

Using Condor

Alain Roy
Condor Project
NMI Nanohub Project

- This presentation assumes you've seen the Introduction to Condor
- We'll do a brief review...

I need a Mac!

$$E = mc^2$$

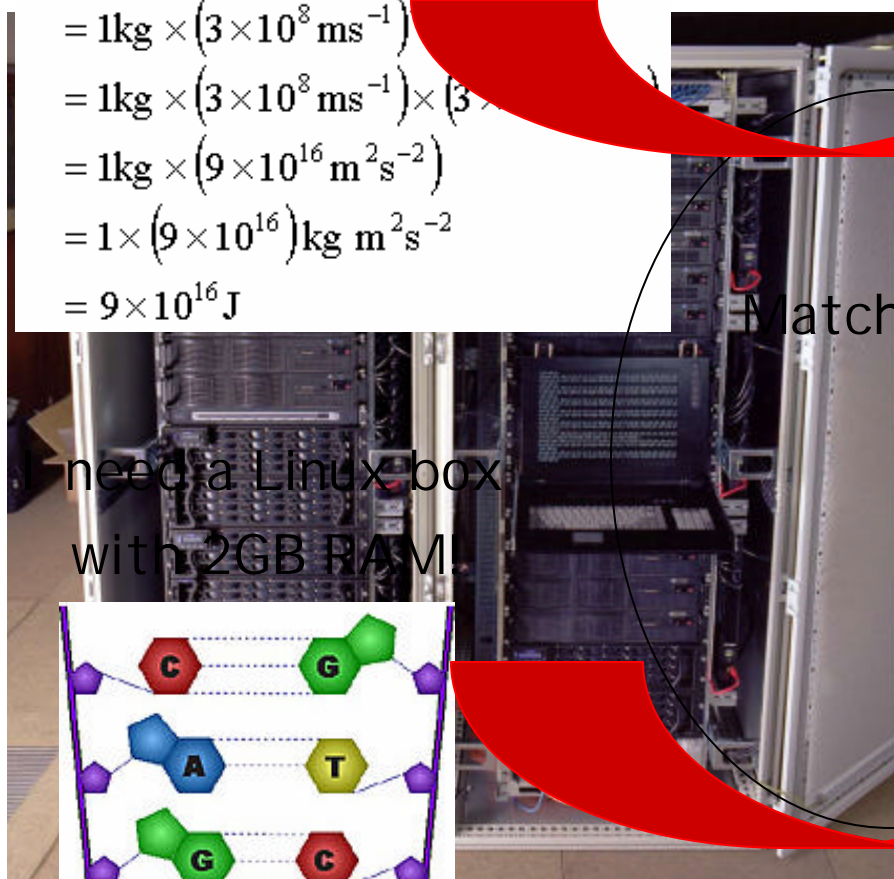
$$= 1\text{kg} \times (3 \times 10^8 \text{ms}^{-1})^2$$

$$= 1\text{kg} \times (3 \times 10^8 \text{ms}^{-1}) \times (3 \times 10^8 \text{ms}^{-1})$$

$$= 1\text{kg} \times (9 \times 10^{16} \text{m}^2 \text{s}^{-2})$$

$$= 1 \times (9 \times 10^{16}) \text{kg m}^2 \text{s}^{-2}$$

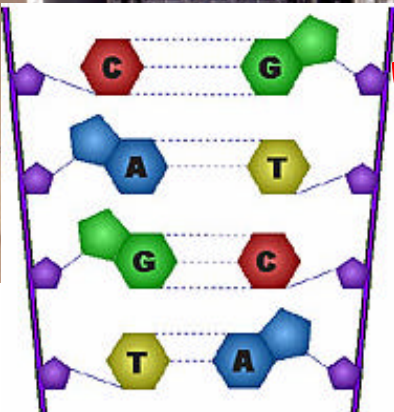
$$= 9 \times 10^{16} \text{J}$$



Matchm



I need a Linux box
with 2GB RAM!



- **Cluster**: A dedicated set of computers not for interactive use
- **Pool**: A collection of computers used by Condor
 - May be dedicated
 - May be interactive

- Matchmaking is fundamental to Condor
- Matchmaking is two-way
 - Job describes what it requires:
I need Linux && 2 GB of RAM
 - Machine describes what it requires:
I will only run jobs at night
- Matchmaking allows preferences
 - I **need** Linux, and I **prefer** machines with more memory but will run on any machine you provide me

- ClassAds state facts
 - My job's executable is analysis.exe
 - My machine's load average is 5.6
- ClassAds state preferences
 - I require a computer with Linux

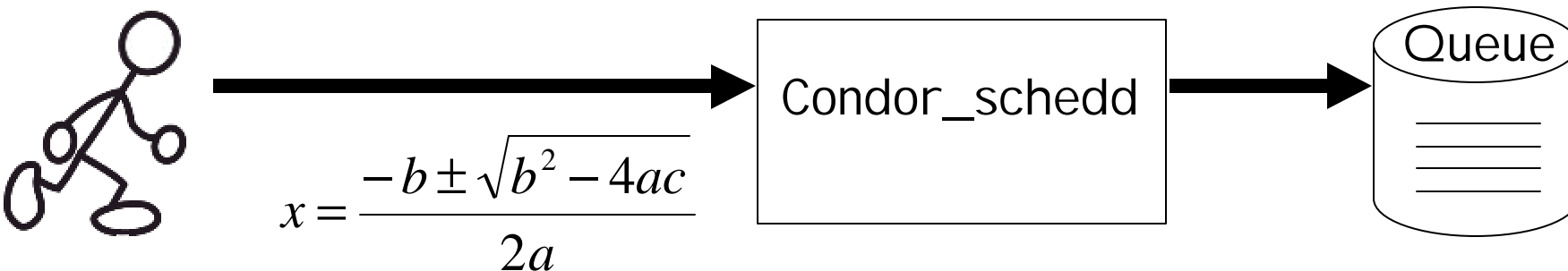
•ClassAds are:

- semi-structured
- user-extensible
- schema-free
- Attribute = Expression

Example:

```
MyType           = "Job"
TargetType       = "Machine"
ClusterId        = 1377
Owner            = "roy"
Cmd              = "analysis.exe"
Requirements     =
    (Arch == "INTEL")
&& (OpSys == "LINUX")
&& (Disk >= DiskUsage)
&& ((Memory * 1024) >= ImageSize)
...
```

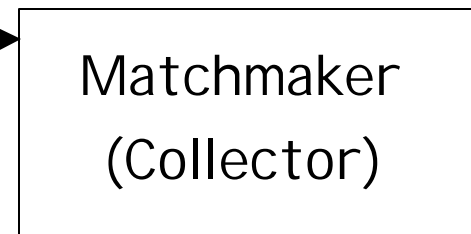
- Users submit jobs from a computer
 - Jobs described as a ClassAd
 - Each submission computer has a queue
 - Queues are **not** centralized
 - Submission computer watches over queue
 - Can have multiple submission computers
 - Submission handled by condor_schedd



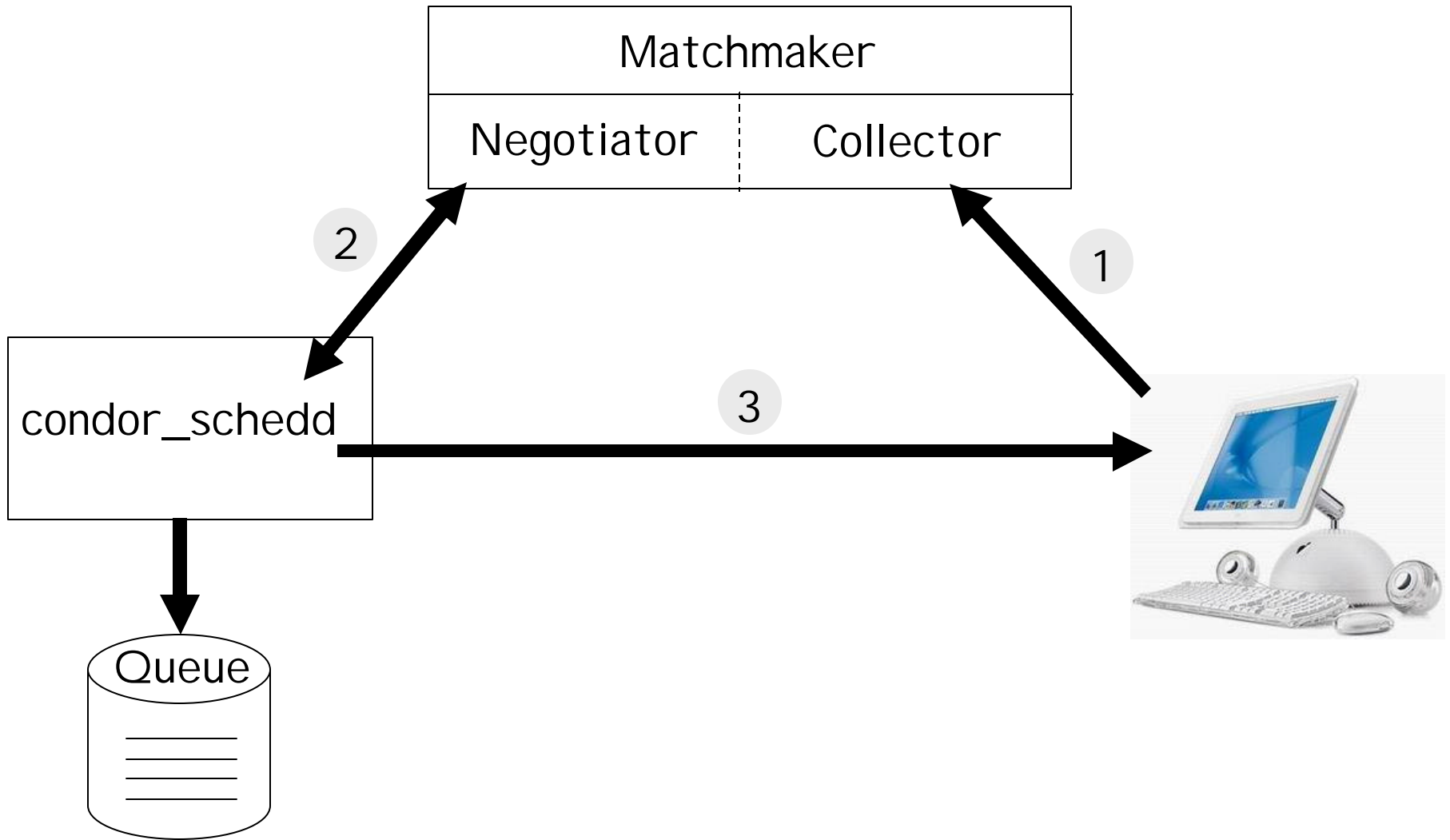
- Machine owners describe computers
 - Configuration file extends ClassAd
 - ClassAd has dynamic features
 - ✓ Load Average
 - ✓ Free Memory
 - ✓ ...
 - ClassAds are sent to Matchmaker



```
ClassAd
Type = "Machine"
Requirements = "..."
```



- Negotiator collects list of computers
- Negotiator contacts each schedd
 - What jobs do you have to run?
- Negotiator compares each job to each computer
 - Evaluate requirements of job & machine
 - Evaluate in context of both ClassAds
 - If both evaluate to true, there is a match
- Upon match, schedd contacts execution computer



- Available as a free download from
<http://www.cs.wisc.edu/condor>
- Available for many UNIX platforms:
 - Linux, Solaris, HPUX, IRIX, Tru64...
- Available for Mac OS X
- Available for Windows

- Naming scheme similar to the Linux Kernel...
- Major.**minor**.release
 - Stable: Minor is even (a.**b**.c)
 - ✓ Examples: 6.**4**.3, 6.**6**.8, 6.**6**.9
 - ✓ Very stable, mostly bug fixes
 - Developer: Minor is odd (a.**b**.c)
 - ✓ New features, may have some bugs
 - ✓ Examples: 6.**5**.5, 6.**7**.5, 6.**7**.6

- Condor:

- on your own workstation
- no root access required
- no system administrator intervention needed

Personal Condor?!

What's the benefit of a Condor Pool
with just one user and one machine?

Your Personal Condor will ...

- ... keep an eye on your jobs and will keep you posted on their progress
- ... implement your policy on the execution order of the jobs
- ... keep a log of your job activities
- ... add fault tolerance to your jobs
- ... implement your policy on when the jobs can run on your workstation

- When a Personal Condor pool works for you...
 - Convince your co-workers to add their computers to the pool
 - Add dedicated hardware to the pool

1. Choose a Universe for your job
2. Make your job batch-ready
3. Create a *submit description* file
4. Run *condor_submit*

- There are many choices
 - Vanilla: any old job
 - Standard: checkpointing & remote I/O
 - Java: better for Java jobs
 - MPI: Run parallel MPI jobs
 - ...
- For now, we'll just consider vanilla

- Must be able to run in the background: no interactive input, windows, GUI, etc.
- Can still use `STDIN`, `STDOUT`, and `STDERR` (the keyboard and the screen), but files are used for these instead of the actual devices
- Organize data files

- A plain ASCII text file
- Condor does **not** care about file extensions
- Tells Condor about your job:
 - Which executable, universe, input, output and error files to use, command-line arguments, environment variables, any special requirements or preferences

```
# Simple condor_submit input file
# (Lines beginning with # are comments)
# NOTE: the words on the left side are not
#       case sensitive, but filenames are!
```

```
Universe      = vanilla
Executable    = analysis
Log           = my_job.log
Queue
```

- You give `condor_submit` the name of the submit file you have created:

```
condor_submit my_job.submit
```

- `condor_submit` parses the submit file, checks for errors, and creates a ClassAd that describes your job.

- `condor_submit` sends your job's ClassAd to the schedd
 - Manages the local job queue
 - Stores the job in the job queue
 - ✓ Atomic operation, two-phase commit
 - ✓ "Like money in the bank"
- View the queue with `condor_q`

```
% condor_submit my_job.submit
Submitting job(s).
1 job(s) submitted to cluster 1.

% condor_q

-- Submitter: perdita.cs.wisc.edu : <128.105.165.34:1027> :
ID          OWNER      SUBMITTED      RUN_TIME ST PRI  SIZE  CMD
  1.0       roy        7/6 06:52      0+00:00:00 I  0    0.0  analysis

1 jobs; 1 idle, 0 running, 0 held

%
```

- Condor sends you email about events
 - Turn it off: `Notification = Never`
 - Only on errors: `Notification = Error`
- Condor creates a log file (user log)
 - “The Life Story of a Job”
 - Shows all events in the life of a job
 - Always have a log file
 - Specified with: `Log = filename`

000 (0001.000.000) 05/25 19:10:03 Job submitted from host: <128.105.146.14:1816>

...

001 (0001.000.000) 05/25 19:12:17 Job executing on host: <128.105.146.14:1026>

...

005 (0001.000.000) 05/25 19:13:06 Job terminated.

(1) Normal termination (return value 0)

Usr 0 00:00:37, Sys 0 00:00:00 - Run Remote Usage

Usr 0 00:00:00, Sys 0 00:00:05 - Run Local Usage

Usr 0 00:00:37, Sys 0 00:00:00 - Total Remote Usage

Usr 0 00:00:00, Sys 0 00:00:05 - Total Local Usage

9624 - Run Bytes Sent By Job

7146159 - Run Bytes Received By Job

9624 - Total Bytes Sent By Job

7146159 - Total Bytes Received By Job

...

```
# Example condor_submit input file
```

```
Universe      = vanilla
```

```
Executable   = /home/roy/condor/my_job.condor
```

```
Log           = my_job.log
```

```
Input        = my_job.stdin
```

```
Output       = my_job.stdout
```

```
Error        = my_job.stderr
```

```
Arguments    = -arg1 -arg2
```

```
InitialDir   = /home/roy/condor/run_1
```

```
Queue
```

- If you want to remove a job from the Condor queue, you use `condor_rm`
- You can only remove jobs that you own (you can't run `condor_rm` on someone else's jobs unless you are root)
- You can give specific job ID's (cluster or cluster.proc), or you can remove all of your jobs with the "-a" option.
 - `condor_rm 21.1` -Removes a single job
 - `condor_rm 21` -Removes a whole cluster

```
% condor_status
```

| Name | OpSys | Arch | State | Activity | LoadAv | Mem | ActvtyTime |
|---------------|--------|-------|-----------|-----------|--------|-----|------------|
| haha.cs.wisc. | IRIX65 | SGI | Unclaimed | Idle | 0.198 | 192 | 0+00:00:04 |
| antipholus.cs | LINUX | INTEL | Unclaimed | Idle | 0.020 | 511 | 0+02:28:42 |
| coral.cs.wisc | LINUX | INTEL | Claimed | Busy | 0.990 | 511 | 0+01:27:21 |
| doc.cs.wisc.e | LINUX | INTEL | Unclaimed | Idle | 0.260 | 511 | 0+00:20:04 |
| dsonokwa.cs.w | LINUX | INTEL | Claimed | Busy | 0.810 | 511 | 0+00:01:45 |
| ferdinand.cs. | LINUX | INTEL | Claimed | Suspended | 1.130 | 511 | 0+00:00:55 |
| vm1@pinguino. | LINUX | INTEL | Unclaimed | Idle | 0.000 | 255 | 0+01:03:28 |
| vm2@pinguino. | LINUX | INTEL | Unclaimed | Idle | 0.190 | 255 | 0+01:03:29 |

How can my jobs access
their data files?



- Use shared filesystem if available
- No shared filesystem?
 - **Condor can transfer files**
 - ✓ Can automatically send back changed files
 - ✓ Atomic transfer of multiple files
 - ✓ Can be encrypted over the wire
 - Remote I/O Socket
 - Standard Universe can use remote system calls (more on this later)

- ShouldTransferFiles = YES
 - Always transfer files to execution site
- ShouldTransferFiles = NO
 - Rely on a shared filesystem
- ShouldTransferFiles = IF_NEEDED
 - Will automatically transfer the files if the submit and execute machine are not in the same FileSystemDomain

```
Universe      = vanilla
```

```
Executable   = my_job
```

```
Log           = my_job.log
```

```
ShouldTransferFiles = IF_NEEDED
```

```
Transfer_input_files = dataset$(Process), common.data
```

```
Transfer_output_files = TheAnswer.dat
```

```
Queue 600
```

Some of the machines in the
Pool do not have enough
memory or scratch disk space
to run my job!



- An expression (syntax similar to C or Java)
- Must evaluate to True for a match to be made

```
Universe      = vanilla
Executable    = my_job
Log           = my_job.log
InitialDir    = run_$(Process)
Requirements  = Memory >= 256 && Disk > 10000
Queue        600
```

- All matches which meet the requirements can be sorted by preference with a Rank expression.
- Higher the Rank, the better the match

```

Universe      = vanilla
Executable    = my_job
Log           = my_job.log
Arguments     = -arg1 -arg2
InitialDir    = run_$(Process)
Requirements  = Memory >= 256 && Disk > 10000
Rank          = (KFLOPS*10000) + Memory
Queue        600
    
```

- Condor can do much more than this...
 - In some cases, Condor can checkpoint your job
 - Condor can run parallel jobs (like MPI)
 - Condor can run collections of jobs in proper order
 - Condor can submit jobs to remote grid resources
 - Condor provides policy mechanisms such as:
 - ✓ What jobs are allowed on a computer?
 - ✓ When should Condor give up on a job?
 - And much, much more...

You can learn more:

www.cs.wisc.edu/condor