Managing GPUs in HTCondor 8.1/8.2

John (TJ) Knoeller
Condor Week 2014
Better support for GPUs in HTCondor 8.1/8.2

- GPUs as a form of custom resource
- Custom resources enhanced
  - Assign a specific GPU to a job
- Simpler configuration
Define a custom STARTD resource

- `MACHINERESOURCE_<tag>`
- `MACHINERESOURCE_INVENTORY_<tag>`

- `<tag>` is case preserving, case insensitive
- For GPU resources use the tag “GPUs”
  - The plural, not the singular. (like “Cpus”)
  - Because matchmaking
Fungible resources

- Works with HTCondor 8.0
- For OS virtualized resources
  - Cpus, Memory, Disk
- For intangible resources
  - Bandwidth
  - Licenses?
- Works with Static and Partitionable slots
Fungible custom resource example : bandwidth (1)

> condor_config_val -dump Bandwidth
MACHINE_RESOURCE_Bandwidth = 1000

> grep -i bandwidth userjob.submit
REQUEST_Bandwidth = 200
Fungible custom resource example : bandwidth (2)

Assuming 4 static slots

```bash
> condor_status -long | grep -i bandwidth
Bandwidth = 250
DetectedBandwidth = 1000
TotalBandwidth = 1000
TotalSlotBandwidth = 250
```
Non-fungible resources

› New for HTCondor 8.1/8.2
› For resources not virtualized by OS
  • GPUs, Instruments, Directories
› Configure by listing resource ids
  • Quantity is inferred
› Specific id(s) are assigned to slots
› Works with Static and Partitionable slots
Non-fungible custom resource example: GPUs (1)

> condor_config_val -dump gpus
MACHINE_RESOURCE_GPUs = CUDA0, CUDA1
ENVIRONMENT_FOR_AssignedGPUs = CUDA_VISIBLE_DEVICES
ENVIRONMENT_VALUE_FOR_UnAssignedGPUs = 10000

> grep -i gpus userjob.submit
REQUEST_GPUs = 1
Non-fungible custom resource example: GPUs (2)

```bash
> condor_status -long slot1| grep -i gpus

AssignedGpus = "CUDA0"
DetectedGPUs = 2
GPUs = 1
TotalSlotGPUs = 1
TotalGPUs = 2
```
Non-fungible custom resource example: GPUs (3)

› Environment of a job running on that slot

```
> env | grep -i CUDA
_CONDOR_AssignedGPUs = CUDA0
CUDA_VISIBLE_DEVICES = 0
```
Additional resource attributes

› Run a resource inventory script
  • MACHINE_RESOURCE_INVENTORY_<tag>

› Script must return
  • Detected<tag> = <quantity>
    or
  • Detected<tag> = "<list-of-ids>"

› All script output is published in all slots
  • Script output must be ClassAd syntax
> condor_gpu_discovery -properties
DetectedGPUs = "CUDA0, CUDA1"
CUDACapability = 2.0
CUDADeviceName = "GeForce GTX 480"
CUDADriverVersion = 4.2
CUDAECCEnabled = false
CUDAGlobalMemoryMb = 1536
CUDARuntimeVersion = 4.10
condor_gpu_discovery extra

- More attributes with `-extra` option
  - Clock speed, CUs
- Dynamic attributes with `-dynamic` option
  - Fan speed, Power usage, Die temp
- Non homogeneous attributes have GPU id in their name
  - `CUDA0PowerUsage_mw`
- Fake it with `-simulate[:n,m]` option
Using condor_gpu_discovery

› In your configuration file, add
  
  use feature : gpus

› The line above expands to

  MACHINE_RESOURCE_INVENTORY_GPUs = \n      $(LIBEXEC)/condor_gpu_discovery -properties \n      $(GPU_DISCOVERY_EXTRA)

  ENVIRONMENT_FOR_AssignedGPUs = \n      GPU_DEVICE_ORDINAL=/(CUDA|OCL)\ // CUDA_VISIBLE_DEVICES

  ENVIRONMENT_VALUE_FOR_UnAssignedGPUs=10000
Taking a GPU offline

› Add the following to your configuration

```bash
OFFLINE_MACHINERESOURCE_GPUs=CUDA0
```

› Configuration can be set remotely

```bash
condor_config_val -startd -set
```

› Then restart the STARTD

```bash
condor_restart [--peaceful] -startd
```
What’s new in 8.1 (review)

› Non-fungible custom resources
› Take a custom resource offline
› condor_gpu_discovery now defines non-fungible GPUs resource
› STARTD policy for custom resources
  • Don’t abort when resource quantity is 0
  • Give out resource until gone, then give out 0
Any Questions?