



# A dedicated kernel named **TORO**

Matias Vara Larsen

FOSDEM'15



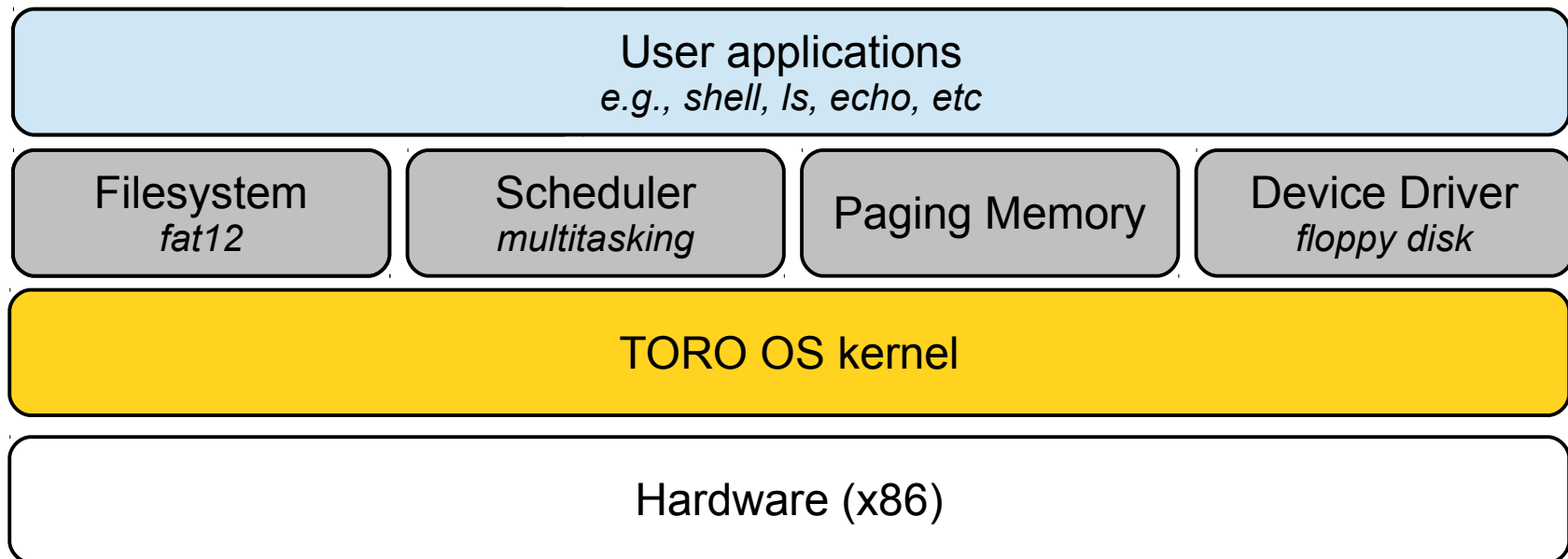
# Who am I?



- 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 shell



# LS



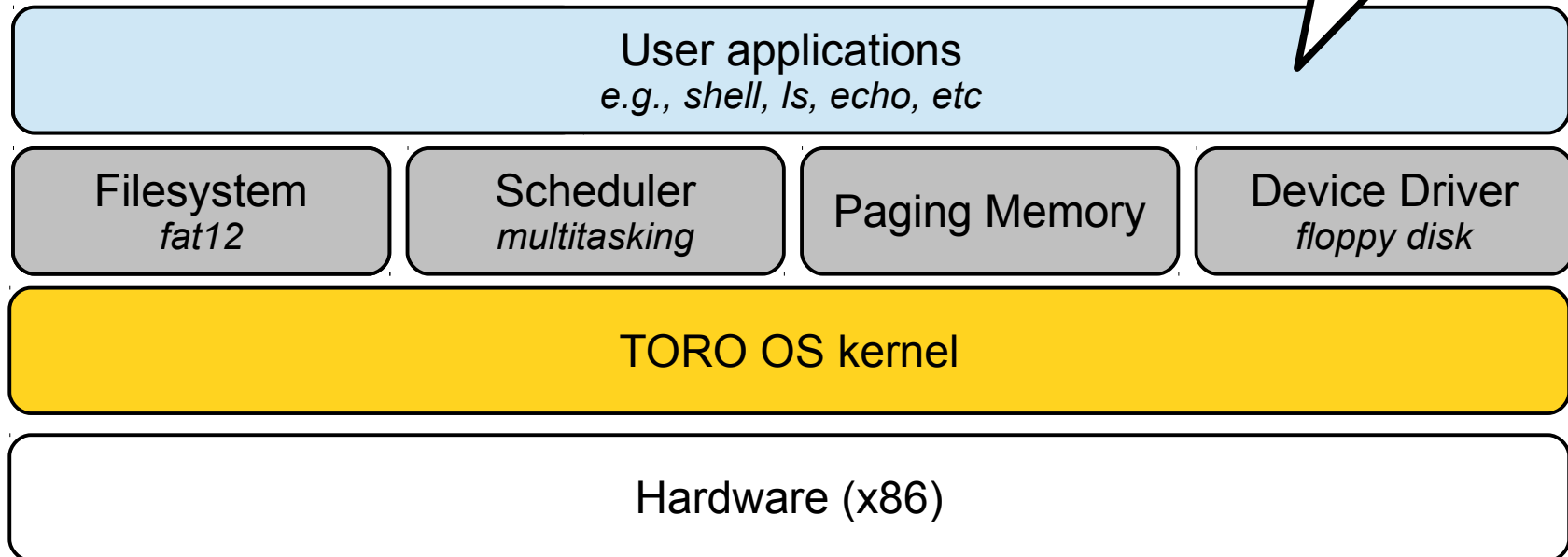
# LS



# What is TORO

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

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



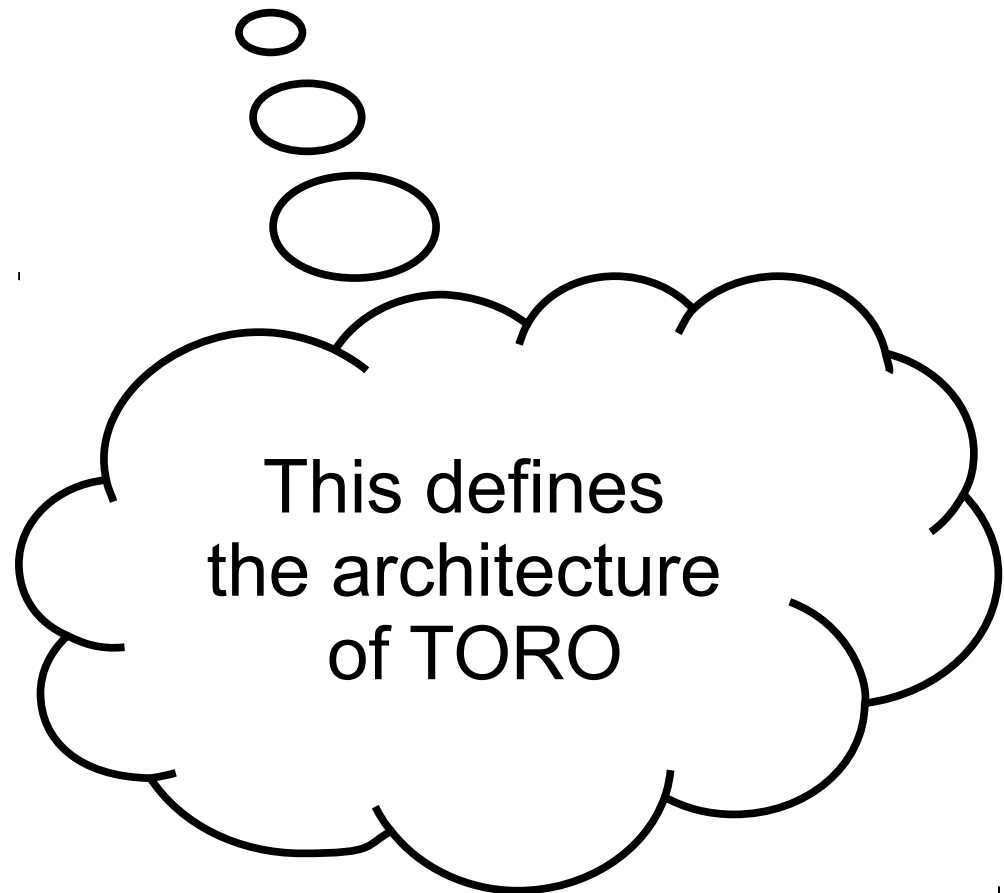
# What is TORO kernel?

- 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***



# 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

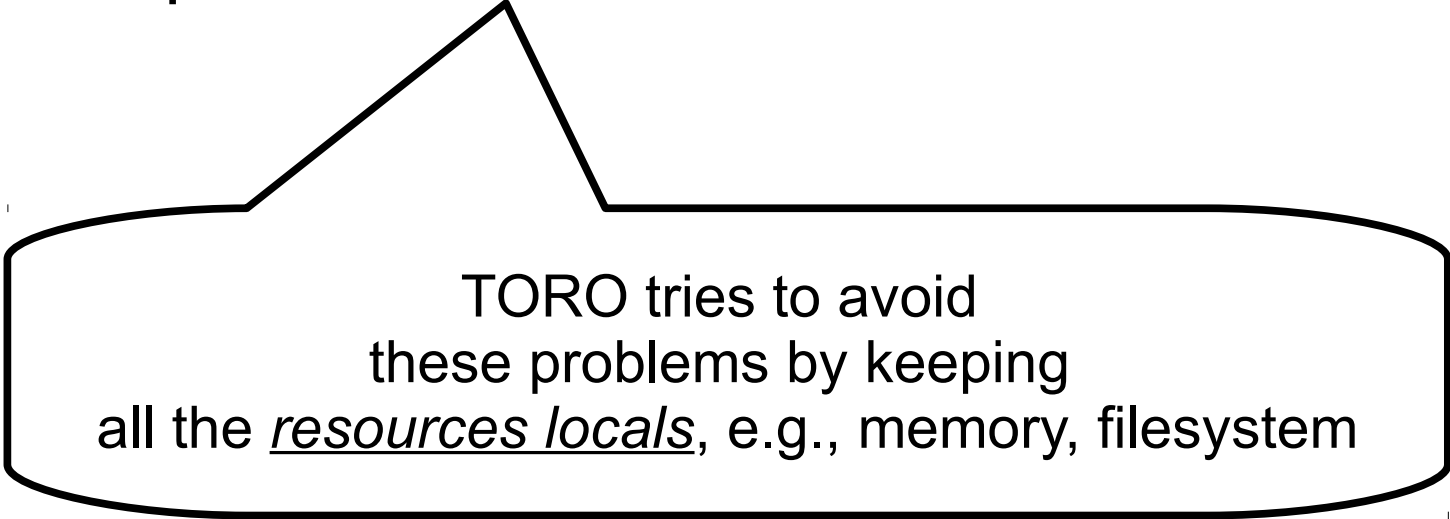
- Light context switching
- In this sense, TORO is a *library OS-like designing*.

# 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.

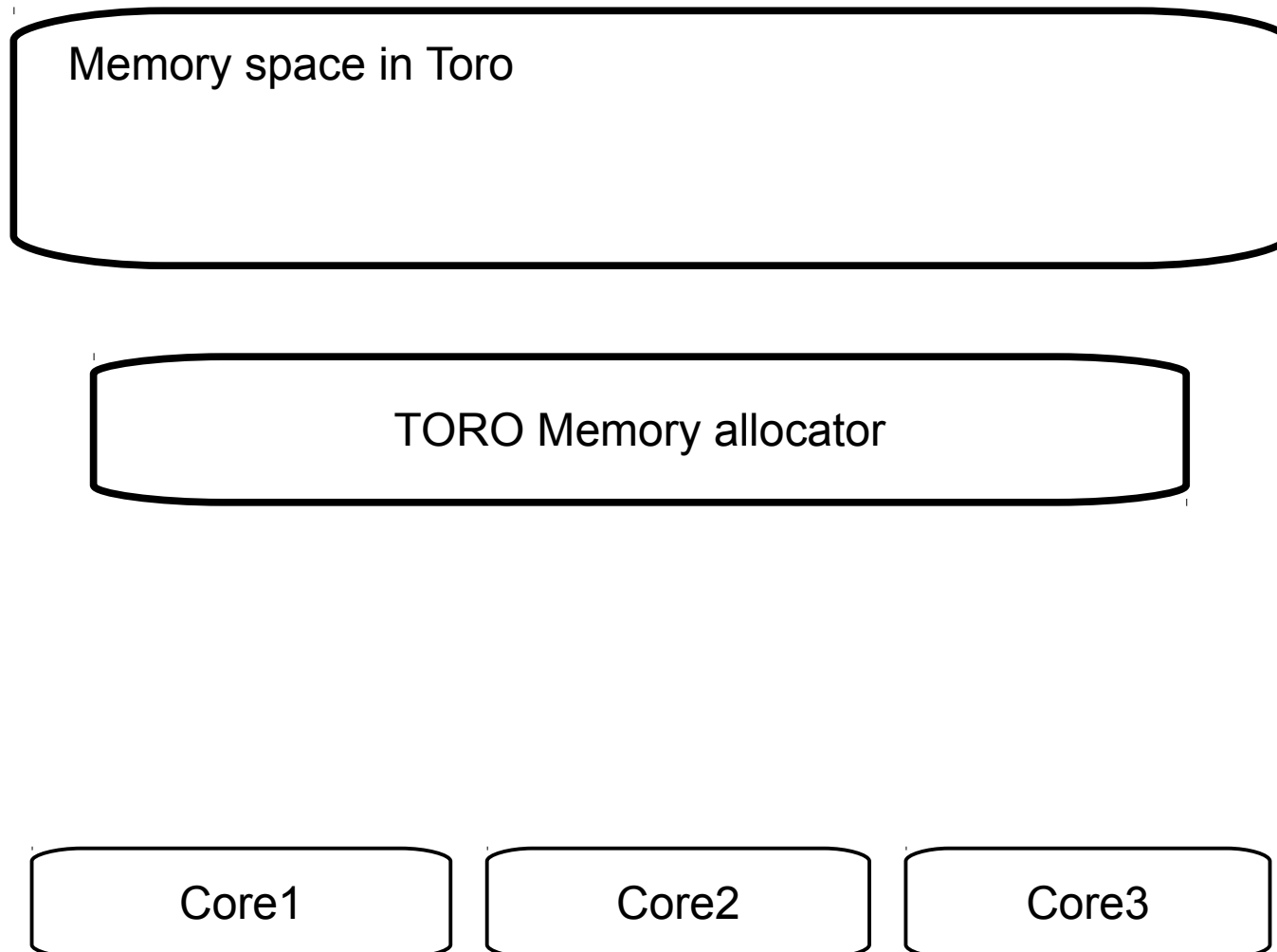
# Dedicated Resources

- 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



# 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

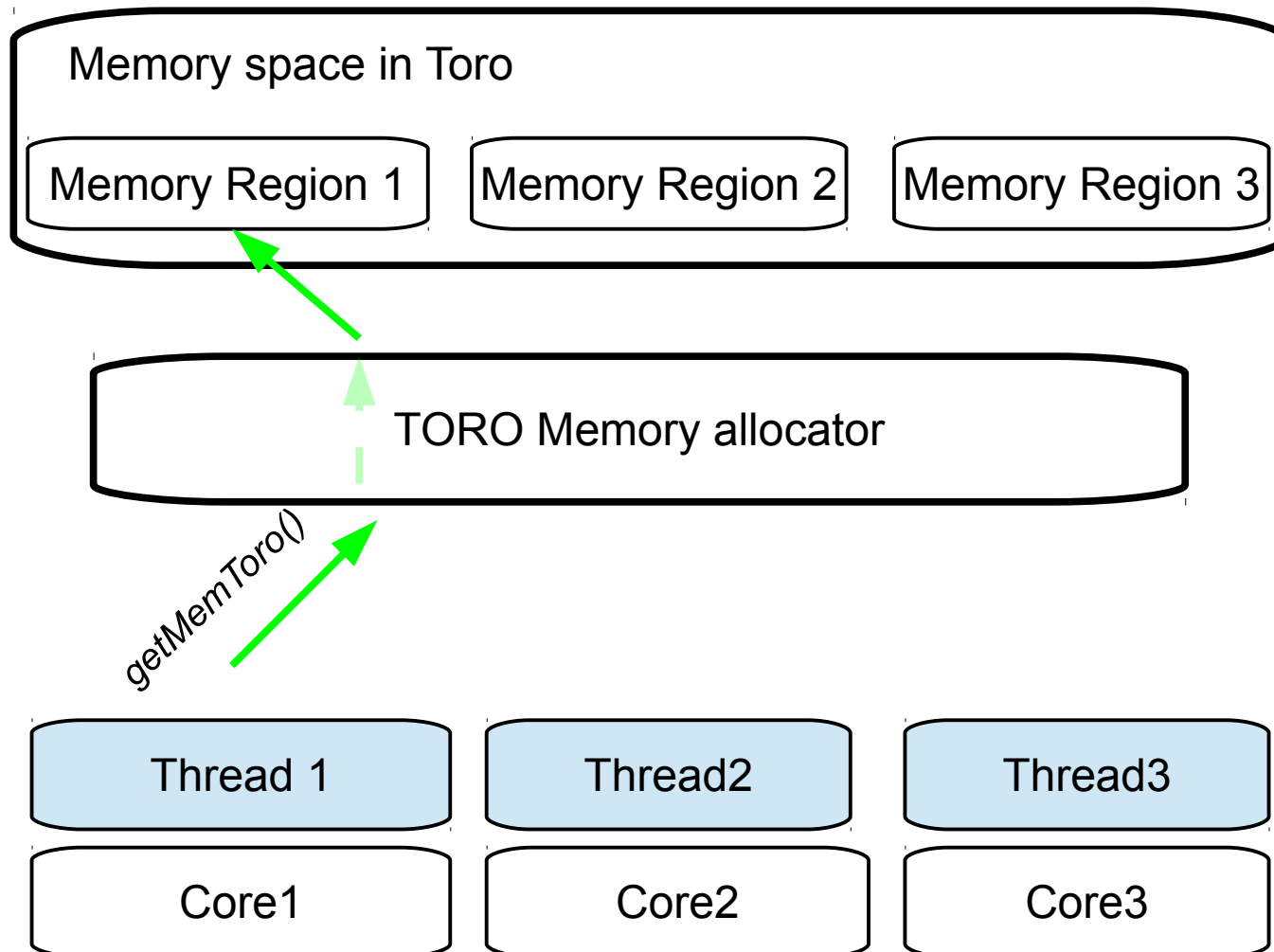
Core1

Core2

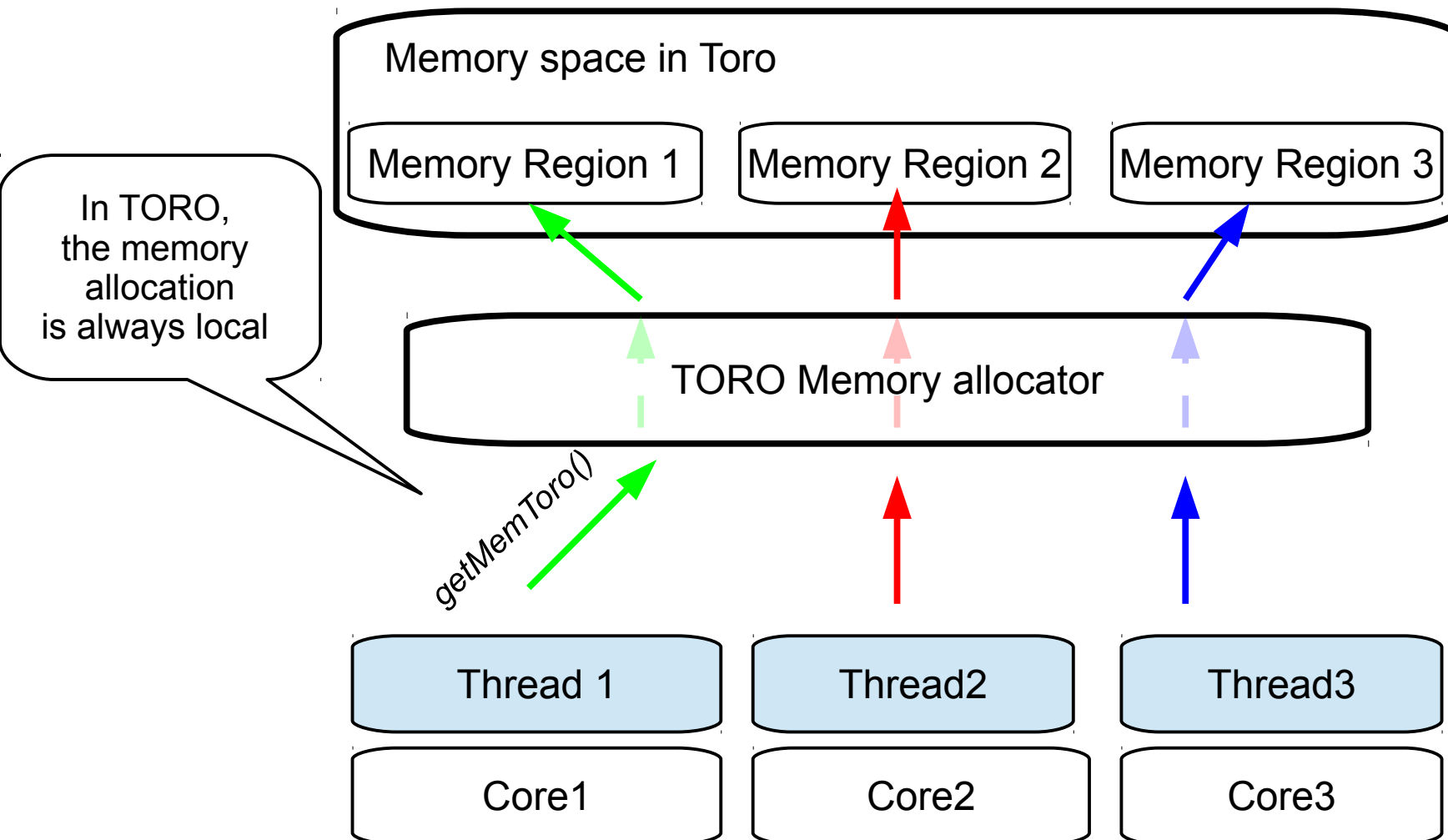
Core3



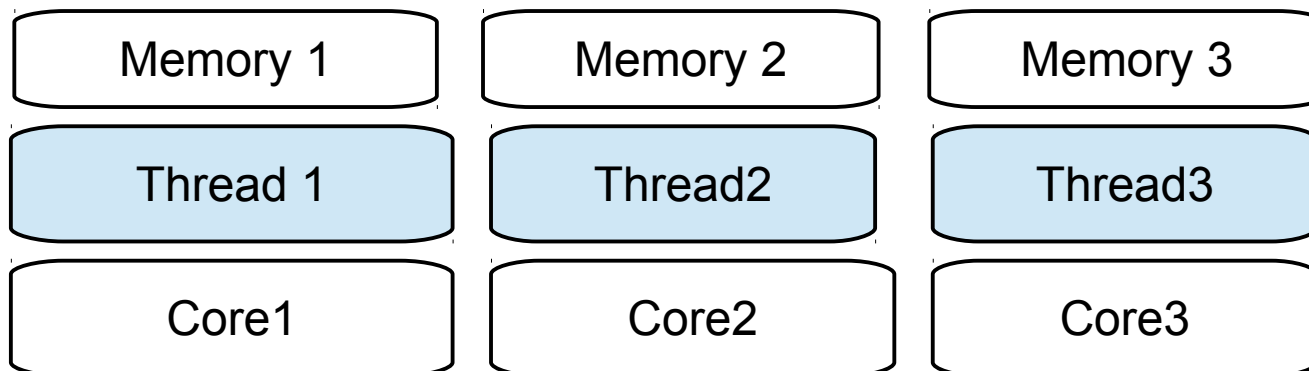
# Dedicated Memory Allocation



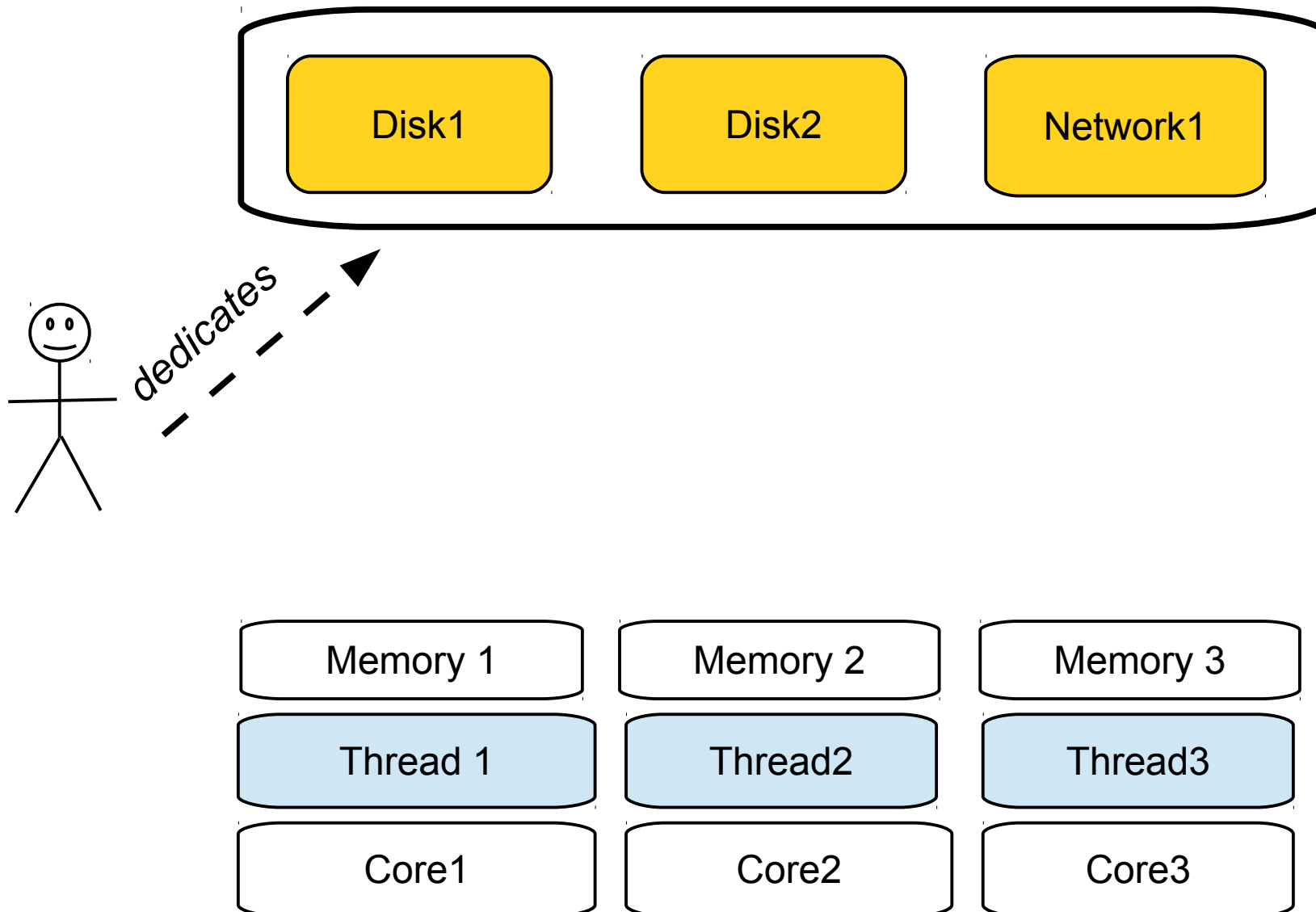
# Dedicated Memory Allocation



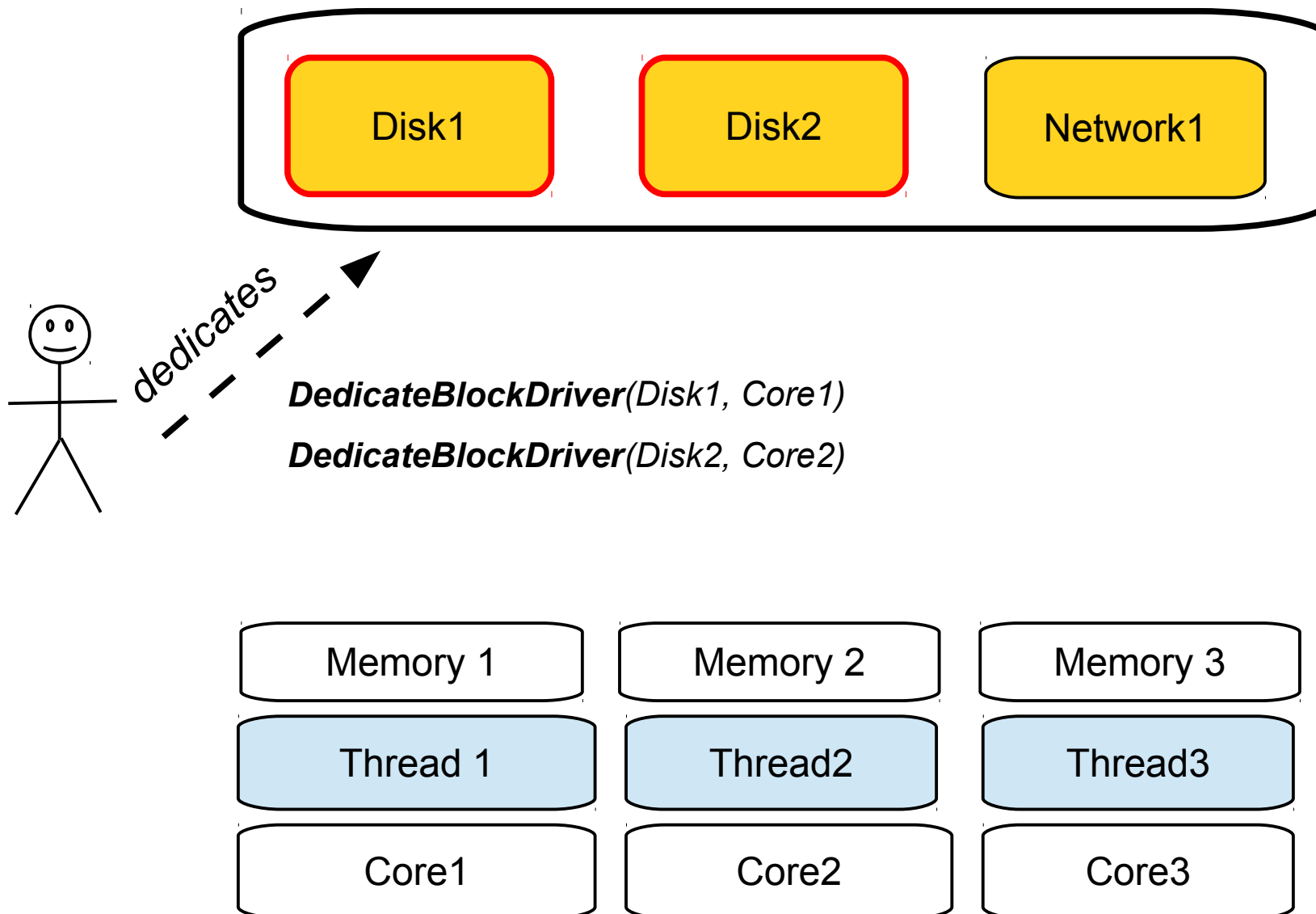
# Locality of memory



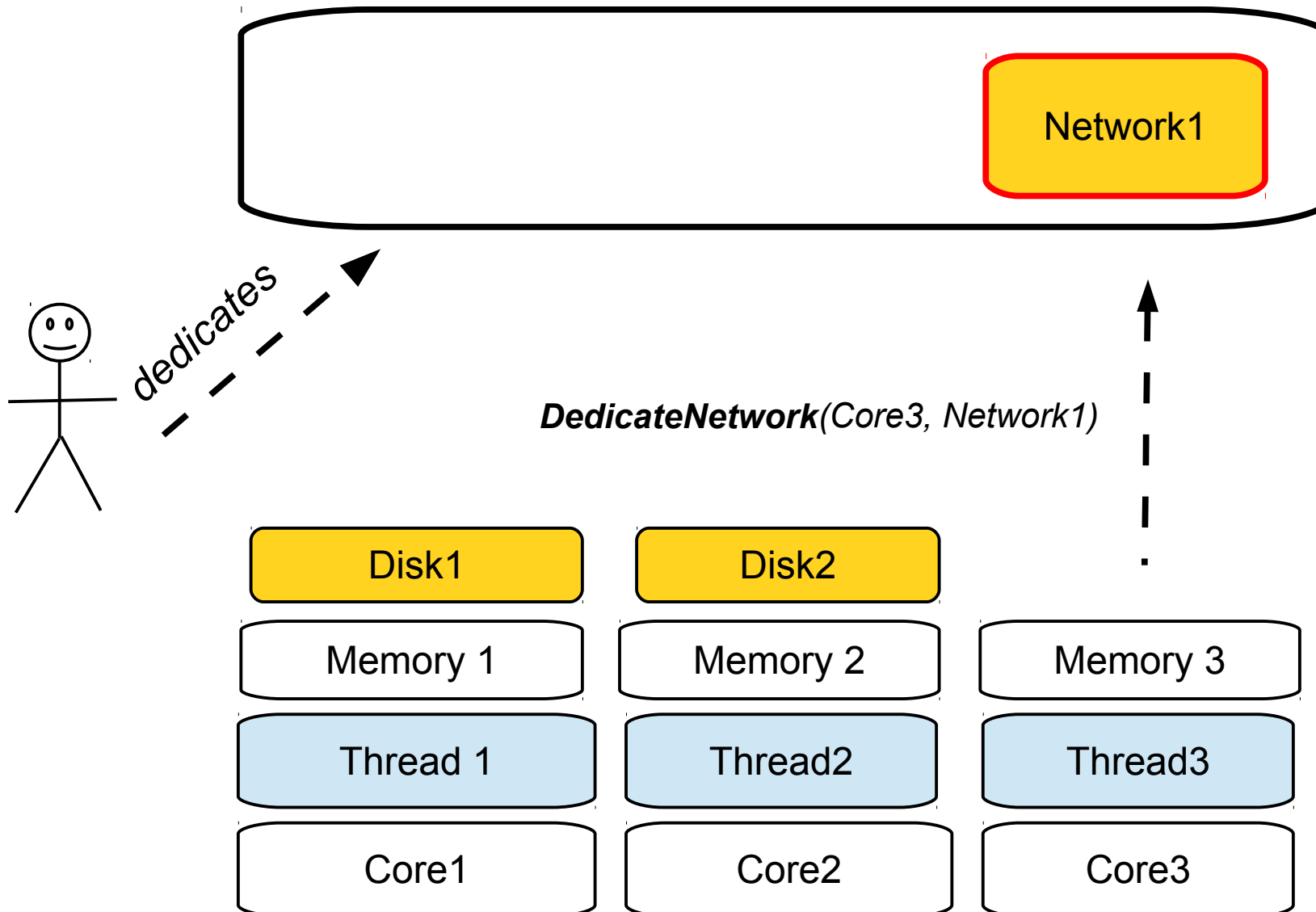
# Locality of resources



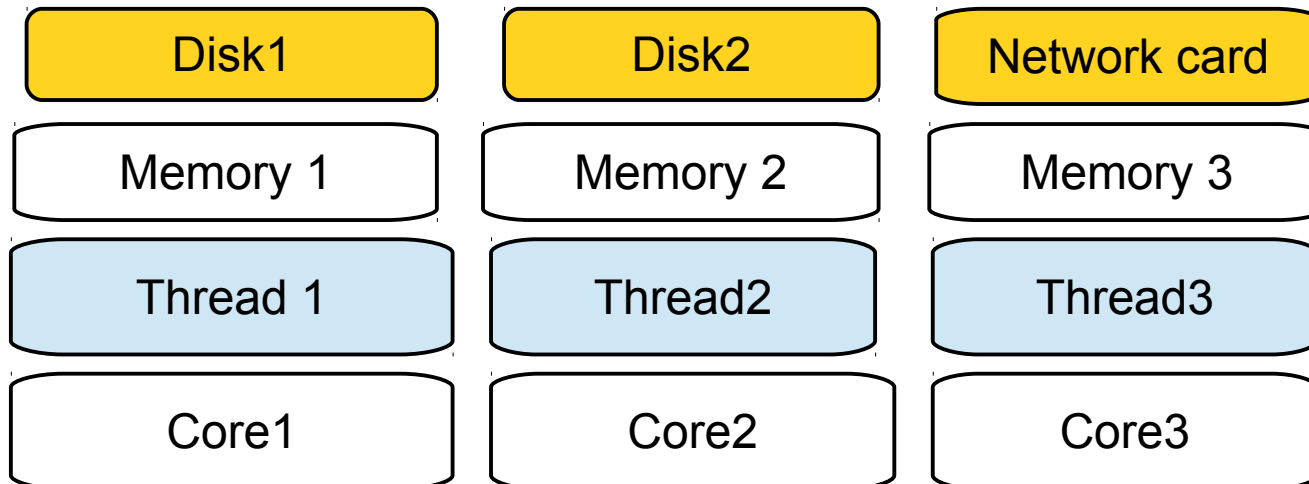
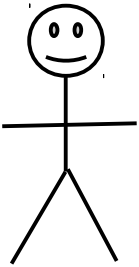
# Locality of resources



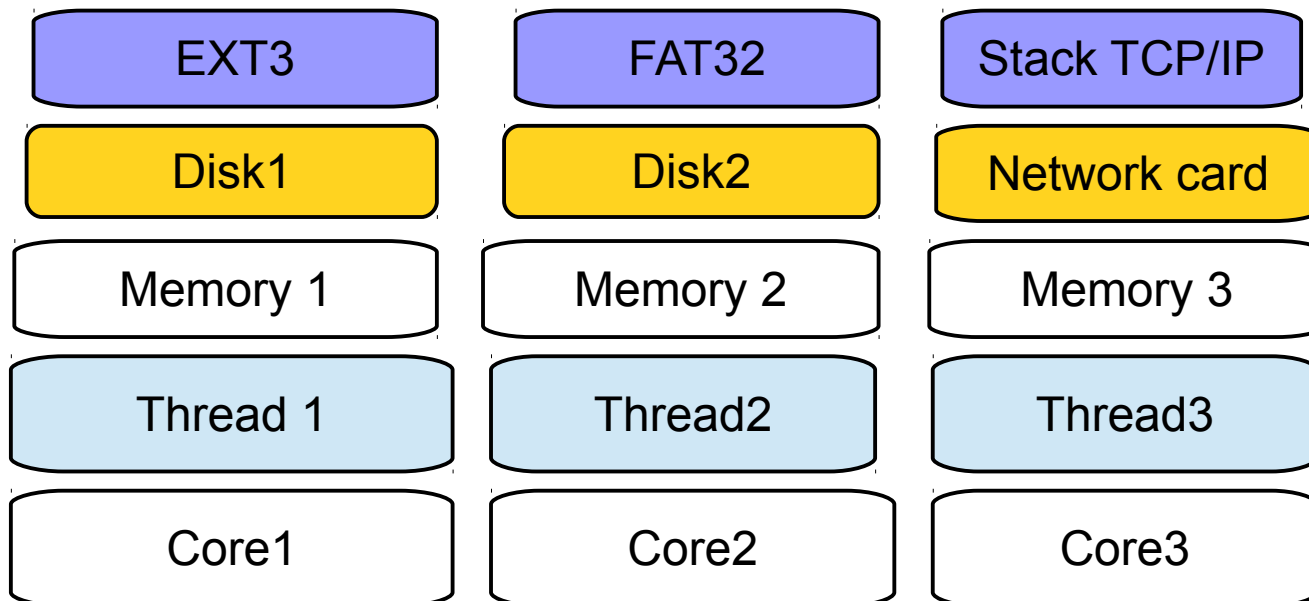
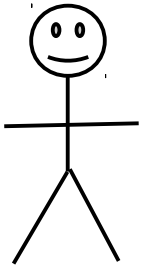
# Locality of resources



# Locality of resources



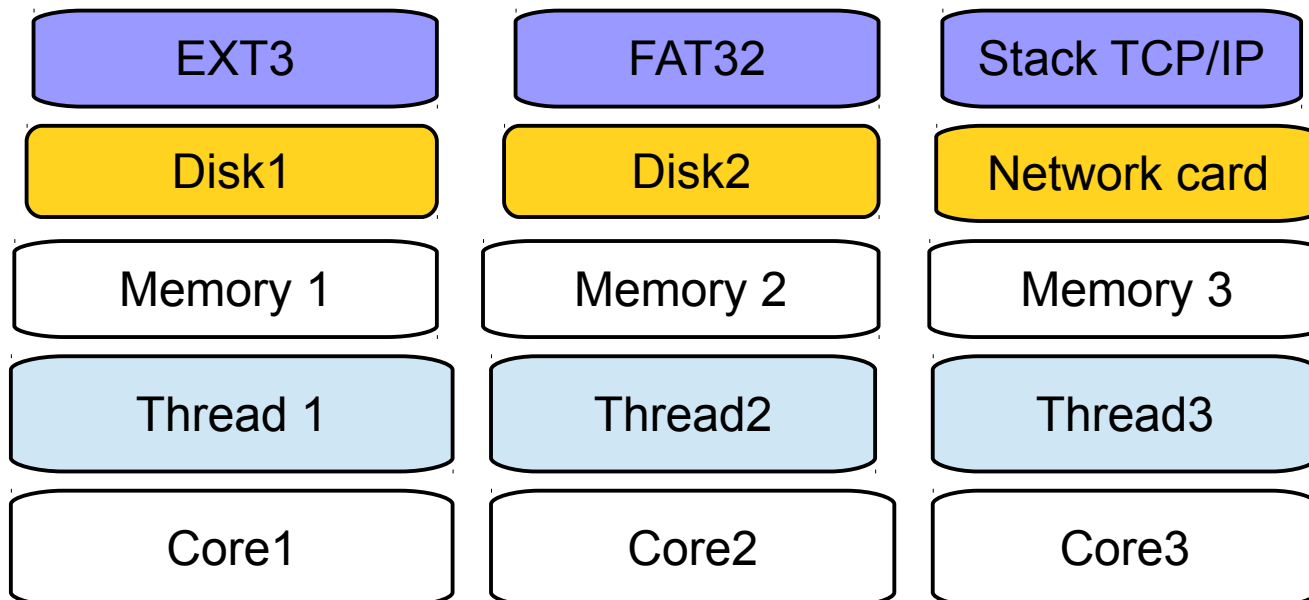
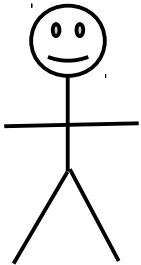
# Locality of resources



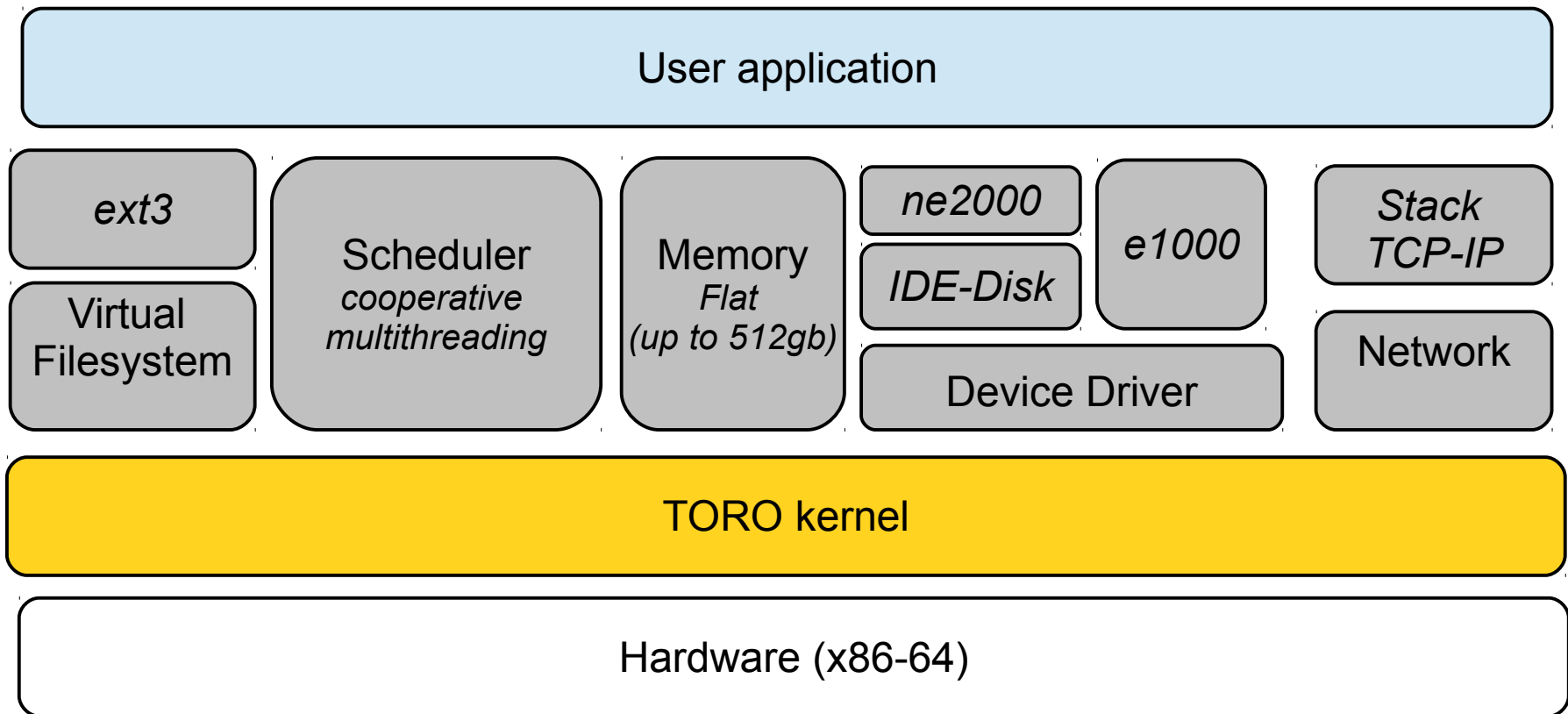


# Locality of resources

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



# Current state of project



# Thoughts

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

# Questions?



# Thanks!

torokernel.io

[matiasevara@gmail.com](mailto:matiasevara@gmail.com)