Oracle® Databaseパフォーマンス・チューニング・ガイド
11*g*リリース2 (11.2)
B56312-05

8 I/O構成および設計

I/Oサブシステムは、Oracleデータベースに不可欠なコンポーネントです。基本的なI/O概念を紹介し、データベースの様々な部分のI/O要件について説明し、I/Oサブシステムの設計のための構成例を示します。

この章には次の項があります。

- I/Oについて
- I/O構成
- データベース内部のI/O測定
- Oracle Orion測定ツールによるI/O測定

8.1 I/Oについて

Oracle Databaseでディスク上のデータを読み書きする際は、必ず**ディスクI/O**が生成されます。多くのソフトウェア・アプリケーションのパフォーマンスは、本質的にディスクI/Oによって制限されます。CPUタイムの大部分をI/Oアクティビティが完了するまでの待機に使用するアプリケーションは**I/Oバウンド**と呼ばれます。

Oracle Databaseは、適切に作成されたアプリケーションのパフォーマンスが、I/Oで制限されないように設計されています。I/Oシステムが最大限またはそれに近い状態で動作しており、許容時間内にI/Oリクエストに対応できない場合は、I/Oのチューニングを行うと、アプリケーションのパフォーマンスを向上できます。ただし、アプリケーションがI/Oバウンドではない場合(たとえば、CPUが制限要因である場合)、I/Oをチューニングしてもパフォーマンスを改善できません。

I/Oシステムを設計するときは、次のデータベース要件を考慮してください。

- ディスクの最小容量などの記憶域
- 継続的(24時間×7日)または営業時間のみなどの可用性
- I/Oスループットやアプリケーション・レスポンス時間などのパフォーマンス

多くのI/O設計では、パフォーマンスが問題とならないと仮定して記憶域要件および可用性要件の計画を立てています。ただし、これに該当しない場合もあります。構成するディスクおよびコントローラの数はI/Oスループットおよび冗長性の要件で判断することが最適です。この場合、ディスクのサイズは記憶域要件で判断できます。

I/O設計計画を作成する場合は、**Oracle Automatic Storage Management(Oracle ASM)**の使用を検討します。Oracle ASMは、記憶域の管理を管理者に任せるのではなく、データベースで管理するという方針に基づいた、高いパフォーマンスを実現する統合データベース・ファイル・システムおよびディスク・マネージャです。

データベース・ファイル記憶域には、RAWデバイスやオペレーティング・システムのファイル・システムではなく、Oracle ASMを使用することをお薦めします。Oracle ASMの主な利点を次に示します。

- ストライプ化
- ミラー化
- オンラインでの記憶域の再構成と動的リバランス
- 管理対象ファイルの作成と削除

> **関連項目:**
> Oracle ASMの詳細は、*『Oracle Automatic Storage Management管理者ガイド』*を参照してください。

8.2 I/O構成

この項では、収集する基本情報およびシステムのI/O構成を定義するときの決定事項について説明します。必要な可用性、リカバリ能力およびパフォーマンスは維持しながら、できるだけ単純な構成を維持します。構成が複雑になるに従って、管理、メンテナンスおよびチューニングが困難になります。

この項では、次の項目について説明します。

- オペレーティング・システムまたはハードウェアのストライプ化を使用したファイルのレイアウト
- 手動によるI/Oの分散
- ファイルを分割する場合
- 3つの構成サンプル
- Oracle Managed Files
- データ・ブロック・サイズの選択

8.2.1 オペレーティング・システムまたはハードウェアのストライプ化を使用したファイルのレイアウト

オペレーティング・システムにLVMソフトウェアまたはハードウェア・ベースのストライプ化がある場合、これらのツールを使用してI/Oを分散できます。LVMまたはハードウェア・ストライプ化を使用するときの決定事項には、**ストライプ深度**および**ストライプ幅**が含まれます。

- ストライプ深度は、ストライプのサイズで、ストライプ単位とも呼ばれます。
- ストライプ幅は、ストライプ深度とストライプ・セットを構成するドライブ数の積です。

これらの値を適切に選択して、システムが必要なスループットを維持できるようにします。Oracleデータベースにおける適切なストライプ深度は、256KBから1MBです。ストライプ深度によって、得られるアプリケーションの利点の種類が異なります。最適なストライプ深度およびストライプ幅は、次の項目により異なります。

- リクエストされたI/Oサイズ
- I/Oリクエストの同時実行性
- 物理ストライプ境界とブロック・サイズ境界との位置合せ
- 提案されたシステムの管理性

8.2.1.1 リクエストされたI/Oサイズ

表8-1に、I/Oサイズの設定で使用できるOracle Databaseおよびオペレーティング・システムのパラメータを示します。

表8-1 Oracle Databaseおよびオペレーティング・システム操作パラメータ

パラメータ	説明
DB_BLOCK_SIZE	単一ブロックI/Oリクエストのサイズ。また、このパラメータをマルチブロック・パラメータと組み合せて使用して、マルチブロックI/Oリクエスト・サイズを決定します。
OSブロック・サイズ	REDOログおよびアーカイブ・ログ操作のI/Oサイズを決定します。
最大OS I/Oサイズ	単一I/Oリクエストのサイズに上限を設けます。
DB_FILE_MULTIBLOCK_READ_COUNT	全表スキャンの最大I/Oサイズは、このパラメータにDB_BLOCK_SIZEを乗算して計算されます。(上限値は、オペレーティング・システムの制限を受けます。)この値が明示的に設定されていない場合(または0に設定されている場合)のデフォルト値は、効率的に実行可能な最大I/Oサイズですが、プラットフォームごとに異なります。
SORT_AREA_SIZE	ソート操作のためのI/Oサイズおよび同時実行性を決定します。
HASH_AREA_SIZE	ハッシュ操作のためのI/Oサイズを決定します。

I/Oサイズの他に、同時実行性の程度も理想的なストライプ深度を調べる上で役立ちます。ストライプ幅およびストライプ深度を選択する場合は、次の点を考慮してください。

- 同時実行性の低い(順次)システムでは、単一I/Oが同じディスクに2回アクセスしないようにします。たとえば、ストライプ幅が4つのディスクで、ストライプ深度が32KBであると仮定します。1MBの単一I/Oリクエスト(たとえば、全表スキャンの場合)がOracleサーバー・プロセスで発行された場合、ストライプ内の各ディスクは要求されたデータを戻すために8回I/Oを実行する必要があります。このような状況を回避するために、平均I/Oのサイズは、ストライプ深度とストライプ幅の積より小さいサイズにしてください。そうでない場合は、オペレーティング・システムに対してOracle Databaseが単一I/Oリクエストを行うと、同じディスクに対して複数の物理I/Oリクエストが発生します。
- 同時実行性が高い(ランダム)システムでは、単一I/Oリクエストが複数の物理I/Oコールに分解されないようにしてください。そうでなければ、システムで実行される物理I/Oリクエスト数が何倍にもなり、I/Oレスポンス時間が大幅に下がります。

8.2.1.2 I/Oリクエストの同時実行性

従来のOLTP環境などの高度な小さい同時I/Oリクエストが存在するシステムでは、ストライプ深度を大きく保つことが有効です。I/Oサイズより大きいストライプ深度を使用することは**粗密なストライプ化**と呼ばれます。同時実行性の高いシステムのストライプ深度は次のようになります(n > 1)。

```
n * DB_BLOCK_SIZE
```

粗密なストライプ化ではアレイ内の1ディスクで複数のI/Oリクエストに対応できます。この方法では、一連のストライプ化ディスクで多数の同時I/Oリクエストに対応でき、I/Oセットアップ・コストも最小限で済みます。粗密なストライプ化は、全体的なI/Oスループットの最大化を目指します。ストライプ深度が大きく、1つのドライブからサービスできる場合、全表スキャンによるマルチブロック読取りにメリットをもたらします。データ・ウェアハウス環境のパラレル問合せも、粗密なストライプ化の候補です。これは、それぞれ個別のI/Oを発行する個々のプロセスが多数存在するためです。粗密なストライプ化は、同時リクエストが少ないシステムで使用されると、ホット・スポットが生成される可能性があります。

従来のDSS環境や同時実行性の低いOLTPシステムなどの大きいI/Oリクエストが少ないシステムでは、ストライプ深度を小さく保つことが有益です。これは、**ファイングレイン・ストライプ化**と呼ばれます。このようなシステムでは、ストライプ深度は次のようになります。nはマルチブロック読取りパラメータ(DB_FILE_MULTIBLOCK_READ_COUNT)よりも小さくなります。

```
n * DB_BLOCK_SIZE
```

ファイングレイン・ストライプ化では、複数のディスクで単一I/Oリクエストを処理できます。ファイングレイン・ストライプ化では、個々のI/Oリクエストまたはレスポンス時間のパフォーマンスが最大化されます。

8.2.1.3 物理ストライプ境界とブロック・サイズ境界との位置合せ

一部のOracle Databaseポートでは、データベース・ブロック境界がストライプと合わない可能性があります。ストライプ深度がデータベースのブロック・サイズと同じである場合、Oracle Databaseから発行される1つのI/Oによって、2つの物理I/O操作が生じる場合があります。

これは、OLTP環境では最適ではありません。1つの論理I/Oで物理I/Oが1つのみ発生する確率が高くなるようにするには、最小のストライプ深度がOracleブロック・サイズの少なくとも2倍である必要があります。表8-2に、ランダム・アクセスおよび順次読取りでお薦めする最小ストライプ深度を示します。

表8-2 最小ストライプ深度

ディスク・アクセス	最小ストライプ深度
ランダム読取りおよび書込み	最小ストライプ深度は、Oracleブロック・サイズの2倍です。
順次読取り	最小ストライプ深度は、OracleブロックサイズをDB_FILE_MULTIBLOCK_READ_COUNTの値の2倍です。

関連項目:
プラットフォームの固有のマニュアル

8.2.1.4 提案されたシステムの管理性

LVMの場合、最も管理の簡単な構成は、使用可能なすべてのディスク上に単一ストライプ化ボリュームを構成したものです。この場合、ストライプ幅はすべての使用可能なディスクを包含します。すべてのデータベース・ファイルはそのボリューム内に常駐し、効果的に負荷を均等分散します。この単一ボリューム・レイアウトは、ほとんどの状況で適切なパフォーマンスを実現します。

単一ボリューム構成は、RAID 1などの簡単なリカバリを可能にするRAID技術と併用する場合のみ有効です。それ以外の場合、単一のディスクを失うことはすべてのファイルを同時に失うことであり、完全なデータベースのリストアおよびリカバリを実行する必要があることを意味します。

パフォーマンスの他に、管理性の問題があります。システムの設計で、ディスクを簡単に追加できるようにして、データベースの拡張を可能にする必要があります。課題は、負荷のバランスを均一に維持しながら拡張を行うことです。

たとえば、初期構成で、64個の16GBのディスク上に単一ストライプ化ボリュームを作成する必要があるとします。これはプライマリ・データの1TBの合計ディスク領域になります。場合によっては、システムが動作した後に、将来のデータベース拡張を可能にするためにさらに80GB(すなわち、5つのディスク)を追加する必要があります。

この領域をデータベースで使用できるようにするオプションには、5つの新しいディスクを含む第2のボリュームの作成があります。ただし、これらの新しいディスクがその上に配置されたファイルに必要なI/Oスループットを保持できない場合、I/Oボトルネックが発生する可能性があります。

もう1つのオプションは、元のボリュームのサイズを増やすことです。LVMは高度になりつつあり、ストライプ幅の動的な再構成を可能にするので、システムがオンライン中にディスクを追加できます。このため、本番環境で単一ストライプ化ボリューム上にすべてのファイルを配置できるようになってきました。

LVMがストライプへのディスクの動的な追加をサポートできない場合は、より小さく管理しやすいストライプ幅を選択する必要があります。そのようにすると、新しいディスクを追加する場合、ストライプ幅だけシステムを拡張できます。

前述の例で、8個のディスクがより管理しやすいストライプ幅といえます。これは、8個のディスクで1秒間に必要な数のI/Oを維持できる場合のみ可能です。したがって、追加のディスク領域が必要なときは、別の8ディスク・ストライプを追加して、ボリューム間でI/Oのバランスを維持できます。

注意:
ストライプ幅が小さくなるほど、ボリューム上にファイルを分散する時間の必要性が高くなり、このプロシージャは手動によるI/Oの分散に近づきます。

8.2.2 手動によるI/Oの分散

システムにLVMまたはハードウェアのストライプ化がない場合、各ファイルのI/O要件に従ってファイルを分散することにより、使用可能なディスク間でI/Oを手動でバランス化する必要があります。ファイルの配置に関する決定を行うには、データベース・ファイルのI/O要件およびI/Oシステムの機能についてよく理解している必要があります。このようなデータに慣れておらず、解析対象の代表的なワークロードを取得できない場合は、まず推定を行い、次に使用量がわかったときにこのレイアウトをチューニングします。

ディスクを手動でストライプ化するには、ファイルの記憶域要件をI/O要件と関連付ける必要があります。

1. ファイルおよびディスクのサイズをチェックして、データベースのディスク記憶域要件を評価します。
2. 1ファイル当たりの予測I/Oスループットを識別します。最高のI/O率を持つファイルおよび多数のI/Oを持たないファイルを判断します。I/O率を均等にするために、すべての使用可能なディスク上にファイルをレイアウトします。

手動I/O分散の一般的なアプローチとして、頻繁に使用される表をその索引から分離することがあげられます。これは正しくありません。一連のトランザクション中は、索引が読み取られてから表が読み取られます。これらのI/Oは順次に発生するので、表と索引を同じディスク上に格納しても競合は発生しません。データファイルには索引や表データが含まれているため、これを単純に分離するだけでは十分ではありません。ファイルを分離する決定は、そのファイルのI/O率がデータベースのパフォーマンスに影響を与える場合にのみ行ってください。

8.2.3 ファイルを分割する場合

オペレーティング・システムのストライプ化または手動I/O分散を使用するかどうかに関係なく、I/OシステムまたはI/Oレイアウトが要求されたI/O率をサポートできない場合は、I/O率の高いファイルをそれ以外のファイルから分離する必要があります。このようなファイルは計画段階かシステムの本稼働後に確認できます。

ファイルを分離する決定は、I/O率、リカバリ能力の問題、管理性の問題によってのみ影響を受けます。(たとえば、LVMがストライプ幅の動的な再構成をサポートしない場合、同一構成の新しいストライプを作成するために一度に*n*個のディスクを追加できるように、さらに小さいストライプ幅を作成する必要がある場合があります。)

ファイルを分離する前に、ボトルネックが実際にI/Oの問題であるかどうかを検証します。ボトルネックの調査から生成されたデータでは、最高のI/O率を持つファイルを識別します。

後続の項では、次のファイル・タイプを分離する方法について説明します。

- 表、索引およびTEMP表領域
- REDOログ・ファイル
- アーカイブREDOログ

関連項目:
「高負荷SQLの識別」

8.2.3.1 表、索引およびTEMP表領域

表および索引を含む表領域に属するデータファイルがI/Oの多いファイルである場合は、これらのファイルのI/OをSQLのチューニングまたはアプリケーション・コードのいずれかで削減できるかどうかを識別します。

I/Oの多いファイルがTEMP表領域に属するデータファイルである場合は、このアクティビティを回避するためにディスク・ソートを実行するSQL文をチューニングするか、あるいはソートをチューニングするかを調べます。

不要なI/Oを回避するようにアプリケーションをチューニングした後、I/Oレイアウトが引き続き必要なスループットを維持できない場合は、I/Oの多いファイルの分離を考慮してください。

関連項目:
「高負荷SQLの識別」

8.2.3.2 REDOログ・ファイル

I/Oの多いファイルがREDOログ・ファイルである場合は、REDOログ・ファイルをその他のファイルから分離することを考慮してください。可能な構成には、次のことが含まれています。

- すべてのREDOログを、他のファイルのない1つのディスクに置きます。可用性も考慮します。すなわち、リカバリ可能性の目的で、同じグループのメンバーは異なる物理ディスクおよ

びコントローラ上にあることが必要です。

- 他のファイルを格納しない個別のディスク上に各REDOログ・グループを置きます。
- オペレーティング・システムのストライプ化ツールを使用して、複数のディスクにまたがってREDOログ・ファイルをストライプ化します。(この場合、手動ストライプ化は不可能です。)
- REDOログにRAID 5を使用しないでください。

REDOログ・ファイルは、ログ・ライター(LGWR)プロセスで順次書き込まれます。同じディスクに対する同時実行アクティビティが存在しない場合、この操作はさらに高速にできます。REDOログ・ファイルに別々の専用ディスクを割り当てると、さらにチューニングしなくても通常はLGWRが円滑に実行されます。システムが非同期I/Oをサポートせず、この機能が現在構成されていない場合、この機能を使用することが有効かどうかを確認します。LGWRに関連するパフォーマンス上のボトルネックはめったにありません。

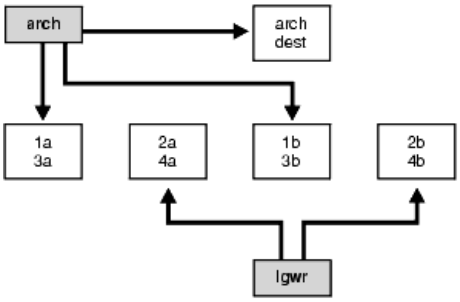
8.2.3.3 アーカイブREDOログ

アーカイバが遅い場合、アーカイバ読取りとLGWR書込みが分離されるようにして、アーカイバ・プロセスとLGWRの間のI/O競合を防止することが賢明です。これは、ログを代替ドライブに置くことで達成されます。

たとえば、システムに4つのREDOログ・グループが存在し、各グループが2つのメンバーを持っているとします。個別ディスク・アクセスを作成するには、8つのログ・ファイルにそれぞれ1a、1b、2a、2b、3a、3b、4a、4bのラベルを付けてください。この場合、最低でも4つのディスクと、アーカイブ・ファイル用に1つのディスクが必要です。

図8-1は、競合を最小限にするために、ディスク間でREDOメンバーを分散する方法を示しています。

図8-1 ディスク間でのREDOメンバーの分散



「図8-1 ディスク間でのREDOメンバーの分散」の説明

この例では、LGWRはログ・グループ1(メンバー1aと1b)から切り替えられ、ログ・グループ2(2aと2b)に書込みを行います。同時に、アーカイバ・プロセスはグループ1から読取りをして、アーカイブ先に書込みを行います。REDOログ・ファイルがどのようにして競合から分離されているかに注意してください。

> **注意:**
> REDOログ・ファイルをミラー化する、すなわち各REDOログ・ファイルの複数のコピーを別々のディスク上に保持することで、LGWRが大幅に遅くはなりません。LGWRは、各ディスクに対して並列して書込みを行い、並列書込みの各部が完了するまで待機します。したがって、並列書込みが、最も長い単一のディスク書込みよりも長くなることはありません。

REDOログはシリアルに書き込まれるので、REDOログ・アクティビティ専用のドライブでは一般にヘッドの移動はわずかです。このため、ログ書込みのスピードが大幅に向上します。

8.2.4 3つの構成サンプル

この項では、I/Oシステムを構成する高水準の例を3つ示します。これらの例には、ディスク・トポロジやストライプ深度などを定義する計算例が含まれています。

- すべてのディスクにまたがったすべての内容のストライプ化
- 異なるディスクへのアーカイブ・ログの移動
- 個別のディスクへのREDOログの移動

8.2.4.1 すべてのディスクにまたがったすべての内容のストライプ化

I/O構成の最も簡単なアプローチは、すべての使用可能ディスクにまたがってストライプ化された、1つの大きなボリュームを作成することです。リカバリ能力を考慮して、ボリュームがミラー化されます(RAID 1)。ディスク当たりのストライプ化の単位は、頻繁なI/O操作のための最大I/Oサイズより大きくする必要があります。そうすると、ほとんどの場合は十分なパフォーマンスが得られます。

8.2.4.2 異なるディスクへのアーカイブ・ログの移動

アーカイブREDOログ・ファイルが他のファイルと同じディスク・セット上でストライプ化されている場合、REDOログがアーカイブされる際、これらのディスク上のI/Oリクエストが影響を受ける可能性があります。個別のディスクにアーカイブREDOログ・ファイルを移動する場合の利点は次のとおりです。

- 非常に高速なアーカイブを実行できます(順次I/Oを使用)。
- アーカイブ先のディスク上でレスポンス時間が低下しても、その影響を受けるものは他にありません。

アーカイブ・ログ用のディスク数は、アーカイブ・ログ生成頻度およびアーカイブ記憶域の必要量により決定されます。

8.2.4.3 個別のディスクへのREDOログの移動

更新の多いOLTPシステムでは、REDOログは書込み中心です。他のディスクおよびアーカイブREDOログ・ファイルから分離されたディスクにREDOログ・ファイルを移動すると、次の利点があります。

- REDOログの書込みは、可能なかぎり高速で行われます。したがって、トランザクション処理のパフォーマンスは最高になります。
- REDOログの書込みが他のI/Oで損われることはありません。

REDOログ用のディスク数は、ほとんどの場合に、現行技術のディスク・サイズと比較して一般に小さいREDOログ・サイズで決定されます。一般に、2つのディスクを持つ構成(フォルト・トレランスのために4つのディスクにミラー化など)で十分です。特に、2つのディスク上でREDOログ・ファイルを交互に使用すると、1つのファイルにREDOログ情報を書き込む場合、アーカイブの完了したREDOログの読取りが妨げられません。

8.2.5 Oracle Managed Files

ファイル・システムにすべてのOracle Databaseデータを取り込むことができる場合、データベース管理は**Oracle Managed Files**を使用して簡単に行えます。表領域、一時ファイル、オンライン・ログおよび制御ファイルについては、Oracle Databaseは内部的に標準ファイル・システム・インタフェースを使用して、必要に応じてファイルを作成および削除します。管理者は、特定のタイプのファイルに使用するファイル・システム・ディレクトリのみを指定します。データファイルについてはデフォルトの場所を1つ、制御ファイルおよびオンラインREDOログ・ファイルについては複数の場所を最大5つ指定できます。

Oracle Databaseでは、一意のファイルが作成された後、そのファイルが不要になると削除されます。このため、管理者が誤ったファイルを指定したことにより発生する破損、および廃止されたファイルで消費される無駄なディスク領域が減り、テスト・データベースおよび開発データベースの作成が簡素化されます。また、オペレーティング・システム固有のファイル名をSQLスクリプトに設定する必要がないため、ポータブルなサード・パーティのツールの開発を容易にします。

新しいファイルはOracle Managed Filesとして作成できますが、古いファイルは古い方法で管理されます。したがって、データベースにはOracle Managed Filesとユーザー管理ファイルの両方を配置できます。

> **注意:**
> Oracle Managed Filesは、RAWデバイスでは使用できません。

Oracle Managed Filesをチューニングする場合はいくつかの点に注意する必要があります。

- Oracle Managed Filesではファイル・システムを使用する必要があるため、DBAはデータのレイアウト方法は管理しません。したがって、ファイル・システムを正しく構成することが重要です。
- Oracle Managed Filesのファイル・システムは、ストライプ化をサポートするLVM上に構築します。ロード・バランシングおよびスループットの向上のために、ファイル・システム内のディスクをストライプ化します。
- Oracle Managed Filesは、動的に拡張可能な論理ボリュームをサポートするLVM上で使用するときに最高の機能を発揮します。それ以外の場合は、論理ボリュームをできるだけ大きく構成します。
- Oracle Managed Filesは、ファイル・システムに大きな拡張可能なファイルがある場合、最高の機能を発揮します。

> **関連項目:**
> Oracle Managed Filesの使用方法の詳細は、*『Oracle Database管理者ガイド』*を参照してください。

8.2.6 データ・ブロック・サイズの選択

8KBのブロック・サイズはほとんどのシステムにとって最適です。ただし、OLTPシステムではより小さなブロック・サイズを、DSSシステムではより大きなブロック・サイズを使用することがあります。この項では、最適なパフォーマンスを得るためにデータベース・ブロック・サイズを選択するときの考慮事項を説明します。内容は次のとおりです。

- 読取り
- 書込み
- ブロック・サイズの長所と短所

> **注意:**
> 管理性の問題があるため、単一データベース・インスタンスでの複数のブロック・サイズの使用はお薦めしません。

8.2.6.1 読取り

データのサイズとは関係なく、目標は必要なデータを取り出すために必要な読取り回数を最小にすることです。

- 行が小さく、アクセスがきわめてランダムな場合は、小さなブロック・サイズを選択します。
- 行が小さく、アクセスがきわめて順次である場合は、大きなブロック・サイズを選択します。
- 行が小さく、アクセスがランダムかつ順次である場合は、大きなブロック・サイズを選択するのが有効です。
- 行が大きい(たとえば、ラージ・オブジェクト(LOB)データが含まれている)場合は、大きなブロック・サイズを選択します。

8.2.6.2 書込み

同時実行性の高いOLTPシステムで、大きなブロック・サイズを使用する場合は、`INITRANS`、`MAXTRANS`および`FREELISTS`の適切な値について検討します。これらのパラメータは、ブロック内で許可されている更新の同時実行性の程度に影響を与えます。ただし、自動セグメント領域管理を使用する場合は、`FREELISTS`の値を指定する必要はありません。

選択する必要があるブロック・サイズが不明な場合、多数のトランザクションを処理するシステムでは、8KBのデータベース・ブロック・サイズを試行してください。これは優れた妥協案であり、通常は有効です。8KBを超えるサイズが必要なのはLOBデータを処理するシステムのみです。

> **関連項目:**
> 使用しているプラットフォームの最小および最大のブロック・サイズは、オペレーティング・システム固有のOracle Databaseインストール・マニュアルを参照してください。

8.2.6.3 ブロック・サイズの長所と短所

表8-3に、様々なブロック・サイズの長所と短所を示します。

表8-3 ブロック・サイズの長所と短所

ブロック・サイズ	長所	短所
小さい	行数が少なく大量のランダム・アクセスに適しています。 ブロック競合が低減されます。	メタデータ(すなわち、ブロック・ヘッダー)による比較的大きな領域のオーバーヘッドがあります。 行が大きい場合にはお薦めできません。1行がブロックに収まらない場合、1ブロック当たりの格納行数がわずかになったり、さらには行の連鎖が発生する可能性があります。
大きい	オーバーヘッドが少ないため、データを格納する空間が多くなります。 1回のI/Oでバッファ・キャッシュに複数の行を読み取れます(行サイズおよびブロック・サイズにより異なります)。 順次アクセスまたは非常に大きな行(LOBデータなど)に適しています。	ブロック・サイズが大きい場合に少数の行にランダム・アクセスすると、バッファ・キャッシュ内の領域が消費されます。たとえば、8KBのブロック・サイズと50バイトの行サイズでは、ランダム・アクセスを行うときにバッファ・キャッシュ内の7,950バイトが消費されます。 OLTP環境で使用される索引ブロックには適しません。これは、索引リーフ・ブロック上のブロック競合が増えるためです。

8.3 データベース内部のI/O測定

Oracle DatabaseのI/O測定機能では、ストレージ・サブシステムのパフォーマンスを評価し、I/Oパフォーマンス問題がデータベースとストレージ・サブシステムのどちらを原因とするかを判別できます。I/Oを連続的に発行する外部のI/O測定ツールとは異なり、Oracle DatabaseのI/O測定機能では、Oracleデータファイルを使用してI/Oをランダムに発行し、ストレージ・メディアにアクセスするため、よりデータベースの実際のパフォーマンスに近い結果を得ることができます。

この項では、Oracle DatabaseのI/O測定機能の使用方法について説明します。この項には、次の項目があります。

- I/O測定の前提条件
- I/O測定の実行

Oracle Databaseは、OrionによるI/O測定も提供しています。Orionは、Oracleのインストールやデータベースの作成を行わなくてもOracle databaseのパフォーマンスを予測できるツールです。他のI/O測定ツールとは異なり、Oracle Orionは、Oracleと同じI/Oソフトウェア・スタックを使用してOracleデータベースのI/Oワークロードをシミュレートするように特別に設計されたものです。またOrionは、Oracle Automatic Storage Managementによって実行されるストライプ化の効果もシミュレートすることができます。詳細は、「Oracle Orion測定ツールによるI/O測定」を参照してください。

8.3.1 I/O測定の前提条件

I/O測定を実行する前に、次の前提条件が満たされていることを確認してください。

- SYSDBA権限をユーザーに付与する必要があります。
- timed_statisticsがTRUEに設定されている必要があります。
- 非同期I/Oが有効化されている必要があります。

  ファイル・システムを使用している場合、初期化パラメータFILESYSTEMIO_OPTIONSをSETALLに設定することで非同期I/Oを有効化できます。
- 次の問合せを実行し、データファイルに対して非同期I/Oが有効化されていることを確認してください。

```
COL NAME FORMAT A50
SELECT NAME,ASYNCH_IO FROM V$DATAFILE F,V$IOSTAT_FILE I
WHERE  F.FILE#=I.FILE_NO
AND    FILETYPE_NAME='Data File';
```

また、1つのデータベース・インスタンスに一度に実行できる測定は、1回のみです。

8.3.2 I/O測定の実行

Oracle DatabaseのI/O測定機能には、DBMS_RESOURCE_MANAGER.CALIBRATE_IOプロシージャを使用してアクセスします。このプロシージャは、データベース・ファイルにI/O集中型の読取り専用ワークロード(1MBのランダムI/Oで構成されるワークロード)を発行し、ストレージ・サブシステムが耐えられる最大IOPS(1秒当たりのI/Oリクエスト)とMBPS(1秒当たりのI/OのMB)を判定します。

I/O測定は、次の2つの手順で行われます。

- I/O測定の最初の手順では、DBMS_RESOURCE_MANAGER.CALIBRATE_IOプロシージャにより、すべてのデータベース・インスタンスのすべてのデータファイルに対してデータベース・ブロック・サイズのランダムな読取り(デフォルトは8KB)を発行します。この手順では、出力パラメータmax_iopsに、データベースが維持できる最大IOPSが示されます。max_iopsの値は、OLTPデータベースに重要なメトリックです。出力パラメータactual_latencyには、このワークロードの平均待機時間が示されます。特定の目標待機時間が必要な場合は、入力パラメータmax_latencyを使用して目標待機時間を指定できます(データベース・ブロック・サイズのIOリクエストの最大許容待機時間をミリ秒単位で指定)。
- 2番目の測定の手順では、DBMS_RESOURCE_MANAGER.CALIBRATE_IOプロシージャにより、すべてのデータベース・インスタンスのすべてのデータファイルに対して1MBのランダムな読取りを発行します。2番目の手順では、出力パラメータmax_mbpsに、データベースが維持できるI/Oの最大MBPSが示されます。この手順は、データ・ウェアハウスに重要なメトリックを提供します。

ユーザーが入力パラメータnum_physical_disksを指定すると(データベース・ストレージ・システム内の物理ディスクの概数を指定)、より効率的に測定を実行できます。

I/Oワークロードの実行によりオーバーヘッドが発生するため、I/O測定はデータベースがアイドル状態のとき(つまりピーク時以外の時間)にのみ実行し、通常のデータベース・ワークロードに対するI/Oワークロードの影響を最小化する必要があります。

I/O測定を実行し、Oracle Databaseで使用されるストレージ・サブシステムのI/O性能を評価するには、次のようにDBMS_RESOURCE_MANAGER.CALIBRATE_IOプロシージャを使用します。

```
SET SERVEROUTPUT ON
DECLARE
  lat  INTEGER;
  iops INTEGER;
  mbps INTEGER;
```

```
BEGIN
-- DBMS_RESOURCE_MANAGER.CALIBRATE_IO (<DISKS>, <MAX_LATENCY>, iops, mbps, lat);
   DBMS_RESOURCE_MANAGER.CALIBRATE_IO (2, 10, iops, mbps, lat);

  DBMS_OUTPUT.PUT_LINE ('max_iops = ' || iops);
  DBMS_OUTPUT.PUT_LINE ('latency  = ' || lat);
  dbms_output.put_line('max_mbps = ' || mbps);
end;
/
```

DBMS_RESOURCE_MANAGER.CALIBRATE_IOプロシージャを実行する場合、次のことを考慮してください。

- 同じストレージ・サブシステムを使用するデータベースで一度に実行できる測定は1回のみです。同じストレージ・サブシステムを使用する別のデータベースで同時に測定を実行すると、測定は失敗します。
- インスタンスでのI/Oを最小化するため、データベースを停止します。
- Oracle Real Application Clusters(Oracle RAC)構成の場合、ノード間でストレージ・サブシステムを測定できるようすべてのインスタンスがオープン状態にあることを確認します。
- Oracle Real Application Clusters(Oracle RAC)データベースでは、ワークロードはすべてのインスタンスから同時に生成されます。
- 入力パラメータnum_physical_disksはオプションです。num_physical_disksパラメータにデータベースのストレージ・システム内にある物理ディスクの概数を設定すると、より高速かつ正確に測定できます。
- 場合によっては、非同期I/Oがデータファイルで許可されている一方で、非同期I/Oを発行するI/Oサブシステムが最大限度に達し、I/O測定を継続できなくなります。このような場合、ポート固有のドキュメントを参照して、システムの非同期I/Oの最大限度をチェックしてください。

I/O測定プロセス中は、いつでもV$IO_CALIBRATION_STATUSビューで測定ステータスを問い合せることができます。I/O測定が正常に完了したら、DBA_RSRC_IO_CALIBRATE表でその結果を参照できます。

関連項目:

- DBMS_RESOURCE_MANAGER.CALIBRATE_IOプロシージャの実行方法の詳細は、*『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』*を参照してください。
- V$IO_CALIBRATION_STATUSビューおよびDBA_RSRC_IO_CALIBRATE表の詳細は、*『Oracle Databaseリファレンス』*を参照してください。

8.4 Oracle Orion測定ツールによるI/O測定

この項では、Oracle Orion測定ツールについて説明します。この項には次の項目があります。

- Oracle Orion測定ツールの概要
- Orionの開始
- Orionの入力ファイル
- Orionのパラメータ
- Orionの出力ファイル
- Orionのトラブルシューティング

8.4.1 Oracle Orion測定ツールの概要

Oracle Orionは、Oracleのインストールやデータベースの作成を行わなくてもOracleデータベースのパフォーマンスを予測できるツールです。他のI/O測定ツールとは異なり、Oracle Orionは、Oracleと同じI/Oソフトウェア・スタックを使用してOracleデータベースのI/Oワークロードをシミュレートするように特別に設計されたものです。またOrionは、Oracle Automatic Storage Managementによって実行されるストライプ化の効果もシミュレートすることができます。

表8-4に、OrionがサポートするI/Oワークロードのタイプを示します。

Orionは、表8-4に示すワークロードの各タイプについて、様々なI/O負荷を使用してテストを実行し、MBPS、IOPS、およびI/O待機時間などのパフォーマンス・メトリックを測定できます。負荷は、未処理の非同期I/Oの数で表されます。内部的には、Orionソフトウェアはこの各負荷レベルに対して、できるだけ高速にI/Oリクエストを発行し続けてI/O負荷をそのレベルに保ちます。ランダム・ワークロードでは、大規模I/Oと小規模I/Oのどちらを使用する場合でも、負荷レベルは未処理I/Oの数です。大規模な順次ワークロードでは、負荷レベルは順次ストリームの数とストリーム当たりの未処理I/Oの数の組合せです。指定されたワークロードを一定範囲の負荷レベルでテストすると、負荷がパフォーマンスに与える影響を理解するのに役立ちます。

Orionを使用する際には、次の点に注意してください。

- Orionはストレージがアイドル状態(またはほぼアイドル状態)のときに実行してください。Orionは、生成したI/O負荷に基づいてストレージのパフォーマンスを測定するため、Orion以外のI/Oワークロードが同時に実行されていると、パフォーマンスを適切に評価できません。
- ストレージにすでにデータベースが作成されている場合は、別の方法としてPL/SQLルーチンdbms_resource_manager.calibrate_io()を使用してストレージを測定できます。

表8-4 Orionがサポートする I/Oワークロード

ワークロード	説明
小規模なランダムI/O	OLTPアプリケーションは通常、データベースのブロック・サイズと同じサイズ(通常は8KB)のランダム読取りおよびランダム書込みを生成します。このようなアプリケーションでは通常、1秒当たりのI/O(IOPS)で表すスループットと1リクエスト当たりの平均待機時間(I/O応答時間)が重要になります。これらのパラメータは、アプリケーション・レイヤーでトランザクション速度とトランザクション応答時間に変換されます。 Orionは、指定された読取りの書込みに対する割合、指定されたI/Oサイズ、そして指定された未処理I/Oの数を使用して、ランダムI/Oワークロードをシミュレートします。このOrionのワークロード・シミュレーションでは、I/Oはすべてのディスクに配分されます。

大規模な順次I/O	データ・ウェアハウス・アプリケーション、データのロード、バックアップ、およびリストアでは、複数の1MBの未処理I/Oからなる順次読取りおよび順次書込みのストリームが生成されます。このようなアプリケーションは、表全体やデータベース全体など、大量のデータを処理し、通常、1秒当たりのMB(MBPS)で表すデータ全体のスループットが重要になります。 Orionは、指定された数の順次読取りまたは順次書込みストリームを、指定されたI/Oサイズと指定された未処理I/O数でシミュレートできます。Orionは、順次ストリームのテスト時に、オプションでOracle Automatic Storage Managementのストライプ化をシミュレートできます。
大規模なランダムI/O	順次ストリームは通常、他のデータベース通信と同時にディスクにアクセスします。ストライプ化により、順次ストリームは多数のディスクに分散されます。結果として、ディスク・レベルでは、複数の順次ストリームが1MBのランダムI/Oとなります。
混合ワークロード	Orionは、2つの同時ワークロード(小規模なランダムI/Oと、大規模な順次I/Oか大規模なランダムI/Oのどちらか)をシミュレートできます。このワークロード・タイプにより、たとえば8KBのランダム読取りおよびランダム書込みのOLTPワークロードと、4つの1 MB I/Oの順次読取りストリームのバックアップ・ワークロードを一緒にシミュレートできます。

Orionの各データ・ポイントは、一定期間持続する小規模I/O負荷と大規模I/O負荷の特定の混合に対するテストです。Orionテストは複数のデータ・ポイント・テストからなります。これらのデータ・ポイント・テストは、2次元マトリックスとして表すことができます。マトリックスの各列は、同じ小規模I/O負荷によるデータ・ポイント・テストを表しますが、大規模I/O負荷は異なります。各行は、同じ大規模I/O負荷によるデータ・ポイント・テストを表しますが、小規模I/O負荷は異なります。Orionテストは、1つのポイント、1つの行、1つの列に対して、またはマトリックス全体に対して行うことができます。

8.4.1.1 Orionテストのターゲット

Orionを使用して、非同期I/Oをサポートするディスクベースのキャラクタ・デバイスをテストできます。Orionは次のターゲット・タイプについて検証済です。

- DAS(直接接続型)ストレージ: Orionを使用して、1つ以上のローカル・ディスク、ボリューム、またはローカル・ホスト上のファイルのパフォーマンスをテストできます。
- SAN(ストレージ・エリア・ネットワーク): Orionは、そのすべてまたは一部のSANストレージがキャラクタ・デバイスとしてマップされている任意のホストで実行できます。デバイスは、ストレージ・アレイによってエクスポートされたストライプ・ボリュームまたは非ストライプ・ボリューム、個々のディスク、または1つ以上のアレイ全体に相当します。
- NAS(ネットワーク接続ストレージ): Orionを使用して、NASストレージ上のデータファイルのパフォーマンスをテストできます。一般的には、NASストレージでのパフォーマンス結果は、データファイルの作成および更新に使用されたI/Oパターンによって異なります。そのため、Orionを実行する前にデータファイルを適切に初期化する必要があります。

8.4.1.2 Oracle管理者のためのOrion

Oracle管理者は、Orionを使用して、予測されるワークロードに基づいて、様々なストレージ・アレイを評価および比較することができます。またOracle管理者は、Orionを使用して、予測されるピーク・ワークロードに対して、ネットワーク接続、ストレージ・アレイ、ストレージ・アレイ・コントローラ、およびディスクの最適な数を判断できます。

8.4.2 Orionの開始

Orionの使用を開始するには、次のようにします。

1. Orion `-testname`パラメータにより、使用するテスト名を選択します。このパラメータにより、Orionの実行に対して固有の識別子が指定されます。たとえば、テスト名「mytest」を使用します。詳細は、「Orionのパラメータ」を参照してください。
2. テスト名に基づいて、Orion入力ファイルを作成します。たとえば、`mytest.lun`という名前のファイルを作成します。入力ファイルに、テストするRAWボリュームまたはファイルをリストします。1行ごとに1つのボリューム名を追加します。`.lun`ファイルにはコメントなど他のものは入力しないでください。

   たとえば、Orion入力ファイルには次を含めることができます。

   ```
   /dev/raw/raw1
   /dev/raw/raw2
   /dev/raw/raw3
   /dev/raw/raw4
   /dev/raw/raw5
   /dev/raw/raw6
   /dev/raw/raw7
   /dev/raw/raw8
   ```

   詳細は、「Orionの入力ファイル」を参照してください。
3. 入力ファイルで指定されたすべてのボリューム、たとえばmytest.lunが、コマンド`dd`または他の同等のファイル表示ユーティリティを使用してアクセスできることを確認します。たとえば、一般的な健全性チェックでは、Linuxシステムで次のことを試してください。

   ```
   $ dd if=/dev/raw/raw1 of=/dev/null bs=32k count=1024
   ```

   プラットフォームによって、使用するファイル表示ユーティリティとそのインタフェースは異なる場合があります。
4. プラットフォームに非同期I/Oに必要なライブラリがインストールされていることを確認します。Orionテストは非同期I/Oに完全に依存しています。LinuxおよびSolarisでは、ライブラリ`libaio`が標準`lib`ディレクトリにあるか、またはシェル環境のライブラリパス変数(通常はシェルに応じて`LD_LIBRARY_PATH`または`LIBPATH`)を使用してこのライブラリにアクセスできることが必要です。Windowsには組込みの非同期I/Oライブラリがあるため、この問題は適用されません。
5. Orionでの最初のテストとして、`-run`に`oltp`または`dss`のどちらかのオプションを付けて使用します。データベースが主としてOLTPの場合は、`-run oltp`を使用します。データベースが主としてデータ・ウェアハウスまたは分析用の場合は、`-run dss`を使用します。

   たとえば、デフォルトの入力ファイル名、`orion.lun`を使用してOLTPのようなワークロードを実行するには、次のコマンドを使用します。

   ```
   $ ./orion -run oltp
   ```

   Orionは、テストされるディスク・スピンドルの数(または`-num_disks`パラメータで指定された数)を考慮してI/O負荷レベルを生成します。スピンドルの数が入力ファイルで指定されたボリュームの数に*関係するかどうか*は、これらのボリュームがどのようにマップされているかによって異なることを覚えておいてください。
6. 「Orionの出力ファイル」の項に、Orionの出力ファイルを示す結果例を記載しています。手始めとしてサンプル・ファイルmytest_summary.txtを使用すると、入力パラメータの検証や出力の分析に役立ちます。サンプル・ファイルmytest_*.csvには、複数のI/Oパフォーマンス測定の値がカンマ区切りで含まれています。詳細は、「Orionの出力ファイル」を参照してください。

8.4.3 Orionの入力ファイル

Orion -testname <*testname*>パラメータを指定すると、Orionの入力および出力ファイル名にテスト名の接頭辞が設定されます。-testnameオプションのデフォルト値は「orion」です。

Orion入力ファイル、<*testname*>.lunには、改行で区切ったLUNのリストを含める必要があります。

8.4.4 Orionのパラメータ

I/Oワークロードのタイプや他のOrionオプションを指定するには、Orionコマンド・パラメータを使用します。

8.4.4.1 Orionに必要なパラメータ

Orionコマンドには-runパラメータが必要です。表8-5で-runパラメータについて説明します。

表8-5 必要なOrionパラメータ

オプション	説明	デフォルト

-run *level*	テスト実行レベルを*level*に指定します。このオプションは実行レベルを定義するもので、複雑なコマンドに高度なレベルを指定することができます。-run advancedを設定しないで、-cache_sizeと-verbose以外の他のパラメータを設定すると、結果はエラーになります。
	
	advancedを除くすべての-run *level*設定は、あらかじめ指定されているパラメータ・セットを使用します。
	
	*level*には次のいずれかを指定します。
	
	• **oltp**
	
	小規模な(8K)ランダムI/Oで負荷を増加させて最大IOPSを特定するためのテストです。
	
	このパラメータは、次のOrion呼び出しに相当します。
	
	`%> ./orion -run advanced \`
	`-num_large 0 -size_small 8 -type rand \`
	`-simulate concat -write 0 -duration 60 \`
	`-matrix row`
	
	• **dss**
	
	大規模な(1M)ランダムI/Oで負荷を増加させて最大スループットを特定するためのテストです。
	
	このパラメータは、次のOrion呼び出しに相当します。
	
	`%> ./orion -run advanced \`
	`-num_small 0 -size_large 1024 -type rand \`
	`-simulate concat -write 0 -duration 60 \`
	`-matrix column`
	
	• **simple**
	
	一定範囲の負荷レベルでの小規模なランダムI/Oと大規模なランダムI/Oのワークロードを生成します。このオプションでは、小規模I/Oと大規模I/Oが個別にテストされます。この実行レベルで指定できるオプション・パラメータは、-cache_sizeと-verboseのみです。
	
	このパラメータは、次のOrion呼び出しに相当します。
	
	`%> ./orion -run advanced \`
	`-size_small 8 -size_large 1024 -type rand \`
	`-simulate concat -write 0 -duration 60 \`
	`-matrix basic`
	
	• **normal**
	
	simpleと同じですが、一定範囲の負荷レベルでの小規模なランダムI/Oと大規模なランダムI/Oの組合せも生成されます。この実行レベルで指定できるオプション・パラメータは、-cache_sizeと-verboseのみです。
	
	このパラメータは、次のOrion呼び出しに相当します。
	
	`%> ./orion -run advanced \`
	`-size_small 8 -size_large 1024 -type rand \`
	`-simulate concat -write 0 -duration 60 \`
	`-matrix detailed`
	
	• **advanced**
	
	オプション・パラメータで指定したワークロードをテストします。この実行レベルではどのオプション・パラメータでも指定できます。
		normal

8.4.4.2 Orionのオプション・パラメータ

表8-5 Orionのオプション・パラメータ

オプション	説明	デフォルト
-cache_size *num*	ストレージ・アレイの読取りまたは書込みキャッシュのサイズ(MB)。大規模な順次I/Oワークロードでは、Orionは各データ・ポイントの前に大規模なランダムI/Oを行うことでキャッシュをウォーミングします。Orionはキャッシュ・サイズを使用して、このキャッシュ・ウォーミング操作の期間を決定します。0に設定すると、キャッシュ・ウォーミングは行われません。 このオプションを0に設定しない場合は、Orionは大規模な順次データ・ポイントごとに、その前に複数の未測定のランダムI/Oを発行します。これらのI/Oは、ストレージ・アレイのキャッシュ(存在する場合)をランダム・データで満たし、あるデータ・ポイントからのI/Oが次のデータ・ポイントのキャッシュ・ヒットにならないようにします。読取りテストではジャンク読取りを行い、書込みテストではジャンク書込みを行います。こ	デフォルト値 指定しないと、ウォーミングはデフォルト時間(2分間)で実行されます。つまり、各データ・ポイントの前に、未測定のランダムI/Oを2分間発行します。

	のキャッシュ・ウォーミングは、指定されると、I/Oの*num* MBの読取りまたは書込みが終了するまで実行されます。	
-duration *num_seconds*	各データ・ポイントをテストする時間を値*num_seconds*に秒単位で設定します。	デフォルト値: 60
-help	Orionのヘルプ情報を出力します。helpとともに設定されたその他のオプションはすべて無視されます。	
-matrix *type*	一定範囲の負荷でテストする混合ワークロードのタイプ。Orionテストは複数のデータ・ポイント・テストからなります。データ・ポイント・テストは、2次元マトリックスとして表すことができます。 マトリックスの各列は、同じ小規模I/O負荷によるデータ・ポイント・テストを表しますが、大規模I/O負荷は異なります。各行は、同じ大規模I/O負荷によるデータ・ポイント・テストを表しますが、小規模I/O負荷は異なります。Orionテストは、次のマトリックスの*type*に応じて、1つのポイント、1つの列、1つの行に対して、またはマトリックス全体に対して行うことができます。 • **basic:** 混合ワークロードなし。小規模なランダム・ワークロードと、大規模なランダムまたは順次ワークロードは、個別にテストされます。小規模I/Oのみをテストし、次に大規模I/Oのみをテストします。 • **detailed:** 小規模なランダム・ワークロードと、大規模なランダムまたは順次ワークロードは、組み合せてテストされます。マトリックス全体をテストします。 • **point:** *S* 未処理の小規模なランダムI/Oおよび*L* 未処理の大規模なランダムI/Oまたは順次ストリームによるデータポイント。*S*は、-num_smallパラメータで設定されます。*L*は、-num_largeパラメータで設定されます。-num_smallにより小規模I/Oを、-num_largeにより大規模I/Oをテストします。 • **col:** 大規模なランダムまたは順次ワークロードのみ。-num_small小規模I/Oを使用して、大規模I/Oの負荷を変化させてテストします。 • **row:** 小規模なランダム・ワークロードのみ。-num_large大規模I/Oを使用して、小規模I/Oの負荷を変化させてテストします。 • **max:** detailedと同じですが、パラメータ-num_smallと-num_largeで指定された最大負荷でのワークロードのみをテストします。-num_smallと-num_largeの制限値まで負荷を変化させてテストします。	デフォルト値: basic
-num_disks *value*	テストに使用する物理的なディスクの数を指定します。これを使用して負荷の範囲が生成されます。ディスク(物理的なスピンドル)の数を指定します。この数の*value*を使用して、Orionがテストする必要のある負荷の範囲を判断します。このパラメータを増やすと、Orionはより高いI/O負荷を使用します。	デフォルト値: <*testname*>.lun内のLUNの数
-num_large *value*	大規模I/O負荷を制御します。 注意: このオプションは、-matrixがrow、point、またはmaxに指定されている場合のみ適用されます。 -typeオプションがrandに設定されている場合は、パラメータ引数の*value*で未処理の大規模I/Oの数を指定します。 -typeオプションがseqに設定されている場合は、パラメータ引数の*value*で順次I/Oストリームの数を指定します。	デフォルト値: デフォルトなし
-num_small	小規模なランダムI/Oワークロードに対する未処理I/Oの最大数を指定します。 注意: このオプションは、-matrixがcol、point、またはmaxに指定されている場合のみ適用されます。	デフォルト値: デフォルトなし
-num_streamIO *num*	1ストリーム当たりの同時I/Oの数を*num*に指定します。 注意: このパラメータは、-typeがseqの場合のみ使用されます。	デフォルト値: 4
-simulate *type*	大規模な順次I/Oワークロードのシミュレーションのためのデータ・レイアウト。Orionは、これらのいずれかの方法で指定されたLUNの組合せによって形成された仮想LUN上でテストします。*type*には次のいずれかを指定します。 • **concat:** 仮想ボリュームは、指定されたLUNをシリアルにつなぐことによりシミュレートされます。この仮想ボリュームに対する順次テストは、あるポイントから各LUNの終わりまで行われ、続けて次のLUNの先頭から最後までというように続きます。 • **raid0:** 仮想ボリュームは、指定されたLUNをまたいでストライプ化してシミュレートされます。各順次ストリームは、raid0ストライプ化を使用してすべてのLUNをまたいでI/Oを発行します。ストライプ深度はデフォルトで1Mで、Oracle Automatic Storage Managementのストライプ深度に一致しています。これは-stripeパラメータを使用して変更できます。 I/Oのオフセットは、次のようにして決定されます。 小規模なランダム・ワークロードおよび大規模なランダム・ワークロードの場合: • LUNは単一の仮想LUN(VLUN)に連結され、VLUN内でランダム・オフセットが選択されます。 大規模な順次ワークロードの場合: • ストライプ化あり(-simulate raid0)。LUNを使用して単一のストライプ化されたVLUNが作成されます。小規模なランダム・ワークロードが同時にない場合は、順次ストリームはストライプ化されたVLUN内の固定のオフセットで開始されます。*n*ストリームの場合、*n*が1でなければ、ストリーム*i*はオフセットVLUNsize * (*i* + 1) / (*n* + 1)で開始されます。小規模なランダム・ワークロードが同時にある場合は、ストリームはストライプ化されたVLUN内のランダム・オフセットで開始されます。 • ストライプ化なし(-simulate CONCAT)。LUNは単一のVLUNに連結されます。ストリームは単一のVLUN内のランダム・オフセットで開始されます。 このパラメータは通常、-typeがseqの場合のみ使用されます。	デフォルト値: concat
-size_large *num*	大規模なランダムまたは順次I/Oワークロードの I/Oのサイズ(KB)を*num*に指定します。	デフォルト値: 1024

-size_small *num*	小規模なランダムI/OワークロードのI/Oのサイズ(KB)をnumに指定します。	デフォルト値: 8
-testname *tname*	*tname*にテスト実行の識別子を指定します。指定する場合は、LUNディスクを含む入力ファイルまたはファイル名を<*tname*>.lunという名前にする必要があります。 出力ファイルは、接頭辞<*tname*>_を付けた名前になります。	デフォルト値: orion
-type [rand | seq]	大規模I/Oワークロードのタイプ。 • **rand**: ランダムに分配された大規模I/O。 • **seq**: 順次ストリームの大規模I/O。	デフォルト値: rand
-verbose	ステータスとトレース情報を標準出力に出力します。	デフォルト値: オプションの設定なし
-write *num_write*	書込みI/Oの割合を*num_write*に指定します。残りは読取りI/Oになります。 このパラメータは大規模I/Oワークロードと小規模I/Oワークロードの両方に適用されます。大規模な順次I/Oの場合、各ストリームは読取り専用か書込み専用のどちらかです。そのため、パラメータは書込み専用のストリームの割合を指定します。ディスクに書き込まれるデータは不要データで、ディスク上の既存データとは無関係です。 **注意**: 書込みテストでは、指定されたLUN上のすべてのデータが消去されます。	デフォルト値: 0

注意:
書込みテストでは、指定されたLUN上のすべてのデータが消去されます。

8.4.4.3 Orionのコマンドライン例

次に様々なタイプのI/Oワークロードに対するOrionコマンドの例を示します。

1. OLTPデータベースのストレージを評価する場合:

   ```
   -run oltp
   ```

2. データ・ウェアハウスのストレージを評価する場合:

   ```
   -run dss
   ```

3. データの基本セット用:

   ```
   -run normal
   ```

4. 読取り専用の小規模なランダムI/Oワークロードと大規模なランダムI/Oワークロードでのストレージ・パフォーマンスを理解する場合:

   ```
   $ orion -run simple
   ```

5. 小規模なランダムI/Oワークロードと大規模なランダムI/Oワークロードの混合でのストレージ・パフォーマンスを理解する場合:

   ```
   $ orion -run normal
   ```

6. 32KBと1MBの読取りの組合せをランダムな位置に生成する場合:

   ```
   $ orion -run advanced -size_small 32 \
   -size_large 1024 -type rand -matrix detailed
   ```

7. 複数の1MBの順次書込みストリームを生成して、1MB RAID0ストライプ化をシミュレートする場合:

   ```
   $ orion -run advanced -simulate raid0 \
   -stripe 1024 -write 100 -type seq -matrix col -num_small 0
   ```

8. 32KBと1MBの読取りの組合せをランダムな位置に生成する場合:

   ```
    -run advanced -size_small 32 -size_large 1024 -type rand -matrix detailed
   ```

9. 複数の1MBの順次書込みストリームを生成して、RAID0ストライプ化をシミュレートする場合:

   ```
    -run advanced -simulate raid0 -write 100 -type seq -matrix col -num_small 0
   ```

8.4.5 Orionの出力ファイル

テスト実行の出力ファイルには、接頭辞<*testname*>_<*date*>が付きます。ここで*date*は*yyyymmdd_hhmm*です。

表8-7にOrionの出力ファイルを示します。

表8-7 Orionで生成される出力ファイル

出力ファイル	説明
<*testname*>_<*date*>_hist.csv	I/O待機時間のヒストグラム。
<*testname*>_<*date*>_iops.csv	小規模I/Oのパフォーマンス結果(IOPS)。
<*testname*>_<*date*>_lat.csv	小規模I/Oの待機時間(マイクロ秒)。
<*testname*>_<*date*>_mbps.csv	大規模I/Oのパフォーマンス結果(MBPS)。
<*testname*>_<*date*>_summary.txt	入力パラメータのサマリーが、小規模I/Oの最短待機時間(秒)、最大MBPS、および最大IOPSとともに表示されます。
<*testname*>_<*date*>_trace.txt	拡張された、未処理の出力。

> **注意:**
> 書込みテストを実行する場合は、LUNに格納されたデータが失われることに注意してください。

8.4.5.1 Orionの出力ファイル例

Orionは、表8-7に示すように、複数の出力ファイルを作成します。「Orionの開始」の項で示した例「mytest」の場合、出力ファイルは次のようになります。

- **mytest_summary.txt**: このファイルには次が含まれます。
  - 入力パラメータ
  - 大規模なランダムまたは順次ワークロードで観測された最大スループット
  - 小規模なランダム・ワークロードで観測された最大I/O率
  - 小規模なランダム・ワークロードで観測された最大待機時間
- **mytest_mbps.csv:** 大規模なランダムまたは順次ワークロードのデータ転送速度(MBPS)結果を含むカンマ区切りのファイル。一般的には、このファイルと他のすべてのCSVファイルには、2次元の表が含まれます。表の各行は大規模なI/O負荷レベルに対応し、各列は特定の小規模なI/O負荷レベルに対応します。このため、列のヘッダーは未処理の小規模I/Oの数、行のヘッダーは未処理の大規模I/Oの数(大規模なランダムI/Oテストの場合)または順次ストリームの数(大規模な順次I/Oテストの場合)になります。

  例8-1に、「mytest」に対するOrion MBPSのCSV出力ファイルのデータ・ポイントを先頭からいくつか示します。単純なmytestコマンドラインでは、大規模I/Oと小規模I/Oの組合せはテストされません。このため、MBPSファイルには未処理0の小規模I/Oに対応する1列のみが含まれます。例8-1では、8つの未処理の大規模な読取りと小規模I/Oなしの負荷レベルで、レポート・データは103.06 MBPSのスループットを示しています。

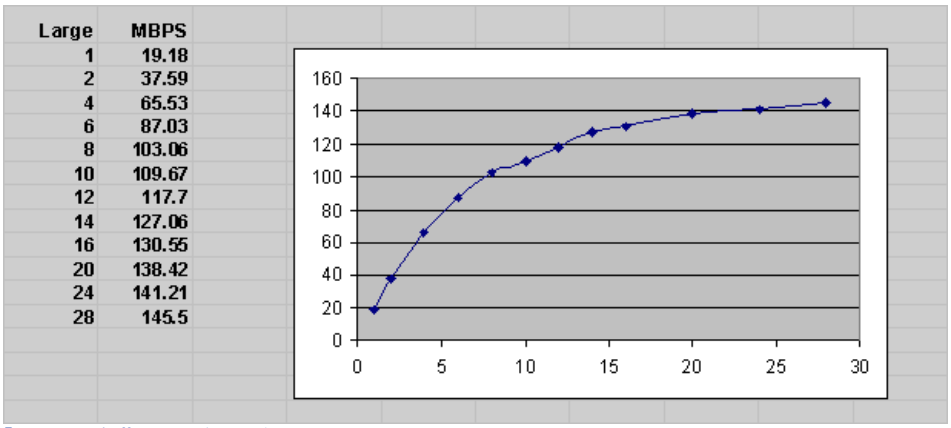
  例8-1 Mytestのデータ・ポイント例

```
Large/Small,       0

1,    19.18
2,    37.59
4,    65.53
6,    87.03
8,   103.06
10,  109.67
. .    . . .
. .    . . .
```

  図8-2に、様々な大規模I/O負荷レベルで測定されたデータ転送速度の例を示します。この図は、mytest_mbps.csvをスプレッドシートに読み込んでデータ・ポイントをグラフ化して生成できます。Orionはこのようなグラフを直接的には生成しません。X軸は未処理の大規模な読取りの数に対応し、Y軸は測定されたスループットに対応します。

  図8-2に示すグラフは、標準的なストレージ・システムの動作を示しています。未処理のI/Oリクエストの数が増えるにつれて、スループットは上がります。ただし、特定のポイントでスループット・レベルは変化しなくなり、ストレージ・システムの最大スループット値であることを示します。

  図8-2 I/O負荷レベルの例



「図8-2 I/O負荷レベルの例」の説明

- **mytest_mbps.csv:** 小規模なランダム・ワークロードのI/Oスループット(IOPS)結果を含むカンマ区切りのファイル。MBPSファイルと同様に、列のヘッダーは未処理の小規模I/Oの数、行のヘッダーは大規模なランダムI/Oのテストでは未処理の大規模I/Oの数、大規模な順次I/Oのテストでは順次ストリームの数です。

  一般的には、CSVファイルには2次元の表が含まれます。ただし、大規模I/Oと小規模I/Oの組合せをテストしない単純なテストの場合は、結果ファイルは1行のみになります。このため、IOPSの結果ファイルは大規模I/Oを含まない1行のみになります。例8-2で示すように、12の未処理の小規模な読取りと大規模I/Oなしのデータ・ポイント例では、スループットは951 IOPSです。

  例8-2 12の小規模な読取りと大規模な読取りなしのデータ・ポイント例

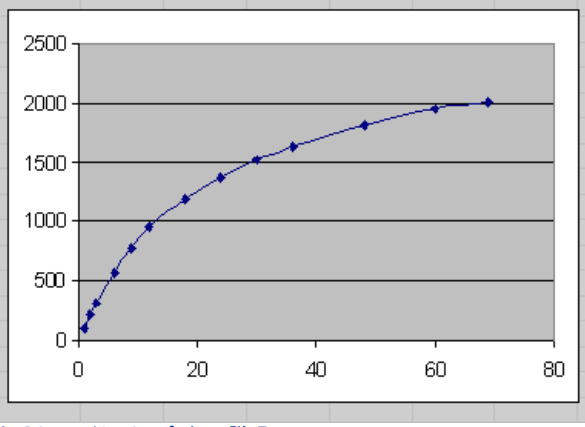
```
Large/Small,      1,      2,      3,      6,      9,     12 . . . .

0,              105,    208,    309,    569,    782,    951 . . . .
```

  図8-3に示すグラフは、mytest_iops.csvをExcelに読み込んでデータをグラフ化して生成したもので、様々な小規模I/O負荷レベルで観測されたIOPSスループットを示しています。

  図8-3は、標準的なストレージ・システムの動作を示しています。未処理のI/Oリクエストの数が増えるにつれて、スループットは上がります。ただし、特定のポイントでスループット・レベルは変化しなくなり、ストレージ・システムが最大スループット値に達したことを示します。スループット・レベルが上がるほど、I/Oリクエストの待機時間は大幅に増加します。そのため、このデータをmytest_lat.csvに生成された待機時間結果で提供される待機時間データとともに表示することが重要です。

  図8-3 様々な小規模I/O負荷レベルでのI/Oスループット

Small	IOPS
1	105
2	208
3	309
6	569
9	782
12	951
18	1189
24	1369
30	1521
36	1638
48	1815
60	1941
69	2013



「図8-3 様々な小規模I/O負荷レベルでのI/Oスループット」の説明

- **mytest_lat.csv:** 小規模なランダム・ワークロードの待機時間結果を含むカンマ区切りのファイル。MBPSファイルやIOPSファイルと同様に、列のヘッダーは未処理の小規模I/Oの数、行のヘッダーは未処理の大規模I/Oの数(大規模なランダムI/Oをテストする場合)または順次ストリームの数です。

  一般的には、CSVファイルには2次元の表が含まれます。ただし、大規模I/Oと小規模I/Oの組合せをテストしない単純なテストの場合は、結果ファイルは1行のみになります。このため、IOPSの結果ファイルは大規模I/Oを含まない1行のみになります。例8-3に示す例では、12の未処理の小規模な読取りと大規模I/Oなしの負荷レベルで、生成される結果は22.25ミリ秒のI/O応答待機時間を示しています。

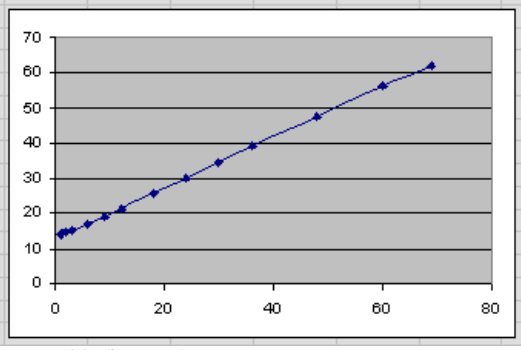
  例8-3 12の小規模な読取りと大規模な読取りなしのCSVファイル例

```
Large/Small, 1, 2, 3, 6, 9, 12 . . . .

0, 14.22, 14.69, 15.09, 16.98, 18.91, 21.25 . . . .
```

  図8-4のグラフは、mytest_lat.csvをExcelに読み込んでデータをグラフ化して生成したもので、mytestの様々な小規模I/O負荷レベルでの小規模I/Oの待機時間を示しています。

  図8-4 小規模I/O負荷レベルでの待機時間

1	14.22
2	14.69
3	15.09
6	16.98
9	18.91
12	21.25
18	25.92
24	30.2
30	34.8
36	39.04
48	47.68
60	56.3
69	62.16



「図8-4 小規模I/O負荷レベルでのI/O待機時間」の説明

- **mytest_trace.txt**: 拡張された、未処理のテスト出力を含みます。

> **注意:**
> Orionは、標準出力でのテスト中に発生したエラーを報告します。

8.4.6 Orionのトラブルシューティング

1. <*testname*>.lunファイルで指定した1つ以上のボリュームでI/Oエラーが発生する場合;
   - ddなどのファイル・コピー・プログラムを使用して、テスト、読取り、または書込みと同じモードでボリュームにアクセスできることを確認してください。
   - ホストのオペレーティング・システムのバージョンが非同期I/Oを行えることを確認してください。
   - LinuxおよびSolarisでは、ライブラリlibaioが標準libディレクトリにあるか、またはシェル環境のライブラリ・パス変数(通常はシェルに対応してLD_LIBRARY_PATHまたはLIBPATH)を使用してこのライブラリにアクセスできることが必要です。
2. NASストレージで実行している場合:
   - Orionを実行するには、ファイル・システムを正しくマウントする必要があります。詳細は、Oracleインストレーション・ガイドを参照してください(たとえば、Databaseインストレーション・ガイド for Linux x86のセクション、付録B「NASデバイスの使用」など)。
   - mytest.lunファイルには既存のファイルの1つ以上のパスを含める必要があります。Orionはディレクトリやマウント・ポイントに対して機能することはありません。意味のあるテストにするには、ファイルは十分な大きさにする必要があります。このファイルのサイズは、データファイルの最終的な予測サイズ(つまり数年使用後のサイズ)にする必要があります。
   - Linux(2.6カーネルを含む)のNFSで非同期I/Oを行うと、パフォーマンスが低下することがあります。
   - 読取りテストを行っていて、初期化されていないかまだ書込みが行われていないファイルの未使用ブロックを読取りがヒットしている場合、高性能なNASシステムではゼロ設定されたブロックを返すことで読取りに見せかけることがあります。これが発生した場合は、予想外によいパフォーマンスになります。

これを回避するには、読込みテストを実行する前に、ddなどのツールを使用してすべてのブロックに書込みを行います。

3. OrionをWindowsで実行する場合: RAWパーティションでのテストでは、パーティションを一時的にドライブ文字にマップして、これらのドライブ文字をtest.lunファイルで指定する必要があります。
4. Orion 32ビットLinux/x86バイナリをx86_64システムで実行する場合: 32ビットlibaio.soファイルを、同じLinuxバージョンを実行している32ビットのコンピュータからコピーしてください。
5. 多数のディスク(num_disksが30を超える)を使用してテストする場合:
   - 各データ・ポイントに長時間(120秒以上など)を指定するには、-durationオプションを使用してください(詳細はオプション・パラメータのセクションを参照)。Orionはすべてのスピンドルを特定の負荷レベルで実行を継続させようとするため、各データ・ポイントで立ち上げ時間が必要となり、これによってテストに要する時間が長くなることがあります。
   - 期間の値を増やすように指示する次のエラー・メッセージが表示されることがあります。

     ```
     Specify a longer -duration value.
     ```

     おおよその目安として、期間はスピンドル数の2倍程度にするとよいようです。使用しているディスク技術によって、プラットフォームで必要とする時間は上下します。
6. Orionによって使用されるライブラリに関するエラーが発生した場合:
   - Linux/Solaris: I/Oエラーのトラブルシューティングを参照してください。
   - **NT-Only:** 配布に含まれているOracleライブラリの移動や削除は行わないでください。これらはorion.exeと同じディレクトリに置く必要があります。
7. 「信じられないほどよい」パフォーマンス数値になった場合:
   - 大規模な読取りまたは書込みキャッシュがあるか、Orionプログラムとディスク・スピンドルの間のどこかでキャッシュの読取りと書込みを行っている場合があります。通常は、最も効果が高いのは、ストレージ・アレイ・コントローラです。このキャッシュのサイズを調べて、-cache_size advancedオプションを使用してそれをOrionに指定してください(詳細はオプション・パラメータのセクションを参照)。
   - ボリュームの合計サイズが、これまでの1つ以上のキャッシュと比較して極端に小さい場合があります。キャッシュを無効にするようにしてください。ストレージを共有する他のボリュームが本番環境で著しいI/Oアクティビティを示す(そして結果的に共有キャッシュの大部分を使用する)場合には、これが必要になります。
8. Orionが長い予測実行時間を報告している場合:
   - -num_disksが高いときに実行時間は長くなります。Orionは内部的に線形計算式を使用して、指定された数のディスクを維持するためにかかる時間を特定します。
   - -cache_sizeパラメータは、指定していない場合でも、実行時間に影響します。Orionはデフォルトでデータ・ポイントごとに2分間のキャッシュ・ウォーミングを行います。キャッシュを無効にする場合は、-cache_size 0を指定します。
   - -durationに大きい値を指定すると、予想通り、実行時間は長くなります。