アドレス・レンジ・パーシャル・メモリー・ミラーリング

この記事は、インテル® デベロッパー・ゾーンに公開されている「Address Range Partial Memory Mirroring」の日本語参考訳です。

オペレーティング・システム・インターフェイスの仕様

目次

表

はじめに

インテル® Xeon® プロセッサー・ファミリー・ベースのプラットフォームは、メモリー・ミラーリングと呼ばれる RAS (信頼性、可用性、保守性) 機能を提供します。この機能を利用すると、ユーザーはメモリー・コンポーネントが訂正不能なエラーの影響を受ける場合でもシステムのアップタイムを維持できる、非常に信頼性の高いモードでメモリーを設定することができます。この機能を有効にした場合、メモリー容量と信頼性のトレードオフが発生します。例えば、ユーザーがアクセス可能なメモリーは利用可能なメモリー全体の半分に減ります。

インテル® Xeon® プロセッサー E7 v3 製品ファミリーベースのプラットフォームでは、アドレス・レンジ・ミラーリングと呼ばれるパーシャル・メモリー・ミラーリングの新しいレイヤーのサポートが行われました。アドレス・レンジ・ミラーリングを使用すると、そのプラットフォームでは OS で利用可能なメモリー全体の中からミラーリングする範囲を指定することができます (オプションで、レンジ 0–4GB をミラーリングするかどうかも指定できます)。この機能により、ユーザーはミラーリングされていないメモリーレンジとミラーリングされたメモリーレンジ間で適切なトレードオフを行うことができます。つまり、非常に信頼性の高いメモリーレンジを基幹業務系のワークロードやカーネル空間向けに確保しながら、利用可能なメモリー全体を最適化することができます。

従来のメモリー・ミラーリングは OS に透過的でしたが、アドレス・レンジ・ミラーリングではユーザーがミラーリングするメモリーのサブセットを指定するため、ファームウェアと OS 間のインターフェイスが必要になります。アドレス・レンジ・ミラーリングを活用するには、次の操作を行います。

- プラットフォームでミラーリングされたメモリーの一部またはすべてを OS に示します。
- 以降のブートで効力を発するミラーリングされるメモリーの量をリクエストする方法を OS に提供します。

この仕様では、ミラーリングされたメモリーのブロックを示すため、ファームウェア (BIOS/SMM) と OS 間のインターフェイスを定義します。また、OS が以降のブートでメモリーの割合をどのように割り当てることができるか (また、オプションで 0–4GB をミラーリングするかどうか) 定義します。

ミラーリングされたメモリーを OS に示す—ブート時

既存の UEFI 呼び出しは、プラットフォームにより示されたアドレスレンジをすべて OS に返します。ミラーリングするように設定するメモリーレンジは、EFI_MEMORY_DESCRIPTOR の属性フィールドで示されます。ミラーリングされたメモリーは次の属性と関連付けられます。

```
#define EFI_MEMORY_MORE_RELIABLE     0x0000000000010000
```

この仕様の適用範囲はメモリー・アドレス・レンジ・ミラーリングをサポートすることですが、上記の UEFI スキームを使用して OS にミラーリングされたメモリー量を明確にすることもできます。

UEFI 呼び出し GetMemoryMap() は UEFI ブートサービスの一部であるため、ExitBootServices() が実行された後には起動できないことに注意してください。そのため、このスキームはブート時にのみミラーリングされたメモリーの状態を定義します。つまり、ホットアドされたメモリーのミラーリング機能は、このスキームを使用して記述できません。同様に、ブート時よりも後に発生したメモリーの冗長性の損失は、このスキームではキャプチャーされません。

ミラーリングされたメモリーのランタイム中のホットアド/リムーブ

ホットアドされたメモリーは、ACPI 名前空間オブジェクト PNP0C80 を使用して OS に示すことができます。詳細は、ACPI v5.1 仕様のセクション 5.2.21.10 を参照してください。

メモリー・デバイス・オブジェクトは、メモリーデバイスに対応するアドレス空間を記述する _CRS メソッド (Current Resource Setting – ACPI v5.1 仕様のセクション 6.2.2 を参照) を含みます。ホットプラグ・メモリーの場合、デバイスはホットアドされたメモリーを示します。ブート時に静的に示されたメモリーについて、プラットフォームは非連続のアドレスレンジを理解する _CRS メソッドを使用して 1 つのメモリーデバイス全体のメモリーレンジを表します (必要な場合)。

ミラーリングされるメモリーレンジは、拡張アドレス空間ディスクリプターの _ATT (型指定属性) 属性フィールドを使用して、_CRS メソッドで示すことができます。詳細は、ACPI v5.1 仕様のセクション 6.4.3.5.4 を参照してください。

メモリーの冗長性ステータスが変更される特別なケースでは (メモリー・ミラーリングの冗長性の損失など)、プラットフォームは対応するメモリーデバイスでバス・チェック・イベント (Notify 0x00) を使用して OSPM (OS 電力管理) レイヤーを通知します。イベントトリガーは _CRS の再評価を引き起こし、_ATT 属性フィールドで冗長性の損失が示されます。

```
Scope (\_SB){
Device (MEM0) {
Name (_HID, EISAID ("PNP0C80"))
Name (_CRS, ResourceTemplate () {
        …
})
} // MEM0
} // _SB
Scope(\_GPE)
{
Method(_L00) {
        Notify(\_SB.MEM0, 0x00) // Bus Check
}
}
```

注: _ATT 属性は UEFI 属性を使用します。

メモリー・アドレス・レンジ・ミラーリングの UEFI 変数

UEFI 変数を利用すると、OS やシステム・ソフトウェアでプラットフォームがミラーリングしているメモリーブロックを識別することができます。これらの変数は、以降のブートで OS にミラーリングのリクエストを許可するためにも使用されます。

サポートしている機能は次のとおりです。

- 以降のブートで 4GB[1] 未満のメモリーのミラーリングを有効または無効にします。
- 以降のブートでミラーリングされるメモリーの合計容量 (50 パーセント以内) を指定します。
- 現在のブートで 4GB 未満のメモリーがミラーリングされたかどうかを示します。
- 現在のブートでミラーリングされたメモリー量を示します。
- 最後のブート中にリクエストしたメモリー冗長性のステータス (4GB 以下のパーセント) およびリクエストのステータス (SUCCESS、FAILURE または PARTIAL) を示します。

UEFI 変数について

このセクションでは、UEFI 変数の高レベルの概要を示します。UEFI 変数の使用についての詳細な説明は、UEFI 仕様 v2.4 のセクション 7.2 を参照してください。

変数は、情報と属性 (キー) および任意のデータ (値) を一意に識別して構成される、キー/値のペアとして定義されます。また、変数は、プラットフォームで実装される EFI 環境ローダーやその他のアプリケーションと EFI 環境で実行する EFI OS 間で渡されるデータを格納する手段として使用されます。

変数の管理には、次の UEFI ランタイムサービスを利用します。

表: UEFI ランタイムサービス

名前	種類	説明
GetVariable	ランタイム	変数の値を返します。
GetNextVariableName	ランタイム	現在の変数名を列挙します。
SetVariable	ランタイム	変数の値を設定します。
QueryVariableInfo	ランタイム	EFI 変数に関する情報を返します。

メモリー・アドレス・レンジ・ミラーリングの UEFI 変数

次の GUID を使用して両方の変数と関連付けます。

```
#define ADDRESS_RANGE_MIRROR_VARIABLE_GUID { 0x7b9be2e0, 0xe28a, 0x4197, 0xad,
0x3e, 0x32, 0xf0, 0x62, 0xf9, 0x46, 0x2c }
```

変数データの構造は次のとおりです。

```
typedef struct{

UINT8       MirrorVersion;

BOOLEAN     MirrorMemoryBelow4GB;

UINT16      MirroredAmountAbove4GB;

UINT8       MirrorStatus;

} ADDRESS_RANGE_MIRROR_VARIABLE_DATA;
```

- MirrorVersion は初期実装では 1 に設定します。
- MirrorMemoryBelow4GB は 4GB 未満のメモリーをミラーリングする場合は true に設定します。
- MirroredAmountAbove4GB はミラーリングする必要がある 4GB を超える利用可能なメモリーの量で、ベーシスポイント (1/100%; 12.75% = 1275) で設定します。
- MirrorStatus は、ミラーリングをセットアップしたときに指定のパラメーターが適用されたかどうかを示します。

表: MirrorStatus の値

ステータス文字列	値	説明
SUCCESS	0	成功。
MIRROR_INCAPABLE	1	プラットフォームはアドレス・レンジ・ミラーリングをサポートしていません。
VERSION_MISMATCH	2	無効なバージョン番号です。
INVALID_REQUEST	3	MirroredMemoryAbove 4GB > 50.00%
UNSUPPORTED_CONFIG	4	DIMM 設定でミラーリングが無効です。
OEM[2]_SPECIFIC_CONFIGURATION	5	OEM 固有の方法で、この構造では表すことができないミラー設定が行われました。

マルチソケット・システムでは、プラットフォームは各 NUMA ノードのメモリー量にほぼ [3] 比例するようにミラーリングするメモリーレンジを分配する必要があります。例えば、ノード 0 が 64GB、ノード 1 が 32GB の 2 つのノードのマシンで 12GB のメモリーをミラーリングする場合、ノード 0 に 8GB、ノード 1 に 4GB を割り当てます。

例えば、システムのメモリーの合計が 48GB で 12GB のメモリーを (4GB 未満のメモリーを含めて) ミラーリングする必要がある場合、次のように計算します。

- メモリーの合計 [4] = 48GB
- 4GB を超えるメモリーの合計 = 46GB (TOLM = 2GB と仮定)
- 割合 = 10/46 * 100 = 21.74% = 2174 ベーシスポイント

ソケット 0 に 32GB、ソケット 1 に 16GB のメモリーが割り当てられた 2 ソケットのシステムを考えます。リクエストされた 21.74% の割合を保つため、ソケット 0 で 8GB のメモリーをミラーリングし、ソケット 1 で 4GB のメモリーをミラーリングします。その結果、OS が各 NUMA 領域で適切なメモリー量をミラーリングすることが保証されます。

ADDRESS_RANGE_MIRROR_VARIABLE_DATA のサイズは次のように計算します。

```
#define ADDRESS_RANGE_MIRROR_VARIABLE_SIZE
sizeof(ADDRESS_RANGE_MIRROR_VARIABLE_DATA)
```

変数は次の属性と関連付けられます [5]。

```
#define ADDRESS_RANGE_MIRROR_VARIABLE_ATTRIBUTE \
(EFI_VARIABLE_NON_VOLATILE | EFI_VARIABLE_BOOTSERVICE_ACCESS | \
EFI_VARIABLE_RUNTIME_ACCESS)
```

説明:

```
#define EFI_VARIABLE_NON_VOLATILE  0x00000001
#define EFI_VARIABLE_BOOTSERVICE_ACCESS  0x00000002
#define EFI_VARIABLE_RUNTIME_ACCESS  0x00000004
```

変数は次の名前と関連付けられます。

```
#define ADDRESS_RANGE_MIRROR_VARIABLE_CURRENT “MirrorCurrent”
```

- この変数は、プラットフォームのアドレス・レンジ・ミラーリングの現在のステータスを伝えるため、BIOS により書き込まれ、OS により読み取られます。

```
#define ADDRESS_RANGE_MIRROR_VARIABLE_REQUEST “MirrorRequest”
```

- この変数は、次のブートで新しいミラー設定をリクエストするため、OS により書き込まれます。

GetVariable()

GetVariable() は次のパラメーターを使用して起動される UEFI ランタイムサービスです。

プロトタイプ

```
typedef
EFI_STATUS
GetVariable(
        IN CHAR16     *VariableName,
        IN EFI_GUID  *VendorGuid,
        OUT UINT32    *Attributes,
        IN OUT UINTN  *DataSize,
        OUT VOID    *Data
);
```

パラメーター

```
VariableName ADDRESS_RANGE_MIRROR_VARIABLE_{CURRENT,REQUEST},
VendorGuid   ADDRESS_RANGE_MIRROR_VARIABLE_GUID,
```

```
Attributes   ADDRESS_RANGE_MIRROR_VARIABLE_ATTRIBUTE,
DataSize     ADDRESS_RANGE_MIRROR_VARIABLE_SIZE,
Data         ADDRESS_RANGE_MIRROR_VARIABLE_DATA
```

SetVariable()

SetVariable() は次のパラメーターを使用して起動される UEFI ランタイムサービスです。

プロトタイプ

```
typedef
EFI_STATUS
SetVariable(
        IN CHAR16     *VariableName,
        IN EFI_GUID  *VendorGuid,
        IN UINT32    *Attributes,
        IN UINTN    *DataSize,
        IN VOID    *Data
);
```

パラメーター

```
VariableName ADDRESS_RANGE_MIRROR_VARIABLE_{CURRENT,REQUEST},
VendorGuid  ADDRESS_RANGE_MIRROR_VARIABLE_GUID,
Attributes  ADDRESS_RANGE_MIRROR_VARIABLE_ATTRIBUTE,
DataSize    ADDRESS_RANGE_MIRROR_VARIABLE_SIZE,
Data        ADDRESS_RANGE_MIRROR_VARIABLE_DATA
```

マネージメント・インターフェイス・フロー—BIOS

最初のブートで、BIOS はシステムがアドレス・レンジ・ミラーリング機能をサポートしていることを OS に示す「MemoryCurrent」変数を作成すべきです。メモリーはミラーリングしないため、変数を次のように設定します。

```
MirrorVersion = 1
MirrorMemoryBelow4GB = false
MirroredAmountAbove4GB = 0
MirrorStatus = 0
```

以降のブートで、BIOS は MemoryRequest 変数を探すべきです。変数が存在する場合、BIOS はそれらのパラメーターでミラーリングを設定し、最後に使用されたパラメーターと (成功、または失敗の理由を示す) MirrorStatus フィールドを含む「MemoryCurrent」変数を設定すべきです。

マネージメント・インターフェイス・フロー—OS

初期のブート中、OS は GetMemoryMap() により返された属性ビットを使用してミラーリングするメモリーのレンジを見つけます (同じデータは _ATT 属性から取得することもできます)。

ミラーの設定にエラーがあった場合、OS は MemoryCurrent.MirrorStatus およびレポートをチェックします。

ホットプラグ・メモリー操作中およびミラー失敗イベント時に、OS は _ATT 属性フィールドを使用してメモリー・ミラーリング設定への変更を確認します。

OS でミラー設定を変更する場合は、「MemoryRequest」を次のように設定します。

```
MirrorVersion = 1
MirrorMemoryBelow4GB = desired mirror below 4 GB
MirroredAmountAbove4GB = desired percentage above 4 GB
MirrorStatus = 0
```

リブートして BIOS が新しい設定を適用できるようにします。

---

[1] サポートを 4GB 未満の連続するアドレスレンジのメモリーに制限します。実際のサイズはプラットフォームの TOLM 設定により決まります。

[2] OEM: Original Equipment Manufacturer の略。日本語訳は相手先ブランド名製造など。

[3] プラットフォームの制約により、リクエストされた正確な量を提供できない場合、プラットフォームのメモリーを切り上げるべきです。

[4] メモリーの合計はミラーリングしない OS がアクセス可能なすべての範囲の和で計算します。

[5] OS が更新または削除しないように、BIOS が「MirrorCurrent」の認証書き込みアクセス属性を追加で指定することがあります。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。