National e-Science Centre

Tutorial 4: Introduction to Condor

Main Points

National e-Science Centre

- Introduction
- Pool Architecture
- Job Preparation
- Job Submission
- Job Monitoring
- Example



Condor
High Throughput Computing

National e-Science Centre

- **Background**
  - **University of Wisconsin-Madison, USA**
    - Team led by Prof. Miron Livny
  - **The Remote-Unix (RU) project – 1986**
    - First version of Condor Resource Management System
    - Focussed on load balancing in distributed systems
  - **Condor Project – 1988**
    - Shift towards Distributively Owned computing environments (i.e. resource owners have full control over access)
  - **FREE software, but NOT open-source**
    - Binaries for all platforms available from web site
    - http://www.cs.wisc.edu/condor



Condor
High Throughput Computing

National e-Science Centre

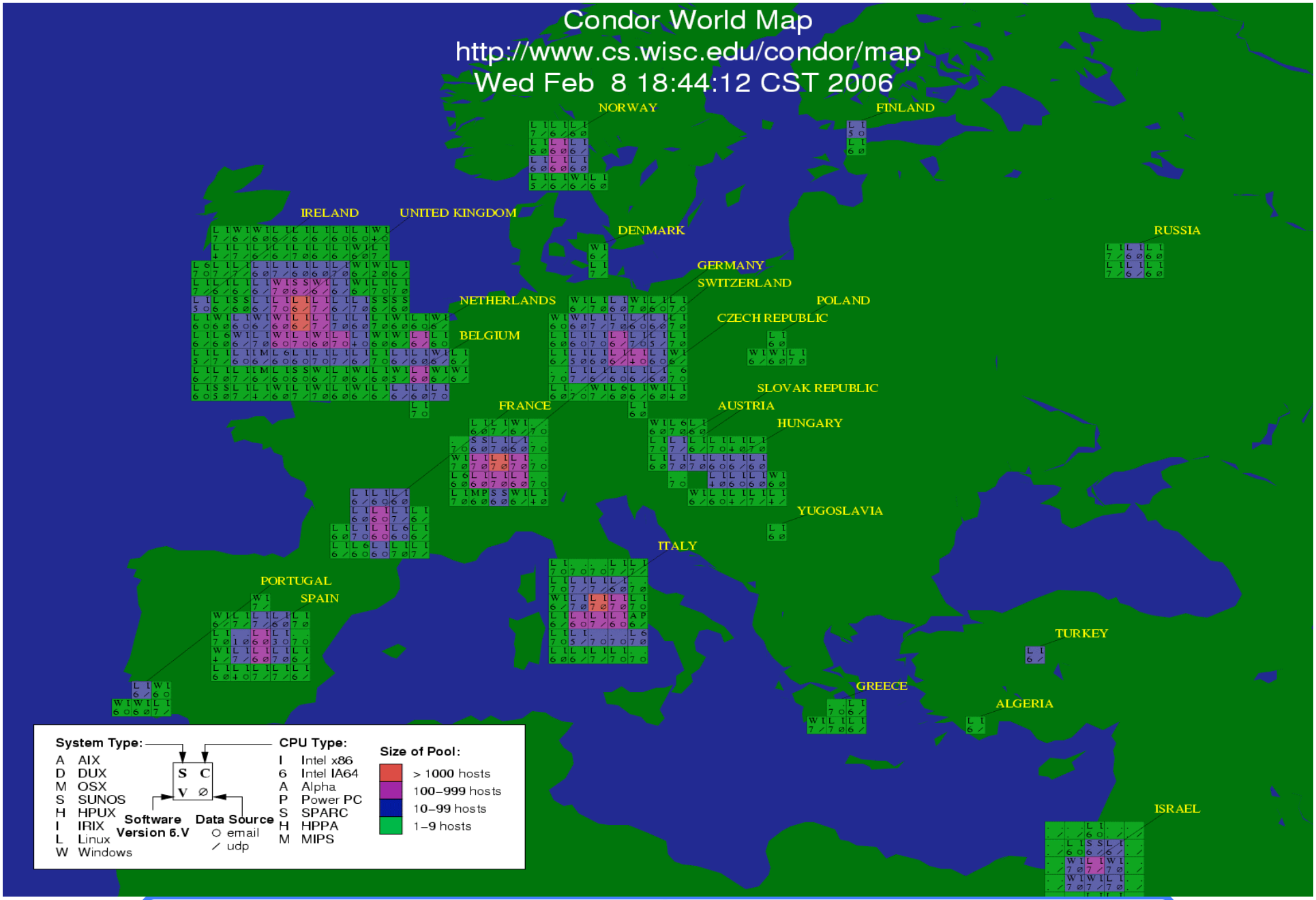
- **What is Condor?**
  - **A cycle-harvesting system**
    - Carries out computation while the machine is sitting idle
  - **A batch scheduling system**
    - Provides submission and processing of batch jobs
  - **A workload management system**
    - Allows for user to prioritise jobs
  - **A production level software package for High Throughput Computing (HTC)**
    - Can be installed on desktop pcs, workstations, clusters, laptops…

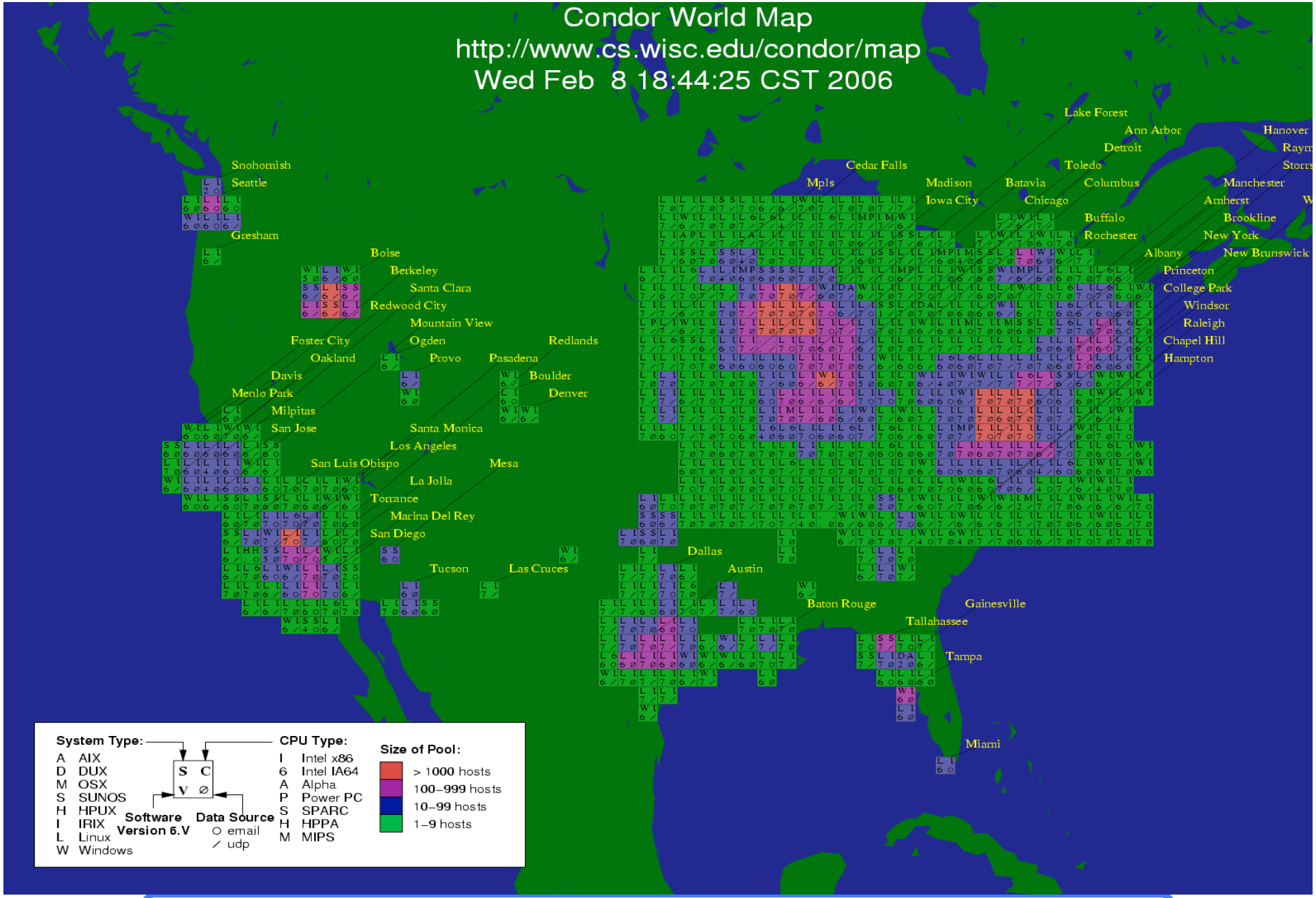


Condor
High Throughput Computing

National e-Science Centre

- **Why use Condor?**
  - **Easy support for large node clusters**
  - **Harnesses spare CPU time from idle workstations**
    - Fully configurable behaviour
  - **Powerful ClassAd job description mechanism**
    - Complex job requirements may be expressed
  - **Works with or without a shared file system**
  - **Transparent migration and checkpointing**
    - If machine detects a key press, can checkpoint your job and move it to another resource
  - **Condor Pools may “flock” together**


Condor World Map
http://www.cs.wisc.edu/condor/map
Wed Feb 8 18:44:12 CST 2006
NORWAY
FINLAND
IRELAND
UNITED KINGDOM
DENMARK
RUSSIA
GERMANY
SWITZERLAND
NETHERLANDS
POLAND
BELGIUM
CZECH REPUBLIC
SLOVAK REPUBLIC
FRANCE
AUSTRIA
HUNGARY
YUGOSLAVIA
ITALY
PORTUGAL
SPAIN
TURKEY
GREECE
ALGERIA
ISRAEL
System Type:
A AIX
D DUX
M OSX
S SUNOS
H HPUX
I IRIX
L Linux
W Windows
Software Version 6.V
Data Source
o email
/ udp
CPU Type:
I Intel x86
6 Intel IA64
A Alpha
P Power PC
S SPARC
H HPPA
M MIPS
Size of Pool:
> 1000 hosts
100–999 hosts
10–99 hosts
1–9 hosts



UNIVERSITY of GLASGOW

e-Science dti

National e-Science Centre

Pool Machines

- **Pool machines assume various roles:**
  - **Central Manager**
    - One machine in the pool which is responsible for coordinating all the pool activity (matching jobs, monitoring job status)
  - **Submit Machine**
    - One or more machines in the pool which allow users to submit jobs to the pool
  - **Worker Node**
    - One or more machines in the pool which jobs can run on
- **These roles are implemented by condor daemon programs…**

Condor Daemons

National e-Science Centre

- **There are 5 main Condor daemons:**
  - **Master**
    - All resources
  - **Schedd**
    - All submit machines
  - **Startd**
    - All worker nodes
  - **Collector**
    - Central Manager
  - **Negotiator**
    - Central Manager

Condor master

- **Runs on all machines regardless of role**
  - **Keeps all the other master daemons in the pool running**
  - **Spawns all other daemons required on its machine**
    - Plus can check for updates if required
  - **e-mails the condor pool administrator if a node crashes**
  - **Supports global and local admin commands**
    - start, stop, reconfig, restart
  - **Runs under low-privilege “condor” user**
    - May be installed as root, but still runs under condor user

Condor startd

National e-Science Centre

- **Represents a resource in the pool**
  - **Runs on all execute machines**
  - **It advertises resource attributes to the rest of the condor pool**
  - **Enforces local resource policy on how jobs are to be started, killed, suspended etc..**
  - **Spawns another local daemon when a job is ready to run**
    - condor_starter
    - Sets up execution environment, monitors job, contacts submit machine on completion

Condor schedd

National e-Science Centre

- **Deals with job submission on a resource**
  - **Runs on all submit machines**
  - **Manages and Advertises the number of jobs in its queue and claims allocated resources to fulfil its requests**
  - **schedd tools include `condor_submit`, `condor_q`, `condor_rm`**
  - **Spawns condor_shadow daemon once a resource is chosen**
    - Acts as resource manager for the request
    - System calls on a remote resource are sent back to the shadow and performed there

Condor collector

- **Collects information about the status of the pool**
  - Runs only on Central Manager
  - Collector receives periodic “ClassAd” updates from all daemons (except negotiator) describing the state of each daemon/resource/request
  - Daemons may also query the collector to find out what address to use for transactions between machines
  - `condor_status` queries the collector for information

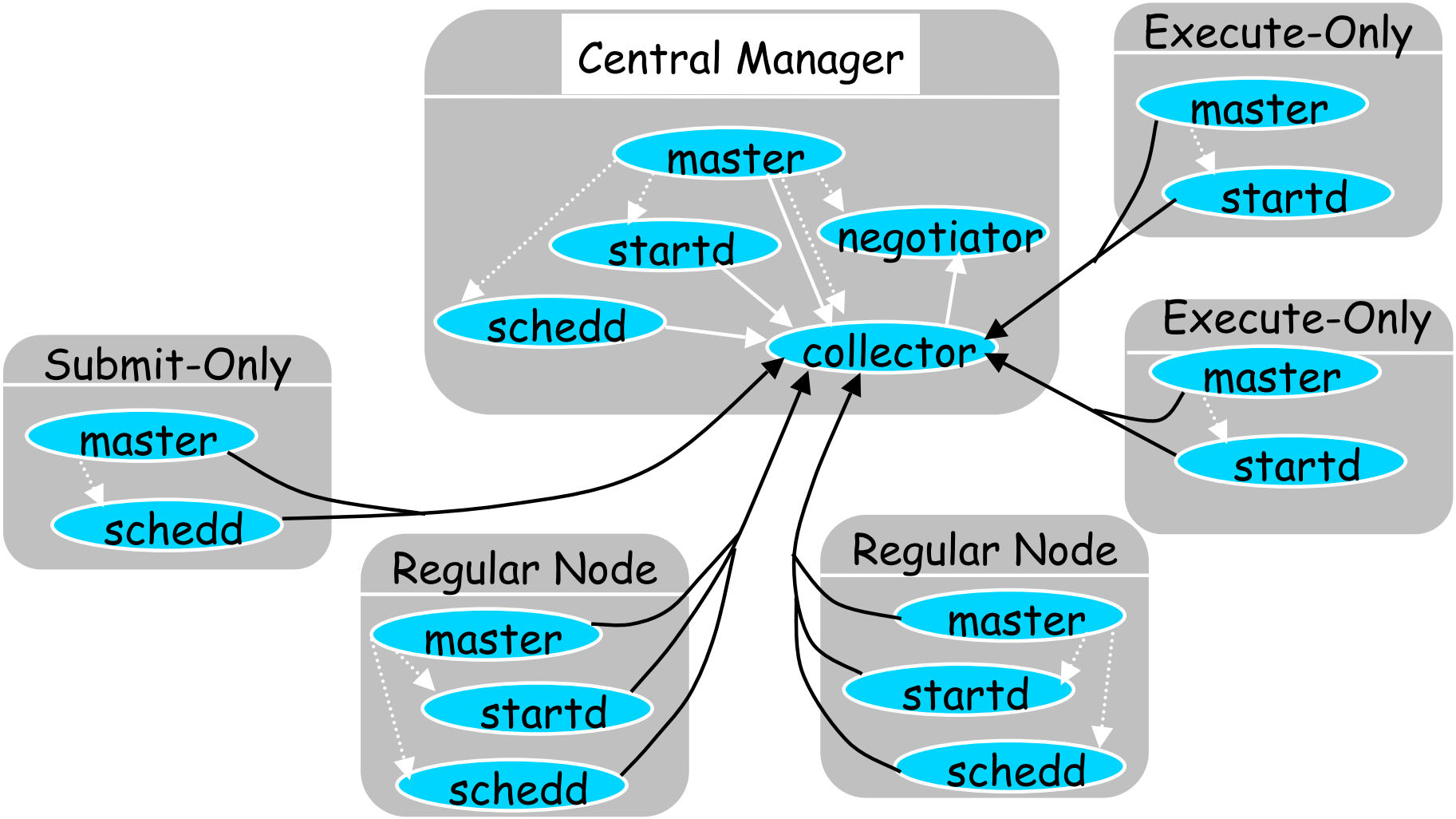
▶ We will look at this information shortly in “ClassAds”

Condor negotiator

National e-Science Centre

- **Handles matching of jobs to resources**
  - **Negotiator initiates a periodic query of the collector for information about the state of the pool**
  - **Contacts each schedd that has waiting requests and attempts to match job properties with appropriate machines.**
  - **Enforces user priorities**
    - e.g. The more resources a user claims the less priority he has for job submission
    - Condor priority of 0.5 is best, with priority decreasing as the number goes up

A Typical Condor Pool



ClassAds

- **Condor is a ClassAd match-maker**
  - **Think of a user as a buyer, resource as a seller**
  - **A resource advertises its attributes (product)**
    - RAM, CPU speed, load average…
  - **A user advertises their job requirements**
    - Minimum CPU speed, high RAM requirement, priority
    - Or.. “I’ll take anything…”
  - **The Condor *collector* continuously reads the machine ClassAds and the job ClassAds and the *negotiator* matches jobs with suitable resources.**

condor_status

- `condor_status` can be used to query machine ClassAds

```
Name               OpSys      Arch   State     Activity   LoadAv Mem   ActvtyTime

labpc-1.nesc.      LINUX      INTEL  Unclaimed Idle       0.000   494  0+02:45:04
labpc-10.nesc      LINUX      INTEL  Unclaimed Idle       0.000   494  0+02:45:04
labpc-11.nesc      LINUX      INTEL  Unclaimed Idle       0.000   494  0+02:45:04
labpc-12.nesc      LINUX      INTEL  Unclaimed Idle       0.000   494  0+02:45:04
labpc-13.nesc      LINUX      INTEL  Unclaimed Idle       0.000   494  0+02:40:04
labpc-14.nesc      LINUX      INTEL  Unclaimed Idle       0.000   494  0+02:40:04

                     Machines Owner Claimed Unclaimed Matched Preempting

        INTEL/LINUX         6     0       0         6       0          0

              Total         6     0       0         6       0          0
```

condor_status

- `condor_status -l`
  - Displays the whole ClassAd for every node
- `condor_status -l labpc-1.nesc.gla.ac.uk`
  - Displays the whole ClassAd for the specified machine.
- `condor_status -available`
  - Show only machines which are currently able to run jobs
- `condor_status -run`
  - Show machines running jobs

Job Submission

National e-Science Centre

- **Job submission checklist**
  - Prepare the code and environment
  - Choose a “Universe” to run in
  - Create a condor submit file
  - Submit the job to Condor
- **Job Monitoring**
  - View the condor q
  - Set priorities?

Code Preparation Check

National e-Science Centre

- **Condor jobs must be able to**
  - **Run as a background batch job**
    - Must be able to run unattended
  - **Handle any interactive I/O**
    - Jobs cannot do any keyboard input or console output
    - But you can provide files that contain any necessary keystrokes etc
  - **Check that your job runs OK with any interactive input/output files before attempting Condor submission**

Condor Universes

National e-Science Centre

- A Universe is a runtime execution environment
- Universes available include
  - Standard
  - Vanilla
  - Java (JVM)
  - Parallel Virtual Machine (PVM)
  - Message Passing (MPI)
  - Globus
- The first three are of interest to us…

The Standard Universe

National e-Science Centre

- Supports remote system calls, checkpointing and job migration
- Programs require relinking with the Condor libraries
  - i.e. You need the source code
- Programs are relinked using condor_compile
  - `condor_compile cc main.o -out a.out`

The Vanilla Universe

National e-Science Centre

- Programs that cannot be re-linked need to run under the Vanilla universe (e.g. shell scripts)
  - Checkpointing and migration cannot be implemented without Condor re-linking.
- Depends highly on a shared filesystem (or shared files)
  - Two approaches:
  - Either implement with NFS, AFS
  - Or copy the required input/output files to every machine that could be used to run the job.

The Java Universe

National e-Science Centre

- Runs jobs on any machine with a Java Virtual Machine
- Condor will locate the binary and set the classpath
  - Will have been configured when Condor is installed on the resource
- Is independent of JVM location, version and owner

Creating a condor submit file

National e-Science Centre

- **Once the code is checked and a universe chosen, Condor requires some final information before submission**
  - **Job submitted with condor_submit command**
  - `$ condor_submit <script_name>`
  - **`condor_submit` takes a submit description file as an argument**
    - Contains information about executable, directories, command line arguments, and keywords and variables that direct job queueing…

Example submit script

National e-Science Centre

```
# Condor submit file example
universe                    = vanilla
executable                  = sh_loop
output                      = sh_loop.out
error                       = sh_loop.err
log                         = sh_loop.log
arguments                   = 60
should_transfer_files       = IF_NEEDED
when_to_transfer_output     = ON_EXIT
queue
```

- Transfer files if we don't use a shared filesystem and transmit files back to submit machine only when job exits itself

condor_q

- Displays information about the progress of a job
  - Contacts the local schedd by default
  - idle job:

```
-- Submitter: voyager.nesc.gla.ac.uk : <130.209.58.187:9671> :
voyager.nesc.gla.ac.uk
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
 996.0   tomcat          2/3  13:58   0+00:00:00 I  0   0.0  sh_loop 60
```

  - running job:

```
-- Submitter: voyager.nesc.gla.ac.uk : <130.209.58.187:9671> :
voyager.nesc.gla.ac.uk
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
 996.0   tomcat          2/3  13:58   0+00:00:27 R  0   0.0  sh_loop 60

1 jobs; 0 idle, 1 running, 0 held
```

Examples

National e-Science Centre

-
- `/usr/local/condor-6.7.13/examples`
- 1. Try the sh_loop submission
- 2. Try the loop.c example