/apache/hadoop/fs/FSDataInputStream` として記録されます。

  この問題は、`/etc/spark2/conf/spark-defaults.conf` ファイルを次のように手動で編集すると解決できます。

  ```
  #spark.hadoop.mapreduce.application.classpath=
  #spark.hadoop.yarn.application.classpath=
  ```

- 同じシステムに 2 つの JDK バージョンがインストールされていると、Cloudera は JDK 1.7 を使用し、一方、Spark 2.2 は JDK 1.8 を使用します。Apache Spark 2.x で `forcespark` ジョブまたは `forcehadoop` ジョブを実行すると、すべてのジョブが次のエラー・メッセージを返して失敗します。

```
Execution failed. Reason: org/apache/spark/api/java/function/PairFunction : Unsupported
major.minor version 52.0
```

Cloudera について、Cloudera Manager の **Analytic Server Advanced Configuration Snippet (Safety Valve) for server.env** セクションに以下の行を追加してください。

```
JAVA_HOME=/usr/java/jdk1.8.0_152
```

Apache Hive UDF ユーザーへの admin 権限の付与

Analytic Server Apache Hive UDF の登録後に Invalid function エラーが表示されることがあります。デフォルトでは、2 つの Hive 役割 (admin と public) があります。Hive ユーザーは、public 役割に属しています。Hive UDF は、登録ユーザーが admin 特権を持つ (Hive セキュリティーが有効である) ことを要求します。

Hive UDF ユーザーに admin 権限を付与するには、次の手順を実行します。

1. 次のように、Hive として Beeline にログインします。

   ```
   !connect jdbc:hive2://localhost:10000/default;principal=hive/cdh51501.fyre.ibm.com@IBM.COM
   ```

2. Beeline で、次のコマンドを実行します。

   ```
   grant admin to user hive WITH ADMIN OPTION;
   ```

   注 : その他の便利な SQL コマンドを次に挙げておきます。

   ユーザー hive に既に割り当てられている役割を表示します

   ```
   show role grant user hive;
   ```

   public 役割に既に割り当てられているユーザーを表示します

   ```
   show principals public;
   ```

3. Hive を再始動し、Analytic Server Hive UDF を再登録します。

   ```
   sudo -u hive kinit -k -t hive.keytab hive/cdh51501.fyre.ibm.com@IBM.COM
   sudo -u hive hive -f /opt/cloudera/parcels/AnalyticServer/bin/udfUnregister.sql
   sudo -u hive hive -f /opt/cloudera/parcels/AnalyticServer/bin/udfRegister.sql
   ```

HiveDB エラー

HiveDB への書き込み時に次のエラーが表示されることがあります。

```
(AEQAE4805E) 実行が失敗しました。理由: com.google.common.io.Closeables.closeQuietly(Ljava/io/Closeable;)
```

このエラーは、Hadoop Cluster 上の複数のバージョンの guava-*.jar ファイルが原因で発生します。次の手順を実行することで、エラーを解決できます (この例では、HDP 3.1 が使用されています)。

1. Ambari コンソールを開き、Analytic Server サービスを停止します。
2. /usr/hdp/3.1.0.0-78/spark2/jars/guava-14.0.1.jar を {AS_ROOT}/ae_wlpserver/usr/servers/aeserver/apps/AE_BOOT.war/WEB-INF/lib にコピーします。
3. Ambari コンソールで、the Analytic Server サービスをリフレッシュしてから、Analytic Server サービスを開始します。

Hive および HCatalog のデータ・ソースの混合

Analytic Server では、同じ IBM SPSS Modeler ストリームでの Hive データ・ソースと HCatalog データ・ソースの混合はサポートされません。

パフォーマンスの調整

このセクションでは、システムのパフォーマンスを 最適化する方法を説明します。

Analytic Server は Ambari フレームワークのコンポーネントであり、HDFS、YARN、および Spark などの他のコンポーネントを利用します。Hadoop、HDFS、および Spark の一般的なパフォーマンス調整手法が Analytic Server ワークロードに適用されます。Analytic Server ワークロードはそれぞれ異なるため、特定のデプロイメントのワークロードに基づいて調整を試行する必要があります。以下のプロパティーおよび調整のヒントは、Analytic Server のベンチマーク・テストおよびスケーリング・テストの結果に影響を与えるキーの変更です。

最初のジョブが Analytic Server で実行されると、このサーバーは永続 Spark アプリケーションを開始します。このアプリケーションは Analytic Server がシャットダウンされるまでアクティブのままになります。この永続 Spark アプリケーションは、Analytic Server ジョブがアクティブに実行されていない場合でも、すべてのクラスター・リソースを割り当て、割り当てられたすべてのクラスター・リソースを Analytic Server が実行されている間中保持します。Analytic Server Spark アプリケーションに割り当てるリソースの量については慎重に考慮する必要があります。すべてのクラスター・リソースが Analytic Server Spark アプリケーションに割り当てられた場合、他のジョブが遅延したり実行されなかったりする可能性があります。これらのジョブは、十分な空きリソースを待機するためにキューに入れられる可能性があります。それらのリソースは Analytic Server Spark アプリケーションによって消費されます。

複数の Analytic Server サービスが構成およびデプロイされている場合、それぞれのサービス・インスタンスが独自の永続 Spark アプリケーションを割り当てる可能性があります。例えば、高可用性フェイルオーバーをサポートするために 2 つの Analytic Server サービスがデプロイされている場合、2 つの永続 Spark アプリケーションがアクティブになっており 、それぞれがクラスター・リソースを割り当てていることがあります。

さらに複雑性が増すこととして、特定の状況において、クラスター・リソースを必要とするマップ削減ジョブを Analytic Server が開始する可能性があるということが挙げられます。これらのマップ削減ジョブは、Spark アプリケーションに割り当てられていないリソースを必要とします。マップ削減ジョブを必要とする特定のコンポーネントは、PSM モデル・ビルドです。

Spark アプリケーションにリソースを割り当てるように、以下のプロパティーを構成できます。これらが Spark インストール済み環境の spark-defaults.conf で設定されている場合、その環境で実行されるすべての Spark ジョブに割り当てられます。これらが Analytic Server 構成の「Custom analytic.cfg」セクションでカスタム・プロパティーとして設定されている場合、Analytic Server Spark アプリケーションのみに割り当てられます。

spark.executor.memory

executor プロセスあたりに使用するメモリーの量。

spark.executor.instances

開始する executor プロセスの数。

spark.executor.cores

executor プロセスあたりの executor ワーカー・スレッドの数。この値は、1 から 5 の範囲内でなければなりません。

3 つのキー Spark プロパティーの設定例。HDFS クラスターには、10 個のデータ・ノードがあります。それぞれのデータ・ノードには、24 個の論理コアと 48 GB のメモリーがあります。データ・ノードは HDFS プロセスのみを実行しています。この環境のプロパティーを構成する 1 つの方法を以下に示します。これは、この環境で Analytic Server ジョブのみを実行しており、単一の Analytic Server Spark アプリケーションに最大の割り当てを行うという前提に基づいています。

- spark.executor.instances=20 に設定。これにより、データ・ノードごとに 2 つの Spark executor プロセスの実行が試行されます。
- spark.executor.memory=22G に設定。これにより、各 Spark executor プロセスの最大ヒープ・サイズが 22 GB に設定され、各データ・ノードに 44 GB が割り当てられます。他の JVM およびその OS には、追加のメモリーが必要です。

- spark.executor.cores=5 に設定。これにより、各 Spark executor に 5 個のワーカー・スレッドが提供され、データ・ノードあたり合計 10 個のワーカー・スレッドになります。

ジョブ実行のための Spark UI のモニター

パフォーマンスに影響を及ぼす可能性があるディスクへのスピルが判明した場合: 考えられる解決策には以下があります。

- メモリーを増やし、そのメモリーを **spark.executor.memory** によって Spark executor に割り当てる。
- **spark.executor.cores** の数を減らす。これにより、メモリーを割り当てている同時作業スレッドの数が減りますが、ジョブの並列処理の量も減ります。
- Spark メモリー・プロパティーを変更する。Spark 用の Spark executor ヒープの **spark.shuffle.memoryFraction** 割り振りパーセンテージおよび **spark.storage.memoryFraction** 割り振りパーセンテージ。

名前ノードに十分なメモリーを確保する

HDFS のブロックの数が多く、さらに増え続けている場合は、その増加に対応するように名前ノードのヒープを増やしてください。これは、HDFS 調整の一般的な推奨事項です。

キャッシュに使用するメモリーの量の変更

デフォルトでは、**spark.storage.memoryFraction** には値 0.6 が指定されています。この値は、データの HDFS ブロック・サイズが 64 MB の場合に 0.8 まで増やすことができます。入力データの HDFS ブロック・サイズが 64 MB よりも大きいときは、タスクあたりに割り当てられたメモリーが 2 GB より大きい場合にのみこの値を増やすことができます。

モデル・スコアリングのパフォーマンスの調整

以下のステップを実行することにより、大規模なデータ・セットでの Apache Spark エンジンを使用したモデル・スコアリング・ジョブのパフォーマンスを向上させることができます。これらのステップはクラスター上の非 Analytic Server サービスの処理には影響を与えないことに注意してください。

1. libtcmalloc_minimal.so{/バージョン} がクラスター内の各ノードに既にインストールされていることを確認します。

   ```
   whereis libtcmalloc_minimal.so.*
   ```

2. libtcmalloc_minimal.so がインストールされていない場合は、libtcmalloc_minimal ライブラリーが含まれるオペレーティング・システムに固有のパッケージをクラスター内の各ノードにインストールするか、または libtcmalloc_minimal を手動でビルドおよびインストールしてください。例:

   Ubuntu:

   ```
   sudo apt-get install libgoogle-perftools-dev
   ```

   Red Hat Enterprise Linux 6.x (x64):

   a. RedHat の EPEL リポジトリーをインストールします (まだインストールされていない場合)

      ```
      wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
      sudo rpm -Uvh epel-release-6*.rpm
      ```

   b. `sudo yum install gperftools-libs.x86_64`

   手動ビルド:

   a. gperftools-2.4.tar.gz をリンク (https://github.com/gperftools/gperftools/releases) からダウンロードします。

   b. `tar zxvf gperftools-2.4.tar.gz`

   c. `cd gperftools-2.4`

d. `./configure --disable-cpu-profiler --disable-heap-profiler --disable-heap-checker --disable-debugalloc --enable-minimal`

e. `make`

f. `sudo make install`

3. インストール済みのライブラリー・ファイル libtcmalloc_minimal.so{.バージョン} の場所の 1 つをメモします。この場所は、1 つ以上のノードで以下のコマンドを実行すると返されます。

```
whereis libtcmalloc_minimal.so.*
```

さまざまなオペレーティング・システムを実行しているノードがクラスターに含まれている場合は、このファイルに対して複数の場所が存在する可能性があります。

4. Ambari コンソールで、Analytic Server 構成に移動し、「Custom analytics.cfg」セクションでライブラリーの場所を値として使用してキーの spark.executorEnv.LD_PRELOAD を構成します。この変更を行った後に Analytic Server サービスを再始動してください。例えば、ライブラリーが /usr/lib64/libtcmalloc_minimal.so.4 にインストールされている場合、構成は以下のようになります。

```
spark.executorEnv.LD_PRELOAD=/usr/lib64/libtcmalloc_minimal.so.4
```

複数の場所が必要な場合は、以下の例のようにスペースを使用して区切ってください。

```
spark.executorEnv.LD_PRELOAD=/usr/lib64/libtcmalloc_minimal.so.4 /usr/lib/
libtcmalloc_minimal.so
```

いずれのノードも構成済みの場所の 1 つに `libtcmalloc_minimal.so` ライブラリーがインストールされていない場合、それによってエラーは生じませんが、それらのノードでモデル・スコアリングのパフォーマンスが低下する可能性があります。

Spark の map 側結合

Analytic Server の Spark 結合実装では、map 側結合機能はサポートされません (Spark 結合は主に reduce 側です)。この実装では、ある入力が小さい場合、結合の最適化に map 側結合は利用されません。map 側結合を利用しないと、Spark ジョブは極端にリソースを使用し、最終的に失敗します。

Analytic Server Spark の map 側結合 (または最小 RDD サイズに基づくネイティブの Spark ジョブ) を実行するときに結合を最適化するには、`spark.msj.maxBroadcast` プロパティーを `analytics.cfg` ファイル (`SPSS Analytic Server/Configs/Custom analytics.cfg`) または `analytics-meta` に追加します。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。 この資料は、IBM から他の言語でも提供されている可能性があります。 ただし、これを入手するには、本製品または当該言語版製品を所有している必要がある場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、または サービスのみが使用可能であることを意味するものではありません。 これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の 製品、プログラム、またはサービスを使用することができます。 ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について 実施権を許諾することを意味するものではありません。 実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 *103-8510*
東京都中央区日本橋箱崎町 *19* 番 *21* 号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、 商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示 もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。 本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、 改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムと その他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

本プログラムに関する上記の情報は、適切な使用条件の下で使用すること ができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

記載されている性能データとお客様事例は、例として示す目的でのみ提供されています。 実際の結果は特定の構成や稼働条件によって異なります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、 予告なしに変更または撤回される場合があり、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、 現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。 記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。 より具体性を与えるために、それらの例には、個人、企業、ブランド、 あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、類似する個人や企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。 より具体性を与えるために、それらの例には、個人、企業、ブランド、 あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、類似する個人や企業が実在しているとしても、それは偶然にすぎません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次の ように、著作権表示を入れていただく必要があります。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Adobe、Adobe ロゴ、PostScript、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

IT Infrastructure Library は AXELOS Limited の登録商標です。

インテル、Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Centrino、Intel Centrino ロゴ、Celeron、Xeon、Intel SpeedStep、Itanium、および Pentium は、Intel Corporation または子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。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

Microsoft、Windows、Windows NT 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。

ITIL は AXELOS Limited の登録商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

Cell Broadband Engine は、Sony Computer Entertainment, Inc. の米国およびその他の国における商標であり、同社の許諾を受けて使用しています。

Linear Tape-Open、LTO、LTO ロゴ、Ultrium および Ultrium ロゴは、HP、IBM Corp. および Quantum の米国およびその他の国における商標です。