powerpc linux で oprofile
2006年1月17日　第57回カーネル読書会

上川　純一

debian

Debian Developer: システムテスト担当

debian

chrootを構築してアプリのテストをする際に
試験環境のデータをコピーするところが
ボトルネックだというのはわかっていた

debian

アプリケーションがこれから変更するファイルだけを
コピーで編集させてそれ以外のファイルは
最初はハードリンクにしておくアプローチを取ると
高速で処理ができるのではないか

ptraceでシステムコールをトラップしたら実装できるのではないか
LD_PRELOADでlibcをオーバライドすればよいのではないか

debian

あれは2005年末のクリスマス

debian

ptraceというシステムコールがある
なんとなく標準、使い込むと不便

debian

マニュアルを眺めていたらシステムコールをトラップして
アプリケーションの挙動を確認できる雰囲気
実際問題として
そういうアプリケーションは実装できるものか

debian

ptraceのマニュアルを見る

- long ptrace(enum __ptrace_request request, pid_t pid, void *addr, void *data);
- PTRACE_GETREGS, PTRACE_GETFPREGS
  - それぞれ、子プロセスの汎用レジスタ、浮動小数点レジスタを親プロセスの data の位置にコピーする。この data の書式に関しては <linux/user.h> を参照すること。(addr は無視される。)
- PTRACE_SETREGS, PTRACE_SETFPREGS
  - それぞれ、子プロセスの汎用レジスタ、浮動小数点レジスタに親プロセスの date の位置からコピーする。
    PTRACE_POKEUSER と同様に、汎用レジスタによっては変更が禁止されている場合がある。(addr は無視される。)
- 出典： JM

debian

ptraceの非互換

- PTRACE_GETREGS
  - 汎用レジスタの値を取得する
    - そもそもアーキテクチャによりシステムコールの発行方法は全然違うので、前提条件
  - sparc, ppc64(PPC64_PTRACE_GETREGS): data/addr逆
  - i386, ia64, m32r, m68k, x86-64: 正しい実装っぽい
  - ppc, alpha, s390, mips, parisc: 実装していない
- PTRACE_PEEKUSR/POKEUSR, PEEKUSER/POKEUSER
  - glibcのマニュアルとkernel :PTRACE_PEEKUSR
  - glibcのヘッダ： PTRACE_PEEKUSER

debian

ptraceの実装

- arch/ppc/kernel/ptrace.c に巨大なcase文がある
- そこでPTRACE_GETREGSを定義してあげる
- 一部はinclude/linux/ptrace.hにて定義されているが、PTRACE_GETREGSはアーキテクチャ毎にカーネルのヘッダファイルで定義しているものらしい
  - include/asm-ppc/ptrace.hに定義する必要がある
- ptraceされているときにはタスクスイッチ時にスタックにレジスタをすべて保存してくれるらしい
  - child->thread.regs[]をput_user

```
#define PTRACE_TRACEME          0
#define PTRACE_PEEKTEXT         1
#define PTRACE_PEEKDATA         2
#define PTRACE_PEEKUSR          3
#define PTRACE_POKETEXT         4
#define PTRACE_POKEDATA         5
#define PTRACE_POKEUSR          6
#define PTRACE_CONT             7
#define PTRACE_KILL             8
#define PTRACE_SINGLESTEP       9
```


あれは2005年末のクリスマス

debian

なぜか僕はiBookをひらいてカーネルをハックしていた

debian

iBookでoprofileが動かないじゃないか

(タイマー割り込みモードを除く)

debian

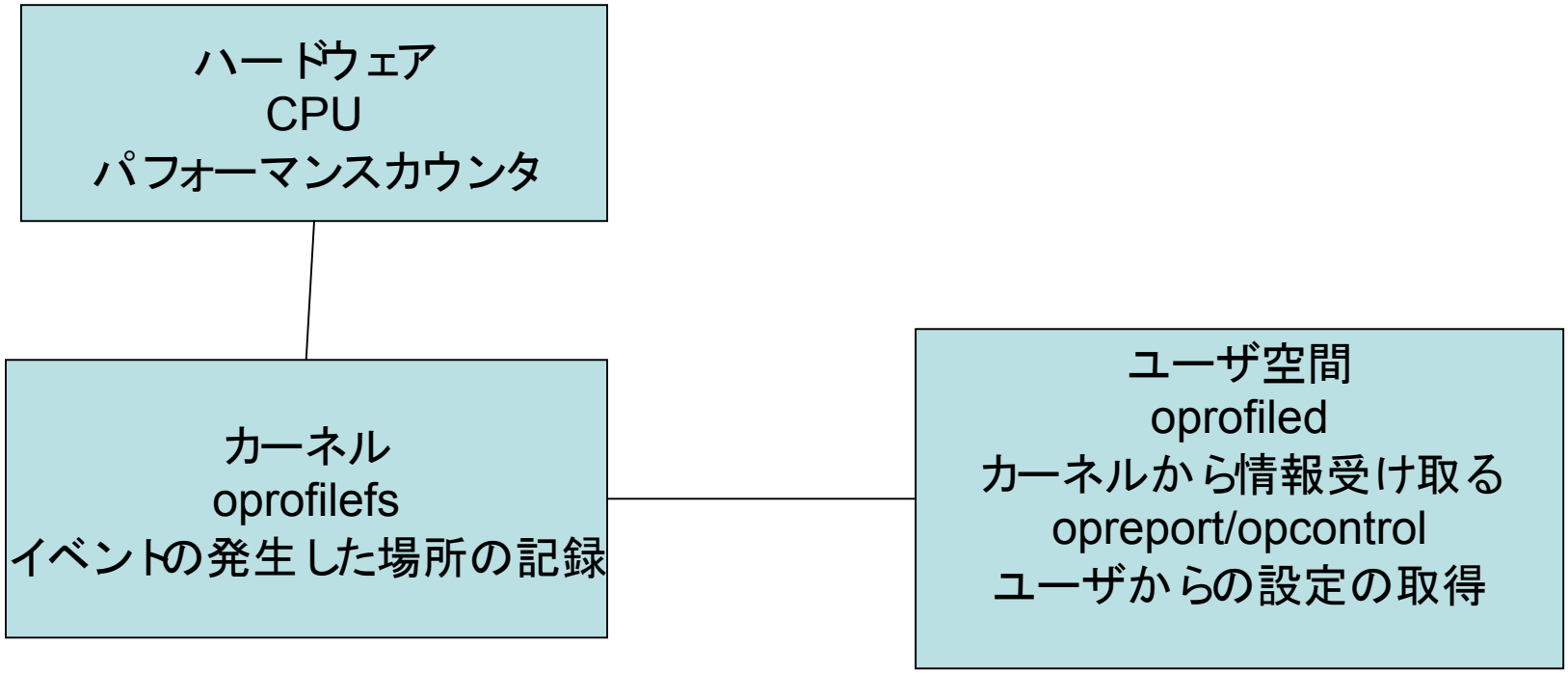
はじめに

- oprofileというプロファイルツールが安定版Linux Kernel に標準で入っている（2.6.14時点）
- iBook G4にはMPC7447AというPowerPCチップが乗っている。
  - oprofileは現時点のカーネル・ユーザランドとも対応していない
- oprofileでアプリのパフォーマンスを確認したいのだが、タイマー割り込みの挙動が不満
  - oprofileに7447A CPU対応を追加しよう

- Intel搭載のMacBook発表になりましたが、あえてiBook G4の話題を

debian

oprofileの構成



debian

oprofileでいじる部分

- Linux Kernel
  - arch/ppc/oprofile/
  - arch/ppc/kernel/perfmonXXX.c, head.S, traps.c
  - include/asm-ppc/reg.h
- oprofile　（ユーザ空間側）
  - libop/op_cpu_type.[ch]： CPUを定義してあげる
  - libop/op_events.c: 7447A用のデフォルトを設定
  - events/ppc/7447A/ :どんなイベントがあるのか

debian

powerpcの パフォーマンスカウンタサポート

- ppc向けのoprofileのサポート自体は存在している
- 32bit CPU
  - 604 (/proc/ppc_htabという独自規格)
  - BOOK_E
- 64bit CPUのサポート
  - power3
  - power4
  - 970 （PowerMac G5）
- power3,4などは複数のパフォーマンスカウンタがそれぞれ独立して設定できなく、グループが定義されている。そのため、実装がx86系アーキテクチャなどと大きく違っている
- パフォーマンスカウンタの観点からはMPC7447Aは、MPC7450系とでもいうべき流派のチップらしい

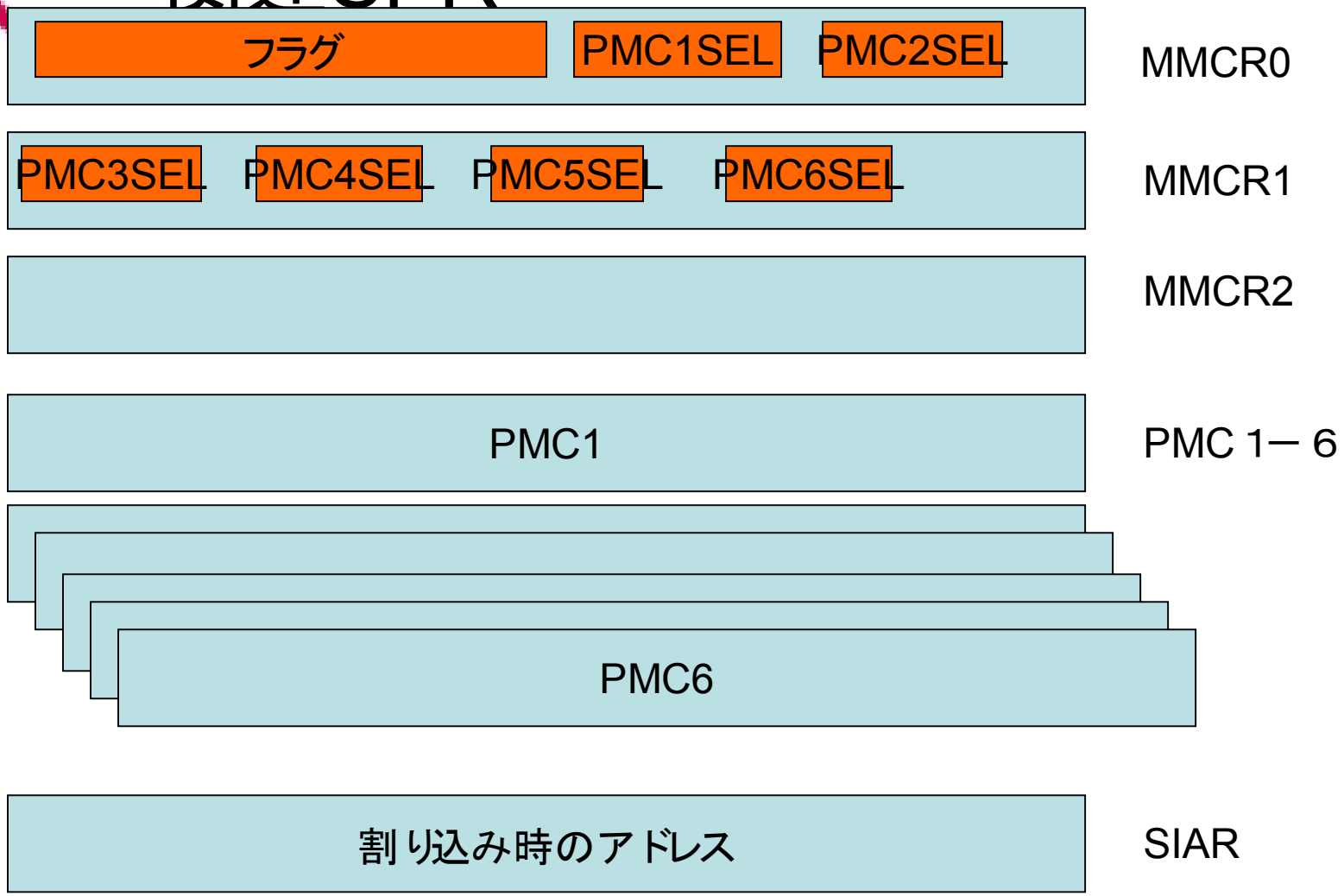
debian

powerpcのレジスタ構成 抜粋

- GPR0-31: 汎用レジスタ
- SPR : 制御用の特殊用途のレジスタ、mfspr命令などでアクセス
  - MMCR0,1,2
  - PMC1,2,3,4,5,6 など
  - 値＝mfspr(＃), mtspr(#, 値）
- MSR: マシンステートレジスタ
  - MSR_PMMビット:パフォーマンスモニタ動作
  - MSR_EEビット: 外部割り込み許可
  - MSR_PRビット: 特権モードかユーザモードか
  - 値＝mfmsr(), mtmsr(値）

debian

MPC7450のパフォーマンスカウンタ関連SPR



MPC7450で重要なフラグ抜粋

- MMCR0
  - FC: カウンタ停止フラグ
  - FCECE：イベントによってカウンタを停止
  - PMXE: パフォーマンス例外を有効化
  - PM1CE： PM1をenable
  - PMnCE: PM2-6　をenable
  - FCS:　特権モードのときにカウンタ停止
  - FCP:　ユーザモードのときにカウンタ停止
  - FCM1：マークMSR[PMM]=1のときにカウンタ停止
  - FCM0：マークMSR[PMM]=0のときにカウンタ停止
- MMCR0,1
  - PMC1-6が何をカウントするのかを指定
    - カウンタによって指定できる値の範囲と意味が違う

debian

実装：oprofileの操作の実装

- arch/ppc/oprofile/common.c (oprofile_arch_init)：
  model=&op_model_6xx として代入する。
  中身をarch/ppc/oprofile/op_model_6xx.cなどで実装
  - struct op_ppc32_model op_model_6xx = {
  - .reg_setup = model_6xx_reg_setup,
  - .start = model_6xx_start,
  - .stop = model_6xx_stop,
  - .handle_interrupt = model_6xx_handle_interrupt,
  - }
- .startが呼ばれたときにパフォーマンスカウンタを初期化してあげ、カウンタの条件をすべて満たすとPMCxが増加する
- PMCxの値が0x80000000に到達すると例外「0xf00 」が発生

debian

実装：パフォーマンスカウンタ

- arch/ppc/kernel/perfmon.c
  - パフォーマンスカウンタのリソース共有部分はここで制御していることになっている
  - oprofile/common.cから呼び出している
- arch/ppc/kernel/perfmon_XXXX.c
  - 実装する
- arch/ppc/kernel/head.S
  - 例外を処理する部分
- arch/ppc/kernel/traps.c
  - 例外処理の実装の実体がある

debian

何も考えずに レジスタだけ設定してみる

```
Bad trap at PC: c0032ac8, MSR: 200b032, vector=f00 Not tainted
Oops: Exception in kernel mode, sig: 5 [#1]
NIP: C0032AC8 LR: C0032A9C SP: C40FDE50 REGS: c40fdda0 TRAP: 0f00 Not tainted
MSR: 0200b032 EE: 1 PR: 0 FP: 1 ME: 1 IR/DR: 11
TASK = c409ac50[3733] 'opcontrol' THREAD: c40fc000
Last syscall: 120
GPR00: 00000000 C40FDE50 C409AC50 00000000 FFFCE08C 00000001 43B5FEB3 0(
GPR08: 000F423F 00000000 0005C490 00000727 10624DD3 100D9234 100D0000 100
GPR16: 00000000 7FF854F8 00000EDB C02D0000 00000000 C40FDF50 7FF851D0 30
GPR24: C35BA758 00000000 C02D13C0 C025C630 C35BA7D4 00000000 C40FDE78 C
NIP [c0032ac8] do_posix_clock_monotonic_gettime_parts+0x68/0x80
LR [c0032a9c] do_posix_clock_monotonic_gettime_parts+0x3c/0x80
Call trace:
[c0032b00] do_posix_clock_monotonic_get+0x20/0x70
[c0019dac] copy_process+0x32c/0x1040
[c001ab30] do_fork+0x70/0x220
[c0008288] sys_clone+0x68/0x90
[c0003da4] ret_from_syscall+0x0/0x44
```

どうやら何かは動いているようだ

debian

割り込みの処理

- 例外が発生するので、arch/ppc/kernel/head.Sを編集
- 0xf00を処理している部分を見つけて、UnknownExceptionをPerformanceMonitorExceptionに変更

```
. = 0xf00
b Trap_0f
. = 0xf20
b AltiVecUnavailable
Trap_0f:
EXCEPTION_PROLOG
13
addi r3,r1,STACK_FRAME_OVERHEAD
EXC_XFER_EE(0xf00, UnknownException)
```

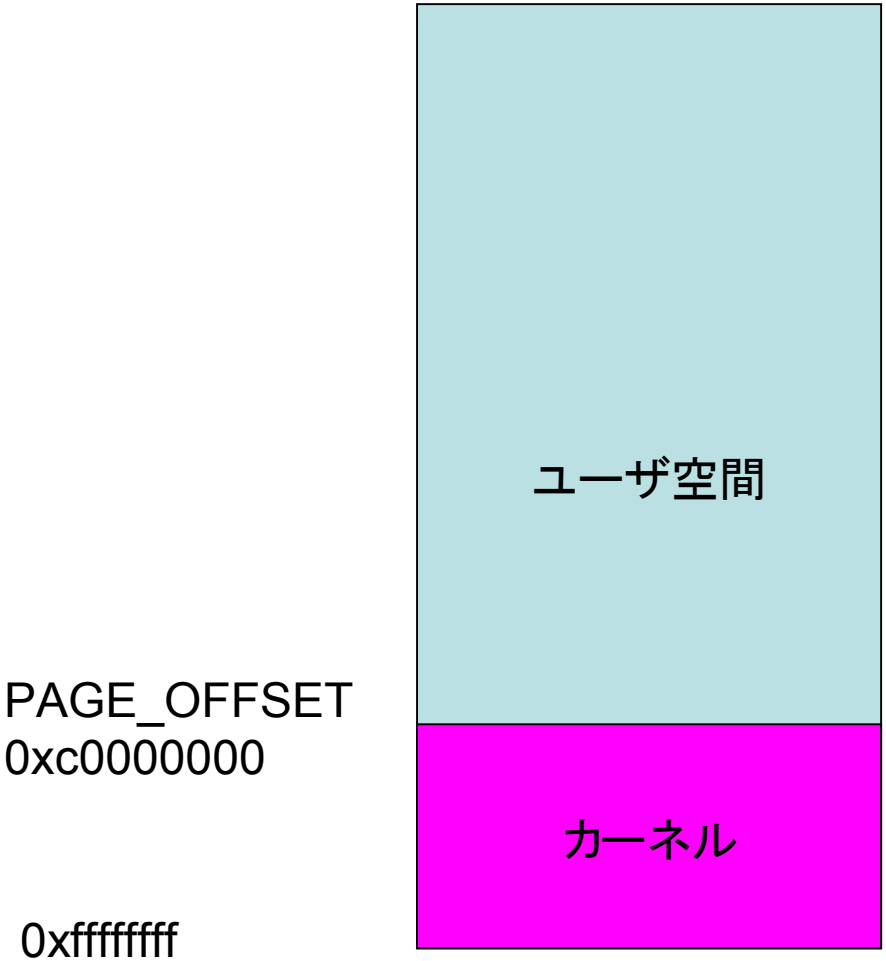

debian

実際の詳細の処理

- PerformanceMonitorExceptionでは model_6xx_handle_interruptを読んでもらい、そこで oprofile_add_pc(pc, is_kernel, i)　を呼び出す
  - 割り込み時のアドレスは、SIARというSPRに入っているのでその値をmfsprで取得し、pcに代入
  - is_kernelは仮想アドレスがPAGE_OFFSETとくらべてどちら側であるかを確認すればよい。
  - SRR0がr12に保存されているが、それは割り込みから帰るためのアドレスで、次の命令のアドレス
  - カウンタを（0x80000000-カウントしたい値）にリセットし、割り込みからもどる

debian

powerpc32 の仮想メモリ空間



debian

ユーザ空間側の対応

- 300種類近くあるため、イベントを全部定義するのは面倒
  イベント1と2番だけ登録してみました

```
diff -urNp -x Makefile.in -x aclocal.m4 -x configure -x oprofile.1 oprofile-0.9.1/events/ppc/7447A/ev--- oprofil
+++ oprofile-0.9.1-patched/events/ppc/7447A/events 2005-12-29 14:34:19.000000000 +0900
@@ -0,0 +1,4 @@
+# MPC7450: 7447A events
+# a hacked-up quickie.
+event:0x1 counters:0,1,2,3,4,5 um:zero minimum:3000 name:CYCLES : Counts every processor cycle
+event:0x2 counters:0,1,2,3,4,5 um:zero minimum:3000 name:COMPLETED_INSNS : Completed Instructi
oprofile-0.9.1/events/ppc/7447A/unit_masks 1970-01-01 09:00:00.000000000 +0900
+++ oprofile-0.9.1-patched/events/ppc/7447A/unit_masks 2005-12-29 14:34:44.000000000 +0900
@@ -0,0 +1,4 @@
+# 7450 possible unit masks; what's this for? is this functional?
+#
+name:zero type:mandatory default:0x0
+ 0x0 No unit mask
```


debian

ユーザ空間側の対応

- カーネルから出てくるCPUタイプの対応を追加
  - 面倒だったので自分のCPUに対応するように
    - 本当はoprofile_arch_init側で似ている実装のCPUをまとめて「モデル」を作ってあげるとよい

```
--- oprofile-0.9.1/libop/op_cpu_type.c 2005-07-12 05:46:23.000000000 +0900
+++ oprofile-0.9.1-patched/libop/op_cpu_type.c 2005-12-29 14:42:08.000000000 +0900
@@ -55,6 +55,7 @@ static struct cpu_descr const cpu_descrs
{ "NEC VR5432", "mips/vr5432", CPU_MIPS_VR5432, 2 },
{ "NEC VR5500", "mips/vr5500", CPU_MIPS_VR5500, 2 },
{ "e500", "ppc/e500", CPU_PPC_E500, 4 },
+ { "ppc 7447A", "ppc/7447A", CPU_PPC_7447A, 6 },
};
```


debian

oprofileで対応するのが面倒そうな機能

- PMC1がトリガーしたらPMC2-6をカウントアップし始める、という機能などがあるため、PMC1は他のカウンタと違う意味を持つ
- oprofileでそれを実現する方法としては、グループというのを定義してがんばる

debian

余談 :Altivec対応

- PowerPCでは、Altivec命令を利用した瞬間にAltivecUnavailable例外が発生
- Altivec用のレジスタをロードして、MSRにAltivec利用可能というフラグを立てる
- Altivec利用可能であるかを確認してswitch_toのときにレジスタを保存する

debian

余談 :cpu_features

- powerpcのLinuxカーネルは起動時にCPU検出して、持っていない機能に関しては、NOPで上書きしてくれるという仕様になっている
  - BEGIN_FTR_SECTION
  - END_FTR_SECTION(CPU_FTR_XXX)

debian

余談 :ppc、ppc64→powerpc

- 2.6.14までは、
  - arch/ppc
  - arch/ppc64
- 2.6.15から
  - arch/powerpc :基本こちらにマージ
    - #ifdefしただけ？
  - arch/ppc, arch/ppc64 :以前ののこりかす

debian

余談 :当初の目的　cowdancer パフォーマンスチューニング

- oprofileでパフォーマンスチューニングをした対象はcowdancer
  - fscanfは結構遅いのでバイナリデータをmmap
  - 検索にmemmemを利用すると500KB（L2キャッシュのサイズ）くらいを超えると遅いので、bsearchに変更
- 100回cowdancerを実行するのに必要な時間が1分30秒→30秒程度に削減
- あたりまえな考察
  - よくアクセスするデータ構造はキャッシュにおさまるサイズに
  - データ全体を読むより一部を読むほうがはやい

debian

チューニング前

samples % image name app name symbol name
121327 30.5375 libc-2.3.5.so bash _IO_vfscanf
74946 18.8636 libc-2.3.5.so bash memcmp
39221 9.8718 libc-2.3.5.so bash ____strtol_l_interna
33493 8.4300 libc-2.3.5.so bash memmem
14853 3.7384 libc-2.3.5.so bash memset
13545 3.4092 vmlinux vmlinux __flush_dcache_icach
12345 3.1072 vmlinux vmlinux clear_pages
11064 2.7848 libc-2.3.5.so bash _IO_sputbackc
8806 2.2164 libc-2.3.5.so bash fscanf
8393 2.1125 libc-2.3.5.so bash _wordcopy_fwd_aligne
4524 1.1387 libc-2.3.5.so bash __strtol_internal
3824 0.9625 vmlinux vmlinux __copy_tofrom_user

vscanfとmemcmpが主のオーバヘッド

debian

チューニング後

samples % image name app name symbol name
4154 10.6425 vmlinux vmlinux __flush_dcache_icache
3301 8.4572 vmlinux vmlinux clear_pages
2855 7.3145 bash bash (no symbols)
2727 6.9866 libc-2.3.5.so bash memset
2078 5.3238 ld-2.3.5.so bash do_lookup_x
1033 2.6465 vmlinux vmlinux copy_page
813 2.0829 vmlinux vmlinux default_idle
769 1.9702 ld-2.3.5.so bash strcmp
741 1.8984 libc-2.3.5.so xargs getc
434 1.1119 vmlinux vmlinux flush_hash_patch_B
417 1.0684 vmlinux vmlinux unmap_vmas
379 0.9710 ld-2.3.5.so bash _dl_relocate_object

fork/execに必要なオーバヘッドが主となっている

debian

余談: realksh.c

- LKMLに投稿したらGregKHから、 面白い」と返事が返ってきた

```
$ sudo ./realksh.c
REAL ksh: printk ("hello¥n");
  Building modules, stage 2.
KMSG: <4>hello

REAL ksh: printk ("%x¥n", mfmsr());
  Building modules, stage 2.
KMSG: <4>9032

REAL ksh: printk ("%x¥n", mfspr(SPRN_MMCR0));
  Building modules, stage 2.
KMSG: <4>0

REAL ksh:
```


debian

iBook G4以外でこの知識は適用できるか

- 未確認だが、応用は簡単にできそう
- 2000年以降のiBook G3はIBMの750チップ
  - MPC7450と似ている機構
- 2001年以降のPowerBook G4は初代以外はMPC7450系
- iBook G4は初代が7455、2004年以降は7447Aのようだ
- MacMiniは7447A
- 参考：
  http://apple-history.com/?page=gallery&model=ibook_g4_ear_04

debian

おち

- おち
  - 実はMPC7450向けのoprofile support用のパッチはすでに書かれていた
    - Andy Fleming 「Patch: G4+ oprofile support」
    - http://patchwork.ozlabs.org/linuxppc/patch?id=3745
  - 事前に調査しているときには見つけられなかったのだが、自分で実装したあとだとgoogleでひっかかりそうなキーワードがわかるので、それで検索してみると見つかった

debian