Optimizing simulations by exploring design space with CORHPEX

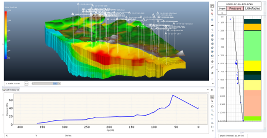
December 9, 2024

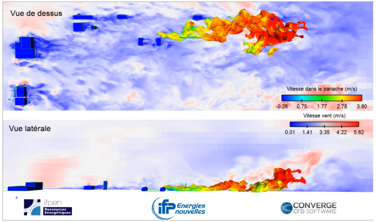
Lana Scravaglieri

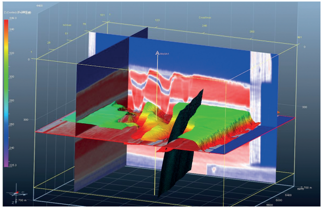
supervised by Ani Anciaux-Sedrakian, Thomas Guignon, Olivier Aumage, and Mihail Popov

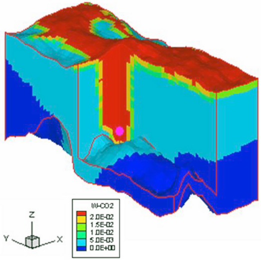
IFPEN, R11 - Inria STORM

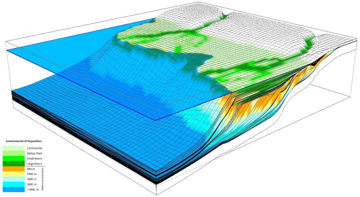
Simulators

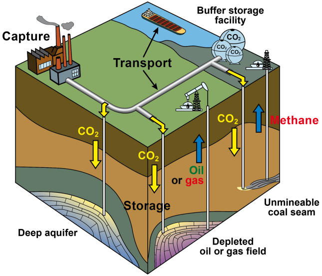

Pressure
Lithofacies


Vue de dessus
Vitesse dans le panache (m/s)
-0.26 0.75 1.77 2.78 3.80
Vitesse vent (m/s)
0.01 1.41 3.35 4.22 5.62
Vue latérale
ifpen Ressources Energétiques
ifp Energies nouvelles
CONVERGE CFD SOFTWARE




W-CO2
2.0E-02
1.5E-02
1.0E-02
5.0E-03
0.0E+00
Z
Y
X


Environments of Deposition


Capture
Buffer storage facility
Transport
CO_2
CO_2
CO_2
Methane
Oil
or gas
Storage
Unmineable coal seam
Deep aquifer
Depleted oil or gas field

Complex systems

Complex systems

- Number of threads
- Memory hierarchy : memory, L3, L2, L1
- Non-Uniform Memory Access (NUMA) effects
- Simultaneous Multithreading (SMT)
- Thread placement (binding policy)
- Prefetchers (may require root privileges)
- Frequency (may require root privileges)
- Instruction set
- Accuracy (simple or double precision, compliance with IEEE standard)
- Compiler optimizations

Complex systems

- Number of threads
- Memory hierarchy : memory, L3, L2, L1
- Non-Uniform Memory Access (NUMA) effects
- Simultaneous Multithreading (SMT)
- Thread placement (binding policy)
- Prefetchers (may require root privileges)
- Frequency (may require root privileges)
- Instruction set
- Accuracy (simple or double precision, compliance with IEEE standard)
- Compiler optimizations

Values can improve or reduce execution time and/or energy consumption.
Parameters influence each other.

Impact on simulator codes

- Simulators are often memory-bound

Impact on simulator codes

- Simulators are often memory-bound
  - computation could go faster

Impact on simulator codes

- Simulators are often memory-bound
  - computation could go faster
  - data doesn't travel fast enough in the memory hierarchy

Impact on simulator codes

- Simulators are often memory-bound
  - computation could go faster
  - data doesn't travel fast enough in the memory hierarchy
- Simulators are not always vectorizable (data layout, branches)

Impact on simulator codes

- Simulators are often memory-bound
  - computation could go faster
  - data doesn't travel fast enough in the memory hierarchy
- Simulators are not always vectorizable (data layout, branches)
- Different behaviors

Impact on simulator codes

- Simulators are often memory-bound
  - computation could go faster
  - data doesn't travel fast enough in the memory hierarchy
- Simulators are not always vectorizable (data layout, branches)
- Different behaviors
  - For different simulators

Impact on simulator codes

- Simulators are often memory-bound
  - computation could go faster
  - data doesn't travel fast enough in the memory hierarchy
- Simulators are not always vectorizable (data layout, branches)
- Different behaviors
  - For different simulators
    - For different phases in a simulator

Impact on simulator codes

- Simulators are often memory-bound
  - computation could go faster
  - data doesn't travel fast enough in the memory hierarchy
- Simulators are not always vectorizable (data layout, branches)
- Different behaviors
  - For different simulators
    - For different phases in a simulator
  - On different architectures

Impact on simulator codes

- Simulators are often memory-bound
  - computation could go faster
  - data doesn't travel fast enough in the memory hierarchy
- Simulators are not always vectorizable (data layout, branches)
- Different behaviors
  - For different simulators
    - For different phases in a simulator
  - On different architectures
  - With different inputs (size and nature)

Impact on simulator codes

- Simulators are often memory-bound
  - computation could go faster
  - data doesn't travel fast enough in the memory hierarchy
- Simulators are not always vectorizable (data layout, branches)
- Different behaviors
  - For different simulators
    - For different phases in a simulator
  - On different architectures
  - With different inputs (size and nature)

Impact on simulator codes

- Simulators are often memory-bound
  - computation could go faster
  - data doesn't travel fast enough in the memory hierarchy
- Simulators are not always vectorizable (data layout, branches)
- Different behaviors
  - For different simulators
    - For different phases in a simulator
  - On different architectures
  - With different inputs (size and nature)

There is no recipe to get the best performance and energy consumption on every machine for every application.

A framework for optimization space exploration: CORHPEX

A framework for optimization space exploration: CORHPEX

- Few points can be tested manually

A framework for optimization space exploration: CORHPEX

- Few points can be tested manually
- Manual exploration may be biased

A framework for optimization space exploration: CORHPEX

- Few points can be tested manually
- Manual exploration may be biased
- An assumption may be valid at some point but that can change overtime

A framework for optimization space exploration: CORHPEX

- Few points can be tested manually
- Manual exploration may be biased
- An assumption may be valid at some point but that can change overtime

A framework for optimization space exploration: CORHPEX

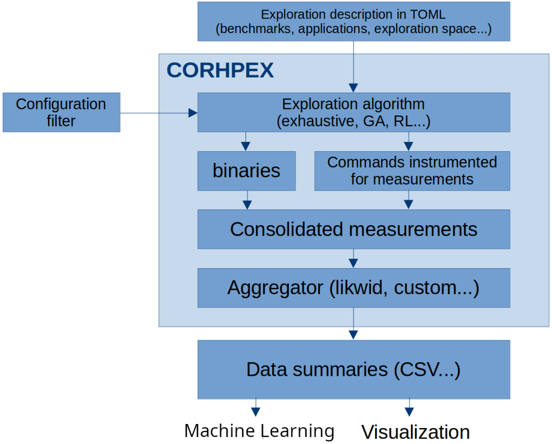
- Few points can be tested manually
- Manual exploration may be biased
- An assumption may be valid at some point but that can change overtime

> We need a faster, portable and unbiased way to find optimizations.

A framework for optimization space exploration: CORHPEX

- Few points can be tested manually
- Manual exploration may be biased
- An assumption may be valid at some point but that can change overtime

We need a faster, portable and unbiased way to find optimizations.



Target and constraints

Target and constraints

Optimization target

- Time (performance)
- Energy (energy savings)
- EDP (time x energy)

Target and constraints

Optimization target

- Time (performance)
- Energy (energy savings)
- EDP (time x energy)

Constraints

- Accuracy (simple, double, mixed precision)
- Micro-architecture (the machine used for the execution)

The space

Dimension	Sub-dimension	Options	Size
Number of threads*		1,4,8,16,32,64	6
Thread binding	package	First,Last	2
	die	First,Last	2
	L3	First,Last	2
	smt	First,Last	2
Prefetchers	DCU IP-correlated	On,Off	2
	DCU	On,Off	2
	L2 Adj. Cache Line	On,Off	2
	L2 Streamer	On,Off	2
Compiler flags	Optimization flags	-O2,-O3, -O3 without vectorization	3
	Vecto cost model	very cheap, cheap, dynamic	3
	fast-maths	On, Off	2
	Instruction set*	-msse4,-mavx2,-march=native	3
Precision†	Storage	float, double	2
	Computing	float, double	2

The space

Dimension	Sub-dimension	Size	IFPEN	[1]	[2]	[3]
Number of threads*		6	X		X	X
Thread binding	package	2		partial with data mapping	partial with data mapping	partial with data mapping
	die	2				
	L3	2				
	smt	2				
Prefetchers	DCU IP-correlated	2				X
	DCU	2				
	L2 Adj. Cache Line	2				
	L2 Streamer	2				
Compiler flags	Optimization flags	3	partial			
	Vecto cost model	3				
	fast-maths	2				
	Instruction set*	3				
Precision†	Storage	2	X			
	Computing	2				

1 M. Diener et. al. "Characterizing communication and page usage of parallel applications for thread and data mapping," 2015, doi: 10.1016/j.peva.2015.03.001.

2 M. Popov et. al., "Efficient thread/page/parallelism autotuning for numa systems," 2019, doi: 10.1145/3330345.3330376.

3 I. Sánchez Barrera et. al., "Modeling and optimizing numa effects and prefetching with machine learning," 2020. doi: 10.1145/3392717.3392765.

The applications to learn from

The applications to learn from

- From IFPEN

The applications to learn from

- From IFPEN
  - Physics : CapillaryPressure, RelativePermeability...

The applications to learn from

- From IFPEN
  - Physics : CapillaryPressure, RelativePermeability...
    - Carbon capture: ShArc and Geoxim

The applications to learn from

- From IFPEN
  - Physics : CapillaryPressure, RelativePermeability...
    - Carbon capture: ShArc and Geoxim
  - Basic linear algebra kernels used in linear solvers

The applications to learn from

- From IFPEN
  - Physics : CapillaryPressure, RelativePermeability...
    - Carbon capture: ShArc and Geoxim
  - Basic linear algebra kernels used in linear solvers
    - MCGSolver

The applications to learn from

- From IFPEN
  - Physics : CapillaryPressure, RelativePermeability...
    - Carbon capture: ShArc and Geoxim
  - Basic linear algebra kernels used in linear solvers
    - MCGSolver
- Benchmarks

The applications to learn from

- From IFPEN
  - Physics : CapillaryPressure, RelativePermeability...
    - Carbon capture: ShArc and Geoxim
  - Basic linear algebra kernels used in linear solvers
    - MCGSolver
- Benchmarks
  - NAS (NPB)

The applications to learn from

- From IFPEN
  - Physics : CapillaryPressure, RelativePermeability...
    - Carbon capture: ShArc and Geoxim
  - Basic linear algebra kernels used in linear solvers
    - MCGSolver
- Benchmarks
  - NAS (NPB)
  - Rodinia

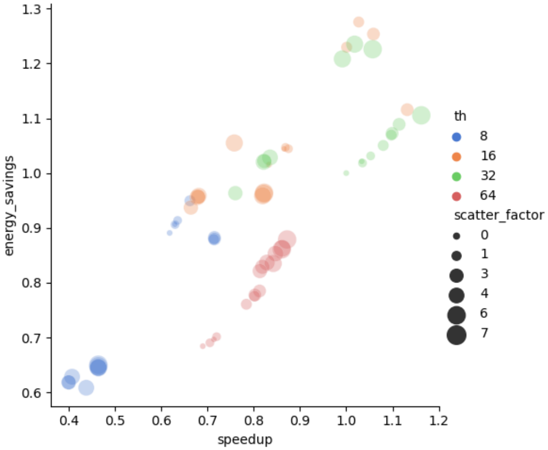
The applications to learn from

- From IFPEN
  - Physics : CapillaryPressure, RelativePermeability...
    - Carbon capture: ShArc and Geoxim
  - Basic linear algebra kernels used in linear solvers
    - MCGSolver
- Benchmarks
  - NAS (NPB)
  - Rodinia
  - LULESH

The applications to learn from

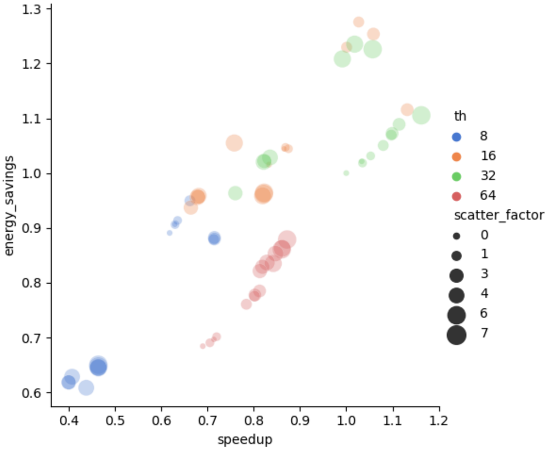
- From IFPEN
  - Physics : CapillaryPressure, RelativePermeability...
    - Carbon capture: ShArc and Geoxim
  - Basic linear algebra kernels used in linear solvers
    - MCGSolver
- Benchmarks
  - NAS (NPB)
  - Rodinia
  - LULESH
  - PARSEC

Experimental results: small exhaustive exploration



- CapillaryPressureLaw
- AMD EPYC 7301 (Zen), 2 CPUs, 16 cores/CPU (Grid5000)
- Space
  - Number of threads
  - Binding policy

Experimental results: small exhaustive exploration

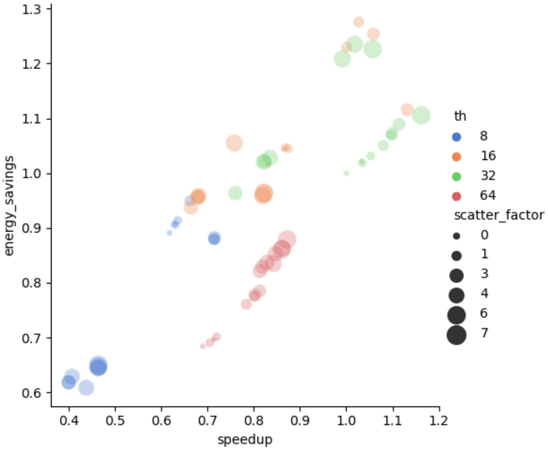


- CapillaryPressureLaw
- AMD EPYC 7301 (Zen), 2 CPUs, 16 cores/CPU (Grid5000)
- Space
  - Number of threads
  - Binding policy

4 configurations in Pareto set:

- 3 with 16 th on 1 CPU,
- 1 with 32 th on 2 CPUs.

Experimental results: small exhaustive exploration



- CapillaryPressureLaw
- AMD EPYC 7301 (Zen), 2 CPUs, 16 cores/CPU (Grid5000)
- Space
  - Number of threads
  - Binding policy

4 configurations in Pareto set:

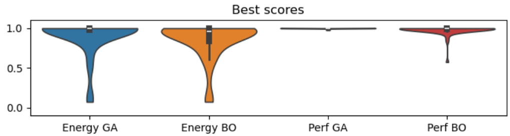
- 3 with 16 th on 1 CPU,
- 1 with 32 th on 2 CPUs.

Need to cooptimize the 2 dimensions.

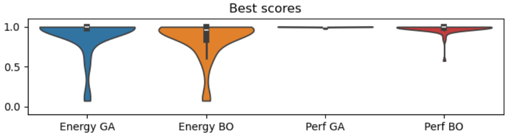
Experimental results: optimization algorithms

- NAS Parallel benchmark, Rodinia, PARSEC benchmark, LULESH, CLOMP
- Intel Xeon Gold 6130 (Skylake)
- Space
  - Number of threads
  - Binding policy
  - Page mapping
  - Prefetchers
  - Multithreading
- Algorithms
  - Genetic Algorithm
  - Bayesian Optimization

Experimental results: optimization algorithms

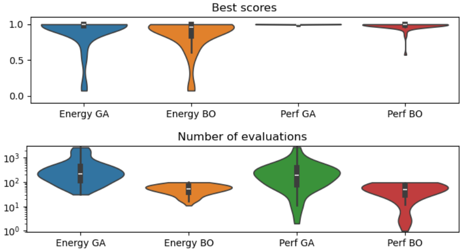


Experimental results: optimization algorithms



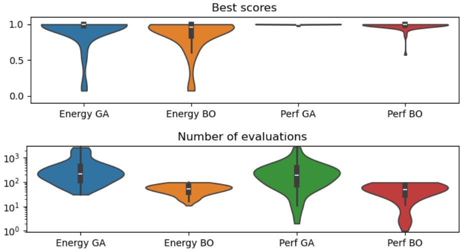
Performance easier to optimize than energy.

Experimental results: optimization algorithms



Performance easier to optimize than energy.

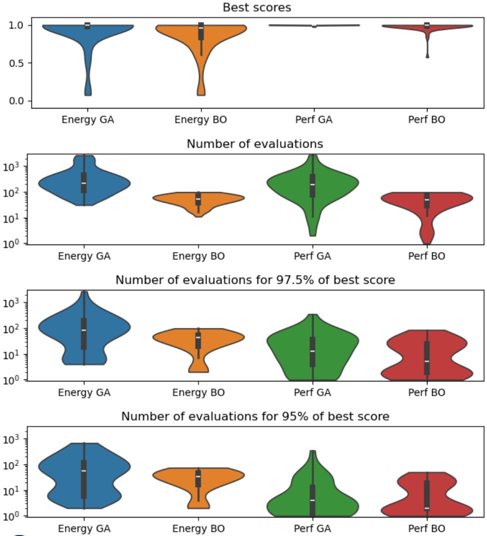
Experimental results: optimization algorithms



Performance easier to optimize than energy.

GA achieves better scores but BO is faster.

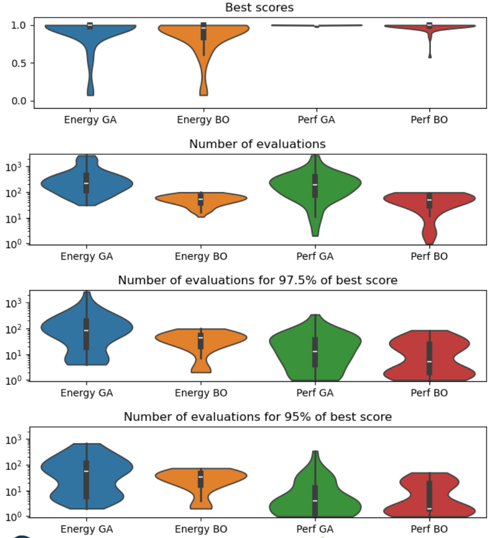
Experimental results: optimization algorithms



Performance easier to optimize than energy.

GA achieves better scores but BO is faster.

Experimental results: optimization algorithms

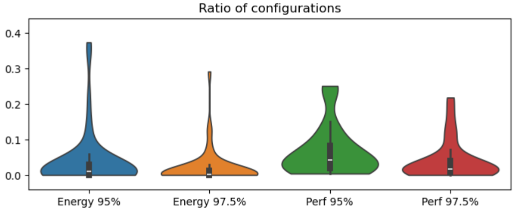


Performance easier to optimize than energy.

GA achieves better scores but BO is faster.

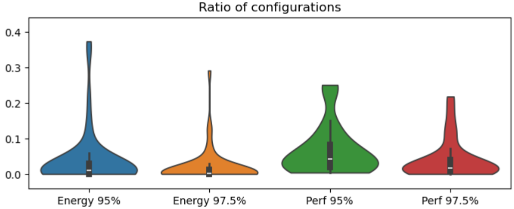
Achieving 97.5% or 95% of optimal score is faster.

Experimental results: optimization algorithms



For most codes less than 5% of the configurations achieve at least 95% of the gains.

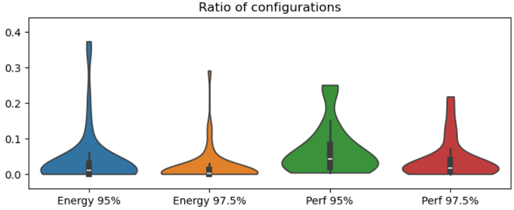
Experimental results: optimization algorithms



For most codes less than 5% of the configurations achieve at least 95% of the gains.

Less configurations achieve it for energy than for time.

Experimental results: optimization algorithms



For most codes less than 5% of the configurations achieve at least 95% of the gains.

Less configurations achieve it for energy than for time.

It is difficult.

Experimental results: Surrogate models

- What ?

Experimental results: Surrogate models

- What ?
  - predictive model of kernel performance/energy

Experimental results: Surrogate models

- What ?
  - predictive model of kernel performance/energy
- How ?

Experimental results: Surrogate models

- What ?
  - predictive model of kernel performance/energy
- How ?
  - BO trains a predictive model

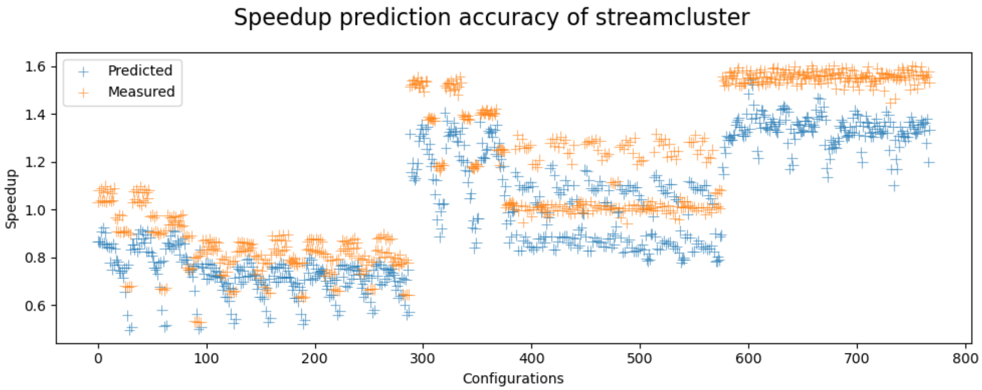
Experimental results: Surrogate models

- What ?
  - predictive model of kernel performance/energy
- How ?
  - BO trains a predictive model
- Why ?

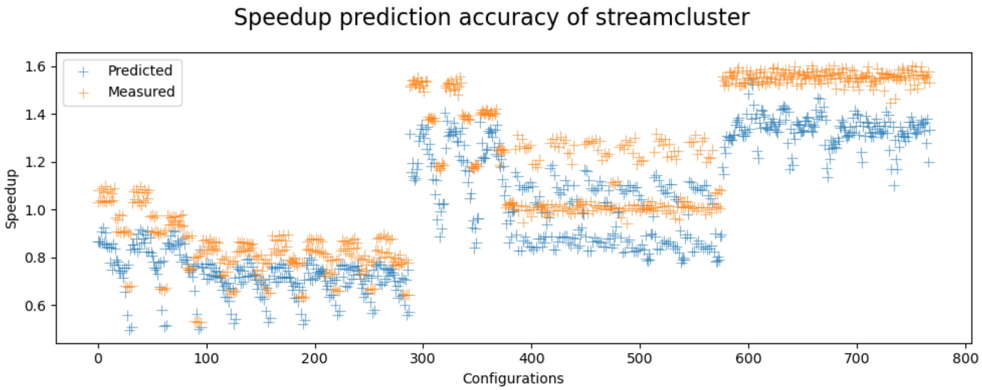
Experimental results: Surrogate models

- What ?
  - predictive model of kernel performance/energy
- How ?
  - BO trains a predictive model
- Why ?
  - faster than execution

Experimental results: Surrogate models

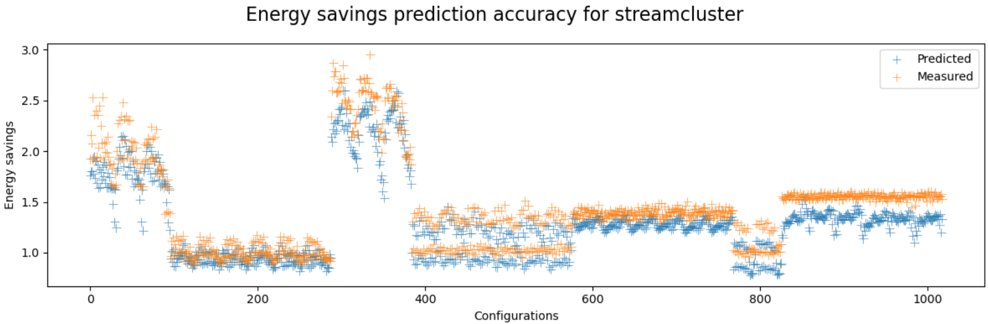


Experimental results: Surrogate models



Predictions are usually below the measures but trend is captured with 9% of the space explored.

Experimental results: Surrogate models



Conclusion

- complex hardware-software interactions with many parameters at play

Conclusion

- complex hardware-software interactions with many parameters at play
- CORHPEX: framework to explore large and complex parameter space to optimize codes execution

Conclusion

- complex hardware-software interactions with many parameters at play
- CORHPEX: framework to explore large and complex parameter space to optimize codes execution
    - draw conclusions on good practices for developers and users

Conclusion

- complex hardware-software interactions with many parameters at play
- CORHPEX: framework to explore large and complex parameter space to optimize codes execution
  - draw conclusions on good practices for developers and users
  - train surrogate models (full space virtual exploration, cooptimize multiple kernels. . .)

Conclusion

- complex hardware-software interactions with many parameters at play
- CORHPEX: framework to explore large and complex parameter space to optimize codes execution
  - draw conclusions on good practices for developers and users
  - train surrogate models (full space virtual exploration, cooptimize multiple kernels. . .)
- CORHPEX current status

Conclusion

- complex hardware-software interactions with many parameters at play
- CORHPEX: framework to explore large and complex parameter space to optimize codes execution
  - draw conclusions on good practices for developers and users
  - train surrogate models (full space virtual exploration, cooptimize multiple kernels...)
- CORHPEX current status
  - parameters: compiler flags, runtime flags, environement variable, commands

Conclusion

- complex hardware-software interactions with many parameters at play
- CORHPEX: framework to explore large and complex parameter space to optimize codes execution
  - draw conclusions on good practices for developers and users
  - train surrogate models (full space virtual exploration, cooptimize multiple kernels. . .)
- CORHPEX current status
  - parameters: compiler flags, runtime flags, environement variable, commands
  - including GPU

Conclusion

- complex hardware-software interactions with many parameters at play
- CORHPEX: framework to explore large and complex parameter space to optimize codes execution
  - draw conclusions on good practices for developers and users
  - train surrogate models (full space virtual exploration, cooptimize multiple kernels. . .)
- CORHPEX current status
  - parameters: compiler flags, runtime flags, environement variable, commands
  - including GPU
  - 3 exploration algorithms: GA, BO, exhaustive (extendable)

Conclusion

- complex hardware-software interactions with many parameters at play
- CORHPEX: framework to explore large and complex parameter space to optimize codes execution
  - draw conclusions on good practices for developers and users
  - train surrogate models (full space virtual exploration, cooptimize multiple kernels. . .)
- CORHPEX current status
  - parameters: compiler flags, runtime flags, environement variable, commands
  - including GPU
  - 3 exploration algorithms: GA, BO, exhaustive (extendable)
  - metrics collected with likwid (extendable)

Future work

- use code embeddings to find similarities between kernels

Future work

- use code embeddings to find similarities between kernels
- optimize with CORHPEX

Future work

- use code embeddings to find similarities between kernels
- optimize with CORHPEX
  - GPU applications

Future work

- use code embeddings to find similarities between kernels
- optimize with CORHPEX
  - GPU applications
  - software-defined radio applications (internship)

Future work

- use code embeddings to find similarities between kernels
- optimize with CORHPEX
  - GPU applications
  - software-defined radio applications (internship)
- add features

Future work

- use code embeddings to find similarities between kernels
- optimize with CORHPEX
  - GPU applications
  - software-defined radio applications (internship)
- add features
  - support for PAPI metrics collection

Future work

- use code embeddings to find similarities between kernels
- optimize with CORHPEX
  - GPU applications
  - software-defined radio applications (internship)
- add features
  - support for PAPI metrics collection
  - automatic visualization