HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Energy-Aware Computing in Heterogeneous Data Centers

Terminology

Max Plauth, *Sven Köhler*, Felix Eberhardt, Lukas Wenzel and Andreas Polze
Operating Systems and Middleware Group

Terminology

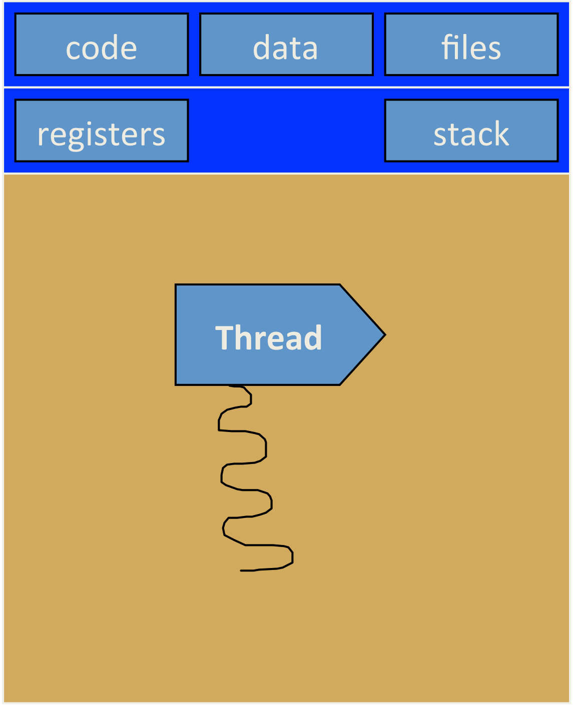
Energy-Aware Computing Seminar

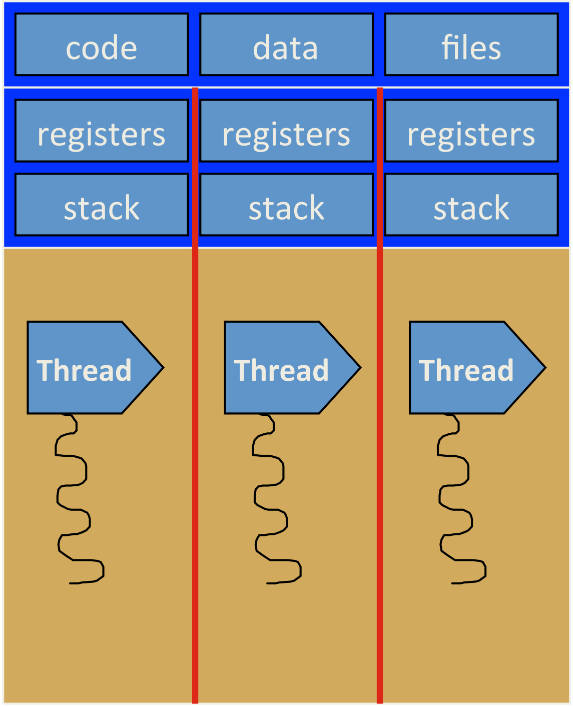
Sven Köhler

Terminology

- **Concurrency**
  - Capability of a system to have two or more activities in progress at the same time
  - May be independent, loosely coupled or closely coupled
  - Classical operating system responsibility for a better utilization of CPU, memory, network, and other resources
  - Demands scheduling and synchronization
- **Parallelism**
  - Capability of a system to execute activities simultaneously
  - Demands parallel hardware, concurrency support, (and communication)
- Any parallel program is a concurrent program
- Some concurrent programs cannot be run as parallel program

Example: Operating System





Concurrency Is Hard

- **Deadlock**
  - Two or more processes / threads are unable to proceed
  - Each is waiting for one of the others to do something
- **Livelock**
  - Two or more processes / threads continuously change their states in response to changes in the other processes / threads
  - No global progress for the application
- **Race condition**
  - Two or more processes / threads are executed concurrently
  - Final result of the application depends on the relative timing of their execution

Race Condition

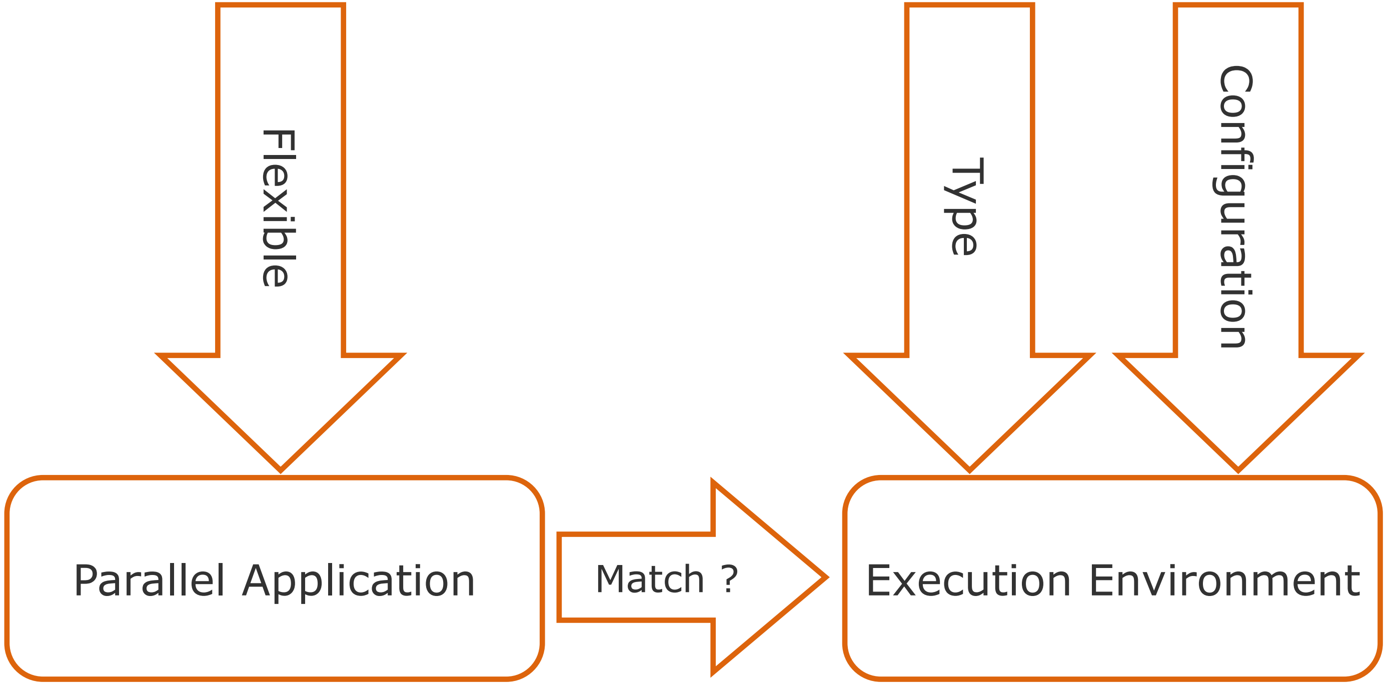
```
void echo() {
        char_in = getchar();
        char_out = char_in;
        putchar(char_out);
}
```

- One piece of code in one process, executed at the same time …
    - … by two threads on a single core.
    - … by two threads on two cores.
- What happens ?

Terminology

- **Starvation**
  - A runnable process / thread is overlooked indefinitely
  - Although it is able to proceed, it is never chosen to run (dispatching / scheduling)
- **Atomic Operation**
  - Function or action implemented as a sequence of one or more instructions
  - Appears to be indivisible - no other process / thread can see an intermediate state or interrupt the operation
  - Executed as a group, or not executed at all
- **Mutual Exclusion**
  - The requirement that when one process / thread is using a resource, no other shall be allowed to do that

The Parallel Programming Problem

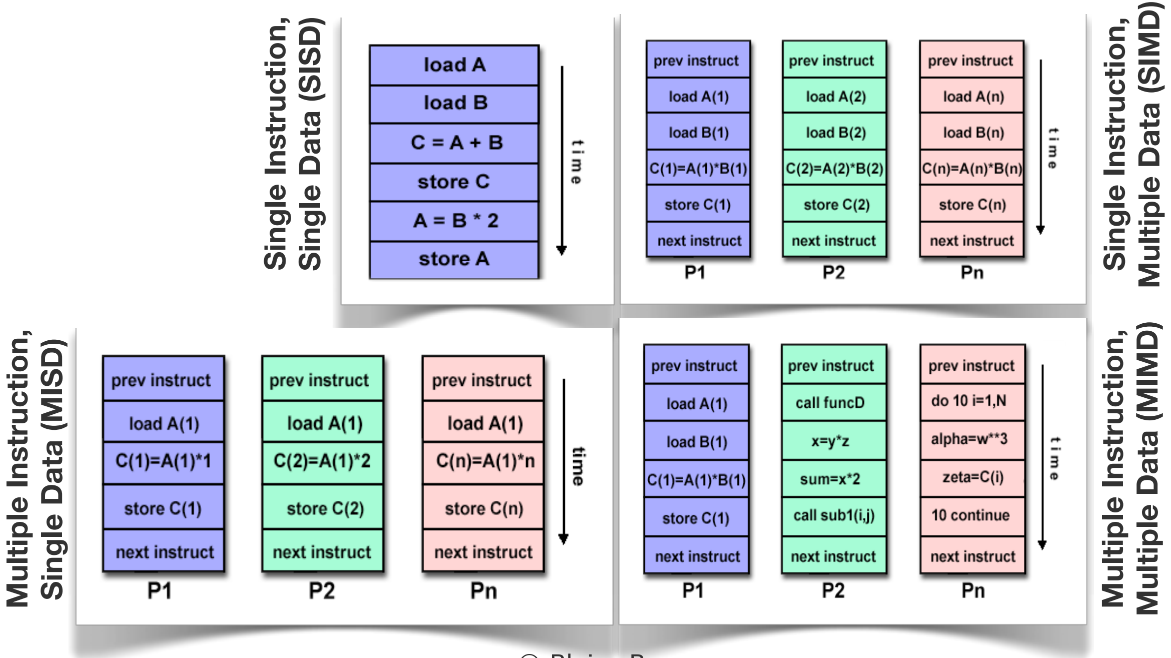


Energy-Aware Computing Seminar

Sven Köhler

Multiprocessor: Flynn‘s Taxonomy (1966)

- Classify multiprocessor architectures among **instruction** and **data** processing **dimension**



Single Instruction, Single Data (SISD)

load A
load B
C = A + B
store C
A = B * 2
store A
time

Single Instruction, Multiple Data (SIMD)

P1: prev instruct, load A(1), load B(1), C(1)=A(1)*B(1), store C(1), next instruct
P2: prev instruct, load A(2), load B(2), C(2)=A(2)*B(2), store C(2), next instruct
Pn: prev instruct, load A(n), load B(n), C(n)=A(n)*B(n), store C(n), next instruct
time

Multiple Instruction, Single Data (MISD)

P1: prev instruct, load A(1), C(1)=A(1)*1, store C(1), next instruct
P2: prev instruct, load A(1), C(2)=A(1)*2, store C(2), next instruct
Pn: prev instruct, load A(1), C(n)=A(1)*n, store C(n), next instruct
time

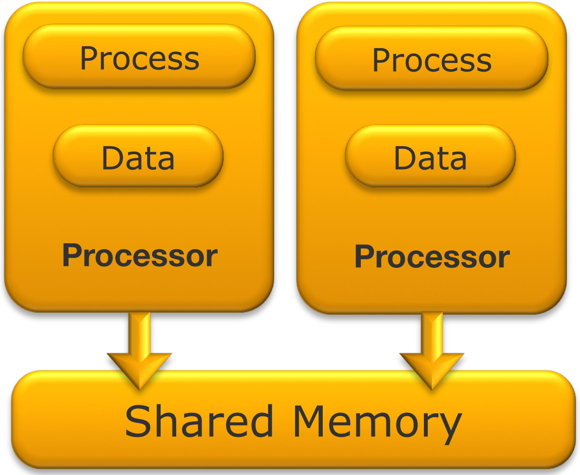
Multiple Instruction, Multiple Data (MIMD)

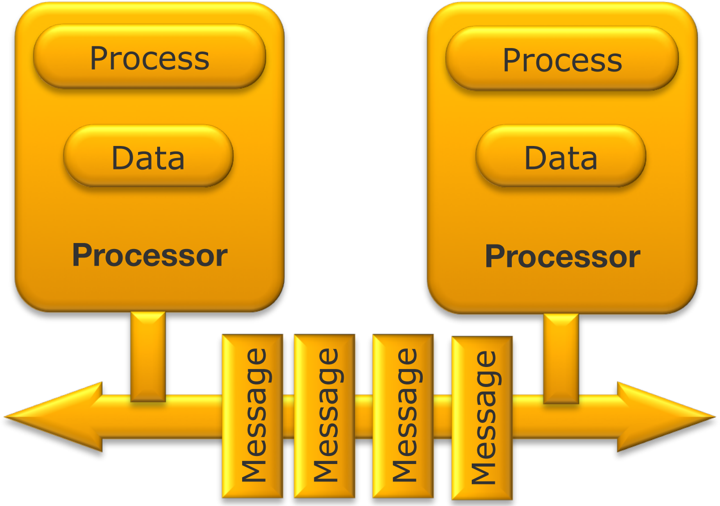
P1: prev instruct, load A(1), load B(1), C(1)=A(1)*B(1), store C(1), next instruct
P2: prev instruct, call funcD, x=y*z, sum=x*2, call sub1(i,j), next instruct
Pn: prev instruct, do 10 i=1,N, alpha=w**3, zeta=C(i), 10 continue, next instruct
time

© Blaise Bamey

Shared Memory vs. Shared Nothing

- Pfister: „shared memory“ vs. „distributed memory“
- Foster: „multiprocessor“ vs. „multicomputer“
- Tannenbaum: „shared memory“ vs. „private memory“

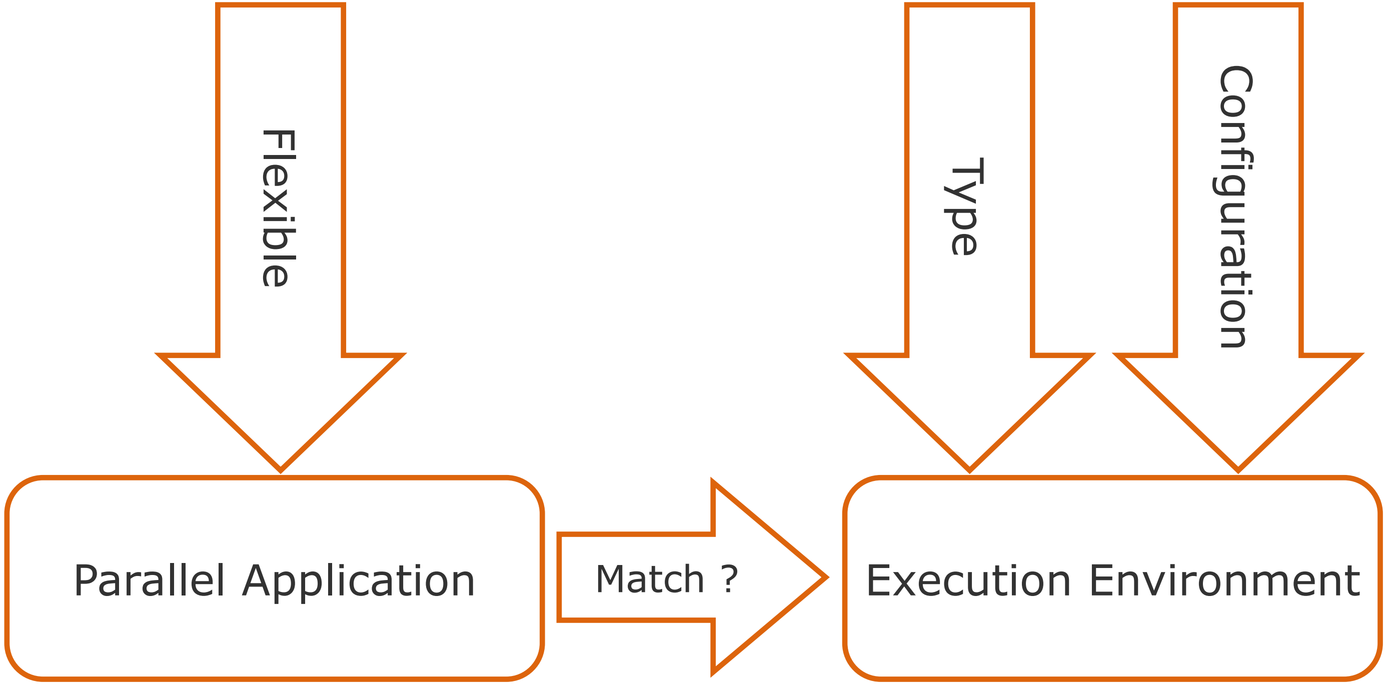




Shared Memory vs. Shared Nothing

- Organization of parallel processing hardware as …
  - **Shared memory system**
    - Concurrent processes can directly access a common address space
    - Typically implemented as memory hierarchy, with different cache levels
    - Examples: SMP systems, distributed shared memory systems, virtual runtime environment
  - **Shared nothing system**
    - Concurrent processes can only access local memory and exchange messages with other processes
    - Message exchange typically order of magnitudes slower than memory
    - Examples: Cluster systems, distributed systems (Hadoop, Grids, …)

The Parallel Programming Problem



Energy-Aware Computing Seminar

Sven Köhler

Programming Paradigm

- **Programming paradigm**
  - Coding convention or standard
  - Something a majority of people agrees upon
- **Parallel programming** is one of these paradigms
  - Other: Declarative, constraint-based, object-oriented
- Each paradigm can be realized by a set of **programming models**
- **Programming model**: *„set of rules for a game“* [Almasi]
  - Programs and algorithms as game strategies
  - Point where execution environment and application meet
  - High-level view of the application on it�something run time environment
  - Hardware might imply a model, but does not enforce it
  - For uniprocessor, no question due to „von Neumann‟
  - Delivering performance while raising the level of abstraction