

# Linux 障害解析の状況と今後の 展望

Oct.15, 2004

NTT Data 先端技術株式会社

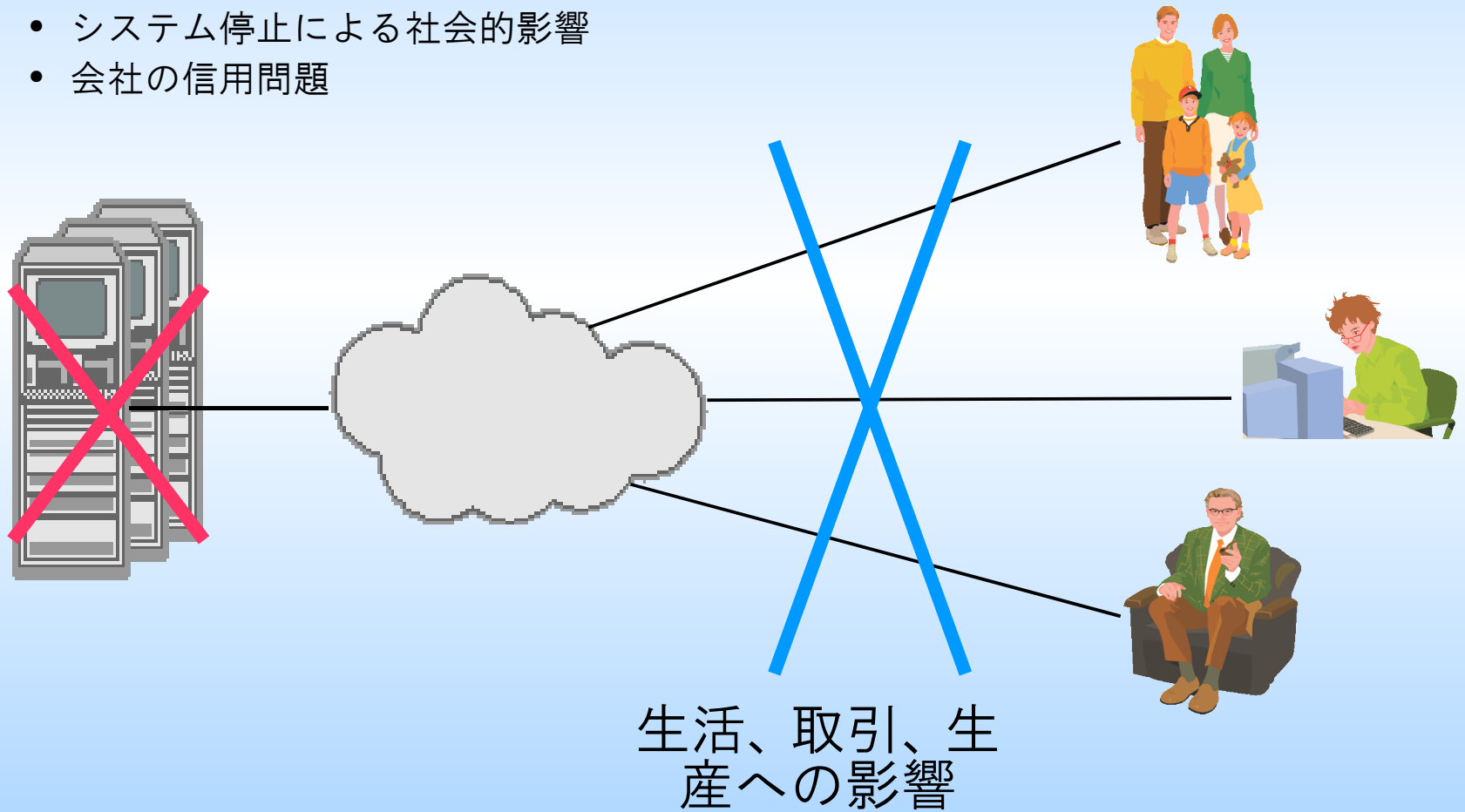
鈴木幸市

koichi@intellilink.co.jp

エンタープライズ環境で Linux の利用を促進するうえで、カーネル障害解析は重要な位置を占める。カーネルダンプとしてはLKCD や NetDump が知られているが、より高い確実性、信頼性を目指した Linux カーネルダンプ取得ツールの提案もなされてきている。これらの動向について紹介する。

# 背景：なぜ障害解析？

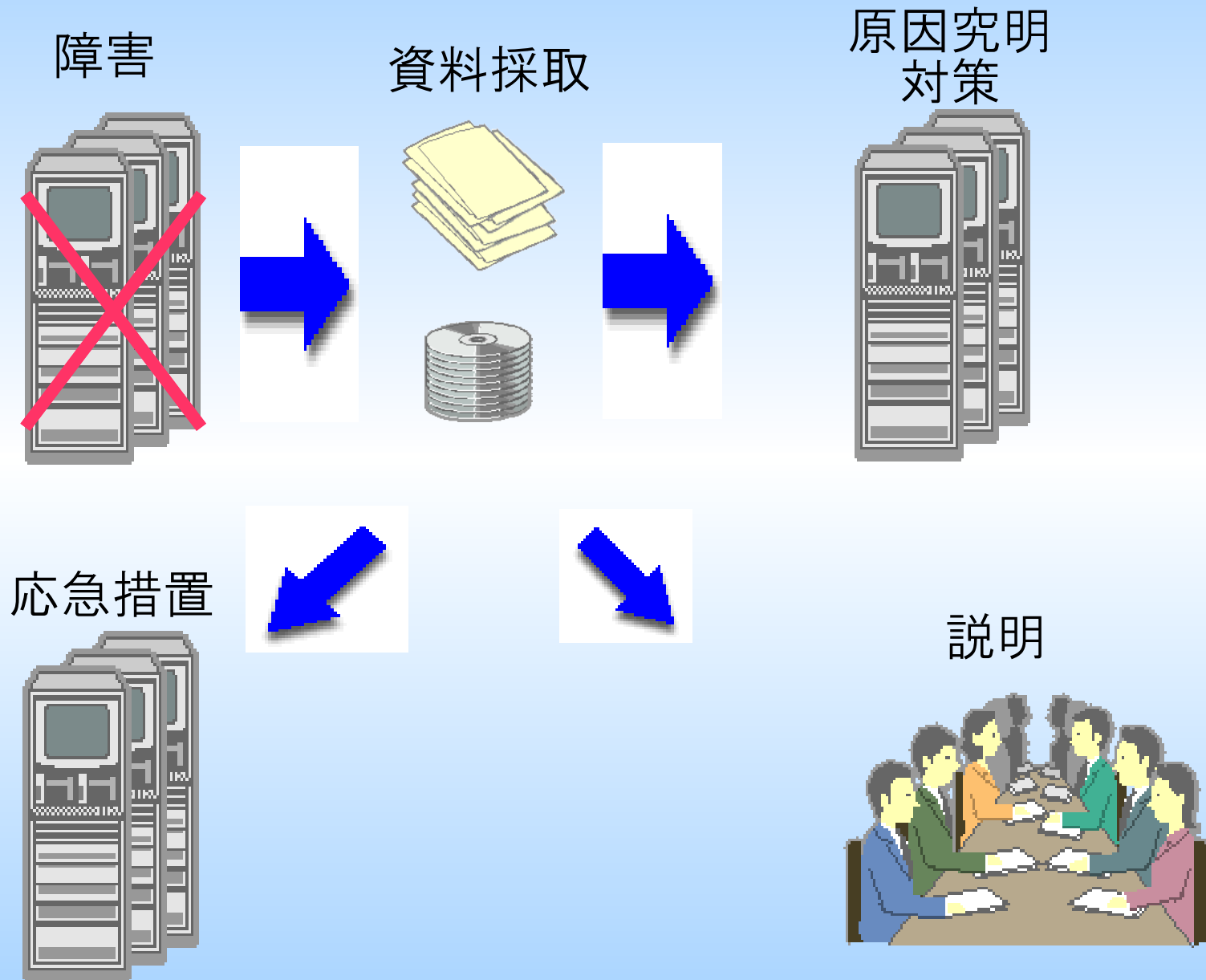
- エンタープライズや政府機関での利用、ミッションクリティカルなアプリケーションへの適用の増大
  - 高信頼化、高可用性追求も重要であるが
  - 障害時の対応が非常に重要になりつつある
    - システム停止による社会的影響
    - 会社の信用問題



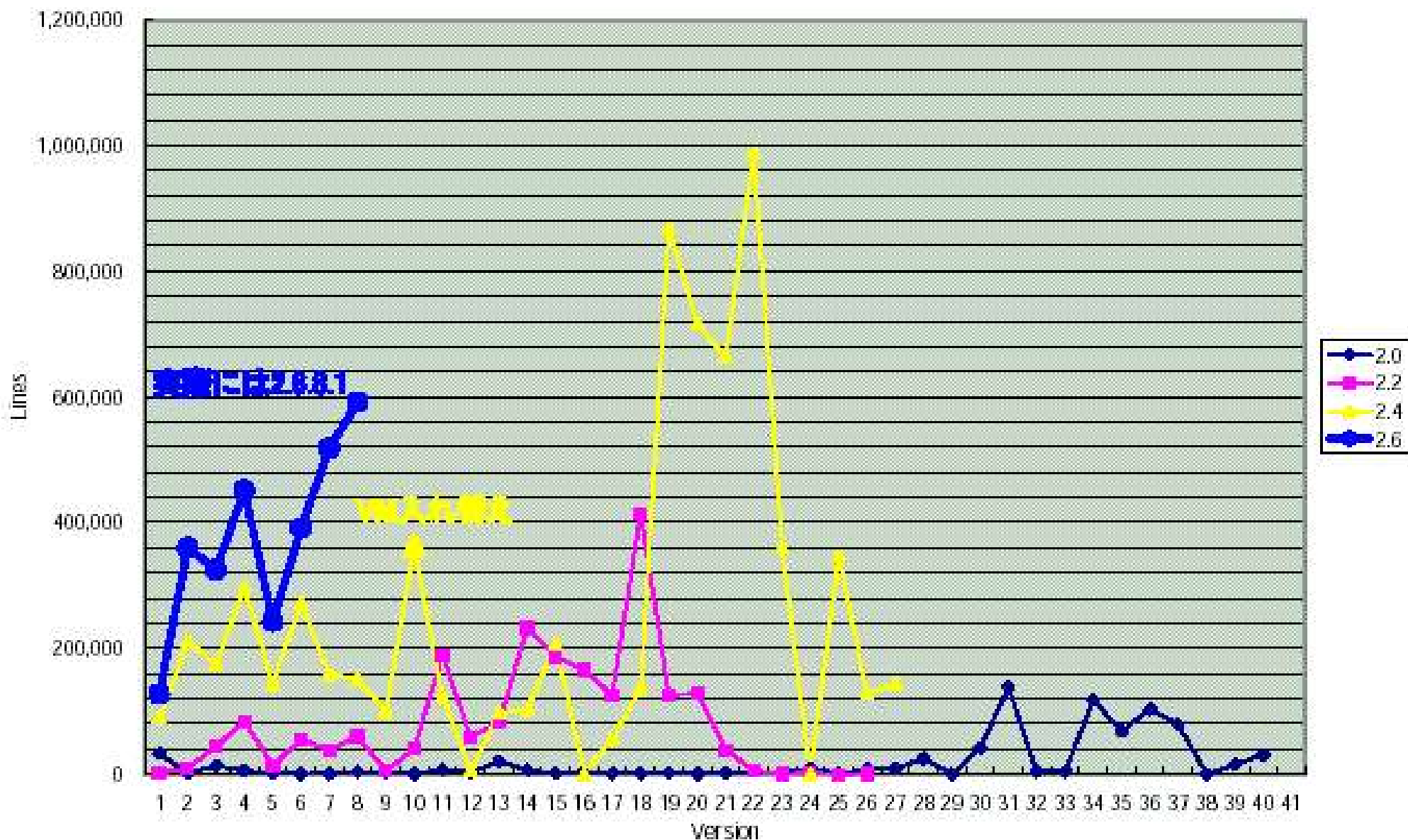
## 背景：なぜ障害解析？(cont.)

- システムが停止したら
  - リポートして動いただけでは対応は不十分
  - 障害原因と対策の説明責任
    - 障害解析が必須の機能
    - 今後同様な障害が起こらないようにどのような対策を講じるか
    - その対策は十分か
    - きちんと原因がわかっていることが大前提
    - ソフト障害でも緊急にバグがとれない場合、回避策を講じる必要がある
- システムが停止しなくとも
  - クラスタノードの障害など、システム全体は停止しないが部分的な障害は起こる
  - これらの不具合の原因を究明し、システム停止に至らないように対策を講じることが重要

# 背景：なぜ障害解析？ (cont.)



# カーネルは安定しているか？



# LINUX 故障解析の現状

- エンタープライズシステムでは、故障が発生した場合、原因を素早く特定し、迅速に対応することが求められている
- 障害を解析することによって
  - ハードとソフトの切り分け
  - カーネルとアプリケーションの切り分けを行わなければならない
- 原因の究明や切り分けのためには、カーネルのダンプやトレースデータが必須である
  - クラッシュダンプによる原因究明が有効
- しかし、L I N U X には標準でダンプ機能がない

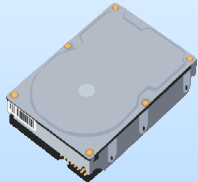
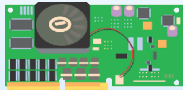
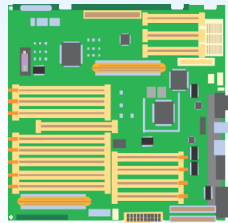
# LINUX 故障解析の現状

## システム障害の事象例

- カーネルパニックが発生してシステムが停止する・・・しかし再現性がない
- システムが突然スローダウンし、果てにはハングする
- 方式検討を行い環境構築してみたが、どうも思ったような性能が出ない
- 原因特定の手段は？
  - (カーネルが原因だと思われる場合) KDB や KGDB によるデバッグ
    - ▶ すぐさま再起動が必要な状況であるため、システムを止めて作業することはできない
    - ▶ たとえ可能だとしても、作業者がその場に赴いて全て解析するのは難しい
  - 同じ環境を用意して、再現待ち → KDB 等で解析
    - ▶ 再現してくれるとは限らない
  - ダンプ解析を行う
    - ▶ LINUX には標準でメモリダンプ機能がない
    - ▶ ダンプが取れない場合がある

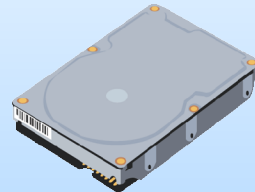
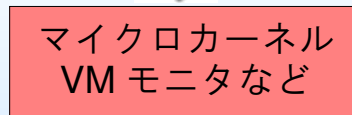
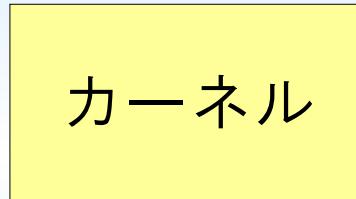
# カーネルダンプの取得

- カーネル自身がダンプをイニシエートする必要がある
    - マイクロカーネルのように、上位でカーネルの動作を制御する機構がない
    - ハードウェアサポートもない
      - メインフレーム等のLPARを使うと外部でダンプがとれるがIAアーキテクチャのハードではこのようなものはほとんどない
- (絵を入れる)



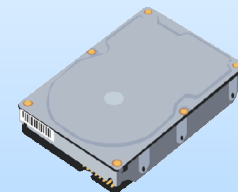
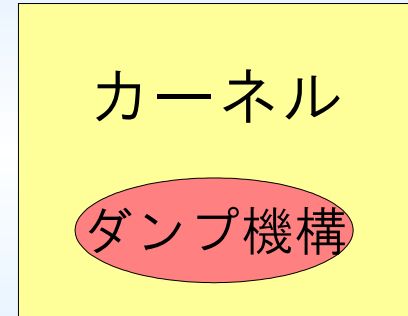
ハードウェアを用いた  
カーネルダンプ

メインフレーム  
など



マイクロカーネル等による  
カーネルダンプ

Mach, VMWare  
など



カーネル自身による  
カーネルダンプ

ほとんどのLinux  
実装



## カーネルダンプの取得 (cont.)

- 信頼性のあるダンプをとる必要がある
  - ダンプ取得中にメモリの状態が変化してしまい、ダンプ内容の一貫性が失われないようにする必要がある
- 確実にダンプをとる必要がある
  - 障害が発生したがダンプが取れないケースは最小限にとどめる必要がある
- ユーザ空間の再現も必要である
  - アプリケーション起因で障害が発生するケースもある
  - メモリダンプと共に、スワップを保全しておけばユーザ空間の再現は可能

# カーネルダンプ用既存ツール (1)

## LKCD

- 最初に SGI が開発し、オープンソースとして公開したもの
- その後、IBM、NEC、日立、富士通共同で拡張を行った
- クラッシュダンプ採取のしくみと解析ツールからなる

### 長所

- 活動が活発である
- ディスクダンプ、ネットワークダンプ、メモリダンプと 3 つの手段がサポートされている。

### 短所

- ディスクダンプでデバイスがビジー (使用中) だとダンプが取れない
- ディスクダンプの場合、割り込み許可して処理するので、他の割り込みと同時に処理が行われる。そのため、資源の競合やダンプデータの不整合が起こり得る。

## カーネルダンプ用既存ツール (2)

### diskdump

- 富士通が開発したディスクダンプを実現する仕組み
- SCSI のポーリングモードを用い ( 割り込みは使わない )、時  
サーバの HDD へダンプを採取

#### 長所

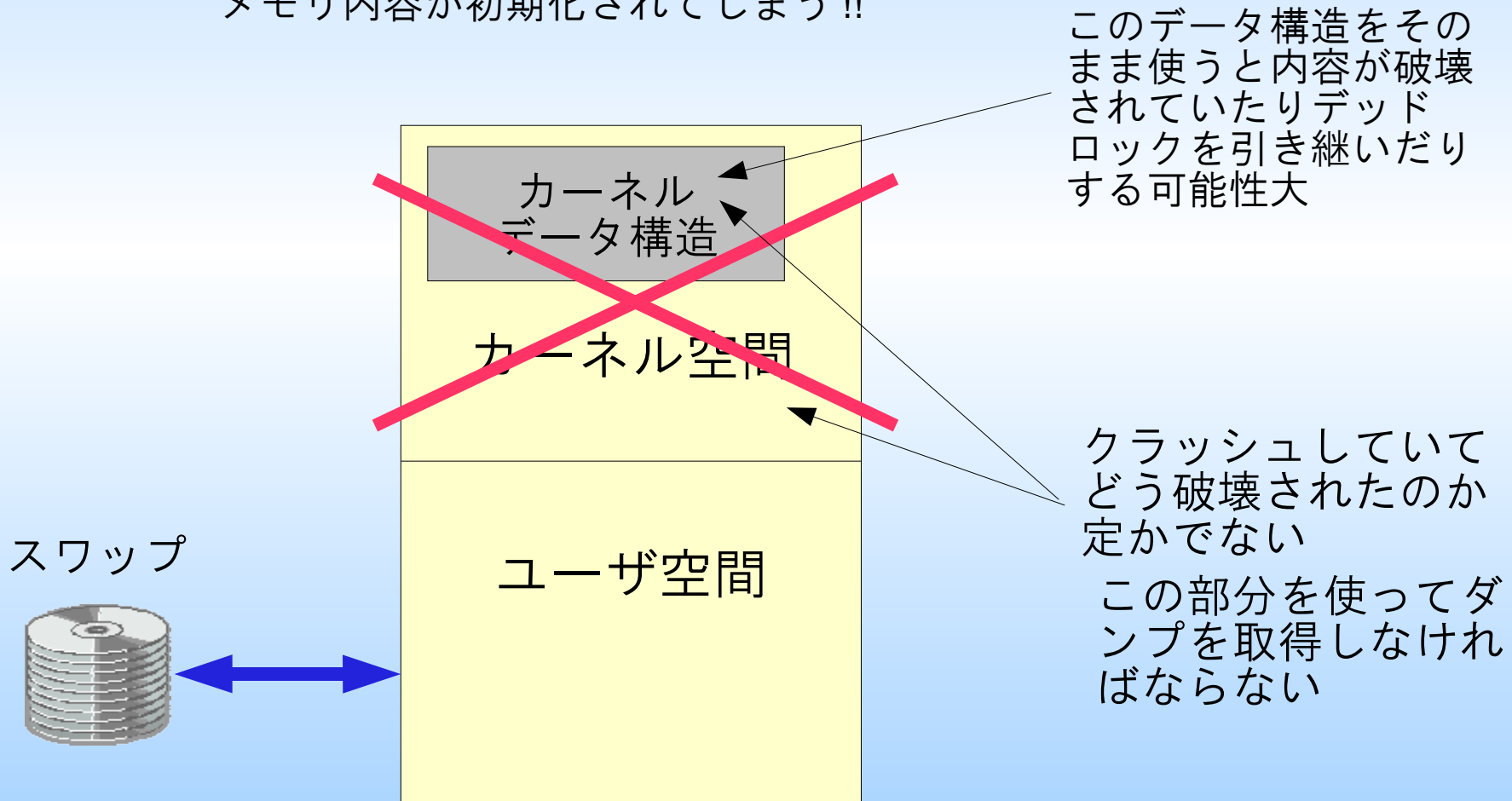
- 割り込みを用いず、ポーリングモードで書き込みを行うことで、ダンプ取得の確率が高まる

#### 短所

- SCSI の最下位レイヤのドライバの修正が必要となる
- これらをコンパイルしてカーネルを再構築する必要がある

# カーネル 2.6 での新しいカーネルダンプ取得技法 (1)

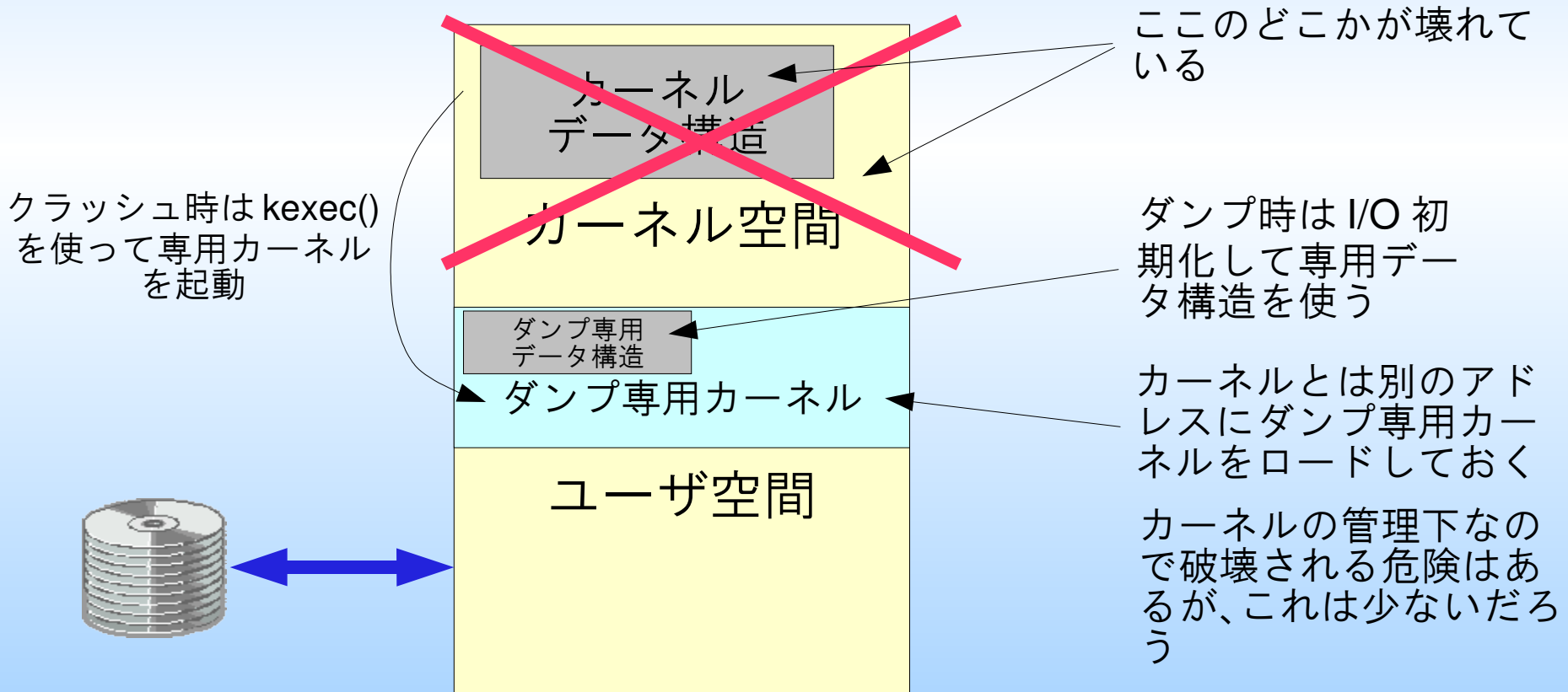
- クラッシュしたカーネルを用いたダンプ取得には確度、一貫性の限界がある。
  - クラッシュしたカーネルを使わずにダンプを取る
  - リブートして新しいカーネルを走らせようとする、BIOS でダンプすべきメモリ内容が初期化されてしまう!!



カーネルがクラッシュしたということは ...

# カーネル 2.6 での新しいカーネルダンプ取得技法 (2)

- カーネル 2.6 の新たな機能
  - Linux カーネルから別のカーネルをブートする内部関数の実装
    - kexec()
  - これを使えばクラッシュしたカーネルから別のカーネルを使ってダンプができる



# 新しいカーネルダンプの実装例

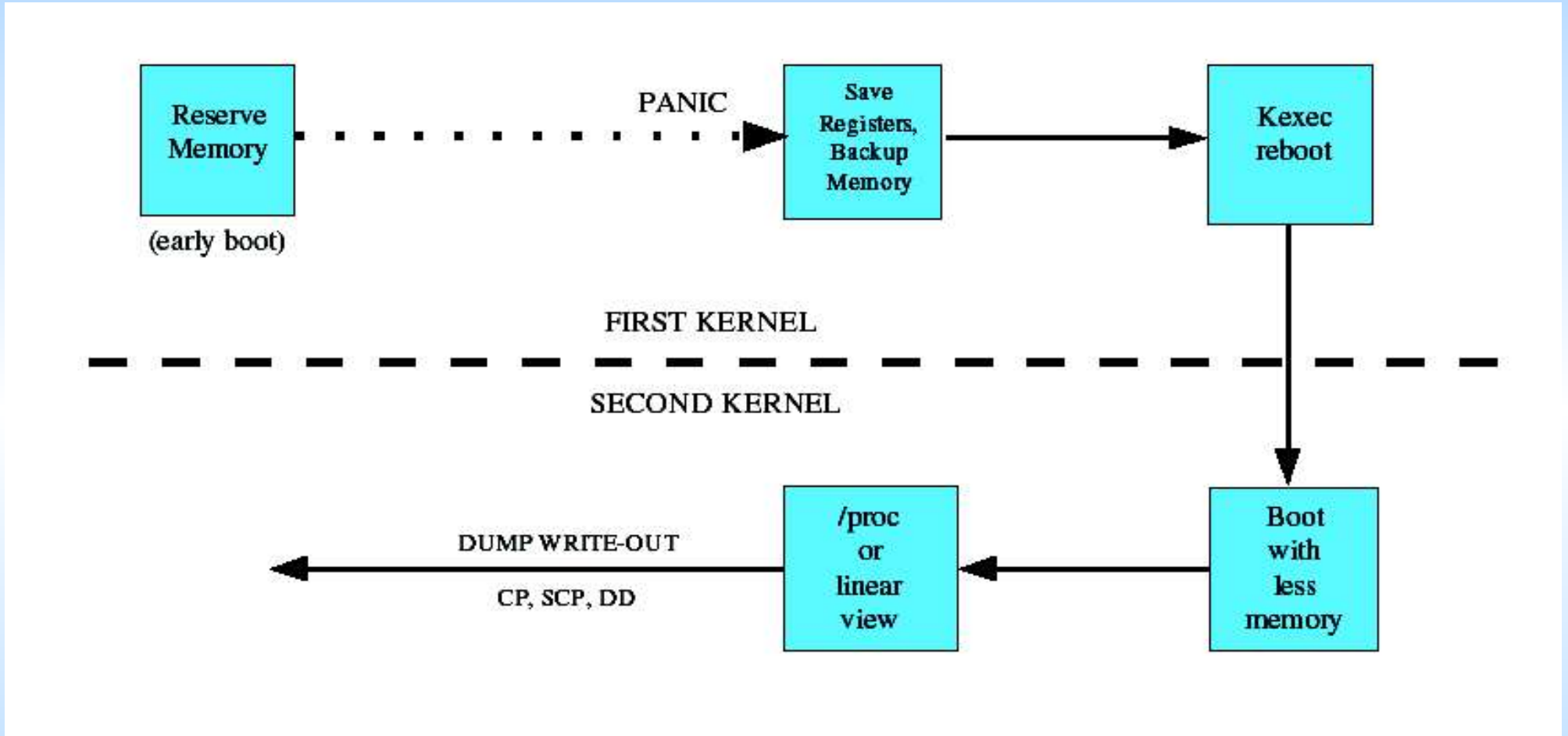
## IBM India Software Labs

Hariprasad Nellitheertha,  
Linux Technology Center, ISL

- kexec() を介してダンプ専用カーネルを起動
- ダンプしたデータはローカルに /dev/oldmem デバイスに蓄積
  - /dev/mem 同様にメモリーイメージでアクセス可能
  - /dev/oldmem デバイスはあらかじめ作成しておく

出典： The kexec Way to Lightweight Reliable System Crash Dumping,  
Hariprasad Nellitheertha, Linux Technology Center, ISL, IBM

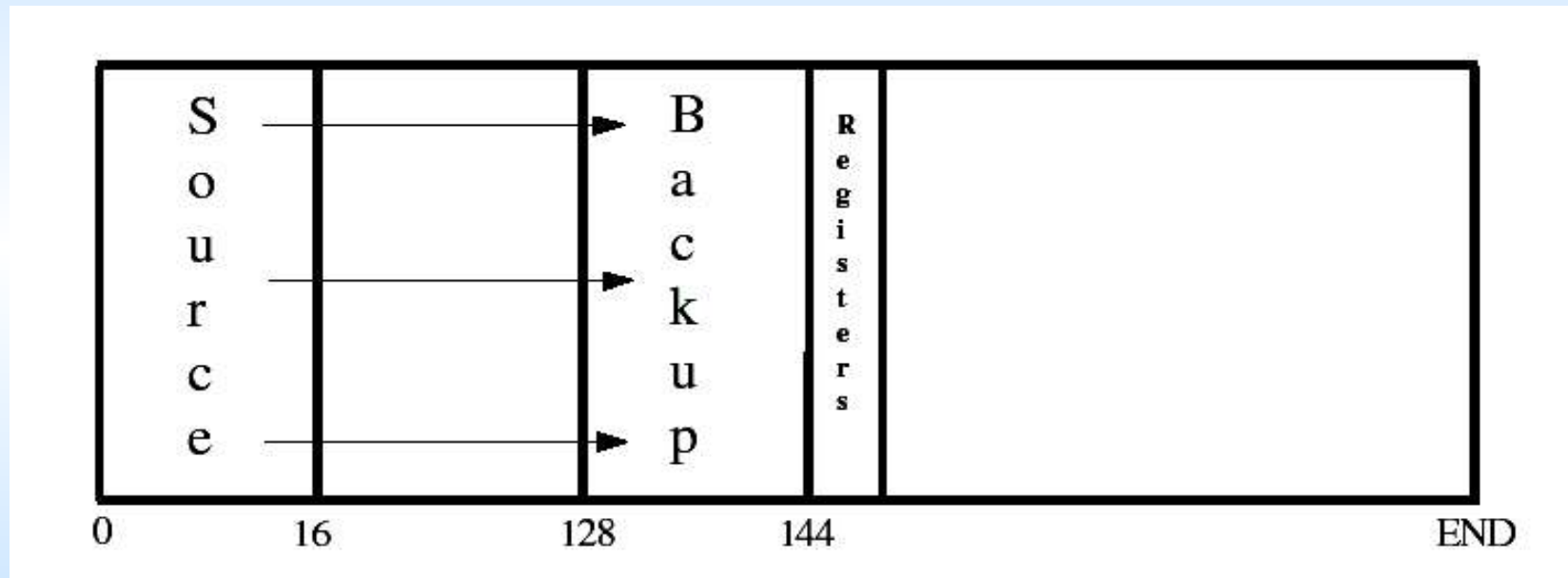
# ダンプ取得のプロセス



出典：The kexec Way to Lightweight Reliable System Crash Dumping, Hariprasad Nellitheertha, Linux Technology Center, ISL, IBM

# kexec() 利用上の工夫

- kexec() は、古いカーネルを上書きしてしまう
- 上書きされる前にこの部分のメモリーイメージを退避してからダンプ



出典：The kexec Way to Lightweight Reliable System Crash Dumping, Hariprasad Nellitheertha, Linux Technology Center, ISL, IBM



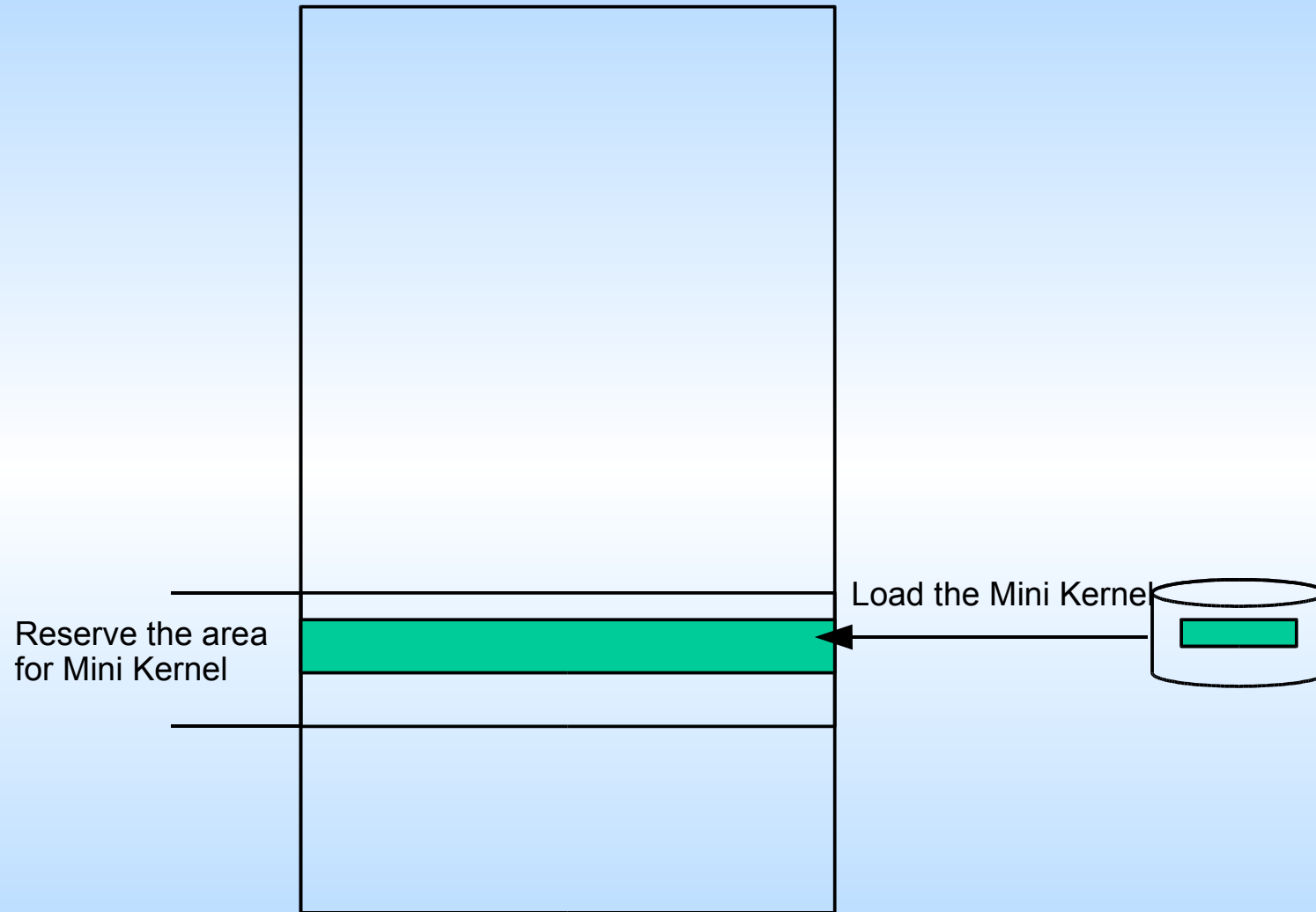
# kexec() を使ったもう一つのダンプ取得技法

## Linaccident ダンプツール

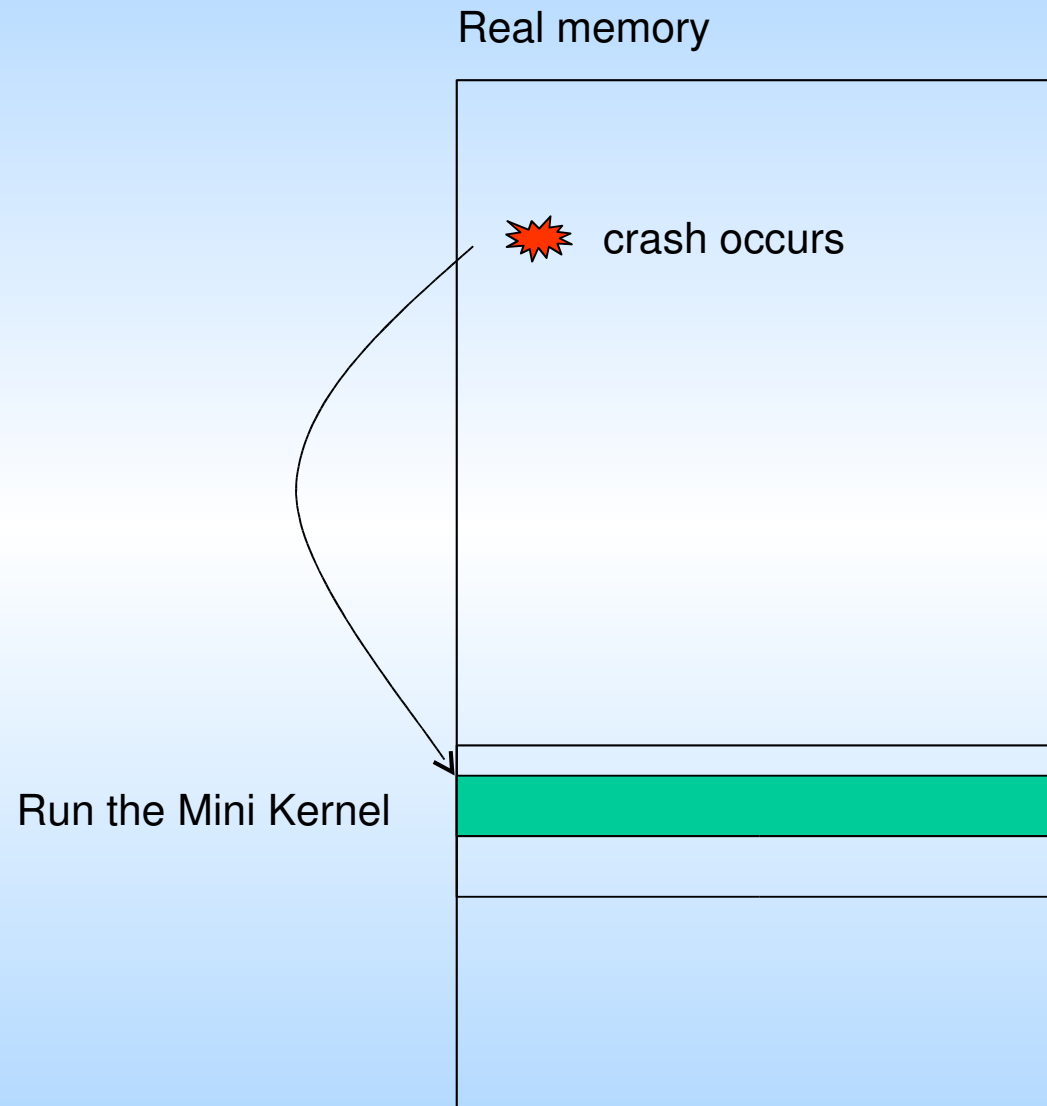
### NTT DATA & VA Linux Japan

- kexec() では古いカーネルイメージが上書きされる
  - 任意の場所にダンプミニカーネルをロードして起動できるように kexec() を変更 → mkexec()
- ダンプイメージは事前に設定したディスクパーティションに書き出す
- クラッシュしたカーネルのコードもデータ構造も用いない
  - 安全なミニカーネル内のコードとデータ構造で実行
  - ダンプ採取の確率向上
- ミニカーネルは、以下のコードをベースに開発
  - Linux カーネル
  - LKCD
  - kexec
- ミニカーネルにはダンプ処理と安全かつ必要最小限のドライバで構成
  - ドライバは Linux カーネルのものを使用
  - ミニカーネル内にロードされている
- カーネルダンプは lcrash(LKCD の解析ツール) を用いて解析

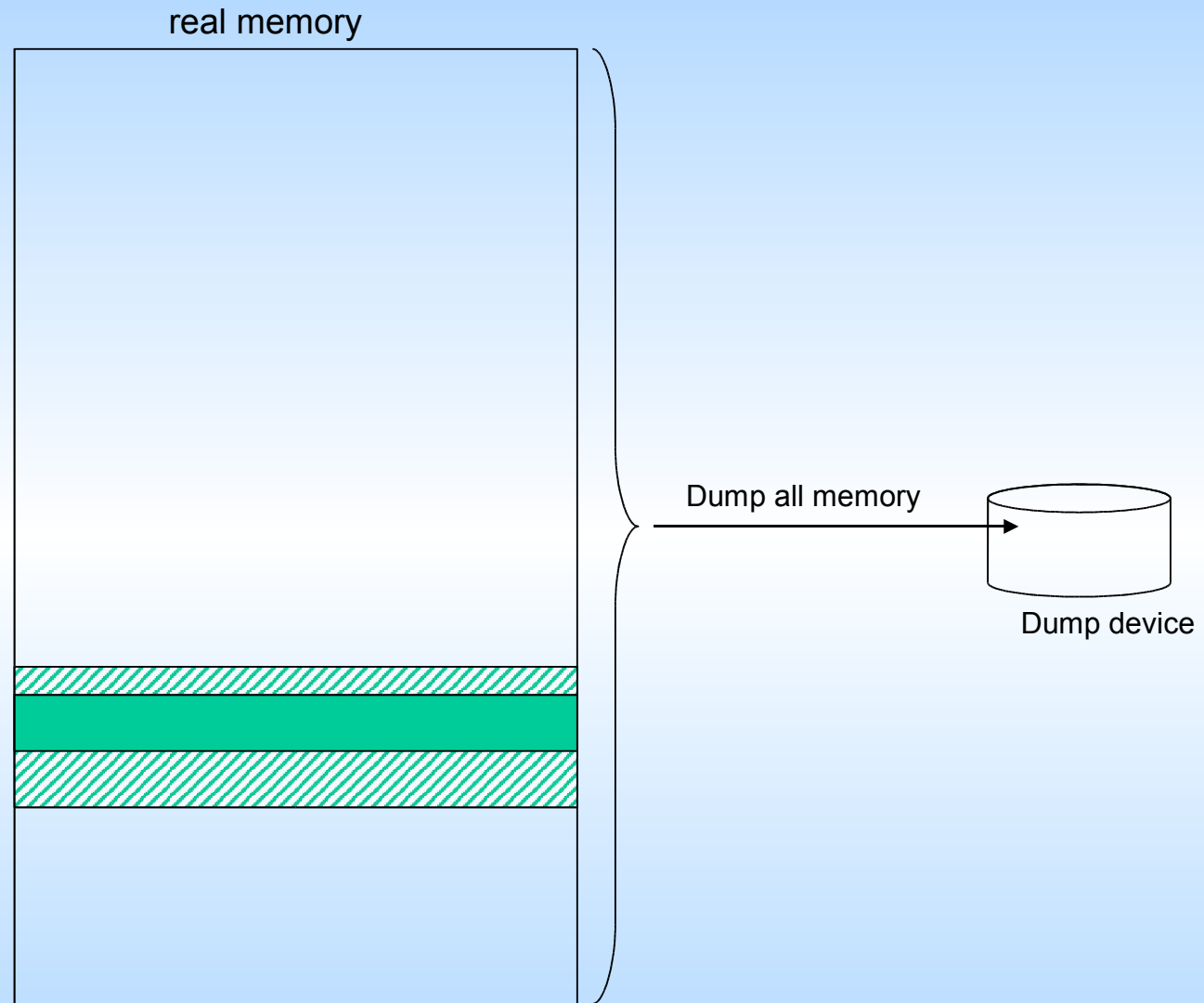
# ダンプの取得 (1)



# ダンプの取得 (2)



# ダンプの取得 (3)



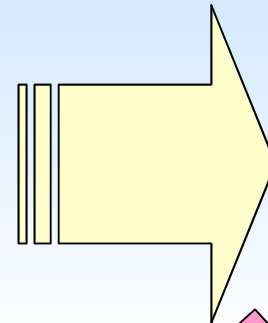
# ダンプ手法の比較

		LKCD(disk)	Netdump / LKCD(net)	mcore / LKCD(mem)	diskdump	Linaccident
採取できる情報	実行コンテキスト	○	○	○	○	○
	メモリ	○	○	○	○	○
	キャッシュメモリ	×	×	×	×	×
	スワップ領域	×	×	×	×	○
確実性	資源のロック	×	△	○	△	○
	メモリ破壊時	△	△	△	△	○
採取の契機	panic / oops	○	○	○	○	○
	nmi_watchdog	○	○	○	○	○ + α
	手動	○	○	○	○	○
	nmi スイッチ	×	×	×	×	×
ライブダンプの可否	○	○	×	○	×	× (検討項目)

# スワップと組み合わせたユーザ空間の再現 と障害解析

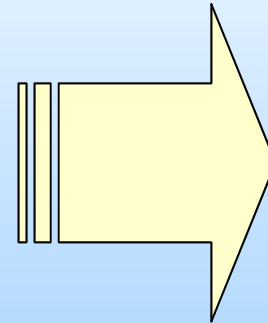
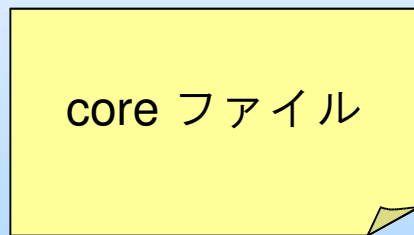
## 各種解析資料

- ダンプファイル
- カーネル構造情報 (Kerntypes)
- カーネルイメージ
- スワップ領域ダンプファイル
- System.map (シンボルマップ)
- .config ファイル
- カーネルソース
- 共有ライブラリのバージョン



lcrash によるカーネル解析

core ファイルの作成



gdb によるアプリケーション解析

# lcrash によるカーネル解析

```

kikuchi@dhcp154:/root
ファイル(F) 編集(E) 表示(V) ターミナル(T) 進む(G) ヘル
>>>
>>>
>>>
>>>
>>>
>> symbol per_cpu_runqueues
  ADDR  OFFSET SECTION  NAME  TYPE
-----
0xc0428a00  0 GLOBAL_DATA  per_cpu_runqueues  (unknown)
-----
1 symbol found
>>>
>>>
>>>
>>>
>> symbol __per_cpu_offset
  ADDR  OFFSET SECTION  NAME  TYPE
-----
0xc042b020  0 GLOBAL_DATA  __per_cpu_offset  (unknown)
-----
1 symbol found
>>>
>>>
>>>
>>>
>> dump 0xc042b020
0xc042b020: 011e1f60 : '...'
>>>
>>>
>>>
>> print 0xc0428a00 + 011e1f60
3244337504
>>>
>>>
>>>
>> base 3244337504
-----
hex: 0xc160a960
decimal: 3244337504
octal: 030130124540
binary: 0b11000001011000001010100101100000
-----
>> []

kikuchi@dhcp154:/root
ファイル(F) 編集(E) 表示(V) ターミナル(T) 進む(G) ヘルプ(H)
>> print *(struct runqueue*)0xc160a960 | more
struct runqueue {
  lock = spinlock_t {
    lock = 1
  }
  nr_running = 1
  cpu_load = 64
  nr_switches = 4272879
  expired_timestamp = 0
  nr_uninterruptible = 4294967228
  timestamp_last_tick = 159483440345561
  curr = 0xed2b87e0
  idle = 0xc037f000
  prev_mm = 0x0
  active = 0xc160a998
  expired = 0xc160ae10
  arrays = {
    [0] struct prio_array {
      nr_active = 1
      bitmap = {
        [0] 0
        [1] 0
        [2] 0
        [3] 2097152
        [4] 4096
      }
      queue = {
        [0] struct list_head {
          next = 0xc160a9b0
          prev = 0xc160a9b0
        }
        [1] struct list_head {
          next = 0xc160a9b8
          prev = 0xc160a9b8
        }
        [2] struct list_head {
          next = 0xc160a9c0
          prev = 0xc160a9c0
        }
        [3] struct list_head {
          next = 0xc160a9c8
          prev = 0xc160a9c8
        }
        [4] struct list_head {
          next = 0xc160a9d0
          prev = 0xc160a9d0
        }
      }
    }
  }
}

```

# gdb によるアプリケーション解析

```

kikuchi@dhcp154:/mnt/eos-fs/share/fas
ファイル(F) 編集(E) 表示(V) ターミナル(T) 進む(G) ヘルプ(H)
dhcp193:/usr/local/lkcd/lkcdutils-core #
dhcp193:/usr/local/lkcd/lkcdutils-core # ls
-
core.9328_lcrash.9328      libklib
core_bash.22322          librl
core_file_mmap.25928     libsial
.config                  core_httpd2-prefork.3808  libutil
GPATH                   core_vi.22330            lkcd_config
GRTAGS                  core_vi.22332            lkcd_ksyms
GSYMS                   core_vi.22333            log_stderr.txt
GTAGS                   core_vi.22915            log_stderr2.txt
Makefile                core_vi.22967            log_stderr3.txt
README                  core_vi.24943            log_stderr4.txt
README.lkcd             core_vi.28438            log_stdout.txt
README.netdump          core_vi.28689            log_stdout2.txt
README.ppc64            core_vi.bcup             log_stdout3.txt
README.sial              crash                    log_stdout4.txt
README.xlcrash          docs                     netdump
Rules.make              get_bfd_version.c       qlcrash
configure                lcrash                  scripts
configure.backup        lcrashd                  spec
configure.orig          liballoc

dhcp193:/usr/local/lkcd/lkcdutils-core #
dhcp193:/usr/local/lkcd/lkcdutils-core #
dhcp193:/usr/local/lkcd/lkcdutils-core #
dhcp193:/usr/local/lkcd/lkcdutils-core # gdb -c core_httpd2-prefork
.3808 /usr/sbin/httpd2-prefork

kikuchi@dhcp154:/root
ファイル(F) 編集(E) 表示(V) ターミナル(T) 進む(G) ヘルプ(H)
'/usr/local/lkcd/lkcdutils-core/core_httpd2-prefork.3808',
file type elf32-i386.
0x08048000 - 0x08098000 is load1
0x08098000 - 0x0809b000 is load2
0x0809b000 - 0x0818a000 is load3
0x40000000 - 0x40016000 is load4
0x40016000 - 0x40017000 is load5
0x40017000 - 0x40018000 is load6
0x40018000 - 0x4001a000 is load7
0x4001a000 - 0x4001b000 is load8
0x4001b000 - 0x4001d000 is load9
0x4001d000 - 0x4001e000 is load10
0x4001e000 - 0x40020000 is load11
0x40020000 - 0x40021000 is load12
0x40021000 - 0x40023000 is load13
0x40023000 - 0x40024000 is load14
0x40024000 - 0x40026000 is load15
0x40026000 - 0x40027000 is load16
0x40027000 - 0x4002e000 is load17
0x4002e000 - 0x4002f000 is load18
0x40031000 - 0x40041000 is load19
0x40041000 - 0x40042000 is load20
---Type <return> to continue, or q <return> to quit---

kikuchi@dhcp154:/root
ファイル(F) 編集(E) 表示(V) ターミナル(T) 進む(G) ヘルプ(H)
Loaded symbols for /lib/libnss_compat.so.2
Reading symbols from /lib/libnss_nis.so.2...(no debugging symbols found)
...done.
Loaded symbols for /lib/libnss_nis.so.2
#0 0xffffe410 in ?? ()
(gdb)
(gdb) bt
#0 0xffffe410 in ?? ()
#1 0xbffff948 in ?? ()
#2 0x40171588 in __JCR_LIST__ () from /usr/lib/libapr-0.so.0
#3 0xbffff920 in ?? ()
#4 0x401ee111 in accept () from /lib/tls/libpthread.so.0
#5 0x40165fbb in apr_socket_accept () from /usr/lib/libapr-0.so.0
#6 0x4016610b in apr_accept () from /usr/lib/libapr-0.so.0
#7 0x080878d8 in unixd_accept ()
#8 0x08067d43 in child_main ()
#9 0x080680fc in make_child ()
#10 0x080681c6 in startup_children ()
#11 0x08068a4b in ap_mpm_run ()
#12 0x0806f627 in main ()
(gdb)

```

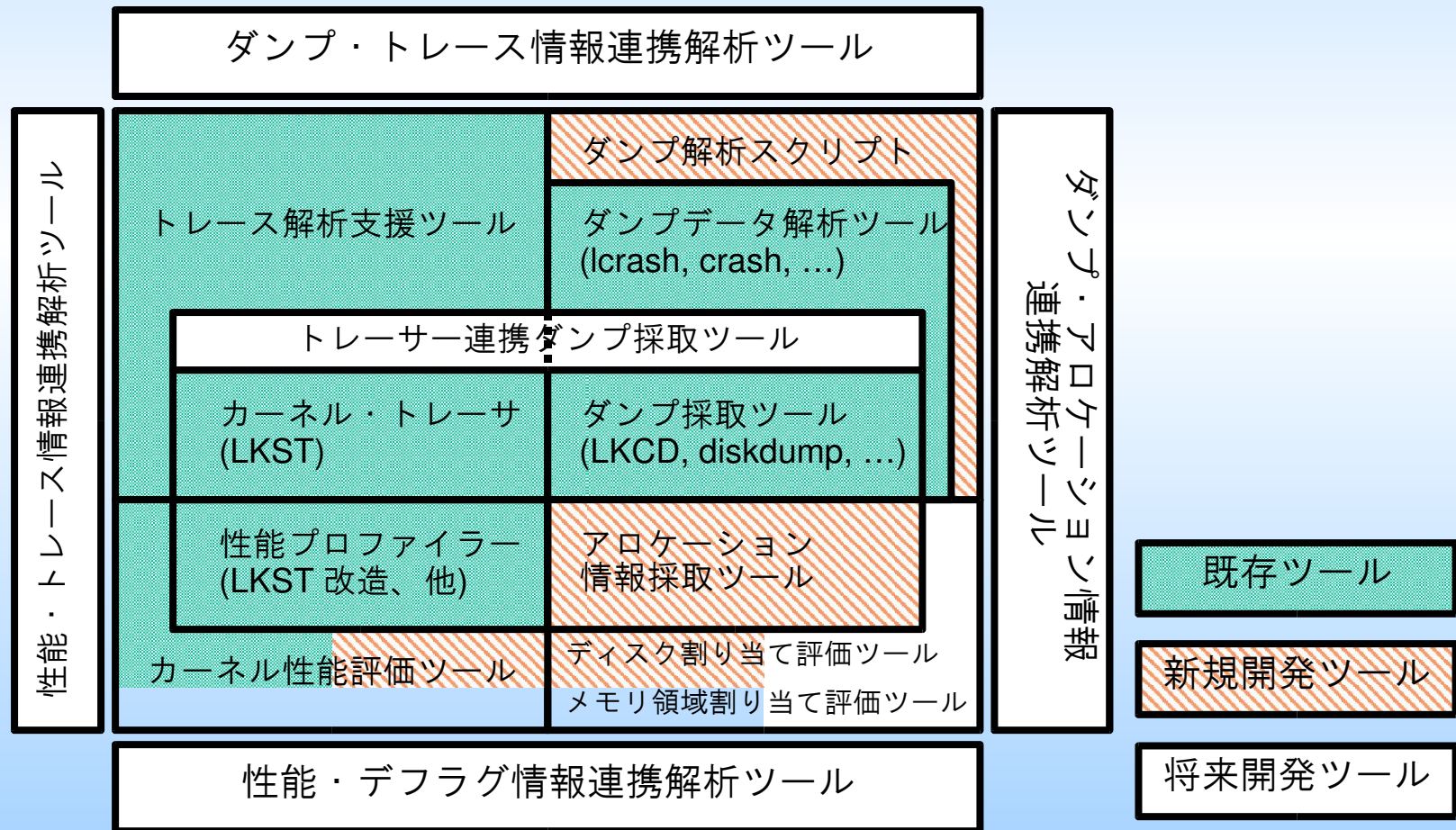


# 日本 OSS 推進フォーラム開発基盤 WG での検討状況

## 高信頼化ツール開発プラン

- ミッションクリティカル対応には RAS の強化が必須 (ダウンタイムの最小化、障害の早期検出)
- 高信頼化に必要なツール群を「トレースツール、ダンプツール、性能評価ツール、アロケーション情報評価ツール」に分類
- 不足している領域を新規に整備 (ダンプ解析スクリプト、アロケーション情報評価)

### ミッションクリティカル対応高信頼化ツール群



出典：OSS セプテンバーセミナー in 札幌

# 今後の期待

- 障害解析ツールの開発の促進
  - OSS 推進フォーラムなど
- カーネルダンプのメインツリーへの反映
  - 2004 年のカーネルサミットでも話題に
- カーネルダンプ情報の安全な運用方法
  - 暗号化を施した転送、蓄積

# Thank you very much

koichi@intellilink.co.jp