High Throuput Computing – HTCondor

Andreas Hirczy

TU Graz
Institut für Theoretische Physik – Computational Physics

January 27, 2014

High Performance Computing (HPC)

I want my data **right now!**
Or at least **really soon**.

I don't care (a lot) about the cost!

High Performance Computing (HPC)

I want my data **right now!**
Or at least **really soon**.

I don't care (a lot) about the cost!

MPI is for YOU!

High Performance Computing (HPC)

I want my data **right now!**
Or at least **really soon**.

I don't care (a lot) about the cost!

MPI is for YOU!
For best results go to a supercomputing center.

High Throughput Computing (HTC)

I am interested to get as much CPU cycles as possible from the system till my **presentation next month**!

High Throughput Computing (HTC)

I am interested to get as much CPU cycles as possible from the system till my **presentation next month**!

What's your chance to get a month of wall time on a supercomputing center?

High Throughput Computing (HTC)

I am interested to get as much CPU cycles as possible from the system till my **presentation next month**!

What's your chance to get a month of wall time on a supercomputing center?

You should not really care about FLOPS, but **FLOPM** (FLoating point Operations Per Month) or **FLOPY**.

So what is HTCondor?

One of several existing possibilities to use a collection of mostly idle computers for some additional work.

HTCondor is looking for unused resources in our cluster and tries to use them in the most effective way:

- fair sheduling
- do not overcommit resources
- do not annoy the foreground user (needs some help by the programmer)

What's going on on our Cluster?

```
condor_userprio -all -allusers

condor_status -total
condor_status -servers -total
condor_status -submitters -total

condor_q -global
```

A graphical hint about condor usage is available on our Webserver:
`http://itp.tugraz.at/Comp/Stat_CONDOR/`

Which programmes work with HTCondor?

- No manual interaction!
- somewhat predictable behaviour - you have to know how many CPUs and how many memory you code will use

programm.sh

```
#!/bin/sh
echo "Executable:         $0"
echo "Argument(s):        $*"
echo "excuting on         $(hostname --fqdn)"
echo "Start time is       $(date --iso-8601=seconds)"
echo ; env; echo

CPUs=$(awk '/^Cpus = [[:digit:]]+$/ {print $3}' $_CONDOR_MACHINE_AD
echo "CPUs : $CPUs"

echo "running payload ...."
sleep ${1}

echo "finishing time is $(date --iso-8601=seconds)"
```

submit - part 1

```
# PROGRAMM AND COMMANDLINE PARAMETER(S)
Executable = programm.sh
Arguments  = $$([30 + 10 * $(Process)])

# INPUT
# Input = in.$(Process)

# OUTPUT
Output     = $(Cluster).$(Process).out
Error      = $(Cluster).$(Process).err
Log        = CondorLog
Universe   = vanilla

# REQUIREMENTS
request_cpus   = 2
```

submit - part 2

```
Universe   = vanilla

# REQUIREMENTS
request_cpus   = 2
request_memory = 128
               # MiBytes
request_disk   = 512
               # kiBytes

# NOTIFICATIONS
notification = Error
      # <Always | Complete | Error | Never>

# START PROCESSES
Queue 20
```

excerpt from Makefile

```
$ EXECDIR=/temp/ahi/CondorTest
$ install --directory --mode=755  $(EXECDIR)
$ install programm.sh --mode=755  $(EXECDIR)/
$ install submit      --mode=644  $(EXECDIR)/
$ (cd $(EXECDIR); condor_submit submit)
$ watch -n 1 condor_q # -global
```

What we get!

Let's start those jobs and have a look at the working directory `/temp/ahi/CondorTest/`

OpenMP? . . .

OpenMP? . . .

Works out of the box: request the number of CPUs you need

- more CPUs – more power
- more CPUs – less available machine slots

OpenMPI?

MPICH?

The programme is yours to write :)

The programme is yours to write :) — but to start your programme use a little shell skript:

mpi_launch.sh

```
#!/bin/sh
CPUs=$(awk '/^Cpus = [[:digit:]]+$/ {print $3}' \
          $_CONDOR_MACHINE_AD)

echo "running payload ...."
mpirun -np ${CPUs} your_own_MPI_programm
```

submit file

Modify your submit file:

- set your desired CPU count
- Getenv = true
  no reason for this is known to us, but some environment variable from your running session is missing in the usual condor setup
- run_as_owner = true
  might also work without this setting

submit

```
Executable = mpi_launch.sh
Universe   = vanilla

Error      = job.$(Cluster).$(Process).err
Log        = job.$(Cluster).$(Process).log
Output     = job.$(Cluster).$(Process).out

request_cpus   =  8
request_memory = 12 * 1024
Getenv         = true
run_as_owner   = true
requirements = (TARGET.Machine =!= "faepop12.tu-graz.ac.at")

Queue
```

Jobs with dependencies

Maybe sometime later – if you are nosy, have a look at
`http://research.cs.wisc.edu/htcondor/dagman/dagman.html`

The End

These slides and some accompanying files are available on `http://itp.tugraz.at/~ahi/Computer/HTCondor/`.

You might also be interested in tomorrows webinar “Managing Large Datasets with Python and HDF5” by Andrew Collette: `http://oreillynet.com/pub/e/2990`.