Solució d'alt rendiment per a química computacional

Albert Herrero Rosello
Màster en Enginyeria computacional i Matemàtica
Computació d'altes prestacions

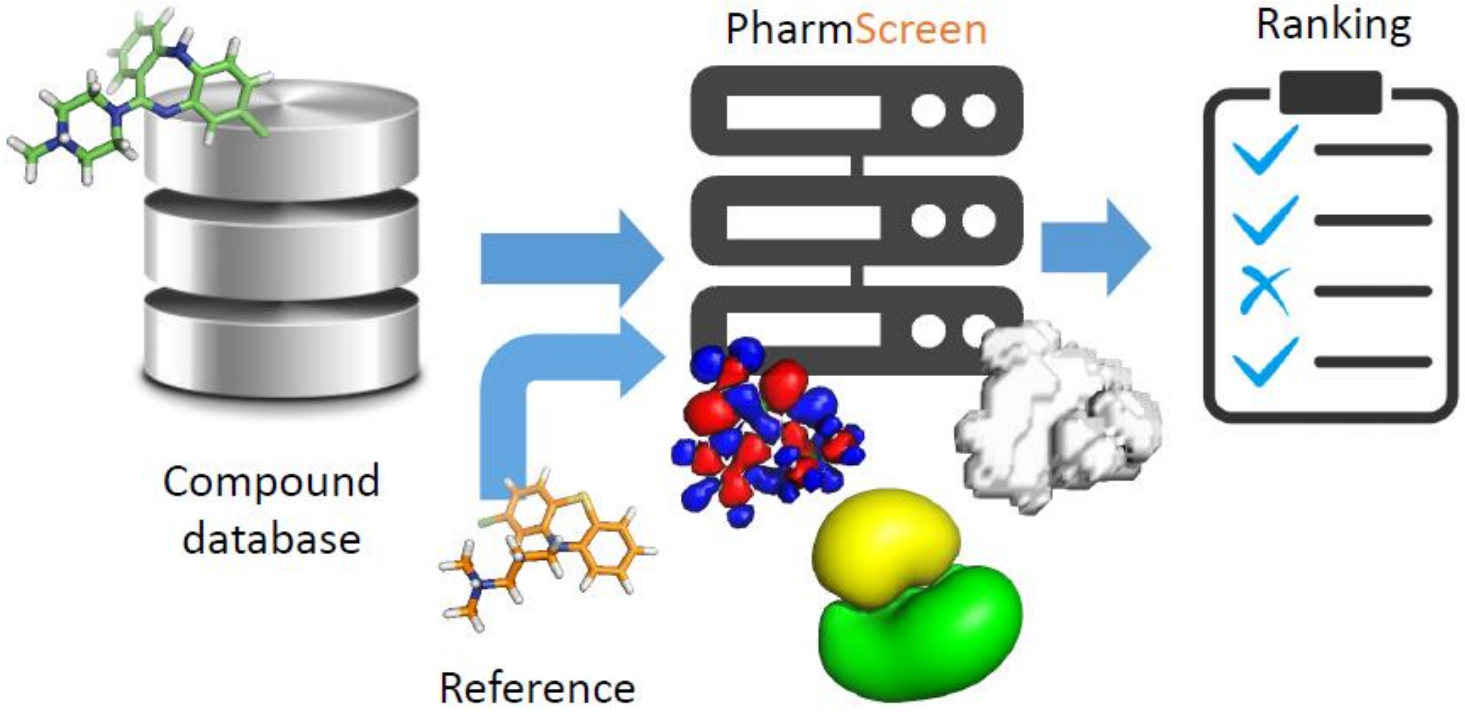
Ivan Rodero Castro
Josep Jorba Esteve

Juny del 2017

1 – Participants en el projecte

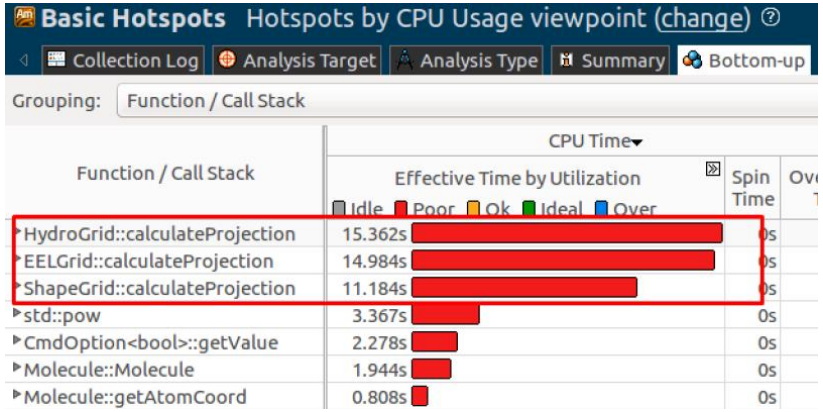
2 – PharmScreen

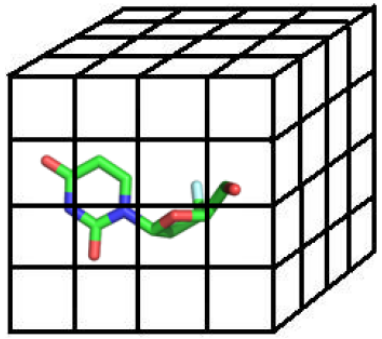
- Eina de virtual screening d'alta presició
- Ajuda en les primers fases de la recerca de medicaments amb l'obtenció de resultats més precisos
- Incrementa el rati de molècules trobades i la diversitat dels lligands gràcies al seu algoritme basat en els camps d'interacció

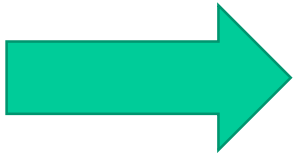


3 – Funció de projecció

- Gasta molt temps de calcul
- Calculen la contribució de tots els àtoms de la molècula en tots els punts de l'espai







```
Projection_function
{
    for each x {
        for each y {
            for each z {
                for each atom {
                    //some calculations
                }
            }
        }
    }
}
```

4 – Acceleració amb Cuda

- Cada thread computa un punt de l'espai

- Cada thread computa la contribució d'un sol àtom en N punts de la coordenada Z

- Un kernel per cada àtom. Cada thread computa tots els punts de la coordenada Z

- Z/N kernels. Cada thread computa N punts de la coordenada Z.

- Cada thread computa un punt de l'espai fent ús de memòria compartida.

4.1 – Primera implementació

- Es llença un sol kernel
- El kernel executa la interacció de tots els àtoms
- Cada thread computa un sol punt de l'espai

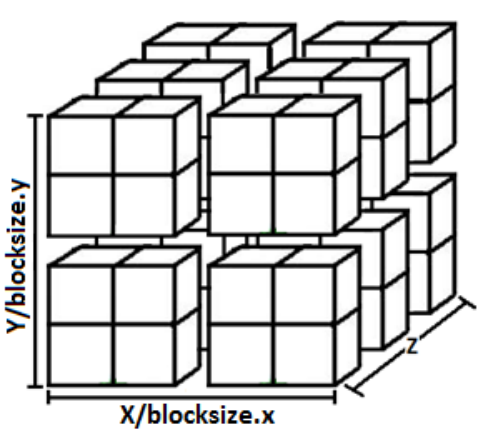
CPU

```
Projection_function
{
    //Launch Kernel
}
```

GPU

```
kernel
{
    ...
    for each atom {
       // some calculations
    }
     ...
}
```

Hi han X * Y * Z threads



BlockSize = {8 ,8 ,1}

GridSize = { X/Blocksize.x, Y/Blocksize.y, Z}

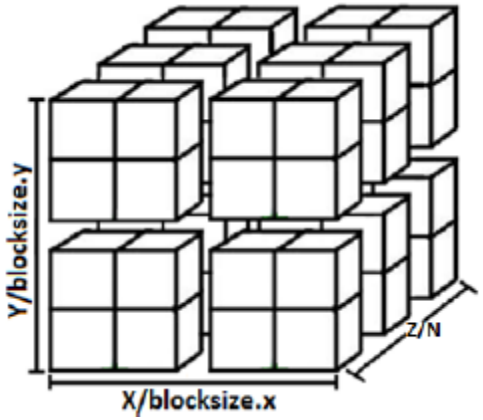
4.2 – Segona implementació

- Es llença un kernel per cada àtom
- El kernel executa la interacció d'un àtom en N punts de la coordenada Z
- Cada thread computa N punts de la coordenada Z

CPU

```
Projection_function
{
    for each atom {
        //Launch Kernel
    }
}
```

Hi han X * Y * (Z/N) threads



GPU

```
kernel
{
    ...
    for i=0 to N {
       // some calculations
    }
     ...
}
```

BlockSize = {8 ,8 ,1}

GridSize = { X/Blocksize.x, Y/Blocksize.y, Z/N}

4.3 – Tercera implementació

- Es llença un kernel per cada àtom de manera concurrent
- El kernel executa la interacció d'un àtom en tots els punts de la coordenada Z
- Cada thread computa tots els punts de la coordenada Z

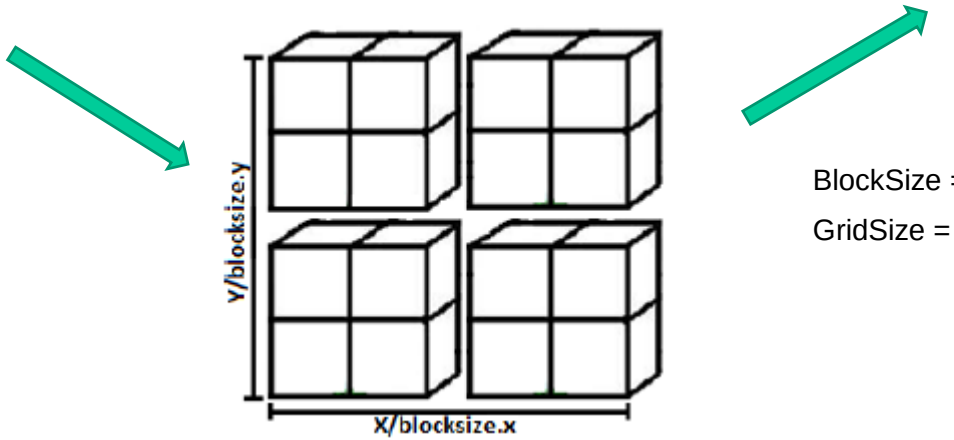
CPU

```
Projection_function
{
     cudaStream_t streams[numAtoms];
     for each atom {
          cudaStreamCreate(&streams[i])
         //Launch Kernel in streams[i]
     }
}
```

GPU

```
kernel
{
     ...
     for i=0 to Z {
        // some calculations
     }
      ...
}
```

Hi han X * Y threads



BlockSize = {8 ,8 ,1}

GridSize = { X/Blocksize.x, Y/Blocksize.y, 1}

4.4 – Quarta implementació

- Es llença concurrentment un kernel per cada grup de N punts de la coordenada Z
- Cada thread computa la contribució de tots els àtoms en N punts de la coordenada Z

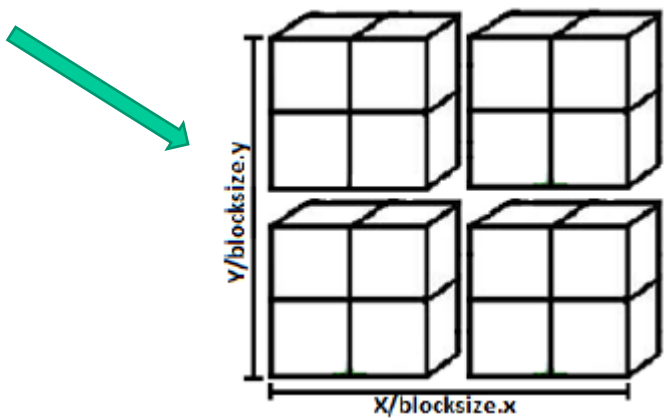
CPU

```
Projection_function
{
     cudaStream_t streams[Z/N];
     for each Z/N {
          cudaStreamCreate(&streams[i])
         //Launch Kernel in streams[i]
     }
}
```

GPU

```
kernel
{
     ...
     for i=0 to N {
          for each atom {
             // some calculations
          }
     }
      ...
}
```

Hi han X * Y threads



BlockSize = {8 ,8 ,1}

GridSize = { X/Blocksize.x, Y/Blocksize.y, 1}

4.5 – Quinta implementació

- Com la primera implementació però, es fa ús memòria compartida
- Es llença un sol kernel
- El kernel executa la interacció de tots els àtoms
- Cada thread computa un sol punt de l'espai

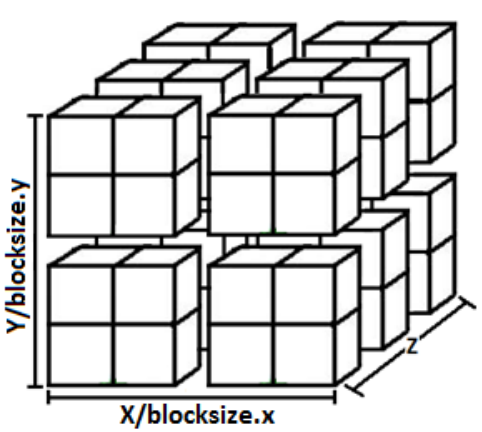
CPU

```
Projection_function
{
    //Launch Kernel
}
```

GPU

```
kernel
{
    ...
    for each atom {
       // some calculations
    }
     ...
}
```

Hi han X * Y * Z threads

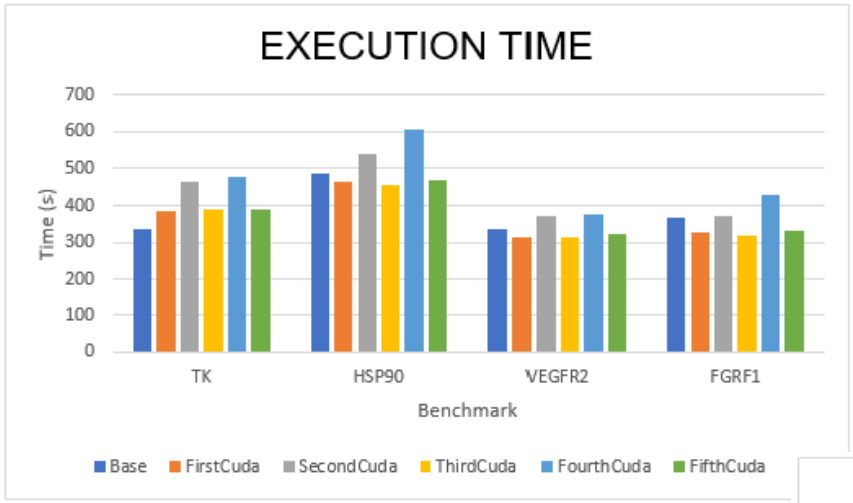


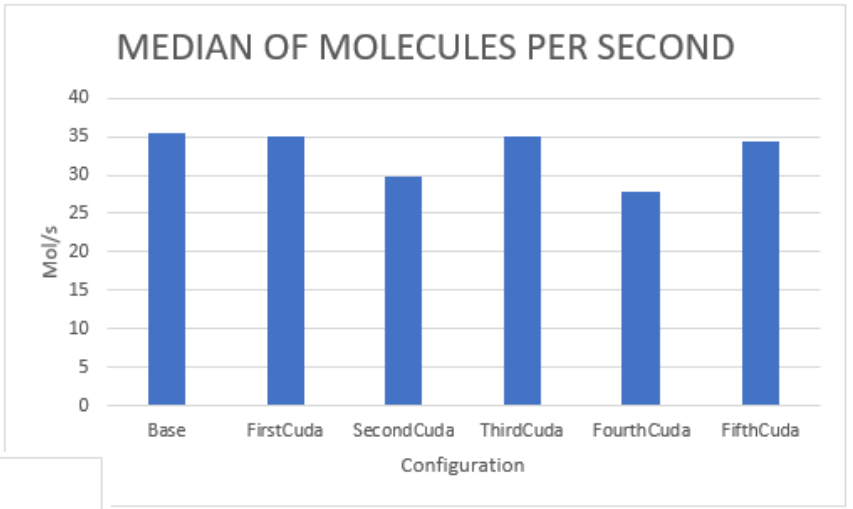
BlockSize = {8 ,8 ,1}

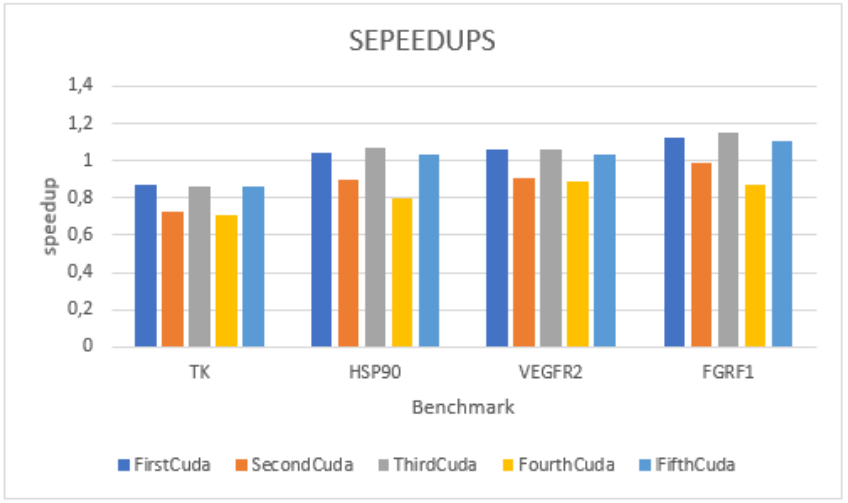
GridSize = { X/Blocksize.x, Y/Blocksize.y, Z}

4.6 – Resultats

- GPU Nvidia Quadro M4000
- 5 execucions = el màxim i el mínim s'exclouen, els resultats són la mitjana de les 3 execucions restants

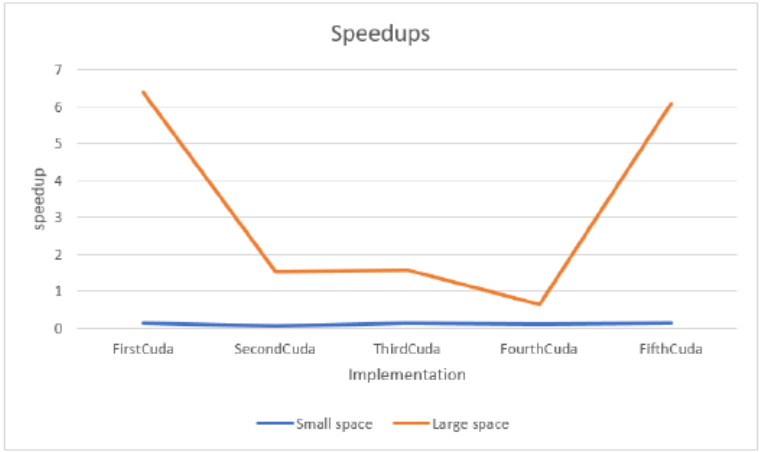


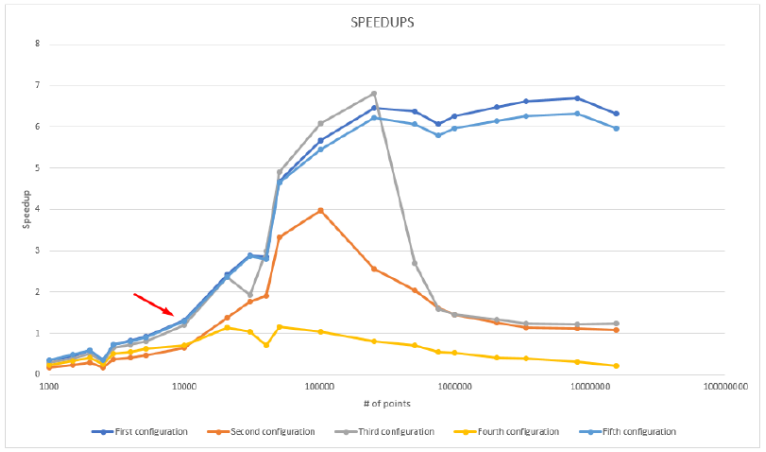


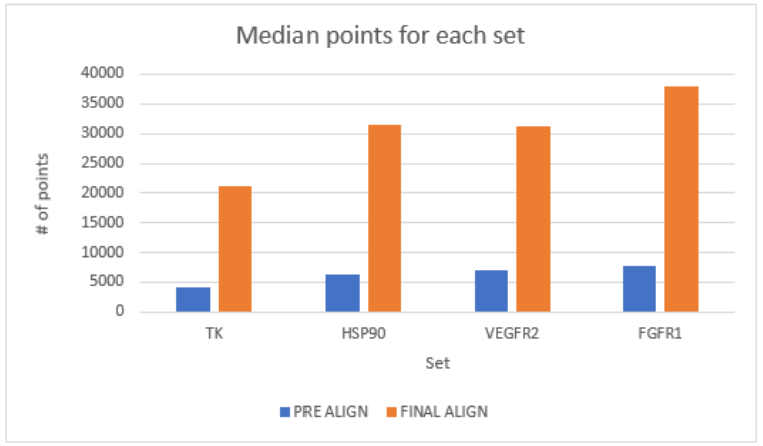


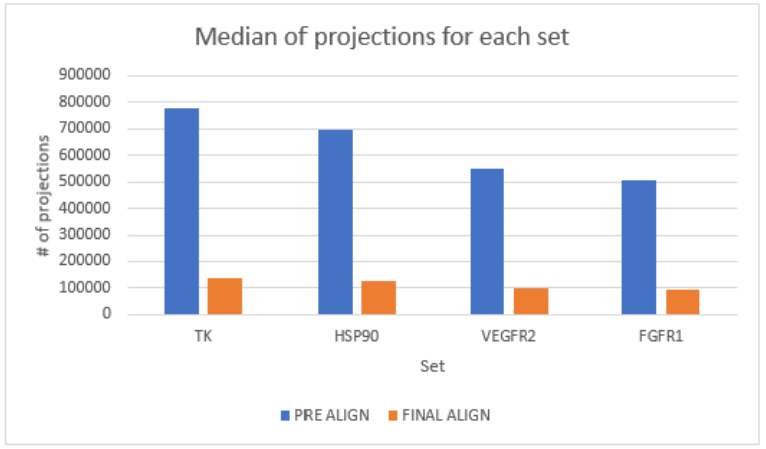
4.7 – Estudi nombre de punts

- Dos mides de grid: 1000 i 1000000 de punts
- 5 execucions = el màxim i el mínim s'exclouen, els resultats són la mitjana de les 3 execucions restants









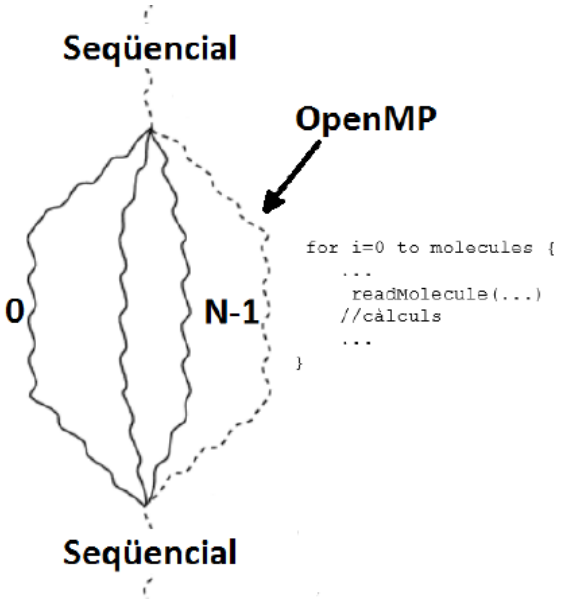
5 – Acceleració amb OpenMP

- Per cada molècula de referència es fa l'alineament de les molècules
- En l'alineament es processen les molècules una a una
- L'alineament de les molècules es pot fer en paral·lel

```
Aligin_functions(ref_molecule)
{
    ...
    molecules = M
    for i=0 to molecules {
        ...
        readMolecule(...)
        //càlculs
        ...
    }
    ....
}
```

```
Aligin_functions(ref_molecule)
{
    ...
    molecules = M
    #pragma omp parallel {
        for i=0 to molecules {
            ...
            readMolecule(...)
            //càlculs
            ...
        }
    }
    ....
}
```

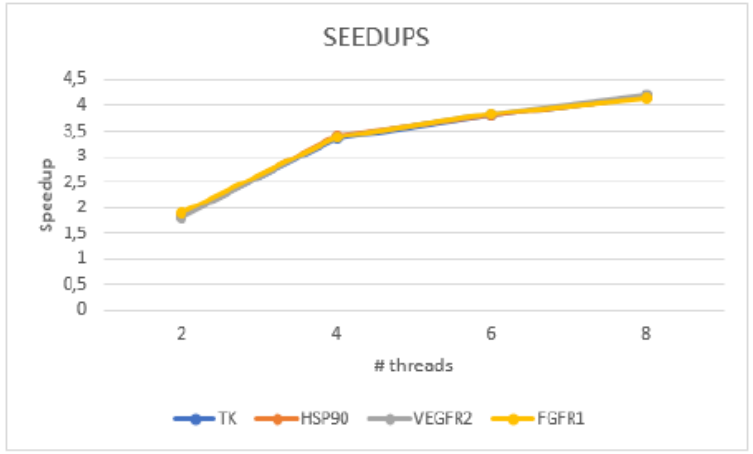
Processament paral·lel



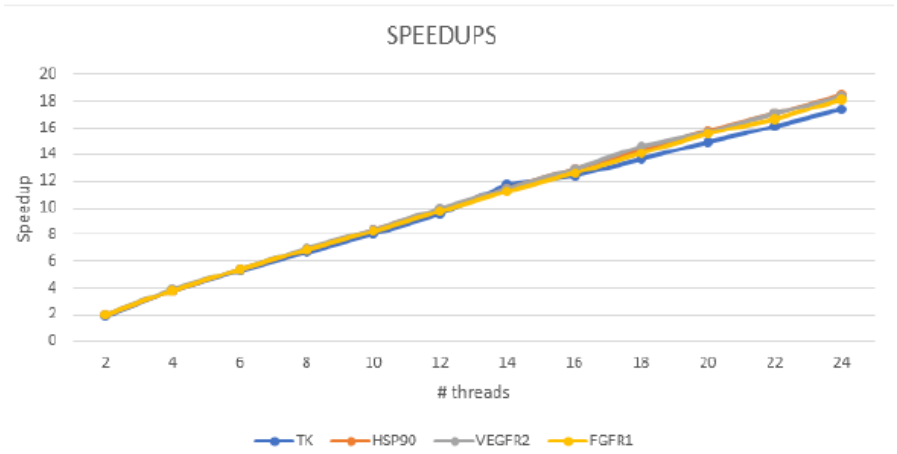
5.1 – Resultats

- Tres workstations: Pharmacelera, Amazon Web Services, Rutgers
- 5 execucions = el màxim i el mínim s'exclouen, els resultats són la mitjana de les 3 execucions restants

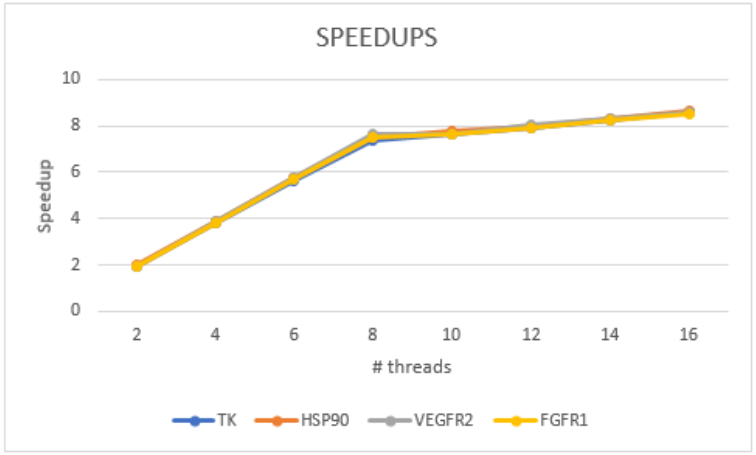
Pharmacelera



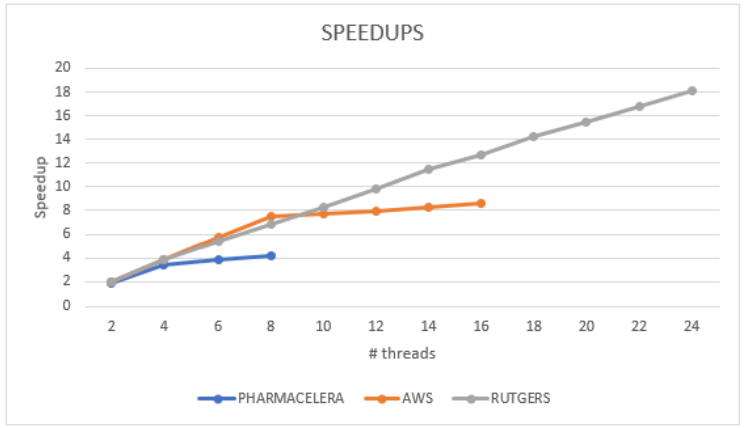
Rutgers



Amazon Web Services

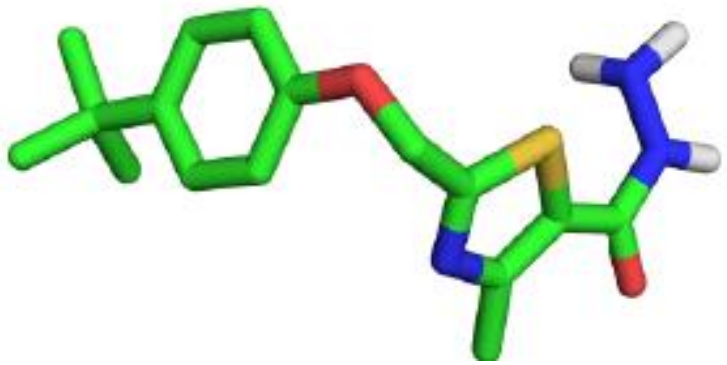


Comparativa



6 – Mopac

- Software de química quàntica
- Integrat com a llibreria estàtica amb PharmScreen
- Input: una molècula
- Output: diferents propietats fisicoquímiques de la molècula

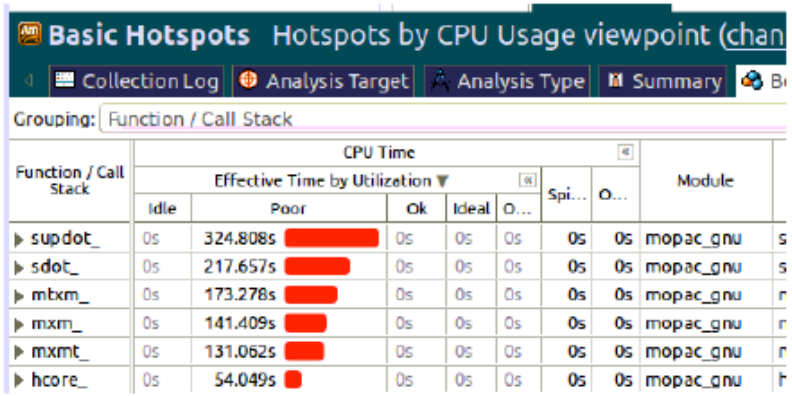


MOPAC

Diferents propietats químiques de la molècula

6.1 – Anàlisi Mopac

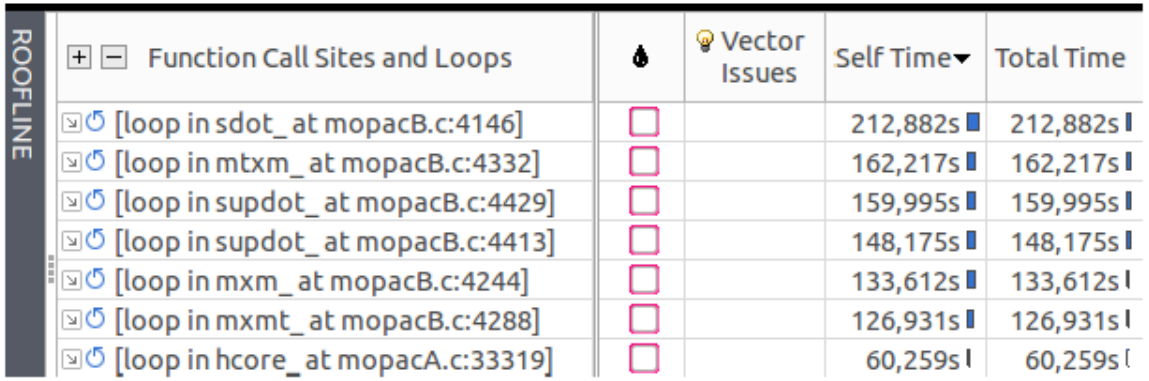
VTune



Gproof

% of time	spend seconds	calls	self s/call	function	execution time
27,98	318,15	7940947	0	supdot_	6,84
18,52	210,57	617062750	0	sdot_	
15,58	177,14	382429	0	mtxm_	
13,05	148,36	600123	0	mxm_	
11,94	135,74	318484	0	mxmt_	
4,76	64,12	222	0,17	hcore_	

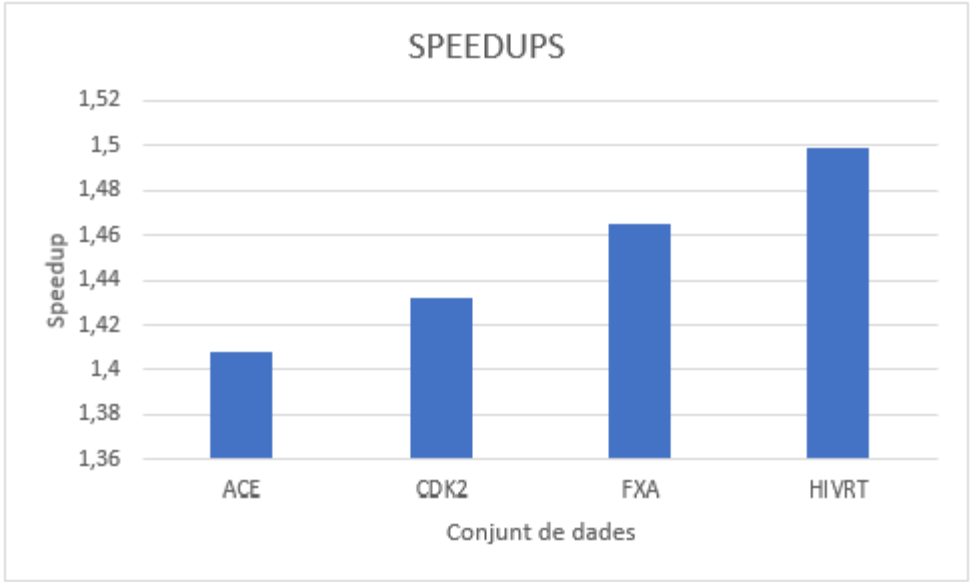
Intel Advisor



6.2 – Solució I resultats

- Els bucles de les funcions treballen sobre matrius de dades
- Les dades estan guardades en memòria per columnes
- Sols es possible vectoritzar el bucle de la funció ‘hcore_’

A11	A21	...	AN1	A12	A22	...	AN2	...	...	A1N	A2N	...	ANN



7 – Conclusions

- No es pot millora el rendiment de les funcions de projecció amb Cuda

- Millora del rendiment de les funcions de projecció de **4x**, **8x** i **18x** amb OpenMP

- Millora del rendiment entre **1.4x-1.5** en la vectorització de la funció ‘hcore_’ de Mopac

Preguntes?