TALLER DE PROGRAMACIÓN SOBRE GPUS

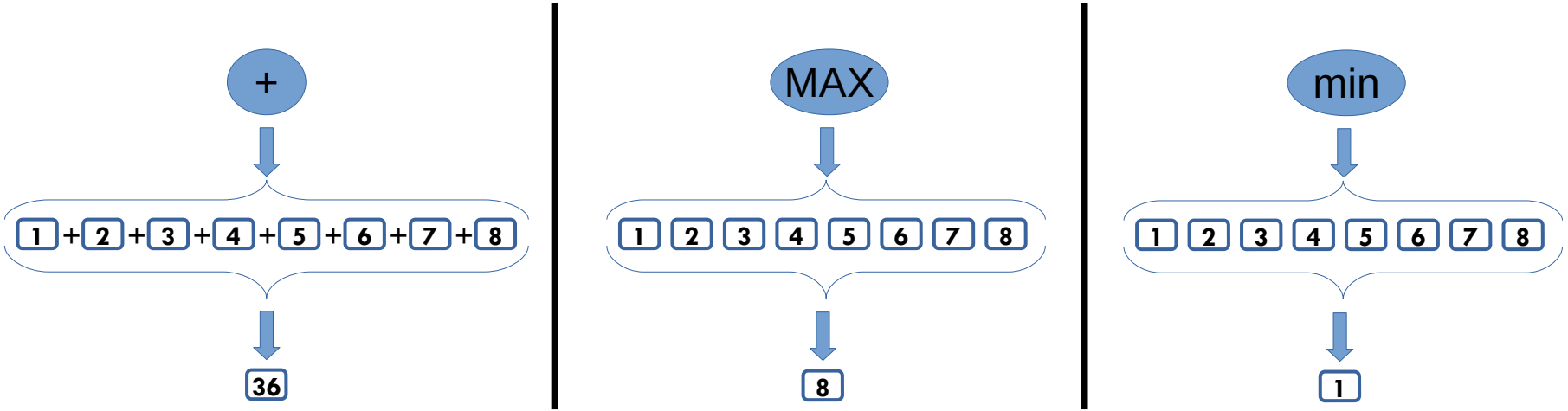
Facultad de Informática – Universidad Nacional de La Plata

Dr. Adrián Pousa

Ejemplo: Reducción

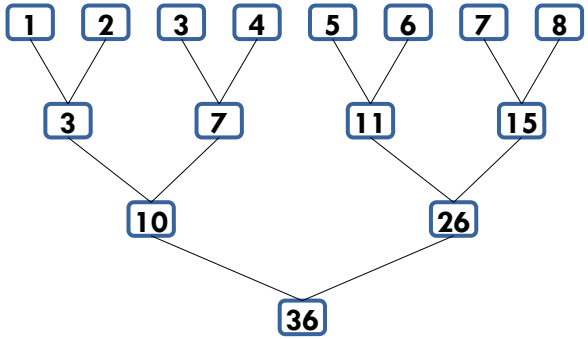
Reducción

- **Reducción:** reducir los valores de un vector a un valor simple.
- Es posible hacerlo aplicando un **operador asociativo** (+, *, MAX, min etc).



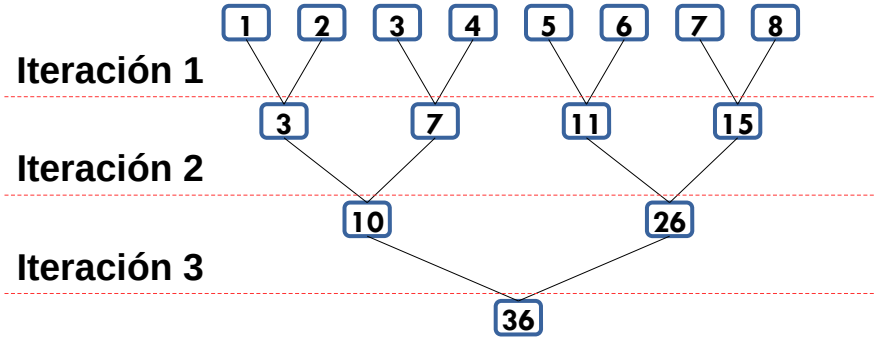
Ejemplo: Suma por Reducción

- En GPU, la solución consiste en una implementación basada en árbol.



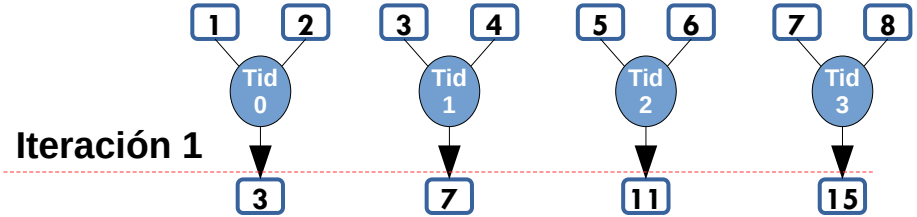
Ejemplo: Suma por Reducción

- Suponer que se tiene un vector de N elementos.
- La solución requiere varias iteraciones, en cada iteración se realiza una nueva llamada al kernel y se aprovecha el hecho de que los valores de las variables en memoria global no cambian entre llamados al kernel.

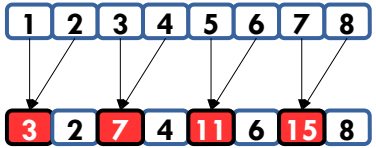


Ejemplo: Suma por Reducción

- Para la primera iteración se crean N/2 hilos.
- Cada hilo suma su posición y la siguiente.

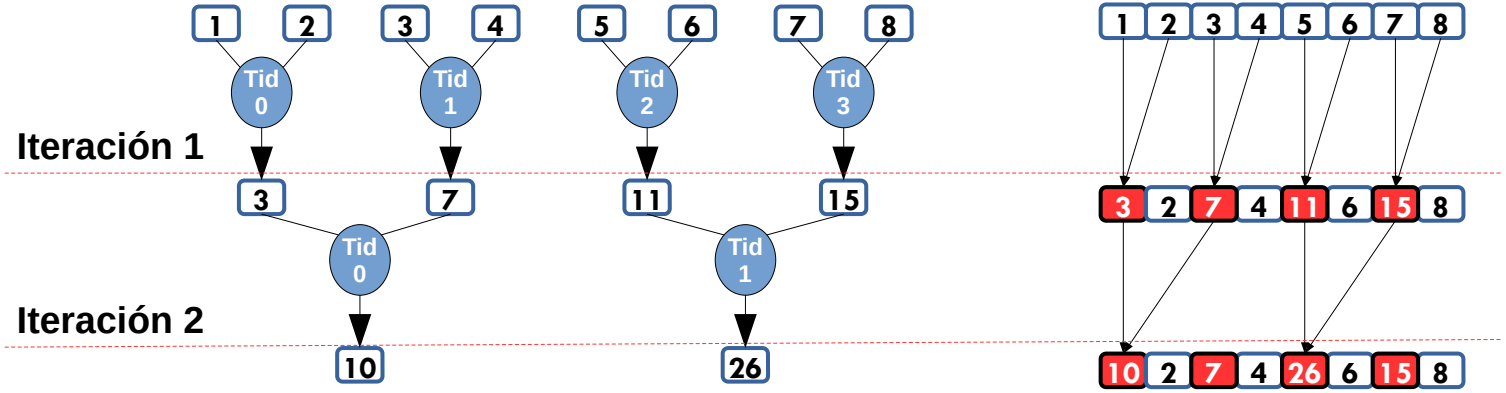


- Para minimizar el espacio de almacenamiento, el resultado de cada suma parcial se deja en la posición del primer operando.



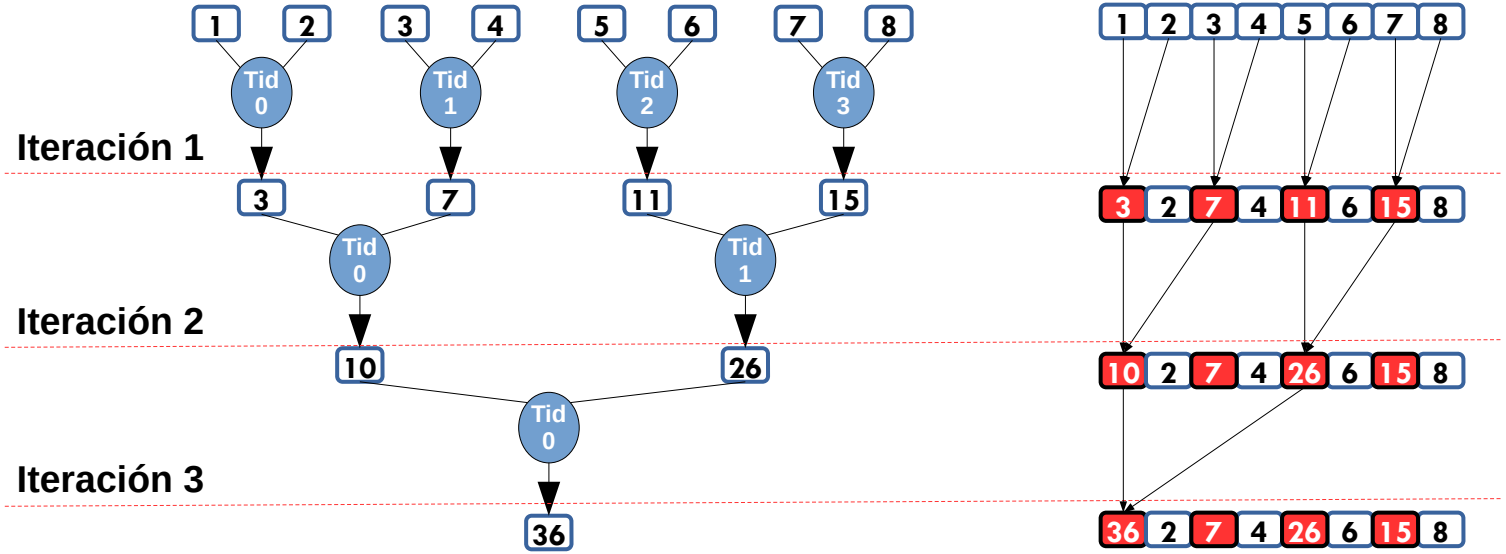
Ejemplo: Suma por Reducción

- Para la siguiente iteración se invoca nuevamente al kernel con la mitad de hilos de la iteración anterior y se sigue con la misma estrategia.



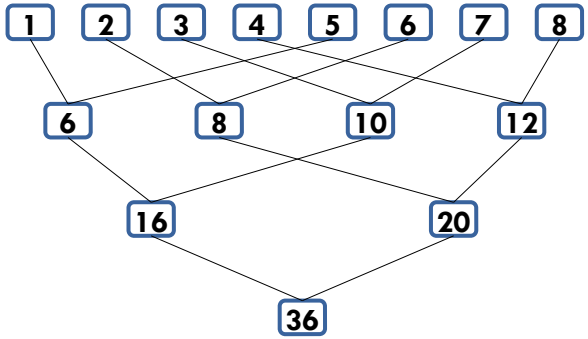
Ejemplo: Suma por Reducción

- En la última iteración se invoca al kernel con un sólo hilo, el resultado final queda en la primer posición del vector.



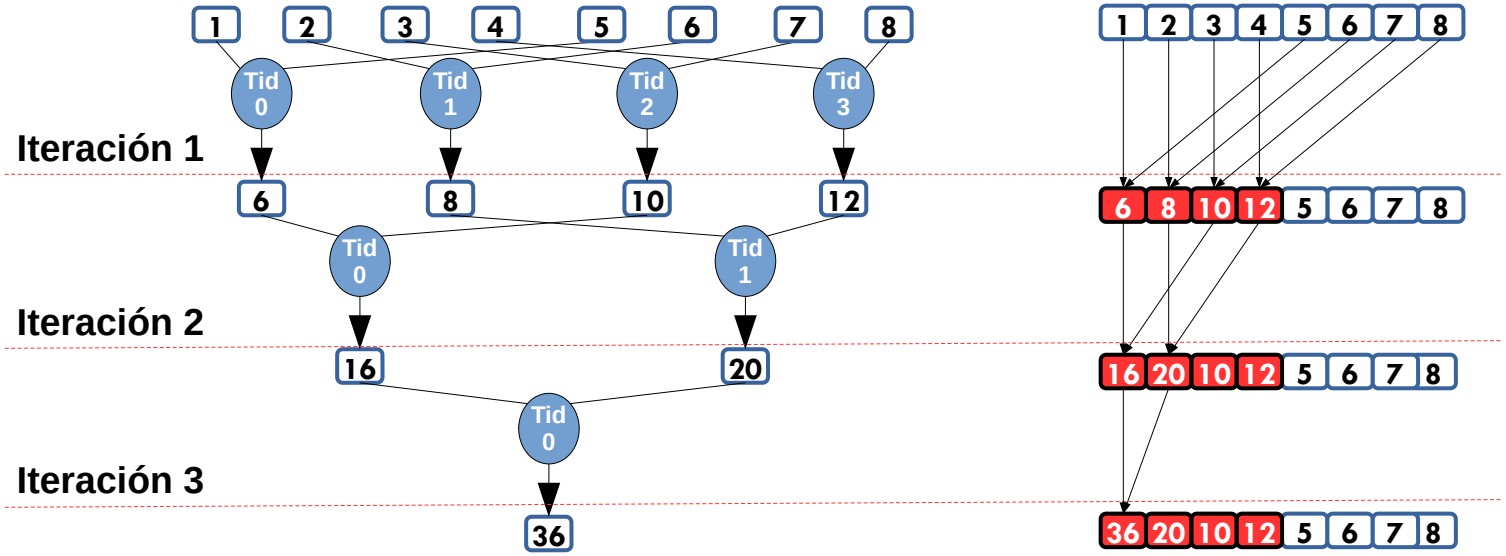
Ejemplo: Suma por Reducción

- Una alternativa, que mejora el acceso a memoria, es utilizar un patrón de acceso a posiciones contiguas.



Ejemplo: Suma por Reducción

- La estrategia con el nuevo patrón de acceso:



Ejemplo: Suma por Reducción

- Pseudocódigo:

```
__global__ kernelReduccion("parámetros"){
 int idx = blockDim.x*blockIdx.x + threadIdx.x;
 If (idx < "límite"){
  vglobal[idx] += vglobal[idx + "distancia"];
 }
}
```

```
int main(int argc, char* argv[]){
 int blockSize = 256;
 dim3 dimBlock(blockSize);
…
 for ("nro iteraciones"){
   dim3 dimGrid("en función de N y dimBlock")
   kernelReduccion<<<dimGrid, dimBlock>>>("parámetros");
   cudaDeviceSynchronize();
 }
...
}
```

Ejemplo: Suma por Reducción

- Pseudocódigo:

???

```
__global__ kernelReduccion("parámetros"){
 int idx = blockDim.x*blockIdx.x + threadIdx.x;
 If (idx < "límite"){
  vglobal[idx] += vglobal[idx + "distancia"];
 }
}
```

```
int main(int argc, char* argv[]){
 int blockSize = 256;
 dim3 dimBlock(blockSize);
…
 for ("nro iteraciones"){
   dim3 dimGrid("en función de N y dimBlock")
   kernelReduccion<<<dimGrid, dimBlock>>>("parámetros");
   cudaDeviceSynchronize();
 }
...
}
```

Ejemplo: Suma por Reducción

- Pseudocódigo:

```
__global__ kernelReduccion("parámetros"){
 int idx = blockDim.x*blockIdx.x + threadIdx.x;
 If (idx < "límite"){
  vglobal[idx] += vglobal[idx + "distancia"];
 }
}
```

El tamaño del problema puede ser menor a la cantidad de hilos por bloque.
Un hilo podría leer una posición de memoria más allá de los límites del problema.

```
int main(int argc, char* argv[]){
 int blockSize = 256;
 dim3 dimBlock(blockSize);
…
 for ("nro iteraciones"){
   dim3 dimGrid("en función de N y dimBlock")
   kernelReduccion<<<dimGrid, dimBlock>>>("parámetros");
   cudaDeviceSynchronize();
 }
...
}
```