Parallel Computation of Non-Bonded Interactions in Drug Discovery: Nvidia GPUs vs. Intel Xeon Phi

Jianbin Fang, Ana Lucia Varbanescu, Baldomero Imbernón, José M. Cecilia, and Horacio Peréz-Sánchez

Bioinformatics and High Performance Computing Research Group

http://bio-hpc.eu

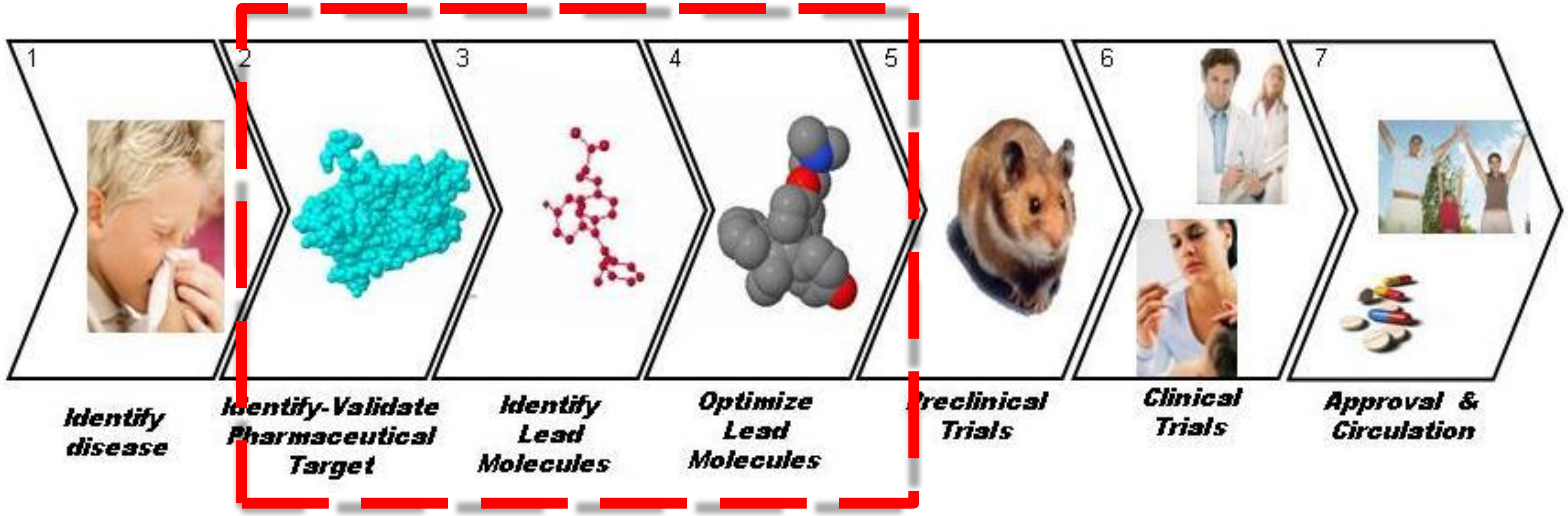
Universidad Católica San Antonio de Murcia (UCAM)

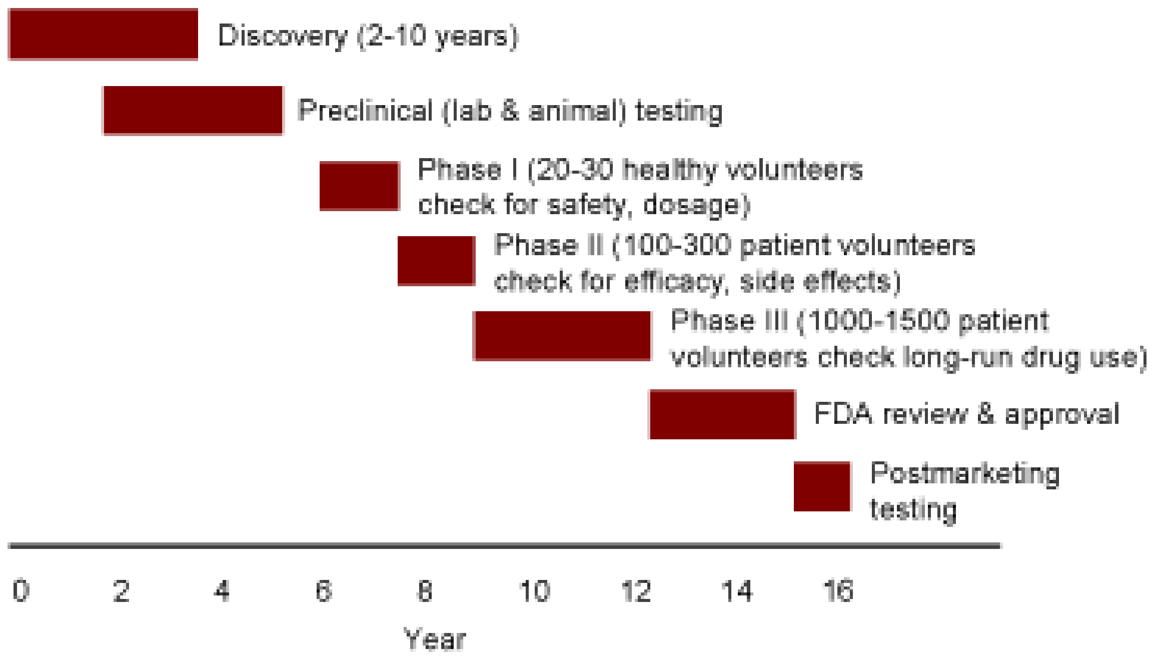
- **Drug Discovery and Virtual Screening**

- Non-bonded interactions kernel implementations
  - Cell Broadband Engine
  - GPU
  - Cluster; MPI/OpenMP
  - Intel Xeon Phi

- Conclusions and outlook

DRUG DISCOVERY PROCESS

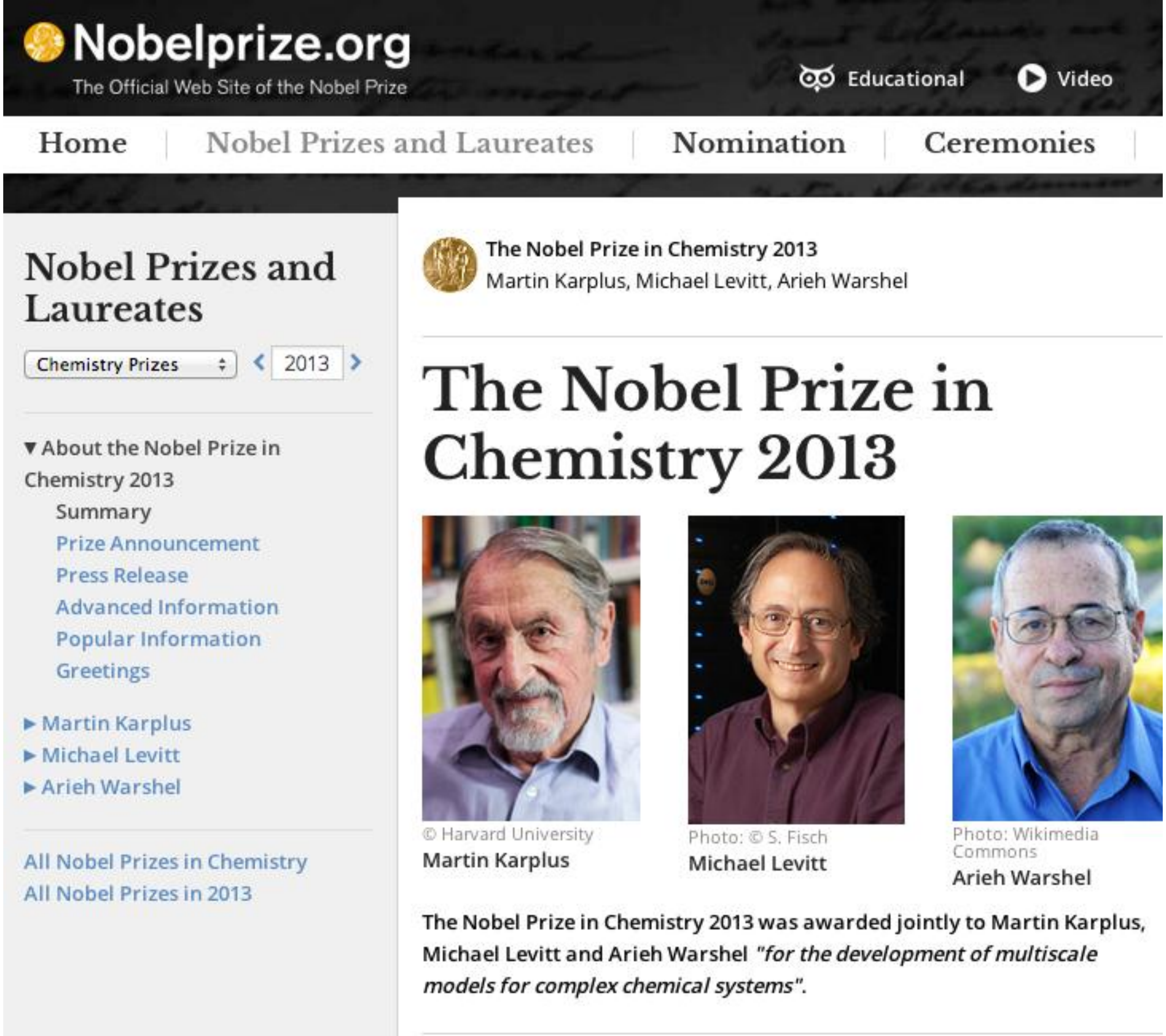




Methods for ligand database screening:

Screening in laboratory:

- Automatized,
- but expensive
- and time-consuming



The Nobel Prize in Chemistry 2013
Martin Karplus, Michael Levitt, Arieh Warshel

The Nobel Prize in Chemistry 2013

© Harvard University
Martin Karplus

Photo: © S. Fisch
Michael Levitt

Photo: Wikimedia Commons
Arieh Warshel

The Nobel Prize in Chemistry 2013 was awarded jointly to Martin Karplus, Michael Levitt and Arieh Warshel *"for the development of multiscale models for complex chemical systems"*.

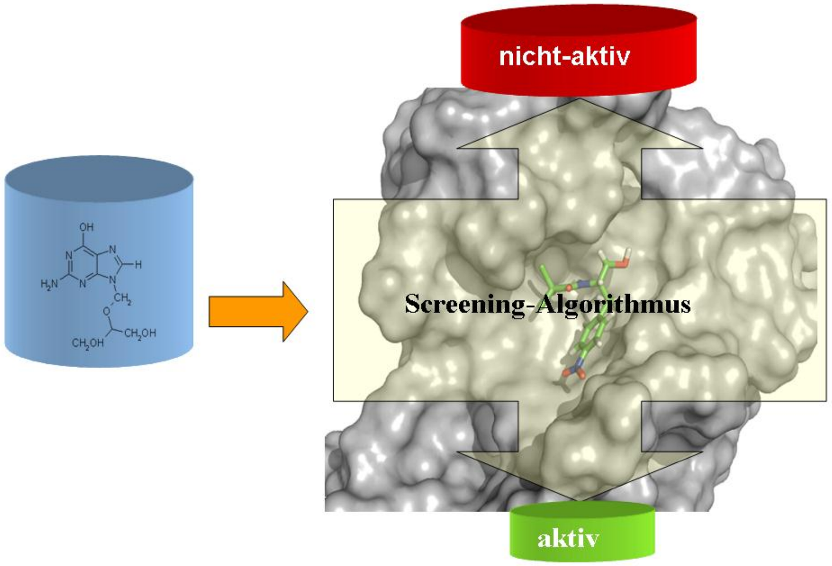
Methods for ligand database screening:

Screening in laboratory:

- Automatized,
- but expensive
- and time-consuming

Virtual Screening

•Search for leads
•As pre-stage for exp. tests



Definition of Virtual Screening

Use of *high-performance computing* to analyze large databases of chemical compounds in order to indetify possible drug candidates.

W.P. Walters, M.T. Stahl and M.A. Murcko, “Virtual Screening-An Overview”, Drug Discovery Today, 3, 160-178 (1998))

Databases of chemical compounds used

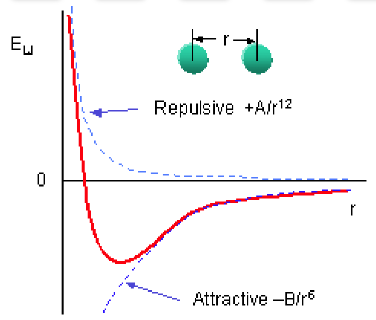
- ZINC database
    - free database of commercially-available compounds for virtual screening
    - contains over 13 million purchasable compounds in ready-to-dock, 3D formats
    - http://zinc.docking.org/, Irwin and Shoichet, J. Chem. Inf. Model. 2005;45(1):177-82
- In-house generated libraries
- Chemical synthesis of interesting compounds
- Experimental determination of activities

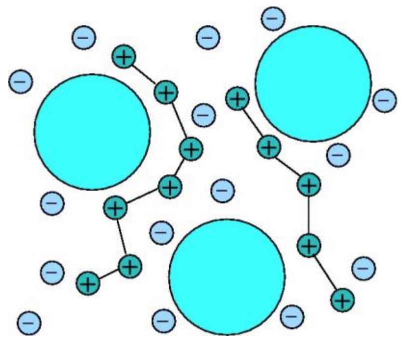
Scoring functions used in most VS methods (“biomolecular dwarfs”)

NON-BONDED INTERACTIONS

Van der Waals (VDW) + Electrostatics (ES)

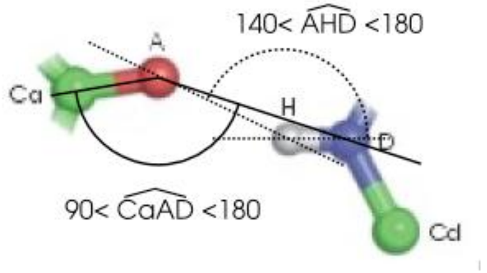
$$\sum_{protein} \sum_{lig, flSC} \left(\frac{R_{ij}}{r_{ij}^{12}} - \frac{A_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{r_{ij}} \right)$$





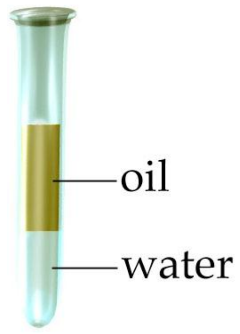
+ Hydrogen Bonds (HBOND)

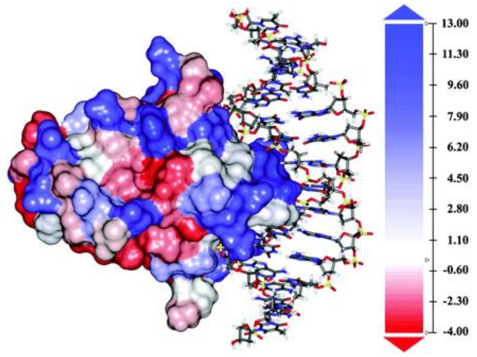
$$\sum_{h-bonds} \cos\Theta_{ij} \left(\frac{\tilde{R}_{ij}}{r_{ij}^{12}} - \frac{\tilde{A}_{ij}}{r_{ij}^{6}} \right)$$



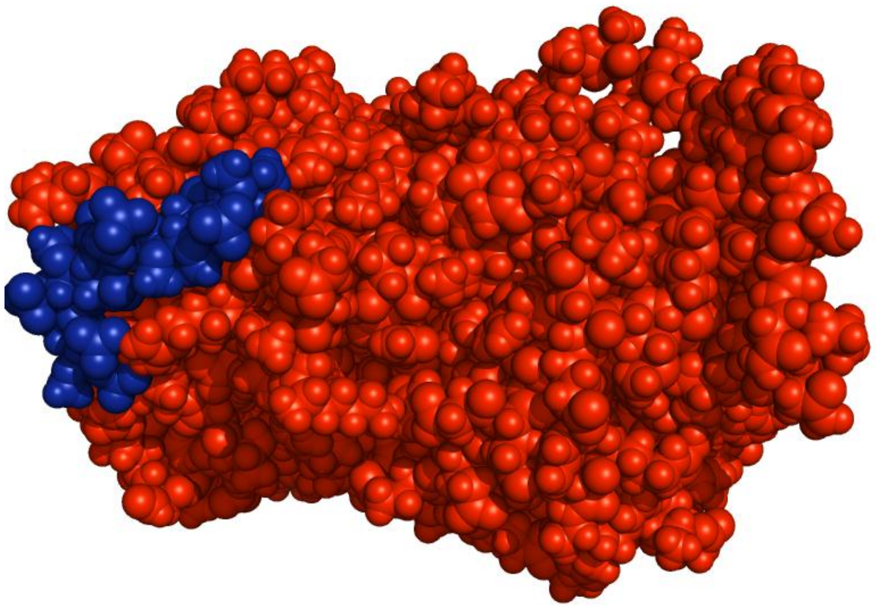
+ Solvation (SASA)

$$\sum_{SASA} \sigma_i A_i$$





CALCULATION OF PROTEIN-LIGAND INTERACTIONS IS EXPENSIVE!!!



- Virtual Screening of a database of one million of compounds in a 100 node cluster can take between one and six months or even more, depending on the accuracy of the VS method used

- In most Virtual Screening methods up to **80 %** of the time is spent in the calculation of Non-bonded interactions

Non-bonded interactions Kernels

For the description of the interaction between two molecules (protein and ligand) we need to calculate the interactions between each particle of the ligand with all particles of the protein

$$\boxed{V = \boxed{\sum_{i,j}\left(\frac{q_i q_j}{r_{ij}}\right.} + \left.\frac{R_{ij}}{r_{ij}^{12}} - \frac{A_{ij}}{r_{ij}^{6}}\right)}$$

FULL KERNEL

STUDY CASE: ELECTROSTATIC INTERACTIONS

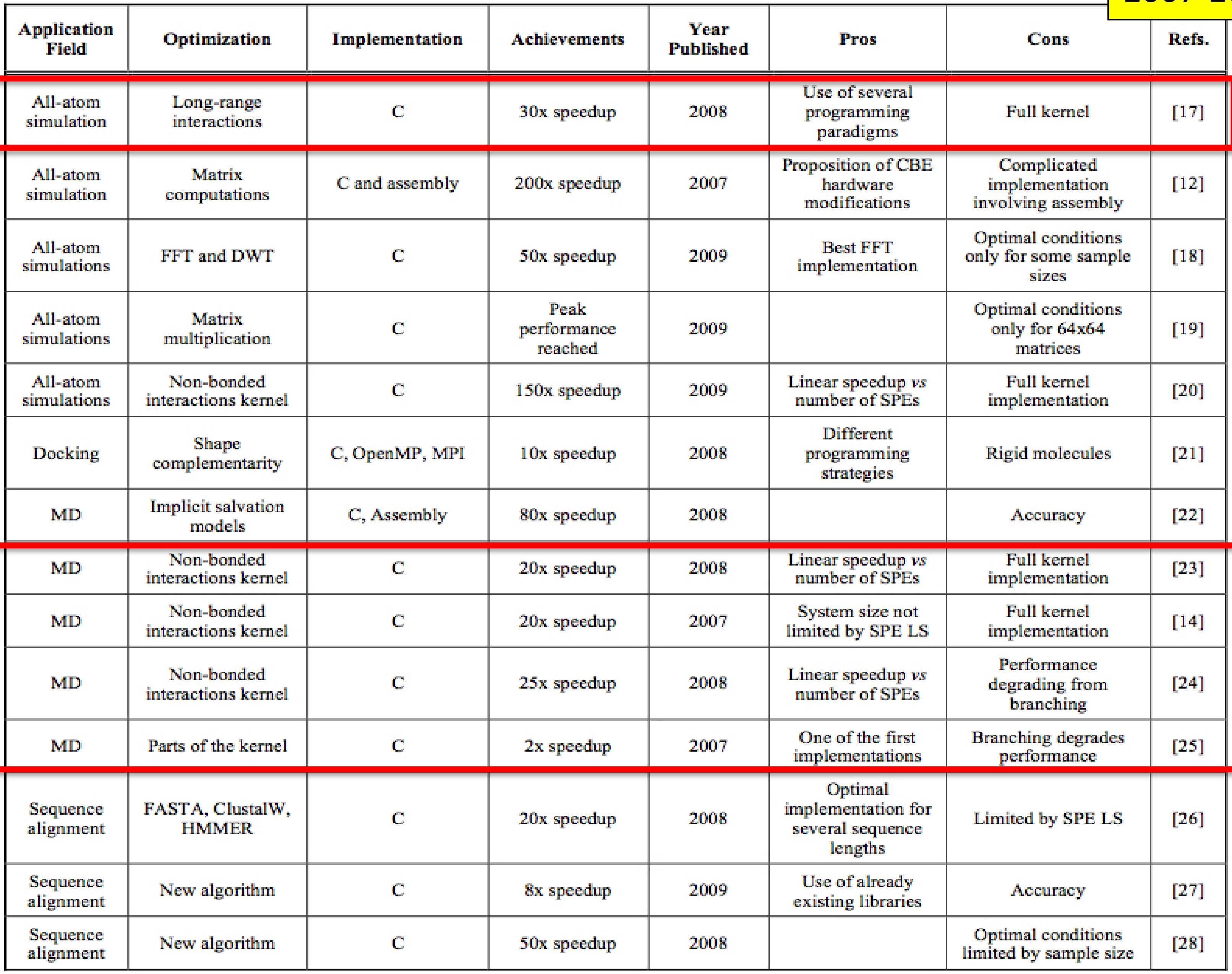
- Drug Discovery and Virtual Screening

- Non-bonded interactions kernel implementations
  - **Cell Broadband Engine**
  - GPU
  - Cluster; MPI/OpenMP
  - Intel Xeon Phi

- Conclusions and outlook

PREVIOUS RESULTS. CELL BROADBAND ENGINE

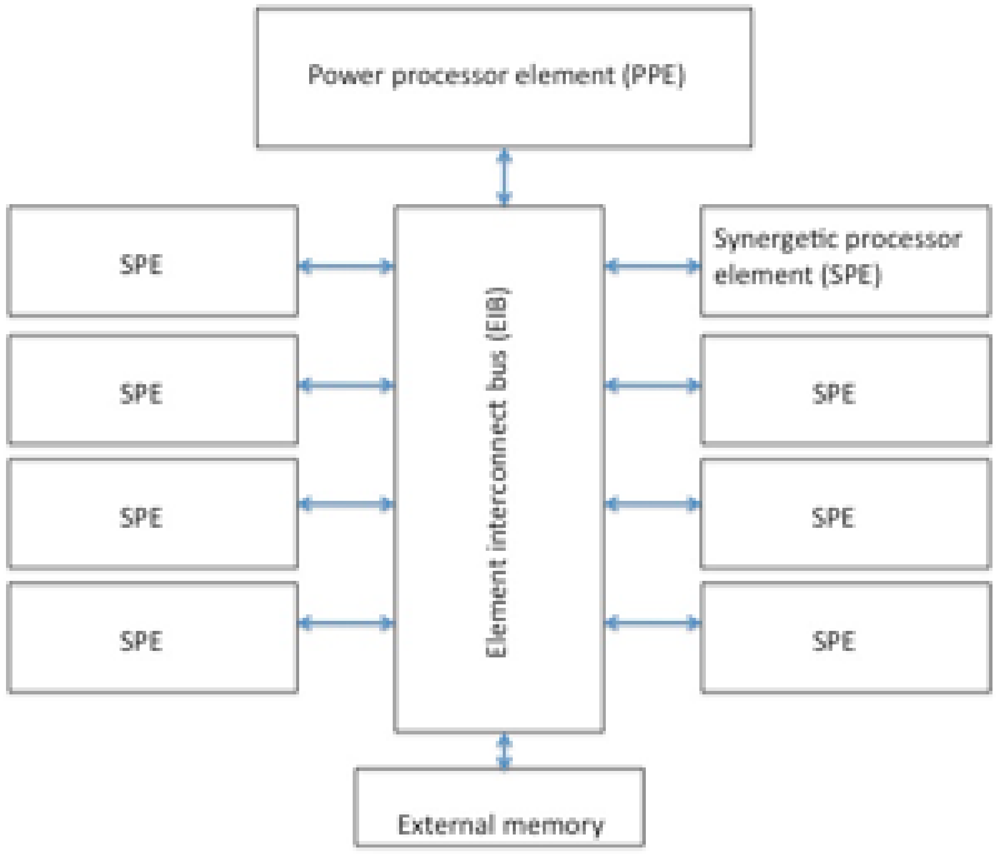
2007-2009



Application Field	Optimization	Implementation	Achievements	Year Published	Pros	Cons	Refs.
All-atom simulation	Long-range interactions	C	30x speedup	2008	Use of several programming paradigms	Full kernel	[17]
All-atom simulation	Matrix computations	C and assembly	200x speedup	2007	Proposition of CBE hardware modifications	Complicated implementation involving assembly	[12]
All-atom simulations	FFT and DWT	C	50x speedup	2009	Best FFT implementation	Optimal conditions only for some sample sizes	[18]
All-atom simulations	Matrix multiplication	C	Peak performance reached	2009		Optimal conditions only for 64x64 matrices	[19]
All-atom simulations	Non-bonded interactions kernel	C	150x speedup	2009	Linear speedup *vs* number of SPEs	Full kernel implementation	[20]
Docking	Shape complementarity	C, OpenMP, MPI	10x speedup	2008	Different programming strategies	Rigid molecules	[21]
MD	Implicit salvation models	C, Assembly	80x speedup	2008		Accuracy	[22]
MD	Non-bonded interactions kernel	C	20x speedup	2008	Linear speedup *vs* number of SPEs	Full kernel implementation	[23]
MD	Non-bonded interactions kernel	C	20x speedup	2007	System size not limited by SPE LS	Full kernel implementation	[14]
MD	Non-bonded interactions kernel	C	25x speedup	2008	Linear speedup *vs* number of SPEs	Performance degrading from branching	[24]
MD	Parts of the kernel	C	2x speedup	2007	One of the first implementations	Branching degrades performance	[25]
Sequence alignment	FASTA, ClustalW, HMMER	C	20x speedup	2008	Optimal implementation for several sequence lengths	Limited by SPE LS	[26]
Sequence alignment	New algorithm	C	8x speedup	2009	Use of already existing libraries	Accuracy	[27]
Sequence alignment	New algorithm	C	50x speedup	2008		Optimal conditions limited by sample size	[28]

Pérez-Sánchez and Wenzel. Optimization methods for virtual screening on novel computational architectures. Curr Comput Aided Drug Des (2011) vol. 7 (1) pp. 44-52

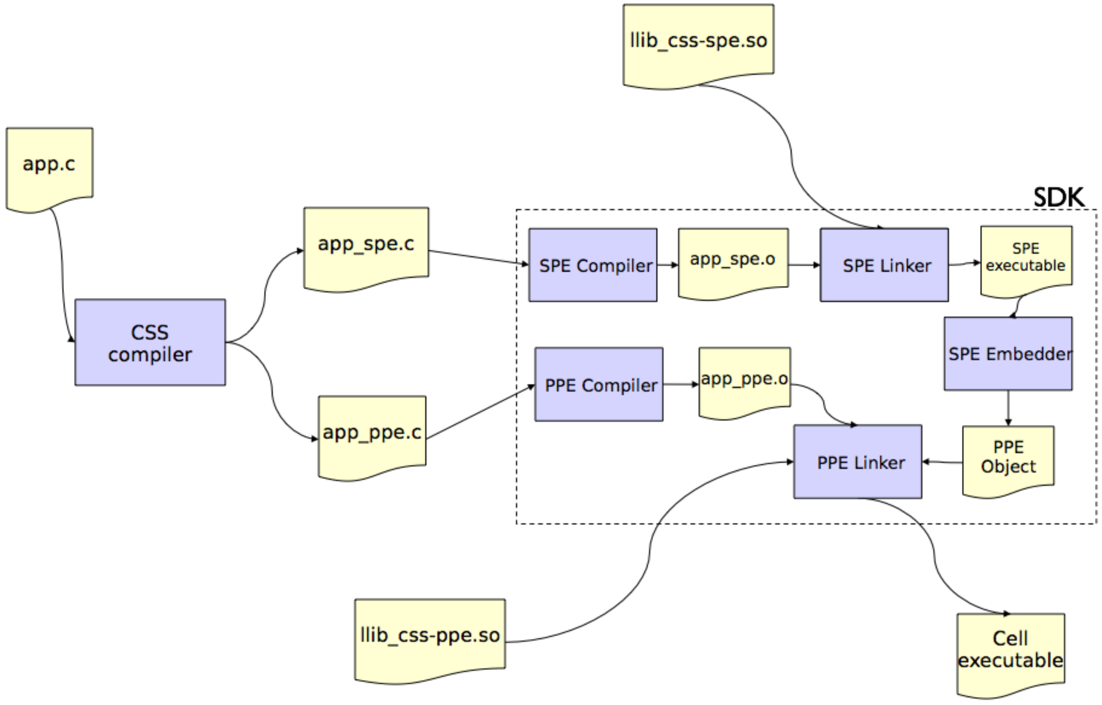
<u>Cell Broadband Engine (CBE)</u>


Power processor element (PPE)
SPE
SPE
SPE
SPE
Element interconnect bus (EIB)
Synergetic processor element (SPE)
SPE
SPE
SPE
External memory

- 250 GFLOPS theoretical performance single float
- approx 1/10 double prec
- 256KB Local Storage per SPE
- Vector operators
- Branching limitation

Cell Superscalar (CSS)

- metacompiler for **C** and Fortran. Linux
- no separate files for both PPU and SPU
- code annotation
- autoparallelization, loop unrolling
- developed at BSC and sponsored by IBM



Code vectorization

SPE vector operators

- similar to SIMD extensions for x86
- *our choice*: four 32-bit single-precision floating point numbers

```
vec_res.x = v1.x + v2.x;
vec_res.y = v1.y + v2.y;
vec_res.z = v1.z + v2.z;
vec_res.w = v1.w + v2.w;
```

```
vector float v1, v2, vec_res;
vec_res = spu_add(v1,v2);
```

Lots of operators:

spu_sub
spu_mul
spu_splats
spu_rsqrte
…………

PPE implementation, non vectorized code

$$V = \sum_{i,j}\left(\frac{q_i q_j}{d_{ij}}\right)$$

Docking example; Receptor - Ligand interaction

(NREC(j) and NLIG(i) particles)

$$\mathrm{Rec}_x - \mathrm{Lig}_x$$

...

$$(\mathrm{Rec}_x - \mathrm{Lig}_x)(\mathrm{Rec}_x - \mathrm{Lig}_x)$$

...

$$\frac{1}{d_{ij}} = \sqrt{\begin{matrix}(\mathrm{Rec}_x - \mathrm{Lig}_x)^2 + \\ (\mathrm{Rec}_y - \mathrm{Lig}_y)^2 + \\ (\mathrm{Rec}_z - \mathrm{Lig}_z)^2\end{matrix}}$$

$$\boxed{q_i q_j \frac{1}{d_{ij}}}$$

```
for(j=0;j<NREC;j++){
 for(i=0;i<NLIG;i++){
   difx=rec[j][0]-lig[i][0];
   dify=rec[j][1]-lig[i][1];
   difz=rec[j][2]-lig[i][2];
   mod2x=difx*difx;
   mod2y=dify*dify;
   mod2z=difz*difz;
   temp_sqrt=sqrt(mod2x+mod2y+mod2z);
   temp_div=1/temp_sqrt;
   S=S+ql[i]*temp_div*qr[j];
 }
}
```

loop is done *NLIG x NREC* times

SPE implementation, vectorized code

$$V = \sum_{i,j} \left(\frac{q_i q_j}{d_{ij}} \right)$$

$$\mathrm{Rec}_x - \mathrm{Lig}_x$$

calculated for 4 Lig atoms at the same time

...

$$(\mathrm{Rec}_x - \mathrm{Lig}_x)(\mathrm{Rec}_x - \mathrm{Lig}_x)$$

calculated for 4 Lig atoms at the same time

...

we get 1/d directly, for 4 Lig atoms

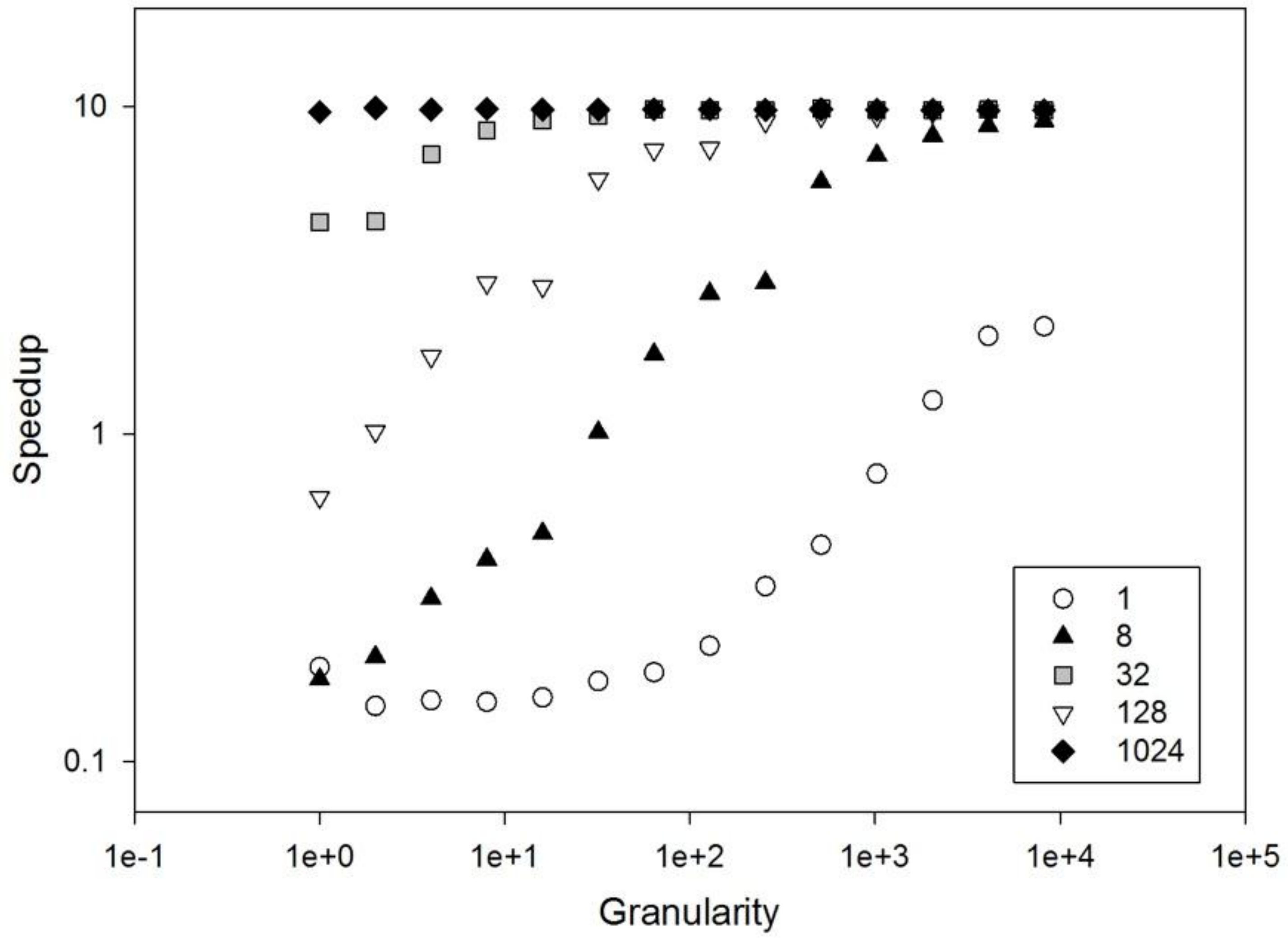
charges ...

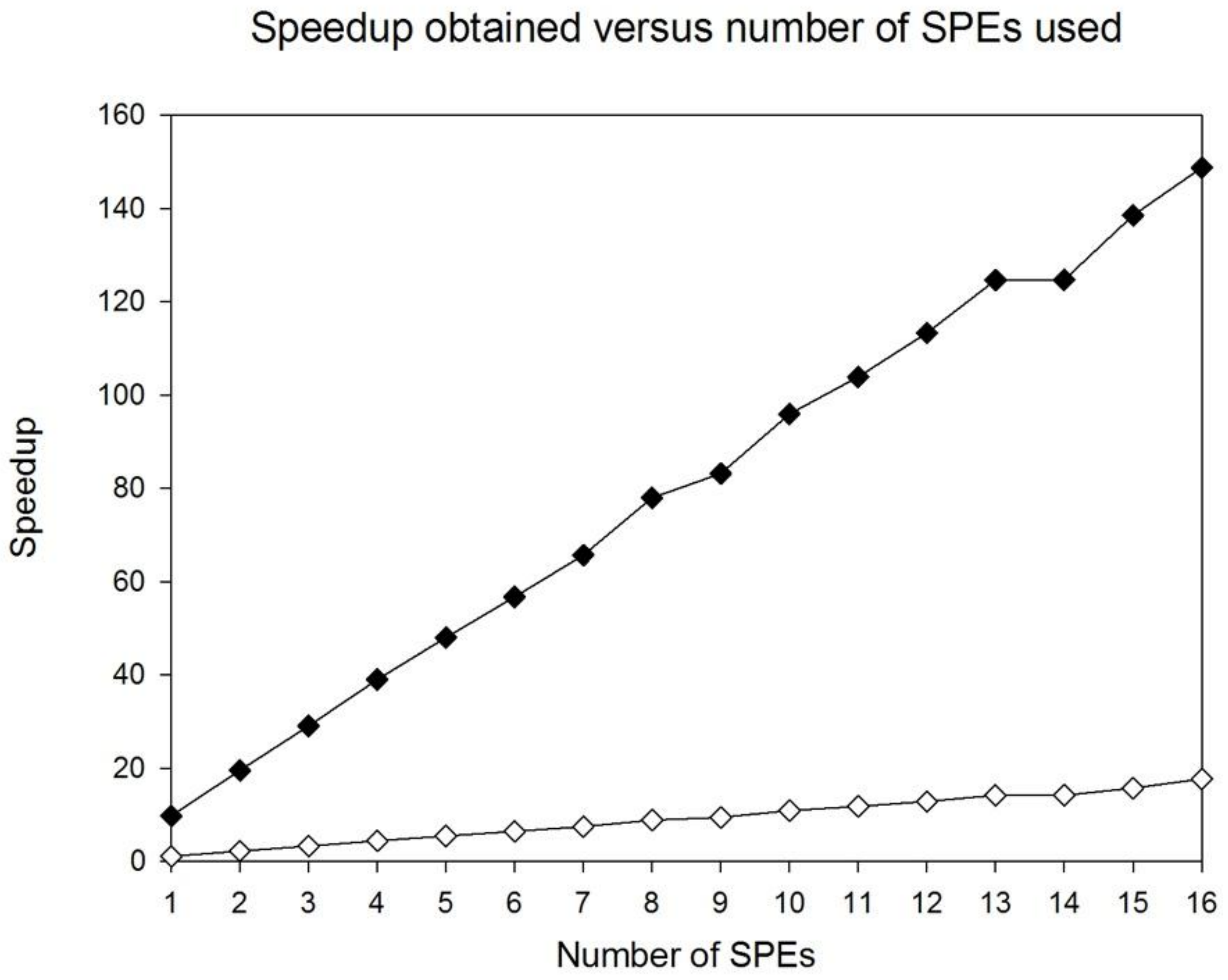
$$q_i q_j \frac{1}{d_{ij}} + q_i q_{j+1} \frac{1}{d_{i,j+1}} + q_i q_{j+2} \frac{1}{d_{i,j+2}} + q_i q_{j+3} \frac{1}{d_{i,j+3}}$$

```
for(j=0;j<NLIG/4;j++){ // "j" is related with nparticles of ligand / 4

        sum_inv_dist = spu_splats(zero);

        for(i=0;i<NREC;i++){ // "i" is related with nparticles of receptor

                temp_Rjx = spu_splats(Rjx[i]);
                temp_Rjy = spu_splats(Rjy[i]);
                temp_Rjz = spu_splats(Rjz[i]);
                temp_qr  = spu_splats(qr[i]);

                difx=spu_sub(Rix_v[j],temp_Rjx);
                dify=spu_sub(Riy_v[j],temp_Rjy);
                difz=spu_sub(Riz_v[j],temp_Rjz);

                prodx=spu_mul(difx,difx);
                prody=spu_mul(dify,dify);
                prodz=spu_mul(difz,difz);
                mod2=spu_add(spu_add(prodx,prody),prodz);
                inv_dist=spu_rsqrte(mod2);
                q_inv_dist=spu_mul(inv_dist,temp_qr);
                sum_inv_dist=spu_add(sum_inv_dist,q_inv_dist);

        }

        sum_inv_dist = spu_mul(ql_v[j],sum_inv_dist);
        sum_Ei=spu_add(sum_Ei,sum_inv_dist);
}
```

loop is done (*NLIG x NREC)/4* times

Speedup obtained versus granularity and system size





Speedup obtained versus number of SPEs used
Speedup
Number of SPEs
0
20
40
60
80
100
120
140
160
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

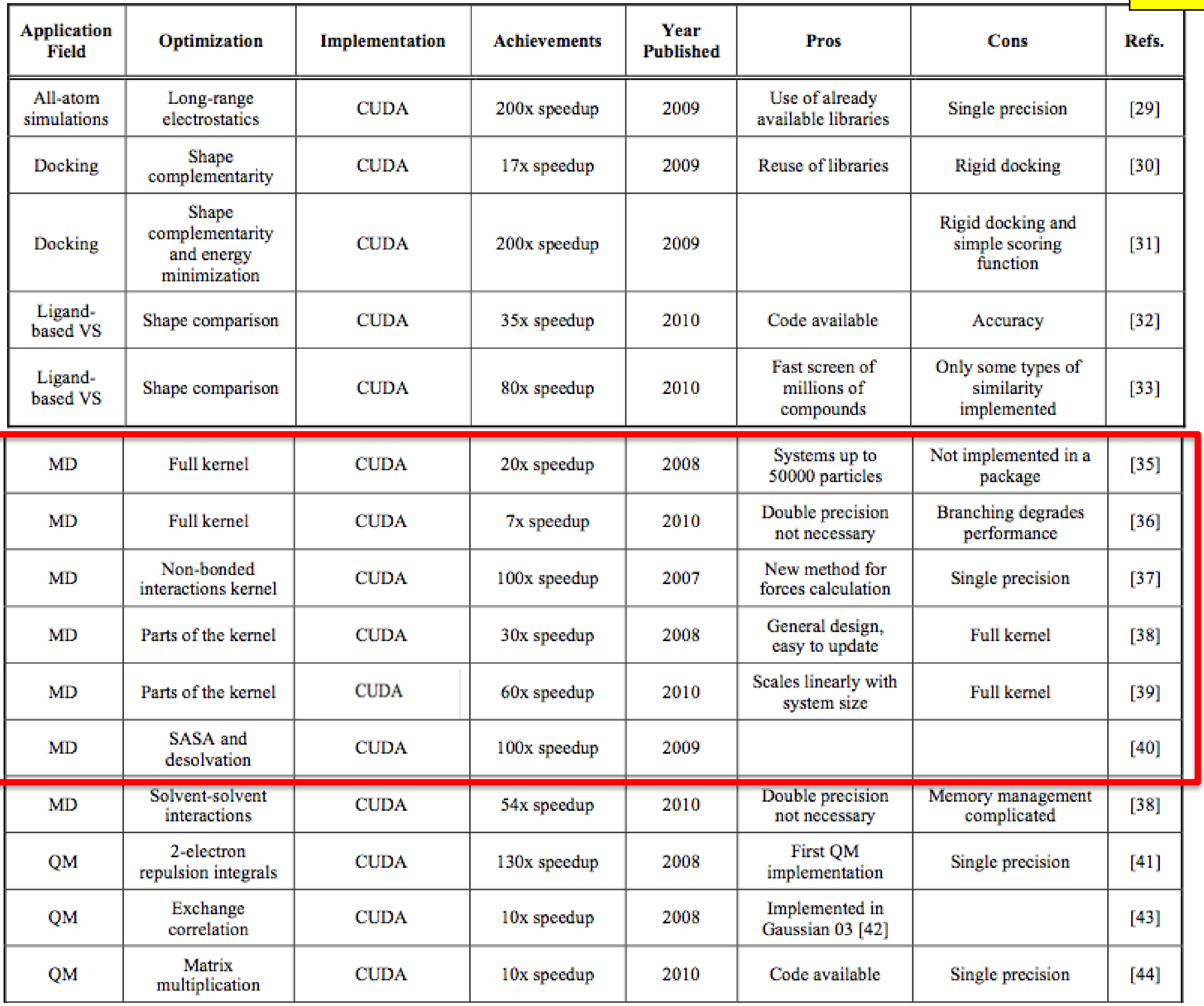
- Drug Discovery and Virtual Screening

- Non-bonded interactions kernel implementations
  - Cell Broadband Engine
  - **GPU**
  - Cluster; MPI/OpenMP
  - Intel Xeon Phi

- Conclusions and outlook

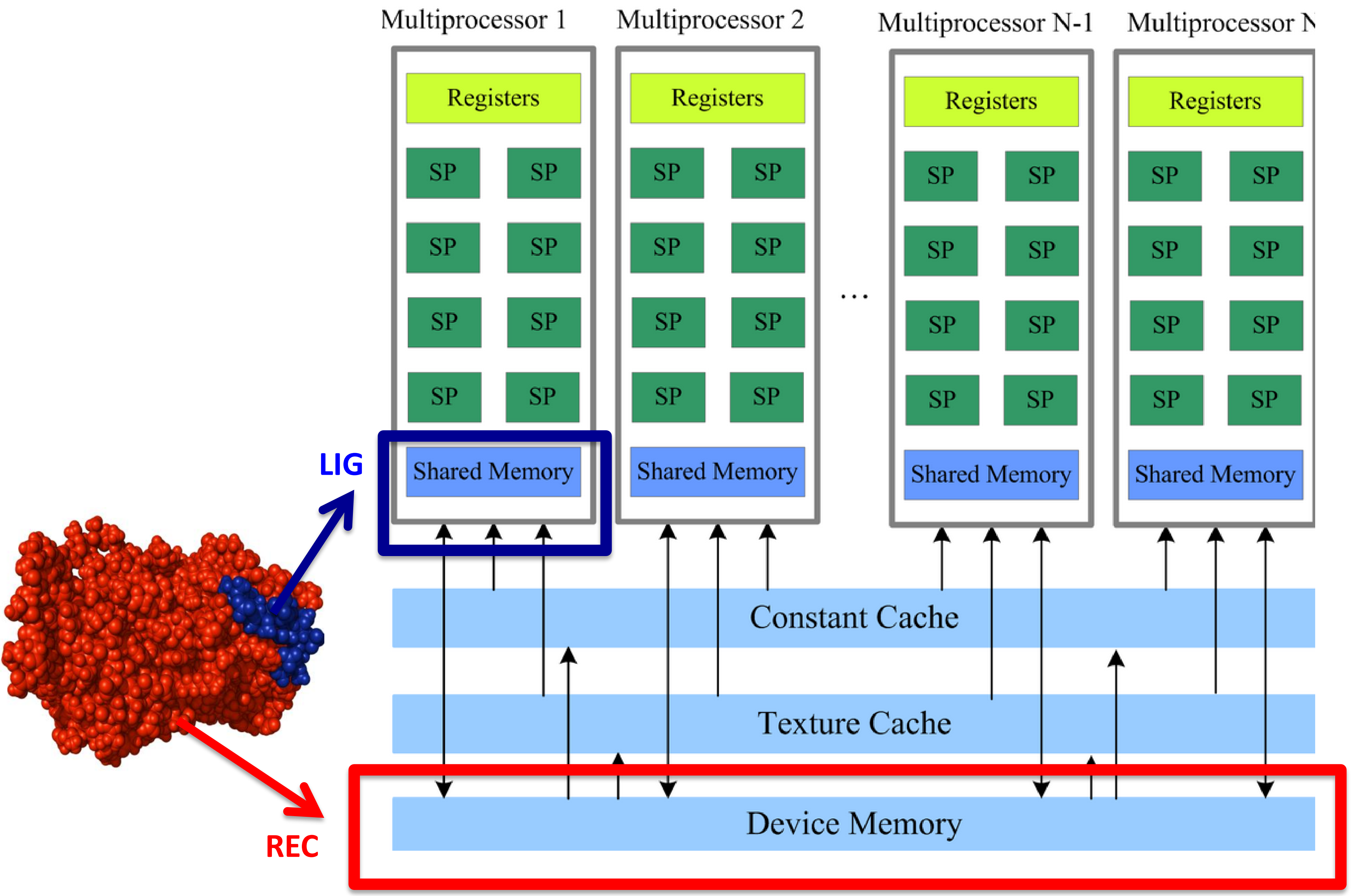
PREVIOUS RESULTS. GPUs

2007-2010

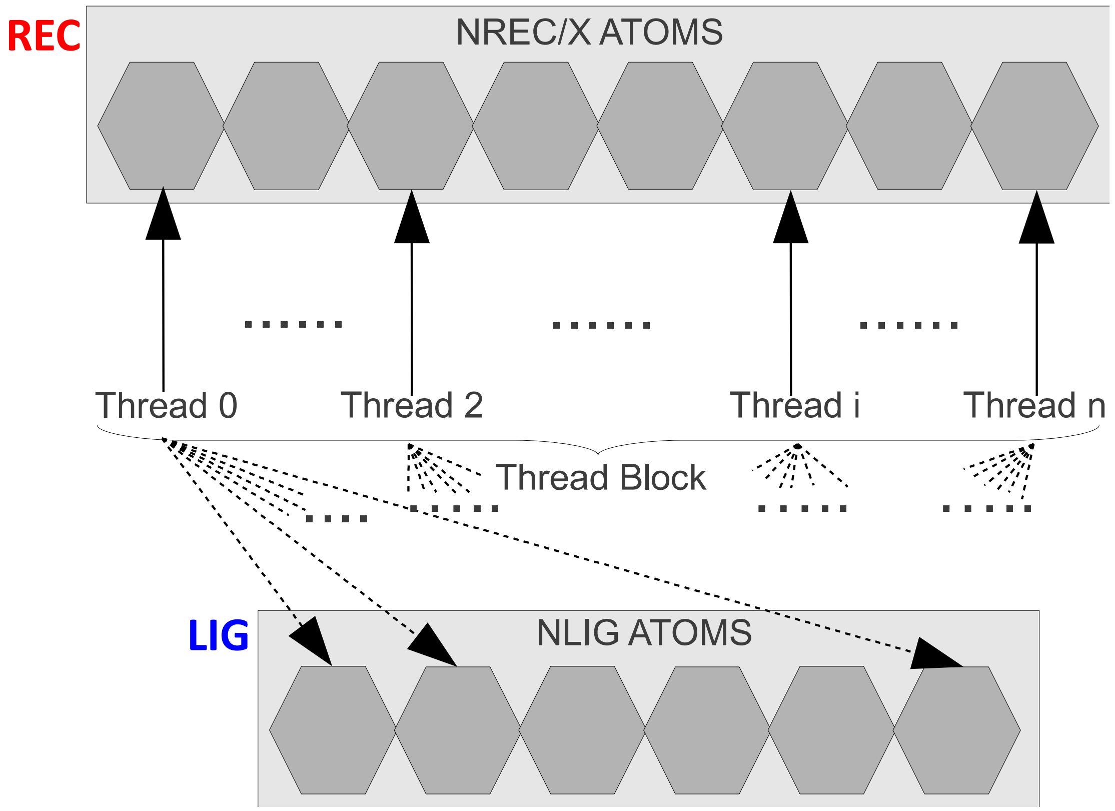


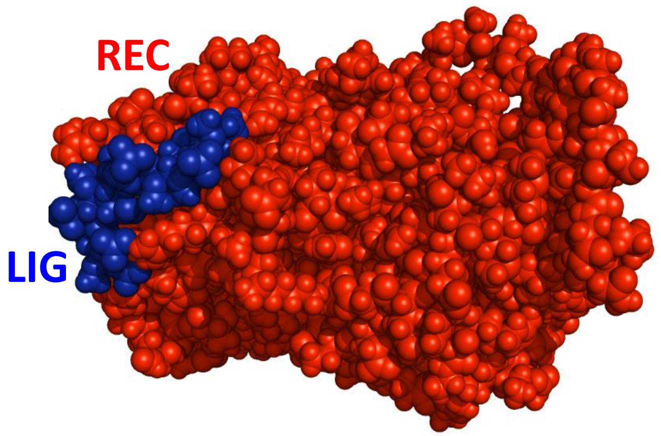
Application Field	Optimization	Implementation	Achievements	Year Published	Pros	Cons	Refs.
All-atom simulations	Long-range electrostatics	CUDA	200x speedup	2009	Use of already available libraries	Single precision	[29]
Docking	Shape complementarity	CUDA	17x speedup	2009	Reuse of libraries	Rigid docking	[30]
Docking	Shape complementarity and energy minimization	CUDA	200x speedup	2009		Rigid docking and simple scoring function	[31]
Ligand-based VS	Shape comparison	CUDA	35x speedup	2010	Code available	Accuracy	[32]
Ligand-based VS	Shape comparison	CUDA	80x speedup	2010	Fast screen of millions of compounds	Only some types of similarity implemented	[33]
MD	Full kernel	CUDA	20x speedup	2008	Systems up to 50000 particles	Not implemented in a package	[35]
MD	Full kernel	CUDA	7x speedup	2010	Double precision not necessary	Branching degrades performance	[36]
MD	Non-bonded interactions kernel	CUDA	100x speedup	2007	New method for forces calculation	Single precision	[37]
MD	Parts of the kernel	CUDA	30x speedup	2008	General design, easy to update	Full kernel	[38]
MD	Parts of the kernel	CUDA	60x speedup	2010	Scales linearly with system size	Full kernel	[39]
MD	SASA and desolvation	CUDA	100x speedup	2009			[40]
MD	Solvent-solvent interactions	CUDA	54x speedup	2010	Double precision not necessary	Memory management complicated	[38]
QM	2-electron repulsion integrals	CUDA	130x speedup	2008	First QM implementation	Single precision	[41]
QM	Exchange correlation	CUDA	10x speedup	2008	Implemented in Gaussian 03 [42]		[43]
QM	Matrix multiplication	CUDA	10x speedup	2010	Code available	Single precision	[44]

Pérez-Sánchez and Wenzel. Optimization methods for virtual screening on novel computational architectures. Curr Comput Aided Drug Des (2011) vol. 7 (1) pp. 44-52


Multiprocessor 1
Multiprocessor 2
Multiprocessor N-1
Multiprocessor N
Registers
SP
Shared Memory
...
LIG
Constant Cache
Texture Cache
Device Memory
REC

GPU IMPLEMENTATION

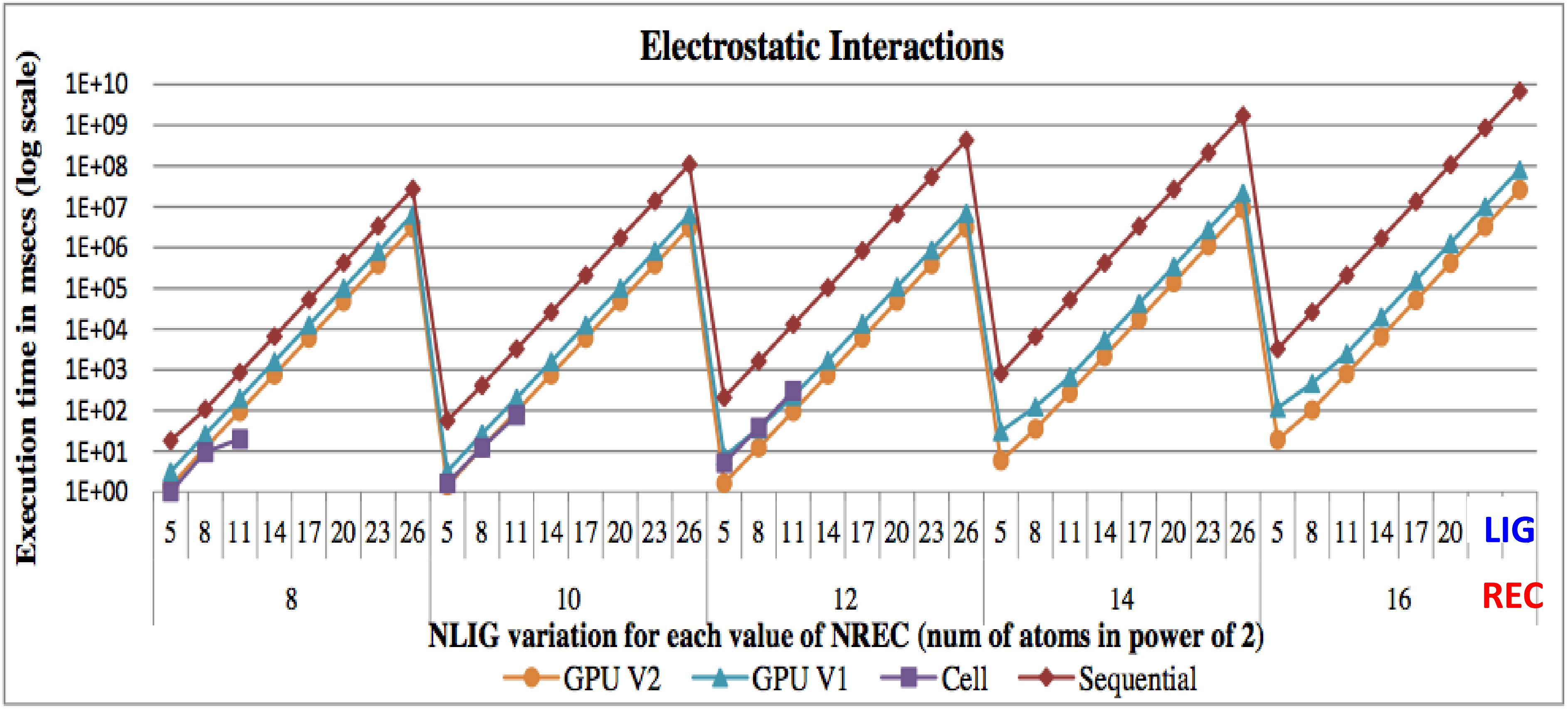




CUDA design for *X* thread blocks (with *X*=1) with *n* threads layout

- As many thread blocks as the number of *nrec* atoms divided by the number of threads within a block, this number is a configuration parameter of our application
- As many threads as *nrec* atoms, each thread computes the energy calculations with the entire ligand data.
- **We group atoms of the ligand molecule in tiles**, and thus threads can collaborate in order to bring that information to the shared memory

GPU IMPLEMENTATION



- CUDA 4.0 and NVIDIA Tesla C2050; max speedup around 213x
- speedup factor between GPU and CPU increases with *nrec* or/*and nlig*; number of thread blocks running in parallel is higher; GPU resources are fully used. However, it remains flat for a configuration greater than 256 threads per block.

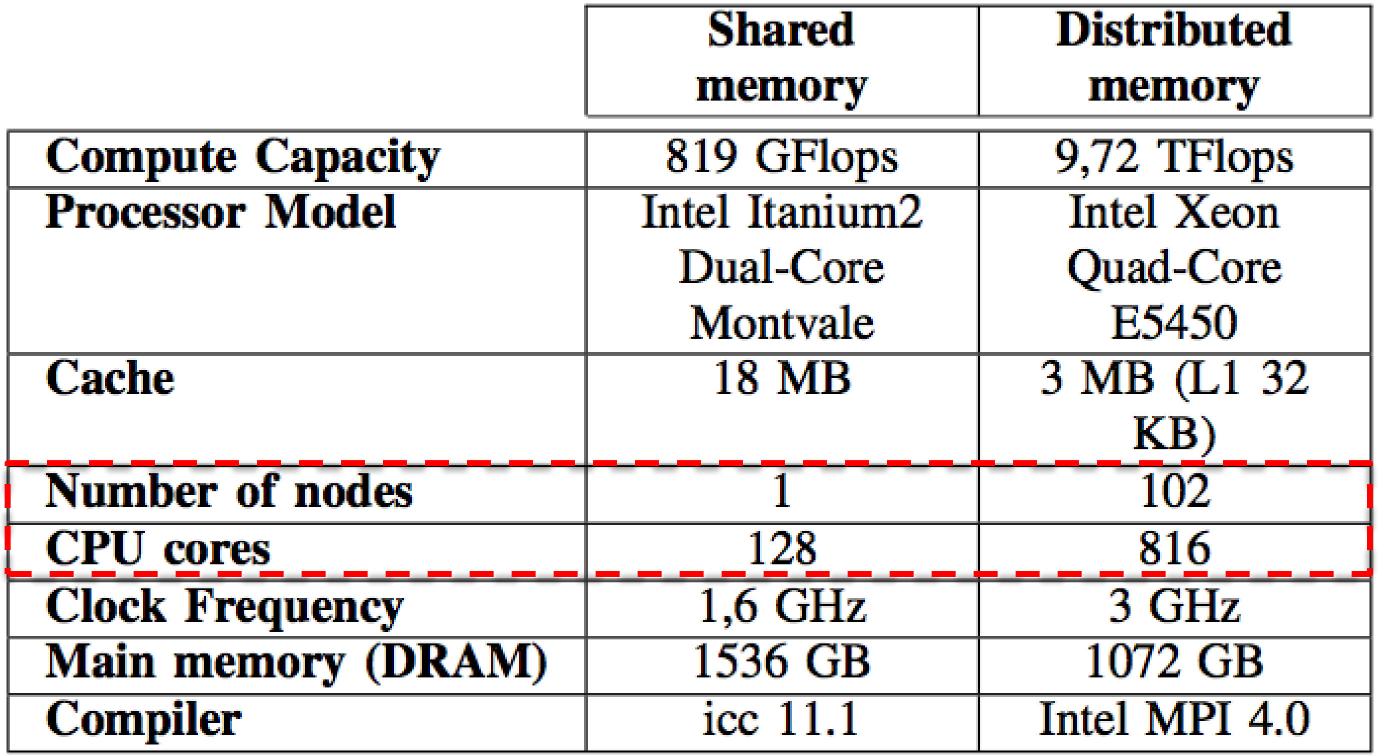
- Drug Discovery and Virtual Screening

- Non-bonded interactions kernel implementations
  - Cell Broadband Engine
  - GPU
  - **Cluster; MPI/OpenMP**
  - Intel Xeon Phi

- Conclusions and outlook

DISTRIBUTED AND SHARED MEMORY IMPLEMENTATIONS

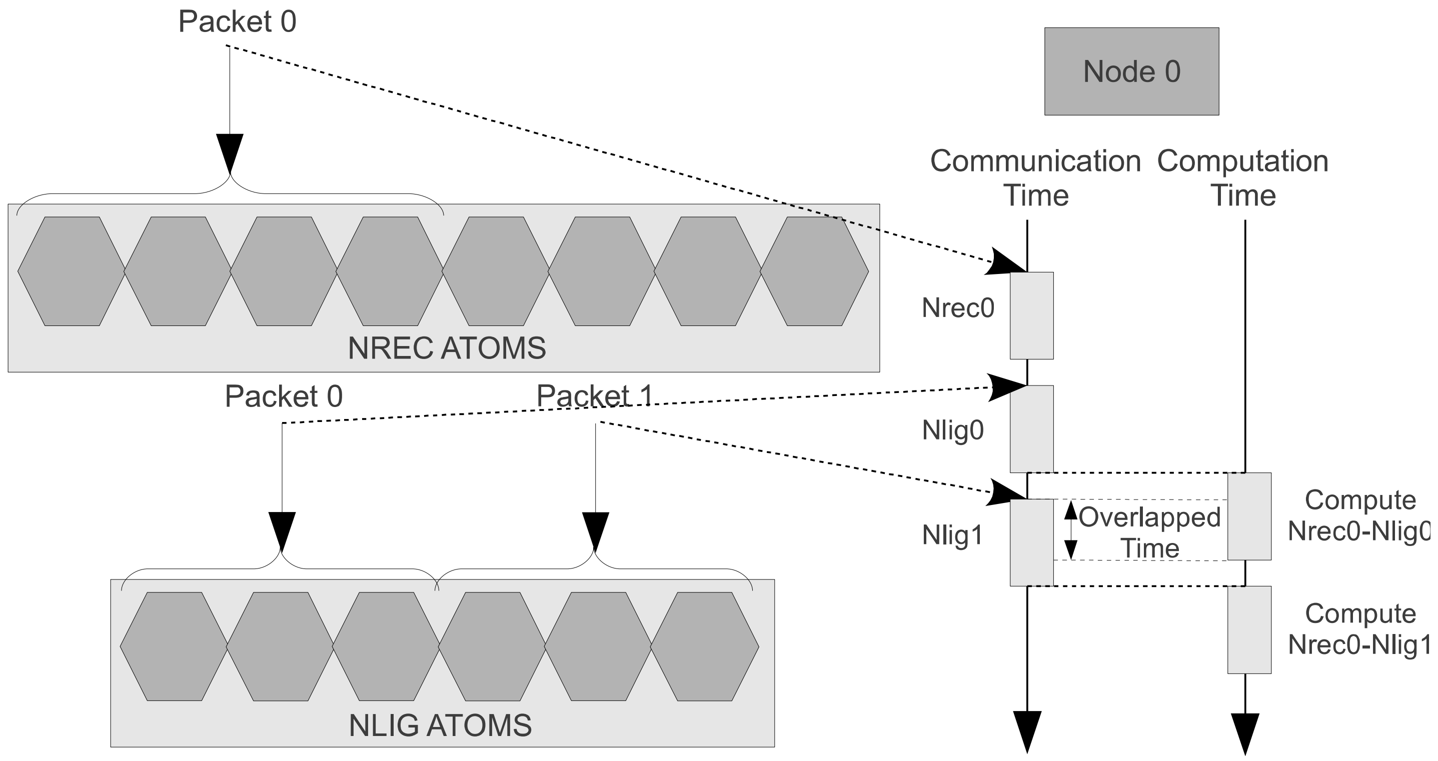
SUMMARY OF HARDWARE AND SOFTWARE FEATURES FOR THE PLATFORM USED DURING OUR EXPERIMENTAL SURVEY.



	Shared memory	**Distributed memory**
Compute Capacity	819 GFlops	9,72 TFlops
Processor Model	Intel Itanium2 Dual-Core Montvale	Intel Xeon Quad-Core E5450
Cache	18 MB	3 MB (L1 32 KB)
Number of nodes	1	102
CPU cores	128	816
Clock Frequency	1,6 GHz	3 GHz
Main memory (DRAM)	1536 GB	1072 GB
Compiler	icc 11.1	Intel MPI 4.0

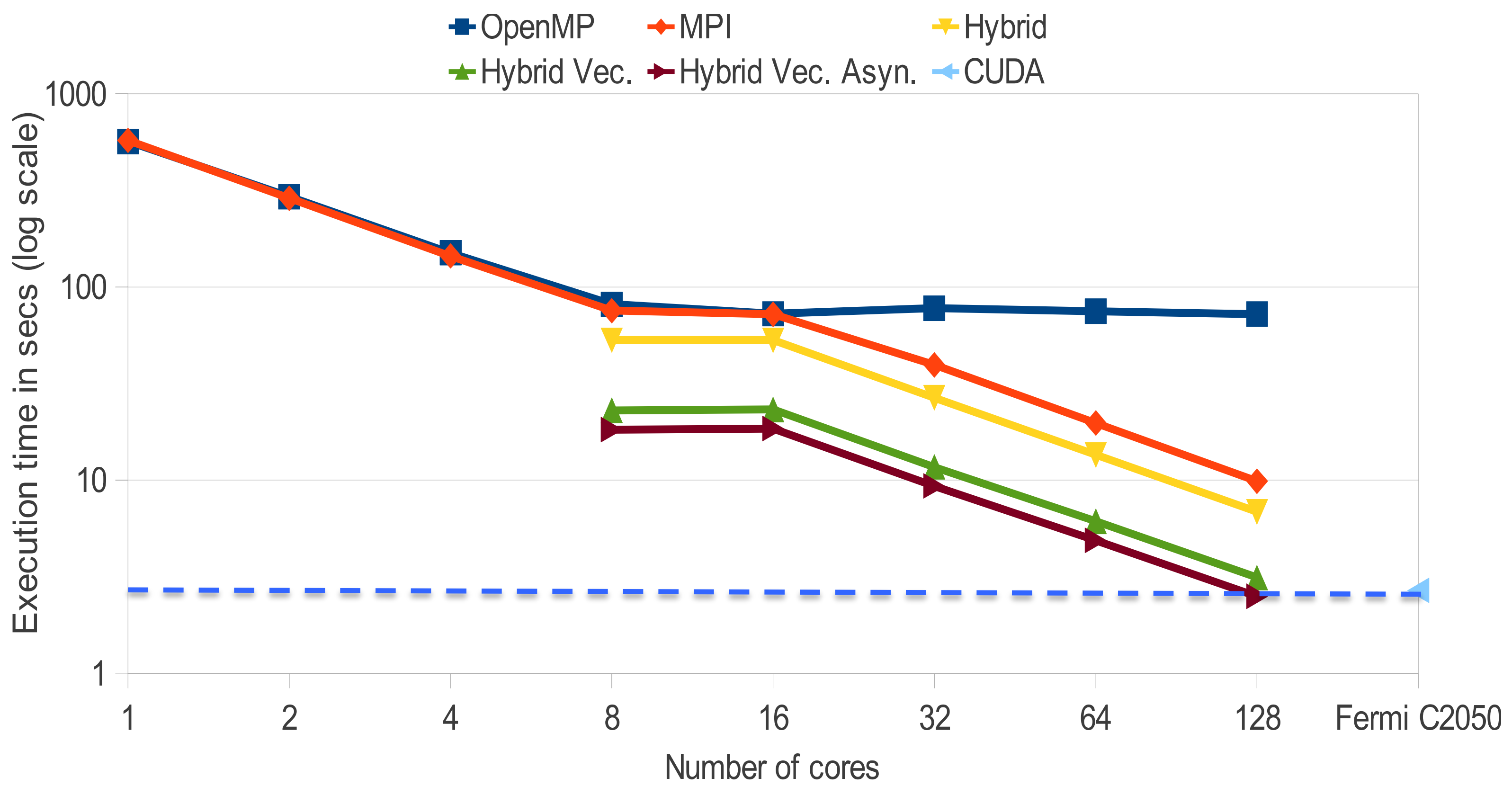
Centro de Supercomputación de la Fundación Parque Científico de Murcia

HYBRID OPENMP-MPI IMPLEMENTATION



- Communication and computation can be overlapped by asynchronous send/receive instructions
  - Data sent with MPI_Isend and MPI_Irecv
  - As soon as a **nlig** packet is received by a node, processors start computations while waiting for further data
- Code is also vectorized
  - x86 SSE instructions set
  - nlig info is copied four times into 128 bytes vectors

PERFORMANCE COMPARISON



- Performance: Supercomputing Center (SC) similar to GPU for Virtual Screening kernels
- Price: SC (M€) >>> GPU (K€) !!! (do you want to spare thousands of euros???)
- Power consumption: SC >>> GPU !!! (do you want to be green???)

YOU ARE WASTING YOUR TIME AND MONEY !!! INVEST YOUR SC BUDGET IN GPUs !!!

- Drug Discovery and Virtual Screening

- Non-bonded interactions kernel implementations
  - Cell Broadband Engine
  - GPU
  - Cluster; MPI/OpenMP
  - **Intel Xeon Phi**

- Conclusions and outlook

NVidia GPU vs. Intel MIC

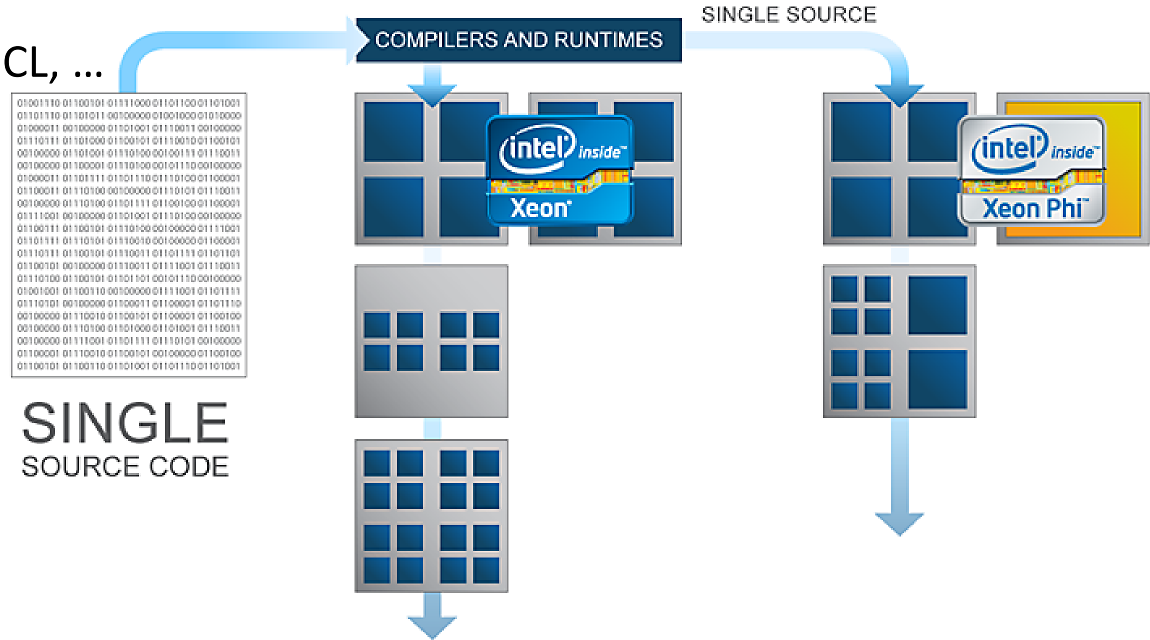
Hardware summary:

	Intel MIC (*Knights Corner*) → Nov 2012	**NVidia GPU (*Kepler*) → 2012/2013**
processors	~ 62 Pentium x86 cores	14 streaming multiprocessors (SMX)
per-processor concurrency	4 hyperthreads x 8 (512 bit) SIMD units	192 CUDA cores (SIMT)
total nominal concurrency	1984 = 62x4x8	2688 = 14x192
performance (DP)	~ 1 TFlops	~ 1 TFlops
memory	8 GB	6...12 GB
data transfer with host CPU	PCIe Gen2 (8 GB/s)	PCIe Gen3 (16 GB/s)
programming model/ software stack	OpenMP + SIMD vectorization Intel compilers, libraries, tools + proprietary offload directives	CUDA, OpenACC NVidia libraries, tools

Programming Xeon Phi

- Ease-of-use and programmability are selling points of XeonPhi, **what is the truth?**
- 2 running modes
  - offload mode - the main application is running on the host, and it only offloads selected (highly parallel, computationally intensive) work to the coprocessor
  - **native mode** - the application runs independently, on the Xeon Phi only, and can communicate with the main processor or other coprocessors through the system bus.

- Programming models
  - Pthreads, **OpenMP**, OpenCL, …
  - C/Fortran
  - MPI
- Libraries
  - MKL, …

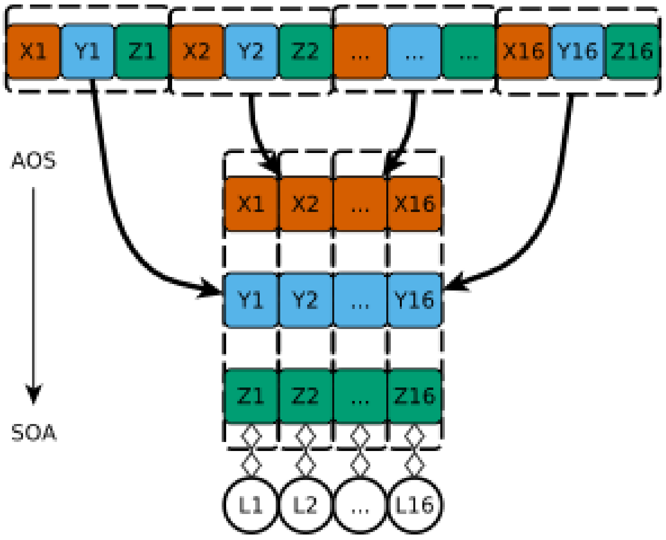


Intel Xeon Phi: Vectorization

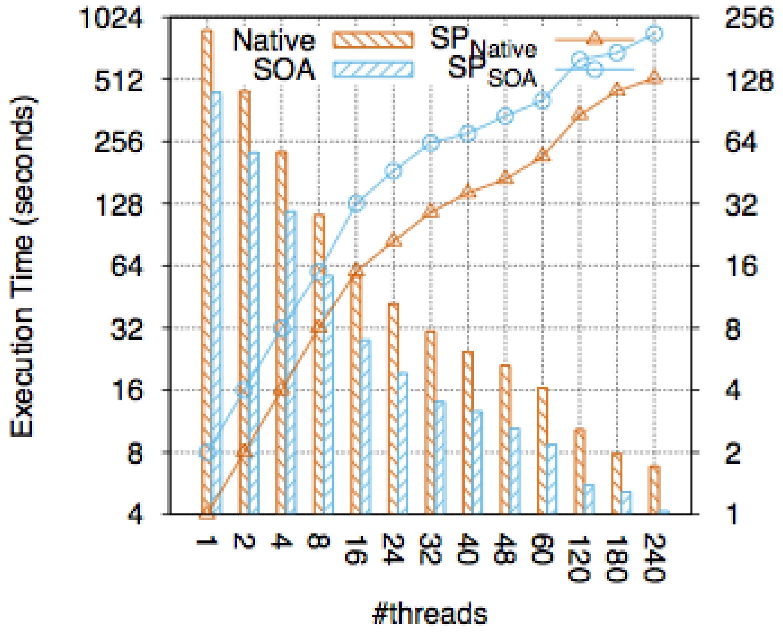
A straightforward way to parallelize our kernel using OpenMP is to add an **omp parallel** construct over the outer loop and rearrange data structures

$$V = \boxed{\sum_{i,j} \left(\frac{q_i q_j}{r_{ij}} \right.} \left. + \frac{R_{ij}}{r_{ij}^{12}} - \frac{A_{ij}}{r_{ij}^{6}} \right)$$

Native (Array of Structures, AOS)



(Structure of Arrays, SOA)



AOS; 128x

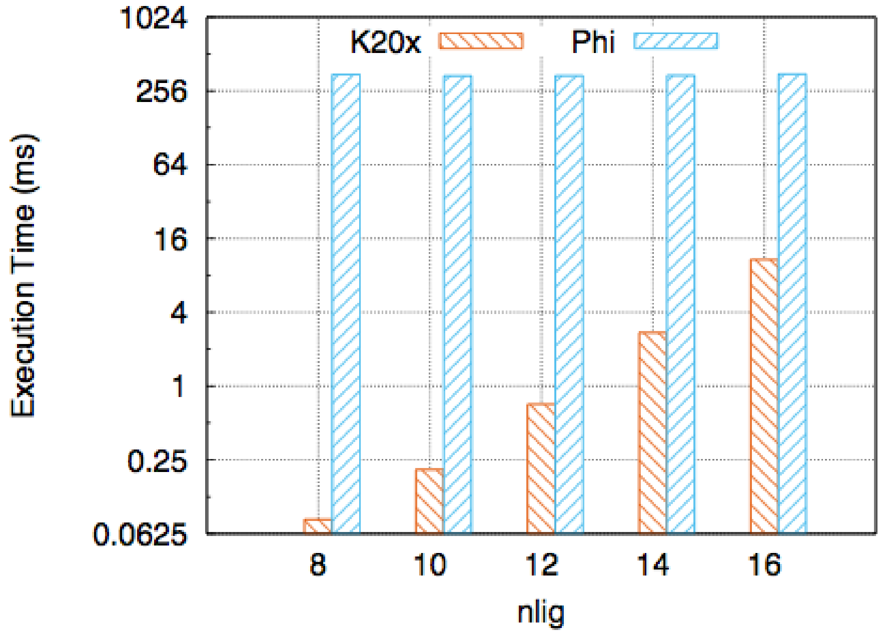
$\downarrow$

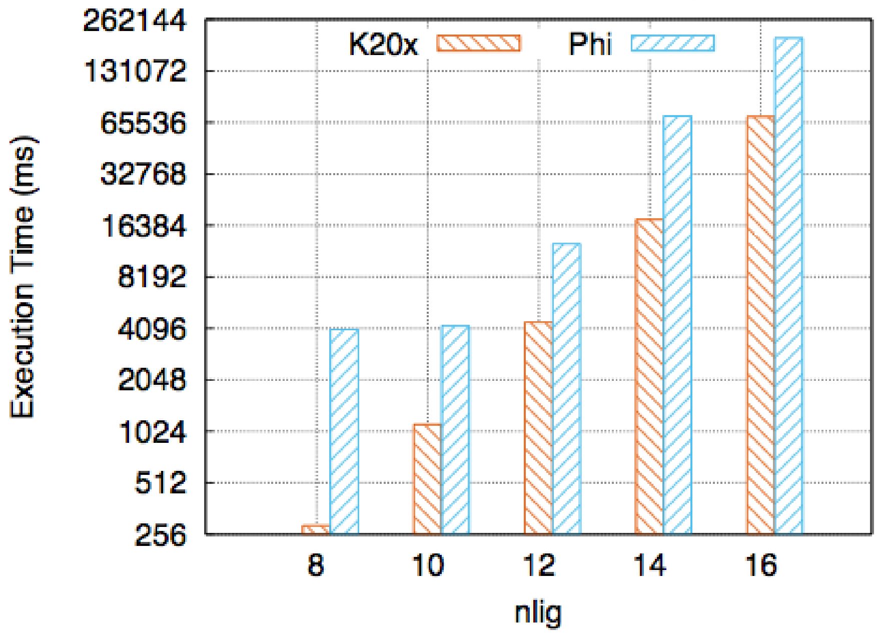
SOA; 256x

Performance: Phi and GPU

GPU/Phi gap reduction

~16x $\longrightarrow$ ~4x





2048 receptor particles $\xrightarrow{\textit{larger systems}}$ 6.7*10^7 receptor particles

- Single precision calculations for relatively small sized systems are more suitable for GPUs (K20x completely out- performs Xeon Phi)
- For large systems, they achieve a similar order of magnitude performance

- Drug Discovery and Virtual Screening

- Non-bonded interactions kernel implementations
  - Cell Broadband Engine
  - GPU
  - Cluster; MPI/OpenMP
  - Intel Xeon Phi

- **Conclusions and outlook**

Conclusions

- – **Porting legacy (sequential) code in OpenMP for Xeon Phi comes almost for free**. However, optimizing the outcome is relatively time-consuming, as a thorough understanding of the architectural features of the processor is mandatory.

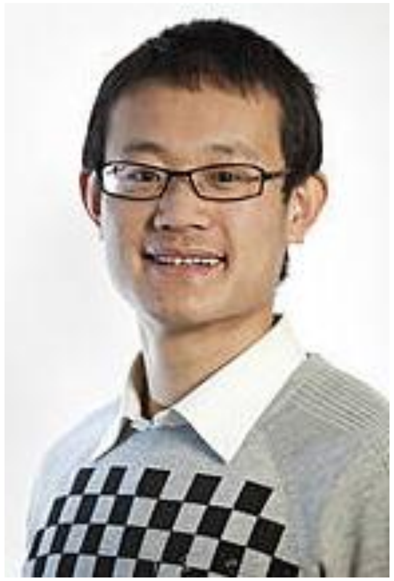
- – **On Xeon Phi, it is essential to select suitable data structures** (SOA instead of AOS, for caching) to enable the full utilization of the SIMD units. By comparison, GPUs like Nvidia K20x prefer the AOS-style data structures.

- – **Nvidia K20x significantly outperforms Intel Xeon Phi on Virtual Screening when using single-precision** floating-point data elements. We expect the performance for double precision computations to be much closer.

Outlook

- Evaluate the **double precision** computation for both the GPU and the Xeon Phi
- Aiming to use a unified programming model, we will evaluate an **OpenCL** solution for VS on both GPUs and Xeon Phi, thus evaluating the impacts of the chosen programming model on the overall performance of the application
- Extension to **other Virtual Screening Kernels** (Van der Waals, Hydrogen Bonds, etc)
- Characterize **Phi Power Consumption** in an heterogeneous computing environment

COLLABORATORS



Jianbin Fang, TUDelft

Ana Lucia Varbanescu, University of Amsterdam

BIOINFORMATICS AND HIGH PERFORMANCE COMPUTING RESEARCH GROUP (UCAM, Murcia, Spain)

http://bio-hpc.eu

- 1 Full time research associate
- 5 Full time associate professors
- 4 PhD students
- Collaboration with more than 20 international research groups

Acknowledgments

UNIVERSIDAD CATÓLICA SAN ANTONIO
IN LIBERTATEM VOCATI
AÑO MCMXCVI

UCAM
UNIVERSIDAD CATÓLICA
SAN ANTONIO

CENTRO EXTREMEÑO DE TECNOLOGÍAS AVANZADAS
CETA Ciemat

PRACE

scbi uma

NVIDIA
CUDA™ TEACHING CENTER

intel®

OMICS INTERNATIONAL

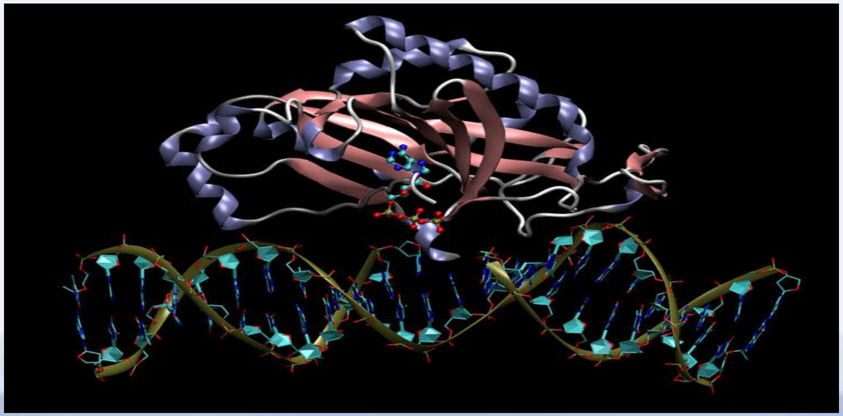
www.omicsonline.org

OMICS International (and its subsidiaries), is an Open Access publisher and international conference Organizer, which owns and operates peer-reviewed Clinical, Medical, Life Sciences, and Engineering & Technology journals and hosts scholarly conferences per year in the fields of clinical, medical, pharmaceutical, life sciences, business, engineering, and technology. Our journals have more than 3 million readers and our conferences bring together internationally renowned speakers and scientists to create exciting and memorable events, filled with lively interactive sessions and world-class exhibitions and poster presentations. Join us!

OMICS International is always open to constructive feedback. We pride ourselves on our commitment to serving the Open Access community and are always hard at work to become better at what we do. We invite your concerns, questions, even complaints. Contact us at contact.omics@omicsonline.org. We will get back to you in 24-48 hours. You may also call 1-800-216-6499 (USA Toll Free) or at +1-650-268-9744 and we will return your call in the same timeframe.

Drug Designing Open Access Related Journals

- Journal of Clinical Trials
- Journal of Pharmacovigilance
- Journal of Developing Drugs



Drug Designing Open Access Related Conferences

- http://www.conferenceseries.com/