Improving GPU performance for GooFit

Software engineering studies

Brad Hittle

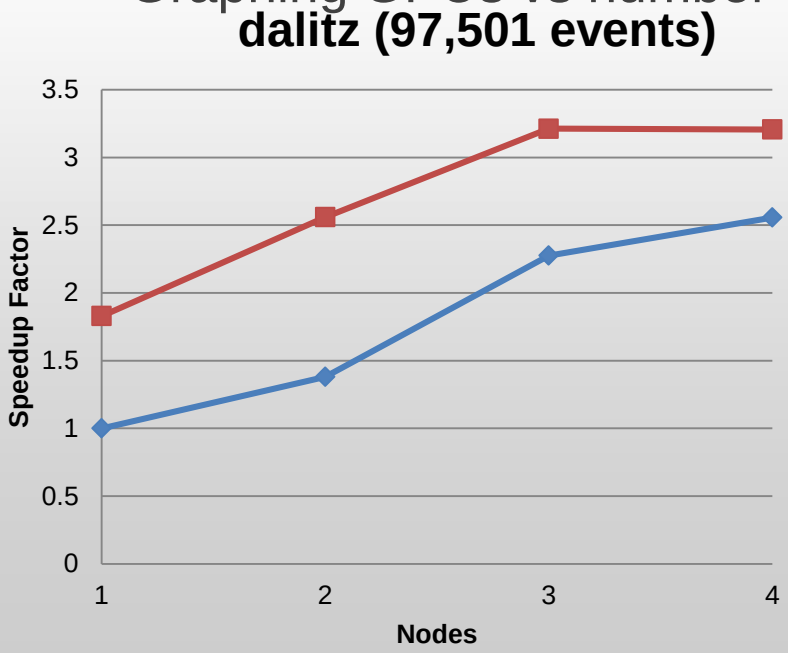
February 12th, 2018

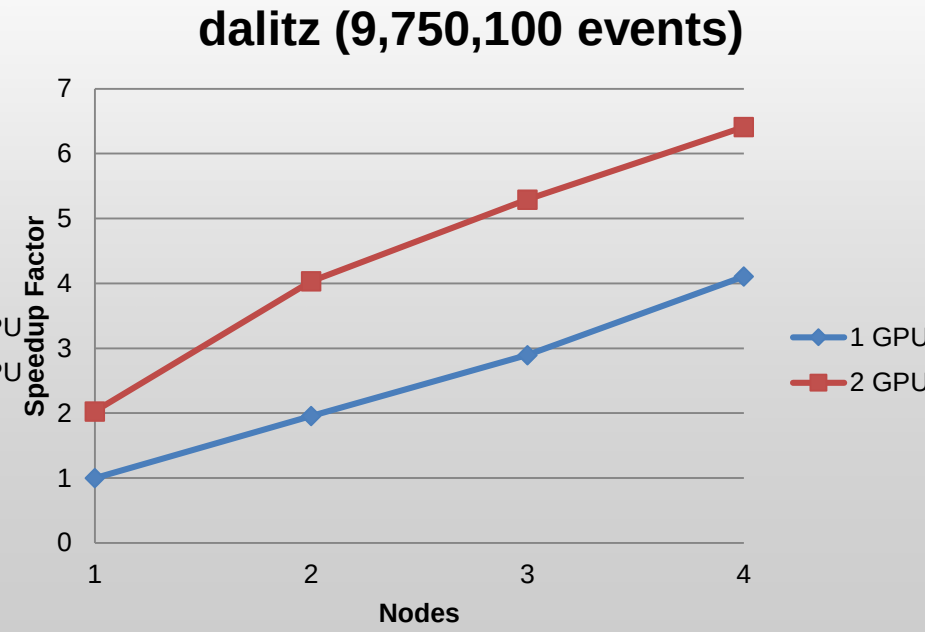
GooFit library

- Probability Density Function (PDF) fitting library
- GPUs for better performance
- Modifications to library for performance
  - Multiple GPU Improvements
  - Memory improvements
  - New PDFs in C++

Optimizations (Multi-GPU performance)

- Add Multiple GPUs using MPI [**GooFit 2.0+**]
- Each GPU gets portion of data
- Graphing GPUs vs number of workstations (**nodes**)

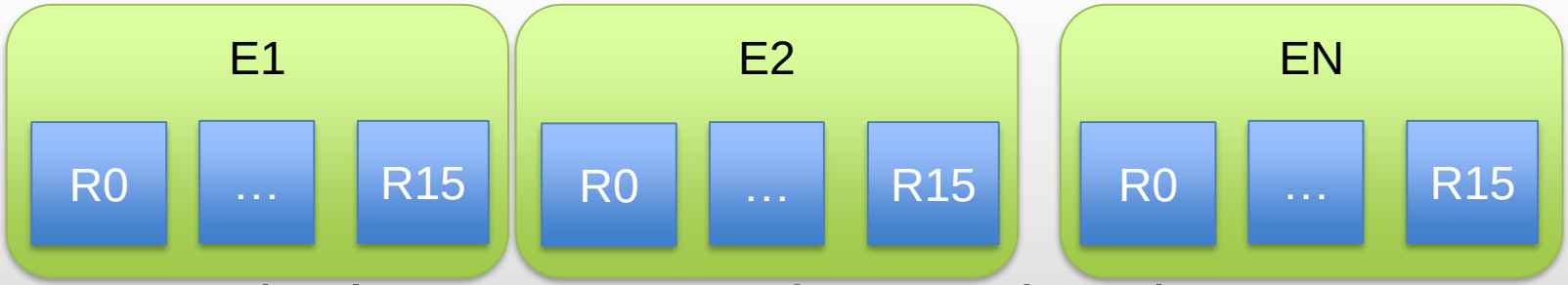




Intel Xeon x5650 (Westmere), 2x NVIDIA Tesla M2070 GPUs

Memory Cache Layout

- New memory layout for cached values, **[GooFit 2.0+]**
- Old method used a single array (**AoS**)



- New method uses 16 sets of arrays (**SoA**)

Over 25% improvement for *dalitz 9,750,100 events* over GooFit 1.0!

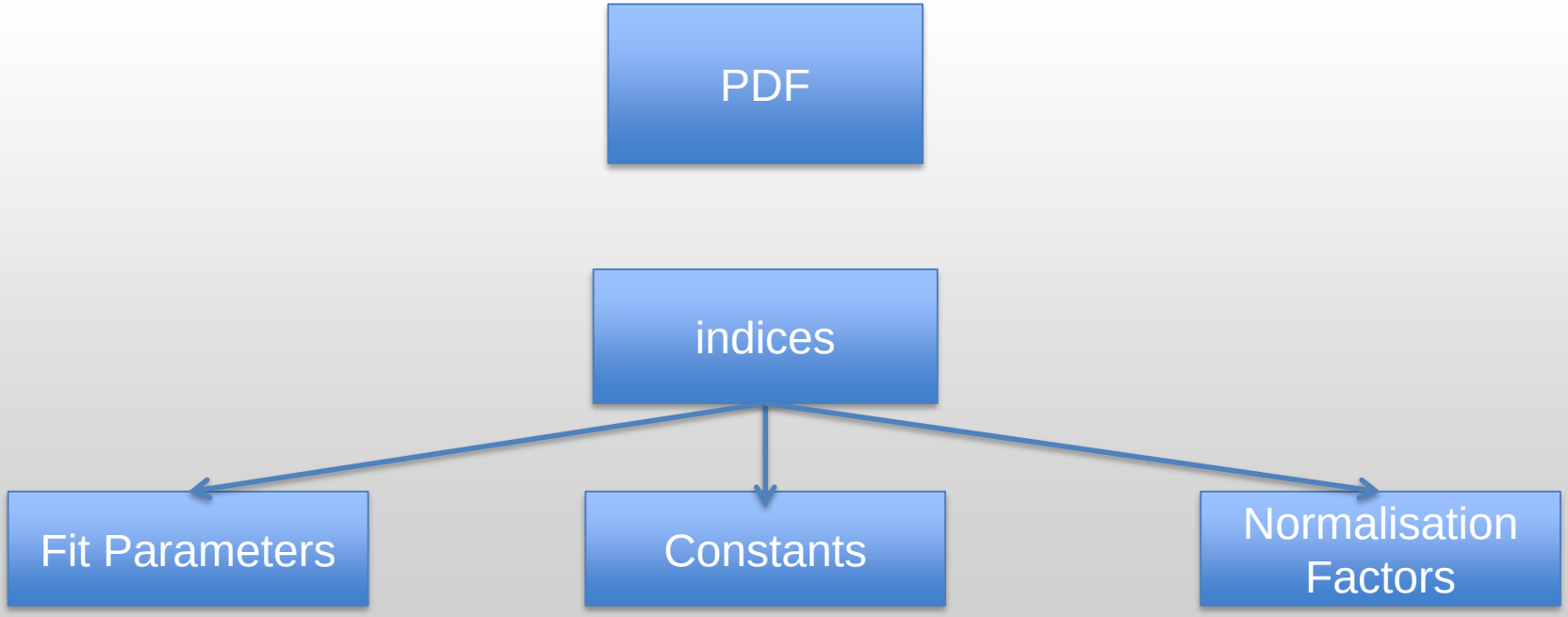
Memory Through Hardware Cache

- All GPU's have read-only cache for gaming
- Uses read-only cache instruction (ROC) [**GooFit 2.0+**]
  - Instruction requires new hardware (Kepler+)
- ROC used for parameters, constants, observable indices, normalisation factors. Unable to use on Observables.

dalitz	97,501 events	9,750,100 events
No ROC	2.27	161.7
ROC	2.09	137.1

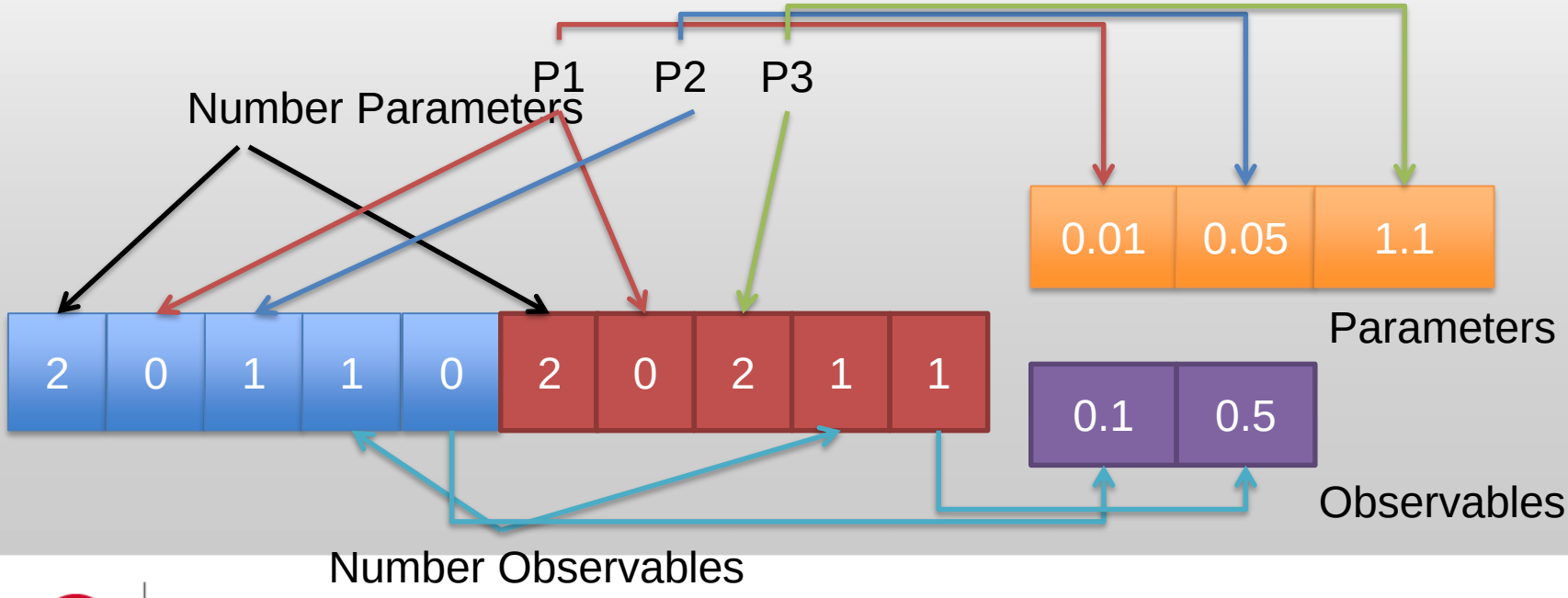
GooFit Indexing [1]

- Indexing scheme up to **GooFit 2.1**



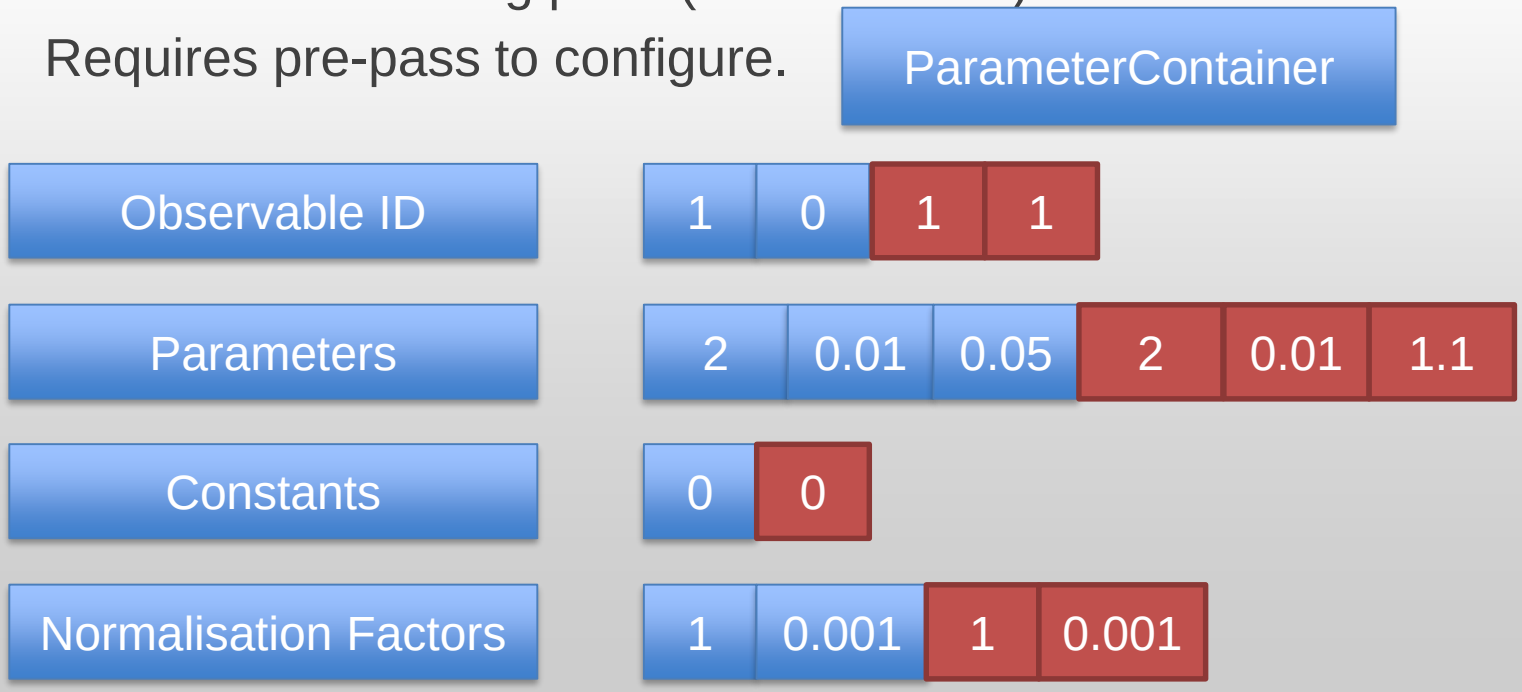
GooFit Indexing [2]

- 2 memory lookups
- Multiple PDFs cause random memory accesses
- Internal representation of two Gaussian PDFs



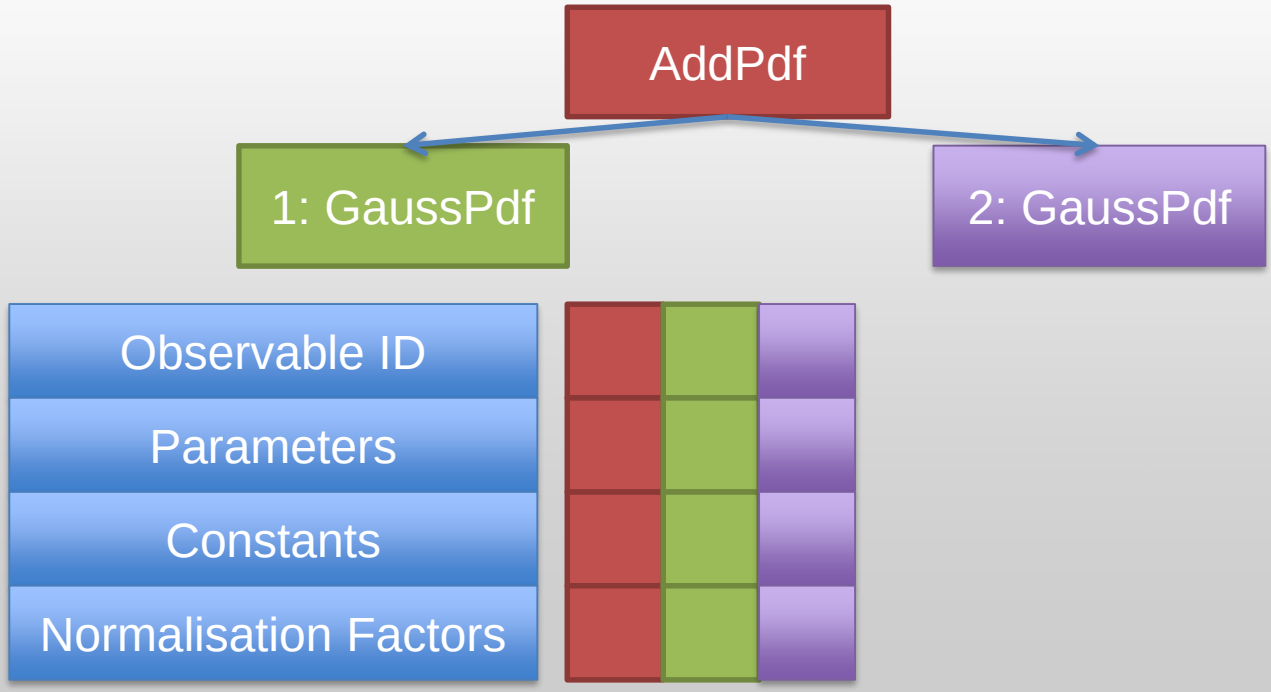
GooFit Indexing [3]

- **GooFit 2.2** scheme
- 4 separate arrays
- All constants floating point (cast constant)
- Requires pre-pass to configure.



GooFit Indexing [4]

- Arrays are populated in order.



GooFit Indexing [5]

- Reduce random memory accesses
- Reduce amount of memory accesses
- Simplified PDF model
  - Easier development and debugging of PDFs.

GooFit Indexing [5]

- Performance between OMP/GPU versions
- Suspect cache related, requires further profiling OMP version

OMP=28	v2.0 (s)	v2.2 (s)
pipipi0DPFit toy 0 10	240.92	276.26
pipipi0DPFit toy 0 50	1092.2	1247.3
dalitz 9,750,100	151.9	230.54

-65%

GPU=Tesla P100	v2.0 (s)	v2.2 (s)
pipipi0DPFit toy 0 10	57.099	48.248
pipipi0DPFit toy 0 50	265.34	210.19
dalitz 9,750,100	112.86	85.132

+32%

Intel Xeon E5-2680 v4 (Broadwell), 1x NVIDIA Tesla P100

New PDF creation C++ [1]

- New PDF creation [**GooFit 2.2**], easier development and debugging
- Constructor
- recursiveSetIndices
- ‘Device’ function

New PDF creation C++ [2]

- 3 internal structures per PDF
  - Parameters
  - Observables
  - Constants
- registerParameter, registerConstant, registerObservable
- Once complete, `initialize();` from constructor

New PDF creation C++ [3]

```
GaussianPdf::GaussianPdf(std::string n,
        Observable _x,
        Variable mean,
        Variable sigma) : GooPdf(n, _x)
{
  int idx1 = registerParameter(mean);
  int idx2 = registerParameter(sigma);

  initialize();
}
```

New PDF creation C++ [4]

- Copy local information to global
- Copy current PDF, recursively call components
- Copy global structures to GPU

```
__host__ void GaussianPdf::recursiveSetIndices() {
 GET_FUNCTION_ADDR(ptr_to_Gaussian);
 host_function_table[num_device_functions]=host_fcn_ptr;
 functionIdx = num_device_functions++;
 populateArrays();
}
```

New PDF creation C++ [5]

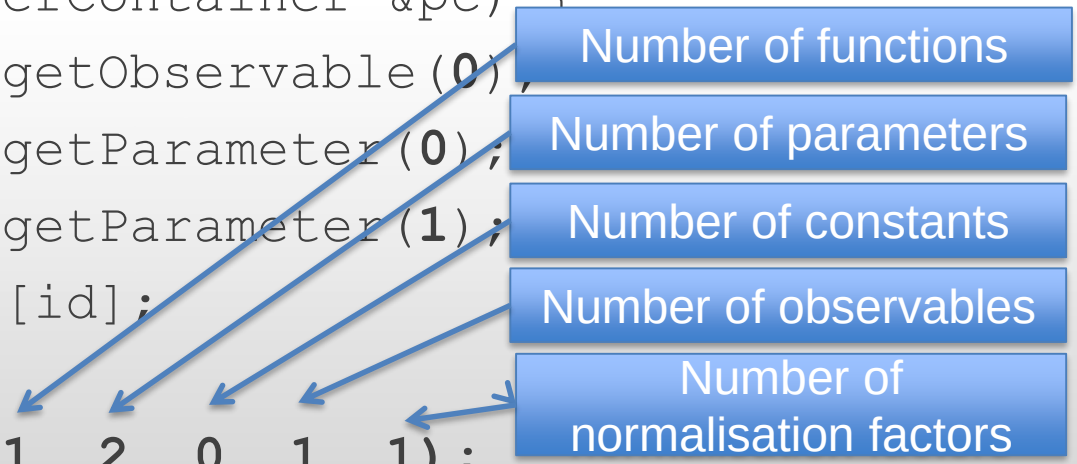
- ParameterContainer
  - Access values per PDF
  - Hide implementation details
- Two increment methods
  - **incrementIndex(# functions, # parameters, # constants, # observables, # normalisations);**
  - incrementIndex ();
- Constructor index == device index

New PDF creation C++ [6]

```
fptype device_Gaussian(fptype *evt,
                       ParameterContainer &pc) {
    int id       = pc.getObservable(0);
    fptype mean  = pc.getParameter(0);
    fptype sigma = pc.getParameter(1);
    fptype x     = evt[id];

    pc.incrementIndex(1, 2, 0, 1, 1);

    return exp(-0.5 * (x - mean) * (x - mean) /
               (sigma * sigma));
}
```



Future improvements

- [**GooFit 2.3+**]
- SoA-style observable data access
- Reducing register count per GPU kernel

Conclusions

- Multi-GPU support [**GooFit 2.0+**]
  - Large events, 1 GPU 4 workstations [**4x speedup**]
- Memory optimizations
  - Changed **AoS** to **SoA** memory layout [**25% improvement, GooFit 2.0+**]
  - Utilized read only cache [**20% improvement, GooFit 2.0+**]
  - New indexing [**32% improvement, GooFit 2.2+**]
- Simplified PDF creation [**GooFit 2.2+**]
  - GPU improvement, readability & debugging
  - Degraded performance for OMP

Memory layout matters

Questions, Feedback?

Brad Hittle

bhittle@osc.edu

Acknowledgement: GooFit's development is supported by the National Science Foundation under grant number 1414736 and was developed under grant number 1005530. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the developers and do not necessarily reflect the views of the National Science Foundation.

Optimizations (Improved if/else condition)

- Current code:

```
EXEC_TARGET bool inDalitz (fptype m12, fptype m13, fptype bigM, fptype dm1,
fptype dm2, fptype dm3) {
  if (m12 < POW(dm1 + dm2, 2)) return false;
  if (m12 > POW(bigM - dm3, 2)) return false;
  // Calculate energies of 1 and 3 particles in m12 rest frame.
  fptype e1star = 0.5 * (m12 - dm2*dm2 + dm1*dm1) / SQRT(m12);
  fptype e3star = 0.5 * (bigM*bigM - m12 - dm3*dm3) / SQRT(m12);
  // Bounds for m13 at this value of m12.
  fptype minimum = POW(e1star + e3star, 2) - POW(SQRT(e1star*e1star -
dm1*dm1) + SQRT(e3star*e3star - dm3*dm3), 2);
  if (m13 < minimum) return false;
  fptype maximum = POW(e1star + e3star, 2) - POW(SQRT(e1star*e1star -
dm1*dm1) - SQRT(e3star*e3star - dm3*dm3), 2);
  if (m13 > maximum) return false;

  return true;
}
```

Optimizations (Improved if/else condition) [2]

- [**GooFit 2.0+**]
- GPU's have no branch-prediction.
- Referred to as *Divergent threads* when some take the 'if' path and others take the 'else' path
- Each warp executes the 'if' portion of the if/else with some threads stalling, then the 'else' portion executes similarly if needed.
- Wasted cycles waiting for if/else portions to complete per warp if there are divergent threads.
- Depending on situation, computing both sides is more efficient than handling if/else
- Improved by 21%

Optimizations (Improved if/else condition) [3]

```
EXEC_TARGET bool inDalitz (fptype m12, fptype m13, fptype bigM, fptype dm1, fptype dm2, fptype dm3) {
  bool dm1_dm2 = (m12 < POW(dm1 + dm2, 2)) ? true : false;
  bool bigM_dm3 = (m12 > POW(bigM - dm3, 2)) ? true : false;
  // Calculate energies of 1 and 3 particles in m12 rest frame.
  fptype e1star = 0.5 * (m12 - dm2*dm2 + dm1*dm1) / SQRT(m12);
  fptype e3star = 0.5 * (bigM*bigM - m12 - dm3*dm3) / SQRT(m12);
  // Bounds for m13 at this value of m12.
  fptype minimum = POW(e1star + e3star, 2) - POW(SQRT(e1star*e1star - dm1*dm1) + SQRT(e3star*e3star - dm3*dm3), 2);
  bool m13_minimum = (m13 < minimum) ? true : false;
  fptype maximum = POW(e1star + e3star, 2) - POW(SQRT(e1star*e1star - dm1*dm1) - SQRT(e3star*e3star - dm3*dm3), 2);
  bool m13_maximum = (m13 > maximum) ? true : false;

  return dm1_dm2 && bigM_dm3 && m13_minimum && m13_maximum;
```

GPU Grid size [1]

- [**GooFit 2.0+**]
- (**Group size**, **Group size**)
  - Determines GPU threads/block distribution
  - Thrust has fixed distribution
- Override for runtime performance; unique per PDF

dalitz	97, 501 events	9,750,100 events
ruby (128, 7)	2.09892	137.133
ruby (544, 7)	1.66221	110.333