More HTCondor

2013 OSG User School, Monday, Lecture 2

Tim Cartwright
`cat@cs.wisc.edu`

University of Wisconsin–Madison

OSG Software Team Manager
OSG Education Coordinator

Questions so far?

Goals For This Session

- Understand the mechanisms of HTCondor (and HTC in general) a bit more deeply
- Use a few more HTCondor features
- Run more (and more complex) jobs at once

HTCondor in Depth

Open Science Grid

Why Is HTC Difficult?

- System must track jobs, machines, policy, …
- System must recover gracefully from failures
- Try to use all available resources, all the time
- Lots of variety in users, machines, networks, …
- Sharing is hard (e.g., policy, security)

- More about the principles of HTC on Thursday

Open Science Grid

Main Parts of HTCondor

Open Science Grid

Main Parts of HTCondor

Function
Track waiting/running jobs
Track available machines
Match jobs and machines
Manage one machine
Manage one job (on submitter)
Manage one job (on machine)

Open Science Grid

Main Parts of HTCondor

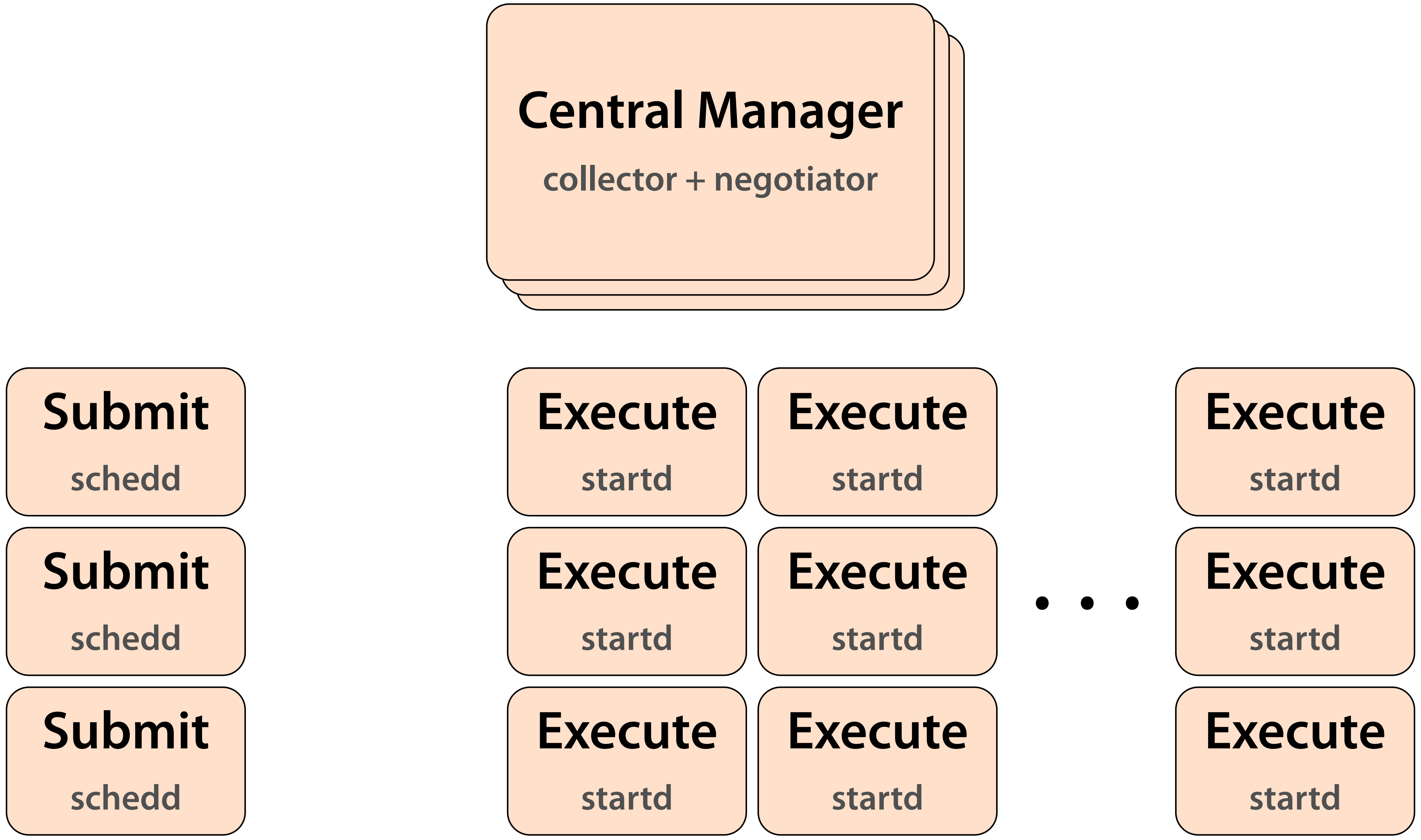
Function	HTCondor Name
Track waiting/running jobs	schedd (“sked-dee”)
Track available machines	collector
Match jobs and machines	negotiator
Manage one machine	startd (“start-dee”)
Manage one job (on submitter)	shadow
Manage one job (on machine)	starter

Open Science Grid

Main Parts of HTCondor

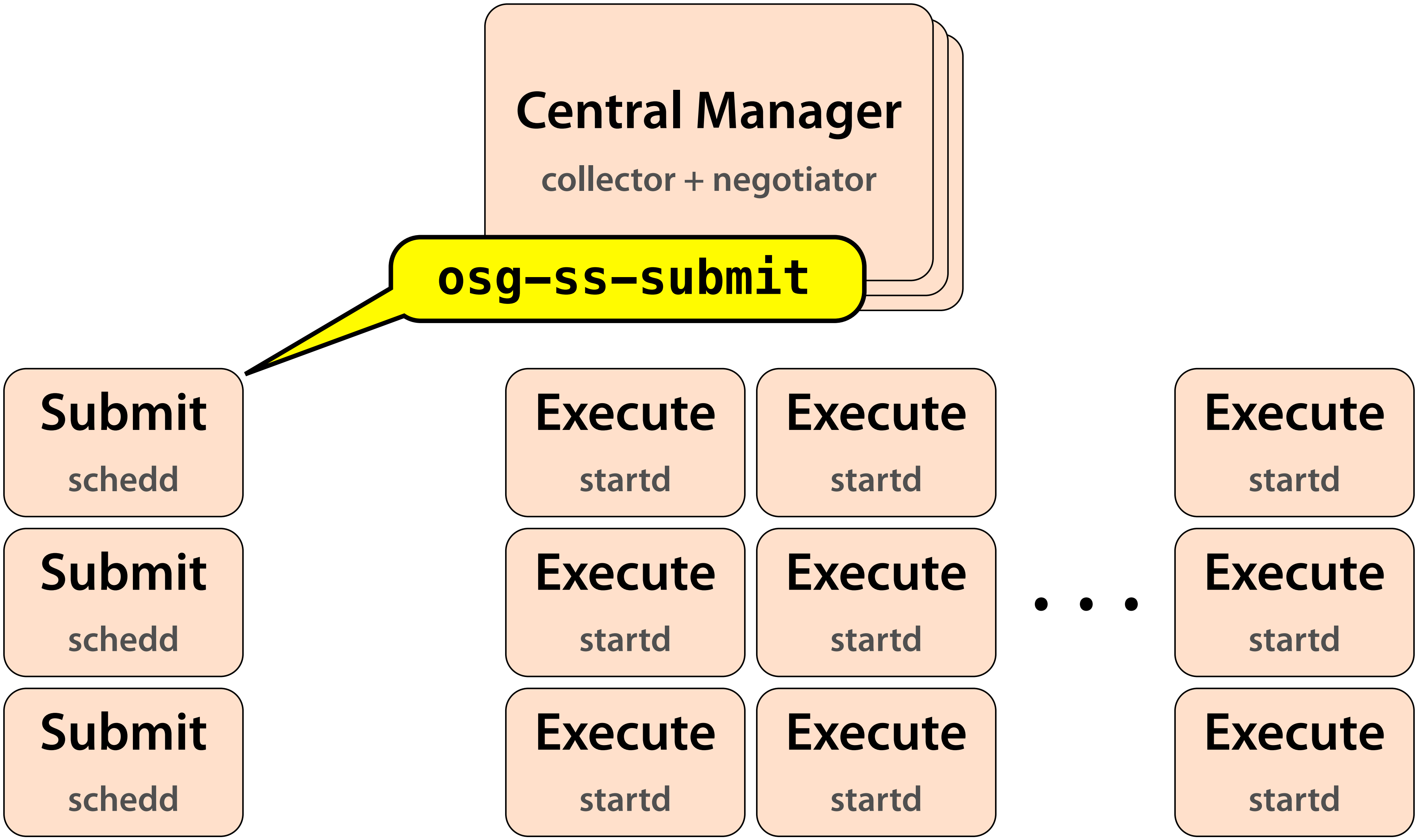
Function	HTCondor Name	#
Track waiting/running jobs	schedd ("sked-dee")	1+
Track available machines	collector	1
Match jobs and machines	negotiator	1
Manage one machine	startd ("start-dee")	per machine
Manage one job (on submitter)	shadow	per job running
Manage one job (on machine)	starter	per job running

Typical Architecture

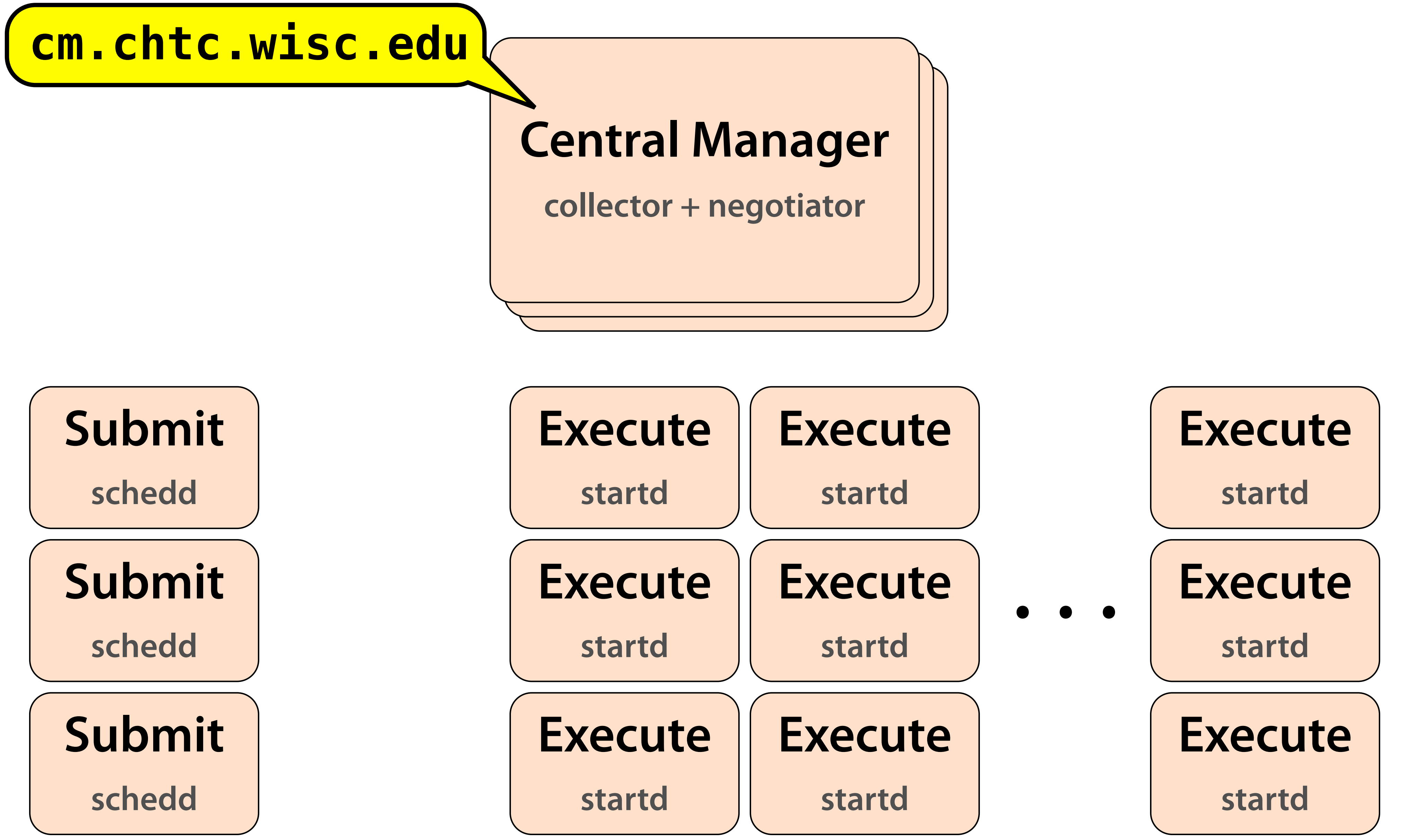


Open Science Grid

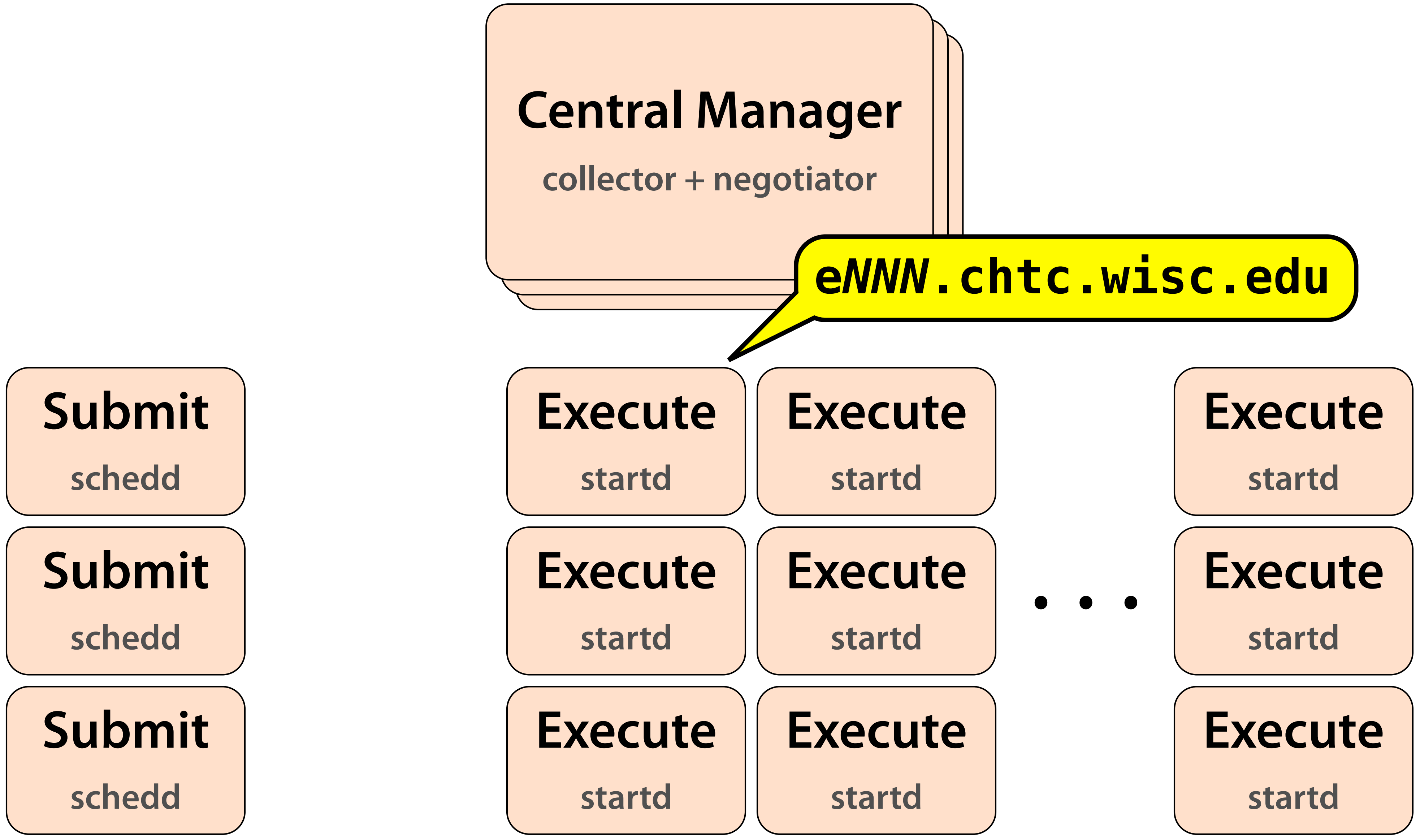
Typical Architecture



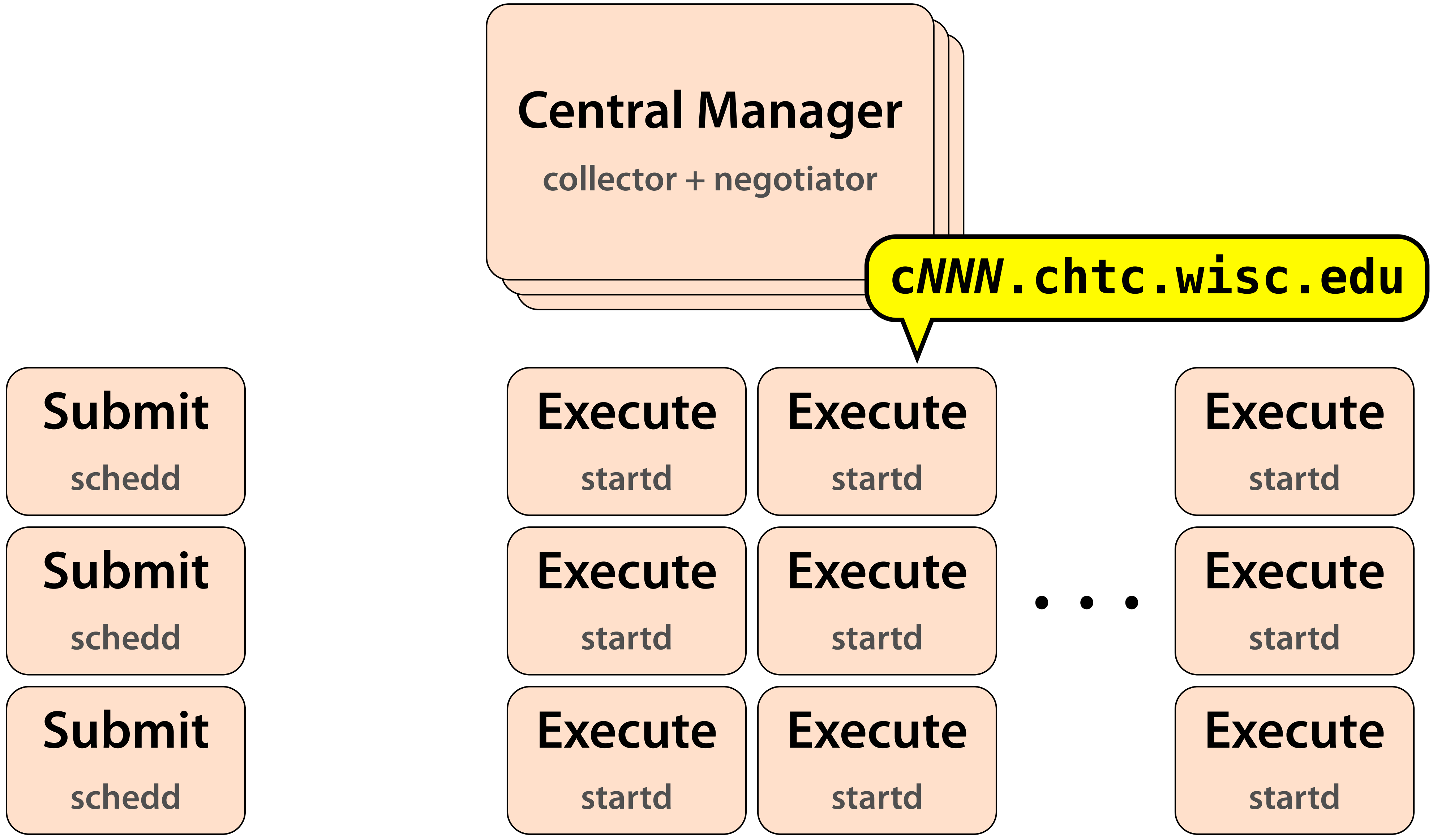
Typical Architecture



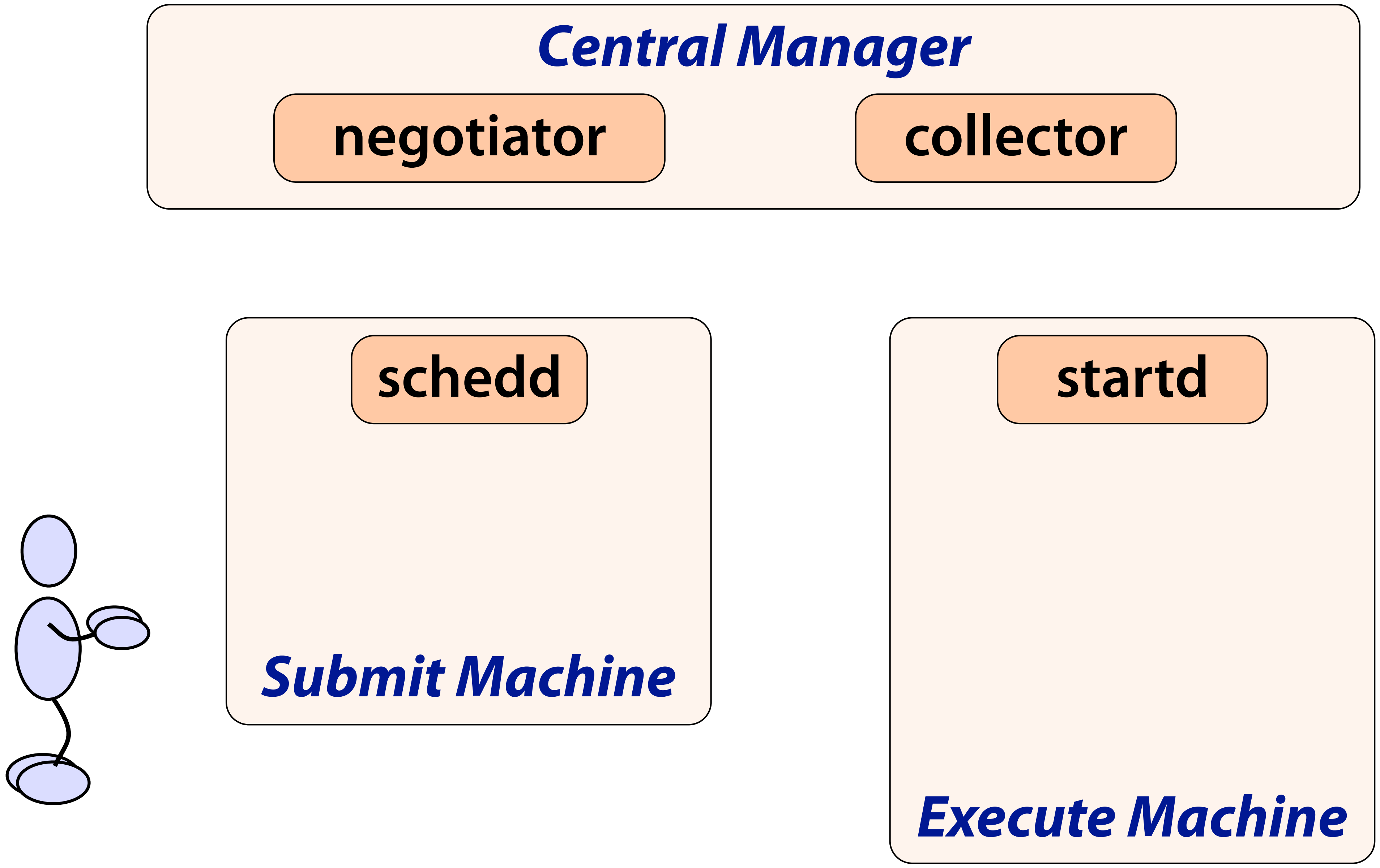
Typical Architecture



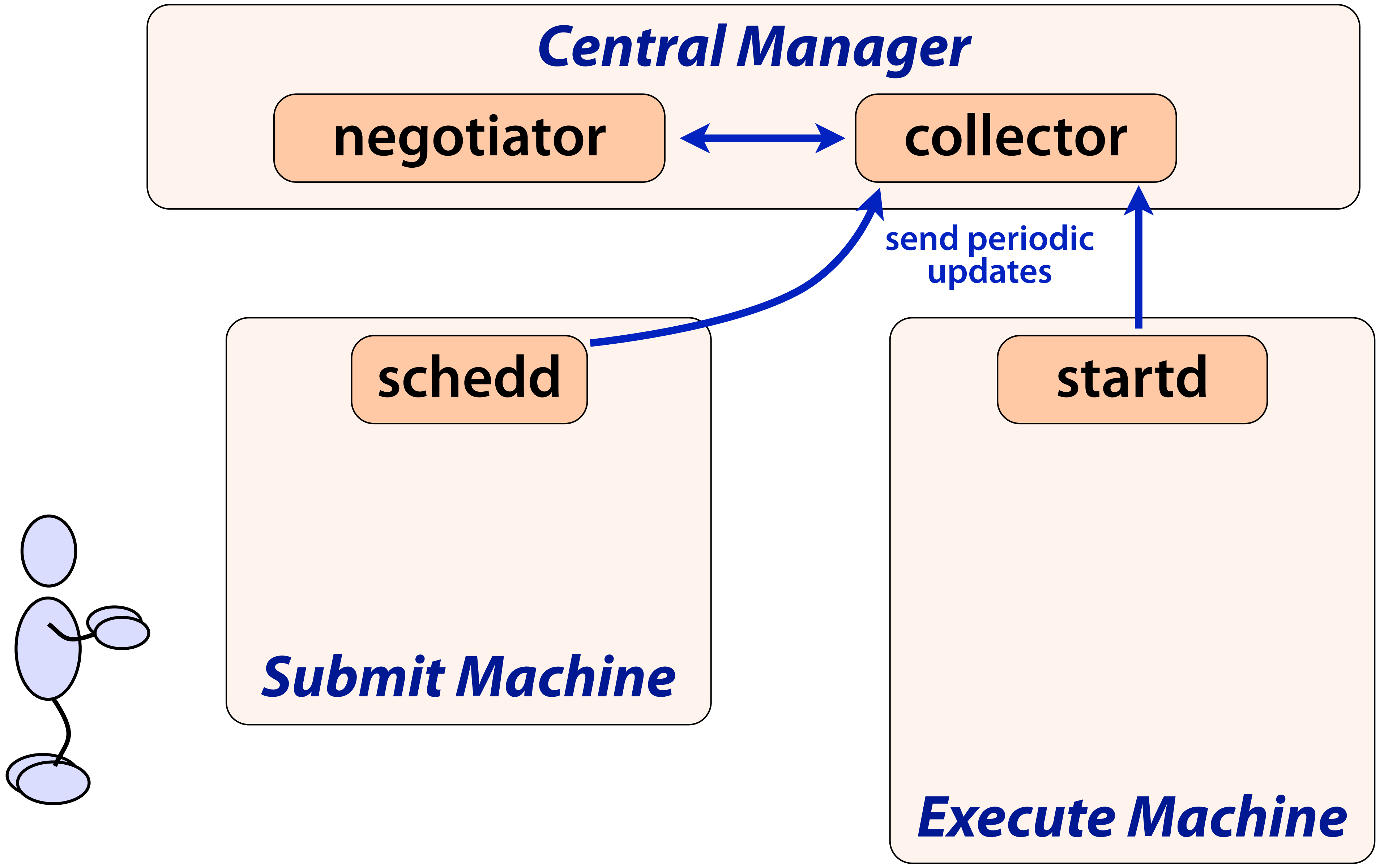
Typical Architecture



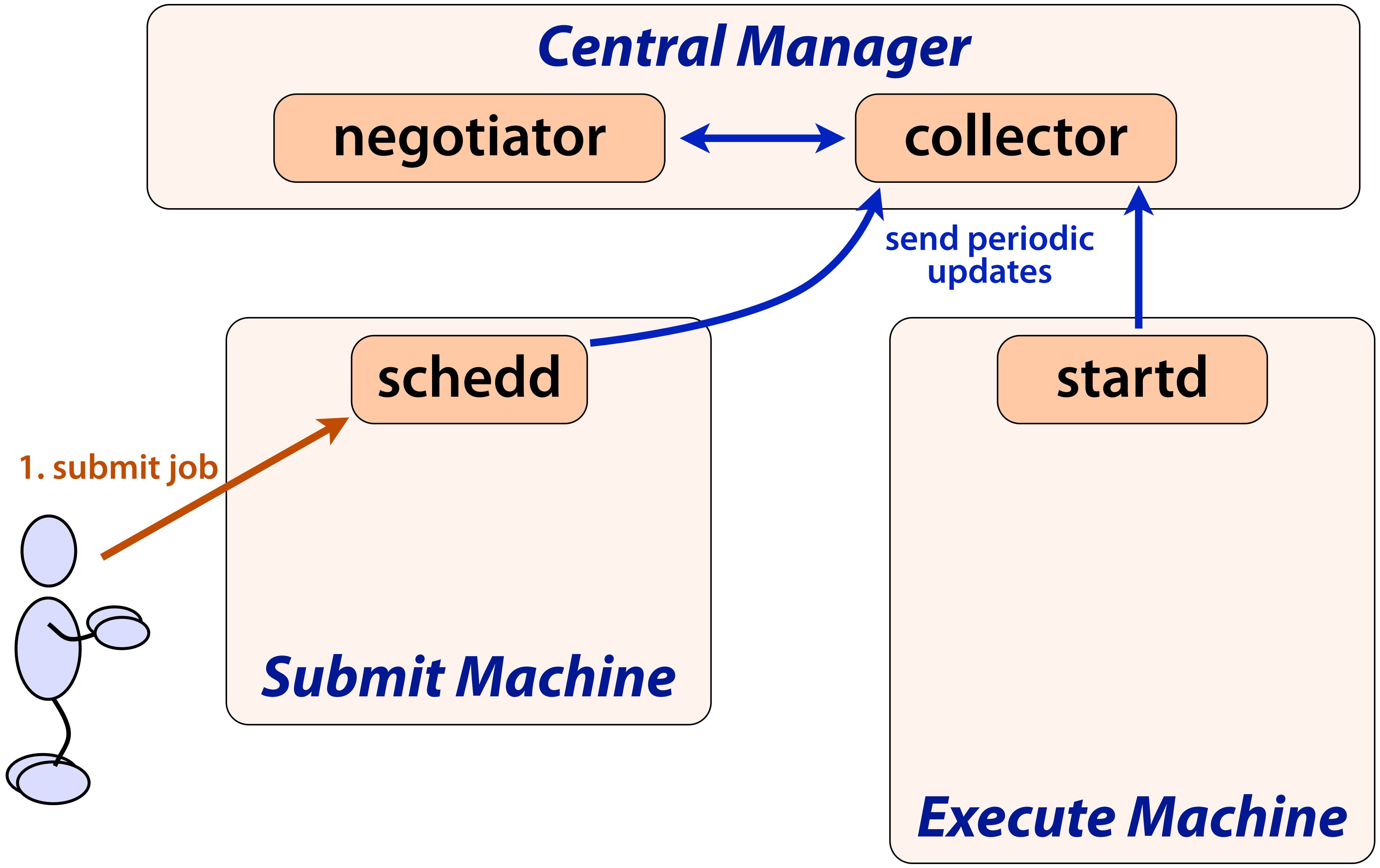
The Life of an HTCondor Job



The Life of an HTCondor Job

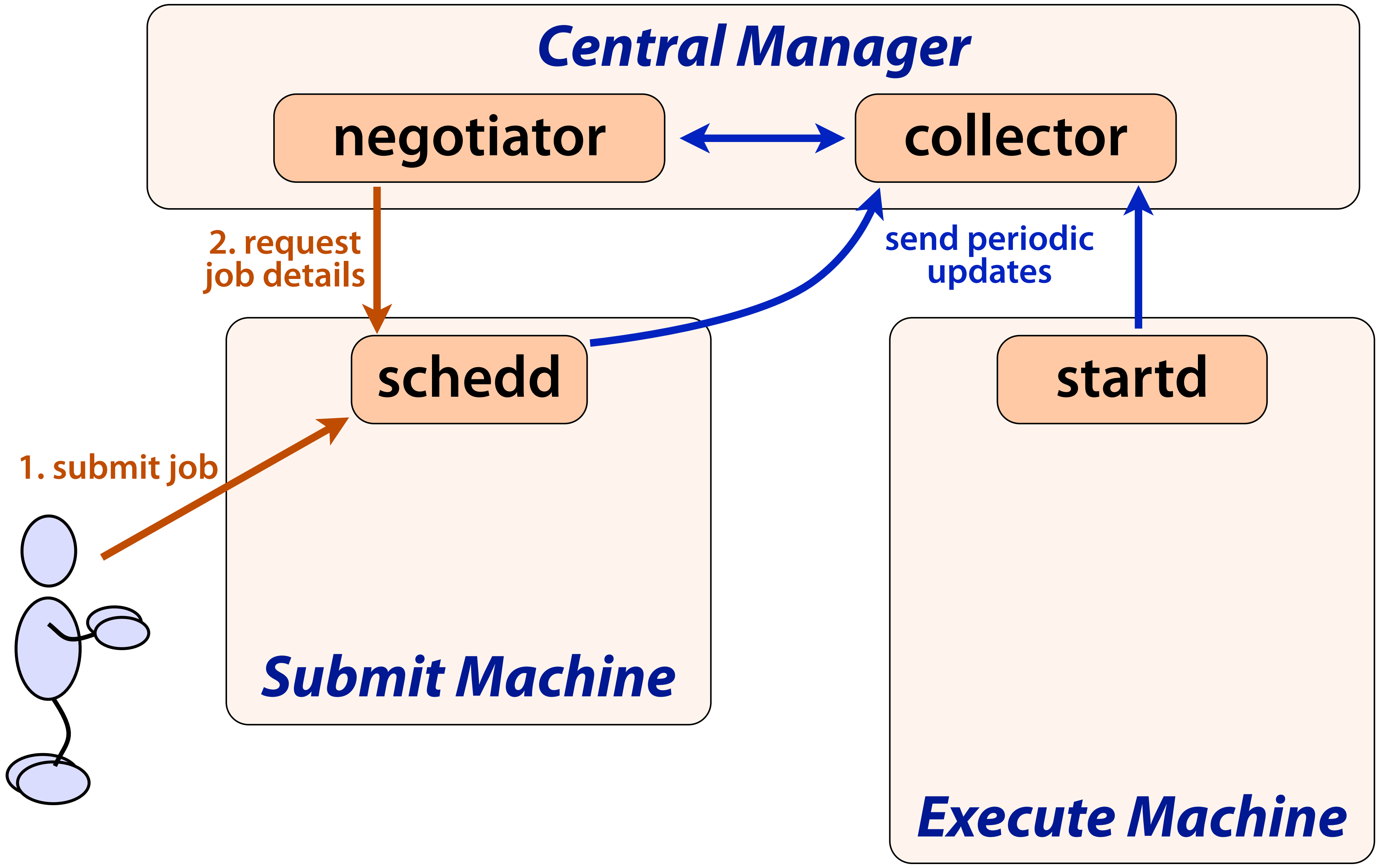


The Life of an HTCondor Job

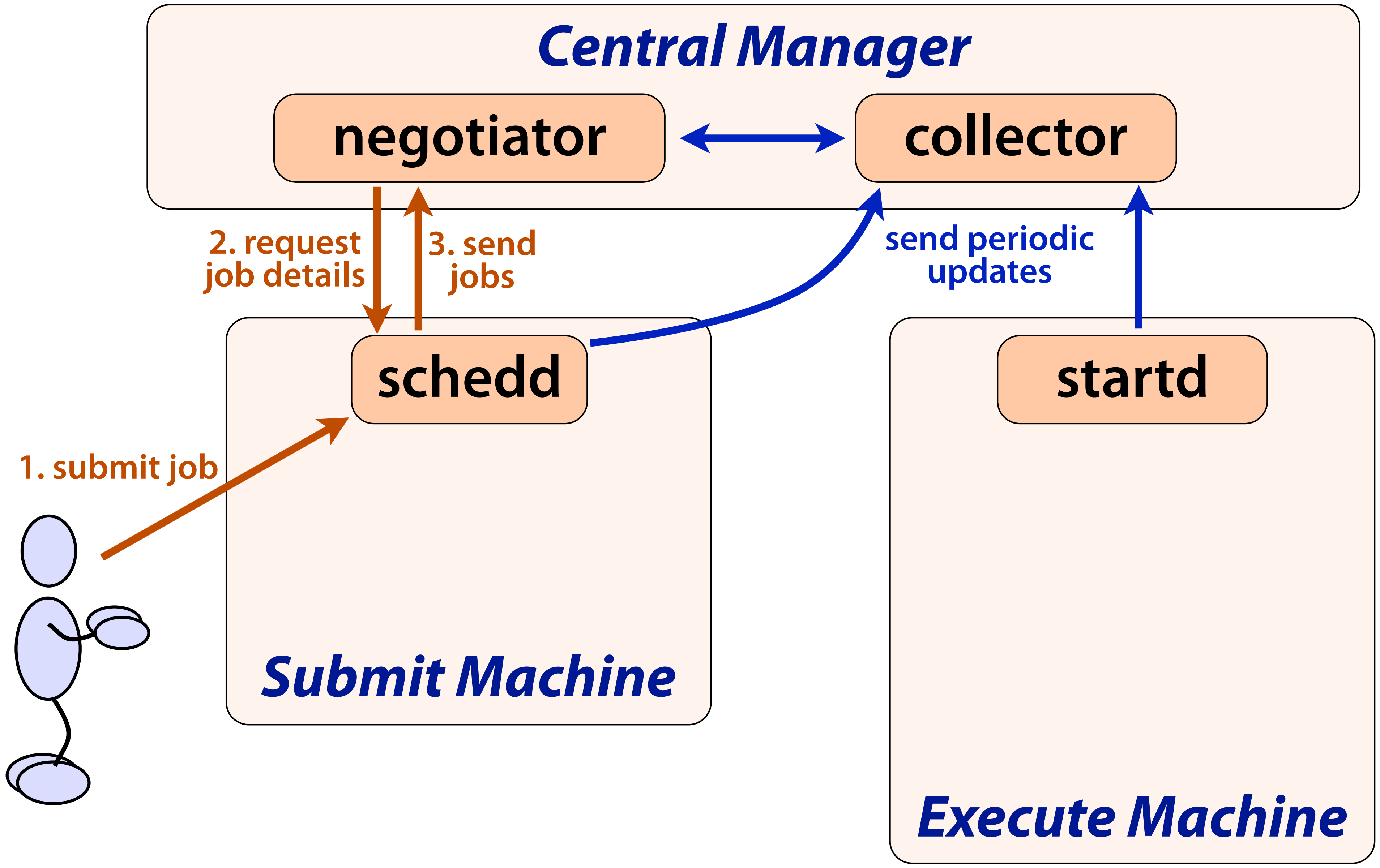


Open Science Grid

The Life of an HTCondor Job

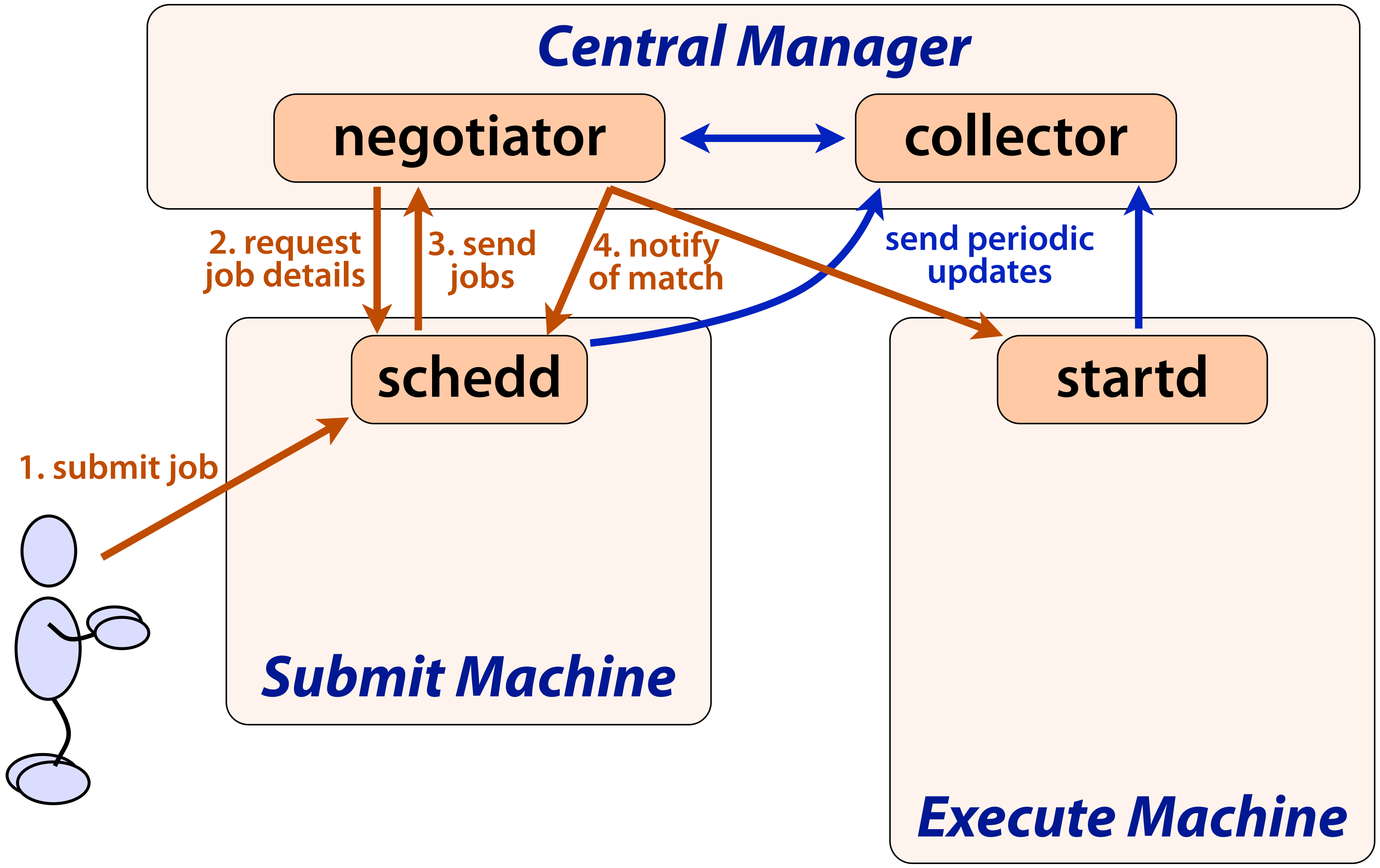


The Life of an HTCondor Job



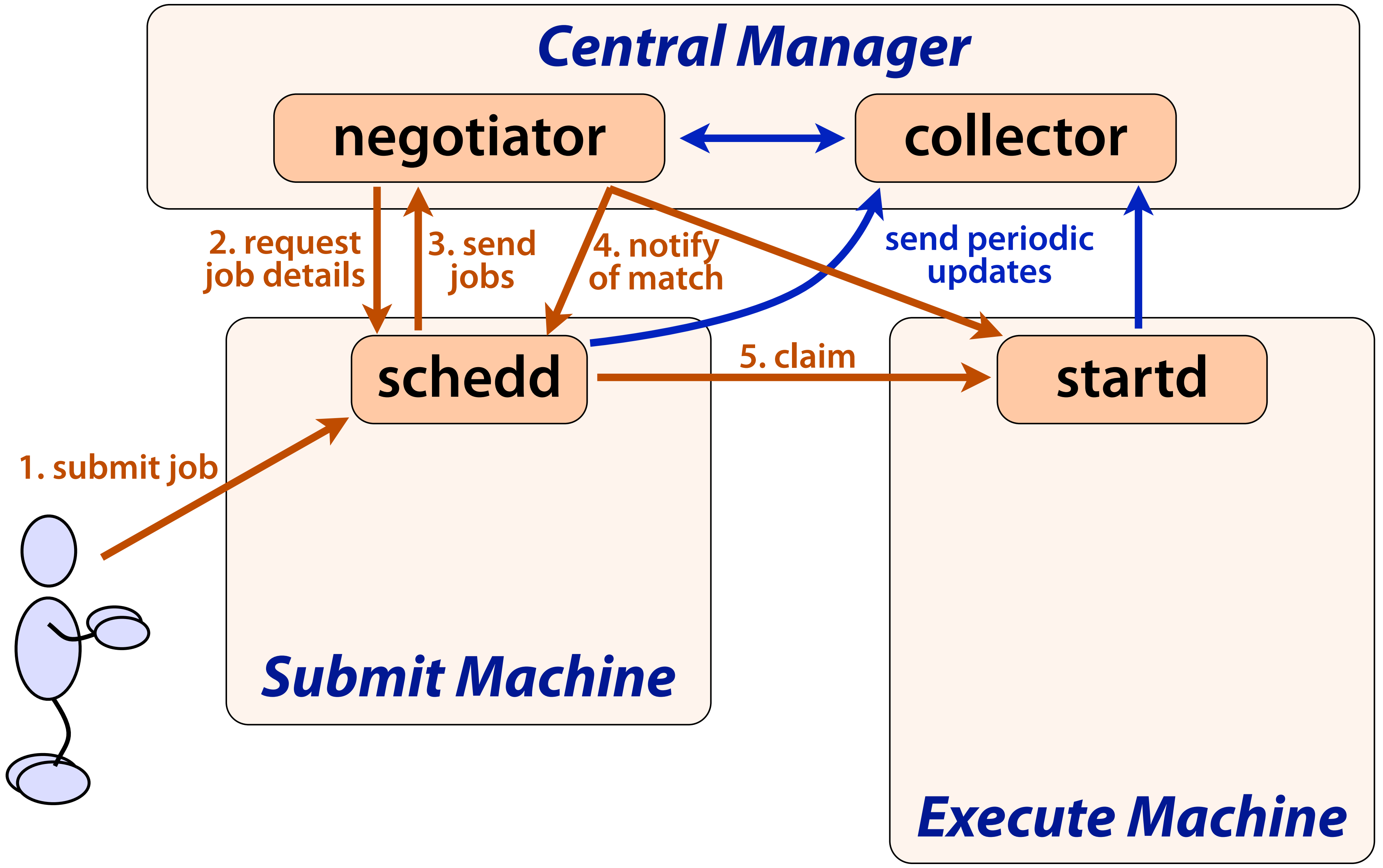
Open Science Grid

The Life of an HTCondor Job

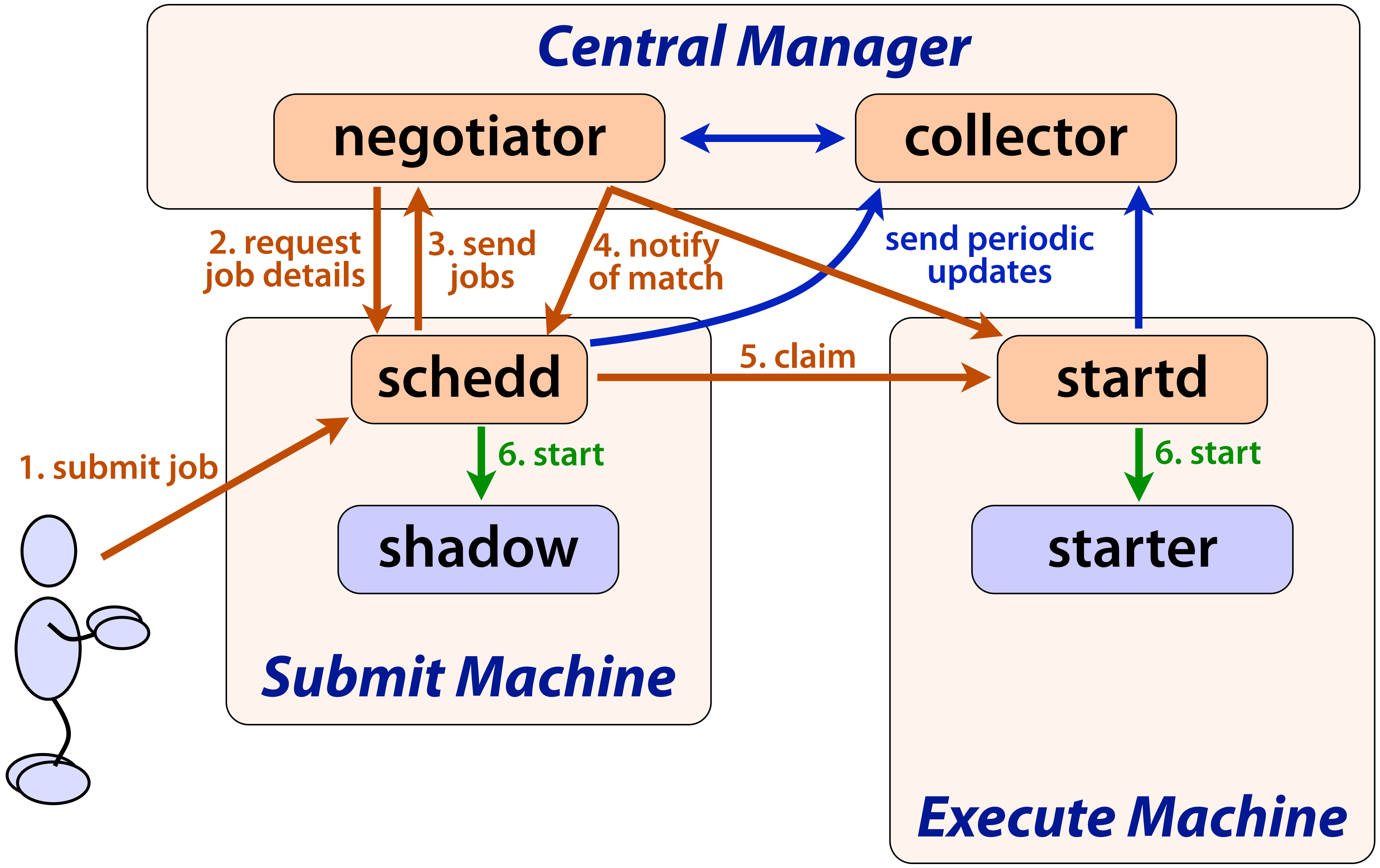


Open Science Grid

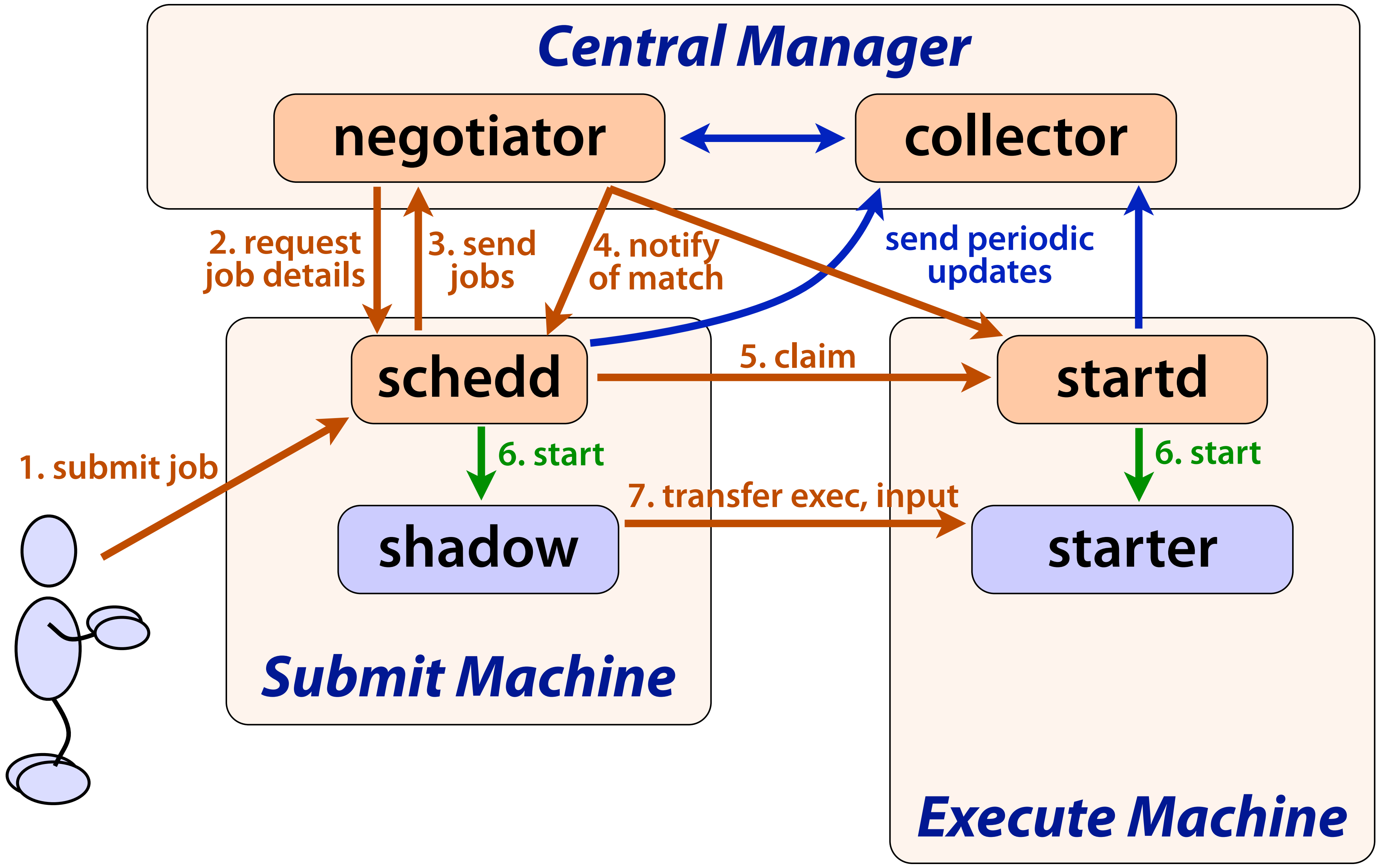
The Life of an HTCondor Job



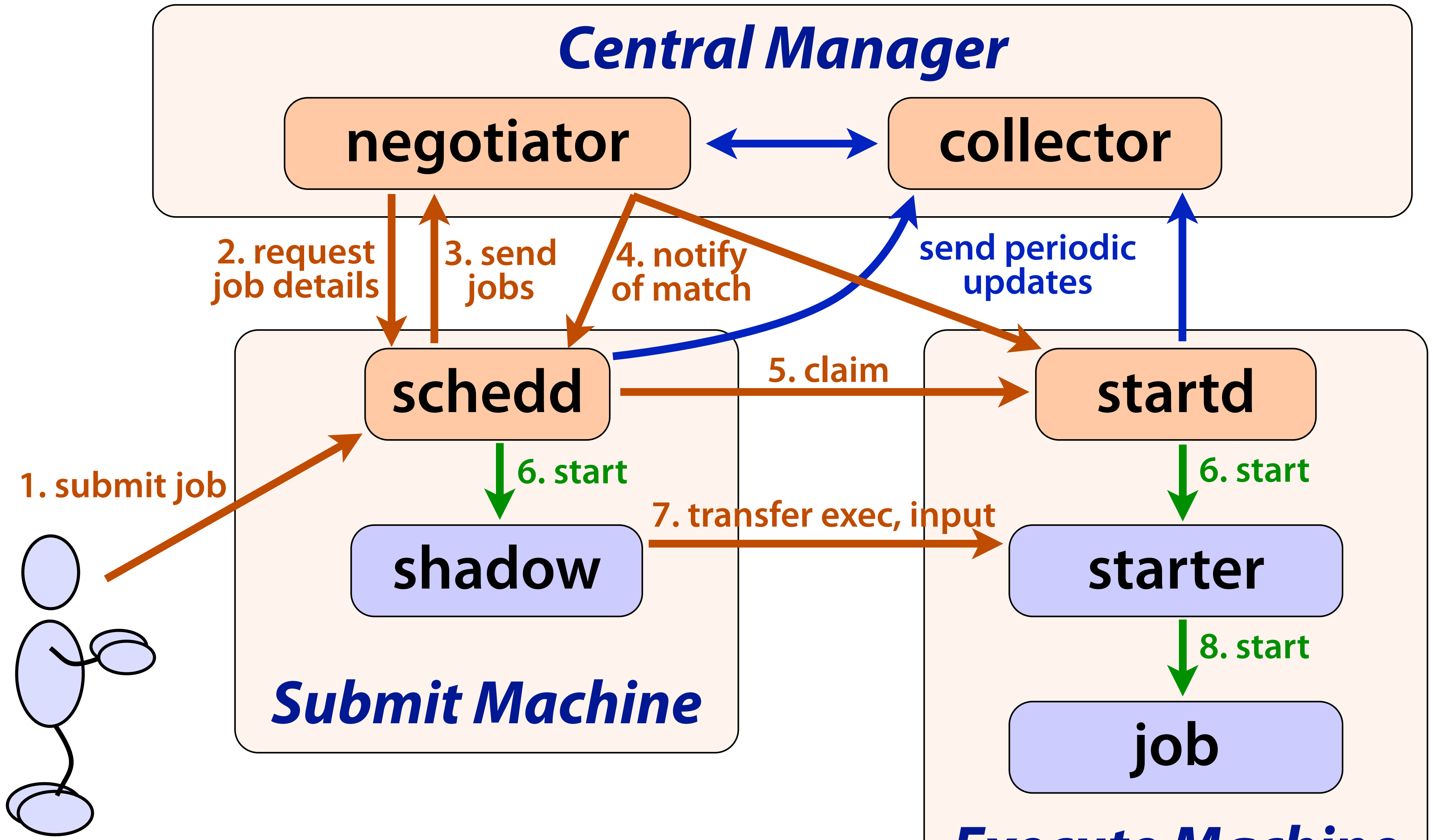
The Life of an HTCondor Job



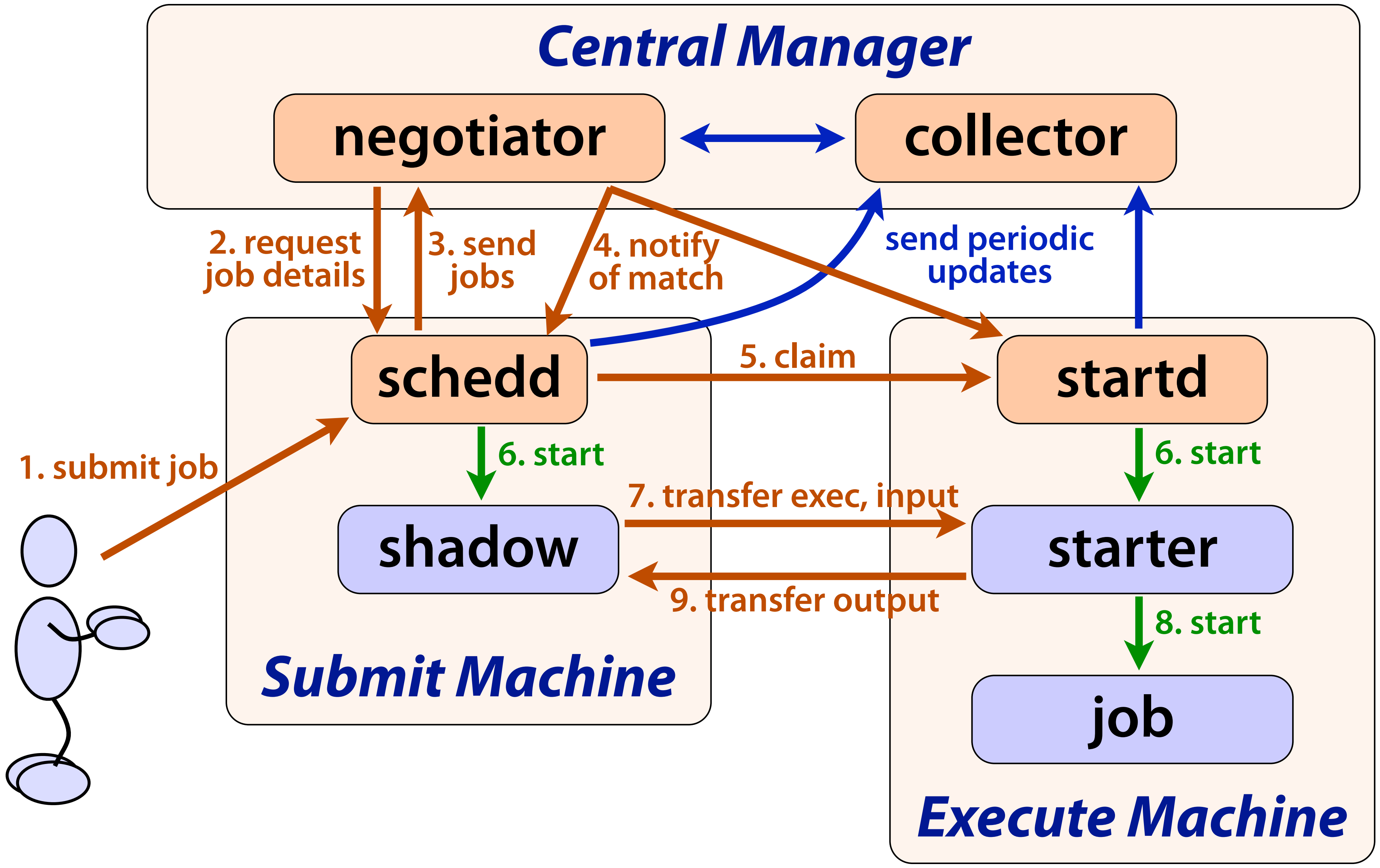
The Life of an HTCondor Job



The Life of an HTCondor Job



The Life of an HTCondor Job



Open Science Grid

Matchmaking Algorithm (sort of)

A. Gather lists of machines and waiting jobs

B. For each user:

   1. Compute maximum # of slots to allocate to user (the user's "fair share", a % of whole pool)
   2. For each job (up to user's maximum slots):
      a. Find all machines that are acceptable (i.e., machine ***and*** job requirements are met)
      b. If there are no acceptable machines, skip to next job
      c. Sort acceptable machines by job preferences
      d. Pick the best one
      e. Record match of job and slot

ClassAds

- In HTCondor, information about machines and jobs (and more) are represented by ClassAds
- You do not write ClassAds (much), but reading them may help understanding and debugging
- ClassAds can represent persistent facts, current state, preferences, requirements, …
- HTCondor uses a core of predefined attributes, but users can add other, new attributes, which can be used for matchmaking, reporting, etc.

Open Science Grid

Sample ClassAd Attributes

```
MyType = "Job"
TargetType = "Machine"
ClusterId = 14
Owner = "cat"
Cmd = "/.../test-job.py"
Requirements = (Arch == "X86_64") &&
               (OpSys == "LINUX") &&
               ...
Rank = 0.0
In = "/dev/null"
UserLog = "/.../test-job.log"
Out = "test-job.out"
Err = "test-job.err"
NiceUser = false
ShoeSize = 10
```

Open Science Grid

Sample ClassAd Attributes

string

```
MyType = "Job"
TargetType = "Machine"
ClusterId = 14
Owner = "cat"
Cmd = "/.../test-job.py"
Requirements = (Arch == "X86_64") &&
               (OpSys == "LINUX") &&
               ...
Rank = 0.0
In = "/dev/null"
UserLog = "/.../test-job.log"
Out = "test-job.out"
Err = "test-job.err"
NiceUser = false
ShoeSize = 10
```

Open Science Grid

Sample ClassAd Attributes

```
MyType = "Job"
TargetType = "Machine"
ClusterId = 14
Owner = "cat"
Cmd = "/.../test-job.py"
Requirements = (Arch == "X86_64") &&
               (OpSys == "LINUX") &&
               ...
Rank = 0.0
In = "/dev/null"
UserLog = "/.../test-job.log"
Out = "test-job.out"
Err = "test-job.err"
NiceUser = false
ShoeSize = 10
```

number

Open Science Grid

Sample ClassAd Attributes

```
MyType = "Job"
TargetType = "Machine"
ClusterId = 14
Owner = "cat"
Cmd = "/.../test-job.py"
Requirements = (Arch == "X86_64") &&
               (OpSys == "LINUX") &&
               ...
Rank = 0.0
In = "/dev/null"
UserLog = "/.../test-job.log"
Out = "test-job.out"
Err = "test-job.err"
NiceUser = false
ShoeSize = 10
```

operations/ expressions

Open Science Grid

Sample ClassAd Attributes

```
MyType = "Job"
TargetType = "Machine"
ClusterId = 14
Owner = "cat"
Cmd = "/.../test-job.py"
Requirements = (Arch == "X86_64") &&
               (OpSys == "LINUX") &&
               ...
Rank = 0.0
In = "/dev/null"
UserLog = "/.../test-job.log"
Out = "test-job.out"
Err = "test-job.err"
NiceUser = false
ShoeSize = 10
```

boolean

Open Science Grid

Sample ClassAd Attributes

```
MyType = "Job"
TargetType = "Machine"
ClusterId = 14
Owner = "cat"
Cmd = "/.../test-job.py"
Requirements = (Arch == "X86_64") &&
               (OpSys == "LINUX") &&
               ...
Rank = 0.0
In = "/dev/null"
UserLog = "/.../test-job.log"
Out = "test-job.out"
Err = "test-job.err"
NiceUser = false
ShoeSize = 10
```

arbitrary

Open Science Grid

HTCondor Universes

- Different combinations of configurations and features are bundled as ***universes***:

`vanilla`	A "normal" job; default, fine for today
`standard`	Supports checkpointing and remote I/O
`java`	Special support for Java programs
`parallel`	Supports parallel jobs (such as MPI)
`grid`	Submits to remote system (more tomorrow)
… and more!	

Open Science Grid

HTCondor Priorities

- **Job priority**
  - Set per job by the user (owner)
  - Relative to that user's other jobs
  - Set in submit file or change later with **`condor_prio`**
  - Higher number means run sooner
- **User priority**
  - Computed based on past usage
  - Determines user's "fair share" percentage of slots
  - Lower number means run sooner (0.5 is minimum)
- **Preemption**
  - Low priority jobs stopped for high priority ones (stopped jobs go back into the regular queue)
  - Governed by fair-share algorithm and pool policy
  - Not enabled on all pools

HTCondor Commands

Open Science Grid

List Jobs: **condor_q**

- Select jobs: by user (e.g., you), cluster, job ID
- Format output as you like
- View full ClassAd(s), typically 80–90 attributes (most useful when limited to a single job ID)
- Ask HTCondor why a job is not running
  - May not explain everything, but can help
  - Remember: Negotiation happens periodically
- Explore **condor_q** options in next exercises

Open Science Grid

List Slots: `condor_status`

- Select slots: available, host, specific slot
- Select slots by ClassAd expression
  E.g., slots with SL 6 (OS) and ≥ 10 GB memory
- Format output as you like
- View full ClassAd(s), typically 120–250 attributes
  (most useful when limited to a single slot)
- Explore **`condor_status`** options in exercises

Submit Files

Open Science Grid

Resource Requests

```
request_cpus = ClassAdExpression
request_disk = ClassAdExpression
request_memory = ClassAdExpression
```

- Ask for minimum resources of execute machine
- May be dynamically allocated (very advanced!)
- *Check job log for actual usage!!!*

```
request_disk = 2000000  # in KB by default
request_disk = 2GB      # KB, MB, GB, TB

request_memory = 2000   # in MB by default
request_memory = 2GB    # KB, MB, GB, TB
```

Open Science Grid

File Access in HTCondor

- **Option 1:** Shared filesystem
  - Easy to use (jobs just access files)
  - But, must exist and be ready handle load

```
should_transfer_files = NO
```

- **Option 2:** HTCondor transfers files for you
  - Must name all input files (except executable)
  - May name output files; defaults to all new/changed

```
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
transfer_input_files = a.txt, b.tgz
```

Email Notifications

```
notification = Always|Complete|Error|Never
```

- When to send email
  - `Always`: job checkpoints or completes
  - `Complete`: job completes (default)
  - `Error`: job completes with error
  - `Never`: do not send email

```
notify_user = email
```

- Where to send email
- Defaults to ***`user@submit-machine`***

Open Science Grid

Requirements and Rank

```
requirements = ClassAdExpression
```

- Expression must evaluate to **`true`** to match slot
- HTCondor adds defaults! Check ClassAds …
- See HTCondor Manual (esp. 2.5.2 & 4.1) for more

```
rank = ClassAdExpression
```

- Ranks matching slots in order by preference
- Must evaluate to a FP number, higher is better
  - False becomes 0.0, True becomes 1.0
  - Undefined or error values become 0.0
- Writing rank expressions is an art form

Open Science Grid

Arbitrary Attributes

```
+AttributeName = value
```

- Adds arbitrary attribute(s) to job's ClassAd
- Useful in (at least) 2 cases:
  - Affect matchmaking with special attributes
  - Report on jobs with specific attribute value
- Experiment with reporting during exercises!

Many Jobs Per Submit File, Pt. 1

- Can use **queue** statement many times
- Make changes between **queue** statements
  - Change **arguments**, **log**, **output**, input files, …
  - Whatever is not explicitly changed remains the same

```
executable = test.py
...
log          = test.log

output    = test-1.out
arguments = "test-input.txt 42"
queue

output    = test-2.out
arguments = "test-input.txt 43"
queue
```

Open Science Grid

Many Jobs Per Submit File, Pt. 1

- Can use **queue** statement many times
- Make changes between **queue** statements
  - Change **arguments**, **log**, **output**, input files, …
  - Whatever is not explicitly changed remains the same

```
executable = test.py
...
log          = test.log

output    = test-1.out
arguments = "test-input.txt 42"
queue

output    = test-2.out
arguments = "test-input.txt 43"
queue
```

`log = test.log` (still)

Many Jobs Per Submit File, Pt. 2

```
queue N
```

- Submits ***N*** copies of the job
  - One cluster number for all copies, just as before
  - Process numbers go from 0 to (***N***–1)
- What good is having ***N*** copies of the same job?
  - Randomized processes (e.g., Monte Carlo)
  - Job fetches work description from somewhere else
  - But what about overwriting output files, etc.?
- Wouldn’t it be nice to have different files and/or arguments automatically applied to each job?

Open Science Grid

Separating Files by Run

```
output = program.out.$(Cluster).$(Process)
```

- Can use these variables anywhere in submit file
  - Often used in `output`, `error`, and `log` files
- Maybe use `$(Process)` in arguments?
  - Can't perform math on values; code must accept as is

```
...

output = test.$(Cluster)_$(Process).out
log    = test.$(Cluster)_$(Process).log

arguments = "test-input.txt $(Process)"
queue 10
```

Open Science Grid

Separating Directories by Run

```
initialdir = path
```

- Use ***path*** (instead of submit dir.) to locate files
  - E.g.: **`output`**, **`error`**, **`log`**, **`transfer_input_files`**
  - *Not* **`executable`**; it is still relative to submit directory
- Use **`$(Process)`** to separate all I/O by job ID

```
initialdir = run-$(Process)
transfer_input_files = input-$(Process).txt
output = test.$(Cluster)-$(Process).out
log    = test.$(Cluster)-$(Process).log

arguments = "input-$(Process).txt $(Process)"
queue 10
```

Your Turn!

Open Science Grid

Exercises!

- Ask questions!
- Lots of instructors around
- Reminder: Get your X.509 certificate today!
- Coming next:

Now – 12:15	Hands-on exercises
12:15–1:15	Lunch
1:15–5:30	Afternoon sessions with Zach