Condor—A Hunter of Idle Workstations

M.L. Litzkow, M. Livny and M. W. Mutka

Presented by

Priya Ranjan

CS816S, Thursday, Feb 19, 2004
Motivation

1. Workstations used efficiently by individual users?
2. Usage patterns?
3. Increasing the efficiency of workstation-cluster?
4. Any ideas for management of idle workstation capability?
5. Remote execution facilities?
Condor—A Hunter of Idle Workstations

Introduction

1. A rudimentary capacity scheduling system designed in 1988
2. Aims to optimize the utilization of workstations
3. Protecting the integrity of local users and their activity
4. Outlines the design, implementation and performance of Condor scheduling system

Current Status is @http://www.cs.wisc.edu/condor/

The goal of the Condor Project is to develop, implement, deploy, and evaluate mechanisms and policies that support High Throughput Computing (HTC) on large collections of distributively owned computing resources. Guided by both the technological and sociological challenges of such a computing environment, the Condor Team has been building software tools that enable scientists and engineers to increase their computing throughput.
The Condor® Project Homepage

The goal of the Condor Project is to develop, implement, deploy, and evaluate mechanisms and policies that support High Throughput Computing (HTC) on large collections of distributively owned computing resources. Guided by both the technological and sociological challenges of such a computing environment, the Condor Team has been building software tools that enable scientists and engineers to increase their computing throughput.

Note: We're hiring a system programmer to work on the Virtual Data Toolkit. For more information, see the PVL (Position Vacancy Listing).

Recent News

(Feb 2004) Condor 6.6.1 released!
6.6.1 is the latest version of our current stable release series. It includes full support (with checkpointing) on Linux Red Hat 8.0 and 9.0 on x86 platforms. It also contains bug fixes, scalability improvements, and other changes. See the Version History and Release Notes for details. Condor 6.6.1 is available from our Download page, which has links to the UW servers and our European mirror site (where the binaries will be mirrored shortly).

(Feb 2004) Parody-Condor Week April 14-16th, 2004!
The annual Condor meeting will take place as part of Parody-Condor week at the Pluto Center on University of Wisconsin, Madison campus on April 14th-16th, 2004. You can learn more on the conference web page.
Condor—A Hunter of Idle Workstations

Three issues addressed

1. Analysis of workstation usage patterns?
2. Design of remote capacity allocation algorithms
3. Development of Remote execution facilities

Understanding of these three can give you a leverage of up to 2000..... What is leverage?

\[
\text{Leverage} = \frac{\text{Capacity gained by remote execution}}{\text{Capacity consumed locally to support it}}
\]

1 hour of remote execution by investing 30 seconds of local support then leverage will be:

\[
\text{Leverage} = \frac{3600}{30} = 120
\]
System Setup and Specifications for the project

1. 100 VAX station II capable of remote execution
2. Light and heavy users (Load-balancing, neural networks, combinatorial optimization problems etc.)
3. Transparent submission of background jobs
4. Automatic restarting of job by Condor
5. Protect the integrity of local user
6. Mechanism should consume minimum possible local capacity
Condor—A Hunter of Idle Workstations

Scheduling Structure: Centralized Vs Distributed

- Centralized processor gathers information and schedules things accordingly. Hard to scale/extend and one point of failure
- Distributed mechanism will have a contention for remote processing cycles. Negotiations may be chatty!
- Hybrid Approach in Condor—Best of both worlds?
- One central coordinator: Only assigns capacities!

- Workstations keep the state-info and schedule their jobs, and also decide the priority of their own jobs
Condor—A Hunter of Idle Workstations

Scheduling structure
Local Scheduler and a Local Queue
Global coordinator

Figure 1: The Condor Scheduling Structure.
Condor—A Hunter of Idle Workstations

Scheduling mechanism:

- Central coord. polls every 2 min for available CPU and also jobs waiting
- Local scheduler decides if it can volunteer its CPU
- Local scheduler also checks every 30 Sec for local activity to free the workstation from any remote activity
- Central coord. allocates capacity to local schedulers if they have a job waiting

- Keep it SIMPLE and ROBUST!
- Central coord. consumes at most 1% of CPU on a workstation
Remote Unix (RU):
- Transforms idle workstations into cycle servers
- RU invoking starts a shadow process on local machine as a surrogate of remote process
- Remote system calls can be viewed as remote procedure calls
- Checkpointing is the most powerful feature of RU!
  - Checkpointing is the saving of the state of a program during its execution for restarting purposes.
- Saves text, date, bss, and the stack segments.

Very useful in condor as a checkpointed program can be moved to the next available workstation
Condor—A Hunter of Idle Workstations

Fair access to remote capacity:

- Heavy users vs Light users
- Up-down algorithm for fair access
- Maintains a schedule index for every workstation
- Index increases if one gets capacity
- Index decreases if one is denied capacity
- Maintain a balance between waiting time and capacity allocation
# Performance: User Profiles

<table>
<thead>
<tr>
<th>User</th>
<th>Number of Jobs</th>
<th>% of Jobs</th>
<th>Demand/Job (in Hours)</th>
<th>Demand (in Hours)</th>
<th>% of Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>690</td>
<td>75</td>
<td>6.2</td>
<td>4278</td>
<td>90</td>
</tr>
<tr>
<td>B</td>
<td>138</td>
<td>15</td>
<td>2.5</td>
<td>345</td>
<td>7</td>
</tr>
<tr>
<td>C</td>
<td>39</td>
<td>4</td>
<td>2.6</td>
<td>101</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>40</td>
<td>4</td>
<td>0.7</td>
<td>28</td>
<td>0.6</td>
</tr>
<tr>
<td>E</td>
<td>11</td>
<td>1</td>
<td>1.7</td>
<td>19</td>
<td>0.4</td>
</tr>
<tr>
<td>Total</td>
<td>918</td>
<td>100</td>
<td>5.2</td>
<td>4771</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 1: Profile of User Service Requests.
Condor - A Hunter of Idle Workstations

Performance: Profile of Service Demand

Figure 2: Profile Of Service Demand.
Performance: Queue Length & Avg. wait ratio

Figure 3: Queue Length.

Figure 4: Average Wait Ratio.
Performance: utilization of remote resources over a month and a week, 25% avg. utilization
Performance impact locally: Rate of Checkpointing

Figure 8: Rate Of Checkpointing.
Condor—A Hunter of Idle Workstations

Overall performance and leverage

Figure 9: Remote Execution Leverage.
Conclusion

1. Possible to design/implement an efficient system to make optimal use of capacity
2. Improved productivity in terms of leverage upto 2000
3. Lessons learnt about jobs which should not be remotely executed!
Motivation

- Network oriented future computing
- Service based
- Adapts to the user’s demand
- Universal access
- Online “optimal” scheduling
- Limited support for computing on web

“A demand based computing can be characterized by its universal accessibility and its ability to make automatic cost/performance trade-off decision at run time”

PUNCH=The Purdue Univ. Network Computing Hub
Two categories of tools:

a) Providing support for global scalability
Ex: ATLAS, Globe, Globus-GUSTO, IceT, ParaWeb etc

b) Tools to access and use globally distributed resources
Ex: CCS, MMM, MOL, NetSolve, Ninf, PUNCH, RCS, VNC etc.

-Source/object code may not be available for commercial applications
-Application Independent framework

“Power multi-user operating system than can do almost everything!”
Characterization of a network computing system

- Interface to the external world
- Internal architecture
- Class of software it can support
- Capabilities of resource management framework

Design Issues:
Scalability
Reliability
Security

2/26/2004
Information management Factors

- Portability
- Run-specific resource usage characteristics like CPU Usage, network usage, memory requirements etc
- Administrative policies and configuration across multiple admin domains
- Dynamic, incremental and distributed nature of information
- Vertically distributed architecture
System Architecture
message passing hierarchy of client, management and execution units
Thin client approach
Demand driven scheduling engines
Optimal resource selection
Policy enforcement
Scheduling of implementation and execution unit
Execution units implement the schedules
Update the system status at management unit
Scalability and Reliability
-Users, multiple front ends
Logical Vs physical accounts

Components as users abstraction

-Software resources
Distribution and replication
-Hardware resources
Request forwarding
-Admin domains
Security and Access control with management and execution units
Scalability and Reliability

- Replication and caching of components for QoS
- Dynamically encoding meta-information into resource identifiers to constant-time access
- Information retrieval from database about access control, run time profile, authentication etc.
- Access codes to allow $o(1)$ access time

Figure 12. Effects of using access codes on authentication time.
Scalability and Access control issues in Punch

- Virtual memory management to control access to resources
- Mapping of virtual to physical resource
- Dependent on user process and User login ID
- Can be used to customize response
The Purdue University Network Computing Hub

Interface issues
- standard www interface?
- State management?
- Access to source/object code?
- Interactive text/graphics based user interface
- File management issues
- Transparency
- Usability

Performance issues
- Reduced execution times
- Efficient use of resources
- How to manage the overhead?

- Punch doesn’t require source/object code
- VNC for GUI
- HTML interface for text based programs
- Run time binding of software/hardware
- Dynamic adaptation via learning the user characteristics
Front End and Scion as Middleware

Figure 6. The PUNCH infrastructure.
User’s view

- Users can upload files
- Run programs
- Download files and view the output
- Background mode of execution
Conclusion:
- Functional demand based network computing infrastructure
- Useful for several hundred student and researchers
- 30 tools from 8 universities and 4 vendors
- One million hits and 70 thousand simulations
- Good response during congestion