4A-2

エンタープライズLinux向け高信頼メモリダンプ機能の開発

小笠原 克久　　杉田 由美子

（株）日立製作所 システム開発研究所

1. はじめに

企業基幹業務におけるLinuxシステムの需要が増えるに従い，その信頼性向上が求められている．Linuxに不足する信頼性強化を目的とした障害解析支援機能の中でも，メモリダンプ機能は，障害発生時におけるメモリ内容の参照および保存を可能とすることから，障害要因特定のための最も重要な手がかりを提供する．

本稿では，ミッション・クリティカル・システムで利用されるIA-64プラットフォームに対応した，Linux高信頼メモリダンプ機能（Kdump for IA-64）の設計・実装について述べる．

2. Kdump for IA-64の概要

現在，Linuxではいくつかのメモリダンプ機能が利用可能である．しかし，これらは信頼性が低下した障害OS上でダンプ処理を行なうため，成功率や安全性に問題がある．この問題に対し，我々は，システム障害発生時においても確実かつ安全にメモリダンプを取得可能なダンプ機能LTDを独自に開発してきた[1]．LTDは，OS障害時に別のOSを起動し，正常動作しているシステム上で障害OSのメモリ内容をダンプデバイスに保存する．一方，Linux開発コミュニティでも，類似方式であるKexec-based Crash Dump（Kdump）が開発されている．このKdumpは，Linux標準機能としてi386版カーネルに取り込まれたが，IA-64サーバへの対応は行われていない．大規模基幹向けシステムでは，IA-64サーバの需要も多く，障害解析支援機能の強化は必須である．そこで，LTDの基本機能を基にKdump for IA-64（図1，KdmupおよびKdump tools）を開発し，Linux標準ダンプ機能として提案中である．

Kdump for IA-64の動作概要を以下に示す．

2.1. 初期化処理

運用システムは，起動時，ダンプ取得システムを保存するためのメモリ領域を予め確保する．運用システム起動後，Kdumpは以下の初期化処理を行い，障害時のダンプ処理に備える．

(1) 予約メモリ領域へダンプ取得システムを保存

ダンプ取得システムは，カーネル，RAMDISK，起動パラメータ，ダンプヘッダおよびダンプ取得システム用物理メモリマップ・テーブルから構成され，Kdump toolsを使用して作成する．

(2) OS INITハンドラを運用OSに登録

INITハンドラは，ダンプ手動指示等によって発行されるINIT割り込みに対応した処理を行う．

2.2. ダンプ処理

Kdumpによるダンプ処理に実行制御が移る契機は，OS障害およびダンプ手動指示がある（図1(a)(b)）．これらを契機に以下の処理を行う．

(1) ダンプ取得OS起動CPUを除く稼働中全CPUの機能停止

(2) 全CPUレジスタ情報の保存

(3) 外部デバイスの停止

(4) 障害OSからダンプ取得OSへの空間切替

(5) 障害システムメモリ領域を/proc/vmcoreを介して参照し，ダンプの取得／解析を実行

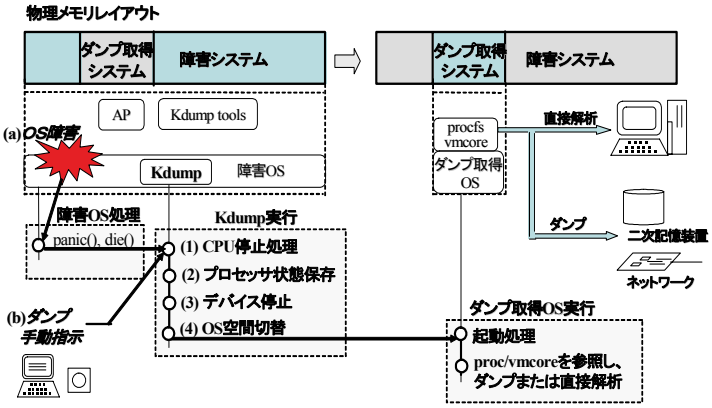

図1　Kdump for IA-64のダンプ処理動作概要

3. Kdump for IA-64における実装の課題

2章で述べたKdump for IA-64機能を実現するためには，以下の課題を解決する必要があった．

(1) 障害システムメモリ領域の保持

Kdumpは，障害発生時にハードウェアによる初期化処理をスキップし，ダンプ取得OSを直接起動することによってハードウェア／ファームウェアによるメモリ内容消去を回避する．しかし，ダンプ取得OSをそのまま起動すると，OSメモリ管理初期化時に障害発生メモリ領域もページング対象としてしまい，障害時のメモリ内容を保証できない．したがって，障害システムのメモリ領域を保持する機能が必要となる．

(2) CPU停止処理およびレジスタ情報保存

Kdumpは，ダンプ取得OSを任意のひとつの

Development of Highly Reliable Memory Dump Facility for Enterprise Linux
Katsuhisa OGASAWARA and Yumiko SUGITA
Systems Development Laboratory, Hitachi, Ltd.

CPU で起動する．この時，ダンプ取得 OS 起動 CPU 以外の CPU を確実に停止状態にし，動作不定となることを回避する必要がある．また同時に，メモリダンプからバックトレース解析等を可能にするため CPU レジスタ情報をメモリ上に保存する機能が必要となる．

(3) OS 空間切替

障害 OS からダンプ取得 OS を正常起動するためには，OS 起動エントリポイントが定める CPU モードに設定し，かつ，起動パラメータをダンプ取得 OS へ渡す必要がある．さらに，仮想−物理アドレス変換値の不正を回避するために TLB エントリを無効化する機能が必要となる．

4. Kdump for IA-64 実現方式

3章で述べた課題に対して，以下の実現方式を適用することで解決した．

(1) 障害システムメモリ領域保持機能

障害発生時のメモリ内容を保持するため，EFI ファームウェアに代わり Kdump が障害メモリ領域を使用しない物理メモリマップ・テーブルを作成し，ダンプ取得 OS がそれを参照するように実装した（初期化処理(1)）．図 2はダンプ取得 OS が動作する物理メモリマップである．物理メモリマップ・テーブル（図 2，EFI mem map）は，ダンプ取得 OS 領域として確保した ELF header を除く *Xs*～*Xe* のメモリ領域を利用可能とする（Conventional）．一方，それ以外の障害 OS がメモリ管理していた領域はすべてダンプ取得 OS から使用不可能とする（Reserved）．ファームウェア領域とメモリマップド IO 領域は変更せずにそのまま引き継ぐ．

(2) CPU レジスタ保存機能および CPU 停止機能

障害発生時，任意 CPU が割り込み禁止状態であっても確実に全 CPU を Kdump 処理へ移行させるため，INIT モードのプロセッサ間割り込み（INIT-IPI）を利用し，実行中の命令制御を変更する機能を実装した（ダンプ処理(1)(2)）．障害を検知した CPU は，自身を含め全 CPU へ INIT-IPI を発行する．OS INIT ハンドラは各 CPU で逐次処理される．まず，バックトレース解析が可能なように CPU レジスタ情報を保存する．次に，最初に INIT ハンドラ処理を行った CPU をダンプ取得 OS 起動 CPU とし，他 CPU は CPU 停止命令を実行する．

(3) OS 空間切替機能

障害 OS からダンプ取得 OS 起動へ制御移行し正常起動するため，OS 空間切替機能を実装した（ダンプ処理(4)）．OS 空間切替機能は，まず，ダンプ取得 OS 起動 CPU を仮想モードから物理モードへ移行する．これは，OS 起動エントリポイントが物理モードで実行されることを想定しているためである．次に TLB を無効化する．これは CPU が物理モードへ移行した後に行う必要がある．なぜなら，CPU が仮想モードで動作していると，仮想アドレスから物理アドレスへの変換が行われてしまうためである．最後に，起動パラメータ（図 2，ia64 boot param）の物理アドレスをレジスタに設定し，ダンプ取得 OS 起動エントリポイントへ制御を移す．

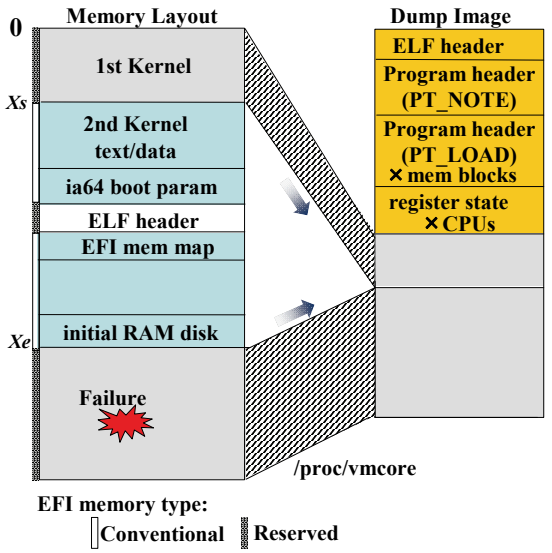

図 2　Kdump による物理メモリマップ

以上の実現方式を基に開発した Kdump for IA-64 により，障害発生時のメモリ内容を保持したままダンプ取得 OS を正常起動し，そのダンプ取得 OS 上でダンプ取得／解析を確実かつ安全に実施できることを確認した．

5. おわりに

システム障害時においても確実かつ安全にメモリダンプを取得することを目的とし，障害システムのメモリ内容を保持したままダンプ取得 OS を起動し，その OS 上でダンプ解析／取得を行なう Kdump for IA-64 を開発した．ソースコードは，GPL ライセンスで公開済みである．（http://sourceforge.net/projects/dle/）

6. 参考文献

[1] Linux Tough Dump, Hitachi, http://www.hitachi.co.jp/Prod/comp/linux/products/solution.html

[2] Vivek Goyal, Eric W. Biederman and Hariprasad Nellitheertha, "Kdump, A Kexec-based Kernel Crash Dumping Mechanism", Proceedings of the Linux Symposium, Vol.1, pp.169-180, 2005.