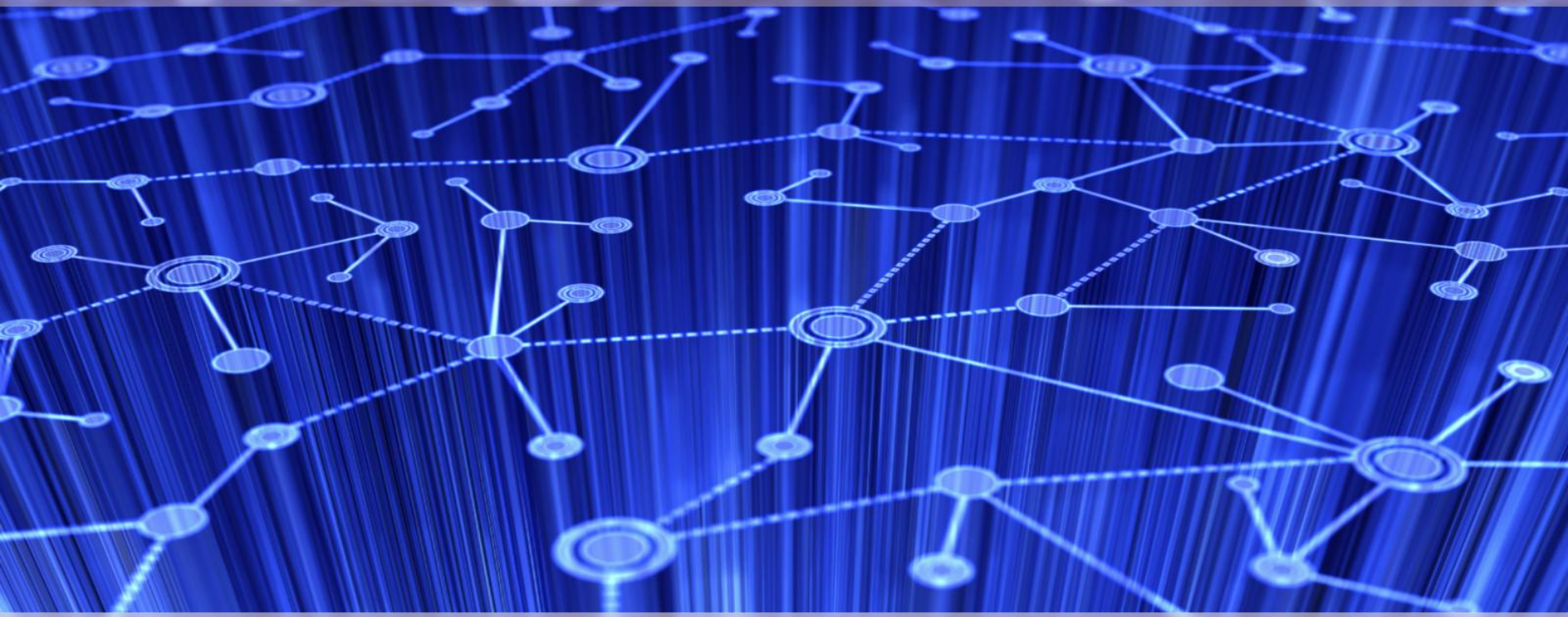


Foundations of High Performance Computing

Approaching the exa-scale era

Luca Tornatore – I.N.A.F

Why this talk ?

Data-intensive *and* Computation-intensive applications

will be **forced** to face an epocal **major shift**

in the computation paradigm,

which indeed started several years ago.

Why this talk ?

"CRUCIAL PROBLEMS that we can only hope to address computationally REQUIRE US TO DELIVER EFFECTIVE COMPUTING POWER ORDERS-OF-MAGNITUDE GREATER THAN WE CAN DEPLOY TODAY."

DOE's Office of Science, 2012

Exa-scale, a minimal definition

- A *lot* of flops
- With *very* little power consumption
- By *2020*

Exa-scale, a simple definition

"EXA-scale" is the necessary upscale step that HPC needs to achieve in the next (few) years.

Basically, it is defined as the frontier of a **sustained** performance around 10^{18} **flop/s**.

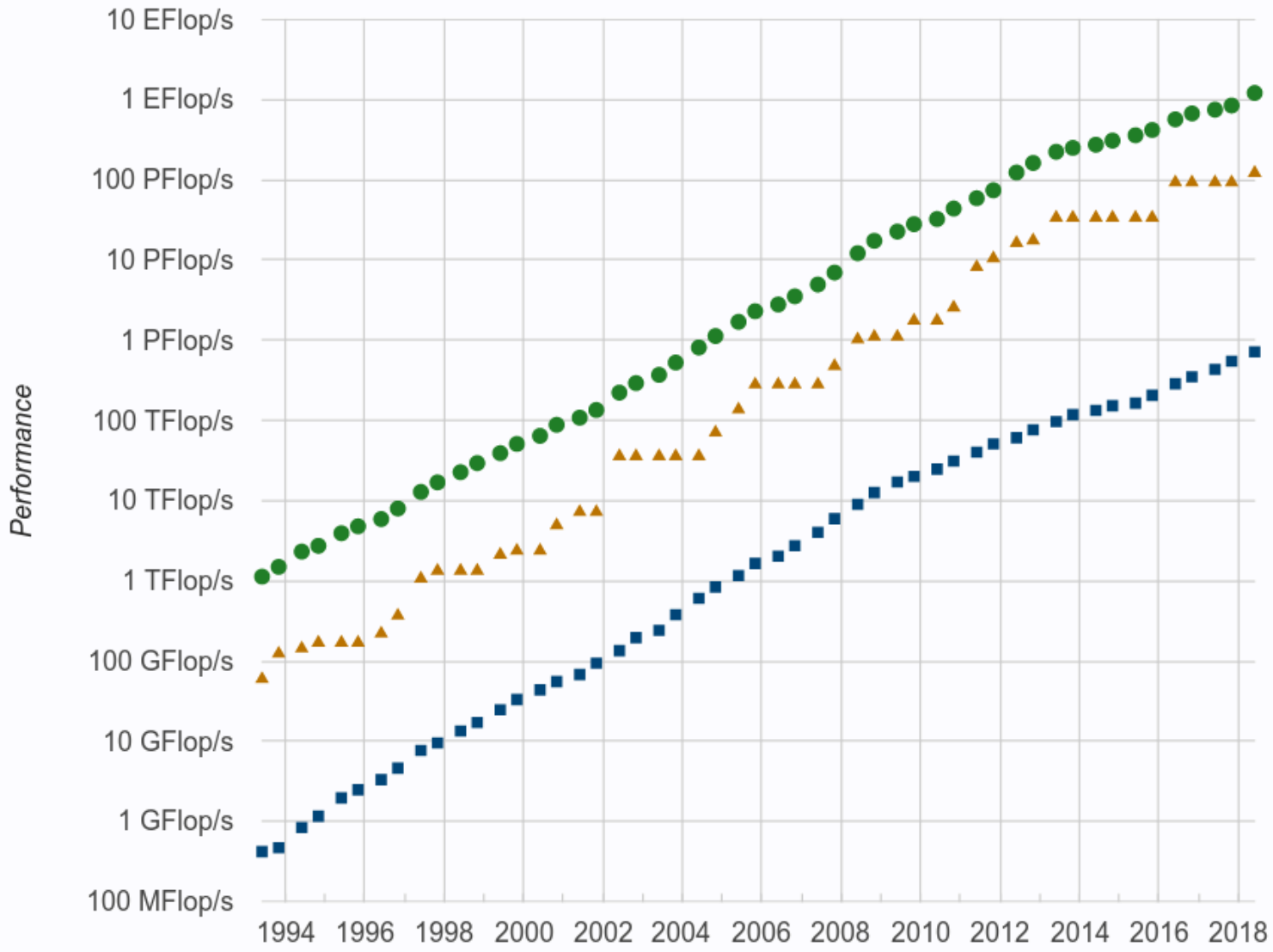
There are profound consequences on the way we design, write and optimize scientific and data-intensive codes.

Exa-scale challenges

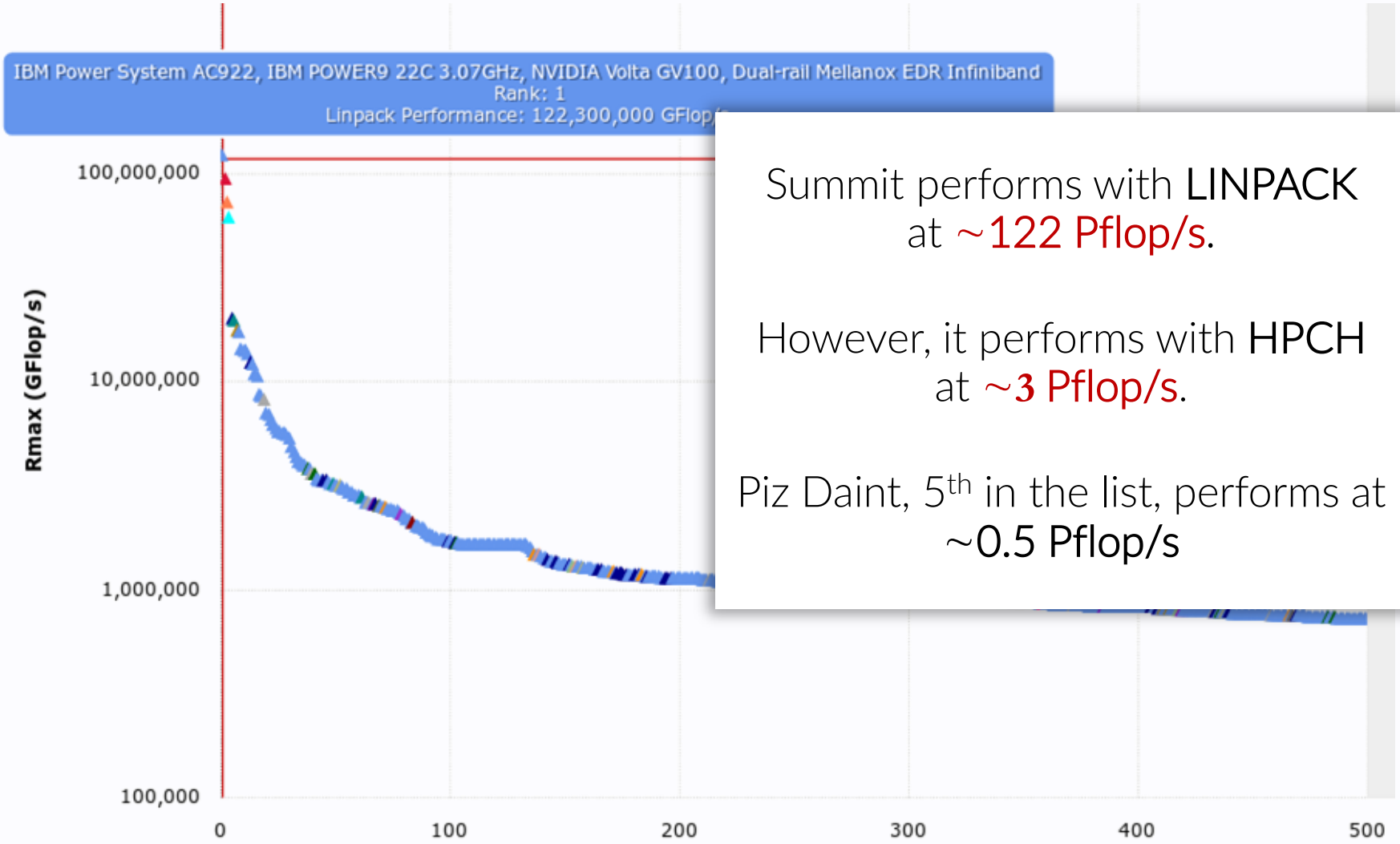
- **Performance**
- **Energy** consumption
- **Memory** amount and usage

The performance challenge

Exa-scale challenge: performance



Exa-scale challenge: performance



Summit performs with **LINPACK** at ~122 Pflop/s.

However, it performs with **HPCH** at ~3 Pflop/s.

Piz Daint, 5th in the list, performs at ~0.5 Pflop/s

Exa-scale challenge: performance

Message I

Exascale is the achievement of a sustained performance around 10^{18} Pflops.

It is a **relative term** pointing to 1000-fold better capability than that representative of the *petascale*.

Exa-scale challenge: performance

Message I

Exascale is the achievement of a **sustained performance around 10^{18} Pflops.**

It is a **relative term** pointing to **1000-fold better capability** than that representative of the *petascale*.

Is this achievement dependent only on improvements in **hardware technology** ?

i.e: shall **our codes** plug-in as they are, and just run faster, saturating a exa-scale machine ?

Flash-back:
why there’s no more free-lunch ?

Because

"There ain't no such thing as free lunch"

R.A. Heinlein
The Moon is a harsh mistress

Why there is no more "free lunch" ?

" *Moore's law* " predicted exponential growth of transistor density on chips:

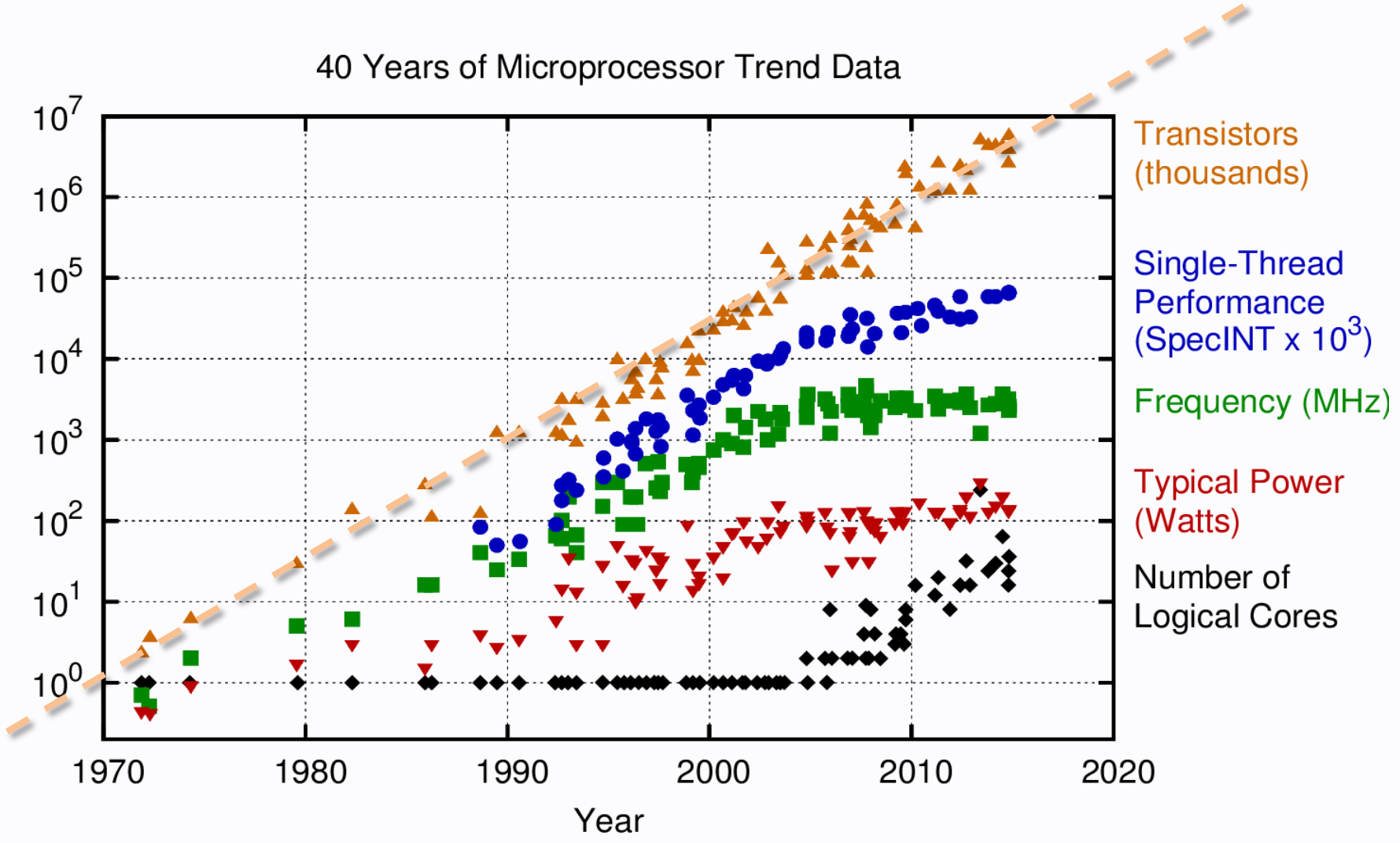
every 18 months the density of transistor will double *at the same manifacturing cost* (*).

meaning that every 18 months you could buy a commodity CPUs with 2× logics than the previous generation, at the same cost

(*) there have been even faster growths in other fields, for instance in data storage density

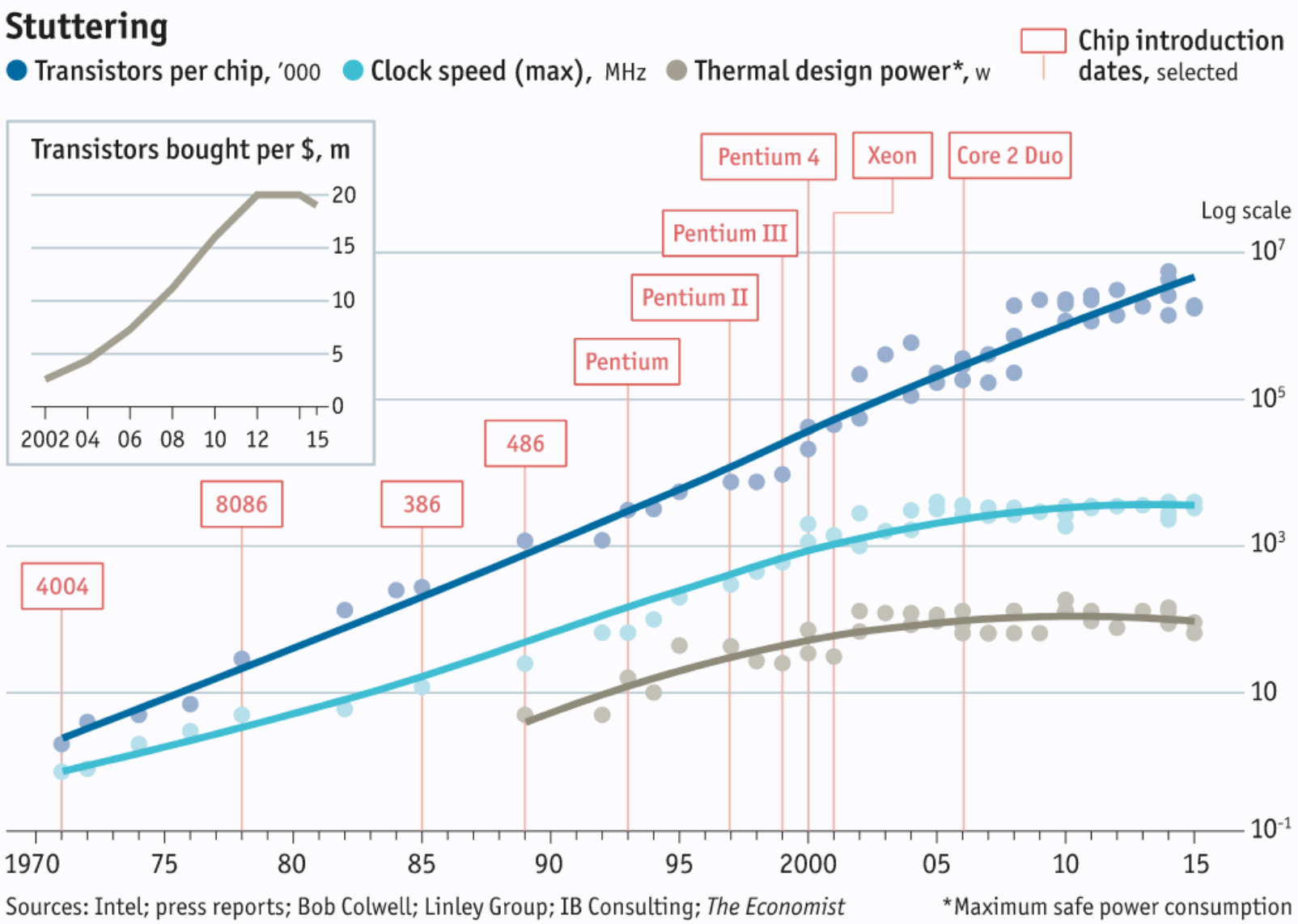
All exponential growths get to their ends..

Is the Moore's law actually ruled out ?



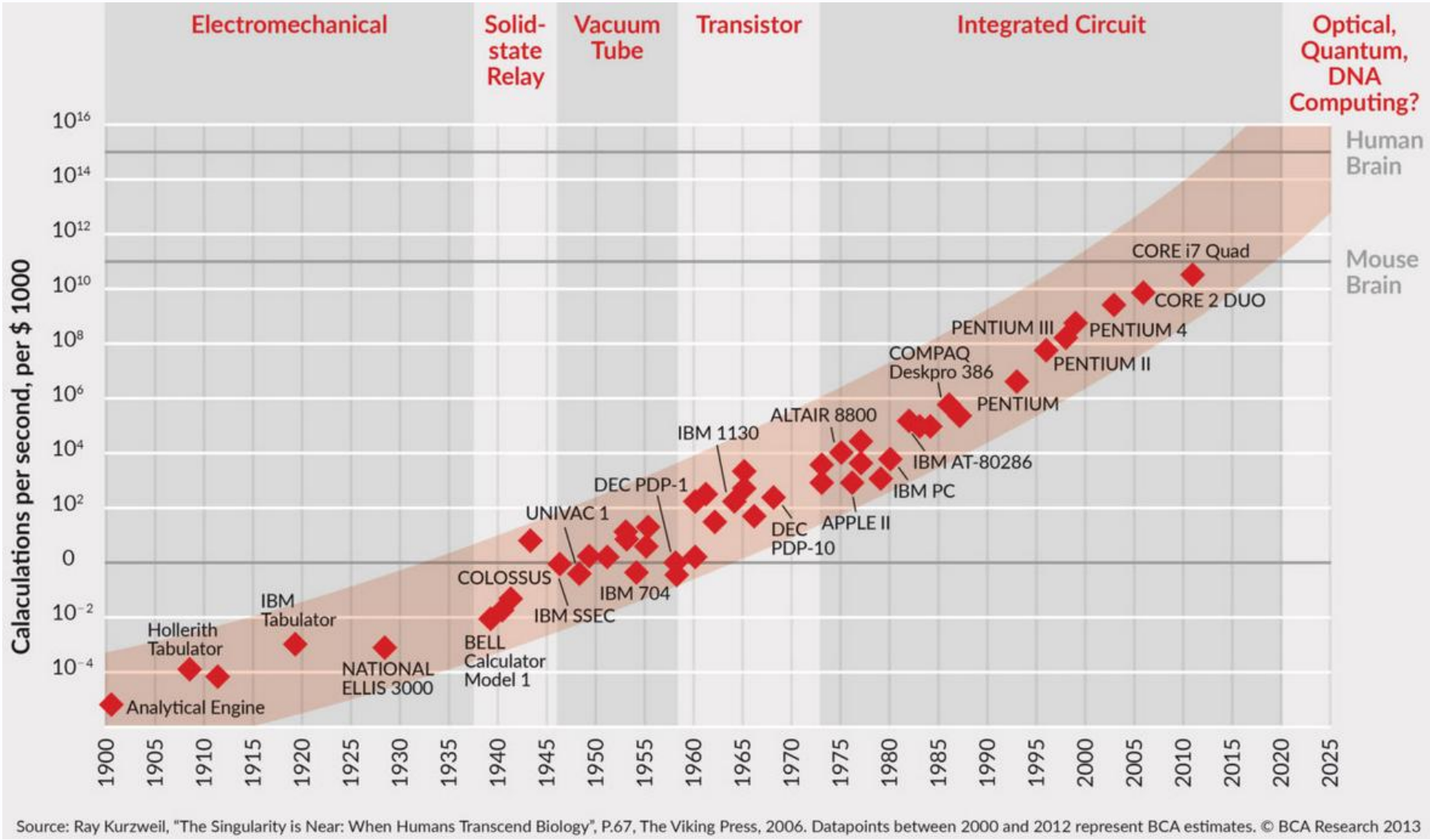
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

Is the Moore's law actually ruled out ?



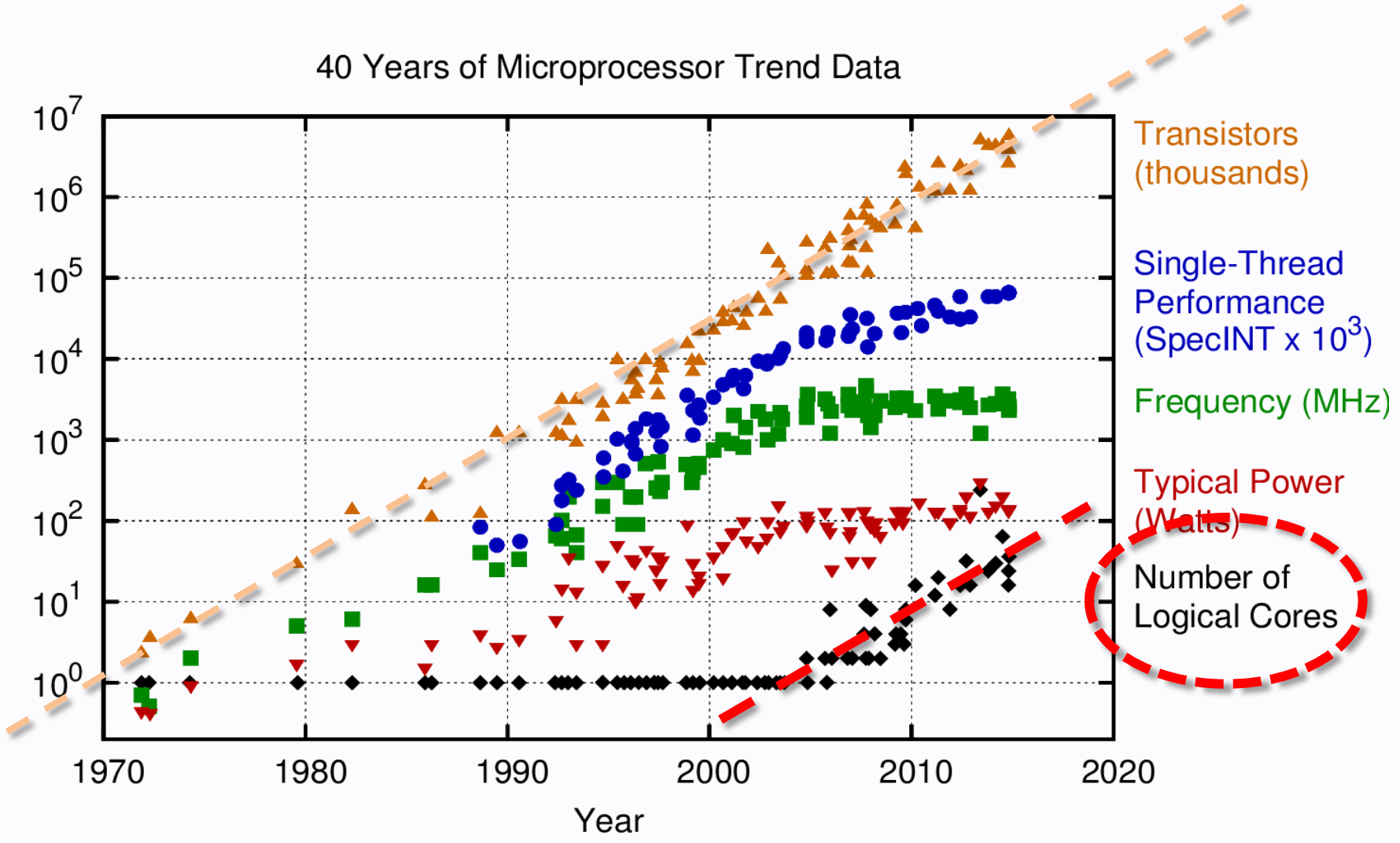
Taken from The Economist

Is the Moore's law actually ruled out ?



Source: Ray Kurzweil, "The Singularity is Near: When Humans Transcend Biology", P.67, The Viking Press, 2006. Datapoints between 2000 and 2012 represent BCA estimates. © BCA Research 2013

Is the Moore's law actually ruled out ?



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

Is the Moore's law actually ruled out ?

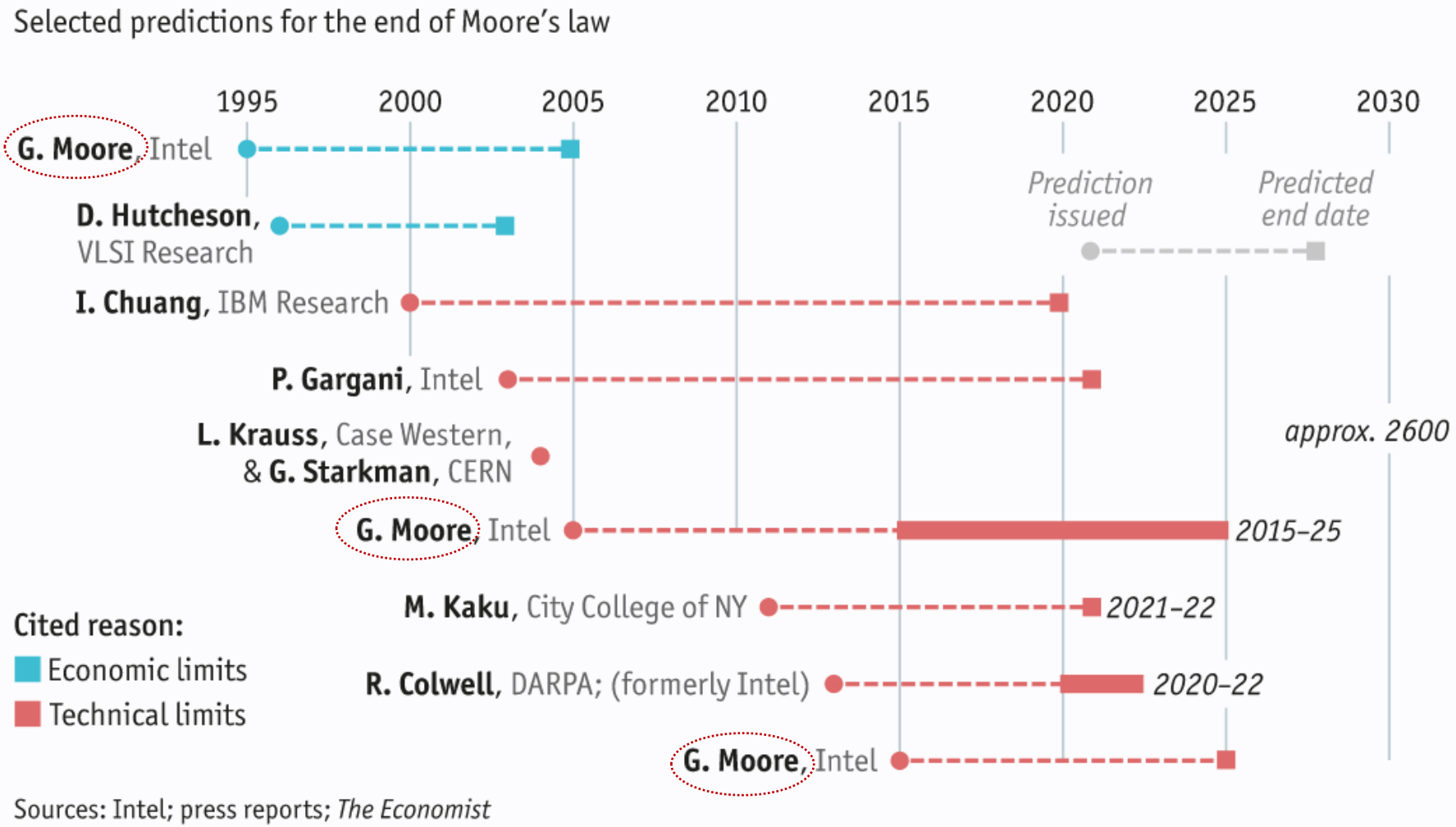
Simplest answer is:

No

(yet)

But, then...

(… however, there still are rumors about it)



Why there is no more “free lunch” ?

GOOD NEWS:
processors has continued to become “more powerful”

OTHER NEWS:
They are *(increasingly)* “differently” more powerful

Why there is no more “free lunch” ?

Until half of the ‘00s, engineers succeeded in gaining performance by essentially 3 ways:

1. Increasing **clock** speed
2. Optimizing **execution**
3. Enlarging/improving **cache**

Why there is no more "free lunch" ?

Until half of the '00s, engineers succeeded in gaining performance by essentially 3 ways:

1. Increasing **clock** speed → You get more cycles per unit time; more or less that means doing the same bunch of instructions faster
2. Optimizing **execution**
3. Enlarging/improving **cache**

Why there is no more "free lunch" ?

Until half of the '00s, engineers
performance by essentially 3 w

1. Increasing **clock** speed
2. Optimizing **execution** →
3. Enlarging/improving **cache**

- More powerful instructions
- Pipelining
- Branch predictions
- Out-of-order execution
- …

Under enormous pressure, CPUs manufacturers risked (and did) to break the semantic of your code.

Why there is no more "free lunch" ?

Until half of the '00s, engineers
performance by essentially 3 w

1. Increasing **clock** speed
2. Optimizing **execution** →
3. Enlarging/improving **cache**

- More powerful instructions
- Pipelining
- Branch predictions
- Out-of-order execution
- …

Under enormous pressure, CPUs manufacturers risked (and did) to break the semantic of your code.

Or introduce horrible bugs..
(*have you heard about* ***Meltdown*** *and* ***Spectre*** *?)*

Why there is no more "free lunch" ?

Until half of the '00s, engineers
performance by essentially 3 w

1. Increasing **clock** speed
2. Optimizing **execution** →
3. Enlarging/improving **cache**

Core 1	Core 2
`mov [X], value`	`mov [Y], value`
`mov reg1, [Y]`	`mov reg2, [X]`

- More powerful instructions
- Pipelining
- Branch predictions
- Out-of-order execution
- …

Under enormous pressure, CPUs manufacturers risked (and did) to break the semantic of your code.

Or introduce horrible bugs..
(*have you heard about* ***Meltdown*** *and* ***Spectre*** *?)*

Why there is no more "free lunch" ?

Until half of the '00s, engineers succeeded in gaining performance by essentially 3 ways:

1. Increasing **clock** speed
2. Optimizing **execution**
3. Enlarging/improving **cache** → More on that later..

Why there is no more "free lunch" ?

Since 15 years, the gain in performance is essentially due to **fundamentally different factors**:

1. Multi-core + **Multi-threads**
2. Enlarging/improving **cache**
3. Hyperthreading *(smaller contribution)*

Applications no longer get more performance for free without significant re-design, since 15 years

Why there is no more "free lunch" ?

Since 15 years, the gain in performa
due to **fundamentally different facto**

1. Multi-core + **Multi-threads** →
2. Enlarging/improving **cache**
3. Hyperthreading (*smaller contribut*

2 Cores at 3GHz are basically 1 Core at 6GHz..

False

× Cores coordination for cache-coherence
× Threads coordination
× Memory access

Applications no longer get more performance for free without significant re-design, since 15 years

Will there be any more “free lunch” ?

That is highly unlikely, unless some fundamental breakthrough in underlying physics appears.

Let’s summarize some physical argument in the following slides.

Why there is no more "free lunch" ?

Moore's law	Dennard's scaling (MOSFET)
Manifacturing cost/area is costant while the transistors' dimension halves every ~2 years → The number of transistors doubles in a CPU every ~2 years	- *voltage*, *capacitance*, *current* scale with λ - Transistor power scales as λ^2 → Power density remains constant

$$P \propto C \cdot V^2 \cdot f$$

Related to area

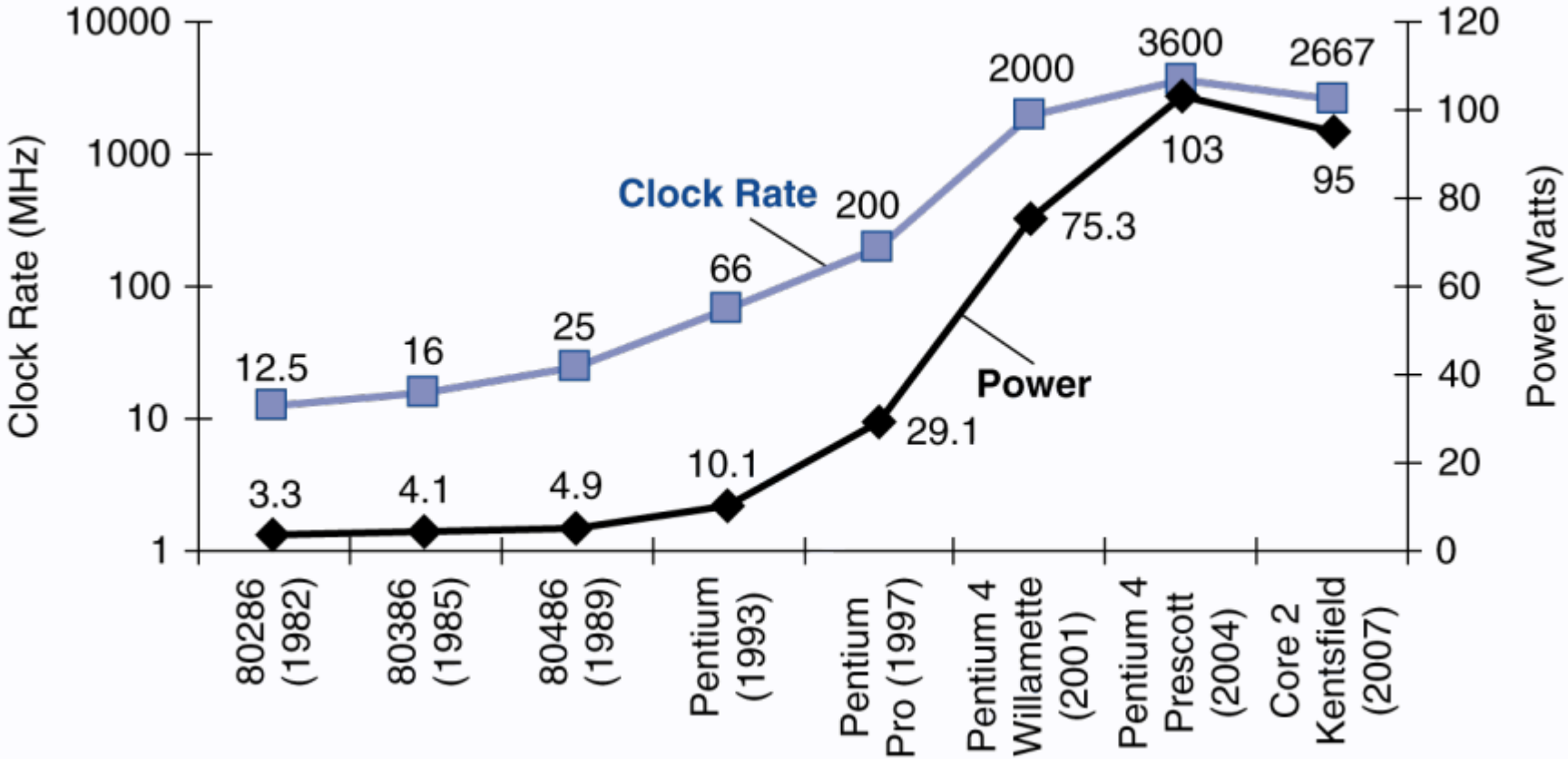
The Dennard's scaling

$$P \propto C \cdot V^2 \cdot f$$

C is the capacitance, scales as the ares
V is the voltage, scales as the linear dimension
f is the frequency

so, the linear size of transistors shrinks and so do the voltage; if the area remains equal (more transistor on the same die), then the frequency can become larger

The Dennard's scaling



$$P \propto C \cdot V^2 \cdot f$$

$\times \sim 30$ $\quad$ 5V → 1V $\quad$ $\times \sim 1000$

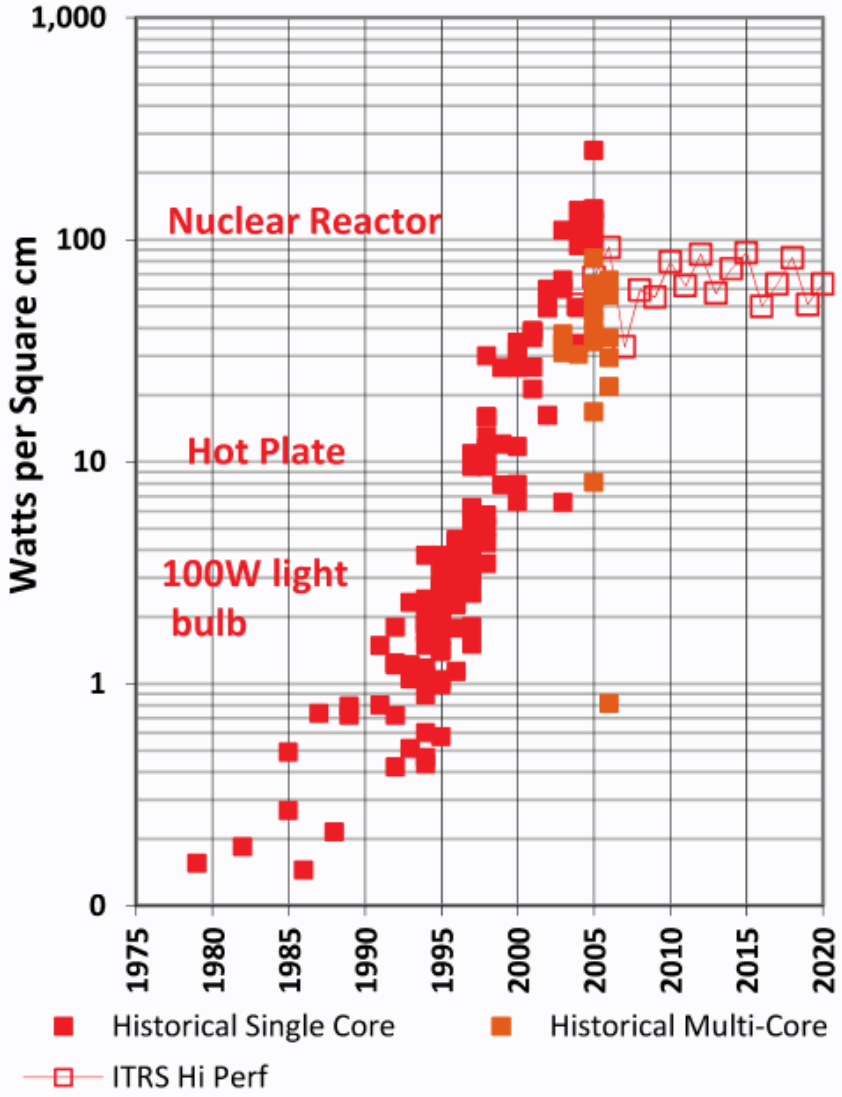
No more Dennard's scaling

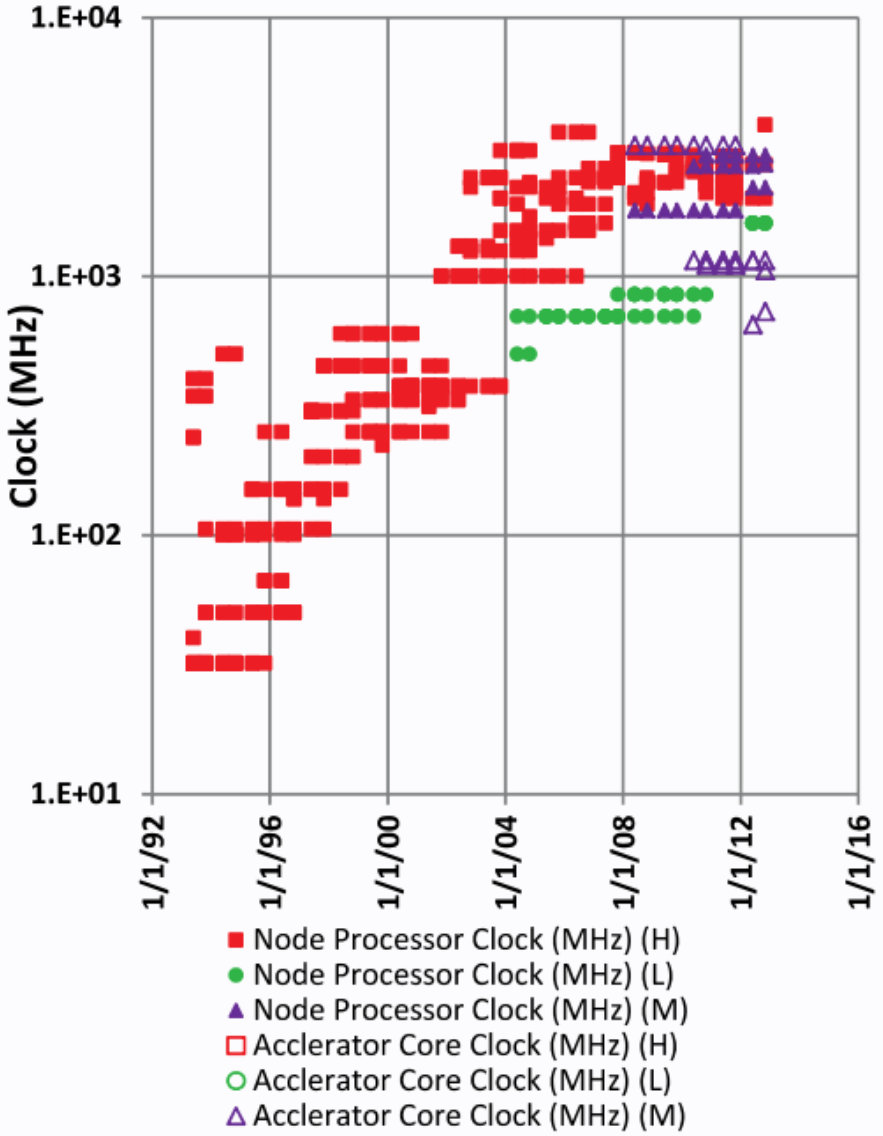
The *Dennard's scaling* has broken down to

- Leakage current
- Threshold voltage
- Physical limits ad atomic scales

So, as transistors get smaller the power density actually increases.
The result is a "power wall" that prevented the clock frequency to increase beyond ~4GHs since ~12 years

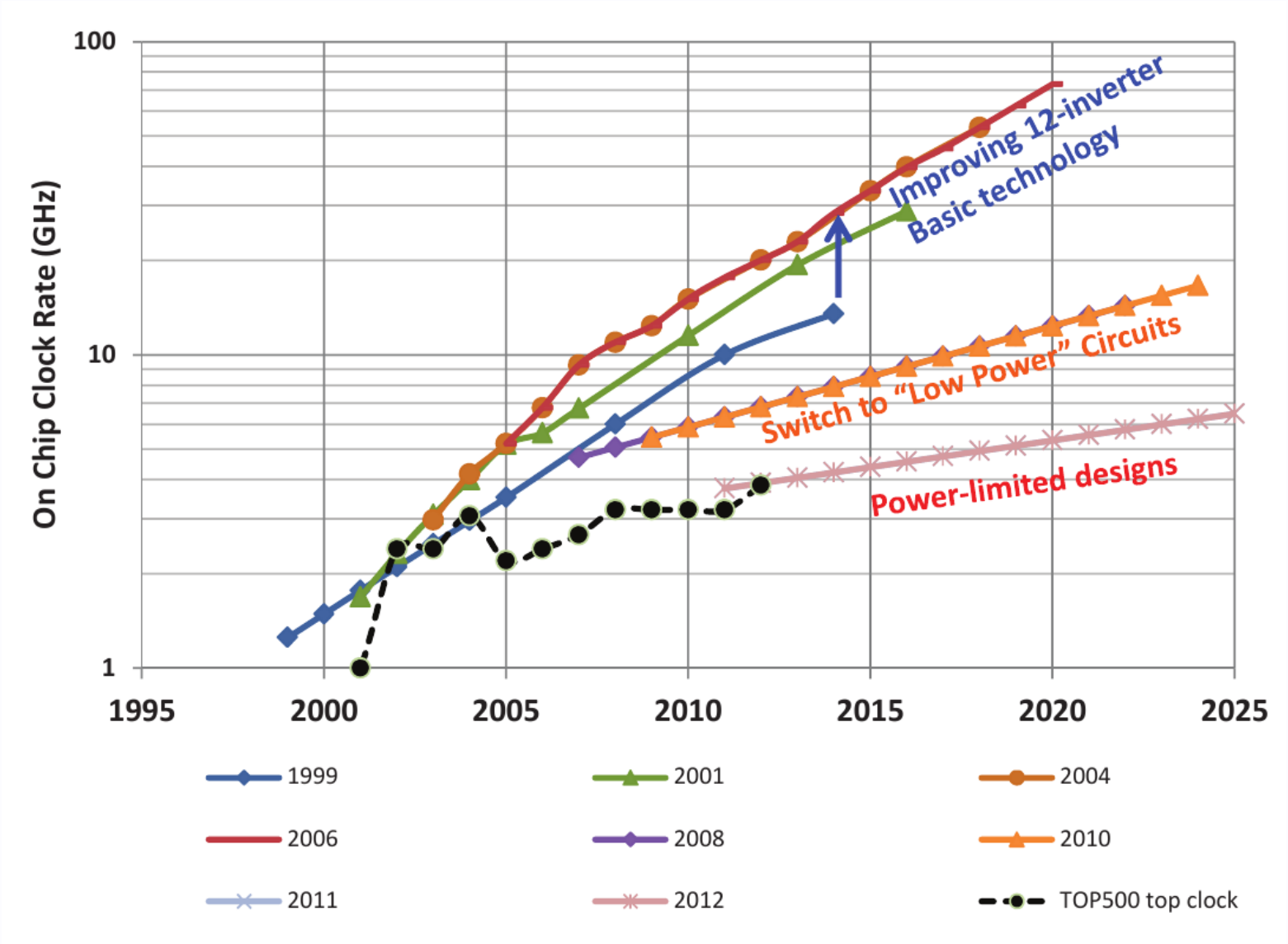
The "Power Wall"





Source: Kogge and Shalf, IEEE CISE

A lot of research is going on here

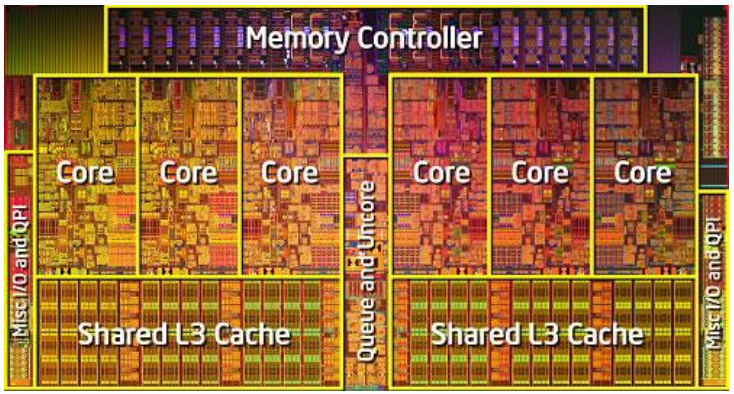


Taken from a W. Gropp's talk

Back to the future: as for now, no more "free lunch"

Message II

Many-cores CPUs are here to stay



- Concurrency-based model programming (which is different than both *parallel* and *ILP*): means work subdivision in as many independent tasks as possibile
- Specialized, heterogeneous cores
- Multiple memory hierarchies

Energy consumption

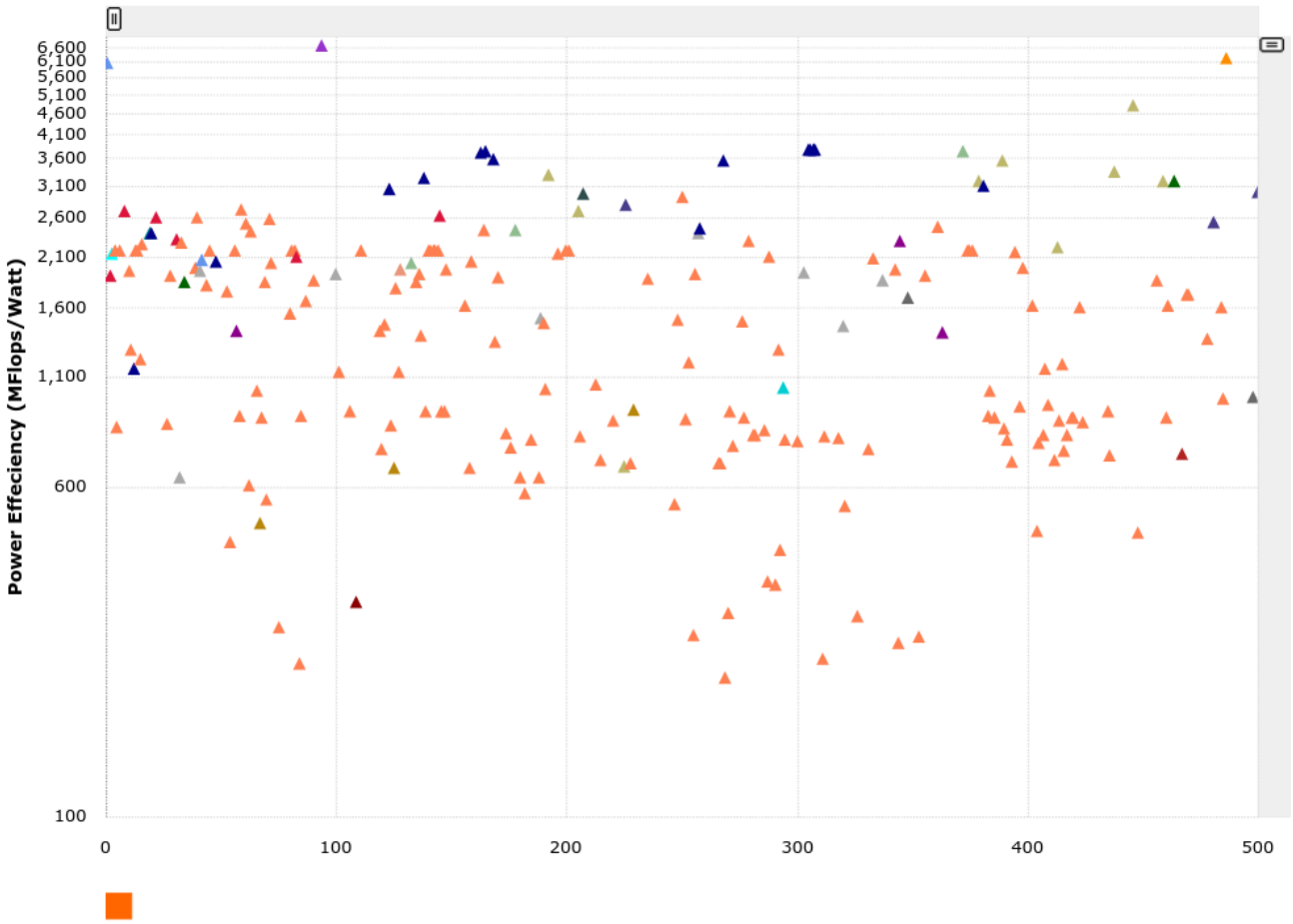
Exa-scale challenge: energy consumption

Sunway performs for some apps at ≈ 10 Pflop/s consuming ≈ 18MW.

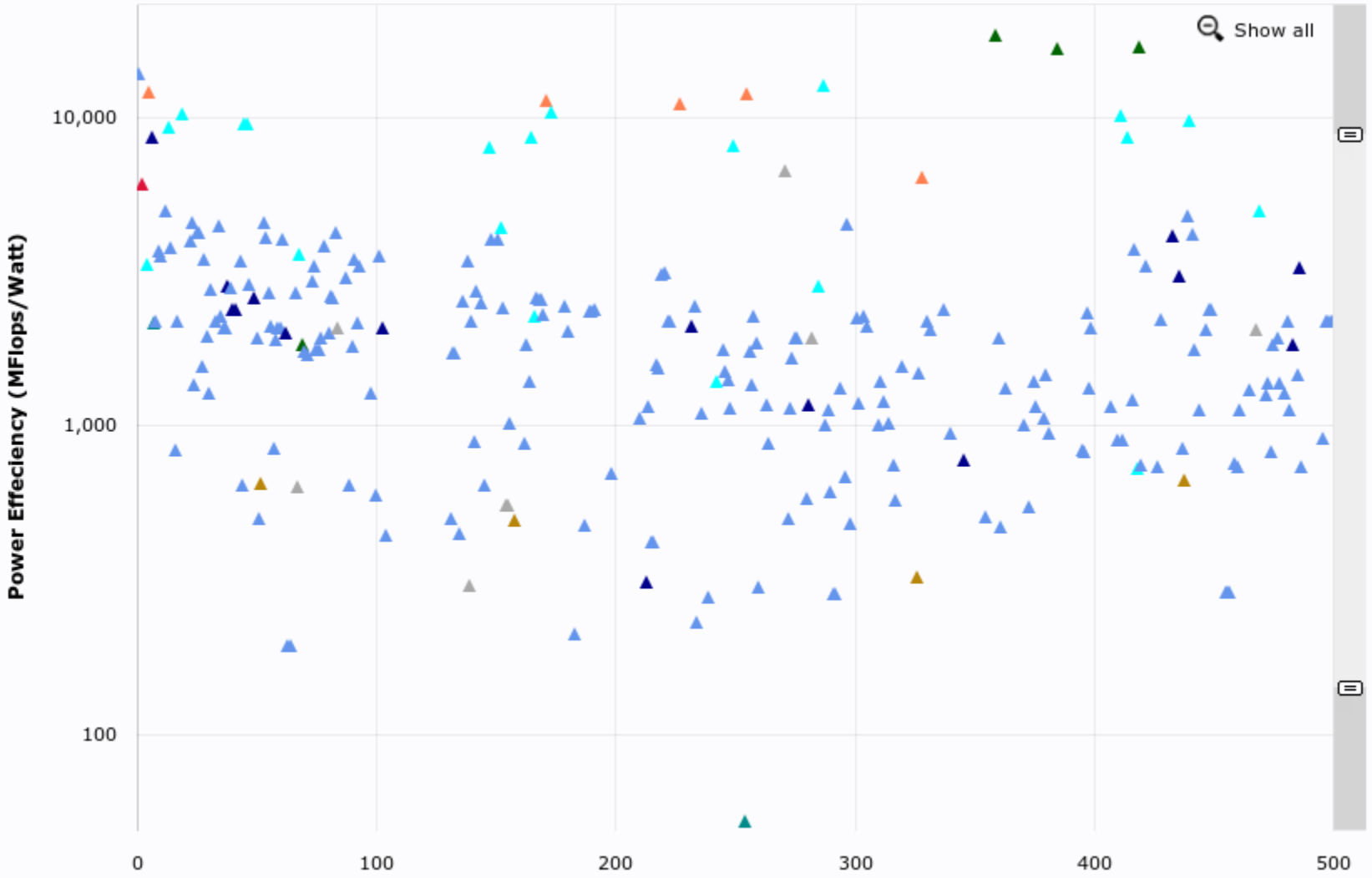
Simply rescaling to Eflop/s, it would consume ≈1.8GW.

The exa-scale goal is to reach Eflop/s at 20MW of electric power, i.e. 50 Gflops/W

Rule of thumb: 1MW = $1M / yr



Exa-scale challenge: energy consumption



Exa-scale challenge: energy consumption

What dominates the energy consumption in computation ?

Operation	*pJoules*
64bits FP 28nm CMOS	12
32bits integer operations on 28nm CMOS	3
64bits FP single-issue in-order core	200
64bits multiple-issue out-of-order core	1000
reading 32bits instruction from 32KB cache	20
reading 64bits operands from DRAM	2000

Exa-scale challenge: energy consumption

Message III

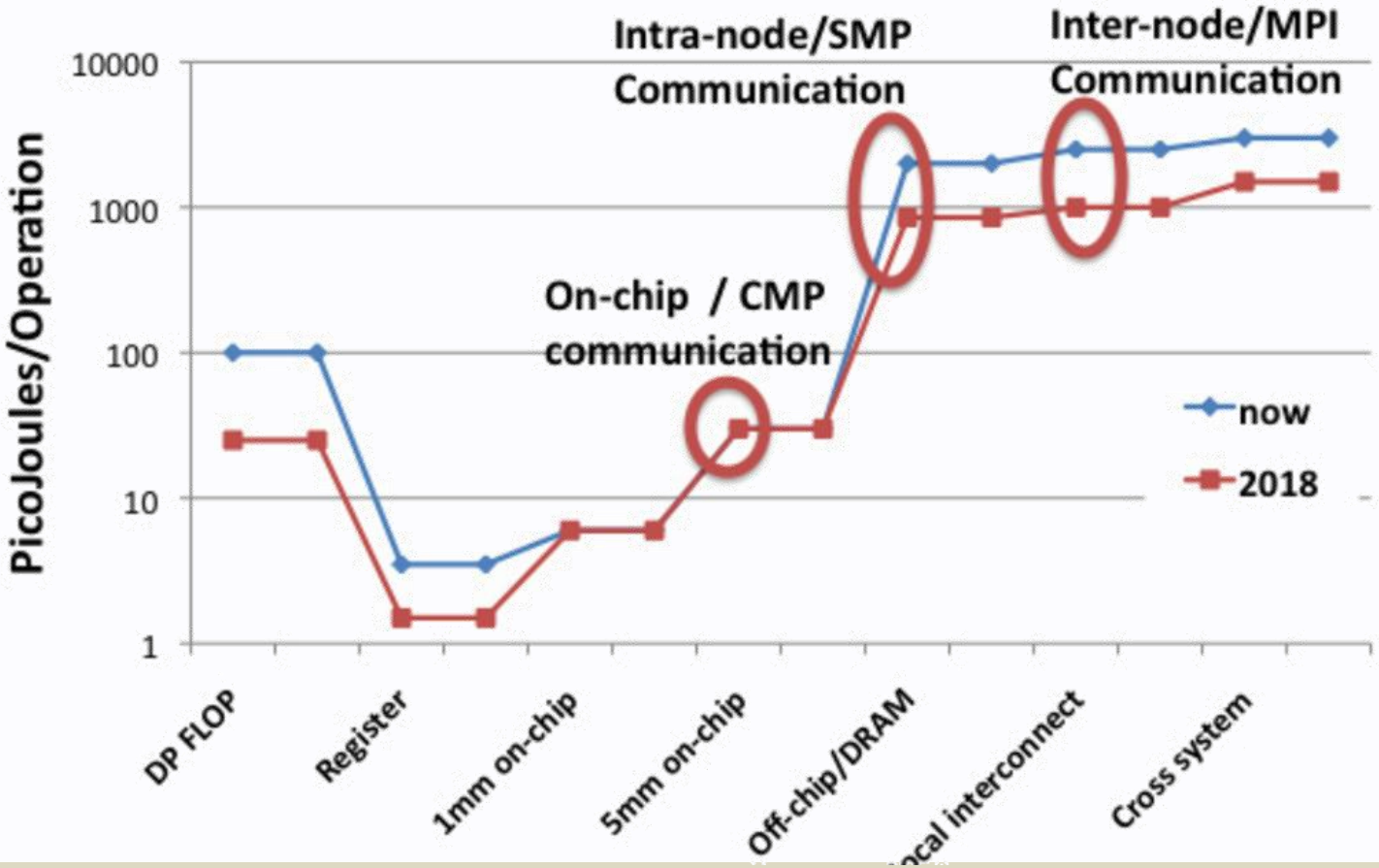
Moving memory is among most expensive operations,
x100 or more than a 64bits FP instruction.

By 2018 FP will cost **10 pJ on 11nm chips**,
while reading from DRAM will still cost >**1000pJ**.

A **10 Tflop** chip will require **100W**.
It shall take **2000W** of power to supply
memory bandwidth for a modest **Bytes/FP of 0.2**

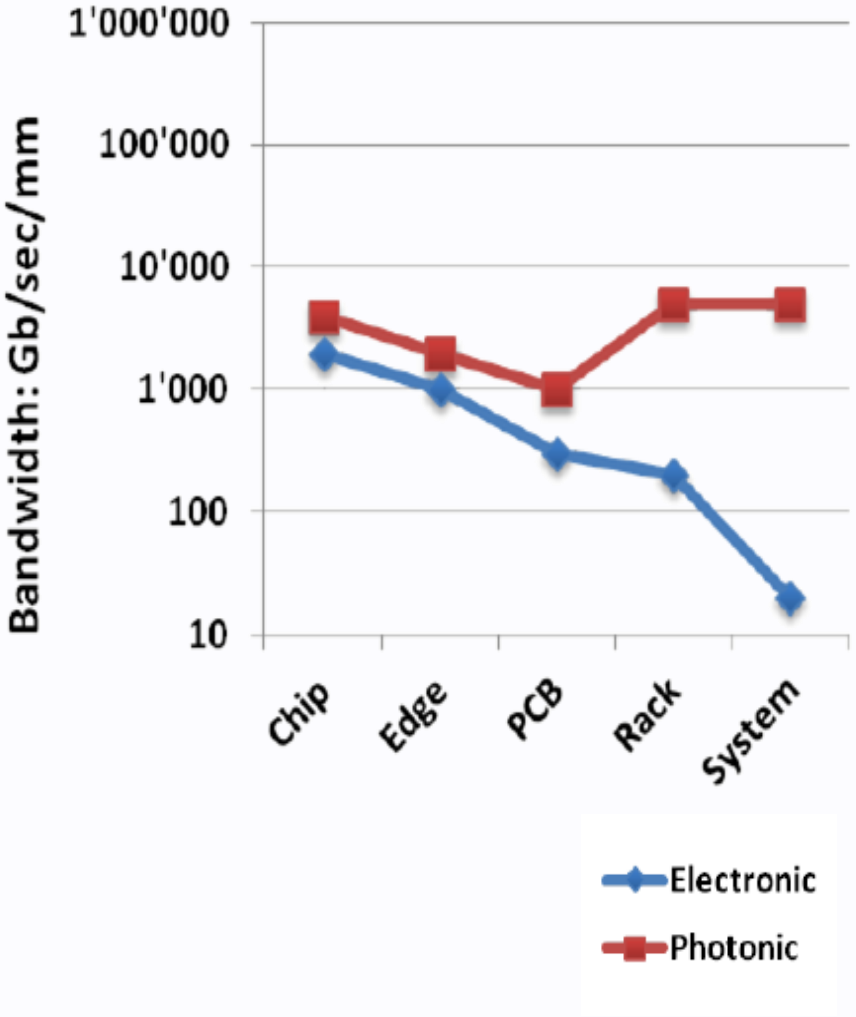
Exa-scale challenge: memory capacity

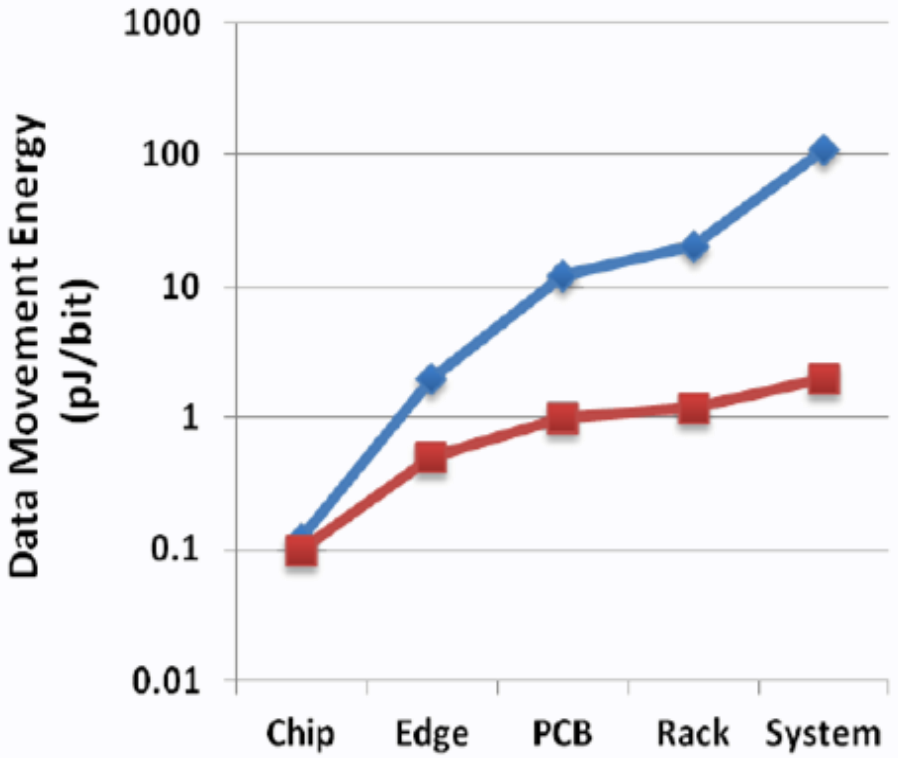
Data movement presents the most daunting engineering and computer architecture challenge.



Exa-scale challenge: memory capacity

..unless more efficient memory technologies are developed





Exa-scale challenge: memory capacity

Message IV

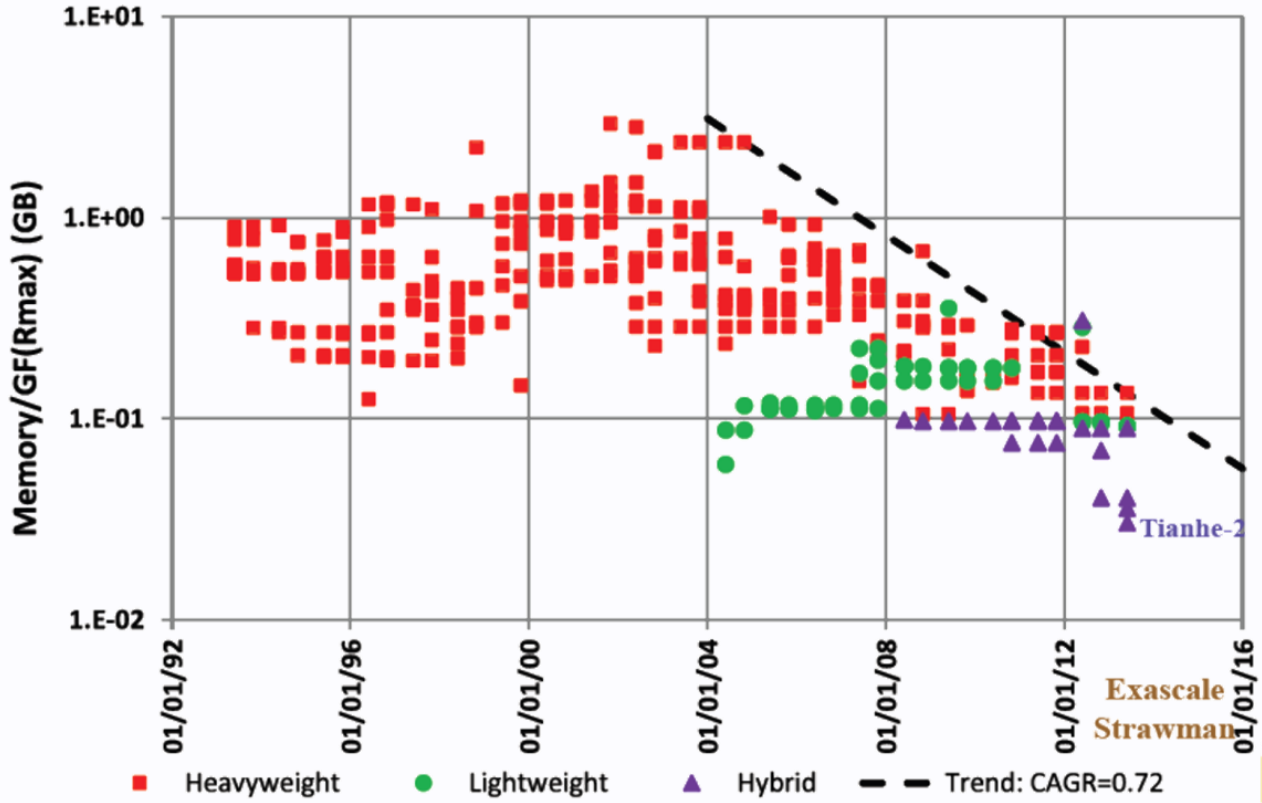
Memory Capacity is critical to applications.

- weak scaling
- in-memory checkpoints
- message logging/replay for resilience
- algorithms that buy performance by using data structures that may not be minimal in their memory footprint.

Exa-scale challenges: memory capacity

The machines at the top of the TOP500 **do not have sufficient memory to match historical requirements of 1B/Flop**, and the situation is getting worse.

This is a big change: it places the burden increasingly on **strong-scaling** of applications for performance, rather than on **weak-scaling** like in tera-scale era.



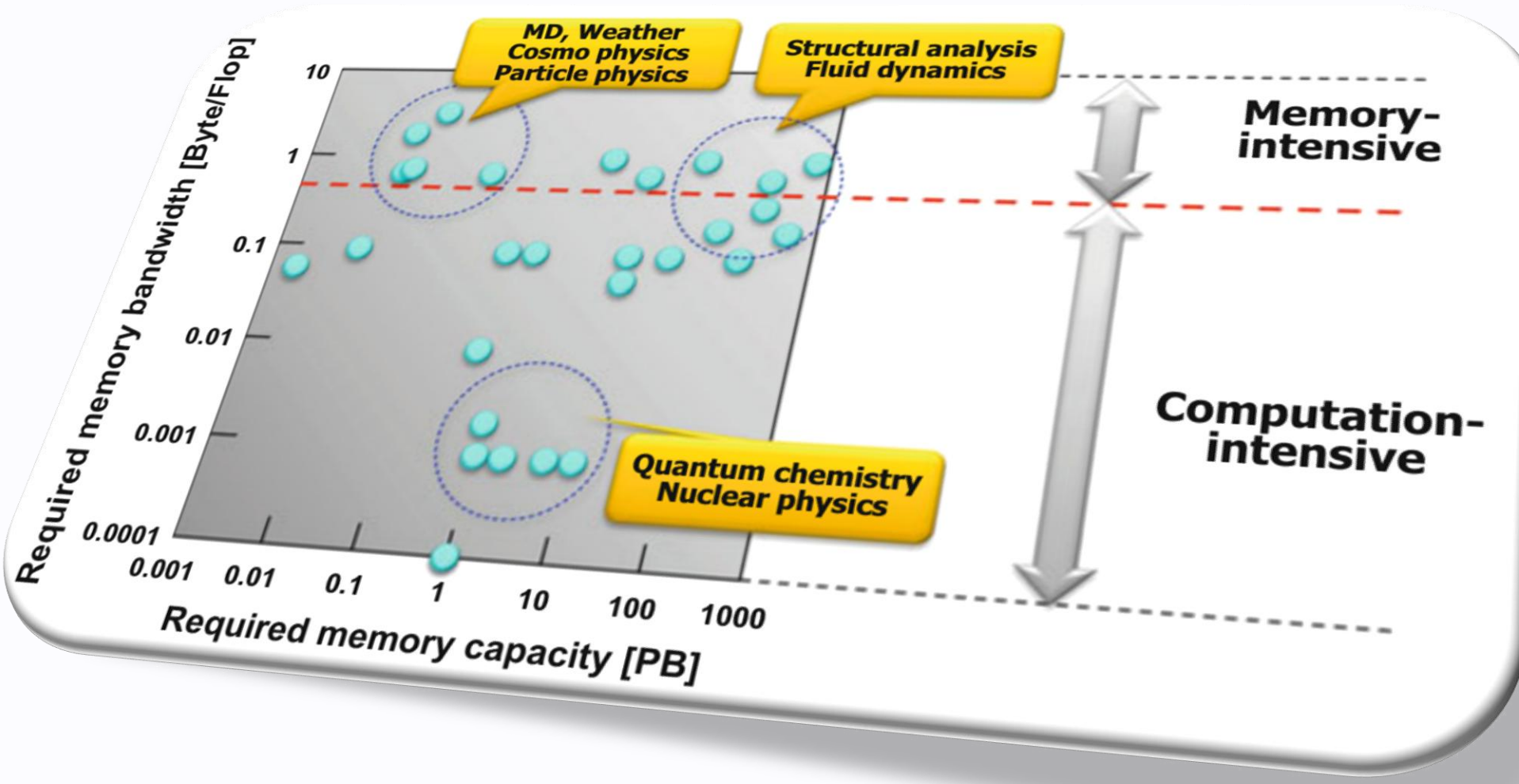
Exa-scale challenges: memory capacity

Memory power consumption $\propto$ Bw $\times$ Length^2 / Area

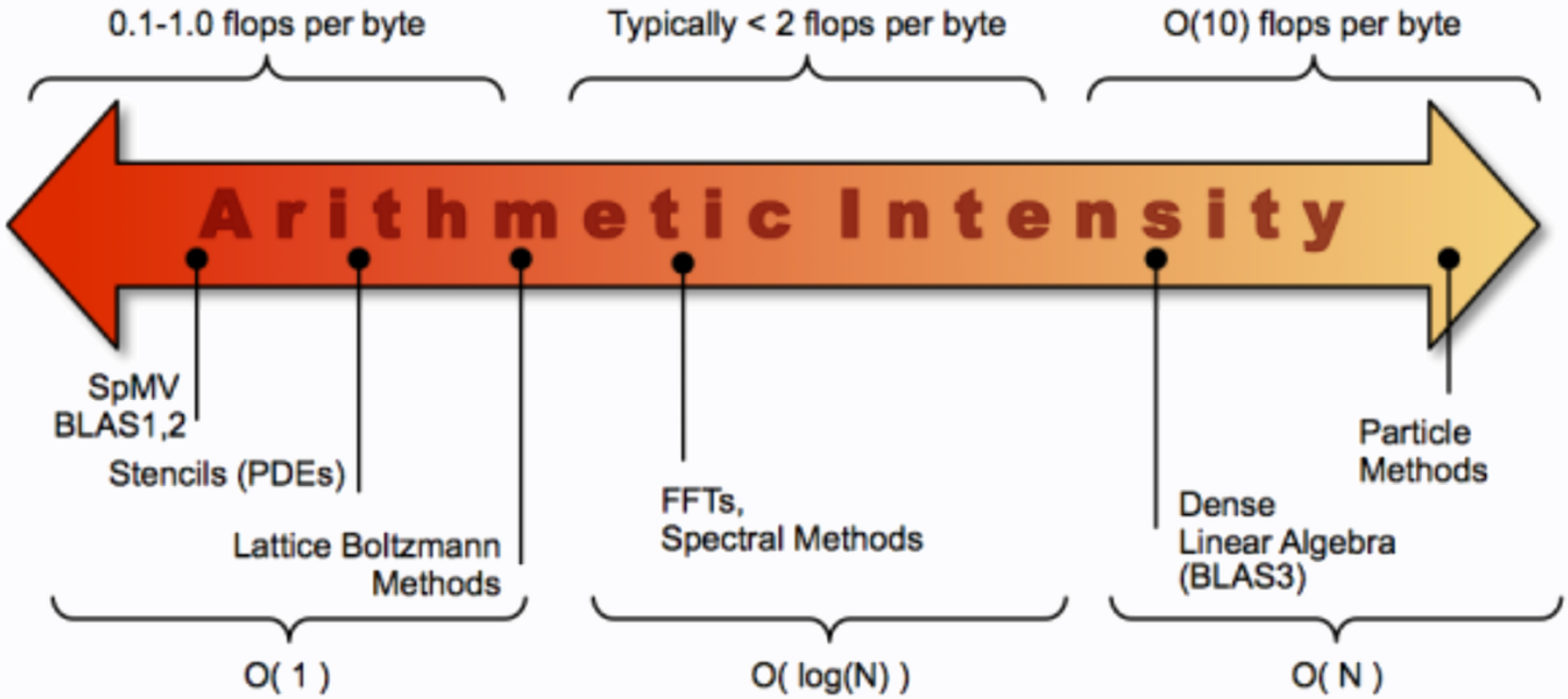
	AMD Radeon R9 290X	NVIDIA GeForce GTX 980 Ti	AMD Radeon R9 Fury X	Samsung's 4-Stack HBM2 based on 8 Gb DRAMs	Theoretical GDDR5X 256-bit sub-system
Total Capacity	4 GB	6 GB	4 GB	16 GB	8 GB
Bandwidth Per Pin	5 Gb/s	7 Gb/s	1 Gb/s	2 Gb/s	10 Gb/s
Number of Chips/Stacks	16	12	4	4	8
Bandwidth Per Chip/Stack	20 GB/s	28 GB/s	128 GB/s	256 GB/s	40 GB/s
Effective Bus Width	512-bit	384-bit	4096-bit	4096-bit	256-bit
Total Bandwidth	320 GB/s	336 GB/s	512 GB/s	1 TB/s	320 GB/s
Estimated DRAM Power Consumption	30W	31.5W	14.6W	n/a	20W

Feeding 1B / flop for 10^{18} flop/s ~28 MW ~60 MW

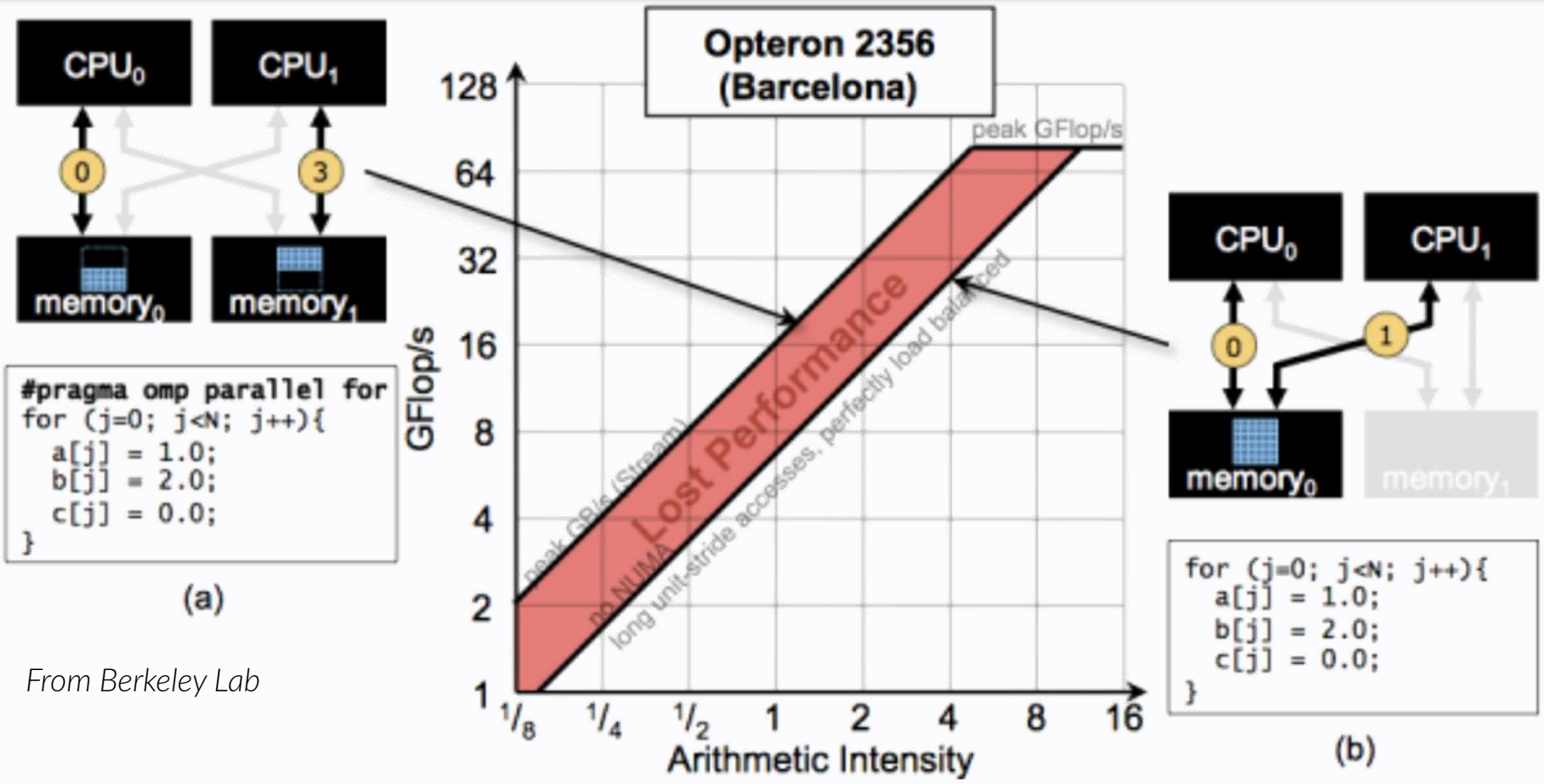
Memory vs computation intensity



Memory vs computation intensity



Impact of NUMA on computation efficiency



From Berkeley Lab

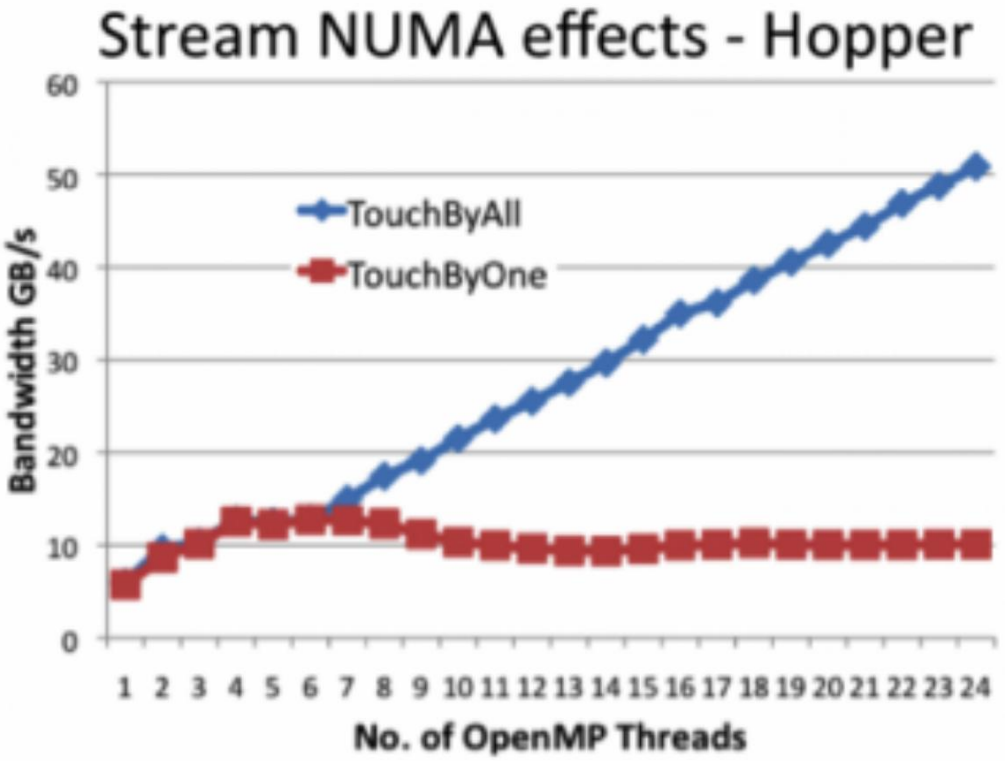
Depending on the locality of data and on where the threads are running, the memory bandwidth can change a lot.

The above example illustrates the impact of first-touch data allocation in OpenMP

Impact of NUMA on computation efficiency

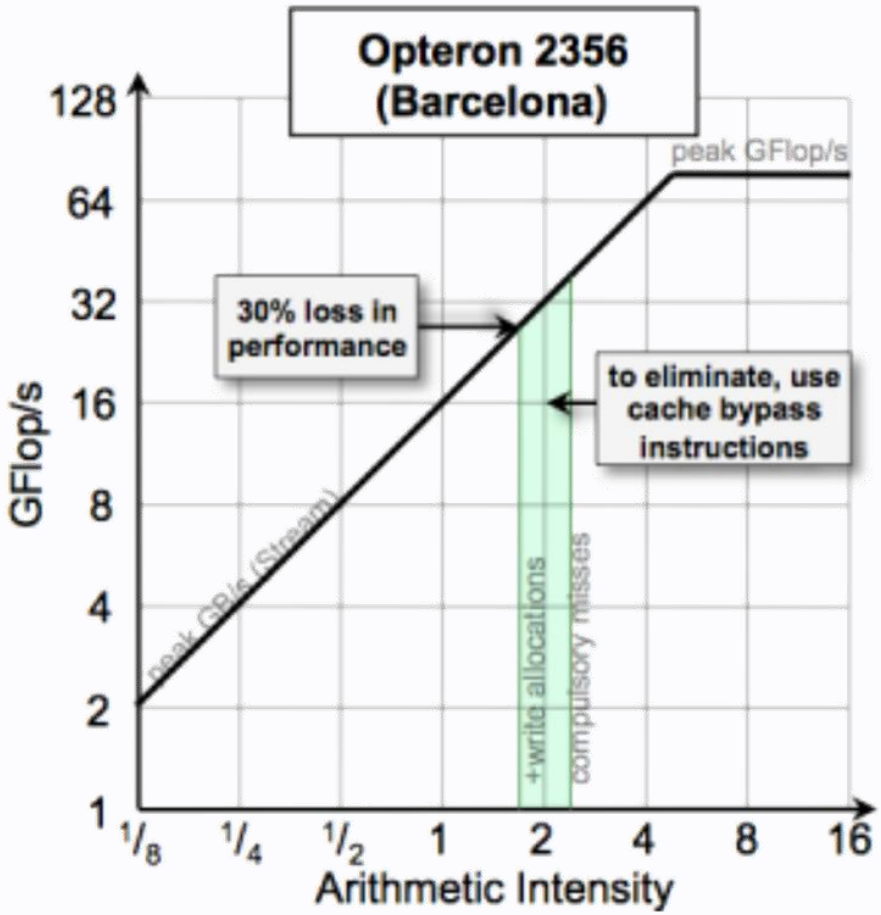
```
! Initialization
#pragma omp parallel for
 for (j=0; j<VectorSize; j++) {
 a[j] = 1.0; b[j] = 2.0; c[j] = 0.0;}

! Compute
#pragma omp parallel for
 for (j=0; j<VectorSize; j++) {
 a[j]=b[j]+d*c[j];}
```

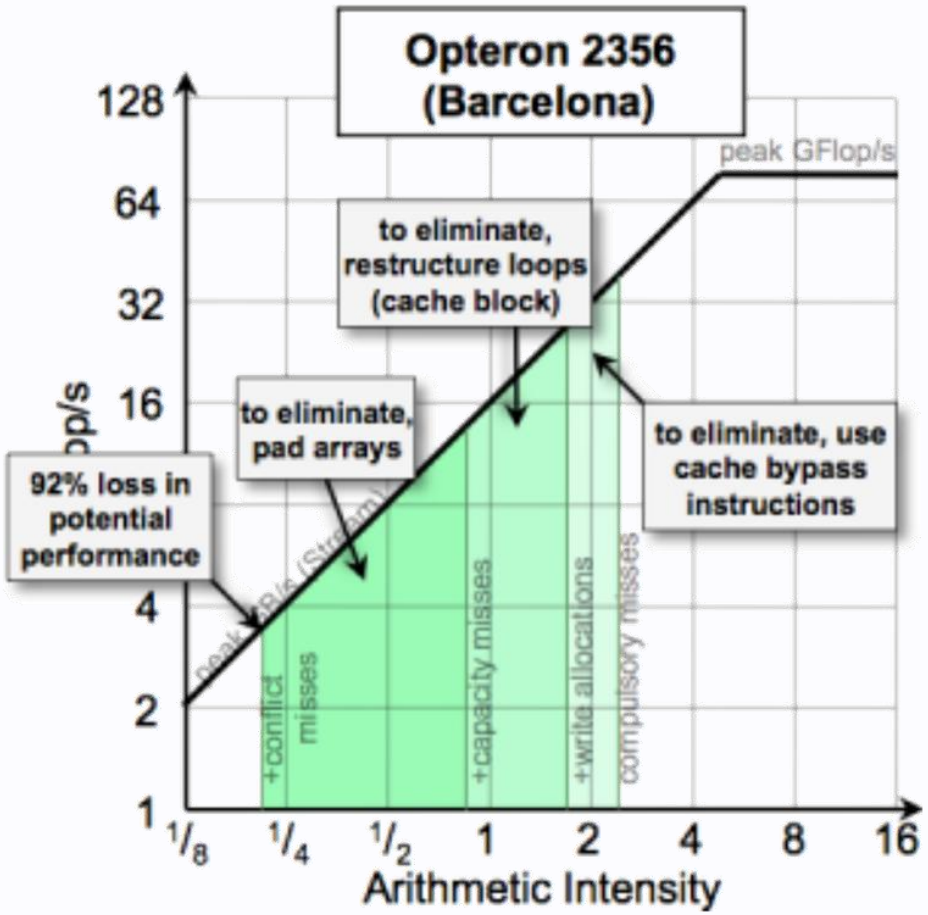
Taken from NERSC; try on your own machine



The effect of «first-touch»

Impact of NUMA on computation efficiency

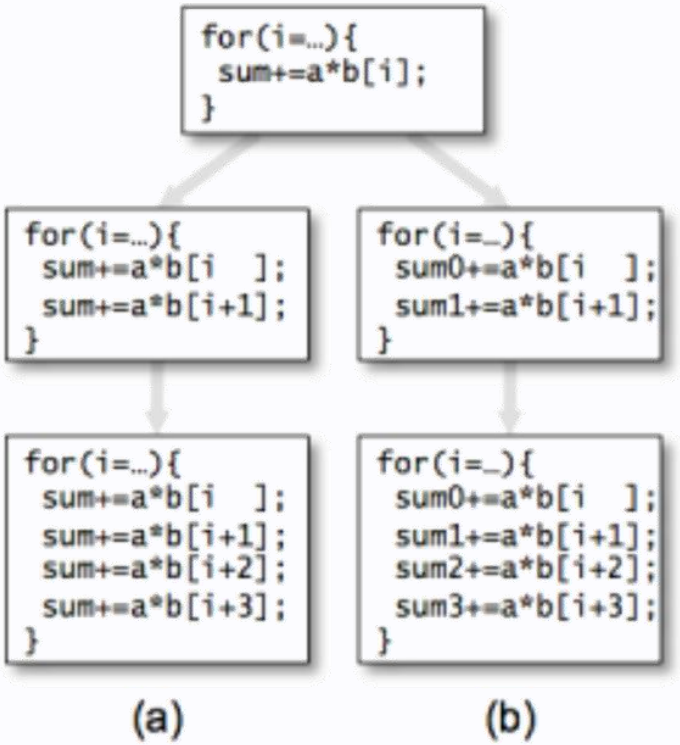


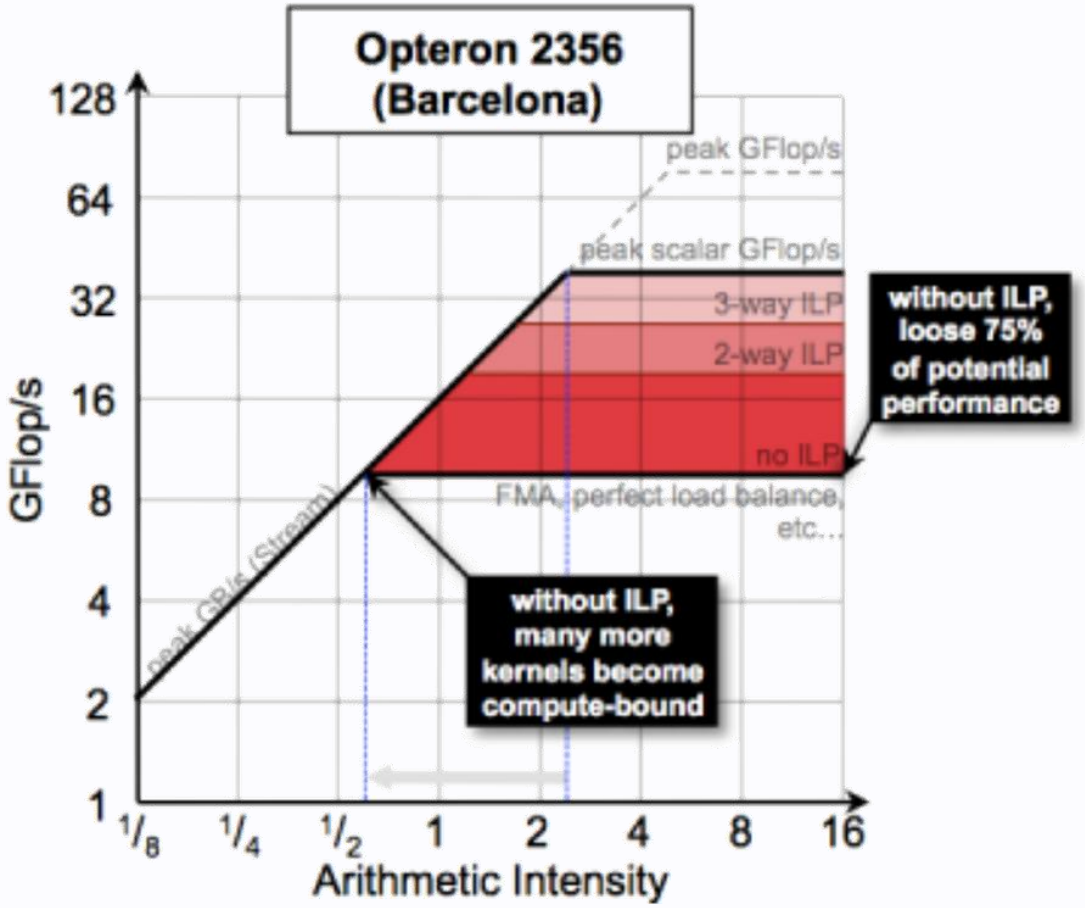


Wrong use of **cache** may result in a catastrophe.

Caches capacity and misses and useless write-allocations can result in redundant data movements

Impact of ILP on computation efficiency





In order that pipelines and ILP are an advantage, you have to shape your code so not to obstruct the compiler.

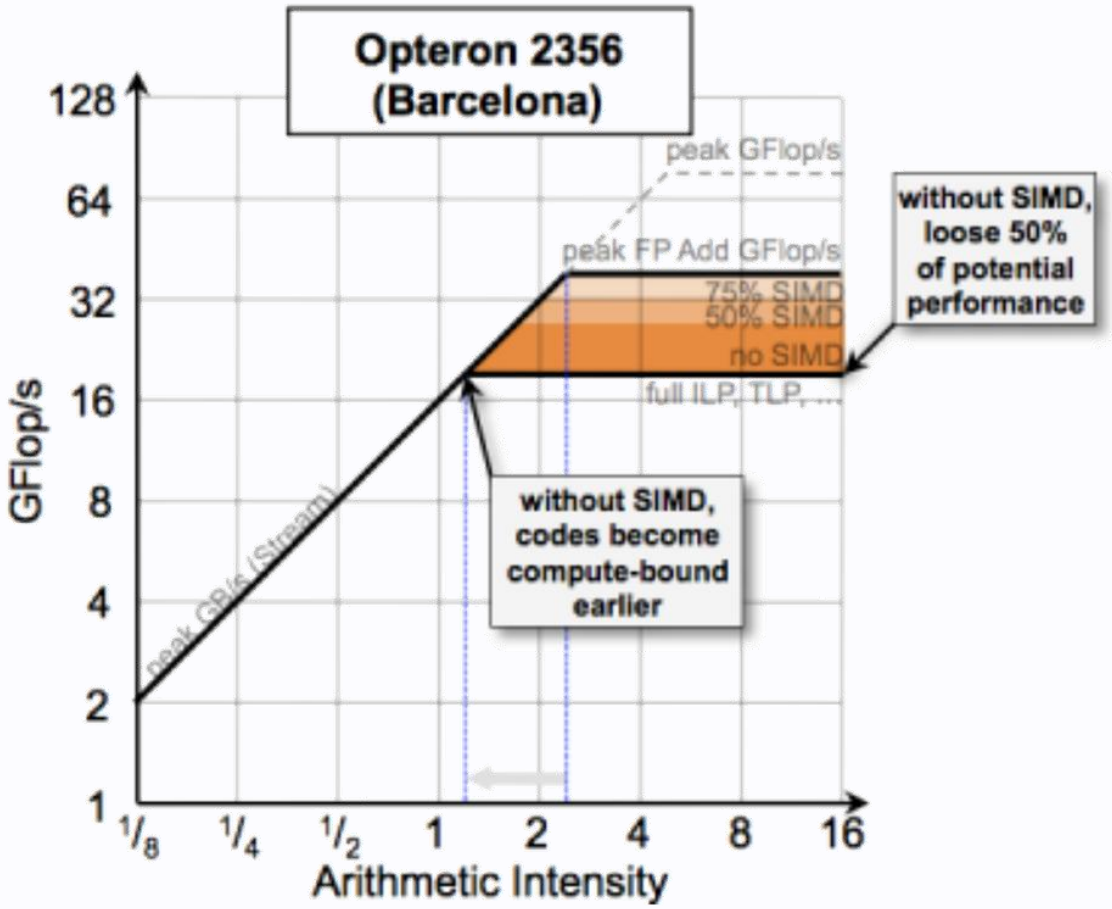
Important for compute-intensive, not that much for memory-intensive

Impact of SIMD on computation efficiency

```
for(i=…){
 sum0+=b[i  ];
 sum1+=b[i+1];
 sum2+=b[i+2];
 sum3+=b[i+3];
}
```

```
for(i=…){
 sum0=_mm_add_sd(sum0,…b[i  ]…);
 sum1=_mm_add_sd(sum1,…b[i+1]…);
 sum2=_mm_add_sd(sum2,…b[i+2]…);
 sum3=_mm_add_sd(sum3,…b[i+3]…);
}
```

```
for(i=…){
 sum01=_mm_add_pd(sum01,…b[i  ]…);
 sum23=_mm_add_pd(sum23,…b[i+2]…);
 sum45=_mm_add_pd(sum45,…b[i+4]…);
 sum67=_mm_add_pd(sum67,…b[i+6]…);
}
```

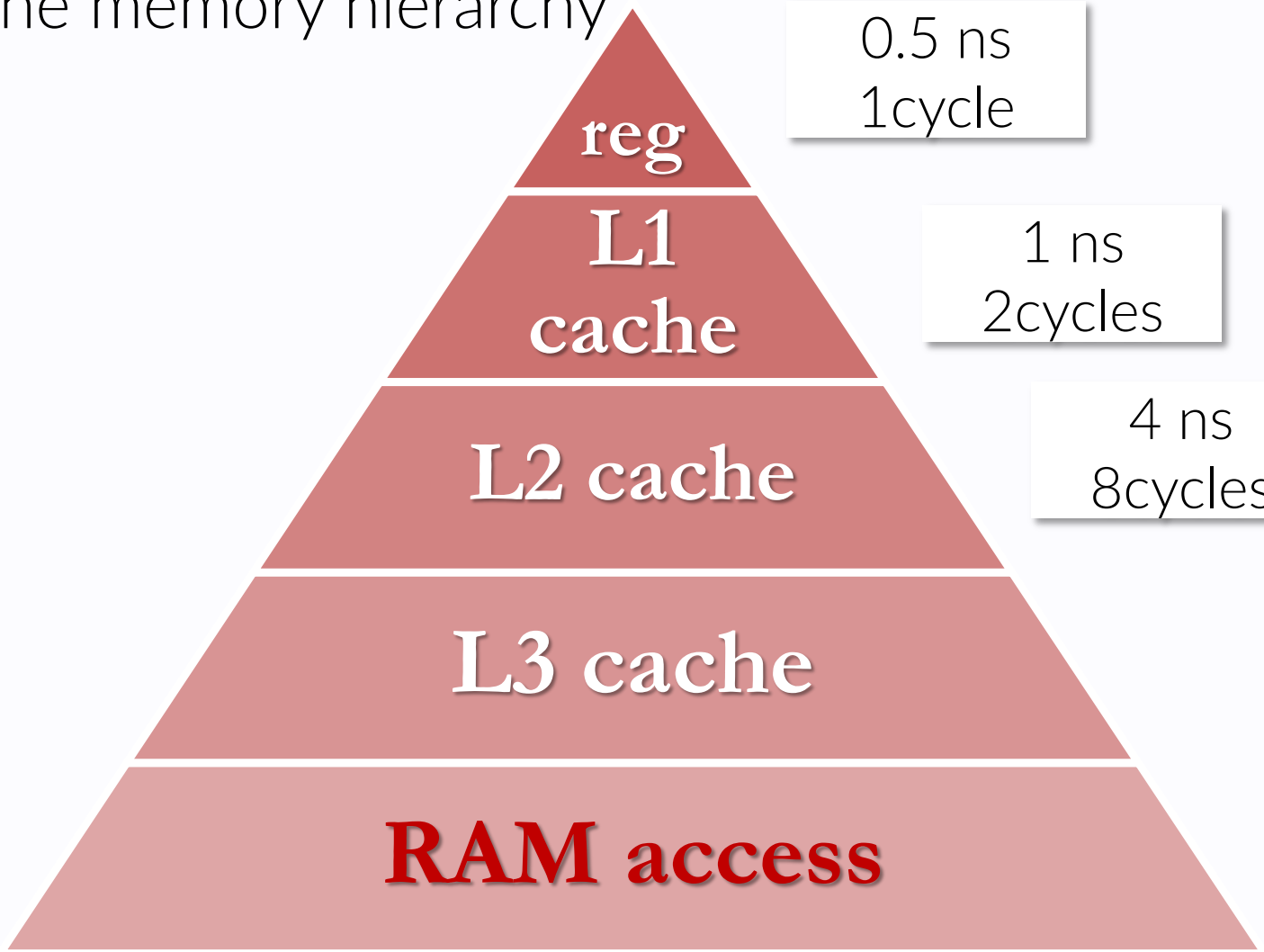


As for ILP, it may be highly dependent on you whether or not the compiler is able to attain peak performance.

Or you may choose to use intrinsics function directly.

Is cache so useful ?

The memory hierarchy



0.5 ns
1cycle

1 ns
2cycles

4 ns
8cycles

100 ns
100cycles

Is cache so useful ?

1st example: 1 level of cache

Average Access Time

99% of L1 hit

$$AAT = 1 + 0.01 \times 100 = 2 \text{ cycles}$$

97% of L1 hit

$$AAT = 1 + 0.03 \times 100 = 4 \text{ cycles}$$

Is cache so useful ?

2nd example: 2 levels of cache

Average Access Time

99% of L1 hit, 99% of L2 hit

$$AAT = 1 + 0.01 \times 2 + 0.0001 \times 100 = 1.03 \text{ cycles}$$

97% of L1 hit, 99% of L2 hit

$$AAT = 1 + 0.03 \times 2 + 0.0003 \times 100 = 1.09 \text{ cycles}$$

97% of L1 hit, 97% of L2 hit

$$AAT = 1 + 0.03 \times 2 + 0.0009 \times 100 = 1.15 \text{ cycles}$$

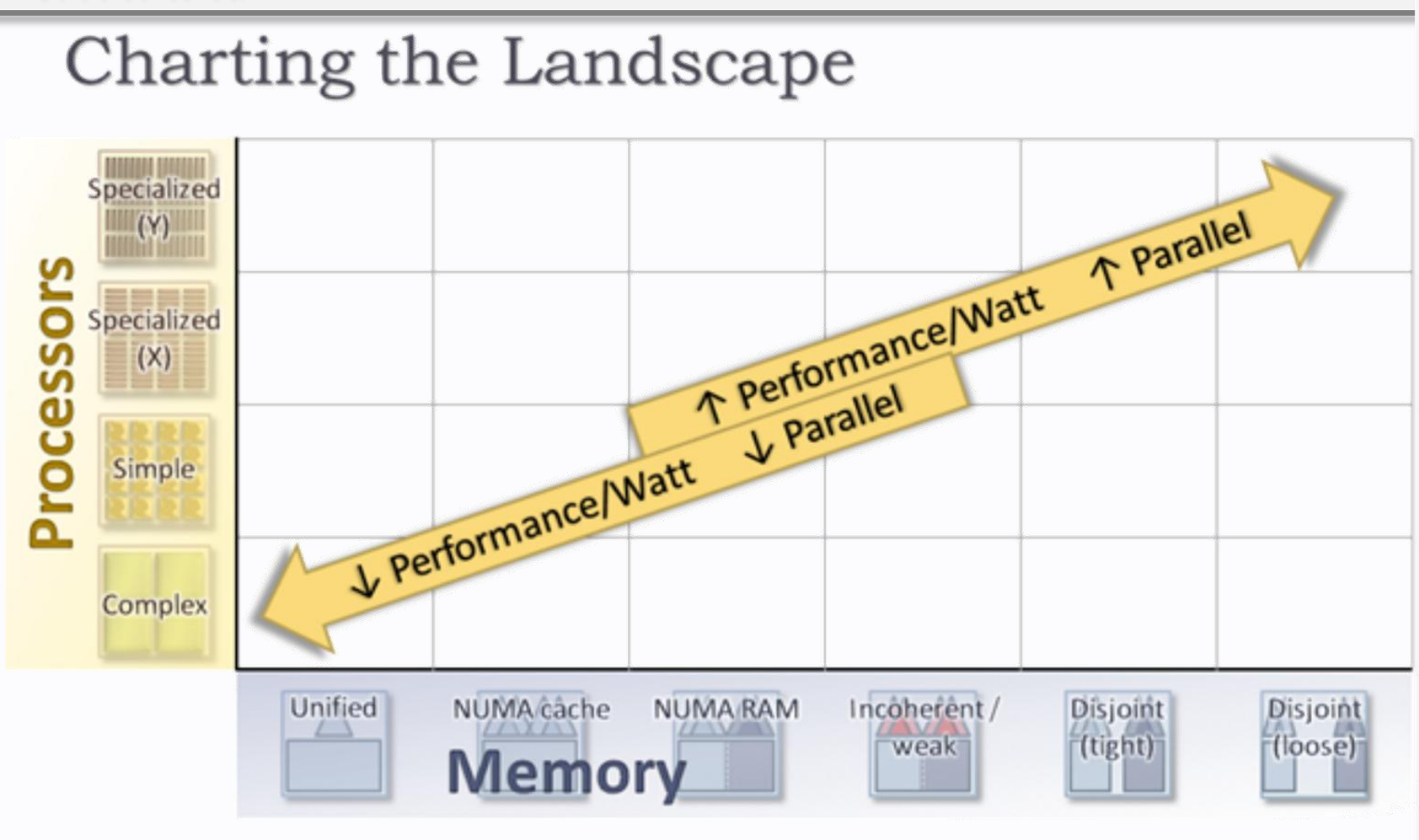
Is cache so useful ?

3rd example: CPU @ 2GHz, base CPI = 1
Miss rate/instruction = 2%
RAM access time = 100ns

1 level of cache
Miss penalty = 100ns/0.5ns = 200 cycles
Effective CPI = 1 + 0.02 × 200 = 5

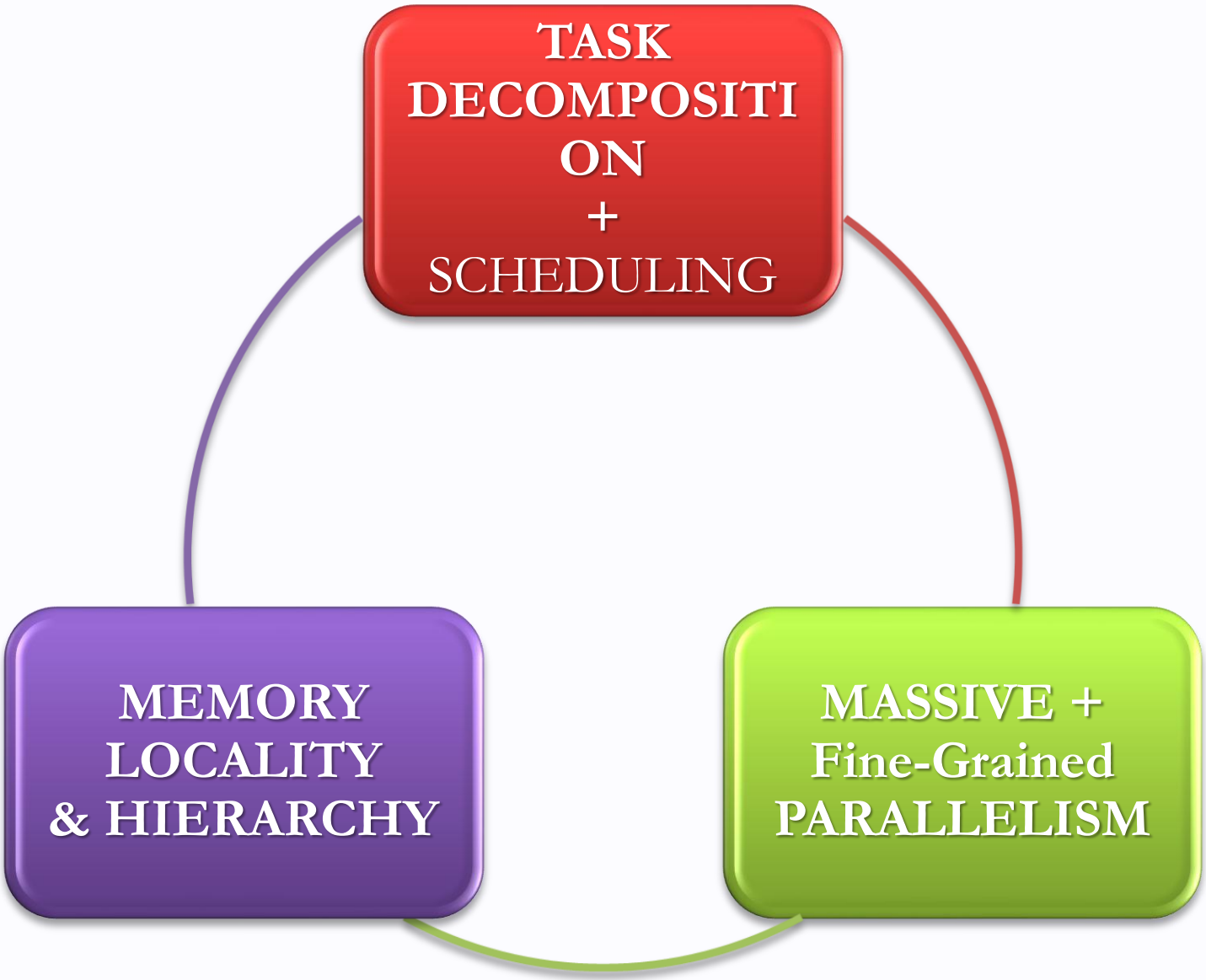
2 levels of cache
L2 access time = 5ns ; Miss rate = 1%
L1 Miss penalty = 5ns/0.5ns = 10 cycles
L2 Miss penalty = 200cycles
Effective CPI = 1 + 0.02 × 10 + 0.01 × 200 = 3.2

Outlook



By far, no more a Von Neumann machine…

Consequences for applications



Consequences for applications