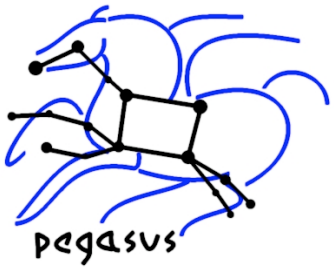
Corral: A Glide-in Based Service for Resource Provisioning

Gideon Juve
USC Information Sciences Institute
juve@usc.edu



Outline

- Throughput Applications
- Grid Computing
- Multi-level scheduling and Glideins
- Corral
- Example: SCEC CyberShake
- Future Work

Throughput Applications

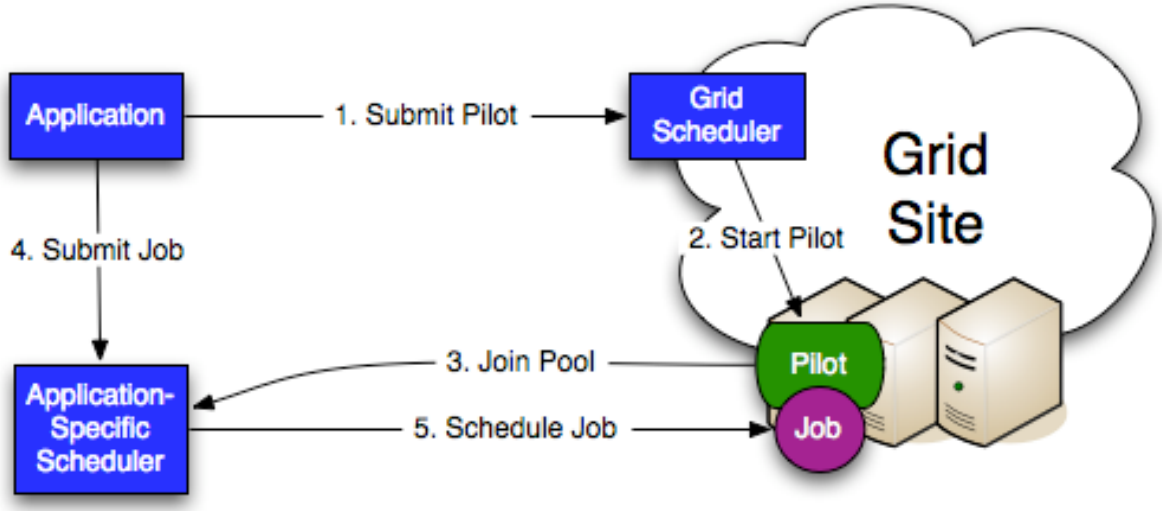
- Characterized by
  - Many tasks: Thousands to millions
  - Short task runtimes: May be less than 60s
  - Tasks are commonly serial
- Key performance metric: **time-to-solution**
- Examples:
  - scientific workflows
  - parameter sweep
  - master-worker
  - “pleasantly parallel”

Grid Computing

- Grids
  - Benefit: Provide plenty of computing resources
  - Challenge: Using those resources effectively
- Grid Overheads
  - Queuing Delays
  - Software Overheads
  - Scheduling Delays
  - Scheduling Policies
  - => Bad performance for throughput applications!
- Some solutions
  - Task clustering (workflows)
  - Advance reservations

Multi-level Scheduling

- Way for an application to use grid without the overheads
- Overlay a **personal cluster** on top of grid resources
- **Pilot jobs** install and run a **user-level resource manager**, which contacts an **application-specific scheduler** to be matched with application jobs
- **Glidein**: How to do MLS using Condor



Benefits of MLS and Glideins

- Running short jobs on the grid
  - Condor dispatches jobs faster than, e.g. Globus
- Bypass site scheduling policies
  - Use application-specific policies
  - e.g. prioritize jobs based on application needs
- Avoid competition for resources
  - Glideins reserve resources for multiple jobs
  - Minimizes queuing delays
- Better application scalability
  - Compared to GT2 GRAM, for example
  - Fewer jobmanagers => reduced load on gateway

Corral

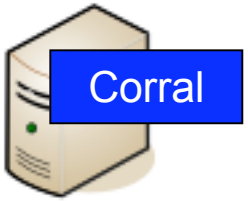
- Resource provisioning system
  - Uses multi-level scheduling model
  - Allocate resources explicitly rather than implicitly
  - Pay to allocate resources once and reuse them
  - Effectively minimizes grid overheads
  - Requires resource specification
- Corral web service
  - Automates the installation and configuration of Condor on grid sites
  - Submits **glideins** to provision resources

How Corral works

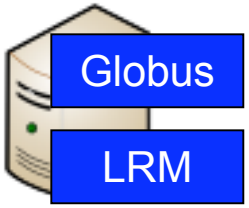
LOCAL SITE

GRID SITE

User

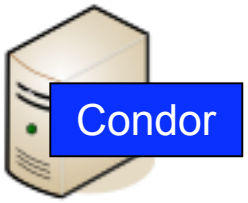


Corral Server



Head Node

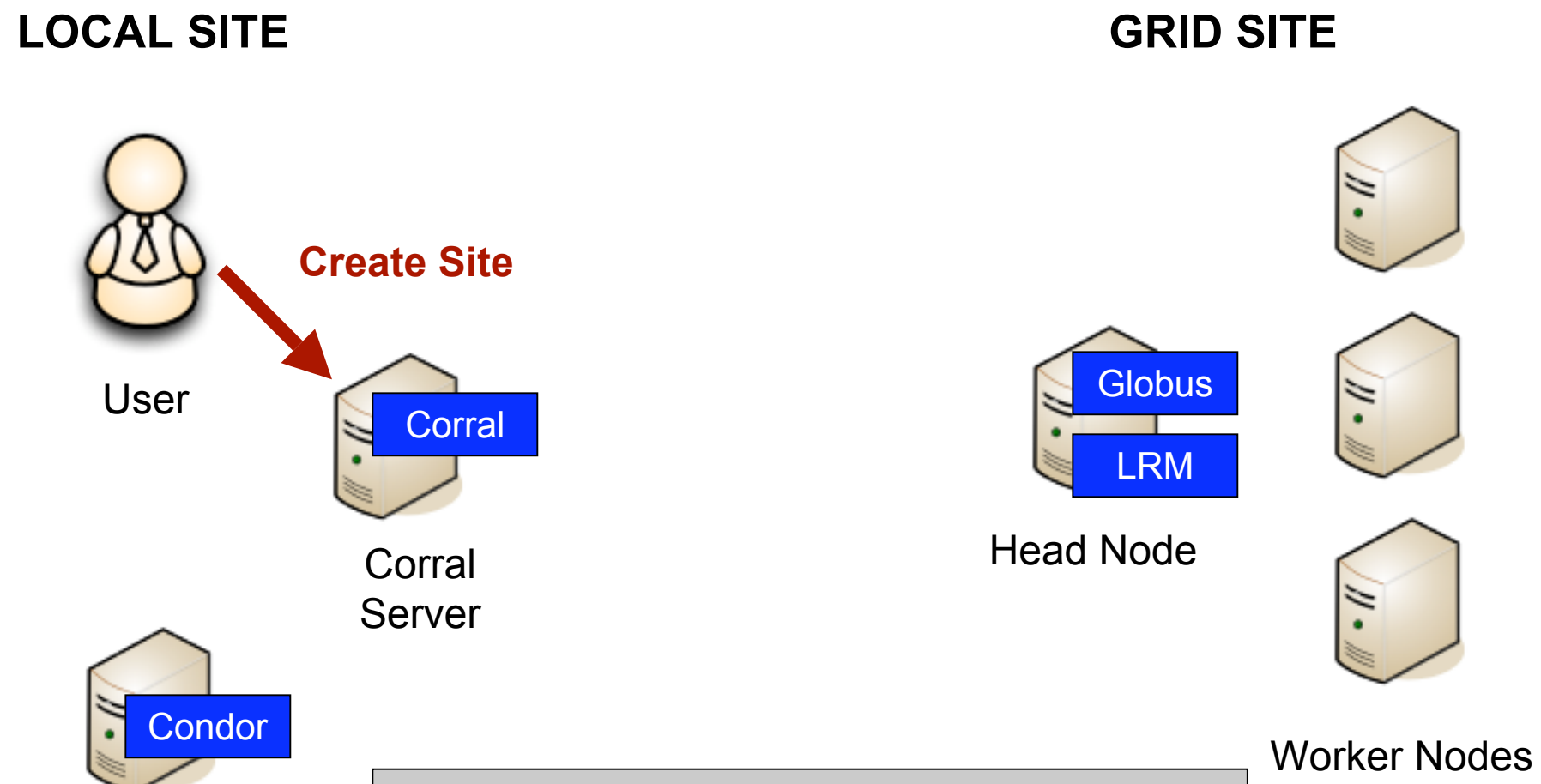
Worker Nodes



Condor Central Manager

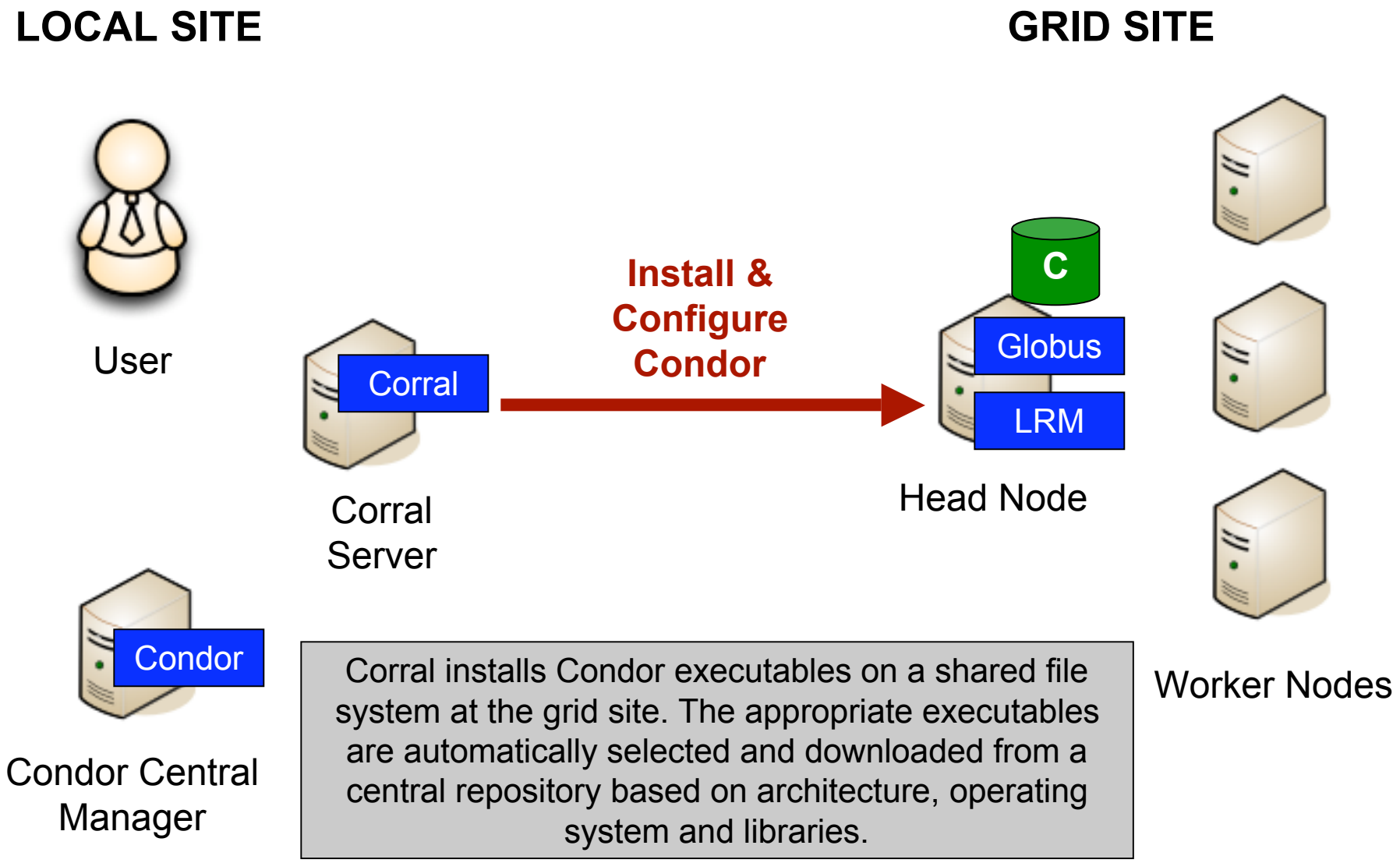
In addition to the Condor central manager, the user runs a service container that hosts the Corral web service.

How Corral works

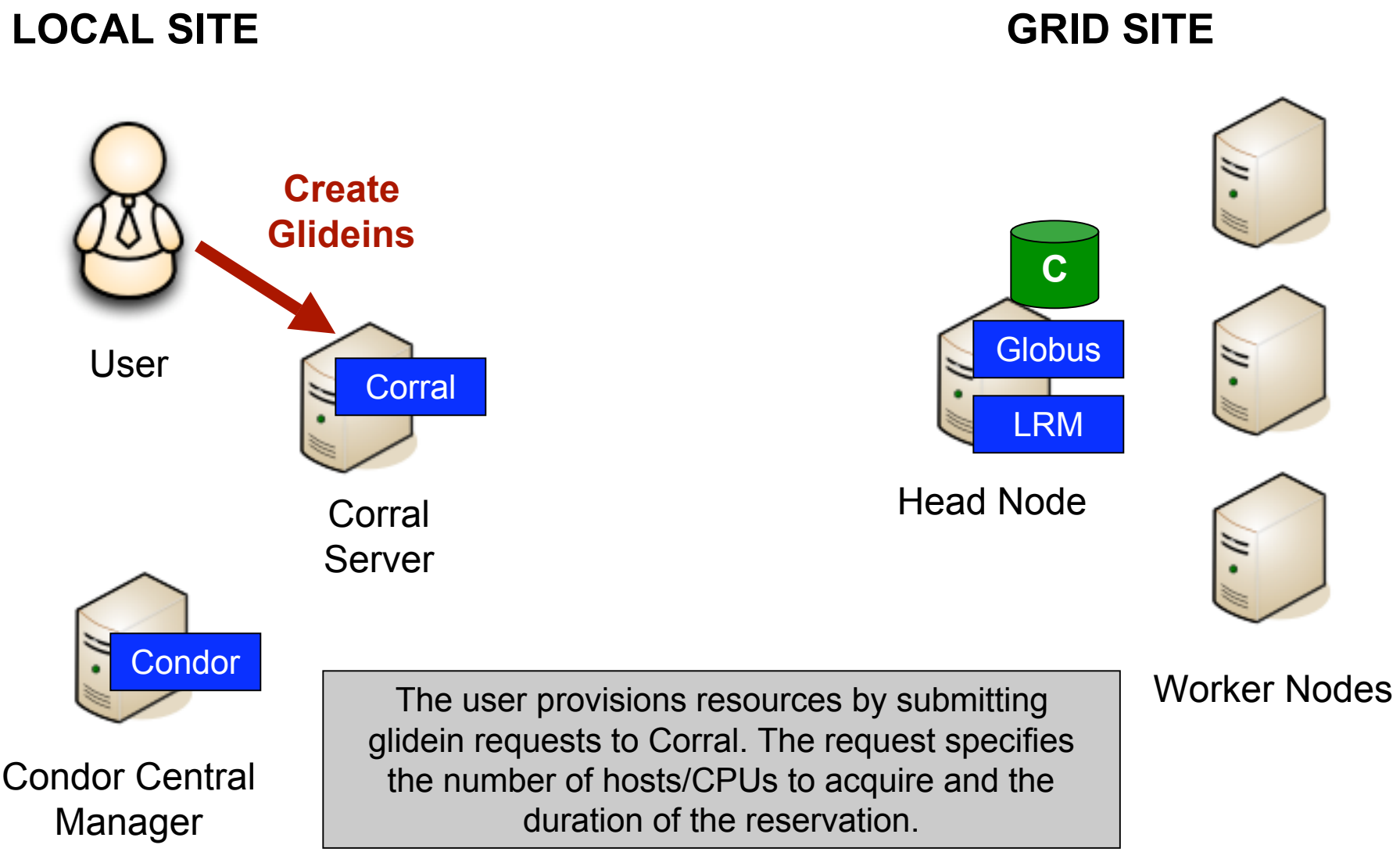


The user creates a new site by sending a request to Corral. The request specifies details about the grid site including: job submission information, file system paths, etc.

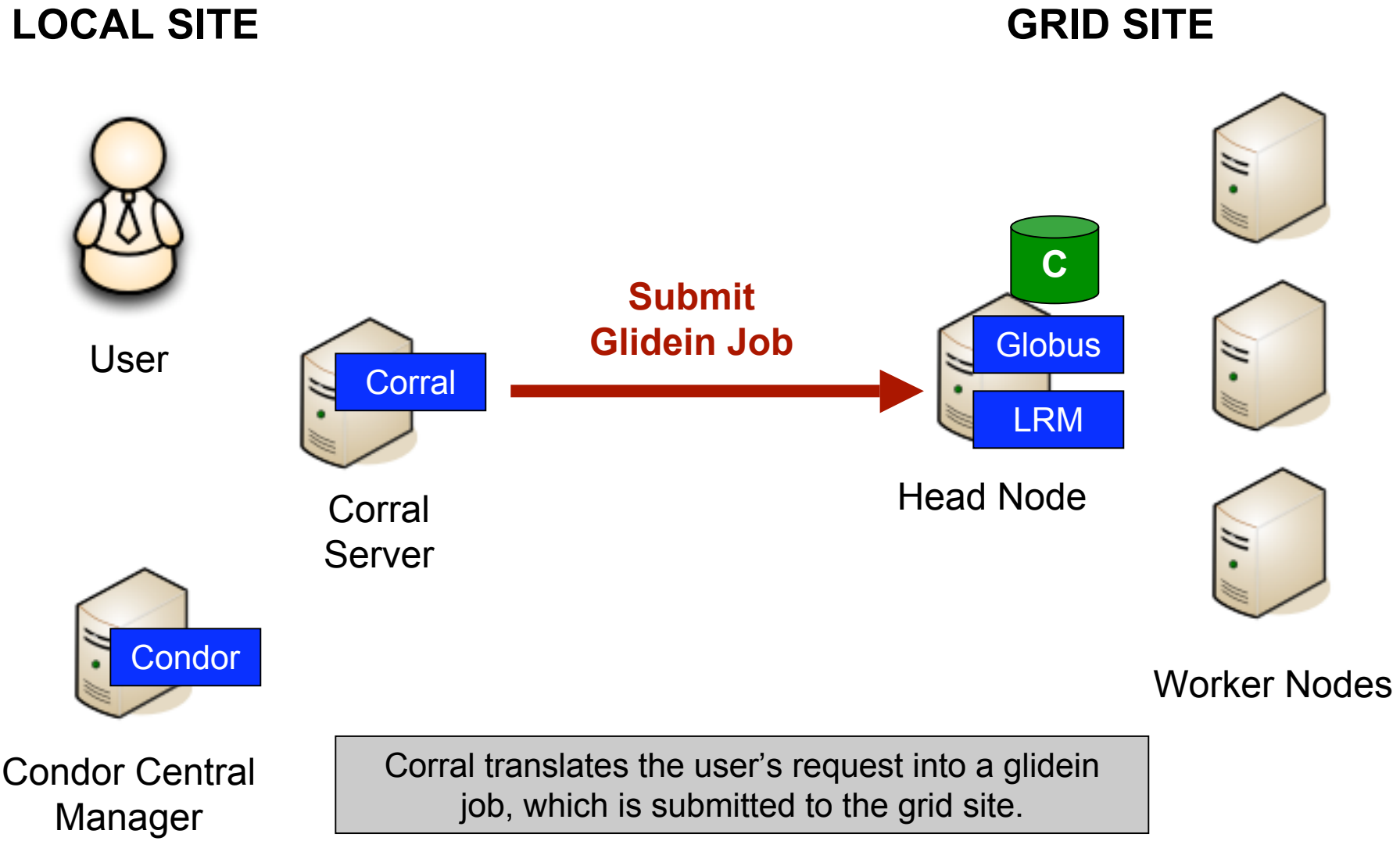
How Corral works



How Corral works

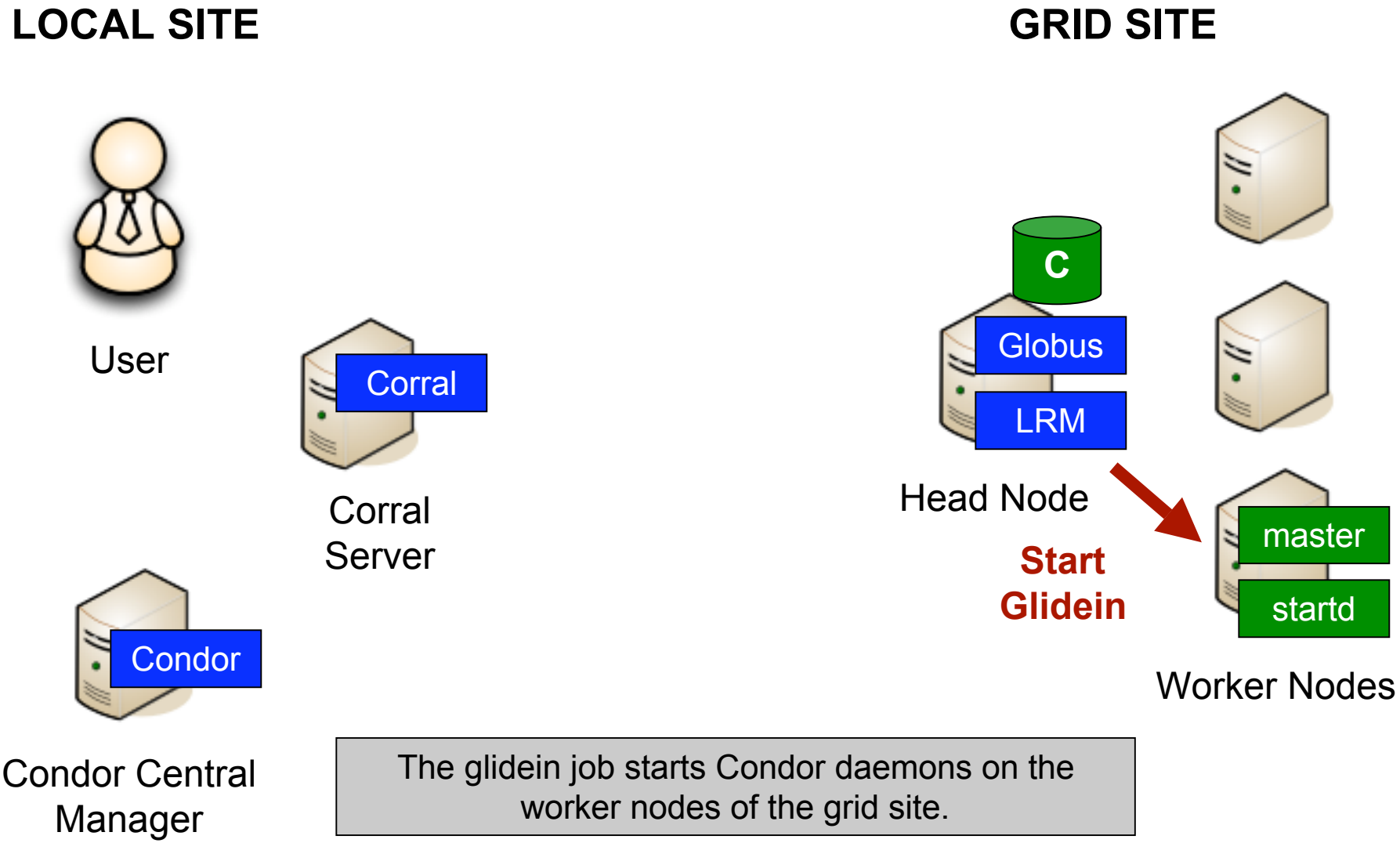


How Corral works

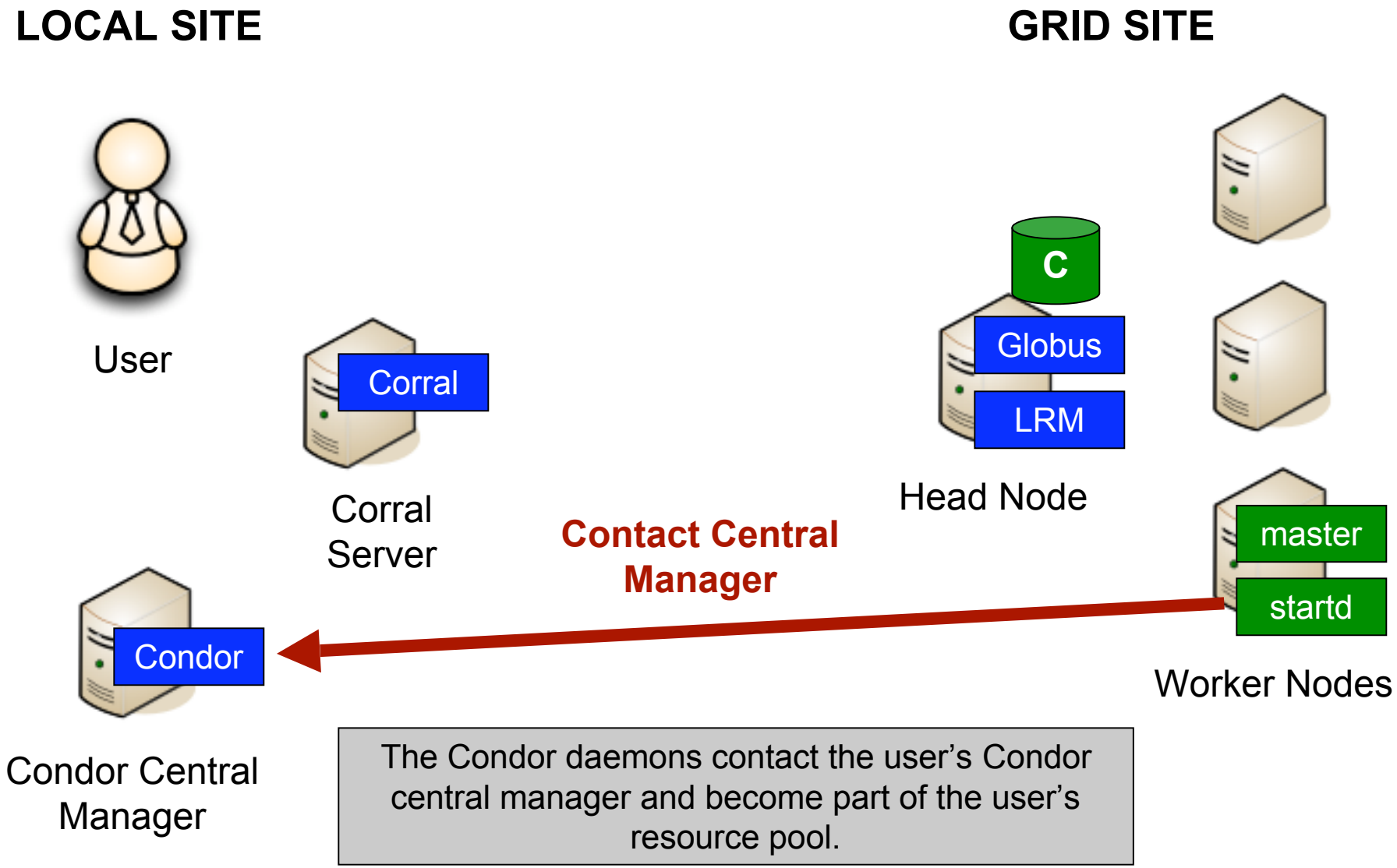


Corral translates the user's request into a glidein job, which is submitted to the grid site.

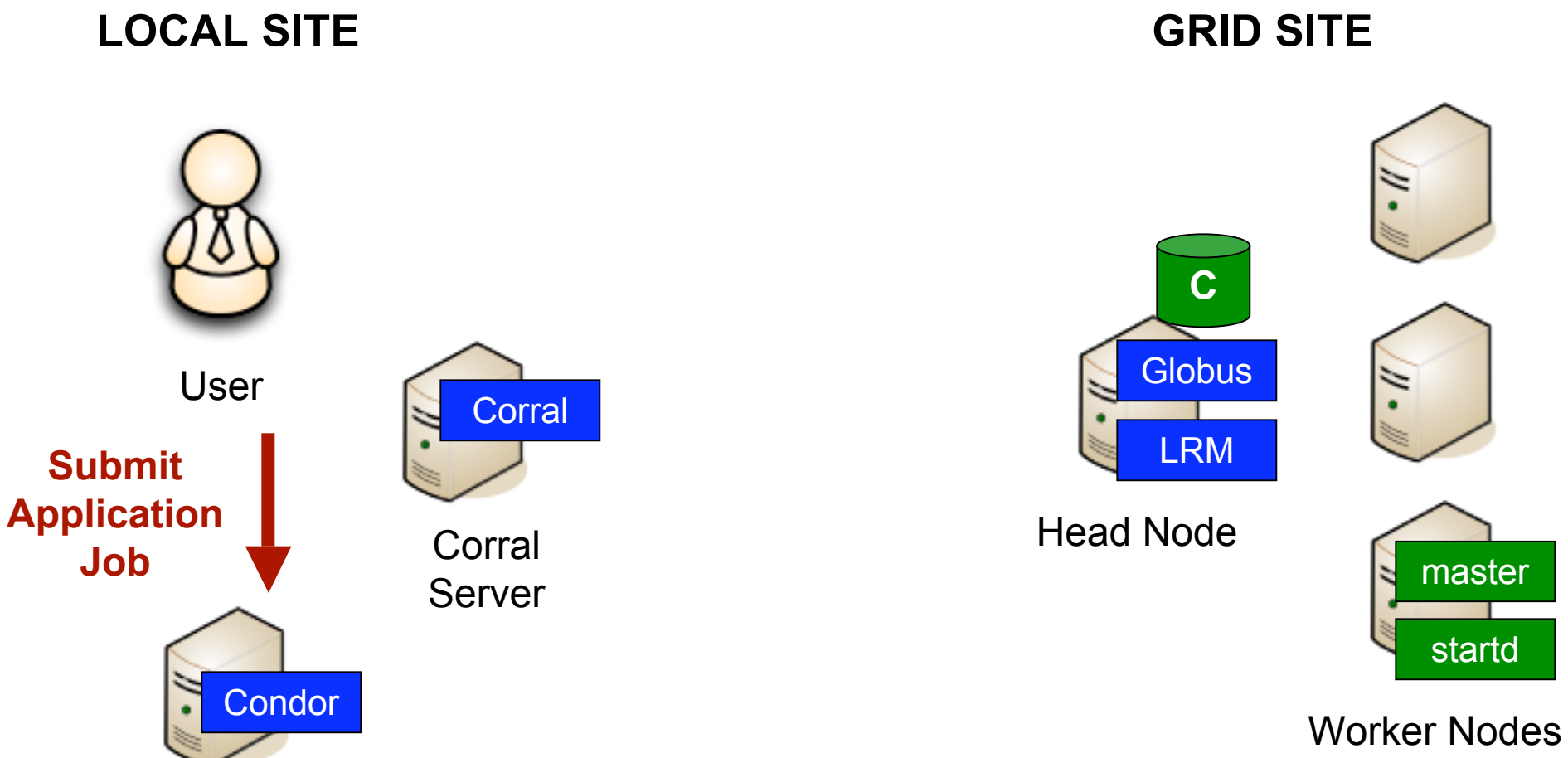
How Corral works



How Corral works

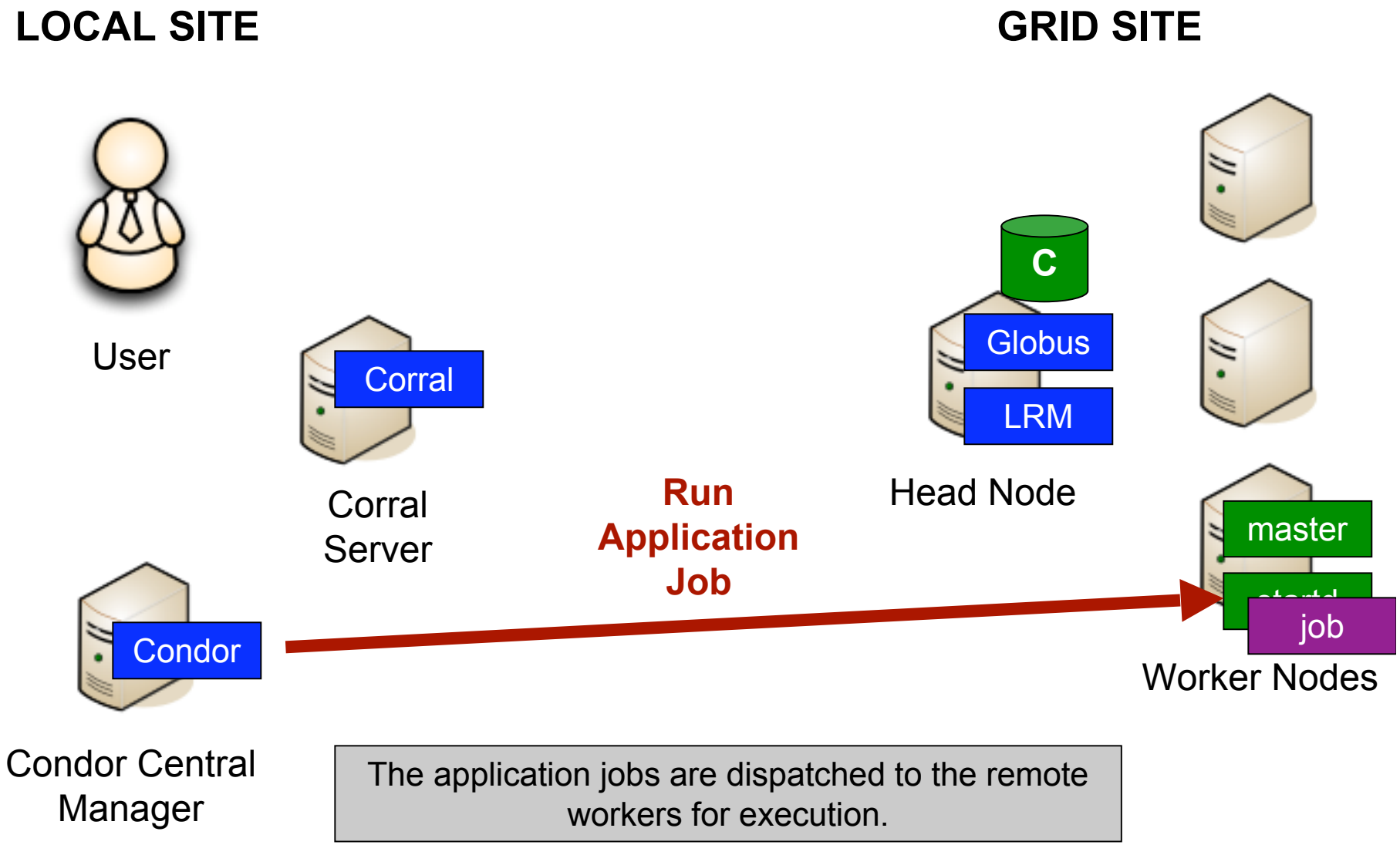


How Corral works

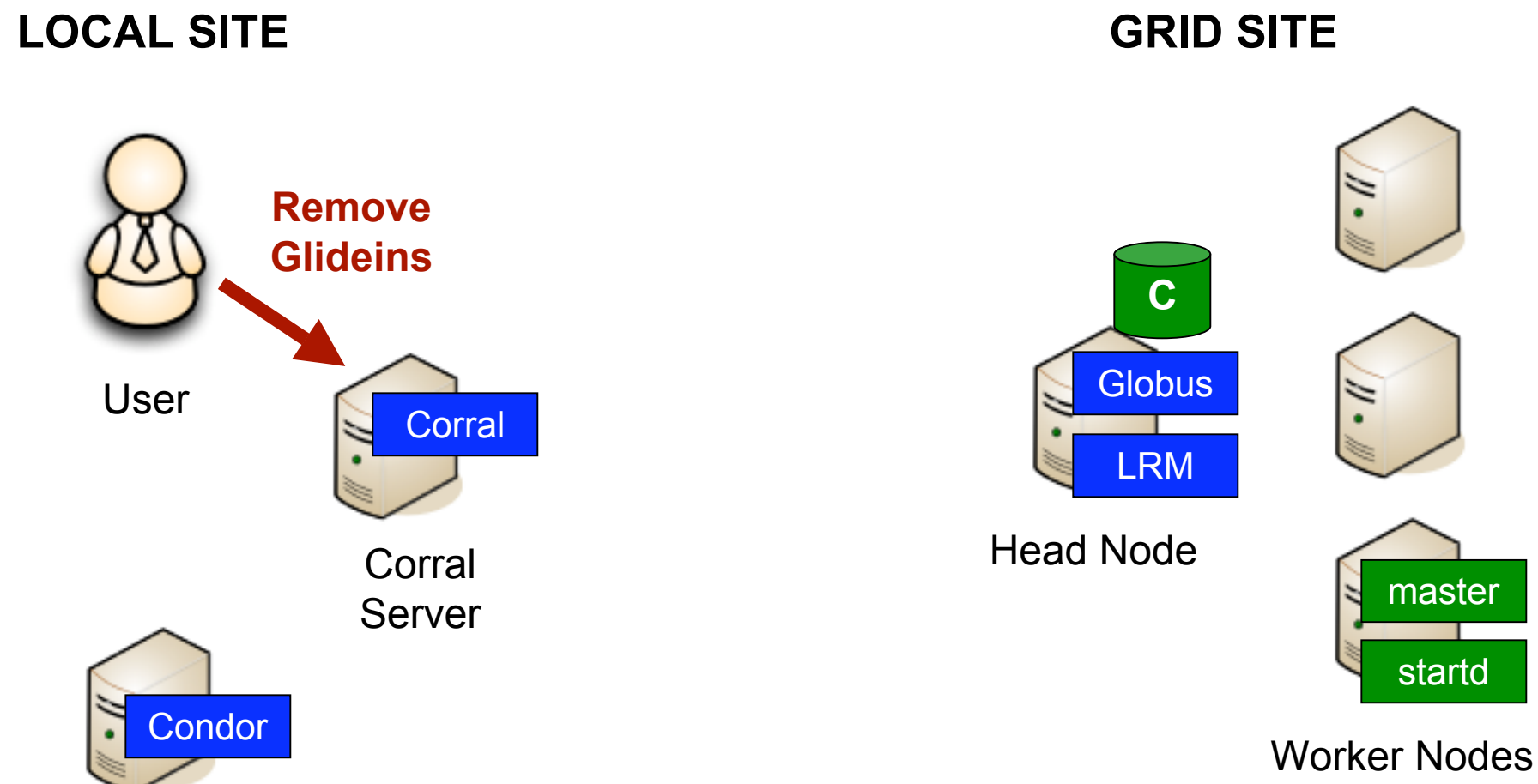


The user submits application jobs to their Condor pool. The jobs are matched with available glidein resources.

How Corral works

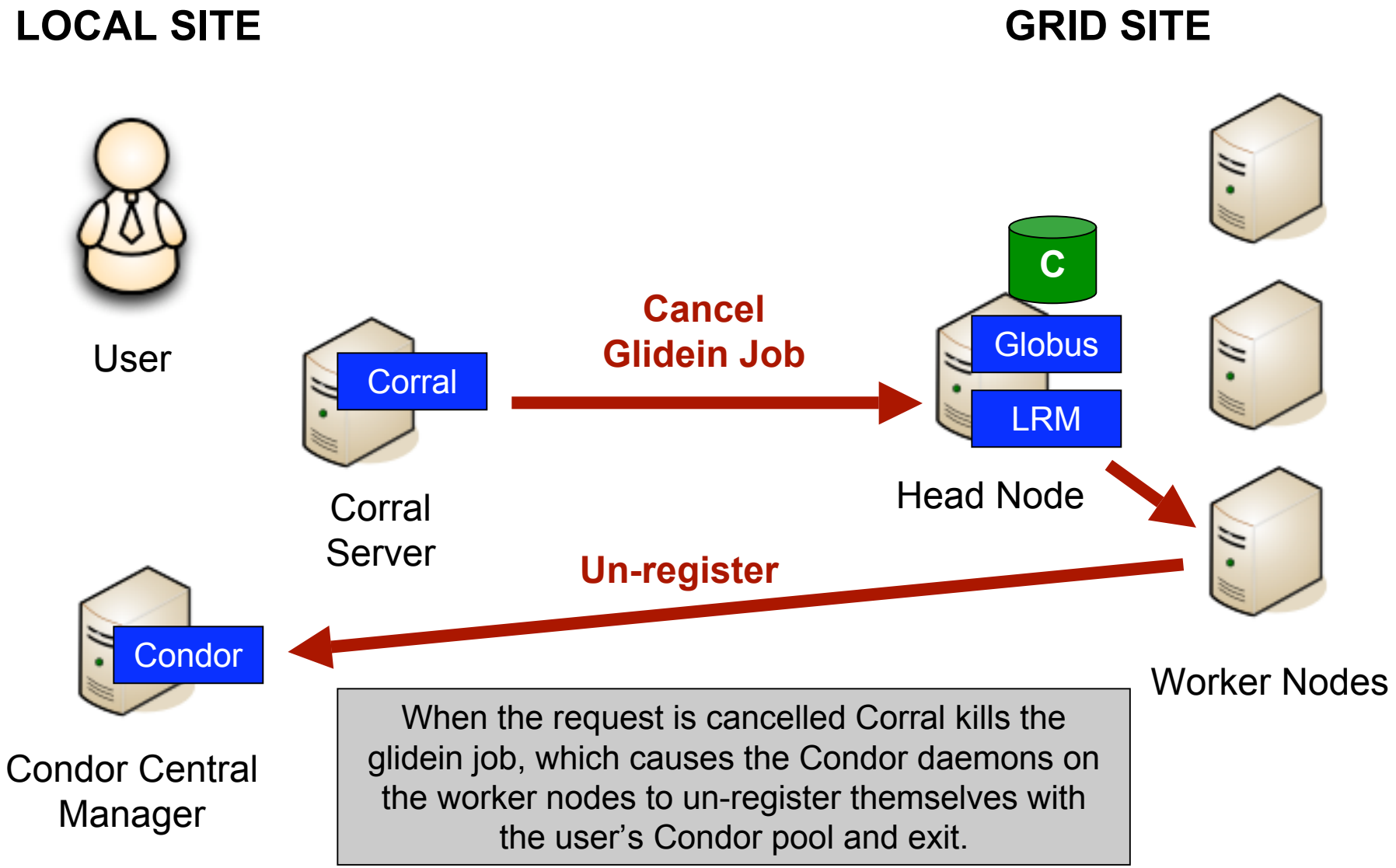


How Corral works

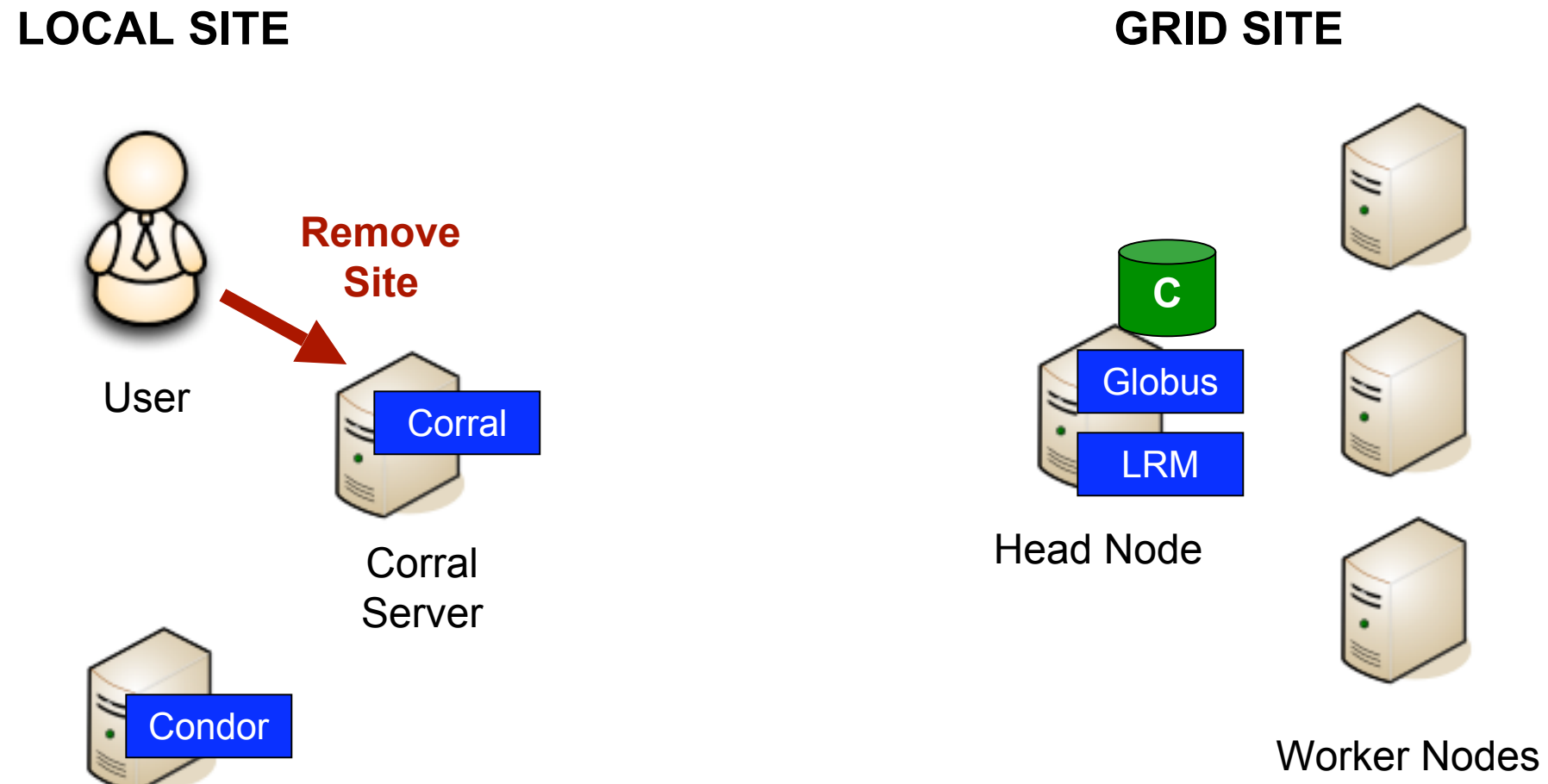


When the user is finished with their application they cancel their glidein request (or let it expire automatically).

How Corral works

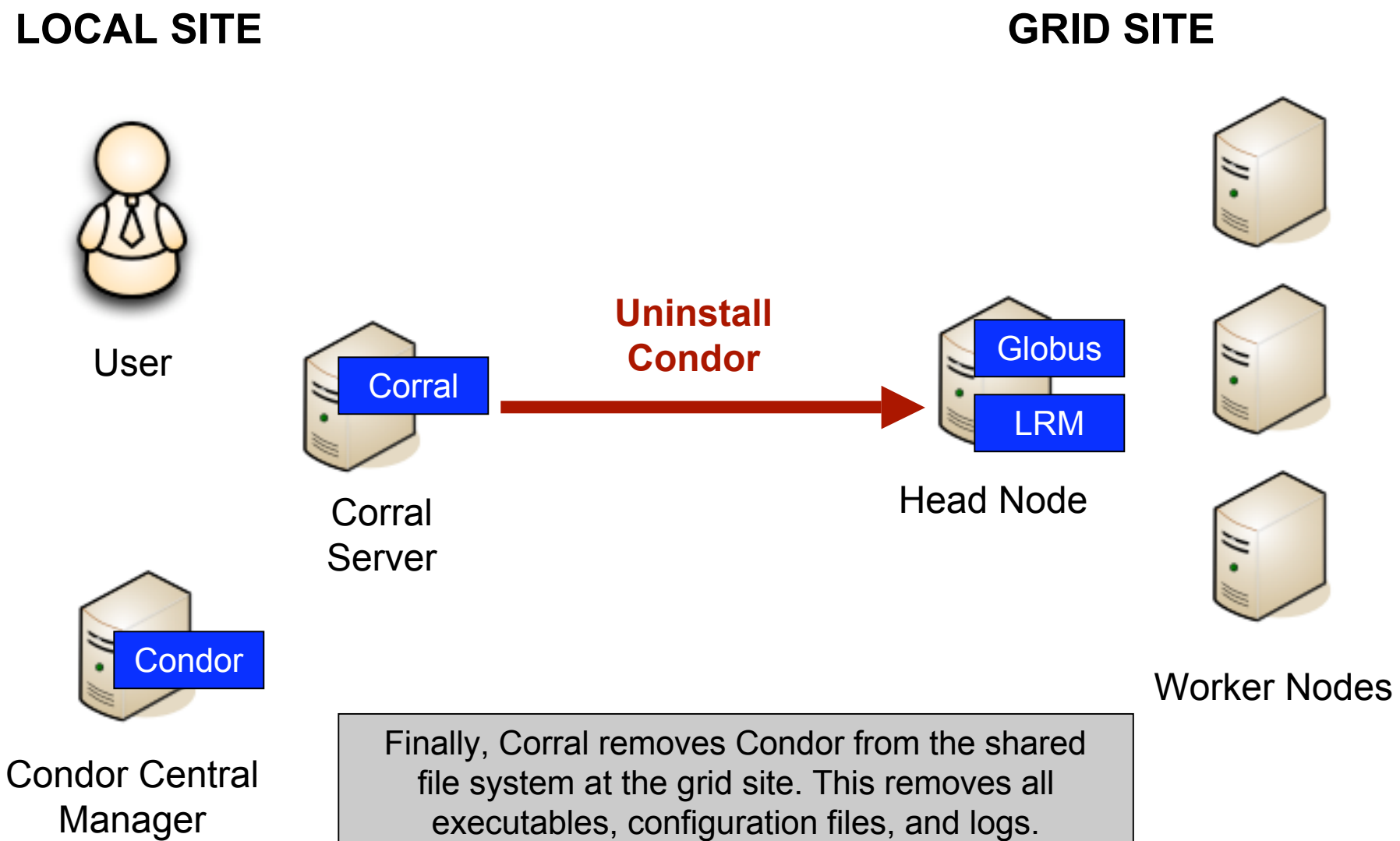


How Corral works



The user can submit multiple glidein requests for a single site. When the user is done with the site they ask Corral to remove the site.

How Corral works



Corral Features

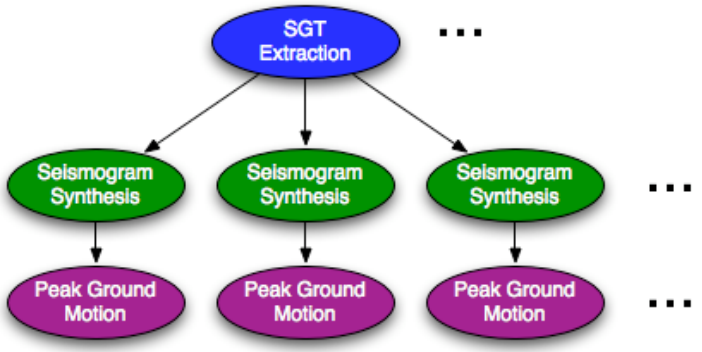
- Auto-configuration
  - Detect architecture, OS, glibc => Condor package
  - Determine public IP (if any)
  - Generates Condor configuration file
- Large requests
  - 1 glidein job = N slots
- Multiple interfaces
  - Command-line, SOAP, Java API
- Automatic resubmission
  - Indefinitely, N times, until date/time
- Notifications
  - Asynchronous API for integration with other tools

Networking Challenges

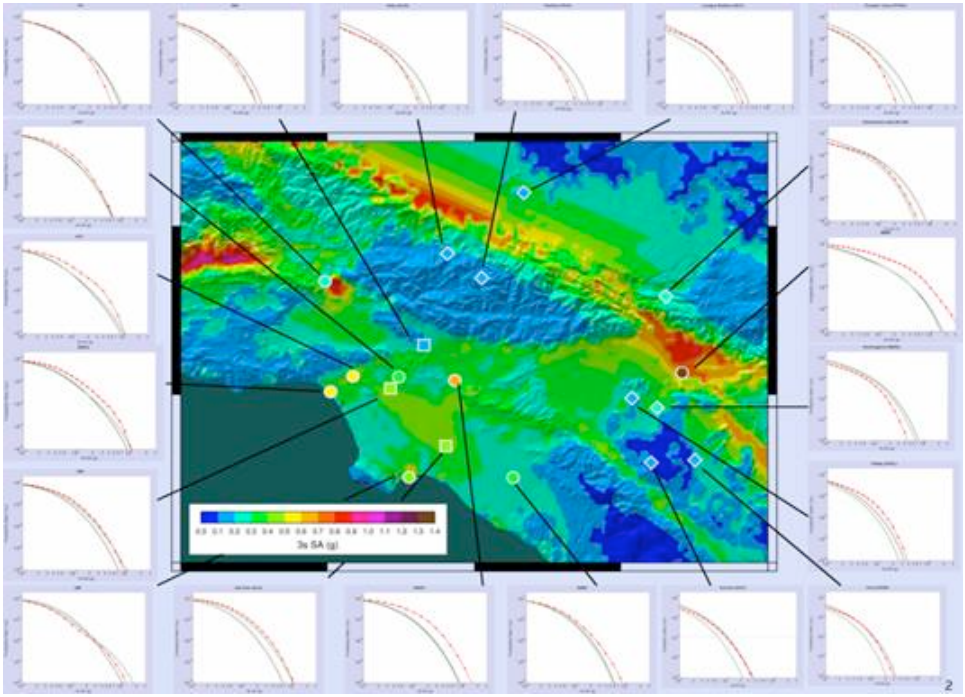
- Firewalls / Private IPs
  - Block communication between glideins and pool
  - Use: GCB/CCB, VPN, or CM on head node
  - Glideins can't be used on some sites
- Port Usage
  - Condor requires many ports
  - Issue for LOWPORT/HIGHPORT firewall holes
  - TCP TIME_WAIT can consume ports
- WAN issues
  - Large glidein pools look like DDoS attacks
  - Traffic gets blocked sometimes

SCEC CyberShake

- Probabilistic seismic hazard analysis workflow
  - How hard will the ground shake in the future?
- Uses Pegasus and DAGMan for workflow management



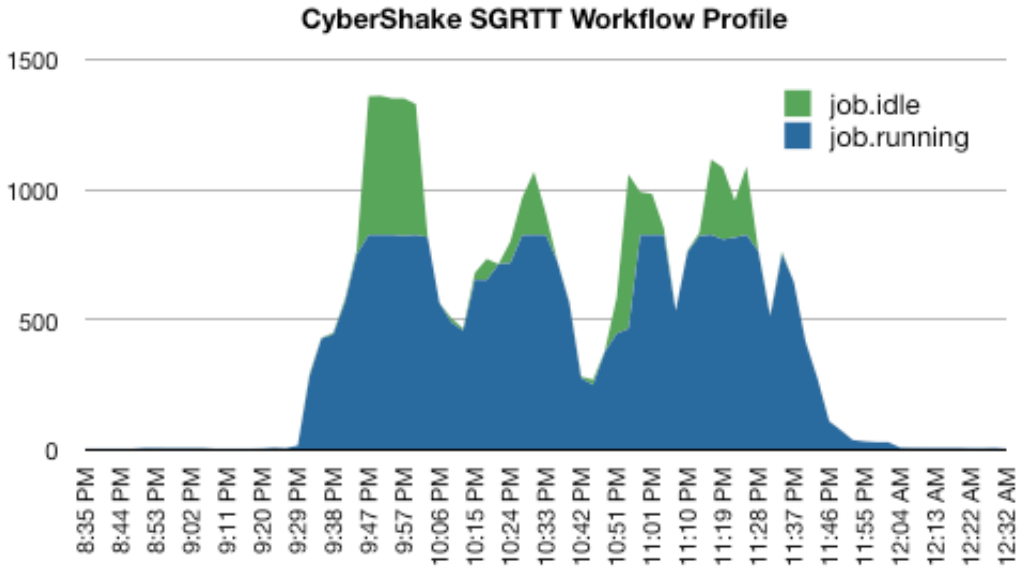
Transformation	Tasks	Runtime (s)
SGT Extraction	7,000	139
Seismogram Synthesis	420,000	48
Peak Ground Motion	420,000	1
Total Tasks:		847,000
Mean Runtime:		25.45



SCEC
an NSF + USGS center

CyberShake Progress

- Using Corral since January
  - Provisioning resources from the TeraGrid
  - Requests: 185
  - Slots: 33,137
  - CPU Hours: 240,496
- Application Progress
  - Jan 2009-Apr 2009
    - Tasks: >11.3M
    - Jobs: >352K
  - May 2009 (planned)
    - Tasks: ~168M
    - Jobs: ~5M



- With glideins a site can be completed in ~3 hours on 800 cores (down from 18+ hours)

Future Work

- Dynamic Provisioning
  - Automatically grow/shrink pool according to application needs
- Support for other features
  - GSI security
  - CCB for firewall traversal (GCB already)
  - Grid matchmaking
  - Parallel universe
- Remote Pool?
  - Deploy Collector/Negotiator/Schedd as well

Try it out

- Website:
  - http://pegasus.isi.edu/corral
- Problems:
  - juve@usc.edu

Questions?