An Introduction to High-Throughput Computing With Condor

Tuesday morning, 9am

Zach Miller <zmiller@cs.wisc.edu>

University of Wisconsin-Madison

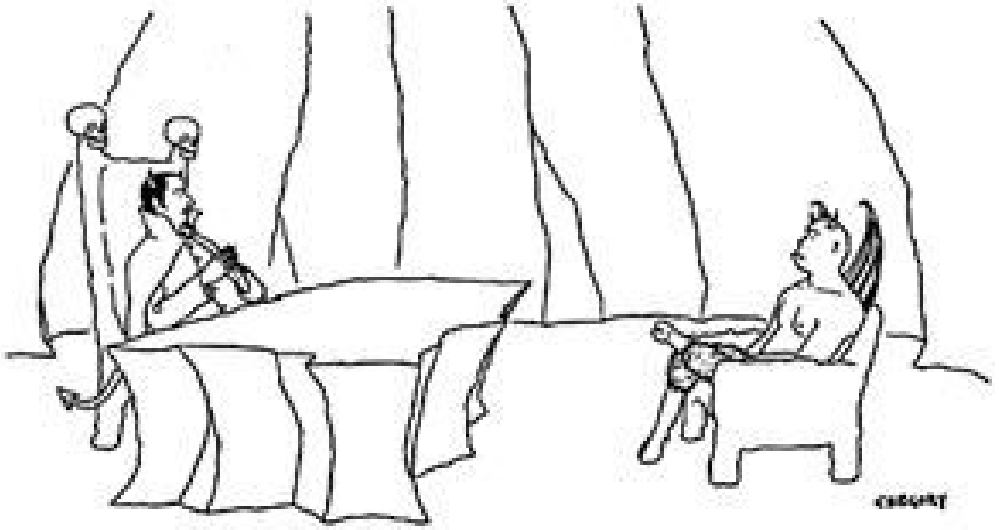
Open Science Grid

Who Am I?

- With Condor since 2001
- Developer of the Core Condor Software
- Part of the CHTC (Center for High-Throughput Computing) engagement team at UW-Madison
- Taught at previous summer schools
- Eu gosto muito a musica de brasil! :)

Overview of day

- Lectures alternating with exercises
  -
  -



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

Open Science Grid

Overview of day

- 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"

Most important!

- Please ask me questions!
  -
  -
  -
  -

Before we start

- If you haven't already, do the exercise from Monday on getting a certificate sometime today!

Open Science Grid

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.
  -
  -
- Implications:
  -
  -

Open Science Grid

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”

What's not HTC?

- The need for "real-time" results:
  -
  -
- The need to maximize FLOPS
  -

An example problem: BLAST

- A scientist has:
  -
  -
  -
- More throughput means
  -
  -
  -
- We'll try out BLAST later today

Why is HTC hard?

- The HTC system has to keep track of:
  -
  -
- The system has to recover from failures
  -
- You have to share computers
  -
  -
  -
- If you use a lot of computers, you have to deal variety:
  -
  -
  -

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) progress we'll take in the school.

Large Distributed

Discussion

- For 5 minutes, talk to a neighbor: If you want to run one job in a local cluster of computers:
  -
  -
    -

One answer: What does the user provide?

- A “headless job”.
  -
- A set of input files
- A set of output files.
- A set of parameters (command-line arguments).
- Requirements:
  -
  -
- Control:
  -
  -
  -

One answer: What does the system provide?

- Methods to:
  -
  -
  -
- Processes to:
  -
  -
  -
  -
  -

Surprise! Condor does this (and more)

- Methods to:
  -
  -
  -
- Processes to:
  -
  -

A brief introduction to Condor

Open Science Grid

Quick Terminology

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

Matchmaking

- Matchmaking is fundamental to Condor
- Matchmaking is two-way
  - I need Linux && 8 GB of RAM
  - I will only run jobs from the Physics department
- Matchmaking allows preferences
  -

Why Two-way Matching?

- Condor conceptually divides people into three groups:
  -
  -
  -

} May or may not be the same people

- All three of these groups have preferences

ClassAds

- ClassAds state facts
  –
  –
- ClassAds state preferences
  –

  with Linux

ClassAds

- ClassAds are:
  -
  -
  -
  -

Example:

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

Open Science Grid

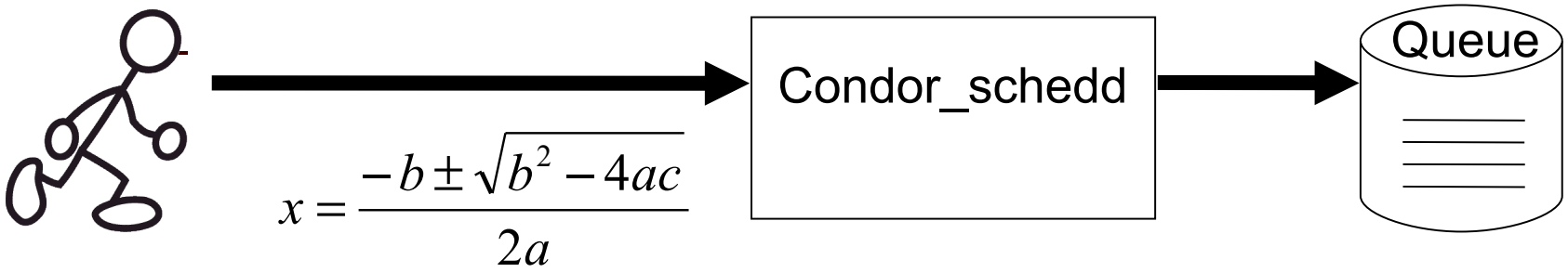
Schema-free ClassAds

- Condor imposes some schema
  -
- But users can extend it however they like, for jobs or machines
  -
  -
  -
- Matchmaking can use these attributes
  -

```
&& HasJava_1_4 == TRUE
```

Submitting jobs

- Users submit jobs from a computer
  -
  -
  -
  -
  -
  -



Advertising computers

- Machine owners describe computers
  -
  -
    -
    -
    -
  -

ClassAd

`Type = "Machine"`

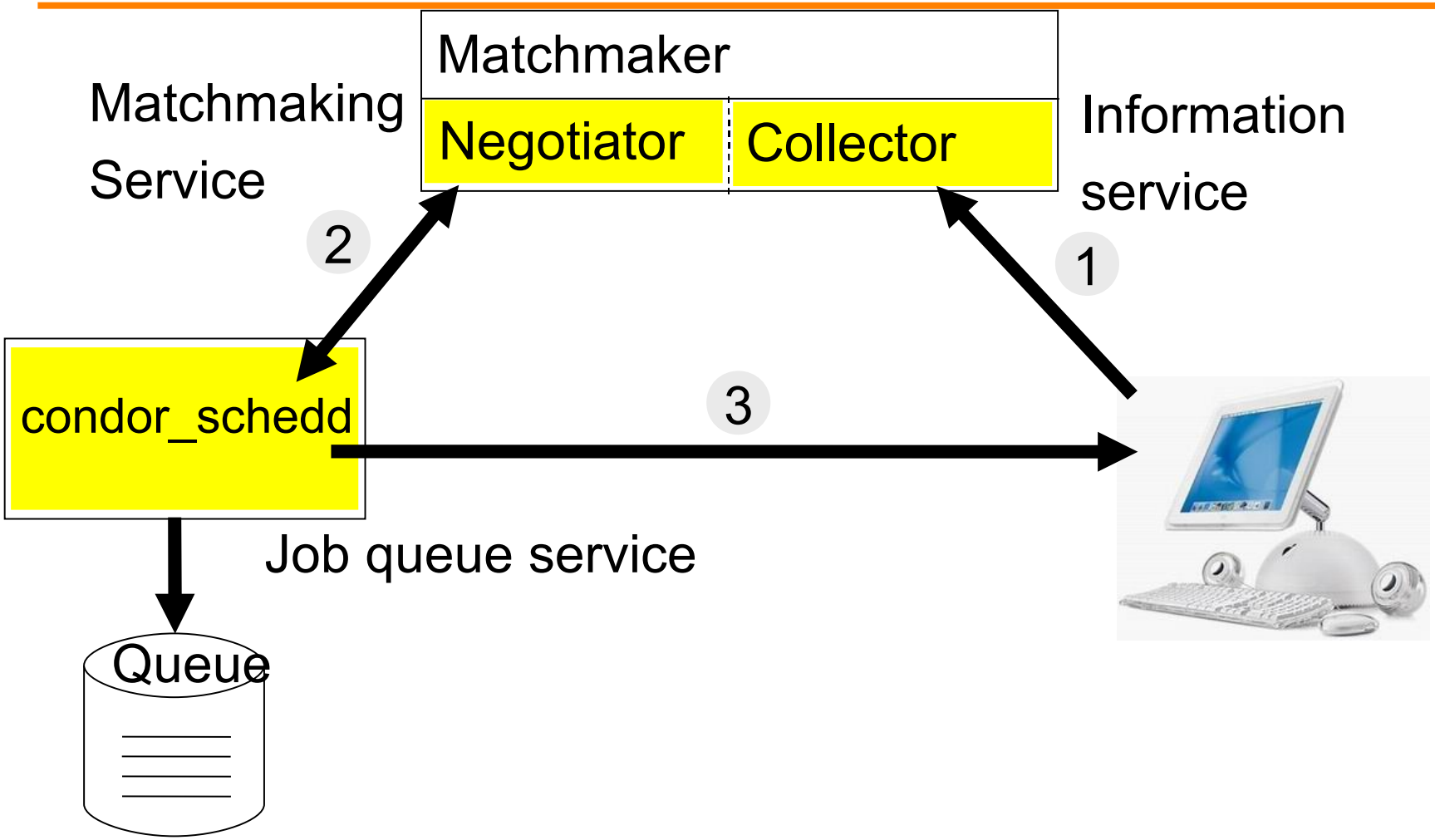
`Requirements = "…"`

Matchmaker (Collector)

Matchmaking

- Negotiator collects list of computers
- Negotiator contacts each schedd
  -
- Negotiator compares each job to each computer
  -
  -
  -
- Upon match, schedd contacts execution computer

Matchmaking diagram



Running a Job

condor_submit

Matchmaker

condor_negotiator | condor_collector

Manages Machine

condor_schedd

condor_startd

Queue

condor_shadow

Manages
Local Job

condor_starter

Job

Manages
Remote Job

Open Science Grid

Condor processes

- 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 (execution side)

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

- Master: Takes care of other processes
- Collector: Stores ClassAds
- Negotiator: Performs matchmaking

Schedd: Manages job queue

- Shadow: Manages job (submit side)

Startd: Manages computer

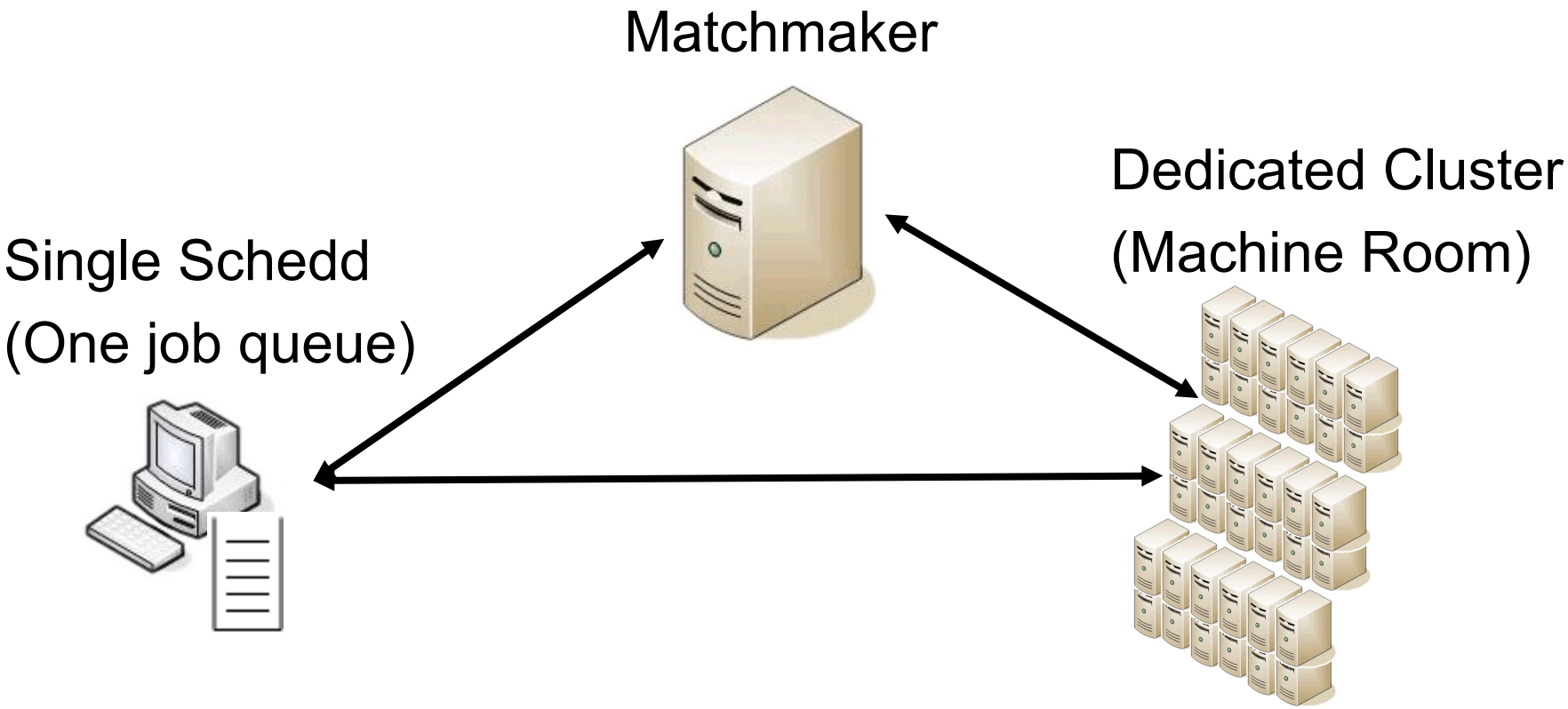
- Starter: Manages job (execution side)

Open Science Grid

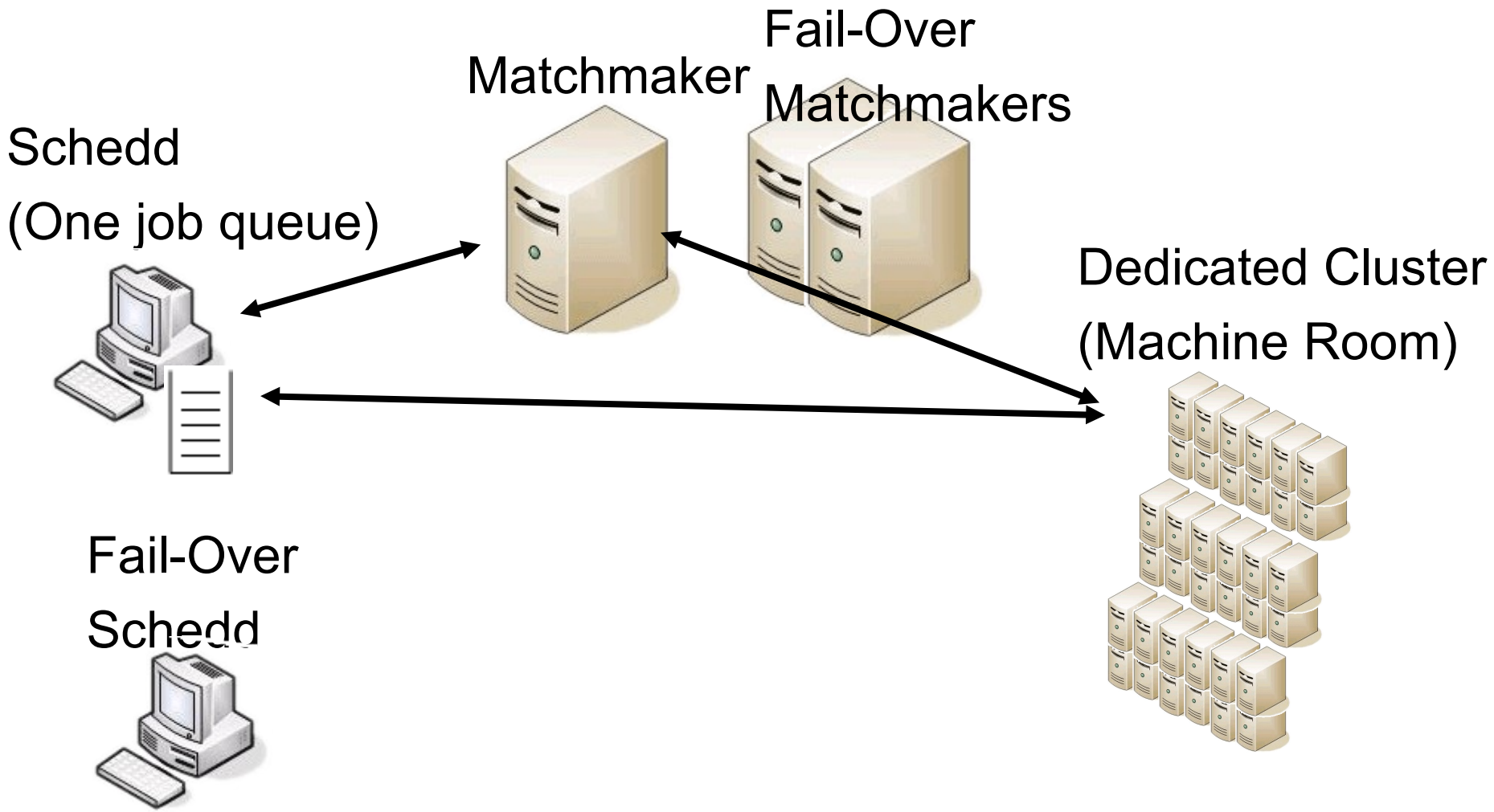
Some notes

- One negotiator/collector per pool
- Can have many schedds (submitters)
- Can have many startds (computers)
- A machine can have any combination of:
  -
  -
  -

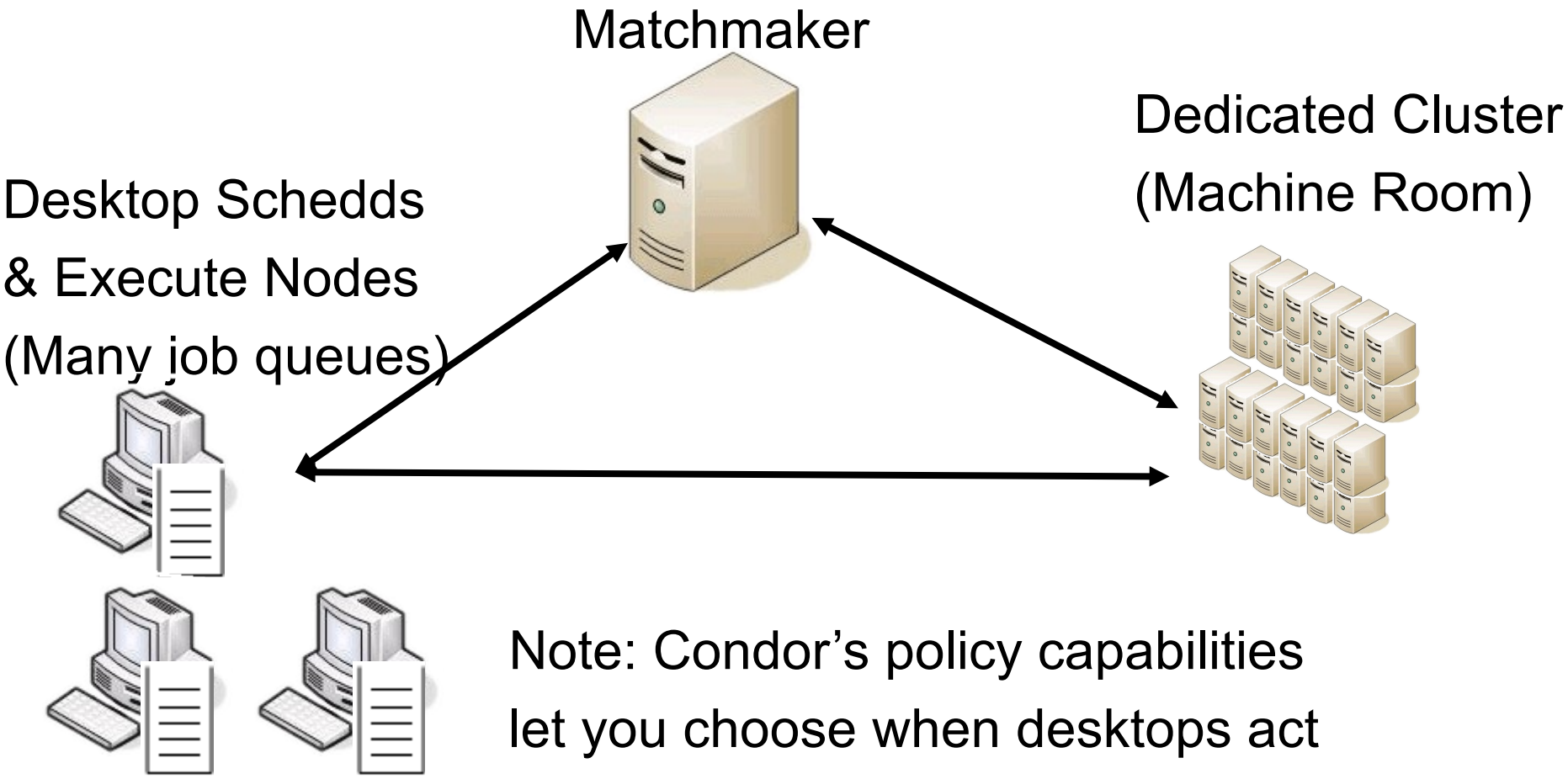
Example Pool 1



Example Pool 1a



Example Pool 2



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

Open Science Grid

Our Condor Pools

- Each computer is also a submit computer
  -
- One local set of Condor execute nodes:
  -
- Remote resources on “the Grid”
  -

Our Condor Pool

```
Name                OpSys      Arch   State     Activity LoadAv Mem   ActvtyTime

slot1@treinamento0 LINUX      X86_64 Unclaimed Idle     0.570   498  0+01:52:06
slot2@treinamento0 LINUX      X86_64 Unclaimed Idle     0.000   498  0+05:52:20
slot3@treinamento0 LINUX      X86_64 Unclaimed Idle     0.000   498  0+05:52:21
slot4@treinamento0 LINUX      X86_64 Unclaimed Idle     0.000   498  0+05:52:22
slot1@treinamento0 LINUX      X86_64 Unclaimed Idle     0.230   498  0+01:51:07
slot2@treinamento0 LINUX      X86_64 Unclaimed Idle     0.000   498  0+05:51:24
slot3@treinamento0 LINUX      X86_64 Unclaimed Idle     0.000   498  0+05:51:25
slot4@treinamento0 LINUX      X86_64 Unclaimed Idle     0.000   498  0+05:51:26

...

                     Total Owner Claimed Unclaimed Matched Preempting Backfill

       X86_64/LINUX     72     0       0        72       0          0        0

              Total     72     0       0        72       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.

Questions?

- Questions? Comments?
  - Zach Miller <zmiller@cs.wisc.edu>
- Upcoming sessions
  -
  -
  -
  -