Propuesta de Tesis:
Calendarización de flujos de trabajo en cómputo en la nube

Fernando Aguilar Reyes

Maestría en Ciencias en Computación
Instituto Tecnológico Autónomo de México
Río Hondo #1, Progreso Tizapán, Del. Álvaro Obregón, 01080
México, Distrito Federal

6 de enero de 2014

Resumen

Los flujos de trabajo, o conjunto de pasos que modelan la ejecución de un proceso, son utilizados en aplicaciones como el etiquetado de proteínas o el ajuste de cuentas bancarias al final del día. Como requieren vastos recursos computacionales, se han empleado enfoques de cómputo distribuido (clusters y grids) para ejecutar estos flujos. Sin embargo, el enfoque de cómputo en la nube ha tomado un gran interés tanto por la academia como por la industria, porque el esquema de pagos "pay as you go" y la elasticidad de los recursos de la nube posibilitan la ejecución de estos flujos sin necesidad de contar con infraestructura especial. Una parte importante para la ejecución de los flujos es la calendarización de sus tareas. Con ello, podríamos hacer planeaciones que maximicen el uso del sistema distribuido o que minimicen el tiempo de ejecución total. En esta propuesta de tesis, proponemos crear un calendarizador de flujos de trabajo que tome en cuenta las características únicas del enfoque de cómputo en la nube para que se pueda minimizar el tiempo de ejecución y, también, el dinero utilizado por la renta de los servicios en la nube.

Propuesta de Tesis

Análisis de antecedentes: Revisión inicial de literatura

Flujos de trabajo

Entendemos que un flujo de trabajo, o *workflow* en inglés, es un conjunto de pasos que modelan la ejecución de un proceso [5]. Un ejemplo de un flujo de trabajo es el que el se desarrolló en la Escuela Imperial de Londres para el proyecto *e-Protein* [7], cuyo objetivo era la idenficación y anotación de partes de proteínas que expliquen la estructura y la función de dicha proteína.

Una proteína está compuesta de varios aminoácidos y se estima que existen al menos 500 aminoácidos diferentes [9], dando lugar a un número muy grande de posibles proteínas. De este modo, los investigadores de la Escuela Imperial diseñaron un flujo de trabajo compuesto de varios programas especializados para hacer la anotación de proteínas. En la figura 1 podemos ver el flujo de trabajo [7], donde las flechas indican el flujo de datos y los colores indican el tiempo de ejecución aproximado. Las cajas (o nodos) del diagrama son los diferentes programas que se utilizan para hacer el proceso de anotación de proteínas. Así, un flujo de trabajo puede ser representado por un grafo dirigido acíclico.

Por lo general, los flujos de trabajo requieren de un alto poder de cómputo, lo cual hace prohibitivo su ejecucion en una sola computadora. Por ello, diversos enfoques se han aplicado para distribuir la ejecución de un flujo de trabajo entre varias computadoras. De acuerdo a Buyya et al., los enfoques de cómputo más importantes para los flujos de trabajo son los *clusters*, los *grids* y las *nubes* [4]. A continuación, explicaremos cada uno de los enfoques.

- Los *clusters* son sistemas distribuidos, paralelos, compuestos de varias computadoras que son vistas como un único recurso de cómputo [4]. Un ejemplo de un cluster es la instalación de la Universidad Autónoma Metropolitana, campus Iztapalapa, llamada *Aitzaloa*, compuesta por 270 nodos de cómputo, cada uno equipado con dos procesadores Intel Xeon Quad-Core y 16GB en RAM; los nodos están conectados entre sí por medio de switches Ethernet e Infiniband. La capacidad real de cómputo del cluster Aitzaloa es de 18.4 teraFLOPS [2].

- Los *grids* son sistemas distribuidos, paralelos, compuestos de computadoras autónomas y geográficamente distribuidas que pueden trabajar en conjunto o de manera independiente de acuerdo a los objetivos, políticas y mecanismos de uno o varios administra-

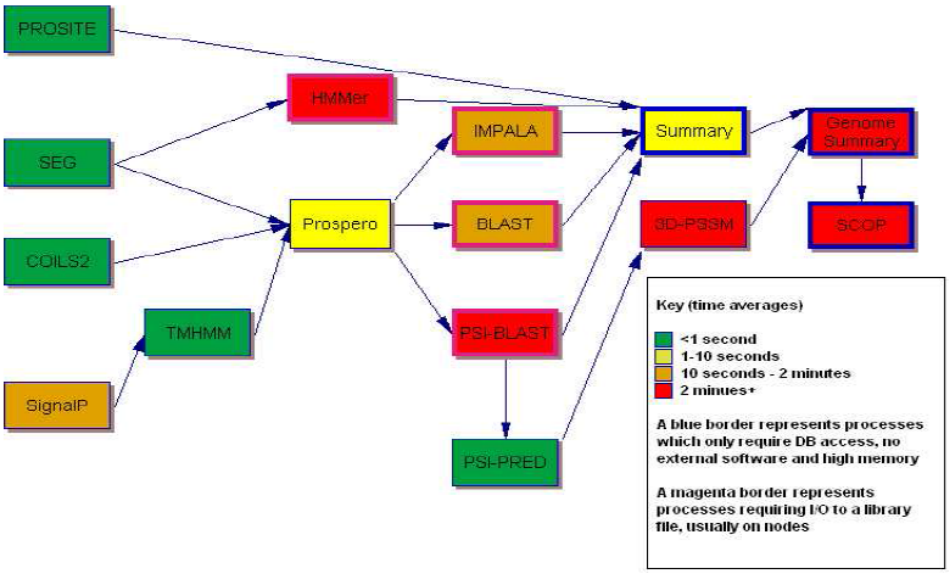

Figura 1: Flujo de trabajo para la anotación de proteínas, diseñado en la Escuela Imperial de Londres para el proyecto *e-Protein*

dores del sistema, es decir, un grid puede ser compartido entre varias instituciones [4]. El proyecto *LANCAD*[1] es un buen ejemplo, pues une el cluster *Aitzaloa* de la UAM, el cluster de la UNAM *KamBalam*, y el cluster *Xiuhcoatl* del CINVESTAV por medio de una red de fibra óptica instalada en las estaciones del Sistema de Transporte Colectivo Metro. La suma de la potencias reales de cada *nodo robusto* del grid es de 48.55 teraFLOPS [1].

- Las *nubes* (clouds) son sistemas distribuidos, paralelos, compuestos de computadoras o máquinas virtuales interconectadas que son aprovisionadas para usarse como uno o varios recursos de cómputo, de acuerdo a un contrato de nivel de servicio acordado entre el proveedor de la nube y el cliente [4]. Empresas nuevas y existences proveen servicios de cómputo en la nube, tales como GoGrid, Rackspace, Amazon, Microsoft, IBM, Oracle, entre otras. La forma en que operan es la siguiente: se paga cierta cantidad por utilizar servicios de cómputo o almacenamiento durante determinado tiempo. Así, los clientes no tienen que invertir grandes cantidades de dinero para contar con una gran infraestructura como en el caso de los los clusters y los grids.

Cómputo en la nube

El *cloud computing* o enofque de cómputo en la nube está tomando mucho interés tanto por la industria como por la comunidad científica, porque hace accesible una gran cantidad

[1]Laboratorio Nacional de Cómputo de Alto Rendimiento

de recursos computacionales con cantidades razonables de presupuesto.

Cuando trabajamos con flujos de trabajo en los dos primeros enfoques, se utiliza software para administrar la ejecución del flujo de trabajo. Estos programas planean y organizan la ejecución de un flujo de trabajo, es decir, *calendarizan*. Por ejemplo, Open Grid Scheduler[2] tiene algoritmos para distribuir tareas paralelas —programadas con la herramienta Parallel Virtual Machines[3] o con alguna biblioteca que implemente Message Passing Interface[4], como OpenMPI[5]— en un grid. También tiene políticas y mecanismos para administrar trabajos secuenciales, por medio de un de un sistema que pone en una cola trabajos de procesamiento en lotes.

Pero, ¿cómo administramos flujos de trabajo que se ejecutan en la nube? Intuitivamente, podríamos pensar que el proveedor en la nube tiene software especializado para administrar el uso de las máquinas virtuales y los sistemas de almacenamiento. Lamentablemente, estos programas sólo administran la ejecución de las máquinas virtuales que pida el cliente, mas no administran flujos de trabajo en ejecución.

Podríamos, entonces, crear un cluster con máquinas virtuales en la nube y utilizar los administradores de flujo de trabajo que se utilizan en clusters locales, como Open Grid Scheduler. De hecho, en un experimento elaborado por ScalableLogic, la empresa que mantiene gran parte del código fuente de Open Grid Scheduler, se creó un cluster virtual en la nube con 10,000 instancias de cómputo del servicio Amazon AWS [3], con el propósito de mostrar la capacidad del software para administrar tal cantidad de nodos de cómputo.

Aunque la solución anterior suena viable, el enfoque de la nube agrega una característica que no está presente ni en los grids ni en los clusters: podemos solicitar nuevas instancias de cómputo sin ninguna restricción física, el único límite es nuestro presupesto. Si tuviéramos un cluster propio para la ejecución de los flujos de trabajo, nuestro límite está marcado por las características del hardware, no podemos aumentarlo de manera "instantánea". De manera análoga, aunque podamos unir varios clusters distribuidos geográficamente, estamos limitados a la capacidad total del grid, sin tomar en cuenta si éste es compartido por varias instituciones. Además, el hecho de que existan varios proveedores de servicios de cómputo en la nube nos da la oportunidad de evaluar diferentes presupuestos que cada empresa ofrece para la ejecución de nuestro flujo de trabajo, dejando abierta la posibilidad de elegir la configuración más conveniente.

De esta forma, podríamos pensar en un calendarizador de flujos de trabajo que utilice el enfoque de la nube, tomando en cuenta las características únicas que hacen diferente este enfoque de los clusters y los grids. Con ello, podríamos elegir si queremos que el flujo sea ejecutado lo más rápido posible o que se optimice una cantidad fija de recursos; o mejor aún, ejecutar el flujo en el menor tiempo utilizando la menor cantidad de dinero.

[2] `http://gridscheduler.sourceforge.net/`

[3] `http://www.csm.ornl.gov/pvm/pvm_home.html`

[4] `http://www.dmoz.org/Computers/Parallel_Computing/Programming/Libraries/MPI/`

[5] `http://www.open-mpi.org/`

Problemas similares

Para indagar si es posible diseñar y construir el calendarizador con cómputo en la nube, podríamos investigar problemas similares, como la calendarización de procesos en un sistema operativo o la calendarización de flujos de trabajo en clusters y grids.

Calendarizadores en sistemas operativos

Los sistemas operativos multitareas utilizan un administrador de tareas para repartir los recursos de cómputo (tiempo de CPU y memoria). Esta organización se ejecuta varias veces entre varios procesos, dando la ilusión de que todos los procesos se ejecutan al mismo tiempo. Con las limitaciones en disipación en calor, los fabricantes de microprocesadores optaron por diseñar CPU's con varios núcleos de procesamiento en un sólo chip. Por ello, los diseñadores de sistemas operativos también tuvieron que cambiar los algoritmos de calendarización de tareas. ¿Cómo es que estos calendarizadores están relacionados con los calendarizadores de flujos de trabajo con cómputo distribuido? La respuesta está en que a partir de las ideas con las que se diseñan estos calendarizadores, podemos proponer un calendarizador para flujos de trabajo.

Por ejemplo, en un trabajo elaborado por Saifullah et al. [8] proponen una forma de descomposición de tareas paralelas homogéneas, de tal modo que se puedan identificar segmentos paralelos que puedan ser ejecutados por un CPU multinúcleo. Después, los autores afirman que su modelo de descomposición de tareas puede ser aplicado a tareas con dependencias, las cuales son modeladas con grafos dirigidos acíclicos.

El trabajo de Saifullah et al. propone un método determinísitico para descomponer tareas con secciones secuenciales y paralelas en una forma que sean fácilmente calendarizables con recursos homogéneos, es decir, de la misma capacidad, como son los núcleos de un microprocesador. Sin embargo, en cómputo en la nube es deseable contar con recursos no homogéneos y especializados con el fin de destinar ciertas tareas del flujo de trabajo que tengan una prioridad más alta comparada con las otras tareas que componen el flujo de trabajo. Pero, la idea de la descomposición de tareas es de gran ayuda para dar perspectiva al problema de calendarización de flujos de trabajo en la nube, ya que si queremos planear y organizar en el tiempo y espacio las diferentes tareas del flujo de trabajo, es necesario que hagamos alguna descomposición de tareas para poder repartir las tareas entre los recursos disponibles en la nube.

Calendarizadores en clusters y grids

Ahora bien, antes del repunte del enfoque del cómputo en la nube, se desarrollaron diversos programas para calendarizar flujos de trabajo en clusters y grids. En el trabajo de Jia Yu y Rajkumar Buyya [11] se hace una compilación de algoritmos de calendarización de flujos de trabajo para grids. Destaca en este trabajo la gran clasificación de los algoritmos en dos grandes grupos: algoritmos de mejor esfuerzo (best-effort) y algoritmos de calidad de servicio (quality-of-service). Los primeros son usados en grids compartidos, mientras que los otros tienen un uso más comercial, porque se asume que existe un costo por usar el grid. El

primer grupo de algoritmos se caracteriza porque tratan de minimizar el tiempo de ejecución total del flujo de trabajo. El segundo grupo, los algoritmos de calidad de servicio tratan de satisfacer ciertas restricciones que impone el usuario sobre los recursos a utilizar.

Sin duda, la compilación de algoritmos de calendarización hecha por Yu y Buyya ayuda dar esclarecer los esfuerzos actuales por asignar recursos computacionales a tareas de un flujo de trabajo. El reto con este trabajo es hallar la forma de adaptar estos algoritmos a las características únicas del cómputo en la nube.

Otro trabajo destacado es el de Liu et al. [6] en donde hablan de un algoritmo de calendarización de ejecución de múltiples flujos de trabajo que son simples de calcular, pero que son numerosos. Éstos son ejecutados sobre servicios de cómputo en la nube. Los autores presentan una solución en donde se da prioridad a la optimización del costo necesario para ejecutar el flujo de trabjo, sacrificando un poco el tiempo de ejecucion total. A grandes rasgos, el algoritmo de calendarización relaja los tiempos límite de ejecución de las tareas de los flujos, con el motivo de no forzar competencia sobre los recursos más baratos como lo hace el algoritmo Deadline-MDP. El algoritmo de calendarización desarrollado, CTC, utiliza en promedio 15 % menos tiempo de ejecución que el algoritmo Deadline-MDP y también reduce en promedio 20 % el costo de ejecución reportado por el algoritmo de comparación.

La idea importante del trabajo de Liu et al. es el hecho de ejecutar múltiples flujos de trabajo en un grid porque, normalmente, se diseñan algoritmos para calendarizar un sólo flujo de trabajo con múltiples tareas.

Calendarizadores en cómputo en la nube

Los trabajos de calendarización de flujos de trabajo en la nube son variados. Un ejemplo notable es el trabajo de Buyyaa et al., en el cual proponen que un esquema de mercado regido por la oferta y demanda de servicios en la nube podría ayudar a que tanto los usuarios puedan satisfacer sus necesidades de cómputo y como los proveedores maximicen las utilidades que podrían generar sus servicios [4]. Esto sería posible si existieran agentes o brokers que negociaran con los usuarios y los proveedores los niveles de servicio requeridos para sus necesidades. Esta idea está plasmada en Aneka, un administrador de flujos de trabajo que puede optimizar los recursos (tiempo o presupuesto) requeridos de los usuarios.

La representación del flujo de trabajo y la calendarización

Por otra parte, la representación del flujo de trabajo juega un papel importante en la calendarización, junto con la representación (modelo) de los recursos disponibles, ya que con estos dos elementos tenemos que encontrar un mapeo entre tareas y recursos que permita ejecutar el flujo de trabajo de acuerdo a lo planeado por el algoritmo de calendarización.

Se había dicho al principio del documento que los flujos de trabajo se pueden modelar como grafos dirigidos acíclicos. Sin embargo. Hay algunos flujos de trabajo que requieren ser representados con detalles adicionales. Por ejemplo, supongamos que queremos modelar un flujo con tres tareas, A, B y C, donde B y C dependen del resultado de A, pero solamente

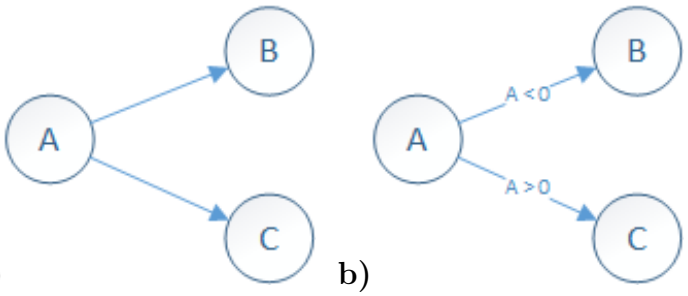

Figura 2: a) Flujo de trabajo con una tarea condicional, modelado sólo con grafos dirigidos acíclicos. b) Flujo de trabajo con una tarea condicional modelado con un grafo dirigido acíclico con anotaciones

queremos ejecutar la tarea B ó la tarea C, dependiendo de la salida de la tarea A. Entonces, se dice que A es una *tarea condicional*.

Si representáramos nuestro flujo de trabajo con un grafo dirigido acíclico, entonces tenemos tres tareas A, B y C, representadas por nodos en un grafo dirigido acíclico. Ahora, decimos que las tareas B y C dependen de la tarea A. De esta forma, dibujamos dos flechas que salen del nodo A y que apuntan, cada una, a los nodos B y C. La figura 2 a) muestra nuestra tarea condicional representada como un grafo dirigido acíclico. Sin embargo, con esta representación, no tenemos forma de representar la condición de ejecutar la tarea B o la tarea C dependiendo del resultado de la la tarea A. Por lo tanto, el modelo de flujos de trabajo con grafos dirigidos acíclicos (nodos y aristas, solamente) no resulta suficiente para expresar flujos de trabajo más complejos.

De esta forma, podriamos agregar a las aristas del grafo una etiqueta con información adicional, como la condición que deben cumplir para que ocurra la transición que define dicha arista. En la figura 2 b) podemos ver nuestro grafo dirigido acíclico con una notacion extendida.

Así, una vez definido el modelo de un flujo de trabajo, podemos aplicar una gran variedad de métodos de optimización accesibles a la comunidad científica para generar un calendarizador de flujos de trabajo que pueda optimizar de manera simultánea el tiempo y el presupuesto disponibles.

Aunque ya hay desarrollos enfocados a buscar un modelo suficiente para los flujos de trabajo actuales, es necesario encontrar una forma de modelar los principales elementos que componen un flujo de trabajo, a saber:

- las tareas, que representan cada uno de los pasos que componen el flujo;
- las restricciones, que son las limitaciones a las que estamos sujetos para ejecutar el flujo. Dentro de la amplia gama de restricciones que podemos formular, las más importantes en el contexto del enfoque de cómputo en la nube son:
  - tiempo,

- dinero
- y recursos disponibles del proveedor de servicios en la nube.

De hecho, en el trabajo de Marek Wieczorek et al. se señala que es razonable separar en dos modelos la representación del flujo de trabajo y la representación de los recursos [10], para facilitar la construcción de una arquitectura que sea escalable para el último.

Ahora, el problema de la calendarización de flujos de trabajo con restricciones intradependientes es, en general, un problema NP-completo [10].

Objetivo de la investigación

Hacer un calendarizador de flujos de trabajo que optimice de manera simultánea el tiempo y el presupuesto necesario para ejecutar el flujo de trabajo, tomando en cuenta las restricciones de recursos disponibles. Una parte fundamental de este trabajo es que el calendarizador se encuentre adaptado a las características de la nube, que son la elasticidad de los recursos y la integración de un modelo económico.

Preguntas de investigación

1. ¿Cuál es la mejor forma de modelar un flujo de trabajo de manera tal que pueda ser ejecutado en un entorno de cómputo en la nube?

2. ¿Cómo podemos evaluar el consumo de tiempo y el presupuesto necesario para ejecutar un flujo de trabajo en la nube?

3. ¿Cómo se puede optimizar el consumo de tiempo y el presupuesto del cliente necesario para ejecutar n flujo de trabajo en la nube?

4. ¿Qué hace especial el problema de la calendarización de flujos de trabajo en cómputo en la nube de otros problemas de calendarización, como un administrador de procesos de un sistema operativo?

5. ¿Existe la posibilidad de poder modelar todas las restricciones de la calendarización del flujo de trabajo con las variables costo y tiempo?

Metodología

Enseguida se describen brevemente los pasos que se tienen que llevar a cabo para el desarrollo de la tesis:

Marco teórico

1. Revisión de los calendarizadores de flujos de trabajo y sus algoritmos
2. Revisión de los modelos de cómputo en la nube
3. Revisión de métodos de optimización
4. Revisar si existen simuladores de ejecución de flujos de trabajo sobre entornos en cómputo en la nube

Solución

5. Elaborar un modelo de flujos de trabajos apegado a las pautas propuestas en el trabajo de Wieczorek et al. [10]
6. Buscar un algoritmo de calendarización de flujos de trabajo que sea ampliamente aceptado para ser tomado como punto de comparación con el algoritmo que vayamos a crear
7. Definir un caso de estudio que pueda ser modelado como flujo de trabajo

Experimentación

8. Crear un algoritmo de calendarización de flujos de trabajo inicial, de tal forma que pueda optimizar de manera simultánea para tiempo de ejecución y presupuesto
9. Comprobar mediante simulación que el algoritmo funciona y justificarlo
10. Probar el algoritmo con el flujo de trabajo del caso de estudio y hacer un estudio comparativo con el algoritmo tomado como punto de comparación
11. Investigar mejoras para el algoritmo inicial

Calendario de actividades

En la figura 3 se presenta la planeación de las actividades a realizar para concluir este trabajo. Estas actividades están relacionadas con los puntos establecidos en la metodologia descrita en la sección anterior.

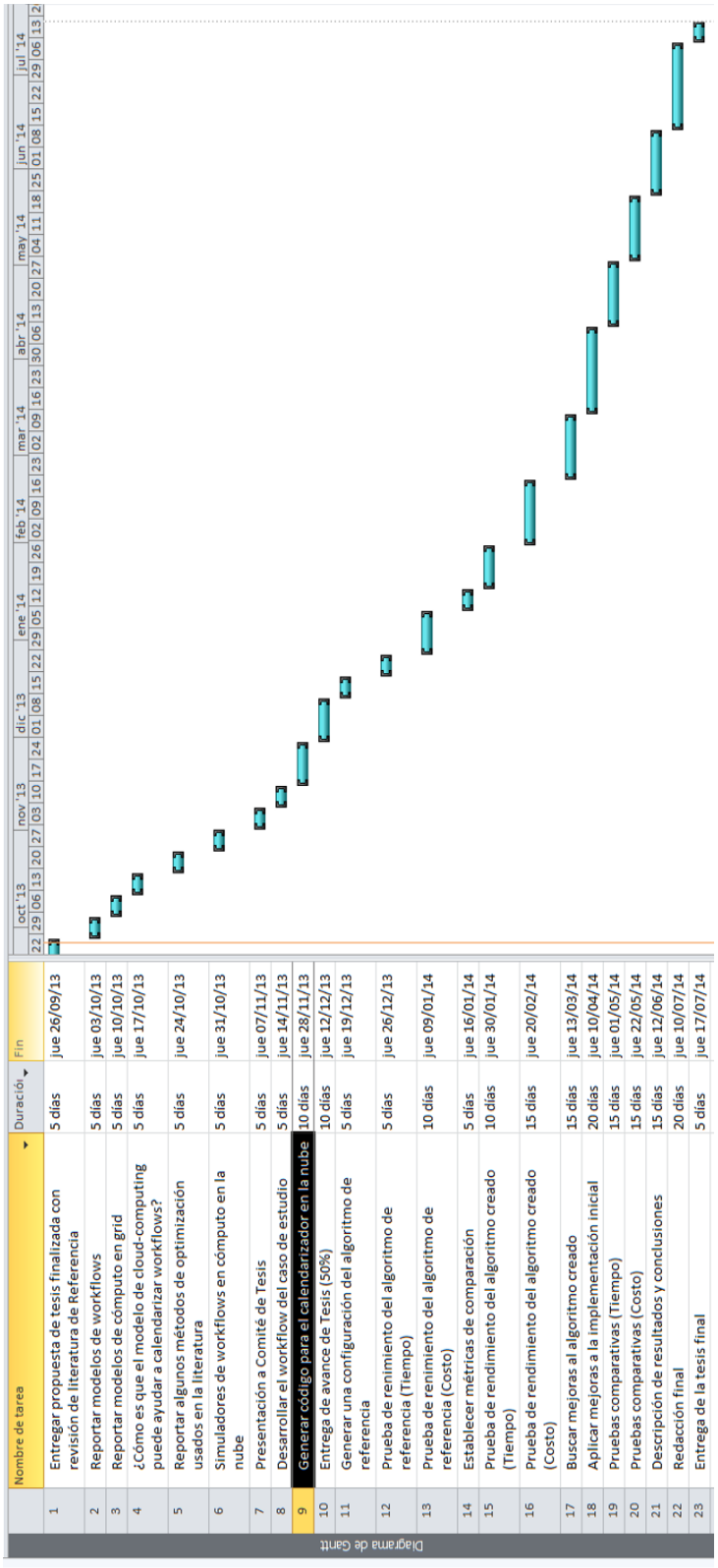

Figura 3: Gráfica de Gantt que muestra la planeación de las actividades a realizar para concluir esta tesis

Planteamiento del problema

De manera muy abstracta, podemos plantear nuestro problema de la siguiente forma: se tiene una funcion f que, dadas una descripción de recursos R y una descripción del flujo de trabajo W, obtiene una estimación del presupuesto necesario para ejecutar el problema, así como el tiempo necesario para su ejecución.

$$(B, T) = f(R, W) \tag{1}$$

Así, nuestra función f es una función vectorial, donde:

$$\begin{aligned} R &= \text{Recursos} \\ W &= \text{Descripción del flujo de trabajo} \\ B &= \text{Presupuesto} \\ T &= \text{Tiempo de ejecución} \end{aligned}$$

También, se tiene una función de calendarización $S(R, W)$, que también depende la descripción de recursos y del flujo de trabajo. Esta función mapea tareas del flujo de trabajo W a los recursos descritos en R. Entonces, hay una relación entre la función de calendarización $S(\cdot)$ y la función de estimación $f(\cdot)$ porque, de acuerdo a la asignación que realice S, la estimación f arrojará un par de números con medidas aproximadas de tiempo y presupuesto.

Ahora bien, note que la función vectorial f es una *función multiobjetivo*. Esto significa que cada componente representa un objetivo diferente. De esta forma, se reescribe la funcion de estimación f de la siguiente forma:

$$(B, T) = (m(R, W), t(R, W)) = f(R, W) \tag{2}$$

Sin embargo, sucede que las funciones que estiman el presupuesto y el tiempo de ejecución, respectivamente, tienen objetivos contrarios. Por ejemplo, si se requiere ejecutar un flujo de trabajo en el menor tiempo posible, entonces se necesitaría un presupuesto muy alto. Por otro lado, si se requiere ejecutar el flujo con la menor cantidad de presupuesto, es muy probable que la ejecución sea muy lenta debido a que los recursos lentos suelen ser más baratos.

Por lo tanto, el problema consiste en optimizar f de tal modo que los valores de B y de T sean los más pequeños posibles.

Subproblemas a resolver

Este problema planteado de optimización sobre la función implica la resolución de otros subproblemas más pequemos que, al unir todas las soluciones, ayudan a construir una nueva solución. Con esta idea en mente, se identificaron subproblemas cuya solución es de vital importancia para este problema de optimización.

En la siguiente lista se enumeran los subproblemas que tienen que ser resueltos para optimizar la función de estimación f.

- **Algoritmos de Calendarización.** Este es el problema más importante a resolver en esta tesis. Ya que como hemos visto anteriormente, la forma en se calendarizan estos recursos con tareas del flujo, determina en gran medida el po
- **Recursos elásticos**
- **Representación del flujo de trabajo**
- **Estimación de tiempo**
- **Estimación de presupuesto**

Consideraciones del planteamiento

A continuación, enumeraremos algunos aspectos del problema que serán simplificados, con el fin de hacer del problema un reto posible:

- Sólo se consideran para esta solución los flujos de trabajo que estén representados por grafos dirigidos acíclicos, sin extensiones.
- El tiempo de ejecución de cada tarea es dado, es decir, no es estimado.
- Se dará preferencia para resolver la calendarización de flujos de trabajo quee requieran de cómputo internsivo.
- Se asumirá que los recursos de cómputo en la nube tienen un perfil de rendimiento constante; es decir, se tiene una cantidad finita de tipos de computadoras (nodos de cómputo) en una nube.
- Se utilizará una cantidad limitada de recursos en la nube, i.e., se estudiarán problemas que sólo requieran como máximo 20 recursos (nodos de cómputo) en la nube.

Propuesta de solución

Si bien se ha planteado el problema de calendarización de flujos de trabajo que se quiere resolver en esta tesis, tambien se plantea una solución a éste, con el siguiente esbozo descrito a continuación:

1. **Cálculo de los recursos.** El objetivo de este paso es conocer las posibles configuraciones de cómputo en la nube disponibles para ejecutar nuestro flujo de trabajo. En otras palabras, queremos generar en este paso un conjunto C que contenga todas las configuraciones disponibles de recursos, a saber, $C = \{R_1, R_2, \ldots, R_k\}$.

2. **Descomposición de tareas.** Con este paso se pretende transformar la descripción del flujo de trabajo W en una representación que involucre un programa secuencial con secciones paralelas y secuenciales, de modo que estas secciones del programa sean fácilmente calendarizables. Ahora bien, esta descomposición de tareas depende de los recursos que tengamos disponibles. Así, tendriamos un conjunto de pares ordenados $D = \{(R_1, W_{d1}), (R_2, W_{d2}), \ldots, (R_k, W_{dk})\}$

3. **Estimación del tiempo total de ejecución.** Una vez que tengamos la representación secuencial de nuestro flujo de trabajo, podemos calcular el tiempo total de ejecución sumando los tiempos de ejecución más grandes de cada sección de la representación secuencial. Cabe aclarar que establecimos en las consideraciones del problema que los tiempos de ejecución de cada tarea son especificados. De esta forma, la solución propuesta no tiene que estimar el tiempo de ejecución de cada tarea. Con esta idea, tendríamos un conjunto de tiempos estimados.

4. **Estimar posibles presupuestos a partir de la estimación del tiempo total de ejecución.**

5. **Buscar mejor solución con un algoritmo de optimización.** Con el espacio de soluciones generados en los pasos anteriores, utilizar un alogritmo de optimización para encontrar la mejor solución.

6. **Devolver la mejor solución encontrada.**

Referencias

[1] Cluster Híbrido de Supercómputo — Xiuhcoatl — Cinvestav. `http://clusterhibrido.cinvestav.mx/`. Consultado el 10 de noviembre de 2013.

[2] Laboratorio de Supercómputo y Visualización en Paralelo. `http://supercomputo.izt.uam.mx/infraestructura/aitzaloa.php`. Consultado el 10 de noviembre de 2013.

[3] The open source grid engine blog: Running a 10,000-node Grid Engine Cluster in Amazon EC2. `http://blogs.scalablelogic.com/2012/11/running-10000-node-grid-engine-cluster.html`. Consultado el 2 de diciembre de 2013.

[4] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009.

[5] J Octavio Gutierrez-Garcia and Kwang Mong Sim. Agent-based cloud workflow execution. *Integrated Computer-Aided Engineering*, 19(1):39–56, 2012.

[6] Ke Liu, Hai Jin, Jinjun Chen, Xiao Liu, Dong Yuan, and Yun Yang. A compromised-time-cost scheduling algorithm in swindew-c for instance-intensive cost-constrained workflows on a cloud computing platform. *International Journal of High Performance Computing Applications*, 24(4):445–456, 2010.

[7] Angela O'Brien, Steven Newhouse, and John Darlington. Mapping of scientific workflow within the e-protein project to distributed resources. In *UK e-Science All Hands Meeting*, pages 404–409, 2004.

[8] Abusayeed Saifullah, Kunal Agrawal, Chenyang Lu, and Christopher Gill. Multi-core real-time scheduling for generalized parallel task models. In *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, pages 217–226. IEEE, 2011.

[9] Ingrid Wagner and Hans Musso. New naturally occurring amino acids. *Angewandte Chemie International Edition in English*, 22(11):816–828, 1983.

[10] Marek Wieczorek, Andreas Hoheisel, and Radu Prodan. Towards a general model of the multi-criteria workflow scheduling on the grid. *Future Generation Computer Systems*, 25(3):237–256, 2009.

[11] Jia Yu, Rajkumar Buyya, and Kotagiri Ramamohanarao. Workflow scheduling algorithms for grid computing. In *Metaheuristics for scheduling in distributed computing environments*, pages 173–214. Springer, 2008.