CERN School of Computing

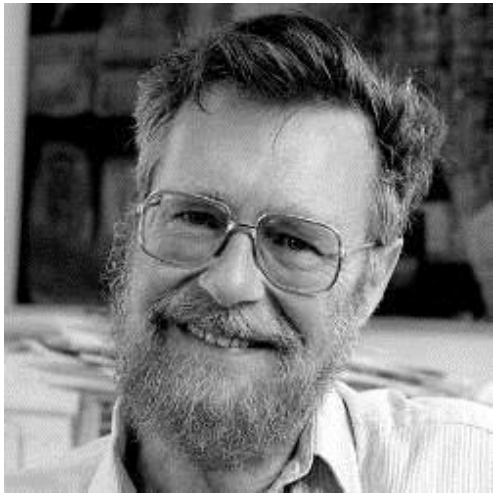
Shared memory and Message passing revisited in the many-core era

Aram Santogidis

CERN

Inverted CERN School of Computing, 29 February – 2 March 2016

The pioneers of concurrent programming

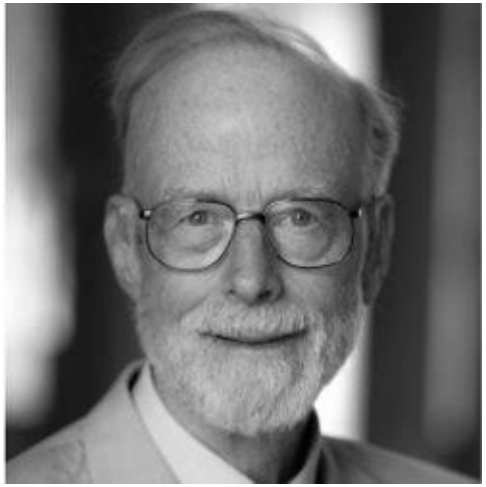


Edsger Dijkstra

- Mutual exclusion
- Cooperating Sequential Processes
- Semaphores

Per Brinch Hansen

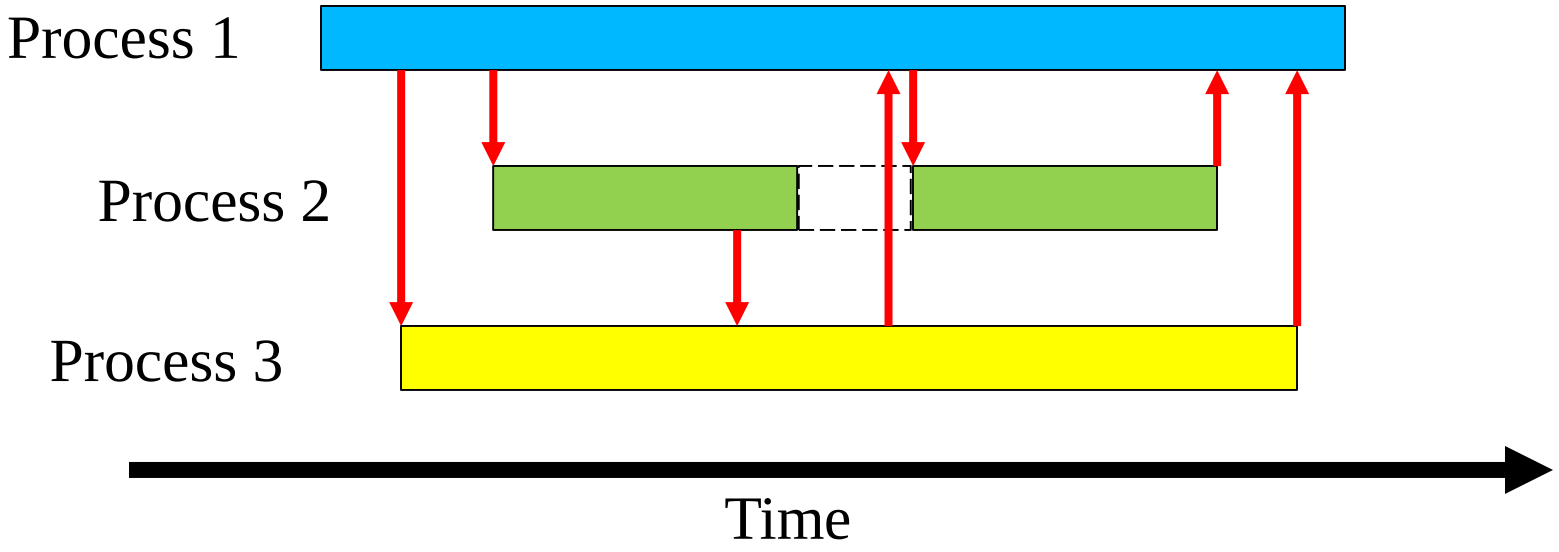
- Concurrent Pascal
- Shared Classes
- The Solo OS
- Distributed Processes

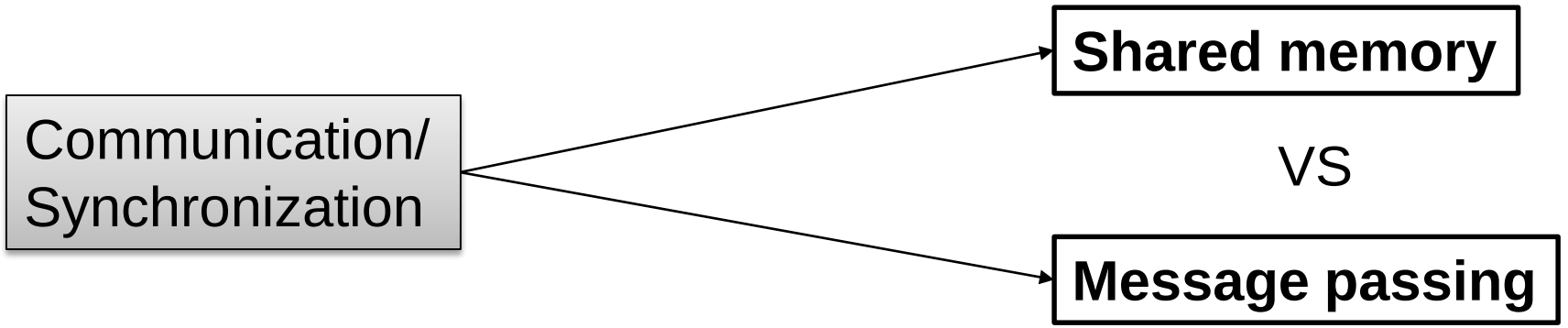


C.A.R Hoare

- Communicating Sequential Processes (CSP)
- Monitors

Communication is important



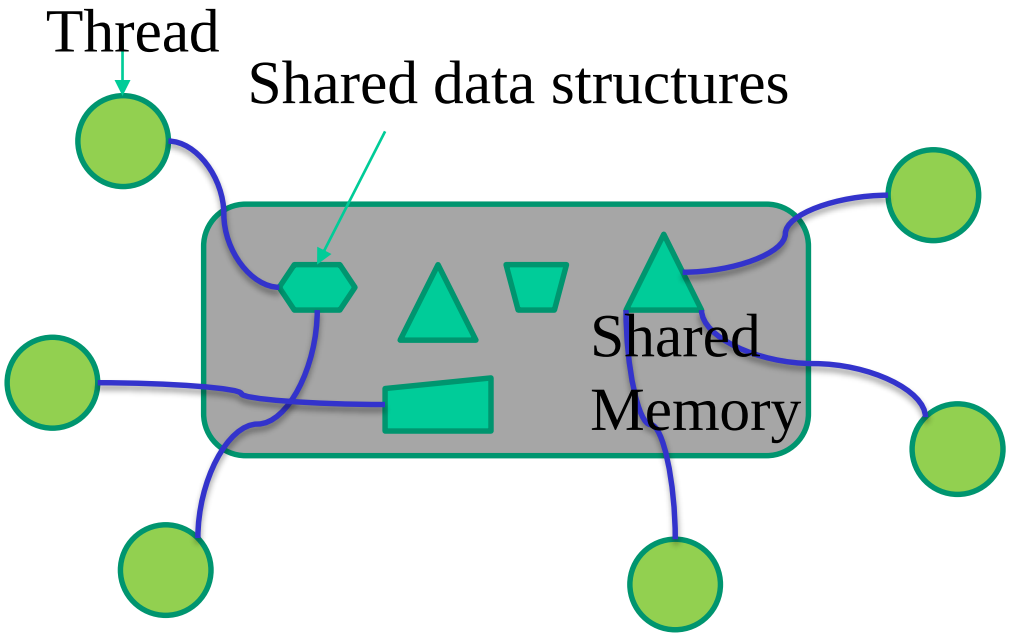


Agenda of the talk

- **Concurrency and communication**
- **Two basic examples of the two models**
- **Conventional wisdom for the two models**
- **Cache coherence and manycore processors**
- **Emerging paradigm shift in OS architectures**
- **The future perspective**

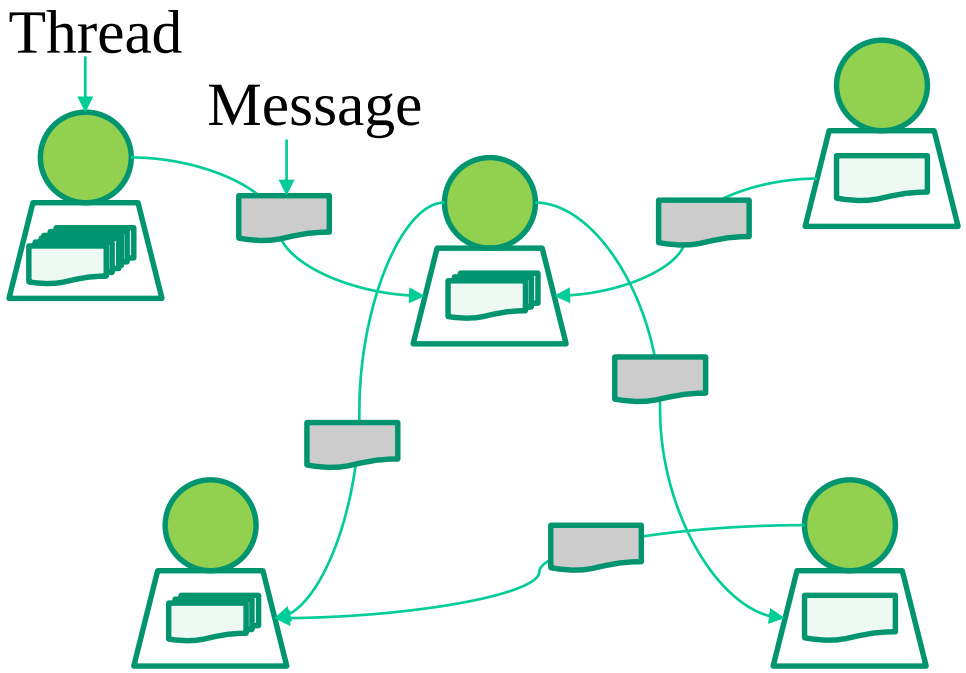
The Shared Memory model

- Threads communicate **implicitly** with each other via shared data structures
- Synchronization primitives (locks, semaphores, etc.)



The message passing model

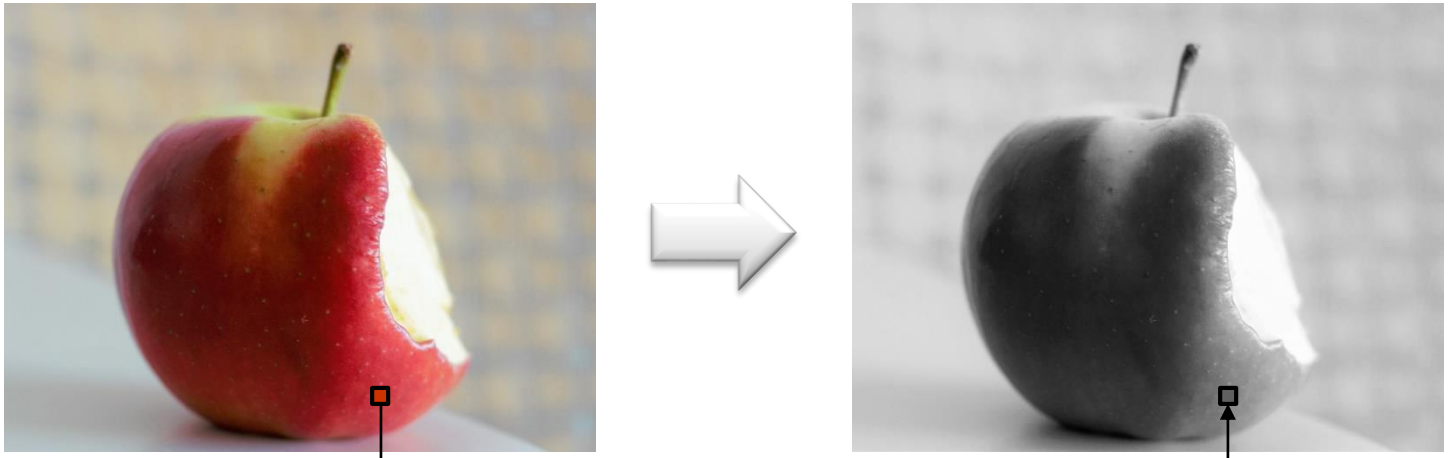
- Threads communicate **explicitly** with each other by exchanging messages
- Is the more **fundamental** class from the two
- Synchronous or asynchronous communication



Lets see an example for each model

1. **Image processing (shared memory)**
2. **Simple GUI (message passing)**

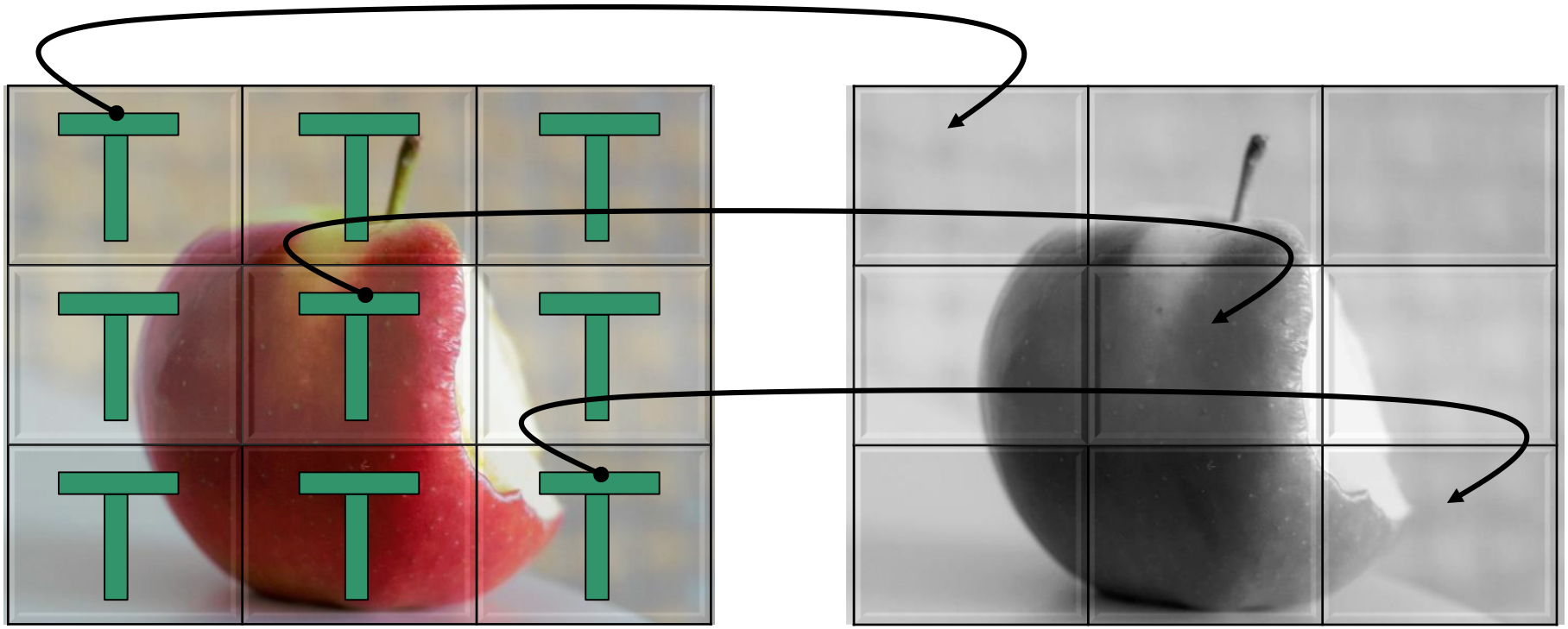
A shared memory-based example: Convert from colour to grayscale



A shared memory-based example: Convert from colour to grayscale

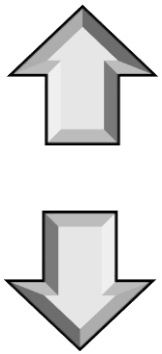
SPEED

We parallelize the computation by assigning tiles (pieces) of the image to threads which execute the conversion in parallel.



A message passing-based example

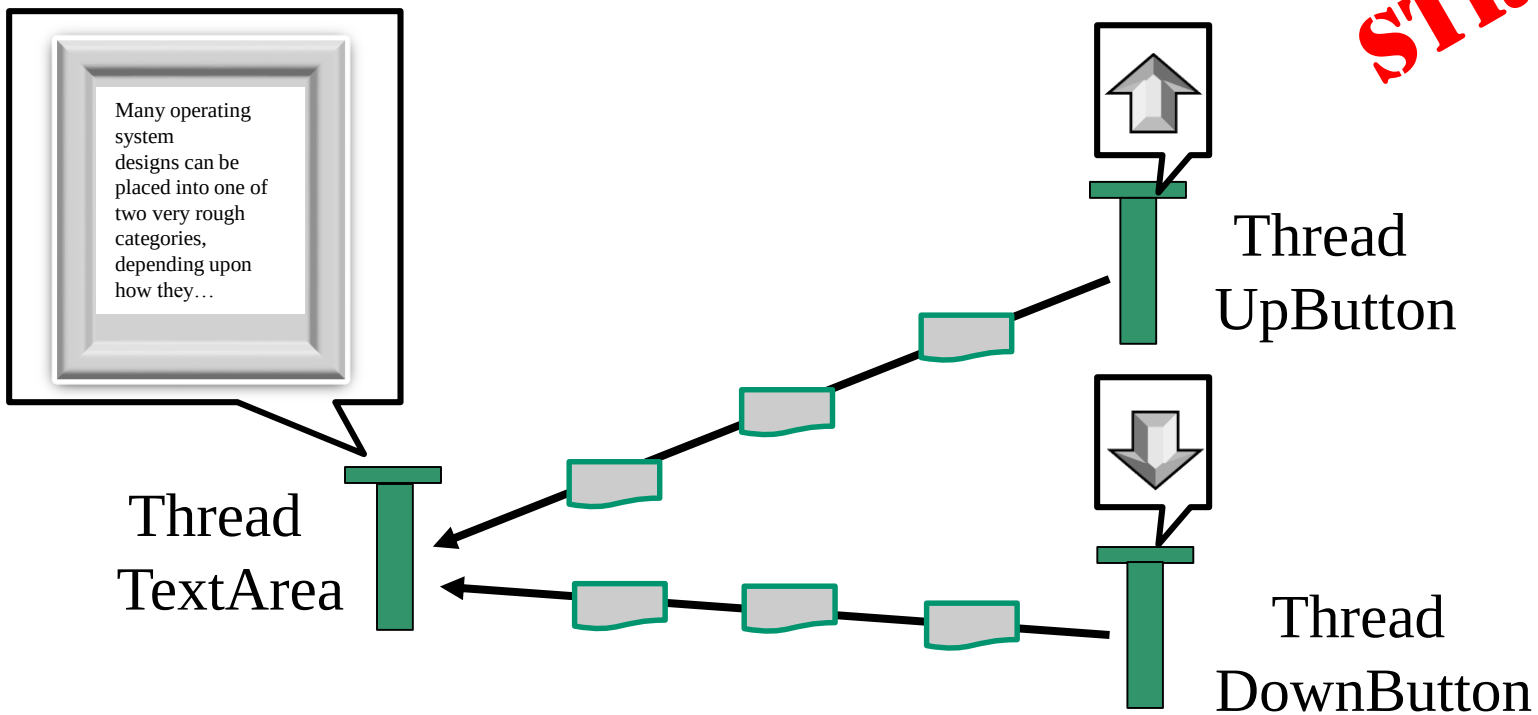
Many operating system designs can be placed into one of two very rough categories, depending upon how they…



- A GUI with 3 widgets
  - Text Area
  - Up scroll button
  - Down scroll button
- Must be interactive (Immediate feedback)

GUI example implementation: Message passing solution

STRUCTURE



Conventional wisdom about the characteristics of the two models

1. **Performance**
2. **Programmability**

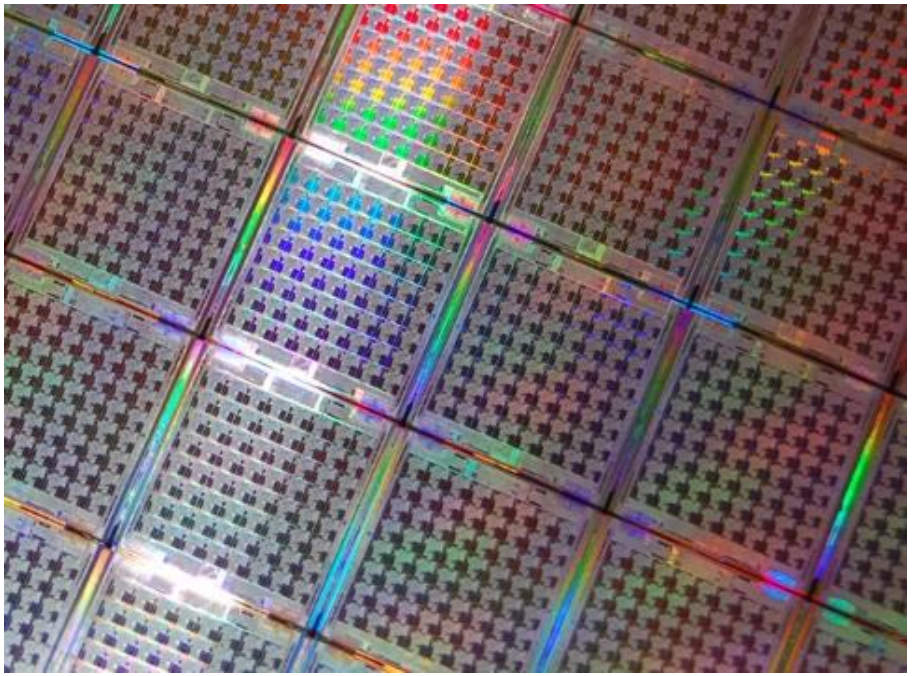
Performance comparison

	Shared Memory	Message Passing
Hardware support	**Extensive** (All popular architectures)	**Limited** (Only special purpose architectures)
Data transfer Overhead	**Low** (Cache block management in HW)	**High** (Data replication)
Access/Sync overhead	**Sometimes high** (Critical section contention, NUMA effects)	**Low** (Local private memory access)

Programmability comparison

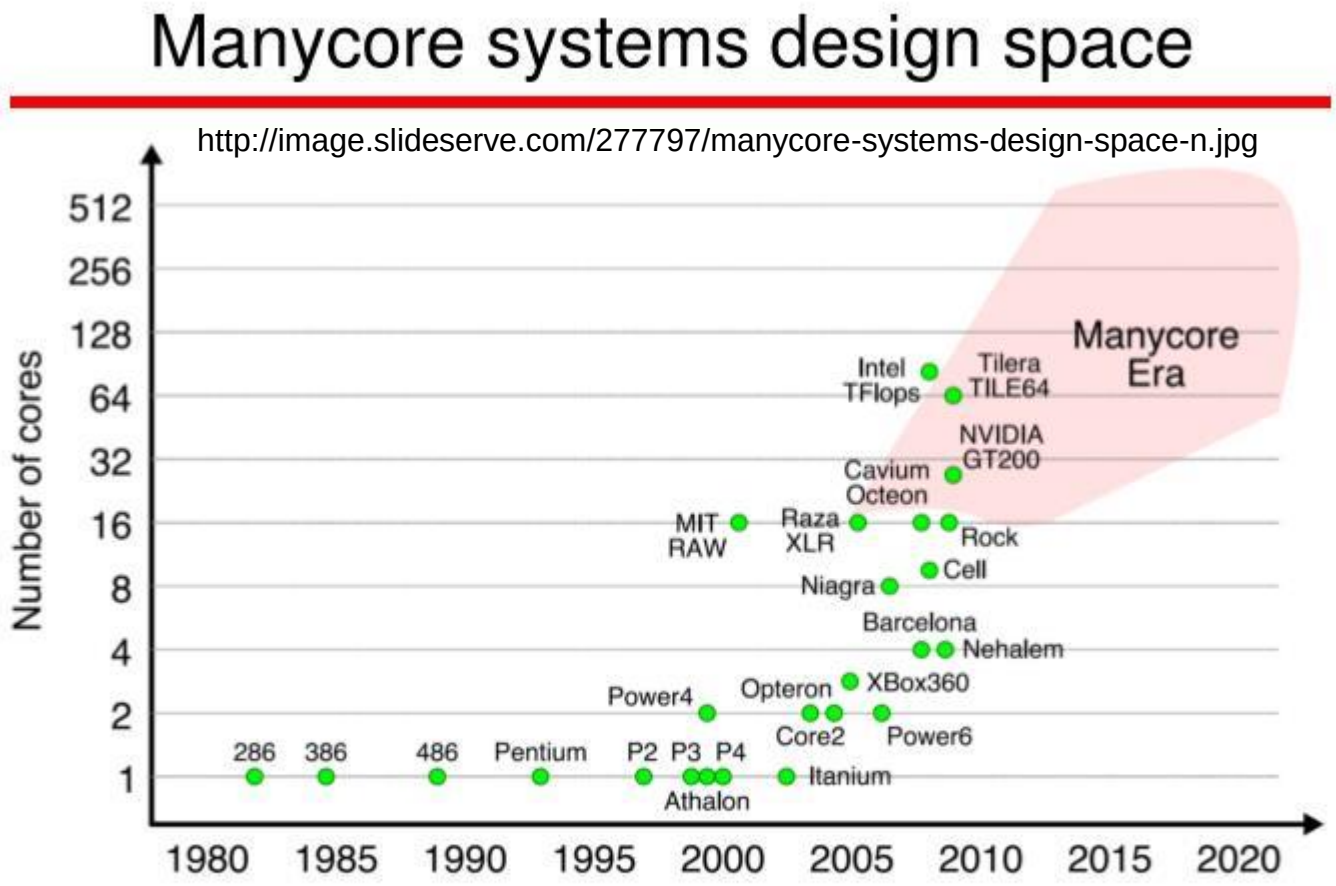
	Shared Memory	Message Passing
Communication	**Implicit**	**Explicit**
Synchronization	**Explicit** (locks etc.)	**Implicit** (side-effect)
Interface (API)	Read/write shared data structures, mutex primitives	Send/Receive messages, Multicast
Hazards	Race conditions, Deadlocks, Starvation	Deadlocks, Starvation

Towards the manycore architectures



http://www.wired.com/images_blogs/gadgetlab/2009/10/tilera-wafer-1.jpg

The manycore era



- **Power limits the frequency increase of the processor.**
- **Moore's law: The transistors keep doubling every two years**
- **Replication: Increasing number of cores**

The graph is from (presentation): "Joshi, Ajay, et al. "Building manycore processor-to-DRAM networks using monolithic silicon photonics." *High Performance Embedded Computing (HPEC) Workshop*. 2008."

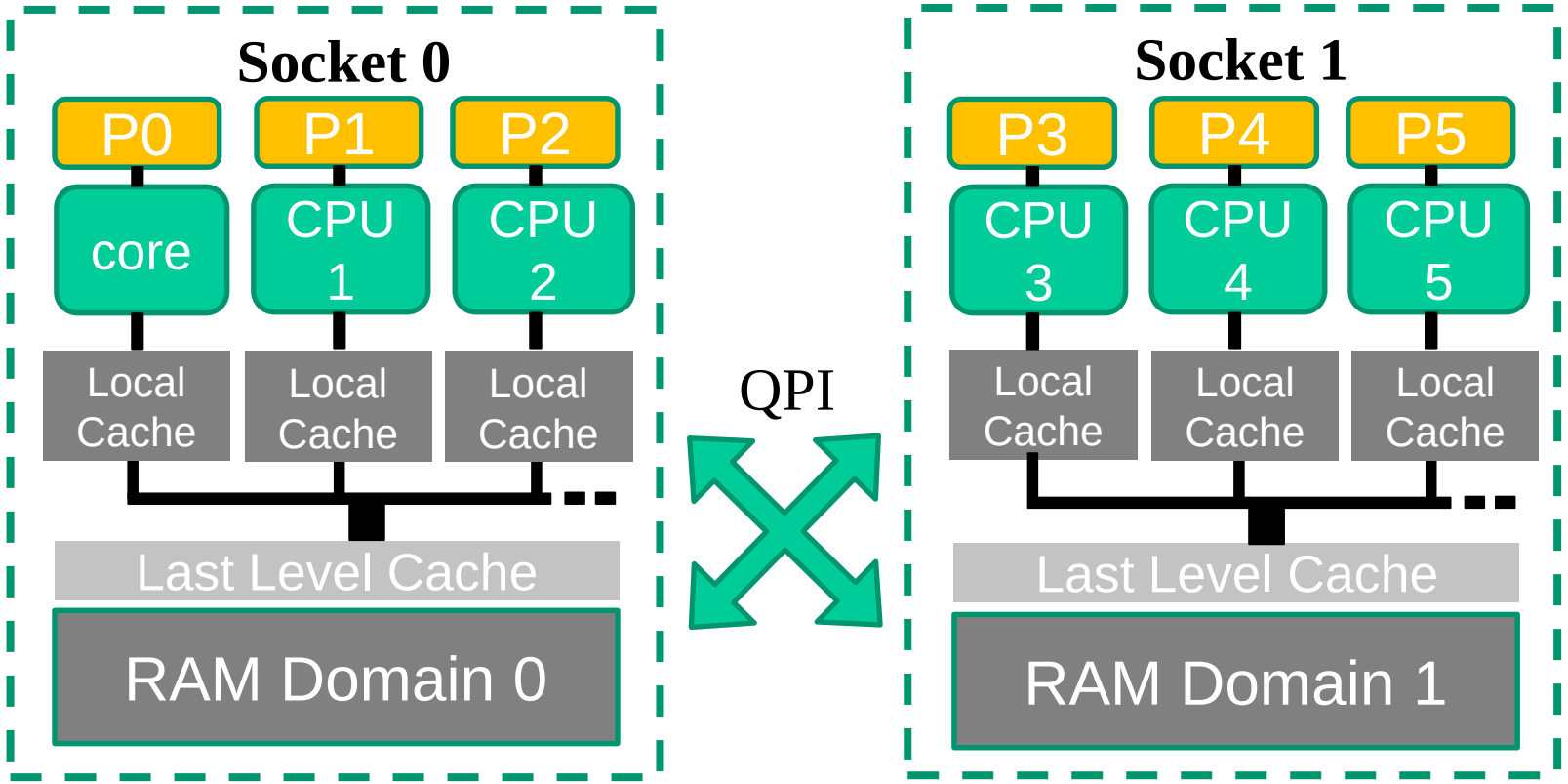
MIT/UCB

On the duality of operating systems structures

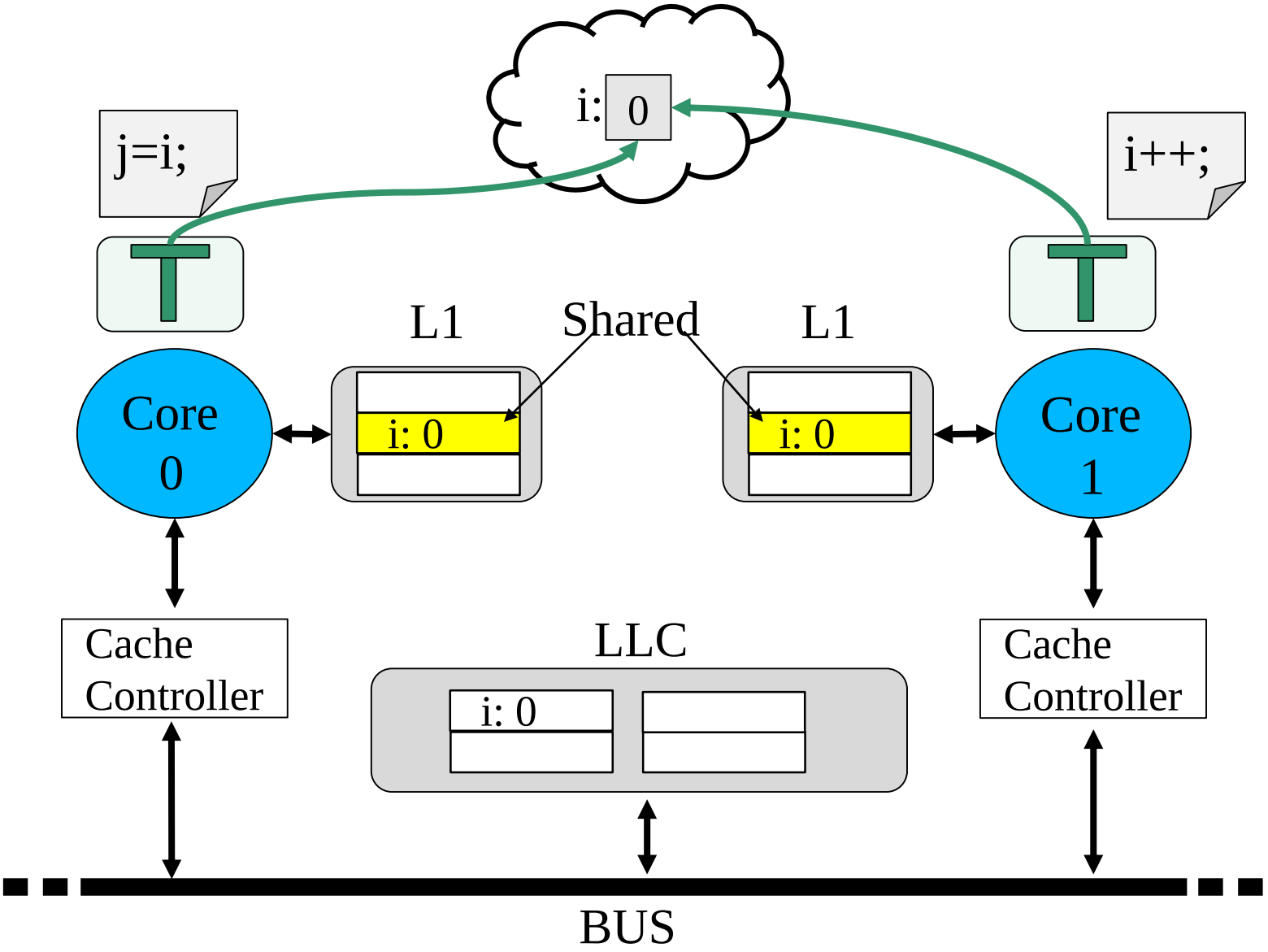
- **Operating Systems are generally classified as:**
  - Message passing oriented
  - Procedure-oriented (shared memory)
- **Each system from one category has the other category.**
- **Neither model is inherently better than the other (depends on the machine architecture).**

From: Lauer, Hugh C., and Roger M. Needham. "On the duality of operating system structures." *ACM SIGOPS Operating Systems Review* 13.2 (1979): 3-19.

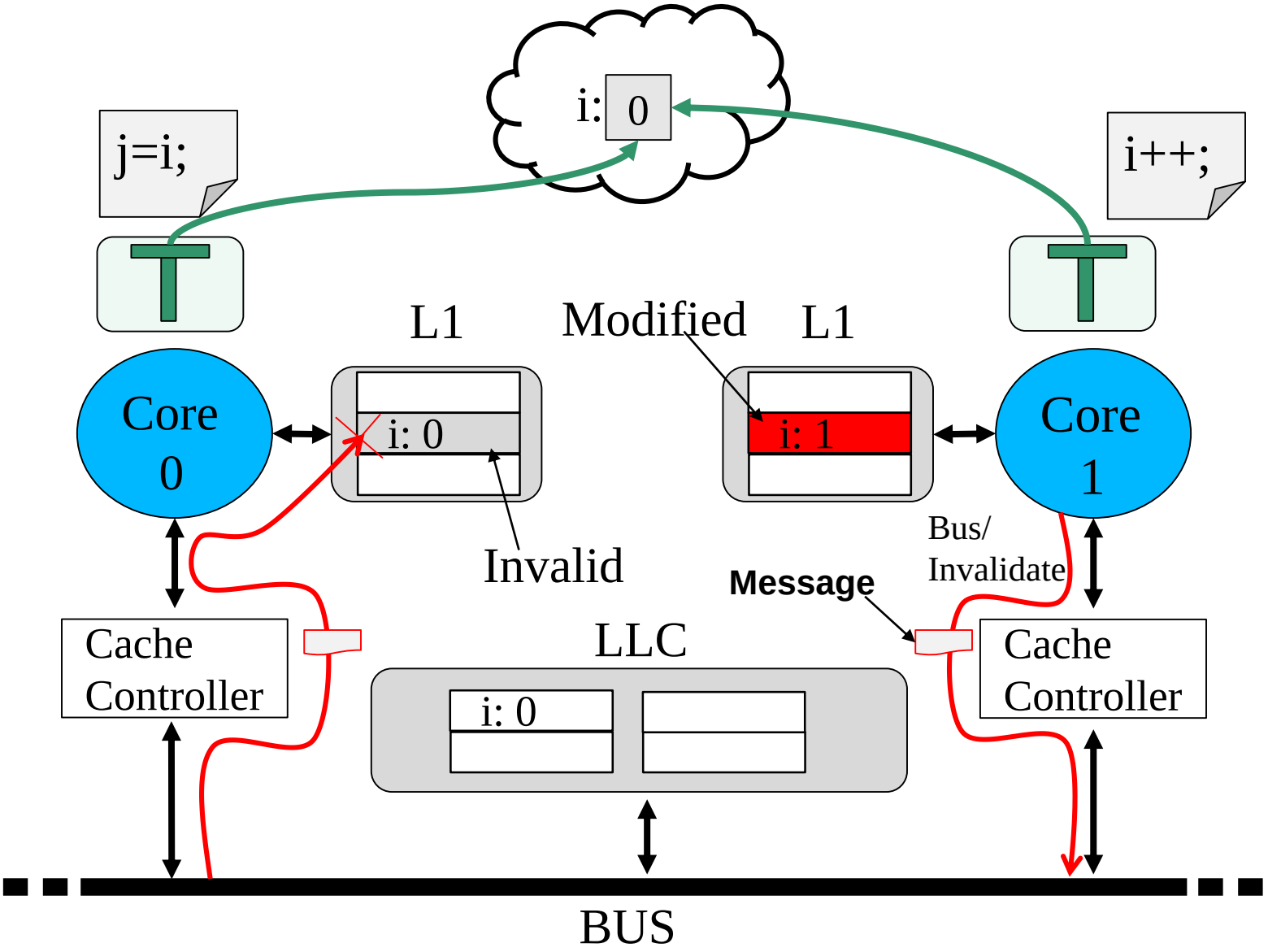
Non Uniform Memory Access (NUMA)



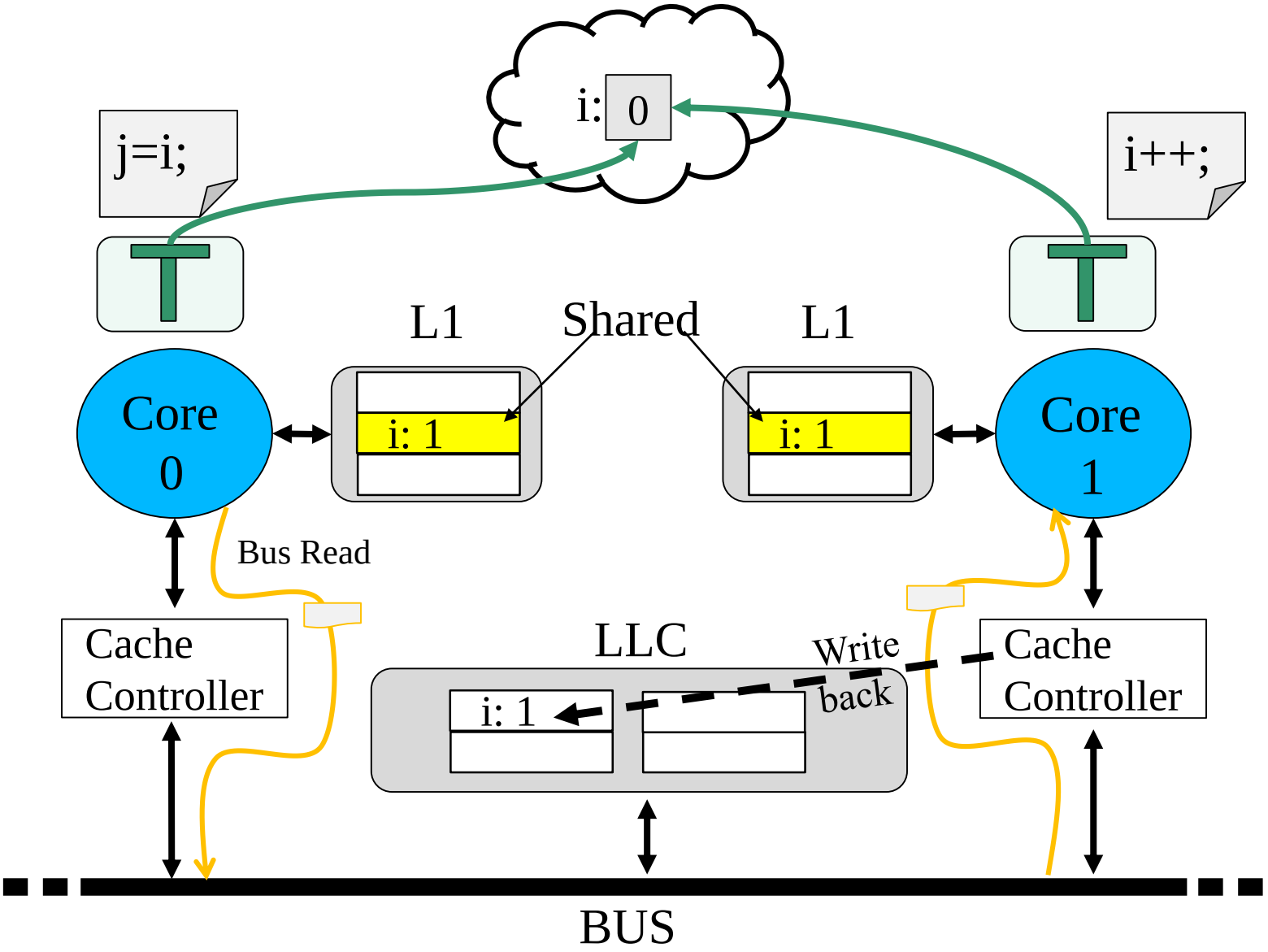
Cache coherence



Cache coherence



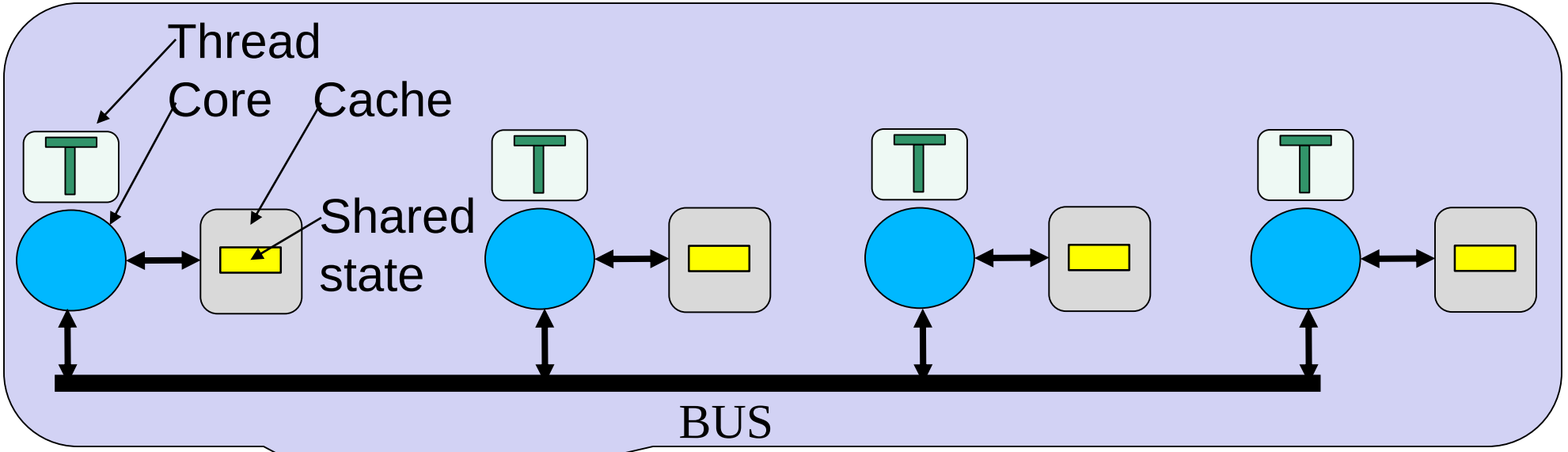
Cache coherence

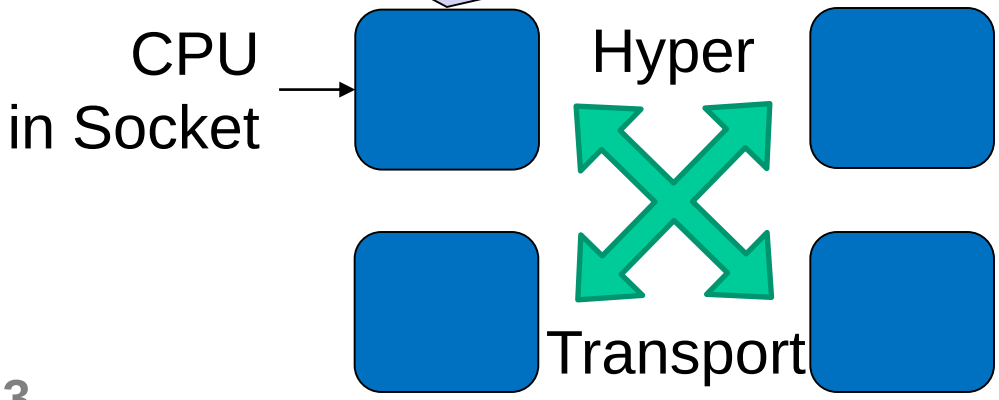


A key question

When updating shared state, which uproach is more expensive (in terms of latency), Shared memory or Message passing?

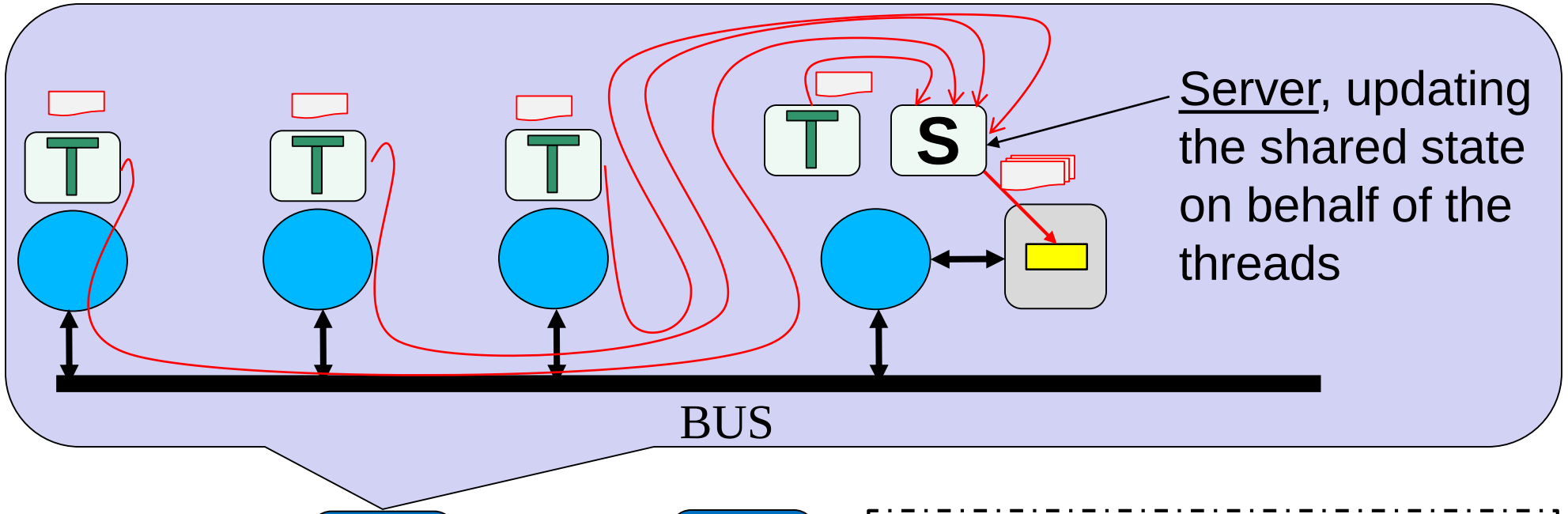
An experiment of shared memory vs message passing performance

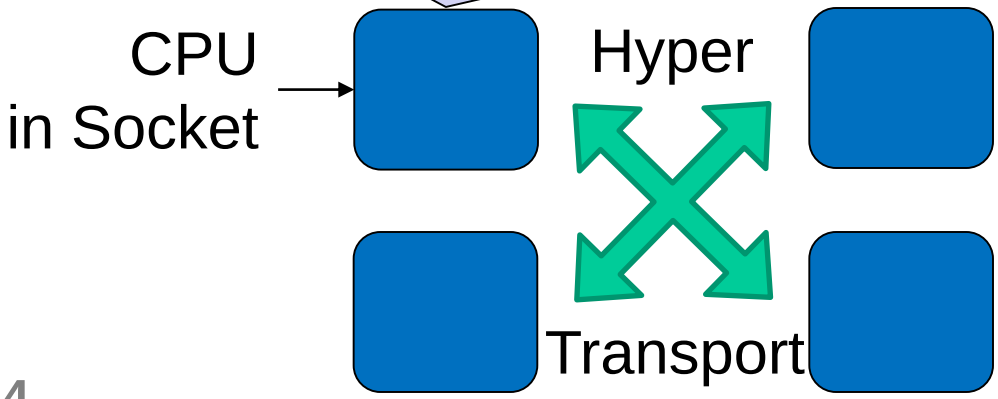




Updating shared state of size [1,8] cachelines, relying on cache coherent shared memory on 4x4 AMD system

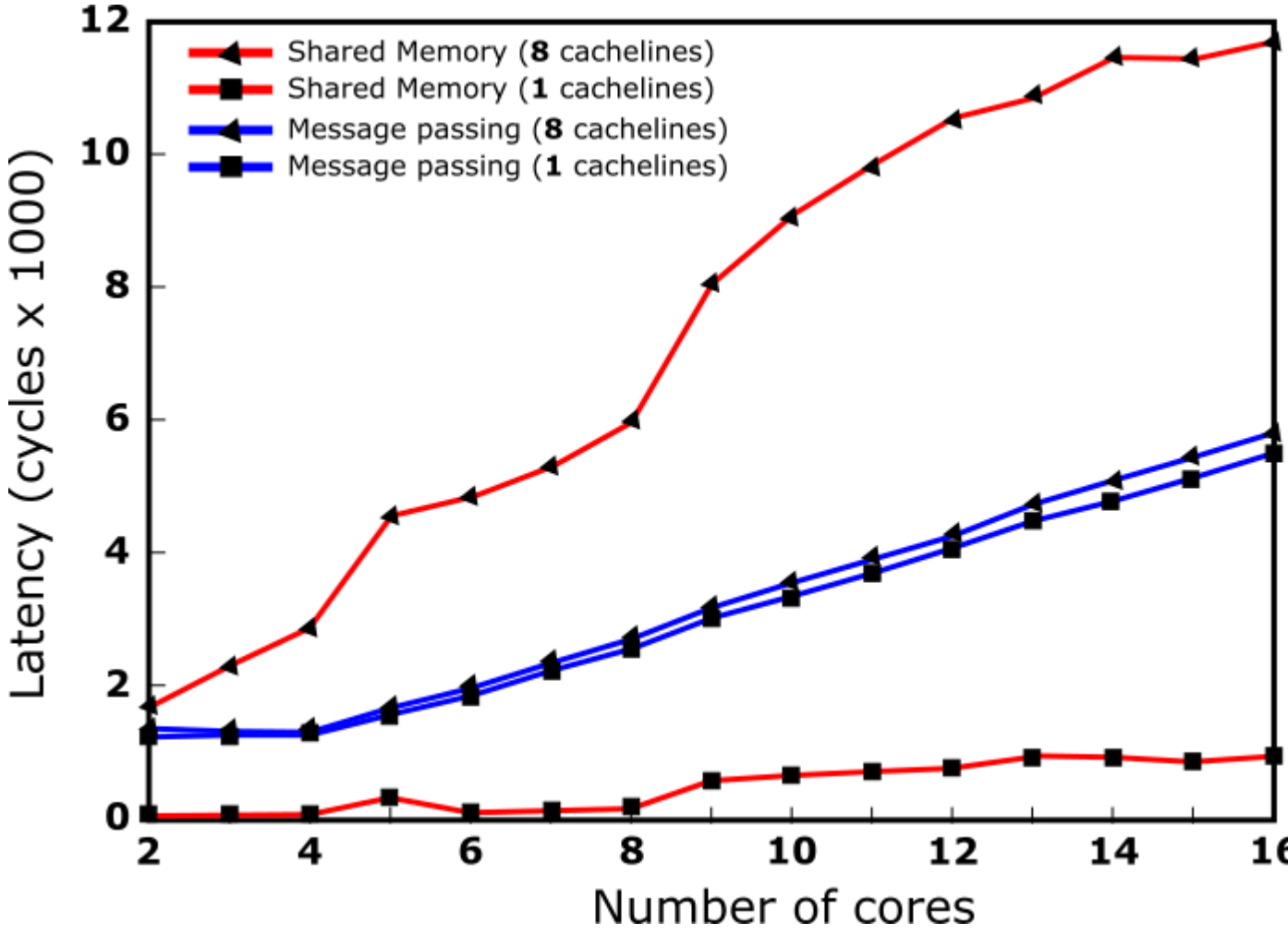
An experiment of shared memory vs message passing performance





Updating shared state of size [1,8] cachelines, relying on synchronous Lightweight Remote Procedure Calls (message passing)

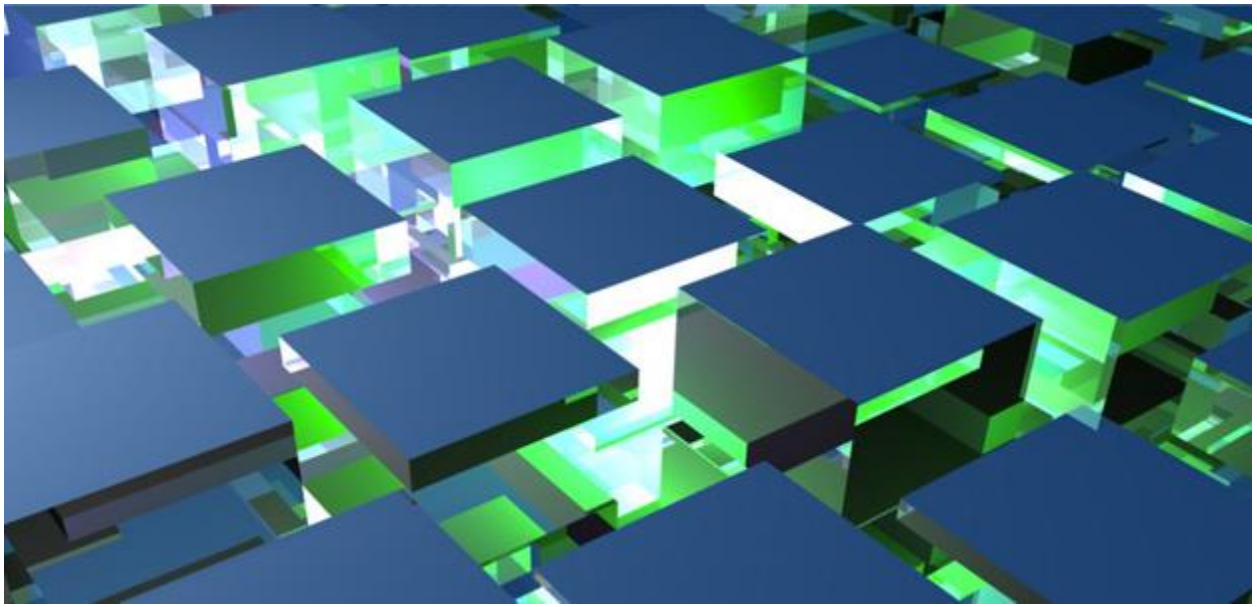
Messages scale better than shared memory



Message passing scales better than shared memory when increasing the core count and the size of the shared state.

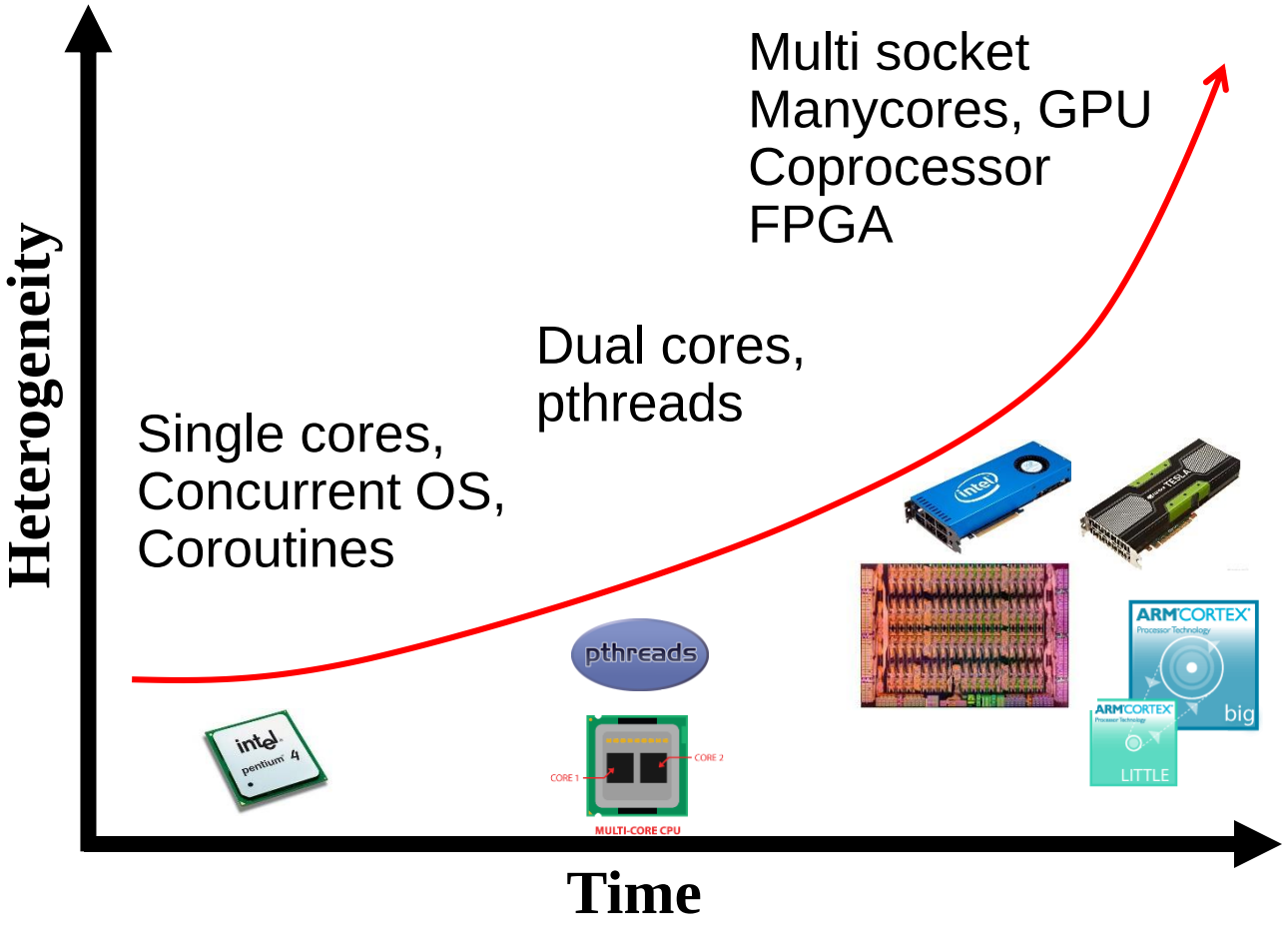
The plot is adapted from: Baumann, Andrew, et al. "The multikernel: a new OS architecture for scalable multicore systems." *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 2009.

…some other hints that may lead to further fragmentation of coherency domains



http://www.racktopsystems.com/wp-content/uploads/2013/01/sql-server-fragmentation.jpg

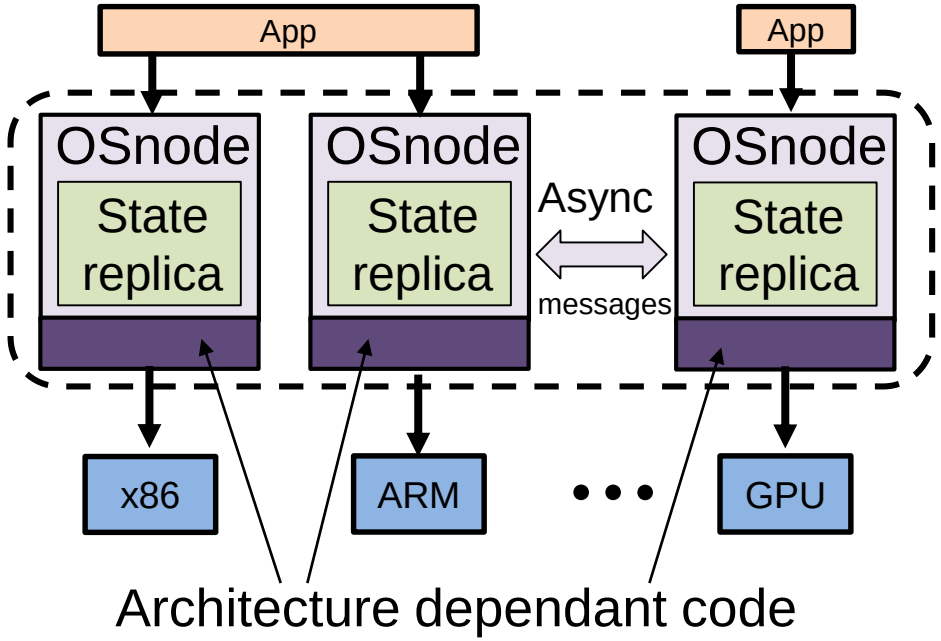
Increasing Heterogeneity of computing platforms



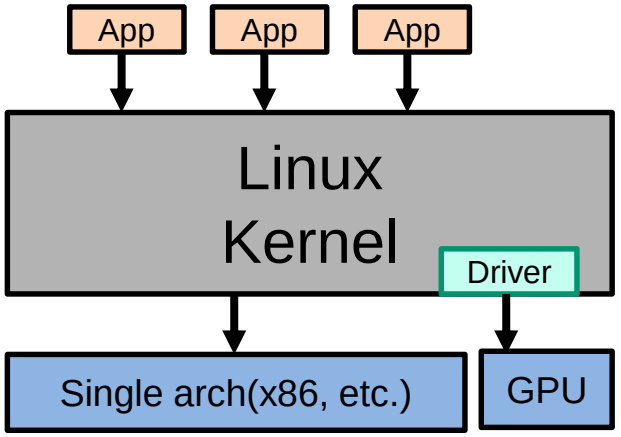
- **Message passing: Fundamental for communication in heterogeneous environment**
- **Shared memory: Hard to implement in a heterogeneous environment**

Message passing OS vs Shared memory OS

Barrelfish OS (Message passing)



Linux OS (Shared memory)



Adapted from : Baumann, Andrew, et al. "The multikernel: a new OS architecture for scalable multicore systems." *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 2009

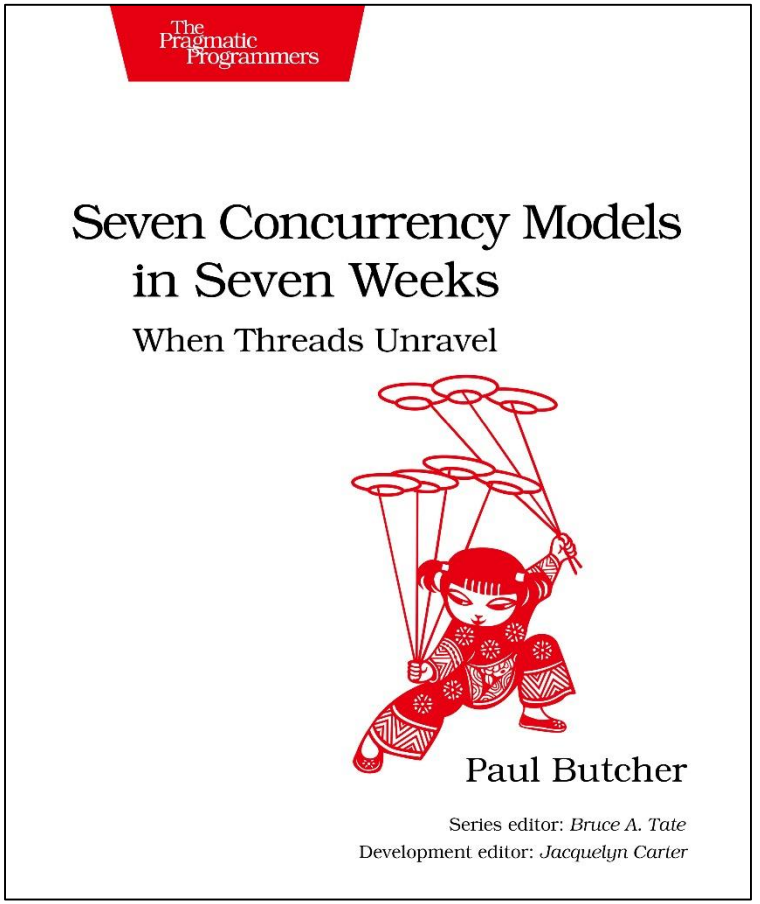
What to expect ?

http://tech.co/wp-content/uploads/2014/12/future-marketing.jpg

Emerging concurrency paradigms

New high level paradigms are being developed, based on shared memory and/or message passing constructs.

- **Asynchronous tasks (Futures/Promises)**
- **Partitioned Global Address Space (PGAS) languages/libraries**
- **Actor Model**
- **Functional Concurrency**



The future perspective

- **Communication is the key**
  - For **energy efficiency**
  - For runtime **performance**
  - To manage **software complexity**
  - To manage **hardware heterogeneity**
- **Innovation in the hardware sector pressures to systems software engineers to develop appropriate support**
  - At the operating system level
  - Concurrent programming frameworks level
  - Communication-oriented tools and techniques to design, implement, analyse concurrent programs

http://globe-views.com/dcim/dreams/surprise/surprise-05.jpg

References

- Baumann, Andrew, et al. "The multikernel: a new OS architecture for scalable multicore systems." *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 2009.
- Gerber, et al. "Not Your Parents' Physical Address Space", *Proceedings ot the 15th Workshop on Hot Topics in Operating Systems (HotOS 15)*
- A Primer on Memory Consistency and Cache Coherence Daniel J. Sorin, Mark D. Hill, and David A. Wood
- Martin, Milo MK, Mark D. Hill, and Daniel J. Sorin. "Why on-chip cache coherence is here to stay." *Communications of the ACM* 55.7 (2012): 78-89.
- Hansen, Per Brinch. "The invention of concurrent programming." *The origin of concurrent programming*. Springer New York, 2002. 3-61.
- Butcher, Paul. *Seven Concurrency Models in Seven Weeks: When Threads Unravel*. Pragmatic Bookshelf, 2014.
- Hoare, Charles Antony Richard. "Communicating sequential processes."*Communications of the ACM* 21.8 (1978): 666-677.

Thank you for your attention

Many thanks to my supporters and mentors for this presentation:

Sebastian Lopienski, *Sebastian.Lopienski@cern.ch*, CERN
Andreas Joachim Peters, *Andreas.Joachim.Peters@cern.ch*, CERN
Andreas Hirstius, *andreas.hirstius@intel.com*, Intel GmbH
Spyros Lalis, *lalis@inf.uth.gr*, University of Thessaly

This work is support by the Marie Curie Early European Industrial Doctorates Fellowship of the European Community's Seventh Framework Programme under contract number (PITN-GA-2012-316596-ICE-DIP).