Dylan Stark
Department of Computer Science
Louisiana State University
28 de julio de 2008

Supporting cast:
Chirag Dekate

Practical High Performance Computing for the Modern Age

Lab 1

Course Overview

- Introduction
  - An overview of HPC
  - Throughput computing
  - The Condor middleware service
- Distributed Memory: MPI
  - Commodity Clusters
  - Communicating sequential processes (CSP)
  - MPI programming
- Shared Memory: OpenMP
  - Single Node Architecture
  - Enabling Technologies – Memory, Core Architectures,..
  - OpenMP programming
- System Software
  - Operating Systems
  - Parallel I/O
  - Visualization
- Conclusions
  - What's beyond the scope of this course
  - What form will the future of HPC take

Lab Overview

- Review lab setup
  - Workstations
  - Celeritas
- Familiarize students with practical HPC tools
  - Linux environment
  - Remote connections with SSH toolchain
  - PBS batch scheduler


Lab Overview

- Following the lecture series
  - Condor today
  - MPI tomorrow
  - OpenMP Wednesday

- Worksheets of annotated exercises
  - Useful references
  - Explanation alongside examples
  - Self-pacing


Today's Outline

- The Unix environment
  - Setting up directories and moving around
  - Writing and manipulating files
  - Compiling and executing C programs
- Getting connected with SSH
  - Setting up SSH environment for password-less login
  - Using SSH for remote command invocation
  - Moving files between machines
- Condor
  - Working with ClassAds
  - Monitoring the queue
  - Using DAGMAN


Decoupled Work Queue Model

- Concurrent disjoint tasks
- Parametric Studies
  - SPMD (single program multiple data)
- Very coarse grained
- Processor farms and clusters
- Example software package: Condor
- This lecture covers this model of parallelism


Topics

- Condor components


ClassAds

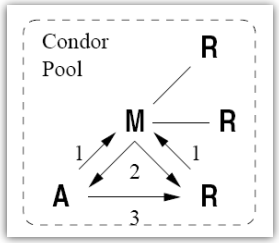
Job ClassAd	Machine ClassAd
[	[
MyType = "Job"	**MyType** = "Machine"
TargetType = "Machine"	**TargetType** = "Job"
Requirements =	**Machine** = "nostos.cs.wisc.edu"
((other.Arch=="INTEL" &&	**Requirements** =
other.OpSys=="LINUX")	(LoadAvg <= 0.300000) &&
&& other.Disk > my.DiskUsage)	(KeyboardIdle > (15 * 60))
Rank = (Memory * 10000) + KFlops	**Rank** = other.Department==self.Department
Cmd = "/home/tannenba/bin/sim-exe"	**Arch** = "INTEL"
Department = "CompSci"	**OpSys** = "LINUX"
Owner = "tannenba"	**Disk** = 3076076
DiskUsage = 6000	]
]	

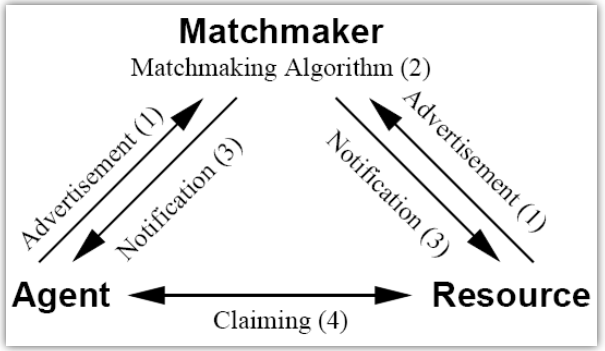
Figure 12. **Two Sample ClassAds from Condor.**


MatchMaker

Condor MatchMaker

- MatchMaker, a crucial part of the Condor architecture, uses the job description classAd provided by the user and matches the Job to the best resource based on the Machine description classAd
- MatchMaking in Condor is performed in 4 steps :
  1. Job Agent (**A**) and resources (**R**) advertise themselves.
  2. Matchmaker (**M**) processes the known classAds and generates pairs that best match resources and jobs
  3. Matchmaker informs each party of the job-resource pair of their prospective match.
  4. The Job agent and resource establish connection for further processing. (Matchmaker plays no role in this step, thus ensuring separation between selection of resources and subsequent activities)



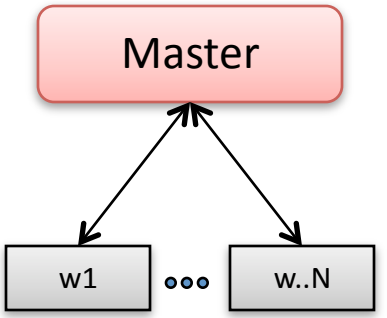


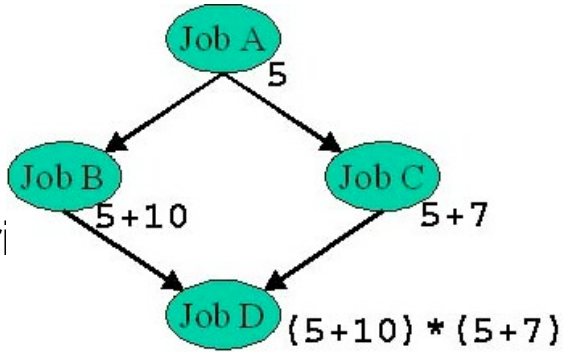
Src : Douglas Thain, Todd Tannenbaum, and Miron Livny, "Distributed Computing in Practice: The Condor Experience" *Concurrency and Computation: Practice and Experience*, Vol. 17, No. 2-4, pages 323-356, February-April, 2005. http://www.cs.wisc.edu/condor/doc/condor-practice.pdf

Problem Solvers

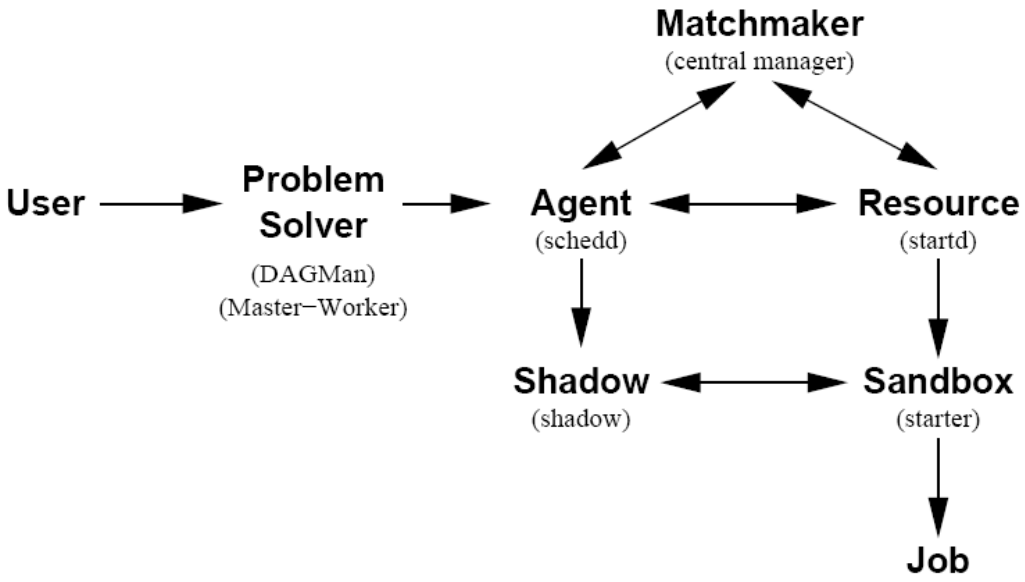
Condor Problem Solvers

- **Master-Worker (MW)** is a problem solving system that is useful for solving a coarse grained problem of indeterminate size such as parameter sweep etc.
- The MW Solver in Condor consists of 3 main components : work-list, a tracking module, and a steering module. The work-list keeps track of all pending work that master needs done. The tracking module monitors progress of work currently in progress on the worker nodes. The steering module directs computation based on results gathered and the pending work-list and communicates with the matchmaker to obtain additional worker processes.
- **DAGMan** is used to execute multiple jobs that have dependencies represented as a Directed Acyclic Graph where the nodes correspond to the jobs and edges correspond to the dependencies between the jobs. DAGMan provides various functionalities for job monitori and fault tolerance via creation of rescue DAGs.





Management Middleware



Indepth Coverage :

http://www.cs.wisc.edu/condor/publications.html

Recommended Reading :

Douglas Thain, Todd Tannenbaum, and Miron Livny, "Distributed Computing in Practice: The Condor Experience" *Concurrency and Computation: Practice and Experience*, Vol. 17, No. 2-4, pages 323-356, February-April, 2005. [PDF]

Todd Tannenbaum, Derek Wright, Karen Miller, and Miron Livny, "Condor - A Distributed Job Scheduler", in Thomas Sterling, editor, *Beowulf Cluster Computing with Linux*, The MIT Press, 2002. ISBN: 0-262-69274-0 [Postscript] [PDF]

Core components of Condor

- **condor_master:** This program runs constantly and ensures that all other parts of Condor are running. If they hang or crash, it restarts them.
- **condor_collector:** This program is part of the Condor central manager. It collects information about all computers in the pool as well as which users want to run jobs. It is what normally responds to the condor_status command. It's not running on your computer, but on the main Condor pool host (Celeritas head node).
- **condor_negotiator:** This program is part of the Condor central manager. It decides what jobs should be run where. It's not running on your computer, but on on the main Condor pool host (Celeritas head node).
- **condor_startd:** If this program is running, it allows jobs to be started up on this computer--that is, hal is an "execute machine". This advertises hal to the central manager (more on that later) so that it knows about this computer. It will start up the jobs that run.
- **condor_schedd** If this program is running, it allows jobs to be submitted from this computer--that is, hal is a "submit machine". This will advertise jobs to the central manager so that it knows about them. It will contact a condor_startd on other execute machines for each job that needs to be started.
- **condor_shadow** For each job that has been submitted from this computer, there is one condor_shadow running. It will watch over the job as it runs remotely. In some cases it will provide some assistance You may or may not see any condor_shadow processes running, depending on what is happening on the computer when you try it out.

Source : http://www.cs.wisc.edu/condor/tutorials/cw2005-condor/intro.html


Topics

- Condor: Useful commands


Condor: A Walkthrough of Condor commands

condor_submit: submits a job to condor pool

condor_submit_dag: submits a DAGMAN job

condor_q: provides current job queue

    -analyze: provides additional job information

condor_rm: deletes a job from the job queue

condor_status: provides current pool status

    -l: lists machine resources


What machines are available? (condor_status)

condor_status queries resource information sources and provides the current status of the condor pool of resources

- Some common condor_status command line options :
  - -help : displays usage information
  - -avail : queries condor_startd ads and prints information about available resources
  - -claimed : queries condor_startd ads and prints information about claimed resources
  - -ckptsrvr : queries condor_ckpt_server ads and display checkpoint server attributes
  - -pool *hostname* queries the specified central manager (by default queries $COLLECTOR_HOST)
  - -verbose : displays entire classads
  - For more options and what they do run *"condor_status –help"*


Example: *condor_status*

```
[cdekate@celeritas ~]$ condor_status

Name               OpSys       Arch   State      Activity   LoadAv Mem   ActvtyTime

vm1@compute-0 LINUX       X86_64 Unclaimed  Idle       0.000  1964  3+13:42:23
vm2@compute-0 LINUX       X86_64 Unclaimed  Idle       0.000  1964  3+13:42:24
vm3@compute-0 LINUX       X86_64 Unclaimed  Idle       0.010  1964  0+00:45:06
vm4@compute-0 LINUX       X86_64 Owner      Idle       1.000  1964  0+00:00:07
vm1@compute-0 LINUX       X86_64 Unclaimed  Idle       0.000  1964  3+13:42:25
vm2@compute-0 LINUX       X86_64 Unclaimed  Idle       0.000  1964  1+09:05:58
vm3@compute-0 LINUX       X86_64 Unclaimed  Idle       0.000  1964  3+13:37:27
vm4@compute-0 LINUX       X86_64 Unclaimed  Idle       0.000  1964  0+00:05:07
…
…
vm3@compute-0 LINUX       X86_64 Unclaimed  Idle       0.000  1964  3+13:42:33
vm4@compute-0 LINUX       X86_64 Unclaimed  Idle       0.000  1964  3+13:42:34

                     Total Owner Claimed Unclaimed Matched Preempting Backfill

        X86_64/LINUX    32     3       0        29       0          0        0

               Total    32     3       0        29       0          0        0
```


CSC Department of Computer Science Louisiana State University

What jobs are currently in the queue? *condor_q*

- *condor_q* provides a list of job that have been submitted to the Condor pool
- Provides details about jobs including which cluster the job is running on, owner of the job, memory consumption, the name of the executable being processed, current state of the job, when the job was submitted and how long has the job been running.

- Some common condor_q command line options :
  - -global : queries all job queues in the pool
  - -name : queries based on the *schedd* name provides a queue listing of the named *schedd*
  - -claimed : queries condor_startd ads and prints information about claimed resources
  - -goodput : displays job *goodput* statistics ("*goodput* is the allocation time when an application uses a remote workstation to make forward progress." – Condor Manual)
  - -cputime : displays the remote CPU time accumulated by the job to date...
  - For more options run : "*condor_q –help*"

Example: *condor_q*

```
[cdekate@celeritas ~]$ condor_q

-- Submitter: celeritas.cct.lsu.edu : <130.39.128.68:40472> : celeritas.cct.lsu.edu
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  30.0   cdekate         1/23 07:52   0+00:01:13 R  0   9.8  fib 100
  30.1   cdekate         1/23 07:52   0+00:01:09 R  0   9.8  fib 100
  30.2   cdekate         1/23 07:52   0+00:01:07 R  0   9.8  fib 100
  30.3   cdekate         1/23 07:52   0+00:01:11 R  0   9.8  fib 100
  30.4   cdekate         1/23 07:52   0+00:01:05 R  0   9.8  fib 100

5 jobs; 0 idle, 5 running, 0 held
[cdekate@celeritas ~]$
```


Creating Condor submit file (job as a ClassAds)

- Condor submit file contains key-value pairs that help describe the application to condor.
- Condor submit files are job ClassAds.
- Some of the common descriptions found in the job ClassAds are :

executable = (path to the executable to run on Condor)
input = (standard input provided as a file)
output = (standard output stored in a file
log = (output to log file)
arguments = (arguments to be supplied to the)
queue

How to submit your Job? condor_submit

- Create a job classAd (condor submit file) that contains Condor keywords and user configured values for the keywords.
- Submit the job classAd using "condor_submit"
- Example :

  condor_submit matrix.submit

- condor_submit –h provides additional flags

```
[cdekate@celeritas NPB3.2-MPI]$ condor_submit -h
Usage: condor_submit [options] [cmdfile]
        Valid options:
        -verbose                verbose output
        -name <name>            submit to the specified schedd
        -remote <name>          submit to the specified remote schedd
                                (implies -spool)
        -append <line>          add line to submit file before processing
                                (overrides submit file; multiple -a lines ok)
        -disable                disable file permission checks
        -spool                  spool all files to the schedd
        -password <password>    specify password to MyProxy server
        -pool <host>            Use host as the central manager to query
        If [cmdfile] is omitted, input is read from stdin
```

condor_submit: Example

```
[cdekate@celeritas ~]$ condor_submit fib.submit
Submitting job(s).....
Logging submit event(s).....
5 job(s) submitted to cluster 35.
[cdekate@celeritas ~]$ condor_q

-- Submitter: celeritas.cct.lsu.edu : <130.39.128.68:51675> :
 celeritas.cct.lsu.edu
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  35.0   cdekate         1/24 15:06   0+00:00:00 I  0   9.8  fib 10
  35.1   cdekate         1/24 15:06   0+00:00:00 I  0   9.8  fib 15
  35.2   cdekate         1/24 15:06   0+00:00:00 I  0   9.8  fib 20
  35.3   cdekate         1/24 15:06   0+00:00:00 I  0   9.8  fib 25
  35.4   cdekate         1/24 15:06   0+00:00:00 I  0   9.8  fib 30

5 jobs; 5 idle, 0 running, 0 held
[cdekate@celeritas ~]$
```


CSC Department of Computer Science Louisiana State University

How to delete a submitted job? condor_rm

- condor_rm : Deletes one or more jobs from Condor job pool. If a particular Condor pool is specified as one of the arguments then the *condor_schedd* matching the specification is contacted for job deletion, else the local *condor_schedd* is contacted.

```
[cdekate@celeritas ~]$ condor_rm -h
Usage: condor_rm [options] [constraints]
 where [options] is zero or more of:
  -help                Display this message and exit
  -version             Display version information and exit
  -name schedd_name    Connect to the given schedd
  -pool hostname       Use the given central manager to find daemons
  -addr <ip:port>      Connect directly to the given "sinful string"
  -reason reason       Use the given RemoveReason
  -forcex              Force the immediate local removal of jobs in the X state
                       (only affects jobs already being removed)
 and where [constraints] is one or more of:
  cluster.proc         Remove the given job
  cluster              Remove the given cluster of jobs
  user                 Remove all jobs owned by user
  -constraint expr     Remove all jobs matching the boolean expression
  -all                 Remove all jobs (cannot be used with other constraints)
[cdekate@celeritas ~]$
```


condor_rm: Example

```
[cdekate@celeritas ~]$ condor_q
-- Submitter: celeritas.cct.lsu.edu : <130.39.128.68:51675> :
   celeritas.cct.lsu.edu
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  41.0   cdekate         1/24 15:43   0+00:00:03 R  0   9.8  fib 100
  41.1   cdekate         1/24 15:43   0+00:00:01 R  0   9.8  fib 150
  41.2   cdekate         1/24 15:43   0+00:00:00 R  0   9.8  fib 200
  41.3   cdekate         1/24 15:43   0+00:00:00 R  0   9.8  fib 250
  41.4   cdekate         1/24 15:43   0+00:00:00 R  0   9.8  fib 300

5 jobs; 0 idle, 5 running, 0 held
[cdekate@celeritas ~]$ condor_rm 41.4
Job 41.4 marked for removal
[cdekate@celeritas ~]$ condor_rm 41
Cluster 41 has been marked for removal.
[cdekate@celeritas ~]$
```


CSC Department of Computer Science Louisiana State University