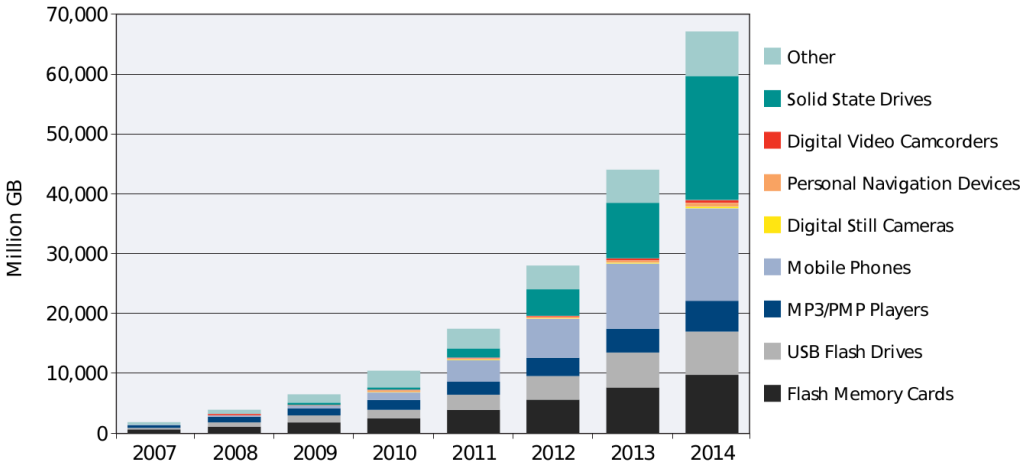
A Fine-Grained Analysis of Embedded Flash Storage Performance and Power Consumption

Pierre Olivier

Lab-STICC / Université de Bretagne Occidentale, Brest

pierre.olivier@univ-brest.fr

Context and Motivation



Source: Forward Insights, NAND Quarterly Insights Q3/09, www.forward-insights.com/
Report No. FI-NFL-NQI-Q309 September 2009, accessed 4/14/2010; used with permission.

Flash is here to stay

- Embedded domain
- But also servers, HPC, etc.

Performance and power consumption: critical metrics

- Flash has specific usage constraints → complex management mechanisms
- Explain performance / power consumption behavior from the application point of view → multiple factors

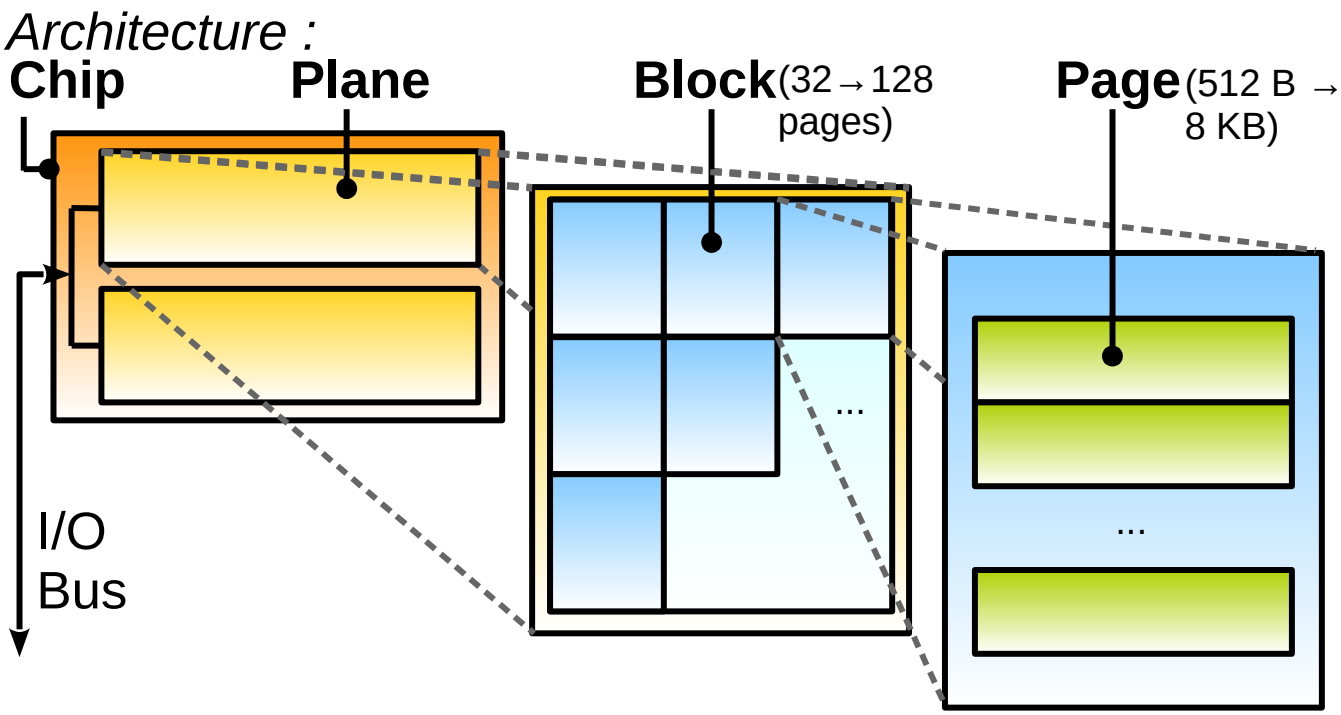
What are these factors ?

Case study : Embedded Linux Flash File Systems

Outline

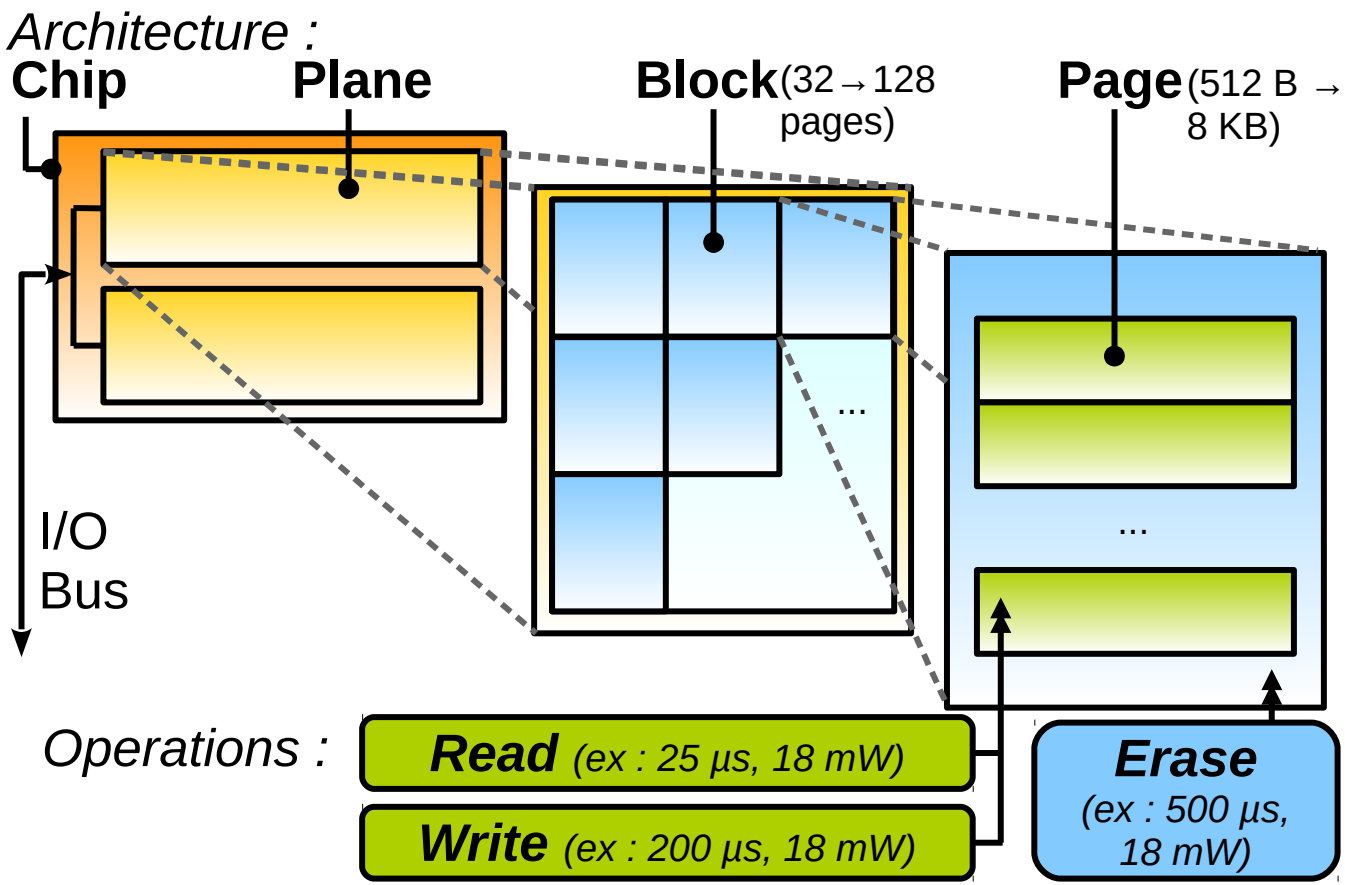
Outline

NAND Flash Architecture and Operations



- Hierarchical structure
- Latencies and power consumption: [1]

NAND Flash Architecture and Operations



- Hierarchical structure
- Latencies and power consumption: [1]

Constraints & Flash Management

Erase-before-write rule

- A page must be erased before being written
  - It means erasing the entire containing block
- **No in-place data updates**
  - Out-of-place updates
  - Logical to physical address mapping → invalid state
  - Garbage collection

Wearout of memory cells

- **Wear leveling**

Constraints management

- Complex **flash management systems**
  - FTL, **dedicated Flash File Systems (FFS)**

Constraints & Flash Management

Erase-before-write rule

- A page must be erased before being written
  - It means erasing the entire containing block
- **No in-place data updates**
  - Out-of-place updates
  - Logical to physical address mapping → invalid state
  - Garbage collection

Wearout of memory cells

- **Wear leveling**

Constraints management

- Complex **flash management systems**
  - FTL, **dedicated Flash File Systems (FFS)**

Constraints & Flash Management

Erase-before-write rule

- A page must be erased before being written
  - It means erasing the entire containing block
- **No in-place data updates**
  - Out-of-place updates
  - Logical to physical address mapping → invalid state
  - Garbage collection

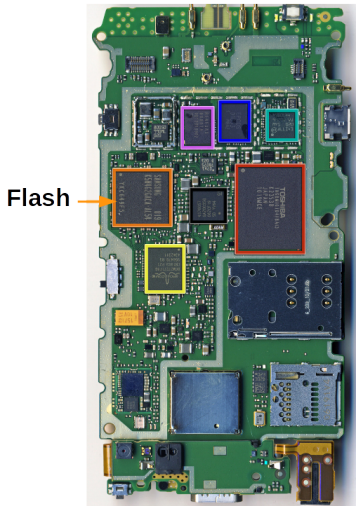
Wearout of memory cells

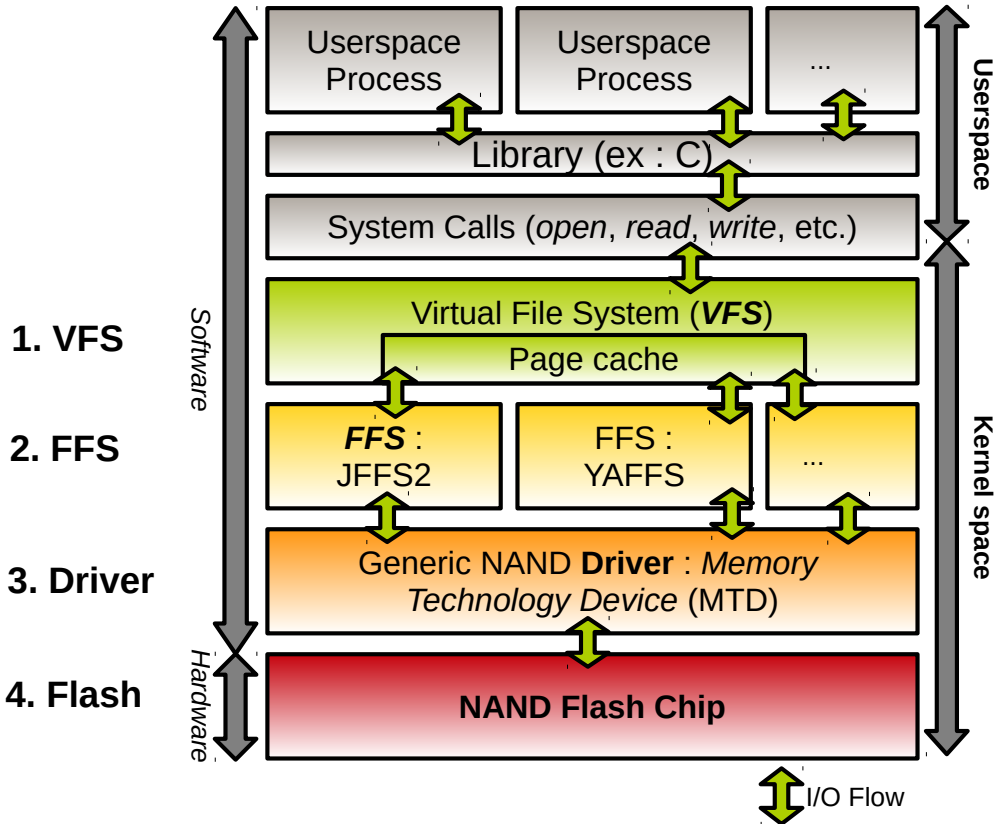
- **Wear leveling**

Constraints management

- Complex **flash management systems**
  - FTL, **dedicated Flash File Systems (FFS)**

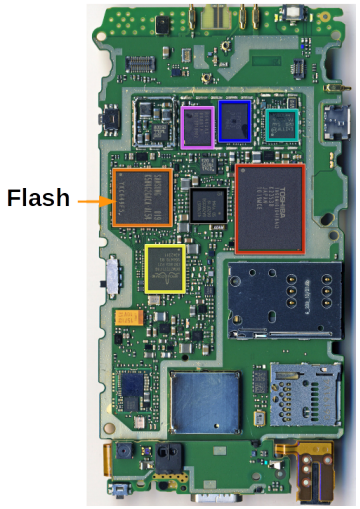
FFS with Linux

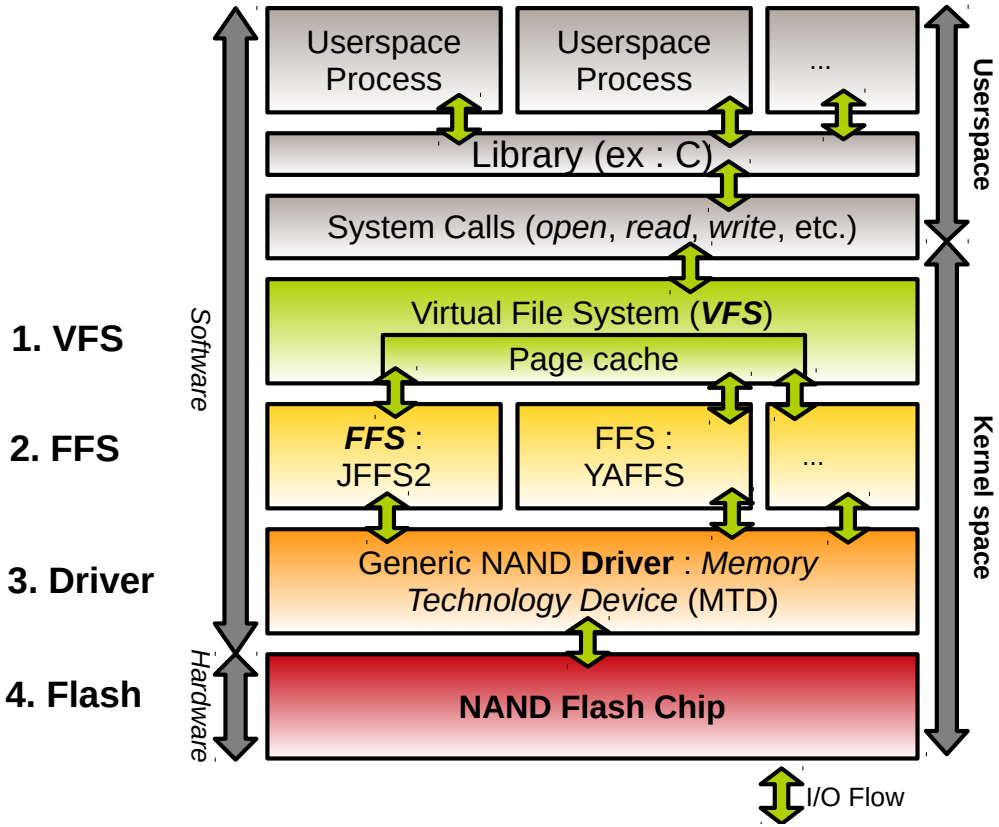




- Hard to explain performance and power consumption at the applicative level
  - Complex management, multiple layers of abstraction
  - Existing studies: no deep analysis considering the entire stack

FFS with Linux

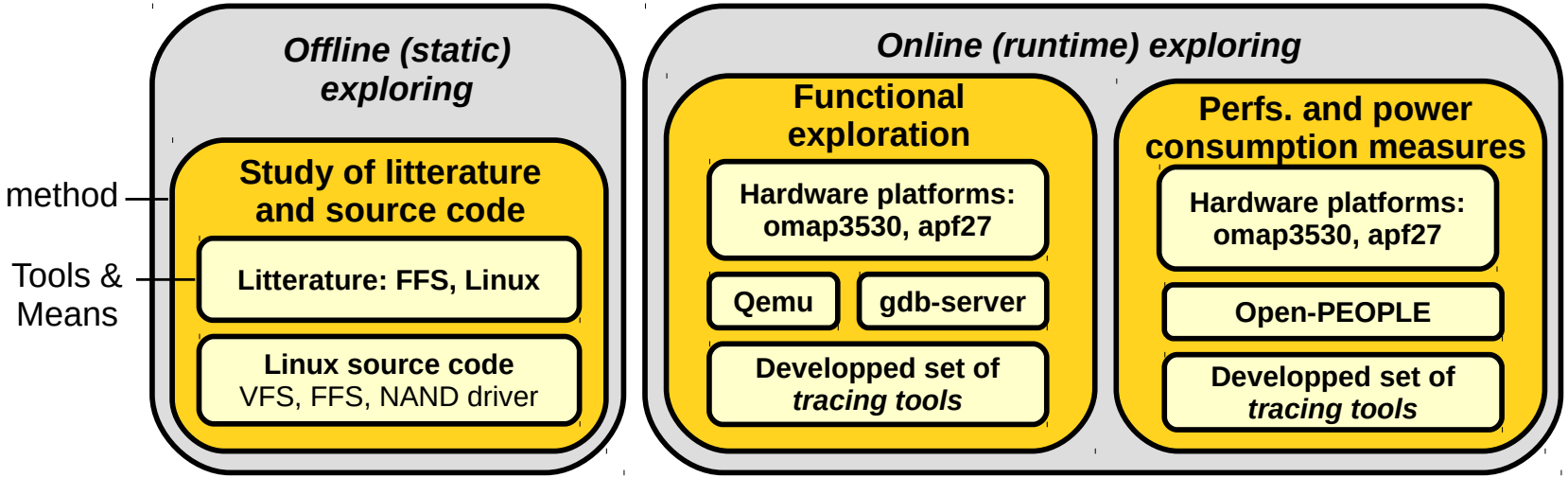




- **Hard to explain performance and power consumption at the applicative level**
  - Complex management, multiple layers of abstraction
  - Existing studies: no deep analysis considering the entire stack

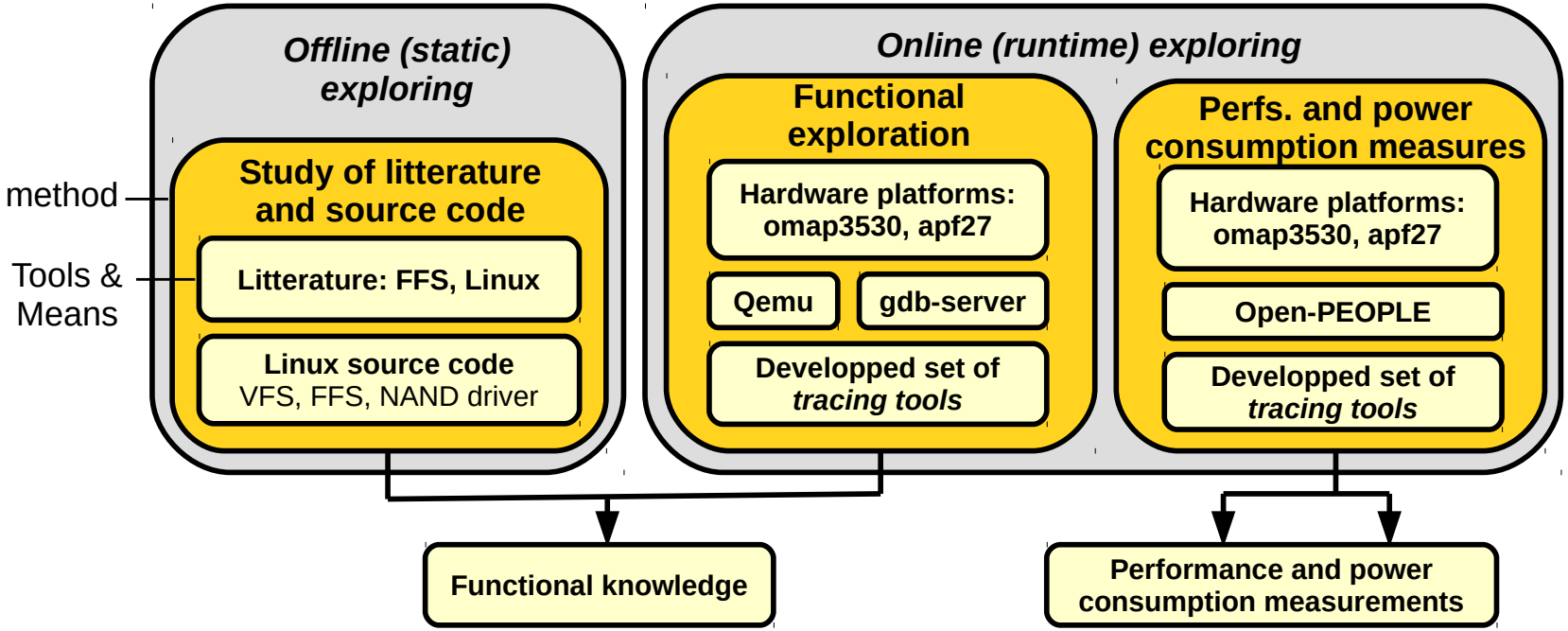
Outline

Methodology



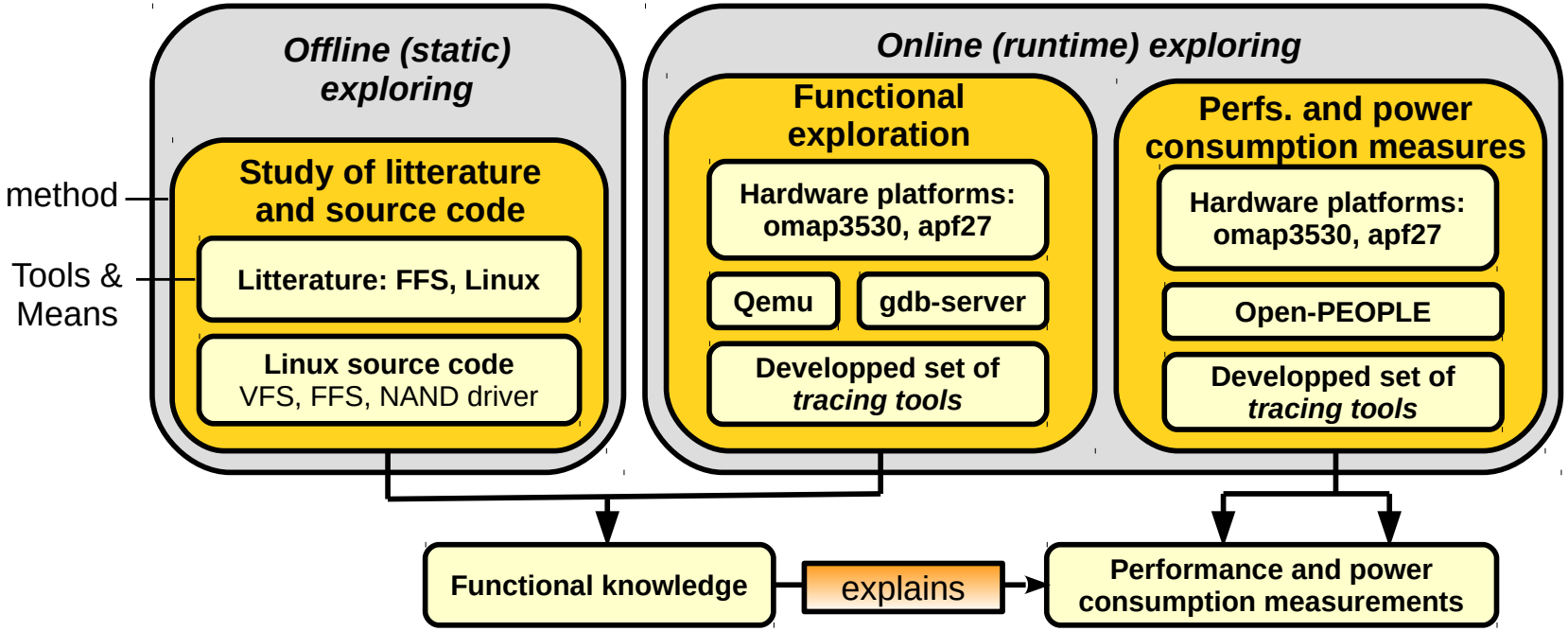
- Hardware platform: Mistral Omap3evm
  - Cortex A8 @720 Mhz, RAM 256 MB, NAND 256 MB [1]
- Power consumption measures: Open-PEOPLE [2] (NI-PXI-4472B digitizer [3])

Methodology



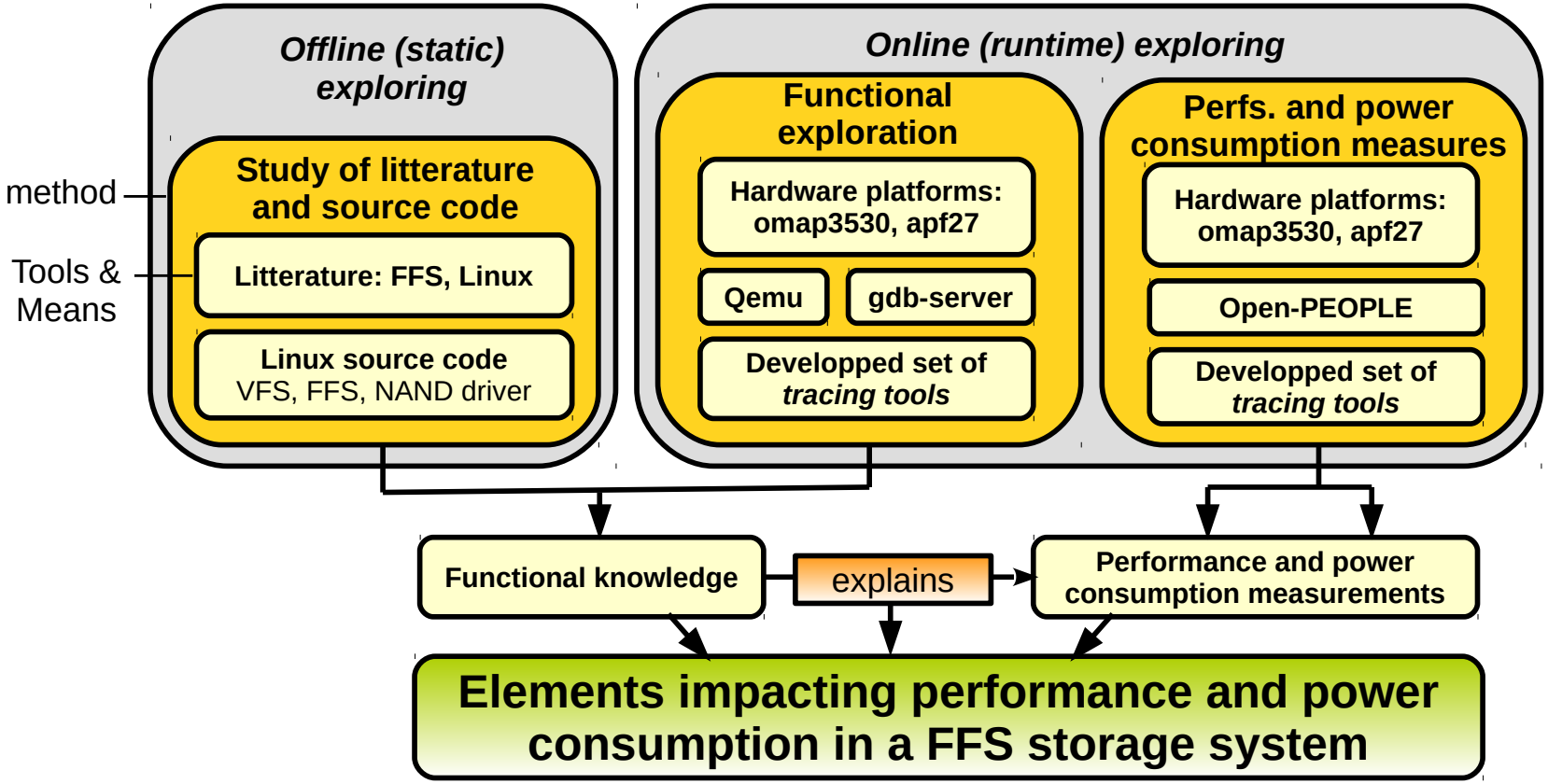
- Hardware platform: Mistral Omap3evm
  - Cortex A8 @720 Mhz, RAM 256 MB, NAND 256 MB [1]
- Power consumption measures: Open-PEOPLE [2] (NI-PXI-4472B digitizer [3])

Methodology



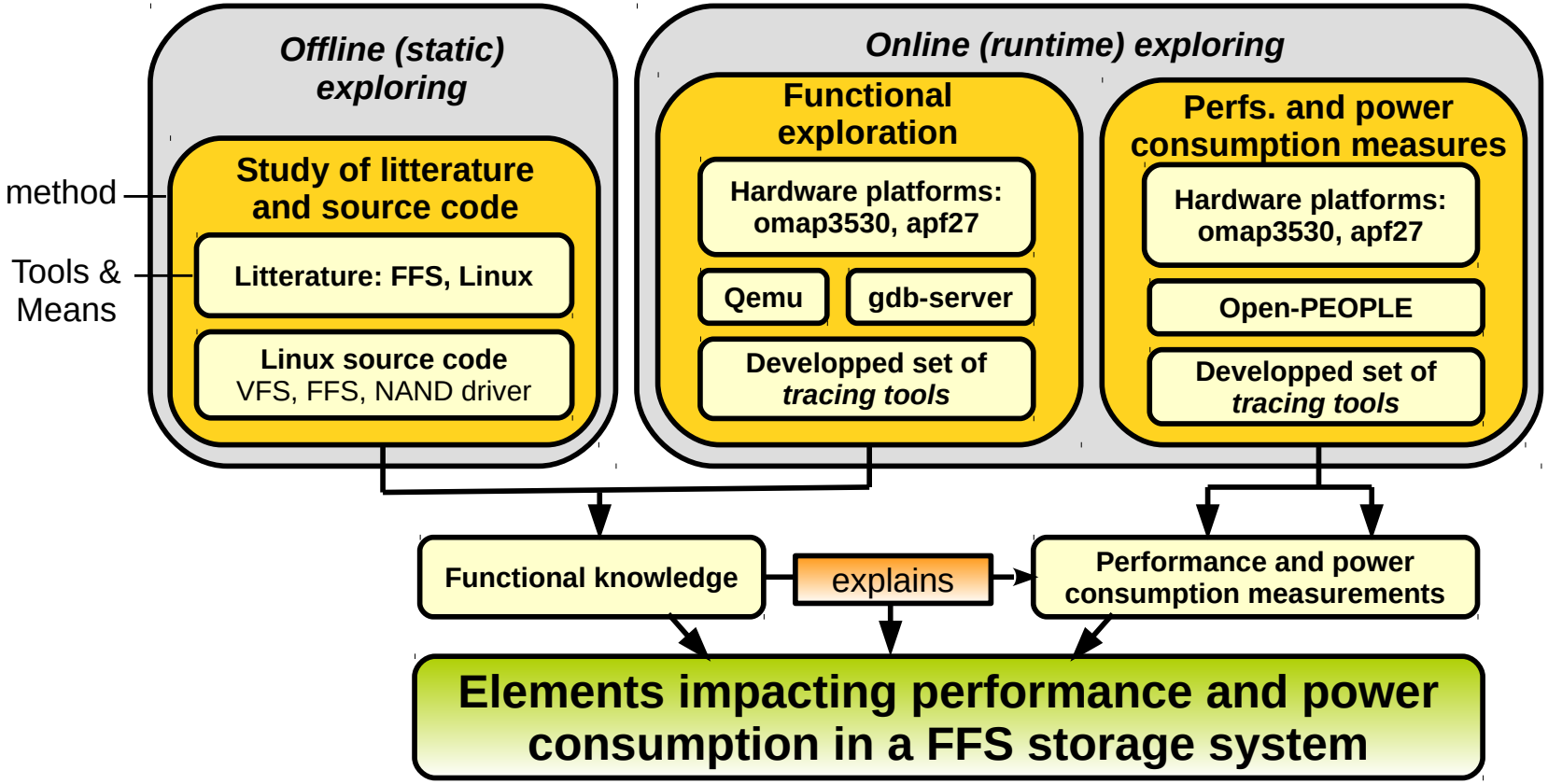
- Hardware platform: Mistral Omap3evm
  - Cortex A8 @720 Mhz, RAM 256 MB, NAND 256 MB [1]
- Power consumption measures: Open-PEOPLE [2] (NI-PXI-4472B digitizer [3])

Methodology



- Hardware platform: Mistral Omap3evm
  - Cortex A8 @720 Mhz, RAM 256 MB, NAND 256 MB [1]
- Power consumption measures: Open-PEOPLE [2] (NI-PXI-4472B digitizer [3])

Methodology

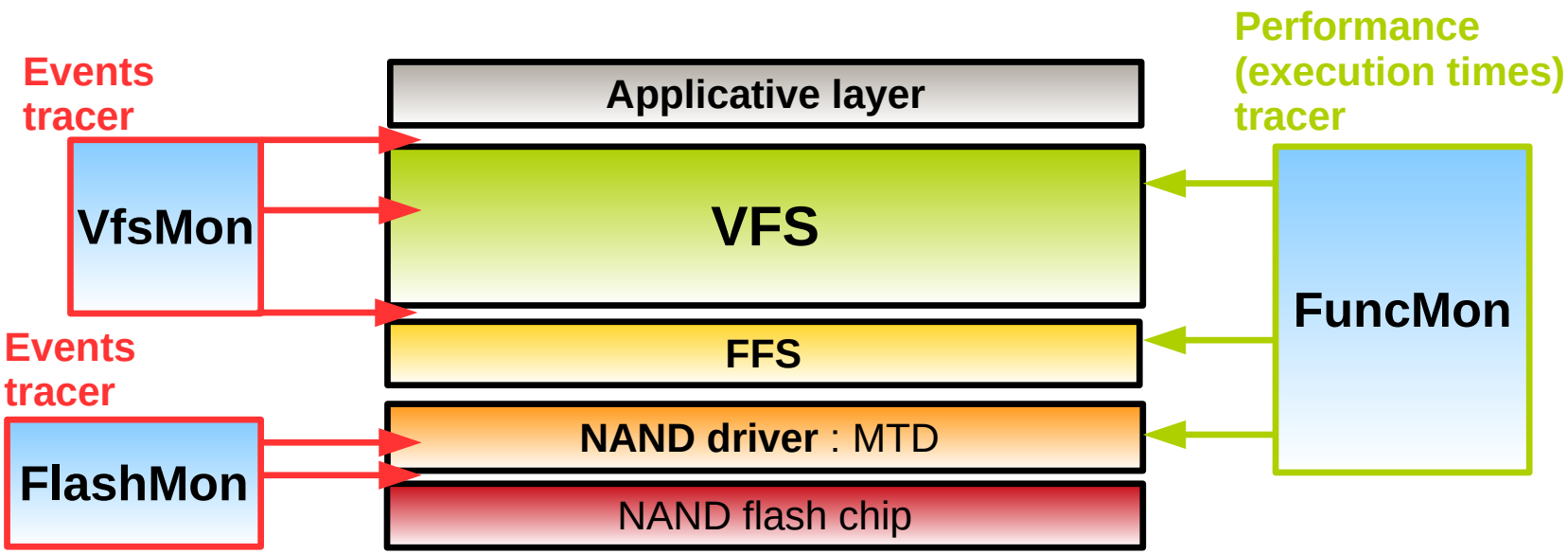


- Hardware platform: Mistral Omap3evm
  - Cortex A8 @720 Mhz, RAM 256 MB, NAND 256 MB [1]
- Power consumption measures: Open-PEOPLE [2] (NI-PXI-4472B digitizer [3])

Developped Tools

Existing tools

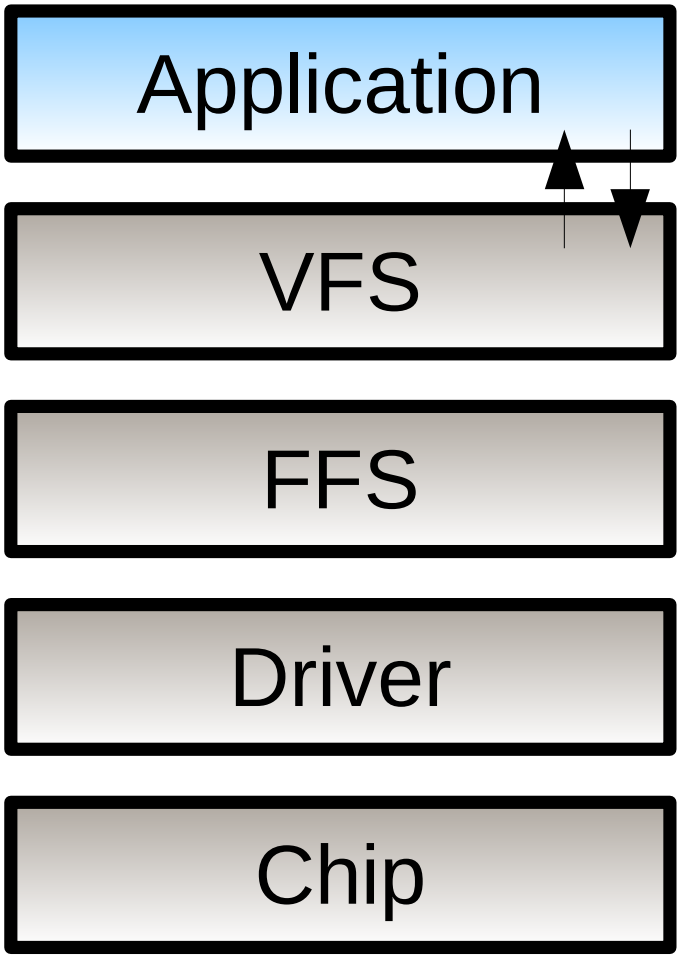
- Ftrace [4], SystemTap [5], Oprofile [6] → not dedicated to FFS storage inspection



- Flashmon [7], VfsMon & FuncMon [8]
- **Linux kernel modules**, core functions using **kernel probes** [9]

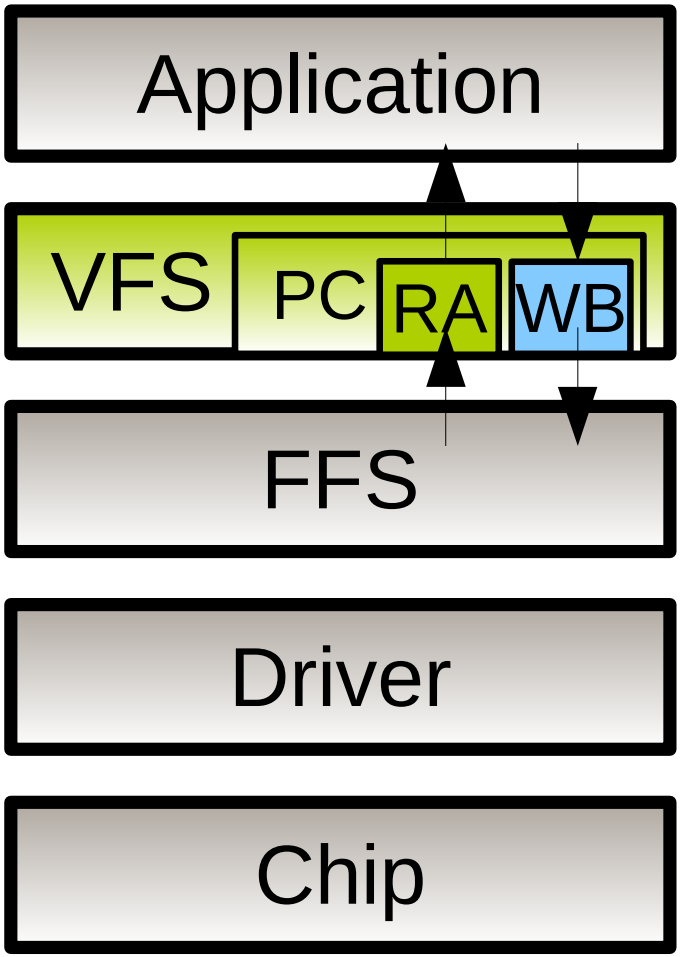
Outline

Functional Exploration



System calls:
`read` & **`write`**

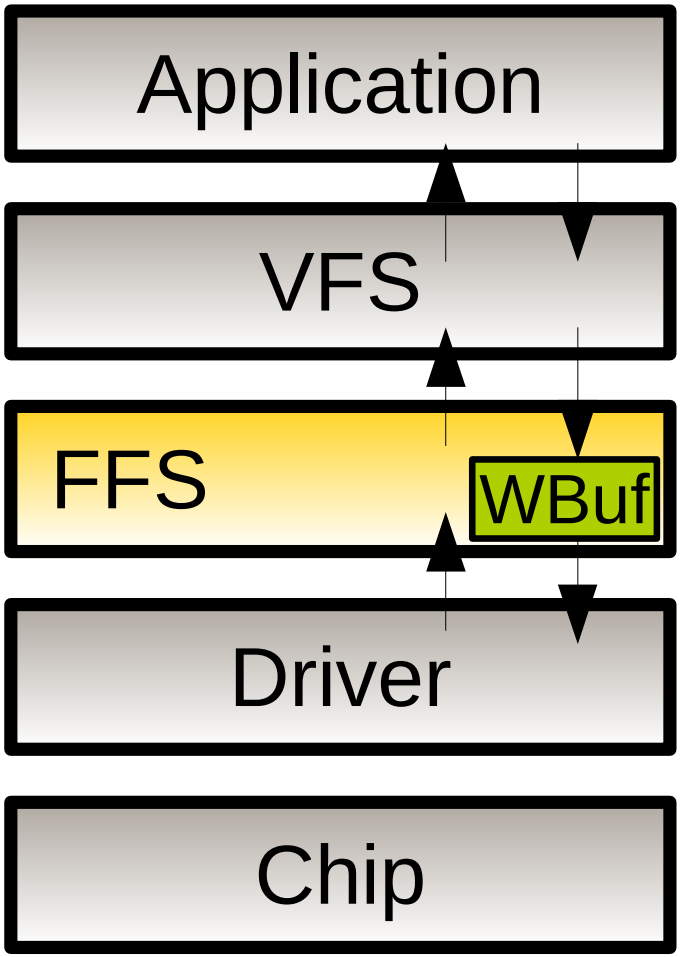
Functional Exploration



VFS level:

- Requests divided in **Linux pages** (4KB)
- RAM buffer : **Page Cache**
- Page cache level optimizations :
  - **Read-Ahead**
  - **Write-Back**

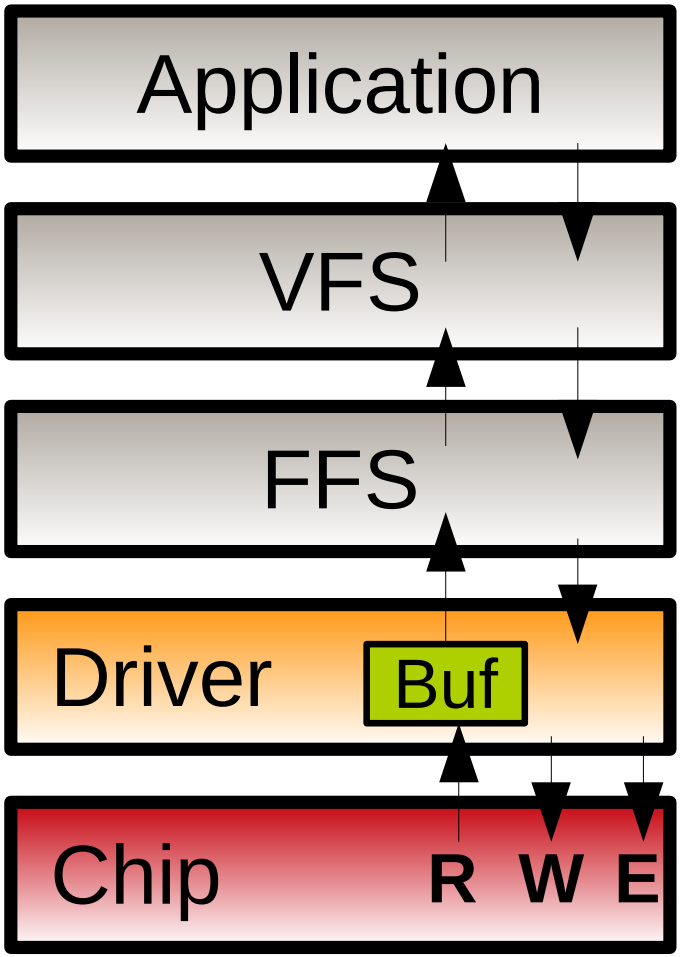
Functional Exploration



FFS level (ex JFFS2):

- Data divided in on-flash **nodes**
  - Variable size & on-flash location
- (small) **write buffer**
- **Garbage Collector**
  - **Synchronous** with I/O requests
  - **Asynchronous (kernel thread)**

Functional Exploration



Driver level :

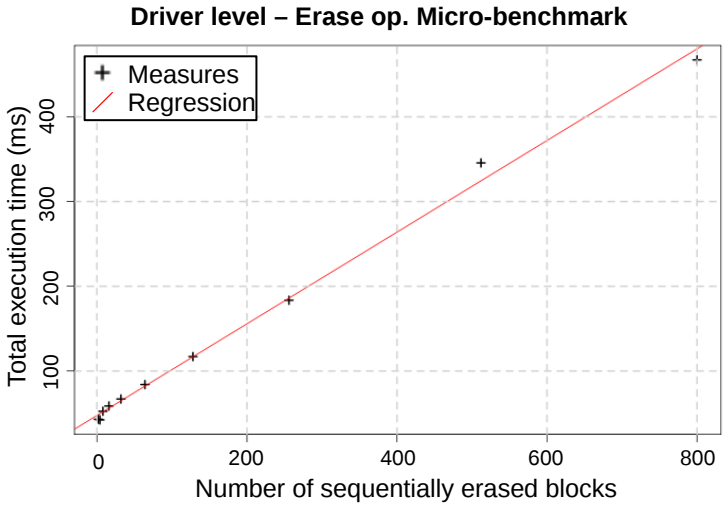
- Perform flash operations :
  - Read, Write, Erase
- Read buffer

Driver Level

- Driver level **micro-benchmarks**
- Linear regression → single access exec. time
- Mean power measurement

$$E_{CPU} > E_{Mem}$$

[10]

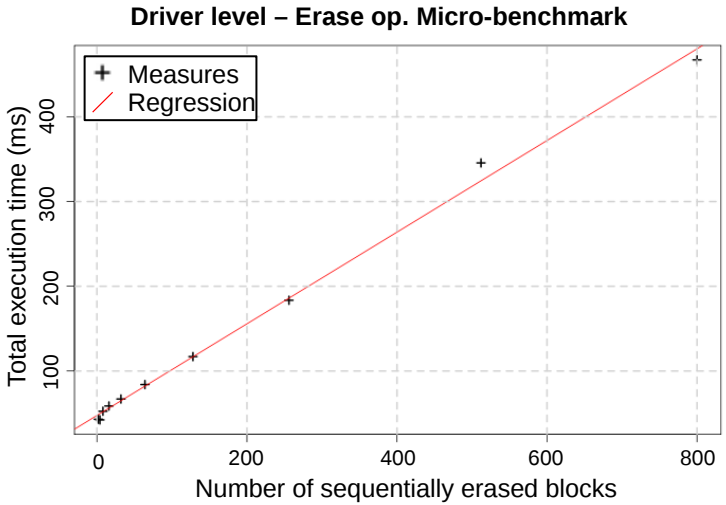


Driver Level

- Driver level **micro-benchmarks**
- Linear regression → single access exec. time
- Mean power measurement

$$E_{CPU} > E_{Mem}$$

[10]



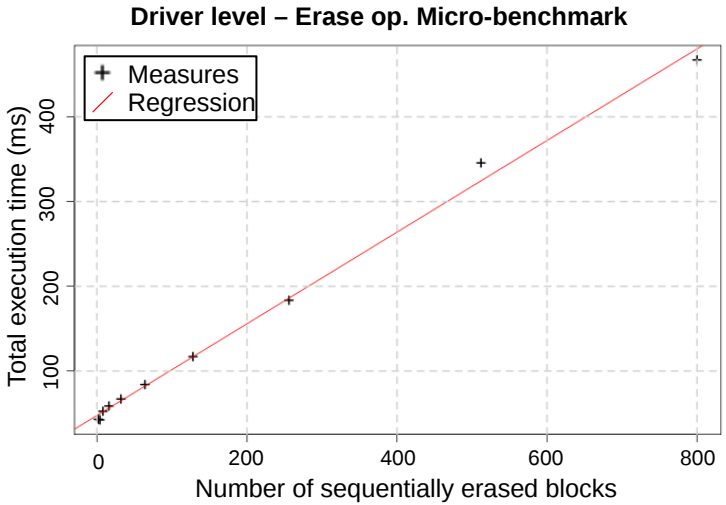
Driver Level

- Driver level **micro-benchmarks**
- Linear regression → single access exec. time
- Mean power measurement

Performance (µs)		**Power consumption** (µJ)			
Operation	**Latency**	**Operation**	**CPU**	**Memory**	**Total**
Read	185.065	**Read**	**34.6**	2.2	36.8
Write	407.6	**Write**	**74.6**	6.3	81
Erase	536.4	**Erase**	**97.5**	8.5	106

$$E_{CPU} > E_{Mem}$$

[10]



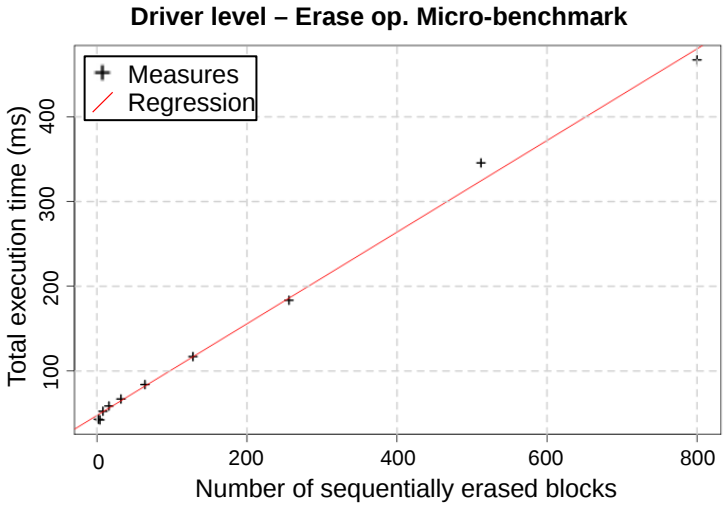
Driver Level

- Driver level **micro-benchmarks**
- Linear regression → single access exec. time
- Mean power measurement

Performances (µs)		Driver overhead / datasheet	
Operation	**Latency**	**Operation**	**Value**
Read	185.065	**Read**	**+25 %**
Write	407.6	**Write**	**+33 %**
Erase	536.4	**Erase**	+7 %

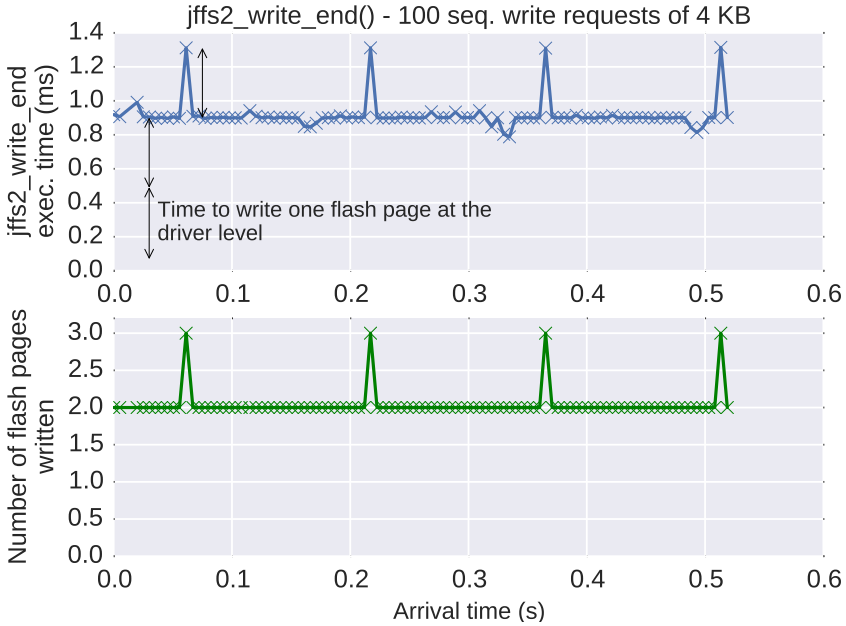
$$E_{CPU} > E_{Mem}$$

[10]



FFS Level: read & write operations

Case study: JFFS 2



- Most of the time & energy at the FFS level:
  - **Flash accesses**

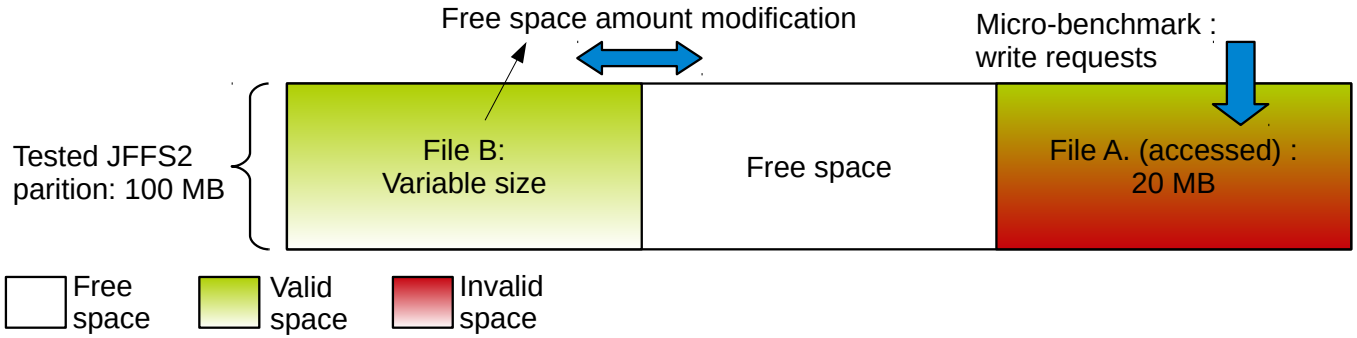
FFS Level: read & write operations

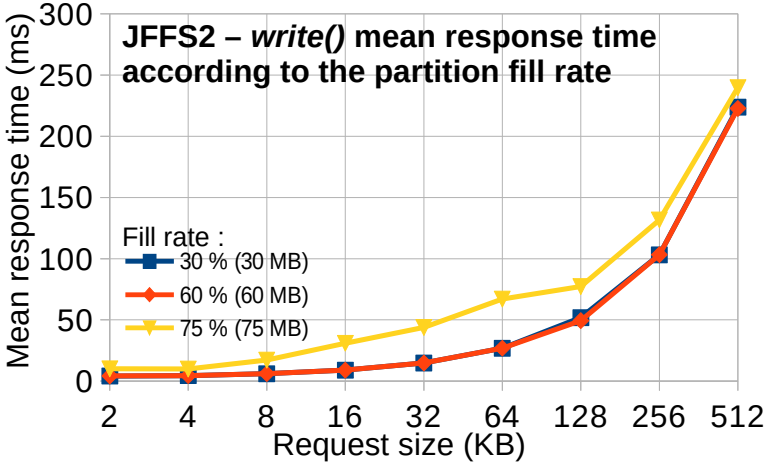
Case study: JFFS 2

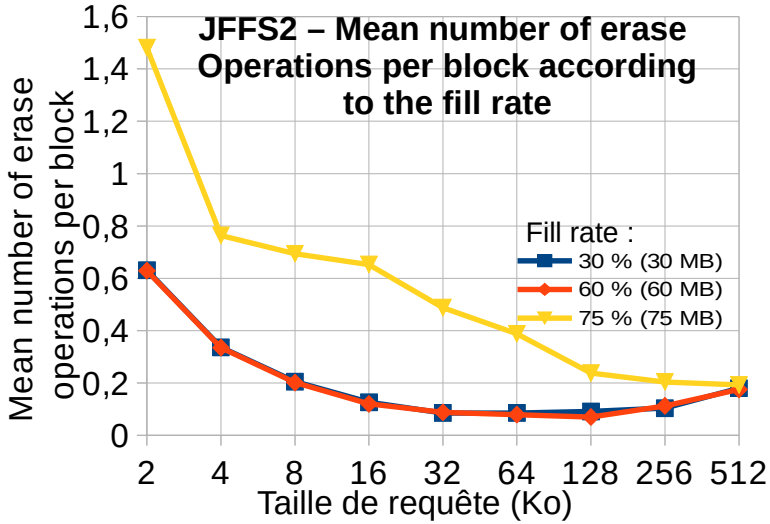
- Most of the time & energy at the FFS level:
  - **Flash accesses**

- On-flash **fragmentation** defines flash accesses for read operations
  - **Write history** of the file

FFS Level: garbage collection
JFFS2 - Synchronous GC





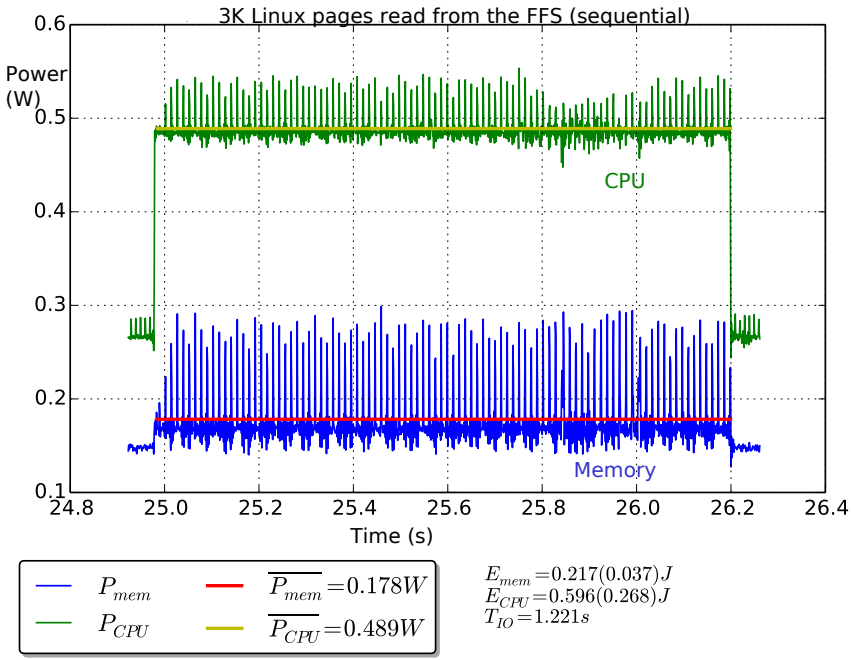


- Because of the GC, the **initial state** (amount of valid / invalid / free space) has an impact on performance / power consumption [11]
- Asynchronous GC: uses I/O timeouts to reclaim invalid data

VFS Level

Page cache impact

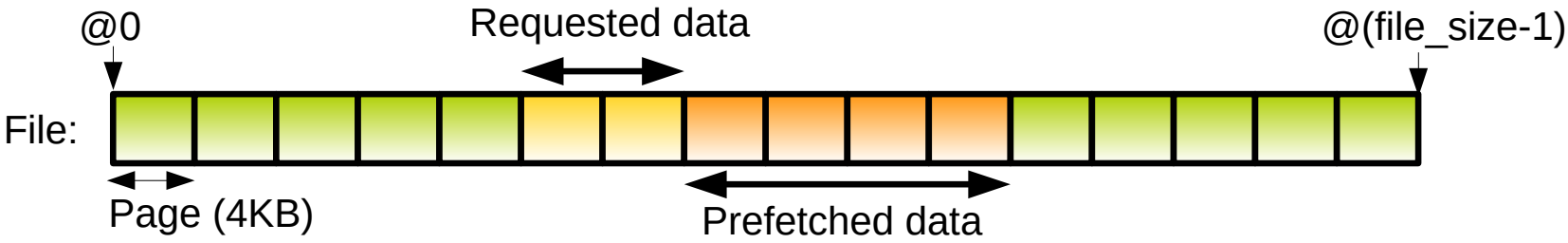
- Buffers all file accesses
- Speeds up future read operations ...
- ... and reduces the energy consumed



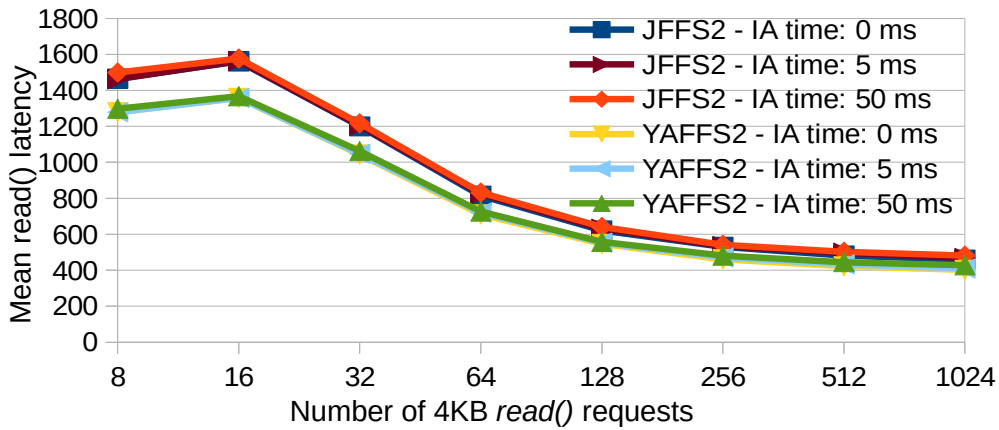
VFS Level: Read-Ahead

Page cache optimizations

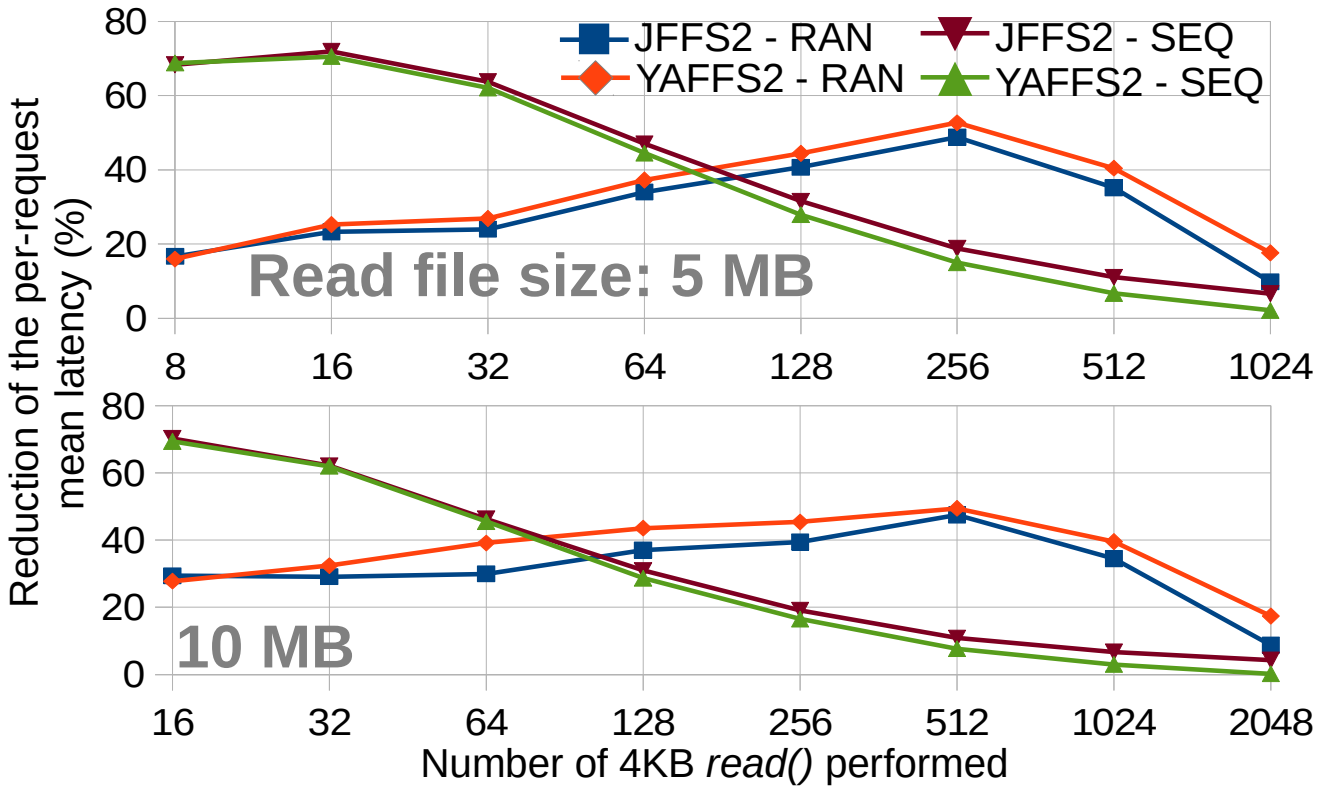
- **Read-ahead**: read prefetching from the FFS in the page cache
- Page cache **write-back** (only with UBIFS)



- Designed for HDD with good sequential performance
- Rely on **asynchronous I/O**
- ... but the NAND driver is **synchronous**



VFS Level: Read-Ahead (2)



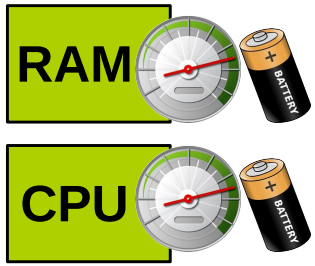
- Read-Ahead → negative impact on performance and power consumption [12]
  - When prefetched data is not used

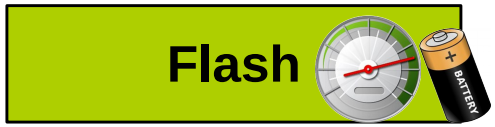
Outline

Conclusion

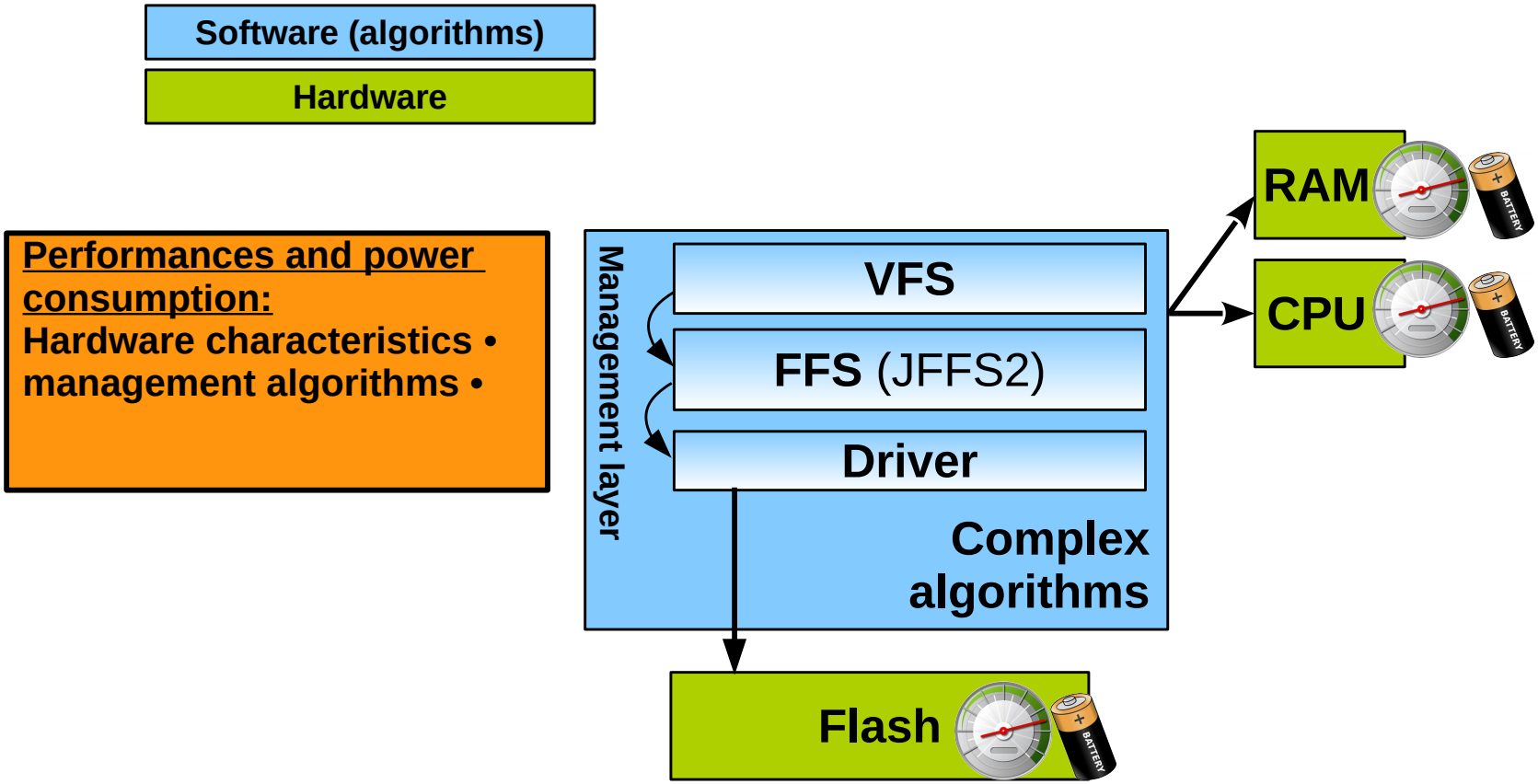
Hardware

Performances and power consumption:
Hardware characteristics •

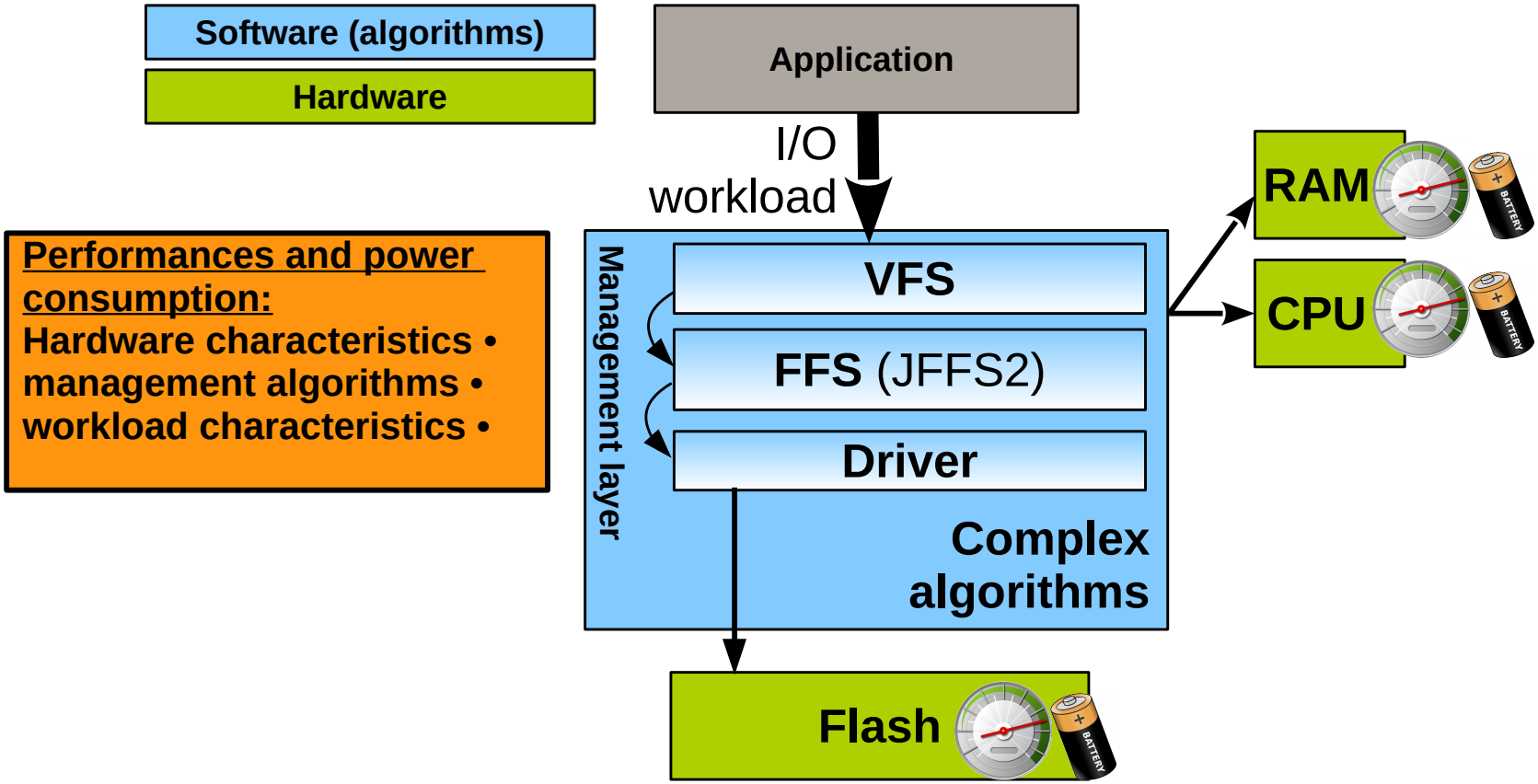




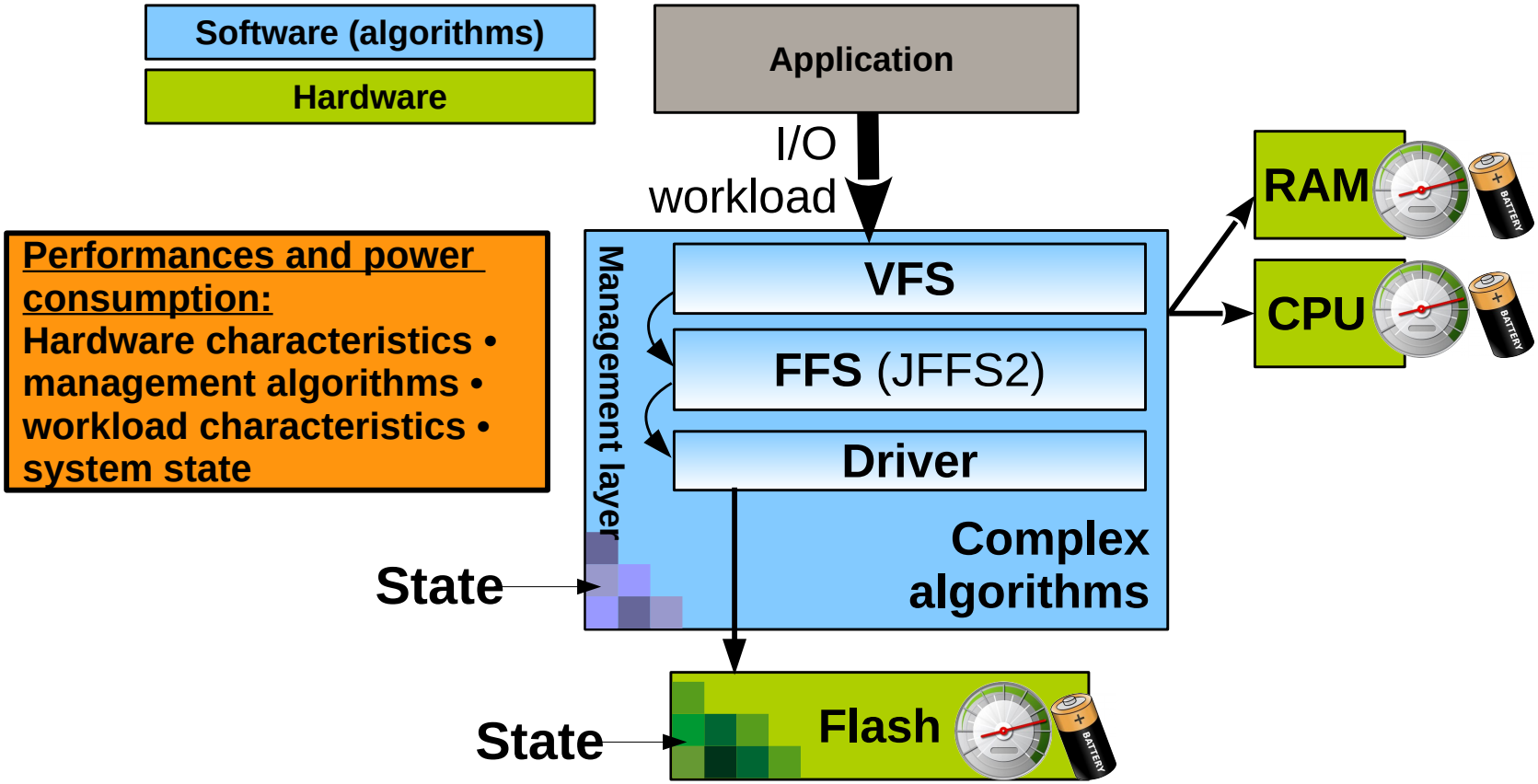
Conclusion


Software (algorithms)
Hardware
Performances and power consumption:
Hardware characteristics •
management algorithms •
Management layer
VFS
FFS (JFFS2)
Driver
Complex algorithms
RAM
CPU
Flash
BATTERY

Conclusion


Software (algorithms)
Hardware
Application
I/O workload
Performances and power consumption:
Hardware characteristics •
management algorithms •
workload characteristics •
Management layer
VFS
FFS (JFFS2)
Driver
Complex algorithms
RAM
CPU
Flash
BATTERY

Conclusion


Software (algorithms)
Hardware
Application
I/O workload
Performances and power consumption:
Hardware characteristics •
management algorithms •
workload characteristics •
system state
Management layer
VFS
FFS (JFFS2)
Driver
Complex algorithms
RAM
CPU
State
State
Flash

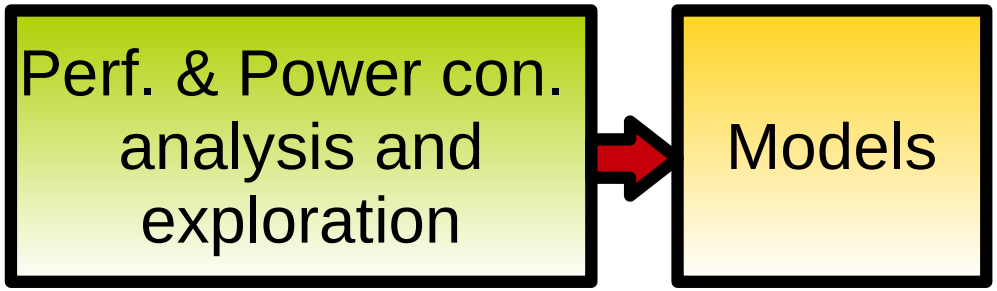
Following Work

Perf. & Power con. analysis and exploration

Building a performance and power consumption simulator

- Building **models** of various types
  - Functional, performance, power consumption, etc.
- Models implemented in a **simulator**, OpenFlash
  - Performance and power consumption estimations, flash management prototyping
  - Validated against real measures (error $< 10\%$)

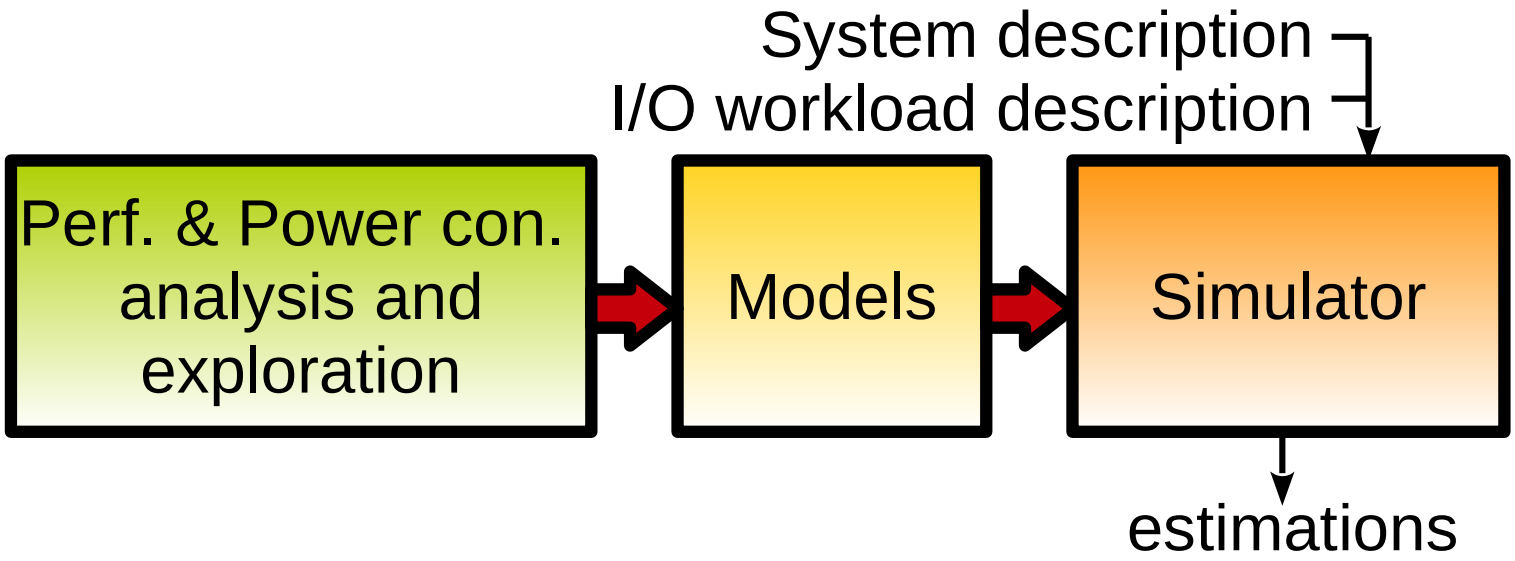
Following Work



Building a performance and power consumption simulator

- Building **models** of various types
  - Functional, performance, power consumption, etc.
- Models implemented in a **simulator**, OpenFlash
  - Performance and power consumption estimations, flash management prototyping
  - Validated against real measures (error < 10%)

Following Work



Building a performance and power consumption simulator

- Building **models** of various types
  - Functional, performance, power consumption, etc.
- Models implemented in a **simulator**, OpenFlash
  - Performance and power consumption estimations, flash management prototyping
  - Validated against real measures (error < 10%)

References 1

[1] Mistral Solutions, "OMAP35x evaluation module," 2013. [Online]. Available: http://www.mistralsolutions.com/pes-products/development-platforms/omap35x-evm-.html

[2] E. Senn, D. Chillet, O. Zendra, C. Belleudy, S. B. Bilavarn, R. B. Atitallah, C. Samoyeau, and A. Fritsch, "Open-people: Open power and energy optimization PLatform and estimator," in *Digital System Design (DSD), 2012 15th Euromicro Conference on*. IEEE, 2012, p. 668–675. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6386956

[3] National Instruments, "Carte d'acquisition NI PXI-4472B," 2014. [Online]. Available: http://sine.ni.com/nips/cds/print/p/lang/fr/nid/12184

[4] S. Rostedt, "Ftrace documentation," 2008. [Online]. Available: https://www.kernel.org/doc/Documentation/trace/ftrace.txt

[5] F. C. Eigler, V. Prasad, W. Cohen, H. Nguyen, M. Hunt, J. Keniston, and B. Chen, *Architecture of systemtap: a Linux trace/probe tool*, 2005.

[6] J. Levon, "OProfile manual," *Victoria University of Manchester*, 2004. [Online]. Available: https://pixhawk.ethz.ch/_media/software/optimization/oprofile/oprofilemanual.pdf

[7] P. Olivier, J. Boukhobza, and E. Senn, "Flashmon v2: Monitoring raw NAND flash memory I/O requests on embedded linux," *SIGBED Rev.*, vol. 11, no. 1, p. 38–43, 2014. [Online]. Available: http://doi.acm.org/10.1145/2597457.2597462

[8] P. Olivier, J. Boukhobza, M. Soula, M. Le Grand, I. Chaib Draa, and E. Senn, "A tracing toolset for embedded linux flash file systems," in *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS)*, Bratislava, Slovaquie, 2014, (A paraître).

[9] J. Keniston, P. S. Panchamukhi, and M. Hiramatsu, "Linux kernel probes documentation," 2014. [Online]. Available: https://www.kernel.org/doc/Documentation/kprobes.txt

[10] P. Olivier, J. Boukhobza, and E. Senn, "Modeling driver level NAND flash memory I/O performance and power consumption for embedded linux," in *2013 11th International Symposium on Programming and Systems (ISPS)*, 2013, pp. 143–152.

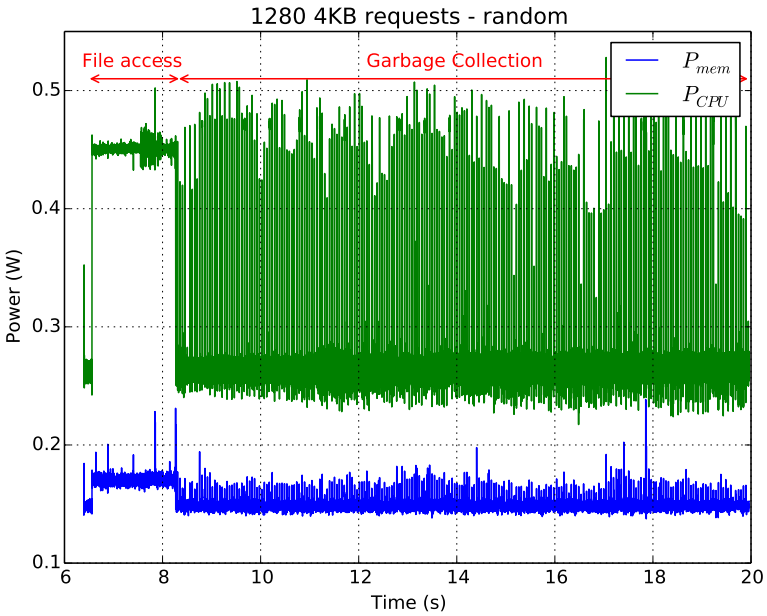
References II

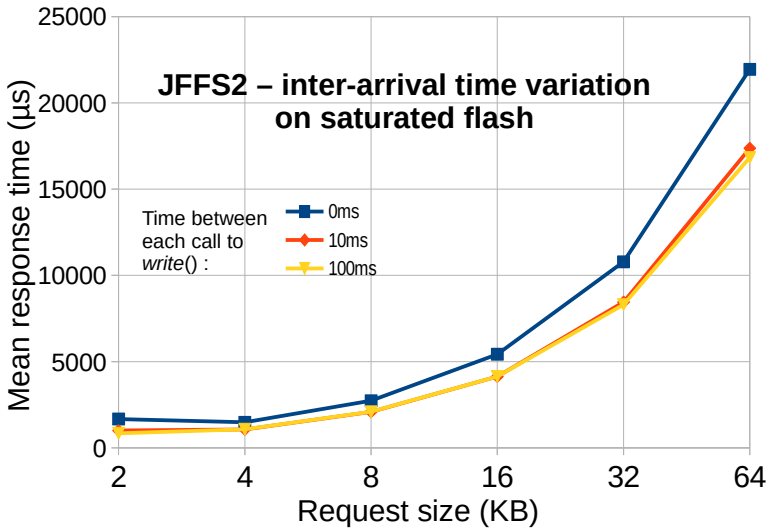
[11] ——, "Micro-benchmarking flash memory File-System wear leveling and garbage collection: A focus on initial state impact," in *Proceedings of the 2012 IEEE 15th International Conference on Computational Science and Engineering*, ser. CSE '12. Washington, DC, USA: IEEE Computer Society, 2012, p. 437–444. [Online]. Available: http://dx.doi.org/10.1109/ICCSE.2012.67

[12] ——, "Revisiting read-ahead efficiency for raw nand flash storage in embedded linux," *Proceedings of the 4th Embed With Linux (EWiLi) Workshop*, 2014.

Backup slide - FFS Level: garbage collection

JFFS2 - Asynchronous GC





- Asynchronous GC uses I/O timeouts to reclaim invalid data