Development of Crash Dump Analysis Tool “Alicia”

April 15, 2005

Hideki Takahashi (UNIADEX, Ltd.)
Member of Development Infrastructure WG
Japan OSS Promotion Forum
1. Background of Alicia development

Dump Analysis for mainframe

- Problem is resolved in most cases if crash dump was taken.
  - Dump analysis is a critical factor for mission critical support.
  - Most of necessary information is being saved in memory by operating system.
  - Crash dump can be saved under any condition. (No failure for taking a dump.)
  - Dump tool is easy to use and it has many useful functions.

Dump Analysis for Linux

- Dump Analysis hasn’t been useful for resolving problems.
  - It’s very difficult to find a dump analysis report for Linux systems.
  - There’s a possibility to lose memory data at system fault due to failure on dumping.
  - Efficiency of dump analysis is bad because dump tool itself is poor.
### Purpose

Target is to provide the mainframe class maintainability to Linux. To realize a **mission critical Linux**, implement a mechanism for **know-how sharing** and dump analysis time reducing.

### Current Problems

- Dump format is not regulated and different among distributions.
- Dump analyzer must take care of dump format for his analysis.
- Dump analyzer can’t have necessary data quickly.
- Simple operation must be repeated and data must be put in order for getting target information.
- Analysis know-how is never shared among engineers since technique of dump analysis is just a personal skill.

### Improving Items

- Define a common interface to the existing dump analysis tool (crash/\lcrash) and develop a new tool that can handle dump analysis scripts.

### Actions

- Dump Analysis Environment must be improved
  - Dump environment that is not dependent on dump format.
  - Dump environment where new command is added easily.
  - Dump environment where analysis procedure can be saved and shared.
3. Strong Point of Alicia

- Dump Analyzer can have many advantage in operations. (Analysis time can be reduced.)
- Dump Analyzer can ask non kernel experts for dump editing*. (Customers can use Alicia.)
- Dump Analyzer can verify actual data at kernel study and can learn expert’s know-how.
- Dump Analyzer can use functions of existing analysis tools seamlessly. Alicia is wrapping the existing tools and it is not built from the scratch for getting merits of each tool.

*Dump editing : To visualize necessary information by reading necessary data from memory dump.

Advantage of Alicia

- Ease of use
- Programmable
- High speed edit
- Accumulation of analysis know-how
- High extend-ability
- Powerful functions
4. Positioning and main components of Alicia

Development of integrated dump analysis tool “Alicia”

Know-how accumulation:
- LDAS is an interpreter and has a good visibility and it can be accumulated on a know-how D/B.
- Modification is easy and new command can be made quickly by just modifying existing scripts during dump analysis.

Perl Scripts (LDAS)
(Linux Dump Analysis Scripts)
- New analysis commands will be developed by using the Wrapper Module (LDAI).
- Dump for similar problem is analyzed quickly by making scripts needed for the dump analysis and reusing them.

Wrapper Module (LDAI)
(Linux Dump Analysis Interface)
- Interfaces to existing dump analysis tools (crash, lcrash) will be developed.
- Access methods to kernel structure will be implemented by using functions of existing dump analysis tools. (called Alicia API)
5. Design of Alicia

Alicia itself is written by Perl

LDAI class diagram

LDAI
+existing tool : mode
+run()
+kernel() :

WPSH (shell)
+prompt : characters
+prompt()
+add_history() :

Abstract class of LDAI

Inheritance of LDAI

crash
+init()
+pass_through()
+kernel() :

GDB

Icrash

PKAS

Out of scope this time

Out of scope this time
Alicia has 4 major functions. - Integrated dump analysis tool -

1. **Wrapper function:** User can use existing crash/lcrash commands in addition to Alicia commands.
2. **Interactive Perl function:** User can analyze dump interactively and can have command results as a variable and can use the variable for another command.
3. **Scripting function:** User can execute LDAS (Perl script) interactively and can use LDAS in another LDAS.
4. **Alicia APIs:** User can use Alicia’s unique functions that are common to crash/lcrash.

---

![Diagram](Image)
7. Operations of Alicia

There are 3 patterns in Alicia operation.

1. Interactive Standard Command Analysis
   - Alicia starts
   - Standard command
   - Save in variable
   - Command with variable
   - Make own command
   - Execute own command
   - Results

2. Interactive Alicia Unique Command Analysis
   - Alicia starts
   - Make own command
   - Execute own command
   - Command with variable
   - Save in variable
   - Standard command
   - Results

3. Alicia scripts (LDAS) Analysis
   - LDAS starts
   - Execute own command
   - Make own command
   - Command with variable
   - Save in variable
   - Standard command
   - Results

There are 3 patterns in Alicia operation.
8. Demonstration of Alicia

Alicia (Advanced Linux Crash-dump Interactive Analyzer) Demonstration Profile

Environment that the dump was taken:
OS: Miracle Linux V3.0(kernel 2.4.21-9.35AX)
Hardware: Unisys ES7000 (CPU: Intel Xeon 2.8GHz × 32, Memory: 4GB)
Situation: LKCD dump was taken while the aim7 benchmark tool is running.

Environment that dump is analyzed today’s demo:
OS: Miracle Linux V3.0(kernel 2.4.21-9.35AX)
Hardware: 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
(1) Linked list – Case of VMA (Virtual Memory Area)

Get necessary information from each VMA using the chain pointer.
### 9. Demonstration Scenario (2)

#### デモンストレーションシナリオ (2)

#### デモンストレーションシナリオ (2)

<table>
<thead>
<tr>
<th>ステップ</th>
<th>内容</th>
<th>サブステップ</th>
<th>注意点</th>
</tr>
</thead>
<tbody>
<tr>
<td>ステップ1</td>
<td>内容1</td>
<td>サブステップ1</td>
<td>注意1</td>
</tr>
<tr>
<td>ステップ2</td>
<td>内容2</td>
<td>サブステップ2</td>
<td>注意2</td>
</tr>
<tr>
<td>ステップ3</td>
<td>内容3</td>
<td>サブステップ3</td>
<td>注意3</td>
</tr>
<tr>
<td>ステップ4</td>
<td>内容4</td>
<td>サブステップ4</td>
<td>注意4</td>
</tr>
</tbody>
</table>

**9. Demonstration Scenario (2)**

### デモンストレーションシナリオ (2)

#### アリシアリストVMの例

**task f2894000**

```
alicia>list_task_vma'f2894000';
task f2894000
  comm multitask
  pid 30239
  mm 0xf27c5200
  mm.pgd 0xf6879ac0
  mm.mm_count.counter 1
  start of vm_nextlist from 0xf15cc584
```

<table>
<thead>
<tr>
<th>アドレス</th>
<th>スタート</th>
<th>エンド</th>
<th>フラグ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xf15cc584</td>
<td>0x8048000</td>
<td>0x8058000</td>
<td>0x1875</td>
</tr>
<tr>
<td>0xf15cc5c8</td>
<td>0x8058000</td>
<td>0x8059000</td>
<td>0x101873</td>
</tr>
<tr>
<td>0xf15cc188</td>
<td>0x8059000</td>
<td>0x8105000</td>
<td>0x100073</td>
</tr>
<tr>
<td>0xf15cc210</td>
<td>0xb7463000</td>
<td>0xb746e000</td>
<td>0x75</td>
</tr>
<tr>
<td>0xf15cc2dc</td>
<td>0xb746e000</td>
<td>0xb746f000</td>
<td>0x100073</td>
</tr>
<tr>
<td>0xf15cc320</td>
<td>0xb747f000</td>
<td>0xb7480000</td>
<td>0x100073</td>
</tr>
<tr>
<td>0xf15cce8c</td>
<td>0xb75e7000</td>
<td>0xb75eb000</td>
<td>0x100073</td>
</tr>
<tr>
<td>0xf15ccf9c</td>
<td>0xb75eb000</td>
<td>0xb7600000</td>
<td>0x875</td>
</tr>
<tr>
<td>0xf15cc71c</td>
<td>0xb7600000</td>
<td>0xb7601000</td>
<td>0x100873</td>
</tr>
<tr>
<td>0xf15cc980</td>
<td>0xbfff8000</td>
<td>0xc0000000</td>
<td>0x100177</td>
</tr>
</tbody>
</table>

#### アップデータとデータベースのシナリオ

- アップデータをデータベースに保存
- データベースをデータベースに移動
- データベースをデータベースに保存
- データベースをデータベースに移動
- データベースをデータベースに保存
- データベースをデータベースに移動

### エンドオブアリシアリストVMの次リスト

---

Copyrigh © 2005, Development Infrastructure WG, Japan OSS Promotion Forum
(2) Unwritten log data at panic.

Retrieves dirty buffer where data hasn’t been written into /var/log/messages just before panic by using Alicia APIs, LDASs and crash commands.

1. Make a hash for the specified path and get the address of structure "dentry".
2. Get the address of structure "inode".
3. Get the address of dirty buffer linked to the structure "inode".
4. Get the address of structure "page", if it is dirty buffer.
5. Get the page address from the structure "page".
6. Dump the image of page.

Path name = /var/log/messages
TIPS!

You can make an LDAS script easily by using Alicia’s history function which saves your key-in images during your dump analysis. You also can use the tab complement function for easy input.
11. Demonstration Scenario (1)

(3) CPU resource.

Get information about which processes have a lot of CPU time. We will get the CPU time of each process from the task_struct and add the CPU time to original “ps” command display and then sort them.
Alicia can invoke another LDAS from an LDAS.
12. Effect provided by Alicia

Example of dump analysis acceleration by Alicia

Request from dump analyst:
He wants to pick up CPU time usage from the task_struct for all processes and append the CPU time data to standard PS command output right now!
- Dump actually taken: 32 CPUs, 4GB
- Actual number of processes = 2166

The time required:
- Time for acquiring CPU time = 10 secs
- Time for making LDAS = 1 hour
- Time for executing LDAS = 1 min

Current = Acquiring CPU time # of processes
Alicia = Making LDAS + Executing LDAS
Alicia = Executing existing LDAS

Get CPU time from structure
Take a memo for analysis
Get next link address

Time required

- About 1 hour
- About 1 minute

(More than 5 hours)

(It depends on the length of linked list.)
13. Result Summary for 2004

Result summary of Alicia development in FY2004

- Implemented as an integrated dump analysis tool for accumulating and sharing the dump analysis know-how.
  - Development of Alicia main structure and crash interface: Completed.
  - Infrastructure for know-how accumulation and sharing: Now available.
  - Script (LDAS) samples for dump analysis procedures: Included in Alicia 1.0.0

- Dump analysis speed is actually accelerated by Alicia
  - Execution time without using LDAS was measured
  - Execution time with making and executing LDAS was measured
  - Execution time with using existing LDAS was measured
  - Speed-up was evaluated from the point of view of operability and serviceability
  - Sample LDASs like “Linked List Search”, “Display of unwritten log data at panic”, etc. were made and executed for measurement.

- Synergy effects by co-working approach in development
  - Development speed and quality were brushed up by cooperation between 3 companies
  - Development scope were expanded and operability/reliability were improved
14. Activities in 2005 and future

To be a standard dump analysis tool initiated from Japan

- **Development of lcrash interface**
  lcrash interface must be made for completing the integrated dump analysis tool development. *lcrash interface will be made in first half of 2005.*

- **Popularization of Alicia**
  - Key factor is to make useful LDASs for quick dump analysis.
  - TIPs for making LDAS will be collected and released.
  - LDAS database (administration & search functions) should be provided.
  - Dump analysis reports by using LDAS will be opened to the public.
  - Comparison with Unix dump tools (mdb/adb/crash) and feedback of them.

- **Development of a default dump edit script**
  - LDAS for editing minimum set of memory information that is commonly required for most cases of Linux problems should be developed and released.

- **Release of dump related information including “how to”**
  - Release information about how to take a dump, etc. to appeal the importance of dump analysis as first step to our goals.
15. Information about Alicia

Alicia 1.0.0 release date: March 22, 2005 (Plan)
Copyright: UNIADEX
License: GPL (General Public License)
Open Source: Source code will be opened on SourceForge.net provided by OSDN Open Source Development Network)

Co-working company: NTT DATA
MIRACLE LINUX

URLs for downloading Alicia

- “Alicia” project
  - http://sourceforge.net/projects/alicia/
- “Alicia” homepage
  - http://alicia.sourceforge

The development of this program is partly supported by IPA (Information-technology Promotion Agency), Japan
Help from China and Korea would be appreciated!