オープンソースカンファレンス2005

クラッシュダンプ解析ツール「Alicia」の開発
ー　Aliciaの概要　ー

2005年3月25日

ユニアデックス（株）
プロダクト事業グループ
高橋　秀樹

1. Aliciaの開発背景

メインフレームでのダンプ解析

●ダンプさえあれば、ほとんどのトラブルは原因判明する
- ーダンプ解析は、ミッション・クリティカル・サポートの大きな柱
- ーダンプ解析に必要な情報のほとんどがメモリ上にセーブされている
- ーどんな状況下にあってもダンプ採取が可能
- ーダンプ解析ツールの使い勝手が良く、利便性の高い機能が豊富

Linuxでのダンプ解析

●ダンプ解析が原因の特定に有効活用されていない
- ーエンタープライズ分野でのダンプ活用事例が少ない
- ーダンプを採取できず、障害発生時のメモリ内容をロストすることがある
- ーダンプ解析ツールの使い勝手が悪く、効率的な解析が難しい

2. Linuxダンプ解析の課題解決

目的

目指すのはメインフレーム並みの保守環境の提供。ミッション･クリティカルLinuxを実現するために、ダンプ解析ノウハウの共有化とダンプ解析時間の短縮化を図る

現状の問題点

- 各ディストリビュータによってLinuxカーネルダンプの形式が統一されていない
- 解析者はダンプ形式を意識して解析を行わなければならない
- データを抽出するまでに単純なオペレーションを繰り返し実行したり、取得データを別途編集し直したりで、解析効率が悪い
- ダンプ解析技術は個人スキルであり、解析ノウハウが共有できない

改善項目

ダンプデータ解析環境の充実が必要

- ダンプ形式に依存しないダンプ解析環境
- 容易に新規ダンプ解析コマンドを追加作成できる環境
- 解析手順を蓄積し共有できる環境

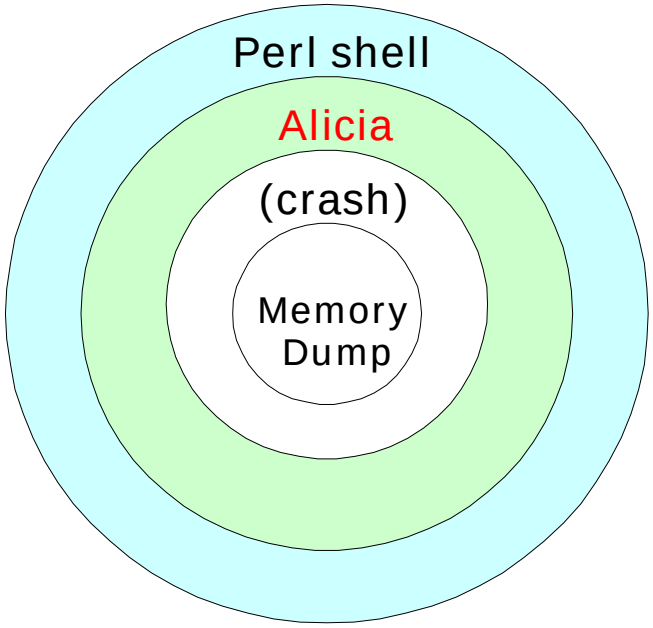
対策

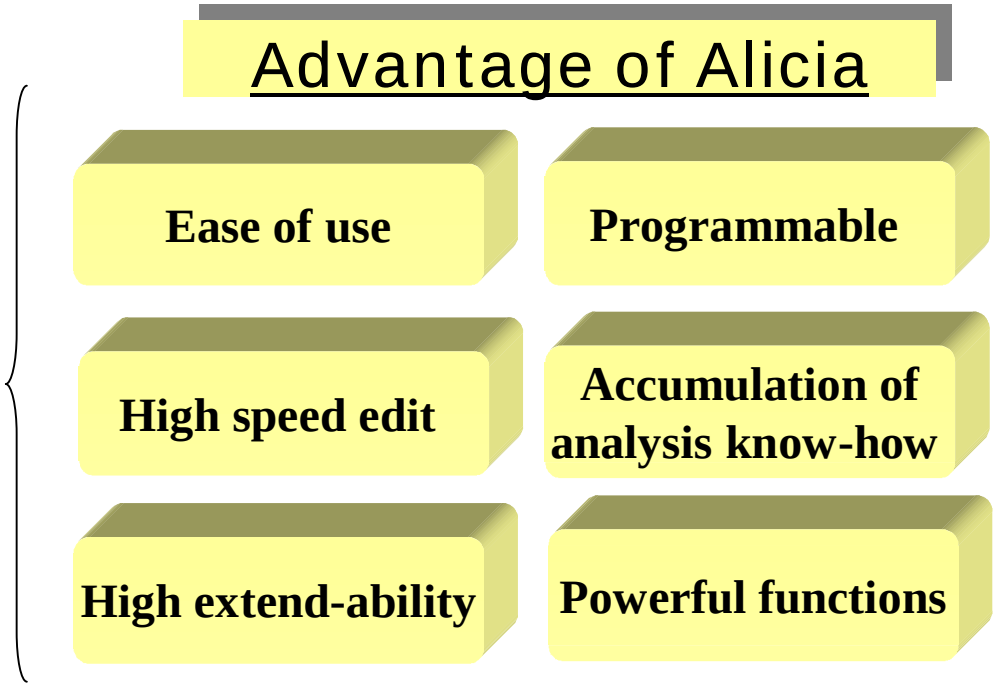
- 既存ダンプ解析ツール(crash/lcrash) に対する共通インタフェースを定義し、ダンプ解析処理手順をスクリプト化できるツールを開発する

3. Aliciaの特長

1. ダンプ解析者に対して多くの利便性を提供　(解析作業の短縮化が可能)
2. ダンプ解析知識を持たない人にダンプ編集*作業の依頼が可能　(客先配布が可能)
3. Linuxの構造解析等で実データの検証や熟練ダンプ解析者のノウハウ習得が可能
4. 一から作り直した新しいダンプ解析ツールではなく、既存ツールを統合し、既存ツールの良さをそのまま利用可能

*ダンプ編集：メモリダンプから解析に必要なデータを読み込み、ダンプ解析に必要な情報を可視化すること





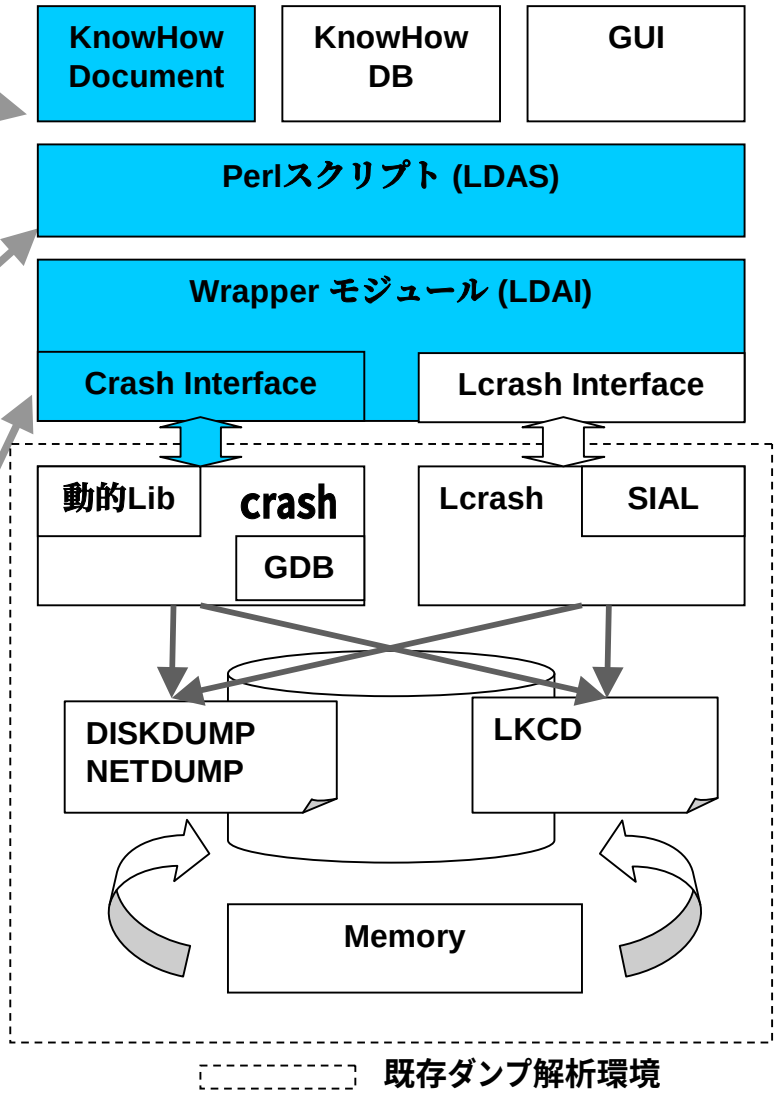
4. Aliciaの位置づけと主要コンポーネント

既存解析ツールに対する統合型ダンプ解析ツールAliciaの開発

ノウハウ蓄積:
- LDASはインタプリタであり、可視性がよく、そのままDump解析のノウハウとして蓄積可能
- 修正も容易であり、Dump解析中に既に作っておいたスクリプトを変更して、新しい解析コマンドを即座に作成可能

Perlスクリプト(LDAS)
(Linux Dump Analysis Scripts)
- Wrapperモジュールを利用して新規解析コマンドを開発
- これまでにDump解析で必要とされた手順をスクリプト化することで同種のトラブルを迅速に解析

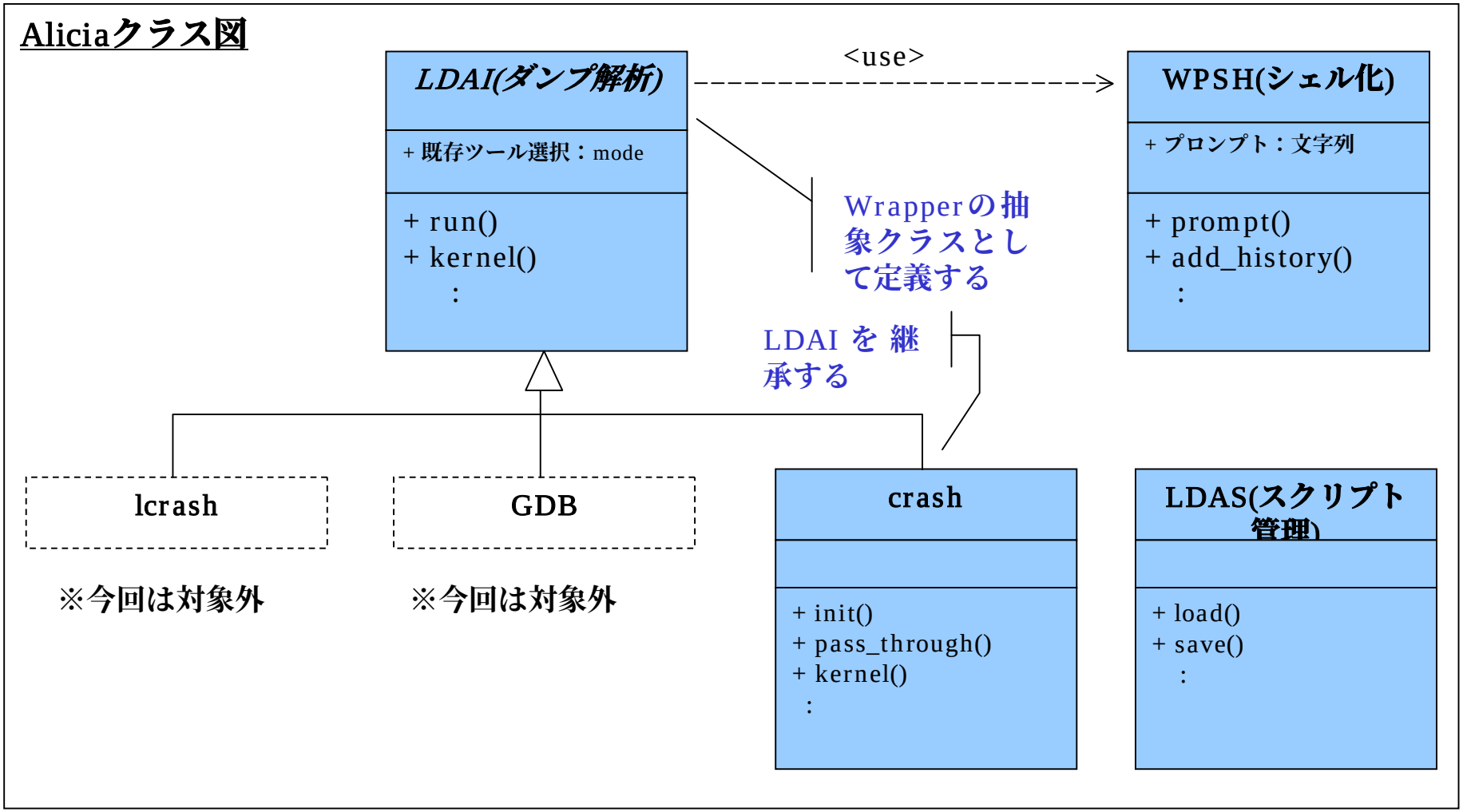
Wrapperモジュール(LDAI)
(Linux Dump Analysis Interface)
- 既存Dump解析ツール(Crash、Lcrash)コマンド群に対する入出力インターフェースの開発
- 既存Dump解析ツールが持っている機能を利用して、Kernel構造体へのアクセスを実現 (Alicia API)



5. Aliciaのデザイン

Alicia自身もPerlで記述

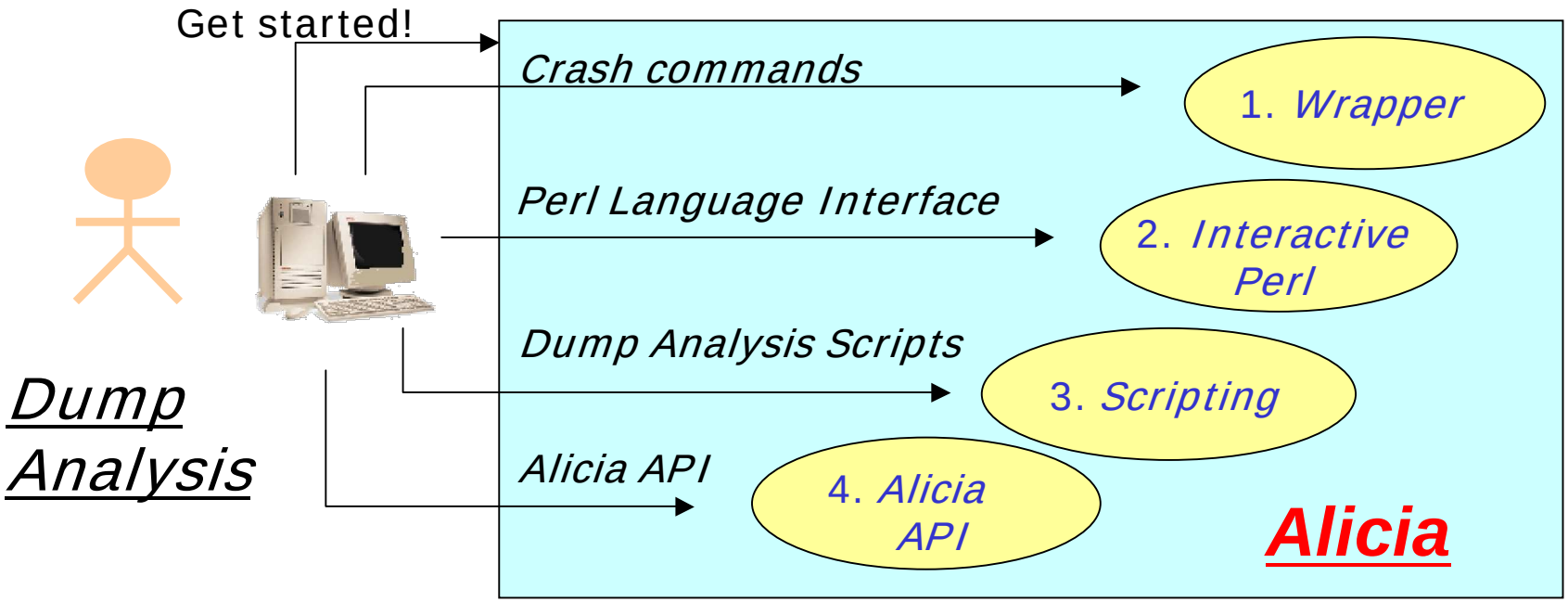
Aliciaクラス図



6. Aliciaの機能

Aliciaの機能は大きく4つ。ツール統合とスクリプト化機能を提供。

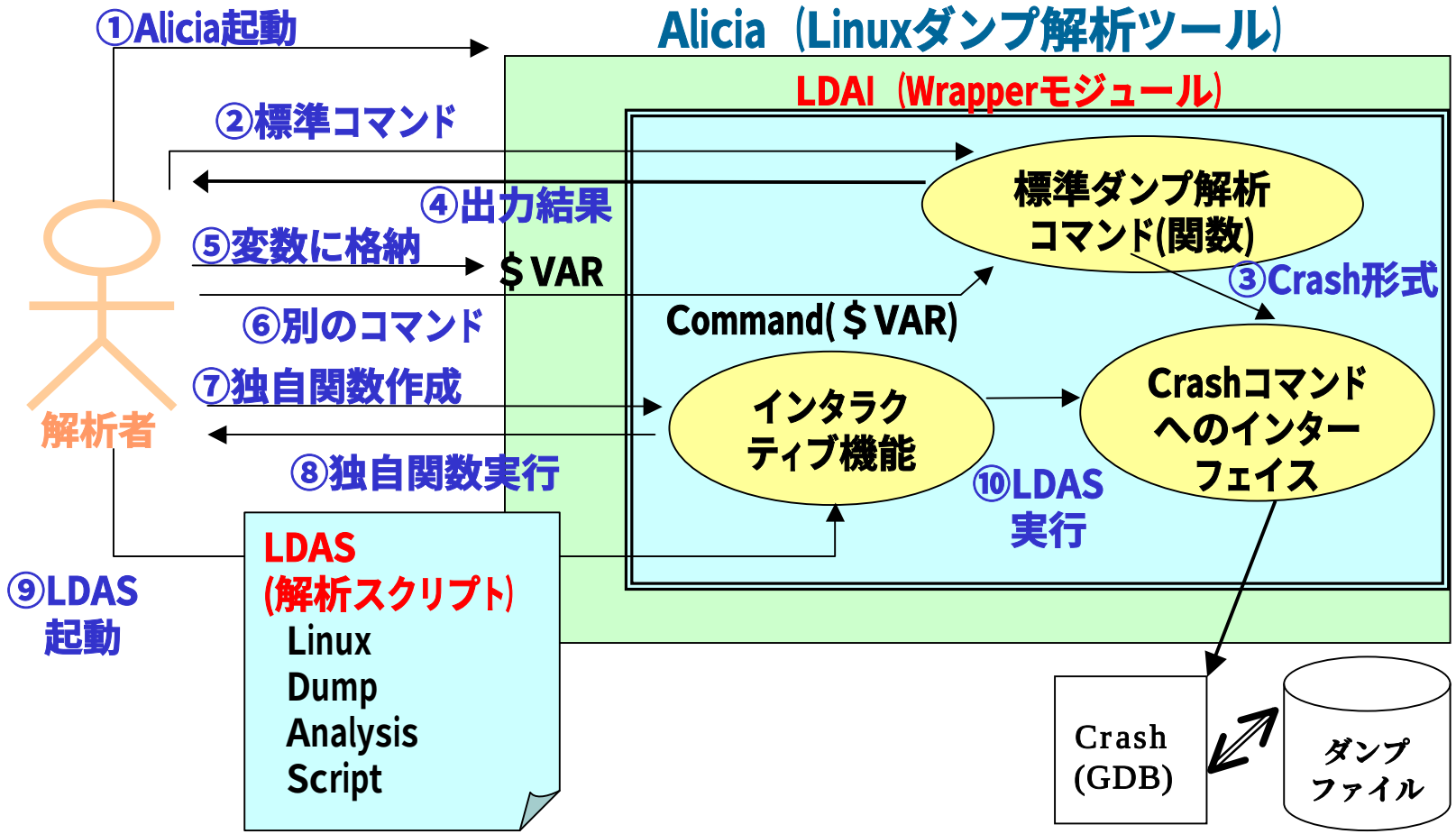
1. Wrapper：既存解析ツールをラッピングし、既存ツールのコマンドセットをそのまま利用可能にする機能
2. Interactive Perl：コマンドの実行結果を変数に格納し別コマンドの入力にするような対話型の解析を可能にする機能
3. Scripting：スクリプト(LDAS)を実行するための機能。独自コマンドをその場で作成しながら対話的に実行することが可能
4. Alicia API：スクリプトの中で利用しやすいAliciaの独自関数機能。全既存ツールに共通な関数



7. Aliciaの操作

Aliciaの操作パターンは大きく3つ。下図はその全体イメージ。

1. インタラクティブに標準コマンドの実行：①→②→③→④→⑤→⑥
2. インタラクティブに独自コマンドの実行：①→⑦→⑧→③～⑥
3. 解析スクリプト(LDAS)の実行：①→⑨→⑩→③～⑥



8. Aliciaの簡易デモ

Alicia(Advanced LInux Crash-dump Interactive Analyzer)
簡易デモ･プロファイル

●ダンプ採取環境：

OS：　Miracle Linux　V3.0(kernel 2.4.21-9.35AX)
ハードウェア：　Unisys ES7000 (CPU: Intel Xeon　2.8GHz×32、Memory: 4GB)
採取状況：　aim7ベンチマーク測定ツールを実行中に、LKCDのダンプ採取機能を利用して採取

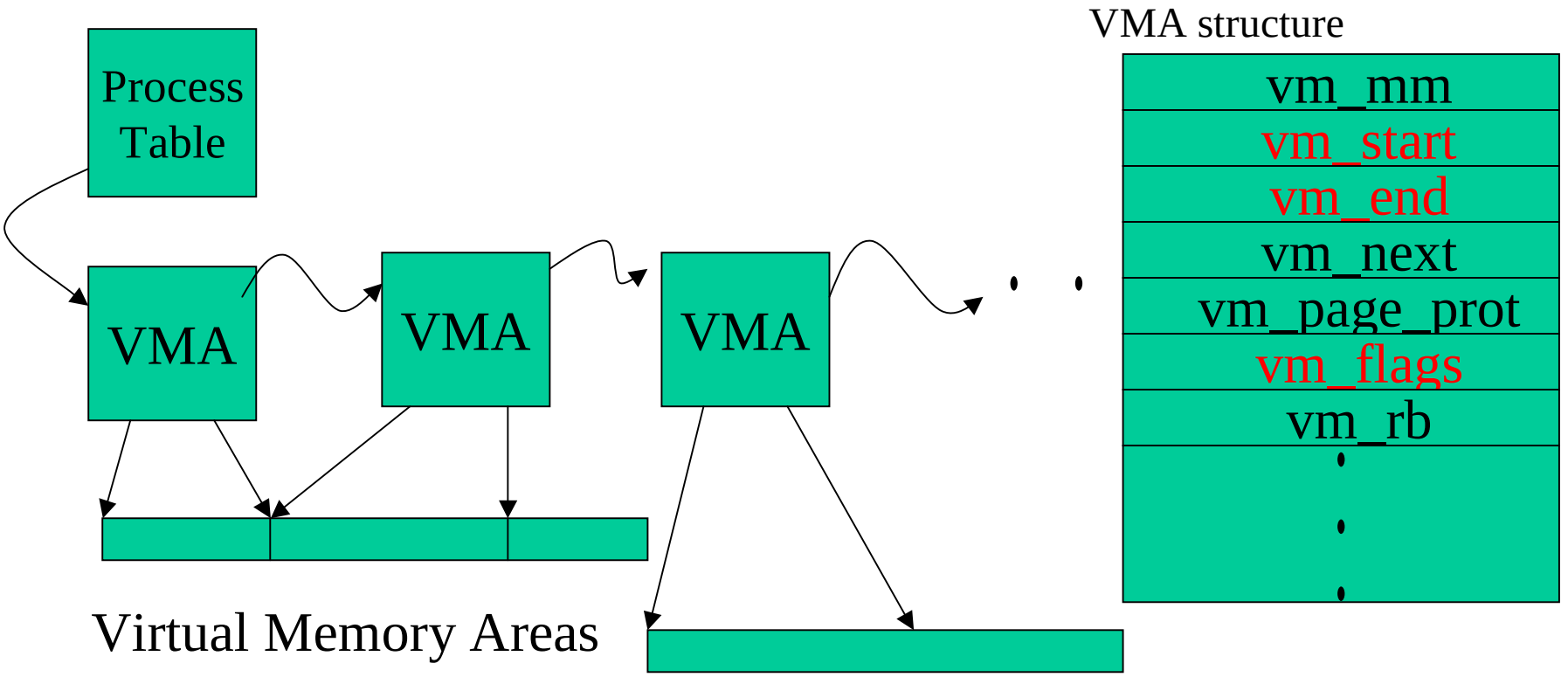
●ダンプ解析環境：

OS：　Miracle Linux　V3.0(kernel 2.4.21-9.35AX)
ハードウェア：　IBM (CPU: Intel(R) Pentium(R) M processor 1300MHz、Memory: 512MB
perl：　v5.8.0 with Term::ReadKey、Term::ReadLine
crash：　3.8-5
Alicia：　1.0.0

9. スクリプト化の例題 （1）

(1) Linked list – Case of VMA(Virtual Memory Area)

指定されたタスクのVMAの情報を表示するLDAS：list_task_vmaをAlicia上にロードし実行する例を示す。



9. スクリプト化の例題 (2)

```
alicia> list_task_vma 'f2894000';

task f2894000
  comm multitask
  pid 30239
  mm 0xf27c5200
    mm.pgd 0xf6879ac0
    mm.mm_count.counter 1
start of vm_next list from 0xf15cc584
  address     vm_start    vm_end      vm_flags
  0xf15cc584   0x8048000   0x8058000      0x1875
  0xf15cc5c8   0x8058000   0x8059000    0x101873
  0xf15cc188   0x8059000   0x8105000    0x100073
  0xf15cc210  0xb7463000  0xb746e000        0x75
  0xf15cc2dc  0xb746e000  0xb746f000    0x100073
  0xf15cc320  0xb747f000  0xb7480000    0x100073
       :
  0xf15cce8c  0xb75e7000  0xb75eb000    0x100073
  0xf15ccf9c  0xb75eb000  0xb7600000       0x875
  0xf15cc71c  0xb7600000  0xb7601000    0x100873
  0xf15cc980  0xbfff8000  0xc0000000    0x100177
end of vm_area_struct.vm_next list
```

※LDASデモ前に以下のコマンド実行要

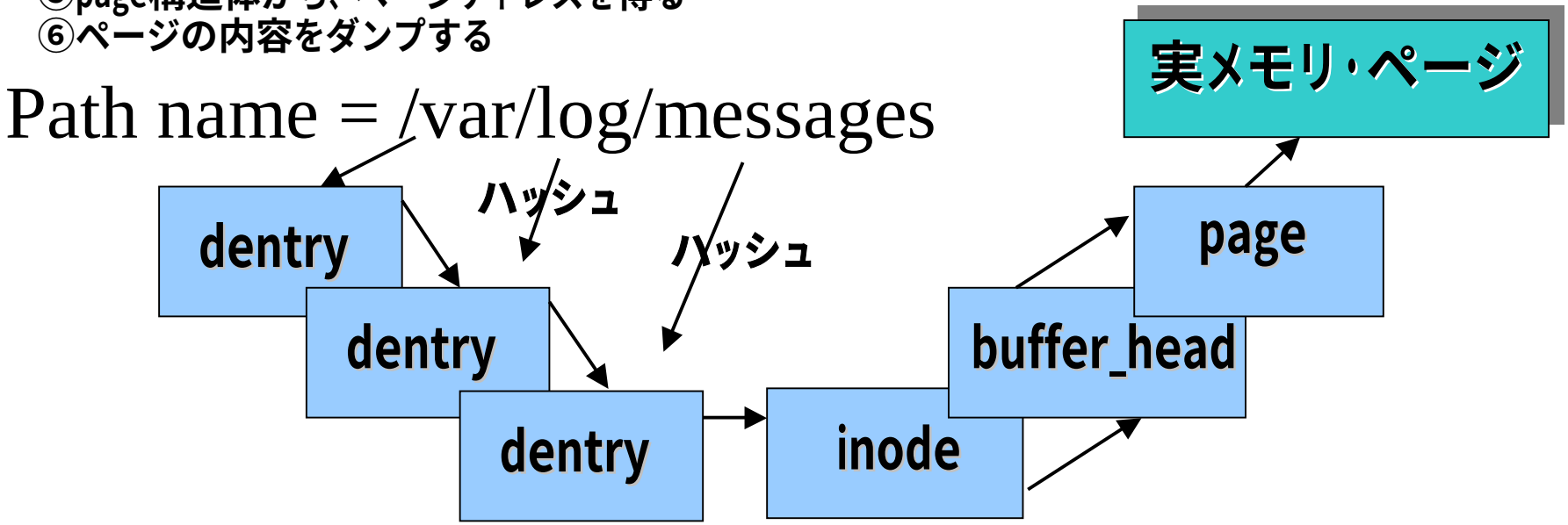
alicia> load_all '/usr/share/ldas'

9. スクリプト化の例題 (3)

(2) Unwritten log data at panic.

カーネルがクラッシュする直前の/var/log/messageに、まだ書き込まれていないイメージを採取する解析手順例を示す。

解析手順:
①指定されたパス名のハッシュを計算し、dentry構造体のアドレスを算出する
②dentry構造体から、inode構造体のアドレスを得る
③inode構造体にリンクされているdirtyバッファのアドレスを得る
④dirtyバッファが存在するなら、そのpage構造体のアドレスを得る
⑤page構造体から、ページアドレスを得る
⑥ページの内容をダンプする



9. スクリプト化の例題 (4)

```
alicia> retrieve('/var/log/messages');

ff081000:  69686320 7920646c 676e756f 6f207265   child younger o
ff081010:  7265646c 74634f0a 20392020 303a3130   lder.Oct  9 01:0
ff081020:  30333a39 6e6c6d20 6b203178 656e7265   9:30 mlnx1 kerne
ff081030:  69203a6c 2074696e 20202020 20202020   l: init
ff081040:  43205320 42463430 20303832 20202020    S C04FB280
ff081050:  20202034 20312020 20202020 20203020   4     1      0
ff081060:  34332020 20202020 32202020 20202020     34       2
ff081070:  28202020 4c544f4e 4f0a2942 20207463      (NOTLB).Oct
ff081080:  31302039 3a39303a 6d203033 31786e6c   9 01:09:30 mlnx1
ff081090:  72656b20 3a6c656e 6c614320 7254206c    kernel: Call Tr
ff0810a0:  3a656361 5b202020 3130633c 37623932   ace:   [<c0129b7
                          :
                          :
                          :
```

※retrieveはデモ用コマンドでありAliciaリリース版には含まれていない。実行するには、簡易版として、

sub retrieve {less LDAS::Data::get_dirty_data(@_);}1

をLDASとして作成し、ロードする。

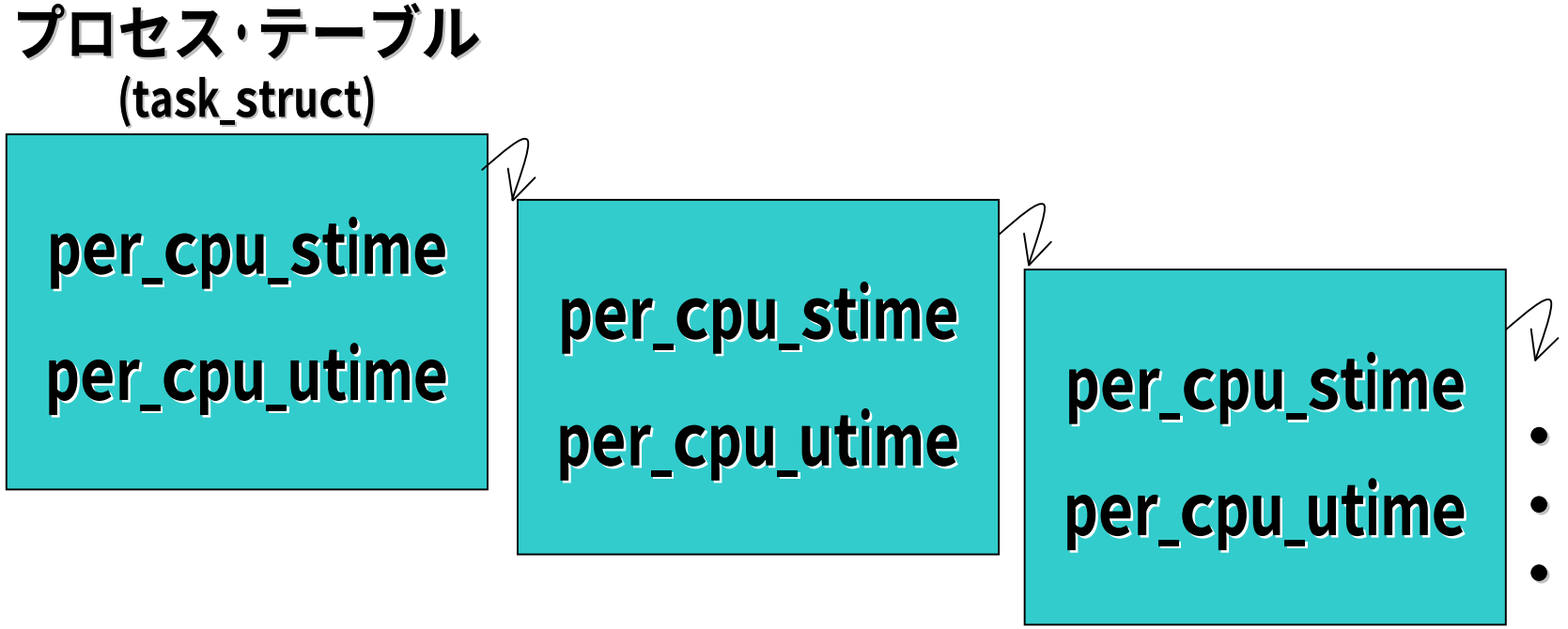
TIPS !

Aliciaのヒストリ機能を用いることにより、ダンプ解析時にキーインした内容を元に、簡単にLDASスクリプトを作成することができる。入力時のTab補完機能も利用できる。

9. スクリプト化の例題 (5)

(3) CPU resource.

あるプロセスが大量にCPUを使用していないか調査したい。ここでは、psコマンドの結果に、それぞれのプロセスのCPU使用時間を付け加え、使用時間順にソートして表示するスクリプトの実行例を示す。



9. スクリプト化の例題 (6)

```
alicia> ps_cpu();
  PID    PPID  CPU   TASK     ST  %MEM     VSZ    RSS  Cputime  COMM
 2090       1    0  f42dc000  RU   0.0    3044   1672    22.79 oprofiled
28657   28233   25  ef2da000  UN   0.0    2328   1340    11.82 multitask
30210   28233   21  e9ff2000  RU   0.0    3352   1404    10.68 multitask
 1271    1264    5  f47f6000  IN   0.2   49384   6932    10.58 ocssd.bin
29229   28233    6  f0ec4000  RU   0.0    3336   1388    10.11 multitask
28480   28233   23  e9bd0000  UN   0.0    2328   1340     9.78 multitask
29582   28233    3  ebcc6000  UN   0.0    2328   1340     9.78 multitask
29004   28233   24  e8fc4000  UN   0.0    2328   1340     9.51 multitask
28686   28233    0  f2776000  UN   0.0    2328   1340     9.42 multitask
30188   28233    3  e9f68000  RU   0.1    3308   2320     9.30 multitask
                                    :
                                    :
```

※ps_cpuはデモ用コマンドでありAliciaリリース版には含まれていない。実行するには、簡易版として、

sub ps_cpu {less join("\n",LDAS::Cputime::cputime_all());}1

をLDASとして作成し、ロードする。

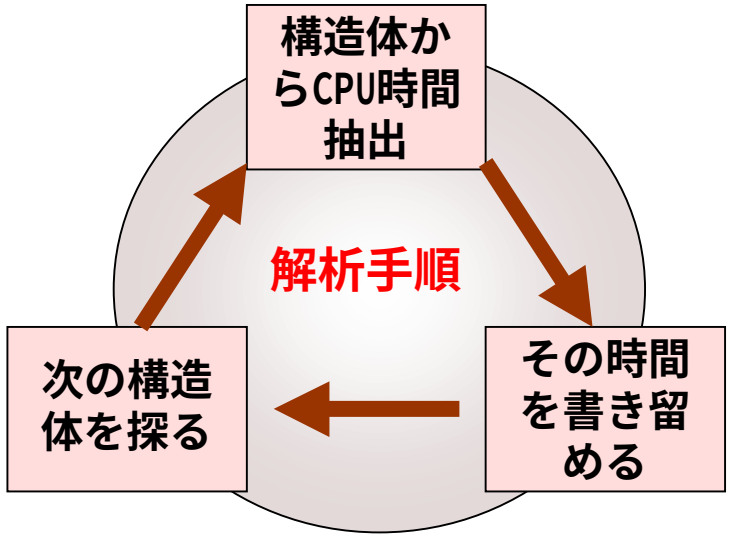
TIPS ! Aliciaは、LDASの中から別なLDASを呼び出すことができる。

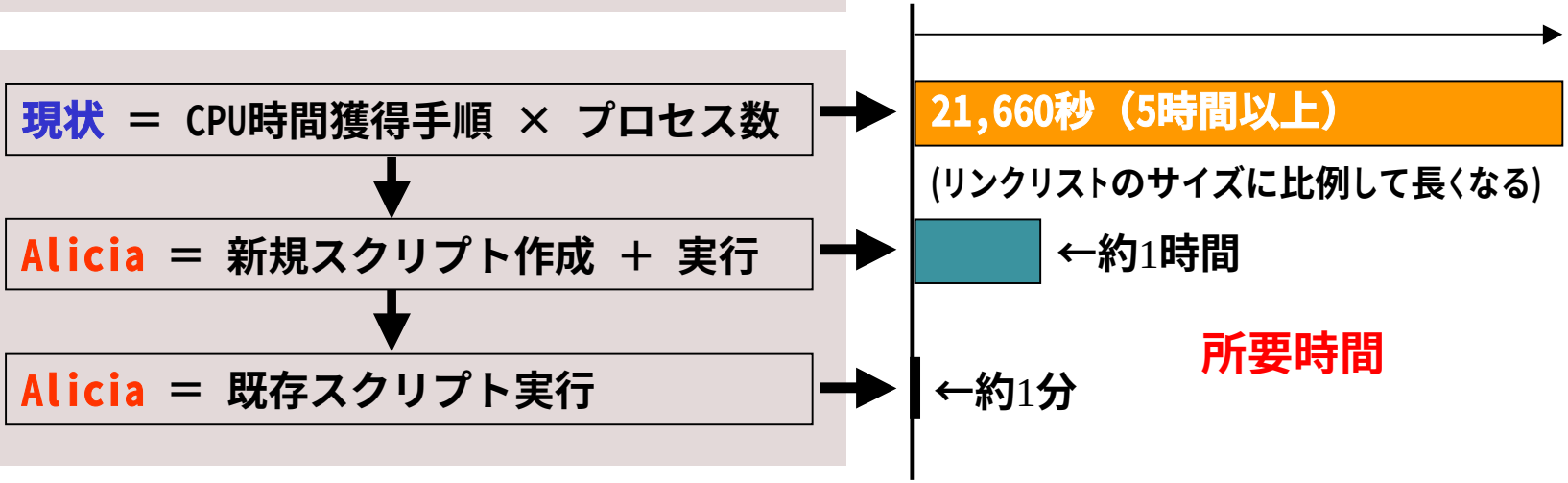
9. Aliciaによる効果 (7)

Aliciaによる解析手順の高速化例

ダンプ解析者の要求：
今すぐ、task_struct構造体から全プロセスのCPU使用時間を抽出し、PSコマンドの出力に付加して表示したい！
- 実ダンプ採取環境：32CPUs，4GB
- 実プロセス数 ＝ 2166

推定所要時間：
- CPU時間獲得手順の時間 ＝ 約10秒
- スクリプト作成時間 ＝ 約1時間
- スクリプト実行時間 ＝ 約1分





10. まとめ

2004年度の開発結果の評価

● ダンプ解析ノウハウを蓄積/共有するための統合型ダンプ解析ツールとして

- Alicia本体部分とcrashインタフェースの開発を完了
- 解析ノウハウを蓄積/共有するための手段を提供

● Aliciaによるダンプ解析の高速化に向けて

- ダンプ解析作業全体の高速化はまだ実証できていない。(事例の蓄積が必要)
- ダンプ解析作業全体の高速化にはLDASの蓄積が必須
- ツールを使いこなすのは人間。Linuxダンプ解析者の育成が重要
- 現行メモリダンプに不足している情報の洗い出しが必要　(トレース/ログ等)

● 企業間協力による開発成果 – シナジー効果を発揮

- 3社による協力体制により、開発速度、品質面のブラッシュアップに大きな効果を発揮
- Alicia本体機能の開発範囲拡大と操作性/信頼性の向上を実現

11. 今後の課題

日本発のスタンダード･ダンプ解析ツールを目指して

- **lcrash対応**
  ダンプ解析ツール統合化に向けてlcrashユーティリティへの対応を実施する。すでにAlicia本体は完成しているが、lcrashインタフェース部分の実装が必要。

- **Aliciaの普及**
  ･LDASの充実が最大のカギ
  ･LDAS作成にあたってのTIPSの公開
  ･蓄積されたLDASの管理/検索機能の提供
  ･LDASの実環境での成果/効果の公開
  ･Unixのダンプ解析ツール（mdb/adb/crash）との比較検証とフィードバック

- **デフォルト･ダンプ編集スクリプトの提供**
  ･どんな種類の障害に対しても、デフォルトで入手しておきたいメモリダンプ情報を自動的に編集するLDASの作成と公開、客先配布

- **ダンプ採取オペレーションに対する情報公開**
  ･ダンプ解析の重要性をアピールしていくためのファーストステップとして、ダンプ採取方法等の情報公開

クラッシュダンプ解析ツール「Alicia」の開発

Alicia関連URL

- ダンプデータ解析ツール（Alicia)の評価と考察
  - http://www.ipa.go.jp/software/open/forum/
- 「Alicia」プロジェクト
  - http://sourceforge.net/projects/alicia/
- 「Alicia」ホームページ
  - http://alicia.sourceforge
- 「Alicia」のプレスリリース
  - http://www.uniadex.co.jp/news/syosai/n1_20050322_02.html

クラッシュダンプ解析ツール「Alicia」は、「独立行政法人　情報処理推進機構オープンソースソフトウェア活用基盤整備事業」に係る委託業務の一環として開発したものです。