


Linux Kernel State Tracer

Systems Development Lab.
Central Research Lab.
Hitachi, Ltd.

LKSTとは

カーネル内の状態遷移をトレースする
イベントトレーサ

開発の背景

- スタックトレースのみでは障害情報収集は不十分
 - ◆ 一連の処理の流れが追えないケースがある
 - ◆ 関数の呼び出し関係より、更新されるデータの方が重要
 - ユーザサイトで起こった障害の情報も採取したい
 - ◆ 障害がうまく再現しないケースでも早期対応要
- 
- 十分な障害情報の提供が必要
 - いつでも障害情報が得られることが必要

設計方針

十分な情報を提供するために:

- 重要な関数 , 引数 , 変数を記録する
- トレース箇所ですべてのデータを採集

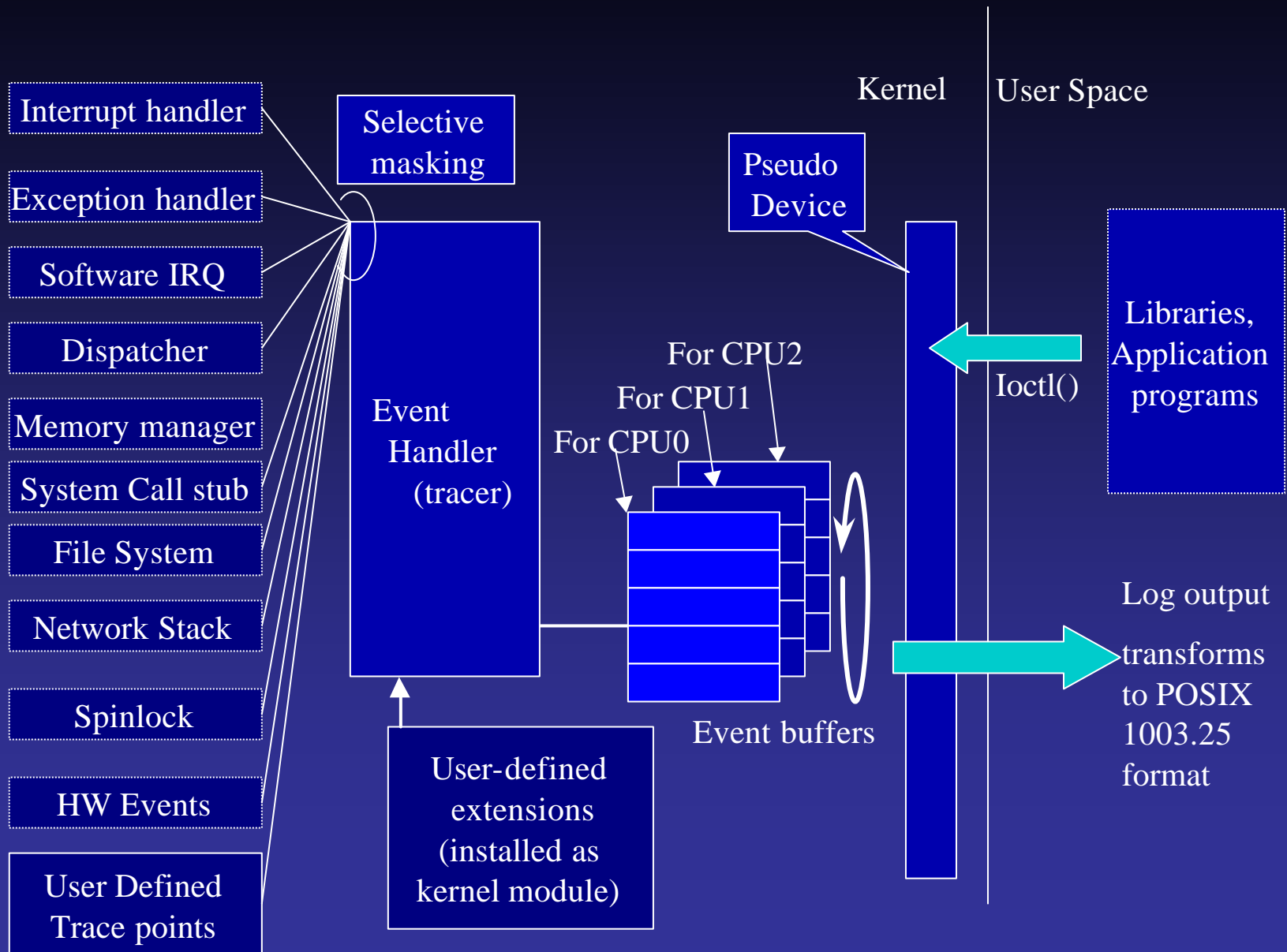
いつでも情報が入手できるように:

- OSがクラッシュまたは無限ループに入っても , 記録された情報は回収可能とする
- トレース記録のオーバーヘッドは最小限にとどめる

実際に役に立つ情報を採取するためには
さらに柔軟な機能拡張への対応が必要

Basic Structure

4



特徴

- システム性能への影響を抑えつつ柔軟な拡張性を提供
 1. 任意の記録関数の登録
 2. 記録関数の動的な切替
 3. 動的なバッファの切替

特徴 1

■ 任意の記録関数の登録

機能:

- ◆ カーネルモジュールを作ることによって任意の関数を登録可能 .
- ◆ 呼び出す関数はトレース箇所ごとに独立に設定可能

適用例:

- ◆ トレース記録の方式を変える

ex) 呼び出し回数だけのカウント, 経過時間の計測

- ◆ トレース記録の条件を詳細に設定可能

ex) プロセスごと, パケット種別ごと, タイマ構造体ごと

- ◆ 条件に応じてカーネルの動作を変える

ex) ある条件でのパケットを捨てる, タイマの期限を変更する

特徴 2

■ 記録関数の動的な切替

機能:

- ◆ 呼び出す関数とトレース箇所との対応付けを一括変更
- ◆ 対応付けの変更は記録関数内で実行可能 (カーネル動作への影響を考慮して設計)

適用例:

- ◆ トレース範囲の動的な変更

ex) 普段は数の多いイベントをマスクしておき、障害の前兆となるトレース箇所を通過したときそれらの記録を有効化する。

特徴 3

■ 動的なバッファの切替

機能:

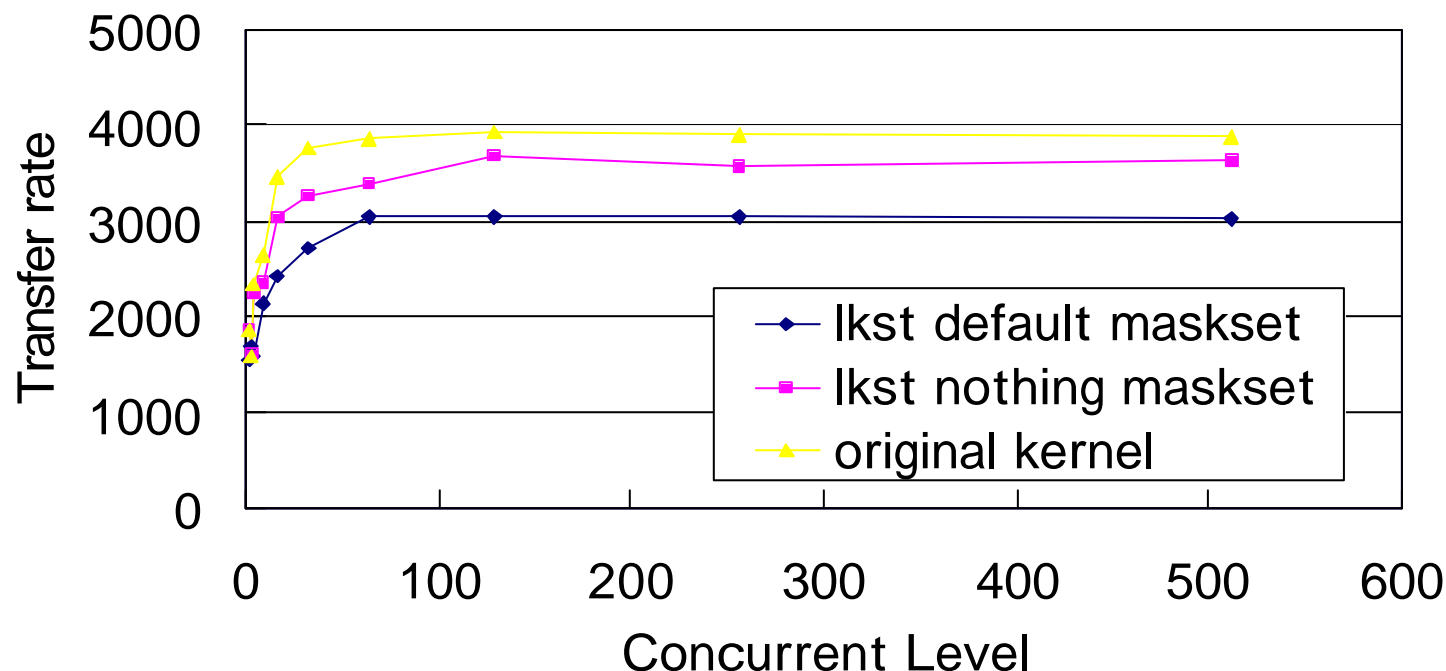
- ◆ バッファの変更は記録関数内で実行可能(カーネル動作への影響を考慮して設計)

適用例:

- ◆ イベント記録の優先順位付け
ex) 重要なイベントは別バッファに記録することでオーバーライトを回避 .

性能 1 ApacheBench

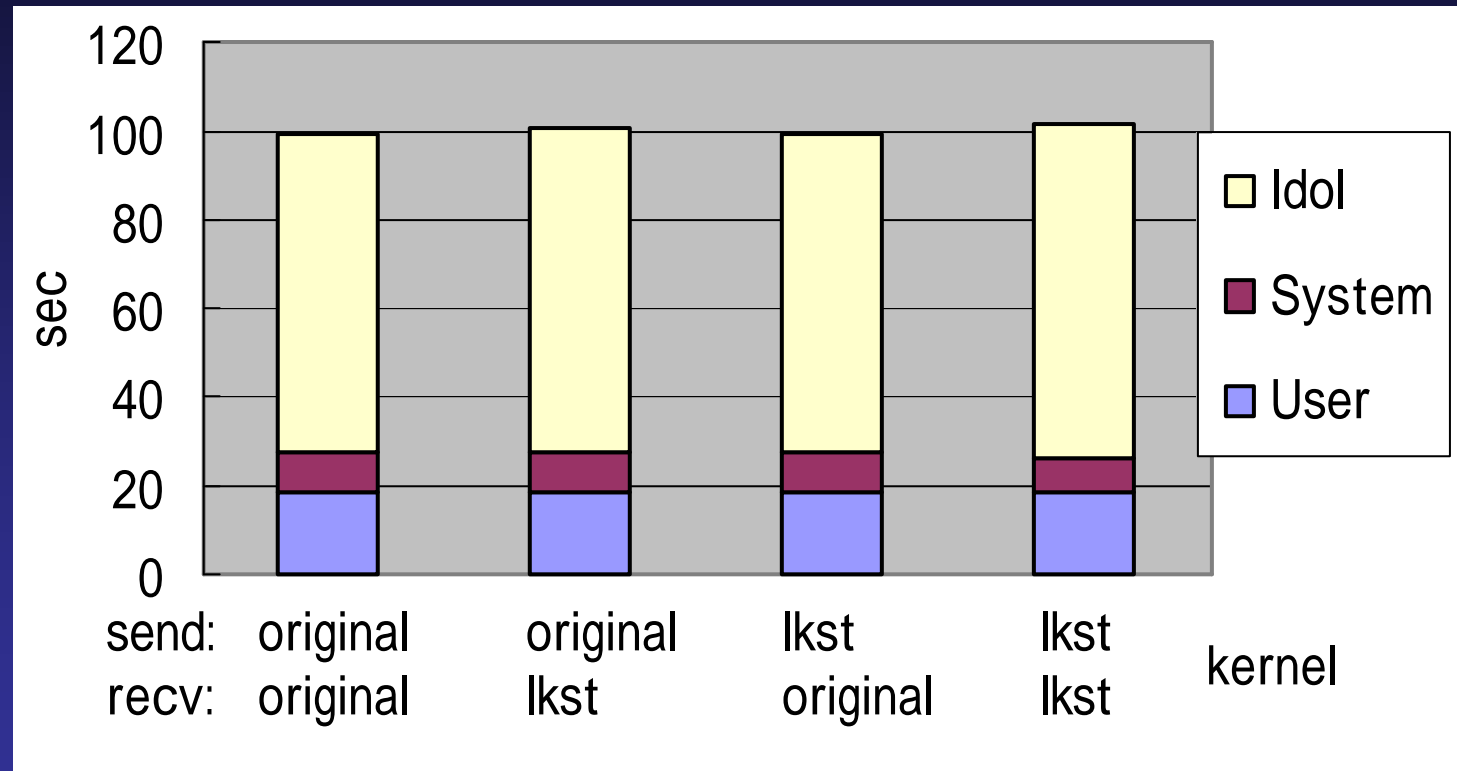
- ApacheBench(ab)を使用して,3KBのファイルを20秒間Getし続けた際のスループットを測定。



マスク時の性能改善方式を実装中
記録時処理は再検討中

性能 2 smtp

- Send側からReceive側へ1MBのメールを100通送信し、経過時間を測定



オーバーヘッドは数%程度

デバッグでの適用事例

問題:

- ◆ カーネル内のあるフラグを立てると、不正メモリアクセスでカーネルがダウン
- ◆ ダウンする箇所はタイマハンドラであり、原因となっている処理は別のコンテキスト(プログラム)

スタックトレースでは、障害の原因となる処理までは追跡不可

調査方法:

- ◆ タイマ操作関数の全て、および、上記タイマを含むデータAの操作関数全てにトレース箇所を設定
データAを消去してから、これに含まれるタイマが expire し、不正なメモリがハンドラとして参照しているのをトレース上で確認

原因:

- ◆ フラグの定数の間違い . 指定したフラグの値は、データAが既に無効化されている、という意味に解釈されていた
そのため上記のデータAが解放されてしまい、その領域が再利用されていた . 再利用の後タイマが expire していた

開発状況

ベータ2版をリリース (2002.1.17)

■ SourceForgeへ登録

- ◆ <https://sourceforge.net/projects/lkst/>
- ◆ 随時 CVS へcommit
- ◆ メーリングリスト(lkst-users, lkst-develop)

■ 日立Open Sourceのページでも公開

- ◆ <http://oss.hitachi.co.jp/>

■ 本年 3月のVersion 1.0リリースを目標に開発中

カーネルハッカー向けのご説明

トレースの入れ方

- ・ソースに下記を追加してリコンパイル

```
#include <linux/lkst.h>
...
lkst_entry_log(event_type, arg1, arg2, arg3, arg4);
...
```

event_type:

イベントの識別子

0x100 - 0x1ff あたりの任意の整数

arg1, arg2, arg3, arg4:

ログに記録される各 8byte の任意のデータ

障害時のトレース出力

方法 1. panic() にあるフックを利用する

- ◆ panic() イベントにバッファ出力のハンドラを割り当て
- ◆ 上記処理を行うカーネルモジュールを用意

デモ

方法 2. SysRQキー

デモ

方法 3. LKCD

デモ

トレースポイントでのスタックトレース

目的:

スピンロック, リファレンスカウンタなどのバグ調査:

- ◆ トレースポイントが一番下のオペレーションに入れるのが簡単
- ◆ 処理の流れを追うためには, どこから呼ばれているかが知りたい

例)

追いたい流れ

```
register_netdevice(dev);    - - -  
...                        - - -  
dev_queue_xmit(skb);       - - -  
...                        - - -  
netif_rx(skb);             - - -  
...                        - - -  
unregister_netdevice(dev);
```

トレースポイント

```
dev_hold(dev);  
dev_put(dev);  
dev_hold(dev);  
dev_put(dev);  
dev_hold(dev);  
dev_put(dev);
```


マスクセット(1)

・マスクセットの切り替え

```
% lkstm set -m 切り替え先マスクセットHD番号
```

・登録されているマスクセットのリスト表示

```
% lkstm list
```

・マスクセットの内容の表示

```
% lkstm read -m 読み込みマスクセットHD番号
```

マスクセット(2)

・マスクセットの登録

```
% lkstm write -m マスクセットID -f ファイル名 -n マスクセット名
```

入力ファイルはlkstm readコマンドの出力形式と同様

・マスクセットの削除

```
% lkstm delete -m 削除マスクセットID番号
```

バッファ (1)

・バッファの切り替え

バッファはCPU毎のリスト構造

下記コマンドでリストの次のバッファへと切り替える

```
% lkstbuf shift -c バッファを切り替えるCPU番号
```

・登録されているバッファのリスト表示

```
% lkstbuf list
```

・登録されているバッファの内容を出力

```
% lkstbuf read -b バッファID -n リードエントリ数 -f 出力ファイル
```

バッファ (2)

・バッファの追加生成

指定したCPUのバッファリストにバッファを追加

```
% lkstbuf create -c CPU番号 -b バッファID -s サイズ (バイト)
```

・バッファの削除

```
% lkstbuf delete -b バッファID
```

イベントハンドラ (1)

LKSTソースファイル testtools/lkst_mod_panicdump.c 参照

```
int lkst_mod_panicdump_init()
{
    ...
    retval = lkst_evhandler_register(LKST_EVHANDLER_ID_DEFAULT+1,
                                    "lkst_evhandler_panicdump",
                                    &lkst_evhandler_panicdump,
                                    &lkst_evhandler_test_ctrl);
    ...
    if (body.entry[i].event_type == LKST_ETYPE_O_PANIC ) {
        body.entry[i].id = LKST_EVHANDLER_ID_DEFAULT+1;
    }
    ...
    if ((retval = lkst_maskset_write(&maskset_param)))
    ...

```

ハンドラ登録関数

ハンドラID

ハンドラ関数

マスクセットのハンドラ変更

マスクセット登録関数

イベントハンドラ (2)

- 登録されているマスクセットのリスト表示

```
% lksteh list
```

- イベントハンドラ制御関数の呼び出し

```
% lksteh ctrl -e イベントハンドラID -f ファイル名
```

デモ

おしながき

■ 基本機能

- ◆ バッファ出力(SysRQ-V, lkstbuf),
- ◆ mask 切替

■ スタックトレース付きイベントトレース

■ パニック時のダンプ

- ◆ panicdump モジュール
- ◆ lkcd