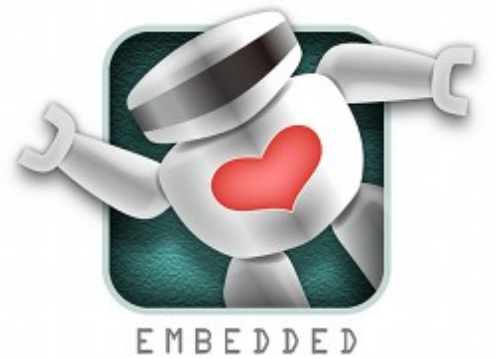


Summer Project: Microkernel

Jim Huang (黃敬群)<jserv.tw@gmail.com>,
June 2013

動機：

透過開放發展的模式，打造一個真正能用的系統軟體環境，提供給物聯網與醫療電子等應用

FAQ #1

「聽起來不錯，但我對 Operating System 不是這麼熟悉，有什麼項目能作？」

FAQ #1（答覆）

以 Linux 核心來說，佔了九成的程式碼是驅動程式與檔案系統，而非系統呼叫、排程器，或者任何典型你在教科書學習到的項目

後者的總和根本不到一成！

FAQ #2
「現在有哪些項目需要協助？」

FAQ #2（答覆）

目前的 microkernel 針對物聯網與醫療電子產品的需求去開發，專注於低功耗、無線通訊，以及系統的擴充能力。本體已有可運作的雛型。

但缺乏以下：

(A) 更好開發工具，得以分析執行時期的表現（功耗、效能，不當的系統呼叫等），進而調整系統

(B) 應用程式，特別是涉及 Bluetooth 4.0

(C) 文件！

FAQ #3
「這是用來牟利，還是做興趣的？」

FAQ #3（答覆）
兩者都有 :-)

Linus Torvalds 曾在論壇表示：
"I'm not doing anything big. Just something for fun."

最早此 microkernel 針對 AcoMo 公司的產品 [*] 而開發，但我們認為這符合多種新型應用的需求，沒必要藏私，更該透過社群的力量，使基礎建設變得更好，所以我們開放 microkernel 的原始碼 (BSD 授權），並招募開發者，以獲得更好發展。

[*] http://www.acomotech.com/en/portfolio/acomo-baby-hrv-monitor/

FAQ #4

「一直提” microkernel”，到底有沒有名稱？」

FAQ #4（答覆）

有！將在 COSCUP 2013 研討會發表相關成果，

目前的命名為” **f9 microkernel**”:

`https://github.com/f9micro/`

FAQ #5
「還打算用在哪裡呢？除了開發產品」

FAQ #5（答覆）

事實上，這預期用於大學課程，作為一個具體而微的教學系統，讓修課的同學得以分析研究，接著改善其系統效能並擴充特定的功能。

課程網址：

wiki.csie.ncku.edu.tw/embedded/schedule

FAQ #6
「我還是不懂，為何不用 Linux 呢？」

FAQ #6（答覆）

你想過在 Arduino 等級的硬體跑 Linux 嗎？這基本上是不可能的，除非你想重寫 Linux 0.11 這樣二十年前的老舊版本。

我們優先考慮低功耗但具備足夠運算能力 (CPU 時脈約 72 MHz) 的硬體環境，期許能發揮硬體特性

FAQ #7

「參與這個開發項目，對我來說有什麼好處？」

FAQ #7（答覆）

首先，這比較像是課程參與，會先安排一些教育訓練，然後大家討論出可行的題目，接著各自去實作，也鼓勵大家去提交 bug report。當然，若能打造相關的應用，就更好了 :-)

我們會對＊學生參與者＊提供工讀金，嘗試特定有挑戰的項目

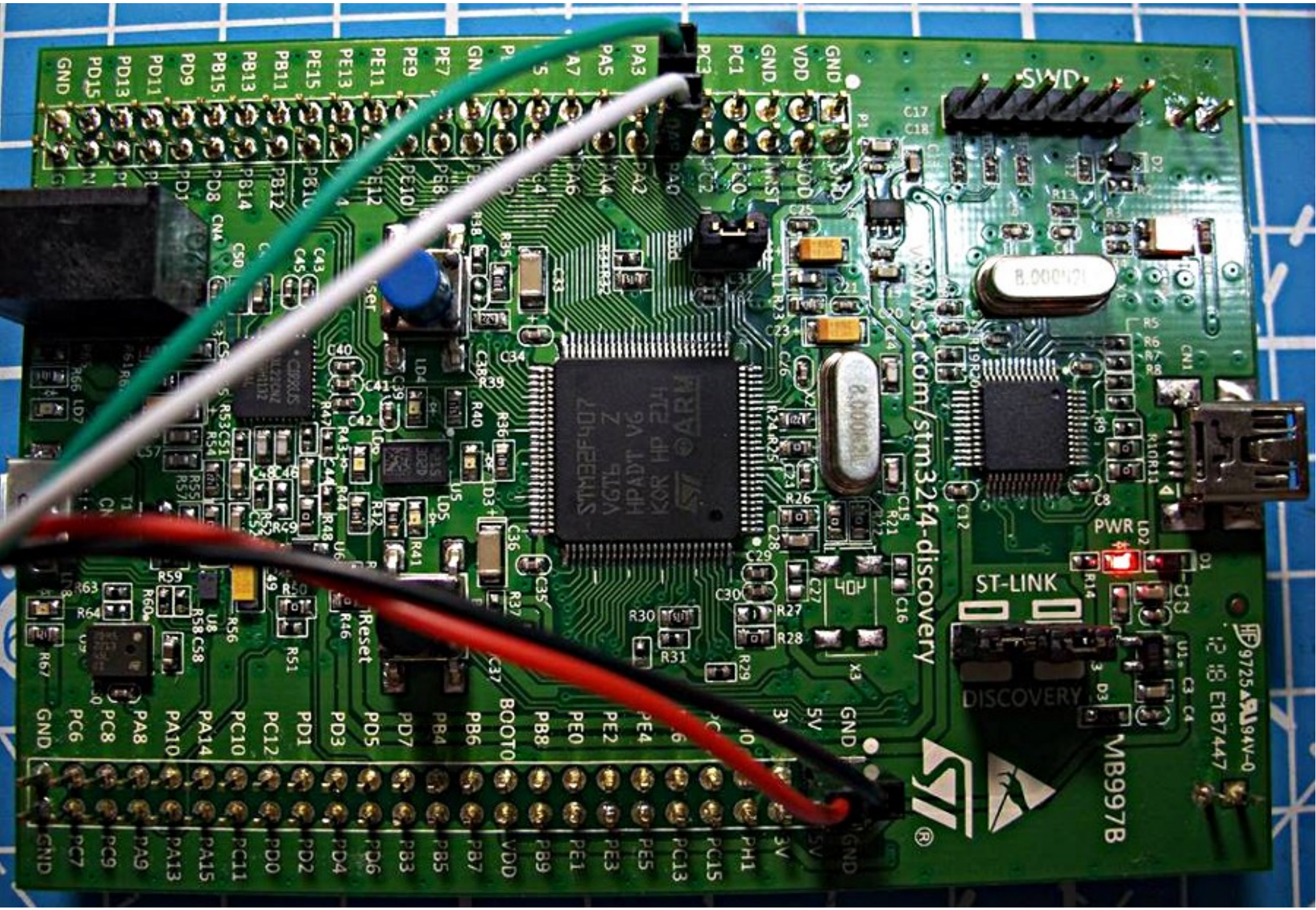
F9 Microkernel 快速回顧

F9 其實不只是個 microkernel...

(a) 一個遵循 L4 microkernel[1] 設計的實作
(b) 針對 ARM Cortex-M[2] 高度優化的系統
(c) 提供 Bluetooth 4.0 / BLE (Bluetooth Low-energy) 通訊功能的系統，並且著墨於整體功耗的改善

[1] https://en.wikipedia.org/wiki/L4_microkernel_family
[2] http://www.arm.com/zh/products/processors/cortex-m/

F9 目前的參考硬體 :STM32F4-Discovery



STM32F4-Discovery 簡要特徵規格

- 物美價廉：USD $20
- ARM Cortex-M4; 168 MHz; 210 DMIPS
- Flash size: 1 MB
- RAM size: 192 KB
- DSP

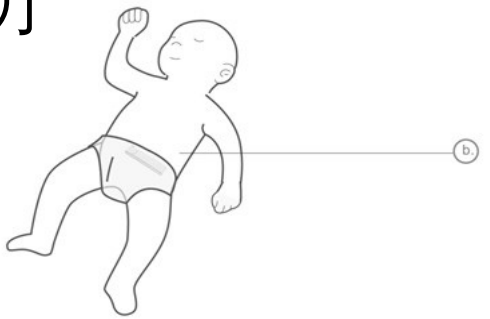
  ...
- 2x 12-bit DAC
- Over 24 12-bit ADC channels
- Up to 17 timers
- USB OTG = Host or Client
- 10/100 Ethernet MAC


kernel/syscall.c
kernel/systhread.c
kernel/thread.c
kernel/lib/fifo.c
kernel/lib/ktable.c
kernel/lib/stdio.c
platform/stm32f4/discoveryf4.c
platform/stm32f4/misc.c
platform/stm32f4/exti.c
platform/stm32f4/gpio.c
platform/stm32f4/rcc.c
platform/stm32f4/usart.c
platform/stm32f4/system.c
platform/debug_uart.c
platform/bitops.c
platform/spinlock.c
platform/mpu.c
user/root_thread.c
LINK out/f9.elf
sat/bin/arm-none-eabi-objcopy -O ihex ./out/f9.elf ./out/f9.hex
sat/bin/arm-none-eabi-objcopy -O binary ./out/f9.elf ./out/f9.bin
sat/bin/arm-none-eabi-objdump -S ./out/f9.elf > ./out/f9.list
en@ubuntu:~/workarea/zplab/f9-kernel-stm32f4$ ls
THORS f9.ld f9_mem.ld include kernel Makefile out platform README.md user
en@ubuntu:~/workarea/zplab/f9-kernel-stm32f4$ cd out/
Memory display
Address: 0x08000000
Size: 0xD000
Device Memory
File : f9.bin
Device Memory
23:58:52 : Connected via SWD.
23:58:52 : Connetion mode : Normal.
23:58:52 : Device ID:0x413
23:58:52 : Device flash Size : 1MBytes
23:58:52 : Device family :STM32F40x/STM32F41x
23:59:53 : [f9.bin] opened successfully.
00:00:33 : Flash memory programmed in 2s and 496ms.
00:01:14 : Disconnected from device.
00:01:14 : Connection to device is lost!
Disconnected
Device ID : ---------
KDB
commands:
K: print kernel tables
e: dump ktimer events
n: show timer (now)
s: show softirqs
t: dump threads
m: dump memory pools
a: dump address spaces
KDB
-------THREADS------
type global local state parent
IDLE 00000000 00000000 RUN 00000000
ROOT 00008000 00000000 SVC 00000000
KERN 00004000 00000000 RUN 00000000
[USR] 00400000 00000040 RUN 00008000
[USR] 00404000 00000080 RECV 00008000
在 Discovery 硬體運作 F9

參考的產品應用:嬰幼兒生理監控

- 藉由偵測寶寶動作或是聲音的變化作為監控依據之產品

- 以電生理訊號中量測心電圖的偵測技術為主軸,藉由無線網路將資訊傳輸到伺服器做分析,而裝置本身也具備即時告警與初步解讀的能力



開放原始碼策略

這符合多種新型應用的需求，沒必要藏私，更該透過社群的力量，使基礎建設變得更好

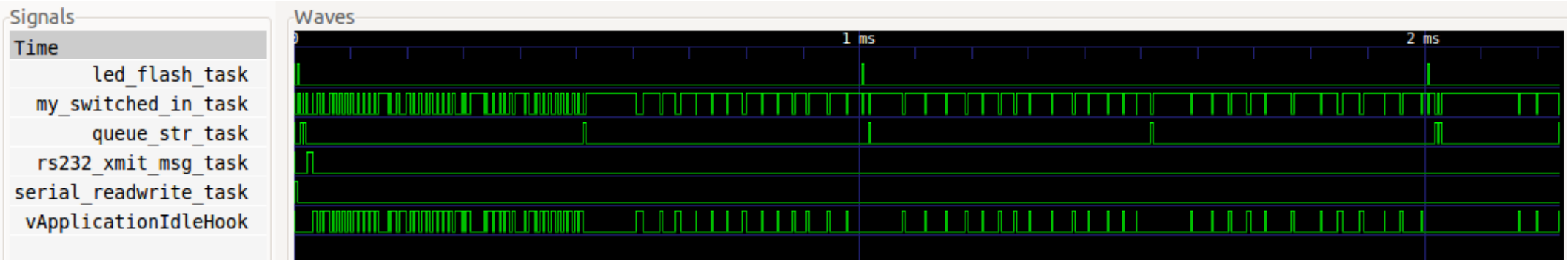
以 BSD 授權釋出 F9 microkernel 的主體及通訊系統，允許在這之上建構商業應用

期許開放原始碼社群參與
（均有基礎建設，只要擴充即可）

(a) 系統視覺化分析工具
(b) 進階電源管理機制
(c) 應用案例 + 文件

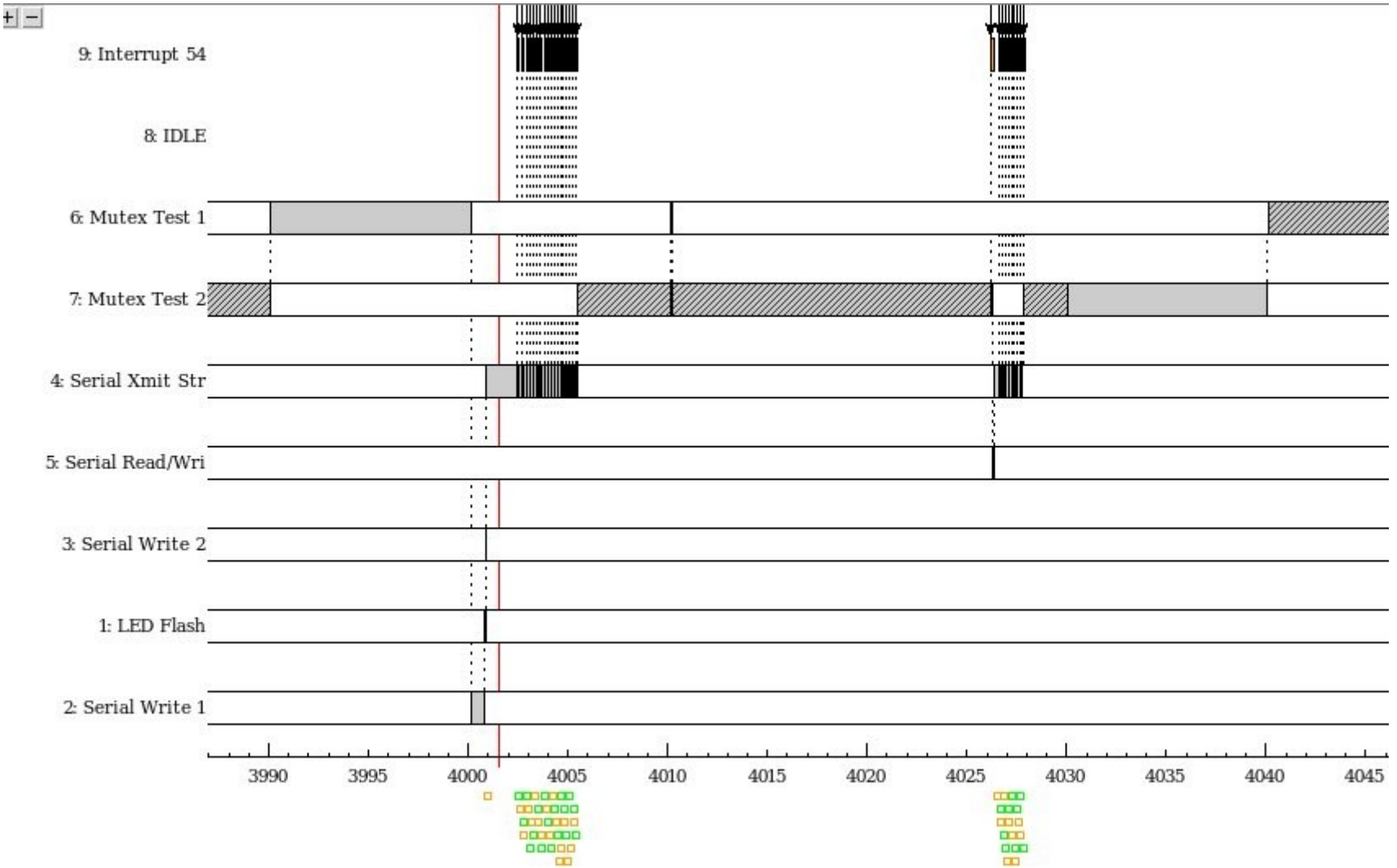
系統視覺化工具

觀察 context switch 的過程
分析 interrupt 發生的時序以及相關的系統處理
有效的收集系統資訊，並即時解析與呈現



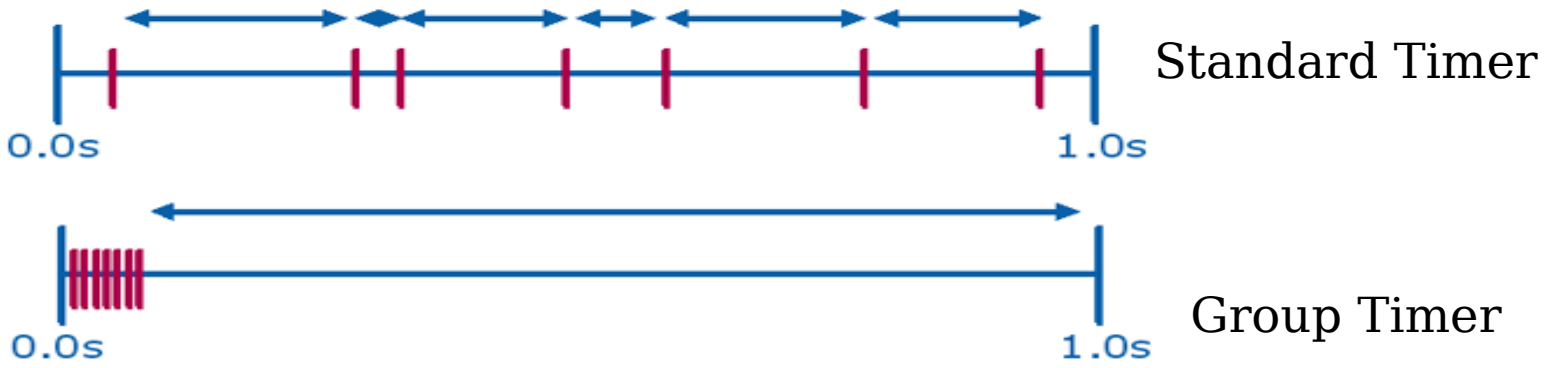
系統視覺化工具

進一步得知 mutex, semaphore, critical section 具體狀況

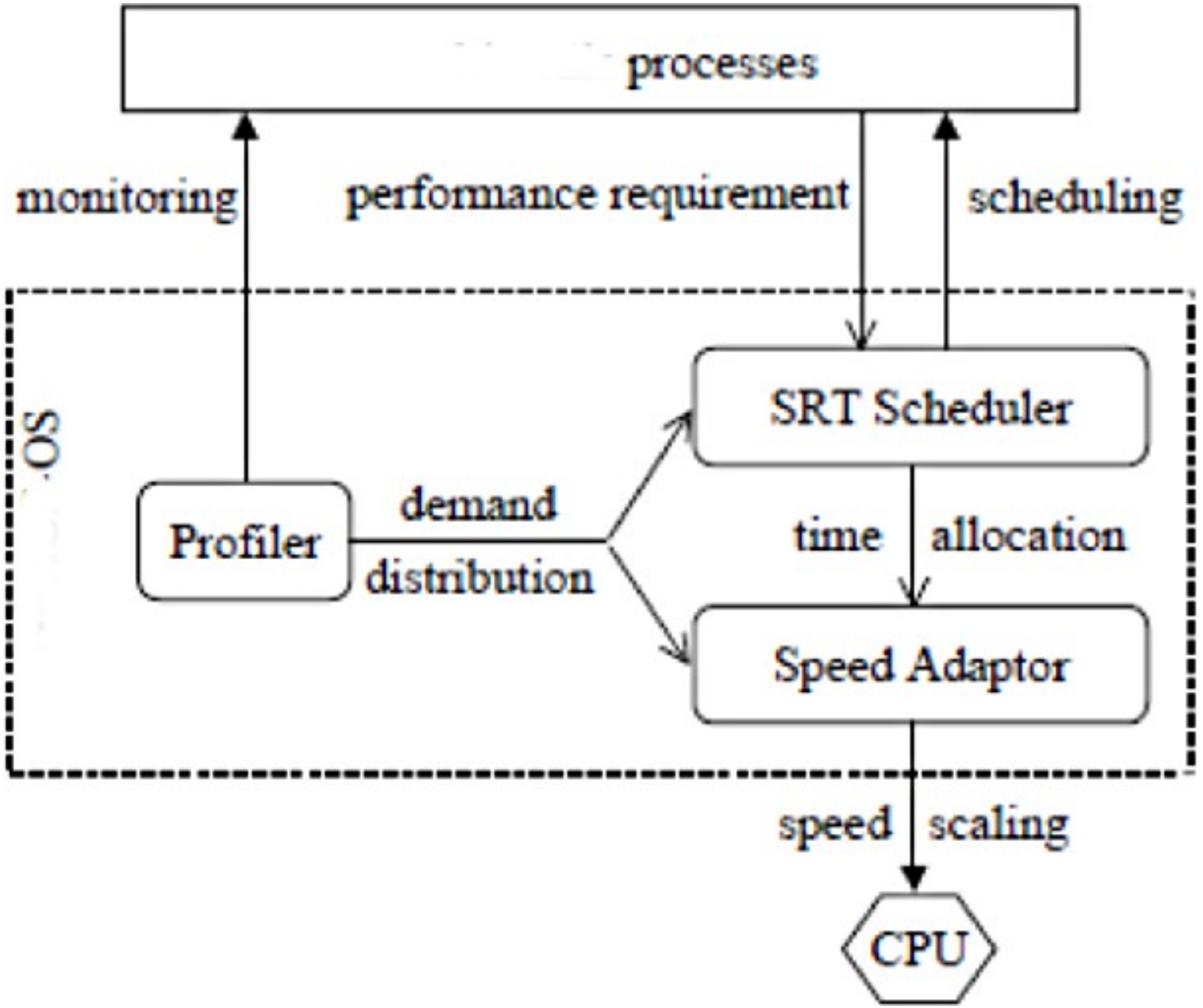


進階電源管理機制

低功耗設計：發揮 ARM Cortex-M 特性
Tickless scheduling
動態 profiling + hotspot 分析



進階電源管理機制


processes
monitoring
performance requirement
scheduling
OS
SRT Scheduler
Profiler
demand
distribution
time
allocation
Speed Adaptor
speed
scaling
CPU

該如何參與？

前期準備

(a) 參閱 L4 microkernel 相關文件

http://www.slideshare.net/jserv/microkernel-evolution

(b) 研讀 BLE 資訊

http://en.wikipedia.org/wiki/Bluetooth_low_energy

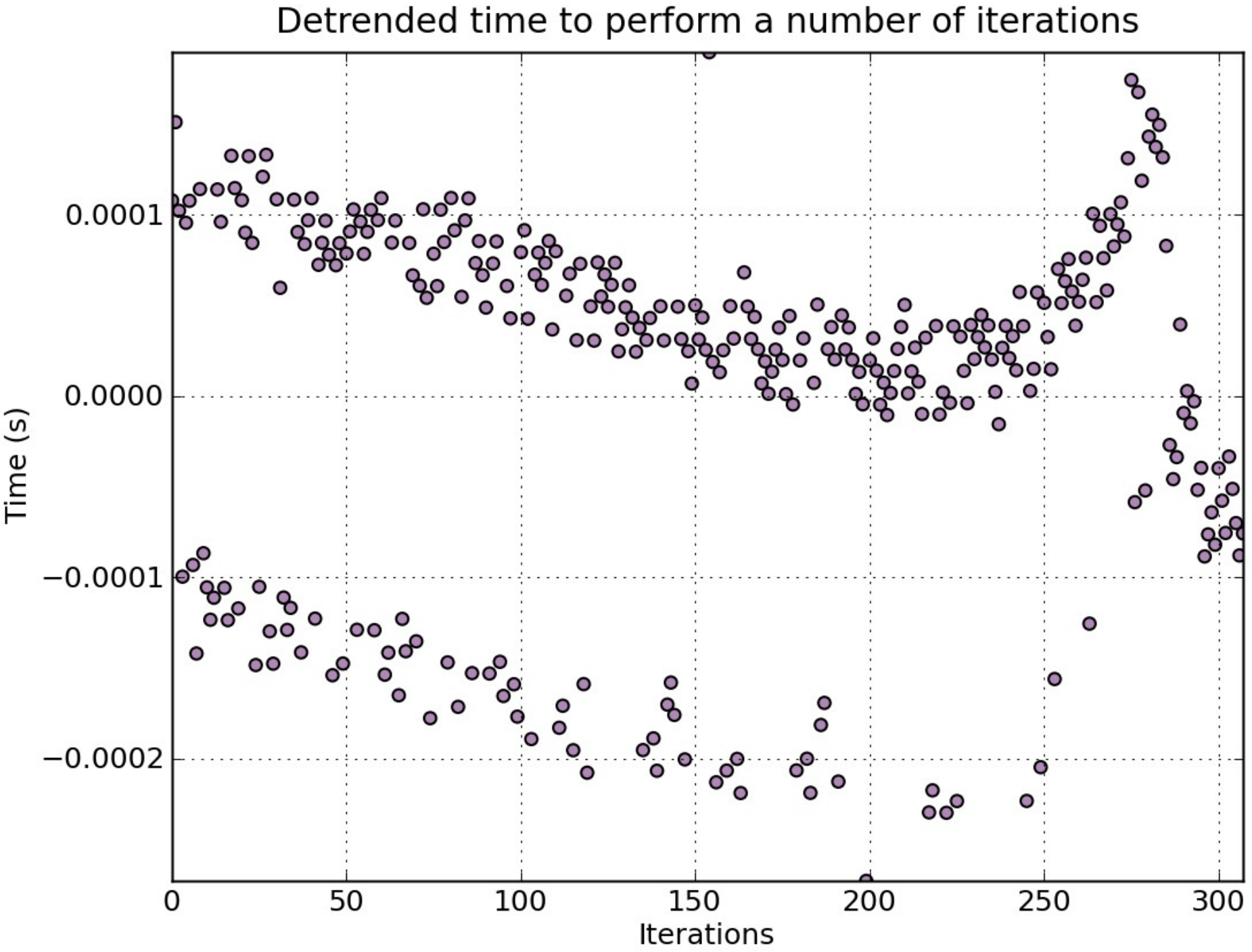
(c) 參考「嵌入式系統開發」的開放教材

http://wiki.csie.ncku.edu.tw/embedded/schedule

工作項目的切入點

(a) 以 Web 技術改寫所有的工具展現方式
(b) 提供與其他 RTOS 的 API 相容能力
(c) 撰寫技術文件，分析 microkernel ，並由具體而微的設計，去理解相關的原理
(d) 改善效能、功能、穩定度

預期的工作輸出：
F9 microkernel 在實際硬體表現的統計


Detrended time to perform a number of iterations
Time (s)
0.0001
0.0000
−0.0001
−0.0002
0
50
100
150
200
250
300
Iterations

Idle

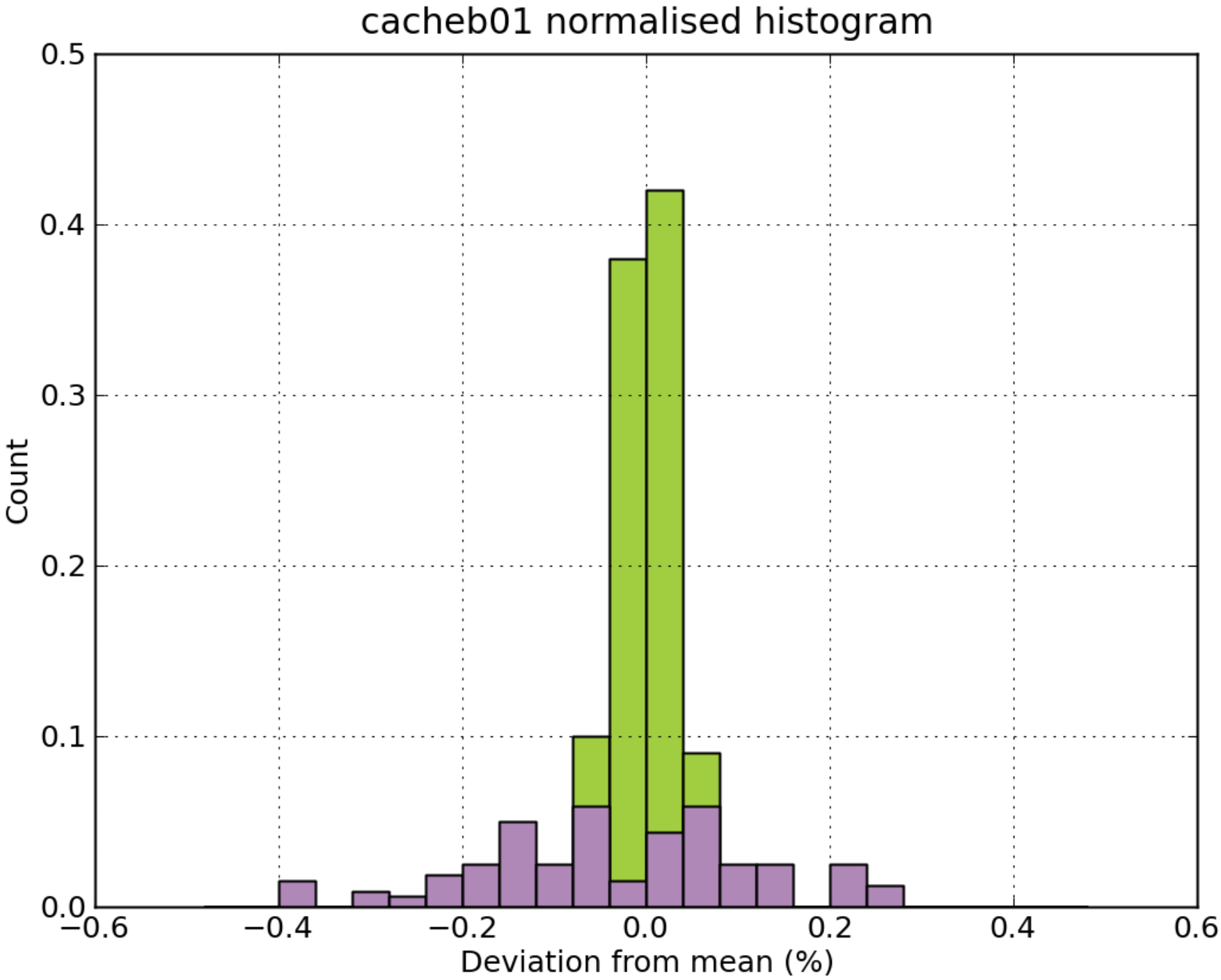
	Wall mean	Mean	Stddev	Mean vs plain	Stddev vs plain
Plain	4.993	4.993	0.000353		
High	4.993	4.993	0.000190	100.001%	53.875%
Nice	4.993	4.993	0.000255	99.999%	72.225%

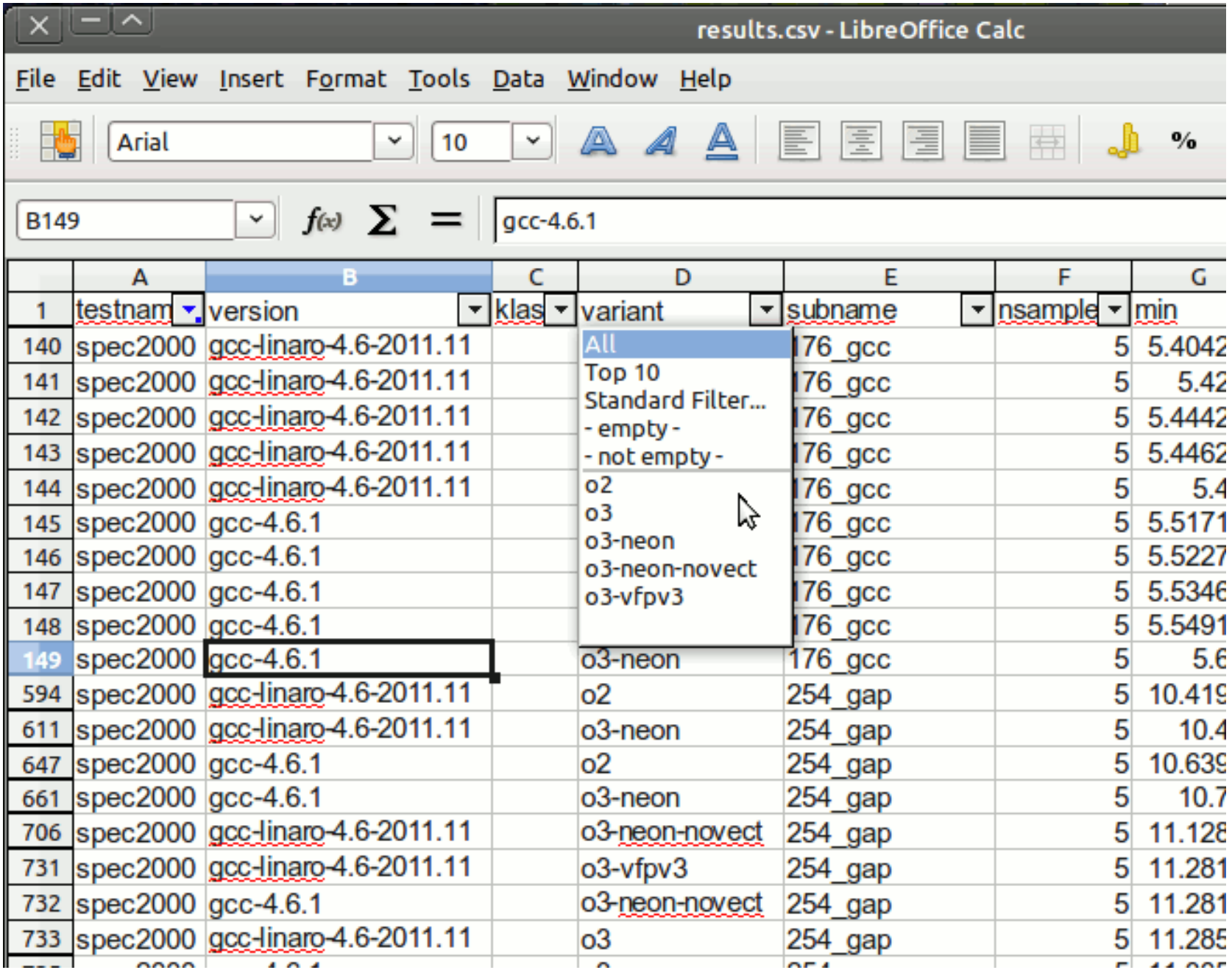
CPU load

	Wall mean	Mean	Stddev	Mean vs plain	Stddev vs plain
Plain	14.966	4.992	0.007007		
High	5.096	4.992	0.004316	99.989%	61.589%
Nice	65.311	4.991	0.005056	99.971%	72.149%

Network load

	Wall mean	Mean	Stddev	Mean vs plain	Stddev vs plain
Plain	5.134	5.099	0.005952		
High	5.112	5.099	0.005798	100.009%	97.397%
Nice	5.134	5.099	0.005952	100.000%	100.000%


cacheb01 normalised histogram
0.5
0.4
0.3
0.2
0.1
0.0
Count
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
Deviation from mean (%)



	A	B	C	D	E	F	G
1	testnam	version	klas	variant	subname	nsample	min
140	spec2000	gcc-linaro-4.6-2011.11			176_gcc	5	5.4042
141	spec2000	gcc-linaro-4.6-2011.11			176_gcc	5	5.42
142	spec2000	gcc-linaro-4.6-2011.11			176_gcc	5	5.4442
143	spec2000	gcc-linaro-4.6-2011.11			176_gcc	5	5.4462
144	spec2000	gcc-linaro-4.6-2011.11			176_gcc	5	5.4
145	spec2000	gcc-4.6.1			176_gcc	5	5.5171
146	spec2000	gcc-4.6.1			176_gcc	5	5.5227
147	spec2000	gcc-4.6.1			176_gcc	5	5.5346
148	spec2000	gcc-4.6.1			176_gcc	5	5.5491
149	spec2000	gcc-4.6.1		o3-neon	176_gcc	5	5.6
594	spec2000	gcc-linaro-4.6-2011.11		o2	254_gap	5	10.419
611	spec2000	gcc-linaro-4.6-2011.11		o3-neon	254_gap	5	10.4
647	spec2000	gcc-4.6.1		o2	254_gap	5	10.639
661	spec2000	gcc-4.6.1		o3-neon	254_gap	5	10.7
706	spec2000	gcc-linaro-4.6-2011.11		o3-neon-novect	254_gap	5	11.128
731	spec2000	gcc-linaro-4.6-2011.11		o3-vfpv3	254_gap	5	11.281
732	spec2000	gcc-4.6.1		o3-neon-novect	254_gap	5	11.281
733	spec2000	gcc-linaro-4.6-2011.11		o3	254_gap	5	11.285

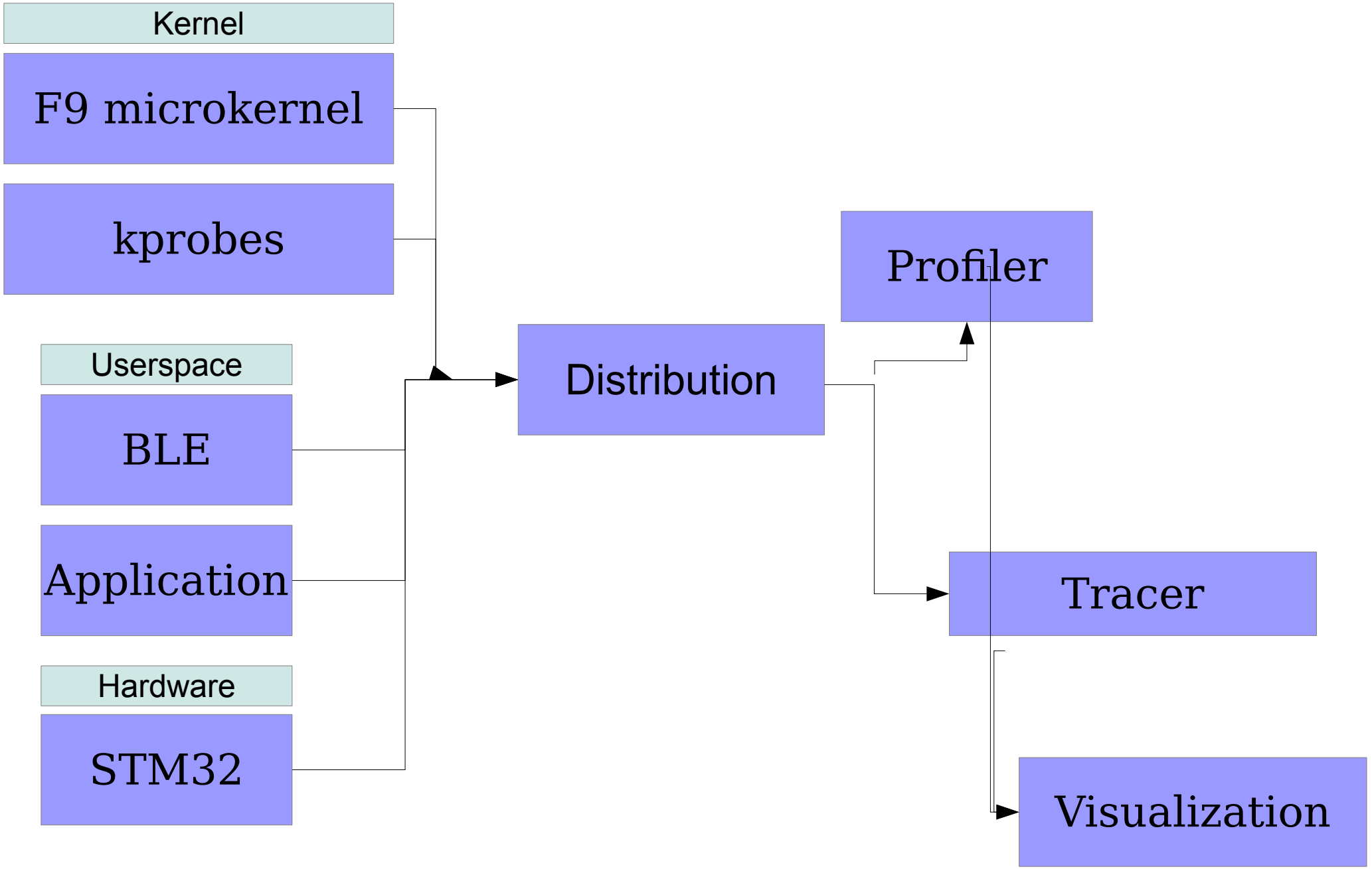
http://help.libreoffice.org/Calc/Applying_AutoFilter

預期的工作輸出：
整合既有的 open source 工具

```
michaelh@leo1:~/coremark_v1.0$ perf report -n -d coremark.exe --stdio
# dso: coremark.exe
# Events: 15K cycles
#
# Overhead  Samples         Command                  Symbol
# ........ ..........  ............  ....................
#
    37.96%       5980  coremark.exe  [.] core_state_transition
    27.21%       4197  coremark.exe  [.] core_bench_list
    16.25%       2522  coremark.exe  [.] matrix_test
     7.04%       1110  coremark.exe  [.] crcu32
     6.50%        931  coremark.exe  [.] crc16
     2.42%        396  coremark.exe  [.] core_bench_state
     1.70%        279  coremark.exe  [.] crcu16
     0.87%        143  coremark.exe  [.] calc_func
     0.04%          6  coremark.exe  [.] core_bench_matrix
     0.02%          3  coremark.exe  [.] main


#
# (For a higher level overview, try: perf report --sort comm,dso)
#
```

預期的工作輸出：
健全的系統設計


Kernel
F9 microkernel
kprobes
Userspace
BLE
Application
Hardware
STM32
Distribution
Profiler
Tracer
Visualization


We Want
YOU!