ECE 786 Final Project Report

Task 1:

Result for Task 1 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	Benchmark category
Rodinia	HS	0	701.37	707.63	0.89%	Cache Insensitive
Rodinia	BP	0	670.19	666.36	-0.57%	Cache Insensitive
Rodinia	BP	1	557.29	357.44	-35.86%	Cache Friendly
Rodinia	LUD	0	0.70	0.72	2.86%	Cache Insensitive
Rodinia	LUD	1	5.17	5.18	0.19%	Cache Insensitive
Rodinia	LUD	2	79.00	80.55	1.96%	Cache Insensitive
ISAPASS	NQU	0	30.42	30.77	1.15%	Cache Insensitive
ISAPASS	LPS	0	383.11	408.86	6.72%	Cache Unfriendly
ISAPASS	BFS	0	217.57	167.91	-22.82%	Cache Friendly
ISAPASS	BFS	1	211.63	156.70	-25.96%	Cache Friendly
ISAPASS	BFS	2	193.78	138.25	-28.66%	Cache Friendly
ISAPASS	BFS	8	37.33	61.46	64.64%	Cache Unfriendly

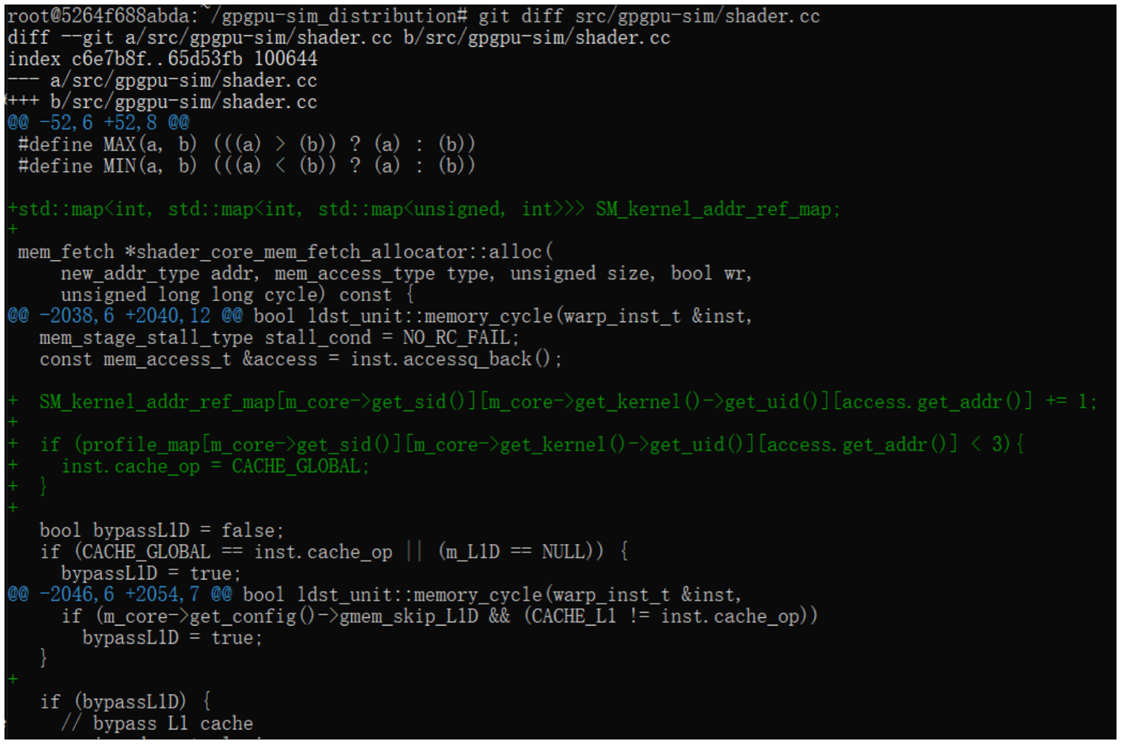
*assuming less than 5% difference is cache insensitive.

Task 2:

Profile-based cache bypass implementation is divided into two major steps - statics collecting and bypass implementing. For statics collecting, a triple-nested map is used to collect the number of access on each address on each SMs and each kernels.

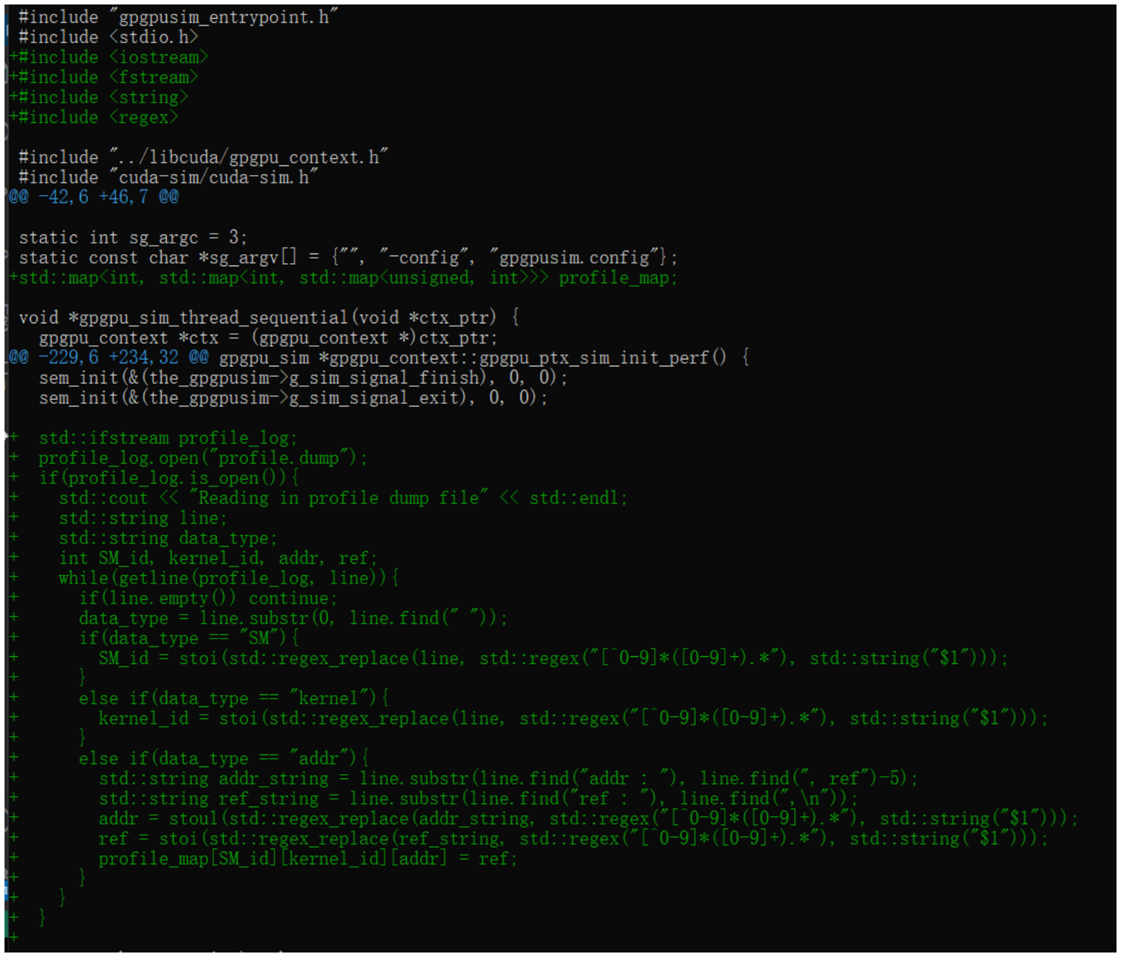
The map is then written to a dump file, which will be read automatically in the second or further runs of simulation. If the dump file is read, another triple-nested map will be created to record the data in the dump file. Then when memory need to be accessed, the map will be accessed based on current SM, kernel, and accessing address to determine if bypass will be used. The whole implementation is shown as code snippet below in the following page.

For the two cache unfriendly benchmarks from task 1 - LPS and BFS 8, after implementing profile-based cache bypass, the new IPC values are 667.44 and 98.60, which are both ~50% higher than the L1 cache bypass.



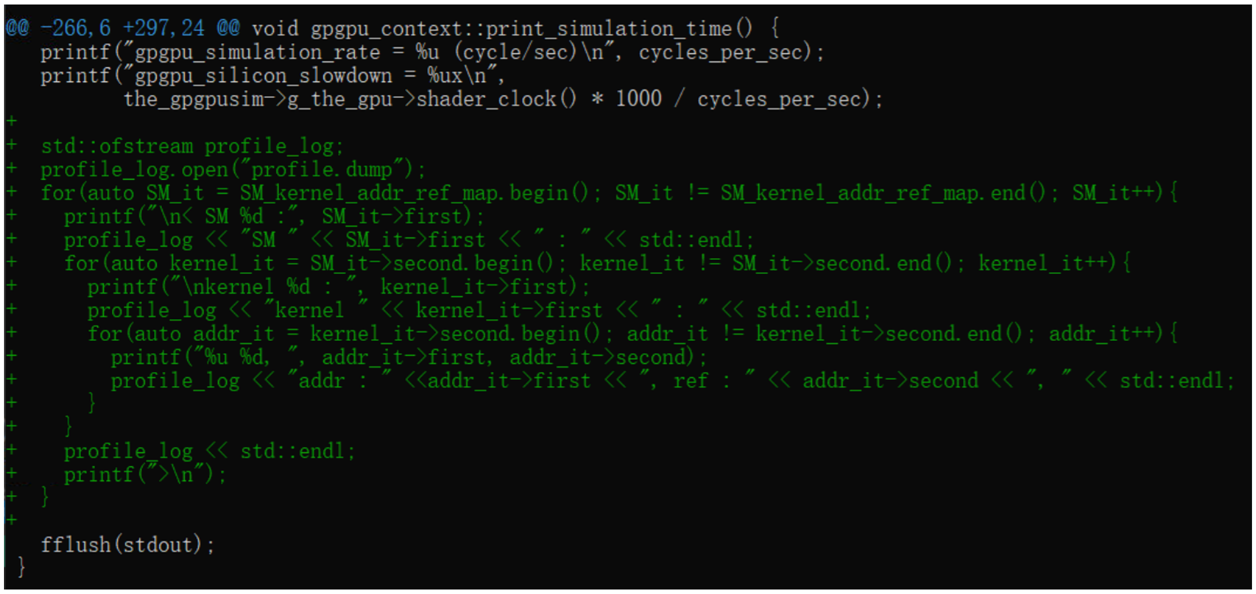
```
root@5264f688abda:~/gpgpu-sim_distribution# git diff src/gpgpu-sim/shader.cc
diff --git a/src/gpgpu-sim/shader.cc b/src/gpgpu-sim/shader.cc
index c6e7b8f..65d53fb 100644
--- a/src/gpgpu-sim/shader.cc
+++ b/src/gpgpu-sim/shader.cc
@@ -52,6 +52,8 @@
 #define MAX(a, b) (((a) > (b)) ? (a) : (b))
 #define MIN(a, b) (((a) < (b)) ? (a) : (b))

+std::map<int, std::map<int, std::map<unsigned, int>>> SM_kernel_addr_ref_map;
+
 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 {
@@ -2038,6 +2040,12 @@ bool ldst_unit::memory_cycle(warp_inst_t &inst,
   mem_stage_stall_type stall_cond = NO_RC_FAIL;
   const mem_access_t &access = inst.accessq_back();

+  SM_kernel_addr_ref_map[m_core->get_sid()][m_core->get_kernel()->get_uid()][access.get_addr()] += 1;
+
+  if (profile_map[m_core->get_sid()][m_core->get_kernel()->get_uid()][access.get_addr()] < 3){
+    inst.cache_op = CACHE_GLOBAL;
+  }
+
   bool bypassL1D = false;
   if (CACHE_GLOBAL == inst.cache_op || (m_L1D == NULL)) {
     bypassL1D = true;
@@ -2046,6 +2054,7 @@ bool ldst_unit::memory_cycle(warp_inst_t &inst,
     if (m_core->get_config()->gmem_skip_L1D && (CACHE_L1 != inst.cache_op))
       bypassL1D = true;
   }
+
   if (bypassL1D) {
     // bypass L1 cache
```

* shader.cc, implementing statics collecting & profile-based bypass logic in ldst_unit::memory_cycle



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

 #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<int, std::map<int, std::map<unsigned, int>>> profile_map;

 void *gpgpu_sim_thread_sequential(void *ctx_ptr) {
   gpgpu_context *ctx = (gpgpu_context *)ctx_ptr;
@@ -229,6 +234,32 @@ 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");
+  if(profile_log.is_open()){
+    std::cout << "Reading in profile dump file" << std::endl;
+    std::string line;
+    std::string data_type;
+    int SM_id, kernel_id, addr, ref;
+    while(getline(profile_log, line)){
+      if(line.empty()) continue;
+      data_type = line.substr(0, line.find(" "));
+      if(data_type == "SM"){
+        SM_id = stoi(std::regex_replace(line, std::regex("[^0-9]*([0-9]+).*"), std::string("$1")));
+      }
+      else if(data_type == "kernel"){
+        kernel_id = stoi(std::regex_replace(line, std::regex("[^0-9]*([0-9]+).*"), std::string("$1")));
+      }
+      else if(data_type == "addr"){
+        std::string addr_string = line.substr(line.find("addr : "), line.find(", ref")-5);
+        std::string ref_string = line.substr(line.find("ref : "), line.find(",\n"));
+        addr = stoul(std::regex_replace(addr_string, std::regex("[^0-9]*([0-9]+).*"), std::string("$1")));
+        ref = stoi(std::regex_replace(ref_string, std::regex("[^0-9]*([0-9]+).*"), std::string("$1")));
+        profile_map[SM_id][kernel_id][addr] = ref;
+      }
+    }
+  }
+
```

* gpgpusim_entrypoint.cc, implementing dump file reading & mapping.



```
@@ -266,6 +297,24 @@ 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");
+  for(auto SM_it = SM_kernel_addr_ref_map.begin(); SM_it != SM_kernel_addr_ref_map.end(); SM_it++){
+    printf("\n< SM %d :", SM_it->first);
+    profile_log << "SM " << SM_it->first << " : " << std::endl;
+    for(auto kernel_it = SM_it->second.begin(); kernel_it != SM_it->second.end(); kernel_it++){
+      printf("\nkernel %d : ", kernel_it->first);
+      profile_log << "kernel " << kernel_it->first << " : " << std::endl;
+      for(auto addr_it = kernel_it->second.begin(); addr_it != kernel_it->second.end(); addr_it++){
+        printf("%u %d, ", addr_it->first, addr_it->second);
+        profile_log << "addr : " <<addr_it->first << ", ref : " << addr_it->second << ", " << std::endl;
+      }
+    }
+    profile_log << std::endl;
+    printf(">\n");
+  }
+
   fflush(stdout);
 }
```

* gpgpusim_entrypoint.cc, implementing statics file dumping.