Porting and Optimizing Molecular Docking Simulations on SX-Aurora Vector Engine

Leonardo Solis-Vasquez (TU Darmstadt)
Erich Focht (NEC Germany)
Andreas Koch (TU Darmstadt)

31st Workshop on Sustained Simulation Performance (virtual), March 16-19, 2021

Embedded Systems & Applications

Computer-Aided Drug Design

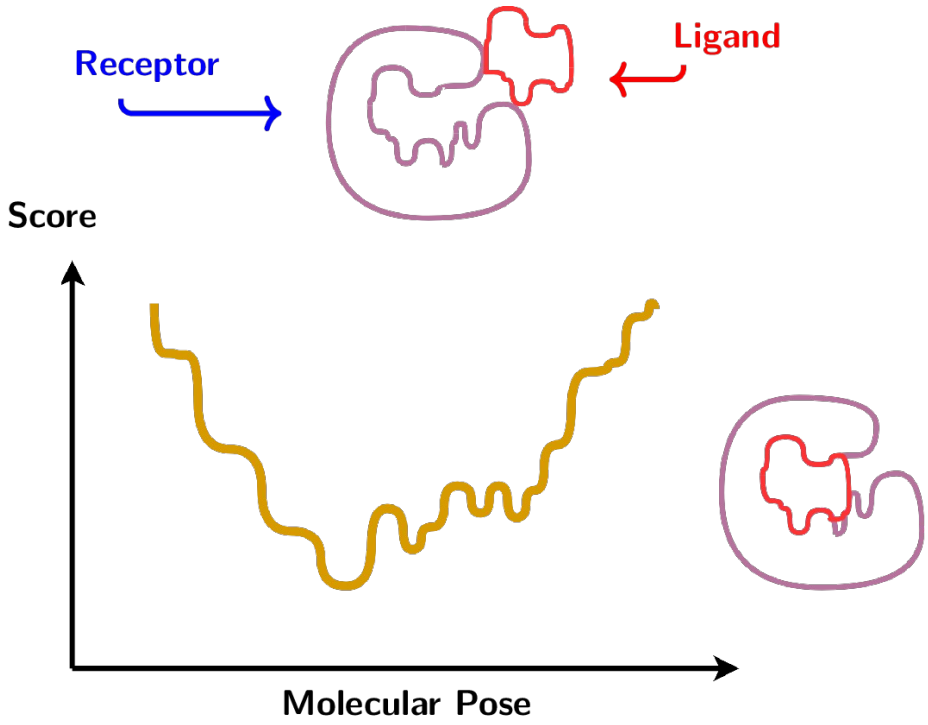
- Contributes fighting against diseases
  - AIDS, cancer, COVID
- Molecular docking simulations
  - Key methods in computer-aided drug design
  - Predict molecular interactions at *short distances*
    - Receptor: macromolecule
    - Ligand: small molecule ----> drug candidate
  - Benefits
    - Shorten the task of identifying drug candidates
    - Subsequent lab experiments can be performed on a narrowed list of promising ligands
    - *Reduces the overall need for costly and slow lab experiments*

NEC

Molecular Docking

- It aims to find poses of strong interaction

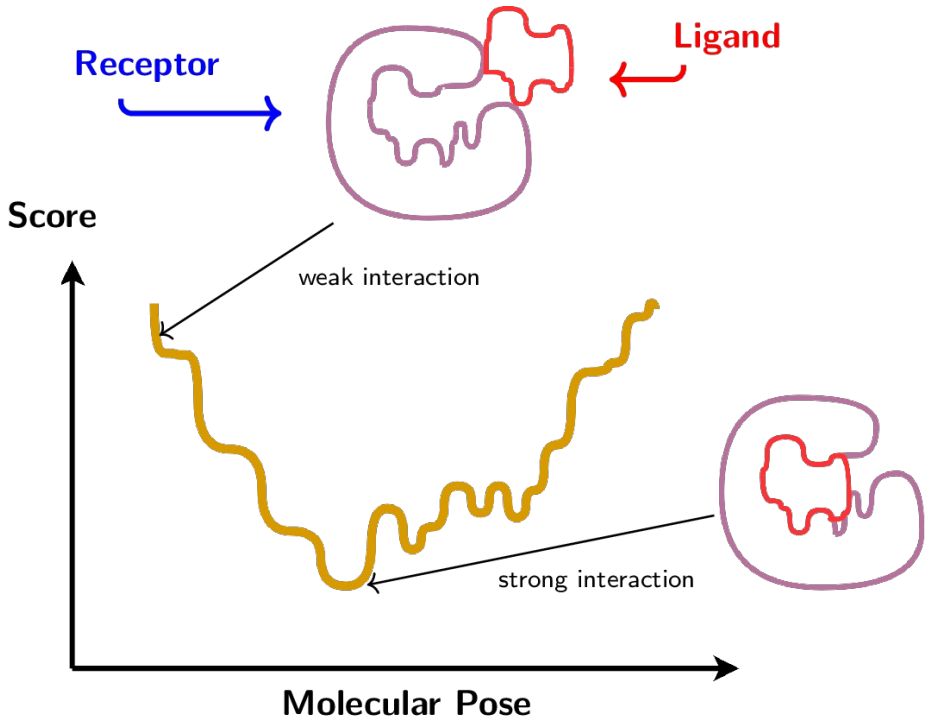
- Scoring function
  - Measures a pose strength
  - Computational expensive
    - $N_{atom}^{receptor} > 1000$
    - $N_{atom}^{ligand} < 100$



Molecular Docking

- Search methods
  - Finds an optimal pose
    - *Strong interaction*
  - Usually based on heuristic

- Representation
  - Encodes a pose in terms of e.g., translation, rotation, torsion

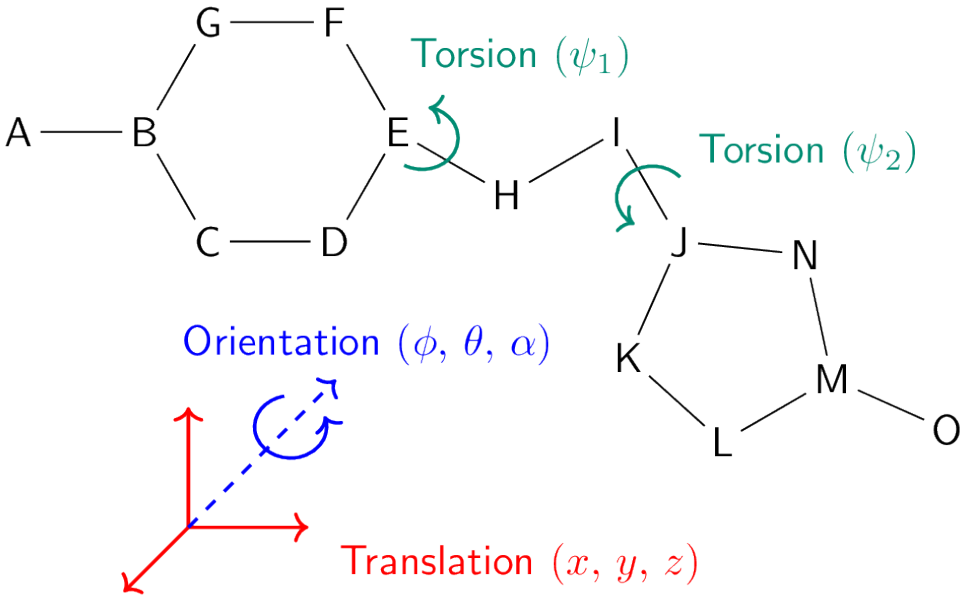


AutoDock

- A widely used software for molecular docking simulation
  - Developed by Scripps Research Institute (USA)
- Applicability
  - *FightAIDS@Home project*
  - *OpenPandemics: COVID-19*
- Receptor-ligand docking
- Lamarckian Genetic Algorithm
  - Hybridizes search methods
  - Performs compute-intensive score calculations

Encoding Ligand Poses

- Receptor
  - Treated as a rigid body
- Ligand poses are
  - Encoded as a set of variables
    - Translation
    - Orientation
    - Torsion
  - The solutions of the docking problem



$$\text{Pose}_{\text{Lig}} = \{x,\ y,\ z,\ \phi,\ \theta,\ \alpha,\ \psi_1,\ \psi_2,\ \ldots,\ \psi_{N_{\text{rot}}}\}$$

Mapping Docking into Genetic Evolution

- Pose ----> individual
  - An individual is
    - A member of a population
    - Represented by its genome

- Pose variables ----> genes

$$\text{Pose}_{\text{Lig}} = \{x,\ y,\ z,\ \ldots,\ \psi_{N_{\text{rot}}}\}$$

$$\text{Genome} = \{\text{gene}_1,\ \text{gene}_2,\ \text{gene}_3, \ldots,\ \text{gene}_M,\}$$

Genetic Algorithm (GA)

- New individuals are generated through genetic evolution

- Mimicking Darwinian evolution
  - Crossover
    - New individuals inherit genes from either parent
  - Mutation
    - Genes change by a random amount
  - Selection
    - Better-suited individuals reproduce

Lamarckian Genetic Algorithm (LGA)

- Hybrid search
  - LGA = Genetic Algorithm + Local Search

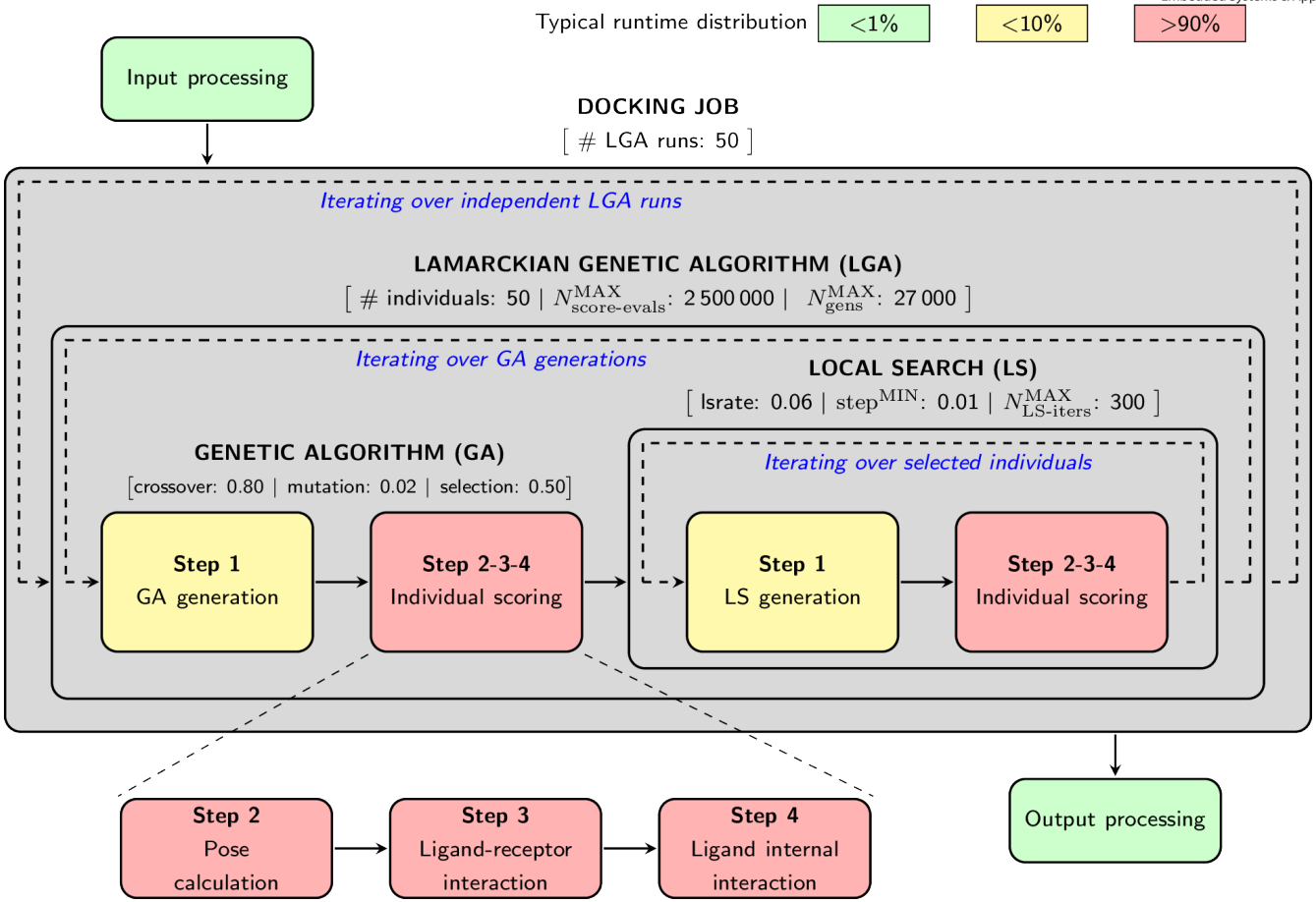
- Local Search
  - Score minimization
  - Genes experience change, which is
    - A constrained random amount
    - Adapted depending on score improvement
  - Self adaptive
    - Variable runtime

Scoring Function

- Binding energy (Kcal/mol)
  - Molecular mechanics
- Coefficients and Look-Up tables
  - W_{vdw} , W_{hb} , W_{el} , W_{ds} , W_{rot}
  - A, B, C, D, S, V , E, q
- Interatomic distance r_{ij}
  - Between atoms i and j
  - *Varies with every new pose*

$$SF = \sum_{i,j} \Bigg[\underbrace{W_{vdw}\left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}\right)}_{\text{Lennard-Jones}} + \underbrace{W_{hb}E(t)\left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}}\right)}_{\text{Hydrogen bonding}} + \underbrace{W_{el}\left(\frac{q_i q_j}{\epsilon(r_{ij})r_{ij}}\right)}_{\text{Coulomb's law}} + \underbrace{W_{ds}\left(S_i V_j + S_j V_i\right)e^{\frac{-r_{ij}^2}{2\sigma^2}}}_{\text{Desolvation}} \Bigg]$$

Functionality



Embedded Systems & Applications

Baseline Project

- **O**pen**CL** **A**ccelerated Molecular **Dock**ing on **FPGA**s
  - https://git.esa.informatik.tu-darmstadt.de/docking/ocladock-fpga
  - Accelerated AutoDock developed at TU Darmstadt

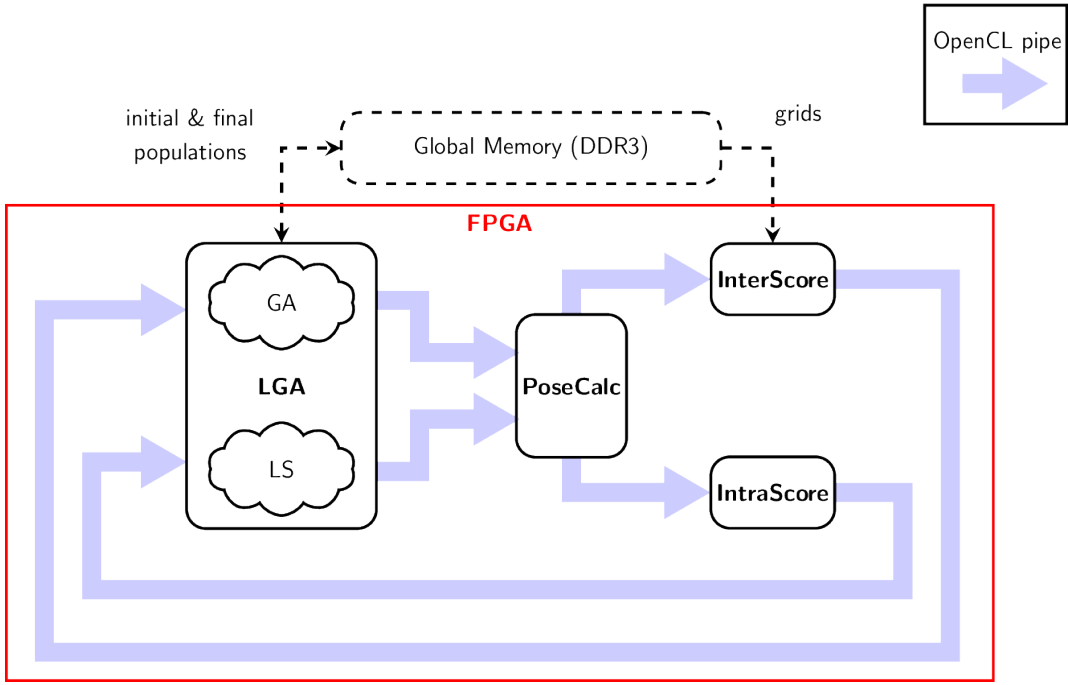
- Computational-intensive parts already well defined
  - LGA run is offloaded onto an FPGA device

- Optimized for Intel FPGAs
  - Refactored code for leveraging
    - Custom hardware pipelines & memory hierarchies

NEC

Baseline: Device Code

- Task-parallel approach
  - Several kernels running simultaneously
  - Kernels communicate with each other using OpenCL pipes
- Each kernel executes a different task
  - GA, LS, energy interactions
  - Some kernels are replicated
- Each kernel is single threaded
  - In contrast to index-based multithreading



Porting from OpenCL to SX-Aurora

- Host
  - Replaced OpenCL host APIs with VEO API
    - Some resemble each other

clCreateBuffer	veo_alloc_mem
clEnqueueWriteBuffer	veo_write_mem

  - Added custom-made wrapper functions for some VEO APIs
    - For more descriptive/extra error messages

```
uint64_t wrapper_veo_get_sym (veo_proc_handle* proc, uint64_t libhdl, const char* symname) {
  uint64_t symbol = veo_get_sym (proc, libhdl, symname);
  if (symbol == 0) {
    std::cout << "\tveo_get_sym():\tfailed to find symbol." << std::endl;
    std::exit (EXIT_FAILURE);
  }
  return symbol;
}
```

Embedded Systems & Applications

Porting from OpenCL to SX-Aurora

- Device
  - OpenCL kernels ----> C functions
    - Main function: performs a single LGA run
  - OpenCL pipes were removed
    - Inter-kernel communication is performed via *function calls instead*
  - Single-threaded baseline
    - OpenCL C structure was maintained
    - Minor code modifications on loop structures

NEC

Initial Optimizations

- Based on NEC tuning guidelines
  - Removing data dependencies
  - Using 4-byte variables for index and loop-control variable
  - Parallelization of LGAs
    - LGA runs are independent
    - ----> distributed among cores using OpenMP

- Fully vectorization time-consuming score/energy functions
  - Ligand-receptor interaction
  - Ligand internal interaction

- Still some room for improvement ...

Performance Tuning: Analysis

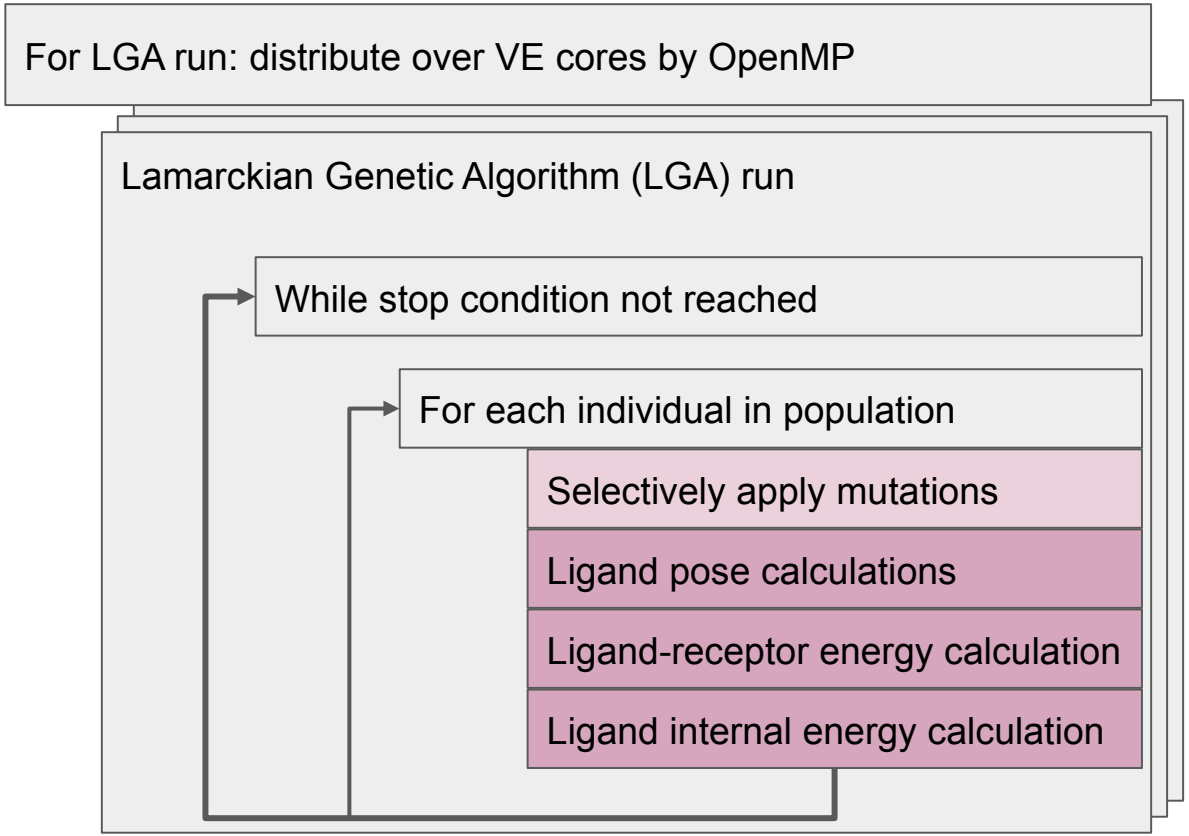
First results:

- Loops short
- Loops often not vectorized
- *Single precision computations!*

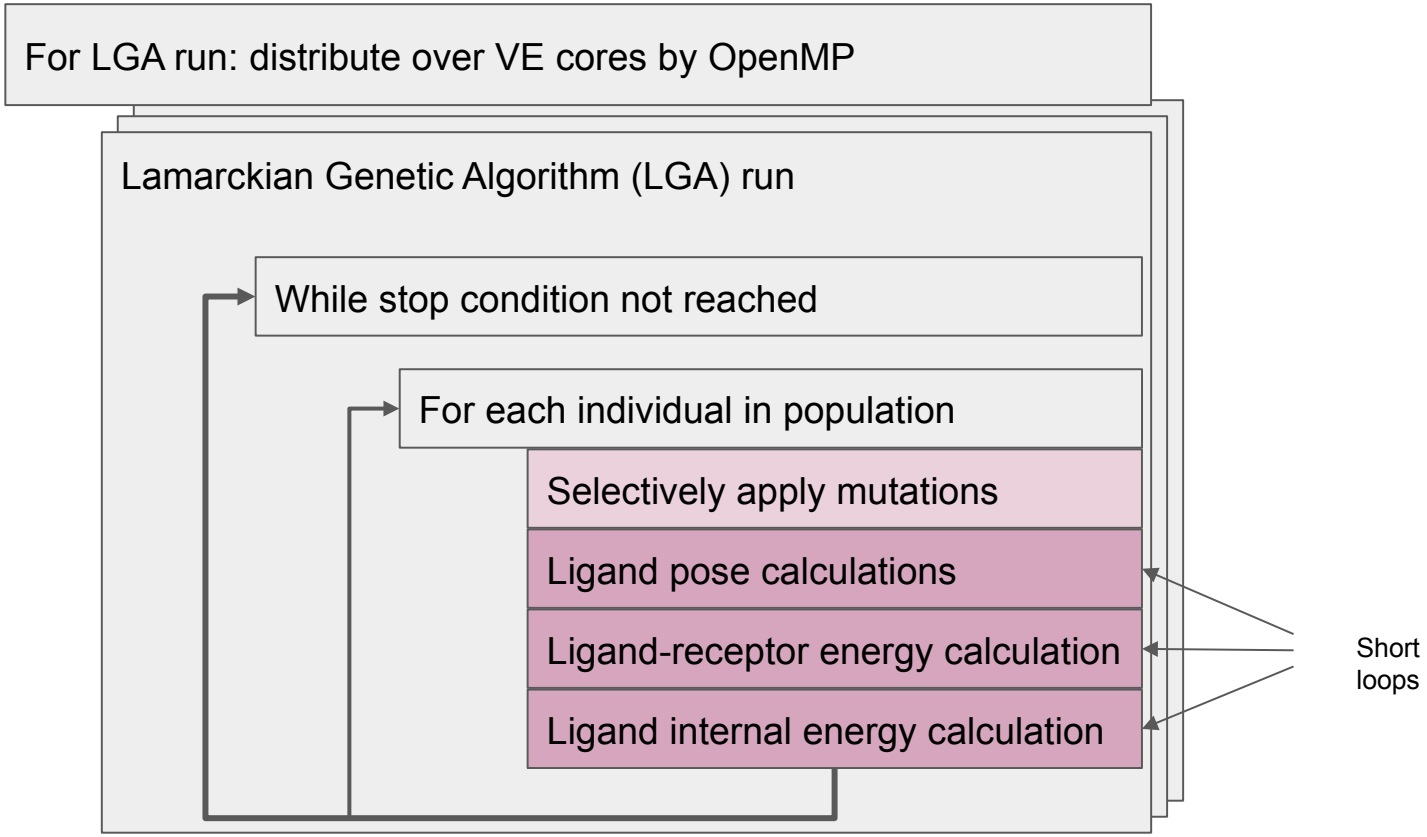
```
******** Program Information ********
Real Time (sec)              :      114.700255
User Time (sec)              :      114.692841
Vector Time (sec)            :       10.757880
Inst. Count                  :    124837499135
V. Inst. Count               :      857519799
V. Element Count             :    31273502842
V. Load Element Count        :     9653868399
FLOP Count                   :    28193233210
MOPS                         :     1361.635301
MOPS (Real)                  :     1361.498080
MFLOPS                       :      245.823994
MFLOPS (Real)                :      245.799221
A. V. Length                 :       36.469715
V. Op. Ratio (%)             :       20.609206
L1 Cache Miss (sec)          :        3.521309
CPU Port Conf. (sec)         :        0.000000
V. Arith. Exec. (sec)        :        1.064510
V. Load Exec. (sec)          :        7.402312
VLD LLC Hit Element Ratio (%)  :       99.994095
FMA Element Count            :      910694400
Power Throttling (sec)       :        0.000000
Thermal Throttling (sec)     :        0.000000
Memory Size Used (MB)        :      302.000000
```

- Key functions:
  - Calculate ligand atoms positions from “genome”
  - Compute ligand-receptor energy
  - Compute ligand internal energy contributions
- Inner loops: over ligand atoms, atom pairs or ligand torsions

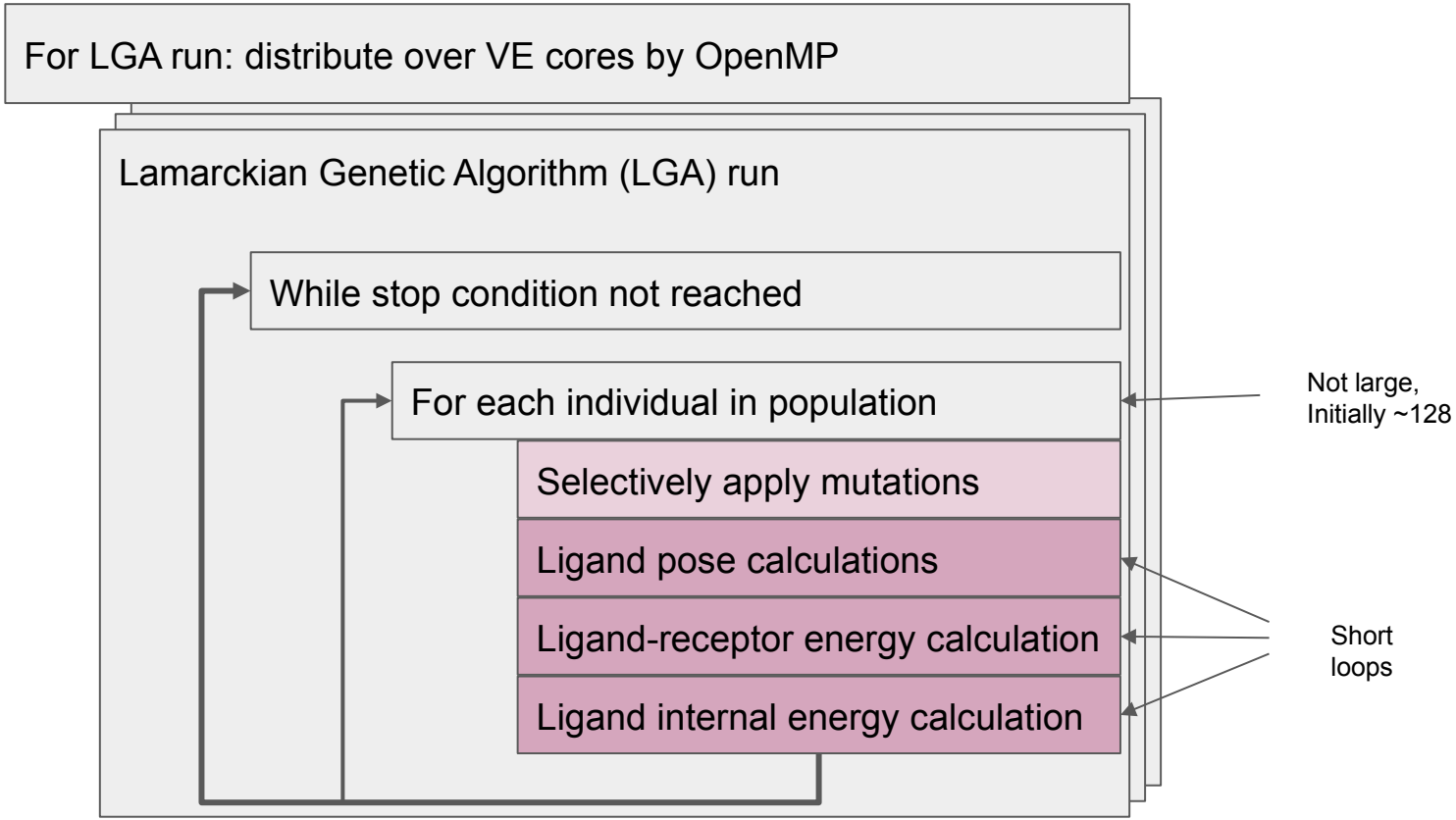
Code Structure



Code Structure



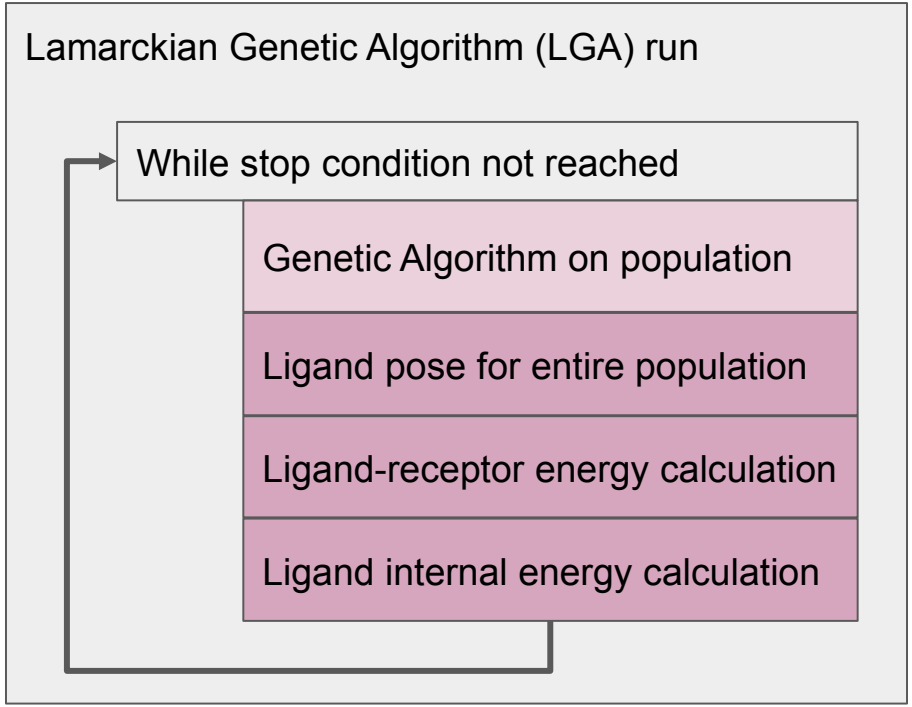
Code Structure



Loop Pushing

- Loop over individuals (genomes)
  - Increase population: 2048
  - Move loop inside functions, make it innermost (for ncc)
  - Tricky for genetic algorithm part

- Replaced random generator by ASL function call
  - Pull many random numbers at once



Loop Pushing: First Results

Case: **1t46**, 40 atoms, 111 rotations
100 LGA runs, population size: 2048

Much better!

```
******** Program Information ********
Real Time (sec)          :       45.922564
User Time (sec)          :      354.048237
Vector Time (sec)        :      313.792794
Inst. Count              :    352157100866
V. Inst. Count           :    108164054847
V. Element Count         :  23871398751364
V. Load Element Count    :   2412434798072
FLOP Count               :  17850210880912
MOPS                     :    85464.630971
MOPS (Real)              :   654920.857127
MFLOPS                   :    50724.155528
MFLOPS (Real)            :   388702.402827
A. V. Length             :      220.589096
V. Op. Ratio (%)         :       99.188735
L1 Cache Miss (sec)      :        8.237327
CPU Port Conf. (sec)     :       12.740749
V. Arith. Exec. (sec)    :      197.959744
V. Load Exec. (sec)      :      105.976551
VLD LLC Hit Element Ratio (%) :  96.077466
FMA Element Count        :  5960253027502
Max Active Threads       :               8
Available CPU Cores      :               8
Average CPU Cores Used   :       7.709679
Memory Size Used (MB)    :    1492.000000
```

Loop Pushing: First Results

Case: **1t46**, 40 atoms, 111 rotations
100 LGA runs, population size: 2048

Much better!

```
******** Program Information ********
Real Time (sec)          :       45.922564
User Time (sec)          :      354.048237
Vector Time (sec)        :      313.792794
Inst. Count              :    352157100866
V. Inst. Count           :    108164054847
V. Element Count         :  23871398751364
V. Load Element Count    :   2412434798072
FLOP Count               :  17850210880912
MOPS                     :     85464.630971
MOPS (Real)              :    654920.857127
MFLOPS                   :     50724.155528
MFLOPS (Real)            :    388702.402827
A. V. Length             :       220.589096
V. Op. Ratio (%)         :        99.188735
L1 Cache Miss (sec)      :         8.237327
CPU Port Conf. (sec)     :        12.740749
V. Arith. Exec. (sec)    :       197.959744
V. Load Exec. (sec)      :       105.976551
VLD LLC Hit Element Ratio (%) :       96.077466
                         :  5960253027502
                         :             8
                         :             8
                         :         7.709679
                         :   1492.000000
```

```
******** FTRACE ********
FREQUENCY  EXCLUSIVE       AVER.TIME     MOPS   MFLOPS  V.OP  AVER.    VECTOR L1CACHE CPU PORT VLD LLC PROC.NAME
        TIME[sec](  % )    [msec]                 RATIO V.LEN      TIME    MISS     CONF HIT E.%

  262319   187.783( 55.4)     0.716 124156.5  76670.0  99.24 222.2    182.542   0.766    0.000   97.34 energy_ia
  262319    59.600( 17.6)     0.227  49133.4  14087.6  99.66 220.8     58.935   0.293   12.676   92.05 energy_ie
  262319    51.431( 15.2)     0.196  64558.4  49827.2  99.47 216.9     49.268   0.838    0.000   89.96 calc_pc
    1106    19.847(  5.9)    17.945  18214.0   2132.0  99.48 217.7     18.815   0.838    0.003   87.31 perform_ls
       8    15.535(  4.6)  1941.846   2034.9      0.0   0.00   0.0      0.000   1.084    0.000    0.00 libkernel_ga$1
  262219     3.871(  1.1)     0.015  28794.5   2029.0  99.32 155.2      2.156   1.072    0.000   98.95 randf_vec
     100     0.824(  0.2)     8.239   3931.7    585.5  86.26  13.5      0.797   0.013    0.062   96.12 lga
```

(cumulated times, 8 OpenMP threads)

Loop Pushing: First Results

Case: **1t46**, 40 atoms, 111 rotations
100 LGA runs, population size: 2048

```
******** Program Information ********
Real Time (sec)        :    45.922564
```

VE 10B, 8 core, 1.4GHz
UMA mode

Comparison with other systems:

Testcase	Runtimes [s]: popsize = 2048						
	NVIDIA V100	NVIDIA RTX2070	NVIDIA RTX3070	NVIDIA A100	AMD EPYC 7502 2x32-core	AMD EPYC 7742 2x64-core CPU	Intel SKL Gold 6126 12-core
1t46	6.56	14.89	10.81	2.62	108.04	61.59	425.64

V100 vs. VE10
3x SP peak
but 7x better

Embedded Systems & Applications

Packed Single Precision Struggles

- VE reaches ⅓ of v100 single precision performance only in packed mode!
  - 2 x 32 bit floats in 64 bit word
  - Vector length effectively up to 512 elements, must be even
- Code vs. Compiler (ncc)
  - SP functions: *floorf(), ceilf(), sinf(), expf()* **must** be used, not *floor(), ceil(), sin(), exp()*.
  - Issues that hinder packed vectorization
    - IF blocks inside loop (waiting for fix)
    - *unsigned int* loop index (“ “ “)
    - Gather/Scatter (won’t fix)
  - GPU code uses *native_exp(), native_sin(),* etc…
    - Defined in OpenCL, can be implemented with reduced accuracy
- => unpacked vectorized code, with proper SP functions:
  - From ~43s down to **34.3s** (factor 5.2 to v100)

NEC

Embedded Systems & Applications

LLVM-VE and RegionVectorizer

- Simon Moll (NEC HPCE)
- RV is an outer loop vectorizer, can deal excellently with predication (IF blocks)
  - Whole function vectorization of SLEEF math library functions
- + recent work on backend support for packed vectorization
- See Simon's talk on Friday, 8:35

- Compiling only energy_ia.c with LLVM-VE-RV: packed vectorization!

NEC

Embedded Systems & Applications

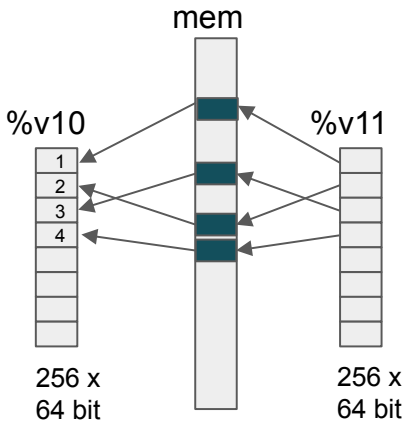
Further Tunings

- Multi-VE Processes for NUMA
  - In NUMA mode an ncc OpenMP process can only get 4 threads
    - Actually could have used LLVM-VE...
  - Using AVEO multi-VE capability to:
    - Support multiple processes on one VE => NUMA with OpenMP is possible
    - Support multiple VEs

- Reduced accuracy implementations of *sqrtf()* and *expf()*
- No packed mode possible for energy_ie.c
  - Dominated by indirect loads of data / vector gather operations

NEC

Packed Single Precision: VE ISA Deficits



Vector Gather / Scatter are impossible in packed mode

```
VGT %v10, %v11, %s12, %s13 [,%vm14]
```

target 64 bit

addresses 64 bit

overlap hints

"Packed" loading 512 x 32 bit elements requires 2 x VGT, VSHF, VMRG

For arrays smaller than 2^{32} elements the address VREG could be instead interpreted as a packed array of 4-byte aligned offsets, requiring just one such packed gather/scatter. Example:

```
PVGTO %v10, %v11, %s12
```

target 32 bit

offsets 32 bit

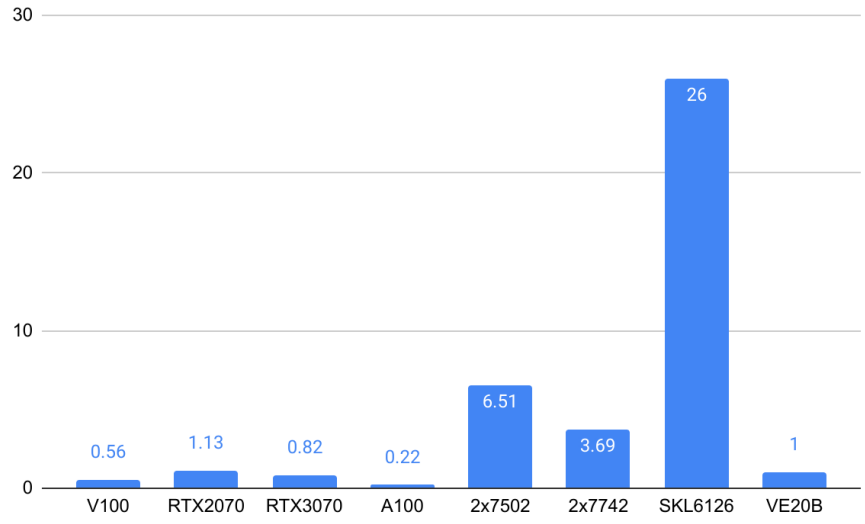
base address 64 bit

Finally: Evaluation Results

Testcase	Runtimes [s]: popsize = 2048							
	NVIDIA V100	NVIDIA RTX2070	NVIDIA RTX3070	NVIDIA A100	AMD EPYC 7502 2x32-core	AMD EPYC 7742 2x64-core CPU	Intel SKL Gold 6126 12-core	NEC SX-Aurora VE20B (NUMA)
1t46	6.56	14.89	10.81	2.62	108.04	61.59	425.64	**12.28**

Geometric mean over 31 validation tests

- Runtimes normalized to VE20B results
- (Includes many small cases)
- popsize=2048, nrun=100
- AutoDock-GPU v1.1



Embedded Systems & Applications

Conclusions

- Porting OpenCL-AutoDock onto SX-Aurora
  - Relatively smooth, main code structure was maintained
  - Leveraged VEO/AVEO
- Optimization for SX-Aurora was much more involved
  - Increased the population size (2048) from default (150) for vector length
  - Loop pushing improved performance *significantly*
    - Local-Search, score/energy calculations
  - Packed mode is mandatory for full single precision performance
    - LLVM-VE is currently the most advanced
- Competitive performance
  - > 3.7x times faster than modern 128 core servers, at lower power consumption
  - Better than expected performance compared to GPUs

NEC

Embedded Systems & Applications

Outlook

- Working with Scripps Research on joining the Open Pandemics - COVID19 effort with spare SX-Aurora cycles
  - Virtual screening of COVID19 drug candidates

NEC