Condor-G Stork and DAGMan

An Introduction

Condor Project
Computer Sciences Department
University of Wisconsin-Madison
condor-admin@cs.wisc.edu
http://www.cs.wisc.edu/condor

Outline

- Background and principals
- The Story of Frieda, the scientist
  - Using Condor-G to manage jobs
  - Using DAGMan to manage dependencies
  - Condor-G Architecture and Mechanisms
    - Globus Universe
    - Glide-In
  - Using Stork to manage Data Placement jobs
- Future and advanced topics

Claims for "benefits" provided by Distributed Processing Systems

- High Availability and Reliability
- High System Performance
- Ease of Modular and Incremental Growth
- Automatic Load and Resource Sharing
- Good Response to Temporary Overloads
- Easy Expansion in Capacity and/or Function

"What is a Distributed Data Processing System?" , P.H. Enslow, Computer, January 1978

Benefits to Science

- Democratization of Computing – "you do not have to be a SUPER person to do SUPER computing." (accessibility)
- Speculative Science – "Since the resources are there, lets run it and see what we get." (unbounded computing power)
- Function shipping – "Find the image that has a red car in this 3 TB collection." (computational mobility)

The Ethernet Protocol

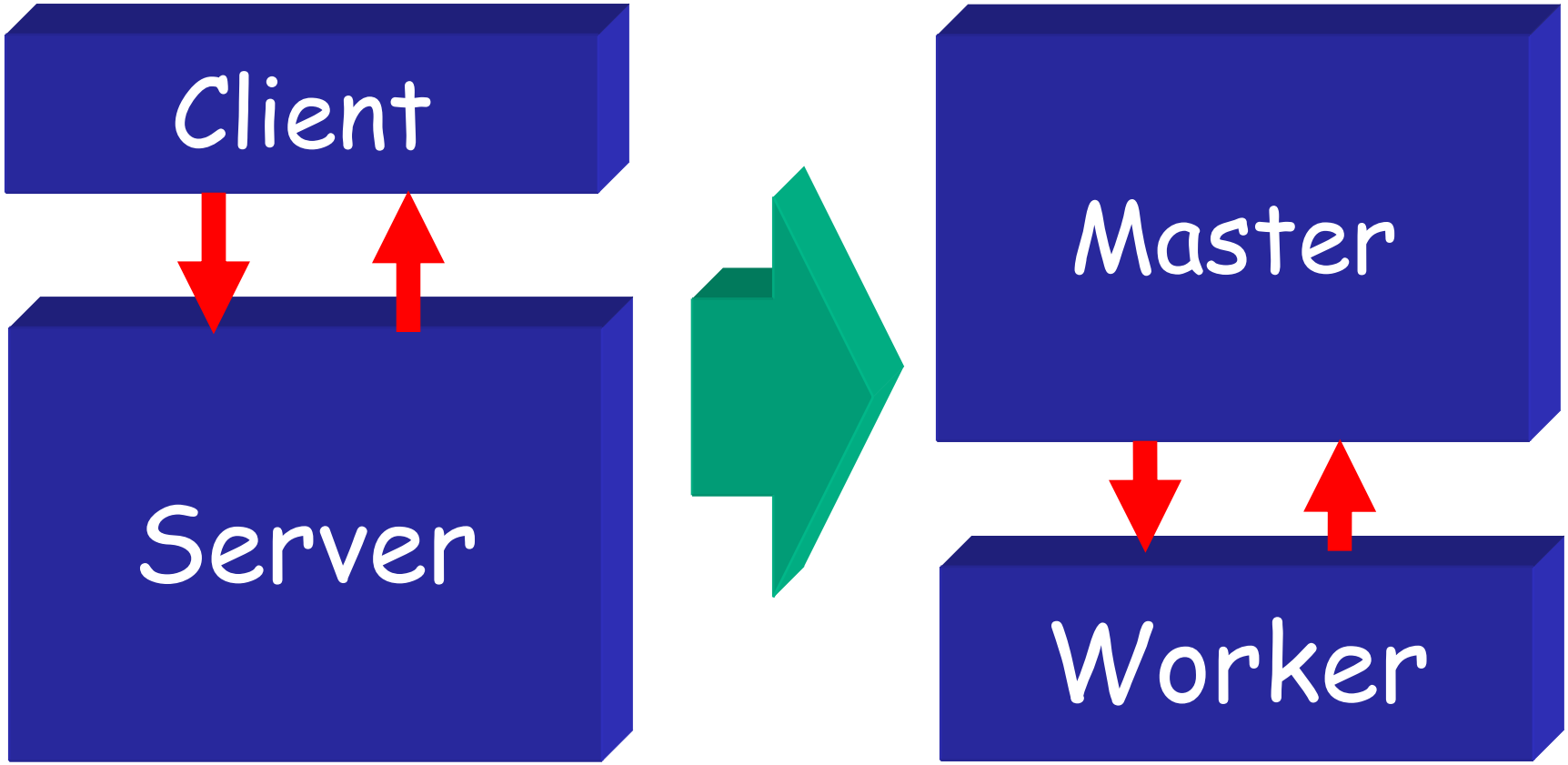
IEEE 802.3 CSMA/CD - A truly distributed (and very effective) access control protocol to a shared service.

- Client responsible for access control
- Client responsible for error detection
- Client responsible for fairness


desktop

WWW

Grid



Being a Master

Customer "delegates" task(s) to the master that is responsible for:

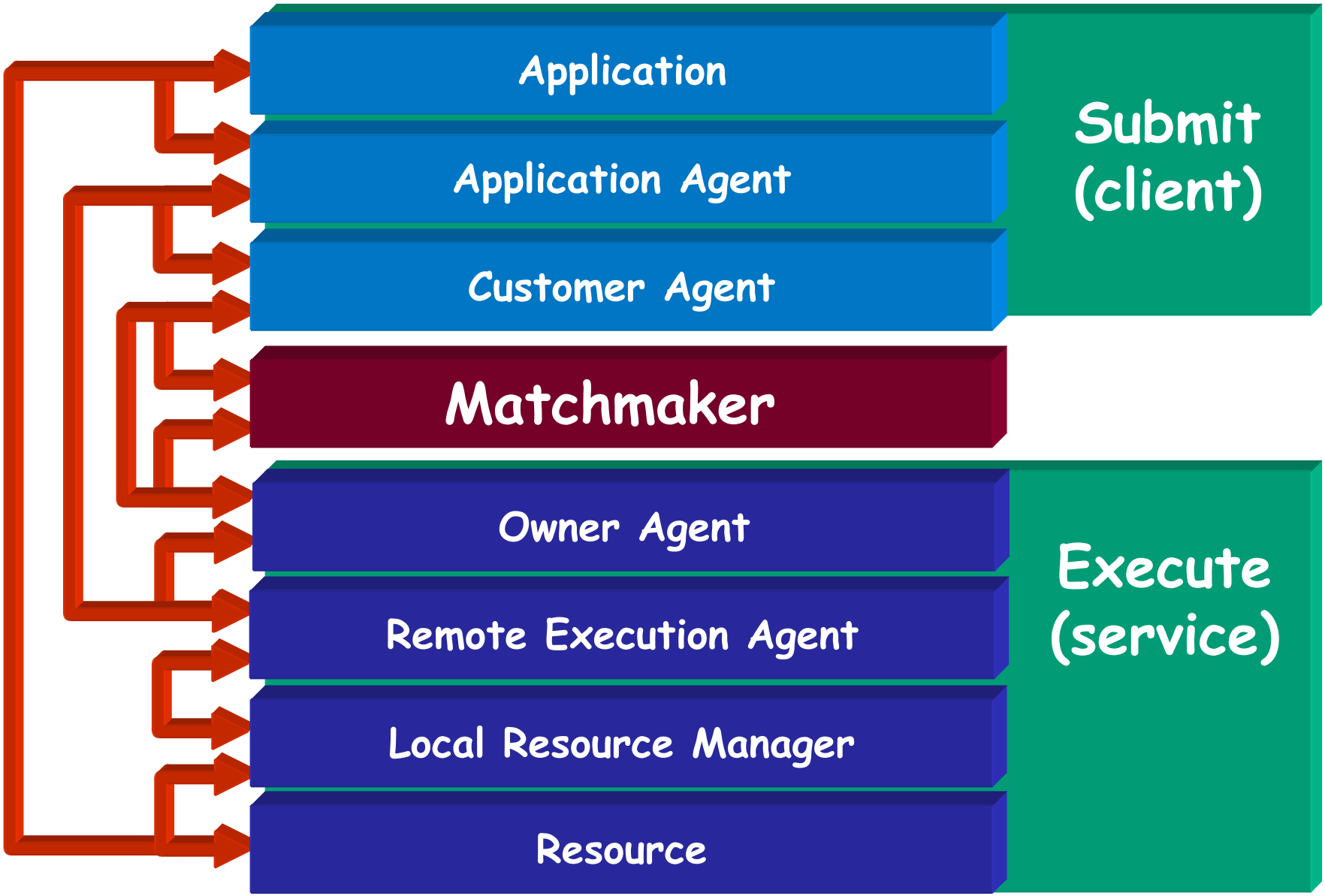
- Obtaining resources and/or workers
- Deploying and managing workers on obtained resources
- Assigning and delivering work unites to obtained/deployed workers
- Receiving and processing results
- Notify customer.

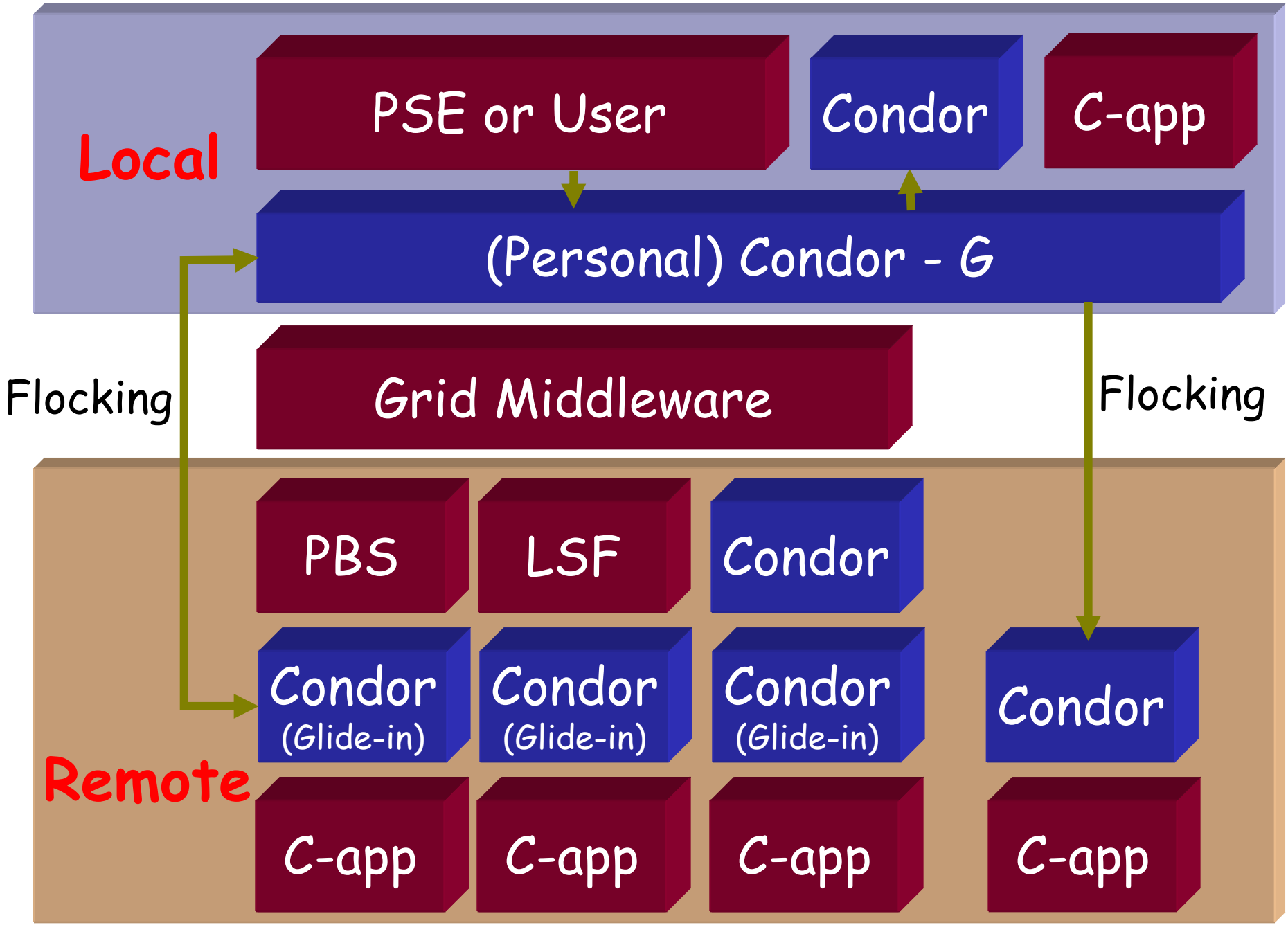
Application Responsibilities

- Use algorithms that can generate very large numbers of independent tasks – "*use pleasantly parallel algorithms*"
- Implement self-contained portable workers – "*this code can run anywhere*!"
- Detect failures and react gracefully – "*use exponential back off, please*!"
- Be well informed and opportunistic – "*get your work done and out of the way*!"

our
answer to
High Throughput MW Computing
on commodity resources

The Layers of Condor




Local
PSE or User
Condor
C-app
(Personal) Condor - G
Flocking
Grid Middleware
Flocking
PBS
LSF
Condor
Condor
(Glide-in)
Condor
(Glide-in)
Condor
(Glide-in)
Condor
Remote
C-app
C-app
C-app
C-app

The NUG30 Quadratic Assignment Problem (QAP)

Solved!

$$\min_{p \in \Pi} \sum_{i=1}^{30} \sum_{j=1}^{30} a_{ij} b_{p(i)p(j)}$$

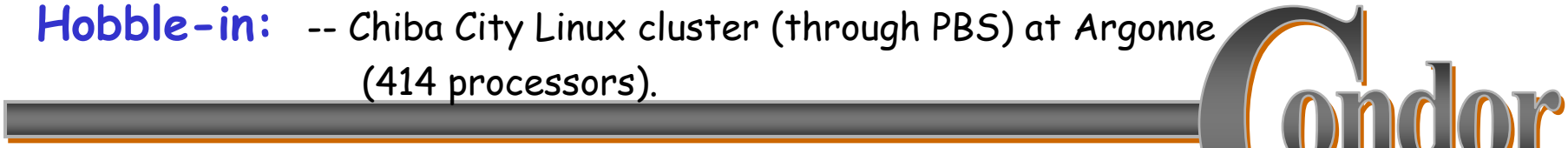
NUG30 Personal Grid …

Managed by one Linux box at Wisconsin

Flocking:
-- Condor pool at Wisconsin (500 processors)
-- Condor pool at Georgia Tech (284 Linux boxes)
-- Condor pool at UNM (40 processors)
-- Condor pool at Columbia (16 processors)
-- Condor pool at Northwestern (12 processors)
-- Condor pool at NCSA (65 processors)
-- Condor pool at INFN Italy (54 processors)

Glide-in:
-- Origin 2000 (through LSF) at NCSA. (512 processors)
-- Origin 2000 (through LSF) at Argonne (96 processors)

Hobble-in:
-- Chiba City Linux cluster (through PBS) at Argonne (414 processors).



Solution Characteristics.

Scientists	4
Wall Clock Time	6:22:04:31
Avg. # CPUs	653
Max. # CPUs	1007
Total CPU Time	Approx. 11 years
Nodes	11,892,208,412
LAPs	574,254,156,532
Parallel Efficiency	92%

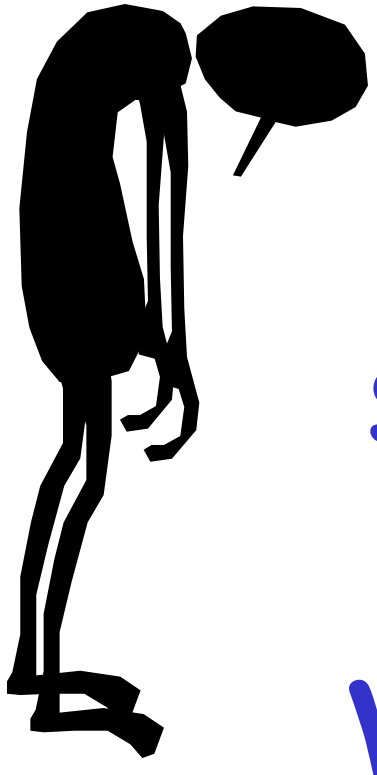
Meet Frieda

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

Frieda's Application

Simulate the behavior 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" (500MB) amount of memory
- **F** performs "moderate" I/O - (*x,y,z*) is 5 MB and **F**(*x,y,z*) is 50 MB



Frieda has 600 simulations to run.

Where can she get help?

Condor-G: Condor+Globus (and more)

Globus

- middleware deployed across entire Grid
- remote secure access to computational resources
- dependable, robust data transfer

Condor

- job scheduling across multiple resources
- strong fault tolerance with checkpointing and migration
- layered over grid middleare as "personal batch system" for a grid

Installing Condor-G

- Get Condor from the UW web site: http://www.cs.wisc.edu/condor
  - Condor-G is "included" as Globus Universe.

-- OR --

- Install from NMI: http://www.nsf-middleware.org

-- OR --

- Install from VDT: http://www.griphyn.org/vdt
- Condor-G can be installed on your own workstation, no root access required, no system administrator intervention needed

Condor-G will ...

- ... keep an eye on your jobs and will keep you posted on their progress
- ... implement your policies for the execution order of your jobs
- ... keep a log of your job activities
- ... add fault tolerance to your jobs
- ... implement your policies on how your jobs respond to grid and execution failures

Other Remote Submission

Condor-G can also "talk" other protocols besides GRAM (2.4)

- GRAM (3.2) (prototype)
- Oracle
- PBS (prototype)
- Condor (prototype)
- NorduGrid (prototype)
- LSF (in development)

Getting Started: Submitting Jobs to Condor-G

- Make your job "grid-ready"
- Get permission to run jobs on a grid site.
- Create a *submit description* file
- Run *condor_submit* on your submit description file

Making your job grid-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

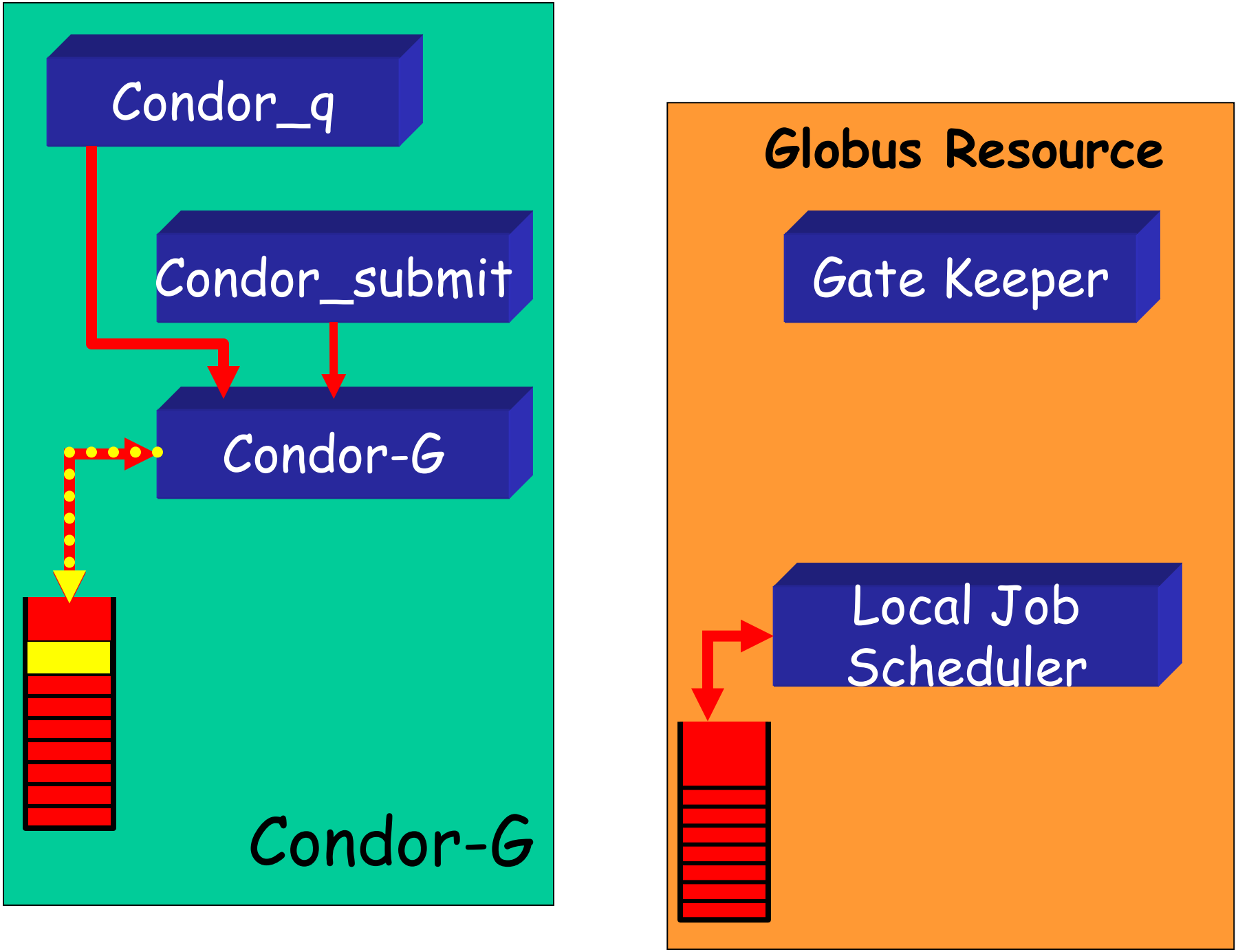
- A plain ASCII text file
- Tells Condor-G about your job:
  - Which executable, grid site, input, output and error files to use, command-line arguments, environment variables, etc.
- Can describe many jobs at once (a "cluster") each with different input, arguments, output, etc.

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   = globus
GlobusScheduler = host.domain.edu/jobmanager
Executable = my_job
Queue
```

Running condor_submit

- You give *condor_submit* the name of the submit file you have created
- *condor_submit* parses the file, checks for errors, and creates a "ClassAd" that describes your job(s)
- Sends your job's ClassAd(s) and executable to the Condor-G *schedd*, which stores the job in its queue
  - Atomic operation, two-phase commit
- View the queue with *condor_q*


Condor_q
Condor_submit
Condor-G
Condor-G
Globus Resource
Gate Keeper
Local Job Scheduler

Running condor_submit

```
% condor_submit my_job.submit-file
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

%
```

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   = globus
GlobusScheduler = host.domain.edu/jobmanager
Executable = /home/wright/condor/my_job.condor
Input      = my_job.stdin
Output     = my_job.stdout
Error      = my_job.stderr
Arguments  = -arg1 -arg2
InitialDir = /home/wright/condor/run_1
Queue
```

Using condor_rm

- If you want to remove a job from the Condor-G 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 specify specific job ID's, or you can remove all of your jobs with the "-a" option.

Temporarily halt a Job

- Use *condor_hold* to place a job on hold
  - Kills job if currently running
  - Will not attempt to restart job until released
  - Sometimes Condor-G will place a job on hold itself ("system hold") due to grid problems.
- Use *condor_release* to remove a hold and permit job to be scheduled again

Using condor_history

- Once your job completes, it will no longer show up in *condor_q*
- You can use *condor_history* to view information about a completed job
- The status field ("ST") will have either a "C" for "completed", or an "X" if the job was removed with *condor_rm*

Getting Email from Condor-G

- By default, Condor-G will send you email when your jobs completes
  - With lots of information about the run
- If you don't want this email, put this in your submit file:

```
notification = never
```

- If you want email every time something happens to your job (failure, exit, etc), use this:

```
notification = always
```

Getting Email from Condor-G

- If you only want email in case of errors, use this:

```
notification = error
```

- By default, the email is sent to your account on the host you submitted from. If you want the email to go to a different address, use this:

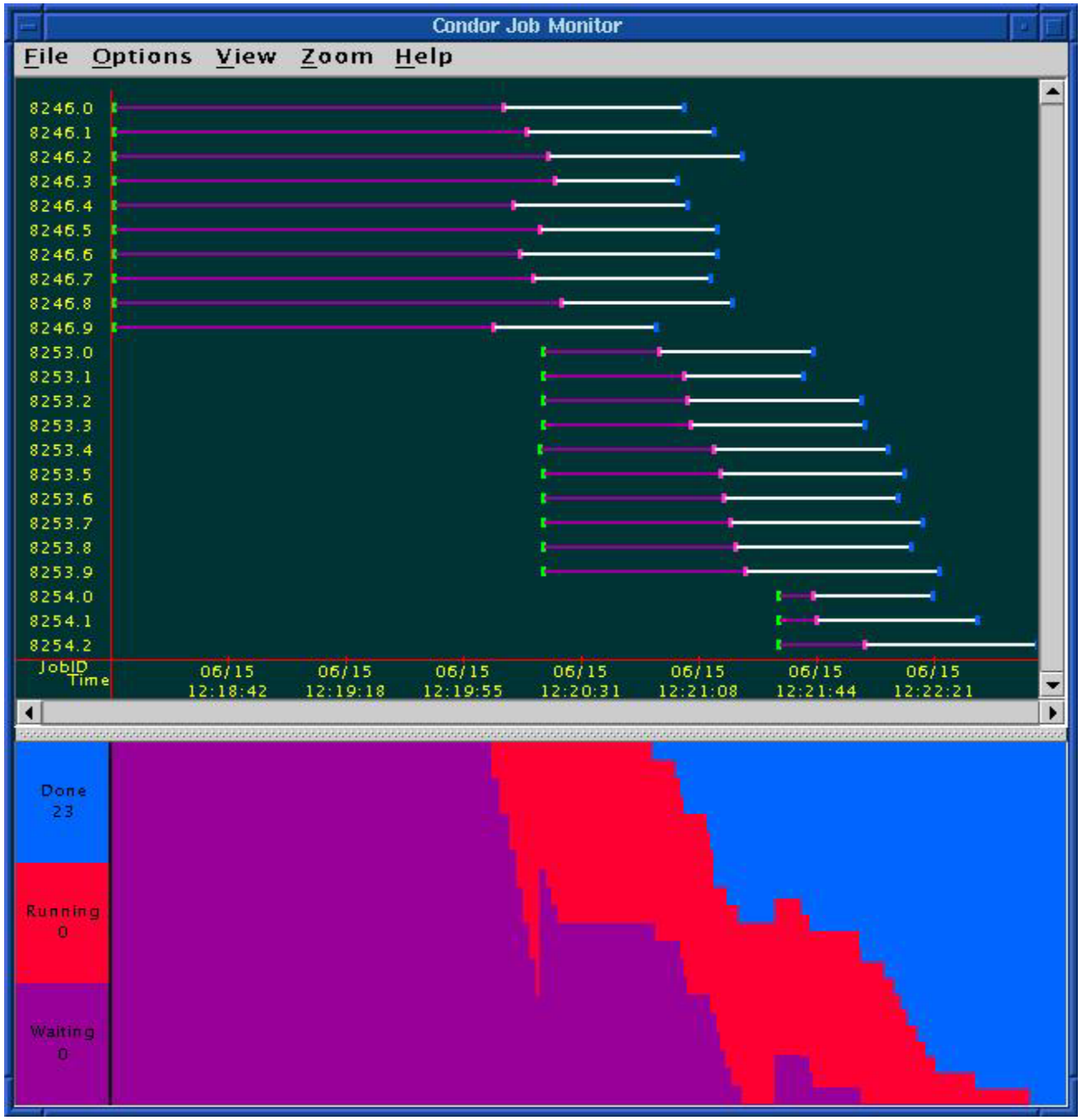
```
notify_user = email@address.here
```

A Job's life story: The "User Log" file

- A UserLog must be specified in your submit file:
  - Log = filename
- You get a log entry for everything that happens to your job:
  - When it was submitted to Condor-G, when it was submitted to the remote Globus jobmanager, when it starts executing, completes, if there are any problems, etc.
- Very useful! Highly recommended!

Uses for the User Log

- Easily read by human or machine
  - C++ library and Perl Module for parsing UserLogs is available
- Event triggers for meta-schedulers
  - Like DAGMan…
- Visualizations of job progress
  - Condor-G JobMonitor Viewer



Condor-G JobMonitor Screenshot

Want other Scheduling possibilities? Use the Scheduler Universe

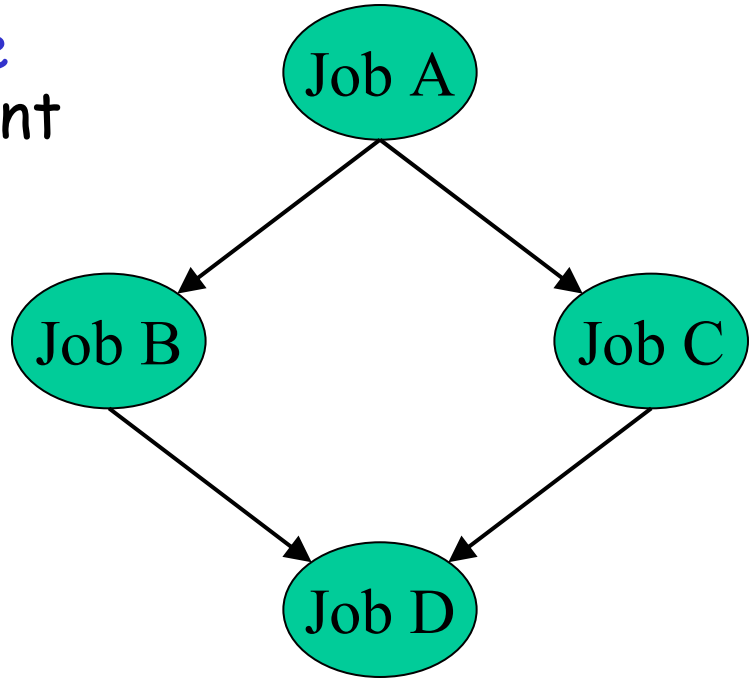
- In addition to Globus, another job universe is the *Scheduler Universe*.
- Scheduler Universe jobs run on the submitting machine.
- Can serve as a meta-scheduler.
- DAGMan meta-scheduler included

DAGMan

- **Directed Acyclic Graph Manager**
- DAGMan allows you to specify the *dependencies* between your Condor-G jobs, so it can *manage* them automatically for you.
- (e.g., "Don't run job "B" until job "A" has completed successfully.")

What is a DAG?

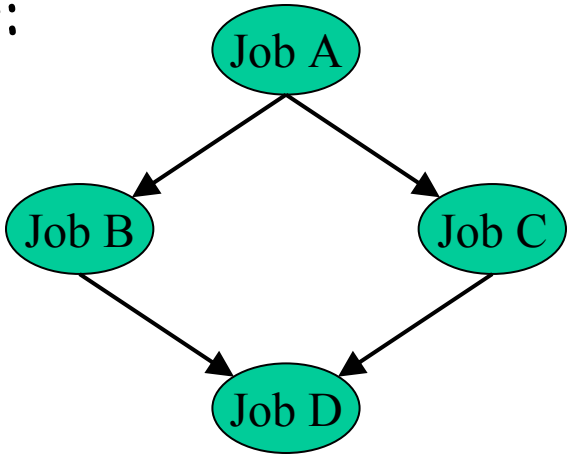
- A DAG is the data structure used by DAGMan to represent these dependencies.
- Each job is a "node" in the DAG.
- Each node can have any number of "parent" or "children" nodes – as long as there are no loops!



Defining a DAG

- A DAG is defined by a *.dag file*, listing each of its nodes and their dependencies:

```
# diamond.dag
Job A a.sub
Job B b.sub
Job C c.sub
Job D d.sub
Parent A Child B C
Parent B C Child D
```



- each node will run the Condor-G job specified by its accompanying Condor submit file

Submitting a DAG

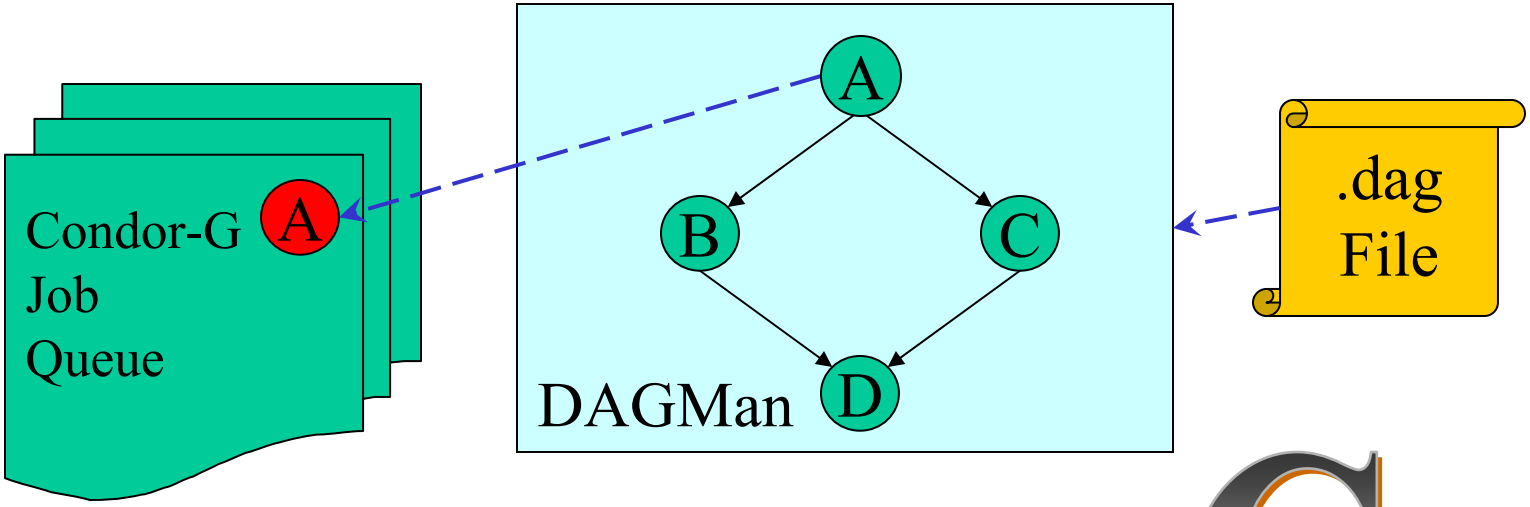
- To start your DAG, just run ***condor_submit_dag*** with your .dag file, and Condor will start a personal DAGMan daemon which to begin running your jobs:

```
% condor_submit_dag diamond.dag
```

- condor_submit_dag submits a Scheduler Universe Job with DAGMan as the executable.
- Thus the DAGMan daemon itself runs as a Condor-G scheduler universe job, so you don't have to baby-sit it.

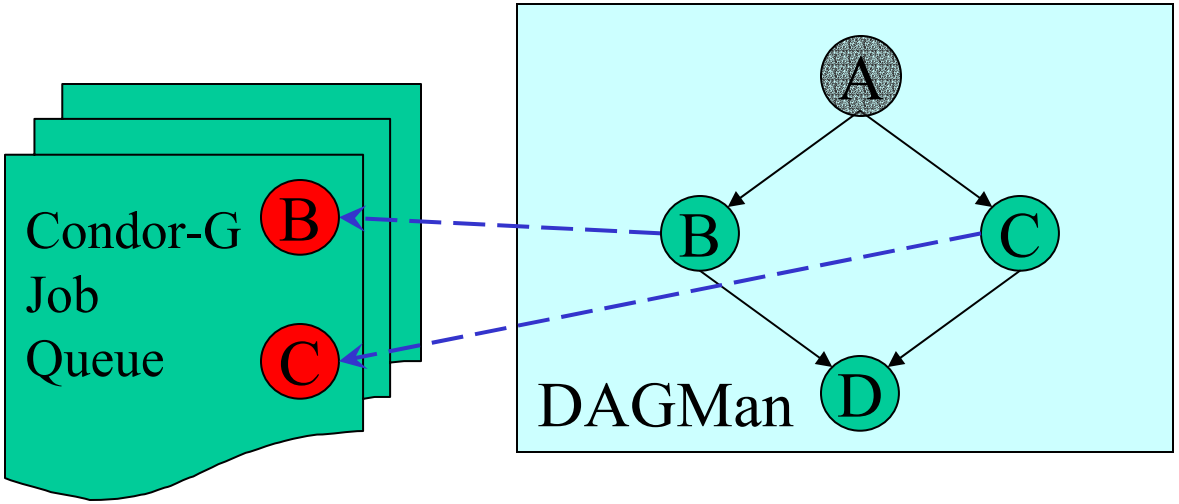
Running a DAG

- DAGMan acts as a "meta-scheduler", managing the submission of your jobs to Condor-G based on the DAG dependencies.



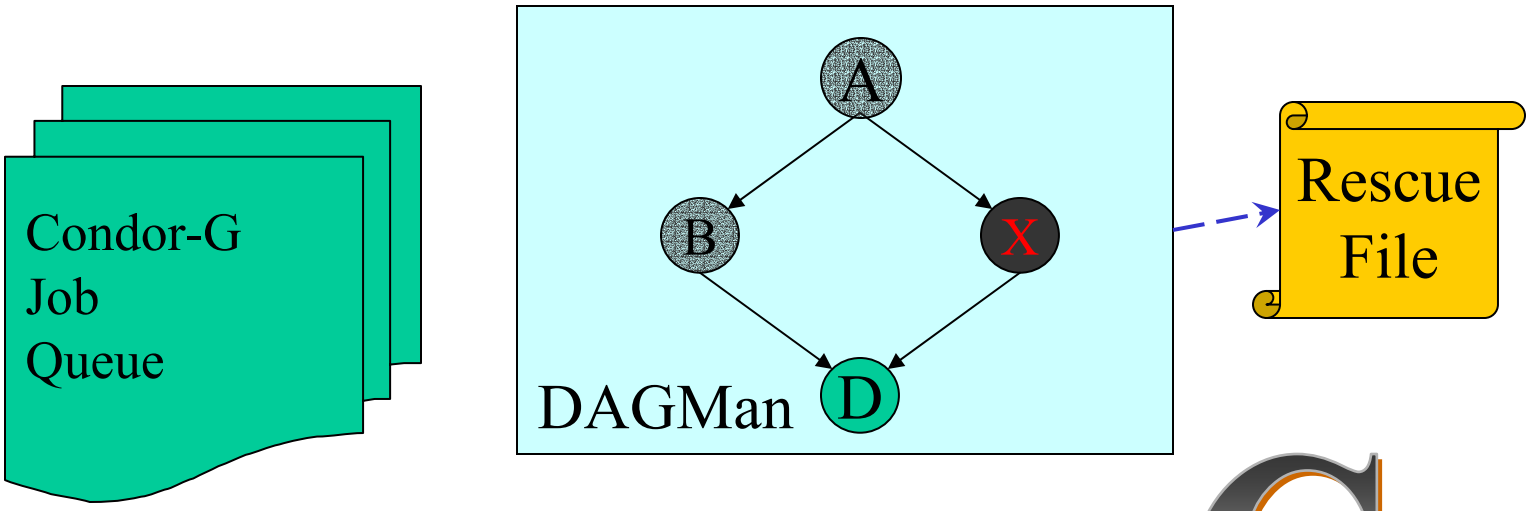
Running a DAG (cont'd)

- DAGMan holds & submits jobs to the Condor-G queue at the appropriate times.



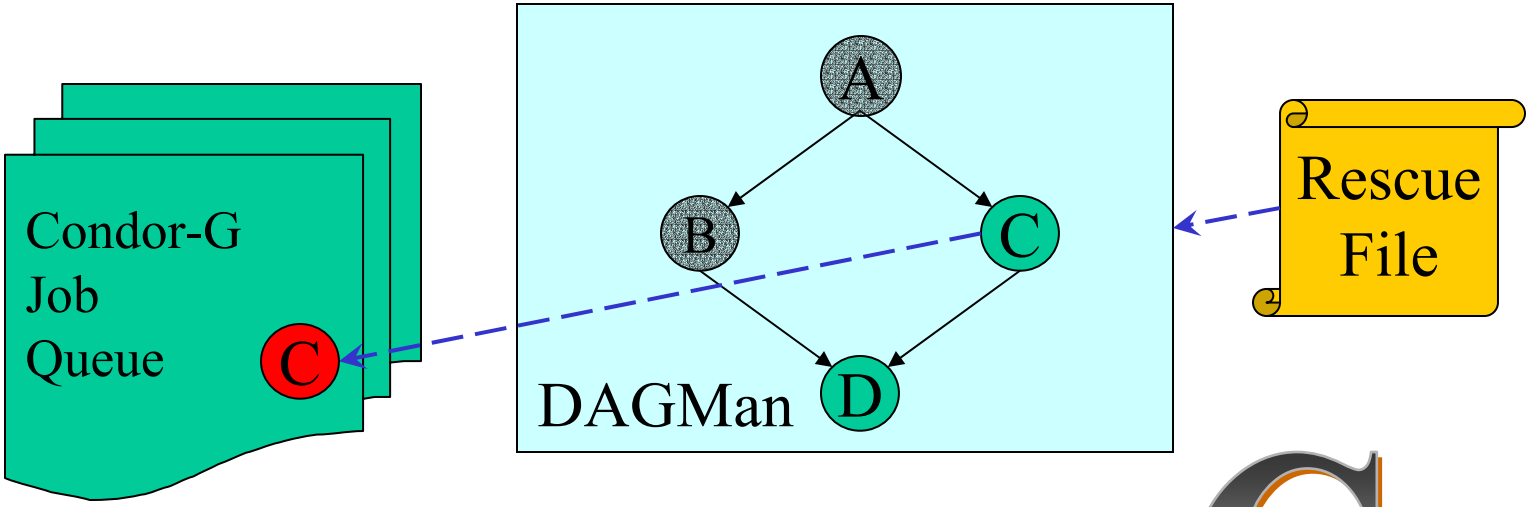
Running a DAG (cont'd)

- In case of a job failure, DAGMan continues until it can no longer make progress, and then creates a *"rescue" file* with the current state of the DAG.



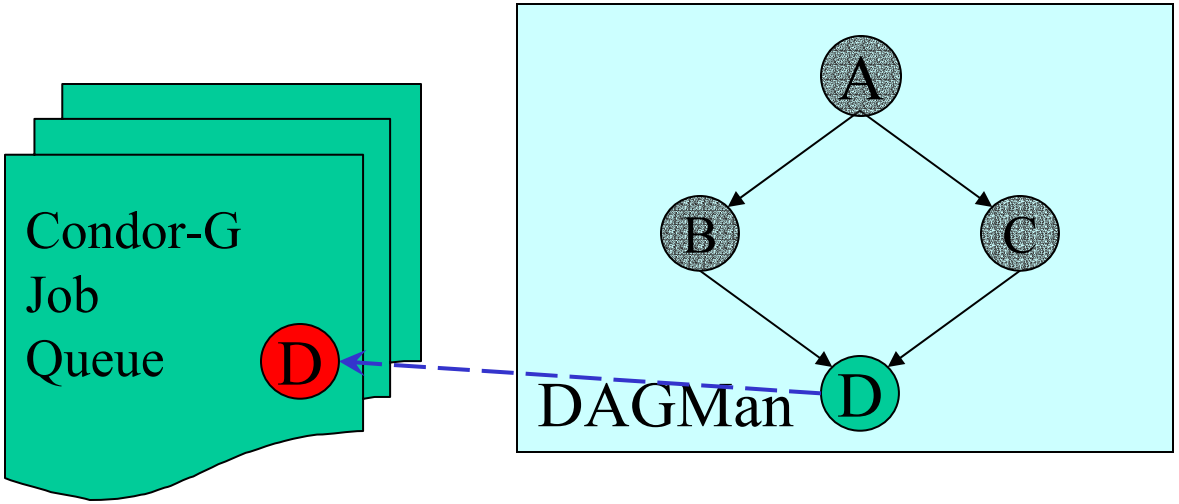
Recovering a DAG

- Once the failed job is ready to be re-run, the rescue file can be used to restore the prior state of the DAG.



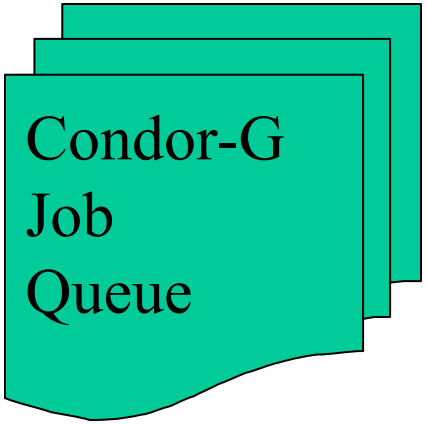
Recovering a DAG (cont'd)

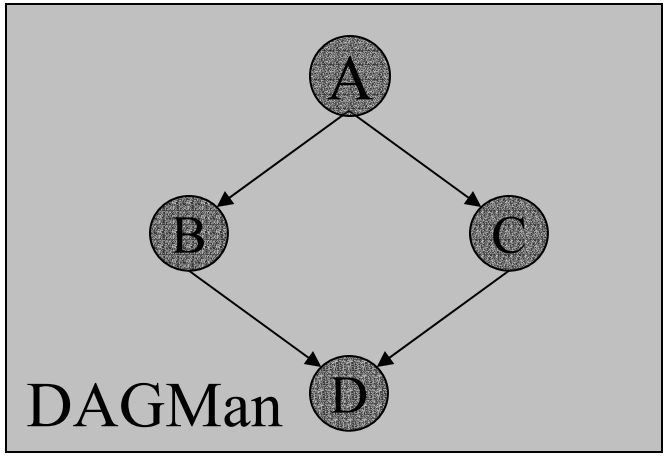
- Once that job completes, DAGMan will continue the DAG as if the failure never happened.



Finishing a DAG

- Once the DAG is complete, the DAGMan job itself is finished, and exits.

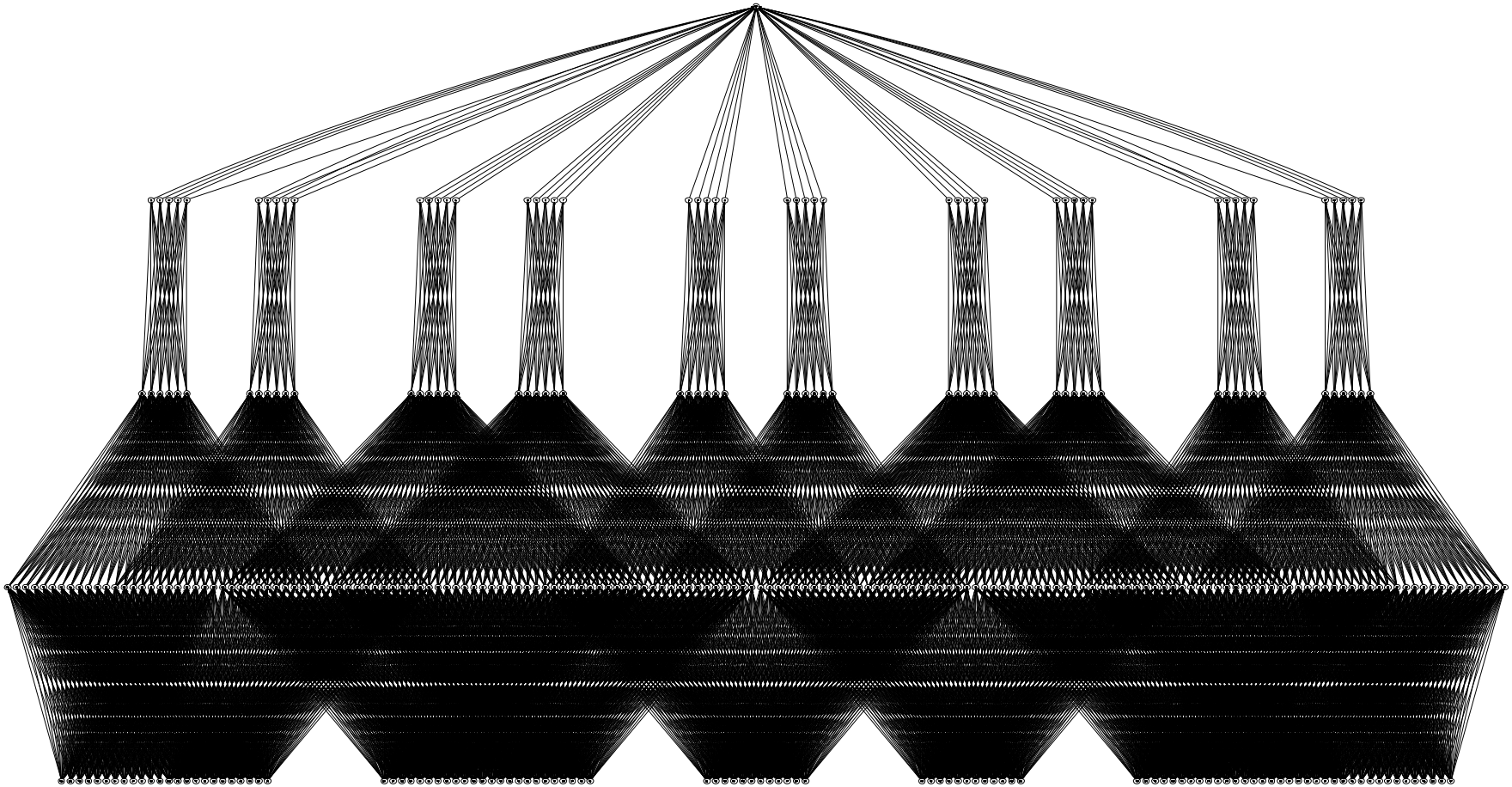




Additional DAGMan Features

- Provides other handy features for job management…
  - nodes can have PRE & POST scripts
  - failed nodes can be automatically re-tried a configurable number of times
  - job submission can be "throttled"

And Even Bigger: 744 Files, 387 Nodes



We've seen how Condor-G 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 ?

condor_master

- Starts up the Condor-G daemon
- If there are any problems and the daemon exits, it restarts it and sends email to the administrator
- Checks the time stamps on the binaries of the other Condor-G daemons, and if new binaries appear, the master will gracefully shutdown the currently running version and start the new version

condor_master (cont'd)

- Acts as the server for many Condor-G remote administration commands:
  - *condor_reconfig, condor_restart, condor_off, condor_on, condor_config_val*, etc.

condor_schedd

- Represents users to the Condor-G system
- Maintains the persistent queue of jobs
- Responsible for contacting available grid sites and sending them jobs
- Services user commands which manipulate the job queue:
  - *condor_submit, condor_rm, condor_q, condor_hold, condor_release, condor_prio, …*

condor_collector

- Collects information on available resources from multiple grid sites
  - "Directory Service" / Database for Condor-G
- Each site sends a periodic update called a "ClassAd" to the collector
- Services queries for information:
  - Queries from Condor-G
  - Queries from users (*condor_status*)

condor_negotiator

- Performs "matchmaking" for Condor-G
- Gets information from the collector about available grid resources and idle jobs, and tries to match jobs with sites
- Not an exact science due to the nature of the grid
  - Information is out of date by the time it arrives.
  - …but good for large-scale assignment of jobs to avoid idle sites or overstuffed queues.
  - …and policy expressions can be used to "re-match" jobs to new sites if things don't turn out as expected…

Job Policy Expressions

- User can supply job policy expressions in the submit file.
- Can be used to describe a successful run.

```
on_exit_remove = <expression>
on_exit_hold = <expression>
periodic_remove = <expression>
periodic_hold = <expression>
```

Job Policy Examples

- Do not remove if exits with a signal:

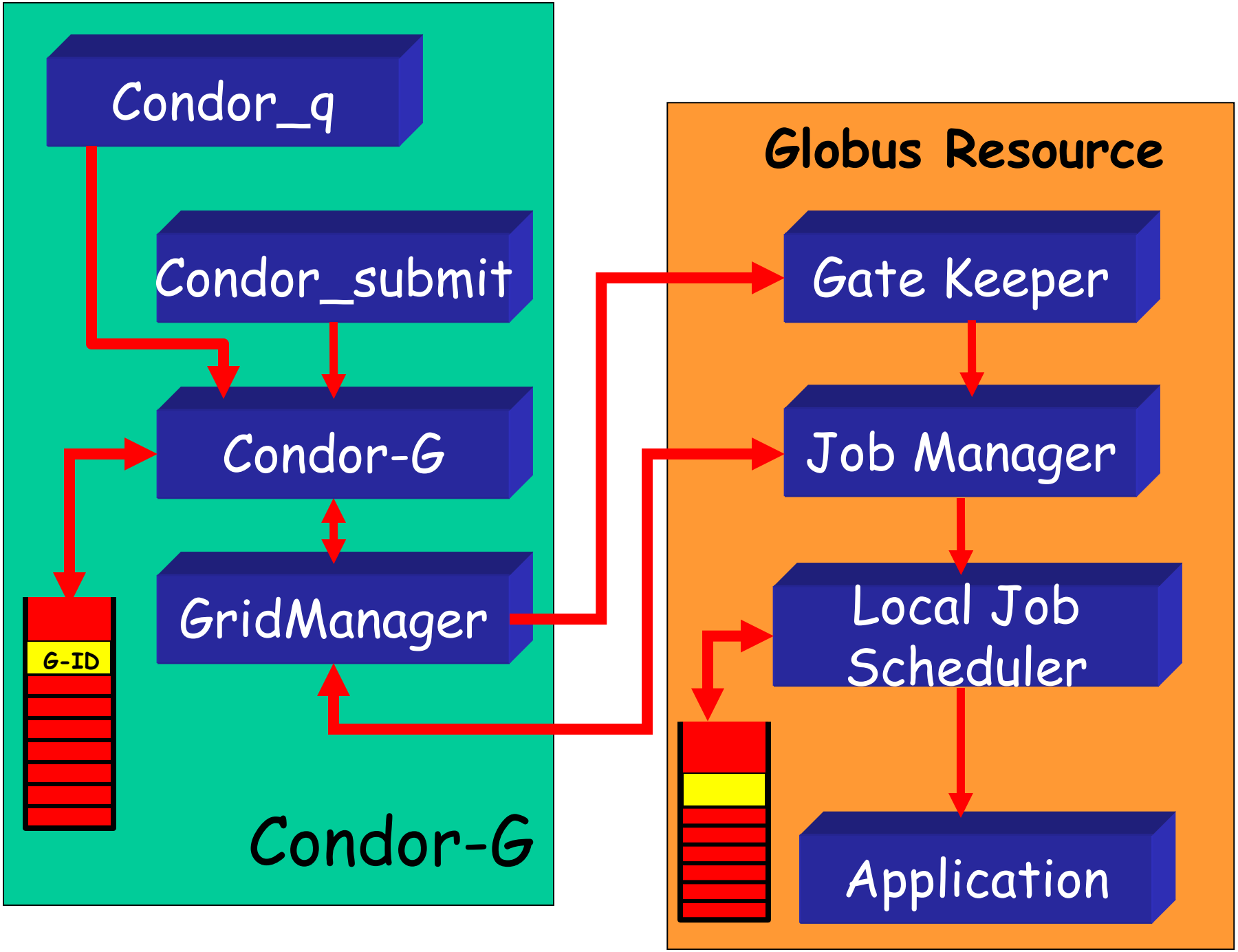
  on_exit_remove = ExitBySignal == False

- Place on hold if exits with nonzero status or ran for less than an hour:

  on_exit_hold = ((ExitBySignal==False) && (ExitSignal != 0)) || ((ServerStartTime – JobStartDate) < 3600)

- Place on hold if job has spent more than 50% of its time suspended:

  periodic_hold = CumulativeSuspensionTime > (RemoteWallClockTime / 2.0)


Condor_q
Condor_submit
Condor-G
GridManager
G-ID
Condor-G
Globus Resource
Gate Keeper
Job Manager
Local Job Scheduler
Application

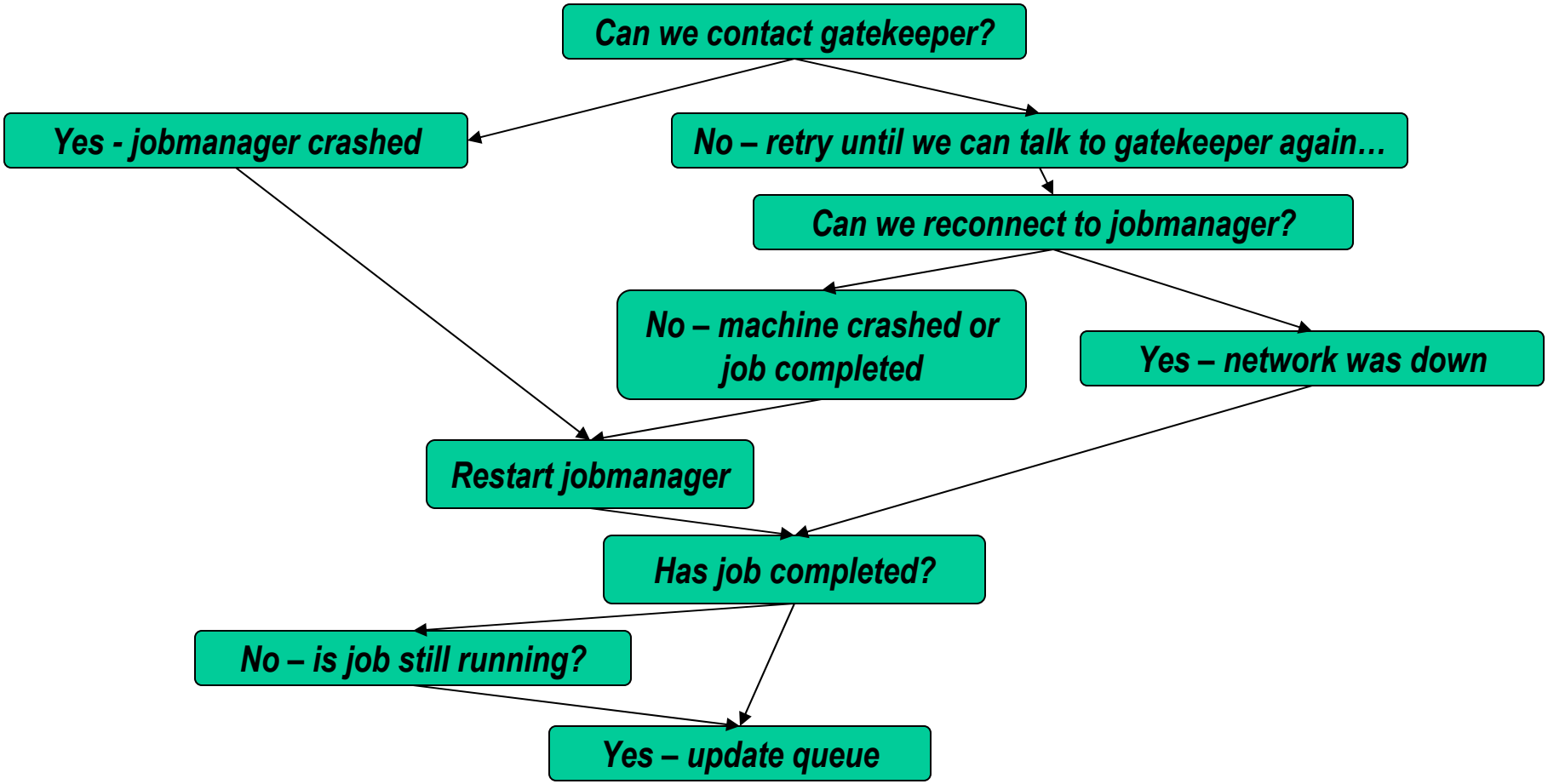
Grid Job Concerns

- What about Fault Tolerance?
  - Local Crashes
    - What if the Condor-G machine goes down?
  - Network Outages
    - What if the connection to the remote Globus jobmanager is lost?
  - Remote Crashes
    - What if the remote Globus jobmanager crashes?
    - What if the remote machine goes down?

Condor-G Fault-Tolerance: Submit-side Failures

- All relevant state for each submitted job is stored persistently in the Condor-G job queue.
- This persistent information allows the Condor-G GridManager upon restart to read the state information and reconnect to JobManagers that were running at the time of the crash.
- If a JobManager fails to respond…

Globus Universe Fault-Tolerance: Lost Contact with Remote Jobmanager



Globus Universe Fault-Tolerance: Credential Management

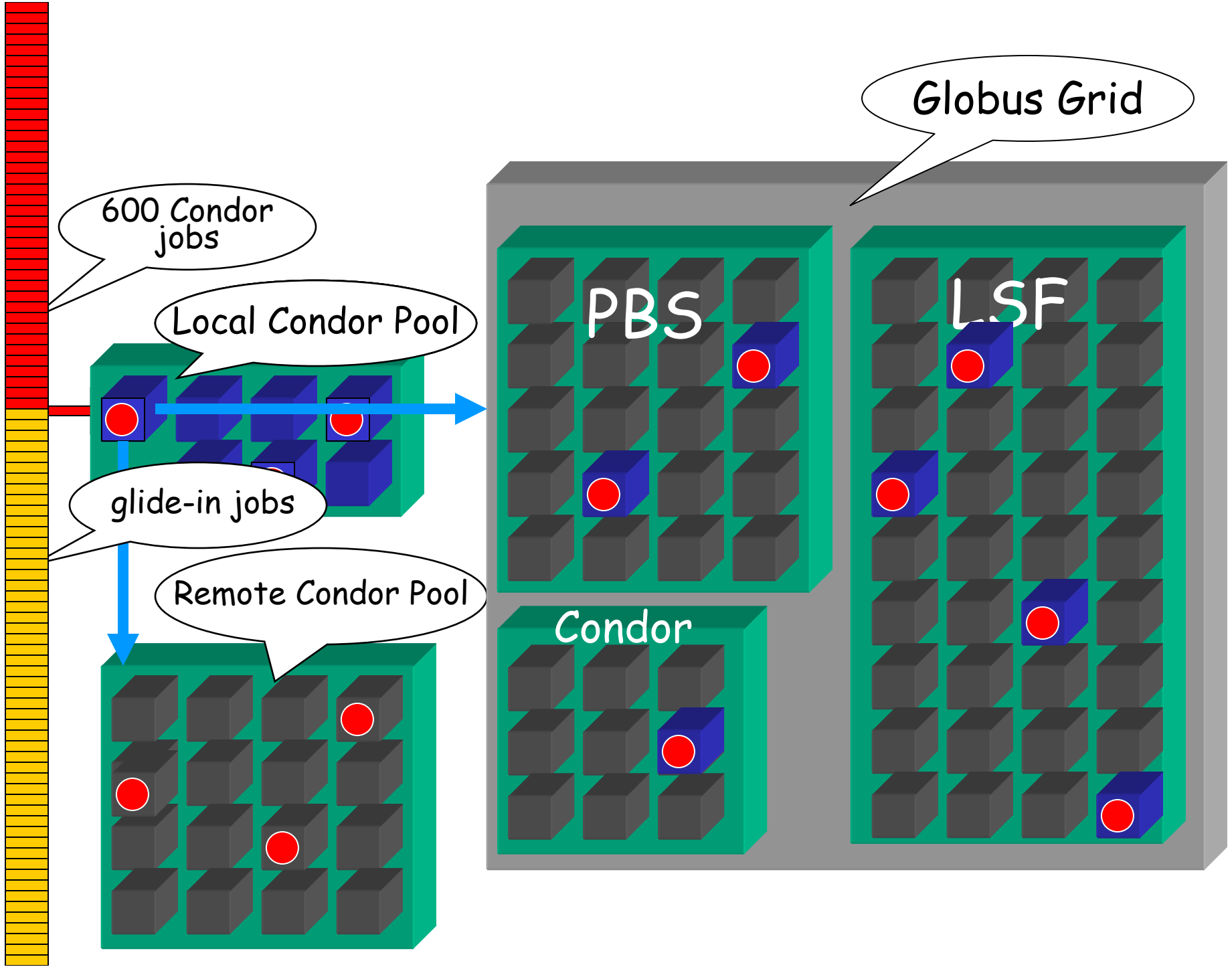
- Authentication in Globus is done with limited-lifetime X509 proxies
- Proxy may expire before jobs finish executing
- Condor can put jobs on hold and email user to refresh proxy …
- or Interface with MyProxy.

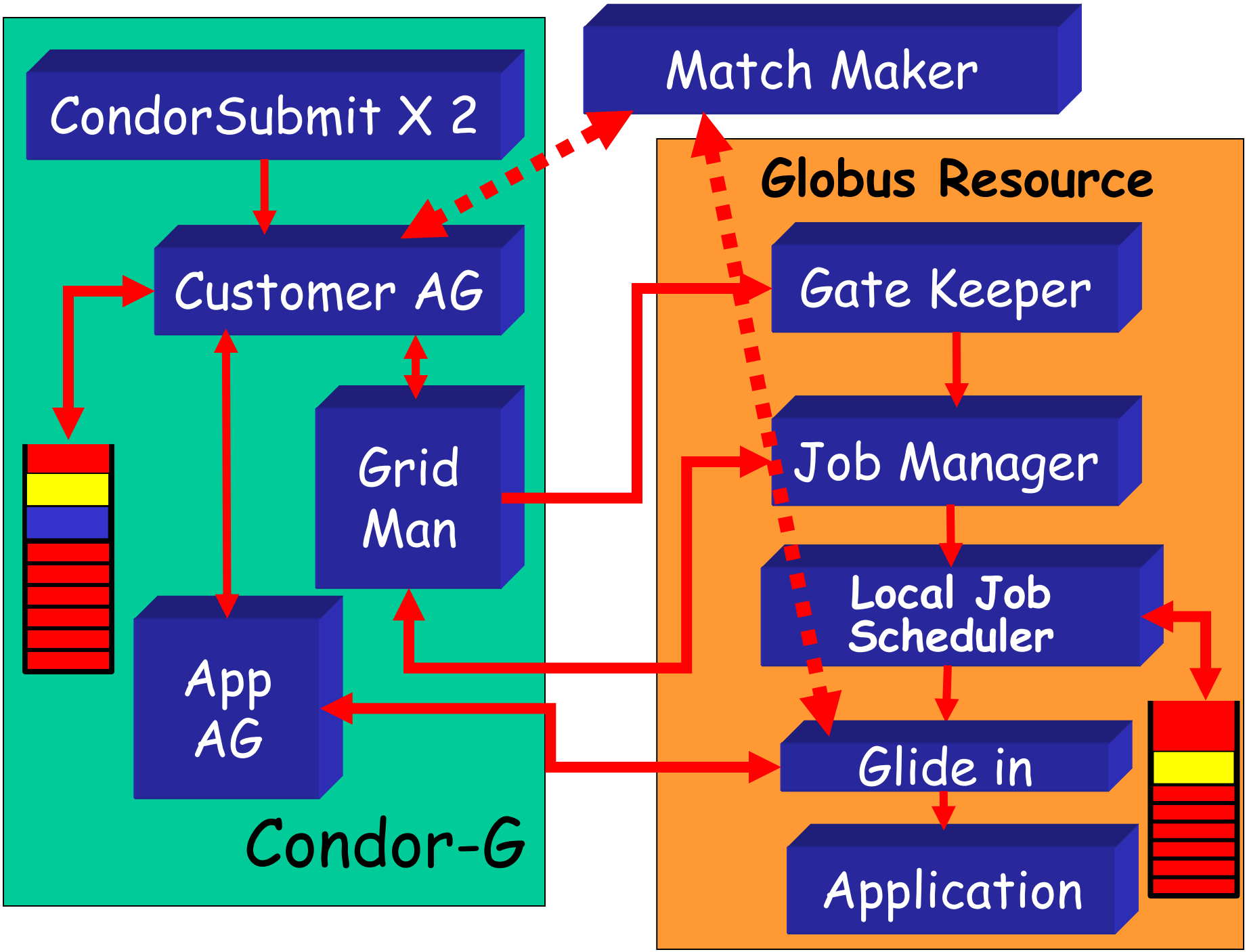
But Frieda Wants More…

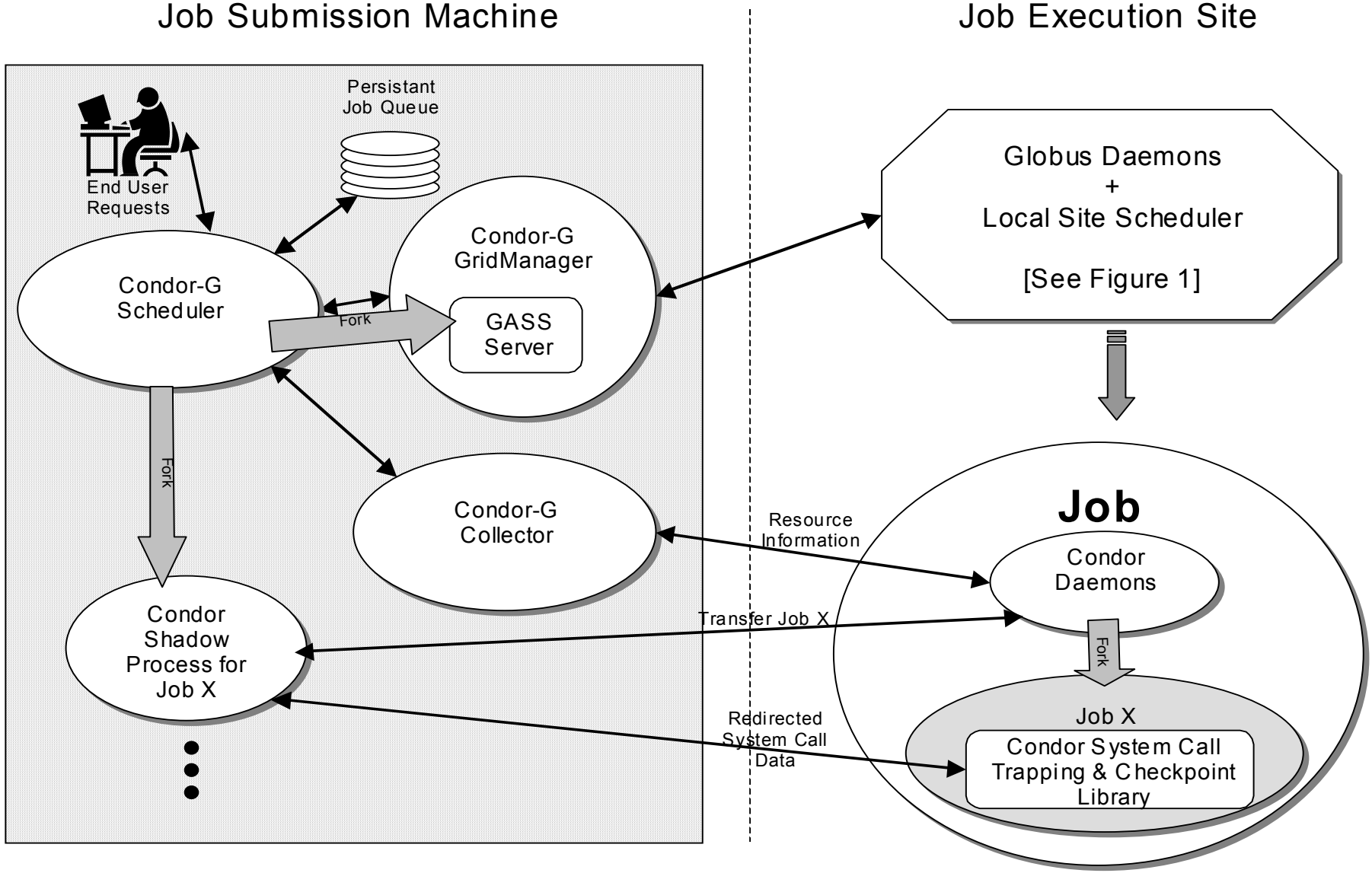
- She wants to run standard universe jobs on Globus-managed resources
  - For matchmaking and dynamic scheduling of jobs
    - Note: Condor-G will now do matchmaking!
  - For job checkpointing and migration
  - For remote system calls

Solution: Condor-G GlideIn

- Frieda can use Condor-G to launch Condor daemons on Globus resources
- When the resources run these GlideIn jobs, they will join a temporary Condor Pool
- She can then submit Condor Standard, Vanilla, PVM, or MPI Universe jobs and they will be matched and run on the Globus resources, as if they were "opportunistic" Condor resources.


Globus Grid
600 Condor jobs
Local Condor Pool
PBS
LSF
glide-in jobs
Remote Condor Pool
Condor


CondorSubmit X 2
Match Maker
Globus Resource
Customer AG
Gate Keeper
Grid Man
Job Manager
Local Job Scheduler
App AG
Glide in
Application
Condor-G


Job Submission Machine
Job Execution Site
Persistant
Job Queue
End User
Requests
Condor-G
Scheduler
Condor-G
GridManager
GASS
Server
Fork
Fork
Condor-G
Collector
Condor
Shadow
Process for
Job X
Globus Daemons
+
Local Site Scheduler
[See Figure 1]
Job
Resource
Information
Condor
Daemons
Transfer Job X
Fork
Job X
Redirected
System Call
Data
Condor System Call
Trapping & Checkpoint
Library

GlideIn Concerns

- What if a Globus resource kills my GlideIn job?
  - That resource will disappear from your pool and your jobs will be rescheduled on other machines
  - Standard universe jobs will resume from their last checkpoint like usual
- What if all my jobs are completed before a GlideIn job runs?
  - If a GlideIn Condor daemon is not matched with a job in 10 minutes, it terminates, freeing the resource

In Review

With Condor-G Frieda can…

- … manage her compute job workload
- … access remote compute resources on the Grid via Globus Universe jobs
- … carve out her own personal Condor Pool from the Grid with GlideIn technology

Condor-G Matchmaking

- Alternative to Glidein: Use Condor-G matchmaking with globus universe jobs
- Allows Condor-G to dynamically assign computing jobs to grid sites
- An example of *lazy planning*

Condor-G Matchmaking, cont.

- Normally a globus universe job must specify the site in the submit description file via the "globusscheduler" attribute like so:

```
Executable = foo
Universe = globus
Globusscheduler = beak.cs.wisc.edu/jobmanager-pbs
queue
```

Condor-G Matchmaking, cont.

- With matchmaking, globus universe jobs can use requirements and rank:

```
Executable = foo
Universe = globus
Globusscheduler = $$(GatekeeperUrl)
Requirements = arch == LINUX
Rank = NumberOfNodes
Queue
```

- The $$(x) syntax inserts information from the target ClassAd when a match is made.

Condor-G Matchmaking, cont.

- Where do these target ClassAds representing Globus gatekeepers come from? Several options:
  - Simple script on gatekeeper publishes an ad via *condor_advertise* command-line utility (method used by D0 JIM, USCMS)
  - Program to query Globus MDS and convert information into ClassAd (method used by EDG)
  - Run HawkEye with appropriate plugins on the gatekeeper
- An explanation of Condor-G matchmaking setup see `http://www.cs.wisc.edu/condor/USCMS_matchmaking.html`

DAGMan Callouts

- Another mechanism to achieve lazy planning: DAGMan callouts
- Define DAGMAN_HELPER_COMMAND in condor_config (usually a script)
- The helper command is passed a copy of the job submit file when DAGMan is about to submit that node in the graph
- This allows changes to be made to the submit file (such as changing GlobusScheduler) at the last minute

Some Recent or soon to arrive Condor-G / DAGMan features

- Condor-G can submit and manage jobs not only in Condor and Globus managed grids, but also to
  - Nordugrid (http://www.nordugrid.org/)
  - Oracle Database (using Oracle Call Interface [OCI] API)
  - UNICORE
- Dynamic DAGs

Some recent or soon to arrive Condor-G / DAGMan features, cont.

- Multi-Tier job submission
  - Allows jobs to be submitted from a machine which need not be always connected to the network (e.g. a laptop)
  - condor_submit sends job Classad and job "sandbox" to a remote condor_schedd
  - condor_fetch_sandbox used to retrieve output from remote condor_schedd when job completes
- SOAP interface
- Job submission to additional remote systems
- Full support for matchmaking

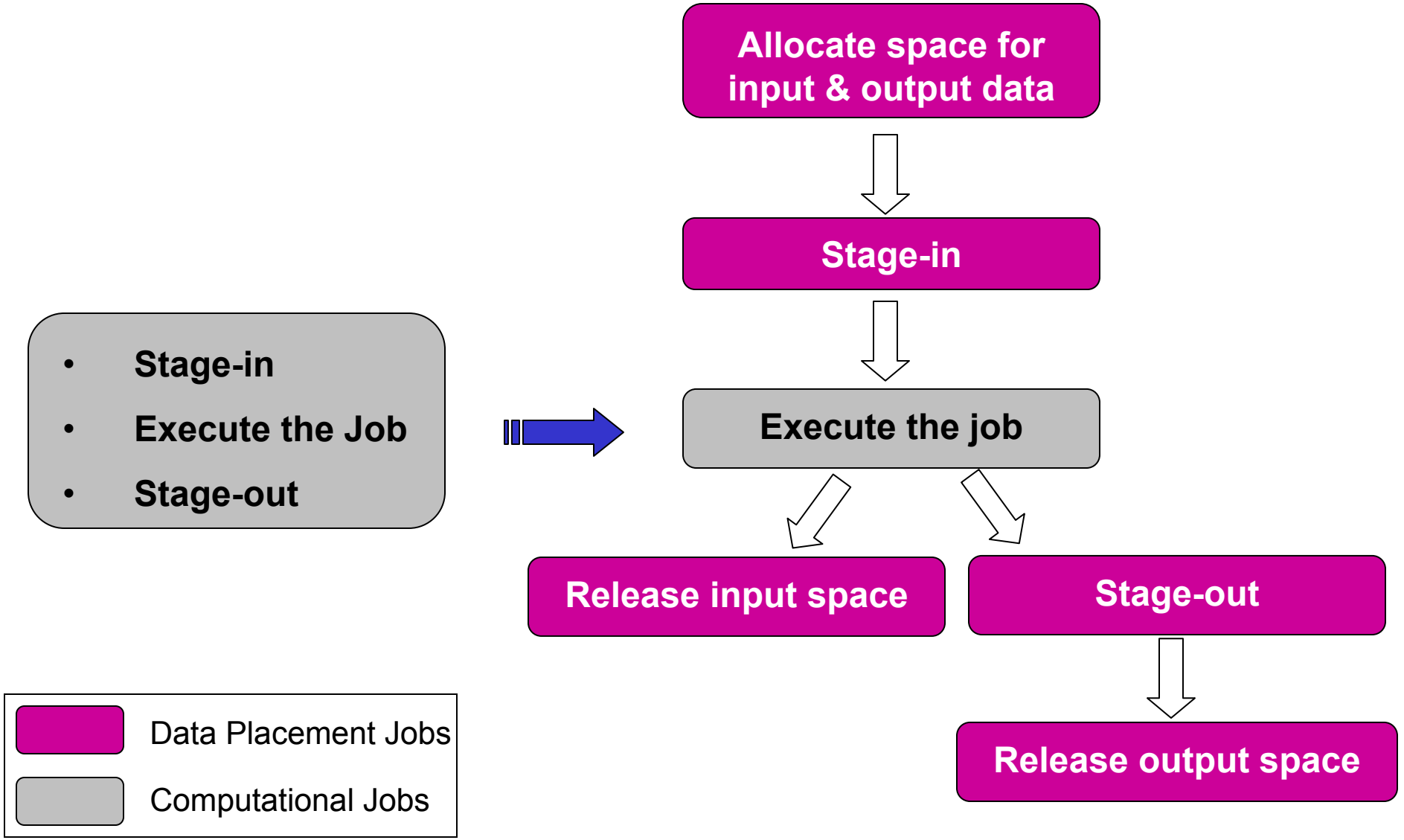
Data Placement* (DaP) must be an integral part of the end-to-end solution

* Space management and Data transfer

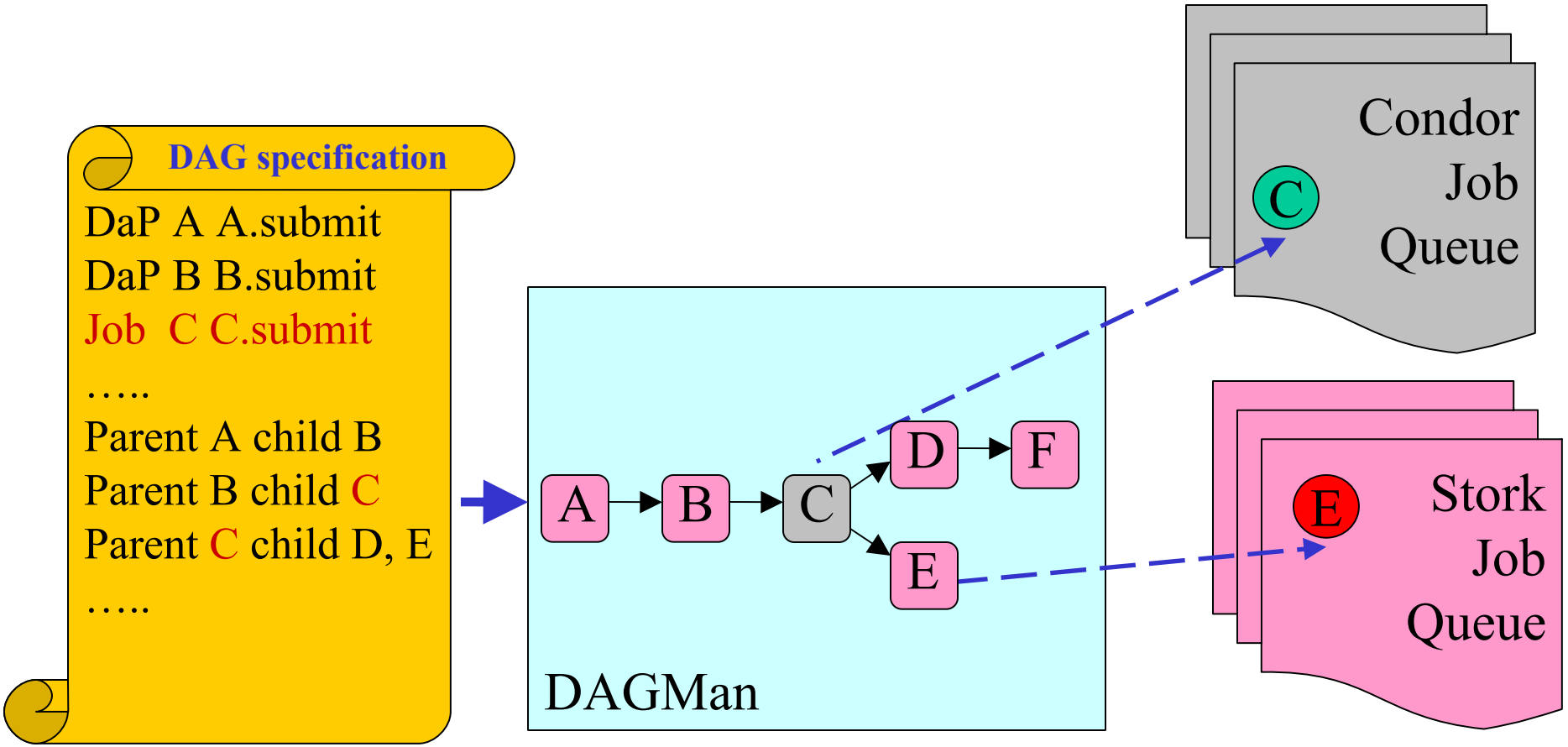
Stork

- A scheduler for data placement activities in the Grid
- What Condor is for computational jobs, Stork is for data placement
- Stork comes with a new concept:

  "Make data placement a first class citizen in the Grid."


Allocate space for input & output data
Stage-in
Execute the job
Release input space
Stage-out
Release output space
• Stage-in
• Execute the Job
• Stage-out
Data Placement Jobs
Computational Jobs

DAG with DaP


DAG specification
DaP A A.submit
DaP B B.submit
Job C C.submit
…..
Parent A child B
Parent B child C
Parent C child D, E
…..
A
B
C
D
F
E
DAGMan
C
Condor
Job
Queue
E
Stork
Job
Queue

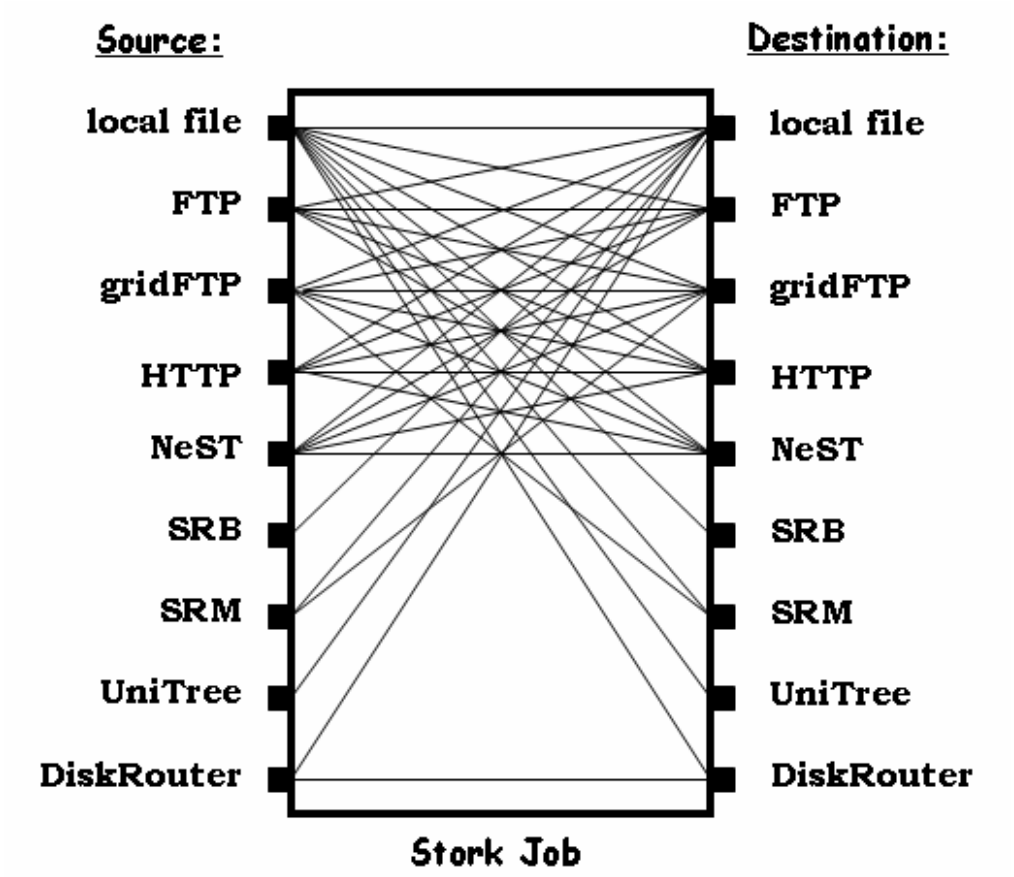
Why Stork?

- Stork understands the characteristics and semantics of data placement jobs.
- Can make smart scheduling decisions, for reliable and efficient data placement.

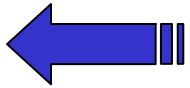
Failure Recovery and Efficient Resource Utilization

- Fault tolerance
  - Just submit a bunch of data placement jobs, and then go away..
- Control number of concurrent transfers from/to any storage system
  - Prevents overloading
- Space allocation and De-allocations
  - Make sure space is available

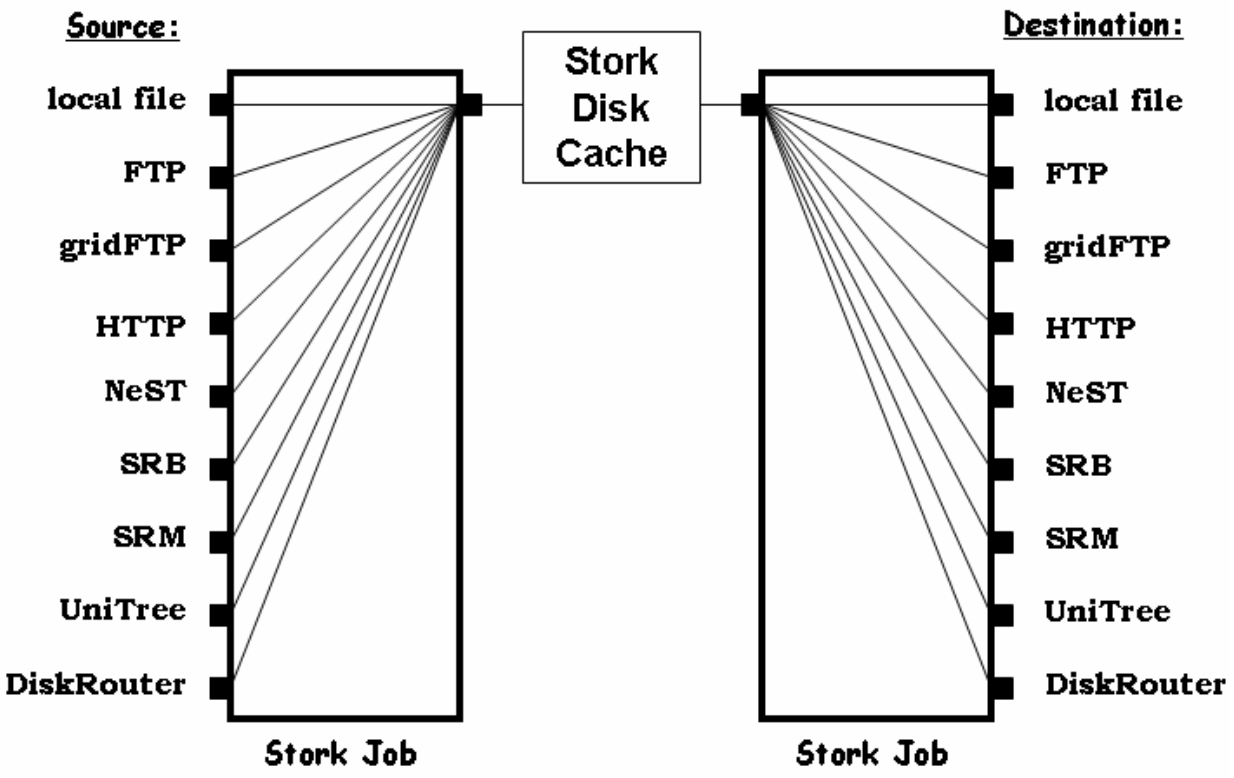
Support for Heterogeneity



Protocol translation using Stork memory buffer.



Support for Heterogeneity



Protocol translation using Stork Disk Cache.

Flexible Job Representation and Multilevel Policy Support

```
[
      Type     = "Transfer";
      Src_Url  = "srb://ghidorac.sdsc.edu/kosart.condor/x.dat";
      Dest_Url = "nest://turkey.cs.wisc.edu/kosart/x.dat";
      ......
      ......
      Max_Retry = 10;
      Restart_in = "2 hours";
]
```

Run-time Adaptation

- Dynamic protocol selection

```
[
  dap_type = "transfer";
  src_url    = "drouter://slic04.sdsc.edu/tmp/test.dat";
  dest_url  = "drouter://quest2.ncsa.uiuc.edu/tmp/test.dat";
  alt_protocols = "nest-nest, gsiftp-gsiftp";
]

[
  dap_type = "transfer";
  src_url    = "any://slic04.sdsc.edu/tmp/test.dat";
  dest_url  = "any://quest2.ncsa.uiuc.edu/tmp/test.dat";
]
```

Run-time Adaptation

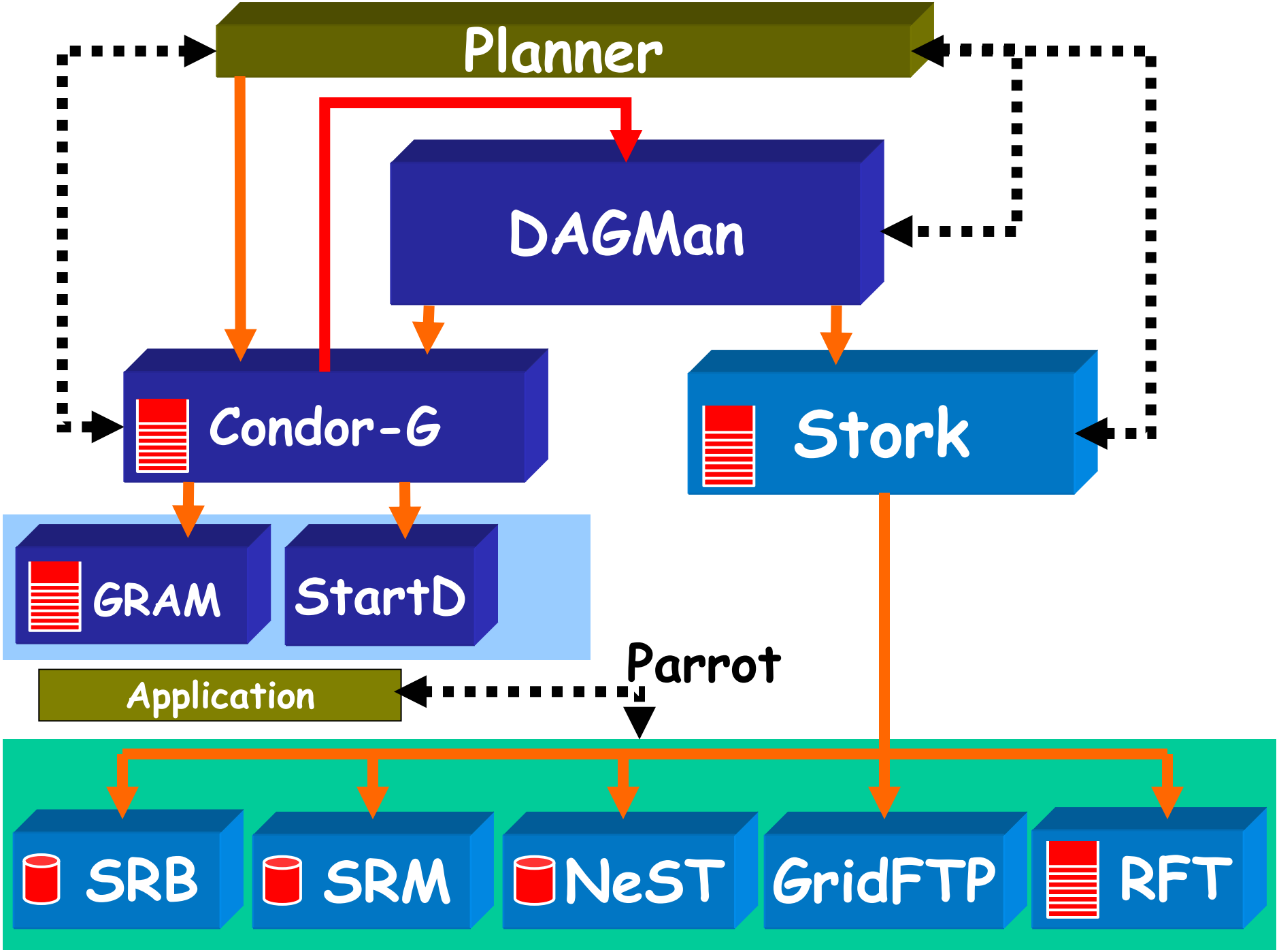
- Run-time Protocol Auto-tuning

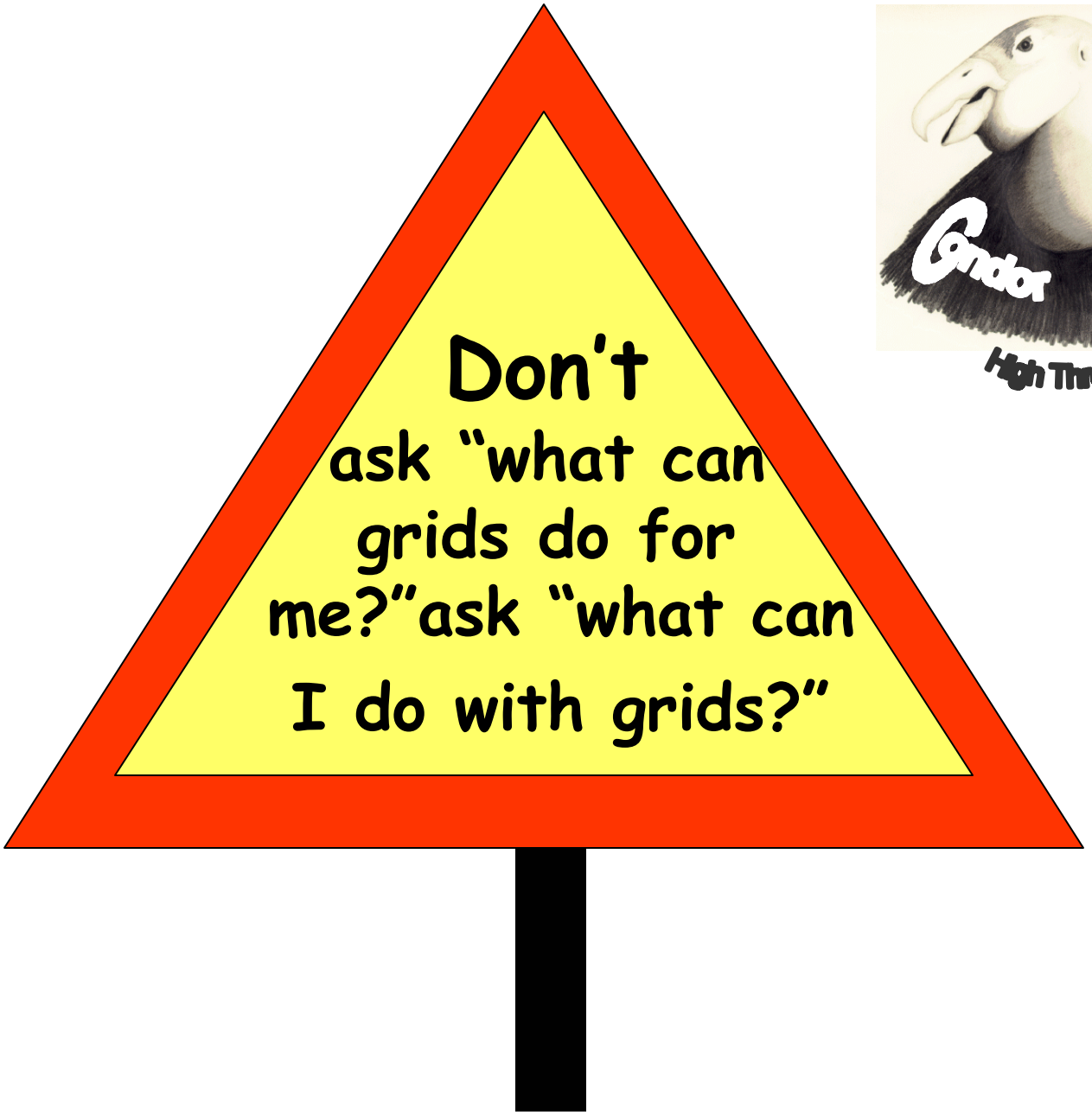
```
[
   link       = "slic04.sdsc.edu – quest2.ncsa.uiuc.edu";
   protocol = "gsiftp";

   bs         = 1024KB;      //block size
   tcp_bs   = 1024KB;      //TCP buffer size
   p          = 4;
]
```

Customer requests:

Place $y = F(x)$ at L!

Master delivers.


Planner
DAGMan
Condor-G
Stork
GRAM
StartD
Parrot
Application
SRB
SRM
NeST
GridFTP
RFT


Don't
ask "what can
grids do for
me?"ask "what can
I do with grids?"


Condor
High Throughput Computing

Thank you!

Check us out on the Web:

http://www.cs.wisc.edu/condor

Email:

condor-admin@cs.wisc.edu