HTPC on the OSG

Greg Thain
Computer Sciences Department
University of Wisconsin-Madison

Overview

- What/Why HTPC
- How to set up local HTPC
- GlideIn/WMS for HTPC

Terms: HPC

- Run one job fast
- MPI
- SuperComputer
- Metric:
  - FLOPS
- Fast interconnect

HPC

Goal: Run jobs as fast as possible

Relatively few,

Big parallel jobs

(usually…)

HTC

- Metric:
  - CPU hours/year
- Utilization
- Each core independent
- Lots of serial jobs
- More, cheaper machines
  - No fast interconnects

Hardware trends

- Moore's Law
- Moore cores

- 12 core deployed today
- 48 core tomorrow?
- Machines are not homogenous

Combine the two…

- What do you get?

HTPC, or "Whole Machine"

- HTPC schedules whole machines,
- Not cores

- When an HTPC job lands, not other jobs are running on that machine.

What is this good for?

- Need > 1 core,
  - i.e. “small parallel” jobs
- Need more memory
- Need all local disk throughput
- Need GPU or other dedicated hardware

HTPC is parallel agnostic

- Just schedules a machine,
- Parallelism is up to you
  - MPI
  - Threads
  - Processes
  - OpenMP, etc.
- "Bring it with you"

Sample Apps

- Several MPI-ported apps good fit
- Molecular dynamics:
  - Gromacs, Namd, CHARMM, etc.
- Parallel enabled matlab
- Simple fork-fork-fork jobs

Hybrid HTC - HTPC

- Very powerful approach to science
- Use HTPC to search broadly
- HTC to search deeply

HTPC is not…

- Ability to schedule partial machines
  - (yet)
- Built-in MPI library support
- Magic bullet for scheduling parallel jobs

HTPC gotchas

- Ideally, job needs to run on arbitrary number of cores that are there
- Need to bring MPI or runtimes along
- Limited resource on OSG today
  - Running on half dozen sites
- Still have time limits at most sites

Setting up HTPC locally

- Start locally, act globally
- Depends on local batch system:

- Submitting via GRAM to local scheduler, needs RSL magic token

SGE

- Dunno? Anyone?

LSF

- “exclusive” flag to job description
- Can only ask for whole machines
  - Rsl =(jobtype=single)(exclusive=1)(maxWallTime=2800)"

PBS

- Use host_count and host_xcount:
- rsl="(jobtype=single)(xcount=8)(host_xcount=1)(maxWallTime=2800)"

Condor

- See recipe at
  - http://condor-wiki.cs.wisc.edu
  - https://condor-wiki.cs.wisc.edu/index.cgi/wiki?p=WholeMachineSlots

- #we will double-allocate resources to overlapping slots
- NUM_CPUS = $(DETECTED_CORES)*2
- MEMORY = $(DETECTED_MEMORY)*2 # single-core slots get 1 core each
- SLOT_TYPE_1 = cpus=1
- NUM_SLOTS_TYPE_1 = $(DETECTED_CORES)
- # whole-machine slot gets as many cores and RAM as the machine has
- SLOT_TYPE_2 = cpus=$(DETECTED_CORES), mem=$(DETECTED_MEMORY)
- NUM_SLOTS_TYPE_2 = 1
- # Macro specifying the slot id of the whole-machine slot
- # Example: on an 8-core machine, the whole-machine slot is 9.
- WHOLE_MACHINE_SLOT = ($(DETECTED_CORES)+1)
- # ClassAd attribute that is True/False depending on whether this slot is
- # the whole-machine slot CAN_RUN_WHOLE_MACHINE = SlotID == $(WHOLE_MACHINE_SLOT) STARTD_EXPRS = $(STARTD_EXPRS) CAN_RUN_WHOLE_MACHINE
- # advertise state of each slot as SlotX_State in ClassAds of all other slots
- STARTD_SLOT_EXPRS = $(STARTD_SLOT_EXPRS) State
- # Macro for referencing state of the whole-machine slot.
- # Relies on eval(), which was added in Condor 7.3.2. WHOLE_MACHINE_SLOT_STATE = \ eval (strcat("Slot",$(WHOLE_MACHINE_SLOT),"_State")) # Macro that is true if any single-core slots are claimed
- # WARNING: THERE MUST BE AN ENTRY FOR ALL SLOTS
- # IN THE EXPRESSION BELOW. If you have more slots, you must
- # extend this expression to cover them. If you have fewer
- # slots, extra entries are harmless. SINGLE_CORE_SLOTS_CLAIMED = \ ($(WHOLE_MACHINE_SLOT_STATE) =?= "Claimed") < \ (Slot1_State =?= "Claimed") + \ (Slot2_State =?= "Claimed") + \ (Slot3_State =?= "Claimed") + \ (Slot4_State =?= "Claimed") + \ (Slot5_State =?= "Claimed") + \ (Slot6_State =?= "Claimed") + \ (Slot7_State =?= "Claimed") + \ (Slot8_State =?= "Claimed") + \ (Slot9_State =?= "Claimed") + \ (Slot10_State =?= "Claimed") + \ (Slot11_State =?= "Claimed") + \ (Slot12_State =?= "Claimed") + \ (Slot13_State =?= "Claimed") + \ (Slot14_State =?= "Claimed") + \ (Slot15_State =?= "Claimed") + \ (Slot16_State =?= "Claimed") + \ (Slot17_State =?= "Claimed") + \ (Slot18_State =?= "Claimed") + \ (Slot19_State =?= "Claimed") + \ (Slot20_State =?= "Claimed") + \ (Slot21_State =?= "Claimed") + \ (Slot22_State =?= "Claimed") + \ (Slot23_State =?= "Claimed") + \ (Slot24_State =?= "Claimed") + \ (Slot25_State =?= "Claimed") + \ (Slot26_State =?= "Claimed") + \ (Slot27_State =?= "Claimed") + \ (Slot28_State =?= "Claimed") + \ (Slot29_State =?= "Claimed") + \ (Slot30_State =?= "Claimed") + \ (Slot31_State =?= "Claimed") + \ (Slot32_State =?= "Claimed") + \ (Slot33_State =?= "Claimed") # Single-core jobs must run on single-core slots

Condor

- This is
  - A) Clever use of Condor's flexibly policy
  - B) An egregious hack
  - C) There must be a better way

Example local submit file

```
universe = grid
grid_type       = gt2
globusscheduler = lepton.rcac.purdue.edu/jobmanager-pbs
globusrsl = (jobType=mpi)(queue=standby)(xcount=8)(host_xcount=1)

executable = wrapper.sh

should_transfer_files = yes
when_to_transfer_output = on_exit
transfer_input_files = mdrun, mpiexec, topol.tpr
transfer_output_files = confout.gro, ener.edr, traj.xtc, traj.trr, md.log

output = out.$(CLUSTER)
error  = err.$(CLUSTER)
log    = log
queue
```

Note the wrapper

```
#!/bin/sh
touch output1 output2
chmod 0755 ./mdrun ./mpiexec
./mpiexec -np 8 mdrunome_input_file
```

Common problems

- Usual scheduling problems
- New HTPC problem:
  - Job run, but not exclusively
  - condor_ssh_to_job help
  - run a ps command to verify

Computer Engineer Barbie says:

Parallel programming is hard

HTPC doesn't make it much easier (maybe it is easier to debug on one machine

That was painful
Can there be a better way

- Problems with site-selection
- Everyone needs to know every site
- Everyone needs to know site RSL

Usual OSG Solution:

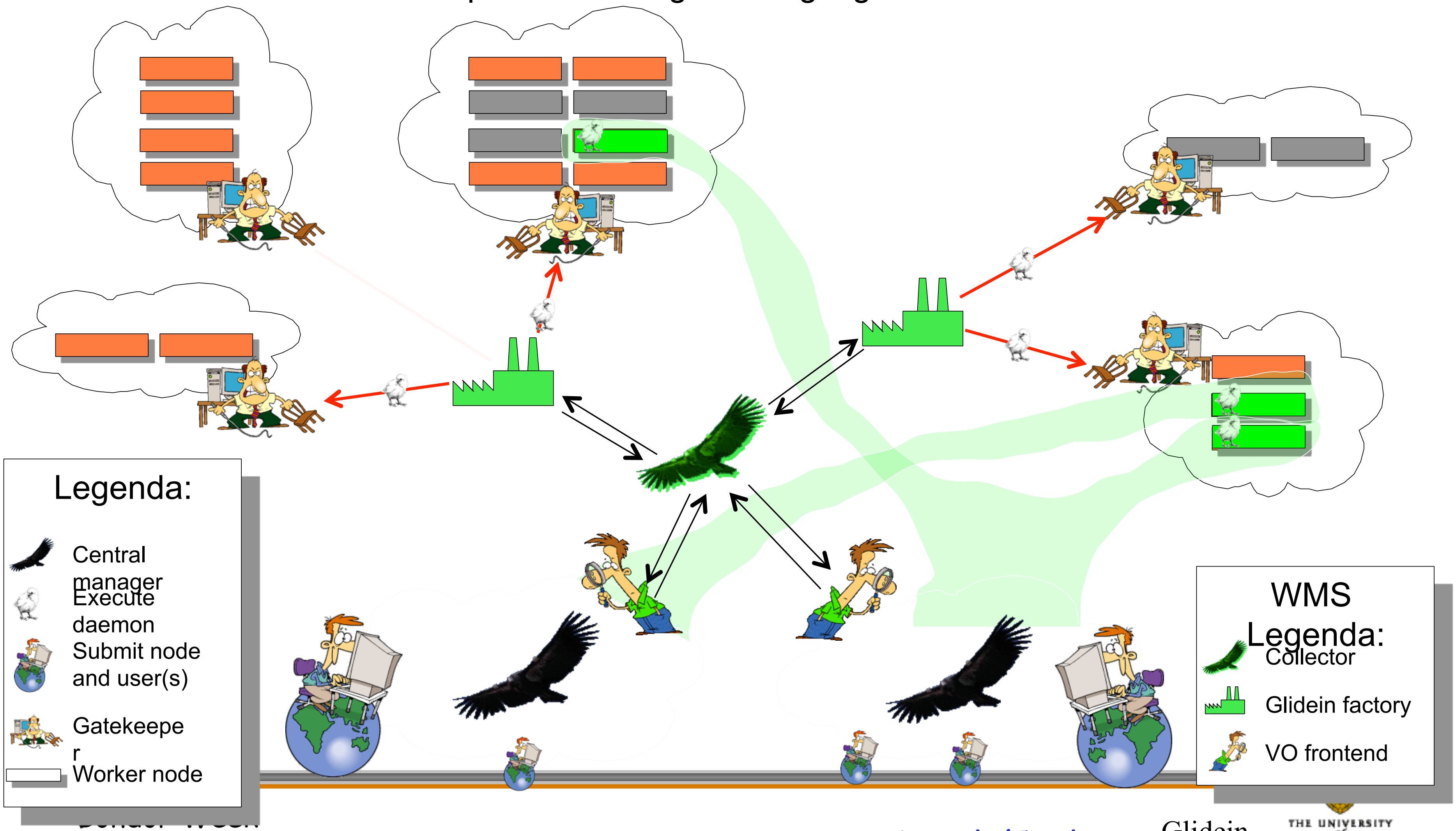
Primary theorem of Computer Science:

Any problem can be fixed with another layer of abstraction

i.e. Pilots

The glideinWMS

http://home.fnal.gov/~sfiligoi/glideinWMS/



Legenda:

- Central manager
- Execute daemon
- Submit node and user(s)
- Gatekeeper
- Worker node

WMS Legenda:

- Collector
- Glidein factory
- VO frontend

Use Glidein “Groups”

- HTPC is a glidein “group”
- A subset of machines with some property
- Each glidein lands on a whole machine
- Each glidein advertises one slot
  - Represents the whole machine

Edit the frontend config:

```
<groups>
      <group name="HTPC" enabled="True">
         <config>
            <idle_glideins_per_entry max="100"
reserve="5"/>
            <idle_vms_per_entry curb="5" max="100"/>
            <running_glideins_per_entry max="10000"
relative_to_queue="1.15"/>
         </config>
         <downtimes/>
```

The trick to select jobs

```
<job query_expr="RequiresWholeMachine">
               <match_attrs>
               </match_attrs>
               <schedds>
               </schedds>
            </job>
```

Machine Attrs

```
<attrs>
            <attr name="CAN_RUN_WHOLE_MACHINE" glidein_publish="True" job_publish="False" parameter="True" type="expr" value="True"/>
            <attr name="GLIDECLIENT_Group_Start" glidein_publish="True" job_publish="False" parameter="True" type="string" value="TARGET.RequiresWholeMachine"/>
         </attrs>
```

Example local submit file

```
universe = grid
grid_type       = gt2
globusscheduler = lepton.rcac.purdue.edu/jobmanager-pbs
globusrsl = (jobType=mpi)(queue=standby)(xcount=8)(host_xcount=1)

executable = wrapper.sh

should_transfer_files = yes
when_to_transfer_output = on_exit
transfer_input_files = mdrun, mpiexec, topol.tpr
transfer_output_files = confout.gro, ener.edr, traj.xtc, traj.trr, md.log

output = out.$(CLUSTER)
error  = err.$(CLUSTER)
log    = log
queue
```

New job submission file

```
universe = vanilla
executable = wrapper.sh
should_transfer_files = yes
when_to_transfer_output = on_exit
transfer_input_files = mdrun, mpiexec, topol.tpr
#transfer_output_files = confout.gro,
  #ener.edr,traj.xtc, traj.trr, md.log
+RequiresWholeMachine=true
REQUIREMENTS = CAN_RUN_WHOLE_MACHINE
output = out.$(CLUSTER)
error  = err.$(CLUSTER)
log    = log
```

Why is this man smiling?

No User proxy!

- Not an HTPC feature
- Plain-old Glidein In feature
- Controversial, but very user friendly.

Glidein provides uniformity

- condor_ssh_to_job works
- file transfer works
- periodic_exprs work

“Glidein”

- What, exactly do we me my glidein?
- What about Panda? (or your fav. Pilot)

HTPC Glue Schema

- New schema for HTPC
  - htpc = enabled
  - htpc_queues = queue1, queue2, queue3
  - # can also take "*"
  - htpc_blacklist_queues =
  - htpc_rsl = (foo=bar) # this is the default for HTPC queues
  - htpc_rsl_queue1 = (zork=quux) # this is a specialized rsl for queue1 only

Want to export your HTPC cluster for all?

- Great! Talk to us

Final benefits of HTPC

- HTC and HTC can work together

- Remove small jobs from supercomputers

Summary

Real users doing real work with HTPC today

Talk to Dan or Greg for more info

Check out HTPC twiki for more info