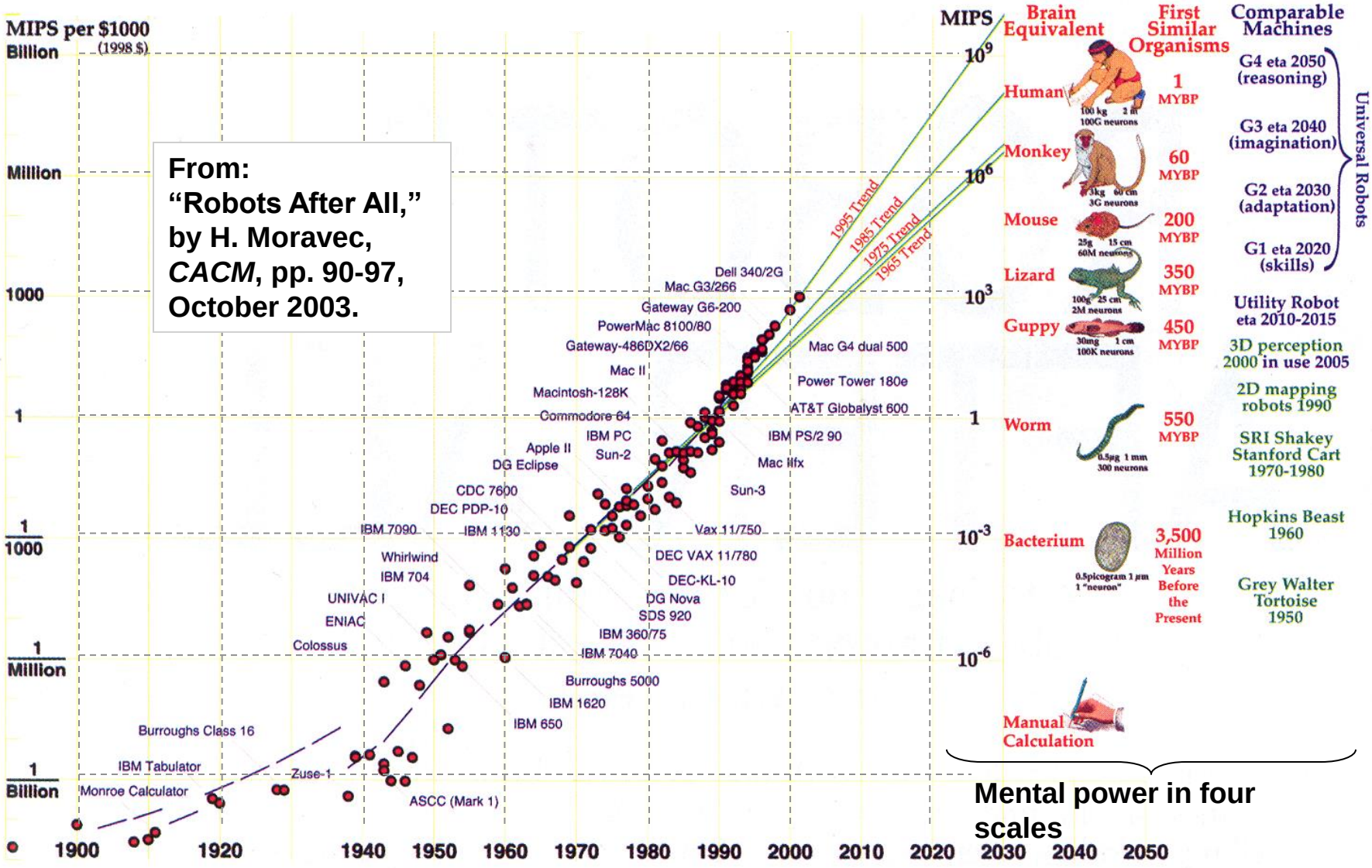
Referenced Books

- **Scalable Parallel Computing, Kai Hwang, Zhiwei, McGraw-Hill, 1998.**
- **Advanced Computer Architecture: Parallelism, Scalability, Programmability, Kai Hwang, McGraw-Hill, 1993.**
- **Advanced Computer Architecture: Parallel Processing, at the University of California, Santa Barbara.**
- ***Introduction to Parallel Processing: Algorithms and Architectures*, Parhami, B., Plenum Press, 1999.**
- **Computer Architecture and Parallel Processing by Kai Hwang, Tata McGraw-Hill.**

Motivation



From:
"Robots After All,"
by H. Moravec,
CACM, pp. 90-97,
October 2003.

Mental power in four scales

Introduction to Parallel Processing

- **Parallel Computer Architecture: Definition & Broad issues involved**
  - **A Generic Parallel Computer Architecture**
- **The Need And Feasibility of Parallel Computing**
  - **Scientific Supercomputing Trends**
  - **CPU Performance and Technology Trends, Parallelism in Microprocessor Generations**
  - **Computer System Peak FLOP Rating History/Near Future**

Why?

- **The Goal of Parallel Processing**
- **Elements of Parallel Computing**
- **Factors Affecting Parallel System Performance**
- **Parallel Architectures History**
  - **Parallel Programming Models**
  - **Flynn's 1972 Classification of Computer Architecture**
- **Current Trends In Parallel Architectures**
  - **Modern Parallel Architecture Layered Framework**
- **Shared Address Space Parallel Architectures**
- **Message-Passing Multicomputers: Message-Passing Programming Tools**
- **Data Parallel Systems**
- **Dataflow Architectures**
- **Systolic Architectures: Matrix Multiplication Systolic Array Example**

Parallel Computer Architecture

A parallel computer (or multiple processor system) is a collection of communicating processing elements (processors) that cooperate to solve large computational problems fast by dividing such problems into parallel tasks, exploiting Thread-Level Parallelism (TLP).

i.e Parallel Processing

- **Broad issues involved:**

Task = Computation done on one processor

  - **The concurrency and communication characteristics of parallel algorithms for a given computational problem (represented by dependency graphs)**
  - **Computing Resources and Computation Allocation:**
    - **The number of processing elements (PEs), computing power of each element and amount/organization of physical memory used.**
    - **What portions of the computation and data are allocated or mapped to each PE.**
  - **Data access, Communication and Synchronization**
    - **How the processing elements cooperate and communicate.**
    - **How data is shared/transmitted between processors.**
    - **Abstractions and primitives for cooperation/communication and synchronization.**
    - **The characteristics and performance of parallel system network (System interconnects).**
  - **Parallel Processing Performance and Scalability Goals:**

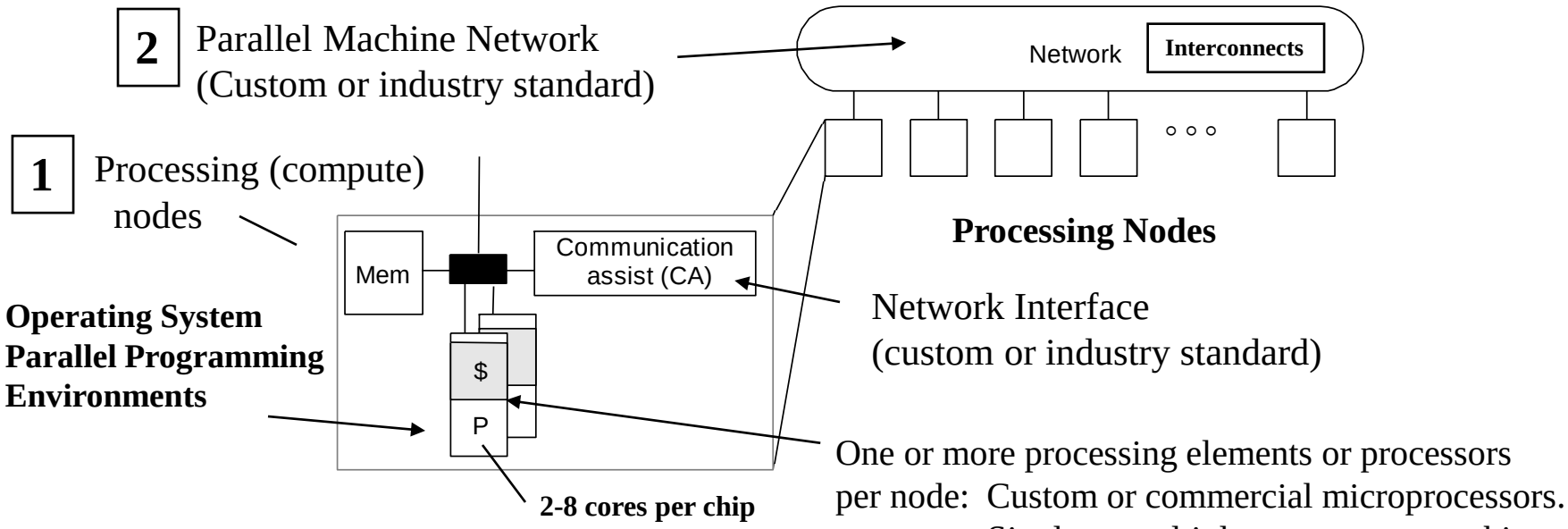
Goals

    - **Maximize performance enhancement of parallelism: Maximize Speedup.**
      - **By minimizing parallelization overheads and balancing workload on processors**
    - **Scalability of performance to larger systems/problems.**

Processor = Programmable computing element that runs stored programs written using pre-defined instruction set

Processing Elements = PEs = Processors

A Generic Parallel Computer Architecture



1 **Processing Nodes:**

Each processing node contains one or more processing elements (PEs) or processor(s), memory system, plus *communication assist:* (Network interface and communication controller)

2 **Parallel machine network (System Interconnects).**

Function of a parallel machine network is to efficiently (reduce communication cost) transfer information (data, results ..) from source node to destination node as needed to allow cooperation among parallel processing nodes to solve large computational problems divided into a number parallel computational tasks.

Parallel Computer = Multiple Processor System

The Need And Feasibility of Parallel Computing

- **Application demands: More computing cycles/memory needed**

Driving Force

  - *Scientific/Engineering computing*: CFD, Biology, Chemistry, Physics, ...
  - *General-purpose computing*: Video, Graphics, CAD, Databases, Transaction Processing, Gaming…
  - Mainstream multithreaded programs, are similar to parallel programs

- **Technology Trends:**

Moore's Law still alive

  - Number of transistors on chip growing rapidly. Clock rates expected to continue to go up but only slowly. Actual performance returns diminishing due to deeper pipelines.
  - Increased transistor density allows integrating multiple processor cores per creating Chip-Multiprocessors (CMPs) even for mainstream computing applications (desktop/laptop..).

- **Architecture Trends:**

+ multi-tasking (multiple independent programs)

  - Instruction-level parallelism (ILP) is valuable (superscalar, VLIW) but limited.
  - Increased clock rates require deeper pipelines with longer latencies and higher CPIs.
  - Coarser-level parallelism (at the task or thread level, TLP), utilized in multiprocessor systems is the most viable approach to further improve performance.
    - Main motivation for development of chip-multiprocessors (CMPs)

Multi-core Processors

- **Economics:**
  - The increased utilization of commodity of-the-shelf (COTS) components in high performance parallel computing systems instead of costly custom components used in traditional supercomputers leading to much lower parallel system cost.
    - Today's microprocessors offer high-performance and have multiprocessor support eliminating the need for designing expensive custom PEs.
    - Commercial System Area Networks (SANs) offer an alternative to custom more costly networks

Why is Parallel Processing Needed?

Challenging Applications in Applied Science/Engineering

Traditional Driving Force For HPC/Parallel Processing

- Astrophysics
- Atmospheric and Ocean Modeling
- Bioinformatics
- Biomolecular simulation: Protein folding
- Computational Chemistry
- Computational Fluid Dynamics (CFD)
- Computational Physics
- Computer vision and image understanding
- Data Mining and Data-intensive Computing
- Engineering analysis (CAD/CAM)
- Global climate modeling and forecasting
- Material Sciences
- Military applications
- Quantum chemistry
- VLSI design
- ….

Such applications have very high 1- computational and 2- data memory requirements that cannot be met with single-processor architectures.

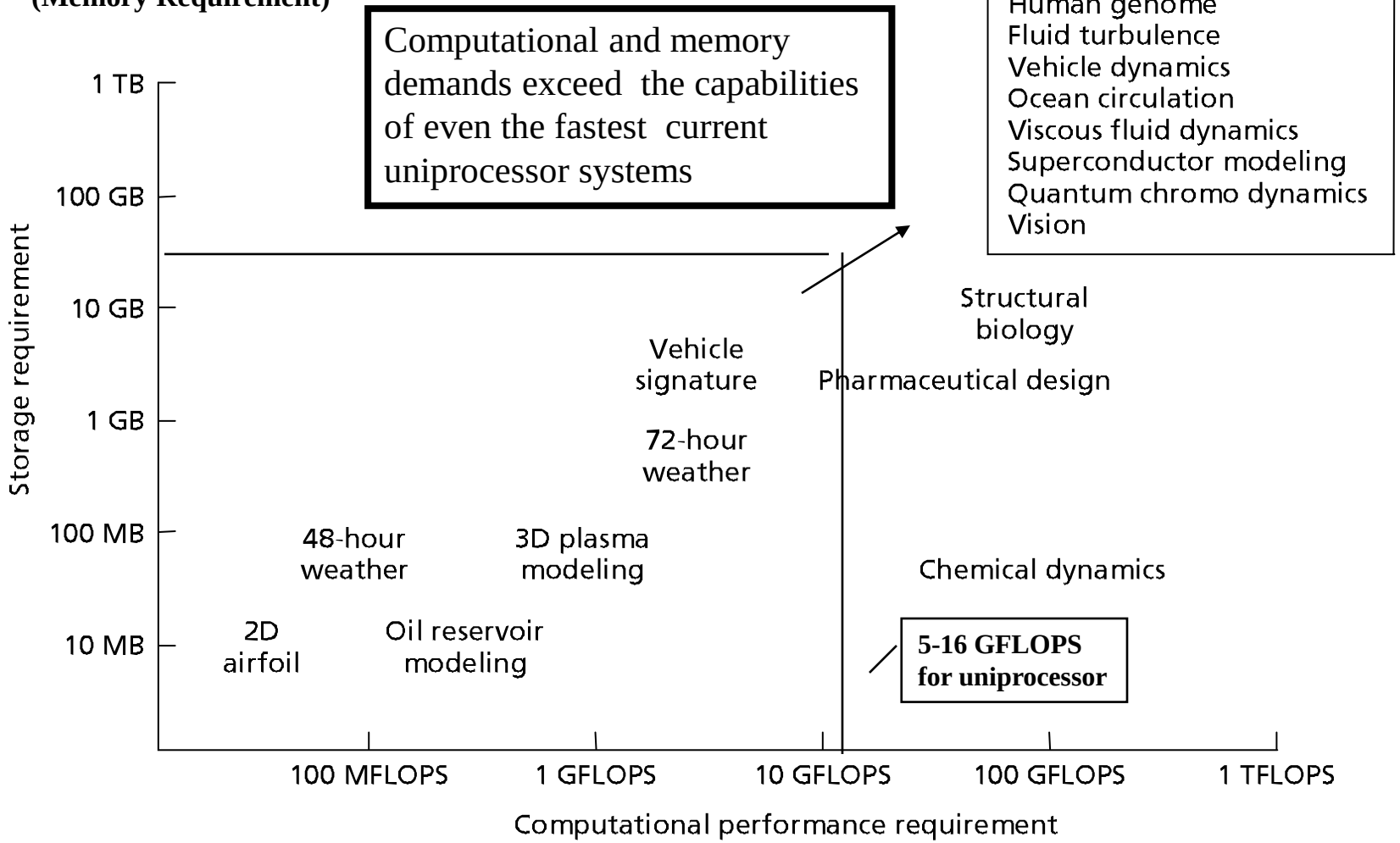
Many applications contain a large degree of computational parallelism

Driving force for High Performance Computing (HPC) and multiple processor system development

Why is Parallel Processing Needed?

Scientific Computing Demands

Driving force for HPC and multiple processor system development



GFLOP = 10^9 FLOPS TeraFLOP = 1000 GFLOPS = 10^{12} FLOPS
PetaFLOP = 1000 TeraFLOPS = 10^{15} FLOPS

Scientific Supercomputing Trends

- **Proving ground and driver for innovative architecture and advanced high performance computing (HPC) techniques:**
  - **Market is much smaller relative to commercial (desktop/server) segment.**
  - **Dominated by costly vector machines starting in the 1970s through the 1980s.**
  - **Microprocessors have made huge gains in the performance needed for such applications:**
    - **High clock rates. (Bad: Higher CPI?)**
    - **Multiple pipelined floating point units.**
    - **Instruction-level parallelism.**
    - **Effective use of caches.**
    - **Multiple processor cores/chip (2 cores 2002-2005, 4 end of 2006, 6-12 cores 2011)**

Enabled with high transistor density/chip

16 cores in 2013

However even the fastest current single microprocessor systems still cannot meet the needed computational demands.

As shown in last slide

- ***Currently:*** ***Large-scale microprocessor-based multiprocessor systems and computer clusters are replacing (replaced?) vector supercomputers that utilize custom processors.***

Uniprocessor Performance Evaluation

- **CPU Performance benchmarking is heavily program-mix dependent.**
- **Ideal performance requires a perfect machine/program match.**
- **Performance measures:**
  - **Total CPU time = $T = TC / f = TC \times C = I \times CPI \times C$**
    $= I \times (CPI_{execution} + M \times k) \times C$ (in seconds)

    TC = Total program execution clock cycles

    f = clock rate C = CPU clock cycle time = 1/f I = Instructions executed count

    CPI = Cycles per instruction $CPI_{execution}$ = CPI with ideal memory

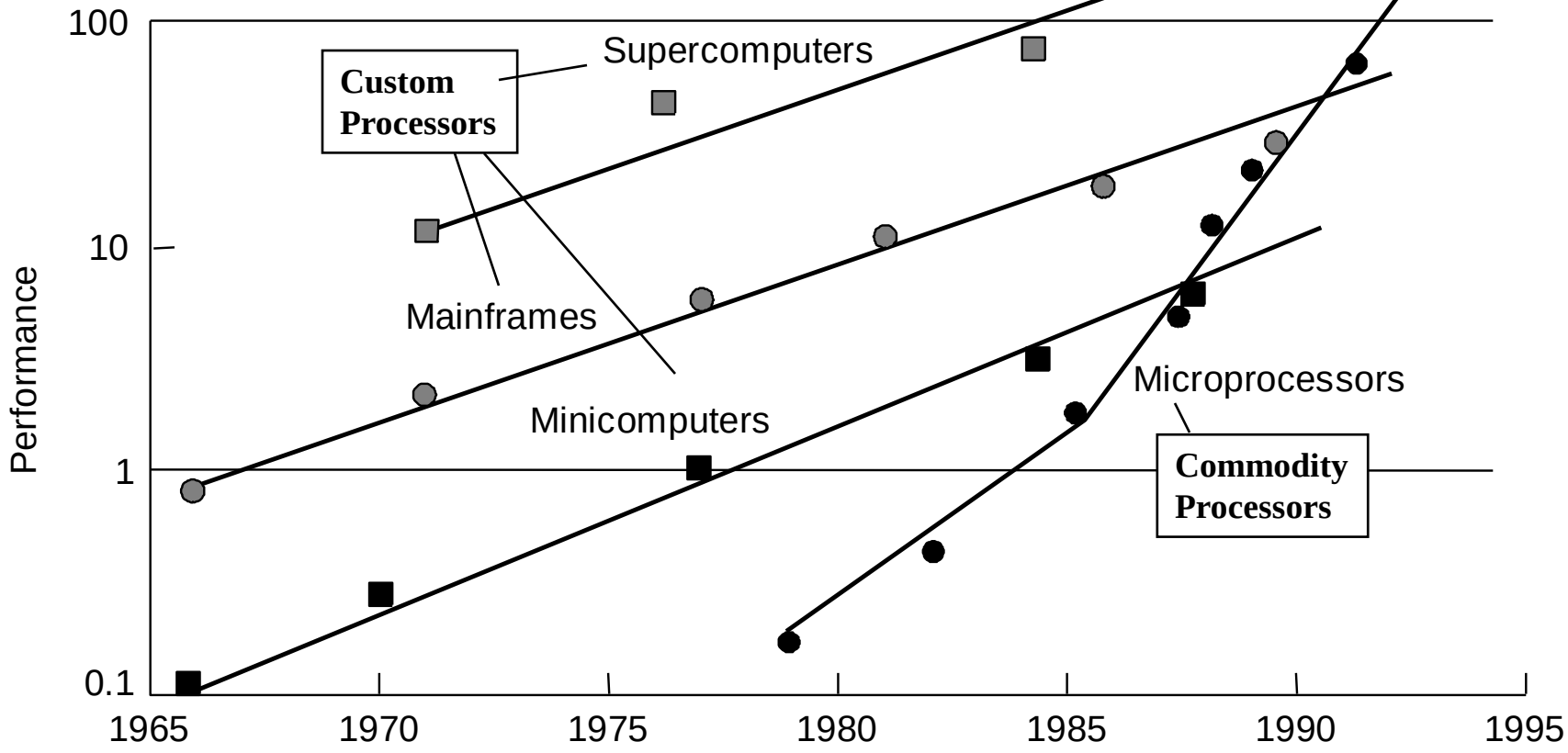
    M = Memory stall cycles per memory access

    k = Memory accesses per instruction
  - **MIPS Rating = $I / (T \times 10^6) = f / (CPI \times 10^6) = f \times I / (TC \times 10^6)$**
    (in million instructions per second)
  - **Throughput Rate: $W_p = 1/T = f / (I \times CPI) = (MIPS) \times 10^6 / I$**
    (in programs per second)
- **Performance factors: (I, $CPI_{execution}$, m, k, C) are influenced by: instruction-set architecture (ISA) , compiler design, CPU micro-architecture, implementation and control, cache and memory hierarchy, program access locality, and program instruction mix and instruction dependencies.**

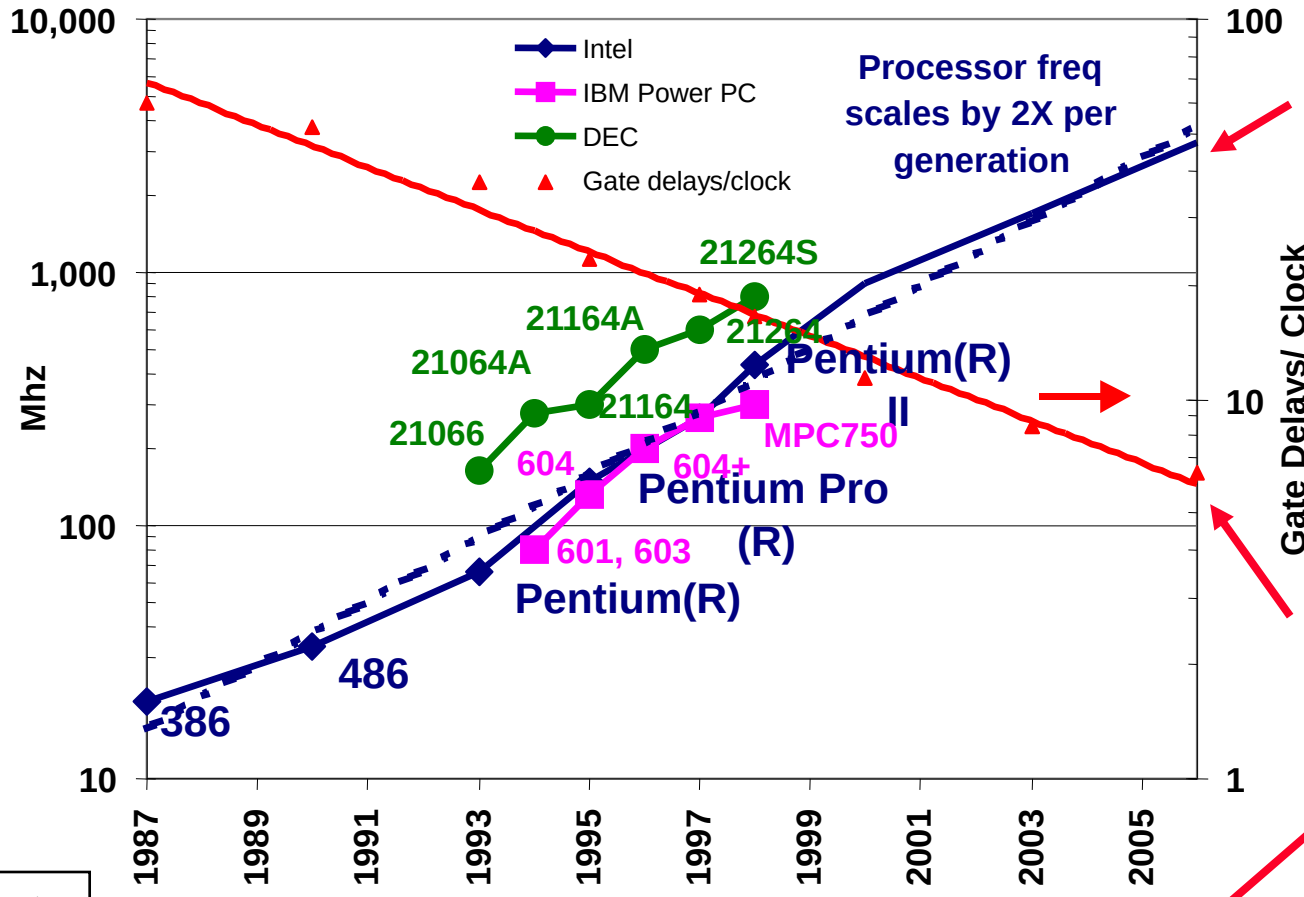
$T = I \times CPI \times C$

Single CPU Performance Trends

- **The microprocessor is currently the most natural building block for multiprocessor systems in terms of <u>cost and performance.</u>**
- **This is even more true with the development of cost-effective multi-core microprocessors that support TLP at the chip level.**



Microprocessor Frequency Trend



Realty Check:
Clock frequency scaling is slowing down!
(Did silicone finally hit the wall?)

Why?
1- Static power leakage
2- Clock distribution delays

Result:
Deeper Pipelines
Longer stalls
Higher CPI
(lowers effective performance per cycle)

No longer the case

- **Frequency doubles each generation** ?
- **Number of gates/clock reduce by 25%**
- **Leads to deeper pipelines with more stages**
  (e.g Intel Pentium 4E has 30+ pipeline stages)

Solution:
Exploit TLP at the chip level, Chip-multiprocessor (CMPs)

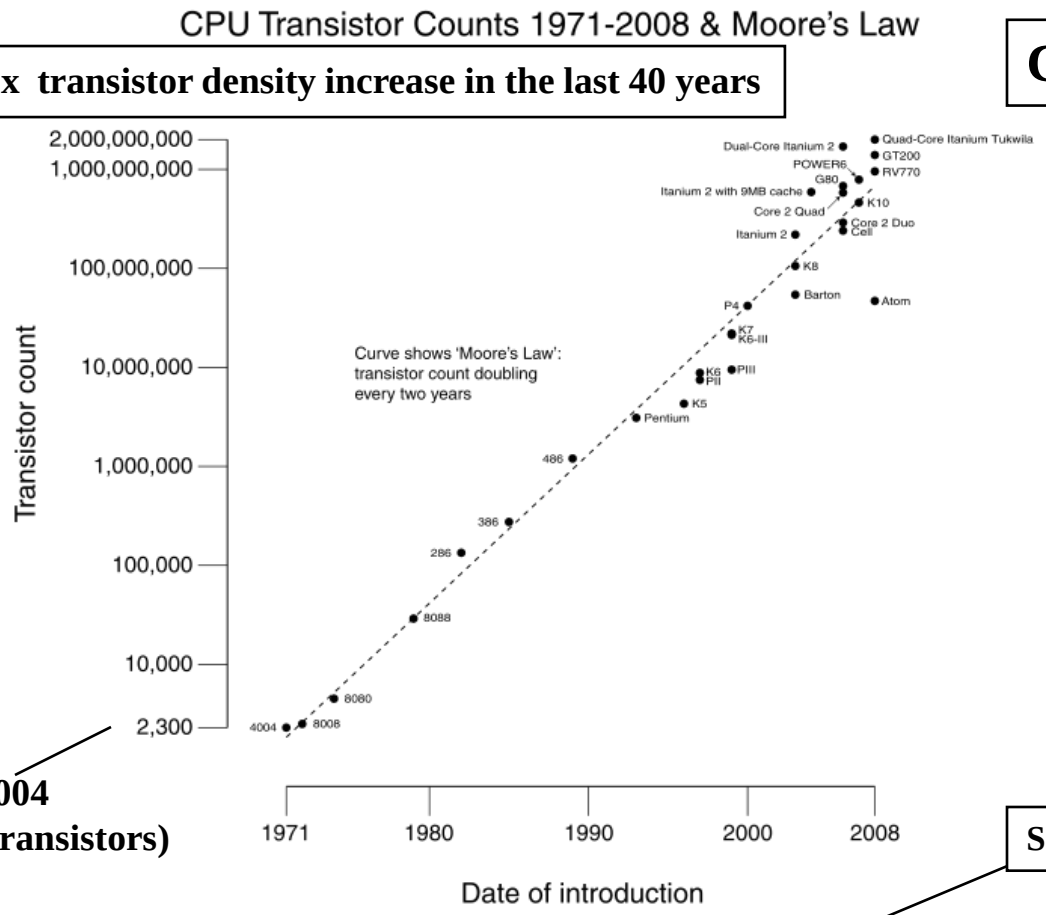
$$T = I \times CPI \times C$$

Transistor Count Growth Rate

Enabling Technology for Chip-Level Thread-Level Parallelism (TLP)

~ 3,000,000x transistor density increase in the last 40 years

Currently ~ 7 Billion



Moore's Law:

2X transistors/Chip Every 1.5 years (circa 1970) still holds

Enables Thread-Level Parallelism (TLP) at the chip level: Chip-Multiprocessors (CMPs) + Simultaneous Multithreaded (SMT) processors

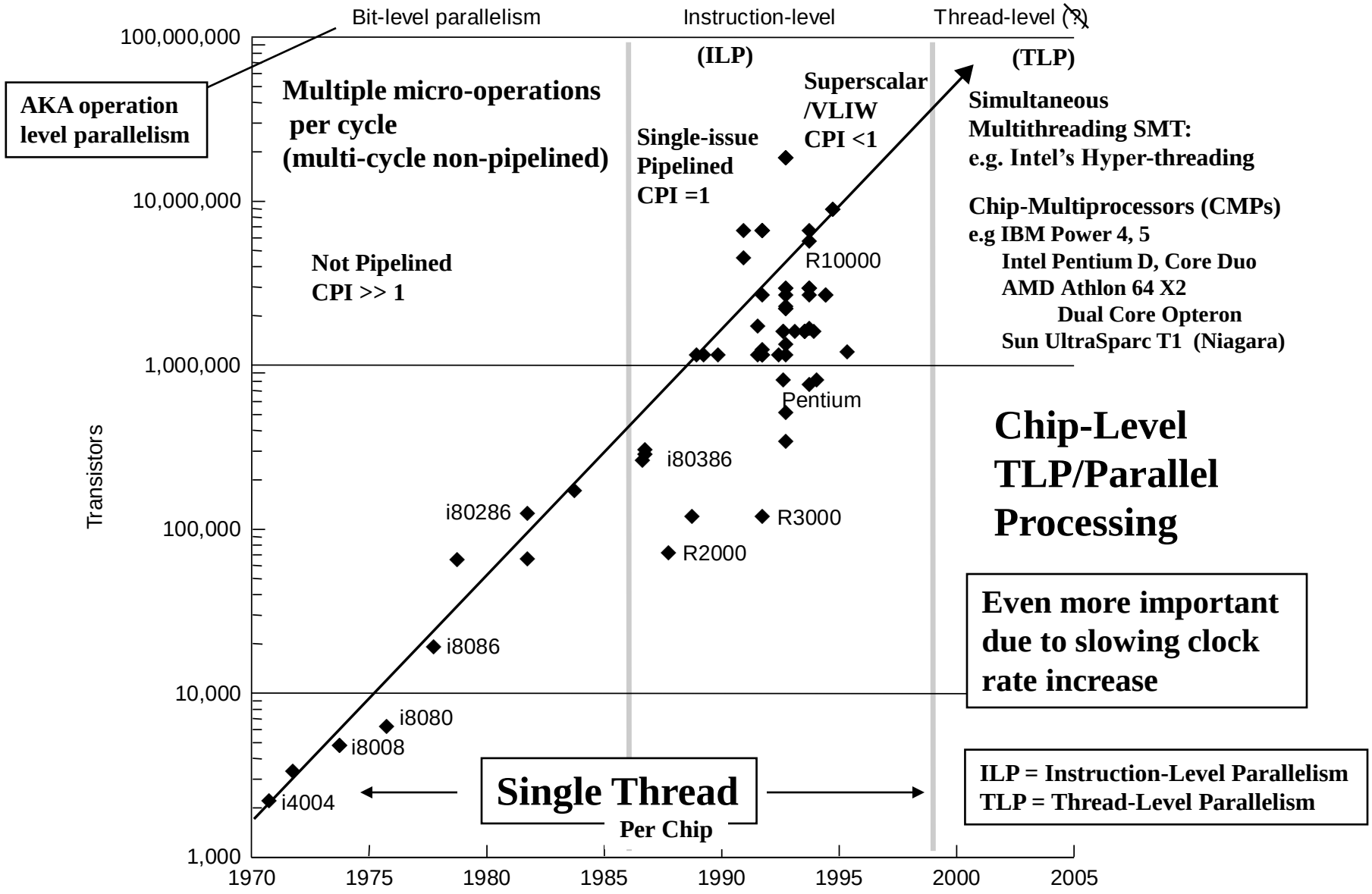
Intel 4004 (2300 transistors)

Solution

- **One billion transistors/chip reached in 2005, two billion in 2008-9, Now ~ seven billion**
- **Transistor count grows faster than clock rate: Currently ~ 40% per year**
- **Single-threaded uniprocessors do not efficiently utilize the increased transistor count.**

Limited ILP, increased size of cache

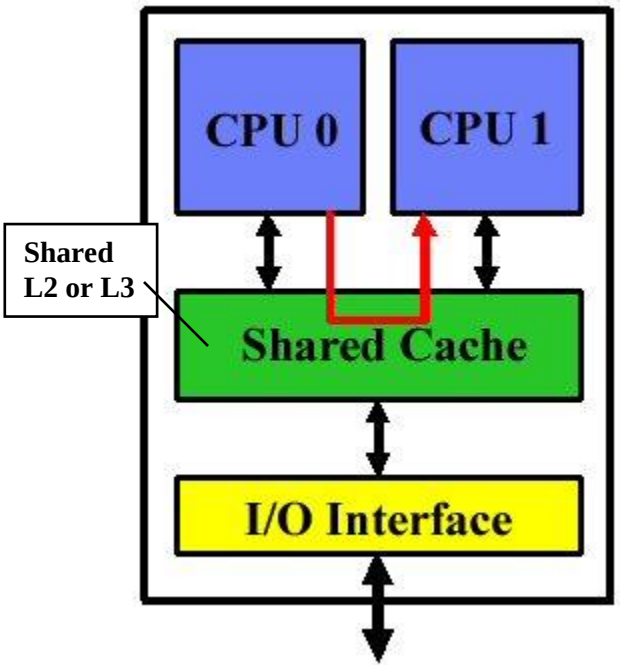
Parallelism in Microprocessor VLSI Generations



Improving microprocessor generation performance by exploiting more levels of parallelism

Dual-Core Chip-Multiprocessor (CMP) Architectures

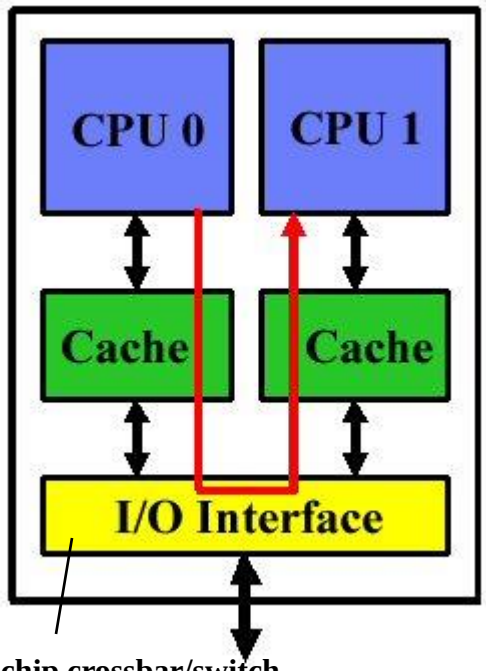
Single Die
Shared L2 Cache



Cores communicate using shared cache (Lowest communication latency)

Examples:
IBM POWER4/5
Intel Pentium Core Duo (Yonah), Conroe (Core 2), i7, Sun UltraSparc T1 (Niagara)
AMD Phenom ….

Single Die
Private Caches
Shared System Interface

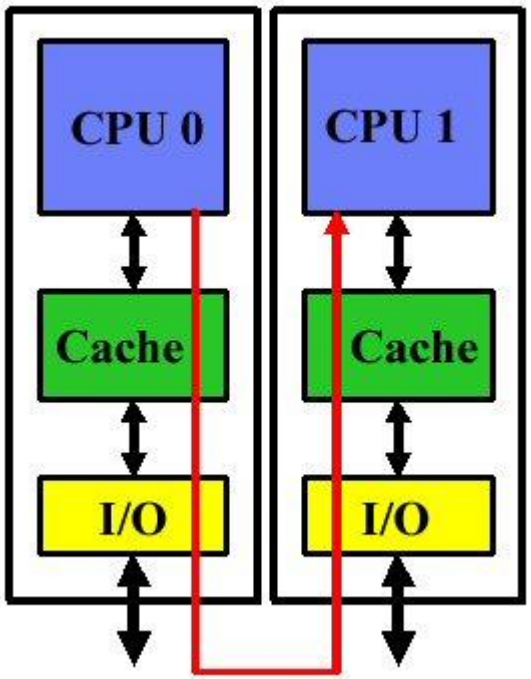


On-chip crossbar/switch

Cores communicate using on-chip Interconnects (shared system interface)

Examples:
AMD Dual Core Opteron, Athlon 64 X2
Intel Itanium2 (Montecito)

Two Dies – Shared Package
Private Caches
Private System Interface



FSB

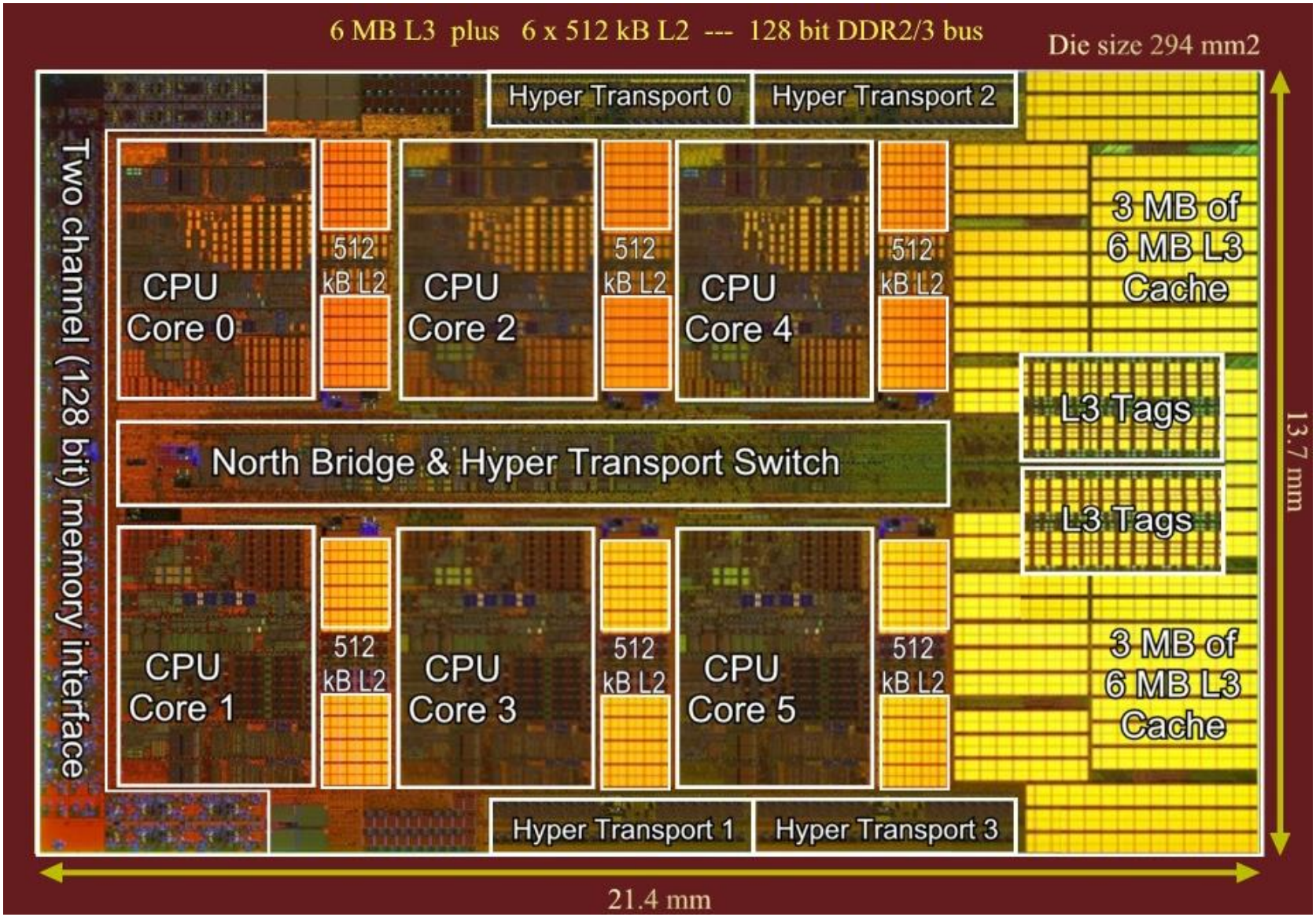
Cores communicate over external Front Side Bus (FSB) (Highest communication latency)

Examples:
Intel Pentium D,
Intel Quad core (two dual-core chips)

Source: Real World Technologies,
http://www.realworldtech.com/page.cfm?ArticleID=RWT101405234615

Example Six-Core CMP: AMD Phenom II X6

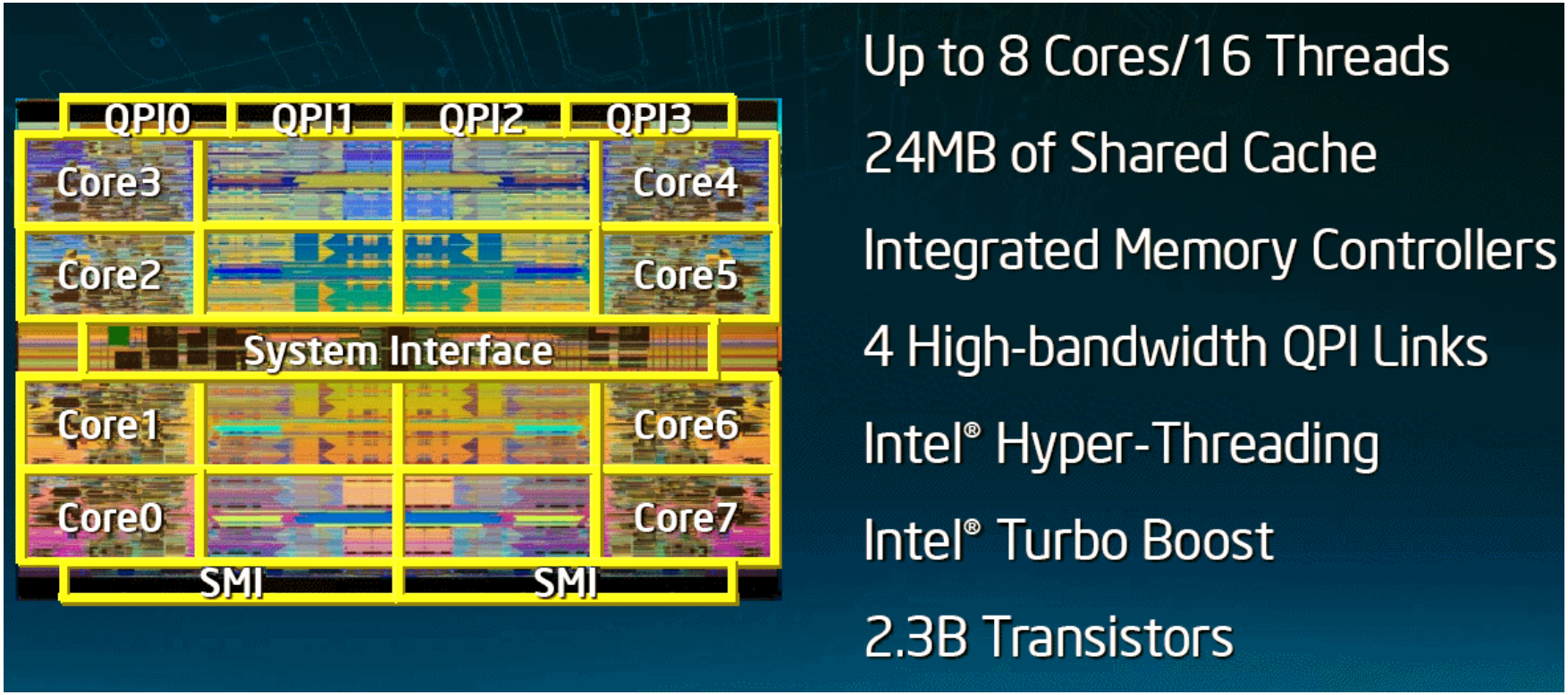
Six processor cores sharing 6 MB of level 3 (L3) cache



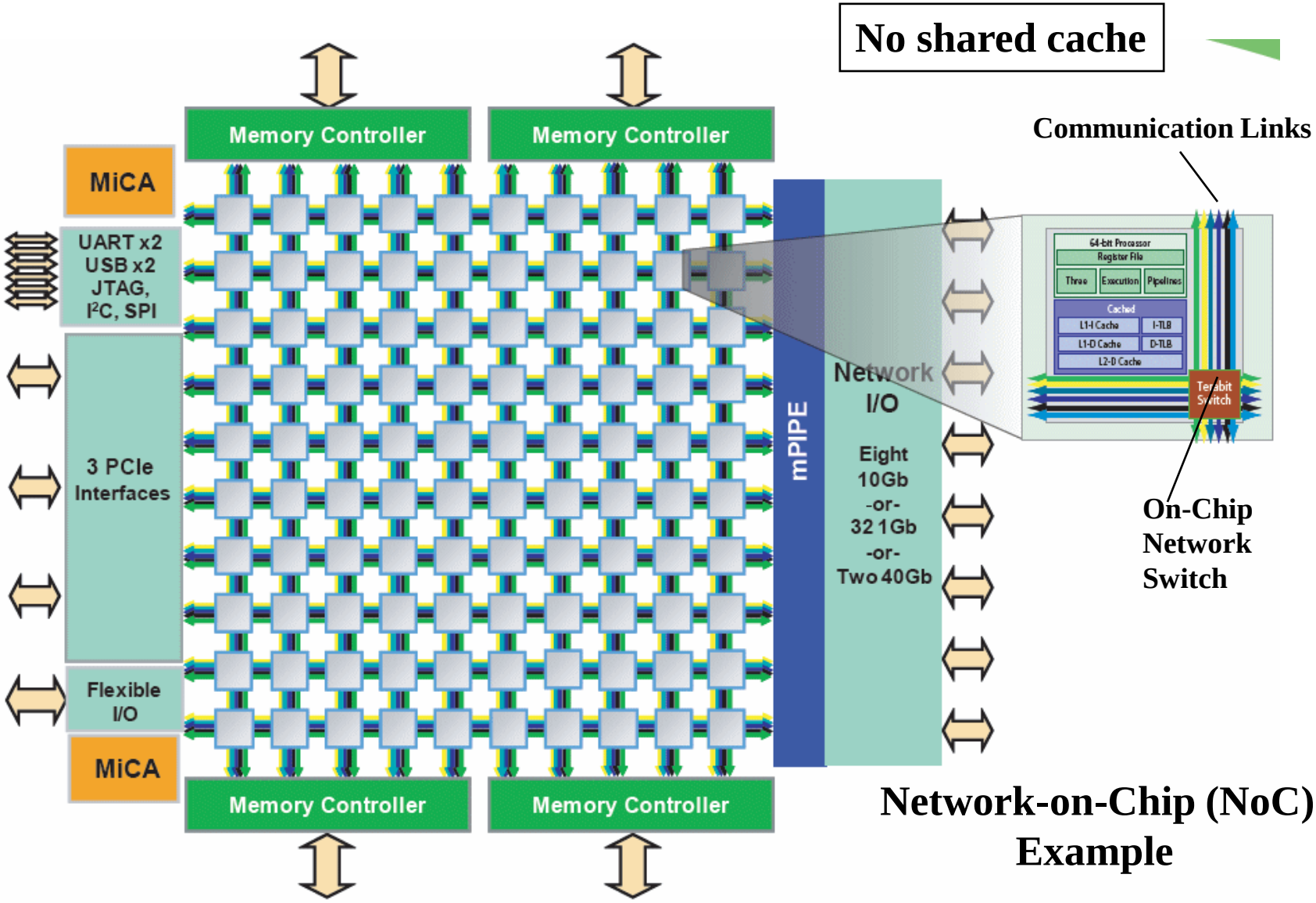
CMP = Chip-Multiprocessor

Example Eight-Core CMP: Intel Nehalem-EX

Eight processor cores sharing 24 MB of level 3 (L3) cache
Each core is 2-way SMT (2 threads per core), for a total of 16 threads



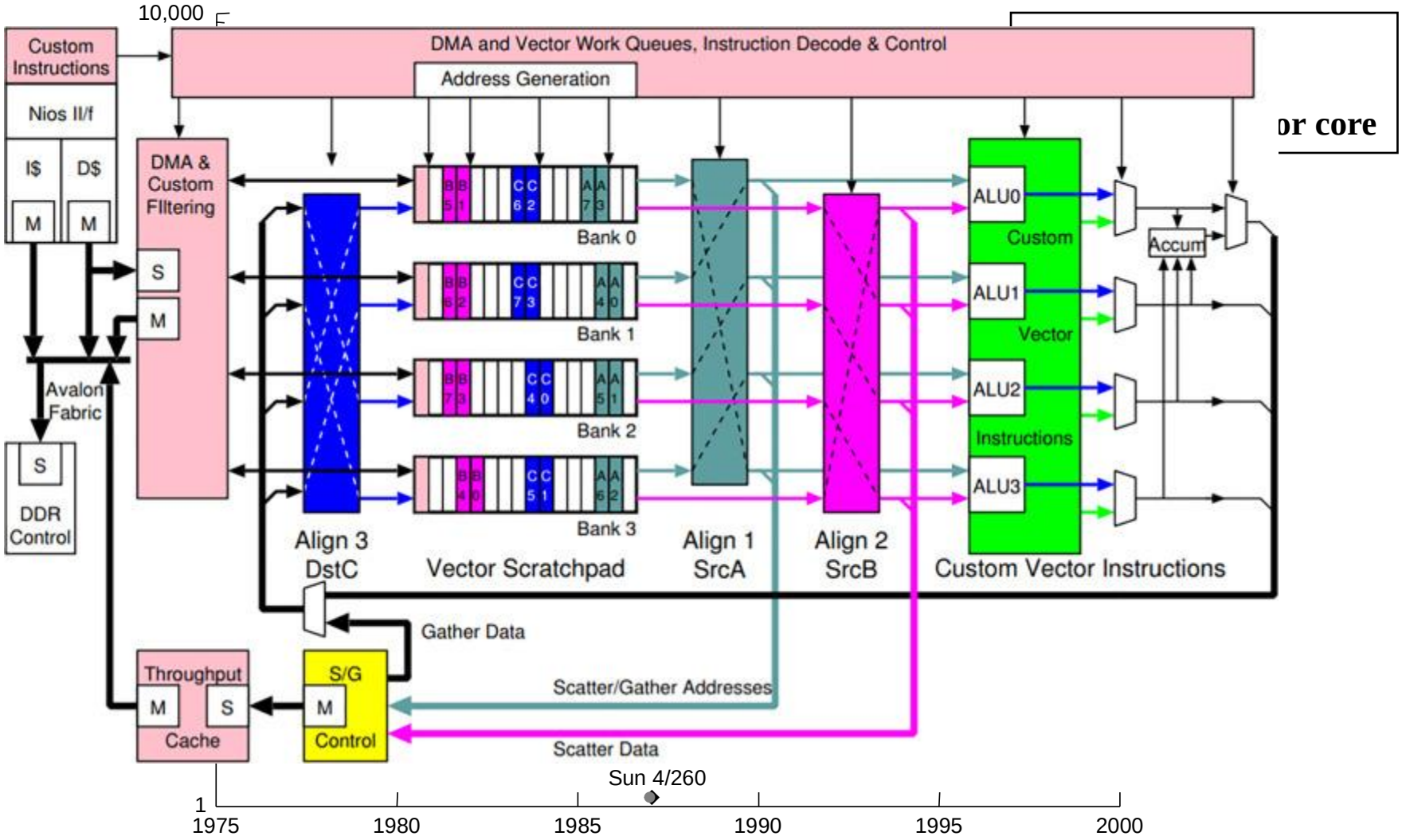
Example 100-Core CMP: Tilera TILE-Gx Processor



For more information see: http://www.tilera.com/

Microprocessors Vs. Vector Processors

Uniprocessor Performance: LINPACK

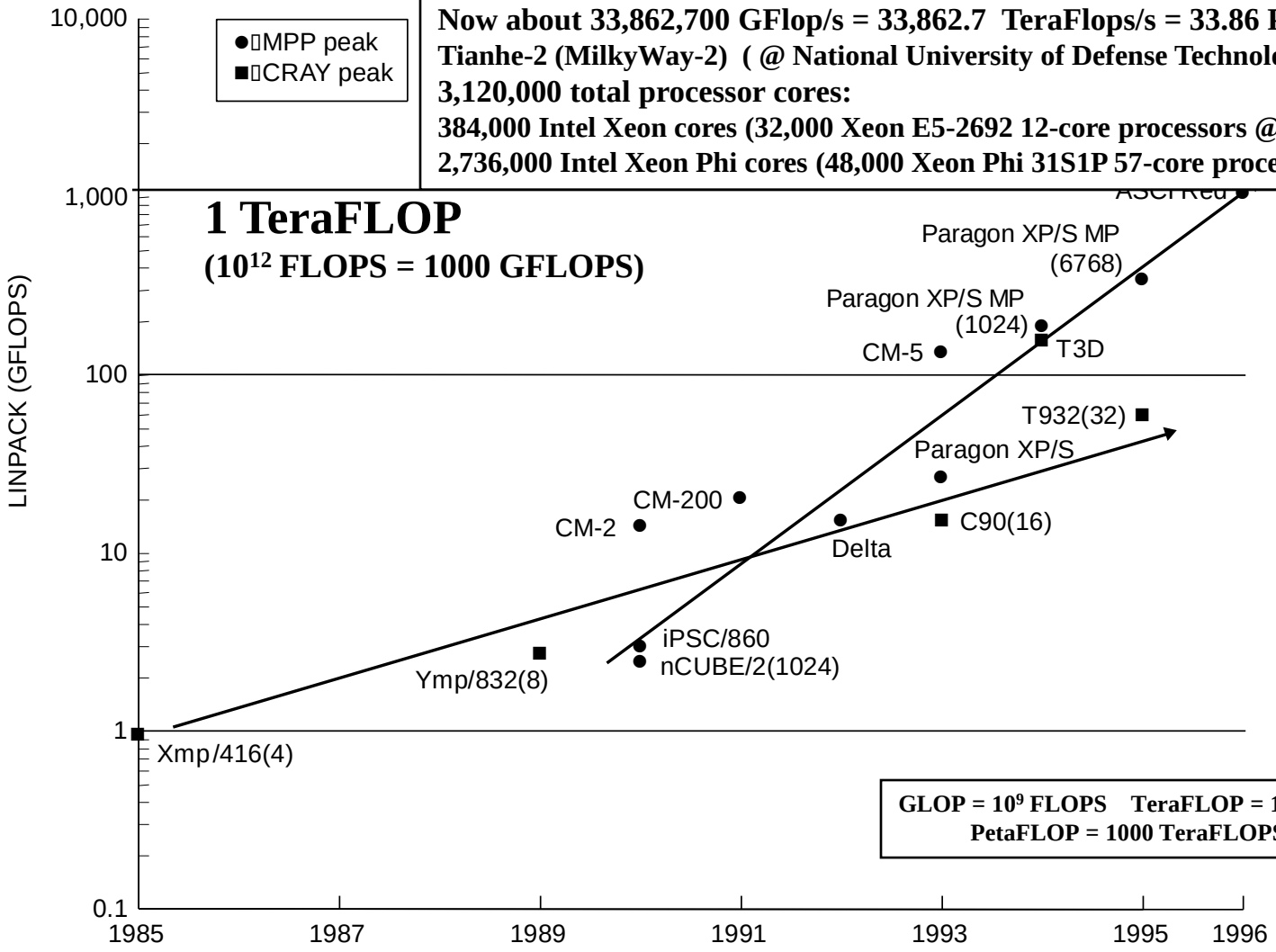


Parallel Performance: LINPACK

Since ~ June 2013

Current Top LINPACK Performance:

Now about 33,862,700 GFlop/s = 33,862.7 TeraFlops/s = 33.86 PetaFlops/s
Tianhe-2 (MilkyWay-2) (@ National University of Defense Technology, Changsha, China)
3,120,000 total processor cores:
384,000 Intel Xeon cores (32,000 Xeon E5-2692 12-core processors @ 2.2 GHz) +
2,736,000 Intel Xeon Phi cores (48,000 Xeon Phi 31S1P 57-core processors @ 1.1 GHz)



GLOP = 10^9 FLOPS TeraFLOP = 1000 GFLOPS = 10^{12} FLOPS
PetaFLOP = 1000 TeraFLOPS = 10^{15} FLOPS

Current ranking of top 500 parallel supercomputers in the world is found at: www.top500.org

26th List: The TOP10

November 2005

	Manufacturer	Computer	Rmax [TF/s]	Installation Site	Country	Year	#Proc
1	IBM	BlueGene/L eServer Blue Gene	280.6	DOE/NNSA/LLNL	USA	2005	131072
2	IBM	BGW eServer Blue Gene	91.29	IBM Thomas Watson	USA	2005	40960
3	IBM	ASC Purple eServer pSeries p575	63.39	DOE/NNSA/LLNL	USA	2005	10240
4	SGI	Columbia Altix, Infiniband	51.87	NASA Ames	USA	2004	10160
5	Dell	Thunderbird	38.27	Sandia	USA	2005	8000
6	Cray	Red Storm Cray XT3	36.19	Sandia	USA	2005	10880
7	NEC	Earth-Simulator	35.86	Earth Simulator Center	Japan	2002	5120
8	IBM	MareNostrum BladeCenter JS20, Myrinet	27.91	Barcelona Supercomputer Center	Spain	2005	4800
9	IBM	eServer Blue Gene	27.45	ASTRON University Groningen	Netherlands	2005	12288
10	Cray	Jaguar Cray XT3	20.53	Oak Ridge National Lab	USA	2005	5200

Source (and for current list): www.top500.org

TOP500 Supercomputers

32nd List (November 2008): **The Top 10**

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	DOE/NNSA/LANL United States	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz , Voltaire Infiniband / 2008 IBM	129600	1105.00	1456.70	2483.47
2	Oak Ridge National Laboratory United States	Jaguar - Cray XT5 QC 2.3 GHz / 2008 Cray Inc.	150152	1059.00	1381.40	6950.60
3	NASA/Ames Research Center/NAS United States	Pleiades - SGI Altix ICE 8200EX, Xeon QC 3.0/2.66 GHz / 2008 SGI	51200	487.01	608.83	2090.00
4	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution / 2007 IBM	212992	478.20	596.38	2329.60
5	Argonne National Laboratory United States	Blue Gene/P Solution / 2007 IBM	163840	450.30	557.06	1260.00
6	Texas Advanced Computing Center/Univ. of Texas United States	Ranger - SunBlade x6420, Opteron QC 2.3 Ghz, Infiniband / 2008 Sun Microsystems	62976	433.20	579.38	2000.00
7	NERSC/LBNL United States	Franklin - Cray XT4 QuadCore 2.3 GHz / 2008 Cray Inc.	38642	266.30	355.51	1150.00
8	Oak Ridge National Laboratory United States	Jaguar - Cray XT4 QuadCore 2.1 GHz / 2008 Cray Inc.	30976	205.00	260.20	1580.71
9	NNSA/Sandia National Laboratories United States	Red Storm - Sandia/ Cray Red Storm, XT3/4, 2.4/2.2 GHz dual/quad core / 2008 Cray Inc.	38208	204.20	284.00	2506.00
10	Shanghai Supercomputer Center China	Dawning 5000A - Dawning 5000A, QC Opteron 1.9 Ghz, Infiniband, Windows HPC 2008 / 2008 Dawning	30720	180.60	233.47	

KW

Source (and for current list): www.top500.org

TOP500 Supercomputers

34th List (November 2009): **The Top 10**

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	Oak Ridge National Laboratory United States	Jaguar - Cray XT5-HE Opteron Six Core 2.6 GHz / 2009 Cray Inc.	224162	1759.00	2331.00	6950.60
2	DOE/NNSA/LANL United States	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband / 2009 IBM	122400	1042.00	1375.78	2345.50
3	National Institute for Computational Sciences/University of Tennessee United States	Kraken XT5 - Cray XT5-HE Opteron Six Core 2.6 GHz / 2009 Cray Inc.	98928	831.70	1028.85	
4	Forschungszentrum Juelich (FZJ) Germany	JUGENE - Blue Gene/P Solution / 2009 IBM	294912	825.50	1002.70	2268.00
5	National SuperComputer Center in Tianjin/NUDT China	Tianhe-1 - NUDT TH-1 Cluster, Xeon E5540/E5450, ATI Radeon HD 4870 2, Infiniband / 2009 NUDT	71680	563.10	1206.19	
6	NASA/Ames Research Center/NAS United States	Pleiades - SGI Altix ICE 8200EX, Xeon QC 3.0 GHz/Nehalem EP 2.93 Ghz / 2009 SGI	56320	544.30	673.26	2348.00
7	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution / 2007 IBM	212992	478.20	596.38	2329.60
8	Argonne National Laboratory United States	Blue Gene/P Solution / 2007 IBM	163840	458.61	557.06	1260.00
9	Texas Advanced Computing Center/Univ. of Texas United States	Ranger - SunBlade x6420, Opteron QC 2.3 Ghz, Infiniband / 2008 Sun Microsystems	62976	433.20	579.38	2000.00
10	Sandia National Laboratories / National Renewable Energy Laboratory United States	Red Sky - Sun Blade x6275, Xeon X55xx 2.93 Ghz, Infiniband / 2009 Sun Microsystems	41616	423.90	487.74	

KW

Source (and for current list): www.top500.org

TOP500 Supercomputers

36th List (November 2010): **The Top 10**

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	National Supercomputing Center in Tianjin China	Tianhe-1A - NUDT TH MPP, X5670 2.93Ghz 6C, NVIDIA GPU, FT-1000 8C / 2010 NUDT	186368	2566.00	4701.00	4040.00
2	DOE/SC/Oak Ridge National Laboratory United States	Jaguar - Cray XT5-HE Opteron 6-core 2.6 GHz / 2009 Cray Inc.	224162	1759.00	2331.00	6950.60
3	National Supercomputing Centre in Shenzhen (NSCS) China	Nebulae - Dawning TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU / 2010 Dawning	120640	1271.00	2984.30	2580.00
4	GSIC Center, Tokyo Institute of Technology Japan	TSUBAME 2.0 - HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows / 2010 NEC/HP	73278	1192.00	2287.63	1398.61
5	DOE/SC/LBNL/NERSC United States	Hopper - Cray XE6 12-core 2.1 GHz / 2010 Cray Inc.	153408	1054.00	1288.63	2910.00
6	Commissariat a l'Energie Atomique (CEA) France	Tera-100 - Bull bullx super-node S6010/S6030 / 2010 Bull SA	138368	1050.00	1254.55	4590.00
7	DOE/NNSA/LANL United States	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband / 2009 IBM	122400	1042.00	1375.78	2345.50
8	National Institute for Computational Sciences/University of Tennessee United States	Kraken XT5 - Cray XT5-HE Opteron 6-core 2.6 GHz / 2009 Cray Inc.	98928	831.70	1028.85	3090.00
9	Forschungszentrum Juelich (FZJ) Germany	JUGENE - Blue Gene/P Solution / 2009 IBM	294912	825.50	1002.70	2268.00
10	DOE/NNSA/LANL/SNL United States	Cielo - Cray XE6 8-core 2.4 GHz / 2010 Cray Inc.	107152	816.60	1028.66	2950.00

KW

Source (and for current list): www.top500.org

TOP500 Supercomputers

38th List (November 2011): **The Top 10**

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect / 2011 Fujitsu	705024	10510.00	11280.38	12659.9
2	National Supercomputing Center in Tianjin China	NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 / 2010 NUDT	186368	2566.00	4701.00	4040.0
3	DOE/SC/Oak Ridge National Laboratory United States	Cray XT5-HE Opteron 6-core 2.6 GHz / 2009 Cray Inc.	224162	1759.00	2331.00	6950.0
4	National Supercomputing Centre in Shenzhen (NSCS) China	Dawning TC3600 Blade System, Xeon X5650 6C 2.66GHz, Infiniband QDR, NVIDIA 2050 / 2010 Dawning	120640	1271.00	2984.30	2580.0
5	GSIC Center, Tokyo Institute of Technology Japan	HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows / 2010 NEC/HP	73278	1192.00	2287.63	1398.6
6	DOE/NNSA/LANL/SNL United States	Cray XE6, Opteron 6136 8C 2.40GHz, Custom / 2011 Cray Inc.	142272	1110.00	1365.81	3980.0
7	NASA/Ames Research Center/NAS United States	SGI Altix ICE 8200EX/8400EX, Xeon HT QC 3.0/Xeon 5570/5670 2.93 Ghz, Infiniband / 2011 SGI	111104	1088.00	1315.33	4102.0
8	DOE/SC/LBNL/NERSC United States	Cray XE6, Opteron 6172 12C 2.10GHz, Custom / 2010 Cray Inc.	153408	1054.00	1288.63	2910.0
9	Commissariat a l'Energie Atomique (CEA) France	Bull bullx super-node S6010/S6030 / 2010 Bull	138368	1050.00	1254.55	4590.0
10	DOE/NNSA/LANL United States	BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband / 2009 IBM	122400	1042.00	1375.78	2345.0

KW

Source (and for current list): www.top500.org

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
2	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	16324.8	20132.7	7890
3	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510.0	11280.4	12660
4	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786432	8162.4	10066.3	3945
5	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393216	4141.2	5033.2	1970
6	Leibniz Rechenzentrum Germany	**SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR IBM	147456	2897.0	3185.1	3423
7	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi Dell	204900	2660.3	3959.0	
8	National Supercomputing Center in Tianjin China	**Tianhe-1A** - NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 NUDT	186368	2566.0	4701.0	4040
9	CINECA Italy	**Fermi** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	163840	1725.5	2097.2	822
10	IBM Development Engineering United States	**DARPA Trial Subset** - Power 775, POWER7 8C 3.836GHz, Custom Interconnect IBM	63360	1515.0	1944.4	3576

KW

TOP500 Supercomputers

40th List (November 2012) The Top 10

Cray XK7 - Titan
(@ Oak Ridge National Laboratory)

LINPACK Performance:
17.59 PetaFlops/s (quadrillion Flops per second)

Peak Performance: 27.1 PetaFlops/s

560,640 total processor cores:

299,008 Opteron cores
(18,688 AMD Opteron 6274 16-core processors @ 2.2 GHz)
+
261,632 GPU cores
(18,688 Nvidia Tesla Kepler K20x GPUs @ 0.7 GHz)

Source (and for current list): www.top500.org

TOP500 Supercomputers

41st List (June 2013): **The Top 10**

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National University of Defense Technology China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3120000	33862.7	54902.4	17808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	17173.2	20132.7	7890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510.0	11280.4	12660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786432	8586.6	10066.3	3945
6	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462462	5168.1	8520.1	4510
7	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458752	5008.9	5872.0	2301
8	DOE/NNSA/LLNL United States	**Vulcan** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393216	4293.3	5033.2	1972
9	Leibniz Rechenzentrum Germany	**SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR IBM	147456	2897.0	3185.1	3423
10	National Supercomputing Center in Tianjin China	**Tianhe-1A** - NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 NUDT	186368	2566.0	4701.0	4040

KW

Source (and for current list): www.top500.org

TOP500 Supercomputers

42nd List (Nov. 2013): **The Top 10**

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3120000	33862.7	54902.4	17808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	17173.2	20132.7	7890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510.0	11280.4	12660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786432	8586.6	10066.3	3945
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115984	6271.0	7788.9	2325
7	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462462	5168.1	8520.1	4510
8	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458752	5008.9	5872.0	2301
9	DOE/NNSA/LLNL United States	**Vulcan** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393216	4293.3	5033.2	1972
10	Leibniz Rechenzentrum Germany	**SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR IBM	147456	2897.0	3185.1	3423

KW

Source (and for current list): www.top500.org

TOP500 Supercomputers

43rd List (June 2014): **The Top 10**

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3120000	33862.7	54902.4	17808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	17173.2	20132.7	7890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510.0	11280.4	12660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786432	8586.6	10066.3	3945
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115984	6271.0	7788.9	2325
7	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462462	5168.1	8520.1	4510
8	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458752	5008.9	5872.0	2301
9	DOE/NNSA/LLNL United States	**Vulcan** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393216	4293.3	5033.2	1972
10	Leibniz Rechenzentrum Germany	**SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR IBM	147456	2897.0	3185.1	3423

KW

Source (and for current list): www.top500.org

The Goal of Parallel Processing

- **Goal of applications in using parallel machines:**

  Maximize Speedup over single processor performance

Parallel

$$\textit{Speedup (p processors)} = \frac{\textit{Performance (p processors)}}{\textit{Performance (1 processor)}}$$

- **For a fixed problem size (input data set), performance = 1/time**

Fixed Problem Size Parallel Speedup

Parallel Speedup, Speedup_p

$$\textit{Speedup}_{\textit{fixed problem}}\textit{ (p processors)} = \frac{\textit{Time (1 processor)}}{\textit{Time (p processors)}}$$

- **Ideal speedup = number of processors = p**

Very hard to achieve

+ load imbalance

Due to parallelization overheads: communication cost, dependencies ...

The Goal of Parallel Processing

- **Parallel processing goal is to maximize parallel speedup:**

Fixed Problem Size Parallel Speedup

$$\text{Speedup} = \frac{\text{Time(1)}}{\text{Time(p)}} \leq \frac{\text{Sequential Work on one processor}}{\textit{Max}\ (\text{Work + Synch Wait Time + Comm Cost + Extra Work})}$$

Or time

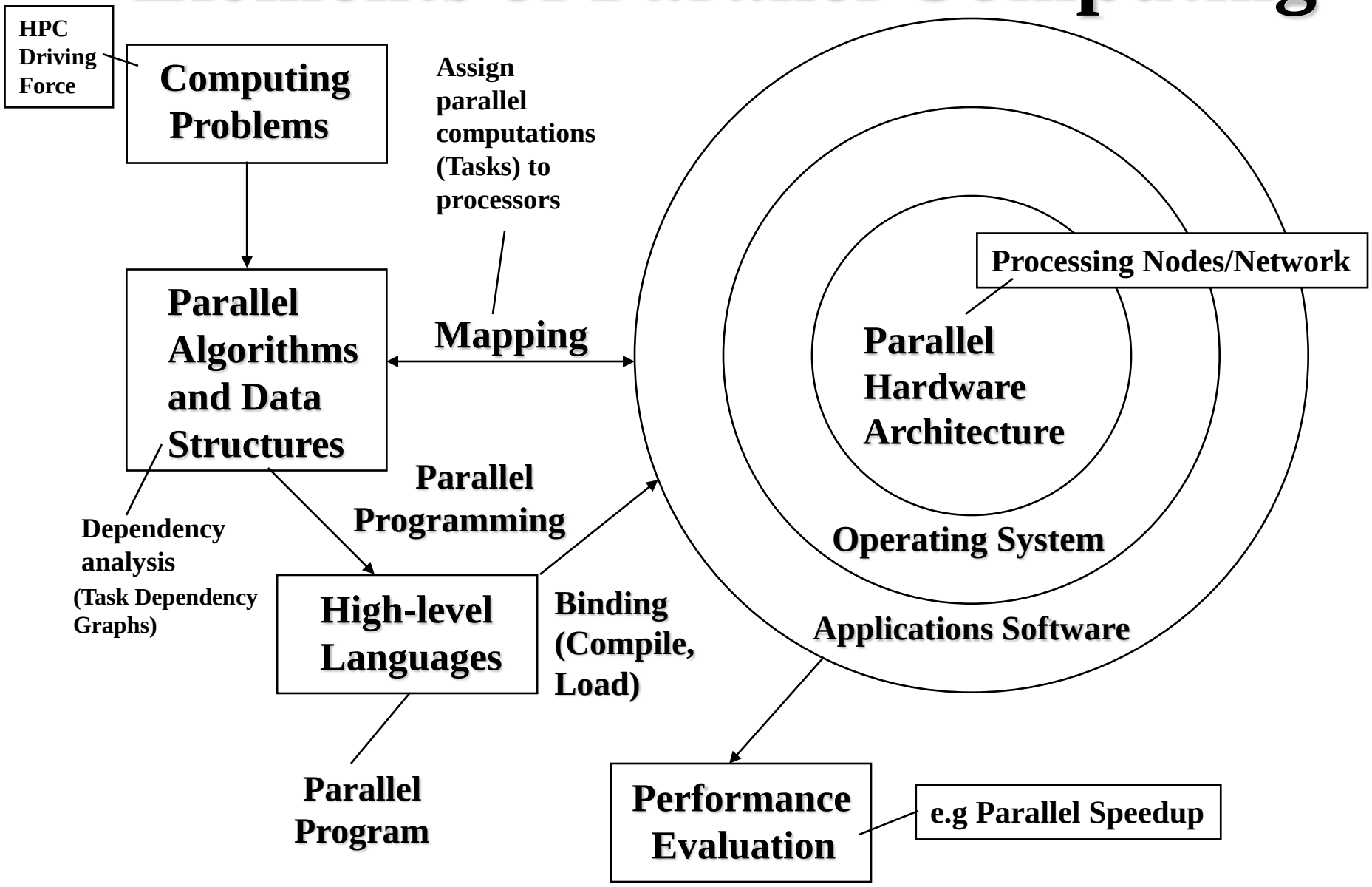
Time ← **Parallelization overheads** →

i.e the processor with maximum execution time

- **Ideal Speedup = p = number of processors** **Implies or Requires**
  - **Very hard to achieve: Implies no parallelization overheads and perfect load balance among all processors.**
- **<u>Maximize parallel speedup by:</u>**
  - 1 – **<u>Balancing computations on processors</u> (every processor does the same amount of work) and the same amount of overheads.**
  - 2 – **<u>Minimizing communication cost and other overheads</u> associated with each step of parallel program creation and execution.**
- **<u>Performance Scalability:</u>**

+ **Achieve a good speedup for the parallel application on the parallel architecture as problem size and machine size (number of processors) are increased.**

Elements of Parallel Computing



Elements of Parallel Computing

1 **Computing Problems:**

Driving Force

- **Numerical Computing: Science and and engineering numerical problems demand intensive integer and floating point computations.**
- **Logical Reasoning: Artificial intelligence (AI) demand logic inferences and symbolic manipulations and large space searches.**

2 **Parallel Algorithms and Data Structures**

- **Special algorithms and data structures are needed to specify the computations and communication present in computing problems (from dependency analysis).**
- **Most numerical algorithms are deterministic using regular data structures.**
- **Symbolic processing may use heuristics or non-deterministic searches.**
- **Parallel algorithm development requires interdisciplinary interaction.**

Elements of Parallel Computing

3 **Hardware Resources** **Parallel Architecture** **Computing power**

A – **Processors, memory, and peripheral devices (processing nodes) form the hardware core of a computer system.**

B – **Processor connectivity (system interconnects, network), memory organization, influence the system architecture.**

Communication/connectivity

4 **Operating Systems**

– **Manages the allocation of resources to running processes.**

– **Mapping to match algorithmic structures with hardware architecture and vice versa: processor scheduling, memory mapping, interprocessor communication.**

- **Parallelism exploitation possible at: 1- algorithm design, 2- program writing, 3- compilation, and 4- run time.**

Elements of Parallel Computing

5 **System Software Support**

- **Needed for the development of efficient programs in high-level languages (HLLs.)**
- **Assemblers, loaders.**
- **Portable parallel programming languages/libraries**
- **User interfaces and tools.**

6 **Compiler Support** *Approaches to parallel programming*

(a) – Implicit Parallelism Approach

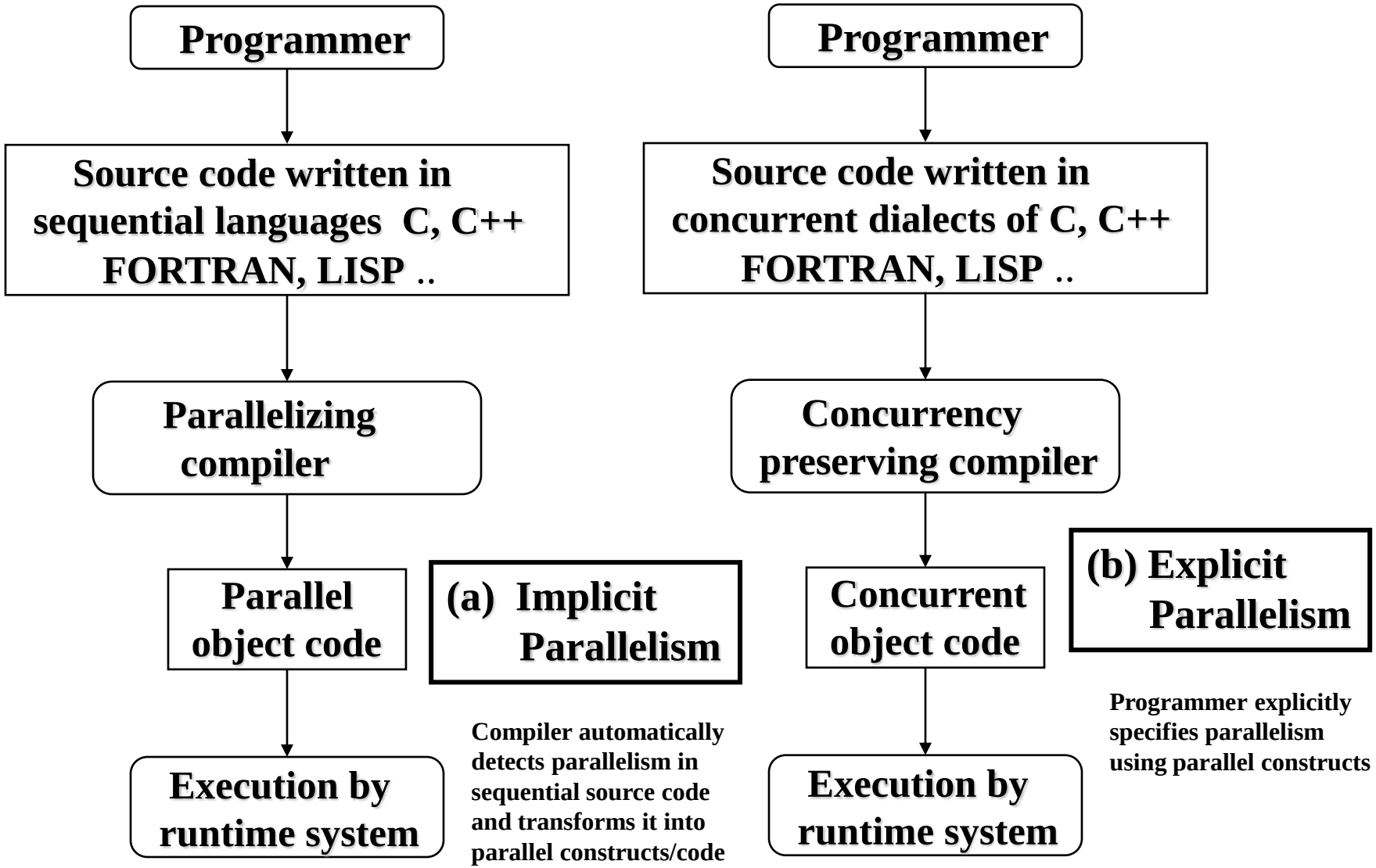
- **Parallelizing compiler: Can automatically detect parallelism in sequential source code and transforms it into parallel constructs/code.**
- **Source code written in conventional sequential languages**

(b) – Explicit Parallelism Approach:

- **Programmer explicitly specifies parallelism using:**
  - **Sequential compiler (conventional sequential HLL) and low-level library of the target parallel computer , or ..**
  - **Concurrent (parallel) HLL .**
- **Concurrency Preserving Compiler: The compiler in this case preserves the parallelism explicitly specified by the programmer. It may perform some program flow analysis, dependence checking, limited optimizations for parallelism detection.**

Illustrated next →

Approaches to Parallel Programming



Factors Affecting Parallel System Performance

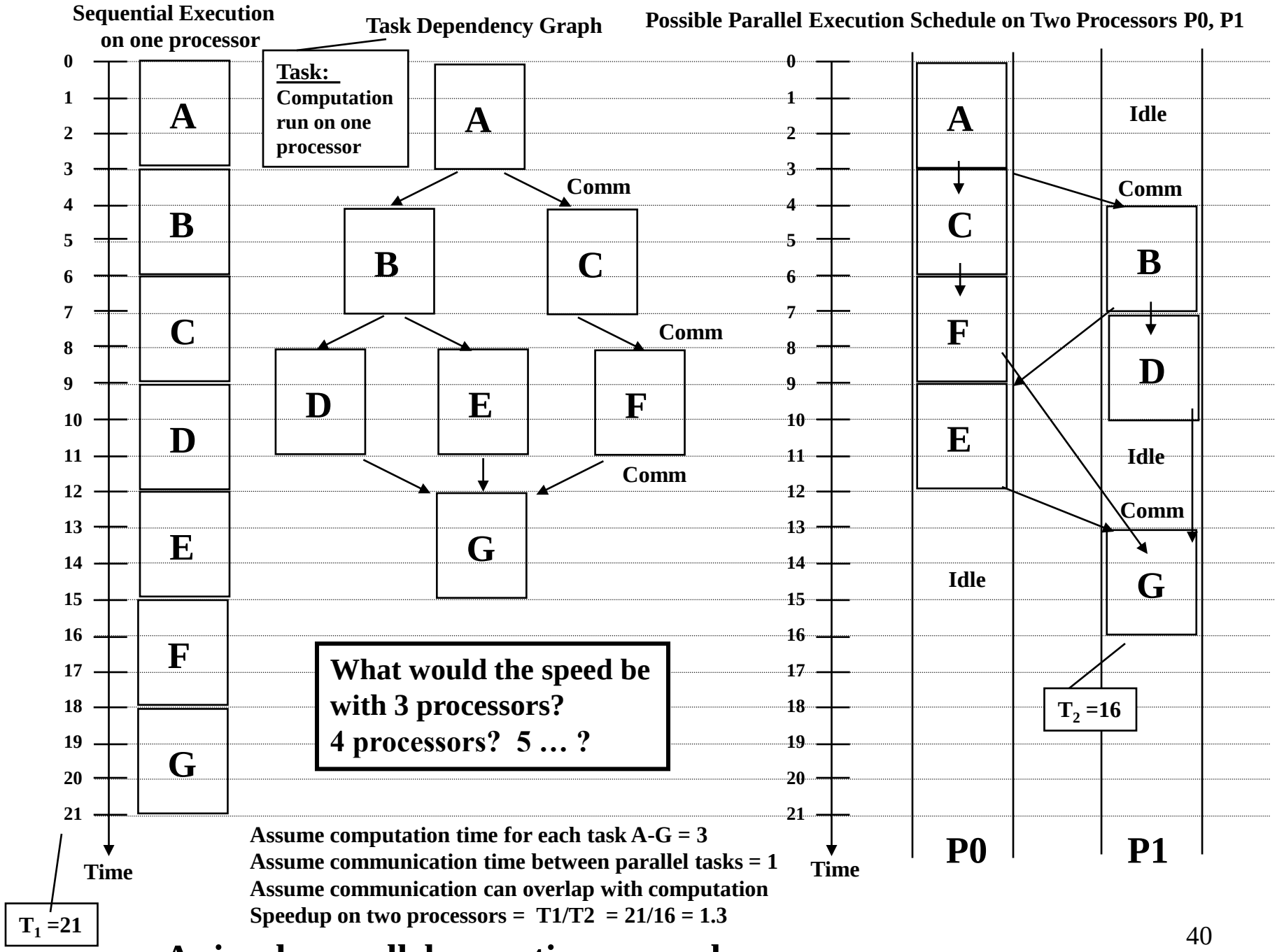
- **Parallel Algorithm Related:**
  - Available concurrency and profile, grain size, uniformity, patterns.
    - Dependencies between computations represented by dependency graph
  - Type of parallelism present: Functional and/or data parallelism.
  - Required communication/synchronization, uniformity and patterns.
  - Data size requirements.
  - Communication to computation ratio (C-to-C ratio, lower is better).

i.e Inherent Parallelism

- **Parallel program Related:**
  - Programming model used.
  - Resulting data/code memory requirements, locality and working set characteristics.
  - Parallel task grain size.
  - Assignment (mapping) of tasks to processors: Dynamic or static.
  - Cost of communication/synchronization primitives.
- **Hardware/Architecture related:**
  - Total CPU computational power available.
  - Types of computation modes supported.
  - Shared address space Vs. message passing.
  - Communication network characteristics (topology, bandwidth, latency)
  - Memory hierarchy properties.

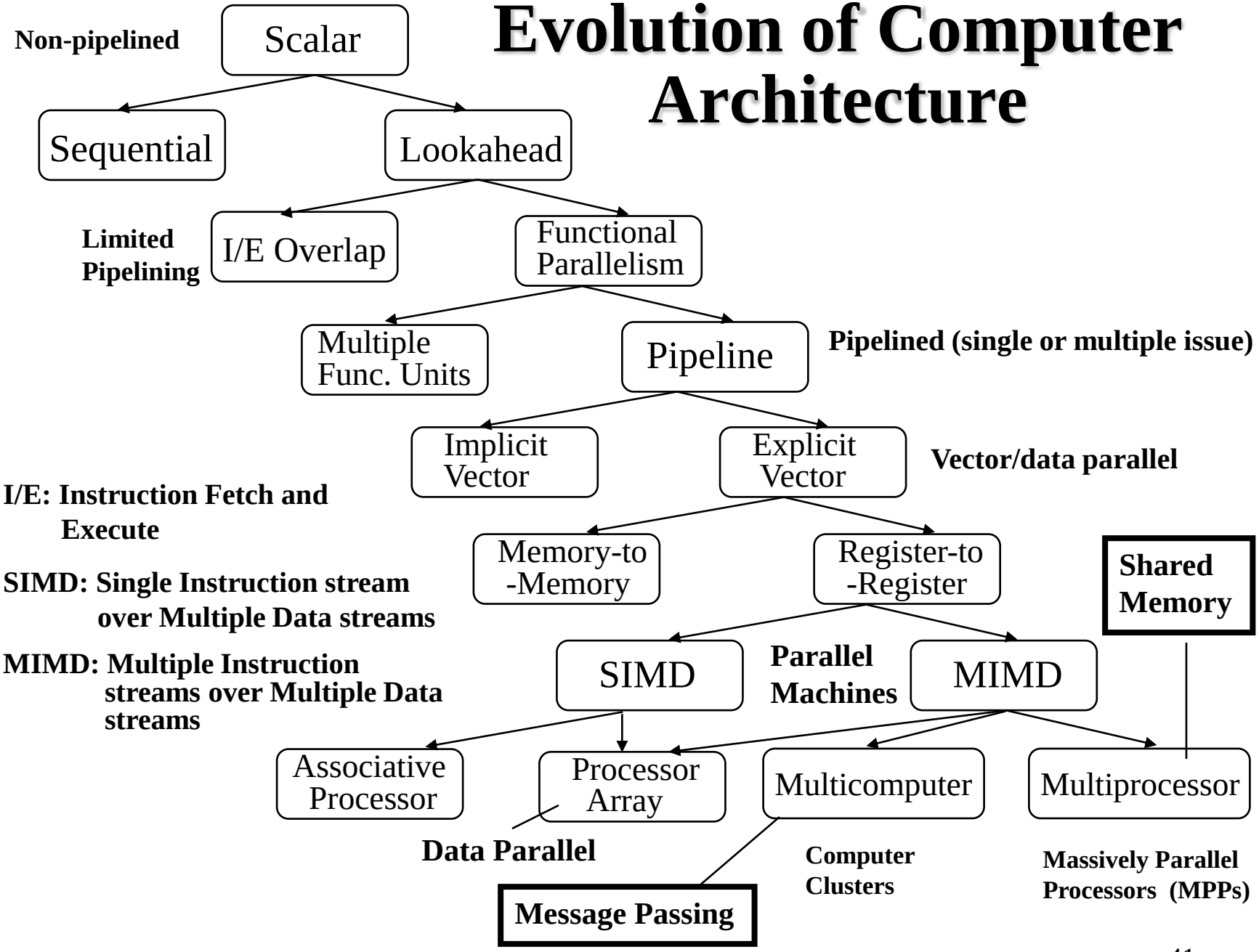
+ Number of processors (hardware parallelism)

Concurrency = Parallelism



A simple parallel execution example

Evolution of Computer Architecture



Parallel Programming Models

- **Programming methodology used in coding parallel applications**
- **Specifies: 1- communication and 2- synchronization**
- **Examples:**

However, a good way to utilize multi-core processors for the masses!

  - ***Multiprogramming:*** ***or Multi-tasking (not true parallel processing!)***

    No communication or synchronization at program level. A number of independent programs running on different processors in the system.

  - ***Shared memory address space (SAS):***

    Parallel program threads or tasks communicate implicitly using a shared memory address space (shared data in memory).

  - ***Message passing*:**

    Explicit point to point communication (via send/receive pairs) is used between parallel program tasks using messages.

  - ***Data parallel*:**

    More regimented, global actions on data (i.e the same operations over all elements on an array or vector)

    - **Can be implemented with shared address space or message passing.**

Flynn's 1972 Classification of Computer Architecture
(Taxonomy)

Instruction Stream = Thread of Control or Hardware Context

(a) **Single Instruction stream over a Single Data stream (SISD): Conventional sequential machines or uniprocessors.**

(b) **Single Instruction stream over Multiple Data streams (SIMD): Vector computers, array of synchronized processing elements.**

Data parallel systems (GPUs?)

(c) **Multiple Instruction streams and a Single Data stream (MISD): Systolic arrays for pipelined execution.**

(d) **Multiple Instruction streams over Multiple Data streams (MIMD): Parallel computers:**

Tightly coupled processors

- **Shared memory multiprocessors.**

Loosely coupled processors

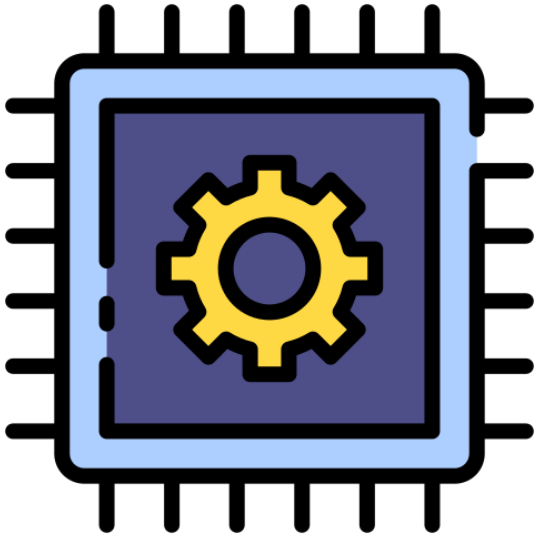
- **Multicomputers: Unshared distributed memory, message-passing used instead (e.g clusters)**

Classified according to number of instruction streams (threads) and number of data streams in architecture

Hardware architectures for parallel processing

Classification based on the number of instruction and data streams:

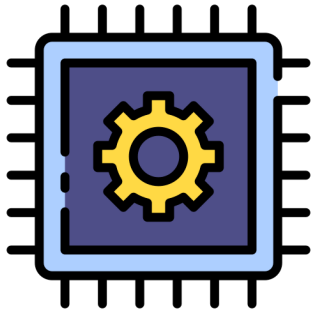
1. **Single-instruction, single-data (SISD) stream systems**
2. **Single-instruction, multiple-data (SIMD) stream systems**
3. **Multiple-instruction, single-data (MISD) stream systems**
4. **Multiple-instruction, multiple-data (MIMD) stream systems**



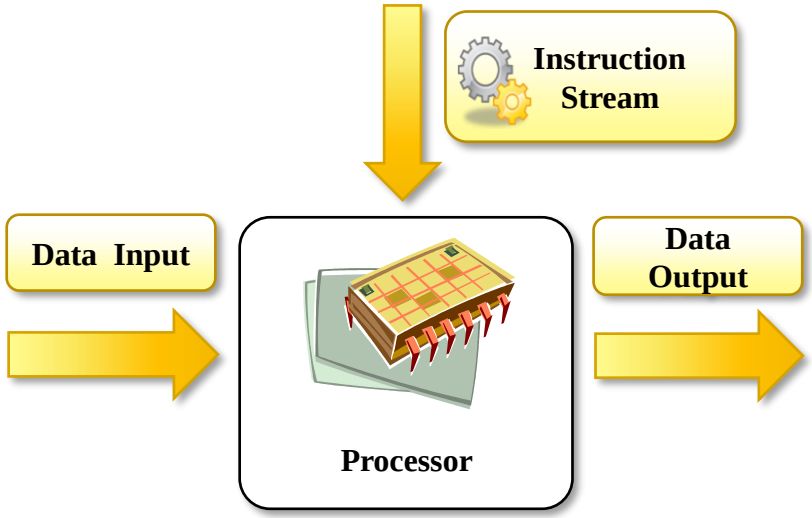
Hardware architectures for parallel processing

Single-instruction, single-data (SISD) systems :

An SISD computing system is a uniprocessor machine capable of executing a single instruction, which operates on a single data stream.

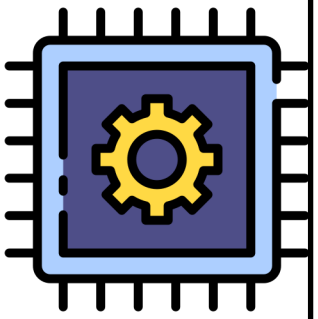


Dominant representative SISD systems are IBM PC, Macintosh, and workstations.



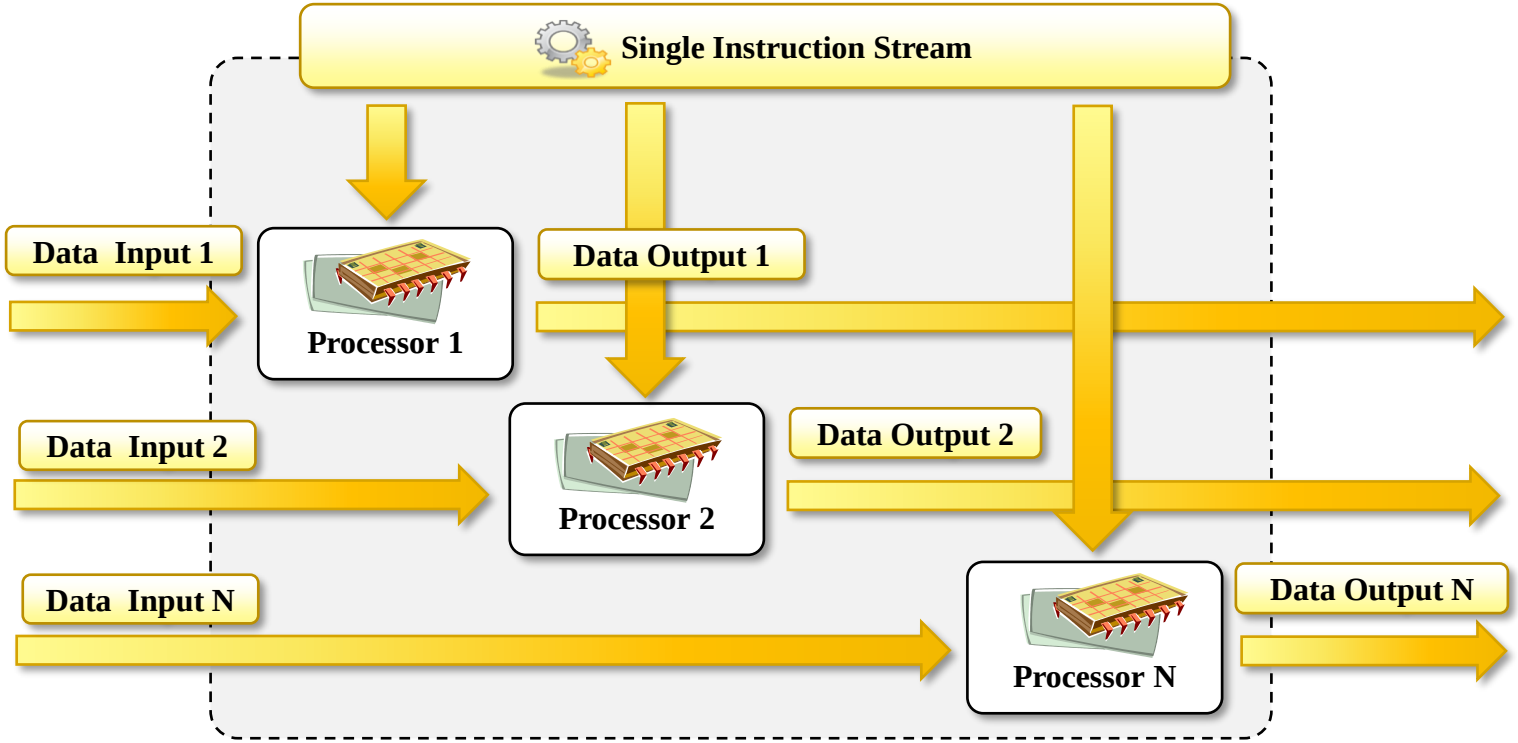
Hardware architectures for parallel processing

Single-instruction, multiple-data (SIMD) systems :



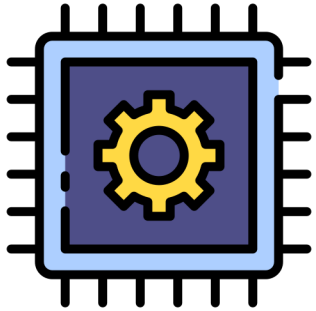
SIMD is a multiprocessor machine capable of executing the same instruction on all the CPUs but operating on different data streams.
SIMD models are well suited to scientific computing since they involve lots of vector and matrix operations.
Ex. Ci=Ai*Bi;
Dominant representative SIMD system is Cray's vector processing machine

Hardware architectures for parallel processing



Hardware architectures for parallel processing

Multiple-instruction, single-data (MISD) systems :

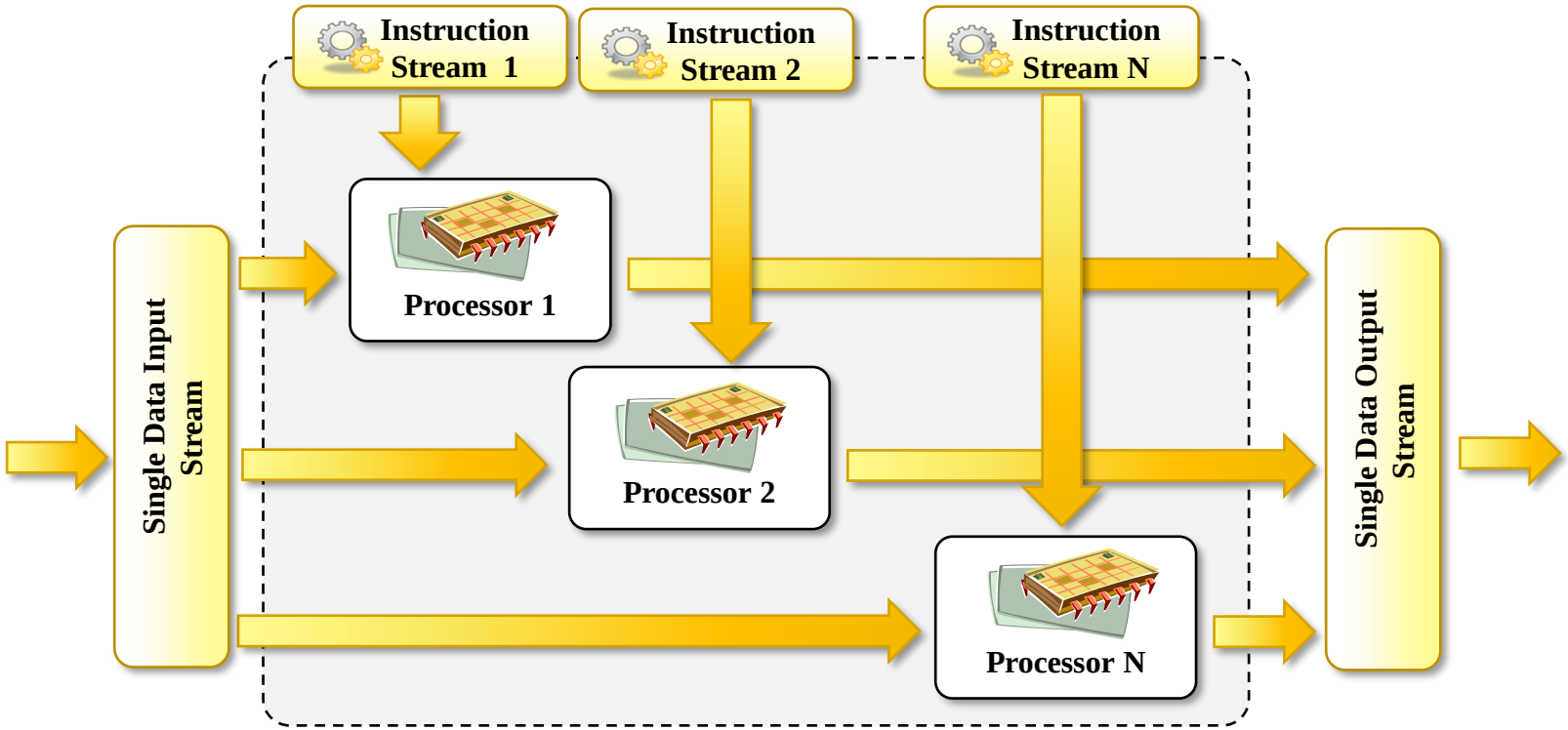


An MISD is a multiprocessor machine capable of executing different instructions on different PEs but all of them operating on the same data set.

Ex: Y=sin(x)+cos(x)+tan(x)

Machines built using the MISD model are not useful in most of the applications; a few machines are built, but none of them are available commercially

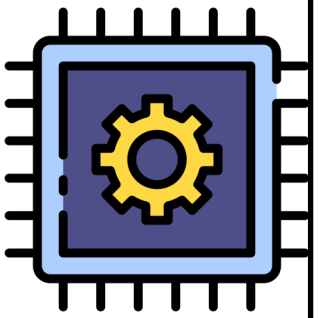
Hardware architectures for parallel processing



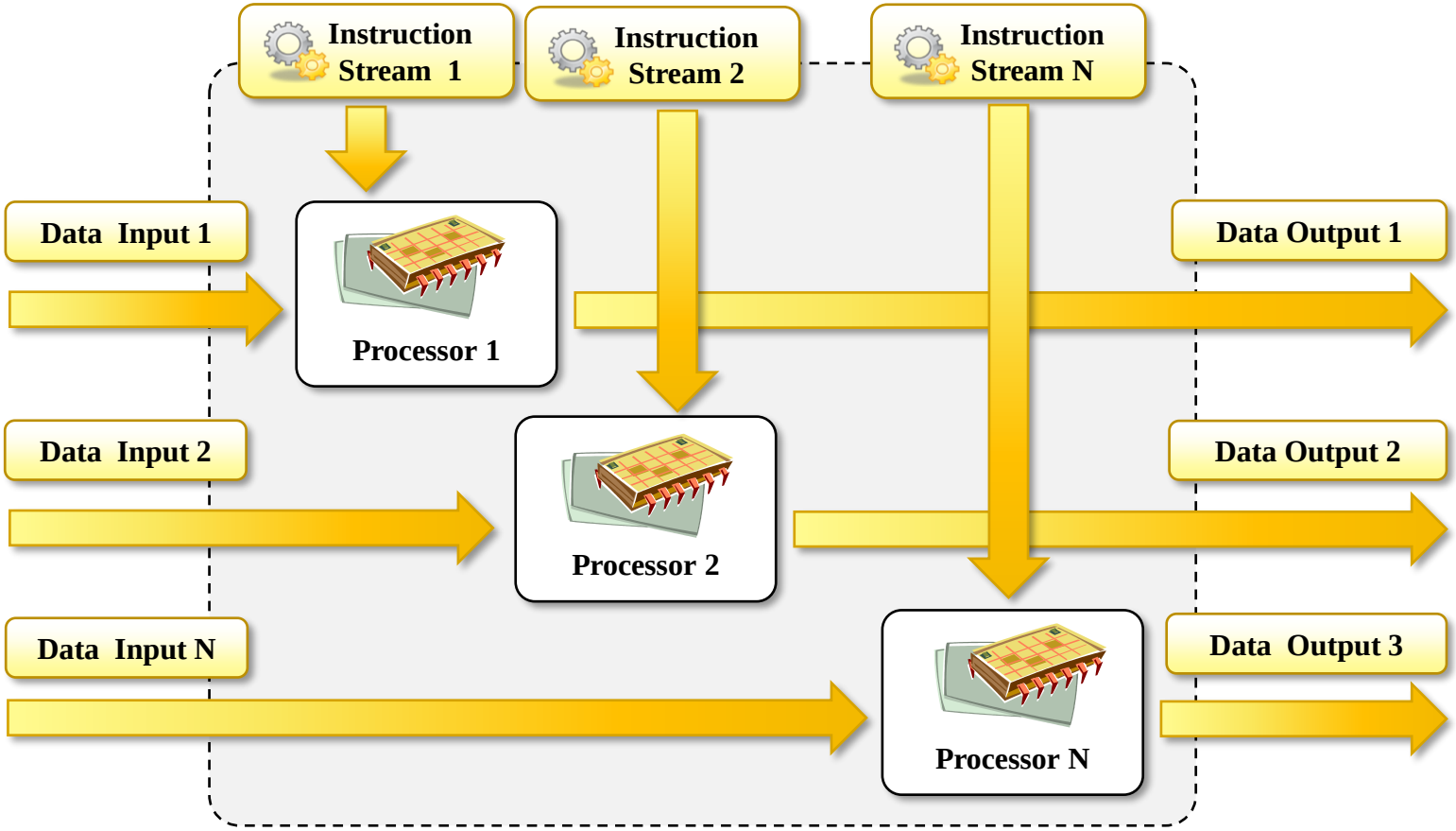
Hardware architectures for parallel processing

Multiple-instruction, Multiple-data (MIMD) systems :

An MIMD is a multiprocessor machine capable of executing multiple instructions on multiple data sets.
Each PE in the MIMD model has separate instruction and data streams;
well suited to any kind of application.
Unlike SIMD and MISD machines, PEs in MIMD machines work asynchronously.



Hardware architectures for parallel processing



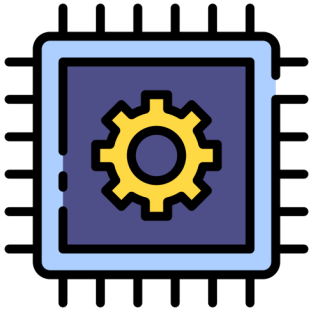
MIMD machines are broadly categorized into shared-memory MIMD and distributed-memory MIMD.

Hardware architectures for parallel processing

Shared-memory MIMD:

- **All the PEs are connected to a single global memory and they all have access to it, referred as tightly coupled multiprocessor systems.**
- **Communication among PEs through the shared memory.**
- **Modification of the data stored in the global memory by one PE is visible to all other PEs.**

Ex: Silicon Graphics machines and Sun/IBM's SMP, API: OpenMP

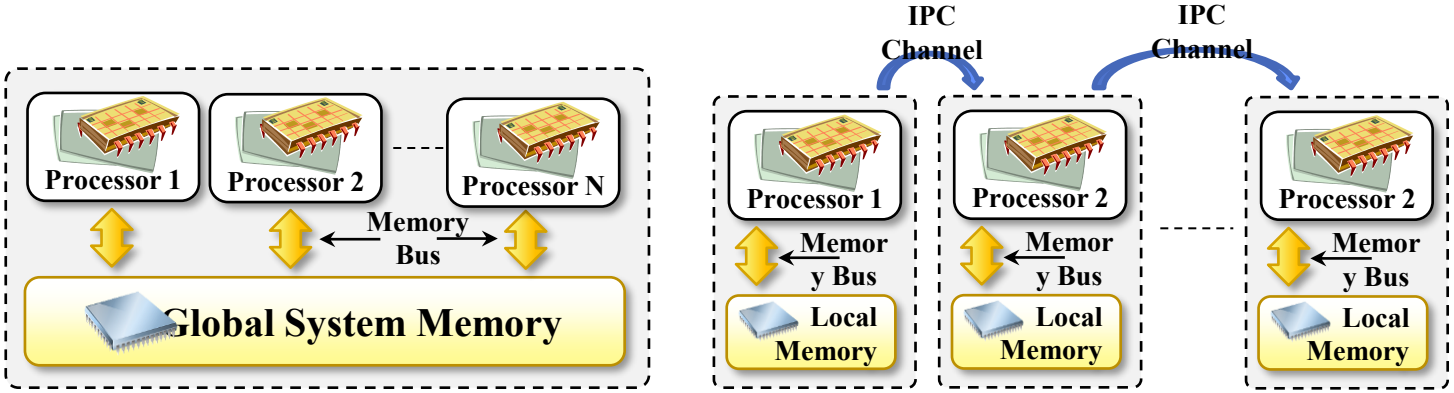


Distributed-memory MIMD:

- **All PEs have a local memory, referred as loosely coupled multiprocessor systems, communication between PEs through the interconnection network.**
- **PEs can be configured via to tree, mesh, cube topology and so on.**
- **Each PE operates asynchronously, and if communication/synchronization among tasks is necessary, they can do so by exchanging messages between them. Programming API:MPI.**

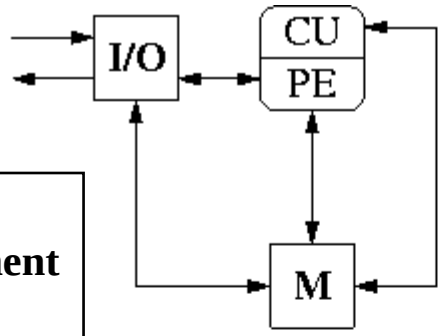
Hardware architectures for parallel processing

- The shared-memory MIMD architecture is easier to program but is less tolerant to failures and harder to extend as in distributed memory MIMD model.
- Failures in a shared-memory MIMD affect the entire system, whereas this is not the case of the distributed model.
- Shared memory MIMD architectures are less likely to scale because the addition of more PEs leads to memory contention.
- This is a situation that does not happen in the case of distributed memory, in which each PE has its own memory.
- As a result, distributed memory MIMD architectures are most popular today.



Flynn's Classification of Computer Architecture
(Taxonomy)

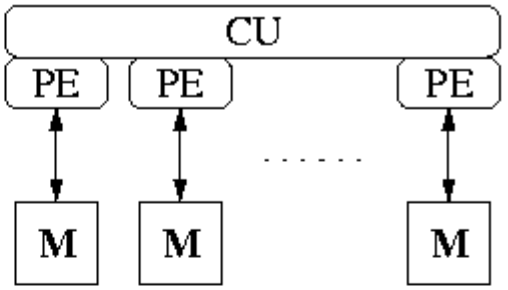
Uniprocessor



(a) SISD

CU = Control Unit
PE = Processing Element
M = Memory

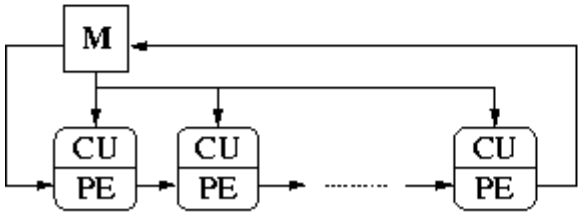
Single Instruction stream over Multiple Data streams (SIMD): Vector computers, array of synchronized processing elements.



(b) SIMD

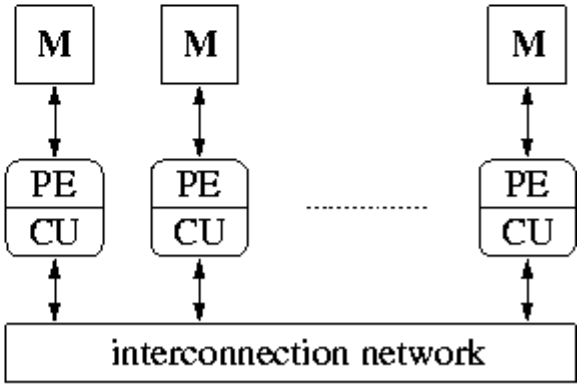
Shown here: array of synchronized processing elements

Single Instruction stream over a Single Data stream (SISD): Conventional sequential machines or uniprocessors.



(c) MISD

Multiple Instruction streams and a Single Data stream (MISD): Systolic arrays for pipelined execution.



(d) MIMD

Parallel computers or multiprocessor systems

Multiple Instruction streams over Multiple Data streams (MIMD): Parallel computers: Distributed memory multiprocessor system shown

Classified according to number of instruction streams (threads) and number of data streams in architecture

Current Trends In Parallel Architectures

- **The extension of "computer architecture" to support communication and cooperation:**

  Conventional or sequential

  - **OLD: Instruction Set Architecture (ISA)**
  - **NEW: *Communication Architecture***

- **Defines:**

  1 – **Critical abstractions, boundaries, and primitives (interfaces).**

  2 – **Organizational structures that implement interfaces (hardware or software)**

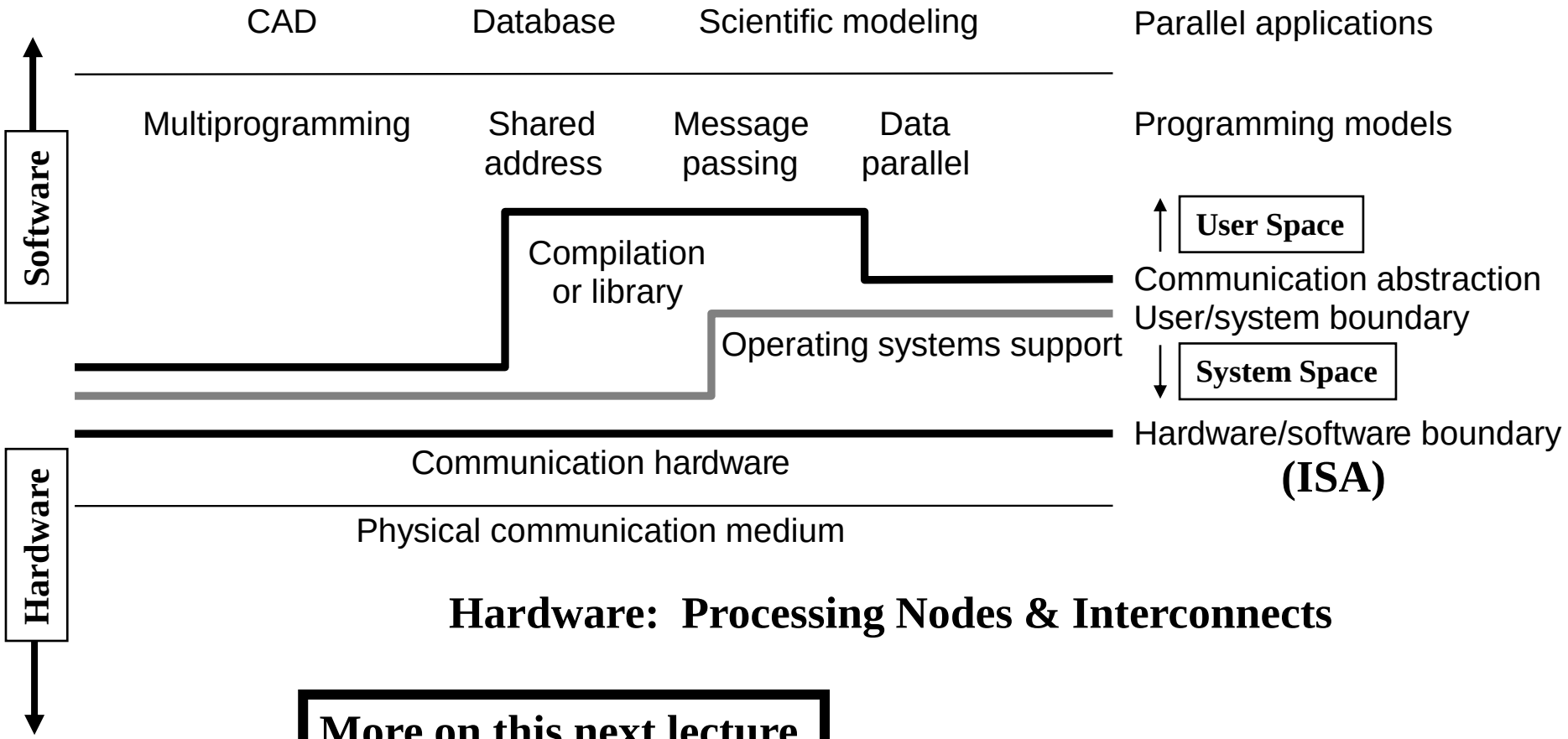
  Implementation of Interfaces

- **Compilers, libraries and OS are important bridges today**

  i.e. software abstraction layers

More on this next lecture

Modern Parallel Architecture Layered Framework



Hardware: Processing Nodes & Interconnects

More on this next lecture

Shared Address Space (SAS) Parallel Architectures

- Any processor can directly reference any memory location (in shared address space)
  - Communication occurs implicitly as result of loads and stores
- Convenient: — Communication is implicit via loads/stores
  - Location transparency
  - Similar programming model to time-sharing in uniprocessors
    - Except processes run on different processors
    - Good throughput on multiprogrammed workloads — i.e multi-tasking
- Naturally provided on a wide range of platforms
  - Wide range of scale: few to hundreds of processors
- Popularly known as *shared memory* machines or model
  - Ambiguous: Memory may be physically distributed among processing nodes. — i.e Distributed shared memory multiprocessors

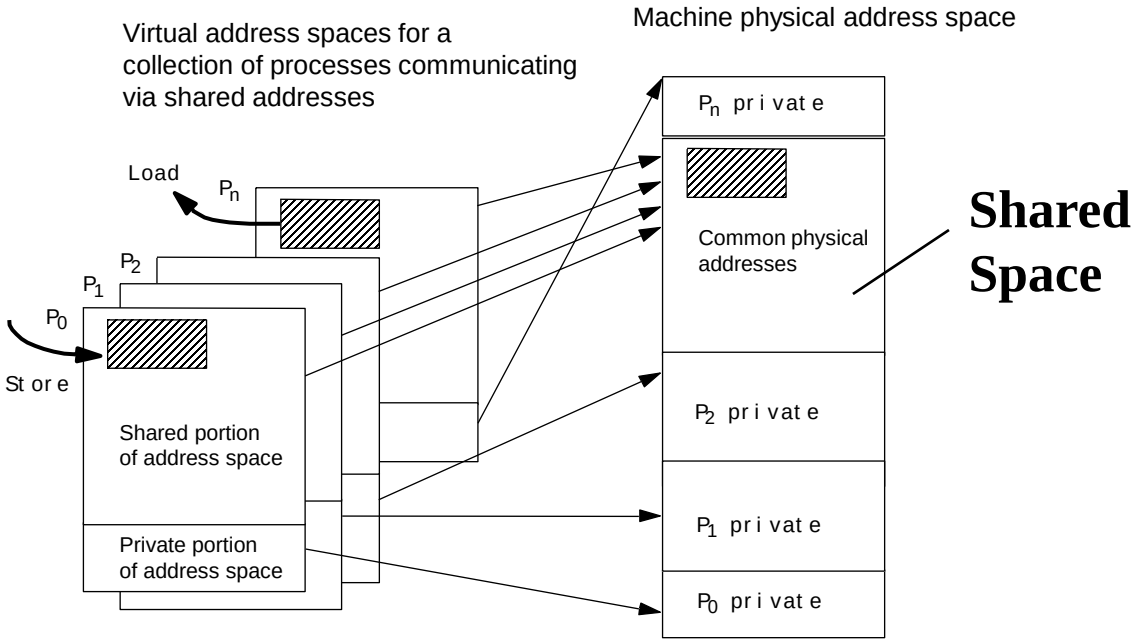
Sometimes called Tightly-Coupled Parallel Computers

Shared Address Space (SAS) Parallel Programming Model

- **Process: virtual address space plus one or more threads of control**
- **Portions of address spaces of processes are shared:**

In SAS:
Communication is implicit via loads/stores.

Ordering/Synchronization is explicit using synchronization Primitives.



- **Writes to shared address visible to other threads (in other processes too)**
- **Natural extension of the uniprocessor model:**
  - **Conventional memory operations used for communication**
- **Special atomic operations needed for synchronization:**
  - **Using Locks, Semaphores etc.**
- **OS uses shared memory to coordinate processes.**

Thus communication is implicit via loads/stores

i.e for event ordering and mutual exclusion

Thus synchronization is explicit

Models of Shared-Memory Multiprocessors

1 • **The Uniform/Centralized Memory Access (UMA) Model:**

- **All physical memory is shared by all processors.**
- **All processors have equal access (i.e equal memory bandwidth and access latency) to all memory addresses.**
- **Also referred to as Symmetric Memory Processors (SMPs).**

2 • **Distributed memory or Non-uniform Memory Access (NUMA) Model:**

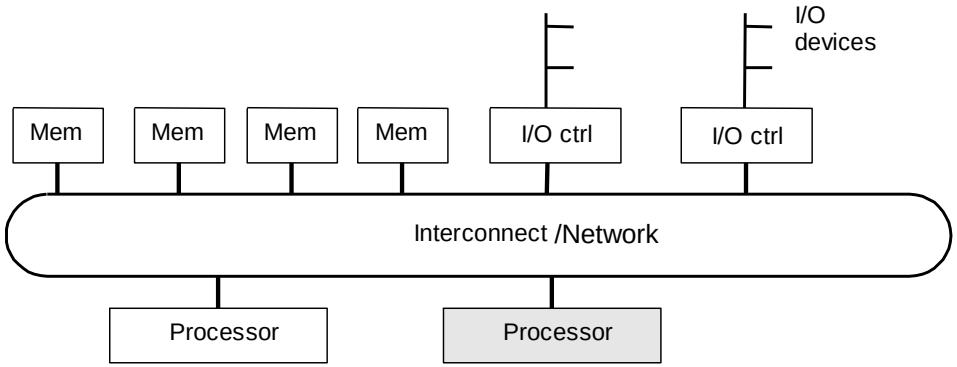
- **Shared memory is physically distributed locally among processors. Access latency to remote memory is higher.**

3 • **The Cache-Only Memory Architecture (COMA) Model:**

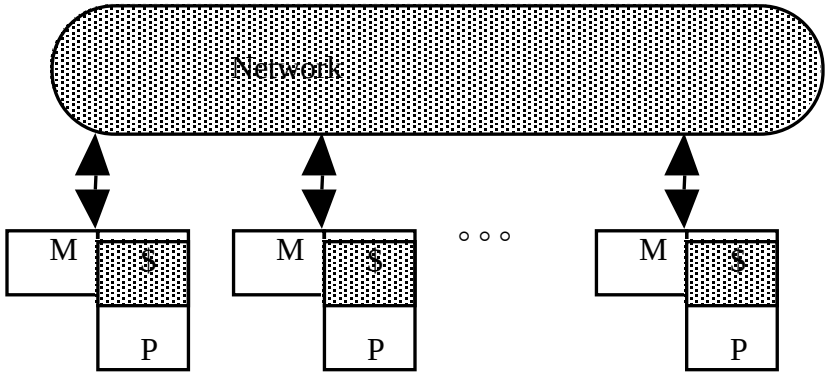
- **A special case of a NUMA machine where all distributed main memory is converted to caches.**
- **No memory hierarchy at each processor.**

Models of Shared-Memory Multiprocessors

1 **Uniform Memory Access (UMA) Model or Symmetric Memory Processors (SMPs).** **UMA**



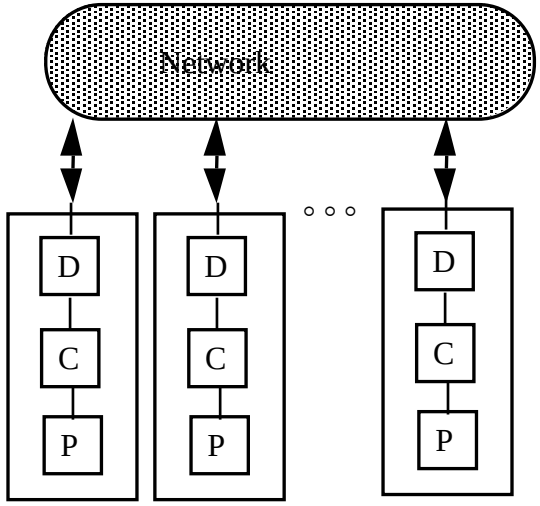
Interconnect:
Bus, Crossbar, Multistage network
P: Processor
M or Mem: Memory
C: Cache
D: Cache directory



NUMA

2 **Distributed memory or Non-uniform Memory Access (NUMA) Model**

3

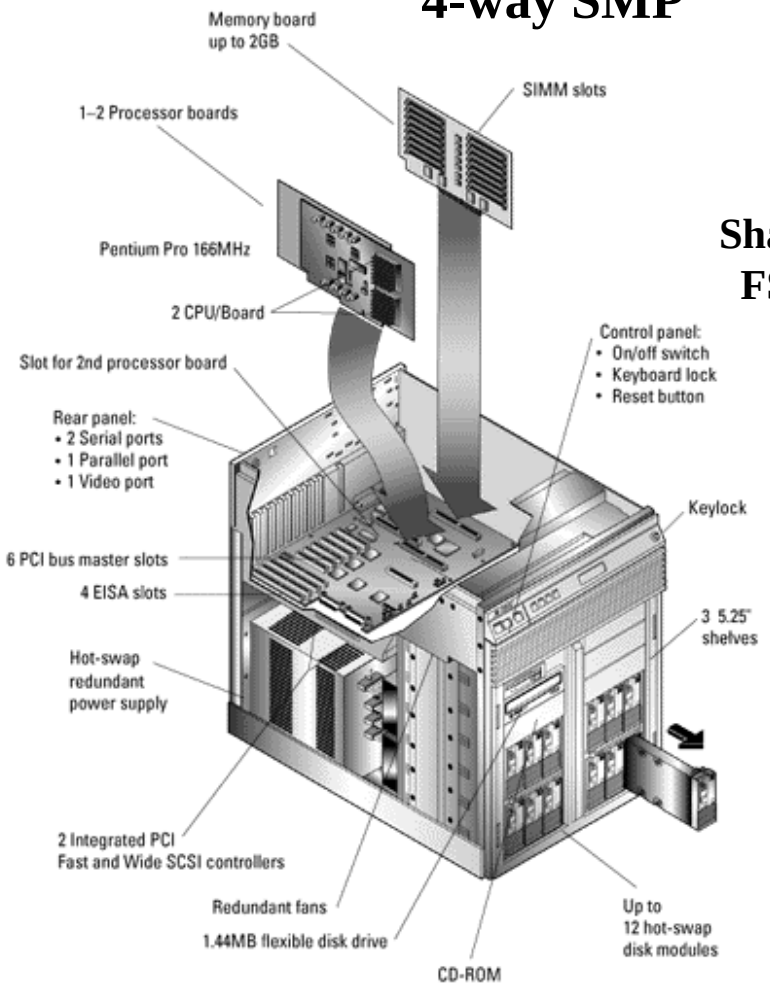


Cache-Only Memory Architecture (COMA)

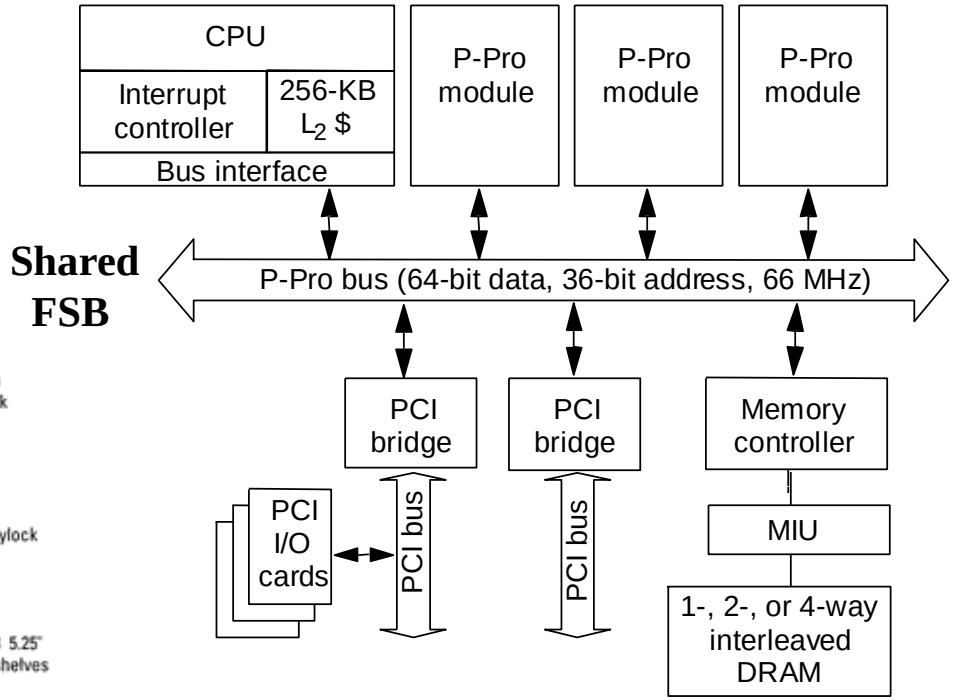
Uniform Memory Access (UMA) Example: Intel Pentium Pro Quad

Circa 1997

4-way SMP



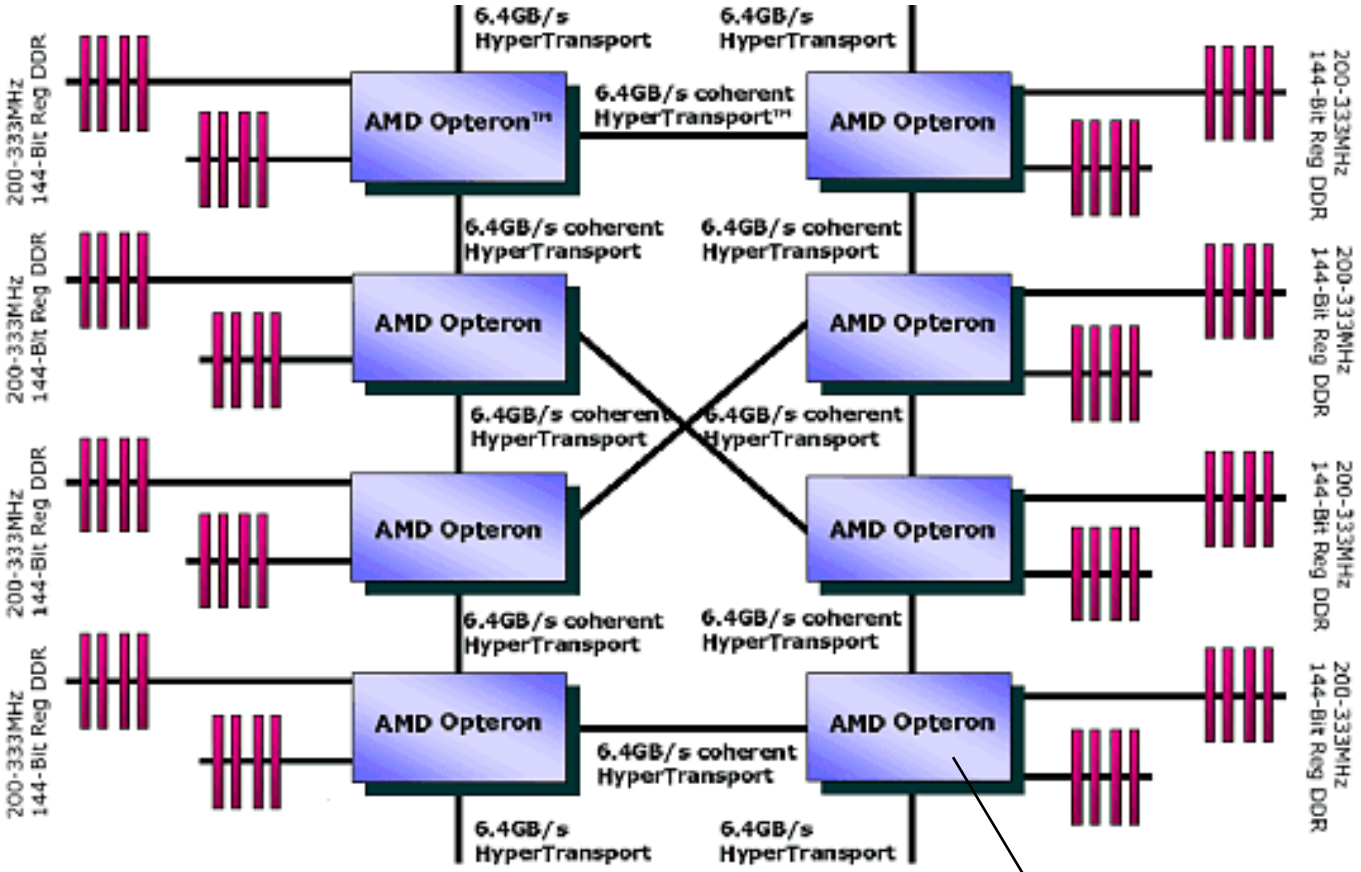
Bus-Based Symmetric Memory Processors (SMPs).



- **All coherence and multiprocessing glue in processor module**
- **Highly integrated, targeted at high volume**
- **Computing node used in Intel's ASCI-Red MPP**

A single Front Side Bus (FSB) is shared among processors
This severely limits scalability to only ~ 2-4 processors

Non-Uniform Memory Access (NUMA) Example: 8-Socket AMD Opteron Node

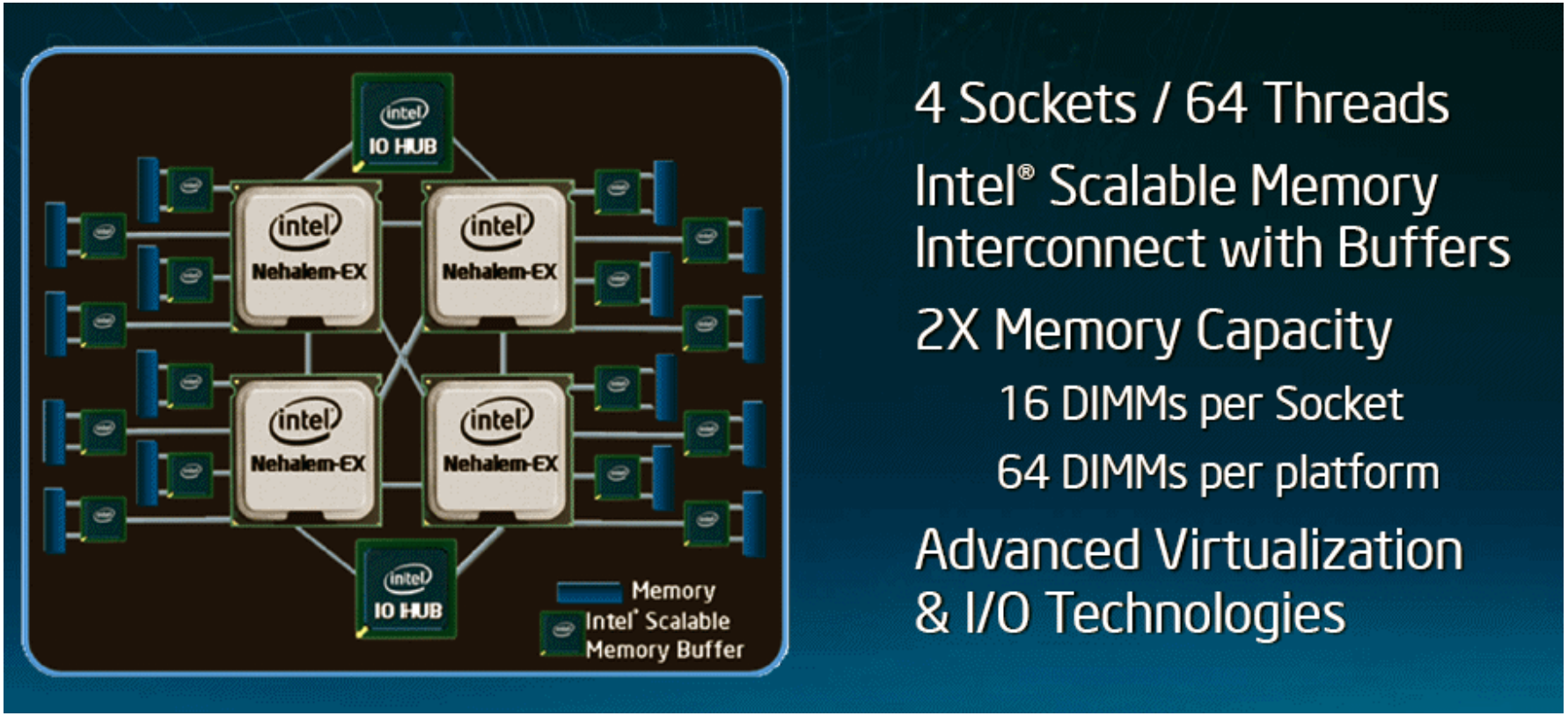


Circa 2003

Dedicated point-to-point interconnects (HyperTransport links) used to connect processors alleviating the traditional limitations of FSB-based SMP systems.
Each processor has two integrated DDR memory channel controllers: memory bandwidth scales up with number of processors.
NUMA architecture since a processor can access its own memory at a lower latency than access to remote memory directly connected to other processors in the system.

Total 32 processor cores when quad core Opteron processors used (128 cores with 16-core processors)

Non-Uniform Memory Access (NUMA) Example: 4-Socket Intel Nehalem-EX Node



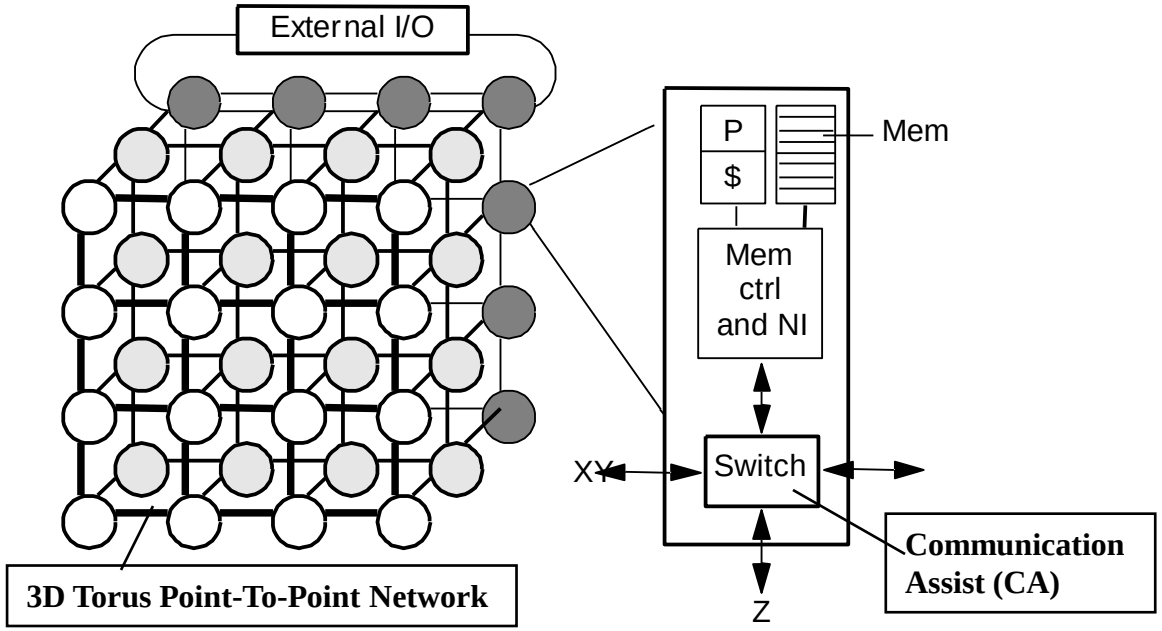
Distributed Shared-Memory Multiprocessor System Example: Cray T3E

NUMA MPP Example

Circa 1995-1999

MPP = Massively Parallel Processor System

More recent Cray MPP Example: Cray X1E Supercomputer



- **Scale up to 2048 processors, DEC Alpha EV6 microprocessor (COTS)**
- **Custom 3D Torus point-to-point network, 480MB/s links**
- **Memory controller generates communication requests for non-local references**
- **No hardware mechanism for coherence (SGI Origin etc. provide this)**

Example of Non-uniform Memory Access (NUMA)

Message-Passing Multicomputers

- **Comprised of multiple autonomous computers (computing nodes) connected via a suitable network.**

  Industry standard System Area Network (SAN) or proprietary network

- **Each node consists of one or more processors, local memory, attached storage and I/O peripherals and Communication Assist (CA).**
- **Local memory is only accessible by local processors in a node (no shared memory among nodes).**
- **Inter-node communication is carried explicitly out by message passing through the connection network via send/receive operations.**

  Thus communication is explicit

- **Process communication achieved using a message-passing programming environment (e.g. PVM, MPI).**

  Portable, platform-independent

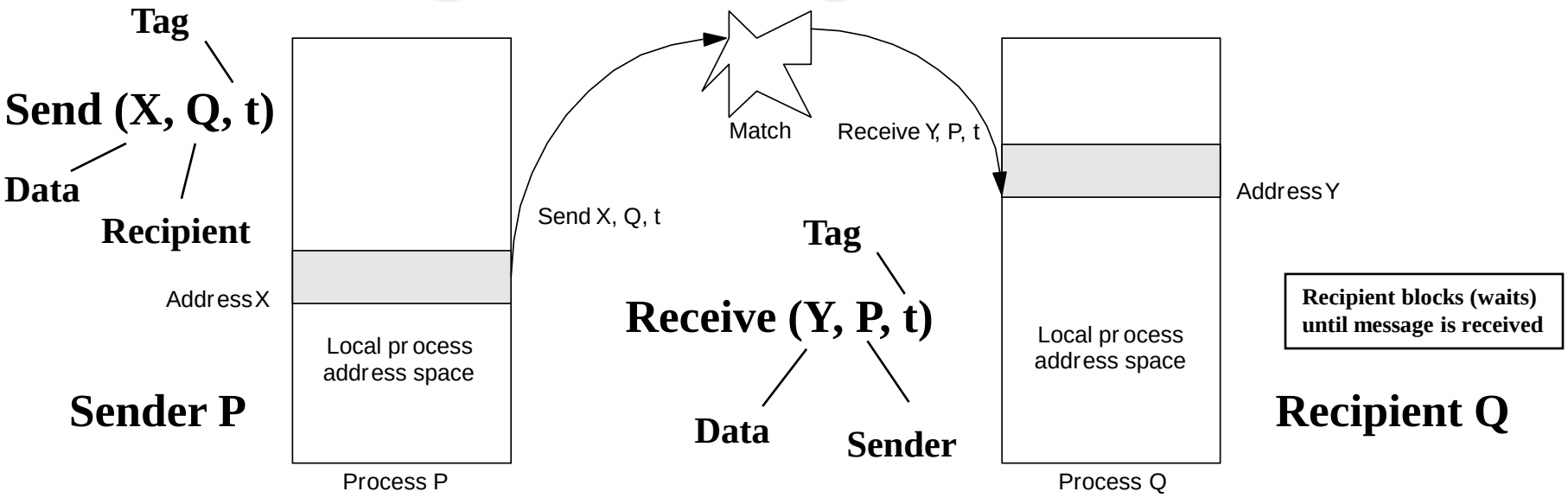
  - **Programming model more removed or abstracted from basic hardware operations**
- **Include:**

  1 – **A number of commercial Massively Parallel Processor systems (MPPs).**

  2 – **Computer clusters that utilize commodity of-the-shelf (COTS) components.**

Also called Loosely-Coupled Parallel Computers

Message-Passing Abstraction



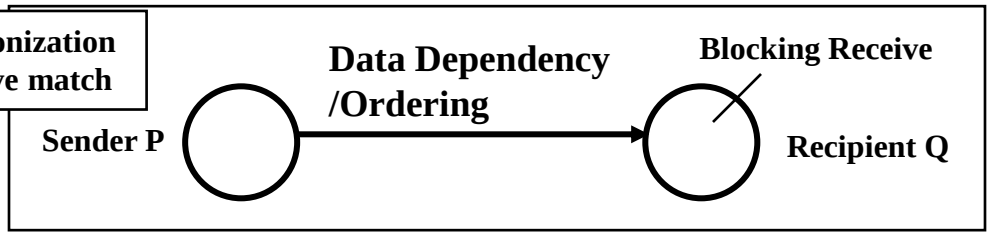
- Send specifies buffer to be transmitted and receiving process.
- Receive specifies sending process and application storage to receive into.
- Memory to memory copy possible, but need to name processes.
- Optional tag on send and matching rule on receive.
- User process names local data and entities in process/tag space too
- In simplest form, the send/receive match achieves implicit pairwise synchronization event
  - Ordering of computations according to dependencies
- Many possible overheads: copying, buffer management, protection ...

Communication is explicit via sends/receives

i.e event ordering, in this case

Synchronization is implicit

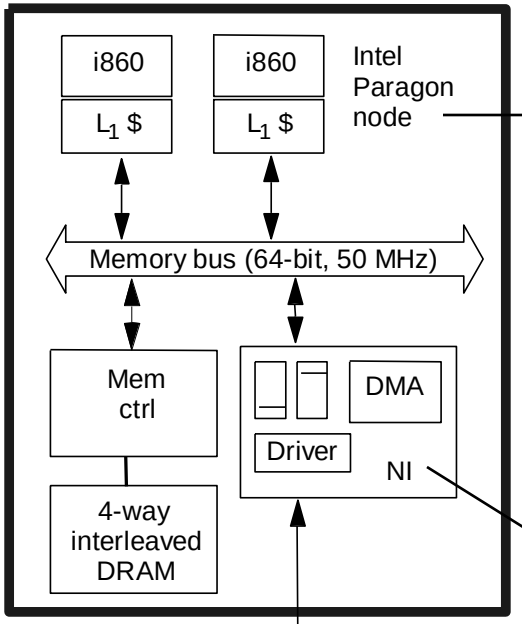
Pairwise synchronization using send/receive match



Message-Passing Example: Intel Paragon

Circa 1983

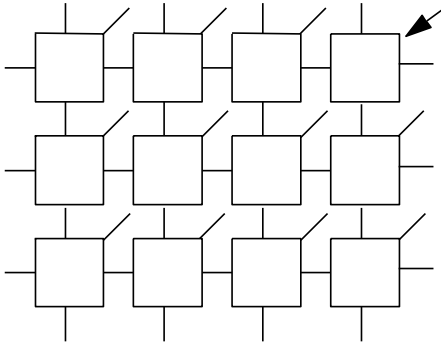
Sandia' s Intel Paragon XP/S-based Supercomputer



Each node Is a 2-way-SMP

Communication Assist (CA)

8 bits, 175 MHz, bidirectional



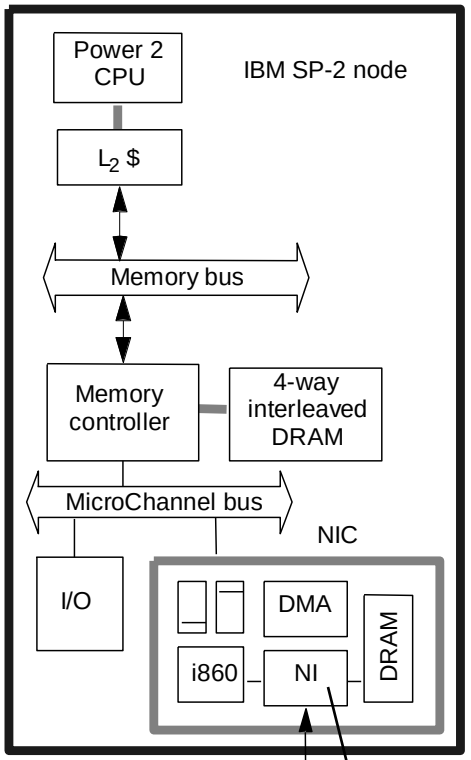
2D grid network with processing node attached to every switch

2D grid point to point network

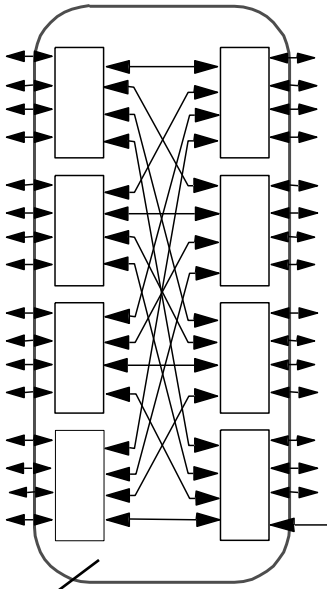
Message-Passing Example: IBM SP-2

MPP

Circa 1994-1998



General interconnection network formed from 8-port switches



- **Made out of essentially complete RS6000 workstations**
- **Network interface integrated in I/O bus (bandwidth limited by I/O bus)**

Communication Assist (CA)

Multi-stage network

MPP = Massively Parallel Processor System

Message-Passing MPP Example:

IBM Blue Gene/L

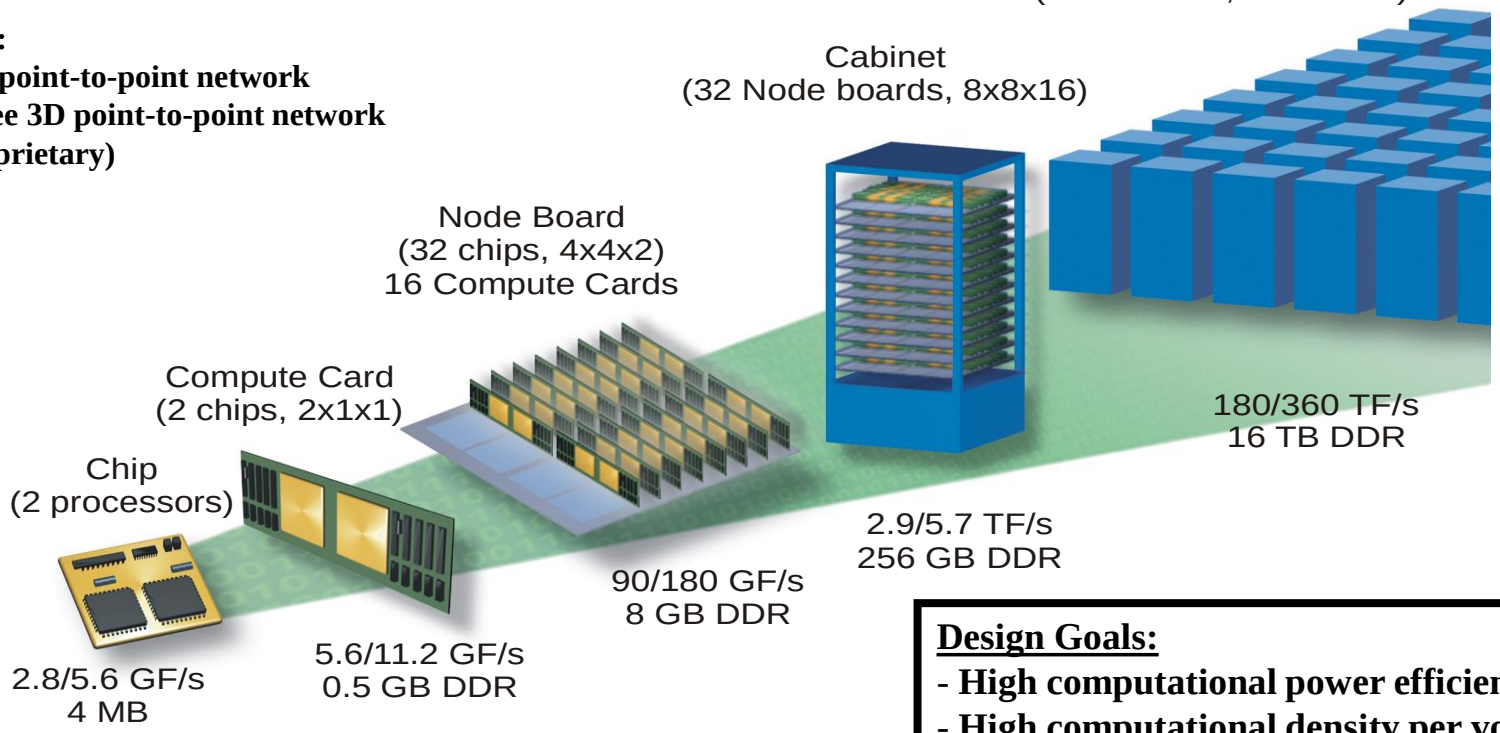
Circa 2005

(2 processors/chip) • (2 chips/compute card) • (16 compute cards/node board) • (32 node boards/tower) • (64 tower) = <u>128k = 131072 (0.7 GHz PowerPC 440) processors</u> (64k nodes)

2.8 Gflops peak per processor core

System Location: Lawrence Livermore National Laboratory

Networks:
3D Torus point-to-point network
Global tree 3D point-to-point network
(both proprietary)

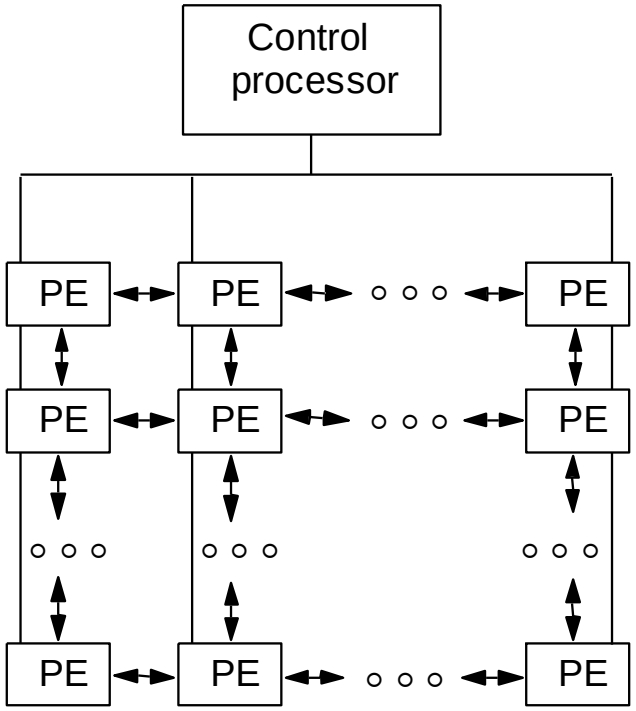


<u>Design Goals:</u>
- High computational power efficiency
- High computational density per volume

<u>LINPACK Performance:</u>
280,600 GFLOPS = 280.6 TeraFLOPS = 0.2806 Peta FLOP
<u>Top Peak FP Performance:</u>
Now about 367,000 GFLOPS = 367 TeraFLOPS = 0.367 Peta FLOP

Data Parallel Systems SIMD in Flynn taxonomy

- **Programming model (Data Parallel)**
  - **Similar operations performed in parallel on each element of data structure**
  - **Logically single thread of control, performs sequential or parallel steps**
  - **Conceptually, a processor is associated with each data element**
- **Architectural model**
  - **Array of many simple processors each with little memory**
    - **Processors don't sequence through instructions**
  - **Attached to a control processor that issues instructions**
  - **Specialized and general communication, global synchronization**



All PE are synchronized (same instruction or operation in a given cycle)

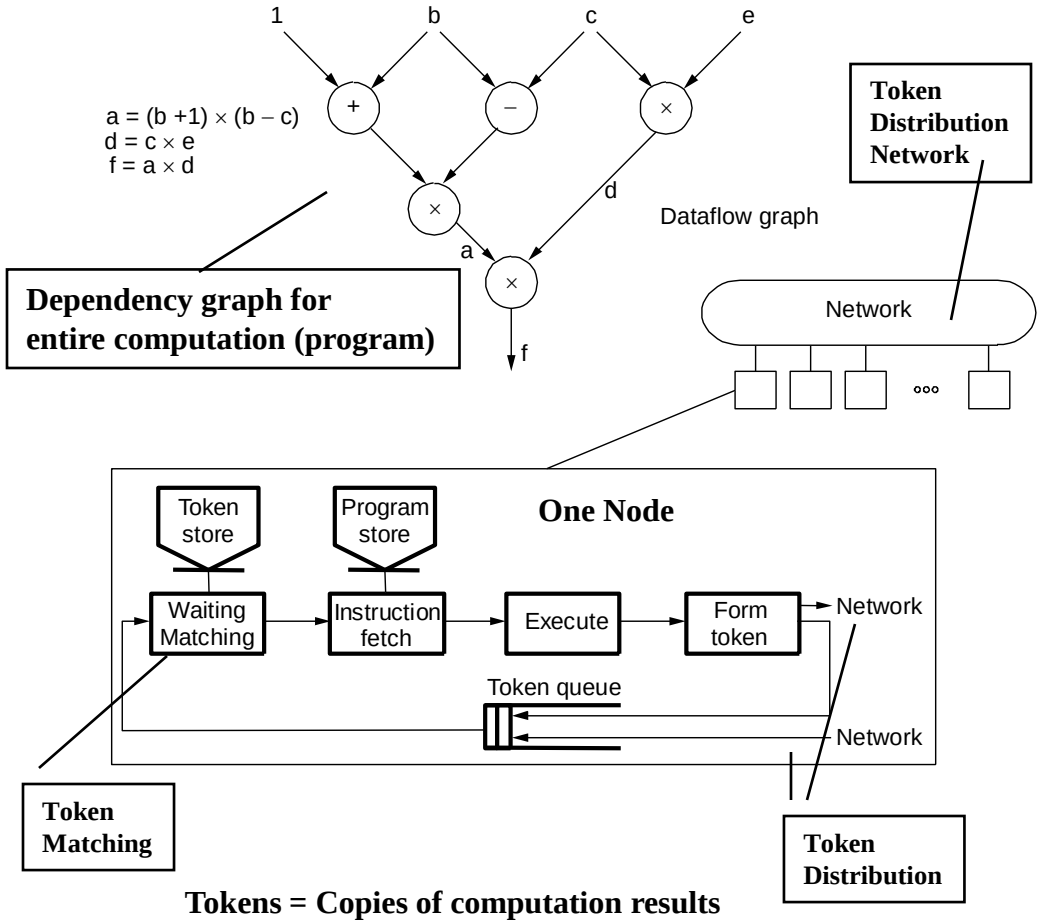
Other Data Parallel Architectures: Vector Machines

PE = Processing Element

Dataflow Architectures

- **Represent computation as a graph of essential data dependencies**
  - **Non-Von Neumann Architecture (Not PC-based Architecture)**
  - **Logical processor at each node, activated by availability of operands**
  - **Message (tokens) carrying tag of next instruction sent to next processor**
  - **Tag compared with others in matching store; match fires execution**

i.e data or results



Research Dataflow machine prototypes include:

- The MIT Tagged Architecture
- The Manchester Dataflow Machine

The Tomasulo approach for dynamic instruction execution utilizes a dataflow driven execution engine:

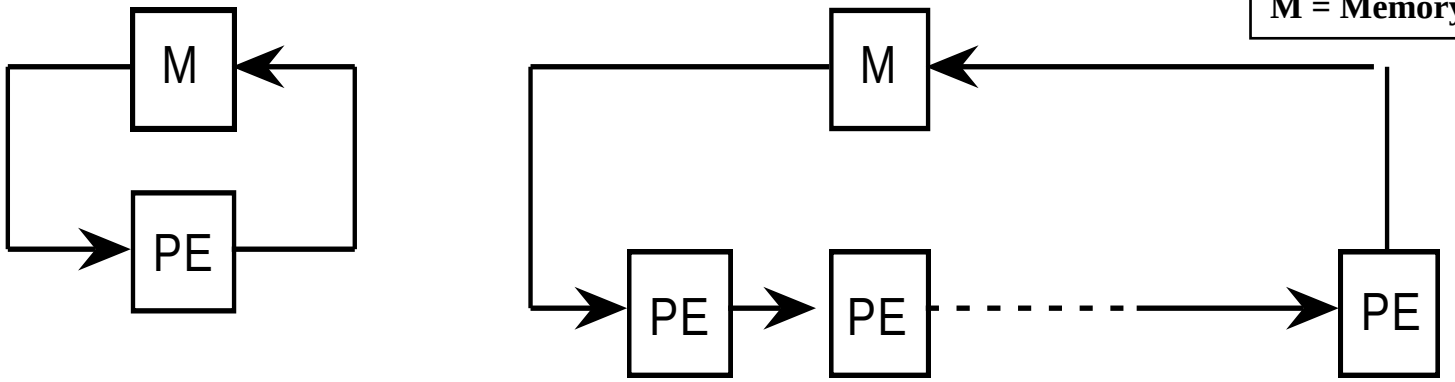
- The data dependency graph for a small window of instructions is constructed dynamically when instructions are issued in order of the program.
- The execution of an issued instruction is triggered by the availability of its operands (data it needs) over the CDB.

Example of Flynn's Taxonomy's MISD (Multiple Instruction Streams Single Data Stream):

Systolic Architectures

- Replace single processor with an array of regular processing elements
- Orchestrate data flow for high throughput with less memory access

PE = Processing Element
M = Memory



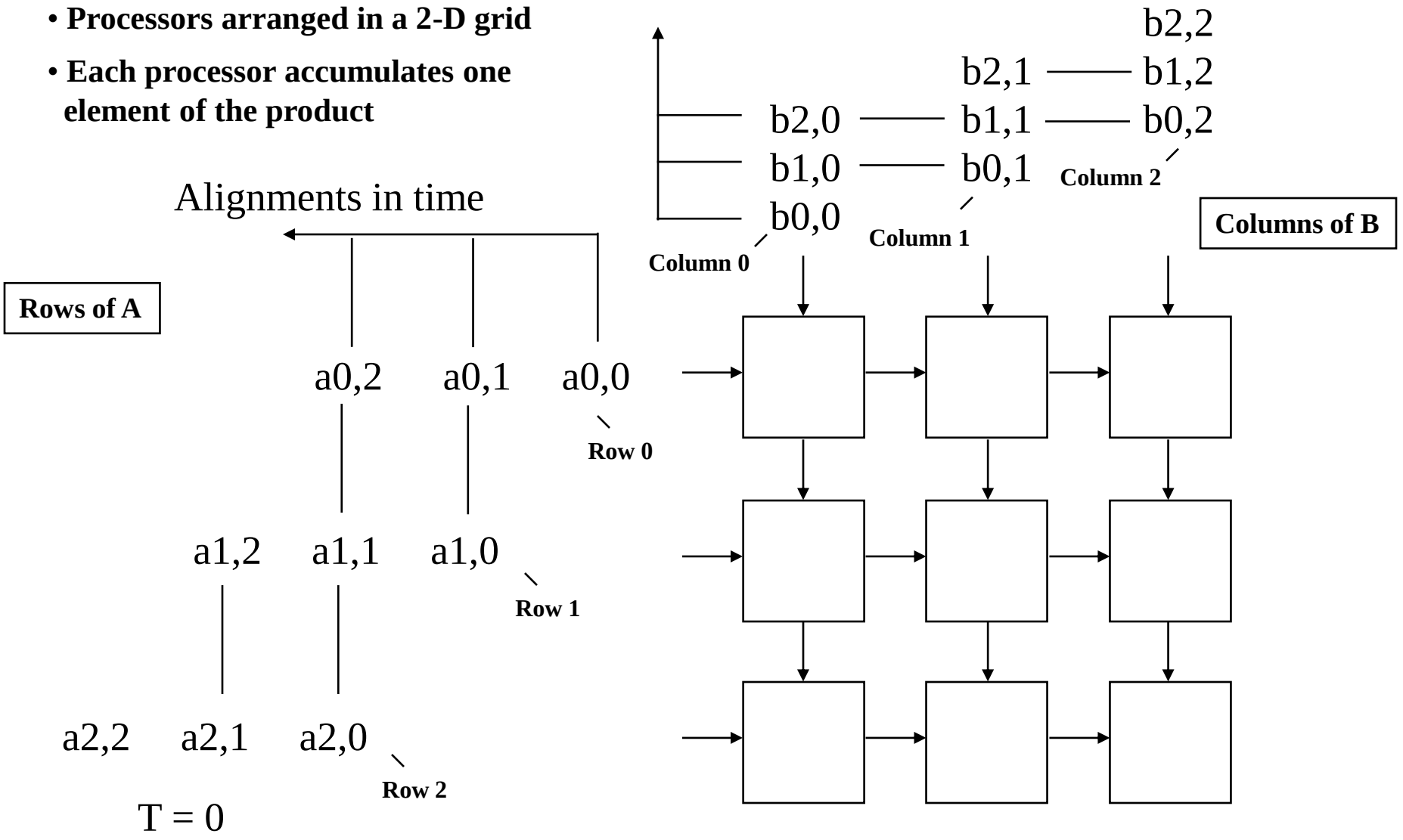
- Different from linear pipelining
  - Nonlinear array structure, multidirection data flow, each PE may have (small) local instruction and data memory
- Different from SIMD: each PE may do something different
- Initial motivation: VLSI Application-Specific Integrated Circuits (ASICs)
- Represent algorithms directly by chips connected in regular pattern

A possible example of MISD in Flynn's Classification of Computer Architecture

Systolic Array Example: C = A X B

3x3 Systolic Array Matrix Multiplication

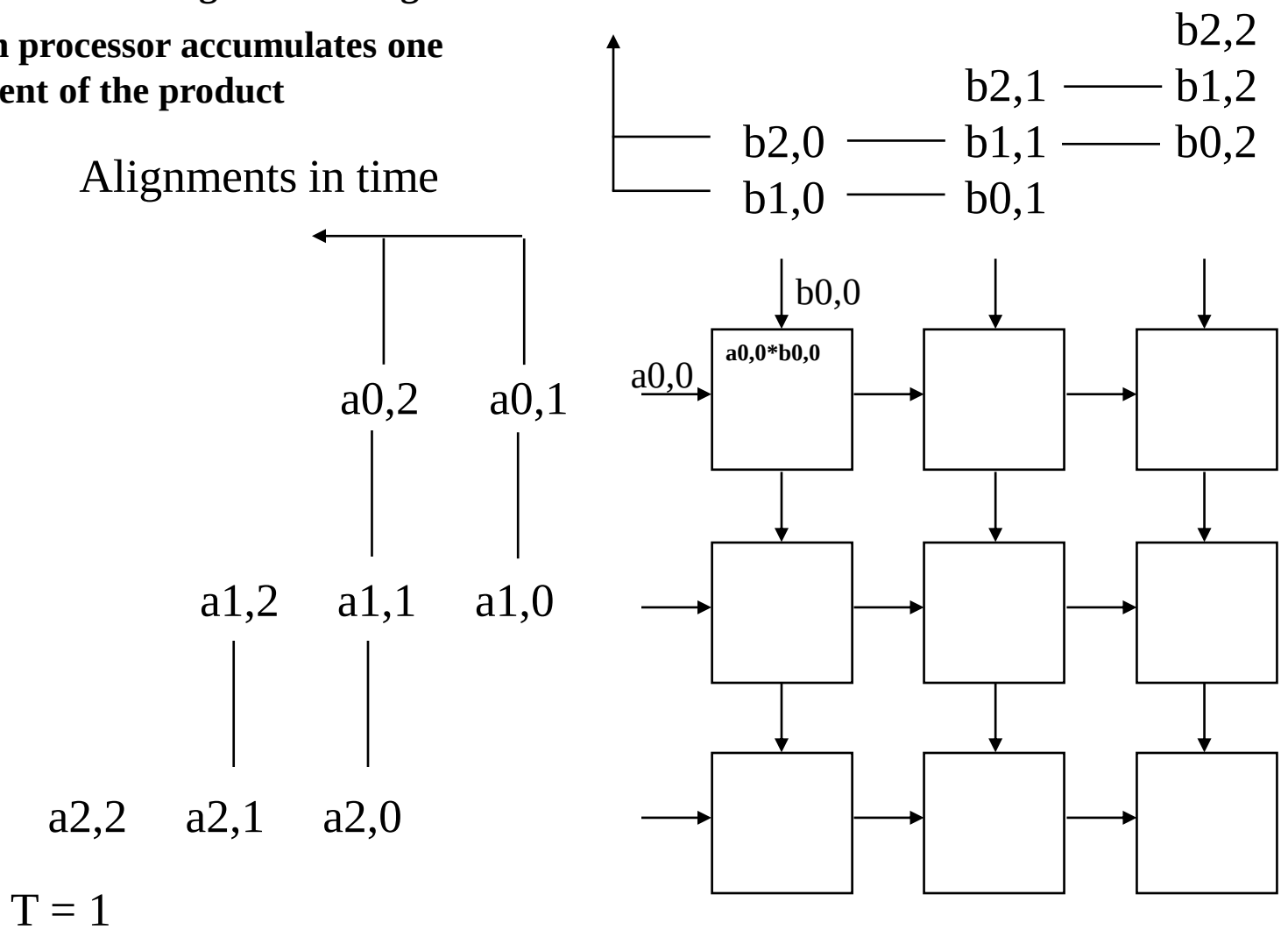
- **Processors arranged in a 2-D grid**
- **Each processor accumulates one element of the product**



Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

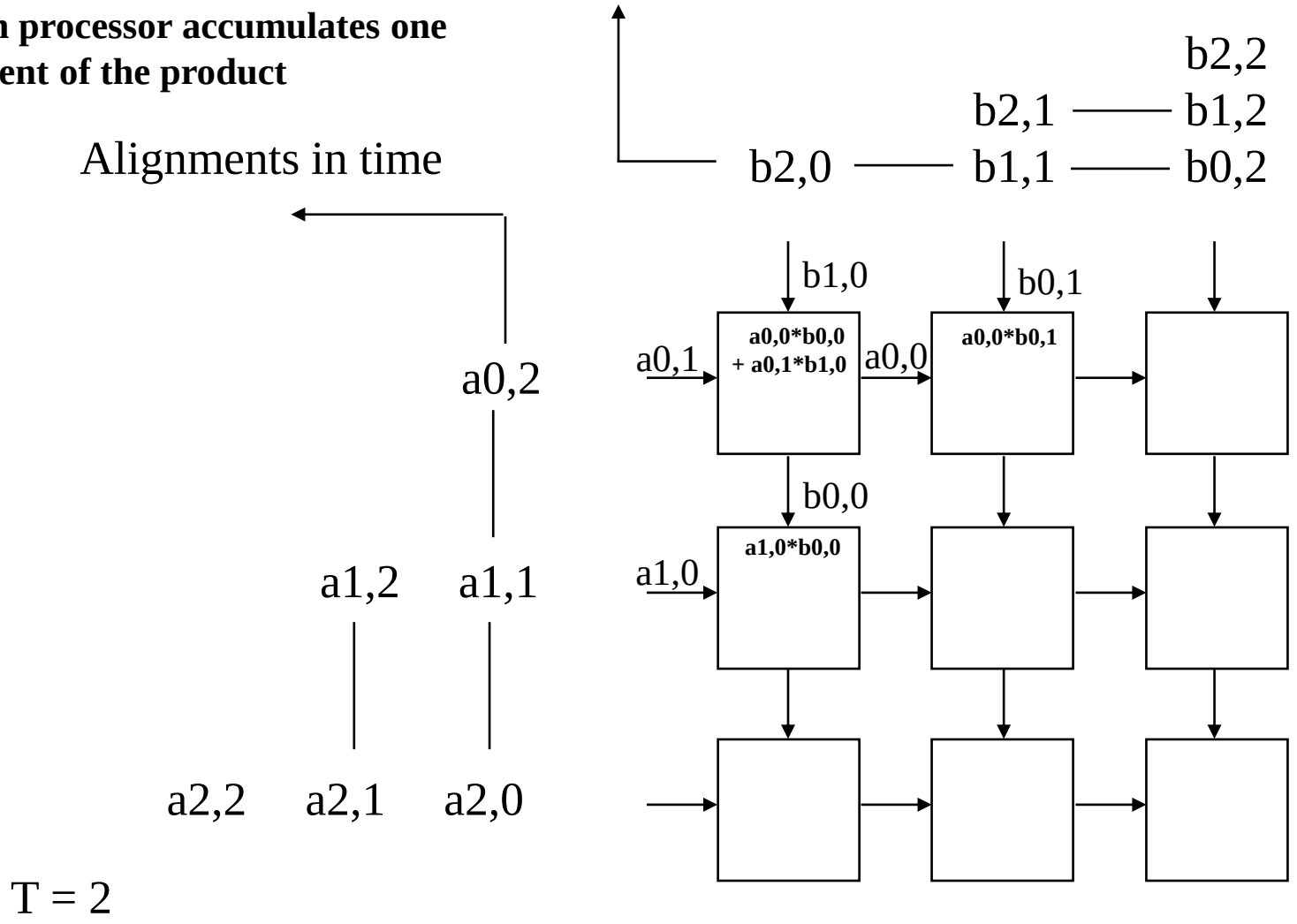
- **Processors arranged in a 2-D grid**
- **Each processor accumulates one element of the product**



Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

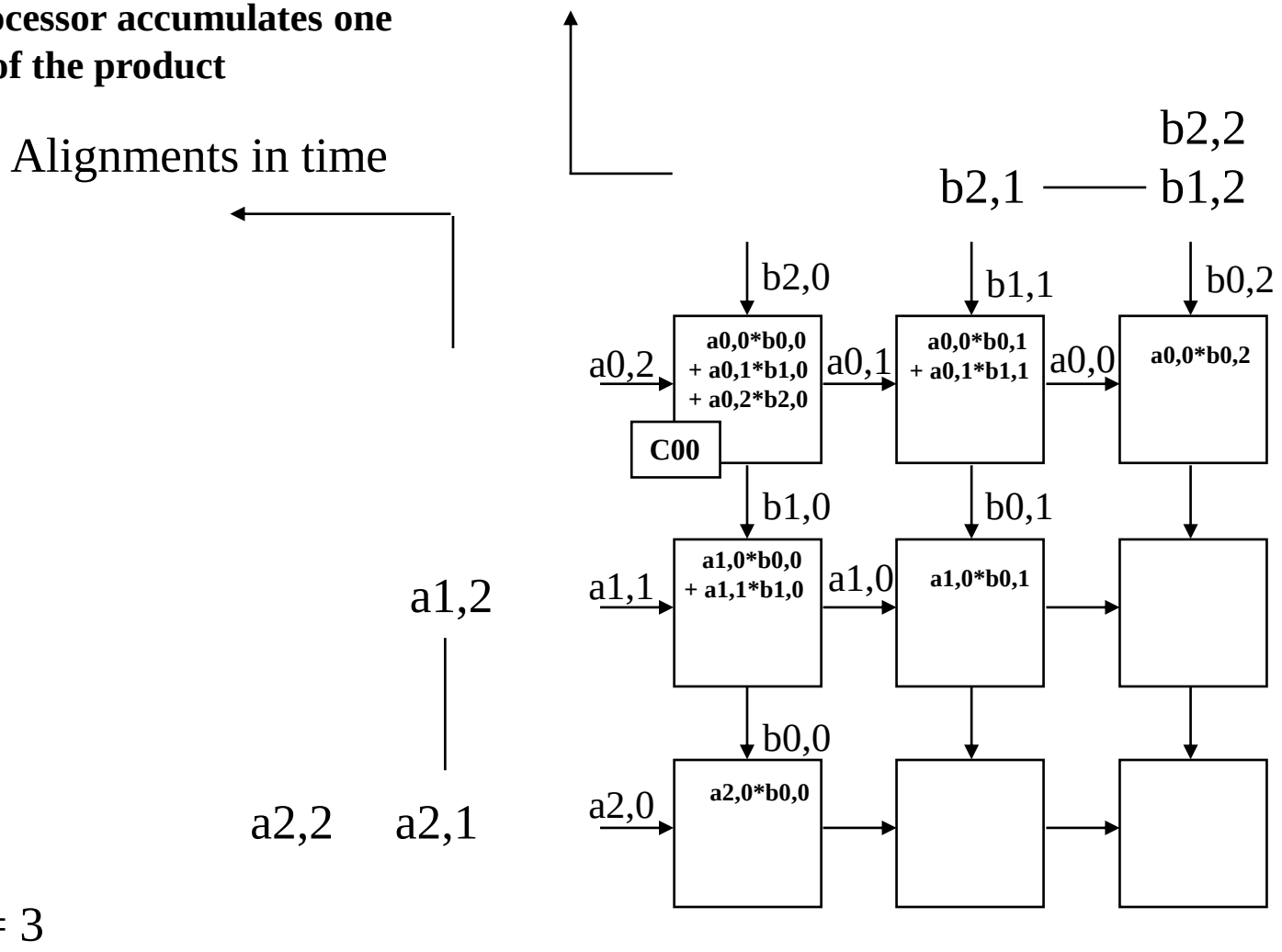
- **Processors arranged in a 2-D grid**
- **Each processor accumulates one element of the product**



Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

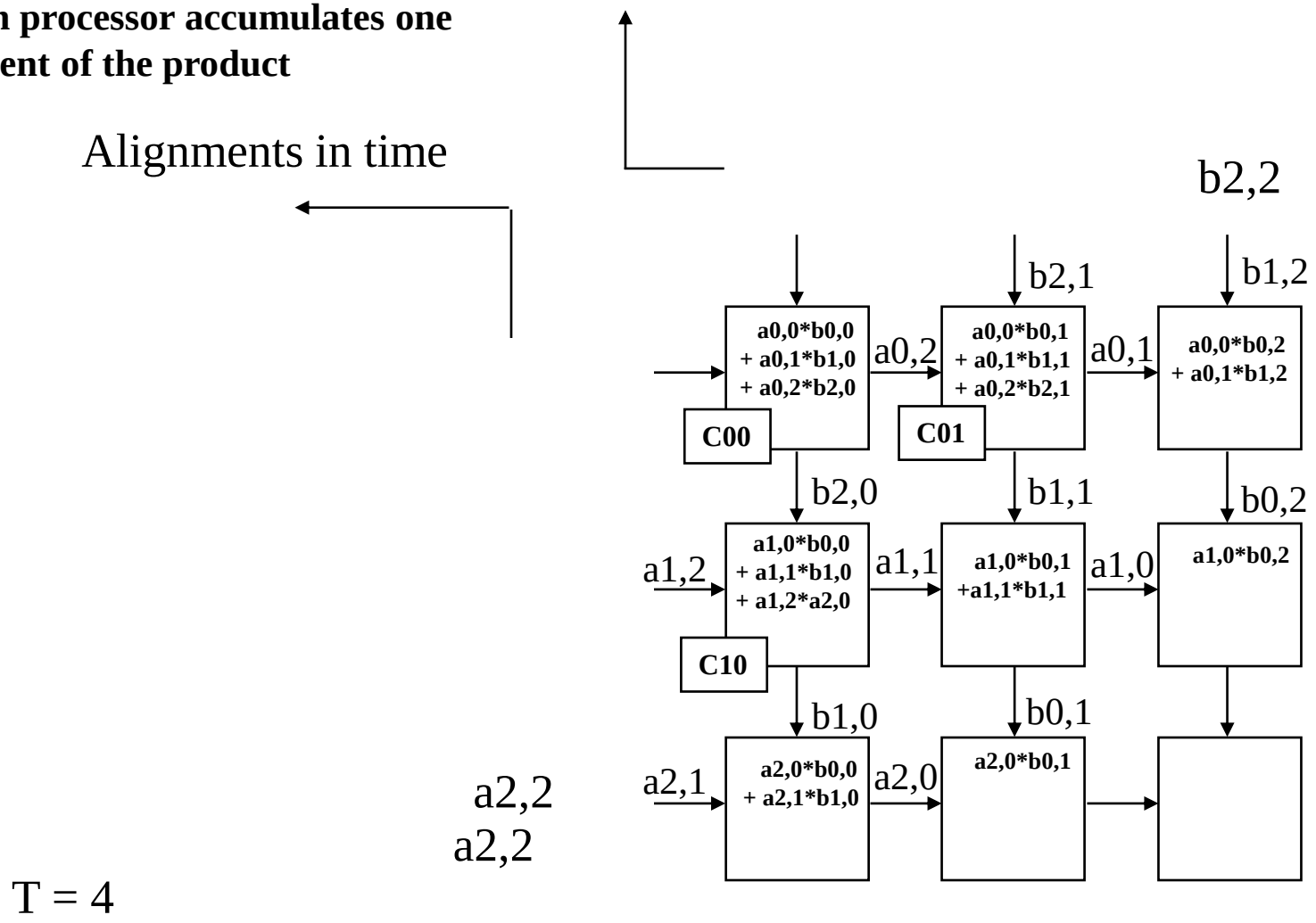
- **Processors arranged in a 2-D grid**
- **Each processor accumulates one element of the product**



Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- **Processors arranged in a 2-D grid**
- **Each processor accumulates one element of the product**

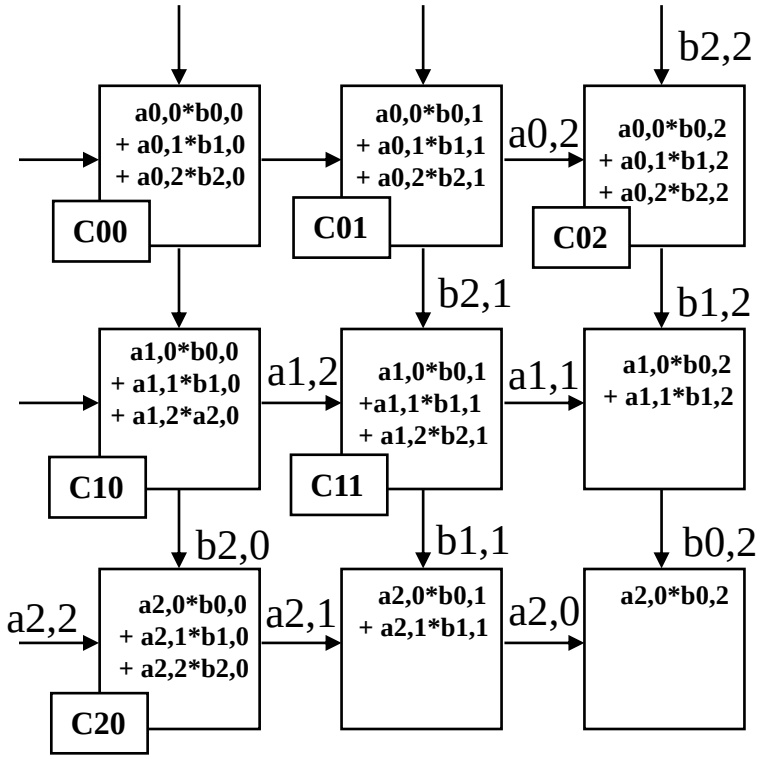


Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- **Processors arranged in a 2-D grid**
- **Each processor accumulates one element of the product**

Alignments in time

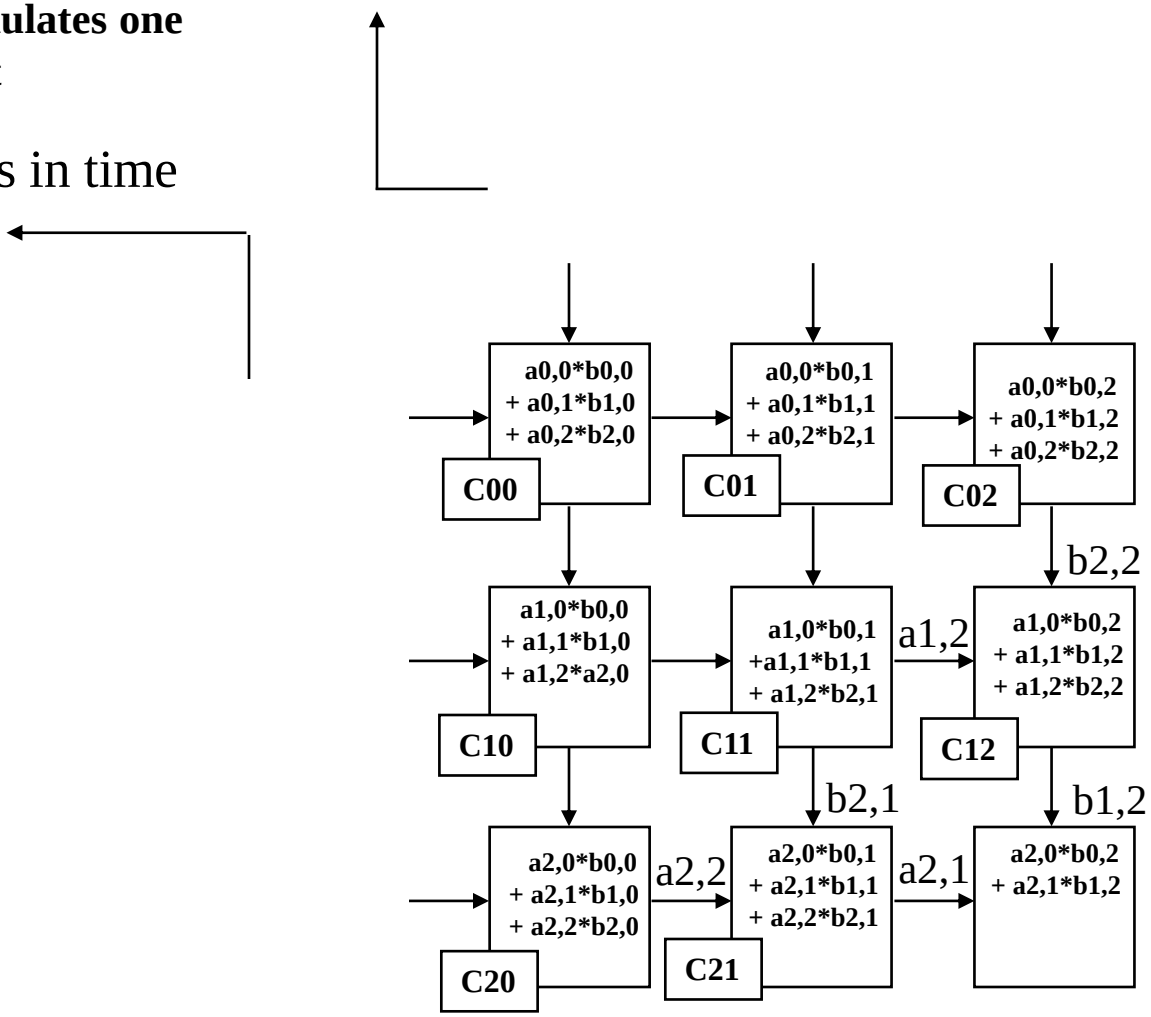


T = 5

Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- **Processors arranged in a 2-D grid**
- **Each processor accumulates one element of the product**



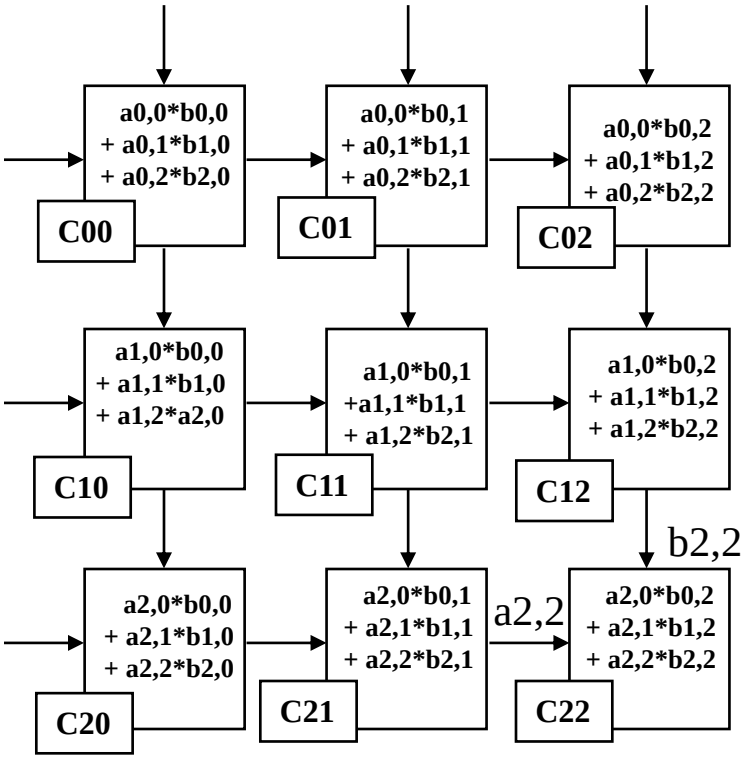
Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- **Processors arranged in a 2-D grid**
- **Each processor accumulates one element of the product**

On one processor = $O(n^3)$ t = 27?
Speedup = 27/7 = 3.85

Alignments in time



Done

T = 7

Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/