C++ Performance Portability Frameworks

Kokkos vs Openfpm_data vs RAJA

Kokkos Vs Raja

- “Evaluation of performance portability frameworks for the implementation of a particle-in-cell code”, **2019**
  - A particle-in-cell model was parallelized using OpenMP, OpenACC, CUDA, Kokkos, and RAJA, targeting multi-core (CPU) and graphics (GPU) processors.
  - Single node only.
- Goals were to
  - Evaluate how much abstraction degrades performance.
  - Ease of development.

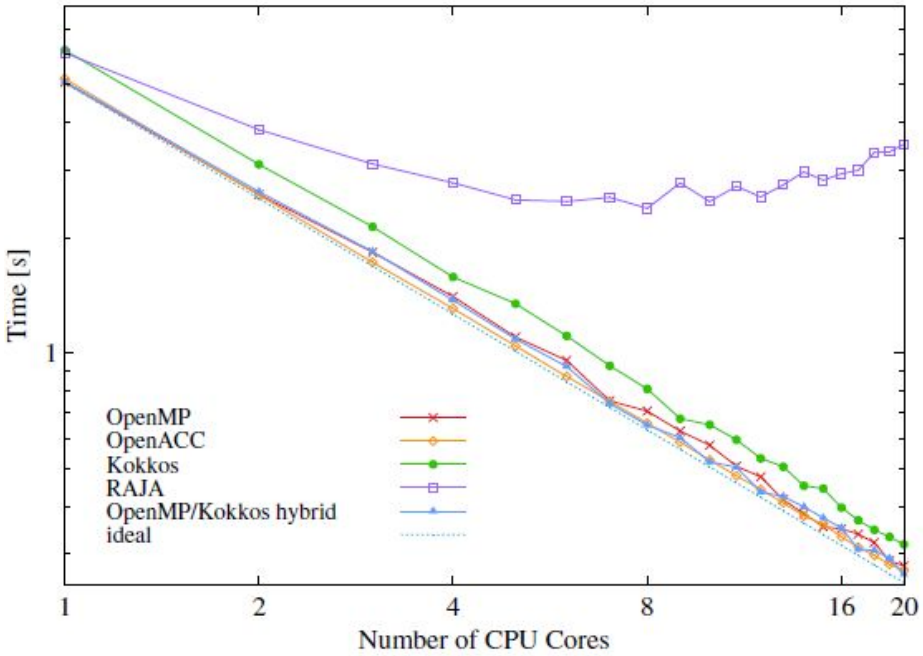
Kokkos Vs Raja

- Conclusion from authors: “Use Kokkos”!!
- In their website, Raja still cites it as Work-in-progress.
- Raja requires extra code while porting multi-cpu code to GPU one, Kokkos does not.
- Qualitative comparison of the programming models based on a subjective ranking.

criterion	OpenMP	OpenACC	CUDA	Kokkos	RAJA
code clarity	high	high	low	medium	medium
productivity	high	medium	low	medium	medium
portability	low	medium	low	high	high
performance	high	high	high	high	medium

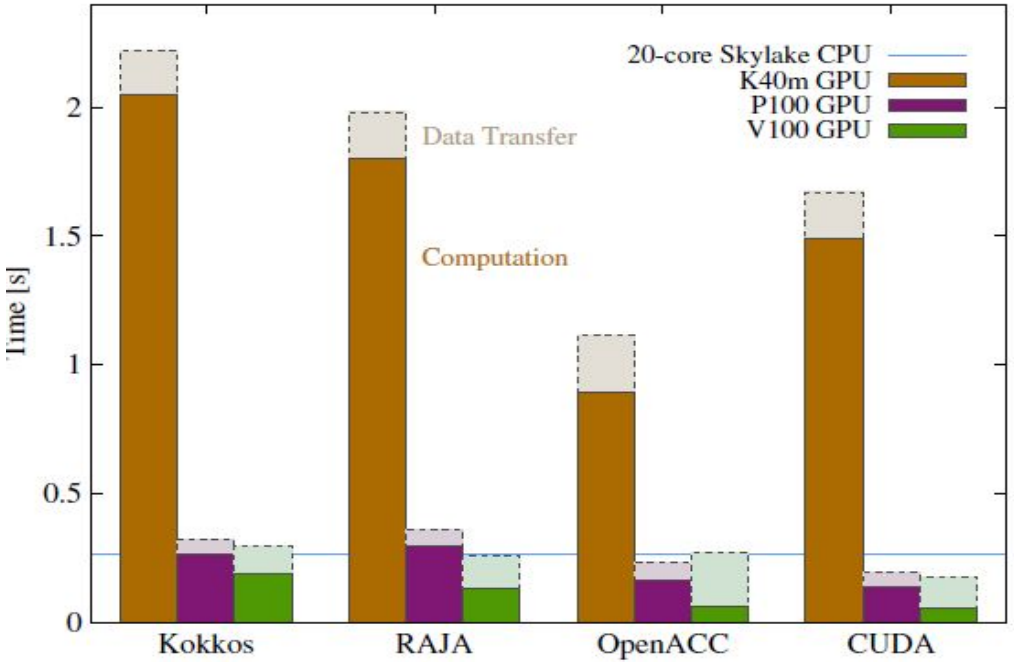
Kokkos Vs Raja

- CPU Performance:



Kokkos Vs Raja

- GPU Performance:

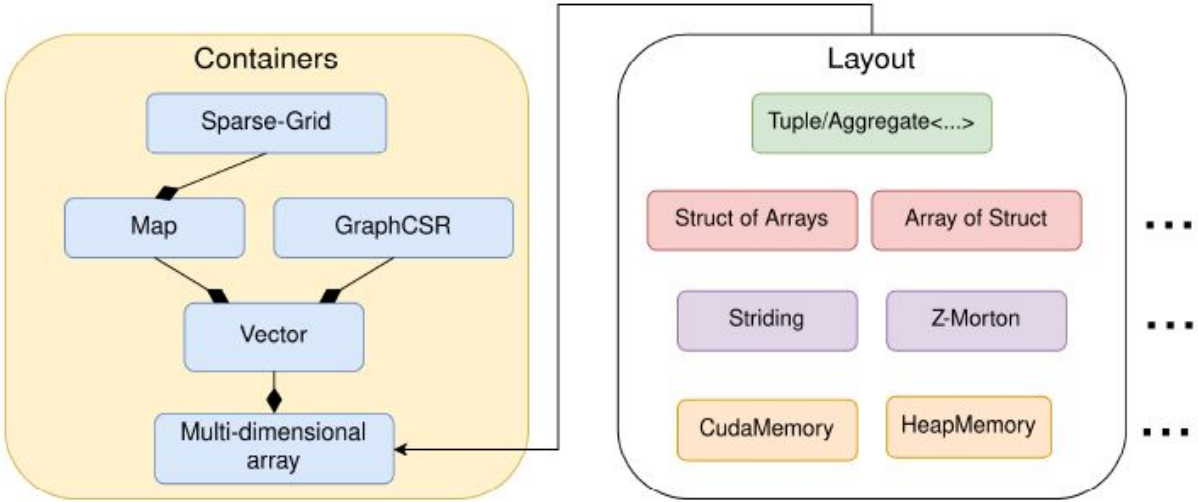


Kokkos vs Openfpm_data

- Using the paper: “A C++ library for memory layout and performance portability of scientific applications” (from openFPM developers).
- Extends std:tuple for complex data types.
- Several functional advantages over Kokkos:
  - Supports more than Multi-dimensional arrays (Sparse data structure, Map)
  - Allows native serializing deserializing
  - **Memory layout restructuring capabilities for non-primitive type**.
- Zero-copy interfaces are possible with other libraries (incl Kokkos).

Kokkos vs Openfpm_data

- UML diagram summarizing openfpm_data library:




```
grid.get<stress>(element)[x][y]
```

Kokkos vs Openfpm_data

- Perhaps the biggest focus of the openfpm paper is memory performance.
- Couldn’t compare based on ease of use, but the openfpm should be more intuitive.

Hardware	`openfpm_data`	Kokkos	Plain C++
A100	(1390/1212)	(1375/1131)	(1394/1226)
RTX 3090	(868/818)	(869/819)	(868/818)
M1	(47.5/27.5)	(43.1/28.6)	(47.8/26.1)
POWER 9	(120.2/109.8)	(143.0/112.8)	(121.6/111.8)
Ryzen 3990X	(70.8/37.7)	(54.0/32.8)	(70.6/37.7)
EPYC 7702	(242.5/135.3)	(243.6/134.7)	(243.9/133.2)
Xeon 8276	(137.1/87.7)	(142.9/89.6)	(144.3/89.6)
RXVega 64	(359/358)	(323/293)	(359/360)

Kokkos vs Openfpm_data

- For compute performance, no direct comparison data between the two is available.
- However, openfpm was compared against several implementations including kokkos, and maintained comparable performance with the best one among baselines across hardwares:

Hardware	`openfpm_data`/miniBude	best miniBude
A100	1.00 ± 0.07	CUDA
RTX 3090	1.00 ± 0.04	CUDA
M1	$\mathbf{1.05 \pm 0.01}$	OpenMP
POWER 9	0.80 ± 0.09	OpenMP
Ryzen 3990X	$\mathbf{1.08 \pm 0.04}$	OpenMP
EPYC 7702	1.01 ± 0.03	OpenMP

SoA vs AoS

- Structure of Arrays (SoA) vs Arrays of Structs (AoS):
  - SoA should help with vectorization.
  - Openfpm natively supports both, kokkos natively doesn’t (though Cabana does).
  - Interestingly, first paper noted this, but didn’t attempt to implement.
- Openfpm doesn’t note how much SoA layout can improve performance over AoS (Todo), but does show that vectorization is being carried out by compiler.

Arrays of Multiple Types

- One of the biggest advantage of openfpm_data, I’m still not sure how it works.
- Not aware of any other library doing it apart from openfpm.
- Openfpm itself mentions multi-phase verlet list, but the example uses multiple lists. It looks an algorithmic thing and less of an enhanced memory layout technique.
    - And the example only uses single type of data.
    - Needs to look into how this is implemented.

String Literal as template Parameter

- https://stackoverflow.com/questions/54278201/what-is-c20s-string-literal-operator-template
- https://ctrpeach.io/posts/cpp20-string-literal-template-parameters/
-

End

- FDPS
- OpenFPM
  - OpenFPM_data vs Kokkos
- LAMMPS
- AMBER

(Vs Kokkos)

Key Questions

- The distributed vector
  - Can it handle multiple types of particles?
  - If yes, how are data stored in memory? (layout)
  - Other pros and cons vs others
- Nearest neighbor search
- How widely used in biological domain?
  - What is used for that paper?