CONDOR And The GRID

By
Karthik Ram Venkataramani
Department of Computer Science
University at Buffalo
kv8@cse.buffalo.edu
Abstract

- Origination of the Condor Project
- Condor as Grid Middleware
- Condor working mechanism
- Matchmaking in Condor
- Condor-G - Interoperability with Globus
- Condor working Universe
  (Shadow, Sandbox, Remote I/O)
Condor
High Throughput Computing

- Resource Management System for Compute Intensive Jobs
- Job Management
- Scheduling Policies
- Resource Monitoring
- Priority Schemes
How does it work??

- User code is re-linked with condor binaries
- Users submit jobs to the condor daemon (with job preferences)
- ClassAd mechanism matches resource requests in queue with machines in the pool.
- Job is executed remotely, with remote system calls if any.
## Grid Architecture

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Applications</td>
<td>4</td>
</tr>
<tr>
<td>Brokering, Diagnostics, Monitoring (Collective Services)</td>
<td>3</td>
</tr>
<tr>
<td>Secure Access to Resources and Services (Resource and Connectivity Protocols)</td>
<td>2</td>
</tr>
<tr>
<td>Computers, Storage Media, Networks (FABRIC)</td>
<td>1</td>
</tr>
</tbody>
</table>
Condor In The Grid

Application, Problem Solver, etc

CONDOR-G

GLOBUS TOOLKIT

CONDOR

Processing, Storage, Networks

Grid Middleware
Kernel Daemons

- Users submit jobs to *agents*
- *Agent* keeps job in persistent storage
- *Agents* and *resources* advertise to *matchmaker*
- *Matchmaker* introduces potentially compatible *agents* and *resources*
- At the *resource*, a sandbox ensures safe execution environment for the job
Condor Pools
(agents, resources and matchmakers)

- Machine running both agent and resource daemons is capable of submitting and running jobs.
- Multiple instances of agents can be run on the same machine
- Agents, resources and matchmakers are independent (individually enforce their owners policy)
- Matchmaker enforces the community policy
Flocking

- Facilitates sharing across organizations
- Allows jobs that cannot be run immediately to run in a different condor pool.
- Gateway Flocking
Gateway Flocking

- Structure of existing pools is preserved
- Gateways advertise free resources between themselves
- Flocking is subject to *matchmaker* policies. (may not be bidirectional)
- Flocking is transparent to participants
- Sharing is only at organizational level
- Individual user cannot join multiple pools.
Direct Flocking

- An *agent* can report to multiple *matchmakers*
- Flocking depends on user initiatives.

```
Condor Pool B

R  R  R
  M  

Condor Pool A

R
  M  R
  A  R
```
CONDOR-G
Inter-operability with GLOBUS

- Condor Pool is divided into two
  1. Job Management
  2. Resource Management
- Submit Machine - Job Management
- Execute Machine - Resource Management
- Condor-G is the Job Management part of Condor
Job Management using CONDOR-G

- Submit jobs into Queue
- Maintain Log files detailing Job life cycle
- Manage Input, Output files
- Monitor queued/running jobs
- Notifications of Job status
- Fault-Tolerant
Condor-G and Globus

- Authentication
- Remote Program Execution, Data Transfer
- Uses Globus protocols to access resources at multiple sites
- Substitute for `globusrun`
- Maintain Globus credentials which might expire while job is running
Condor-G with GLOBUS Protocols

- GSI (Grid Security Infrastructure) for authentication (single sign-on)
- GRAM (Grid Resource Allocation Management) Submission of computational request to remote resource and subsequent monitoring
- GASS (Global Access to Secondary Storage) Data Transfer to/from remote machine of the executable and stdin, stdout files.
- RSL for Job Specification
Accessing the Grid with Condor-G

Job Submission

user

Condor-G Scheduler

Condor-G Grid Manager

Job Execution Site

Globus Gatekeeper

Globus Job Manager

Site Job Scheduler (PBS, Condor, LSF)

Job x

Job y
Condor Planning & Scheduling using *matchmaking*

- Centralized scheduling algorithms are inefficient for Grids.
- *Agents and Resources* advertise their characteristics to *matchmakers*.
- *Matchmakers* scan *ClassADS* and creates *agent-resource* pairs.
- *Agents* then claim the resource in a separate step.
ClassADS

- Set of uniquely named expressions called *attributes*
- No Specific schema
- Each attribute has a *name* and a *value*
- *Attributes* are evaluated using three-valued logic (true, false or undefined)
- *Requirements* and *rank* are pivotal attributes
ClassAds ...

- *Requirements* indicate constraints and *rank* measures the desirability of a match.
- For two ClassAds to match their *requirements* should evaluate to *true*. 

```
Matchmaker
<table>
<thead>
<tr>
<th>Matchmaking algorithm(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>advertisement(1)</td>
</tr>
<tr>
<td>Agent</td>
</tr>
<tr>
<td>claiming(4)</td>
</tr>
<tr>
<td>Notification(3)</td>
</tr>
<tr>
<td>Resource</td>
</tr>
<tr>
<td>advertisement(1)</td>
</tr>
</tbody>
</table>
```


Sample ClassAds

Job ClassAd
[
  MyType="job"
  TargetType="machine"
  Requirements=
  ((other.Arch=="INTEL" &&
    Other.OpSys=="Linux")
  Rank=(Memory *10000)+KFLOPS
  Owner="karthik"
]

Machine ClassAd
[
  MyType="Machine"
  TargetType="Job"
  Machine="xyz.ccr.buff.edu"
  Requirements=
  (LoadAvg <= 0.300000)
  Rank=other.depart==self.depart
  OpSys="LINUX"
  Arch="INTEL"
]
Condor Problem Solvers

- Higher level structure built on top of the Condor Agent
- Provides a unique programming model for managing jobs
- Blindly trusts *Agents* for reliability
- Deals with only application specific details of ordering and task selection
- Runs as a normal condor job at submission site
Master-Worker

- System for problems of indeterminate size on a unreliable workforce.
- Eg. Parameter searches where problem space is huge and can be examined independently.
- Master Process directs the computation, with assistance of worker processes (compute nodes)
Directed Acyclic Graph Manager

- Service for executing multiple jobs with dependencies
- Similar to make, accepts a declaration that lists the work to be done and the constraints.

Job A a.condor
Job B b.condor
Job C c.condor
Job D d.condor
Job E e.condor
PARENT A CHILD B C
PARENT C CHILD D E
SCRIPT PRE C in.pl
SCRIPT POST C out.pl
DAGMAN ...

- Job statement associates an abstract name with a .condor file which describes a complete condor job
- PARENT-CHILD statement describes relation between two or more jobs
- PRE and POST jobs are run by DAGMan on the submitting machine
Split Execution

- Facilitates successful remote execution of jobs.
- **Shadow** - represents the user to the system. Provides everything needed to specify the job at runtime (executable, arguments, input files, environment etc)
- **Sandbox** - Responsible for providing the a safe execution environment for jobs.
- A matched Sandbox and Shadow form the *universe*
Standard Universe

- Only universe supplied by early versions of condor
- The goal is to faithfully reproduce users home POSIX environment at remote site
- Emulates vast majority of system calls
- Supports checkpointing, facilitating process migration
Recent additions to Condor

Earlier, entire JVM was submitted as a standard universe job for executing Java Programs

The new Java universe provides a complete Java environment

All Java runtime components are placed in a private execution directory along with user credentials.
Java Universe ...

- **Shadow I/O Server**
  - Local
  - Sys calls
  - Home fs

- **Job Setup**
  - RPC

- **Sandbox I/O proxy**
  - Fork
  - Wrapper
  - Job
  - I/O library

- **JVM**

---

- **Local**
- **Sys calls**
References

- Grid Computing
  Fran Berman, Geoffrey Fox, Anthony Hey

- Condor Project
  University of Wisconsin at Madison
  http://www.cs.wisc.edu/condor