

PARALLEL ARCHITECTURES

Course “Parallel Computing”

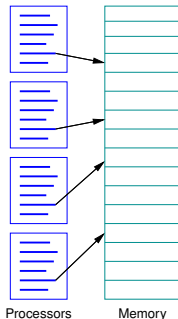
Parallel Random Access Machine (PRAM)

■ A simple abstract machine model.

- Arbitrarily many processors execute same program on single shared memory.
- Processors run synchronously in “lock-step”, possibly on different memory locations.
- Cost of accessing memory is $O(1)$.

■ Read/write conflicts have to be resolved.

- EREW (exclusive read, exclusive write).
- CREW (concurrent read, exclusive write).
- CRCW (concurrent read, concurrent write) with multiple modes: common (only same value may be written), arbitrary (random value is written), priority (value of lowest numbered processor is written), . . .

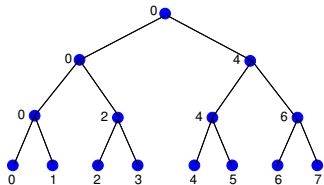


Traditional model for the analysis of parallel algorithms.

A PRAM Program

Multiply two matrices A and B of dimension $n = 2^m$.

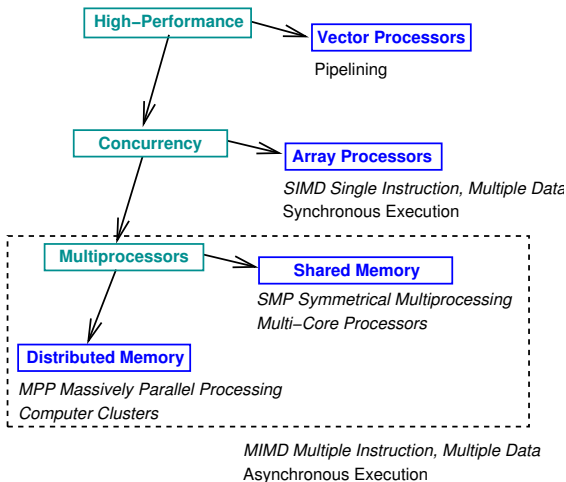
```
for i=0 to n-1 do in parallel
  for j=0 to n-1 do in parallel
    for k=0 to n-1 do in parallel
      P[i,j,k] = A[i,k]*B[k,j]
    for s=0 to m-1 do
      for t=0 to n-1 do in parallel
        if t%(2^(s+1)) = 0 then
          P[i,j,t] = P[i,j,t]+P[i,j,t+2^s]
      C[i,j] = P[i,j,0]
```



- First compute $P[i, j, k] = A[i, k] \times B[k, j]$ in time $O(1)$.
- Then compute $C[i, j] = \sum_k P[i, j, k]$ in time $O(\log n)$.
 - Computation of sum by $\log n$ stages of pairwise additions.

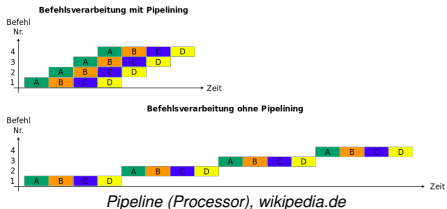
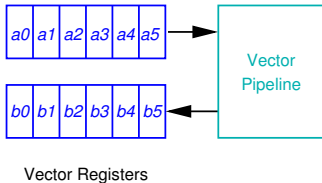
Matrix multiplication in time $O(\log n)$ with $O(n^3)$ processors.

High-Performance Architectures



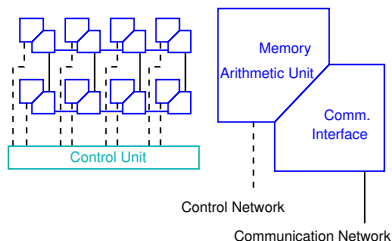
Flynn's Classification: SIMD versus MIMD.

Vector Processors



Vectors of data are processed by pipelines; speedup is limited by the (fixed) pipeline depth.

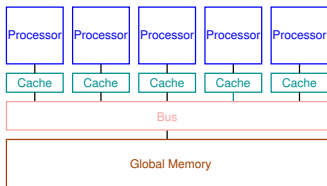
Array Processors



```
plural double matrix_multiply(A, B)
  plural double A,B;
{
  int i;
  plural double C = 0.0;
  ...
  for (i=0; i<nxproc; i++) {
    C += A*B;
    xnetW[1].A = A;
    xnetN[1].B = B;
  }
  return C;
}
```

Array of arithmetic units operates in lock-step.

Shared Memory Multi-Processors

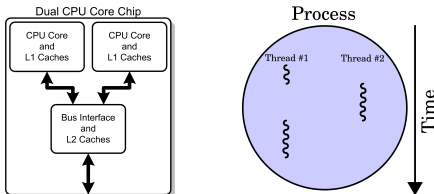


Alternative Term: SMP (Symmetric Multiprocessing)

- Multiple asynchronously operating processors.
- Single OS image schedules processes to processors.
- Single shared memory accessible via central bus.
 - Only one processor at a time can read/write memory.
 - Processors connected to bus via *coherent* caches.
 - *Snooping protocol*: whenever a cache sees another processor's write, it updates its local cache copies.

Scalable to 16 processors or so.

Multi-Core Processors

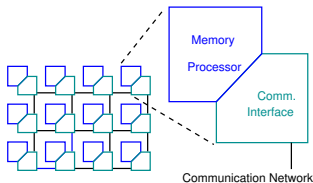


Multi-core processor, Thread, en.wikipedia.org

- Processors hold multiple processing units (“cores”).
 - Each core has a separate Level 1 cache.
 - Cores share a common Level 2 cache.
- Cores may execute multiple threads independently.
 - Threads: light-weight processes that can be independently scheduled for execution.
 - Processes: containers that hold multiple threads that have access to the same memory.

Today, actually every processor is by itself an SMP system. 7/19

Distributed Memory Multi-Processors



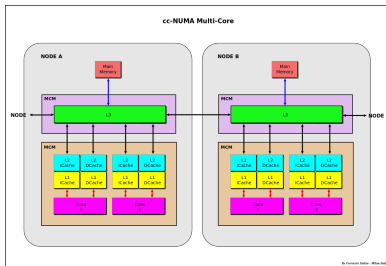
Alternative Term: MPP (Massively Parallel Processing)

- Many identical nodes that operate asynchronously.
 - Processor, local memory, communication interface.
- Each node runs its own OS image.
 - New processes are scheduled to the local processor.
- Nodes connected by high-bandwidth/low-latency network.
 - Different topologies (grid, tree, hypercube, ...).
 - Different network technologies (InfiniBand, OmniPath, ...)
 - Remote processes can communicate by *message passing*.

Scalable to thousands of processors.

Virtual Shared Memory Multi-Processors

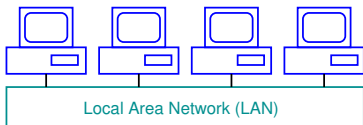
- ccNUMA: “cache coherent non-uniform memory access”.
- All local memories combined to single address space.
- NUMA: access to remote memory is more expensive.
- Directory keeps track of which nodes hold cache copies of which lines of local memory.
- If local memory line is updated, nodes with copies are informed.



Cache memory, en.wikipedia.org

Implementation of SMP model on top of MPP hardware.

Computer Clusters



Also called “Beowulf” systems.

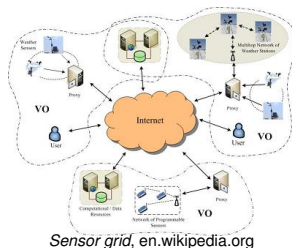
- A cheap alternative to MPP systems.
 - Each node is an independent of-the-shelf computer.
 - Connectivity provided by conventional Ethernet (or dedicated high-speed, e.g., InfiniBand) connections.
- A software stack implements MPP capabilities.
 - MPI, cluster managers, workload schedulers, ...
- Special programming and data processing software.
 - Apache Hadoop/MapReduce, Apache Spark, ...

Also MPPs are based on cluster-technology, so categories blur.

Computational Grids

Infrastructures composed of resources from multiple networks.

- Heterogenous combination of various kinds of resources.
 - Computing power, data storage, sensors, . . .
 - Located in different administrative domains.
- Connected by “Grid middleware”.
 - Globus Toolkit, gLite, Unicore, . . .



Software-based implementation of a (widely distributed) “virtual supercomputer”.

The JKU Supercomputer “Mach”

SGI UV-1000 an der Johannes Kepler Universität

- Rechnerarchitektur
- Prozessortyp
- Prozessoranzahl
- Speicher
- Betriebssystem
- Prozessorleistung

SGI UV-1000 shared Memory/cc-numa Architektur

Intel E78837 (Westmere - EX)
X86-64, 2.66GHz / 8-Cores / 24MB Cache
256 (2048 Cores)

16 TB shared Memory

Linux – Suse SLES 11 mit SGI Performance Suite

gesamt Peak = 21,3 TFlops

Spec_2006_INT Rate = ~39.000

Spec_2006_FP Rate = ~29.000

Stream = 5,8 Tbyte/s

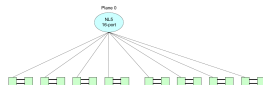
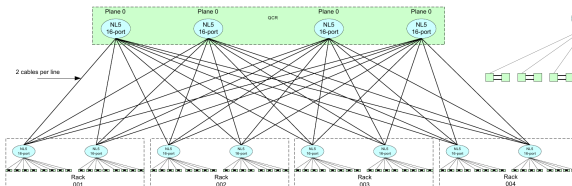
Linpack 100 = ~2,2 Gflop/s

Linpack NxN = 18,5 Tflop/s

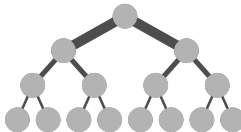
- Memory-Bandbreite
- Bisection-Bandbreite

7,5 TB / s

480 GB / s

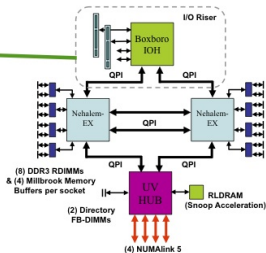


The Interconnection Topology “Fat Tree”



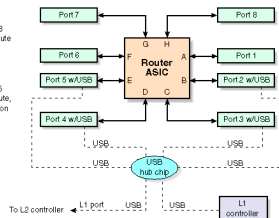
Fat tree, en.wikipedia.org

- Data links nearer to the top have higher bandwidth.
- Approximately same bisection bandwidth at each level.
- Implemented by proprietary NUMalink[®] technology.



Ports 1, 6, 7, and 8
can connect to compute
or MPX modules

Ports 2, 3, 4, and 5
can connect to compute,
MPX, or PCI expansion
modules



The Data Access Hierarchy

Data Hierarchy Layer	Latency	Normalized Access Time
L1 Cache	1.4 ns	1×
L3 Cache	23 ns	16×
Local Memory	75 ns	53×
Remote Memory	1 μ s	700×
Disk	2 ms	3.6·10 ⁶ ×

Processors	Cores	Router Hops
2	16	0
16	256	1
256	2048	3

Considering the placement of processes and data is important for achieving high performance on a NUMA system.

Our Course Machine

Actually, we are going to use Mach's "little brother" Zusie.

■ Configurations

- ☐ 1 blade: $2 \times 8 = 16$ cores, 128 GB RAM, 24 MB cache.
- ☐ Mach: $4 \times 32 = 128$ blades, 2048 cores, 16 TB RAM.
- ☐ Zusie: 32 blades, 512 cores, 2 TB RAM.

■ Login

```
ssh -X -l ... zusie.edvz.uni-linz.ac.at
```

- ☐ Only from JKU network.
- ☐ Another possibility for use from other networks.

Be considerate: this machine is shared by many users.

Architecture Information

```
zusie> topology
Serial number: UV-00000044
Partition number: 0
    32 Blades
    1024 CPUs
2002.99 Gb Memory Total
    64.00 Gb Max Memory on any blade
    0.00 Gb Partition Base Address
4031.98 Gb Partition Last Address
    4 I/O Risers
    1 InfiniBand Controller
    4 Network Controllers
    2 SCSI Controllers
    8 USB Controllers
    1 VGA GPU
```

```
zusie> cat /proc/cpuinfo
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 6
model          : 46
model name     : Intel(R) Xeon(R) CPU
stepping       : 6
cpu MHz        : 2267.156
cache size     : 24576 KB
....
processor       : 1023
vendor_id      : GenuineIntel
...
```

Hyper-Threading: 2 virtual cores per physical core,
thus 1024 cores in total.

User and Process Information

```
zusie> who
```

```
root      console      2016-07-11 09:06
ws98      pts/0             2016-09-30 12:29
hans      pts/2             2017-02-21 17:04
hans      pts/3             2017-02-22 08:24 (:2.0)
hans      pts/4             2017-01-13 14:31 (lilli.edvz.uni-linz.ac.at)
k313270   pts/5             2017-02-22 08:38 (amir.risc.uni-linz.ac.at)
k313270   pts/6             2017-02-22 08:42 (amir.risc.uni-linz.ac.at)
...
```

```
zusie> ps -fu k313270
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
k313270	82369	1	0	Feb13	?	00:00:00	ssh -N -L 9999:localhost:37
k313270	447710	447708	0	08:38	?	00:00:00	sshd: k313270@pts/5
k313270	447711	447710	0	08:38	pts/5	00:00:00	-bash
k313270	449736	447800	99	08:50	pts/6	00:00:01	ps -fu k313270

Displays all users and all processes running on your behalf.

Thread Information

```
zusie> top -H -u k313270
```

```
top - 08:52:17 up 226 days, 52 min, 21 users,  load average: 214.24, 227.65, 24
Tasks: 17374 total, 218 running, 17155 sleeping,   1 stopped,   0 zombie
Cpu(s): 91.5%us,  5.7%sy,  2.3%ni,  0.3%id,  0.1%wa,  0.0%hi,  0.1%si,  0.0%st
Mem:   2051061M total, 1288263M used,   762798M free,          0M buffers
Swap:   131071M total,    568M used,   130503M free,   1119275M cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
449747	k313270	20	0	62600	15m	1880	R	77	0.0	0:02.51	top
82369	k313270	20	0	60672	3432	2532	S	0	0.0	0:00.11	ssh
447710	k313270	20	0	106m	2540	1476	S	0	0.0	0:00.00	sshd
447711	k313270	20	0	55824	5684	2760	S	0	0.0	0:00.11	bash
447799	k313270	20	0	106m	2556	1476	S	0	0.0	0:00.16	sshd
447800	k313270	20	0	55824	5756	2800	S	0	0.0	0:00.28	bash

Displays all threads that are running on your behalf.

CPU Sets

```
zusie> cpuset -i /Upper256sh
```

```
zusie> my_cpuset.csh
```

```
cpuset: /Upper256sh
```

```
allowed resource Ids:
```

```
Cpus_allowed_list: 256-511
```

```
Mems_allowed_list: 32-63
```

```
zusie> jkutopt -s /Upper256sh
```

```
...
```

```
zusie> man jkutopt
```

```
...
```

JKUtop is a rewrite of top geared towards speed, especially on big SMP systems. ... It also has no functionality to display individual threads. Thanks to these measures it achieves a noticeable speedup on ridiculously large SMP systems compared to top from procs.

```
...
```

Creates new shell that confines processes to certain nodes.