# Managing multicore caches

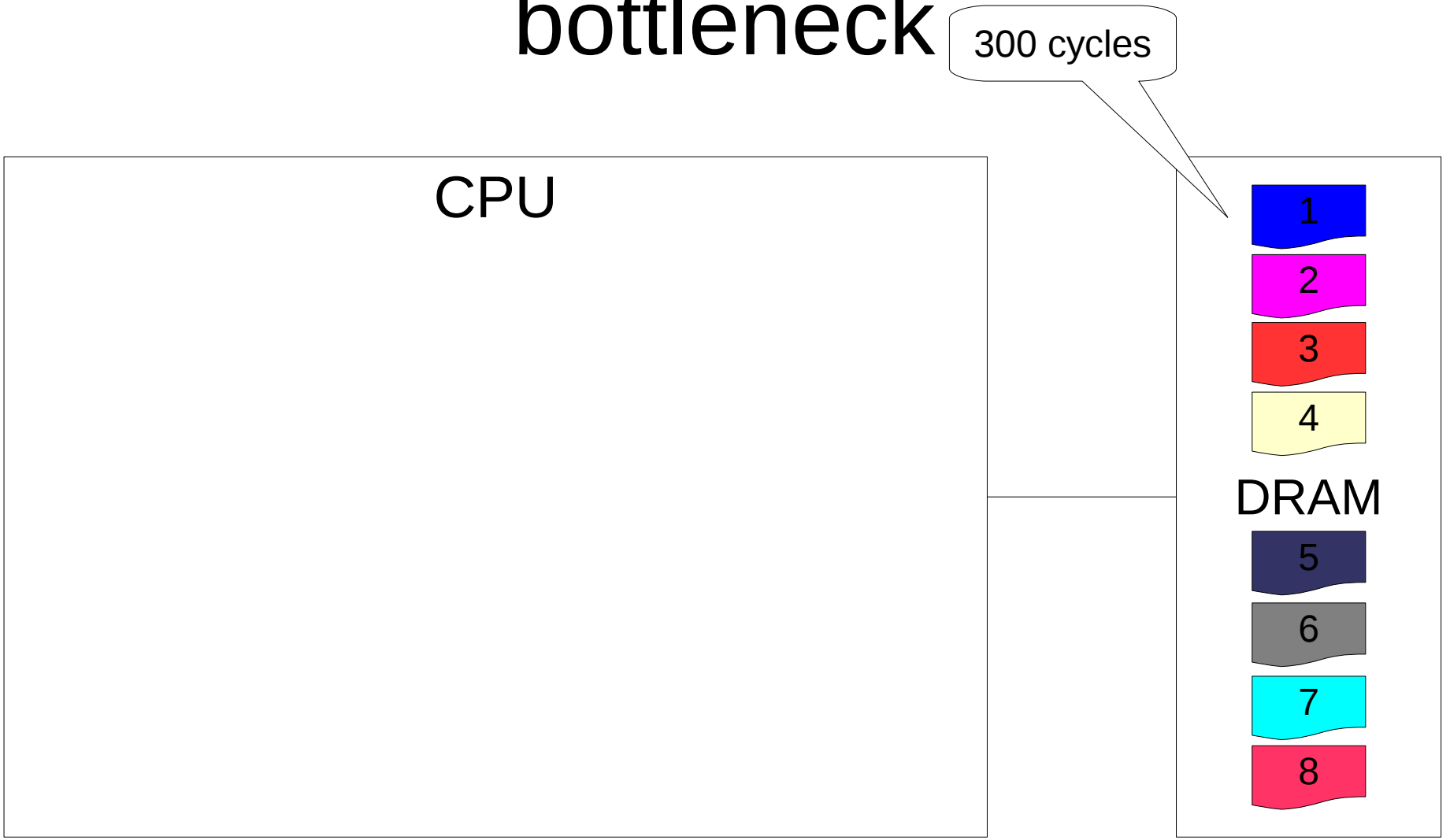
Silas Boyd-Wickizer, Robert Morris, Nickolai Zeldovich, M. Frans Kaashoek

# What this talk is about

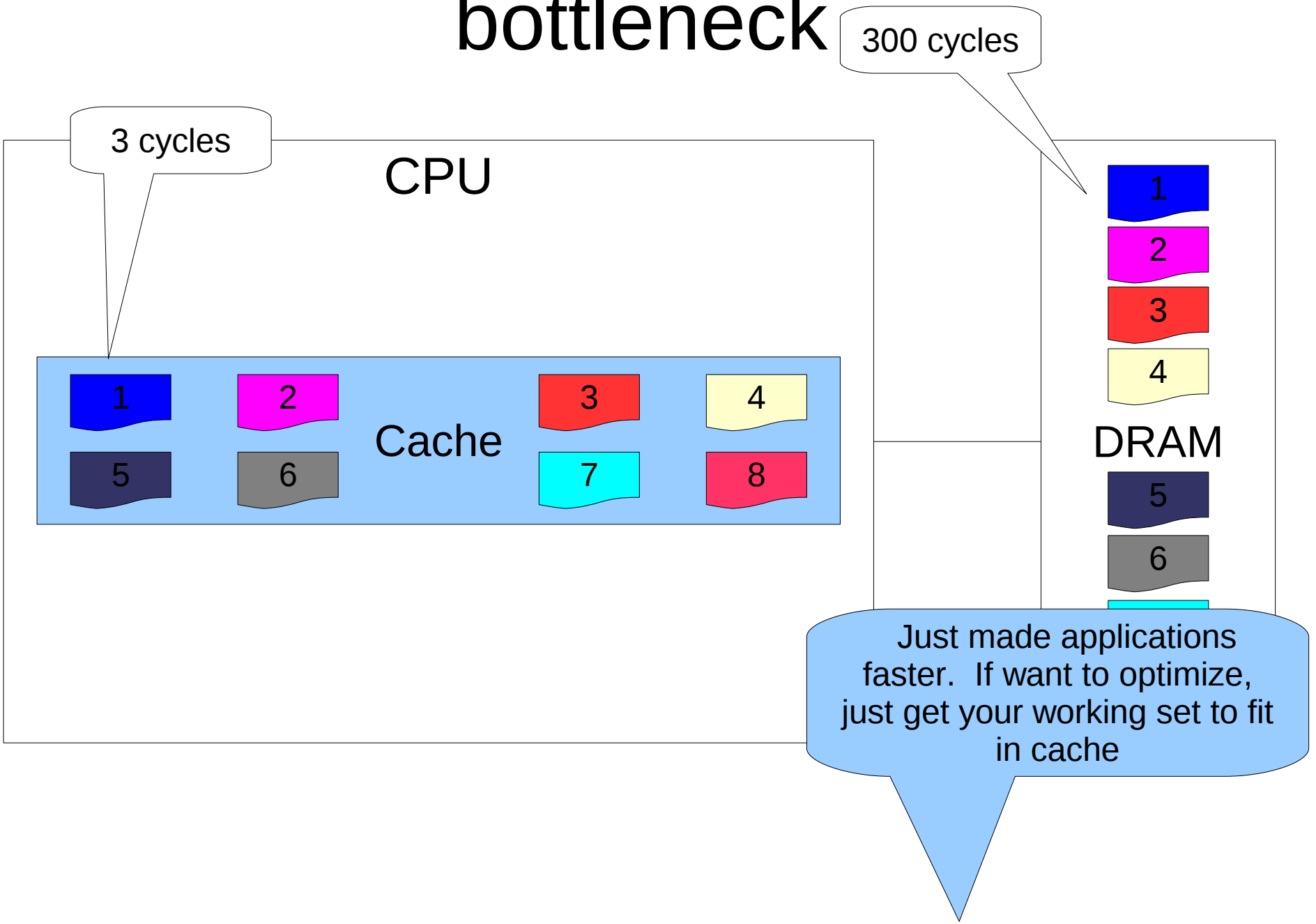
- Managing multicore caches more challenging than uniprocessor caches
- One technique to get better performance from multicore caches
  - Preliminary result
  - Curious if approach applies to your applications

# Caches invented to avoid DRAM bottleneck

# Caches invented to avoid DRAM bottleneck



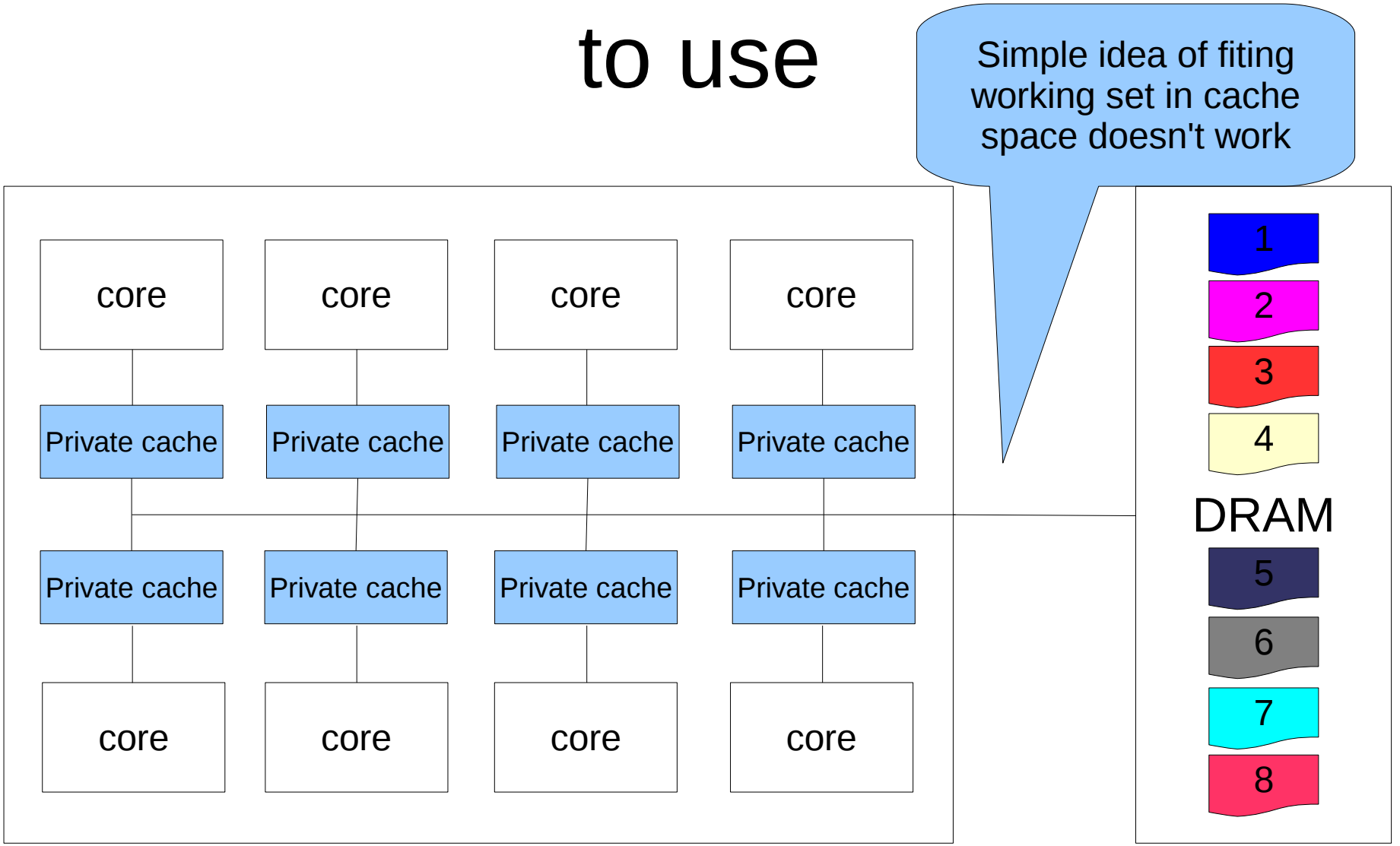
# Caches invented to avoid DRAM bottleneck



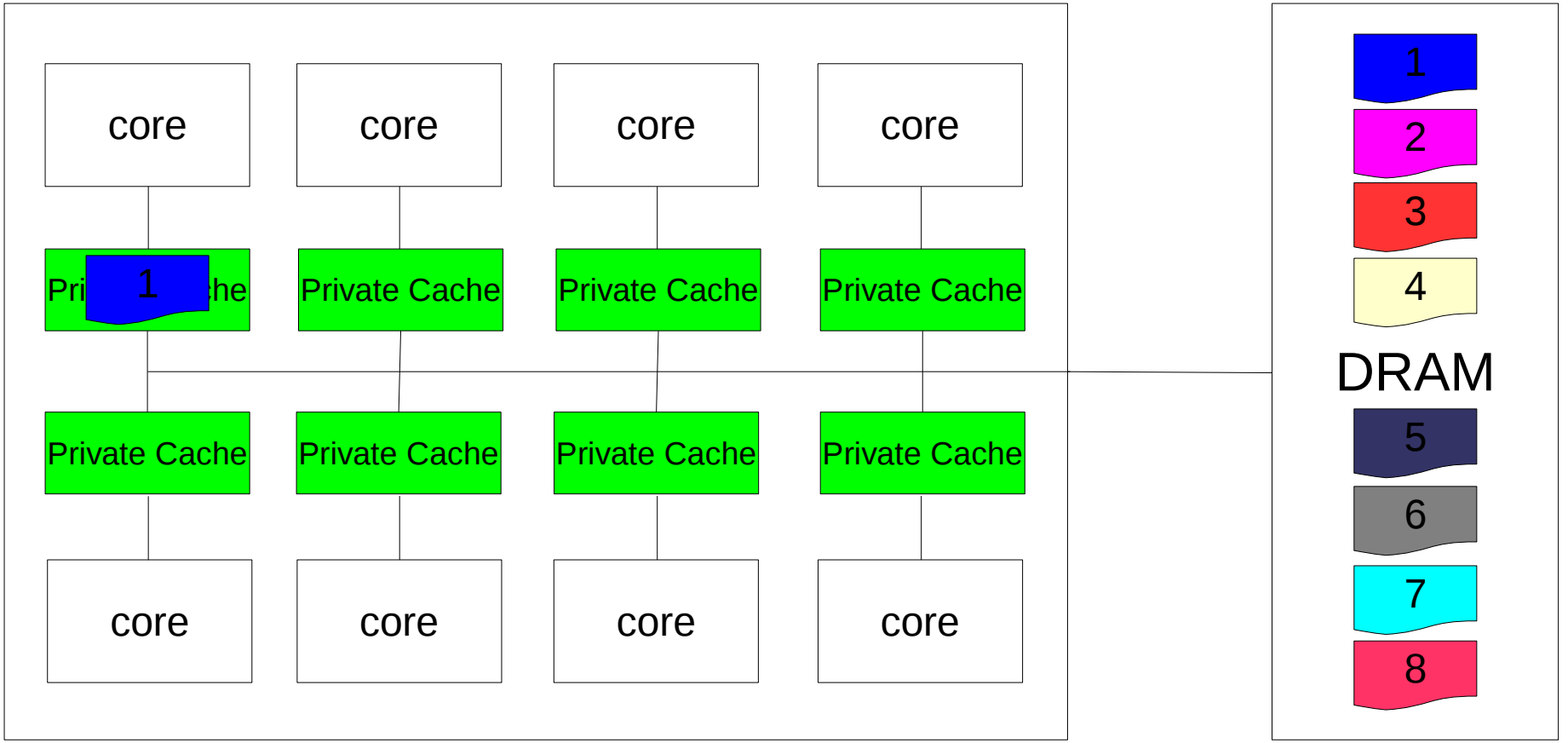
# Multicores have lots of cache

- More cores = more cache space
  - Instead of 2Mbytes, 16Mbytes!
- Many applications can benefit from more cache:
  - Apache
  - memcached
  - ClamAV
  - ...

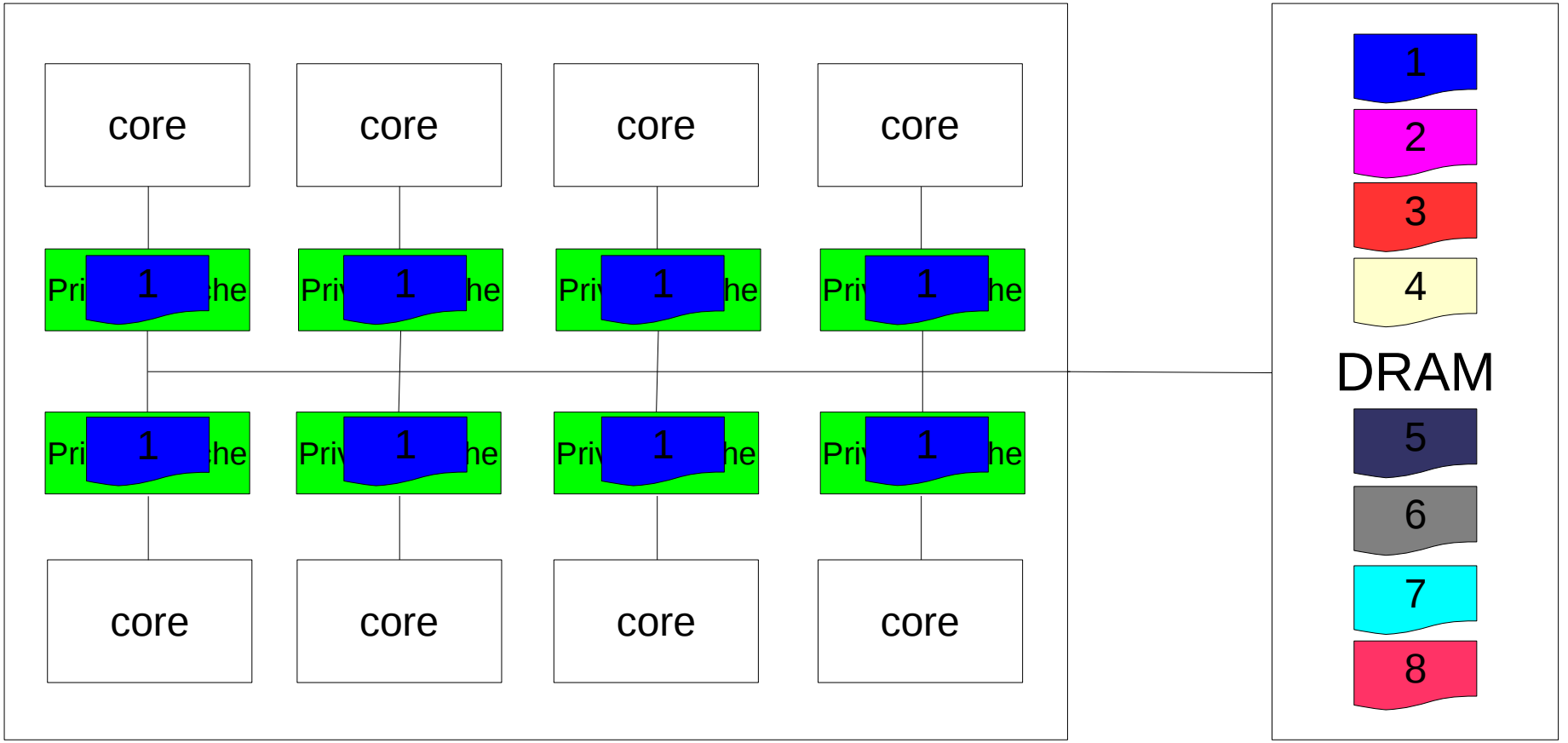
# Multicore caches are more difficult to use



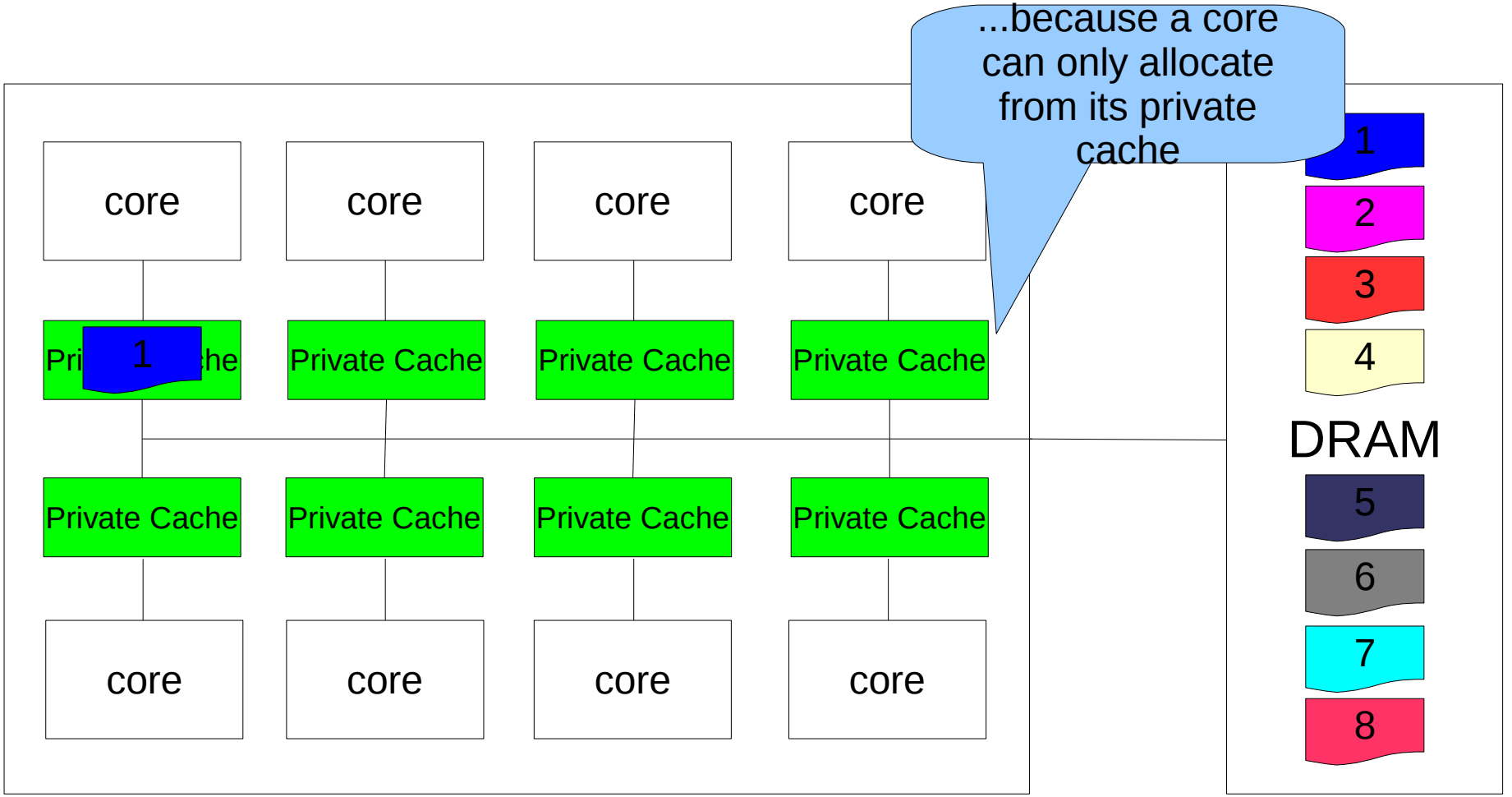
# Multicore caches: duplicate data



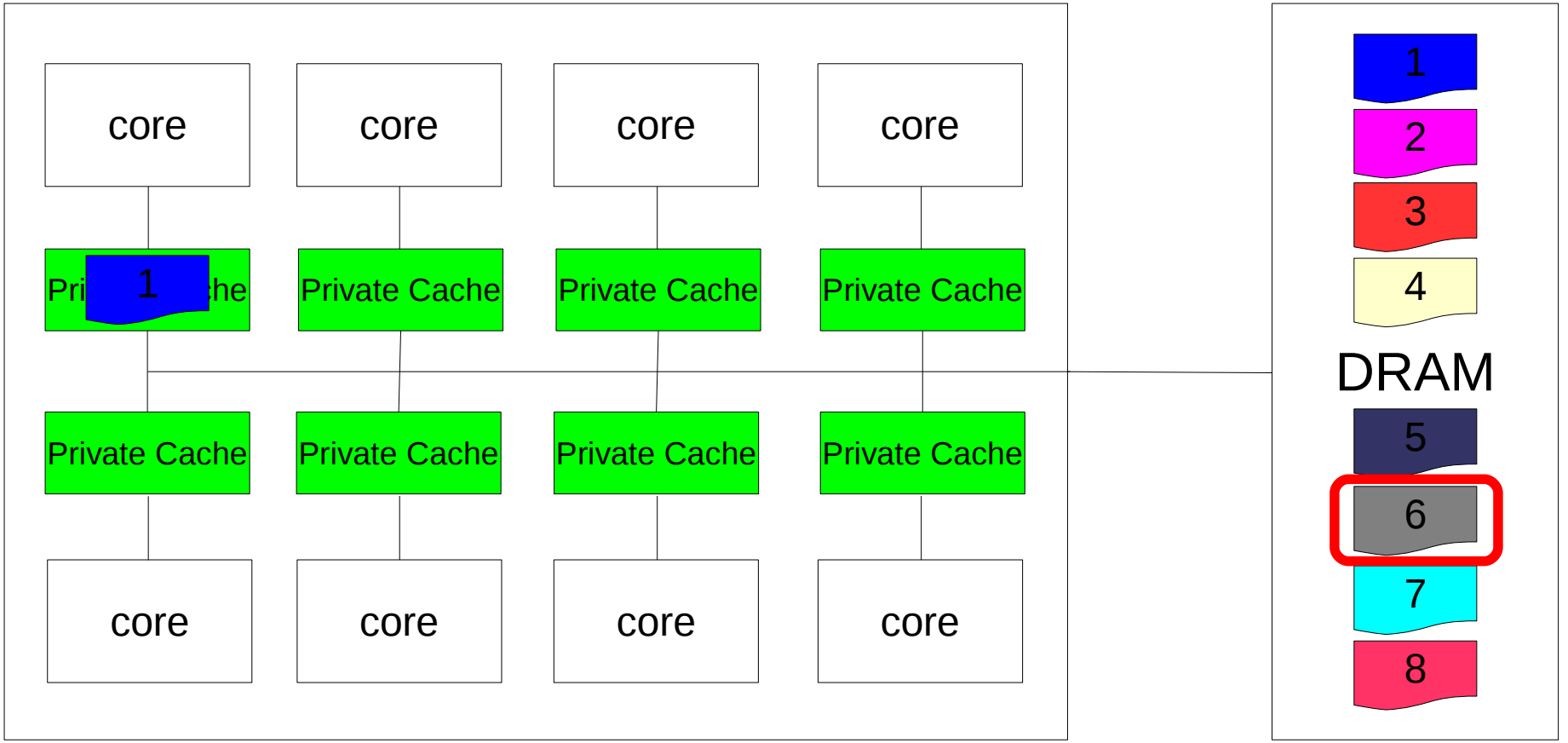
# Multicore caches: duplicate data



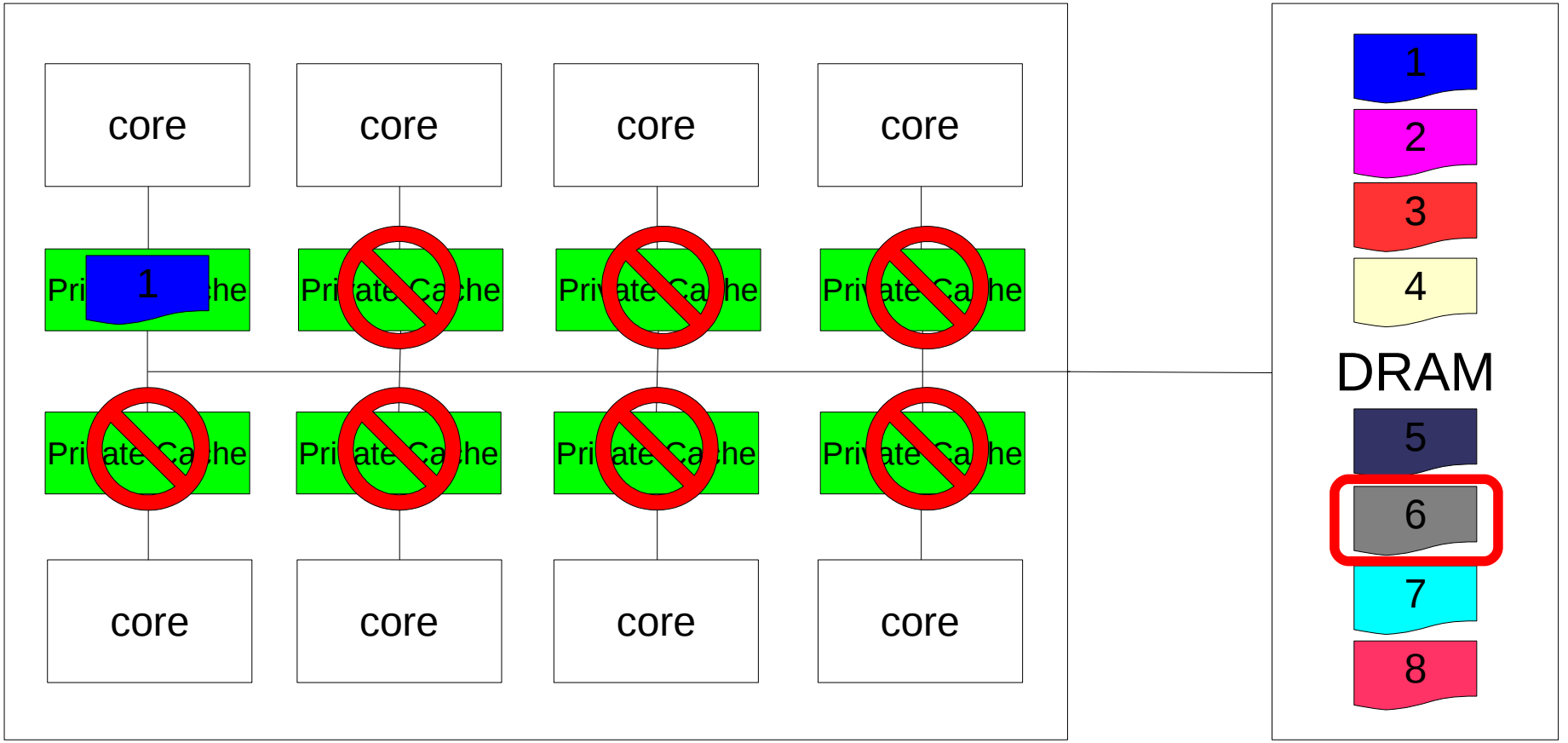
# Multicore caches: local allocation



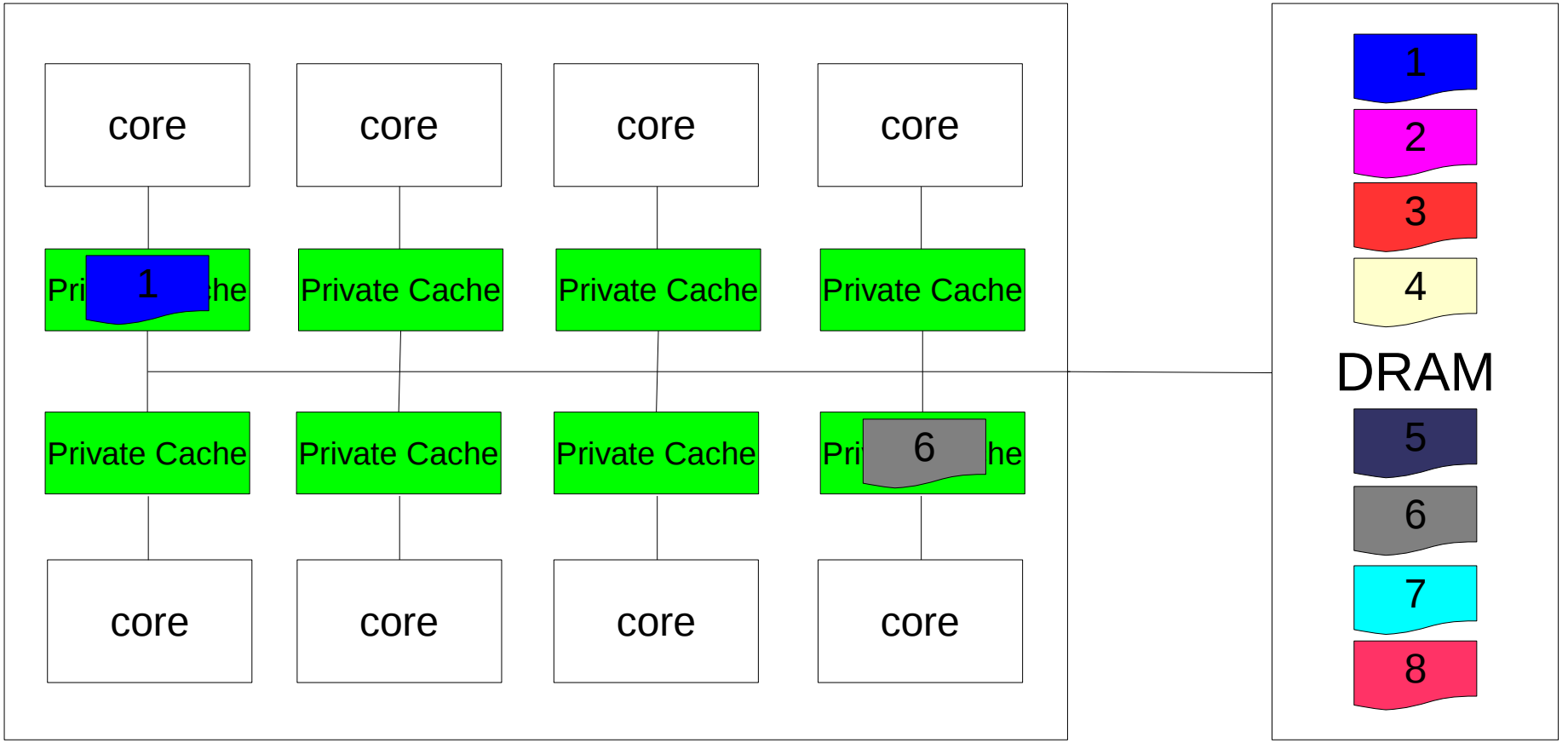
# Multicore caches: local allocation



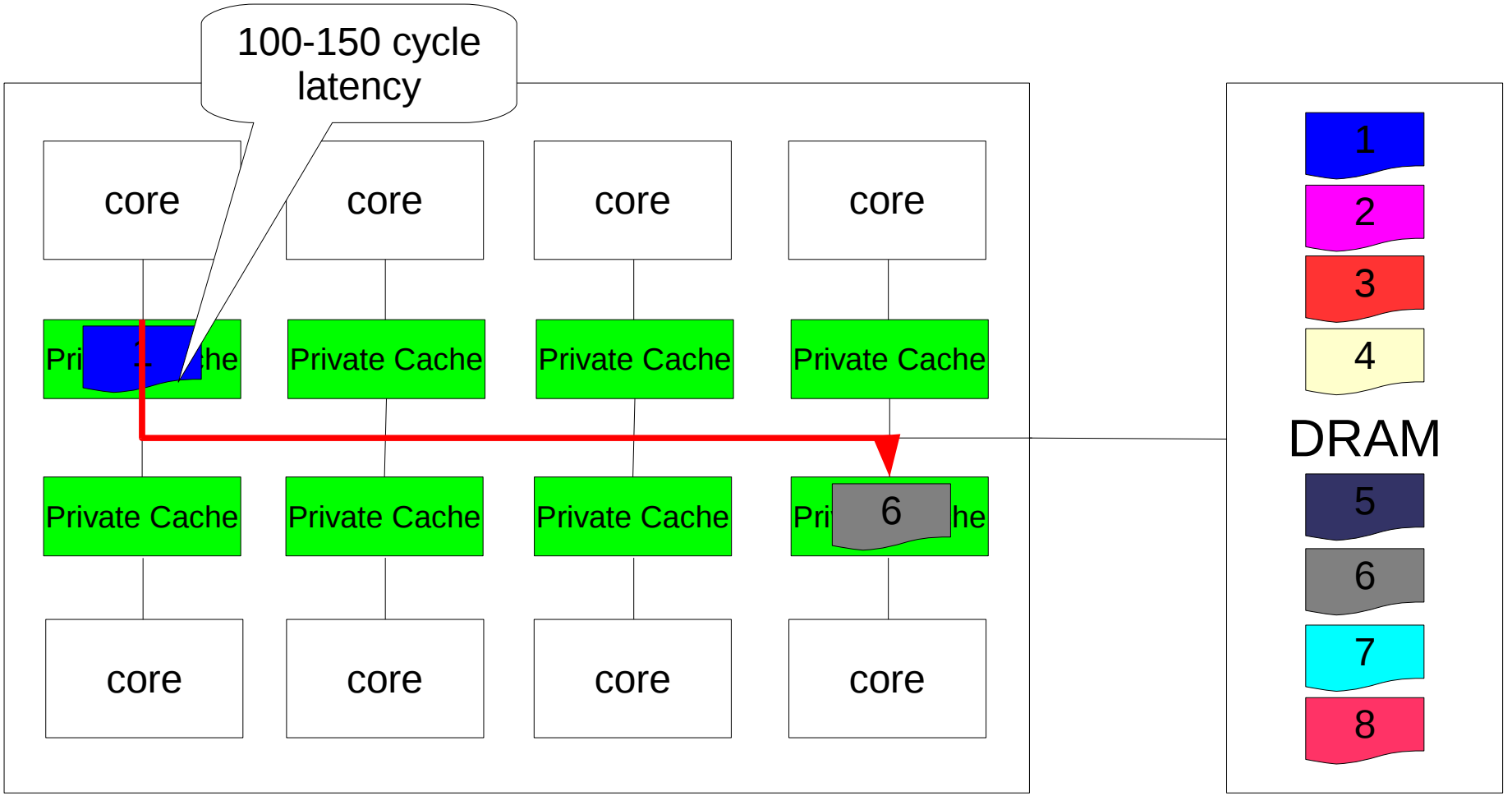
# Multicore caches: local allocation



# Multicore caches: remote caches are slow

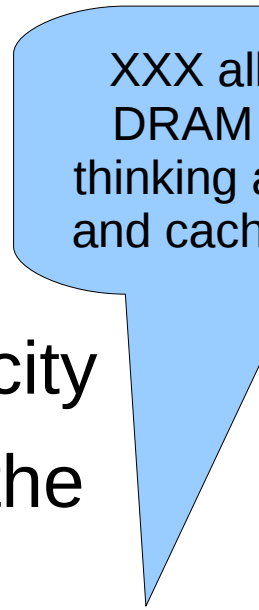


# Multicore caches: remote caches are slow

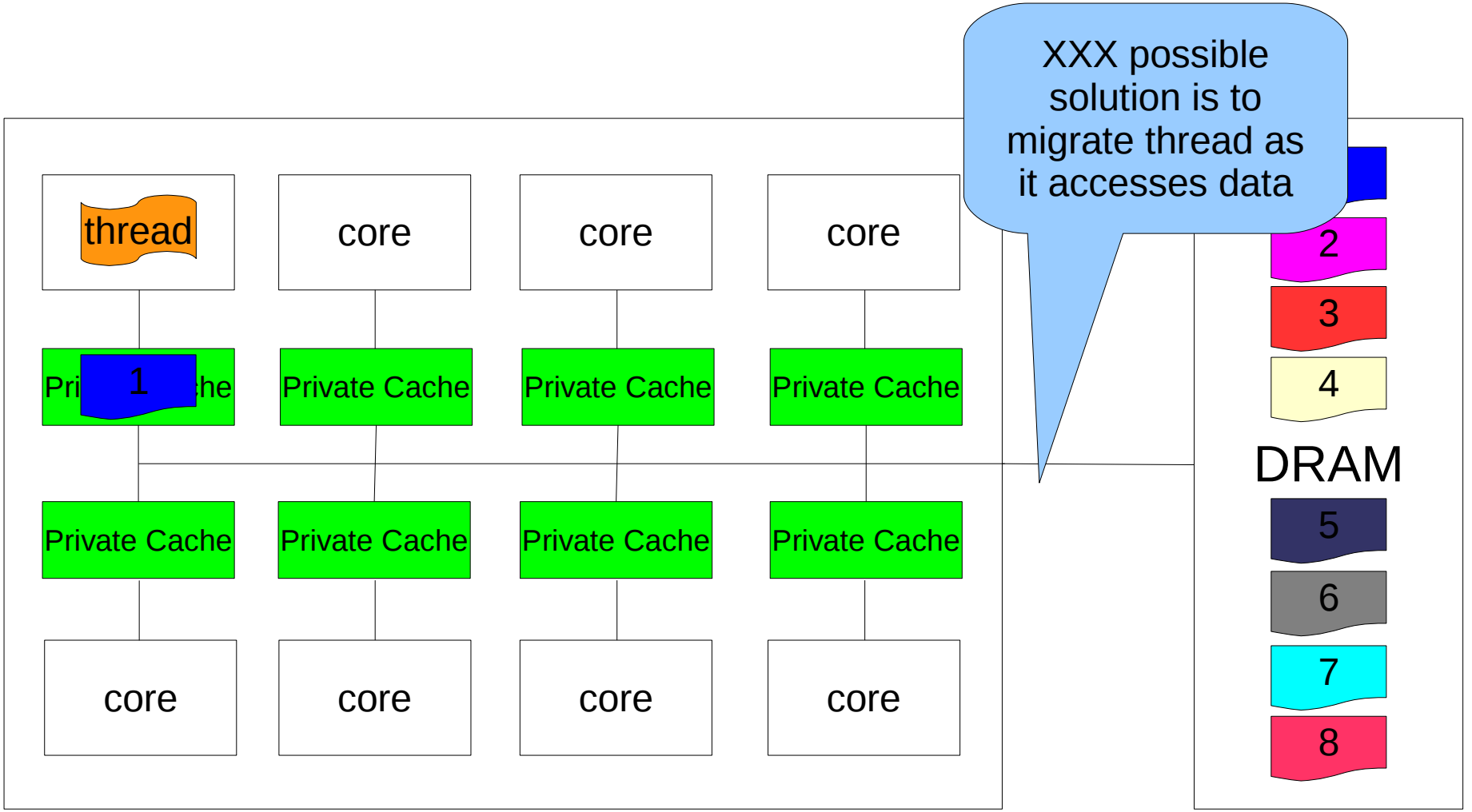


# Software should manage caches

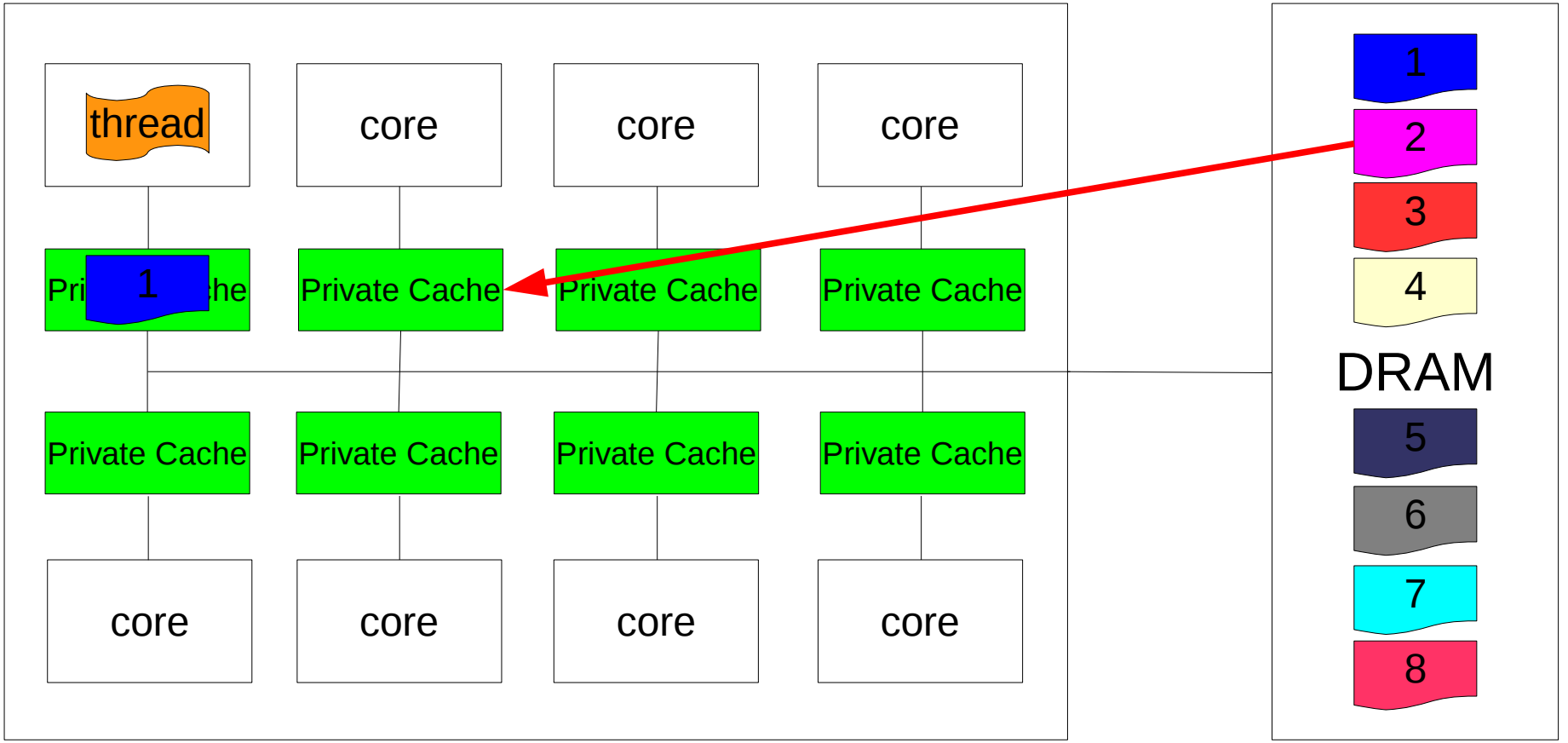
- Software already forced to manage cores
- Software must also be aware that:
  - remote caches are slow
  - risk of duplicating data, reducing effective capacity
  - single core can allocate only a small portion of the cache
- Uniprocessor cache model is too simplistic



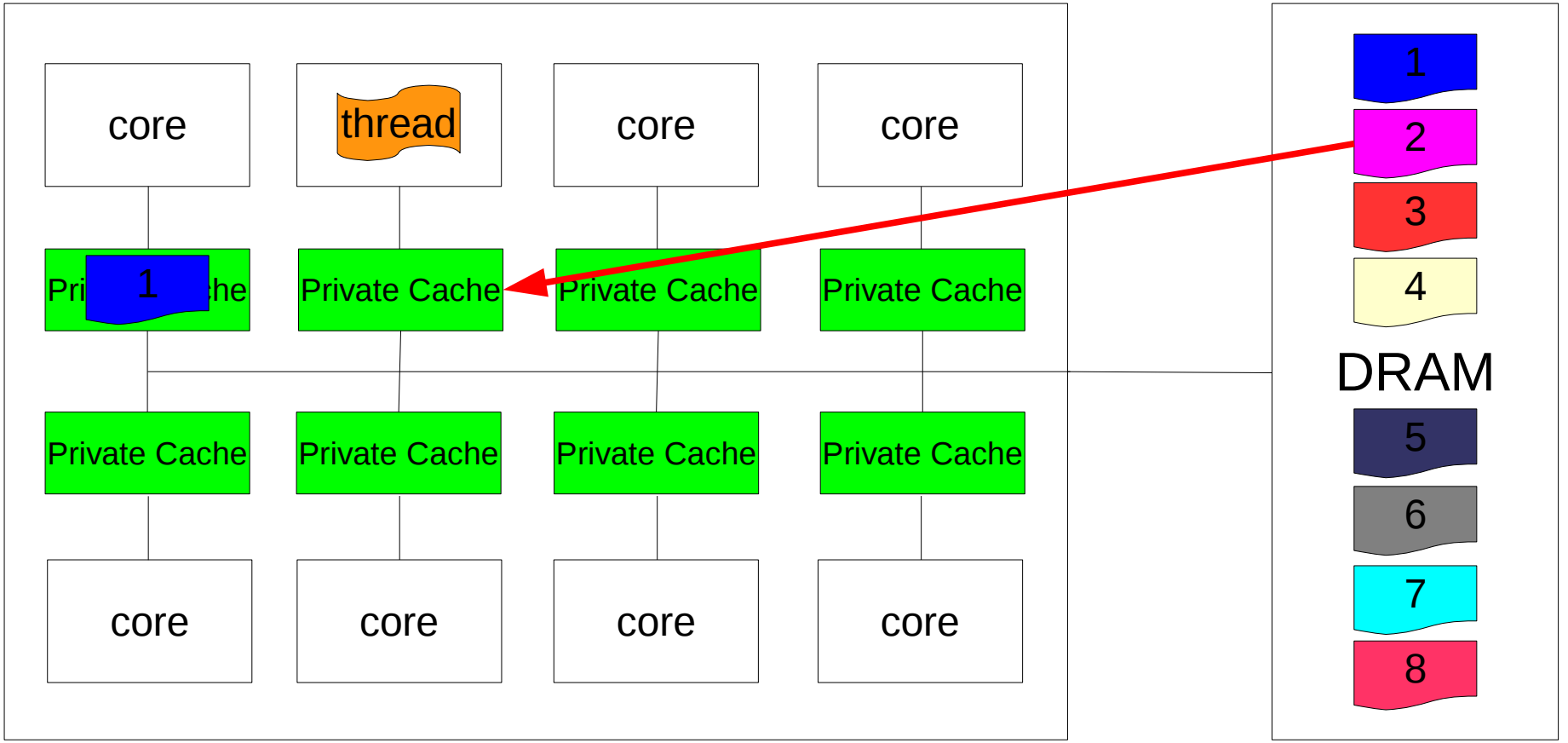
# Software solution



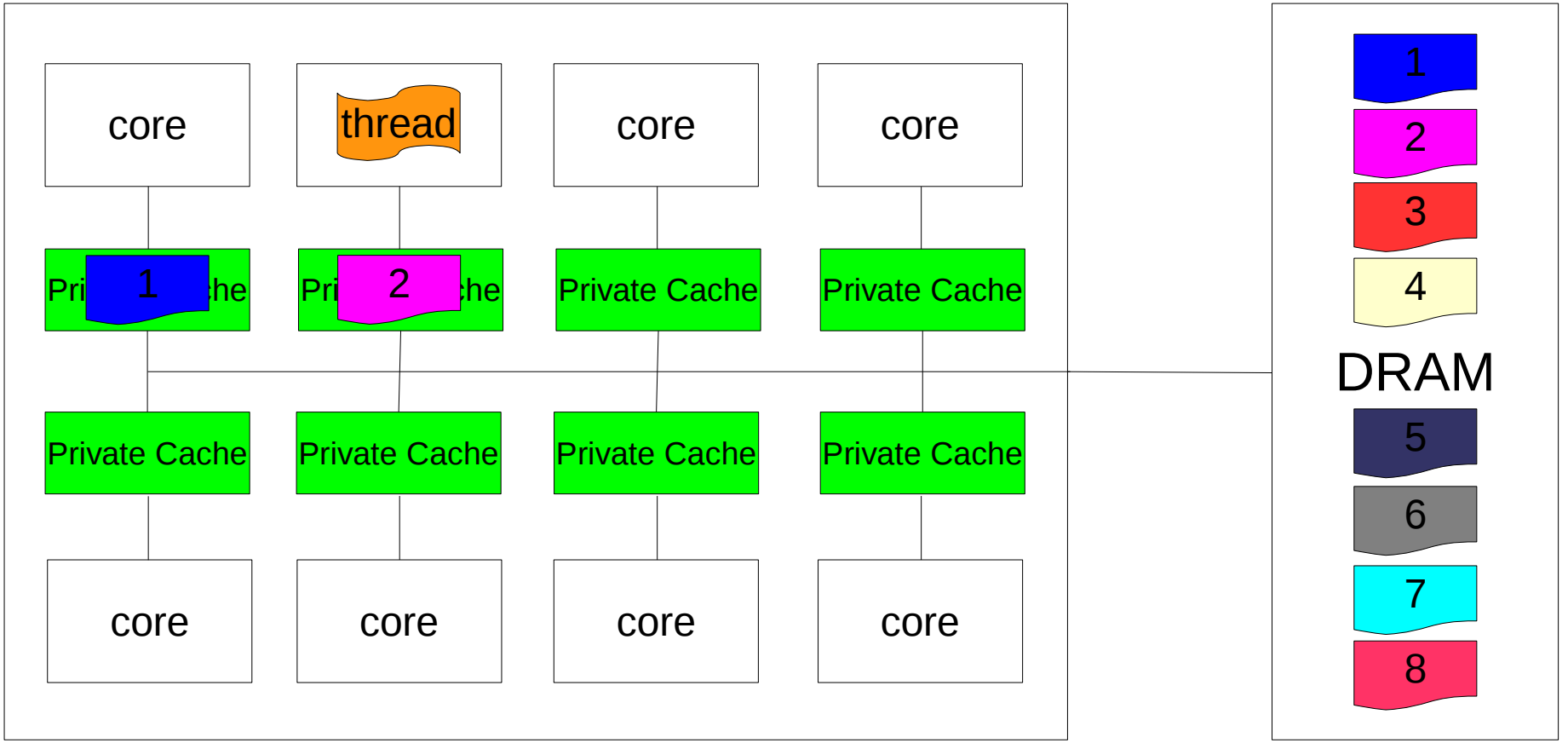
# Software solution



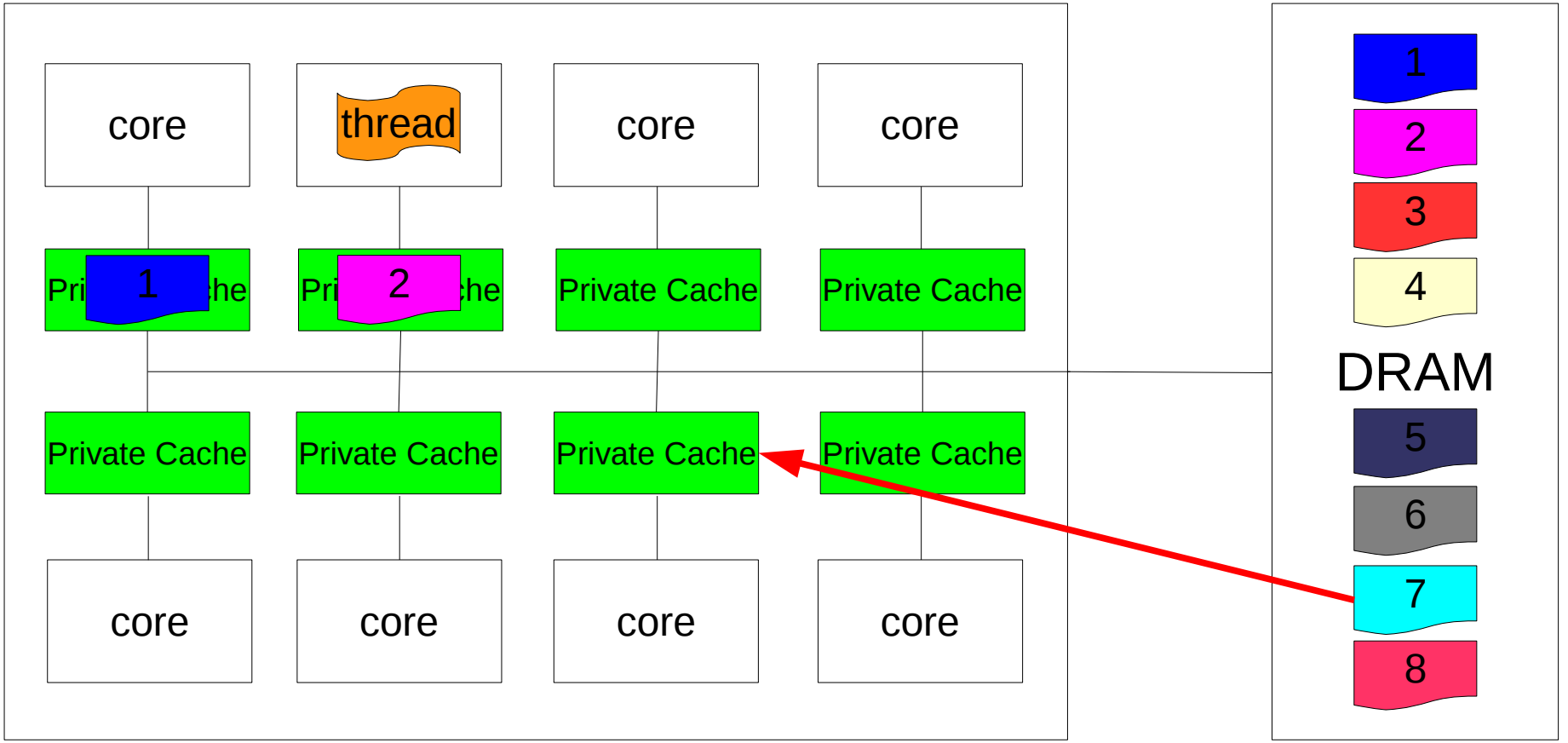
# Software solution



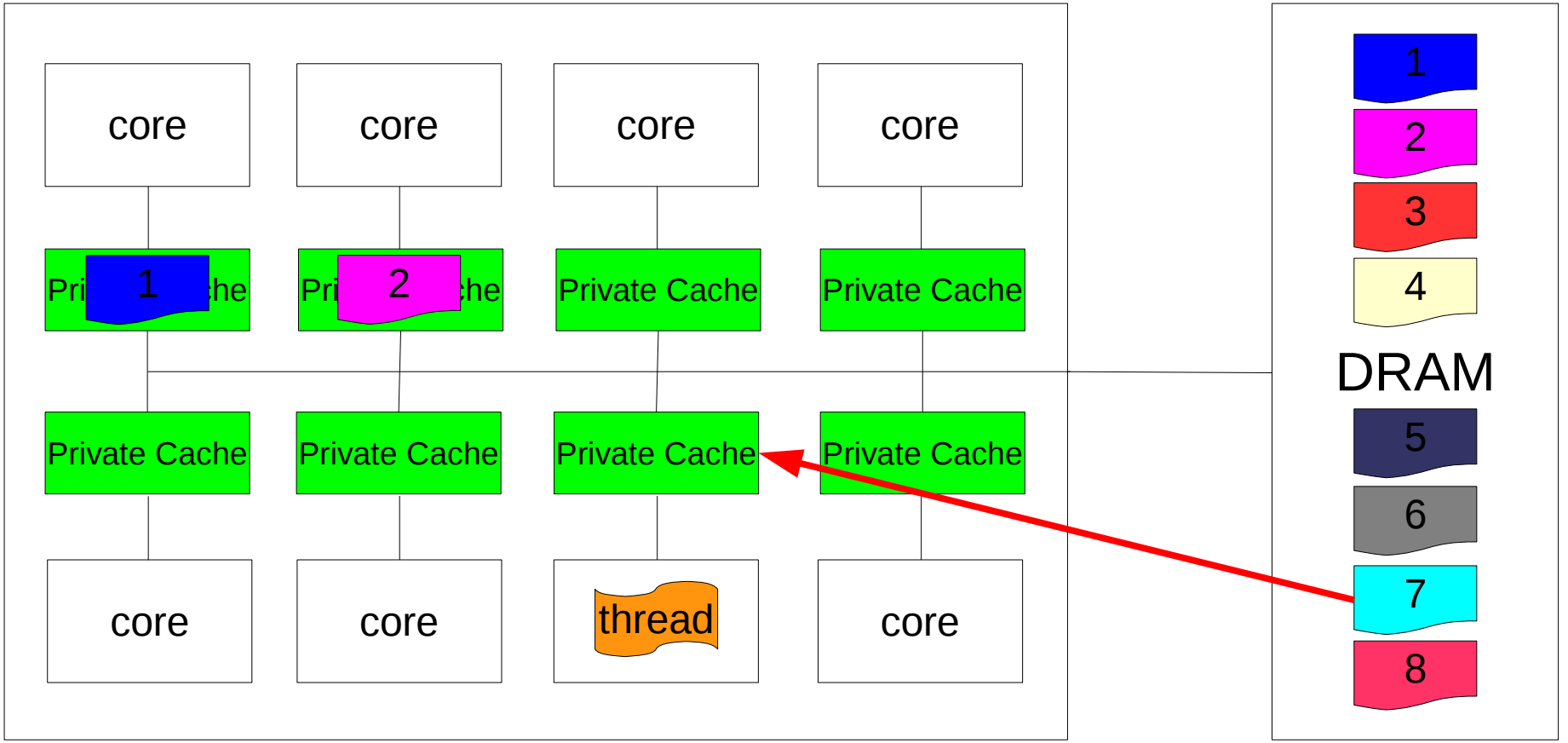
# Software solution



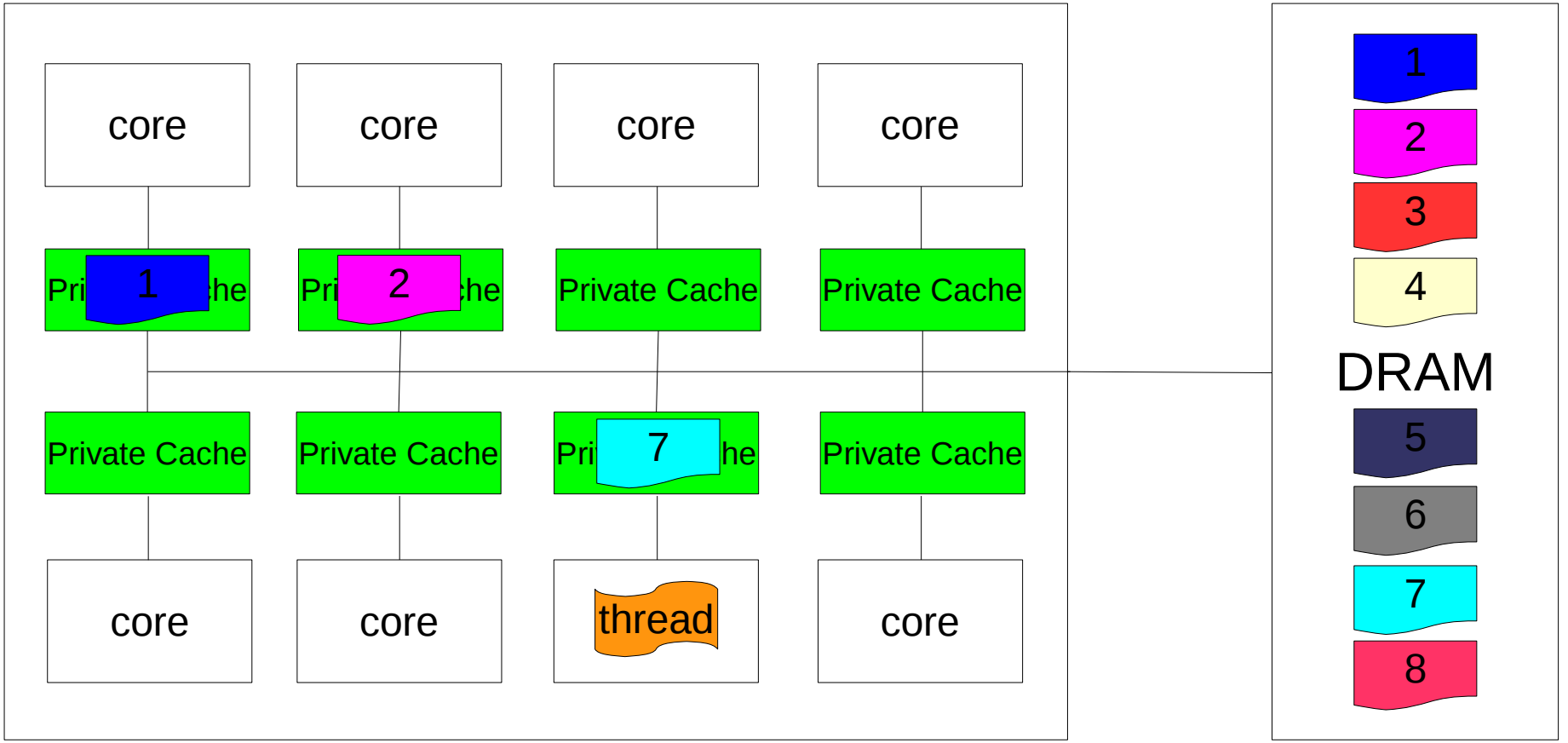
# Software solution



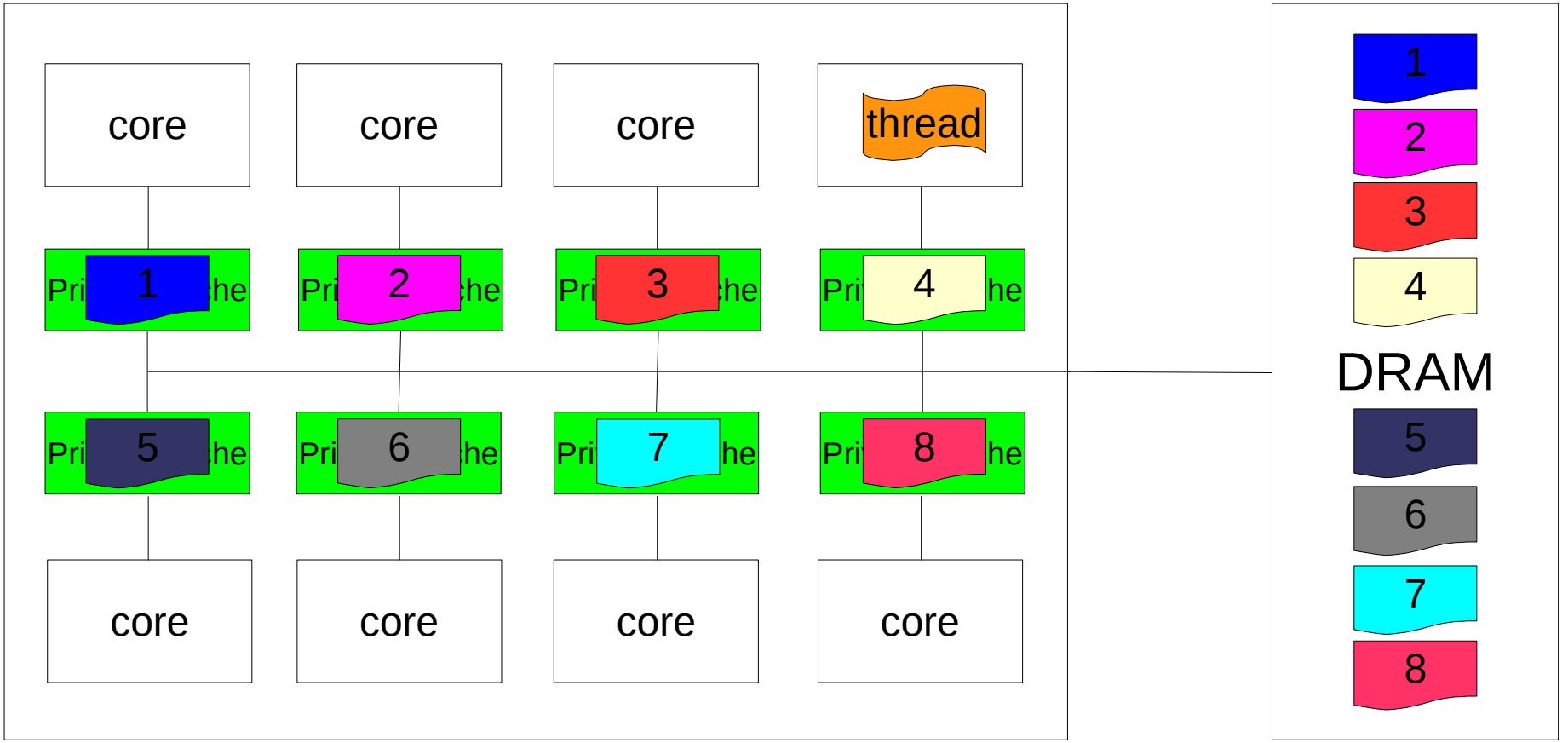
# Software solution



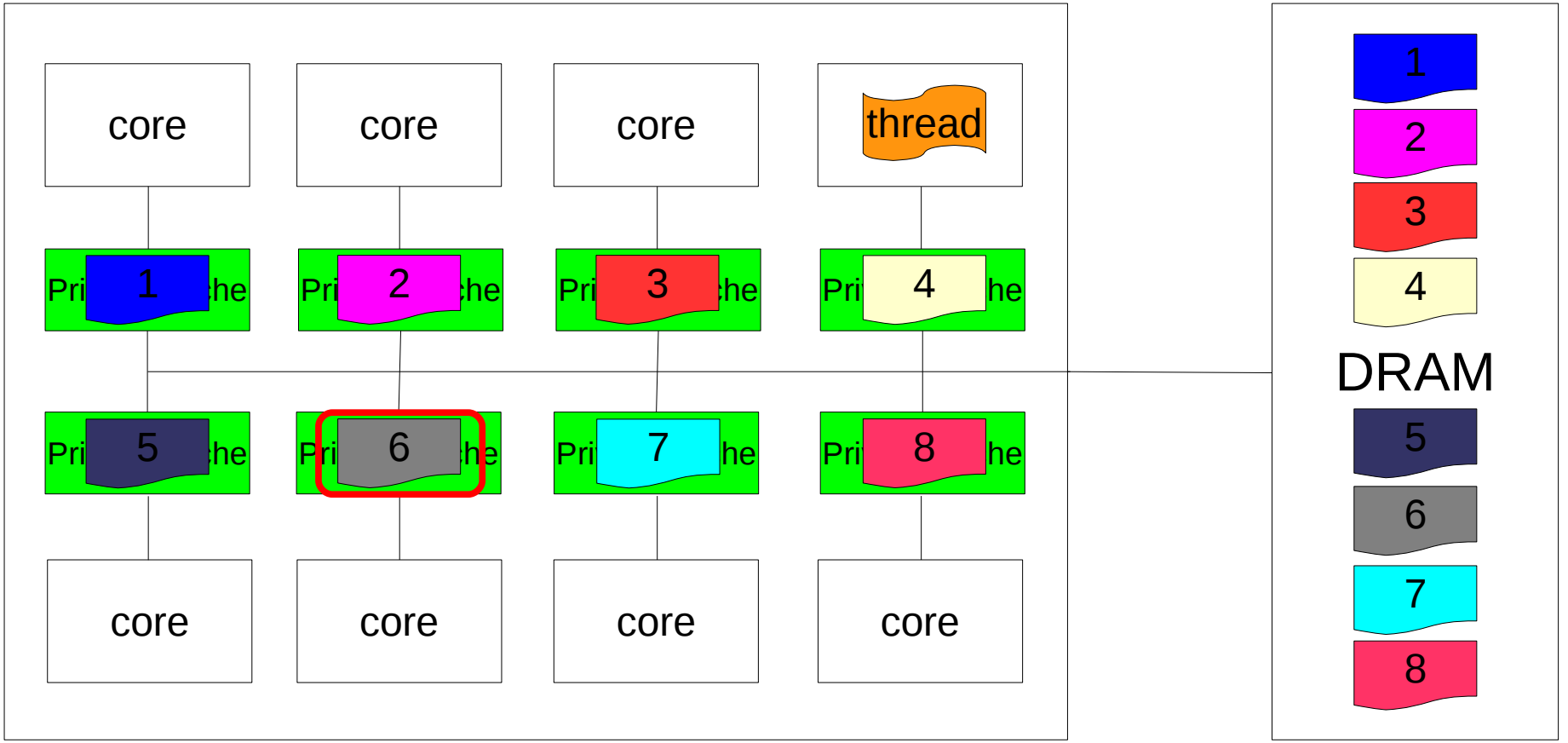
# Software solution



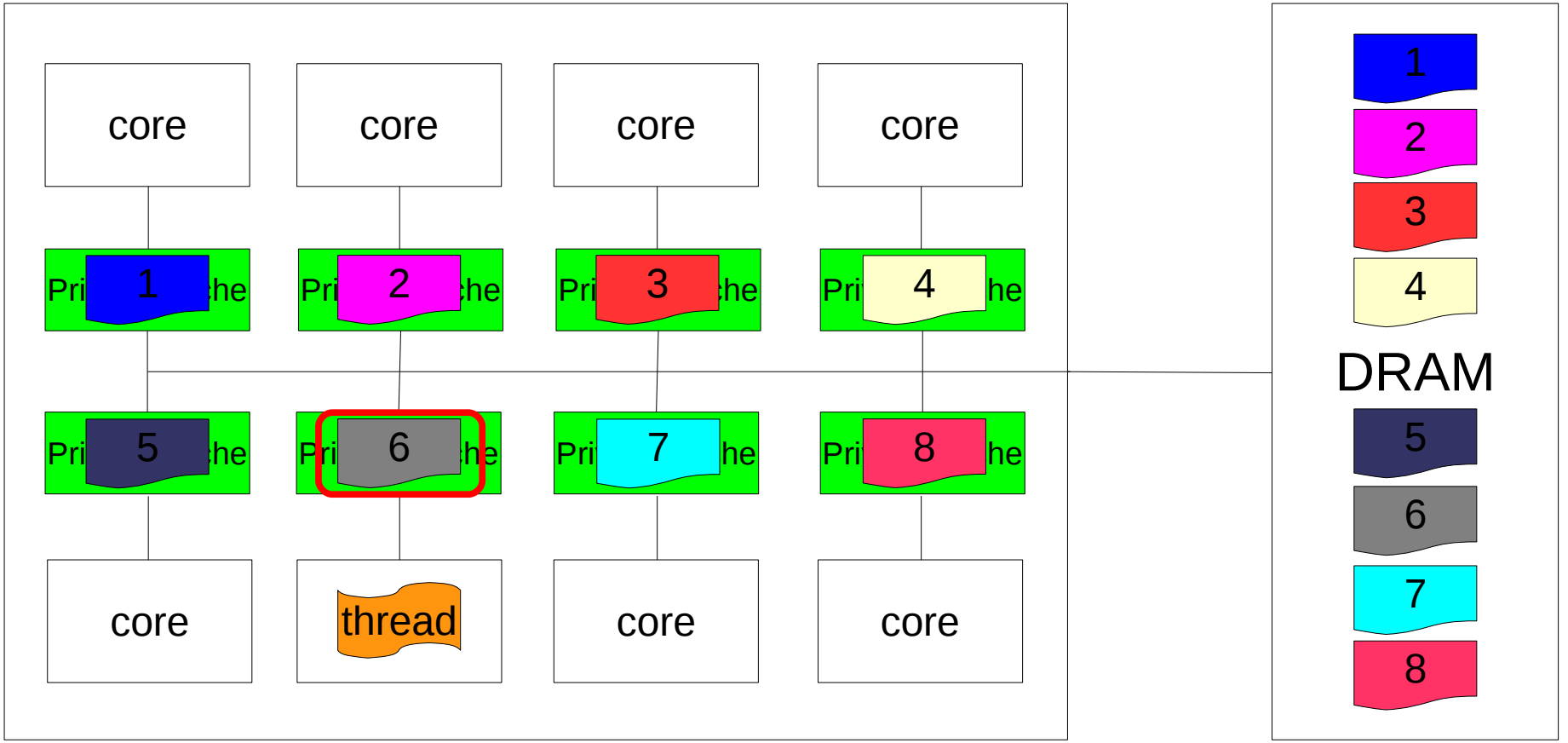
# Software solution



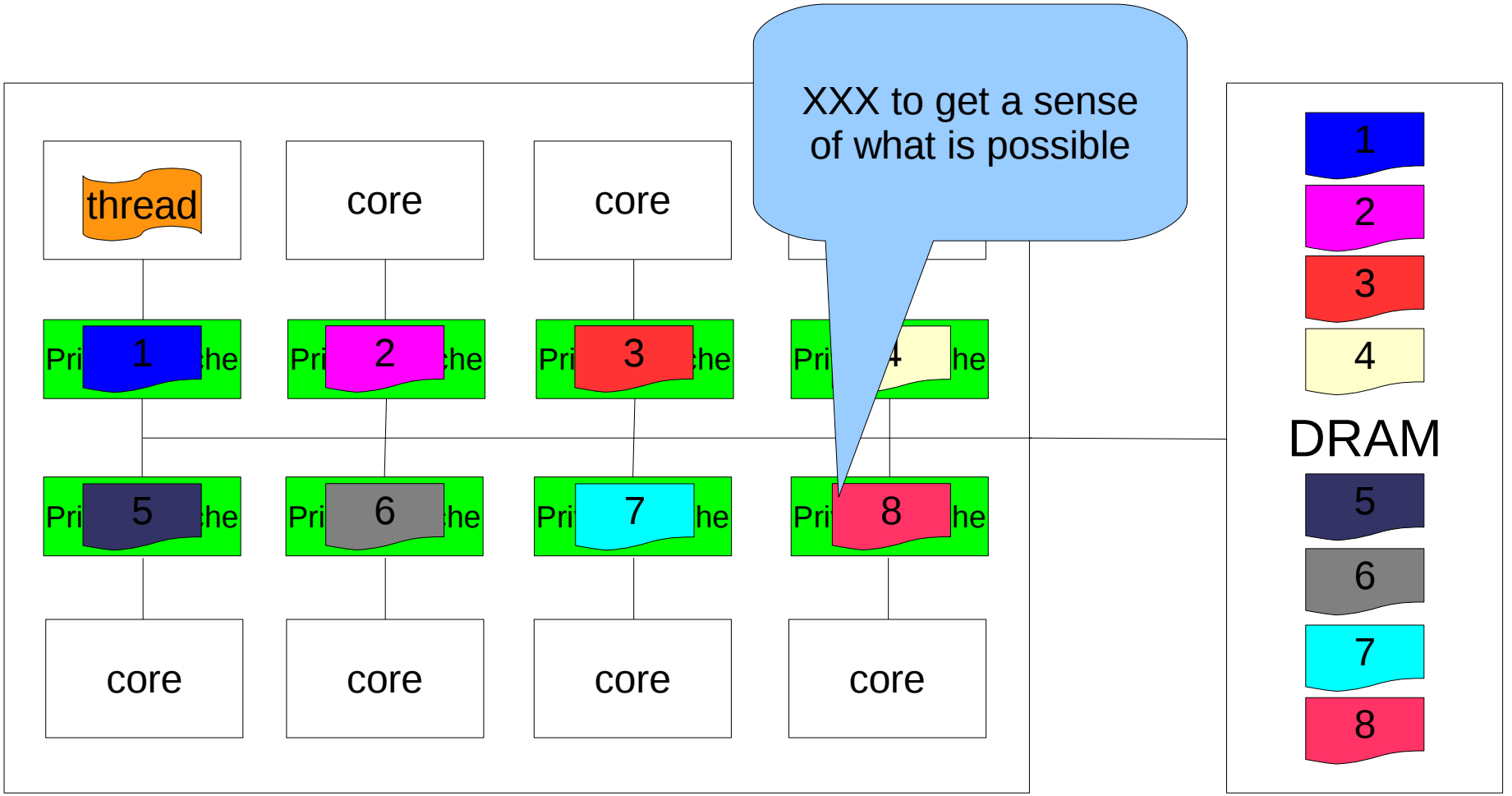
# Software solution



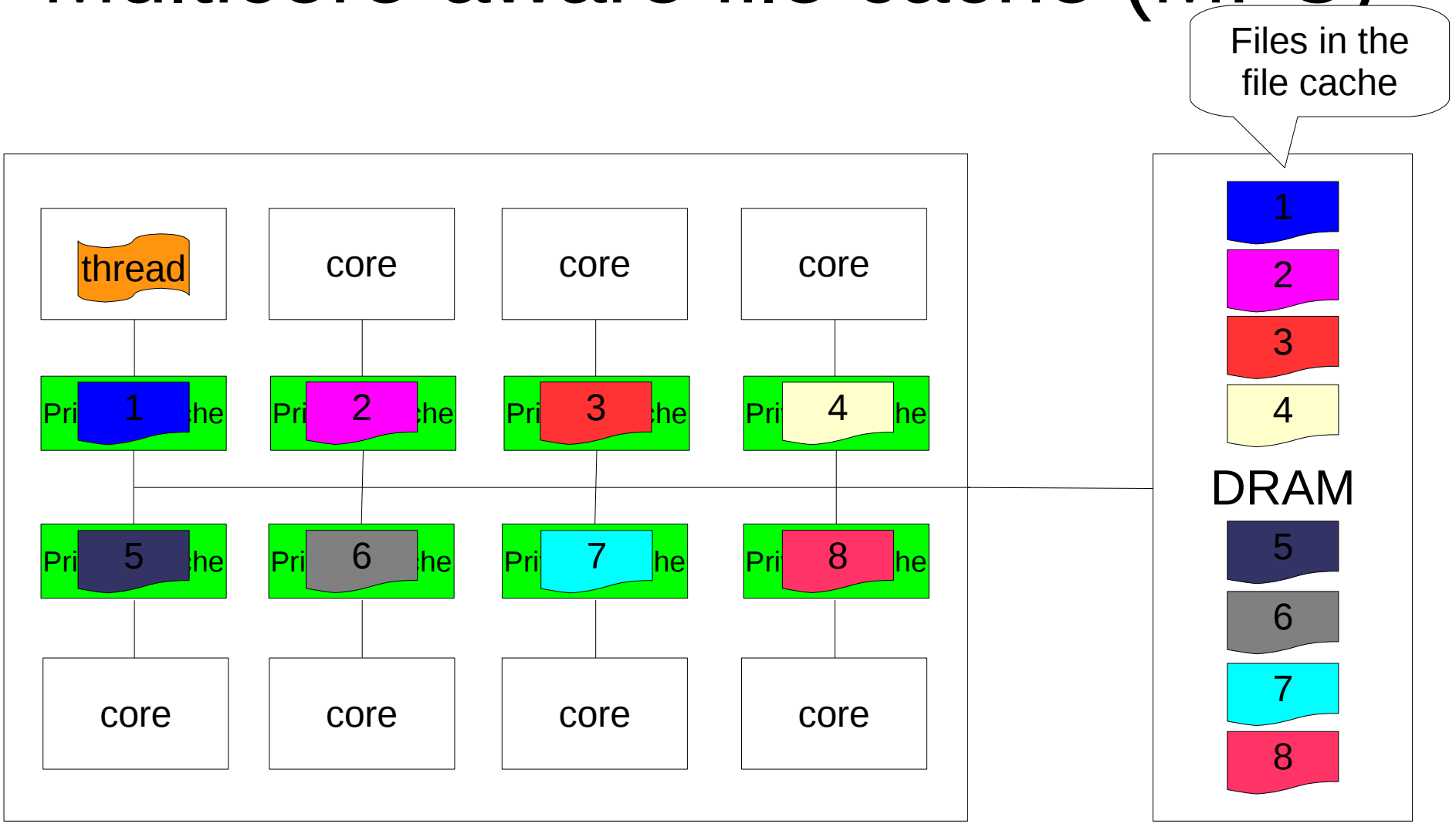
# Software solution



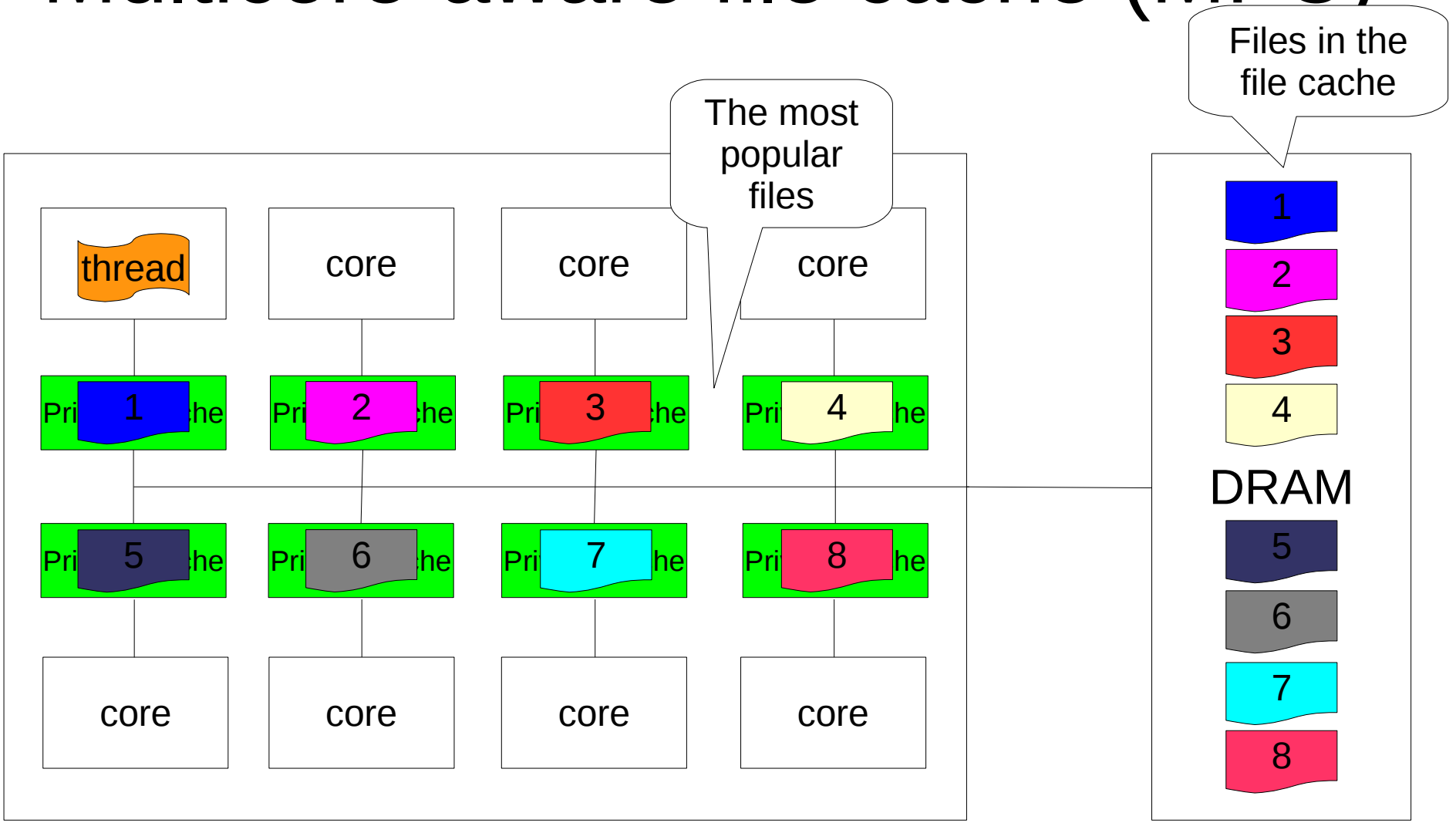
# Multicore-aware file cache (MFC)



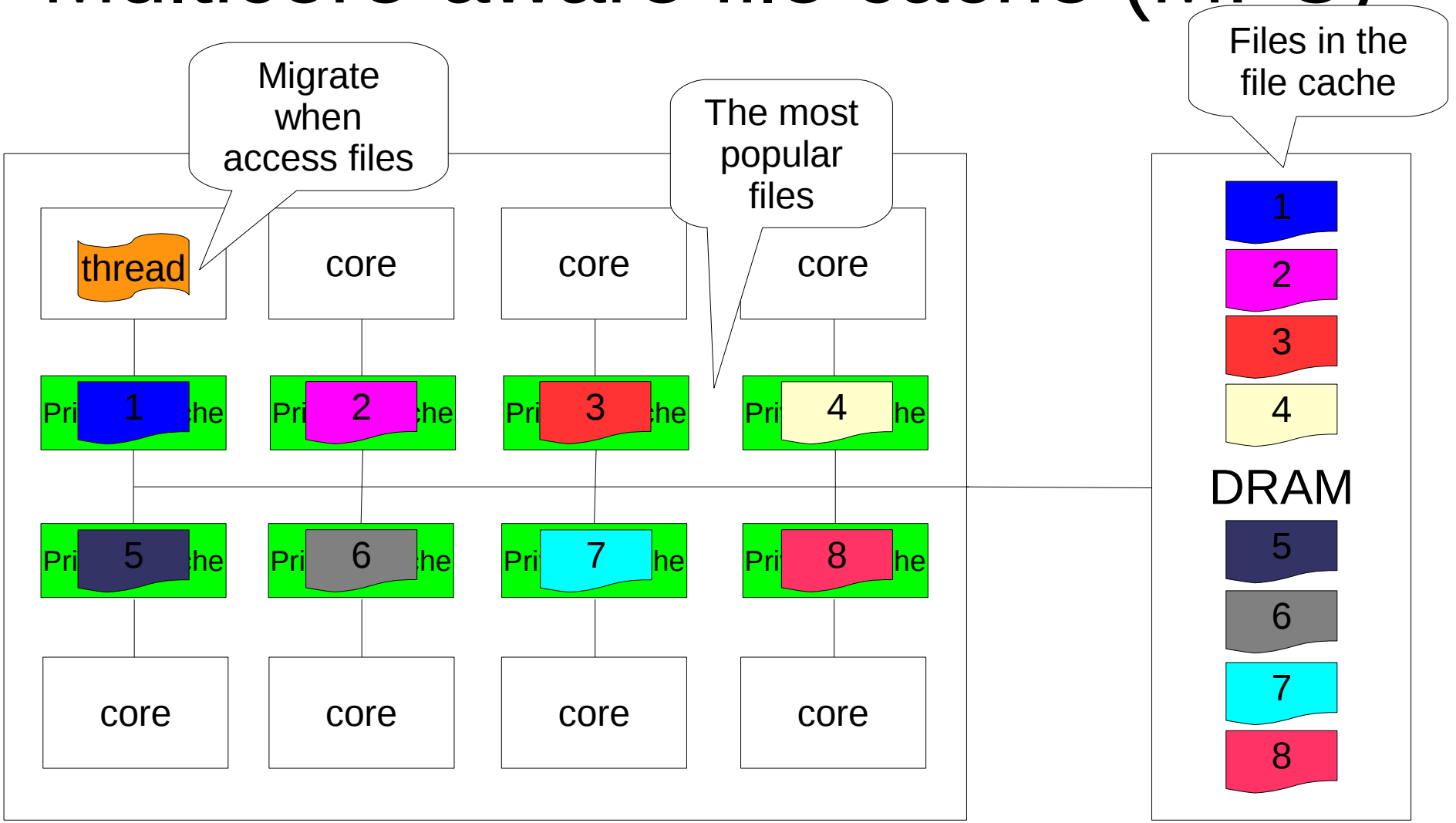
# Multicore-aware file cache (MFC)



# Multicore-aware file cache (MFC)



# Multicore-aware file cache (MFC)



# MFC challenges

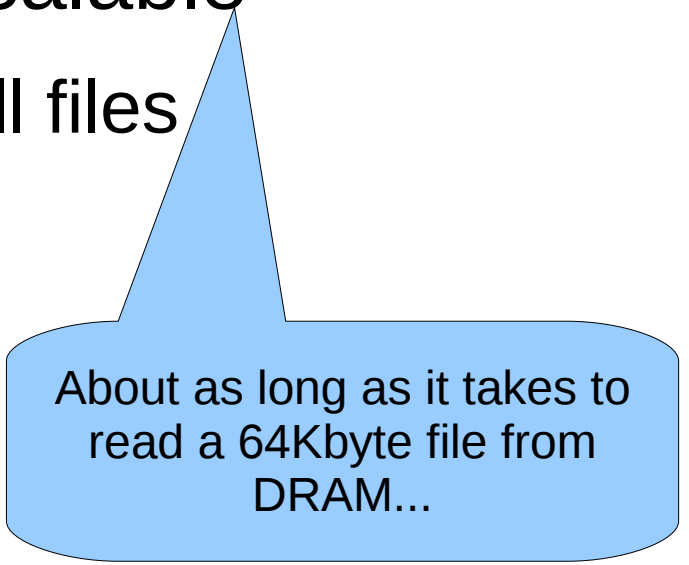
- Tracking files
  - MFC metadata
- Very popular files
  - Replicate
- Unpopular files
  - Read with non-caching loads
- Sharing on-chip caches with non-file data
  - Hardware event counters
- ...

# MFC implementation

- Implemented on top of the Linux file cache
- Modified `read` to invoke the MFC
  - Extended kernel API to support `mmap`
- About 1,000 lines of C

# Implementing migration in Linux

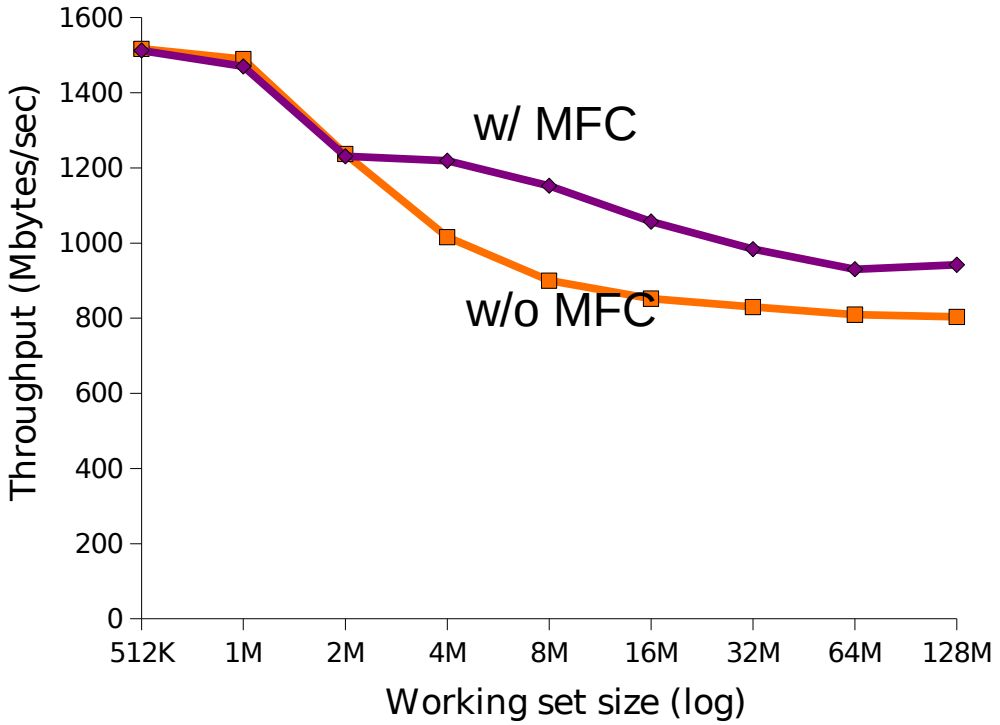
- Linux has a run queue per-core
  - MFC adds a thread to the target core's run queue
- Cost is 9 microseconds, not scalable
  - So don't migrate to access small files



# Does MFC improve file reading speed?

- Compare Linux w/ MFC and w/o MFC.
- grep reads random files
  - Zipfian file selection
- 16-core AMD, 16Mbytes cache

# MFC gets speedup with one thread

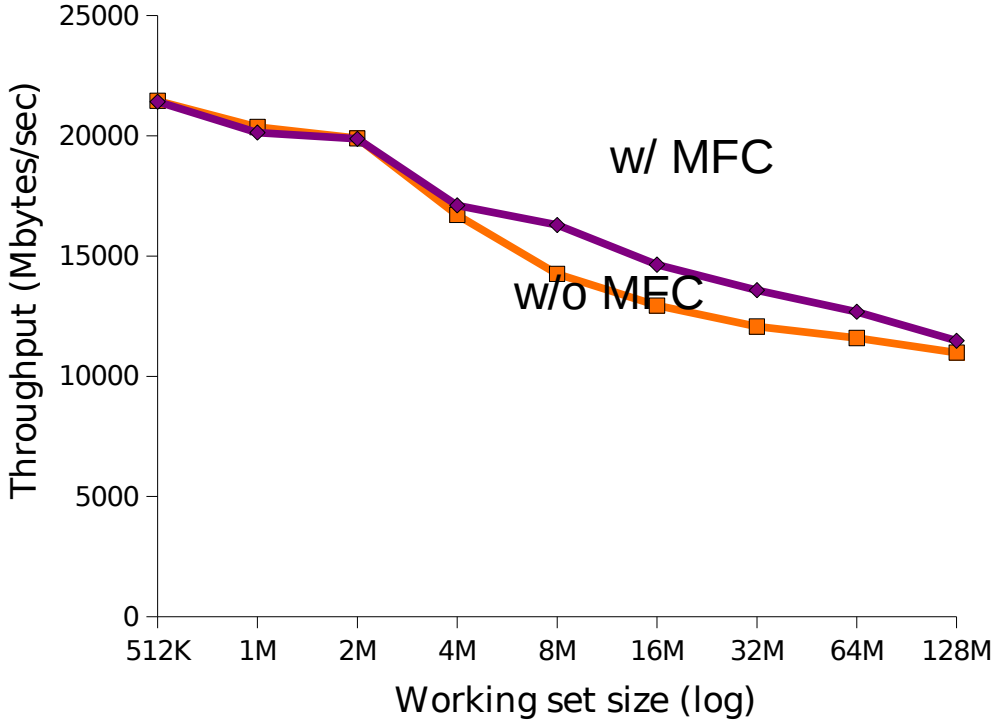



- Improves single thread performance
  - Large working set

# Does MFC improve multi-threaded workloads?

- Ran multiple instances of grep
  - Generate parallel workload for MFC

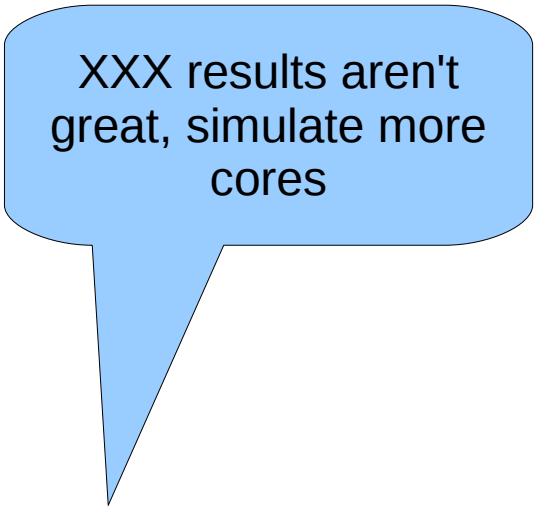
# MFC gets speedup with multiple threads



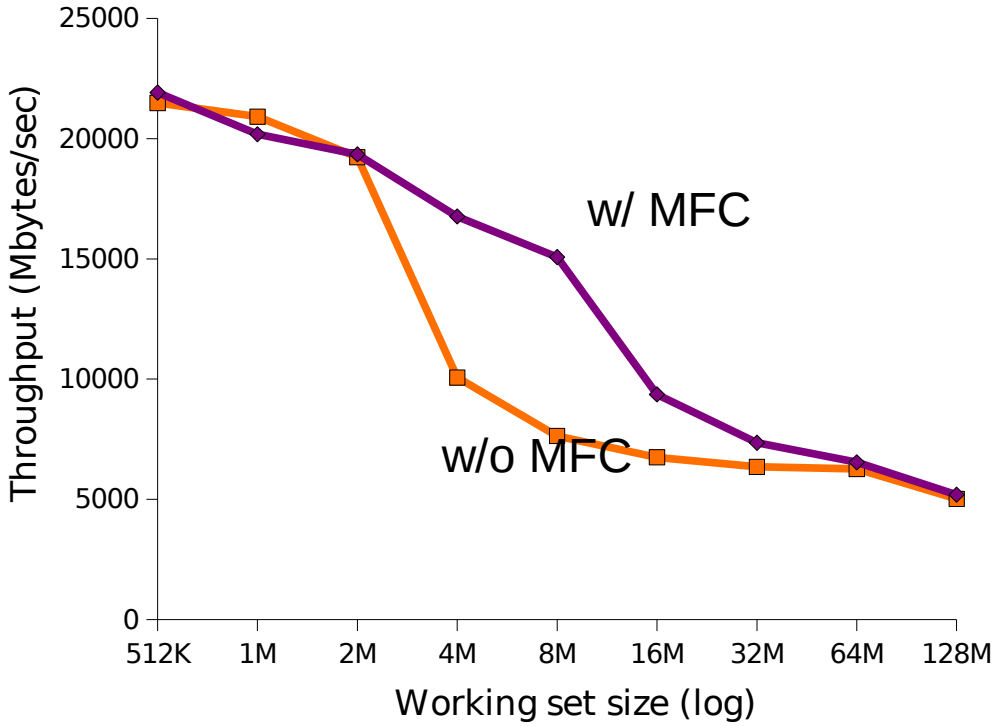

- Not as much as expected
  - Linux thread migration doesn't scale

# Will MFC matter for future multicores?

- Future chips will have more cores
- But DRAM bandwidth won't scale
- Simulate this future by disabling DRAM controllers



# Expect more speedup on future chips



# Generalizing MFC

- Showed one experiment with MFC
- Believe ideas generalize to more applications
  - ClamAV
  - Apache
  - Linux
  - ...
- Do these ideas apply to your apps?

# Related work

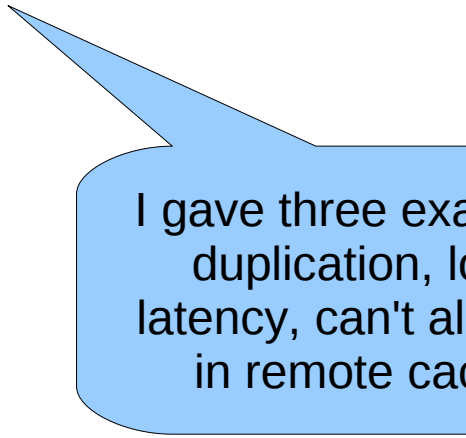
- Function shipping is an old CS trick
  - Work on NUMA OSs
- Using caches efficiently
  - Cache partitioning
  - Minimizing cache conflicts
  - Scalable locking
  - ...

XXX differe
hardware, cor
addressabl
cache line sm
than page

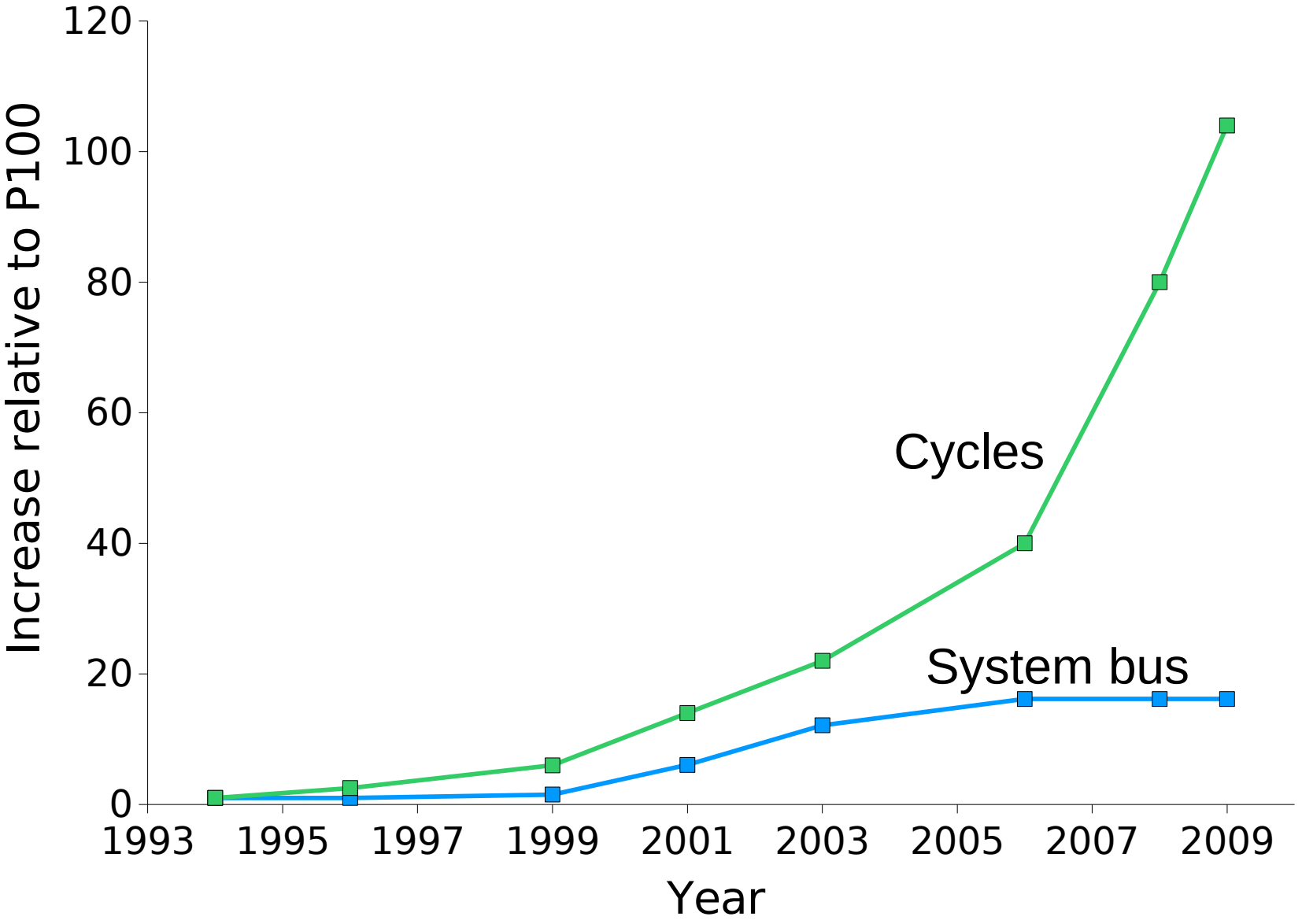
Designing software for multicore hardware that uses entire cache capacity

# Conclusion

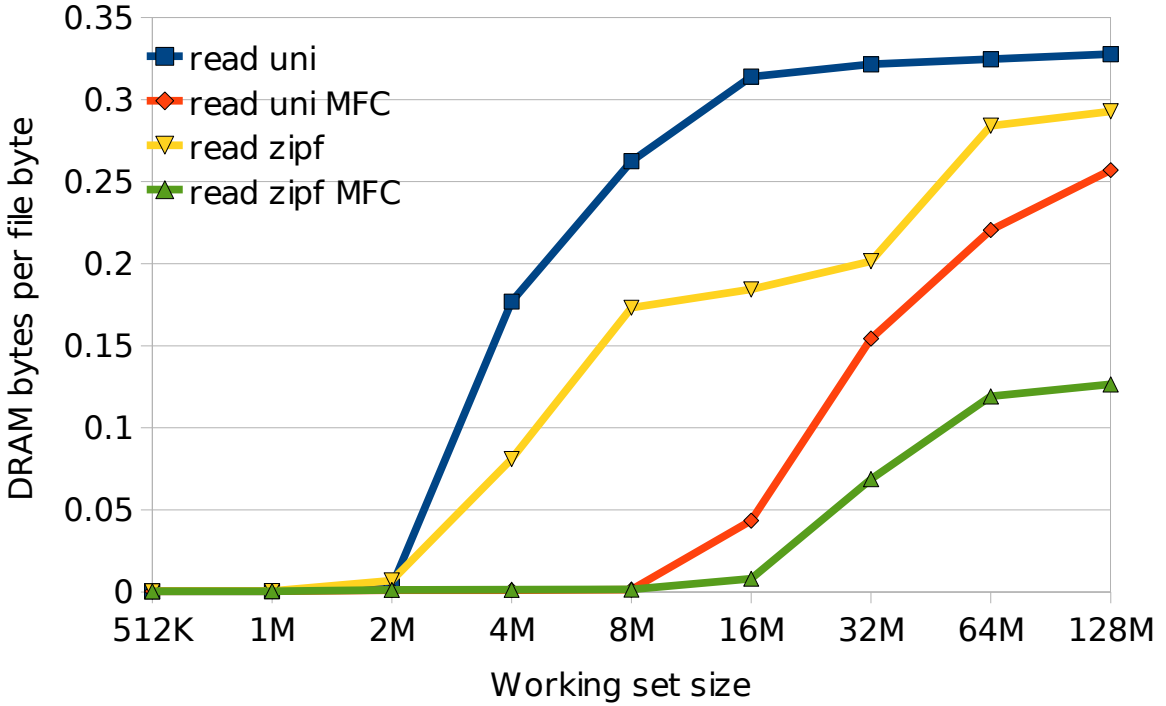
- Improve performance by managing multicore caches
  - Simple uniprocessor model doesn't work
- MFC is one approach
  - Techniques should be generalizable



# DRAM vs CPU trend



# DRAM loads




- MFC makes fewer loads from DRAM
  - Even for large working set sizes