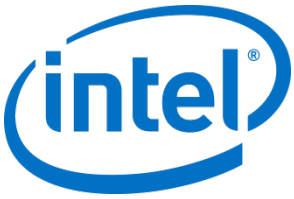


インテル® キャッシュ・アクセラレーション・ソフトウェア Linux* 版 v3.1

管理者ガイド

2016 年 7 月

注文番号：328496-011JA

内容

1 はじめに

このガイドは、インテル® キャッシュ・アクセラレーション・ソフトウェア (インテル® CAS) Linux* 版 v3.1 版をインストールして使用を開始するための最も速い方法を提供します。このガイドでは、ユーザーがストレージとアプリケーション管理および基本的な Linux* システム管理の基本知識を持っていることを前提にしています。

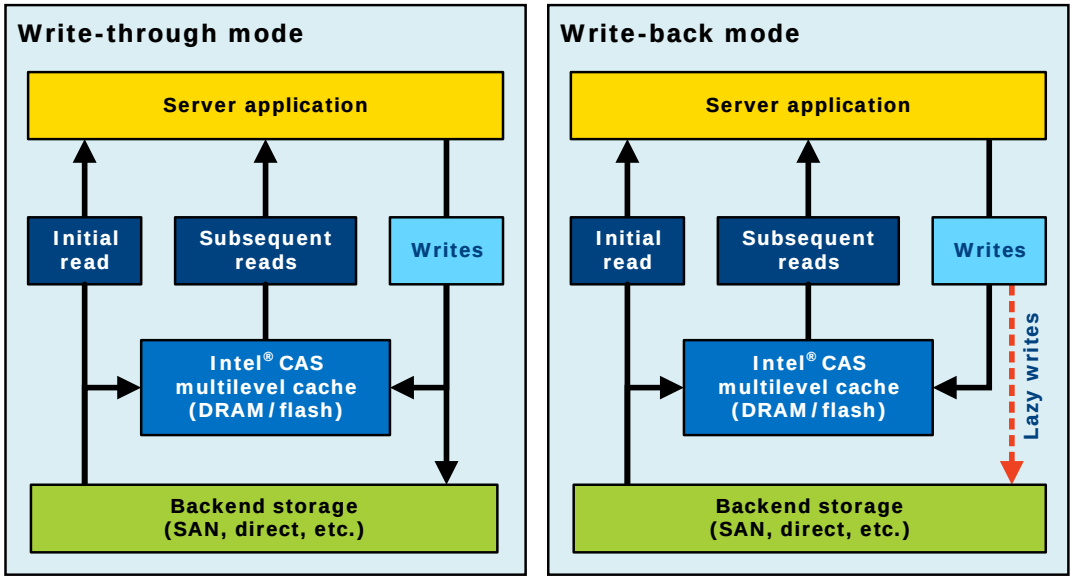
1.1 インテル® CAS の概要

インテル® CAS はアクティブ (*ホット*) データをサーバー内のローカル・フラッシュ・デバイスにキャッシュすることによって、Linux* アプリケーションを加速します。インテル® CAS は、ローカルの高性能フラッシュメディアをアプリケーション・サーバー内のキャッシュドライブ・メディアとして使用して、キャッシュをサーバーレベルで実装するため、ストレージのレイテンシーが縮減されます。

インテル® CAS は Linux* オペレーティング・システムにカーネルモジュールとしてインストールされます。この統合の特質がユーザー、アプリケーションおよび既存のストレージ・インフラストラクチャーに透過的なキャッシュ・ソリューションを提供します。ストレージの移行またはアプリケーションの変更は不要です。

図 1 で示すように、初期読み取りデータはバックエンド・ストレージから取得され、インテル® CAS キャッシュにコピーされます。2 番目の読み取りはデータをシステムメモリーに進めます。後続の読み取りは高性能 RAM またはフラッシュの速度で返されます。ライトスルー・モードでは、すべてのデータがバックエンド・ストレージとキャッシュの両方に同時に書き込まれます。ライトバック・モードでは、すべてのデータがキャッシュに書き込まれ、最終的にバックエンド・ストレージに送られます。キャッシュがいっぱいの場合、インテル® CAS の所有権保有の排除アルゴリズムを使用して新しく識別されたアクティブデータがキャッシュから古いデータを排除します。

図 1 : ブロック図 (ライトスルー・モードとライトバック・モード)



インテル® CAS Linux* 版は、デフォルトで、選択されたコアデバイスのすべてのアクティビティをキャッシュするブロックに基づくキャッシュ・アーキテクチャ-を採用します。-I オプションまたは *include.conf* 構成ファイル (15 ページから始まるインテル® CAS の設定を参照) を使用して、特定のファイルまたは複数のファイルのリストを排他的にキャッシュすることができます。

1.2 今回のリリースの新機能

このリリースには、次のような新機能があります：

- **NEW！ノンストップ・アップグレード機能** - I/O を停止または再起動する必要なしに CAS の今後のバージョンへのアップグレードを行えるようにします。この保守機能は、システムを CAS の新しいバージョンにアップグレード中に作業負荷の実行を続行できるようにします。Ceph* などのソフトウェア・デファインド・ストレージ・ソリューションに関しては、この機能は CAS にアップグレード中にストレージデバイスを停止する必要をなくします (3.5 章を参照)。
- **選択可能なキャッシュ・ライン・サイズ** - キャッシュ・ライン・サイズを 4k (デフォルト)、8k、16k、32k、64k のキャッシュ・ラインから選択して、メモリー内メタデータ・ストレージの DRAM フットプリントの要件を削減できます。
- **自動パーティション・マッピング** - CAS はデバイス全体を高速化する際に、基礎をなすコアデバイスのパーティション・スキームに自動的に一致させます。この機能により、Ceph* の導入およびアクティベーション・スクリプトとの互換性が向上されています。
- **キャッシュのフラッシュ中の連続 I/O** - CAS はキャッシュのフラッシュ操作中に、エクスポートされたデバイス (例：intelcas1-1) への I/O 処理を続行します。(以前はフラッシュ中に I/O は一時停止されました。)
- **改善されたデバイスのエラー処理** - キャッシュデバイスの I/O エラーが発生した場合、CAS は I/O 操作をインテリジェントに続行します。
- **v3.0 からのバグ修正。**

重要： 次の変更に注意して、ユーザーが作成したスクリプトをそれらの変更に応じて更新してください：

- **--Start-cache コマンドからの -eviction-policy と --metadata-variant の廃止**：これらのオプションは、今まではメモリー内キャッシュメタデータに必要な DRAM メモリー・フットプリントを削減するために使用されました。.新しい --cache-line-size 機能がこの目的を果たし、古くなったコマンドに置き換わります。

1.3 ドキュメントの表記規則

このマニュアルでは、次の表記規則が使用されます：

- `Courier` フォント - コードの例、コマンドラインのエントリー。
- *斜体* - ユーザー・インターフェイスのアイテム、ファイル名など。
- **< >** - 必須コマンドを示します
- **[ ]** - オプションコマンドを示します

1.4 参考資料

インテル® CAS のテストと操作のヘルプを得るには、またはキャッシュとアプリケーションの I/O パフォーマンス管理の詳細については、表 1 に示すリソースとツールを参照してください。.

表 1： 参考資料

名前	場所
DT 汎用データ・テスト・プログラム	http://www.scsifaq.org/RMiller_Tools/dt.html
FIO	http://freecode.com/projects/fio
Vdbench	http://vdbench.sourceforge.net

1.5 改訂履歴

改訂	詳細	日付
001B	ドキュメントのベータ初版。	2013 年 11 月
001	ドキュメントの初版。	2013 年 2 月
002	Linux* v2.1 用のドキュメントの更新。	2013 年 5 月
003	Linux* v2.5 用のドキュメントの更新。	2013 年 8 月
004	Linux* v2.6 用のドキュメントの更新。	2013 年 12 月
005	Linux* v2.6.1 用のドキュメントの更新。	2014 年 4 月
006	Linux* v2.7 用のドキュメントの更新。	2014 年 5 月
007	Linux* v2.7 GA 用のドキュメントの更新。	2014 年 7 月
008	Linux* v2.8 GA 用のドキュメントの更新。	2014 年 12 月
009	Linux* v2.9 GA 用のドキュメントの更新。	2015 年 6 月
010	Linux* v3.0 GA 用のドキュメントの更新。	2016 年 1 月
011	Linux* v3.1 GA 用のドキュメントの更新。	2016 年 7 月

2 製品仕様とシステム要件

2.1 対応オペレーティング・システム

表 2 は、インテル® CAS がサポートする 64 ビット・プロセッサーのプラットフォームを示します。

表 2： 対応オペレーティング・システム

オペレーティング・システム	カーネル
Red Hat* Enterprise Linux* (RHEL*) 6.6	x86_64、カーネル 2.6.32-504.3.3
Red Hat* Enterprise Linux* (RHEL*) 6.7	x86_64、カーネル 2.6.32-573
Red Hat* Enterprise Linux* (RHEL*) 7.0	x86_64、カーネル 3.10.0-123.13.2
Red Hat* Enterprise Linux* (RHEL*) 7.1	x86_64、カーネル 3.10.0-229
Red Hat* Enterprise Linux* (RHEL*) 7.2	x86_64、カーネル 3.10.0-327
CentOS* 6.6	x86_64、カーネル 2.6.32-504.3.3
CentOS* 6.7	x86_64、カーネル 2.6.32-573
CentOS* 7.0	x86_64、カーネル 3.10.0-123.13.2
CentOS* 7.1	x86_64、カーネル 3.10.0-229
CentOS* 7.2	x86_64、カーネル 3.10.0-327
Oracle 6.6	x86_64、カーネル 2.6.32-504.3.3
Oracle 6.7	x86_64、カーネル 2.6.32-573
SUSE Linux* Enterprise Server (SLES*) バージョン 11 SP4	x86_64、カーネル 3.0.101-63.1
Red Hat* Enterprise Linux* (RHEL*) 6.5 (カスタム)	x86_64、カーネル 3.10.0-123.4.4 (カスタム)
Ubuntu* Server 14.04.3	x86_64、カーネル 3.19.0-39
その他のディストリビューション - インテル® CAS は他のディストリビューションと他のカーネルのソースからのインストールとコンパイルを行いますが、ユーザーがサポートを受けるには、検証されているディストリビューションとカーネルで問題を再現する必要が生じることがあります。	その他のカーネル

表 3 はインテル® CAS でサポートされているハイパーバイザー/OS の組み合わせを示します。

表 3： サポートされているハイパーバイザー/OS の組み合わせ

ハイパーバイザー	サポートされている構成	注
Xen*	ハイパーバイザーまたはゲストでサポートされます。	準仮想化ドライバーはサポートされていません。
KVM*	ハイパーバイザーまたはゲストでサポートされます。	
VMware*	ゲストでサポートされています。	

2.2 システム要件

表 4 はインテル® CAS のシステム要件を示します。

表 4 : インテル® CAS システム要件

メモリー	RAM の要件は約 1GiB + 2% キャッシュデバイス容量 * 4KiB/<選択したキャッシュ・ライン・サイズ> です。たとえば 4KiB (デフォルト) キャッシュ・ライン・サイズを使用するときは、RAM の要件は 1GiB + 2% キャッシュデバイス容量となります。.
CPU オーバーヘッド	インテル® CAS は最小限の CPU 帯域幅 (ほとんどの場合 10% 未満)。インテル® CAS が最適に機能するのに十分な CPU 容量があることを確認してください。
フラッシュ/SSD	Linux* のあらゆるフラッシュデバイス (SAS、SATA、PCIe*、ファイバーチャネル、RAID) がサポートされ、直接接続、エキスパンダーを使用して接続、または SAN 経由で接続 (単一ワーカー使用) することができます。 さらに、次のインテル SSD が完全に対応しています : • インテル® SSD DC S3700 シリーズ • インテル® SSD DC P3700 シリーズ キャッシュデバイスの論理ブロックサイズがコア・ストレージ・デバイスの論理ブロックサイズ以下でなければなりません。60GB 以上の容量を推奨します。
ストレージ	プライマリー・ストレージ・デバイス : 論理ブロックデバイスとして表されるあらゆるデバイス。たとえば、ローカルディスク、RAID、SAN iSCSI (ファイバーチャネル、インフィニバンド、イーサネット) など。 コアデバイス (HDD) 論理ブロックサイズは 512 バイト以上でなければなりません。 オペレーティング・システムは、コア・ストレージ・デバイスがキャッシュされるものとは別のパーティションまたはドライブにインストールされている必要があります。
ファイルシステム	プライマリー・ストレージまたはコアストレージでは次のファイルシステムがサポートされます。 • ext3 (16TiB ボリュームサイズまでに制限されます)。これは ext3 ファイルシステムの制限です。 • ext4 • xfs **注 :** 最高のパフォーマンスを得るために、コアデバイスのファイルシステムのブロックサイズがコアデバイスのパーティション論理ブロックサイズに一致する必要があります。 **注 :** 最高のパフォーマンスを得るために、4KiB の倍数単位の I/O リクエストサイズを推奨します。
ソフトウェア	インテル® CAS をインストールする前に次の前提条件ソフトウェアをインストールする必要があります : • bzip2 • tar • sed • make • gcc* • kernel-devel • kernel-headers 次のソフトウェアはオプションです。希望する場合はインテル® CAS のインストールを行う前にインストールする必要があります : • dkms (http://linux.dell.com/dkms/)
電力の管理	電源管理のサスペンド (S3) と休止 (S4) の状態は無効になっている必要があります。

3 インテル® CAS のインストール

このセクションは、インテル® CAS の標準インストールの方法を説明します。インストール・パッケージは、キャッシュの動作を制御するキャッシュ・レイヤーと管理 CLI を提供する読み込み可能なカーネルモジュールで構成されます。

3.1 スリープステータスを無効にする

インテル® CAS を設定する前に OS で電源管理のサスペンド (S3) と休止 (S4) 状態を無効にする必要があります。お使いの OS でこの操作を行う方法については、OS のマニュアルを参照してください。

3.2 フラッシュデバイスの設定

インテル® CAS を設定する前に、フラッシュデバイスをインストールする必要があります。フラッシュデバイスは、Linux* オペレーティング・システムでサポートされるソリッドステート・ドライブ (SSD) または PCIe* フラッシュドライブ (詳細については、10 ページの 表 4 を参照してください) のどれでも使用できます。

フラッシュデバイスに含まれているインストール・ガイドの手順に従い、フラッシュデバイスのシリアル番号または WWN を書き留めて、システムが起動して実行されたら、その番号を識別できるようにします。インテル® CAS で使用するすべてのデバイスはアンマウントして、すべての自動マウント操作を無効にする必要があります。デバイスのパーティションをキャッシュ領域として使用する場合は、最高のパフォーマンスを得るために、そのパーティションがデバイスに割り当てられていることを確認してください。

最高のパフォーマンスを得るために、キャッシュデバイス (SSD) で *noop* IO スケジューラーを使用することを推奨します。最初に、キャッシュを有効にしたときに作成された intelcas 仮想ブロックデバイスがプライマリー・ストレージ・デバイスの IO スケジューラーを継承します。希望する場合は、ユーザーは仮想ブロックデバイスを作成後に仮想ブロックデバイスとプライマリー・ストレージ・デバイスの IO スケジューラーを個別に変更できます。

3.3 DKMS のインストール

インテル® CAS Linux* では、インテル® CAS のインストール前に DKMS がインストールされインテル® CAS のインストール中に選択された場合、カーネルのあらゆる更新時に DKMS 経由でソースから自動的に再コンパイルされます。

最新の DKMS ビルドは：http://linux.dell.com/dkms/ からダウンロードできます。

注： DKMS のインストールはオプションでインテル® CAS のインストール時に DKMS 経由でのインテル® CAS の自動再コンパイルを選択できます。

注： インテル® CAS を自動的に再コンパイルするために DKMS を使用することを強く推奨します。DKMS のインストールを選択しない場合、または自動再コンパイルをしない場合は、カーネルが更新されるたびに、ユーザーが手動でインテル® CAS を再インストールするまでインテル® CAS が無効になります。

3.4 ローカル・インストール

インストールを開始する前に管理者としてログオンするか、ソフトウェアをインストールするのに必要な権限があることを確認してください。インストールを実行するには、“root” 権限を持っているか、“root” としてログインする必要があります。

注： インテル® CAS 3.1 降が既にインストールされている場合、ノンストップ・アップグレードを実行して、キャッシュされたデバイスへの I/O を停止せずに最新のバージョンをインストールできます (詳細については、13 ページの インテル® CAS のアップグレード を参照してください)。

注： インテル® CAS 3.0 以前のバージョンが既にインストールされている場合は、インストールを行う前にまずインテル® CAS 3.0 以前のバージョンをアンインストールする必要があります (詳細については、13 ページの ソフトウェアのアンインストール を参照してください)。

1. インストール先のインテル® CAS サーバーでインテル® CAS インストール・ファイルをダウンロードするかディレクトリーにコピーします。インストールの説明ではサーバーのファイルシステム上の ~ または ~/ ($HOME と同等) の例を使用します。インストーラー・ファイル名は次の形式になります：

   ```
   installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run
   ```

   (*XX.XX.XX.XXXXXXXXX* はバージョン情報です。)たとえば、以下のようなことが可能になります。

   ```
   installer-Intel-CAS-03.01.00.00000000.run
   ```

   注： インストーラー・ファイルを必ずバイナリー形式でコピー/転送してください。ASCII またはバイナリー以外の形式で転送すると、インストーラーが機能しません。

   注： インストーラーは Linux* の正しいディストリビューションを自動的に検出します。

2. インテル® CAS インストール・ファイルのディレクトリーに移動します。

3. インストーラー・ファイルを実行可能にします：

   ```
   # chmod u+x installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run
   ```

4. インストールを開始します：

   ```
   #./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run
   ```

   注： インストーラーの詳細については、以下を入力します：

   ```
   #./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run --help
   ```

5. エンドユーザー使用許諾契約書 (EULA) を読み同意してインストールを続行します。

   使用許諾契約書に同意すると、インテル® CAS のインストールが正常に実行されたことを確認する次のテキストが表示されるはずです：

```
Checking for validated GNU/Linux distribution              [  OK  ]
Checking for validated kernel                              [  OK  ]
Verifying installation dependencies...
- Looking for bzip2...                                     [  OK  ]
- Looking for tar...                                       [  OK  ]
- Looking for sed...                                       [  OK  ]
- Looking for gcc...                                       [  OK  ]
- Looking for make...                                      [  OK  ]
- Looking for bash...                                      [  OK  ]
- Looking for kernel-devel...                              [  OK  ]
- Looking for Linux kernel headers in
   /lib/modules/3.10.0-123.13.2.el7.x86_64/build/...       [  OK  ]
Building userspace management utility...                   [ DONE ]
Compiling kernel module...
[================================================]100%   [ DONE ]
```

```
Verifying build products...                                    [ DONE ]
- Looking for dkms...                                          [  OK  ]
Intel(R) CAS installer has detected presence of DKMS on your GNU/Linux
operating system.

The installer can configure DKMS to recompile Intel(R) CAS after each Linux
kernel version upgrade. By accepting, you acknowledge that CAS may cease to
work properly if your new kernel version is not compatible with the software as
validated against supported configurations. By declining, you acknowledge that
Intel(R) CAS won't load after any Linux kernel upgrade until you manually
reinstall the software.

Do you want to configure DKMS for Intel(R) CAS? [y/N]
y
Installing casadm utility...                                   [ DONE ]
Setting up DKMS on a machine...                                [ DONE ]
Installing DKMS module... (this may take up to few minutes)[ DONE ]
Configuring service...                                         [ DONE ]
Module intelcas loaded successfully!
Setting module autoload
Installation successful!
Starting intelcas (via systemctl):                             [  OK  ]
```

3.5 インテル® CAS のアップグレード

インテル® CAS はノンストップ・アップグレード機能を備えています。この機能によりユーザーは、キャッシュデバイスへの I/O を停止または再起動する必要なしに CAS の新しいバージョンへのアップグレードを行えます。この保守機能は、インテル® CAS をアップグレードするためにユーザーがクラスターを停止する必要がないので、ソフトウェア・デファインド・ストレージ環境で特に役立ちます。

> ***注：*** インテル® CAS 3.0 以前のバージョンが既にインストールされている場合は、インストールを行う前にまずインテル® CAS 3.0 以前のバージョンをアンインストールする必要があります (詳細については、13 ページの ソフトウェアのアンインストール を参照してください)。

ノンストップ・アップグレードを実行するには：

1. インストーラー・ファイルを実行可能にします：

   ```
   # chmod u+x ./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run
   ```

2. --Reinstall オプションを使用してインテル® CAS を実行します：

   ```
   # ./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run –-reinstall
   ```

3.6 ソフトウェアのアンインストール

インテル® CAS をアンインストールする前に、20 ページの インテル® CAS を停止する に記述されているようにまずインテル® CAS を停止することが重要です。

さらに、インテル® CAS をアンインストールする前に、すべてのアプリケーションを閉じて、すべてのファイルシステムをアンマウントして、インテル® CAS を使用する既存のあらゆる LVM ボリュームを削除してください。インテル® CAS が使用中の場合、アンインストールのプロセスは開始されません。

インテル® CAS をローカルでアンインストールするには、アンインストールするインテル® CAS インストーラーを含むディレクトリーに移動します。次の手順はインテル® CAS ソフトウェアをアンインストールしますがインテル® CAS の新しいバージョン用に構成情報を保持します。

1. インストーラー・ファイルを実行可能にします：

   ```
   # chmod u+x ./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run
   ```

2. アンインストールを開始します：

   a. インテル® CAS 構成ファイルをシステムに残したい場合：

      ```
      # ./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run --uninstall
      ```

   b. 構成ファイルを含むすべてのインテル® CAS ファイルを削除したい場合：

      ```
      # ./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run --purge
      ```

   インテル® CAS が正常にアンインストールされたことを確認する次のテキストが表示されるはずです：

   ```
   Checking Intel(R) CAS configuration [ DONE ]
   Stopping intelcas service [ DONE ]
   Cleaning up... [ DONE ]
   Unloading kernel module... [ DONE ]
   Removing DKMS configuration... [ DONE ]
   Removing files... [ DONE ]
   Running depmod... [ DONE ]
   Uninstallation successful
   ```

3.7 詳細インストール・オプション

インテル® CAS インストーラーでは、配布された環境全体に渡って自動化されたインストールを実行できます。このタイプのインストールの詳細については、52 ページの リモート・インストーラーのパラメーター を参照してください。

4 インテル® CAS の設定

この章ではインテル® CAS の設定方法を説明します。作業を開始する前に、次の制限を理解することが重要です：

- インテル® CAS を開始、停止、または設定するには、root としてログオンするか root 権限を持っている必要があります。
- オペレーティング・システムがインストールされているパーティションを高速化することはできません。

インテル® CAS は、casadm というユーザーレベルのユーティリティーを提供し、このキャッシュ・ソフトウェアを簡単に設定できるようにします。このユーティリティーはデフォルトで */sbin* ディレクトリーにインストールされます。

注： スーパーユーザーによって root に昇進した場合は、$PATH 環境変数に /sbin ディレクトリーがあることは保証されません。casadm にアクセスできない場合は、最初にこの変数を調べます。コマンドの完全パス (/sbin/casadm) を使用します。

インテル® CAS シェルで casadm -H を入力して設定ユーティリティーを起動すると、コマンドラインのオプションと引き数のリストが返されます。異なるコマンドライン・オプションの詳細については、40 ページの 構成ツールの詳細 を参照してください。

インテル® CAS の設定では、「キャッシュデバイス」という用語は、遅い方のデバイスからデータをキャッシュするために使用する SSD/NVM デバイスまたは RAM ディスクをさし、「コアデバイス」という用語はキャッシュされる遅い方のデバイスをさします。

/dev/disk/by-uuid または*/dev/disk/by-id* ディレクトリで *ls -l* を実行する必要が生じることがあります。

4.1 設定ユーティリティーを使用する

次の手順では、*casadm* ユーティリティーを使用して一般的なディプロイメント用にインテル® CAS を設定する方法を詳しく説明します。*casadm* ユーティリティーで使用可能なコマンドの詳細については、40 ページから始まる 構成ツールの詳細 を参照してください。

以下は後続の手順を想定しています：

- キャッシュデバイス (SSD) は */dev/sdc* です。キャッシュデバイスはローブロックデバイスまたはブロックデバイスとしてアクセスされる RAM ディスクです。キャッシュデバイスにファイルシステムがなく、マウントされていないことを確認します。

  注： すべてのデータが上書きされるため、これらの手順を完了する前にキャッシュデバイス上のすべてのデータをバックアップしてください。

- キャッシュされるコアデバイス (プライマリー・ストレージ) は */dev/sdb* です。

  コアデバイスはファイルシステム (データの有無は問いません) を含むことができ、またはロー・ブロック・デバイスを使用することもできます。インテル® CAS 特有のファイルシステムのタイプと制限については、10 ページの システム要件 を参照してください。デバイスがマウントされていないことを確認します。

- 必要な場合は、正しいデバイスが設定されることを確認するために */dev/disk/by-uuid* または */dev/disk/by-id* ディレクトリーで *ls -l* または *ll* を実行してください。
- コアデバイス (HDD) 論理ブロックサイズは 512 バイト以上でなければなりません。
- キャッシュデバイスの論理ブロックサイズがコア・ストレージ・デバイスの論理ブロックサイズ以下でなければなりません。

- 両方のデバイスが /etc/fstab およびキャッシュデバイスまたはコアデバイスのいずれかを自動マウントする他のあらゆるメカニズムから削除されていることを確認します。

次のコマンドを使用してインテル® CAS モジュールが正しく読み込まれたことを確認します。

```
# service intelcas status
```

インテル® CAS モジュールが読み込まれない場合は、11 ページから始まるインストールの手順に従ってください。インテル® CAS インストールに失敗した場合は、カスタマーサポートにお問い合わせください。

4.2 ライトスルー・モードの設定

ライトスルー・モードでは、キャッシュ・ソフトウェアはデータをフラッシュデバイスに書き込み、同時に同じデータをコアデバイス (ディスクドライブ) に書き込み (ライトスルー) ます。ライトスルーはコアデバイスが 100% 同期され、そのデータがそのストレージを共有する他のサーバーに常に使用可能になることを確実にします。ただし、このタイプのキャッシュは読み取り集中の操作のみを高速化します。

1. コアデバイス (*/dev/sdb*) がマウントされていない、およびキャッシュデバイス (*/dev/sdc*) がマウントされておらず保存するデータも含んでいないことを確認してください。次のコマンドを入力すると、すべてのマウントポイントが表示されます：

   ```
   # mount
   ```

2. “1” の ID で新しいキャッシュを開始します：

   ```
   # casadm –S –i 1 –d /dev/sdc
   ```

   casadm -S コマンドを入力後、短い遅延に気づくことがあるかもしれません。通常これは 60 秒未満ですが、長引くこともあります。

   キャッシュデバイスがフォーマットされているか、ファイルシステムが既に存在する場合は “-f” 強制フラグ (例：*casadm -S -d /dev/sdc -f*) を使用する必要があります。

   注： -f オプションを使用すると、キャッシュデバイス上のすべての情報が削除されます。すべてのデータが別のデバイスにバックアップされていることを確認します (詳細については、40 ページの構成ツールの詳細 を参照してください)。

3. コアデバイスをこの新しいキャッシュとペアにします：

   ```
   # casadm -A –i 1 –d /dev/sdb
   ```

   Add-core コマンドは、次の名前形式で */dev* ディレクトリーに新しいデバイスを作成します：

   intelcas<cache ID>-<core #> 例：*/dev/intelcas1-1*

   この新しいデバイスは通常のブロックデバイスとして扱うことができます。

4.3 ライトバック・モードの設定

ライトバック・モードでは、キャッシュ・ソフトウェアはまずデータをキャッシュに書き込み、データがコアデバイスに書き込まれる前に書き込みが完了したことをアプリケーションに承認します。定期的に、これらの書き込みは好機をねらってディスクにライトバックされます。ライトバック・キャッシュは、書き込み集中と読み取り集中の両方の操作を向上する一方で、データがコアデバイスに書き込まれる前にキャッシュデバイスに障害が起きた場合はデータが失われるというリスクをもっています。

ライトバック・モードは、“*-c wb*” オプションを使用して新しいキャッシュデバイスを開始するときに有効になります：

```
# casadm –S –i 1 –d /dev/sdc –c wb
```

コアデバイスのペアリングは、前の ライトスルー・モードの設定 セクションに記載のステップ 3 に似ています。

4.4 ライトアラウンド・モードの設定

ライトアラウンド・モードでは、キャッシュ・ソフトウェアはそのブロックがキャッシュに既に存在する場合のみデータをフラッシュデバイスに書き込み、同時に同じデータをコアデバイス (ディスクドライブ) に書き込み (ライトスルー) ます。ライトアラウンドは、コアデバイスがキャッシュと 100% 同期していることと、このタイプのキャッシュが読み取り集中の操作のみを高速化するのを確実にすることにおいて、ライトスルーに似ています。ただし、ライトアラウンドは、データが書き込まれても時折後続的に再読取りされない場合にキャッシュへの害を防ぐためキャッシュをさらに最適化します。

ライトアラウンド・モードは、“*-c wa*”オプションを使用して新しいキャッシュデバイスを開始するときに有効になります：

```
# casadm -S -i 1 -d /dev/sdc -c wa
```

コアデバイスのペアリングは、ライトスルー・モードの設定 セクションに記載の 3 に似ています。

4.5 パススルーモードの設定

パススルーモードでは、キャッシュ・ソフトウェアはすべての操作でキャッシュをバイパスします。これにより、ユーザーは実際にキャッシュを有効にする前に、希望するすべてのコアデバイスをキャッシュするために関連付けすることができます。コアデバイスを関連付けると、希望するキャッシュモードに動的に切り替えられます (詳細については、42 を参照してください)。

パススルーモードは、“*-c pt*”オプションを使用して新しいキャッシュデバイスを開始するときに有効になります：

```
# casadm -S -i 1 -d /dev/sdc -c pt
```

コアデバイスのペアリングは、ライトスルー・モードの設定 セクションに記載の 3 に似ています。

4.6 自動パーティション・マッピング

高速化するコアデバイスに既存のパーティションがある場合は親デバイスを高速化するコアデバイス (例：上記の例では /dev/sdc) として選択すると、単一のコマンドで、基礎をなすすべてのパーティションが高速化されます。

インテル® CAS は自動的に既存のコアデバイスのパーティション (例：/dev/sdc1、/dev/sdc2 など) をオペレーティング・システムから非表示にして、エクスポートされたデバイス (例：/dev/intelcas1-1p1、/dev/intelcas1-1p2 など) でパーティションを作成して、エクスポートされたデバイスをコアデバイスの子デバイスにします。

表 5： インテル® CAS 高速化前と後の論理デバイスレイアウトの比較

前	後
/dev/sdc /dev/sdc1 /dev/sdc2 /dev/sdc3 /dev/nvme0n1	/dev/sdc /dev/intelcas1-1 /dev/intelcas1-1p1 /dev/intelcas1-1p2 /dev/intelcas1-1p3 /dev/nvme0n1

このスキームは、一般的なソフトウェア・デファインド・ストレージ・システム (Ceph* など) からの既存のオブジェクトのストレージデバイスの作成スクリプトおよびアクティベーション・スクリプトとの互換性を確保します。

4.7 インテル® CAS デバイスをマウントする

キャッシュデバイスとコアデバイスのペアリングの設定が完了したら、次の手順に従います：

1. casadm -list-caches コマンドを使用して設定を確認します：

   ```
   # casadm -L
   ```

2. コアデバイスに既にファイルシステムがある場合は、ステップ 3 に進みます。そうでない場合は、デバイスをマウントするために、ファイルシステムを設定する必要があります。*mkfs* コマンドを使用してインテル® CAS デバイスでファイルシステムを作成する例を以下に示します：

   ```
   # mkfs -b 4096 -t ext3 /dev/intelcas1-1
   ```

   ```
   # mkfs.ext4 -b 4096 /dev/intelcas1-1
   ```

   ```
   # mkfs.xfs -f –i size=2048 –b size=4096 –s size=4096 /dev/intelcas1-1
   ```

3. 新しいインテル® CAS デバイスをマウントします：

   a. 新しいキャッシュデバイスをマウントするには、ディレクトリーが必要です。この例では、マウントポイントは */mnt/cache1* です：マウントポイント・ディレクトリーがない場合は作成してください。

      ```
      # mkdir -p /mnt/cache1
      ```

   b. ディレクトリーをマウントします：

      ```
      # mount /dev/intelcas1-1 /mnt/cache1
      ```

   注： アプリケーションに変更が必要でないことをインテル® CAS が確認すると、前に使用したコアデバイスと同じファイルシステムのマウントポイント (例：*/local/data*) を使用できます。

   注： アプリケーションが直接ローデバイスを使用する場合 (たとえば、いくつかの Oracle* インストール) は、この例も代わりにアプリケーションがエクスポートされたデバイス (例：intelcas1-1) を使用するように設定する必要があります。

   アプリケーションがキャッシュにアクセスできるよう、マウントポイントの許可を変更する必要が生じることがあります。以下の例では、“bob” というユーザーにマウントポイントへの管理者権限を与える 2 つのコマンドを示します：

   ```
   # chown bob:adm /mnt/cache1 –R
   ```

   あるいは

   ```
   # chmod  g+rw  /mnt/cache1
   ```

5 インテル® CAS を実行する

この章では、インテル® CAS で一般的な管理操作を実行する方法を説明します。

5.1 再起動と電源入れ直し

インテル® CAS デバイスは、システムを再起動または電源を入れ直ししたあとに自動的に使用可能になりません。コアストレージ・デバイスとキャッシュ・デバイスのキャッシュのペアリングを再度作成する必要があります、各システムの再起動または電源入れ直しの際には、次の手順を使用してインテル® CAS キャッシュデバイスの使用を続行してください。

1. 次のコマンドを使用してインテル® CAS カーネルモジュールが正しく読み込まれたことを確認します：

   ```
   $ service intelcas status
   ```

   intelcas モジュールが読み込まれた場合は、ステップ 2 に進みます。モジュールが正しく読み込まれなかった場合は、インテル® CAS のすべてのコンポーネントがインストールされているか確認してください。たとえば、インストールファイルの *--list* オプションを使用できます：

   ```
   # ./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run --list
   ```

   いずれかのコンポーネントが欠落している場合は、インテル® CAS をアンインストールして、もう一度インストールする必要があります (詳細については、11 ページの インテル® CAS のインストール を参照してください)。

2. デバイスがシステムの再起動または電源の入れ直し前と同じであることを確認してください。

   ディレクトリーを */dev/disk/by-uuid* または */dev/disk/by-id* のいずれかに変更します。

   ls -l または *ll* コマンドを使用して物理デバイスとそれらの識別子の関連を表示します。

   デバイス/パーティションの識別子が正しい物理デバイスに関連付けられていることを確認します。キャッシュ SSD の識別子が認識されているのを確認することは極めて重要です。誤ったデバイスを使用すると、システムの動作が不確定になり、データが失われる可能性があります。

3. 次のコマンド構文を使ってインテル® CAS を開始します：

   a. インテル® CAS をライトスルー・モードで開始するには、次のコマンドを使ってキャッシュのペアを有効にします：

      ```
      # casadm -S –i <cache_id> -d <cache_device>
      ```

      ```
      # casadm -A –i <cache_id> -d <core_device>
      ```

   b. インテル® CAS をライトバック・モードで開始するには、次のコマンドを使ってキャッシュのペアを有効にします：

      ```
      # casadm –S –i <cache_id> -d <cache_device> -c wb
      ```

      ```
      # casadm -A –i <cache_id> -d <core_device>
      ```

   c. インテル® CAS をライトアラウンド・モードで開始するには、次のコマンドを使ってキャッシュのペアを有効にします：

      ```
      # casadm –S –i <cache_id> -d <cache_device> -c wa
      ```

      ```
      # casadm -A –i <cache_id> -d <core_device>
      ```

d. インテル® CAS をパススルーモードで開始するには、次のコマンドを使ってキャッシュのペアを有効にします：

```
# casadm –S –i <cache_id> -d <cache_device> -c pt
```

```
# casadm -A –i <cache_id> -d <core_device>
```

オプションで、前回のシャットダウンの前に前のコンテキストを保持してキャッシュを開始する方法があります。このオプションは、キャッシュをウォームアップする時間を縮減します。これは、*--load* パラメーターを使用してキャッシュデバイスを開始することによって実行できます。たとえば、以下のようなことが可能になります。

```
# casadm –-start-cache –-cache-id <cache_id> --cache-device <cache_device> /
      --load
```

あるいは

```
# casadm –S –i <cache_id> -d <cache_device> –l
```

注意： 前回のシャットダウンが次のセクション インテル® CAS を停止する の説明に従っていることを確認する必要があります。キャッシュを有効にする前にコアデータに何らかの変更があった場合、データは正しく同期されず損傷します。

4. インテル® CAS デバイスを引き続き通常に使用するためにマウントします：

```
# mount /dev/intelcas1-1 /mnt/cache1
```

インテル® CAS デバイスを使用する準備が整いました。

5.2 再起動時にキャッシュを自動的に有効にする

起動時にインテル® CAS ソフトウェアを自動的に開始するには、16 ページの ライトスルー・モードの設定 に記載されているように、必要な *casadm* コマンドを実行するためにスクリプトを追加して *intelcasx-x* デバイスをマウントします。アプリケーションを自動起動する詳細については、インテル® CAS のマンページと Web Wiki を参照してください。

インテル® CAS インストール・パッケージには *intelcas* という初期化スクリプトが含まれます。このスクリプトは、*/etc/intelcas* にある構成ファイルで定義されたすべてのキャッシュデバイスを作成します。このスクリプトを手動で実行するには、次のコマンドを入力します：

```
# service intelcas start
```

注： キャッシュを自動的に開始してコアデバイスを関連付けたい場合は、上記のスクリプトを実行するための構成ファイルが必要です。詳細については、28 ページの 構成ファイルとセットアップ・ファイルの使用 (詳細設定) を参照してください。このスクリプトは、構成ファイルをテストするときにも役立ちます。

次の項目を検討する必要があります：

- インテル® CAS デバイスまたはマウントポイントに依存するあらゆるアプリケーションを起動する *前に* インテル® CAS を開始する必要があります。
- デバイスのサイズによっては、起動にかかる時間が大幅に増加します。進捗インジケーターは表示されず、システムがハングしているか応答していない状態に見えることがあります。

5.3 インテル® CAS を停止する

システムをシャットダウンまたは再起動する前に、インテル® CAS をクリーンに停止することを推奨します。

システムのシャットダウンでインテル® CAS をクリーンに停止するには、*intelcas* デバイスをアンマウントするスクリプトを追加して *casadm -T* コマンドを続けます。

インテル® CAS デバイスへのアクセスを試行するあらゆるアプリケーションの*後に*インテル® CAS を停止する必要があります。キャッシュにダーティーデータがある場合、キャッシュのシャットダウン中に、データが自動的にディスクに転送されます。

次の例では *cache_id = 1* を使用します。インテル® CAS を停止するには、root としてログインして実行します：

```
# umount /mnt/cache1
# casadm -T -i 1
```

キャッシュが正常に停止したことを再確認するには、キャッシュがすでにリストに示されていないことを確認します：

```
# casadm -L
```

コマンドライン・オプションについては、40 ページから始まる 構成ツールの詳細 を参照してください。

注： many-to-one (多対一) の関連を削除するには、*casadm -T -i 1* を使用して、<cache_id> = “1” と関連付けられているすべてのキャッシュとコアを停止することもできます。

さらに、インテル® CAS インストール・パッケージには、シャットダウン時にすべてのキャッシュを削除するスクリプトが含まれています。このスクリプトを手動で実行するには、次のコマンドを入力します：

```
# service intelcas stop
```

5.4 予定外のシャットダウンの処理

予定外のシャットダウンとは、インテル® CAS デバイスが前のセクションに記述されたように正しくシャットダウンしないときを示します。これには、インテル® CAS デバイスを正しくシャットダウンせずに行ったシステムの電源入れ直し、システムの電源喪失が含まれます。

予定外のシャットダウン後に、キャッシュを再度開始するには 2 つのオプションがあります。最初のオプションは、キャッシュをリカバリーモードで開始することです。このモードでは、現在キャッシュにあるすべてのダーティーデータをフラッシュしてから、空のキャッシュを使用してキャッシュを再度開始します。2 番目のオプションは、新しいメタデータを使用してキャッシュを再度初期化することです。これはキャッシュをクリアして、現在キャッシュにあるすべてのダーティーデータをなくします。

注意： キャッシュが実行されていないときにファイルシステムが変更されると、データの損失が起きる可能性があります。以下のいずれかの方法を使用してキャッシュが再度開始されるまで、コアデバイスに IO を送らないでください。

5.4.1 キャッシュのリカバリー

リカバリーを有効にしてインテル® CAS を開始するには、次のコマンドを入力します：

```
# casadm -S -d /dev/sdc -r
```

```
# casadm -S -d /dev/sdc
```

```
# casadm -A -i 1 -d /dev/sdb
```

詳細については、40 ページの「-S | --start-cache」を参照してください。

5.4.2 キャッシュを再度初期化する

キャッシュをクリアする (キャッシュ全体を無効にする) には、次の手順に従います：

1. root としてログオンします。
2. 以下に示すように、-f パラメーターを含めて (-l パラメーターは除外) キャッシュを開始します：

```
# casadm –S -d /dev/sdc –f
# casadm -A -i 1 -d /dev/sdb
```

これは、古い状態を読み込む代わりにキャッシュを再度初期化します。

新しいメタデータを使用してキャッシュを再度初期化すると、まだディスクにフラッシュされていないすべての書き込みキャッシュデータが破棄されます。

インテル® CAS は再起動全体で状態を保持します。キャッシュ・サブシステムを開始せずにシステムが再起動される場合、キャッシュデータの状態がプライマリー・ストレージ (コアデバイス) にあるデータと同期しなくなることがあります。この場合、キャッシュをクリアせずにキャッシュシステムを再度開始すると、データが損傷する可能性があります。

5.5 デバイス I/O エラー処理

キャッシュデバイス (SSD) またはコアデバイス (HDD) から読み取りまたは書き込み I/O エラーが起きた場合、インテル® CAS はインテリジェントにエラーを処理し、可能な場合は要求されたデータを返し、特定の場合においては I/O の処理を続行します。

インテル® CAS が ***データを返して I/O の処理を続行***する場合、次を試行するときに ***キャッシュデバイス I/O エラー*** が含まれます：

- キャッシュからクリーンデータを読み込む
- 読み取り操作でキャッシュにデータを書き込む (読み取られたデータをキャッシュすることを試行する)
- 書き込み操作でライトスルーまたはライトアラウンド・モードでキャッシュにデータを書き込む

インテル® CAS が呼び出し元のアプリケーションに ***エラーを返し***、I/O エラーが発生したキャッシュ SSD によって使われるエクスポートされたいずれかのデバイス (例：Intelcas1-1) への ***I/O 理を停止***した場合、次を試行するときに ***キャッシュデバイス I/O エラー*** が含まれます：

- キャッシュからダーティーデータを読み取る
- ライトバック・モードでキャッシュにデータを書き込む

インテル® CAS が呼び出し元のアプリケーションに ***エラーを返して I/O の処理を続行***する場合、次を試行するときに ***キャッシュデバイス I/O エラー***が含まれます：

- コアデバイスからデータを読み取る
- コアデバイスにデータを書き込む

6 IO 分類

6.1 IO クラス構成

インテル® CAS には、今までよりさらに細分化してデータのキャッシュを制御する機能があります。インテル® CAS Linux* 版は各 IO を即座に分析し、要請されたブロックがファイルシステムのメタデータであるかデータであるかを判別し、データである場合は、要求されたファイル IO の合計サイズを決定できます。この情報を使用して管理者は一般的な作業負荷に対する最善の IO クラス構成の設定を決定し、どの IO クラスをキャッシュに設定し、どの IO クラスを設定しないか、および各 IO クラスの優先レベルを設定できます。キャッシュラインを排除する必要になると、インテル® CAS は最初に、使用できる最も低い優先度のキャッシュラインを排除します (従来の LRU 排除に比べて改善)。

1. IO 分類と選択的な割り当てを有効にするには、最初に、提供された例の IO クラス構成ファイルを見てニーズに合わせて編集します：

   ```
   # vi /etc/intelcas/ioclass-config.csv
   ```

   例の IO クラス構成ファイル形式は次のようになります：

表 6： IO クラス構成ファイルフィールド

フィールド	詳細
IO クラス ＩＤ	IO クラスの固有 ID。このフィールドの値は固定されているので変更しないでください。(例：メタデータの ID は常に 1 でなければなりません)
IO クラス名	IO クラスの文字列名。ユーザーは IO クラスを表すのに任意の文字列を使用できますが、特定のクラスの意味は同じに保たれます (例：IO クラス ID 1 は、このフィールドに割り当てられた文字列に関わらず、常にメタデータを表します)。
排除の優先度	IO クラスの優先番号。排除が行われると、最初に使用可能な最も優先度が低い IO クラスからキャッシュラインが排除されます。
割り当て	この IO クラスのデータをキャッシュするかどうかをユーザーが決められるブール値。0=キャッシュしない、1=キャッシュする。

```
IO class id,  IO class name,  Eviction priority,  Allocation
0,            Unclassified,   22,                 1
1,            Metadata,       0,                  1
11,           <=4KiB,         10,                 1
12,           <=16KiB,        11,                 1
13,           <=64KiB,        12,                 1
14,           <=256KiB,       13,                 1
15,           <=1MiB,         14,                 1
16,           <=4MiB,         15,                 1
17,           <=16MiB,        16,                 1
18,           <=64MiB,        17,                 1
19,           <=256MiB,       18,                 1
20,           <=1GiB,         19,                 1
21,           >1GiB,          20,                 1
22,           O_DIRECT,       21,                 1
23,           Misc,           22,                 1
```

2. IO クラス構成への変更を完了してファイルを保存したら、ファイル <FILE> から <ID> で表される ID 番号を使用してキャッシュデバイスの設定を読み込む必要があります：

```
# casadm –-io-class –-load-config –-cache-id <ID> –f <FILE>
```

3. 構成ファイルが正常に読み込まれたことを確認します：

```
# casadm –-io-class --list --cache-id <ID>
```

7 詳細オプション

7.1 Many-to-one (多対一) オプション

1 つのキャッシュデバイスに複数のコアデバイスを追加できます。

1. 次のコマンドを入力して既存のキャッシュ・フレームワーク (<cache_id> = “1” に関連) にさらにコアデバイスを追加します：

   ```
   # casadm -A -i 1 -d /dev/sd<n>
   ```

   <n> は固有のコアドライブ/パーティションの文字です。(例：*casadm -A -i 1 -d /dev/sdd*)

2. Many-to-one (多対一) キャッシュ・フレームワークに追加する固有の各コアドライブにステップ 1 を繰り返します。

   追加する新しい各コアデバイスに、次の命名形式を使用してキャッシュとコアのペアが作成されます：

   <u>/dev/intelcas1-1</u> は最初のコアデバイス *<cache_id> =1* に関連付けられて作成されます。
   (例：*/dev/sdd*)

   /dev/intelcas1-2 は 2 番目のコアデバイス *<cache_id>=1* に関連付けられて作成されます。
   (例：*/dev/sdf*)

   など続きます。

3. 次のコマンドのいずれかを使用して各デバイスの作成を確認します：

   ```
   # ls /dev/intelcas<cache_id>-<core_id>
   ```

   あるいは

   ```
   # casadm -L
   ```

   あるいは

   ```
   # casadm -P –i 1
   ```

4. (オプション) 1 つのファイルまたはファイルのリストをキャッシュするには、-I 引き数を使用します。

   ```
   # casadm -I -f /mnt/cache1/info.db
   ```

新しく作成されたキャッシュデバイスでの連続的な読み取りなど、場合によっては、キャッシュデバイスがオーバーロードして、パフォーマンスが遅くなり、システムバッファを消費してキューが増加することがあります。この問題を防ぐには、使用事例に特有のキャッシュ書き込みキュー向けに最適なキューのサイズを手動で強制する必要が生じることがあります。デフォルトの *max_writeback queue_size* の値は 65,536 です。このキューのブロック解除しきい値を判別するために、別のパラメーターを使うこともできます：*writeback_queue_unblock_size* (デフォルト値は 60,000 に設定されています)。次のコマンドを使用してデフォルト値を上書きし、*max_writeback queue_size* を設定するか *writeback_queue_unblock_size* しきい値を変更できます：

```
# modprobe intelcas max_writeback_queue_size=<size>
writeback_queue_unblock_size=<size>
```

あるいは

```
# insmod intelcas max_writeback_queue_size=<size>
writeback_queue_unblock_size=<size>
```

注： 上の例は各一行です。

システムでインテル® CAS が有効になり、高速化されるファイルを */mnt/cache1* に書き込むことができます。

7.2 マルチレベル・キャッシュ

インテル® CAS Linux* 版はマルチレベル・キャッシュをサポートします。たとえば、ユーザーは遅い HDD メディアからもっと速い SSD メディアにウォームデータをキャッシュして、SSD からホットデータを RAMdisk などのさらに速いメディアにキャッシュすることができます。この場合、DRAM の固定された部分がバッファに割り当てられ、SSD は完全に包含的キャッシュとして保持されるので、DRAM キャッシュのデータは常に SSD で使用できます。.

RAMdisk ベースのキャッシュレベルを設定するには、次の手順に従います：

1. RAMdisk を作成します (40 MB 以上)。

   たとえば RHEL* では、デフォルトの RAMdisk サイズは 8 MB なので、それを 40 MB 以上に設定する必要があります。

   a. 3.0 より前のカーネルでは *grub.conf* を変更し *ramdisk_size* パラメーターを希望する容量 (KiB) に変更します。以下に例が**赤**でハイライトされています：

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/hda5
#          initrd /initrd-version.img
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Hat Linux (2.4.20-20.9)
           root (hd0,0)
        kernel /vmlinuz-2.4.20-20.9 ro root=LABEL=/ hdc=ide-scsi
        ramdisk_size=40000
        initrd /initrd-2.4.20-20.9.img
```

   b. 3.0 以降のカーネルでは、次のコマンドを入力して RAMdisk のサイズを 40 MB に増やします。

```
# modprobe brd rd_size=40000
```

注： ここでは、ファイルシステムをフォーマットまたは作成したり RAMdisk をマウントしないでください。

注： */dev/ram0* はカーネルによって予約されているため、使用しないでください。使用可能な RAMdisk */dev/ram1* 以降を使用してください。この例では */dev/ram1* を使用します。

2. システムを再起動して、必要な場合はインテル® CAS のフレッシュ・インストールを行います。
3. SSD キャッシュ・フレームワークを作成します。*/dev/sdc* は使用する SSD キャッシュデバイスです：

   ```
   # casadm -S -d /dev/sdc
   ```

4. SSD キャッシュ・フレームワークにコアデバイス (*/dev/sdb*) マッピングを追加します：

   ```
   # casadm –A -i 1 -d /dev/sdb
   ```

5. RAMdisk キャッシュ・フレームワークを作成します：

   ```
   # casadm –S –d /dev/ram1 –i 2
   ```

6. RAMdisk キャッシュ・フレームワークに階層型コアデバイス (*/dev/intelcas1-1*) マッピングを追加します：

   ```
   # casadm –A -i 2 –d /dev/intelcas1-1
   ```

7. 新しく作成された階層型キャッシュドライブからファイルシステムを作成します：

   ```
   # mkfs -b 4096 –t ext3 /dev/intelcas2-1
   ```

   あるいは

   ```
   # mkfs.ext4 -b 4096 /dev/intelcas2-1
   ```

   あるいは

   ```
   # mkfs.xfs -f –i size=2048 –b size=4096 –s size=4096 /dev/intelcas2-1
   ```

8. キャッシュ・ディレクトリーを作成して、そこにキャッシュをマウントします：

   ```
   # mkdir –p /mnt/cache1
   # mount /dev/intelcas2-1 /mnt/cache1
   ```

新しく作成されマウントされたキャッシュドライブ */mnt/cache1* を使用する準備ができました。

デフォルトでは RAMdisk 階層型バッファリング設定はどれも揮発性で、システムの再起動後にもう一度設定する必要があります。クリーンな再起動後にあらゆるデータが SSD に保持されるはずです。インテル® CAS の停止の詳細については、20 ページの インテル® CAS を停止する を参照してください。

注意： RAMdisk キャッシュ階層がライトバック・モードで開始され、ダーティー・シャットダウンが起きた場合は、データが失われることがあります。

7.3 Linux* LVM サポート

インテル® CAS は 2 つの方法で LVM をサポートしています。

1. LVM 物理ボリュームはインテル® CAS デバイスに直接作成できます。これは、その上にボリュームグループを作成できるようにして、後でそれらのグループで論理ボリュームを作成できます。

   注： LVM の古いバージョンでは、インテル® CAS デバイスでそのデバイスの高速化を行う前にまずコアデバイスでパーティションを作成する必要があり (例：コアデバイスが /dev/sdc である場合は /dev/sdc1 を作成する必要があります)、物理ボリュームの作成または論理ボリュームの作成は失敗します。*「警告：重複構成ノード：タイプ(タイプを検索、重複したパーティション・ボリュームが見つかりました」*という警告が表示された場合は、この回避策を使用する必要があります。

2. LVM 論理ボリュームは、通常のパーティションのように、コアデバイスとして使用できます。

インテル® CAS で使用する前にシステムで LVM を設定する必要があり、その作業は本書の範囲外です。

詳細設定に次の行を追加して、*/etc/lvm/lvm.conf* に インテル® CAS デバイスが許可されるブロックデバイスのタイプとして示されていることを確認します：

```
# Advanced settings.
# List of pairs of additional acceptable block device types found
# in /proc/devices with maximum (non-zero) number of partitions.
types = [ "inteldisk", 16 ]
```

LVM が設定されインテル® CAS デバイスが作成されたら (詳細については、15 ページから始まる インテル® CAS の設定 を参照してください)、その上に物理ボリュームを作成できます：

pvcreate /dev/intelcas1-1 *(または、インテル® CAS デバイスでパーティションから物理ボリュームを作成する場合は、#pvcreate /dev/intelcas1-1p1)*

物理ボリュームが存在すると、ユーザーはそのボリューム上にボリュームグループ (例：*vg_cas*)、そして論理ボリューム (例：*lv_cas*) を作成できます：

```
# vgcreate vg_cas /dev/intelcas1-1
# lvcreate -n lv_cas -l 100%FREE vg_cas
```

LVM ファイルシステムを作成して “*vfs_cas*” にマウントします：

```
# mkfs.ext4 <-b 4096> /dev/vg_cas/lv_cas
# mkdir –p /mnt/vfs_cas
# mount /dev/vg_cas/lv_cas /mnt/vfs_cas
```

LVM デバイスを削除するには、次の手順に従います：

```
# umount /mnt/vfs_cas
# vgchange –an vg_cas
# casadm -R -i<cache_id> -j <core_id>
# casadm -T -i 1
```

動的論理ボリュームの縮小または拡張は自動的にはサポートされていません (例：*vgreduce* または *vgextend*)。ユーザーはインテル® CAS デバイス (例：*intelcas1-1*)) を物理ボリュームとして削除し、必要なサイズ変更を行ってからインテル® CAS デバイスに基づいて物理ボリュームを再度作成する必要があります。LVM の使用と管理方法の詳細については、インテル® CAS LVM マンページと Web の Wiki を参照してください。

7.4 構成ファイルとセットアップ・ファイルの使用 (詳細設定)

構成ファイルとセットアップ・ファイルを使用してインテル® CAS の設定を行えます。すべてのファイルは */etc/intelcas* ディレクトリーに保存する必要があります。

/etc/intelcas ディレクトリーにはフレッシュインストール後サンプルファイルが含まれます。このセクションで説明する手順を実行する前に、それらを確認してください。

/etc/intelcas ディレクトリーには次のファイルが存在します：

- intelcas.conf - キャッシュドライブとコアドライブの一致情報が含まれます。
- include.conf - 排他的にキャッシュされるために含まれるファイルのリストが含まれます。

installation_setup_file というスクリプトを作成して使用し、ローカル構成ファイルを使用してリモートインストールと設定を自動化することもできます。

7.4.1 セットアップ・ファイルと構成ファイルの形式

installation_setup_file スクリプトでは、次の形式と構文が使用されます：

```
# Comments begin with hash symbol
config: </etc/intelcas/intelcas.conf> </etc/intelcas/include.conf>
<IP Address of system A>
<IP Address of system B>
<IP Address of system N>

# Empty lines are ignored
```

installation_setup_file の例：

```
# INSTALL_SETUP_FILE
# Sample installation_setup_file for nodes 192.168.1.101 through
# 192.168.1.112

config: /etc/intelcas/intelcas.conf /etc/intelcas/include.conf
192.168.1.101
192.168.1.102
...
192.168.1.112
```

特定のグループのマシンにコピーされる各構成ファイルはそのグループの config: 行にスペースで区切って示される必要があります。特定のグループにアドレスが示されるすべてのマシンは、グループヘッダーに示される構成ファイルを受け取ります。グループの最後に、新しい構成グループヘッダーまたはファイルの終わりが示されます。

intelcas.conf ファイルの形式と構文：

```
# Make sure that devices listed below exist and correspond to proper roles.

# Cache configuration section
[caches]
# Cache ID    Cache device            Load      Mode             Extra flags
<cache_id>    /dev/disk/by-uuid/<uuid>   <yes,no>  <WT,WB,WA,PT>
   ioclass_file=<file>,cleaning_policy=<alru,nop>
#Core Device Configuration
 [cores]
# Cache ID    Core device             Extra flags
<cache_id>    /dev/disk/by-uuid/<uuid>   core_id=<core_id>
```

各フィールドの説明を以下に示します：

- *<cache_id>* は 1 から 16,384 までの数値です (有効なキャッシュ・インスタンスの数字)。
- キャッシュデバイスとコアデバイスは既存の SSD デバイスを指していなければなりません。理想的には、UUID により参照されるようにします (インテル® CAS のコマンド ‘ls -l /dev/disk/by-uuid’ と同様)。または、デバイス名 (例：/dev/sdb) によってデバイスを参照できます。
- 読み込みでは次の値が使用されます (大文字と小文字が区別されます)：
    - *“Yes”* - キャッシュデバイスのメタデータが起動時に読み込まれます (すなわち、キャッシュデバイスから読み取られます。予定外のシャットダウンが起きた場合に危険なので推奨されません)。
    - “*No*” - キャッシュデバイスのメタデータが起動時に再度初期化されます (すなわち、キャッシュデバイスからは読み取れません)。
- Mode は、ライトスルー、ライトバック、ライトアラウンド、パススルーのいずれかのキャッシュモードを決定します。
- *Optional:* 追加のフラグはキャッシュデバイスとコアデバイスのその他の設定を許可し、これらはコンマで区切られます。

- *ioclass_file* ユーザーがこのキャッシュの IO クラスポリシーを指定するファイルを読み込むのを許可します。
- *cleaning_policy* は、ユーザーがこのキャッシュで使用されるキャッシュのクリーニング・ポリシー (alru または nop) を指定するのを許可します。
- *<core_id>* は 1 から 16,384 までの数値です (有効なコア・インスタンスの数字)。

intelcas.conf の例。

```
# Cache configuration section
[caches]
# Cache ID    Cache device  Load      Mode      Extra flags
  1       /dev/sdc1 no     WT
  2       /dev/sdd      no     WB     ioclass_file=ioclass_config.csv

#Core device configuration
[cores]
# Cache ID    Core device   Extra flags
  1       /dev/sde1
  2       /dev/sdf1 core_id=7
```

include.conf ファイルの形式と構文：

```
# Intel® CAS file inclusion list

# List of files delimited by a new line
/mnt/cache/file1
/mnt/cache/file2
/mnt/storage/docs/doc1.txt
```

include.conf の例：

```
# Cache 1
# Files from core 1
/usr/local/phonebook.db
/usr/doc/database.db

# Files from core 2
/home/user/webpage.cgi-bin
/home/user/webpage.html
```

7.4.2 Installation_setup_file を使用してインストールする

リモート・インストール・ユーティリティ (54 ページの リモート・インストール を参照してください) は、以前に作成された 構成ファイルを適切なリモートターゲットのシステムに自動的に (*ssh* 経由で複数のシステムの IP アドレスに) コピーすることにより、複数のターゲットシステムでのバッチ設定をサポートします。.

注： リモート・インストール・ユーティリティは、リモートマシンが *ssh* の接続とインストール・システム/サーバーからのファイル転送をサポートすることを必要とします。

設定の手順は次のようになります：

注： 同じ installation_setup_file がインストールと設定の両方 (およびアンインストールにも) に使用されます。

注： インストールとセットアップは別々のステップで 1 つだけ実行することができます (すなわち、ターゲットシステムにインテル® CAS がインストールされていない場合でも、ターゲットシステムを事前設定できます)。

1. 54 ページの リモート・インストール に記載されているすべてのステップを完了後、次の手順に従って 1 つのターゲットシステムに 1 つずつ構成ファイルを配置します：

```
# ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
 --setup --one-machine <IP ADDRESS> </etc/intelcas/intelcas.conf>
```

および/または

```
# ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
 --setup --one-machine <IP ADDRESS> </etc/intelcas/include.conf>
```

2. または、*installation_setup_file* スクリプトに上記の 2 つのステップを含めて次のコマンドを入力することにより、上の同じ 2 つのステップを実装できます：

```
# ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
 --setup --use-config <path>/installation_setup_file
```

注： インストールの構成ファイルのグループヘッダーに示すファイルがファイルシステム内の既存のファイルを指していることを確認してください。そうしないと、それらのファイルがコピーされません。

注： インテル® CAS がシステムに正しくインストールされ設定されたら、そのシステムを再起動する前に構成ファイルが有効な情報を含んでいることを確認します。

注： 参考までに、インストール・パッケージには、この章に記載されている各セットアップ・ファイルと構成ファイルの例を示す *<filename>.example* という例のファイルが */etc/intelcas* ディレクトリーに含まれます (標準インストールプロセスの一部として、そのディレクトリに配置されます。ユーザーが行う追加手順は不要です)。これらのファイルで、正しい構文と形式の詳しい例を参照してください。

8 インテル® CAS のモニタリング

多くのパフォーマンス・カウンターを表示できます。これらのカウンターは、casadm -P -i <cache_id> コマンドを入力してアクセスできます。このセクションは、*stats* コマンドライン・オプションに含まれるデータの概要を示します。

```
# casadm –-stats –-cache-id <ID>
```

あるいは

```
# casadm –P -i <ID>
```

詳細については、44 ページの「-P | --stats」 セクションを参照してください。

このコマンドの出力は、キャッシュ・システムのアクティビティを説明する 3 つの表を含みます：

- **使用状況の統計**
- **リクエストの統計**
- **ブロックの統計**

エントリーは、統計、実際の数字、総合のパーセント、単位の形式になります。

これらの統計を使って、データの使用について知ることができます。たとえば、連続読み取りとランダム読み取りの統計を見ると、データがどのように使用されているかわかります。

次の表にインテル® CAS が記録する統計 (カウンター) を示します：

表 7： 使用状況の統計

統計	詳細
占有率	キャッシュされたブロック数
空き	キャッシュの空白ブロック数
クリーン	クリーンブロック数 (キャッシュデータがコアデータと一致)
ダーティー	ダーティーブロック数 (キャッシュに書き込まれたがまだコアデータと同期されていないブロック)

表 8： リクエストの統計

統計	詳細
読み取りヒット	キャッシュヒットだった読み取り数
読み取り一部ミス	キャッシュにあったデータとキャッシュになかったデータの両方に及ぶ読み取り数
読み取り完全ミス	キャッシュミスだった読み取り数
読み取り合計	読み取りの合計数
書き込みヒット	キャッシュヒットだった書き込み数
書き込み一部ミス	キャッシュにあったデータとキャッシュになかったデータの両方に及ぶ書き込み数
書き込み完全ミス	キャッシュミスだった書き込み数
書き込み合計	書き込みの合計数
パススルー読み取り	直接コアデバイスに送られた (インテル® CAS によってキャッシュされていない) 読み

統計	詳細
	取りリクエスト数。 キャッシュがパススルーモードのときにリクエストが処理されると、この統計は増分になります (詳細については、-Q | --set-cache-mode を参照してください)。
パススルー書き込み	直接コアデバイスに送られた (インテル® CAS によってキャッシュされていない) 書き込みリクエスト数。 キャッシュがパススルーモードのときにリクエストが処理されると、この統計は増分になります (詳細については、-Q | --set-cache-mode を参照してください)。
処理されたリクエスト	処理された (キャッシュをバイパスしなかった) IO リクエストの合計数
合計リクエスト数	IO リクエストの合計数 (処理されたリクエストとキャッシュをバイパスしなかったリクエストの両方)

パーセンテージは合計リクエストのパーセンテージとして計算されます。(例：読み取りヒット % = 100 * (読み取りヒット数/合計リクエスト数))。

表 9： ブロックの統計

統計	詳細
コアデバイスからの読み取り	コアデバイスからの読み取りブロック数
コアデバイスへの書き込み	コアデバイスへの書き込みブロック数
コアデバイスからの/コアデバイスへの合計	コアデバイスからの読み取りブロック数またはコアデバイスへの書き込みブロック数の合計
キャッシュからの読み取り	キャッシュデバイスからの読み取りブロック数
キャッシュへの書き込み	キャッシュデバイスへの書き込みブロック数
キャッシュデバイスからの/キャッシュデバイスへの合計	キャッシュデバイスからの読み取りブロック数またはキャッシュデバイスへの書き込みブロック数の合計
エクスポートされたオブジェクトからの読み取り	エクスポートされたオブジェクトからの読み取りブロック数 (例：intelcas1-1)
エクスポートされたオブジェクトへの書き込み	エクスポートされたオブジェクトの書き込みブロック数
エクスポートされたオブジェクトへの/エクスポートされたオブジェクトからの合計	エクスポートされたオブジェクトからの読み取りブロック数またはエクスポートされたオブジェクトへの書き込みブロック数の合計数

表 10： エラーの統計

統計	詳細
キャッシュの読み取りエラー	キャッシュデバイスへの読み取りエラー数
キャッシュの書き込みエラー	キャッシュデバイスへの書き込みエラー数
キャッシュの合計エラー	キャッシュデバイスへの読み取りまたは書き込みエラー数の合計
コアの読み取りエラー	コアデバイスへの読み取りエラー数
コアの書き込みエラー	コアデバイスへの書き込みエラー数
コアの合計エラー	コアデバイスへの読み込みまたは書き込みエラー数の合計
合計エラー	キャッシュデバイスまたはコアデバイスへの読み取りまたは書き込みエラー数の合計

8.1 キャッシュの統計の表示

キャッシュレベルの統計の使用例：

```
# casadm -P -i 1
```

返された出力：

```
Cache Id                 1
Cache Size               12055031 [4KiB Blocks] / 45.99 [GiB]
Cache Device             /dev/nvme0n1p1
Core Devices             3
Write Policy             wt
Eviction Policy          lru
Cleaning Policy          alru
Cache line size          4 [KiB]
Metadata Memory Footprint 639.1 [MiB]
Dirty for                0 [s] / Cache clean
Status                   Running
```

Usage statistics	Count	%	Units
Occupancy	12055001	100.0	4KiB blocks
Free	30	0.0	4KiB blocks
Clean	12055001	100.0	4KiB blocks
Dirty	0	0.0	4KiB blocks

Request statistics	Count	%	Units
Read hits	50532296	32.8	Requests
Read partial misses	202850	0.1	Requests
Read full misses	48537517	31.5	Requests
Read total	99272663	64.4	Requests
Write hits	31762847	20.6	Requests
Write partial misses	436	0.0	Requests
Write full misses	16197432	10.5	Requests
Write total	47960715	31.1	Requests
Pass-Through reads	2779876	1.8	Requests
Pass-Through writes	4027209	2.6	Requests
Serviced requests	147233378	95.6	Requests
Total requests	154040463	100.0	Requests

Block statistics	Count	%	Units
Reads from core(s)	471298381	57.4	4KiB blocks
Writes to core(s)	349624858	42.6	4KiB blocks
Total to/from core(s)	820923239	100.0	4KiB blocks
Reads from cache	140451806	16.5	4KiB blocks
Writes to cache	713000302	83.5	4KiB blocks
Total to/from cache	853452108	100.0	4KiB blocks
Reads from exported object(s)	611750187	63.6	4KiB blocks
Writes to exported object(s)	349624858	36.4	4KiB blocks
Total to/from exported object(s)	961375045	100.0	4KiB blocks

Error statistics	Count	%	Units
Cache read errors	0	0.0	Requests
Cache write errors	0	0.0	Requests
Cache total errors	0	0.0	Requests
Core read errors	0	0.0	Requests
Core write errors	0	0.0	Requests
Core total errors	0	0.0	Requests
Total errors	0	0.0	Requests

コアレベルの統計の使用例：

```
# casadm –P -i 1 –j 1
```

返された出力：

```
Core Id          1
Core Device      /dev/sdd1
Exported Object /dev/intelcas1-1
Core Size        36620800 [4KiB Blocks] / 139.70 [GiB]
Dirty for        0 [s] / Cache clean
```

Usage statistics	Count	%	Units
Occupancy	4363950	36.2	4KiB blocks
Free	1	0.0	4KiB blocks
Clean	4363950	100.0	4KiB blocks
Dirty	0	0.0	4KiB blocks

Request statistics	Count	%	Units
Read hits	21075939	29.7	Requests
Read partial misses	77109	0.1	Requests
Read full misses	18448305	26.0	Requests
Read total	39601353	55.9	Requests
Write hits	17240919	24.3	Requests
Write partial misses	105	0.0	Requests
Write full misses	9972336	14.1	Requests
Write total	27213360	38.4	Requests
Pass-Through reads	983305	1.4	Requests
Pass-Through writes	3053282	4.3	Requests
Serviced requests	66814713	94.3	Requests
Total requests	70851300	100.0	Requests

Block statistics	Count	%	Units
Reads from core	156393665	57.0	4KiB blocks
Writes to core	118061392	43.0	4KiB blocks
Total to/from core	274455057	100.0	4KiB blocks
Reads from cache	49888641	17.7	4KiB blocks
Writes to cache	231251909	82.3	4KiB blocks
Total to/from cache	281140550	100.0	4KiB blocks
Reads from exported object	206282306	63.6	4KiB blocks
Writes to exported object	118061392	36.4	4KiB blocks
Total to/from exported object	324343698	100.0	4KiB blocks

Error statistics	Count	%	Units
Cache read errors	0	0.0	Requests
Cache write errors	0	0.0	Requests
Cache total errors	0	0.0	Requests
Core read errors	0	0.0	Requests
Core write errors	0	0.0	Requests
Core total errors	0	0.0	Requests
Total errors	0	0.0	Requests

IO クラスレベルの統計の使用例：

```
# casadm -P -i 1 -j 1 -d
```

返された出力：

```
IO class ID          1
IO class name        Metadata
Eviction priority    0
Selective allocation Yes
```

Usage statistics	Count	%	Units
Occupancy	864660	7.2	4KiB blocks
Free	0	0.0	4KiB blocks
Clean	864660	100.0	4KiB blocks
Dirty	0	0.0	4KiB blocks

Request statistics	Count	%	Units
Read hits	8097835	11.4	Requests
Read partial misses	0	0.0	Requests
Read full misses	17	0.0	Requests
Read total	8097852	11.4	Requests
Write hits	414587	0.6	Requests
Write partial misses	104	0.0	Requests
Write full misses	273056	0.4	Requests
Write total	687747	1.0	Requests
Pass-Through reads	2	0.0	Requests
Pass-Through writes	5517	0.0	Requests
Serviced requests	8785599	12.4	Requests
Total requests	8791118	12.4	Requests

Block statistics	Count	%	Units
Blocks reads	8097854	3.9	4KiB blocks
Blocks writes	9007260	7.6	4KiB blocks

注： このコマンドは、定義された各 IO クラスに上記の形式を出力します。

IO クラスレベルの統計の使用例は csv 形式で出力されファイルに保存されます。

```
# casadm -P -i 1 -j 1 -d -o csv > stats.txt
```

返された出力：

```
IO class ID,IO class name,Eviction priority,Selective allocation,Occupancy [4KiB
blocks],Occupancy [%],Free [4KiB blocks],Free [%],Clean [4KiB blocks],Clean
[%],Dirty [4KiB blocks],Dirty [%],Read hits [Requests],Read hits [%],Read partial
misses [Requests],Read partial misses [%],Read full misses [Requests],Read full
misses [%],Read total [Requests],Read total [%],Write hits [Requests],Write hits
[%],Write partial misses [Requests],Write partial misses [%],Write full misses
[Requests],Write full misses [%],Write total [Requests],Write total [%],Pass-
Through reads [Requests],Pass-Through reads [%],Pass-Through writes
[Requests],Pass-Through writes [%],Serviced requests [Requests],Serviced requests
[%],Total requests [Requests],Total requests [%],Blocks reads [4KiB blocks],Blocks
reads [%],Blocks writes [4KiB blocks],Blocks writes [%]
0,Unclassified,22,Yes,0,0.0,97,100.0,0,0.0,0,0.0,0,0.0,0,0.0,82,0.0,82,0.0,0,0.0,0,
0.0,0,0.0,0,0.0,1,0.0,0,0.0,82,0.0,83,0.0,83,0.0,0,0.0
1,Metadata,0,Yes,864660,7.2,0,0.0,864660,100.0,0,0.0,8068949,11.4,0,0.0,17,0.0,8068
966,11.4,413037,0.6,104,0.0,273056,0.4,686197,1.0,2,0.0,5517,0.0,8755163,12.4,87606
82,12.4,8068968,3.9,8979669,7.6
```

注： このコマンドは、定義された各 IO クラスに上記の形式を出力します。

注： 前に参照された例はいずれも csv 形式で出力できます。

8.2 パフォーマンス・カウンターのリセット

パフォーマンス・カウンタ-は、キャッシュが開始されるたびに自動的にリセットされます。パフォーマンス・カウンタ-を手動でクリアするには、casadm -Z -i <cache_id> -j <core_id> コマンドライン・オプションを使用してください。

9 構成ツールの詳細

インテル® CAS 製品には、同キャッシュ・ソフトウェアの完全制御を提供するユーザーレベルの構成ツールが含まれます。このツールで使用できるコマンドとパラメーターは、本章で詳しく説明しています。

CLI からヘルプにアクセスするには、*-H* または *--help* パラメーターを入力して詳細を表示します。次のコマンドを入力して、この製品のマンページを表示することもできます：

```
# man casadm
```

9.1 -S | --start-cache

使用： casadm --start-cache --cache-device <DEVICE> [option...]

例：

```
# casadm --start-cache --cache-device /dev/sdc
```

あるいは

```
# casadm -S -d /dev/sdc
```

説明：他のブロックデバイスをキャッシュするためのデバイスとして使用するためにブロックデバイスを準備します。一般にキャッシュデバイスには SSD または他の NVM ブロックデバイスあるいは RAM ディスクを使用します。このプロセスは、特定キャッシュ ID に関するデバイスのマッピング向けたフレームワークを開始します。*-l* または *--load* パラメーターを使用して古い状態 (前のキャッシュのメタデータは無効としてマークされません) で、またはデフォルトで新しい状態 (前のキャッシュのメタデータは無効としてマークされます) で読み込めます。

***必要なパラメーター*：**

[-d, --cache-device <DEVICE>]：使用されるキャッシュ・デバイス。これは */dev* ディレクトリーに表示される SSD または NVM ブロックデバイスまたは RAM ディスクです。<Device> は使用するキャッシュ・デバイスを示す完全パスである必要があります。例：/dev/sdc

***オプション・パラメーター*：**

[-i, --cache-id <ID>]：作成するキャッシュ ID、<1 から 16384>。この ID は指定することができ、デフォルトではコマンドは最初に使用可能な最小値を使用します。

[-l, --load]：キャッシュ・デバイスから既存のキャッシュのメタデータを読み込みます。キャッシュデバイスが以前に使用されて無効になり (再起動した場合など)、キャッシュデバイスが使用されてからコアデバイスのデータが変更されていないことが判別された場合、このオプションは、データでキャッシュを再度ウォームアップする必要なく、キャッシュデバイスのデータを引き続き使用できるようにします。

> ***注意：*** 前回のシャットダウンが 20 ページの インテル® CAS を停止する に記載された手順に従っていることを確認する必要があります。キャッシュを有効にする前にコアデータに何らかの変更があった場合、データが正しく同期されず損傷します。

[-f, --force]：キャッシュデバイスにファイルシステムが存在する場合もキャッシュの作成を強制します。通常これは、以前にキャッシュデバイスとして使用されたデバイスに使用されます。

> ***注意：*** これはキャッシュデバイスにあるファイルシステムと既存のすべてのデータを削除します。

[-c, --cache-mode <NAME>] *:* 初めて開始された、または作成されたキャッシュのインスタンスにキャッシュモードを設定します。次のいずれかのモードを設定できます :

wt : (フォルトモード) ライトスルー・モードをオンにします。このパラメーターを使用する場合、読み取り集中操作のみの高速化を行えるライトスルー機能が有効になります。

wb : ライトバック・モードをオンにします。このパラメーターを使用する場合、読み取り集中操作と書き込み集中操作の両方の高速化を行えるライトバック機能が有効になります。

注意 : キャッシュデバイスの障害は、コアデバイスにまだフラッシュされていないデータの損失を引き起こすことがあります。

wa : ライトアラウンド・モードをオンにします。このパラメーターを使用する場合、読み取りのみの高速化を行えるライトアラウンド機能が有効になります。キャッシュに既に存在しないすべての書き込み場所 (すなわち、場所がまだ読み取られていないか排除されている) 場所は、キャッシュをバイパスして直接コアドライブに書き込まれます。書き込みの場所が既にキャッシュに存在する場合がは、キャッシュドライブとコアドライブの両方が更新されます。

pt : キャッシュをパススルーモードで開始します。このモードではキャッシュは効率的に無効になります。これにより、ユーザーは実際にキャッシュを有効にする前に、希望するすべてのコアデバイスをキャッシュするために関連付けすることができます。コアデバイスが関連付けられると、ユーザーは希望するキャッシュモードに動的に切り替えます (詳細については、42 ページの「-Q | --set-cache-mode」を参照してください)。

[-r, --recovery-mode] *:* クリーンでないシャットダウン (危険) の後に開始する際は、ダーティーページをフラッシュします。ライトバック・キャッシュを使用中に電源が切れた場合、このオプションはキャッシュデバイスからダーティーデータをコアデバイスにフラッシュできます。

注 : このコマンドはキャッシュからダーティーデータをフラッシュすることのみを行います。キャッシュは開始しません。リカバリーコマンドを入力後、キャッシュを通常通り開始する必要があります。

注意 : 割り込みのためメタデータが損傷している場合は、データが正しくフラッシュされない可能性があります。これは、データの損傷または損失を引き起こすことがあります。

[-x, --cache-line-size <SIZE>] *:* キャッシュのラインサイズを設定します {4 (デフォルト)、8、16、32、64}。キャッシュのラインサイズはキャッシュを開始する時のみ設定でき、キャッシュが開始された後には変更できません。

9.2 -T | --stop-cache

使用：casadm --stop-cache --cache-id <ID> [option...]

例：

```
# casadm --stop-cache --cache-id 1
あるいは
# casadm -T -i 1
```

説明：キャッシュデバイスに関連するすべてのキャッシュとコアのペアを停止します。

必要なパラメーター：

[-i, --cache-id <ID>]：キャッシュの固有識別子 <1 から 16384>。

オプション・パラメーター：

[-n, --no-data-flush] *:* 終了時にダーティーデータをフラッシュしないでください (危険)。このパラメーターは、キャッシュの停止時にキャッシュデバイスからコアデバイスにダーティーデータをフラッシュすることを許可しません。これにより、キャッシュを停止するのに必要な時間を大幅に短縮して、高速再起動などのアクティビ

ティを行えるようにします。--load パラメーターを使用してキャッシュを再度開始するまでコアデバイスを使わないでください。その後インテル® CAS デバイスを通常に使用できます。

注意： デバイスの停止時はコアデバイスのデータが完全でなくなるかキャッシュデバイスと同期されなくなります。インテル® CAS のキャッシュを開始せずにコアデバイスを使用すると、データの損傷または損失が発生します。

注： ユーザーは Ctrl-C を押してブロックしている --stop-cache 操作を中断できます。ダーティーデータが存在する場合、キャッシュが完全に停止する前に操作を中断すると、キャッシュが引き続き実行されます。ダーティーデータをフラッシュせずにキャッシュを停止したい場合は、--no-data-flush コマンドを使用してください。

9.3 -Q | --set-cache-mode

使用：casadm --set-cache-mode --cache-mode <NAME> --cache-id <ID> --flush-cache <yes/no>

例：

```
# casadm --set-cache-mode --cache-mode wb --cache-id 1 --flush-cache yes
あるいは
# casadm -Q -c wb -i 1 -f yes
```

説明：キャッシュを実行中にユーザーはキャッシュモードを動的に変更できます。

必要なパラメーター：

[-c, --cache-mode <NAME>] *：*

- **wt** - 現在のキャッシュモードからライトスルー・モードに変更します。
- **wb** - 現在のキャッシュモードからライトバック・モードに変更します。
- **wa** - 現在のキャッシュモードからライトアラウンド・モードに変更します。
- **pt** - 現在のキャッシュモードからパススルーモードに変更します。

注： パススルーモードに動的に切り替えると、たとえばメンテナンスを行うときなど、キャッシュへの害を防ぐのに役立ちます。

[-i, -cache-id <ID>] *：* キャッシュの固有識別子 <1 から 16384>。

[-f, --flush-cache] *：* **(ライトバック・モードから切り替えるときのみ必要)**

注意： 次の選択肢を慎重に検討してください。

- **yes** - 新しいキャッシュモードに切り替える前にキャッシュのコンテンツを直ちにコアドライブにフラッシュします。

  Yes を選択してキャッシュを直ちにフラッシュする場合、ダーティーブロックの数によっては、操作に時間がかかることがあります。フラッシュが完了するまで、デバイスへの I/O はパフォーマンスを落として続行されます。

- **no** - 新しいキャッシュモードへの移行を直ちに開始しますが、キャッシュのコンテンツは好機をねらってフラッシュされます。

  no を選択すると、I/O が続行しますが、キャッシュが移行状態となり、キャッシュが完全にフラッシュされるまで新しく選択された状態にならないことに留意する必要があります。新しい状態への移行は *yes* オプションを選択する場合よりも時間がかかります。casadm -L コマンドを使用してキャッシュの現在の状態とフラッシュの % を確認できます。

9.4 -A | --add-core

使用：casadm --add-core --cache-id <ID> --core-device <DEVICE> [option...]

例：

```
# casadm --add-core --cache-id 1 --core-device /dev/sdb
あるいは
# casadm -A -i 1 -d /dev/sdb
```

説明：指定されたキャッシュ ID に関連付けられたフレームワークをコアデバイス (フルデバイスまたはパーティションのいずれか) に追加/マップします。同じ *cache-id* 番号を使用してこのコマンドを繰り返せば、複数のコアデバイスを同じキャッシュデバイスにマップできます。

必要なパラメーター：

[-i, --cache-id <ID>]：キャッシュの固有識別子 <1 から 16384>。

[-d, --core-device <DEVICE>] *:* HDD ストレージ/コアデバイスの場所。
/dev ディレクトリーでデバイスの完全パスを使用する必要があります。例：/dev/sdb

オプション・パラメーター：

[-j, --core-id <ID>]：コアの固有識別子 <1 から 16384>。

9.5 -R | --remove-core

使用：casadm --remove-core --cache-id <ID> --core-id <ID>

例：

```
# casadm --remove-core --cache-id 1 --core-id 1
あるいは
# casadm -R -i 1 -j 1
```

説明：デバイスのキャッシュを無効にする 1 つの方法であるキャッシュ/コアデバイスのマッピングを削除します。

必要なパラメーター：

[-i, -cache-id <ID>] *:* キャッシュの固有識別子 <1 から 16384>。

[-i, -core-id <ID>] *:* コアの固有識別子 <1 から 4096>。
casadm -L コマンドを使用して特定のコアデバイスに割り当てられた値を識別できます。

注意： *casadm -R* を使用する前に、マップされたコアデバイスへのすべての I/O を停止してデバイスが使用されていないことを確認し、アンマウントしてください。.

注： ユーザーは Ctrl-C を押してブロックしている --remove-core 操作を中断できます。ダーティーデータが存在する場合、コアデバイスが完全に停止する前に操作を中断すると、コアデバイスが引き続きキャッシュされます。

9.6 -L | --list-caches

使用：casadm --list-caches

例：

```
# casadm --list-caches
```
あるいは
```
# casadm -L
```

説明：システムで現在実行中の各キャッシュ・インスタンスのリストを表示します。さらに以下も表示します：

- インスタンスで使用されるフラッシュ/SSD デバイス
- インスタンスで使用されるストレージデバイス
- インスタンスの書き込みポリシー (デフォルトではライトスルー)
- インスタンスのステータス

出力の例：

```
type      id   disk               status             write policy   device
cache     1    /dev/sda1          Running            wb             -
+core     1    /dev/sdc1          -                  -              /dev/intelcas1-1
```

9.7 -P | --stats

使用：casadm --stats --cache-id <ID> [option...]

例：

```
# casadm --stats --cache-id 1
```
あるいは
```
# casadm -P -i 1
```

説明：特定のキャッシュ・インスタンスのパフォーマンスとステータスカウンターを印刷します。キャッシュの統計の表示 は詳しい出力を示します。

必要なパラメーター：

[-i, --cache-id <ID>]：キャッシュの固有識別子 <1 から 16384>。

オプション・パラメーター：

[-j, --core-id <ID>]：コアの固有識別子 <1 から 16384>。特定のコアデバイスの統計を表示します。

[-d, --io-class-id <ID>]：IO クラスの固有識別子 <0 から 23>。特定の IO クラスの統計を表示します。

注： <ID> はオプションです。<ID> を指定せずに --io-class-id パラメーターが指定された場合、個別の各 IO クラスの統計が表示されます。

[-f, --filter <filter-spec>]：コンマで区切られたフィルターのリスト (例：--filter=conf, req)。要求された統計のみへ統計の出力をフィルターします。

- ***all：*** (デフォルトモード) 使用可能なすべてのキャッシュの統計を表示します。
- ***conf：*** キャッシュとコアの構成情報とダーティー・タイムスタンプを表示します。
- ***Usage：*** 占有率、空き、クリーン、ダーティーの統計を表示します。
- ***req：*** IO リクエストレベルの統計を表示します。
- ***Blk：*** ブロックレベルの統計を表示します。
- ***err：*** I/O のエラー統計を表示します。

[-o, --output-format <format>]：希望する統計の出力形式を設定します。

- ***table：*** (デフォルトモード) 統計情報の表を表示します。

- ***csv :*** コンマで区切られた統計のリストを出力します。この出力はファイルにパイプして簡単に解析したりスプレッドシート・エディターで開くことができます。

9.8 -Z | --reset-counters

使用：casadm --reset-counters --cache-id <ID> --core-id <ID>

例：

```
# casadm --reset-counters --cache-id 1 --core-id 1
あるいは
# casadm -Z -i 1 -j 1
```

説明：特定のキャッシュ/コアのペアのパフォーマンスとステータスカウンターをリセットします。

必要なパラメーター：

[-i, --cache-id <ID>]：キャッシュの固有識別子 <1 から 16384>。

[-j, --core-id <ID>]：コアの固有識別子 <1 から 4096>。*casadm -L* コマンドを使用して、特定のコアデバイスに割り当てられた値を識別できます。

9.9 -I | --include-files

使用：casadm --include-files --files <FILE ...>

例：

```
# casadm --include-files --files /root/samplefile /usr/bob/anotherfile ...
あるいは
# casadm -I -f /root/samplefile /usr/bob/anotherfile ...
```

説明：キャッシュ・インスタンスの包含表に *1 のファイル*または *ファイルのリスト*を追加します。デフォルトでは、含める特定のファイルが --include-files オプションに示されていない限り、インテル® CAS ソフトウェアはすべてのファイルをキャッシュします。

--include-files オプションが使用されると、リストに示されるファイルのみが含まれ、他のすべてのファイルはキャッシュされません。複数のファイル名はスペースで区切られます。含まれるすべてのファイルはインテル® CAS によって静的アドレスの範囲のリストに変換されます (ソフトウェアの上限は計 65536 アドレス範囲です)。元のファイルがその範囲より大きくなるか小さくなる場合、またはファイルが別の場所に移動される場合、カーネルモジュールはそれを認識しないので動的に調整されません。そのため、追加データはキャッシュされません。このオプションを使用する場合はマッピングを定期的に更新してください。

ファイル名とパスは、絶対パス、相対パス、またはシンボリック・リンクとして指定できます。ユーザーの特定のシェルで解析可能なワイルドカード文字のみサポートされることに注意してください。サポートされる特定のワイルドカード文字については、お使いの Linux* シェルのマニュアル/マンページを参照してください。

- 入力すると、ファイルの包含リストは表示したりシステムで取得できません。
- 1 つのインクルード・ファイルまたは複数のファイルが追加されると、新しいインクルード・ファイル・リストのコマンドを使って上書きするか、--reset_files コマンド (以下参照) を実行する以外に、それらを‘exclude’ (除外) することはできません。
- -I 引き数を使用した casadm の後続の各実行では、既存のリストがリセットされ、古いインスタンスは新しいリストで上書きされます。
- 重複した、無効な、欠落した、または存在しないファイル名は使用できません。

必要なパラメーター：

[-f, --files <FILE …>]：z 含める 1 つまたは複数のファイルのパス。

オプション・パラメーター：

[-v, --verbose]：冗長な出力を設定します。インテル® CAS は選択したインクルード・ファイルのファイルリストに基づいてキャッシュされる LBA 範囲のリストを出力します。

9.10 -X | --reset-files

使用：casadm --reset-files --cache-id <ID> --core-id <ID>

例：

```
# casadm --reset-files --cache-id 1 --core-id 1
```
あるいは
```
# casadm -X -i 1 -j 1
```

説明：特定のキャッシュとコアのペア (*<cache_ID>* と *<core_ID>* で識別) に関連付けられたインクルード・ファイルをリセットします。多対一マッピングでの特定のキャッシュとこのペアでインクルード・ファイルをリセットするのにも使用されます。。

必要なパラメーター：

[-i, -cache-id <ID>] *:* キャッシュの固有識別子 <1 から 16384>。

[-i, -core-id <ID>] *:* コアの固有識別子 <1 から 4096>。

9.11 -F | --flush-cache

使用：casadm --flush-cache --cache-id <ID>

例：

```
# casadm --flush-cache --cache-id 1
```
あるいは
```
# casadm -F -i 1
```

説明：すべてのダーティーデータを指定されたキャッシュデバイスから指定された関連コアデバイスにフラッシュします。

必要なパラメーター：

[-i, --cache-id <ID>]：キャッシュの固有識別子 <1 から 16384>。

注： Ctrl-C を押してブロックしている --flush-cache 操作を中断できます。ダーティーデータが存在する場合、キャッシュが完全にフラッシュされる前に操作を中断すると、ダーティーデータの一部が引き続きキャッシュに残ります。ダーティーデータは好機をねらって通常どおりにフラッシュされます。キャッシュのフラッシュ中、デバイスへの I/O はパフォーマンスを落として続行されます。

9.12 -E | --flush-core

使用：casadm --flush-core --cache-id <ID> --core-id <ID> [option...]

例：

```
# casadm --flush-core --cache-id 1 --core-id 2
あるいは
# casadm -E -i 1 -j 2
```

説明：すべてのダーティーデータを指定されたキャッシュデバイスから指定された関連コアデバイスにフラッシュします。

必要なパラメーター：

[-i, --cache-id <ID>]：キャッシュの固有識別子 <1 から 16384>。

[-i, -core-id <ID>]：コアの固有識別子 <1 から 4096>。

注： Ctrl-C を押してブロックしている --flush-core 操作を中断できます。ダーティーデータが存在する場合、キャッシュが完全にフラッシュされる前に操作を中断すると、ダーティーデータの一部が引き続きキャッシュに残ります。ダーティーデータは好機をねらって通常どおりにフラッシュされます。キャッシュのフラッシュ中、デバイスへの I/O はパフォーマンスを落として続行されます。

9.13 -D | --flush-parameters

使用：casadm --flush-parameters --cache-id <ID> --cleaning-policy-type <NAME> [option...]

例：

```
# casadm --flush-parameters --cache-id 1 --cleaning-policy-type alru --wake-up 20 --
staleness-time 120 --flush-max-buffers 100 --activity-threshold 10000
あるいは
# casadm -D -i 1 -c alru -w 20 -s 120 -b 100 -t 10000
```

説明：このコマンドは、ダーティーデータのフラッシュのさまざまな動作をカスタマイズできます。

注意：このコマンドを使用する前にキャッシュの概念を深く理解していなければなりません。デフォルト設定を変更すると、キャッシュのパフォーマンスに悪影響を与えることがあります。

必要なパラメーター：

[-i, -cache-id <ID>]：キャッシュの固有識別子 <1 から 16384>。

[-c, --cleaning-policy-type <NAME>]：このパラメーターは、指定されたキャッシュに使用するフラッシュポリシーを指定します。デフォルトは alru で、これはダーティーデータを定期的にフラッシュする、変更された最も長い間使われていない方法です。他の唯一のオプションは nop で、これはキャッシュラインの置き換えに必要な場合を除き、単にキャッシュのフラッシュを無効にします。

オプション・パラメーター：

[-w, --wake-up <NUM>]：フラッシュスレッドのウェイクアップの時間を秒で示します。デフォルト値は 20 秒です。

[-s, --staleness-time <NUM>]：キャッシュのブロックがフラッシュのためにスケジュールされる前に、最後の書き込み操作後に経過する必要のある時間を秒で示します。デフォルト値は 120 秒です。

[-b, --flush-max-buffers <NUM>]：1 つのフラッシュサイクルでフラッシュされるダーティー・キャッシュ・ブロックの最大数。デフォルト値は 100 です。

[-t, --activity-threshold <NUM>]：スレッドのクリーニングを開始する前に最後の I/O 操作から経過する必要のある時間をミリ秒で示します。デフォルトは 10000 ミリ秒です。

9.14 -H | --help

使用：casadm --help または casadm --<command> --help

例：

```
# casadm –help
あるいは # casadm –H

# casadm –-start-cache --help
あるいは
# casadm –S -H
```

説明：*casadm* コマンドのリストを簡潔な説明とともに表示します。特定のコマンドについての詳細を見るためにも、このコマンドを使用してください。

9.15 -V | --version

使用：casadm --version

例：

```
# casadm --version
あるいは
# casadm –V
```

説明：インテル® CAS カーネルモジュールとコマンドライン・ユーティリティーのバージョン番号を表示します。

オプション・パラメーター：

[-o, --output-format <format>]：IO クラス構成の希望する出力形式を表示します。

table : (デフォルトモード) IO クラス構成の表を表示します。

csv : IO クラス構成のコンマで区切ったリストを出力します。この出力はファイルにパイプして簡単に解析したりスプレッドシート・エディターで開くことができます。

9.16 -C | --io-class

9.16.1 -C | --load-config

使用：casadm --io-class --load-config --cache-id <ID> --file <file_path>

例：

```
# casadm --io-class --load-config --cache-id 1 --file /etc/intelcas/ioclass-config.csv
あるいは
# casadm –C –C –i 1 –f /etc/intelcas/ioclass-config.csv
```

説明：選択されたキャッシュの IO クラス構成の設定を読み込みます。

必要なパラメーター：

[-i, --cache-id <ID>]：キャッシュの固有識別子 <1 から 16384>。

[-f,--file] : 読み込む IO クラス構成の csv ファイルを指定します。

9.16.2 -L | --list

使用：casadm --io-class --list --cache-id <ID>

例：

```
# casadm --io-class --list --cache-id 1 --file /etc/intelcas/my-config.csv
```

あるいは

```
# casadm -C -L -i 1 -f /etc/intelcas/ioclass-config.csv
```

説明：指定されたキャッシュ ID の IO クラス構成の設定を表示します。オプションでこれらの設定をファイルにエクスポートできます。

必要なパラメーター：

[-i, --cache-id <ID>]：キャッシュの固有識別子 <1 から 16384>。

オプション・パラメーター：

[-o, --output-format <format>]：IO クラス構成の希望する出力形式を表示します。

table：(デフォルトモード) IO クラス構成の表を表示します。

csv：IO クラス構成のコンマで区切ったリストを出力します。この出力はファイルにパイプして簡単に解析したりスプレッドシート・エディターで開くことができます。

10 インストーラーのパラメーター

インテル® CAS インストーラーとリモート.・インストーラーには、インストールをより便利に行えるいくつかのオプションがあります。これらのオプションについて、以下で詳しく説明します。

10.1 -al | --accept-license

インテル® CAS をインストールする前に、ユーザー使用許諾契約を受け入れる必要があります。このオプションは、ライセンスの自動承諾用で、インストールを完了するためにそれ以上の操作を行う必要はありません。このオプションは、インテル® CAS のインストールのスクリプトを作成するのに便利です。

このフラグを使用することにより、使用許諾契約書を読み承諾したことになります。

10.2 -am | --accept-unsupported-module

allow_unsupported_modules フラグの設定を自動的に承諾します。

このパラメーターは、SLES 1 以降の環境で etc/modprobe.d/unsupported-modules にある “unsupported module flag” (サポートされないモジュールのフラグ) を更新します。フラグを設定すると、それが保持され、他のすべてのモジュールに適用されます。このフラグを 1 に設定することは、SLES 1 環境でインテル® CAS を開始するのに極めて重要です。

SLES 11 以外のシステムでインストールする場合、このオプションは影響しません。

このフラグを使用することによって、インストール・スクリプトでこのフラグを設定することに承諾したとみなされます。

10.3 -ad | --accept-dkms

このフラグは、管理者が DKMS をインストールし、DKMS をサポートしてインテル® CAS ソフトウェアをインストールしたいことを示します。これは、各 OS カーネルバージョンの更新時にインテル® CAS カーネルモジュールの自動再構築を有効にします。

10.4 -rd | --reject-dkms

このフラグは、管理者が DKMS をサポートせずにインテル® CAS ソフトウェアをインストールしたいことを示します。

注：このフラグを指定する場合、OS カーネルバージョンのあらゆる更新後に管理者がインテル® CAS を手動で再度インストールする必要があります。

10.5 -as | --auto-start

このオプションは、インテル® CAS 構成ファイルが検出された場合、インストール完了後にインテル® CAS サービスの自動開始を強制します。

注意： このオプションを実行することにより、構成ファイルにキャッシュデバイスとして指定されたハードドライブを上書きすることがあります。

10.6 -d | --display-license

このオプションは、インストーラーを起動せずにエンドユーザー使用許諾契約書を表示します。

10.7 -p | --purge

インテル® CAS をアンインストールして */etc/intelcas* に保存されたすべての構成ファイルを削除します。これは、インテル® CAS を完全に削除するために使用されます。

10.8 -f | --force

このフラグを設定すると、インテル® CAS のインストールのすべてのシステムチェックが無視されます。これには、-am フラグ設定も含まれます。

10.9 -h | --help

このオプションは、インストーラーのオプションのリストを簡潔な説明とともに表示します。

10.10 -l | --list

このオプションは、パッケージによりインストールされたファイルのリストを表示します。これは、パッケージ内のすべてのファイルが正しくインストールされたことを確認するのに役立ちます。

10.11 -t | --try-run

このフラグを設定すると、標準インストール中のみシステムチェックを実行して結果を報告します。

10.12 -r | --reinstall

キャッシュデバイスへの I/O を停止せずに、インテル® CAS の新しいバージョンへのノンストップ・アップグレード (または既存のバージョンのノンストップ再インストール) を実行します。

10.13 -u | --uninstall

インテル® CAS のファイルをアンインストールしますが、今後の使用のために設定を保持します。通常、インテル® CAS の新しいバージョンをインストールするために、古いバージョンをアンインストールするときに使用します。

11 リモート・インストーラーのパラメーター

インテル® CAS リモート・インストーラーには、インストールをより便利に行えるいくつかのオプションがあります。これらのオプションについて、以下で詳しく説明します。

11.1 -al | --accept-license

インテル® CAS をインストールする前に、ユーザー使用許諾契約を受け入れる必要があります。このオプションは、ライセンスの自動承諾用で、インストールを完了するためにそれ以上の操作を行う必要はありません。このオプションは、インテル® CAS のインストールのスクリプトを作成するのに便利です。

このフラグを使用することにより、使用許諾契約書を読み承諾したことになります。

11.2 -am | --accept-unsupported-module

allow_unsupported_modules フラグの設定を自動的に承諾します。

このパラメーターは、SLES 1 以降の環境で etc/modprobe.d/unsupported-modules にある “unsupported module flag” (サポートされないモジュールのフラグ) を更新します。フラグを設定すると、それが保持され、他のすべてのモジュールに適用されます。このフラグを 1 に設定することは、SLES 1 環境でインテル® CAS を開始するのに極めて重要です。

SLES 11 以外のシステムでインストールする場合、このオプションは影響しません。

このフラグを使用することで、インストーラーがインテル® CAS をインストールするすべてのリモートシステムにこのフラグを設定することに承諾したとみなされます。

11.3 -ad | --accept-dkms

このフラグは、管理者が DKMS をインストールし、DKMS をサポートしてインテル® CAS ソフトウェアをインストールしたいことを示します。これは、各 OS カーネルバージョンの更新時にインテル® CAS カーネルモジュールの自動再構築を有効にします。

11.4 -rd | --reject-dkms

このフラグは、管理者が DKMS をサポートせずにインテル® CAS ソフトウェアをインストールしたいことを示します。

注： このフラグを指定する場合、OS カーネルバージョンのあらゆる更新後に管理者がインテル® CAS を手動で再度インストールする必要があります。

11.5 -as | --auto-start

このオプションは、インテル® CAS 構成ファイルが検出された場合、インストール完了後にインテル® CAS サービスの自動開始を強制します。

注意： このオプションを実行すると、インテル® CAS は、構成ファイルにキャッシュデバイスとして選択されたデバイス上のあらゆるデータを自動的に上書きします。

11.6 -d | --display-license

このオプションは、インストーラーを起動せずにエンドユーザー使用許諾契約書を表示します。

11.7 -f | --force

このフラグを設定すると、インテル® CAS のインストールのすべてのシステムチェックが無視されます。これには、-am フラグ設定も含まれます。

11.8 -h | --help

このオプションは、インストーラーのオプションのリストをそれぞれの簡潔な説明とともに表示します。

11.9 -r | --reinstall

キャッシュデバイスへの I/O を停止せずに、インテル® CAS の新しいバージョンへのノンストップ・アップグレード (または既存のバージョンのノンストップ再インストール) を実行します。

11.10 -u | --uninstall

インテル® CAS のファイルをアンインストールしますが、今後の使用のために設定を保持します。通常、インテル® CAS の新しいバージョンをインストールするために、古いバージョンをアンインストールするときに使用します。

11.11 -1 | --one-machine

1 台のマシンでリモート・インストールを実行します。

11.12 -c | --use-config

リモート・インストーラーの構成ファイルのパスを指定します。

11.13 -s | --setup

リモートマシンにアップロードするために 1 つの CAS 構成ファイルのパスを指定します。

11.14 -S | --use-sudo

root 以外のユーザーがインストーラーを実行し、sudo を使用してリモートタスクを実行できるようにします。

注： この方法は推奨されません。root ユーザーとしてリモート・インストーラーを実行することをお勧めします。

注： RHEL* 7.x または CentOS* 7.x にインストールする場合、--use-sudo が機能するようにするために sudoers ファイルから次のディレクティブを削除する必要があります：Defaults requiretty

12 詳細インストールの方法

12.1 リモート・インストール

このステップは、12 ページの ローカル・インストール に記載されたインストール方法の代替となりますが、そのセクションの初めの注記と同じ前提が適用されます。

リモート・インストールを実行するには、2 つのエントリを定義する必要があります：

- *インストール・サーバー*(またはシステム) - インストール・ユーティリティーを起動するマシン (または仮想マシン)。
- *インストール・クライアント*(またはターゲットシステム) -インテル® CAS がインストール/設定されるマシン。

リモート・インストール・スクリプトは次の 3 つの方法のいずれかで実行できます：

1. インテル® CAS をインストールするすべてのインストール・クライアントでパスワードなしの *ssh* ログイン。

   この場合、ユーザーはインストール中に root パスワードの入力を要求されません。このセキュリティのリスクを軽減するには、インストール・サーバーと各インストール・クライアント間で *ssh-keygen* を使用して、セキュアなキーのペアリング (公開鍵と秘密鍵) を作成することを強く推奨します。*ssh-keygen* の実装方法の詳細については、本書の範囲外ですが (この情報はインターネットで自由に入手できます)、この方法による *ssh* キーの生成方法と生成したキーの配布方法の短い汎用例が付録 C で提供されています。*これは、推奨されるリモート・インストール方法です。*

   注： ssh-keygen のインストール方法を使用する場合、リモート・インストールを開始する前に、インストール・サーバー/システムからすべてのインストール・クライアントにパスワードを求められることなく root として *ssh* に ログインできることを確認してください。

2. パスワードなしの *ssh* ログインを使用しないが、インストール・サーバーに *expect* ユーティリティー・プログラムがインストールされている。

   この場合、インテル® CAS リモート・インストーラーはインストール・クライアントごとに一度 root パスワードの入力を求めます。

3. パスワードなしの *ssh* ログインを使用せず、*expect* ユーティリティー・プログラムがインストールされていない。

   この場合、インテル® CAS リモート・インストーラーはインストール・クライアントでアクティビティが実行されるごとに root パスワードの入力を求めます (インストール・クライアント・マシンごとに約 4 回)。

リモート・インストールを開始するには：

1. インストール先の Linux* サーバーまたは VM ゲストでホーム・ディレクトリーにインテル® CAS リモート・インストーラー・ファイルをダウンロードするかコピーします。

   インストールの手順では、サーバー・ファイル・システムで ~ ($HOME と同等) の例を使用します。インストーラーのファイル名は、次の形式になります：installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run (*XX.XX.XX.XXXXXXXX* はバージョン情報です)

2. インストーラー・ファイルを実行可能にします：

   ```
   # chmod u+x ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
   ```

リモート・インストールでは、2 つのオプションを利用できます：

- 1 台のリモートマシンにインストールする。
- 複数のリモートマシンにインストールする。

注： リモート・インストーラーの詳細については、次のコマンドを入力してヘルプを表示できます：

```
# ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run –-help
```

12.1.1 1 台のリモートマシンへのインストール

インテル® CAS を 1 台のリモートホストにインストールするには、次の手順に従います：

1. 次のコマンドを入力します。

   ```
   # ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
    --one-machine <IP ADDRESS>
   ```

2. エンドユーザー使用許諾契約書 (EULA) を読み同意してインストールを続行します。
3. リモート・インストールが正常に完了したのを確認するには、リモートマシンに *ssh* でログインして、インテル® CAS シェルに次のコマンドを入力します。

   ```
   # casadm -H
   ```

   ヘルプコマンドが正常に機能してヘルプのリストが表示されると、インテル® CAS のリモート・インストールは正常に完了しています。また、インストール完了後インストールと同じディレクトリーにインストール・ログファイルが配置されます。

12.1.2 複数のリモートマシンへのバッチ・インストール

複数のリモートマシンへのインストールには、installation_setup_file (セットアップと構成ファイルの形式と構文の詳細については、構成ファイルとセットアップ・ファイルの使用 (詳細設定) を参照してください) が必要です。

1. installation_setup_file の準備が整ったら、次のコマンドを入力して、リモート・バッチ・インストールを開始します：

   ```
   # ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
    --use-config installation_setup_file
   ```

   注： インテル® CAS の現在のバージョンまたは前のバージョンが既にリモートホストに存在する場合、インストーラーがそれらを自動的に検出して次のシステムに進みます。

   注： 構成ファイルのパスは相対パスにできません。絶対パスを指定する必要があります。

2. エンドユーザー使用許諾契約書 (EULA) を読み同意してインストールを続行します。
3. リモート・バッチ・インストールが正常に完了したことを確認します。

   各リモートマシンに *ssh* でログインして Linux* シェルに次のコマンドを入力します：

   ```
   # casadm -H
   ```

   ヘルプコマンドが正常に機能してヘルプのリストが表示されると、インテル® CAS のリモート・インストールは正常に完了しています。また、インストール完了後インストールと同じディレクトリーにインストール・ログファイルが配置されます。

12.1.3 リモート・インストール

インテル® CAS ソフトウェアをリモートでアンインストールするには、13 ページの ソフトウェアのアンインストール に記載されている前提のステップを行ってから、次の手順に従います：

*1 台*のリモートホストからのアンインストールを行うには、次のコマンドを入力します：

```
# ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
--uninstall --one-machine <IP ADDRESS>
```

*複数*のリモートホストからのバッチ・アンインストールを行うには、次のコマンドを入力します：

```
# ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
--uninstall --use-config installation_setup_file
```

注： バッチ・アンインストールで記載されている *installation_setup_file* は、インテル® CAS のインストールに使用されるファイルと同じです。

13 用語

表 11： 用語と定義

用語	定義
キャッシュ	データの透過的なストレージで、そのデータへの今後のリクエストがより速く処理できます。
キャッシュヒット	リクエストされたデータがキャッシュに含まれ、キャッシュから返されたときです。
キャッシュミス	リクエストされたデータがキャッシュにないため、そのデータのプライマリー・ストレージの場所から取得しなければならないときです。
コアデバイス	キャッシュされるデバイスです。
ダーティーデータ	キャッシュ内で変更されているが、メインメモリーでは変更されていないデータをさします。
ゲスト	仮想マシン (VM) 環境で実行されるオペレーティング・システム。
ホスト	仮想マシン (VM) 環境を実行 (ホスティング) し、ゲスト・オペレーティング・システムをインストールできるオペレーティング・システム。
ハイパーバイザー	ホスト・コンピューターで複数のオペレーティング・システム、命名されたゲストを同時に実行することを許可するハードウェアの仮想化テクニック。
I/O	データのフローに関連する入力/出力の省略形。
レイジー書き込み	プライマリー・ストレージへのミラー化のプロセス。*ライトバック*も参照してください。
NAS	ネットワーク接続型ストレージストレージ・エリア・ネットワークまたは他のネットワークに直接リンクされているファイルレベルのデータストレージ (固定ディスクや磁気テープドライブなど)。
パススルー	すべての操作でキャッシュがバイパスされるキャッシュモード。
プライマリー・ストレージ	キャッシュに関して、データが保管されるストレージシステムまたは場所 (DAS、SAN、NAS など)。
SAN	ストレージ・エリア・ネットワーク。リモート・コンピューター・ストレージ・デバイスをサーバーに接続するのに使用するフレームワーク。ストレージ・デバイスはオペレーティング・システムにローカルに接続されているように表示されます。
SSD (ソリッドステート・ディスク)	回転ディスクの代わりにメモリーチップを使用するデータストレージに使用されるデバイス。
階層型ストレージ	価格、パフォーマンス、容量、機能の 4 つの主な特徴で区別される 2 つ以上の種類のストレージ間にデータを移すデータストレージのテクニック。
ライトアラウンド	書き込み操作のいくつかがキャッシュされないキャッシュモード。キャッシュに存在しないブロックへの書き込みは、キャッシュをバイパスして直接コアデバイスに書き込まれます。前の読み取り操作のため、既にキャッシュにあるブロックに書き込み操作が発行されると、書き込みはコアデバイスとキャッシュデバイスの両方に送られます。ライトアラウンド・キャッシュは、書き込み操作がめったに行われず、そのデータにそれ以上の読み取りのアクセスが行われない (したがってキャッシュの利点がない) ときに作業負荷のパフォーマンスを向上します。
ライトバック	データが最初にキャッシュに書き込まれ、I/O 帯域幅が利用可能なときにプライマリー・ストレージにミラー化されるキャッシュモード。プライマリー・ストレージへのミラー化のプロセスは*レイジー書き込み*としても知られています。
ライトスルー	キャッシュへのすべての書き込みがプライマリー・ストレージへの同期書き込みを引き起こすキャッシュモード。

A. FAQ

この付録では、お問い合わせ先と、よくある質問への回答を提供します。

テクニカルサポートへのお問い合わせ方法

テクニカルサポートについては、米国 800-404-2284 までお電話いただくか、次の URL をご覧ください：
http://www.intel.com/support/go/cas。

インテル® CAS Linux* 版がいくらかの DRAM 容量を使用するのはなぜですか？

インテル® CAS Linux* 版は少量のシステムメモリーをメタデータに使用し、メタデータはどのデータが SSD にあり、どのデータが HDD にあるかを知らせます。必要なメモリーの容量は、キャッシュ領域のサイズに比例します。これは、他のどんなキャッシュ・ソフトウェア・ソリューションでも同様です。

インテル® CAS Linux* 版は、インテル® SSD 以外でも機能しますか？

はい。インテル® CAS Linux* 版はあらゆる SSD で機能しますが、当社ではインテル製の SSD でのみ検証を行います。さらに、インテル® SSD と共にご購入いただくと、インテル® CAS をお得な価格でお求めいただけます。

パフォーマンスをテストするには、どうすればよいですか？

提供されている統計に加えて (詳細については、インテル® CAS のモニタリング を参照してください)、ご使用のアプリケーションとシステムで I/O パフォーマンスをテストするのに役立つサードパーティー・ツールを利用できます。これらのツールの一例を次に示します：

FIO (*http://freecode.com/projects/fio*)

ディスクへのアクセスのシミュレーション向け dt (*http://www.scsifaq.org/RMiller_Tools/dt.html*)。

キャッシュを使用中に HDD のパフォーマンスがより遅くなることがありますか？

はい、ありえます。たとえば、キャッシュがライトバック・モードで、キャッシュ全体にダーティーデータが多く存在し、かつ新しいブロックをキャッシュに読み込む必要のある読み取りが発生した場合、読み取りが連続であってもパフォーマンスが劣化します。キャッシュは最初にダーティーブロックを排除する必要があり (これは HDD へのランダムな書き込みを必要とします)、次に HDD から新しいデータを読み取り、最後にそのデータをキャッシュに書き込みます。それに反して、キャッシュなしでは、単に HDD からの単一読み取りとなります。これらの状況を防ぐために、インテル® CAS は IO のアイドル時間中に好機を狙ってキャッシュからダーティーデータをフラッシュします。

キャッシュされたファイルは、どこにありますか？

インテル® CAS Linux* 版はディスクにファイルを保存せずに、キャッシュとして SSD 上のブロックのパターンを使用します。従って、キャッシュされたファイルを見ることはできません。

インテル® CAS Linux* 版のすべてのインストール・ファイルを削除するには、どうしたらよいですか？

20 ページの インテル® CAS を停止する の手順に従ってインテル® CAS を停止し 13 ページの ソフトウェアのアンインストール に記載された手順に従ってインテル® CAS をアンインストールします。

インテル® CAS Linux* 版は、ライトバック・キャッシュをサポートしていますか？

はい。インテル® CAS Linux* 版 v2.6 以降では、ライトバック・キャッシュをサポートしています。詳細については、16 ページの ライトバック・モードの設定 を参照してください。

キャッシュ・コアデバイスの新しいペアを追加する前にキャッシュを停止する必要がありますか？

いいえ。他のインスタンスが実行されている間に、新しいキャッシュのインスタンスを作成できます。

複数のコアデバイスを 1 つのキャッシュに割り当てられますか？

はい。インテル® CAS Linux* 版 v2.5 以降では、1 つのキャッシュドライブまたはインスタンスに複数のデバイス (32 まで検証済みです) を関連付けることができます。casadm -A コマンドを使用して、複数のデバイスを追加できます。

1 つのコアデバイスに複数のキャッシュを追加できますか？

いいえ、1 つのコアデバイスに複数のキャッシュをマップしたい場合、キャッシュデバイスは RAID-0 などのシステムを使用して 1 つのブロックデバイスとして表示されなければなりません。

インテル® CAS でときどきツールがデータの損傷を通知するのはなぜですか？

いくつかのアプリケーション、特に *dt* と *FIO* などのマイクロ・ベンチマークは、デバイスを使用して直接またはローアクセスを実行することがあります。これらのアプリケーションのいくつかは、デバイスのアライメントとブロックサイズの制限などの値を(デバイスから要求せずに)明示的に設定できます。これらのプログラムが機能するには、キャッシュデバイスのブロックサイズとアライメントがツールで選択されたブロックサイズとアライメントと一致する必要があります。

キャッシュデバイスにパーティションを作成する必要がありますか？

いいえ。パーティションを指定しない場合、インテル® CAS はそのデバイス全体をキャッシュデバイスとして使用します。

SSD のパーティションをキャッシュデバイスとして使用できますか？

はい。ただし、最高のパフォーマンスを得るためには、SSD デバイス全体をキャッシュデバイスとして使用することを強く推奨します。

パーティションまたはキャッシュデバイスとして設定されたデバイスをフォーマットする必要がありますか？

いいえ、キャッシュデバイスにはフォーマットの要件はありません。フォーマットが使用された場合、それはインテル® CAS に透過的になります。

インテル® CAS キャッシュボリューム (エクスポートされたオブジェクト用) の論理ブロックサイズと物理ブロックサイズはどのくらいですか？

インテル® CAS キャッシュボリュームの論理ブロックサイズはコアデバイスから継承され、物理ブロックサイズは、キャッシュデバイスまたはコアデバイスの物理ブロックサイズの大きい方として表されます。

> ***注意：*** キャッシュデバイスの論理ブロックサイズがコアデバイスの論理ブロックサイズより大きい場合 (例：SSD の論理ブロックサイズが 4KiB で HDD の論理ブロックサイズが 512B の場合)

SSD または HDD が応答しなくなった場合または切断された場合はどうなりますか？

キャッシュデバイスまたはコアデバイスが応答しなくなると、インテル® CAS はすべての IO を関連キャッシュのエクスポートされたすべてのデバイス (例：/dev/intelcas1-1、/dev/intelcas1-2 など) にフェイルオーバーします。指定されたキャッシュのためにエクスポートされたデバイスへの IO を再開するには、ユーザーは影響されたキャッシュ (この場合はキャッシュ ID 1) を再度開始する必要があります。

デバイスが切断されると、インテル® キャッシュ・アクセラレーション・ソフトウェアから何らかの通知を受け取りますか？

いいえ。OS はデバイスの切断をインテル® CAS に通知しないので、デバイスに IO が試行されるまで、インテル® CAS はデバイスの切断を認識しません。デバイスは --list-caches と --stats 出力に表示され、デバイスへの IO が試行されるまで警告がログに記録されません。Linux* デバイスの標準 IO エラーについては、/var/log/messages と dmesg を確認してください。

ログファイルはどこに記録されていますか？

すべてのイベントは、標準のインテル® CAS システムログに記録されます。*dmesg* コマンドを使用するか、*/var/log/messages* ファイルを確認してください。

テストまたはカーネルのデバッグ中にすべてのメッセージをログに記録するには、echo 8 > /proc/sys/kernel/printk コマンドを使用します。

正常に完了したキャッシュの初期化の一般的なログのサンプル：

```
[Intel(R) CAS] Cache line size: 64 KiB
[Intel(R) CAS] Metadata capacity: 25 MiB
[Intel(R) CAS] Parameters (policies) accepted:: 0 1 1 4
[Intel(R) CAS] Pre-existing metadata, Clean shutdown
[Intel(R) CAS] Done saving cache state!
[Intel(R) CAS] Cache 1 successfully added
[Intel(R) CAS] IO Scheduler of intelcas1-1 is cfq
[Intel(R) CAS] Core "/dev/sdc" successfully added
```

正常に完了したキャッシュの削除の一般的なログのサンプル：

```
[Intel(R) CAS] Removing Cache 1
[Intel(R) CAS] Trying to remove 1 cached device(s)
[Intel(R) CAS] Removed device intelcas1-1.
[Intel(R) CAS] Done saving cache state!
[Intel(R) CAS] Cache 1 successfully removed
```

ライトバック・モードでのキャッシュドライブのフラッシュにかなり時間がかかるのはなぜですか？

フラッシュにかかる時間は、キャッシュデバイスの容量、ストレージのパフォーマンス、および全体的なシステムの使用状況を含みますが、がそれらに限定されない多くの要因に左右されます。フラッシュにかかる時間は、これらの要因によってかなり異なることがあります。casadm で -L オプションを使用して、フラッシュされるキャッシュデバイスのステータスを確認できます。

```
# casadm –L
```

IO デバイスのアクティビティと読み込みを検証するために、次のコマンドを使用することもできます。

```
# iostat –xmt 1
```

B. インテル® SSD DC P3608 シリーズデバイスの設定

この付録は、インテル® SSD DC P3608 シリーズ Solid-State Drive の設定の詳細を示します。

インテル® SSD DC P3608 シリーズデバイスをキャッシュドライブとして使用する場合は、SSD キャッシュドライブ全体を (RAID-0 を使用して) ローとして設定するか、*parted* コマンド (以下に記載) を使用し (パーティションの作成が必要な場合) 単一のパーティションを使用することを強く推奨します。最適なキャッシュのパフォーマンスを得るには、インテル® SSD DC P3608 シリーズのデバイスでパーティションを複数に分割しないでください。

RAID-0 を使用するロー・パーティション分割

SSD キャッシュドライブ全体でともに RAID を使用したい場合は、次のコマンドを使用して最適に行ってください。

```
# mdadm --create --verbose /dev/md0 --level=0 –-chunk=64
  --raid-devices=2 /dev/nvme0n1 /dev/nvme1n1
```

md0 は RAID-0 論理ドライブの結果出力で、*nvme0n1* と *nvme1n1* は対応する 2 つの RAID 入力ドライブです (インテル® SSD DC P3608 シリーズデバイスに適用)。RAID レベルは “0”、チャンクサイズは “64K” に設定されます。

注： *mdadm* コマンドは、システムの再起動または電源が切れたときにこの設定を保存しません。システムが RAID にマウントされている場合、*/etc/madadm.conf* や */etc/fstab* などのシステムファイルに上記のコマンドを追加する必要があります。

***parted* コマンドを使用した単一パーティション**

露出された 2 つのインテル® SSD DC P3608 デバイス (合計デバイス容量の半分) のいずれかを使用して、単一パーティションを持つキャッシュドライブを作成します。

```
# parted -s -- /dev/nvme0n1 mkpart primary 2048s 100%
```

/dev/nvme0n1p1 は、結果として作成された単一パーティションです。

注： *fdisk* の古いリビジョン (*util-linux-ng パッケージ2.17.2* 以前のリビジョン) はパーティションを正しく調整しないことがあるため、インテルは *fdisk* の代わりに *parted* ユーティリティ-を使用することを推奨します。*parted* ツールは、デフォルトでパーティションを 1 MB の境界に調整します。

C. 汎用 ssh-keygen コードスニペット

この付録では、パスワード不要のログインを有効にして、2 つのマシンのペアを設定できる *ssh-keygen* コードスニペットを提供します。

ssh-keygen コマンドを使用してキーを生成します。

```
# ssh-keygen
```

注： 指示されたときに、オプションのパスフレーズを指定することをお勧めします。パスフレーズを空白のままにしないでください。

これは、*~/.ssh* ディレクトリに秘密鍵と公開鍵のペアを生成します。クライアント・マシンに公開鍵をコピーするには、インテル® CAS シェルに次のコマンドを入力します：

```
# ssh-copy-id –i ~/.ssh/id_rsa.pub <admin>@<host_ip>
```

このセキュアなペアリングが正しく機能したことを確認するため、次のコマンドを使用してリモートシステムに *ssh* 接続します：

```
# ssh <admin>@<host_ip>
```

ssh セットアップが機能したことを確認するためだけでなく、RSA 鍵のフィンガープリントを受け入れるために、この操作を行う必要があります。(行わない場合、自動スクリプトが失敗します。)

注： このトピックの追加コマンド情報は、本書の範囲外ですが、詳細については、インテル® CAS の *man* ページまたは自由に利用できるさまざまな使用方法の Web サイトで *ssh* と *ssh-keygen* の例を参照できます。

D. インストール・ファイルのツリー

この付録は、インテル® CAS Linux* 版 v3.1 インストーラーによってインストールされるファイルのパスとファイル名のリストを提供します。

すべての OS にインストールされるファイル：

```
/sbin/casadm
/etc/init.d/intelcas
/etc/intelcas/*
/usr/share/man/man8/casadm.8.gz
/lib/modules/`uname -r`/extra/intelcas.ko  (注：DKMS が有効になっている場合、このパスは異なることがあります)
/lib/modules/`uname -r`/extra/inteldisk.ko (注：DKMS が有効になっている場合、このパスは異なることがあります)
```

DKMS が有効になっている場合にインストールされる追加のファイル：

```
/usr/src/intelcas-<CAS_version>/*
```

SystemD ベースの OS (例：RHEL* 7.0) にインストールされる追加のファイル：

```
/usr/lib/systemd/system/intelcas.service
```