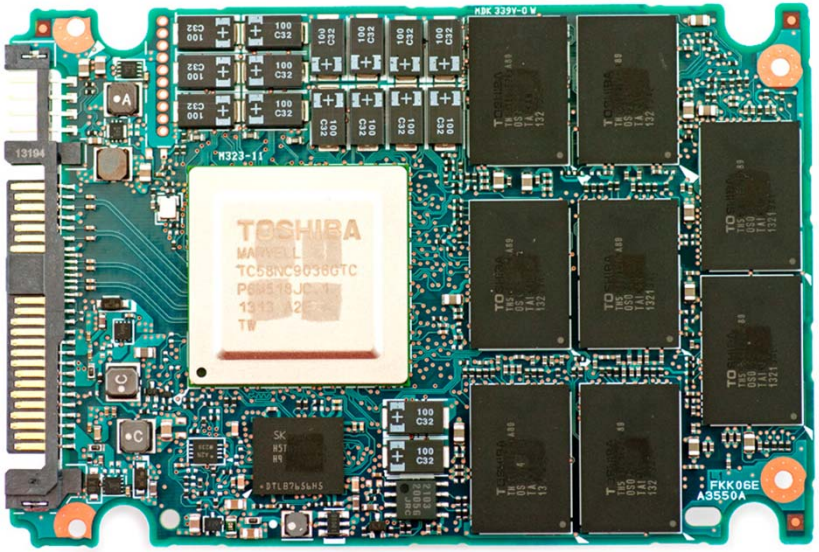
國立中正大學 資工 OSLab 羅習五
shiwulo@gmail.com

中正快速開機技術

HD

Flash drive



Flash drive在IOPS上大幅領先HHD

- HHD
- 3TB約3000~4000元
- 讀寫：150MB/s
- IOPS：100~200

- Flash drive
- 0.48TB約13000元
  - 約3.5X
- 讀寫：550/470
  - 約4.5X
- IOPS：75000/36000
  - 約750X/360X

我們所研發的快速開機技術是基於flash drive有良好的IOPS（隨機存取）

議題

- 展示及快速開機的特色 (11)
- 現行方法的缺失 (6)
- 簡介中正快速開機方法 (8)
- 相關技術及專利情況(11)
- 實做細節及成果分析 (14)

展示及快速開機的特色

平台一：硬體規格

SAMSUNG

Samsung Galaxy II

「13」倍快的處理器
「21」倍快的記憶體
「4」倍快的資料庫讀寫
「21」秒開機

21秒開機

altek

國立中正大學
National Chung Cheng University

Altek Leo

一般的處理器
一般的記憶體
一般的資料庫讀寫
52秒開機加速至「12」秒

12秒開機

較三星快一倍

更少的硬體需求，更快的開機效能

平台二：Beagleboard規格

- 作業系統為Android 2.2
- 記憶體使用512MB
- 處理器為ARM Cortex A8
- 使用class 4的SD Card

平台三

- 作業系統為 Android 4.2.2
- 記憶體使用 512MB
- 處理器為 Sitara AM3358 (ARM Cortex-A8)
- 使用 Class 10 的SD Card

已開發的軟硬體平台

- 硬體
  - ARM 11(Altek Leo)、ARM Cortex A8 (beagle board)、ARM Cortex A9 (panda board)
  - x86、x86-64 (Thinkpad x200)
- 軟體：
  - GNU/Linux（ Ubuntu 10.04 ）
  - Android 2.2 ~ Android 4.1（ panda board ）

現行方法的缺失

現在市面上有二種方法

- 零耗電開機，但慢
- 耗電型待機，但不符合法令及需求

以Sony BD player 試算耗電

	運轉耗電	待機耗電	開機時間
Sony	**12.87**	**6.72w**	**2s97**

假設一天觀看BD影片「2個小時」

使用耗電：12.87瓦*2＝ 25.74瓦

待機耗電：6.72瓦*22＝147.84瓦

換句話說，$\frac{4}{5}$ 的能源浪費在「待機」

12

中華電信MOD
也有相同問題

開機時的耗電量（12瓦）

關機時的耗電量（11瓦）

中華電信MOD
也有相同問題

假設一天觀看MOD影片「2個小時」

使用耗電：12瓦*2＝ 24瓦

待機耗電：11瓦*22＝242瓦

換句話說，$\frac{9}{10}$ 的能源浪費在「待機」

中華電信MOD
『每個月浪費的』電費約

非營業用	120度以下部分	2.10	15.25元/月
	121-330度部分	3.02	21.93元/月
	331-500度部分	4.39	31.87元/月
	501-700度部分	4.97	36.08元/月
	701度以上部分	5.63	40.87元/月
營業用	330度以下部分	3.76	27.3元/月
	331-700度部分	4.62	33.54元/月
	701-1500度部分	5.48	39.78元/月
	1501度以上部分	5.92	42.98元/月

其他技術

智慧型電視（聯發科）「零耗電」快速開機12秒 耗電型快速開機2秒

- 目前的 Android 智慧型電視開機時間約 25 秒，
- 聯發科提供電視廠商客戶「冷開機」與「暖開機」解決方案。
  - 在冷開機模式下，可利用針對Android程式碼的軟體最佳化，將開機時間縮短至 12 秒；
  - 同時藉由在DRAM內建立的待命「暫停(suspend)」模式，其暖開機方案能將開機時間大幅減少為 2 秒。

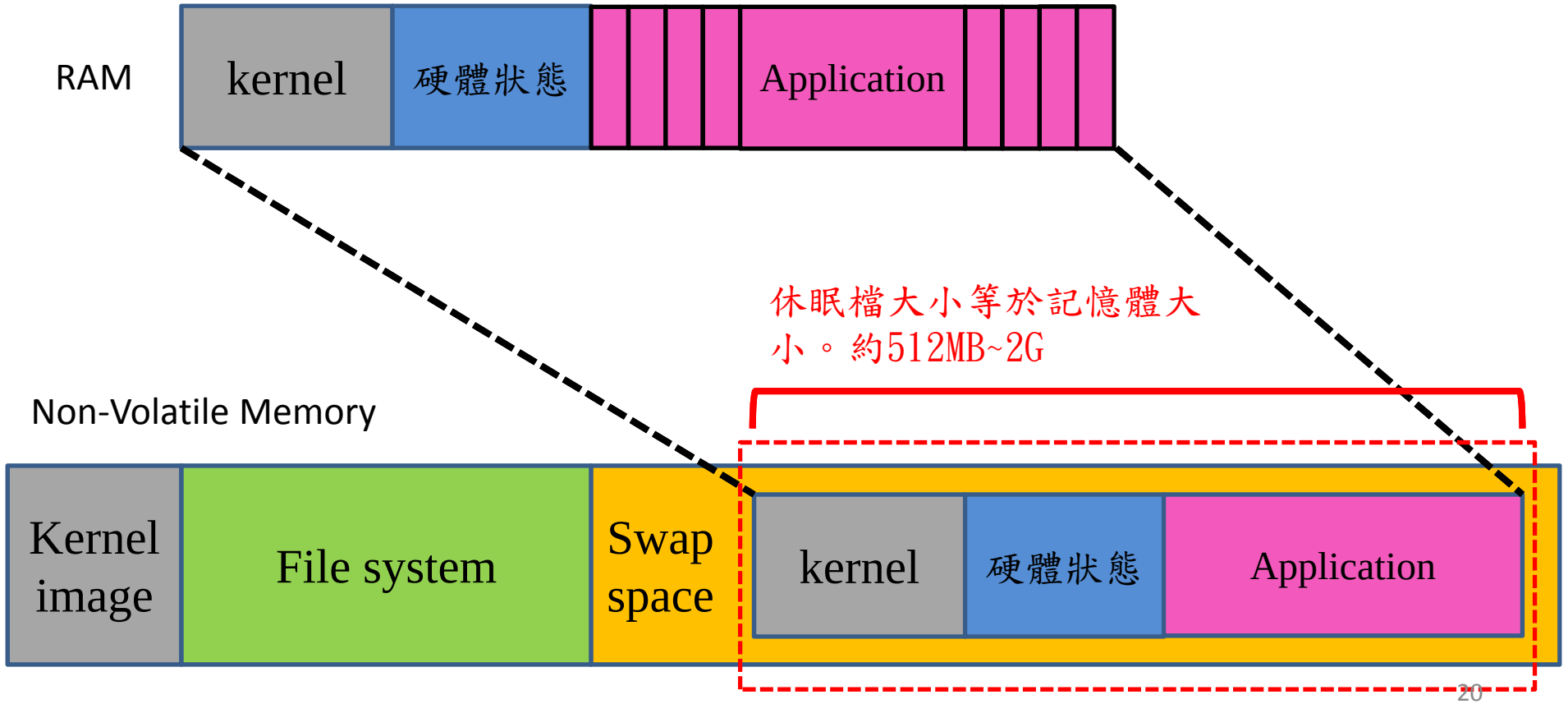
資料來源：http://www.eettaiwan.com/ART_8800681267_622964_NT_f5698935.HTM

車用多媒體導航（內建電池）
「零耗電」，約25秒
電池用盡前，約6秒

- 怡利 Multi-media Navigation System 多媒體導航系統
- 系統參數
  - 1 適用電壓： DC 10V~ 16V 。
  - ACC 關閉模式，電源消耗： 24mW 以下 (12V ， 2mA) 。
  - 電源關閉模式： 12W (12V ， 1A)
  - 最大功率消耗： 45W×4 。
- 本系統一般正常使用為暖開機，開機只需要 4-8 秒，但在下列特殊情況會進入冷開機，開機需要20-30 秒
  - 本系統剛裝車完畢第一次開機為冷開機，開機需要較長的時間 (約20-30 秒) ，之後日常使用為暖開機。
  - 但若長時間未使用（超過3 天以上)，超過三天之後第一次開機使用，即為冷開機。

簡介中正快速開機方法

傳統休眠機制


RAM
kernel
硬體狀態
Application
休眠檔大小等於記憶體大小。約512MB~2G
Non-Volatile Memory
Kernel image
File system
Swap space
kernel
硬體狀態
Application

休眠式快速開機

使用特性：

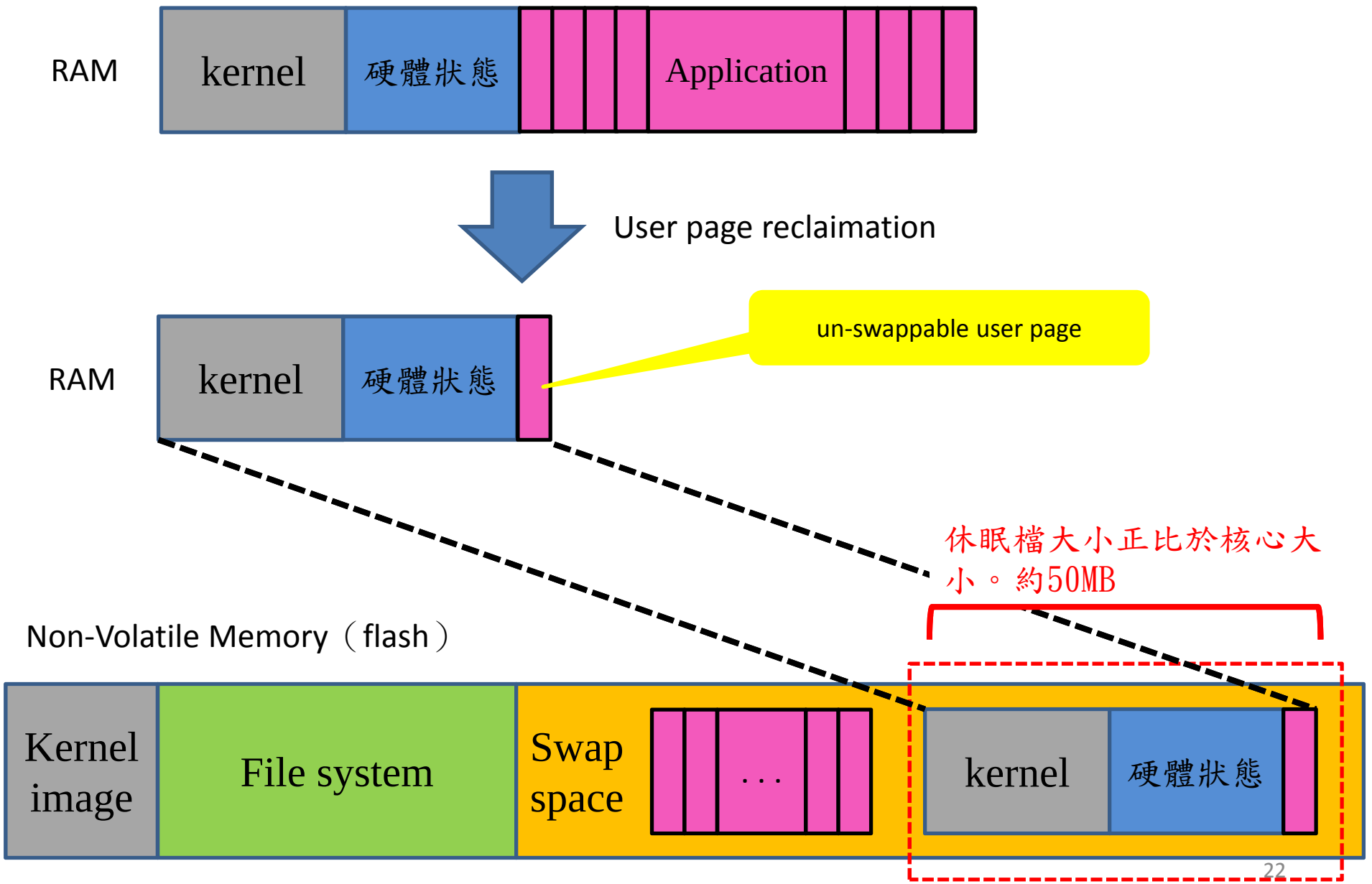
1. 系統的工作記憶體(working set)遠小於系統總記憶體
2. 快閃儲存裝置隨機存取時間接近循序存取時間(假設 rand : seq = 1 : 1)

目的：

最大限度的縮小休眠檔

動作：

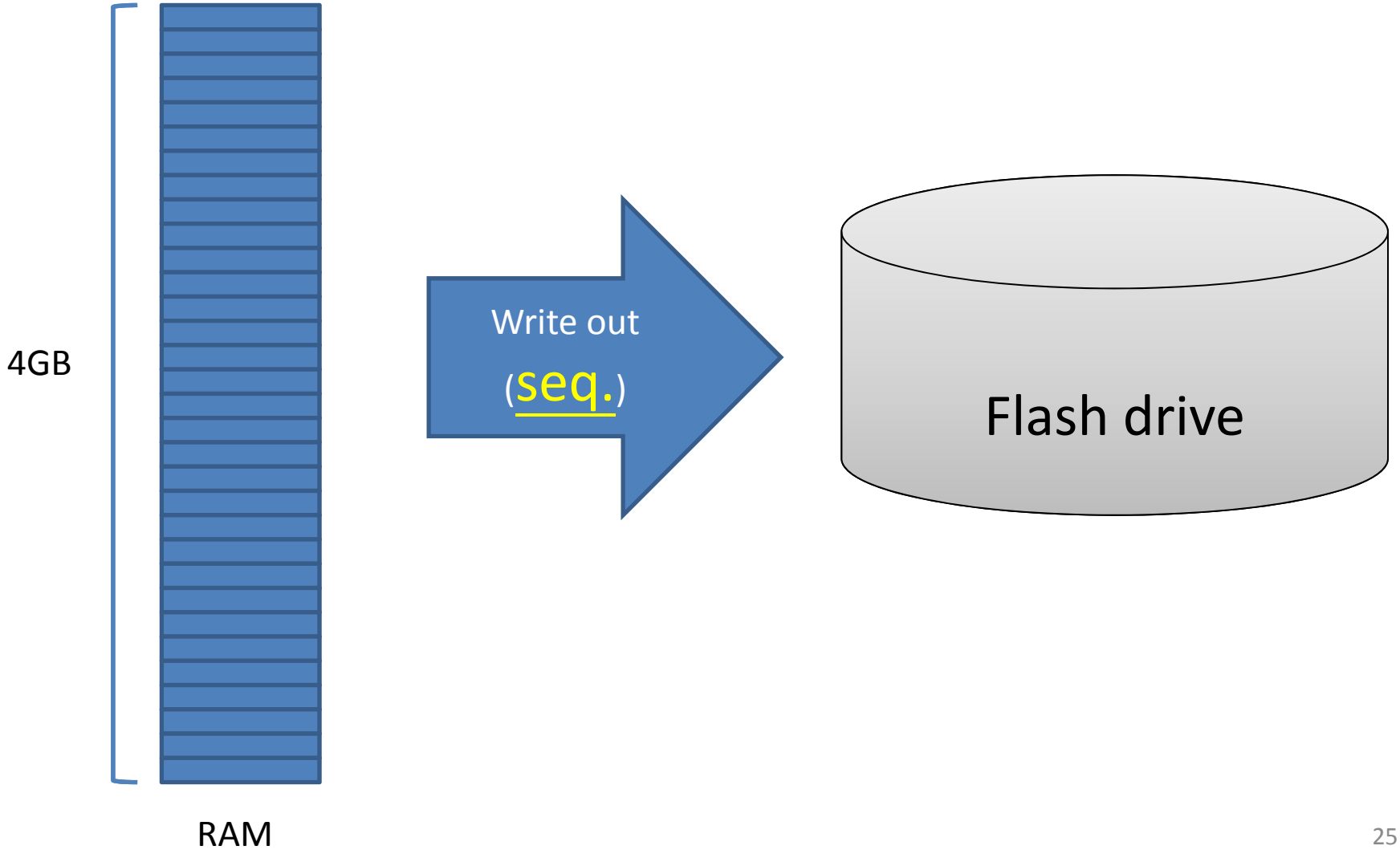
休眠時大量回收user space的記憶體


RAM
kernel
硬體狀態
Application
User page reclaimation
RAM
kernel
硬體狀態
un-swappable user page
休眠檔大小正比於核心大小。約50MB
Non-Volatile Memory（flash）
Kernel image
File system
Swap space
. . .
kernel
硬體狀態

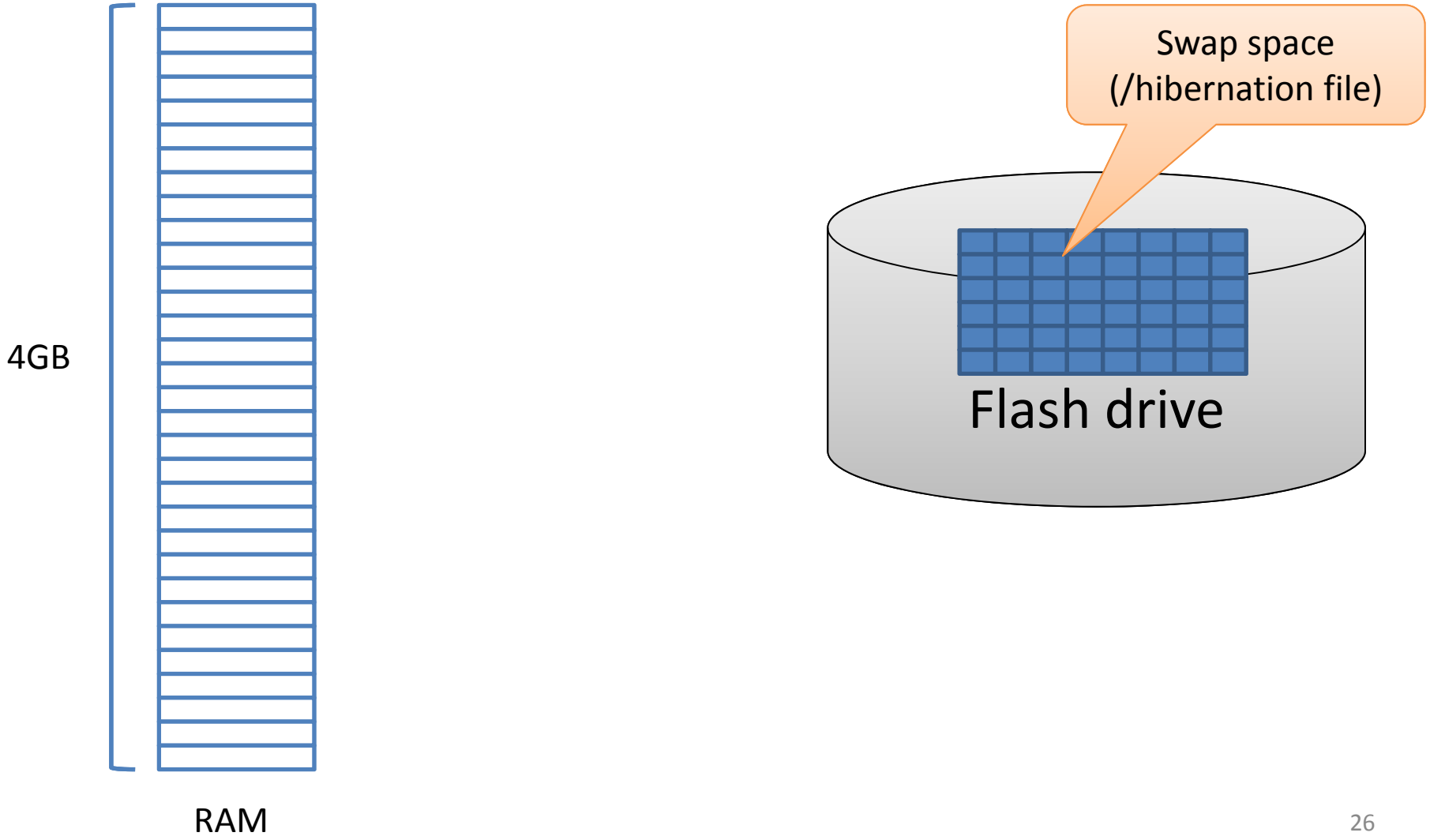
技術概念

先討論傳統休眠機制
之
關機流程

現有的休眠機制 (關機)

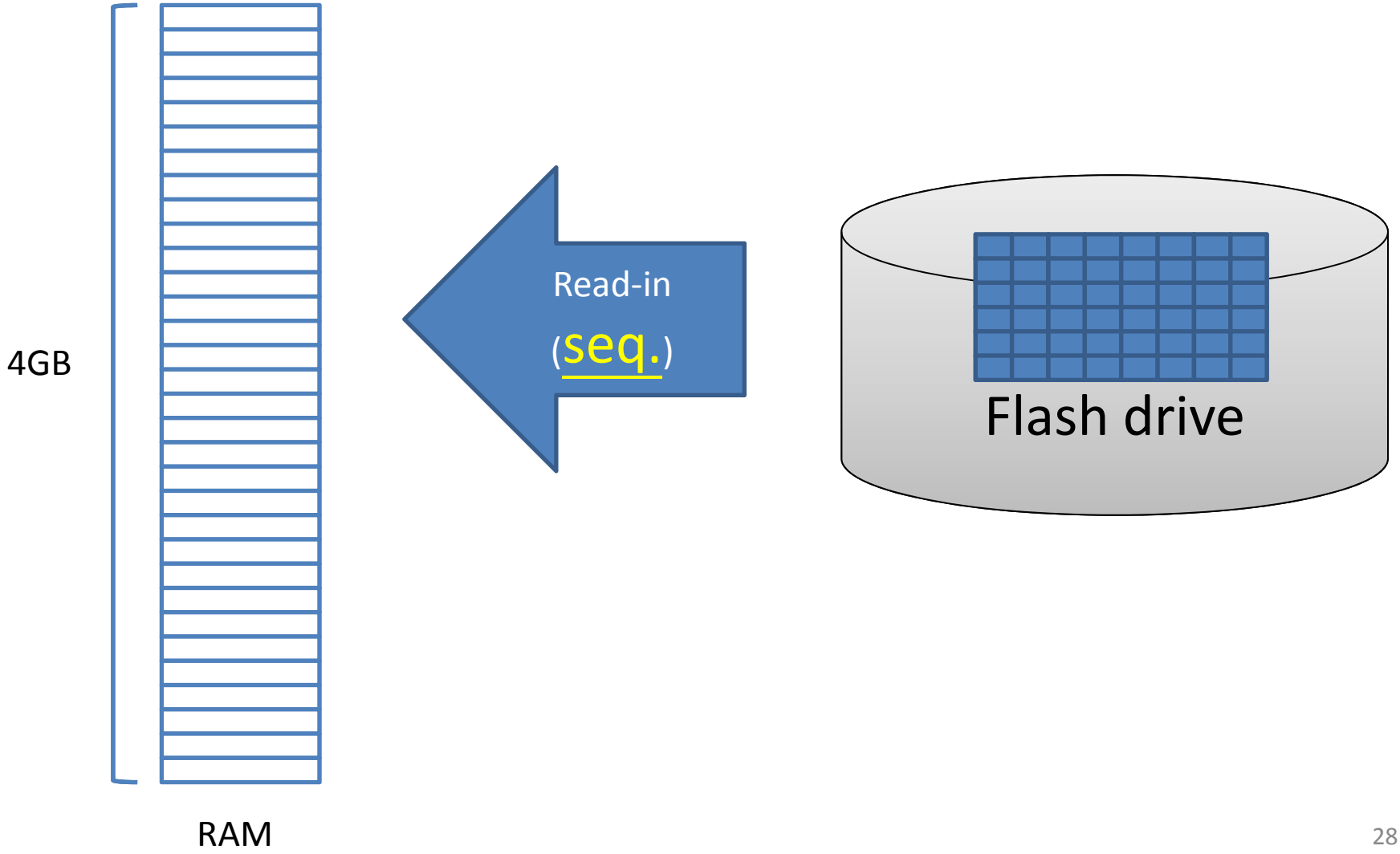


現有的休眠機制 (關機)

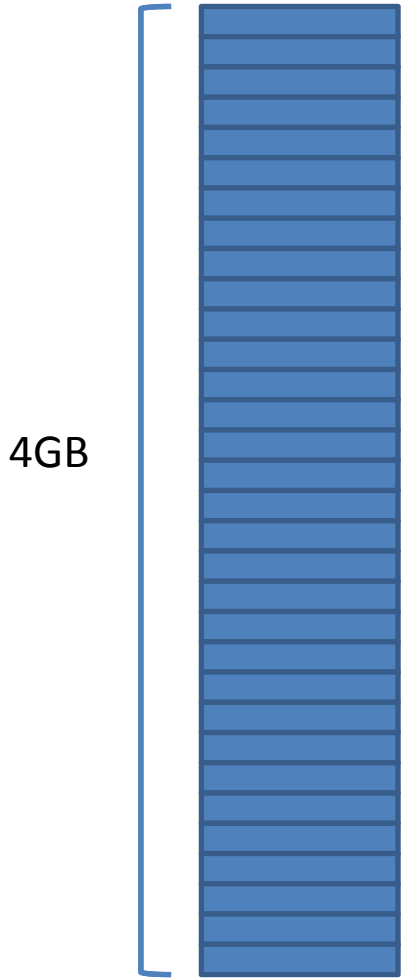


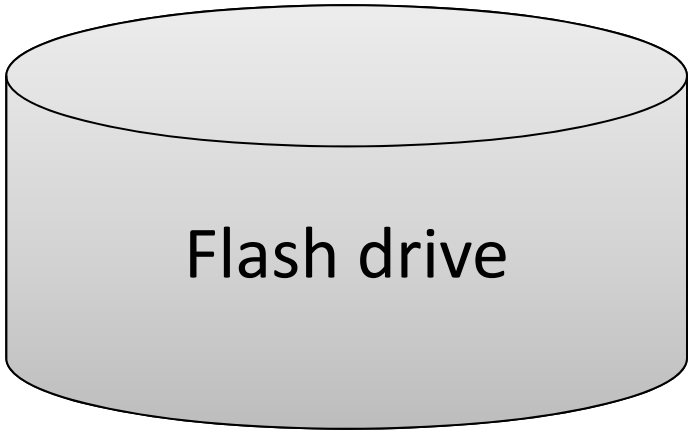
傳統休眠機制
之
開機流程

現有的休眠機制 (開機)



現有的休眠機制 (開機)





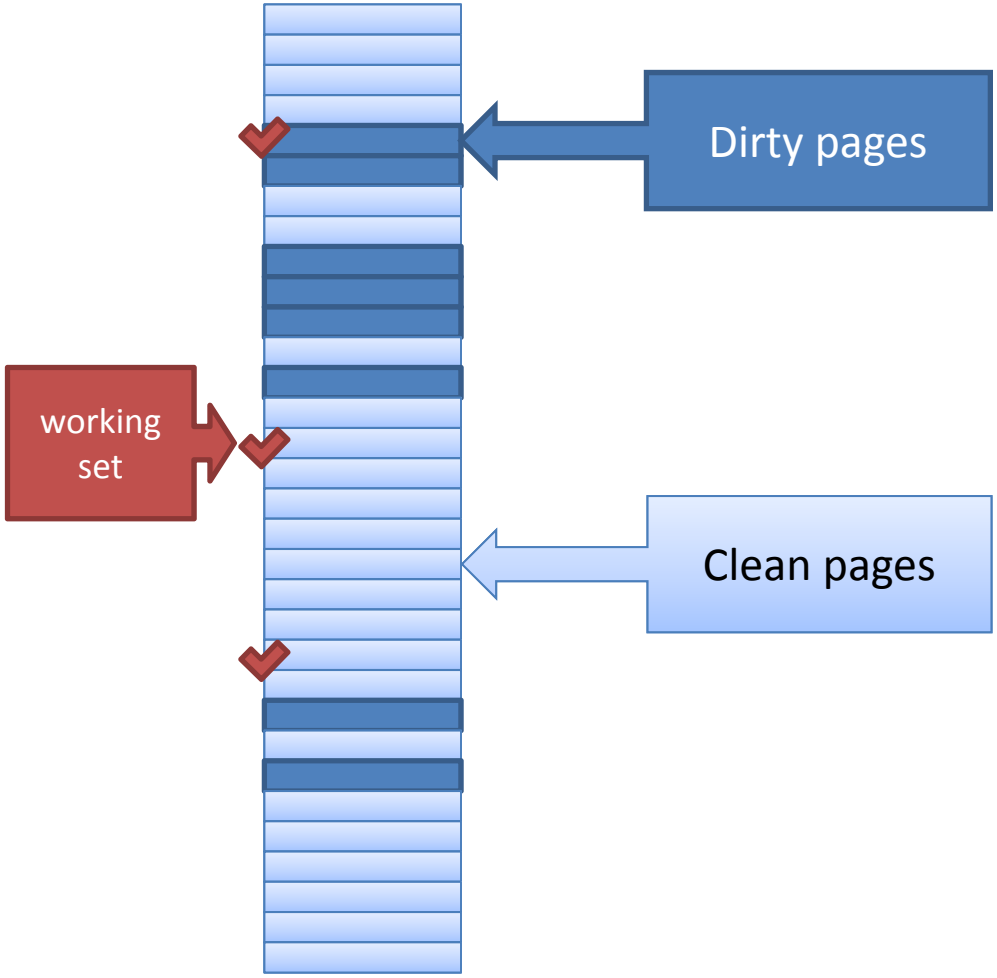
傳統方法開關機時間

傳統方法需要載入全部的記憶體資料，假設flash drive的讀取速度為20M/sec，則主記憶體大小對應的載入時間為（不包含裝置初始化時間）

- 4G：4096M/20M=204.8 sec
- 3G：3072M/20M=153.6 sec
- 2G：2048M/20M=102.4 sec
- 1G：1024M/20M=51.4 sec
- 0.5G：512M/20M=25.7 sec

開始介紹fastBooting
（先介紹工作原理）

分析：記憶體使用情況



1. Most of the memory status is unmodified (clean), with an exact copy in secondary storage.
2. The size of working set is very very small.

Working set

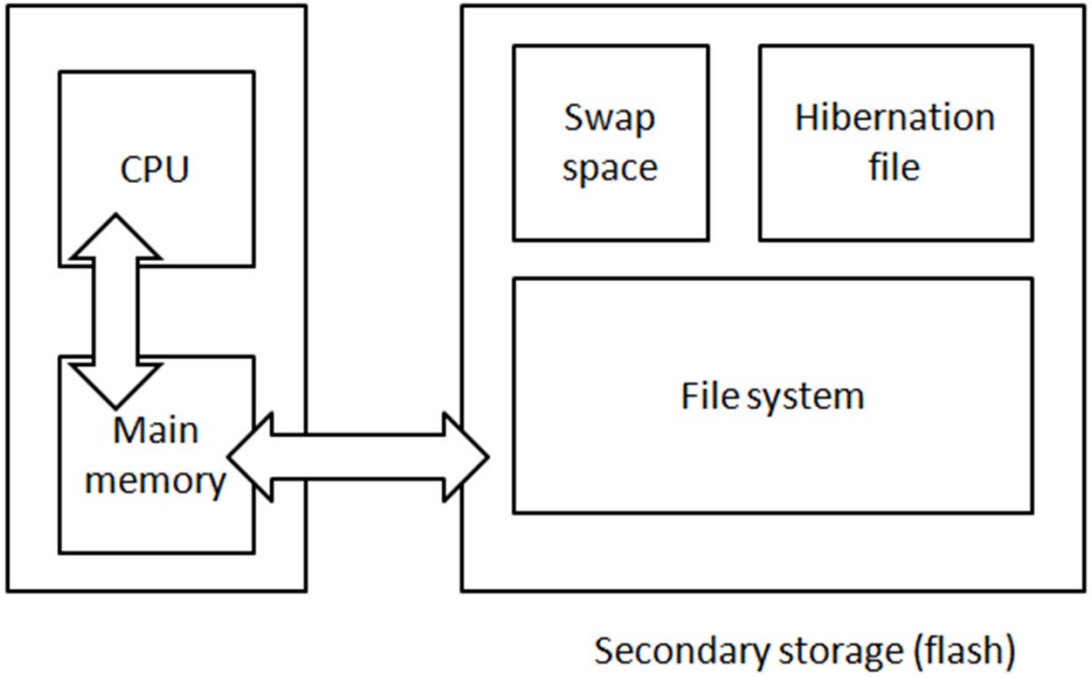
- 一個程式(application)在執行時於某一段時間，所存取的所有記憶體所乘的集合。
- 換句話說，只要載入working set，這個應用程式可以很流暢的執行
- 根據我們的實驗working set的大小約20-30MB

Linux kernel內部是否紀錄working set

- Linux kernel內部使用LRU（Least Recently Used）紀錄
  - 對於每一種「page type」使用2個LRU list紀錄，分別稱之為active list及inactive list，其中active list「被視為」working set
  - 使用2個LRU list的原因是：避免串流資料干擾了working set的正確性
  - 每個LRU list使用的是second chance algorithm
- 可以將working set記錄在hiberfile中

fastBooting
之
關機流程

System architecture

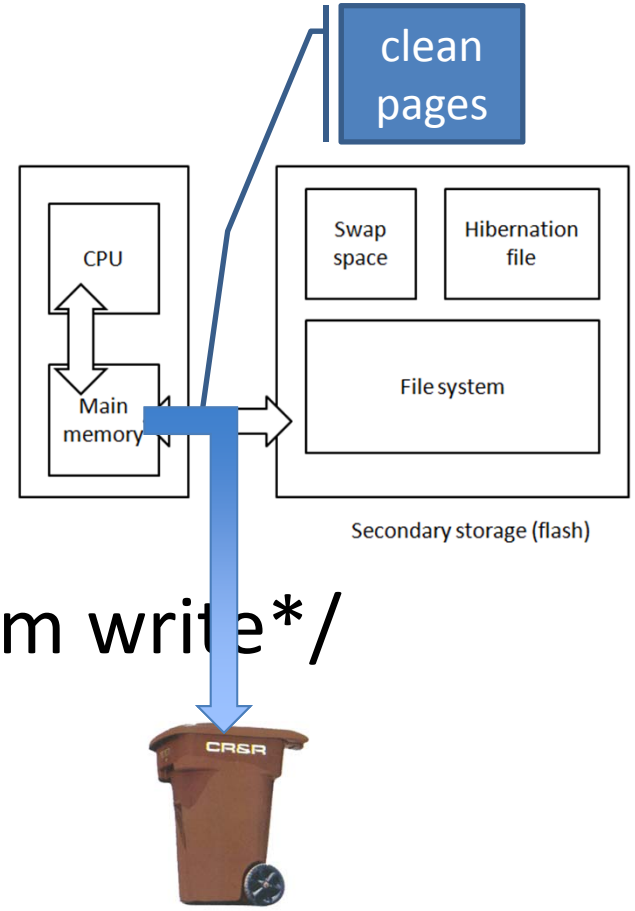


Our algorithm (w/o write-combing)

```
while (has_swappabe_pages())
{
    page = swapout();
    page_set = join(set, page);

    write(page_set);        /*random write*/
}
suspend();
```

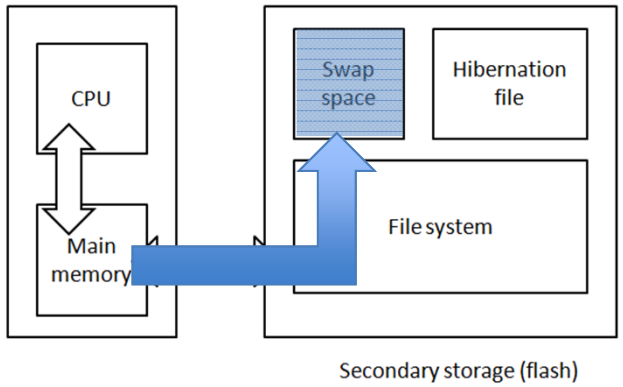
Our algorithm (w/o write-combing)

```
while (has_swappabe_pages())
{
    page = swapout();
    page_set = join(set, page);

    write(page_set);        /*random write*/
}
suspend();
```



Our algorithm (w/o write-combing)

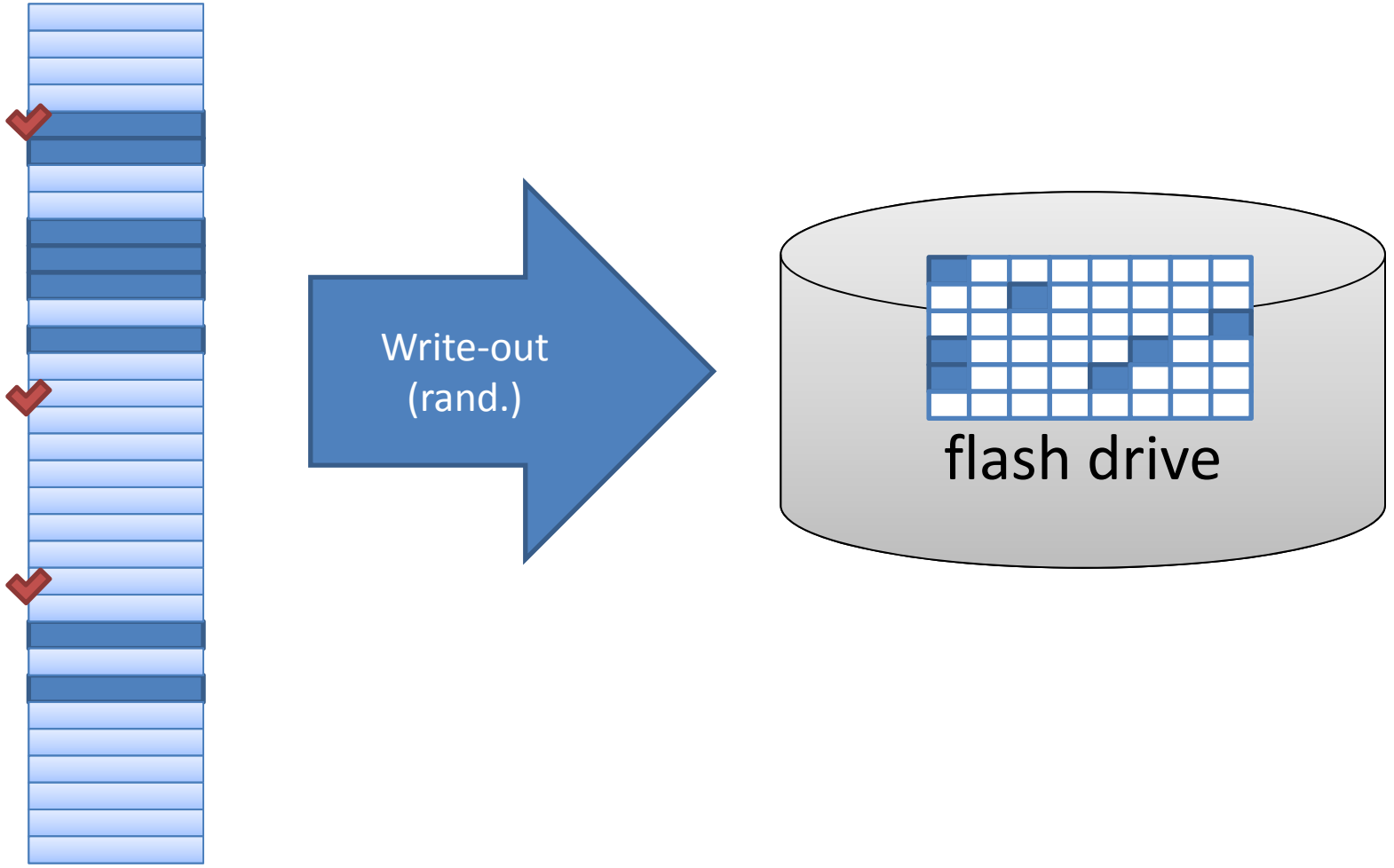
```
while (has_swappabe_pages())
{
    page = swapout();
    page_set = join(set, page);

    write(page_set);        /*random write*/
}
suspend();
```



fastBooting（關機）
swap-out()

```
1.  while(1) {
2.          swappable_page = get_swappable_page();
3.          switch( page_type(swappable_page))
4.          {
5.                  case  “clean page”  :
6.                     /*do nothing*/
7.                     break;
8.                  case  “dirty page”
9.                     return swappable_page;
10.         }
11.}
```

fastBooting（關機）

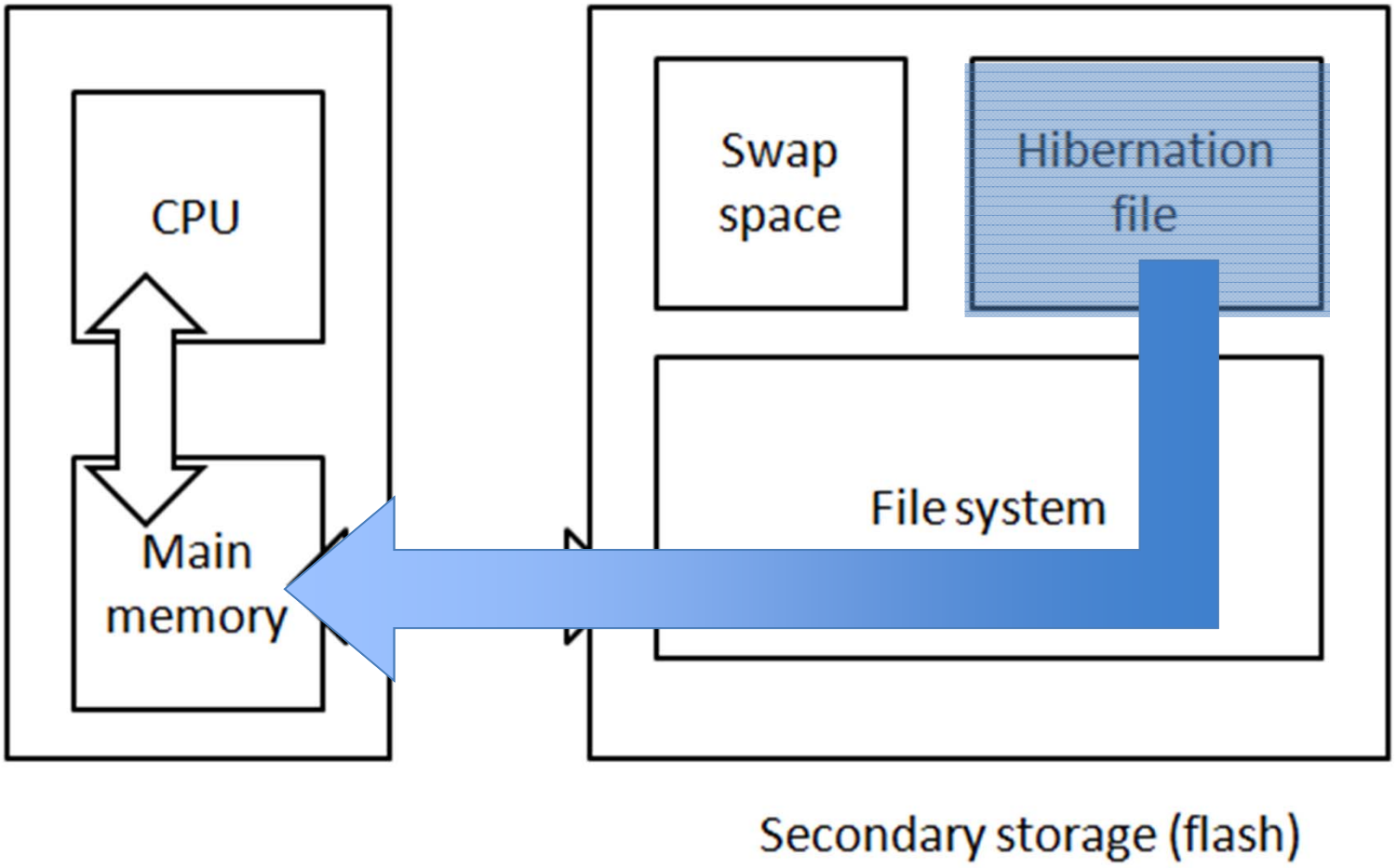


fastBooting
之
開機流程

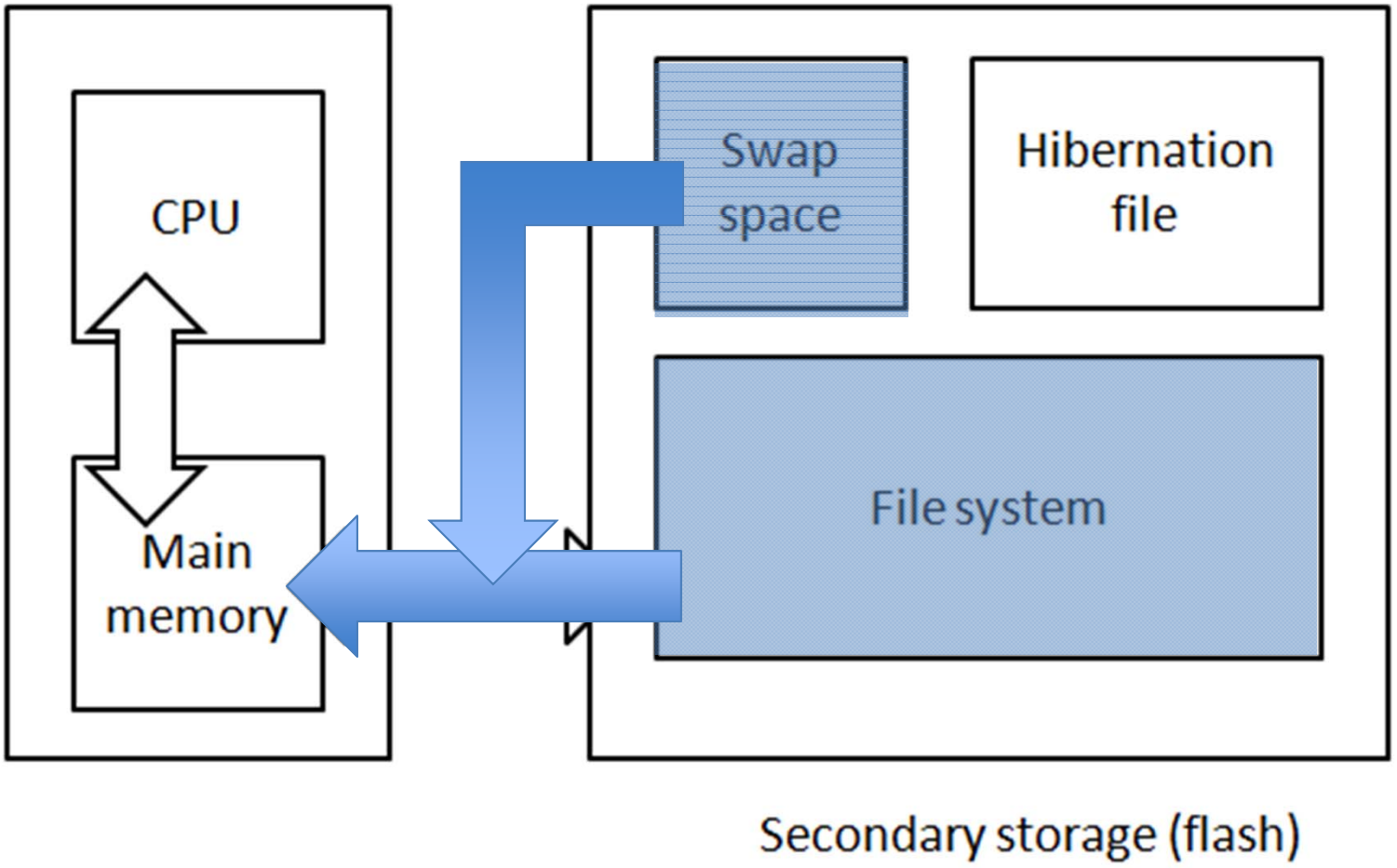
Booting (resume)

- In normal booting, we have to load “booting code and data” of an OS and applications
- In our method, the only data we need to load is the “working set.”
  - The working set is very small compare to that of the main memory.

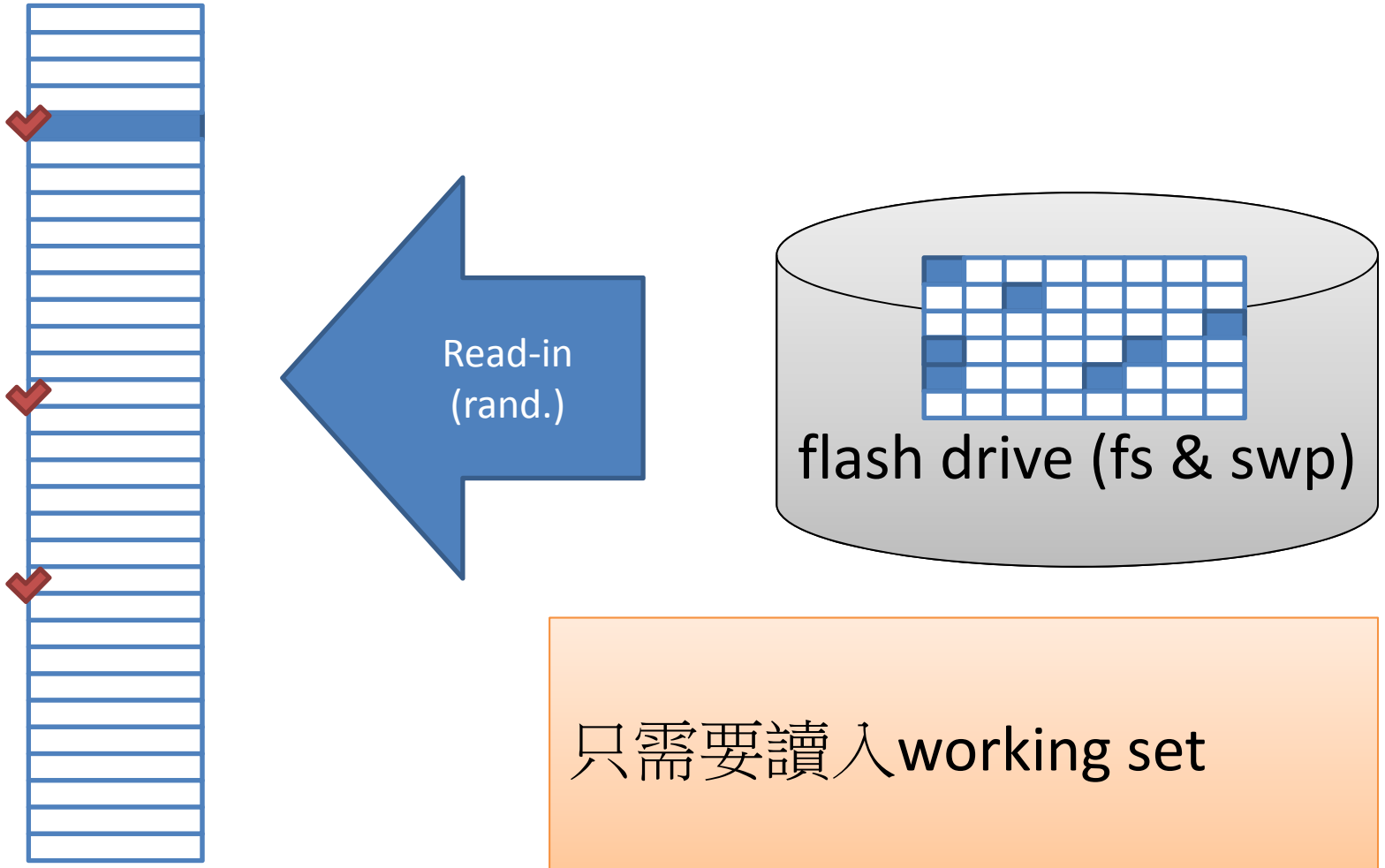
booting (resume)



booting (resume)

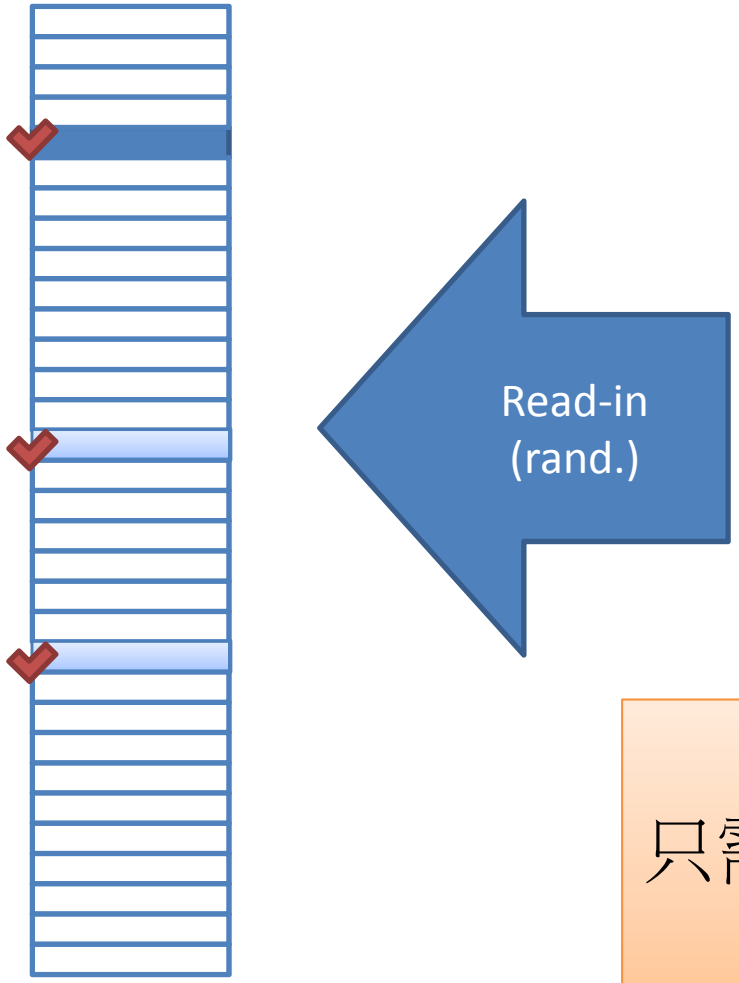


fastBooting（開機）

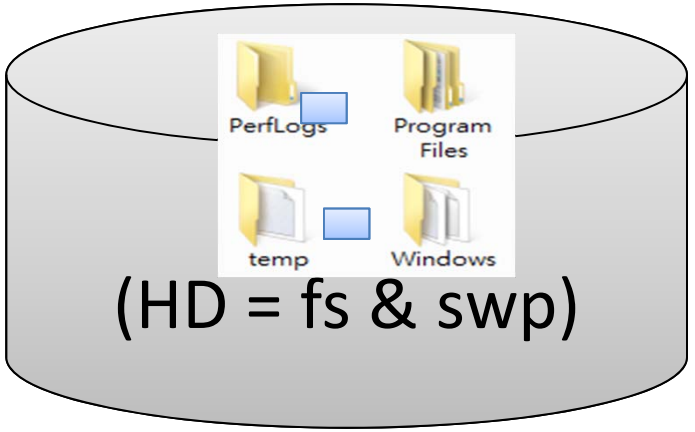


只需要讀入working set

fastBooting（開機）



Read-in
(rand.)



(HD = fs & swp)

只需要讀入working set

可否提升讀取效率

- Linux一次讀入（swap-in）8個page，共32KB的資料
  - 這是Linux內建的pre-fetch機制
- 根據統計Linux內建的pre-fetch機制的命中率（hit ratio）為50%
  - 可以對swap space進行優化，將hit ratio提升到80%

休眠式快速開機

- Power On
- Boot loader
- Enter Linux entry point
- Device initialization
- Read hibernate file /*boot-up completed*/
- Application execution /*swap in*/

休眠式快速開機

- Power On
- Boot loader
- Enter Linux entry point
- Device initialization
- Read hibernate file /*boot-up completed*/
  - 傳統方法在這個步驟需要載入全部的記憶體，耗時約512MB~2G/20MB = 『25 sec ~ 100 sec』
  - fast-on方法在此步驟約載入50MB記憶體，耗時『2.5 sec』
- Application execution

開機時間 計算公式

傳統休眠：

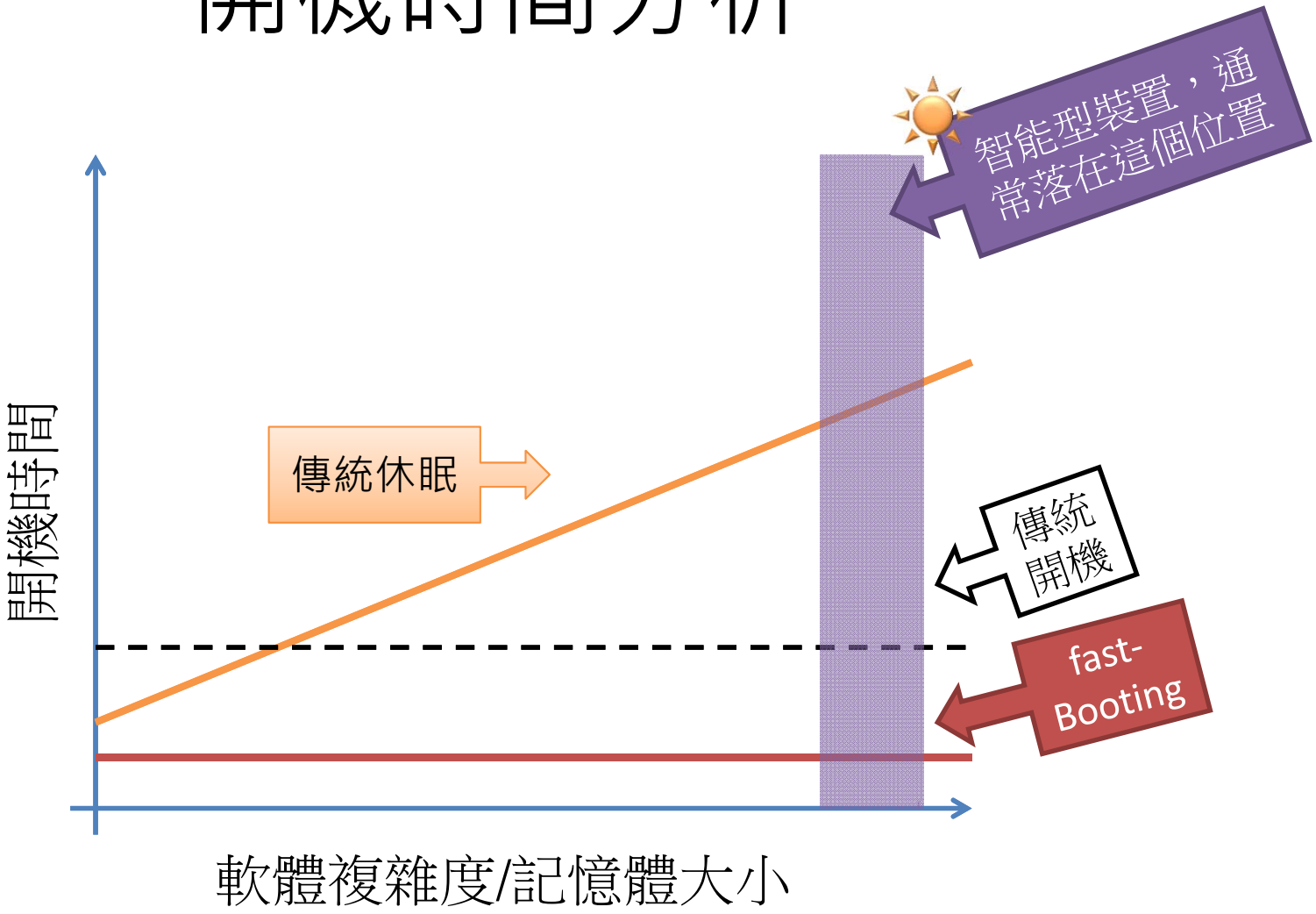
$$\frac{sizeof(main\_memory)}{speed_{seq.}} + device\_init$$

fastBooting：

$$\frac{sizeof(hiberfile)}{speed_{seq.}} + \frac{sizeof(workingset)}{speed_{rand.(32k)}} + device\_init$$

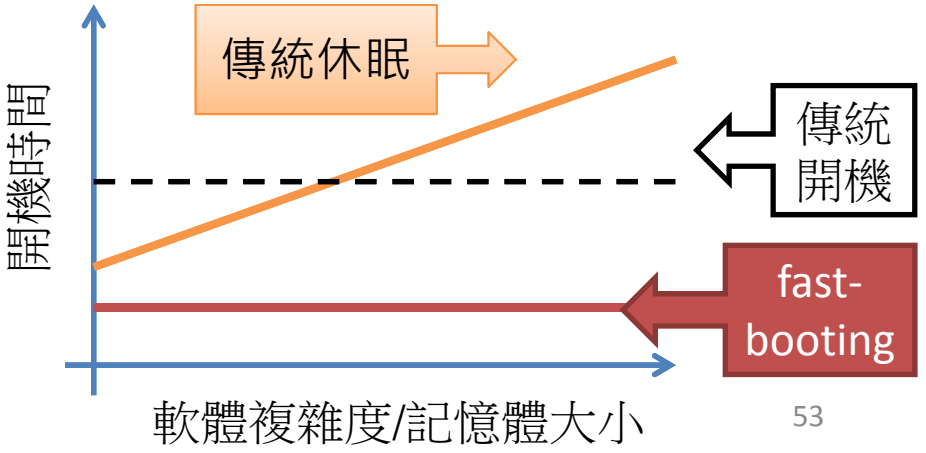
sizeof(hiberfile)=sizeof(unswappable)
=sizeof(kernel$_{runtime}$)

開機時間分析



fast-on的重要性質

- 開機時只載入「作業系統」與「應用程式」的關鍵記憶體
- 開機時間與「軟體複雜度」及「記憶體大小」無明顯關係



fastBooting

相較於傳統的休眠方法，更長的flash壽命

較少的寫入（**約為1/3以下**）、更快的關機