Kernels automáticos, vectorización

CUDA Fortran permite la generación de **kernels automáticos**. Esto significa que, con una instrucción especial, *ciclos simples* y *anidados* pueden ser paralelizados de forma automática por el compilador.

Funcionamiento

La directiva es

```
!$cuf kernel do[(n)] <<< grid, block  [optional stream] >>>
```

donde el número `n` indica cuántos ciclos anidados se van a paralelizar. Lo demás son elementos que ya se conocen, como el número de *mallas* y el número de *bloques*.

Cuando el compilador lee esta directiva, se va a construir el código necesario para crear un *kernel*, y de esta forma pueda ser leído y ejecutado por la GPU.

Ejemplo: la distancia Euclideana

La distancia Euclideana es la distancia, en un espacio Euclideano, entre dos puntos, y está definida como

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_j (p_j - q_j)^2}$$

donde $\mathbf{p}$ y $\mathbf{q}$ son dos puntos en este espacio. Estos puntos pueden ser de dimensión arbitraria.

La implementación es sencilla, pues se puede hacer un ciclo de forma inmediata

```
do i = 1,n
    diff = pos_a(i) - pos_b(i)
    dist = dist + (diff ** 2)
end do
dist = sqrt(dist)
```

Y esto es un excelente ejemplo para generar un *kernel automático* en CUDA Fortran.

Restricciones y contraindicaciones

Los *kernels automáticos* son muy útiles cuando se tienen operaciones vectoriales bien definidas, y sobre todo, compactas.

Algunas contraindicaciones son:

- Nunca se debe emplear un *kernel automático* si los índices entre ciclos dependen entre sí.

- No se permite la sintáxis de arreglos de Fortran dentro de un *kernel automático.*
- No pueden existir los comandos `goto` o `exit` dentro de un *kernel automático.*

Algunas restricciones son el uso de los *bloques* y *mallas*, pues aunque el compilador puede inferir estos valores, es **responsabilidad** del usuario definir correctamente el número de estas cantidades para lograr los resultados esperados.