Performance Portability for HPC Applications through the RAJA abstraction layer

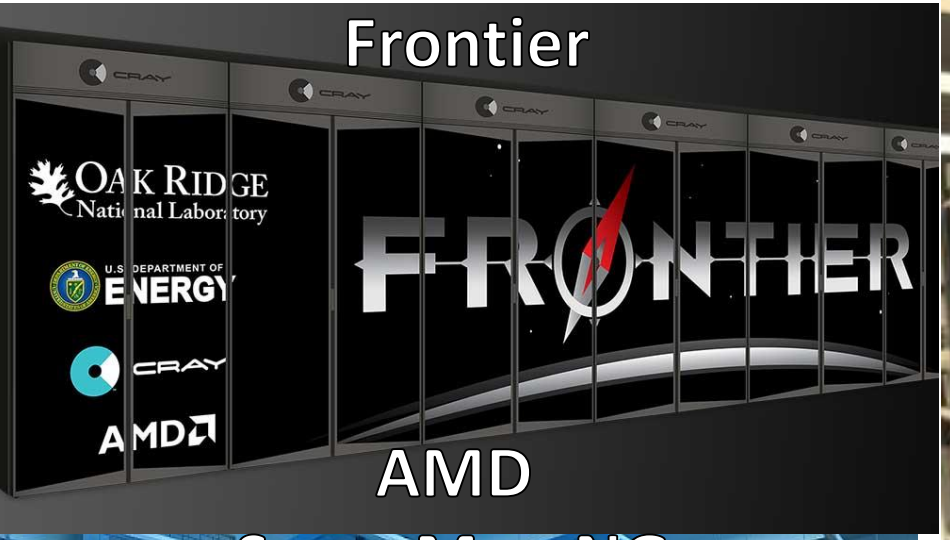
Philipp Pindl

Technical University of Munich

05.07.2022

Munich, Germany

Problem


Frontier
OAK RIDGE
National Laboratory
U.S. DEPARTMENT OF
ENERGY
CRAY
AMD
FRONTIER
AMD

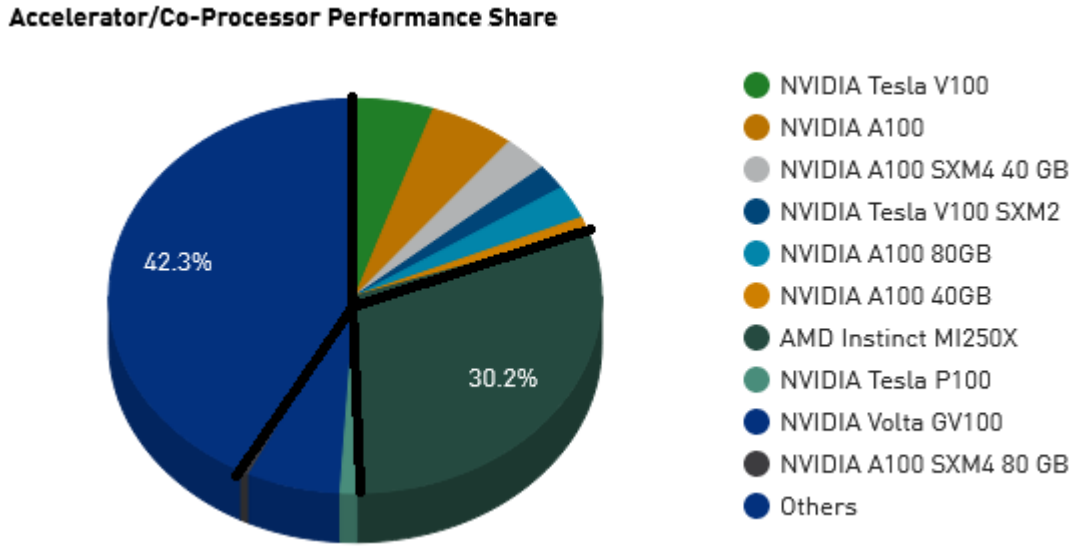

SuperMuc-NG
SUPERMUC-NG
NVIDIA


Perlmutter
AMD +
NVIDIA


Fugaku
Centric Innovation
a Trusted Future
FUJITSU
No GPU

Problem

- Multiple different architectures used



- Multiple programming frameworks

RAJA

- Developed by the Lawrence Livermore National Laboratory (LLNL)
- Used by HPC applications for the Sierra supercomputer at LLNL
- Abstraction layer in the form of a C++ library
- Tries to achieve similar performance on different architectures with a single source code

RAJA - Concept

Different backends

Target Platforms

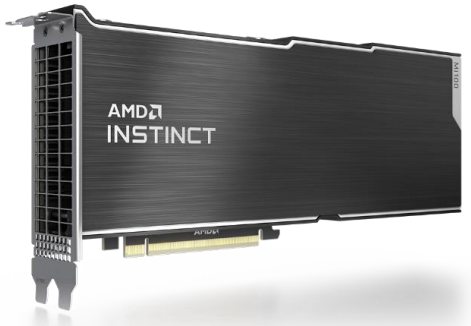
RAJA

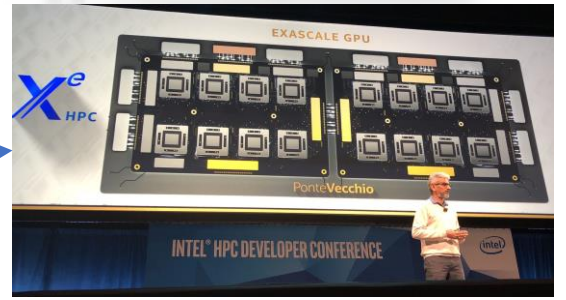
NVIDIA CUDA

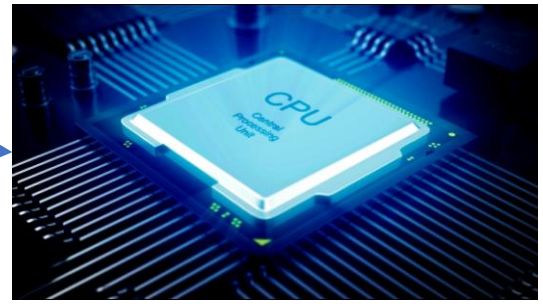
AMD HIP RT

SYCL™

OpenMP®







RAJA - Example

C++ version

```
for (int i = 0; i < N; ++i)
{
        a[i] += c * b[i];
}
```

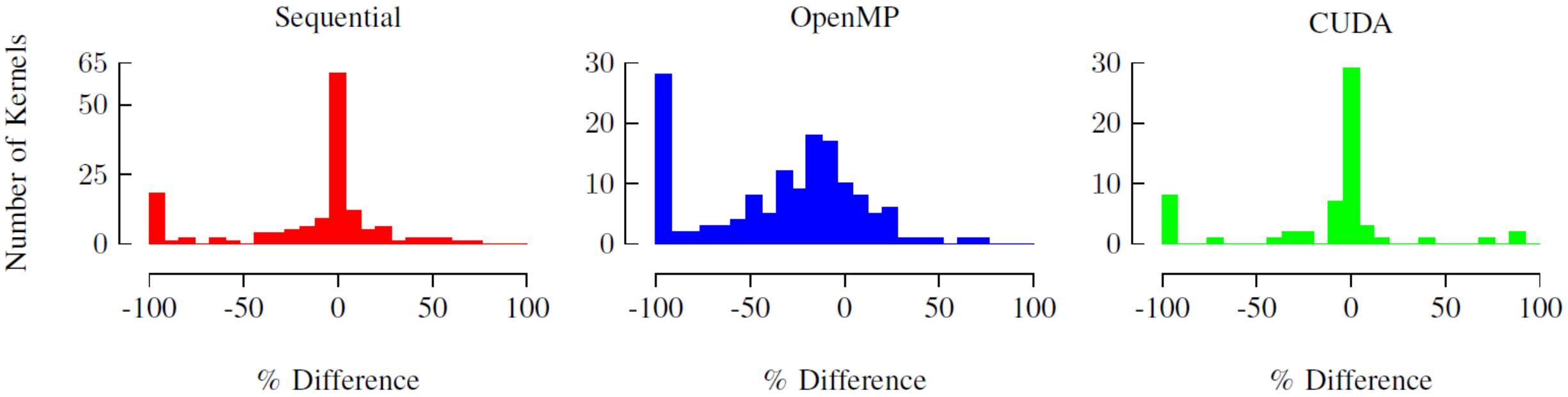
RAJA version

```
using namespace RAJA;
RangeSegment seg (0, N);
forall<loop_exec> (seg, [=] (int i) {
        a[i] += c * b[i];
});
```

- Loop range can be translated to a Iteration Space
- Loop body stays the same
- for loop is expressed by Traversal Template
- Execution Policies specify which backend to use

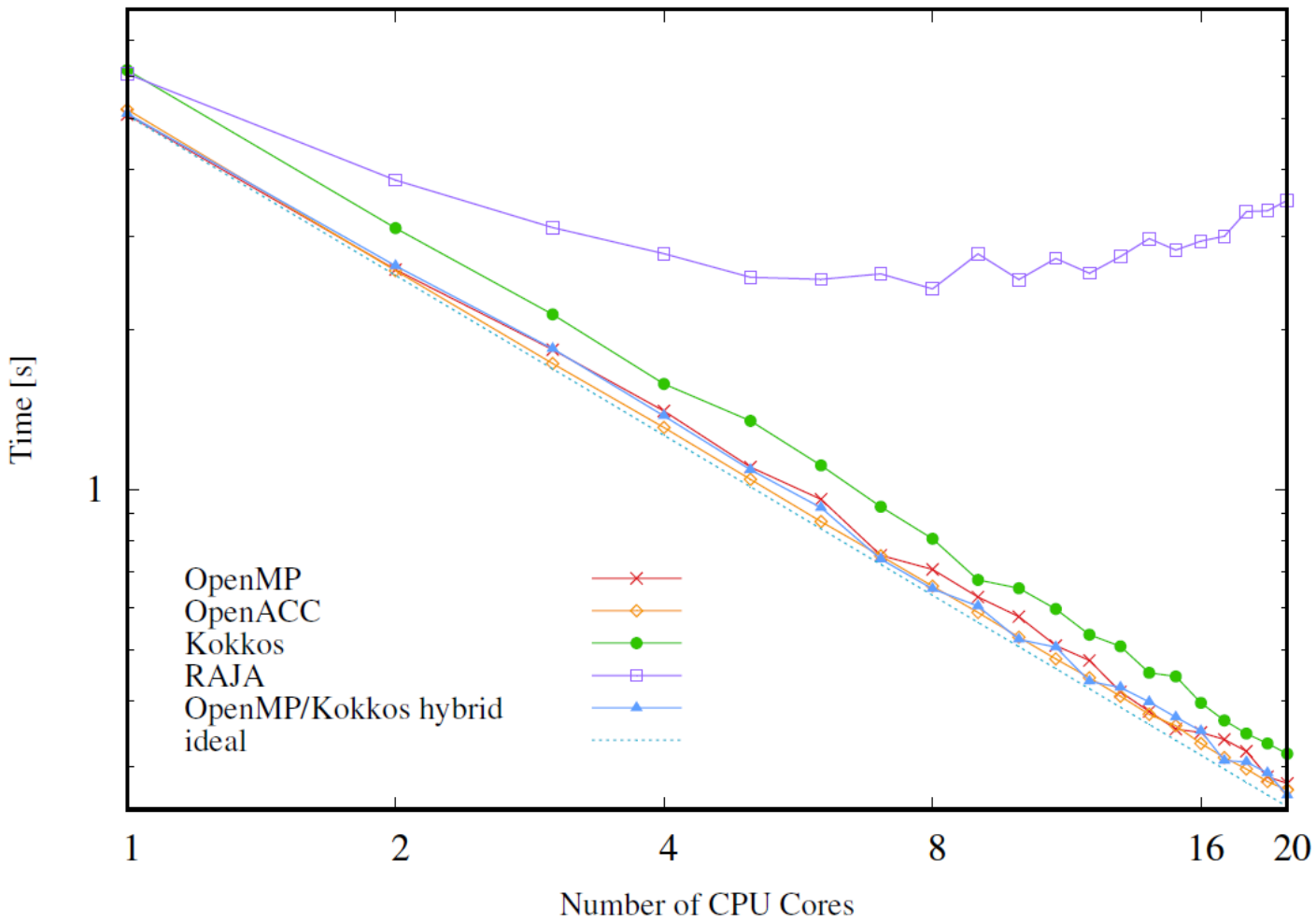
Raja Performance Suite

- Tool for comparing RAJA performance to native implementations



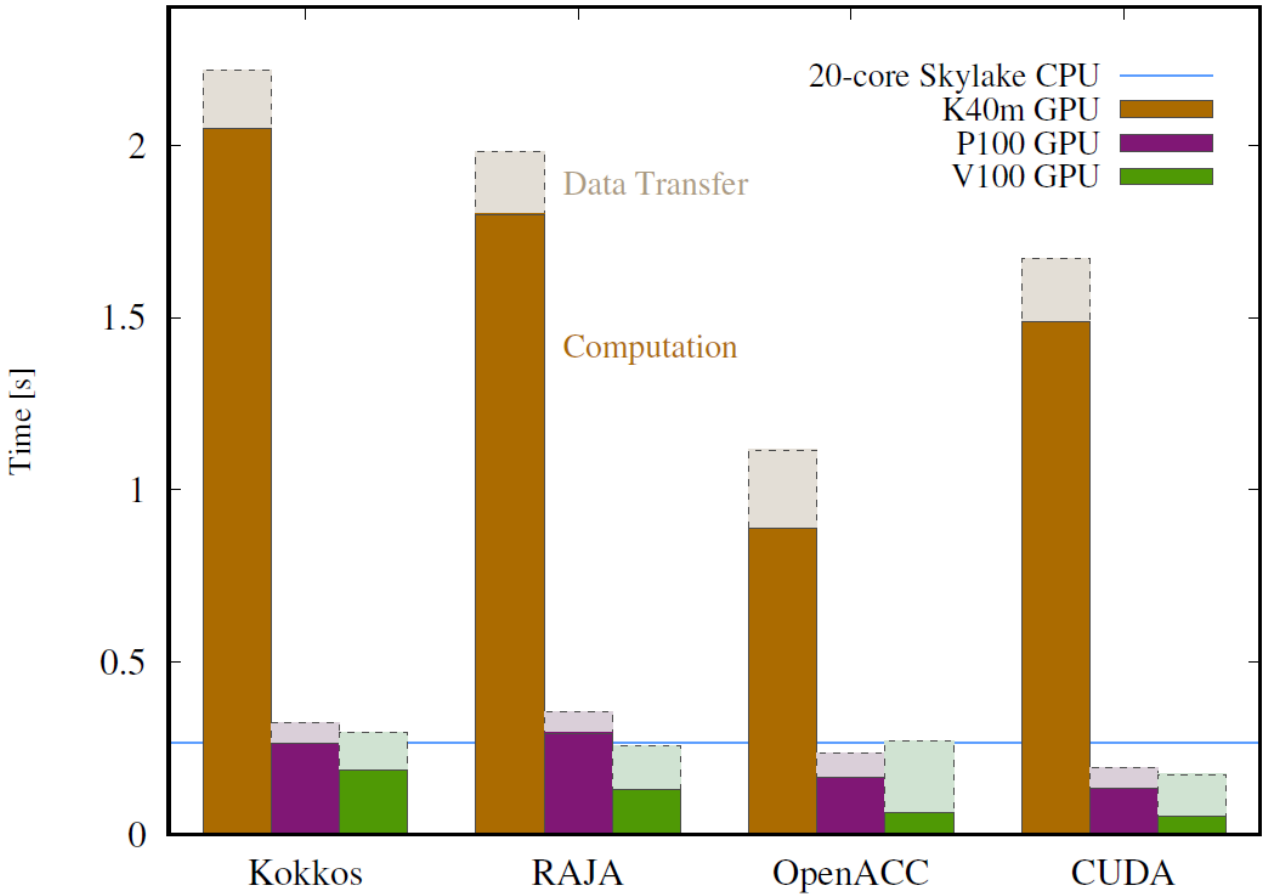
- Spike at 0% mark for all backends
- OpenMP shows a greater variance

PIC–code - CPU



- RAJA does not scale for more CPU cores
- Attributed to false sharing memory issue
- Manual optimization would be necessary to solve problem

PIC–code - GPU



- Data transfer and allocation are excluded in measurements
- Measurements on NVIDIA GPUs
- RAJA implementation was slower than CUDA by about a factor of two
- Significant Overhead may be unacceptable

SW4

GPU	Kernel	AI_k	FR_k (GF/s)	Bound	P_k (GF/s)
Intel Gen9	curvilinear4sg_ci:1296	0.101	4.86	Memory	7.11
NVIDIA A100	curvilinear4sg_ci:1296	3.73	1620	Memory	4700
AMD MI100 (est)	curvilinear4sg_ci:1296	0.774	536.	Memory	692.

Application	Kernel	Intel Gen9	NVIDIA A100	AMD MI100	Perf.Port. Metric(%)	Std.Dev. /Avg	Min /Max
AMR-Wind	MLABec	74.2	79.1	42.8	60.6	0.302	0.541
	MLPoisson	34.9	54.1	8.47	18.2	0.705	0.157
	MLNode	37.7	34.1	34.4	35.3	0.057	0.904
HACC	BarExtras	69.7	45.5	83.5	62.1	0.291	0.544
	Corrections	94.6	36.5	89.1	60.9	0.437	0.385
	DuDt	71.3	49.1	75.8	63.0	0.218	0.648
	Geometry	80.9	55.3	74.1	68.3	0.189	0.683
SW4	curvilinear4sg	68.3	34.5	77.5	53.0	0.377	0.445
RI-MP2	RIMP2$omp	28.4	18.6	4.50	9.64	0.700	0.158
XSBench	XSBench$omp	61.2	32.7	23.1	33.2	0.508	0.377
TestSNAP	FusedDeiDrj	65.1	35.5	9.99	20.9	0.748	0.153
	Ui	38.8	21.7	2.36	6.06	0.871	0.061
	Yi	5.11	27.0	10.5	9.15	0.804	0.189

- Performance was measured as ratio to theoretical peak performance
- RAJA achieves best portability on AMD GPU
- Overall Performance Portability Metric of 53% but only for one kernel
- Compared to Kokkos framework (TestSNAP) it displays much better Performance

Portability

Pro

- Successful compilation for multi-core CPU and GPU architectures
- Easy concept for making C++ code portable

Con

- Necessity of C++ lambdas requires restructuring
- Manual optimization may be required

Performance

- Maximum measured performance overhead of about a factor of two against CUDA
- Different studies find varying performance loss
- Shown to be more performant than the Kokkos framework
- Performance portable on GPU architectures

Summary

- RAJA is a C++ library for performance portability
- Targets diverse GPU and multi-core CPU architectures
- Acceptable performance on GPUs
- Unreliable performance on CPUs

Sources

- https://www.nersc.gov/systems/perlmutter/
- https://www.olcf.ornl.gov/frontier/
- https://upload.wikimedia.org/wikipedia/commons/thumb/c/c6/FugakuSupercomputerSC19.jpg/397px-FugakuSupercomputerSC19.jpg
- https://www.top500.org/statistics/list/
- https://en.wikipedia.org/wiki/CUDA
- https://gpuopen.com/hiprt/
- https://de.wikipedia.org/wiki/OpenCL
- https://computing.llnl.gov/projects/raja-managing-application-portability-next-generation-platforms
- https://de.wikipedia.org/wiki/OpenMP
- https://howto.lintel.in/how-to-find-out-cpu-make-and-model-in-linux/
- https://www.nvidia.com/de-de/data-center/v100/
- https://serverhero.de/wissen/schnellste-gpu