Advanced parallel programming

Dottorato di Ricerca
Dip. Informatica di Pisa

M. Danelutto
Giugno-Luglio 2007

Program

- Introduction
- Classical programming models
- Structured programming models
- Skeleton parallel programming environments
- Open problems
- Project

Goal

- parallel programming:
  - problems and solutions
- strcutured parallel programming tools
  - from Pisa and from abroad :-)
- advanced topics
  - grids
  - components
  - autonomic managers
  - semi formal reasoning

Final exercise

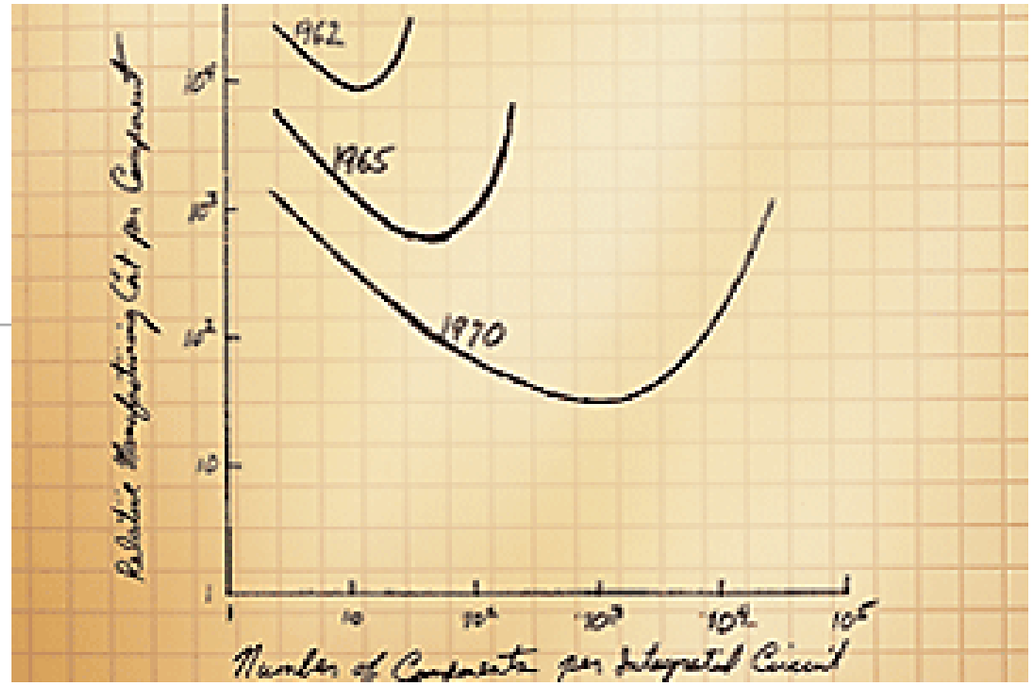
- Project:
  - implementation of a simple prototype on a Linux cluster
  - application (student choice)
    (possibly from his/her own research framework)
- discussed in a seminar
  - application design
  - parallelisation techniques
  - tools
  - results

Technology ...

Classification(s) (used in the following ...)

- Flynn (1972)
  - SISD, SIMD, MISD, MIMD
- MIMD
  - distributed memory, shared memory, distributed shared memory
- Memory access
  - UMA, NUMA, NORMA
- Interconnection networks
  - bus, crossbar, mesh, hypercube, fat-tree, ...

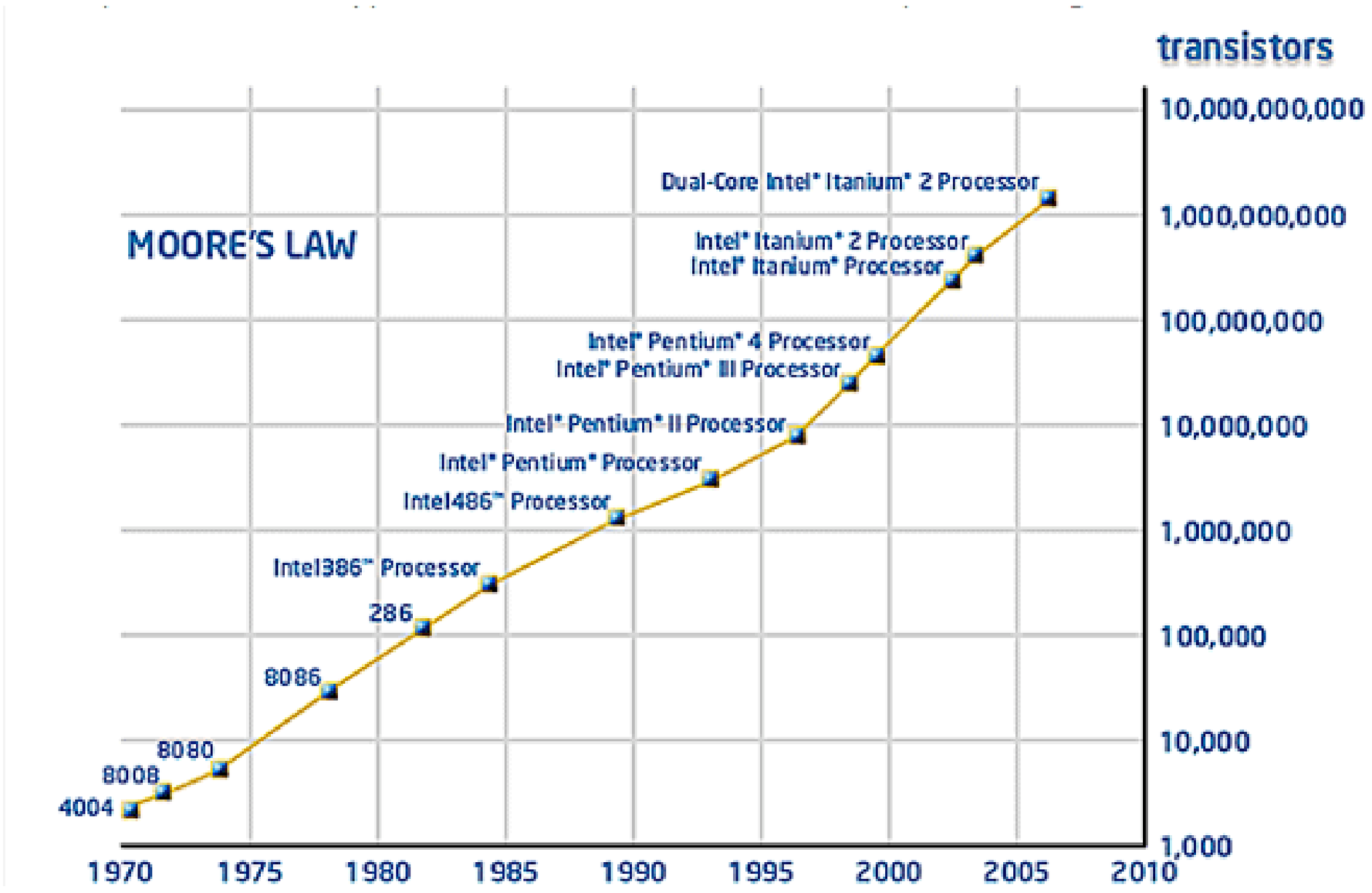
Moore’s Law



- Moore's original statement can be found in his publication "Cramming more components onto integrated circuits", Electronics Magazine 19 April 1965:

- “The complexity for minimum component costs has increased at a rate of roughly a factor of two per year ... Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000. I believe that such a large circuit can be built on a single wafer.”

Moore - Intel


transistors
MOORE'S LAW
Dual-Core Intel® Itanium® 2 Processor
Intel® Itanium® 2 Processor
Intel® Itanium® Processor
Intel® Pentium® 4 Processor
Intel® Pentium® III Processor
Intel® Pentium® II Processor
Intel® Pentium® Processor
Intel486™ Processor
Intel386™ Processor
286
8086
8080
8008
4004
10,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
1,000
1970
1975
1980
1985
1990
1995
2000
2005
2010

Moore’s law (revisited)

- The original one:
  - number of active components/transistors doubles each year and a half
- limits:
  - usage ? processor model ?
  - big chip areas for small performance improvements !
- and therefore
  - law moved to cores per chip !!!

Top500

- rating publiches every 6 months (june and november)
  - at the same time of Supercomputing conference
- precise benchmark suite
- measures (raw) computing power
  - but also the trend (archive from 1993)

Evolution: architectures

- 1993

Architecture	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
SIMD	35	7.00 %	64	135	54272
MPP	119	23.80 %	400	826	14766
Single Processor	97	19.40 %	147	186	99
SMP	249	49.80 %	511	640	1983
Totals	**500**	**100%**	**1122.84**	**1786.21**	**71120**

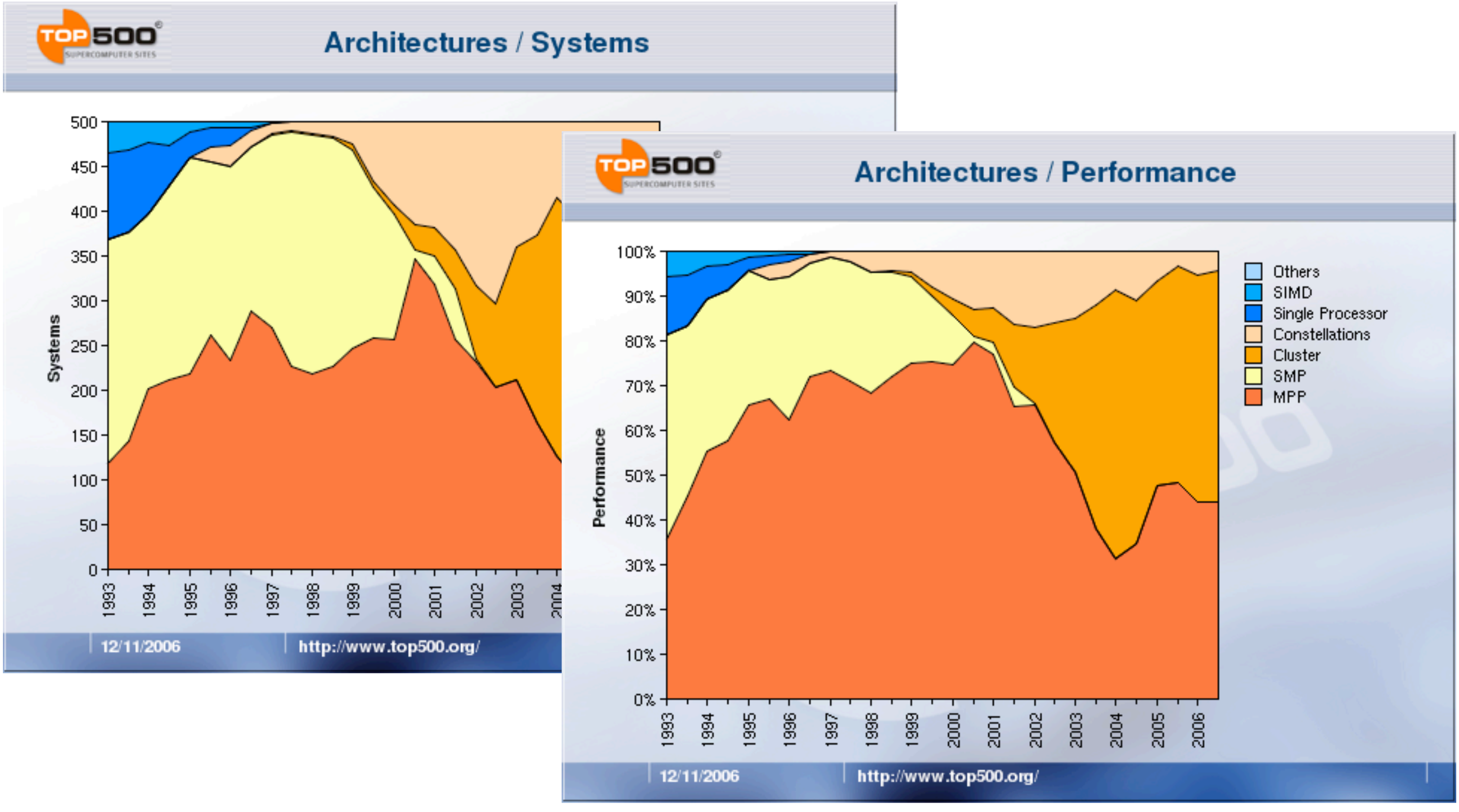
- 2000

Architecture	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
Constellations	93	18.60 %	6941	11187	10432
MPP	257	51.40 %	48026	72377	92081
Cluster	11	2.20 %	2260	3719	3668
SMP	139	27.80 %	7003	8392	9477
Totals	**500**	**100%**	**64230.11**	**95676.11**	**115658**

- 2006

Architecture	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
Constellations	31	6.20 %	156728	250009	51274
MPP	108	21.60 %	1552019	1970903	484370
Cluster	361	72.20 %	1818711	2992636	485317
Totals	**500**	**100%**	**3527458.35**	**5213548.18**	**1020961**

Evolution: architectures



Evolution: processor architecture (top500)

- 1993

Processor Architecture	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
Vector	334	66.80 %	650	792	1242
Scalar	131	26.20 %	408	859	15606
SIMD	35	7.00 %	64	135	54272
Totals	**500**	**100%**	**1122.84**	**1786.21**	**71120**

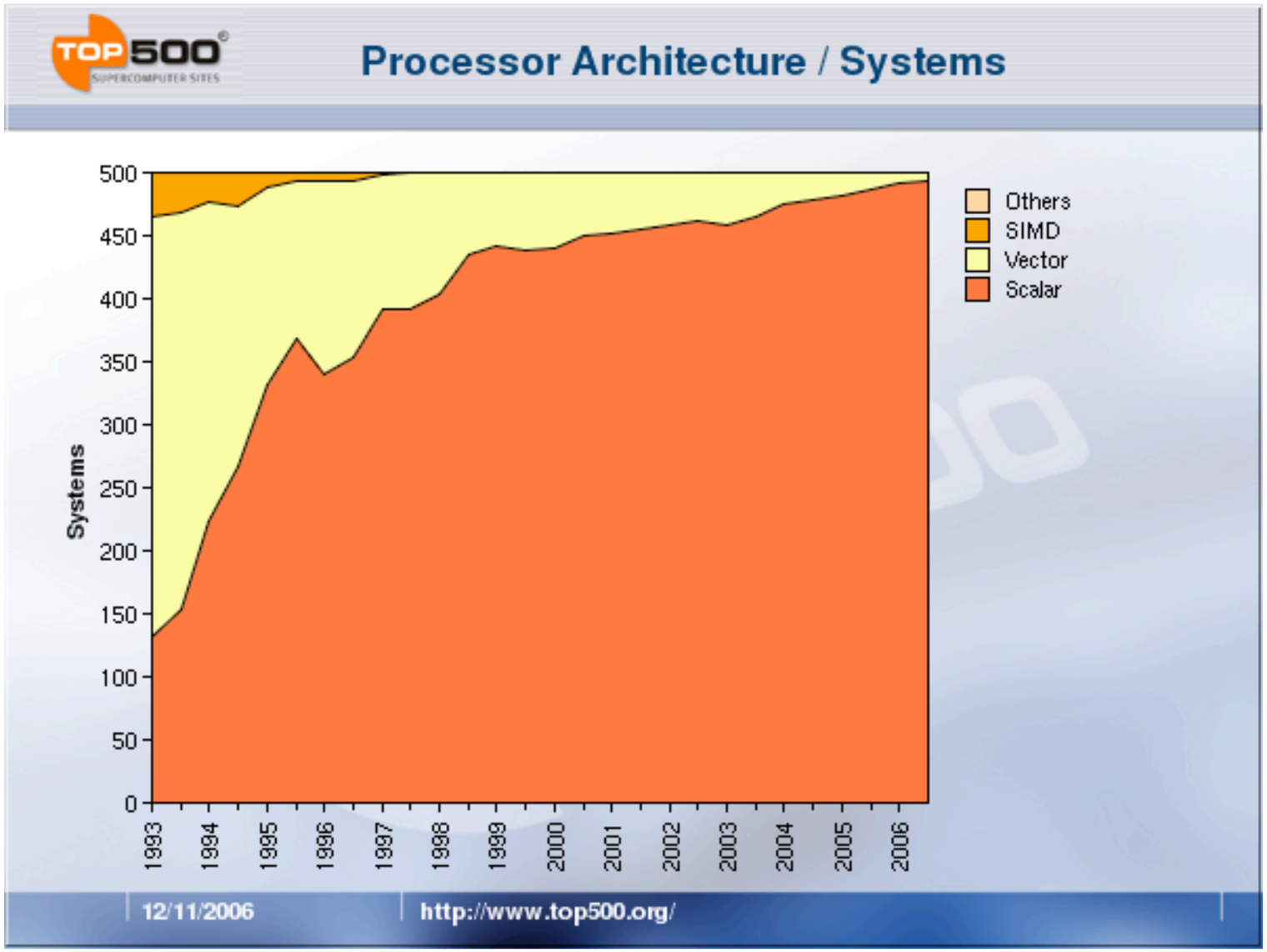
- 2000

Processor Architecture	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
Vector	60	12.00 %	11685	13322	2648
Scalar	440	88.00 %	52545	82354	113010
Totals	**500**	**100%**	**64230.11**	**95676.11**	**115658**

- 2006

Processor Architecture	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
Vector	7	1.40 %	85074	97805	8426
Scalar	493	98.60 %	3442384	5115743	1012535
Totals	**500**	**100%**	**3527458.35**	**5213548.18**	**1020961**

Evolution: processor architecture (top500)



Evolution: interconnection family (top500)

- 1993

Interconnect Family	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
N/A	398	79.60 %	709	1007	22478
Crossbar	63	12.60 %	151	206	44162
Fat Tree	39	7.80 %	263	573	4480
Totals	**500**	**100%**	**1122.84**	**1786.21**	**71120**

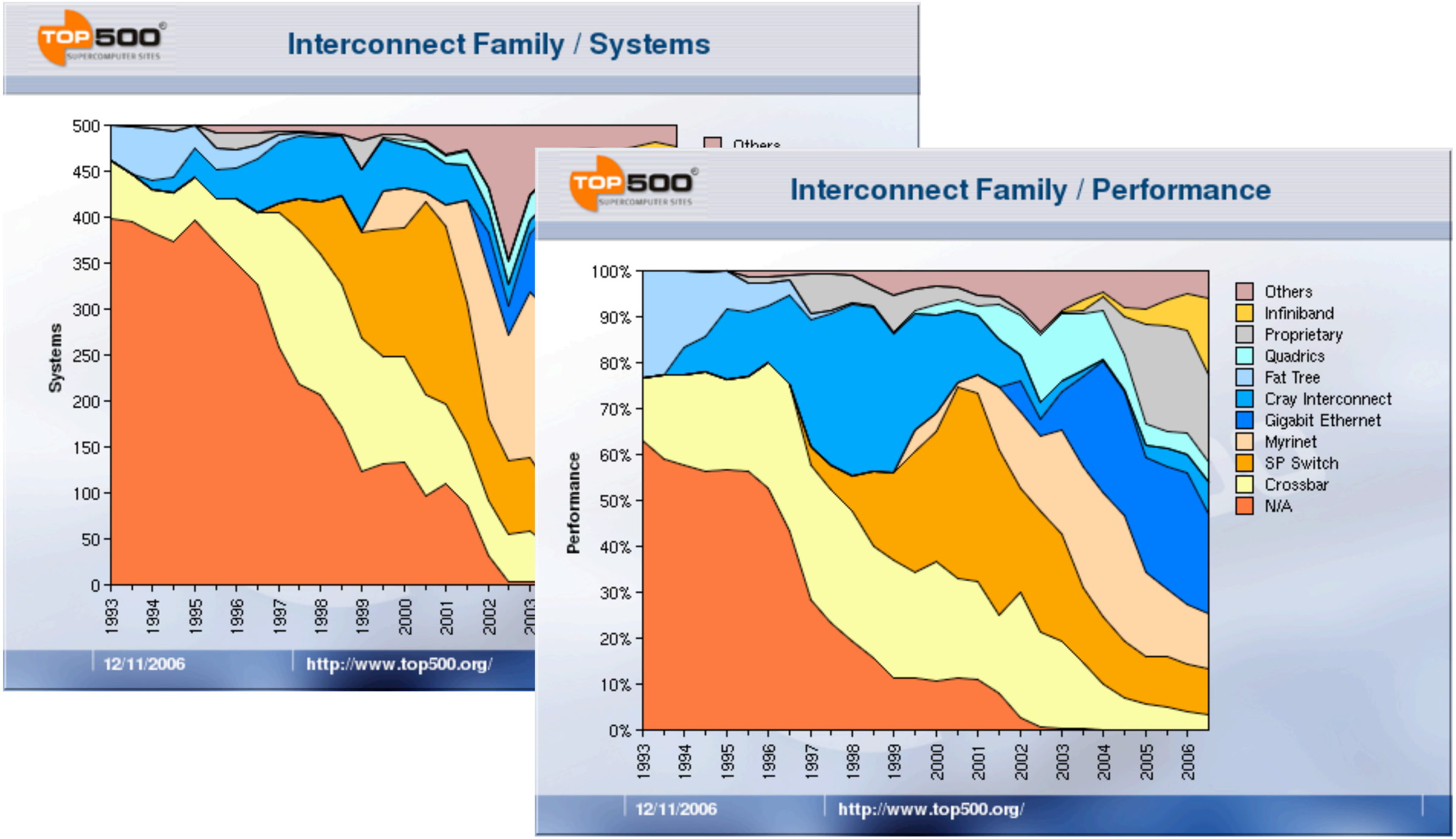
- 2000

Interconnect Family	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
N/A	133	26.60 %	6851	8588	13398
Myrinet	42	8.40 %	2501	5905	3768
Quadrics	5	1.00 %	1535	2101	1576
Ethernet	6	1.20 %	309	726	1376
Fast Ethernet	3	0.60 %	159	418	652
Crossbar	115	23.00 %	16600	19957	17900
SP Switch	141	28.20 %	18327	31268	38728
HIPPI	1	0.20 %	1608	3072	6144
Proprietary	6	1.20 %	2635	3770	9952
Cray Interconnect	48	9.60 %	13706	19869	22164
Totals	**500**	**100%**	**64230.11**	**95676.11**	**115658**

- 2006

Interconnect Family	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
Myrinet	79	15.80 %	420552	642430	110270
Quadrics	14	2.80 %	149888	192668	40420
Gigabit Ethernet	213	42.60 %	765828	1477104	243420
Infiniband	78	15.60 %	592273	841745	122136
Crossbar	11	2.20 %	112242	147150	16426
Mixed	5	1.00 %	74198	106008	16320
NUMAlink	17	3.40 %	130621	142743	22400
SP Switch	42	8.40 %	361555	495078	74916
Proprietary	30	6.00 %	669508	855210	311776
Cray Interconnect	9	1.80 %	244757	305978	61188
RapidArray	2	0.40 %	6037	7435	1689
Totals	**500**	**100%**	**3527458.35**	**5213548.18**	**1020961**

Evolution: interconnection family (top500)



Evolution: operating system (top500)

- 1993

Operating system Family	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
Unix	468	93.60 %	992	1640	71073
BSD Based	23	4.60 %	123	135	38
N/A	9	1.80 %	8	11	9
Totals	**500**	**100%**	**1122.84**	**1786.21**	**71120**

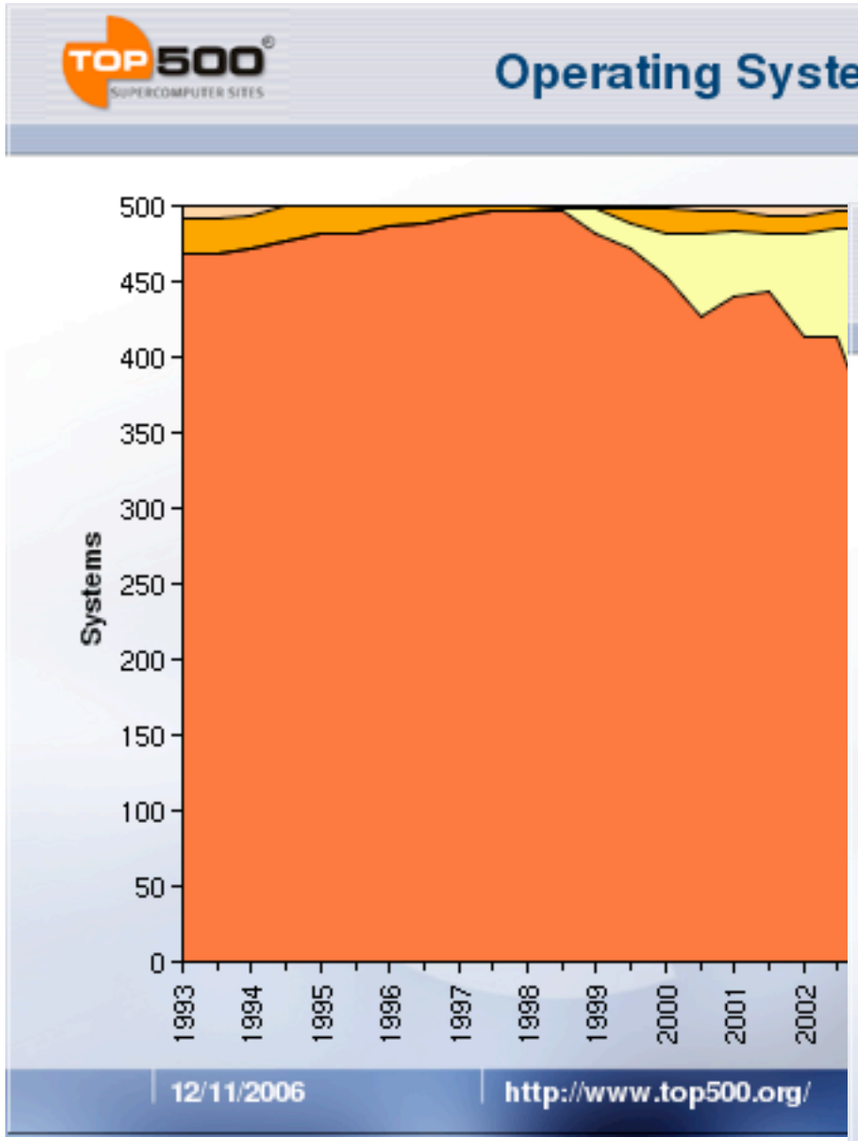
- 2000

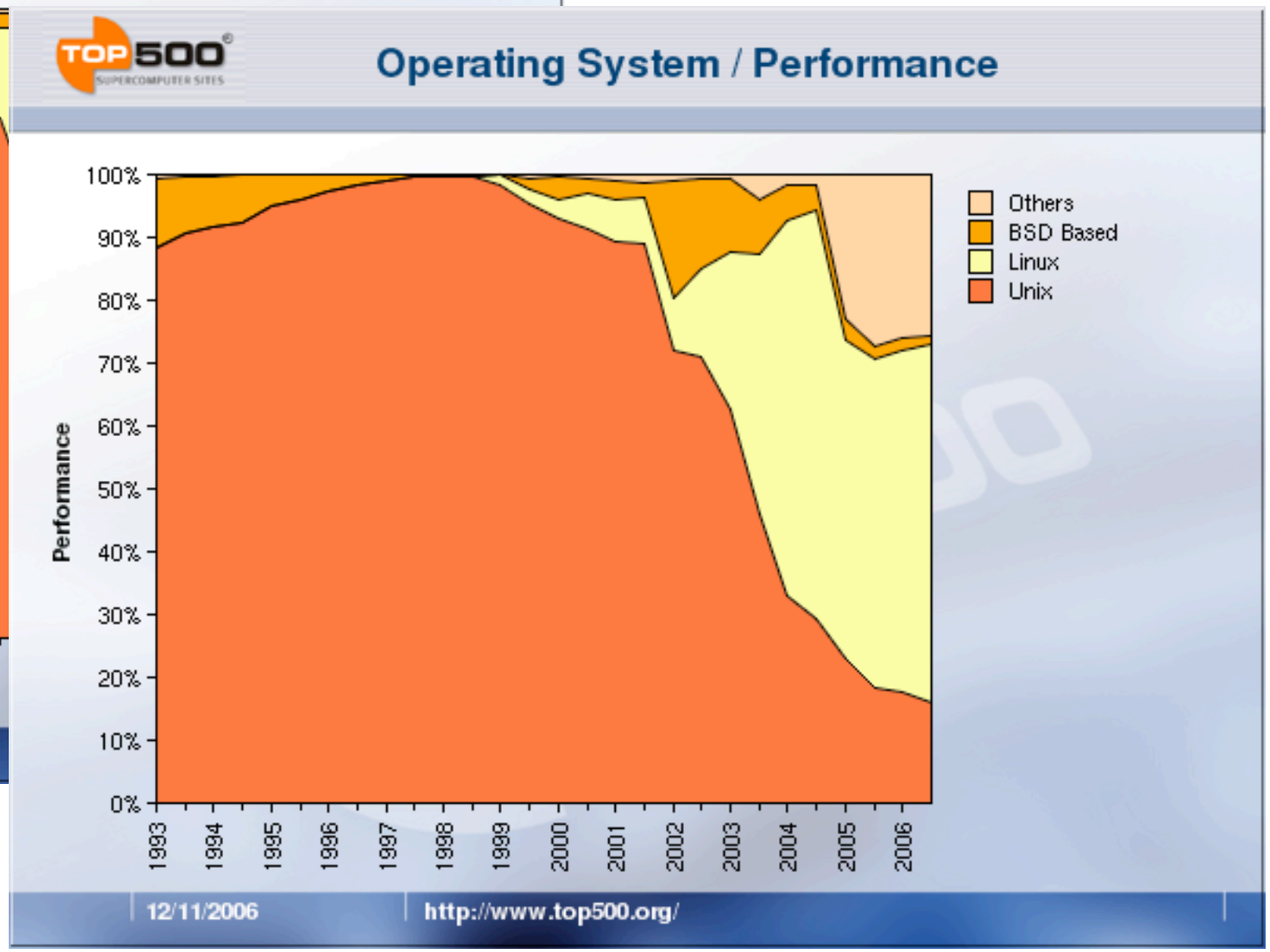
Operating system Family	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
Linux	28	5.60 %	1935	2763	4560
Unix	453	90.60 %	59805	89873	110064
BSD Based	17	3.40 %	2197	2320	314
N/A	2	0.40 %	294	720	720
Totals	**500**	**100%**	**64230.11**	**95676.11**	**115658**

- 2006

Operating system Family	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
Linux	376	75.20 %	2014910	3195766	516189
Unix	86	17.20 %	559636	807423	142104
BSD Based	3	0.60 %	47697	53248	5888
Mixed	32	6.40 %	872226	1104103	350484
Mac OS	3	0.60 %	32989	53008	6296
Totals	**500**	**100%**	**3527458.35**	**5213548.18**	**1020961**

Evolution: operating system (top500)





Evolution: number of processors (top500)

- 1993

Number of Processors	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
1	95	19.00 %	140	176	95
2	72	14.40 %	83	94	144
3-4	98	19.60 %	144	194	380

- 2000

Number of Processors	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
5-8	6	1.20 %	299	320	40
9-16	10	2.00 %	1195	1248	158
17-32	19	3.80 %	2270	2452	550
33-64	161	32.20 %	10680	14405	9887
65-128	119	23.80 %	10392	13880	13356
129-256	104	20.80 %	8508	14660	20141
257-512	43	8.60 %	7899	12753	15632
513-1024	23	4.60 %	8927	13229	15948
1025-2048	11	2.20 %	7825	12468	15860
2049-4096	1	0.20 %	104	125	2502
4k-8k	2	0.40 %	3752	6928	11952
8k-16k	1	0.20 %	2379	3207	9632
Totals	**500**	**100%**	**64230.11**	**95676.11**	**115658**

- 2006

Number of Processors	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
33-64	4	0.80 %	16204	22989	196
65-128	2	0.40 %	18072	21504	160
129-256	3	0.60 %	9630	10854	696
257-512	36	7.20 %	117710	162157	17836
513-1024	192	38.40 %	716626	1144890	171117
1025-2048	185	37.00 %	865405	1423050	262844
2049-4096	38	7.60 %	372432	584622	98884
4k-8k	19	3.80 %	357877	470662	95140
8k-16k	17	3.40 %	542883	717847	159128
16k-32k	2	0.40 %	138730	173286	42928
32k-64k	1	0.20 %	91290	114688	40960
64k-128k	1	0.20 %	280600	367000	131072
Totals	**500**	**100%**	**3527458.35**	**5213548.18**	**1020961**

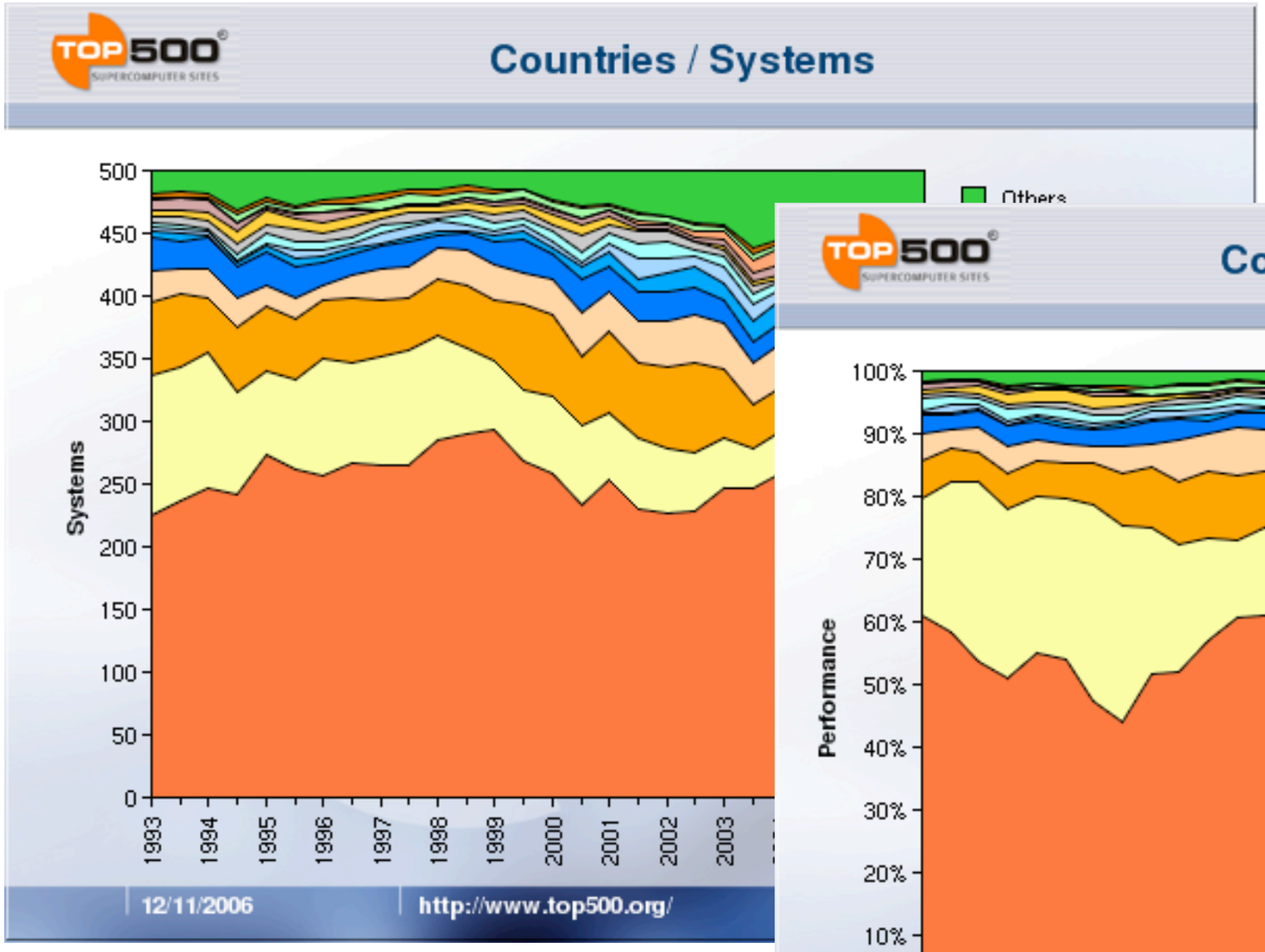
Evolution: country (top500)

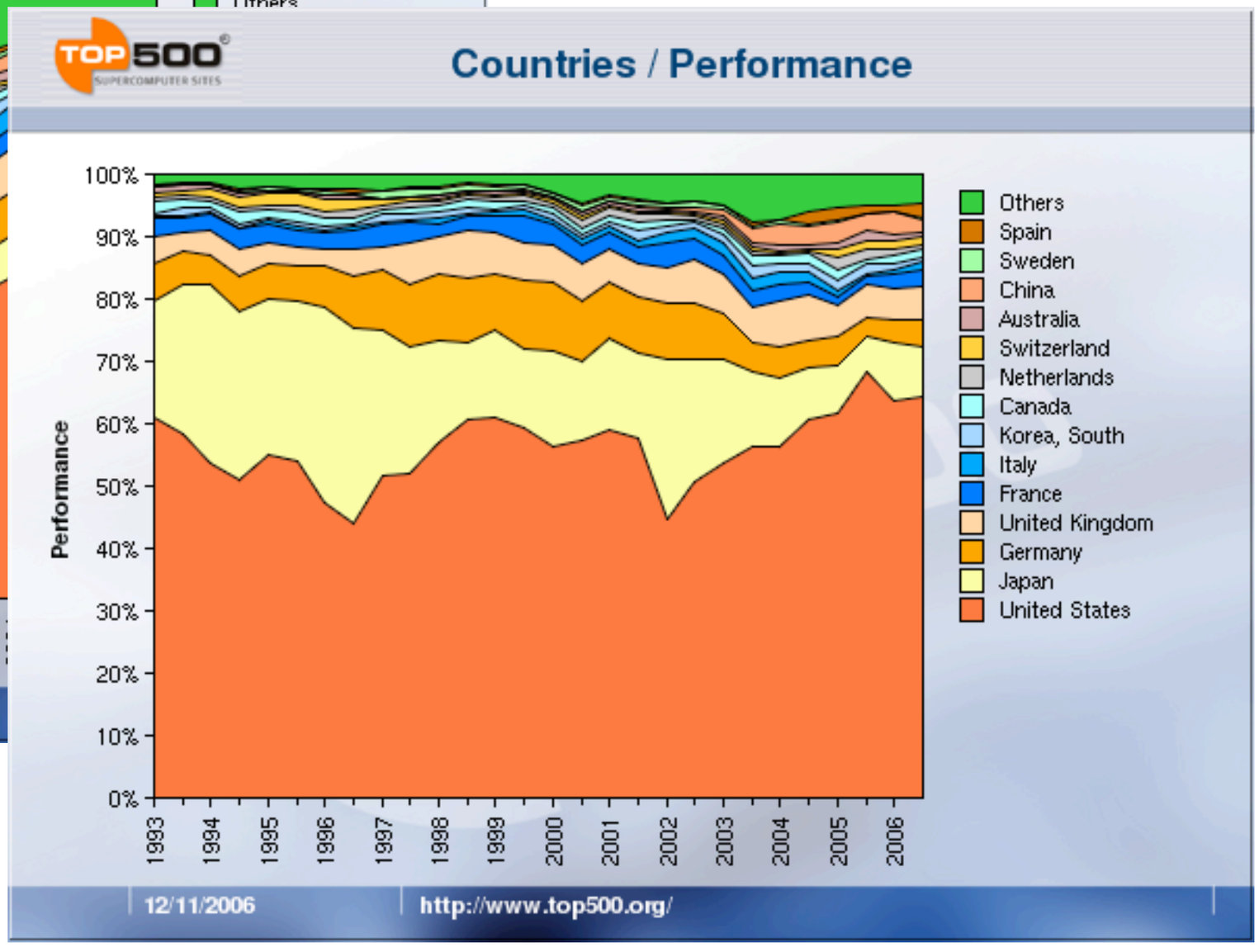
Countries	Count	Share %
Australia	9	1.80 %
Austria	2	0.40 %
Brazil	2	0.40 %
Canada	3	0.60 %
Denmark	3	0.60 %
Finland	2	0.40 %
France	26	5.20 %
Germany	59	11.80 %
Greece	1	0.20 %
Hong Kong	1	0.20 %
Italy	6	1.20 %
Japan	111	22.20 %
Korea, South	3	0.60 %
Mexico	1	0.20 %
Netherlands	6	1.20 %
Norway	3	0.60 %
Slovenia	1	0.20 %
Spain	2	0.40 %
Sweden	2	0.40 %
Switzerland	4	0.80 %
Taiwan	3	0.60 %
United Kingdom	25	5.00 %
United States	225	45.00 %
Totals	**500**	**100%**

Countries	Count	Share %
Australia	3	0.60 %
Brazil	1	0.20 %
Canada	9	1.80 %
China	2	0.40 %
Denmark	1	0.20 %
Finland	2	0.40 %
France	20	4.00 %
Germany	65	13.00 %
Greece	1	0.20 %
Italy	6	1.20 %
Japan	61	12.20 %
Korea, South	4	0.80 %
Luxembourg	6	1.20 %
Mexico	4	0.80 %
Netherlands	5	1.00 %
New Zealand	1	0.20 %
Peru	1	0.20 %
Poland	1	0.20 %
Saudia Arabia	2	0.40 %
Singapore	1	0.20 %
Spain	2	0.40 %
Sweden	5	1.00 %
Switzerland	8	1.60 %
Taiwan	2	0.40 %
United Kingdom	28	5.60 %
United States	259	51.80 %
Totals	**500**	**100%**

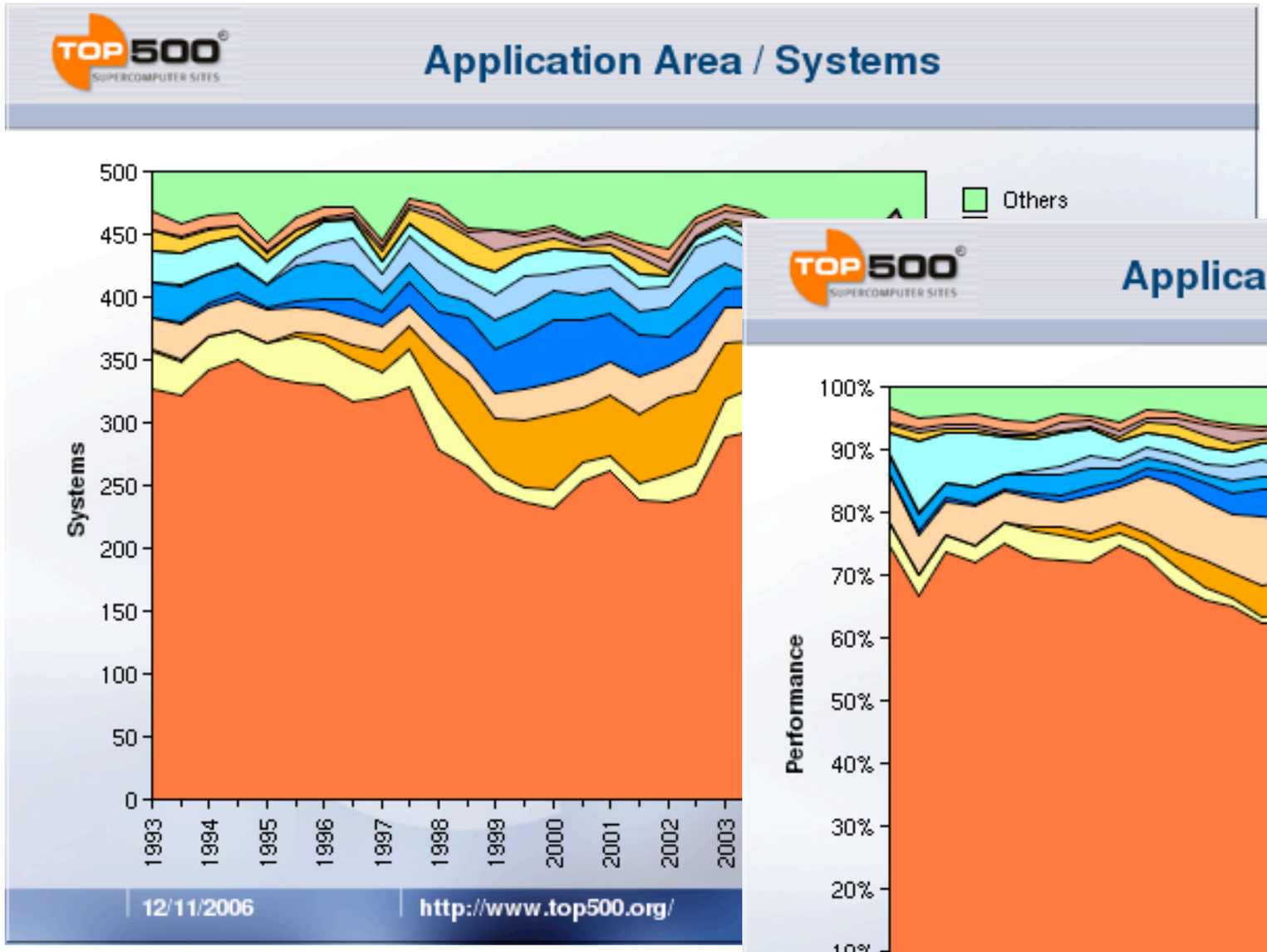
Countries	Count	Share %	Rmax Sum (GF)
Australia	4	0.80 %	20671
Belgium	1	0.20 %	4493
Brazil	4	0.80 %	13668
Canada	8	1.60 %	37317
China	18	3.60 %	72192
Denmark	1	0.20 %	2791
Finland	1	0.20 %	8200
France	12	2.40 %	99870
Germany	18	3.60 %	145407
India	10	2.00 %	34162
Ireland	1	0.20 %	3142
Israel	2	0.40 %	7510
Italy	8	1.60 %	39172
Japan	30	6.00 %	286674
Korea, South	6	1.20 %	33715
Malaysia	3	0.60 %	12125
Mexico	1	0.20 %	5090
Netherlands	2	0.40 %	31114
New Zealand	1	0.20 %	3755
Norway	3	0.60 %	17473
Russia	2	0.40 %	9705
Saudia Arabia	4	0.80 %	11966
Singapore	2	0.40 %	6324
South Africa	2	0.40 %	5696
Spain	7	1.40 %	91600
Sweden	1	0.20 %	4999
Switzerland	5	1.00 %	47682
Taiwan	2	0.40 %	5535
Turkey	1	0.20 %	3288
United Arab Emirates	1	0.20 %	4713
United Kingdom	30	6.00 %	186420
United States	309	61.80 %	2270990
Totals	**500**	**100%**	**3527458.35**

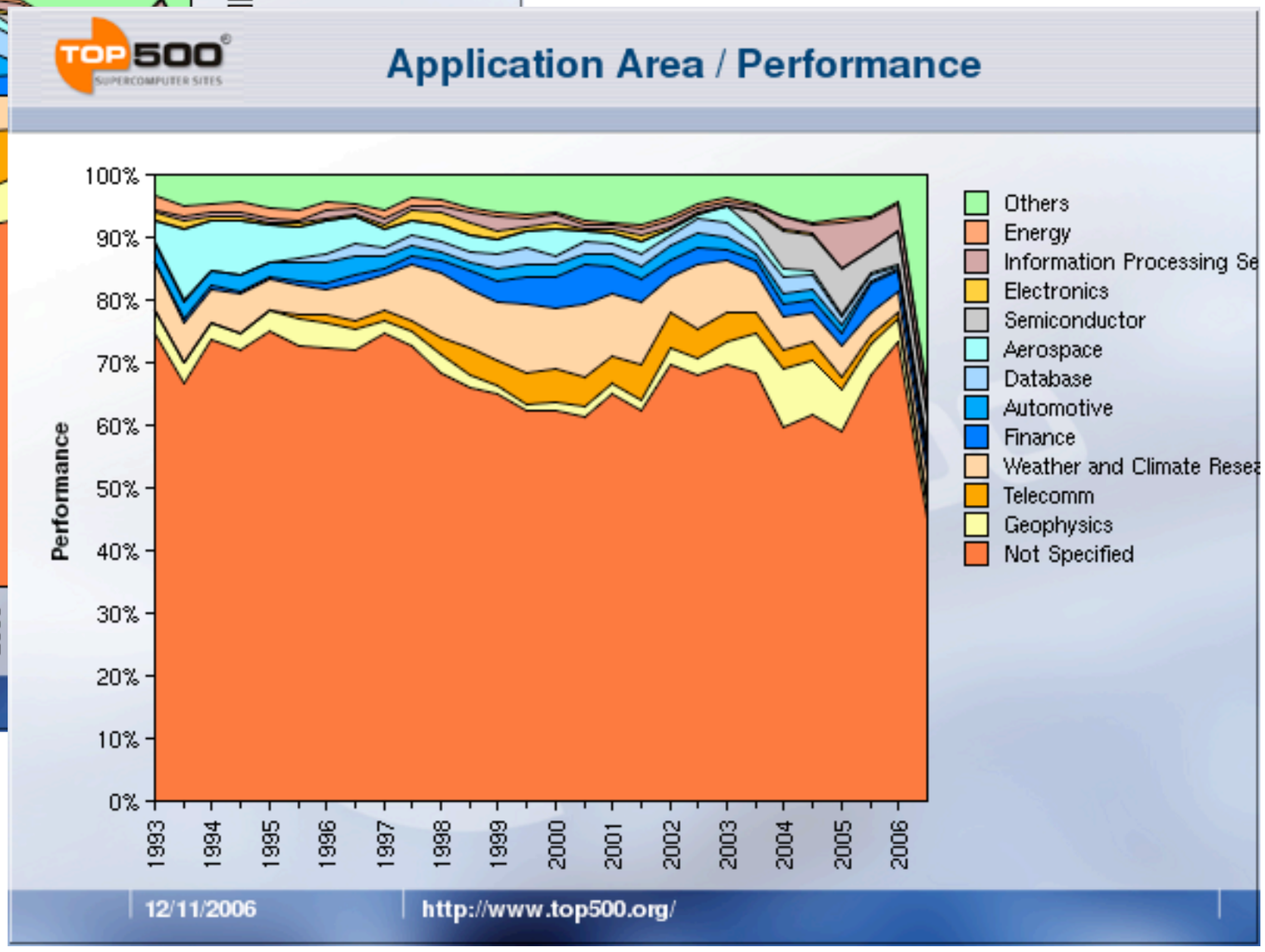
Evolution: country (top500)





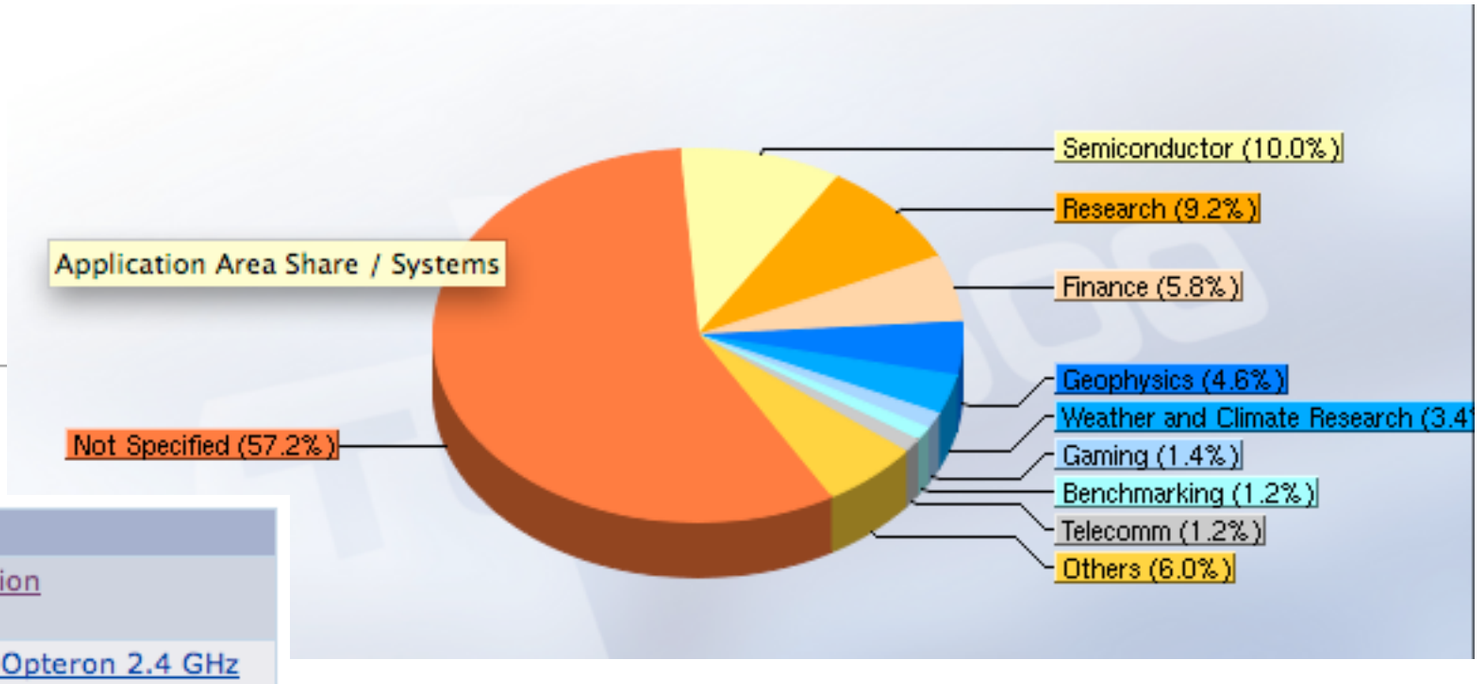
Evolution: applications

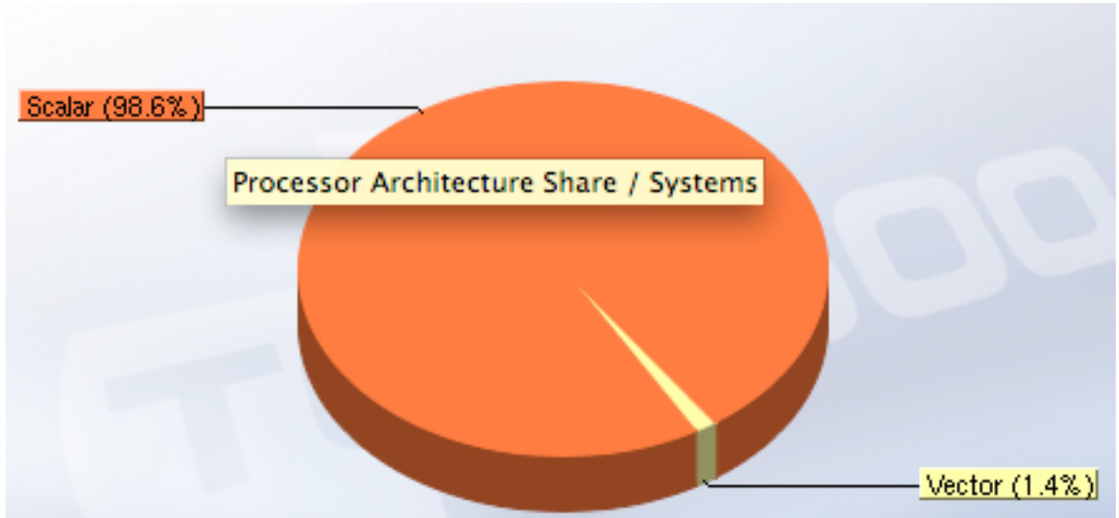


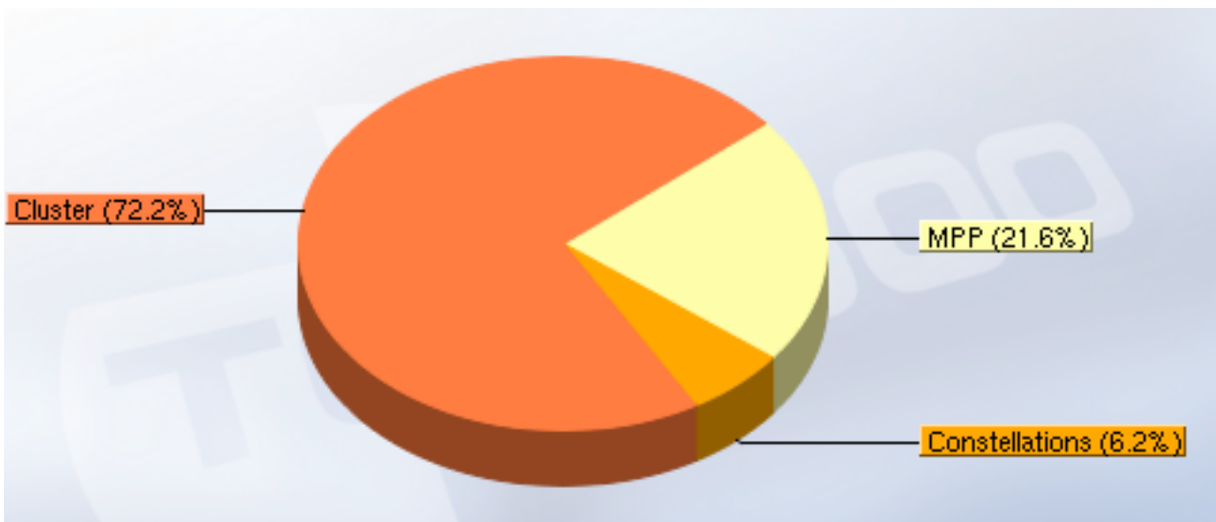


current Top500

Rank	Site	Computer
1	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution IBM
2	NNSA/Sandia National Laboratories United States	Red Storm - Sandia/ Cray Red Storm, Opteron 2.4 GHz dual core Cray Inc.
3	IBM Thomas J. Watson Research Center United States	BGW - eServer Blue Gene Solution IBM
4	DOE/NNSA/LLNL United States	ASC Purple - eServer pSeries p5 575 1.9 GHz IBM
5	Barcelona Supercomputing Center Spain	MareNostrum - BladeCenter JS21 Cluster, PPC 970, 2.3 GHz, Myrinet IBM
6	NNSA/Sandia National Laboratories United States	Thunderbird - PowerEdge 1850, 3.6 GHz, Infiniband Dell
7	Commissariat a l'Energie Atomique (CEA) France	Tera-10 - NovaScale 5160, Itanium2 1.6 GHz, Quadrics Bull SA
8	NASA/Ames Research Center/NAS United States	Columbia - SGI Altix 1.5 GHz, Voltaire Infiniband SGI
9	GSIC Center, Tokyo Institute of Technology Japan	TSUBAME Grid Cluster - Sun Fire x4600 Cluster, Opteron 2.4/2.6 GHz and ClearSpeed Accelerator, Infiniband NEC/Sun
10	Oak Ridge National Laboratory United States	Jaguar - Cray XT3, 2.6 GHz dual Core Cray Inc.







IBM Blue Gene

Performance	Peak performance per rack—5.73 teraflops Linpack performance per rack—4.71 teraflops	Highest available performance benefits capability customers
Power	27.6 kW power consumption per rack (maximum) 7 kW power consumption per rack (idle) 208 VAC 3-phase; 100 amp service per rack	Low power draw enables dense packaging
Cooling	Air conditioning 8 tons/rack (minimum) 2800 CFM (compute rack); 350 CFM (power supplies)	Low cooling requirements enable extreme scale-up
Acoustics	9.0 LwAD and 8.7 LwAm	
Dimensions (include air duct)	Height—77" Width—36" Depth—36" Weight—1810 lbs Service clearances—30" front and back Raised floor height—16" minimum	Design allows dense floor plan layout for better floor space utilization

IBM System: Blue Gene Solution

Blue Gene At a Glance

Attribute	Description	Benefit
Processor	PowerPC 440 700 MHz; two per node	Low power allows dense packaging; better processor-memory balance
Memory per node	512MB SDRAM-DDR (Model 0203-700) 1 GB SDRAM-DDR (Model 0203-900)	
Networks	1) 3D Torus - 175 MB/sec in each direction 2) Collective Network-350 MB/sec; 1.5 µsec latency 3) Global Barrier/Interrupt 4) Gigabit Ethernet (I/O & connectivity) 5) Control (system boot, debug, monitoring)	Special networks speed up internode communications; designed for MPI programming constructs; improve systems management
Computer nodes	Dual processor; 1024 per rack	Double FPU improves performance
I/O nodes	Dual processor; 16-128 per rack	Facilitates job launch and I/O, raising efficiency of compute nodes
Operating Systems	Compute Node—Lightweight proprietary kernel I/O Node—Embedded Linux Front End and Service Nodes—SUSE SLES 9 Linux	Kernel tailored to processor design; industry-standard distribution on front-end and service nodes preserves familiarity to end users and administrators

What's up in Italy ?

Home » Database » **Sublists**

Sublists

R_{max} and **R_{peak}** values are in GFlops. For more details about other fields, check the TOP500 description.

For more information about the system and site, click the respective links in the table

8 entries found.

Rank	Site	System	Processors	R_{max}	R_{peak}
44	CINECA Italy	eServer 326 Cluster, Opteron Dual Core 2.6 GHz, Infiniband IBM	5120	12608	26624
84	SCS S.r.l. Italy	ProLiant BL460c EM64T Xeon 51xx 3GHz Hewlett-Packard	1024	7987.2	12288
340	CINECA Italy	eServer pSeries p5 575 1.9 GHz IBM	512	3392	3891.2
347	CINECA Italy	xSeries, Xeon 3.06 GHz, Myrinet IBM	1024	3328	6266.88
354	Alenia Aeronautica/Quadrics Italy	Cluster Platform 4000 DL145 G2 Opteron Dual Core 2.6 GHz Quadrics Hewlett-Packard	800	3286	4160
433	CINECA Italy	BladeCenter LS20, Opteron 2.2 GHz Dual core, Infiniband IBM	1064	2874.54	4681.6
471	Telecom Italia Italy	SuperDome 875 MHz/HyperPlex Hewlett-Packard	1536	2848	5376
472	Telecom Italia Italy	SuperDome 875 MHz/HyperPlex Hewlett-Packard	1536	2848	5376

“Desk processor”

- http://www.intel.com/pre
- 1993
- 2000
- 2006

Desktop

Processor	Clock Speed(s)	Intro Date(s)	Mfg. Process/	Typical Use

Intel® Pentium® III Xeon™ Processor

Processor	Clock Speed(s)	Intro Date(s)	Mfg. Process/ Transistors	Cache	Addressable Memory	Bus Speed	Typical Use
Intel® Pentium® III Xeon™ Processor	900 MHz	Mar. 21, 2001	0.18-micron 28 million	2 MB Advanced Transfer L2 cache	64 GB	100 MHz	High-end servers, 4- and 8-way multiprocessing systems
Intel® Pentium® III Xeon™ Processor	933 MHz	May 24, 2000	0.18-micron 28 million	256 KB Advanced Transfer L2 cache	64 GB	133 MHz	Business and consumer PCs, 1- and 2-way servers and

1993

March 22, 1993
Intel® Pentium® Processc
66 MHz, 60 MHz

February 14,
Mobile Intel®
500 MHz, 450

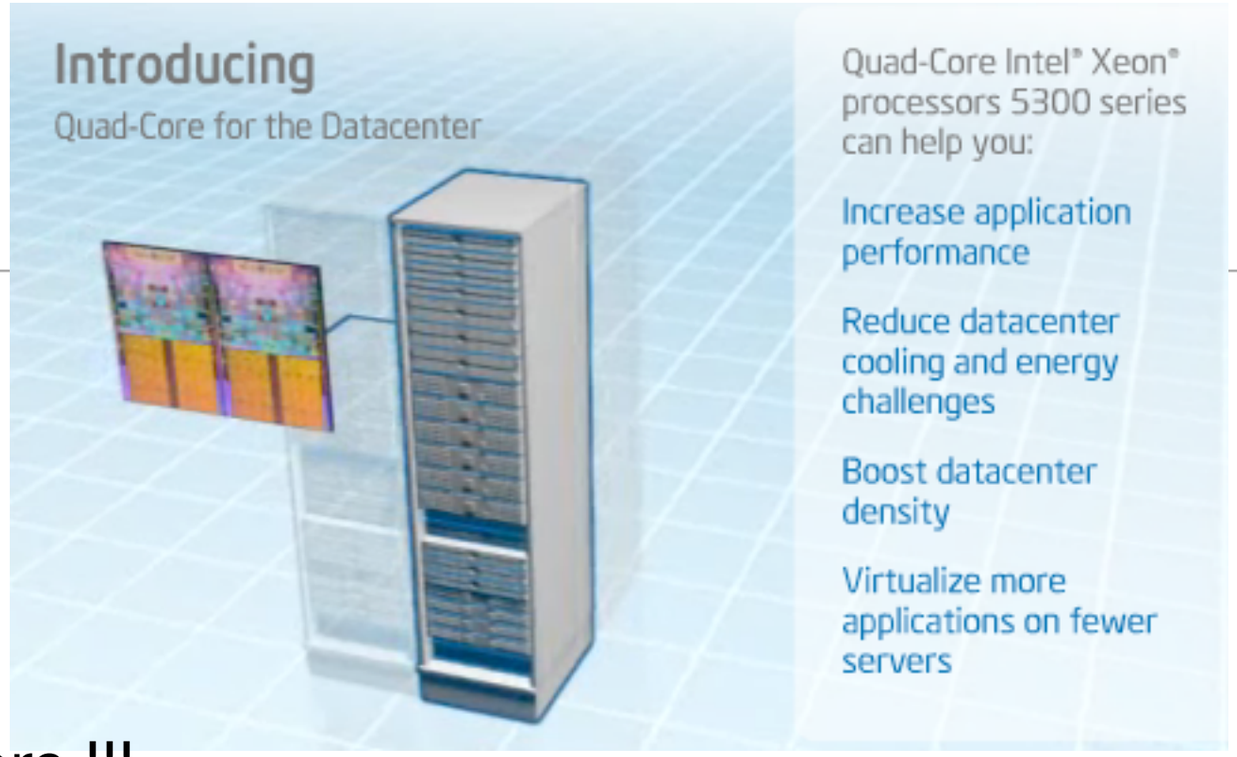
January 18, 2
Mobile Intel®
650 MHz, 600

January 12, 2
Intel® Pentiun
800 MHz

Intel® Co
Great computin

	Intel® Itanium® 2 Processor Remove	Intel® Itanium® 2 Processor Low Voltage Remove
Clock Speed	1.66 GHz	1.30 GHz
Front Side Bus Speed	667 MHz	400 MHz
L3 Cache	9MB	3MB
L2 Cache	256KB	256KB
L1 Cache	32KB	32KB
Power	130 watts	62 watts
System Type	MP	DP
Architecture	.13 micron	.13 micron
Other Intel Technologies	EPIC, Machine Check Architecture	EPIC, Machine Check Architecture
Package	PAC-611	PAC-611
Server Chipset	Intel® E8870 Chipset, OEM custom chipset	Intel® E8870 Chipset, OEM custom chipset
Memory Type	DDR, SDRAM	DDR, SDRAM
Server Platforms	Intel® Server Platform SR870BN4	Intel® Server Platform SR870BH2

Then ?



- Quad core available 5300

- Proof of concept per 80-core !!!

Intel fabs 80-core teraflop processor

By Tony Smith in San Francisco [More by this author]
26th September 2006 20:44 GMT

Receive the days biggest stories by email, sign up here

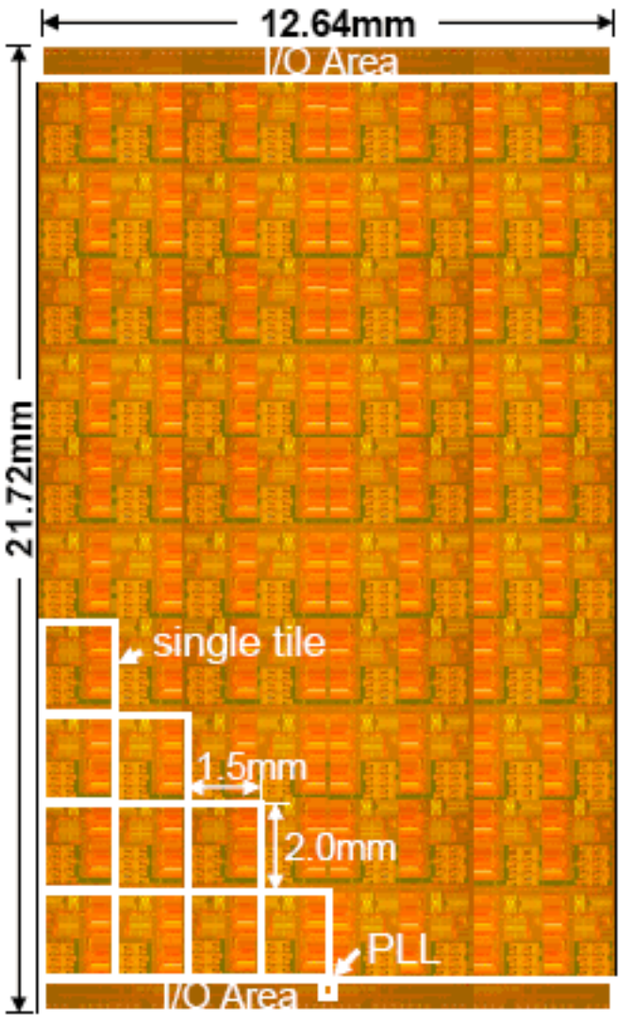
IDF Quad-core? Pah! Intel has produced an 80-core chip, the world's first programmable microprocessor with teraflop performance capabilities, the chip giant claimed today. It's not compatible with the x86 instruction set - it's a proof of concept part designed to show how a production processor might operate.

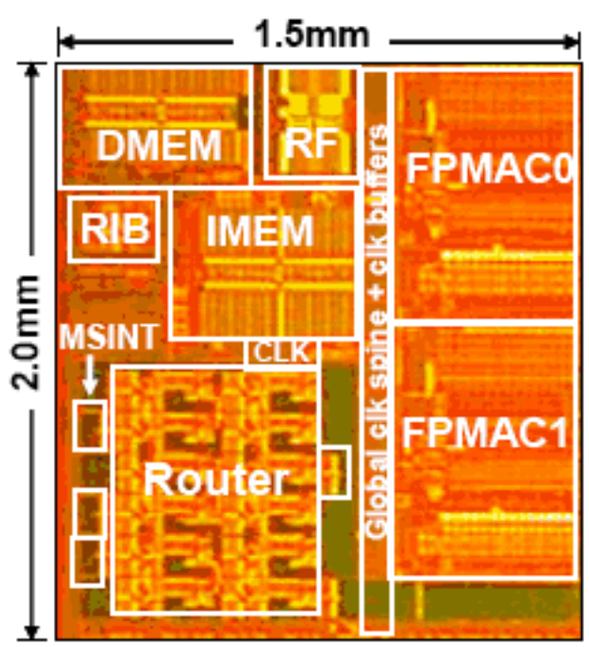
The monster part incorporates not only the usual data-processing facilities - essentially they're just floating point maths co-processors - but also features a network processing unit on each core to control core-to-core communication. The cores are linked in a mesh configuration.

Each core's designed to be clocked to 3.1GHz and is mounted with 20MB of SRAM stacked up on top of the die. Connecting memory this way provides an aggregate bandwidth of a trillion bytes per second, Intel said.

80 core Intel proof of concept

- 4Ghz chip with mesh (logical and physical) 10x8 core FP, 1,28 TFlops
- Tile:
  - router: addesses each core on chip, implements the mesh
- VLIW processor (96 bit x instruction, up to 8 ops per cycle), in-order-execution, 32 registers (6Read/4Write), 2K Data, 3K Instruction cache, 2 FPU (9 stages, 2FLOPs/cycle sustained),
  - Cicli: FPU:9, Ld/St:2, Snd/ Rcv:2, Jmp/Br:1





Technology	65nm CMOS Process
Interconnect	1 poly, 8 metal (Cu)
Transistors	100 Million
Die Area	275mm²
Tile area	3mm²
Package	1248 pin LGA, 14 layers, 343 signal pins

Intel trends

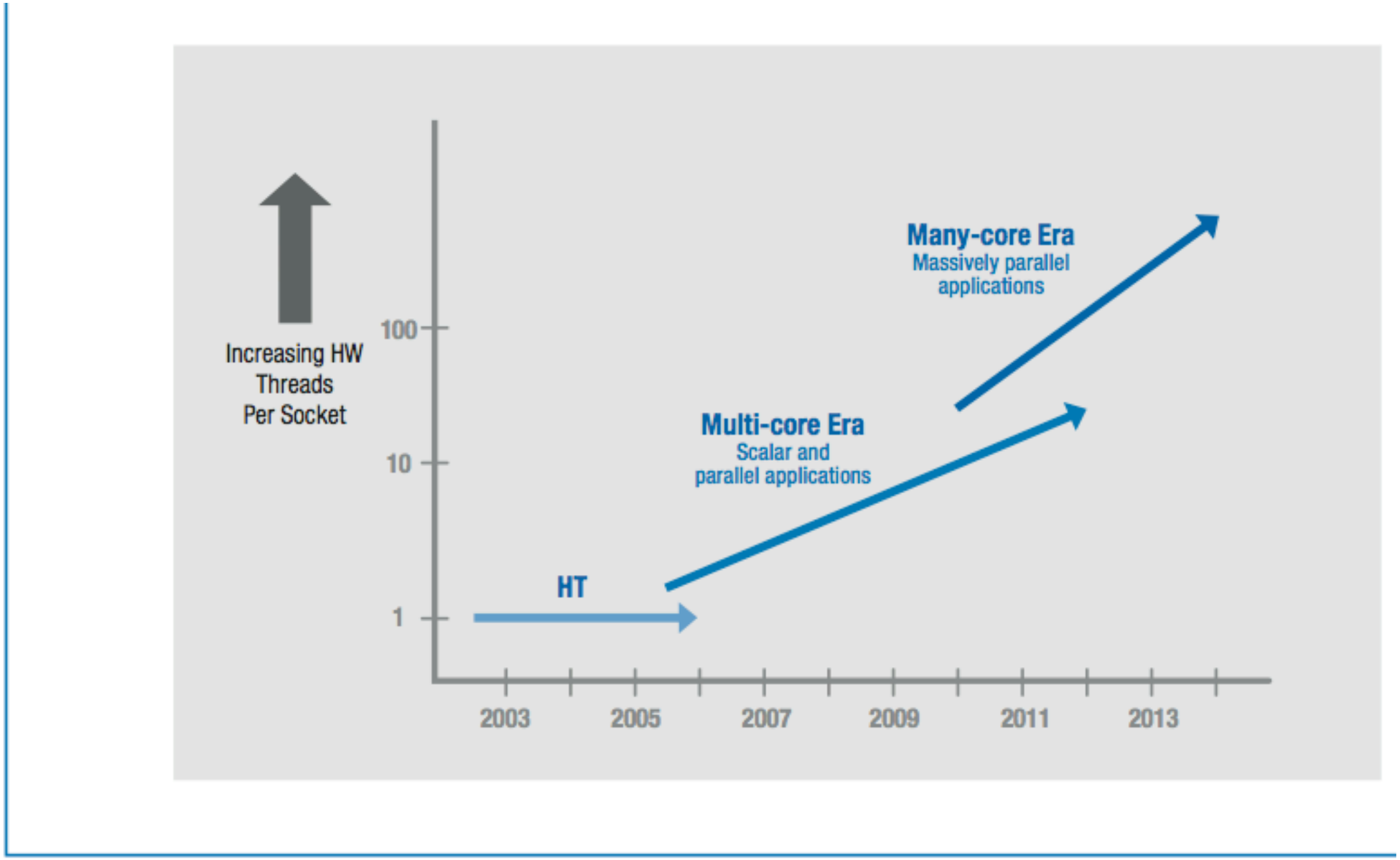


Figure 1: Current and expected eras of Intel® processor architectures

Intel trends (2)

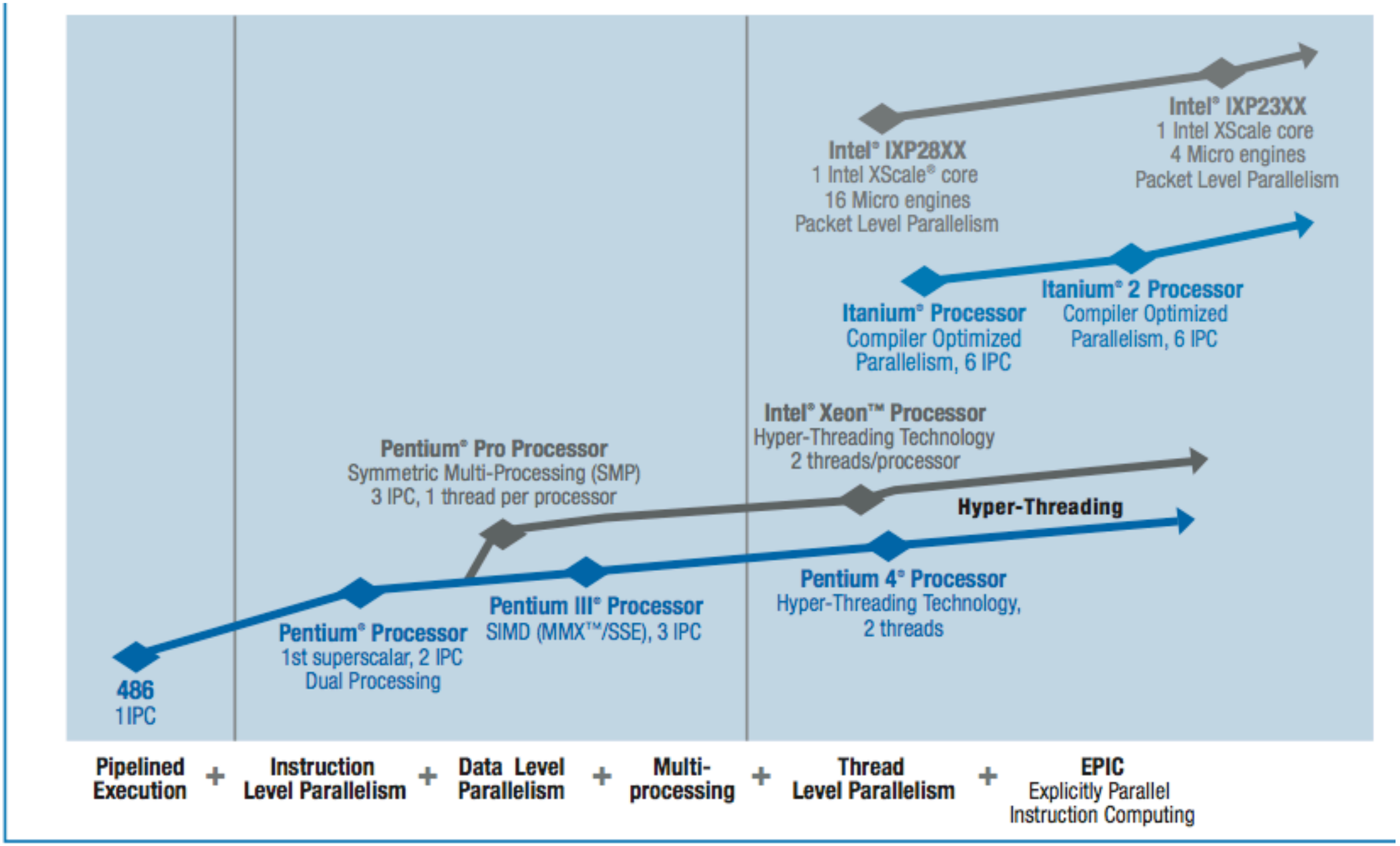


Figure 2: Driving increasing degrees of parallelism on Intel® processor architectures

Programming models ?

- *message passing*
  - communications
  - remote procedure/method call
- *shared memory*
  - binding for different programming languages (C, C++, F77)
  - different implementations (open source, proprietary)
  - different performance
  - *all responsabilies completely in charge of the programmer*

“Production” programming models

- Pure message passing
  - PVM 1989 (Univ. Tennessy, Emory and Oak Ridge Nat. Lab.)
  - MPI (Draft 1.0 1994)
    - SPMD
- RPC
  - Sun RPC, Java RMI, CORBA,
- Shared memory
  - HPF (1993)
  - OpenMP (1997)

“Research” programming models

- Skeleton (1989)
- Parallel design patterns (metà anni ‘90)
- Coordination languages
  - Linda (1992): tuple space
  - Manifold (ultimi ‘90): control oriented, strongly typed, block structured, event-driven
- Component programming models
  - CCA
  - CCM

The Pisa experience

- Skeleton
  - P3L (1990)
- Skeleton + coordination + components
  - ASSIST (2000)
- Skeleton + macro data flow
  - Muskel (2000)

Compute *science*

Definitions

- McDaniel, George, ed. IBM Dictionary of Computing. New York, NY: McGraw-Hill, Inc., 1994.
  - *Parallel computing* a computer system in which interconnected processors perform concurrent or simultaneous execution of two or more processes
- *Dicotomy:*
  - parallel vs. distributed
- Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990
  - *Distributed computing* a computer system in which several interconnected computers share the computing tasks assigned to the system

Quantitative aspects

- characterising parallel/distributed computations
- through quantitative parameters
    - Completion time & Service time
    - Speedup / Scalability / Efficiency
    - Communication time

Times

- Completion time
  - time spent from computation init to computation end
  - includes all the overhead

$$T_c$$

- Service time
  - time spent to deliver next item

$$T_s$$

Speedup

- Speedup

$$\text{speedup(n)} = \frac{miglior\ tempo\ sequenziale}{tempo\ su\ n\ processori}$$

$$sp(n) = \frac{T_{seq}}{T(n)}$$

- superlinear speedup
  - e.g. due to better usage of cache
- in the general case, must be no more than linear

Amdahl law

- serial fraction of the algorithm bounds speedup

$$\text{speedup(n)} = \frac{t_{seq}}{f t_{seq} + (1-f) t_{seq}/p} = \frac{p}{fp + (1-f)} = \frac{p}{1 + (p-1)f}$$

$$\lim_{n \to \infty} speedup(n) = \frac{1}{f}$$

- serial fraction = 1% , then
  - max speedup = 100 !

Efficiency

- Efficiency

$$\epsilon(n) = \frac{sp(n)}{n} = \frac{T_{seq}}{n \times T(N)}$$

- Alternatively:

$$\epsilon(n) = \frac{1}{1+\frac{T_{ov}}{T_{seq}}}$$

$$nT_n = T_{seq} + T_{ov}$$

$$T_n = \frac{T_{seq}+T_{ov}}{n}$$

$$speedup(n) = \frac{T_{seq}}{T_n} = \frac{T_{seq}}{\frac{T_{seq}+T_{ov}}{n}} = \frac{pT_{seq}}{T_{seq}+T_{ov}}$$

$$\epsilon(n) = \frac{T_{seq}}{T_{seq}+T_{ov}} = \frac{1}{1+\frac{T_{ov}}{T_{seq}}}$$

Isoefficiency

- determines scalability of an application onto a given architecture ...
- $\epsilon(n) = \frac{1}{1+\frac{T_{ov}}{T_{seq}}}$

$$\frac{1}{\epsilon(n)} = 1 + \frac{T_{ov}}{T_{seq}}$$

$$\frac{T_{ov}}{T_{seq}} = \frac{1}{\epsilon(n)} - 1 = \frac{(1-\epsilon(n))}{\epsilon(n)}$$

$$T_{seq} = Wt_c$$

$$\frac{T_{ov}}{Wt_c} = \frac{(1-\epsilon(n))}{\epsilon(n)}$$

$$W = \frac{1}{t_c}(\frac{\epsilon(n)}{1-\epsilon(n)})T_{ov} = KT_{ov}$$

- overhead function of n tells how to scale problem dimension to keep efficient constant (not possible in some cases!)

Scalability

- similar to speedup
  - measured on T(1)
  - $scalab(n) = \frac{T(1)}{T(n)}$
- measures somehow the efficiency “in insulation”
- in some cases it is not significant (bad T(1) case ...)

Comm time

- Simple model

$$T_{comm} = t_0 + dt_1$$

- Comm network impact
  - comm coprocessor
    - reduces t_0
    - better bandwidth (smaller t_1)
    - smaller effect of “packetization”

The problem

- Definition: Parallel computing is the use of two or more processors (computers) in combination to solve a single problem. The programmer has to figure out how to break the problem into pieces, and has to figure out how the pieces relate to each other. For example, a parallel program to play chess might look at all the possible first moves it could make. Each different first move could be explored by a different processor, to see how the game would continue from that point. At the end, these results have to be combined to figure out which is the best first move. Actually, the situation is even more complicated, because if the program is looking ahead several moves, then different starts can end up at the same board position. To be efficient, the program would have to keep track of this, so that if one processor had already evaluated that position, then others would not waste time duplicating the effort. This is how must parallel chess-playing systems work, including the famous IBM Deep Blue machine that beat Kasparav.
(http://www.eecs.umich.edu/~qstout/parallel.html)

Parallel programming environments

- DEF “classical” environment:
  - qualitative and quantitative details related to parallelism exploitation in charge to the programmer
  - but also: mechanisms to implement parallelism in full charge to the programmer

- DEF “structured” environment:
  - qualitative aspects in charge to the programmer
  - quantitative aspects and mechanisms in charge to the system (designer)

In general:

- handle more control flows (processes, threads, ...)
    - define them: which ones? how much?
    - mapping: where?
    - scheduling: when?
- interactions among control flows (comms, sharing) ambiente locale o globale?
    - comm grain ?
- and what about I/O ?

Define control flows

- which are concurrent activities
  - implications on parallleism and therefore on speedup/efficiency
  - implications on communications
  - implications on synchronization

- Simple case: enhance image quality, stream of images ...
  - each image can be processed independently
    - file accesses (read/write) =>
      - comms, synchro ... overhead

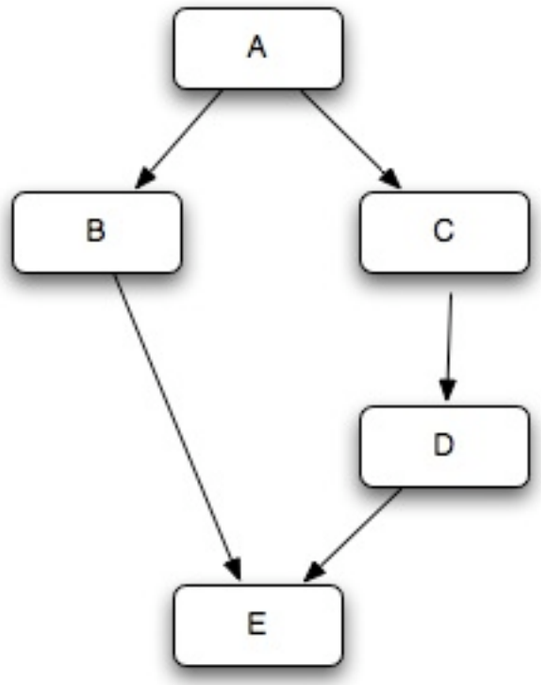
Mapping

- strong dependency on architecture
  - homogeneous resources
    - SMP
    - local network (TCP/IP) of homogeneous resources
  - heterogeneous resources
    - distributed/geographic network
    - hypercube, mesh, fat tree architecture ...
- “contiguous” allocation helps comms and synchro ...

Scheduling

- usually: dependency graph
  - execution order
  - “workflow” problem
    - single shot execution
- different problems when processing streams

Example



- one shot execution
  - schedule A, then B and C, then D and eventually E
  - proper choice of the critical resources
    - e.g. A -> C -> D -> E is the critical path
      - comm B -> E can be longer ... B scheduled on a far node anc A, C, D and E on the same node
- stream execution
  - different nodes to exploit pipeline parallelism

Communications

- synchronous and asynchronous
  - often can both be used in the same situation
  - synchronous comms => “more natural” control flow
    - unless there are very loose time constrains
  - asycnhronous comms => more efficient in general
    - problem: comm completion probe
    - problem: error handling
- Symmetric and asymmetric comms
  - implications on the control flow

Synchronization

- needed to guarantee correctness
  - alternative mechanisms (data flow tagged token for loops)
- heavy consequences if implementation is not efficient
  - block, delay of all the associated actitivies
- implementation “cares” about activity mapping
  - as in comms

Processi

- entità a grana grossa
  - “programma in esecuzione”
  - life cycle:
    - creazione: allocazione di tutte le risorse, caricamento del codice, schedulazione
    - esecuzione: schedulazione
    - terminazione: rilascio delle risorse

Thread

- entità a grana più limitata
  - “flusso di controllo indipendente *all’interno* di un programma”
  - life cycle:
    - creazione: schedulazione
    - esecuzione: accesso a risorse condivise (memoria)
    - terminazione: modello fork/join, cobegin/coend

Computation grain

- activities executed on a different processing element

$$g = \frac{T_{CalcoloRemoto}}{T_{Comunicazione}}$$

  - RemoteComputationTime: *time spent on the remote node to compute the result out of the input data*
  - CommTime: *time spent to send input data to the remote node plus the time spent to send results from the remote node to the current one (possibly including serialization time)*

- grain and efficiency:
  - larger grain = *better efficiency*

Computation grain and comms

- impact of communication mechanism on computation grain
  - same architecture with
    - Socket RAW
    - Java RMI
    - WS (serializzazione XML)
  - grain smaller and smaller
- if $g < 1$
  - serial computation is better than parallel one

Computation grain and sharing

- same problems as in the explicit comm case
- hidden in shared variable access
  - with transparent sharing
  - or through ad hoc primitives (read/write, get/put, ...)
- usually:
  - shared memory programming has a finer grain than message passing
    - it’s a programmer fault, of course :-)

Complexities

- Just one case: mapping
  - G program graph
  - M machine graph
  - find an optima mapping of G onto M (optimality w.r.t. a cost fucntion F)
    - NP complete for arbitrary graphs
      - non-approximable

- most problems are “hard”
  - solved by using heuristics and “scope” restriction

More problems ...

- Load balancing

- Reliability (fault tolerance)

- Security

- Autonomic control

Load balancing

- resource partitioning
    - (non) homogeneous
- computation partitioning
    - (non) homogeneous
- in both case: static or dynamic
    - resource available change (or the load changes)
    - multigrid algorithms (static or dynamic)

Reliability

- fault tollerance (hardware & software)
  - checkpointing
  - forward recovery

- problems related to dynamic execution environments
  - “adaptivity”

Security

- authentication
    - user rights / user classes
- sensible data handling
- code handling (maybe legacy)
- non intrusiveness

Autonomic control

- program in the application code
  - **self configuration**
    - adapt to the environment, ...
  - **self healing**
    - handle faults, ..
- **self optimisation**
  - exploit peculiar features, ...
- **self protection**
  - handle security, ...