Condor

A Distributed Job Scheduler

Mikkel Bystrup Stensgaard (s001434)
Matias Dons Dollerup (s991367)

Agenda

- Introduction
- Scheduling
- Universes
- Architechture
- Administration
- Conclusion

Introduction to Condor (1)

- Condor is a distributed job scheduler
- Provides environment for handling several jobs simultaneously
- Provides snapshot mechanism for restoration and migration of jobs

Introduction to Condor (2)

- A Condor pool is a set of grids, opportunistic worksations and dedicated workstations
- Multiple pools can be connected through Condor
- Condor takes advantage of opportunistic workstations to minimize idle time

Introduction to Condor (3)

- Example:

  A Condor cluster at UW-Madison Department of Sciences containing more than a 1000 workstations including a 500 CPU Beowulf cluster, delivers around 650 CPU-days on a typical day.

Condor advertising (1)

- Condor operates through an advertising interface
- Both jobs and workstations advertise their presence in the pool
- A job advertisement presents the specifications for a job
- A workstation adverticement presents the capabilities of a workstations

Condor advertising (2)

- Jobs are specified in a Job Description File that Condor uses to create an advertisement
- The main fields in a description file is the requirements and the ranking expressions
- Expressions is given by a logical syntax, hence allowing very specific formulation

Condor advertising (3)

- Example: A job description file

```
universe = vanilla       # select runtime environment
executable = some_job
requirements = (Arch=="INTEL" && OpSys=="LINUX") #target
rank = (Memory * 10000) + KFlops                 #target
arguments = -verbose
input = in.dat           # redirect to stdin
output = out.dat         # redirect to stdout
log = log.txt
queue                    # add job to queue
```

Job scheduling in Condor

- Jobs can be added from any terminal within the pool
- Condor matches job advertisements with machine advertisements and schedules execution

Universes in Condor

- **Condor provides scheduling for a variety of applications by using different execution environments called universes.**
  - Vanilla Universe: Sequencial programs based on a shared file system.
  - MPI Universe: Message Parsing programs, based on process parallelism. These jobs is only run on Beowulf clusters – not on the opportunistic workstations.
  - Parallel Virtual Machine - PVM Universe: Supports jobs where machines can enter and/or leave during job execution.
  - Globus Univers: Grid computing environment.
  - Standard Universe: Special Condor environment uptimized for effective job scheduling.

Standard Universe

- Application checkpointing enables process migration and process restoration
- Requires the application to be linked with Condor libraries
- Limitations: (running in user mode)
  - Multi-process jobs are not allowed including multiple kernel threads
  - Network communication must be brief
  - No interprocess communication (shared memory, pipes, semaphores)

Checkpointing (1)

- a ”snapshot” of the jobs current state
- Freedom to preempt job
  - Rescheduling a higher priority job
  - User reclaims non-dedicated PC
  - Enables migration
- Increases fault tolerance

Checkpointing (2)

- Condor writes the following to a file/socket
  - CPU state, including registers
  - Stack
  - State of all open files
  - Signal handlers
  - Pending signals

Program must be linked with the Condor library!

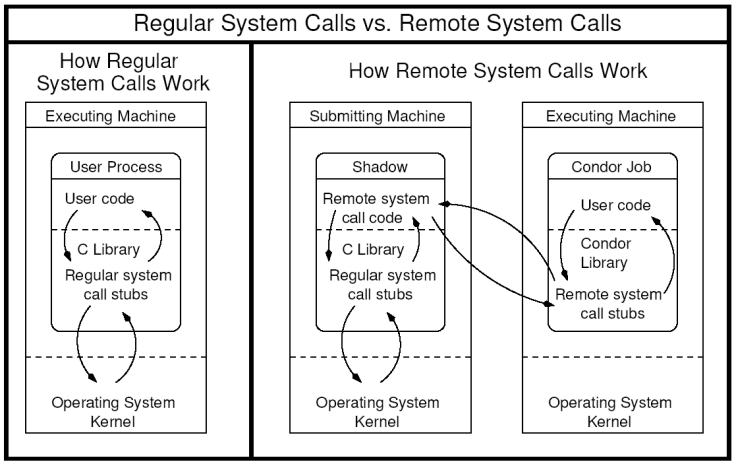
Access to data

- Shared file system
  - Common on Beowulf cluster
  - Must be specified as a requirement expression
- Condor File Transfer mechanism
  - Condor automatically transfer specified files before starting the job. Also transfer output files back when job are finished of pre-empted.
- Remote I/O calls
  - Only possible in the Standard universe

Remote I/O operations

- Shared file system
  - Bandwidth to central server limits possible number of nodes. (or performance)
  - Requires account on file server
- Condor Remote System operations
  - No special account. (easier administration)
  - Open/Read/write operation are handled on submitter machine.
  - Implemented by replacing call stub

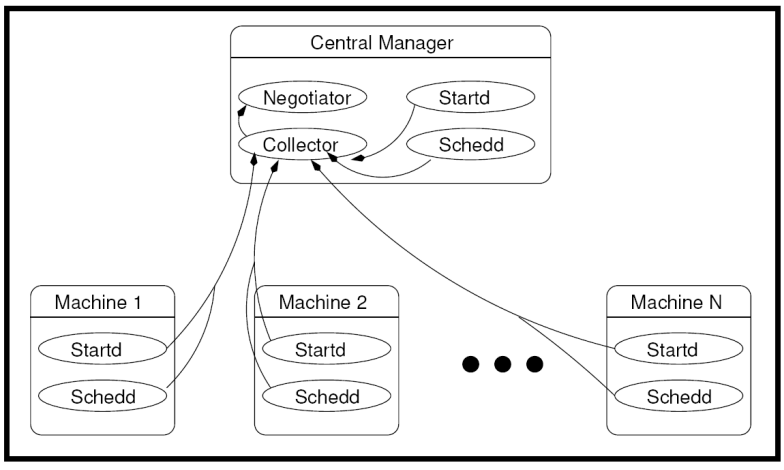
Remote I/O operations



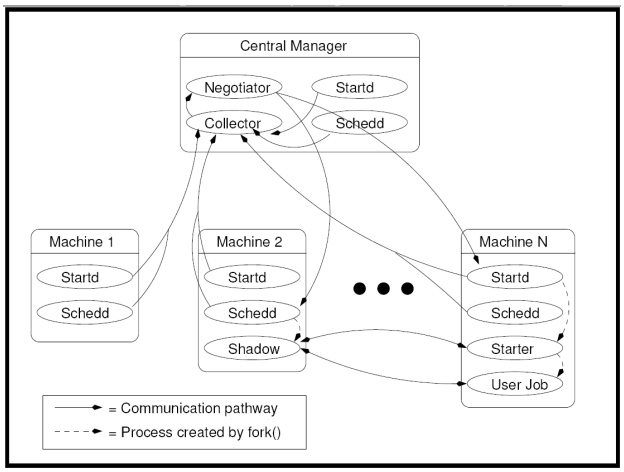
Architecture

- One central manager
  - Collects Class-adds
  - Match jobs to machines
- Nodes
  - Contains local job-queue
  - Negotiates with central manager

Deamons, Idle



Deamons, job



Deamons (1)

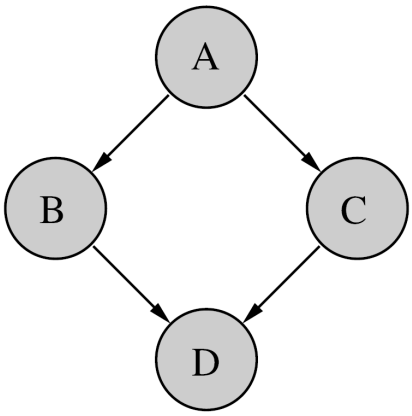
- *"condor_master"* process runs on each node
  - Can restart other deamons
  - Enable network update of binary files
  - Inform administrator of problems
  - Enable remote start/stop of deamons

Deamons (2)

- *“start_d”*
  - Enables the node to excecute jobs
- *”condor_schedd”*
  - Manages local job-queue
- *”condor_shadow”*
  - *Manages file-transfers, logs etc.*

Dependencies

- Dependencies are specified as a Directed acyclic graph (DAG)
- Submitted via the DAGMan scheduler

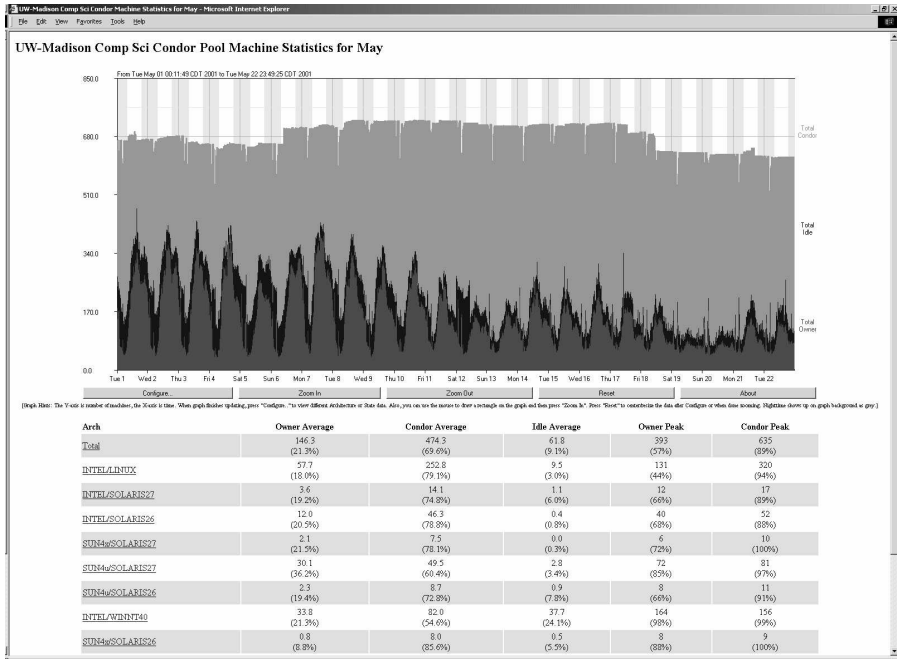


SMPS in condor

- Represented as multible virtual machines
- Possibilty to manually specify resources for each VM
  - Cpu's
  - Memory
  - Swap
  - Disk

Administration

- Rich set of tools for administration
- Includes
  - Status of pool, including detailed information about each node. (current load, memory etc)
  - Display job stats, including finished jobs
  - Changing scheduling policy on-the-fly



UW-Madison Comp Sci Condor Pool Machine Statistics for May

Arch	Owner Average	Condor Average	Idle Average	Owner Peak	Condor Peak
Total	146.3 (21.3%)	474.3 (69.6%)	61.8 (9.1%)	393 (57%)	635 (39%)
INTEL/LINUX	57.7 (18.0%)	252.8 (79.1%)	9.5 (3.0%)	131 (44%)	320 (94%)
INTEL/SOLARIS27	3.6 (19.2%)	14.1 (74.8%)	1.1 (6.0%)	12 (66%)	17 (89%)
INTEL/SOLARIS26	12.0 (20.5%)	46.3 (78.8%)	0.4 (0.8%)	40 (68%)	52 (88%)
SUN4u/SOLARIS27	2.1 (21.5%)	7.5 (78.1%)	0.0 (0.3%)	6 (72%)	10 (100%)
SUN4u/SOLARIS27	30.1 (36.2%)	49.5 (60.4%)	2.8 (3.4%)	72 (85%)	81 (97%)
SUN4u/SOLARIS26	2.3 (19.4%)	8.7 (72.8%)	0.9 (7.8%)	8 (66%)	11 (91%)
INTEL/WINNT40	33.8 (21.3%)	82.0 (54.6%)	37.7 (24.1%)	164 (98%)	156 (99%)
SUN4u/SOLARIS26	0.8 (8.8%)	8.0 (85.6%)	0.5 (5.5%)	8 (88%)	9 (100%)

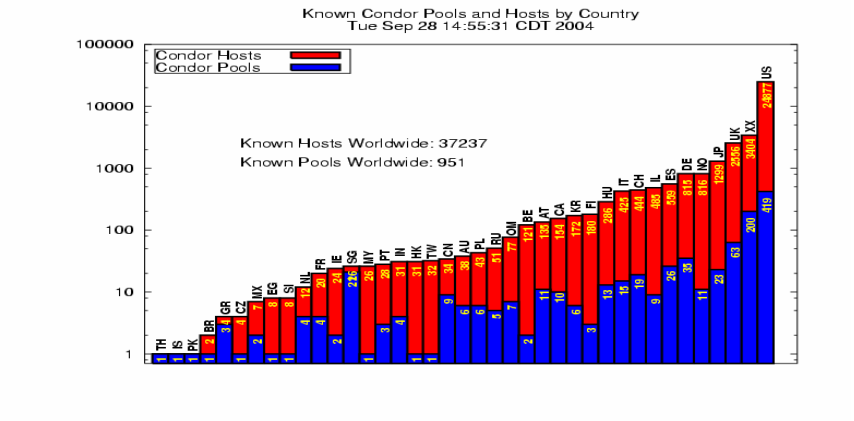
Conclusion

- Powerfull tool for scheduling jobs across multible platforms
- Works with deticated clusters such as beowolf
- Utilizes resources on non-dedicated PCs
- Includes checkpointing and job migration.

More information

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

Known Condor Pools



Questions?

42