Tema 2. Arquitecturas y aplicaciones paralelas

Computación de Altas Prestaciones

Carlos García Sánchez

2 de septiembre de 2022

- "Introduction to High Performance Computing for Scientists and Engineers", Georg Hager, Gerhard Wellein
- "Parallel Computer Architecture: A Hardware/Software Approach", David Culler, Jaswinder Pal Singh, Anoop Gupta

Outline

A New Golden Age for Computer Architecture

- Communications of ACM (Febrero 2019)[1]

DOI:10.1145/3282307

Innovations like domain-specific hardware, enhanced security, open instruction sets, and agile chip development will lead the way.

BY JOHN L. HENNESSY AND DAVID A. PATTERSON

A New Golden Age for Computer Architecture

WE BEGAN OUR Turing Lecture June 4, 2018[11] with a review of computer architecture since the 1960s. In addition to that review, here, we highlight current challenges and identify future opportunities, projecting another golden age for the field of computer architecture in the next decade, much like the 1980s when we did the research that led to our award, delivering gains in cost, energy, and security, as well as performance.

engineers, including ACM A.M. ing Award laureate Fred Brooks thought they could create a single

[1] A New Golden Age for Computer Architecture, John L. Hennessy, David A. Patterson Communications of the ACM, February 2019

Primeras ISAs

- En 1960 IBM tenía 4 gamas de computadores incompatibles:
  - Pequeño negocio, gran negocio, tiempo real y científico
  - Rutas de datos desde 8-bits - 64-bits
- Solución técnica
  - Compartir la misma ISA
  - **Control microprogramado** (traducción a μinstrs)
- Circuito integrado = memorias más grandes
  - ISA + complejas $\rightarrow$ CISC

CISCs vs RISC

- En la década de los 80s aparecen nuevos cuestionamientos
  - **¿Qué instrucciones genera un compilador de un código escrito en alto nivel?**

RISC

- Repertorio reducido
  - Las instrucciones no necesitan ser traducidas (μinstrs)
- La memoria para el interprete del microcódigo puede ser reemplazada
  - Pequeña cache de instrucciones
- Asignación de registros es más eficiente
  - ISA registro-registro

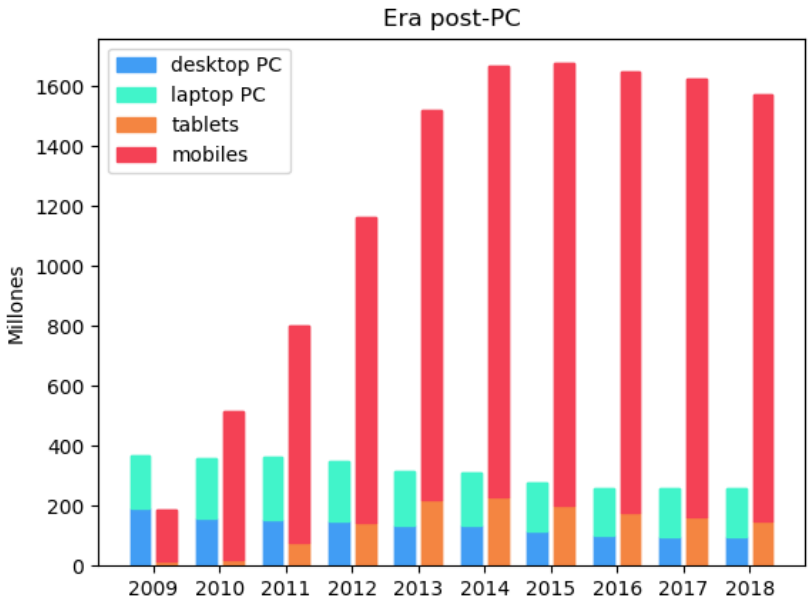
VLIW & EPIC

- Considerado el *"primo"* de los RISCs y CISCs
  - Empaquetar instrucciones sencillas en una más larga
- EPIC permite la ejecución de 2 *bundles*
  - 6 insts independientes=2LD/ST+2INT+2FP

VLIW

- Funciona bien para código estructurados (códigos *fp*)
- Decae rendimiento en *int*, o códigos menos predecibles (saltos y fallos de cache)
  - Proyecto *Itanic*
  - Aún usado en aplicaciones con saltos sencillos (Procesado Señal)

Post PC



- El pico de ventas de PCs en 2011
  - 350 millones de procesadores x86 (AMD+Intel)
- ... pero desde la irrupción del iPhone (2007)
  - SoC de los móviles foco en **eficiencia energética**

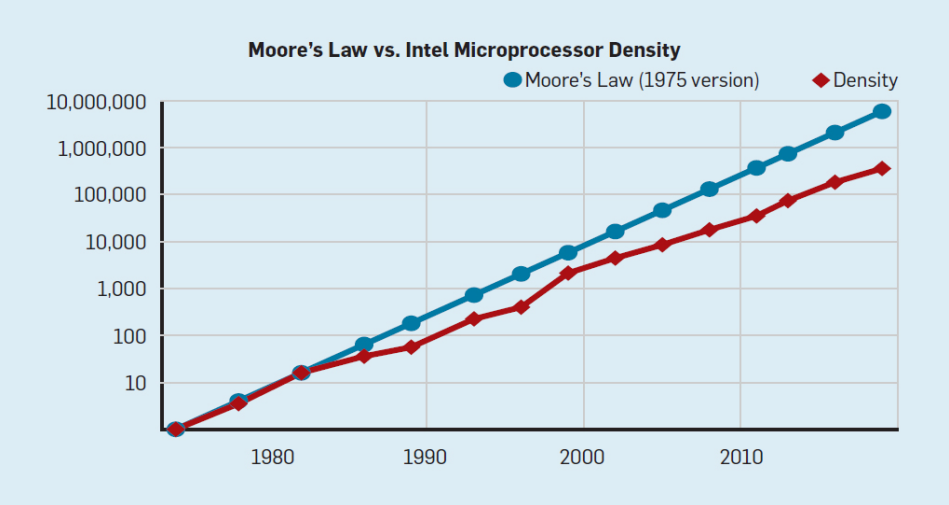
Ventas en la era post-PC

- *x86* han caido un 10 % por año desde 2011
- 20 mil Millones de ISA RISC
  - 99 % de 32-bits y 64-bits

Final de la ley de Moore

- Algo comenzó a cambiar sobre el 2000[2]
  - La distancia entre la densidad y la ley de Moore crece
- En el 2018 esta distancia es del orden de 18×

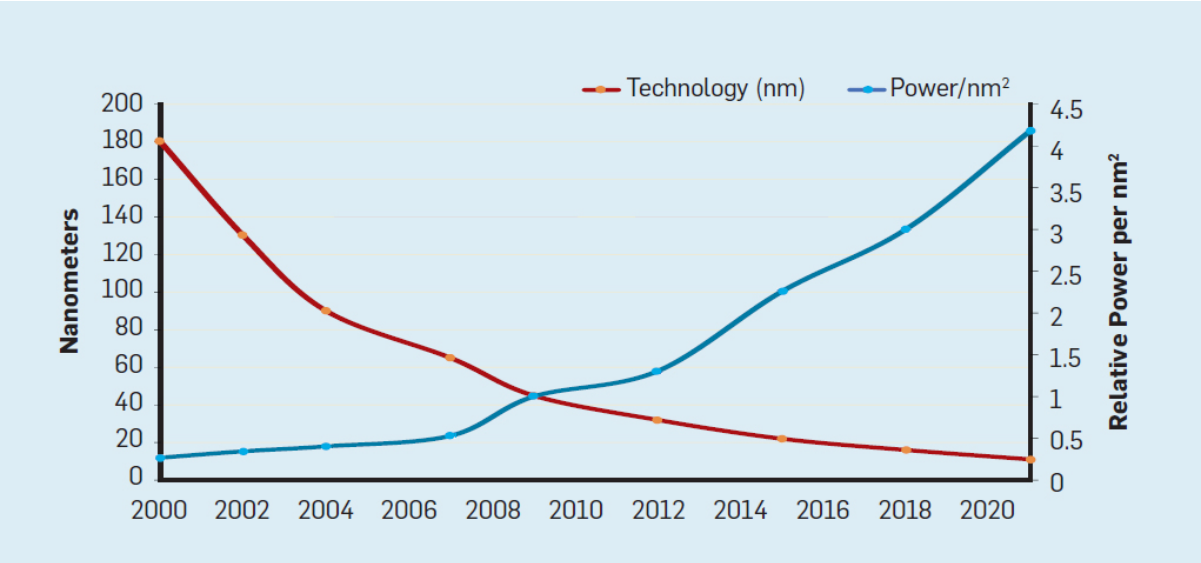
... y en los próximos años la brecha se acentúa



[2]Moore, Gordon E. "No exponential is forever: but 'Forever' can be delayed!" Solid-State Circuits Conference, 2003

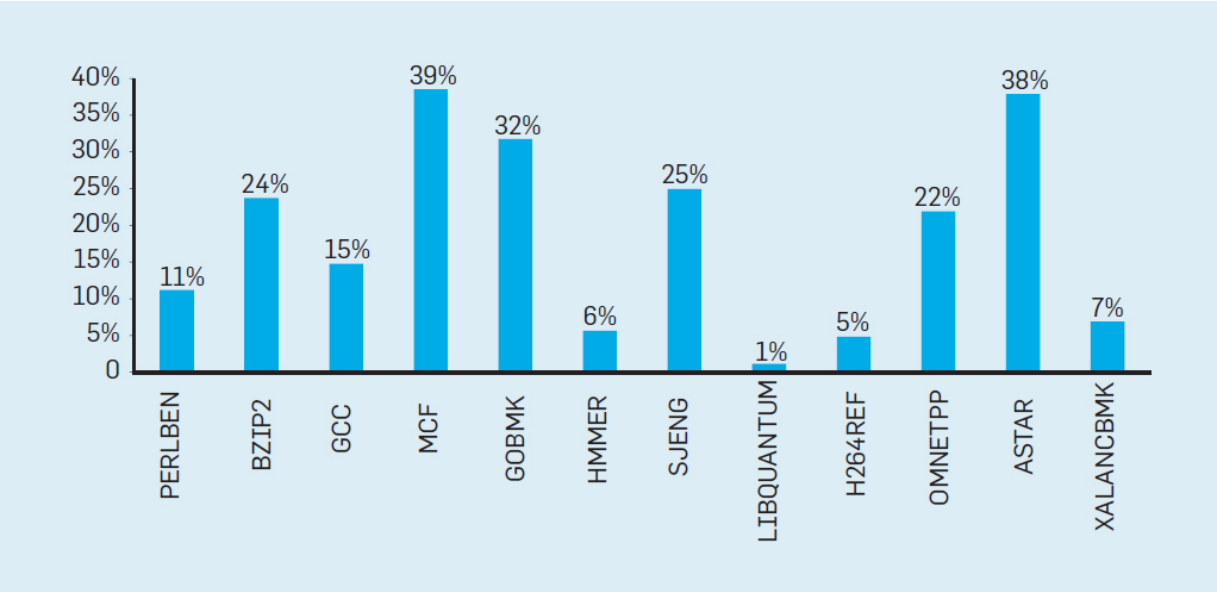
Final del escalado de Dennard

- Proyecta el consumo/área (~ constante) a mayor densidad de transistores
- Idealmente, computadores **eficientes energéticamente** en generaciones sucesivas
  - A partir del 2007 se observa un cambio importante
  - Más acuciante a partir del 2013



Consecuencias del final de ley de Moore y escalado Dennard

- ILP es la máxima entre 1986-2002
  - Incremento del 50 % rendimiento debido a mejoras ILP
- Ej: pipelines de 15 etapas y 4 instrucciones/ciclo
  - Instrucciones desechadas



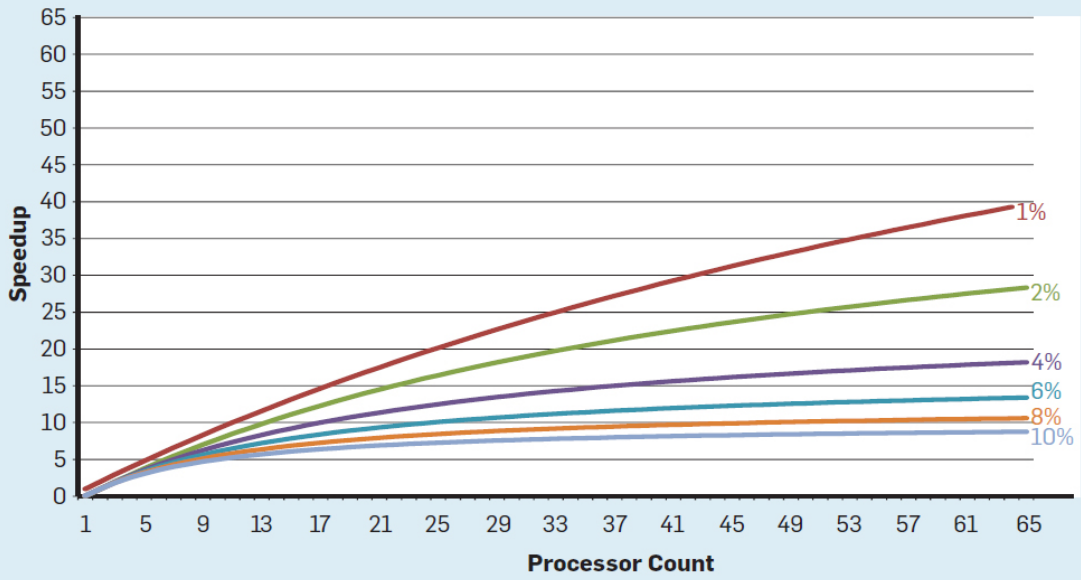
Consecuencias del final de ley de Moore y escalado Dennard

- ILP es la máxima entre 1986-2002
  - Incremento del 50 % rendimiento debido a mejoras ILP
- Ej: pipelines de 15 etapas y 4 instrucciones/ciclo
- ... pero 15/60 instrucciones son de salto (SPEC)
  - **Especulación** a expensas de **incrementar consumo**
    - Acierto: especulación mejora rendimiento
    - Fallo: deshacer instrs → energía desperdiciada
  - **Predictor de salto**
    - Si se quiere perder solamente un 10 % de tiempo → tasa acierto del 99.3 %

Era *Multicore*

- La explotación del paralelismo recae ahora en el programador
- No resuelve el problema el reto de **computación eficiente**
  - Cada core gasta energía (útil o no)

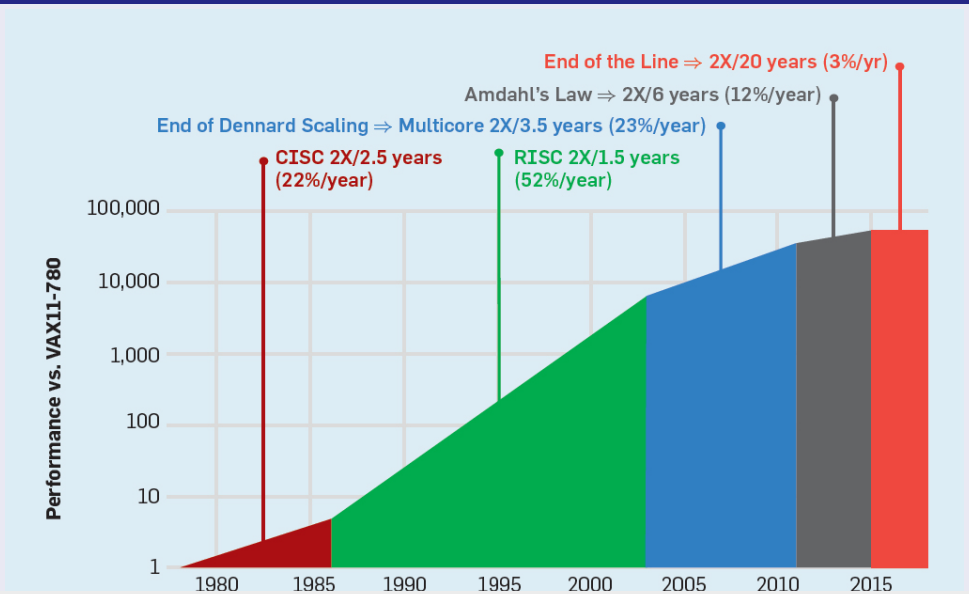
Ley de Amdahl



Resumen

- Final escalado de Dennard: incremento cores = incremento consumo
- Límite del TDP =“dark silicon” (bajada reloj o parada de cores vs sobrecalentamiento)

Mejora del rendimiento = nuevos enfoques arquitectónicos



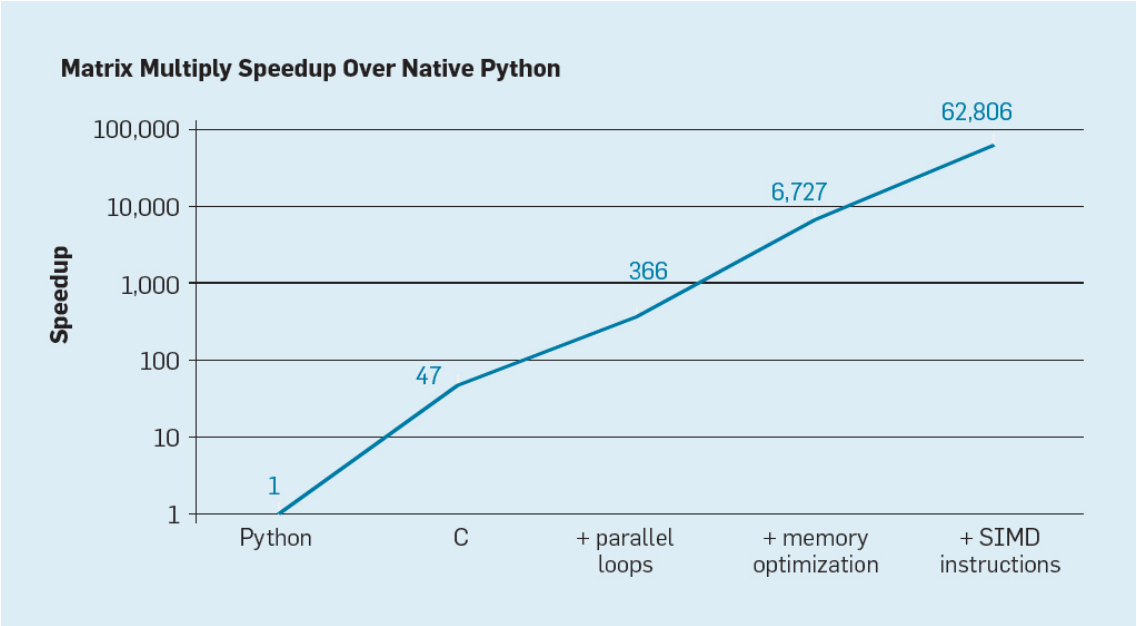
Nuevas Oportunidades

John Gardner, 1965

"What we have before us are some breathtaking opportunities disguised as insoluble problems"

- Técnicas para crear *sw* con tipos y gestión de memoria dinámicos
  - Suelen ser lenguajes ineficientes
- Especialización del *hw* para dominios de problemas específicos

Oportunidades sw



Aceleraciones[3]

- C vs. Python: 47×
- Opt. vs Pyton: 62000×

[3] Leiserson. There's plenty of room at the top

Oportunidades hw

Arquitecturas de Dominio específico (DSAs)

Comúnmente conocidos como aceleradores

- ... las populares GPUs
- Procesadores para redes neuronales
- DSAs explotan el **paralelismo** de forma eficiente
- DSAs gestión de memoria más forma eficiente

Oportunidades hw (DSAs)

Paralelismo

- SIMD más eficiente que MIMD
  - Aunque SIMD menos flexible
- VLIW más eficiente que Superescalar fuera orden
  - Más eficiente = control más simple

Oportunidades hw (DSAs)

Memoria

- Acceso a memoria es muy costoso
  - Acceder a bloque cache 32Kb = 200×  sumador 32 bits
- Caches consumen 1/2 procesador

Desventajas de las caches

- *datasets* grandes no funcionan bien
  - Baja localidad espacial y temporal
- Si funcionan bien (localidad alta), mayoría cache sin usar

Oportunidades hw (DSAs)

Precisión

- CPUs: *fp* 32-bits y 64-bits
  - Inferencia de DNNs: *int* 4, 8, 16-bits es suficiente
  - Entrenamiento de DNNs: *fp* 16- o 32-bits suficiente

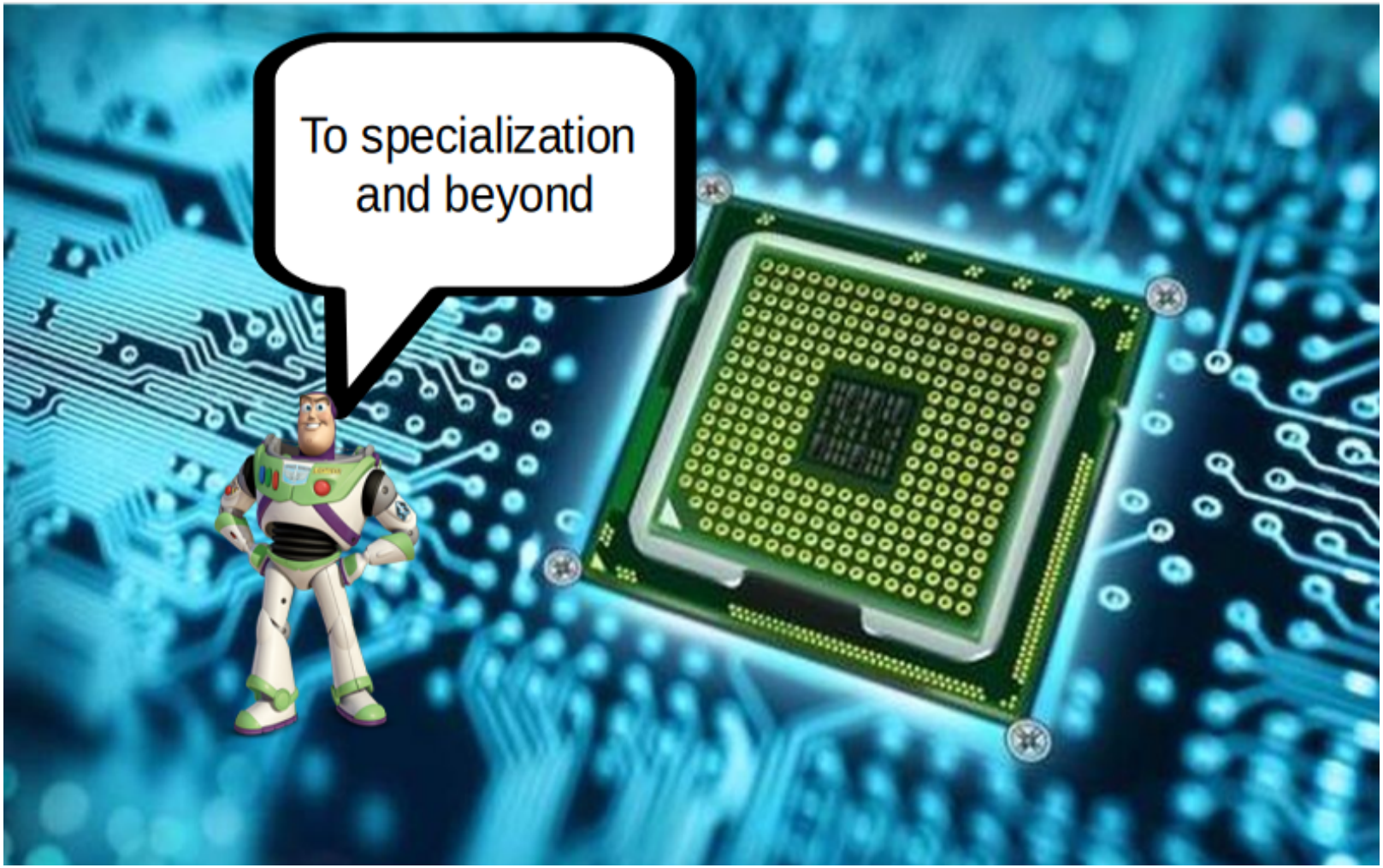
Nuevas oportunidades arquitectónicas

- **GPUs**: Nvidia usa muchos cores, hilos en *vuelo* y caches
- **TPUs**: ASIC con matriz sistólica de multiplicadores (256x256) de 8-bits
- **FPGAs**: Microsoft ha desarrollado soluciones de DNNs en Azure basadas en FPGA
- **CPUs**: Intel ofrece SIMD con baja precisión: AVX512_VNNI (FMA con mults 8-bits)

Especialización del hw

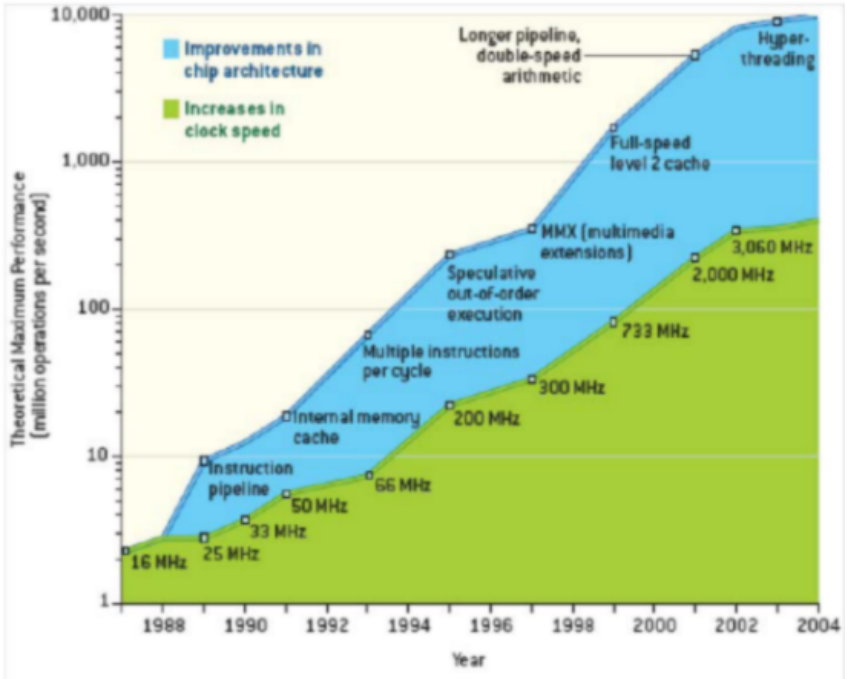
- Aceleradores destinados al procesamiento DNNs usados en otro contexto

... hacia la especialización



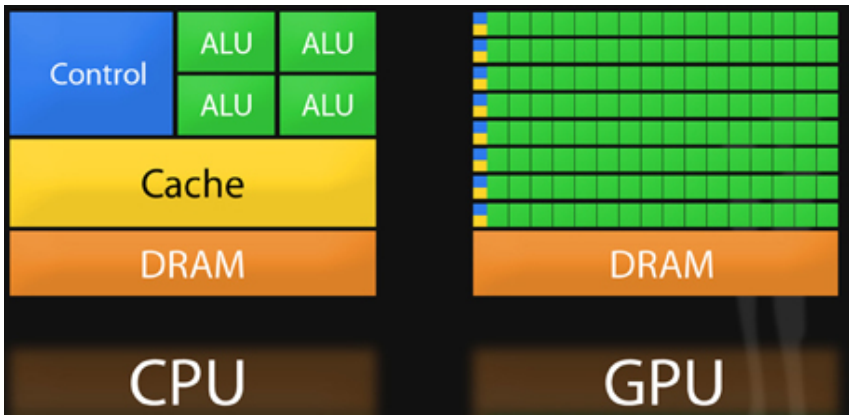
Especialización en μarquitectura

- Con la ley de Moore y el "excedente" tecnológico, ya se ha ido especializado en el GPP



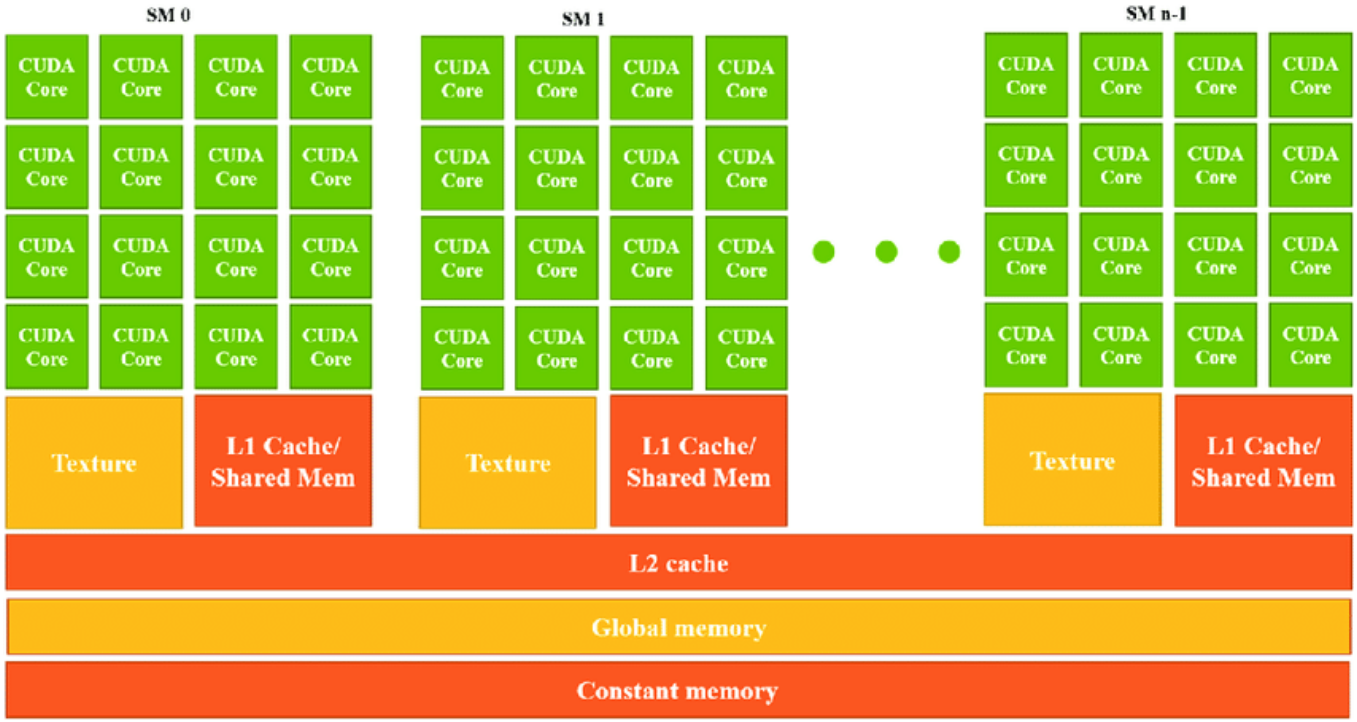
Especialización en μarquitectura (GPUs)

- Silicio destinado a lógica (cores) vs caches (GPPs)
- GPUs basadas en procesadores de fujos (localidad temporal)
- Muchos hilos en vuelo, pero...
  - ¿Como se mitiga el *memory-wall*?



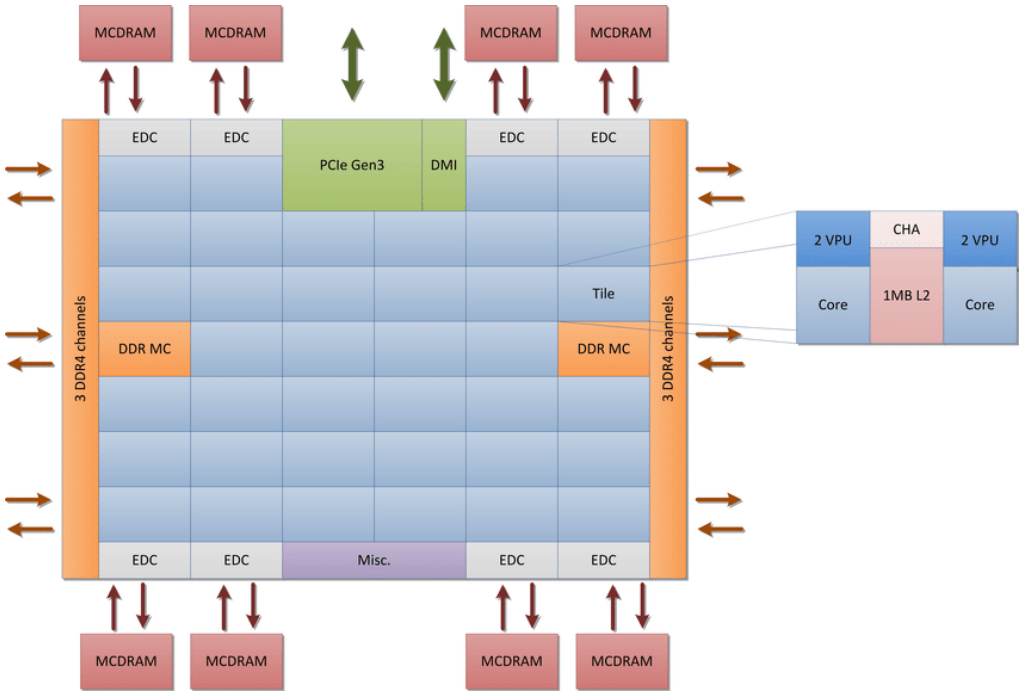
Especialización en μarquitectura (GPUs)

- NVIDIA G80, arquitectura: *StreamMultiprocessors* vs *CUDA-core*



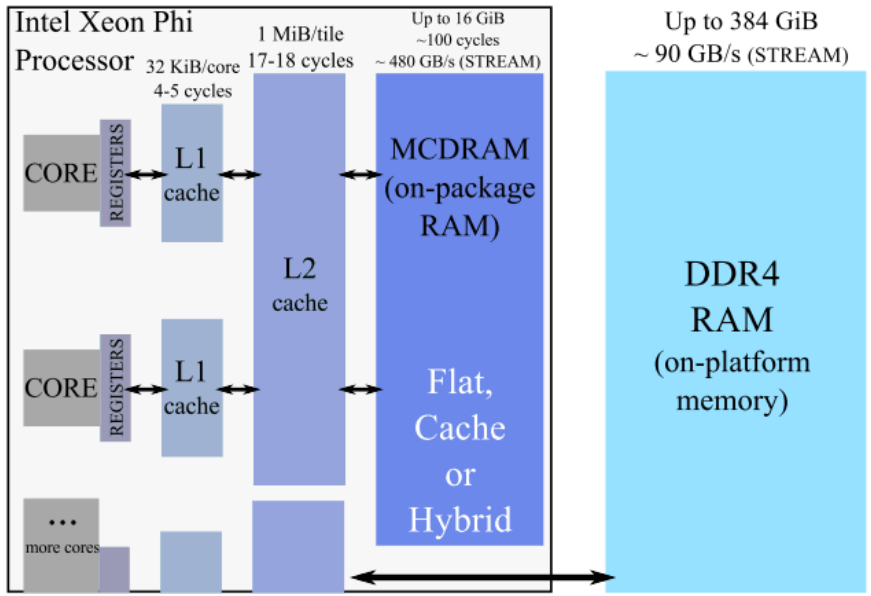
Especialización en μarquitectura (Xeon-KNL)

- 36 *Tiles* conectada en una malla 2D (2 cores)
  - Cada core es un *fork* de Intel Atom *ooo*
  - 2 cores (1 legacy: compatiblidad *x86*) con 1 VPU



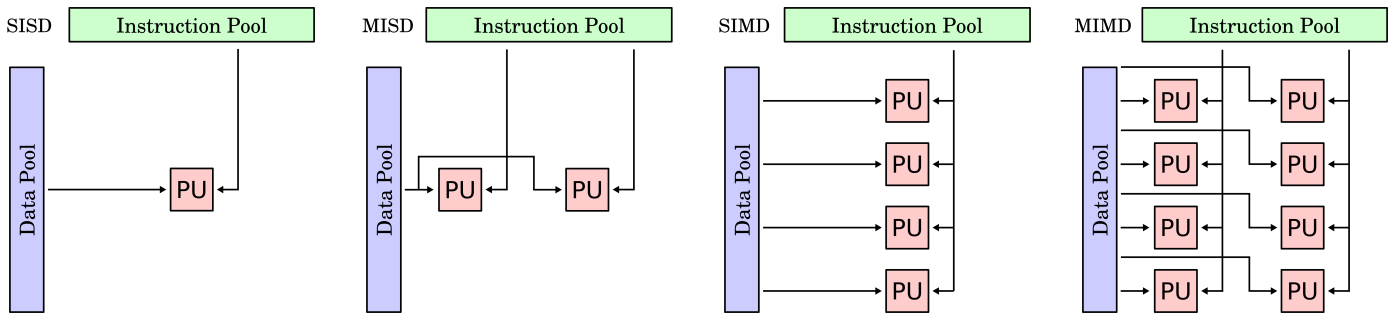
Especialización en μarquitectura (Xeon-KNL)

- Memoria de MCDRAM: High-Bandwidth Memory (8x2GBytes)
  - 450GB/s vs 90GB/s DDR4
  - Modos: *cache*, *flat (numa-shared)*, *hybrid*



Taxonomía de Flynn

- Clasificación de arquitecturas de computadores propuesta por Michael J. Flynn
  - Instrucciones vs Datos

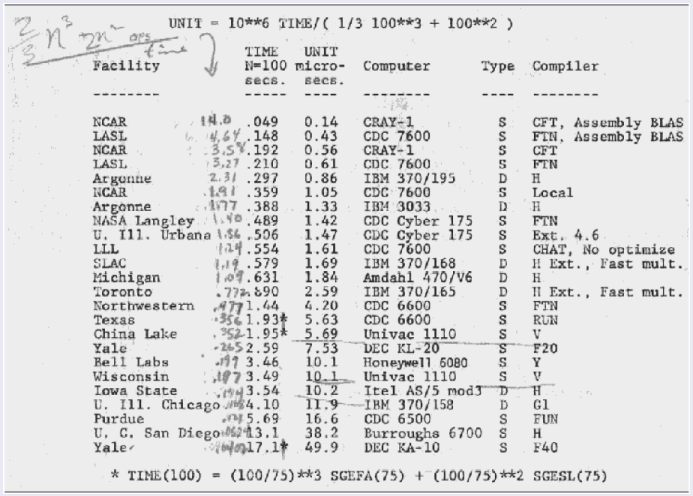


Algunas listas

top500

`www.top500.org` mantiene una lista de las computadoras más rápidas en el mundo, de acuerdo con un programa de referencia particular. Sale una nueva lista cada junio y noviembre.

- Desde 1979... la primera lista



UNIT = 10**6 TIME/(1/3 100**3 + 100**2)

Facility	TIME N=100 secs.	UNIT micro-secs.	Computer	Type	Compiler
NCAR	.049	0.14	CRAY-1	S	CFT, Assembly BLAS
LASL	.148	0.43	CDC 7600	S	FTN, Assembly BLAS
NCAR	.192	0.56	CRAY-1	S	CFT
LASL	.210	0.61	CDC 7600	S	FTN
Argonne	.297	0.86	IBM 370/195	D	H
NCAR	.359	1.05	CDC 7600	S	Local
Argonne	.388	1.33	IBM 3033	D	H
NASA Langley	.489	1.42	CDC Cyber 175	S	FTN
U. Ill. Urbana	.506	1.47	CDC Cyber 175	S	Ext. 4.6
LLL	.554	1.61	CDC 7600	S	CHAT, No optimize
SLAC	.579	1.69	IBM 370/168	D	H Ext., Fast mult.
Michigan	.631	1.84	Amdahl 470/V6	D	H
Toronto	.890	2.59	IBM 370/165	D	H Ext., Fast mult.
Northwestern	1.44	4.20	CDC 6600	S	FTN
Texas	1.93*	5.63	CDC 6600	S	RUN
China Lake	1.95*	5.69	Univac 1110	S	V
Yale	2.59	7.53	DEC KL-20	S	F20
Bell Labs	3.46	10.1	Honeywell 6080	S	Y
Wisconsin	3.49	10.1	Univac 1110	S	V
Iowa State	3.54	10.2	Itel AS/5 mod3	D	H
U. Ill. Chicago	4.10	11.9	IBM 370/158	D	G1
Purdue	5.69	16.6	CDC 6500	S	FUN
U. C. San Diego	13.1	38.2	Burroughs 6700	S	H
Yale	17.1*	49.9	DEC KA-10	S	F40

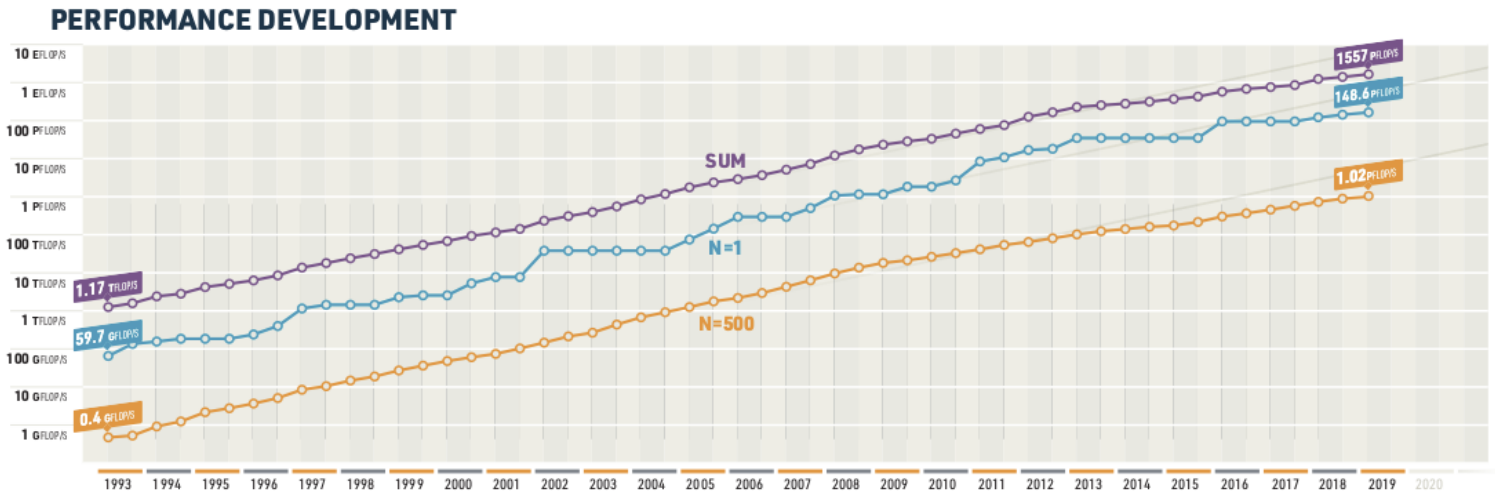
* TIME(100) = (100/75)**3 SGEFA(75) + (100/75)**2 SGESL(75)

top500 (Jun 2021)

RANK	SITE	SYSTEM
1	Supercomputer Fugaku Riken Center Japan	Fugaku ARM-A64FX (48Cores por nodo)+SIMD512b **158976 nodos**
2	DOE/SC/Oak Ridge National Lab. USA	Summit, **4608 nodes** 2xIBM Power9(22c)+6xNVIDIA Volta V100s
3	DOE/NNSA/LLNL USA	Sierra **4474 nodes** 2xIBM Power9(22c)+6xNVIDIA Volta V100
4	National Supercomputing Center in Wuxi China	SunWay 26010, **260 Cores**

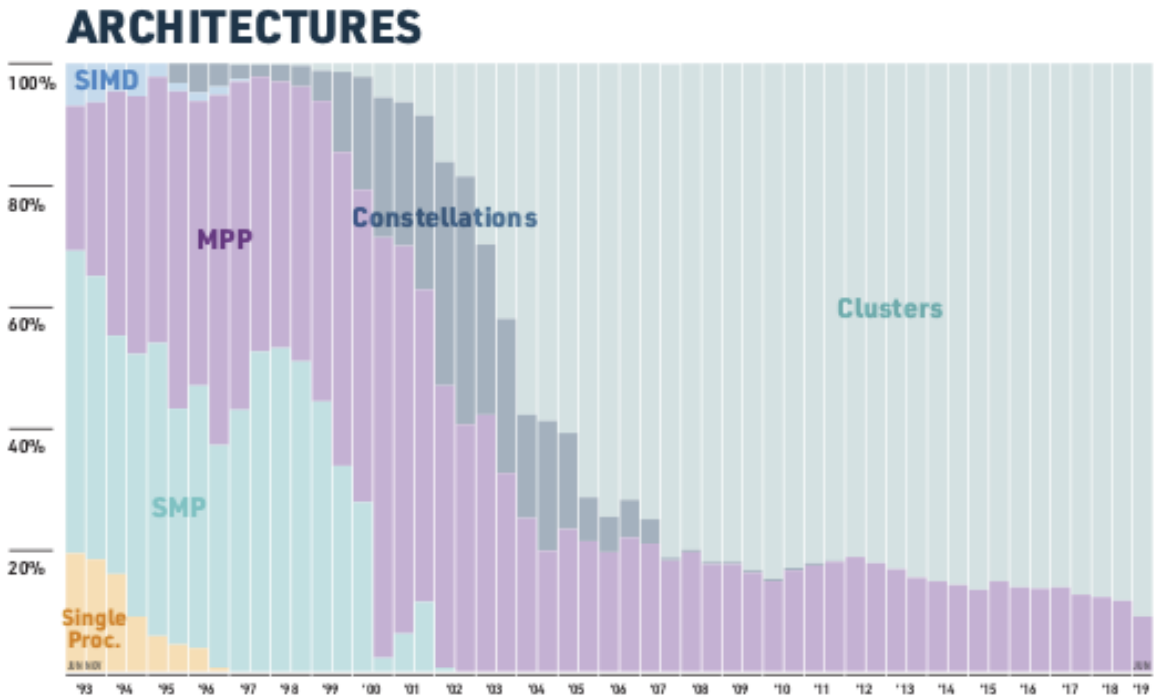
top500 (Jun 2019)

- Tendencia al crecimiento exponencial (SUM)
  - ... pero parece que disminuye a partir del 2013



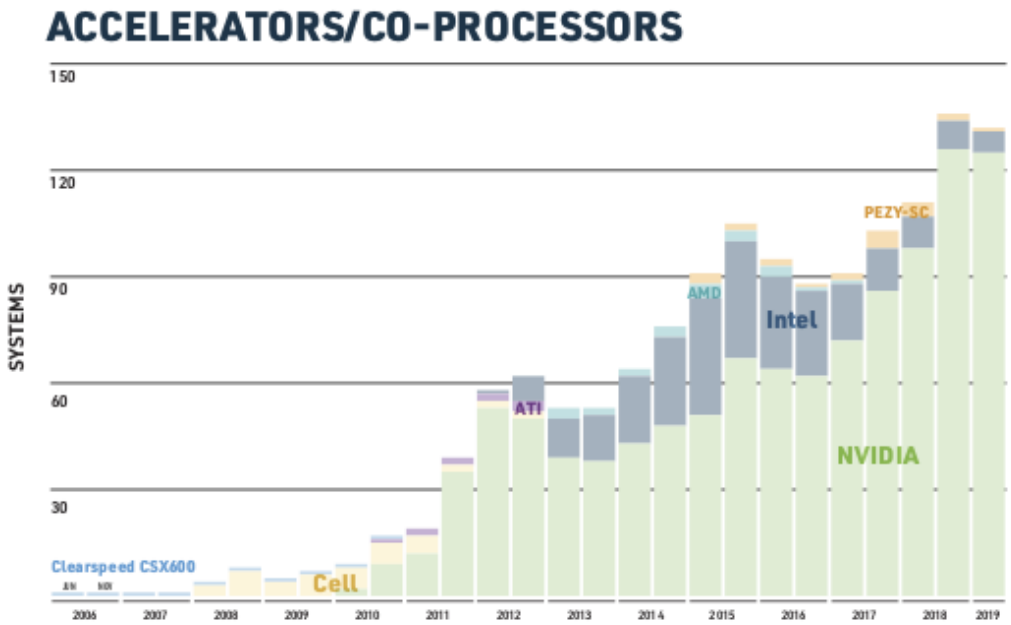
top500 (Jun 2019)

- Mayoría de arquitecturas basada en cluster... aunque algo de MPP



top500 (Jun 2019)

- Irrupción de aceleradores (GPUs y Xeon Phi) para crecer el rendimiento



Algunas listas

green500

`www.green500.org` concepto de eficiencia energética. La lista es una clasificación las supercomputadoras listadas en el TOP500 desde el punto de vista de la eficiencia energética. Para ello utiliza una medida de potencia por vatio: "FLOPS-per-Watt". Desde el SC09 aparece la lista tanto en Jun y Nov.

Grados de paralelismo

- **Massively Parallel**: se refiere al hardware que comprende un sistema paralelo dado, que tiene muchos procesadores/cores
  - El significado de "muchos" sigue aumentando, pero actualmente, las computadoras paralelas más grandes pueden estar formadas por procesadores numeración en los cientos de miles (ejemplo: GPU)
- **Embarrassingly Parallel**: resolver muchas tareas similares, pero independientes simultáneamente; poca o ninguna necesidad de coordinación entre las tareas
  - Grado de paralelismo evidente
- **Scalability**: se refiere a la capacidad de un sistema paralelo para demostrar un aumento proporcional en aceleración paralela con la adición de más procesadores. Idealmente sería alcanzar el speedup ideal

Paso 1: Identificación paralelismo

- **Importante identificar el máximo paralelismo posible**

Descomposición funcional (*task parallelism*)

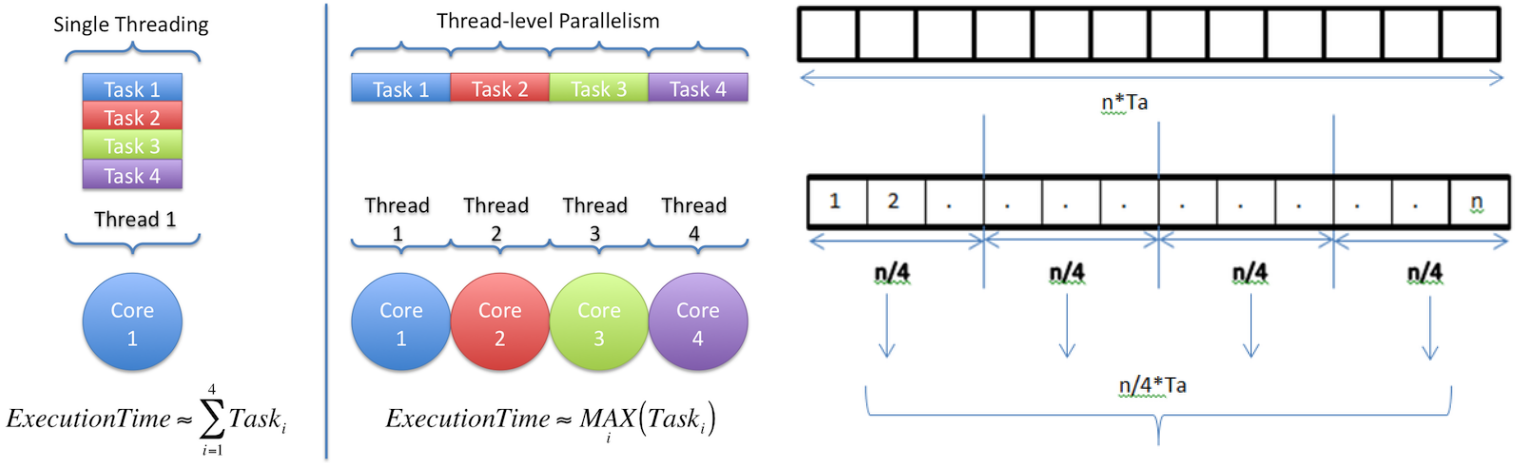
- Que partes de nuestra aplicación pueden ejecutarse en paralelo

Descomposición de datos (*data parallelism*)

- Habitualmente descrita mediante bucles

Paso 1: Identificación paralelismo

- *task parallelism* vs *data parallelism*



Paso 2: Eligiendo la granularidad correcta

- Normalmente se puede elegir el *tamaño* de la descomposición del problema: *trozos pequeños* vs *trozos grandes*
- Idealmente elegiriamos grano **grueso**
  - Produce menos overheads
  - Suele haber menos comunicaciones y sincronizaciones
  - **... pero puede producir desbalanceos de carga apreciables**
- Usar grano **fino**
  - Menor desbalanceo de carga
  - **... pero mayores overheads, comunicación y sincronización**

Paralelismo Multi-nivel

- Explotar ambos niveles grueso y fino
  - Incluso utilizan diferentes modelos de programación

Paso 3: Paralelización

- Elegir uno de los modelo de paralelización
- ... o combinarlos para mejorar el rendimiento de la aplicación

Paso 4: Evaluación-Resolver problemas

Resultados incorrectos

- Paralelización incorrecta
- Condiciones de carrera
- Interbloqueos o deadlocks
- Irreproducibilidad

Rendimiento pobre

- Desbalanceo de carga
- Falsa compartición (*false sharing*)
- Serialización
- Poca localidad

False sharing

- Compartición verdadera:
  - Un dato es compartido por dos hilos (en dos cores/procs)
  - Uno de los hilos escribe en ese dato: **protocolo de coherencia**
- Pero cuando un hilo escribe en una *línea de cache* (invalida línea)
  - ... y otro hilo (en otro core/procs) lee de la misma aunque otra posición de memoria