(1) Parallelization Abstraction

1. Parallelization steps:

1. Decomposition of applications into functional tasks/data blocks.
   - Factors:
     1. Application Type
     2. Concurrency degree (Embarrassingly parallel, Pleasingly parallel, Ember. serial)
     3. Granularity (Fine, medium, coarse)
     4. Target system (Shared-memory [fine-grain], Distributed-memory architectures [coarse-grain])
   - Types
     1. Data decomposition [matrix mult.] (work sharing, work stealing)
     2. Functional [diff. extraction] (list scheduling, work sharing, work stealing)
     3. Recursive decomp (Fibonacci, Mergesort)
     4. Exploratory decomp (min, max, pattern matching)
     5. Speculatory (if)
        - Can be combined.

2. Dependence analysis of decomposed functional task or data blocks.
   - Types:
     1. Control dependencies (if u=0 [control] then ... else ...)
     2. Data dependencies (a=2, u=a+2, v=a*7, x=u+v)

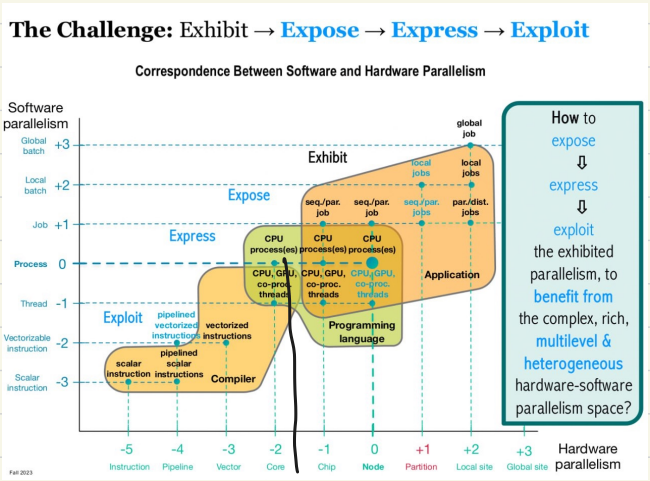
        Flow dependencies – one task wriets and other reads.
     3. Name dependencies
        - anti-dep – one reads and another writes a var
        - output-dep – both tasks write that var.

3. Mapping: of the execution of functional task on data blocks onto target system.
   - When? ◦ Temporal ordering – [when] start time to each subtask.
   - Where? ◦ Spatial assignment – [where] placment on processing elements, according to temporal ordering
   - How do map? ◦ Static | Dynamic ⟷ [or] offline | online.

4. Programming: express all the parallelism 1-3 via a prog. language.
   1. Shared memory (with threads)
   2. Shared memory (without threads)
   3. Distributed-memory
   4. Data parallel
   4. Hybrid distributed shared-memory
   5. Single Program Multiple Data
   6. Multiple Program Multiple Data

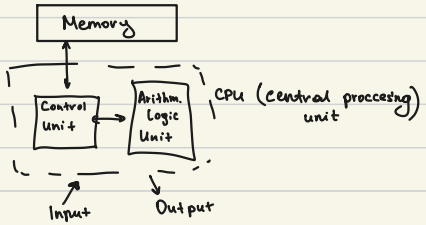
App —(Decomp. + Depend. Anal.)→ Algorithm —(Mapping)→ Software —(remapping)→ Hardware —(Optimization)→ Execution

2. Parallelism: Forms and Levels



3. Architecture computer:

1. Von Neumann 1945



2. Flynn's Taxonomy (Parallel computers)

Simple Instruction Stream / Single Data Stream	
SISD	SIMD
MISD	MIMD ← most popular

- Small processing unit (PU)
- CU, control unit

- Global task executed in parallel require synchronization/data exchange over network.

a) SIMD

b) MIMD:

	shared var	message passing
Global	GMSV	GMMP
Distributed	DMSV	DMMP

Limitations: nothing about organization of memory

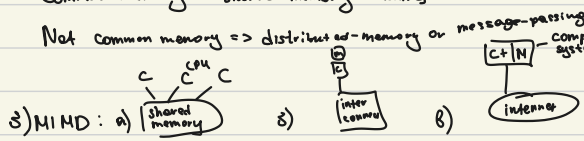
3. Flynn - Johnson's 1988

Location: Centralized / Distributed

Access policy: Full / No access

Common memory => shared-memory machines

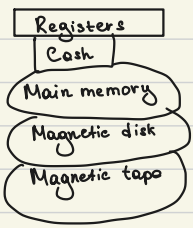
Not common memory => distributed-memory or message-passing machines



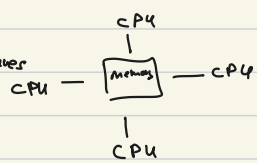
3) MIMD: a) β) γ)

4. **Memory architectures**

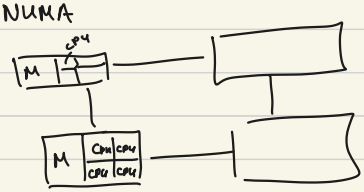
1. Hierarchy



- Shared memory machines
  - UMA, Uniform Memory Access architectures
    - visible to all CPUs

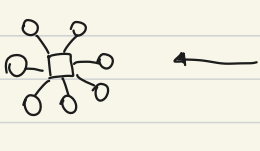


  - NUMA



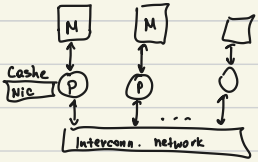
Model 1: Shared memory: (Symmetric multiprocessing and UMA)

=> UAS: UMA

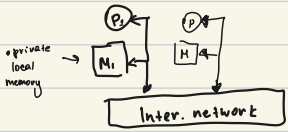


UMA + UAS (Unified address space)

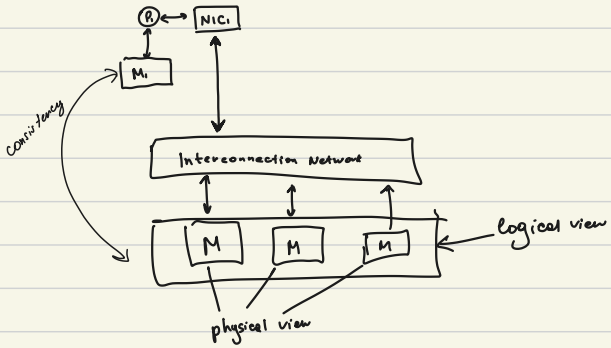
- Model 2: Distributed memory (Asymmetric multiprocessing and NUMA)



- Model 3: Hybrid distributed shared-memory (
  - GAS - Global Address Space
  - A/symmetric Multiprocessing



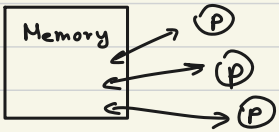
- Model 4: Distributed shared-memory



Model 5: Fully distributed (connection via WAN)

Parallel Programming models

1. Shared memory model (without threads)

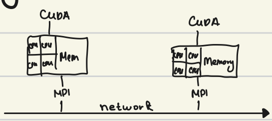


(read and write asynchronously)

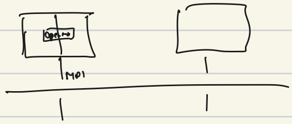
2. Shared-memory (with threads)
   - Processes have threads, that can be concurrent.
   - Each thread has local data, shares global memory.
   - „Communicate" via global memory.

3. Distributed-memory / Message-Passing

4. Hybrid Model



or



a) with MPI (message passing model) and CUDA

b) with MPI and OpenMP (threads model)

5. SPMD

6. MPMD (program)

Network topologies

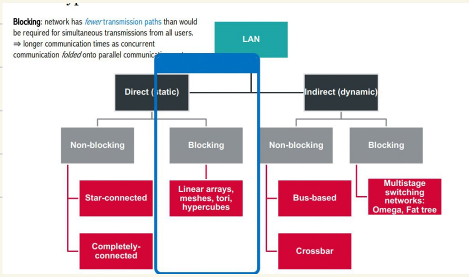
1) On chip network
2) System\Storage area networks
3) Local area networks
4) Wide area network.

$\mu = 0$

$= \exp\left(\frac{\varepsilon u(d,r)}{2\Delta u}\right)$ $\Delta u =$

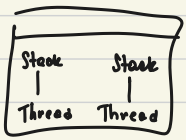
$\sigma^2 = \frac{2s^2 \ln\left(\frac{1.25}{\delta}\right)}{\varepsilon^2}$

3: LAN



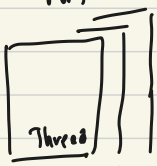
o Parallel Programming Shared Memory

OpenMP



Shared - Memory Ap

MPI



Single - thread process

Message passing approach