NMSU Update

Dinupa

August 9, 2022

Qt App fro Hodoscope Monitoring

- `Qt5` framework(`python/c++`) was used.
- `Qt` frmaework is;
  - Cross-platform software for creating graphical user interfaces (KDE Plasma DE).
  - Easy-to-read code
  - Rich choice of modules.
  - Multiple libraries and high level of control.
  - API for easier development.
- Drawback: can be slow with `ssh`. Better to use textual.
- github

- A common feature among all array-oriented languages (except `Fortran 90`) is that they are also interactive languages.
- Typically, you perform one operation on a whole dataset, see what that does to the distribution, then apply another (easy to debug).
- `ROOT` :
  - `RDataFrame`
  - `RVec`
- `python`
  - `numpy`
  - `pandas`
  - `Awkward Array`
- github
- New kid in the block: `julia` (`https://github.com/JuliaLang/julia`)

- Vectorization;

- Vectorization is effectively loop unrolling
  - In effect, the compiler unrolls by 4 iterations, if 4 elements fit into a vector register

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

```
for (i=0; i<N; i+=4) {
    c[i+0]=a[i+0]+b[i+0];
    c[i+1]=a[i+1]+b[i+1];
    c[i+2]=a[i+2]+b[i+2];
    c[i+3]=a[i+3]+b[i+3];
}
```

```
Load a(i..i+3)
Load b(i..i+3)
Do 4-wide a+b->c
Store c(i..i+3)
```

- Parallel processing: `OpenMP` for multi-threading.
- github

Results*

- Original Serial pi program with 100000000 steps ran in 1.83 seconds.

```c
#include <omp.h>
static long num_steps = 100000;        double step;
#define NUM_THREADS 2
void main ()
{    int i, nthreads;  double pi, sum[NUM_THREADS];
     step = 1.0/(double) num_steps;
     omp_set_num_threads(NUM_THREADS);
    #pragma omp parallel
    {
           int i, id,nthrds;
           double x;
           id = omp_get_thread_num();
           nthrds = omp_get_num_threads();
           if (id == 0)   nthreads = nthrds;
           for (i=id, sum[id]=0.0;i< num_steps; i=i+nthrds) {
                   x = (i+0.5)*step;
                   sum[id] += 4.0/(1.0+x*x);
           }
    }
   for(i=0, pi=0.0;i<nthreads;i++)pi += sum[i] * step;
}
```

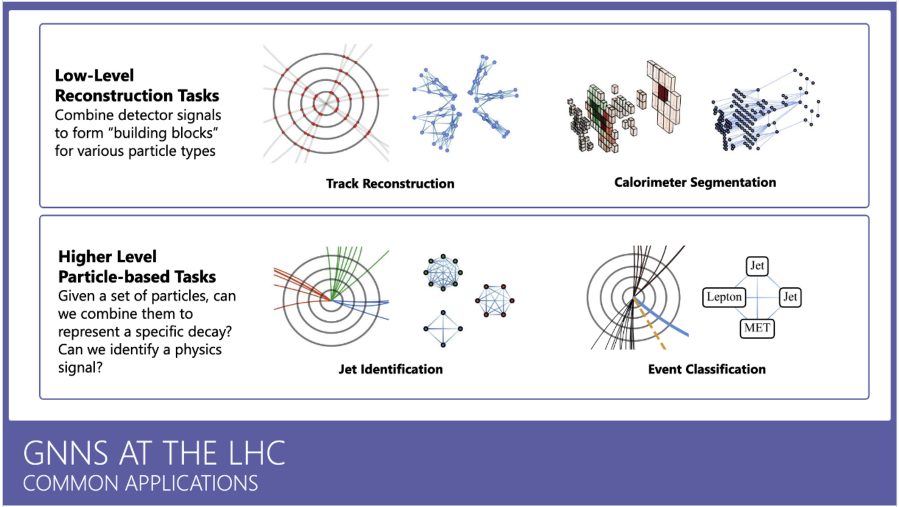
threads	1st SPMD*
1	1.86
2	1.03
3	1.08
4	0.97

Intel compiler (icpc) with default optimization level (O2) on Apple OS X 10.7.3 with a dual core (four HW thread) Intel® Core™ i5 processor at 1.7 Ghz and 4 Gbyte DDR3 memory at 1.333 Ghz.

*SPMD: Single Program Multiple Data

37

- github


Low-Level
Reconstruction Tasks
Combine detector signals
to form "building blocks"
for various particle types
Track Reconstruction
Calorimeter Segmentation
Higher Level
Particle-based Tasks
Given a set of particles, can
we combine them to
represent a specific decay?
Can we identify a physics
signal?
Jet Identification
Jet
Lepton
Jet
MET
Event Classification
GNNS AT THE LHC
COMMON APPLICATIONS

GNNS AT THE LHC
WHY GNNS?

Common Justifications (task-dependent)
- Many LHC datasets have inherent relational structure and/or no inherent ordering
- Grids, sequences, etc. cannot naturally represent irregular detector geometries
  - A small fraction of sensors are activated in any given event ➔ data is sparse
  - Many different data sizes (particle counts, sensor readings, etc.)
- LHC data is heterogeneous
  - Data recorded from multiple subdetectors
  - Different types of particles
- Excellent performance
  - Relational inductive bias
  - Message passing leverages low-level detector info in addition to global (or otherwise human-devised) info
  - Generally smaller architectures (qualitatively speaking)

For SpinQuest

- Vectorize/Parallelize the `KTracker`.
- Use `GNN` for track building.