Optimizing Allen Back-End with AMD HIP

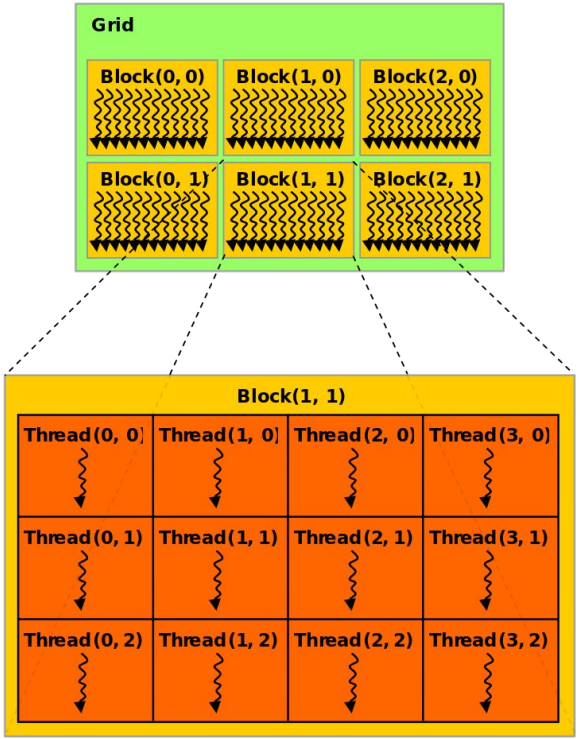
Emile Savard, Arizona State University
Mentors: Daniel Campora and Tom Beottcher

Overview

- The Allen Initiative (Upgrading LHCb by using GPUs)
- Parallelizing more of the code
- AMD GPUS only use HIP
- Trying to increase AMD’s throughput (rate of events per second) to a similar level as NVIDIA's GPUS

GPUS

- Kernels = Functions that runs on the GPU
- Threads = Workers allocated to each kernel
- Block Dimensions = Logical group of threads that share memory

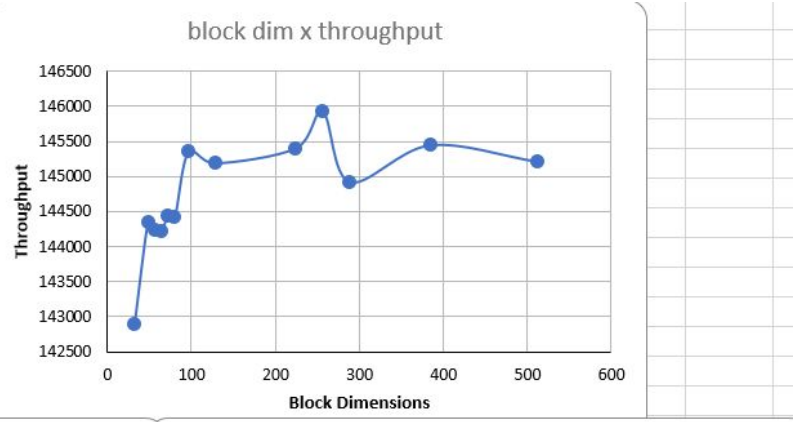


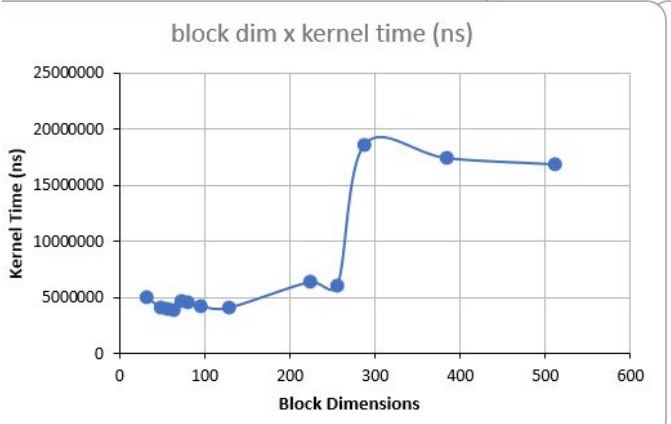
Profiling using rocprof

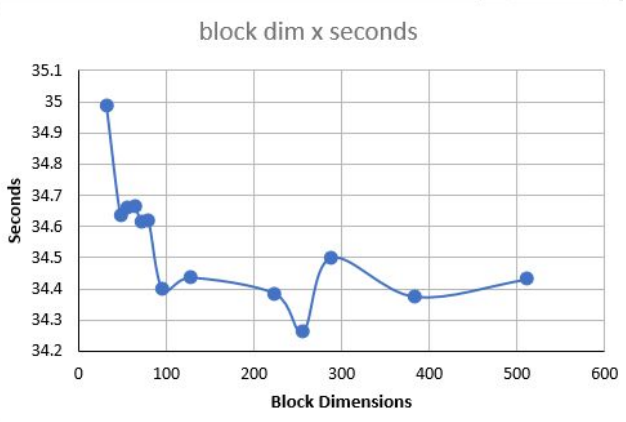
Name	Calls	TotalDurationNs	AverageNs	Percentage
velo_search_by_triplet::velo_search_by_triplet(velo_search_by_triplet::Parameters, VeloGeometry const*)	1	6755150	6755150	12.03661723
If_triplet_seeding::If_triplet_seeding(If_triplet_seeding::Parameters, LookingForward::Constants const*)	1	4827485	4827485	8.601820705
velo_consolidate_tracks::velo_consolidate_tracks(velo_consolidate_tracks::Parameters)	1	4434047	4434047	7.90077593
is_muon::is_muon(is_muon::Parameters, Muon::Constants::FieldOfInterest const*, float const*)	1	3555334	3555334	6.335047258
velo_estimate_input_size::velo_estimate_input_size(velo_estimate_input_size::Parameters)	1	3145577	3145577	5.604924587
ut_search_windows::ut_search_windows(ut_search_windows::Parameters, UTMagnetTool*, float const*, unsigned int const*, float const*)	1	2273263	2273263	4.050597929
velo_calculate_phi_and_sort::velo_calculate_phi_and_sort(velo_calculate_phi_and_sort::Parameters)	1	2235184	2235184	3.982747126
muon_populate_hits::muon_populate_hits(muon_populate_hits::Parameters)	1	2171184	2171184	3.86870917
muon_add_coords_crossing_maps::muon_add_coords_crossing_maps(muon_add_coords_crossing_maps::Parameters)	1	2060785	2060785	3.671995476
velo_masked_clustering::velo_masked_clustering(velo_masked_clustering::Parameters, VeloGeometry const*, unsigned char const*, float const*, float const*)	1	1847347	1847347	3.291682455
scifi_raw_bank_decoder_v4::scifi_direct_decoder_v4(scifi_raw_bank_decoder_v4::Parameters, char const*)	1	1685908	1685908	3.004023491
If_create_tracks::If_extend_tracks(If_create_tracks::Parameters, LookingForward::Constants const*)	1	1575028	1575028	2.806452731
compass_ut::compass_ut(compass_ut::Parameters, UTMagnetTool*, float const*, float const*, unsigned int const*)	1	1388950	1388950	2.474890936
If_create_tracks::If_triplet_keep_best(If_create_tracks::Parameters, LookingForward::Constants const*)	1	1311351	1311351	2.336621695
pv_beamline_peak::pv_beamline_peak(pv_beamline_peak::Parameters, unsigned int)	1	1192311	1192311	2.124511096
If_search_initial_windows::If_search_initial_windows(If_search_initial_windows::Parameters, char const*, LookingForward::Constants const*, float const*)	1	1183191	1183191	2.108260687
pv_beamline_multi_fitter::pv_beamline_multi_fitter(pv_beamline_multi_fitter::Parameters, float const*)	1	1029752	1029752	1.834856468
pv_beamline_extrapolate::pv_beamline_extrapolate(pv_beamline_extrapolate::Parameters)	1	1021113	1021113	1.819463125
scifi_pre_decode_v4::scifi_pre_decode_v4(scifi_pre_decode_v4::Parameters, char const*)	1	961592	961592	1.713406044
pv_beamline_histo::pv_beamline_histo(pv_beamline_histo::Parameters, float*)	1	943833	943833	1.681762293
scifi_consolidate_tracks::scifi_consolidate_tracks(scifi_consolidate_tracks::Parameters, LookingForward::Constants const*, float const*)	1	916154	916154	1.632442658
global_decision::global_decision(global_decision::Parameters)	1	825114	825114	1.470223665
void ut_pre_decode::ut_pre_decode<3, false>(ut_pre_decode::Parameters, char const*, char const*, unsigned int const*, unsigned int const*, unsigned int const*)	1	822233	822233	1.465090175
velo_kalman_filter::velo_kalman_filter(velo_kalman_filter::Parameters, float*)	1	749754	749754	1.335943971
ut_find_permutation::ut_find_permutation(ut_find_permutation::Parameters, unsigned int const*)	1	728955	728955	1.298883417
two_track_evaluator::two_track_evaluator(two_track_evaluator::Parameters, float const*, int const*, float const*, int const*, int const*, int const*, int)	1	623196	623196	1.110437475

Resource Management (if_create_tracks extend_tracks)

block dim	throughput	seconds	kernel time
32	142906.793	34.9879	5034057.67
48	144357.549	34.63629	4129133.33
56	144248.326	34.6625	4005830.33
64	144228.943	34.66711	3849033.67
72	144449.432	34.61424	4694100.67
80	144432.698	34.61835	4590398.33
96	145354.652	34.39873	4195394
128	145191.962	34.43717	4041174.67
224	145407.71	34.38622	6362756
256	145925.541	34.26442	6035305
288	144925.596	34.50064	18635641.7
384	145451.013	34.3759	17398425.3
512	145213.953	34.43214	16839025.7



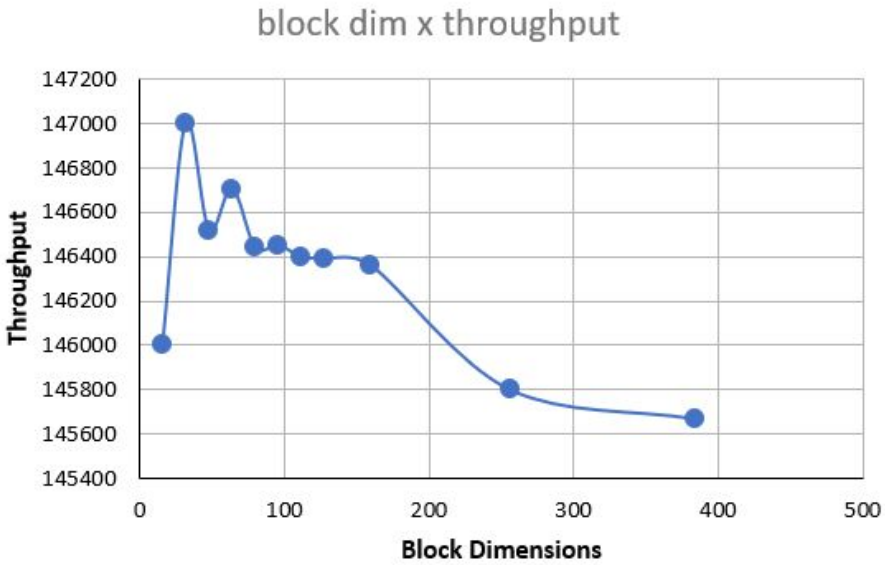




Muon_add_coords_crossing_maps

```
for (unsigned i = threadIdx.x;
     i < Muon::Constants::n_stations * Muon::Constants::n_regions * Muon::Constants::n_quarters;
     i += blockDim.x) {
```

block dim	throughput	seconds	kernel time
16	146005.3082	34.245535	4287123.333
32	147001.7532	34.013343	4133435
48	146519.6627	34.125144	4085992
64	146705.936	34.081831	4066652.333
80	146439.4977	34.143901	4322932.333
96	146450.6666	34.141327	4384733.333
112	146395.9016	34.153987	4360015.667
128	146389.5447	34.155445	4341330.667
160	146358.2723	34.162799	4605462.333
256	145798.4542	34.294069	5460710
384	145666.3661	34.325033	9592492.667



1st Possible Optimization Spot

```
for (unsigned i = threadIdx.x;
     i < Muon::Constants::n_stations * Muon::Constants::n_regions * Muon::Constants::n_quarters;
     i += blockDim.x) {

  // Note: The location of the indices depends on n_layouts.
  const auto start_index = storage_station_region_quarter_offsets[2 * i] - event_offset;
  const auto mid_index = storage_station_region_quarter_offsets[2 * i + 1] - event_offset;
  const auto end_index = storage_station_region_quarter_offsets[2 * i + 2] - event_offset;

  if (start_index != end_index) {
    const auto tile = Muon::MuonTileID(storage_tile_id[start_index]);
    const auto station = tile.station();
    const auto region = tile.region();
    unsigned number_of_hits_in_station = 0;
```

1st Attempt

```
__shared__ int sharedArray[Muon::Constants::n_stations * Muon::Constants::n_regions * Muon::Constants::n_quarters];
__shared__ uint sharedCounter;

if (threadIdx.x == 0)
{
  sharedCounter = 0;
}

__syncthreads();


for (unsigned i = threadIdx.x;
     i < Muon::Constants::n_stations * Muon::Constants::n_regions * Muon::Constants::n_quarters;
     i += blockDim.x) {

  // Note: The location of the indices depends on n_layouts.
  const auto start_index = storage_station_region_quarter_offsets[2 * i] - event_offset;
  const auto end_index = storage_station_region_quarter_offsets[2 * i + 2] - event_offset;

  if (start_index != end_index)
  {
    sharedArray[sharedCounter] = i;
    atomicAdd(&sharedCounter, 1);
  }
}

__syncthreads();
```

2nd Possible Optimization Spot

```
if (start_index != end_index) {
  const auto tile = Muon::MuonTileID(storage_tile_id[start_index]);
  const auto station = tile.station();
  const auto region = tile.region();
  unsigned number_of_hits_in_station = 0;

  const auto x1 = getLayoutX(
    parameters.dev_muon_raw_to_hits.get()->muonTables, Muon::MuonTables::stripXTableNumber, station, region);
  const auto y1 = getLayoutY(
    parameters.dev_muon_raw_to_hits.get()->muonTables, Muon::MuonTables::stripXTableNumber, station, region);
  const auto x2 = getLayoutX(
    parameters.dev_muon_raw_to_hits.get()->muonTables, Muon::MuonTables::stripYTableNumber, station, region);
  const auto y2 = getLayoutY(
    parameters.dev_muon_raw_to_hits.get()->muonTables, Muon::MuonTables::stripYTableNumber, station, region);
```

2nd Possible Optimization Spot (cont)

```
__device__ inline unsigned int
getLayoutX(MuonTables* muonTables, size_t tableNumber, unsigned int station, unsigned int region)
{
  return static_cast<unsigned int>(muonTables->gridX[tableNumber][station * Constants::n_regions + region]);
}

__device__ inline unsigned int
getLayoutY(MuonTables* muonTables, size_t tableNumber, unsigned int station, unsigned int region)
{
  return static_cast<unsigned int>(muonTables->gridY[tableNumber][station * Constants::n_regions + region]);
}
```

2nd Attempt

```
// initializing all shared arrays
int stripXTableNumber = Muon::MuonTables::stripXTableNumber;
int stripYTableNumber = Muon::MuonTables::stripYTableNumber;

__shared__ int gridXstripXTable[Muon::Constants::n_stations * Muon::Constants::n_regions];
__shared__ int gridXstripYTable[Muon::Constants::n_stations * Muon::Constants::n_regions];
__shared__ int gridYstripXTable[Muon::Constants::n_stations * Muon::Constants::n_regions];
__shared__ int gridYstripYTable[Muon::Constants::n_stations * Muon::Constants::n_regions];


for (int i = threadIdx.x; i < Muon::Constants::n_stations * Muon::Constants::n_regions; i += blockDim.x) {
  gridXstripXTable[i] = parameters.dev_muon_raw_to_hits.get()->muonTables->gridX[stripXTableNumber][i];
  __syncthreads();
  gridXstripYTable[i] = parameters.dev_muon_raw_to_hits.get()->muonTables->gridX[stripYTableNumber][i];
  __syncthreads();
  gridYstripXTable[i] = parameters.dev_muon_raw_to_hits.get()->muonTables->gridY[stripXTableNumber][i];
  __syncthreads();
  gridYstripYTable[i] = parameters.dev_muon_raw_to_hits.get()->muonTables->gridY[stripYTableNumber][i];
  __syncthreads();
}

__syncthreads();
```

2nd attempt (Cont)

```
const auto value = station * Muon::Constants::n_regions + region;

const unsigned int x1 = gridXstripXTable[value];
const unsigned int y1 = gridXstripYTable[value];
const unsigned int x2 = gridYstripXTable[value];
const unsigned int y2 = gridYstripYTable[value];
```

Recap

- Learned how to profile code
- Understand resource management by changing and testing block dimension sizes
- Made attempts to parallelize and speed up code
- Understand and build off an existing project