FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS
UNIVERSIDAD DE CHILE

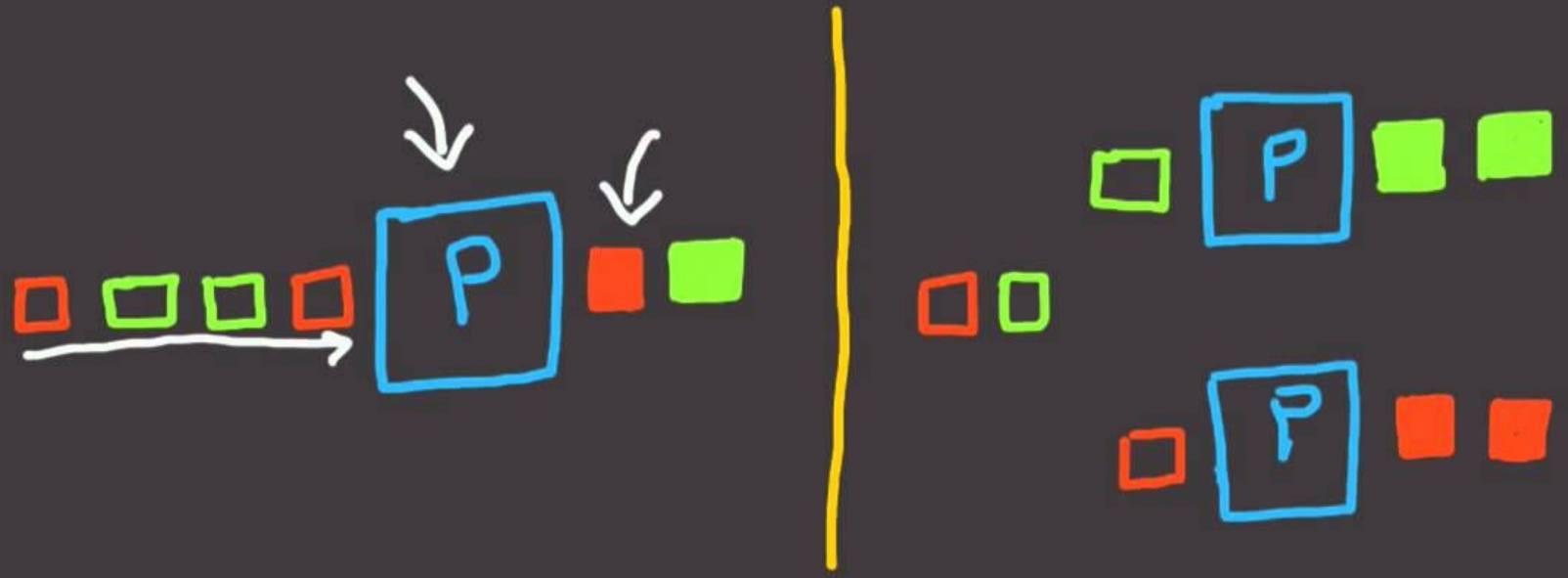
Programación Paralela

Joaquín Torres Paris

¿Qué es la programación paralela?

Programa que ejecuta una serie de operaciones en paralelo (al mismo tiempo)

Parallel Computing



¿Qué es la programación paralela?

Puede convertir la ejecución a una no determinista.

Toma ventaja de las arquitecturas actuales.

Hay que mirar los problemas desde otro punto de vista.

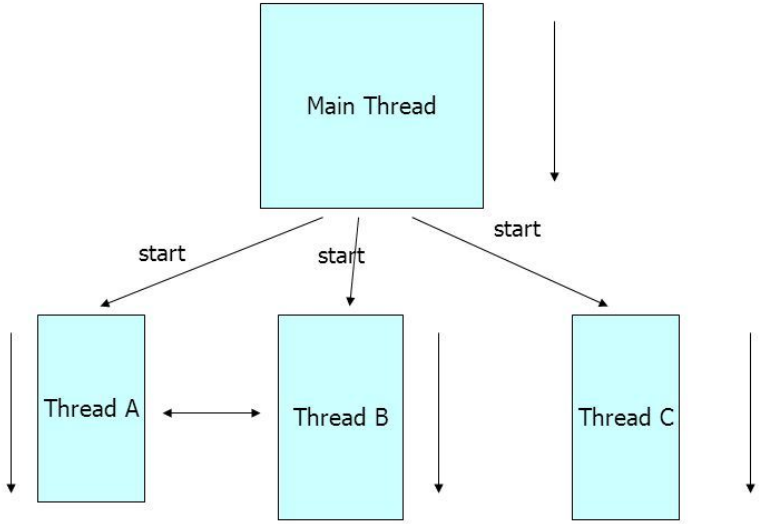
Nociones básicas

Task: Secuencia de instrucciones que deben ejecutarse secuencialmente.

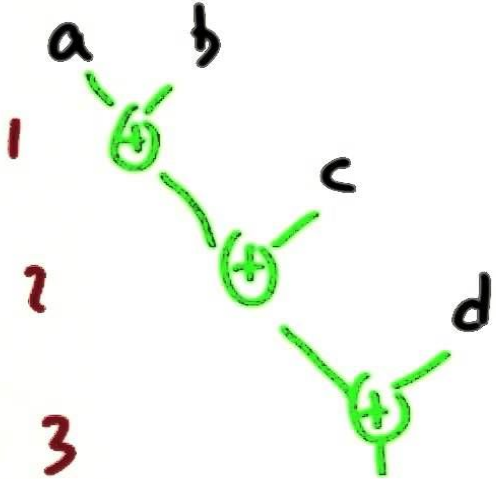
Ejecución concurrente: Múltiples tareas independientes que **pueden** ejecutarse simultáneamente. Si dos tareas son dependientes entonces no son concurrentes.

Nociones básicas

A Multithreaded Program



Serial Reduce

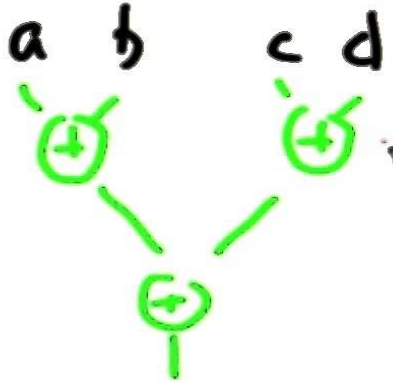


3 work

steps

$((a+b)+c)+d$

Parallel Reduce



3 work

steps

$(a+b)+(c+d)$

¿Qué ganamos con el paralelismo?

Una solución más rápida.

Resolver problemas más grandes.

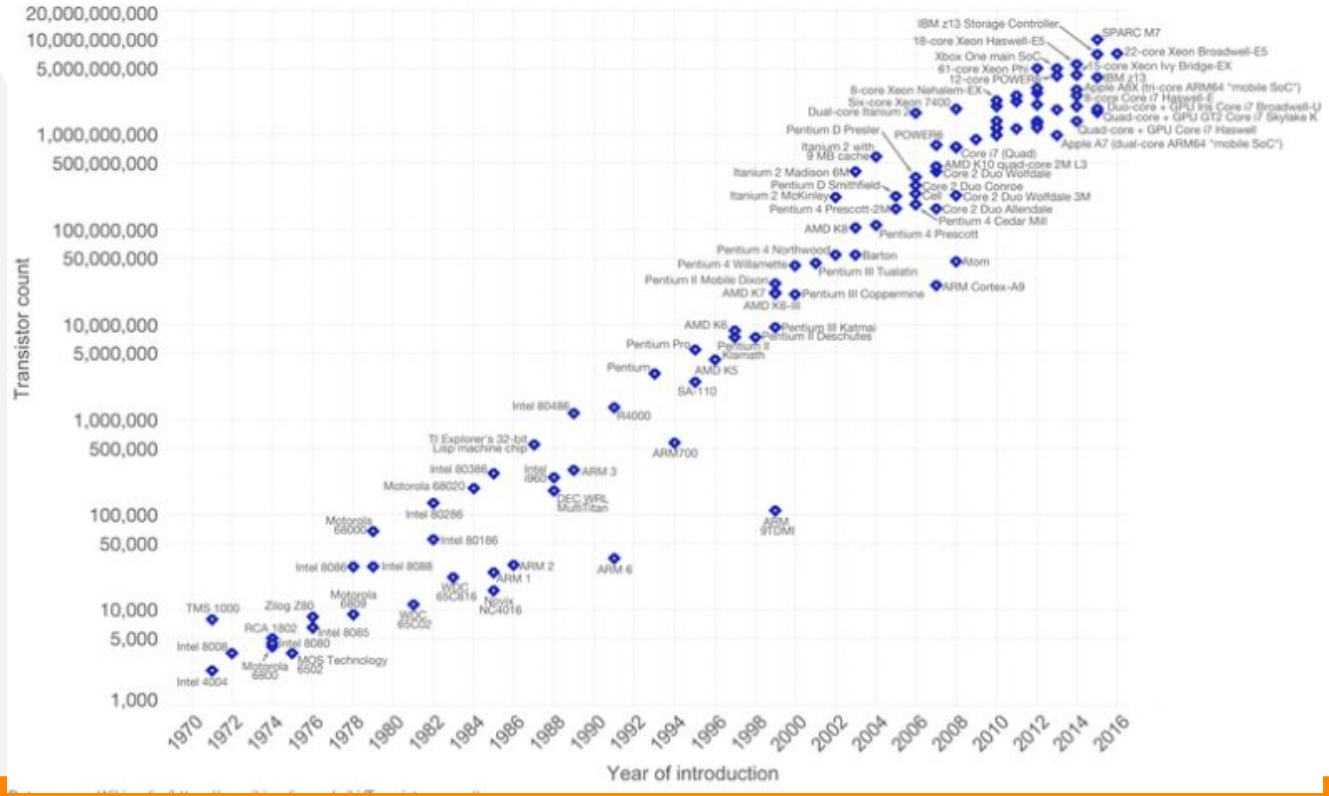
Uso efectivo de los recursos del computador.

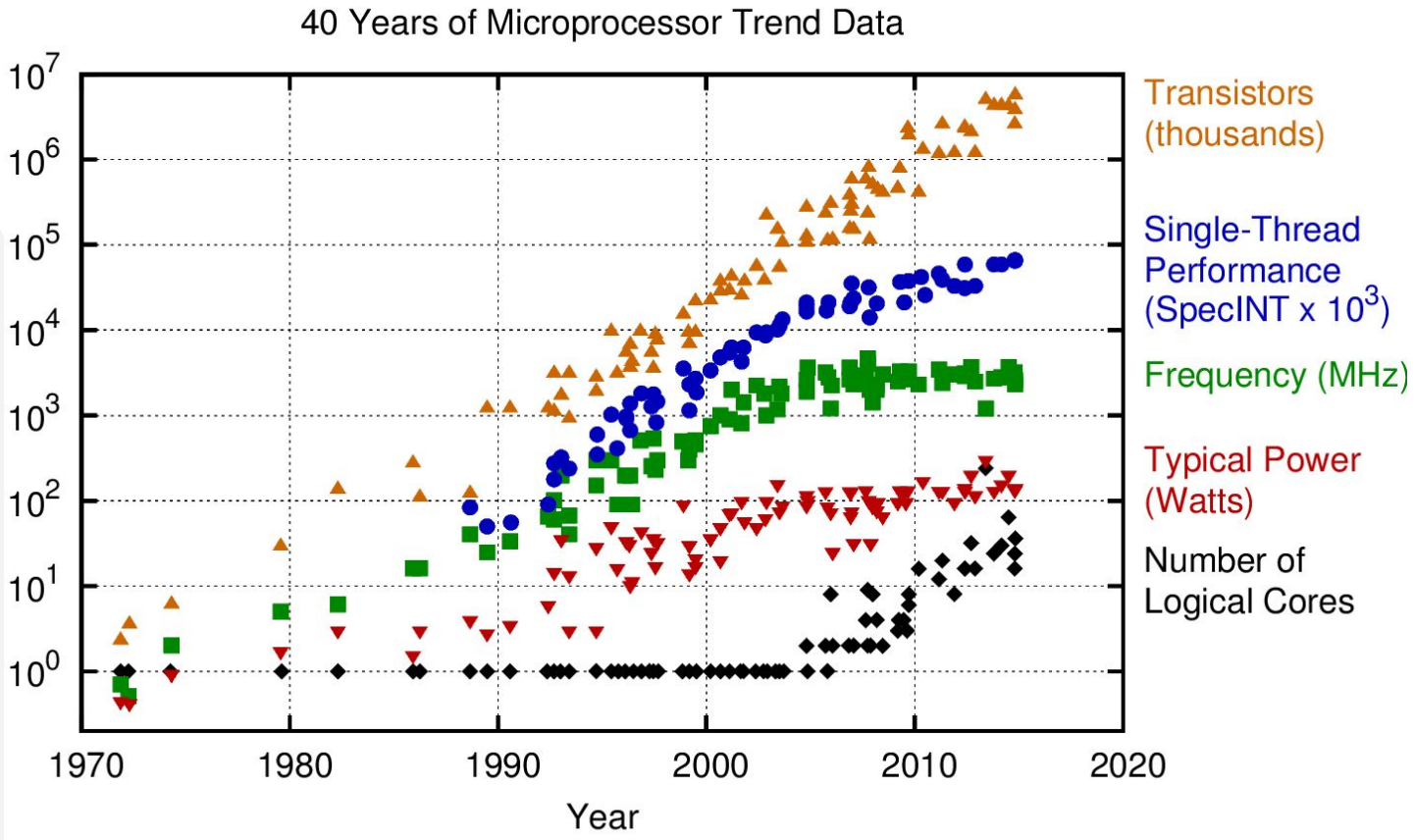
¿Para qué usar paralelismo si podemos mejorar los procesadores?

Intel Core-X Series (Kabylake-X, Skylake-X)							
Processor	Cores/ Threads	L3 Cache	PCIe Lanes	Base Clock	Turbo Clock 2.0	Turbo Clock 3.0	Launch
Core i9-7920X	12C/24T	16.5 MB	44	TBD	TBD	TBD	August
Core i9-7900X	10C/20T	13.75 MB	44	3.3 GHz	4.3 GHz	4.5 GHz	June
Core i9-7820X	8C/16T	11 MB	28	3.6 GHz	4.3 GHz	4.5 GHz	June
Core i9-7800X	6C/12T	8.25 MB	28	3.5 GHz	4.0 GHz	-	June
Core i7-7740K	4C/8T	8 MB	16	4.3 GHz	4.5 GHz	-	June
Core i7-7640K	4C/4T	6 MB	16	4.0 GHz	4.2 GHz	-	June

¿Qué nos impide aumentar la cantidad de transistores?

Moore's Law

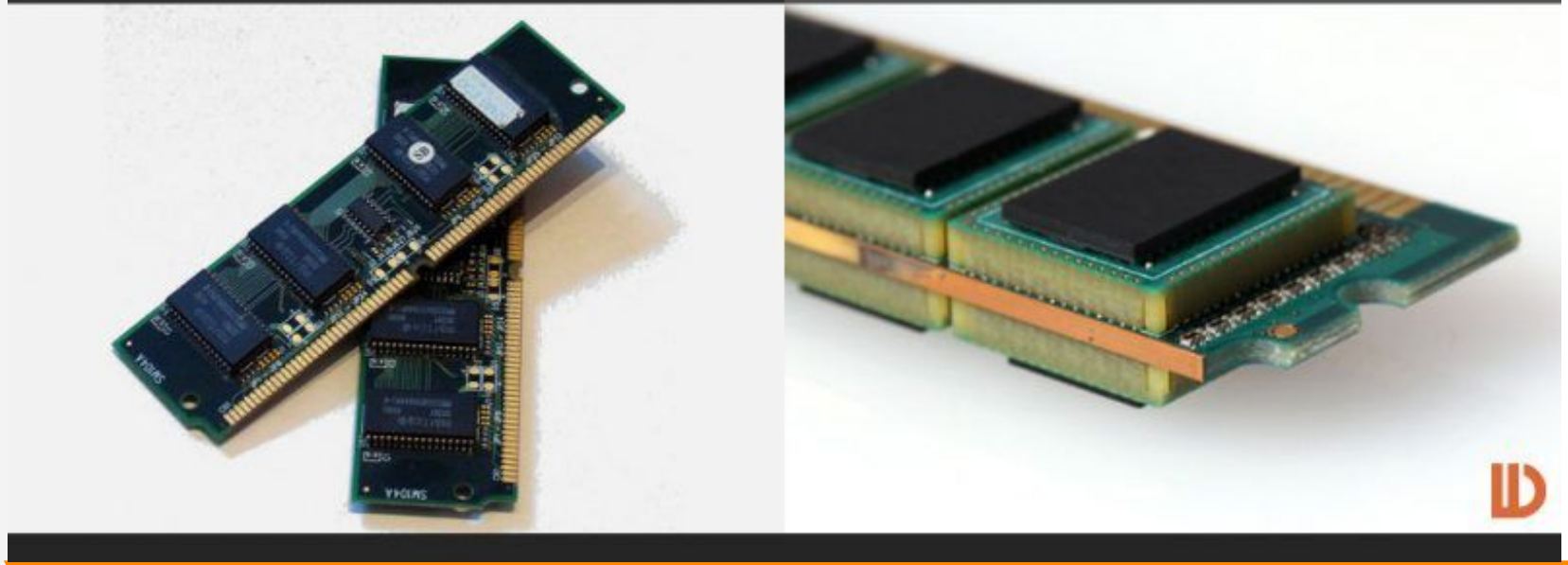

Transistor count
20,000,000,000
10,000,000,000
5,000,000,000
1,000,000,000
500,000,000
100,000,000
50,000,000
10,000,000
5,000,000
1,000,000
500,000
100,000
50,000
10,000
5,000
1,000
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
Year of introduction
IBM z13 Storage Controller
SPARC M7
18-core Xeon Haswell-E5
22-core Xeon Broadwell-E5
Xbox One main SoC
61-core Xeon Phi
15-core Xeon Ivy Bridge-EX
12-core POWER8
IBM z13
8-core Xeon Nehalem-EX
Apple A8X (tri-core ARM64 "mobile SoC")
Six-core Xeon 7400
6-core Core i7 Haswell-E
Dual-core Itanium 2
Dual-core + GPU Iris Core i7 Broadwell-U
Quad-core + GPU GT2 Core i7 Skylake K
Pentium D Presler
POWER6
Quad-core + GPU Core i7 Haswell
Itanium 2 with 9 MB cache
Apple A7 (dual-core ARM64 "mobile SoC")
Core i7 (Quad)
AMD K10 quad-core 2M L3
Itanium 2 Madison 6M
Core 2 Duo Wolfdale
Pentium D Smithfield
Core 2 Duo Conroe
Itanium 2 McKinley
Cell
Core 2 Duo Wolfdale 3M
Pentium 4 Prescott-2M
Core 2 Duo Allendale
Pentium 4 Cedar Mill
AMD K8
Pentium 4 Prescott
Pentium 4 Northwood
Barton
Atom
Pentium 4 Willamette
Pentium III Tualatin
Pentium II Mobile Dixon
ARM Cortex-A9
AMD K7
Pentium III Coppermine
AMD K6-III
AMD K6
Pentium III Katmai
Pentium II Deschutes
Pentium Pro
Pentium II
Klamath
Pentium
AMD K5
SA-110
Intel 80486
R4000
TI Explorer's 32-bit Lisp machine chip
ARM700
Intel 80386
Intel i960
ARM 3
Motorola 68020
DEC WRL MultiTitan
Intel 80286
ARM 9TDMI
Motorola 68000
Intel 80186
Intel 8086
Intel 8088
ARM 2
ARM 6
ARM 1
WDC 65C816
Motorola 6809
Novix NC4016
TMS 1000
Zilog Z80
WDC 65C02
RCA 1802
Intel 8085
Intel 8008
Intel 8080
MOS Technology 6502
Motorola 6800
Intel 4004



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

¿Cuáles son las memorias del procesador?

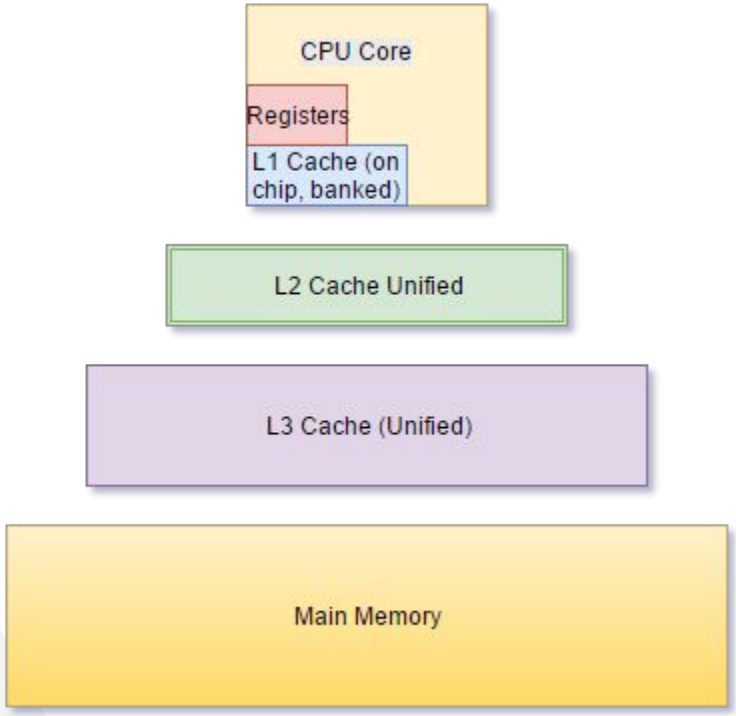
SRAM VS DRAM



¿Qué tipos existen?

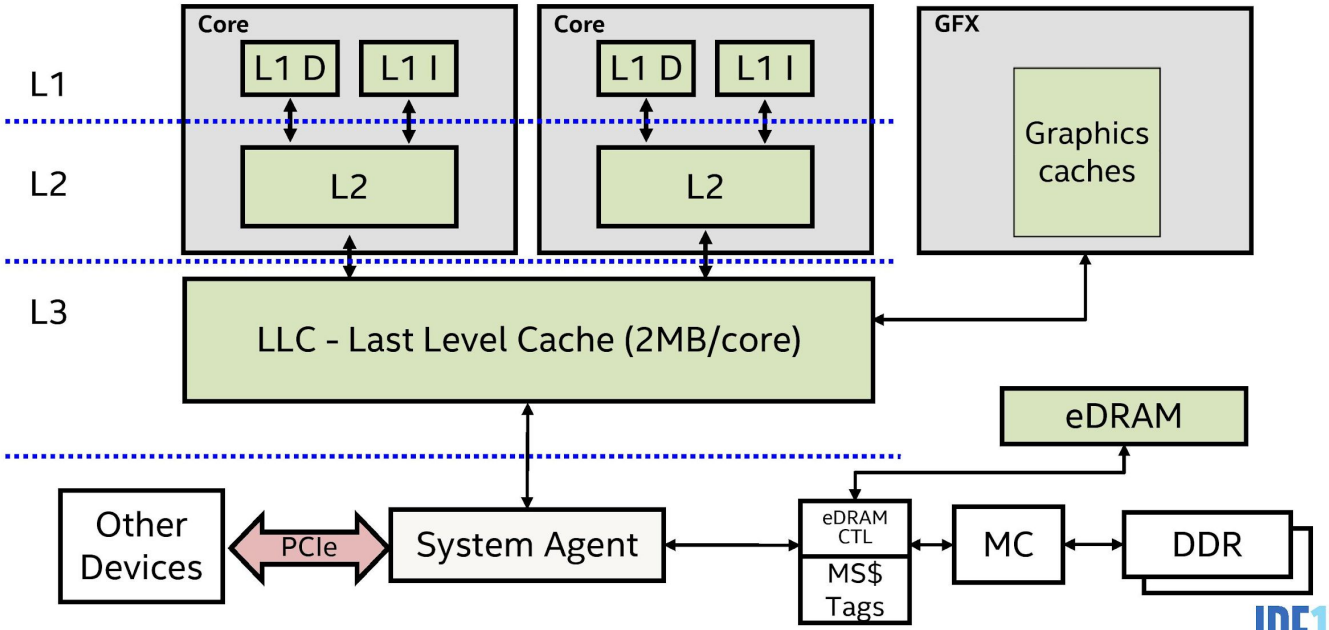
Memory technology	Typical access time	$ per GB in 2004
SRAM	0.5–5 ns	$4000–$10,000
DRAM	50–70 ns	$100–$200
Magnetic disk	5,000,000–20,000,000 ns	$0.50–$2

Niveles de memoria

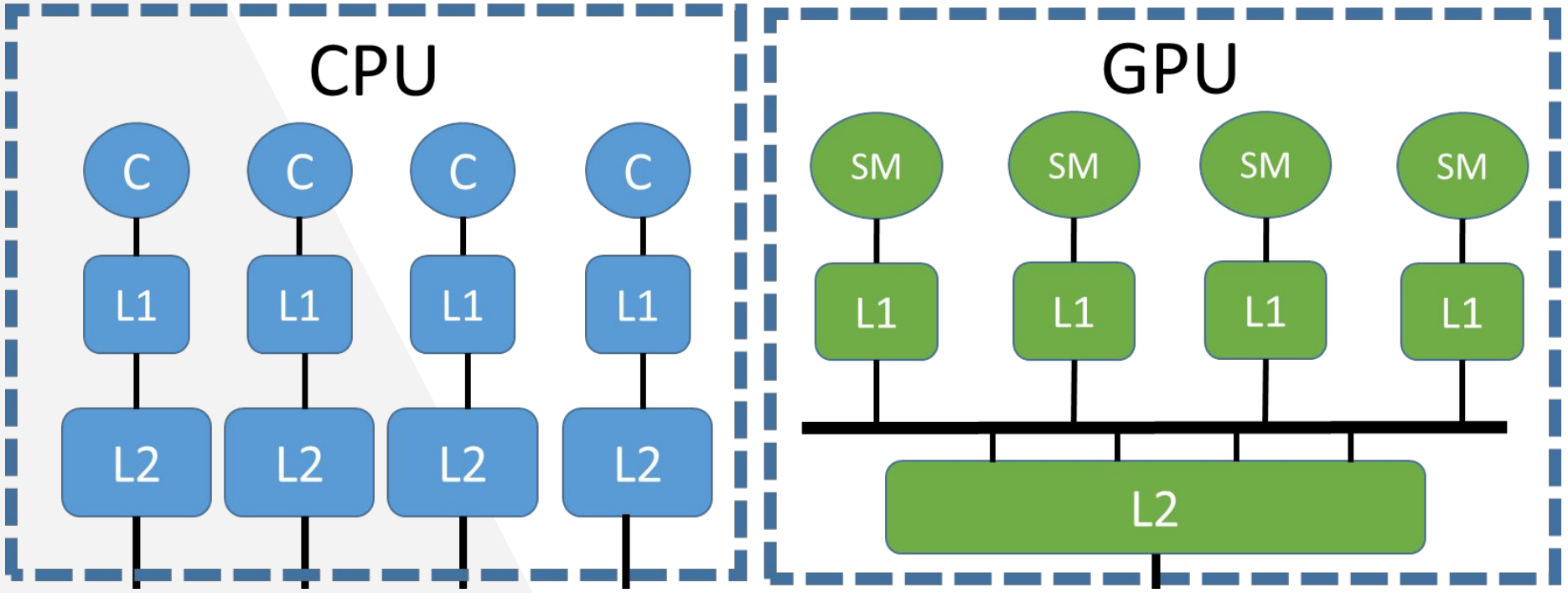


Jerarquía de memoria

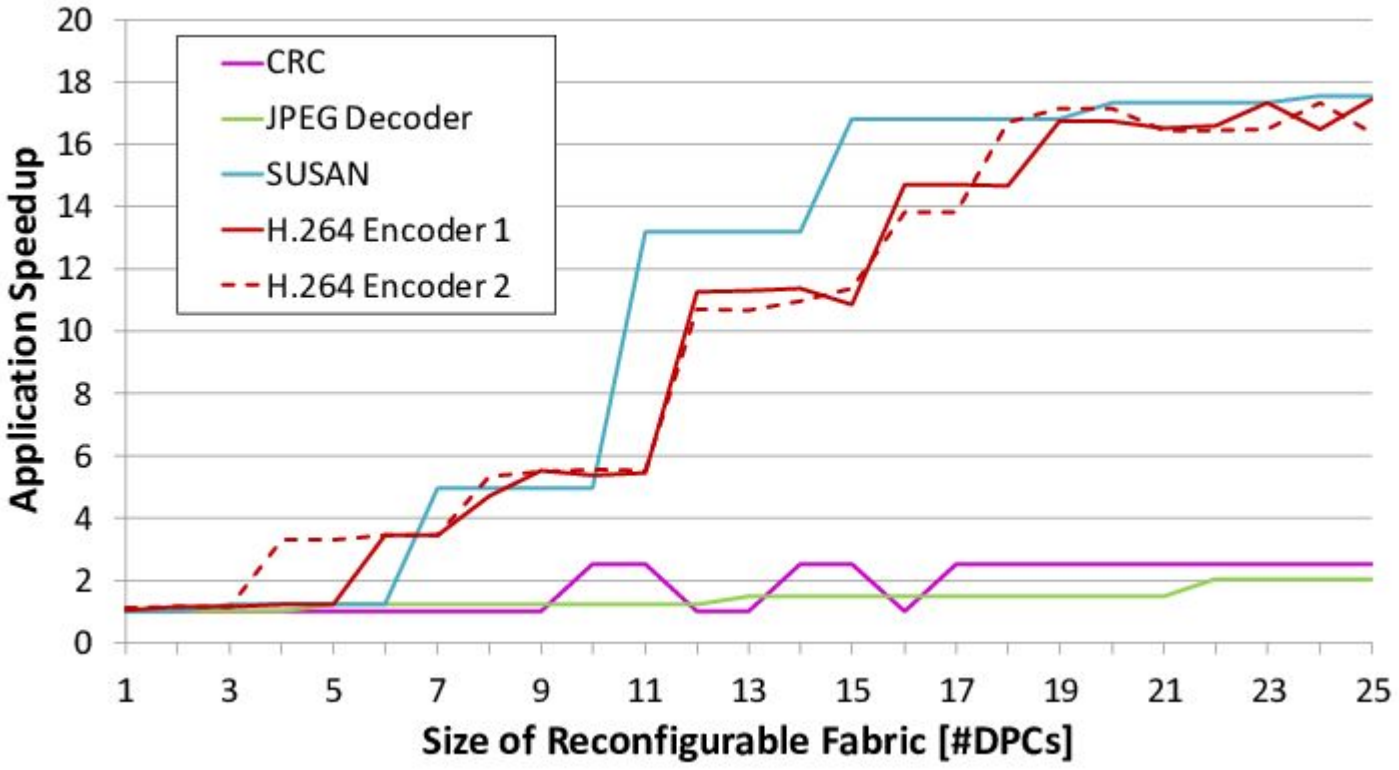
eDRAM Based Cache



Jerarquía de memoria en la GPU

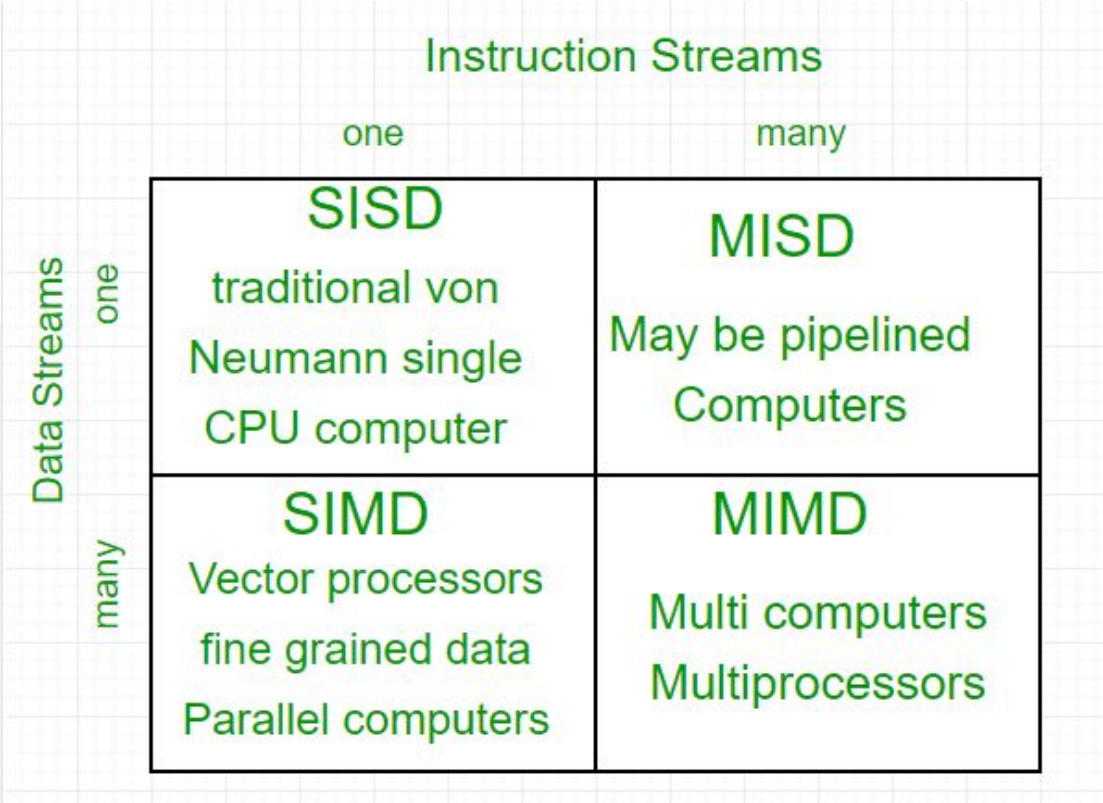


Efectos del cache en los resultados

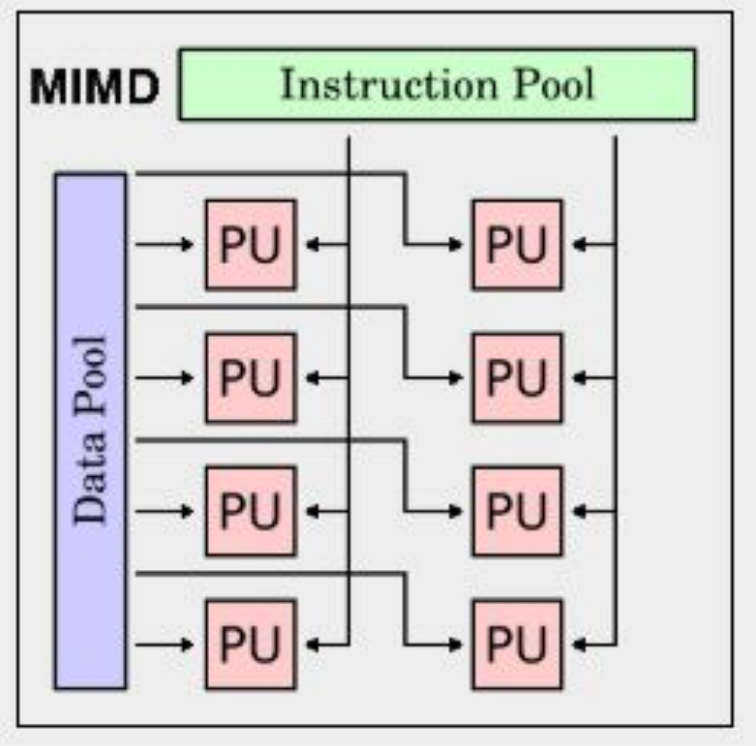


Data parallel vs Task parallel

Flynn's taxonomy of processors



Task Parallel



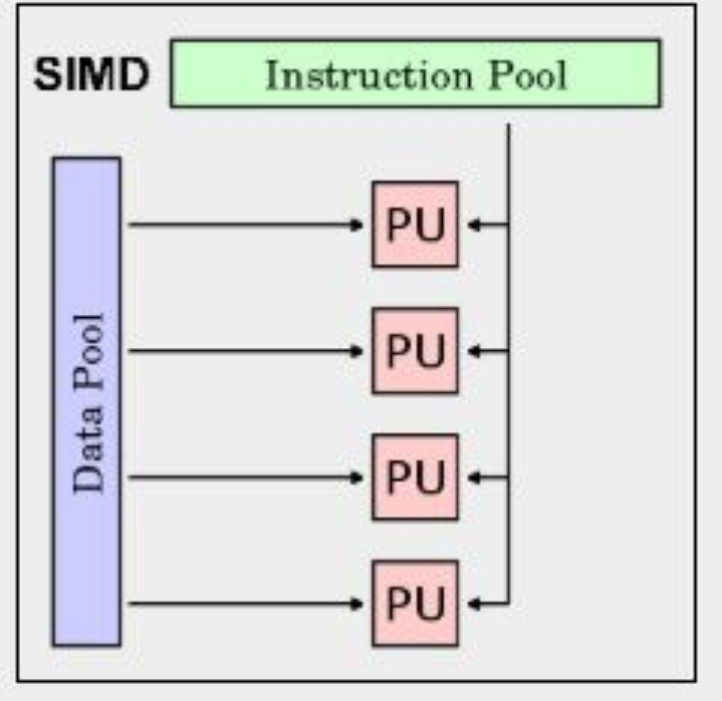
Fibonacci

step1)

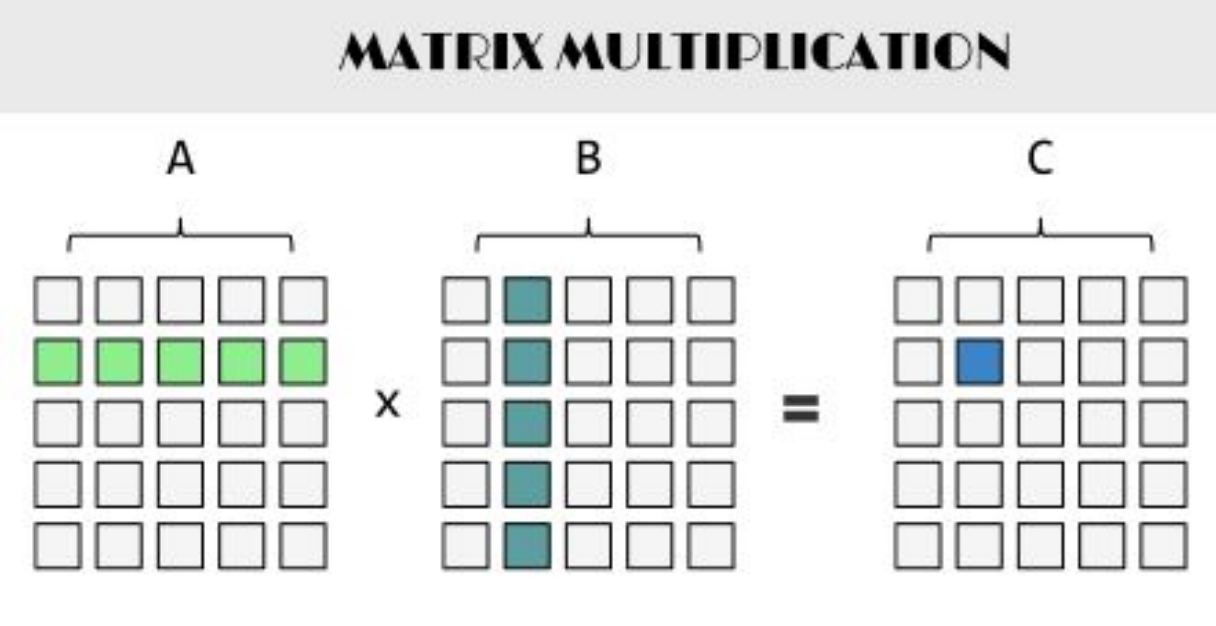
```
x = []
a = 1
b = 1
c = a + b
```

step2)

Data Parallel



Multiplicación de matrices



Programación en paralelo

Patrones programación paralela

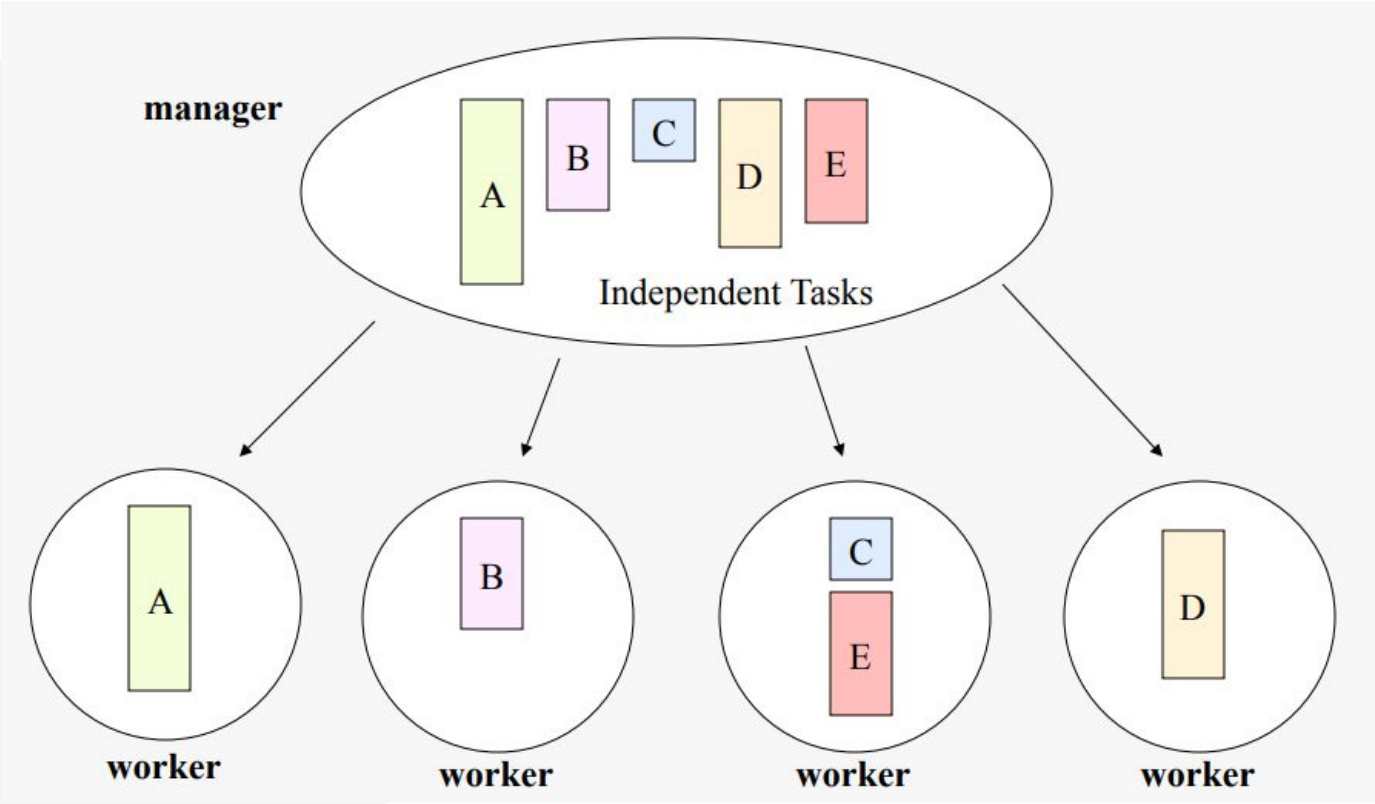
Embarrassingly Parallel

Replicable (reduce)

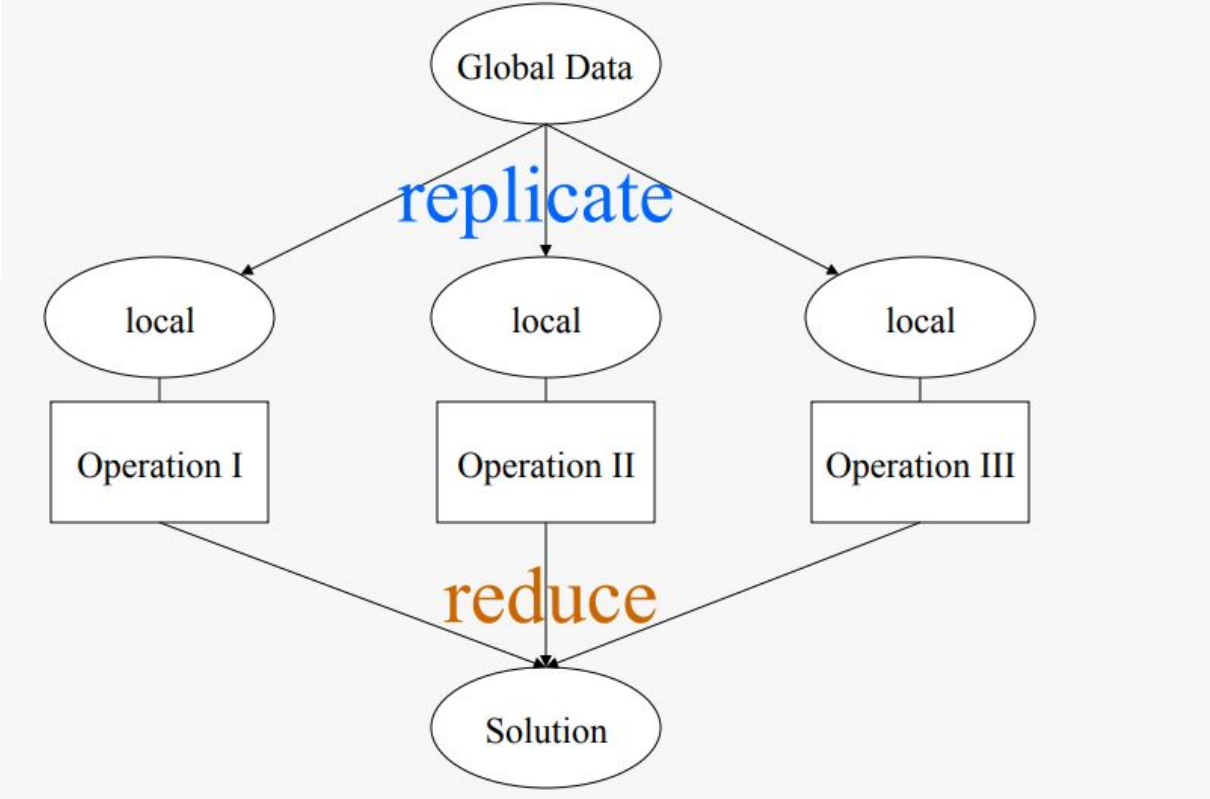
Divide & Conquer

Recursive Data

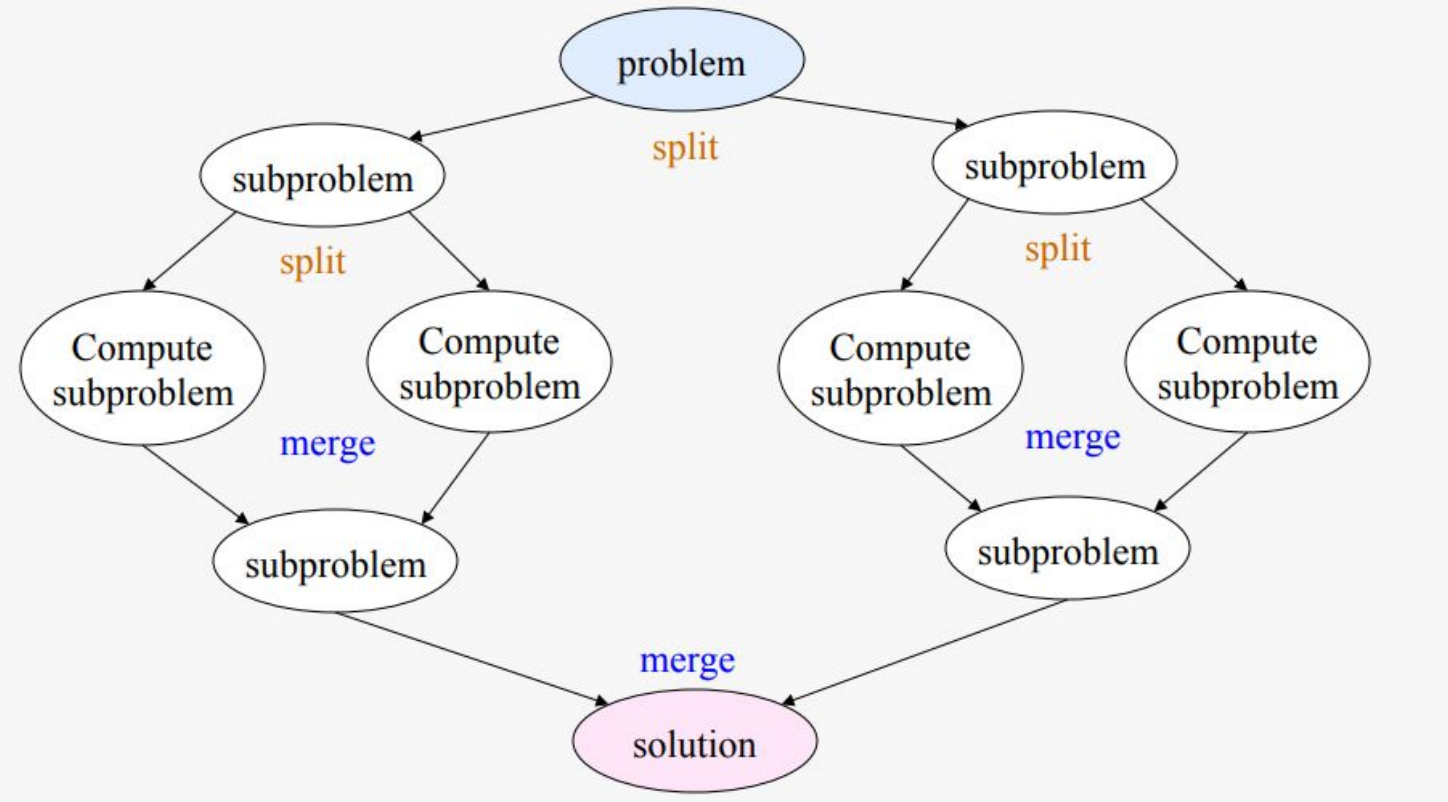
Embarrassingly Parallel



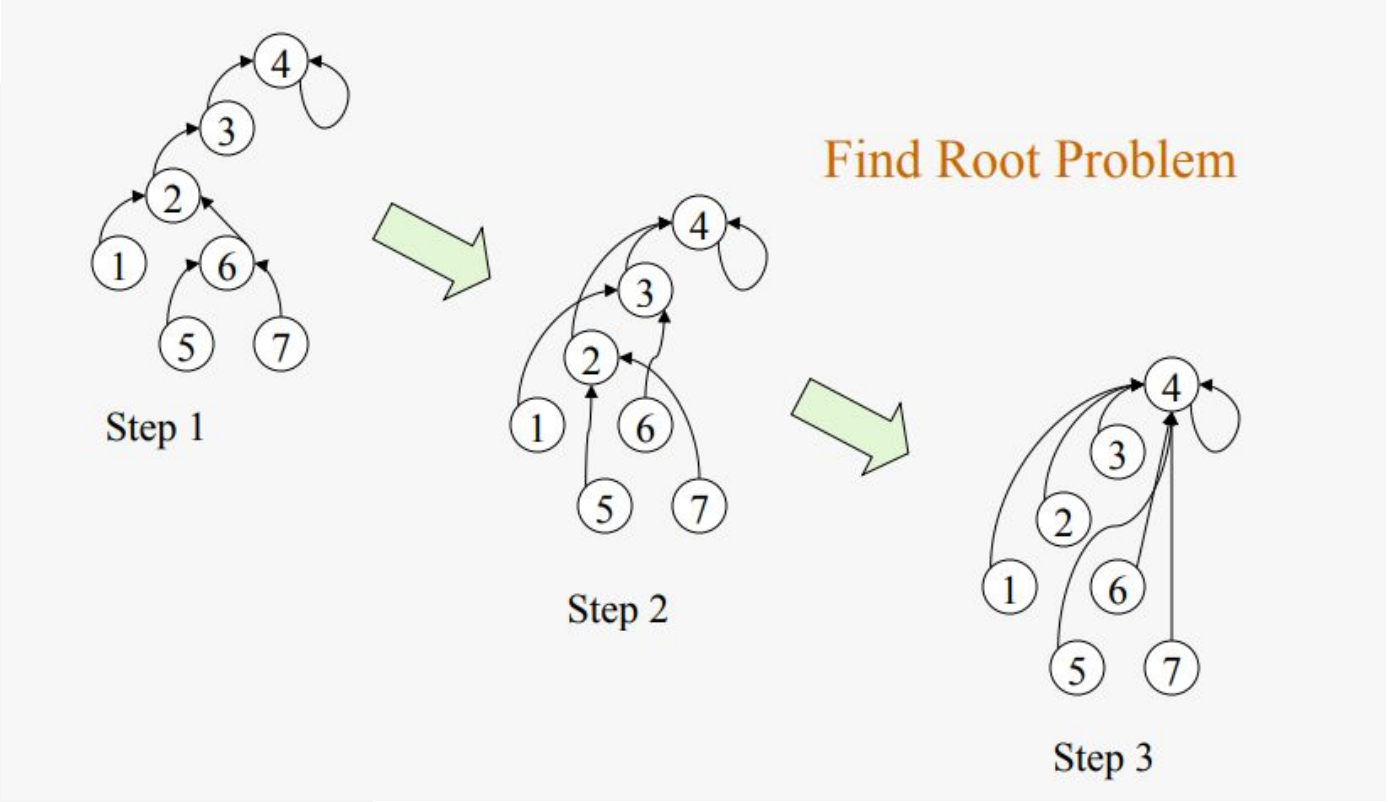
Replicable


Global Data
replicate
local
local
local
Operation I
Operation II
Operation III
reduce
Solution

Divide & Conquer



Recursive Data

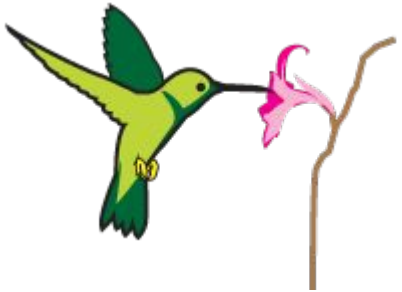


¿Con qué se programa en paralelo?

Programación Multicore

Multicore: Cilk Plus

Extensión de C y C++ para soportar programación paralela.



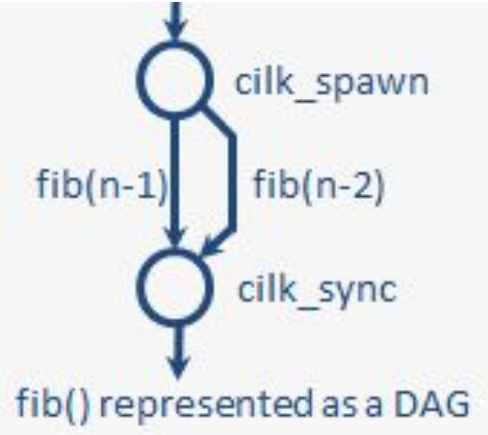
https://www.cilkplus.org/cilk-plus-tutorial

Keywords importantes

Parallel_for

Spawn

Sync

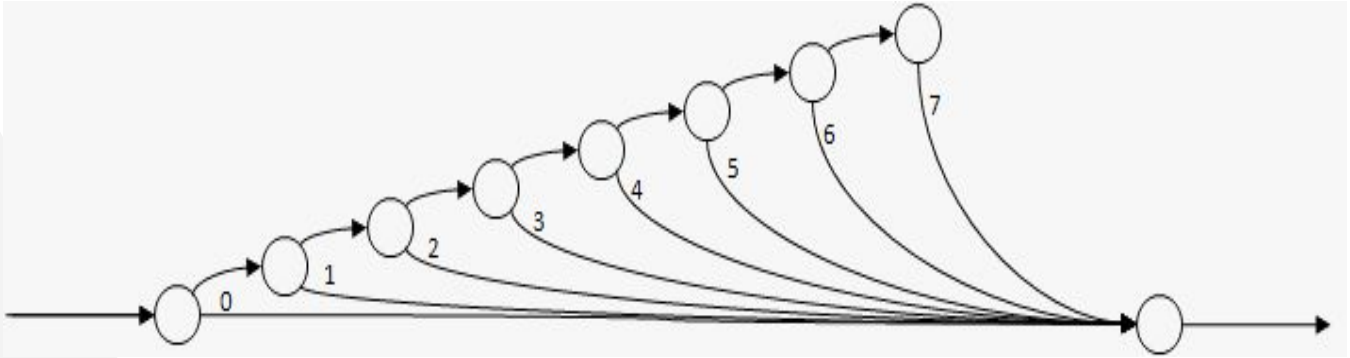


¿Cómo paralelizamos algo simple?

```
for (int i = 0; i < 8; ++i)
{
    do_work(i);
}
```

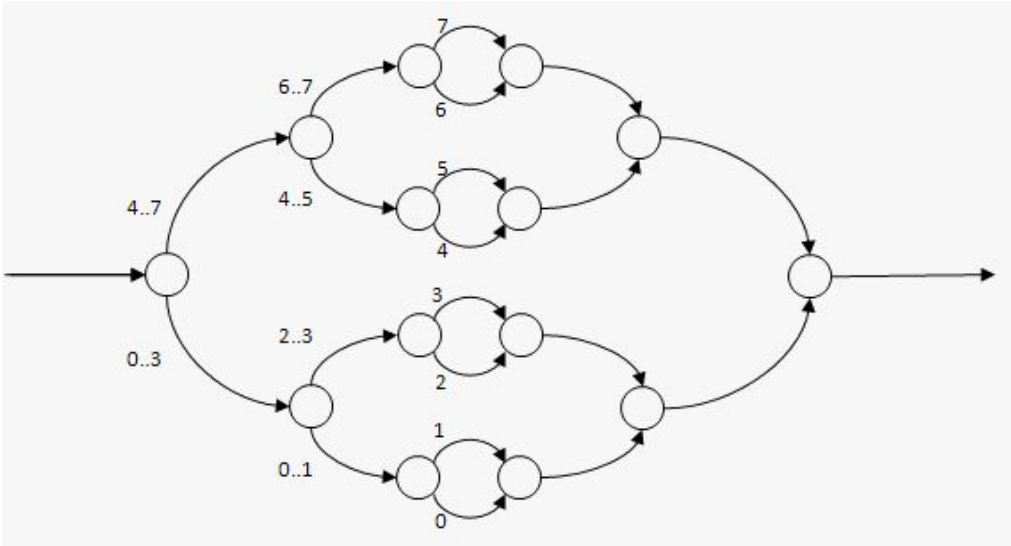
Formas de paralelizar

```
for (int i = 0; i < 8; ++i)
{
    cilk_spawn do_work(i);
}
cilk_sync;
```



Formas de paralelizar

```
cilk_for (int i = 0; i < 8; ++i)
{
    do_work(i);
}
```



Fibonacci

step1)

```
x = []
a = 1
b = 1
c = a + b
```

step2)

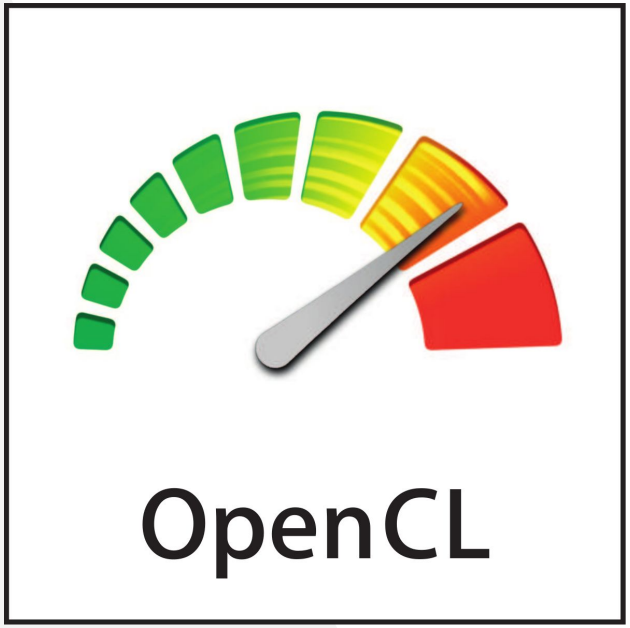
Fibonacci

```
int fib(int n)
{
    if (n < 2)
        return n;
    int x = fib(n-1);
    int y = fib(n-2);
    return x + y;
}
```

```
int fib(int n)
{
    if (n < 2)
        return n;
    int x = cilk_spawn fib(n-1);
    int y = fib(n-2);
    cilk_sync;
    return x + y;
}
```

Programación en GPU

GPU: OpenCL y CUDA



https://www.khronos.org/opencl/resources
https://devblogs.nvidia.com/even-easier-introduction-cuda/

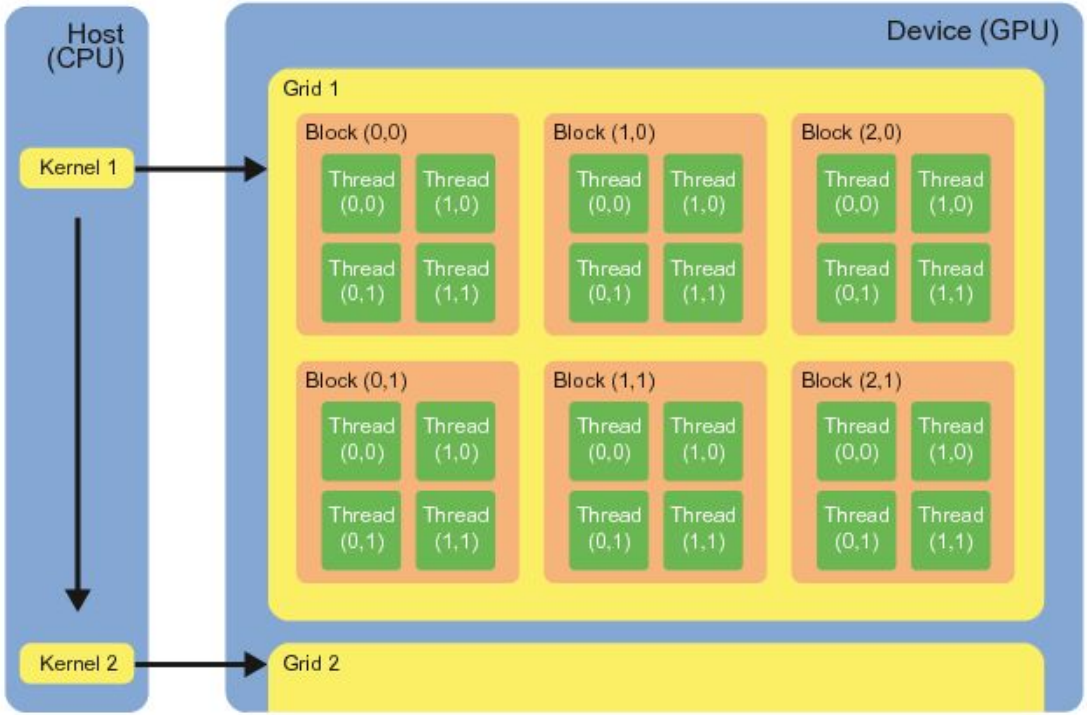
Keywords importantes

Kernel

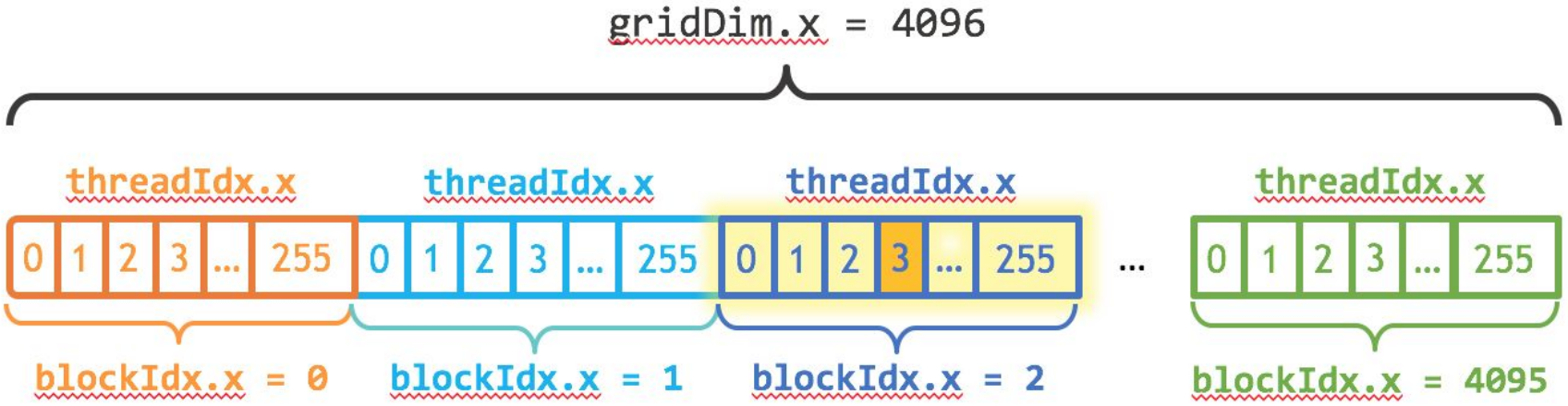
Thread

Block

Threads & Blocks



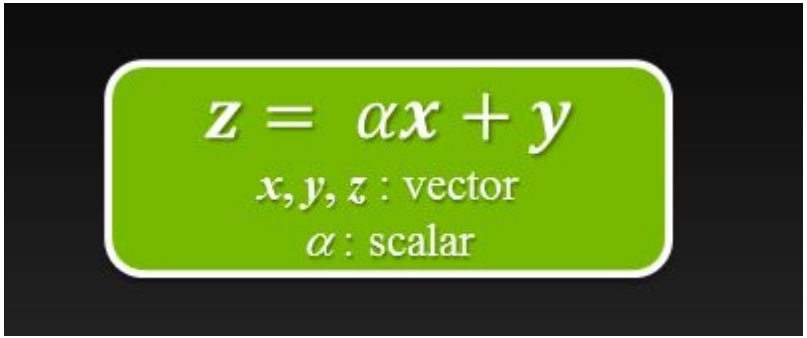
Threads & Blocks



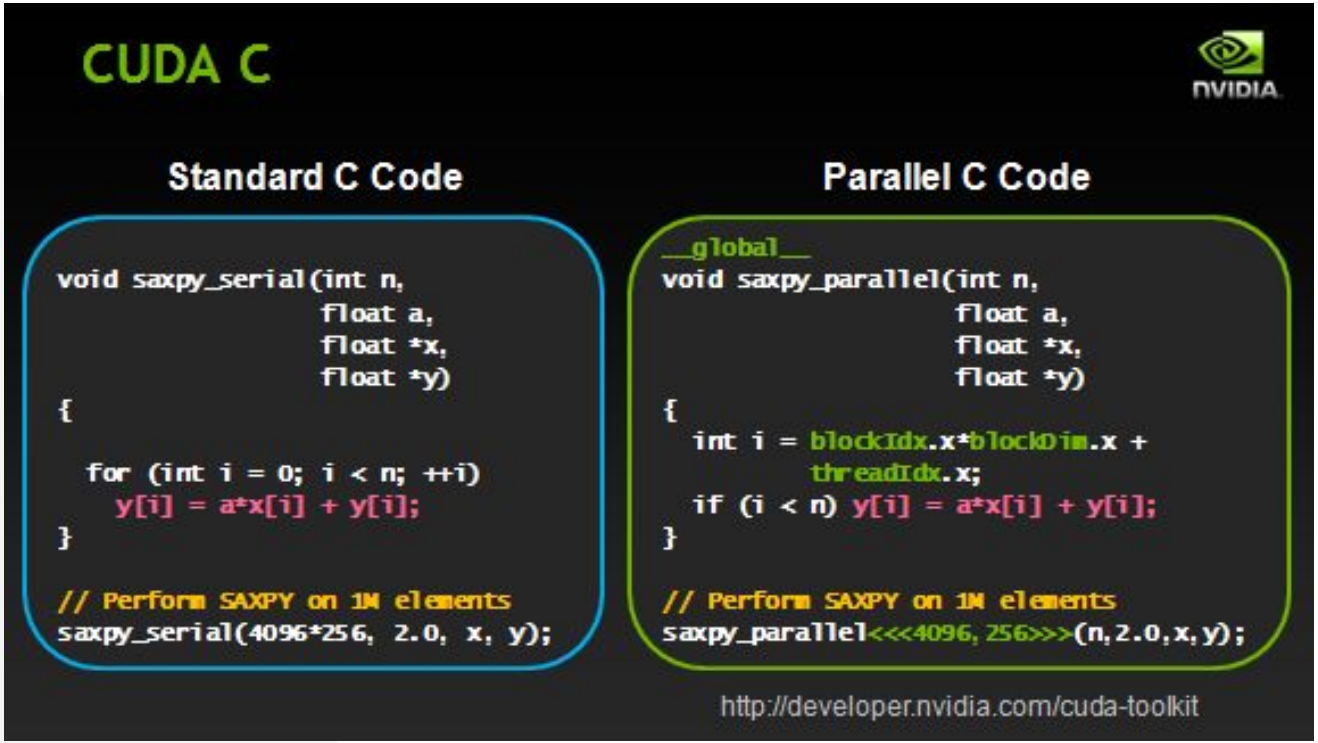

```
index = blockIdx.x  * blockDim.x + threadIdx.x

index =    (2)      *   (256)    +     (3)      = 515
```

Single Precision A□X Plus Y



Ejemplo CUDA



Ejemplo OpenCL

```
__kernel void SAXPY (__global float* x,
__global float* y, float a)
{
    const int i = get_global_id (0);

    y [i] = a * x [i] + y [i];
}
```

FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS
UNIVERSIDAD DE CHILE

Programación Paralela

Joaquín Torres Paris