SIE's favourite pet:

Condor

(or how to easily run your programs in dozens of machines at a time)

Adrián Santos Marrero

Tutorial Outline

The Story of Frieda, the Scientist

- Using Condor to manage jobs
- Using Condor to manage resources
- Condor Architecture and Mechanisms

Stop me if you have any questions!

Meet Frieda.

She is a scientist. But she has a big problem.

Frieda's Application …

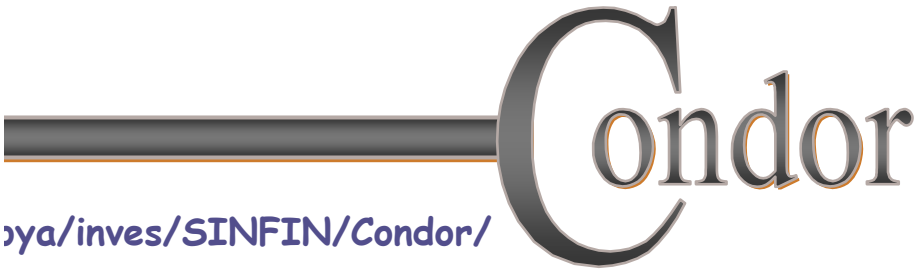
Run a Parameter Sweep of **F**(*x,y,z*) for 20 values of *x*, 10 values of *y* and 3 values of *z* (20*10*3 = 600 combinations)

- **F** takes on the average 6 hours to compute on a "typical" workstation (total = 3600 hours)
- **F** requires a "moderate" (128MB) amount of memory
- **F** performs "moderate" I/O - (*x,y,z*) is 5 MB and **F**(*x,y,z*) is 50 MB

I have 600
simulations to run.

Where can I get help?

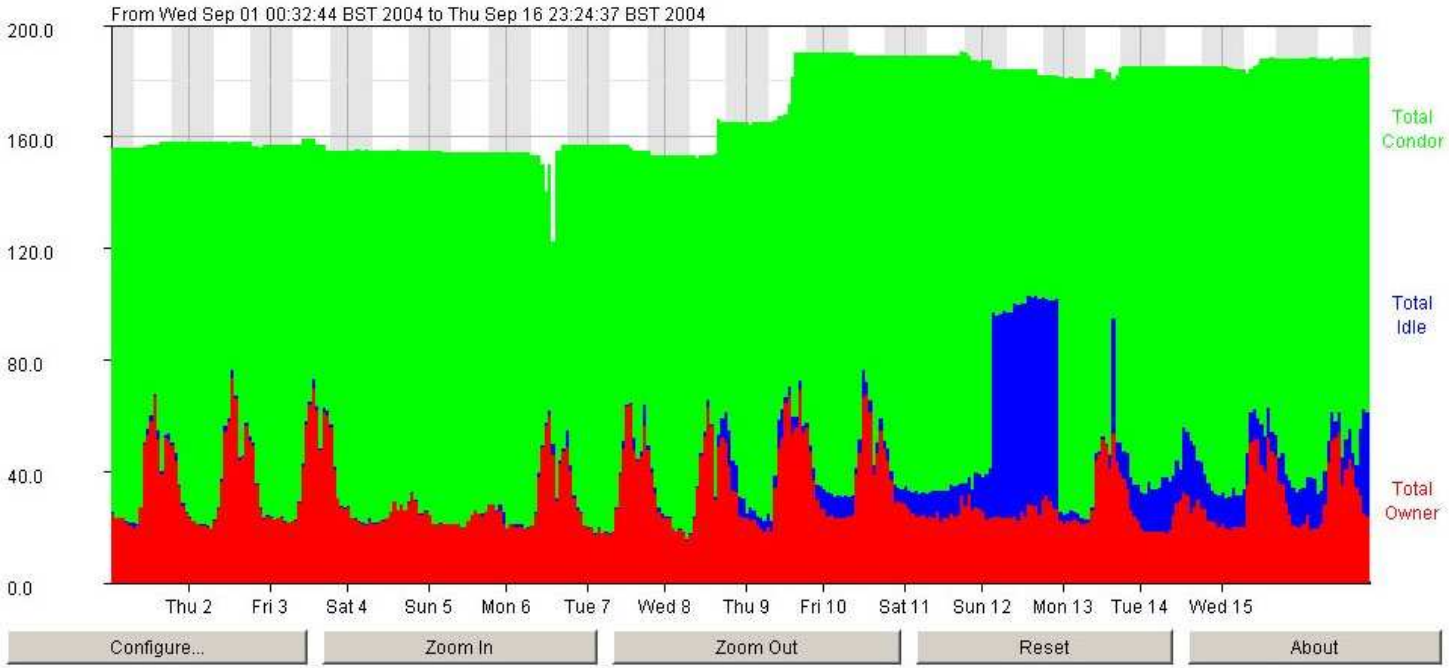
As if by magic, a genie appears from a lamp, and says, "Use Condor!"



What is Condor?

- Batch system
- Takes advantage off free machines at IAC.
- Results: It saves time to us and optimizes the available resources.

IAC-Condor Condor Pool Machine Statistics for Sep



[Graph Hints: The Y-axis is number of machines, the X-axis is time. When graph finishes updating, press "Configure.." to view different Architecture or State data. Also, you can use the mouse to draw a rectangle on the graph and then press "Zoom In". Press "Reset" to center/resize the data after Configure or when done zooming. Nighttime shows up on graph background as grey.]

Arch	Owner Average	Condor Average	Idle Average	Owner Peak	Condor Peak
Total	31.3 (18.6%)	129.9 (76.5%)	9.0 (4.9%)	76 (47%)	159 (90%)
SUN4u/SOLARIS29	11.5 (13.1%)	72.0 (81.5%)	5.0 (5.5%)	37 (43%)	82 (95%)
INTEL/LINUX	19.8 (24.7%)	57.9 (70.9%)	4.0 (4.4%)	44 (60%)	77 (86%)

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

What machines will Condor use?

- Condor is only going to use machines that:
  - Nobody is currently using (neither locally nor remotelly)
  - Do not have any applications running with a high CPU load
- Condor leaves a machine when this one becomes busy.

Testimonial

- Daniel Bramich has been using Condor at IAC for intensive calculations.
- Without Condor, he would have spent about 458 days to complete. But his simulations only take a couple of weeks using 30 of the fastest computers at IAC (limited by IDL's licences).

Getting Started: Submitting Jobs to Condor

- Creating a *submit description* file
- Choosing a "Universe" for your job
  - Just use VANILLA for now
- Make your job "batch-ready"
- Run *condor_submit* on your submit description file

Making your job batch-ready

- 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

Creating a Submit Description File

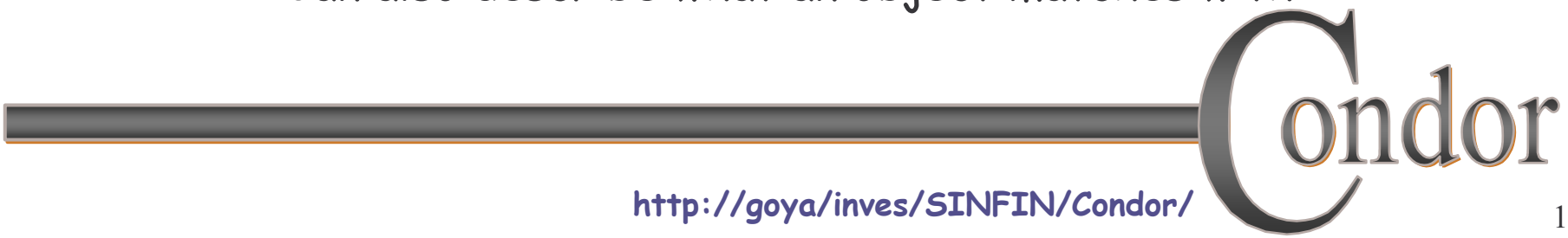
- A plain ASCII text file
- Condor does ***not*** care about file extensions
- Tells Condor about your job:
  - Which executable, universe, input, output and error files to use, command-line arguments, environment variables, any special requirements or preferences (more on this later)
- Can describe many jobs at once (a "cluster"), each with different input, arguments, output, etc.

Simple Submit Description File

```
# 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 = my_job
Queue
```

Running condor_submit

- 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 it errors, and creates a "ClassAd" that describes your job(s)
  - ClassAds: Condor's internal data representation
    - Similar to classified ads (as the name inplies)
    - Represent an object & it's attributes
    - Can also describe what an object matches with



The Job Queue

- condor_submit sends your job's ClassAd(s) to the schedd
  - Manages the local job queue
  - Stores the job in the job queue
- View the queue with *condor_q*

Running condor_submit

```
% 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    frieda           6/16 06:52    0+00:00:00 I  0    0.0  my_job

1 jobs; 1 idle, 0 running, 0 held

%
```

More information about jobs

- Controlled by submit file settings
- 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
  - To turn it on: `Log = filename`

Sample Condor User Log

```
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
...
```

Another Submit Description File

```
# Example 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 = /home/frieda/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/frieda/condor/run_1
Queue
```

“Clusters” and “Processes”

- If your submit file describes multiple jobs, we call this a “cluster”
- Each cluster has a unique “cluster number”
- Each job in a cluster is called a “process”
  - Process numbers always start at zero
- A Condor “Job ID” is the cluster number, a period, and the process number (“20.1”)
  - A cluster can have only one process (“21.0”)

Example Submit Description File for a Cluster

```
# Example submit description file that defines a
# cluster of 2 jobs with separate working directories
Universe   = vanilla
Executable = my_job
log        = my_job.log
Arguments  = -arg1 -arg2
Input      = my_job.stdin
Output     = my_job.stdout
Error      = my_job.stderr
InitialDir = run_0
Queue                 · Becomes job 2.0
InitialDir = run_1
Queue                 · Becomes job 2.1
```

Submitting The Job

```
% condor_submit my_job.submit-file
Submitting job(s).
2 job(s) submitted to cluster 2.
% condor_q
-- Submitter: perdita.cs.wisc.edu : <128.105.165.34:1027> :
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  1.0    frieda          4/15 06:52  0+00:02:11 R  0   0.0  my_job
  2.0    frieda          4/15 06:56  0+00:00:00 I  0   0.0  my_job
  2.1    frieda          4/15 06:56  0+00:00:00 I  0   0.0  my_job
3 jobs; 2 idle, 1 running, 0 held
%
```

Submit Description File for a *BIG* Cluster of Jobs

- The initial directory for each job can be specified as run_$(Process), and instead of submitting a single job, we use "Queue 600" to submit 600 jobs at once
- The $(Process) macro will be expanded to the process number for each job in the cluster (0 - 599), so we'll have "run_0", "run_1", … "run_599" directories
- All the input/output files will be in different directories!

Submit Description File for a *BIG* Cluster of Jobs

```
# Example condor_submit input file that defines
# a cluster of 600 jobs with different directories
Universe   = vanilla
Executable = my_job
Log        = my_job.log
Arguments  = -arg1 -arg2
Input      = my_job.stdin
Output     = my_job.stdout
Error      = my_job.stderr
InitialDir = run_$(Process)
Queue 600
```

- run_0 … run_599
- Becomes job 3.0 … 3.599

Using condor_rm

- 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

```
% 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?

Access to Data in Condor

- Use Shared Filesystem
- Put your files in a location shared between all machines, e.g., your home directory (/home/<user>/) or /net/<machine>/scratch/

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

Specify Requirements!

- 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
```

Specify Rank!

- 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
```

We've seen how Condor can:

… keeps an eye on your jobs and will keep you posted on their progress

… keeps a log of your job activities

… implement your policy on when the jobs can run on your workstation

My jobs run for 20 days…

- What happens when they get pre-empted?
- How can I add fault tolerance to my jobs?

Condor's Standard Universe to the rescue!

- Condor can support various combinations of features/environments in different "Universes"
- Different Universes provide different functionality for your job:
  - Vanilla – Run any Serial Job
  - Standard – Support for transparent process checkpoint and restart

Process Checkpointing

- Condor's Process Checkpointing mechanism saves the entire state of a process into a checkpoint file
  - Memory, CPU, I/O, etc.
- The process can then be restarted *from right where it left off*
- Typically no changes to your job's source code needed – however, your job must be *relinked* with Condor's Standard Universe support library

Relinking Your Job for Standard Universe

To do this, just place "*condor_compile*" in front of the command you normally use to link your job:

```
% condor_compile gcc -o myjob myjob.c
```

- OR -

```
% condor_compile f77 -o myjob filea.f fileb.f
```

Limitations of the Standard Universe

- Condor's checkpointing is not at the kernel level. Thus in the Standard Universe the job may not:
  - Fork()
  - Use kernel threads
  - Use some forms of IPC, such as pipes and shared memory
- Many typical scientific jobs are OK

When will Condor checkpoint your job?

- Periodically, if desired
  - For fault tolerance
- When your job is preempted by a higher priority job
- When your job is vacated because the execution machine becomes busy
- When you explicitly run *condor_checkpoint* command

General User Commands

- condor_status — View Pool Status
- condor_q — View Job Queue
- condor_submit — Submit new Jobs
- condor_rm — Remove Jobs
- condor_prio — Intra-User Prios
- condor_checkpoint — Force a checkpoint
- condor_compile — Link Condor library

PRACTICAL EXAMPLE

Condor Job Universes

- Serial Jobs
  - Vanilla Universe
  - Standard Universe
- Java Universe

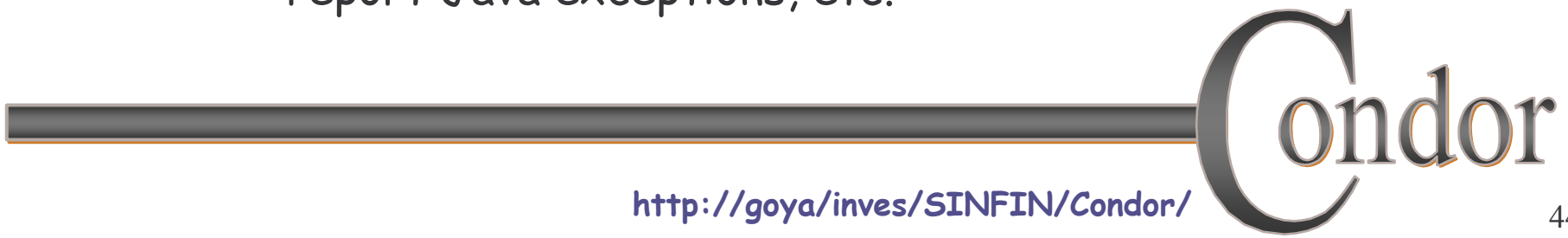
Java Universe Job

condor_submit →

```
universe = java
executable = Main.class
jar_files = MyLibrary.jar
input = infile
output = outfile
arguments = Main 1 2 3
queue
```

Why not use Vanilla Universe for Java jobs?

- Java Universe provides more than just inserting "java" at the start of the execute line
  - Knows which machines have a JVM installed
  - Knows the location, version, and performance of JVM on each machine
  - Provides more information about Java job completion than just JVM exit code
    - Program runs in a Java wrapper, allowing Condor to report Java exceptions, etc.



Java support, cont.

```
condor_status -java
```

```
Name              JavaVendor   Ver    State       Activity LoadAv Mem

aish.cs.wisc.  Sun Microsy 1.2.2  Owner       Idle     0.000   249
anfrom.cs.wis  Sun Microsy 1.2.2  Owner       Idle     0.030   249
babe.cs.wisc.  Sun Microsy 1.2.2  Claimed     Busy     1.120   123
...
```

Thank you!

Check us out on the Web:
http://goya/inves/SINFIN/Condor/

Email:
condor@iac.es