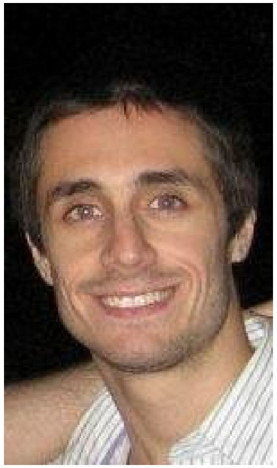
A dedicated kernel named **TORO**

Matias Vara Larsen

FOSDEM'15

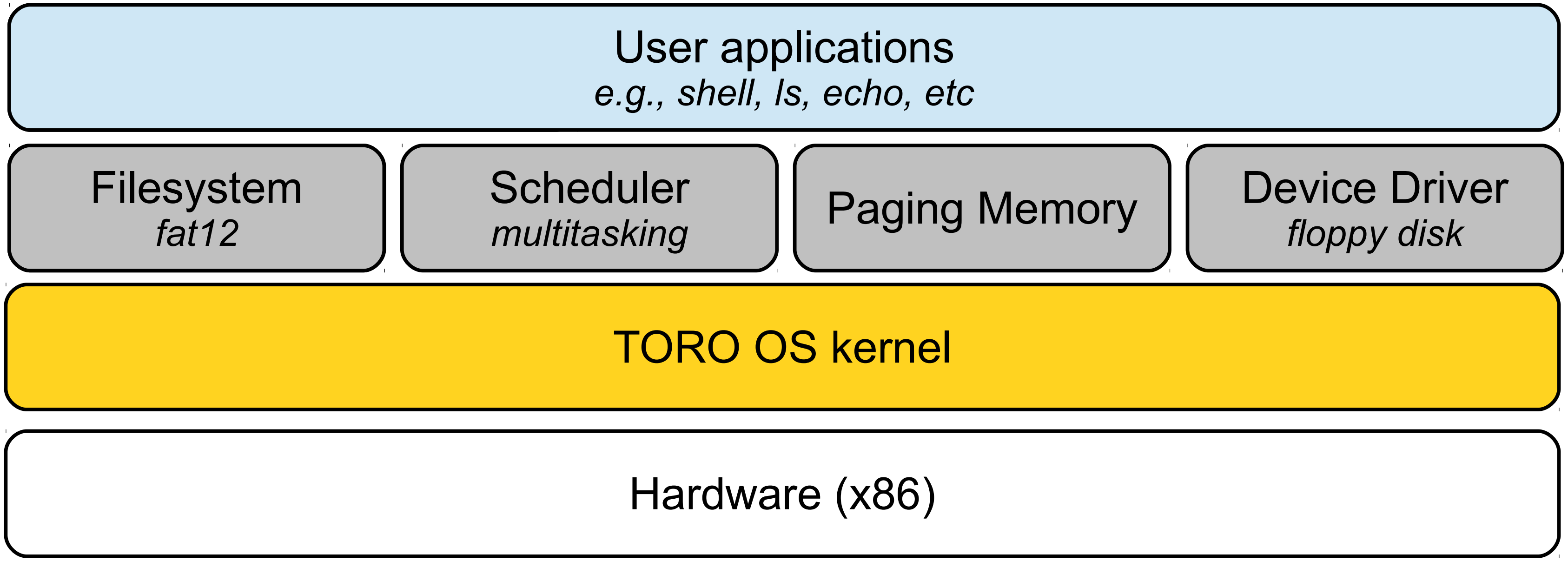
Who am I?



- Electronic Engineer from **Universidad Nacional de La Plata, Buenos Aires, Argentina**.
- PhD in Computer Science at **INRIA / CNRS, Nice, France** (finishing in 2015).
- I am the main (and the only ;)) developer of TORO

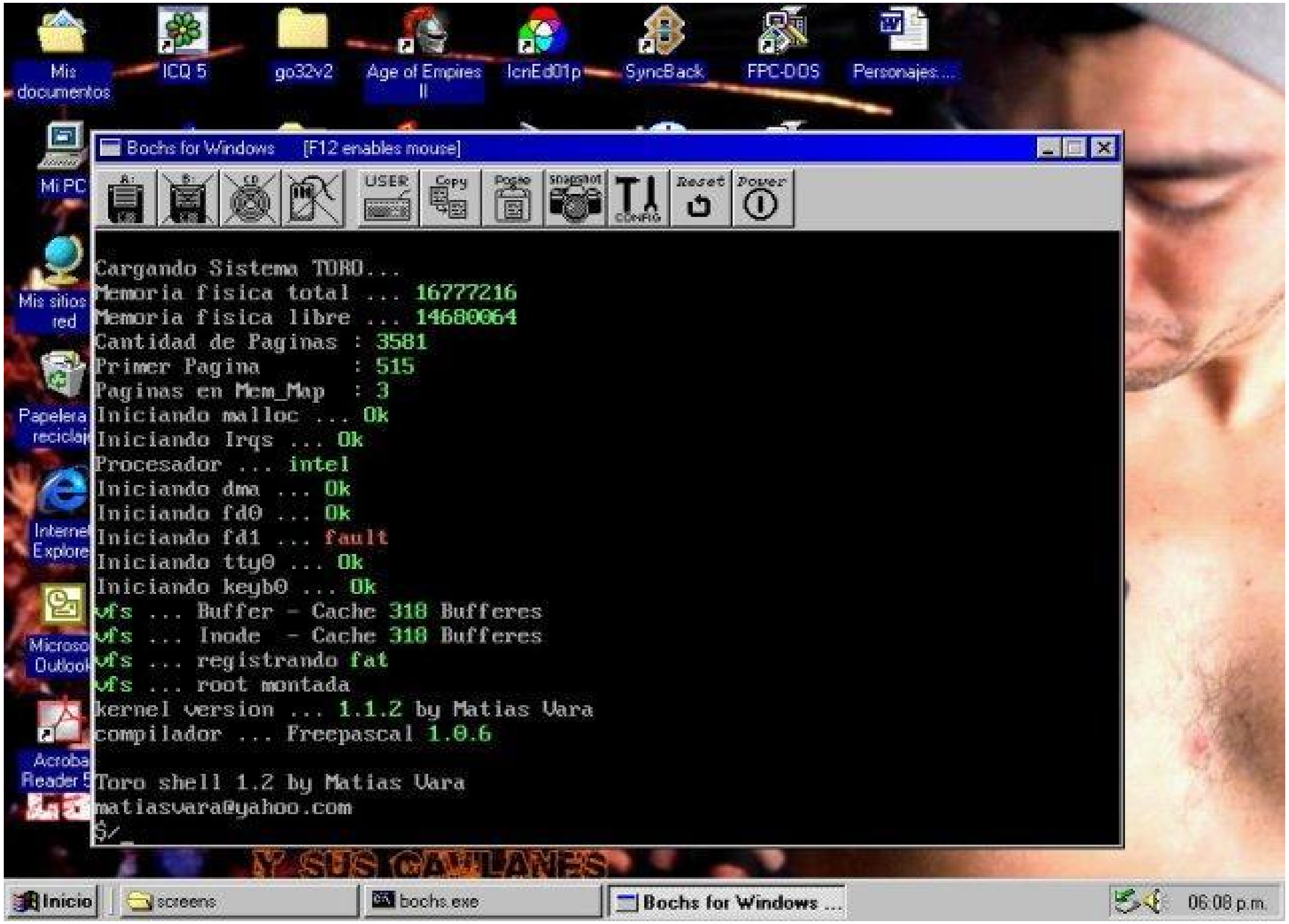
What is TORO OS?

- TORO OS started in 2003, and in 2004, I released the first stable version.



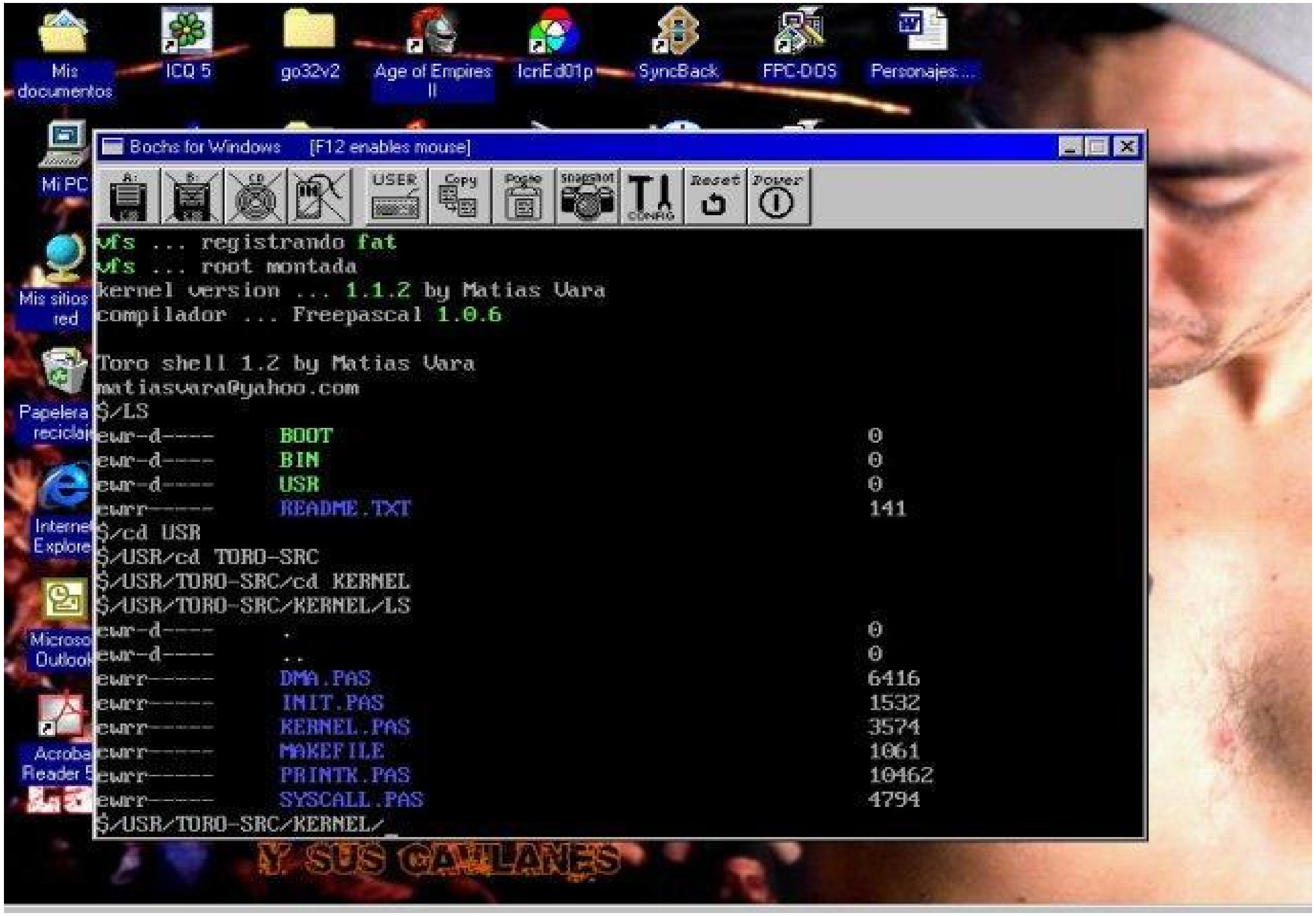
TORO – 1.1.3

TORO shell



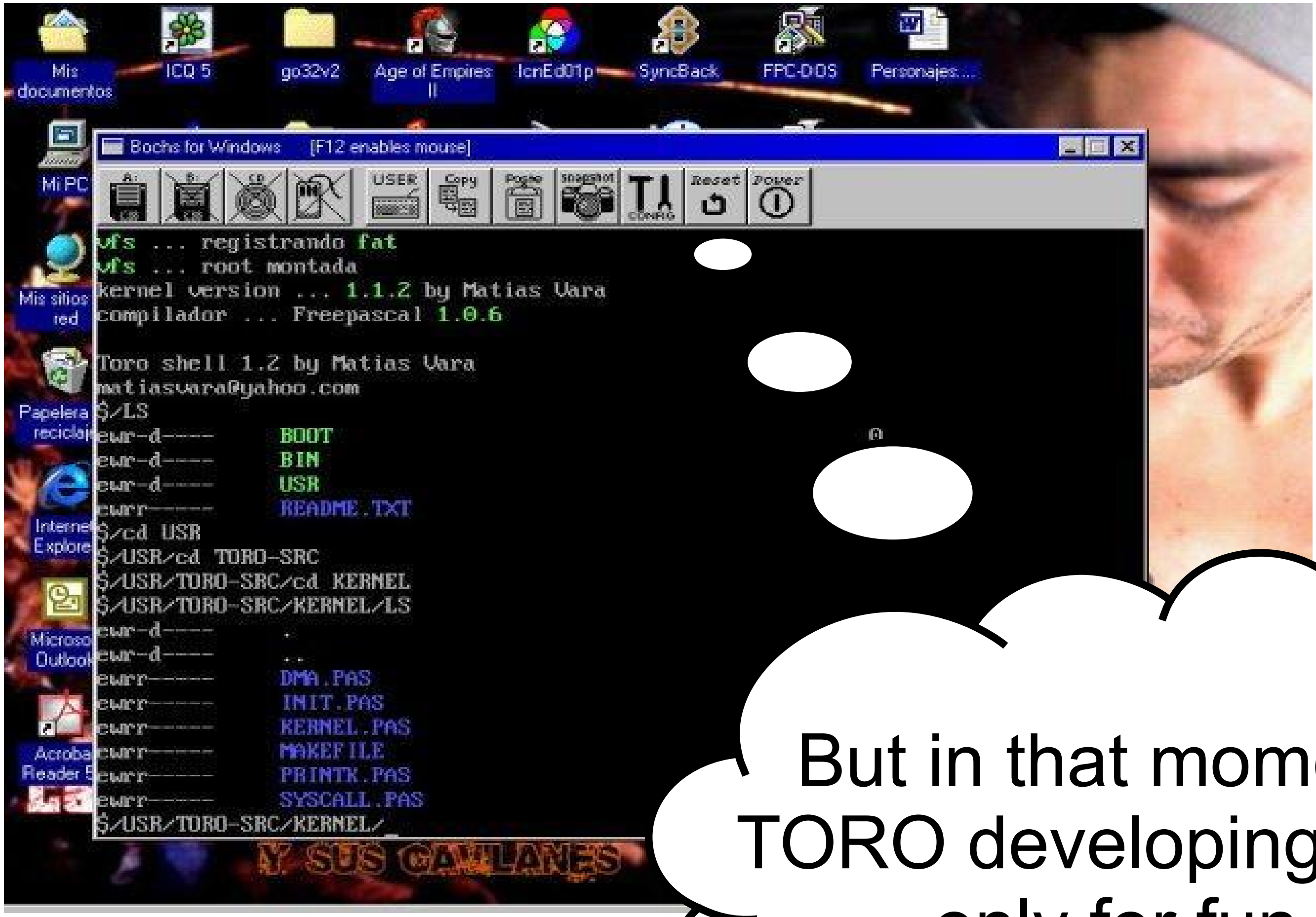
```
Cargando Sistema TORO...
Memoria fisica total ... 16777216
Memoria fisica libre ... 14680064
Cantidad de Paginas : 3581
Primer Pagina       : 515
Paginas en Mem_Map  : 3
Iniciando malloc ... Ok
Iniciando Irqs ... Ok
Procesador ... intel
Iniciando dma ... Ok
Iniciando fd0 ... Ok
Iniciando fd1 ... fault
Iniciando tty0 ... Ok
Iniciando keyb0 ... Ok
vfs ... Buffer - Cache 318 Bufferes
vfs ... Inode  - Cache 318 Bufferes
vfs ... registrando fat
vfs ... root montada
kernel version ... 1.1.2 by Matias Vara
compilador ... Freepascal 1.0.6

Toro shell 1.2 by Matias Vara
matiasvara@yahoo.com
$/_
```

LS



```
vfs ... registrando fat
vfs ... root montada
kernel version ... 1.1.2 by Matias Vara
compilador ... Freepascal 1.0.6

Toro shell 1.2 by Matias Vara
matiasvara@yahoo.com
$/LS
ewr-d----      BOOT                                              0
ewr-d----      BIN                                               0
ewr-d----      USR                                               0
ewrr-----      README.TXT                                        141
$/cd USR
$/USR/cd TORO-SRC
$/USR/TORO-SRC/cd KERNEL
$/USR/TORO-SRC/KERNEL/LS
ewr-d----      .                                                 0
ewr-d----      ..                                                0
ewrr-----      DMA.PAS                                           6416
ewrr-----      INIT.PAS                                          1532
ewrr-----      KERNEL.PAS                                        3574
ewrr-----      MAKEFILE                                          1061
ewrr-----      PRINTK.PAS                                        10462
ewrr-----      SYSCALL.PAS                                       4794
$/USR/TORO-SRC/KERNEL/_
```

LS



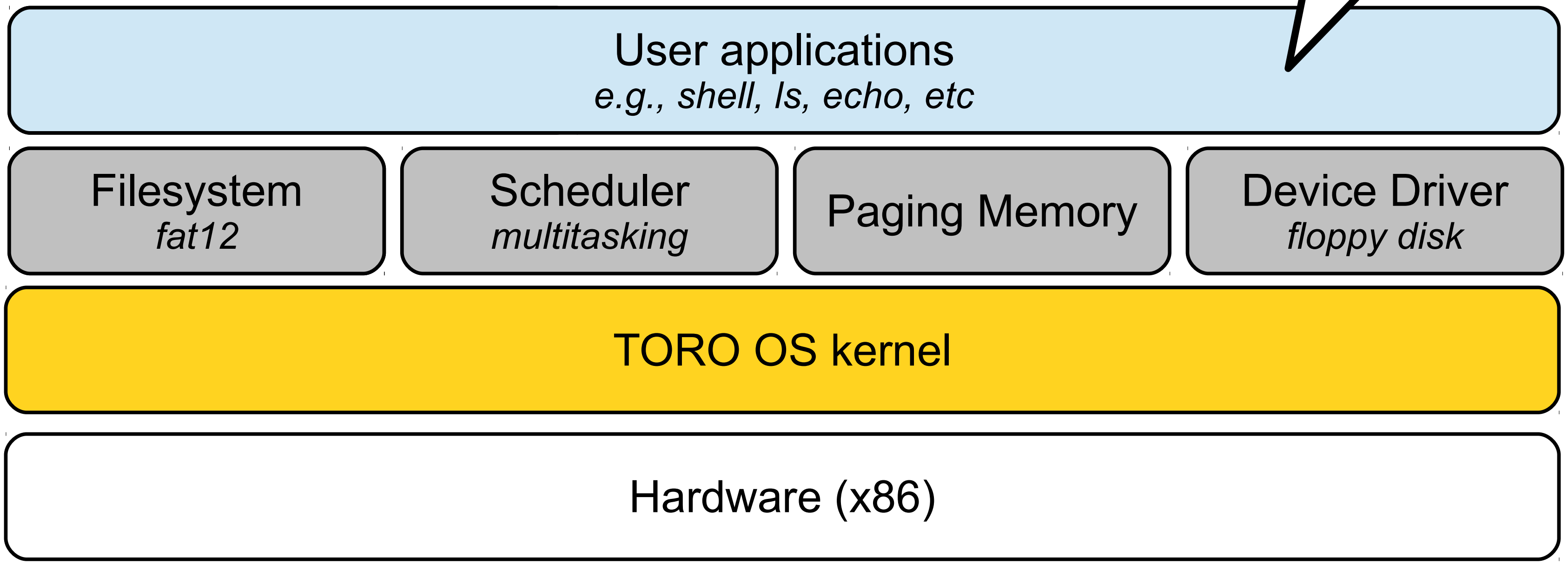
```
vfs ... registrando fat
vfs ... root montada
kernel version ... 1.1.2 by Matias Vara
compilador ... Freepascal 1.0.6

Toro shell 1.2 by Matias Vara
matiasvara@yahoo.com
$/LS
ewr-d----      BOOT
ewr-d----      BIN
ewr-d----      USR
ewrr-----      README.TXT
$/cd USR
$/USR/cd TORO-SRC
$/USR/TORO-SRC/cd KERNEL
$/USR/TORO-SRC/KERNEL/LS
ewr-d----     .
ewr-d----     ..
ewrr-----     DMA.PAS
ewrr-----     INIT.PAS
ewrr-----     KERNEL.PAS
ewrr-----     MAKEFILE
ewrr-----     PRINTK.PAS
ewrr-----     SYSCALL.PAS
$/USR/TORO-SRC/KERNEL/_
```

But in that moment, TORO developing was only for fun

What is TO

- TORO OS starts in 2003, released the first stable version.

How we can optimize a ***general purpose kernel*** for a given purpose? i.e., ***application-oriented***



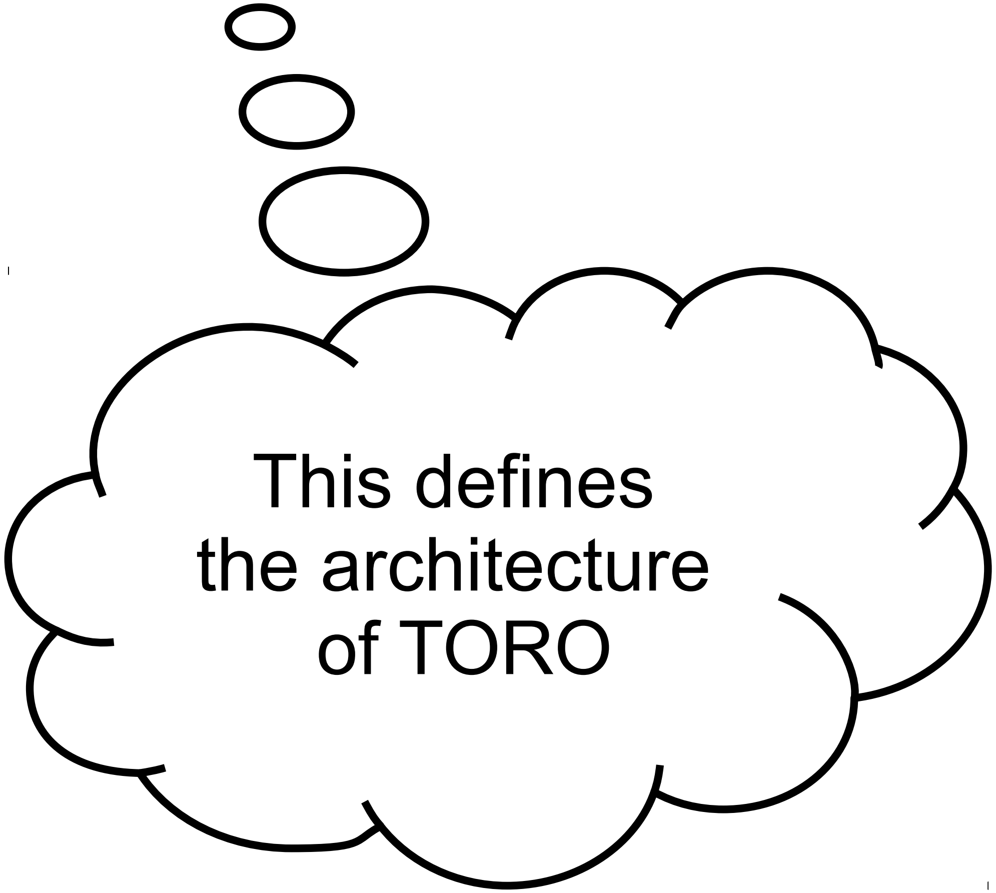
TORO – 1.1.3

What is TORO kernel?

- In 2006, the kernel is optimized to run a ***single user application*** in a ***multicore environment***

What is TORO kernel?

- In 2006, the kernel is optimized to run a ***single user application*** in a ***multicore environment***



This defines
the architecture
of TORO

What is TORO kernel?

- In 2006, the kernel is optimized to run a ***single user application*** in a ***multicore environment***

TORO integrates the user application with the kernel, and dedicates resources to a given core ***e.g.***, memory, devices and so on

This defines the architecture of TORO

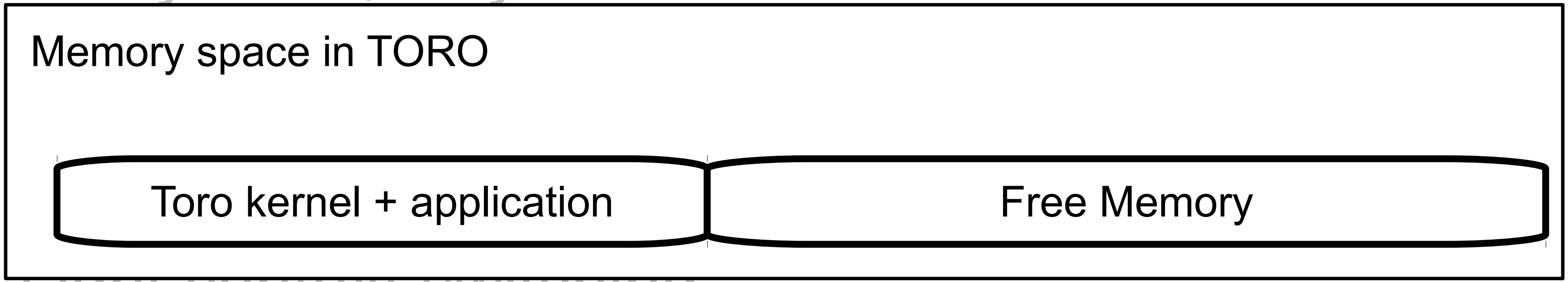
Kernel + user application

- Only ring 0
- The application is compiled with the kernel
- No syscalls, only calls.
- Threads instead of process
- Flat memory, no pagination
- Light context switching

- In this sense, TORO is a ***library OS-like designing***.

Kernel + user application

- Only ring 0
- The application is compiled with the kernel
- No syscalls, only calls.



Memory space in TORO

Toro kernel + application

Free Memory

- Light context switching

- In this sense, TORO is a ***library OS-like designing***.

Dedicated Resources

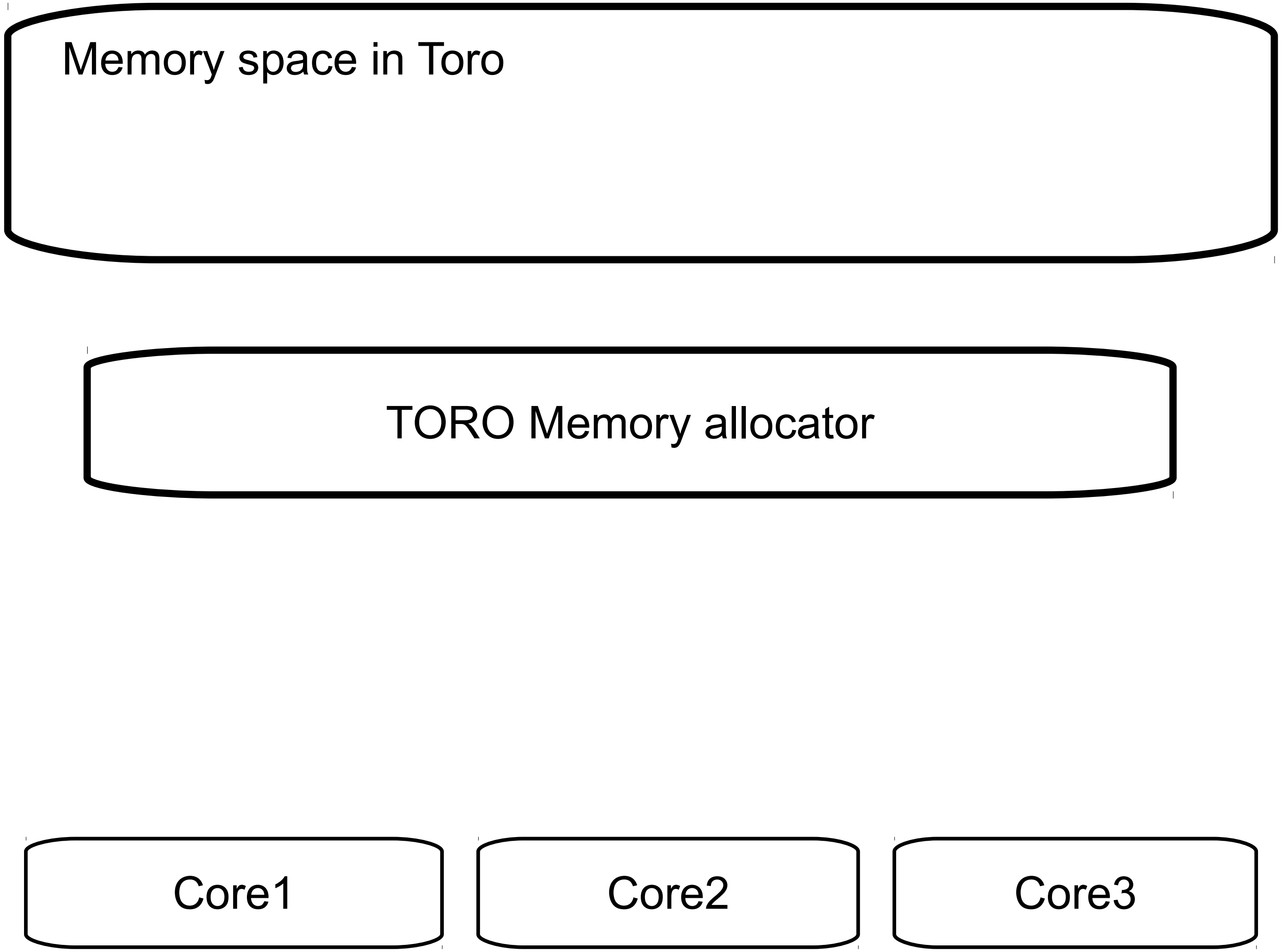
- In a **multicore** system the problematic resource is the ***shared memory***.
- The use of shared memory causes:
  - Overhead in the memory bus.
  - Overhead in the cache to keep it coherent.
  - Overhead in spin locks for mutual exclusion.

Dedicated Resources

- In a **multicore** system the problematic resource is the ***shared memory***.
- The use of shared memory causes:
  - Overhead in the memory bus.
  - Overhead in the cache to keep it coherent.
  - Overhead in spin locks for mutual exclusion.

TORO tries to avoid
these problems by keeping
all the *resources locals*, e.g., memory, filesystem

Dedicated Memory Allocation



Memory space in Toro

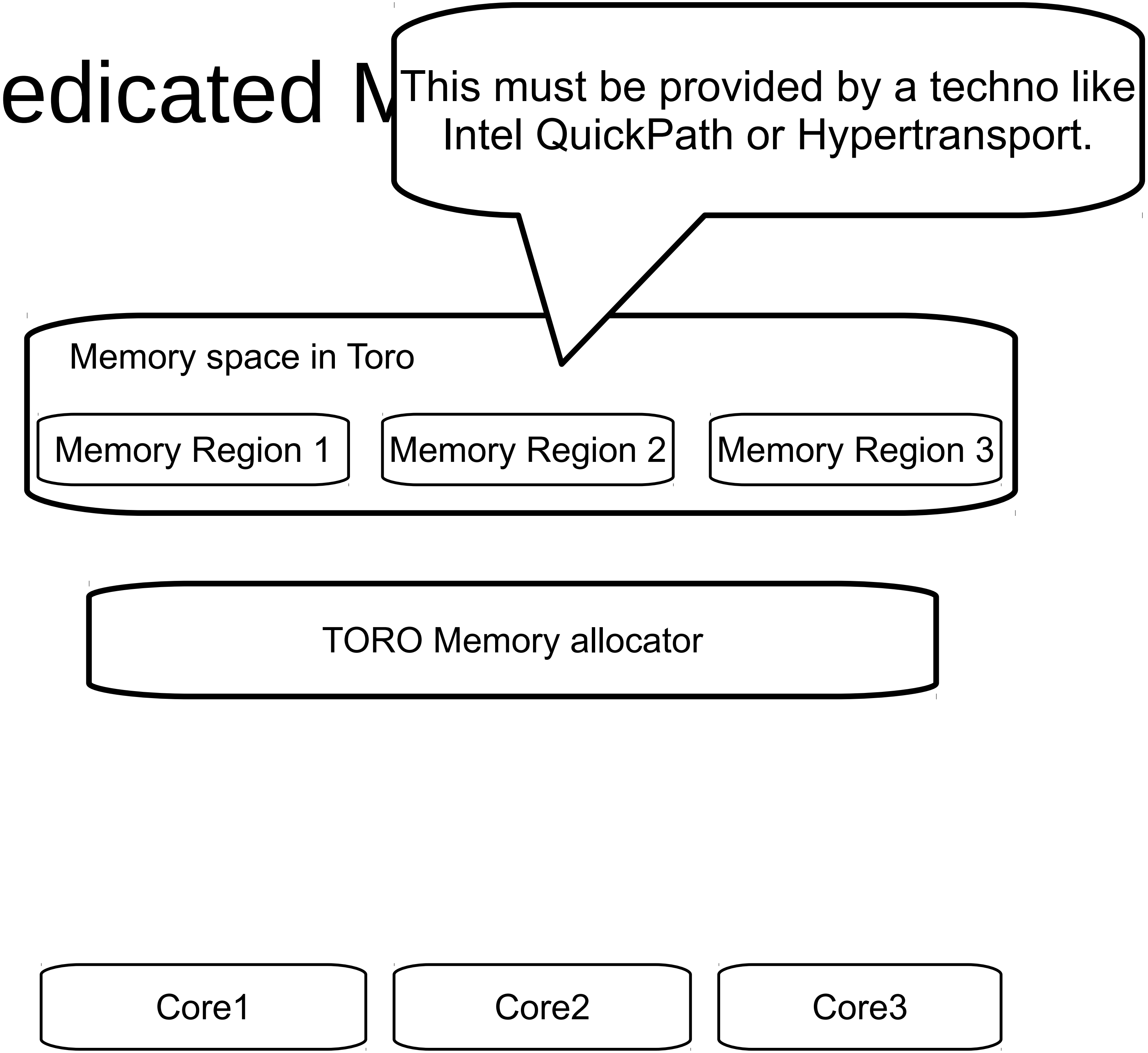
TORO Memory allocator

Core1

Core2

Core3

Dedicated M



This must be provided by a techno like Intel QuickPath or Hypertransport.

Memory space in Toro

Memory Region 1

Memory Region 2

Memory Region 3

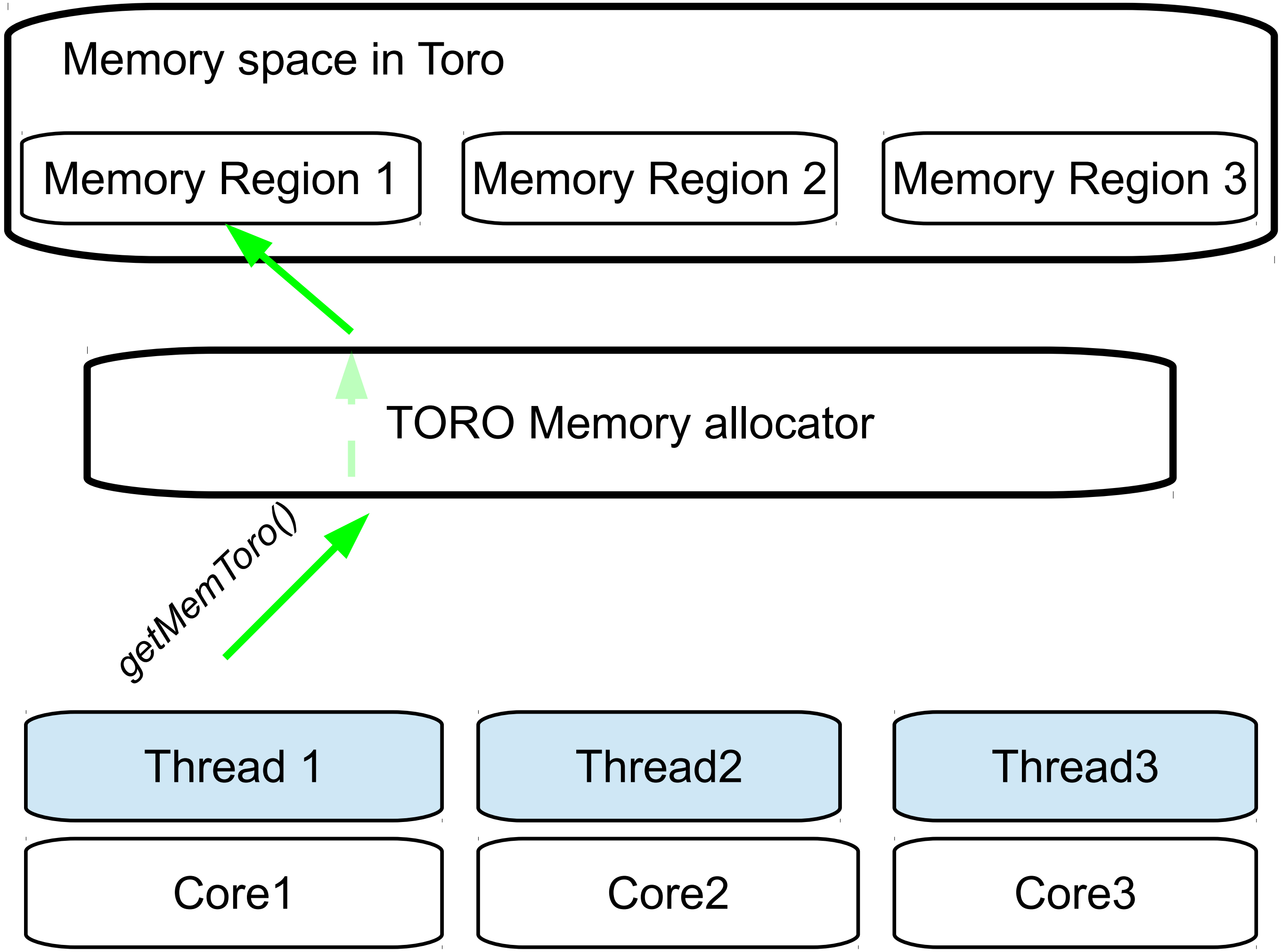
TORO Memory allocator

Core1

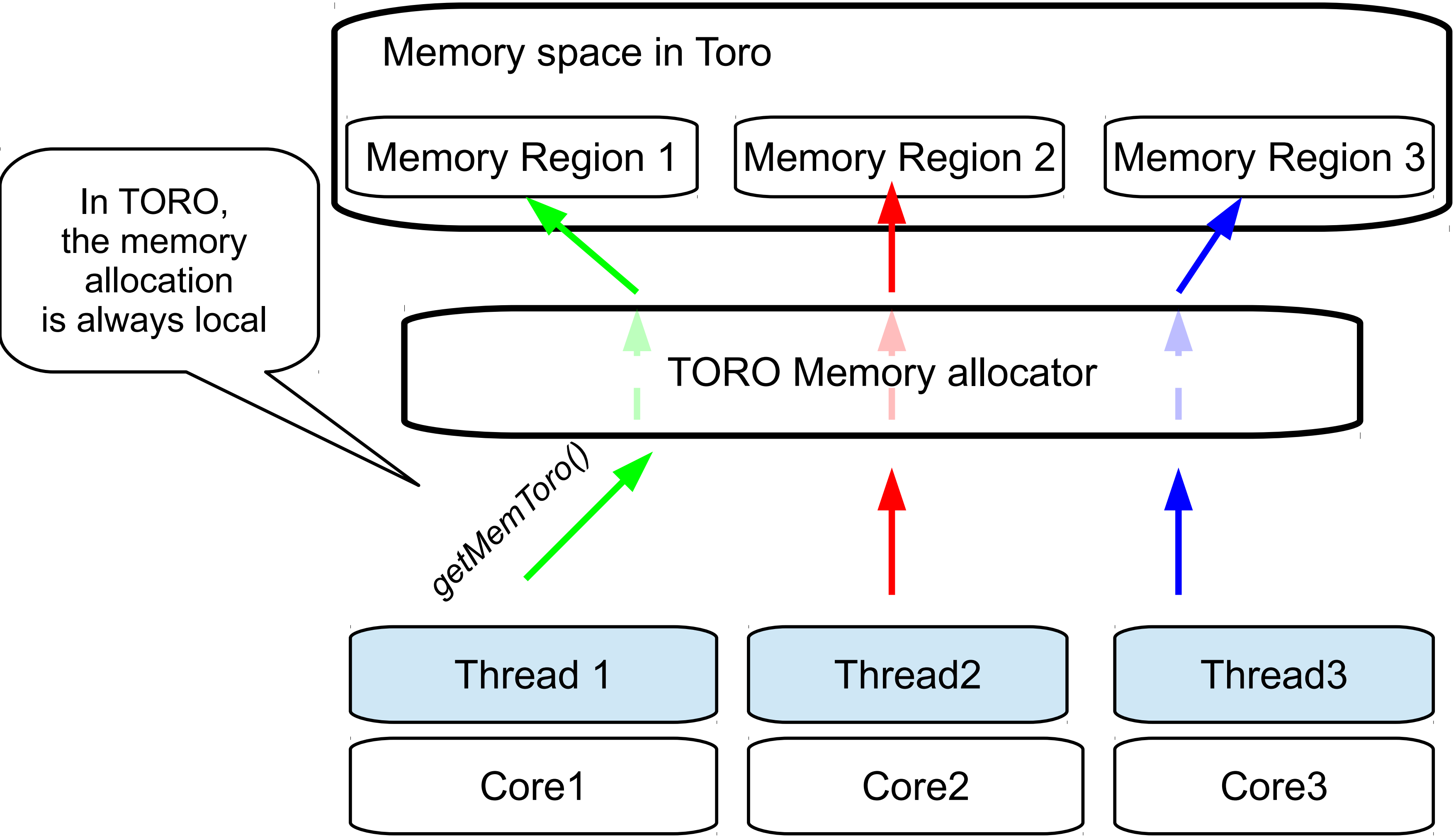
Core2

Core3

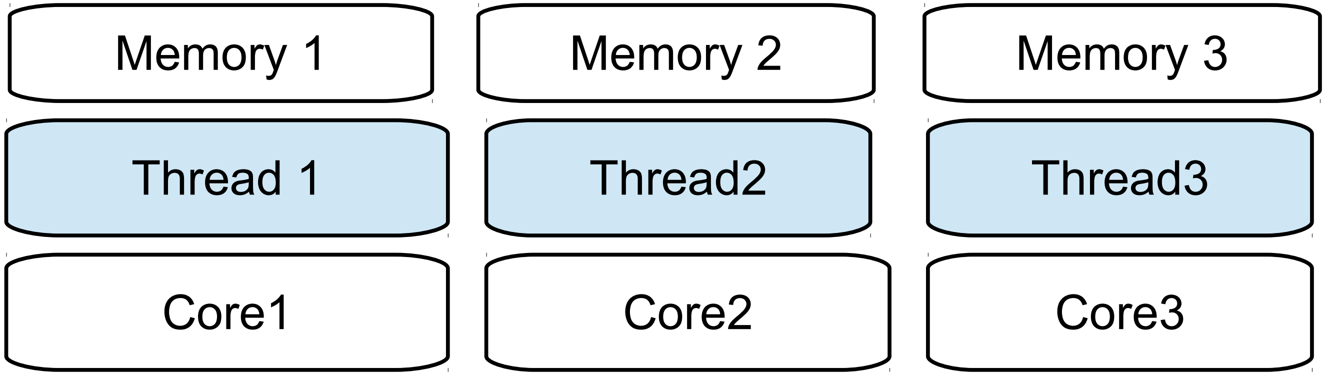
Dedicated Memory Allocation



Dedicated Memory Allocation



Locality of memory



Memory 1

Memory 2

Memory 3

Thread 1

Thread2

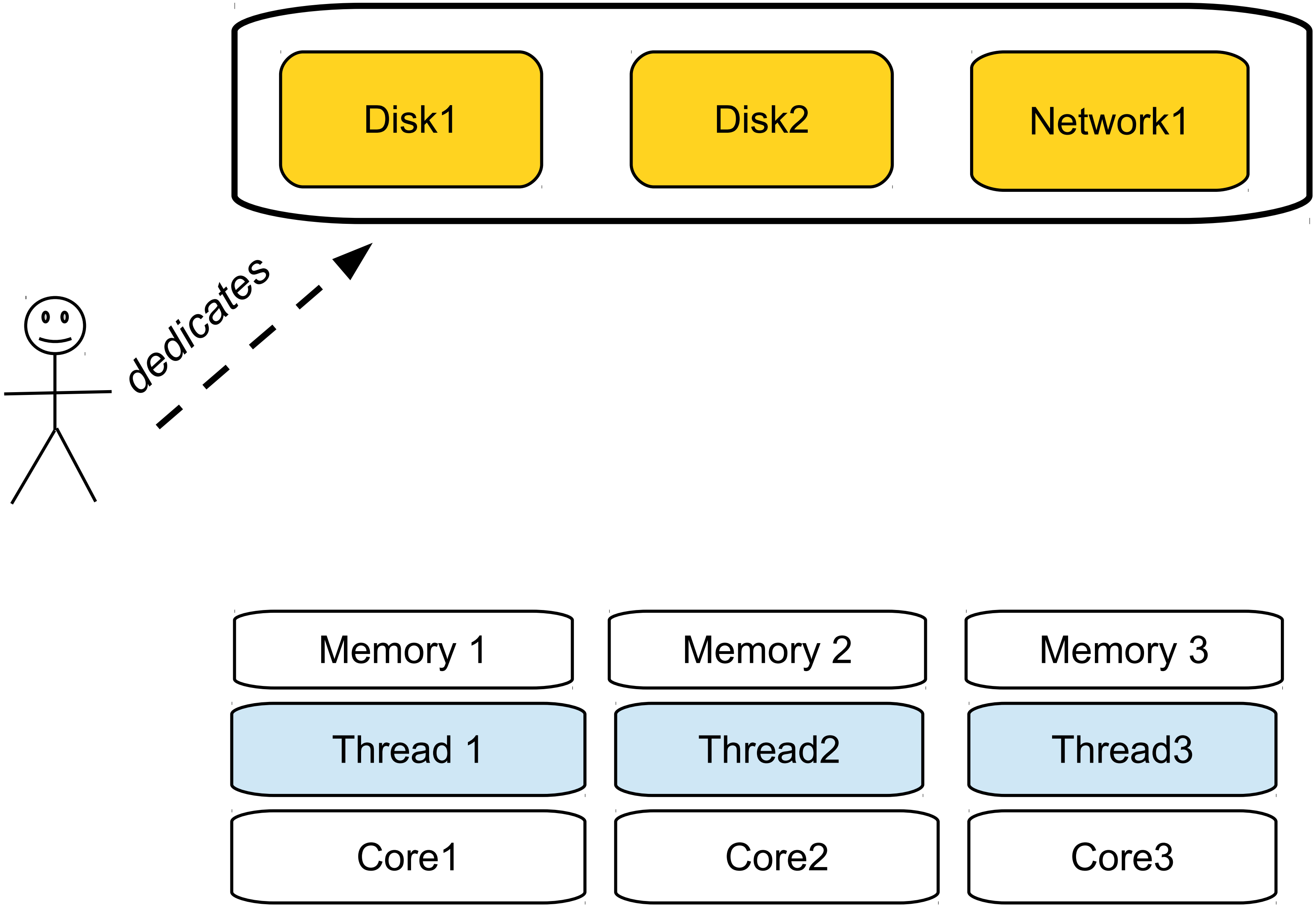
Thread3

Core1

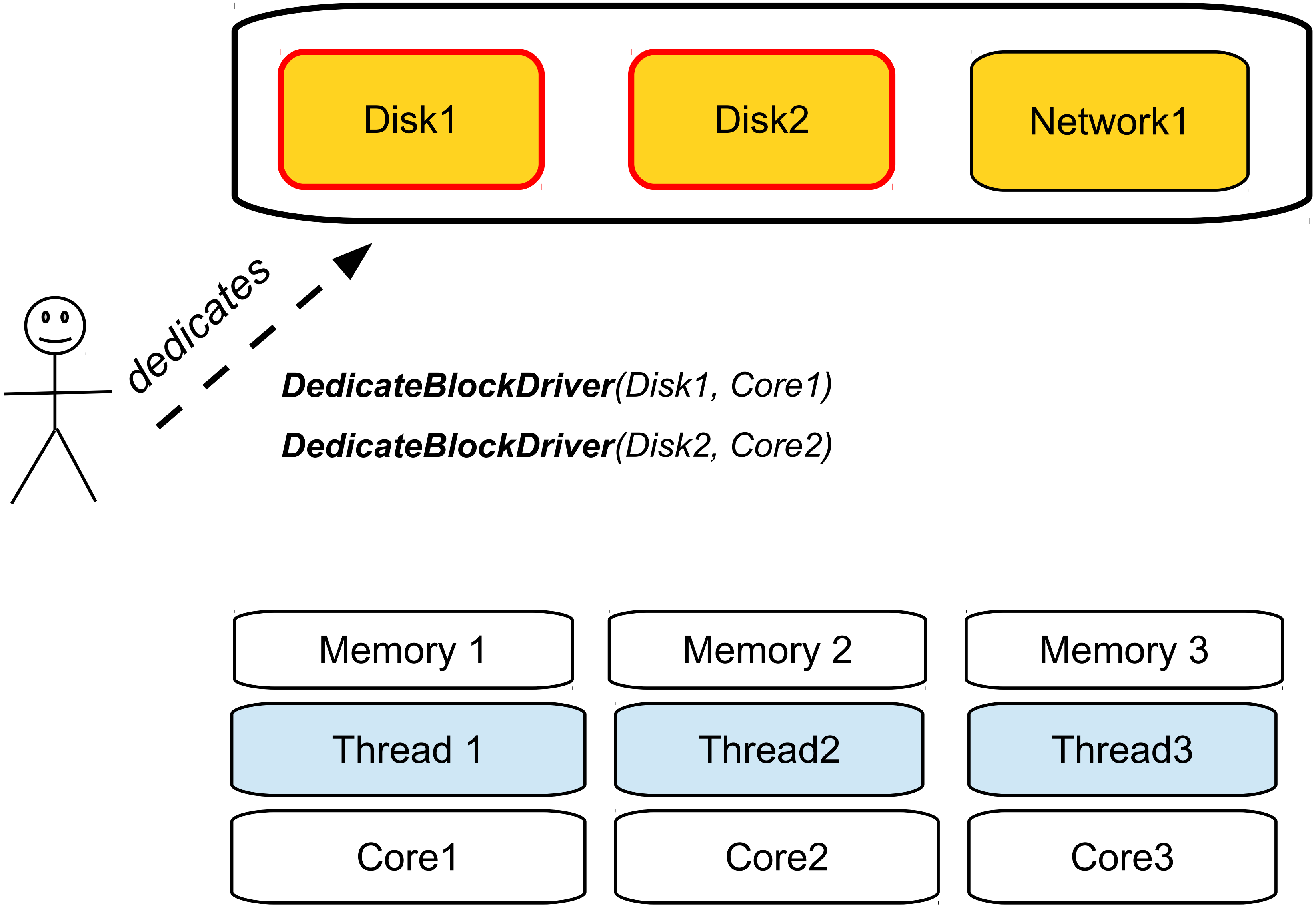
Core2

Core3

Locality of resources



Locality of resources



Disk1

Disk2

Network1

dedicates

DedicateBlockDriver*(Disk1, Core1)*

DedicateBlockDriver*(Disk2, Core2)*

Memory 1

Memory 2

Memory 3

Thread 1

Thread2

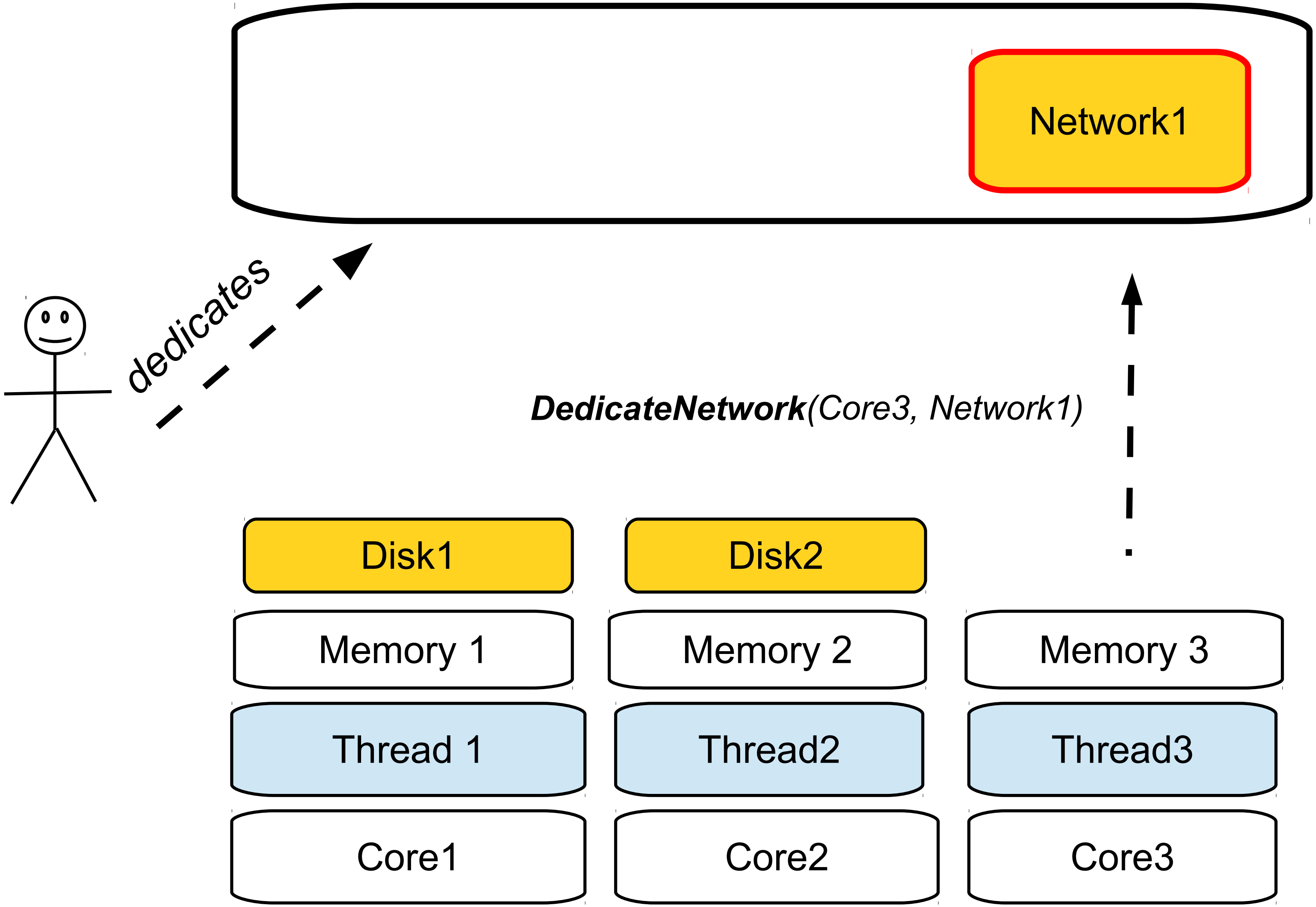
Thread3

Core1

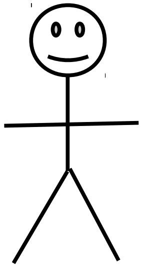
Core2

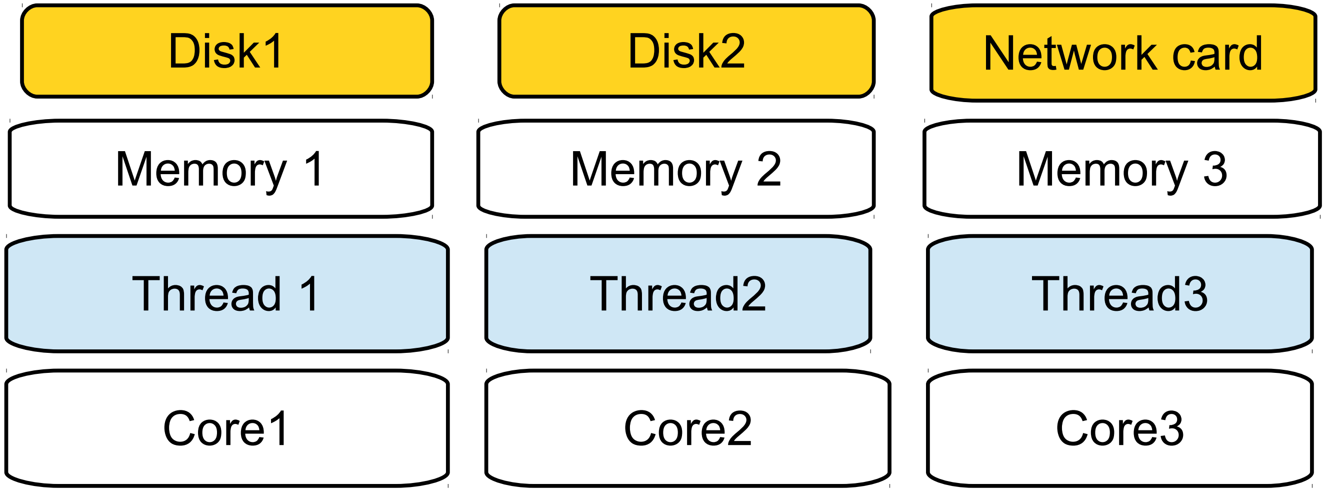
Core3

Locality of resources



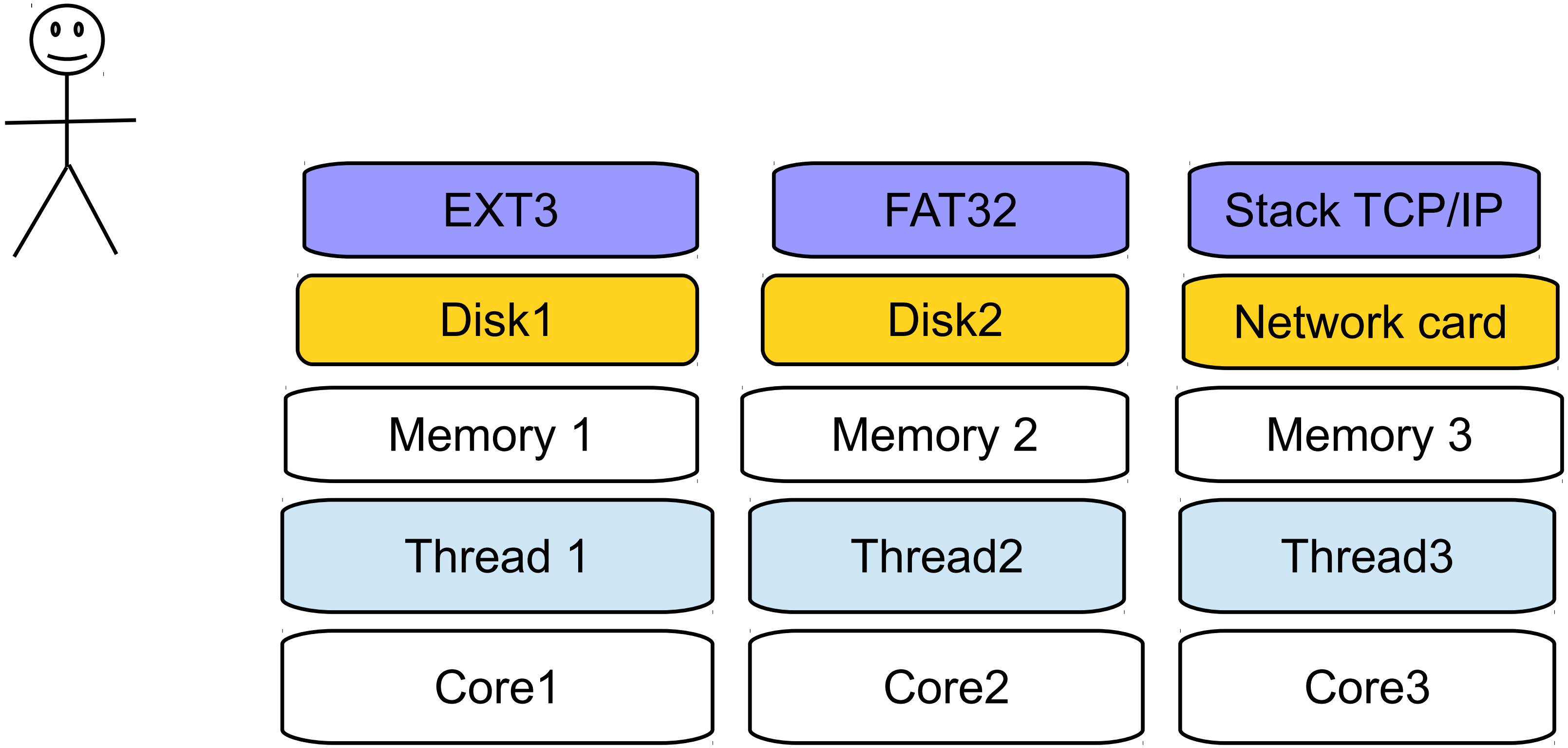
Locality of resources





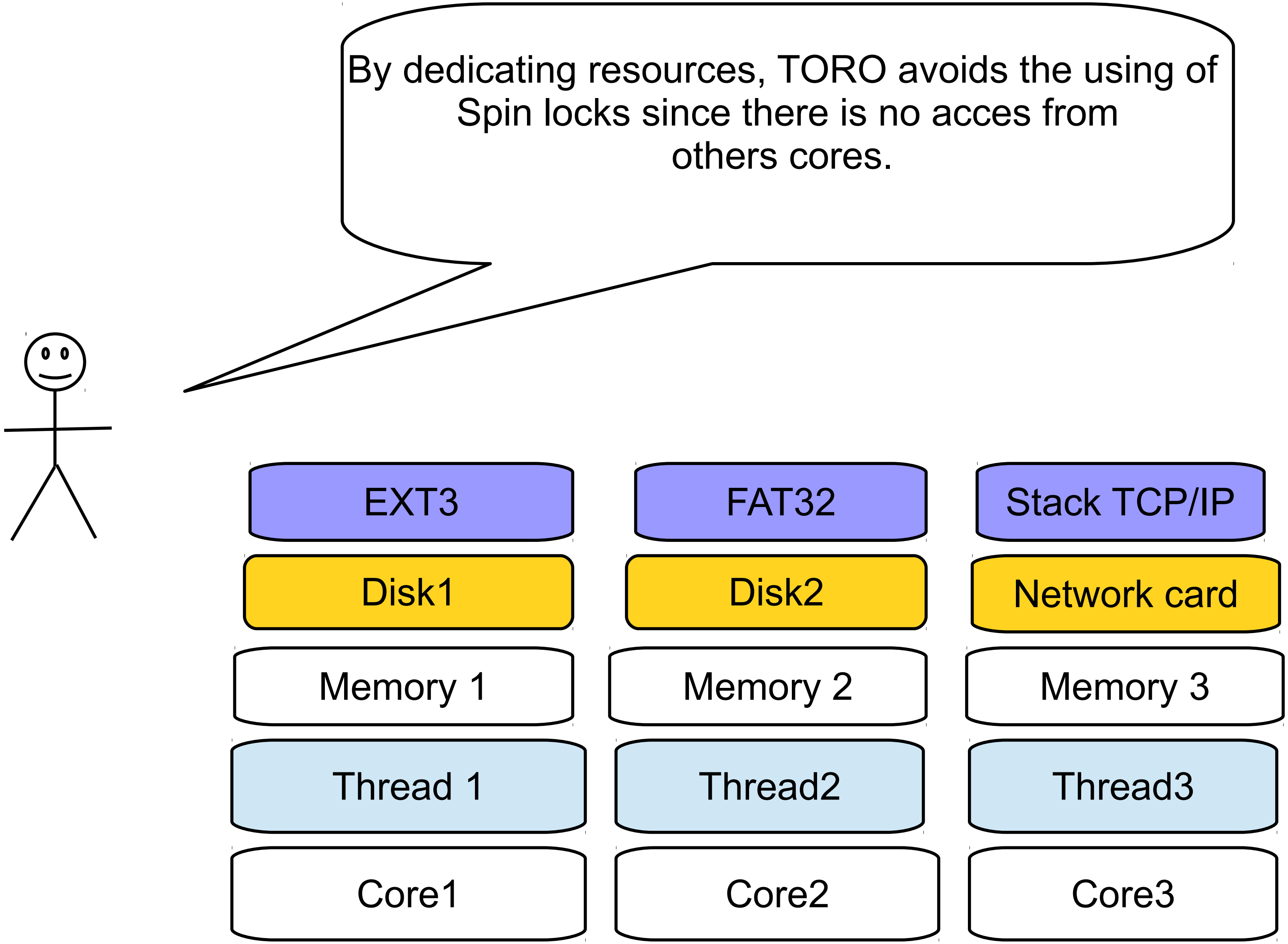
Disk1	Disk2	Network card
Memory 1	Memory 2	Memory 3
Thread 1	Thread2	Thread3
Core1	Core2	Core3

Locality of resources



EXT3	FAT32	Stack TCP/IP
Disk1	Disk2	Network card
Memory 1	Memory 2	Memory 3
Thread 1	Thread2	Thread3
Core1	Core2	Core3

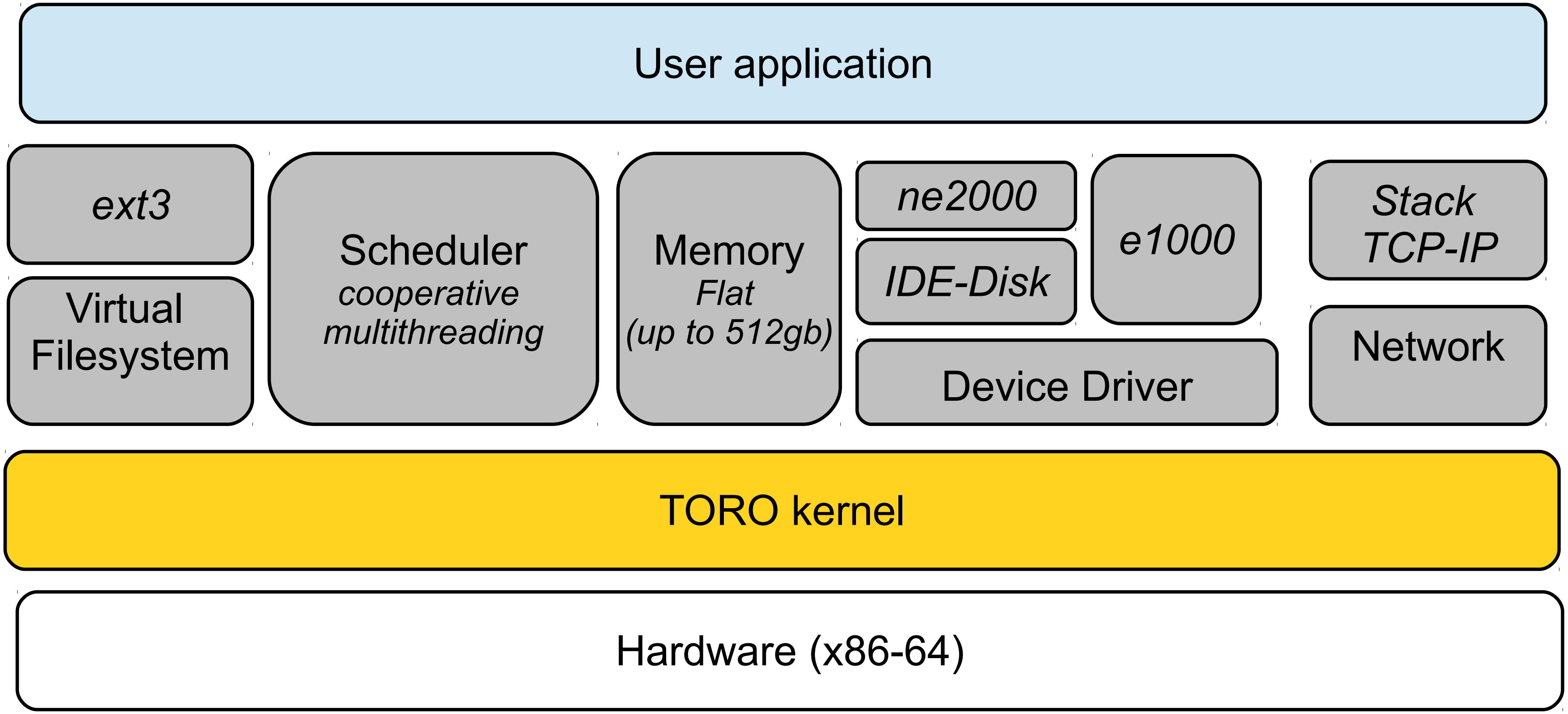
Locality of resources



By dedicating resources, TORO avoids the using of Spin locks since there is no acces from others cores.

EXT3	FAT32	Stack TCP/IP
Disk1	Disk2	Network card
Memory 1	Memory 2	Memory 3
Thread 1	Thread2	Thread3
Core1	Core2	Core3

Current state of project



Thoughts

- The difference between the kernel and application is becoming more thin.
- What is the rol of the kernel?
- When/Why we need a kernel?
- When we dedicate a kernel, it becomes simpler.
- TORO represens a compromise between optimization and portability.

Questions?

Thanks!

torokernel.io

matiasevara@gmail.com