Tor Vergata, Rome University
Department of Engineering

Data Transfers between User and Kernel Memory via Dedicated DMA Engine

Candidate:
Antonio Papa

Supervisor:
Prof. Marco Cesati

Ass. Supervisor:
Eng. Emiliano Betti

2013-2014

Presentation Outline

Memory Copy Operations

- Memory copy operations are implemented through a sequence of load and store instructions realised by CPU .

- The **load** and **store** instructions use a set of resources as:
  - CPU registers (32 or 64 bit)
  - Cache memory

Memory Copy Operations

- Memory copy operations are implemented through a sequence of load and store instructions realised by CPU .

- The **load** and **store** instructions use a set of resources as:
  - CPU registers (32 or 64 bit)
  - Cache memory

Furthermore it may happen to have to deal with the **Cache Pollution** problem

Cache Pollution

An executing computer program loads data into CPU cache unnecessarily, thus causing other useful data to be evicted from the cache into lower levels of the memory hierarchy, degrading performance.

Memory-Memory operations between kernel and user space

- Every system call makes copy operations between kernel and user space

Memory-Memory operations between kernel and user space

- Every system call makes copy operations between kernel and user space
- Device drivers have to copy data from/to buffer in user space via *file operations*

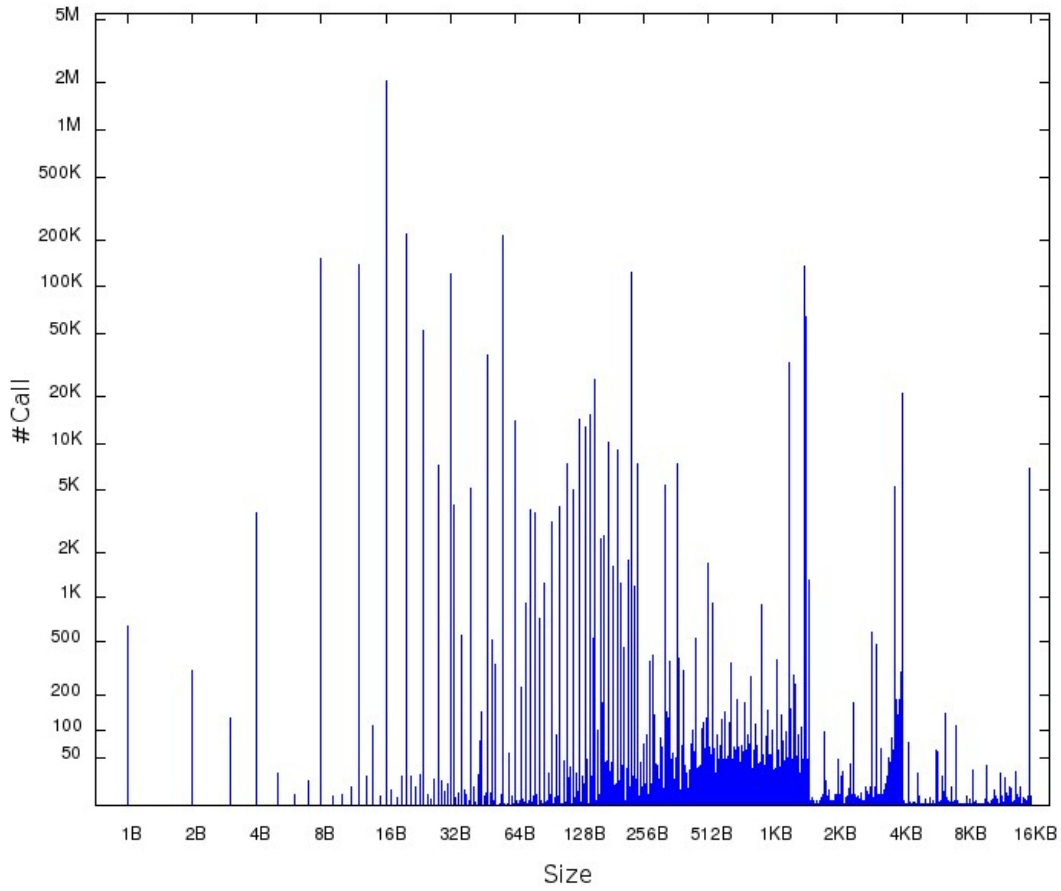
Memory-Memory operations between kernel and user space

- Every system call makes copy operations between kernel and user space
- Device drivers have to copy data from/to buffer in user space via *file operations*
- Furthermore data transfers between Kernel and User space are common in some Kernel sub-system

  For example:
  - The TCP/IP stack has to copy the payload (*receive-side*) from kernel memory to user application memory

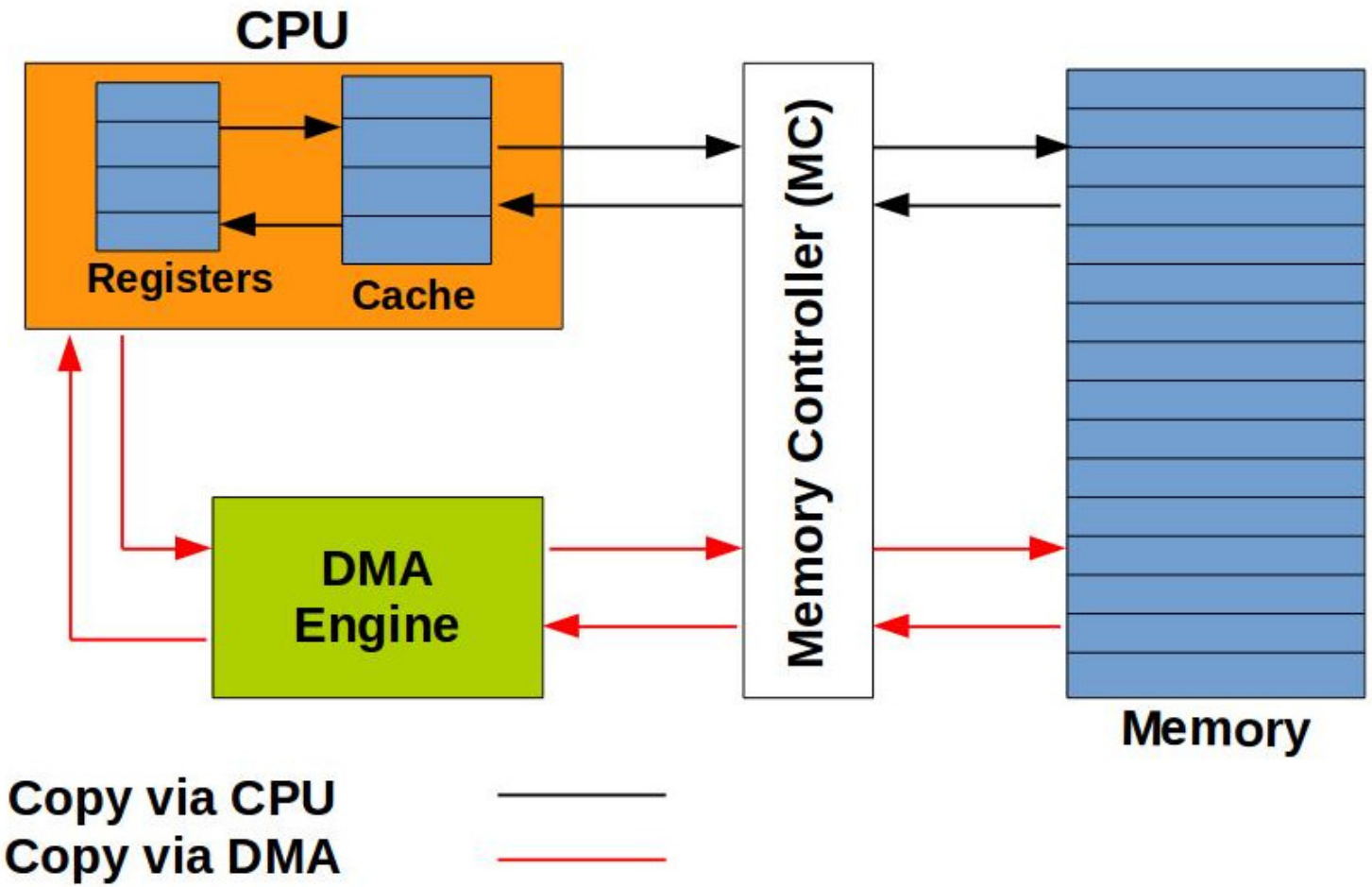
Data transfers between kernel and user space

- O.S. exposed to a medium workload
- Time slice of 100 seconds



~ 3,6 M copy operations between kernel and user space

Motivations for Hardware Copy Engine (1)



Motivations for Hardware Copy Engine (2)

Advantages:

- Optimisation of CPU resources
- Offload copy operations between kernel and user space

Motivations for Hardware Copy Engine (2)

Advantages:

- Optimisation of CPU resources
- Offload copy operations between kernel and user space

Goals:

- Better performance in data transfer
- Predictability of copy operations
- Control of cache pollution

DMA-CFTU Extension(1)

Copy memory support

- Generic interface to make DMA trasfers
- Kernel copy functions set-up
- Two Policies to manage the DMA channels:
  - *Exclusive Channel*: DMA channel is bound to one CPU (private channel)
  - *Shared Channel*: Every DMA channel is shared (public channel)

DMA-CFTU Extension(1)

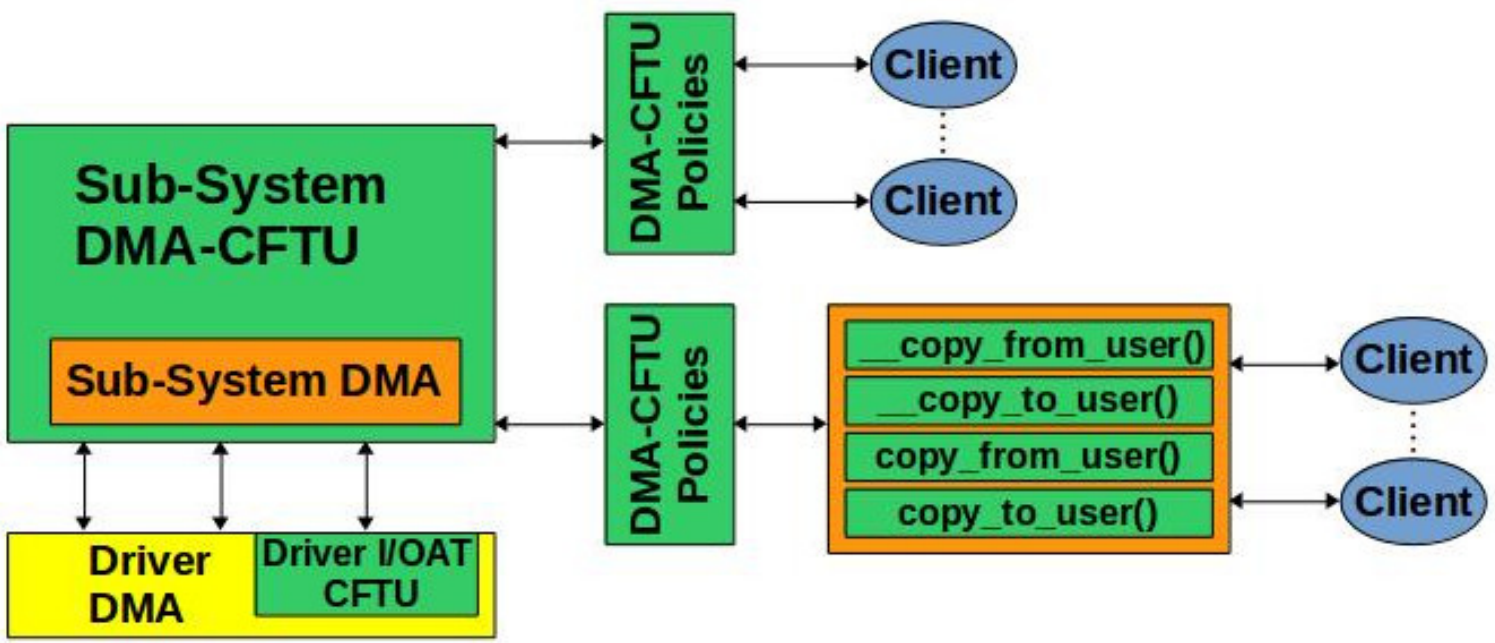
Copy memory support

- Generic interface to make DMA trasfers
- Kernel copy functions set-up
- Two Policies to manage the DMA channels:
  - *Exclusive Channel*: DMA channel is bound to one CPU (private channel)
  - *Shared Channel*: Every DMA channel is shared (public channel)

Real-time applications support

- Interface to create Real-time policies
- *Priority Channel* Policy

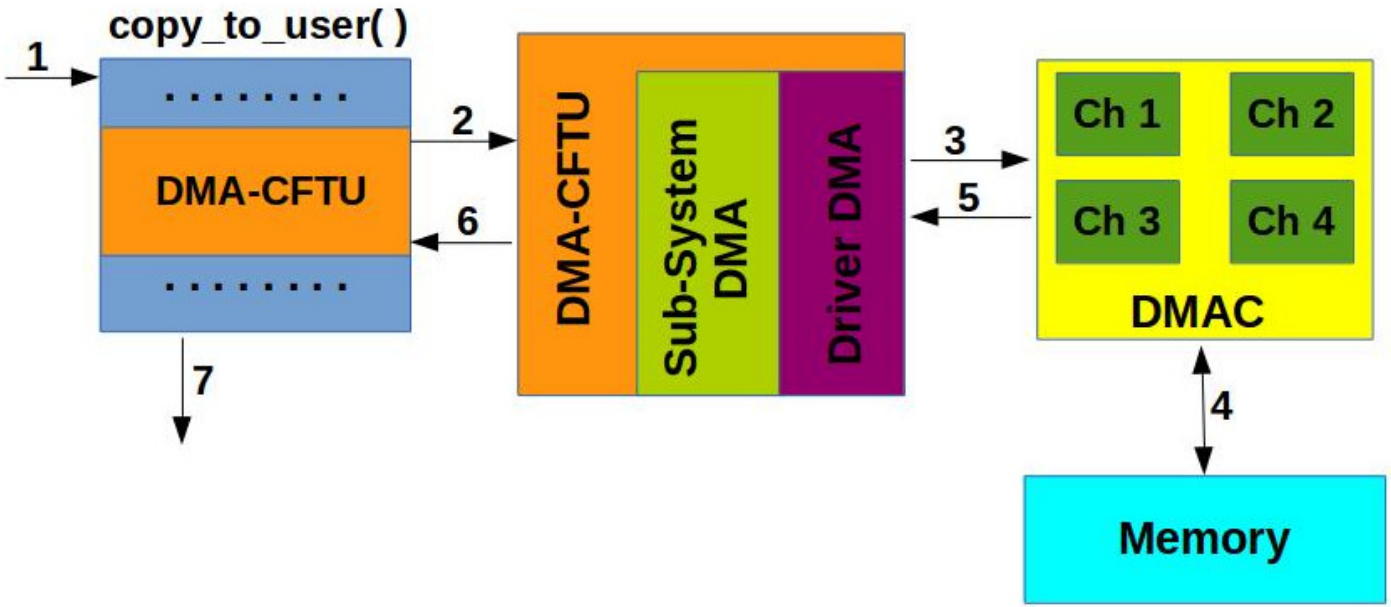
DMA-CFTU Extension(2)



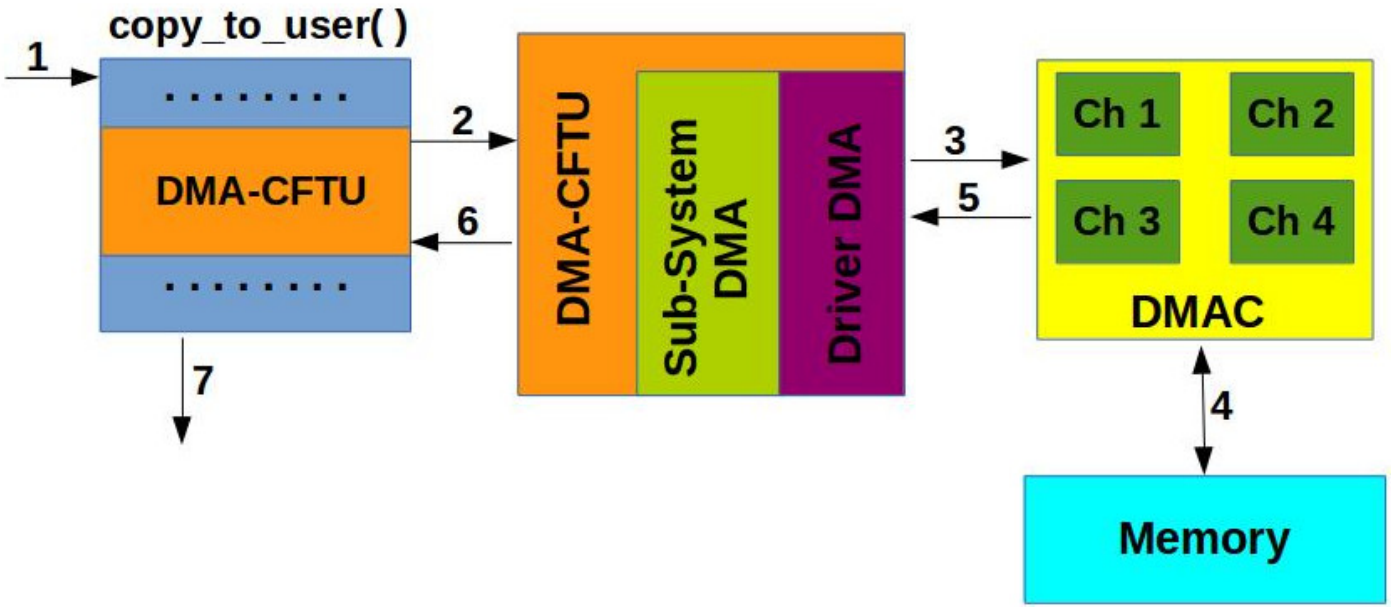
Components:

- Driver I/OAT-CFTU
- Sub-system DMA-CFTU
- Policies DMA-CFTU

copy_to_user() via DMA engine

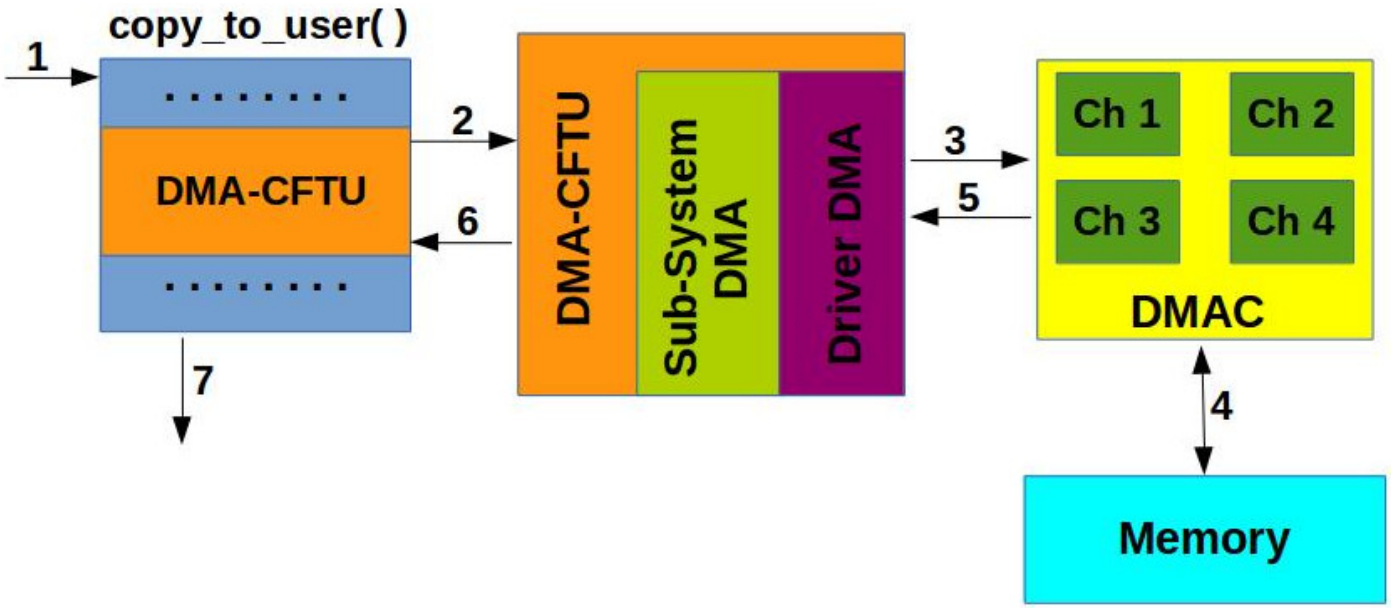


copy_to_user() via DMA engine



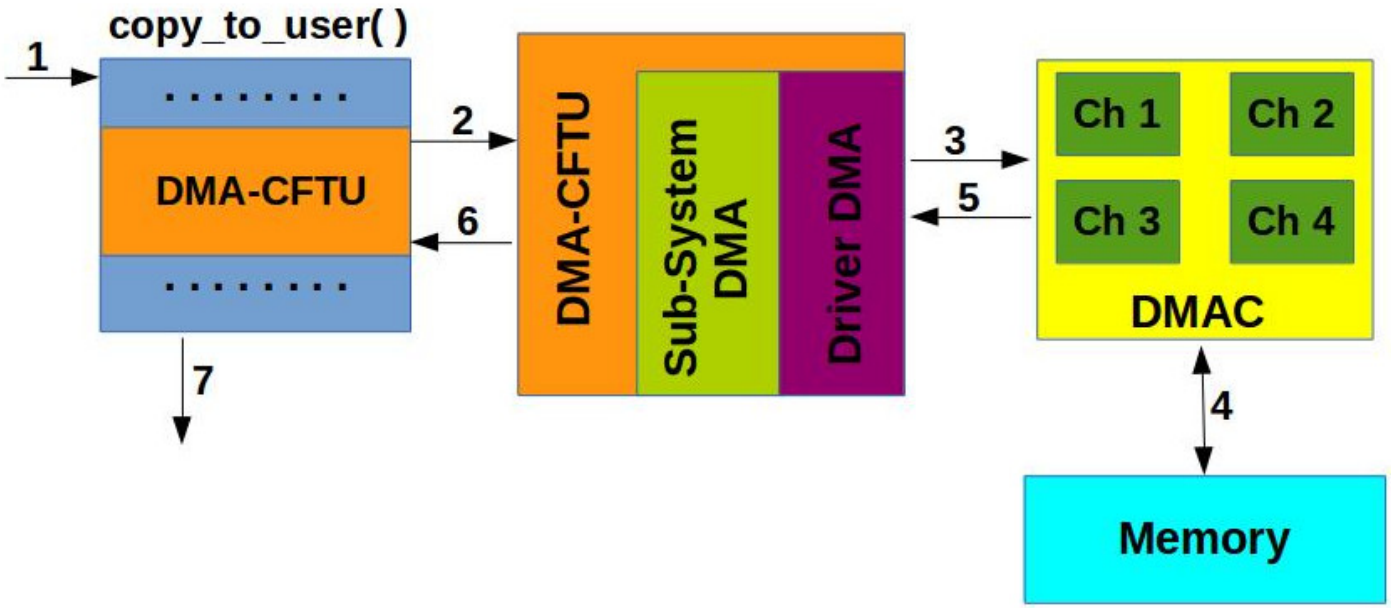
1 Kernel application calls the copy_to_user() function

copy_to_user() via DMA engine



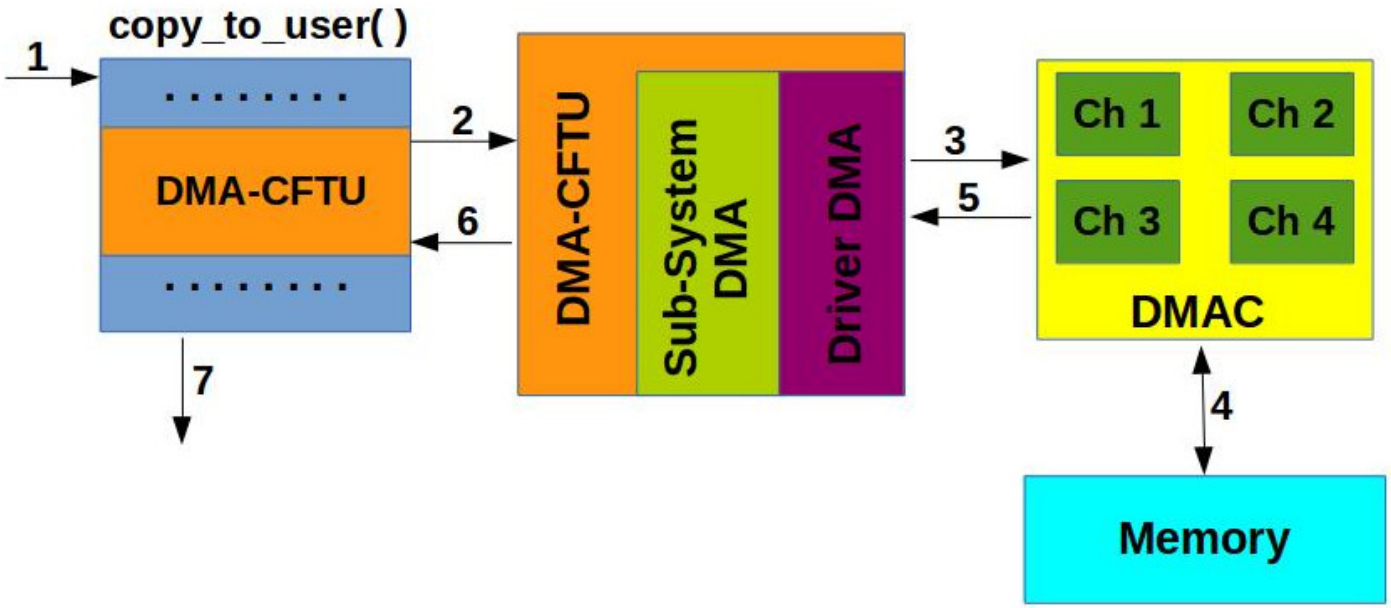
1. Kernel application calls the copy_to_user() function
2. The copy operation is submitted to DMA-CFU System

copy_to_user() via DMA engine



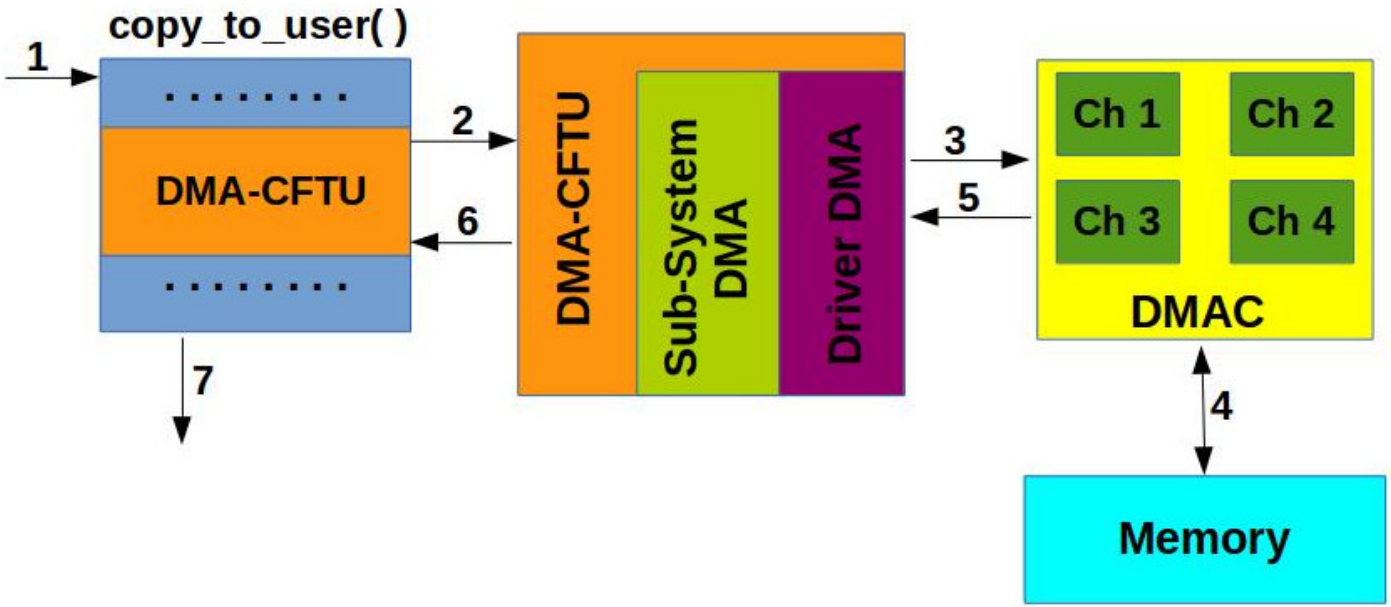
1. Kernel application calls the copy_to_user() function
2. The copy operation is submitted to DMA-CFU System
3. Set-up of DMA engine

copy_to_user() via DMA engine



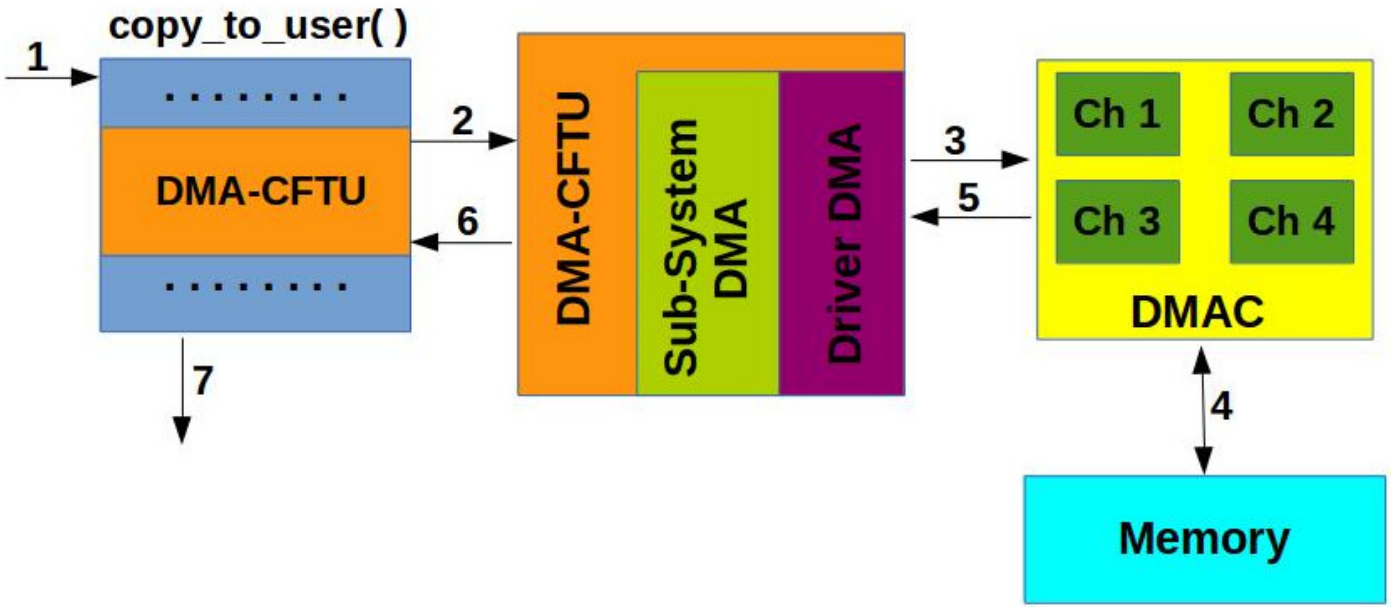
1. Kernel application calls the copy_to_user() function
2. The copy operation is submitted to DMA-CFU System
3. Set-up of DMA engine
4. The copy operation is performed via DMA engine

copy_to_user() via DMA engine



1. Kernel application calls the copy_to_user() function
2. The copy operation is submitted to DMA-CFU System
3. Set-up of DMA engine
4. The copy operation is performed via DMA engine
5. The DMA engine terminates and sends an interrupt signal

copy_to_user() via DMA engine



1 Kernel application calls the copy_to_user() function
2 The copy operation is submitted to DMA-CFU System
3 Set-up of DMA engine
4 The copy operation is performed via DMA engine
5 The DMA engine terminates and sends an interrupt signal
6-7 Return to copy_to_user() and termination

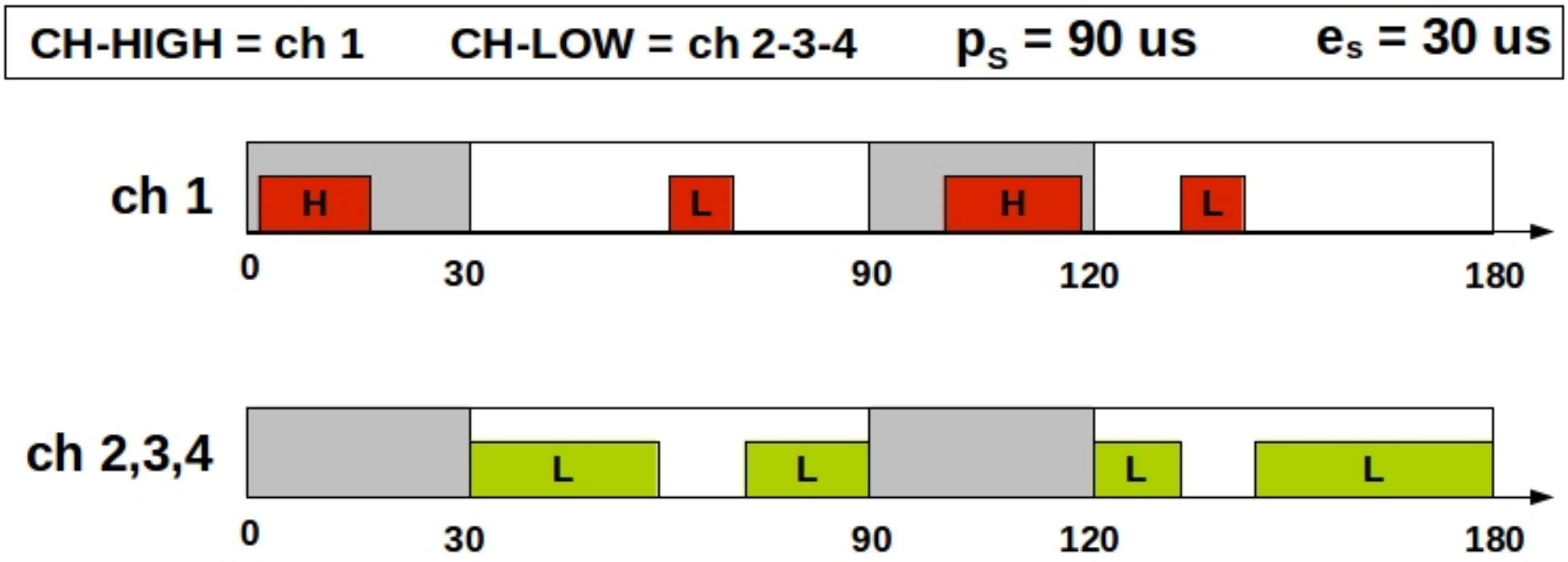
Priority Channel

- *CH-LOW* channels are suspended $\mathbf{e}_s$ every $\mathbf{p}_s$
- *CH-HIGH* channels use the DMA bus in a exclusive way $\mathbf{e}_s$ every $\mathbf{p}_s$

Priority Channel

- *CH-LOW* channels are suspended $\mathbf{e}_s$ every $\mathbf{p}_s$
- *CH-HIGH* channels use the DMA bus in a exclusive way $\mathbf{e}_s$ every $\mathbf{p}_s$

For example:



System Configuration

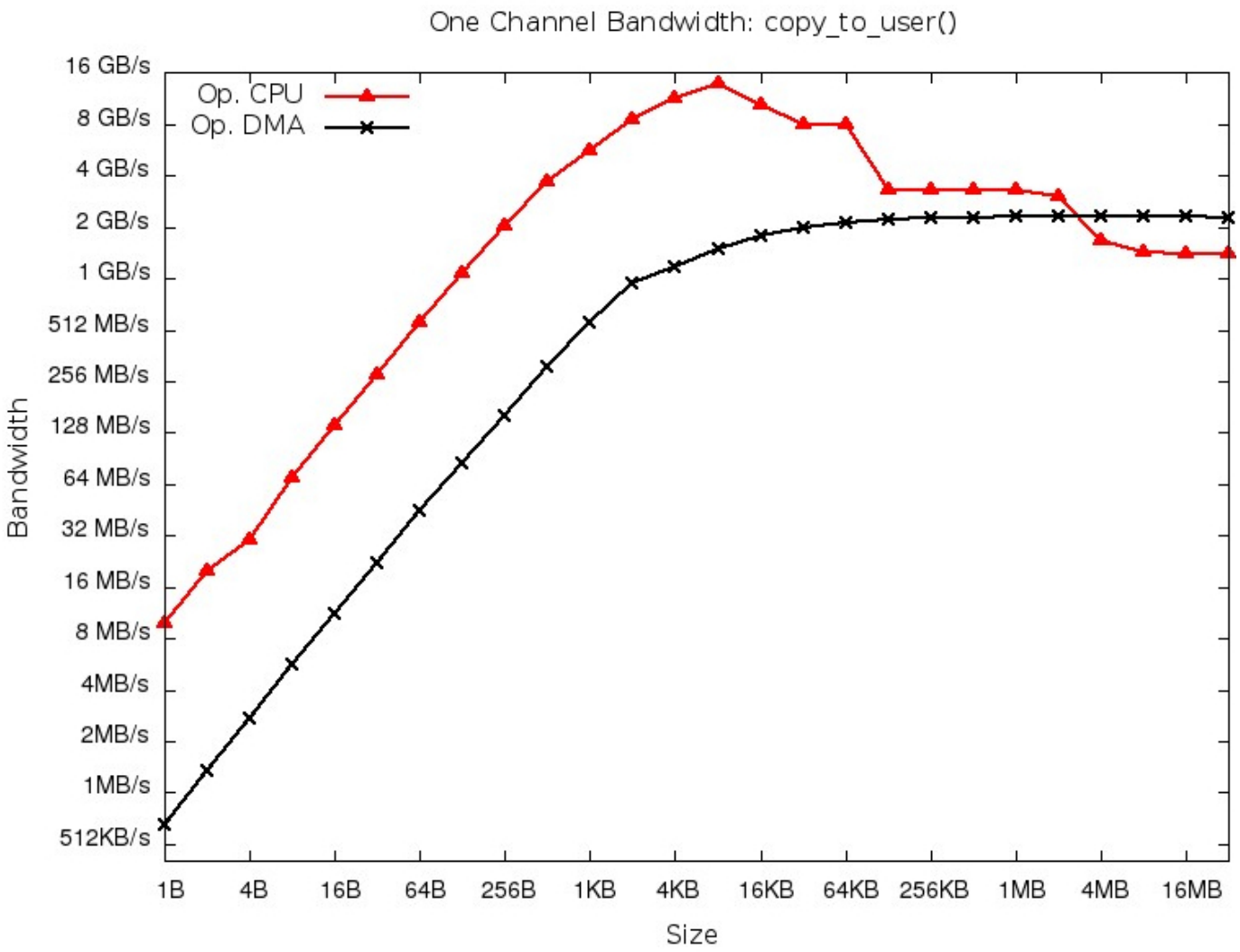
Intel's I/O Acceleration Technology (I/OAT)

- Asynchronous DMA Copy Engine all'interno dell' MCH
- Memory-Memory Data Transfer
- 4 DMA channels

System specification

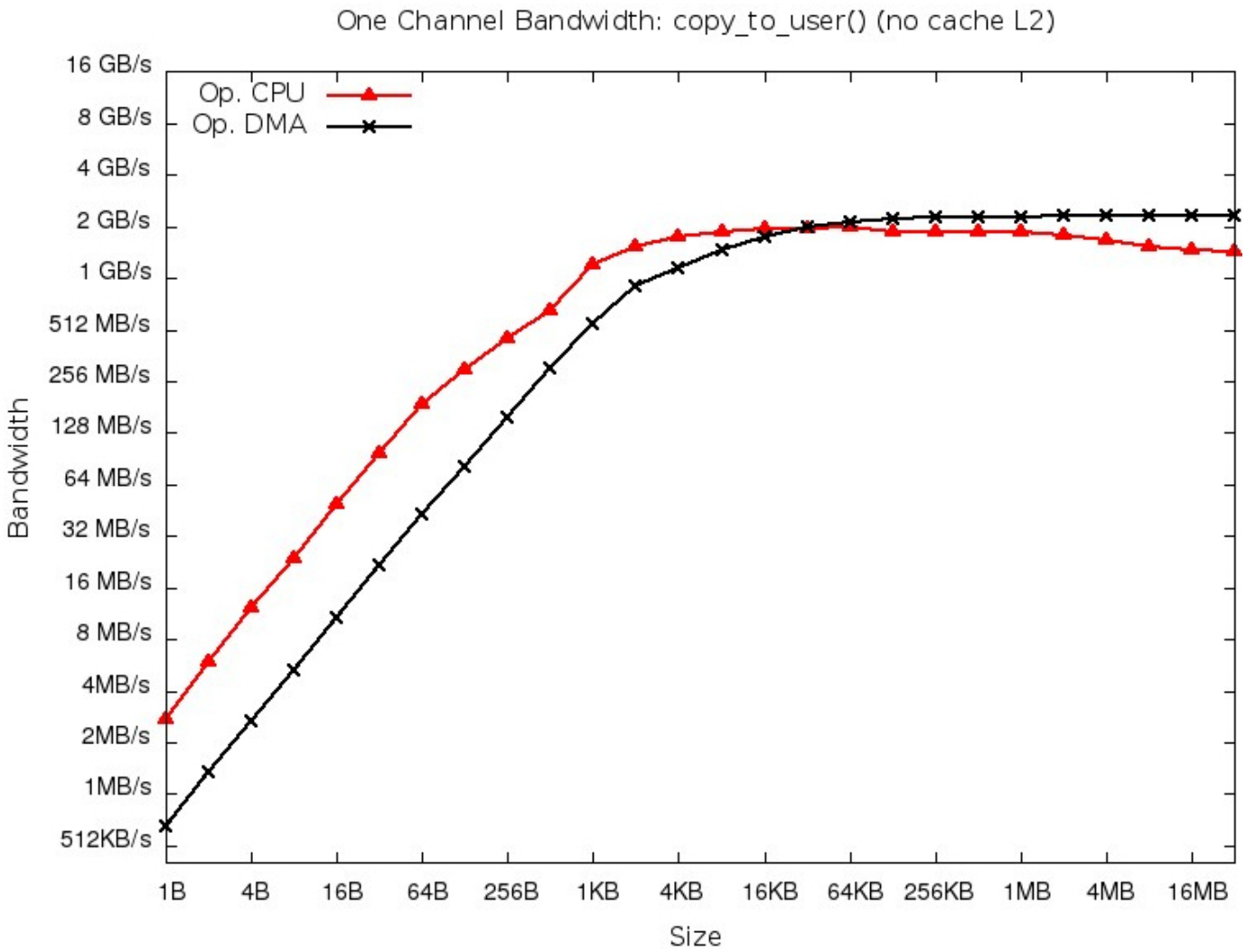
- Intel Xeon 5150 Dual-Core 2.66 GHZ (X2)
- Cache $L1_{d/i}$ 32KB, L2 4MB
- Memory RAM 4GB
- Intel 5000X Chipset MCH with DMA Engine
- Linux Slackware, Kernel version 3.9.2 (patch CFTU)

One Channel Bandwidth (1)



- Buffer $>$ 2MB, the DMA engine has better performance than CPU

One Channel Bandwidth: (2)

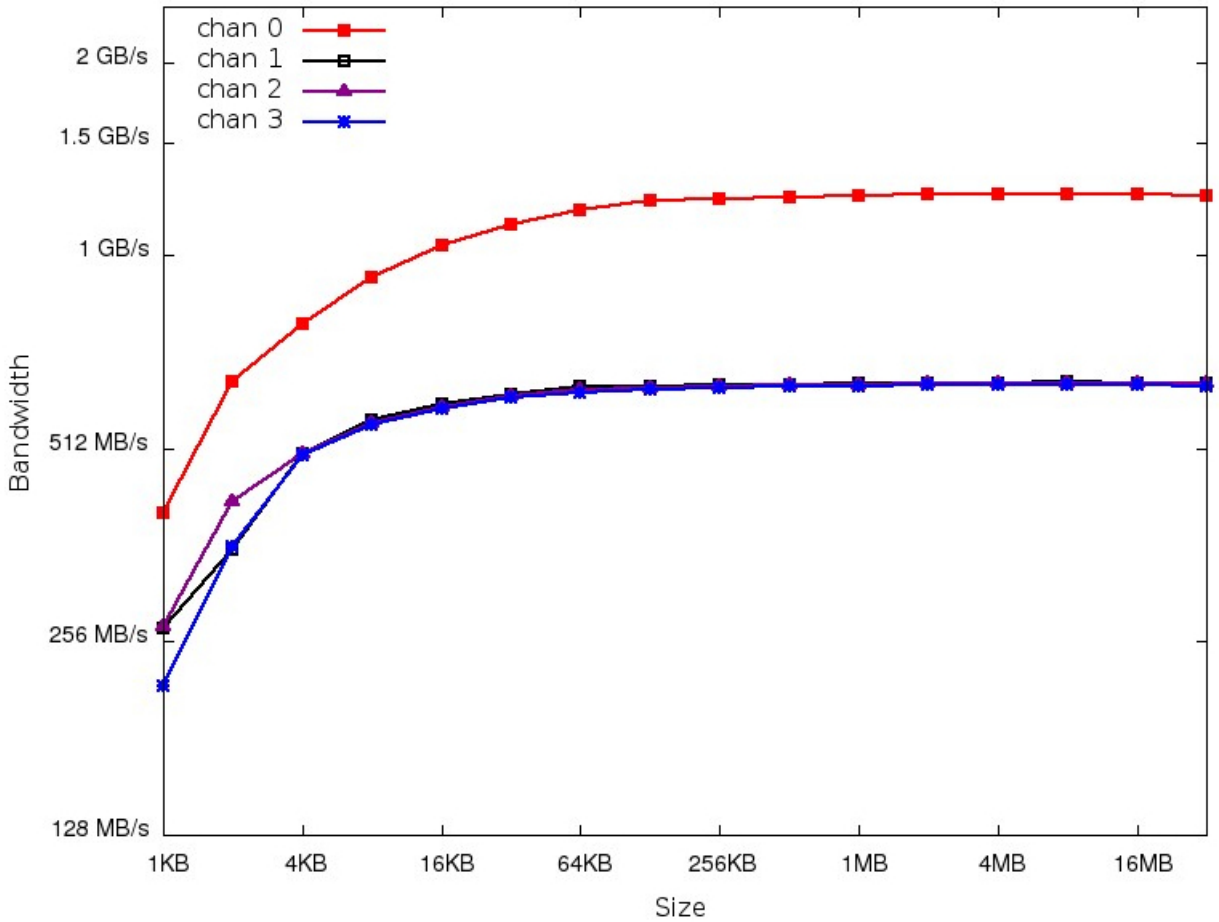


- Buffer $>$ 16 KB, the DMA engine has better performance than CPU

Priority Channel Bandwidth

Politica Priority Channel: $p_s = 90\mu s$, $e_s = 30\mu s$

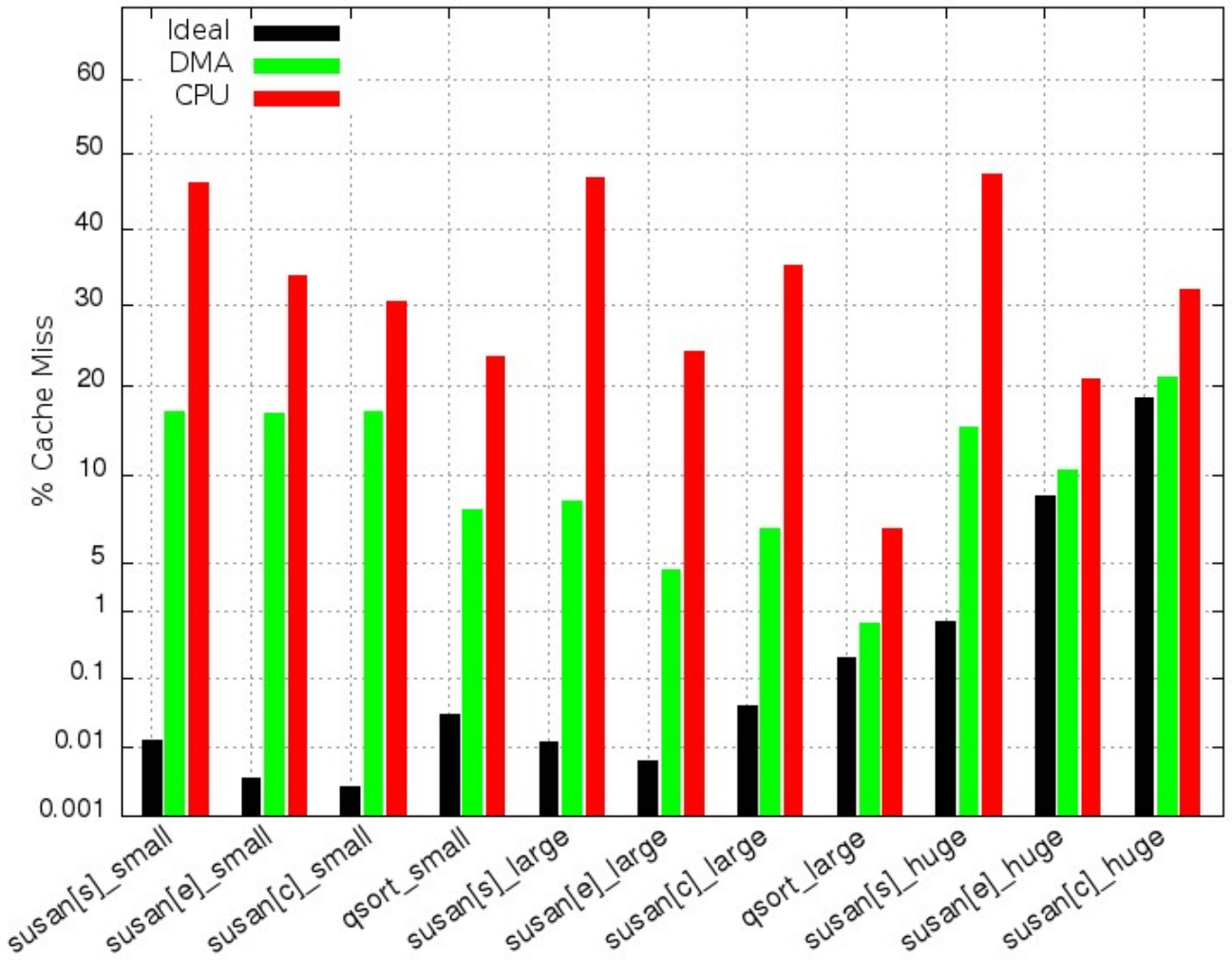
- *CH-HIGH* = chan0 e *CH-LOW* = chan1 , chan2, chan3



- Buffer $>$ 64 KB: chan0: $\sim$ 1.2 GB/s, chan1-2-3: $\sim$ 0.65 GB/s

Cache Pollution: 16064 Byte Data transfers

- To study the Cache Pollution problem, it has been used the tests of MiBench Benchmark



Conclusions and Future Works

Conclusions

- DMA engine performance are better than CPU if the data are not in cache memory
- The DMA transfer rate is not depend on cache memory
- It is possible to control the Cache pollution using the DMA transfer
- It is possible to define scheduling policies to manage the different priorities of DMA channels

Future Works

- New real-time policies
- Memcopy, Memset And XOR operations via DMA engine
- Processor DMA

Thank you for your attention!