

IRIS Custom Policy

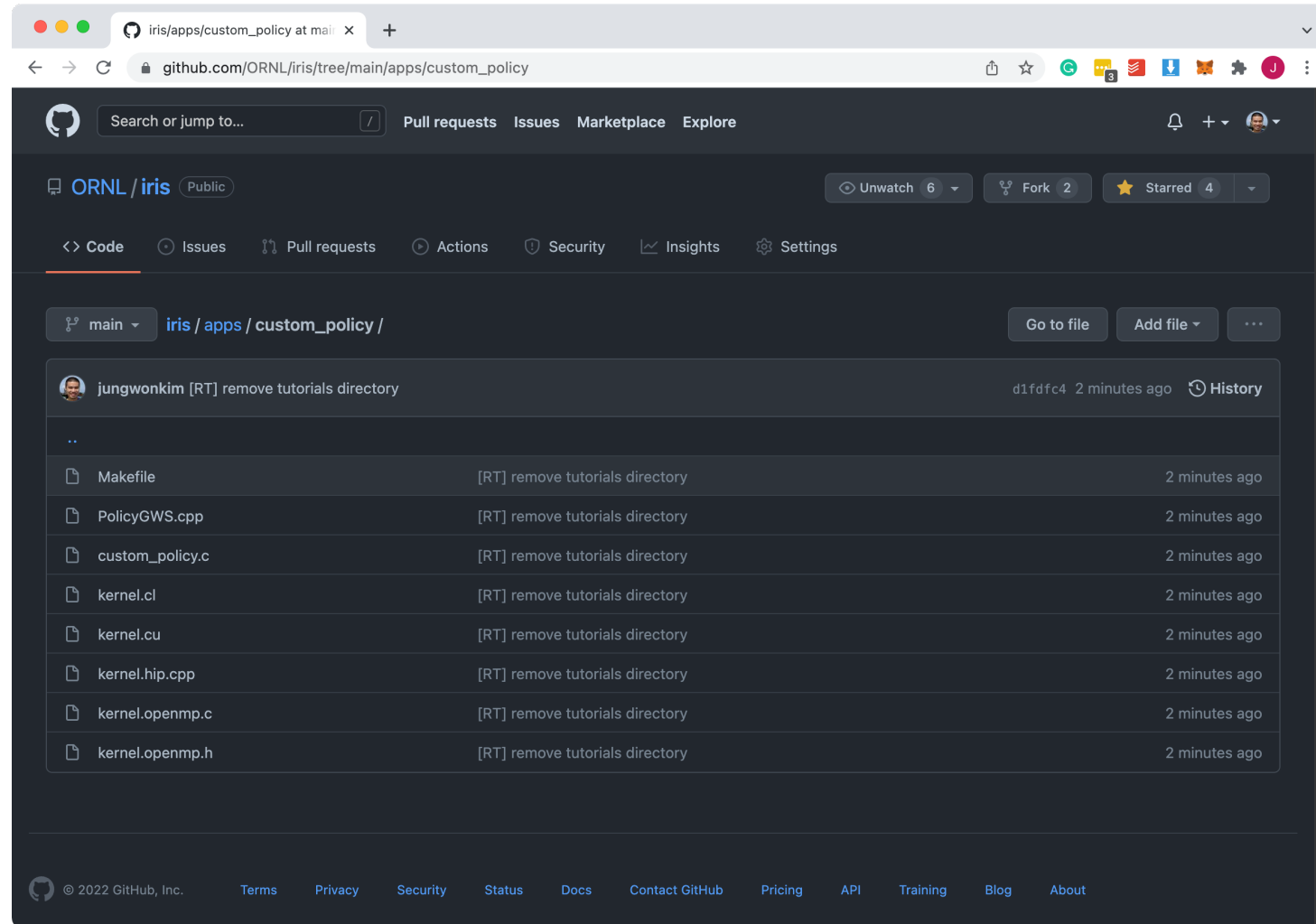
Jungwon Kim

IRIS mini workshop 2022

January 5, 2022

Application: tutorials/custom_policy

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



Machine: ExCL/Cousteau

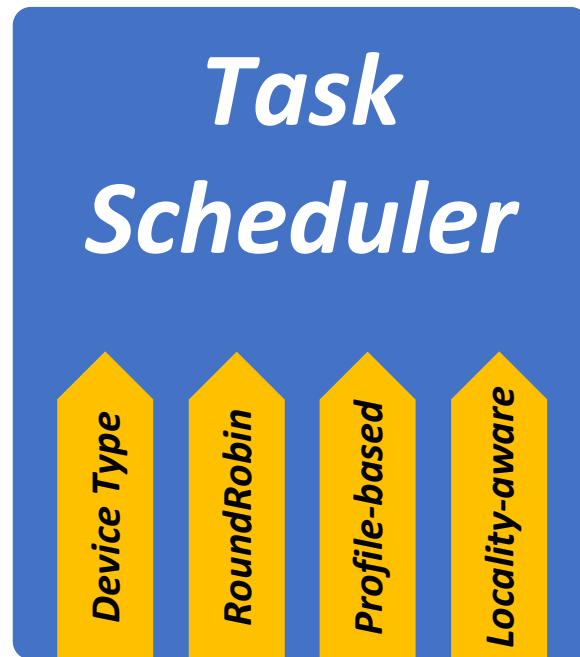
- 2x AMD EPYC 7272 CPUs + 2x AMD MI100 GPUs

```
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

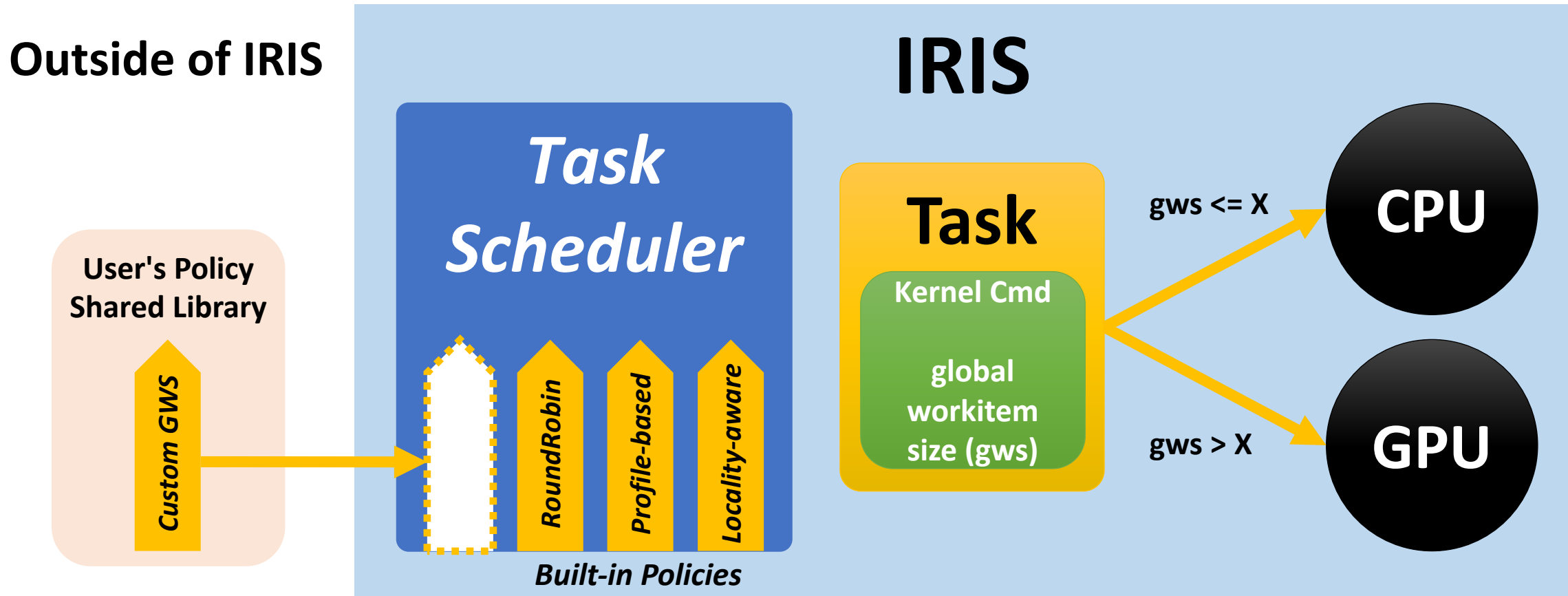
- IRIS can provide intelligent policies
 - Device selection policies
 - Kernel selection policies



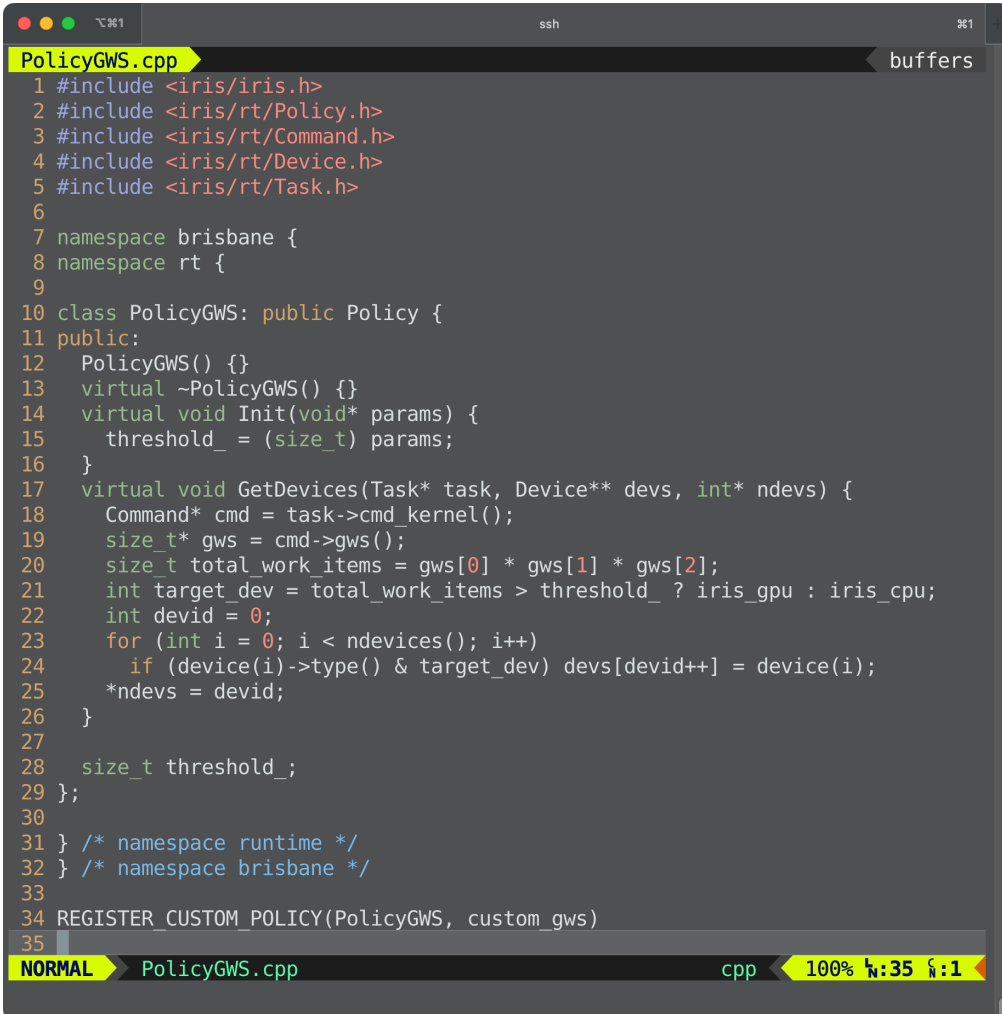
Built-in Device Selection Policies

Pluginable Custom Policies

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



custom_policy/PolicyGWS.cpp



```
1 #include <iris/iris.h>
2 #include <iris/rt/Policy.h>
3 #include <iris/rt/Command.h>
4 #include <iris/rt/Device.h>
5 #include <iris/rt/Task.h>
6
7 namespace brisbane {
8 namespace rt {
9
10 class PolicyGWS: public Policy {
11 public:
12     PolicyGWS() {}
13     virtual ~PolicyGWS() {}
14     virtual void Init(void* params) {
15         threshold_ = (size_t) params;
16     }
17     virtual void GetDevices(Task* task, Device** devs, int* ndevs) {
18         Command* cmd = task->cmd_kernel();
19         size_t* gws = cmd->gws();
20         size_t total_work_items = gws[0] * gws[1] * gws[2];
21         int target_dev = total_work_items > threshold_ ? iris_gpu : iris_cpu;
22         int devid = 0;
23         for (int i = 0; i < ndevs(); i++)
24             if (device(i)->type() & target_dev) devs[devid++] = device(i);
25         *ndevs = devid;
26     }
27
28     size_t threshold_;
29 };
30
31 } /* namespace runtime */
32 } /* namespace brisbane */
33
34 REGISTER_CUSTOM_POLICY(PolicyGWS, custom_gws)
35
```

- 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
1 #include <iris/iris.h>
2 #include <iris/rt/Policy.h>
3 #include <iris/rt/Command.h>
4 #include <iris/rt/Device.h>
5 #include <iris/rt/Task.h>
6
7 namespace brisbane {
8 namespace rt {
9
10 class PolicyGWS: public Policy {
11 public:
12     PolicyGWS() {}
13     virtual ~PolicyGWS() {}
14     virtual void Init(void* params) {
15         threshold_ = (size_t) params;
16     }
17     virtual void GetDevices(Task* task, Device** devs, int* ndevs) {
18         Command* cmd = task->cmd_kernel();
19         size_t* gws = cmd->gws();
20         size_t total_work_items = gws[0] * gws[1] * gws[2];
21         int target_dev = total_work_items > threshold_ ? iris_gpu : iris_cpu;
22         int devid = 0;
23         for (int i = 0; i < ndevices(); i++)
24             if (device(i)->type() & target_dev) devs[devid++] = device(i);
25         *ndevs = devid;
26     }
27
28     size_t threshold_;
29 };
30
31 } /* namespace runtime */
32 } /* namespace brisbane */
33
34 REGISTER_CUSTOM_POLICY(PolicyGWS, custom_gws)
35
```

NORMAL PolicyGWS.cpp cpp 100% 35:1

```
Scheduler.cpp
101 if (task->marker()) {
102     std::vector<Task*>* subtasks = task->subtasks();
103     for (std::vector<Task*>::iterator I = subtasks->begin(), E = subtasks->end(); I
104 != E; ++I) {
105         Task* subtask = *I;
106         int dev = subtask->devno();
107         workers_[dev]->Enqueue(subtask);
108     }
109     return;
110 }
111 if (!task->HasSubtasks()) {
112     SubmitTask(task);
113     return;
114 }
115 std::vector<Task*>* subtasks = task->subtasks();
116 for (std::vector<Task*>::iterator I = subtasks->begin(), E = subtasks->end(); I !=
117 E; ++I)
118     SubmitTask(*I);
119
120 void Scheduler::SubmitTask(Task* task) {
121     int brs_policy = task->brs_policy();
122     char* opt = task->opt();
123     int ndevs = 0;
124     Device* devs[BRISBANE_MAX_NDEVS];
125     if (brs_policy < BRISBANE_MAX_NDEVS) {
126         if (brs_policy >= ndevs_) ndevs = 0;
127         else {
128             ndevs = 1;
129             devs[0] = devs_[brs_policy];
130         }
131     } else policies->GetPolicy(brs_policy, opt)->GetDevices(task, devs, &ndevs);
132     if (ndevs == 0) {
133         int dev_default = platform->device_default();
134         trace("no device for policy[0x%x], run the task on device[%d]", brs_policy, dev
135 _default);
136         ndevs = 1;
137         devs[0] = devs_[dev_default];
138     }
139     for (int i = 0; i < ndevs; i++) {
140         devs[i]->worker()->Enqueue(task);
141         if (hub_available_) hub_client->TaskInc(devs[i]->devno(), 1);
142     }
143 }
144 } /* namespace brisbane */
```

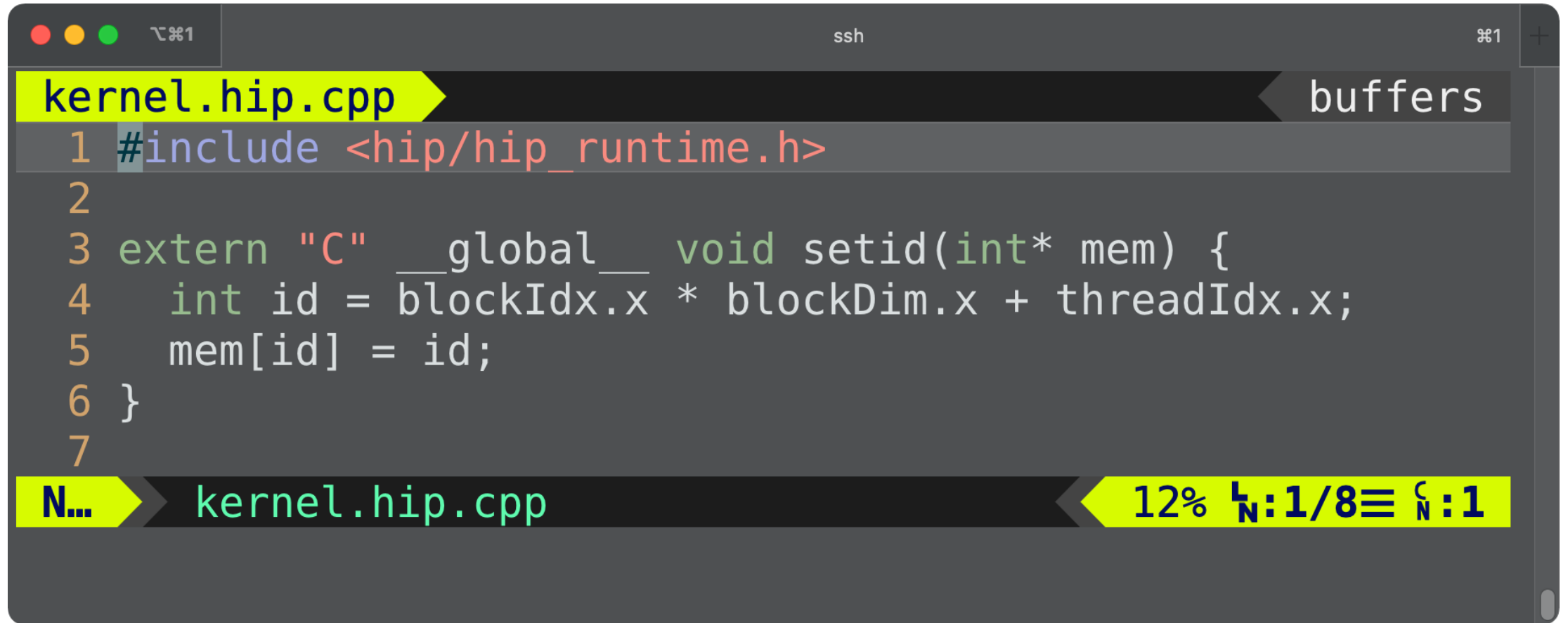
NORMAL Scheduler.cpp cpp utf-8[unix] 100% 144:144 26: [58]tra...

custom_policy/custom_policy.c



```
1 #include <iris/iris.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 int main(int argc, char** argv) {
6     iris_init(&argc, &argv, 1);
7
8     size_t SIZE = argc > 1 ? atol(argv[1]) : 8;
9     int* A = (int*) malloc(SIZE * sizeof(int));
10
11     iris_mem memA;
12     iris_mem_create(SIZE * sizeof(int), &memA);
13
14     iris_register_policy("libPolicyGWS.so", "custom_gws", (void*) 16);
15
16     void* params[1] = { memA };
17     int params_info[1] = { iris_w };
18     iris_task task;
19     iris_task_create(&task);
20     iris_task_kernel(task, "setid", 1, NULL, &SIZE, NULL, 1, params, params_info);
21     iris_task_d2h_full(task, memA, A);
22     iris_task_submit(task, iris_custom, "custom_gws", 1);
23
24     printf("A[");
25     for (int i = 0; i < SIZE; i++) printf("%3d", A[i]);
26     printf("]\n");
27
28     iris_finalize();
29
30     return 0;
31 }
32
```


custom_policy/kernel.hip.cpp



The image shows a code editor window with a dark theme. The title bar at the top has three colored circles (red, yellow, green) on the left, the text 'ssh' in the center, and a tab icon on the right. The editor displays the file 'kernel.hip.cpp' with the following code:

```
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
```

At the bottom of the editor, there is a progress bar. The left side of the bar is labeled 'N...' and the right side is labeled '12% N:1/8 ≡ N: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 Processing] 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 54528,21990232 54528,21990232 54528] 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 54528,21990232 54528,21990232 54528] 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 Processing] 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 54528,21990232 54528,21990232 54528] 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 54528,21990232 54528,21990232 54528] 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$
```