TECHNION
Israel Institute
of Technology

oneAPI

Israel oneAPI CoE @ Technion

Gal Oren | galoren@cs.technion.ac.il

oneAPI CoE

הטכניון מכון טכנולוגי לישראל TECHNION Israel Institute of Technology

oneAPI Centers of Excellence

Academic Centers of Excellence

oneAPI Centers of Excellence

"By openness, we mean to deliver productivity and support innovation in an open, collaborative way that benefits the entire ecosystem."
— Pat Gelsinger, CEO, Intel

oneAPI Centers of Excellence contribute to open accelerated computing, propelling the next generation of innovation with open standards, collaboration, and support as part of this ecosystem. Led by top influencers in academia and industry, a oneAPI Center of Excellence delivers the acceleration and adoption of oneAPI, enabling open source code bases, curriculum development, and furthering the oneAPI ecosystem initiative. Primarily, they drive innovation for open, standards-based, cross-architecture, unified programming models.

Explore the world's most prestigious universities and organizations that are now part of the oneAPI academic community and centers of excellence.

Technion oneAPI CoE

The CoE



Technion University - Israel Institute of Technology

Professor Hagit Attiya is an Israeli computer scientist who holds the Harry W. Labov and Charlotte Ullman Labov Academic Chair of Computer Science at Technion.

Dr. Gal Oren is a visiting scientist in the Computer Science department at Technion and a senior researcher in the Scientific Computing Center at the Negev Nuclear Research Center.

Professor Danny Hendler is a faculty member in the department of Computer Science at Ben-Gurion University.

This center will facilitate classroom teaching in contemporary scientific computing using the power of CPUs, GPUs, and other accelerators with oneAPI. Dr. Gal Oren and Professor Hagit Attiya from the Technion Computer Science Department will work in collaboration with Professor Danny Hendler from the Computer Science Department of Ben-Gurion University in teaching oneAPI in classrooms at their universities.

Press Release

Project

Educator Program

Success Stories

"I am excited to establish a new oneAPI OpenMP* training program with Intel. As heterogeneous supercomputers worldwide are on the rise, and diverse high-performance computing is practically ubiquitous, there is a need to raise a new generation of developers who can push legacy and new-generation applications performance to the limit. With oneAPI, we can close the gap between software and hardware and exploit the full potential."

—Gal Oren, Technion, Israel Institute of Technology

Student

Re'em Harel, Ben-Gurion University

Re'em develops and integrates scientific applications in HPC systems using MPI+X paradigms, focusing on MPI and OpenMP* on multicore and heterogeneous architectures. He is also developing a scalable framework benchmark named ScaleSALE and implementing OpenMP schemas to numerical schemas. In addition, Re'em is researching AI techniques for OpenMP directives that can be incorporated in the Intel® Advisor.

Project

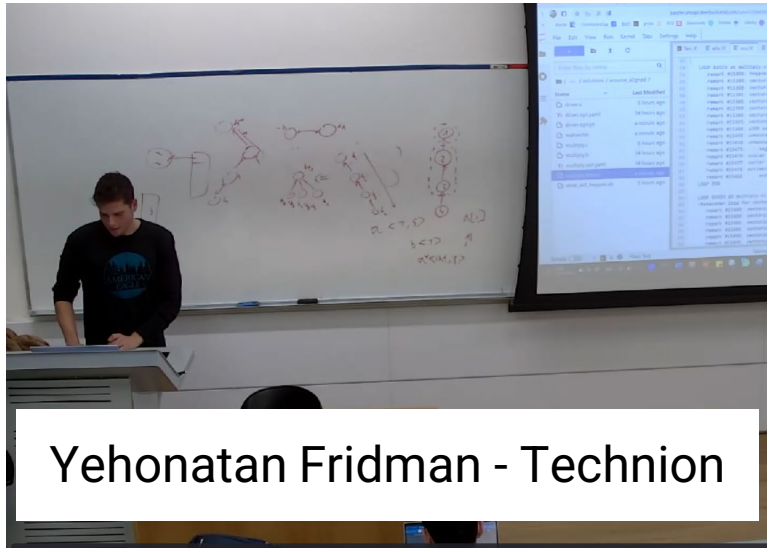
Yehonatan Fridman, Ben-Gurion University

Yehonatan researches the recoverability of scientific applications in HPC systems using the non-volatile RAM (NVRAM) technology, focusing on the Intel® Optane™ Persistent Memory product. Yehonatan is specifically interested in implementing recoverable mechanisms in OpenMP to enable concurrent algorithms running with OpenMP to run reliably.

Project

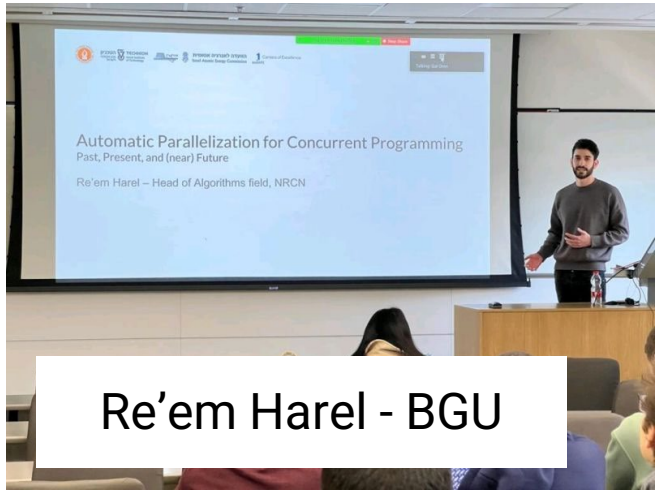
Teaching Program - Technion + Tel Aviv University


Gal Oren - TAU

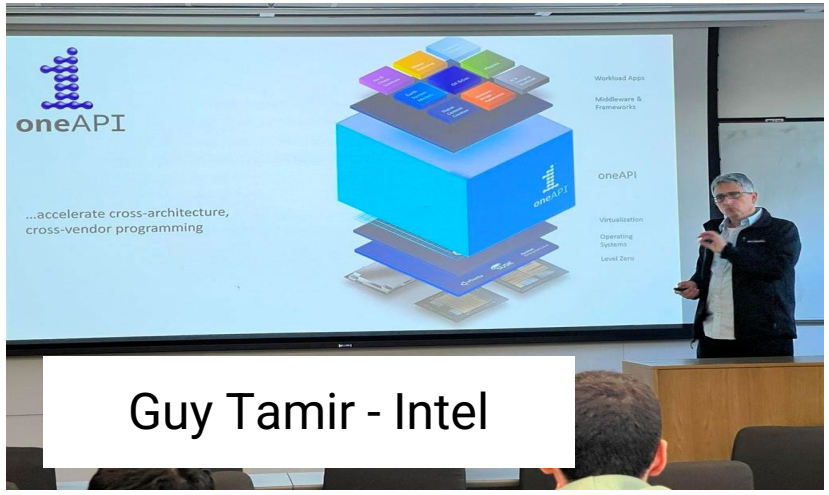

Yehonatan Fridman - Technion

Tim Mattson - Intel Labs


Gal Oren - Technion

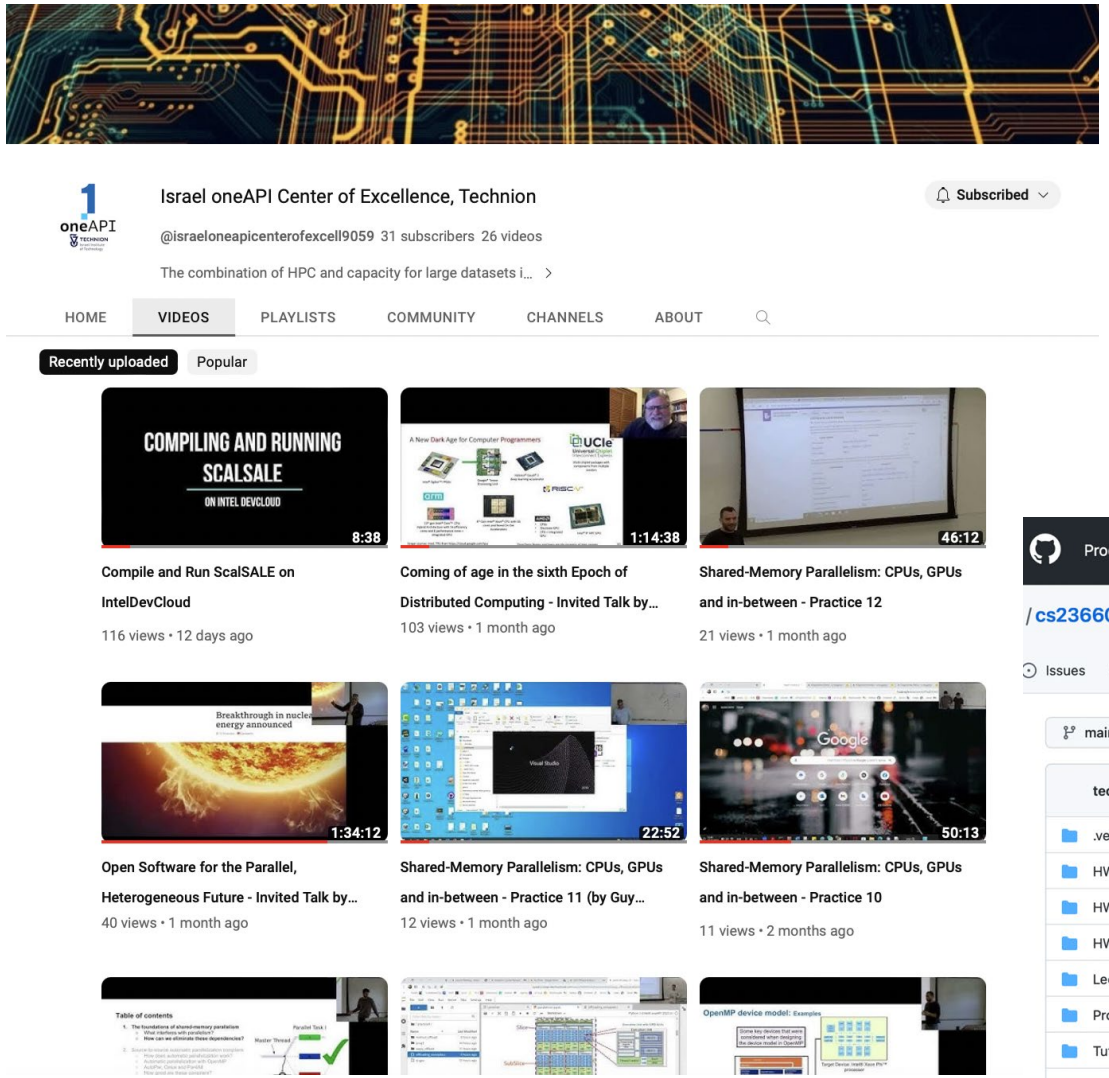


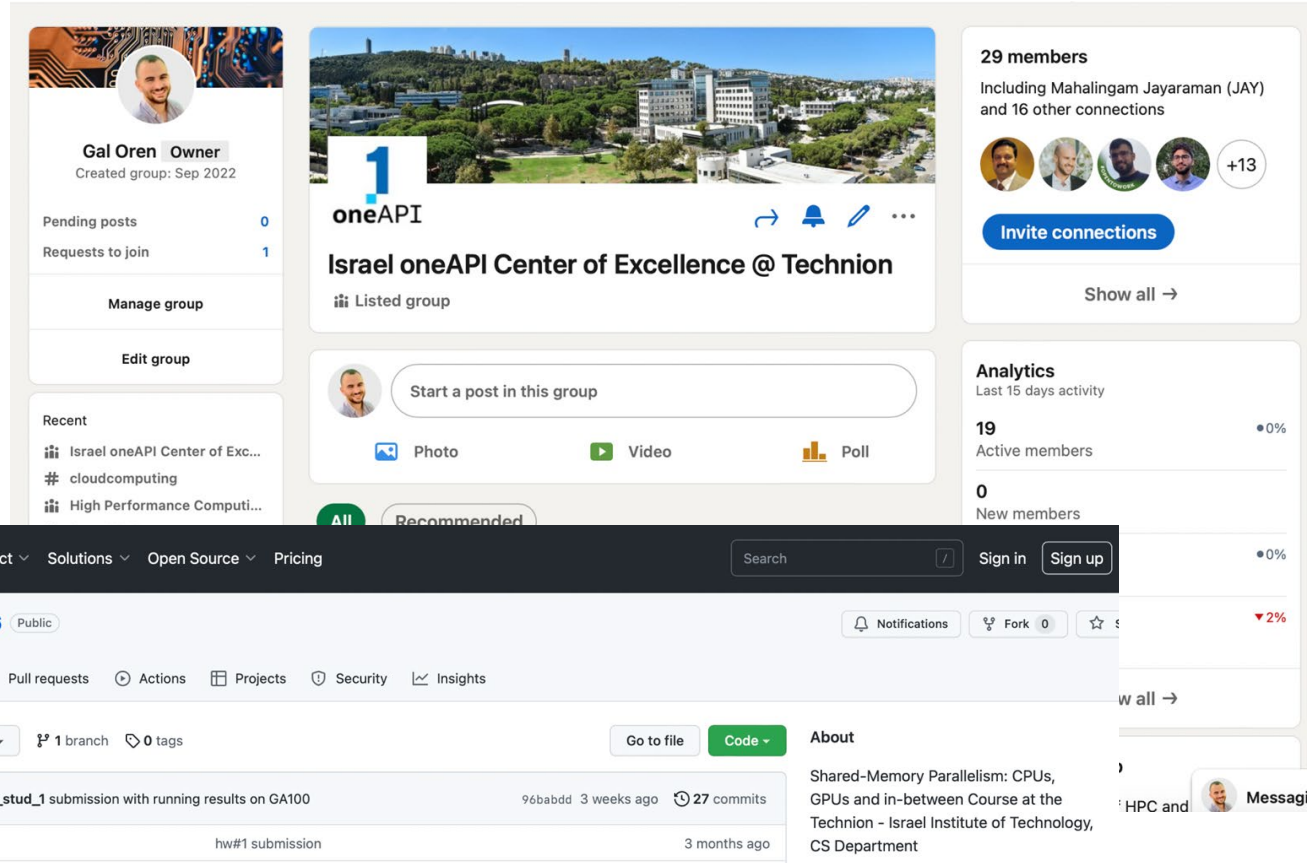
Re'em Harel - BGU



Guy Tamir - Intel

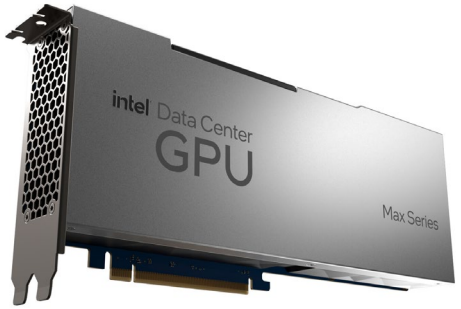
Teaching Program - YouTube, LinkedIn, GitHub





ScalSALE and oneAPI - Example

Facilitating Scientific Computing in the New Heterogenous World with Intel



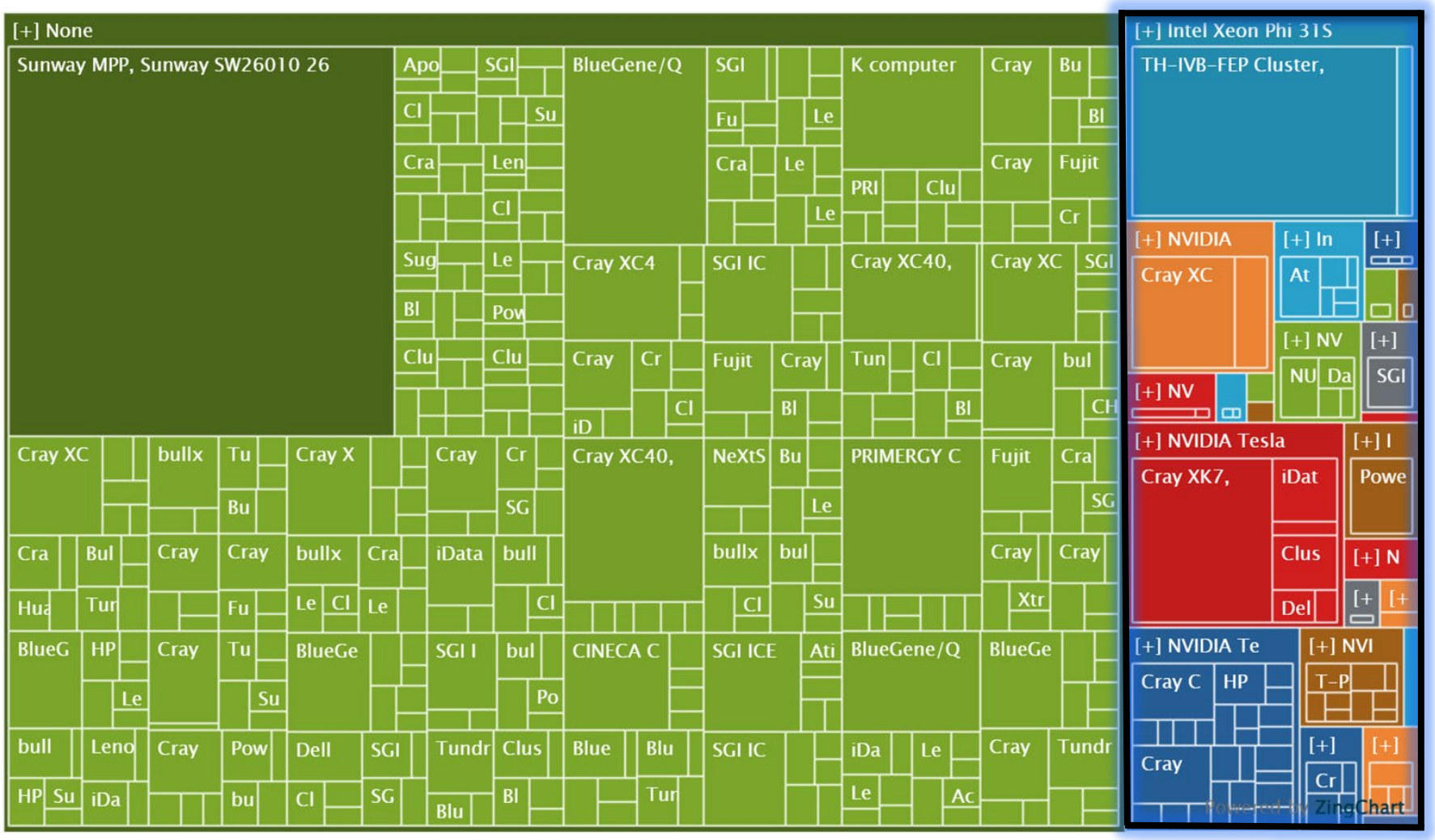
Portability and Scalability of OpenMP Offloading on State-of-the-art Accelerators

Yehonatan Fridman, Guy Tamir, Gal Oren

Top 500 supercomputers empowered by Accelerator/Co-Processor

Nov 2011

Top 500 supercomputers empowered by Accelerator/Co-Processor

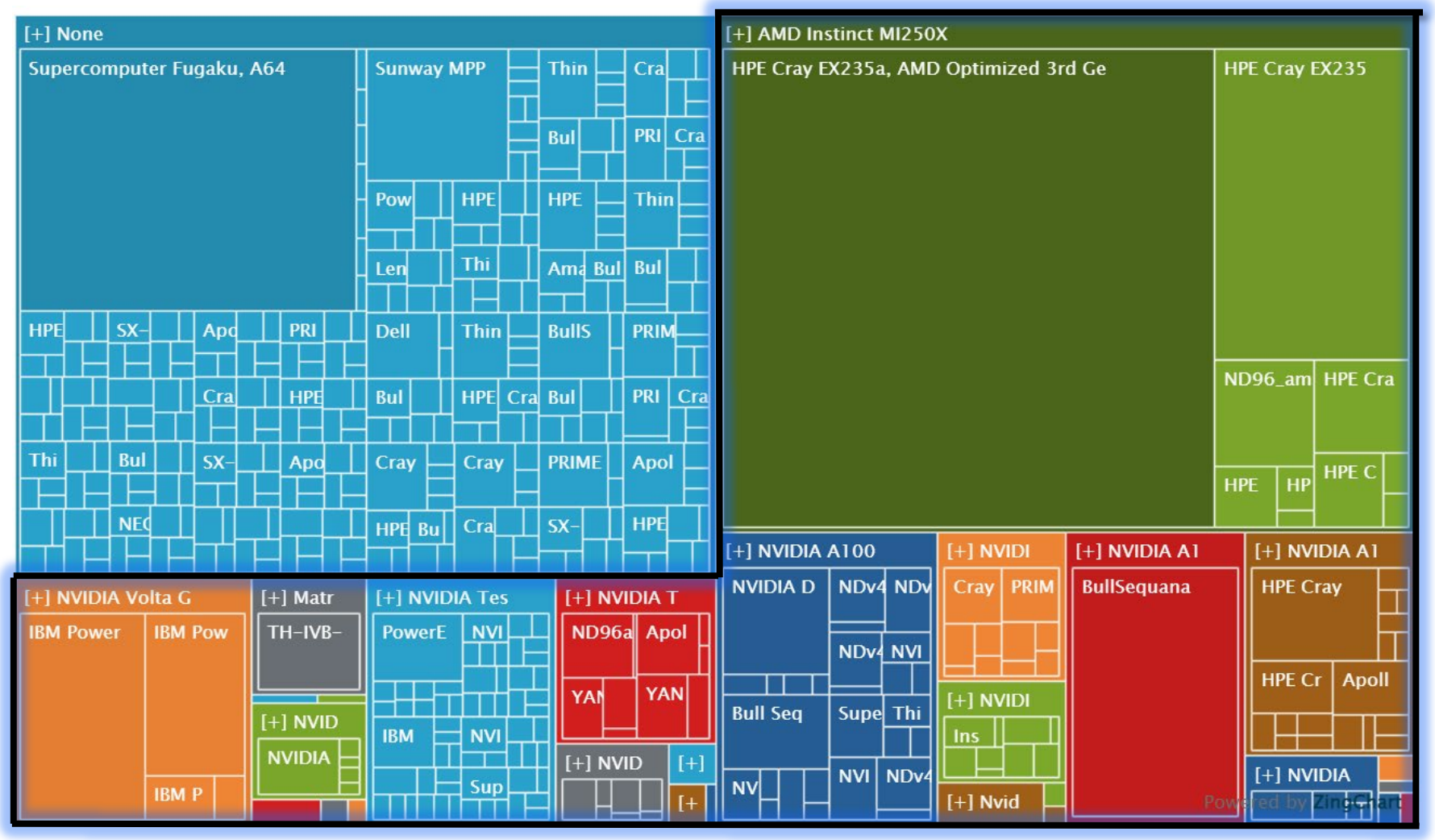


Nov 2016

Top 500 supercomputers empowered by Accelerator/Co-Processor

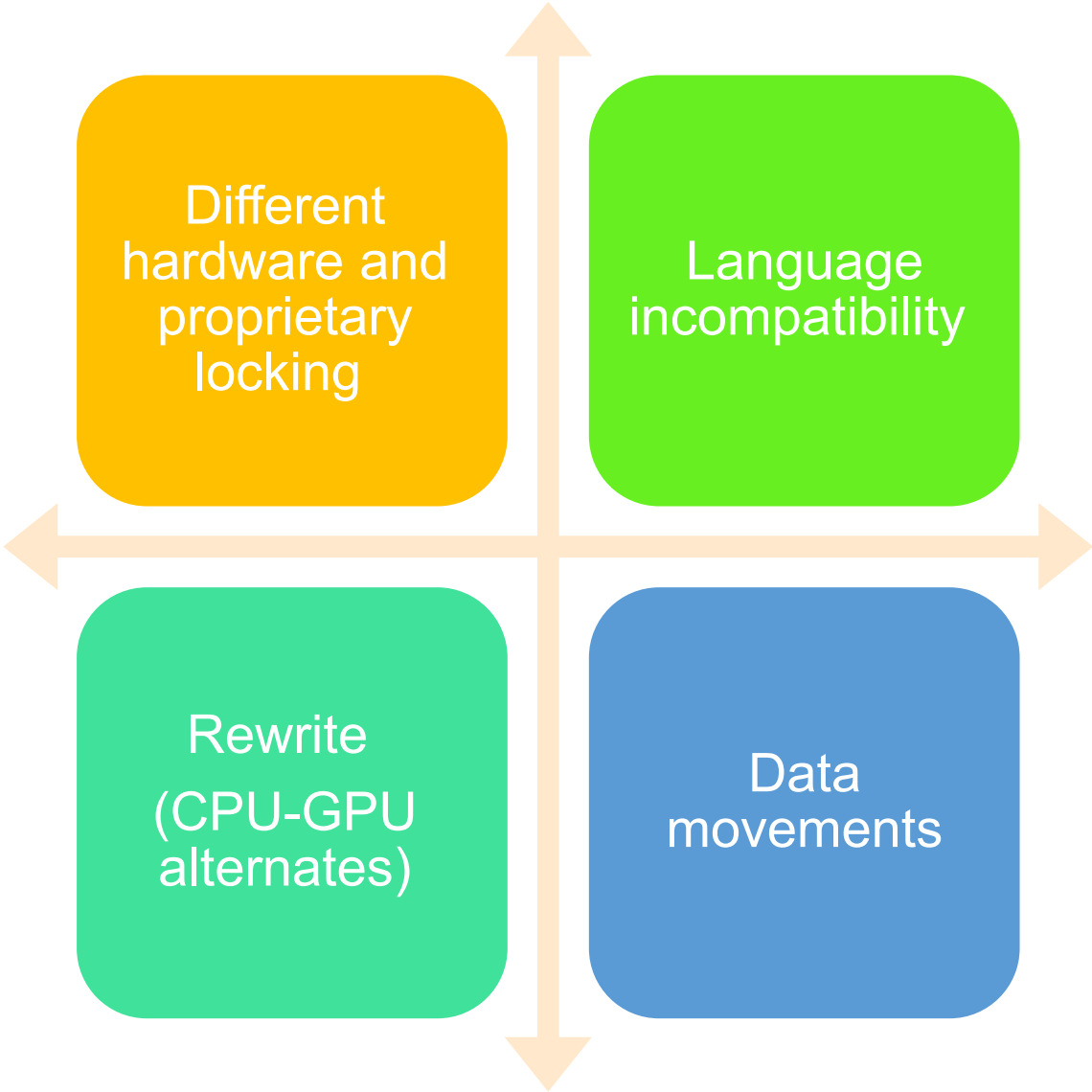
Nov 2020

Top 500 supercomputers empowered by Accelerator/Co-Processor



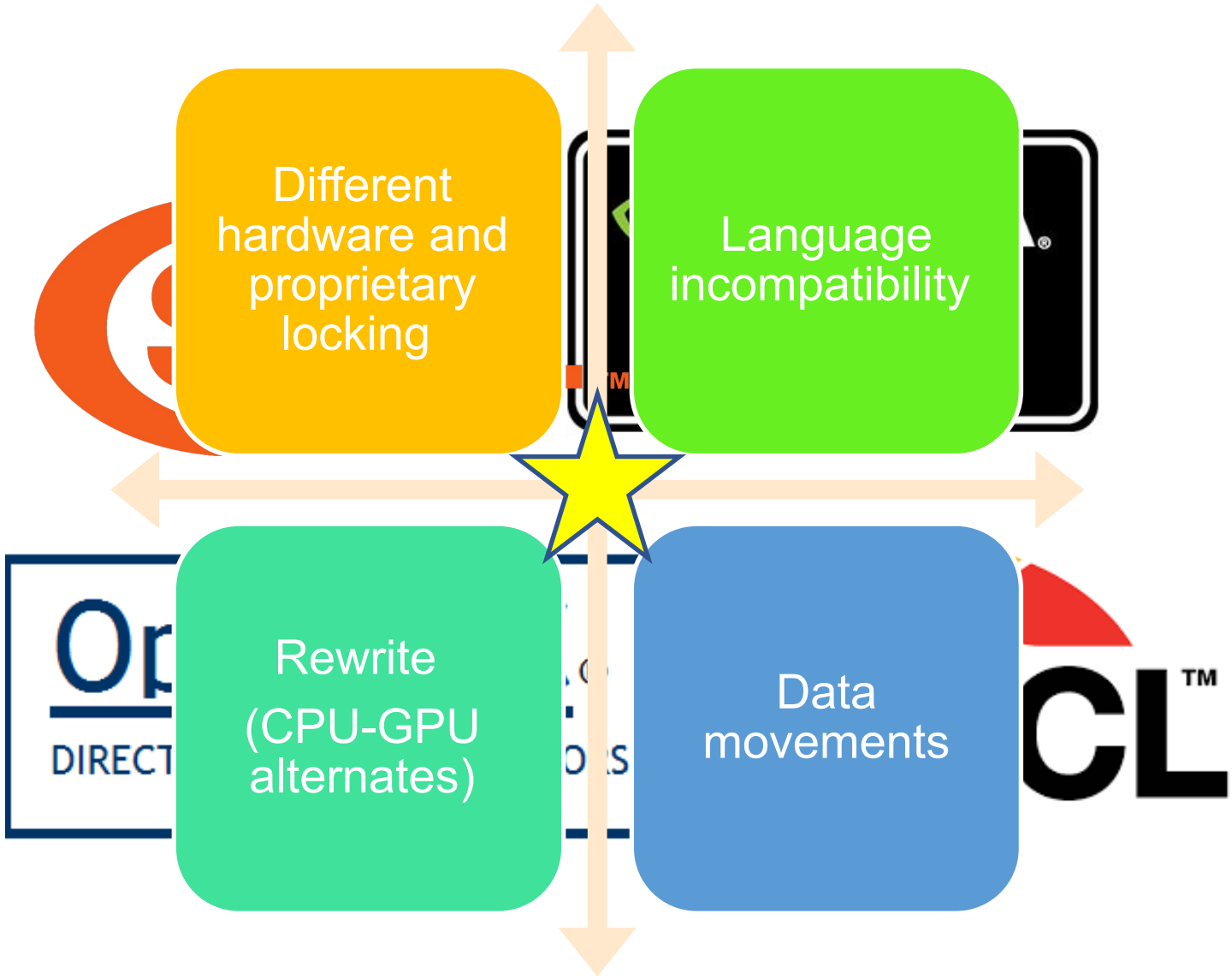
June 2023

Challenges in Targeting GPU

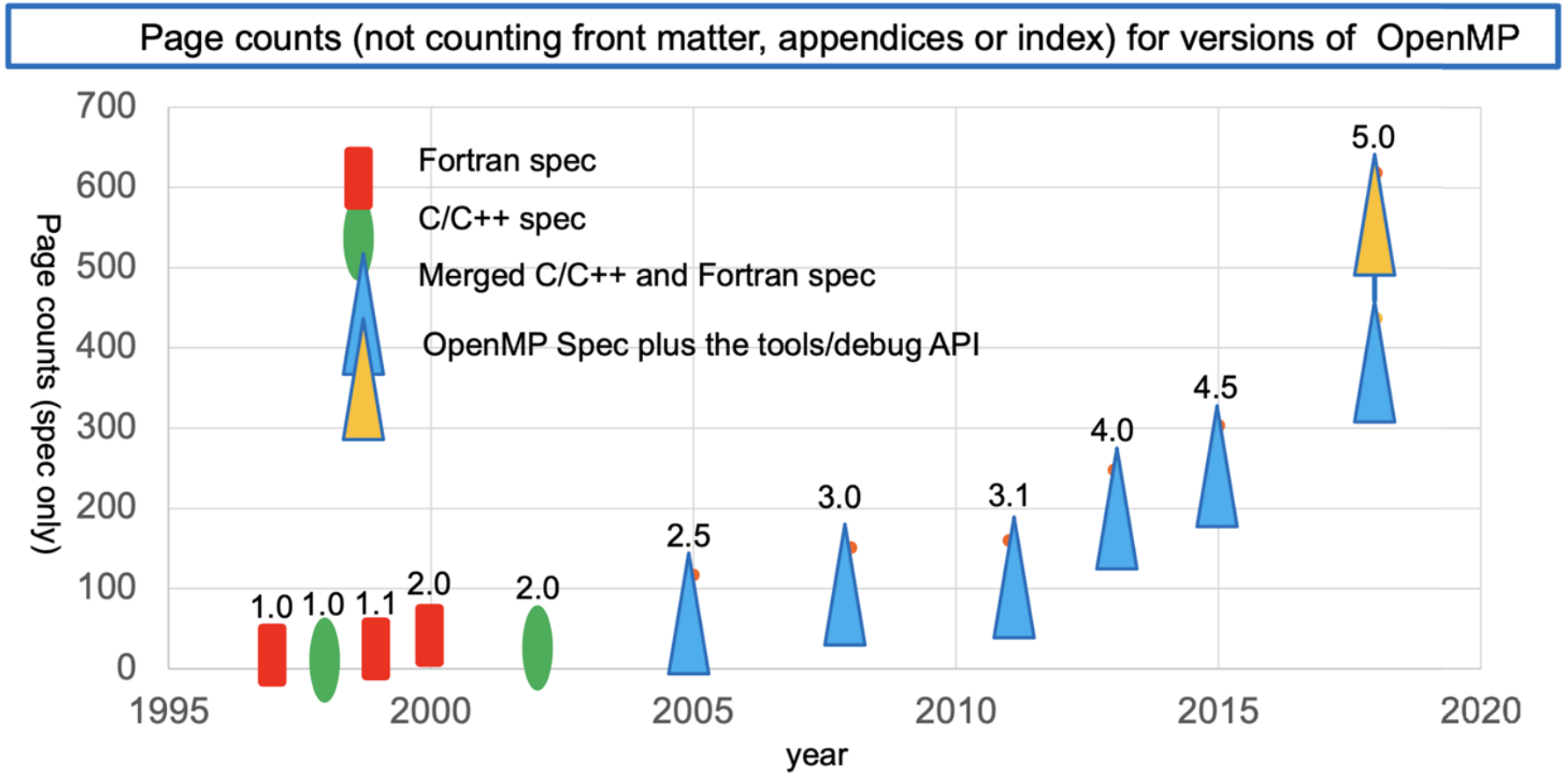


Programming GPU

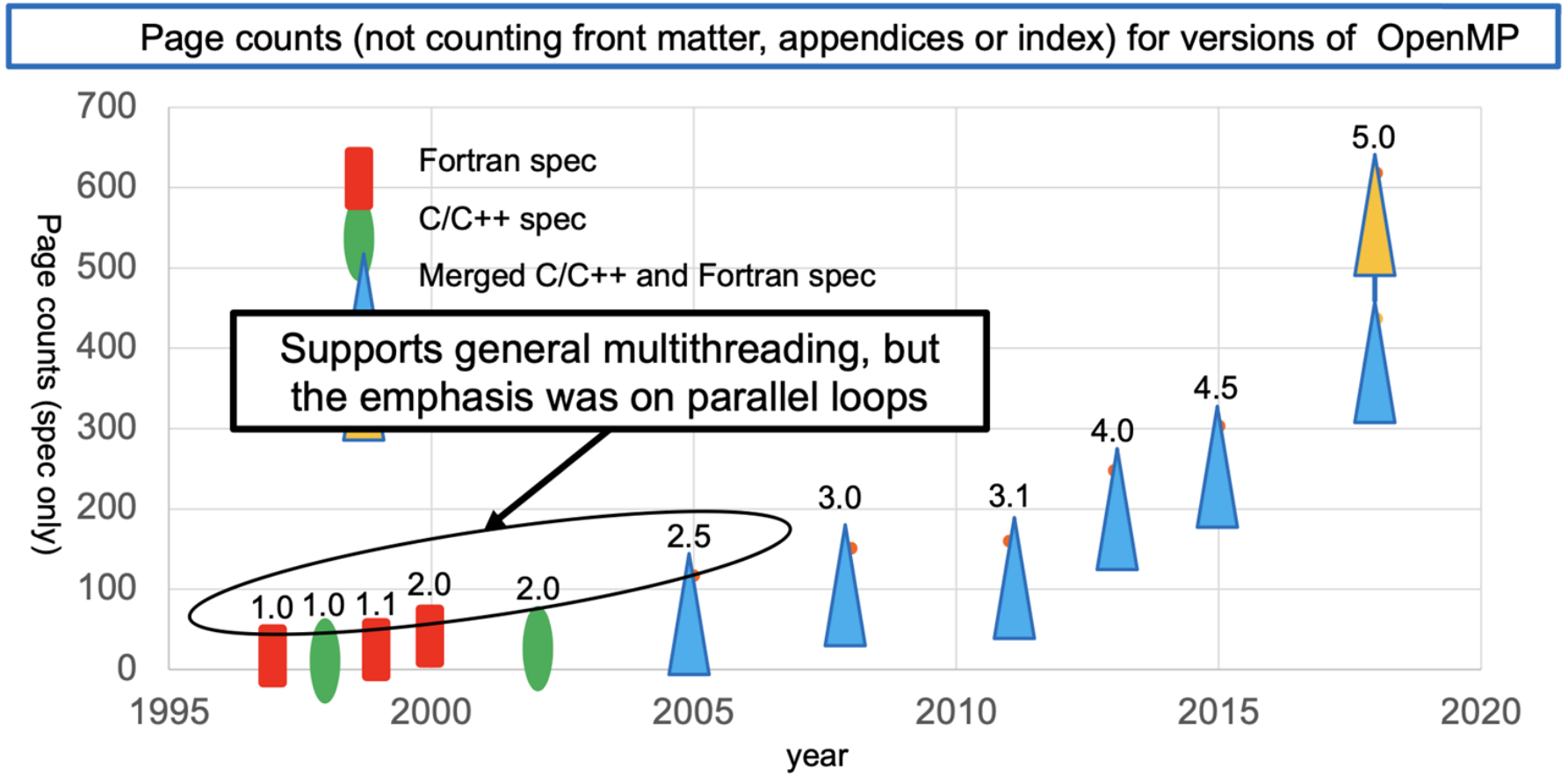
Programming GPU



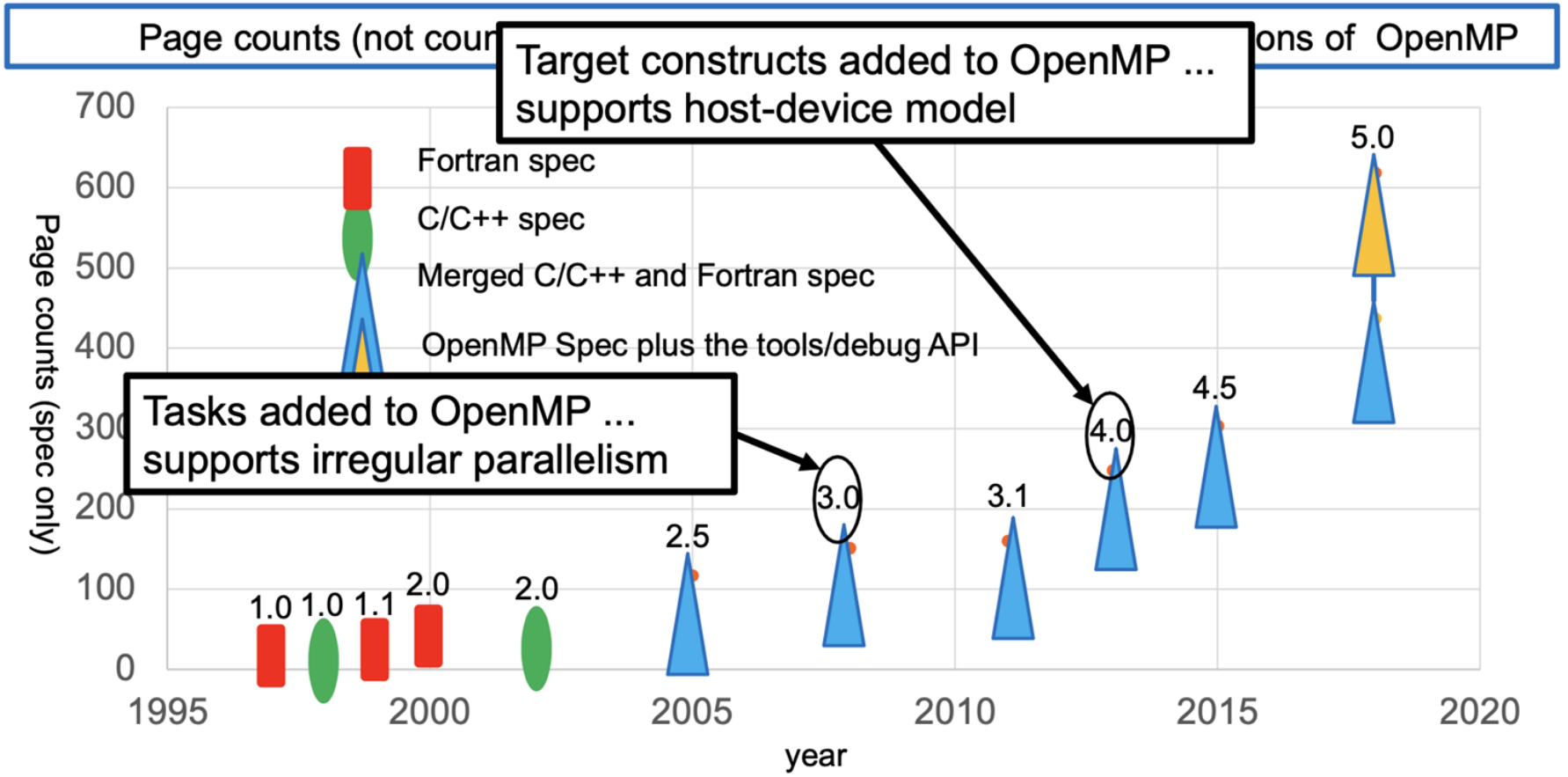
The growth complexity in OpenMP



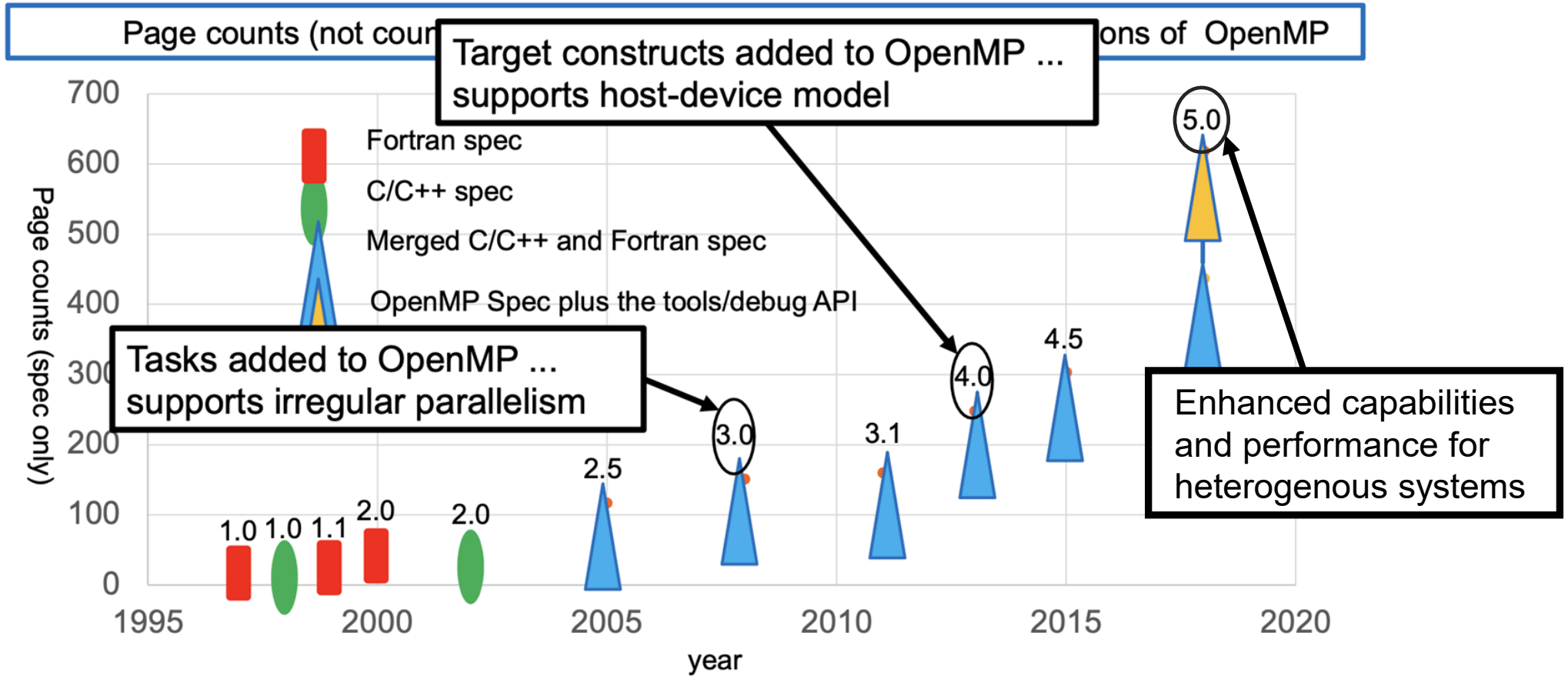
The growth complexity in OpenMP



The growth complexity in OpenMP



The growth complexity in OpenMP



The growth complexity in OpenMP

https://www.youtube.com/watch?v=W2Zcrhegq-1

OpenMP 5 and ecosystem

- OpenMP 5 adds features to make writing performance portable programs simpler.
- Highlighting some applicable to target:
  - **Loop construct**
  - Mappers
  - **Unified Shared Memory (USM)**
  - Function variants
  - **Reverse offload**
  - OMP_TARGET_OFFLOAD
  - Reduction result mapping
    - Reduction variables now implicitly map(tofrom)

Thanks: https://github.com/uob-hpc/openmp-tutorial

OpenMP 5.0: loop construct

- Assert that the iterations in a loop nest may execute in any order, including concurrently
  - Let the compiler figure our how to best utilize parallel resources

```
double a[N], b[N], c[N];

#pragma omp target
#pragma omp loop
for (int i=0; i<N; i++)
  a[i] = FUNC(b[i], c[i]);
```

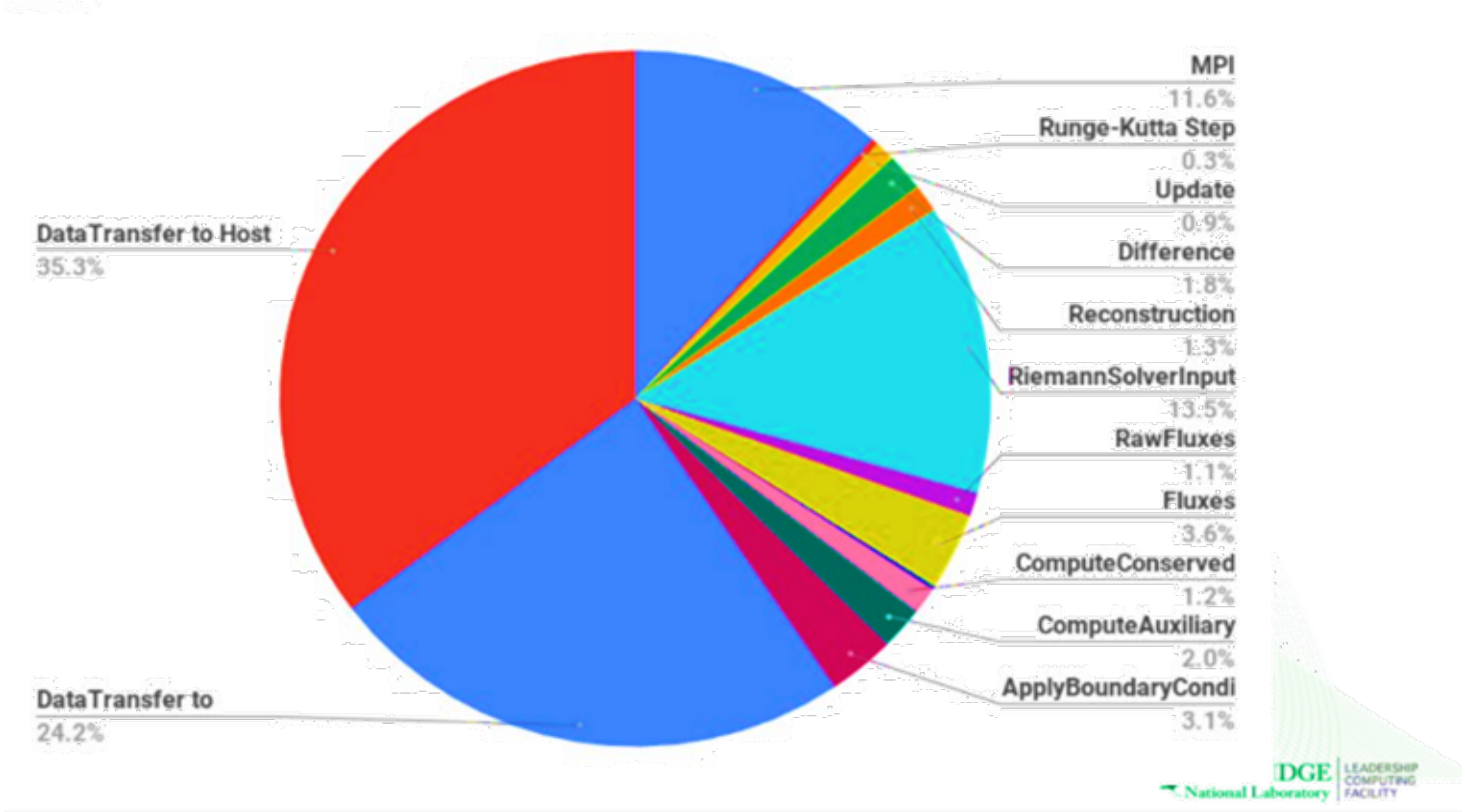
Iterations can execute in any order. Rely on the compiler to schedule iterations across teams, threads, simd, …

Thanks: https://github.com/uob-hpc/openmp-tutorial

Data Transfer between Host and Device

Porting a Fluid Dynamics Application: Riemann Problem

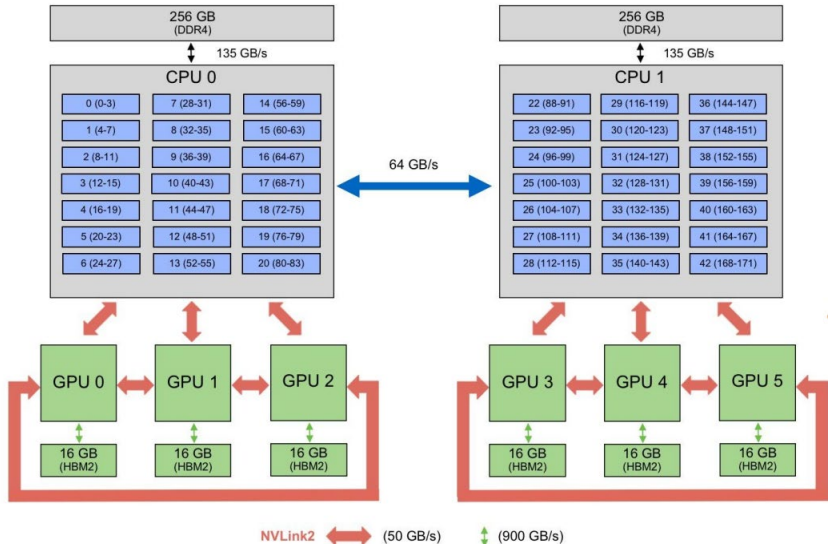
Performance Results: Timing Distribution



https://www.olcf.ornl.gov/wp-content/uploads/fdp.ajdraidub_pohskrow_timmus/2018/12

OAK RIDGE National Laboratory

Summit node with (2) IBM Power9
+ (6) NVIDIA Volta V100



הוועדה לאנרגיה אטומית
Israel Atomic Energy Commission

TECHNION Israel Institute of Technology

oneAPI

OpenMP 5.0: #pragma omp requires

- Code requires specific features, e.g., shared memory between host and devices.

```
typedef struct mypoints {
    struct myvec * x;
    struct myvec scratch;
    double useless_data[500000];
} mypoints_t;

#pragma omp requires unified_shared_memory

mypoints_t p = new_mypoints_t();

#pragma omp target
{
    do_something_with_p(&p);
}
```

This code assumes that the host and device share memory.

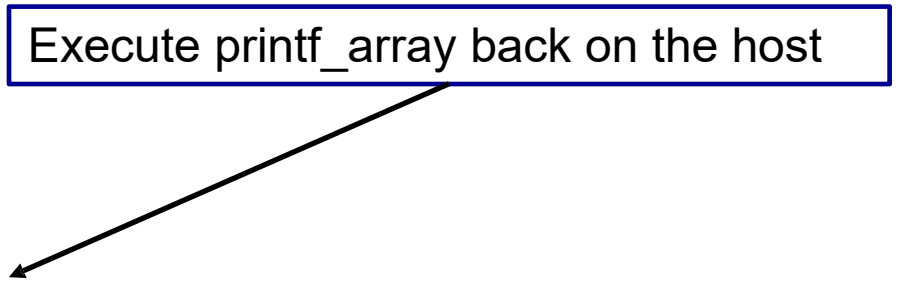
No map clauses. All of p is shared between the host and device.

Thanks: https://github.com/uob-hpc/openmp-tutorial

OpenMP 5.0: reverse offload

- Execute a region of code back on the host from within a target region.
  - A target device may not be able to execute this code.

```
double a[N], b[N], c[N];

#pragma omp target map(to:b,c) map(from:a)
{

  for (int i=0; i<N; i++)
    a[i] = FUNC(b[i], c[i]);

 #pragma omp target device(ancestor:1)
  printf_array(a);

  ...
}
```



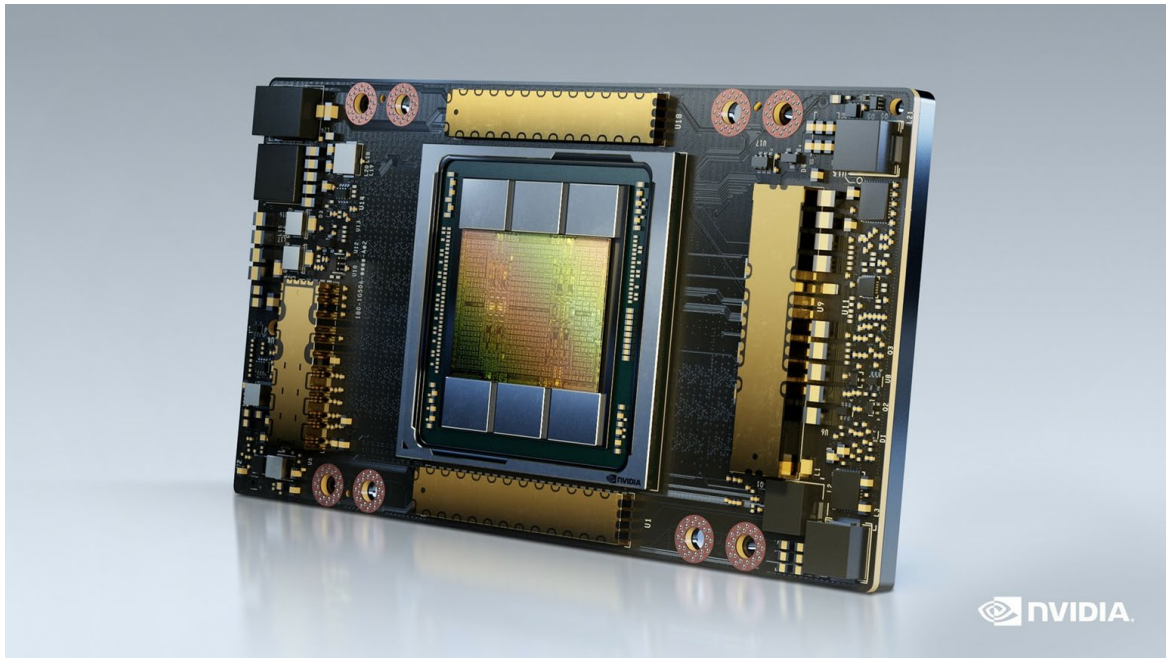
Thanks: https://github.com/uob-hpc/openmp-tutorial

2 State-of-the-art accelerators

Intel Max 1100 GPU (Ponte Vecchio, PVC)

https://www.facebook.com/intelgraphics/videos/2088359258031184

NVIDIA A100 GPU



https://www.nvidia.com/fr-fr/data-center/a/100

GPUs specs

	Intel PVC1100	NVIDIA A100
GPU Architecture	Xe-HPC	NVIDIA Ampere
Memory	48GB HBM2e	40GB HBM2e
Memory Bandwidth	1228.8 GB/s	1555 GB/s
Compute Cores	7168	6912

Full system and compilers

System	CPU (host)	GPU (device)	Compiler
#1	×2 Intel 4th Gen Xeon (Sapphire Rapids) processors	Intel Data Center GPU Max 1100	oneAPI 2023 ifx/icpx/icx
#2	×2 Intel Xeon Gold 6338 processors	NVIDIA A100 Tensor Core GPU	NVHPC 23.3 nvfortran/nvc++/nvc

Compile flags

System	Compilation flags
#1	`-O3 -qopenmp -fopenmp-targets=spir64 -fiopenmp -fopenmp-version={50,51,52}`
#2	`-O3 -mp=gpu -gpu=cc80`

GPUs specs

	Intel PVC1100	NVIDIA A100
GPU Architecture	Xe-HPC	NVIDIA Ampere
Memory	48GB HBM2e	40GB HBM2e
Memory Bandwidth	1228.8 GB/s	1555 GB/s
Compute Cores	7168	6912

Full system and compilers

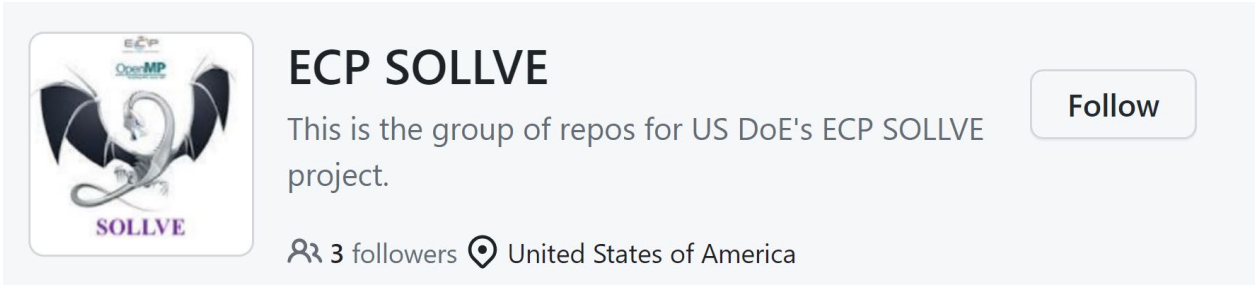
System	CPU (host)	GPU (device)	Compiler
#1	×2 Intel 4th Gen Xeon (Sapphire Rapids) processors	Intel Data Center GPU Max 1100	oneAPI 2023 ifx/icpx/icx
#2	×2 Intel Xeon Gold 6338 processors	NVIDIA A100 Tensor Core GPU	NVHPC 23.3 nvfortran/nvc++/nvc

Compile flags

System	Compilation flags
#1	`-O3 -qopenmp -fopenmp-targets=spir64 -fiopenmp -fopenmp-version={50,51,52}`
#2	`-O3 -mp=gpu -gpu=cc80`

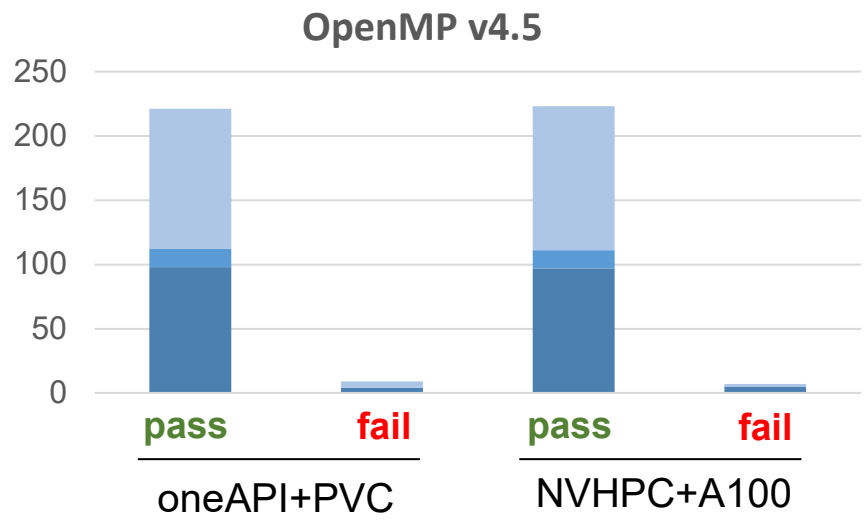
PORTABILITY: SOLLVE OpenMP V&V

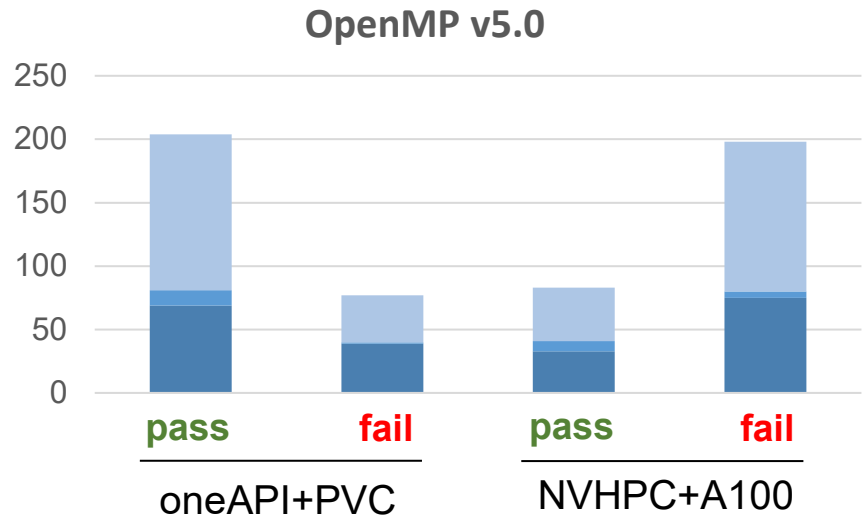
- **SOLLVE** = **S**caling **O**penMP with **LLV**m for **E**xascale.
- The OpenMP sub-project in US DoE's ECP.
- Advancing the OpenMP specification and its implementations to address Exascale application challenges.
- Proposing a validation suite (the V&V suite) to assess their progress and that of vendors to ensure that quality implementations of OpenMP are being delivered to Exascale systems.

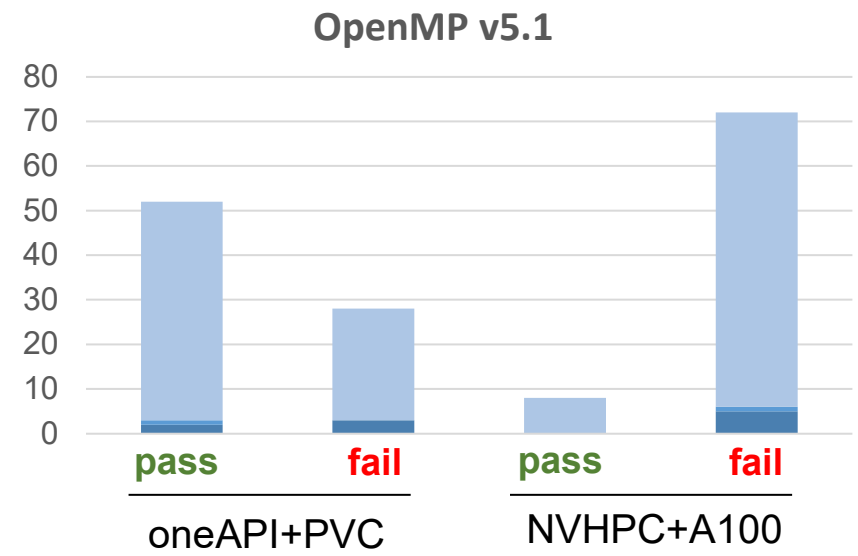


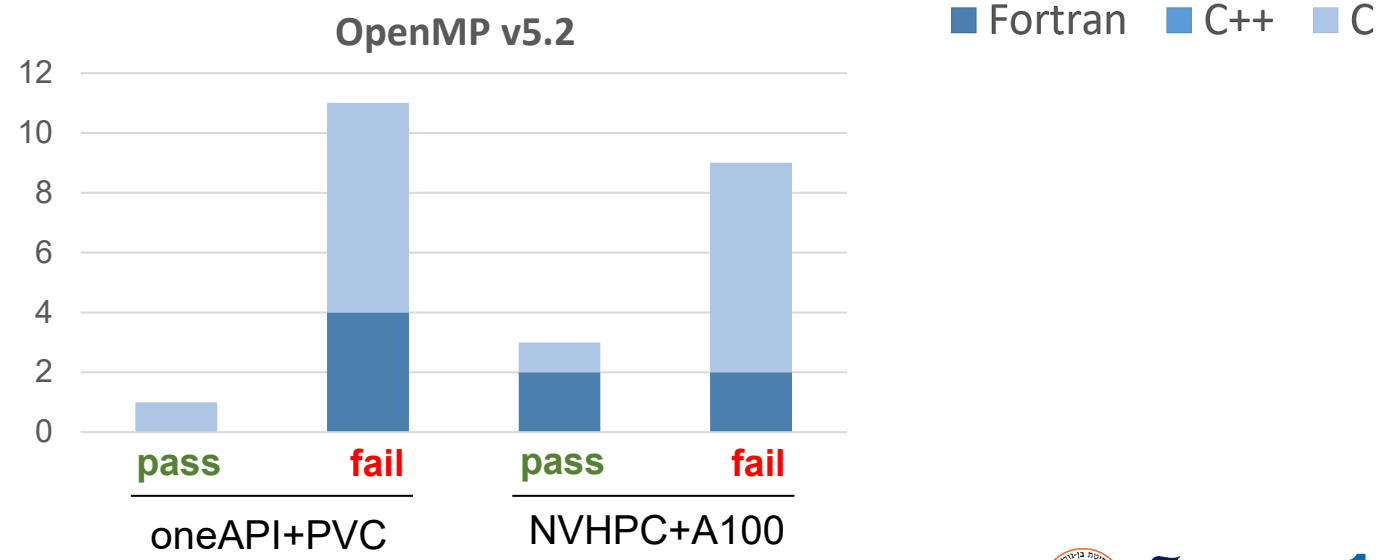
- https://sollve.github.io
- https://crpl.cis.udel.edu/ompvvsollve/
- Huber, Thomas, et al. "ECP SOLLVE: Validation and Verification Testsuite Status Update and Compiler Insight for OpenMP." *arXiv preprint arXiv:2208.13301* (2022).

SOLLVE OpenMP V&V with oneAPI & NVHPC for PVC1100 & A100









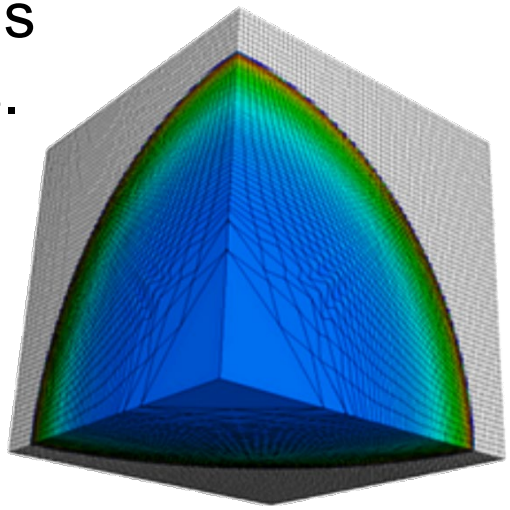
OpenMP Portability Discussion

Test Name	OMP Version	oneAPI 2023	NVHPC 23.3
test_requires_reverse_offload.c	5.0	FAIL	FAIL
test_requires_unified_address.c	5.0	FAIL	FAIL
test_requires_unified_shared_memory_static.c	5.0	FAIL	FAIL
test_requires_unified_shared_memory _heap_is_device_ptr.F90	5.1	FAIL	FAIL
test_requires_unified_shared_memory _stack_is_device_ptr.F90	5.1	FAIL	FAIL

- Directives with promising importance are not supported by both compilers (NVHPC and oneAPI).

SCALABILITY: LULESH

- Simulates shock hydrodynamics using an unstructured mesh.
- Proxy-app that is designed to represent the computational patterns and performance characteristics of complex scientific applications.
- Multi-bound (compute-bound, memory-bound, …).
- LULESH offload to GPU implementation with OpenMP 4.0.
  - by AMD https://github.com/AMDComputeLibraries/OpenMPApps/tree/master/lulesh-mp4
  - Very basic – OpenMP 4.0 does not support advanced capabilities for effectively optimizing data movements.



LLNL/**LULESH**

Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics (LULESH)

6 Contributors · 7 Issues · 80 Stars · 64 Forks

grep -ni “pragma omp target” lulesh.cc

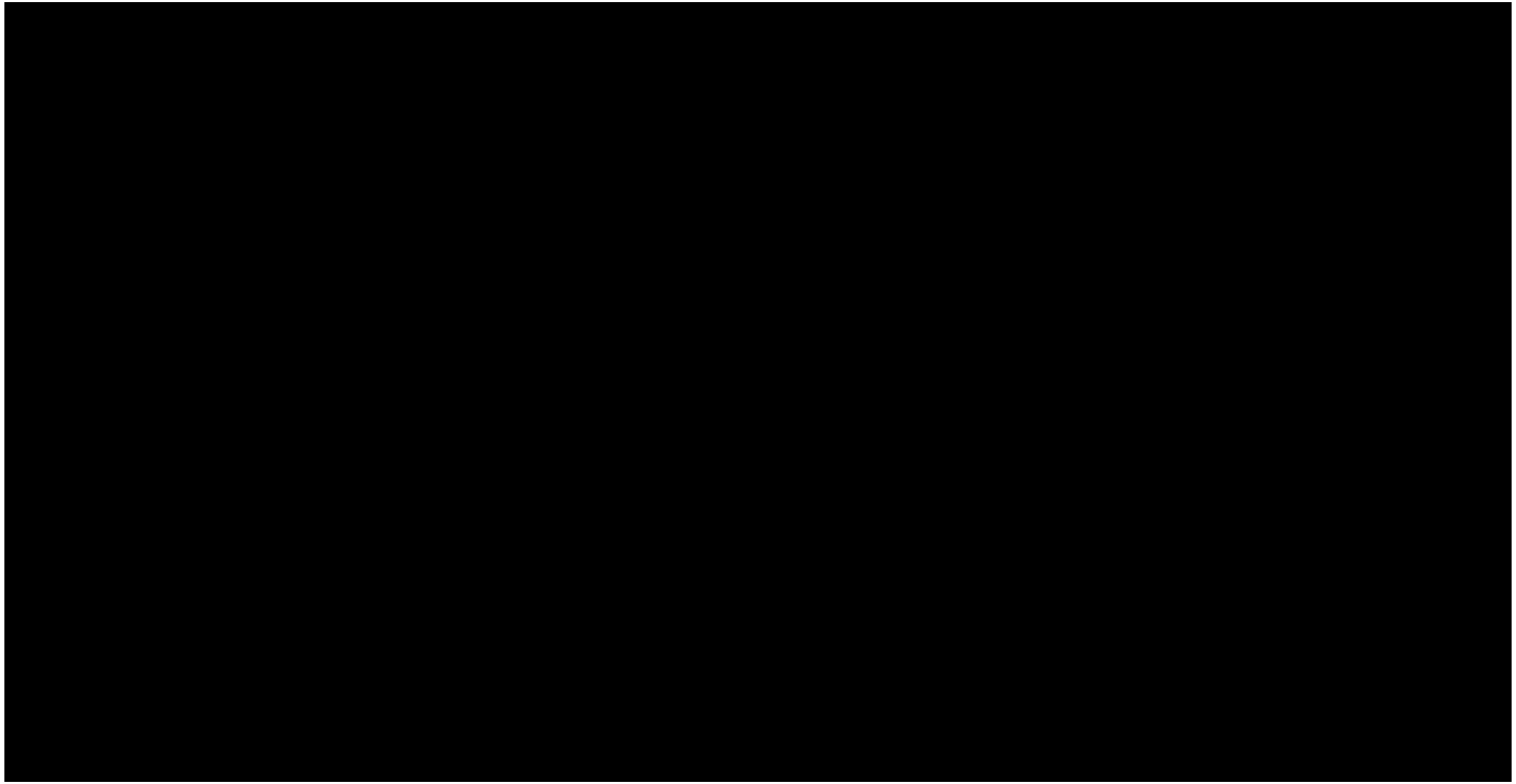
```
(base) [yonif@gpu002 lulesh-mp4]$ grep -rni "pragma omp target" lulesh.cc
358:   #pragma omp target data map(to: p[:numElem], q[:numElem]) \
362:    # pragma omp target teams if (USE_GPU == 1)
619:  #pragma omp target enter data map(alloc:fx[:numNode],fy[:numNode],fz[:numNode])  \
622:  #pragma omp target data map(alloc: x[:numNode], y[:numNode], z[:numNode]) \
628:    # pragma omp target teams if (USE_GPU == 1)
711:     #pragma omp target data map(to: nodeElemStart[:len1], nodeElemCornerList[:len2])  if(USE_GPU == 1)
714:        # pragma omp target teams if (USE_GPU == 1)
737:     #pragma omp target exit data map(delete:fx_elem[:numElem8], fy_elem[:numElem8], fz_elem[:numElem8]) if(USE_GPU == 1)
921:   #pragma omp target enter data map(alloc:fx_elem[:numElem8], fy_elem[:numElem8], fz_elem[:numElem8]) \
926:   #pragma omp target data  map(alloc: dvdx[:numElem8], dvdy[:numElem8], dvdz[:numElem8], \
935:   # pragma omp target teams if (USE_GPU == 1)
1160:      #pragma omp target data map(to: nodeElemStart[:len1], nodeElemCornerList[:len2]) if (USE_GPU == 1)
1163:        # pragma omp target teams if (USE_GPU == 1)
1186:      #pragma omp target exit data map(delete:fx_elem[:numElem8], fy_elem[:numElem8], fz_elem[:numElem8]) if(USE_GPU == 1)
1219:   #pragma omp target enter data map(alloc:dvdx[:numElem8], dvdy[:numElem8], dvdz[:numElem8], \
1225:   #pragma omp target data map(alloc: x[:numNode], y[:numNode], z[:numNode], \
1229:   # pragma omp target teams if (USE_GPU == 1)
1316:   #pragma omp target exit data map(delete: dvdx[:numElem8], dvdy[:numElem8], dvdz[:numElem8], \
1350:      #pragma omp target enter data map(alloc:sigxx[:numElem], sigyy[:numElem], sigzz[:numElem]) if (USE_GPU == 1)
1359:      #pragma omp target enter data map(alloc:determ[:numElem]) if(USE_GPU == 1)
1360:      #pragma omp target enter data map(alloc:x[:numNode], y[:numNode], z[:numNode]) if(USE_GPU == 1)
1369:      #pragma omp target update from(determ[:numElem]) if(USE_GPU == 1)
1383:      #pragma omp target exit data map(delete:sigxx[:numElem], sigyy[:numElem], sigzz[:numElem]) if(USE_GPU == 1)
1387:      #pragma omp target exit data map(delete:determ[:numElem]) if(USE_GPU == 1)
1413:# pragma omp target data map(alloc: fx[:numNode], fy[:numNode], fz[:numNode]) if (USE_GPU == 1)
1416:  # pragma omp target teams if (USE_GPU == 1)
1459:   #pragma omp target data map(alloc: fx[:numNode], fy[:numNode], fz[:numNode]) \
1464:      # pragma omp target teams if (USE_GPU == 1)
1496:   # pragma omp target data map(to: symmX[:numNodeBC], symmY[:numNodeBC], symmZ[:numNodeBC]) \
1500:     # pragma omp target teams if (USE_GPU == 1)
1527:   # pragma omp target data map(alloc: xdd[:numNode], ydd[:numNode], zdd[:numNode]) \
1532:     # pragma omp target teams if (USE_GPU == 1)
1568:  #pragma omp target data map(alloc: xd[:numNode], yd[:numNode], zd[:numNode]) \
1573:    # pragma omp target teams if (USE_GPU == 1)
1917:  # pragma omp target data map(alloc:    xd[:numNode], yd[:numNode], zd[:numNode]) \
1926:    # pragma omp target teams if (USE_GPU == 1)
2039:     #pragma omp target enter data map (alloc:dxx[:numElem], dyy[:numElem], dzz[:numElem]) if (USE_GPU == 1)
2049:     # pragma omp target data map(from: vdov[:numElem]) \
2054:       # pragma omp target teams if (USE_GPU == 1)
2086:     #pragma omp target exit data map (delete:dxx[:numElem], dyy[:numElem], dzz[:numElem]) if (USE_GPU == 1)
2128:  # pragma omp target data map(alloc: xd[:numNode], yd[:numNode], zd[:numNode]) \
2137:   # pragma omp target teams if (USE_GPU == 1)
2323:   # pragma omp target data map(to:    vdov[:numElem], elemMass[:numElem], volo[:numElem]) \
2332:        # pragma omp target teams if (USE_GPU == 1)
2523:     #pragma omp target enter data map (alloc:delv_xi[0:allElem],   delx_xi[0:numElem], \
2559:     #pragma omp target exit data map (delete:delv_xi[0:allElem],   delx_xi[0:numElem], \
2598:     # pragma omp target data map(to: vnew[:numElem], length, v_cut) \
2602:         # pragma omp target teams if (USE_GPU == 1)
2654:   # pragma omp target data map(to: ql[:numElem], qq[:numElem], delv[:numElem], elemRep[:numElem], elemElem[:numElem], \
2661:      # pragma omp target teams if (USE_GPU == 1)
2892:  #pragma omp target enter data map (alloc:vnew[0:numElem]) if (USE_GPU == 1)
2901:  #pragma omp target exit data map (delete:vnew[0:numElem]) if (USE_GPU == 1)
3250:   #pragma omp target enter data map(to:x[0:numNode],y[0:numNode],z[0:numNode], \
3273:  #pragma omp target exit data map(from:x[0:numNode],y[0:numNode],z[0:numNode], \
```

grep -ni “pragma omp target” lulesh.cc

```
(base) [yonif@gpu002 lulesh-mp4]$ grep -rni "pragma omp target" lulesh.cc
358:   #pragma omp target data map(to: p[:numElem], q[:numElem]) \
362:    # pragma omp target teams if (USE_GPU == 1)
619:  #pragma omp target enter data map(alloc:fx[:numNode],fy[:numNode],fz[:numNode])  \
622:  #pragma omp target data map(alloc: x[:numNode], y[:numNode], z[:numNode]) \
628:    # pragma omp target teams if (USE_GPU == 1)
711:     #pragma omp target data map(to: node
714:        # pragma omp target teams if (US
737:     #pragma omp target exit data map(del
921:   #pragma omp target enter data map(allo
926:   #pragma omp target data  map(alloc: dv
935:   # pragma omp target teams if (USE_GPU
1160:     #pragma omp target data map(to: no
1163:        # pragma omp target teams if (US
1186:     #pragma omp target exit data map(d
1219:   #pragma omp target enter data map(all
1225:    #pragma omp target data map(alloc: x[
1229:    # pragma omp target teams if (USE_GPU
1316:   #pragma omp target exit data map(dele
1350:     #pragma omp target enter data map(
1359:     #pragma omp target enter data map(
1360:     #pragma omp target enter data map(
1369:     #pragma omp target update from(det
1383:     #pragma omp target exit data map(d
1387:     #pragma omp target exit data map(d
1413:# pragma omp target data map(alloc: fx[:
1416:  # pragma omp target teams if (USE_GPU
1459:   #pragma omp target data map(alloc: fx
1464:      # pragma omp target teams if (USE_
1496:   # pragma omp target data map(to: symm
1500:     # pragma omp target teams if (USE_G
1527:   # pragma omp target data map(alloc: x
1532:     # pragma omp target teams if (USE_G
1568:  #pragma omp target data map(alloc: xd[
1573:    # pragma omp target teams if (USE_GP
1917:  # pragma omp target data map(alloc:
1926:    # pragma omp target teams if (USE_GP
2039:     #pragma omp target enter data map
2049:     # pragma omp target data map(from:
2054:       # pragma omp target teams if (US
2086:     #pragma omp target exit data map (
2128:  # pragma omp target data map(alloc: xd
2137:   # pragma omp target teams if (USE_GPU
2323:   # pragma omp target data map(to:    v
2332:        # pragma omp target teams if (U
2523:     #pragma omp target enter data map
2559:     #pragma omp target exit data map (delete:delv_xi[0:allElem],   delx_xi[0:numElem]), \
2598:     # pragma omp target data map(to: vnew[:numElem], length, v_cut) \
2602:         # pragma omp target teams if (USE_GPU == 1)
2654:   # pragma omp target data map(to: ql[:numElem], qq[:numElem], delv[:numElem], elemRep[:numElem], elemElem[:numElem], \
2661:     # pragma omp target teams if (USE_GPU == 1)
2892:  #pragma omp target enter data map (alloc:vnew[0:numElem]) if (USE_GPU == 1)
2901:  #pragma omp target exit data map (delete:vnew[0:numElem]) if (USE_GPU == 1)
3250:   #pragma omp target enter data map(to:x[0:numNode],y[0:numNode],z[0:numNode], \
3273:  #pragma omp target exit data map(from:x[0:numNode],y[0:numNode],z[0:numNode], \
```

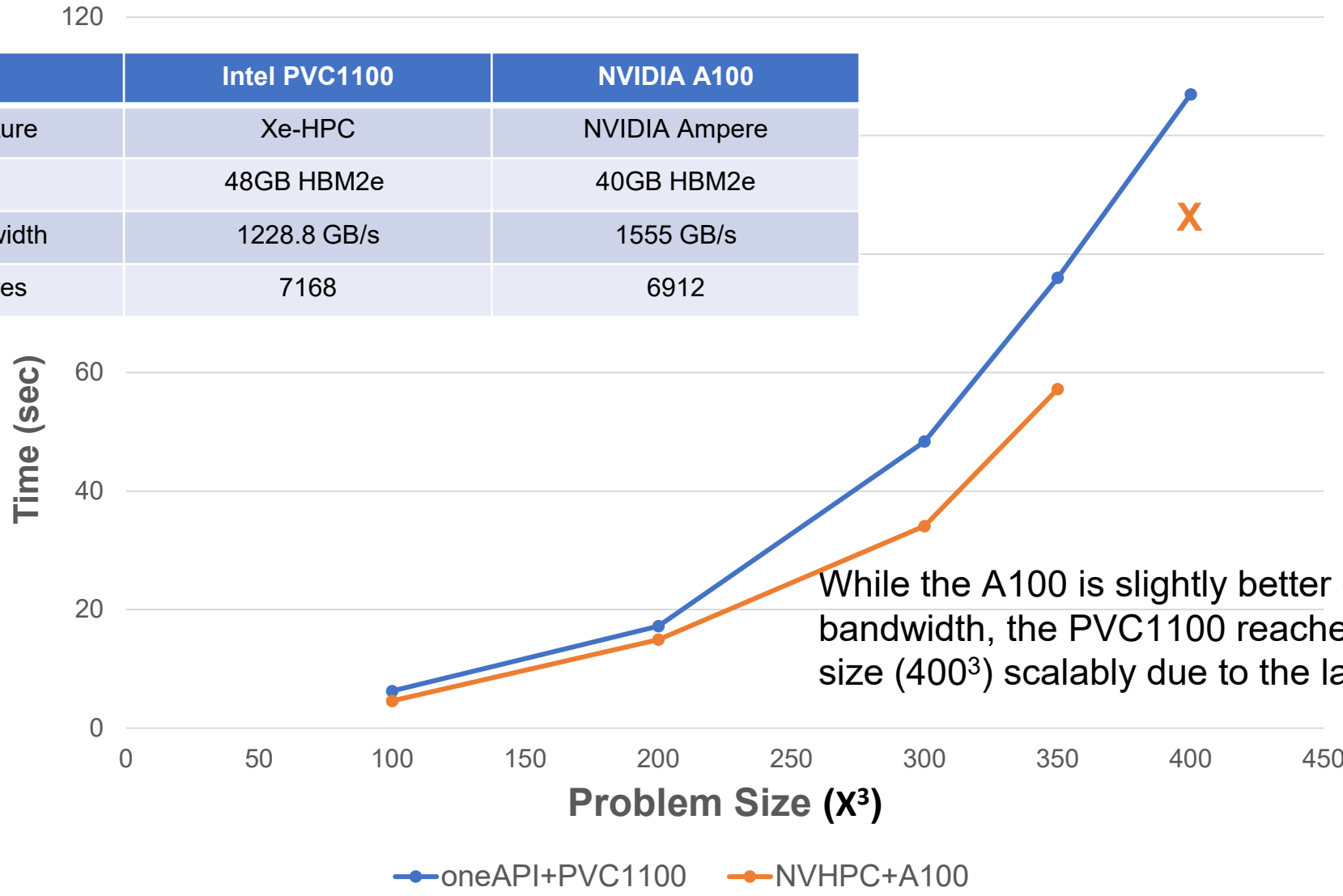
```
// Collect the data from the local arrays into the final force arrays
#pragma omp target data map(to: nodeElemStart[:len1], nodeElemCornerList[:len2]) if (USE_GPU == 1)
{

# pragma omp target teams if (USE_GPU == 1)
# pragma omp distribute parallel for
for( Index_t gnode=0 ; gnode<numNode ; ++gnode )
{
   // element count
   Index_t count = nodeElemStart[gnode+1] - nodeElemStart[gnode];//domain.nodeElemCount(gnode) ;
   // list of all corners
   Index_t *cornerList = &nodeElemCornerList[nodeElemStart[gnode]];//domain.nodeElemCornerList(gnode) ;
   Real_t fx_tmp = Real_t(0.0) ;
   Real_t fy_tmp = Real_t(0.0) ;
   Real_t fz_tmp = Real_t(0.0) ;
   for (Index_t i=0 ; i < count ; ++i) {
      Index_t elem = cornerList[i] ;
      fx_tmp += fx_elem[elem] ;
      fy_tmp += fy_elem[elem] ;
      fz_tmp += fz_elem[elem] ;
   }
   fx[gnode] += fx_tmp ;
   fy[gnode] += fy_tmp ;
   fz[gnode] += fz_tmp ;
}
}

#pragma omp target exit data map(delete:fx_elem[:numElem8], fy_elem[:numElem8], fz_elem[:numElem8]) if(USE_GPU == 1)
```

DEMO: OpenMP Offload to Intel PVC1100 with oneAPI - LULESH



Total LULESH time for PVC1100 and A100 (lower is better)

	Intel PVC1100	NVIDIA A100
GPU Architecture	Xe-HPC	NVIDIA Ampere
Memory	48GB HBM2e	40GB HBM2e
Memory Bandwidth	1228.8 GB/s	1555 GB/s
Compute Cores	7168	6912



While the A100 is slightly better due to faster memory bandwidth, the PVC1100 reaches the next problem size (400^3) scalably due to the larger memory size.

Conclusions and Future Work

- **Hardware** – While there are further advanced GPUs, we reach convergence in performance (Moore's law, 5nm). Need more software support in optimization.
- **Software** - OpenMP is now advanced to support and optimize scientific workloads on GPUs. OpenMP covers enough functionality to be able to offload data to GPUs with optimal data movement.
- **Compiler** – Relatively speaking, good compilers support but lacking in main important directives. We believe according to the roadmap we see that this support will be given.
- **Application** – Lulesh with OpenMP5.

Portability and Scalability of OpenMP Offloading on State-of-the-art Accelerators

Yehonatan Fridman, Guy Tamir, Gal Oren