Condor and the Grid

D. Thain, T. Tannenbaum, M. Livny

Presented by

Govind Kothari

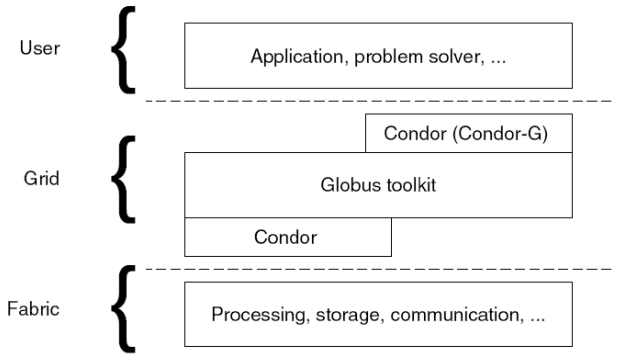
Condor: a system for high throughput computing

- Resource finder
- Batch queue manager
- Scheduler
- Checkpoint/Restart
- Process migration
- Remote system calls

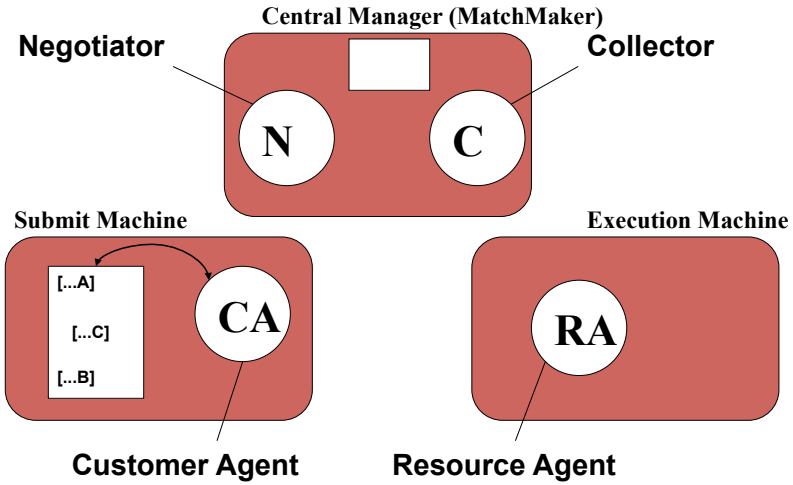
All jobs

Jobs linked with the Condor library

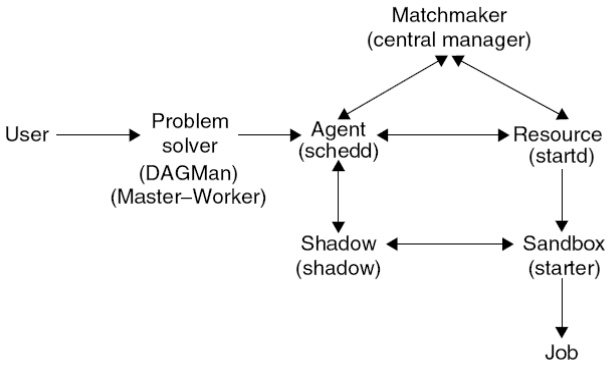
Condor-G: a computation management agent for Grid Computing



Condor System Structure



Condor - Kernel



Major processes in a Condor System

Customer Agent

- Maintains queue of submitted jobs
- Advertises status
- Selects jobs to run

Resource Agent

- Monitors system status
  - Load average
  - Keyboard and mouse idle time
  - Memory, disk space, ...
- Advertises status
- Listens for requests to run jobs

Central Manager

- Collector
  - Accepts ads from resource agents and customer agents
- Negotiator
  - Matches customers with resources
- Accountant
  - Records resource usage by customers

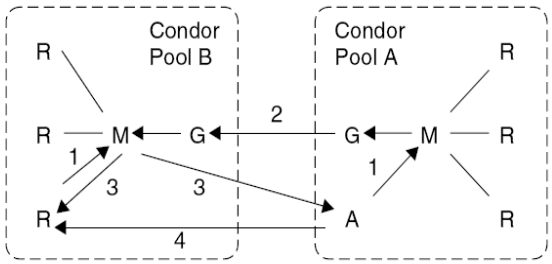
Condor-Kernel

- User submits job to *Agent*
- *Agent* is responsible for finding *resource* for user
- *Agents* and *resources* advertise themselves to a *matchmaker*
- *Matchmaker* is responsible for finding compatible *agents* and *resources*
- *Agent* starts a *shadow process* and *resource,* a *sandbox*
- *Shadow is responsible for details to execute the job*
- *Sandbox* is responsible for creating a safe execution environment

Sharing across organizations

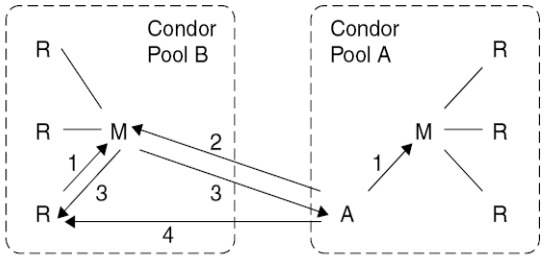
- Gateway flocking
- Direct flocking

Gateway flocking



- Gateway pass the information (detects idle agents or resources) about participants between the pools
- Advantage
  - Completely transparent to the User
- Disadvantage
  - Since each pool is represented by single gateway, the accounting of use by individual remote users is impossible
  - Does not permit individual user to join multiple communities

Direct flocking



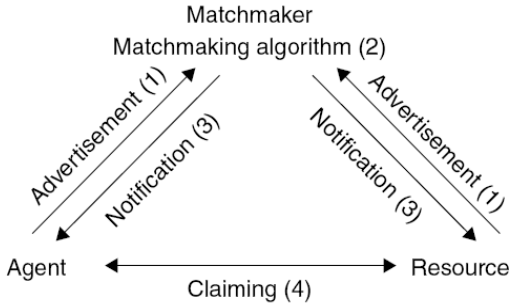
Direct flocking

- An agent may simply report to multiple matchmakers
- Advantage
  - Requires only agreement between one individual and another organization
- Disadvantage
  - Only helps users who take initiative

Planning and Scheduling

- Planning: Acquisition of resources by users
  - Response time, turnaround time, throughput of job
- Scheduling: Management of resource by its owner
  - Efficiency, utilization and throughput
- Condor uses matchmaking to bridge the gap between planning and scheduling

Matchmaking



Sample class adds for Condor

Job ClassAd

```
[
MyType = "Job"
TargetType = "Machine"
Requirements =
((other.Arch=="INTEL"&&
other.OpSys=="LINUX")
&& other.Disk > my.DiskUsage)
Rank = (Memory * 10000) + KFlops
Cmd = "/home/tannenba/bin/sim-exe"
Department = "CompSci"
Owner = "tannenba"
DiskUsage = 6000
]
```

Machine ClassAd

```
[
MyType = "Machine"
TargetType = "Job"
Machine = "nostos.cs.wisc.edu"
Requirements =
(LoadAvg <= 0.300000) &&
(KeyboardIdle > (15 * 60))
Rank = other.Department==self.Department
Arch = "INTEL"
OpSys = "LINUX"
Disk = 3076076
]
```

Two sample ClassAds from Condor

Matchmaking - features

- ClassAds require no fixed schema
- Different algorithms
  - Gang matching
    - permits the coallocation of more than once resource, such as a license and a machine
  - Collections
    - provide persistent storage for large numbers of ClassAds with database features such as transactions and indexing.
  - Set matching
    - permits the selection and claiming of large numbers of resource using a very compact expression representation
  - Indirect references
    - permit one ClassAd to refer to another and facilitate the construction of the I/O communities

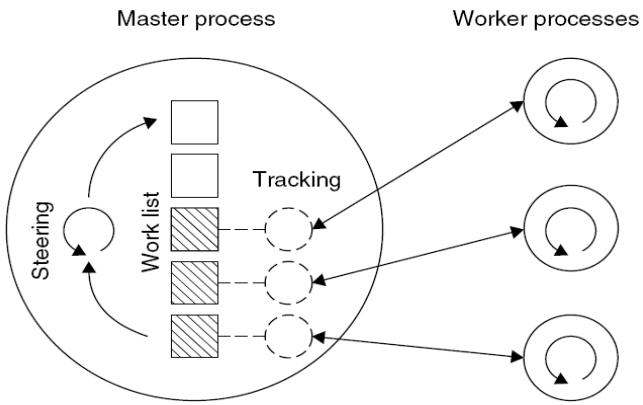
Problem Solvers

- It's a high level structure built on top of the Condor Agent
- Problem solver need only concerned with application specific details of ordering and task selection
- Two types provided with Condor
  - Master-Worker (MW)
  - Directed Acyclic Graph Manager (DAGMan)
- Each provide unique programming model for managing large number of jobs

Master-worker

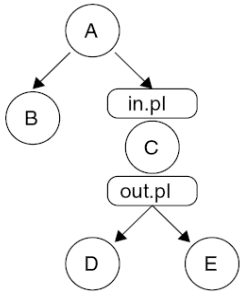
- Suitable for problems with indeterminate size on large and unreliable workforce
- Three components
  - Work list
    - Record of all outstanding work
  - Tracking module
    - Accounts of remote worker processes and assigns them uncompleted work
  - Steering module
    - Directs computation by examining results, modifying the worklist and communicating with Condor to obtain sufficient number of worker processes

Structure of Master-Worker Program



Directed Acyclic Graph Manager (DAGMan)

- Service for executing multiple jobs with dependencies in a declarative form




```
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
RETRY C 3
```

Split Execution

- Once a job is allocated a particular resource, lot of problems can arise
  - Absence of required files
  - Firewall issues
  - Credentials to access the data
- Only the ***execution machine*** knows what file systems, networks, and databases may be accessed and how they must be reached
- Only the ***submission machine*** knows at run time what precise resources the job must actually be directed to.
- Co operation is required – it is called ***split execution***

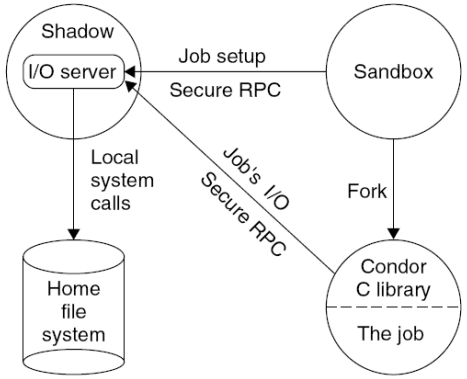
Split Execution

- Split Execution is accomplished by two distinct components: *shadow* and *sandbox*
- *Shadow:* represents user to the system
- Sandbox: creates right environment for the job
  - Two components
    - Sand: provide whatever is needed by the job
    - Box: protect the resource from any harm that a malicious job might cause
- *Universe*: It is defined by matched sandbox and shadow

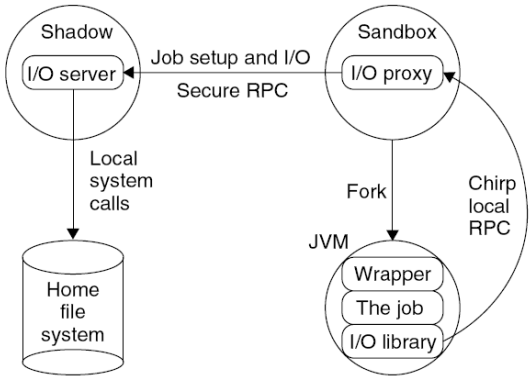
Universe

- Different types of Universe provided by Condor
  - Standard Universe
  - Java Universe

Standard Universe



Java Universe



Conclusion

- Condor has been successfully used for solving real world problems
- Examples
  - Micron Technology Inc
  - C.O.R.E. Digital pictures

Thank You