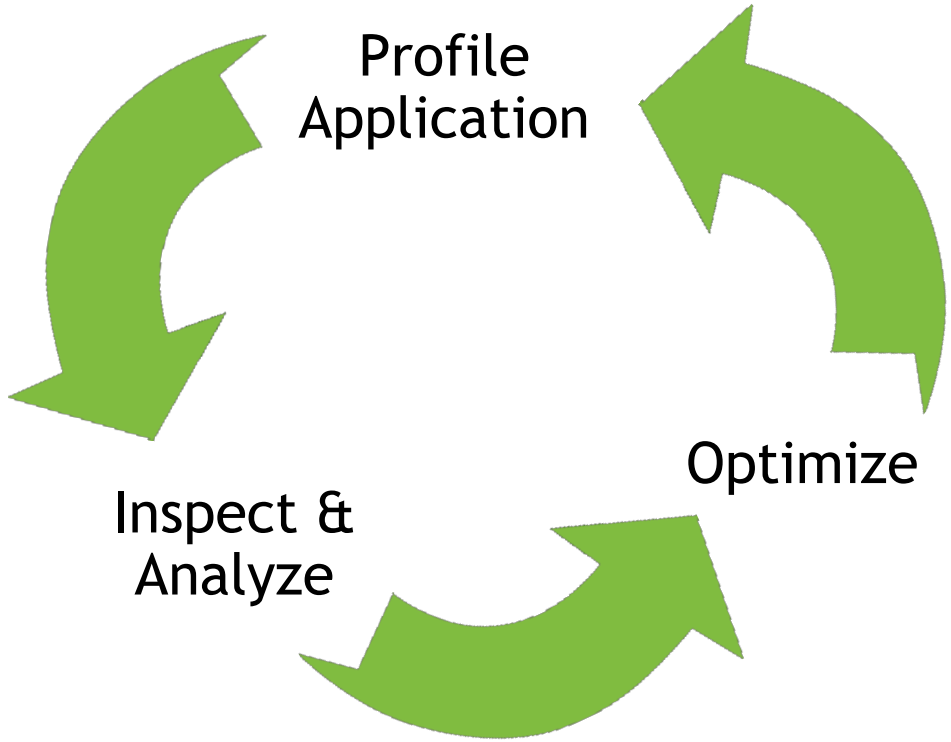
Nsight Systems – Introduction

OpenACC
More Science, Less Programming

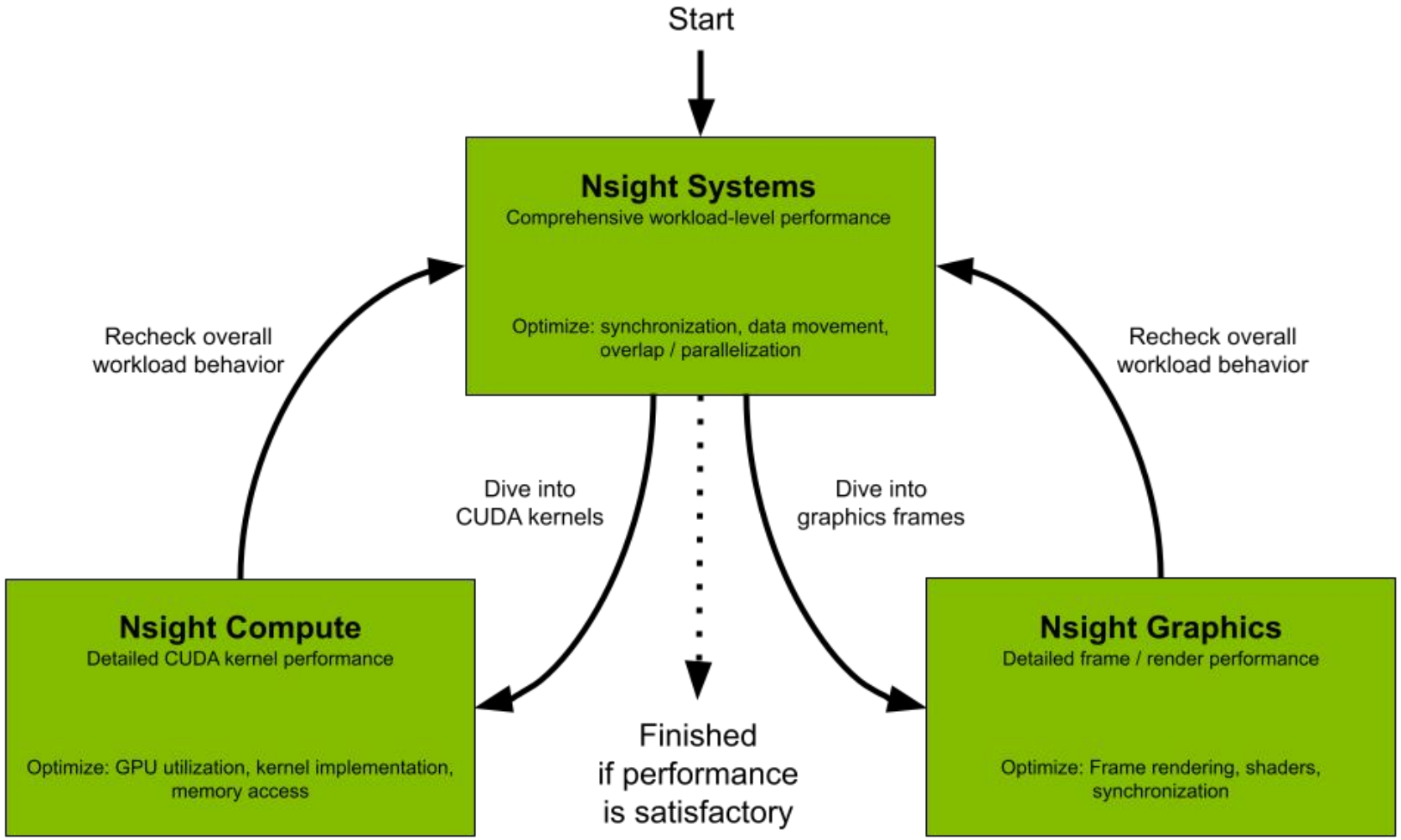
OPEN HACKATHONS

Typical Optimization Workflow



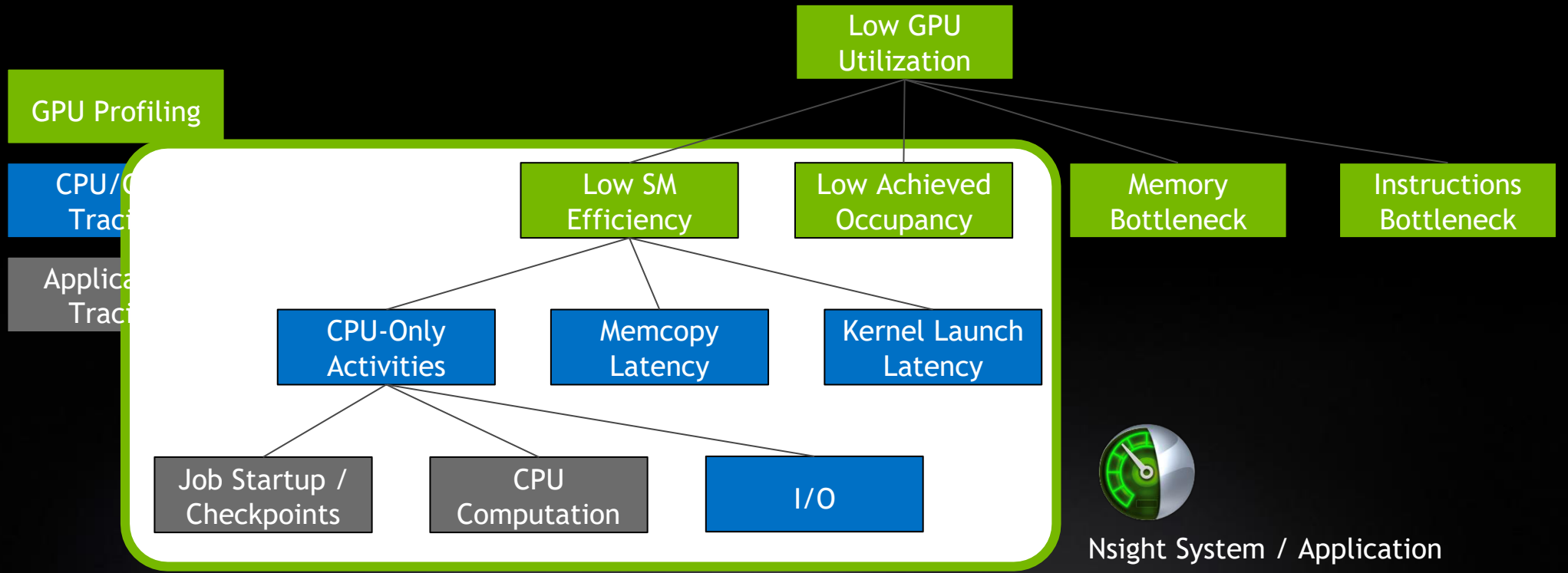
Iterate until desired performance is achieved

NVIDIA Performance Analysis Tools



PROFILING GPU APPLICATION

Focusing System Operation



Nsight System / Application Tracing

NVIDIA

PROFILING GPU APPLICATION

Focusing GPU Computing

GPU Profiling

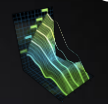
CPU/GPU Tracing

Application Tracing

Low GPU Utilization

- Low SM Efficiency
- Low Achieved Occupancy
  - Too few threads
  - Register limit
  - Large shared memory
  - …
- Memory Bottleneck
  - Cache misses
  - Bandwidth limit
  - Access pattern
  - …
- Instructions Bottleneck
  - Arithmetic
  - Control flow
  - …

NVIDIA (Visual) Profiler / Nsight Compute



NVIDIA Supports them with cuDNN, cuBLAS, and so on

System-Wide Application Tuning

Maximize your GPU Investment

Locate optimization opportunities

- Visualize millions of events on a timeline
- See gaps of unused CPU and GPU time

Balance your workload across multiple CPUs and GPUs

- CPU utilization and thread state
- GPU streams, kernels, memory transfers, etc.
- NIC performance metrics & tracing of network libraries

Multi-platform support

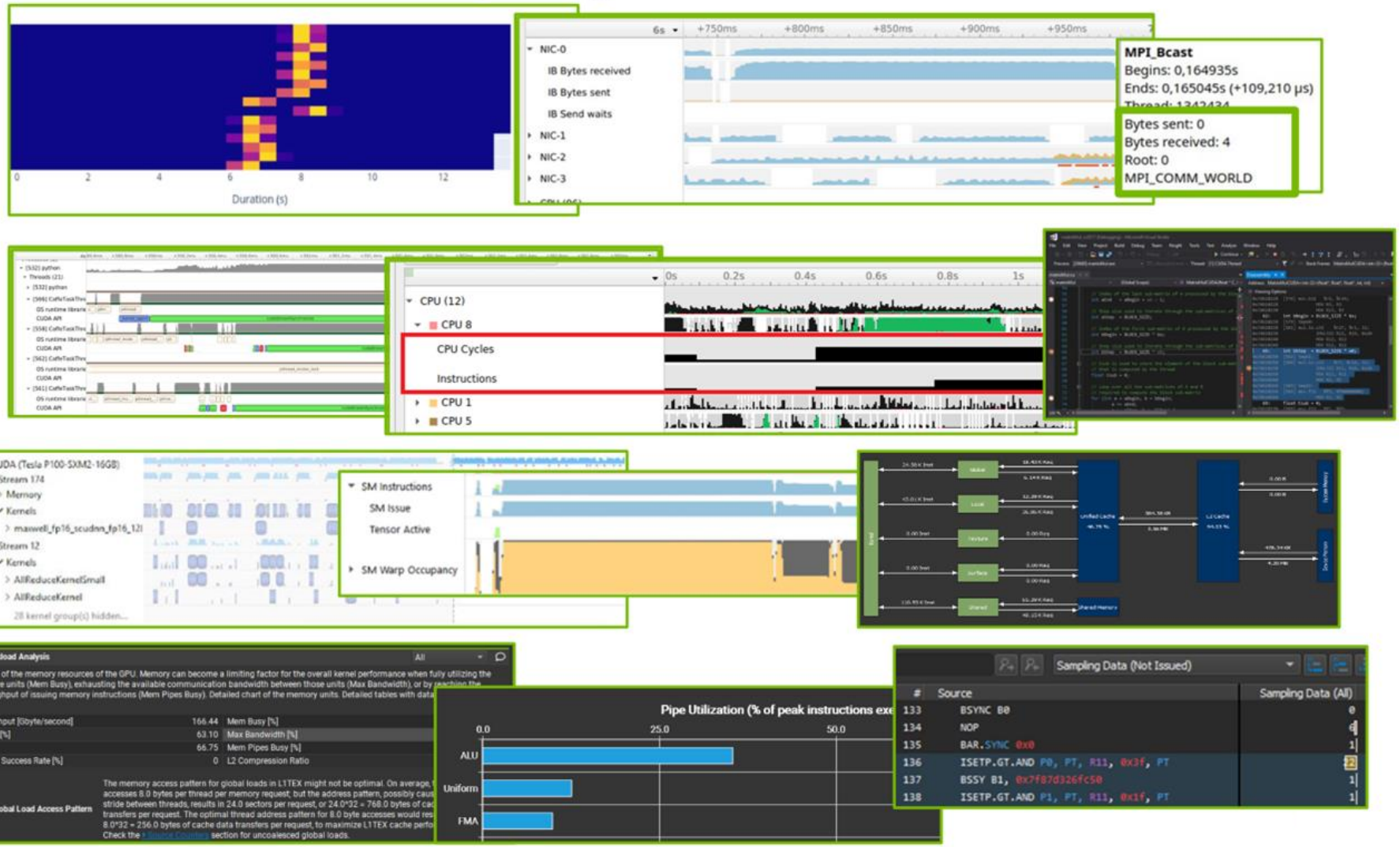
- Linux, Windows and Mac OS X (host-only)
- x86-64, Power9, ARM server, Tegra (Linux & QNX)

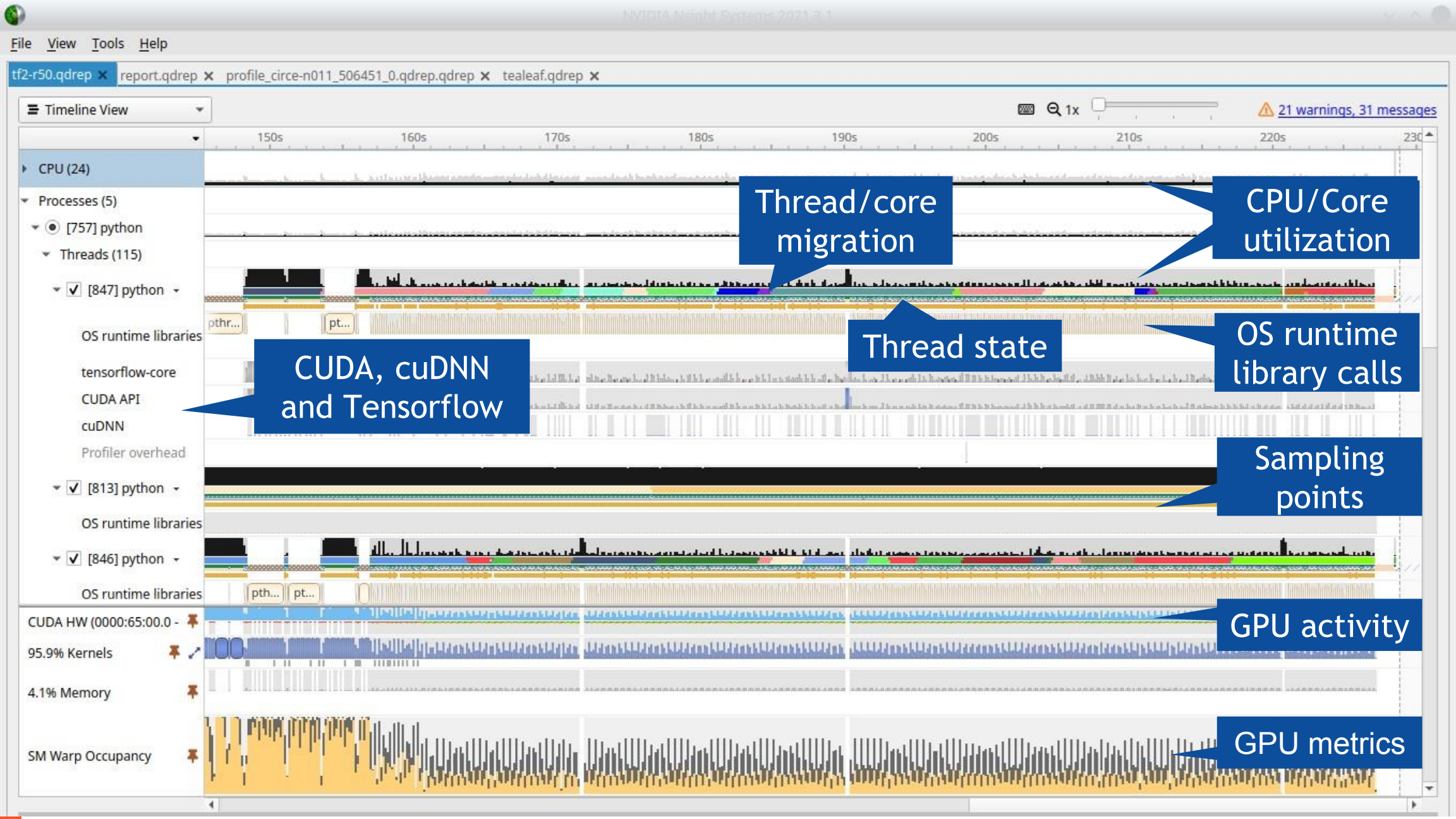
OpenACC More Science, Less Programming

OPEN HACKATHONS

From the Macro to the Micro

Putting it all together




File View Tools Help
tf2-r50.qdrep
report.qdrep
profile_circe-n011_506451_0.qdrep.qdrep
tealeaf.qdrep
Timeline View
1x
21 warnings, 31 messages
150s
160s
170s
180s
190s
200s
210s
220s
CPU (24)
Processes (5)
[757] python
Threads (115)
[847] python
OS runtime libraries
tensorflow-core
CUDA API
cuDNN
Profiler overhead
[813] python
OS runtime libraries
[846] python
OS runtime libraries
CUDA HW (0000:65:00.0 -
95.9% Kernels
4.1% Memory
SM Warp Occupancy
Thread/core migration
CPU/Core utilization
Thread state
OS runtime library calls
CUDA, cuDNN and Tensorflow
Sampling points
GPU activity
GPU metrics

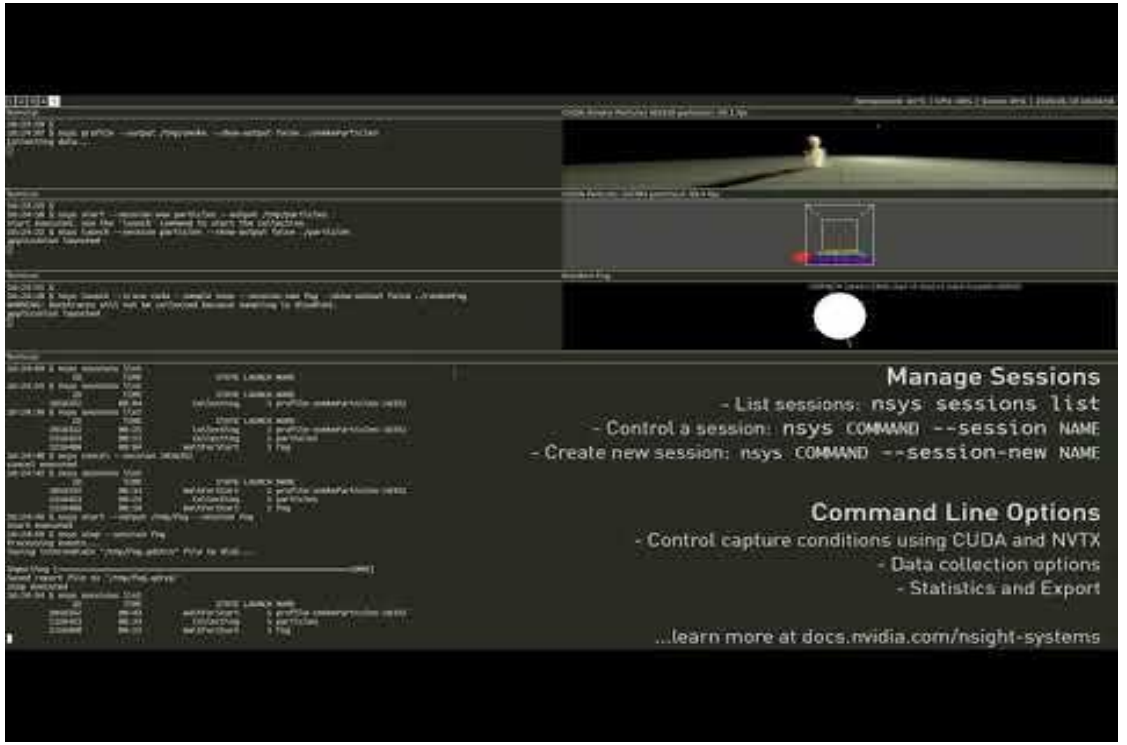
Command Line Interface (CLI) Statistics and Export to SQLite, JSON, etc.

OpenACC
More Science, Less Programming

OPEN HACKATHONS

Command Line Interface

- The Nsight Systems CLI provides several different commands
  - Basic profiling session
    `nsys profile ./app`
  - Interactive sessions (scriptable)
    `nsys start|launch|stop|cancel`
    `nsys session list`
    `nsys status|shutdown`
  - Statistics and export
    `nsys stats|export`
    (export to sqlite, hdf, text, json, info)



YouTube: Nsight Systems - Command Line Sessions Feature spotlight

CLI Profiling - Some Useful Switches

- API tracing
- `-t, --trace=cuda,nvtx,osrt,opengl`
- `(cublas,cusparse,cudnn,mpi,oshmem,ucx,openacc,openmp,vulkan,none)`
- Overwrite existing report
- `-f, --force-overwrite=[true|`**`false`**`]`
- Summary statistics (profile output on command line)
- `--stats=[true|`**`false`**`]`
- Report file name
- `-o, --output=report#`
  `(patterns for hostname, PID and environment variables)`

OpenACC More Science, Less Programming

OPEN HACKATHONS

CLI Profiling - Some Useful Switches

- Callstack sampling
  - `-s, --sample=[`**`cpu`**`|none]`
  - `--sampling-period=`*number of CPU Instructions Retired events*
  - `-b, --backtrace=[`**`lbr`**`|fp|dwarf|none]`
  - `--samples-per-backtrace={`**`1`**`..12}`

    (The number of CPU IP samples collected for every CPU IP sample backtrace collected.)

- Set the paranoid level: "`sudo sh -c 'echo 1 >/proc/sys/kernel/perf_event_paranoid`'
- CUDA memory usage
  - `--cuda-memory-usage=[true|`**`false`**`]`
- (Use `nsys profile --help` for a list of available options.)

CLI Profiling - MPI Programs

- **Single Node**
- `nsys profile [nsys_args] mpirun [mpirun_args] your_executable`
- ⇨ This creates one report file.

- **Multiple Nodes**
- `mpirun [mpirun_args] nsys profile [nsys_args] your_executable`
- Set output report name with `-o report_name_%q{OMPI_COMM_WORLD_RANK}`
- (for OpenMPI, `PMI_RANK` for MPICH and `SLURM_PROCID` for Slurm)

- ⇨ This creates one report file per MPI rank.

Profile only specific ranks. ⇨

```
#!/bin/bash
# OMPI_COMM_WORLD_LOCAL_RANK for node local rank
if [ $OMPI_COMM_WORLD_RANK -eq 0 ]; then
  nsys profile -t mpi "$@"
else
  "$@"
fi
```

CLI Profiling - Additional Output

```
WARNING: The command line includes a target application therefore the CPU context-switch scope has been set to process-tree.
Collecting data…
  … APP OUTPUT …
Processing events...
Saving temporary "/tmp/nsys-report-2b96-9038-d0bd-2600.qdstrm" file to disk...

Creating final output files...
Processing [==========================================================100%]
Saved report file to "/tmp/nsys-report-2b96-9038-d0bd-2600.nsys-rep"
Report file moved to "/final/destination/report.nsys-rep"
```

Temporary data is written to **/tmp/nvidia/nsight_systems** by default. Set **TMPDIR** to specify another location.

CLI Stats Output

- CUDA API, kernels and memory operations (by time and by size)
- OS runtime
- NVTX

NVTX Push-Pop Range Statistics:

Time(%)	Total Time (ns)	Instances	Average	Minimum	Maximum	Range
80,0	16.193.376.534	21.268	761.396,0	6.823	2.438.848	MPI:MPI_Waitall
18,0	3.680.937.672	8.072	456.013,0	1.726	1.932.681	MPI:MPI_Allreduce
0,0	88.587.708	4	22.146.927,0	21.714.183	22.579.671	MPI:MPI_Init
0,0	57.888.066	21.268	2.721,0	1.319	297.466	MPI:MPI_Isend
0,0	24.282.234	21.268	1.141,0	872	16.931	MPI:MPI_Irecv
0,0	8.793.048	4	2.198.262,0	2.181.292	2.215.232	MPI:MPI_Finalize
0,0	3.814.158	28	136.219,0	2.302	1.265.786	MPI:MPI_Barrier
0,0	266.630	32	8.332,0	832	104.980	MPI:MPI_Reduce

CUDA Kernel Statistics:

Time(%)	Total Time (ns)	Instances	Average	Minimum	Maximum	Name
37,0	7.344.350.347	8.568	857.183,0	851.546	876.922	device_tea_leaf_ppcg_solve_calc_sd_new(kernel_info_t, dou
36,0	7.229.217.310	8.568	843.746,0	839.323	899.482	device_tea_leaf_ppcg_solve_update_r(kernel_info_t, double
10,0	2.050.961.573	2.010	1.020.378,0	1.005.849	1.036.569	device_tea_leaf_cg_solve_calc_ur(kernel_info_t, double, d
9,0	1.879.919.365	2.010	935.283,0	913.754	1.214.808	device_tea_leaf_cg_solve_calc_w(kernel_info_t, double*, d
5,0	1.008.596.179	1.980	509.392,0	501.469	521.756	device_tea_leaf_cg_solve_calc_p(kernel_info_t, double, do
0,0	61.606.046	30	2.053.534,0	1.176.600	2.890.541	device_tea_leaf_calc_rrn(kernel_info_t, double const*, do
0,0	38.094.843	12.336	3.088,0	1.887	13.184	void reduction<double, (REDUCTION_TYPE)0>(int, double*)

NVTX
NVIDIA Tool Extension Library

NVTX

NVIDIA Tool Extension Library

NVTX provides means to correlate the profile data with the application code.

```
#include <nvtx3/nvToolsExt.h>
…
nvtxMark("Point in time");

nvtxRangePush("Name of your code region");
// your code goes here
nvtxRangePop();

nvtxRangeId_t rid = nvtxRangeStart("Name of another code region"
// stop might be on another thread
nvtxRangeEnd(rid)
```

Add **nvtx** to the tracing options of **nsys profile**.)

NVTX

Domains and Resource Naming

```
nvtxDomainHandle_t yourDomain = nvtxDomainCreateA("My Domain");

nvtxEventAttributes_t evtAttr = {0};
evtAttr.version = NVTX_VERSION;
evtAttr.size = NVTX_EVENT_ATTRIB_STRUCT_SIZE;
evtAttr.messageType = NVTX_MESSAGE_TYPE_ASCII;
eventAttrib.message.ascii = "My Range";

nvtxDomainRangePushEx(yourDomain, &evtAttr);
// your code
nvtxDomainRangePop(yourDomain);


// #include <sys/syscall.h>
nvtxNameOsThreadA(syscall(SYS_gettid), "My Thread");
```

NVTX

Registered Strings

```
nvtxDomainHandle_t yourDomain = nvtxDomainCreateA("My Domain");
nvtxStringHandle_t yourStrHdl =
     nvtxDomainRegisterStringA(yourDomain, "My String Handle");

nvtxEventAttributes_t evtAttr = {0};
evtAttr.version = NVTX_VERSION;
evtAttr.size = NVTX_EVENT_ATTRIB_STRUCT_SIZE;
evtAttr.messageType = NVTX_MESSAGE_TYPE_REGISTERED;
eventAttrib.message.registered = yourStrHdl;

nvtxDomainRangePushEx(yourDomain, &evtAttr);
// your code
nvtxDomainRangePop(yourDomain);
```

OpenACC More Science, Less Programming

OPEN HACKATHONS

https://github.com/NVIDIA/NVTX

NVTX

Fortran Bindings

The NVIDIA HPC SDK Fortran compiler provides NVTX bindings.

- **`libnvhpcwrapnvtx.[a|so]`** has to be linked
- Documentation can be found here: https://docs.nvidia.com/hpc-sdk/compilers/fortran-cuda-interfaces/index.html#cfnvtx-runtime

```
use nvtx
…
call nvtxStartRange("YourRange")
! some Fortran code
call nvtxEndRange
```

For other compilers, write your own Fortran NVTX bindings or use existing ones, e.g. https://raw.githubusercontent.com/maxcuda/NVTX_example/master/nvtx.f90

NVTX

Automatic code instrumentation

The NVIDIA HPC SDK and GNU compilers provide an automatic code instrumentation mechanism using respectively the options **-Minstrument** and **-finstrument-functions** with the __cyg_profile_* funcs

- **C/C++ example with gcc/g++ and option `-finstrument-functions` :**

https://developer.nvidia.com/blog/cuda-pro-tip-generate-custom-application-profile-timelines-nvtx/

- **Fortran documentation with nvfortran:**

https://docs.nvidia.com/hpc-sdk/compilers/fortran-cuda-interfaces/index.html#cfnvtx-auto

NVTX

"Python developers can either use decorators `@nvtx.annotate()` or a context manager with `nvtx.annotate(..)`"

```
import nvtx

@nvtx.annotate("f()", color="purple")
def f():
    for i in range(5):
        with nvtx.annotate("loop", color="red"):
            # Python code goes here
```

Get NVTX Python module:
`python -m pip install nvtx`

Also see this blog or the docs: https://nvtx.readthedocs.io/en/latest/index.html

PyTorch CUDA provides NVTX bindings

```
from torch.cuda import nvtx

nvtx.range_push("YourCode")
# your Python code
nvtx.range_pop()
```

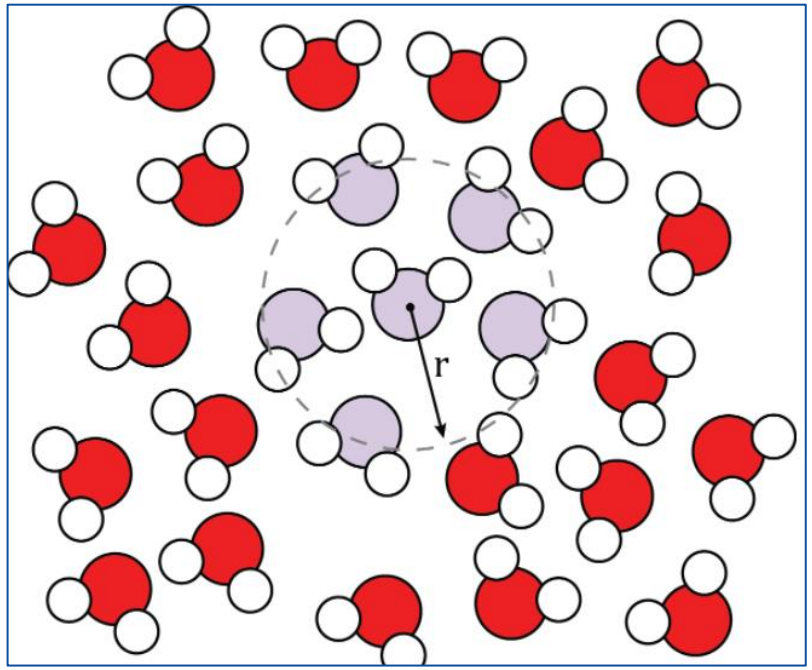
Application

APPLICATION

Molecular Simulation

RDF

The radial distribution function (RDF) denoted in equations by g(r) defines the probability of finding a particle at a distance r from another tagged particle.



RDF

Pseudo Code - C

```
for (int frame=0;frame<nconf;frame++){

            for(int id1=0;id1<numatm;id1++)
            {
                        for(int id2=0;id2<numatm;id2++)
                        {
                                    dx=d_x[]-d_x[];
                                    dy=d_y[]-d_y[];
                                    dz=d_z[]-d_z[];
                                    r=sqrtf(dx*dx+dy*dy+dz*dz);

                                    if (r<cut) {
                                                ig2=(int)(r/del);
                                                d_g2[ig2] = d_g2[ig2] +1 ;
                                    }
                        }
            }
}
```

- Across Frames
- Find Distance
- Reduction

RDF

Pseudo Code - Fortran

```
do iconf=1,nframes
  if (mod(iconf,1).eq.0) print*,iconf

  do i=1,natoms
    do j=1,natoms
                    dx=x(iconf,i)-x(iconf,j)
                    dy=y(iconf,i)-y(iconf,j)
                    dz=z(iconf,i)-z(iconf,j)

                    r=dsqrt(dx**2+dy**2+dz**2)
                    if(r<cut)then
                              g(ind)=g(ind)+1.0d0
                    endif
          enddo
  enddo
enddo
```

- Across Frames
- Find Distance
- Reduction