UNIVERSIDAD POLITÉCNICA DE CARTAGENA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Estudio, configuración y prueba de un entorno de computación Condor

Autora

María Victoria Bueno Delgado

Director

Pablo Pavón Mariño

Titulación

Ingeniería Técnica de Telecomunicación, especialidad Telemática

Cartagena, Diciembre 2002

Índice

Índice figuras

Índice tablas

Capítulo1

Introducción

1.1 Antecedentes

Desde la aparición de las llamadas “macrocomputadoras” y hasta los actuales “ordenadores personales” miles de ingenieros y especialistas en todo el mundo han centrado su trabajo en resolver las necesidades que surgen en los centros de investigación donde se hace uso de este tipo de máquinas. La razón estriba en que la calidad del trabajo investigador se hace muy dependiente en determinadas ocasiones del rendimiento de procesamiento de las máquinas que se utilizan. Al mismo tiempo, la complejidad del software utilizado en estos entornos se incrementa de forma exponencial. Esta situación lleva consigo un incremento a su vez del tiempo de ejecución de dicho software, lo que acarrea una creciente necesidad de ciclos de CPU.

Una de las alternativas en la búsqueda de una mayor capacidad de computación es la conocida como *High Throughput Computing*, que intenta aprovechar el tiempo de inactividad de microcomputadores convencionales, permitiendo agregar y compartir recursos, transformando una simple red de trabajo en un sistema completo de gestión de recursos y aprovechando al máximo el tiempo de ejecución de las CPUs de las máquinas.

Un ejemplo real de este tipo de entornos es el Área de Ingeniería Telemática de la Universidad Politécnica de Cartagena (UPCT). El entorno de los laboratorios que la forman está orientado hacia el trabajo individualizado de los usuarios de las máquinas. Pero desde el punto de vista del rendimiento computacional, todos esos recursos que se encuentran infrautilizados ya que dichas máquinas no se utilizan por las noches y eso equivale a ciclos de CPU desaprovechados.

De esta situación surgió la idea de utilizar los laboratorios como un entorno de alto rendimiento computacional, aprovechando los tiempos de inactividad de CPU y así poder satisfacer todas esas necesidades de recurso computacional que surgen por parte de las investigaciones del área. Esto permite dar una potente solución con mínimo coste, repartiendo la carga de procesamiento entre un gran número de máquinas. Esta idea permite asimismo conseguir en algunos casos la disminución en tiempo de ejecución de ciertas tareas, cuando la carga puede ser repartida de forma paralela.

Para poder llegar a la solución propuesta en cuanto a las dificultades de procesamiento antes comentadas, aparece la figura de Condor. Condor es una aplicación software creada en la Universidad de Winsconsin-Madison (WM-Madison) con un objetivo: intentar solucionar esos problemas de “rendimiento computacional” que surgen por parte de las tareas que necesitan para su resolución el desarrollo de miles de operaciones, es decir millones de ciclos de CPU.

Condor actúa minimizando las dificultades de gestión compartida de los recursos de CPU, tomando esos recursos de cómputo inutilizados por algunas máquinas y necesitados por otras para la ejecución de las tareas. En este sentido, Condor crea un entorno de trabajo estable, asegurando la ejecución de los programas, independientemente del tiempo de duración de estos y todo ello de forma transparente al usuario.

1.2 Objetivos

El objetivo del proyecto que se presenta se centra en el estudio de las aportaciones que puede ofrecer el software Condor en los laboratorios del Área de Ingeniería Telemática de la UPCT, con el fin de crear un entorno eficiente de aprovechamiento de recursos y así permitir la ejecución de los programas de investigación, asegurando un alto rendimiento computacional y un mínimo coste en tiempo de ejecución.

El estudio debe valorar las distintas alternativas que Condor ofrece, centrándose en el funcionamiento de la aplicación en las redes de trabajo distribuidas, la estructura del software, la necesidad de hardware, configuraciones, entornos de ejecución, formas de ejecutar tareas…y así decidir la instalación y configuración más conveniente para los recursos de los que se dispone.

Una vez realizado el estudio previo se presenta la instalación, configuración, puesta en funcionamiento y prueba de un entorno Condor en una red "prototipo", como paso previo a una futura instalación en un laboratorio del Área de Telemática.

Finalmente, se ha colaborado en el diseño de una herramienta de gestión de simulaciones que está siendo desarrollada dentro del Área, que permite el lanzamiento y recopilación de datos del simulador PASS (*Packet Switch Simulator* –también desarrollado dentro del Área), en ejecuciones sobre un entorno Condor.

1.3 Estructura del contenido

Los capítulos que a continuación se presentan dividen el trabajo realizado de la siguiente manera:

Capítulo 2. Condor. Descripción y funcionalidad. En este capítulo se hace un resumen sobre el concepto y la historia de Condor, introduciendo una breve descripción de la base en la cual se asienta esta aplicación, High-Throughput Computing y Migración de procesos.

Una vez conocida la idea básica de Condor se describen los elementos que forman parte de esta aplicación, y el funcionamiento de los mismos, destacando los diferentes tipos de roles que pueden jugar las máquinas, los entornos de configuración de la versión escogida, los procesos condor que intervienen, los estados y actividades que se pueden dar en las máquinas, una idea general sobre la ejecución de tareas y los procesos que intervienen en dichas ejecuciones.

Capítulo3. Universo Standard. Conocida la Aplicación de una manera más que intuitiva se estudian las alternativas que ofrece Condor en función de las necesidades que se tienen, eligiendo el universo que da nombre al capítulo como entorno de ejecución y desarrollando en profundidad los elementos que lo forman, como Checkpoint, Remote System Calls y aquellos que interesan que convivan con la aplicación por la situación actual de los laboratorios del Área de Telemática, es decir, NFS (Network File System)y NIS (Network Information System).

Por último se estudia el comportamiento a tener con las tareas en este universo, las restricciones que se presentan, la forma de compilar y ejecutar y la necesidad de ficheros dedicados para someter tareas.

Capítulo 4. Universo Standard en una red. Instalación y Configuración. En este capítulo se presenta la red prototipo utilizada para la instalación, configuración, puesta en marcha y obtención de resultados de la Aplicación Condor en vistas hacia una

instalación a mayor escala en uno de los laboratorios del Área de Ingeniería de Telemática. Se da una visión general sobre el software de Condor.

Se indican los aspectos a tener en cuenta para instalar Condor, tales como la instalación de Condor como usuario privilegiado Root, la posibilidad del directorio home distribuido, las partes del sistema de Condor, el espacio de disco requerido… Una vez conocidos los condicionantes, se presenta la solución que se adopta para la red prototipo y se indican los pasos a la hora de instalar, así como la configuración de las máquinas para el correcto funcionamiento en cuanto a la solución propuesta.

Capítulo 5. Universo Standard en una red. Ejecución y obtención de resultados. Una vez instalado y configurado el software de Condor, el siguiente paso es la ejecución de este para comprobar el funcionamiento de la aplicación.

Se propone una tarea ejemplo para someter a ejecución y comprobar los resultados obtenidos. Se comentan algunos comandos y mecanismos de Condor que pueden ser útiles a la hora de manejar la aplicación y se comprueba el funcionamiento y los resultados. Por último se explica la colaboración en el proyecto de una interfaz gráfica que gestiona ejecuciones del simulador PASS (Packet Switch Simulator) desarrollado en el Área de Ingeniería Telemática para que, por medio de dicha interfaz, esas ejecuciones puedan ser manejadas bajo la Aplicación Condor.

Capítulo 6. Conclusiones y líneas futuras. Para finalizar se exponen las conclusiones del trabajo realizado, analizando si los objetivos propuestos han sido cubiertos. Se proponen como líneas futuras el estudio de posibles ampliaciones del entorno de ejecución desarrollado y nuevos mecanismos de Condor. Además, se comentan las novedades que más pueden interesar sobre las últimas versiones del software, en vistas al futuro de este en la instalación que se propone en el Área de Ingeniería Telemática.

Capítulo 2

Condor. Descripción y funcionalidad

2.1 El origen de Condor. High-Throughput Computing

Para poder entender el origen de Condor se deben dar a conocer otros mecanismos que son la base del funcionamiento de esta aplicación. El primer mecanismo es el que se conoce como High-Throughput Computing. Antes de entrar en detalles sobre HTC, se puede definir este como un mecanismo de cálculo de un alto rendimiento de procesamiento.

HTC aparece por el concepto de máximo rendimiento en ciclos por segundo de las CPUs bajo circunstancias ideales. La clave de este mecanismo de alto rendimiento de procesamiento reside en el uso eficiente de los recursos con los que se dispone para trabajar. Todo esto se verá con más detalle en los siguientes apartados.

2.1.1 Origen de HTC

El origen de HTC viene dado por las necesidades de muchos científicos a la hora de resolver sus problemas de investigación en términos de computación. Para la gran mayoría de ellos, la calidad de sus investigaciones es muy dependiente del rendimiento de procesamiento de las computadoras que se utilizan para dichas investigaciones.

Hoy en día es normal encontrar ciertos problemas de cómputo que requieren para su resolución semanas o incluso meses de ejecución de procesador. Los científicos implicados en este tipo de investigaciones necesitan un entorno de ejecución que sea capaz de soportar grandes cantidades de tareas y por supuesto que sean capaces de soportar grandes periodos de procesamiento de dichas tareas. Tal entorno de ejecución es lo que se conoce como High-Throughput Computing (HTC) [1].

Hace años, la comunidad científica confió en las llamadas macrocomputadoras. Estas máquinas trabajaban en procesamiento paralelo, usando microprocesadores ultrarrápidos que operaban en sincronía para resolver problemas complejos. Para el uso de dichas computadoras se necesitaban numerosos grupos de investigación que debían reunir sus recursos financieros para poder hacer frente a un material tan costoso. Por ello no era raro el encontrar una única macrocomputadora que cubriera las necesidades de varias instituciones de investigación.

En un principio el uso de estas macrocomputadoras parecía ser el mejor recurso del que se podía disponer a la hora de resolver grandes problemas de computo, pero este mecanismo de trabajo hacía que los investigadores de las distintas instituciones tuvieran que organizarse en turnos de trabajo para poder usar la microcomputadora. Además de este inconveniente, tenían un tiempo asignado de uso, para administrar el uso de la microcomputadora entre todos, por lo que para algunas de sus tareas debían de limitar su tamaño para cerciorarse de terminar en el tiempo asignado. Otra dificultad que se añadía a este recurso era que los investigadores no tenían acceso continuo a la información que se encontraba en dicha máquina. Esta situación de trabajo era

bastante incomoda para los usuarios de la macrocomputadora, aunque era lo más eficiente que se podía tener en ese momento.

Conforme fueron pasando los años, las computadoras comenzaron a ser más y más pequeñas, y más y más económicas, por lo que los científicos decidieron comenzar a adquirir para sus laboratorios los llamados "ordenadores personales". A partir de ese momento un científico o un grupo pequeño de investigadores mediante una red de trabajo podía producir un recurso, que se podía computar y tenerlo en la red por si se necesitaba, solucionando así los problemas que añadía la macrocomputadora, incluyendo el problema de acceso a los recursos, ya que en este caso el acceso era exclusivo.

2.1.2 HTC vs HPC

En contraste con High-Throughput Computing, aparece otro mecanismo conocido como High-Performance Computing (HPC) [2]. Este es un mecanismo de cálculo de alto rendimiento de procesamiento pero para periodos cortos de tiempo. Existen muchos factores que diferencian a HTC de HPC. A continuación se comentan algunos de ellos para dar constancia de porqué se utiliza HTC en Condor.

HPC puede ser usado para soportar decisiones o aplicaciones de tiempo limitado. Sin embargo, para hacer análisis altamente sensibles, estudios o simulaciones que necesitan para su ejecución largos periodos de tiempo se necesita un mecanismo como HTC. Esta es una de las ventajas que ofrece HTC en cuanto a HPC, pero existen muchas más.

El software que opera con un mecanismo de alto rendimiento computacional HTC, en lugar de con un mecanismo HPC, es capaz de organizar las máquinas en cluster llamados pools, en el caso de Condor (apartado 2.3.3), o colecciones de clusters [54].

También ocurre que los entornos de HPC se miden a menudo en términos de operaciones de coma flotante por segundo (FLoating points Operations per Second, FLOPS) [3]. Muchos científicos no tratan hoy en día las tareas FLOPS, ya que los problemas que esto trae consigo son de gran importancia. Las tareas en las que se centran los investigadores son aquellas que ocupan un largo periodo de tiempo de procesamiento para su resolución y un alto rendimiento de procesamiento, es decir las tareas FLOPY (FLoating Point Operations Per Year) [4] bajo el entorno HTC.

Aquí es donde actúa la aplicación Condor. Esta trabaja en un cluster detectando las máquinas que se encuentran sin procesar trabajos en la red y seleccionándolas para ejecutar ciertas tareas. Si el propietario de la misma vuelve, Condor migra la tarea hacia otra máquina. Todo esto se explicará más detalladamente en el apartado 2.3.

2.1.3 HTC en el presente

El rendimiento de cómputo hoy en día en una institución en su totalidad es bastante mejor, pero los recursos disponibles para los usuarios de las máquinas siguen teniendo dificultades. Mientras que este entorno es el más conveniente para los usuarios de las máquinas, es también el menos eficiente. Muchas máquinas tienen que trabajar con tareas durante un largo periodo de tiempo, mientras que los usuarios de estas necesitan realizar a la vez otras cosas en dichas máquinas.

Del intento de solucionar dichas dificultades aparece la figura de Condor, solucionando en cierta medida los problemas sobre el uso de los recursos, tomando esos recursos de cómputo inutilizados por unas máquinas y necesitados por otras para la ejecución de las tareas y sometiéndolos en un entorno High-Throughput Computing.

Este comportamiento por parte de los centros de investigación y científicos ha llevado a que hoy en día se encuentren en sus instituciones, en vez de una gran computadora, cientos de ordenadores personales que trabajan de forma distribuida. Como ejemplo de este cambio por parte de los científicos se citan a continuación algunos de los proyectos que se han llevado a cabo en el mundo gracias al uso del High-Throughput Computing y a las aplicaciones que trabajan como Condor:

- La extensión a recursos conectados a través de Internet.
- La red Entropía que agregó en dos años 30.000 ordenadores, logrando por ejemplo calcular el mayor numero primo conocido [5].
- El sistema SETI que funciona en más de medio millón de PCs analizando los datos del radio telescopio Arecibo a la búsqueda de señales de inteligencia extraterrestre [6].

La siguiente fotografía muestra el ejemplo de un centro de investigación donde se ha pasado del uso de una macrocomputadora al uso de decenas de ordenadores personales trabajando en un entorno de High-Throughput Computing.

Figura 2. 1 Entorno HTC

Este entorno de trabajo es lo que se conoce hoy en día como “propiedad distributiva”, donde todos los usuarios de las máquinas de una misma organización poseen sus propios recursos.

2.2 Migración de procesos

La ejecución de tareas en máquinas que trabajan con la aplicación Condor aplica conceptos relativos a la migración de procesos. Por ello es conveniente hacer un resumido estudio de esta técnica para conocer mejor el funcionamiento de esta aplicación.

La migración de procesos consiste en transferir en tiempo de ejecución un proceso entre varias máquinas [7]. A continuación se definen algunos términos que van a ser claves a partir de ahora para la migración de procesos:

- Proceso: Abstracción del sistema operativo que representa una instancia de ejecución de un programa.
- Migrar: Migrar un proceso es transferido de una máquina hacia otra durante su ejecución.

Los objetivos de la migración de procesos están fuertemente relacionados con el tipo de aplicación que usa la migración. Los objetivos son los siguientes:

- Lograr un mayor procesamiento, utilizando la migración de procesos para distribuir la carga (scheduling distribuido, donde un nodo poco cargado anuncia que está disponible e inicia la migración de procesos de un nodo sobrecargado). La distribución de carga depende del manejo de la información de carga así como del scheduling distribuido. Con esto, se aprovecha el poder de cómputo de las estaciones de trabajo que están libres, migrando los procesos hasta que el dueño de dicha máquina regresa.
- Hacer locales los recursos, ya que es mucho más eficiente acceder a los recursos locales en vez de a los remotos.
- La tolerancia a fallos mejora debido a la migración ya que pueden existir fallos en uno de los nodos (caída de la red, de una máquina...) durante la ejecución prolongada de aplicaciones. En estas situaciones se puede utilizar migración con otras técnicas como el checkpoint (qué se verá en el siguiente capítulo), como ocurre con la aplicación Condor.
- La administración de los sistemas se simplifica si las aplicaciones que llevan un largo tiempo de cómputo se transfieren a otros nodos.

A pesar de lo interesante de estas posibilidades, y del esfuerzo invertido en investigaciones, no se ha logrado aún que la migración de procesos sea utilizada popularmente. Gracias al creciente desarrollo de los sistemas distribuidos y de los sistemas operativos distribuidos, la migración de procesos está recibiendo atención no solo en el mundo académico sino también en el comercial. La tendencia a utilizar redes con estaciones de trabajo en vez de grandes computadoras y la presencia de la "world wide web" hace ver que la migración logrará al cabo de un futuro próximo una importante difusión

Existen muchos sistemas operativos que vienen implementados con migración de procesos, además de que algunos de ellos proveen mecanismos que permiten detener la ejecución de los procesos de una máquina y continuarlos en el mismo estado en otra, como es el caso de Condor.

Aunque la migración ofrece muchas ventajas, hoy en día todavía no ha alcanzado un uso generalizado. Una de las razones que fundamentan este hecho es la complejidad de agregar migración transparente a sistemas que fueron originalmente diseñados para funcionar en "stand-alone". Otra razón tiene que ver con el aspecto

comercial de los sistemas operativos ya que no hay un buen argumento para dar pie a la migración. De cualquier manera, la migración de procesos sigue siendo una tarea para la investigación. Puede ser que esto se deba principalmente al potencial que ofrece a los diferentes tipos de aplicaciones y al interés académico debido a los problemas no resueltos de esta técnica.

2.2.1 Características

Las características que influyen directamente en la efectividad y el desarrollo de la migración de procesos se citan a continuación.

2.2.1.1 Complejidad y soporte del sistema operativo

La complejidad de la implementación y la dependencia con el sistema operativo son dos de los obstáculos que se oponen a la migración de procesos. La migración puede clasificarse según el nivel donde se implemente. Puede implementarse como parte del Kernel del sistema operativo, en espacio de usuario, como parte de un entorno de sistema, o como parte de la aplicación. Las implementaciones a distintos niveles determinan diferente complejidad, transparencia...

2.2.1.2 Rendimiento

El segundo factor que afecta a la migración de procesos es el rendimiento. El coste inicial y el coste en tiempo de ejecución provocado por la migración afecta al rendimiento. Si solo se transfiere una parte del estado hacia otro nodo, la tarea puede ejecutarse antes y por lo tanto, el costo inicial de migración es menor.

2.2.1.3 Transparencia

Para lograr la transparencia se debe evitar que tanto la tarea lograda como las demás tareas del sistema perciban la migración. La comunicación con un proceso migrado puede retrasarse durante la migración pero no puede perderse ningún mensaje. Después de la migración, el proceso debe de ser capaz de continuar comunicándose a través de los mismos canales I/O.

2.2.1.4 Tolerancia a fallos

Los fallos son importantes en la implementación de la migración de procesos pues pueden ocurrir en un nodo fuente, en un nodo destino o en medio de una comunicación. La tolerancia a fallos puede mejorarse de varias formas: se puede reducir el impacto de los fallos durante la migración si se mantiene el estado del proceso en los nodos fuente y destino hasta que la instancia del nodo destino se convierta en un proceso normal y se informe de esto al nodo fuente.

2.2.1.5 Escalabilidad

La escalabilidad es una característica de la migración de procesos que puede medirse con respecto a la cantidad de nodos del sistema, o a la cantidad de veces que puede migrar un proceso a lo largo de su tiempo de vida, o a la complejidad y tipo de los procesos (como el número de canales / archivos abiertos y tamaño en memoria).

2.2.1.6 Heterogeneidad

La heterogeneidad es la capacidad de realizar migración de procesos entre máquinas heterogéneas en una red dada. En un principio la heterogeneidad no era necesidad de la migración. El reciente crecimiento de la computación distribuida ha aumentado el interés de esta.

Para lograr la heterogeneidad el estado debe almacenarse en una representación independiente a la arquitectura. Una aplicación se compila por adelantado para cada arquitectura y se agrega un código para poder conocer que procedimientos y variables existen en cada momento y además se identifican unos puntos en los cuales la aplicación puede interrumpirse (como el caso de checkpoint en Condor).

2.2.2 Resumen e investigaciones futuras

Se cree que sí existe un futuro para la migración de procesos, además de que las diferentes corrientes de desarrollo pueden llevar a popularizarla. Algunas de las posibles tendencias futuras son:

- La posibilidad de ubicar los mecanismos de migración a nivel de usuario, tratando de lograr un rendimiento comparable al de sistemas cuyos mecanismos están implementados a nivel más bajo sin tener tantas complicaciones. El modelo de checkpoint / restart esta siendo difundido y paquetes como Condor [8], LSF [9] y LoadLever [10] están siendo utilizados en aplicaciones batch y en ambientes de producción donde la demanda por los recursos de computación es alta y es posible aprovechar el balance de carga.
- Un segundo camino involucra a los clusters de las estaciones de trabajo. Los avances recientes en las redes de alta velocidad han reducido el coste de la migración, permitiendo un desarrollo de implementaciones costosas.
- Otra alternativa está cada vez más cercana a los usuarios de hoy en día (Windows sobre plataformas Intel). El objetivo es llevar la migración a los hogares. Algunas compañías ya han puesto a disposición de los usuarios su arquitectura (Sun Jini) [11].
- Un cuarto argumento para ir a favor de la migración de procesos consisten en realizar algún tipo de esfuerzo nacional o internacional de computación.

A pesar de todas estas tendencias futuras, todavía queda mucho para que la migración de procesos sea una realidad en las estaciones de trabajo del mundo.

2.3 Condor

Una vez comentados los elementos que jugarán un papel importantísimo en Condor, tales como HTC o “Migración de Procesos”, se introduce el concepto de Condor. Condor es una aplicación software, diseñada para trabajar en redes HTC con el objetivo de poder conseguir el mayor rendimiento de las máquinas que forman parte de una red. Pero Condor es algo más que eso. A continuación se comenta la evolución de este software, su definición más detallada, conceptos de Condor…

2.3.1 Historia de Condor

El origen Condor surgió de la necesidad por parte de las distintas ramas de las ciencias de resolver diferentes problemas complejos que necesitaban para su resolución el desarrollo de miles de operaciones, es decir millones de ciclos de CPU.

De todas estas necesidades nació el proyecto Condor, que se remonta a mediados de los años ochenta, donde un grupo de investigación de la Universidad de Winsconsin-Madison (WM-Madison), dirigidos por el investigador Miron Livny y apoyándose en el antes mencionado entorno High-Throughput Computing crearon un sistema software bajo el sistema operativo Linux. Este sistema era capaz de

aprovechar el tiempo de inactividad de las máquinas, permitiendo agregar y compartir recursos, transformándose en un sistema completo de gestión de los mismos aprovechando al máximo el tiempo de ejecución de las CPUs de las máquinas.

La instalación de este software en una red de ordenadores sirvió para tener la mayor fuente de computación de dicha universidad, satisfaciendo tanto a las facultades y departamentos como a investigadores y estudiantes. Hoy en día Condor maneja más de mil estaciones de trabajo solo en el departamento de trabajo del equipo de Condor, cubriendo las necesidades de los investigadores, científicos, personal docente…

A lo largo de los años, han ido apareciendo nuevas versiones de Condor, que han ido solucionando las necesidades que han ido surgiendo en las redes de trabajo. Esto ha hecho que hoy en día Condor haya podido ser establecido en campus universitarios de todo el mundo, en cientos de organizaciones, industrias y gobiernos.

2.3.2 Definición de Condor

Antes de entrar en la definición de Condor hay que conocer el concepto de "cluster". Un cluster es un tipo de sistema de procesamiento en paralelo o distribuido, formado de un conjunto de computadoras autónomas interconectadas y que trabajan en forma cooperativa como si formaran un único recurso computacional [12].

A partir de aquí se puede dar una definición de lo que es la aplicación Condor definiéndola como un sistema software que crea un entorno High Throughput Computing (HTC) mediante la efectividad del poder de los cluster de Unix o NT en una red. Pero Condor es algo más. El sistema Condor soluciona el reparto de tareas entre máquinas del cluster y controla la ejecución de tareas que son sometidas a Condor para que sean ejecutadas en una máquina en concreto, cuando el software de Condor de esa máquina detecta que lleva un tiempo sin actividad del CPU. Para entender el concepto de la aplicación y el objetivo de la misma, es necesario explicar otros conceptos que serán fundamentales a lo largo del proyecto.

2.3.3 Condor Pool

Una primera idea de un Condor Pool es la de un número arbitrario de máquinas, con posibles arquitecturas y sistemas operativos diversos con el sistema software Condor que se conectan a una red. Para supervisar el estado de dicha red aparecen los llamados "procesos" Condor. Conceptualmente se dice que un Condor Pool es una colección de recursos y de peticiones de recursos [8].

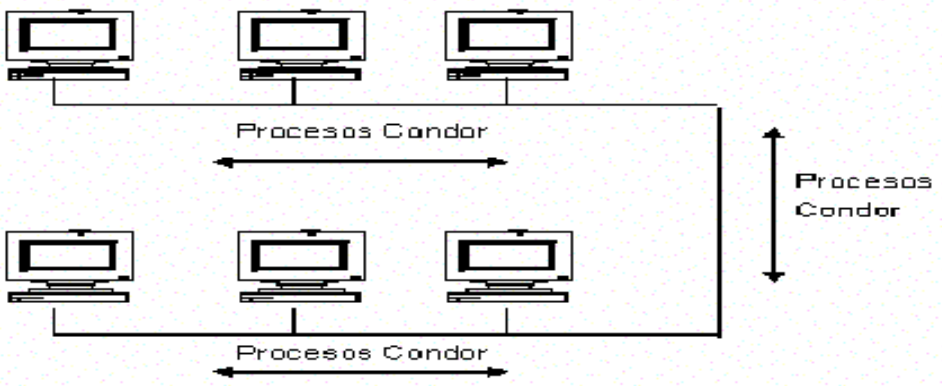

Figura 2. 2 Condor Pool general

2.3.4 Administrador de Condor

Otro elemento que se debe conocer es la figura del Administrador de Condor, que es quien se encarga de configurar el Condor Pool para conseguir mantener a los recursos y los usuarios del Condor Pool sin conflictos [8].

La siguiente figura muestra intuitivamente cómo sería un Condor Pool. Físicamente consiste en una red de ordenadores con el sistema software Condor, un administrador del sistema cuya función es la de realizar la configuración de la red y mantener un perfecto funcionamiento del Condor Pool y de los procesos Condor que se encuentran ejecutándose en todas las máquinas del Pool, interactuando entre ellos en la red.

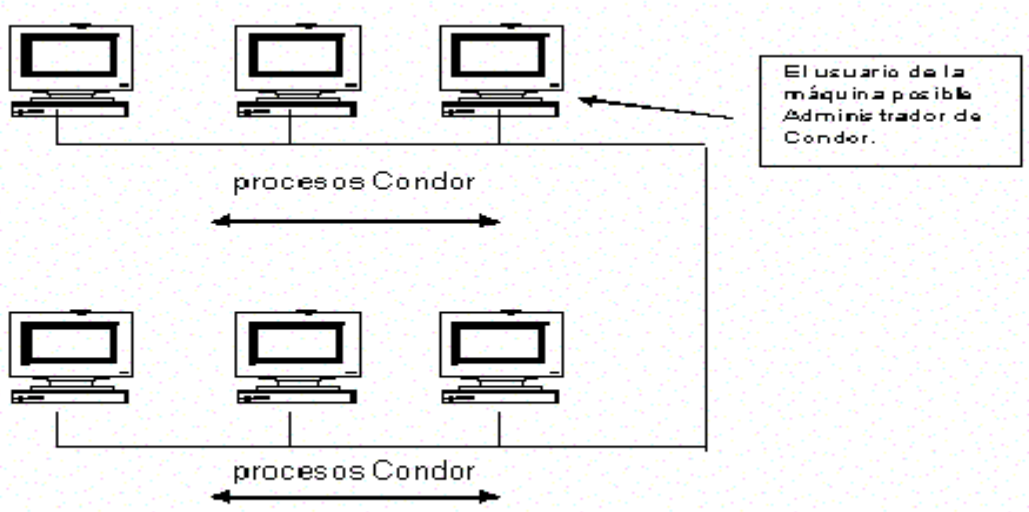

Figura 2. 3 Administrador Condor Pool

2.3.5 Funcionamiento de Condor

El funcionamiento de Condor en una red consiste en combinar los recursos de los que se dispone en la red con las peticiones que llegan de las máquinas que forman parte de dicha red.

Cada recurso tiene un propietario, el usuario que trabaja en la máquina. Esta persona tiene absoluto poder sobre sus recursos. De esta manera, Condor minimiza el impacto sobre el propietario causado por la aplicación. Se da de alta al recurso del propietario, definiendo una política para cuando se sirvan o no peticiones Condor. Cada petición que se requiere tiene un propietario que es el usuario al cual somete una tarea en el Pool. Estos usuarios utilizan Condor para sacar el máximo rendimiento a sus estaciones de trabajo para procesar trabajos. Puede ocurrir que los intereses de los recursos de los propietarios sean un conflicto con los intereses de los recursos pedidos.

El objetivo de Condor en una red es el de alcanzar el mayor rendimiento de procesamiento posible. Para ello proporciona dos funciones importantes. Primero pone a trabajar a las máquinas que poseen los recursos más eficientes. En segundo lugar amplía los recursos disponibles para un usuario, funcionando en un ambiente de "propiedad distributiva".

Condor aprovecha aquellos recursos de cómputo que se pueden desaprovechar y los usa para otras ejecuciones. La tarea principal de este sistema software es la de poner en marcha todos aquellos trabajos que se pueden someter de forma inmediata en una red, encontrando máquinas que estén disponibles para ello. De esta manera, las enormes cantidades de cómputo para ejecutar tareas se pueden hacer con la intervención mínima del usuario. Por otra parte, Condor permite que los usuarios de la red con Condor se aprovechen de las máquinas con buenos recursos, pudiendo contar con ellas para ejecutar tareas, donde, sin este sistema no sería posible, proporcionando el acceso uniforme a los recursos distribuidos que se poseen.

Condor proporciona un número de ventajas importantes a sus usuarios en cuanto a las tareas a ejecutar en el Condor Pool. El código de los programas que se ejecuten no tiene que ser modificado, aunque debe ser ligado a las bibliotecas de Condor. Estos programas son las llamadas “tareas standard" (que se verá al final de este capítulo) de Condor. Condor también proporciona un mecanismo a los binarios que no son ligados a las librerías, estos son los llamadas “tareas vainilla" (que se verá al final de este capítulo).

2.4 Roles de una máquina en el Pool

Una vez conocido el funcionamiento de Condor de forma intuitiva, hay que saber que todas esas máquinas que forman parte de un Pool pueden actuar de diferentes formas, dependiendo de cómo estén configuradas. De las diferentes formas de actuar de una máquina en un Pool aparece el concepto de Rol.

Rol se define en Condor como el funcionamiento que adoptará una máquina para formar parte de un Condor pool. Por tanto el comportamiento que se le de a una máquina en un Pool será el rol que juegue en ese Condor Pool. Las máquinas que forman parte de un Condor Pool pueden funcionar también realizando varios roles simultáneamente [13].

Los siguientes párrafos describen cuales son los roles que pueden jugar las máquinas y que recursos se requieren en una máquina para que esta pueda dar el servicio correspondiente.

2.4.1 Central Manager

Uno de los Roles más importantes que puede jugar una máquina en un Condor Pool es el de Central Manager.

La idea fundamental de la máquina que actúa como Central Manager es que esta debe ser una máquina con ciclos de CPU libres para poder gestionar los recursos del Condor Pool con el fin de poder realizar un reparto de las tareas que se someten en el Pool ejecutándolas en las máquinas destinadas a ello.

El Central Manager en un Condor Pool actúa como el colector de información de todas las máquinas que forman parte del Pool y también es el negociador entre los recursos que hay en el pool y las peticiones de recursos que se generan por parte de las máquinas que forman parte del mismo.

Las funciones antes mencionadas que realiza el Central Manager son ejecutadas por procesos Condor. Hay un proceso encargado de recolectar la información de las máquinas del Pool y un proceso encargado de negociar los recursos de las máquinas con las peticiones de recursos que haya en el Condor Pool. Estos procesos pueden ejecutarse en máquinas diferentes, con lo cual se tendría la funcionalidad del Central Manager distribuida en dos máquinas. Por un lado, el

colector de información y por otro el negociador de recursos del pool. Lo normal y más lógico es que ambos procesos convivan en la misma máquina.

La máquina que realiza el papel de Central Manager desempeña una de las funcionalidades más importantes de un Condor Pool, por tanto, dicha máquina debe ser muy fiable. Si esta máquina falla, no se podrá ejecutar ninguna máquina adicional con el sistema Condor.

Se debe elegir como Central Manager una máquina que está continuamente activa, o al menos una que pueda ser reiniciada rápidamente si algo va mal. Uno de los requisitos más importantes para una máquina que tiene la función de Central Manager es que debe tener una buena conexión en la red hacia todas las máquinas del Condor Pool, así, las máquinas que pertenecen al Pool pueden enviar actualizaciones sobre la red de trabajo al Central Manager sin problemas. Es decir, toda la información del Condor Pool residirá en el Central Manager, así como también este será el centro de peticiones de recursos que haya por parte de cualquier máquina en el Condor Pool.

Normalmente, sólo suele haber un Central Manager por cada Pool, pero se puede dar el caso de que un Central Manager forme parte de varios pools o que haya más de un Central Manager en un mismo Pool. Todo esto depende de las necesidades del Pool en que nos encontremos.

Introduciendo este concepto a un Condor Pool se puede deducir que es lógico que el Administrador de Condor sea el Central Manager, ya que este va a manejar la información del todo el Condor Pool.

La siguiente figura muestra una red configurada como Condor Pool, donde una de las máquinas se configura para actuar como Central Manager y por tanto también es como el Administrador del Pool.

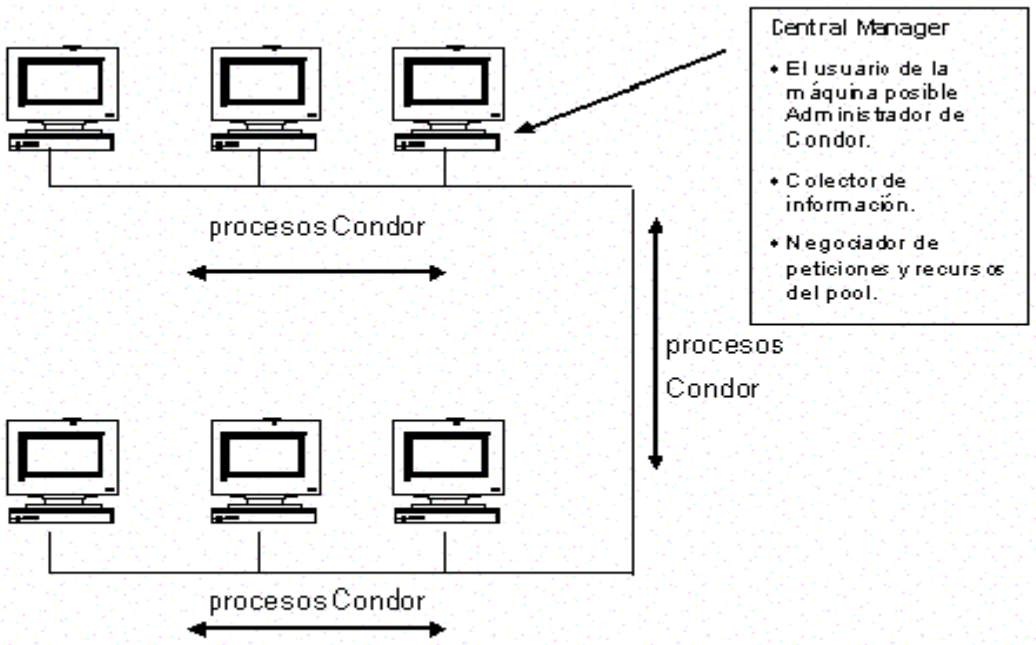

Figura 2. 4 Central Manager en un Condor Pool

2.4.2 Execute Machine

Una vez que se tiene el Central Manager en el Condor Pool se necesitan máquinas que sean capaces de ejecutar tareas de las máquinas del Pool, ya que ese es el propósito de Condor.

Las máquinas que son configuradas para tal fin juegan el rol que se conoce como Execute Machine. Obviamente, algunas de las máquinas que forman parte del Condor Pool tienen que servir esta función, porque sino el Condor Pool no tiene sentido alguno.

Las máquinas que son configuradas para realizar el papel de Execute no deben de cumplir muchos requisitos. El recurso más importante que se le puede exigir a una máquina que funciona como Execute es el espacio de disco ya que si las demás máquinas del Condor Pool quieren ejecutar tareas y esta máquina es la encargada de ejecutar dichas tareas, las máquinas verterán dichas tareas al disco local de la máquina Execute para ser ejecutadas antes de volver a reenviarlas a la máquina del propietario de la tarea. De todas formas, si no hay mucho espacio de disco, Condor limitará el tamaño del fichero donde se guarda la tarea en el Execute al tamaño de la tarea remota.

En general la mayoría de los recursos que tienen las máquinas (espacio de swap, memoria real, velocidad de la CPU...) son los requisitos que se exigen en las máquinas que forman parte del Condor Pool. Si hay peticiones de ejecución de tareas que no requieren muchos recursos para su ejecución, cualquier máquina en el Pool puede servir dichas peticiones.

La siguiente figura muestra un Condor Pool con una máquina configurada como Central Manager (explicado en el apartado anterior) y varias máquinas que son configuradas como Execute Machine.

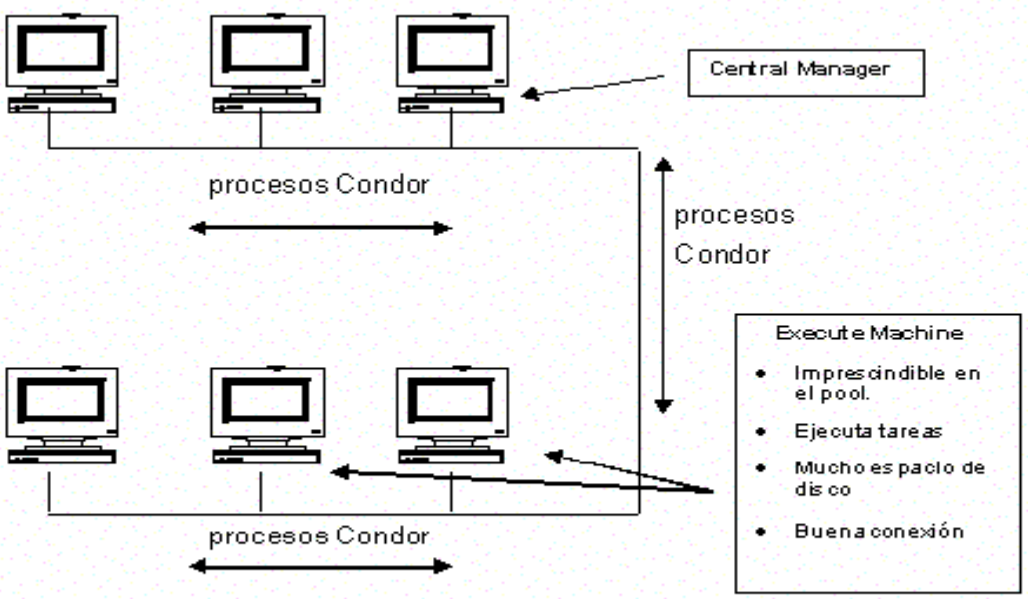

Figura 2. 5 Execute Machine en un Condor Pool

2.4.3 Submit Machine

Conocidos en el Condor Pool el Central Manager (repositorio de información y negociador de peticiones) y las máquinas Execute (ejecutan tareas de otras máquinas), es obvio que falta otro papel importante en el Pool que es el de las máquinas que quieren enviar tareas para que las ejecuten otras máquinas. Esta función es desempeñada por las máquinas que se configuran como Submit Machine. Es otro de los Roles que se pueden dar en un Condor Pool.

Los requisitos que debe de cumplir una máquina Submit son mucho mayores que los recursos que debe cumplir una máquina Execute. Esto es así porque aunque no se ejecuten tareas en la máquina Submit, sino en las máquinas que funcionan como Execute, estas generan un proceso en la máquina Submit cada vez que se somete una tarea y dcho proceso se debe de mantener. Por ello, si las máquinas Execute tienen muchos programas ejecutándose, eso quiere decir que cada una de esas tareas sometidas habrá generado un proceso en la máquina Submit. Es por ello que las máquinas que se configuran para trabajar como máquinas Submit necesitan una cantidad considerable de espacio swap y/o memoria real.

También se requiere que la máquina que actúa como Submit Machine tenga bastante espacio de disco duro. Esto es así porque todas las máquinas que ejecutan tareas generan unos ficheros llamados “ficheros checkpoint” (se verá en el capítulo 3) y estos ficheros son almacenados también en la máquina Submit. Además estos ficheros tienen una gran imagen de memoria y si se someten muchas tareas, se necesitará mucho espacio de disco para dar cabida a los ficheros. El espacio de disco requerido puede ser menor si se utiliza un “Servidor Checkpoint” (se verá en el capítulo 3).

En la siguiente figura se muestra la incorporación de una máquina Submit a un Condor Pool y como interactúan las máquinas Execute con la máquina Submit generando procesos para mantener las tareas.

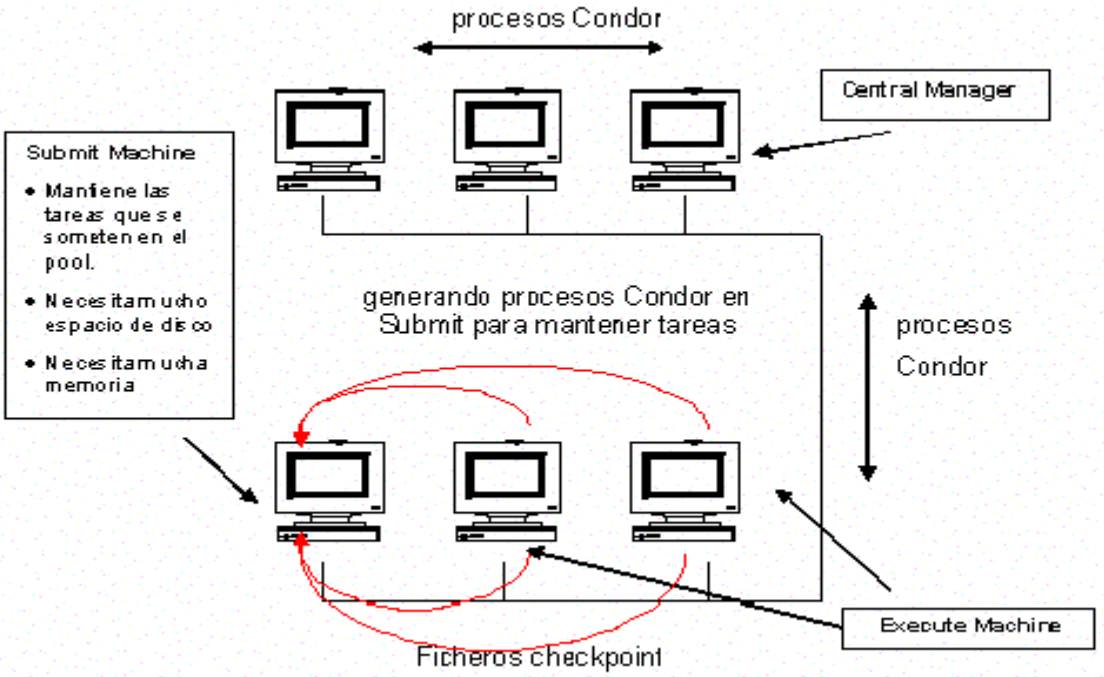

Figura 2. 6 Submit Machine en un Condor Pool

2.4.4 Servidor Checkpoint

Además de los Roles antes mencionados, es conveniente comentar de una manera resumida el Servidor Checkpoint ya que se verá más detalladamente en el siguiente capítulo.

El Servidor Checkpoint es una máquina centralizada que almacena todos los "ficheros checkpoint" (ver capítulo 3) de las tareas que se someten en el pool, solucionando así el problema que podía surgir en la máquina Submit sobre el espacio de disco duro y teniendo una única máquina con las imágenes de las tareas. La máquina que funciona como Servidor Checkpoint debe tener bastante espacio de disco duro y una buena conexión a la red de trabajo para el resto de máquinas del Condor Pool, ya que el tráfico puede ser bastante intenso debido al gran tamaño de los "ficheros checkpoint".

La funcionalidad de un Servidor Checkpoint en un Condor Pool es opcional y depende de las necesidades del Pool que se esté configurando. Además dicha configuración no aparece como parte estándar de la distribución binario de Condor, sino en un módulo aparte.

La siguiente figura muestra un Condor Pool con una máquina configurada como Servidor Checkpoint. Se observa como la carga de información al someter una tarea se distribuye entre la máquina Submit y la máquina Execute.

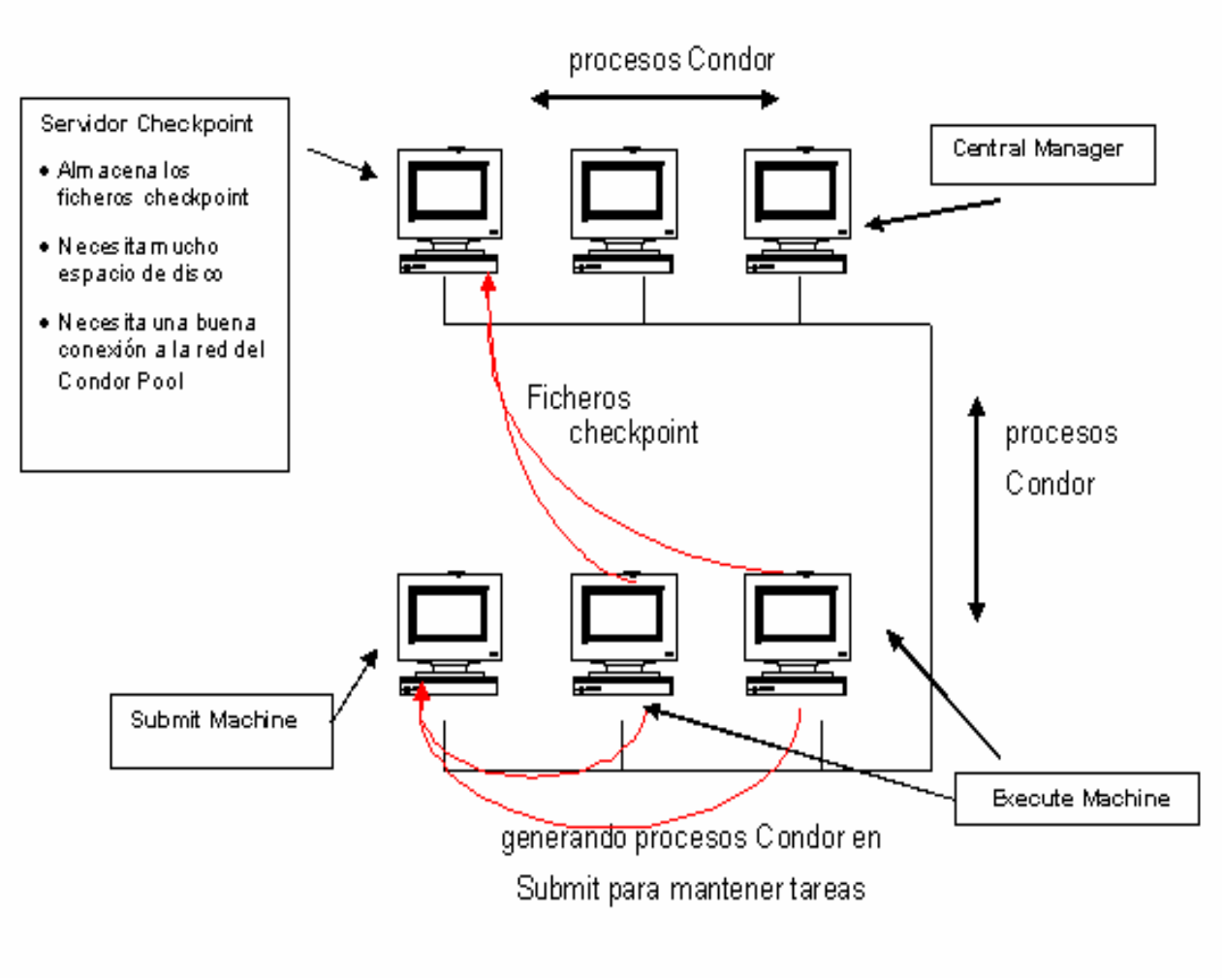

Figura 2. 7 Servidor Checkpoint en un Condor Pool

Importante:

Aunque se ha dicho que el Servidor Checkpoint puede ser configurado como una máquina independiente en un Condor Pool, lo más conveniente en caso de utilizar un sistema de ficheros distribuidos en la red, es instalar el Servidor Checkpoint en la máquina donde se encuentre el servidor del sistema de ficheros. Dicho servidor de ficheros se instalará (en general) en el Central Manager ya que este es el repositorio centralizado de la información del Condor Pool. Todo esto se verá con más detalle en el capítulo 3.

2.5 Elementos que trabajan con Condor

Antes de introducirse en las entrañas de Condor, hay varias figuras que son de obligada mención para entender el funcionamiento de Condor, ya que a partir de este momento, aparecerán en los siguientes apartados y capítulos de Condor.

2.5.1 ClassAd

Condor Pool esta formado por máquinas que poseen características distintas (recursos del Condor Pool) y tareas con características distintas que someten dichas máquinas (requisitos de las tareas). Un Condor Pool necesita por tanto, de un mecanismo de descripción de tareas y de descripción de recursos disponibles en las máquinas para hacer una asignación entre la tarea que se somete y la máquina que posee las cualidades necesarias para ejecutar dicha tarea.

Este mecanismo debe ser flexible, abstracto (no dependiente de ningún sistema operativo) y capaz de proporcionar una concordancia de los requisitos que se necesitan para someter una tarea y los recursos que se ofrecen para someter dicha tarea…Este mecanismo, que es figura indispensable en un Condor Pool es lo que se conoce como ClassAd [14].

Uno de los resultados a primera vista del uso de ClassAd en un Condor Pool es que los usuarios del Pool pueden fácilmente someter cualquier tarea, y el ClassAd se encargará de encontrar la máquina que cumpla los requisitos que se piden para someter la tarea mediante la información que posee de todas las máquinas que forman parte del Condor Pool.

2.5.2 Remote System Calls

"Remote System Calls" es un mecanismo que permite a un programa que se ejecuta en una máquina de un Condor Pool acceder a los ficheros de otra máquina distinta del mismo Pool para poder ejecutar dicho programa [14].

El uso de este mecanismo en Condor viene dado por la necesidad de que las tareas sometidas por una máquina Submit en un Condor Pool se ejecuten en máquinas remotas Execute.

Los usuarios de las máquinas que someten tareas en un Condor Pool no tienen que preocuparse sobre lo que se está haciendo en los ficheros de datos que son enviados a las estaciones de trabajo específicas para ejecutarlos. La tarea que se somete a ejecución se comporta bajo Condor como si estuviera ejecutándose en la máquina donde el usuario ha sometido la tarea, no en la máquina donde en realidad se está produciendo la ejecución.

El mecanismo "Remote System Call" resulta indispensable en ciertos entornos de ejecución de los Condor Pool para su correcto funcionamiento, como se verá en el siguiente apartado.

Para entender mejor el funcionamiento de este mecanismo se realiza una comparación de lo que ocurre con las llamadas del sistema si una tarea se ejecuta en una máquina solo y lo que ocurre cuando una máquina realiza las llamadas del sistema remoto a la hora de ejecutar una tarea.

En la siguiente figura se muestra el comportamiento interno de una máquina cuando se ejecuta un programa. El código generado por el usuario se somete a ejecución interactuando con el Kernel por medio de llamadas internas del sistema operativo con el fin de ejecutar dicha tarea [15].

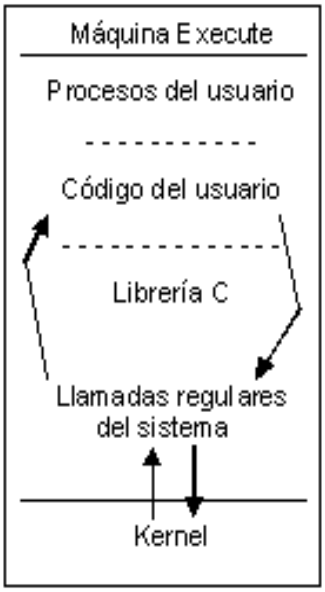

Figura 2. 8 Vista de llamadas internas de un sistema

En la siguiente figura se muestra cómo trabaja el mecanismo de llamadas al sistema remoto para ejecutar un programa y como interactúan las dos máquinas específicas de un Condor Pool.

Cuando se inicia la ejecución del código del usuario en la máquina Execute, algunas llamadas al sistema que involucran, por ejemplo el acceso a ficheros, son interceptadas por el código Condor enlazado al código del usuario de la tarea que se ejecuta (que ha debido por tanto ser recompilada mediante las librerías de Condor (ver capítulo 3, condor_compile)). Las librerías Condor convierten estas llamadas al sistema, en llamadas a procedimiento remoto (Remote System Calls) en la máquina Submit donde se ha sometido la tarea para ser ejecutada. La invocación a esa llamada al sistema se realiza por tanto en la máquina Submit, y su resultado se devuelve a la máquina Execute. Por tanto se trabaja de forma transparente, haciendo sentir que la tarea se ejecuta en la máquina de forma local.

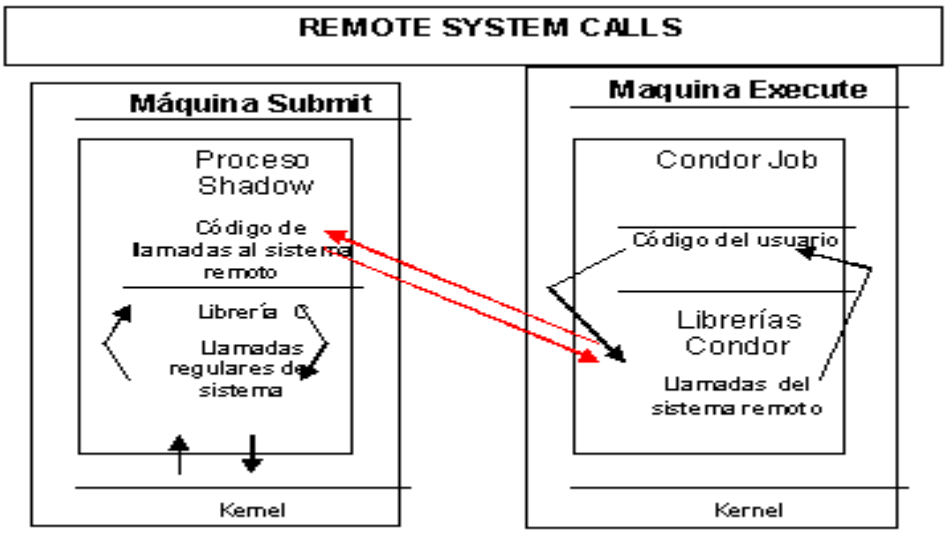

Figura 2. 9 Vista de Remote System Calls entre dos máquinas de un Condor Pool

2.6 Entornos de ejecución de Condor

Universe se define en Condor como el entorno en el cual se configura un Condor Pool para poder ejecutar tareas. Este entorno de trabajo puede variar, dependiendo de las necesidades que se tengan en cuanto a las tareas a ejecutar y de los recursos de los que se disponga en el Pool.

Como no todas las redes poseen los mismos recursos ni las mismas necesidades, el software de Condor está preparado para soportar diferentes configuraciones de entorno, tales como Vanilla Universe, Standard Universe…, donde cada una de ellas ofrece una funcionalidad diferente. Estos entornos se verán a continuación [16].

La versión de Condor utilizada para realizar el proyecto (versión 6.2.1) soporta muchos entornos diferentes con los cuales trabajar. Se llevará a la práctica el Universo Standard ya que es el que mejor se acoge a los recursos de los que se disponen.

2.6.1 Vanilla Universe

El entorno Vanilla en Condor se utiliza para someter tareas que no se re-compilan de manera satisfactoria. Los scripts shell son un caso de este tipo de tareas, donde el entorno Vanilla es el que mejor se acoge a las necesidades de estas [16].

Desafortunadamente, la ejecución de tareas bajo el entorno Vanilla no puede chequear o utilizar el mecanismo de llamadas del sistema remoto (Remote System Calls). Esto tiene consecuencias negativas para una tarea que se completa parcialmente cuando la tarea que se ejecuta en una máquina remota debe ser devuelta a la máquina que la sometió. Condor tiene dos opciones en este caso. Puede suspender la tarea y esperar para completarla más tarde o puede dejarla y relanzarla desde el principio en otra máquina en el Condor Pool.

Las tareas sometidas en un entorno de trabajo Vanilla deben también contar con otro mecanismo externo para acceder a los ficheros de datos de máquinas diferentes tales como NFS o AFS (que se verá en el siguiente capítulo).

2.6.2 Standard Universe

De este entorno se hablará en el capítulo siguiente ya que ha sido el entorno escogido para realizar el proyecto. Este entorno utiliza para su funcionamiento "Checkpointing" y "remote system calls". Estas dos figuras hacen una tarea más fiable y mantiene un acceso uniforme para obtener los recursos por parte de cualquiera en el pool [16].

2.6.3 Globus Universe

El Universo Globus en Condor no es objetivo de este proyecto, por ello se da solo una idea general del entorno.

El entorno Globus se entiende como el entorno usado para proporcionar el interfaz estándar de Condor a los usuarios que desean someter tareas del sistema Globus (tecnología necesaria para la construcción de entornos computacionales capaces de soportar aplicaciones software, integrar instrumentación…) en Condor [17].

2.6.4 Condor PVM

Este entorno no es objetivo de este proyecto. A continuación se comenta de forma resumida su funcionalidad.

El entorno PVM (Parallel Virtual Machine) es un entorno de trabajo que permite la ejecución en paralelo por medio de máquinas virtuales [18]. El funcionamiento de PVM reside en el paralelismo que este permite a la hora de programar. Ofrece una serie de mensajes pasando primitivas para el uso en lenguajes de programación C y C++, permitiendo una programación mucho más sencilla. Para la ejecución de las aplicaciones de este entorno, es necesario el uso de máquinas dedicadas. Condor soluciona este problema de recursos de PVM.

Condor puede utilizar las aplicaciones diseñadas para PVM. Condor trabaja con PVM ofreciendo una hebra de trabajo para poder ejecutar este tipo de aplicaciones en el entorno PVM. Donde PVM necesita máquinas dedicadas para ejecutar aplicaciones PVM, Condor no lo hace.

Condor puede ser usado para construir dinámicamente máquinas virtuales PVM de un Pool de máquinas Condor, así Condor soluciona el problema de las máquinas virtuales. En Condor-PVM, Condor actúa como el principal recurso para los demonios PVM [19].

Condor-PVM es un módulo opcional de Condor. No se instala automáticamente en Condor. Para obtenerlo, bastará con ir a la página principal de Condor.

2.7 Procesos Condor

Una vez explicado todos los elementos que forman parte de un Condor Pool y los entornos de ejecución que se pueden dar en un el mismo es necesario conocer todos aquellos procesos que hacen posible la comunicación entre las máquinas en el Pool y la función que desempeñan cada uno de ellos en la máquina en la que se ejecutan y en el Pool para que se puedan cumplir los objetivos impuestos en la Aplicación [20].

2.7.1 Proceso condor_master

Este proceso es el responsable de mantener al resto de los procesos (demonios) Condor que se ejecuten en la máquina donde se encuentre ejecutándose el condor_master.

En un Condor Pool el condor_master se ejecutará en todas las máquinas que formen parte del mismo, sin tener en cuenta qué rol se juegue en cada una de ellas.

condor_master es el proceso encargado de lanzar y mantener a los demás procesos que se ejecutarán en las máquinas y que se citarán a continuación. También es el encargado de chequear periódicamente la información que posee sobre los procesos que se ejecutan en la máquina para ver si hay modificaciones y en caso de caída de algún proceso, este se encarga de arrancar los procesos afectados.

Si hay procesos que por alguna razón dejan de ejecutarse y caen, el condor_master enviará un e-mail al administrador del Condor Pool y volverá a arrancar los procesos afectados.

El condor_master también soporta varios comandos administrativos que permiten comenzar, parar o reconfigurar los procesos de manera remota. Algunos de estos comandos son:

Comando	**Descripción**
condor_on	Pone en marcha los demonios Condor
condor_off	Para los demonios Condor
condor_reconfig	Reconfigura los demonios Condor
condor_restart	Reinicia el condor_master
condor_preen	Limpia los ficheros que Condor deja en la máquina y no se necesitan

Tabla 2. 1 Comandos referenciados a condor_master

Hay que destacar que el proceso condor_master es el único que necesita de la ejecución del comando “condor_preen” para limpiar la máquina de ficheros (el resto de demonios los limpian ellos mismos [20]).

Una primera visión de este proceso en un Condor Pool se puede observar en la siguiente figura. Este proceso debe de ejecutarse en todas las máquinas que formen parte del Condor Pool sin importar el rol en el que estén configuradas.

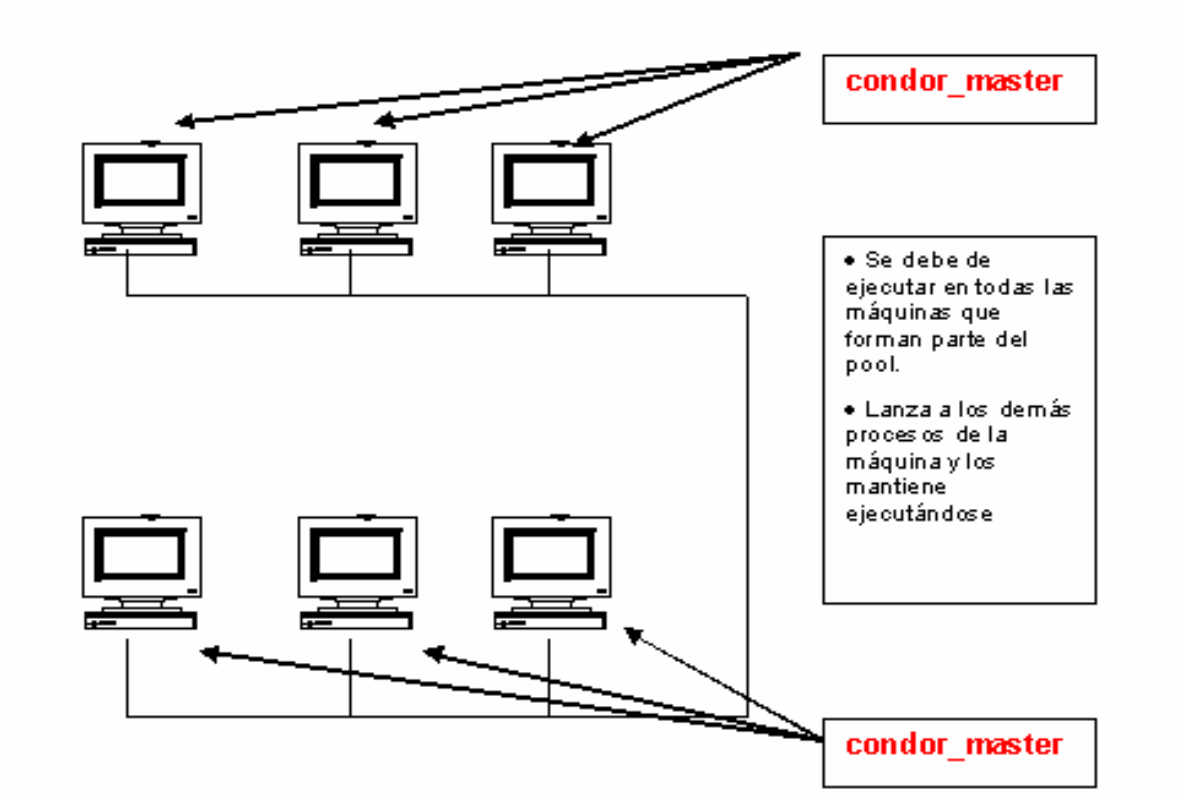

Figura 2. 10 Ejecución de condor_master en un Condor Pool

2.7.2 Proceso condor_startd

Un proceso condor_stard representa un recurso dado, es decir, este proceso se ejecuta en las máquinas capaces de ejecutar tareas en un Condor pool, por tanto se deduce que se ejecutará en las máquinas que estén configuradas como Execute Machines.

El proceso condor_startd es el responsable de reforzar la política que los propios recursos de las máquinas han configurado y los cuales determinan bajo qué condiciones las tareas remotas se ejecutan, se suspenden, reinician, desocupan, o destruyen [20].

Cuando el condor_startd está preparado para ejecutar una tarea Condor, lanza el proceso “condor_starter”, que se verá a continuación, dependiendo del tipo de tarea a ejecutar. El condor_startd está provisto también de otros comandos administrativos como el comando “condor_vacate”, cuya función consiste en desocupar tareas que están ejecutándose en una máquina determinada.

La siguiente figura muestra en un Condor Pool con todos los roles que pueden ser configurados en el mismo, donde se ejecutarían los procesos condor_startd. Como en todas las máquinas del Condor Pool se ejecuta el comando condor_master, no se indica.

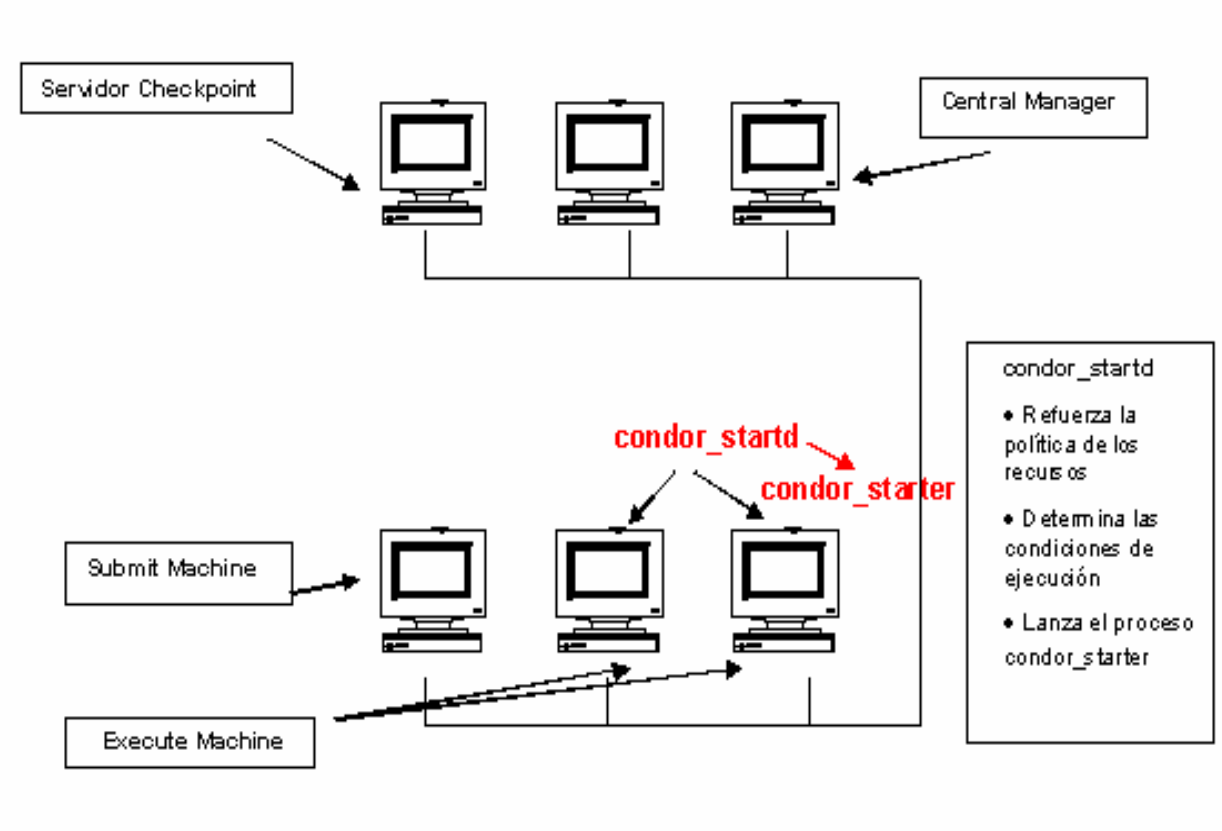

Figura 2. 11 Ejecución de condor_startd en un Condor Pool

2.7.3 Proceso condor_starter

Este proceso es un programa cuya figura no es más que la entidad que lanza una tarea condor remota que se encuentra en una máquina Execute para ejecutar.

El proceso condor_starter es lanzado por el proceso condor_startd, por tanto se deduce de esto que este proceso se ejecuta en las máquinas Execute. Este proceso es el responsable de establecer el entorno de ejecución y monitorear la tarea una vez que esta ejecutándose en la máquina Execute [20]. Cuando termina la ejecución de la tarea en la máquina, el proceso condor_starter lo notifica enviando la información del estado de la misma a la máquina que ha sometido la tarea. Una vez hecho esto, el proceso deja de ejecutarse. Por tanto este proceso solo se ejecuta cuando se está ejecutando una tarea en una máquina Submit.

2.7.4 Proceso condor_schedd

El proceso condor_schedd es el encargado de representar en las máquinas en las que se somete una tarea, la petición de ejecución de dicha tarea. Es decir, cuando en una máquina Submit se somete una tarea, es este proceso el encargado de realizar las negociaciones necesarias para poder ejecutar la tarea en una máquina Execute. Así, este proceso representa las peticiones de los recursos de un Condor Pool por parte de una máquina [20]. El proceso condor_schedd deberá ejecutarse siempre en las máquinas que deseen someter tareas, es decir, en las máquinas Submit, para poder realizar dicha negociación.

Este proceso también es el encargado introducir las tareas que se someten en una cola de tareas. En cada máquina Submit habrá una cola de tareas, que indicará las tareas que hay sometidas en dicha máquina. Por tanto, este proceso

controlará la cola de tareas. Hay varias herramientas para visualizar y controlar la cola de tareas que deben conectar con el proceso condor_schedd para hacer su trabajo. Los comandos que realizan estas tareas son:

Comando	Descripción
condor_submit	Encola tareas para su ejecución en maquinas remotas
condor_rm	Borra tareas de la cola de Condor
condor_q	Muestra información sobre las tareas en la cola

Tabla 2. 2 Comandos referenciados a condor_schedd

Algunas de las funcionalidades del proceso condor_schedd son por ejemplo anunciar el número de tareas en espera en la cola de tareas, reclamar los recursos disponibles para servir esas peticiones… Una vez que el condor_schedd ha estado haciendo juego con los recursos dados, lanzará al "condor_shadow" para servir la petición en particular. Este proceso se verá a continuación.

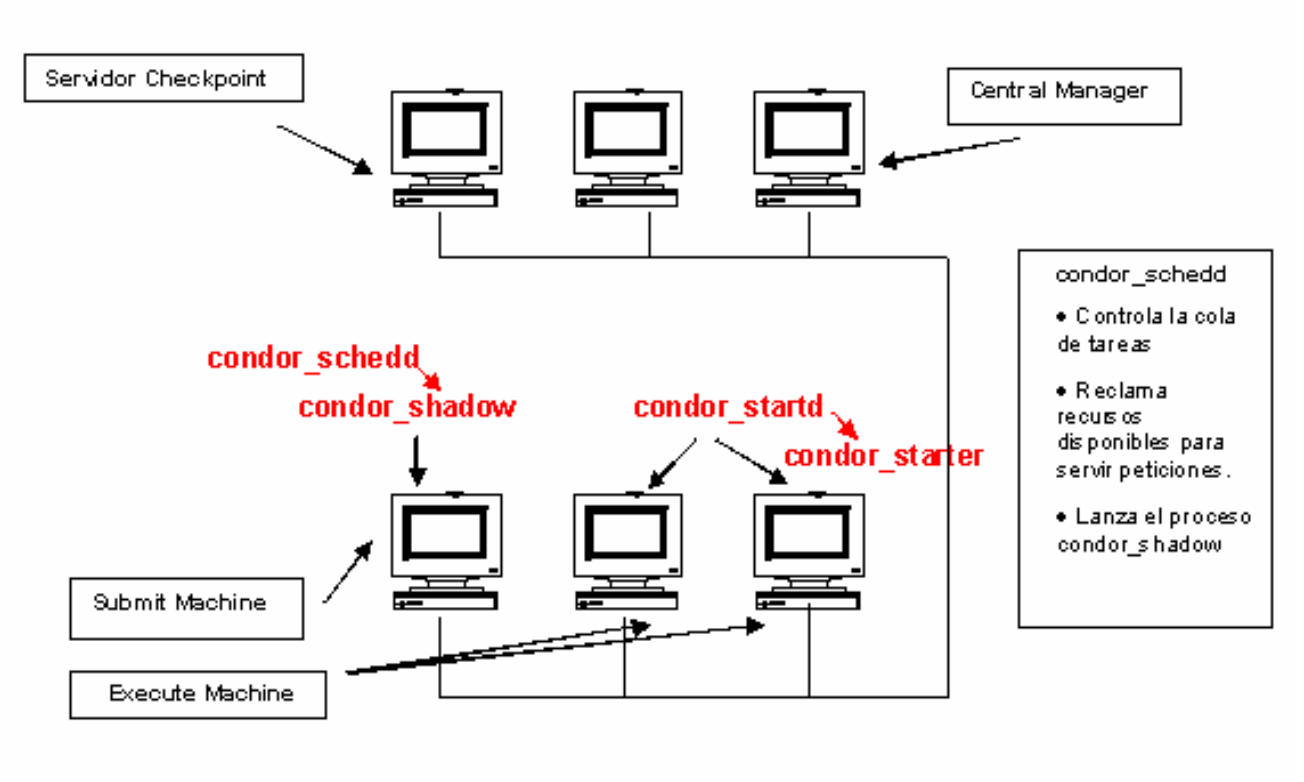

Figura 2. 12Ejecución de condor_schedd en un Condor Pool

2.7.5 Proceso condor_shadow

Este proceso es un programa que se ejecuta en la máquina donde ha sido sometida una petición, es decir en la máquina que ha sido configurada como Submit Machine y actúa como el recurso manager de la petición [20].

Cuando se conecta con una máquina Execute para ejecutar una tarea, esa llamada al sistema de la máquina Execute se envía sobre la red, bajo el "condor_shadow" que es quien en realidad ejecuta la llamada del sistema y el resultado se envía sobre la red de trabajo a la tarea remota.

El proceso condor_shadow también es responsable de tomar decisiones sobre las peticiones (tales como donde deberán ser almacenados los ficheros checkpoint, como será el acceso de ciertos ficheros...)

La siguiente figura muestra como sería esa comunicación entre los procesos que ya se conocen si se quiere someter una tarea. El proceso condor_submit, se verá en el capítulo 3.

Como se observa, cuando se quiere someter una tarea, se llama al proceso condor_schedd, este lanza el proceso condor_shadow que es quien en verdad realiza la comunicación con la máquina que va a ejecutar la tarea, es decir la máquina Execute a través del proceso condor_startd. Este lanza el proceso condor_starter que establece el entorno de ejecución y los recursos necesarios para ejecutar la tarea.

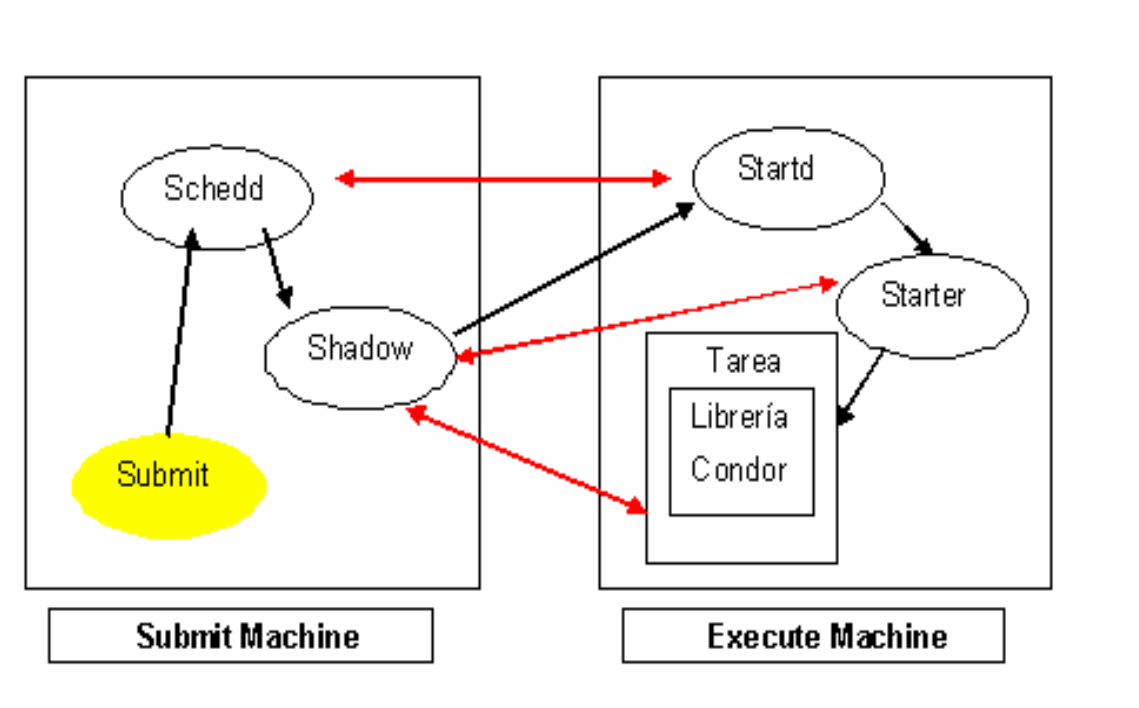

Figura 2. 13 LLamadas de los procesos en las máquinas al someter tareas (I)

Hay que tener en cuenta, que todos estos procesos, llamadas de unas máquinas a otras, etc... deben estar controlados por el Central Manager, ya que es el colector de información del Condor Pool y el negociador entre los recursos y los requisitos del Condor Pool. Así que a continuación se verá como influye el Central Manager a través de los procesos que se ejecutan en el.

2.7.6 Proceso condor_collector

Este demonio es responsable de recolectar toda la información sobre el estado del Condor pool, por tanto, este proceso se ejecutará en el Central Manager. Todos los procesos de las máquinas que forman parte del Pool enviarán actualizaciones (ClassAd) periódicamente al condor_collector. Estas ClassAds contienen información sobre el estado de los demonios, los recursos que ellos representan o las peticiones de recursos en el pool (tales como las tareas que han sido sometidas por un schedd dado).

Algunos de los comandos que pueden utilizar las máquinas que forman parte del pool es el comando “condor_status” y cuya función es la de preguntar al condor_collector por información especifica sobre el estado del Condor Pool (que

máquinas forman parte del Pool, las direcciones de dichas máquinas, en que estado se encuentran…) [20].

Antes de mostrar en una figura cómo actúa este proceso en un Condor Pool es conveniente explicar el segundo proceso que se ejecuta en el Central Manager, que es el condor_negotiator.

2.7.7 Proceso condor_negotiator

Este demonio es responsable de todas las negociaciones con el sistema Condor. Periódicamente, el condor_negociator comienza un ciclo de negociación, donde pregunta al proceso condor_collector por el estado general de todos los recursos del Pool. También es el responsable de contactar con cada schedd que tiene una petición esperando en orden de prioridad, intentando establecer prioridades a los recursos con esas peticiones [20].

El condor_negotiator es responsable de hacer cumplir al usuario las prioridades del sistema. Si un usuario con una buena prioridad tiene tareas que están esperando para ejecutarse, y los recursos son reclamados por otro usuario con una mala prioridad, el condor_negotiator puede adelantarse al recurso estableciendo con el usuario con la mejor prioridad.

Nota: Un valor numérico alto de prioridad de usuario en Condor significa una mala prioridad para el usuario. La mejor prioridad que se puede tener es de 0.5, que es el valor numérico más bajo.

A continuación se muestra una figura donde se pueden ver la comunicación entre los procesos de una máquina Execute, una máquina Submit y un Central Manager.

Los pasos que se siguen en la figura son los siguientes:

- “condor_submit” (que se verá en el capítulo 3) contacta con “condor_schedd” y añade la tarea a la cola de trabajo.
- “condor_schedd” envía ClassAd al “condor_collector” solicitando una máquina.
- “condor_negotiator” empareja la solicitud con una máquina disponible.
- “condor_schedd” solicita la máquina y lanza “condor_shadow”.
- “condor_shadow” contacta con condor_startd y solicita el apropiado “condor_starter”.
- “condor_starter” normalmente lanza la aplicación y la conecta con el “condor_shadow”.
- “condor_startd” monitorea la máquina y espera los comandos.

Puede que la aplicación finalice, o que “condor_startd” la fuerce a suspenderla o dejarla en estado vacante.

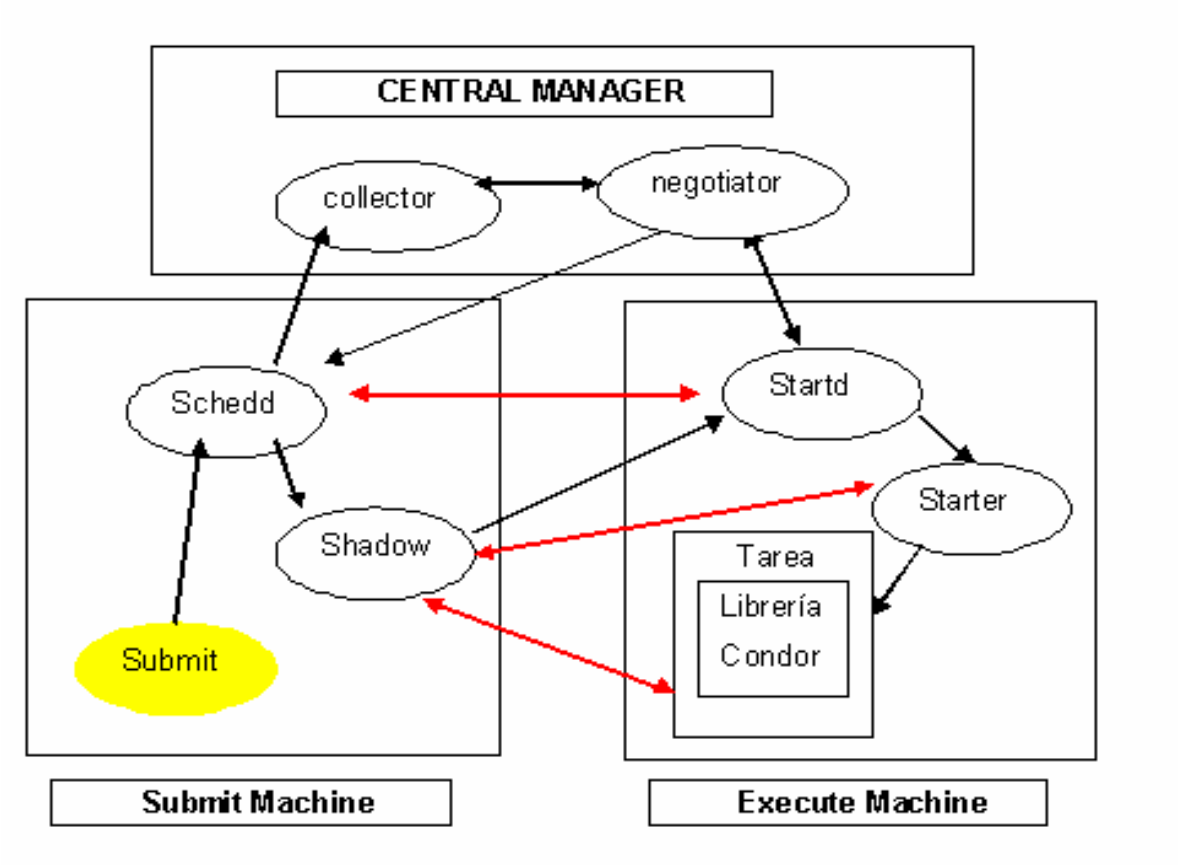

Figura 2. 14 LLamadas de los procesos en las máquinas al someter tareas (II)

2.7.8 Proceso condor_kbdd

Este proceso necesario solamente en Sistemas Digital Unix e IRIX. En estas plataformas, el "condor_startd" no puede determinar la actividad de la consola (teclado o ratón) directamente desde el sistema, así que el "condor_kbdd" conecta al X Server y chequea periódicamente para ver si ha habido alguna actividad. Si ha habido, el kbdd envía un comando al startd [20].

Esta manera de actuar por parte del proceso condor_kbdd hace que el startd sepa que la máquina del propietario esta usando la máquina otra vez y puede ejecutar cualquiera de las acciones necesarias, dado que la política ha sido configurada para hacerse cumplir.

2.7.9 Proceso condor_ckpt_server

Este proceso es el proceso característico del Servidor Checkpoint. La funcionalidad de este es el de servir peticiones para almacenar y recuperar los ficheros checkpoint. Si el Condor Pool está configurado para usar un Servidor Checkpoint pero la máquina (o el servidor mismo) cae, Condor volverá a enviar los ficheros checkpoint a la máquina Submit, actuando como cuando se configura un Condor Pool sin Servidor Checkpoint.

La siguiente figura muestra la comunicación de procesos en un Condor Pool en el caso de configurar el Central Manager para trabajar como un Servidor Checkpoint, con lo cual, los ficheros checkpoint se almacenarán en el Central Manager.

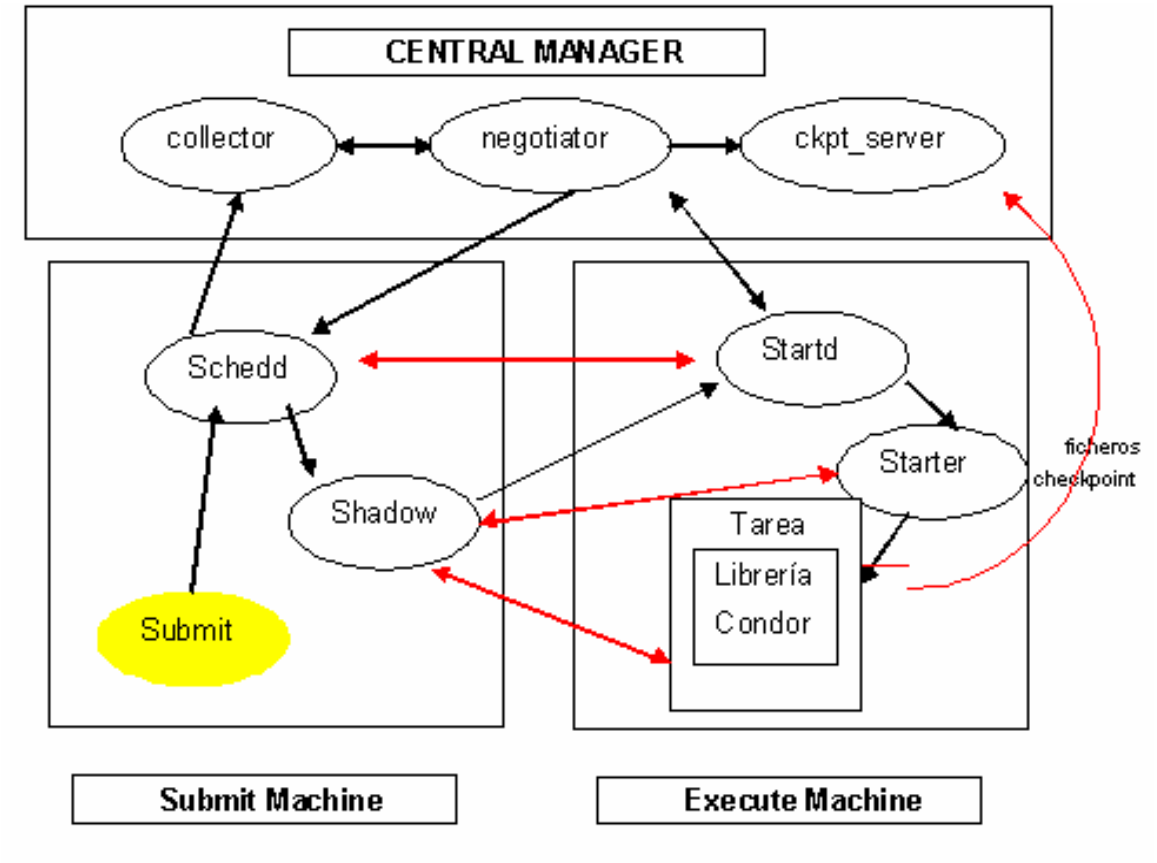

Figura 2. 15 LLamadas de los procesos con Servidor Checkpoint

En caso de que dicho servidor fallara o cayera, los ficheros checkpoint se almacenarán en la máquina Submit, disminuyendo de esta manera los recursos de esta máquina en cuanto a espacio de disco, memoria…

De todas formas, el funcionamiento de un Servidor Checkpoint se verá más detalladamente en el capítulo 3.

2.7.10 Comportamiento de los procesos en un Condor Pool

En este apartado se van a visualizar las comunicaciones entre los distintos elementos que forman parte de un Condor Pool.

En la siguiente figura se representa un Condor Pool general, donde se observan los distintos procesos que se ejecutan en las máquinas dependiendo del rol que se juegue en cada una de ellas y la comunicación entre los procesos en la máquina y entre las máquinas que forman parte del pool. En este pool no se están ejecutando tareas.

Se observan que algunas de las máquinas han sido configuradas para jugar el papel de Execute Machine, otras como Submit Machine y también hay unas máquinas llamadas Nodos regulares.

Un nodo regular es una máquina que juega dos roles, pudiendo tanto someter tareas como ejecutarlas. No es un rol especificado ya que es la combinación de Submit y Execute. Esta configuración es la más común en las estaciones de trabajo que forman parte de un pool ya que normalmente cuando una máquina forma parte de

un Pool esta dispuesta tanto a someter tareas cuando se ve en la necesidad como ejecutar tareas cuando la máquina esta infrautilizada.

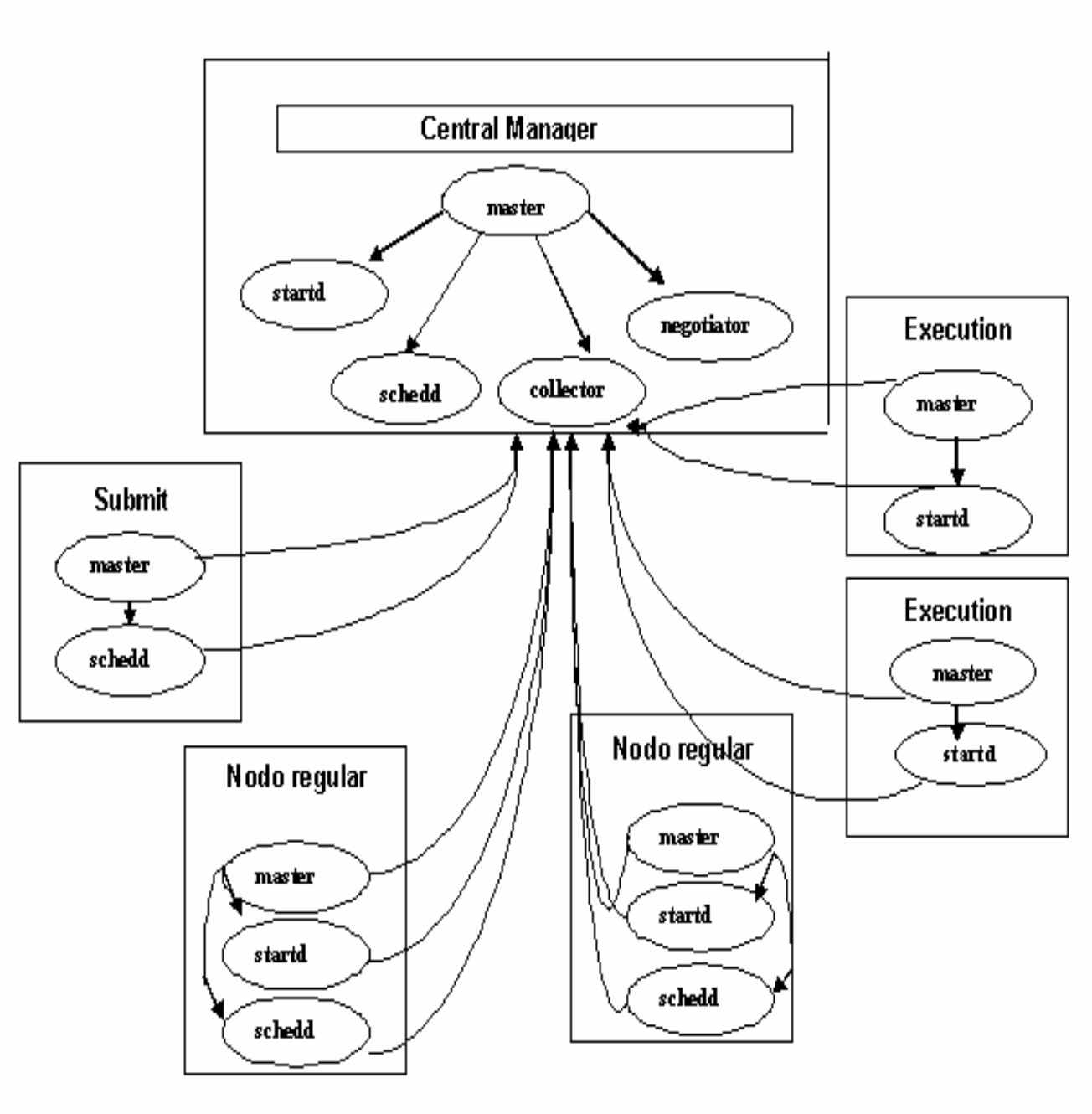

Figura 2. 16 Vista de Condor Pool

En el siguiente dibujo se muestra la arquitectura de un Condor Pool donde se somete una tarea en la máquina 2 y es ejecutada en la máquina N. De esta forma se puede observar la intercomunicación de los procesos para poder ejecutar dicha tarea.

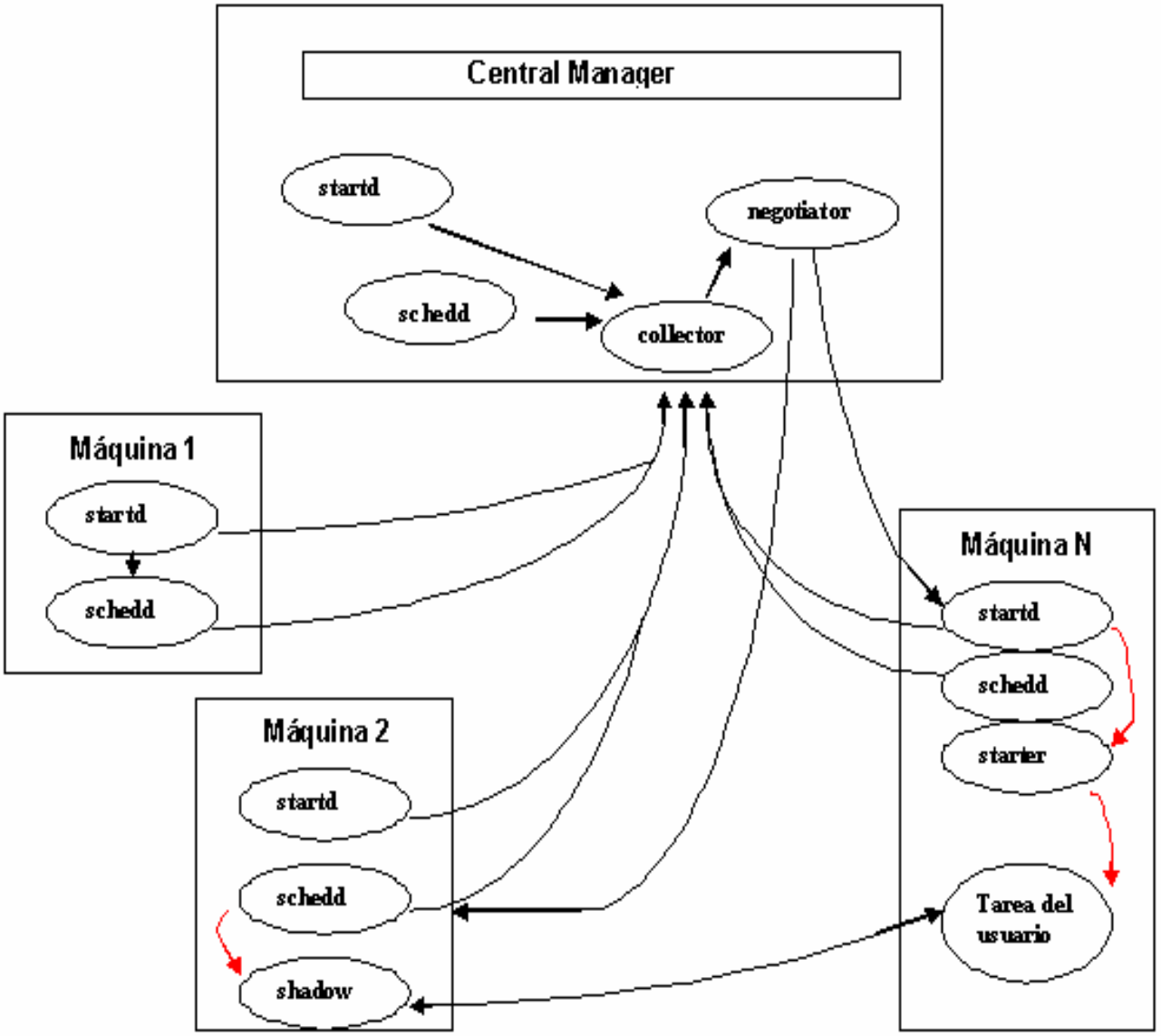

Figura 2. 17 Vista de Condor Pool sometiendo tareas

2.8 Estados y actividades de las máquinas

Además de los tipos de Roles que puede desempeñar una máquina y de los procesos que se ejecutan en ellas, hay otra característica que hay que destacar en las máquinas que forman parte de un Condor Pool, éste es el estado en el que se encuentra la máquina y la actividad que está realizando dependiendo del estado en el que se encuentre [21].

Condor asigna a cada máquina que forma parte del pool un estado. Este estado depende de si la máquina esta ejecutando o no tareas Condor. Si es así, el estado dependerá del punto de negociación que haya sido alcanzado. Además, en algunos de estos estados se definen algunas actividades. Así pues, un par estado/actividad describe una máquina. Los posibles estados y actividades son los siguientes.

2.8.1 Owner

Este estado indica que la máquina esta siendo usada por el usuario de la estación de trabajo en ese momento y/o no está disponible para ejecutar tareas Condor. Cuando arrancamos la máquina, esta siempre se encuentra en este estado. La única actividad que permite este estado es la actividad Idle.

Idle Condor establece que la máquina es Idle, si no está realizando ninguna comunicación con Condor para someter tareas, es decir, el usuario se encuentra en la máquina realizando trabajos independientes de Condor.

2.8.2 Unclaimed

En este estado la máquina está disponible para ejecutar tareas Condor, pero no lo esta haciendo actualmente. Las actividades que se ejecutan en este estado son:

Idle Este es la actividad normal de las máquinas Unclaimed. Una máquina con este par estado/actividad indica que la máquina se encuentra activa, permitiendo el ejecutar tareas Condor en ellas, pero Condor no está usando la máquina para ejecutarlas.

Benchmarking La máquina está ejecutando benchmarks para determinar la rapidez de la máquina [21]. Esta actividad solo sucede en el estado Unclaimed. El cálculo de cada cuanto tiempo ocurre una actividad viene determinado por la expresión “RunBenchmarks” (ver apéndice).

2.8.3 Matched

La máquina en este estado está disponible para ejecutar tareas, y ha sido correspondida con el proceso schedd específico por medio del proceso negotiator. La única actividad en la que puede encontrarse es Idle.

Idle En esta actividad, la máquina se encuentra activa para ejecutar tareas pero todavía no está siendo utilizada para ello.

2.8.4 Claimed

Cuando la máquina se encuentra en este estado es porque ha sido seleccionada por un schedd. Pueden darse tres actividades diferentes:

Idle En esta actividad, la máquina ha sido seleccionada para ejecutar una tarea, pero el schedd que lo seleccionó todavía tiene que activar la selección por medio de una petición a “condor_starter” para ser lanzado y servir la tarea.

Busy Cuando el schedd consigue contactar con el “condor_starter” para lanzar la tarea, la máquina pasa a la actividad Busy, indicando así que está ocupada ejecutando una tarea.

Suspended Si la tarea es suspendida por Condor, la máquina pasa a la actividad Suspended. La correspondencia entre el schedd y la máquina no se ha roto (el claim es todavía válido), pero la tarea no esta haciendo ninguna progresión y Condor no está generando ningún resultado de la tarea.

2.8.5 Preempting

El estado de Prempting se usa para terminar con una tarea condor desde una máquina dada porque el propietario de la estación de trabajo ha vuelto a su puesto y se interrumpe la tarea. Cuando una máquina entra en el estado Prempting, chequea la expresión WANT_VACATE (ver apéndice) para determinar la actividad.

Vacating En la actividad Vacating, la tarea que fue ejecutada está en el proceso checkpointing. Tan pronto como el proceso de chequeo se complete, la máquina se mueve hacia el estado Owner o el estado Claimed.

Killing Este estado significa que la máquina ha preguntado por la tarea ejecutada para finalizar la máquina inmediatamente sin hacer checkpointing.

2.8.6 Gráfico Estado-Actividad

A continuación se representan de manera gráfica todos los estados y actividades de cada uno de esos estados en los que puede encontrarse una máquina en un Condor Pool. Cada una de las flechas indica el orden en el que se puede pasar de un estado a otro en un pool.

1- En un principio, cuando se inicia Condor en una máquina, esta comienza con el estado Owner, y por tanto con la actividad Idle, que se la única que tiene este estado.

2- Dependiendo de cómo este configurado Condor, pasado cierto tiempo de inactividad por parte de la máquina, esta pasará al estado Unclaimed, avisando al Central Manager de que la máquina esta disponible para poder ejecutar tareas, en caso de ser una máquina Execute. Normalmente el estado Unlcaimed vendrá acompañado de la actividad Idle.

3- Si la máquina vuelve a ser utilizada por el usuario de la misma, entonces, el estado de la máquina que estaba en Unclaimed pasará a Owner. En caso de que la máquina siga disponible para poder ejecutar tareas y el proceso schedd contacte con el proceso negotiator del Central Manager la máquina pasará al estado Matched, donde en este estado, solo se podrá estar en la actividad Idle.

4- Si la máquina se encuentra en el estado Matched, pueden ocurrir dos cosas:

   - Que la máquina vuelva directamente al estado Owner debido a que el usuario de la máquina se encuentra utilizándola, con lo cual no se pueden ejecutar tareas.
   - Que se confirme que la máquina va a ejecutar una tarea, con lo cual pasaría al estado Claimed, pudiéndose encontrar en la actividad Idle, donde el schedd no ha recibido la petición del condor_starter para servir una tarea, en la actividad Busy donde el condor_starter ha sido iniciado o en la actividad suspended, donde ha ocurrido la suspensión de la tarea por parte de Condor.

5- Del estado Claimed, el único estado al que se puede pasar es el estado Preempting. En este estado la tarea finalizará. En este estado la máquina puede encontrarse en la actividad Vacating, que indica que la tarea se encuentra en proceso de checkpoint (ver capítulo 3) o en la actividad killing donde se finaliza con la tarea sin realizar checkpoint.

6- En el estado Preempting se pueden ocurrir dos cosas:

   - Que la máquina vuelva al estado Claimed, para terminar la ejecución de alguna tarea que no ha finalizado del todo.
   - Que la máquina pase al estado Owner donde se terminaría con el ciclo de ejecución de una tarea en una máquina.

En la figura, se muestran un conjunto de flechas rojas. Estas indican cual sería el ciclo estados por los que pasaría una máquina en la ejecución de una tarea, sin que hubieran en este ciclo interrupciones por parte de usuarios ni fallos en ejecución, etc…

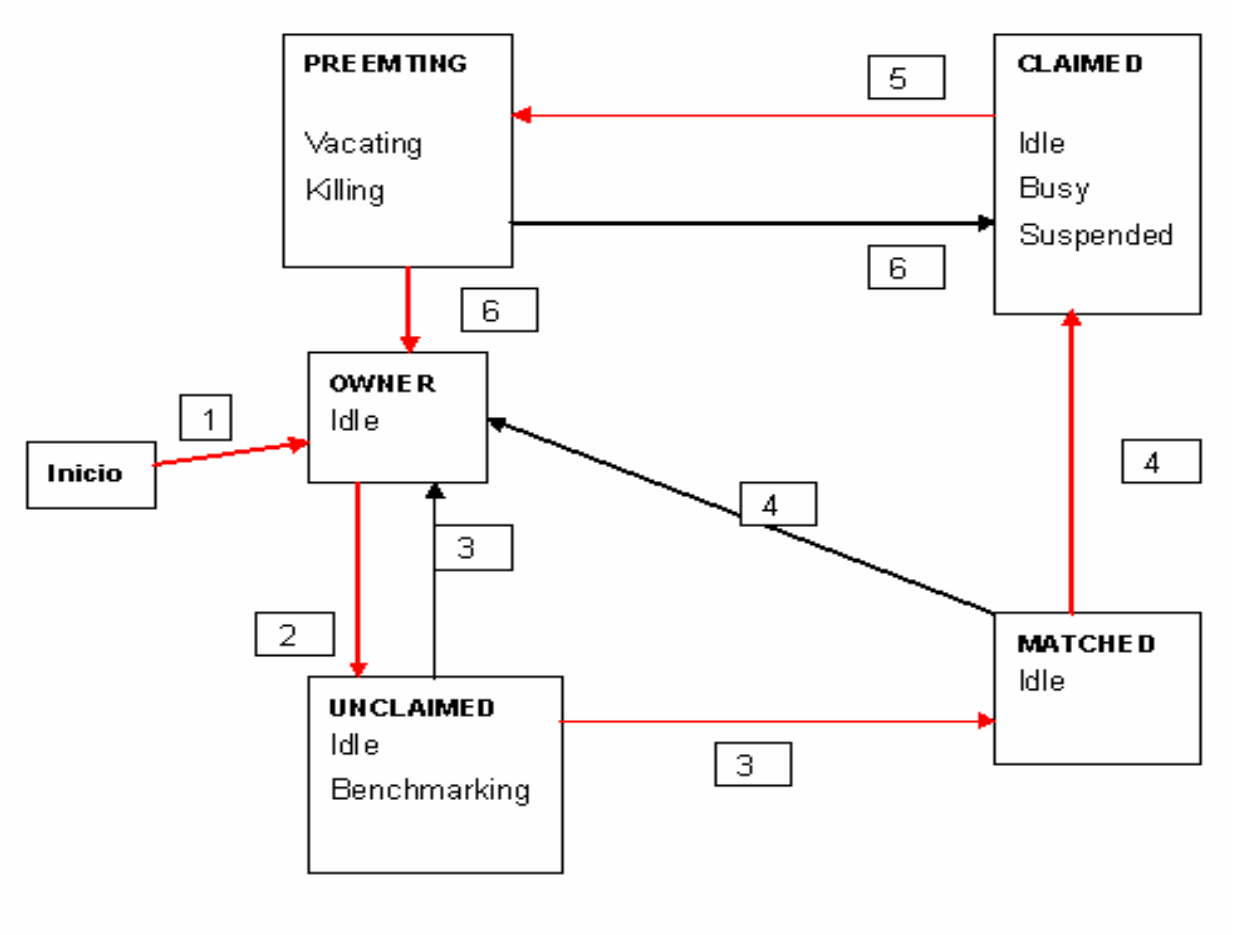

Figura 2. 18 Gráfico estado-actividad de una máquina

2.9 Ejecución de tareas

Para ejecutar tareas mediante Condor, cualquiera que sea el universo en el que se trabaje, se necesita un programa que sea capaz de someter dichas tareas. El programa que somete tareas en Condor es el programa “condor_submit” (se verá con detalle en el capítulo 3).

Este programa utiliza para su ejecución un fichero de descripción submit. Este fichero contiene desde comandos para dirigir la tarea a la cola de tareas hasta otros parámetros para especificar donde se escribirá la salida de la tarea. Un fichero de descripción puede contener por tanto, diferentes especificaciones tanto para la entrada como para la salida de las tareas (ver apéndice, fichero de descripción submit).

En cuanto a las tareas a someter, se puede dar el caso de encolar varias tareas en una única invocación del programa condor_submit, encolando estas tareas en un mismo cluster. Esto ocurre cuando se ejecuta condor_submit y el parámetro de este es un fichero de descripción submit que se encuentra definido de forma que al someter el fichero, se ejecutan varias tareas del mismo ejecutable en una sola llamada del fichero de descripción submit, (por ejemplo que se ejecute un programa varias veces, con

distintos valores de entrada para cada ejecución) (ver apéndice, Fichero de descripción submit).

Es ventajoso el someter múltiples tareas como un único cluster (en el mismo ejecutable) ya que:

- Solo se necesita una copia del fichero checkpoint para representar todas las tareas en un cluster hasta que comience la ejecución de las mismas.
- Es mucho menos sobrecargado el concebir Condor para comenzar la siguiente tarea en un cluster, que para Condor comenzar en un nuevo cluster. Esto puede provocar una gran diferencia si se están sometiendo muchas tareas y de poco tamaño.
- Si se usa checkpointing y "remote system calls" (capítulo 3), el usuario que someta la tarea en la aplicación Condor necesita recompilar sus binarios. Esto ocurre en el universo Standard.
- Los programas que no son recompilados por Condor, se ejecutan en el llamado universo "Vanilla" de Condor, pero la desventaja principal de este es que no puede chequear ni migrar. También se pueden ejecutar tareas como máquinas independientes sin sistema de ficheros distribuido, como "Personal Condor".

2.9.1 condor_submit

Para todos los universos, a la hora de someter una tarea, es imprescindible el realizar la llamada de dicha tarea mediante el comando condor_submit [22]. Dicho comando tendrá como argumento el fichero de descripción submit que describe a la tarea. Dependiendo del universo en el cual se esté trabajando, así se deberá de realizar la construcción del fichero de descripción submit (ver apéndice).

Por tanto, como se podía observar en las figuras de ejemplo para entender la comunicación entre procesos (apartado 2.7), aparece el condor_submit, como el proceso que se utiliza para someter una tarea en un Pool.

El proceso condor_submit es el encargado de contactar con el proceso condor_schedd y encuentre un medio en el Pool para poder ejecutar la tarea que se ha sometido mediante este proceso, introduciendole como argumento el fichero de descripción submit que lleva el programa que se desea ejecutar.

En la siguiente figura se muestra la comunicación entre procesos, indicando en el proceso condor_submit el comando que se debe de introducir en la línea de comandos para poder someter una tarea en una máquina. "ficherosubmit" indica el nombre del fichero que contendrá el nombre del ejecutable a someter. Todo esto se verá con más detalle en el siguiente capítulo al estudiar como se someten tareas en el Universo Standard.

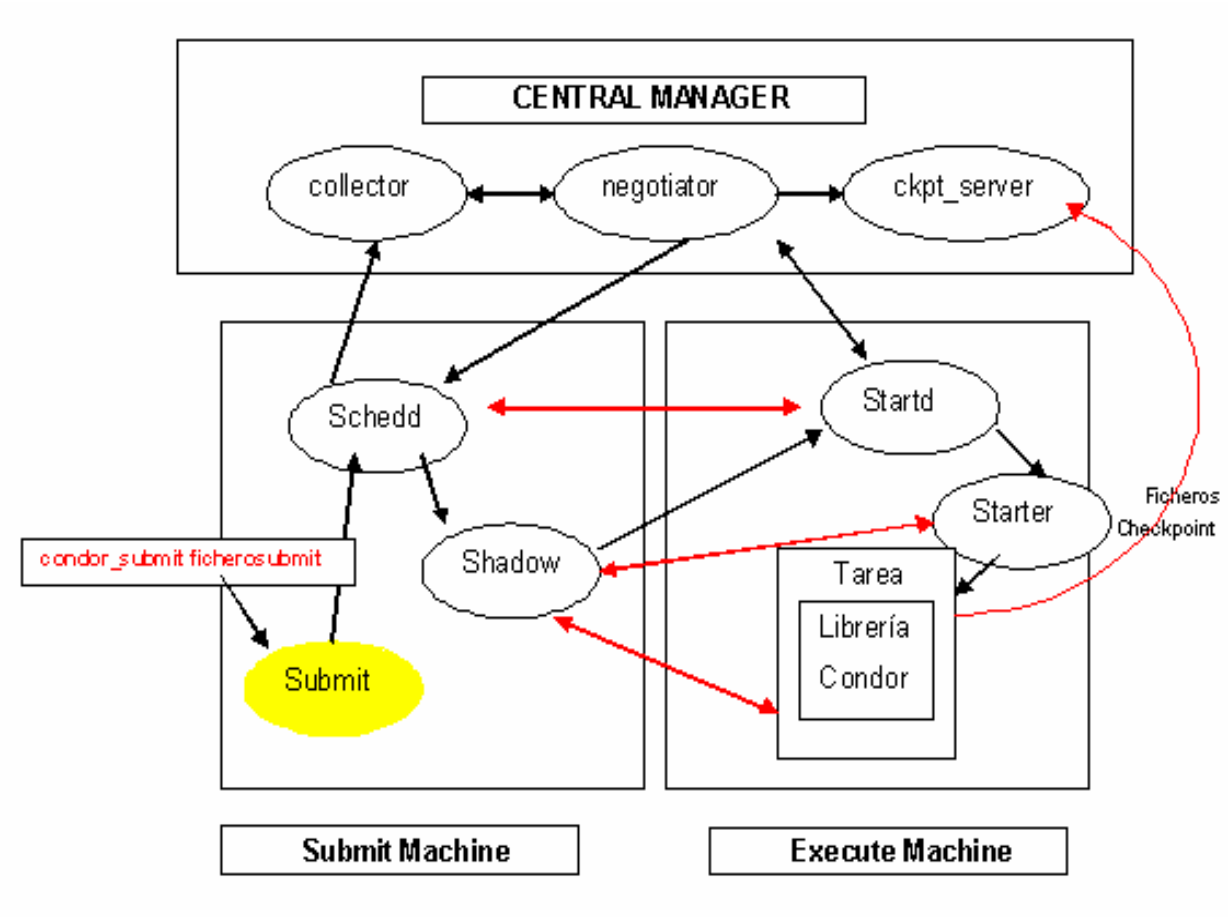

Figura 2. 19 LLamadas de los procesos al someter tareas

2.10 Condor. La solución

En los apartados anteriores se han estudiado todas las aportaciones que puede ofrecer la aplicación Condor a una red de trabajo, analizando los roles que se pueden jugar, los diferentes entornos que se pueden realizar, los procesos que se ejecutan en las máquinas, los estados y actividades, las diferentes formas de ejecutar tareas…

Con este estudio previo se demuestra que con una aplicación software como Condor se puede solucionar el reparto de tareas entre máquinas de una red, controlando así la ejecución de dichas tareas sometidas a Condor y aprovechando el tiempo de inactividad de los CPU's. Todas estas razones hacen de Condor una aplicación muy interesante para su estudio y puesta en marcha con vistas a los objetivos mencionados en el capítulo 1.

A partir de este momento se estudiarán las alternativas que ofrece Condor para su instalación, configuración y puesta en marcha, con el fin de encontrar el mejor entorno que se adapte a las necesidades que se tienen y a los recursos disponibles, con vistas al Área de Ingeniería Telemática de la UPCT.

Capítulo3

Universo Standard

3.1 Introducción a Standard Universe

En el capítulo 2 se explican los entornos más destacados en los que un Condor Pool puede trabajar. Estos son el “Vanilla Universe”, “Standard Universe”, “Globus Universe”, “PVM Universe” y “MPI”. Se describe cada uno de ellos, dejando el Universo Standard para este capítulo ya que es el Universo que se ha elegido para la instalación de Condor. Las razones del porqué de este Universo se mostrarán a lo largo de este capítulo. A continuación se describe el Universo Standard.

El Universo Standard se caracteriza por la utilización de mecanismos que aseguran la ejecución de una tarea en un Condor Pool configurado con este entorno. De una manera intuitiva, este entorno permite que una máquina que somete una tarea, percibe de forma transparente que dicha tarea se está ejecutando en su máquina de forma local. En caso de que la ejecución de la tarea falle en una máquina, su ejecución continua en otra máquina distinta desde donde se quedó (no comienza la ejecución desde el principio), ahorrando así tiempo de ejecución, ofreciendo un buen rendimiento de procesamiento y proporcionando al usuario una ejecución de sus tareas estable y fiable [23]. Todo esto lo ofrece Condor en el Universo Standard, mediante los elementos:

- Checkpointing
- Remote System Calls

Estas figuras permiten que se mantenga un acceso uniforme para obtener los recursos por parte de cualquier usuario en un Condor Pool, además de la estabilidad y fiabilidad antes comentada. Por ello, este universo es el más seguro a la hora de ejecutar tareas en Condor, pero a la vez, el más laborioso tanto en la instalación y configuración, como en la puesta en funcionamiento y la ejecución de tareas, como se verá en el siguiente capítulo.

El funcionamiento de un Condor Pool con estos dos elementos instalados y configurados en el mismo es el siguiente [24]:

En el caso del Checkpoint, cuando una máquina Submit somete una tarea en un Condor Pool, esta se ejecuta en la máquina Execute. Mientras la tarea se encuentra en proceso de ejecución, aparece la figura del Checkpoint, cuyo trabajo consiste en chequear la tarea que se está ejecutando a intervalos regulares, creando en cada uno de esos intervalos lo que se conoce como “imagen checkpoint”. Una imagen no es más que una instantánea del estado de la tarea en ese momento de chequeo. El objetivo de las imágenes checkpoint es el de mantener la ejecución de las tareas en caso de fallos.

La siguiente figura muestra el funcionamiento de Checkpoint en un Condor Pool cuando una máquina Submit somete una tarea. La máquina Execute ejecuta dicha tarea y se van creando las imágenes checkpoint que se guardan en la máquina Submit, aunque se verá más adelante que dichas imágenes Checkpoint se pueden guardar en un Servidor Checkpoint, en caso de tener uno instalado y configurado en un Condor Pool (apartado 3.2.3).

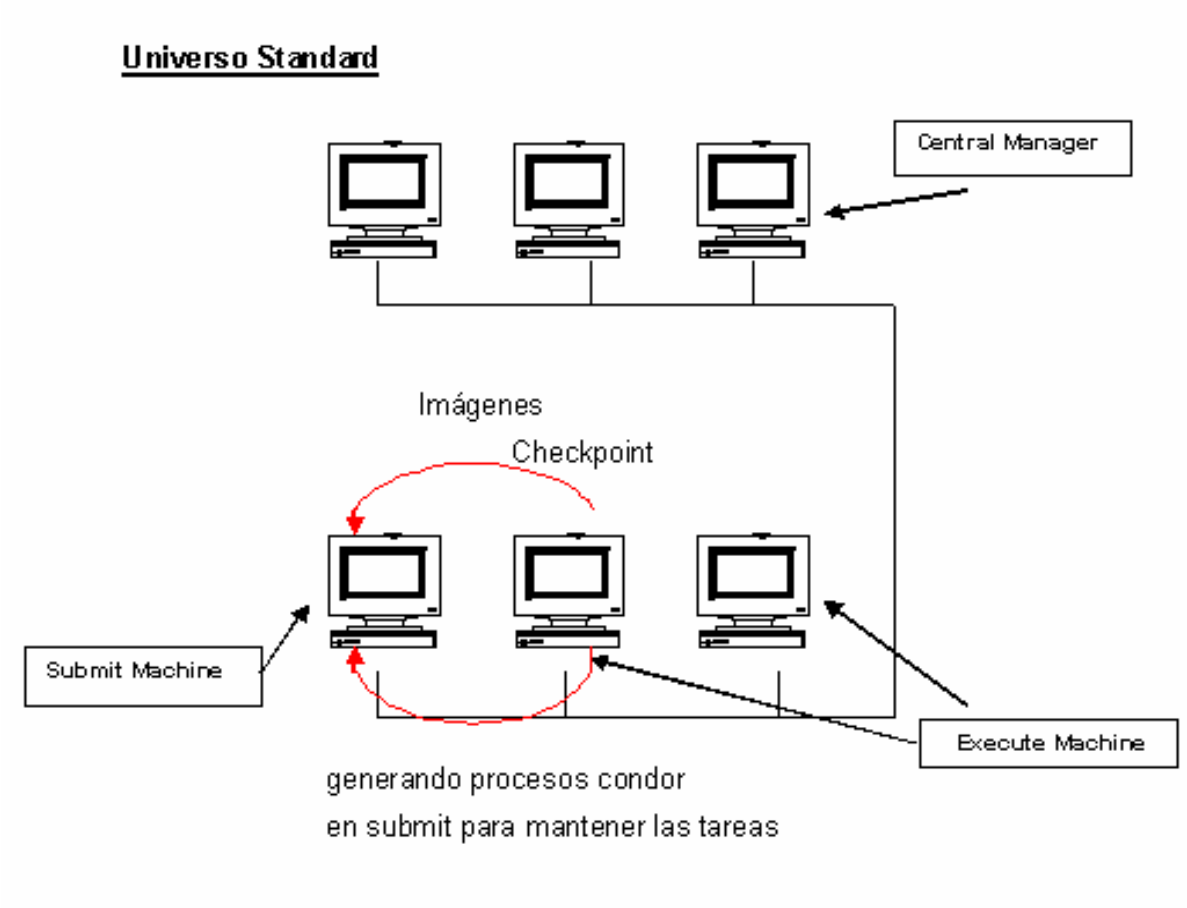

Figura 3. 1 Sometiendo tareas en un Condor Pool

Si una máquina Execute falla mientras se ejecuta una tarea en ella, o simplemente, la tarea, sufre algún fallo en tiempo de ejecución, para que la ejecución de dicha tarea pueda continuar, esta se debe de migrar a otra máquina Execute. Aquí aparece la figura de la imagen del checkpoint.

El mecanismo es el siguiente:

1- Condor hace una copia de la última imagen checkpoint de la tarea que se estaba ejecutando en la máquina que falló.

2- Envía esa imagen a otra máquina Execute que se encuentre disponible en el Condor Pool para ejecutar tareas, reiniciando la tarea desde la imagen checkpoint que se ha enviado a la máquina.

La negociación de la nueva máquina que debe ejecutar la tarea irá a cargo del Central Manager (se verá en los siguientes apartados). Este mecanismo permite que las tareas en un Condor Pool puedan estar ejecutándose durante meses o años, aunque ocurran fallos en dichas máquinas [25].

En la siguiente figura se muestra lo que ocurriría en caso de que una máquina Execute fallara mientras está ejecutando una tarea en un Condor Pool (figura 3.1.2). Una vez encontrada la máquina Execute en el Condor Pool para poder seguir la ejecución, se inicia la ejecución de la tarea desde la imagen y continúa la ejecución (figura 3.1.3).

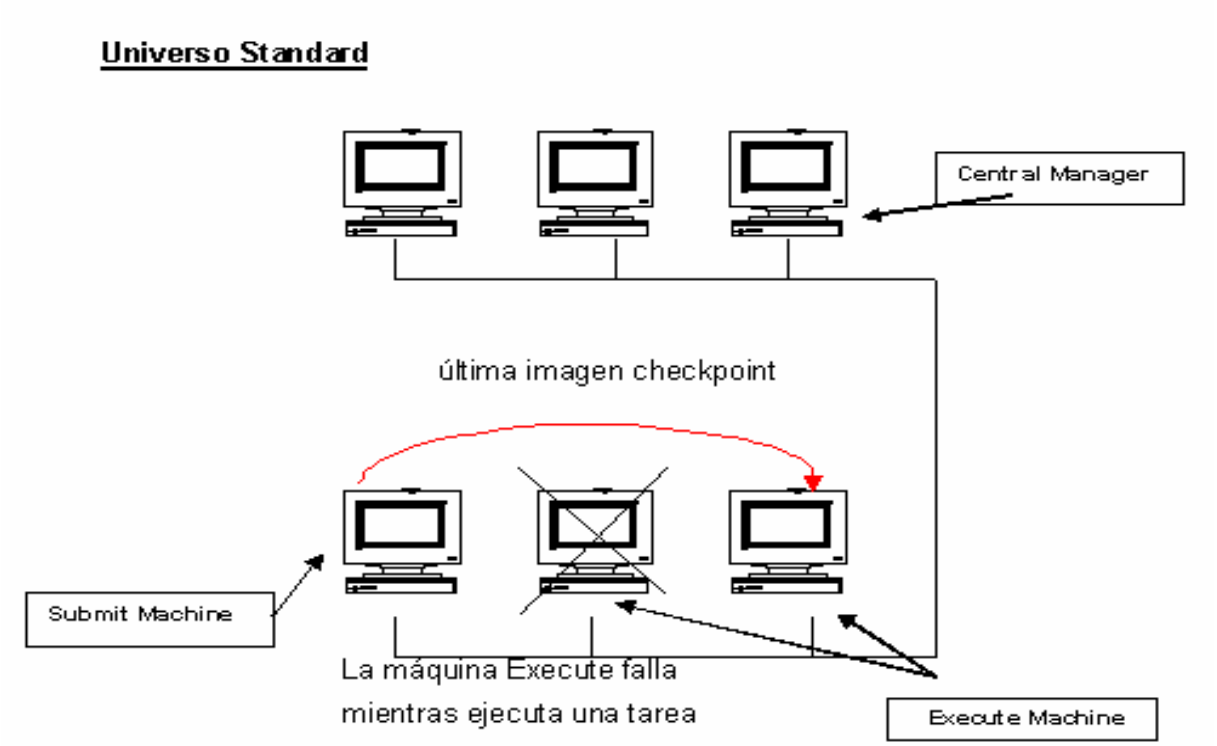

Figura 3. 2 Fallo de una máquina Execute sometiendo tareas

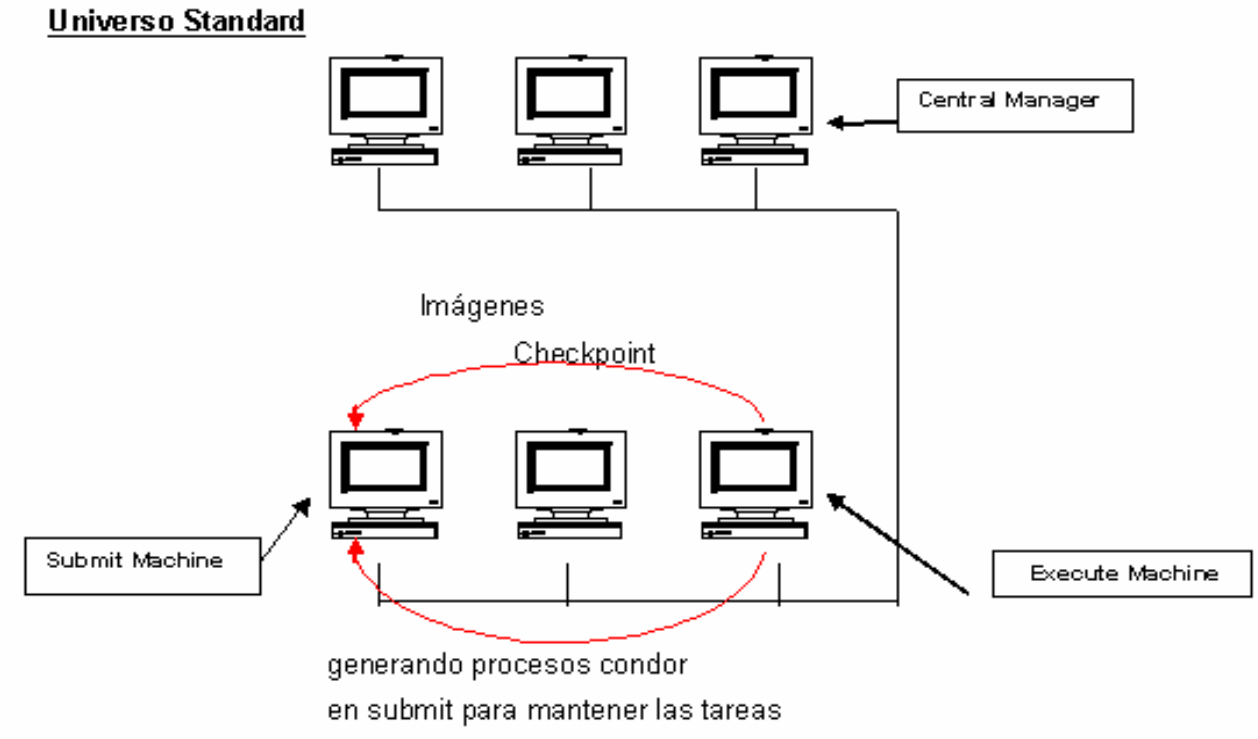

Figura 3. 3 Sometiendo tareas en la nueva máquina Execute

El segundo elemento imprescindible en el Universo Standard es Remote System Calls (capítulo 2). Las llamadas del sistema remoto hacen que una tarea perciba que

se está ejecutando en la máquina local, incluso aunque pueda ejecutarse en máquinas diferentes en el tiempo de vida [15].

Cuando una tarea se ejecuta en una máquina remota, un segundo proceso llamado “condor_shadow” se ejecuta en la máquina donde está sometida la tarea. En ningún momento es la tarea la que realiza la llamada al sistema, es el “condor_shadow” quien lo hace y envía el resultado.

Esto es muy útil, ya que si por ejemplo, al someter una tarea, dicha tarea debe abrir un fichero para guardar un resultado en la máquina Submit, es el proceso “condor_shadow” el que se encarga de buscar dicho fichero en la máquina y enviar los datos del fichero a la máquina donde se encuentra la tarea para ser ejecutada.

La siguiente figura muestra como se realizan las comunicación mediante Remote System Calls entre una máquina Submit y una máquina Execute y cómo es el proceso condor_shadow el encargado de mantener dicha comunicación.

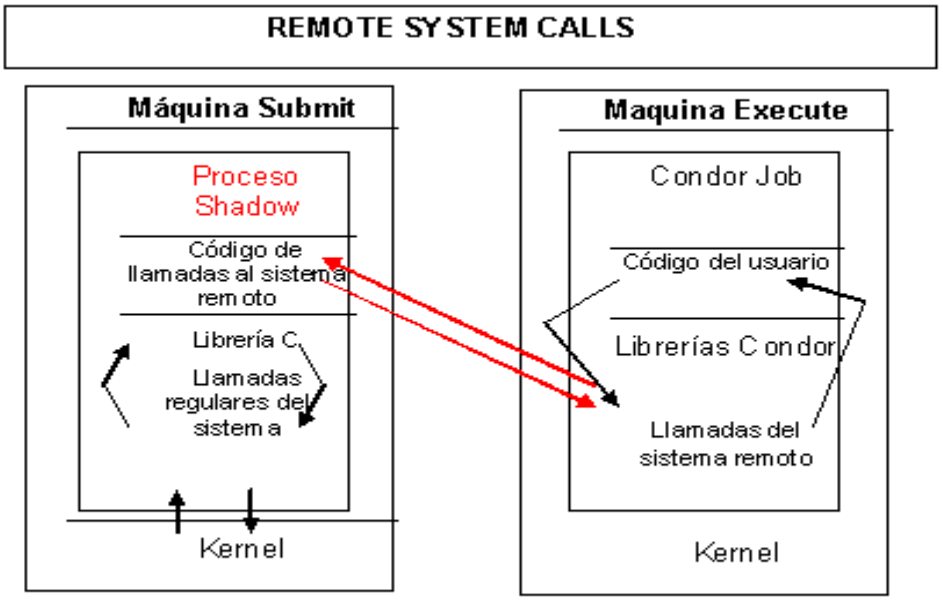

Figura 3. 4 Remote Systema Calls al someter tareas en un Pool

Una vez analizado el Universo Standard ya se tiene una idea de todos los universos que ofrece Condor en la versión que se va a trabajar (6.2.1).

Teniendo en cuenta las necesidades del entorno en el que en un futuro se pretende instalar Condor, es decir, una red con NFS y NIS, dicha instalación podría decantarse por el entorno Vanilla, pero se necesita un entorno que sea capaz de mantener tareas durante días e incluso semanas debido al simulador que se pretende someter a ejecución. Para este tipo de simulaciones, se debe asegurar la ejecución en todo momento, aunque alguna máquina falle. Este requisito hace que la elección final sea el Universo Standard, aunque dicho universo suponga una instalación y configuración mucho más compleja, como se verá en el siguiente capítulo, pero posee los mecanismos que interesan para la situación de la red que se pretende instalar, como es el caso de RPC, que a continuación se comentará.

3.2 Elementos en Standard

Los elementos que se citan a continuación son mecanismos que pueden mejorar el rendimiento de Condor con la instalación del Universo Standard en un Condor Pool.

Los elementos que se citan a continuación son mecanismos que pueden mejorar el rendimiento de Condor con la instalación del Universo Standard en un Condor Pool. Estos son el sistema de ficheros distribuido (NFS), servicio de información distribuida (NIS), Servidor Checkpoint…

Algunos de estos elementos, como es el caso de NFS y NIS se encuentran instalados actualmente en los laboratorios del Área de Ingeniería Telemática de la UPCT, con lo cual, interesa que Condor pueda convivir con estos mecanismos ya que en un futuro próximo estos laboratorios serán el escenario final del proyecto.

Tanto NFS como NIS son servicios cuya base reside en el mecanismo de las "llamadas de procedimiento remoto", RPC. Por ello, lo primero de todo será definir este mecanismo para poder entender el funcionamiento de los demás.

3.2.1 Remote Procedure Call

RPC (Remote Procedure Call) es una infraestructura cliente/servidor que incrementa la interoperabilidad, portabilidad y flexibilidad de una aplicación permitiendo que esta pueda distribuirse por medio de múltiples plataformas heterogéneas [26].

La llamada de procedimiento remoto RPC fue creada por Birrell y Nelson en el año 1984 con el fin de intentar que los programas pudieran llamar a procedimientos localizados en otras máquinas de manera similar a como se realizan las llamadas de procedimiento local [57].

El objetivo de esta infraestructura es reducir la complejidad del desarrollo de aplicaciones que abarcan diferentes sistemas operativos y protocolos de red al aislar al programador de los detalles de estos entornos resolviendo los diferentes problemas que suelen surgir en comunicaciones cliente/servidor:

- Ambos procesos están en espacios de direcciones diferentes.
- Transferencia de parámetros y resultados.
- Heterogeneidad.
- Fiabilidad.
- Localización y selección de servicios.
- Seguridad.

La tecnología RPC aumenta la flexibilidad de una arquitectura, al permitir al componente cliente de una aplicación que utilice una llamada RPC para acceder a un servicio ubicado en un sistema remoto. Gracias a RPC, para acceder a este componente remoto no se necesita saber la dirección de red ni ningún otro dato de bajo nivel [27].

Las llamadas de procedimiento remoto RPC vienen dadas por servicios RPC que forman parte de la comunicación entre cliente y servidor. Estos servicios son:

- Portmapper. En cada máquina se ejecuta un portmapper que mantiene un registro de todos los servicios RPC que se están ejecutando en la máquina.

La función del portmapper es la de localizar el puerto correcto de un servicio basándose en el número de programa y el número de versión.

- El rpcinfo es el componente administrativo de RPC. Con la opción –p este comando lista de los servicios que se están ejecutando en la máquina, el puerto, la versión y el número de programa. A continuación se muestra el resultado que daría la ejecución de este comando en una máquina que actúa como servidor de una red.

```
maquina1:~ # > rpcinfo -p
   programa vers proto   puerto
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp    735  status
    100024    1   tcp    737  status
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100021    1   udp   1024  nlockmgr
    100021    3   udp   1024  nlockmgr
    100021    4   udp   1024  nlockmgr
    100005    1   udp   1026  mountd
    100005    1   tcp   1024  mountd
    100005    2   udp   1026  mountd
    100005    2   tcp   1024  mountd
    100005    3   udp   1026  mountd
    100005    3   tcp   1024  mountd
```

La siguiente figura muestra como se produce dicha comunicación entre el servidor y el cliente mediante el portmapper de las máquinas. Los pasos que siguen los procesos son:

- El servidor lanza al portmapper información sobre el programa que está utilizando y el puerto a partir del cual se puede acceder a dicho programa.
- El portmapper añade a su lista dicho programa, la versión del mismo, el protocolo que utiliza y el puerto por el cual se puede acceder.
- El cliente quiere ejecutar dicho programa y de forma transparente pregunta al portmapper por el puerto donde se encuentra dicho programa para ejecutar. El portmapper le da la información mirando las tablas y de esa manera el cliente ya puede acceder al programa.

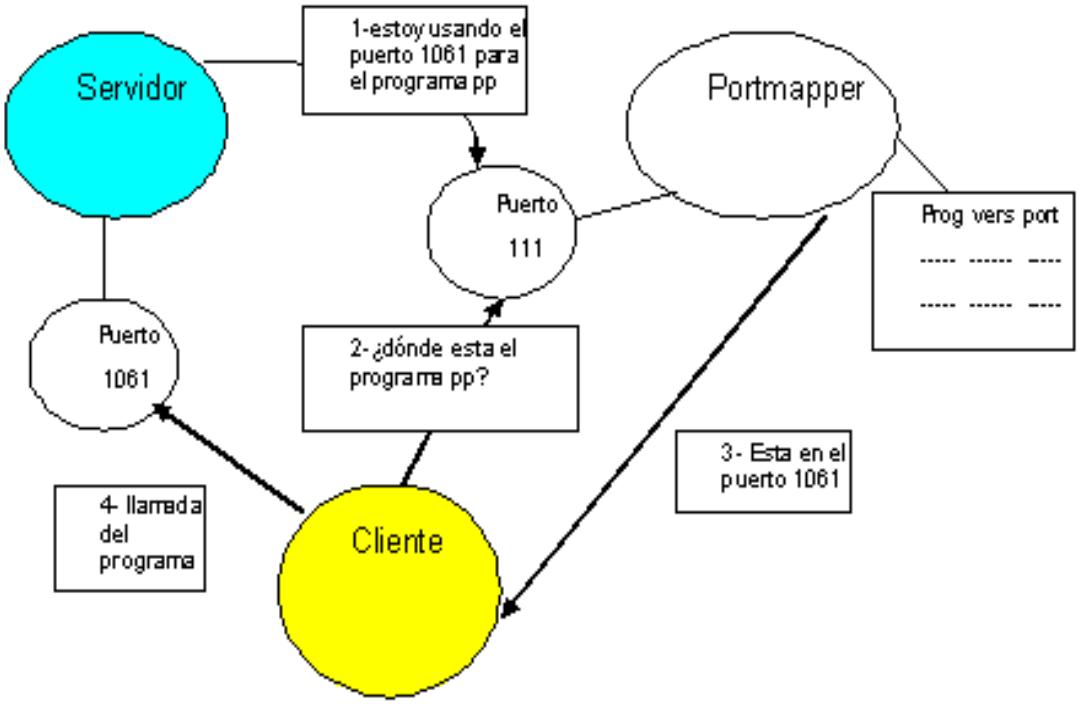

Figura 3. 5 Registro y localización de un servicio RPC

RPC es el mecanismo apropiado para aplicaciones cliente/servidor en las que el cliente genera una petición y puede esperar la respuesta del servidor antes de continuar con su propio proceso.

Dado que muchas implementaciones de RPC no admiten la interacción de igual a igual, o asíncrona, entre cliente y servidor, RPC no está indicada para aplicaciones que implican objetos distribuidos o programación orientada a objetos.

3.2.2 NFS

El Universo Standard puede ser instalado en una red provista de un sistema de ficheros distribuida para poder mejorar la calidad de procesamiento de Condor en la red. Por esta razón, parece obvio explicar el funcionamiento de estos sistemas, en particular de NFS, ya que será el sistema de ficheros distribuido utilizado en el Condor Pool de la red diseñada para el proyecto (capítulo 4).

NFS (Network File System) es un sistema de ficheros distribuido obra de Rick Sladkey y respaldado por Sun cuyo objetivo es el de poder compartir archivos en Red mediante el protocolo TCP/IP [28].

Este servicio permite compartir directorios y ficheros en Unix en una red LAN o WAN, aunque, hoy en día también es soportado por el Sistema Operativo Windows. Los usuarios que disfrutan de este sistema gozan de una transparencia total, es decir, se tiene la impresión de que los ficheros son locales a la máquina. Los archivos no presentan diferencia aparente entre la lectura y escritura de los locales y los remotos. Esto se logra mediante RPC, como se comentó en el apartado anterior [29].

3.2.2.1 Funcionamiento de NFS

El funcionamiento de este sistema de ficheros distribuido es bastante sencillo. Existe una máquina, que se encarga de exportar los ficheros que interesan en la red de forma jerárquica, exportando tanto los directorios como los subdirectorios que lo

componen (servidor NFS), y otra u otras máquinas que son las que montan el sistema de archivos (cliente NFS). Los clientes deben montar el sistema de archivos de acuerdo al servidor NFS si quiere tener acceso. Puede darse el caso de que el cliente no tenga disco. Si esto es así, se monta uno en memoria. Los clientes pueden montar los directorios de tres maneras diferentes:

-Manualmente

-Usando el comando mount

-Modificando el fichero etc/fstab

Una máquina puede ser cliente y servidor NFS al mismo tiempo. Hay que tener en cuenta que el servidor es sin estado, es decir, que no mantiene información sobre los clientes.

Los directorios que se exportan se encuentran en /etc/exports/. También se encuentran en este archivo la lista con los diferentes clientes que están autorizados a acceder y algunas restricciones más. Para ver los directorios montados bastará con utilizar el comando mount. Este comando devuelve los directorios montados de todos los sistemas de ficheros que el Kernel reconoce. Los que aparecen con el tipo nfs son directorios NFS.

Para entender mejor el funcionamiento se muestra a continuación una figura como ejemplo de la configuración y comunicación entre un cliente y servidor NFS. Estos se configuran de la siguiente manera:

- El servidor exporta /usr/local/pp.
- El cliente configura su punto de anclaje en /home/pp

Así, el cliente accederá a los ficheros y directorios que ha exportado el servidor, desde el path /home/pp

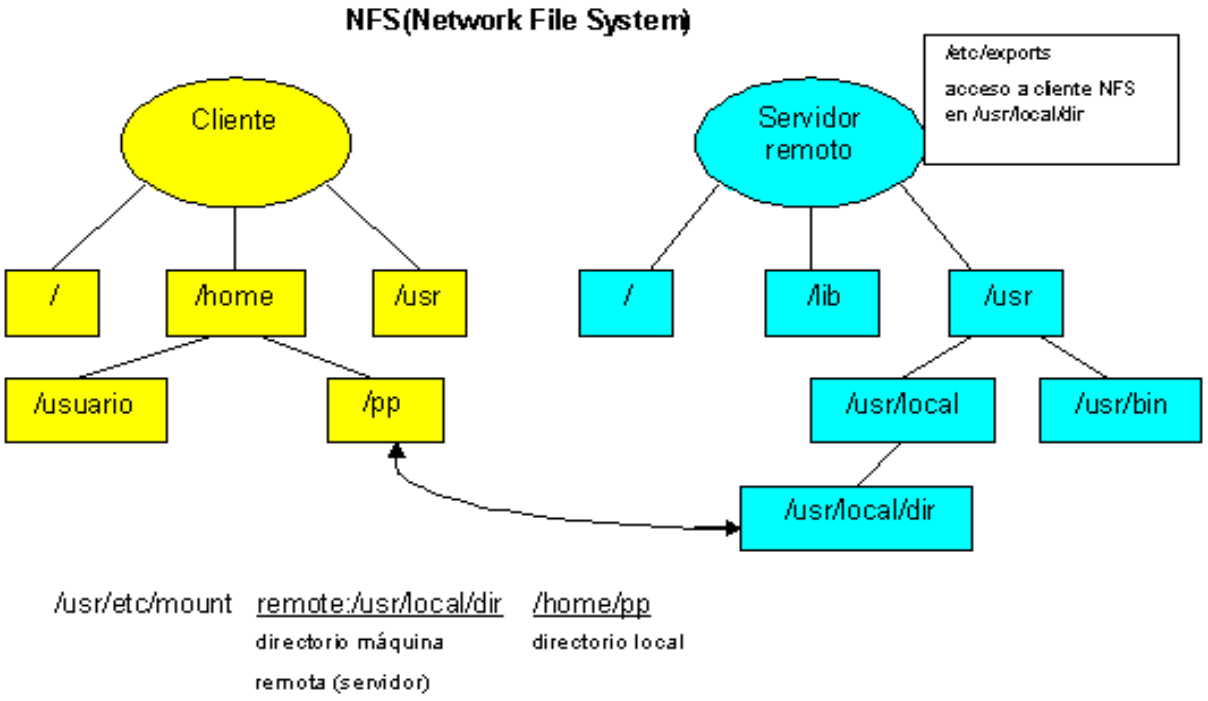

Figura 3. 6 Comunicación enter cliente y servidor NFS

3.2.2.2 Comunicación NFS mediante RPC

En cuanto a la comunicación mediante RPC, se muestra otra figura que muestra como NFS es un servicio que trabaja mediante RPC, teniendo el mismo cliente y servidor.

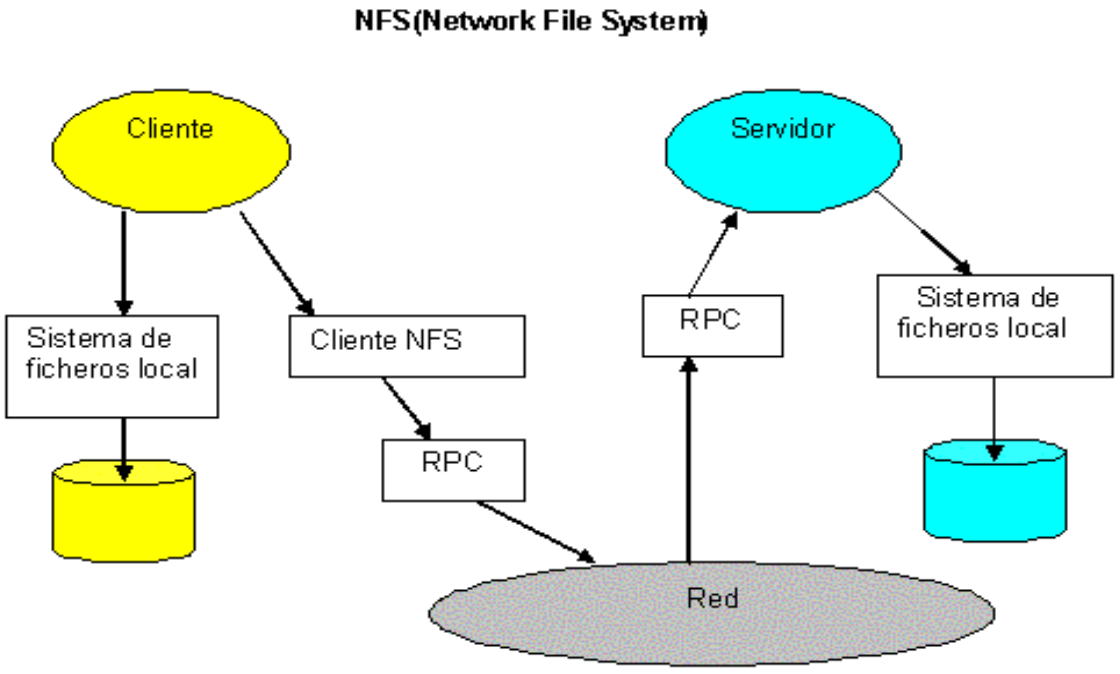

Figura 3. 7 Comunicación entre cliente y servidor NFS con RPC

Algunos de los problemas que se pueden tener con NFS son:

1- Se debe tener en cuenta que NFS es un protocolo cliente-servidor que no sincroniza los relojes de las máquinas, por lo que el servidor puede ir avanzado con respecto al cliente y viceversa, lo que puede producir inconsistencias en las horas de los archivos [30].

2- Otro problema que se presenta en el protocolo NFS es el de la identificación de usuario (UID) y la identificación de grupo (GID). Se sabe que en Unix todo usuario tiene asociada un UID y un GID. El problema es que estos identificadores de un usuario en la máquina cliente y el mismo usuario en la máquina servidor pueden ser diferentes, por lo que el usuario no puede tener acceso a sus propios archivos. Para solucionar este problema bastará con instalar NIS en la red de trabajo, que se verá en el siguiente apartado.

3.2.3 NIS

NIS (Network Information Service) conocido también como “páginas amarillas” (yellow pages) es un servicio de información desarrollado por Sun que proporciona una gestión centralizada de información de red (no solo de máquinas y direcciones IP, también de passwrods, servicios, grupos…) [31].

El uso de este elemento en Condor viene a raíz del uso de NFS. Como se mencionó antes, NFS trae consigo el problema de identificación de usuario (UID) y la identificación de grupo (GID). El problema de estos identificadores reside en que un

usuario de una máquina cliente puede tener una UID distinta en la máquina del servidor. Por ello, la solución a este problema es la instalación de NIS en la red de trabajo donde se encuentre la red NFS.

3.2.3.1 Elementos de NIS

En cuanto a los elementos que forman parte de NIS en una red son:

- Un dominio de trabajo. El dominio NIS es el nombre aplicado a un grupo de mapas NIS. Las máquinas que necesitan ver la información de esos mapas debe pertenecer a dicho dominio. El dominio define un entorno de manejo de sistemas. Cada dominio tiene un nombre. Cada máquina debe pertenecer a un dominio. No existen restricciones de donde una máquina debe estar para pertenecer a un dominio u a otro.

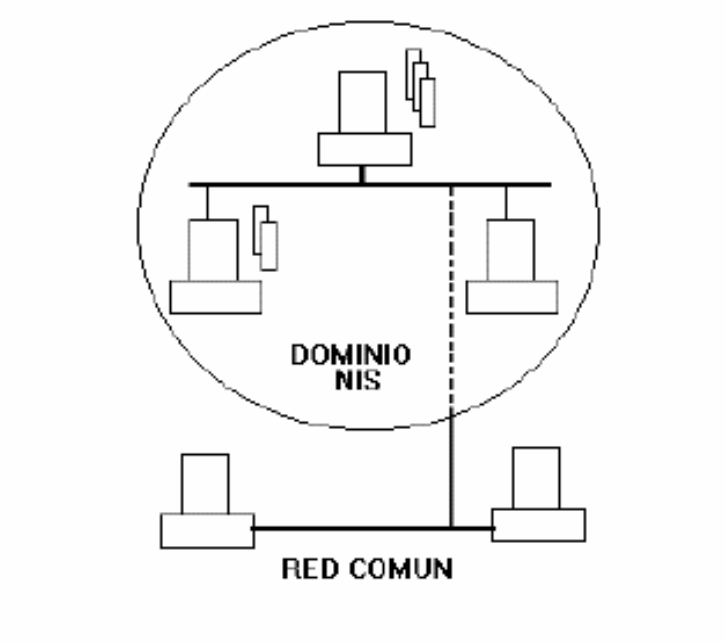

Figura 3. 8 Dominio NIS en una red

- Los mapas. Son los archivos de datos que maneja el NIS, de los cuales los dientes obtienen la información. La información NIS se almacena como una base de datos relacional en los mapas, que no son más que archivos dbm (database system) localizados en el directorio etc/yp en las máquinas que trabajan como servidores NIS.

  Los mapas son almacenados en un nodo central que ejecuta el servidor NIS y del que los clientes pueden obtener información a través de varias llamadas RPC. Estos mapas contienen pares clave-valor. Algunos ejemplos de mapas NIS son: hosts, protocols, passwords, rpc, services, group, netgroup, timezone...

- Demonios. Estos se refieren a los demonios que representan el servidor NIS y los clientes NIS. El servidor NIS suele llamarse tradicionalmente “ypserv” y los clientes “ypbind”. Para una red de tipo medio con un servidor suele ser suficiente, como en el caso que se tratará en el siguiente capítulo.

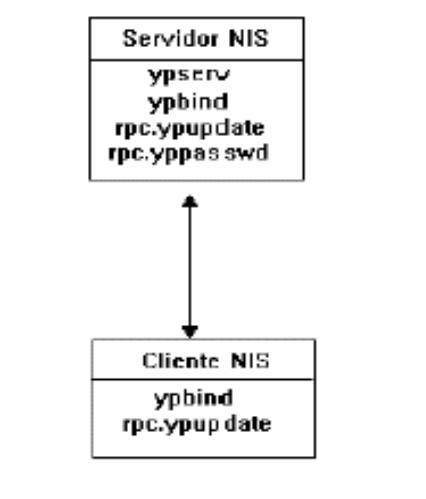

Figura 3. 9 Servidor y Cliente NIS

- Utilidades que se verán más adelante: ypcat, ypwhich, ypmatch, ypinit, yppoll….
- Utilidades auxiliares: yppush, ypset, ypxfr, makedbm…

Hay tres tipos de máquinas que pueden formar parte de NIS y estas son el Servidor Maestro, el Servidor esclavo y el Cliente.

NIS está construido bajo el modelo cliente servidor. Cualquier máquina puede ser cliente NIS, pero sólo las que poseen disco (por razones obvias) pueden ser servidores (maestro y esclavo). Los servidores pueden, a su vez, ser también clientes.

Servidor NIS

Por definición un servidor NIS es una máquina que contiene los archivos de datos, llamados mapas. Existen dos tipos de servidores, los servidores maestros y los servidores esclavo.

El servidor maestro

Es el que contiene los mapas y es dueño de los mismos, él es responsable de mantenerlos y distribuirlos a los servidores esclavos, en él solo se pueden hacer las actualizaciones.

El servidor esclavo

Es un servidor secundario que tiene una copia exacta de todos los mapas NIS que tiene el servidor maestro, en el servidor esclavo nunca se pueden hacer actualizaciones (reconstruir los mapas). El servidor esclavo permite minimizar el impacto que ocurrirá en la red, si el servidor maestro por alguna razón no está operativo. Un servidor esclavo puede, al igual que un servidor maestro, contestar una solicitud realizada por alguna máquina cliente.

Cliente

El cliente NIS, ejecuta un proceso que permite obtener los datos de un mapa del servidor, obteniendo la información del servidor a través del proceso “binding”.

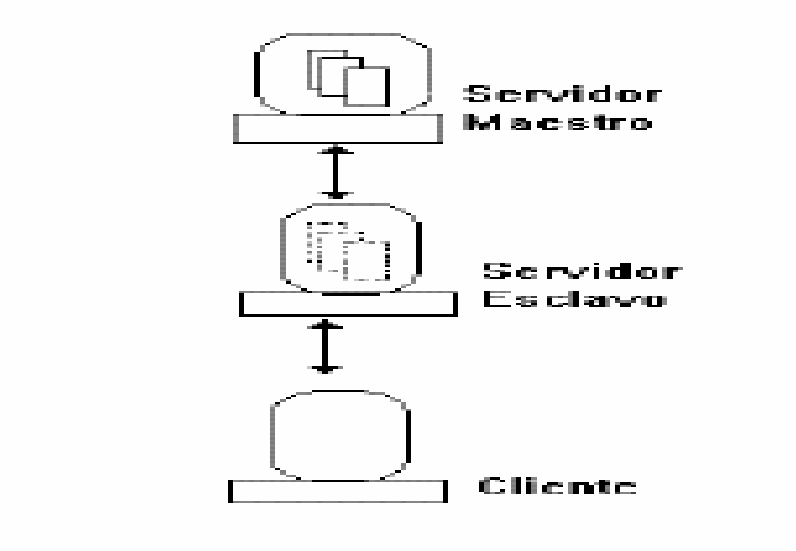

Figura 3. 10 Servidor Maestro, esclavo y cliente NIS

3.2.3.2 Funcionamiento de NIS

El funcionamiento de NIS en una red se reduce a que cuando un cliente NIS en vez de acceder a los archivos de etc/, hacen una llamada RPC al servidor cada vez que el cliente necesite información de una base de datos NIS [32].

A continuación se muestra una figura donde se dan lugar esas llamadas entre un cliente y un servidor NIS. Los pasos que se siguen son los siguientes:

1- Un programa que se está ejecutando en el cliente, necesita información provista por un mapa NIS, así que pregunta al ypbind por el nombre del servidor.

2- La función del ypbind es la de recordar la dirección en la cual el proceso ypserv está atento para responder; ypbind envía un "rpc.broadcast".

3- El proceso ypserv responde; con la información solicitada por el cliente guardada en el archivo "domainname".

4- ypbind le dice al proceso cliente cuál es el servidor que debe hablarle. El cliente envía la solicitud al servidor.

5- El demonio ypserv en el servidor NIS maneja la solicitud como una consulta al mapa y este envía la información de vuelta al cliente.

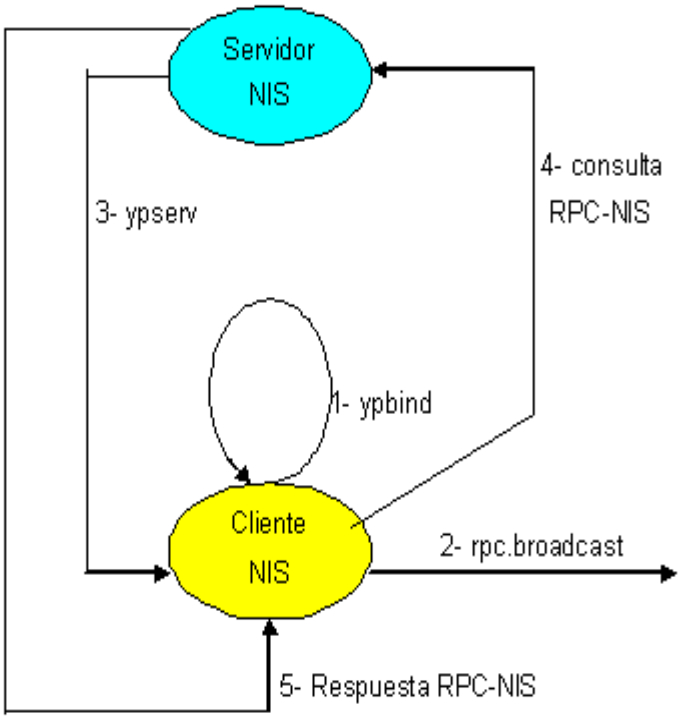

Figura 3. 11 Cliente-Servidor NIS

Por último se comentan los comandos más comunes utilizados en NIS:

ypinit

Crea en el servidor maestro o esclavo un resumen de los comandos NIS.

ypbind

Demonio. Crea unos ficheros con información para enlazar (bind) un cliente con su servidor. Cada 20 segundos chequea el servidor.

ypwhich

Devuelve el nombre del servidor o el nombre del servidor master si se da un mapa.

ypcat

Devuelve los campos clave de un mapa.

ypserv

Demonio. Recibe las peticiones de los clientes y controla el acceso a los mapas NIS.

domainname

Devuelve el nombre del dominio de la máquina.

rpcinfo

Lista los servicios RPC que hay registrados en una máquina.

3.2.4 Servidor Checkpoint

Otro de los elementos que forman parte del Universo Standard es el Servidor Checkpoint. Este no es un elemento indispensable para que Condor funcione correctamente en un Condor Pool, pero si que reduce la necesidad de bastantes recursos en las máquinas que forman parte del Pool, como se verá a continuación.

Antes de entrar en detalle en el Servidor Checkpoint se comenta el concepto de imagen de checkpoint, que como se explicó en la introducción de este capítulo, no es más que una instantánea del estado de una tarea mientras se encuentra dicha tarea en proceso de ejecución.

La importancia de la imagen Checkpoint reside en Condor en que si, por ejemplo, una tarea que se está ejecutando se interrumpe, quizás porque el propietario de la máquina ha vuelto a su puesto, Condor chequea la tarea y la instantánea de la tarea actualizada, es decir, la última instantánea realizada a la tarea. Condor la manda a otra máquina para poder continuar con la ejecución y así mantiene en el Condor Pool la fiabilidad de la ejecución de las tareas, aunque estas tengan una larga duración.

Para entender la necesidad del uso de un Servidor Checkpoint en un Condor Pool, se propone la siguiente figura, que no es más que un ejemplo de un Condor Pool, donde una máquina Submit somete una tarea (capítulo 2). Varias máquinas Execute se encargan de ejecutar las tareas sometidas por la máquina Submit del Condor Pool, y las imágenes checkpoint de esas tareas se van almacenando en la máquina Submit.

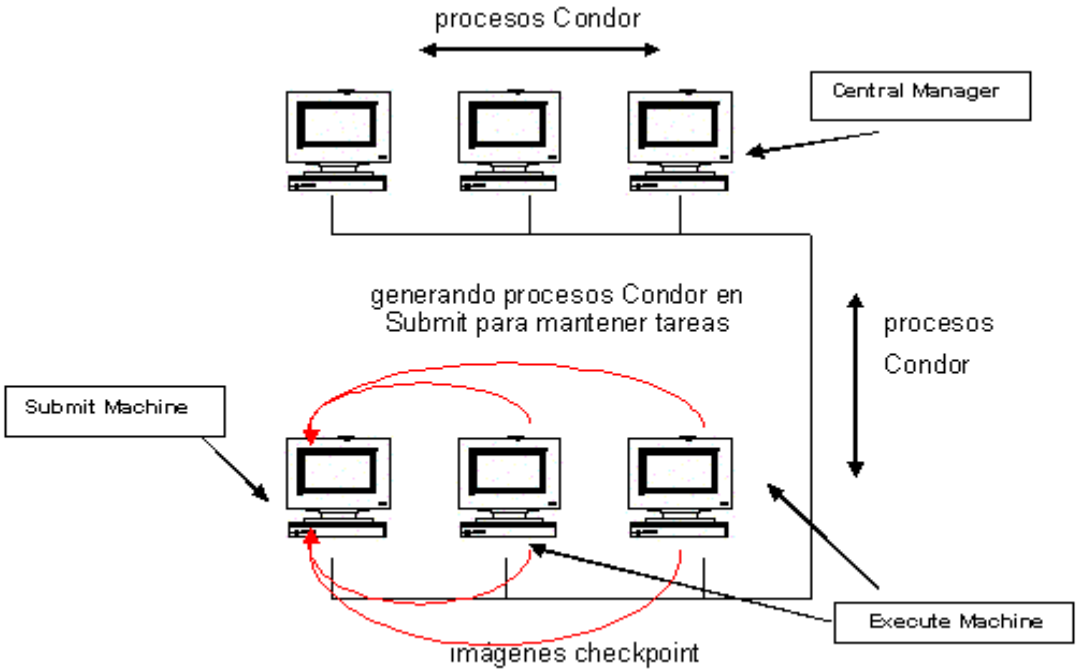

Figura 3. 12 Sometiendo tareas en un Condor Pool

Si se configura un Condor Pool de esta manera, es obvio que la máquina Submit debe tener mucho espacio de disco para poder almacenar todas esas imágenes checkpoint que se van generando de las tareas, ya que las imágenes checkpoint ocupan bastante espacio de disco. También hay que tener en cuenta que se esta

trabajando en el Condor Pool con un entorno Standard, donde se puede configurar el Pool para utilizar un servidor de ficheros, como se explicó en el apartado anterior. En caso de instalarlo, dicho servidor de ficheros deberá estar situado en la máquina que hace de Central Manager ya que este es el repositorio centralizado de la información del Condor Pool.

Un servidor de ficheros en un Condor Pool trabajando con el entorno Standard tiene una funcionalidad muy importante, ya que en caso de que una máquina Execute falle cuando está ejecutando una tarea, otra máquina Execute puede continuar con la ejecución de la tarea por medio de la imagen checkpoint y la imagen puede estar al alcance de las máquinas del Condor Pool almacenando dicha imagen en el Servidor de Ficheros. De esta manera, todas las máquinas clientes de ese servidor de ficheros (y con los permisos adecuados) tendrán acceso a las imágenes de las tareas de un Pool.

Si se utiliza el Condor Pool de la figura 3.2.4.1 y se configura en el un servidor de ficheros, y las demás máquinas se configuran como clientes del sistema de ficheros, las imágenes checkpoint que se creen de las tareas que se ejecuten harán dos viajes por la red:

1- De la máquina Execute a la máquina Submit.

2- De la máquina Submit a la máquina Servidor de Ficheros.

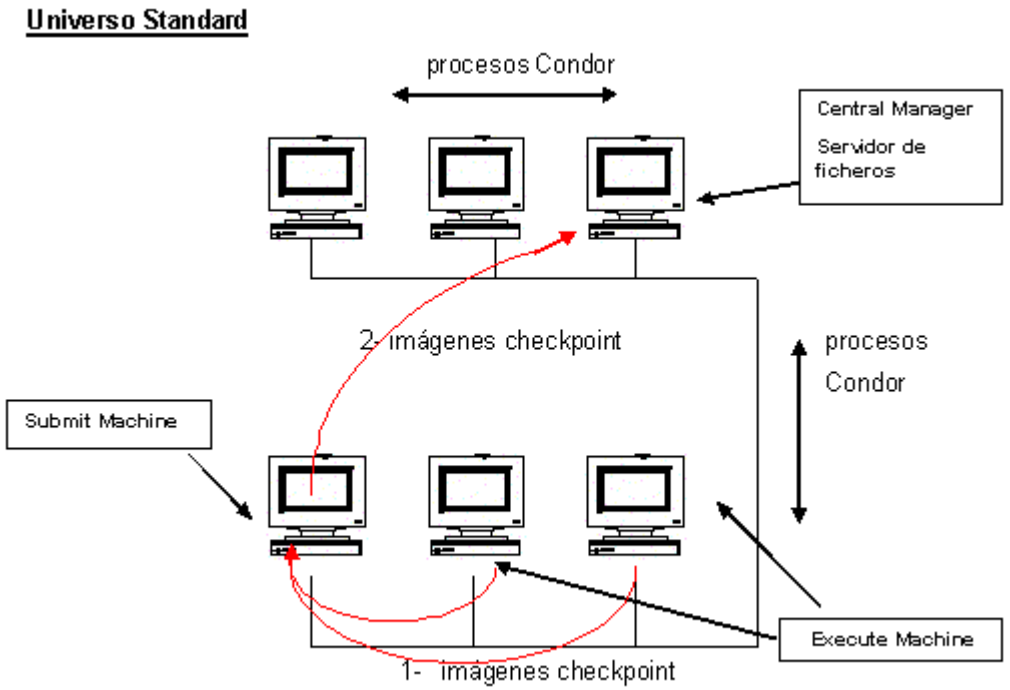

Figura 3. 13 Servidor de ficheros en un Condor Pool

Ante la situación del Condor Pool representado en la figura anterior se puede deducir que:

- El uso de las máquinas Submit como repositorios de los ficheros checkpoint sería comprensible si todas las máquinas que fueran a actuar como máquinas Submit en el Condor Pool tuvieran grandes recursos, pero hay que tener en cuenta que no todas las máquinas que deseen someter tareas en un

Condor Pool van a tener las mismas características ni los mismos recursos. Seguro que muchas de las máquinas que necesitan someter tareas en el Condor Pool se ven incapaces de hacerlo por la imposibilidad de almacenamiento de las imágenes checkpoint que generarán las tareas sometidas.

- Es un error permitir que los ficheros checkpoint realicen dos viajes por la red del Condor Pool por la mala configuración de las máquinas ya que se produce un tráfico innecesario en la red, además de una sobrecarga en la misma ya que estos ficheros tienen un gran tamaño.

Una vez vistos los problemas que surgen, se deduce que es bastante importante el introducir la figura de un Servidor Checkpoint en un Condor Pool para intentar solucionar los problemas antes mencionados [33].

Como se explicó en el capítulo 2, un Servidor Checkpoint, no es más que un repositorio de ficheros checkpoint de las tareas que se someten en un Pool. Utilizando un Servidor Checkpoint se reduce la necesidad de una gran capacidad de espacio de disco en las máquinas que someten las tareas, ya que, dicho espacio de disco, se requería para almacenar las imágenes checkpoint, pero usando este servidor, esas imágenes se almacenarán en el servidor, en vez de en las máquinas Submit.

Una vez visto la importancia que tiene el instalar un Servidor Checkpoint en un Condor Pool la pregunta que puede surgir ahora es saber donde se debe instalar dicho servidor. Se sabe que la máquina que se configure como Servidor Checkpoint debe ser una máquina que cumpla estos dos requisitos:

- Tener suficiente espacio de disco.
- Conexión rápida de las máquinas que forman parte del Condor Pool.

Si en el Condor Pool que se ha visto en la figura anterior, se configura una máquina que posee las características anteriores para actuar de Servidor Checkpoint, como se muestra en la figura 3.2.4.3, se puede observar como con esta distribución si que se distribuye la carga de la información entre la máquina Submit y el Servidor Checkpoint, solucionando ya uno de los problemas que surgían sobre los recursos de las máquinas Submit.

Pero todavía quedará otro problema por resolver, ya que si en el Condor Pool se está utilizando un sistema de ficheros distribuido, y el servidor de este sistema de ficheros no se encuentra en la máquina donde está instalado y configurado el Servidor Checkpoint, las imágenes checkpoint seguirán haciendo dos viajes por la red, pero en este caso los viajes serán entre:

1- La máquina Execute y la máquina Servidor Checkpoint

2- El Servidor Checkpoint y el Servidor del sistema de ficheros distribuido.

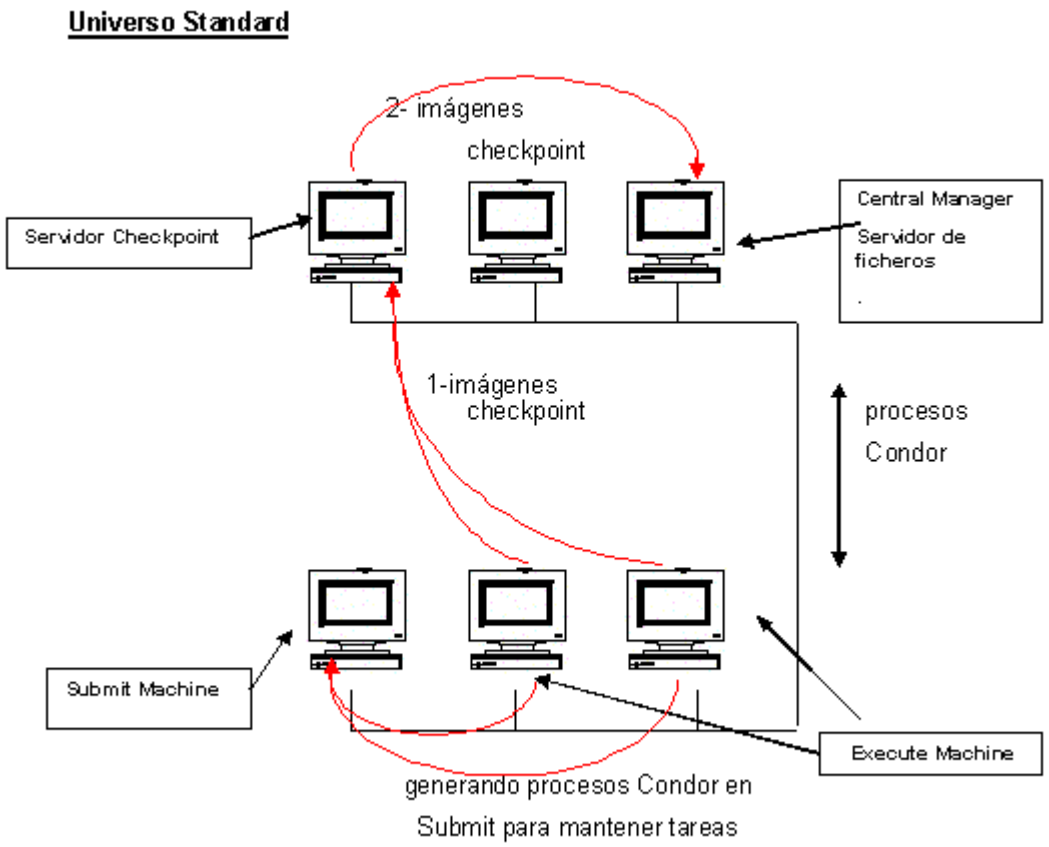

Figura 3. 14 Servidor Checkpoint en un Condor Pool

De la figura anterior se deduce que lo más lógico es instalar el Servidor Checkpoint en la máquina que va a ser configurada como el servidor del sistema de ficheros distribuidos. Además, como es conveniente instalar servidor de ficheros distribuido en el Central Manager, el Servidor Checkpoint se instalará en el Central Manager.

Por tanto, la máquina que actúe como Central Manager debe ser una máquina muy estable, con una buena conexión a la red, y por supuesto, tener bastante espacio de disco para almacenar los ficheros checkpoint de todas las máquinas del Condor Pool que sometan tareas [34].

La siguiente figura muestra como quedaría configurado un Condor Pool con un sistema de ficheros distribuido instalado en el Pool y un Servidor Checkpoint instalado en la misma máquina que el Central Manager y el servidor del sistema de ficheros. Con una configuración del Condor Pool como la de la figura, todos los problemas anteriores que podían surgir desaparecen, pero ahora hay que tener en cuenta que la máquina que tiene instalado el Servidor Checkpoint es la figura más importante del Pool como se verá a continuación.

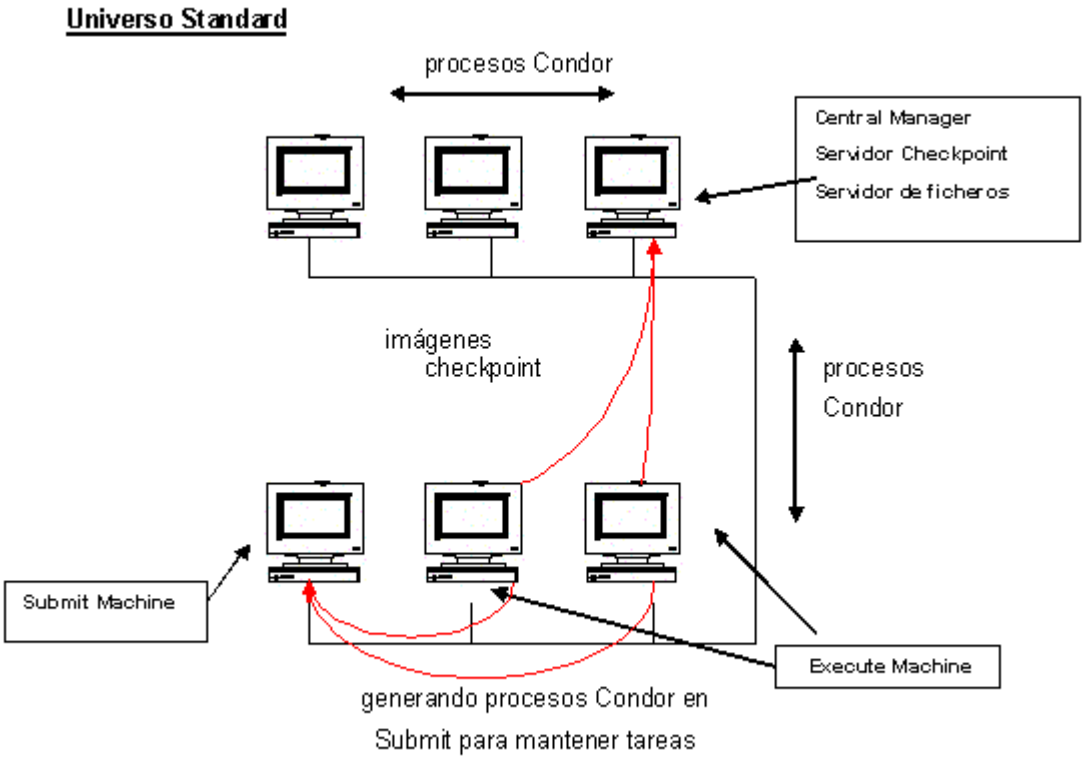

Figura 3. 15 Condor Pool con entorno Standard

3.2.4.1 Fallo del Servidor Checkpoint en el Condor Pool

Si un Servidor Checkpoint cae en un Condor Pool como el de la figura anterior, el Sistema Condor continuará operando, aunque muy pobremente ya que mientras el Sistema Condor intenta recuperar el Servidor Checkpoint caído, hay dos problemas que pueden surgir [34]:

1- Un fichero Checkpoint no puede ser enviado a un Servidor Checkpoint que no este funcionando. Por tanto, nadie almacena las imágenes de las tareas que se están ejecutando. Las tareas seguirán intentando contactar con el Servidor Checkpoint, disminuyendo exponencialmente el tiempo de espera entre intentos. Normalmente, las tareas solo tienen una limitación de tiempo para chequear antes de que se elimine la imagen checkpoint de la máquina. Así que si el servidor está inutilizado durante un largo periodo de tiempo, los cambios que surjan en las tareas que se están ejecutando en la máquina Execute se perderán sin haber sido escritos en el Servidor Checkpoint.

   La siguiente figura muestra el caso de un Condor Pool en el que el Servidor Checkpoint falla. Las máquinas del Pool que se encuentren ejecutando tareas, intentarán por todos los medios contactar con el Servidor Checkpoint para poder enviar las imágenes checkpoint de las tareas que se encuentran en proceso de ejecución en dichas máquinas.

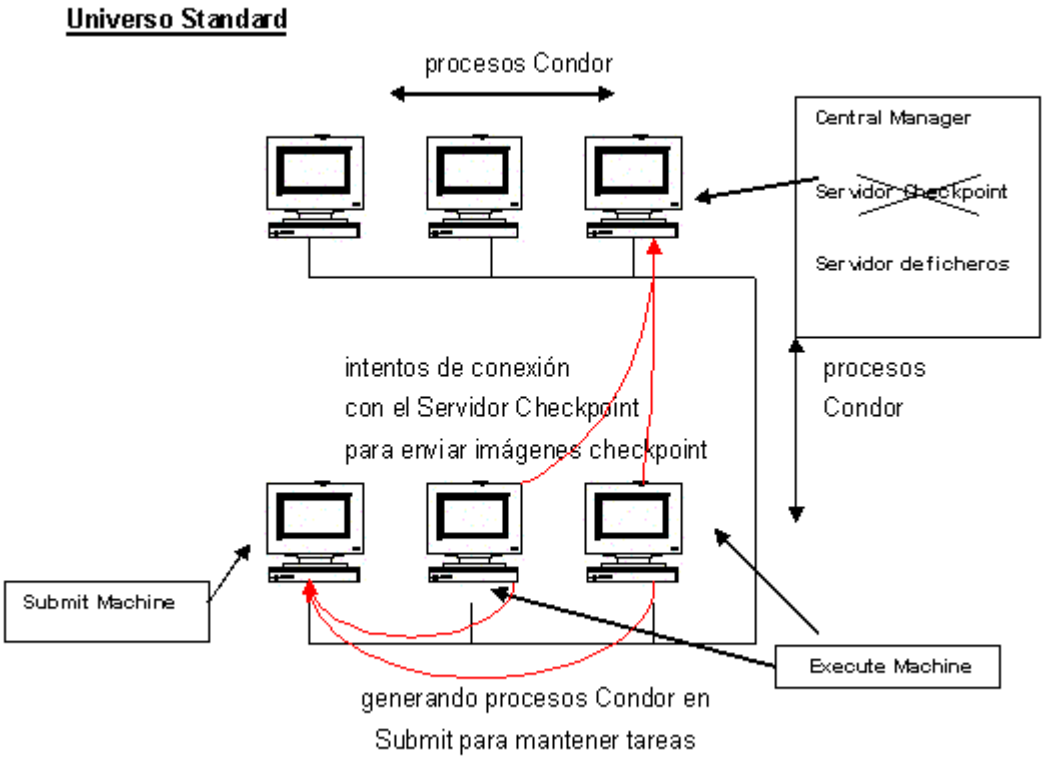

Figura 3. 16 Caida del Servidor Checkpoint

2- Si un Checkpoint no está disponible desde el Servidor Checkpoint, también puede ocurrir que la tarea se quede en estado de espera, hasta que el servidor vuelve a estar en línea. Este comportamiento se controla por medio del parámetro MAX_DISCARED_RUN_TIME en el fichero de configuración (capítulo 4). Este parámetro representa la cantidad máxima de tiempo en el CPU que debe pasar hasta que se descarte una tarea en caso de que el Servidor Checkpoint no responda.

3- El mayor problema que puede surgir es que en caso de que caiga el Servidor Checkpoint del Condor Pool durante un largo periodo de tiempo, y durante el mismo, una máquina Execute corte la ejecución de una tarea, por diversas causas, no se van a tener imágenes checkpoint de la tarea almacenadas en ningún lugar para poder seguir con la ejecución de la tarea que ha sido suspendida en otra máquina. La tarea deberá ser iniciada otra vez desde el principio en otra máquina Execute. Esto es algo que produce un gasto innecesario de tiempo de ejecución de las máquinas, haciendo el Condor Pool ineficiente. Es por ello que se debe estar muy seguro de dónde se instala el Servidor Checkpoint en el Condor Pool porque un fallo en este puede causar muchos problemas a los usuarios del Pool.

3.2.4.2 Múltiples Servidores Checkpoint en un Pool

Es posible configurar un Condor Pool para usar múltiples Servidores Checkpoint. El uso de múltiples Servidores Checkpoint se tiene en cuenta cuando en un Condor Pool hay un gran número de máquinas que someterán tareas y un gran número de máquinas Execute que generarán gran cantidad de ficheros checkpoint.

Toda esta cantidad de ficheros, quizás una única máquina no sea capaz de hacer frente y de ahí la necesidad de tener varios Servidores Checkpoint [35].

El despliegue de Servidores Checkpoint sobre la red de trabajo mejora el rendimiento de las máquinas para chequear los Servidores Checkpoint y enviar las imágenes checkpoint. En el caso de tener múltiples Servidores Checkpoint en un Condor Pool, las máquinas de mismo se configuran para chequear al Servidor Checkpoint más cercano. Hay dos puntos positivos principales sobre el rendimiento al desplegar múltiples Servidores Checkpoint en una red.

- El tráfico de la red de trabajo del Checkpoint se localiza mediante un emplazamiento inteligente de los Servidores Checkpoint.
- La rapidez del Checkpoint implica que las tareas tienen que gastar menos tiempo chequeando y más tiempo haciendo bien el trabajo, así las tareas tienen una buena oportunidad para chequear satisfactoriamente los Servidores Checkpoint, y los propietarios de las estaciones de trabajo pueden ver como las imágenes Checkpoint abandonan las máquinas con rapidez.

Uno de los conceptos a introducir cuando se instalan varios Servidores Checkpoint en un Condor Pool es el de "dominios del Servidor Checkpoint" [35].

Las configuraciones descritas anteriormente aseguran que las tareas siempre escribirán ficheros checkpoint al servidor más cercano.

En algunas circunstancias, también es bueno configurar Condor para localizar las transferencias de lectura de checkpoint, lo cual ocurre cuando las tareas se reinician desde la última imagen checkpoint en otra máquina Execute. Para localizar esas transferencias, se necesita fijar la tarea en la máquina que este más cerca del Servidor Checkpoint en el cual las imágenes checkpoint son almacenadas.

Cuando se dice que todas las máquinas están configuradas para usar un servidor Checkpoint "A", son en los dominios de dicho servidor. Para localizar las transferencias checkpoint, se requiere que las tareas que se ejecutan en las maquinas en un dominio de Servidor Checkpoint dado continúen ejecutándose en las máquinas en el dominio, transfiriendo ficheros Checkpoint. Con las tareas pueden ocurrir dos cosas:

1- La tarea puede quedarse en el estado de "idle" hasta que la estación de trabajo en el dominio del Servidor Checkpoint llegue a estar disponible.

2- La tarea puede intentar ejecutarse inmediatamente en una maquina en otro dominio de Servidor Checkpoint. En este caso, la tarea se transfiere a otro nuevo dominio de Servidor Checkpoint.

3.3 Tareas en el Universo Standard

La mayoría de los programas que se pueden someter por parte de las máquinas Submit para ejecutarse en máquinas Execute pueden ser preparados como tareas del entorno Standard, pero hay algunas restricciones [36] a la hora de programar dichas tareas:

3.3.1 Restricciones de las tareas

- No se permiten tareas de multiprocesos. Esto incluye las llamadas a sistemas tales como fork(), exec() y system().

- No se permite la comunicación de inter-procesos. Esto incluye tuberías, memoria compartida...
- La comunicación en la red de trabajo debe de ser breve. Una tarea puede hacer conexiones a la red de trabajo utilizando llamadas al sistema como socket(), pero si se deja la conexión a la red de trabajo abierta durante un largo periodo de tiempo, producirá retrasos en el checkpoint y migración.
- No se mantiene el envió o recepción de señales SIGUSR2 o SIGTSTP. Condor reserva estas señales para su uso propio. Si se mantiene el envió y recepción de otro tipo de señales.
- Tampoco se permiten las alarmas, timers o sleeping. Esto incluye las llamadas al sistema de alarm(), gettimer() y sleep().
- No se permiten múltiples hebras a niveles de Kernel. De cualquier manera si que se mantienen múltiples hebras a nivel de usuario.
- No se permiten ficheros de memoria mapeada. Esto incluye llamadas al sistema tales como map(). y munmap().
- Se permiten los ficheros locks, pero no son conservados entre checkpoints.
- Todos los ficheros se pueden abrir solo para lectura o solo para escritura. Un fichero que se abre con permisos tanto de lectura como de escritura puede causar problemas. Por razones de compatibilidad, un fichero abierto tanto para lectura y escritura resultará un “warning” pero no un error.
- Debe estar disponible en la maquina que somete tareas una cantidad justa de espacio de disco para almacenar las imágenes de las tareas de checkpoint. Una imagen checkpoint es aproximadamente igual a la memoria virtual consumida por una tarea mientras se ejecuta. Si hay poco espacio de disco, un servidor especial de checkpoint se puede designar para almacenar todas las imágenes de checkpoint para el pool.

En plataformas Unix, HP-UX y Linux, las tareas deben ser estáticamente compiladas. La compilación dinámica se mantiene en otro tipo de plataformas.

Una vez que se conocen las restricciones de las tareas, y se realizan los programas que se van a someter en el Condor Pool, estos deben seguir un proceso de compilado y recompilado con las librerías de Condor mediante el comando “condor_compile”.

Para utilizar este comando no es necesario modificar el programa ni el código fuente, pero si que se necesita el acceso para recompilar los ficheros objeto. En el caso de los programas comerciales, la mayoría de ellos se encuentran empaquetados en un único fichero ejecutable. Al no tener acceso al objeto del programa resulta imposible el poder convertirlo a una tarea del entorno Standard para poder someterla en el Pool.

3.3.2 Compilación

La compilación de la tarea, se realizará como en cualquier programa. Se creará el fichero objeto mediante el compilador. Este fichero será el que se utilice después para poder enlazar el la tarea con las librerías de Condor.

Por tanto, los pasos a seguir a la hora de compilar serán los siguientes:

Estando en la máquina Submit, y teniendo el programa que se quiere someter, este se compila con el compilador adecuado. Por ejemplo, un programa cualquiera escrito en c llamado “tareacondor.c” se compila escribiendo en la línea de comandos:

linux:~ # gcc –c tareacondor.c

Si la sintaxis del programa es la correcta, no habrá problemas y se habrá creado el fichero objeto “tareacondor.o”. A partir de este objeto se debe de crear el ejecutable que utilizará la aplicación Condor. Para ello se debe de enlazar con las librerías Condor mediante el comando “condor_compile” que se comenta a continuación.

3.3.2.1 condor_compile

Las tareas que se someten en el Universo Standard necesitan ser recompiladas mediante las librerías de Condor. Para recompilar los programas con Condor, este está provisto de una herramienta especial, “condor_compile”. Esta herramienta viene por defecto en el software de Condor, y trabaja con los siguientes comandos, dependiendo del fichero a compilar [37]:

Compilador	Descripción
gcc	Compilador de C GNU
ld	Sistema enlazador
g77	Compilador de Fortran GNU
acc	Compilador de C ANSI en sistemas Sun
g++	Compilador de C++ GNU
f77	Compilador de sistemas Fortran
f90	Compilador de Fortran 90
cc	Compilador de C++
CC	Compilador de c

Tabla 3. 1 Compiladores soportados en Condor

En plataformas como Solaris o Digital Unix también se soporta f90. En cualquier caso, se puede hacer condor_compile y trabajar transparentemente con todos los comandos en el sistema, incluyendo el comando make.

El objetivo de este comando es el de sustituir el sistema enlazador (ld) de Linux por el sistema enlazador de librerías de Condor. Para poder hacer esto, se debe hacer una instalación completa de condor_compile, siguiendo los siguientes pasos:

- Renombrar el sistema enlazador cambiando ld por ld.real.
- Copiar el enlazador de Condor a la localización del sistema enlazador antes mencionado (ld).
- Establecer como propietario del enlazador a Root.
- Establecer los permisos de este nuevo compilador a 755.

Las localizaciones de los sistemas enlazadores (ld) dependen del sistema operativo de la máquina Submit en la cual se encuentra la tarea a recompilar. Las localizaciones en cuanto a los sistemas operativos con los que Condor puede trabajar se muestran a continuación:

Sistema Operativo	Localización de ld
Linux	/usr/bin
Solaris 2.X	/usr/ccs/bin
OSF/1 (Digital Unix)	/usr/lib/cmplrs/cc

Tabla 3. 2 Localización de ld en S.O

Para diferenciar entre las construcciones normales y las construcciones Condor, el usuario simplemente sitúa condor_compile antes del comando de construcción a la hora de compilar. Este comando establece el entorno apropiado que permite al script compilador de Condor conocer lo necesario para hacerlo.

Siguiendo con el ejemplo anterior, una vez que se tiene el objeto del programa "tareacondor.o", para poder someter esta tarea en la aplicación Condor, se debe de recompilar la tarea mediante condor_compile, usando el compilador del programa y el objeto. De esta manera se crea el ejecutable adecuado para ser sometido en la aplicación Condor.

linux:~ # condor_compile gcc tareacondor.o –o tareacondor

Una vez hecho esto, se debe tener un ejecutable llamado "tareacondor". Si se escribe en la línea de comandos:

linux:~ # ./tareacondor

No dará ningún error en caso de que se ejecute fuera del entorno de Condor. Esto se suele hacer para comprobar que el ejecutable funciona correctamente haciendo lo que se desea.

Una vez que se tiene el ejecutable enlazado con las librerías Condor, el siguiente paso es crear el fichero de descripción submit (que se describe en el siguiente apartado) para poder someter las tareas, ya que, como se explicó en el capítulo anterior, para poder someter tareas en una máquina Submit, es necesario llamar al fichero de descripción submit mediante el comando condor_submit.

3.3.3 Fichero de descripción submit

El fichero de descripción submit es un script necesario para poder someter tareas en un Condor Pool, independientemente del entorno de ejecución de dicho Pool.

Un fichero de descripción submit de una tarea describe un cluster [12] de tareas que puede ser situado en el Condor Pool. Todas las tareas en un cluster deben distribuirse en el mismo ejecutable, pero pueden tener diferentes ficheros de entrada y salida, y diferentes argumentos en el programa, etc…como se verá a continuación.

El fichero de descripción submit debe contener diferentes parámetros para poder someter la tarea. Se necesita que haya al menos un parámetro “executable” que indique el nombre del ejecutable que se desea someter y al menos un comando “queue”, para poder encolar la tarea en una cola de tareas. Todos los demás comandos son opcionales, y se introducirán en el fichero dependiendo de las necesidades de cada usuario para la ejecución de su tarea. En el caso que se está tratando, el universo Standard, se debe indicar que el universo en el que se trabaja es ese, mediante el parámetro Universe.

Por tanto, una primera estructura que se podría tener de este fichero es:

Executable = nombre del fichero ejecutable

Universe = Standard

Arguments = lista de argumentos suministrados al programa.

Initialdir= directorio inicial donde se encuentra la tarea a ejecutar

Input = fichero de entrada de teclado al programa.

Output= fichero que captura la información que sale por pantalla

Error= fichero que captura los mensajes de error

……

Queue

Esta sería la estructura de un fichero submit para el universo Standard. Hay un gran número de parámetros que se pueden añadir a este fichero (ver apéndice) [39].

Un ejemplo real de un Fichero de descripción submit (llamado “hello.submit”) para un programa puede ser el siguiente [40]:

##############;###

 # Ejemplo de fichero descripción submit para entorno standard.

##################

Universe = standard

Executable = hello

output = hello_world.out

error = hello_world.error

log = hello_world.log

Queue

Los datos que indica este fichero son que el ejecutable es el programa “hello”. El resultado del ejecutable se guardará en el fichero indicado por el parámetro “output”. En caso de error en la ejecución, se pintará en el fichero indicado en “error”. El fichero indicado en “log” guardará todos los pasos dados por la tarea desde que se sometió hasta que terminó su ejecución.

Importante:

- Para poder ejecutar tareas hay que dar permisos a todos los directorios del path donde se encuentran las tareas para que cualquier usuario pueda ejecutarlas (chmod 755).
- Para utilizar los comandos "condor_compile" para recompilar y "condor_submit" (siguiente apartado) para ejecutar tareas, se debe trabajar en el directorio donde se encuentren las tareas que se van a manejar. Si no se puede trabajar en dicho directorio, se puede indicar mediante parámetros en el fichero de descripción submit (ver apéndice).

3.3.4 condor_submit

El programa que somete tareas en Condor es el programa condor_submit. Como se comentó en el capítulo 2 este programa requiere un fichero de descripción submit, el cual contiene los comandos para dirigir la tarea a la cola de tareas [37].

Un fichero de descripción puede contener diferentes especificaciones para encolar las tareas. Todas las tareas encoladas por una única invocación de condor_submit deben distribuir el mismo ejecutable, refiriéndose al cluster de la tarea.

Es ventajoso el someter múltiples tareas como un único cluster ya que:

- Solo se necesita una copia del fichero checkpoint para representar todas las tareas en un cluster hasta que comience la ejecución de las mismas.
- Es mucho menos sobrecargado el concebir Condor para comenzar la siguiente tarea en un cluster, que para Condor comenzar en un nuevo cluster. Esto puede provocar una gran diferencia si se están sometiendo muchas tareas y de poco tamaño.

En el universo Standard, el uso de este comando es obligado para someter tareas, y usará como argumento en la línea de comandos el fichero de descripción submit. Si se coge como fichero de descripción submit, el ejemplo real antes descrito, para someter la tarea que en el fichero se indica, se debe de escribir en la línea de comandos lo siguiente:

linux:~ # (path de la tarea)/condor_submit hello.submit

Una vez ejecutado en la línea de comandos, se puede observar en la cola de tareas el estado de la tarea mediante el comando "condor_q" (capítulo 4). Se pueden mirar también los ficheros de salida para ver que se han ejecutado. Además, se puede visualizar el fichero ".log" para ver que ha hecho Condor con nuestra tarea.

Importante:

- Por razones de seguridad, Condor no permite la ejecución de tareas como usuario root (UID= 0) o como un usuario privilegiado. El usuario root o usuarios privilegiados aparecerán establecidos por siempre en la cola en el estado "state" (capítulo 2).
- Todos los nombres de rutas especificados en el fichero de descripción submit deben ser menos de 256 caracteres de longitud, y los

argumentos en la línea de comandos deben ser menos de 4096 caracteres de longitud. De todas maneras, condor_submit dará un mensaje de “warning” pero no se ejecutara correctamente.

- Para inutilizar el Checkpointing en las tareas del entorno Standard, se debe de incluir en el fichero de descripción submit el siguiente comando antes del comando queue: WantCheckpoint = false (ver apéndice).

Capítulo 4

Universo Standard en una red. Instalación y configuración

4.1 Situación de la red prototipo

La red prototipo que se propone en este estudio está formada por tres máquinas (figura 4.4.1). Esta red se ha realizado con vistas a una futura instalación de Condor en el laboratorio IT-3 del Área de Ingeniería Telemática de la UPCT. De esta manera se podrán estudiar las dificultades que surgen a la hora de instalar y configurar Condor, y se podrá verificar el funcionamiento y los resultados obtenidos con la dicha aplicación.

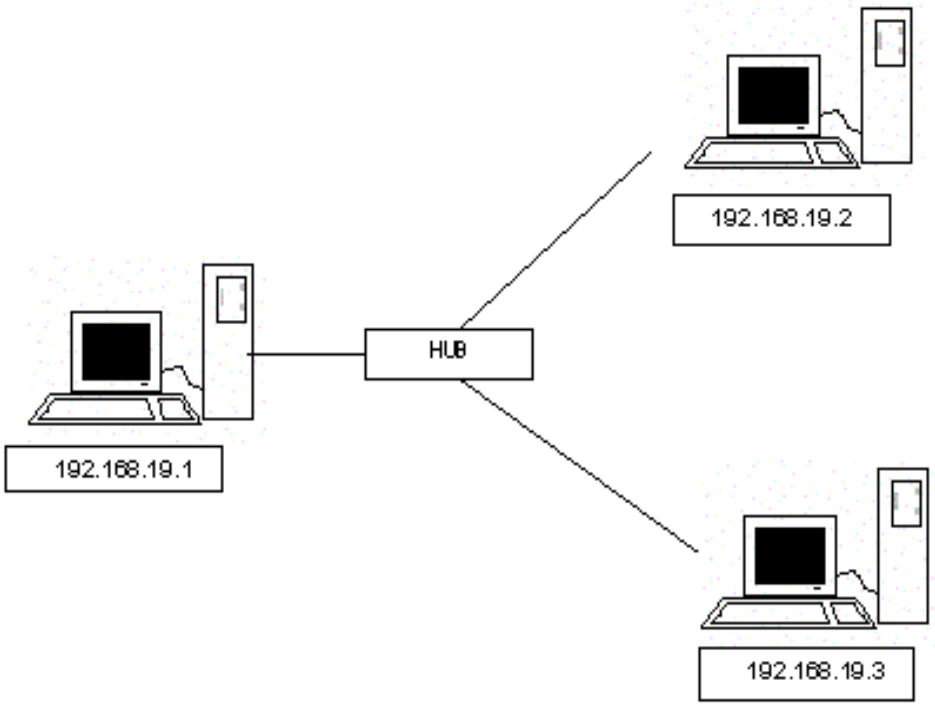

Figura 4. 1 Situación de la red

A continuación se describen las características técnicas de cada una de las máquinas que van a formar parte de la red prototipo. Estas características se tendrán en cuenta para decidir el papel que va a desempeñar cada máquina en el Condor Pool, como se verá en los siguientes apartados.

Procesador	Pentium III, 450 MHz
Disco duro	10 Gb
RAM	128 Mb
Tarjeta de red	3Com Ethernet 10/100
S.O	Suse Linux 7.2 (i386)
S.A.I	Si

Tabla 4. 1 Máquina 1

Procesador	Pentium III, 450 MHz
Disco duro	10 Gb
RAM	128 Mb
Tarjeta de red	3Com Ethernet 10/100
S.O	Suse Linux 7.2 (i386)
S.A.I	No

Tabla 4. 2 Máquina 2

Procesador	Pentium III, 450 MHz
Disco duro	10 Gb
RAM	128 Mb
Tarjeta de red	3Com Ethernet 10/100
S.O	Suse Linux 7.2 (i386)
S.A.I	No

Tabla 4. 3 Máquina 3

Se decide trabajar en todas las máquinas bajo el sistema operativo Unix/Linux ya que este permite una gestión más cómoda en cuanto a los simuladores que se ejecutarán en un futuro.

La instalación de Condor bajo Linux incrementa la dificultad tanto de la instalación y configuración como de la puesta en funcionamiento de la aplicación en las máquinas, ya que la instalación de Condor en máquinas que trabajan bajo Windows es mucho más sencilla, pero a la vez, tiene bastantes restricciones, como por ejemplo, que solo puede trabajar bajo el entorno vanilla, con lo cual, al no poder migrar las tareas hacia otras máquinas, la opción del sistema operativo Windows queda descartada (ver apéndice).

Antes de realizar cualquier instalación lo más conveniente es descargar el software y leer los pasos previos a la instalación. De esta manera se podrá asignar a cada máquina el rol más conveniente.

4.2 Condor Download

El primer paso para instalar un Condor Pool en una red es descargar el software de la aplicación que se encuentra en la página web oficial de Condor http://www.cs.wisc.edu/condor. Los paquetes a descargar están disponibles en la página web http://www.cs.wisc.edu/condor/downloads/.

Una vez dentro de la zona de descarga, el software de Condor se puede obtener de dos zonas diferentes, siendo la zona principal la Universidad de Wisconsin Madison, Madison, Wisconsin, USA. La segunda zona es el Instituto Nacional de Física Nuclear de Boloña, Boloña, Italia. Se debe elegir la zona más cercana a donde se encuentre la red sobre la cual se va a trabajar con la aplicación. La descarga del software dependerá también de la plataforma en la que se quiera instalar Condor. Así que para elegir el módulo a descargar se tendrán en cuenta estos dos factores.

Se encontrarán diferentes versiones de Condor, siendo en estos momentos la versión más actualizada Condor 6.4.3 [41]. Se ha elegido instalar la versión 6.2.1, ya que para el universo en el que se va a trabajar, la versión escogida es muy estable, además de que las siguientes versiones no modifican el funcionamiento del Universo Standard. En cuanto a la elección de la plataforma se ha escogido LINUX y la zona de descarga de Boloña, Italia, ya que es la más cercana a España.

Una vez descargado el software de Condor y antes de instalar la aplicación, es preferible pertenecer a la lista de correo de "condor-world". Esta lista solo se utiliza para que la Institución de Condor pueda llevar un control sobre el manejo de Condor en el mundo. Para la suscripción, se debe mandar un e-mail a "mayordomo@cs.wisc.edu" con el siguiente mensaje:

suscribe condor-world

Una vez enviado el e-mail, el administrador de Condor responderá con un mensaje de bienvenida y algunas referencias en cuanto a la lista de Condor, en caso de querer borrarse de ella, por si se tienen dudas, o por si se quiere contactar con algún miembro de la lista.

Una vez descargado el software, hay que descomprimirlo, para ello, hay que situarse en el path en el cual se ha descargado el software y descomprimirlo:

maquina1:~ /software/inscondor # tar –zxvf condor-6.2.1-linux-x86-glibc21.tar.gz

Al descomprimir el software, aparece un nuevo directorio llamado condor-6.2.1. Dentro de este directorio se encuentra la distribución de binarios descargados de Condor, empaquetado en los siguientes 5 ficheros y 2 directorios.

maquina1:~ /software/inscondor # cd condor-6.2.1

DOC INSTALL LICENSE.TXT README condor_install examples release.tar

La descripción de estos ficheros y directorios es la siguiente:

- DOC Direcciones donde se puede encontrar documentación Condor.
- INSTALL Direcciones de instalación.
- LICENSE.TXT La licencia de Condor.
- README Información general de Condor.

- "condor_install" El script Perl usado para instalar y configurar Condor.
- examples/ Directorio que contiene ejemplos de programas C, Fortran y C++, para ejecutar en Condor.
- "release.tar" Fichero tar que contiene los binarios y las librerías de Condor.

Se debe descomprimir el fichero release.tar. Así que hay que situarse dentro del directorio en el que se encuentra "condor-6.2.1" y descomprimir el fichero para poder realizar la instalación.

maquina1:~ /software/inscondor/condor-6.2.1 # tar –xvf release.tar

Una vez descomprimido, dentro del directorio condor-6.2.1 aparecerán nuevos directorios:

maquina1:~ /software/inscondor/condor-6.2.1 # ls

bin etc include sbin lib *release.tar LICENSE.TXT INSTALL DOC README condor_install examples*

A partir de este momento ya se tiene descargado el software de Condor. Antes de realizar ninguna instalación se debe leer atentamente el siguiente apartado donde se da a conocer de una forma más detallada el software de Condor y los aspectos a tener en cuenta sobre la estructura del mismo a la hora de instalar Condor.

4.3 Aspectos a tener en cuenta para instalar Condor

Una vez que se tiene descargado el software, antes de realizar cualquier tipo de instalación en la red para configurar un Condor Pool, hay que tomar importantes decisiones en cuanto al tipo de instalación que se quiere realizar en las diferentes máquinas, las configuraciones que se deben hacer…Esas decisiones son las que se plantean en los siguientes apartados [42].

4.3.1 Máquina Central Manager

Una de las máquinas que formen parte del Condor Pool deberá trabajar como Central Manager ya que este rol es imprescindible en un Pool. Esta es la primera máquina donde se instalará el software de Condor. Como se comentó en otros capítulos, el Central Manager es el repositorio de información centralizada del Condor Pool, así como también la máquina encargada de las negociaciones en el Pool a la hora de someter tareas entre las máquinas disponibles para ejecutar tareas, Execute Machine, y las máquinas que someten tareas, Submit Machine (capítulo2).

Si la máquina Central Manager falla, las tareas que se encuentran en el Pool se mantendrán ejecutándose en el sistema, pero no se podrán ejecutar las nuevas tareas que se generen en el Pool. Además, muchas herramientas de las que dispone Condor para conocer el estado del Pool, para conocer las tareas que se encuentran ejecutándose, etc… dejarán de tener utilidad.

Como esta instalación es muy importante para el perfecto funcionamiento del Condor Pool, se debe elegir como Central Manager una máquina que esté siempre en línea, o una que se reinicie rápidamente en caso de fallo.

También hay que considerar el tráfico de la red y la disposición de la máquina en la misma cuando se elija esta para funcionar como Central Manager. Todos los procesos (demonios) que se ejecutan en el Pool envían actualizaciones (por defecto cada cinco minutos) a esta máquina, por tanto, dicha máquina debe ser capaz de enviar y recibir una gran cantidad de tráfico de la red.

4.3.2 Condor como usuario privilegiado Root

Al iniciar Condor en un Pool, los demonios se deben ejecutar como usuario privilegiado Root en Unix. Esto es así, porque si no, Condor puede hacer muy poco para fortalecer la seguridad y las decisiones políticas del Pool. Se puede ejecutar Condor como un usuario no privilegiado cualquiera, pero surgirán problemas de seguridad y consecuencias negativas en el rendimiento de la aplicación en la red, sobre todo si la máquina donde se realiza es el Central Manager.

Lo que a continuación se detalla son los efectos que se producen en los principales procesos de Condor al no ejecutar la aplicación en las máquinas como usuario privilegiado Root.

condor_startd

Recordando un poco el funcionamiento de este proceso (capítulo 2), si se establece una máquina para ejecutar tareas Condor (Execute Machine) y no se ejecuta el proceso “condor_startd” como Root, se esta confiando básicamente en la buena voluntad de los usuarios que forman parte del Condor Pool.

Si se ejecuta Condor como usuario Root, se pueden reforzar esas políticas sin importar el comportamiento de los usuarios maliciosos. Aquí se muestra la diferencia entre trabajar como Root o como otro usuario diferente.

1- Trabajando como Root, los procesos de Condor se ejecutan con una UID diferente al de la tarea Condor (las tareas del usuario se inician con la UID del usuario que la sometió o del usuario “nobody”, dependiendo de lo establecido en el dominio UID [43]), con lo cual, se asegura que el acceso a la tarea de Condor será limitado.

2- Si se ejecuta Condor como usuario diferente a Root, todos los procesos que ejecute Condor, incluyendo las tareas de los usuarios, tendrán la misma UID (puesto que no se puede cambiar la UID a no ser que se sea usuario privilegiado Root).

Existen otras restricciones al ejecutar condor_startd como un usuario no privilegiado, como por ejemplo a la hora de obtener información sobre el Condor Pool. En muchos sistemas (por ejemplo IRIX), un usuario cualquiera no puede obtener información del Condor Pool si no se tiene acceso Root, pero esto no ocurre en Unix, que es la plataforma en la que se va a trabajar en este proyecto. De todas formas, para asegurarse de que la política que se elige fortalece de forma efectiva el Condor Pool, condor_startd debe ejecutarse como Root.

condor_schedd

El mayor problema que se puede presentar cuando se está ejecutando el proceso condor_schedd sin acceso Root es que los procesos “condor_shadow”, que son lanzados por el “condor_schedd” (capítulo 2), se les otorga la misma UID que al proceso “condor_schedd”. Cuando las máquinas Submit someten tareas, el resultado de estas lleva consigo el tener que escribir o crear ficheros o directorios en la máquina Submit, por tanto los usuarios de las máquinas Submit deberán garantizar de alguna

forma el acceso de escritura por parte del usuario o grupo condor (o quienquiera que ejecute el schedd).

Si se recuerda del capítulo 2, al ejecutarse el proceso condor_submit en la máquina Submit para someter una tarea, era este el que lanzaba al proceso condor_schedd. Por tanto, hay que tener en cuenta los permisos del "condor_submit".

Por último, Si los usuarios someten tareas que son programas que abren sus propios ficheros, tendrán que saber establecer los permisos correctos en dichos ficheros.

condor_master

El condor_master es el proceso que inicia a los procesos condor_startd y condor_schedd, así que si se quiere que ambos se estén ejecutando como Root, se debe tener el master ejecutándose como Root también.

condor_negotiator

No tiene ninguna restricción.

condor_collector

No es necesario tener o no estos procesos ejecutándose como Root.

condor_kbdd

En plataformas que necesitan el proceso condor_kbdd (Digital Unix e IRIX), este proceso se debe ejecutar como Root. Si se ejecuta como otro usuario, no realizará su función (capítulo 2). Sin el proceso condor_kbdd ejecutándose, el condor_startd no tendrá manera alguna de monitorear la actividad del ratón todo el tiempo, y solo la actividad del teclado será notificada.

4.3.3 Administración de Condor

Cualquier máquina en el Pool puede ser directamente el administrador Condor trabajando como usuario privilegiado Root, aunque cualquier otro usuario también puede realizar esta función. Si Root delega la responsabilidad de la administración en otra persona, pero no quiere conceder a esa otra persona el acceso Root, el usuario privilegiado Root puede especificar un fichero llamado "condor.config.root" (ver apéndice). Con este fichero se logra que el fichero global de configuración del Condor Pool "condor_config" (que se verá en los siguientes apartados) pueda ser controlado y pertenecer a cualquiera que sea administrador de Condor y el fichero "condor.config.root" puede ser controlado y pertenecer solamente a Root. De esta manera se permiten establecer algunos controles de políticas de trabajo u otros específicos del fichero "condor_config" sin acceso Root.

4.3.4 Usuario Condor y directorio home distribuido

Para simplificar la instalación de Condor, se debe crear un usuario Condor en todas las máquinas donde se vaya a instalar el software de Condor ya que en la instalación del software se preguntará por un usuario Condor y en caso de no existir, Condor no se podrá instalar.

Los demonios de la aplicación Condor crearán ficheros que pertenecerán al usuario Condor, y el directorio home será el utilizado para especificar la localización de los ficheros y directorios necesarios para Condor.

El directorio home de este usuario podrá estar compartido a través de la red a todas las máquinas que formen parte del Condor Pool, si se tiene un sistema de ficheros distribuido o podrá estar separado del directorio home en una partición local

en cada máquina, dependiendo de la instalación que se realice en el Pool que se verá más adelante. El situar los ficheros en un lugar u otro tiene ventajas y desventajas.

Tener los directorios centralizados (como el caso del sistema de ficheros distribuido) puede hacer una administración fácil, pero también concentra todos los recursos que se usan en un Condor Pool con lo cual, potencialmente se necesita mucho espacio para un único directorio home/. Si no se elige crear un usuario llamado Condor, entonces se debe especificar la vía del entorno variable CONDOR_IDS (ver apéndice).

4.3.5 Directorios de la máquina específica Condor

Condor utiliza unos directorios específicos para trabajar. Estos directorios son únicos de cada máquina en el Condor Pool. Los directorios son spool/, log/ y execute/.

Generalmente todos estos directorios son los subdirectorios que hay en el directorio de cada máquina específica del Pool. Este directorio es el directorio local de la máquina y el nombre de dicho directorio coincide con el nombre de la máquina (especificado en "LOCAL_DIR" del fichero de configuración. Ver apéndice).

Si se tiene un usuario Condor en Unix con un directorio local home/ en cada máquina, el LOCAL_DIR será el directorio home/ del usuario Condor. Si el directorio home/ de este usuario se distribuye en todas las máquinas del Condor Pool a través de un sistema de ficheros distribuido, se creará un directorio para cada máquina en el directorio local.

En la siguiente figura se muestra el aspecto que tendría un directorio home distribuido con un usuario Condor, en el que cuelgan los directorios locales de las máquinas que forman parte de un Condor Pool. Para este ejemplo se ha supuesto que el Condor Pool lo forman tres máquinas: maquina1, maquina2, maquina3. La máquina1 es la máquina que funciona como sistema de ficheros distribuidos, mientras que la máquina2 y la máquina3 son clientes de este sistema de ficheros distribuido.

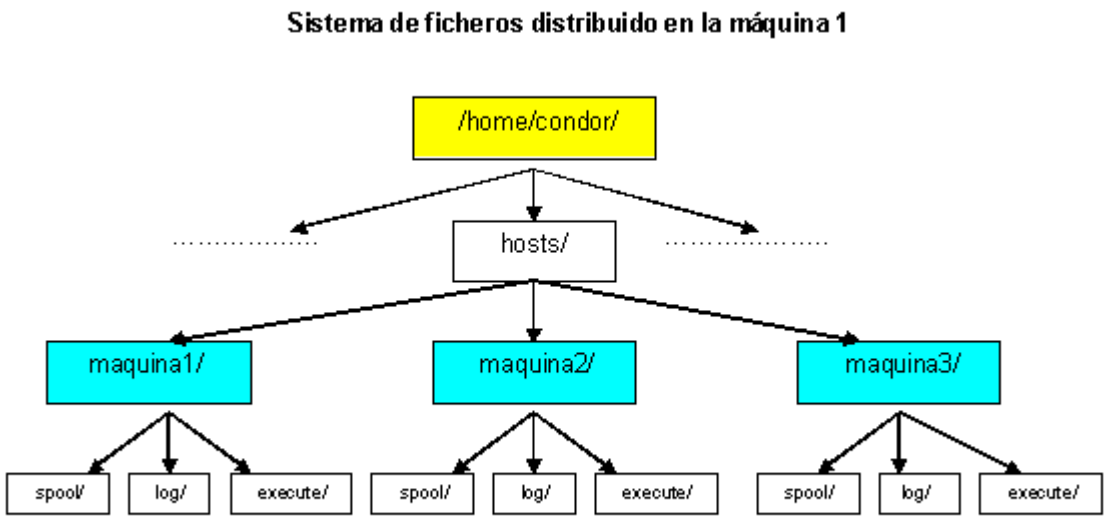

Figura 4. 2 Directorio home distribuido

En caso de no tener un sistema de ficheros distribuido, es decir, en caso de no tener los directorios de Condor colgando del directorio home/, cada máquina podrá tener los ficheros de condor en rutas diferentes.

4.3.5.1 Directorio execute/

Este directorio actúa como el directorio de trabajo general para cada tarea Condor que se ejecuta en una máquina Execute dada. Los binarios de las tareas que somete una máquina Submit se copian en este directorio. Es por ello que debe haber bastante espacio de disco (Condor no permitirá el envío por parte de una máquina Submit de una tarea a una máquina Execute que no tenga bastante espacio de disco para soportarla).

Si la tarea que se encuentra ejecutándose en la máquina Execute debe enviar algún fichero a la máquina Submit que sometió la tarea, dicho fichero primero se verterá en el directorio execute/ de la máquina Execute antes de enviarlo a la máquina Submit. Así que se debe poner el directorio execute/ en una partición con bastante espacio para aguantar todos los ficheros posibles de las tareas del Condor Pool que se ejecuten en la máquina.

4.3.5.2 Directorio spool/

El directorio spool/ es el que soporta la cola de tareas y el fichero que guarda el historial de la máquina en el Condor Pool, además de los ficheros checkpoint de las tareas de la máquina (estos últimos dependerán del Rol que juegue la máquina). Como resultado, el espacio de disco requerido para el directorio spool/ puede ser bastante grande, particularmente si los usuarios someten tareas cuyos ejecutables necesitan un espacio de disco bastante grande.

Usando un Servidor Checkpoint (capítulo 3), se puede disminuir el espacio de disco requerido, ya que todos los ficheros checkpoint son almacenados en el servidor en vez de en el directorio spool/ de la máquina.

De todas maneras, el fichero checkpoint inicial (los ejecutables para todos los cluster que se someten) están almacenados en el directorio spool/, así que se necesitará más espacio aunque se use un Servidor Checkpoint.

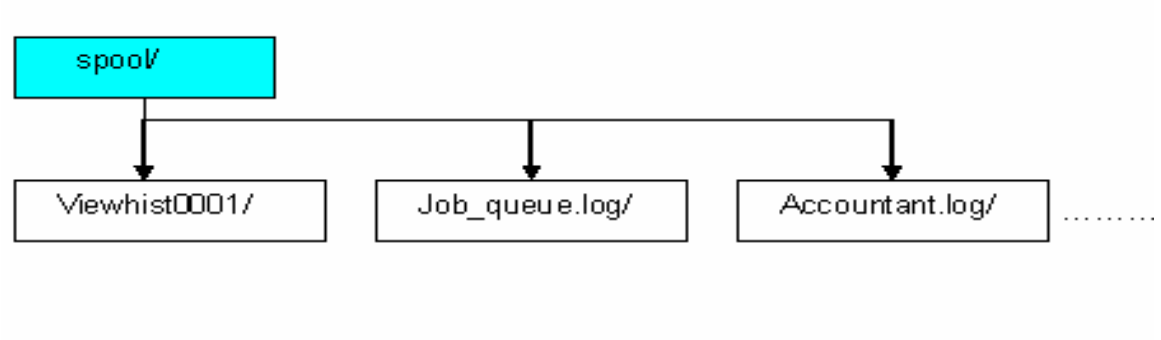

Figura 4. 3 Directorio spool/ de una máquina

4.3.5.3 Directorio log/

Cada una de las máquinas que forman parte de un Condor Pool posee un directorio log. Dentro de este se almacenan los ficheros log de los procesos que se ejecutan en la máquina. En caso de que haya poco espacio de disco en la máquina, no se guardarán ni los ficheros log ni la información de Condor.

En caso de tener establecido un sistema de ficheros distribuido en el Condor Pool, este directorio necesariamente debe colgar de la localización distribuida. De esta

manera se podrán visualizar todos los ficheros log de todas las máquinas que formen parte del Pool a través de dicho sistema de ficheros.

La siguiente figura muestra un fichero log de una máquina funcionando como Central Manager. Los procesos que se ejecutan en ella, es decir, condor_master, condor_collector y condor_negotiator escriben un fichero log al iniciarse.

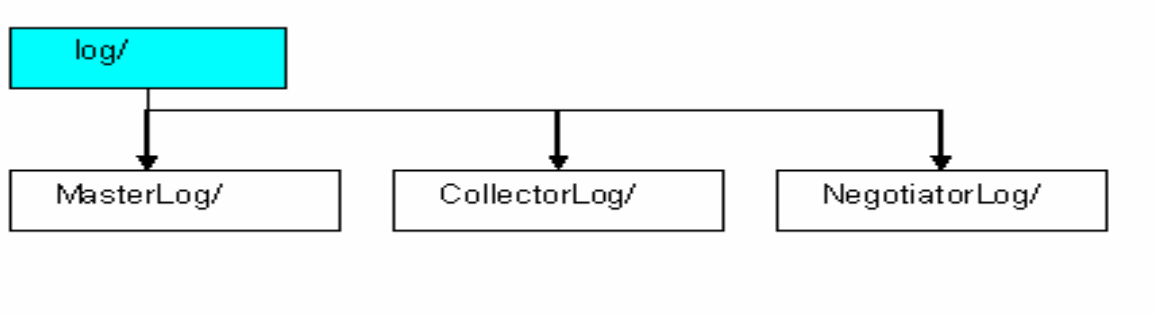

Figura 4. 4 Directorio log/ de una máquina Central Manager

En el caso de ser una máquina Submit-Execute, los ficheros que se crearán son condor_master, condor_schedd y condor_startd, como se muestra en la siguiente figura.

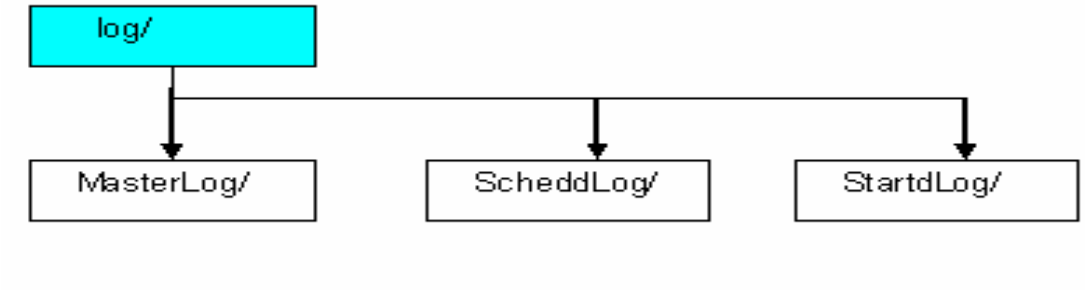

Figura 4. 5 Directorio log de una máquina Submit Execute

4.3.5.4 Directorio lock/

Condor usa ficheros lock/ para sincronizar los accesos para ciertos ficheros que están distribuidos entre múltiples procesos Condor. A pesar de los problemas de sincronismo encontrados con los ficheros “locking” y en los sistemas de ficheros de red de trabajo (particularmente NFS), estos ficheros deberán estar situados en una partición local en cada máquina que, por defecto, se situarán en el path /var/lock/condor. El parámetro LOCK del fichero de configuración es el que se establece con el path de este directorio (ver apéndice).

Si se instala Condor en una máquina, a la hora de instalar el directorio lock, se recomienda que este no se encuentre dentro de la partición de var/ ya que si esta partición se llena causará muchos problemas en la máquina. Idealmente se deben tener separadas las particiones de los directorios Condor. La única consecuencia de llenar los directorios será el mal funcionamiento de Condor, pero no de la máquina.

4.3.6 Partes del sistema de Condor

Las diferentes partes del sistema Condor, software, rutas donde deben instalarse los ficheros y directorios de Condor que se deben conocer, se comentan a continuación:

4.3.6.1 Ficheros de configuración

En Condor aparecen unos ficheros de configuración que mantienen diferentes niveles del control sobre la configuración de la Aplicación Condor en cada máquina que forma parte del Condor Pool. Se pueden distinguir dos tipos de ficheros de configuración:

- Ficheros de configuración local
- Ficheros de configuración global

El fichero de configuración global se distribuye por todas las maquinas que forman parte del Condor Pool. En caso de utilizar un sistema de ficheros distribuido, para facilidad de administración, este fichero debe estar localizado en este sistema de ficheros. De esta manera, a través del sistema distribuido, todas las máquinas que formen parte del Pool, tendrán acceso al mismo fichero de configuración global.

El fichero de configuración local es único en cada una de las máquinas del Condor Pool. Lo que se establezca en este fichero puede sobrescribir al fichero global. De esta manera se permite que en cada una de las máquinas del Condor Pool se tengan diferentes procesos (demonios) ejecutándose, diferentes políticas para cuando se comienza y paran las tareas Condor, etc… teniendo en cada máquina ficheros específicos de configuración.

Se recomienda, como se comentó en el apartado anterior, que se ejecuten los demonios Condor como Root, ya que así los ficheros de configuración que se creen en las máquinas serán propios y estarán controlados por el usuario privilegiado Root.

Si el administrador de Condor no es Root, los ficheros de configuración de Condor podrán ser manipulados por el administrador de Condor. En este caso, Root configura el Pool de manera que ese usuario que trabaja como administrador no puede tener acceso a Root, y por tanto, no podrá modificar los ficheros de configuración de Root.

Cuando se ejecute la aplicación Condor, esta buscará los ficheros de configuración para saber que debe realizar en cuanto a lo configurado. El primer fichero que buscará será el fichero de configuración global. Condor sabe que dicho fichero se puede encontrar en diferentes localizaciones y busca el fichero en un orden de localización establecido. Las rutas donde Condor buscará el fichero de configuración global son:

1- fichero especificado en CONDOR_CONFIG variable de entorno (ver apéndice).

2- /etc/condor/condor_config

3- ~condor/condor_config

En caso de que la máquina no encuentre el fichero de configuración en ninguna de las localizaciones, se mostrará un mensaje de error y la aplicación finalizará.

Después de encontrar el fichero de configuración global, Condor intenta cargar los ficheros de configuración local. La única manera de especificar el fichero de configuración local es en el fichero de configuración global, con el macro LOCAL_CONFIG_FILE (ver apéndice). Si este macro no se establece, el fichero de configuración local no se va a usar. Este macro puede estar en una lista de ficheros o en un único fichero.

Por otro lado el fichero de configuración local Root (condor.config.root) se encuentra en el macro LOCAL_ROOT_CONFIGFILE (ver apéndice). Si este no se establece, el fichero de configuración root no se usa. Este macro puede estar en una lista de ficheros o en un único fichero.

Los ficheros de configuración de root se buscarán en las siguientes rutas:

maquina1:~ # /etc/condor/condor_config.root

maquina1:~ # condor/condor_config.root

4.3.6.2 Directorio Release

En cada máquina que forma parte del Condor Pool existe una distribución de binarios que contiene un fichero "release.tar". Este fichero contiene 4 subdirectorios destacados: /bin, /etc, /lib, /sbin. Donde quiera que se elija instalar estos directorios, el directorio que los contenga se llamará "directorio release".

Este fichero se especifica en el macro RELEASE DIR del fichero de configuración de la máquina. (Ver apéndice) Cada "directorio release" contiene una plataforma dependiente de binarios y librerías, así que se necesitará instalar por separado para cada una de las máquinas que vayan a formar parte de un Condor Pool.

En caso de que en el Condor Pool haya un sistema de ficheros distribuido, para facilidad de administración, este directorio debería estar localizado, si es posible, en dicho sistema de ficheros. A continuación se muestra una figura con la estructura del directorio release.

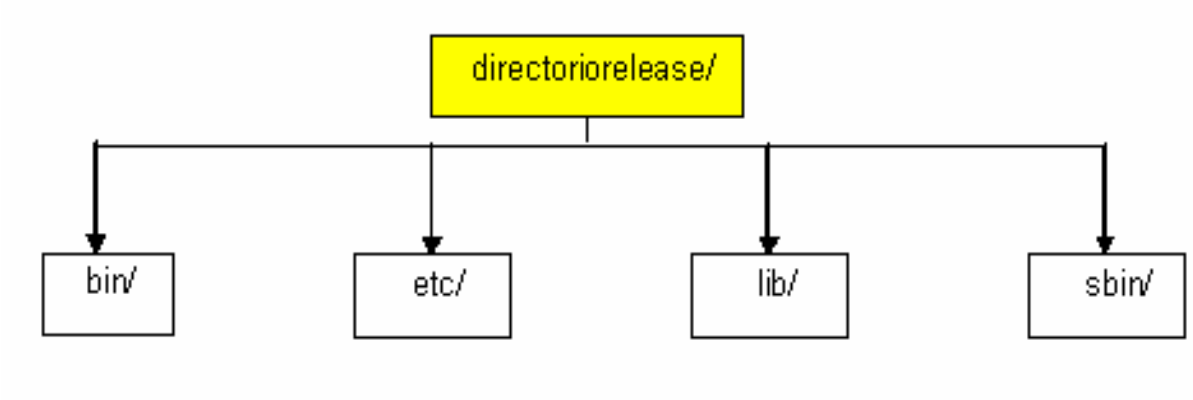

Figura 4. 6 Directorio Release

Binarios de usuario, bin/

Todos los ficheros en el directorio bin/ son programas que al final de la instalación el usuario Condor deberá esperar tener en su ruta. Dichos programas son

comandos que el usuario de la máquina utilizará para su manejo de Condor, obteniendo información del Pool, de la cola de tareas, borrar tareas de una cola, etc…

A la hora de instalar el directorio, este deberá situarse en una localización bien conocida, como por ejemplo /home/condor/bin, en caso de tener un sistema de ficheros distribuido. También se pueden copiar estos ficheros directamente en un lugar conocido en el PATH del usuario (usr/local/bin), en caso de que cada máquina instale Condor de forma local.

La siguiente figura muestra algunos de los scripts que se encuentran dentro del directorio bin/.

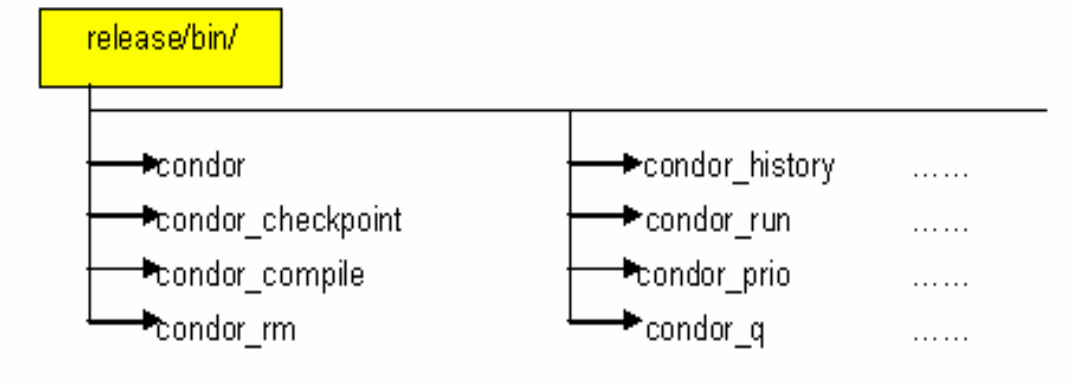

Figura 4. 7 Subdirectorio bin/ de Release

Binarios del sistema, sbin/

Todos los ficheros del directorio sbin/ son procesos Condor y agentes, o programas que solo el administrador de Condor utiliza para ejecutar los diferentes procesos, como por ejemplo los procesos condor_collector y condor_negotiator del Central Manager.

A continuación se muestra una figura con la estructura de este subdirectorio y los programas que contiene.

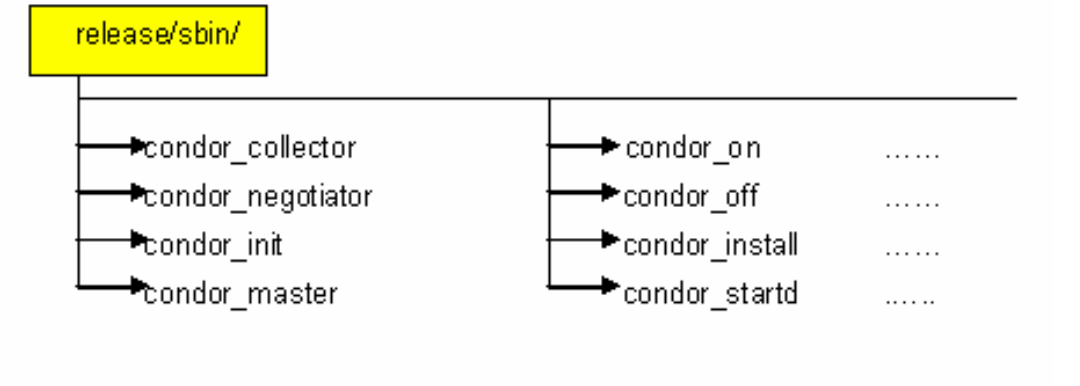

Figura 4. 8 Subdirectorio sbin/ de Release

Directorio lib/

En este directorio se encuentran todos los ficheros de las librerías Condor. Estos ficheros son los utilizados para enlazar los programas con las librerías de Condor para poder someter las tareas, realizar el checkpoint, migrar…

Este directorio contiene también diferentes scripts que son usados por el programa condor_compile para ayudar a recompilar tareas con las librerías Condor. El script condor_compile chequea el fichero de configuración de la máquina en la que se ha ejecutado el comando para la localización del directorio lib.

La siguiente figura muestra la estructura de este subdirectorio con los distintas librerías que contiene.

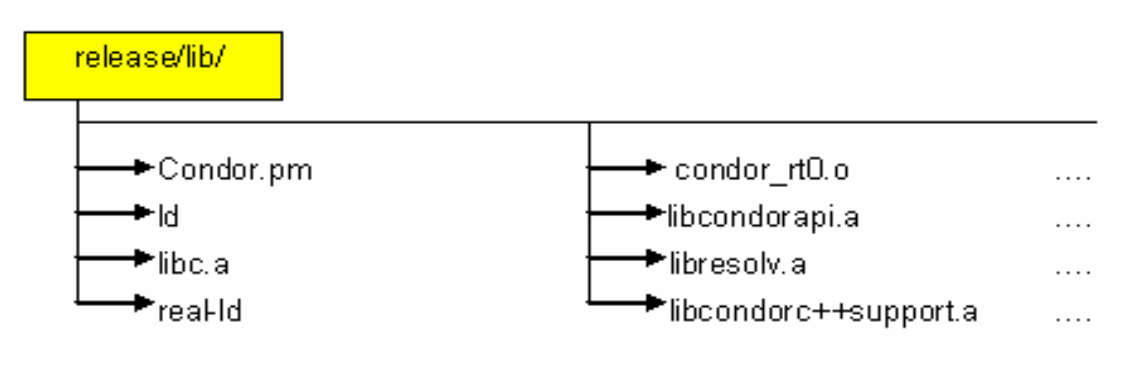

Figura 4. 9 Subdirectorio lib/ de Release

Directorio etc/

El directorio etc/ contiene un subdirectorio “examples”, del cual cuelgan varios ficheros de configuración “example” y otros ficheros usados para la instalación de Condor. El directorio etc/ es la localización recomendada para mantener la copia del fichero de configuración global del Condor Pool. Este fichero se puede poner en diferentes enlaces, como se ha mencionado, ya que Condor realiza un chequeo de forma automáticamente para encontrar el fichero de configuración global.

La siguiente figura muestra la estructura del subdirectorio etc y los distintos ficheros que contiene.

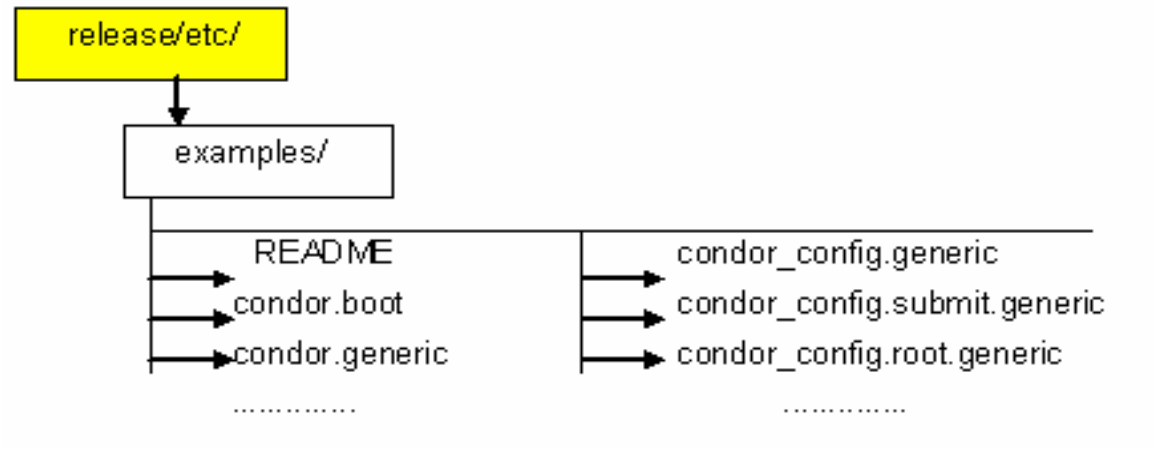

Figura 4. 10 Subdirectorio etc/ de Release

4.3.6.3 Documentación

La documentación de Condor está actualmente disponible en formato HTML, Postscript y PDF (Adobe Acrobat). Se puede encontrar en la web oficial de Condor http://www.cs.wisc.edu/condor/manual.

4.3.7 Usando AFS

Si se está usando AFS como sistema de ficheros distribuido en la red en la que se quiere establecer un Condor Pool, la aplicación Condor no tiene forma alguna de autentificar AFS. Se recomienda en este caso usar NFS, lo que conlleva también a la instalación de NIS, como se comentó en el capítulo 3.

4.3.8 Espacio de disco para Condor

El directorio "release" de Condor necesita una cantidad específica de espacio de disco, una cantidad bastante pequeña si se considera el tamaño de disco que tienen los ordenadores actuales. Esta es otra razón por la cual es una buena idea tener un sistema de ficheros distribuido en el Condor Pool.

Dependiendo de la plataforma, el espacio de disco necesario para el directorio "release" se lista en la siguiente tabla:

Plataforma	Tamaño
Intel/Linux	11 Mbytes (compilado estáticamente)
Intel/Linux	6.5 Mbytes (compilado dinámicamente)
Intel/Solaris	8 Mbytes
Intel/Solaris	10 Mbytes
SGI/IRIX	17.5 Mbytes
Alpha/Digital Unix	15.5 Mbytes

Tabla 4. 4 Espacio de disco para Condor

Además, de esto, también se necesita bastante espacio de disco en el directorio local de las máquinas que someten y ejecutan tareas en el Condor Pool.

4.4 Pasos a la hora de instalar

Antes de realizar las instalaciones del software de Condor en las máquinas que van a formar parte del Condor Pool, se deben conocer los pasos que se van a seguir en el fichero de instalación y así conocer de antemano que tipo de instalación se debe hacer en cada máquina [44].

Este apartado es muy importante ya que, a partir de las respuestas a la hora de instalar, así quedarán configuradas las máquinas en Condor, con lo cual, cualquier error en la instalación puede ocasionar el mal funcionamiento de la aplicación Condor en el Pool.

La manera más fácil de realizar la instalación de Condor es por medio de los scripts de los cuales esta provisto Condor. Estos son "condor_install" y "condor_init" que se encuentran dentro del software descargado. Estos scripts hay que ejecutarlos en el usuario donde se van a ejecutar los demonios de Condor en la máquina. Para

una mayor seguridad, como se explicó anteriormente, es mejor ser usuario privilegiado Root.

Para poder ejecutar los ficheros de instalación hay que asegurarse de tener instalado Perl [45] en /usr/bin/perl, ya que para ejecutar estos scripts se usará este módulo. Un ejemplo de la ejecución de este script es el siguiente:

maquina1:~ #/software/inscondor/condor-6.2.1 > perl condor_install

El script condor_install divide la instalación en varios pasos. Estos varían dependiendo de la instalación que se realice. A continuación se muestran los tipos de instalación que se pueden realizar y los pasos que aparecerán en cuanto a la instalación.

4.4.1 Instalación completa (full-install)

La instalación full-install se debe realizar en la máquina del Condor Pool que vaya a actuar como Central Manager. Como se dijo en el principio de este capítulo, esta instalación debe ser la primera que se realice.

En caso de estar configurando un Condor Pool donde no se ha instalado un servidor de ficheros distribuido, esta instalación también se debe hacer en todas aquellas maquinas del Pool que vayan a trabajar como Execute Machine o como un nodo de tareas donde se pueden someter y ejecutar tareas (capítulo 2), pero siempre, respetando la primera instalación de todas, el Central Manager.

Paso 1. ¿Qué tipo instalación quiere?

Como es obvio se elegirá full-install.

Paso 2. ¿Cuántas máquinas se quieren establecer con la instalación full-install?

Las opciones que se tienen son varias y se debe de elegir la que más se acerque a la situación en la que se encuentra la red Condor Pool:

- Si se quiere instalar Condor en varias máquinas y se tiene un sistema de ficheros distribuido en la red, entonces en este paso se deben indicar los hostnames de las máquinas que van a formar parte de ese Condor Pool.
- Si se quiere instalar Condor en varias máquinas y no se tiene un sistema de ficheros distribuido en la red, entonces, se debe de ejecutar el fichero condor_install localmente en cada máquina. Esta pregunta se contestará en cada una de las máquinas con los hostnames de las máquinas donde se va a instalar Condor.
- Si se está provisto de una lista con los nombres de las máquinas que van a formar parte del Pool, Condor usará los nombres automáticamente creando directorios y ficheros. Al final, condor_install verterá esta lista al fichero “roster” (fichero que se creará durante la instalación y estará situado en el directorio etc/). Este fichero será usado por los scripts de Condor para ayudar a mantener el Condor Pool.
- Si Condor solo se está instalando en una máquina, no se debe de introducir ningún hostname, solamente la máquina local, en caso de que esta no vaya a trabajar como Central Manager.

Paso 3. Instalando el directorio release

El directorio “release” contiene cuatro subdirectorios: /bin, /etc, /lib y /sbin, como se vio en el apartado anterior. Estos son:

- bin: contiene programas ejecutables a nivel de usuario.
- etc: es el lugar recomendado para los ficheros de configuración Condor, y también incluye el directorio “ejemplo” con ficheros de configuración por defecto y otros ficheros por defecto usados para la instalación de Condor.
- lib: contiene librerías para enlazar los programas de usuario Condor y los scripts usados por el sistema Condor.
- sbin: contiene todos los programas ejecutables y los demonios Condor.

Si se tienen múltiples máquinas con un sistema de ficheros distribuido en el cual será ejecutado Condor, se debe de poner el directorio “release” en el sistema de ficheros distribuido, así se tendrá una sola copia de todos los binarios y cuando se actualicen, se puede hacer desde el servidor de ficheros, mejorando así el rendimiento de la red.

En este punto de la instalación, condor_install intentará encontrar un directorio “release” en la máquina. Pueden ocurrir varias cosas:

- Si el directorio esta instalado ya en la máquina, saldrá un mensaje preguntando si el path en el que se encuentra instalado es el path deseado para la instalación. Si no es así se debe indicar el path en el que se quiere situar el directorio.
- Si no se puede encontrar ningún directorio release, saldrá un mensaje preguntando si la máquina tiene algún directorio release instalado. Si no se tiene ninguno instalado, será porque quizás no se ha descomprimido el fichero release.tar de la distribución binaria descargada. Si se tiene un directorio release, se debe indicar el path en el que se encuentra.

Paso 4. ¿Cómo y dónde debería Condor enviar un e-mail si algo va mal?

Si algo va mal, Condor enviará un e-mail al administrador de Condor por si se necesita atención humana. Por tanto, es necesario especificar la dirección de e-mail del administrador. Hay también que especificar la ruta completa para el programa “mail” que Condor usa para enviar el e-mail. El programa mail [46] necesita entender la opción “-s”, para especificar a un sujeto el mensaje saliente. La opción por defecto que aparecerá será la correcta.

Hay que tener cuidado con algunas distribuciones de Linux, ya que pueden surgir variaciones en la ruta de mail. Si el script de la instalación da signos de que no puede encontrar el programa mail que está especificado, se puede intentar buscar la ruta en la que se encuentra de forma manual, ejecutando en la línea de comandos:

maquina1:~ #/which mail

Este comando debe devolver la ruta donde se encuentra el programa mail. La ruta suele ser “/usr/bin/mail”. Si no funciona, intentar:

maquina1:~ #/which mailx

Si sigue sin encontrarse la ruta, se debe preguntar al administrador del sistema, que verificará si el programa usa el soporte “-s”. La página del manual de Linux puede servir de ayuda en estos casos.

Paso 5. Sistema de ficheros y dominios UID

Mientras que Condor no depende de un sistema de ficheros distribuido o espacio común UID para ejecutar tareas, en las tareas del entorno "vanilla" (tareas que no se redireccionan hacia las librerías de Condor) es necesario el uso de un sistema de ficheros distribuido y un espacio común UID. Por lo tanto, es muy importante para una correcta configuración de Condor tener conocimiento de la red para ver si se usa o no un sistema de ficheros distribuido.

Sí se usa un sistema de ficheros distribuido en la red (como el caso que se plantea en el proyecto), “condor_install” configurará el parámetro FILESYSTEM_DOMAIN con el nombre de dominio de las máquinas (ver apéndice). Si el nombre de dominio ha sido establecido antes, al configurar NIS, en el fichero de configuración se deberá observar el parámetro establecido a dicho dominio:

FILESSYSTEM_DOMAIN = “nombre de domino”

Si no se usa sistema de ficheros distribuido, FILESYSTEMDOMAIN será establecido como $(FULL_HOSTNAME), indicando que cada máquina está en su propio sistema de ficheros, con su propio dominio (ver apéndice).

Para el dominio UID, Condor necesita saber si todos los usuarios a través de todas las máquinas en el Condor Pool tienen un único UID. Por ello es conveniente el uso de NIS, para que todas las máquinas que forman parte del entorno NIS tengan la misma UID. Por ello, el parámetro UID_DOMAIN se establece al nombre de dominio de la máquina que esta ejecutando “condor_install”, es decir:

UID_DOMAIN = “nombre de dominio”

En caso de que no sea así, UID DOMAIN será establecido como $(FULL_HOSTNAME), indicando que cada máquina esta en su propio dominio.

En el caso de tener una UID_DOMAIN común, “condor_install” preguntará si se tiene un “soft UID domain”. Esto significa que aunque se tiene una única UID, no todas las máquinas en el Condor Pool tienen a todos los usuarios en ficheros passwod individuales.

Paso 6. ¿Dónde se deberían instalar los programas públicos?

Se recomienda que los programas Condor a nivel de usuario se instalen en el “directorio release” (donde van por defecto). De esta manera, cuando se quiera instalar una nueva versión de los binarios de Condor, se reemplazará solo el directorio release y será actualizado todo al momento, siempre y cuando se utilice un sistema de ficheros distribuido.

Se recomienda poner también el enlace de alguno de estos subdirectorios en la ruta “usr/local/bin”, apuntando a los programas Condor. Todo lo que se tiene que hacer es indicar el directorio para poner esos enlaces. Se puede aceptar la opción por defecto (/usr/local/bin). De esta manera se pueden tener los binarios instalados en una localización propia. Hasta este punto, los anteriores pasos están basados en la instalación "full install" y “submit-only".

Paso 7. ¿Qué máquina será el central manager?

Se debe de indicar el nombre de la máquina que será el Central Manager. Si "condor_install" no puede encontrar información sobre esta máquina, preguntará por su "nameserver", en caso de seguir sin obtener información, pintará un mensaje de error y pedirá la confirmación por parte del usuario.

Paso 8. ¿Dónde se sitúa el directorio local?

"condor_install" intenta averiguar cual será el directorio que se va a utilizar para Condor. Puesto que el directorio necesita ser único, lo normal es usar el hostname de cada máquina en este nombre.

Cuando se indique la ruta del directorio, se puede usar "condor install", de manera que se expandirá el hostname de la máquina que se quiere ejecutar con Condor. "condor_install" intenta crear los directorios correspondientes para todas las máquinas que se le indicaron en el paso 2.

Una vez indicado el directorio local, "condor install" crea todos los subdirectorios necesarios de cada uno con los permisos apropiados. Estos deberán tener los siguientes permisos:

```
drwxr-xr-x   2 condor root 1024 Jul 18 12:39 execute/
drwxr-xr-x   2 condor root 1024 Jul 18 12:39 log/
drwxr-xr-x   2 condor root 1024 Jul 18 12:39 spool/
```

Si el directorio local es un sistema de ficheros distribuido, "condor_install" apunta a la localización de los ficheros "lock". En este caso, cuando "condor_install" termine, se tendrá que ejecutar "condor_init" en cada máquina que forme parte del Condor Pool para crear el directorio "lock" antes de ejecutar Condor.

Paso 9. ¿Dónde estarán situados los ficheros de configuración local (de la máquina específica)?

El fichero de configuración local se instala en la ruta "directoriorelease/etc/condor_config". Hay que tener en cuenta que se tienen máquinas específicas, o ficheros de configuración local, que no tienen en cuenta lo establecido en el fichero global.

Si se esta instalando Condor en múltiples máquinas o se está configurando para una máquina que actuará de Central Manager, se debe seleccionar una localización para los ficheros de configuración local. Las dos opciones más recomendables son tener un único directorio (etc/), del que cuelgan todos los ficheros de configuración local, cada uno llamado "$(HOSTNAME).local", o tener los ficheros de configuración local que van en directorios locales individuales en cada máquina (ver apéndice).

Dado un sistema de ficheros distribuido, se recomienda la primera opción, ya que hace más fácil la configuración del Condor Pool desde una localización centralizada.

A continuación se muestra como sería la estructura del directorio en el cual se encuentran los ficheros de configuración local del Pool de la figura 4.2.5, entre otros.

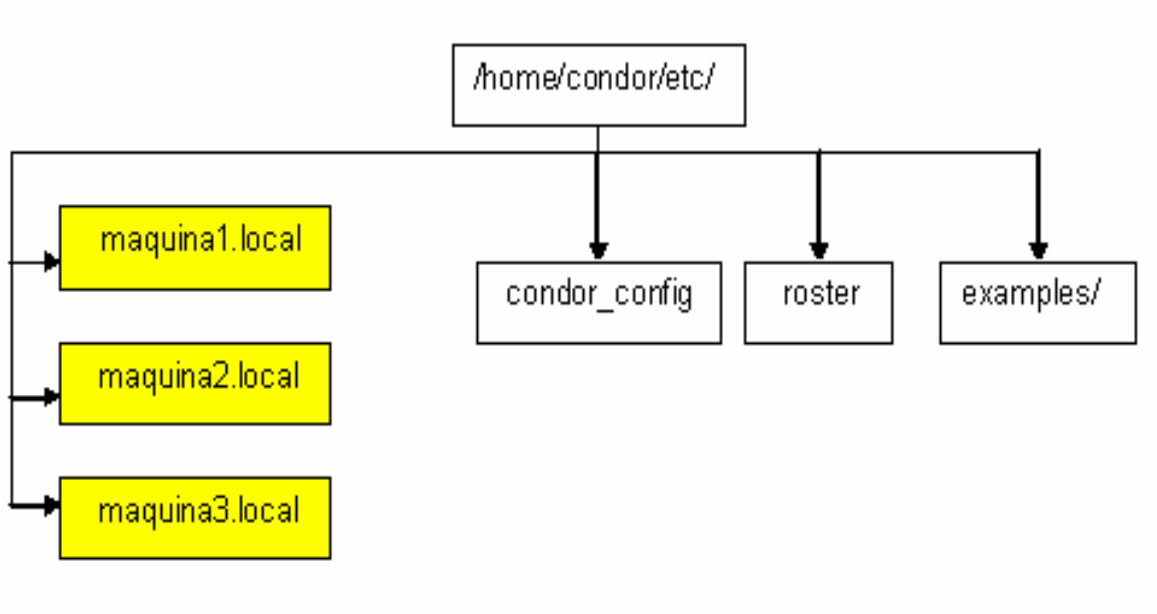

Figura 4. 11 Directorio etc/ distribuido

Paso 10. ¿Cómo podría Condor encontrar el fichero de configuración?

Hay varias rutas donde Condor mira para encontrar el fichero de configuración. Se recomienda que se sitúe en un enlace desde “directoriorelease/etc/condor_config”. De esta manera, se puede mantener la configuración de Condor en una localización centralizada, donde todos los procesos y herramientas de Condor serán capaces de encontrar los ficheros de configuración. “condor_install” preguntará si se quiere crear un enlace o dos localizaciones fijadas para que Condor busque.

De forma alternativa, se puede establecer la variable del entorno CONDOR_CONFIG para que indique lar ruta directoriorelease/etc/condor_config (ver apéndice).

Una vez completado el paso 10, se termina con la instalación. “condor_install” pintará un mensaje en la pantalla aconsejando que antes de ejecutar Condor se deben de editar el fichero de configuración para realizar los cambios necesarios, dependiendo de los parámetros que se deseen modificar para la configuración deseada y ejecutar condor_init.

4.4.2 Instalación del Central Manager

La otra posibilidad de instalación es la de Central Manager. Si la idea es la de realizar una instalación full-install y que la máquina local sea el Central Manager, lo mejor es realizar una instalación full-install, ya que la configuración del Central Manager se especifica en uno de los pasos de instalación.

Por tanto, solo se elegirá la opción de Central Manager en el caso de que el Condor Pool se configure de manera que el servidor de ficheros se encuentre en una máquina y el Central Manager se instale en otra máquina diferente. Como se comentó en el capítulo anterior, esta configuración no será eficiente, y tampoco objetivo del proyecto. De cualquier manera, se comenta para otras posibles situaciones.

Paso 1. ¿Le gustaría establecer esta máquina como Central Manager de Condor?

Solo se debe de elegir esta opción si se tiene una red Configurada como Condor Pool con una instalación full-install en un servidor de ficheros y se quiere establecer la máquina local como Central Manager. En caso contrario, se podrá continuar con la instalación hasta el Paso3, donde se suspenderá, como se verá a continuación.

Paso 2. ¿Cuántas máquinas se establecen para usar Condor?

Como lo que se indica es la instalación de un Central Manager, solo se debe de indicar la máquina local.

Paso 3. Instalación del "directorio release", del cual cuelgan varios binarios, librerías, scripts y ficheros usados por Condor

Si no se ha instalado Condor nunca en la máquina, al llegar a este paso, no se encontrará ningún "directorio release", por lo tanto, Condor preguntará que no puede encontrar un "directorio release" completo de Condor. Preguntará si se ha instalado algún "directorio release" manualmente. Dependiendo de la respuesta así se actuará.

En caso de que la respuesta sea afirmativa, se deberá de especificar la ruta en la cual se encuentra ese "directorio release". En caso de que no se tenga ningún "directorio release", Condor pregunta donde gustaría instalar el "directorio release", dando una ruta por defecto [/usr/local/condor]. En principio este directorio no existe, así que se deberá aceptar la creación del mismo. Si no se desea esta ruta, se puede especificar la ruta donde se quiera tener este directorio.

Al finalizar la instalación del "directorio release", saldrá un mensaje de que no se puede encontrar el fichero de configuración local para la máquina. Esto es debido a que no se ha realizado una instalación full-install todavía, con lo cual, como se explicó anteriormente, se debe de realizar primero una instalación full-install primero, eligiendo la opción de Central Manager en el primer paso.

4.4.3 Instalación Submit Only

Una máquina submit-only puede someter tareas en un Condor pool, pero las tareas que se sometan en el Pool no se podrán ejecutar en dicha máquina.

Si se está planteando el instalar solo una máquina del Condor Pool como Submit-Only, se puede instalar en cualquier máquina Condor como Root o como usuario Condor (siempre teniendo en cuenta las necesidades de las máquinas que trabajan como Submit Machine (capítulo 2)).

La instalación de Submit-Only de Condor implica que la máquina puede someter tareas en uno o varios Pools, por tanto se esta opción se debe considerar en la configuración. Para la configuración de una máquina Submit en un Condor Pool se realizan los mismos pasos que en full-install hasta el paso 6. Los siguientes pasos de instalación se citan a continuación:

¿A cuántos pools diferentes se quiere añadir esta máquina?

Se debe de contestar con el número de Condor Pools a los que dicha máquina va a someter tareas.

¿Cuál es el hostname del Central Manager?

Bastará con indicar la dirección IP de la máquina que en el Condor Pool va a actuar como Central Manager.

¿Dónde se quieren poner los ficheros de configuración de Condor para el Pool en el que se va a trabajar?

La ruta por defecto que da Condor es /root/condor/etc/condor_config. En caso de querer instalar los ficheros en otra ruta diferente, solo habrá que especificarla, teniendo en cuenta que la ruta que se indique debe existir.

¿Qué nombre se quiere usar para el schedd en el Condor Pool?

En esta pregunta el software de instalación da la opción de poder darle un nombre al proceso schedd con el que dicha máquina va a trabajar para someter tareas. En caso de no saber o no entender la pregunta Condor indica que se acepte el nombre por defecto [root@linux.local].

A partir de este punto, se crea el fichero de configuración global, se configura y termina la instalación.

4.5 Situación de la red prototipo. Solución propuesta

Una vez que conocida la estructura del software de Condor, así como los aspectos a tener en cuenta a la hora de instalar en cuanto a las características de las máquinas y las posibles instalaciones, se deciden cuales son los roles que van a jugar cada una de las máquinas, así como el papel a desempeñar para NFS y NIS. Notar que la instalación que se realice posteriormente en el laboratorio IT-3 será de características semejantes.

El servidor NFS y NIS será el servidor que actúe como Central Manager, es decir la máquina1 (192.168.19.1) será el repositorio global de información del Pool. Las demás máquinas actuarán como clientes NFS y NIS.

Una primera visión del papel que van a desempeñar cada una de las máquinas para establecer el Universo Standard en la red es la siguiente:

Nombre	maquina1
Dirección IP	192.168.19.1
NFS	Servidor
NIS	Servidor
Condor Pool	Central Manager

Tabla 4. 5 Máquina 1 en Universo Standard

Nombre	maquina2
Dirección IP	192.168.19.2
NFS	Cliente
NIS	Cliente
Condor Pool	Nodo. Somete y ejecuta tareas Submit-Execute Machine

Tabla 4. 6 Máquina 2 en Universo Standard

Nombre	maquina3
Dirección IP	192.168.19.3
NFS	Cliente
NIS	Cliente
Condor Pool	Nodo. Somete y ejecuta tareas Submit-Execute Machine

Tabla 4. 7 Máquina 3 en Universo Standard

La siguiente figura muestra como quedará la red prototipo (Condor Pool) teniendo en cuenta los aspectos antes comentados, los recursos que se tienen y los requisitos que se piden, en vistas a los laboratorios del Área de Ingeniería Telemática.

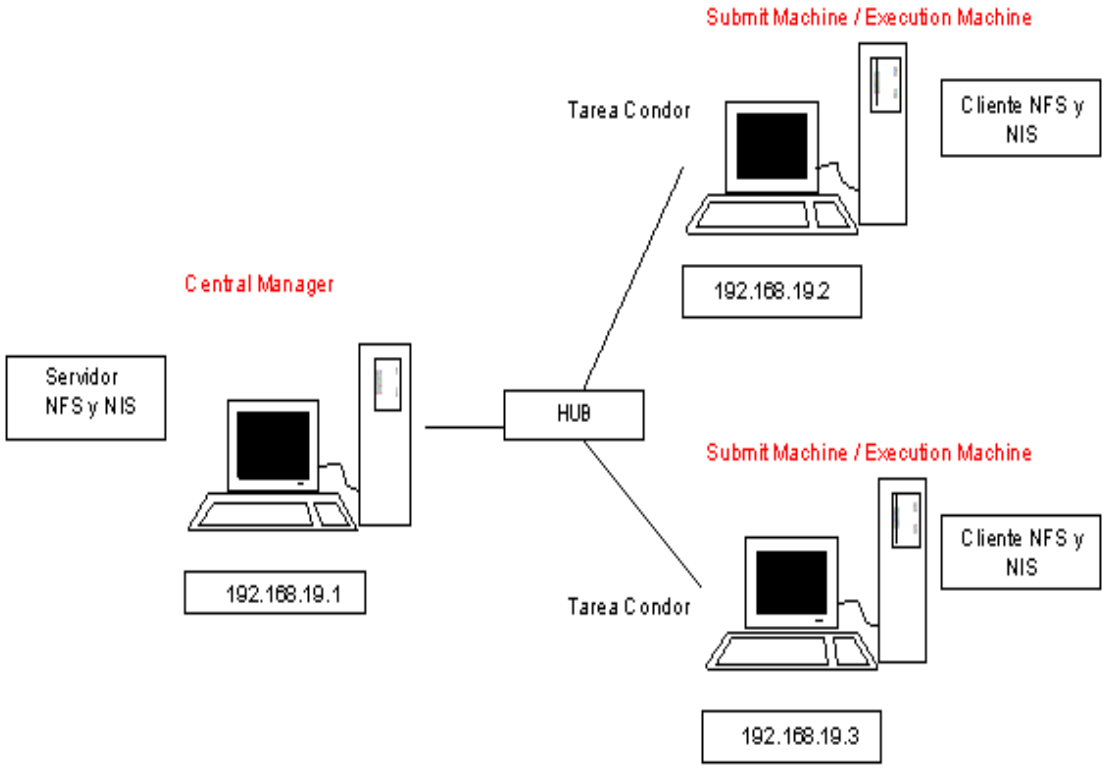

Figura 4. 12 Solución propuesta

A Continuación se comentan las instalaciones y configuraciones de los diferentes mecanismos necesarios para poner en marcha la aplicación. Estos no se han incluido en apéndice para poder tener en un mismo bloque todo lo referente a la instalación y configuración de Condor. De esta manera, será mucho más cómodo a la hora de utilizar el proyecto como referencia para otras instalaciones posteriores.

4.6 Usuario Condor

Para poder hacer uso de la aplicación Condor, es necesario tener en las máquinas que van a formar parte del Condor Pool y se va a instalar en ellas el software de Condor un usuario Condor, ya que a la hora de instalar el software en unos de los pasos se exige tener un usuario Condor para poder continuar con la instalación.

Si se decide instalar Condor en un Pool que tiene un sistema de ficheros distribuido y se decide instalar Condor bajo ese sistema de ficheros no es necesario instalar la Aplicación Condor en todas las máquinas, bastará con instalar la aplicación solamente en una de las máquinas del Condor Pool. Esta máquina será el servidor de ficheros y por tanto el Central Manager. Por tanto la creación de un usuario Condor será solo necesaria para la máquina Central Manager.

Para crear un usuario Condor, se puede hacer de forma manual o usando la herramienta de trabajo Yast2 que proporciona Suse 7.2. Para mayor comodidad se utiliza la herramienta poniendo en la línea de comandos yast2. Por tanto los pasos a seguir son:

1- Una vez ejecutada la herramienta en la línea de comandos se elige la opción de “seguridad & usuarios” y dentro de esta la opción de “crear un nuevo usuario”.

Figura 4. 13 Crear usuario en Yast2

2- Una vez dentro de la opción “crear un nuevo usuario” se debe rellenar el formulario que aparece, dándole un nombre al usuario y una contraseña. Se pulsa crear y el usuario se creará con todas las características indicadas. Se puede el fichero log creado para comprobarlo.

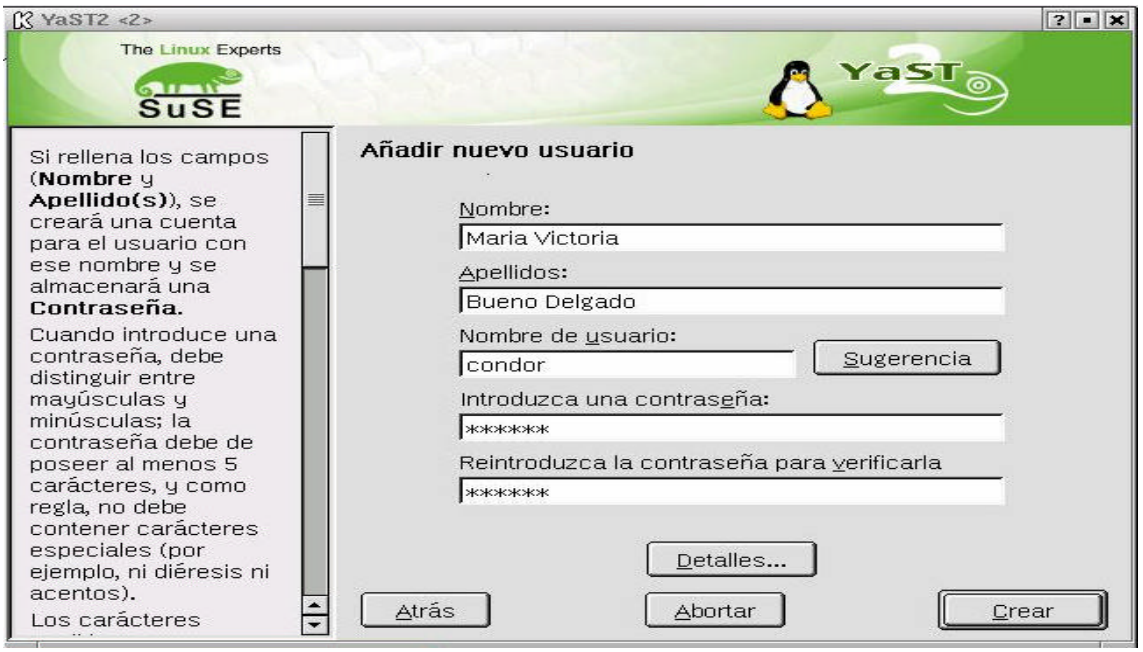

Figura 4. 14 Añadiendo usuario Condor

4.7 Instalación y configuración NFS

Conocida la configuración que seguirá la red propuesta, se procede en primer lugar a la instalación y configuración del servidor y cliente NFS.

4.7.1 Servidor NFS

El servidor NFS, como se comentó en el capítulo 3, será la máquina que en el Condor Pool desempeñe el papel de Central Manager. Se ha elegido para este papel la máquina 1 (192.168.19.1).

La instalación y configuración de dicho Servidor puede realizarse de dos formas diferentes: manualmente o utilizando la herramienta YAST2 de la Suse7.2. En este caso se utilizará la herramienta para facilitar la labor. Los pasos a seguir serán los siguientes:

1- Se ejecuta el comando yast2 en la línea de comandos del terminal. A continuación aparecerá un interfaz gráfico que muestra las diferentes labores que se pueden realizar con dicha herramienta. Se elige la opción de “Red avanzada” y dentro de ella la opción de “Servidor NFS”.

Figura 4. 15 Instalación de Servidor NFS en Yast2

2- Dentro de Servidor NFS se debe elegir la opción de arrancar el servidor y se debe indicar el punto de anclaje de dicho Servidor, a partir del cual todas las máquinas que actúen como clientes NFS tendrán acceso a todos los ficheros y directorios de ese punto de anclaje. Como se indicó en apartados anteriores, si en la máquina se creado un usuario "Condor", es conveniente que dicho usuario se encuentre dentro del punto de anclaje del servidor de ficheros. Por tanto, el punto de anclaje elegido ha sido la ruta /home. Esto quiere decir que los demás usuarios del servidor NFS accederán a él a través de esa ruta (capítulo 3).

Figura 4. 16 Directorios a exportar en NFS

Una vez que se han guardado los cambios en la configuración del servidor NFS, si se ejecuta en la línea de comandos "rpcinfo –p" (capítulo 3) se podrá observar como se ha iniciado el servidor NFS, visualizando los demonios de NFS.

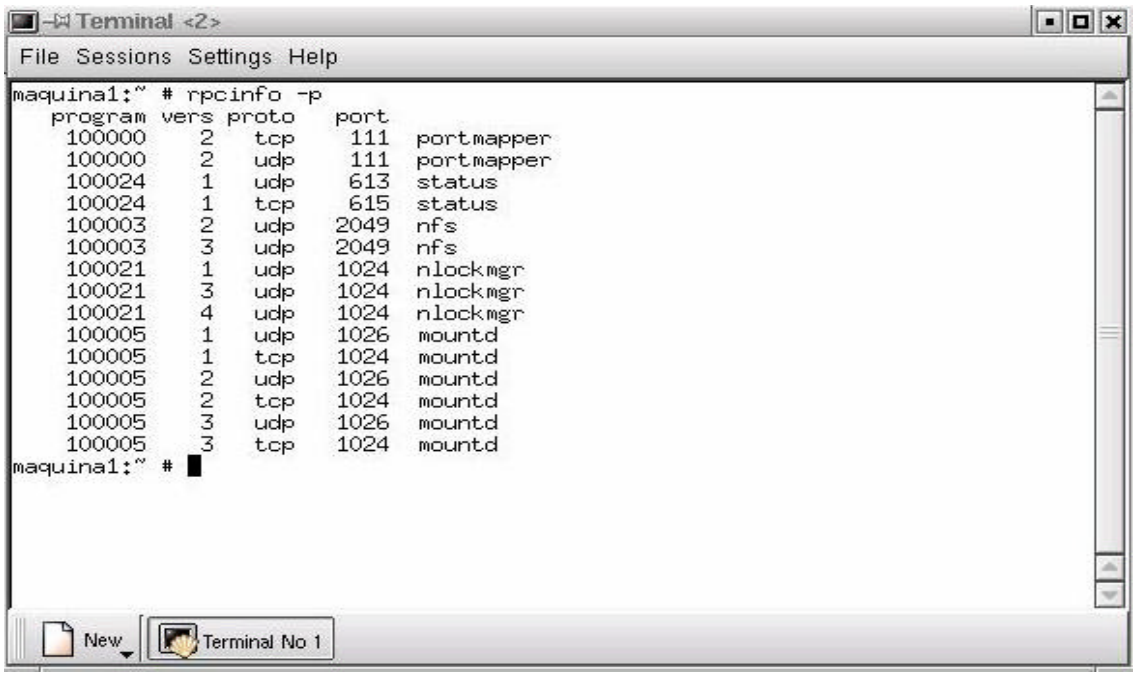

Figura 4. 17 Vista de rpcinfo

4.7.2 Cliente NFS

Al igual que para el servidor NFS, se utilizará la herramienta Yast2 para realizar la configuración de los clientes. En esta configuración se debe indicar, por una parte, la dirección IP del servidor NFS que se ha instalado en la red, y por otro lado, el punto de anclaje del cliente y del servidor NFS.

Los clientes NFS serán las máquinas 2 y 3. En estas se debe elegir cual será el punto de anclaje a través del cual se accederá al Servidor NFS y por tanto a los recursos de la aplicación Condor. Para el cliente NFS también se ha elegido como punto de anclaje local /home, por tanto, la configuración en el cliente será la siguiente:

Figura 4. 18 Cliente NIS

Se pulsa aceptar y se guardan los cambios. A partir de ese momento ya se puede comprobar si ha sido satisfactoria la configuración de NFS. La siguiente figura muestra como queda instalado el sistema de ficheros distribuido NFS en la red.

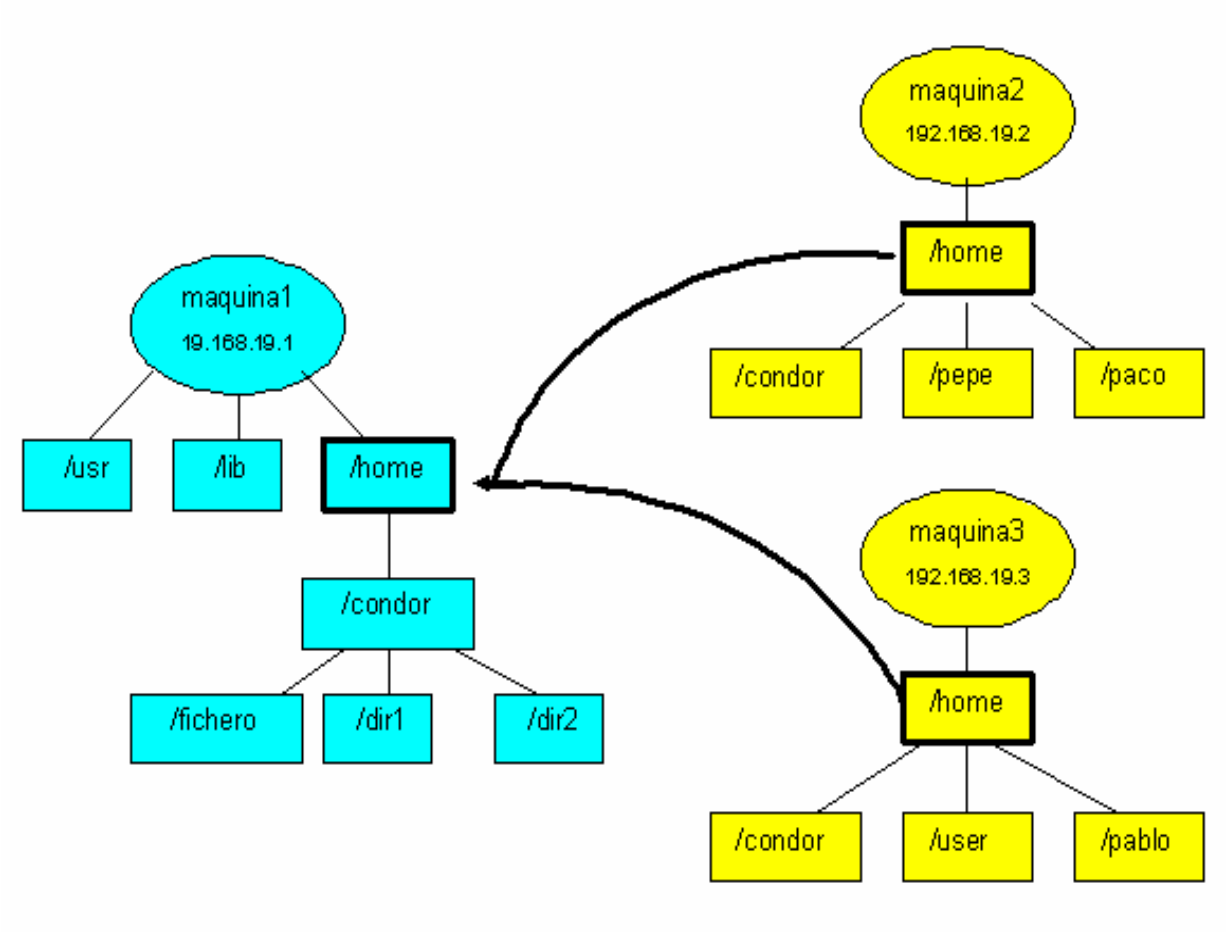

Figura 4. 19 Configuración Cliente-Servidor en red propuesta

4.8 Instalación y configuración NIS

4.8.1 Servidor NIS

Una vez elegida la máquina que será servidor NIS, que obviamente debe ser la misma que la del servidor NFS (máquina1, 192.168.19.1), se debe trabajar como usuario privilegiado Root. En la distribución que viene por defecto, no aparece el paquete de NIS ypserv, por lo tanto habrá que instalarlo. La forma más rápida y sencilla es bajar de la página de Suse http://www.suse.com el paquete ypserv.rpm de la versión de Linux con la que se esté trabajando. Para instalarlo basta con ejecutar en la línea de comandos:

maquina1:~ # rpm –i ypserv.rpm

Ya instalado el paquete, se deberá establecer un dominio NIS, que en este caso será “upct.es”.Los mapas NIS irán al fichero /var/yp/upct.es/ (capítulo 3). De esta manera el servidor determinará si está sirviendo un dominio NIS en particular chequeando si el directorio mapa está presente. Los pasos a seguir para la configuración del servidor NIS son los siguientes:

1- Se debe definir el nombre de dominio, mediante el comando domainname.

maquina1:~ # domainname upct.es

Si ahora se ejecuta domainname en la línea de comandos, nos devolverá el dominio establecido, es decir, “upct.es”.

maquina1:~ # domainname

upct.es

2- Se deben crear los mapas de NIS. Para ello se debe ejecutar en la línea de comandos, el comando “ypinit –m”, pero llamando a la ruta completa, es decir /var/lib/yp/ypinit –m.

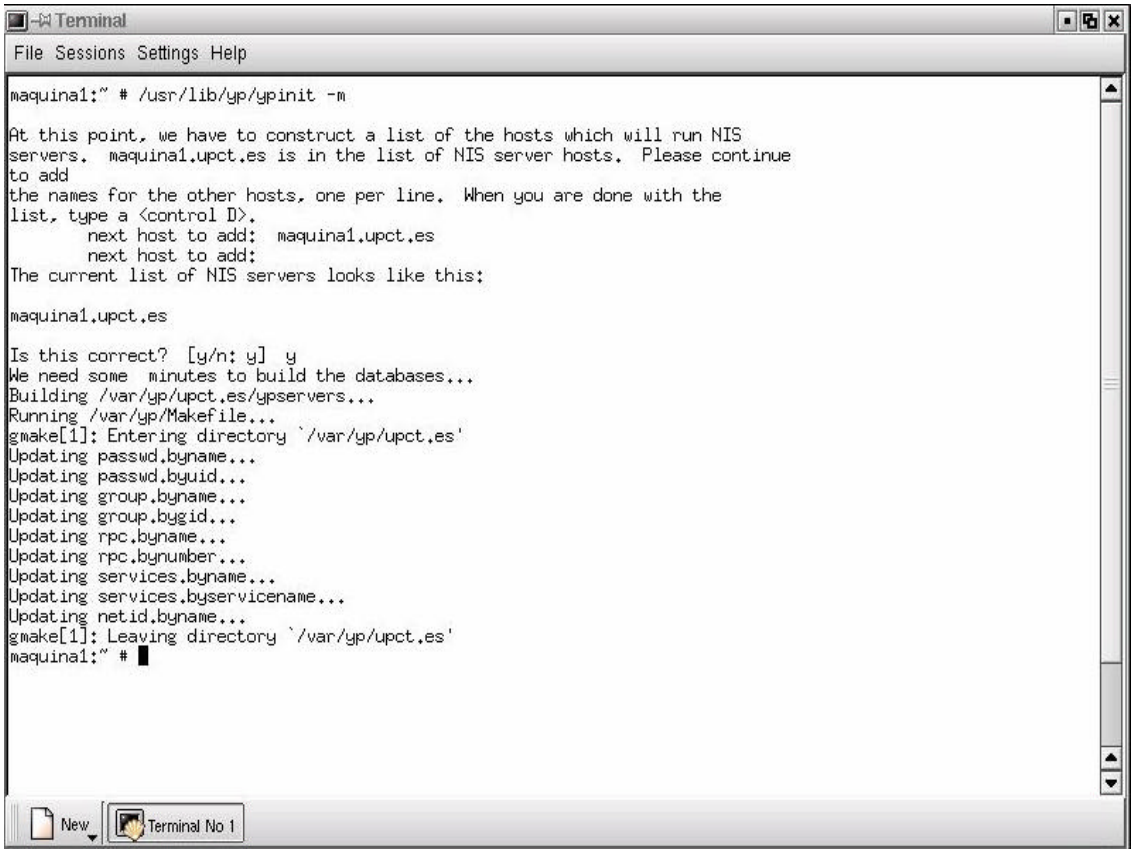

Figura 4. 20 Creando mapas NIS en el servidor

En esta parte se pregunta si se quieren añadir servidores esclavos (capítulo 3) al servidor primario que se está instalando. Como no interesa para esta red, bastará con pulsar CTRL + D. Se crearán automáticamente el dominio y los mapas en /var/yp/upct.es.

3- Una vez hecho esto hay que arrancar el demonio ypserv (en el caso de que haya uno hay que pararlo)

maquina1:~ # ypserv

Si se ejecuta el comando rpcinfo -p se ven ejecutándose los demonios ypserv.

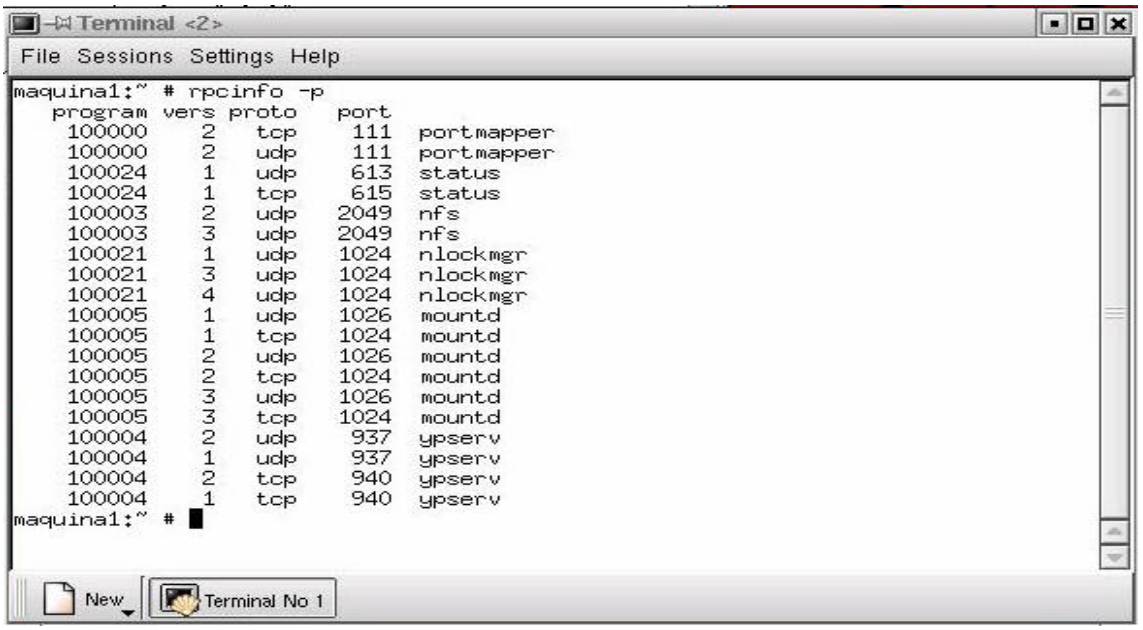

Figura 4. 21 Procesos ypserv del Servidor

4.8.2 Cliente NIS

Una vez configurado el servidor NIS se lleva a cabo la configuración de los clientes NIS (máquina 2 y máquina3). Hay que tener en cuenta que un cliente NIS puede pertenecer a más de un dominio. Los pasos para la configuración son los siguientes:

Configurar el fichero etc/yp.conf. Dentro de este, se debe modificar el último párrafo y escribir el nombre de dominio y el servidor de ese dominio.

domain upct.es server 192.168.19.1.

Hay que asegurarse de que el fichero tiene permiso de lectura para todos los usuarios. Si no es así, se modificarán los permisos mediante el comando chmod [47].

maquina2:~ /etc # chmod 755 yp.conf

Se debe arrancar el proceso ypbind indicando la ruta en la que se encuentra. En caso de que el proceso ya esté arrancado hay que pararlo y volver a arrancarlo [29].

maquina2:~ # usr/sbin/ypbind

A partir de este momento si se introduce en la línea de comandos rpcinfo -p se pueden visualizar los demonios de uno de los clientes NIS ejecutándose en la máquina.

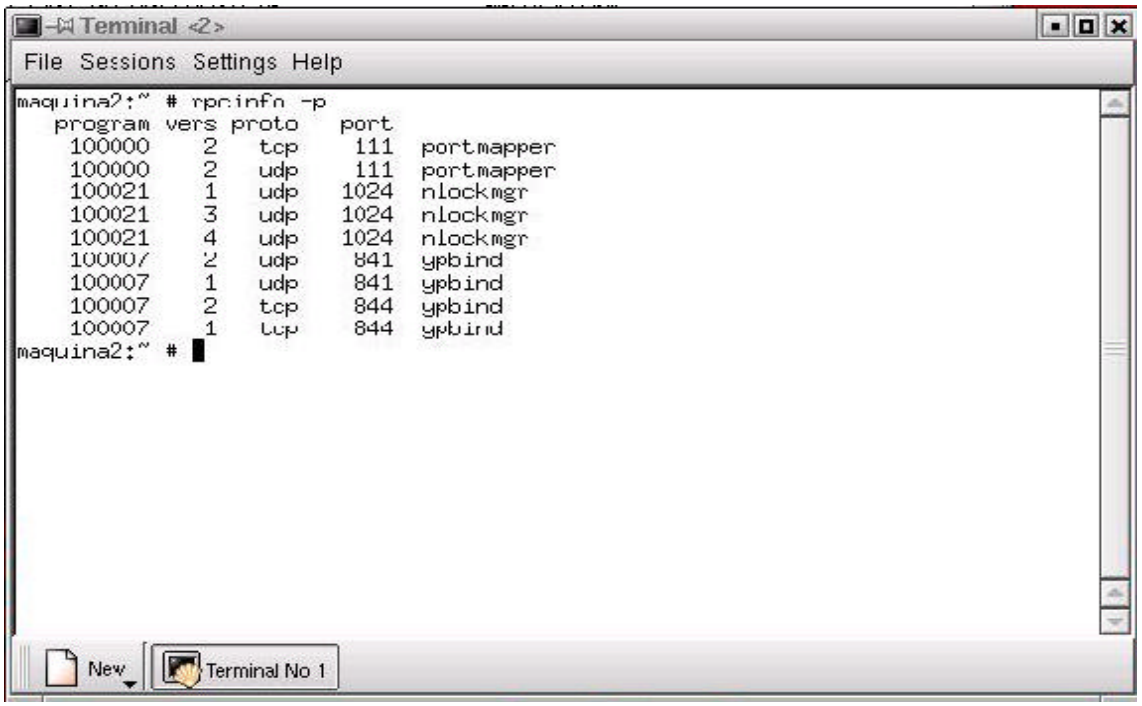

Figura 4. 22 Procesos ypbind en el cliente

Para comprobar el resultado de la configuración basta con ejecutar alguno de los comandos que se comentaban en el capítulo 3 sobre NIS. Por ejemplo, si se ejecuta el comando ypwhich –d “nombre del dominio”, aparece la dirección del servidor NIS al cual pertenece el cliente NIS.

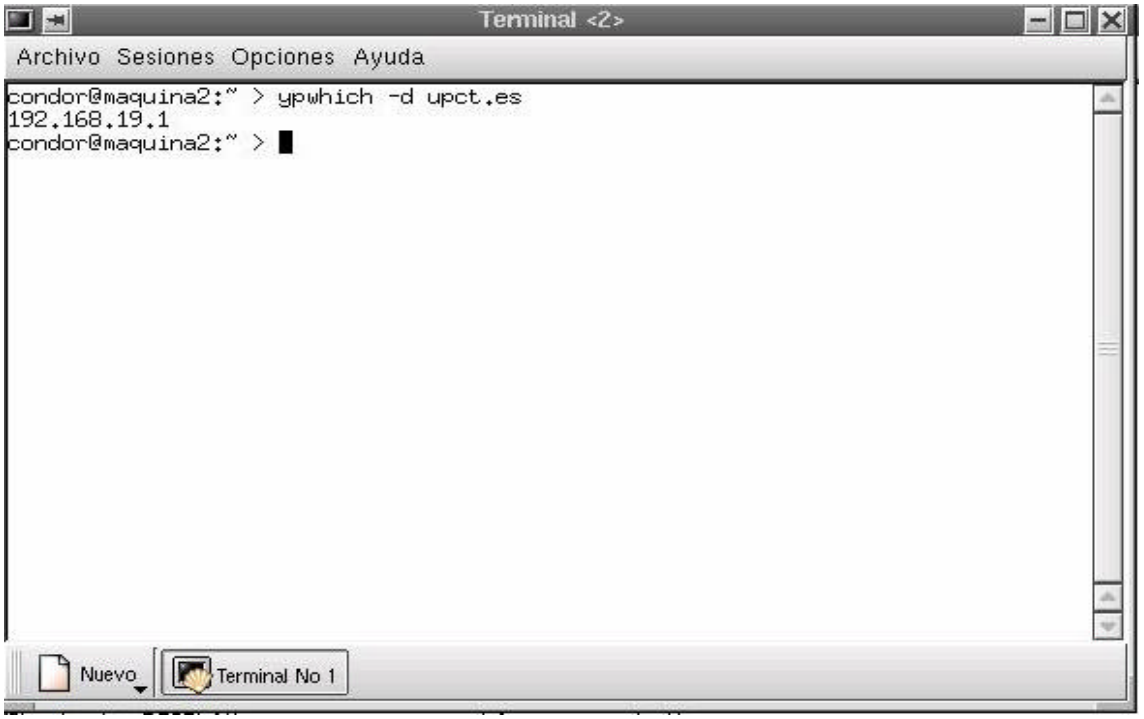

Figura 4. 23 Dominio NIS de la red propuesta

4.9 Instalación y configuración del Pool

Se ha decidido instalar un Condor Pool con un entorno Standard. En el entorno Standard, el uso de un sistema de ficheros distribuido no es obligado, pero en esta red si se va a utilizar, así se tendrá a disposición de los usuarios un sistema de ficheros compartidos.

Normalmente, cuando en un Condor Pool se utiliza un sistema de ficheros distribuido, el software de Condor se instala únicamente en el Servidor de ficheros. De esta manera, todas las máquinas del Condor Pool acceden al software de Condor a través de dicho servidor de ficheros. Esto ocurre en la configuración del Universo Vanilla.

Para la instalación del Condor Pool en la red propuesta, el software de Condor se instalará en cada una de las máquinas que van a formar parte del Condor Pool, independientemente de si se utiliza o no un servidor de ficheros distribuido.

Como se comentó en el inicio del capítulo, Condor se instalará comenzando por la máquina que va a trabajar de Central Manager. Por tanto situándose en dicha máquina (maquina 1 192.168.19.1), y trabajando como usuario privilegiado Root, se procede a la instalación del Central Manager del Condor Pool.

4.9.1 Central Manager

4.9.1.1 Instalación

Una vez aclaradas todas las preguntas de los apartados anteriores sobre los pasos a la hora de instalar, se procede a la instalación de Condor. Situándose en el directorio en el que se encuentra el script “condor_install” se ejecuta en la línea de comandos el programa para comenzar la instalación.

maquina1:~ /software/inscondor/condor-6.2.1 # ls

bin etc include sbin lib *release.tar LICENSE.TXT INSTALL DOC README condor_install examples*

maquina1:~ /software/inscondor/condor-6.2.1 # perl condor_install

```
*******************************************************************
*****
        STEP 1: What type of Condor installation do you want?
*******************************************************************
*****
Would you like to do a full installation of Condor? [yes]

*******************************************************************
*****
        STEP 2: How many machines are you setting up for Condor?
*******************************************************************
*****
Are you planning to setup Condor on multiple machines? [yes] no

*******************************************************************
*****
        STEP 3: Install the Condor "release directory", which holds
        various binaries, libraries, scripts and files used by Condor.
```

```
*******************************************************************
*****
I can't find a complete Condor release directory.

Have you installed a release directory already? [no]

Where would you like to install the Condor release directory?
[/usr/local/condor]
That directory doesn't exist, should I create it now? [yes]
Installing a release directory into /usr/local/condor ...

Using /usr/local/condor as the Condor release directory.

*******************************************************************
*****
        STEP 4: How and where should Condor send email if things go
wrong?
*******************************************************************
*****
If something goes wrong with Condor, who should get email about it?
[root@maquina1.upct.es]

What is the full path to a mail program that understands "-s" means
you want to specify a subject? [/usr/bin/mail]

Using /usr/bin/mail to send email to root@maquina1.upct.es

*******************************************************************
*****
        STEP 5: Filesystem and UID domains.
*******************************************************************
*****
To correctly run all jobs in your pool, including ones that aren't
relinked
for Condor, you must tell Condor if you have a shared filesystem, and
if
so, what machines share it.

Please read the "Configuring Condor" section of the Administrator's
manual
(in particular, the section "Shared Filesystem Config File Entries")
for a complete explaination of these (and other, related) settings.

Do all of the machines in your pool from your domain ("upct.es")
share a common filesystem? [no] yes

Configuring all machines to use "upct.es" for their filesystem domain.

Do all of the users across all the machines in your domain have a
unique
UID (in other words, do they all share a common passwd file)? [no] yes

Configuring all machines to use "upct.es" for their uid domain.

In some cases, even if you have unique UIDs, you might not have all
users
listed in the password file on each machine.
Is this the case at your site? [no]

*******************************************************************
*****
        STEP 6: Where should public programs be installed?
```

```
*******************************************************************
*****
The Condor binaries and scripts are already installed in:
        /usr/local/condor/bin
If you want, I can create some soft links from a directory that is
already
in the default PATH to point to these binaries, so that Condor users
do not
have to change their PATH.  Alternatively, I can leave them where they
are
and Condor users will have to add /usr/local/condor/bin
to their PATH or explicity use a full pathname to access the Condor
tools.

Shall I create links in some other directory? [yes]

Where should I install these files?
[/usr/local/bin]

*******************************************************************
*****
        STEP 7: What machine will be your central manager?
*******************************************************************
*****
What is the full hostname of the central manager?
[maquina1.upct.es]

Your central manager will be on the local machine.

*******************************************************************
*****
        STEP 8: Where will the "local directory" go?
*******************************************************************
*****
Condor will need to create a few directories for its own use

You have a "condor" user on this machine.  Do you want to put all the
Condor directories in /home/condor? [yes] no

Do you want to put all the Condor directories in
/usr/local/condor/home? [yes]

Creating all necessary Condor directories ... done.

*******************************************************************
*****
        STEP 9: Where will the local (machine-specific) config files
go?
*******************************************************************
*****
Condor allows you to have a machine-specific config file that
overrides
settings in the global config file.

You must specify a machine-specific config file.

Should I put a "condor_config.local" file in /usr/local/condor/home?
[yes]
Creating config files in "/usr/local/condor/home" ... done.

Configuring global condor config file ... done.
Created /usr/local/condor/etc/condor_config.
```

```
Press enter to continue.

Setting up maquina1.upct.es as your central manager

What name would you like to use for this pool?  This should be a
short description (20 characters or so) that describes your site.
For example, the name for the UW-Madison Computer Science Condor
Pool is: "UW-Madison CS".  This value is stored in your central
manager's local config file as "COLLECTOR_NAME", if you decide to
change it later.  (This shouldn't include any " marks).
PROYECTO-UPCT-CONDOR

Setting up central manager config file
/usr/local/condor/home/condor_config.local ... done.

*******************************************************************
*****
        STEP 10: How do you want Condor to find its config file?
*******************************************************************
*****
Condor searches a few locations to find it main config file. The first
place
is the envionment variable CONDOR_CONFIG. The second place it searches
is
/etc/condor/condor_config, and the third place is
~condor/condor_config.
/home/condor/condor_config exists.
Renaming to: /home/condor/condor_config.old.

Should I put in a soft link from /home/condor/condor_config to
/usr/local/condor/etc/condor_config [yes]

Installing links for public binaries into /usr/local/bin ... done.

*******************************************************************
*****
Condor has been fully installed on this machine.
*******************************************************************
*****
/usr/local/condor/sbin contains various administrative tools.
If you are going to administer Condor, you should probably place that
directory in your PATH.

To start Condor on any machine, just execute:
/usr/local/condor/sbin/condor_master

Since this is your central manager, you should start Condor here
first.
```

4.9.1.2 Configuración

Se debe ejecutar condor_init en la máquina para crear los directorios LOCK (capítulo 3) en el disco local por si no han sido creados por el programa condor_install, además con este comando se chequea el directorio de la máquina creado por Condor para comprobar la existencia de los directorios que serán imprescindibles en Condor (/log, /spool, /execute), así como también el fichero de configuración global y el fichero de configuración local.

maquina1:~/software/inscondor/condor-6.2.1 # /usr/local/condor/sbin/condor_init

Creating directory /var/lock/condor

/home/condor/condor_config already exists.

/usr/local/condor/home/log already exists.

/usr/local/condor/home/spool already exists.

/usr/local/condor/home/execute already exists.

/usr/local/condor/home/condor_config.local already exists.

Condor has been initialized, but not started.

Además de esto, en caso de instalar Condor en un sistema de ficheros distribuido, condor_init crea enlaces con cada una de las máquinas que van a formar parte del Pool y se indican en el fichero de configuración. Estos enlaces se necesitan para que dichas máquinas puedan encontrar el fichero de configuración global, pero este no es el caso que se trata en esta instalación.

Por otra parte, se debe copiar el script condor.generic de la ruta /usr/local/etc/examples/condor.generic a /usr/local/bin/condor. Una vez copiado, se deben de realizar los siguientes cambios para especificar la localización del fichero de configuración global del Condor Pool.

$default_pool=""; se sustituye por

$default_pool="default";

y

%configlocation = (

);

sustituirlo por

%configlocation = (

"default", "/usr/local/condor/etc/condor_config",

);

Para que Condor se inicie cada vez que se inicie la máquina se debe copiar el script "condor.boot", localizado en el path /usr/local/condor/etc/examples/condor.boot en la ruta que se indica a continuación. La opción –p del comando cp guarda, si es posible, los atributos de los ficheros con los que se trabaja.

maquina1:~ # cp –p /usr/local/condor/etc/examples/condor.boot /etc/rc.d/init.d/condor

Se debe realizar un enlace simbólico entre los ficheros que se citan a continuación:

maquina1:~ # ln –s /etc/rc.d/init.d/condor /etc/rc.d/rc3.d/S95condor

maquina1:~ # ln –s /etc/rc.d/init.d/condor /etc/rc.d/rc5.d/S95condor

maquina1:~ # ln –s /etc/rc.d/init.d/condor /etc/rc.d/rc0.d/KC4condor

maquina1:~ # ln –s /etc/rc.d/init.d/condor /etc/rc.d/rc6.d/KC4condor

El comando ln crea un enlace (link) entre un fichero existente y un nombre nuevo de fichero. Ambos son idénticos para el usuario. Internamente existe sólo un fichero referenciado por el i-nodo y los enlaces asociados. La opción –s realiza un enlace simbólico a pesar del enlace existente.

Para una completa instalación del script condor_compile de debe cambiar el nombre del enlazador ld del path /usr/bin/ por ld.real y copiar el enlazador de condor en dicho path.

maquina1:~ # mv /usr/bin/ld /usr/bin/ld.real

maquina1:~ # cp –p /usr/local/condor/lib/ld /usr/bin/ld

En cuanto a los ficheros de configuración del Condor Pool, los parámetros que forman parte de estos, obligatoriamente deben de ser modificados para que se constituya el Condor Pool y de esta manera las máquinas que forman parte del mismo puedan tener conocimientos del estado del Pool, así como también de las colas de tareas. Se establecen según las necesidades del Condor Pool que se quiere crear, quedando el fichero de configuración de la siguiente manera (ver apéndice Ficheros de configuración):

Fichero de configuración global

Las modificaciones realizadas en este fichero de configuración se muestran a continuación señaladas (ver apéndice).

En esta primera parte del fichero de configuración, muchos de los parámetros ya están establecidos en cuanto a lo especificado en el script condor_install. Se indican los parámetros que se han modificado en cuanto a la configuración deseada.

```
##
##  Part 1:  Settings you must customize:
##########################################################################
##########################################################################

##  What machine is your central manager?
CONDOR_HOST       = maquina1.upct.es

##--------------------------------------------------------------------
##  Pathnames:
##--------------------------------------------------------------------
##  Where have you installed the bin, sbin and lib condor directories?
RELEASE_DIR       = /usr/local/condor

##  Where is the local condor directory for each host?
LOCAL_DIR         = $(RELEASE_DIR)/home
#LOCAL_DIR        = $(RELEASE_DIR)/hosts/$(HOSTNAME)

##  Where is the machine-specific local config file for each host?
#LOCAL_CONFIG_FILE      = $(LOCAL_DIR)/condor_config.local
LOCAL_CONFIG_FILE = $(RELEASE_DIR)/etc/$(HOSTNAME).local

##--------------------------------------------------------------------
##  Host/IP access levels
##--------------------------------------------------------------------
##  Please see the administrator's manual for details on these
##  settings, what they're for, and how to use them.
```

```
##  What machines have administrative rights for your pool?  This
##  defaults to your central manager.  You should set it to the
##  machine(s) where whoever is the condor administrator(s) works
##  (assuming you trust all the users who log into that/those
##  machine(s), since this is machine-wide access you're granting).
HOSTALLOW_ADMINISTRATOR = $(CONDOR_HOST)

##  If there are no machines that should have administrative access
##  to your pool (for example, there's no machine where only trusted
##  users have accounts), you can uncomment this setting.
##  Unfortunately, this will mean that administering your pool will
##  be more difficult.
#HOSTDENY_ADMINISTRATOR = *

##  What machines should have "owner" access to your machines, meaning
##  they can issue commands that a machine owner should be able to
##  issue to their own machine (like condor_vacate).  This defaults to
##  machines with administrator access, and the local machine.  This
##  is probably what you want.
HOSTALLOW_OWNER = $(FULL_HOSTNAME), $(HOSTALLOW_ADMINISTRATOR)

##  Read access.  Machines listed as allow (and/or not listed as deny)
##  can view the status of your pool, but cannot join your pool
##  or run jobs.
##  NOTE: By default, without these entries customized, you
##  are granting read access to the whole world.  You may want to
##  restrict that to hosts in your domain.  If possible, please also
##  grant read access to "*.cs.wisc.edu", so the Condor developers
##  will be able to view the status of your pool and more easily help
##  you install, configure or debug your Condor installation.
##  It is important to have this defined.
HOSTALLOW_READ = 192.168.19.*
#HOSTALLOW_READ = *.your.domain, *.cs.wisc.edu
#HOSTDENY_READ = *.bad.subnet, bad-machine.your.domain, 144.77.88.*

##  Write access.  Machines listed here can join your pool, submit
##  jobs, etc.  Note: Any machine which has WRITE access must
##  also be granted READ access.  Granting WRITE access below does
##  not also automatically grant READ access; you must change
##  HOSTALLOW_READ above as well.
##  If you leave it as it is, it will be unspecified, and effectively
##  it will be allowing anyone to write to your pool.
HOSTALLOW_WRITE = 192.168.19.*
#HOSTALLOW_WRITE = *.your.domain, your-friend's-machine.other.domain
#HOSTDENY_WRITE = bad-machine.your.domain

##  Negotiator access.  Machines listed here are trusted central
##  managers.  You should normally not have to change this.
HOSTALLOW_NEGOTIATOR = $(NEGOTIATOR_HOST)
##  Now, with flocking we need to let the SCHEDD trust the other
##  negotiators we are flocking with as well.  You should normally
##  not have to change this either.
HOSTALLOW_NEGOTIATOR_SCHEDD = $(NEGOTIATOR_HOST),
$(FLOCK_NEGOTIATOR_HOSTS)

##  Config access.  Machines listed here can use the condor_config_val
##  tool to modify all daemon configurations except those specified in
##  the condor_config.root file.  This level of host-wide access
##  should only be granted with extreme caution.  By default, config
##  access is denied from all hosts.
#HOSTALLOW_CONFIG = trusted-host.your.domain
```

```
##  Flocking Configs.  These are the real things that Condor looks at,
##  but we set them from the FLOCK_FROM/TO macros above.  It is safe
##  to leave these unchanged.
HOSTALLOW_WRITE_COLLECTOR = $(HOSTALLOW_WRITE), $(FLOCK_FROM)
HOSTALLOW_WRITE_STARTD    = $(HOSTALLOW_WRITE), $(FLOCK_FROM)
HOSTALLOW_READ_COLLECTOR  = $(HOSTALLOW_READ), $(FLOCK_FROM)
HOSTALLOW_READ_STARTD     = $(HOSTALLOW_READ), $(FLOCK_FROM)

##--------------------------------------------------------------------
##  Network filesystem parameters:
##--------------------------------------------------------------------
##  Do you want to use NFS for file access instead of remote system
##  calls?
USE_NFS           = True

##  Do your machines run AFS?
HAS_AFS           = False

##  Do you want to use AFS for file access instead of remote system
##  calls?
USE_AFS           = False

##--------------------------------------------------------------------
##  Checkpoint server:
##--------------------------------------------------------------------
##  Do you want to use a checkpoint server if one is available?  If a
##  checkpoint server isn't available or USE_CKPT_SERVER is set to
##  False, checkpoints will be written to the local SPOOL directory on
##  the submission machine.
USE_CKPT_SERVER   = True

##  What's the hostname of this machine's nearest checkpoint server?
CKPT_SERVER_HOST  = maquina1.upct.es

##  Do you want the starter on the execute machine to choose the
##  checkpoint server?  If False, the CKPT_SERVER_HOST set on
##  the submit machine is used.  Otherwise, the CKPT_SERVER_HOST set
##  on the execute machine is used.  The default is true.
STARTER_CHOOSES_CKPT_SERVER = true

##--------------------------------------------------------------------
##  Miscellaneous:
##--------------------------------------------------------------------
##  Try to save this much swap space by not starting new shadows.
##  Specified in megabytes.
##  RESERVED_SWAP       = 5

##  What's the maximum number of jobs you want a single submit machine
##  to spawn shadows for?
MAX_JOBS_RUNNING  = 200

##  Condor needs to create a few lock files to synchronize access to
##  various log files.  Because of problems we've had with network
##  filesystems and file locking over the years, we HIGHLY recommend
##  that you put these lock files on a local partition on each
##  machine.  If you don't have your LOCAL_DIR on a local partition,
##  be sure to change this entry.  Whatever user (or group) condor is
##  running as needs to have write access to this directory.  If
##  you're not running as root, this is whatever user you started up
##  the condor_master as.  If you are running as root, and there's a
##  condor account, it's probably condor.  Otherwise, it's whatever
```

```
##  you've set in the CONDOR_IDS environment variable.  See the Admin
##  manual for details on this.
LOCK            = /var/lock/condor

##  If you don't use a fully qualified name in your /etc/hosts file
##  (or NIS, etc.) for either your official hostname or as an alias,
##  Condor wouldn't normally be able to use fully qualified names in
##  places that it'd like to.  You can set this parameter to the
##  domain you'd like appended to your hostname, if changing your host
##  information isn't a good option.  This parameter must be set in
##  the global config file (not the LOCAL_CONFIG_FILE from above).
#DEFAULT_DOMAIN_NAME = your.domain.name

##  Condor can be told whether or not you want the Condor daemons to
##  create a core file if something really bad happens.  This just
##  sets the resource limit for the size of a core file.  By default,
##  we don't do anything, and leave in place whatever limit was in
##  effect when you started the Condor daemons.  If this parameter is
##  set and "True", we increase the limit to as large as it gets.  If
##  it's set to "False", we set the limit at 0 (which means that no
##  core files are even created).  Core files greatly help the Condor
##  developers debug any problems you might be having.
CREATE_CORE_FILES = False


##  Part 3:  Settings control the policy for running, stopping, and
##  periodically checkpointing condor jobs:
##########################################################################
##########################################################################

##  This section contains macros are here to help write legible
##  expressions:
MINUTE            = 60
HOUR        = (60 * $(MINUTE))
StateTimer  = (CurrentTime - EnteredCurrentState)
ActivityTimer     = (CurrentTime - EnteredCurrentActivity)
ActivationTimer = (CurrentTime - JobStart)
LastCkpt    = (CurrentTime - LastPeriodicCheckpoint)

##  The JobUniverse attribute is just an int.  These macros can be
##  used to specify the universe in a human-readable way:
STANDARD    = 1
PVM         = 4
VANILLA           = 5
IsPVM           = (JobUniverse == $(PVM))
IsVanilla       = (JobUniverse == $(VANILLA))
IsStandard      = (JobUniverse == $(STANDARD))

NonCondorLoadAvg  = (LoadAvg - CondorLoadAvg)
BackgroundLoad          = 0.3
HighLoad          = 0.5
StartIdleTime           = 15 * $(MINUTE)
ContinueIdleTime  =  5 * $(MINUTE)
MaxSuspendTime          = 10 * $(MINUTE)
MaxVacateTime           = 10 * $(MINUTE)

KeyboardBusy            = (KeyboardIdle < $(MINUTE))
ConsoleBusy       = (ConsoleIdle  < $(MINUTE))
CPU_Idle          = ($(NonCondorLoadAvg) <= $(BackgroundLoad))
CPU_Busy          = ($(NonCondorLoadAvg) >= $(HighLoad))

BigJob            = (ImageSize >= (50 * 1024))
```

```
MediumJob   = (ImageSize >= (15 * 1024) && ImageSize < (50 * 1024))
SmallJob    = (ImageSize <  (15 * 1024))

JustCPU                 = ($(CPU_Busy) && ($(KeyboardBusy) == False))
MachineBusy       = ($(CPU_Busy) || $(KeyboardBusy))

##  The RANK expression controls which jobs this machine prefers to
##  run over others.  Some examples from the manual include:
##    RANK = ImageSize
##    RANK = (Owner == "coltrane") + (Owner == "tyner") \
##                 + ((Owner == "garrison") * 10) + (Owner ==
"jones")
##  By default, RANK is always 0, meaning that all jobs have an equal
##  ranking.
RANK              = 0

##########################################################################
##  This is a Configuration that will cause your Condor jobs to
##  always run.  This is intended for testing only.
##########################################################################

##  This mode will cause your jobs to start on a machine an will let
##  them run to completion.  Condor will ignore all of what is going
##  on in the machine (load average, keyboard activity, etc.)

TESTINGMODE_WANT_SUSPEND      = False
TESTINGMODE_WANT_VACATE       = False
TESTINGMODE_START       = True
TESTINGMODE_SUSPEND           = True
TESTINGMODE_CONTINUE          = True
TESTINGMODE_PREEMPT           = True
TESTINGMODE_KILL        = False
TESTINGMODE_PERIODIC_CHECKPOINT     = True
TESTINGMODE_PREEMPTION_REQUIREMENTS = True
TESTINGMODE_PREEMPTION_RANK = 0

##  Part4:  Settings you should probably leave alone:
##  (unless you know what you're doing)
##########################################################################
##########################################################################

##########################################################################
##  Daemon-wide settings:
##########################################################################

##  Pathnames
LOG         = $(LOCAL_DIR)/log
SPOOL       = $(LOCAL_DIR)/spool
EXECUTE           = $(LOCAL_DIR)/execute
BIN         = $(RELEASE_DIR)/bin
LIB         = $(RELEASE_DIR)/lib
SBIN        = $(RELEASE_DIR)/sbin
HISTORY           = $(SPOOL)/history

##  Log files
COLLECTOR_LOG     = $(LOG)/CollectorLog
KBDD_LOG    = $(LOG)/KbdLog
MASTER_LOG  = $(LOG)/MasterLog
NEGOTIATOR_LOG    = $(LOG)/NegotiatorLog
SCHEDD_LOG  = $(LOG)/SchedLog
SHADOW_LOG  = $(LOG)/ShadowLog
STARTD_LOG  = $(LOG)/StartLog
```

```
STARTER_LOG = $(LOG)/StarterLog

##  Lock files
SHADOW_LOCK = $(LOCK)/ShadowLock

##  In most cases, your condor_collector and condor_negotiator are
##  going to run on the same machine.  If for some reason, this isn't
##  the case, here's where you'd change it:
COLLECTOR_HOST    = $(CONDOR_HOST)
NEGOTIATOR_HOST   = $(CONDOR_HOST)

##  How long are you willing to let daemons try their graceful
##  shutdown methods before they do a hard shutdown? (30 minutes)
SHUTDOWN_GRACEFUL_TIMEOUT     = 1800

##  How much disk space would you like reserved from Condor?  In
##  places where Condor is computing the free disk space on various
##  partitions, it subtracts the amount it really finds by this
##  many megabytes.  (If undefined, defaults to 0).
RESERVED_DISK           = 5

##  If your machine is running AFS and the AFS cache lives on the same
##  partition as the other Condor directories, and you want Condor to
##  reserve the space that your AFS cache is configured to use, set
##  this to true.
RESERVE_AFS_CACHE = False

##########################################################################
##  Daemon-specific settings:
##########################################################################
##--------------------------------------------------------------------
##  condor_master
##--------------------------------------------------------------------
##  Daemons you want the master to keep running for you:
DAEMON_LIST             = MASTER, STARTD, SCHEDD

##  Which daemons use the Condor DaemonCore library (i.e., not the
##  checkpoint server or custom user daemons)?
DC_DAEMON_LIST = MASTER, STARTD, SCHEDD, KBDD, COLLECTOR, NEGOTIATOR,
EVENTD

##  Where are the binaries for these daemons?
MASTER                          = $(SBIN)/condor_master
STARTD                          = $(SBIN)/condor_startd
SCHEDD                          = $(SBIN)/condor_schedd
KBDD                    = $(SBIN)/condor_kbdd

##  When the master starts up, it can place it's address (IP and port)
##  into a file.  This way, tools running on the local machine don't
##  need to query the central manager to find the master.  This
##  feature can be turned off by commenting out this setting.
MASTER_ADDRESS_FILE = $(LOG)/.master_address

##  Where should the master find the condor_preen binary? If you don't
##  want preen to run at all, just comment out this setting.
PREEN                   = $(SBIN)/condor_preen

##  How do you want preen to behave?  The "-m" means you want email
##  about files preen finds that it thinks it should remove.  The "-r"
##  means you want preen to actually remove these files.  If you don't
##  want either of those things to happen, just remove the appropriate
##  one from this setting.
```

```
PREEN_ARGS              = -m -r

##  How often should the master start up condor_preen? (once a day)
PREEN_INTERVAL              = 86400

##  If a daemon dies an unnatural death, do you want email about it?
PUBLISH_OBITUARIES          = True

##  If you're getting obituaries, how many lines of the end of that
##  daemon's log file do you want included in the obituary?
OBITUARY_LOG_LENGTH         = 20

##  Should the master run?
START_MASTER                = True

##  Should the master start up the daemons you want it to?
START_DAEMONS               = True

##  How often do you want the master to send an update to the central
##  manager?
MASTER_UPDATE_INTERVAL      = 300

##  How often do you want the master to check the timestamps of the
##  daemons it's running?  If any daemons have been modified, the
##  master restarts them.
MASTER_CHECK_NEW_EXEC_INTERVAL      = 300

##  Once you notice new binaries, how long should you wait before you
##  try to execute them?
MASTER_NEW_BINARY_DELAY = 120

##  What's the maximum amount of time you're willing to give the
##  daemons to quickly shutdown before you just kill them outright?
SHUTDOWN_FAST_TIMEOUT       = 120

######
##  Exponential backoff settings:
######
##  When a daemon keeps crashing, we use "exponential backoff" so we
##  wait longer and longer before restarting it.  This is the base of
##  the exponent used to determine how long to wait before starting
##  the daemon again:
MASTER_BACKOFF_FACTOR       = 2.0

##  What's the maximum amount of time you want the master to wait
##  between attempts to start a given daemon?  (With 2.0 as the
##  MASTER_BACKOFF_FACTOR, you'd hit 1 hour in 12 restarts...)
MASTER_BACKOFF_CEILING      = 3600

##  How long should a daemon run without crashing before we consider
##  it "recovered".  Once a daemon has recovered, we reset the number
##  of restarts so the exponential backoff stuff goes back to normal.
MASTER_RECOVER_FACTOR       = 300
##--------------------------------------------------------------------
##  condor_startd
##--------------------------------------------------------------------
##  Where are the various condor_starter binaries installed?
STARTER                 = $(SBIN)/condor_starter
ALTERNATE_STARTER_1     = $(SBIN)/condor_starter.pvm

##  When the startd starts up, it can place it's address (IP and port)
##  into a file.  This way, tools running on the local machine don't
```

```
##  need to query the central manager to find the startd.  This
##  feature can be turned off by commenting out this setting.
STARTD_ADDRESS_FILE     = $(LOG)/.startd_address

##  When a machine is claimed, how often should we poll the state of
##  the machine to see if we need to evict/suspend the job, etc?
POLLING_INTERVAL          = 5

##  How often should the startd send updates to the central manager?
UPDATE_INTERVAL           = 300

##  How long is the startd willing to stay in the "matched" state?
MATCH_TIMEOUT             = 300

##  How long is the startd willing to stay in the preempting/killing
##  state before it just kills the starter directly?
KILLING_TIMEOUT   = 30

##  When a machine unclaimed, when should it run benchmarks?
##  LastBenchmark is initialized to 0, so this expression says as soon
##  as we're unclaimed, run the benchmarks.  Thereafter, if we're
##  unclaimed and it's been at least 4 hours since we ran the last
##  benchmarks, run them again.  The startd keeps a weighted average
##  of the benchmark results to provide more accurate values.
##  Note, if you don't want any benchmarks run at all, either comment
##  RunBenchmarks out, or set it to "False".
BenchmarkTimer = (CurrentTime - LastBenchmark)
RunBenchmarks : (LastBenchmark == 0 ) || ($(BenchmarkTimer) >= (4 *
$(HOUR)))
RunBenchmarks : True

##  Normally, when the startd is computing the idle time of all the
##  users of the machine (both local and remote), it checks the utmp
##  file to find all the currently active ttys, and only checks access
##  time of the devices associated with active logins.  Unfortunately,
##  on some systems, utmp is unreliable, and the startd might miss
##  keyboard activity by doing this.  So, if your utmp is unreliable,
##  set this setting to True and the startd will check the access time
##  on all tty and pty devices.
STARTD_HAS_BAD_UTMP = True

##  This entry allows the startd to monitor console (keyboard and
##  mouse) activity by checking the access times on special files in
##  /dev.  Activity on these files shows up as "ConsoleIdle" time in
##  the startd's ClassAd.  Just give a comma-separated list of the
##  names of devices you want considered the console, without the
##  "/dev/" portion of the pathname.
CONSOLE_DEVICES   = mouse, console

##  The STARTD_EXPRS entry allows you to have the startd advertise
##  arbitrary expressions from the config file in its ClassAd.  Give
##  the comma-separated list of entries from the config file you want
##  in the startd ClassAd.
##  Note: because of the different syntax of the config file and
##  ClassAds, you might have to do a little extra work to get a given
##  entry into the ClassAd.  In particular, ClassAds require "'s
##  around your strings.  Numeric values can go in directly, as can
##  boolean expressions.  For example, if you wanted the startd to
##  advertise its list of console devices, when it's configured to run
##  benchmarks, and how often it sends updates to the central manager,
##  you'd have to define the following helper macro:
MY_CONSOLE_DEVICES = "$(CONSOLE_DEVICES)"
```

```
##  Note: this must come before you define STARTD_EXPRS because macros
##  must be defined before you use them in other macros or
##  expressions.
##  Then, you'd set the STARTD_EXPRS setting to this:
STARTD_EXPRS = MY_CONSOLE_DEVICES, RunBenchmarks, UPDATE_INTERVAL

##  When the startd is claimed by a remote user, it can also advertise
##  arbitrary attributes from the ClassAd of the job its working on.
##  Just list the attribute names you want advertised.
##  Note: since this is already a ClassAd, you don't have to do
##  anything funny with strings, etc.  This feature can be turned off
##  by commenting out this setting (there is no default).
STARTD_JOB_EXPRS = ImageSize, JobUniverse

##  If you want to "lie" to Condor about how many CPUs your machine
##  has, you can use this setting to override Condor's automatic
##  computation.  If you modify this, you must restart the startd for
##  the change to take effect (a simple condor_reconfig will not do).
##  Please read the section on "condor_startd Configuration File
##  Macros" in the Condor Administrators Manual for a further
##  discussion of this setting.  Its use is not recommended.  This
##  must be an integer ("N" isn't a valid setting, that's just used to
##  represent the default).
#NUM_CPUS = N
##--------------------------------------------------------------------
##  condor_schedd
##--------------------------------------------------------------------
##  Where are the various shadow binaries installed?
SHADOW                  = $(SBIN)/condor_shadow
SHADOW_PVM        = $(SBIN)/condor_shadow.pvm
SHADOW_GLOBUS           = $(SBIN)/condor_shadow.globus
SHADOW_NT         = $(SBIN)/condor_shadow.v61

##  When the schedd starts up, it can place it's address (IP and port)
##  into a file.  This way, tools running on the local machine don't
##  need to query the central manager to find the schedd.  This
##  feature can be turned off by commenting out this setting.
SCHEDD_ADDRESS_FILE     = $(LOG)/.schedd_address

##  How often should the schedd send an update to the central manager?
SCHEDD_INTERVAL   = 300

##  How long should the schedd wait between spawning each shadow?
JOB_START_DELAY   = 2

##  How often should the schedd send a keep alive message to any
##  startds it has claimed?  (5 minutes)
ALIVE_INTERVAL          = 300

##  This setting controls the maximum number of times that a
##  condor_shadow processes can have a fatal error (exception) before
##  the condor_schedd will simply relinquish the match associated with
##  the dying shadow.
MAX_SHADOW_EXCEPTIONS   = 5

##  Estimated virtual memory size of each condor_shadow process.
##  Specified in kilobytes.
SHADOW_SIZE_ESTIMATE    = 1800

##  The condor_schedd can renice the condor_shadow processes on your
##  submit machines.  How how "nice" do you want the shadows? (1-19).
##  The higher the number, the lower priority the shadows have.
```

```
##  This feature can be disabled entirely by commenting it out.
SHADOW_RENICE_INCREMENT = 10

##  By default, when the schedd fails to start an idle job, it will
##  not try to start any other idle jobs in the same cluster during
##  that negotiation cycle.  This makes negotiation much more
##  efficient for large job clusters.  However, in some cases other
##  jobs in the cluster can be started even though an earlier job
##  can't.  For example, the jobs' requirements may differ, because of
##  different disk space, memory, or operating system requirements.
##  Or, machines may be willing to run only some jobs in the cluster,
##  because their requirements reference the jobs' virtual memory size
##  or other attribute.  Setting NEGOTIATE_ALL_JOBS_IN_CLUSTER to True
##  will force the schedd to try to start all idle jobs in each
##  negotiation cycle.  This will make negotiation cycles last longer,
##  but it will ensure that all jobs that can be started will be
##  started.
NEGOTIATE_ALL_JOBS_IN_CLUSTER = True

######
## Queue management settings:
######
##  How often should the schedd truncate it's job queue transaction
##  log?  (Specified in seconds, once a day is the default.)
QUEUE_CLEAN_INTERVAL    = 86400

##  How often should the schedd commit "wall clock" run time for jobs
##  to the queue, so run time statistics remain accurate when the
##  schedd crashes?  (Specified in seconds, once per hour is the
##  default.  Set to 0 to disable.)
WALL_CLOCK_CKPT_INTERVAL = 3600

##  Do you want to allow remote machines to be able to submit jobs to
##  this queue as user "nobody"?
ALLOW_REMOTE_SUBMIT     = True

##  What users do you want to grant super user access to this job
##  queue?  (These users will be able to remove other user's jobs).
##  By default, this only includes root.
QUEUE_SUPER_USERS = root, condor
##--------------------------------------------------------------------
##  condor_shadow
##--------------------------------------------------------------------
##  If the shadow is unable to read a checkpoint file from the
##  checkpoint server, it keeps trying only if the job has accumulated
##  more than MAX_DISCARDED_RUN_TIME seconds of CPU usage.  Otherwise,
##  the job is started from scratch.  Defaults to 1 hour.  This
##  setting is only used if USE_CKPT_SERVER (from above) is True.
MAX_DISCARDED_RUN_TIME = 3600

##  Should periodic checkpoints be compressed?
COMPRESS_PERIODIC_CKPT = False

##  Should vacate checkpoints be compressed?
COMPRESS_VACATE_CKPT = False

##  Should we commit the application's dirty memory pages to swap
##  space during a periodic checkpoint?
PERIODIC_MEMORY_SYNC = False

##  Should we write vacate checkpoints slowly?  If nonzero, this
##  parameter specifies the speed at which vacate checkpoints should
```

```
##  be written, in kilobytes per second.
SLOW_CKPT_SPEED = 0


##--------------------------------------------------------------------
##  condor_shadow.pvm
##--------------------------------------------------------------------
##  Where is the condor pvm daemon installed?
PVMD                = $(SBIN)/condor_pvmd

##  Where is the condor pvm group server daemon installed?
PVMGS               = $(SBIN)/condor_pvmgs

##--------------------------------------------------------------------
##  condor_starter
##--------------------------------------------------------------------
##  The condor_starter can renice the processes from remote Condor
##  jobs on your execute machines.  If you want this, uncomment the
##  following entry and set it to how "nice" do you want the user
##  jobs. (1-19)  The larger the number, the lower priority the
##  process gets on your machines.
JOB_RENICE_INCREMENT    = 10

##  Should the starter do local logging to its own log file, or send
##  debug information back to the condor_shadow where it will end up
##  in the ShadowLog?
STARTER_LOCAL_LOGGING   = TRUE

##  If the UID_DOMAIN settings match on both the execute and submit
##  machines, but the UID of the user who submitted the job isn't in
##  the passwd file of the execute machine, the starter will normally
##  exit with an error.  Do you want the starter to just start up the
##  job with the specified UID, even if it's not in the passwd file?
SOFT_UID_DOMAIN   = FALSE

##--------------------------------------------------------------------
##  condor_submit
##--------------------------------------------------------------------
##  the job ClassAds it creates, you can uncomment and define the
##  SUBMIT_EXPRS setting.  It works just like the STARTD_EXPRS
##  described above with respect to ClassAd vs. config file syntax,
##  strings, etc.  One common use would be to have the full hostname
##  of the machine where a job was submitted placed in the job
##  ClassAd.  You would do this by uncommenting the following lines:
MACHINE = "$(FULL_HOSTNAME)"
SUBMIT_EXPRS = MACHINE

## Condor keeps a buffer of recently-used data for each file an
## application opens.  This macro specifies the default maximum number
## of bytes to be buffered for each open file at the executing
## machine.
DEFAULT_IO_BUFFER_SIZE = 524288

## Condor will attempt to consolidate small read and write operations
## into large blocks.  This macro specifies the default block size
## Condor will use.
DEFAULT_IO_BUFFER_BLOCK_SIZE = 32768

##--------------------------------------------------------------------
##  condor_preen
##--------------------------------------------------------------------
##  Who should condor_preen send email to?
```

```
PREEN_ADMIN       = $(CONDOR_ADMIN)

##  What files should condor_preen leave in the spool directory?
VALID_SPOOL_FILES = job_queue.log, job_queue.log.tmp, history, \
                          Accountant.log, Accountantnew.log

##  What files should condor_preen remove from the log directory?
INVALID_LOG_FILES = core
```

Fichero de configuración local

Una vez modificados los parámetros del fichero de configuración global del Pool, se deben modificar los parámetros del fichero de configuración local del Central Manager. Por tanto, hay que situarse en el fichero de configuración del mismo y cambiar los parámetros necesarios. Tal fichero quedará de la siguiente forma:

maquina1:/usr/local/condor/hosts/maquina1 # emacs condor_config.local

```
##  Every pool can have a name associated with it.  This should be a
##  short description (20 characters or so) that describes your site.
##  For example, the name for the UW-Madison Computer Science Condor
##  Pool is: "UW-Madison CS" (you don't need to put in the " marks).
COLLECTOR_NAME         = PROYECTO-UPCT-CONDOR

##  What daemons do you want to run on your central manager?
##  NOTE: For it to be the central manager, you need the NEGOTIATOR
##  and COLLECTOR to run.  It's optional whether or not you want to
##  run the schedd (to allow jobs to be submited) and/or the startd
##  (to allow Condor jobs to execute) on your central manager.
DAEMON_LIST   = MASTER, COLLECTOR, NEGOTIATOR, STARTD, SCHEDD

##  Where are the binaries for these daemons?  (Note: MASTER, SCHEDD,
##  and STARTD are already defined in the global config file).
COLLECTOR     = $(SBIN)/condor_collector
NEGOTIATOR    = $(SBIN)/condor_negotiator


##############################################################################
##############################################################################
##  Settings you should probably leave alone:
##  (unless you know what you're doing)
##############################################################################
##############################################################################

##--------------------------------------------------------------------
##  condor_collector
##--------------------------------------------------------------------
##  How long can a ClassAd remain in the collector before it is
##  discarded as stale information?  (Defaults to 15 minutes.  The
##  daemons send updates every 5 minutes by default).
CLASSAD_LIFETIME  = 900

##  How often should the collector check for machines that don't have
##  ClassAds from the condor_master and send email about it? (3
##  hours by default).
MASTER_CHECK_INTERVAL   = 10800

##  Network timeout when talking to daemons that are sending an
##  update:
CLIENT_TIMEOUT          = 30
```

```
##  Network timeout when talking to anyone doing a query:
QUERY_TIMEOUT           = 60

##  Email address of the condor-developers at UW-Madison.
#CONDOR_DEVELOPERS      = condor-admin@cs.wisc.edu

##  By default, every pool sends periodic updates to a central
##  condor_collector at UW-Madison with basic information about the
##  status of your pool.  This includes only the number of total
##  machines, the number of jobs submitted, the number of machines
##  running jobs, the hostname of your central manager, and the
##  "COLLECTOR_NAME" specified above.  These updates help us see
##  how Condor is being used around the world.  By default, they will
##  be sent to condor.cs.wisc.edu.  If you don't want these updates to
##  be sent from your pool, uncomment the following line.
CONDOR_DEVELOPERS_COLLECTOR = NONE

##  Enable history logging in the collector
#KEEP_POOL_HISTORY = True

##  Set the directory where history files reside
#POOL_HISTORY_DIR = $(SPOOL)

###  The maximum combined size of the history files
#POOL_HISTORY_MAX_STORAGE = 10000000

##  The interval (in seconds) between samples for history logging
#POOL_HISTORY_SAMPLING_INTERVAL = 60

##--------------------------------------------------------------------
##  condor_negotiator
##--------------------------------------------------------------------
##  How often should the negotiator start a negotiation cycle?
NEGOTIATOR_INTERVAL     = 300

##  What timeout should the negotiator use on it's network connections
##  to the schedds and startds?
NEGOTIATOR_TIMEOUT      = 30

##  What is the half-life of the user priorities? (1 day)
PRIORITY_HALFLIFE = 86400

##  The default priority factor for nice users
NICE_USER_PRIO_FACTOR = 10000000

##  The UID_DOMAIN of users who are considered local.  If undefined,
##  all users are considered local.  Remote (non-local) users get a
##  configurable boost to their priority value (giving them worse
##  priority in your pool).  See below.
ACCOUNTANT_LOCAL_DOMAIN = $(UID_DOMAIN)

##  The default priority factor for remote users (only used if
##  ACCOUNTANT_LOCAL_DOMAIN is defined above).
REMOTE_PRIO_FACTOR = 10000

##  The NEGOTIATOR_TRAFFIC_LIMIT macro specifies the maximum
##  amount of network traffic (in KB) the negotiator may initiate per
##  NEGOTIATOR_TRAFFIC_INTERVAL for job placement and preemption.
##  The negotiator uses the job ImageSize and ExecutableSize
##  parameters to track network usage.  The negotiator will try to use
##  bandwidth up to the limit, so if starting a large ImageSize job
```

```
##  would put the negotiator over the limit, it will try to start a
##  small ImageSize job in its place.  Thus, using traffic limits
##  penalizes large ImageSize jobs for the load they place on the
##  network.  This parameter defaults to 0, which disables network
##  usage management in the negotiator.
NEGOTIATOR_TRAFFIC_LIMIT = 300000

##  The NEGOTIATOR_TRAFFIC_INTERVAL macro specifies the interval
##  (in seconds) to be used in maintaining the
##  NEGOTIATOR_TRAFFIC_LIMIT.  This parameter defaults to 0, which
##  disables network usage management in the negotiator.  It is common
##  to set this parameter equal to NEGOTIATOR_INTERVAL.
NEGOTIATOR_TRAFFIC_INTERVAL = $(NEGOTIATOR_INTERVAL)

##  The NEGOTIATOR_SOCKET_CACHE_SIZE macro defines the maximum number
##  of sockets which the negotiator should keep in its open socket
##  cache.  Caching open sockets makes the negotiation protocol more
##  efficient by eliminating the need for socket connection
##  establishment for each negotiation cycle.  The default is
##  currently 16.  To be effective, this parameter should be set to a
##  value greater than the number of schedds submitting jobs to the
##  negotiator at any time.
NEGOTIATOR_SOCKET_CACHE_SIZE = 16
```

4.9.2 Máquinas del Condor Pool

Una vez instalado y configurado el Central Manager, se debe instalar Condor también en cada una de las máquinas que van a formar parte del Condor Pool (en el caso de esta red, la máquina 2 y la máquina 3). Por tanto, se ejecuta en estas máquinas el script condor_install para comenzar la instalación, al igual que se hizo para el Central Manager.

maquina2:~ /software/inscondor/condor-6.2.1 # perl condor_install

maquina3:~//software/inscondor/condor-6.2.1 # perl condor_install

Los pasos a seguir en ambas máquinas serán los mismos que para el Central Manager, exceptuando el siguiente paso del script de instalación:

```
************************************************************************
*****
        STEP 7: What machine will be your central manager?
************************************************************************
*****

What is the full hostname of the central manager?
[maquina2.upct.es]maquina1.upct.es
```

De esta manera quedara establecido en ambas máquinas que la máquina 1 es el Central Manager, y por tanto, los procesos que se ejecuten en maquina2 y maquina3 deberán interactuar con la maquina1 para la someter y ejecutar tareas, como se vio en el capítulo 2.

Como en el caso del Central Manager, se debe ejecutar condor_init en cada una de las máquinas para crear los directorios LOCK (punto 4.2, 4.3) en el disco local de cada máquina.

maquina2: # /usr/local/condor/sbin/condor_init

maquina3:# /usr/local/condor/sbin/condor_init

Al igual que para el Central Manager, se deberá modificar el fichero de configuración global en las máquinas 2 y 3. En este caso, estas máquinas no tendrán un fichero de configuración local para modificar, como en el caso del Central Manager. La modificación de dicho fichero será la misma que la del fichero de configuración global del Central Manager.

Además, se deberán seguir los mismos pasos que para el Central Manager, cambiando el sistema compilador, realizando los enlaces simbólicos necesarios… (4.8.1.2 Configuración).

A partir de este momento ya se ha instalado y configurado el software de Condor en la red prototipo y esta todo listo para empezar a ejecutar la aplicación. Esto se verá en el siguiente capítulo.

Capítulo 5

Universo Standard en una red. Ejecución y obtención de resultados

5.1 Ejecución del Condor Pool

Una vez que se tiene instalado y configurado el software de Condor en la red prototipo, se procede a la ejecución de los procesos Condor estableciendo así el Condor Pool y comprobando el funcionamiento de lo configurado por medio de un programa ejemplo.

5.1.1 Ejecución de los demonios

El primer proceso que se debe ejecutar es el condor_master, ya que va a ser el proceso que lance a los demás procesos que deben ejecutarse en la máquina dependiendo del rol que se juegue. Por tanto se ejecuta condor_master en todas las máquinas, empezando siempre por el Central Manager.

maquina1:~ # /usr/local/condor/sbin/condor_master

maquina2:~ # /usr/local/condor/sbin/condor_master

maquina3:~ # /usr/local/condor/sbin/condor_master

5.1.2 Resultados obtenidos

Una vez que se lanza el condor_master, por lo que se vio en el capítulo 2 sobre el funcionamiento de los procesos de Condor este proceso, dependiendo del rol que se juegue en cada máquina, lanzará a unos u otros procesos para poder desempeñar el papel para el que ha sido configurado.

Para verificar el comportamiento que ha adoptado cada una de las máquinas de la red prototipo, se deben visualizar los procesos Condor que se encuentran ejecutándose en cada máquina. Para ello se debe ejecutar en la línea de comandos de cada máquina el siguiente comando:

"ps –ef | egrep condor_" o "ps –aux | egrep condor_".

Para la configuración realizada, hay que destacar que el Central Manager, además de los procesos característicos de esta máquina como condor_collector y condor_negotiator, también puede someter y ejecutar tareas, ya que en la máquina se ejecutan los procesos condor_startd y condor_schedd.

En la siguiente figura se pueden observar los procesos que se encuentran ejecutándose en el Central Manager.

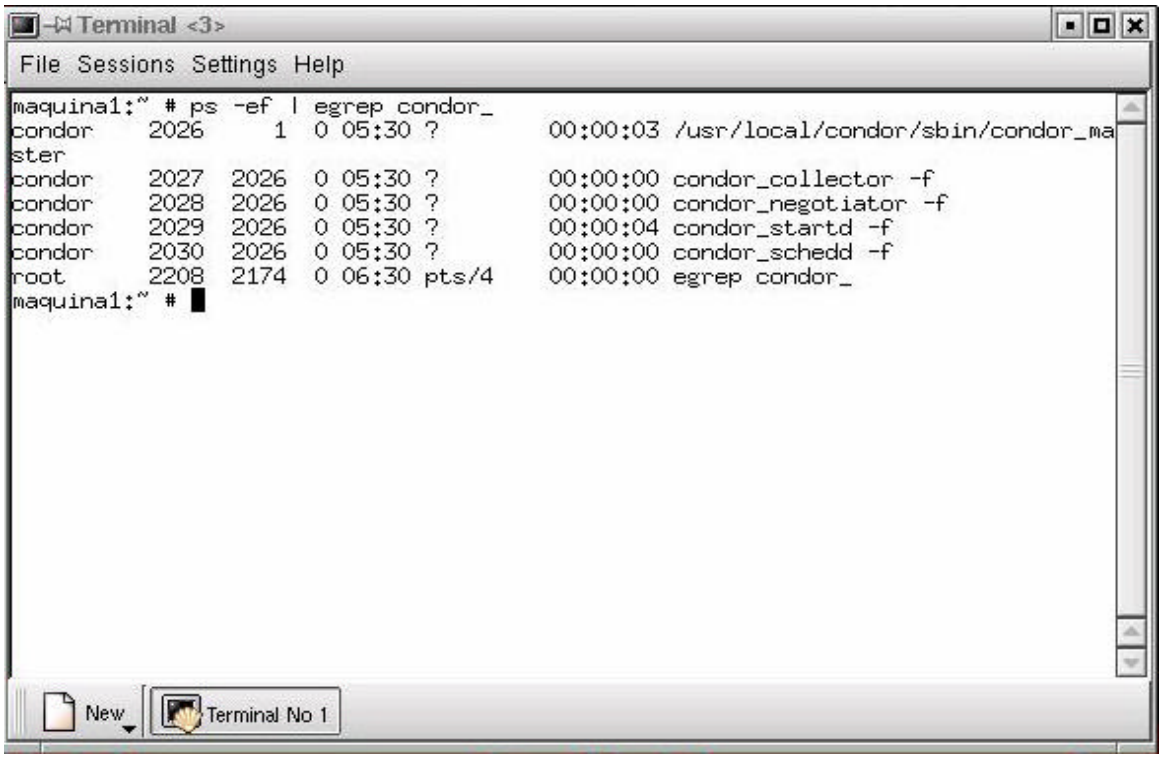

Figura 5. 1 Procesos en el Central Manager

En la siguiente figura se muestran los procesos que se ejecutan cuando se lanza el condor_master en la máquina 2 y la máquina 3 pertenecientes al Pool:

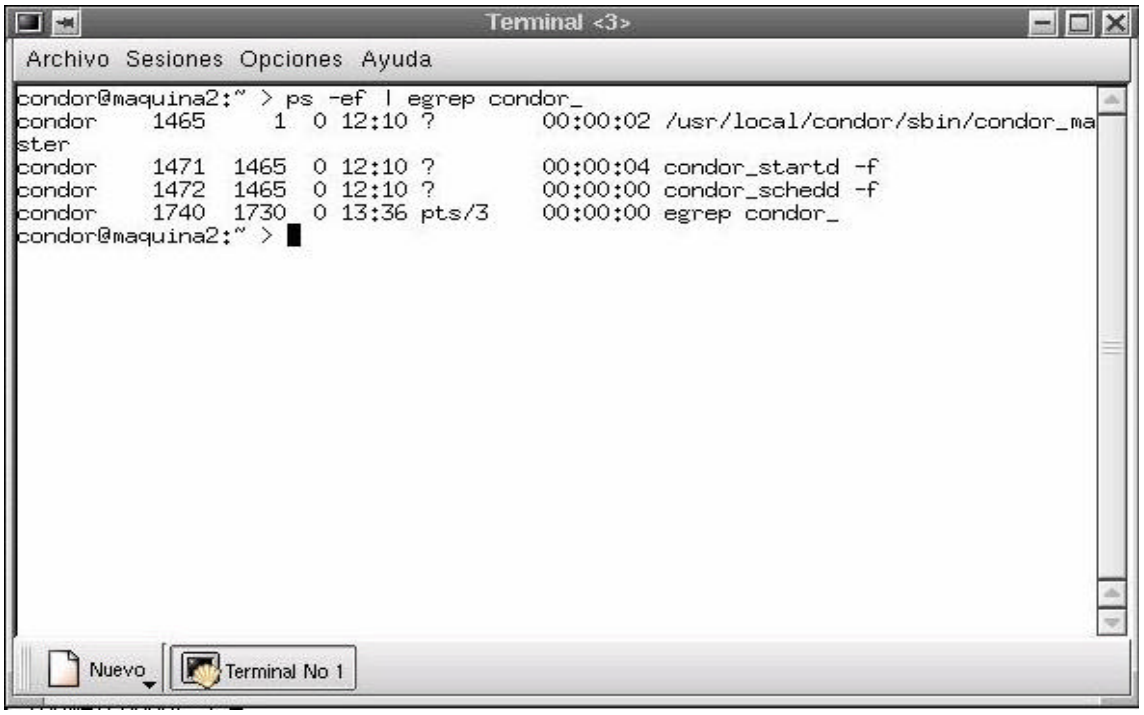

Figura 5. 2 Procesos en las máquinas Submit-Execute

Se puede observar que en la figura anterior, la máquina 2 no esta trabajando como usuario privilegiado Root, sino como usuario Condor. Con esto se demuestra que cualquier usuario de la máquina 2 puede trabajar bajo la Aplicación Condor, como se verá en los siguientes apartados.

Al iniciarse el Condor Pool y aunque no se hayan sometido aún tareas en el Condor Pool, se pueden ejecutar algunos comandos que proporciona la aplicación Condor para obtener información sobre el Pool.

Condor posee un comando, "condor_status", que proporciona información sobre las máquinas que forman parte del Condor Pool en el momento en que se ejecuta dicho comando.

La información que se puede obtener se muestra en la siguiente figura, donde se puede ver desde el Central Manager (máquina 1) las máquinas que forman parte del Pool, el sistema operativo de cada una de ellas (OpSys), la arquitectura de cada una de las máquinas (Arch), el estado en el que se encuentran (State) (capítulo 2), la actividad en la que se hayan (Activity) (capítulo 2), la memoria de cada una de ellas (Mem), el tiempo de actividad de las máquinas (ActivityTime)...

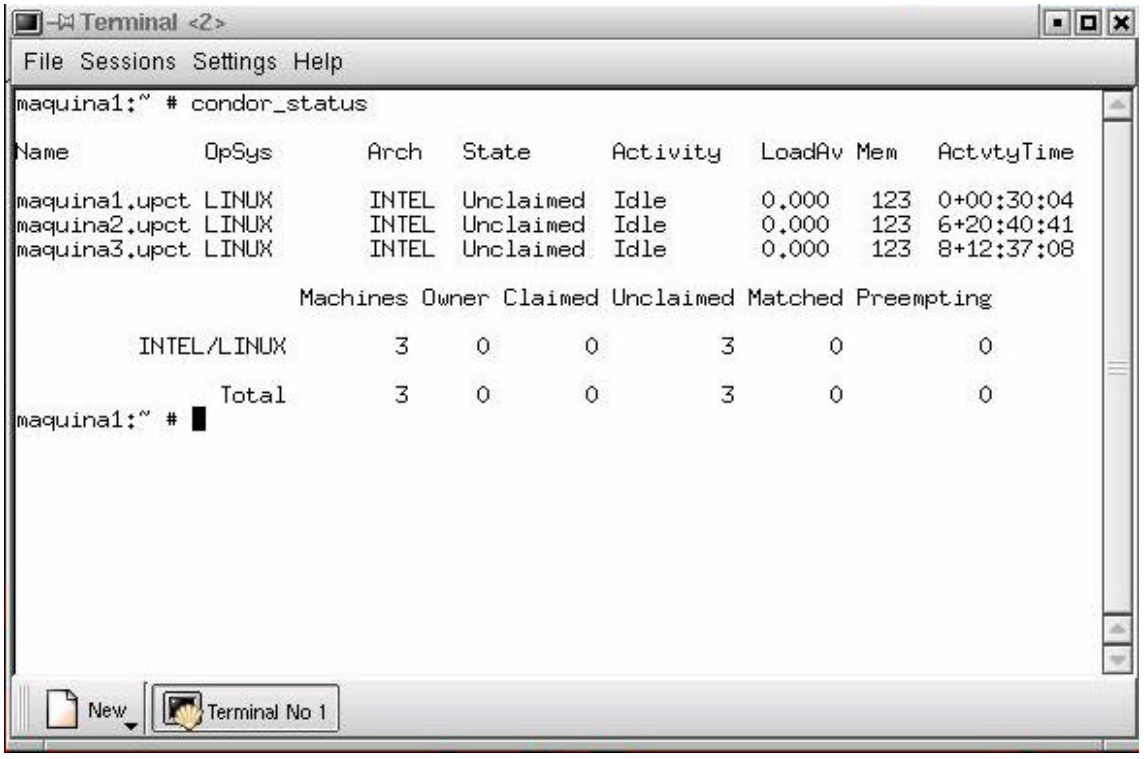

Figura 5. 3 Estado del Condor Pool

Condor también proporciona otros comandos, como por ejemplo "condor_q" que proporciona información sobre la cola de tareas de la máquina donde se ejecuta dicho comando. Si se ejecuta este comando, por ejemplo, en la máquina 1, se puede obtener la siguiente información:

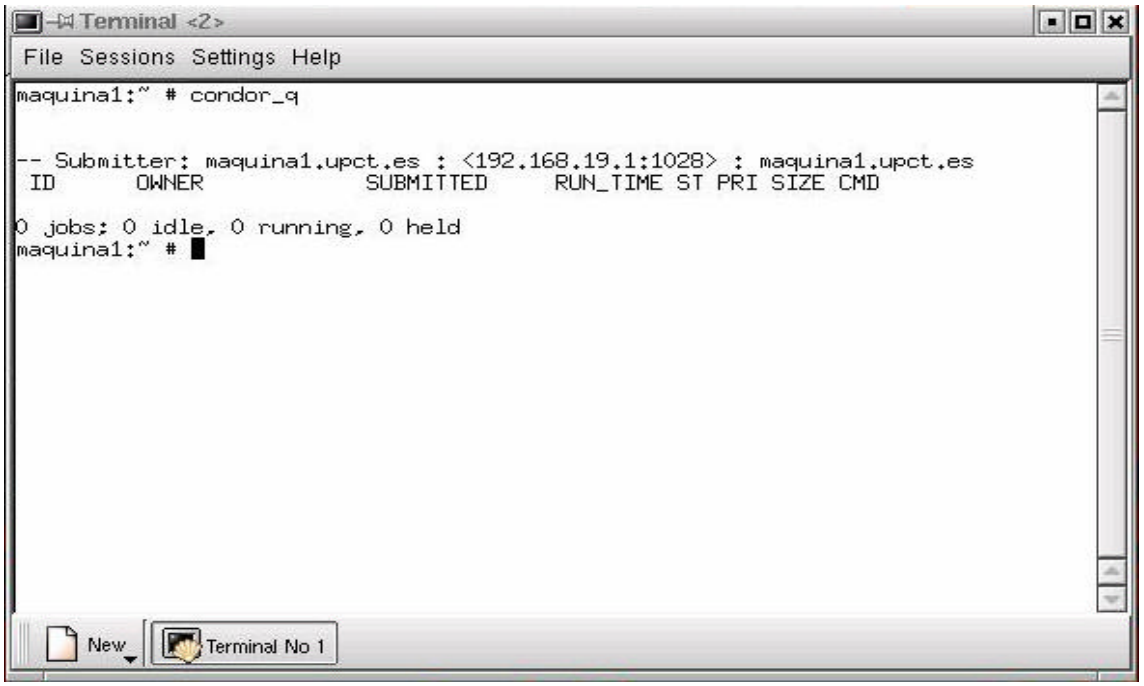

Figura 5. 4 Estado de la cola de la máquina

Como en un principio no se ha sometido ninguna tarea, la cola de tareas aparece vacía.

Una vez comprobada la interconexión de las tres máquinas a través del Condor Pool mediante la ejecución de diferentes comandos de Condor para verificar el funcionamiento, se debe de instalar y configurar el Servidor Checkpoint para el perfecto funcionamiento del Pool a la hora de someter tareas, como se explicó en el capítulo 3.

5.2 Servidor Checkpoint

Como se explicó en el capítulo 3, la figura Checkpoint es imprescindible para la configuración de un Condor Pool con el Universo Standard. A continuación se explica la instalación y configuración del Servidor Checkpoint en la red propuesta. La instalación y puesta en marcha de este elemento facilitará la recuperación de las tareas en caso de interrupción y un continuo seguimiento de las mismas mientras están ejecutándose.

5.2.1 Instalación

Se instala el módulo de checkpoint más apropiado a las características de la aplicación Condor instalada en la red. Como la versión de Condor instalada en el Condor Pool es la versión 6.2.1, el módulo a instalar será ckpt_server-6.2.1-linux-x86-glibc21.tar.gz que se puede encontrar en la página web http://www.cs.wisc.edu/condor/downloads/.

Una vez descargado, se debe descomprimir el .tar.gz como se observa en la siguiente captura de pantalla.

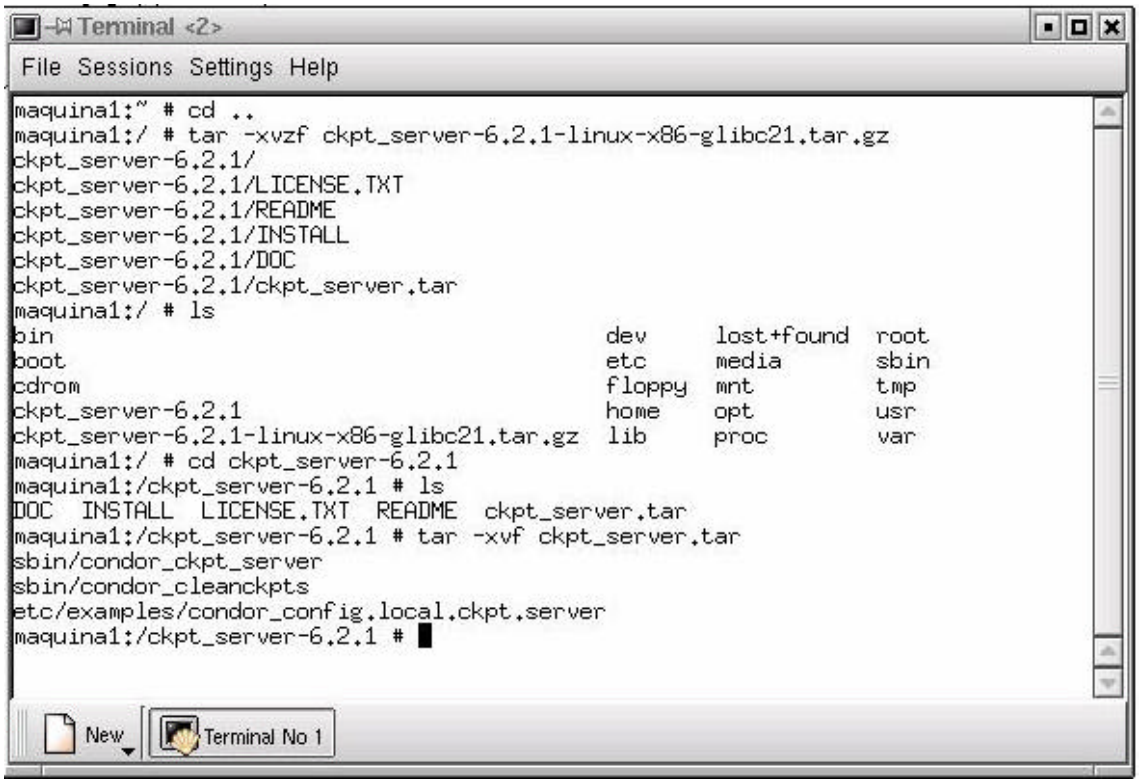

Figura 5. 5 Software del Servidor Checkpoint

Al descomprimir se creará un directorio llamado “ckpt_server-6.2.1”, del que cuelgan los siguientes ficheros:

- License.txt La licencia de Condor
- Readme Información general sobre Condor.
- Install Direcciones de instalación.
- Doc Direcciones donde se puede encontrar documentación de Condor.
- ckpt_server.tar Fichero tar que contiene las librerías y binarios del servidor Checkpoint.

Situándose dentro de dicho directorio, se debe de descomprimir el fichero ckpt_server.tar, donde se crearán dos directorios, como se ve en la captura de pantalla anterior. Dentro de estos directorios se encuentran los procesos que intervendrán en la ejecución del Servidor Checkpoint (condor_ckpt_server, condor_cleanckpts) y el fichero de configuración del Servidor.

Los pasos para la correcta instalación del Servidor Checkpoint serán:

- Configurar el Servidor Checkpoint.
- Ejecutar el servidor Checkpoint.
- Configurar el Condor Pool para usar el servidor Checkpoint.

En la máquina que va a funcionar como Central Manager y Servidor Checkpoint se debe de modificar tanto el fichero de configuración global condor_config, como el fichero de configuración local de la máquina condor_config.local.

En el fichero de configuración global se modificarán las siguientes macros(si no se han modificado antes en la configuración del pool) estableciéndolas como se indica a continuación.

5.2.2 Configuración

Para la configuración deseada del Servidor Checkpoint en cuanto a la red que se propone se debe modificar el fichero de configuración del Servidor Checkpoint que se encuentra dentro del directorio “examples” (ver apéndice).

5.2.2.1 Fichero de configuración local de checkpoint

Se abre el fichero de configuración del Servidor Checkpoint.

maquina1: /ckpt_server-6.2.1/etc/examples # emacs condor_config.local.ckpt.server

A continuación se muestra el fichero de configuración del Servidor Checkpoint. Las macros que se han modificado para la configuración del Condor Pool se muestran a continuación.

```
##  In what directory should the checkpoint server store checkpoint
##  files?
CKPT_SERVER_DIR   = /home/condor

##########################################################################
##########################################################################
##  Settings you may want to customize:
##  (it is generally safe to leave these untouched)
##########################################################################
##########################################################################

##  The checkpoint server creates a child process for each active file
##  transfer.  What is the maximum number of processes it should
##  create?  It will deny any requests when at the maximum.  This is
##  set to 50 processes by default.
CKPT_SERVER_MAX_PROCESSES = 50

##  You can also control the maximum number of processes for
##  checkpoint restores vs. checkpoint stores.  You may want to set a
##  lower maximum for checkpoint restores, so a large number of
##  restores can't starve all checkpoint stores.  The default maximum
##  for both is 50.
CKPT_SERVER_MAX_STORE_PROCESSES = 50
CKPT_SERVER_MAX_RESTORE_PROCESSES = 50


##########################################################################
##########################################################################
##  Settings you should probably leave alone:
##  (unless you know what you're doing)
##########################################################################
##########################################################################

CKPT_SERVER_LOG         =$(LOG)/CkptServerLog
MAX_CKPT_SERVER_LOG     = 640000
CKPT_SERVER_DEBUG = D_ALWAYS
```

5.2.2.2 Fichero de configuración global

Como se comentó antes, también hay que modificar el fichero de configuración global del Condor Pool. Los cambios que hay que realizar en este fichero, si no se han realizado antes al configurar el Central Manager, son:

```
##--------------------------------------------------------------------
##  Checkpoint server:
##--------------------------------------------------------------------
##  Do you want to use a checkpoint server if one is available?  If a
##  checkpoint server isn't available or USE_CKPT_SERVER is set to
##  False, checkpoints will be written to the local SPOOL directory on
##  the submission machine.
USE_CKPT_SERVER   = True

##  What's the hostname of this machine's nearest checkpoint server?
CKPT_SERVER_HOST  = maquina1.upct.es

##  Do you want the starter on the execute machine to choose the
##  checkpoint server?  If False, the CKPT_SERVER_HOST set on
##  the submit machine is used.  Otherwise, the CKPT_SERVER_HOST set
##  on the execute machine is used.  The default is true.
STARTER_CHOOSES_CKPT_SERVER = True
```

En este caso, en el que el Central Manager también va a realizar el papel de Servidor Checkpoint, por tanto las macros CONDOR_HOST y CKPT_SERVER_HOST (ver apéndice) coinciden en cuanto a la máquina que va a realizar dicha función.

5.2.2.3 Fichero de configuración local del Central Manager

Además, también se modificará en el fichero de configuración del Central Manager el apartado referido a los procesos que el condor_master deberá lanzar al iniciarse, incluyendo el proceso del Servidor Checkpoint “condor_ckpt_server”.

```
##  What daemons do you want to run on your central manager?
##  NOTE: For it to be the central manager, you need the NEGOTIATOR
##  and COLLECTOR to run.  It's optional whether or not you want to
##  run the schedd (to allow jobs to be submited) and/or the startd
##  (to allow Condor jobs to execute) on your central manager.
DAEMON_LIST   = MASTER, COLLECTOR, NEGOTIATOR, STARTD, SCHEDD,
CKPT_SERVER
```

5.2.3 Ejecución del servidor

Una vez modificado el fichero de configuración local del Servidor Checkpoint se debe de ejecutar este. Para que los cambios sean captados por el condor_master se debe de ejecutar en la línea de comandos el comando condor_restart.

maquina1:~ # /usr/local/condor/sbin/condor_restart

Este comando hace que el condor_master vuelva a iniciar otra vez los procesos en la máquina donde ha sido ejecutado el condor_restart. De esta manera, el

condor_master observa que debe iniciar también el proceso condor_ckpt_server y así poner a funcionar el Servidor Checkpoint.

También se debe de ejecutar condor_reconfig en todas las máquinas para que los cambios en el fichero de configuración global tengan efecto en el Condor Pool.

maquina1:~ # /usr/local/condor/sbin/condor_reconfig

maquina2:~ # /usr/local/condor/sbin/condor_reconfig

maquina3:~ # /usr/local/condor/sbin/condor_reconfig

A partir de ese momento, el checkpoint estará activo en el Condor Pool. Para poder verificar su funcionamiento, bastará con ejecutar en la línea de comandos de la máquina donde se ha instalado el Servidor Checkpoint "ps –ef |egrep condor_". De esta manera se podrá observar como se está ejecutando el proceso del Servidor Checkpoint.

Aunque no es objetivo del proyecto, también se puede configurar un Condor Pool para poder utilizar múltiples Servidores Checkpoint, y así poder distribuir la carga que suponen las imágenes checkpoint para grandes redes con muchas tareas ejecutándose [37].

5.3 Tareas

Una vez que la aplicación está funcionando correctamente, se puede comenzar a someter tareas. Al haber instalado Condor con el universo Standard, todas las tareas a someter deberán de recompilarse con las librerías de Condor mediante el comando condor_compile (capítulo 3).

También hay que tener en cuenta los permisos que posee el usuario que somete tareas ya que al ejecutar los comandos necesarios, debe tener acceso a los directorios y ficheros de dichos comandos y de las tareas para poder someterlas. Por último, se ha de comprobar que el sistema operativo con el que se trabaja lleva instalados todos los compiladores necesarios para las tareas que se van a someter (gcc, f77...).

Por tanto, los pasos que se deben seguir para someter una tarea en un Condor Pool son:

1- Crear el fichero submit para poder someter dicha la tarea que se desea someter a ejecución mediante el comando condor_submit.
2- Compilar la tarea con su compilador correspondiente, creando el objeto del programa.
3- Recompilar el objeto con el comando condor_compile para crear el ejecutable.

5.3.1 Tarea propuesta

Esta primera tarea, llamada "numero.c", es un sencillo ejemplo para poder visualizar los resultados que se obtienen de ejecutar una tarea mediante condor. Esta tarea ha sido realizada para poder comprobar el comportamiento de Condor ante programas que utilizan ficheros de entrada de datos, salida de datos…en vistas hacia una instalación donde se someterán tareas de las mismas características.

```
# include <stdlib.h>
# include <stdio.h>

int main (void){

  int numero;
  int error;

  error = fscanf(stdin, “%d”, &numero);
  if(error != 1){

        fprintf(stderr, “No puedo encontrar el número \n”);
}

  fprintf(stdout, “el número es %d \n”, número);
  return 0;
}
```

Este programa pinta el string “el numero es ...” pintando el número que ha sido introducido por la entrada stdin (ya sea un fichero o la línea de comandos). En caso de que no se haya pintado un número, o el número sea incorrecto, pinta un mensaje de error, indicando que no puede encontrar dicho número.

5.3.2 Fichero de descripción submit

Una vez creado el script, el ejecutable de este será invocado mediante un fichero de descripción submit , llamado numero.sub. Este fichero tendrá la siguiente estructura:

- Llamada del fichero ejecutable.
- En este caso de tarea, al necesitar leer desde una entrada de datos, la llamada de ficheros Input. Estos ficheros, por supuesto, deben existir.
- Llamada de los ficheros output donde se escribirá el resultado de la ejecución del programa. No es necesario crear de antemano estos ficheros. Pueden ser creados por Condor al someter la tarea (ver apéndice).
- Llamada de los ficheros de error, los cuales se crearán al someter la tarea y se escribirá en el ellos en caso de error en la ejecución del programa.
- Llamada de los ficheros log, los cuales se crearán nada más someterse la tarea y contendrán la información siguiente: la tarea sometida, la máquina que ha sometido la tarea, el cluster en el cual se ha sometido, la hora en la que se sometió la tarea...
- “queue”. La cola, que dependiendo del número que le acompañe, será el número de instancias que se crearán de la tarea a ejecutar.

```
####################
# Fichero de descripción submit
####################
      Executable = numero
      Input = in.$(Process)
      Output = out.$(Process)
      Error = err.$(Process)
      Log = log.$(Process)
      Queue 3
```

En este fichero hay que destacar la macro $(Process). Condor lo sustituirá por 0 para la primera tarea en la cola, 1 para la segunda tarea en la cola y así sucesivamente. En caso de que "queue" no fuera acompañado de ningún número, no sería necesario la utilización de la macro $(Process), ya que solo se crearía un fichero de cada tipo.

Antes de someter la tarea, se deben de crear los ficheros in (in.0, in.1, in.2), con los permisos correspondientes (chmod 755 in.*) para que el programa pueda leer las entradas. Dentro de estos ficheros se encontrarán las entradas para cada tarea sometida, es decir, en el fichero in.0 estará escrito el número 0, en el in.1 el 1 y así sucesivamente. Estos ficheros deben de estar situados en el mismo directorio que el ejecutable.

Situándose, por ejemplo en la máquina 3, se procede a la creación de los ficheros antes mencionados, dando permisos a dichos ficheros, siempre en el directorio en el que se encuentre la tarea a someter y el fichero de descripción submit.

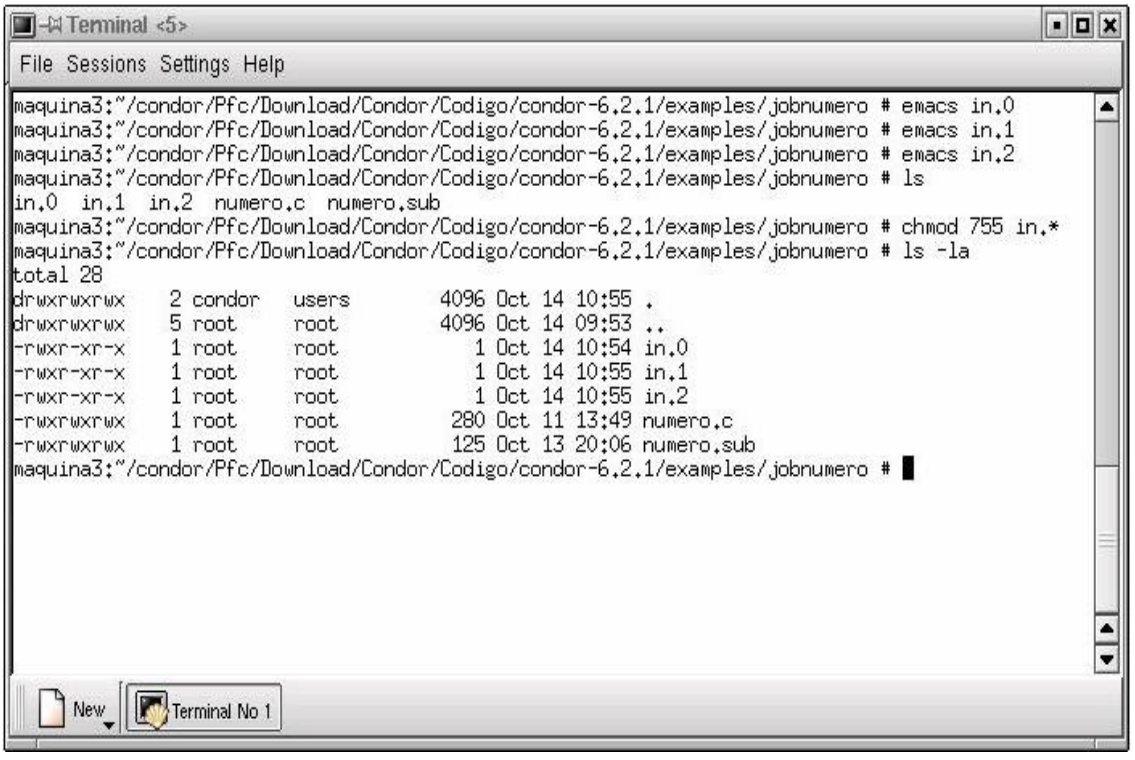

Figura 5. 6 Ficheros de lectura de datos (in.*)

5.3.3 Compilación

Una vez que está establecida la tarea, se debe compilar. Al tratarse de un programa .c, se realizará mediante el compilador gcc, creando así el objeto (numero.o).

Cuando se ha creado el objeto, se debe comprobar que tiene permisos suficientes para que pueda ser utilizado por el usuario de la máquina que interese, ya que este debe ser capaz de recompilar dicho objeto con Condor mediante el comando condor_compile para enlazarlo a las librerías de Condor y crear así el ejecutable.

En la siguiente figura se muestra la compilación del ejemplo propuesto "numero.c, mediante el compilador gcc . Una vez creado el objeto y trabajando bajo un usuario diferente a Root, por razones de seguridad (capítulo 3), por ejemplo bajo el usuario condor, se realiza el enlace del objeto con las librerías Condor para crear el ejecutable "numero", que, por supuesto, debe situarse también en el mismo directorio que el fichero de descripción submit, los ficheros de entrada de datos…

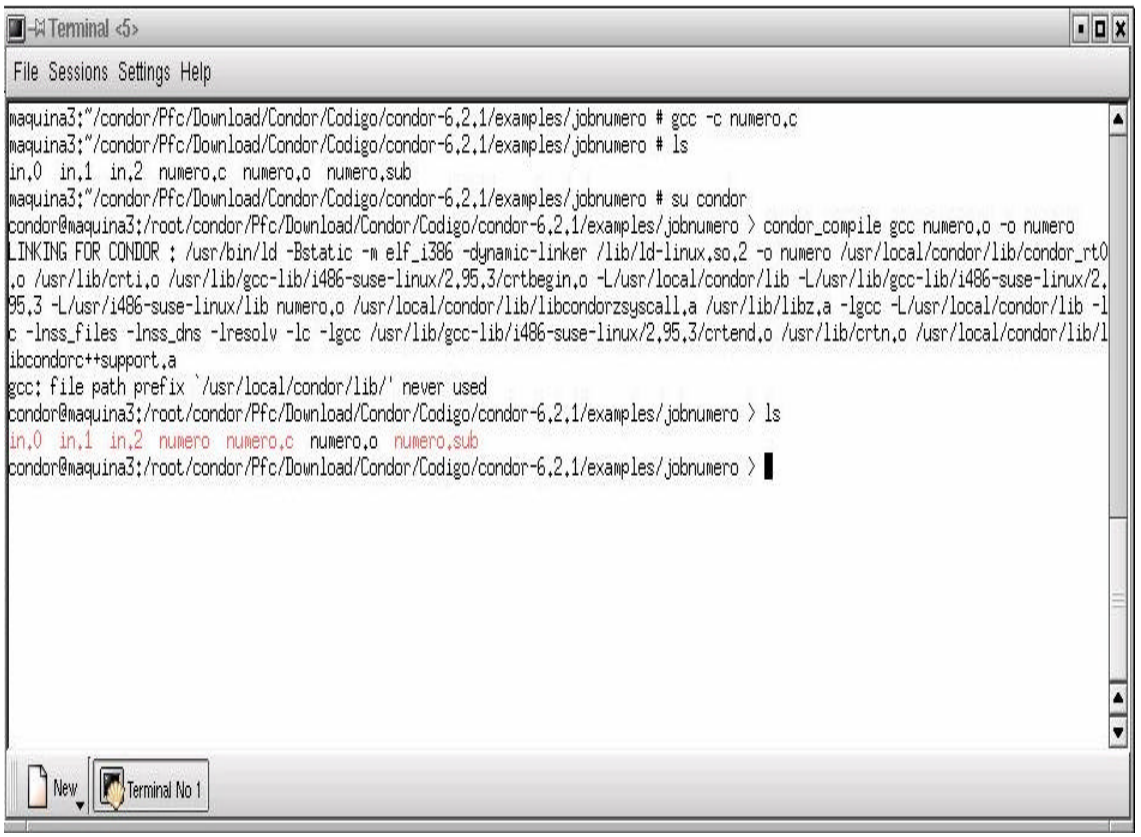

Figura 5. 7 Creación del fichero ejecutable

5.3.4 Sometiendo la tarea

Antes de someter cualquier tarea, hay que verificar la ejecución de la misma. Para ello bastará con llamar al ejecutable en la línea de comandos. Como dicho ejecutable ha sido enlazado a las librerías de Condor, al ejecutar la tarea el código detectará que no se está siendo ejecutado por Condor, con lo cual pintará dos mensajes de notificación.

Estos mensajes de notificación son:

- Se indica que se crea en el mismo directorio donde se encuentra el ejecutable un nuevo fichero imagen checkpoint, donde más adelante se verá la utilidad de este (5.4.1 Standalone Checkpoint).
- Se indica las llamadas al sistema remoto están inactivas, por tanto, la tarea solo se puede ejecutar en esta máquina.

Siguiendo con el mismo ejemplo que en los apartados anteriores, se llama a ejecución al programa “numero”, introduciendo por la línea de comandos el parámetro de entrada. Se puede visualizar en la siguiente figura como aunque se producen las dos notificaciones comentadas anteriormente, el programa se ejecuta correctamente, visualizando en el terminal la salida del mismo.

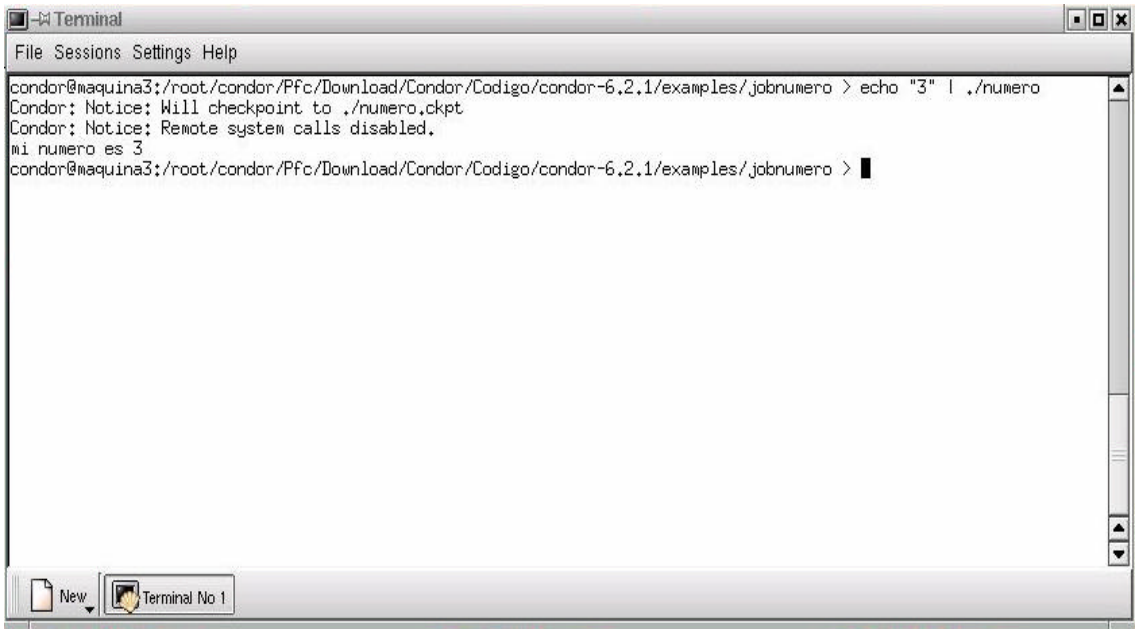

Figura 5. 8 Sometiendo tareas

Una vez comprobado la perfecta ejecución del programa, y habiendo creado el fichero de descripción submit, ya se puede someter la tarea, siguiendo como usuario diferente a root.

Para someter la tarea, se usará el comando condor_submit seguido del fichero de descripción submit. Hay que tener en cuenta, que la llamada de condor_submit se debe hacer desde el directorio donde se encuentre la tarea y el fichero de descripción submit que se invoque en la línea de parámetros.

Utilizando el mismo ejemplo, el ejecutable numero y el fichero de descripción submit numero.sub, se pueden ver los resultados de someter la tarea en la siguiente figura.

Sometida la tarea, si se analiza la cola de tareas mediante el comando condor_q se observarán todas las tareas sometidas, El cluster en el cual ha sido sometida, la máquina que las somete y el estado en el que se encuentran las tareas en el momento de la ejecución del comando.

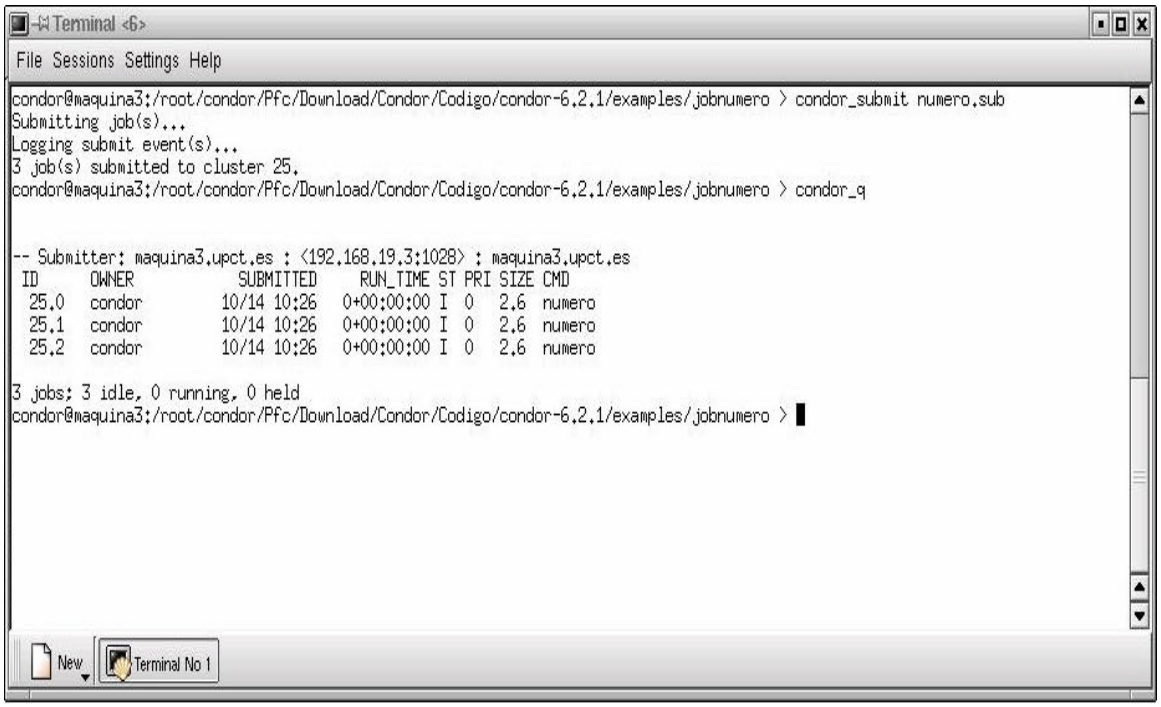

Figura 5. 9 Cola de tareas

Cuando la tarea ha sido sometida, se puede comprobar si se han creado los ficheros indicados en el fichero de descripción submit.

Siguiendo con el mismo ejemplo, situándose dentro del directorio en el que se está trabajando se puede comprobar que se han creado los ficheros log, err y out, como muestra la siguiente figura.

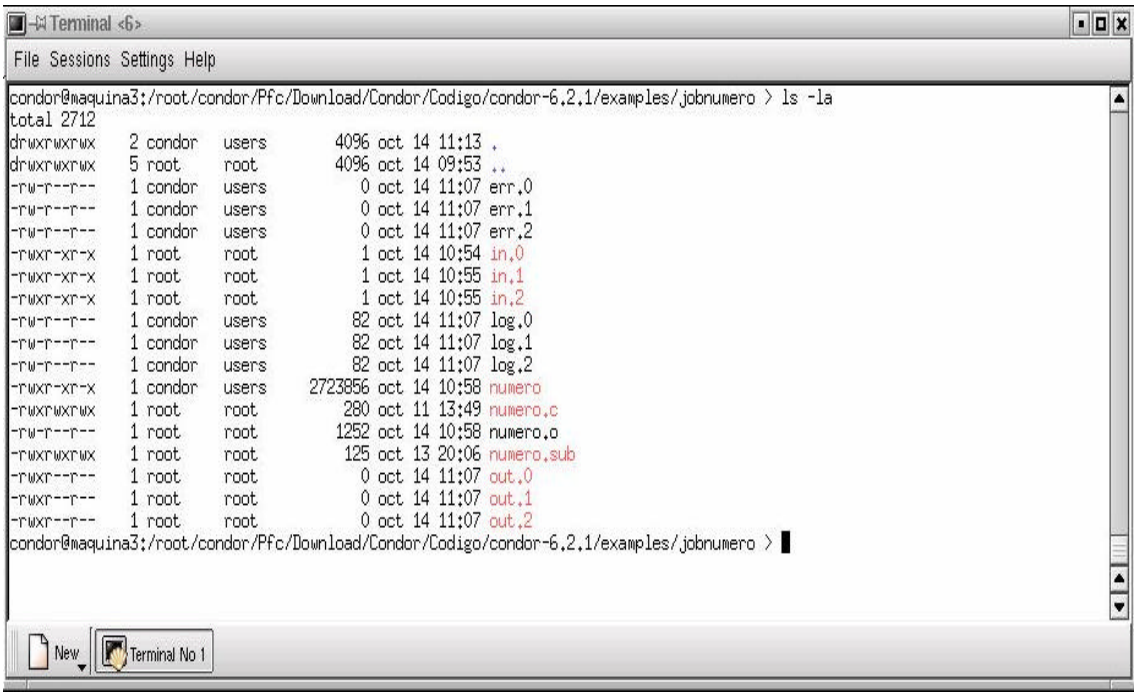

Figura 5. 10 Ficheros de salida

5.3.5 Resultados obtenidos

Una vez que la tarea ha sido ejecutada, se pueden visualizar los ficheros out donde se ha escrito el resultado de la ejecución del programa, los ficheros log en los cuales se podrá ver la siguiente información: la tarea sometida, la máquina que ha sometido la tarea, el cluster en el cual se ha sometido, la hora en la que se sometió la tarea, el checkpoint... y los ficheros err, que si no ha habido ningún error quedarán en blanco. Mientras la tarea esté sometida en la cola de tareas sin ejecutarse, los ficheros out estarán en blanco.

Para el ejemplo utilizado, se podrá visualizar para cada una de las ejecuciones de la tarea el siguiente contenido en los ficheros:

- Fichero out.0:

```
El numero es 0
```

- Fichero log.0:

```
000 (032.000.000) 10/14 11:07:01 Job submitted from host:
<192.128.19.3:1031>
001 (032.000.000) 10/14 11:08:31 Job executing on host:
<192.128.19.2:1031>
005 (032.000.000) 10/14 11:08:31 Job terminated.
        (1) Normal termination (return value 0)
                Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote
Usage
              Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Total Remote
Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local
Usage
```

Los demás ficheros (out.1, out.2, log.1, log.2) tendrán un resultado semejante.

5.4 Otras ejecuciones

A continuación se muestran otros mecanismos que ofrece la Aplicación Condor para proveer al usuario del Condor Pool de una mayor funcionalidad cuando somete tareas. Tales mecanismos vienen definidos por diferentes comandos y módulos de Condor [37].

5.4.1 Standalone Checkpoint

Aunque la configuración de Condor con el entorno Standard puede complicar mucho más la ejecución de un programa debido a que a la hora de compilar los programas estos deben ser recompilados con las librerías de Condor, el usuario puede sacar provecho de esta situación por medio de “Standalone Checkpointing”.

Para entender este mecanismo se debe pensar en una situación real en la que un usuario de una máquina que forma parte de un Condor Pool decide ejecutar un programa que ha sido enlazado con las librerías de Condor fuera del entorno del Pool, es decir, en la máquina local, ya que no le conviene por cualquier circunstancia someter el programa en el Condor Pool. Como se comentó en apartados anteriores, la ejecución de tareas enlazadas con las librerías Condor fuera de un Condor Pool es posible. A continuación se demuestra la utilidad de esto.

Al realizar la ejecución del programa aparecen dos mensajes de Condor avisando primero sobre la creación en el directorio en el que se encuentra el ejecutable de un fichero imagen checkpoint del ejecutable y segundo sobre la inhabilitación del mecanismo de "Remote System Calls" ya que el programa no ha sido sometido a ningún Condor Pool.

El usuario que inicia la ejecución del programa, por cualquier razón se ve obligado a parar la ejecución. En un principio, si el programa hubiera sido compilado sin enlazarlo con las librerías de Condor, una vez suspendida la ejecución, no se podría recuperar el trabajo realizado hasta el momento de suspender la ejecución, y por tanto en caso de reiniciar la ejecución del programa no se puede comenzar desde donde se quedó.

Con el mecanismo "Standalone Checkpointing" esto es posible ya que al someter el programa a ejecutar se crea un fichero checkpoint, haciendo uso del mecanismo de Checkpoint de Condor en el entorno Standard. En el momento en el que se suspende la ejecución del programa, se escribe la última imagen de la ejecución del programa en el fichero checkpoint creado al ejecutarse la tarea. De esta manera, si se desea reiniciar la ejecución del programa, no será necesario comenzar desde el principio, malgastando ciclos de CPU, sino que puede comenzar la ejecución desde donde se dejó por medio del fichero checkpoint.

En las siguientes figuras se muestra el funcionamiento de este mecanismo. Para ello se utiliza el ejemplo que se presentó (numero.c) en la red propuesta, y se compila con las librerías de Condor (como se vio en anteriores apartados). No se usa en este caso el comando condor_submit, sino que se ejecuta la tarea de forma normal en la línea de comandos. El mecanismo de checkpoint de Condor y el mecanismo de llamadas de procedimiento remoto pintarán el mensaje que se muestra a continuación al iniciar la ejecución del programa.

Para forzar al programa a que escriba en el fichero y se pare se debe de enviar la señal SIGTSP, presionando CONTROL –Z (donde aparece el mensaje de "User defined signal 2"). Se ve a continuación que se ha creado el fichero numero.ckpt.

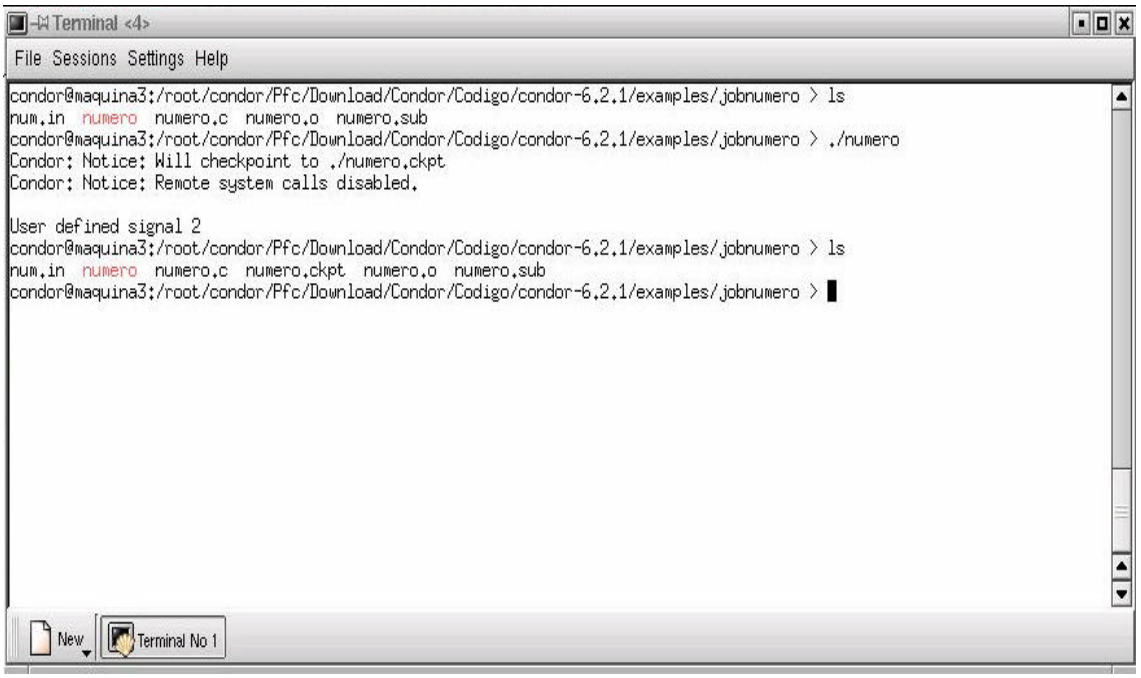

Figura 5. 11 Standalone en una tarea

Para volver a iniciar el programa desde el fichero checkpoint bastará con ejecutar dicho programa añadiendo como argumentos la opción -condor_restart y el nombre del fichero imagen checkpoint. De esta manera, la ejecución del programa comenzará desde donde se paró. En la siguiente figura se muestra como Condor pinta un mensaje indicando que la ejecución se ha reiniciado desde la imagen numero.ckpt (Condor:Notice: Will restart from numero.ckpt).

Figura 5. 12 Reinicio de una tarea desde el checkpoint

5.4.2 Tareas en Estado held

Si se quiere someter una tarea en un Condor Pool pero no se desea que dicha tarea se ejecute en ese mismo momento, esta se puede añadir en la cola de tareas en el estado de held. De esta manera, la tarea no será sometida a ejecución por ninguna máquina Execute.

Para establecer una tarea de la cola de tareas al estado held, en el fichero de descripción submit de la tarea, se debe añadir el parámetro "hold = true" (ver apéndice). De esta manera, al someter la tarea y visualizar el estado de las tareas en la cola de tareas se podrá observar que se encuentra en estado held.

Si se desea someter la tarea que se encuentra en estado held en la tarea, bastará con ejecutar en la línea de comandos el comando "condor_release" seguido del nombre del cluster en el que se encuentra la tarea en estado held. De esta manera, la tarea pasará otra vez a estar disponible para que una máquina Execute pueda ejecutarla.

A continuación se muestra un ejemplo del estado "held" de una tarea. En un principio se someten varias tareas "loop.remote", donde varias de ellas han sido establecidas en estado "held" (H). Si se desea que la tarea, por ejemplo loop.remote 500 este disponible para poder ser ejecutada, se cambia el estado de la misma como se muestra en la figura escribiendo en la línea de comandos condor_release 36.4. Con este comando seguido del cluster donde se encuentra la tarea, esta pasa al estado "idle" (I).

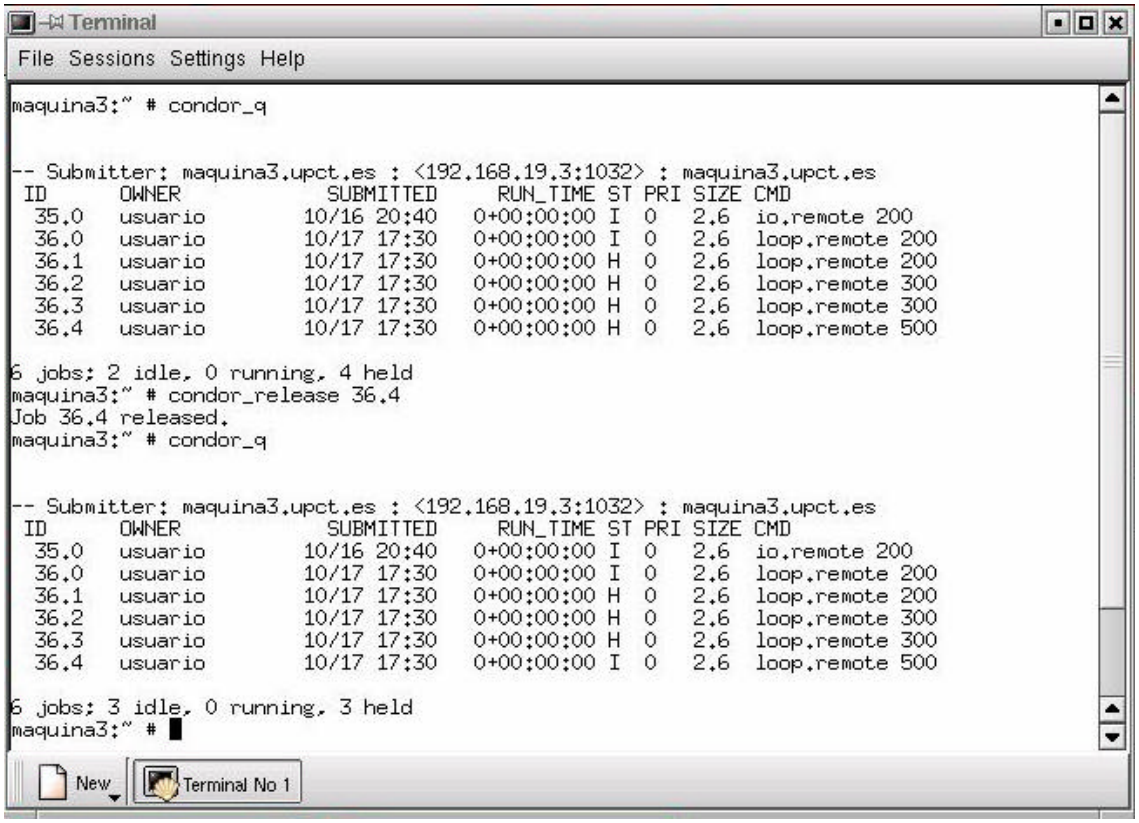

Figura 5. 13 Tareas en estado held

5.4.3 Eliminar una tarea de la cola

Una tarea puede ser eliminada de la cola de tareas usando el comando "condor_rm". Si la tarea que se elimina de la cola de tareas se encuentra en estado de ejecución (running), será eliminada sin la generación de ningún checkpoint, con lo cual, si se decide iniciar de nuevo su ejecución, no se podrá comenzar desde donde quedó la última ejecución ya que no habrá fichero checkpoint para poder reiniciarla.

El siguiente ejemplo muestra la cola de tareas antes y después de eliminar la tarea numero. Para eliminar una tarea, deberá ir acompañando al comando condor_rm el número del cluster perteneciente a la tarea que se quiere eliminar. Una vez eliminada, si se vuelve a examinar la cola de tareas, la tarea marcada para ser eliminada habrá desaparecido de la cola de tareas.

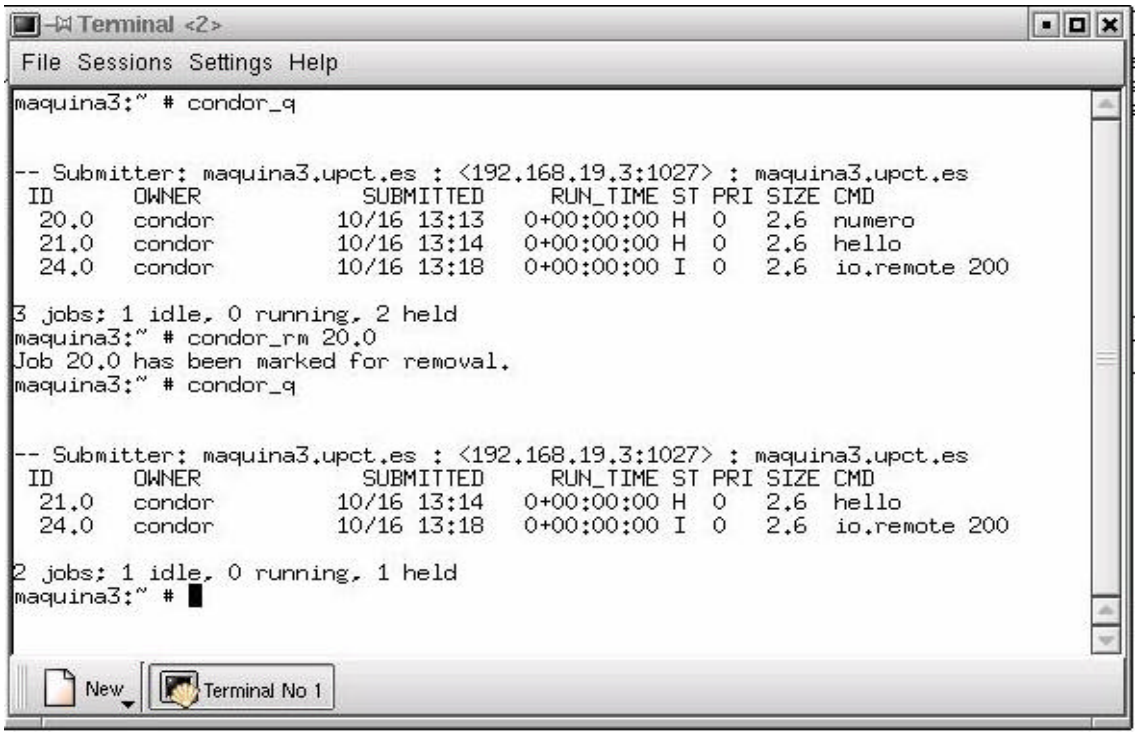

Figura 5. 14 Eliminación de una tarea de la cola de tareas

5.4.4 Cambiar las prioridades de las tareas

Además de las prioridades asignadas a los usuarios, Condor también provee a cada usuario con la capacidad de asignar prioridades a cada tarea que se somete. Estas prioridades son locales de cada cola, es decir, de cada máquina que somete tareas y el rango va desde –20 hasta 20, siendo este último el valor que adjudica la mayor prioridad de todas. Por defecto, cuando se somete una tarea, la prioridad es 0, pero se puede cambiar usando el comando "condor_prio".

En la siguiente figura se muestra un ejemplo del uso de este comando. Por ejemplo, se quiere cambiar la prioridad de la tarea "hello" que se encuentra con prioridad 0 a prioridad 10. Para ello el comando "condor_prio" debe de ir acompañado de la opción –p (que indica prioridad), el valor de la prioridad que se desea y el número del cluster donde se encuentra la tarea a la que se le quiere establecer la prioridad.

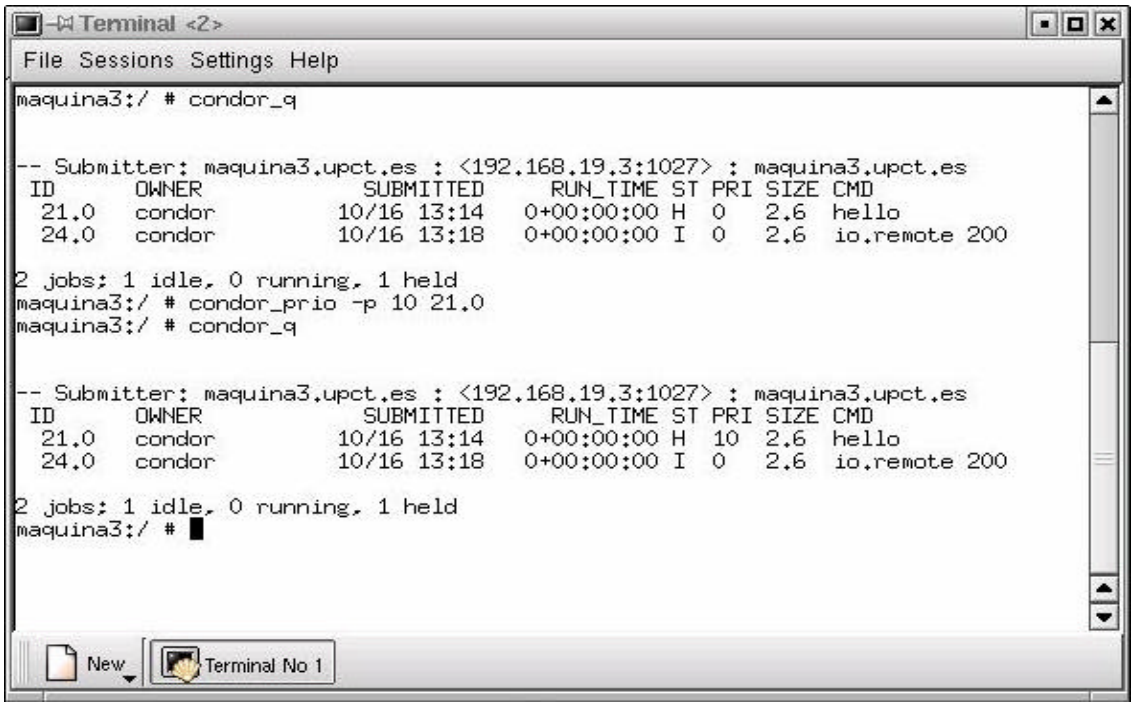

Figura 5. 15 Cambio de prioridad de las tareas

Es importante el indicar que la prioridad de las tareas no tienen nada que ver con las prioridades asignadas por Condor a los usuarios en el fichero de configuración global del Condor Pool (ver apéndice). Las prioridades de las tareas no causan impacto alguno en las prioridades de los usuarios.

5.4.5 Problemas en la ejecución de tareas

En ocasiones un usuario del Condor Pool se puede encontrar que a veces al someter una tarea y visualizar al cabo del tiempo la cola de tareas, la tarea nunca llega a ejecutarse. Hay diferentes causas que pueden hacer que esto ocurra:

- Fallo en las características de la máquina.
- Tareas que tienen preferencias.
- Insuficiente prioridad.

Algunas de estas causas se pueden analizar mediante la opción “-analyze” del comando “condor_q”.

En la siguiente figura se muestra un ejemplo de la tarea “io” que no se ejecuta pasado ya un largo periodo de tiempo. Ejecutando “condor_q –analyze” se puede obtener información sobre las posibles causas que han hecho que la tarea no pueda ejecutarse. Parece ser que para esta tarea la expresión Requirements especificada (ver apéndice) hace que la máquina no cumpla los requisitos establecidos en el fichero de configuración global del Condor Pool, con lo cual la tarea nunca llega a ejecutarse. Una manera de solucionarlo, en caso de no poder acceder a la modificación del fichero de configuración global (que será lo más probable) es indicar en el fichero de descripción submit de la tarea los requerimientos que se deben de cumplir para que la tarea pueda ser ejecutada en cuanto al sistema operativo y arquitectura de las máquinas del Condor Pool.

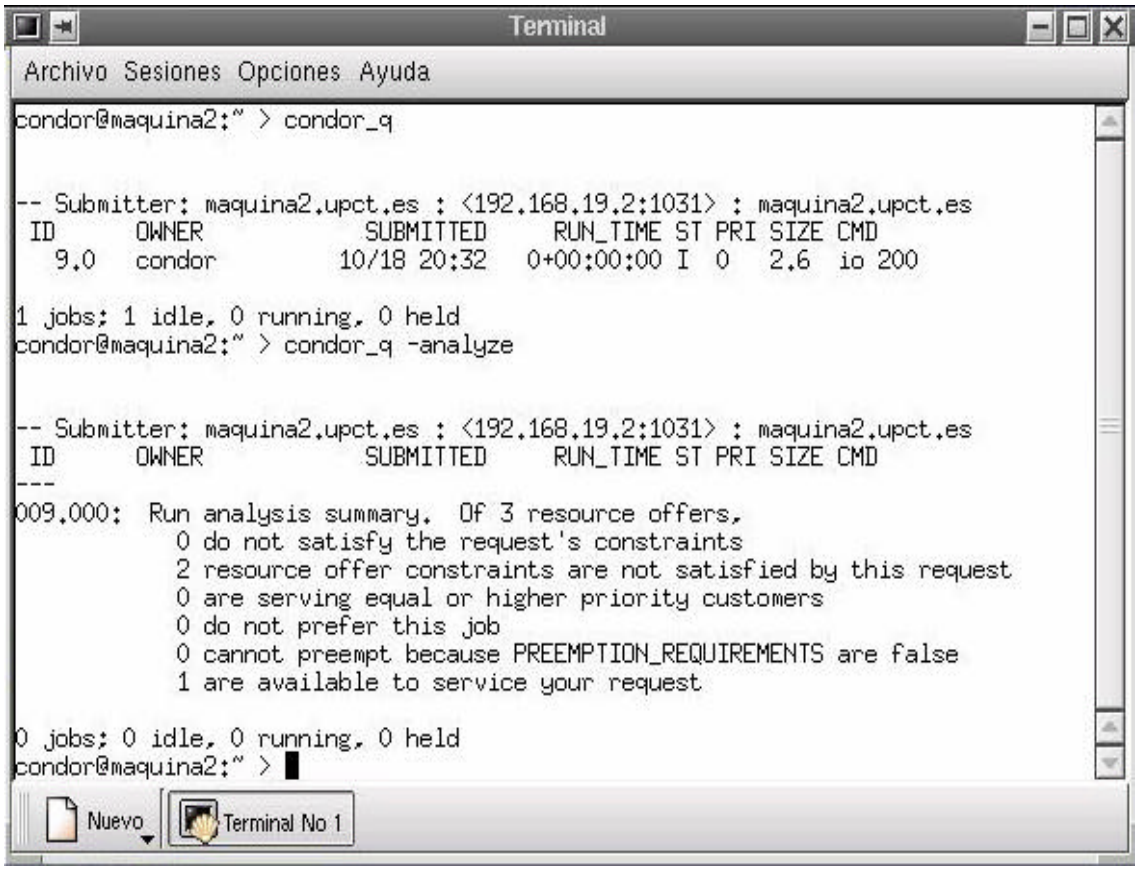

Figura 5. 16 Problemas cuando no se ejecuta una tarea

Mientras que el analizador (opción -analyze) puede diagnosticar algunos de los problemas más comunes, algunas situaciones no pueden detectarse. Puede que el analizador de la respuesta al problema, se realicen los cambios necesarios y que la tarea siga sin ejecutarse. En este momento es donde el retraso es algo puede ser que influya en la no ejecución de la tarea, ya que la tarea no se ejecutará hasta el siguiente ciclo de negociación (establecido en el fichero de configuración global).

Si el problema persiste y la opción –analyze es incapaz de detectar lo que ocurre, puede ser que la tarea este siendo ejecutada e inmediatamente termine a pesar de los problemas que puedan haber. Si la tarea sigue sin ejecutarse, se pueden visualizar los ficheros error y log que se han indicado en el fichero de descripción submit. En caso de no encontrar la solución, y como último caso, lo mejor es contactar con el Administrador del Sistema.

5.4.6 Fin de una tarea

Cuando una tarea finaliza su ejecución, se borra de la cola de tareas (es decir, no aparecerá al ejecutar el comando condor_q) y los pasos dados por la tarea desde que se sometió hasta el fin de su ejecución se insertarán en el fichero del historial de tareas de la máquina. Se puede examinar dicho fichero mediante el comando “condor_history”.

Si se ha especificado un fichero log en el fichero de descripción submit, cuando la tarea finalice, se indicará en el fichero log. Por defecto, cuando la tarea finalice, Condor enviará un email a la dirección que se especificó al realizar la instalación de Condor en la máquina (4.8.1.1 instalación).

5.4.7 Sometiendo tareas como usuario no privilegiado

Normalmente, los usuarios que trabajan bajo una red linux, lo hacen desde un usuario no privilegiado, por razones de seguridad. Por tanto, cuando quieran someter una tarea, la aplicación Condor debe ser capaz de mantenerla a través de dicho usuario sin problema alguno.

Aunque se ha comentado en varias ocasiones el uso de Condor por parte de usuarios no privilegiados en las máquinas, es necesario recordar los pasos a realizar por un usuario cualquiera para poder someter una tarea, ya que, al fin y al cabo, el objetivo final es que un usuario cualquiera, ajeno a la arquitectura y funcionamiento de la Aplicación Condor sea capaz de someter a ejecución sus programas.

Por tanto, en la siguiente figura se muestra como un usuario cualquiera (por ejemplo "usuario@maquina3") de la máquina 3 tiene un programa que se desea someter bajo Condor. Los pasos a realizar para someter dicha tarea serán los siguientes:

- Situarse en el directorio en el cual se encuentre la tarea que se desea ejecutar, es decir, si se desea someter, por ejemplo, la tarea io.c, habrá que situarse en este caso en el directorio /tareas de la máquina 3.
- Compilar dicha tarea, teniendo en cuenta el universo en el que se esté trabajando (en este ejemplo, el universo es standard, por tanto hay que compilar con condor_compile).

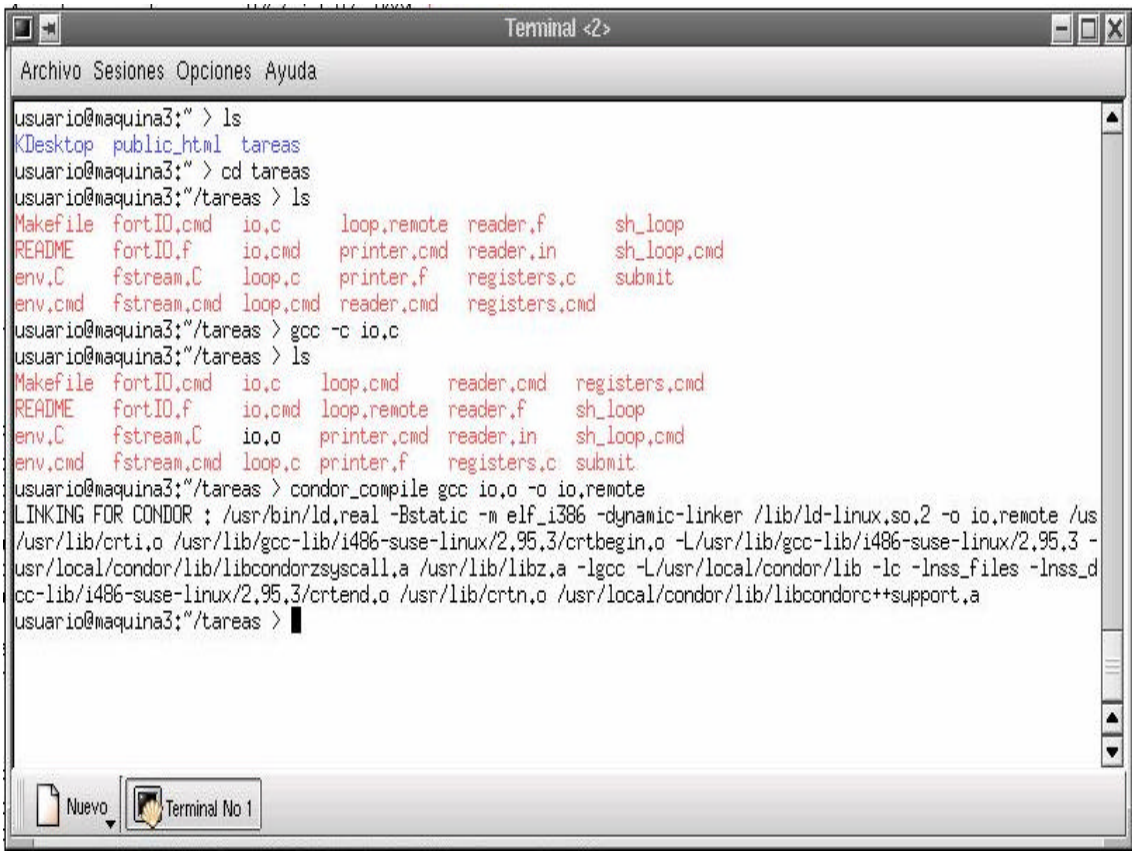

Figura 5. 17 Compilación de tareas como usuario no privilegiado (I)

- Una vez compilada, y teniendo el fichero de descripción submit configurado en cuanto a las necesidades del usuario, bastará con someter la tarea mediante el comando condor_submit. Hay que notar que al someter la tarea, la aplicación Condor pintará en pantalla tantos warning como ficheros se hallan indicado en el fichero de descripción submit que se tienen que crear. Estos warning indican que solo el usuario que está sometiendo la tarea tendrá permisos de lectura y escritura de los ficheros creados al someter la tarea. Este comportamiento por parte de Condor añade mayor seguridad a los datos de los usuarios.

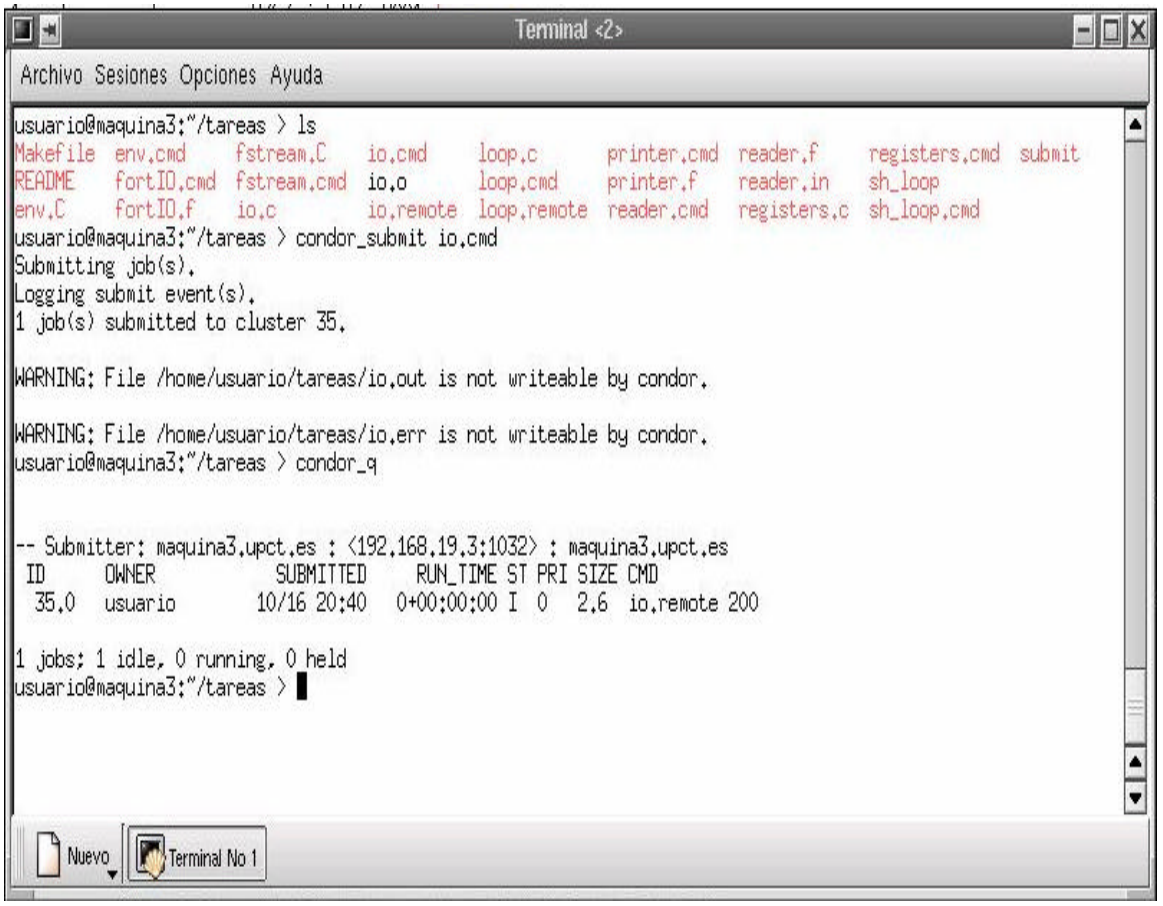

Figura 5. 18 Sometiendo tareas como usuario no privilegiado (II)

- Una vez que la tarea haya sido ejecutada, los resultados, como se vio en apartados anteriores, quedarán guardados en los ficheros .out, además de los ficheros .log donde se podrán visualizar los pasos que se han seguido en el Condor Pool para la ejecución de la tarea.

5.5 Módulo Condor View

El módulo Condor View es una opción que ofrece Condor para conocer de forma gráfica el uso de los recursos de un Condor Pool. Tiene como función principal llevar el seguimiento de los datos en el tiempo para controlar la utilización de la CPU por parte de un Condor Pool. Para ello existe una herramienta gráfica que monitoriza

tal información, mostrándola en modo de gráficos para visualizar la utilización de las máquinas por parte de la Aplicación Condor.

En las siguientes figuras se muestran los gráficos que se pueden obtener gracias al módulo Condor View. En la primera figura se muestra la utilización de ciclos de CPU de las máquinas que forman parte del Condor Pool en función del tiempo. La segunda figura muestra el la carga de procesamiento que tiene el Pool en cuanto a las tareas sometidas.

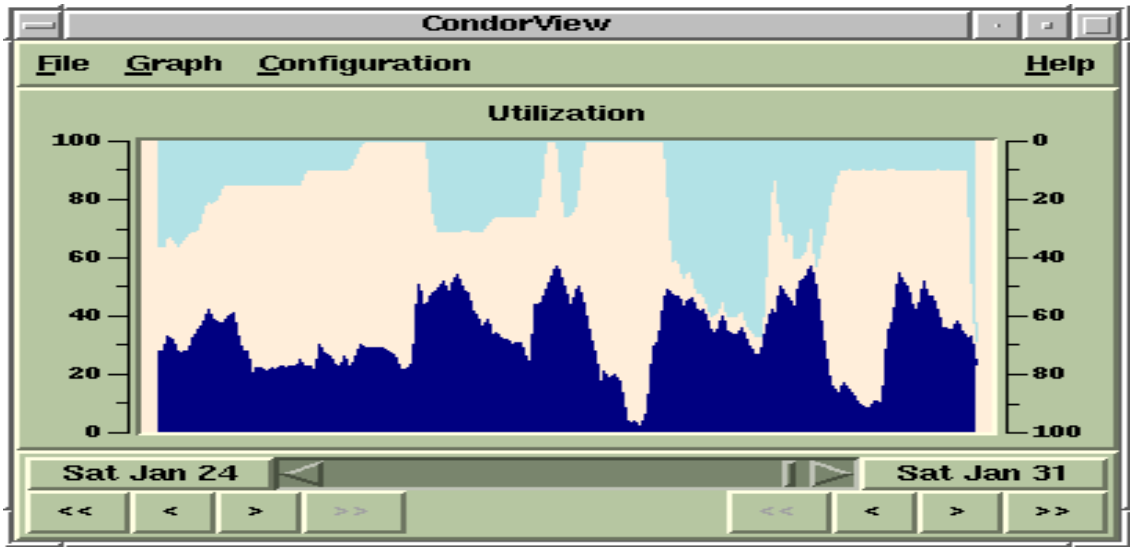

Figura 5. 19 Gráfico de la utilización del CPU en Condor Pool

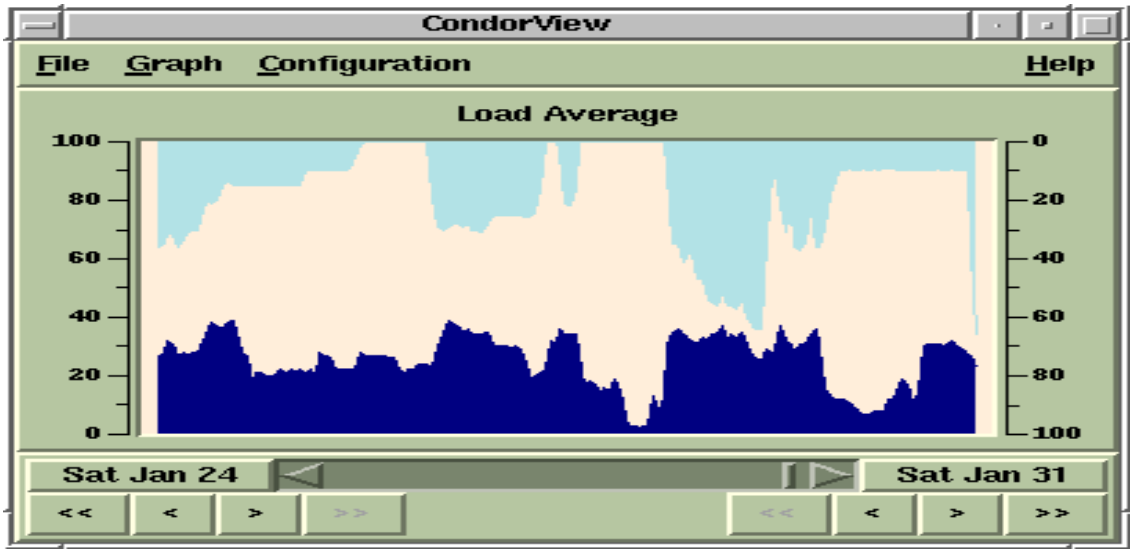

Figura 5. 20 Carga media de procesamiento de las tareas en Condor Pool

En la red prototipo que se ha diseñado, se ha decidido instalar el módulo Condor View para conocer mejor la información que este nos puede ofrecer y comprobar la utilidad de esta herramienta en vistas a la instalación que se tiene pensada en el Condor Pool del laboratorio IT-3 del Área de Telemática.

5.5.1 Instalación del módulo Servidor Condor View

Como en el caso del Servidor Checkpoint, el módulo Condor View también es un módulo opcional, no incluido en la distribución principal de Condor de versiones inferiores a la versión 6.1. A partir de esta versión, y posteriores, este módulo está incluido automáticamente en Condor. Como se ha instalado la versión 6.2.1 de Condor en el Condor Pool, no es necesario instalar ningún módulo de Condor View en la máquina que haga de Servidor.

Para instalar Condor View en un Condor Pool se necesita:

1- El servidor Condor View para el Pool, el cual recolecta la información del Pool. Como tal información reside en el Central Manager, dicho servidor deberá ser instalado en el Central Manager. Esto se demostrará en apartados posteriores.
2- El cliente Condor View, un applet de Java que visualiza estos datos.

El Servidor Condor View es una versión reforzada del "condor_collector" que registra información en disco, dando un historial del estado del Condor Pool. Esta información incluye el estado de la máquina, así como el estado de las tareas sometidas por los usuarios. El historial de información se puede activar o no, dando opción así a poder instalar el Condor View collector en una máquina sin usar espacio de disco para el historial de información si no se quiere.

El Condor View collector es el condor_collector que ha sido especialmente configurado. Si dicho módulo es ejecutado en una maquina diferente a la máquina donde normalmente se ejecuta el condor_collector (Central Manager), entonces el Condor Pool debe ser configurado para enviar actualizaciones a ambas máquinas, es decir, al condor_collector y al Condor View collector.

Si se instala Condor View collector en una maquina diferente a la máquina donde se encuentra el condor_collector, se genera mucho más trafico (se duplican todas las actualizaciones que son enviadas desde cada máquina en el Condor Pool a ambos colectores), por esto se recomienda que el módulo del Condor View sea instalado en la máquina que haga de Central Manager.

Antes de configurar el Condor View collector (máquina 1 de la red prototipo), se deben añadir los siguientes parámetros en el fichero de configuración local (si no se ha realizado ya en la configuración del Pool, capítulo 4) de la máquina elegida para habilitar la colección de datos históricos. Estos parámetros se describen a continuación:

POOL_HISTORY_DIR

Este es el directorio donde se almacenarán los datos históricos. Este directorio debe tener permisos para que cualquier usuario de la máquina donde se encuentra instalado el Condor View collector que se este ejecutando pueda escribir en el.

Los ficheros que se escribirán serán ficheros HTML, por tanto dicho directorio debería colgar de un servidor web. De esta manera, se podría acceder a los ficheros HTML desde el browser de la web.

Nota: este directorio debe estar separado de los directorios spool/ y log/ ya establecidos por Condor. Surgen ciertos problemas al poner estos ficheros en esos directorios, pero no se entrará en detalles [48].

```
## Set the directory where history files reside
POOL_HISTORY_DIR = /usr/local/condor/home/condorview
```

También hay un limite configurable del espacio requerido para los ficheros creados por el Servidor Condor View llamado (**POOL_HISTORY_MAX_STORAGE**). Lo normal es dejar este parámetro con el valor por defecto

```
## The maximum combined size of the history files
POOL_HISTORY_MAX_STORAGE = 10000000
```

Para esta instalación se ha prescindido de situar los ficheros html en un servidor web (ver capítulo 6 Líneas futuras), utilizando el programa "Konqueror" de Linux para visualizar las ficheros html como si de un servidor web se tratara.

KEEP_POOL_HISTORY

Este es un valor booleano que determina si el Condor View collector debería almacenar el historial de información. Esta a FALSE por defecto, el cual se debe especificar a TRUE en el fichero de configuración local para permitir la colección de datos.

```
## Enable history logging in the collector
KEEP_POOL_HISTORY = True
```

Una vez establecidos los parámetros en el fichero de configuración local de la máquina que hace de Servidor Condor View, se debe crear el directorio que se ha especificado en POOL_HISTORY_DIR, es decir, el directorio condorview/ y darle permisos de escritura para el usuario que ejecutará el Condor View collector en la máquina. Este debe ser el mismo usuario que el propietario del fichero CollectorLog en el directorio log, para que así no ocurran errores debido a los permisos.

Después de haber configurado los atributos de Condor View se debe configurar el Condor Pool para que automáticamente se ejecute y comience a informar al Servidor Condor View. Esto se hace añadiendo COLLECTOR al parámetro DAEMON_LIST, en caso de que no esté añadido (en caso de que la máquina no sea el Central Manager) y definiendo que significa COLLECTOR, es decir:

```
COLLECTOR = $(SBIN)/ condor_collector
DAEMON_LIST= MASTER, STARTD, SCHEDD, COLLECTOR.
```

Para que los cambios tengan efecto se debe reiniciar Condor en la máquina (utilizando el comando condor_restart).

```
maquina1:~ # /usr/local/condor/sbin/condor_restart
```

Como ultimo paso, se debe llamar a todas las máquinas del Condor Pool para comenzar a enviar actualizaciones a los colectores. Esto se hace especificando el siguiente parámetro en el fichero de configuración global del Condor Pool.

CONDOR_VIEW-HOST = maquina1.upct.es

Donde se indica el nombre completo del host de la maquina donde se está ejecutando el Condor View collector. Una vez establecidos todos los parámetros, se debe ejecutar "condor_reconfig" a todas las maquinas del Condor Pool para que los cambios tengan efecto.

maquina1:~ # /usr/local/condor/sbin/condor_reconfig

maquina2:~ # /usr/local/condor/sbin/condor_reconfig

maquina3:~ # /usr/local/condor/sbin/condor_reconfig

5.5.2 Instalación del módulo Cliente Condor View

Este módulo se usa para generar de forma automática paginas www y mostrar así las estadísticas sobre tiempo y carga de procesamiento de las máquinas de un Condor Pool. En este módulo está incluido en un script shell que invoca al comando "condor_stats" [37] para generar y actualizar las estadísticas del Servidor Condor View y generar paginas HTML con los resultados obtenidos.

También se incluye un applet, para visualizar en forma de gráficas la información de Condor. Los usuarios pueden interactuar con el applet para personalizar la visualización, para hacer zoom en la trama específica que interese, cambiar los colores de la gráfica….

La siguiente figura es un ejemplo de página web creada por el módulo Condor View en un cliente. Esta pertenece al Condor Pool de la Universidad de Winsconsin-Madison (Universidad que creó la Aplicación Condor). En esta se muestra una estadística sobre el uso de las máquinas de un Condor Pool de la Universidad de Winsconsin durante una hora.

Para visualizar mas estadísticas sobre este Condor Pool, se puede visitar la página web http://www.cs.wisc.edu/condor y entrar en el enlace Condor View.

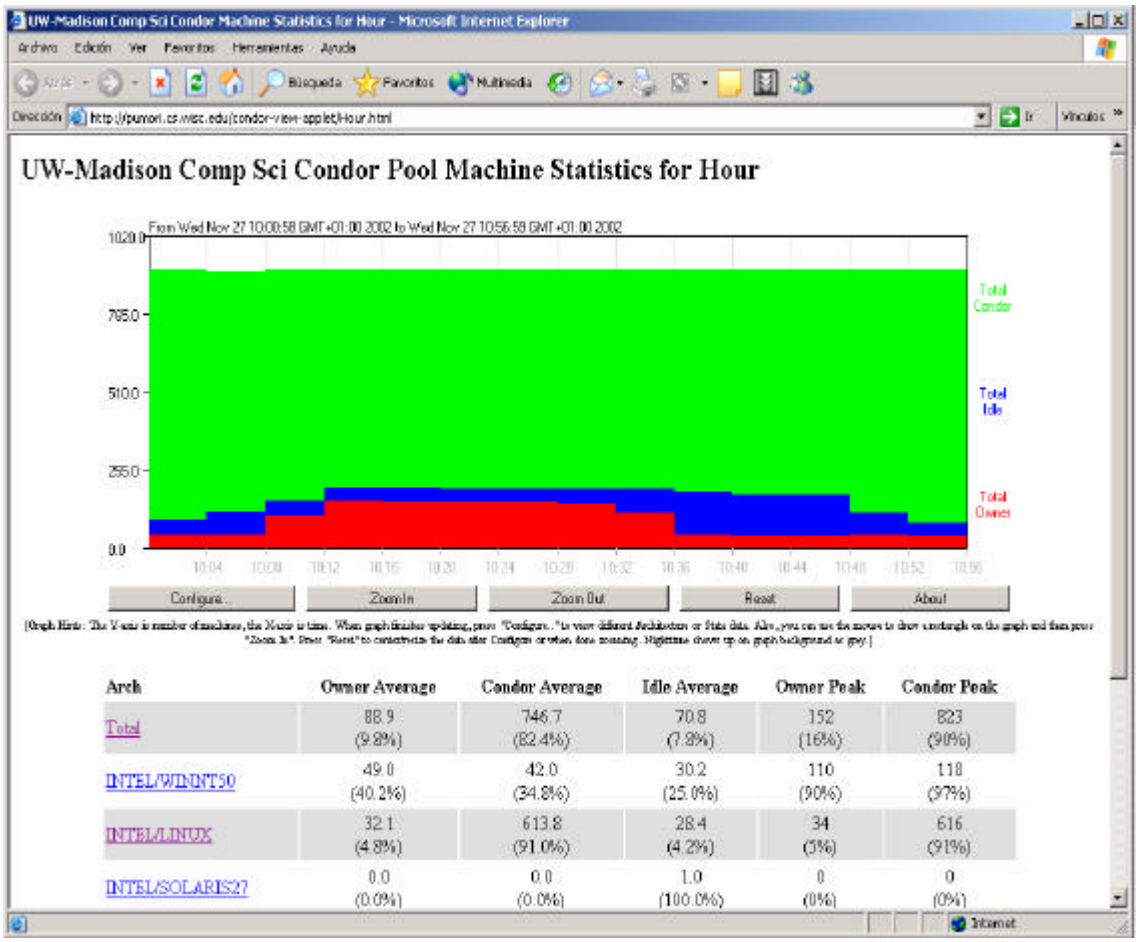

Figura 5. 21 Estadísticas del Condor Pool de UW-Madison

A continuación se comentan los pasos para instalar el módulo del Cliente de Condor View. Para la instalación del Cliente Condor View debe estar instalado y configurado el Servidor Condor View. El servidor registra la información en el disco para suministrar las estadísticas del historial de una base de datos. El Cliente Condor View hace preguntas a través de la red de trabajo a la base de datos.

Se debe descomprimir el módulo Cliente Condor View en el directorio que se especificará en el parámetro VIEWDIR que se verá a continuación. Al descomprimir el módulo se crearan diversos ficheros y directorios.

La siguiente figura muestra los pasos a realizar en la máquina para configurar como Cliente del Condor View, descomprimiendo en primer lugar el módulo descargado, situándose en el directorio creado y descomprimiendo el módulo que hay en su interior y contiene las clases necesarias para crear las gráficas, scripts para la configuración del Cliente...

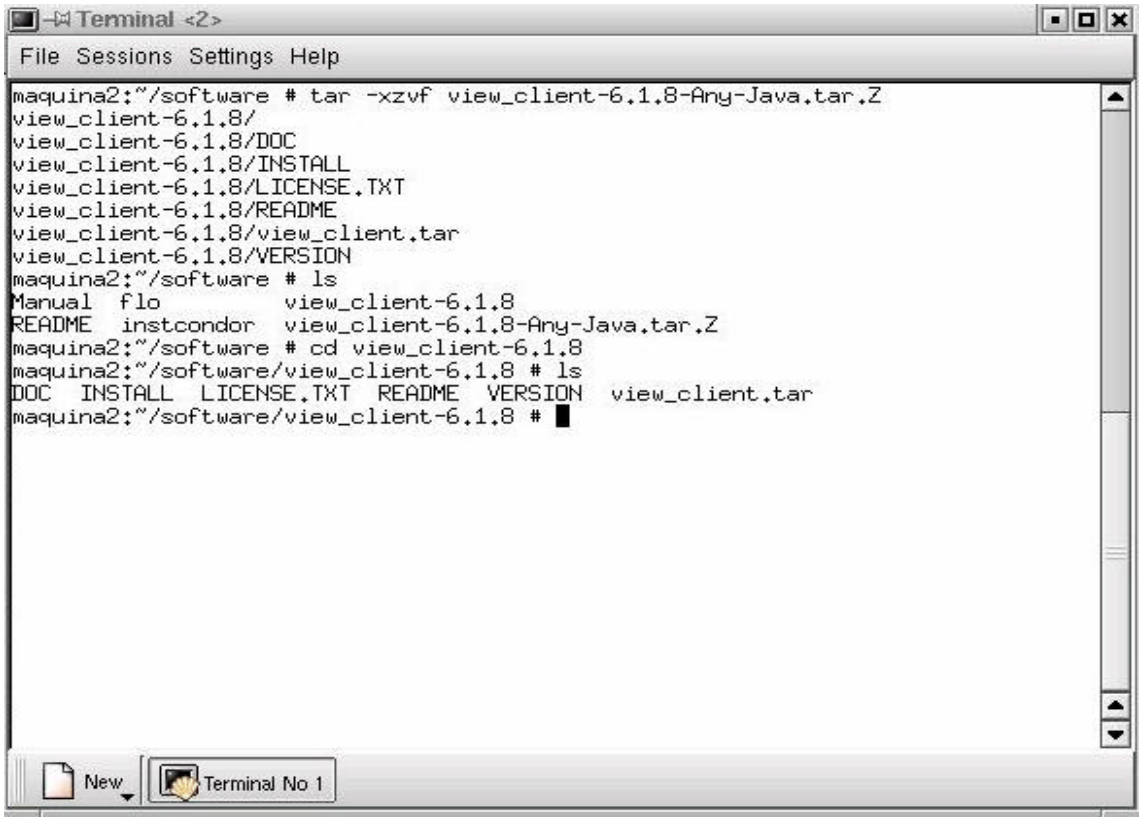

Figura 5. 22 Instalación del módulo Cliente View (I)

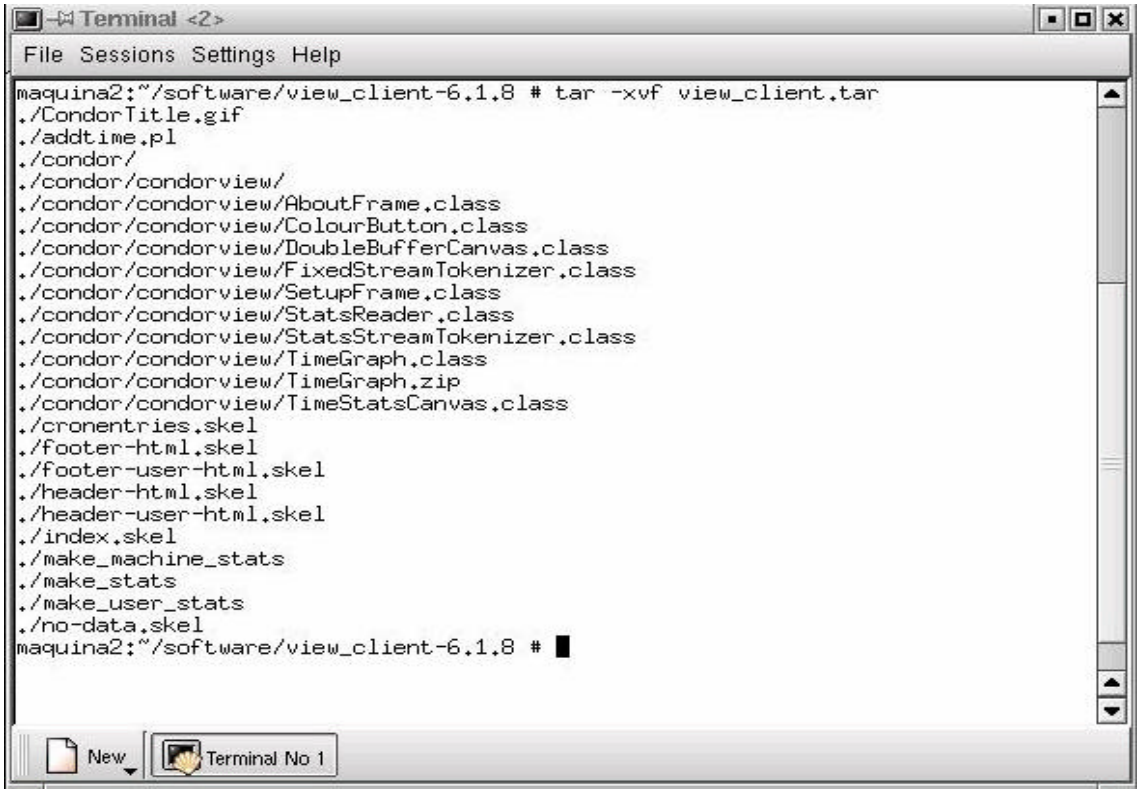

Figura 5. 23 Instalación del módulo Cliente View (II)

Una vez descargado y descomprimido el paquete, se debe editar el script "make_stats". Este fichero contiene seis parámetros que se deben establecer para que el Cliente View funcione correctamente.

ORGNAME

Un breve nombre que identifica una organización. Un ejemplo de este es "Univ. of Wisconsin". No se usan caracteres especiales para el nombre. Se deben evitar los caracteres \, ^, $.

CONDORADMIN

La dirección e-mail del administrador Condor del Condor Pool que se está configurando. Esta dirección aparecerá en la parte inferior de las páginas web.

VIEWDIR

El nombre completo de la ruta del directorio VIEWDIR establecido en la instalación. Dicho directorio debe contener los ficheros java necesarios para realizar las gráficas. Este directorio contiene el script make_stats.

STATSDIR

El nombre completo de la ruta del directorio que contiene el binario condor_stats. El programa condor_stats esta incluido en el directorio <directoriorelease>/sbin a partir de la versión Condor 6.1. El valor STATSDIR se añade al parámetro PATH por defecto.

PATH

Una lista de subdirectorios separados por comas, donde el script make_stats puede encontrar el awk, bc, sed, date, y el programa "condor_stats". Si el módulo perl está instalado [48], la ruta debería también incluir el directorio donde perl está instalado. Se deben comprobar las rutas que vienen por defecto.

```
PATH = /bin:/usr/bin:$STATSDIR:/usr/local/bin
```

Una vez modificados todos los parámetros anteriores se debe ejecutar en la línea de comandos de la máquina el siguiente comando:

```
maquina2:~ # ./make_stats setup
```

De esta manera se generará el fichero index.html inicial. Para verificar que dicha instalación y configuración ha sido correcta se debe abrir el fichero. El fichero que se crea es un fichero estándar que puede ser modificado en cuanto a las necesidades u objetivos que se tengan.

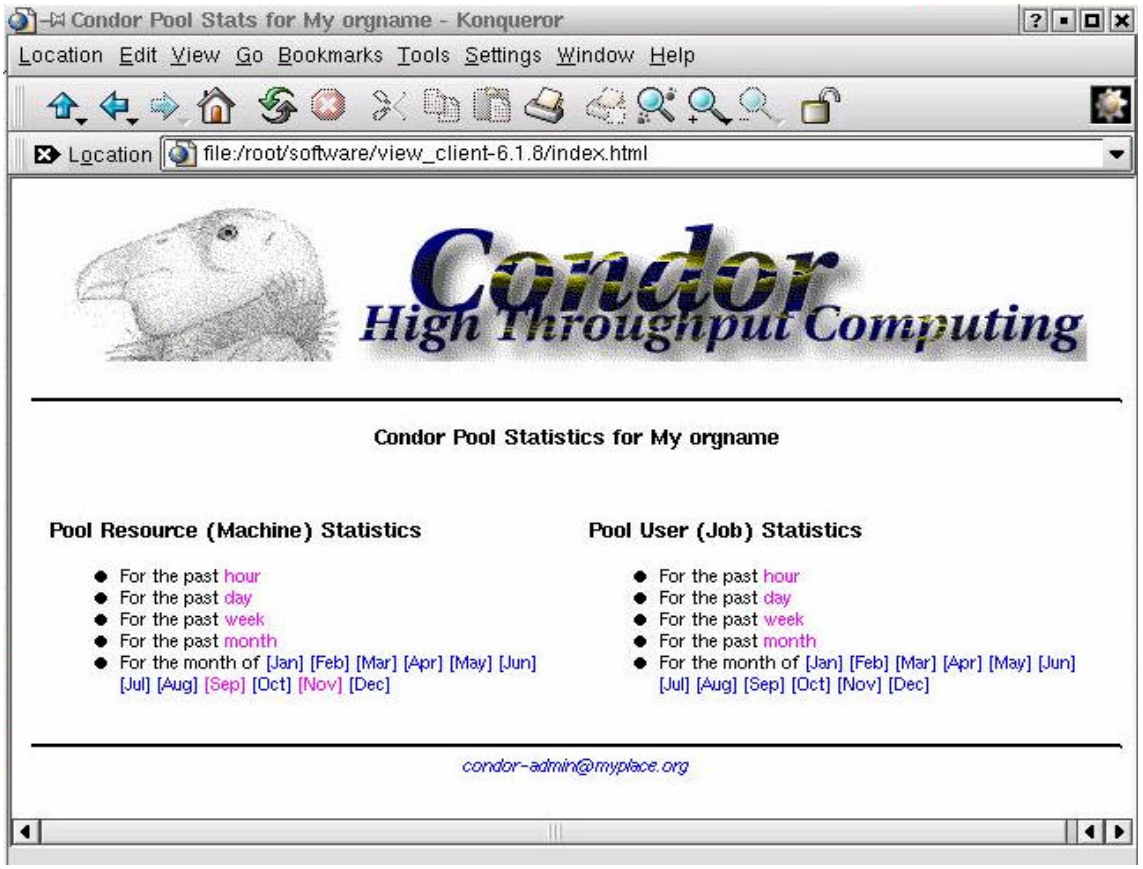

Figura 5. 24 Fichero Index.html inicial

Se puede hacer que se regeneren las gráficas cada cierto tiempo, sincronizando el programa make_machines_stats y make_user_stats, incluidos en el script make_stats. De esta manera, el usuario no debe estar ejecutando el comando cada vez que se quieran ver las gráficas actualizadas. Esto se hace mediante el programa "cron" de Linux [47].

La instalación de este módulo en este proyecto ha sido realizado para poder conocer mejor la información que puede ofrecer esta herramienta de Condor con vistas a una futura instalación, configuración y puesta en marcha, donde se estudiará más a fondo todas las opciones que puede aportar un módulo de estas características (capítulo 6).

5.6 Simulador PASS en Condor

Como se comentó en el capítulo 1, el principal objetivo de este proyecto es la futura instalación de la Aplicación Condor en el laboratorio IT-3 del Área de Ingeniería Telemática de la UPCT. La idea es poder crear un entorno de máximo rendimiento computacional aprovechando los recursos del laboratorio en vistas a las investigaciones que se realizan en dicho área.

Uno de los programas que se someterán a ejecución bajo la Aplicación Condor en este laboratorio será el simulador PASS (Packet Switch Simulator), desarrollado por el Área de Ingeniería Telemática de la UPCT. Como este simulador ha sido desarrollado en lenguaje C++, no habrá problemas para someterlo a Condor, ya que la aplicación soporta los programas en C++.

Para poder manejar dicho simulador de una forma sencilla, y paralelamente a este proyecto, se ha desarrollado un interfaz gráfico programado en lenguaje TCL/TK.

El interfaz ha sido desarrollado para poner a disposición del usuario todos los parámetros necesarios para manejar el simulador y la gestión de las ejecuciones que se realizan de este. Estos son parámetros relacionados con la configuración (número de puertos de entrada y salida, número de etapas…), parámetros relacionados con el tráfico (carga de cada puerto, distribución…), parámetros de duración de la simulación…

Ya se ha comentado que el objetivo final de este proyecto es el de poder realizar un Condor Pool en el laboratorio IT-3 y someter a ejecución bajo la Aplicación Condor el simulador PASS. La situación que se plantea en el laboratorio es la siguiente:

- Condor instalado, configurado, ejecutado en todas las máquinas del laboratorio y listo para dar soporte a las tareas que se sometan a ejecución.
- El simulador PASS y el interfaz gráfico que interactúa con este listos para realizar las gestiones de ejecución del simulador.

Hasta este momento, todo es correcto, pero lo que se pretende con este interfaz gráfica es que el usuario sea capaz de poder gestionar el simulador, los parámetros a introducir, las ejecuciones a realizar, el manejo de resultados obtenidos…y recordando los objetivos planteados, es necesario que dichas ejecuciones del simulador se realicen bajo la Aplicación Condor para así poder aprovechar aquellos recursos infrautilizados y conseguir un mayor rendimiento computacional.

Por tanto, es necesario diseñar el interfaz gráfico de forma que las ejecuciones que se hagan del simulador se sometan bajo la Aplicación Condor y así poder hacer transparente al usuario el uso de esta aplicación.

A partir de este punto es donde se hace necesaria la colaboración en el diseño del interfaz gráfico, para que, además de las gestiones propias que realiza la herramienta en cuanto al simulador, introduzca otras opciones orientadas a la ejecución y control del simulador bajo la Aplicación Condor.

Los objetivos que se plantean en esta colaboración son:

- Conseguir que la interfaz controle la ejecución de las simulaciones de programa PASS que son sometidas bajo la Aplicación Condor.
- Introducir en el diseño de la herramienta diferentes comandos básicos de Condor para permitir al usuario conocer el estado del Condor Pool, someter tareas a ejecución, gestionar la cola de tareas…

Para conseguir dichos objetivos se ha dividido el trabajo en dos partes: Por un lado, se han seleccionado los comandos necesarios para poder someter a ejecución tareas en Condor. Para ello se han analizado los pasos a seguir a la hora de someter una tarea en una máquina con la Aplicación Condor ejecutándose y lista para recibir tareas.

- Como el entorno elegido para la ejecución de tareas es el Universo Standard, se hace necesaria la recompilación de los programas para poder enlazarlos a las librerías de Condor. Para ello se debe utilizar el comando condor_compile [37].

- Una vez creado el ejecutable, este debe ser sometido a Condor bajo un fichero de descripción submit. Para ello, este ejecutable deberá ser indicado en el fichero con los parámetros que se deseen en cuanto a los ficheros que leen la entrada de datos, los ficheros de salida, la opción de ficheros log o error, el uso de Checkpoint, la lectura de datos desde alguna ruta especificada…[39]
- Para someter la tarea se ejecutará el comando condor_submit seguido del nombre del fichero de descripción submit de la tarea que se desea ejecutar [37].

Hay que tener en cuenta que para someter tareas no se puede ser usuario privilegiado Root, por razones de seguridad. Además, se debe tener el ejecutable, el fichero de descripción submit y los ficheros de lectura, en caso de que se utilicen, en el mismo directorio. La ejecución del comando condor_submit se hará desde el directorio donde se encuentra la tarea a someter a Condor.

Recopilados todos los pasos necesarios para someter tareas, se observa que no es fácil hacer que el uso de la Aplicación Condor sea transparente al usuario, porque se deben recompilar los programas con las librerías de Condor, se deben tener ficheros específicos para someter las tareas, se debe trabajar en los directorios donde se encuentren los ejecutables y las tareas…

Por tanto, se ha decidido, que todas las tareas antes comentadas, sean realizadas por el Administrador del simulador, quedando solamente como tarea obligada del interfaz, someter a ejecución los binarios recompilados con Condor mediante unos ficheros de descripción submit ya establecidos. Esto se hará mediante la creación de un vínculo (botón) en el interfaz gráfico. Este llamará al comando condor_submit seguido del nombre del fichero de descripción submit del ejecutable que se desee someter.

Por otro lado se han seleccionado los comandos de mayor interés de Condor para que el usuario pueda tener conocimiento sobre las tareas que se someten a ejecución, gestionar la cola de tareas, conocer el estado del Pool…Algunos de estos comandos son [37]:

- condor_status. Ejecutando este comando, Condor nos da información sobre el estado del Pool, indicando las máquinas que lo forman, el estado de cada una de ellas en ese momento, la arquitectura…
- condor_history. Este comando muestra una lista con todas las tareas Condor especificadas en los ficheros history, que por defecto, se encuentran situados en "/releasedir/spool/history". Si se utiliza el comando condor_history –f "fichero", este comando solo mostrará las características de la tarea de dicho fichero especificado, indicando el ID, OWNER, CPU_USAGE, SIZE…
- condor_q. Muestra información sobre las tareas sometidas en la máquina en la cola de tareas. Se pueden obtener otras informaciones con diferentes opciones como "-global" que muestra la cola de tareas de todas las máquinas del Pool, "-submitter nombremaquina" que muestra solo la cola de tareas de la máquina especificada como submitter…
- condor_rm. Borra una o más tareas de la cola de tareas. También contiene diferentes opciones, como por ejemplo "-all" que borra todas las tareas de la cola de tareas, "cluster nombrecluster" que borra las tareas de un cluster especificado…
- condor_prio. Cambia las prioridades de una o más tareas de una cola de tareas. Este comando puede ser interesante si se tienen varias tareas sometidas en una cola de tareas e interesa que algunas se ejecuten

antes que otras. También posee ciertas opciones como por ejemplo “-a” que cambia la prioridad de todas las tareas de la cola.

Una vez analizados los comandos que pueden ser de utilidad al usuario que somete las ejecuciones del simulador, se deja al programador del interfaz la elección de estos.

Actualmente el desarrollo del interfaz gráfico se esta realizando en un proyecto fin de carrera paralelo a este. Por tanto, los objetivos establecidos sobre el comportamiento que tendrá el interfaz para comunicarse con la Aplicación Condor, serán implementados por el programador de dicho interfaz.

Capítulo 6

Conclusiones y líneas futuras

6.1 Conclusiones

En la introducción de este proyecto se plantearon diferentes objetivos a partir de la idea de aprovechar los ciclos de CPU de las máquinas de los laboratorios del Área de Ingeniería Telemática de la UPCT en el tiempo de inactividad de las mismas.

En este capítulo se analiza si dichos objetivos han sido cubiertos, extrayendo conclusiones sobre los mecanismos empleados, así como las ventajas e inconvenientes de estos.

En la fase inicial del proyecto se ha realizado un estudio sobre la base, estructura y funcionamiento de Condor, analizando así las aportaciones que puede ofrecer una aplicación de estas características a una red de trabajo como la de los laboratorios del Área de Ingeniería Telemática.

Con este estudio previo se ha comprobado que Condor puede ser el software específico para solucionar las necesidades de los laboratorios del Área de Telemática en cuanto a rendimiento computacional. Se ha demostrado de manera teórica que con una aplicación software como Condor se puede solucionar el reparto de tareas entre máquinas de una red, controlando así la ejecución de dichas tareas sometidas a Condor y aprovechando el tiempo de inactividad de los CPU's. Todas estas razones han hecho de Condor una aplicación muy interesante para su estudio y puesta en marcha con vistas a los objetivos comentados en el primer capítulo.

Después del estudio previo realizado se han analizado los recursos tanto hardware como software necesarios para poder instalar Condor, ya que ofrece un amplio abanico de posibilidades (diferentes versiones de la aplicación, diversos sistemas operativos…). Paralelamente se han analizado los diferentes entornos de ejecución que ofrece la aplicación, así como los requisitos de cada uno de ellos con el fin de encontrar aquel que más se acercase a las posibilidades y necesidades planteadas.

Teniendo en cuenta los requisitos del entorno en el que en un futuro se pretende instalar Condor, es decir, una red con NFS y NIS, la elección final de este podría haber sido el Universo Vanilla. Debido a la necesidad de mantener las tareas que se someten a ejecución durante días e incluso semanas, este requisito ha hecho que la elección final haya sido el Universo Standard ya que este universo ofrece la posibilidad de usar mecanismos muy importantes y muy deseados en las redes de trabajo de alto rendimiento computacional: migración, Checkpoint y Remote System Calls.

Con estos mecanismos se consigue, por un lado, que gracias al Checkpoint las tareas puedan ejecutarse durante un largo periodo de tiempo, aunque alguna máquina falle, y por otro lado, que por medio del mecanismo de "Remote System Calls" parezca que las tareas se ejecutan en las máquinas donde han sido sometidas, y todo ello de forma transparente al usuario.

Pero no todo en el Universo Standard ha sido positivo. La elección de este entorno trae consigo una instalación y configuración mucho más compleja, ya que, además de instalar el software de Condor, se debe tener instalado y configurado NFS, NIS y los módulos para utilizar el mecanismo de Checkpoint.

Otro inconveniente que hace complejo este universo es el ejecutar las tareas, ya que se requiere la recompilación de los binarios mediante las librerías de Condor. Este inconveniente puede traer problemas, porque en la mayoría de ocasiones los usuarios no tienen acceso al código fuente del programa o al objeto. Normalmente ocurre con aplicaciones comerciales que solo dan el binario ejecutable, y raramente el código fuente. Este inconveniente no afectará en la futura instalación del Condor Pool ya que ésta está pensada para realizar ejecuciones de programas desarrollados dentro del Área, como es el caso del simulador PASS (*Packet Switch Simulator*).

Aun teniendo todas estas dificultades, se ha decidido realizar una red prototipo para comprobar el funcionamiento de Condor con el entorno elegido. Para ello se ha realizado una red formada por tres máquinas, trabajando bajo el sistema operativo Unix/Linux e interconectadas mediante una red Ethernet.

Se han tenido que instalar y configurar los mecanismos de NFS y NIS en las máquinas de la red propuesta. Una vez comprobado el funcionamiento de estos, se ha instalado y configurado Condor. Para ello, se ha realizado un estudio previo sobre los aspectos a tener en cuenta a la hora de instalar, tales como las características de cada máquina, los recursos de los que dispone, el comportamiento de Condor si el software se instala como un usuario no privilegiado, el funcionamiento de este si dicho software se instala en los directorios a exportar por el sistema de ficheros distribuidos...

Instalado el software, se ha configurado cada una de las máquinas, realizando un estudio de los parámetros que forman parte de los ficheros de configuración para poder elegir la configuración más adecuada ante las necesidades que se plantean en la red. Una vez configuradas, se ha puesto en marcha el Condor Pool, donde se ha comprobado el funcionamiento de este en todas las máquinas mediante tareas ejemplo, algunas, incluidas en el software de Condor y otras realizadas en este proyecto.

Se han hecho pruebas de los comandos y mecanismos más interesantes que ofrece Condor, tales como el Condor View, para obtener información del uso de las CPU's de las máquinas mediante gráficas, el utilización del Checkpoint fuera del entorno del Pool, comandos Condor para utilizar en caso de problemas de ejecución de tareas…

Por último se ha comentado la colaboración con el diseño de un interfaz gráfico que interactúa con el simulador desarrollado en el Área de Telemática y que será una de las tareas a someter en el IT-3. Para someter dicho simulador en Condor mediante el interfaz, se han tenido que introducir diferentes vínculos que hacen las llamadas a los comandos necesarios para poder ejecutar dicha tarea bajo Condor y para poder obtener información sobre esta mientras se encuentra ejecutándose.

Como conclusión final y resumen de la experiencia de trabajar con esta aplicación se puede destacar de Condor que:

- Es una herramienta de gestión de recursos que ofrece múltiples posibilidades en cuanto a trabajo en redes distribuidas ya que permite la ejecución y control de un gran número de programas de una manera simple, pudiendo obtener información de estos en todo momento.
- Mantiene una cola de tareas persistente y el uso de Checkpoint mantiene garantizado el progreso de las tareas, incluso aquellas que su ejecución se extiende durante semanas.
- Maneja un gran número de máquinas de forma sencilla y transparente.

- Los procesos de Condor se ejecutan en todas las máquinas, y se encuentran constantemente monitorizando el estado de las máquinas pudiendo en cualquier momento obtener información sobre las máquinas del Condor Pool mediante comandos y todo esto de forma transparente al usuario.

Por tanto, se demuestra que es posible crear entornos de alto rendimiento de procesamiento con una construcción distribuida de entornos de un hardware cómodo y código fuente abierto, mediante estaciones de trabajo Linux, es decir, se puede crear en un laboratorio del Área de Ingeniería Telemática un entorno eficiente de aprovechamiento de recursos, permitiendo así un mayor rendimiento en las tareas de los investigadores del área en cuanto a tiempo de procesamiento se refiere.

6.2 Líneas futuras

El avance vertiginoso de la informática y las Telecomunicaciones es indiscutible, y su aplicación en las redes de trabajo distribuidas porta múltiples posibilidades, pero estas nuevas tecnologías traen consigo una mayor complejidad en el software, y esto hace que se requiera un gran poder de procesamiento de CPU.

El estudio y aplicación del software de Condor en este proyecto ha sido un paso muy importante hacia la búsqueda del rendimiento computacional, tan necesitado por unas máquinas, y tan desperdiciado por otras.

Cada día surgen nuevas ideas, nuevas versiones para mejorar Condor, y en definitiva para poder mejorar el rendimiento de las redes de trabajo que utilizan un software específico para gestionar las tareas que se someten a ejecución. Por el constante avance de estas aplicaciones, y las nuevas tecnologías que día a día van surgiendo, en un futuro se pretende estudiar y poner en práctica otras posibilidades que ofrece Condor con el fin de aprender las dificultades prácticas que trae consigo la gestión de un Condor Pool.

Se proponen nuevos retos bajo la Aplicación de Condor. El primero de ellos es seguir desarrollando el trabajo realizado en este proyecto. En un futuro, se podría llegar a crear un enlace en el servidor web de la escuela con el fin de dar a conocer el futuro cluster de trabajo de la Universidad (como ocurre en otras universidades que utilizan Condor) por medio de una página que proporcione información sobre el proyecto que se desarrolla, las tareas que se ejecutan, el tiempo de procesamiento de estas, el estado de las máquinas que trabajan en el Pool por medio de gráficas… comenzado a trabajar Se han realizado algunos cambios, como el idioma, la organización a la que pertenece el Pool, los enlaces de la página...

La siguiente figura muestra la modificación del fichero index.html generado por el módulo CondorView (capítulo 5):

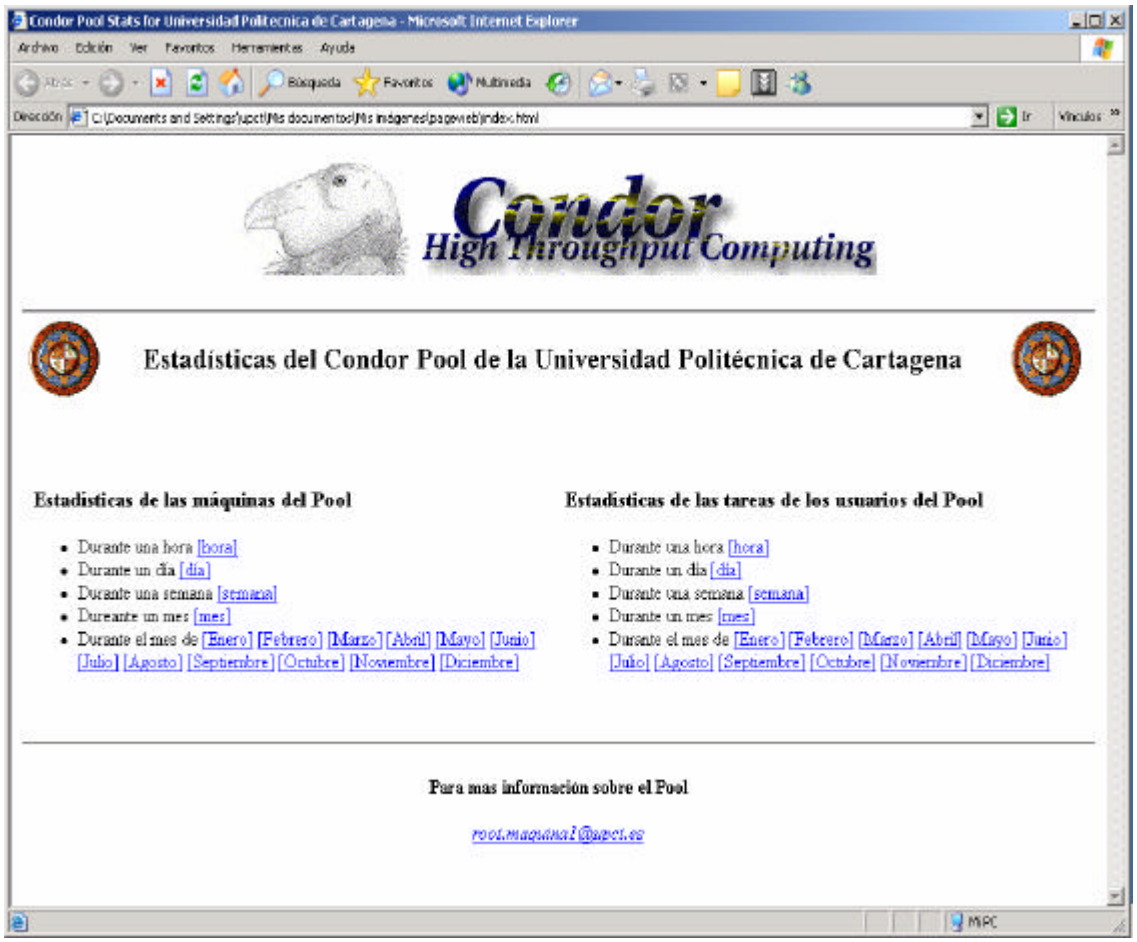

Figura 6. 1 Fichero index.html

También se proponen otros retos como la posibilidad de comunicar varios Condor Pool mediante el mecanismo que ofrece este software, conocido como "Condor Flocking". Este consiste en configurar las máquinas establecidas para someter tareas en un Pool de tal forma que puedan contactar con otros Condor Pools existentes con el fin de ejecutar tareas en esos Pools. Si se configura un Condor Pool con este mecanismo, se establece que dicho Pool también está dispuesto a aceptar tareas de otros Condor Pools.

Con este mecanismo se consigue distribuir la carga de procesamiento generada en un Condor Pool (que en ocasiones puede ser bastante alta) en varios Condor Pools, si estos se encuentran en disposición de procesar dicha carga. Con ello se consigue incrementar más todavía el rendimiento de procesamiento del Condor Pool [58].

La aplicación de este mecanismo puede ser bastante interesante, en vistas a la futura instalación de Condor, pudiendo configurar no uno, sino varios laboratorios del Área de Ingeniería Telemática, tomando cada uno de ellos como un Condor Pool independiente, pero todos ellos utilizando el mecanismo de Condor Flocking, por si se presentan tareas que traen consigo un gran peso computacional.

Se propone también el estudio y puesta en práctica de las aportaciones que llevan las nuevas versiones de Condor. En estas se amplía el abanico de posibilidades en cuanto a los sistemas operativos y versiones de estos que soportan el software. Aparecen nuevas versiones de Condor soportadas por Red Hat 7.x, Linux 2.4.x Kernel, Windows 2000, Mac OSX… haciendo más flexible la instalación de Condor en las redes de trabajo de hoy en día. También, se proporciona una mayor seguridad de la

comunicación en las redes de trabajo mediante múltiples métodos de autentificación, encriptación e integridad.

Además, se introduce un nuevo entorno de ejecución, "Java Universe", ampliando así las posibilidades en cuanto a la ejecución de tareas. Este entorno aparece a partir de las versiones 6.4.X. Se comenta a continuación las ventajas de un entorno como en vistas a una futura instalación debido al interés que genera tener un entorno que sea capaz de ejecutar programas en Java [59].

El universo Java no es solo poner un ejecutable "java" en el parámetro de Universe de un fichero de descripción submit. El Universo java es algo más. Con el entorno Java instalado en un Condor Pool, se puede tener un mayor conocimiento sobre las máquinas de ese Pool que tienen instalado un JVM (Java Virtual Machine). Se obtiene información sobre la localización, versión y rendimiento del JVM de cada máquina. Este entorno aporta mucha más información sobre la tarea Java cuando esta finaliza que el código de salida del JVM en una ejecución normal, en definitiva, es un entorno muy interesante para su estudio y aplicación [60].

Por último, comentar la posibilidad de tomar contacto con diversas universidades en todo el mundo donde se tiene un Condor trabajando, con el fin de conseguir un apoyo en la construcción y seguimiento del Condor Pool, llegando así a conseguir, en un futuro, un cluster de trabajo de Condor en la Universidad Politécnica de Cartagena, y así formar parte de las más de 4000 computadoras que hoy en día hay trabajando con el software de Condor en el mundo, donde más de 1200 de estas se encuentran en la Universidad de Winsconsin, donde tuvo origen la Aplicación Condor, más de 200 se encuentran en INFN (Institute di Fisica Nucleare di Bologna), y más de 800 se encuentran en la industria (Oracle es una de las compañías que hoy en día esta construyendo un cluster de Condor).

Apéndice A

Daemon Core y seguridad en Condor

A.1 Librería DaemonCore

DaemonCore es una librería que se distribuye a través de la mayoría de los procesos Condor y la cual provee a estos de una funcionalidad común [50]. Normalmente, los procesos Condor que usan esta librería son condor_master, condor_startd, condor_schedd, condor_collector, condor_negotiator y condor_kbdd.

DaemonCore provee una interfaz uniforme para los demonios a varias señales UNIX, y provee de unas opciones comunes en la línea de comandos que pueden ser usadas para ejecutar dichos demonios.

La mayoría de los detalles del DaemonCore no son de interés para el administrador, pero se define esta librería ya que algunas de las opciones que trae consigo deben conocerse para que no surjan dudas en un futuro cuando se ejecuten comando de Condor.

Una de las figuras más visibles que DaemonCore provee para los administradores es que todos los demonios que lo usan se comportan de la misma manera ante ciertas señales UNIX. Las señales y el comportamiento de DaemonCore se listan en los siguientes párrafos:

SIGHUP. Causa la reconfiguración del demonio.

SIGTERM. Causa la caída gradual del demonio.

SIGQUIT. Causa la caída rápida del demonio.

Hay que especificar que la caída gradual y la caída rápida dependen del proceso. Para demonios como por ejemplo kbdd, collector o negotiator no hay diferencia entre ambas señales.

En el caso la reconfiguración, la actuación de los demonios ante esta señal será la de volver a leer el /los ficheros de configuración y establecer los cambios que se hayan producido para que hagan efecto.

Otro elemento que se puede observar en cuanto a lo que ofrece el DaemonCore a los administradores es un conjunto de argumentos para establecer en la línea de comandos que todos los demonios entienden. Se destaca –f que causa que el demonio se ejecute en primer plano. Se debe saber que cuando el “condor_master” lanza los demonios, lo hace con esta opción.

Ejemplo:

```
maquina1:~ # ps -ef | egrep condor_
condor           4912             1     0   00:47   ?                    00:00:00
/usr/local/condor/sbin/condor_master
condor    4913  4912  0 00:47 ?        00:00:00 condor_collector -f
condor    4914  4912  0 00:47 ?        00:00:00 condor_negotiator -f
condor    4915  4912  2 00:47 ?        00:00:01 condor_startd -f
```

A.2 Niveles de acceso de seguridad

Es necesario conocer los diferentes niveles de acceso que pueden tener las diferentes figuras de un Condor Pool para hacer una red segura. Dependiendo del nivel de acceso que se les de a las figuras del Condor Pool, se podrá tener acceso a la información del Condor Pool, se podrán ejecutar o no tareas Condor, se podrá tener información acerca de las colas de tareas, las máquinas que forman parte del pool... Por ello hay que tener muy en cuenta este apartado para realizar una buena configuración. Los diferentes niveles de acceso son [51]:

A.2.1 Acceso READ

Las máquinas con acceso Read, pueden leer información de Condor. Por ejemplo, estas pueden ver el estado del pool, la cola de tareas o ver los permisos de los usuarios. Básicamente, una máquina listada con el permiso Read no puede tomar parte del Condor pool ejecutando tareas, solo puede tener un vistazo de la información sobre el pool.

Por defecto, si no se modifican los parámetros referentes a los permisos de lectura en el fichero de configuración, el acceso de lectura está garantizado para todo el mundo. Esto se puede modificar para que esté restringido y solo tengan este nivel de acceso las máquinas del dominio en el que se trabaja. Si es posible, se debe también garantizar el acceso a "*.cs.wisc.edu", así, los desarrolladores de Condor serán capaces de visualizar el estado del pool y ayudarnos de una manera más fácil, instalando, configurando… En los ficheros de configuración, aparecen los siguientes parámetros configurados por defecto y que se deben de ajustar a las necesidades de la red.

```
HOSTALLOW_READ = *
#HOSTALLOW_READ = *.your.domain, *.cs.wisc.edu
#HOSTDENY_READ = *.bad.subnet, bad-machine.your.domain, 144.77.88.*
```

A.2.2 Acceso WRITE

Las máquinas con el acceso Write pueden escribir información de Condor, someter tareas, es decir, que pueden tomar parte del pool enviando actualizaciones ClassAd al central manager y pueden contactar con otras máquinas del pool para someterlas o ejecutar trabajos. Además, cada máquina con el acceso Write puede preguntar al condor_startd.

Importante: Una máquina que realiza tareas en el condor pool debe tener permisos de lectura y escritura en el pool (solo permiso de escritura no es suficiente.) Si se dejan los valores por defecto que aparecen en el fichero de configuración, estarán sin especificar, y efectivamente, permitirá el acceso de escritura a cualquiera en el pool.

```
HOSTALLOW_WRITE = *
#HOSTALLOW_WRITE = *.your.domain, your-friend's-machine.other.domain
#HOSTDENY_WRITE = bad-machine.your.domain
```

A.2.3 Acceso ADMINISTRATOR

Las máquinas con el acceso de ADMINISTRATOR tienen unos derechos de administrador de Condor especiales en el pool. Esto incluye cosas como cambio de las prioridades del usuario (con el comando “condor_userprio_set”), cambio Condor “on” a “off” (mediante el comando condor_off), preguntando a Condor para reconfigurar o

recomenzar, etc... Es típico que se quiera solo una pareja de máquinas en esta lista, quizás las estaciones de trabajo donde los administradores de condor o “sysadms” normalmente trabajan, o quizás solo el Central Manager de Condor.

Por defecto, este parámetro viene establecido con el hostname del Central Manager. Como se ha explicado antes, si se quiere que otras tengan acceso administrator (asumiendo una total responsabilidad a todos los usuarios de las máquinas que tienen el acceso garantizado.)

HOSTALLOW_ADMINISTRATOR = $(CONDOR_HOST)

Si no se desea que otras máquinas asuman el cargo de nivel de acceso administrator en el pool (por ejemplo porque no hay ninguna máquina en el pool la responsabilidad de los usuarios este garantizada), se puede quitar el comentario a la siguiente línea. Desafortunadamente, significará que la administración en el pool será mas difícil.

#HOSTDENY_ADMINISTRATOR = *

A.2.4 Acceso OWNER

Este nivel de acceso se requiere para comandos que el propietario de una máquina (o un usuario local) debería ser capaz de usar, además de los administradores de Condor. Por ejemplo, el comando “condor_vacate” que causa el “condor_startd” para dejar vacante una tarea Condor que esta ejecutándose se registra con el permiso OWNER, así, cualquiera puede hacer condor_vacate a la máquina local y ser registrado.

Por defecto, este parámetro queda establecido con las máquinas que se establecen con nivel de acceso administrator y la máquina local. Esta configuración es la más acertada.

HOSTALLOW_OWNER = $(FULL_HOSTNAME),
$(HOSTALLOW_ADMINISTRATOR)

A.2.5 Acceso NEGOTIATOR

Este nivel de acceso significa que el comando especificado debe venir del Central Manager del pool. Los comandos que tienen este nivel de acceso son los que llaman al comando “condor_schedd” para comenzar a negociar y son los que llaman un condor_startd que ha sido emparejado al condor_schedd con las tareas a ejecutar. No se debería de cambiar este parámetro.

HOSTALLOW_NEGOTIATOR = $(NEGOTIATOR_HOST)

HOSTALLOW_NEGOTIATOR_SCHEDD = $(NEGOTIATOR_HOST),
$(FLOCK_NEGOTIATOR_HOSTS)

A.2.5 Acceso CONFIG

Este nivel de acceso se requiere para modificar la configuración de los demonios sudando el comando “condor_config_val”. Las máquinas con este nivel de acceso serán capaces de cambiar los parámetros de configuración, excepto aquellos especificados en el fichero de configuración condor_config.root. Además, este nivel de acceso debe ser garantizado con extrema precaución. Por defecto, el acceso CONFIG, esta denegado para todos los hosts.

#HOSTALLOW_CONFIG = trusted-host.your.domain

Apéndice B

Parámetros en ficheros de Condor

B.1 Ficheros de configuración

Los ficheros de configuración de Condor se usan para determinar como Condor va a trabajar en una red dada. La configuración básica por defecto debe ser modificada para poder trabajar.

Cada programa Condor, como parte del proceso de iniciación, se configura así mismo mediante la llamada de una rutina. El resultado es una lista de constantes y expresiones que son evaluadas en tiempo de ejecución [52].

El orden en el que los atributos se definen es importante, ya que definiciones posteriores sobrescribirán las definiciones existentes.

B.1.1 Macros del fichero de configuración

Las definiciones de macro son de la forma

<nombre de la macro> = <definición de la macro>

Las invocaciones de macro son de la siguiente forma:

$(nombre de la macro)

Las definiciones de la macro pueden contener referencias a otras macros, incluso a algunas que no estén todavía definidas.

A = xxx

C = $(A)

Este establecimiento de macros es correcto, y el resultado de C es xxx. Otro ejemplo puede ser:

A = xxx

C = $(A)

A = yyy

Es también un macro correcto, y el resultado de C en este caso es yyy. Una macro puede ser definida por invocación a si misma. Por ejemplo:

A = xxx

B = $(A)

A = $(A) yyy

A = $(A) zzz

Es un establecimiento de macro correcto, y el resultado de A en este caso será xxxyyyzzz.

B.1.1.1 Comentarios y líneas

Los ficheros de configuración pueden contener también comentarios o continuación de línea. Un comentario comienza con el carácter “#”.

Una continuación es una entrada que continua a través de múltiples líneas. Una continuación de línea se realiza mediante el carácter “\” y al final de la línea se continúa con otra. Algunos de los ejemplos validos se muestran a continuación.

```
START = (KeyboardIdle > 15 * $(MINUTE)) && \
((LoadAvg - CondorLoadAvg) <= 0.3)
ADMIN_MACHINES = maquina1.upct.es, maquina2.upct.es, maquina3.upct.es .
HOSTALLOW_ADMIN =  $(ADMIN_MACHINES)
```

B.1.1.2 Macros predefinidas

Condor esta provisto de unas macros predefinidas que ayudan a configurarlo. Las macros predefinidas se listan como $(nombre_macro).

Las primeras macros se determinan en tiempo de ejecución y no se pueden sobrescribir. Estas se insertan automáticamente por la librería de rutina, la cual pasa a los ficheros de configuración.

$(FULL_HOSTNAME). El nombre completo de la máquina local (es el hostname más el nombre de dominio).

$(HOSTNAME). El nombre de la máquina local (pero no su dominio).

$(TILDE). El nombre completo de la ruta del directorio home del usuario Condor en Unix, si tal usuario existe en la máquina local.

$(SUBSYSTEM). El nombre del subsistema de los procesos o herramientas que están evaluando la macro. Este es una cadena única que identifica un demonio dado con el sistema Condor. Los posibles nombres de subsistema son:

- STARTD
- SCHEDD
- MASTER
- COLLECTOR
- NEGOTIARTOR
- KBDD
- SHADOW
- STARTER
- CKPT_SERVER
- SUBMIT

Las siguientes macros son entradas que son valores por defecto determinadas automáticamente en tiempo de ejecución pero que pueden sobrescribirse.

$(ARCH). Define el string usado para identificar la arquitectura de la máquina local de Condor.

$(OPSYS). Define el string usado para identificar el sistema operativo de la máquina local de Condor. Si no está definido en el fichero de configuración, automáticamente Condor insertará como nombre del sistema operativo “uname”.

Las macros indicadas automáticamente se establecen con los valores correctos, con lo cual no se recomienda su modificación a no ser que se sepa lo que se está haciendo.

B.1.2 Ficheros de configuración

Todos los ficheros de configuración, ya sean los ficheros de configuración del Central Manager, Submit Only o Full-Install, está dividido en tres partes:

Parámetros locales

Parámetros que pueden ser modificados.

Parámetros que deberían dejarse por defecto.

Algunos de los parámetros que se definen, muestra información sobre el Condor Pool que se ha detallado durante la instalación de Condor como por ejemplo:

```
##  What machine is your central manager?
CONDOR_HOST       = 192.168.19.1

##--------------------------------------------------------------------
##  Pathnames:
##--------------------------------------------------------------------
##  Where have you installed the bin, sbin and lib condor directories?
RELEASE_DIR       = /usr/local/condor

##  Where is the local condor directory for each host?
LOCAL_DIR         = /root/condor
#LOCAL_DIR        = $(RELEASE_DIR)/hosts/$(HOSTNAME)

##  Where is the machine-specific local config file for each host?
LOCAL_CONFIG_FILE = $(LOCAL_DIR)/condor_config.local
#LOCAL_CONFIG_FILE      = $(RELEASE_DIR)/etc/$(HOSTNAME).local

##--------------------------------------------------------------------
##  Mail parameters:
##--------------------------------------------------------------------
##  When something goes wrong with condor at your site, who should get
##  the email?
CONDOR_ADMIN            = root@linux.local

##  Full path to a mail delivery program that understands that "-s"
##  means you want to specify a subject:
MAIL              = /usr/bin/mail
```

Otros, deben ser modificados para el correcto funcionamiento de la aplicación. Para ello se muestran a continuación, todas las partes diferenciadas de los ficheros de configuración que se pueden encontrar y su posible modificación para el correcto funcionamiento de la aplicación según nuestras necesidades.

B.1.2.1 Configuración del sistema de ficheros distribuido

Estas macros controlan como Condor interactúa con varios sistemas de ficheros de redes de trabajo y distribuidos.

Los siguientes parámetros están establecidos a los valores que se han indicado en los pasos de la instalación del fichero “condor_install”.

UID_DOMAIN

A veces, especialmente si todas las máquinas en del Condor Pool están administradas por la misma organización, para ser añadidas en el Condor pool, distribuyen el mismo registro de información. Cada usuario tiene una UID concreta y con un dominio Internet/DNS dado. Este es el estado normal en caso de que actue

una central autoritaria (NIS) que crea registros y los mantiene en un fichero común /etc/passwd en todas las máquinas. Si este es el caso de la red, entonces la macro se establece con el nombre del dominio Internet/DNS.

Condor usa esta información para determinar si debe ejecutar una tarea Condor dada en la máquina Execute remota con el UID de quien es sometida la tarea o el UID del usuario de UNIX “nobody”.

Si el macro se establece a “none” o no se establece, las tareas de Condor siempre se ejecutaran con los permisos “access” de el usuario “nobody”. Por razones de seguridad, no es una buena idea el tener tareas de Condor migrando por una red dada de una organización para ejecutarlas como el usuario “nobody”, ya que tiene los accesos bastante restringidos, sobre todo el acceso a los ficheros de disco de las máquinas.

Las tareas Condor del Universo Standard se ejecutan como usuario “nobody” desde todas las IN/OUT y se redireccionan a través del uso de llamadas del sistema remoto a un proceso shadow ejecutándose en una máquina Submit.

Si solo se planea el ejecutar tareas con el universo Standard, es mejor dejar este parámetro establecido a “none” u omitirlo.

Las tareas del Universo Vanilla dependen de NFS, RFS AFS o algunos sistemas de ficheros distribuidos establecidos para lectura y escritura de ficheros. Si se quieren ejecutar tareas Vanilla y el sistema de ficheros es vía AFS, se puede dejar como usuario “none”, pero si se desea ejecutar tareas Vanilla mediante NFS o RFS, se debe establecer el parámetro al nombre de dominio de la red.

```
UID_DOMAIN   = upct.es
#UID_DOMAIN          = $(FULL_HOSTNAME)
```

FILESYSTEM DOMAIN

Es similar al parámetro UID_DOMAIN, pero en este caso es el nombre de dominio Inernet/DNS donde todas las máquinas con ese dominio pueden acceder al mismo servidor de ficheros NFS. Además, si todas las máquinas en el Condor Pool son administradas por la misma organización, todas las máquinas se añaden al mismo Condor Pool para poder montar el mismo servidor de ficheros NFS en el mismo lugar en el directorio árbol. Si todas las máquinas en el Condor Pool con un especifico dominio Internet/DNS montan el mismo servidor de ficheros NFS en la misma ruta de punto de montaje, entonces, se debe establecer la macro con el nombre del dominio Internet/DNS donde ocurre esto.

```
 FILESYSTEM_DOMAIN        = local
#FILESYSTEM_DOMAIN        = $(FULL_HOSTNAME)
```

USE NFS

Esta macro influye a la manera en la que las tareas de Condor se ejecutan en el Universo Standard. Condor redirecciona el fichero pedido para ser ejecutado en la máquina que somete la tareas. Por ello, como la tarea Condor migra por toda la red de trabajo, el sistema de ficheros siempre aparece para identificar al sistema de ficheros donde la tarea es sometida.

Si la macro se establece a TRUE, Condor intentará leer o escribir ficheros sin redireccionarlos a la máquina Submit si ambas, la máquina Submit y la máquina Execute acceden al mismo servidor NFS.

Establecer el parámetro a FALSE es lo más seguro. Puede tener como resultado mayor tráfico en la red, pero asegura que las tareas remotas de Condor en el Universo Standard siempre usaran mecanismo de llamadas del sistema remotas para enrrutar la

entrada y salida y así ver exactamente el mismo sistema de ficheros que ve el usuario en la máquina donde esta sometiendo la tarea.

#USE_NFS = False

HAS_AFS. USE_AFS

Ambos parámetros se establecerán a false, ya que no interesa el uso de AFS en Condor, como se comentó anteriormente.

#HAS_AFS = False

#USE_AFS = False

B.1.2.2 Configuración del Servidor Checkpoint

Las macros que establecen la configuración del Servidor Checkpoint se dividen en varios ficheros.

Fichero de configuración del Servidor Checkpoint

Situándose en el fichero de configuración del Servidor Checkpoint, las macros que se pueden encontrar son las siguientes:

CKPT_SERVER_DIR

Este parámetro indica donde estarán localizados los ficheros checkpoint. Es mucho mejor si dicha localización es una ruta rápida (como por ejemplo el directorio raíz). La rapidez del sistema de ficheros tendrá un impacto directo en la rapidez con la que los ficheros serán reenviados a la máquina remota.

CKPT_SERVER_DIR = /ckpt_server

DAEMON_LIST

Para que el Servidor Checkpoint pueda estar controlado por el "condor_master", esta entrada debe de tener MASTER y CKPT_SERVER. Se debe añadir STARD si se quieren mantener tareas ejecutándose en el Servidor Checkpoint. De la misma manera hay que añadir SCHEDD si se quieren someter tareas. EL resto de las opciones depende de la versión de Condor que se este utilizando.

CKPT_SERVER_LOG

Indica donde se sitúa el log del Servidor Checkpoint.

CKPT_SERVER_LOG = $(LOG)/CkptServerLog

MAX_CKPT_SERVER_LOG.

Establece el tamaño máximo del log del Servidor Checkpoint antes de que se guarde y el fichero log se reinicie.

MAX_CKPT_SERVER_LOG = 640000

CKPT_SERVER_DEBUG.

Regula la cantidad de información pintada en el fichero log. Normalmente el único nivel debug soportado es D_ALWAYS.

CKPT_SERVER_DEBUG = D_ALWAYS

Fichero de configuración global

Por otro lado en el fichero de configuración global aparecen los siguientes parámetros:

USE_CKPT_ SERVER

Este parámetro debe establecerse a TRUE (por defecto).

USE_CKPT_SERVER = True

CKPT_SERVER_ HOST

En este parámetro debe establecerse el nombre completo de la máquina que esta ejecutando el Servidor Checkpoint.

Es más conveniente establecer los parámetros en el fichero de configuración global. De esta manera, los cambios producidos en este fichero afectarán a todas las máquinas sometidas. De cualquier manera, se puede configurar cada máquina sometida por separado, (usando en este caso el fichero de configuración local) en el caso de que no se quiera que todas las máquinas que forman parte del pool usen el Servidor Checkpoint al mismo tiempo. Si se establece el parámetro USE_CKPT_SERVER a FALSE, la máquina sometida no usará el Servidor Checkpoint.

CKPT_SERVER_HOST = checkpoint-server-hostname.your.domain

B.1.2.3 Configuración condor_master

DAEMON LIST

Esta macro determina que demonios ejecutará condor_master y mantendrá monitoreando. La lista de estos esta separada por comas.

DAEMON_LIST = MASTER, STARTD, SCHEDD

En el Central Manager, la lista de demonios será diferente de la del pool, ya que esta incluye los demonios condor_collector y condor_negotiator.

DAEMON_LIST = MASTER, COLLECTOR, NEGOTIATOR, STARTD, SCHEDD

En las máquinas ejecutando Digital Unix o Irix, la lista de demonios incluirá también KBDD, para el demonio condor_kbdd, que es un demonio especial que se ejecuta para monitorear la actividad del teclado y del ratón en la consola.

DC_DAEMON_LIST

Esta macro lista los demonios de DAEMON_LIST que utilizan la librería de Condor DaemonCore. El condor_master debe diferenciar entre los demonios que usan el DaemonCore y aquellos que no lo usan y usan un mecanismo de comunicación entre procesos adecuado. Esta lista incluye todos los demonio de Condor exceptuando al Servidor Checkpoint.

SUBSYS

Una vez que se han definido los subsistemas que condor_master va a ejecutar, se debe de indicar la ruta completa de cada uno de los binarios:

MASTER = $(SBIN)/condor_master

STARTD = $(SBIN)/condor_startd

SCHEDD = $(SBIN)/condor_schedd

PREEN

Además de los demonios definidos en la lista de demonios DAEMON_LIST, el condor_master también ejecuta algunos procesos especiales como es el caso de condor_preen. Si se comenta la macro, condor_preen no se ejecutará.

PREEN = $(SBIN)/condor_preen

PREEN_ARGS

Esta macro controla como se comporta condor_preen para mantener la especificación de los argumentos en la línea de comandos. Por defecto vienen los argumentos –m y –r.

La opción "–m" significa que se quiere un e-mail sobre ficheros que condor_preen encuentra y borra. La opción " –r" significa que se quiere condor_preen para borrar esos ficheros.

PREEN_ARGS = -m -r

PREEN_INTERVAL

Esta macro determina cada cuanto tiempo se debe de ejecutar condor_preen. Se establece en segundos y por defecto son 86400, es decir, una vez al día.

PREEN_INTERVAL = 86400

PUBLISH_OBITUARIES.

Cuando un demonio cae, el condor_master puede enviar un e-mail a la dirección especificada en la instalación de Condor, informando sobre la caída del demonio, y la causa de esta. Si se quiere esta opción, se debe de establecer a True.

PUBLISH_OBITUARIES = True

START_MASTER.

Si este parámetro se establece a False, cuando condor_master se ejecute, lo primero que hará será terminar. Esto parece extraño, pero se pueden dar casos en que no se quiera ejecutar Condor en ciertas máquinas en el Condor Pool.

START_MASTER = True

START_DAEMONS.

Esta macro es similar a la anterior. Si no se inician los demonios con esta variable, se pueden iniciar después mediante el comando condor_on.

START_DAEMONS = True

MASTER_UPDATE_INTERVAL.

Este parámetro determina cada cuanto tiempo el condor_master envía actualizaciones al condor_collector. Se define en segundos y por defecto a 300 (5 minutos).

MASTER_UPDATE_INTERVAL = 300

MASTER_CHECK_NEW_EXEC_INTERVAL

Esta macro controla cada cuanto el condor_master chequea el tiempo de ejecución de los demonios. Si un demonio ha sido modificado, el condor_master lo reinicia. Se define en segundos y por defecto a 300 (cada 5 minutos).

MASTER_CHECK_NEW_EXEC_INTERVAL = 300

MASTER_NEW_BINARY_DELAY

Una vez que condor_master descubre un nuevo binario, este parámetro controla cuanto tiempo tiene que esperar antes de intentar ejecutar el binario. Este retraso existe ya que el condor_master debe notificar el nuevo binario mientras esta siendo copiado, e intentará ejecutarlo sin resultado alguno. Este parámetro esta definido en segundos y por defecto a 120 (2 minutos).

MASTER_NEW_BINARY_DELAY = 120

SHUTDOWN_FAST_TIMEOUT

Este parámetro determina la capacidad máxima de tiempo que tienen los demonios para realizar el procedimiento de finalización antes de que condor_master los mate. Se define en segundos y por defecto a 300 (5 minutos).

SHUTDOWN_FAST_TIMEOUT = 120

MASTER_BACKOFF_FACTOR

Si un demonio se mantiene caído hay un tiempo de backoff exponencial que pasa antes de volver a iniciarlo. Por defecto son 2 segundos.

MASTER_BACKOFF_FACTOR = 2.0

MASTER_BACKOFF_CEILING

Este parámetro determina el tiempo máximo que debe esperar el condor_master entre intentos para iniciar demonios. Se define en términos de segundos y por defecto es 3600 (1hora).

MASTER_BACKOFF_CEILING = 3600

MASTER_RECOVER_FACTOR

Esta macro establece cuanto tiempo necesita un demonio para ejecutarse sin caer. Una vez que el demonio ha sido recuperado, el numero de reiniciaciones se resetea, así, el backoff exponencial vuelve al estado inicial. Esta macro se define en términos de segundos y por defecto se establece a 300 (5 minutos).

MASTER_RECOVER_FACTOR = 300

B.1.2.4 Configuración condor_schedd

SHADOW

Esta macro determina la ruta completa del binario condor_shadow que lanza el condor_schedd.

SHADOW = $(SBIN)/condor_shadow

SHADOW_PVM

Esta macro determina la ruta completa del binario especial condor_shadow.pvm usado para soportar tareas PVM que condor_schedd lanza.

SHADOW_PVM = $(SBIN)/condor_shadow.pvm

Al igual que para tareas PVM, también aparece este parámetro para otros universos como Globus, o para otros sistemas operativos como Windows NT.

SHADOW_GLOBUS = $(SBIN)/condor_shadow.globus

SHADOW_NT = $(SBIN)/condor_shadow.v61

SCHEDD_ADDRESS_FILE

Cada demonio de Condor que usa la librería DaemonCore tiene un puerto de comandos donde son enviados estos. La IP y el puerto del demonio se ponen en el ClassAd. Esto es así para que otras máquinas en el pool puedan encontrar la dirección de un demonio dado en una máquina dada preguntando simplemente al condor_collector (que lista los puertos conocidos). Las herramientas y los demonios que quieran comunicarse y se encuentren en la misma máquina no hacen uso del

condor_collector, solamente se buscara en un fichero del disco local para encontrar el IP/puerto.

Estableciendo esta macro, los demonios para escribir el IP/puerto de los comandos se comunicarán con un fichero específico. De esta manera, las herramientas locales continuarán operando incluso si condor_collector cae.

Usando este fichero también se generará menos tráfico en el Condor Pool (ya que condor_q, condor_rm y otros no tendrán que enviar mensajes sobre la red para localizar el condor_schedd). Esta macro no es utilizada ni por el collector ni por el negotiator ya que la comunicación de los comandos es con el puerto conocido.

SCHEDD_ADDRESS_FILE = $(LOG)/.schedd_address

SCHEDD_NAME = root@$(FULL_HOSTNAME)

SCHEDD_INTERVAL.

Esta macro determina cada cuanto el schedd envía actualizaciones al condor_collector. Se define por defecto en términos de segundos con un valor de 300 (cada 5 minutos).

SCHEDD_INTERVAL = 300

JOB_START_DELAY

Cuando el condor_schedd termina de negociar y tiene nuevas máquinas en el pool atentas a la ejecución de tareas, este puede retrasar el lanzamiento de condor_shadow para ejecutar las tareas. Ese retraso se define en esta macro. Este macro determina pues cada cuanto tiempo el condor_schedd debe esperar entre lanzamientos de condor_shadow.

También esta macro es usada para determinar el tiempo que debe pasar para la caída gradual de condor_schedd. Durante la caída gradual, esta macro determina cuanto tiempo tiene que esperar entre las preguntas a cada condor_shadow. Se define en términos de segundos y por defecto es 2.

JOB_START_DELAY = 2

ALIVE_INTERVAL

Esta macro determina cada cuanto tiempo el condor_schedd debería enviar un mensaje a cada startd que esta en estado de espera de tarea. Se define en términos de segundos y por defecto es 300 (5 minutos).

ALIVE_INTERVAL = 300

MAX_SHADOW-EXCEPTIONS

Este macro controla el numero máximo de veces que los procesos condor_shadow pueden tener un error fatal (excepción).

MAX_SHADOW_EXCEPTIONS = 5

SHADOW_SIZE_ESTIMATE.

Esta macro establece el tamaño de memoria virtual estimada de cada proceso condor_shadow. Se establece en KiloBytes y el valor por defecto varia dependiendo de la plataforma.

SHADOW_SIZE_ESTIMATE = 1800

SHADOW_RENICE_INCREMENT

Cuando el schedd lanza un nuevo condor_shadow, puede hacerse con un buen nivel (nice-level). Un buen nivel en Uníx es un mecanismo que permite a los usuarios

asignarles a procesos propios escasas prioridades. De esta manera los procesos no interfieren con el uso interactivo de la máquina.

SHADOW_RENICE_INCREMENT = 10

QUEUE_CLEAN_INTERVAL

El schedd es el que mantiene la cola de tareas en una máquina dada. Lo hace de tal manera que si el schedd cae, se puede recuperar un estado valido de la cola de tareas. El mecanismo que usa es un fichero log (job_queue.log). El fichero contiene el estado inicial de la cola de tareas, y una serie de transacciones realizadas en la cola (nuevas tareas sometidas, tareas terminadas...). Periódicamente el schedd trunca todas las transacciones y crea un nuevo fichero que contiene solo el nuevo estado inicial del log. Este macro determina cada cuanto tiempo el schedd debería limpiar la cola de tareas. Esta definido en términos de segundos y por defecto son 86400 (una vez al día).

QUEUE_CLEAN_INTERVAL = 86400

ALLOW_REMOTE_SUBMIT

Desde la versión 6.0 de Condor, los usuarios pueden ejecutar condor_submit en una máquina y normalmente someter las tareas en otra máquina del pool. Esto se llama “remote submit”. Las tareas sometidas de esta manera entran en la cola de tareas propia del usuario nobody. Esta macro determina si se mantendrá o no y por defecto esta establecida a FALSE.

ALLOW_REMOTE_SUBMIT = False

QUEUE_SUPER_USERS

Este macro determina que el nombre de usuario de una máquina dada tiene acceso super-user a la cola de tareas, significando que ellos pueden modificar o borrar las tareas del ClassAds de otros usuarios

QUEUE_SUPER_USERS = root, condor

B.1.2.5 Configuración condor_shadow

MAX_DISCARDED_RUN_TIME

Si el shadow no es capaz de leer el fichero checkpoint desde el servidor Checkpoint, se queda intentándolo solo si la tarea ha acumulado mas segundos que lo usados por el CPU. Por defecto se establece en 3600 (1 hora). Este parámetro solo se usa si la macro $(USE_CPT_SERVER) esta establecido a TRUE.

MAX_DISCARDED_RUN_TIME = 3600

B.1.2.6 Configuración condor_shadow.pvm

Estas macros controlan el condor_shadow.pvm, el shadow especial que soportan las tareas PVM en Condor.

PVMD

Esta macro mantiene la ruta completa al condor_pvmd especial, el demonio PVM de Condor. El demonio se instala en el “directorio release” de Condor por defecto, asi esta macro está normalmente definida en términos de $(SBIN).

PVMD = $(SBIN)/condor_pvmd

Esta macro mantiene la ruta completa al condor_pvmgs especial, el demonio servidor del grupo PVM de Condor, que se necesita para soportar grupos PVM. El demonio se instala en el "directorio release" por defecto, por tanto, esta macro se define también en términos de $(SBIN).

PVMGS = $(SBIN)/condor_pvmgs

B.1.2.7 Configuración condor_submit

Si queremos que condor_submit añada automáticamente una expresión a la expresión "Requirements" o a la expresión "Rank" de las tareas en nuestra zona, debemos de usar los siguientes macros:

#APPEND_REQUIREMENTS = (expression to append job requirements)

#APPEND_RANK = (expression to append job rank)

Las macros que se muestran a continuación son las que se muestran en el fichero de configuración. Dependiendo del universo en el que se este trabajando, así se rellenará con la expresión que se desee.

#APPEND_REQ_VANILLA = (expression to append to vanilla job requirements)

#APPEND_REQ_STANDARD= (expression to append to standard job requirements)

#APPEND_RANK_STANDARD = (expression to append to vanilla job rank)

#APPEND_RANK_VANILLA = (expression to append to standard job rank)

En otras versiones de Condor, las macros APPEND_RANK_STANDARD y APPEND_RANK_VANILLA pueden aparecer con el nombre de APPEND_PREF_STANDARD y APPEND_PREF_VANILLA. Además, también se dispone de una expresión por defecto en caso de que los usuarios no quieran especificar ninguna:

#DEFAULT_RANK = (default rank expression for all jobs)

#DEFAULT_RANK_VANILLA = (default rank expression for vanilla jobs)

#DEFAULT_RANK_STANDARD = (default rank expression for standard jobs)

DEFAULT_IO_BUFFER_SIZE

Condor mantiene un buffer de datos usados recientemente para cada fichero en aplicaciones abiertas. Este macro especifica el máximo número de bytes por defecto que pueden ser almacenados para cada fichero abierto en la máquina de ejecución.

El comando "condor_status buffer_size" sobrescribe el valor por defecto. Si el macro esta indefinido, por defecto el tamaño es de 512 KB para ser usados.

DEFAULT_IO_BUFFER_SIZE = 524288

DEFAULT_IO_BUFFER_BLOCK_SIZE.

Cuando se permite el almacenamiento, Condor intenta consolidar pequeñas operaciones de lectura y escritura en grandes bloques. Este macro especifica el tamaño del bloque por defecto que Condor usa. El comando "condor_status buffer_block_size" sobrescribe el valor por defecto. Si la macro esta indefinida, el tamaño por defecto es de 32 KB.

DEFAULT_IO_BUFFER_BLOCK_SIZE = 32768

B.1.2.8 Configuración condor_preen

PREEN_ADMIN

Esta macro establece la dirección e-mail donde condor_preen enviará un e-mail (si es configurado para ello).

PREEN_ADMIN = $(CONDOR_ADMIN)

VALID_SPOOL_FILES

Esta macro contiene una lista de los ficheros que condor_preen considera ficheros validos para encontrar en el directorio $(POOL). Por defecto, todos los ficheros que hay son ficheros validos. Un cambio en el macro $(HISTORY) requiere un cambio en esta macro también.

VALID_SPOOL_FILES= job_queue.log, job_queue.log.tmp, history, \

Accountant.log, Accountantnew.log

INVALID_LOG_FILES

Esta macro indica que ficheros son los que eliminará condor_preen del directorio log. Por defecto indica que el fichero “core”.

INVALID_LOG_FILES = core

B.1.2.9 Configuración condor_collector

CLASSAD_LIFETIME

Esta macro determina cada cuanto un ClassAd puede permanecer en el collector antes de que se descarte la información. Esta macro se define en términos de segundos siendo por defecto 900 (15 minutos).

CLASSAD_LIFETIME = 900

MASTER_CHECK_INTERVAL

Esta macro define cada cuanto tiempo el collector debería chequear las máquinas que tienen un ClassAd de algún demonio, exceptuando el del condor_master y enviar un e-mail. Se define en segundos siendo 10800 (3 horas).

MASTER_CHECK_INTERVAL = 10800

CLIENT-TIMEOUT

El tiempo en el que la red de trabajo da a los demonios para que envíen sus actualizaciones. Se define en segundos y es 30.

CLIENT_TIMEOUT = 30

QUERY_TIMEOUT

Se especifica como el tiempo de vida que tiene una pregunta que hace alguien en una red de trabajo. Por defecto es especifica en segundos.

QUERY_TIMEOUT = 60

CONDOR_DEVELOPERS

Esta macro determina la dirección e-mail de los administradores de Condor. Este enviará un e-mail una vez a la semana a esta dirección con la información que nos

muestra el condor_status. Si no se desea que esta información sea enviada, este parámetro se deberá de establecer a NONE.

CONDOR_DEVELOPERS = condor-admin@cs.wisc.edu

CONDOR_DEVELOPERS_COLLECTOR

Por defecto, cada pool envia periodicamente actualizaciones al condor_collector central de UW-Madison con información básica sobre el estado del pool. Esta información incluye el numero total de máquinas, el numero de tareas sometidas, el número de máquinas ejecutando tareas, el nombre del central manager... Estas actualizaciones ayudan a visualizar como esta siendo usado Condor en el mundo. Por defecto se enviará a condor.cs.wisc.edu. Si se quiere que dichas actualizaciones se envíen, se debe descomentar este parámetro.

CONDOR_DEVELOPERS_COLLECTOR = NONE

KEEP_POOL_HISTORY.

Esta macro booleana se usa para decidir si el collector escribirá información estadística sobre el pool para el fichero histórico. Por defecto esta a FALSE. La localización, el tamaño, etc, se controla con otras macros.

KEEP_POOL_HISTORY = False

POOL_HISTORY_DIR

Esta macro establece el nombre del directorio donde los ficheros históricos residen. Por defecto es en el directorio spool/.

POOL_HISTORY_DIR = $(SPOOL)

POOL_HISTORY_MAX_STORAGE

Esta macro establece el tamaño máximo de los ficheros históricos. Cuando el tamaño del fichero histórico se establece a este limite, la información más antigua se descarta. Por defecto son 10Mbytes.

POOL_HISTORY_MAX_STORAGE = 10000000

POOL_HISTORY_SAMPLING_INTERVAL

Esta macro establece el intervalo en segundos entre registros históricos. Por defecto el valor es de 60 segundos.

POOL_HISTORY_SAMPLING_INTERVAL = 60

B.1.2.10 Configuración condor_negotiator

NEGOTIATOR_INTERVAL

Establece cada cuanto tiempo el negotiator realiza un ciclo de negociación. Se define en segundos, y por defecto es 300 (cada 5 minutos).

NEGOTIATOR_INTERVAL = 300

NEGOTIATOR_TIMEOUT

Establece el tiempo en el que el negotiator usa las conexiones de la red de trabajo para el schedd y el startd. Se define en segundos y por defecto es 30.

NEGOTIATOR_TIMEOUT = 30

PRIORITY_HALFLIFE

Esta macro define el tiempo medio de vida de las prioridades de los usuarios. Se define en segundos y equivale a un día.

PRIORITY_HALFLIFE = 86400

NICE_USER_PRIO_FACTOR

Esta macro establece los factores de prioridad de los usuarios.

NICE_USER_PRIO_FACTOR = 10000000

ACCOUNTANT_LOCAL_DOMAIN

Esta macro se usa para decidir si un usuario es local o remoto. Un usuario se considera que esta en el dominio local si el UID_DOMAIN es el valor de esta macro. Normalmente esta macro se establece en el UID_DOMIAN local. Si no se define, se considera a todos los usuarios locales.

ACCOUNTANT_LOCAL_DOMAIN = $(UID_DOMAIN)

REMOTE_PRIO_FACTOR

Esta macro define el factor de prioridad de los usuarios remotos.

REMOTE_PRIO_FACTOR = 10000

NEGOTIATOR_TRAFFIC_LIMIT

Esta macro especifica la capacidad máxima de trafico en la red de trabajo en KB. El negotiator usa los parámetros “ImageSize” y “ExecutableSize” para localizar la red de trabajo usada. El negotiator intentara usar el limite de ancho de banda. Este parámetro por defecto es O, el cual se modificará dependiendo de la red de trabajo usada con el negotiator.

NEGOTIATOR_TRAFFIC_LIMIT = 300000

NEGOTIATOR_TRAFFIC_INTERVAL

Este macro especifica el intervalo (en segundos) para ser usado en el mantenimiento del NEGOTIATOR_TRAFFIC LIMIT. Esta macro, por defecto esta evaluada en 0, ya que depende de la red de trabajo utilizada por el negotiator.

NEGOTIATOR_TRAFFIC_INTERVAL = $(NEGOTIATOR_INTERVAL)

NEGOTIATOR_SOCKET_CACHE_SIZE

Esta macro define el numero máximo de sockets que el negotiator debería mantener como sockets abiertos en la cache. Abriendo sockets en la cache se hace que el protocolo de negociación sea mucho mas eficiente eliminando la necesidad de establecimientos de conexión para cada ciclo de negociación. El valor por defecto es de 16. Para que sea más efectivo, esta macro debería establecerse a un valor mayor que el numero de tareas sometidas.

NEGOTIATOR_SOCKET_CACHE_SIZE = 16

B.1.2.11 Configuración de Uwisc-CSDepartment

A pesar del creciente incremento de usuarios Condor y del tamaño de sus tareas, donde algunas tareas han llegado a tener un tamaño e más de 100 Mbytes, se han tenido que mejorar la política e intentar mejorar el rango de Imagesize.

La variable Imagesize se divide en tres categorías posible, las cuales se definen a continuación por medio de macros. Se define en términos de Kilobytes.

```
BigJob          = (ImageSize >= (50 * 1024))
MediumJob       = (ImageSize >= (15 * 1024) && ImageSize < (50 * 1024))
SmallJob        = (ImageSize <  (15 * 1024))
```

La política se puede definir por: si la tarea es pequeña (SmallJob), se realiza una progresión normal de “suspend” a “vacate” y de ahí a “kill”. Si la tarea es de tamaño mediano (MediumJob) entonces, cuando el usuario vuelve a la máquina la tarea se inicia dejando vacante la máquina en la que se encuentra. La idea es que con un inmediato checkpoint, la tarea encuentre todas las páginas en memoria y el checkpoint sea por tanto más rápido. Las páginas de memoria quedarán libres tan pronto como el checkpoint termine.

Lo lógico para una política especial es ajustarlo con las expresiones WANT_. Todas las otras expresiones y macros usan el valor por defecto.

```
UWCS_WANT_SUSPEND      = ( $(SmallJob) || $(JustCpu) || $(IsPVM) \
                       || $(IsVanilla) )
UWCS_WANT_VACATE       = $(ActivationTimer) > 10 * $(MINUTE) \
                       || $(IsPVM) || $(IsVanilla)
UWCS_START             = ( ($(CPU_Idle) && KeyboardIdle > $(StartIdleTime))
                 && (TARGET.ImageSize <= ((Memory - 15)*1024)) \
                       && ( (MemoryRequirements < (Memory - 15)) || \
                       (MemoryRequirements =?= UNDEFINED && \
                       (RemoteUserCpu > 0.0 || Memory > 127)) ) )
UWCS_START             = $(CPU_Idle) && KeyboardIdle > $(StartIdleTime)
UWCS_SUSPEND           = $(MachineBusy)
UWCS_CONTINUE          = $(CPU_Idle) && KeyboardIdle
$(ContinueIdleTime)
UWCS_PREEMPT           = ( ($(ActivityTimer) > $(MaxSuspendTime)) && \
                       (Activity == "Suspended") ) || \
                       ( SUSPEND && (WANT_SUSPEND == False) )
UWCS_KILL              = $(ActivityTimer) > $(MaxVacateTime)

SUSPEND_VANILLA    = $(MachineBusy)
CONTINUE_VANILLA   = $(CPU_Idle) && KeyboardIdle > $(ContinueIdleTime)
PREEMPT_VANILLA    = ( ($(ActivityTimer) > $(MaxSuspendTime)) && \
                       (Activity == "Suspended") ) || \
                      ( SUSPEND_VANILLA && (Activity != "Suspended") )
KILL_VANILLA           = $(ActivityTimer) > $(MaxVacateTime)

UWCS_PERIODIC_CHECKPOINT    = $(LastCkpt) > (3 * $(HOUR))
#UWCS_PERIODIC_CHECKPOINT   = ( (ImageSize < 60000) && \
```

```
#                    ($(LastCkpt) > (6 * $(HOUR))) ) || \
#                   ( $(LastCkpt) > (12 * $(HOUR)) )
```

UWCS_PREEMPTION_REQUIREMENTS = $(StateTimer) > (1 * $(HOUR)) && RemoteUserPrio > SubmittorPrio * 1.2

UWCS_PREEMPTION_RANK = (RemoteUserPrio * 1000000) – ImageSize

B.2 Fichero de descripción submit

Cada fichero de descripción de tareas de Condor describe un cluster de tareas que puede ser situado en el Pool de ejecución de Condor. Todas las tareas en un cluster deben distribuirse en el mismo ejecutable, pero pueden tener diferentes ficheros de entrada y salida, y diferentes argumentos en el programa, etc. El fichero de descripción submit se usa como el único argumento en la línea de comandos para condor_submit.

El fichero de descripción submit debe contener un comando “executable” y al menos un comando “queue”. Todos los demás comandos tienen acciones por defecto. Los comandos que pueden aparecer en un fichero de descripción submit son:

executale= <nombre>

El nombre del fichero ejecutable para el cluster de tarea. Solo se puede introducir un comando “executable” en un fichero de descripción. Si se está sometiendo una tarea trabajando con el Universo Standard, entonces el nombre del ejecutable debe ser recompilado con las librerías de Condor (mediante el comando condor_compile).

input=<nombre de la ruta>

Condor asume que las tareas se pueden ejecutar durante un largo periodo de tiempo y que el usuario no esperará a que finalicen estas. Por esta razón, los ficheros estandard a los cuales se accede normalmente (stdin, stdout, stderr...) deben referirse a los ficheros. De este modo, el fichero especificado con “input” debe contener una entrada de teclado al programa al que se requiere. Si no se especifica, se usa por defecto /dev/null.

output=<nombre de la ruta>

El nombre del fichero output capturará la información del programa que normalmente se escribe en la pantalla. Si no se especifica, se usa por defecto /dev/null. No se debe usar el mismo fichero de salida para más de una tarea, ya que se pueden sobrescribir.

error=<nombre de la ruta>

El nombre de fichero de error captura los mensajes de error del programa que normalmente deben aparecer por pantalla. Si no se especifica, por defecto se usa la ruta /dev/null. No se debe usar el mismo fichero de error para varias tareas ya que puede causar que una tarea sobrescriba los errores de otra.

arguments=<lista de argumentos>

Lista de argumentos para ser suministrados al programa en la línea de comandos.

initialdir=<ruta del directorio>

Usado para especificar el directorio de trabajo general para las tareas de Condor. Debería ser una ruta de un directorio preexistente.

Si no se especifica, condor_submit, automáticamente insertará el directorio de trabajo del usuario en el tiempo en el que condor_submit esta ejecutándose con el valor de "initialdir"

requirements=<Expresion booleana de ClassAd>

El comando que se requiere es una expresión booleana de ClassAd la cual usa programación C como operaciones. Dada una tarea en un cluster para ejecutar en una máquina dada, esta expresión requerida debe evaluar para verificar la máquina dada. Por ejemplo, para que cualquier máquina ejecute un programa tiene que tener por lo menos 64 Megas de RAM. Si esto fuera así, este comando sería:

requirements = Memory >= 64

Solo se puede representar un comando "requirements" en un fichero de descripción submit. Por defecto, condor_submit añade las siguientes cláusulas a la expresión "requierements":

Tanto la arquitectura (Arch), como el sistema operativo (OpSys) deben coincidir con la arquitectura y sistema operativo de la máquina submit. En otras palabras, aunque se requiera lo contrario, Condor le dará la tarea a la máquina con la misma arquitectura y la misma versión de sistema operativo que la máquina que esta ejecutando condor_submit.

El tamaño de disco debe de ser por supuesto mucho mayor que el tamaño del ejecutable. Se debe de cumplir para asegurar que haya suficiente espacio de disco en la máquina para Condor para copiar sobre el ejecutable.

La Memoria Virtual debe ser mayor o igual que el ImageSize. Se debe de cumplir para asegurar que la máquina tiene suficiente memoria virtual para ejecutar la tarea.

Si el Universo se establece como Universo Vanilla, el dominio del sistema de ficheros se establece igual que el dominio de sistema de ficheros de la máquina submit.

Se pueden visualizar los requisitos de una tarea que ya ha sido sometida, mediante el comando "condor_q –l".

rank = <Expresion>

Específicamente, este comando representa preferencia. Un valor numérico grande es igual a un buen "rank". Condor dará la tarea a la máquina con el mayor "rank" Si se utiliza el ejemplo anterior y se añade el siguiente comando:

rank = Memory

Esto hace que se pregunte a Condor para encontrar todas las máquinas disponibles con más de 60 Megabytes de memoria y darla tarea a la máquina con la mayor memoria.

priority = <prioridad>

El rango de prioridad de las tareas de Condor va desde –20 hasta 20, siendo 0 el valor por defecto. Las tareas de Condor con una gran prioridad numérica se ejecutarán antes que las que tengan una baja prioridad numérica. Estableciendo la prioridad se determinará el orden en cuanto a las tareas que se ejecutan, pero no tendrá efecto en si la tarea se ejecuta adelantada o atrasada en cuanto a la tarea de otro usuario.

notification = <cuando>

Los propietarios de las tareas de Condor son notificados mediante e-mail cuando ocurren ciertos eventos. Encontramos varios atributos para rellenar “cuando”:

“cuando”= always, el propietario será notificado cuando quiera que la tarea sea chequeada, y cuando termine.

“cuando” = complete (por defecto), el propietario será notificado cuando la tarea finalice.

“cuando” = error, el propietario solo será notificado si la tarea finaliza de manera anormal.

“cuando” = never, el propietario no recibirá notificación alguna, indiferentemente de lo que le ocurra a la tarea.

notify_user = <dirección e-mail>

Usado para especificar la dirección e-mail para usar cuando Condor envía e-mail sobre las tareas. Si no se especifica, Condor usara por defecto el propietario de la tarea de la siguiente manera:

Job-owner@UID_DOMAIN

Donde el UID_DOMAIN es especificado por el administrador de Condor. Si UID_DOMAIN no ha sido especificado, condor enviará el e-mail a:

Job-owner@submit-machine-name

getenv = <True | False>

Si “getenv” se establece como True, condor_submit copiará todas las variables de entorno shell en curso. La tarea se ejecutará con las mismas variables de entorno que el usuario tenia establecidas en el tiempo de someter. Por defecto está establecido como False.

hold=<True | False>

Si “hold” se establece como True, entonces la tarea será sometida en el estado hold. Las tareas en el estado hold no se ejecutarán si no es mediante el comando “condor_release”.

environment=<lista de parámetros>

Lista de variables de entorno de la siguiente forma:

<parameter> = <value>

Muchas variables de entorno se pueden especificar de forma separada con un semicolon (“;”). Estas variables de entorno se situarán en el entorno de la tarea antes de la ejecución. La longitud de todos los caracteres especificados en el entorno se limitan actualmente a 4096 caracteres.

log=<nombre de la ruta>

Se usa “log” para especificar el nombre de fichero donde Condor escribirá el fichero log de lo que está ocurriendo con el cluster de la tarea. Por ejemplo, Condor registrará en este fichero cuando y donde la tarea comience a ejecutarse, cuando la tarea se cheque y /o migre, cuando la tarea finalice, etc... Muchos usuarios encuentran especificado un fichero log para una mayor habilidad. Se recomienda su uso. Si no se especifica una entrada para log, Condor no creará un log para ese cluster.

universe=<vanilla | standard | pvm | scheduler | globus | mpi >

Especifica el universo Condor para usar cuando se ejecute la tarea. El universo Condor indica el entorno de ejecución Condor. El universo “standard” es el establecido por defecto, y en el, las tareas son recompiladas mediante el comando condor_compile con las librerías Condor, además de soportar Checkpointing y “remote

procedure call". El Universo "Vanilla" es un entorno de ejecución para tareas que no han sido recompiladas con las librerías Condor.

Nota: se debe usar el Universo Vanilla para someter scripts shell en Condor. El Universo PVM es para tareas de paralelas escritas con PVM3.4.

El "scheduler"es para tareas que deberían actuar como "metascheduler".

El universo Globus traduce el fichero de descripción submit a un string RSL Globus y lo pasa al programa de ejecución "globusrun".

El universo MPI es para ejecutar tareas mpi hechas con el paquete MPICH.

image_size=<tamaño>

Este comando llama a Condor con el tamaño de imagen virtual máximo en el cual el programa crecerá durante su ejecución. Condor ejecutará la tarea solo en las máquinas que tengan suficientes recursos (como memoria virtual), para soportar la ejecución de la tarea. Si no se específica el tamaño de imagen de la tarea en el fichero de descripción, Condor automáticamente hará una estimación sobre el tamaño y ajustará esta estimación para que el programa se ejecute.

Si el tamaño de la imagen de la tarea tiene una baja estimación, puede caer inhabilitándolo para adquirir más espacio de dirección. Si el tamaño de imagen es sobreestimado, Condor puede tener dificultades encontrando máquinas que tengan los recursos que se necesitan, por ello, tamaño debe estar en Kbytes.

machine_count = <min..max>| <max>

Si "machine_space" se especifica, Condor no comenzará la tarea hasta que pueda simultáneamente suministrar la tarea con "min" máquinas. Condor continuará intentando proveer hasta "max" máquinas, pero no retrasará el comienzo de las tareas el hacer eso.

Importante: Usar "machine_count" si y solo si se somete en el universo PVM o MPI. Usar "min" y "max" para el universo PVM y solo "max" para el universo MPI.

coresize=<size>

El programa del usuario debería abortar y producir un fichero core, coresize que especifica el máximo tamaño en bytes del fichero "core" que el usuario desea mantener. Si "coresize" no se especifica en el fichero de comandos, el usuario del sistema usa el requisito limite "coredumpsize" (excepto en HP-UX).

nice_user=<True | False>

Normalmente, cuando una máquina llega a estar disponible a Condor, este decide la tarea a ejecutar basándose en la tarea del usuario y en sus prioridades. Estableciendo "nice_user" a True, se llama a Condor no para usar las prioridades del usuario regular, pero la tarea debe tener la mayor prioridad entre todos los usuarios y todas las tareas.

Este parámetro es bastante manejable si el usuario tiene algunas tareas que desea ejecutar pero no desea utilizar los recursos que pueden ser usados por otros usuarios para ejecutar sus tareas Condor. Por defecto este parámetro está establecido a False.

kill_sig = <signal-number>

Cuando Condor necesita eliminar una tarea de una máquina, envía a la tarea la señal específica por medio de "signal-number". Esta, necesita ser un integer el cual representa una señal valida en una máquina Execution.

Para tareas sometidas en el entorno Standard, el valor por defecto es el número para SIGTSTP, el cual llama a las librerías de Condor para iniciar un checkpoint del

proceso. Para tareas sometidas en el entorno Vanilla, por defecto es SIGTERM, que es la manera estándar de terminar un programa en Unix.

buffer_size = <bytes-in-buffer>

Condor mantiene un buffer de datos recientemente usados para cada fichero y aplicación abierta. Esta opción especifica el máximo numero de bytes para se almacenados en el buffer para cada fichero abierto en la máquina execution.

El tamaño del buffer y el efecto en el rendimiento total se puede visualizar con la opción "-io" de "condor_status". En la mayoría de las versiones de Condor para Linux, por defecto, el tamaño del buffer es de 512 KB, aunque el macro DEFAULT_IO_BUFFER_SIZE del fichero de configuración ha sido establecido por defecto diferente por el administrador en la máquina Submit.

buffer_block_size = <bytes-in-block>

Cuando el buffer está habilitado, Condor intenta consolidar pequeñas operaciones de lectura y escritura en grandes bloques. Esta opción especifica el tamaño del bloque. Un tamaño muy pequeño de bloque puede normalmente reducir el rendimiento I/O. Este bloque debería definitivamente ser tan grande como las operaciones I/O del rendimiento del programa.

El tamaño del bloque del buffer y el efecto en el rendimiento puede ser visible mediante la opción "-io" de "condor_status". En esta versión de Condor, por defecto, el tamaño del bloque del buffer es de 32KB, aunque en el fichero de configuración, la macro DEFAULT_IO_BUFFER_BLOCK_SIZE ha sido establecida por defecto con un valor diferente por el administrador en la máquina Submit.

file_remaps = < "nombre = nuevonombre ; nombre2 = nuevonombre2 ...">

Direcciona Condor para usar un nombre de fichero nuevo en lugar de uno antiguo. "nombre" describe el nombre de fichero que la tarea puede intentar abrir y "nuevonombre" describe el nombre del fichero que debe ser reemplazado, "nuevonombre" puede incluir un acceso especifico opcional, "local:" o "remote:". Si se deja sin especificar, el acceso especifico por defecto es "remote:"

file_remaps = "dataset.1 = otther.dataset"

file_remaps = "very.big = local :/bigdisk/bigfile"

file_remaps = "very.big = local: bigdisck/bigfile ; dataset.1 = other.dataset"

rendezvousdir = <ruta del directorio>

Especifica el directorio del sistema de ficheros distribuido para ser usado por la autentificación del sistema de ficheros cuando se somete un "scheduler" remoto. Debe ser una ruta de un directorio existente.

Existen muchos otros macros que no se definen aquí ya que se usan para especificados universos que no son el objetivo de este proyecto, tales como globusscheduler, globusarguments, globusexecutalbe...

Apéndice C

Condor. WindowsNT

C.1 Condor en WindowsNT

La Aplicación Condor también puede implementarse bajo el Sistema Operativo Windows NT. Esto es interesante ya que hoy en día no todo el mundo utiliza Uníx/Linux como sistema operativo pero, como se verá en los siguientes apartados, el instalar Condor bajo este sistema operativo trae consigo muchas restricciones a la hora de trabajar con la aplicación [53].

Básicamente, Condor bajo Windows NT tendrá la misma funcionalidad que Condor para Uníx, aunque más adelante se podrá ver que la funcionalidad está algo más reducida. Algunas opciones implementadas bajo NT son:

- La capacidad para someter, ejecutar y mantener colas de tareas en cluster de máquinas NT.
- Se dispone de todo lo necesario para ejecutar un Central Manager en Windows NT.
- Condor NT incluye un programa de instalación donde se puede realizar una instalación completa (fullinstall) o desinstalación de Condor. La información especificada por el usuario en la ejecución del programa se almacena en el registro de sistema. El programa de ejecución puede actualizar la instalación que tiene en ese momento con un nuevo “release” con un mínimo coste y esfuerzo.

Hay también otras figuras que no están implementadas en Condor bajo NT. Algunas de ellas son:

- Los procesos de checkpoint y migración no son soportados.
- Tampoco se soporta el acceso a ficheros en la red de trabajo, es decir que cada estación de trabajo que somete tareas debe tener todos los ficheros necesarios en el disco duro local.

Para terminar esta introducción se puede hacer mención de Condor en Windows 2000 profesional, donde hoy en día ya es soportado. La excepción reside en Windows 2000 Server. Actualmente se está realizando Condor para ser soportado en Windows XP.

C.1.1 Programa de ejecución de Condor. Instalación

La instalación de Condor la debe realizar un usuario con privilegios. Después de la instalación, los servicios de Condor serán ejecutados bajo el sistema local. Cuando Condor ejecuta una tarea del usuario, cualquiera que sea esta, se ejecutará con permisos de usuario. Condor creará dinámicamente un informe, y después lo borrará cuando la tarea haya finalizado o se haya eliminado esta de la máquina.

Por tanto, se seguirán los mismos pasos que como en instalaciones anteriores, obteniendo Condor de la página oficial http://www.cs.wisc.edu/condor/downloads/. Se debe de descargar la versión de Condor que más convenga, por ejemplo “condor-

6.2.0- winnt40-x86.exe". Se ejecuta el programa en el terminal de MS-DOS de Windows y se siguen los pasos de instalación.

C.1.1.1 Instalación del Central Manager

Al igual que la instalación del Central Manager para Unix, se debe de tener en cuenta la estabilidad de la estación de trabajo, ya que si la máquina falla, Condor deja de trabajar. Los pasos a seguir serán los siguientes:

En la caja de dialogo de la configuración de Condor aparecen dos opciones a elegir dependiendo de la instalación que se desee:

- Trabajar en un Condor Pool que ya existe
- Crear un nuevo Condor Pool donde la máquina en la que se está trabajando va a ser el Central Manager.

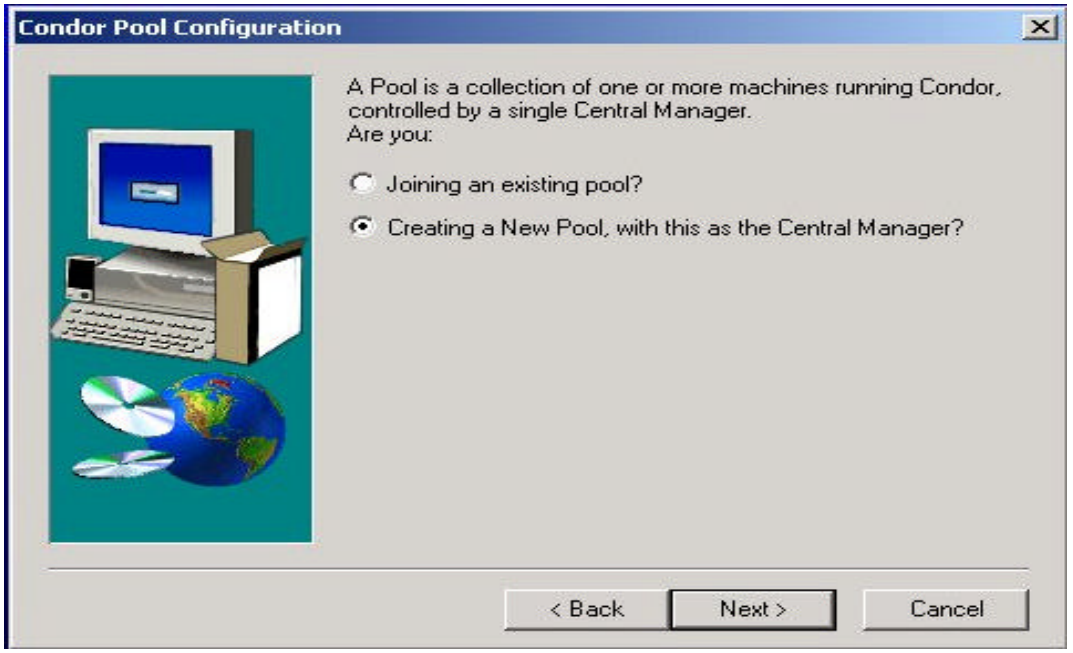

Figura C. 1 Configuración del Condor Pool en WindowsNT

En la siguiente caja de dialogo de configuración de "others machines" aparecen dos opciones donde se deberá elegir la que más convenga dependiendo de la instalación que se esté realizando:

- Si se desea que otras máquinas se comuniquen con el Central Manager que se está configurando.
- Si se desea configurar la máquina como un único nodo de trabajo.

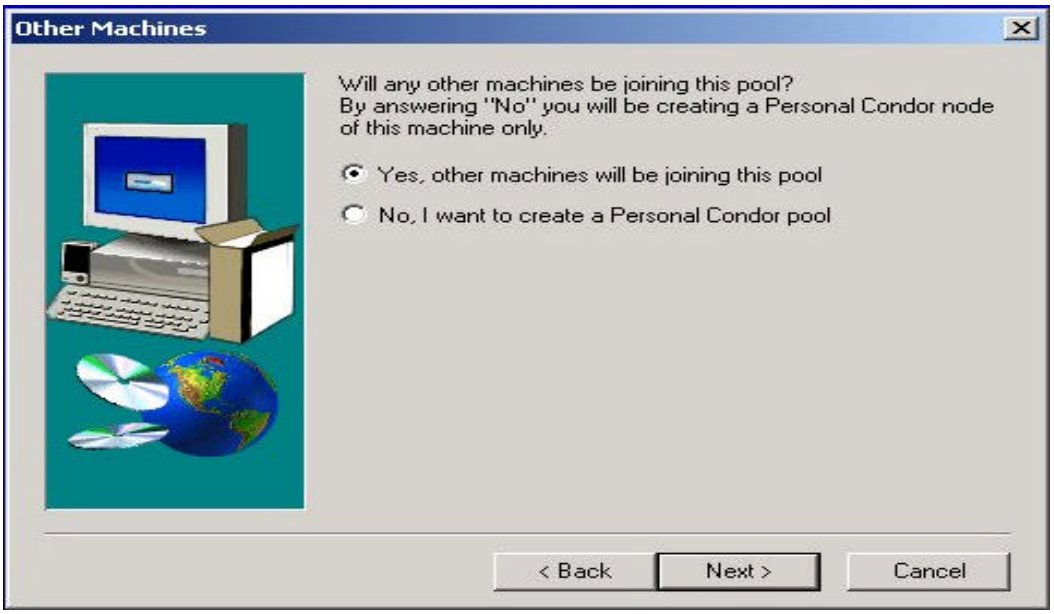

Figura C. 2 Otras máquinas en el Condor Pool en WindowsNT

En la siguiente caja de dialogo, seleccionar “Select This Machines Roles”,que son las características por defecto que se deben aceptar, permitiendo a las estaciones de trabajo tanto someter como ejecutar tareas en el Condor pool. Se requiere que el usuario seleccione el directorio de los ficheros Condor, información sobre e-mail, SMTP, nombre de dominio de la red de trabajo DNS y establecer los permisos de lectura y escritura en la estación de trabajo.

En la caja de dialogo de permisos de Lectura/Escritura no se debe borrar el “*” que está situado delante del dominio de nombres de las primeras dos cajas de texto.

La caja de dialogo de “Job Start Policy” establece las condiciones que gobiernan cuando una tarea Condor puede comenzar su ejecución. Lo normal es establecerlo después de 15 minutos de inactividad de la consola y baja actividad de la CPU como se muestra en la siguiente figura.

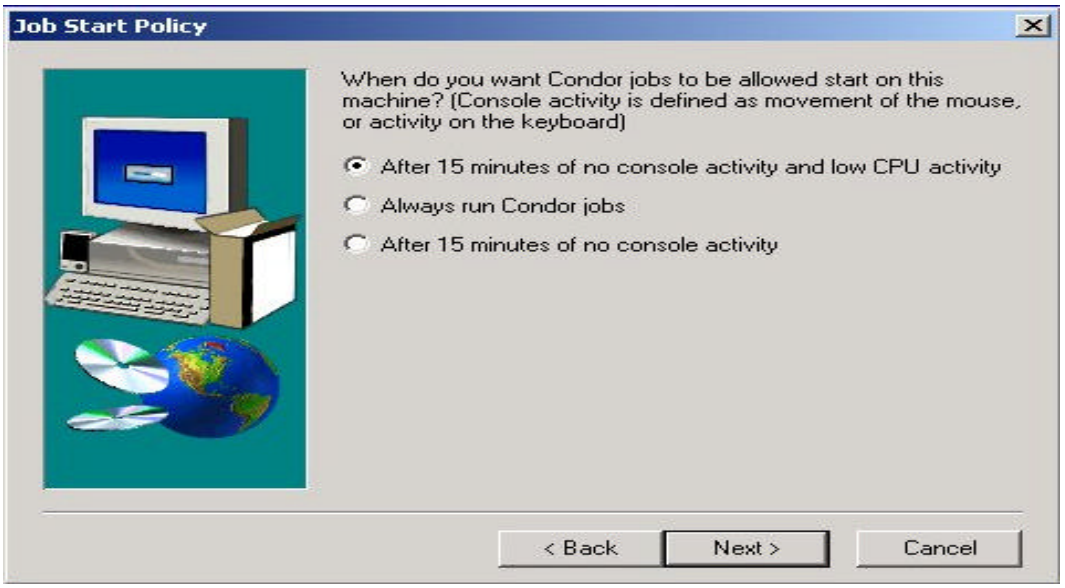

Figura C. 3 Políticas de ejecución de tareas en WindowsNT

Para verificar el funcionamiento también se puede seleccionar la opción de "always run Condor jobs" para simplificar el chequeado de las tareas que están siendo ejecutadas correctamente. Una vez que se completa la comprobación, la política de comienzo de la tarea puede ser alterada por la re-ejecución del programa de instalación.

La caja de dialogo "Job Vacate Policy" permite al usuario especificar que ocurre al ejecutar tareas en la estación de trabajo cuando el estado "idle" no es de larga duración. Seleccionando la opción "Restart the job on another machine" se reiniciará la tarea en otra máquina en el pool. La segunda opción efectivamente suspende la tarea permitiendo continuar cuando la estación de trabajo comience otra vez en "idle". Para tareas relativamente largas, en una red de trabajo ocupada puede ser mejor elegir "Leave it memory and restart it when you leave" donde las tareas pueden ser repetidamente re-localizadas, re-iniciadas y normalmente no se completan nunca.

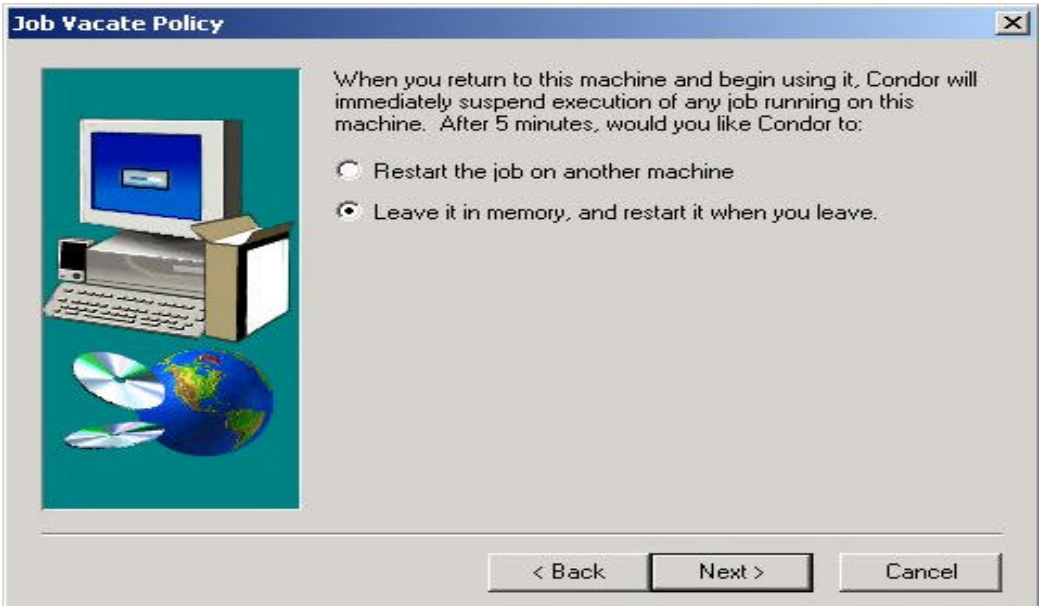

Figura C. 4 Política de tareas en WindowsNT

C.1.1.2 Instalación de estaciones de trabajo en un Condor Pool

De la misma manera que para el Central Manager, se descarga la versión de Condor que más convenga para Windows NT y se ejecuta el programa de instalación. Aparecen las mismas cajas de diálogo que en el caso anterior, pero ahora la instalación es de una estación de trabajo en un pool ya existente, por lo tanto se elige la primera opción "Joining an existing pool?". Por lo demás, los pasos son idénticos a los que se han seguido para el Central Manager.

C.1.2 Una vez instalado

Después de que la instalación de Condor se haya completado, el servicio de Condor debe ser iniciado. Si se usa el programa de ejecución para instalar Condor, el servicio de Condor debería estar ya iniciado. Si Condor se ha instalado manualmente, debe ser iniciado también a mano, o simplemente rebotando la máquina. Para iniciar Condor a mano los pasos a seguir son:

- Seleccionar menú de inicio.
- Seleccionar configuración y elegir panel de control.
- Elegir Servicios.
- Pulsar el botón de “start” que se encuentra a la derecha de la caja de dialogo.

La instalación ya esta terminada y el sistema ya está preparado para ejecutar tareas. De manera alternativa se puede realizar introduciendo en la línea de comandos de MS-DOS el comando:

>net start condor

Como en el caso de Uníx se pueden visualizar que servicios de Condor se están ejecutando mediante el “Task Manager” (CTRL-shift-escape), que serán:

- condor_master.exe
- condor_negotiator.exe
- condor_collector.exe
- condor_startd.exe
- condor_schedd.exe

El condor_collector.exe y el condor_negotiator.exe se estarán ejecutando en la máquina que actúa como Central Manager. El condor_schedd.exe se ejecutará en la máquina donde se someten las tareas del Condor pool y el condor_startd.exe se ejecutará en la máquina que inicia las tareas.

También se pueden utilizar otros comandos de Condor usados en Uníx para visualizar el estado del Condor Pool. Algunos de estos comandos son condor_status y condor_q.

C.1.3 Ejecución de tareas en Condor

Las tareas son sometidas a Condor y encoladas por la máquina Central Manager hasta que los recursos están disponibles para iniciar las tareas. Esto se hace vía MS-DOS, usando el comando “condor_submit” y un fichero de descripción submit, al igual que para Unix.

> condor_submit job_description.sub

El fichero de descripción submit contiene toda la información que Condor necesita para procesar la tareas, por ejemplo, el nombre del ejecutable, los argumentos de la línea de comandos, el directorio de trabajo, etc. La manera mas fácil de entender estos ficheros de descripción submit es trabajar a través de los ficheros dados en el directorio Condor\examples. Por ejemplo, el directorio Condor\examples\printname\name.sub contiene los elementos básicos de un fichero de descripción submit:

```
universe = vanilla
environment = path=c:\winnt\system32
executable = printname.bat
output = printname.out
error = printname.err
log = printname.log
queue
```

- La variable “universe” se debe de establecer siempre a “vanilla” ya que es el único entorno de trabajo que permite Condor en Windows NT. Esto significa

que los ejecutables no se enlazan con las librerías de Condor, es por ello que tampoco se soporta el Checkpointing ni la migración de procesos.

- La variable “enviroment” que indica el path, establece el sistema de ficheros local.

- La variable “executable” especifica el nombre del fichero del programa. En este caso el fichero contiene un nombre de red, de manera que al ejecutarse pinta el nombre y el usuario actual de la estación de trabajo en la pantalla.

- Las variables output, error y log especifican los nombres de ficheros en los cuales la salida estándar, error estándar y la información de registro de Condor de la tarea se escribirán.

- “queue” significa que la tarea será encolada por el servicio Condor.

C.1.4 Problemas

Pero no todo es tan sencillo con Windows NT. Algunos de los problemas que se tienen con este sistema operativo y Condor son:

- Condor no puede trabajar bajo Windows 2000 server. El Central Manager debe ser ejecutado en Windows NT 4.0, Windows 2000 Profesional, o Uníx.

- Es importante recordar que la versión NT de Condor solo puede trabajar con el universo “vanilla”. Esto significa que si una tarea está parcialmente terminada no puede migrar a otra estación de trabajo si se interrumpe su ejecución . Debe ser suspendida hasta que la estación de trabajo en la que se encuentre vuelva al estado “idle” otra vez para que el proceso se reinicie en otra estación de trabajo.

- Pueden coexistir sin problemas estaciones de trabajo con Uníx y estaciones de trabajo con NT en un mismo Condor Pool. La única restricción existente es que las tareas que son sometidas en estaciones de trabajo NT deben de ser ejecutadas en estaciones de trabajo NT.

- Hay unas cajas de diálogo donde se establece la política de “start” y “vacate” de las tareas en Condor. Si la red de trabajo se usa de manera excesiva, las tareas que estén encoladas, raramente podrán iniciarse, además de que podrían ser eliminadas a mitad de proceso. Para el propósito de la red se debe asegurar que la política “start” de las tareas este siempre puesta para ejecutar tareas Condor y “vacate” para suspender tareas, dejándolas en memoria.

Apéndice D

Pliego de condiciones

D.1 Condiciones generales

Las relaciones laborales entre el contratista y el contratado a efectos del desarrollo del presenten proyecto estarán sujetas a la normativa vigente.

Las cargas y salarios deberán ajustarse a los presupuestos en lo que respecta a cantidad y duración. Para cualquier variación se deberá pedir la correspondiente autorización al Ingeniero Proyectista.

Los equipos informáticos deberán ser los mismos que se especifican en la lista de equipos.

El contratista, por el hecho de contratar el proyecto, se compromete a aceptar las cláusulas de este pliego de condiciones y el presupuesto adjunto.

D.2 Condiciones técnicas

A continuación se citan los requisitos hardware y software necesarios para un correcto funcionamiento.

D.2.1 Hardware

Un mínimo de tres ordenadores con las especificaciones técnicas que a continuación se detallan (o similares):

Procesador	Pentium III
Disco duro	10 GB
RAM	128 MB
Tarjeta Red	Ethernet 10/100Mb

Tabla D. 1 Requisitos hardware

- Hub 8 puertos 10/100 Ethernet
- Impresora
- SAI (al menos 1)

D.2.1 Software

- S.O Linux, SUSE 7.2
- Windows 98- 2ª Edición
- Paquette Office
- Paquete software “Condor Application”, versión 6.2.1
- Módulos opcionales Condor: “Checkpoint-Server”, “Condor-view””

Apéndice E

Presupuesto

E.1 Mano de obra

Se contabiliza en este apartado el tiempo empleado para el desarrollo del proyecto. También se incluye el tiempo dedicado a la redacción de la documentación del proyecto. El cálculo total de este apartado, se efectuará del siguiente modo:

- Cálculo de los salarios base según conceptos.
- Evaluación de las diversas cargas sociales.
- Suma de los anteriores conceptos para calcular el salario base efectivo.
- Cálculo total de la mano de obra, debido a los jornales en los conceptos especificados.

E.1.1 Cálculo del salario base

Trabajador	Salario base (euros/dia)
Ingeniero	72.12

Tabla E. 1 Salario base de ingeniero

E.1.2 Evaluación del volumen del trabajo

Trabajador	Dias de trabajo
Ingeniero Técnico	100

Tabla E. 2 Volumen de trabajo

E.2 Coste total de la mano de obra

Trabajador	Dias	Salario base +cargas sociales (euros)	Coste total (euros)
Ingeniero	100	114.74	11474

Tabla E. 3 Coste total de la mano de obra

E.3 Coste del material

Concepto	Coste (euros)
Equipamiento informático (Hardware):	
Ordenador personal (3)	1800
Hub	34.45
Tarjeta de red (3)	122.19
SAI	198.30
Cables / conectores	5.25
Equipamiento informático Software(Windows 98)	138.21
Impresora láser	456.99
Total de equipos informaticos	2755.39

Tabla E. 4 Coste del material

E.4 Material fungible

Concepto	Coste
Papel DIN-A4	9.02
Papel continuo	6.01
Tóner para impresora	77.93
Total de material fungible	92.96

Tabla E. 5 Material fungible

E.6 Presupuesto total

Concepto	Coste
Total de mano de obra	11474
Total de equipos informáticos	2755.39
Total de material fungible	92.96
Suma total	14322.35
IVA (16%)	2291.57
Importe total	16613.92

Tabla E. 6 Presupuesto total

El importe total de este proyecto asciende a la cantidad dieciséis mil seiscientos trece euros con noventa y dos centimos.

Cartagena, . de ……. del ..

Referencias

[1] *HTC. High-ThroughputComputing.* http://thd.pnpi.spb.ru/Cluster/CONDOR/overview/overview.html

[2*] HPC. High-Performance Computing.* http://www.cs.wisc.edu/condor/htc.html

[3] *FLOPS. Floating Points Operations Per Second.* http://www-fp.mcs.anl.gov/~gregor/datorr/report/datorr-report.doc

[4] *FLOPY. Floating Points Operations per Year.* http://oscinfo.osc.edu/notices/r934385034.html

[5] *Red Entropía.* http://www.entropia.com

[6] *Sistema SETI.* http://setiathome.ssl.berkeley.edu

[7] *Migración de procesos.* http://170.210.92.2:300/CACIC2001/trabajos/pdf/PC-00183.pdf

[8] *Aplicación Condor. Página oficial.* http://www.cs.wisc.edu/condor

[9] *LSF (Load Sharing Facility).* http://wwwinfo.cern.ch/pdp/lsf/

[10] *Loadlever.* http://www.unex.es/siue/sp/loadleveler/

[11] *Sun-Jini.* http://www.jini.org/

[12] *Cluster en Linux.* http://www.hispacluster.org

[13] *Roles de la Aplicación Condor.* http://www.cs.wisc.edu/condor/manual/v6.2/3_1Introduction.html

[14] *Elementos que intervienen en la Aplicación Condor.*

http://www.cs.wisc.edu/condor/manual/v6.2/1_3Exceptional_Features.html

[15] *Remote Sytem Calls.*

http://archiv.tu-chemnitz.de/pub/2002/0076/data/condor_presentation.pdf

[16] *Entornos de ejecución de la Aplicación Condor.* http://www.cs.wisc.edu/condor/manual/v6.2/2_4Road_map_running.html

[17]*Entorno de ejecución Globus Universe. Página oficial.* http://www.globus.org

[18] *Entorno de ejecución Universe PVM. Parallel Virtual Machine.* http://www.cs.wisc.edu/condor/manual/v6.2/2_8Parallel_Applications.html

[19] *Entorno de ejecución Universe PVM. Parallel Virtual Machine.* http://www.csm.ornl.gov/pvm/pvm_home.html

[20] *Procesos que intervienen en la Aplicación Condor.* http://www.cs.wisc.edu/condor/manual/v6.2/3_1Introduction.html#SECTION00412000000000000000

[21] *Estados y actividades de las máquinas en la Aplicación Condor. Benchmarks.* http://www.cs.wisc.edu/condor/manual/v6.2/3_6Configuring_Startd.html#SECTION00465000000000000000

[22] *Las tareas bajo condor_submit.* http://pandora.aei-potsdam.mpg.de/merlin/user_information/running_codes_with_condor.html

[23] *Universo Standard en la Aplicación Condor.* http://nuweb.jinr.ru/Texts/Condor/condor-V6_0-Manual/2_5Job_Preparation.html#SECTION003521000000000000000

[24] *Universo Standard en la Äplicación Condor.* http://www.twgrid.org/pdf/5-14%20tutorial%20III-IV-todd.pdf

[25] *Universo Standard en la Aplicación Condor.* http://www.cs.wisc.edu/condor/manual/v6.2/2_4Road_map_running.html

[26] *RPC. Remote Procedure Calls.* http://www.map.es/csi/caibi/ibst/estandar/8/ibrpc.htm

[27] *RPC. Remote Procedure Calls.* http://grasia.fdi.ucm.es/jpavon/dso/rpc.pdf

[28] *Origen de NFS. Network File Systems.* http://www.unixsup.com/unixlinux/historiaunixcuxs.html

[29] *Mecanismo NFS.* http://lucas.hispalinux.es/Manuales-LuCAS/GARL2/garl2/x-087-2-nfs.html

[30] *NFS, inconsistencias.* http://lucas.hispalinux.es/Tutoriales/NISNFS/nis-nfs98/

[31] *NIS. Network Information System.* http://webdia.cem.itesm.mx/ac/rogomez/PagsAdmonUnix/NIS/node1.html

[32] *NIS sobre RPC.* http://www.walc2000.unam.mx/material/track1/PRAC1.pdf

[33] *Mecanismo Checkpoint.* http://www.bo.infn.it/calcolo/condor/condor562.html

[34]] *Mecanismo Checkpoint.* http://www.cs.wisc.edu/condor/manual/v6.2/4_2Introduction_Condor_s.html

[35] *Multiples Servidores Checkpoint. Dominios de Checkpoint.* http://www.cs.wisc.edu/condor/manual/v6.2/3_4Installing_Contrib.html#SECTION00442000000000000000

[36] *Restricciones de las tareas en la Aplicación Condor.* http://www.cs.wisc.edu/condor/manual/v6.2/1_4Current_Limitations.html

[37] *Comandos de la Aplicación Condor.* http://www.cs.wisc.edu/condor/manual/v6.2/8_Command_Reference.html

[38] *Ejecutando tareas en la Aplicación Condor.* http://styx.esrin.esa.it/grid/docs/docs/WP9-task00-Report.doc

[39] *Ficheros de descripción submit, parámetros.* http://www.cs.wisc.edu/condor/manual/v6.2/2_5Submitting_Job.html#SECTION00355000000000000000

[40] *Ficheros de descripción submit, ejemplos.* http://www.cs.wisc.edu/condor/quick-start.html

[41] *Versiones de la Aplicación Condor.* http://www.cs.wisc.edu/condor/manual/v6.4/8_Condor_Version.html

[42] *Restricciones del Condor Pool.* http://www.bo.infn.it/condor-mirror/manual/v6.2/condor-V6_2-Manual.pdf

[43] *Condor trabajando como usuario Root.* http://www.cs.wisc.edu/condor/manual/v6.2/3_12Security_In.html

[44] *Pasos de instalación.* http://www.bo.infn.it/condor-mirror/manual/v6.4/condor-V6_4-Manual.pdf

[45] *Perl.* http://www.perl.com/

[46] *Manual Linux.* http://g.unsa.edu.ar/cgi-bin/pacshow.pl?es+xmail

[47] *Manual Linux.* http://www.die.net/doc/linux/man/man1/

[48] *Condor View.* http://www.cs.wisc.edu/condor/manual/v6.4/3_4Contrib_Module.html

[49] *Condor View.* http://www.physik.rwth-aachen.de/group/IIIphys/compute/condor/condor-x.x.x-java-view-client-6.1.8/INSTALL

[50] *Librería DaemonCore de Condor.* http://www.cs.wisc.edu/condor/manual/v6.2/3_7DaemonCore.html

[51] *Seguridad de Condor.* http://www.cs.wisc.edu/condor/manual/v6.2/3_8Setting_Up.html

[52] *Macros de Condor.* http://www.cs.wisc.edu/condor/manual/v6.2/ref.html

[53] *Condor WindowsNT.* http://www.cs.wisc.edu/condor/manual/v6.2/5_3Installation_Condor.html

[54] *Condor Flocking.* http://www.cs.wisc.edu/condor/manual/v6.1/3_11Setting_up.html#SECTION004114000000000000000

[55] *Java Universe.* http://www-unix.griphyn.org/workspace/VDS/javadoc/org/griphyn/vdl/classes/Executable.html

[56] *Java Universe.* http://www.bo.infn.it/condor-mirror/manual/v6.4/2_8Java_Applications.html

[57] **Andrew S. Tanenbaum**. ”Sistemas Operativos Distribuidos”, Primera Edición. Editorial Prentice Hall, 1995