MODELO DE MEMORIA

DISPOSITIVOS E INFRAESTRUCTURAS PARA SISTEMAS MULTIMEDIA

GRADO EN INGENIERÍA MULTIMEDIA 2018-2019

Albert García-García < agarcia@dtic.ua.es >

Jose García-Rodríguez < jgarcia@dtic.ua.es>


photo by Michael Bulbenko

STEVE WOZNIAK

CO-FUNDADOR DE APPLE (EL QUE SABÍA)

https://apple2history.org/history/ah03/

JOHN CARMACK

CO-CREADOR DE DOOM (PROGRAMADOR)

https://en.wikipedia.org/wiki/Fast_inverse_square_root

```
float Q_rsqrt( float number )
{
        long i;
        float x2, y;
        const float threehalfs = 1.5F;

        x2 = number * 0.5F;
        y  = number;
        i  = * ( long * ) &y;                       // evil floating point bit level hacking
        i  = 0x5f3759df - ( i >> 1 );               // what the fuck?
        y  = * ( float * ) &i;
        y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
//      y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be
removed

        return y;
}
```

MICHAEL ABRASH

TFM (PROGRAMADOR)

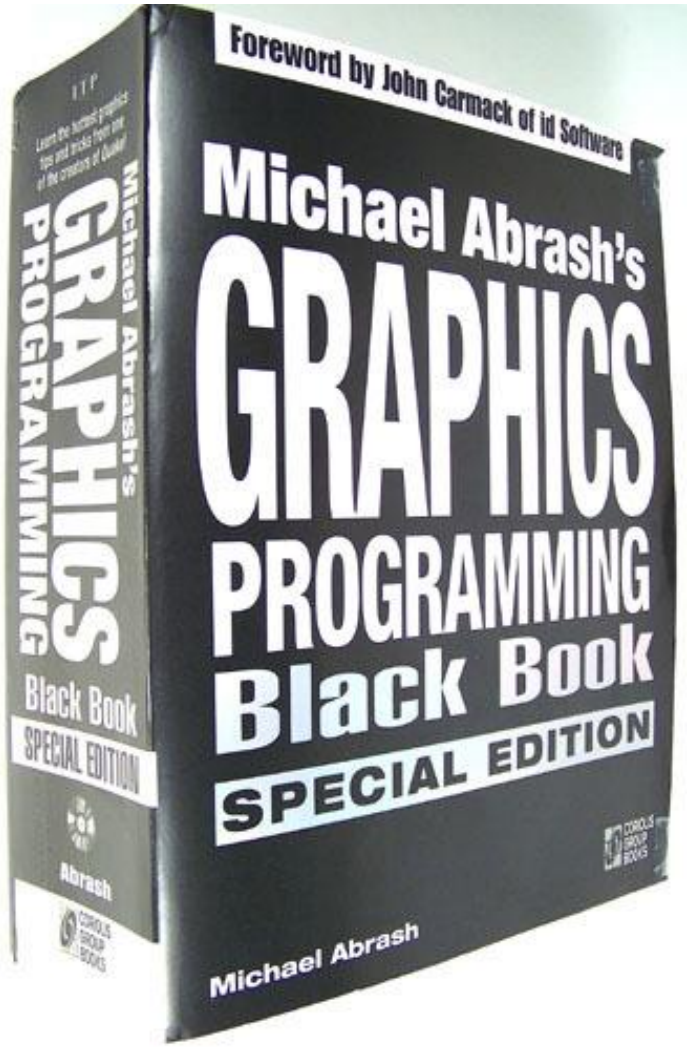
The most useful response I have is drawn from my own life: Do what you love. There are no guarantees, especially in the short run, about where that will lead – but at least you'll enjoy the trip, and it is likely to lead to exciting things.

DO WHAT YOU LOVE
http://blogs.valvesoftware.com/abrash/do-what-you-love/

In general, try things that seem worthwhile, set goals and work hard to achieve them, and see where that leads and how you respond. It'll be clear when something becomes compelling, because it'll be where you choose to spend your time and attention. It may not be what you expected or wanted it to be – but by definition you'll find it fascinating and satisfying. And when you think about what you could do with your studies/career/life, really, what more could you want?

MICHAEL ABRASH

TFM (PROGRAMADOR)




50
AMMO
100%
HEALTH
2 3 4
5 6 7
ARMS
0%
ARMOR
BULL 50 / 200
SHEL 0 / 50
RCKT 0 / 50
CELL 0 / 300


GAME BRAIN
BLACK BOOK
DOOM
FABIEN SANGLARD
i486
i386
DOOM
WOLFENSTEIN 3D 2


STOP
MAKING
STUPID
PEOPLE
FAMOUS

CONTENIDO

JERARQUÍA DE MEMORIA EN LA CPU

JERARQUÍA DE MEMORIA EN LA GPU

CONSIDERACIONES DE RENDIMIENTO

MOTIVACIÓN DE LAS MEMORIAS JERÁRQUICAS

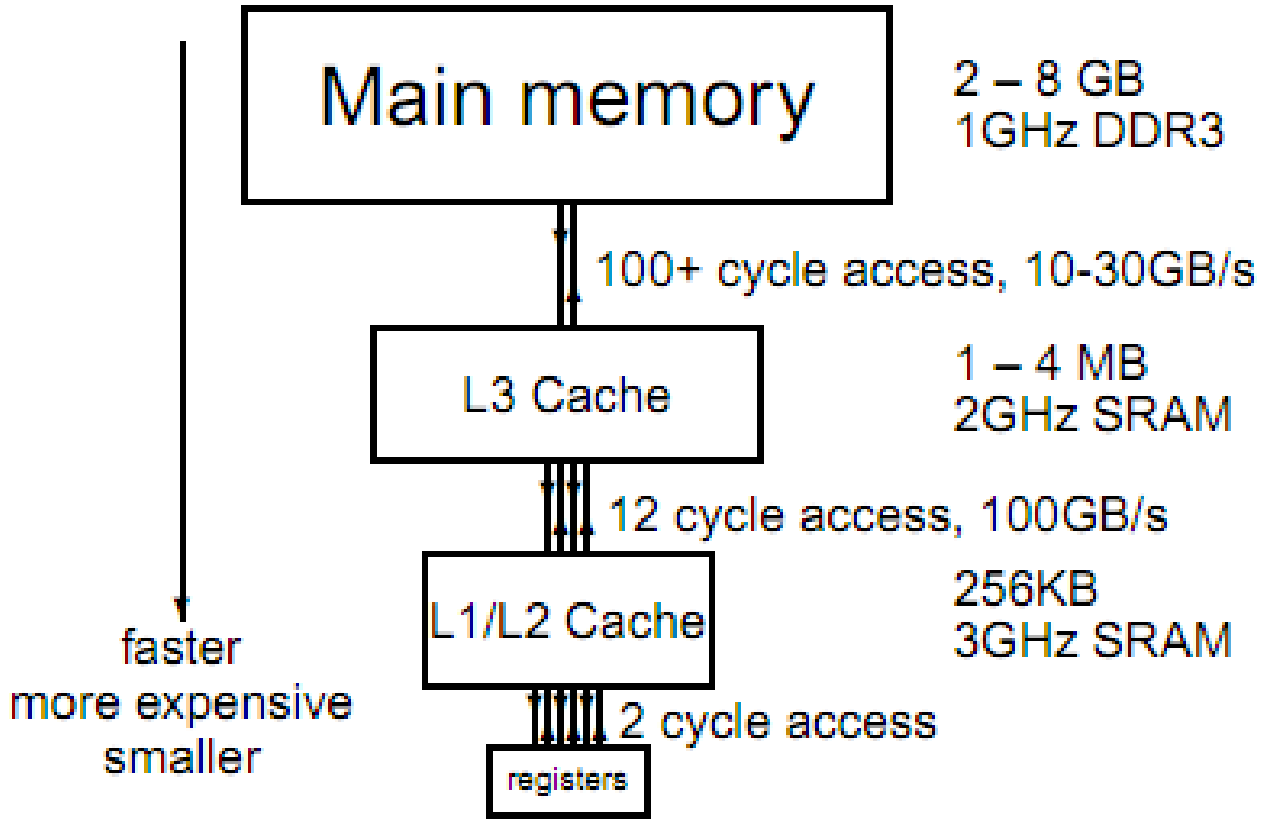
CONCEPTO DE LOCALIDAD

LOCALIDAD TEMPORAL

LOCALIDAD ESPACIAL

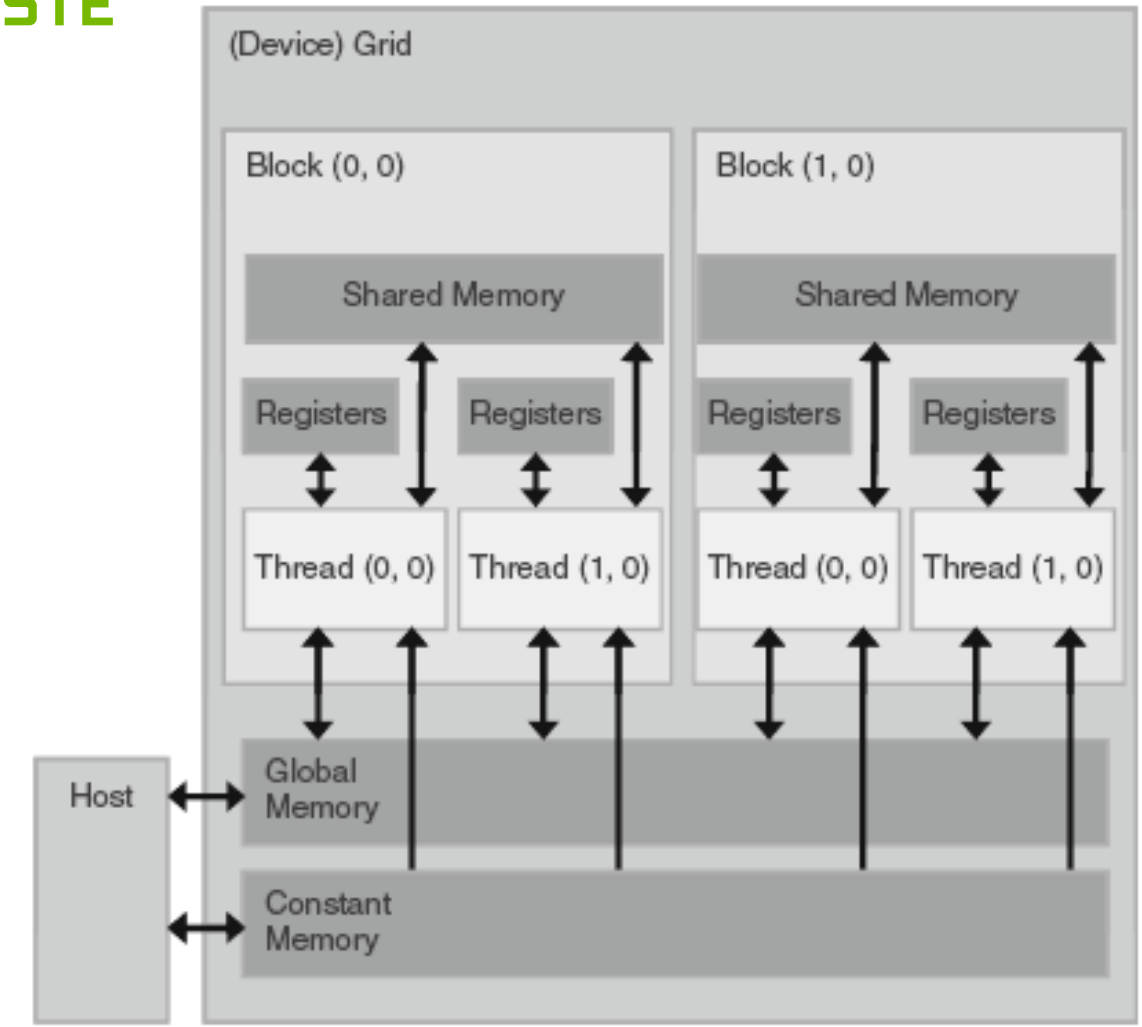
JERARQUÍA DE MEMORIA EN LA CPU

VELOCIDAD CONTRA COSTE



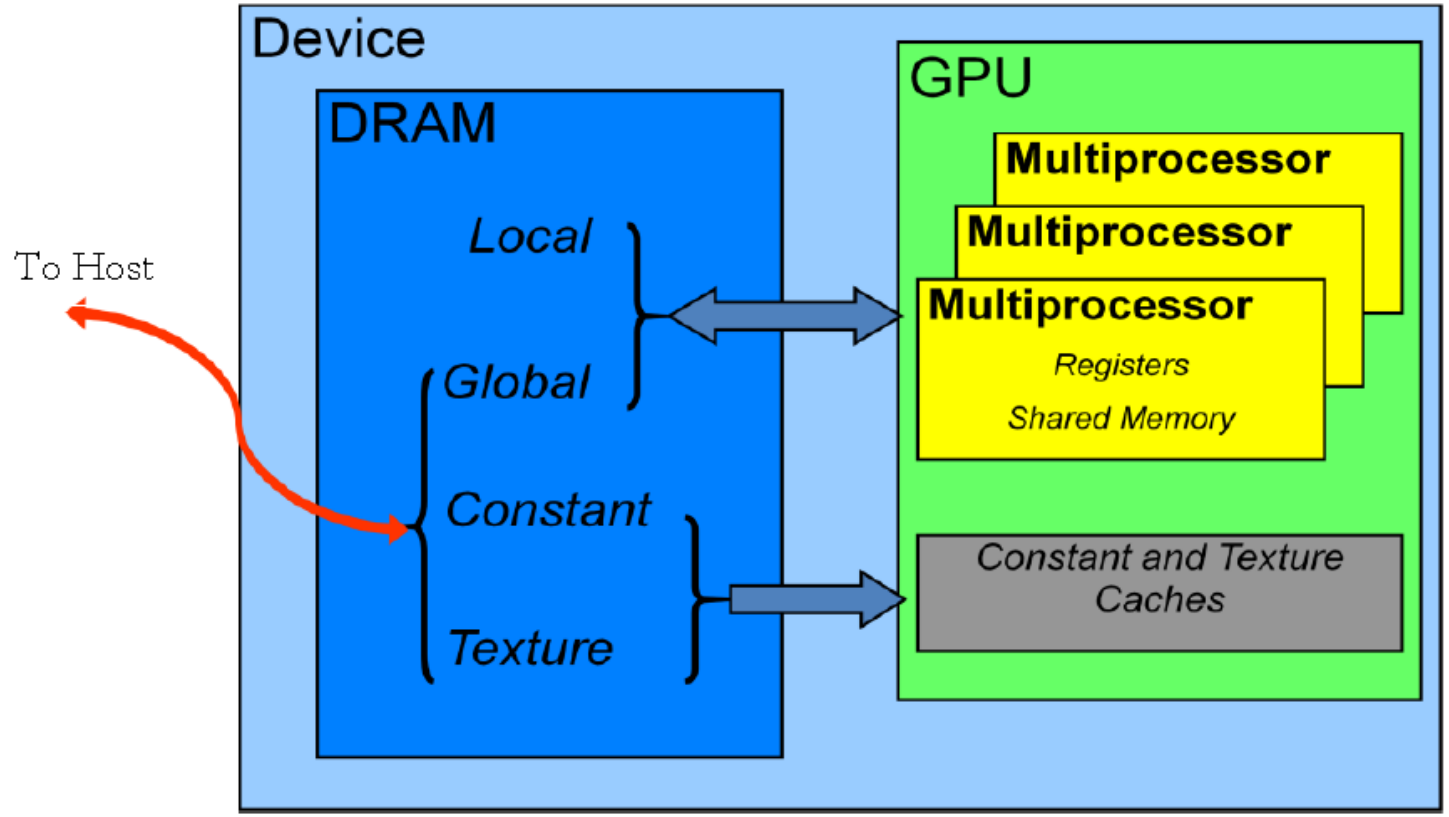
JERARQUÍA DE MEMORIA EN LA GPU

VELOCIDAD CONTRA COSTE



JERARQUÍA DE MEMORIA EN LA GPU

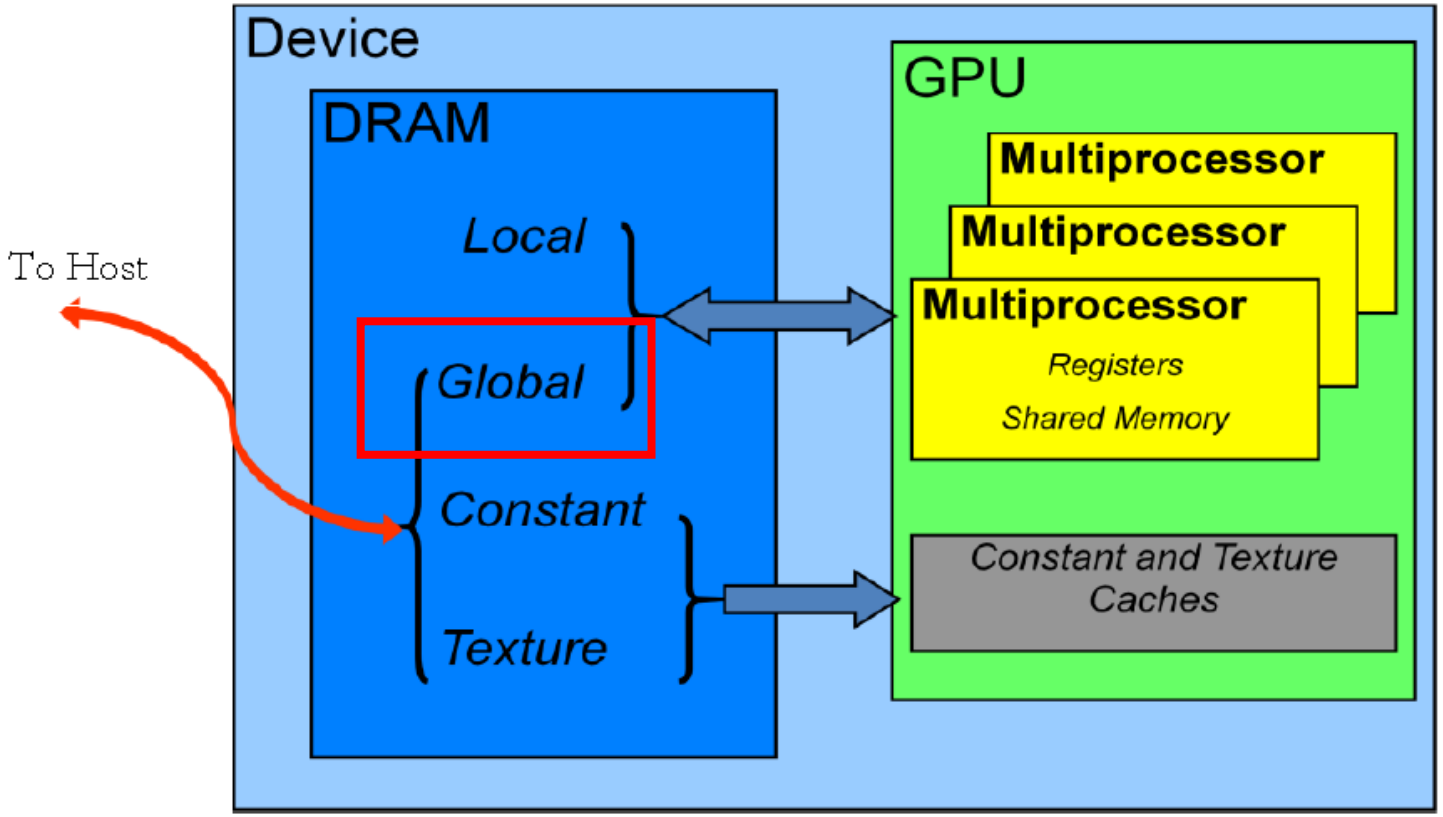
TIPOS DE MEMORIA



JERARQUÍA DE MEMORIA EN LA GPU

MEMORIA GLOBAL

- Compartida por todos los SMs
- Alta latencia y no cacheable
- Alberga datos de aplicación
- Reserva desde CPU (cudaMalloc)
- Liberación desde CPU (cudaFree)
- Transferencia CPU-GPU (cudaMempcy)



JERARQUÍA DE MEMORIA EN LA GPU

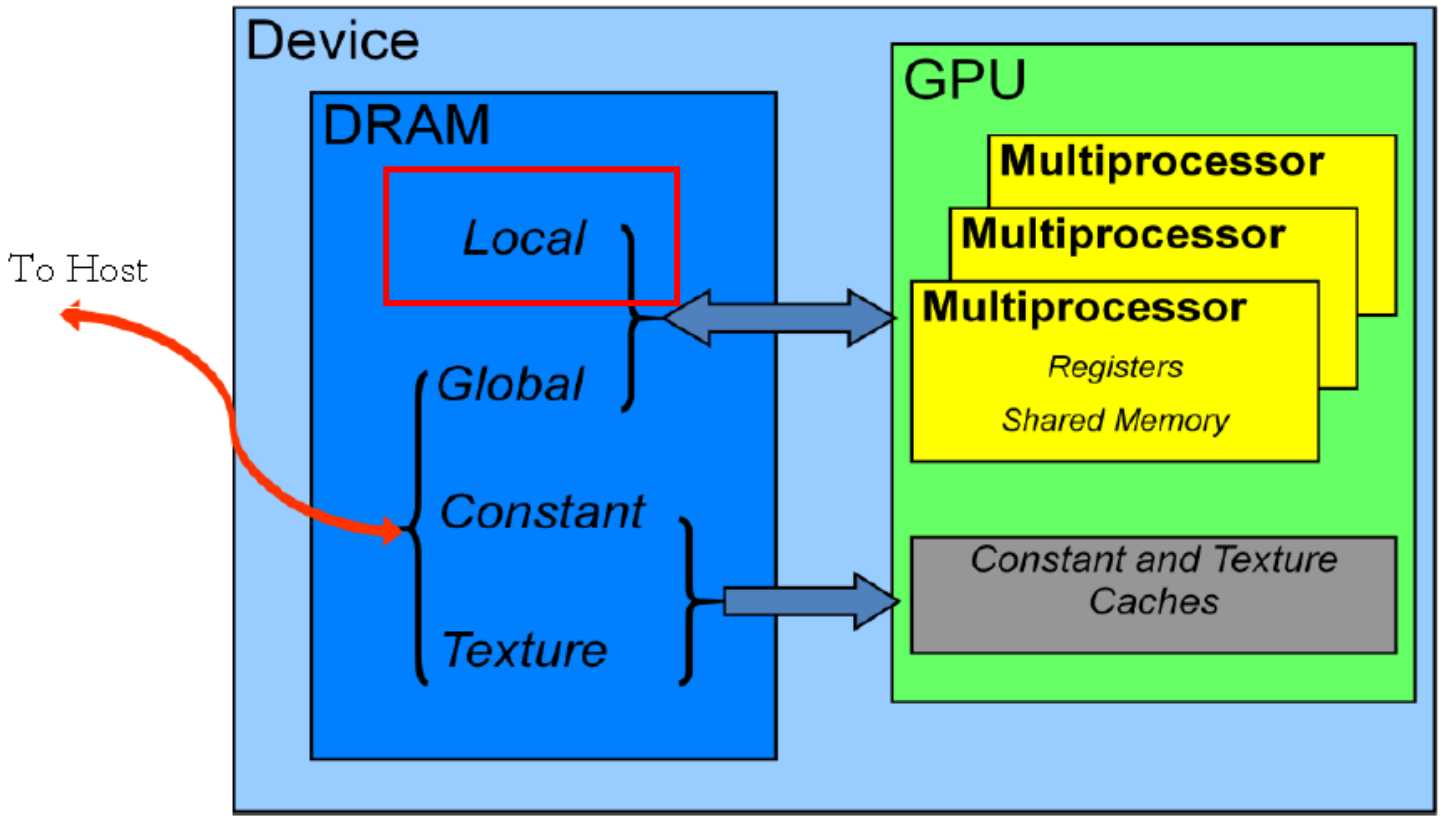
MEMORIA LOCAL

Información local de cada hilo

Parte de DRAM (global) pero cacheable (L1 y L2)

Alberga variables y vectores que no caben en registros

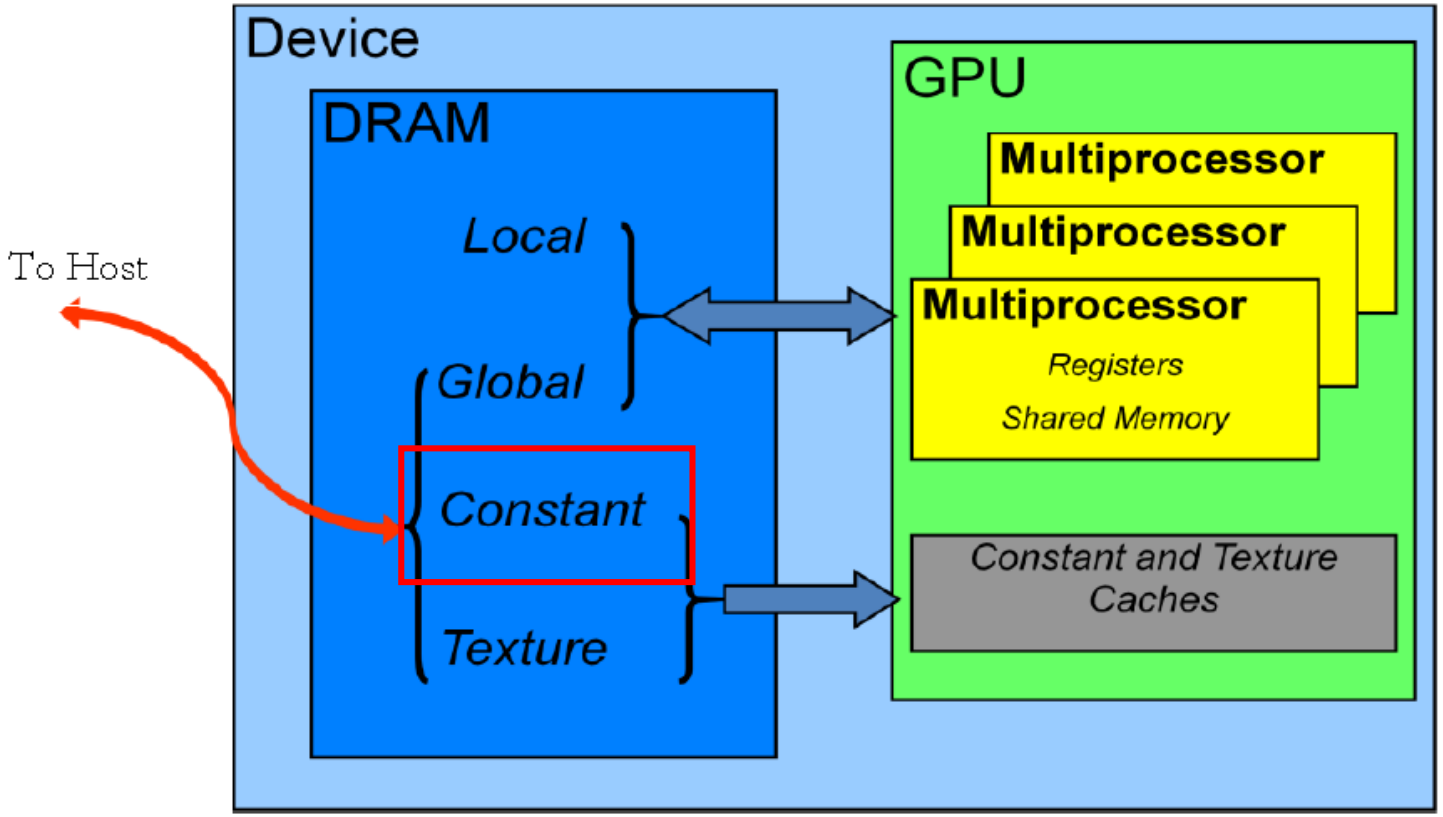
Gestión automática desde GPU



JERARQUÍA DE MEMORIA EN LA GPU

MEMORIA DE CONSTANTES

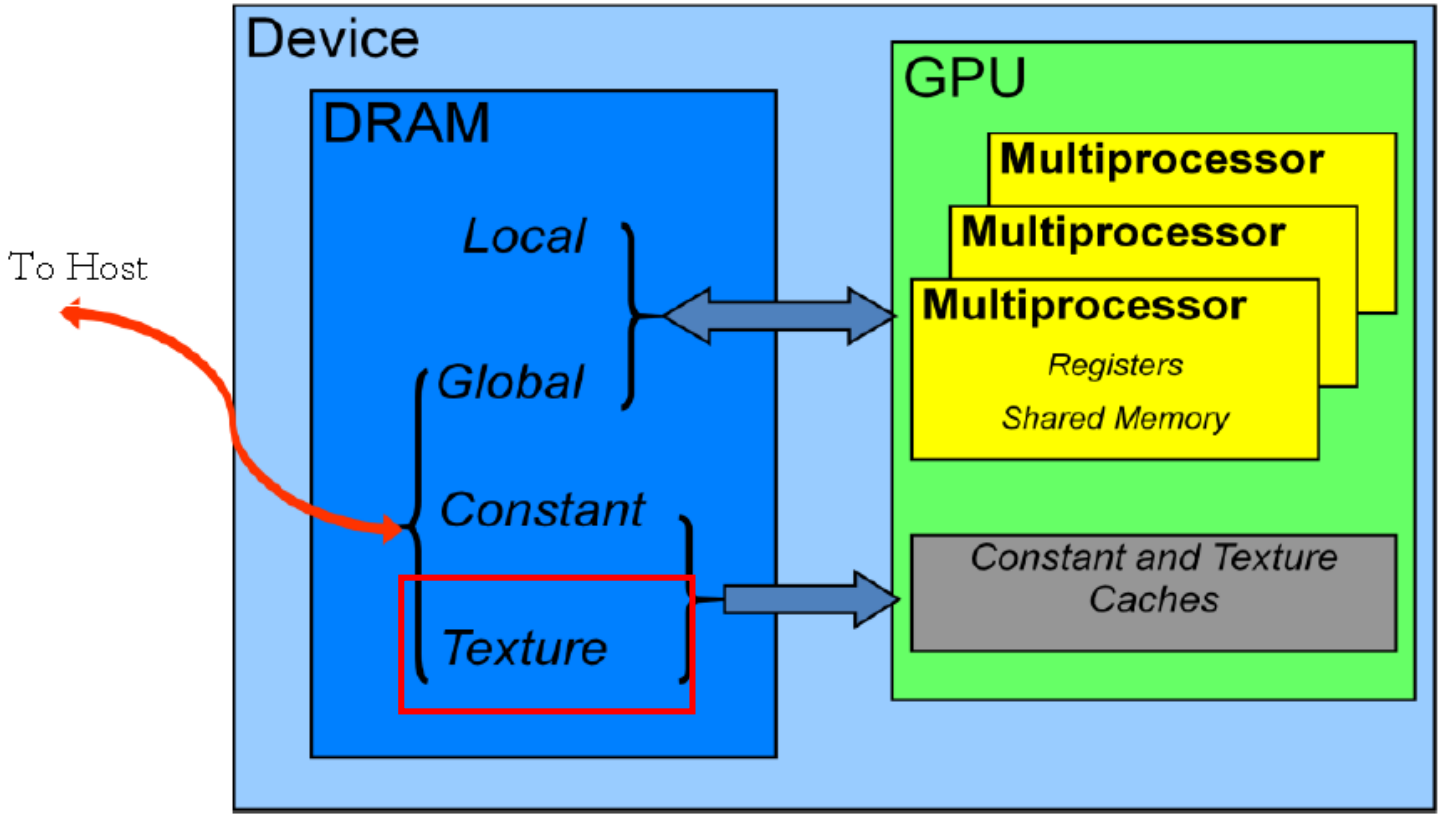
- Compartida por todos los SMs
- Optimizada para solo lectura
- Alberga datos constantes (pero pueden ser modificados)
- Reserva estática desde CPU (__constant__)
- Transferencia CPU-GPU (cudaMempcyToSymbol cudaMemcpyFromSymbol)



JERARQUÍA DE MEMORIA EN LA GPU

MEMORIA DE TEXTURAS

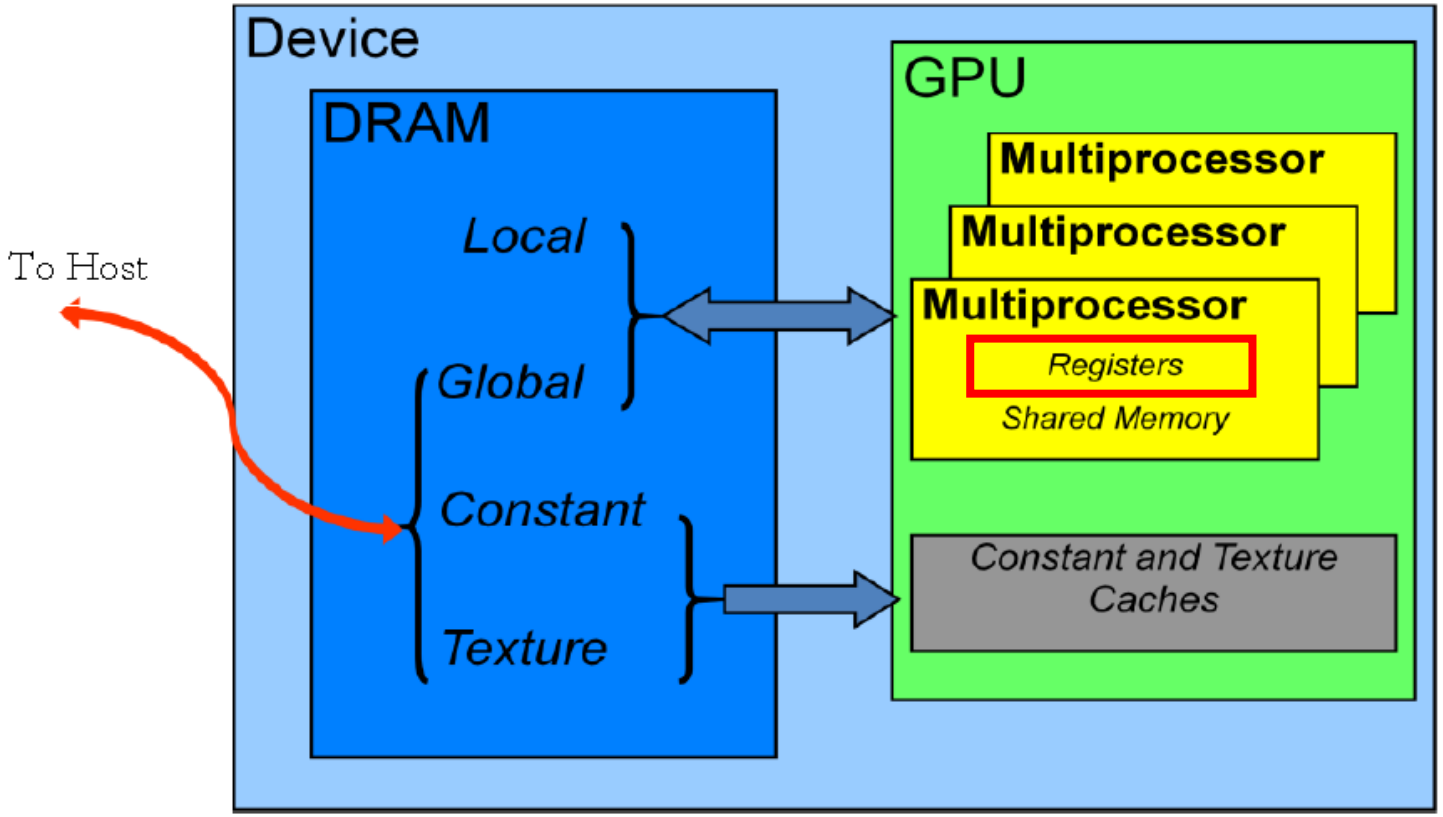
- Compartida por todos los SMs
- Cacheable para solo lectura
- Optimizada para no coalescente
- Alberga datos constantes
- Interpolación o normalización
- Declarada con tipo de datos (cudaArray)



JERARQUÍA DE MEMORIA EN LA GPU

REGISTROS

- Cada SM posee una cantidad fija
- Repartidos entre los hilos
- Extremadamente baja latencia
- Albergan variables de los kernels
- Relativamente limitados en cantidad (depende del número de hilos en ejecución)



JERARQUÍA DE MEMORIA EN LA GPU

MEMORIA COMPARTIDA

Cada SM posee una cantidad fija

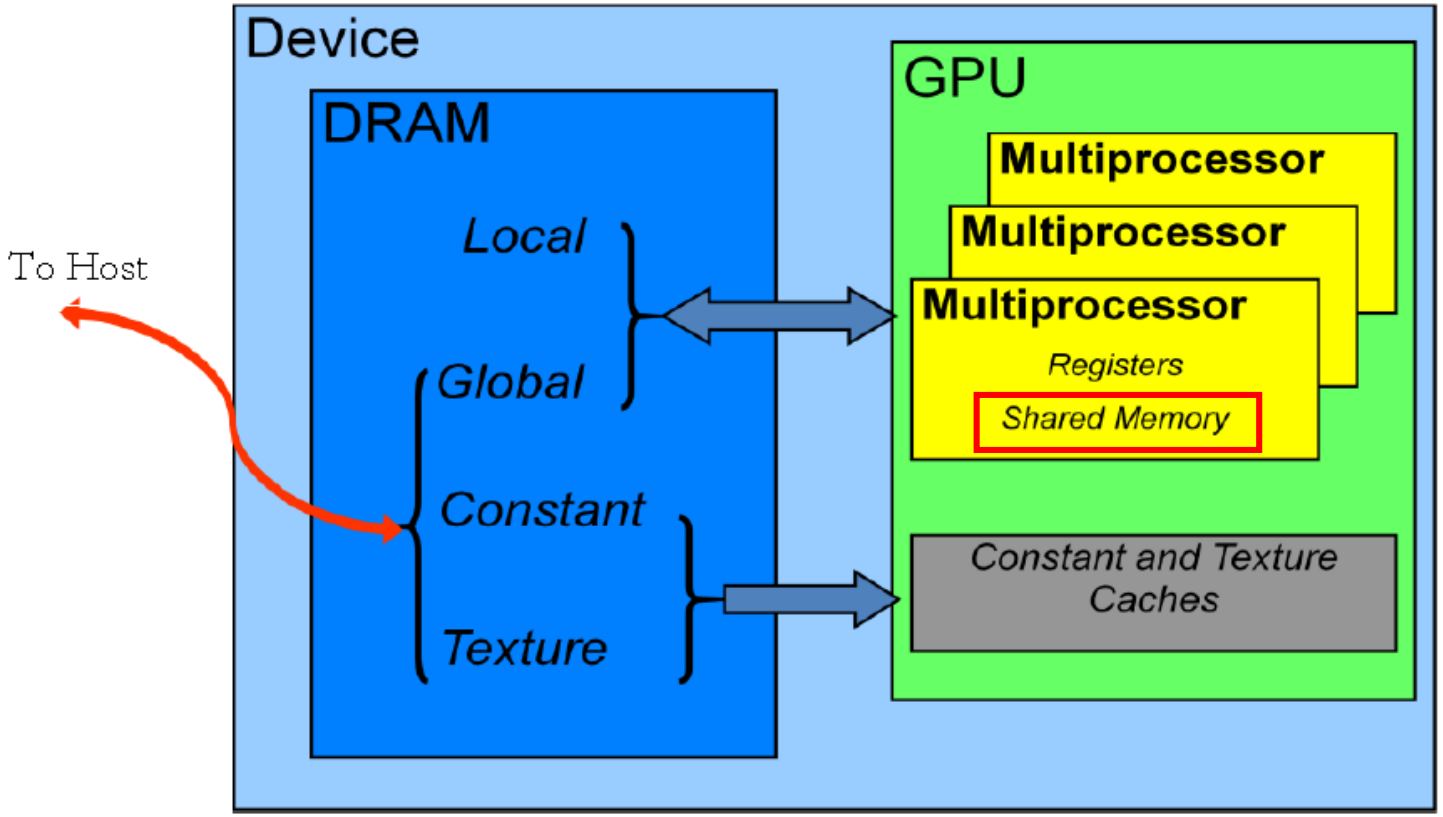
Compartida por los hilos de un mismo bloque

Latencia baja

Albergan datos reutilizables

Reserva en la GPU (__shared__)

Copia de datos manual a posiciones de memoria en kernel



CONSIDERACIONES DE RENDIMIENTO

LIMITACIONES DE LA JERARQUÍA DE MEMORIA (GTX 1080 PASCAL)

Cada SM

64 K registros

96 KiB de memoria compartida

48 KiB caché L1

16 KiB caché constants

2048 hilos

CONSIDERACIONES DE RENDIMIENTO

REGISTER SPILLING

Si la aplicación necesita más registros por hilo de los disponibles:

1) Desbordan a memoria local.
2) Ciertas variables se convierten en arrays para almacenar un elemento por hilo.

PÉRDIDA DE RENDIMIENTO

CONSIDERACIONES DE RENDIMIENTO

ACCESO COALESCENTE

```
__global__ coalesced_access(float*x)
{
  int tid= threadIdx.x+ blockDim.x*blockIdx.x;
  x[tid] = threadIdx.x;
}
```

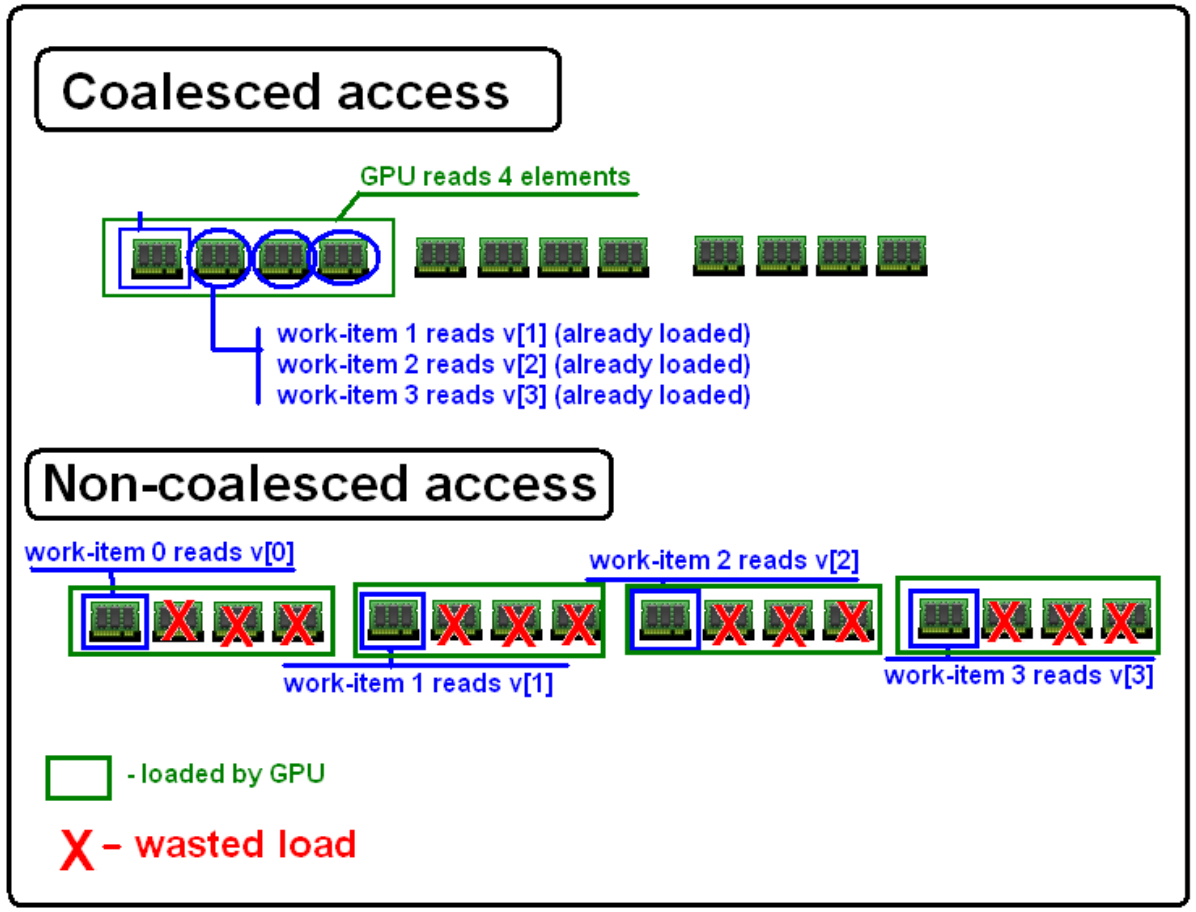
CONSIDERACIONES DE RENDIMIENTO

ACCESO NO COALESCENTE

```
__global__ non_coalesced_access(float*x)
{
  int tid= threadIdx.x+ blockDim.x*blockIdx.x;
  x[tid*100] = threadIdx.x;
}
```

CONSIDERACIONES DE RENDIMIENTO

ACCESO COALESCENTE (OTRO PUNTO DE VISTA)



CONSIDERACIONES DE RENDIMIENTO

USO DE MEMORIA COMPARTIDA

VARIABLES QUE VAN A SER COMPARTIDAS POR TODOS LOS HILOS DE UN BLOQUE

FORMA DE COMUNICACIÓN O SINCRONIZACIÓN ENTRE HILOS DE UN BLOQUE

MEJORA LA REUTILIZACIÓN DE DATOS Y REDUCE LOS ACCESOS A MEMORIA GLOBAL

REDUCE POTENCIALMENTE EL NÚMERO DE REGISTROS NECESARIOS POR HILO

CONSIDERACIONES DE RENDIMIENTO

USO DE MEMORIA COMPARTIDA (DECLARACIÓN)

```
__global__ void medianFilter2D_sm(unsigned char *d_output, unsigned char *d_input)
{
  // Declaración de memoria compartida para almacenar una porción de una imagen
   __shared__ unsigned char d_input_sm_[tamaño de porción compartida];
       […]
}
```

CONSIDERACIONES DE RENDIMIENTO

USO DE MEMORIA COMPARTIDA (COPIA DE DATOS DE GLOBAL A COMPARTIDA)

```
__global__ void medianFilter2D_sm(unsigned char *d_output, unsigned
char *d_input)
{
      […]
  // Copia de datos manual de global a compartida
   d_input_sm[idx_compartida_] = d_input[idx_global_];
      […]
}
```

CONSIDERACIONES DE RENDIMIENTO

USO DE MEMORIA COMPARTIDA (SINCRONIZACIÓN)

```
__global__ void medianFilter2D_sm(unsigned char *d_output, unsigned char *d_input)
{
        […]
  // Sincronización de hilos para asegurar que todos los datos han sido copiados
    __syncthreads();
        […]
}
```

CONSIDERACIONES DE RENDIMIENTO

USO DE MEMORIA COMPARTIDA (UTILIZACIÓN DE DATOS COMPARTIDOS)

```
__global__ void medianFilter2D_sm(unsigned char *d_output, unsigned char *d_input)
{
        […]
  // Utilización de datos de memoria compartida
        north_ = d_input_sm_[idx_north_sm_];
        north_east_ = d_input_sm_[idx_north_east_sm_];
        […]
}
```

MODELO DE MEMORIA

DISPOSITIVOS E INFRAESTRUCTURAS PARA SISTEMAS MULTIMEDIA

GRADO EN INGENIERÍA MULTIMEDIA 2018-2019

Albert García-García < agarcia@dtic.ua.es >

Jose García-Rodríguez < jgarcia@dtic.ua.es>