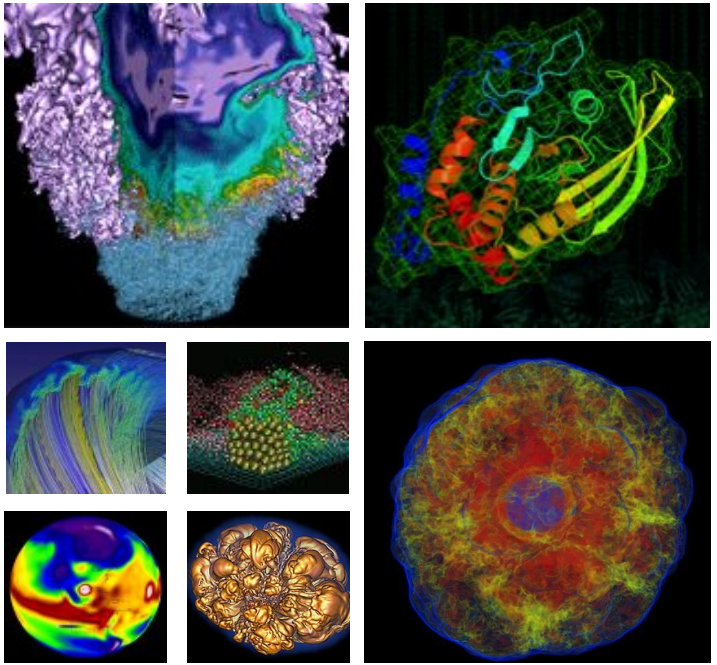
Programming Environments and Software



Rahul Gayatri

Talk by Johannes and Bjoern

- Multiple methods to login
  - ssh
  - NoMachine
- File systems
  - HOME
  - SCRATCH
  - CFS
  - HPSS
- Job submission
  - Thread/process affinity
  - Bundle jobs
  - Query a job status
- Workflow tools
- Superfacility API

Talk by Evan

- C++ sample code
  - OpenMP
    - OpenMP target (offload)
  - OpenACC
  - CUDA
- MPI + CUDA
- Kokkos

As soon as you login

```
(base) rahul-13:~ rgayatri$ ssh perlmutter
***************************************************************************
                          NOTICE TO USERS

Lawrence Berkeley National Laboratory operates this computer system under
contract to the U.S. Department of Energy.  This computer system is the
property of the United States Government and is for authorized use only.
Users (authorized or unauthorized) have no explicit or implicit
expectation of privacy.

Any or all uses of this system and all files on this system may be
intercepted, monitored, recorded, copied, audited, inspected, and disclosed
to authorized site, Department of Energy, and law enforcement personnel,
as well as authorized officials of other agencies, both domestic and foreign.
By using this system, the user consents to such interception, monitoring,
recording, copying, auditing, inspection, and disclosure at the discretion
of authorized site or Department of Energy personnel.

Unauthorized or improper use of this system may result in administrative
disciplinary action and civil and criminal penalties. By continuing to use
this system you indicate your awareness of and consent to these terms and
conditions of use. LOG OFF IMMEDIATELY if you do not agree to the conditions
stated in this warning.

*****************************************************************************


Login connection to host x3113c0s5b0n0:


################################################################################
                 _                 _   _
                | |               | | | |
 _ __   ___ _ __| |_ __ ___  _   _| |_| |_ ___ _ __
| '_ \ / _ \ '__| | '_ ` _ \| | | | __| __/ _ \ '__|
| |_) |  __/ |  | | | | | | | |_| | |_| ||  __/ |
| .__/ \___|_|  |_|_| |_| |_|\__,_|\__|\__\___|_|
| |
|_|

Hello, World!

Welcome to perlmutter!
################################################################################

For all planned outages, see: https://www.nersc.gov/live-status/motd/

For past outages, see: https://my.nersc.gov/outagelog-cs.php/
rgayatri@perlmutter:login02:~>
```

*Logging into the system : Covered by Johannes

In your environment

```
(base) rahul-13:~ rgayatri$ ssh perlmutter
*******************************************************************************
                          NOTICE TO USERS

Lawrence Berkeley National Laboratory operates this computer system under
contract to the U.S. Department of Energy.  This computer system is the
property of the United States Government and is for authorized use only.
Users (authorized or unauthorized) have no explicit or implicit
expectation of privacy.

Any or all uses of this system and all files on this system may be
intercepted, monitored, recorded, copied, audited, inspected, and disclosed
to authorized site, Department of Energy, and law enforcement personnel,
as well as authorized officials of other agencies, both domestic and foreign.
By using this system, the user consents to such interception, monitoring,
recording, copying, auditing, inspection, and disclosure at the discretion
```

```
rgayatri@perlmutter:login07:~> module list

Currently Loaded Modules:
  1) craype-x86-milan                     5) PrgEnv-gnu/8.3.3        9) craype/2.7.19          13) xalt/2.10.2      17) craype-accel-nvidia80
  2) libfabric/1.15.2.0                   6) cray-dsmml/0.2.2       10) gcc/11.2.0             14) Nsight-Compute/2022.1.1  18) gpu/1.0
  3) craype-network-ofi                   7) cray-libsci/23.02.1.1  11) perftools-base/23.02.0 15) Nsight-Systems/2022.2.
  4) xpmem/2.5.2-2.4_3.30__gd0f7936.shasta  8) cray-mpich/8.1.24    12) cpe/23.02              16) cudatoolkit/11.7
```

```
|_|

Hello, World!

Welcome to perlmutter!
#################################################################################

For all planned outages, see: https://www.nersc.gov/live-status/motd/

For past outages, see: https://my.nersc.gov/outagelog-cs.php/
rgayatri@perlmutter:login02:~>
```

Modules Environment

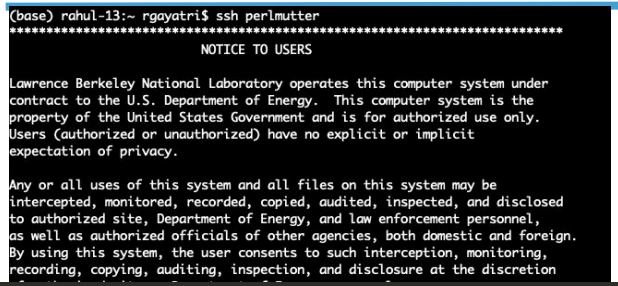
- Modules are used to manage the user environment
  - https://docs.nersc.gov/environment/#nersc-modules-environment

`module`	
`list`	To list the modules in your environment

`avail`
 	
`avail -S`
	To list available modules
	To see all available modules: `% module avail`
	To see all available *netcdf* modules: `% module spider netcdf`

`load/unload`	To load or unload module
`show/display`	To see what a module loads
`whatis`	Display the module file information
`swap/switch`	To swap two modules For example: to swap architecture target from Haswell to KNL `% module swap craype-haswell craype-mic-knl`
`help`	General help: `$module help` Information about a module: `$ module help PrgEnv-cray`

Default Programming Environment




```
rgayatri@perlmutter:login07:~> module list

Currently Loaded Modules:
  1) craype-x86-milan                      5) PrgEnv-gnu/8.3.3      9) craype/2.7.19          13) xalt/2.10.2      17) craype-accel-nvidia80
  2) libfabric/1.15.2.0                    6) cray-dsmml/0.2.2      10) gcc/11.2.0            14) Nsight-Compute/2022.1.1  18) gpu/1.0
  3) craype-network-ofi                    7) cray-libsci/23.02.1.1 11) perftools-base/23.02.0 15) Nsight-Systems/2022.2.
  4) xpmem/2.5.2-2.4_3.30__gd0f7936.shasta 8) cray-mpich/8.1.24     12) cpe/23.02             16) cudatoolkit/11.7
```

Compile simple code (GNU)

C++ : g++ main.cpp

C : gcc main.c

Fortran : gfortran main.F90

```cpp
int main(int argc, char **argv) {

    const int N = argc > 1 ? atoi(argv[1]) : 10;

    // host arrays
    int *a = new int[N];
    int *b = new int[N];
    int *c = new int[N];

    // initialize inputs
    random_ints(a, N);
    random_ints(b, N);

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

    // cleanup
    delete[] a;
    delete[] b;
    delete[] c;

    return 0;
}
```

main.cpp

Multiple PrgEnv

```
rgayatri@perlmutter:login24:~> module avail PrgEnv

------------------------ /opt/cray/pe/lmod/modulefiles/core -----------------------------------
   PrgEnv-aocc/8.3.3    PrgEnv-cray/8.3.3    PrgEnv-gnu/8.3.3 (L)    PrgEnv-nvhpc/8.3.3    PrgEnv-nvidia/8.3.3

  Where:
   L:  Module is loaded
```

Compiler wrappers map to the native compilers of the programming environment.

CC : C++ code
cc : C code
ftn : Fortran code

```
rgayatri@perlmutter:login24:~> CC --version
g++ (GCC) 11.2.0 20210728 (Cray Inc.)
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

rgayatri@perlmutter:login24:~> module load PrgEnv-nvidia

Lmod is automatically replacing "gcc/11.2.0" with "nvidia/22.7".


Lmod is automatically replacing "PrgEnv-gnu/8.3.3" with "PrgEnv-nvidia/8.3.3".


Due to MODULEPATH changes, the following have been reloaded:
  1) cray-mpich/8.1.24

rgayatri@perlmutter:login24:~> CC --version

nvc++ 22.7-0 64-bit target on x86-64 Linux -tp zen3-64
NVIDIA Compilers and Tools
Copyright (c) 2022, NVIDIA CORPORATION & AFFILIATES.  All rights reserved.
rgayatri@perlmutter:login24:~>
```

Compile simple code (Wrapper)

C++ : CC main.cpp
C : cc main.c
Fortran : ftn main.F90

```cpp
int main(int argc, char **argv) {

    const int N = argc > 1 ? atoi(argv[1]) : 10;

    // host arrays
    int *a = new int[N];
    int *b = new int[N];
    int *c = new int[N];

    // initialize inputs
    random_ints(a, N);
    random_ints(b, N);

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

    // cleanup
    delete[] a;
    delete[] b;
    delete[] c;

    return 0;
}
```

main.cpp

Compile MPI code

C++ : CC main.cpp
C : cc main.c
Fortran : ftn main.F90

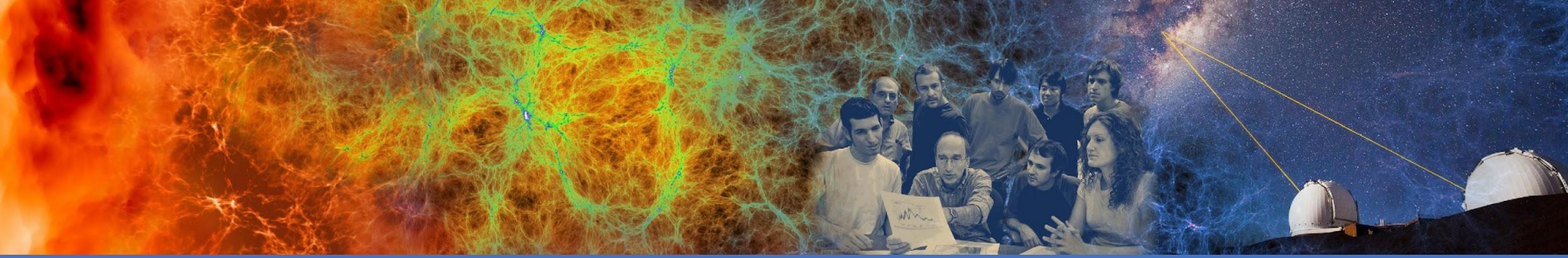
main.cpp

```
int main(int argc, char* argv[]) {
    int myid, namelen, world_size;
    char myname[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Get_processor_name(myname, &namelen);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    char* lrank = getenv("SLURM_PROCID");

    printf("Lrank from MPI = %s", lrank);

    char my_gpu[15];

    fprintf(stdout,
            "Hello from processor %s, rank = %d out of %d processors"
            "\n",
            myname, myid, world_size);

    // synchronize so the loop guarantees to prints all the information of one
    // rank before progressing.
    MPI_Barrier(MPI_COMM_WORLD);

    fprintf(stdout,
            "\n*************************************************************"
            "****************** \n");
    MPI_Finalize();

    return 0;
}
```

Software Environment (Summary)

- Available compilers: GNU, NVIDIA, AOCC, Cray
- Use compiler wrappers to build. It calls native compilers for each compiler (such as ifort, mpiicc, etc.) underneath.
  - Do not use native compilers directly.
  - ftn for Fortran codes: **ftn my_code.F90**
  - cc for C codes: **cc my_code.c**
  - CC for C++ codes: **CC my_code.cc**
- Compiler wrappers add header files and link in MPI and other loaded Cray libraries by default
  - Builds applications dynamically by default. Can add “-static” to build statically if chosen

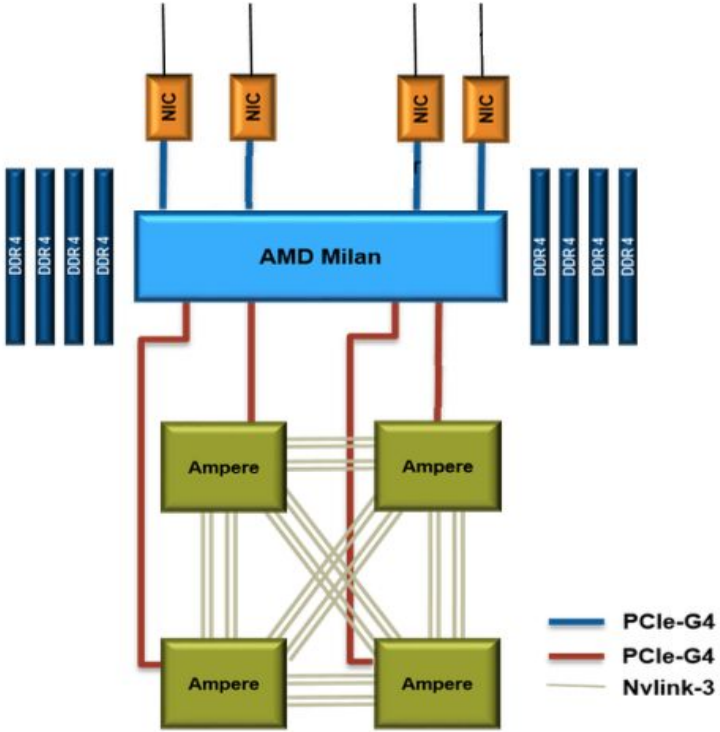
*Slide courtesy Helen (2020 NERSC Training)



GPU programming frameworks

CUDA, OpenMP, OpenACC

GPU node

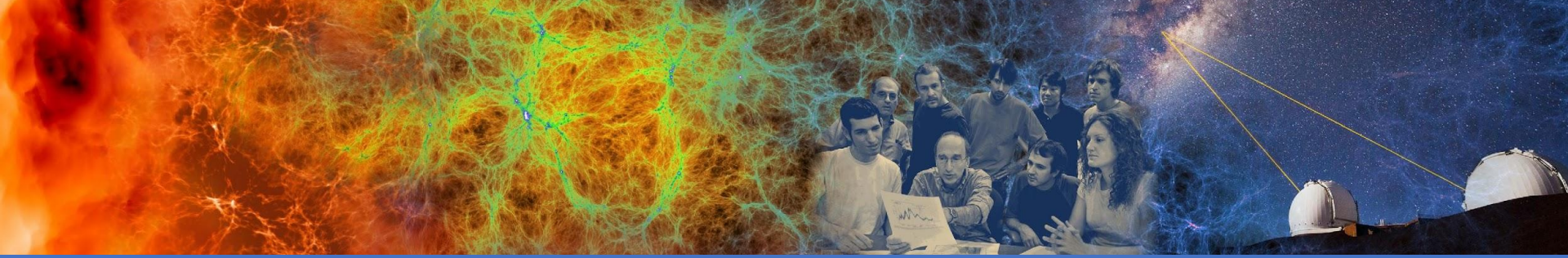


1 AMD Milan CPU - Host

4 NVIDIA A100 GPUs - Device

Nvlink-3 : 50 Gbit/s
PICe-G4 : 16 GT/s

*Figure courtesy Johannes (IPAM - 2023)



CUDA

CUDA module

```
rgayatri@perlmutter:login24:~> module show cudatoolkit
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------
   /opt/cray/pe/lmod/modulefiles/core/cudatoolkit/11.7.lua:
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------
family("cudatoolkit")
conflict("libsci_acc")
conflict("nvhpc")
conflict("nvhpc-nompi")
conflict("nvhpc-byo-compiler")
conflict("cudatoolkit")
help([[
The module file defines the system paths and variables for the HPC SDK Toolkit.

]])
whatis("The module file defines the system paths and variables for the SDK Toolkit.")
setenv("CUDATOOLKIT_HOME","/opt/nvidia/hpc_sdk/Linux_x86_6
setenv("CUDA_HOME","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/
setenv("NVHPC_CUDA_HOME","/opt/nvidia/hpc_sdk/Linux_x86_64
prepend_path("PATH","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5
bnvvp:/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/profilers/Nsight_Compute/:/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/profilers/Nsigh
t_Systems/bin"
prepend_path("MANPATH","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/doc/man")
```

CUDA - Native programming model for NVIDIA GPUs
nvcc - cuda compiler

Compile CUDA (separate files)

main.cu

```
__global__ void kernel(int N, int *a, int *b, int *c)
{
    const int i= blockIdx.x * blockDim.x + threadIdx.x;
    if(i < N)
        c[i] = a[i] + b[i];
}

void cuda_kernel(int N, int *a, i
{
    const int threads = 32;
    const int blocks = N/threads + 1;
    kernel<<<blocks,threads>>> (N, a, b, c);
    cudaDeviceSynchronize();
}
```

main.cpp

```
int main(int argc, char **argv) {
    const int N = argc > 1 ? atoi(argv[1]) : 10;

    int *a = new int[N];
    int *b = new int[N];
    int *c = new int[N];

    random_ints(a, N);
    random_ints(b, N);

    int *d_a, *d_b, *d_c;
    cudaMalloc(&d_a, N * sizeof(int));
    cudaMalloc(&d_b, N * sizeof(int));
    cudaMalloc(&d_c, N * sizeof(int));

    cudaMemcpy(d_a, a, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaDeviceSynchronize();

    cuda_kernel(N, d_a, d_b, d_c);

    daMemcpy(c, d_c, N * sizeof(int), cudaMemcpyDeviceToHost);
    daDeviceSynchronize();

    lete[] a;
    delete[] b;
    delete[] c;

    cudaFree(d_a);
    cudaFree(d_b);
    cudaFree(d_c);

    return 0;
}
```

nvcc **-O3 -o main.ex main.cpp main.cu**

CUDA compiler

PrgEnv-GNU

Compile CUDA (same file)

PrgEnv-GNU

CUDA kernel

nvcc **--x cu** **-O3 -o main.ex main.cpp**

CUDA compiler

CUDA code in cpp file

kernel call

main.cpp

```
__global__ void kernel(int N, int *a, int *b, int *c)
{
    const int i= blockIdx.x * blockDim.x + threadIdx.x;
    if(i < N)
        c[i] = a[i] + b[i];
}

nt main(int argc, char **argv) {
  ...

    cudaMemcpy(d_a, a, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaDeviceSynchronize();

    const int threads = 32;
    const int blocks = N/threads + 1;
    kernel<<<blocks,threads>>> (N, d_a, d_b, d_c);
    cudaDeviceSynchronize();

    ...

    return 0;
}
```

Compile CUDA (same file)

PrgEnv-NVIDIA

cc

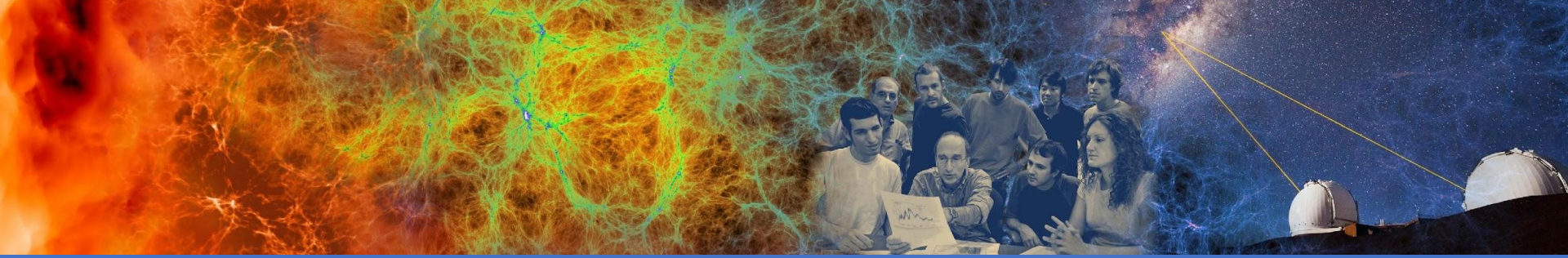
nvc++ **--x cu** **-O3 -o main.ex main.cpp**

CUDA compiler

CUDA code in cpp file

main.cpp

```
__global__ void kernel(int N, int *a, int *b, int *c)
{
    const int i= blockIdx.x * blockDim.x + threadIdx.x;
    if(i < N)
        c[i] = a[i] + b[i];


ain(int argc, char **argv) {
...

    cudaMemcpy(d_a, a, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaDeviceSynchronize();

    const int threads = 32;
    const int blocks = N/threads + 1;
    kernel<<<blocks,threads>>> (N, d_a, d_b, d_c);
    cudaDeviceSynchronize();

    ...

    return 0;
}
```



OpenMP

OpenMP

- Traditionally for CPUs
- GPU directives from OpenMP standard 4.0
  - `target` keyword
- Current standard - 5.2
- Enjoys support across multiple vendors
  - At Least on CPUs ;-)

21

OpenMP CPU

_OPENMP : compiler macro

GNU, Cray, AOCC

CC -O3 -fopenmp -o main.ex main.cpp

NVIDIA

CC -O3 -mp -o main.ex main.cpp

```
int main(int argc, char **argv) {
    const int N = argc > 1 ? atoi(argv[1]) : 10;

    int *a = new int[N];
    int *b = new int[N];
    int *c = new int[N];

    random_ints(a, N);
    random_ints(b, N);

#if defined(_OPENMP)
#pragma omp parallel for num_threads(64)
#endif
    for (int i = 0; i < N; i++)
        c[i] = a[i] + b[i];

    delete[] a;
    delete[] b;
    delete[] c;

    return 0;
}
```

OMP_NUM_THREADS

OpenMP CPU (summary)

Module	Compiler	Flags
PrgEnv-gnu	GCC	-fopenmp
PrgEnv-aocc	llvm	-fopenmp
PrgEnv-cray	CC	-fopenmp
PrgEnv-nvidia	nvc++	-mp
PrgEnv-intel	icpx	-fiopenmp
PrgEnv-llvm	clang++	-fopenmp

CC = compiler

OpenMP GPU

Cray

CC -O3 -fopenmp -o main.ex main.cpp

NVIDIA

CC -O3 -mp=gpu -o main.ex main.cpp

LLVM (NPE)

clang++ -O3 -fopenmp --offload-arch=sm_80 -o main.ex main.cpp

```
int main(int argc, char **argv) {
    const int N = argc > 1 ? atoi(argv[1]) : 10;

    int *a = new int[N];
    int *b = new int[N];
    int *c = new int[N];

    random_ints(a, N);
    random_ints(b, N);

#if defined(_OPENMP)
#pragma omp target teams distribute parallel for \
    num_teams(512) num_threads(32) \
    map(to: a[:N], b[:N]) map(from: c[:N])
#endif
    for (int i = 0; i < N; i++)
        c[i] = a[i] + b[i];

    delete[] a;
    delete[] b;
    delete[] c;

    return 0;
}
```

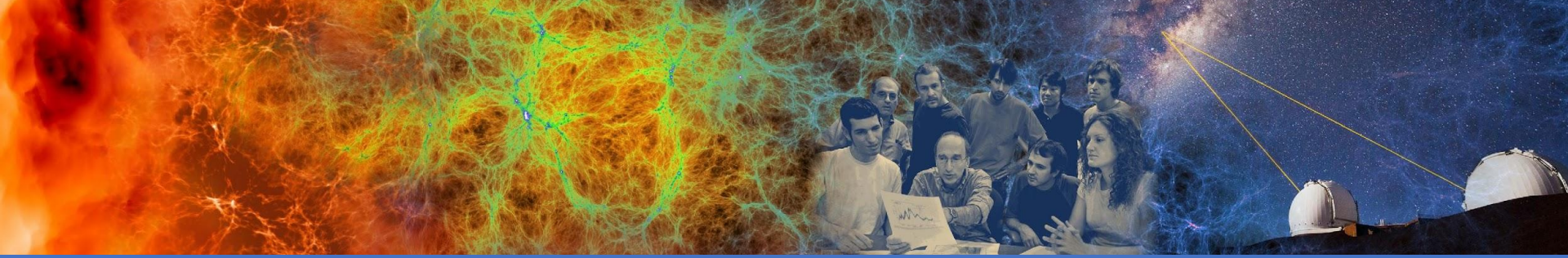
OMP_NUM_TEAMS
OMP_NUM_THREADS

OpenMP GPU (summary)

Module	Compiler	Flags (offload to GPUs)
PrgEnv-cray	CC	-fopenmp
PrgEnv-nvidia	CC/nvc++	-mp=gpu
PrgEnv-llvm NPE - NERSC Documentation	clang++	-fopenmp --offload-arch=sm_80

module load llvm/16

module load llvm/nightly



OpenACC

OpenACC directives

Cray

Only fortran

NVIDIA

CC -O3 -acc -o main.ex main.cpp

```
int main(int argc, char **argv) {
    const int N = argc > 1 ? atoi(argv[1]) : 10;

    int *a = new int[N];
    int *b = new int[N];
    int *c = new int[N];

    random_ints(a, N);
    random_ints(b, N);

#if defined(_OPENACC)
#pragma acc parallel loop gang vector \
    copyin(a[:N], b[:N]) copyout(c[:N])
#endif
    for (int i = 0; i < N; i++)
        c[i] = a[i] + b[i];

    delete[] a;
    delete[] b;
    delete[] c;

    return 0;
}
```

OpenACC

Module	Compiler	Flags (offload to GPUs)
PrgEnv-nvidia	CC/nvc++	-acc

Module	Compiler	Flags (CPU)
PrgEnv-nvidia	CC/nvc++	-acc=multicore

MPI+CUDA

MPI

CUDA

```
int main(int argc, char **argv) {
    const int N = argc > 1 ? atoi(argv[1]) : 10;
    int myid, namelen, world_size;
    char myname[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Get_processor_name(myname, &namelen);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    char* lrank = getenv("SLURM_PROCID");

    printf("Lrank from MPI = %s", lrank);

    char my_gpu[15];

    fprintf(stdout,
            "Hello Rahul from processor %s, rank = %d out of %d processors"
            "\n",
            myname, myid, world_size);

    // synchronize so the loop guarantees to prints all the information of one
    // rank before progressing.
    MPI_Barrier(MPI_COMM_WORLD);

    ...

    cudaMemcpy(d_a, a, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaDeviceSynchronize();

    const int threads = 32;
    const int blocks = N/threads + 1;
    kernel<<<blocks,threads>>> (N, d_a, d_b, d_c);
    cudaDeviceSynchronize();

    ...

    fprintf(stdout,
            "\n*****************************************************************"
            "******************* \n");
    MPI_Finalize();

    return 0;
}
```

MPI+CUDA

PrgEnv-gnu

MPI

```
int main(int argc, char **argv) {
    const int N = argc > 1 ? atoi(argv[1]) : 10;
    int myid, namelen, world_size;
    char myname[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Get_processor_name(myname, &namelen);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    char* lrank = getenv("SLURM_PROCID");

    printf("Lrank from MPI = %s", lrank);

    char my_gpu[15];

    hul from processor %s, rank = %d out of %d processors"

    yid, world_size);

    // synchronize so the loop guarantees to prints all the information of one
    // rank before progressing.
    MPI_Barrier(MPI_COMM_WORLD);

    ...

    cudaMemcpy(d_a, a, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaDeviceSynchronize();

    const int threads = 32;
    const int blocks = N/threads + 1;
    kernel<<<blocks,threads>>> (N, d_a, d_b, d_c);
    cudaDeviceSynchronize();

    ...

    fprintf(stdout,
            "\n*****************************************************************"
            "******************* \n");
    MPI_Finalize();

    return 0;
}
```

nvcc **-ccbin CC** **--x cu -O3 -o main.ex main.cpp**

CUDA compiler

Host compiler

CUDA

MPI+CUDA

PrgEnv-nvidia

MPI

CC --x cu -O3 -o main.ex main.cpp

Compiler wrapper

CUDA

```
int main(int argc, char **argv) {
    const int N = argc > 1 ? atoi(argv[1]) : 10;
    int myid, namelen, world_size;
    char myname[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Get_processor_name(myname, &namelen);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    char* lrank = getenv("SLURM_PROCID");

    printf("Lrank from MPI = %s", lrank);

    char my_gpu[15];

    fprintf(stdout,
            "Hello Rahul from processor %s, rank = %d out of %d processors"
            "\n",
            myname, myid, world_size);

    // synchronize so the loop guarantees to prints all the information of one
    // rank before progressing.
    MPI_Barrier(MPI_COMM_WORLD);

    ...

    cudaMemcpy(d_a, a, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaDeviceSynchronize();

    const int threads = 32;
    const int blocks = N/threads + 1;
    kernel<<<blocks,threads>>> (N, d_a, d_b, d_c);
    cudaDeviceSynchronize();

    ...

    fprintf(stdout,
            "\n*****************************************************************"
            "******************* \n");
    MPI_Finalize();

    return 0;
}
```

MPI+OpenMP (GPU)

PrgEnv-nvidia

CC -mp=gpu -O3 -o main.ex main.cpp

PrgEnv-cray

CC -fopenmp -O3 -o main.ex main.cpp

```
int main(int argc, char **argv) {
    const int N = argc > 1 ? atoi(argv[1]) : 10;
    int myid, namelen, world_size;
    char myname[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Get_processor_name(myname, &namelen);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    char* lrank = getenv("SLURM_PROCID");

    printf("Lrank from MPI = %s", lrank);

    char my_gpu[15];

    fprintf(stdout,
            "Hello Rahul from processor %s, rank = %d out of %d processors"
            "\n",
            myname, myid, world_size);

    // synchronize so the loop guarantees to prints all the information of one
    // rank before progressing.
    MPI_Barrier(MPI_COMM_WORLD);

    int *a = new int[N];
    int *b = new int[N];
    int *c = new int[N];

    random_ints(a, N);
    random_ints(b, N);

#if defined(_OPENMP)
#pragma omp target teams distribute parallel for \
    map(to: a[:N], b[:N]) map(from: c[:N])
#endif
    for (int i = 0; i < N; i++)
        c[i] = a[i] + b[i];


    delete[] a;
    delete[] b;
    delete[] c;

    fprintf(stdout,
            "\n*****************************************************************"
            "******************** \n");
    MPI_Finalize();

    return 0;
}
```

MPI+OpenMP (GPU)

PrgEnv-gnu + llvm/16 + cray-mpich

```
clang++ -fopenmp --offload-arch=sm_80   \
-I$CRAY_MPICH_PREFIX/include            \
-L$CRAY_MPICH_PREFIX/lib -lmpi          \
$PE_MPICH_GTL_DIR_nvidia80              \
$PE_MPICH_GTL_LIBS_nvidia80              \
-O3 -o main.ex main.cpp
```

```cpp
int main(int argc, char **argv) {
    const int N = argc > 1 ? atoi(argv[1]) : 10;
    int myid, namelen, world_size;
    char myname[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Get_processor_name(myname, &namelen);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    char* lrank = getenv("SLURM_PROCID");

    printf("Lrank from MPI = %s", lrank);

    char my_gpu[15];

    fprintf(stdout,
            "Hello Rahul from processor %s, rank = %d out of %d processors"
            "\n",
            myname, myid, world_size);

    // synchronize so the loop guarantees to prints all the information of one
    // rank before progressing.
    MPI_Barrier(MPI_COMM_WORLD);

    int *a = new int[N];
    int *b = new int[N];
    int *c = new int[N];

    random_ints(a, N);
    random_ints(b, N);

#if defined(_OPENMP)
#pragma omp target teams distribute parallel for \
    map(to: a[:N], b[:N]) map(from: c[:N])
#endif
    for (int i = 0; i < N; i++)
        c[i] = a[i] + b[i];


    delete[] a;
    delete[] b;
    delete[] c;

    fprintf(stdout,
            "\n*****************************************************************"
            "******************** \n");
    MPI_Finalize();

    return 0;
}
```

MPI + X

```
#if _OPENACC
#pragma acc parallel loop gang vector \
    copyin(a[:N], b[:N]) copyout(c[:N])
#endif
    for (int i = 0; i < N; i++)
        c[i] = a[i] + b[i];
```

OpenACC

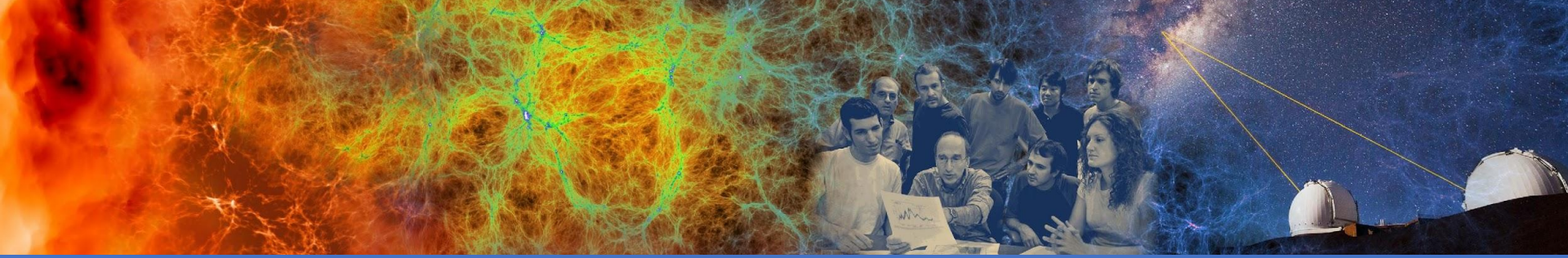
```
#if defined(_OPENMP)
#pragma omp target teams distribute parallel for \
    map(to: a[:N], b[:N]) map(from: c[:N])
#endif
    for (int i = 0; i < N; i++)
        c[i] = a[i] + b[i];
```

OpenMP

```
__global__ void kernel(int N, int *a, int *b, int *c)
{
    const int i= blockIdx.x * blockDim.x + threadIdx.x;
    if(i < N)
        c[i] = a[i] + b[i];
}

void cuda_kernel(int N, int *a, int *b, int *c)
{
    const int threads = 32;
    const int blocks = N/threads + 1;
    kernel<<<blocks,threads>>> (N, a, b, c);
}
```

CUDA

```
int main(int argc, char **argv) {
    int myid, namelen, world_size;
    char myname[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Get_processor_name(myname, &namelen);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    char* lrank = getenv("SLURM_PROCID");

    printf("Lrank from MPI = %s", lrank);

    fprintf(stdout,
            "Hello from processor %s, rank = %d out of %d processors"
            "\n",
            myname, myid, world_size);

    MPI_Barrier(MPI_COMM_WORLD);

    const int N = argc > 1 ? atoi(argv[1]) : 10;

    int *a = new int[N];
    int *b = new int[N];
    int *c = new int[N];

    random_ints(a, N);
    random_ints(b, N);

    // do work
    for (int i = 0; i < N; i++)
        c[i] = a[i] + b[i];

    delete[] a;
    delete[] b;
    delete[] c;

    MPI_Finalize();

    return 0;
}
```



Portability Frameworks / Softwares

Portability Frameworks

- Kokkos
  - CUDA (nvcc,nvc++)
  - OpenMPTarget (llvm,nvc++)
- RAJA
  - CUDA (-DCUDA_TOOLKIT_ROOT_DIR=path/to/cuda/toolkit -DCMAKE_CUDA_COMPILER=path/to/nvcc)
  - OpenMP target
- HIP
  - module load hip/5.3.2
- SYCL
  - SYCL - NERSC Documentation

Softwares available

CUDA
- Math libs
- CUPTI

```
rgayatri@perlmutter:login23:/pscratch/sd/r/rgayatri/IPAM-2023> module show cudatoolkit
-------------------------------------------------------------------------------------------------------------------------------
   /opt/cray/pe/lmod/modulefiles/core/cudatoolkit/11.7.lua:
-------------------------------------------------------------------------------------------------------------------------------
setenv("CUDATOOLKIT_HOME","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7")
setenv("CUDA_HOME","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7")
setenv("NVHPC_CUDA_HOME","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7")
prepend_path("PATH","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/bin:/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/l
ibnvvp:/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/profilers/Nsight_Compute/:/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/profilers/Nsi
ght_Systems/bin")
prepend_path("MANPATH","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/doc/man")
prepend_path("CRAY_LD_LIBRARY_PATH","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/lib64")
prepend_path("CRAY_LD_LIBRARY_PATH","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/math_libs/11.7/lib64")
prepend_path("LD_LIBRARY_PATH","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/lib64")
prepend_path("LD_LIBRARY_PATH","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/nvvm/lib64")
prepend_path("LD_LIBRARY_PATH","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/extras/Debugger/lib64")
prepend_path("LD_LIBRARY_PATH","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/extras/CUPTI/lib64")
prepend_path("LD_LIBRARY_PATH","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/math_libs/11.7/lib64")
append_path("PE_PRODUCT_LIST","CRAY_HPC_SDK")
setenv("CRAY_CUDATOOLKIT_VERSION","22.5_11.7")
setenv("CRAY_CUDATOOLKIT_DIR","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7")
setenv("CRAY_CUDATOOLKIT_PREFIX","/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7")
setenv("XTPE_LINK_TYPE","dynamic")
setenv("CRAYPE_LINK_TYPE","dynamic")
setenv("CRAY_CUDATOOLKIT_INCLUDE_OPTS","-I/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/include -
I/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/nvvm/include -
I/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/extras/Debugger/include -
I/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/extras/CUPTI/include -
I/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/math_libs/11.7/include")
setenv("CRAY_CUDATOOLKIT_POST_LINK_OPTS","-L/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/lib64 -
L/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/nvvm/lib64 -
L/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/extras/Debugger/lib64 -
L/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/extras/CUPTI/lib64 -
L/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/math_libs/11.7/lib64 -Wl,--as-needed,-lcupti,-lcudart,--no-as-needed -lcuda")
```

Softwares available

- FFTW
  - cray-fftw , cufft
- BLAS
  - cublas
- 3rd party softwares
  - Picture ->
  - LAPACK
  - LIBSCI
  - LAMMPS
  - VASP
  - Julia

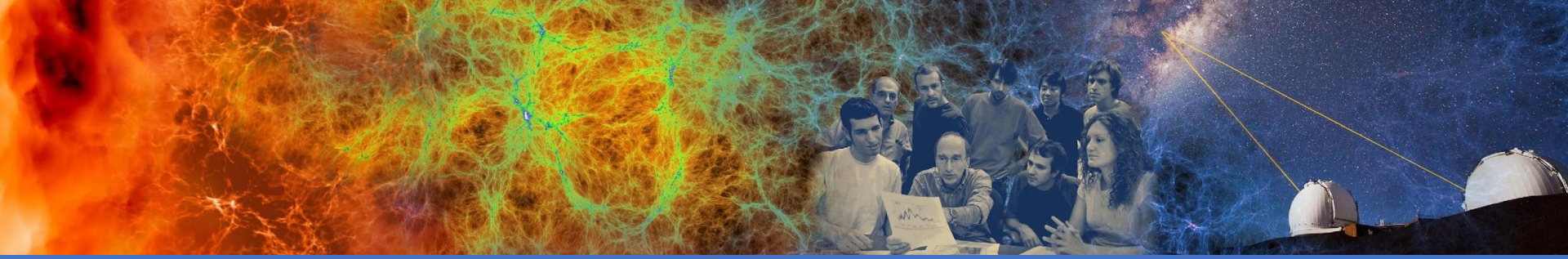
```
------------ /global/common/software/nersc/pm-2022.12.0/extra_modulefiles ----------
   allinea-forge/22.0.4-linux-x86_64        globus-tools/1.0              julia/1.7.2-gnu                pytorch/1.10.0
   allinea-forge/22.1.2-linux-x86_64        gpu-test/1.1                  julia/1.7.2-nvidia             pytorch/1.11.0
   allinea-forge/22.1.3-linux-x86_64 (D)    gpu/1.0               (L)     julia/1.7.2                    pytorch/1.13.1
          (D)
   amber/20-gpu                       (D)   gromacs/2021.5-plumed         julia/1.8.0-beta1-cray         qchem/5.4.2-
cpu
   amber/20                                 gromacs/2021.5                julia/1.8.0-beta1-gnu          qchem/6.0.0-
cpu            (D)
   arm-forge/22.0.4-linux-x86_64            gromacs/2022.3-gpu    (D)     julia/1.8.0-beta1-nvidia       spack/develop
   arm-forge/22.1.2-linux-x86_64            gsl/2.7                       julia/1.8.0-beta1          (D)   spack/e4s-
21.11
   arm-forge/22.1.3-linux-x86_64      (D)   hip/5.3.2                     lammps/2022.11.03              spack/e4s-
22.05
   berkeleygw/3.0.1-cpu                     idl/8.5                       llvm/nightly                   spack/e4s-
22.11
   berkeleygw/3.0.1-gpu               (D)   jamo/dev                      llvm/16                    (D)   spack/0.19.0
          (D)
   cmake/3.22.0                             jamo/prod             (D)     mathematica/13.0.1             spin/2.0
   cmake/3.24.3                       (D)   jgi/python-jamo               matlab/MCR_R2021b              taskfarmer/1.5
   codee/1.6.0                              julia/settings-cray           matlab/R2021b              (D)
 tensorflow/2.6.0
   codee/1.6.0-212                          julia/settings-gnu            mongodb/4.0.28
tensorflow/2.9.0            (D)
   codee/1.6.0-434                    (D)   julia/settings-nvidia         mongodb/100.5.2            (D)
 totalview/2022.2.13
   contrib/1.0                              julia/1.4.2-cray              mumps/5.5.1-gcc-11.2.0
totalview/2022.2.20
   cpu/1.0                                  julia/1.4.2-gnu               mvasp/5.4.4-cpu
 totalview/2022.4.27          (D)
   cudnn/8.2.0                              julia/1.4.2-nvidia            namd/2.15a2-cpu_mpi
 training/perlmutter-jan2022
   cudnn/8.3.2                              julia/1.4.2                   namd/2.15a2-cpu_ofi
 valgrind/3.18.1
   cudnn/8.7.0                        (D)   julia/1.5.4-cray              namd/2.15a2-gpu_ofi        (D)
 valgrind/3.20.0              (D)
   darshan/3.3.1-hdf5                       julia/1.5.4-gnu               nccl/2.11.4                    vasp-
tpc/5.4.4-cpu           (D)
   darshan/3.3.1                            julia/1.5.4-nvidia            nccl/2.14.3                    vasp-
tpc/6.2.1-gpu
   darshan/3.4.0-hdf5                       julia/1.5.4                   nccl/2.15.5-ofi            (D)   vasp-
tpc/6.3.2-cpu
   darshan/3.4.0                      (D)   julia/1.6.0-cray              Nsight-Compute/2022.1.1    (L)   vasp-
tpc/6.3.2-gpu
   e4s/21.11                                julia/1.6.0-gnu               Nsight-Systems/2022.2.1    (L)   vasp/5.4.1-cpu
   e4s/22.05                                julia/1.6.0-nvidia            nvidia-mixed/22.2              vasp/5.4.4-cpu
          (D)
   e4s/22.11                          (D)   julia/1.6.0                   nvidia/22.2                    vasp/6.2.1-gpu
   espresso/7.0-libxc-5.2.2-cpu             julia/1.6.5-cray              parallel/20210922              vasp/6.3.2-cpu
   espresso/7.0-libxc-5.2.2-gpu             julia/1.6.5-gnu               pwanalyzer/1.2.0               vasp/6.3.2-gpu
   espresso/7.1-libxc-6.1.0-cpu       (D)   julia/1.6.5-nvidia            pwanalyzer/1.2.1           (D)   xalt/2.10.2
          (L)
   evp-patch                                julia/1.6.5                   python/3.9-anaconda-2021.11
   fast-mkl-amd/fast-mkl-amd                julia/1.7.2-cray              pytorch/1.9.0
```

Machine Learning

- PyTorch
- Tensorflow

```
rgayatri@perlmutter:login05:~>
rgayatri@perlmutter:login05:~> ml avail pytorch

-------------------------------------------------------------------------------
   pytorch/1.9.0    pytorch/1.10.0    pytorch/1.11.0    pytorch/1.13.1 (D)

  Where:
   D:  Default Module
```

```
rgayatri@perlmutter:login05:~> ml avail tensorflow

-------------------------------------------------------
   tensorflow/2.6.0    tensorflow/2.9.0 (D)

  Where:
   D:  Default Module
```



CUDA-AWARE MPI

Concept

- MPI is aware of GPU memory being allocated inside a process
- Programmer can use pointers to GPU device memory in MPI buffers
- MPI implementation will correctly copy the data in GPU device memory to/from the network interface card's (NIC's) memory
    - implicitly copying the data first to host memory and then copying the data from host memory to the NIC
    - in hardware which supports GPUDirect RDMA, the data will be copied directly from the GPU to the NIC, bypassing host memory altogether
- Always on by default

Example

Each MPI rank creates a host and device memory for a single float variable

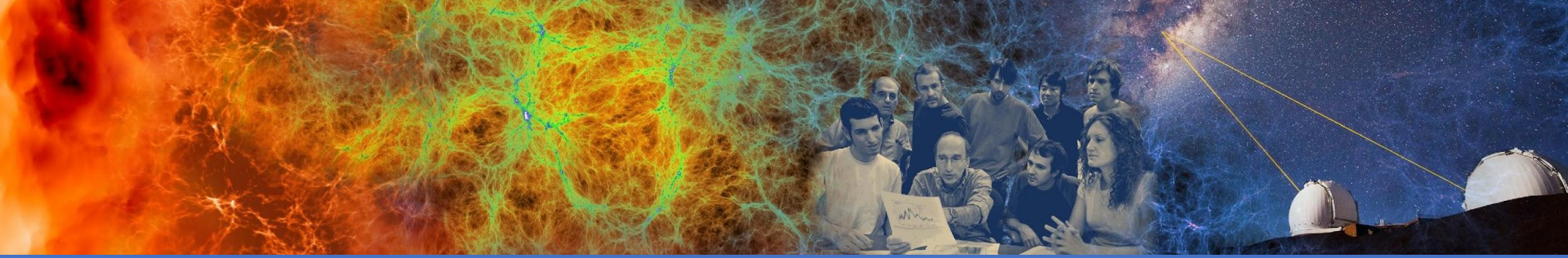
Every rank initializes

Only rank 0 assigns host to a value and copies it to the corresponding device buffer

Device buffer is passed to MPI broadcast

Every rank (except 0) copies the received device buffer value to corresponding host

```c
int main(int argc, char *argv[]) {
    int myrank;
    float *val_device, *val_host;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

    val_host = (float*)malloc(sizeof(float));
    cudaMalloc((void **)&val_device, sizeof(float));

    *val_host = -1.0;
    if (myrank != 0) {
      printf("%s %d %s %f\n", "I am rank", myrank, "and my initial value is:", *val_host);
    }

    if (myrank == 0) {
        *val_host = 42.0;
        cudaMemcpy(val_device, val_host, sizeof(float), cudaMemcpyHostToDevice);
        printf("%s %d %s %f\n", "I am rank", myrank, "and will broadcast value:", *val_host);
    }

    MPI_Bcast(val_device, 1, MPI_FLOAT, 0, MPI_COMM_WORLD);

    if (myrank != 0) {
      cudaMemcpy(val_host, val_device, sizeof(float), cudaMemcpyDeviceToHost);
      printf("%s %d %s %f\n", "I am rank", myrank, "and received broadcasted value:", *val_host);
    }

    cudaFree(val_device);
    free(val_host);

    MPI_Finalize();
    return 0;
}
```



Case Studies

NESAP learnings

- NESAP - NERSC Exascale Science Applications Program - NESAP
  - https://www.nersc.gov/research-and-development/nesap/
  - Collaborate with code teams to prepare them for exascale machines
- GPU case studies
- Some learnings from engagements with ECP teams

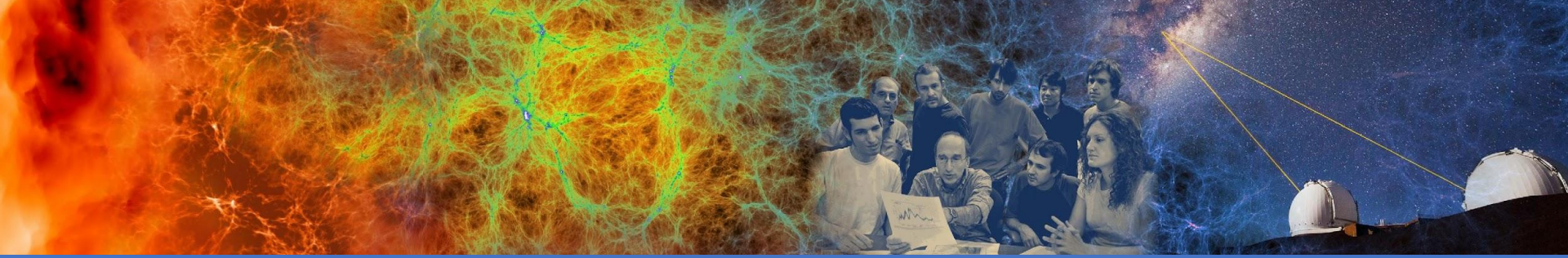
SNAP learnings

- Kernel fission and fusion
  - Fission - Reduce register pressure to improve occupancy
  - Fusion - Improve AI
- Increase parallelism by loop fusions
- Data layout optimizations
  - Row major - Beneficial on CPUs as it avoids false sharing in caches
  - Column major - Beneficial on GPUs as it improves memory Coalescing
- Scratch memory
- Transpose of data structures to switch between row-major and column major in between kernels
  - Transpose - 0.2% of the runtime
  - 20% net performance boost
- 128 bit load/store operations
  - Complex-double datatype (C++ alignas(16))
  - 15% overall performance boost
- Overall 26x speedup at present
- Archive paper && Gordon bell finalist in 2021

SNAP learnings

- Kernel fission and fusion
  - Fission - Reduce register pressure to improve occupancy
  - [illegible]
- Incre[illegible]me
- Data layout optimizations
  - Row major - Beneficial on CPUs as it avoids false sharing in caches
  - Column major - Beneficial on GPUs as it improves memory Coalescing
- Scratch memory
- Transpose of data structures to switch between row-major and column major in between kernels
  - [illegible]
- 128 bit load/store operations
  - Complex-double datatype (C++ alignas(16))
  - 15% overall performance boost
- Overall 26x speedup at present
- Archive paper && Gordon bell finalist in 2021

Perfection is the enemy of Good

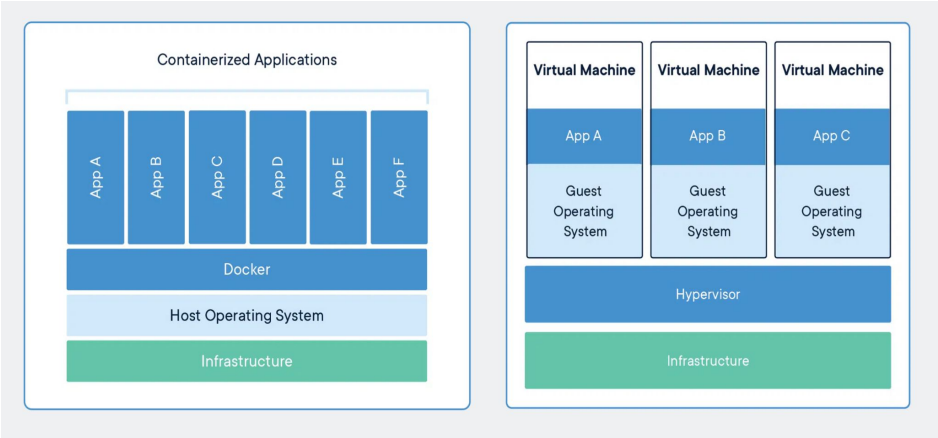


Containers

What is a container?

- A container is similar in purpose to a virtual machine (VM), providing encapsulation for a software application and its runtime environment.
- Implementation differs. Containers use the host linux kernel instead of virtualizing hardware, so they are lightweight compared to VMs.
- Linux Containers rely on kernel features, and are inherently Linux based.

Container



Virtual machine

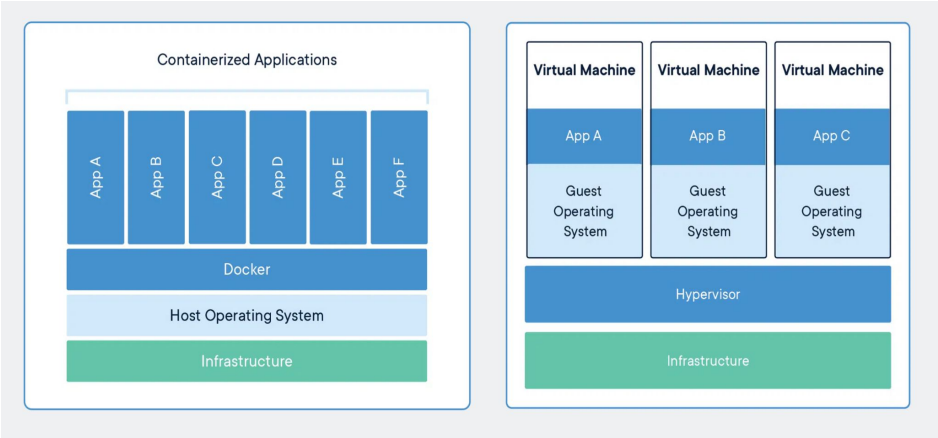
Image from https://www.docker.com/resources/what-container/

What is a container?

- A container is similar in purpose to a virtual machine (VM), providing encapsulation for a software application and it's runtime environment.
- Implementation differs. Containers use the host linux kernel instead of virtualizing hardware, s
- Linux Containers rely on kernel features, and are inherently Linux based.

Data Day 2022 (nersc.gov)

Container



Virtual machine

Image from https://www.docker.com/resources/what-container/

Docker

- Widely used container software
- Write **Dockerfile**
- Build **image** once
- Run multiple times

Podman

- Podman (Pod Manager) is an open-source, OCI-compliant container framework
- Actively developed by Red Hat.
- Podman can be treated as a drop-in replacement for Docker.
- **podman-hpc** is a special wrapper that so far is only at nersc
  - podman-hpc docs

Dockerfile

```
rgayatri@perlmutter:login23:/pscratch/sd/r/rgayatri/IPAM-2023/ipam-2023/dockerfile> cat Dockerfile
```

File: Dockerfile

```
1  FROM docker.io/nvidia/cuda:11.7.0-devel-ubuntu20.04
2  WORKDIR /opt
3
4  ENV DEBIAN_FRONTEND noninteractive
5
6  RUN \
7      apt-get update        && \
8      apt-get install --yes    \
9          --no-install-recommends \
10         build-essential      \
11         python3-dev          \
12         gfortran             \
13         git             \
14         wget              && \
15     apt-get install -y python3-pip && \
16     apt-get clean all
17
18 RUN \
19     cd /home && \
20     git clone https://github.com/rgayatri23/ipam-2023.git && \
21     cd ipam-2023 && \
22     cd container && \
23     nvcc -O3 -std=c++11 -o main.ex main.cu && \
24     ./main.ex
```

```
rgayatri@perlmutter:login23:/pscratch/sd/r/rgayatri/IPAM-2023/ipam-2023/dockerfile>
```

Podman build

```
rgayatri@perlmutter:login23:/pscratch/sd/r/rgayatri/IPAM-2023/ipam-2023/dockerfile> cat Dockerfile
File: Dockerfile
1  FROM docker.io/nvidia/cuda:11.7.0-devel-ubuntu20.04
2  WORKDIR /opt
3
4  ENV DEBIAN_FRONTEND noninteractive
5
6  RUN \
11         python3-dev          \
12         gfortran             \
13         git             \
14         wget              && \
15     apt-get install -y python3-pip && \
16     apt-get clean all
17
18 RUN \
19     cd /home && \
20     git clone https://github.com/rgayatri23/ipam-2023.git && \
21     cd ipam-2023 && \
22     cd container && \
23     nvcc -O3 -std=c++11 -o main.ex main.cu && \
24     ./main.ex
rgayatri@perlmutter:login23:/pscratch/sd/r/rgayatri/IPAM-2023/ipam-2023/dockerfile>
```

podman-hpc build -t ipam2023:cuda11.7 path-to-dockerfile

Podman images

podman-hpc images

```
rgayatri@perlmutter:login23:/pscratch/sd/r/rgayatri/IPAM-2023/ipam-2023/dockerfile> podman-hpc images
REPOSITORY              TAG                        IMAGE ID      CREATED       SIZE
localhost/ipam2023      cuda11.7                   d7f1fc033488  17 hours ago  5.04 GB
docker.io/nvidia/cuda   11.7.0-devel-ubuntu20.04   f441d4635450  3 months ago  4.86 GB
rgayatri@perlmutter:login23:/pscratch/sd/r/rgayatri/IPAM-2023/ipam-2023/dockerfile>
```

Podman run

```
podman-hpc run          \
 --rm --gpu             \
 -it ipam2023:cuda11.7  \
 /bin/bash
```

```
rgayatri@perlmutter:login23:/pscratch/sd/r/rgayatri/IPAM-2023> podman-hpc run --rm --gpu -it ipam2023:cuda11.7 /bin/bash

==========
== CUDA ==
==========

CUDA Version 11.7.0

Container image Copyright (c) 2016-2022, NVIDIA CORPORATION & AFFILIATES. All rights reserved.

This container image and its contents are governed by the NVIDIA Deep Learning Container License.
By pulling and using the container, you accept the terms and conditions of this license:
https://developer.nvidia.com/ngc/nvidia-deep-learning-container-license

A copy of this license is made available in this container at /NGC-DL-CONTAINER-LICENSE for your convenience.

************************
** DEPRECATION NOTICE! **
************************
THIS IMAGE IS DEPRECATED and is scheduled for DELETION.
    https://gitlab.com/nvidia/container-images/cuda/blob/master/doc/support-policy.md

root@f5ded3c7f395:/opt# cd /home/ipam-2023/container/
root@f5ded3c7f395:/home/ipam-2023/container# ./main.ex
c[0] = 2     - expected 2
c[1] = 6     - expected 6
c[2] = 12    - expected 12
c[3] = 20    - expected 20
c[4] = 30    - expected 30
c[5] = 42    - expected 42
c[6] = 56    - expected 56
c[7] = 72    - expected 72
c[8] = 90    - expected 90
c[9] = 110   - expected 110
root@f5ded3c7f395:/home/ipam-2023/container#
```

E4S

- E4S - Extreme-scale Scientific Software Stack
- Curated software stack based on spack
- Tested on multiple platforms and in docker environments
- We only test with PrgEnv-gnu currently

module load e4s lammps

Summary

- Compile MPI + GPU codes
  - CUDA / OpenMP / OpenACC
- Pass device pointers to MPI routines
- A wide variety of softwares available for usage
- Documentation is good/up-to-date
- Documentation on good programming practices
- Containers to build once and use multiple times

Thank You