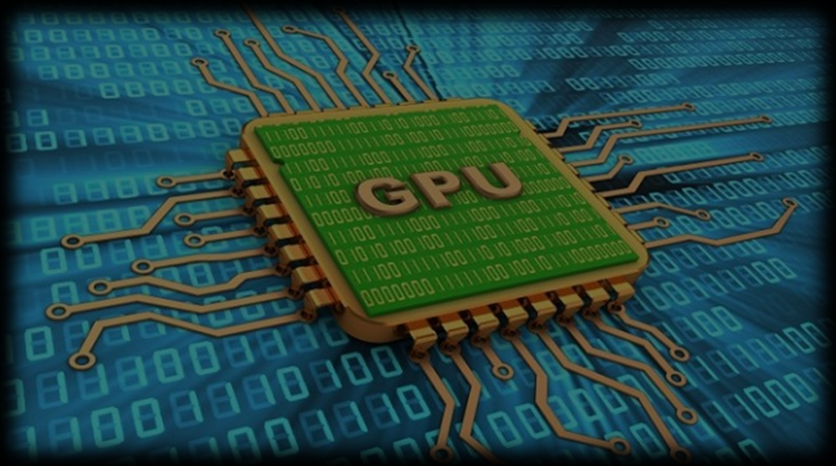
What's in a GPU?

- Heterogeneous chip multi-processor
- Tuned for graphics
- Fast computing

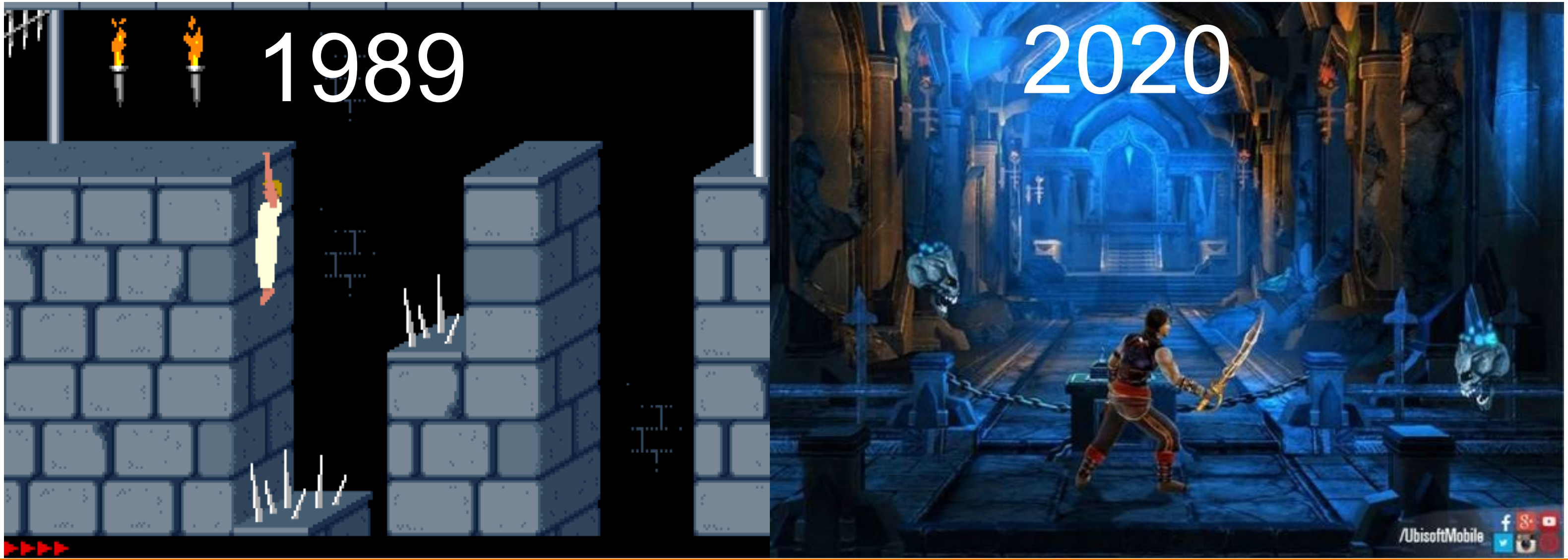


What’s in a GPU?

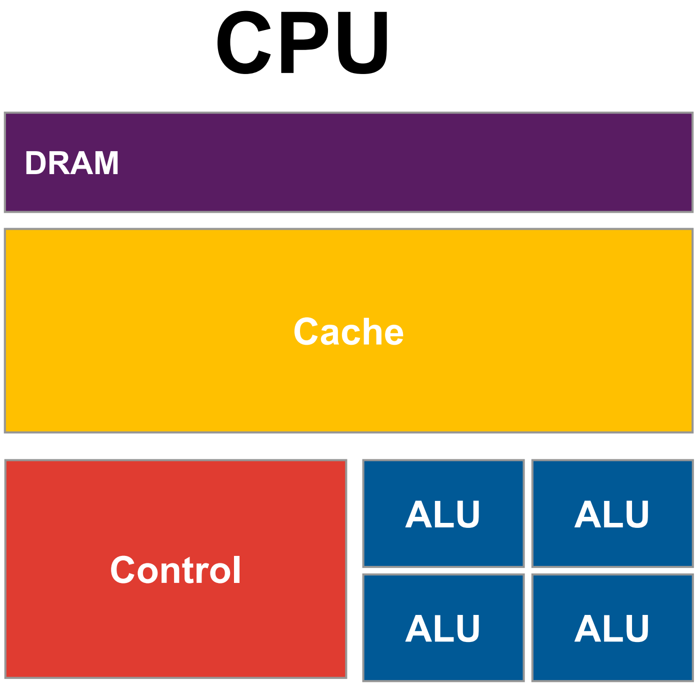
- History :
  - 1990’s real time 3D rendering
    - Video games, Movies, etc.
  - Computationally VERY EXPENSIVE!
  - Before 1990’s graphics were done on CPU’s as well as the framerate!
    - Reduced quality images
    - No 3D rendering

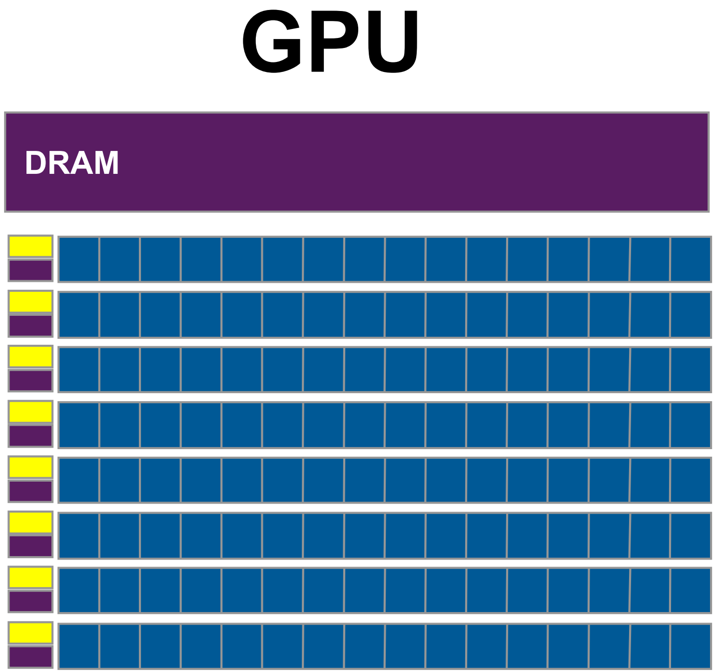
What’s in a GPU?

- History :
  - 1990’s real time 3D rendering
  - Before 1990’s graphics were done on CPU’s



What’s in a GPU?

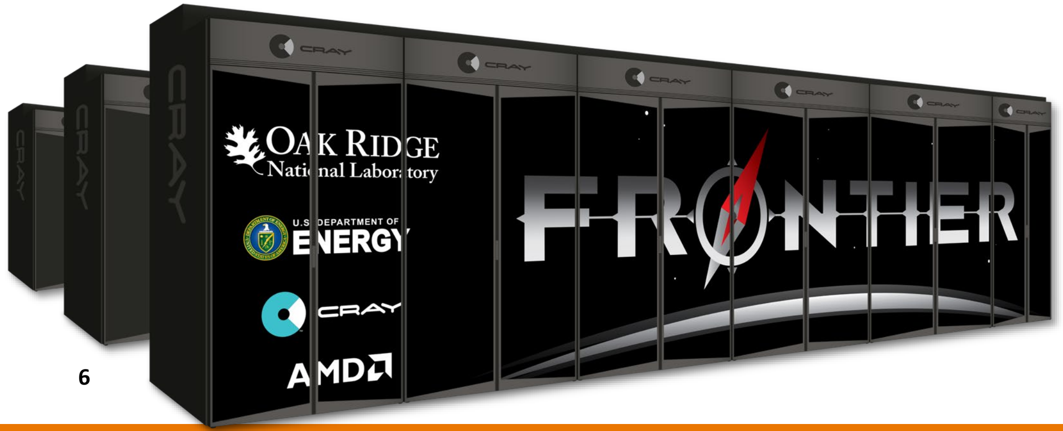




Drug discovery with MegaDOCK on GPUs

Drug discovery with MegaDOCK on GPUs

- **GOAL**
  - Use hip to port a real application to AMD GPUs
    - Evaluate performance due to portability
    - General portability issues
  - Next generation of GPU architectures – Frontier ORNL

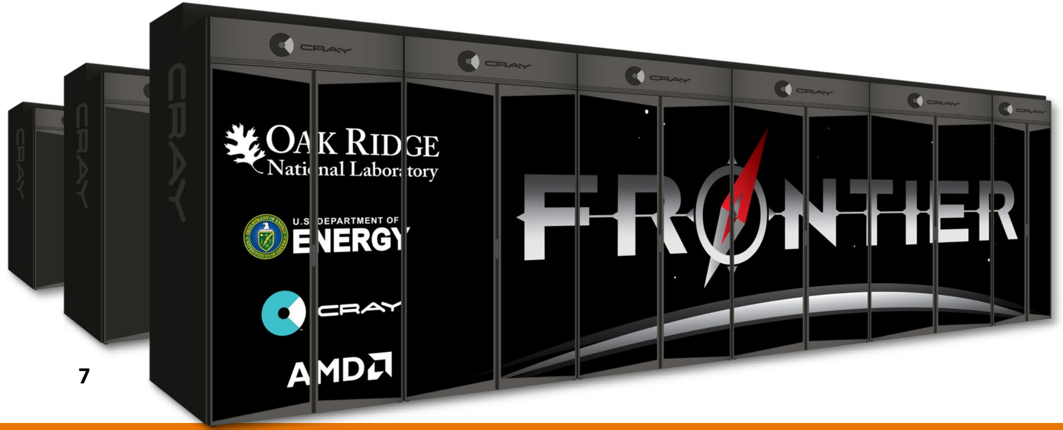


UNIVERSITY OF AMSTERDAM

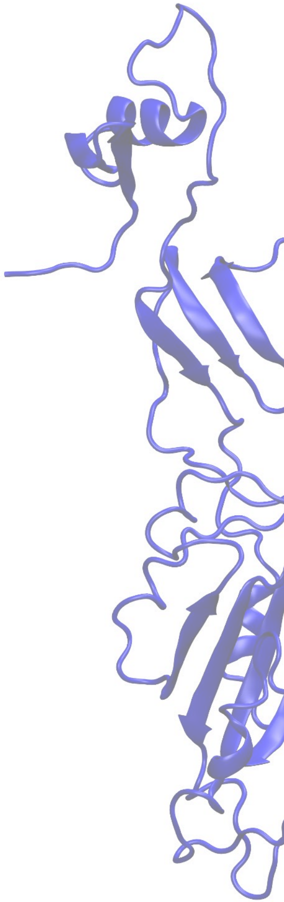
SURF

Drug discovery with MegaDOCK on GPUs

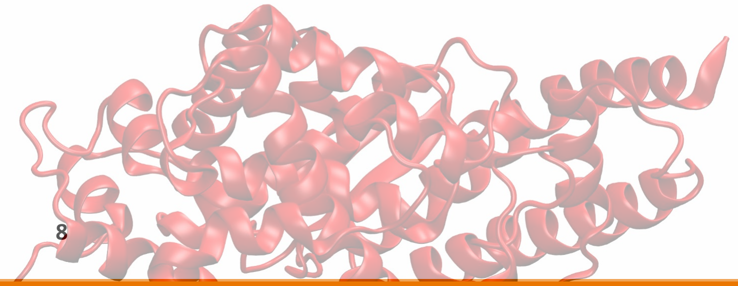
- **GOAL**
  - Preparing for the next generation of GPU architectures
    - Frontier ORNL in 2021
  - Effort of porting applications for these systems
  - Applications working with maximum performance
  - Broad range of applications



Drug discovery with MegaDOCK on GPUs



- **OUTLINE**
  - What is molecular docking?
  - What is MEGADOCK?
  - Algorithm
  - Why to use hip?
  - Timings
  - Zlab benchmark

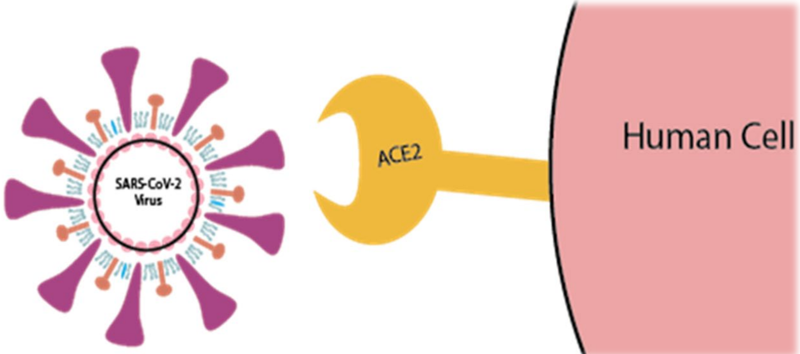


UNIVERSITY OF AMSTERDAM

SURF

Drug discovery with MegaDOCK on GPUs

- **WHAT IS MOLECULAR DOCKING?**
  - Computational tool used to predict protein interactions
    - Determine how two proteins interact in their native environment
    - Aids to understand biological processes at the molecular level
    - Helps to design drugs to fight certain diseases
    - Important to predict conformation and protein interactions
      - COVID-19

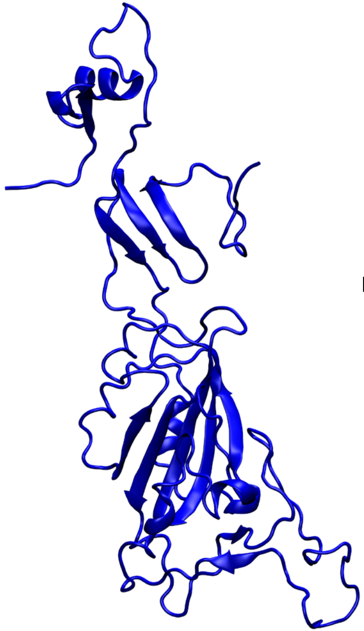


UNIVERSITY OF AMSTERDAM

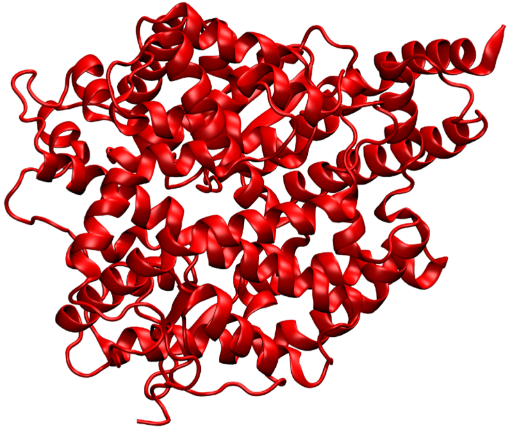
SURF

Drug discovery with MegaDOCK on GPUs

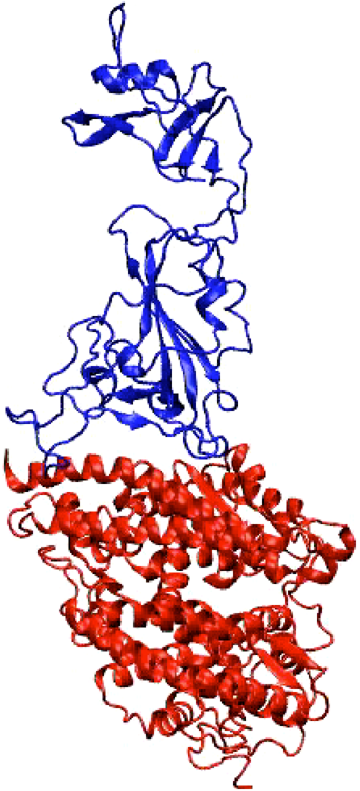
- **WHAT IS MOLECULAR DOCKING?**



+



=



LIGAND

RECEPTOR

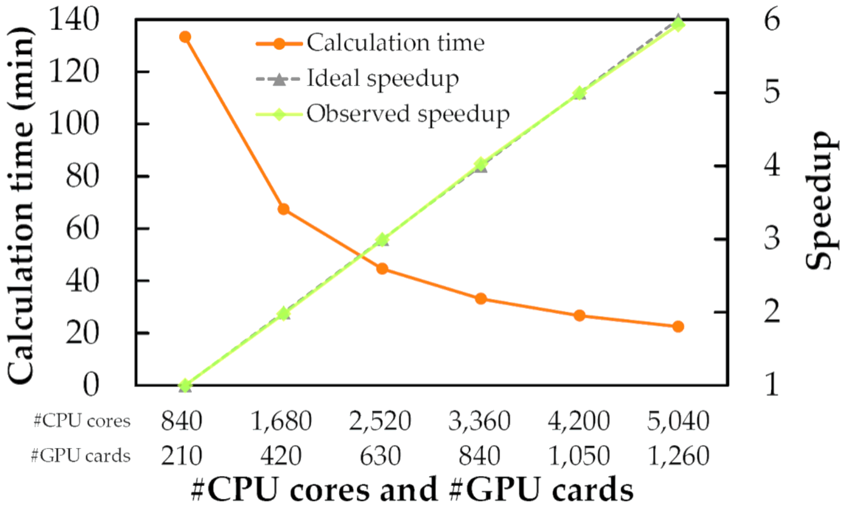
Drug discovery with MegaDOCK on GPUs

- **APPLICATIONS**
  - Different types of docking:
    - Rigid
    - Semi-rigid
    - Flexible

    Different computation demands
  - Within the different applications:
    - **MEGADOCK**
      - Fast
      - Optimized to the GPU by NVIDIA
      - Not very difficult to port

Drug discovery with MegaDOCK on GPUs

- **MEGADOCK**
  - Single node CPU & GPU
  - Multi node CPU and GPU
- **SCALABILITY**



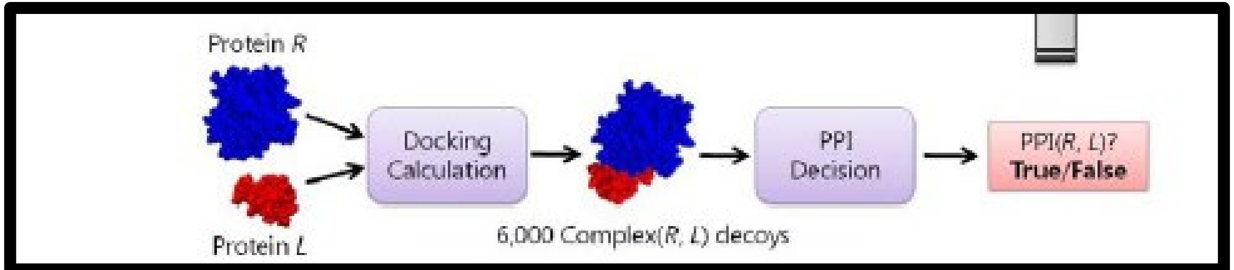
Protein & Peptide Letters, 2014, Vol. 21, No. 8

- MPI/OpenMP & CUDA
- Linear speed up
- Matches the ideal curve
- Speed up of 29x with the GPU usage

UNIVERSITY OF AMSTERDAM

Drug discovery with MegaDOCK on GPUs

- **ALGORITHM**



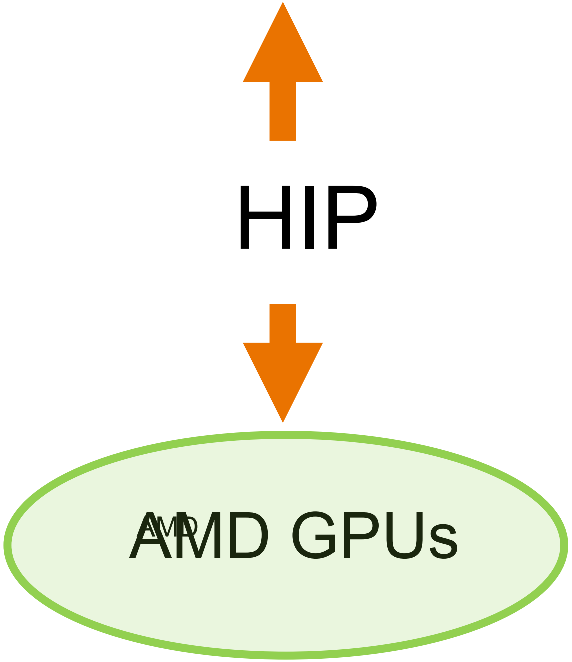

```
calcrg.cpp   control.cpp   cpu_time.h     docking.cpp       fft_process.cu  ligand.cpp  main.cu    parallel.h     pdb_entry.cpp  protein.h
constant.h   control.h     cuda_kernel.cu docking.h         fft_process.h   ligand.h    mpidp.cpp  parameter.cpp  pdb_entry.h    receptor.cpp
contact.py   cpu_time.cpp  decoygen.cpp   fft_process.cpp   helper_cuda.h   main.cpp    mpidp.h    parameter.h    protein.cpp    receptor.h
```

```
cuda_kernel.cu   fft_process.cu   main.cu
```

Protein & Peptide Letters, 2014, Vol. 21, No. 8

Drug discovery with MegaDOCK on GPUs

- **IMPLEMENTATION**
  - Highly optimized for **Nvidia GPUs using CUDA**



Drug discovery with MegaDOCK on GPUs

- **WHY TO USE HIP?**
  - **Example HIP:**
    - **main.cu**

Original Cuda

```
if(nogpu_flag != 1) {
    checkCudaErrors( cudaGetDeviceCount(&device_count_gpu) );
    if (device_count_gpu == 0) {
        fprintf(stderr, "GPU Error: no devices supporting CUDA.\n");
        exit(-1);
    }

    cudaDeviceProp deviceProp;
    checkCudaErrors( cudaGetDeviceProperties(&deviceProp, 0));
    if (deviceProp.major < 1) {
        fprintf(stderr, "GPU Error: device does not support CUDA.\n");
        exit(-1);
    }

    cudaSetDeviceFlags(cudaDeviceMapHost);
    fprintf(stdout, "# Using CUDA device %d: %s\n", 0, deviceProp.name);
    cudaSetDevice(0);
    //fprintf(stdout, "# Init CUDA device OK.\n");

    int cufft_version;
    cufftGetVersion(&cufft_version);
    printf("# CUFFT version : %d\n", cufft_version);
}
#endif
```

After hipify

```
if(nogpu_flag != 1) {
    checkCudaErrors( hipGetDeviceCount(&device_count_gpu) );
    if (device_count_gpu == 0) {
        fprintf(stderr, "GPU Error: no devices supporting CUDA.\n");
        exit(-1);
    }

    hipDeviceProp_t deviceProp;
    checkCudaErrors( hipGetDeviceProperties(&deviceProp, 0));
    if (deviceProp.major < 1) {
        fprintf(stderr, "GPU Error: device does not support CUDA.\n");
        exit(-1);
    }

    hipSetDeviceFlags(hipDeviceMapHost);
    fprintf(stdout, "# Using CUDA device %d: %s\n", 0, deviceProp.name);
    hipSetDevice(0);
    //fprintf(stdout, "# Init CUDA device OK.\n");

    int cufft_version;
    hipfftGetVersion(&cufft_version);
    printf("# CUFFT version : %d\n", cufft_version);
}
#endif
```

Drug discovery with MegaDOCK on GPUs

- ## TIMINGS FOR RIGID DOCKING BENCHMARK

	[1]Summit-dev	Kleurplaat (AMD-ROME)
Single CPU	89.24s	23.96s
Single (CPU & GPU)	5.79s	5.39s

- No loss of performance with Hipify
- Significant speed up from CPU to GPU (~4 times for AMD-ROME)

AMD

[1]Each node has 2 IBM POWER8 CPUs and 4 NVIDIA Tesla P100 GPUs

Drug discovery with MegaDOCK on GPUs

- **ZLAB DOCKING BENCHMARK**

	CPU	GPU
RIGID	138.75s	8.62s
SEMI-RIGID	337.42s	12.07s
FLEXIBLE	607.88s	17.94s

AMD

Drug discovery with MegaDOCK on GPUs

- **CONCLUSIONS**
  - No loss of performance by using HIP
  - Significant speed up by using the AMD GPUs
  - Easy to use
  - Solves issues with portability
  - It is worth trying

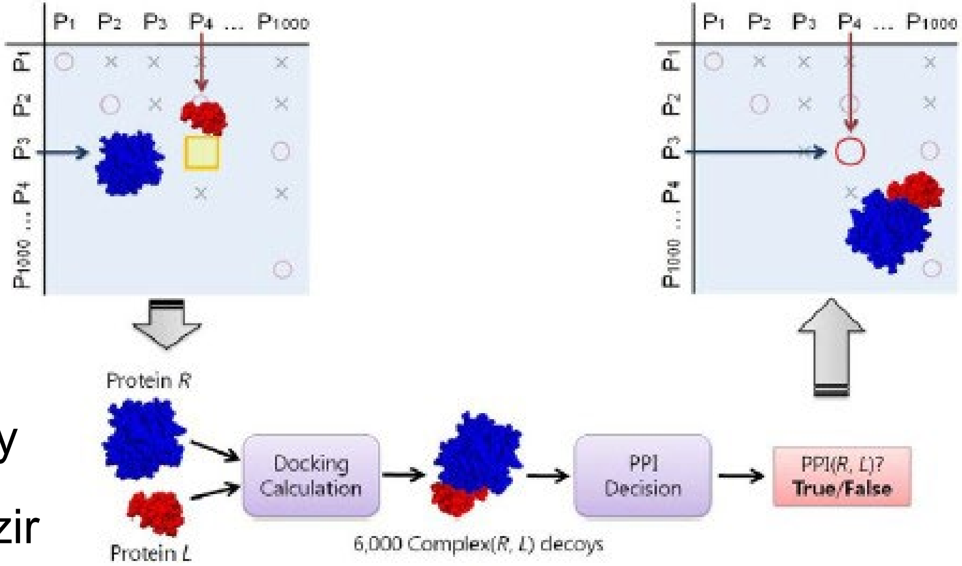
AMD

THANK YOU FOR YOUR TIME

AMD

Porting applications to the GPU using Hip

- **Algorithm**
  - Receptor (R)
  - Ligand (L)
  - FFT based Docking
  - Rigid
  - Pairwise Shape complementarity
  - Scoring Method: Katchalski-Katzir



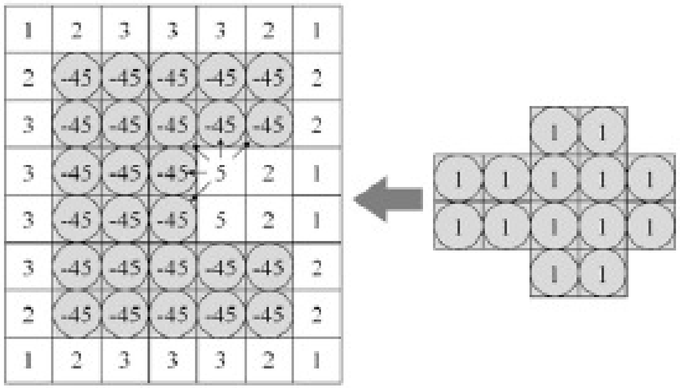
Protein & Peptide Letters, 2014, Vol. 21, No. 8

University of Amsterdam

SURF

Porting applications to the GPU using Hip

- **Algorithm**
  - Scoring
- Three-dimensional voxels
- Square & Circle : area occupied by proteins
- Voxels with only the squares represent unoccupied space



Receptor rPSC G_R

Ligand rPSC G_L

Protein & Peptide Letters, 2014, Vol. 21, No. 8

UNIVERSITY OF AMSTERDAM