Challenges and Opportunities: Preparing PIConGPU for Frontier

The Center for Accelerated Application Readiness (CAAR) Program at ORNL

Sunita Chandrasekaran

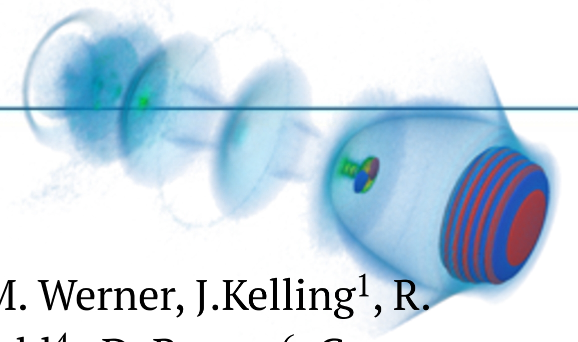
Assistant Professor, Dept. of Computer & Information Sciences

University of Delaware, USA

schandra@udel.edu

OLCF User Group Meeting June 24th, 2021

Application of Interest: PIConGPU



S. Chandrasekaran [2, 3], A. Debus[1], T. Kluge[1], R. Widera[1], S. Bastrakov[1], K. Steiniger[1], M. Garten[1], M. Werner, J.Kelling[1], R. Pausch[1], B. Hernandez[6] , F. Meyer[2,1], M. Leinhauser [2,3], F. Pöschel[2,1], J. Young[2,5], B. Worpitz, A. Huebl[4], D. Rogers[6], G. Juckeland[1], M.Bussmann[2,1]

[1] Helmholtz-Zentrum Dresden-Rossendorf, Dresden, Germany
[2] CASUS, Center for Advanced Systems Understanding, Goerlitz, Germany
[3] University of Delaware, Newark, Delaware, USA
[4] Lawrence Berkeley National Laboratories, Berkeley, CA, USA
[5] Georgia Institute of Technology, Atlanta, GA, USA
[6] Oak Ridge National Laboratory, Knoxville, TN, USA

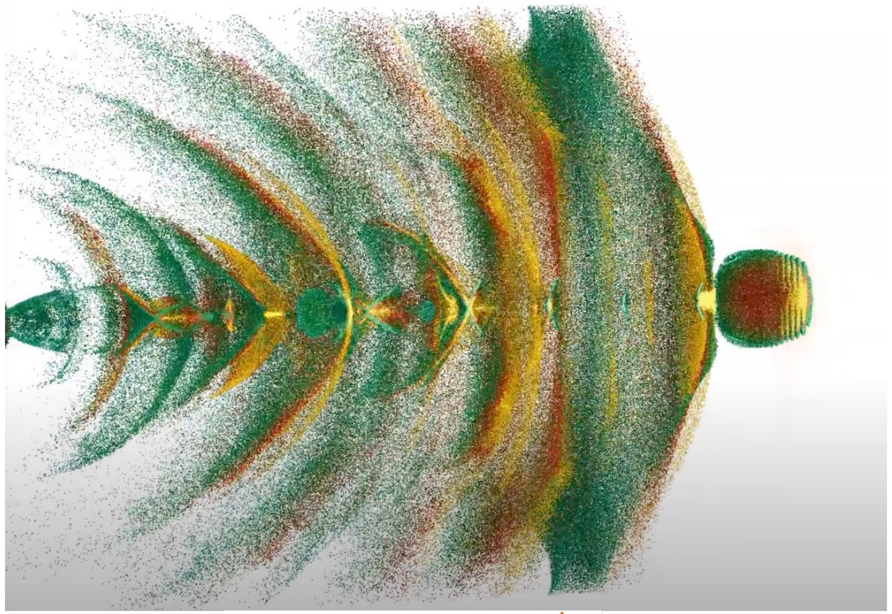
ACK: *This work was partly funded by the **C**enter for **A**dvanced **S**ystems **U**nderstanding (CASUS) which is financed by the German Federal Ministry of Education and Research (BMBF) and by the Saxon Ministry for Science, Art, and Tourism (SMWK) with tax funds on the basis of the budget approved by the Saxon State Parliament."*

ACK: *We would like to acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC).*

ACK: *Thank you very much HPE/AMD Center of Excellence (COE) for your tremendous hardware/software support!*

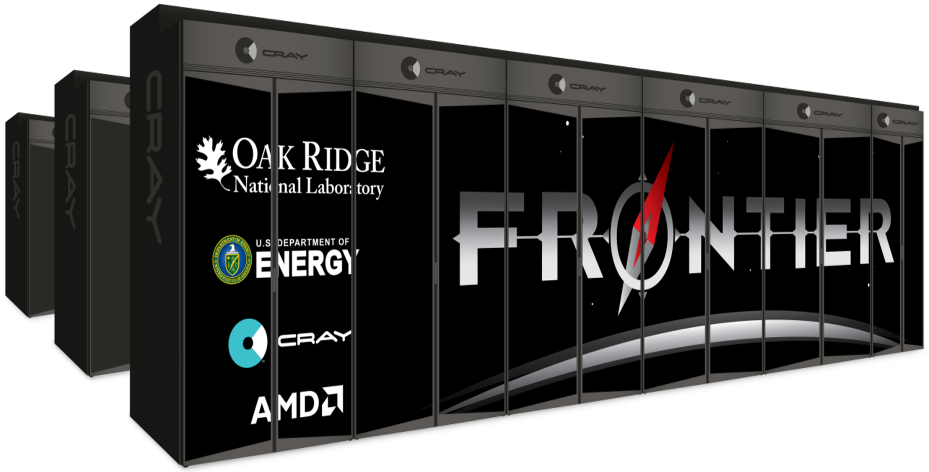
ACK*: This research partially used resources of the Oak Ridge Leadership Computing Facility (OLCF) at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.*

The Center for Accelerated Application Readiness (CAAR) Program at ORNL



~ >= 4 x

vs Summit @ ORNL



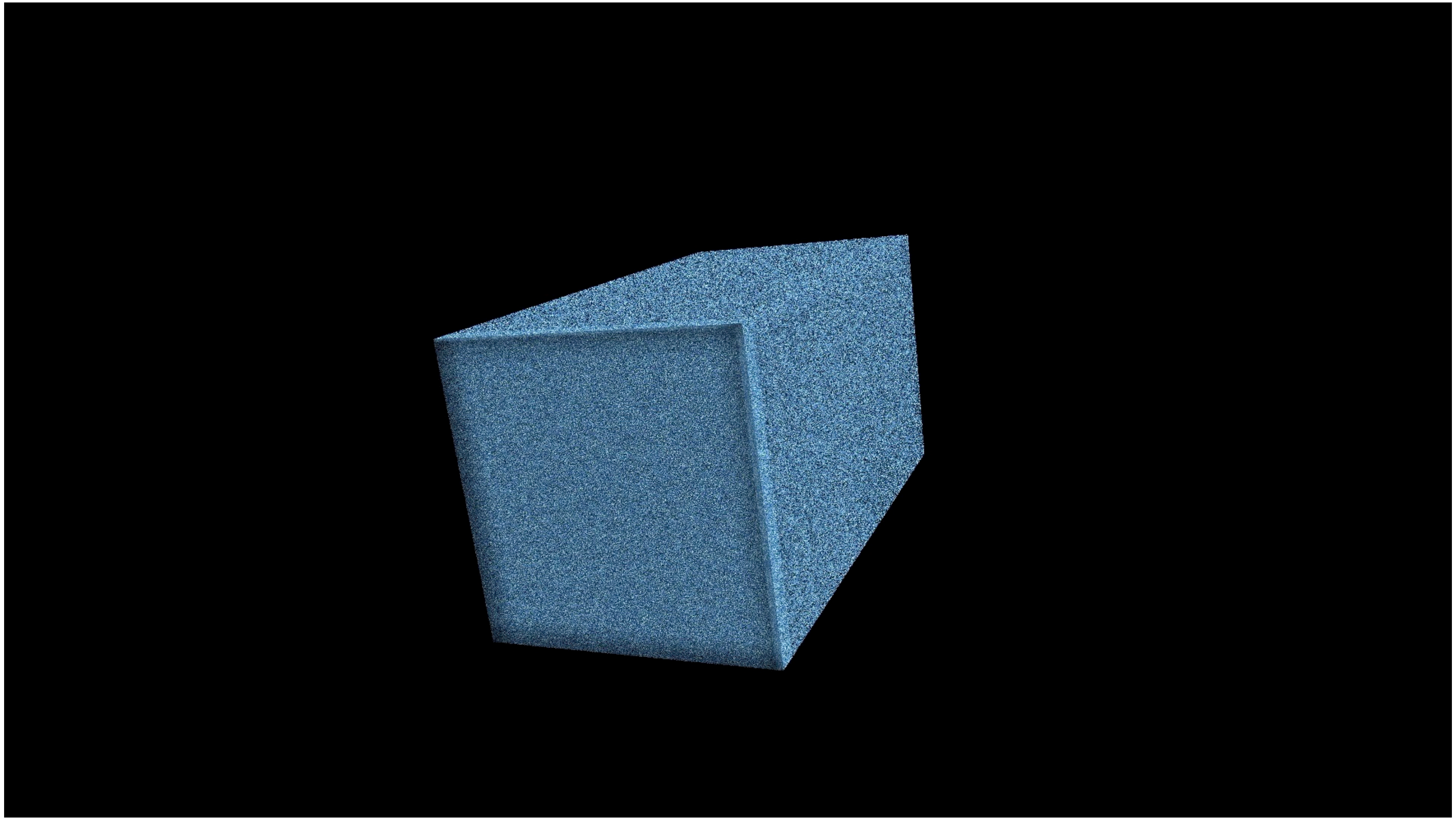
ACK: Felix Meyer, Richard Pausch, HZDR
Still image from an uncut LWFA simulation video using Summit and 48 V100s using ISAAC 1.5

AMD EPYC CPU + AMD Radeon Instinct GPU.

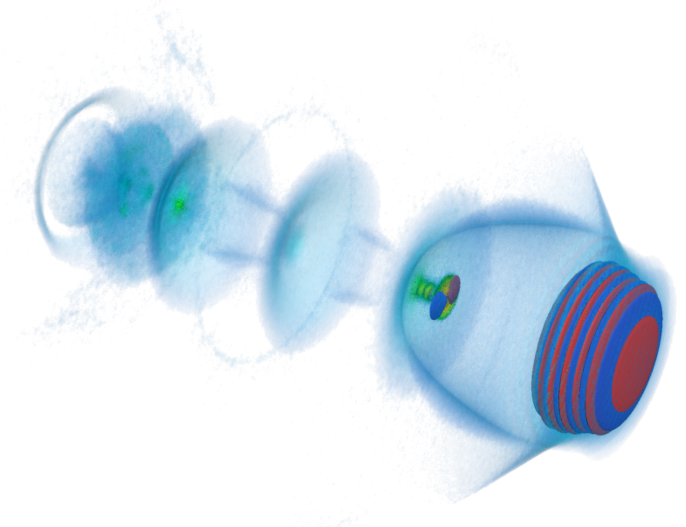
Frontier has an expected peak performance of 1.5 EFlop/s.

What is Particle In Cell on GPU (PIConGPU)

ACK: Vincent Gerber, HZDR, Germany
LWFA, Visualization using ISAAC

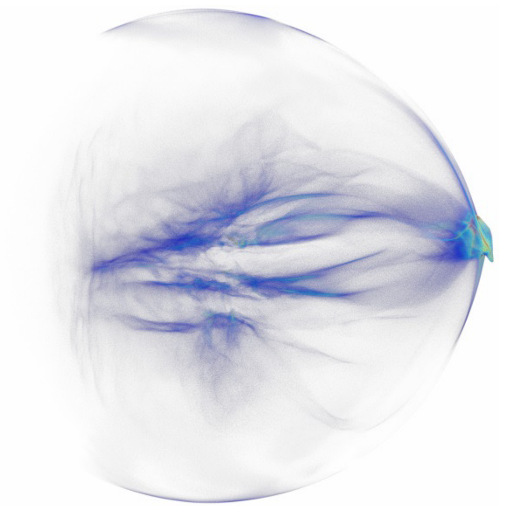


PIConGPU‘s impact on real-world applications



Electron acceleration with lasers

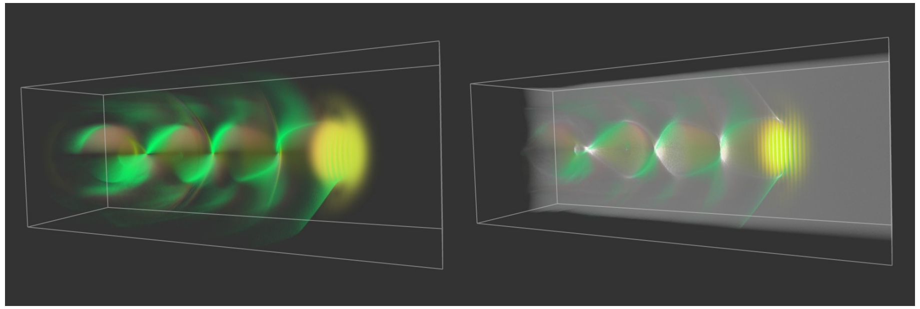
- Compact X-Ray sources of high brightness, e.g. Free-Electron Lasers, to create snapshots of ultrafast processes in material science.
- Extend plasma-based electron accelerators from multi-GeV towards TeV electron energies



Ion acceleration with lasers

- Applications in radiation therapy of cancer.
- Fundamental studies of warm-dense matter and high-energy density physics.

PIConGPU Programmatic Challenges

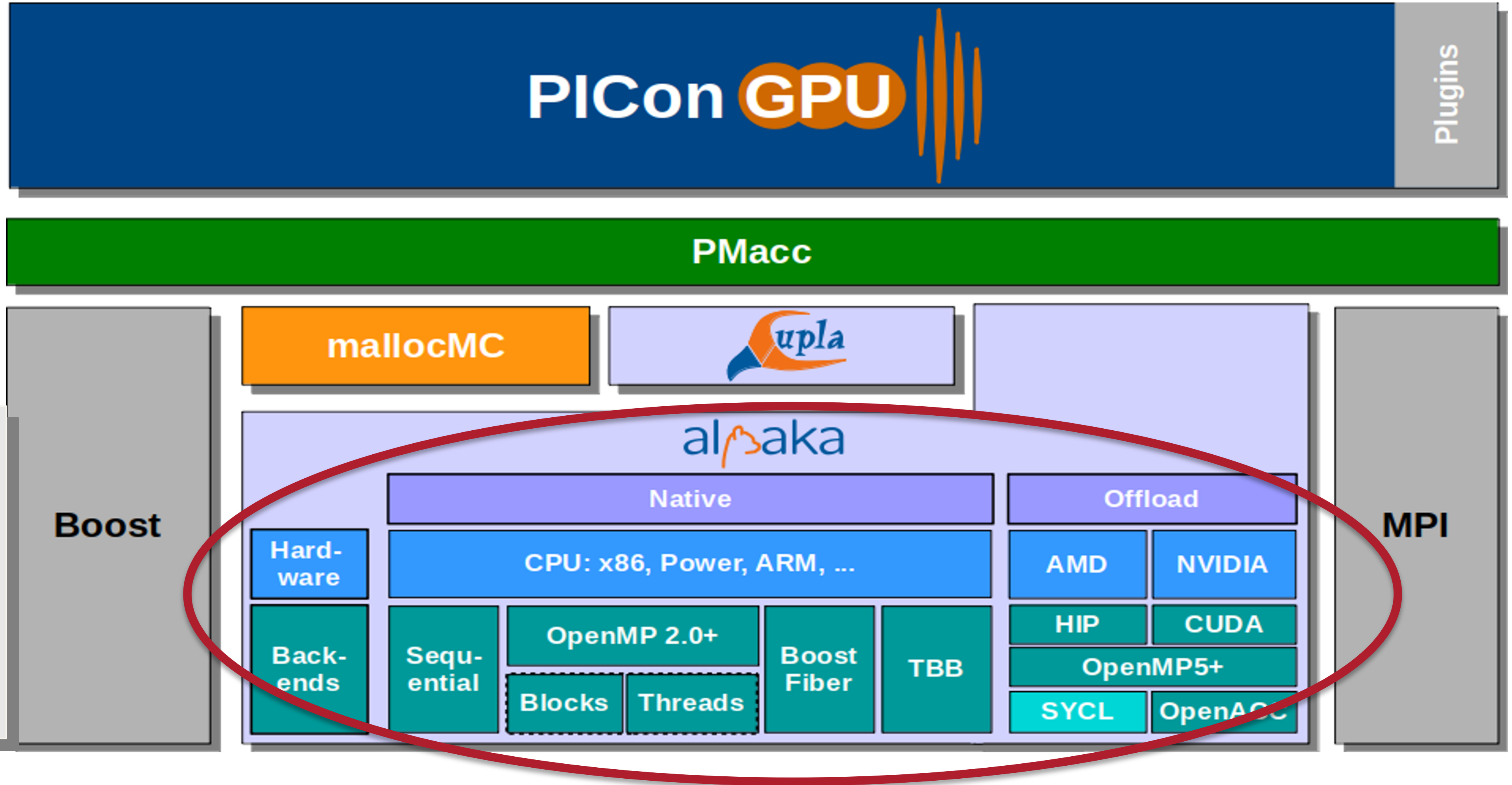


ACK: Benjamin Hernandez, ORNL
LWFA Simulation. Using Summit's 8 nodes (48 V100 GPUs) with ~2 billion particles using ISAAC v1.5.1 running on OLCF's cloud environment (SLATE)

- **Portability:** Run code on different compute architectures (single-source, run everywhere)
- **Performance:** Cannot lose performance while maintaining portability
- **Scalability:** Code profiling & scaling tests to ensure science cases scale to Frontier
- **Visualizations:** Create and develop tools to visualize PIConGPU on the new system
- **Exascale workflows:** Extend I/O capabilities, provide in-situ analysis, data reduction and visualization workflows

PIConGPU Full Software Stack

```
    template< typename T_Acc
>
    ALPAKA_FN_ACC void
operator()(
        T_Acc const & acc,
        // ...
    ) const
    {
        // ...
    }
```



Huebl, Axel, et al. (2018) [Zero Overhead Modern C++ for Mapping to Any Programming Model.]()
Software Stack updated by René Widera (2020)

alpaka software

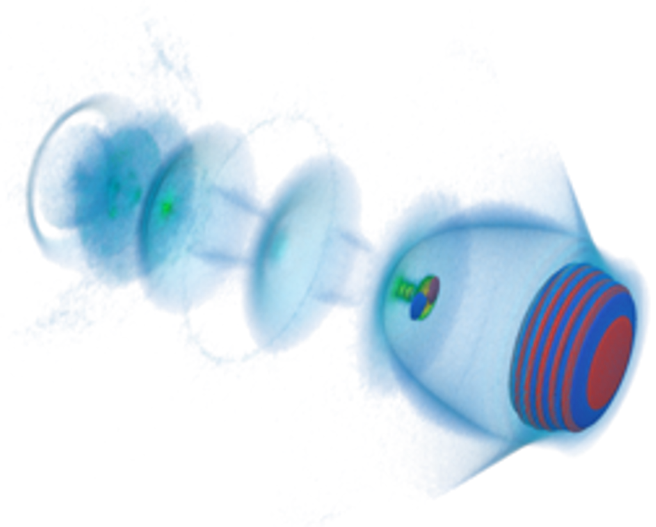
- Open source C++14 header-only library
- alpaka 0.6.0 release - Jan 2021
- New backends: OpenMP 5 target offload and OpenACC
  - This release is adding compatibility to the latest CUDA releases up to 11.2
  - The HIP backend supports HIP 3.5 +
  - Recommendation is to use the latest HIP version
- https://github.com/alpaka-group/alpaka/releases/tag/0.6.0
- Makes kernel performance portability work!

Experimental Setup

- Hardware
  - Summit @ ORNL (IBM POWER 9 CPUs + NVIDIA V100 GPUs)
  - JUWELS @ JSC (AMD EPYC 7402 24-core processor + NVIDIA A100)
  - Spock - AMD/Cray+HPE Early Access System (AMD EPYC AMD EPYC 7662 32-core processor + AMD Instinct MI100
- Software
  - alpaka 0.6.0 (backend OpenMP threading/offloading, OpenACC)
  - NVIDIA CUDA 10.1.243 & 11.0
  - AMD ROCm 4.1.0 & HIP
  - OpenMP Offload compilers and OpenACC

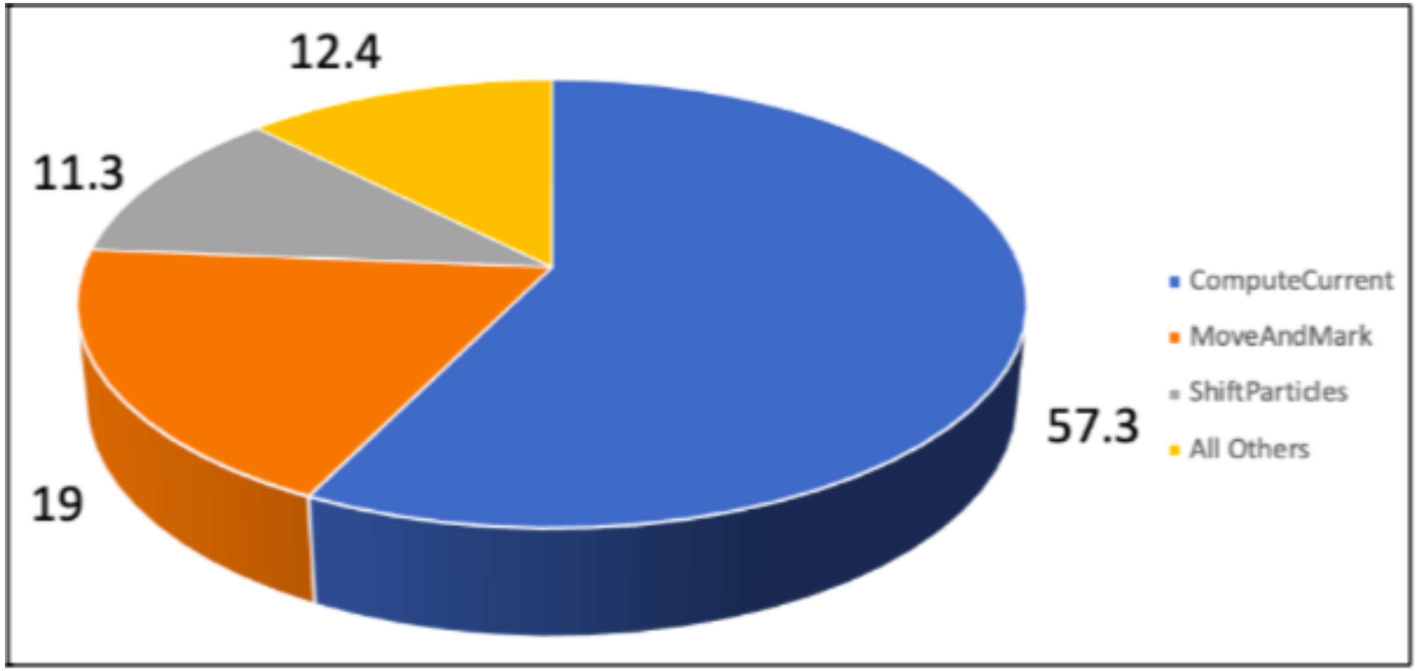
Tools for Profiling and Performance Analysis

- Identifying hot spots in a code (a.k.a. computationally intensive portions in a code)
- Several tools are available including
  - NVIDIA's nvprof, Nsight Compute v2020.3.0, Nsight Systems v2021.1.1
  - AMD's rocProf
- Benchmarks
  - Gpumembench
  - BabelStream



NVIDIA's Nsight Systems tool

Visualization timeline

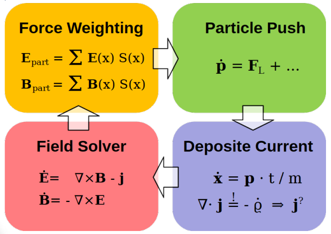


Execution time (%) for different kernels within PIConGPU's Traveling-wave electron acceleration (TWEAC) science case.

The MoveAndMark and ComputeCurrent kernels take up over 75% of the overall runtime

Metric: Figure of Merit (FOM) for CAAR- PIConGPU

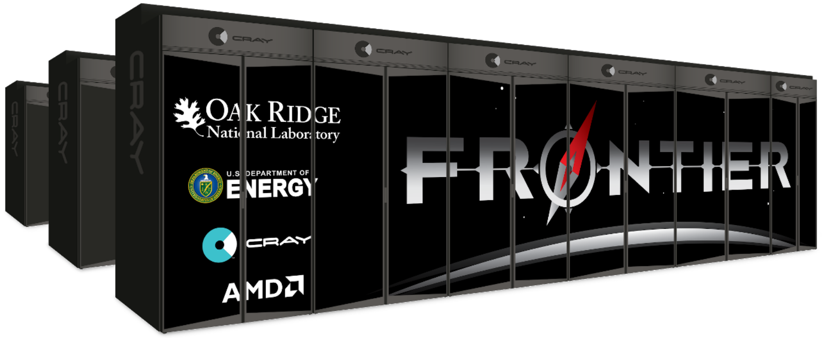
- Weighted sum of the total number of particle updates per second (90%) and the number of cell updates per second (10%).



- Taken as an average over a representative number of time steps

$$\text{FOM} = \frac{(\,90\%\ \text{x particle updates} + 10\%\ \text{x cell updates}\,)}{\text{second}}$$

~ >= 4 x

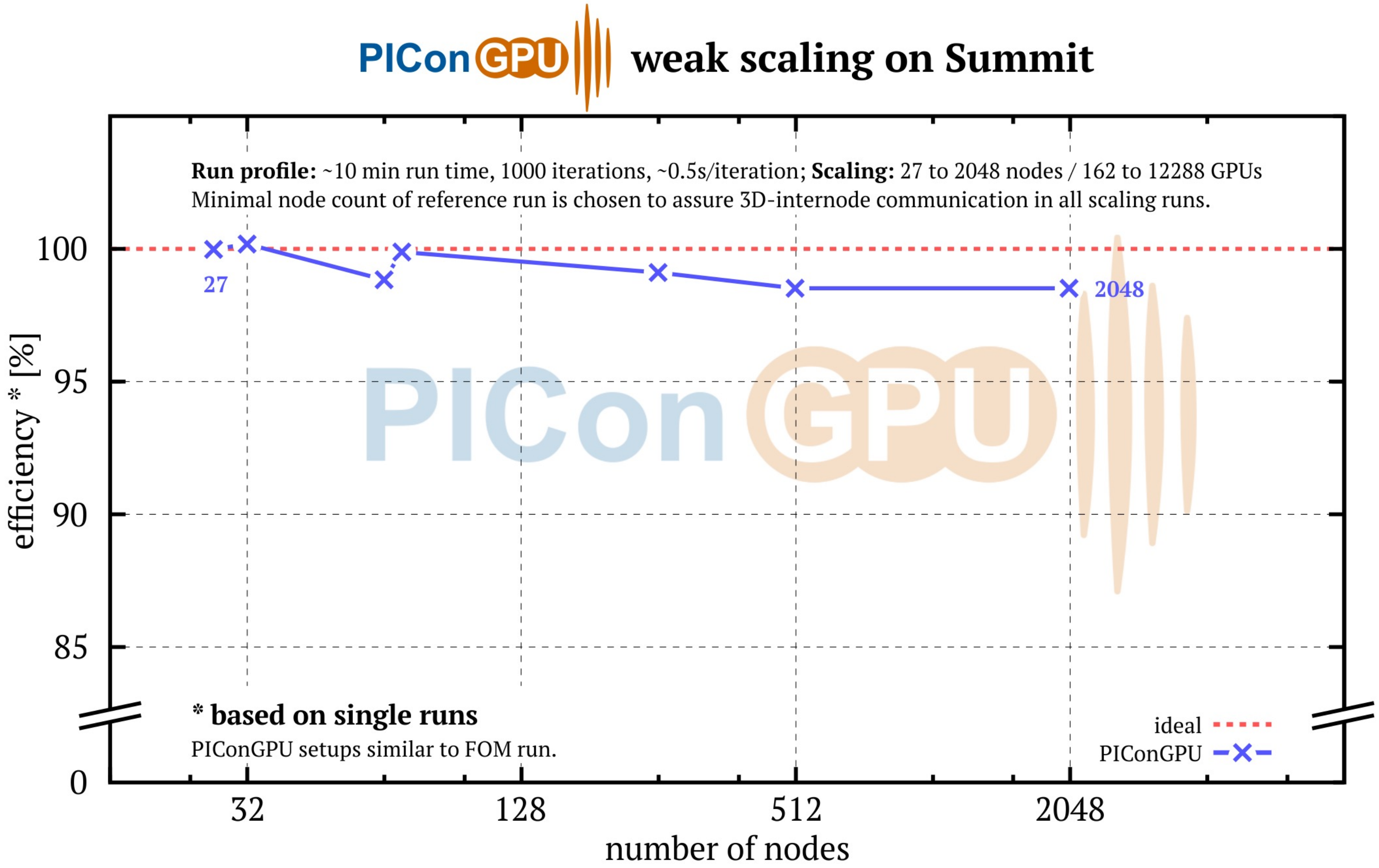


NEW TWEAC (June 2021)

Weak Scaling - FOM on ½ Summit @ ORNL

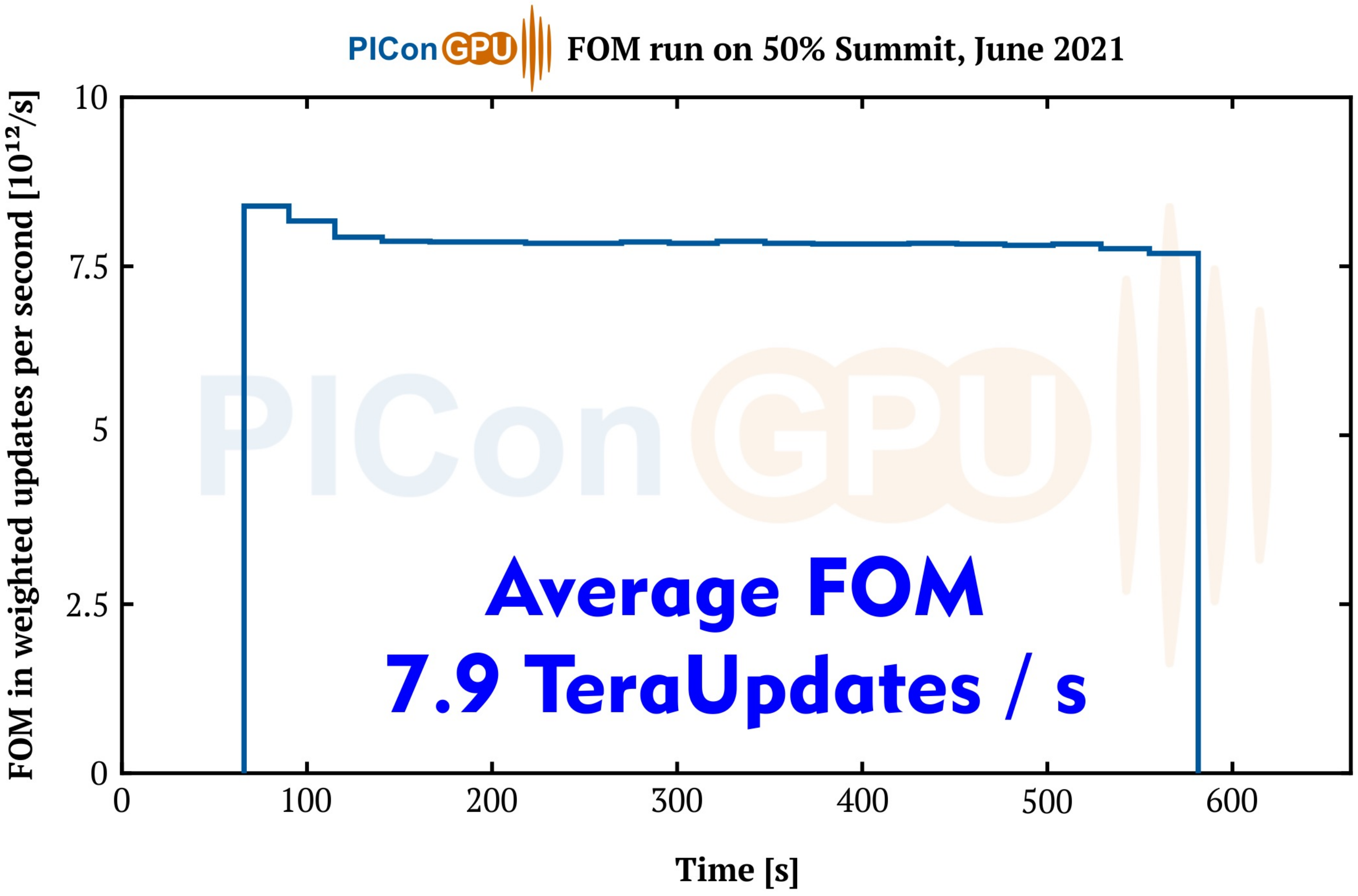
Experimental Setup:

- № Iterations: 1000
- Runtime: ~10 mins
  ~ 0.5 secs per iteration
- FOM Science case
- Scaling:
  - 1 nodes → 2300 nodes
  - 6 GPUs → 12288 GPUs
  - 98-99% GPU utilization



NEW TWEAC (June 2021)

Scaling - FOM on ½ Summit @ ORNL – June 2021



FOM = (0.9 × particle updates + 0.1 × cell updates) / second

PIConGPU

# timesteps	1000
# GPUs	12288
# cells total	$179.\ 10^9$
# cells per GPU	$14.6 \cdot 10^6$
# particles total	$4.4\cdot 10^{12}$
# particles per GPU	$365\cdot 10^6$

NEW Vs OLD TWEAC

- Nov 2019 runs fetched us **6.82 TUP/s** VS June 2021 runs fetched us **7.88 TUP/s** (half of Summit runs)
- So how did that happen? 🤔
  - The ability to model the physics accurately *(but more computations)* for longer iteration simulations has improved
  - A faster *(compensating for more computations)* and numerically stable version of the background TWEAC laser field
  - A new AOFDTD field solver has been implemented for better numerical dispersion properties
  - CurrentInterpolation filtering step dropped – improves FOM a tad bit

NEW TWEAC (June 2021), FOM run on SUMMIT and JUWELS

Execution time: Lower the better
GIPS and Instruction intensity: Higher the better

Move and Mark Kernel

GPU	Summit V100	JSC JUWELS A100
Execution Time (s)	0.089	0.062
GIPS	6.494	9.290
Instruction Intensity (insts/transaction)	0.839	0.860
Achieved FP32 (TFLOPS)	4.7	6.044
Achieved FP64 (TFLOPS)	0.633	0.812

Compute Current Kernel

GPU	Summit V100	JSC JUWELS A100
Execution Time (s)	0.204	0.165
GIPS	7.803	9.588
Instruction Intensity (insts/transaction)	4.183	4.260
Achieved FP32 (TFLOPS)	3.222	3.922
~~Achieved FP64 (TFLOPS)~~		

Takeaway – Summit (V100) and JUWELS (A100)

- MoveAndMark kernel is memory-bound for FP64 and compute-bound for FP32
  - On JUWELS - single precision achieved FLOPS is 40 % of peak theoretical FLOPS, but greater achieved FLOPS when compared to Summit V100
  - On JUWELS – double precision achieved FLOPS is 11% of peak theoretical FLOPS, but greater achieved FLOPS when compared to Summit V100
- ComputeCurrent kernel is compute-bound for FP32
  - On JUWELS - single precision achieved FLOPS is 26 % of peak theoretical FLOPS, but greater achieved FLOPS when compared to Summit V100
- GIPS increases due to faster runtime
- Increase in instructions issued naturally leads to an increase in the instruction intensity

Roofline plots and preliminary performance on the AMD/Cray+HPE Spock system

Instruction Roofline for AMD GPUs

- Instruction Roofline formula revised from Williams et. al

$$GIPS_{peak} = CU \times WFS/CU \times IPC \times frequency$$

$$GIPS_{achieved} = \frac{\frac{instructions}{64}}{1 \times 10^9 \times runtime}$$

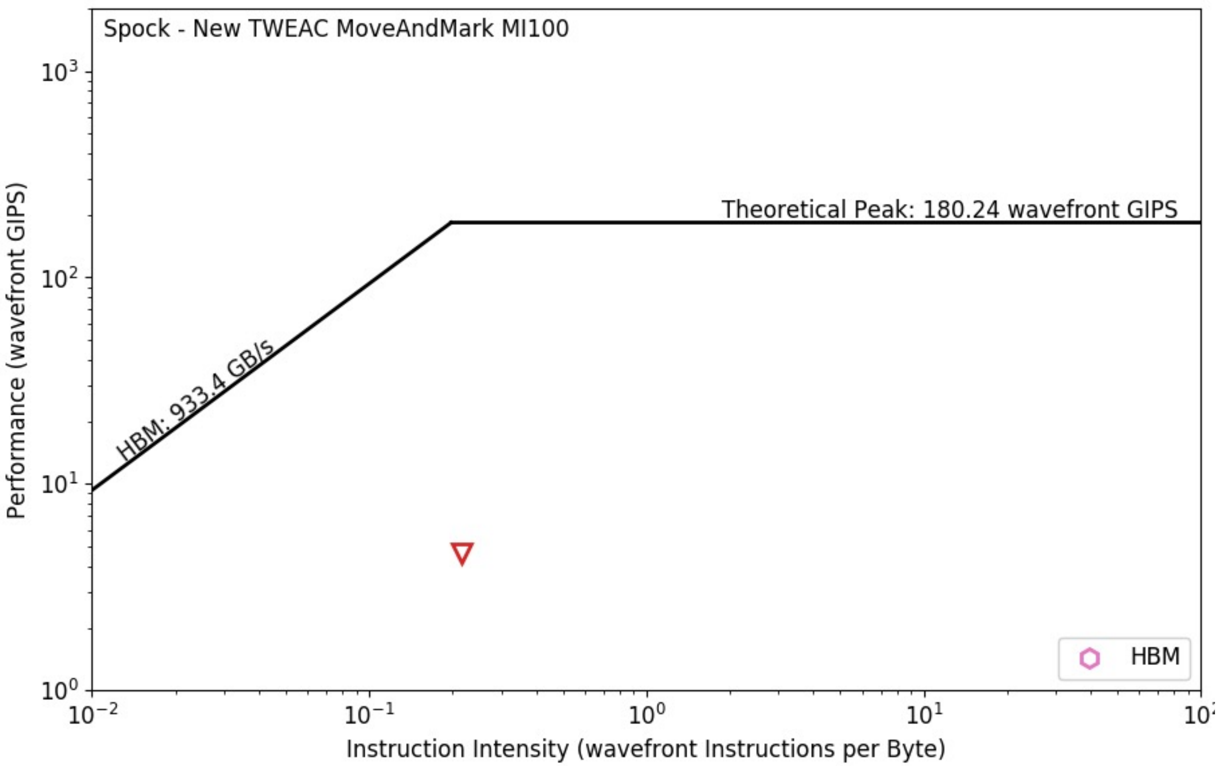
$$InstructionIntensity = \frac{\frac{instructions}{64}}{(bytes\ read + bytes\ written) \times runtime}$$

CU: compute unit
WFS: wavefront schedulers
IPC: Instructions per cycle

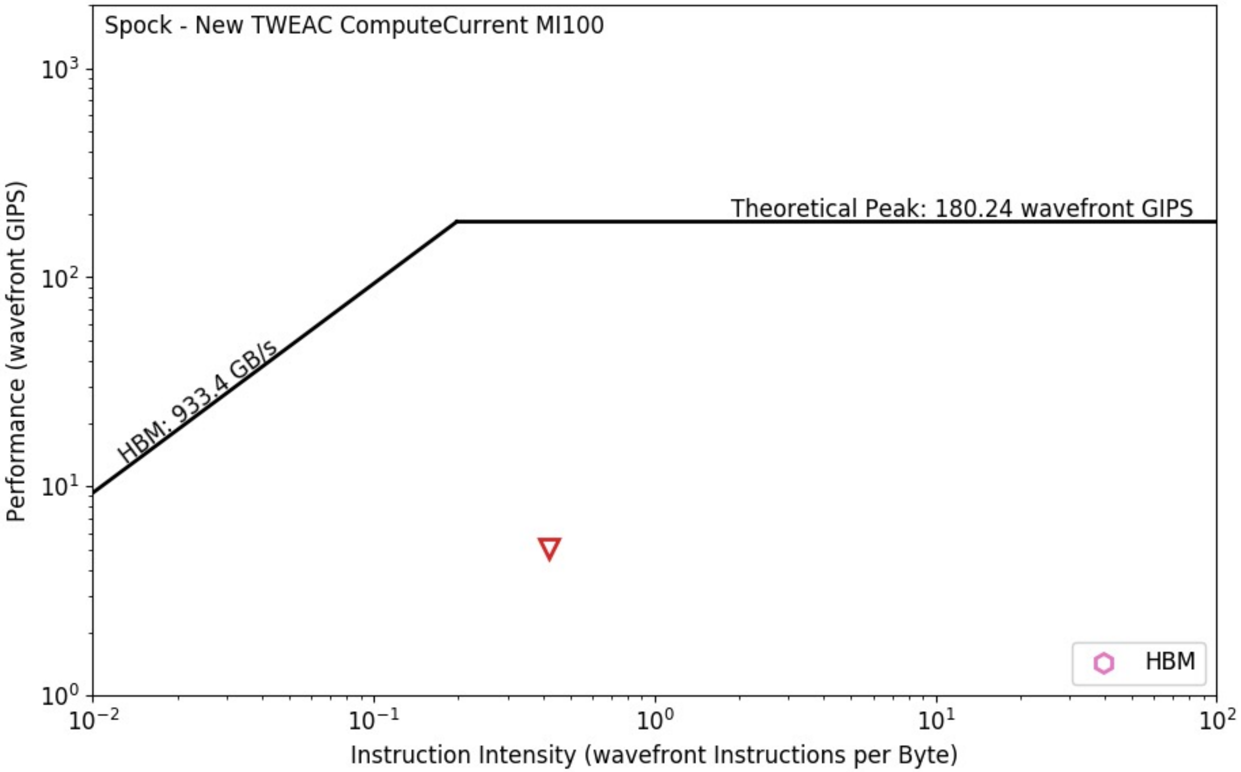
- Used Vector/Scalar-ALU instruction counters from rocProf
  - SQ_INSTS_VALU vs SQ_INSTS_SALU counters

Instruction Roofline for AMD MI100 GPUs

Move and Mark Kernel



Compute Current Kernel



NEW TWEAC (June 2021) FOM run

Execution time: Lower the better
GIPS and Instruction intensity: Higher the better

Move and Mark Kernel

GPU	V100	MI100
Execution Time (s)	0.089	0.098
GIPS	6.494	4.633
Instruction Intensity (insts/byte)	0.029	0.217
FP32 (TFLOPS)	4.70	-
FP64 (TFLOPS)	0.631	-

Compute Current Kernel

GPU	V100	MI100
Execution Time (s)	0.204	0.208
GIPS	7.803	5.033
Instruction Intensity (insts/byte)	0.140	0.421
FP32 (TFLOPS)	3.22	-
~~FP64 (TFLOPS)~~		-

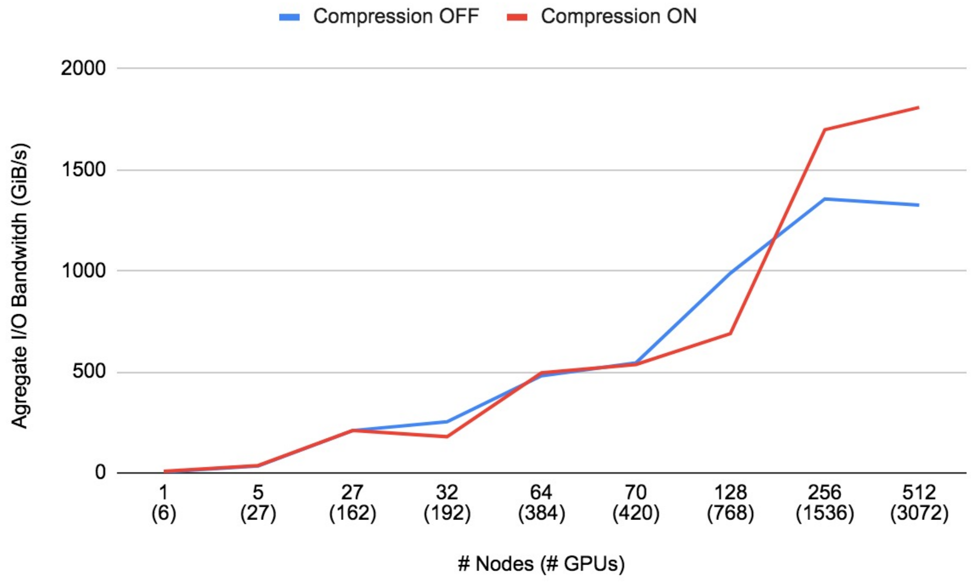
Takeaway – AMD MI100s

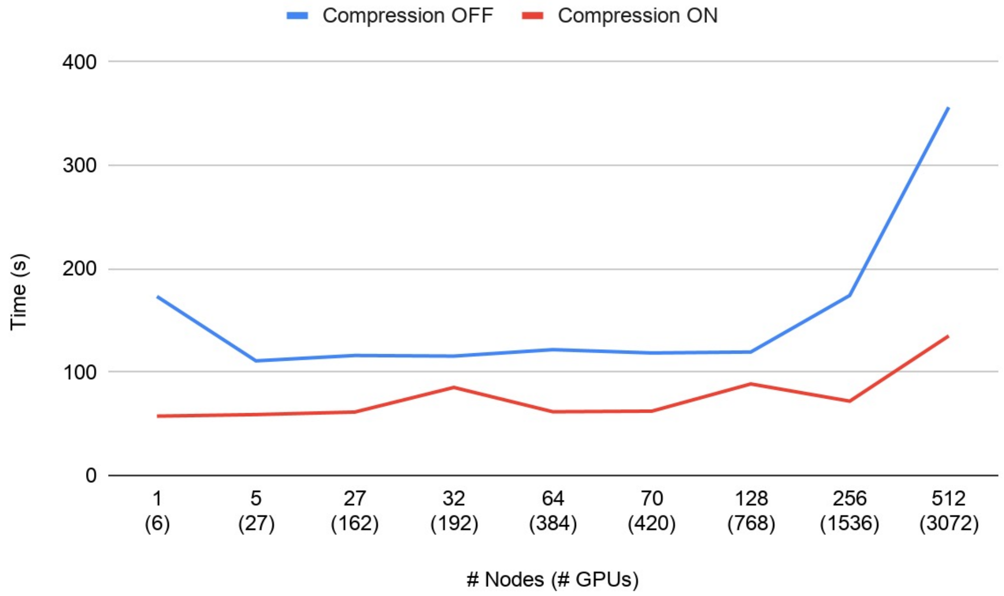
- Execution time for the V100s and the MI100s are neck-to-neck 
- GIPS is higher for the V100s compared to MI100 
- Instruction/byte higher for the MI100s compared to the V100s
  - Depends on the number of bytes fetched from/to GPU memory
  - (note: on the NVIDIA GPUs, one would usually measure instruction/transaction, so those numbers were converted to instructions/byte, just fyi)

Offloading status – PIConGPU

- OpenMP offload and PIConGPU
  - Clang (and AOMP) offload to x86_64 works so far
    - AOMP target offload – bugs, work in progress
  - With Cray CCE omp offload there is a linker error
    - HPE helping fix; work in progress
- OpenACC and GPU
  - NVHPC to GPUs gives a compiler (and/or runtime error)
  - NVHPC 21.1 to x86_64 works

PIConGPU I/O – Summit & Spock





Memory utilization at node level Summit

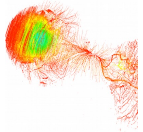
Data Preparation Strategy	GPUs	Total GPU Memory used (GB)	Total RAM used (GB)	Total RAM used during I/O (GB)
Double buffer	6	96	≈ 210	≈ 490
Double buffer	4	64	≈ 146	≈ 335
Mapped memory	4	64	≈ 98	≈ 232*

*Under Spock's RAM limit

Some I/O numbers on Spock

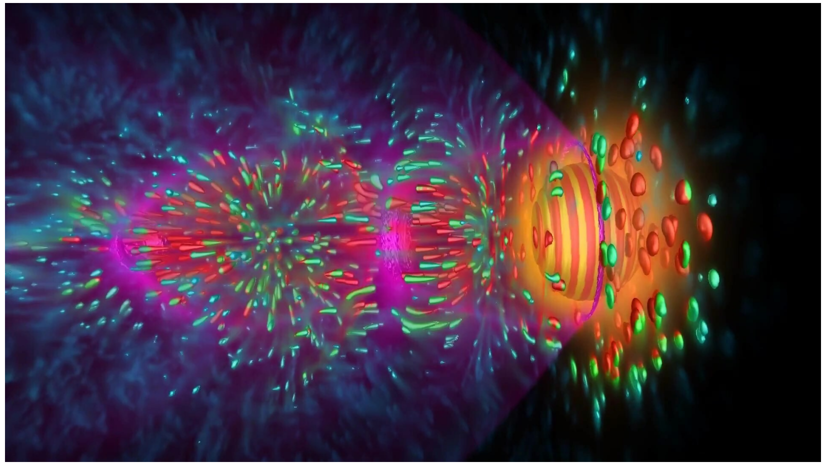
Data Preparation Strategy	GPUs	Total GPU Memory used (GB)*	Total RAM used per node (GB)	Total RAM used during I/O per node (GB)	Runtime (s) Compression OFF	Runtime (s) Compression ON (BLOSC)**
Mapped memory	4	64	≈ 30	≈ 212	1977.753	1915.116
Mapped memory	16	256	≈ 30	≈ 212	1918.01	1910.85

Summary

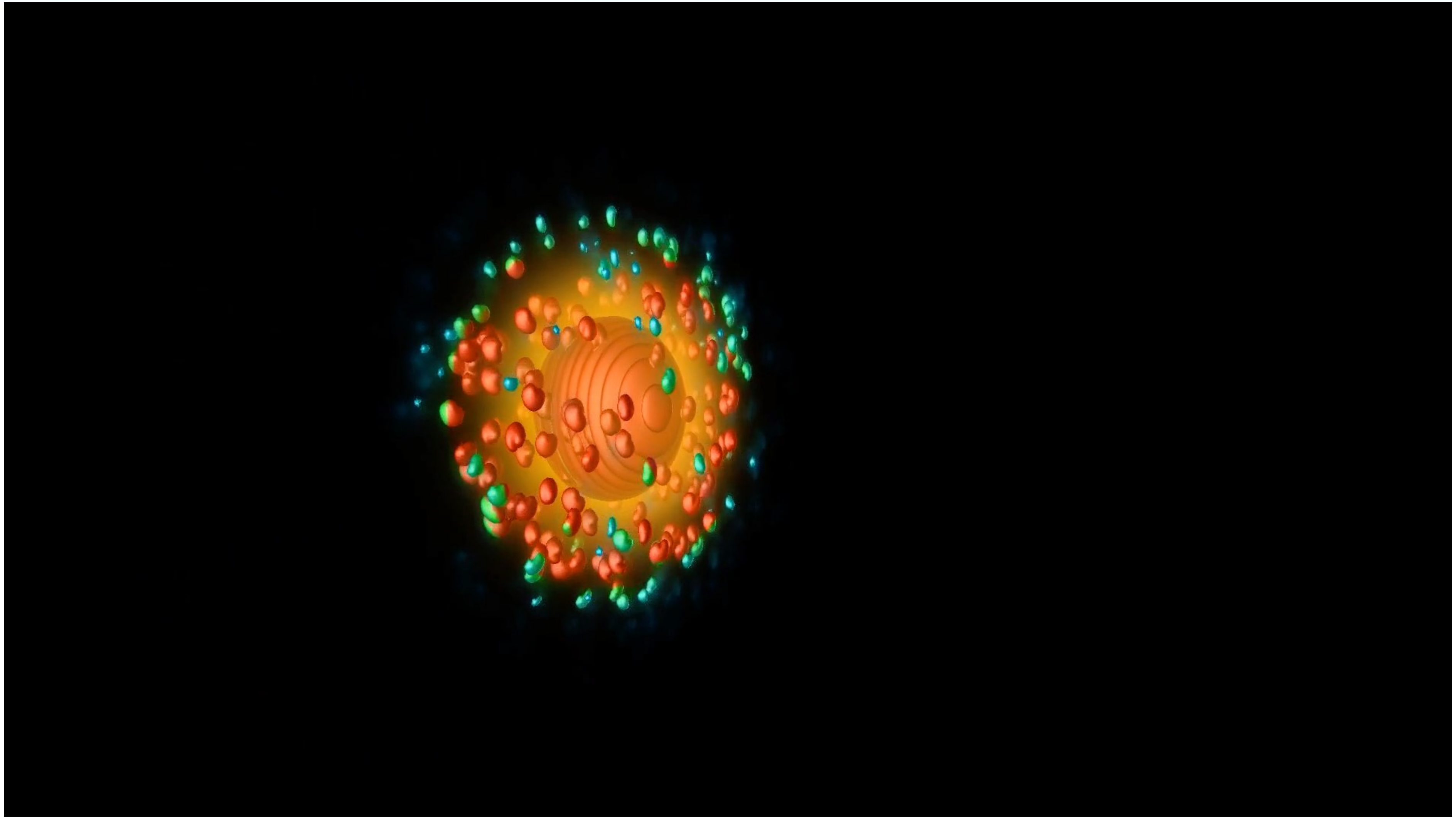


- A100 shows greater FLOP performance over V100
- Acknowledging A100 is not similar to MI100 ;-)
- MI100 is neck-to-neck with V100 for execution time
- Looking forward to using enhanced performance and analysis tools on Frontier
- Need directive-based programming models compiling/executing
- Need increased memory ratio between main and GPU memory on Frontier to tackle I/
- Need tools like ISAAC in-situ library & facilities on Frontier
- Looking forward to pushing Frontier boundaries with PIConGPU case studies

Open Source software(s)



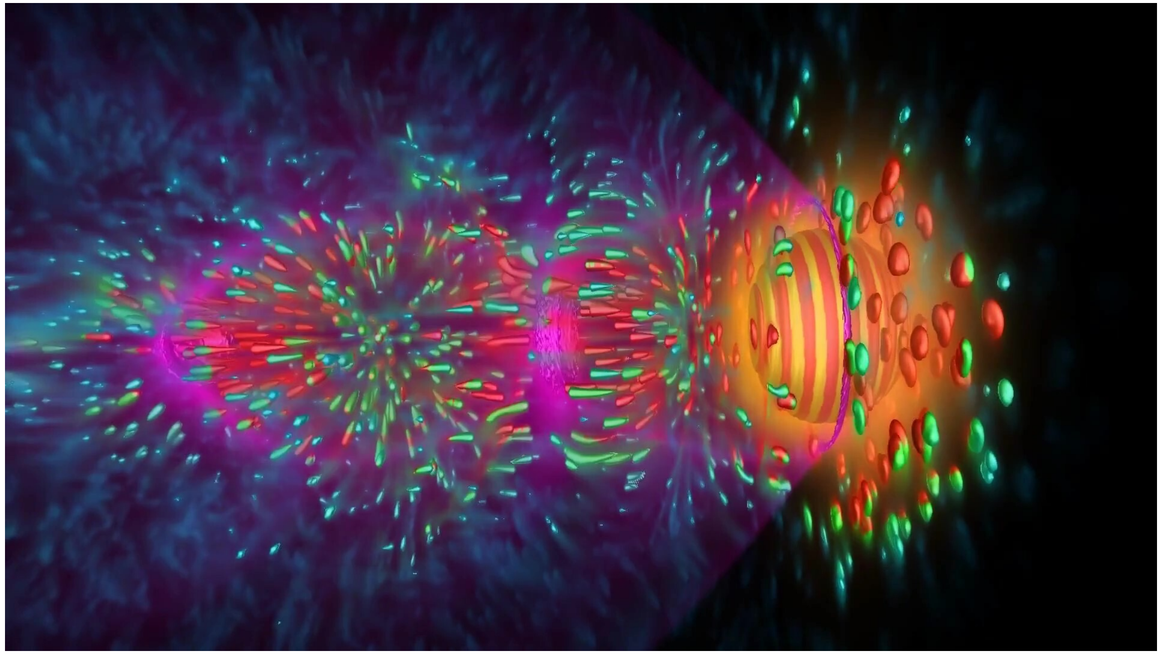
Credit: Felix Meyer, Music: Richard Pausch
Real-Time Vector Field Visualization test using HZDR Hemera Cluster with 4 NVIDIA V100.



Credit: Felix Meyer, Music: Richard Pausch
Real-Time Vector Field Visualization test using HZDR Hemera Cluster with 4 NVIDIA V100.
The video highlights the LWFA Simulation - (Laser Wakefield Accelerator) of PIConGPU visualized with in-situ visualization library ISAAC.

Nicholas Malaya, Tim Mattox, Luke Roskop, Adam Lavely, Noah Wolfe, Noah Reddell and team for your tremendous support!!!

Looking forward to our continued collaborations! :-)



Credit: Felix Meyer, Music: Richard Pausch
Real-Time Vector Field Visualization test using HZDR Hemera Cluster with 4 NVIDIA V100.
The video highlights the LWFA Simulation - (Laser Wakefield Accelerator) of PIConGPU visualized with in-situ visualization library ISAAC.

GitHub is our Social Network

***ACK:** This work was partly funded by the **C**enter for **A**dvanced **S**ystems **U**nderstanding (CASUS) which is financed by the German Federal Ministry of Education and Research (BMBF) and by the Saxon Ministry for Science, Art, and Tourism (SMWK) with tax funds on the basis of the budget approved by the Saxon State Parliament."*

***ACK:** Thank you very much HPE Cray/AMD Center of Excellence (COE) for your tremendous hardware/software support!*

***ACK:** This research **partially** used resources of the **O**ak Ridge **L**eadership **C**omputing **F**acility (**OLCF**) at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.*