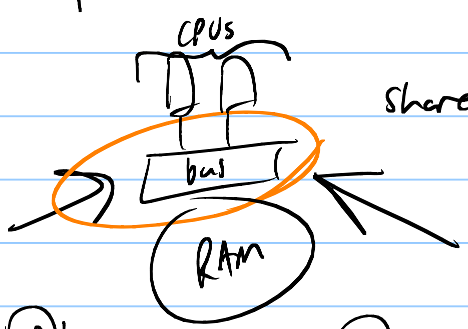
Parallel & Concurrent Prog 1

Note Title 2/1/2006

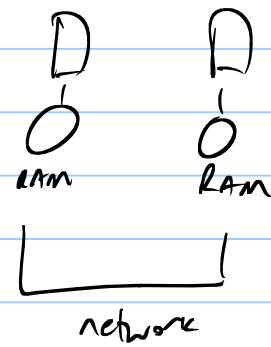
no exams
homework probs. — PLangs, libraries
one term project

---

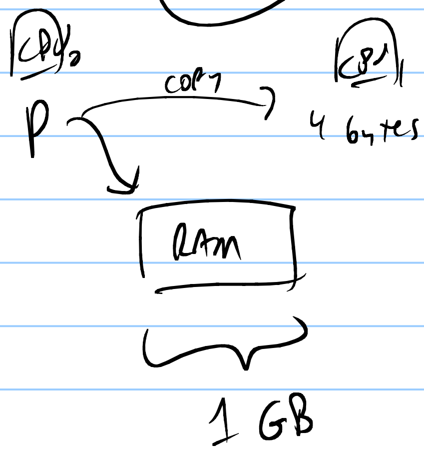
parallel machines

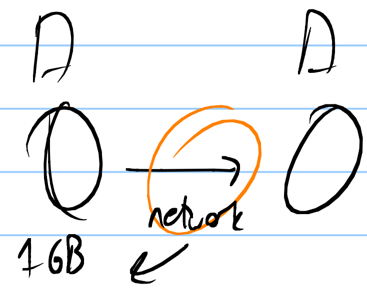


shared memory

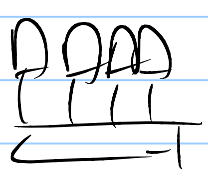


distributed memory.





Contention
Scalability bottleneck



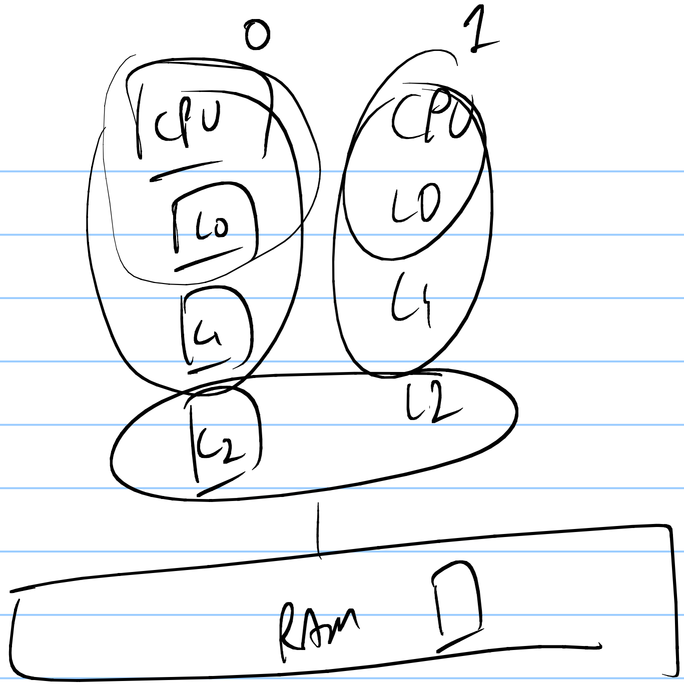
Symmetric Multi Processors

(SMP)

< 64 processors

NUMA

non-uniform memory access

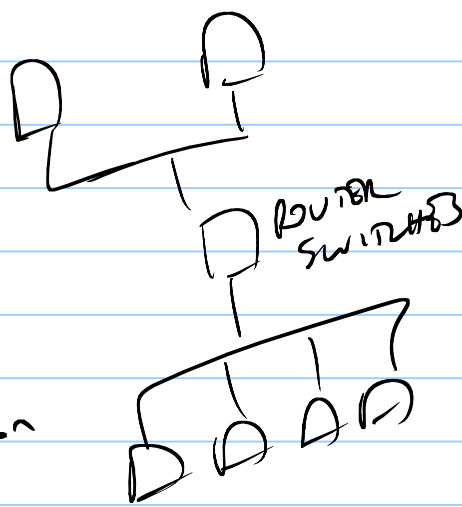


consistent
coherent

cache coherency

CC-NUMA

GRID

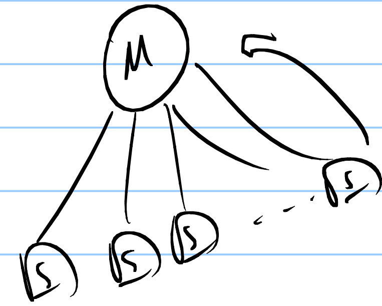


distributed memory

latency of communication is HIGH

b/w is relatively low

master-slave model



embarrassingly parallel — SETI@home
Folding@home

— rendering

scientific computing

divide & conquer

- matrix mult
- FFT
- n-body problems
- SOR
  successive over-relaxation

FORTRAN

extracting ||

express ||ism

+ FORTRAN:

no recursion

no pointers

— automatic ||

architectures

Procedure

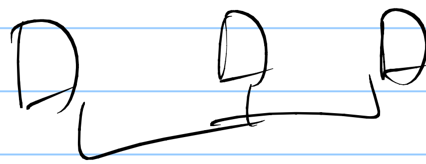
common a b c d . . .

$

a z

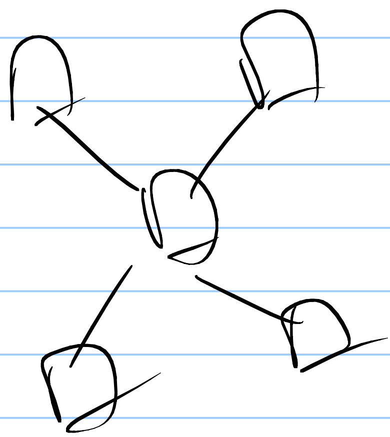
com ÷ z

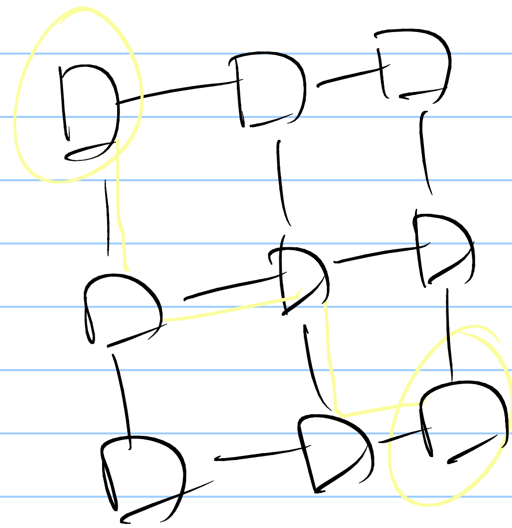
topologies



bus

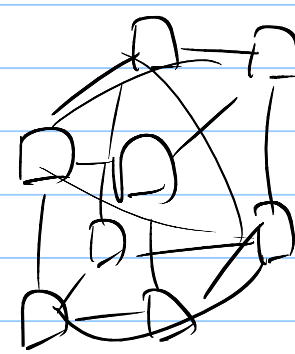
star

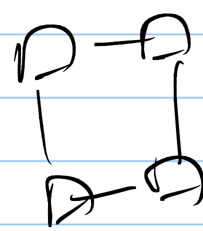




grid

hypercube





ring

libraries PVM, MPI

PL extensions message-passing

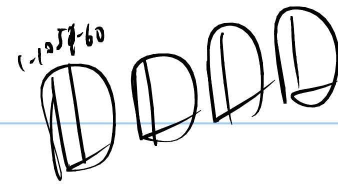
MPI_Send(. . . .)

MPI_Recv(. . . .)

HPF

OpenMP

#pragma openmp stripe(10)
for i = 1 to 1000
{
a[i] = a[i-1] * 2



conventional apps — C / C++ / Java/C#

C / C++: everything else

Java/C#: Eclipse, Azureus

threads

high-performance
low-reliability

C/C++ POSIX threads

pthread_create(fn (t))

pthread_join(t, result)

synchronization

Java { threads, synchronization, PL support

race condition

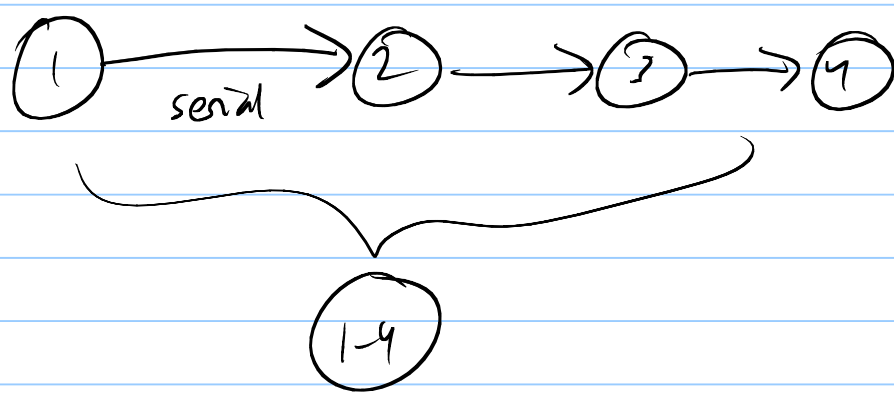
nondeterminism

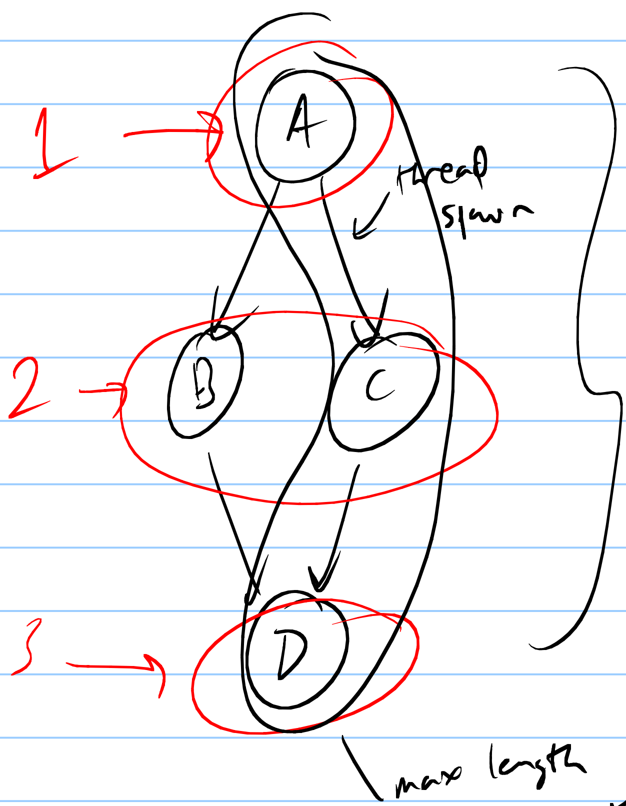
Darpa High-Productivity Computing

Cray - Chapel
Sun - Fortress   Guy Steele
IBM - X10   Vivek Sarkar

explicit parallel langs. Java (w/threads)
implicit parallel langs.

DAGs as a computation model





program: 4 nodes

T_1 = work = 4

time 1 processor

$\frac{T_1}{p}$ } optimal parallel speedup

critical path = T_∞

$$T_p = O\left(\frac{T_1}{p}\right) + cT_\infty$$

"Amdahl's Law"

greedy schedule / Brent schedule

Cilk

```
int fib (int v) {
    if (v ≤ 1) {
        return 1;
    } else {
        int v1 = spawn fib(v-1);
        int v2 = spawn fib(v-2);   sync;
        return v1+v2;
    }
}
```

A

D

0	1	2	3
1,	1,	2,	3

fib(3) :

v1 = fib(2)   2

v2 = fib(1)   1

return 3.

work-stealing