IRIS Custom Policy

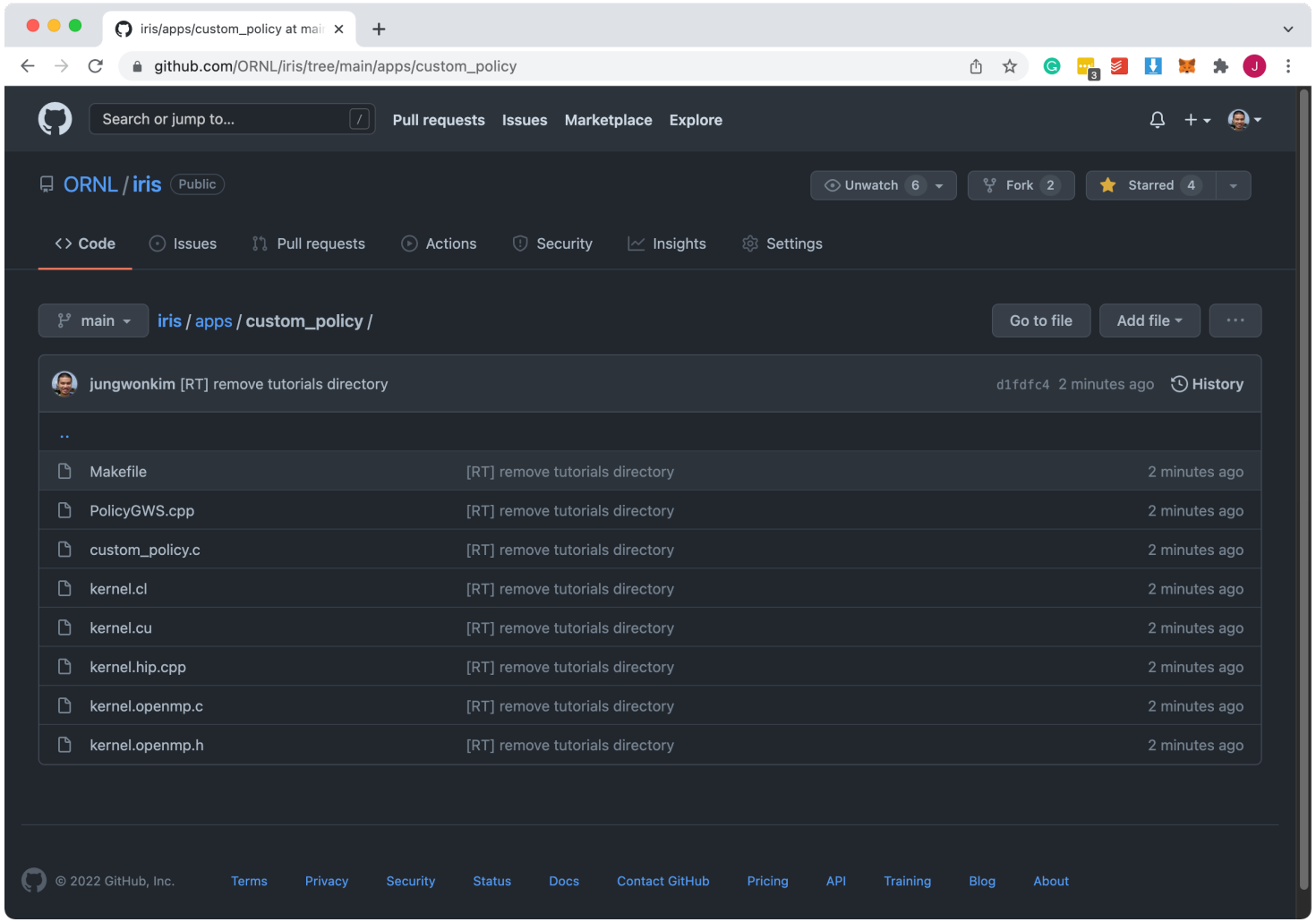
Jungwon Kim

IRIS mini workshop 2022

January 5, 2022

Application: apps/custom_policy

- https://github.com/ORNL/iris/tree/main/apps/custom_policy



Machine: ExCL/Cousteau

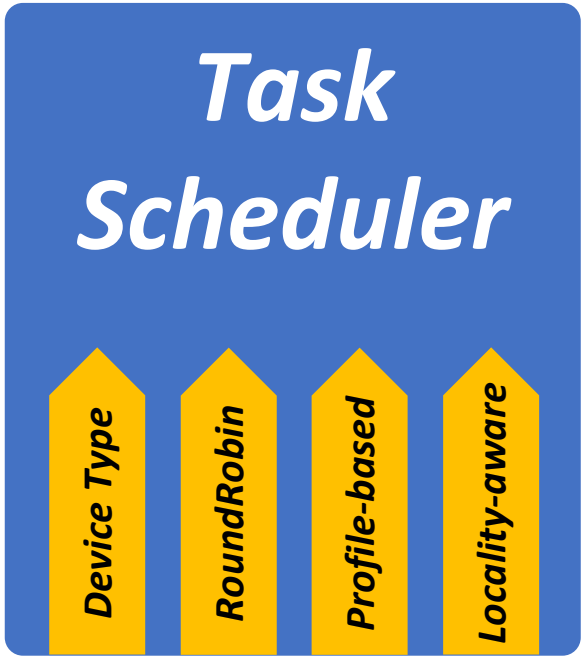
- 2x AMD EPYC 7272 CPUs + 2x AMD MI100 GPUs

```
eck@cousteau:~/work/iris/apps/2tasks$ lscpu | grep 'Socket(s)\|Model name'
Socket(s):                          2
Model name:                         AMD EPYC 7272 12-Core Processor
eck@cousteau:~/work/iris/apps/2tasks$ rocm-smi --showhw


======================== ROCm System Management Interface ========================
============================== Concise Hardware Info ==============================
GPU  DID   GFX RAS  SDMA RAS  UMC RAS  VBIOS              BUS
0    738c  ENABLED  ENABLED   ENABLED  113-D3430500-030   0000:29:00.0
1    738c  ENABLED  ENABLED   ENABLED  113-D3431500-100   0000:85:00.0
===================================================================================
================================ End of ROCm SMI Log ==============================
eck@cousteau:~/work/iris/apps/2tasks$
```

IRIS: *Intelligent* Runtime System

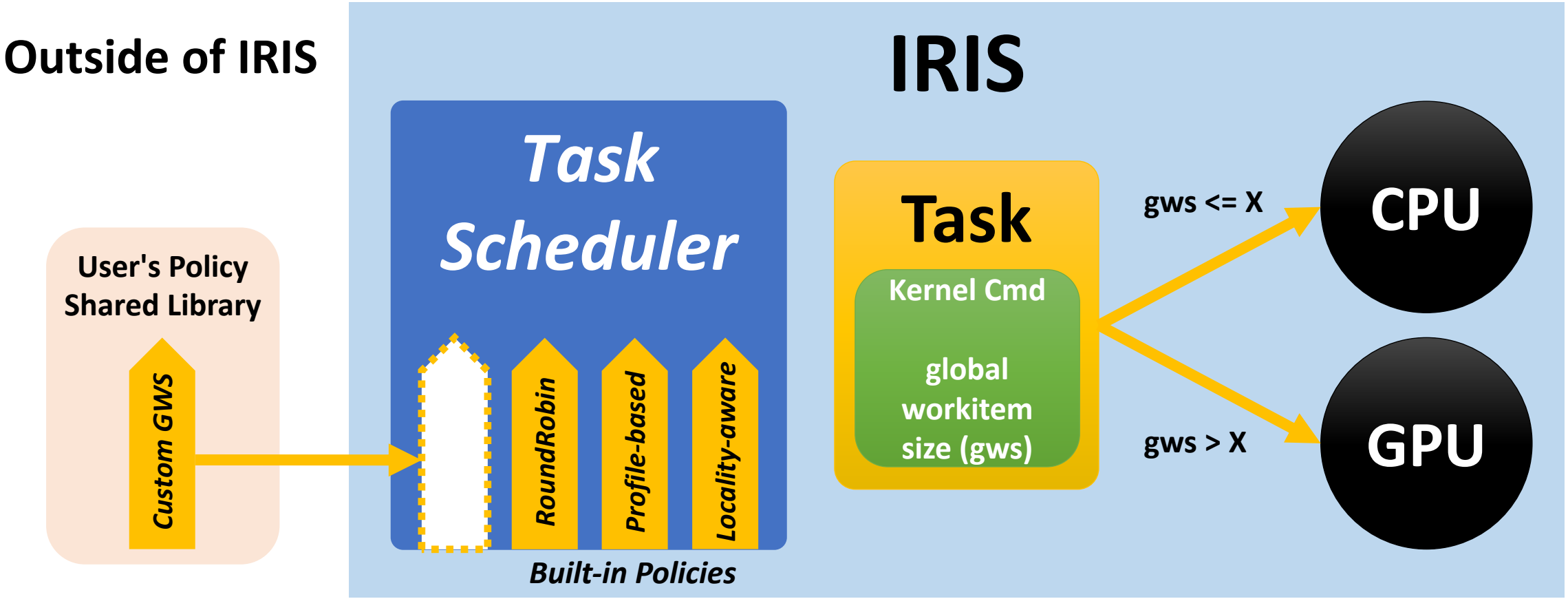
- IRIS can provide intelligent policies
  - Device selection policies
  - Kernel selection policies



Built-in Device Selection Policies

Pluginable Custom Policies

- Users can write their own device selection policies and plug them in IRIS at run time



custom_policy/PolicyGWS.cpp

```cpp
#include <iris/iris.h>
#include <iris/rt/Policy.h>
#include <iris/rt/Command.h>
#include <iris/rt/Device.h>
#include <iris/rt/Task.h>

namespace brisbane {
namespace rt {

class PolicyGWS: public Policy {
public:
  PolicyGWS() {}
  virtual ~PolicyGWS() {}
  virtual void Init(void* params) {
    threshold_ = (size_t) params;
  }
  virtual void GetDevices(Task* task, Device** devs, int* ndevs) {
    Command* cmd = task->cmd_kernel();
    size_t* gws = cmd->gws();
    size_t total_work_items = gws[0] * gws[1] * gws[2];
    int target_dev = total_work_items > threshold_ ? iris_gpu : iris_cpu;
    int devid = 0;
    for (int i = 0; i < ndevices(); i++)
      if (device(i)->type() & target_dev) devs[devid++] = device(i);
    *ndevs = devid;
  }

  size_t threshold_;
};

} /* namespace runtime */
} /* namespace brisbane */

REGISTER_CUSTOM_POLICY(PolicyGWS, custom_gws)
```

- Implement a new subclass of Policy
  - Two virtual functions: Init(), GetDevices()
- Call REGISTER_CUSTOM_POLICY(class_name, policy_name)
- Build a shared library
  - g++ -std=c++11 -fPIC -shared -o **libPolicyGWS.so** PolicyGWS.cpp

src/runtime/Scheduler.cpp

PolicyGWS.cpp buffers

```cpp
#include <iris/iris.h>
#include <iris/rt/Policy.h>
#include <iris/rt/Command.h>
#include <iris/rt/Device.h>
#include <iris/rt/Task.h>

namespace brisbane {
namespace rt {

class PolicyGWS: public Policy {
public:
  PolicyGWS() {}
  virtual ~PolicyGWS() {}
  virtual void Init(void* params) {
    threshold_ = (size_t) params;
  }
  virtual void GetDevices(Task* task, Device** devs, int* ndevs) {
    Command* cmd = task->cmd_kernel();
    size_t* gws = cmd->gws();
    size_t total_work_items = gws[0] * gws[1] * gws[2];
    int target_dev = total_work_items > threshold_ ? iris_gpu : iris_cpu;
    int devid = 0;
    for (int i = 0; i < ndevices(); i++)
      if (device(i)->type() & target_dev) devs[devid++] = device(i);
    *ndevs = devid;
  }

  size_t threshold_;
};

} /* namespace runtime */
} /* namespace brisbane */

REGISTER_CUSTOM_POLICY(PolicyGWS, custom_gws)
```

NORMAL PolicyGWS.cpp cpp 100% ℓ:35 c:1

Scheduler.cpp buffers

```cpp
  if (task->marker()) {
    std::vector<Task*>* subtasks = task->subtasks();
    for (std::vector<Task*>::iterator I = subtasks->begin(), E = subtasks->end(); I != E; ++I) {
      Task* subtask = *I;
      int dev = subtask->devno();
      workers_[dev]->Enqueue(subtask);
    }
    return;
  }
  if (!task->HasSubtasks()) {
    SubmitTask(task);
    return;
  }
  std::vector<Task*>* subtasks = task->subtasks();
  for (std::vector<Task*>::iterator I = subtasks->begin(), E = subtasks->end(); I != E; ++I)
    SubmitTask(*I);
}

void Scheduler::SubmitTask(Task* task) {
  int brs_policy = task->brs_policy();
  char* opt = task->opt();
  int ndevs = 0;
  Device* devs[BRISBANE_MAX_NDEVS];
  if (brs_policy < BRISBANE_MAX_NDEVS) {
    if (brs_policy >= ndevs_) ndevs = 0;
    else {
      ndevs = 1;
      devs[0] = devs_[brs_policy];
    }
  } else policies_->GetPolicy(brs_policy, opt)->GetDevices(task, devs, &ndevs);
  if (ndevs == 0) {
    int dev_default = platform_->device_default();
    _trace("no device for policy[0x%x], run the task on device[%d]", brs_policy, dev_default);
    ndevs = 1;
    devs[0] = devs_[dev_default];
  }
  for (int i = 0; i < ndevs; i++) {
    devs[i]->worker()->Enqueue(task);
    if (hub_available_) hub_client_->TaskInc(devs[i]->devno(), 1);
  }
}

} /* namespace rt */
} /* namespace brisbane */
```

NORMAL Scheduler.cpp cpp utf-8[unix] 100% ℓ:144/144 c:26 [58]tra…

custom_policy/custom_policy.c

custom_policy.c

```c
#include <iris/iris.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
  iris_init(&argc, &argv, 1);

  size_t SIZE = argc > 1 ? atol(argv[1]) : 8;
  int* A = (int*) malloc(SIZE * sizeof(int));

  iris_mem memA;
  iris_mem_create(SIZE * sizeof(int), &memA);

  iris_register_policy("libPolicyGWS.so", "custom_gws", (void*) 16);

  void* params[1] = { memA };
  int params_info[1] = { iris_w };
  iris_task task;
  iris_task_create(&task);
  iris_task_kernel(task, "setid", 1, NULL, &SIZE, NULL, 1, params, params_info);
  iris_task_d2h_full(task, memA, A);
  iris_task_submit(task, iris_custom, "custom_gws", 1);

  printf("A[");
  for (int i = 0; i < SIZE; i++) printf("%3d", A[i]);
  printf("]\n");

  iris_finalize();

  return 0;
}
```

- iris_register_policy(
  shared_library_path,
  policy_name,
  init_params);

- iris_task_submit(…
  iris_custom,
  policy_name, …);

custom_policy/kernel.hip.cpp

```
kernel.hip.cpp                                                    buffers
1 #include <hip/hip_runtime.h>
2
3 extern "C" __global__ void setid(int* mem) {
4   int id = blockIdx.x * blockDim.x + threadIdx.x;
5   mem[id] = id;
6 }
7
N…  kernel.hip.cpp                                  12% ℓ:1/8≡ ℃:1
```

custom_policy/custom_policy 8 → CPU

```
eck@cousteau:~/work/iris/tutorials/custom_policy$ ./custom_policy
[I] cousteau [Platform.cpp:140:Init] IRIS architectures[openmp:cuda:hip:levelzero:hexagon:opencl]
[T] cousteau [Platform.cpp:412:InitOpenMP] OpenMP platform[0] ndevs[1]
[I] cousteau [DeviceOpenMP.cpp:31:DeviceOpenMP] device[0] platform[0] device[AMD EPYC 7272 12-Core Processor] type[64]
[T] cousteau [Loader.cpp:39:LoadHandle] libcuda.so: cannot open shared object file: No such file or directory
[T] cousteau [Platform.cpp:287:InitCUDA] skipping CUDA architecture
[T] cousteau [Platform.cpp:333:InitHIP] HIP platform[1] ndevs[2]
[I] cousteau [DeviceHIP.cpp:29:DeviceHIP] device[1] platform[1] vendor[Advanced Micro Devices] device[] ordinal[0] type[256]
 version[AMD HIP 40421432]
[I] cousteau [DeviceHIP.cpp:29:DeviceHIP] device[2] platform[1] vendor[Advanced Micro Devices] device[] ordinal[1] type[256]
 version[AMD HIP 40421432]
[T] cousteau [Loader.cpp:39:LoadHandle] libze_loader.so: cannot open shared object file: No such file or directory
[T] cousteau [Platform.cpp:356:InitLevelZero] skipping LevelZero architecture
[T] cousteau [Loader.cpp:39:LoadHandle] kernel.hexagon.so: cannot open shared object file: No such file or directory
[T] cousteau [Platform.cpp:428:InitHexagon] skipping Hexagon architecture
[T] cousteau [Platform.cpp:454:InitOpenCL] OpenCL nplatforms[1]
[T] cousteau [Platform.cpp:464:InitOpenCL] OpenCL platform[AMD Accelerated Parallel Processing] from [Advanced Micro Devices
, Inc.]
[T] cousteau [Platform.cpp:471:InitOpenCL] skipping platform[2] [Advanced Micro Devices, Inc. AMD Accelerated Parallel Proce
ssing] ndevs[0]
[T] cousteau [Loader.cpp:39:LoadHandle] kernel.poly.so: cannot open shared object file: No such file or directory
[I] cousteau [Platform.cpp:996:InitScheduler] Scheduler ndevs[3] ndevs_enabled[3]
[I] cousteau [DeviceHIP.cpp:70:Init] devid[1] max_compute_units[120] max_work_group_size_[1024] max_work_item_sizes[21990232
54528,2199023254528,2199023254528] max_block_dims[1024,1024,1024] concurrent_kernels[1]
[T] cousteau [DeviceHIP.cpp:79:Init] dev[2][] kernels[/tmp/iris/kernel.hip-2]
[I] cousteau [DeviceHIP.cpp:70:Init] devid[0] max_compute_units[120] max_work_group_size_[1024] max_work_item_sizes[21990232
54528,2199023254528,2199023254528] max_block_dims[1024,1024,1024] concurrent_kernels[1]
[T] cousteau [DeviceHIP.cpp:79:Init] dev[1][] kernels[/tmp/iris/kernel.hip-1]
[I] cousteau [Platform.cpp:185:Init] nplatforms[2] ndevs[3] ndevs_enabled[3] scheduler[1] hub[0] polyhedral[0] profile[0]
[T] cousteau [Policies.cpp:85:Register] lib[libPolicyGWS.so] name[custom_gws]
[T] cousteau [DeviceOpenMP.cpp:138:KernelLaunch] dev[0] kernel[setid] dim[1] off[0] gws[8]
[T] cousteau [Device.cpp:66:Execute] task[6] complete dev[0][AMD EPYC 7272 12-Core Processor] time[0.030093]
A[  0  1  2  3  4  5  6  7]
[I] cousteau [Platform.cpp:1022:ShowKernelHistory] kernel[brisbane_null] k[0.000000][0] h2d[0.000000][0] d2h[0.000000][0]
[I] cousteau [Platform.cpp:1022:ShowKernelHistory] kernel[setid] k[0.030089][1] h2d[0.000000][0] d2h[0.000004][1]
[I] cousteau [Platform.cpp:1027:ShowKernelHistory] total kernel[0.030089] h2d[0.000000] d2h[0.000004]
[I] cousteau [Platform.cpp:1051:Finalize] total execution time:[2.442537] sec. initialize:[1.398077] sec. t-i:[1.044459] sec
[I] cousteau [Platform.cpp:1052:Finalize] t10[0.000000] t11[0.000000] t12[0.000000] t13[0.000000]
[I] cousteau [Platform.cpp:1053:Finalize] t14[0.000000] t15[0.000000] t16[0.000000] t17[0.000000]
[I] cousteau [Platform.cpp:1054:Finalize] t18[0.000000] t19[0.000000] t20[0.000000] t21[0.000000]
eck@cousteau:~/work/iris/tutorials/custom_policy$
```

custom_policy/custom_policy 32 → GPU

```
eck@cousteau:~/work/iris/tutorials/custom_policy$ ./custom_policy 32
[I] cousteau [Platform.cpp:140:Init] IRIS architectures[openmp:cuda:hip:levelzero:hexagon:opencl]
[T] cousteau [Platform.cpp:412:InitOpenMP] OpenMP platform[0] ndevs[1]
[I] cousteau [DeviceOpenMP.cpp:31:DeviceOpenMP] device[0] platform[0] device[AMD EPYC 7272 12-Core Processor] type[64]
[T] cousteau [Loader.cpp:39:LoadHandle] libcuda.so: cannot open shared object file: No such file or directory
[T] cousteau [Platform.cpp:287:InitCUDA] skipping CUDA architecture
[T] cousteau [Platform.cpp:333:InitHIP] HIP platform[1] ndevs[2]
[I] cousteau [DeviceHIP.cpp:29:DeviceHIP] device[1] platform[1] vendor[Advanced Micro Devices] device[] ordinal[0] type[256]
 version[AMD HIP 40421432]
[I] cousteau [DeviceHIP.cpp:29:DeviceHIP] device[2] platform[1] vendor[Advanced Micro Devices] device[] ordinal[1] type[256]
 version[AMD HIP 40421432]
[T] cousteau [Loader.cpp:39:LoadHandle] libze_loader.so: cannot open shared object file: No such file or directory
[T] cousteau [Platform.cpp:356:InitLevelZero] skipping LevelZero architecture
[T] cousteau [Loader.cpp:39:LoadHandle] kernel.hexagon.so: cannot open shared object file: No such file or directory
[T] cousteau [Platform.cpp:428:InitHexagon] skipping Hexagon architecture
[T] cousteau [Platform.cpp:454:InitOpenCL] OpenCL nplatforms[1]
[T] cousteau [Platform.cpp:464:InitOpenCL] OpenCL platform[AMD Accelerated Parallel Processing] from [Advanced Micro Devices
, Inc.]
[T] cousteau [Platform.cpp:471:InitOpenCL] skipping platform[2] [Advanced Micro Devices, Inc. AMD Accelerated Parallel Proce
ssing] ndevs[0]
[T] cousteau [Loader.cpp:39:LoadHandle] kernel.poly.so: cannot open shared object file: No such file or directory
[I] cousteau [Platform.cpp:996:InitScheduler] Scheduler ndevs[3] ndevs_enabled[3]
[I] cousteau [DeviceHIP.cpp:70:Init] devid[0] max_compute_units[120] max_work_group_size_[1024] max_work_item_sizes[21990232
54528,2199023254528,2199023254528] max_block_dims[1024,1024,1024] concurrent_kernels[1]
[T] cousteau [DeviceHIP.cpp:79:Init] dev[1][] kernels[/tmp/iris/kernel.hip-1]
[I] cousteau [DeviceHIP.cpp:70:Init] devid[1] max_compute_units[120] max_work_group_size_[1024] max_work_item_sizes[21990232
54528,2199023254528,2199023254528] max_block_dims[1024,1024,1024] concurrent_kernels[1]
[T] cousteau [DeviceHIP.cpp:79:Init] dev[2][] kernels[/tmp/iris/kernel.hip-2]
[I] cousteau [Platform.cpp:185:Init] nplatforms[2] ndevs[3] ndevs_enabled[3] scheduler[1] hub[0] polyhedral[0] profile[0]
[T] cousteau [Policies.cpp:85:Register] lib[libPolicyGWS.so] name[custom_gws]
[T] cousteau [DeviceHIP.cpp:184:KernelLaunch] dev[1] kernel[setid] dim[1] grid[32,1,1] block[1,1,1] shared_mem_bytes[0] q[0]
[T] cousteau [Device.cpp:66:Execute] task[6] complete dev[1][] time[0.508504]
A[  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31]
[I] cousteau [Platform.cpp:1022:ShowKernelHistory] kernel[brisbane_null] k[0.000000][0] h2d[0.000000][0] d2h[0.000000][0]
[I] cousteau [Platform.cpp:1022:ShowKernelHistory] kernel[setid] k[0.507782][1] h2d[0.000000][0] d2h[0.000722][1]
[I] cousteau [Platform.cpp:1027:ShowKernelHistory] total kernel[0.507782] h2d[0.000000] d2h[0.000722]
[I] cousteau [Platform.cpp:1051:Finalize] total execution time:[2.420700] sec. initialize:[1.408066] sec. t-i:[1.012634] sec
[I] cousteau [Platform.cpp:1052:Finalize] t10[0.000000] t11[0.000000] t12[0.000000] t13[0.000000]
[I] cousteau [Platform.cpp:1053:Finalize] t14[0.000000] t15[0.000000] t16[0.000000] t17[0.000000]
[I] cousteau [Platform.cpp:1054:Finalize] t18[0.000000] t19[0.000000] t20[0.000000] t21[0.000000]
eck@cousteau:~/work/iris/tutorials/custom_policy$
```