Final Project: Cache Bypassing Analysis and Performance Study

Brian Park

North Carolina State University, Computer Engineering 786

April 2023

1 Cache Efficiency Analysis

The configuration used is `SM2_GTX480`. I set the `gpgpu_max_insn` to 100M. And benchmarked from Rodinia are BP, HS, and LUD benchmarks. From ISPASS, I chose BFS, LPS, and NQU.

To determine the category, cache insensitive is classified as any improvement less than 10% or -10%. Otherwise, it is cache unfriendly or cache friendly. The results are shown in Table 1. Since the full description cannot fit in the table, the columns are as follows: benchmark name, kernel name, kernel launch UID, IPC with no cache bypassing, IPC with cache bypassing, percentage change of comparing the IPC with/without bypassing, and the benchmark category.

Table 1: Cache Efficiency Analysis Table

Benchmark	Kernel	UID	IPC	IPC w/ Bypass	Improvement	Category
BP	_Z22bpnn_layerforward_CUDAPfS_S_S_ii	1	675.6067	671.3728	-0.63%	Insensitive
HS	_Z14calculate_tempiPfS_S_iiiiffffff	1	701.3718	707.6299	0.89%	Insensitive
LUD	_Z12lud_diagonalPfii	1	0.7026	0.7176	2.13%	Insensitive
LUD	_Z13lud_perimeterPfii	2	9.2446	9.1103	-1.45%	Insensitive
LUD	_Z12lud_internalPfii	3	501.2445	567.1572	13.15%	Unfriendly
LUD	_Z12lud_diagonalPfii	4	0.7558	0.7742	2.43%	Insensitive
LUD	_Z13lud_perimeterPfii	5	10.9464	11.8102	7.89%	Insensitive
LUD	_Z12lud_internalPfii	6	497.3745	574.7466	15.56%	Unfriendly
LUD	_Z12lud_diagonalPfii	7	0.7558	0.7741	2.42%	Insensitive
LUD	_Z13lud_perimeterPfii	8	10.1697	10.9718	7.89%	Insensitive
LUD	_Z12lud_internalPfii	9	473.0808	557.2787	17.8%	Unfriendly
LUD	_Z12lud_diagonalPfii	10	0.7558	0.7741	2.42%	Insensitive
LUD	_Z13lud_perimeterPfii	11	9.3893	10.1287	7.87%	Insensitive
LUD	_Z12lud_internalPfii	12	462.4784	529.6388	14.52%	Unfriendly
LUD	_Z12lud_diagonalPfii	13	0.7558	0.7741	2.42%	Insensitive
LUD	_Z13lud_perimeterPfii	14	8.6082	9.2874	7.89%	Insensitive
LUD	_Z12lud_internalPfii	15	378.4012	504.6895	33.37%	Unfriendly
LUD	_Z12lud_diagonalPfii	16	0.7558	0.7742	2.43%	Insensitive
LUD	_Z13lud_perimeterPfii	17	7.8294	8.4467	7.88%	Insensitive
LUD	_Z12lud_internalPfii	18	357.2093	493.7370	38.22%	Unfriendly
LUD	_Z12lud_diagonalPfii	19	0.7558	0.7742	2.43%	Insensitive
LUD	_Z13lud_perimeterPfii	20	7.0473	7.6040	7.9%	Insensitive
LUD	_Z12lud_internalPfii	21	338.0277	453.3258	34.11%	Unfriendly
LUD	_Z12lud_diagonalPfii	22	0.7558	0.7741	2.42%	Insensitive
LUD	_Z13lud_perimeterPfii	23	6.2640	6.7609	7.93%	Insensitive
LUD	_Z12lud_internalPfii	24	324.1251	467.1097	44.11%	Unfriendly
LUD	_Z12lud_diagonalPfii	25	0.7558	0.7741	2.42%	Insensitive
LUD	_Z13lud_perimeterPfii	26	5.4832	5.9163	7.9%	Insensitive
LUD	_Z12lud_internalPfii	27	290.9933	405.2074	39.25%	Unfriendly
LUD	_Z12lud_diagonalPfii	28	0.7558	0.7741	2.42%	Insensitive
LUD	_Z13lud_perimeterPfii	29	4.7006	5.0733	7.93%	Insensitive
LUD	_Z12lud_internalPfii	30	246.8571	344.3503	39.49%	Unfriendly
LUD	_Z12lud_diagonalPfii	31	0.7558	0.7741	2.42%	Insensitive
LUD	_Z13lud_perimeterPfii	32	3.9172	4.2288	7.95%	Insensitive
LUD	_Z12lud_internalPfii	33	208.6225	253.2766	21.4%	Unfriendly
LUD	_Z12lud_diagonalPfii	34	0.7558	0.7741	2.42%	Insensitive
LUD	_Z13lud_perimeterPfii	35	3.1348	3.3833	7.93%	Insensitive
LUD	_Z12lud_internalPfii	36	142.2966	172.1319	20.97%	Unfriendly
LUD	_Z12lud_diagonalPfii	37	0.7558	0.7741	2.42%	Insensitive
LUD	_Z13lud_perimeterPfii	38	2.3514	2.5387	7.97%	Insensitive
LUD	_Z12lud_internalPfii	39	111.9498	134.8471	20.45%	Unfriendly
LUD	_Z12lud_diagonalPfii	40	0.7558	0.7741	2.42%	Insensitive
LUD	_Z13lud_perimeterPfii	41	1.5679	1.6926	7.95%	Insensitive
LUD	_Z12lud_internalPfii	42	39.4499	44.9208	13.87%	Unfriendly
LUD	_Z12lud_diagonalPfii	43	0.7558	0.7741	2.42%	Insensitive
LUD	_Z13lud_perimeterPfii	44	0.8583	0.8467	-1.35%	Insensitive
LUD	_Z12lud_internalPfii	45	16.2623	16.6957	2.67%	Insensitive
LUD	_Z12lud_diagonalPfii	46	0.7558	0.7741	2.42%	Insensitive
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	1	217.5687	167.9066	-22.83%	Friendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	2	206.0139	146.9099	-28.69%	Friendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	3	165.9271	112.0179	-32.49%	Friendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	4	76.2236	61.3361	-19.53%	Friendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	5	21.3021	36.1667	69.78%	Unfriendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	6	22.5533	44.4395	97.04%	Unfriendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	7	46.5675	86.5094	85.77%	Unfriendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	8	354.4445	455.3303	28.46%	Unfriendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	9	473.1056	486.7920	2.89%	Insensitive
LPS	_Z13GPU_laplace3diiiiPfS_	1	383.1095	408.8568	6.72%	Insensitive
NQU	_Z24solve_nqueen_cuda_kerneliiPjS_S_S_i	1	30.4185	30.7699	1.16%	Insensitive

2 Profiling-Based Cache Bypassing

I implemented what was mentioned in the bypassing paper [1]. Surprisingly, the implementation was very simple. I add two data structures, one for storing SM, kernel UID, address information and the reference count in a nested C++ map. I create two of these nested tables, one that is recorded during profiled run and dumped into `profile_dump.out` and one that is reused in the second or future runs where it reads data from `profile_dump.out`. This is simply added in `ldst_unit::memory_cycle` where memory requests are handled. If a memory request has a reference counter less than 3, I bypass the L1D cache. This turns out to be a simple 3 line addition that profiles and does the cache bypassing if a profiled run is available. Additional logic needs to be added in `gpgpusim_entrypoint.cc` where we have to log to `profile_dump.out` and read from `profile_dump.out` using some file and string manipulations. For `git diff` of the implementation, it is available at the bottom of this report.

The benchmarks that I reran and reanalyzed are the ones that were **cache unfriendly**. Those included kernels in LUD and BFS. Table 2 shows the IPC with profiling from V100 configuration. Note that it is not exactly fair to compare against SM2 GTX480, but as the spec stated, SM2 configuration with profiling caused some deadlocks. But, the results look sound and we see that there are some improvements and degradation in IPC with profiling.

Table 2: Cache Efficiency Analysis Table with Profiling

Benchmark	Kernel	UID	IPC	IPC w/ Bypass	IPC w/ Profile	Category
LUD	_Z12lud_diagonalPfii	1	0.7026	0.7176	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	2	9.2446	9.1103	15.3590	Insensitive
LUD	_Z12lud_internalPfii	3	501.2445	567.1572	721.2603	Unfriendly
LUD	_Z12lud_diagonalPfii	4	0.7558	0.7742	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	5	10.9464	11.8102	14.3418	Insensitive
LUD	_Z12lud_internalPfii	6	497.3745	574.7466	642.5734	Unfriendly
LUD	_Z12lud_diagonalPfii	7	0.7558	0.7741	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	8	10.1697	10.9718	13.3236	Insensitive
LUD	_Z12lud_internalPfii	9	473.0808	557.2787	557.6649	Unfriendly
LUD	_Z12lud_diagonalPfii	10	0.7558	0.7741	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	11	9.3893	10.1287	12.3045	Insensitive
LUD	_Z12lud_internalPfii	12	462.4784	529.6388	506.7778	Unfriendly
LUD	_Z12lud_diagonalPfii	13	0.7558	0.7741	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	14	8.6082	9.2874	11.3033	Insensitive
LUD	_Z12lud_internalPfii	15	378.4012	504.6895	429.6447	Unfriendly
LUD	_Z12lud_diagonalPfii	16	0.7558	0.7742	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	17	7.8294	8.4467	10.2805	Insensitive
LUD	_Z12lud_internalPfii	18	357.2093	493.7370	357.2629	Unfriendly
LUD	_Z12lud_diagonalPfii	19	0.7558	0.7742	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	20	7.0473	7.6040	9.2568	Insensitive
LUD	_Z12lud_internalPfii	21	338.0277	453.3258	293.3001	Unfriendly
LUD	_Z12lud_diagonalPfii	22	0.7558	0.7741	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	23	6.2640	6.7609	8.2322	Insensitive
LUD	_Z12lud_internalPfii	24	324.1251	467.1097	235.2133	Unfriendly
LUD	_Z12lud_diagonalPfii	25	0.7558	0.7741	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	26	5.4832	5.9163	7.2186	Insensitive
LUD	_Z12lud_internalPfii	27	290.9933	405.2074	180.4473	Unfriendly
LUD	_Z12lud_diagonalPfii	28	0.7558	0.7741	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	29	4.7006	5.0733	6.1903	Insensitive
LUD	_Z12lud_internalPfii	30	246.8571	344.3503	132.7995	Unfriendly
LUD	_Z12lud_diagonalPfii	31	0.7558	0.7741	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	32	3.9172	4.2288	5.1610	Insensitive
LUD	_Z12lud_internalPfii	33	208.6225	253.2766	92.7536	Unfriendly
LUD	_Z12lud_diagonalPfii	34	0.7558	0.7741	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	35	3.1348	3.3833	4.1308	Insensitive
LUD	_Z12lud_internalPfii	36	142.2966	172.1319	59.3623	Unfriendly
LUD	_Z12lud_diagonalPfii	37	0.7558	0.7741	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	38	2.3514	2.5387	3.1034	Insensitive
LUD	_Z12lud_internalPfii	39	111.9498	134.8471	33.5850	Unfriendly
LUD	_Z12lud_diagonalPfii	40	0.7558	0.7741	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	41	1.5679	1.6926	2.0726	Insensitive
LUD	_Z12lud_internalPfii	42	39.4499	44.9208	14.9430	Unfriendly
LUD	_Z12lud_diagonalPfii	43	0.7558	0.7741	0.7782	Insensitive
LUD	_Z13lud_perimeterPfii	44	0.8583	0.8467	1.0381	Insensitive
LUD	_Z12lud_internalPfii	45	16.2623	16.6957	3.7422	Insensitive
LUD	_Z12lud_diagonalPfii	46	0.7558	0.7741	0.7782	Insensitive
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	1	217.5687	167.9066	104.4680	Friendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	2	206.0139	146.9099	90.2066	Friendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	3	165.9271	112.0179	77.9455	Friendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	4	76.2236	61.3361	72.6998	Friendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	5	21.3021	36.1667	66.7220	Unfriendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	6	22.5533	44.4395	66.8945	Unfriendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	7	46.5675	86.5094	111.0300	Unfriendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	8	354.4445	455.3303	175.9334	Unfriendly
BFS	_Z6KernelP4NodePiPbS2_S1_S2_i	9	473.1056	486.7920	149.3736	Insensitive

To see the improvements on cache unfriendly kernels better, Table 3 is shown with only the cache unfriendly data filtered out. We see that 6 out of 18 kernels that are cache unfriendly are improved in IPC with profiling. It may seem low, but if we look at some of the cache insensitive kernels, many of them are noticeably improved. It seems that the 10% limit set by me could make these results look skewed, but it also tells us that in the extreme case of cache unfriendly kernels with 10% difference with bypassing/no bypassing may not bring drastic improvements.

Table 3: Cache Efficiency Analysis Table with Profiling (Cache Unfriendly Kernels Only)

Benchmark	Kernel	UID	IPC	IPC w/ Bypass	IPC w/ Profile	Category
LUD	`_Z12lud_internalPfii`	3	501.2445	567.1572	721.2603	Unfriendly
LUD	`_Z12lud_internalPfii`	6	497.3745	574.7466	642.5734	Unfriendly
LUD	`_Z12lud_internalPfii`	9	473.0808	557.2787	557.6649	Unfriendly
LUD	`_Z12lud_internalPfii`	12	462.4784	529.6388	506.7778	Unfriendly
LUD	`_Z12lud_internalPfii`	15	378.4012	504.6895	429.6447	Unfriendly
LUD	`_Z12lud_internalPfii`	18	357.2093	493.7370	357.2629	Unfriendly
LUD	`_Z12lud_internalPfii`	21	338.0277	453.3258	293.3001	Unfriendly
LUD	`_Z12lud_internalPfii`	24	324.1251	467.1097	235.2133	Unfriendly
LUD	`_Z12lud_internalPfii`	27	290.9933	405.2074	180.4473	Unfriendly
LUD	`_Z12lud_internalPfii`	30	246.8571	344.3503	132.7995	Unfriendly
LUD	`_Z12lud_internalPfii`	33	208.6225	253.2766	92.7536	Unfriendly
LUD	`_Z12lud_internalPfii`	36	142.2966	172.1319	59.3623	Unfriendly
LUD	`_Z12lud_internalPfii`	39	111.9498	134.8471	33.5850	Unfriendly
LUD	`_Z12lud_internalPfii`	42	39.4499	44.9208	14.9430	Unfriendly
BFS	`_Z6KernelP4NodePiPbS2_S1_S2_i`	5	21.3021	36.1667	66.7220	Unfriendly
BFS	`_Z6KernelP4NodePiPbS2_S1_S2_i`	6	22.5533	44.4395	66.8945	Unfriendly
BFS	`_Z6KernelP4NodePiPbS2_S1_S2_i`	7	46.5675	86.5094	111.0300	Unfriendly
BFS	`_Z6KernelP4NodePiPbS2_S1_S2_i`	8	354.4445	455.3303	175.9334	Unfriendly

To reproduce the results from the analysis, all you have to do is run the same benchmarks twice, but copying the new V100 config over. For convenience, a bash script is provided, but it still requires to be run twice in order to produce the proper results.

Please make sure the root directory has this structure with the script:

```
ls ~
ISPASS
analyis.sh
env
gpgpu-sim_distribution
rodinia
```

Then you can run `analysis.sh` twice, making sure to edit the configs and changes to GPGPU-Sim properly:

```
bash analysis.sh
bash analysis.sh
```

```
diff --git a/src/gpgpu-sim/shader.cc b/src/gpgpu-sim/shader.cc
index c6e7b8f..da102a5 100644
--- a/src/gpgpu-sim/shader.cc
+++ b/src/gpgpu-sim/shader.cc
@@ -52,6 +52,8 @@
 #define MAX(a, b) (((a) > (b)) ? (a) : (b))
 #define MIN(a, b) (((a) < (b)) ? (a) : (b))

+std::map<unsigned, std::map<unsigned, std::map<unsigned, unsigned>>> SM_kernel_addr_table;
+
 mem_fetch *shader_core_mem_fetch_allocator::alloc(
     new_addr_type addr, mem_access_type type, unsigned size, bool wr,
     unsigned long long cycle) const {
@@ -2039,6 +2041,12 @@ bool ldst_unit::memory_cycle(warp_inst_t &inst,
   const mem_access_t &access = inst.accessq_back();

   bool bypassL1D = false;
+  SM_kernel_addr_table[m_core->get_sid()][m_core->get_kernel()->get_uid()][access.get_addr()]++;
```

```
+
+  if (profile_table[m_core->get_sid()][m_core->get_kernel()->get_uid()][access.get_addr()] < 3){
+    bypassL1D = true;
+  }
+
   if (CACHE_GLOBAL == inst.cache_op || (m_L1D == NULL)) {
     bypassL1D = true;
   } else if (inst.space.is_global()) {  // global memory access
diff --git a/src/gpgpu-sim/shader.h b/src/gpgpu-sim/shader.h
index 6481790..ed34d41 100644
--- a/src/gpgpu-sim/shader.h
+++ b/src/gpgpu-sim/shader.h
@@ -70,6 +70,9 @@

 #define WRITE_MASK_SIZE 8

+extern std::map<unsigned, std::map<unsigned, std::map<unsigned, unsigned>>> SM_kernel_addr_table;
+extern std::map<unsigned, std::map<unsigned, std::map<unsigned, unsigned>>> profile_table;
+
 class gpgpu_context;

 enum exec_unit_type_t {
diff --git a/src/gpgpusim_entrypoint.cc b/src/gpgpusim_entrypoint.cc
index f4287d8..9610b0b 100644
--- a/src/gpgpusim_entrypoint.cc
+++ b/src/gpgpusim_entrypoint.cc
@@ -28,6 +28,10 @@

 #include "gpgpusim_entrypoint.h"
 #include <stdio.h>
+#include <iostream>
+#include <fstream>
+#include <string>

 #include "../libcuda/gpgpu_context.h"
 #include "cuda-sim/cuda-sim.h"
@@ -42,6 +46,7 @@

 static int sg_argc = 3;
 static const char *sg_argv[] = {"", "-config", "gpgpusim.config"};
+std::map<unsigned, std::map<unsigned, std::map<unsigned, unsigned>>> profile_table;

 void *gpgpu_sim_thread_sequential(void *ctx_ptr) {
   gpgpu_context *ctx = (gpgpu_context *)ctx_ptr;
@@ -229,6 +234,20 @@ gpgpu_sim *gpgpu_context::gpgpu_ptx_sim_init_perf() {
   sem_init(&(the_gpgpusim->g_sim_signal_finish), 0, 0);
   sem_init(&(the_gpgpusim->g_sim_signal_exit), 0, 0);

+  std::ifstream profile_log;
+  profile_log.open("profile_dump.out");
+  if(profile_log.is_open()){
+    std::string line;
+    size_t SM_id, kernel_uid, addr, ref_cnt;
+    while(getline(profile_log, line)){
+      if(line.empty())
+        continue;
+      sscanf(line.c_str(), "SM: %lu, kernel: %lu, addr: %lu, ref: %lu", &SM_id, &kernel_uid, &addr, &ref_c
+      profile_table[SM_id][kernel_uid][addr] = ref_cnt;
+    }
+    profile_log.close();
+  }
+
   return the_gpgpusim->g_the_gpu;
 }

@@ -266,6 +285,18 @@ void gpgpu_context::print_simulation_time() {
   printf("gpgpu_simulation_rate = %u (cycle/sec)\n", cycles_per_sec);
   printf("gpgpu_silicon_slowdown = %ux\n",
          the_gpgpusim->g_the_gpu->shader_clock() * 1000 / cycles_per_sec);
+
+  std::ofstream profile_log;
+  profile_log.open("profile_dump.out");
+  for (auto SM = SM_kernel_addr_table.begin(); SM != SM_kernel_addr_table.end(); SM++) {
+    for (auto kernel = SM->second.begin(); kernel != SM->second.end(); kernel++) {
+      for (auto addr = kernel->second.begin(); addr != kernel->second.end(); addr++) {
+        profile_log << "SM: " << SM->first << ", kernel: " << kernel->first << ", addr: " << addr->first <<
ndl;
+      }
+    }
+  }
+  profile_log.close();
+
   fflush(stdout);
 }
```

References

[1] C. Li, S. L. Song, H. Dai, A. Sidelnik, S. K. S. Hari, and H. Zhou, "Locality-driven dynamic gpu cache bypassing," in *Proceedings of the 29th ACM on International Conference on Supercomputing*, ICS '15, (New York, NY, USA), p. 67–77, Association for Computing Machinery, 2015.