Parallel Computing
An introduction

Marco Grossi
mgrossi@ectstar.eu

Ricerca ECT*, FBK Trento

Secondo corso di formazione
"Calcolo Parallelo su Grid (CSN4cluster)"
September 26th-28th, 2011
Parma

Parallel computing – What?

- Ingredients
  - **1** well formed problem
  - **n** processing elements(PEs)
  - **k** human of good will
- Recipe
  - The problem it's splitted into indepentent subproblems, each ones assigned to a PE
  - All the PEs run in parallel to solve the main problem
  - The PEs can interact each others in orther to exchange some data
- Constraints
  - The result of the problem computed in parallel must be comparable with the serial ones

Some definitions

$$speedup = \frac{T_1^{exe}}{T_p^{exe}} \leqslant \frac{f_s + f_p}{f_s + \frac{f_p}{p} + overhead} = \frac{1}{f_s + \frac{(1 - f_s)}{p} + overhead}$$

$$efficiency = \frac{speedup}{p} = \frac{T_1^{exe}}{p * T_p^{exe}}$$

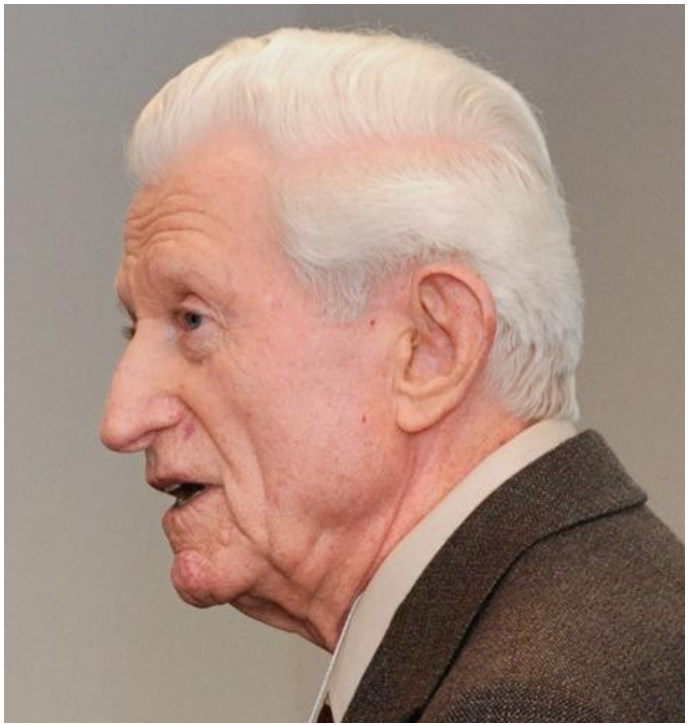
Where:

- f_s Fraction of serial code
- f_p Fraction of parallel code
- T_k^{exe} Execution time with k processors
- p Number of processors

Parallel computing – Why?

- Amdahl's fans(law)
  - With more PEs:
    - keep fixed the global problem size
    - decrease the subproblem size
- Strong scaling approach
- Speedup

$$s \leqslant \frac{1}{f_s + \frac{(1-f_s)}{p}}$$



Attribution: Pkivolowitz at en.wikipedia

Gene Amdahl

Parallel computing – Why?

Attribution: Wikipedia

John Gustafson

- Gustafson's fans(law)
  - With more PEs:
    - keep fixed the subproblem size
    - increase the global problem size
- Weak scaling approach
- Scaled speedup

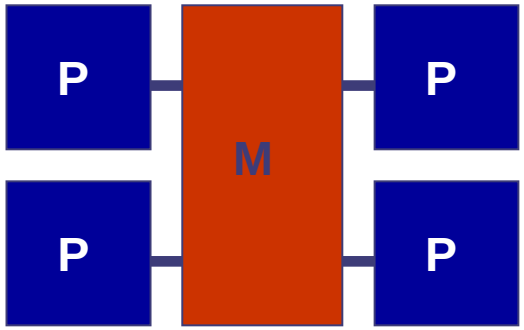
$$s \leqslant p + (1 - p) * f_s$$

Some consideration on theorical speedup

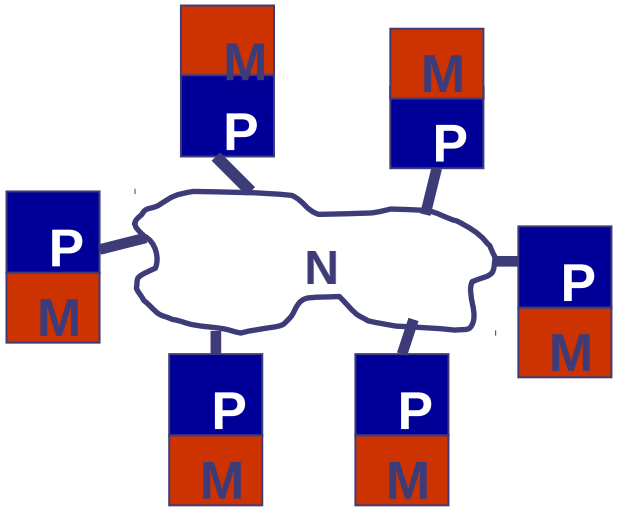
- Both laws ignores the parallelization overhead
  - The speedup may be overstimated
  - Overhead may be considered of a lower order respect to the parallel computation time when the problem size increase
  - Of course, other metrics exists but they are not so trivial
- In the strong scaling approach the speedup may also be understimated
  - This concept it's called ”cache effect” or ”superlinear speedup”:
    - lowering the subproblem size assigned to each PEs can:
      - improve the cache hit ratio
      - reduce wasted computation cycle
- Increasing the number of PEs may also increase the costs of collective operations(syncronization, reduce, ...), so more overhead

Parallel computing – How?

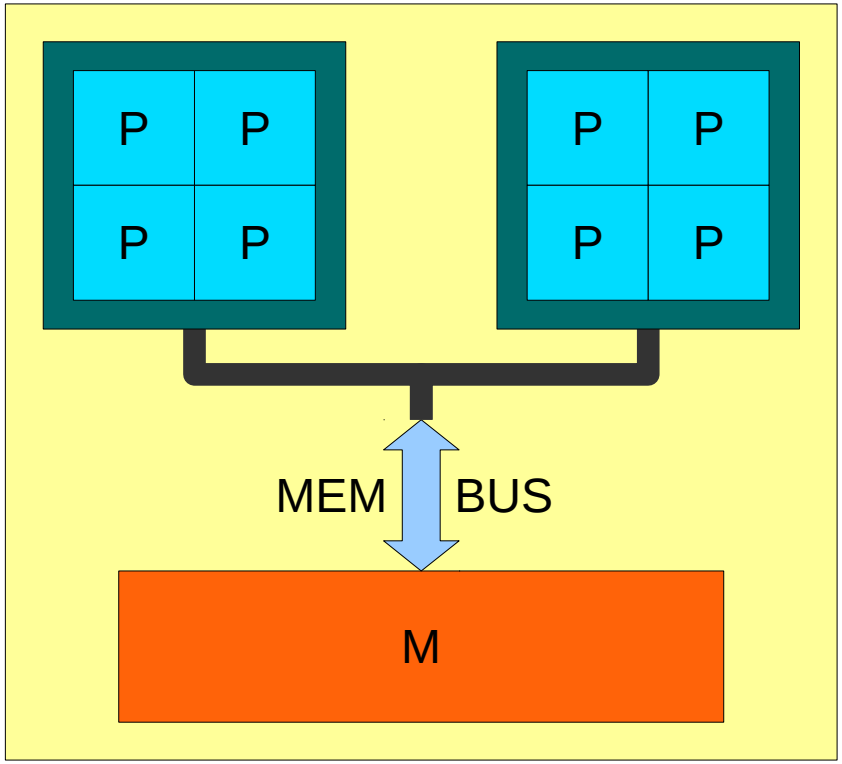
- Shared memory
  - All PEs share the same memory space
  - A PE exchange data using shared data structure



- Distributed memory
  - Each PEs has it's own memory space
  - A PE exchange data sending and receiving messages(i.e. message passing paradigm) over an Interconnection Network

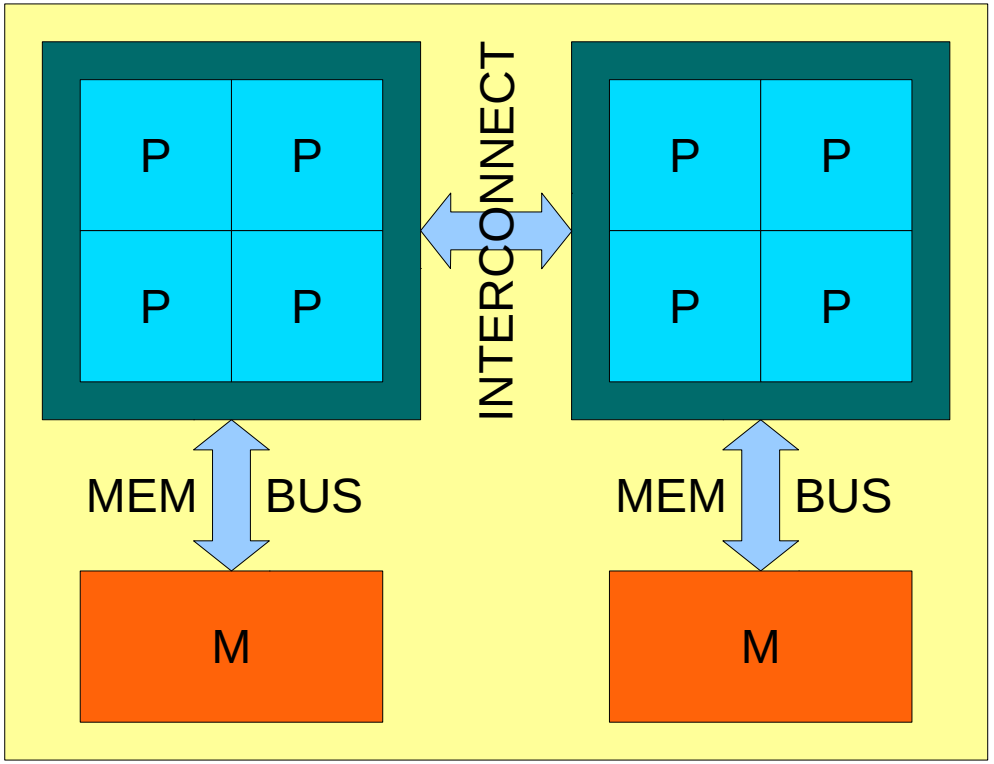


Shared memory architectures



UMA
Uniform Memory Access

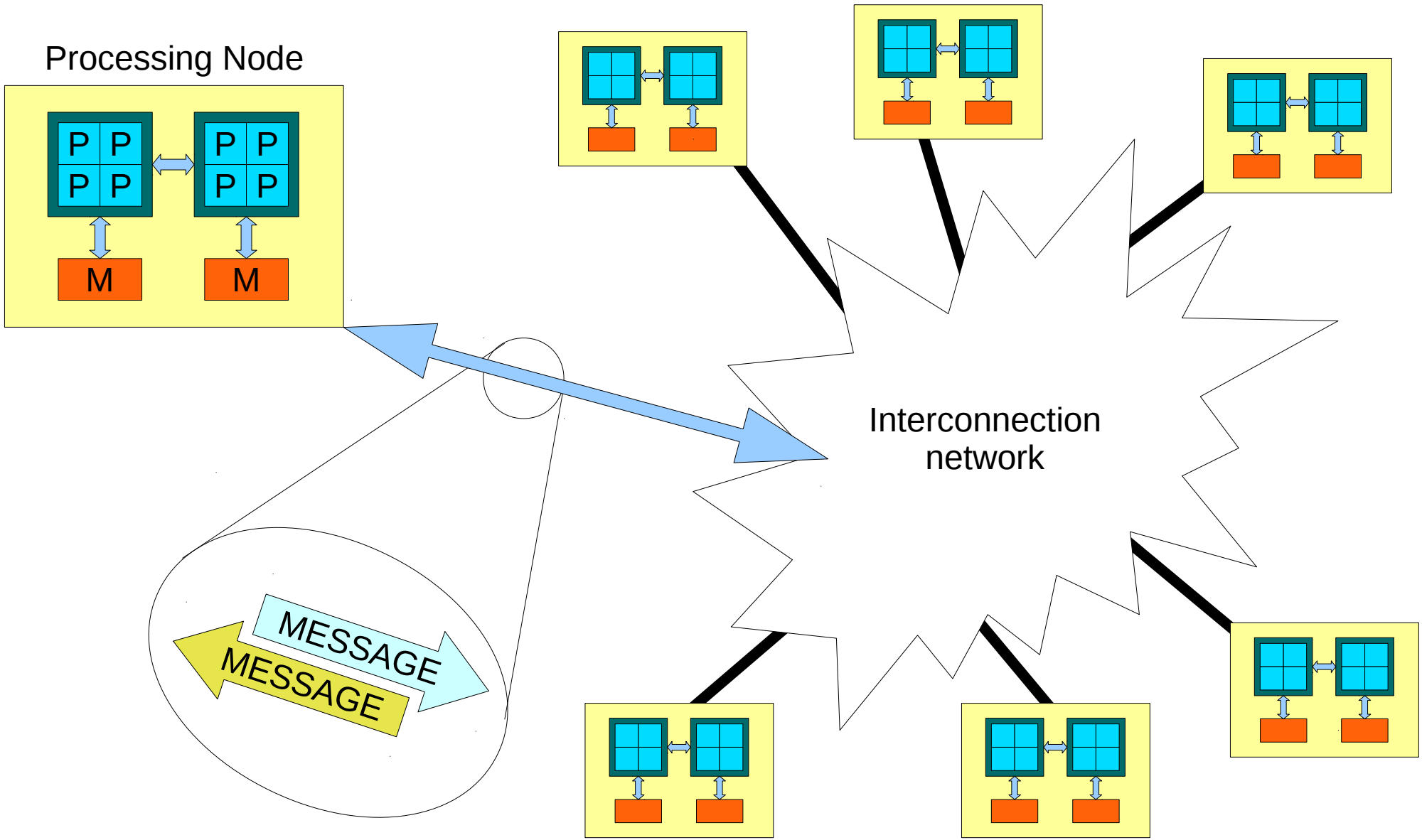
Any memory access
costs the same
for each PEs



NUMA
Non-Uniform Memory Access

Higher latency
(maybe also lower throughput)
accessing the remote memory

Distributed memory architectures

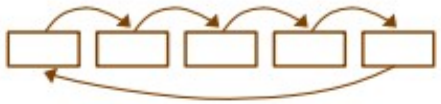


Distributed memory architectures

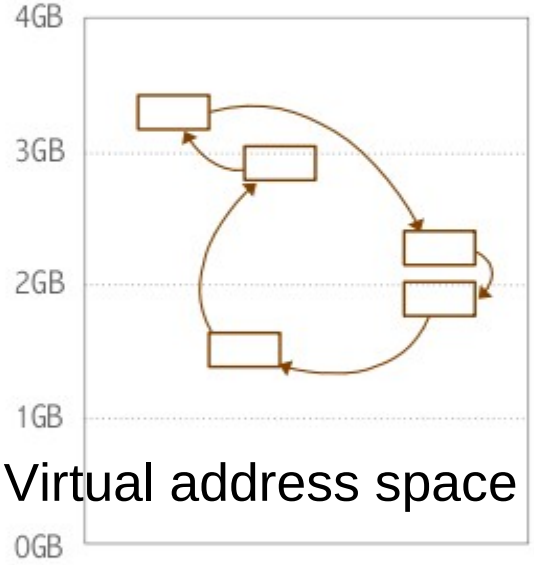
- Each Processing Nodes has at least one PE
  - Internally each node has an UMA or NUMA memory architectures
  - Each PE has it's own private memory
    - Use of message passing paradigm to exchange info between PEs
- An interconnection network enable the exchange of messages between different Processing Nodes
  - Different network topologies and routing algorithms → different interaction available between Processing Nodes: e.g.
    - All-to-all: full connectivity; a node can reach all other nodes
    - One-to-many: a node can reach only a subset of nodes
      - k-dimensional Mesh
      - k-dimensional Torus (Nearest Neighbours)
    - Only collective call
  - RDMA(Remote Direct Memory Access) maybe available

The NUMA age

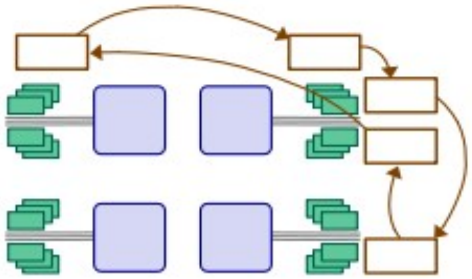
- If your Processing Nodes it's at least dual socket, nowadays has a NUMA memory architectures
- Better throughput and lower latency for local CPU memory access; high penalty for memory access of other CPU
- Some hitchs:
  - The Operating System may use load balancing scheduler, and by defaults your processes or threads can be migrated from one CPU to an other; your allocated memory may not be moved...
  - Shared memory not replicable on all CPU (at most: scattering)
  - (Keep cache coherency between different CPU cache may costs more than the UMA ones)



Logical linked list



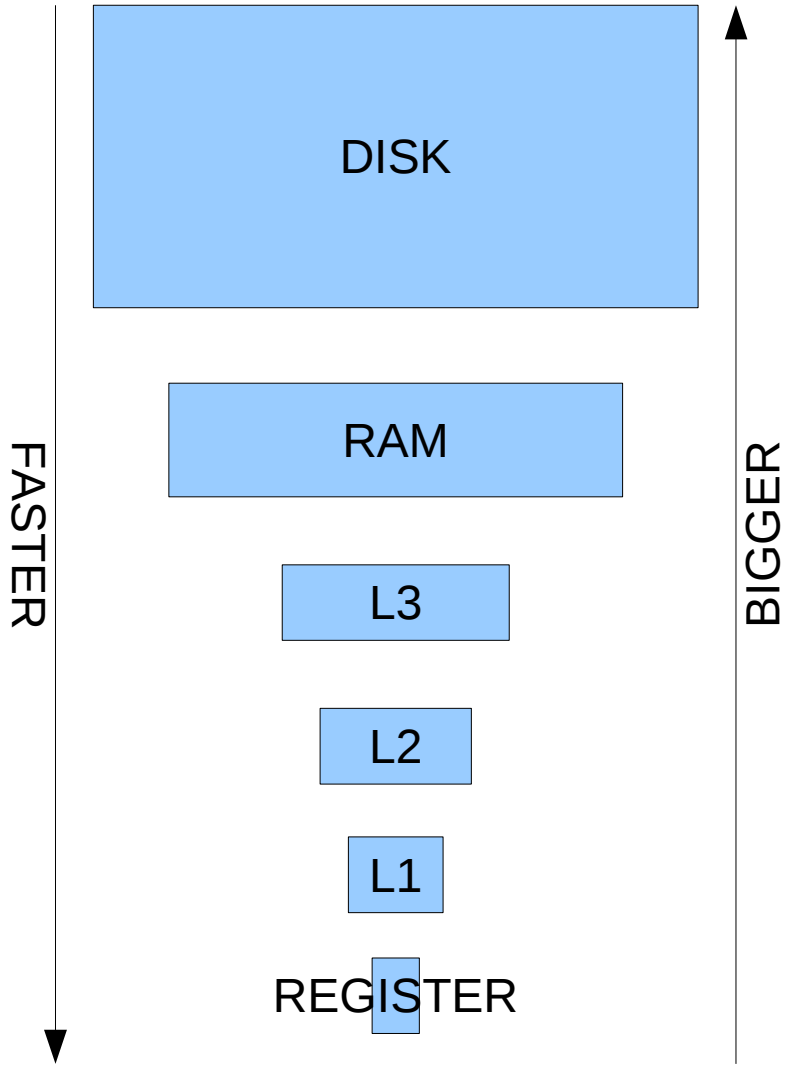
Virtual address space



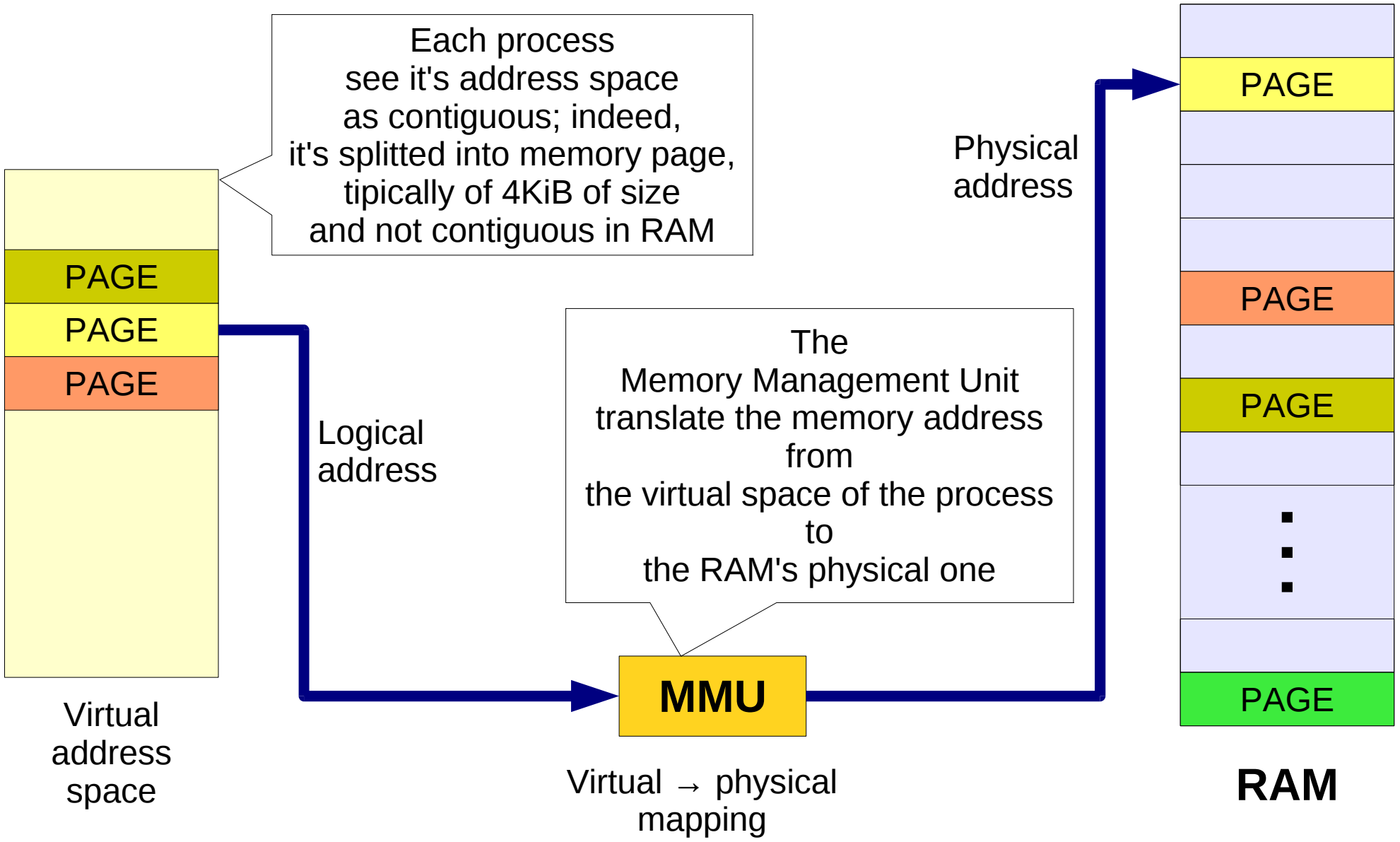
Physical memory

Hierarchical memory, memory wall

- Memory wall: disparity between processor and memory speed
- Keep your data as close as possible to your execution unit
- Adding an additional layer of caching(e.g. L3) may only add latency if you have an out-of-cache access pattern



Memory architecture, paging, virtual address space



Spatial/temporal locality

- Spatial locality
  - If a memory location is accessed than nearby locations in memory are likely to be accessed in the near future
    - Each time you reference a memory area, if not already present in cache, a cache line(e.g.: 64Byte) will be loaded from memory into cache; so accessing the memory with a linear pattern may cost less than a random one
- Temporal locality
  - If a memory location is accessed then it is likely to be accessed again in the near future
    - If you continue to reference small data sets, it's likely that you will find that in the fastest level of the memory hierarchy

Your questions & hints

Thank you for your attention!
For any questions and hints
please send an email to

mgrossi@ectstar.eu