
# An Introduction to High-Throughput Computing

## Monday morning, 9:15am

Alain Roy <roy@cs.wisc.edu>

OSG Software Coordinator

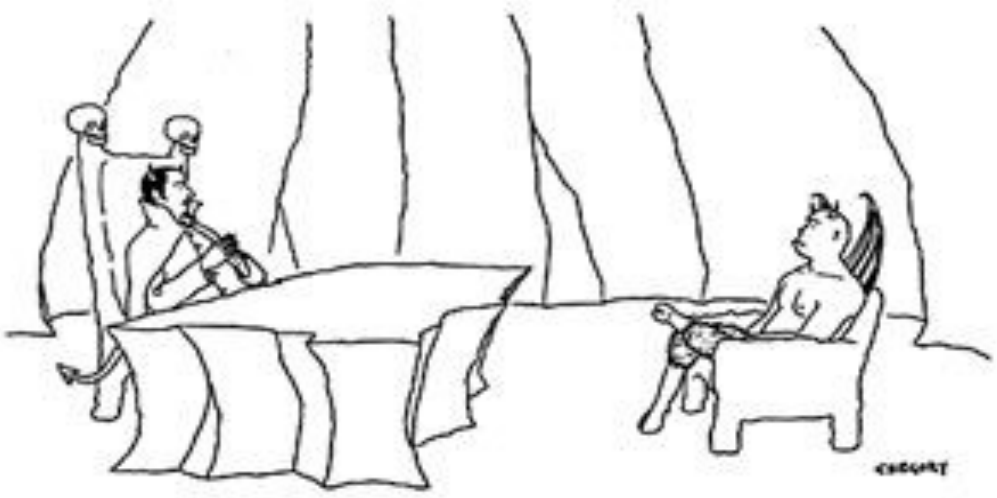
University of Wisconsin-Madison

# Who Am I?

- With Condor since 2001
- Open Science Grid Software Coordinator
- Taught at six previous summer schools
- I have the two cutest kids in the whole world:

# Overview of day

- Lectures alternating with exercises
  - Emphasis on lots of exercises
  - Hopefully overcome PowerPoint fatigue & help you understand better



*"I need someone well versed in the art of torture—do you know PowerPoint?"*

# Some thoughts on the exercises

- It's okay to move ahead on exercises if you have time
- It's okay to take longer on them if you need to
- If you move along quickly, try the "On Your Own" sections and "Challenges"
- We'll have an optional evening session if you want more time, and I'll be there to give help

Open Science Grid

# Most important!

- Please ask me questions!
  - …during the lectures
  - …during the exercises
  - …during the breaks
  - …during the meals
  - …over dinner
  - …the rest of the week

Open Science Grid

# Before we start

- Sometime today, do the exercise on getting a certificate.
  - It is **required** for all exercises Tuesday-Thursday
  - It will be easiest if you do it today

# Goals for this session

- Understand basics of high-throughput computing
- Understand the basics of Condor
- Run a basic Condor job


# What is high-throughput computing? (HTC)

- An approach to distributed computing that focuses on long-term throughput, not instantaneous computing power.
  - We don't care about operations per second
  - We care about operations per year
- Implications:
  - Focus on reliability
  - Use all available resources (not just high-end supercomputers)
    - That slow four-year old cluster down the hall? Include it!

# Good uses of HTC

- “I need as many simulation results as possible before my deadline…”
- “I  have lots of small-ish independent tasks that can be run indepdently”

Open Science Grid

# What's not HTC?

- The need for "real-time" results:
  - (Ignore fact that "real-time" is hard to define.)
  - HTC is less worried about latency (delay to answer) than total throughput
- The need to maximize FLOPS
  - "I must use the a supercomputer, because I need the fastest computer/network/storage/ …"

Open Science Grid

# An example problem: BLAST

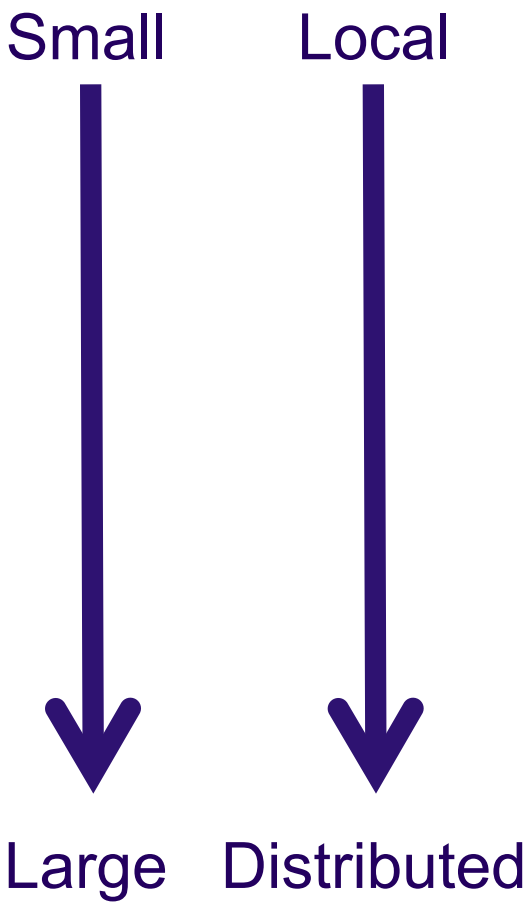
- A scientist has:
  - Question: Does a protein sequence occur in other organisms?
  - Data: lots of protein sequences from various organisms
  - Parameters: how to search the database.
- More throughput means
  - More protein sequences queried
  - Larger/more protein data bases examined
  - More parameter variation
- We'll try out BLAST later today

Open Science Grid

# Why is HTC hard?

- The HTC system has to keep track of:
  - Individual tasks (a.k.a. jobs) & their inputs
  - Computers that are available
- The system has to recover from failures
  - There will be failures! Distributed computers means more chances for failures.
- You have to share computers
    - Sharing can be within an organization, or between orgs
    - So you have to worry about security.
    - And you have to worry about policies on how you share.
- If you use a lot of computers, you have to deal variety:
  - Different kinds of computers (arch, OS, speed, etc..)
  - Different kinds of storage (access methodology, size, speed, etc…)
  - Different networks interacting (network problems are hard to debug!)

# Let's take one step at a time




- Can you run one job on one computer?
- Can you run one job on another local computer?
- Can you run 10 jobs on a set of local computers?
- Can you run 1 job on a remote computer?
- Can you run 10 jobs at a remote site?
- Can you run a mix of jobs here and remotely?

This is the (rough) path we'll take in the school this week

Open Science Grid

# Discussion

- For 5 minutes, talk to a neighbor: If you want to run one job in a local cluster of computers:
  1) What do you (the user) need to provide so a single job can be run?
  2) What does the system need to provide so your single job can be run?
     - Think of this as a set of processes: what needs happen when the job is given? A "process" could be a computer process, or just an abstract task.

# Alain's answer: What does the user provide?

- A "headless job".
  - Not interactive/no GUI: how could you interact with 1000 simultaneous jobs?
- A set of input files
- A set of output files.
- A set of parameters (command-line arguments).
- Requirements:
  - Ex: My job requires at least 2GB of RAM.
  - Ex: My job requires Linux.
- Control/Policy:
  - Ex: Send me email when the job is done.
  - Ex: Job 2 is more important than Job 1.
  - Ex: Kill my job if it's run for more than 6 hours.

# Alain's answer: What does the system provide?

- Methods to:
  - Submit/Cancel job.
  - Check on state of job.
  - Check on state of available computers.
- Processes to:
  - Reliably track set of submitted jobs.
  - Reliably track set of available computers.
  - Decide which job runs on which computer.
  - Manage a single computer.
  - Start up a single job.


# Surprise! Condor does this (and more)

- Methods to:
  - Submit/Cancel job. `condor_submit/condor_rm`
  - Check on state of job. `condor_q`
  - Check on state of avail. computers. `condor_status`
- Processes to:
  - Reliably track set of submitted jobs. `schedd`
  - Reliably track set of avail. computers. `collector`
  - Decide which job runs on where. `negotiator`
  - Manage a single computer `startd`
  - Start up a single job `starter`

Open Science Grid

# But not only Condor

- You can use other systems:
  - PBS/Torque
  - Oracle Grid Engine (né Sun Grid Engine)
  - LSF
  - ...
- But we won't cover them.
  - Our expertise is with Condor
  - Our bias is with Condor
- What should you learn?
  - How do you think about HTC?
  - How can you do your science with HTC?
  - … For now, learn it with Condor, but you can apply it to other systems.

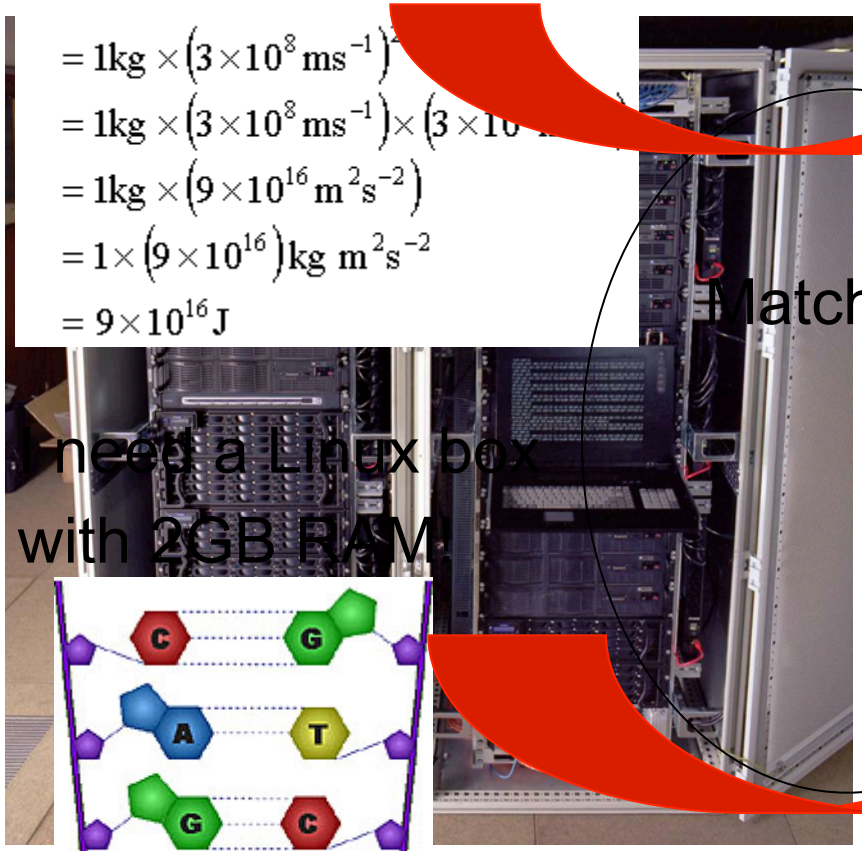

# A brief introduction to Condor

# And finally Takes Computers…

I need a Mac!

Desktop Computers

$$E = mc^2$$
$$= 1\text{kg} \times (3\times10^8\,\text{ms}^{-1})^2$$
$$= 1\text{kg} \times (3\times10^8\,\text{ms}^{-1}) \times (3\times10^8\,\text{ms}^{-1})$$
$$= 1\text{kg} \times (9\times10^{16}\,\text{m}^2\text{s}^{-2})$$
$$= 1\times(9\times10^{16})\,\text{kg m}^2\text{s}^{-2}$$
$$= 9\times10^{16}\,\text{J}$$



Matchi

I need a Linux box with 2GB RAM!

Open Science Grid

# Quick Terminology

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

Open Science Grid

# Matchmaking

- Matchmaking is fundamental to Condor
- Matchmaking is two-way
  - Job describes what it requires:

    I need Linux && 8 GB of RAM
  - Machine describes what it requires:

    I will only run jobs from the Physics department
- Matchmaking allows preferences
  - I need Linux, and I prefer machines with more memory but will run on any machine you provide me

# Why Two-way Matching?

- Condor conceptually divides people into three groups:
  - Job submitters
  - Machine owners
  - Pool (cluster) administrator

} May or may not be the same people

- All three of these groups have preferences

# ClassAds

- 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

Open Science Grid

# ClassAds

- ClassAds are:
  - semi-structured
  - user-extensible
  - schema-free
  - Attribute = Expression

Example:

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

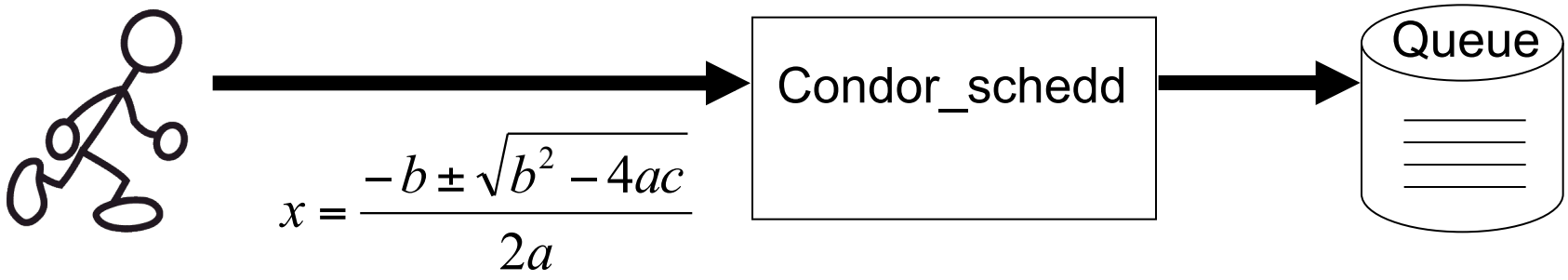
Open Science Grid

# Schema-free ClassAds

- Condor imposes some schema
  - Owner is a string, ClusterID is a number…
- But users can extend it however they like, for jobs or machines
  - `AnalysisJobType = "simulation"`
  - `HasJava_1_4 = TRUE`
  - `ShoeLength = 7`
- Matchmaking can use these attributes
  - `Requirements = OpSys == "LINUX"`
    `&& HasJava_1_4 == TRUE`

# Submitting jobs: condor_schedd

- Users submit jobs from a computer
  - Jobs described as ClassAds
  - 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*



# Advertising computers

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

ClassAd

`Type = "Machine"`

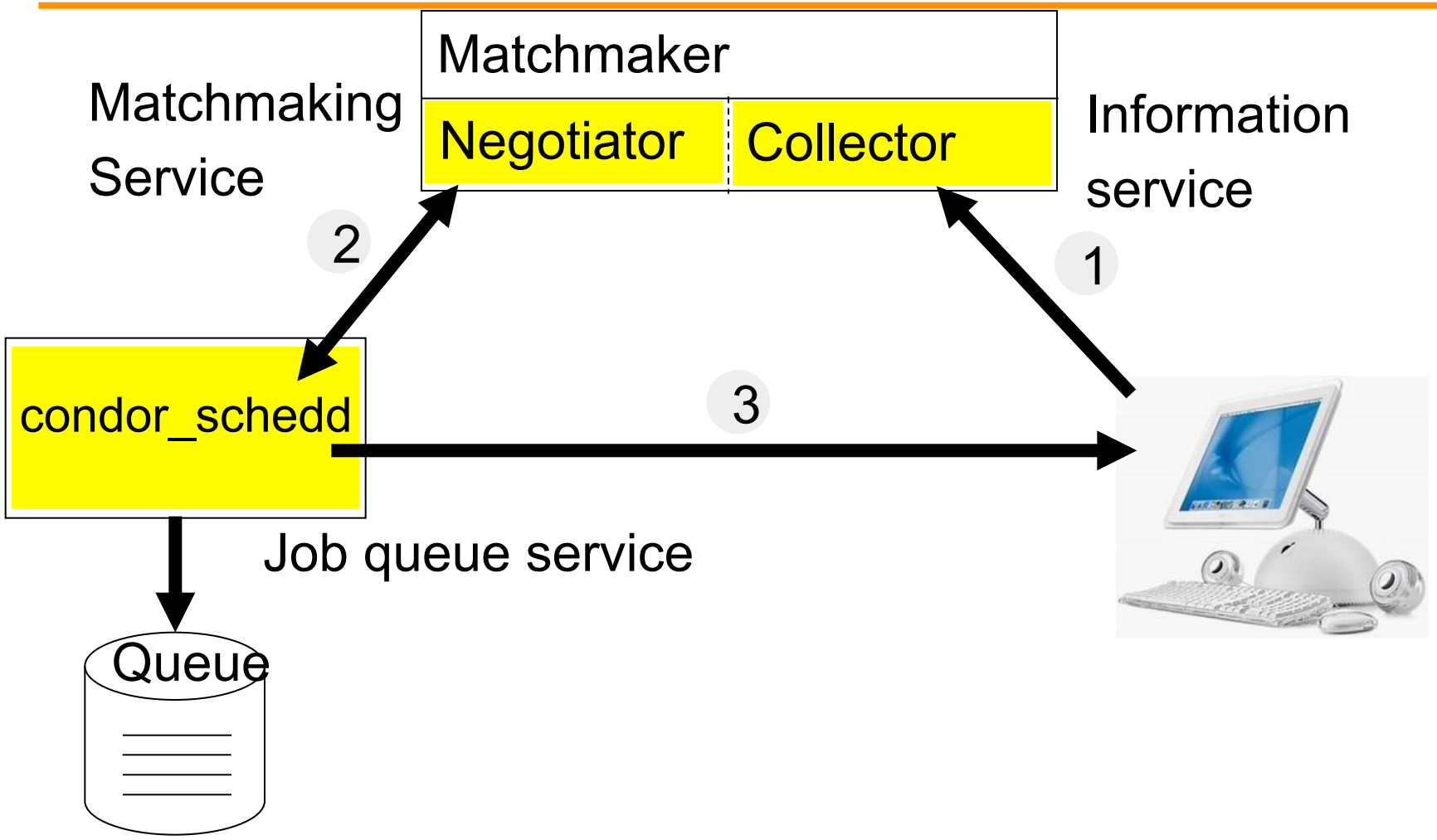
`Requirements = "…"`

Matchmaker (Collector)

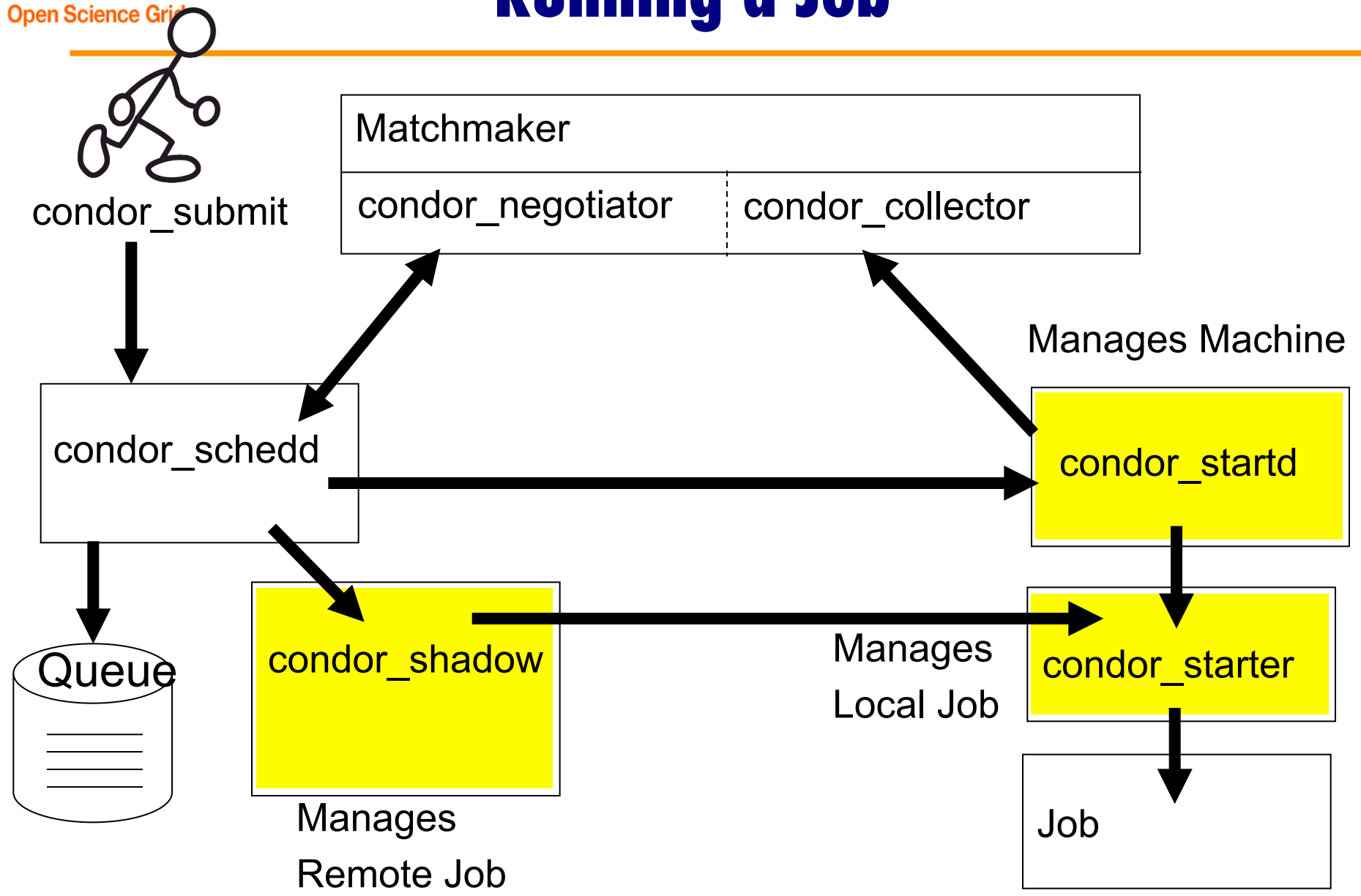

# Matchmaking

- 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

# Matchmaking diagram



# Running a Job



Open Science Grid

# Condor processes

| Proceess | Function |
|---|---|
| Master | Takes care of other processes |
| Collector | Stores ClassAds |
| Negotiator | Performs Matchmaking |
| Schedd | Manages job queue |
| Shadow | Manages job (submit side) |
| Startd | Manages computer |
| Starter | Manages job (execuction side) |


# If you forget most of these remember two (for other lectures)

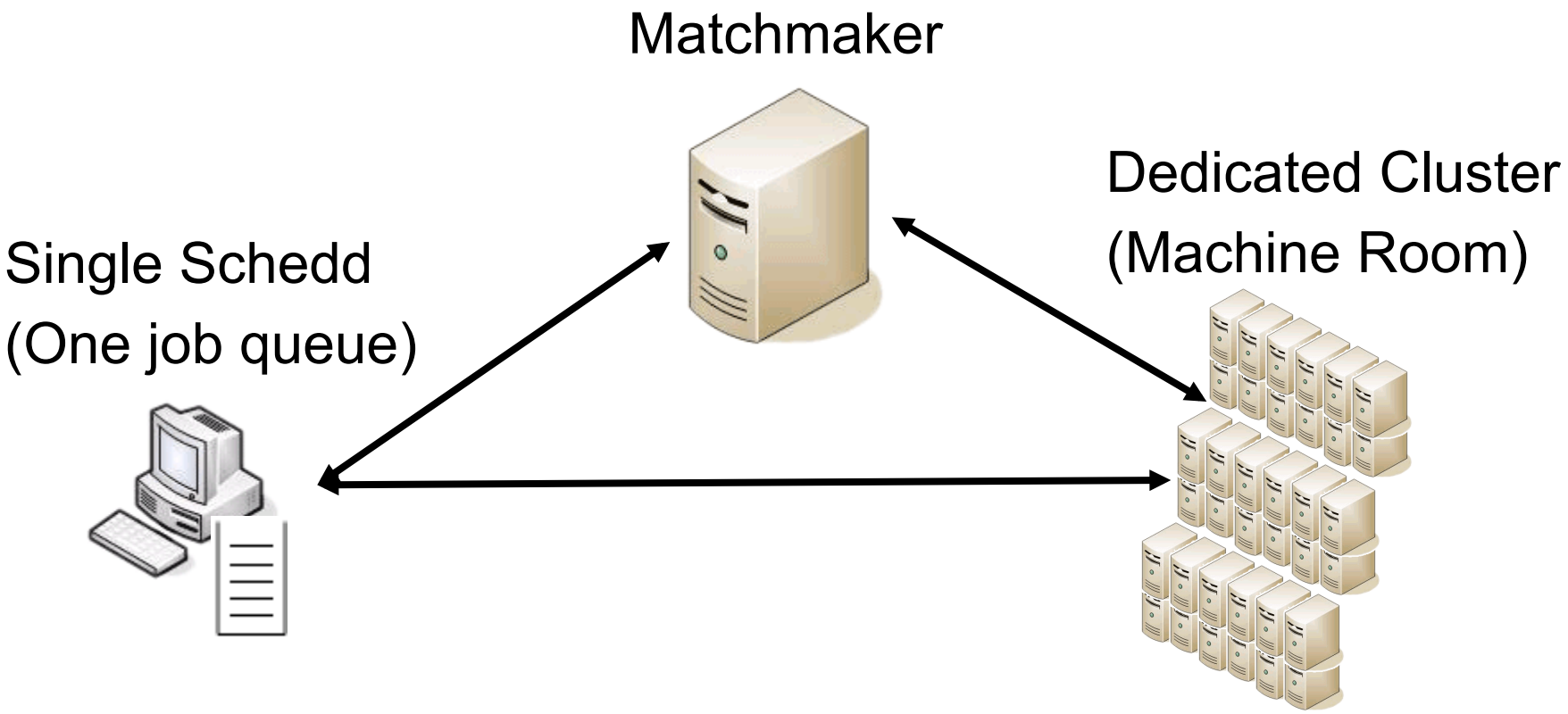
| Proceess | Function |
|---|---|
| Master | Takes care of other processes |
| Collector | Stores ClassAds |
| Negotiator | Performs Matchmaking |
| **Schedd** | **Manages job queue** |
| Shadow | Manages job (submit side) |
| **Startd** | **Manages computer** |
| Starter | Manages job (execuction side) |

Open Science Grid

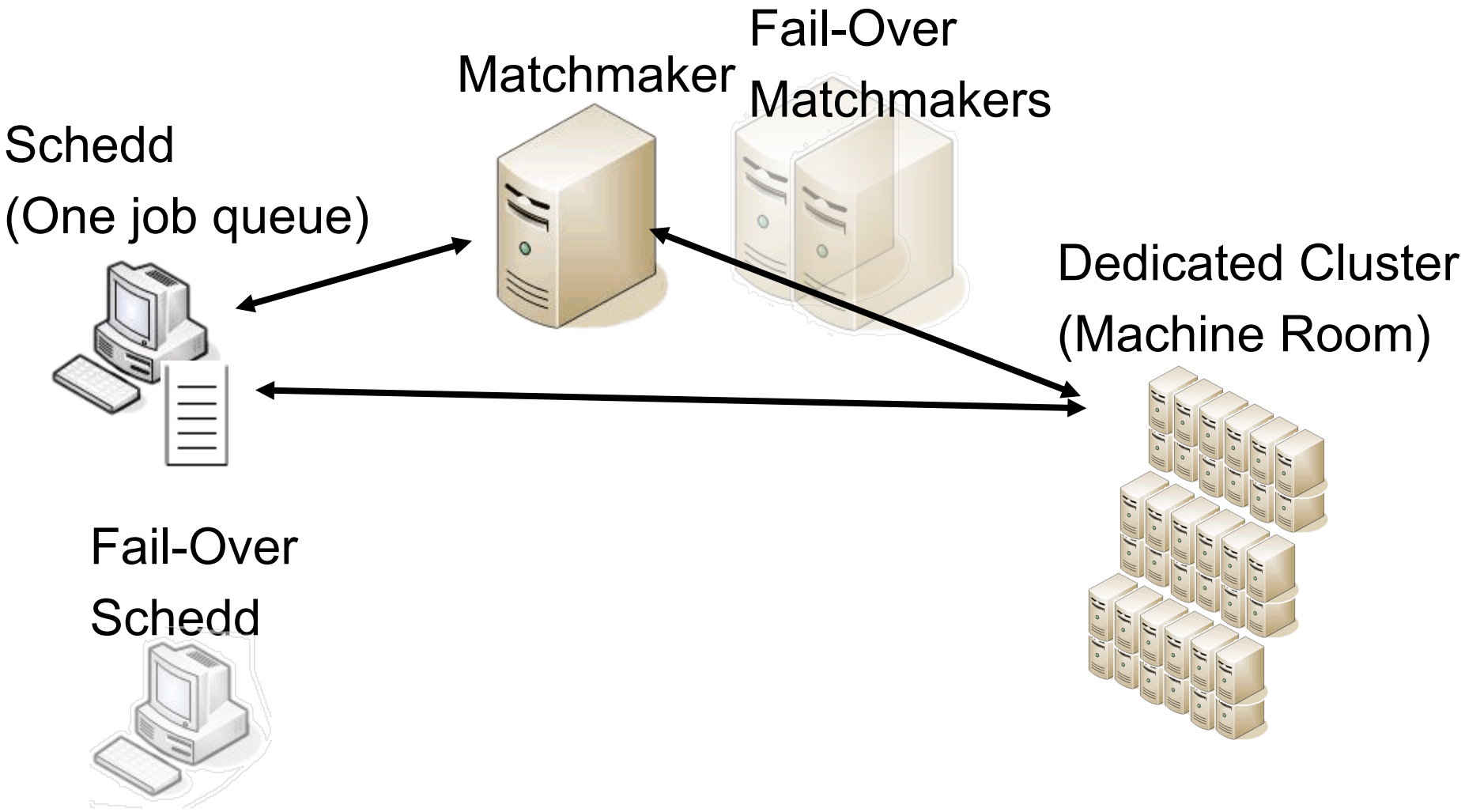
# Some notes

- One negotiator/collector per pool
- Can have many schedds (submitters)
- Can have many startds (computers)
- A machine can have any combination of:
  - Just a startd (typical for a dedicated cluster)
  - schedd + startd (perhaps a desktop)
  - Personal Condor: everything

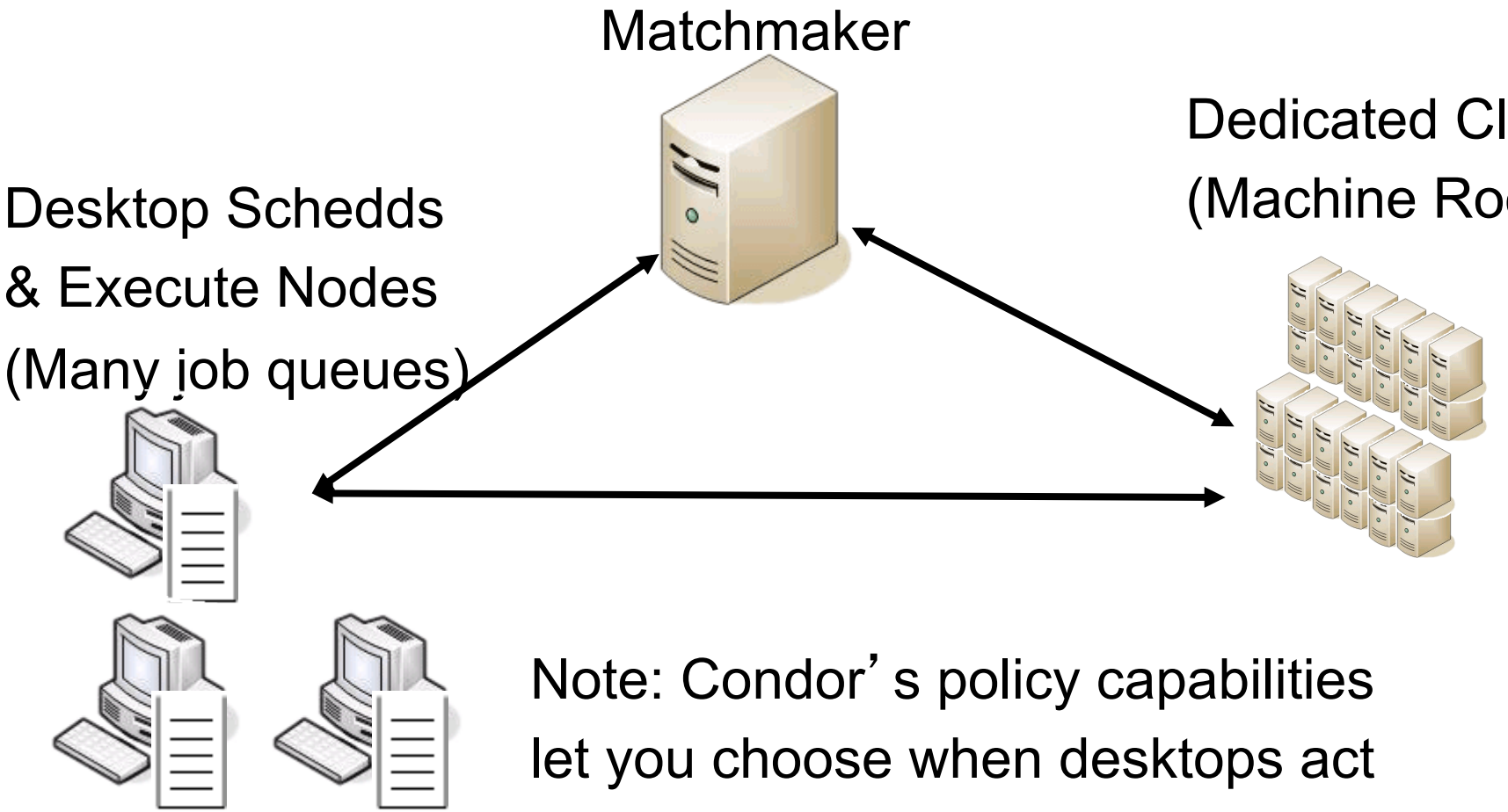
# Example Pool 1



# Example Pool 1a



# Example Pool 2




Note: Condor's policy capabilities let you choose when desktops act as execute nodes.


# Our Condor Pools

- One submit computer
  - One schedd/queue for everyone
  - vdt-itb.cs.wisc.edu
- One local set of dedicated Condor execute nodes:
  - About 8 computers, about 45 available “batch slots”
- Remote resources at UNL
  - For Tuesday, not today.


# Our Condor Pool

```
Name               OpSys      Arch   State     Activity LoadAv Mem   ActvtyTime

slot1@miniosg-c01. LINUX      INTEL  Unclaimed Idle  0.000  2030  0+00:00:04
slot2@miniosg-c01. LINUX      INTEL  Unclaimed Idle  0.000  2030  0+00:00:05
slot3@miniosg-c01. LINUX      INTEL  Unclaimed Idle  0.000  2030  0+00:29:46
slot4@miniosg-c01. LINUX      INTEL  Unclaimed Idle  0.000  2030  0+00:29:47
slot5@miniosg-c01. LINUX      INTEL  Unclaimed Idle  0.020  2030  0+00:07:45
slot6@miniosg-c01. LINUX      INTEL  Unclaimed Idle  0.000  2030 13+01:43:11
slot7@miniosg-c01. LINUX      INTEL  Unclaimed Idle  0.000  2030 13+01:43:12
slot8@miniosg-c01. LINUX      INTEL  Unclaimed Idle  0.000  2030 13+01:43:05
slot1@miniosg-c02. LINUX      INTEL  Unclaimed Idle  0.000  2030  0+05:35:22
slot2@miniosg-c02. LINUX      INTEL  Unclaimed Idle  0.000  2030  0+05:45:29
slot3@miniosg-c02. LINUX      INTEL  Unclaimed Idle  0.000  2030  0+01:35:06
...

                     Total Owner Claimed Unclaimed Matched Preempting Backfill

  INTEL/LINUX           44     0      44         0       0          0

         Total          44     0      44         0       0          0
```

# That was a whirlwind tour!

- Let's get some hands-on experience with Condor, to solidify this knowledge.
- Goal: Check out our installation, run some basic jobs.

Open Science Grid

# Questions?

- Questions? Comments?
  - Feel free to ask me questions later:

    Alain Roy <roy@cs.wisc.edu>
- Upcoming sessions
  - 9:45-10:30
    - Hands-on exercises
  - 10:30 – 10:45
    - Break
  - 10:45 – 12:15
    - More!