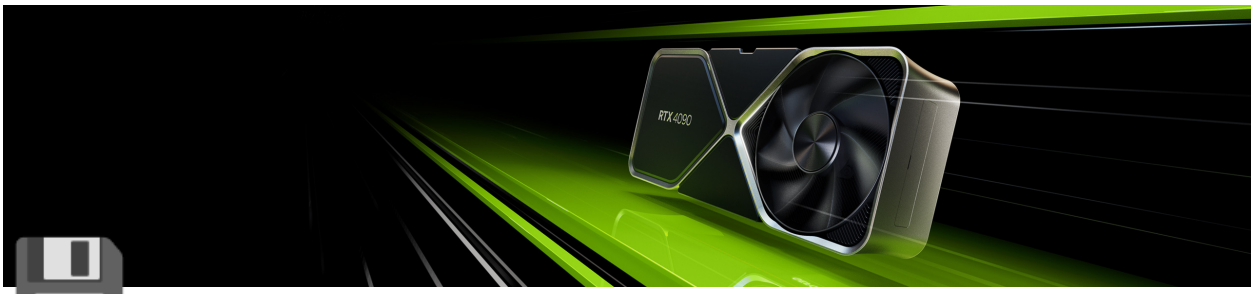


Asignatura PAP

CUDA

▼ Introducción

- Pasos de instalación de entorno CUDA
  - Primero mirar la tarjeta gráfica que tenemos

    panel de control- sistemas y seguridad- sistemas- administrador de dispositivos

    asi podemos ver que tarjeta grafica tenemos
  - buscamos los drives de nuestra tarjeta
    - pagina de nvidia y nos dice que driver admite
    - asi podemos saber hasta que paquete de cuda puedo instalar
  - vamos a pagina de cuda y vemos hasta que versión de cuda puedo instalar
  - Miramos en la documentacion de cuda que version de visual studio tengo que instalar

creo un pryecto en visual estudion con la instalacion de cuda y la version que esta instalada

ejecuto el codigo (suma de vectores)

    - haria una campura de la salida del programa y que se vea mi nombre arriba a la derecha de visual

- Resumen de un articulo
    - Indexado en el jcr
    - presentarlo en formato articulo
        - estilo ieee
    - indice de impacto del jcr, con eso sabemos cuantas veces ha sido referenciado ese articulo,(no del IF)
        - clarivate es la empresa que lleva el jcr
    - resumen de una pagina y media
    - Al menos una cita (de la forma que pone IEEE)
        - mínimo la del propio articulo

Teoría

Arquitectura de una tarjeta CUDA

string multiprocesador

cada uno de estos tienen 8 scalar procesadores (los verdes de la derecha)

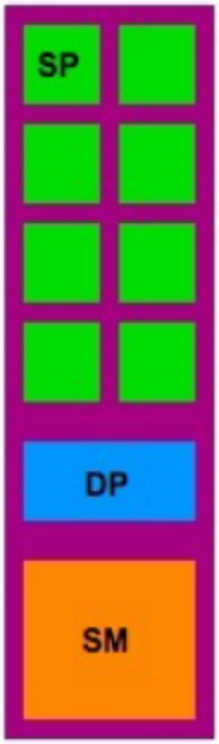
Cada scalar procesador es un núcleo cuda

Cada string procesador contine una memoria compartida para todos los scalar procesadores de ese string

y esta solo puede ver su memoria y la global, pero no la de otro string

Un hilo, un scalar procesador

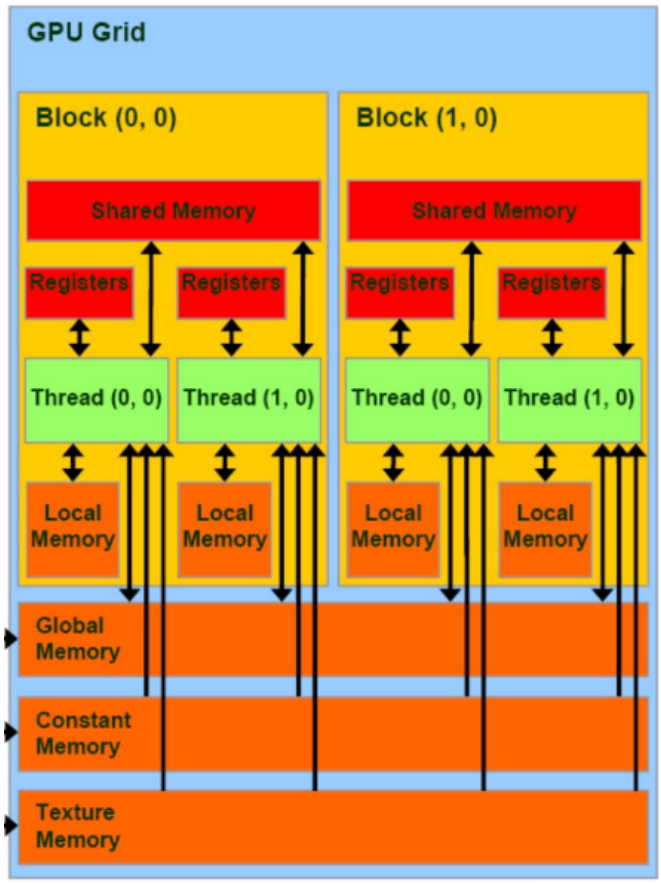
si lanzo 16 hilos sobre, las operaciones se realizarán en 2 ciclos de un string procesador(ya que solo lo he lanzado sobre este)



Caracteristicas

Nuestra unidad de medida va a ser un WARP, que son 32 hilos

los Scalar procesador pueden acceder a su memoria compartida de su String multiprocesador, pero no a los del resto



Lab

Para ejecutar, proceso:

pasar datos a la Gpu

ejecutar

devolver los datos a la cpu

cudaMalloc() asignacion de memoria en la GPU

CudaFree() liberacion de memoria en la GPU

```
T_WIDTH = 64;
Float* Md;
int size = T_WIDTH * T_WIDTH * sizeof(float);

cudaMalloc((void**)&Md, size);
cudaFree(Md);
```

usar en cpu M_hst y en la GPU usar M_dev

como pasar una variable de un lado a otro cudaMemcpy() [Destino, origen, size, orden]

```
cudaMemcpy(Md, M, size, cudaMemcpyHostToDevice);
cudaMemcpy(M, Md, size, cudaMemcpyDeviceToHost);
```

Require 4 parámetros

- Puntero al destino
- Puntero a la origen
- Nº bytes a copiar
- Tipo de la transferencia
  - Host a Host
  - Host a Device
  - Device a Host
  - Device a Device

Donde se ejecuta la funcion:

sobre todo vamos a usar las 2 primeras

	Se ejecuta en	Ejecutable desde
`__device__ float DeviceFunc()`	device	device
`__global__ void KernelFunc()`	device	host
`__host__ float HostFunc()`	host	host

- Si tengo 2 matrices de 4X4 y quiero sumarlas

  no es lo mismo poner el numero de hilos 16 (que se consideraria lineal)

  que (4,4) que ya veria que es una matriz

  <<1,16>> (un bloque y 16 hilos)

  <<1,(4,4)>> (un bloque y una matriz 4*4 de hilos)

```
Primera matriz   +   Segunda matriz   =   Matriz resultado
  1  1  1  1          1  2  3  1           2  3  4  2
  1  1  1  1          2  1  1  1           3  2  2  2
  1  1  1  1          1  1  1  1           2  2  2  2
  1  1  1  1          1  1  1  9           2  2  2  10
```


- Ejercicios propuesto

  Defino ah[0,0,0,0]

  lo paso, y lo cambio a bd[]

  y luego lo devuelvo bh y lo muestro

Prgunta segura (hoja)

La tarjeta gráfica

La tarjeta gráfica NVIDIA GeForce GTX 1080 vs NVIDIA Quadro Plex 2200 D2

	NVIDIA GeForce GTX 1080	NVIDIA Quadro Plex 2200 D2
La fuente de potencia recomendada	500 vatio	sin datos
Conectores de alimentación adicionales	8-pin	sin datos
Compatible con SLI	+	sin datos

Memoria

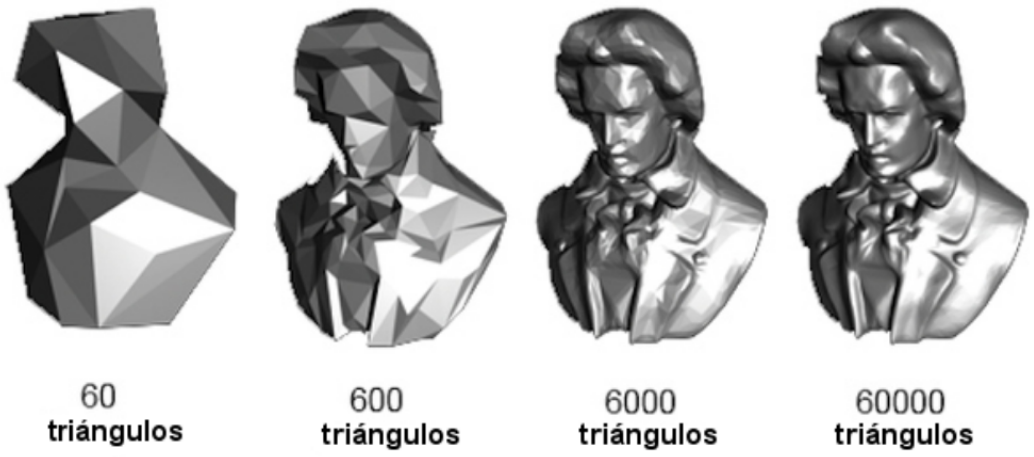
Parámetros de memoria instalada en GeForce GTX 1080 y Quadro Plex 2200 D2 - tipo, tamaño, bus, frecuencia y capacidad del canal. Para las tarjetas de video integradas en el procesador que no tienen su propia memoria, se utiliza una porción compartida de RAM.

Tipo de memoria	GDDR5	GDDR3
La capacidad máxima de RAM	8 GB	4 GB
El ancho del bus de memoria	256 Bit	512 Bit
La frecuencia de la memoria	10 GB/s	1600 MHz
El ancho de banda de memoria	320 GB/s	102.4 GB/s

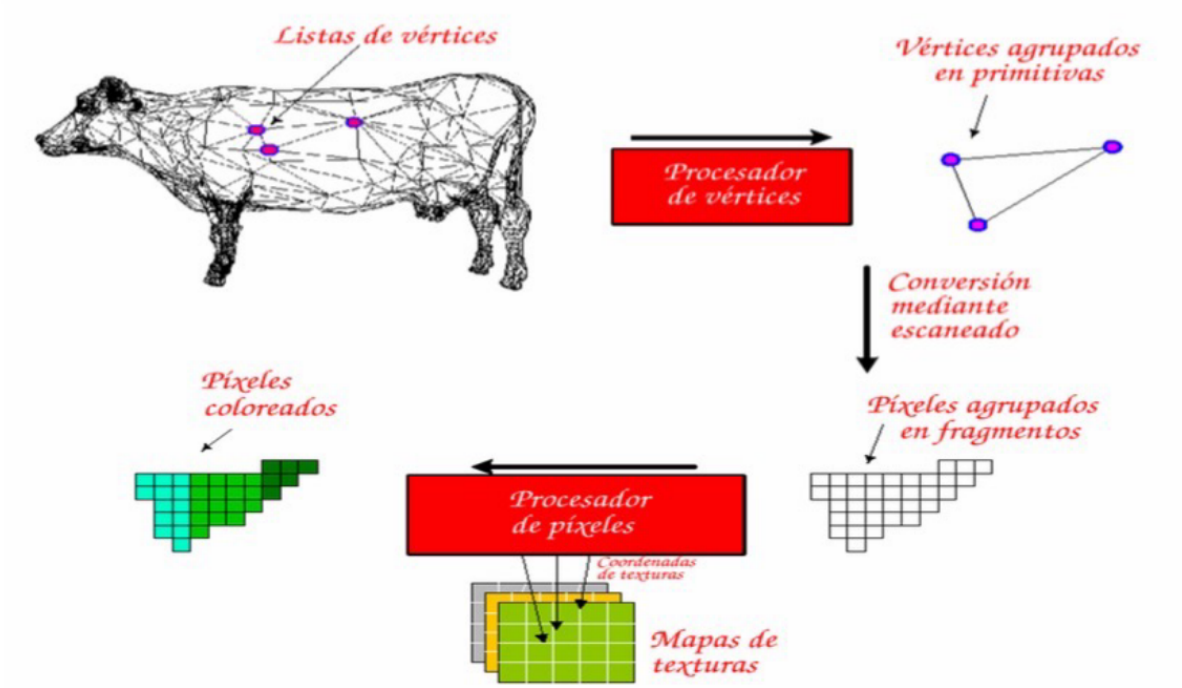
Caudal gráfico (entra)

la info se pasa a la gpu a traves de primitivas(triangulos)

cuanto más pequeños sean los triangulos, mejore se verá la imagen



Con los trapecios igual



Pregunta segura

A la gpu le llega una lista de primitivas (triángulos), que tiene la info asociada a peso, color, ...

Pasa al procesador de vértices y aplica la geometría (y puede hacer pequeños cambios)

cuando llega al procesador, hace el rasterizado (convertir una imagen de 3d a 2d) en 2 fases

- clipping(marca, triángulos que puede eliminar porque estan fuera de la cámara)
- culing(elimina , los triángulos que se habian marcado previamente por el clipping)

transforma los triángulos en tramas de pixeles

y llegan hasta el procesador de píxeles donde se hacen las cosas

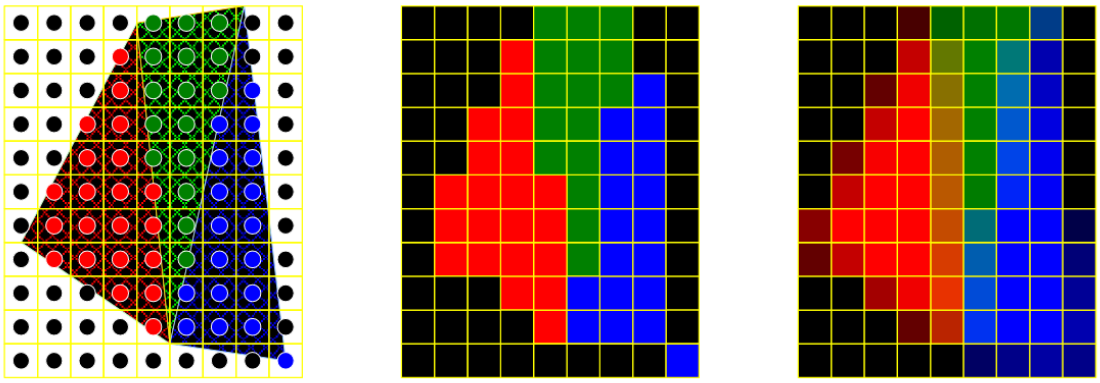
por ultimo se prepara la imagen para sacarlo por pantalla (blending compone la imagen)

TODO EL PROCESO el es proceso de renderizado (transformación de 3d a 2d)

¿que es un renderizado? proceso de transformación de una imagen de 3 dimensiones a 2

¿que es un rasterizado ? comversión de triángulos a fragmentos

- **Anti Aliasing : fdifumina los dientes de sierra de la imagen**



Geometría triangular | Aliased | Anti-Aliased

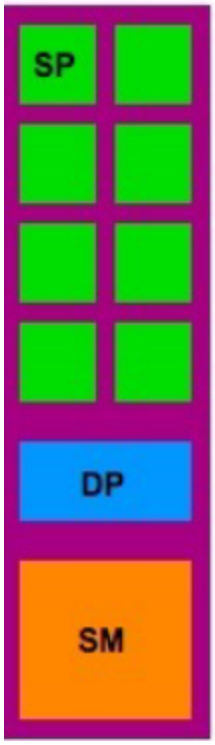
- La GPU utiliza (ENTRA)

Single **P**rogram **M**ultiple **D**ata Stream
Un único programa se ejecuta en múltiples hilos que acceden a diferentes flujos de datos

1 bloque de hilos =512 hilos

2 bloque de hilos o más (hasta 8) = 768 hilos

Si lanzo matriz de 1024 hilos, necesitaré 2 bloques SM (1 de 512 y otro de 512)

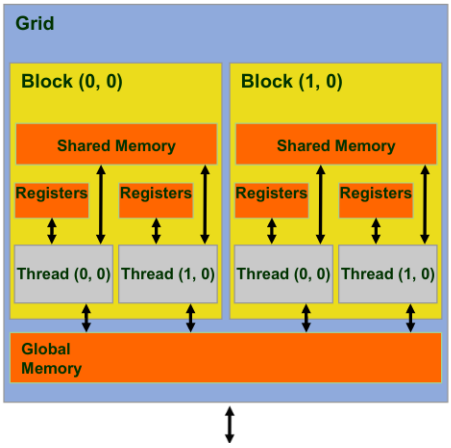


cuantas matrices de 256 hilos puedo ejecutar en un SM, 3 (758/256=3)

si tengo matriz de 1024, no entra en un SM entonces necesitaré 2 SM (512 y 512)

- De memoria

- Global:768 MB – 1024 MB
- Local: 16KB
- Compartida: 16 KB
- Registros: 8kb por multiprocesapor
- Constante 64kB (8KB por TPC)
- Texturas: 8KB por TPC



registros 8 kB

- Compilacion (entra)

  .cu lo pasamos al compilador

  separa lo que se va ejecutar en la cpu y en la gpu

  lo de la cpu pasa al compilador y devuelve un fichero foo.cpp

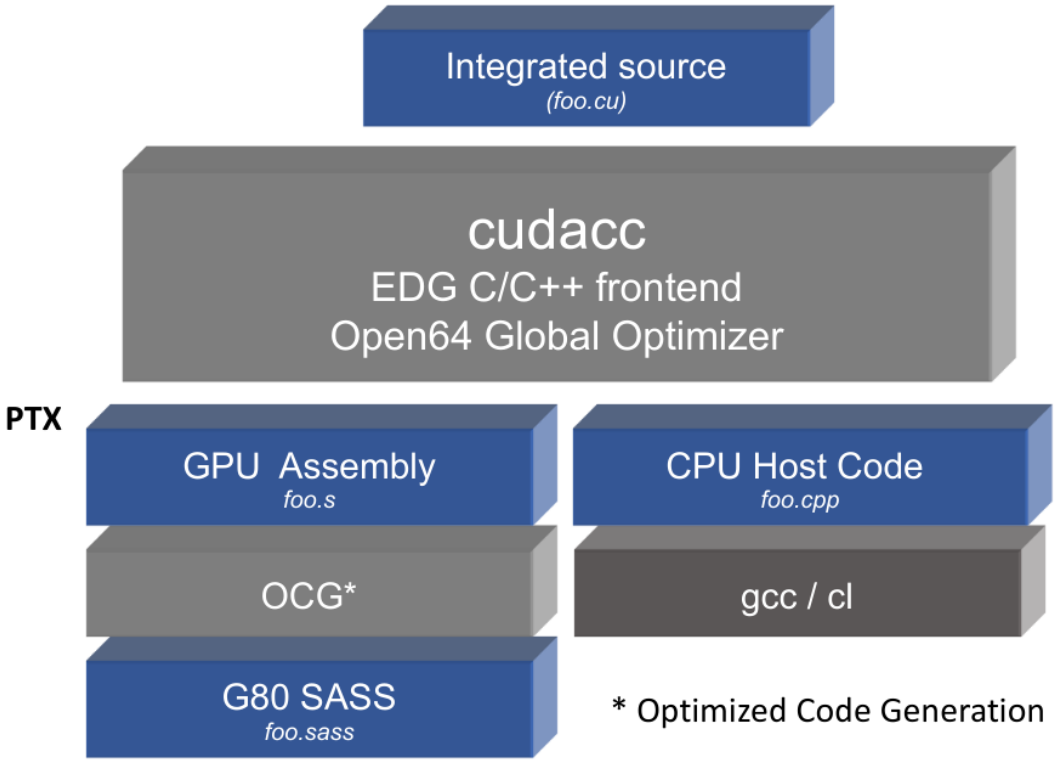
  segun el compilador, nos devuelve el fichero que se va a ejecutar en la cpu

  Tambien crea un fichero en ensamblador para la parte de GPU y crea fichero foo.s

  este indica que procesos son paralelos

  se le pasa al ocg que unirá ese fichero con las caracteristicas de nuestra tarjeta

  y devolverá un fichero foo.sass que es lo que se va a ejecutar sobre la gpu



string multiprocesador puede ejecutar hasta 8 bloques de hilos

como maximo puedo ejecutar 768 hilos

Cuantas matrices de 8x8 puede ejecutar sobre un SM(string multiprocesador)

como maximo puedo ejecutar 8 bloques, pues 8 [] 8*8 =64 ⇒ 768/64*8=8

si tengo matriz de 512x512 cuantos SM necesitaria

necesito 262000 hilos ⇒ ... hoja

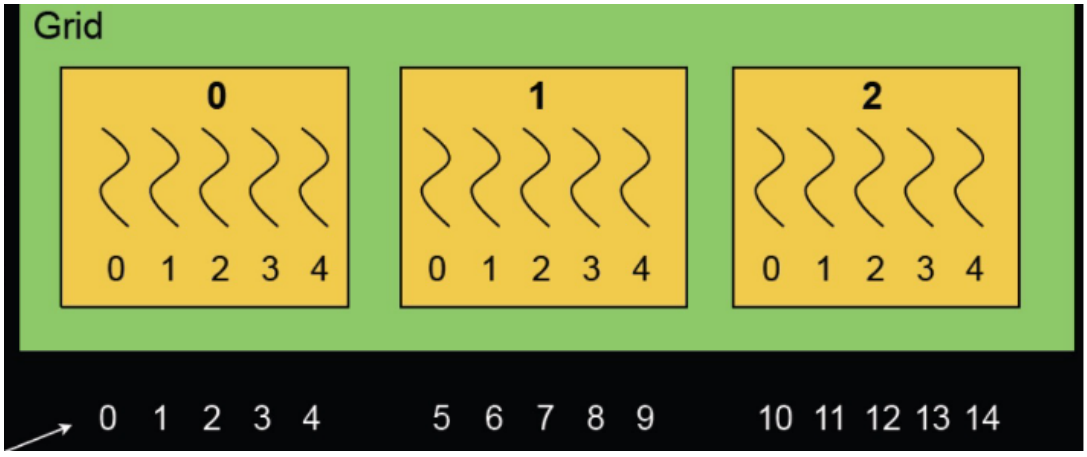
- Para acceder a un hilo en concreto de un bloque = numero de bloque* numero de hilo de bloque+ el thread

acceder al 5

numero de bloque 1

numero de hilos por bloque 5

hilo 0



para acceder al 10 ⇒ 2*5+0

```
int idx = (blockIdx.x * blockDim.x) + threadIdx.x;
```

bloques hasta en 2 dimensiones e hilos hasta 3

```
dim3   DimGrid(100, 50); // 5000 bloques de hilos
dim3   DimBlock(4, 8, 8); // 256 hilos por bloque
```

- la tercera dimensión de grid unicamente puede ser 1 (aunque no se suele poner)

  grid(10,2,1)

  blick(1,2,3) correcto

  grid(10,2,2)

  blick(1,2,3) MAL

LAB

- N hilos
- La segunda linea redondea hacia abajo

```
Int threadsPerBlock = 256;
Int blocksPerGrid =(N + threadsPerBlock - 1) / threadsPerBlock;
VecAdd<<<blocksPerGrid, threadsPerBlock>>>(d A, d B, d C, N);
```

- con esta formula, dándole x e y, me da donde se encuentra el elemento

elemento (i,j) = i_esimo (j+(i-1)*8)

- multiplicar 2 matrices que están divididas en bloques

```
int Row = blockIdx.y*TILE_WIDTH + threadIdx.y;
// Calculate the column idenx of Pd and N
int Col = blockIdx.x*TILE_WIDTH + threadIdx.x;
```

```
for (int k = 0; k < Width; ++k)
    Pvalue += Md[Row*Width+k] * Nd[k*Width+Col];

Pd[Row*Width+Col] = Pvalue;
}
```

Si tengo matriz de 4*4 , necesito 16 hilos ⇒ 4 bloques

width =4 tilewidtg=width/2= 4/2 = 2

```
// Setup the execution configuration

dim3 dimGrid(Width/TILE_WIDTH, Width/TILE_WIDTH);
dim3 dimBlock(TILE_WIDTH, TILE_WIDTH);
```

transpuesta

cambiar eje x por eje y en la formula

---

*p=p[ ]

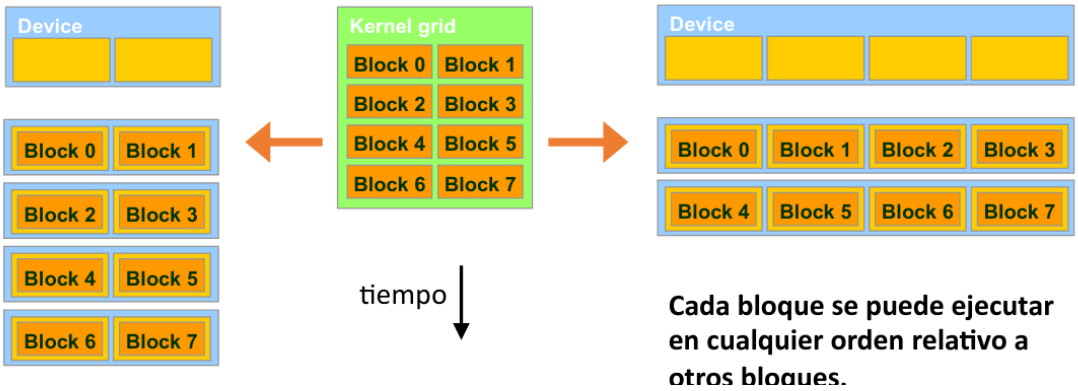
**p=p[ ][ ]

- convolucional
  - cambiar las posiciones del filtro

- la ponemos dentro de la matriz, la suma de todos es el resultado del centro de la matriz
- dará como resultado una matriz mas pequeña
- SOLO pide aplicar la convolucional y obtener el resultado

8 registros ⇒ 8192 cada uno de 4b entonces 32KB

- Yo puedo lanzar para que se ejecuten X bloques pero no se garantiza que en cada ejecucion se van a ejecutar en orden o en el mismo orden



lanzo la matriz y se dividen en warps y se ejecutan segun lo siguiente

- Los warps cuya siguiente instrucción tiene sus operandos listos para su uso son elegibles para su ejecución
- Los warps elegibles son seleccionados para ejecución de acuerdo a una política de **planificación priorizada**
- Todos los hilos de un mismo warp ejecutan la misma instrucción cuando son seleccionados

La memoria compartida es controlada por el programador, los registros y la memoria local la controla el compilador

código de los enlaces del nuncio ¿?¿?¿??¿

La API es una extensión de c

Tipos de **variable** cualificadoras en CUDA

Declaración de variable	Memoria	Ambiente	Vida
`__device__ __local__ int LocalVar;`	local	thread	hilo
`__device__ __shared__ int SharedVar;`	compartida	block	bloque
`__device__ int GlobalVar;`	global	cuadrícula	aplicación
`__device__ __constant__ int ConstantVar;`	constante	cuadrícula	aplicación

- **`__device__`** es opcional al emplear **`__local__`**, **`__shared__`**, o **`__constant__`**
- Variable automáticas, sin cualificar, residen en registros
  - Los arrays residen en la memoria local

tenemos funciones que se ejecutan en la cpu y otras en la gpu

- pow, sqrt, cbrt, hypot
- exp, exp2, expm1
- log, log2, log10, log1p
- sin, cos, tan, asin, acos, atan, atan2
- sinh, cosh, tanh, asinh, acosh, atanh
- ceil, floor, trunc, round

- __pow
- __log, __log2, __log10
- __exp
- __sin, __cos, __tan

Sincronización

puedo sincronizar los hilos de un bloque, pero no los de otro

__synchthreads();

Tiempos de acceso

en coma flotante

4 ciclos de reloj para suma, **producto** y MAD

suma y operaciones a nivel de bit 4 ciclos

logarítmicas 16 ciclos

Enteros, 16 ciclos

accesos a memoria 4 ciclos

accesos a memoria global, entre 400 y 600

sincronizaciones 4 ciclos

*mirar mejor las diapositivas 113

LAB

---

efectos del tamaño de la tesela, la tabla cae seguro

desenrollamiento unrolling

en vez de poner un blucle for, casco a capon las operacions

si tengo un for de 16, con este metodo tendira que poner a capon las 16 operaciones

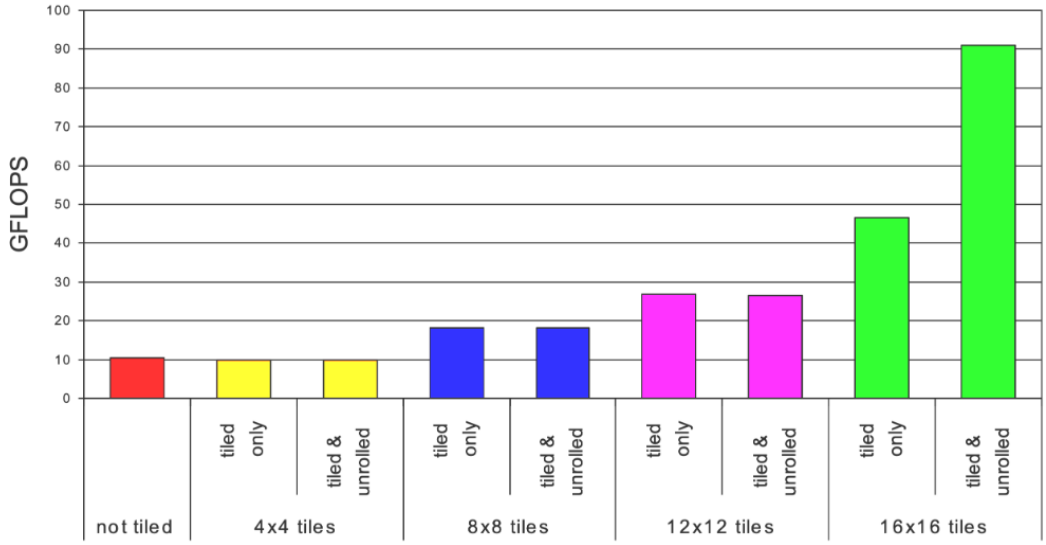
a[1]*b[1]

a[2]*b[2]

a[3]*b[3]

por ejemplo

Efectos del tamaño de la tesela



másximo rendimiento se consigue con teselas 16*16 y con des enrollamiento

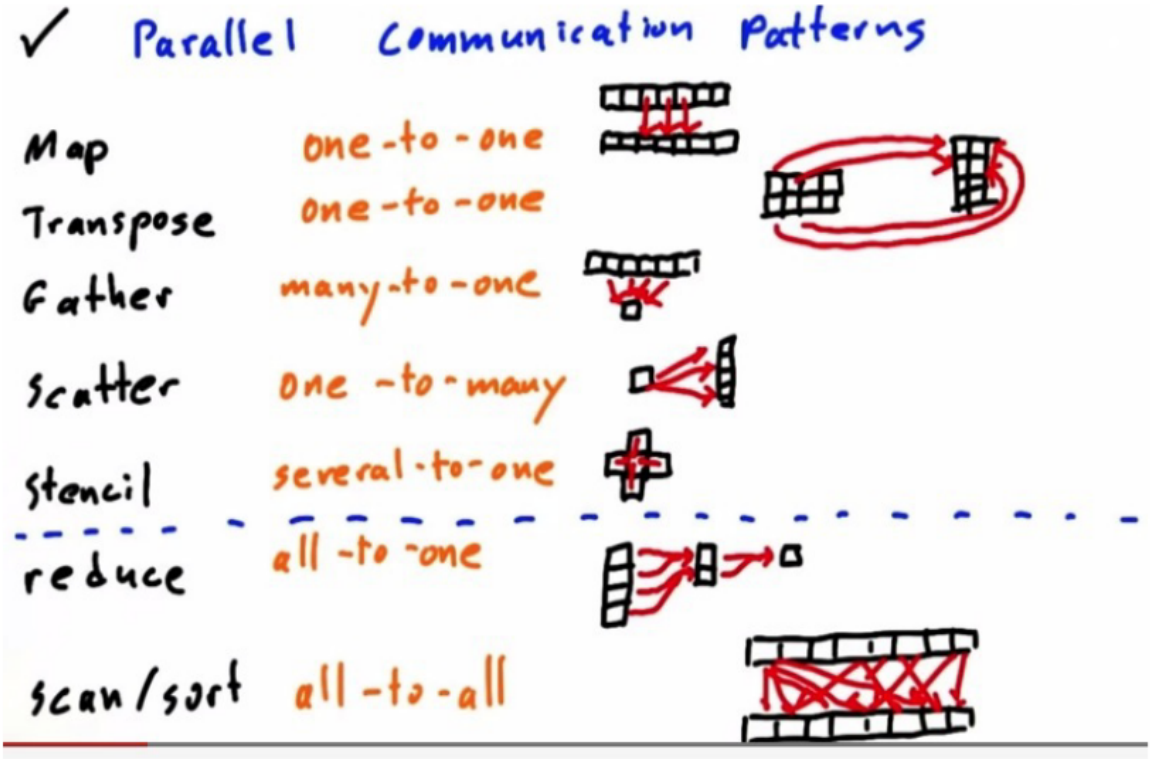
Vuelta a teoria

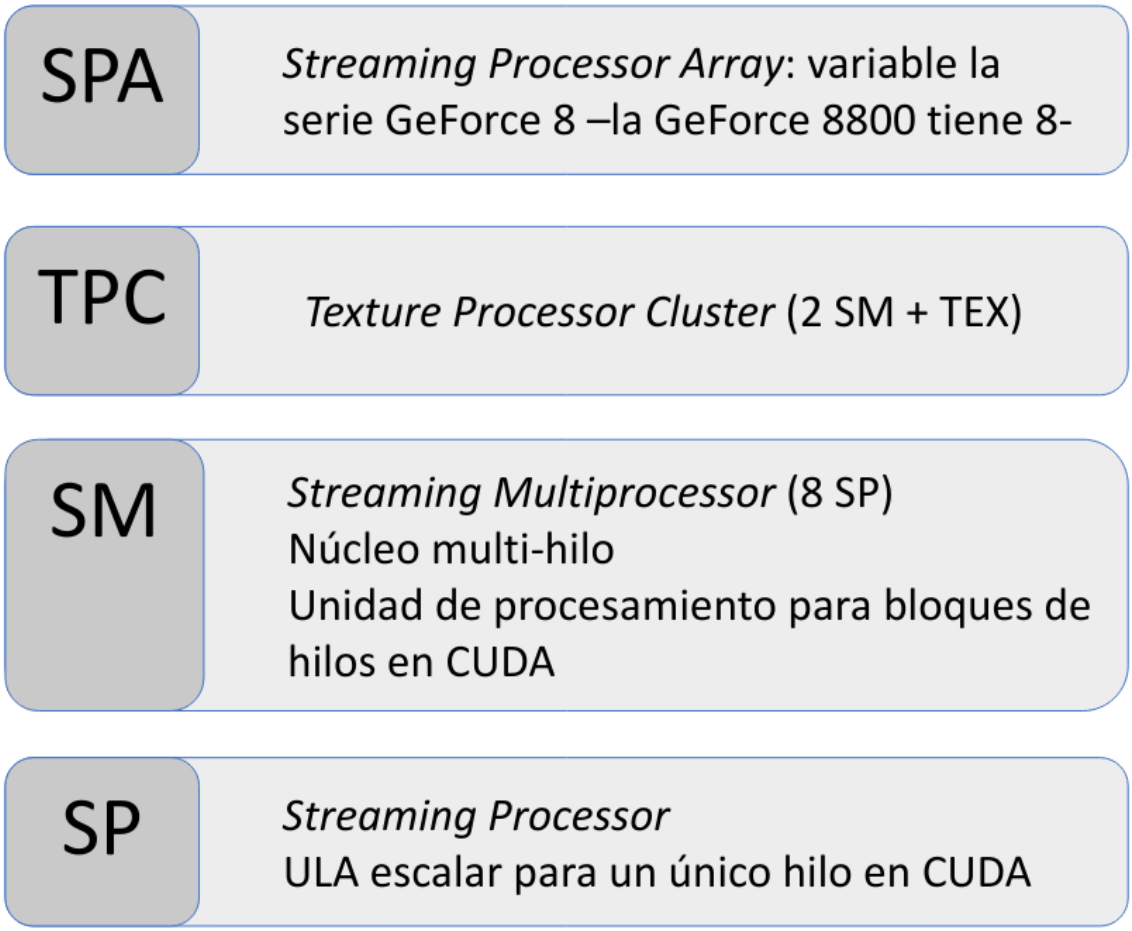
Ancho de banda

86.4GB/s de ancho de banda la G80

pregunta, me da x.x (familia y versión) y decir si una tarjeta puede hacer unas cosas o no, por ejemplo 7.1

patrones


✓ Parallel Communication Patterns
Map
one-to-one
Transpose
one-to-one
Gather
many-to-one
Scatter
one-to-many
Stencil
several-to-one
reduce
all-to-one
scan/sort
all-to-all



que es y que significa las cosas de arribas

LAB

Tratamiento de errores

cudaDeviceSynchronize(); //para la cpu hasta que termine de ejecutar la GPU

Eventos

aun no esta la diapositiva

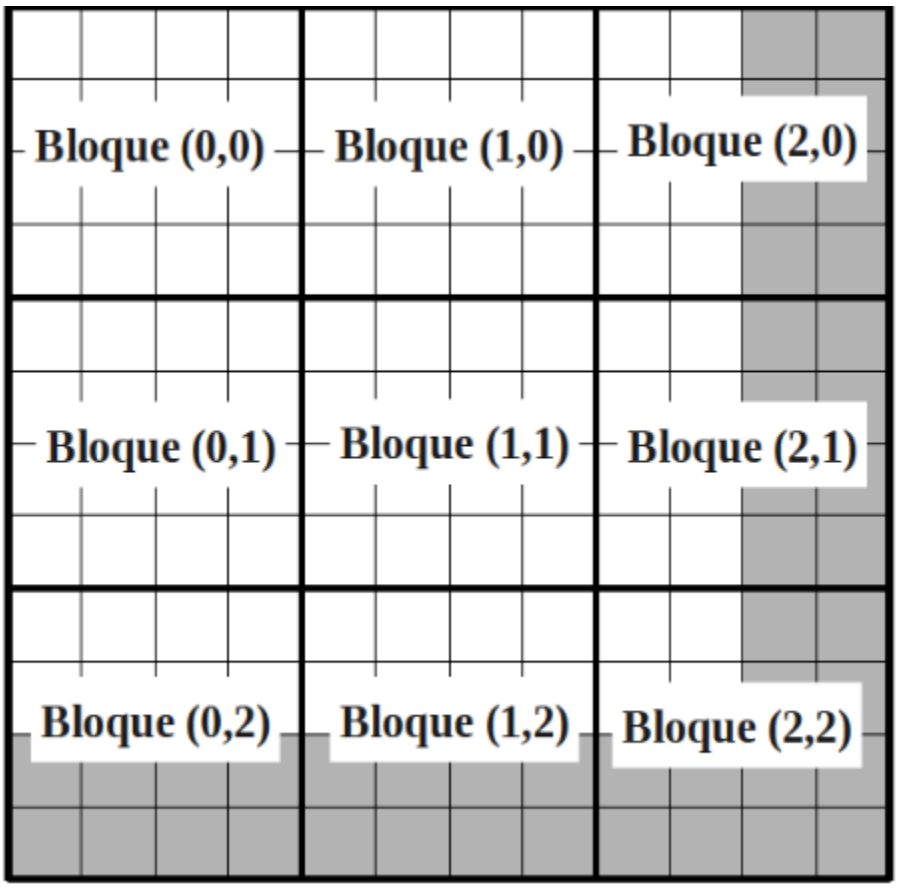
solo para saberlo, no lo vamos a utilizar

Limites en los bloques

matriz 10*10, lo que hago eseñado lineas y columnas adicionales que no las tendré en cuenta a la hora de operar

```
__global__ void sumaMatrices(int *x, int *y, int *z)
{
	int i = blockIdx.x * blockDim.x + threadIdx.x;
	int j = blockIdx.y * blockDim.y + threadIdx.y;

	int indice = i + j * N;
	if (i < N && j < M)
		z[indice] = x[indice] + y[indice];
}
```



si necesito más bloques que la propia matriz, tengo que buscar 2^i

si utilizo bloques adicionales, si o si necesito sincronizarlos

▼ Clase

ancho de banda G80 86,4

86,4/4 ciclos = 21,2 gb/s

Pregunta fija

Funcionamiento de un SM

- Se lanaza la cuadricula en la SPA

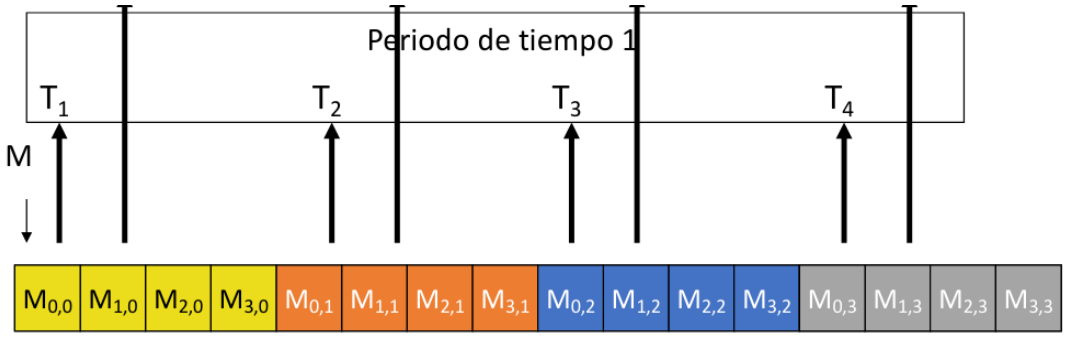
- Cuando lanzo el kernel, se convierte en bloques de hilos y se reparten secuencial mente por los SM´s
- Tiende a que sea 1 bloque de hilos por SM
- cuando ya tenemos el bloques de hilos en el sm, se convierte en warp
- cuando los warp llegan al dispachin se ejecutarán según estén preparados
- segun van terminando, dejan libre para que venga nuevos warps

tiempo de ejecucion de un warp, 4 ciclos

nada mas pillar el warp, se seleciona para que no lo coja otro

coalescencia

- obtengo la mayor velocidad si todo un warp lee los datos de forma consecutiva
- formas de resolver cuando no tengo coalescencia
  - con coalescencia ya que no lee de forma secuencial sino N[1], n[4],....



- transpuesta o memoria compartida

sin cuando lee secuencial mente

La memoria compartida utiliza 16 bancos de 32 bits

como se si tengo un conflicto con los bancos

si el paso entre hilos y el banco (S) es par, me da conflicto

si S es 1 o impar, no me dará conflicto

---

control de flujo

- con divergencia
  - si los saltos de los hilos está en función de un valor
  - puede producir un retardo

```
If (threadIdx.x > 2){…}
```

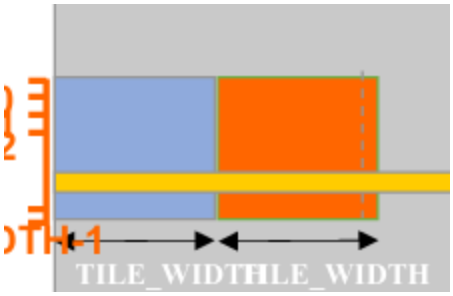
- sin divergencia

- no causará un retardo
- salto en función de warp

```
If (threadIdx.x / WARP_SIZE > 2){…}
```

Técnicas de optimización

- solapar operaciones
  - cargo en memoria la azul
  - sincronizo
  - empiezo a operar
  - cargo la naranja en los registros
  - termino de ejecutar la azul
  - entonces paso a la naranja



- Desenrrollamiento

---

gt 200

datos , en vez de 2 sm tienen 3 por tpc

Algunos de los cambios mas importantes son:

- Los TPCs agrupan 3 SMs
- Las GPUs tienen 10 TPCs, resultado en 30 SMs y 240 CUDA cores
- El número de hilos que cada SM puede ejecutar en forma concurrente se incrementa a 1024 (30720 hilos por GPU)

Fermi

- memoria compartida 16Kb o 48 Kb
- mejora la precisión
- permite el control de errores
- sfus pasa de 2 a 4
- cache 2 de 768 kb
- 1024 hilos por bloques
- 2048 hilos por Multi procesador
- 64kb de registros
- 2 warps
- podemos utilizar ejecuciones de kernel paralelos

kepler

- Mejora el rendimiento y reduce el consumo
- sustituye la frecuencia de shader por la de graficos, entonces le permite aumentar los cores porque reduce la frecuencia
- 2048 hilos por Multi procesador
- 1024 hilos por bloque

SCALA

▼ LAB

Funciones: reciben parámetros

métodos, no reciben parámetros

arrays mutables

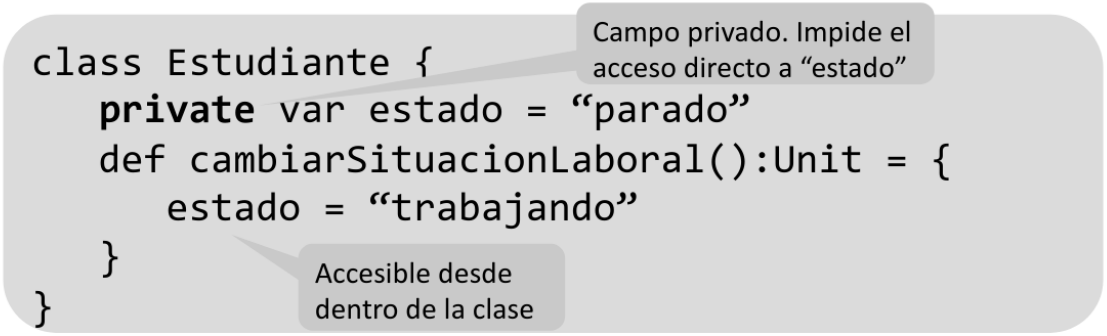
listas in mutables

En un match, si pongo “case _” por encima de otro case, entrará en este y no en el otro especifico

- Primera clase presencial

Efectos colaterales:

si llamo a una funcion y modifica algún valor fuera de la funicion



si llamo desde fuera de la clase a un atributo privado, me va a dar ERROR (tanto variables como métodos)

la solución seria crear un método que lo devolviese

Todos los enteros y los operadores son considerados objetos

para considerarlos como métodos es poner un . delante

(1).+(((2).*(3))./(x))

Los métodos se crean con variables tipo val

Objetos funcionales

son inmutablesw

parámetros siempre de tipo val

```
class Complejo(r:Double,
               i:Double) {
   val real = r
   val imaginaria = i
   override def toString() = {
      "[" + real + (if(i<0) "" else
       "+") + imaginaria + "i]"
   }
}
```

Crear
copia
clase
sean

al definir variables con los valores pasados por argumentos, puedo llamarlos desde fuera de la clase, en otro caso no podria

Los namespaces de Java son: campos, métodos, tipos y paquetes

Los namespaces de Scala son: valores (campos, métodos, paquetes y objetos singleton) y tipos (clases y *traits*)

La razón de que Scala sólo tenga 2, es precisamente para que los métodos sin parámetro se puedan sobreescribir con un val.

102

```
val otroAguila = new Ave

otroAguila.setNombre("Calva")
```

puedo ejecutarlo ya que se esta creando en un nuevo espacio de memoria

switch case

```
abstract class Term
case class Var(name: String) extends Term
case class Fun(arg: String, body: Term) extends Term
case class App(f: Term, v: Term) extends Term
}
Fun("x", Fun("y", App(Var("x"), Var("y"))))
```

- la construcción de instancias de clases Case, no requiere que se utilice la primitiva new.
- Solo tiene sentido definir una clase Case si el reconocimiento de patrones es usado para descomponer la estructura de los datos de la clase

```
object TermTest extends scala.App {
  def printTerm(term: Term) {
    term match {
      case Var(n) =>
        print(n)
      case Fun(x, b) =>
        print("^" + x + ".")
        printTerm(b)
      case App(f, v) =>
        print("(")
        printTerm(f)
        print(" ")
        printTerm(v)
        print(")")
    }
  }
```

```
def isIdentityFun(term: Term): Boolean =
term match {
    case Fun(x, Var(y)) if x == y => true
    case _ => false
  }
  val id = Fun("x", Var("x"))
  val t = Fun("x", Fun("y", App(Var("x"),
Var("y"))))
  printTerm(t)
  println
  println(isIdentityFun(id))
  println(isIdentityFun(t))
}
```

---

si pongo un método como tipo final, NO puede ser sobrescrito

Traits

extends para las clases

with para los trait

queue FIFO

El ***trait Doubling*** hace dos cosas muy interesantes:

- Declara como superclase IntQueue. Esta declaración hace que el trait sólo pueda ser mezclado en una clase que también extienda IntQueue.

- El trait tiene una llamada super a un **método declarado abstracto**. En una clase normal sería ilegal, pero es posible en un ***trait***; los modificadores **abstract override** (sólo admitidos en los métodos de los ***traits***) indican que el trait se debe mezclar en alguna clase que tenga la definición concreta del método.

```
scala> val cola = new BasicIntQueue with Incrementing with Filtering
```

se ejecuta de derecha a izquierda

:: definir una lista a partir de la cabeza y la cola

::: concatena listas

currying

ejecución por partes de una función

rest(5,_:double)

rest(3) ⇒ me devolverá 5-3=2

las sentencias de izquierda a derecha pero al pone ‘:’ cambia y se evalua de derecha a izquierda

mezcla de trait y clases, son llamados “missing”

def poner(pos:Int , col: Int, l:list[int]) : lint[int] {

```
    var pos=random
    val l=List{0,0,00,0,0}
    if (l.isemprty) Nil
    else if (pos==1) col::l.tail
    else l.head::poner(col, l.tail)
}
```

a la hora de recolocar los bloques tras borrar,

voy haciendo el cambio hasta tener 2 veces el mismo tablero

---

integracion

- logis app

iot

- azure maps

Jerarquia de clases

ANY define los siguientes métodos

`== y ! =`

- no se puede sobre escribir ya que está declarado como final

Clase AnyRef

- mirar la diapositiva que la describe

clase null y nothing

mirar la diapositiva que la describe

placeholder

que es una función anónima

clausuras

Recursividad de cola

Los conjuntos son de tipo inmutable por defecto

paso por valor o por nombre

las 6 ultimas diapo algo entra fijo

mirar por mi cuenta

CLOUD

▼ Teoría

spain global azure (5 al 7 de mayo)

- encamina
- ntt data
- prodware
- plain concepts

- cagemini
- insight

Definición

Cloud computing está basado en la computación en Internet, donde los recursos se comparten, el software y la información son proporcionados a computadoras y otros dispositivos bajo demanda.

tipos de nubes

Mirar diapositiva

muy importante

Problemas de Existentes.

raid 0

distribuir el disco

raid 1

espejo

- En local

On Premises

- Infrastructure

- Todo lo gestiona la nube

- Software (as a Service)

- Usu solo gestiona la aplicacion y los datos, el resto la empresa que da servicios

- Platform (as a Service)

azure stat es una nube privada que podemos descargarla y usarla en local

- Importante para el examen

  tipos de nubes y tipos de plataformas

  Lock-in es la dependencia que tenemos con el servidor de servicios