STARTD configuration

John (TJ) Knoeller
Condor Week 2018

Default Behavior

- One static slot per hypercore
- Other resources (Disk, Memory, etc) are Divided evenly among the slots
- Want more/fewer slots?

  NUM_CPUS=<desired-num-slots>
- Each slot is single core slot

Slot Resource Allocation

- Default resources quantities are detected
  - DETECTED_CPUS (maybe hyper)
    - DETECTED_PHYSICAL_CPUS (not hyper)
    - DETECTED_CORES (always hyper)
  - DETECTED_MEMORY
- Use $(DETECTED_CPUS) in config
- There is no “DETECTED_DISK”

Control Available Resources

- Set Arbitrary values for CPUs, Memory
- HTCondor will allocate resources to slots *as if the values are correct*

```
NUM_CPUS = 99
MEMORY = $(DETECTED_MEMORY)*99
```

- Default values:

```
NUM_CPUS = $(DETECTED_CPUS)
MEMORY = $(DETECTED_MEMORY)
```

Control how resources are allocated to slots

- Up to 10 slot ‘types’
- Each type defines
  - A Number of slots of that type
  - An Allocation policy for each

```
NUM_SLOTS_TYPE_1 = $(DETECTED_CPUS)/2
SLOT_TYPE_1 = Cpus=2, Memory=4Gb
```

Custom Attributes

- Define your own slot attributes that jobs can match against.
- If you want the slot to have

```
HasMatlab = true
```

- Define value in config, and then add name to STARTD_ATTRS, like this

```
HasMatlab = true
STARTD_ATTRS = $(STARTD_ATTRS) HasMatlab
```

ex: Custom Attributes

Add this to your configuration

```
if $(IsStartd)
  include command into $(LOCAL_DIR)/soft.cfg : $(BIN)/detect_soft.cmd
else
  include ifexist : $(LOCAL_DIR)/soft.cfg
endif
if defined PythonExecutablePath
  STARTD_ATTRS = $(STARTD_ATTRS) PythonExecutablePath
endif
```

detect_soft.cmd (for windows)

```
@key=HKLM\Software\Python\PythonCore\3.6
@for /F "tokens=1,2*" %%I in ('reg query %key%') do (
  @echo Python%%I="%%~sfK"
)
```

Dynamic Custom Attributes

- Register a script to define attributes
  - Script returns a ClassAd
  - Attributes are merged into Slot ClassAds

```
STARTD_CRON_JOB_LIST = tag
STARTD_CRON_tag_EXECUTABLE = detect.sh
```

- Run once or periodically
- Control which slots get the attributes with `SlotMergeConstraint` or `SlotId`

StartdCron config templates

- use FEATURE :

  StartdCronOneShot(<tag>,<exe>[,<args>])

  - Run <exe> once on startup and on reconfig

  StartdCronPeriodic(<tag>,<sec>,<exe>[,<args>])

  - Run <exe> every <sec> seconds

  StartdCronContinuous(<tag>,<exe>[,<args>])

  - Run <exe> and monitor stdout

ex: Startd Share Check

Add this to your startd configuration

```
use FEATURE : StartdCronOneShot(SHARECHK, \
  $(LOCAL_DIR)/share_check.cmd, \
  \\server\share HasServerShare )
```

share_check.cmd

```
@echo off
@set found=0
for /F "tokens=3" %%I in ('net use') do (
  if "%~1"=="%%~K" do set found=1
)
@echo %2=%found%
```

ex: Dynamic GPU attributes

Add this to your startd configuration

```
use FEATURE : StartdCronPeriodic(DYNGPU, \
  10, \
  $(LOCAL_DIR)/dynamic_gpu_info.cmd, \
  $(LIBEXEC)/condor_gpu_discovery -dynamic)
```

output of condor_gpu_discovery -dynamic

```
DetectedGPUs="CUDA0"
CUDA0FanSpeedPct=18
CUDA0DieTempC=98
...
```

dynamic_gpu_info.cmd

```
$* | grep -v '^Detected'
```

ex: Slot Monitoring

Add this to your startd configuration

```
use FEATURE : StartdCronContinuous(TRIG, \
  $(BIN)/monitor.sh, \
  $(EXECUTE) )
```

output of monitor.sh

```
SlotName="Slot1_1"
ProcessCount=10
-s1_1
SlotName="slot1_2"
ProcessCount=2
-s1_2 update:true
... sleeps for some time, then outputs new attrs
```

Special Purpose Slots

- When you want to have slots designed to fit a specific job
  - Interactive slots
  - Maintenance slots
  - Whole Machine slots
- Use multiple slot types
- Cross-publishing of slot Attributes allows policy of slot1 to depend on state of slot2

Cross publishing Slot Attributes

- Policy expressions sometimes need to refer to attributes from other slots
- Cross-publish with `STARTD_SLOT_ATTRS`

```
STARTD_SLOT_ATTRS = State, Activity
```

- Each slot's attrs published in ALL slot ads with SlotN, so you can do this:

```
START = $(START) && (SlotId==2 \
   && Slot1_State != "Claimed") \
   && …
```

Special Purpose Slots Example

```
# define standard static slots
NUM_SLOTS_TYPE_1 = $(DETECTED_PHYSICAL_CPUS)
SLOT_TYPE_1 = cpus=1, memory=$(DETECTED_MEMORY)-1000

# define a maintenance slot
NUM_CPUS = $(DETECTED_REAL_CPUS)+1
NUM_SLOTS_TYPE_2 = 1
SLOT_TYPE_2 = cpus=1, memory=1000
SLOT_TYPE_2_NAME_PREFIX = maint
SLOT_TYPE_2_START = owner=="tannenba“
SLOT_TYPE_2_PREEMPT = false
```

Advanced Slot Specialization

- `SLOT_TYPE_N_attr` where *attr* is

```
START                       WANT_SUSPEND
SUSPEND                     WANT_VACATE
CONTINUE                    WANT_HOLD
PREEMPT                     WANT_HOLD_REASON
KILL                        PERIOIC_CHECKPOINT
CLAIM_WORKLIFE              START_BACKFILL
MaxJobRetirementTime        EVICT_BACKFILL
MachineMaxVacateTime        FetchWorkDelay
Rank                        ...MORE...
SlotWeight
```

Defining a custom resource

- Define a custom STARTD resource
  - **`MACHINE_RESOURCE_<tag>`**
  - **`MACHINE_RESOURCE_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

- For OS virtualized resources
  - Cpus, Memory, Disk
- For intangible resources
  - Bandwidth
  - Licenses?

Fungible custom resource example : bandwidth (1)

```
> condor_config_val -dump Bandwidth
MACHINE_RESOURCE_Bandwidth = 1000

> grep -i bandwidth userjob.submit
REQUEST_Bandwidth = 200
```

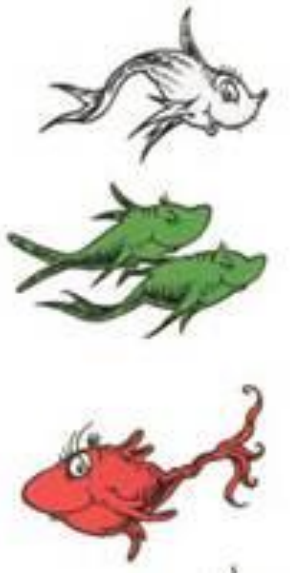
Fungible custom resource example : bandwidth (2)

- Assuming 4 static slots

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

Non-fungible resources

- For resources not virtualized by OS
  - GPUs, Instruments, Directories
- Configure by listing resource ids
  - Quantity is inferred
- Specific id(s) are assigned to 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)

```
> 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

Take a Custom Resource offline

- Add the following to your configuration

```
OFFLINE_MACHINE_RESOURCE_GPUs=CUDA0
```

- Configuration can be set remotely

```
condor_config_val -startd -set
```

- Then restart the STARTD

```
condor_restart [-peaceful] -startd
```

Partitionable Slots

- Partitionable Slot is a Bag of Resouces
  - Usually 1 Partitionable slot per STARTD
- Dynamic Slot carved off to match a job
- Configure

```
SLOT_TYPE_1_PARTITIONABLE=true
NUM_SLOTS_TYPE_1 = 1

SLOT_TYPE_1 = 100%
```

CENTER FOR
HIGH THROUGHPUT
COMPUTING

HTCondor

Any Questions?