Related Technologies

- **HPC covers a wide range of technologies:**
  - **Computer architecture**
  - **Networking**
  - **Compilers**
  - **Algorithms**
  - **Workload and resource manager**
    - **A big HPC system handles many parallel programs from different users**
    - **Task scheduling and resource allocation**
    - **metrics: system throughput, resource utilization, mean response time**

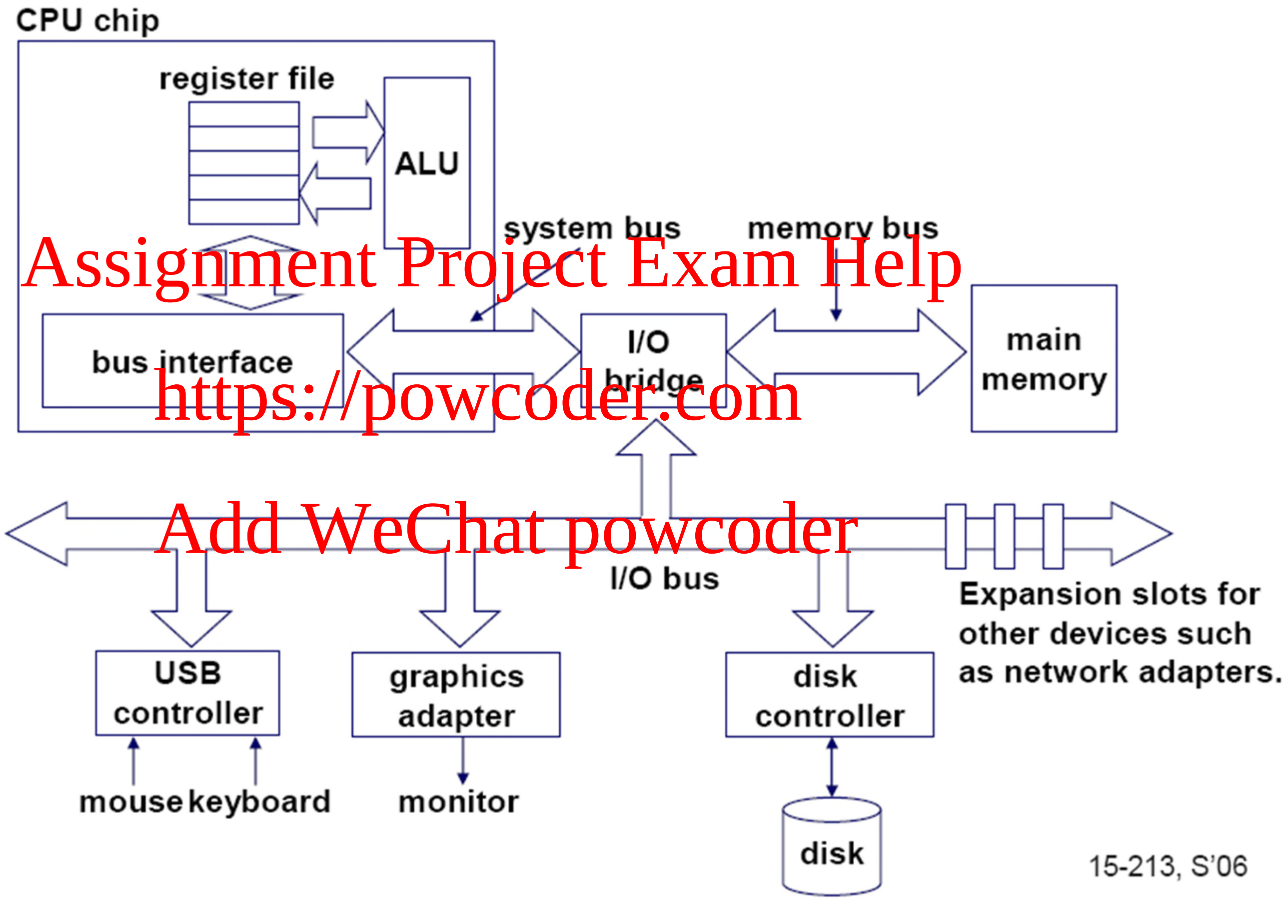
Related Technologies

- **HPC covers a wide range of technologies:**
  - **Computer architecture**
  - **Networking**
  - **Compilers**
  - **Algorithms**
  - **Workload and resource manager**

History and Evolution of HPC Systems

- **1960s: Scalar processor**
  - **Process one data item at a time**

Scalar processor



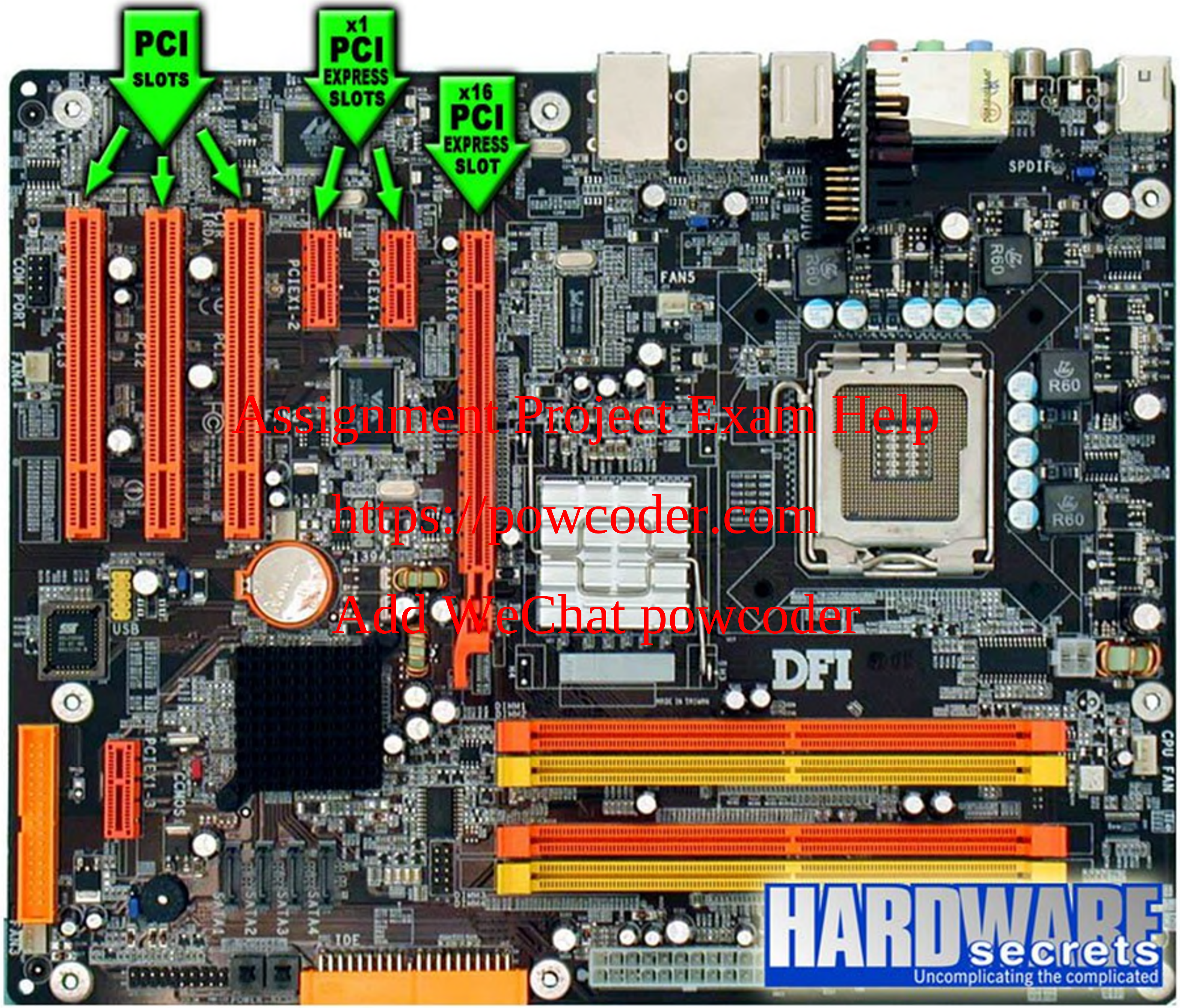
History and Evolution of HPC Systems

- 1960s: Scalar processor
- 1970s: Vector processor
  - Can process an array of data items in one go
  - Architecture: one master processor and many math co-processors (ALU)
  - Each time the master processor fetches an instruction and a vector of data items and feed them to ALUs
  - Overhead: more complicated address decoding and data fetching procedure
  - Difference between vector processor and scalar processor

GPU (Vector processor)

GPU: Graphical Processing Unit

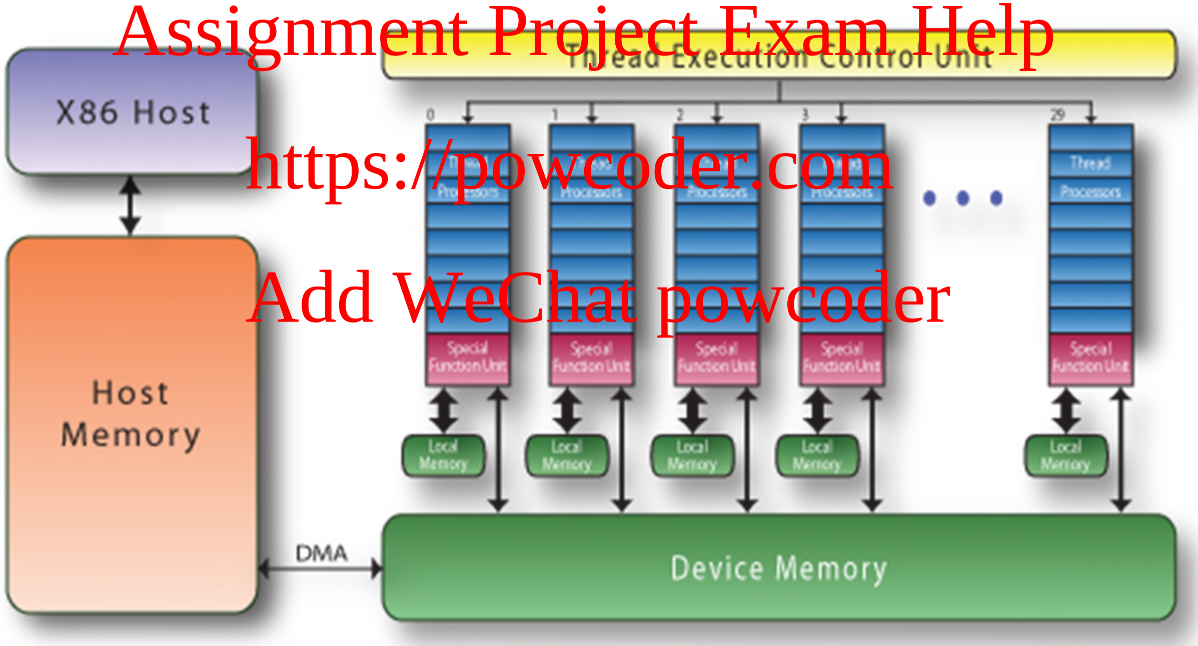
GPU is treated as a PCIe device by the main CPU


PCI
SLOTS
x1
PCI
EXPRESS
SLOTS
x16
PCI
EXPRESS
SLOT
DFI
HARDWARE
secrets
Uncomplicating the complicated

GPU (Vector processor)

GPU: Graphical Processing Unit

GPU is treated as a PCIe device by the main CPU

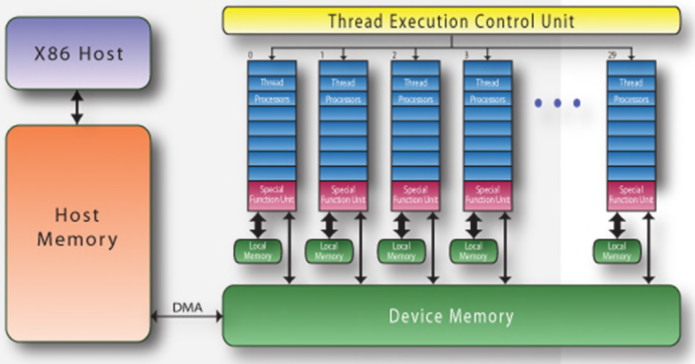


Data processing on GPU

- CUDA: programming on GPU
- Get the array A and B in one memory access operation
- Different threads process different data items
- If no much parallel processing, slower on GPU due to overhead


```
// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}

int main()
{
    ...
    // Kernel invocation with N threads
    VecAdd<<<1, N>>>(A, B, C);
    ...
}
```

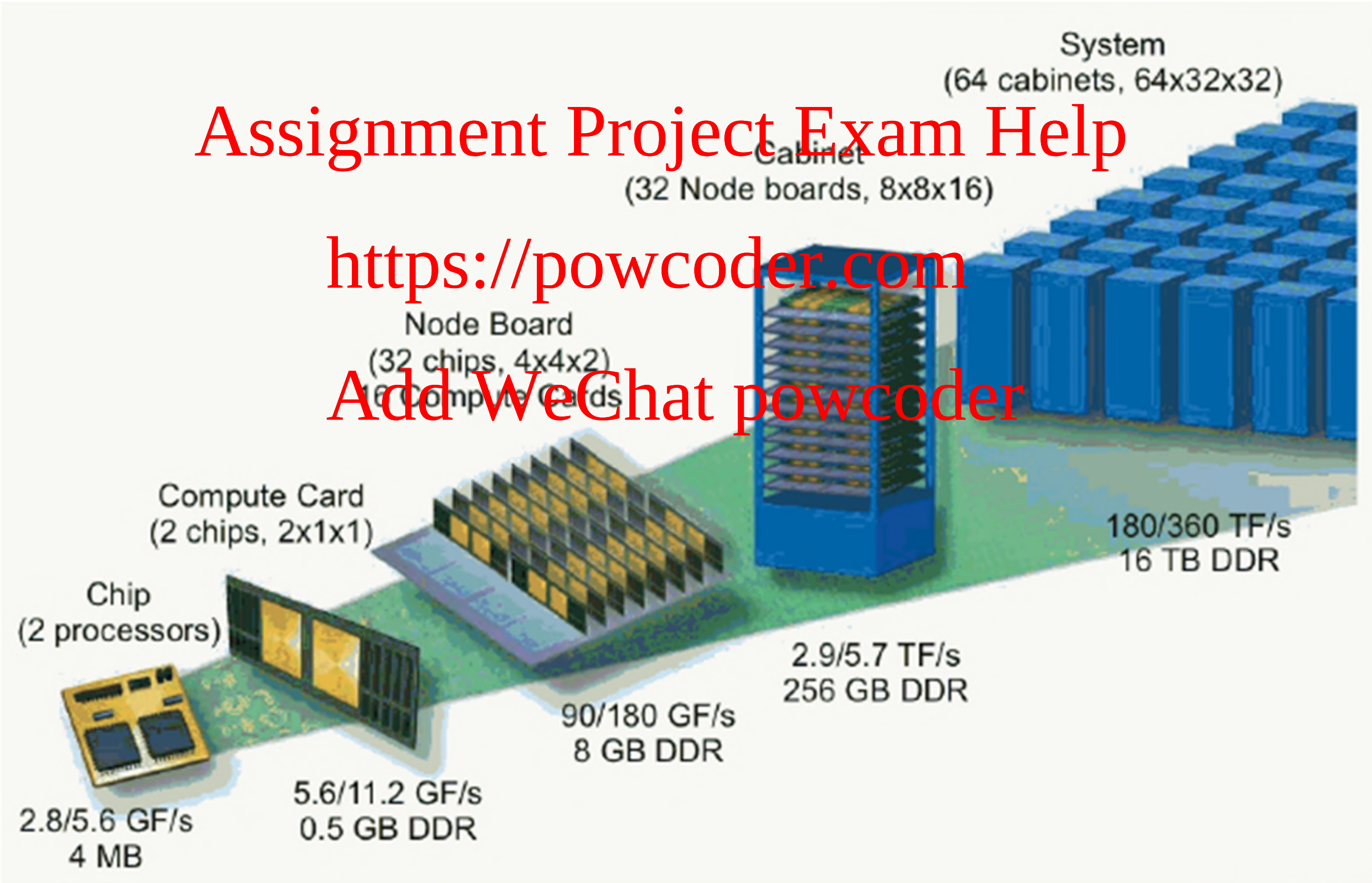


History and Evolution of HPC Systems

- 1960s: Scalar processor
- 1970s: Vector processor
- Later 1980s: Massively Parallel Processing (MPP)
  - Up to thousands of processors, each with its own memory
  - Processors can fetch and run instructions in parallel
  - Break down the workload in a parallel program
    - Workload balance and processor communications
  - Difference between MPP and vector processor

Architecture of BlueGene/L (MPP)

- Create a philosophy of using a massive number of low performance processors to construct supercomputers



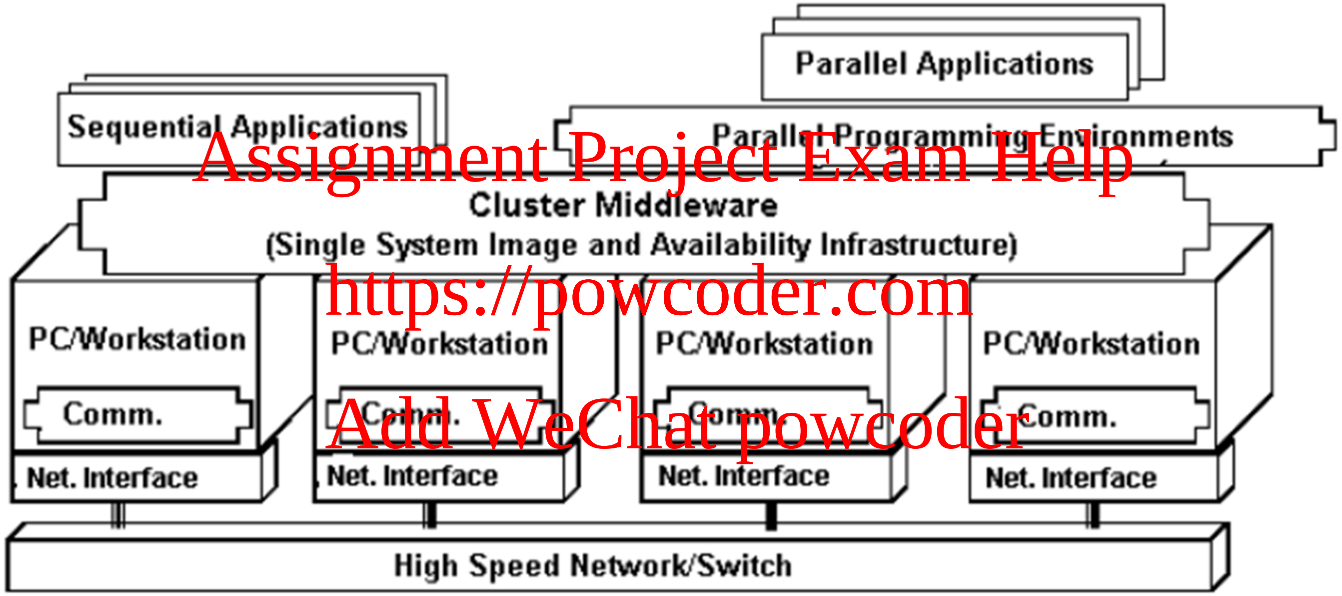
Source: IBM

History and Evolution of HPC Systems

- 1960s: Scalar processor
- 1970s: Vector processor
- Later 1980s: Massively Parallel Processing (MPP)

- Later 1990s: Cluster
  - Connecting stand-alone computers with high-speed network (over-cable networks)
    - Commodity off the shelve computers
    - high-speed network: Gigabit Ethernet, infiniband
    - Over-cable network vs. on-board network
  - Not a new term itself, but renewed interests
    - Performance improvement in CPU and networking
    - Advantage over custom-designed mainframe computers: Good portability

Cluster Arechitecture



History and Evolution of HPC Systems

- 1960s: Scalar processor
- 1970s: Vector processor
- Later 1980s: Massively Parallel Processing (MPP)

- Later 1990s: Cluster

- Later 1990s: Grid
  - Integrate geographically distributed resources

  - Further evolution of cluster computing
  - Draw an analogue from Power grid

History and Evolution of HPC Systems

- 1960s: Scalar processor
- 1970s: Vector processor
- Later 1980s: Massively Parallel Processing (MPP)

- Later 1990s: Cluster

- Later 1990s: Grid

- Since 2000s: Cloud
  - Commercialization of Grid and Cluster computing
  - Use the resources and services provided by the third party (Cloud service provider) and pay for the usage
  - virtualization technology: secure running environment, higher resource utilization

Architecture Types

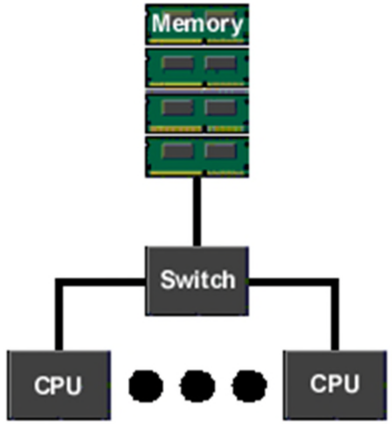
- All previous HPC systems can be divided into two architecture types
  - Shared memory system
  - Distributed memory system

Architecture Types

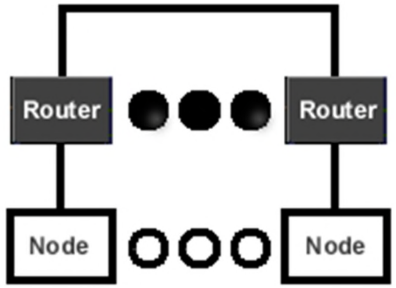
- **Shared memory (uniform memory access - SMP)**
  - **Multiple CPU cores, single memory, shared I/O (Multicore CPU)**
  - **All resources in a SMP machine are equally available to each core**
  - **Due to resource contention, uniform memory access systems do not scale well**
  - **CPU cores share access to a common memory space.**
    - **Implemented over a shared system bus or switch**
  - **Support for critical sections is required**
  - **Local cache is critical:**
    - **If not, bus/switch contention (or network traffic) reduces the systems efficiency.**
    - **Cache introduces problems of coherency (ensuring that stale cache lines are invalidated when other processors alter shared memory).**

Architecture Types

- Shared memory (Non-Uniform Memory Access: NUMA)
  - Multiple CPUs
  - Each CPU has fast access to its local area of the memory, but slower access to other areas
  - Scale well to a large number of processors due to the hierarchical memory access
  - Complicated memory access pattern: local and remote memory address
  - Global address space.



Node of a NUMA machine



NUMA machine

Architecture Types

- Distributed Memory (MPP, cluster)
  - Each processor has it's own independent memory
  - Interconnected through over-cable networks

  - When processors need to exchange (or share data), they must do this through an explicit communication

    - Message passing (MPI language)
  - Typically larger latencies between processors

  - Scalability is good if the task to be computed can be divided properly

Granularity of Parallelism

- Defined as the size of the computations that are being performed in parallel

- Four types of parallelism (in order of granularity size)
  - Instruction-level parallelism (e.g. pipeline)
  - Thread-level parallelism (e.g. run a GPU program)
  - Process-level parallelism (e.g. run an MPI job in a cluster)
  - Job-level parallelism (e.g. run a batch of independent jobs in a cluster)

Dependency and Parallelism

- Dependency: If instruction A must finish before instruction B can run, then B is dependent on A

- Two types of Dependency

  - Control dependency: waiting for the instruction which controls the execution flow to be completed

    - IF (X!=0) Then Y=1.0/X: Y=1.0/X has the control dependency on X!=0

  - Data dependency: dependency because of calculations or memory access
    - Flow dependency: A=X+Y; B=A+C;
    - Anti-dependency: B=A+C; A=X+Y;
    - Output dependency: A=2; X=A+1; A=5;

Parallel computing vs. distributed computing

- **Parallel Computing**
  - **Breaking the problem to be computed into parts that can be run simultaneously in different processors**
  - **Example: an MPI program to perform matrix multiplication**
  - **Solve tightly coupled problems**

- **Distributed Computing**
  - **Parts of the work to be computed are computed in different places (Note: does not necessarily imply simultaneous processing)**
  - **An example: running a workflow in a Grid**
  - **Solve loosely-coupled problems (no much communication)**