Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

Universidad Autónoma de Barcelona
Escuela Técnica Superior de Ingeniería
Departamento de Arquitectura de Computadores y Sistemas Operativos
Barcelona - España 08193

Schedflow: Sistema Integrador De Políticas De Planificación Y Gestores De Workflow En Entornos Grid (Schedflow: An Integrating System Of Scheduling Policies And Workflow Engine In Grid Environments)

Gustavo Martínez, Elisa Heymann, Miguel Angel Senar
gustavo.martinez@caos.uab.es, elisa.heymannp@uab.es, miquelasenar@uab.es

RECIBIDO ENERO 2009 ACEPTADO MARZO 2009

Resumen

Este trabajo describe las principales funcionalidades de SchedFlow, una herramienta que permite a los usuarios la integración de políticas de planificación para workflows, y el enganche con gestores de workflows. Para ello hemos diseñado una arquitectura sencilla y robusta, en la cual nuestro sistema gestiona a través de módulos los elementos necesarios para ejecutar una aplicación workflow correctamente. Para mostrar la potencialidad de nuestro sistema, hemos realizado los experimentos con la aplicación Montage. Toda la experimentación fue realizada en un entorno distribuido para observar el comportamiento de nuestro sistema ante este escenario. Las políticas de planificación utilizadas fueron estáticas y dinámicas (aleatoria estática, y Min-Min dinámica). Finalmente los resultados obtenidos muestran reducciones significativas en referencia al *makespan* de la aplicación. Concluyendo de esta manera que SchedFlow es una herramienta con un potencial importante para ejecutar aplicaciones workflow.

Palabras clave: Gestión de Workflow, Políticas de Planificación, Entornos Distribuidos.

Abstract.

Workflow applications running on distributed environments are a promising solution for resource and computing intensive problems. However, the heterogeneity of resources in these environments complicates resource management and the scheduling of such applications. Sophisticated scheduling policies are being developed for workflows, but they have little impact in practice because their integration into existing workflow engines is complex due to the restrictions imposed by each workflow engine. In this paper, we introduce SchedFlow a system that allows users to easily

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

incorporate scheduling policies in existing workflow engines. Our system provides a simple API with the essential mechanisms needed for scheduling policies to interact with workflow engines. Experiments have been conducted to show the functionality of SchedFlow with both static and dynamic scheduling policies. The workflow engine used in these experiments was Condor-DAGMan, the workflow application was Montage, and three scheduling policies were considered: Random and dynamic Min-Min. As a result we were able to implement once compare different workflow scheduling policies, and see makespan reduction of the scheduling policies with event management about static scheduling policies.

Keywords: Workflow Management, Scheduling Policies, Distributed Environments.

1. Introducción

En la actualidad existe un elevado número de aplicaciones orientadas a cómputo intensivo y grandes transferencias de datos. Ejemplos de estas aplicaciones paralelas son, las de física de altas energías EGEE[1], las genómica tal es el caso de Fura[2], la simulación de sistemas complejos Gridsim[3], ó procesamiento de imágenes espaciales como el Montage[4].

El principal problema de estas aplicaciones es que tienen un elevado tiempo de ejecución. Para mejorar estos tiempos de ejecución en este tipo de aplicaciones se han propuesto diversas soluciones, una de ellas ha sido ejecutar estas aplicaciones en entornos Grid.

Un sistema Grid está definido como un conjunto de ordenadores conectados en una red de alta velocidad, que se comunican y coordinan sus acciones mediante el paso de mensajes para lograr un objetivo común, y en el cual se establece la comunicación mediante un protocolo prefijado por un esquema cliente-servidor[5][6].

El cómputo Grid lo podríamos agrupar en dos vertientes, los dedicados y los no dedicados. Nosotros centraremos nuestra atención básicamente a los sistemas Grid de cómputo no dedicado. Existen también otras arquitecturas como el Internet Computing y el Web Computing[7][8] que están representados la segunda vertiente también pero que de momento no utilizamos en nuestra investigación.

La selección de un entorno distribuido como el Grid y no un clúster, es porque no nos supone un elevado precio de los componentes utilizados que nos permitirán alcanzar mejoras constantes en sus prestaciones, mientras

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

que los usuarios que trabajan con clúster disponen de un tiempo limitado y a unas restricciones impuestas por el grupo de investigación al que pertenece dicho clúster.

Aprovechando las ventajas que ofrece el sistema operativo Unix y la conectividad a Internet, existen actualmente muchos proyectos que desarrollan los elementos necesarios para construir y utilizar infraestructuras distribuidas, ejemplos de ellas son Power Grid [9][10] ó Teragrid [11][12] desarrollados en Estados Unidos, ó a nivel europeo proyectos como EGEE [13] ,CROSSGRID [14] y mas recientemente INT.EU.GRID [15], los cuales están dirigidos a aplicaciones interactivas de altas prestaciones, aplicaciones de procesamiento distribuido de datos ó aplicaciones workflow, siendo estas últimas las que referiremos durante esta investigación.

Actualmente se generan una cantidad de aplicaciones en apoyo a la comunidad científica, en la cual los planificadores de estas aplicaciones juegan un papel importante al momento de ejecutar dichas aplicaciones, motivo por el cual es necesario agotar todos los esfuerzos que sean necesarios para que las aplicaciones estén mejorando constantemente. Y así los usuarios fínales obtengan respuestas más rápidas para realizar los análisis correspondientes de los resultados obtenidos de la ejecución de estas aplicaciones.

Dada la naturaleza de las aplicaciones y del entorno de ejecución distribuido, en ocasiones resulta difícil evaluar el resultado de una política de planificación. Para realizar una evaluación de forma fácil, se utilizan como primera fase simuladores, que permitan ver el comportamiento de estas aplicaciones sobre sistemas distribuidos ó Grid, y con los cuales se pueden modificar diferentes estrategias de planificación de tareas, y proporcionar una idea aproximada sobre cuáles son las mejores condiciones de ejecución. Lo que se traduce en una reducción de los tiempos de ejecución, ahora bien nuestro trabajo se centra fundamentalmente en un entorno de ejecución real, donde las características son totalmente dinámicas y los cambios pueden ocurrir en cuestión de segundos, con estas condiciones se estudiaron los diferentes elementos que nos permitieran dar robustez a nuestra investigación.

Nuestra investigación está inmersa en dos grandes líneas, por un lado las políticas de planificación que existen para el manejo de workflow, y por otro los correspondientes gestores de workflow que se utilizan para ejecutar este tipo de aplicaciones. Esquematizado sin entrar en detalles de las tecnologías existentes para dar solución a estos problemas, nuestro desarrollo y pruebas estarán enmarcados en un entorno Grid[16][17], donde los recursos pueden considerarse como clúster locales y las aplicaciones se ejecutaran en estos clúster de forma coordinada.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

Para trabajar con los clúster locales se necesitan unos servicios (middleware)[18] situados entre el sistema operativo de cada máquina y las aplicaciones de los usuarios. Las aplicaciones utilizan estos servicios a partir de la interfaz de aplicaciones (API)[19] propia de los middlewares. La ejecución eficiente de una aplicación sobre un entorno distribuidos, estas requieren además de servicios que permitan al usuario interactuar cómodamente y de forma transparente con el sistema, herramientas que controlen y planifiquen automáticamente la ejecución de la aplicación sobre los distintos recursos del entorno.

Tomando en cuenta el estado dinámico del sistema con objeto de aprovechar los recursos de la mejor manera posible para ello se hace necesario un planificador, cuya función principal es la de controlar la ejecución de la aplicación sobre los recursos disponibles del entorno, determinando cuantos recursos son necesarios y cuales recursos son más apropiados para ejecutar las diferentes tareas de la aplicación.

Un buen planificador debe tener en cuenta tres aspectos, velocidad de procesamiento del recurso, latencia y ancho de banda de la red, intuitivamente una buena selección del planificador sería la de recursos rápidos para las tareas con mayor computo, pero en muchas ocasiones el planificador debe tener en cuenta si es preferible seleccionar un recurso lejano y rápido donde se pierde un elevado tiempo en comunicación, o por el contrario, proporcionar el mejor rendimiento de un recurso lento pero cercano.

Todo lo antes mencionado resulta complicado gestionar, lo que se traduce en un problema NP completo para las aplicaciones más aun cuando las mismas poseen dependencias entre ellas que es el caso de estudio en esta investigación. Desafortunadamente, el uso de las buenas herramientas gestoras de workflow está lejos de poder brindar la posibilidad de control más cercano al usuario, el cual pueda de manera sencilla integrar diversas políticas de planificación que supongan una ejecución más eficiente de la aplicación.

Los sistemas utilizados actualmente tienen esas limitaciones en referencia a la ejecución de aplicaciones workflow, las cuales representan un grupo importante en el mundo de las ciencias computacionales, existiendo esta necesidad por parte de los usuarios, proponemos la siguiente solución.

Desarrollar una herramienta que integre por un lado las políticas de planificación existentes en la bibliografía, y comúnmente utilizadas para dar soluciones eficientes a estas aplicaciones, y por el otro permita enlazar cualquier gestor de workflow de los existentes en la actualidad. Por tal motivo en nuestro trabajo, mostramos muestro sistema denominado SchedFlow.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

SchedFlow cuenta con una arquitectura capaz de soportar la ejecución de aplicaciones workflow, y actualmente incluye algunas políticas de planificación existentes en la literatura. Nuestro sistema cuenta con una interface capaz de integrar políticas de planificación de workflow que un usuario quiera incluir en nuestro sistema. Dando la posibilidad de incluir políticas definidas por un usuario sin la necesidad de cambiar toda la arquitectura de un gestor de workflow tarea que suele ser muy compleja sino, y que necesita de los conocimientos avanzados en programación para lograrlo.

Actualmente nuestro sistema puede ser enlazado con gestores de workflow como DAGMan, Taverna, o Karajan, dándole al usuario un abanico de opciones con las cuales realizar diferentes pruebas que le permitan obtener los mejores resultados en tiempo de ejecución de la aplicación workflow que desee ejecutar.

El resto de este artículo consta de las siguientes partes, la Sección 2, hace una breve introducción a los entornos Grid, presentando las características más relevantes, así como su arquitectura. La Sección 3 muestra la arquitectura general, modo de funcionamiento y las definiciones de cada módulo de SchedFlow. La Sección 4 contiene la experimentación realizada, y el análisis de los resultados obtenidos. Para finalizar la Sección 5 muestra las conclusiones del trabajo, y las líneas abiertas de la investigación.

2. Introducción a los Entornos Grid

2.1. Estado del Arte

Actualmente existen aplicaciones que requieren una elevada potencia de cómputo, lo que hace necesaria la creación de infraestructuras capaces de dar soporte a estas nuevas exigencias de las aplicaciones. Ejemplos de estas aplicaciones son las que se aplican en campos de medicina, investigaciones industriales, física de altas energías, entre otras. Para solventar estas exigencias se han buscado diferentes soluciones, desde cambiar la arquitectura interna de la máquina donde se ejecuta la aplicación, hasta ejecutarla en varias máquinas distribuidas en una red local ó incluso a través de internet.

Una solución propuesta para incrementar la potencia de cómputo de una máquina, es incrementar la potencia de su procesador. Sin embargo esta solución no siempre es posible, debido a que la velocidad de los procesadores está limitada por la tecnología

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

y el coste, tampoco siempre es posible encontrar un procesador más rápido.

Otra solución muy utilizada en la actualidad para incrementar la potencia de cómputo es la de trabajar con múltiples procesadores. Los sistemas que resultan de esta idea se les denominan paralelos, y permiten resolver un problema determinado a partir de distintos elementos de cálculo (por ejemplo procesadores), que cooperan con el objetivo de mejorar los tiempos ó el costo en la realización de los mismos, a esto se le conoce como procesamiento paralelo. Ahora bien nuestro enfoque está orientado a la computación distribuida específicamente un Grid, el cual se basa en un conjunto de nodos interconectados mediante una red de alta velocidad que permite resolver problemas individuales de gran envergadura.

2.2. Sistemas Grid

Un entorno Grid heterogéneo, significa que las máquinas donde se ejecutan las aplicaciones pueden tener diferente arquitectura (procesadores, discos duros, memorias), y diferentes conexiones de red (Topología, Ancho de Banda, Latencia). El usuario puede trabajar en este tipo de entornos lanzando sus aplicaciones sin preocuparse dónde lanza el trabajo ni de la topología de la red.

2.3. Características de un Sistema Grid

En los sistemas Grid existen una serie de características importantes a tomar en cuenta, cuando se pretende trabajar en este entorno a continuación presentamos las más importantes de ellas.

- **Concurrencia**: Esta característica de los sistemas Grid, permite que los recursos disponibles en la red puedan ser utilizados simultáneamente, tanto por los usuarios como por los agentes que interactúan en la red dado su carácter de compartir los recursos.

- **Carencia de reloj global**: Las coordinaciones para la transferencia de mensajes entre los diferentes componentes para la realización de una tarea, no tienen una temporización general, está es más bien distribuida entre los componentes.

- **Fallos independientes de los componentes**: Cada componente del sistema puede fallar independientemente, con lo cual los demás pueden continuar ejecutando sus acciones. Esto permite la ejecución de las tareas con mayor efectividad, pues el sistema en su conjunto continua trabajando,

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

dependiendo claro del tipo de aplicación que se este ejecutando.

2.4. Ventajas de los Sistemas Grid

Una de las ventajas de los sistemas Grid es la economía, pues es mucho más económico incorporar servidores y clientes cuando se requiere aumentar la potencia de procesamiento.

- **Mayor confiabilidad**: Al estar distribuida la carga de trabajo en varios recursos, cuando falla de una de ellas no afecta a las demás, el sistema sobrevive como un todo.

- **Mayor Capacidad de Cómputo:** la carga de cómputo la realizan diferentes ordenadores.

- **Capacidad de crecimiento incremental**. Se puede incorporar procesadores al sistema de manera sencilla, incrementando su potencia en forma gradual según las necesidades de los usuarios.

2.5. Desventajas de los Sistemas Distribuidos

El principal problema es el software, cuya implantación y uso del software distribuido, puede representar numerosos inconvenientes aquí queremos destacar los más usuales.

- **Sistema Operativo**: Es necesario tener soporte de diferentes lenguajes de programación y contar con las aplicaciones más adecuadas para estos sistemas.

- **Adaptación de los usuarios**: Estos deben conocer la distribución con la cual van a trabajar para así no obtener fallos en sus trabajos lanzados al Grid.

2.6. Tendencias Futuras de los Sistemas Grid

Heterogeneidad de los componentes: La interconexión, sobre todo cuando se usa Internet se da sobre una gran variedad de elementos hardware y software, por lo cual se necesitan de ciertos estándares que permitan esta comunicación. Los Middleware son las soluciones propuestas, ya que siendo elementos de software permiten una abstracción del tipo de lenguaje de programación y la heterogeneidad subyacente sobre las redes[20].

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

Extensibilidad: Determina si el sistema puede extenderse y ser re-implementado en diversos aspectos, como incluir o quitar componentes.

Seguridad: Reviste gran importancia por el valor intrínseco para los usuarios, es necesario mantener la confidencialidad y protección de los datos contra individuos no autorizados. Además de brindar la fiabilidad de los resultados obtenidos de ejecutar una aplicación.

Escalabilidad: El sistema es escalable si conserva su efectividad, esto significa que al ocurrir un incremento considerable en el número de recursos y en el número de usuarios los resultados se mantengan semejantes.

Tratamiento de Fallos: La posibilidad que tiene el sistema para seguir funcionando ante fallos de algún componente en forma independiente, pero para esto se tiene que tener alguna alternativa de solución tanto de software como de hardware[21].

3. Arquitectura de SchedFlow

El gestor de planificación SchedFlow, es una herramienta diseñada e implementada fundamentalmente como un mecanismo integrador entre las 2 vertientes existentes en el campo de planificadores y gestores de workflow. Nuestra herramienta tiene la posibilidad no solo de la ejecución de un workflow, sin importar la complejidad del mismo tal y como se demuestra más adelante en esta memoria, sino de pode establecer a través de 2 interfaces claramente definidas integrar políticas de planificación y gestores de workflow, una donde se plantea integrar cualquier política de planificación, y la otra donde nuestra herramienta pueda enganchar cualquier gestor de workflow.

Hay que destacar como punto importante que la mayoría de los sistemas existentes solo funcionan con una política o varias pero dejan cerrada la posibilidad de incluir otro sin que ello tenga cambios importantes en su arquitectura.

Desde el punto de vista de la planificación y gestión, podemos decir que SchedFlow brinda las 2 posibilidades en sola una herramienta, dando una solución a las aplicaciones encargadas de gestionar la planificación de un workflow.

Nuestro sistema esta provisto de un **Framework**[22] que permite garantizar los mecanismos necesarios que logren además de la integración

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

una ejecución y funciones simples de monitoreo e información de interés para el usuario que ejecuta una aplicación. Esta claro que no es una tarea fácil construir una herramienta con estas características, ya que como se ha mencionado antes las aplicaciones workflows suelen estar consideradas como una de las que requieren muchísimos recursos, por el nivel de paralelismo que pudiera llegar a mostrar alguna aplicación compleja en un nivel del workflow. En los siguientes puntos de este capitulo se trataran los detalles tanto de la arquitectura como de su funcionamiento.

3.1. SchedFlow

El Gestor de Planificación SchedFlow es el encargado de realizar toda la planificación y ejecución de tareas asociadas a un workflow de la forma más eficiente posible para llegar a una correcta ejecución de los diferentes niveles del workflow de acuerdo a los algoritmos de planificación utilizada para su ejecución.

En la Figura 1 se presentan los componentes principales que constituyen el SchedFlow. Típicamente el usuario, a través de las herramientas desarrolladas podrá realizar la ejecución de su aplicación sobre esta arquitectura que a continuación detallamos:
A través de las herramientas desarrolladas en el SchedFlow, un usuario decide lanzar un workflow para ello tendrá que haber realizado la descripción de su Directed Acyclic Graph (DAG) bajo el formato de DAGMan descrito anteriormente. Para cada trabajo enviado, por el usuario, el SchedFlow realiza los mismos pasos cada vez que una tarea esta lista para ser ejecutada.

En primer lugar el **Controller** recibe la descripción del workflow que debemos ejecutar y realiza un envío del mismo. El **Observer** a partir de la descripción del trabajo y sus requisitos mínimos, usando la información de los ficheros de log del gestor de workflow integrado en nuestra herramienta, y a partir de ahí se van tomando decisiones correspondientes a las tareas que se deben ejecutar, ahí entra en un ciclo en el cual el **Scheduling** va indicando de acuerdo a las tareas que terminan exitosamente la ejecución, cuales son las siguientes tareas que deben ser enviadas de acuerdo a la planificación obtenida, respetando las dependencias existentes entre las tareas. Durante todo el proceso de planificación, el Observer va actualizando la información sobre el estado de los trabajos almacenada en los ficheros de log para su consulta por parte del usuario.

El SchedFlow puede funcionar en una maquina, que se vería como un central manager desde la cual el usuario podrá lanzar su workflow, dado que existe un único gestor que realiza la planificación, tenemos a su vez un único punto de fallo del sistema ya que ahí

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

estarán todos los archivos ejecutables correspondientes a las tareas que se deben ejecutar, y con la cual el Scheduling toma las decisiones correspondientes para alimentar al sistema. Por último, cabe la posibilidad de que cada usuario este lanzando workflow con lo cual todos los recursos estarán disponibles y compitiendo con todos los usuarios, lo cual transforma nuestro sistema en un entorno distribuido.

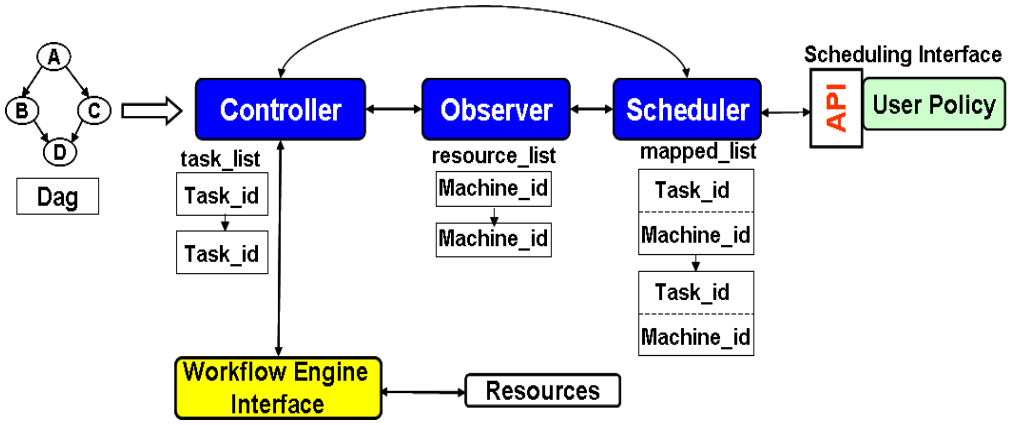

Figura 1. Arquitectura SchedFlow.

Los módulos detallados a continuación muestran cada una de las funciones que ellos cumplen dentro de la arquitectura antes mostrada.

3.2. Módulo Controller

El módulo Controller es el encargado de poner en marcha parte de la arquitectura del SchedFlow, y funciona de la siguiente manera. Una vez recibido el fichero descriptivo con la información del workflow que se desea ejecutar, este realiza la operación de envío del workflow el cual pasa a un primer estado en los ficheros de información, y comienza a ser el encargado de esperar cada una de las tareas siguientes que el modulo scheduling indique que están listas para ejecución. Aquí también se puede tomar decisiones como la de mantener una tarea en espera antes de ser enviada a nuestro gestor de workflow.

3.3. Módulo Observer

El módulo Observer es uno de los principales componentes del SchedFlow, y es el encargado de gestionar todos y cada uno de los estados por los que pasan las tareas del workflow enviado por el usuario, su modo de funcionamiento es el siguiente una vez que el usuario ha escrito su workflow y este ha sido procesado por el

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

Controller, esto significa que tenemos el primer estado de la secuencia que cada tarea tendrá que atravesar para obtener una respuesta exitosa y permitir con ello el paso a las tareas siguientes bajo su nivel de ejecución. Todo esto el Observador lo va revisando desde los ficheros de Log que posee el gestor utilizado, una vez que cada tarea cambia de estado el observador realiza una revisión de estos ficheros y detecta cuando una tarea ha analizado correctamente, y dando información al principal módulo del SchedFlow, el cual indicara cual es la tarea que a continuación debe realizar.

3.4. Módulo Scheduling

El módulo Scheduling es el componente responsable de la planificación de los trabajos, es el encargado de decidir dónde se ejecutaran los mismos. Recibe los cambios de estado del Observer y así tiene cada cierto tiempo los trabajos que han finalizado con lo que mira cuales son las tareas que en ese momento y respetando las dependencias del nivel se pueden ejecutar. Para localizar cuál es el mejor recurso primero llamará al método ***get_resource_list()***, que se actualiza constantemente la lista de recursos dónde se pueden enviar los trabajos, esta lista de recursos dependerá de la cantidad disponible en el ambiente de ejecución, ó de los requisitos colocados por el usuario. Adicionalmente, el Scheduling extrae desde la interfase desarrollada las política de planificación conocidas y utilizadas en el mundo de la planificación de tareas, claro esto ira mejor o pero dependiendo de cada entorno de ejecución utilizado, valorando ciertas métricas de rendimiento como MFLOPS, memoria, entre otras. Así el scheduling podrá basar su decisión de planificación de envío de tareas a ciertas máquinas que se consideren potencialmente mejores garantizando un mejor tiempo de ejecución de la aplicación.

La gestión del trabajo a partir de este momento se lleva a cabo por el gestor de workflow, pero si por alguna razón el envío fallara el scheduling seleccionar ía al siguiente recurso (o conjunto de ellos) de la lista generada para volver a reenviar el trabajo. El número de reenvíos puede ser configurable de manera que tras un número de intentos determinados se pueda decidir que el trabajo ha sido o no ejecutado satisfactoriamente.

3.5. Integración de Políticas de Planificación en el SchedFlow

En esta sección procederemos a describir, el funcionamiento de nuestra API para la planificación de un workflow, detallando las funciones con las cuales un usuario cuenta para ejecutar aplicaciones tipo workflow, haciendo uso de nuestra herramienta de planificación.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

La API consta de 3 Clases fundamentales diseñadas en C++, dichas clases son las siguientes ***Schtask***, ***Schmachine*** y ***Schplanning***, las cuales cuentan a su vez de unos métodos que permiten al usuario de nuestro sistema, realizar planificación de aplicaciones workflow en un ambiente distribuido.

Siguiendo el orden de ideas planteado en esta sección, procedemos a describir las funciones ó módulos de cada una de las clases:

Schtask: Es una de las clases desarrolladas en nuestro sistema, que controla todo lo relacionado con la descripción de las tareas del workflow, y las dependencias que poseen los nodos entre si, para ello utiliza dos sencillos métodos en C++, para indicar básicamente las tareas a ejecutar, estas funciones son:

1. *Task_description* (node_name, node_submit, t_computing), esta función recibe como entradas el nombre del nodo, el archivo o tarea a ejecutar (node_submit), que no es mas que el fichero de envió que necesita DAGMan para ejecutar los trabajos del nodo y así enviarlo a la cola donde esperar hasta que pueda ser asignado a un recurso, y como ultimo parámetro el tiempo de cómputo del nodo, esto de conocerse en caso contrario se posee un método que realiza este calculo, esta función regresa la estructura básica con del fichero de entrada para el SchedFlow.

2. *Task_dependence* (node_name1, node_name2, t_communication), esta función recibe como parámetros de entrada los pares de nodos que se relacionan en el workflow, así mismo como el tiempo de comunicación entre ellos, este valor si se desconoce existe un método que también lo calcula, y regresa las dependencias entre cada par de nodos del workflow con su respectivo tiempo de computo, estos valores también se incluyen en el fichero de entrada para el SchedFlow.

Schmachine: Es otra de las clases del planificador, encargada del manejo de los recursos con los que contaremos a lo largo de la planificación, así como la notificación de los eventos que ocurren a lo largo de la ejecución del workflow, esta clase cuenta con los siguientes métodos:

1. *get_resource_list* (machine_list), esta función recibe como parámetro de entrada, una lista de recursos total de los que se dispone en nuestro entorno de ejecución, y me retorna los recursos disponibles para ejecutar la aplicación en el entorno, también se puede obtener los recursos con mejor Benchmark del entorno.

2. *get_communication_time* (schtask1, schtask2, bandwidth, latency), esta función recibe como parámetros de entrada las tareas a planificar, el ancho

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

de banda del que se dispone, y la latencia asociada a nuestro entorno de ejecución, y me retorna a través de un calculo de media de Benchmark el valor de comunicación entre los nodos del workflow, esto en el caso de que el usuario desconozca toda la jerarquía de comunicación entre los nodos de la aplicación.

3. *get_computing_time* (schtask, task_time), esta función tiene una característica similar al get_communication_time, recibe como parámetros de entrada la tarea asociada del workflow, y el tiempo estimado de ejecución de la tarea, si el usuario también desconoce este valor, realiza un calculo de tiempo de ejecución basado en un Benchmark para cada una de las tareas, y así obtener un valor estimado de ejecución para la tarea asociada al nodo.

Schplanning: Es la última clase con la que cuenta nuestra API, encargada de administrar todo lo relacionado con la asignación de tareas en las máquinas disponibles de nuestro entorno de ejecución, es decir permite a través de la función de planificación organizar todas las tareas y la forma en que estas se ejecutaran, cuenta con las siguientes funciones:

1. *do_scheduling* (schtask, schmachine, priority), esta es la función mas importante de nuestro planificador, la misma recibe como entrada la tarea a planificar, la maquina planificada y la prioridad de ejecución de esta tarea, con lo cual podemos obtener no solo planificación tarea máquina, sino también como desea el usuario que sea la ejecución del workflow, y devuelve la acción de planificación de acuerdo a la heurística utilizada.

2. *undo_scheduling* (schtask), esta función es un valor adicional a nuestro sistema, recibe como parámetro la tarea planificada y permite eliminar esta tarea planificada en un recurso asignado, que por error del usuario o cambio necesario para mejorar las prestaciones de nuestro sistema, este pueda eliminar dicha tarea.

Haciendo uso de la función ***creaScheduler()*** a la cual habrá que agregar la política de planificación que se pretende usar, pues este realiza a través de la función *factory* que se usa para las librerías de objetos SO, y una vez que estén estos pasos se hayan completados solo queda compilar de nuevo el makefile del SchedFlow para que la política pueda ser llamada desde el módulo de planificación haciendo uso de la función do_scheduling descrita en el apartado anterior. Haciendo uso del siguiente método este nos permite integrar una política de planificación no programada en el SchedFlow y que un usuario cualquiera pueda a través de esta interfase realizar la mencionada integración y principal aporte de nuestra herramienta. El siguiente código muestra esta función:

```
1. creaScheduler(){
```

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

```
2. return new RoundRobinScheduler;
}
3. void *hndl = dlopen("libnewshapes.so", RTLD_NOW);
4. if(hndl == NULL){
     4.1 cerr < < dlerror() < < endl;
     4.2 exit(-1);
     }
5. void *creaScheduler = dlsym(hndl, creaScheduler);
6. Scheduller *aScheduller **=(creaScheduler)();
```

4. Experimentación y Resultados

En este capítulo se detallan los experimentos realizados para mostrar la funcionalidad del SchedFlow, y donde queda reflejado el uso del mismo utilizando para ello la aplicación Montage como workflow experimental. Para asegurar en una primera etapa la confiabilidad del gestor de planificación SchedFlow desarrollado, cada uno de los módulos que lo componen se le realizados las respectivas pruebas de funcionamiento, incluyendo cada módulo que aportaba los resultados satisfactorios para asegurar la fiabilidad del gestor. Este proceso consistió en el envío de workflows pequeños para la validación de los estados de una tarea. Adicionalmente se hacia una descripción detallada de los cambios ocurridos en los diferentes archivos que registraban estos cambios, y luego se valido con workflows pequeños las interfaces de las cuales están compuesto el SchedFlow incluyendo algunas políticas de planificación en el gestor, las cuales usaremos más en detalle con una aplicación real que introduce una complejidad más grande al momento de planificarla.

4.1. Aplicación Montage

Montage [25][26] es una aplicación de tecnología de la tierra (ESTO) para tecnologías de cómputo (CT), es un proyecto que desplegará un servicio mosaicos de imagen astronómicas portables, esta aplicación computa intensivamente el procesamiento de las imágenes, todo ello para el observatorio virtual nacional (NVO). Aunque Montage está desarrollado al servicio de cálculo y de procesamiento intensivo de datos, para la astronomía y la comunidad científica que realiza investigaciones sobre imágenes astronómicas también resulta útil para los que trabajamos aplicaciones workflows ya que la misma tiene un comportamiento de flujo de tareas, que se van completando hasta procesar el mosaico deseado, en este proyecto también estamos ayudando a tratar un problema que atraviesa la tierra y la ciencia del espacio, y es cómo tener acceso eficientemente a *multi-Terabyte*[23][24] de procesos, *datasets* distribuidos entre otras cosas.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

En ambos las comunidades, los *datasets* son masivos, y se almacenan adentro archivos distribuidos que están, en la mayoría de los casos, alejados con respecto a los recursos de cómputo disponibles.

4.2. Entorno de Experimentación

En esta sección presentaremos el entorno de ejecución con el cual contamos para realizar los experimentos del SchedFlow ya utilizando políticas de planificación reales programadas para nuestro sistema, además las pruebas realizadas han sido en un entorno distribuido real, lo que representa una apreciación mas clara de los resultados obtenidos para la aplicación workflow de prueba en Grid reales, en nuestro caso el entorno donde se ejecutaron todas las pruebas esta compuesto por 135 nodos, con una arquitectura Intel y un sistema operativo Linux dando como media de benchmark para los recursos disponibles de 0,55 Gflops.

4.3. Experimentos Realizados

El caso más interesante para la planificación de workflow, son aquellos están formados por numerosos trabajos distribuidos en múltiples niveles a lo largo de su estructura, que requieran recursos de altas prestaciones para el mejor rendimiento de las aplicaciones, por esta razón procedemos a dar en detalle los experimentos realizados, que tipo de algoritmos de planificación se usaron y cuales fueron las propuestas para probar la herramienta desarrollada.

A la hora de realizar la planificación del workflow hay que tener en cuenta tanto los tiempos de ejecución de cada uno de los nodos como los tiempos de transferencia de ficheros entre ellos. Para lo cual se han realizado medidas de los tiempos de ejecución en una máquina piloto del cual se obtienen los primeros datos para comenzar la planificación con el SchedFlow, el ancho de banda inicial es estimado de acuerdo al tiempo de transferencia de ficheros entre 2 recursos cualesquiera del entorno de ejecución obteniendo una media de los mismo y siendo este uno de los parámetros de entrada para la política de planificación.

Para hacer las pruebas al SchedFlow se plantearon algunos experimentos que permitirán, por un lado evaluar el correcto funcionamiento de nuestra herramienta, y por otro mostrar los resultados de la aplicación seleccionada para ello. Pudiendo observar que algoritmo de planificación es el que mejor ajusta a los escenarios planteados.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

La primera fase de experimentación consiste en realizar la ejecución del Montage con el algoritmo de planificación aleatoria, Min-Min [25], y Replanificación simple que detallaremos más adelante en este capítulo, el método de trabajo será el siguiente ya que nuestro objetivo principal de momento es la evaluación de nuestra herramienta, se hace necesario no solo validar la ejecución del SchedFlow, sino al mismo tiempo ver que resultados que se obtienen de la ejecución de una aplicación real, compleja tal es el caso del Montage, utilizando para ello diferentes políticas de planificación bajo un ambiente totalmente distribuido.

Los resultados mostrados en la siguiente sección, son los obtenidos de ejecutar un número determinado de veces la aplicación, con diferentes políticas de planificación utilizando el SchedFlow y así ver el funcionamiento del mismo, y dar un paso adelante sobre la orientación que tendrá nuestro trabajo en un futuro próximo.

4.4. Resultados y Análisis Experimentales

En la siguiente sección se presentan los resultados obtenidos de realizar la ejecución de la aplicación Montage utilizando el SchedFlow, con las diferentes políticas de planificación. En nuestra herramienta contamos con 3 políticas de planificación y son las que utilizamos y en las cuales se basan los resultados obtenidos con ellas. Partiendo del ambiente de ejecución mencionado en el punto anterior y que cuenta con 135 recursos disponibles, se planteo una primera prueba que consintió en realizar un envío del workflow Montage con la política de planificación aleatoria, es decir cada tarea seria distribuida de manera aleatoria, un poco para observar el comportamiento de la aplicación durante toda la ejecución, y a partir de ellas encontrar algún método sencillo, que nos permitiera mejorar los tiempos de ejecución, una vez realizada estas pruebas se obtuvieron los siguientes resultados que e muestran en la Figura 2.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

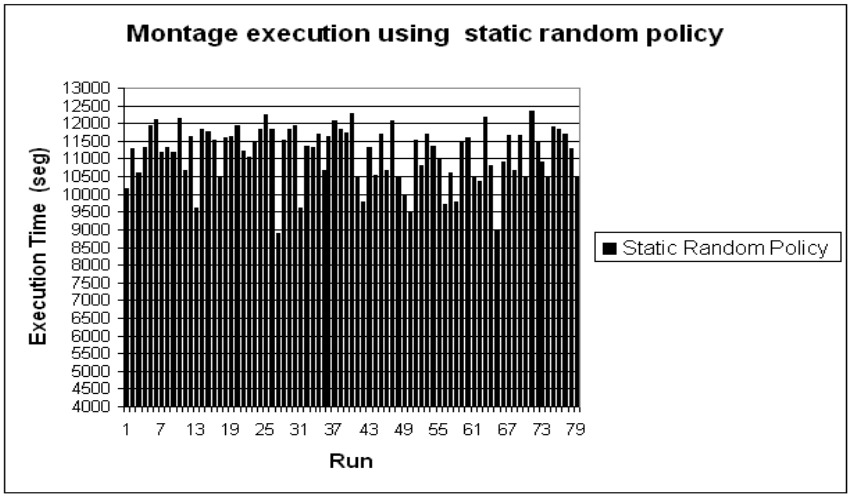

Figura 2: Ejecución del Montage con la Política de Planificación Aleatoria

Analizando este experimento, se puede ver que la ejecución del workflow, en muchos casos tiene presencia de muchas oscilaciones en el tiempo de ejecución, incluyendo tiempos de respuesta por tarea muy elevados, con lo cual se nos plantea un problema, que causan las oscilaciones y como obtener mejores tiempos de ejecución, porque existen tantos retardos para la ejecución del workflow, haciendo uso de nuestra herramienta logramos visualizar que sucedía durante la ejecución, se pudo observar que durante muchas de las ejecuciones las tareas en los nodos permanecían durante mucho tiempo en estado de suspendidas, entiéndase por este estado (suspendido), a la tarea que se esta ejecutando en un recurso y de manera inesperada es sacada de su ejecución, esto para algunos casos llegaron a ser hasta horas de suspensión por tarea, teniendo una penalización grande para la ejecución independientes de los nodos, que en algunos casos tardaban solo minutos ejecución de la tarea, esto nos hizo pensar en una posible solución para adaptar una nueva política de planificación en nuestro sistema.

Partiendo de este hecho y analizando lo que tenemos, y pudiendo obtener el estado del trabajo en cada momento, nos planteamos integrar la política de planificación Min-Min dinámica, que consiste en lo siguiente. Utilizando nuestro sistema de planificación SchedFlow, y tomando la notificación de estados que nos proporciona el observador, durante la ejecución de cada nodo pues generamos la siguiente relación de planificación, si la tarea posee un estado de suspendido por cualquier causa, entonces realizamos una reasignación del trabajo a otro recurso que se encuentre disponibles, esta información es enviada al controlador y este realizara un envió de esta tarea al nuevo recurso, esta política planteada bastante sencilla nos permitirá

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

evaluar de manera más concreta si el API incluida en el planificador del SchedFlow funciona eficazmente ante este cambio de estado, dejando claro la potencialidad de nuestra herramienta, esto nos presupone un mejor comportamiento cuando se utilicen las políticas de planificación como CPOP ó HEFT, esto por citar algunas de las existentes y muy utilizadas en la planificación de workflows. Siguiendo el orden de los experimentos realizados con Min-Min dinámico, los resultados de esta prueba se ven en la Figura 3.

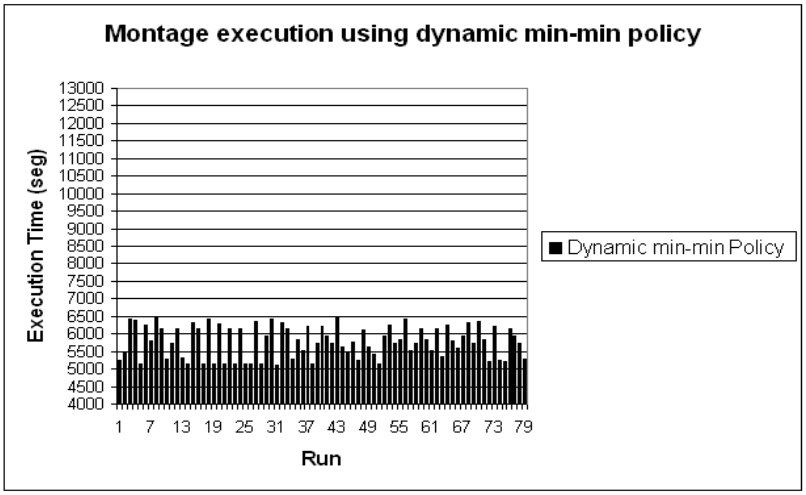

Figura 3: Ejecución del Montage con la Política de Planificación Min-Min dinámica.

Los resultados obtenidos, después de utilizar la política que hemos mencionado anteriormente, llamada Min-Min dinámica, si analizamos estos resultados podemos observar 2 detalles importantes, el primero de ellos seria, el comportamiento de nuestra aplicación inicialmente es menos oscilatorio, que con la ejecución aleatoria, con lo cual se balancea un poco la carga de nuestras tareas en los recursos disponibles, y por otro lado tenemos que nuestra herramienta alcanza en promedio una mejora del rendimiento de ejecución del 31,68% sobre la política aleatoria, esto nos permite confirmar que con políticas de planificación más sofisticadas, se pueden llegar a mejoras importantes, en tiempo de ejecución y balanceo de carga, claro esto dependerá del tipo de aplicación que estemos utilizando, y que rendimientos nos permite alcanzar el mismo sobre la base del conocimiento de la aplicación.

5. Conclusiones y líneas Abiertas

Los sistemas distribuidos presentan un nuevo entorno para el uso compartido de recursos, en el que además de una gran potencia de cálculo se permite la colaboración entre grandes grupos de científicos para abordar nuevos problemas.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

En este tipo de entornos heterogéneos se hace especialmente útil un *middleware* ó software que proporcione una visión unificada de los recursos. SchedFlow ha sido un trabajo creado para extender el uso de planificadores distribuidos a nuevos tipos de aplicaciones poco tratadas hasta el momento en entornos reales, tal es el caso de las aplicaciones tipo workflows. SchedFlow se introduce como un mecanismo novedoso que gestionen estas aplicaciones de forma automática y efectiva en el que se orienta principalmente la integración de políticas de planificación y gestores de workflows.

A la hora de realizar la gestión de los recursos en un sistema distribuido es necesario tener en cuenta las características propias frente a otros sistemas tradicionales, en especial la falta de control sobre los recursos y las tareas suelen dar como resultados tiempos de ejecución elevados, para aplicaciones que solo necesiten escasas horas para completar su ejecución satisfactoriamente.

En este trabajo se mostró la existencia de una serie de propuestas para abordar el problema de gestión de workflows en sistemas Grid, aunque ninguno completa todos los pasos necesarios para la integración tanto de gestores como de políticas de planificación de workflows. Por este motivo hemos propuesto SchedFlow como un planificador capaz de integrar de manera sistemática los gestores existentes y las políticas de planificación sin que ello requiera realizar cambios importantes en la arquitectura de nuestro sistema, logrando así uno de los principales objetivos de esta investigación.

Nuestro sistema de gestión de planificación ha pasado una serie de pruebas para su validación y actualmente es capaz de ejecutar aplicaciones workflow de biomedicina, análisis distribuido en física de altas energías (HEP), prevención de inundaciones, fotografías espaciales como es el caso de la aplicación utilizada para las pruebas. A través del SchedFlow se ha enviado la aplicación Montage con diferentes políticas de planificación, estas políticas requerirán la evaluación de los factores que influyen en la mejora del rendimiento y el diseño de estrategias que los tengan en cuenta para mejorarlo, utilizando para tal fin los cambios de estado de cada nodo en tiempo real lo que hace de nuestro sistema, un evaluador dinámico, aprovechando la información disponible sobre los recursos para la selección del más adecuado para las tareas del workflow.

Adicionalmente podemos mostrar que los resultados obtenidos con nuestro sistema son bien claros, que la potencialidad de poder contar con una herramienta de integración de políticas de planificación son un factor importante dentro de la ejecución de workflows científicos, donde la principal preocupación es mejorar los tiempos de ejecución con los recursos de que se disponen. Adicionalmente los resultados alcanzados de alguna manera

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

muestran una distribución de las tareas ajustada a las realidades de la aplicación llegando a mejorar con políticas más sofisticadas hasta un 30% el tiempo de ejecución.

Las líneas abiertas que se tienen seria usar esquemas más complejos de multiprogramación, donde exista más de un workflows, con distintas prioridades para distintos tipos de tareas. Mejoras generales en el software. Reducción de tiempos con políticas más sofisticadas eliminación de posibles errores en el código. Inclusión de mecanismos de reserva temporal que eviten el *matching* con recursos recientemente asignados cuyo estado no se ha actualizado, poder ejecutar con el SchedFlow más de 2 workflows simultáneamente con su planificación.

Bibliografía

[1] Grids as Production Computing Environments. The engineering aspects of nasa information power grid, proc. 8th symposium on hpdc, ieee computer society press, 1999.

[2] G. Asada, T. Dong, F. Lin, G. Pottie, W. Kaiser, and H. Marcy. Wireless integrated network sensors: Low power systems on a chip, 1998.

[3] Rajkumar Buyya. Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing, 2005.

[4] Guillén C. Descripción de patrones arquitectónicos para sistemas distribuidos, tesis de maestría. universidad central de venezuela, caraca venezuela-2002.

[5] H. Chen and D. D. Ling. Power supply noise analysis methodology for deep-submicron vlsi chip design, pág 638-643, 1997.

[6] Issac P Chua Ching Lian, Tang F and Krishnan A. Gel: Grid execution language. journal of parallel and distributed computing, 2005.

[7] R.D. Cupper. Process and device scheduling, the computer science and engineering, crc press, pág 1677-1708, 1997.

[8] Enol Fernandez del Castillo. Crossbroker: Gestión de recursos en crossgrid 2005.

[9] Yolanda Gil Carl Kesselman Gaurang Mehta Karan Vahi Kent Blackburn Albert Lazzarini Adam Arbree Richard vanaugh Ewa Deelman, James Blythe and Scott Koranda. Mapping abstract complex workflows onto grid environments, journal of grid computing, pages 25-39, 2003.

[10] Yolanda Gil Carl Kesselman Gaurang Mehta Karan Vahi Scott Koranda Albert Lazzarini Ewa Deelman, James Blythe and Maria Alessandra Papa From metadata to execution on the grid: The pegasus pulsar search technical report, griphyn, http://www.griphyn.org/documents, 2003.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 3 Edición No 1 Enr/Jun – Año 2009

[11] D.G Feitelson. Job scheduling in multiprogrammed parallel systems, 2002.

[12] I. Forter. internet computing and the emerging grid, nature, 2000.

[13] I. Foster and C. Kesselman (ed.). The grid: Blueprint for a new computing infrastructure, 1999.

[14] Jacob. D. S. Katz. E. Deelman G. B. Berriman, A. Bergou. Montage: A grid enabled image mosaic service for the national virtual observatory, adass xiii asp conference series vol. xxx, 2004.

[15] Genome@home. http://genomeathome.stanford.edu, 2000.

[16] Salim Hariri Haluk Topcuouglu and Min youWu. Performance-e_ective and low-complexity task scheduling for heterogeneous computing, ieee trans. parallel distrib. syst. pages 260-274, 2003.

[17] Juliane Denhert Hendrik Eshuis. Reactive petri nets for work_ow modeling, in w.m.p. van der aalst and e. best, editors, application and theory of petrinets 2003,volume 2679 of lecture notes in computer science, pages 295-314. springer-verlag, 2003.

[18] et al I. Clarke. Grid resource management state of art and trends, kluwer publishers, freenet: A distributed anonymous information storage and retrieval system, in designing privacy enhancing technologies, springer-verlag, berlin 2001.

[19] I.Foster. The open grid services architecture, 2001.

[20] Rajkumar Buyya Jia Yu. A taxonomy of scienti_c work_ow systems for grid computing., Record 34(3): 44-49 (2005).

[21] Maria M. Lopez, Elisa Heymann, and Miquel A. Senar. Analysis of dynamic heuristics for work_ow scheduling on grid systems. ispdc, 0:199_207, 2006.

[22] Jesús Marco. Grids y e-ciencia jornadas técnicas rediris 2003.

[23] Paul Lu Mark Goldenberg and Jonathan Schae_er. Trellisdag: A system for structured dag scheduling, 2001.

[24] Dagman metascheduler http://www.cs.wisc.edu/condor/dagman.

[25] M. Mutka and M. Livny. The available capacity of a privately owned workstation environment, 2003.