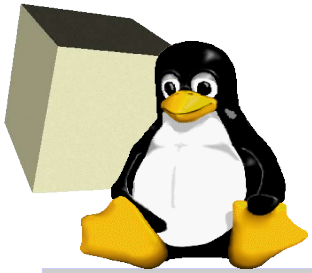


Linux カーネルの 省電力制御機構とパフォーマンス制御

株式会社 NTT データ　事業戦略部

横浜リナックスユーザーズグループ
三浦　広志

Agenda

- Linuxにおけるパワーマネジメントの概要
- Software Suspendの紹介
- CPUFreqの紹介
- デモンストレーション
- まとめ

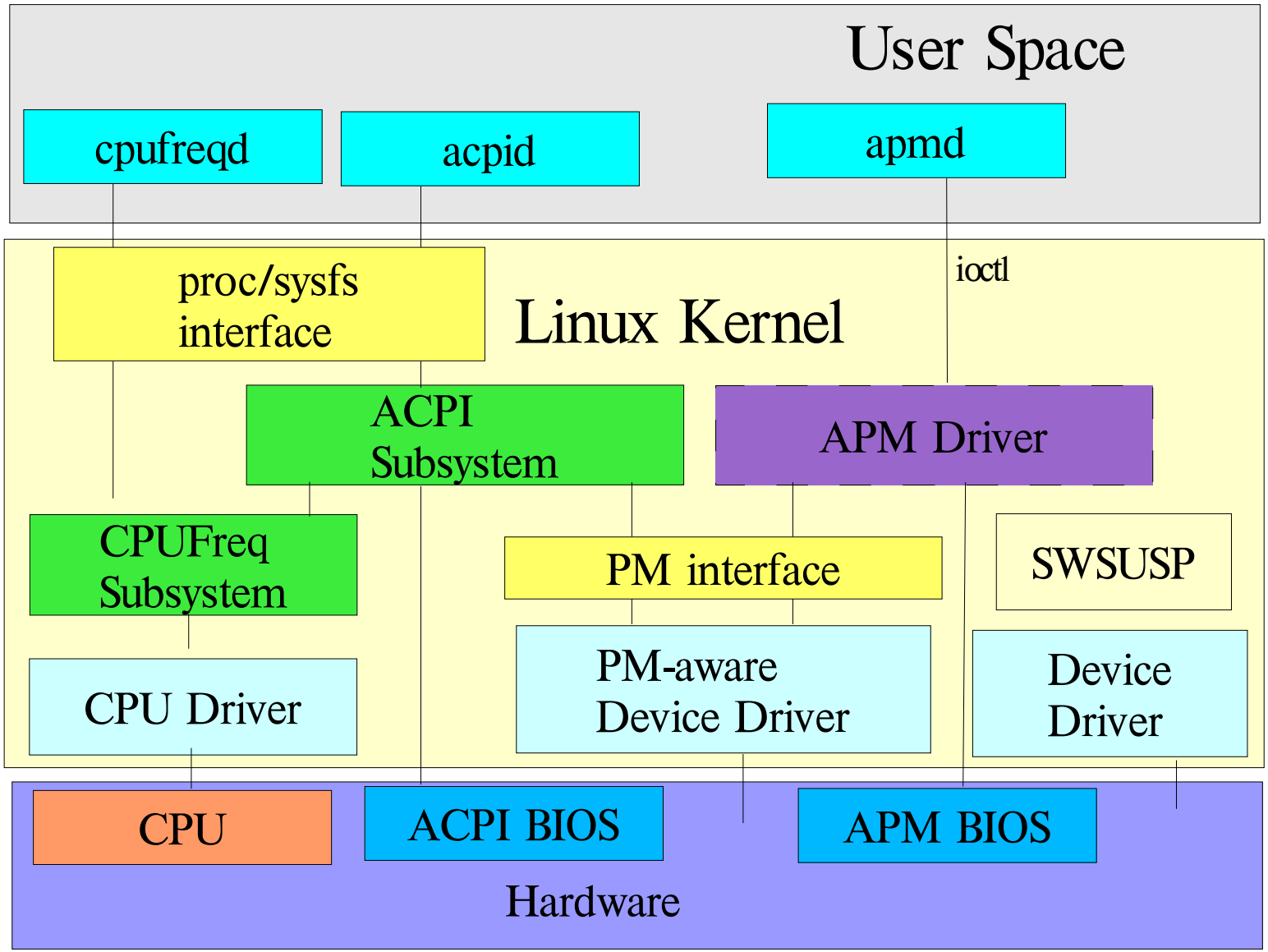
パワーマネジメントモデル

- パワーマネジメントの実装
  - Andrew Henroid による実装
    - PM サブシステムの立ち上げ
      - kernel/pm.c
      - include/linux/pm.h
    - コールバック関数
    - ドライバ呼出の順番を保証できない問題点がある
    - 2.4 系でもマージ済み
  - OSDL の Patrick Mochel による開発 (after 2.6.0)
    - アーキテクチャ非依存
    - コールバック関数の呼出順番を保証
  - 2 種類の管理領域
    - システムパワーマネジメント (SPM)
    - デバイスパワーマネジメント (DPM)

各ドライバの実装状況

- ACPI ドライバの実装
  - Intel の ACPI-CA によりほぼ完成状態
  - 個々のノート PC などで問題あり
    - ACPI では BIOS のバグをドライバで吸収可能
    - Linux カーネル側でバグのある BIOS の対応を全てとることは困難
- APM
  - レガシー API であるが利用可能
  - 古い BIOS のバグへの対処は 2.6 では削除されている
- CPUFreq
  - x86 の有名な CPU のドライバはほぼ実装完了
  - 組込み系に未開拓の分野がある
- SWSUSP
  - ソフトウエアサスペンド→ハイバネーションを実現

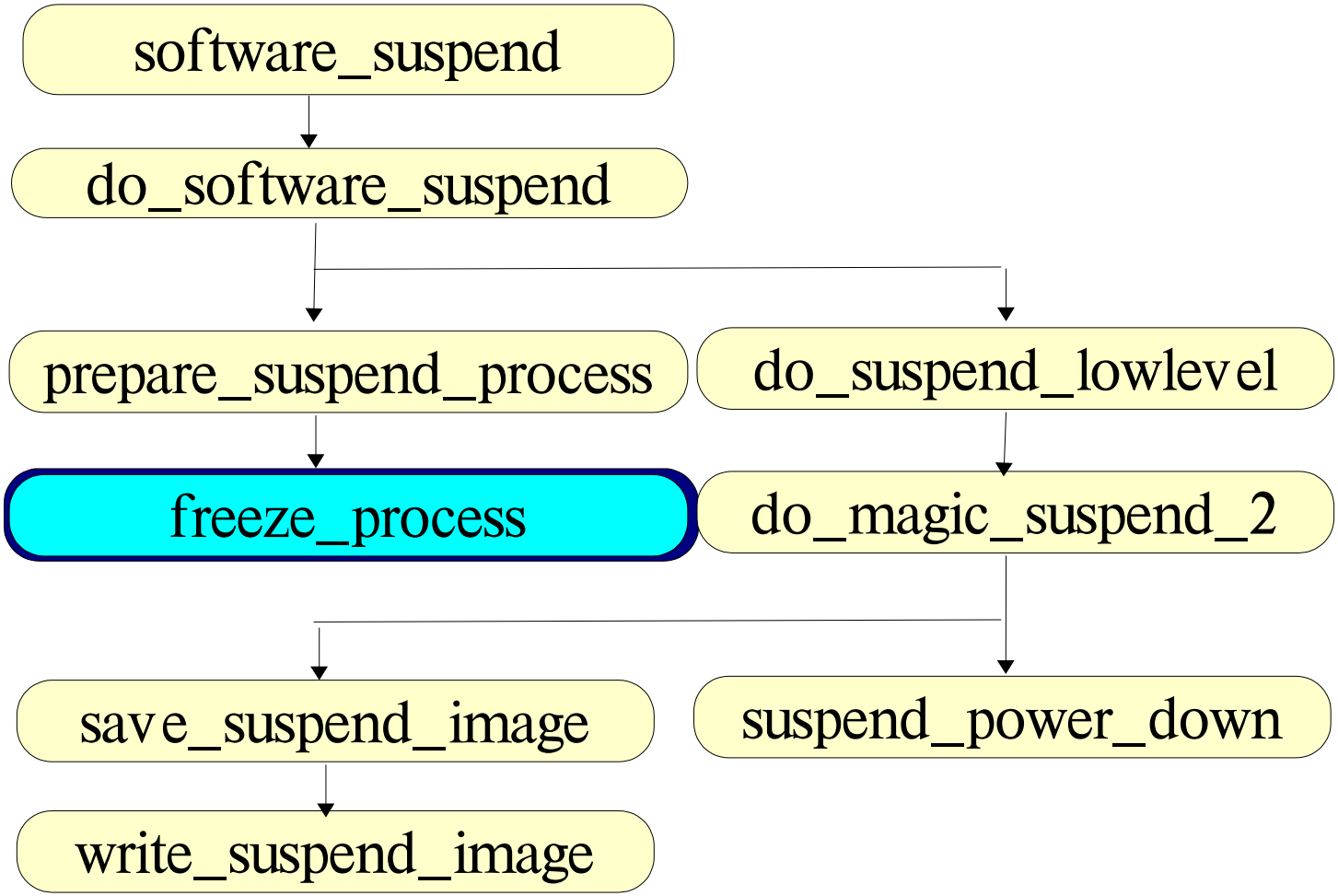
パワーマネジメントシステム俯瞰



Software Suspend

- Software Suspend とは
  - メモリ中の実行状態を一旦停止し、ハードディスクへ保存して、次回起動を高速に行う機構
  - 休止状態、ハイバネーションともいう
- ACPI 以前のシステムでは
  - APM BIOS によって OS の関与なしに実行
    - → OS にはサスペンドとハイバネーションの区別なし
  - Os は、 BIOS の処理内容に関与できない
- その他の特徴
  - スワップパーティションやスワップファイルに退避
  - 復旧時に退避イメージを保存するオプション
    - → ノード移送へ利用可能。

Software Suspend の内部動作



Software Suspend

- Software Suspend の現在の状況
  - カーネル 2.6
    - Pavel Machek によるコードがマージされている状況
    - 主に Pavel と Patrick の間で Software Suspend の扱いで論争になる
      - Power Management サブシステム実装の選択肢の 1 つの位置づけ
  - カーネル 2.4
    - Pavel の後を受けて Nigel Cunningham が完成度を高め、高速化と安定化が進められている
      - http://swsusp.sourceforge.net/
  - Kernel 2.6 へのマージを開始した

    → alpha1 版 (10/25)

      - bk://swsusp25.bkbits.net/main

デモ (SWSUSP)

- Software Suspend の例
  - Linux Kernel 2.4.23-pre6+swsusp 2.0-rc2
  - Let's Note CF-R1
    - ULV-Pentium III-M 800MHz
    - Phoenix ACPI BIOS
    - grub カーネルパラメータが必要

```
kernel      /boot/vmlinuz-2.4.23-pre6 root=/dev/hda2 ro acpi=on vga=791
                resume=/dev/hda3 splash=silent
initrd      /boot/initrd.splash
```

    - hibernate スクリプトによって処理実施
      - スクリプトは未対応のカーネルモジュールをサスペンド前に削除
    - echo 1 > /proc/swsusp/activate
      とすることで、強制的にサスペンド可能

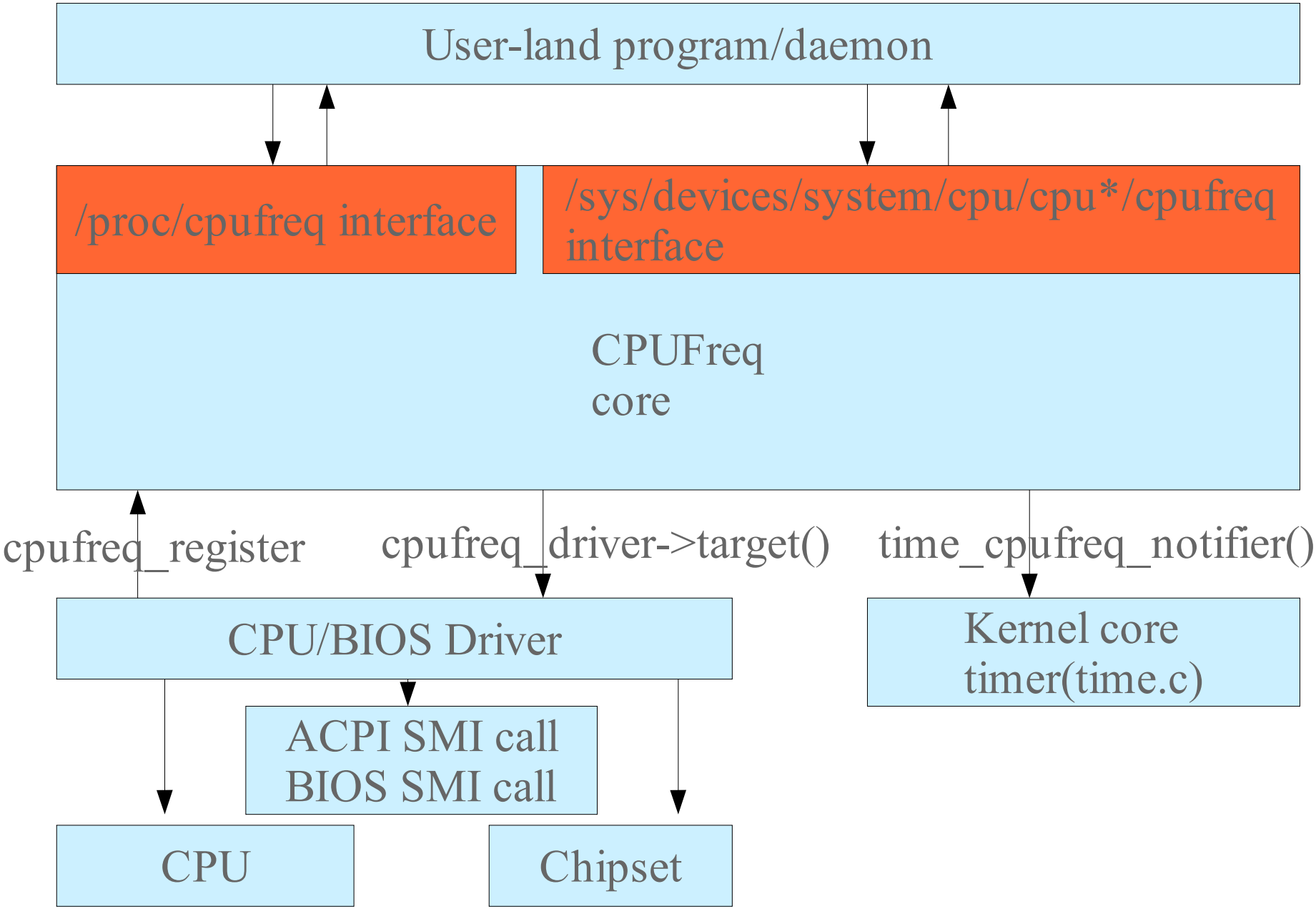
CPUFreq とは

- CPU パフォーマンス制御フレームワーク
  - パフォーマンス制御可能な CPU のサブシステム
    - → 組込み用 CPU、ノート PC 用 CPU を中心にサポート
      - MIPS、 Geode などの組込み CPU
      - Mobile Pentium III や Mobile Athlon などのノート PC 向け
  - 各社の制御技術にネイティブ対応
    - → Intel SpeedStep
    - → AMD PowerNOW!
    - → Transmeta LongRun
- 各種の自動制御用デーモンが活発に開発
  - cpufreqd, cpudynd などによって、 CPU を動的に制御

CPUFreq 実装のポイント

- グローバルタイマー定数の変更
  - カーネル内部の計時は、クロックカウンタ TSC を最大限活用したものになっている
  - 周波数が変わる→１クロックあたりの時間が変わる
    - → 周波数の変更に同期させて大域変数変更
    - → loops_per_jiffy, cpu_khz を調整
- CPU ネイティブドライバ
  - CPU、チップセットによって制御方法、実現の方式が異なる
  - MSR やチップセットのレジスタ設定を行うことで制御
  - BIOS コールを利用するドライバも提供

CPUFreq の API



デモ /CPUFreq

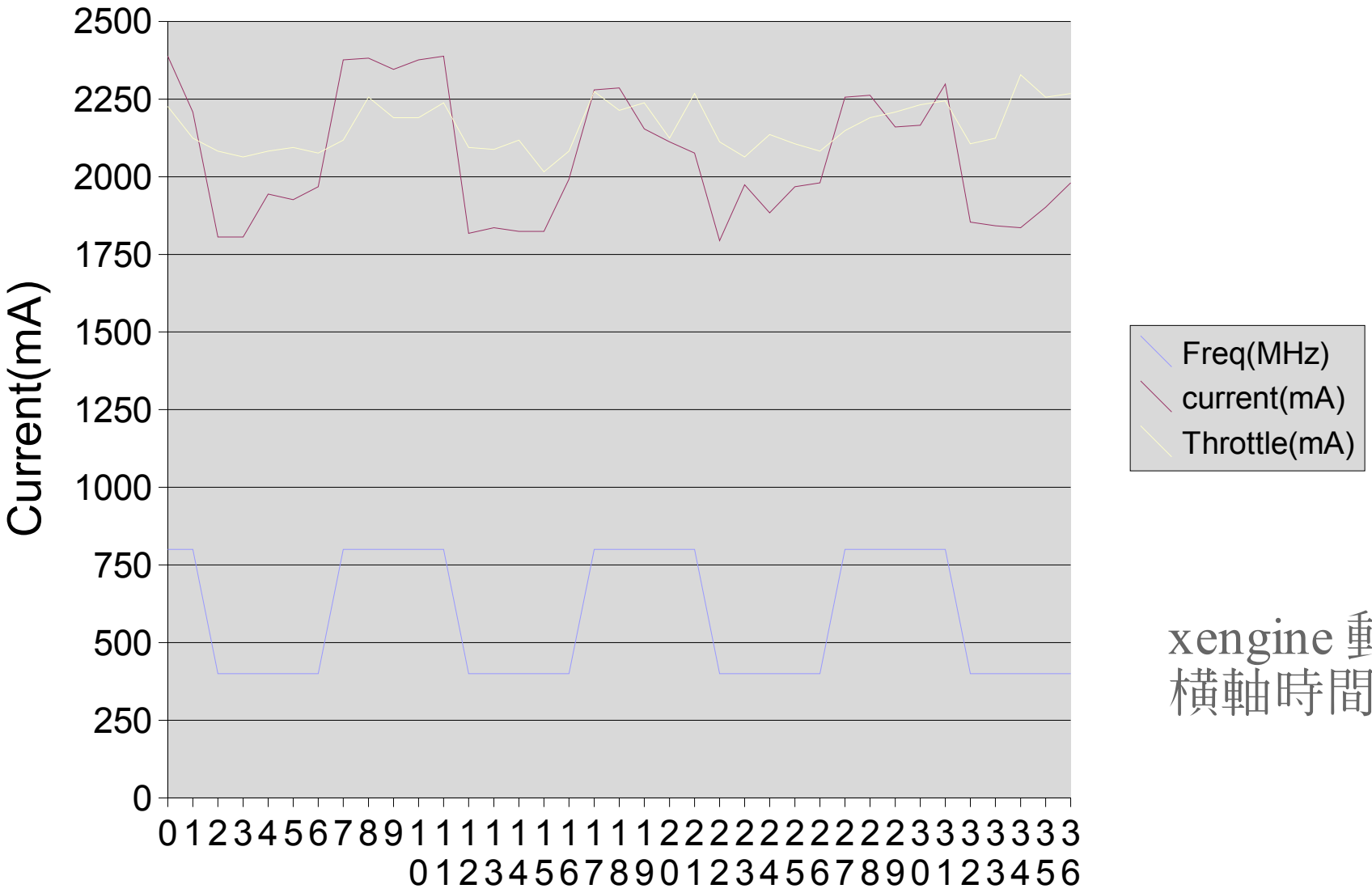
- CPUFreq の例 1
  - Cyrix MediaGXm|NatSemi/AMD Geode GX1 200MHz
    - → Linux Kernel 2.6-test9
    - → gx-suspmod driver
  - userspace governor
    - → １０段階の速度変更が可能
- CPUFreq の例 2
  - Intel SpeedStep Technology
    - → Linux Kernel 2.4.23-pre6
    - → speedstep-smi driver
  - ACPI processor thlottling

Demo code

```
#! /usr/bin/perl


$ver = 2.4;
# $ver = 2.6;
$n_stat = 2;
$max_freq = 800000;
$acpi_batt = "/proc/acpi/battery/BAT A/state";


if ($ver > 2.5) {
        open PROC,">/proc/cpufreq";
        printf PROC "0%100%userspace\n";
        close PROC;
}


for ($i = 0;$i <= 20; $i++) {
        if ($ver > 2.5 ) {
                open SYS, ">/sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed";
                $sfreq  = int((($i % $n_stat +1)/$n_stat) * $max_freq);
                printf SYS "$sfreq\n";
                close SYS;
        } else {
                if ($n_stat == 2) {
                        open PROC,">/proc/cpufreq";
                        printf PROC (($i % $n_stat)?"0:400000:800000:performance\n":"0:400000:800000:powersave\n");
                        close PROC;
                } else {
                        open PROC,"/proc/cpufreq";
                        $sfreq  = int((($i % $n_stat +1)/$n_stat) * $max_freq);
                        printf PROC "0:$sfreq:performance\n";
                        close PROC;
                }
        }
        sleep(1);
        if ( -e $acpi_batt) {
                for ($j = 0; $j < 5; $j ++) {
                        open ACPI, $acpi_batt;
                        @lines = <ACPI>;
                        printf "%s %s\n",(($i % $n_stat)?"800":"400"),substr($lines[3],25,4);
                        close(ACPI);
                        sleep(1)
                }
        }
}
```

電圧／周波数制御と消費電力

ACPI Throttle と SpeedStep



xengine 動作時
横軸時間 (sec)

まとめ

- 開発したこと
  - CPUFreq のための CPU ドライバ (speedstep, geode)
  - SWSUSP と CPUFreq の双方が共存できるよう改善
- カーネルの省電力機構を紹介した
  - CPU 周波数制御、パフォーマンス制御は、ネイティブドライバ、 ACPI、 BIOS ともに整備が完了
  - ACPI S4/SWSUSP は 2.6 カーネルのマージ段階
- 今後の課題
  - ノートパソコンの ACPI BIOS バグに起因する不具合への対応
  - ビジネス / マルチメディアアプリケーションなど、利用シーンに応じた最適制御方法の開発

参考資料

CPUFreq の対応 Daemon

- cpudynd
  - cpudyn (http://mnm.uib.es/~gallir/cpudyn/)
    作者　Ricardo Galli (gallir AT uib dot es)

  - CPU の負荷が増大したときだけ、パフォーマンスを増加させるデーモン
  - CPUFreq のほか、 ACPI processer throttling にも対応
- cpufreqd
  - SpeedStep Applet と同じことを行うデーモン
  - ACPI バッテリ情報や APM 情報を元に、 AC 電源の状態によって、モードのスイッチを行う

CPUFreq の導入方法

- 2.4 系のカーネルの場合
  - alan cox パッチを使う
    - ftp://ftp.jp.kernel.org/pub/linux/people/ac/
  - CVS から開発版を入手する。

    :pserver:cvs@pubcvs.arm.linux.org.uk:/mnt/src/cvsroot

```
$cvs -d :pserver:cvs@pubcvs.arm.linux.org.uk:/mnt/src/cvsroot login
password:
$cvs -d :pserver:cvs@pubcvs.arm.linux.org.uk:/mnt/src/cvsroot co cpufreq
$cd cpufreq
$./patchin.sh /usr/src/linux
```

- 2.5/2.6 系カーネルの場合
  - 統合済み

SWSUSP の入手方法

- 2.6 系カーネル
  - 古い software suspend を元にしたコードがマージ済み。
    - 速度は遅いが安定して利用可能
  - 最新の開発に基づく α 版
    - http://swsusp.sourceforge.net/
- 2.4 系カーネル
  - 現在の開発が 2.4.22 を元に行われている
    - http://swsusp.sourceforge.net/
    - 開発 ML で不具合対策パッチを配布している