Programming Techniques for Supercomputers:

Parallel Computers

Introduction
Shared-memory computers
Distributed-memory computers / Hybrid systems
Networks - Introduction

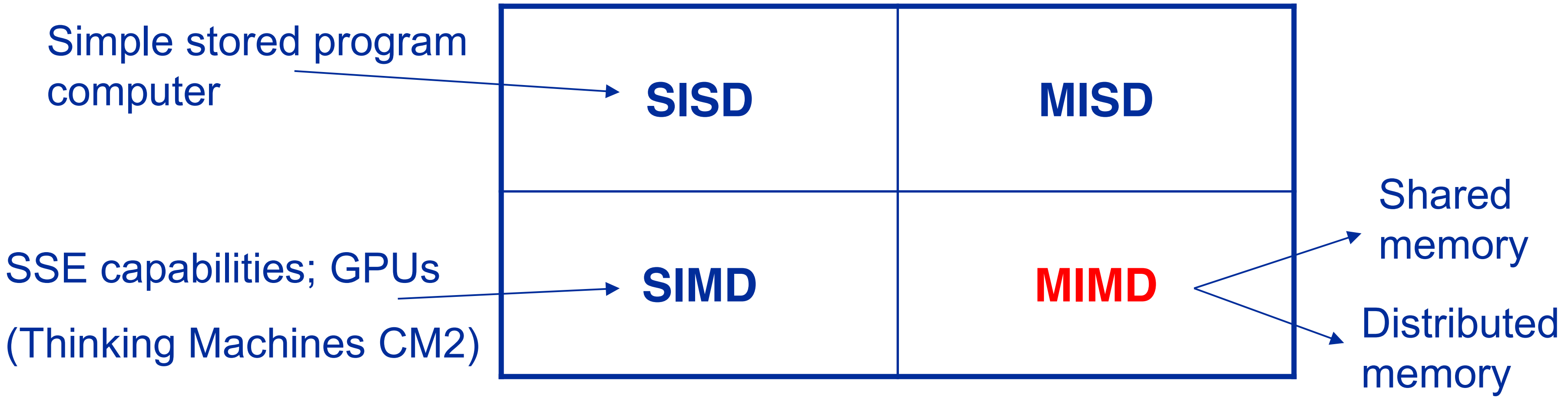
Prof. Dr. G. Wellein[a,b] , Dr. G. Hager[a] , M. Kreutzer[a]

[a]HPC Services – Regionales Rechenzentrum Erlangen
[b]Department für Informatik

University Erlangen-Nürnberg, Sommersemester 2013

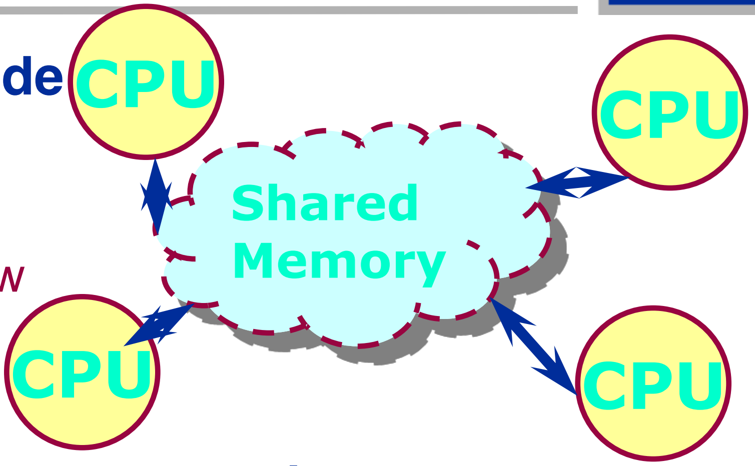
Parallel computers – Introduction

- „Parallel Computing“ : A number of compute elements solve a problem in a cooperative way
- Parallel Computer: A number of compute elements connected such way to do parallel computing for a large set of applications
- Classification according to Flynn: Multiple Instruction Multiple Data (MIMD)



Parallel computers – Shared-Memory Architectures

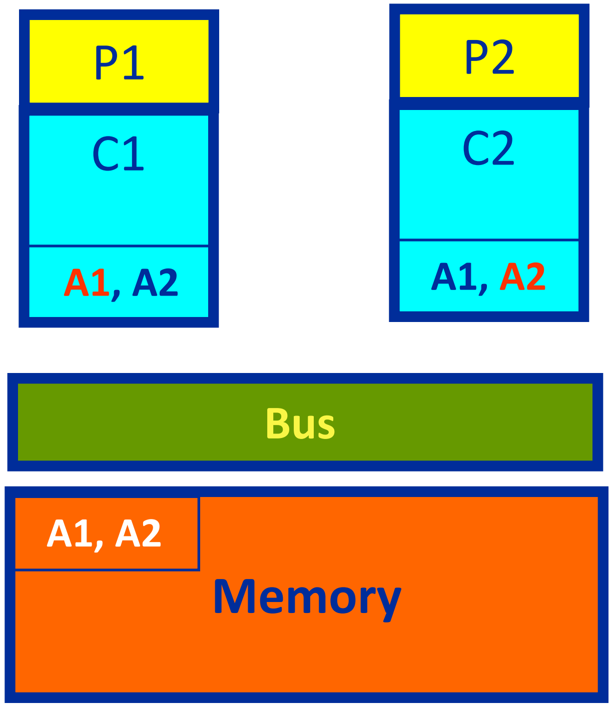
- **Shared memory computers provide**
  - a single shared address space (memory) for all processors
  - All processors share the same view of the address space!



- **Two basic categories of shared memory systems**
  - Uniform Memory Access (UMA):
    Memory is equally accessible to all processors with the same performance (Bandwidth & Latency)
  - cache-coherent Non Uniform Memory Access (ccNUMA):
    Memory is physically distributed but appears as a single address space: Performance (Bandwidth & Latency) is different for local and remote memory access
  - Copies of the same cache line may reside in different caches → Cache coherence protocols guarantees consistency all time (for UMA & ccNUMA)
  - Cache coherence protocols do not alleviate parallel programming for shared-memory architectures!

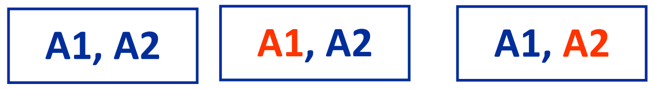
Parallel computers – Cache coherence

- **Data in cache is only a copy of data in memory**
  - Multiple copies of same data on multiprocessor systems
  - Cache coherence protocol/hardware ensure consistent data view
  - Without cache coherence, shared cache lines can become clobbered: (Cache line size = 2 WORD; A1+A2 are in a single CL)

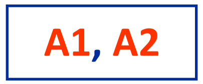


P1	**P2**
`Load A1`	`Load A2`
`Write A1=0`	`Write A2=0`

Write-back to memory leads to incoherent data

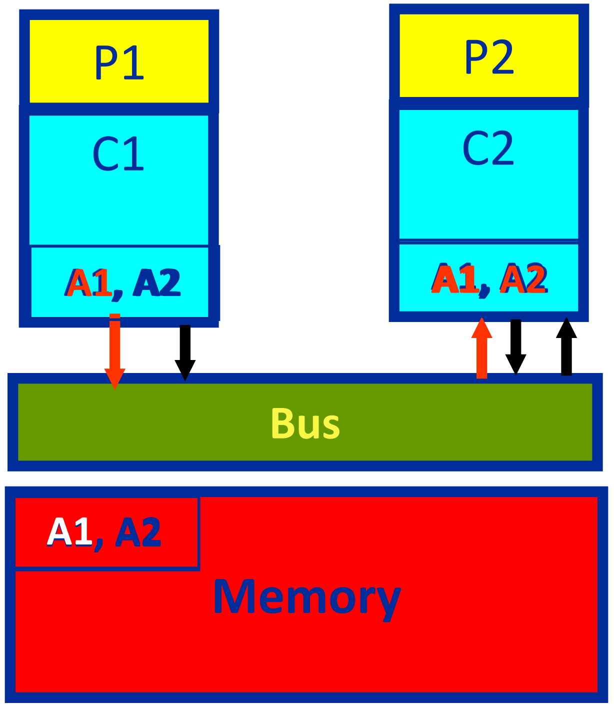


C1 & C2 entry can not be merged to:



Parallel computers – Cache coherence

- Cache coherence protocol must keep track of cache line status



P1	P2
`Load A1`	`Load A2`
`Write A1=0:`	
1. Request exclusive access to CL	
2. Invalidate CL in C2	
3. Modify A1 in C1	
	`Write A2=0:`
	1. Request exclusive CL access
	2. CL write back+ Invalidate
	3. Load CL to C2
	4. Modify A2 in C2

t

C2 is exclusive owner of CL ←

Parallel computers – Cache coherence

- **Cache coherence can cause substantial overhead**
  - may reduce available bandwidth
- **Different implementations**
  - Snoop: On modifying a CL, a CPU must broadcast its address to the whole system
  - Directory, "snoop filter": Chipset ("network") keeps track of which CLs are where and filters coherence traffic
- **Directory-based ccNUMA can reduce pain of additional coherence traffic**
- **But always take care:**

Multiple processors should never write frequently to the same cache line ("false sharing")!

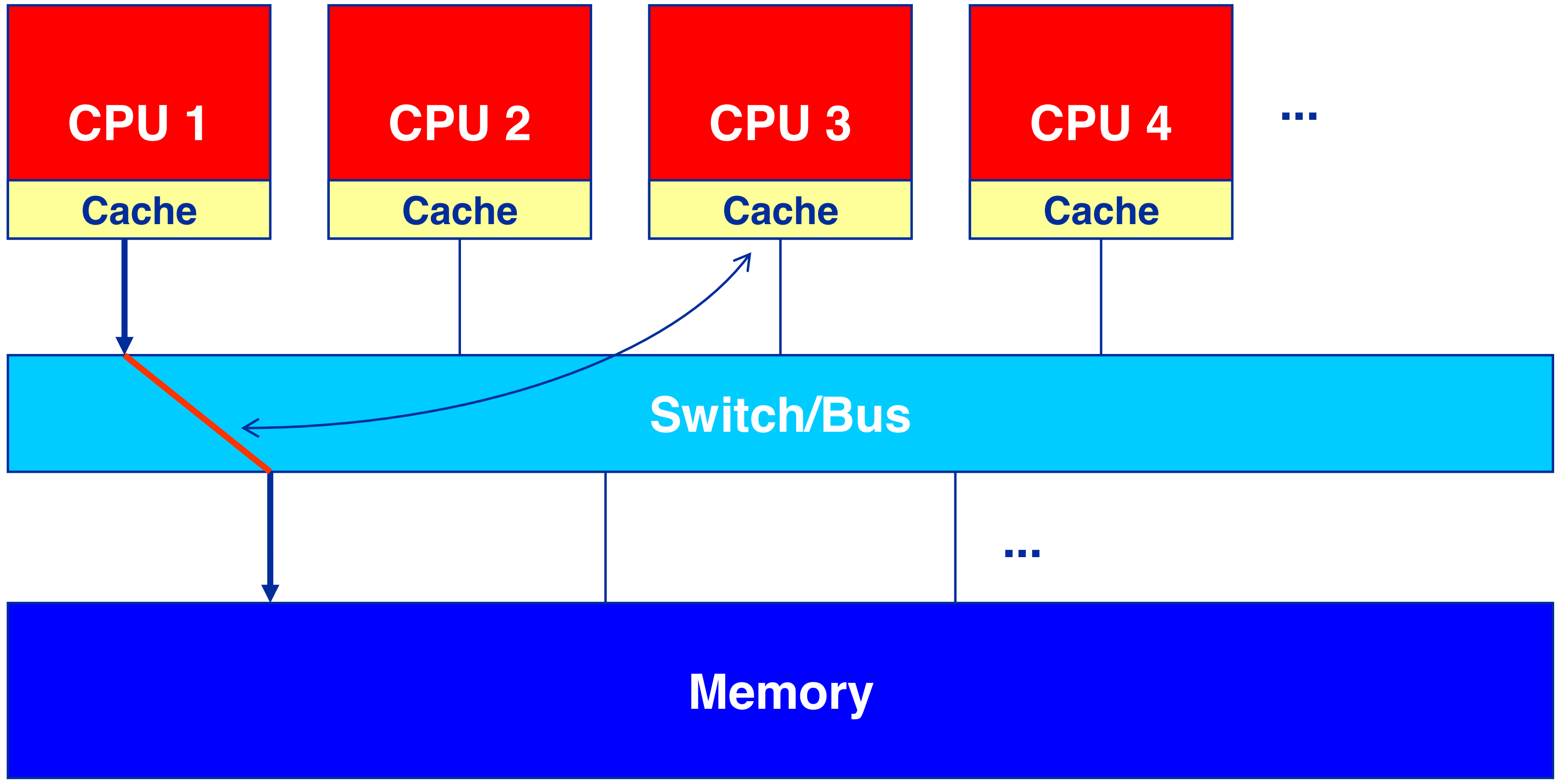
Parallel computers – Cache coherence

- **Widespread cache coherence protocol: MESI protocol**

- **A cache line can have four different states:**
  - Modified: Cache line has been modified in this cache, and it resides in no other cache. Cache line needs to be evicted to ensure memory consistency
  - Exclusive: Cache line has been read from main memory but not (yet) modified. There are no (valid) copies in other caches
  - Shared: Cache line has been read from memory but not modified. There may be valid copies in other caches
  - Invalid: This cache line does not reflect any sensible data. Usually this happens if the cache line was in **S** state and another processor request exclusive ownership

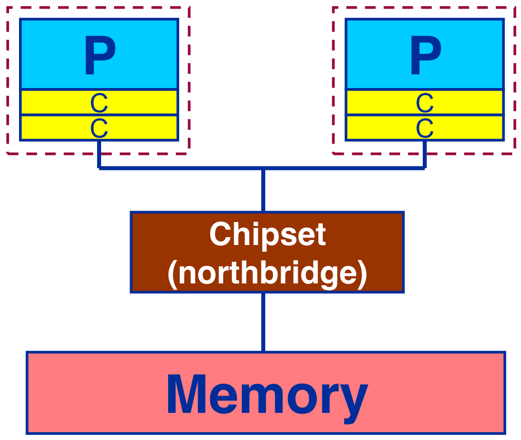
Parallel computers: Shared-memory: UMA

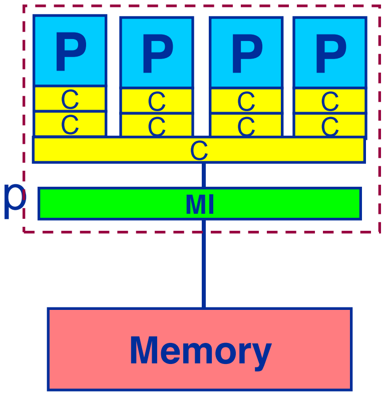
- **UMA Architecture: switch/bus arbitrates memory access**
  - Special protocol ensures cross-CPU cache data consistency
  - Flat memory – also known as „Symmetric Multi-Processor“ (SMP)



Parallel computers: Shared-memory: UMA / Bus based

- **Worst case: bus system provides single bandwidth to multiple processors**
  - Only one CPU at a time can use the bus and access memory at any one time – No need to provide for faster memory
  - Collisions occur frequently, causing one or more CPUs to wait for “bus ready” (contention)
  - Even worse: Shared memory bus in current multi-core chips further reduce available bandwidth/peak FLOP balance!
  - Price/performance ratio was good for certain applications for a long time
  - Similar concept in a single multi-core chip

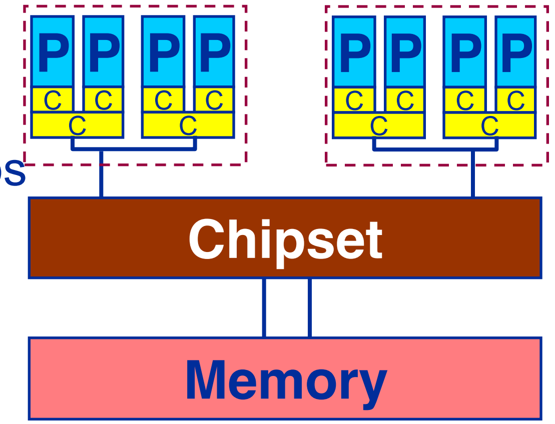




Parallel computers: Shared Memory: UMA / Crossbar based

- **Best case: memory crossbar switch provides separate data path to memory for each CPU**
  - Can saturate full memory bandwidth of every CPU concurrently
  - Bus contention occurs only if same memory module is accessed by more than one CPU
  - Memory interleaving is a must
  - Example:
    - NEC SX-9 node: 16 CPUs, 256 GB/sec each, 4096 GB/sec per node bandwidth

- **Reality: Compromises are made**
  - E.g. 2 of 8 cores can use full bandwidth concurrently
  - May be advantageous not to use all cores on a node
  - Example: Intel Clovertown based on 2 quad-core chips
    2 sockets (8 cores), 10.6 GB/sec each,
    21.2 GB/sec node bandwidth; i.e. 2.66 GB/s per core

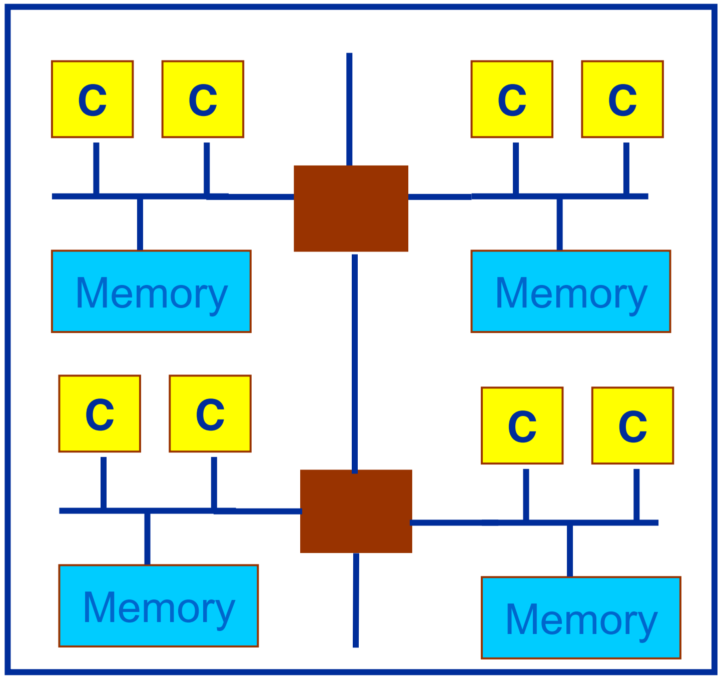


Parallel computers: Shared Memory: UMA Nodes

- **Examples:**
  - Your dual-/quad-/hexa-core laptop/desktop computer
  - IBM BlueGene series
  - NEC vector systems
- **Advantages**
  - Cache Coherence (see below) is "easy" to implement
  - Easy to optimize memory access
  - Incremental parallelization
  - Large memory configuration in a single address space
- **Disadvantages**
  - Memory bandwidth and price (!) often limit scalability
    (2 – 8 cores per UMA node)

Parallel shared memory computers: ccNUMA/Node Layout

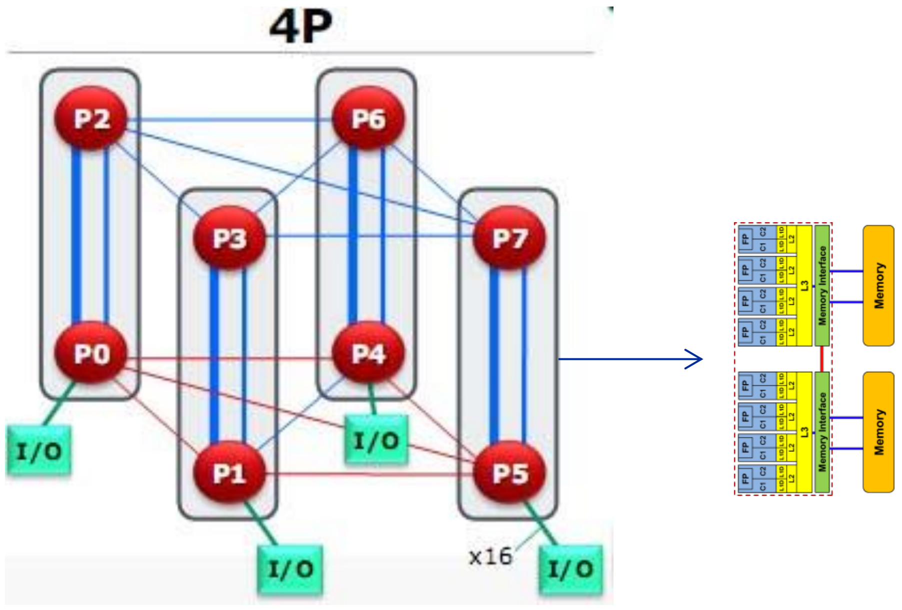
- **ccNUMA:**
  - Single address space although physically distributed memory through proprietary hardware concepts (e.g. NUMALink in SGI systems; QPI for Intel; HT for AMD)



- **Advantages:**
  - Aggregate memory bandwidth is scalable
  - Systems with more 1024 cores are available (SGI)
- **Disadvantages:**
  - Cache Coherence hard to implement / expensive
  - **Performance depends on access to local or remote memory**
- **Examples: All modern multi-socket compute nodes – SGI Altix/UV**

Parallel shared memory computers: larger ccNUMA nodes

- **System Architecture cont'd**
  - Layout for **4-socket AMD Opteron** system → 4 x (2 x 8 cores) = 64 cores with **8 NUMA domains**
  - … and you know this can get really large!



- **Who keeps track of the contents of all those caches?**
  - **Cache coherence mechanisms**

- **How do we make sure that memory pages are as close as possible to the CPUs they are needed on?**
  - **ccNUMA optimization**