Ruby 処理系の構想（妄想）

東京大学大学院
情報理工学系研究科 創造情報学専攻
笹田耕一
<sasada@ci.i.u-tokyo.ac.jp>

偉い人曰く

- **Dave Thomas**@RubyConf 2008 Keynote
  - Programmer は Tool を愛さないといけないよ（意訳）
  - Ruby も愛しているよね（意訳）
  - http://rubyconf2008.confreaks.com/keynote.html
- そろそろ Fork してもいいんじゃない？（意訳）

Ruby の可能性

- Ruby には，まだまだ可能性がいっぱい（多分）
- 理想の Ruby を Fork とか関係なく，とりとめもなく（やまもおちもなく）考えてみよう
- とくに，ささだ視点から

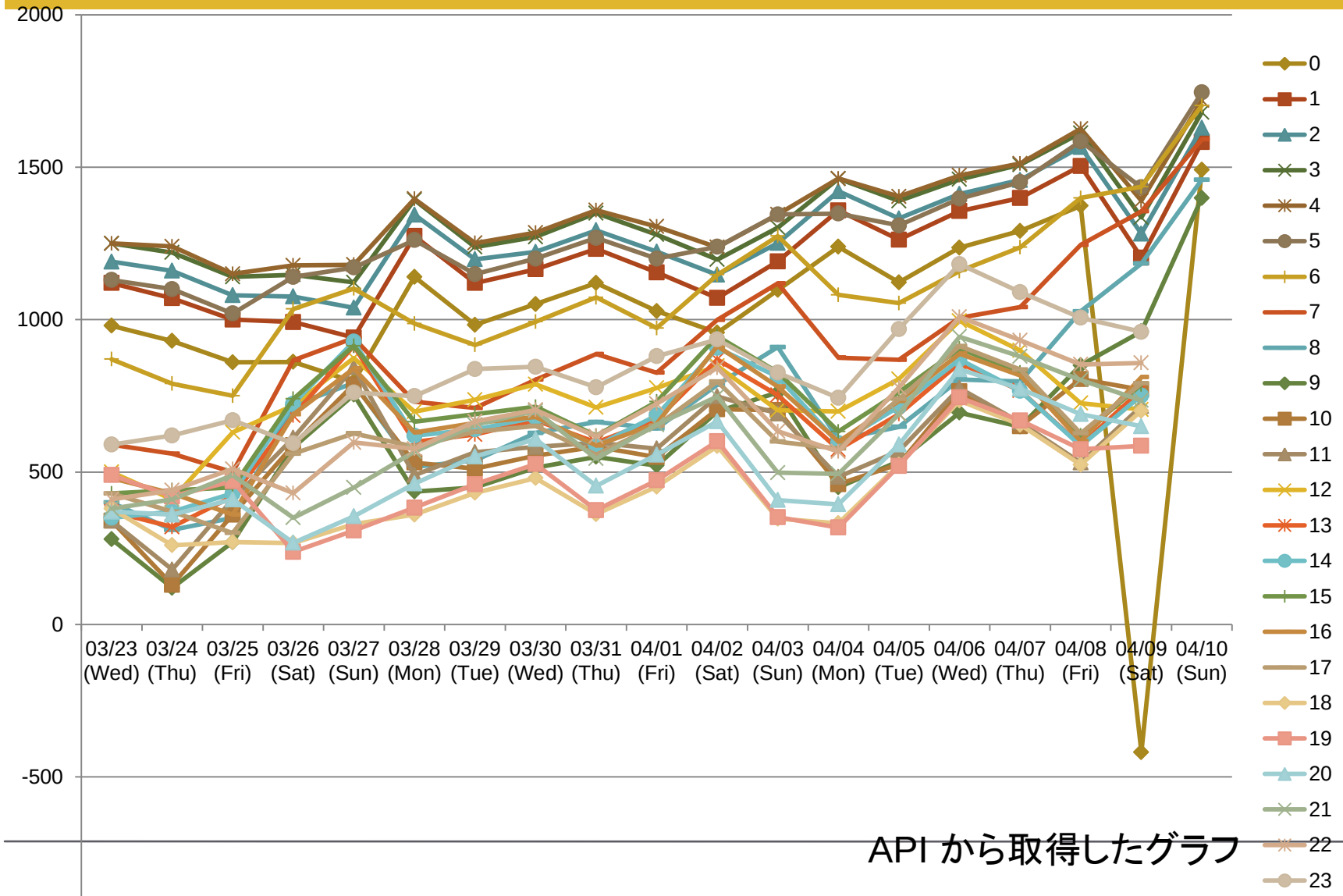
組み込みシステム向けRuby

- ユビキタス Ruby（？）
- 軽量 Ruby
  - → RiteVM
  - Linux Kernel みたいに機能を選択可能に
  - アプリが使う機能を（半）自動判別して組み込んだり組み込まなかったりするコンパイラ（学生さん）
- リアルタイム Ruby
- 省電力 Ruby

省電力Ruby

- 電力消費はいろいろ問題

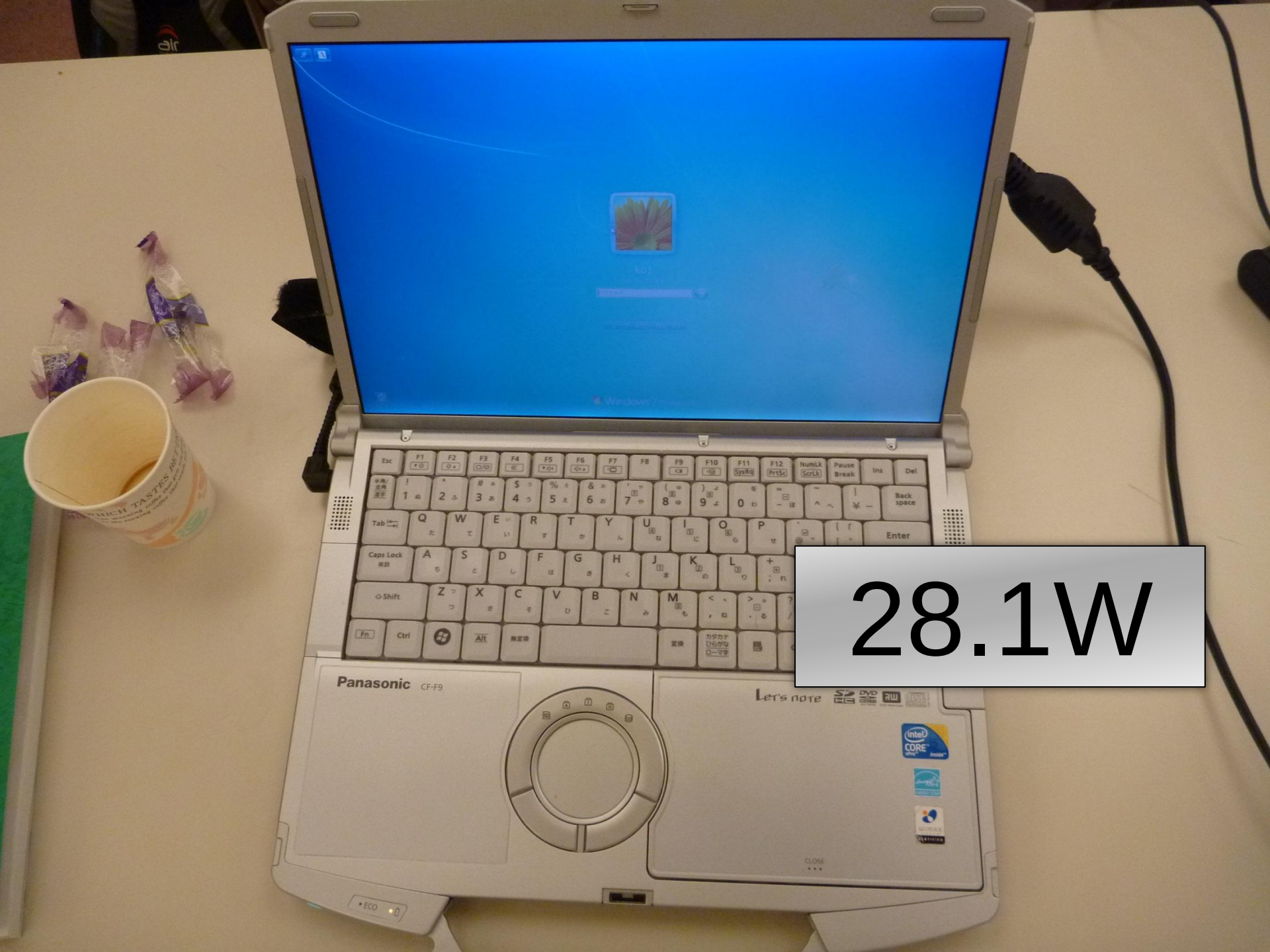
電力の余力（東京電力）



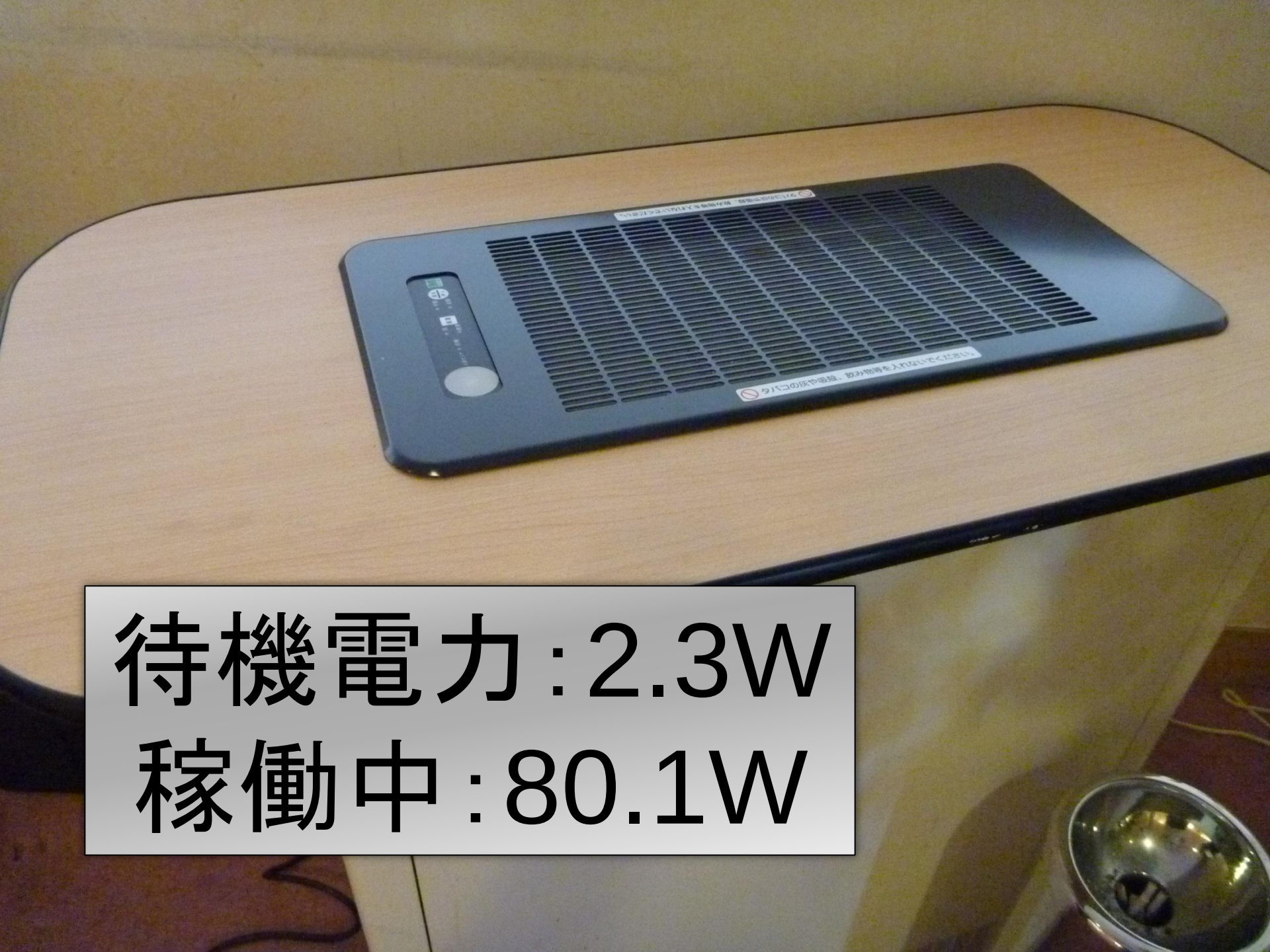
API から取得したグラフ


ES
watts up? PRO
145.7
WATTS
LOG
SELECT
MODE


第52回プログラミング・シンポジウム
様
25.0W


28.1W
Panasonic
CF-F9
Let's note


保温（上）
だけ
145.7W
UNIMAT OFFISCO
適温
保温(下)
保温(上)
入
切
入
切
UNIMAT OFFISCO


待機電力:2.3W
稼働中:80.1W

単位時間あたりに消費する電力を抑えるRuby処理系（4/1）

[ruby-dev:43373] 単位時間あたりに消費する電力を抑えるRuby処理系 - ruby-dev - ko1@atdot.net - Mozilla Thunder...

ファイル(F) 編集(E) 表示(V) 移動(G) メッセージ(M) 予定とToDo(N) OpenPGP(N) ツール(T) ヘルプ(H)

ruby-dev - ko1@atdot.net | カレンダー | [ruby-dev:43373] 単位時間...

差出人 SASADA Koichi <ko1@atdot.net>

件名 **[ruby-dev:43373] 単位時間あたりに消費する電力を抑えるRuby処理系** 2011/04/01 2:30

宛先 ruby developers list <ruby-dev@ruby-lang.org> その他の操作

ささだです．

Ruby 処理系が単位時間あたりに消費する電力を，極力抑えるためのパッチを作成しましたのでお送りします(*1)．先行研究 [1] ではほんの少し抑えるだけでしたが，本パッチを適用することで Ruby 処理系を実行する前とほぼ同様の時間あたりの消費電力で済むようになります．

*1: 本パッチを活用するには DVFS [2] に対応したプロセッサが必要になります．例えば最近のラップトップには殆ど対応していると思います．

まずは，実行結果です．一番簡単に電力を消費するプログラムを作るには，無限ループ（loop{}）を実行するのが簡単です．このプログラムを変更前，変更後の Ruby 処理系で試した結果が図1 になります．

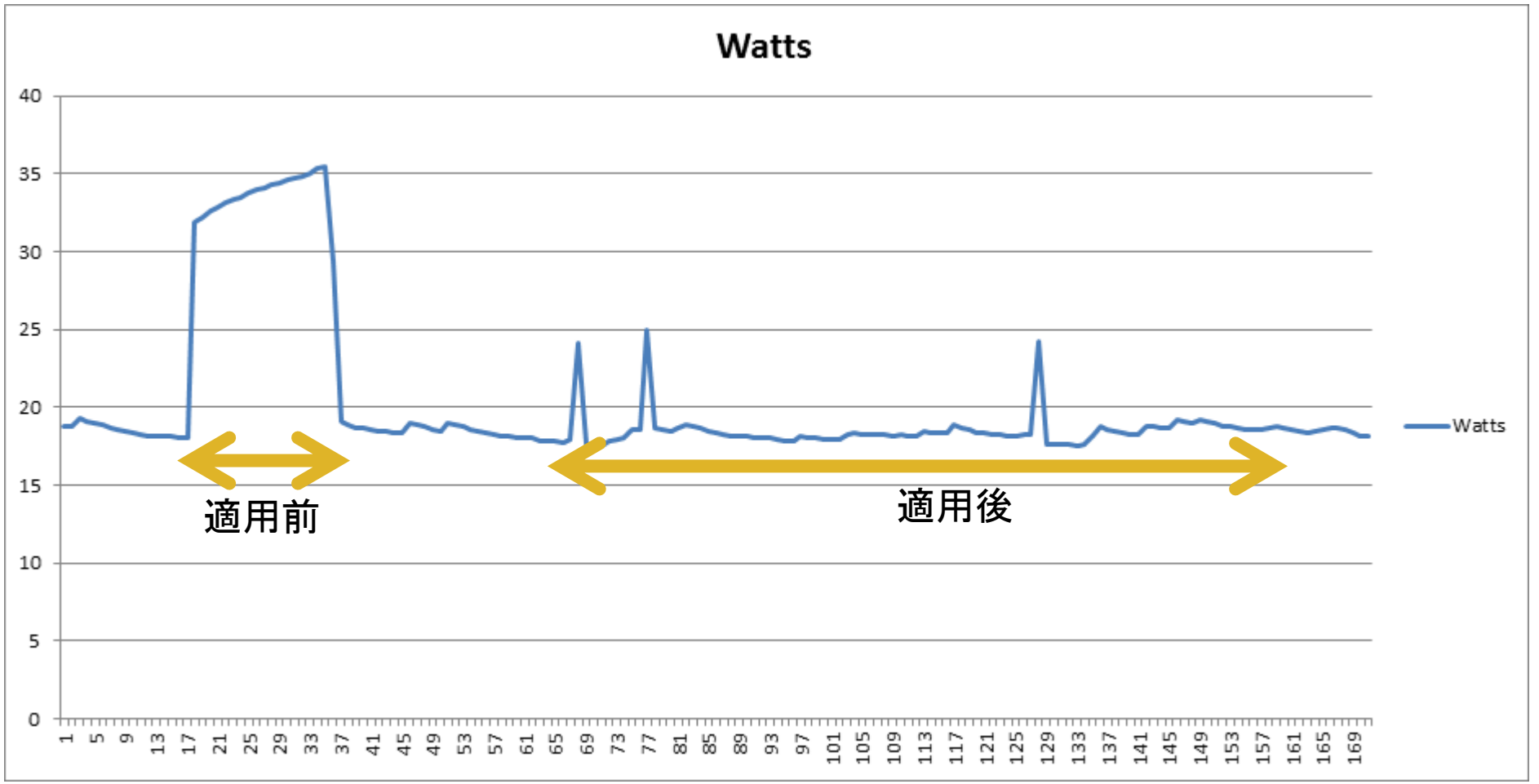
==========

図1 消費電力の推移．横軸が時間（秒），縦軸が消費電力（watt）
評価環境：Let's note F9（Intel(R) Core(TM) i5 CPU M 540 @ 2.53GHz）
Windows 7 上の VirtualBox 上の
Linux 2.6.32-5-686 (Debian/squeeze)
watts up? PRO を用いて計測

http://www.atdot.net/fp_store/f.rplxil/file.graph.png

==========

図1の最初の大きな山（横軸 17〜37 あたり）が現在の trunk（ruby 1.9.3dev


Watts
40
35
30
25
20
15
10
5
0
1
5
9
13
17
21
25
29
33
37
41
45
49
53
57
61
65
69
73
77
81
85
89
93
97
101
105
109
113
117
121
125
129
133
137
141
145
149
153
157
161
165
169
Watts
適用前
適用後

```
Index: vm_core.h
===================================================================
--- vm_core.h          (revision 31226)
+++ vm_core.h          (working copy)
@@ -703,9 +703,10 @@
 void rb_thread_lock_destroy(rb_thread_lock_t *);

 #define RUBY_VM_CHECK_INTS_TH(th) do { ¥
-  if (UNLIKELY((th)->interrupt_flag)) { ¥
-    rb_threadptr_execute_interrupts(th); ¥
-  } ¥
+    sleep(1); ¥
+    if (UNLIKELY((th)->interrupt_flag)) { ¥
+            rb_threadptr_execute_interrupts(th); ¥
+    } ¥
 } while (0)

 #define RUBY_VM_CHECK_INTS() ¥
```

捕捉:
VMの数命令実行するごとに1秒間スリープするので,「単位時間」あたりの仕事量は減ります. が, 仕事時間がものすごい増えるので, トータルな消費電力はものすごい増えます.

嘘は言ってないんだけど, あのグラフを見て「おー」とか言っちゃう人は, 騙されやすい人かもしれません.

真面目に（Prosym52既発表）タイマスレッドによる消費電力増

- 一定時間（t = 10ms）ごとにフラグをセット
  → 現在の実装では，**while (1) {sleep(t); flag = 1}** のような実装
- **誰も flag をチェックしないような状況でも，定期的にタイマスレッドだけは定期的に起動**
  - 1 thread しか走っていないとき
  - すべての Thread が sleep しているとき
- CPUが低消費電力状態（halt）を維持出来ない
  - システム全体がアイドルな場合（**アクセスのないサーバ**等）
  - 最近のCPUが持つDVFS（動的電圧・周波数制御）
  - powertop というツールを使うと簡単に観測可能

消費電力
解決案の検討

- GVL解放タイミングをどのように通知するか？
  (1) setitimer など，タイマシグナルを活用
  (2)「GVLを待っているRubyスレッド」が通知
  　　Pythonの新しい版はこうしているらしい
  (3) やっぱりタイマスレッド
  　　ただし，誰もGVLを待っていない時は起きないように
- 議論
  - (1) は計算機・OSの構成によって定期的にシグナルが来ない時があった（**フェアネスに問題）**
  - (2) は，たくさんスレッドが要る場合面倒
  - というわけで，(3) に

消費電力
解決策の詳細

- 必要なときだけタイマスレッドを活性化
  - 普段は無限に待つ（名前無しセマフォ）
  - GVLを待つスレッドが現れたら定期的に起きるように
  - シグナルなんかも同じ仕組み

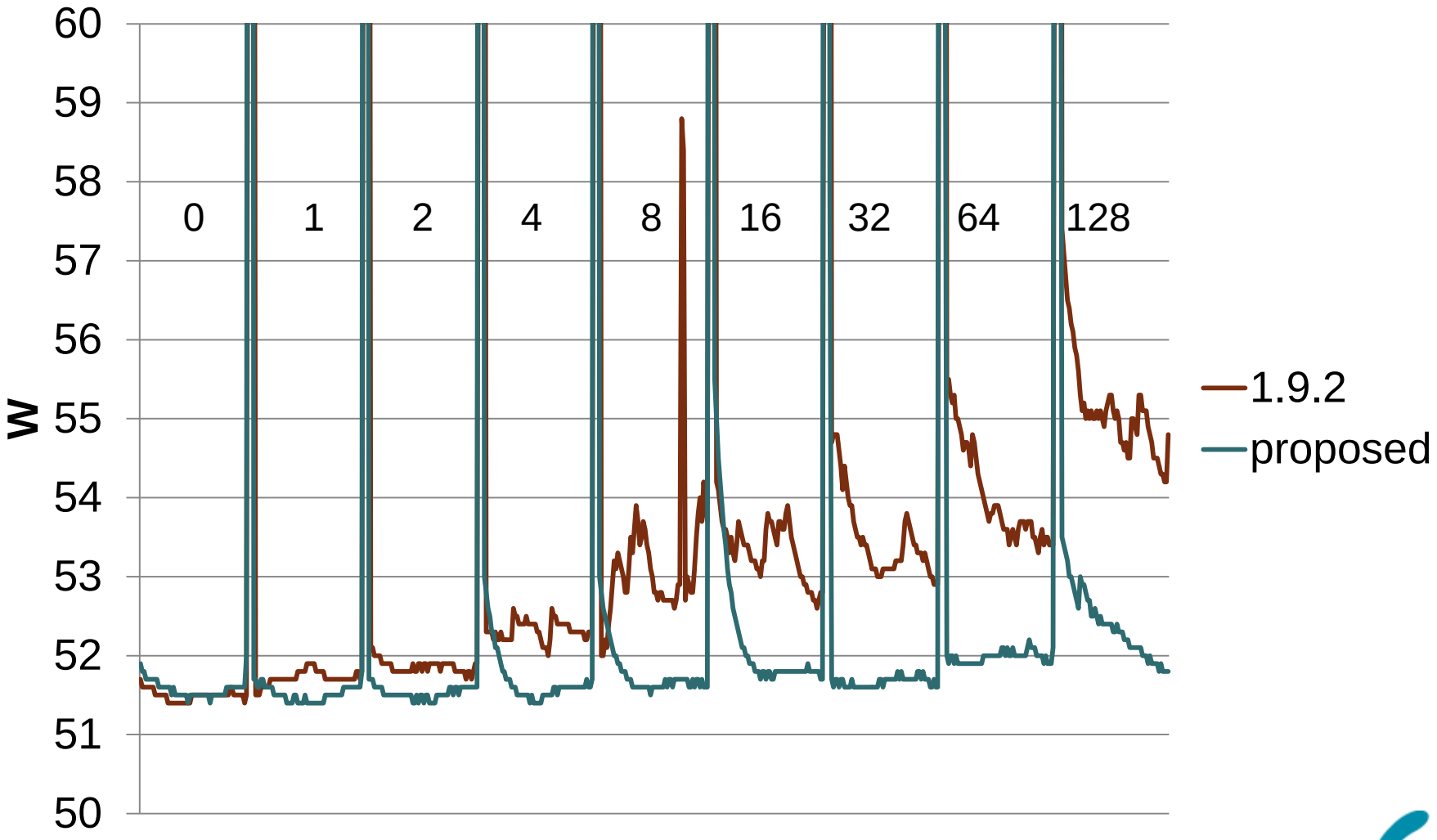
- セマフォが使えない場合は特別のケアが必要
  - 例えば MacOS X では名前無しセマフォはサポートしていないらしい

消費電力
解決した結果の評価

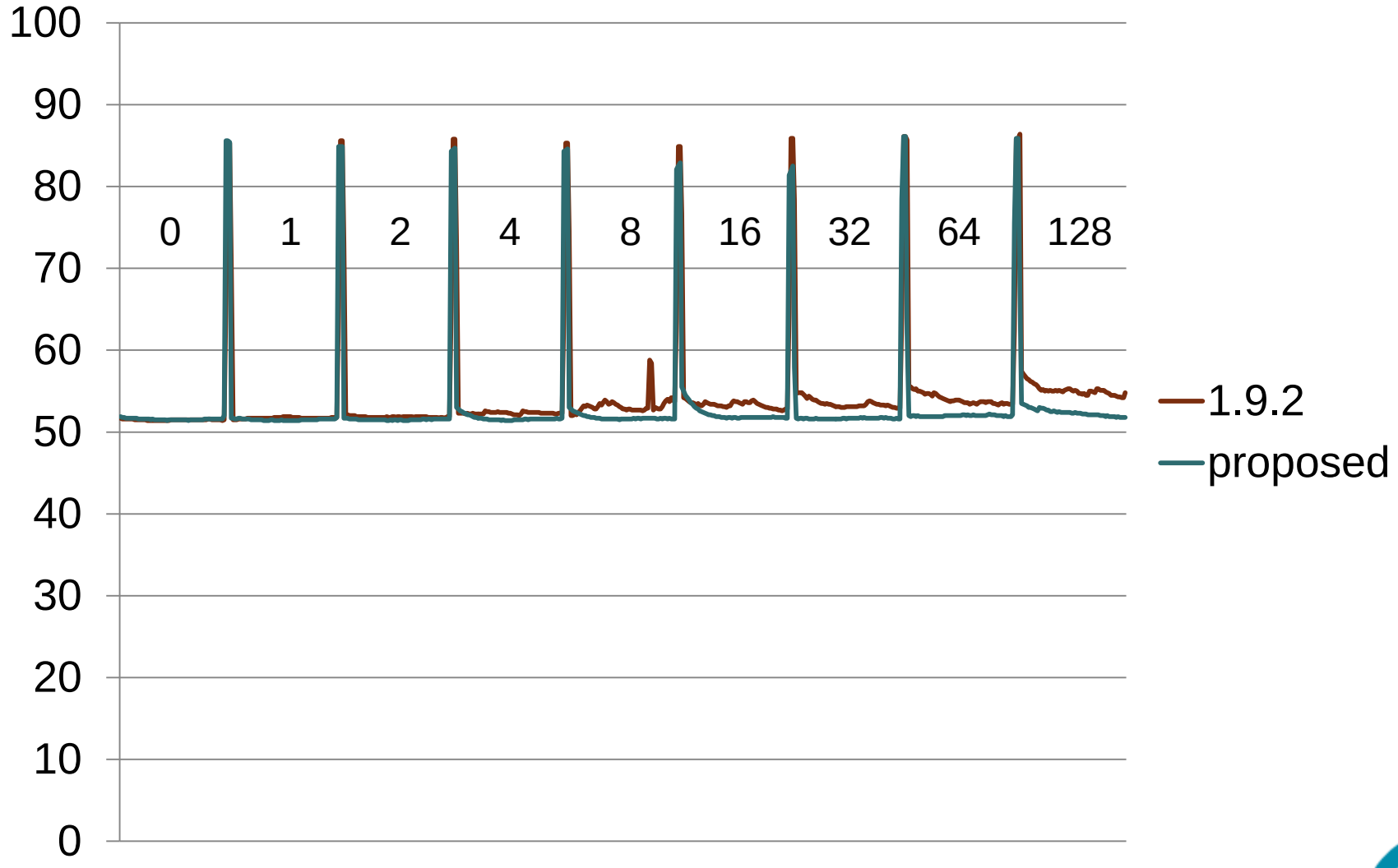
- というか，実際どれくらい「電力消費が上がっていたのか？」の評価
  - 「苦情」来たけど，実際どうなのよ
  - 解決していれば電力消費が 0 になるから自明
- 評価環境
  - Intel Core i5（4 core）
  - Ubuntu Server 10.10
  - 電力計：Watts Up? PRO
    電力消費/秒のログをUSBで取得可能

- Idle Ruby process（sleep するだけ）がどれくらい電力を消費するのか，を測定

消費電力
消費電力の様子（？）



消費電力
消費電力の様子（真）



消費電力
電力消費のまとめ

- IdleなRubyプロセスの消費電力0.3Wくらい削減
  - Idleシステムは **50W** くらい（Intel の速いCPUのPC）
- 電気代（東京電力）
  - 17.87円/1kWh (*1) × 0.3W/1000 × 30 × 24 = 3.86円（1月の電気代の節約）
  - 3.86 * 12 = 46.31円（1年の電気代の節約）
  - 10,000世帯にホームサーバ＋Ruby 1.9.2設置なら 46.31 * 10,000 = **463,190 円（年間）**

この研究は45万円ほどの成果

*1: 最初の 120kWh まで

速いRuby

- VM を書き直す（書き直したい）
  - コンパイラとかで頑張る
    - AOTコンパイラ（学生さん）
- GC とかを頑張る
- Ricsin（Ruby in C）を真面目に取り入れる
  - C only → Ruby + C
- I/O とかをもうちょっとなんとかする
  - I/O の中間層を真面目に作る？

並列実行する Ruby

- 並列実行する Ruby
  - 細粒度並列 → Cメソッド/JITコンパイル＋細粒度並列
  - 疎粒度並列 → MVM＋高速なVM間通信
    - Go language の channel のような実装
    - オブジェクトはきっちり分ける
- 分散環境の上の Ruby
  - “クラウド”（笑）用 Ruby
  - dRuby?
  - 大規模データを扱うためのスクリプト言語
    - Hadoop → Fairy
  - もうちょっと手軽にならないかな？
    - MVM の拡張
    - Migratable VM / Thread / Proc

H/W と仲の良い Ruby

- GPGPU を使う Ruby
  - CUDA
  - OpenCL
- FPGA を使う Ruby
  - 福岡のほうでやっているらしい（GC を boost したり）
  - アプリによって H/W
- RubyOS（農工大並木研）

型とかついてる Ruby

- Optional Type
- Strict Type

ディペンダブルRuby

- Dependability (wikipedia より)
  - Availability: readiness for correct service
  - Reliability: continuity of correct service
  - Maintainability: to undergo modifications and repairs
- SEGV しない Ruby
- プログラマがバグを出せない Ruby
- 高い放射線環境下でも実行できる Ruby
- セルフヒーリングとか

他の言語への変換（Ruby2???）

- JavaScript
  - いくつか，実装あったような
  - language=“rubyscript” は無理だろうな
- C#（学生さん）
- X10（学生さん）

近く実現出来そうな話

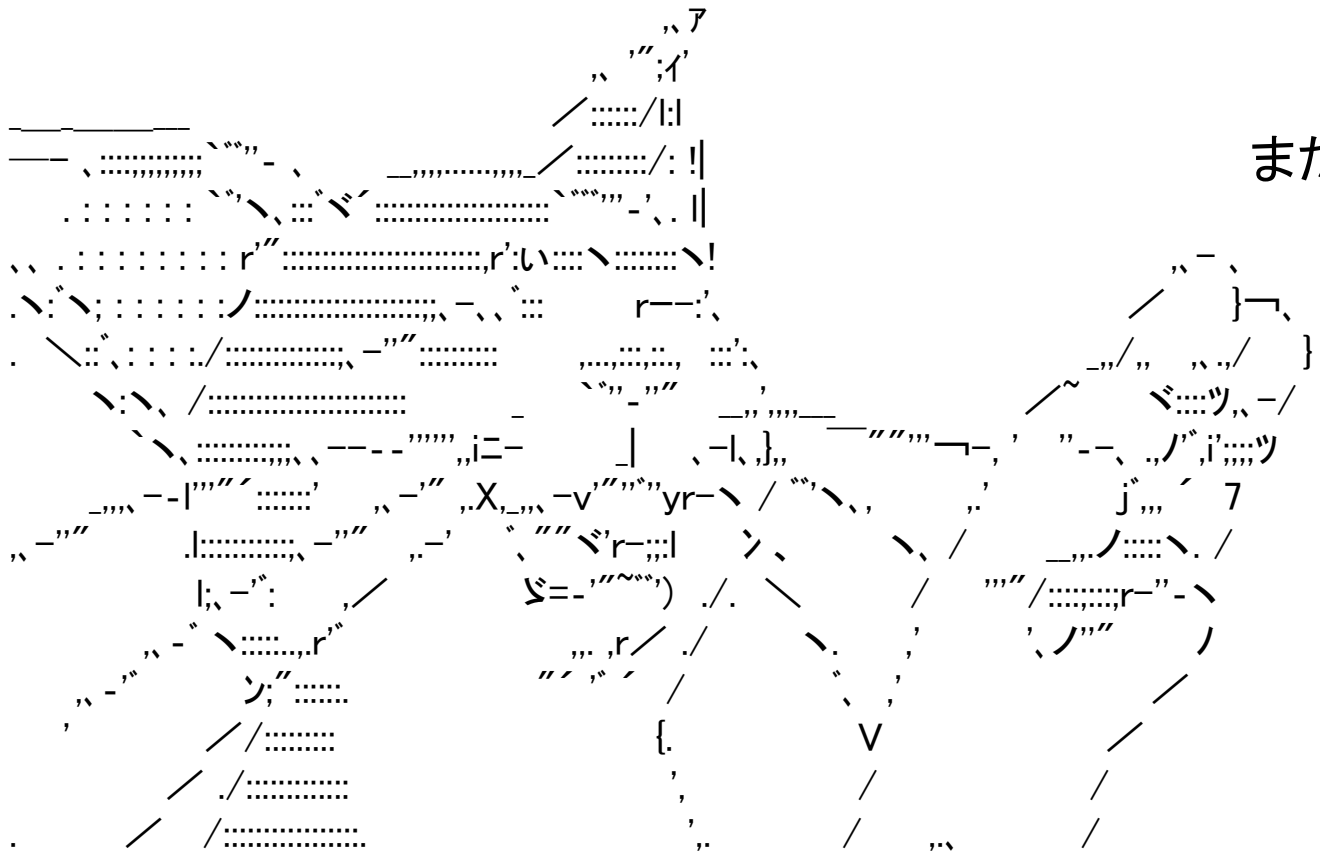
- かっこいいプロファイラ
  - Ruby用リアルタイム性能プロファイラ（学生さん）
    - http://sunagae.net/wiki/doku.php?id=software:llprof
    - まだ宣伝していないらしい
  - リモートプロセス/マシンから監視
  - コールツリーを見ることができる
  - 低負荷
- もうちょっとかっこよくしたい
  - スレッドの対応
  - メモリプロファイラの対応
- もっと，役立つ可視化はないかな？

近くに実現したい話

- 実用をとても意識した Ruby の AOT コンパイラ
  - 複数ファイルをまとめる
  - ロード時間を最小に
    - バイトコード表現の見直し
    - Ricsin ＋ Ruby to C コンパイラの組み合わせ
    - でも，ロードで一番時間がかかるのは定義（動的）な気もする

偉い人曰く

- Dave Thomas@RubyConf 2008 Keynote
- Ruby って，もうだいたい十分なんじゃない？



またまたご冗談を

またまたやること
沢山あるよね

続きは Ruby会議2011で

並列世界のRuby処理系

Ruby Interpreters in the Parallel Worlds

ささだこういち

RubyKaigi2011

2011/07/16-18

東京　練馬文化センター

とりとめもなく，おわり

Ruby 処理系の構想（妄想）

ささだこういち
ko1@rvm.jp

Questions?

Thanks:
ネタだしにつきあってくれた学生さん
猫先生を教えてくれた tarui さん