PLANIFICACIÓN DINÁMICA DE FLUJOS DE TRABAJO EN LA NUBE

Trabajo final Investigación 1

TIMOTHÉE RENE GIVOIS MENDEZ

168914 | tgivoism@itam.mx

Resumen

Un flujo de trabajo es considerado un conjunto de tareas computacionales estructuradas dependientes o independientes que se ejecutan para permitir resolver problemas complejos que son intensivos en datos, computación e instancias para la ciencia y la industria. Computación en la nube ha permitido utilizar bajo demanda, recursos computacionales en la nube basados en la filosofía "agrega lo que necesites y paga lo que uses", haciéndolo más económico, escalable y por lo tanto más utilizado en las áreas científicas y comerciales. Con esta posibilidad para los flujos de trabajo, de poder correr en la nube de manera paralela y distribuida con un tiempo y hardware definidos por el usuario, han necesitado un sistema de manejo, ejecución, planificación y mapeo de las tareas a los recursos. Esta última acción llega a ser un problema NP-completo que ha sido intentado resolver por varios académicos con algoritmos heurísticos y metaheurísticos que encontrasen aproximaciones de combinaciones óptimas de recursos para reducir el tiempo de ejecución y costo de las instancias o permitan ser confiables, redundantes y tolerante a fallos. En este trabajo presentaremos un algoritmo heurístico que tome en cuenta ambos problemas, es decir, que planifique flujos de trabajo con recursos computacionales de la nube de una manera dinámica, que pueda recibir más tareas una vez iniciada la ejecución del flujo de trabajo y que pueda adaptarse a las fallas y retrasos sobre la marcha intentando mantener el tiempo y/o costo debajo de un límite impuesto por el usuario. Para poder hacer esto, se definirá una metodología que permita simular, desplegar y validar una arquitectura y algoritmo genético en la nube que logre planificar tareas y ser tolerante a fallas.

Contenido del documento

Capítulo I

1.1 Antecedentes

Los flujos de trabajo han sido objeto de estudio por la comunidad académica, tanto los flujos de trabajo que están enfocados a negocios [1] [2] [3] como los que están enfocados a la ciencia [4] [5] [6]. Esto es debido a las diferentes aplicaciones que llegan a tener y el impacto que generan. Los flujos de trabajo intensivos en instancias, se enfocan sobre todo en la escalabilidad y por lo mismo son ampliamente usados en las empresas. Entre los ejemplos que tenemos, en [1] podemos verificar como la aplicación de negociación de servicios entre multiagentes puede generar un *framework* que permita simular servicios haciéndolos más escalables. En [2] se presenta una solución de flujos de trabajo para mejorar la integración continua de software en las empresas, intentando dar una retroalimentación más rápida sobre el software con fallas. Por último, RAPTOR [3] es una aplicación en la medicina que presenta una optimización en el flujo de trabajo manual del radiólogo con un flujo de trabajo electrónico que permite automatizar mucho de los procesos manuales a través de bases de datos, servicios web y reconocimiento de imágenes.

Por otra parte, los flujos de trabajo científicos son un poco diferentes pues en su mayoría están enfocados a ser intensivos en datos y/o en computación, normalmente la razón principal de su uso es entender y sacar provecho a los datos con algoritmos específicamente desarrollados para esto [7] que posteriormente ayudarán a validar un experimento o investigación científica. En la física, podemos ver como los flujos de trabajo son utilizados para calcular ondas gravitacionales con LIGO [4]. En [5], Cybershake es utilizado para predecir la probabilidad de riesgos sísmicos en la zona de California. Cluster Flow [6] por su parte, es una herramienta para crear pipelines de bioinformática y hacer análisis de NGS (secuenciación de la siguiente generación).

Para estas concretas aplicaciones de los flujos de trabajo se han desarrollado bastantes investigaciones que permiten hacer que el flujo termine lo más antes posible, consuma menos energía o que no represente un gran costo para el usuario. La estimación, planificación y el monitoreo conjunto a la ejecución de las tareas que componen el flujo son las tres líneas más representativas de investigación en esta área [8]. Al juntar estas líneas, podemos encontrar soluciones óptimas o aproximaciones a estas que permitan ejecutar un flujo de trabajo en la nube bajo una o varias restricciones. Siendo la estimación el primer paso para saber cuánto tiempo/costo/energía nos tomará ejecutar una tarea en un recurso. Por su parte, la planificación nos ayuda, a partir de esta estimación, a encontrar la combinación de recursos subóptima para las restricciones dadas. Finalmente, la ejecución y monitoreo se encarga de revisar que esta planificación se cumpla y llevar políticas para que las mismas lo hagan dentro de las restricciones dadas.

- **Estimación de flujos de trabajo:** Para esta línea se han creado algoritmos que permiten estimar cual es el tiempo de ejecución con diferentes recursos de las tareas del flujo de trabajo. Para esto, se utiliza información que se puede recopilar de la tarea como ser cantidad y tipo de entradas [9], información histórica [10], utilización de recursos por la tarea [11] entre otras para poder dar una aproximación del tiempo de ejecución con cierto recurso específico. Importante notar que muchas veces, los trabajos que generan planificaciones toman en cuenta que estas estimaciones ya las conocen, ya sea por un sistema que lo aproxime o por información que ingresa el usuario [12].
- **Planificación de flujos de trabajo:** Como lo comenta Liu *et al.* la mayoría de los flujos de trabajo se enfocan en optimizar tiempo de ejecución, costo de los recursos y uso de energía [13]. Puesto que planificar un flujo para optimizar ciertos parámetros es una tarea bastante compleja, se han llegado a usar algoritmos que nos dan resultados subóptimos, dentro de estos algoritmos están los heurísticos que por ejemplo, optimizan el costo de ejecución utilizando nubes híbridas (públicas y privadas) [14]. Las optimizaciones heurísticas pueden dar soluciones en tiempos rápidos. Mientras que las optimizaciones metaheurísticos, como los algoritmos genéticos [15], son un poco más lentos pero dan aproximaciones más cercanas al optimo global.

- **Ejecución y monitoreo de flujos de trabajo:** En esta parte, se estudia cuáles deben ser las políticas y la mejor manera de ejecutar y monitorear flujos de trabajo para que estos lleguen a ser confiables y/o más rápidos [16]. Podemos identificar las políticas de prevención como no replicar datos centralizándolos [17] o la creación de tareas duplicadas [18] que permitan hacer un flujo de trabajo confiable. En otros trabajos, se estudia el monitoreo y detección de [19] jerarquizándolas en tres clases: anomalías de infraestructura, aplicación y flujo de trabajo para poder entender el impacto y el método correcto para verificar cuando ocurran.

1.2 Objetivos

Para poder definir el éxito del trabajo, se planteó cuatro objetivos principales que el sistema debe tomar en cuenta. Estos objetivos giran en torno a la planificación dinámica (objetivos uno y cuatro) y la tolerancia a fallos (objetivos dos y tres):

- Planificar una solución subóptima de recursos necesarios para un flujo de trabajo que esté dentro de un tiempo y/o costo de ejecución específicos.
- Monitorear el estado actual de ejecución de flujos de trabajo y aplicar políticas de contingencia de fallos.
- Llevar a cabo políticas necesarias para la prevención de fallos.
- Reajustar la planificación de recursos tomando en cuenta la dinámica del sistema, como por ejemplo nuevas tareas o errores que impliquen retrasos.

1.3 Alcance

El trabajo siguiente presentará un algoritmo heurístico que permita resolver el problema de planificación recursos en la nube para diferentes flujos de trabajo que sean intensivos en datos. Es importante notar que el trabajo no se enfocará en calcular la estimación de las tareas del flujo de trabajo. Se tomará en cuenta que el costo y tiempo de ejecución de las tareas son parámetros ingresados al sistema por el usuario. El algoritmo utilizará esta información para poder hacer una planificación que esté dentro de las restricciones. Así también, se utilizarán los mismos flujos de trabajo científicos que se utilizaron los trabajos más relevantes y recientes de esta línea para poder generar una comparación precisa, los trabajos con los que se hará una comparación son: [5] [14] [20] [21] [22]. Probablemente no podamos optimizar mejor que ellos, sin embargo, se buscará optimizar por debajo de un límite de tiempo y costo que permita hacer que el flujo planificado sea más robusto que los presentados en el trabajo. Es decir, la contribución de este trabajo, será un algoritmo que en teoría puede que no haga una mejor planificación, sin embargo, en la práctica en un ambiente inseguro será el que termine antes o sea más barato para el usuario.

Así también, el sistema se encargará de monitorear el estado actual del flujo de trabajo, revisando cualquier posible evento que pueda acarrear un cambio en la planificación inicial que se hizo. Además, tomara en cuenta políticas de prevención y contingencia de fallos para poder generar un sistema robusto. Para esto, el algoritmo de planificación tomara en cuenta los fallos más comunes y llevara acciones específicas para poder superar cada fallo. El trabajo estudiará de que clase son estos fallos, y cuáles son las mejores maneras de crear un sistema tolerante a estos fallos (aplicando políticas preventivas y paliativas). Para poder tomar en cuenta estos eventos, se utilizará un modelado basado en distribuciones de las frecuencias con que aparecen tomadas en [16]. Al igual que Chen y Deelman [5], no tomaremos en cuenta que estos errores son codependientes y asumiremos que tienen una distribución exponencial que permita evaluar la confianza de los recursos que se utilicen. La lista de eventos que tomaremos en cuenta para sistema será:

1. Nuevos trabajos en el flujo.
2. Retrasos ocasionados por una mala estimación de tiempo de la tarea o por errores inherentes a la tarea.

3. Fallas en las instancias que están corriendo que afecten el tiempo de ejecución planificado.
4. Finalizaciones tempranas de las tareas.

Finalmente, e l sistema de planificación se retroalimentará cada segundo por el monitoreo que se haga, y revisará si la política actual es segura y está dentro de las restricciones. Cuando existan eventos que acarreen un cambio necesario en la planificación, el sistema recalculará la ruta necesaria para cumplir con las restricciones de tiempo y ejecución dadas por el usuario. Posteriormente, se harán los cambios necesarios inmediatamente en la ejecución del flujo de trabajo mediante un sistema que se encargue de ejecución y monitoreo de las tareas actuales.

1.4 Justificación

El uso de los flujos de trabajo en la nube ha llegado en los últimos años a ser una necesidad. Paulatinamente, las tareas computacionales se han convertido más intensivos en software o hardware: ya no es tan fácil tener la arquitectura necesaria para poder soportar estas herramientas. Una computadora de escritorio o personal no llega a soportar la necesidad de las tareas computacionales. Sin embargo, computación en la nube [23] ha permitido tener al alcance de todos desde herramientas computacionales simples a servidores poderosos que se pagan por horas. Con la filosofía de "agrega lo que necesitas y paga lo que consumas", computación en la nube es una de las tecnologías más importantes y de más crecimientos de la industria de TI [24]. Con el apoyo de la virtualización de servidores, computación en la nube puede ofrecer instancias o pedazos de recursos del servidor en un ambiente aislado. Estas instancias se cobran según las especificaciones técnicas que tengan y el tiempo que sean utilizadas [25].

Es así que los flujos de trabajo han ganado relevancia con el surgimiento de computación en la nube: son varias las aplicaciones que se presentan a lo largo de este trabajo, sin embargo, se pueden encontrar muchas más. En la ciencia, no podríamos encontrar un área en la cual no podamos aplicar flujos de trabajo y no generen un impacto positivo, es por esto que algunos autores hablan que la computación ya es una "tercera rama" con la práctica y la teoría puesto que estas áreas utilizan datos que deben ser procesados, analizados, visualizados, por computadoras que implica un estudio de cómo hacerlo de manera eficiente. En la industria, cualquier proceso lógico que implique computación puede ser modelado como un flujo de trabajo [12].

Como expusimos en los antecedentes de este trabajo, si bien el manejo de los flujos de trabajo en la nube nos trae muchas facilidades, también nos añaden algunos problemas. Desde la estimación hasta la planificación y ejecución del flujo de trabajo podemos encontrar problemas que nos aumenten la complejidad de manejo. Una de las preguntas importantes a responder para poder hacer un eficiente manejo es ¿Cuál es la combinación necesaria de instancias que corran el flujo de trabajo en la nube en el menor tiempo y costo posible? Incluso respondiendo esta pregunta (o dando una aproximación subóptima de este dato mínimo) puede llegar a ser bastante ineficiente hacerlo desde una manera estática, es decir, desentender el proceso de planificación y ejecución/monitoreo. Varias recopilaciones de literatura [16] [26] [13] concluyen que uno de los retos a trabajar en esta corriente de investigación es hacer que los algoritmos de planificación sean más dinámicos que puedan recibir tareas en cualquier momento y/o que se integren con el monitoreo para poder tomar decisiones sobre la marcha que hagan que la planificación inicial se cumpla.

Es de aquí de donde nace la razón de este trabajo, de crear un sistema de manejo de flujos de trabajo que pueda planificar, monitorear y replanificar la combinación de recursos. Queremos hacer una planificación que sea dinámica en el sentido que a cualquier eventualidad se pueda replanificar o tomar medidas necesarias que permitan cumplir con las restricciones dadas por el usuario.

1.5 Breve resumen de la metodología

Para poder validar los objetivos del trabajo, la metodología que definimos se resume básicamente en cuatro puntos principales que se presenta a continuación. Es importante tomar en cuenta que en el capítulo III damos una explicación más extensa de estos puntos, dividiéndolos en siete para poder dar los argumentos necesarios que validen cada uno de estos.

1. Definir las restricciones a optimizar y los puntos a tomar en cuenta para la tolerancia a fallos. En esta parte, es importante que definamos cuáles serán las restricciones a optimizar y los eventos que tomaremos en cuenta junto con su impacto en la planificación.
2. Definir una arquitectura y algoritmo heurístico. Ambos serán la solución final que este trabajo presentará como contribución para un sistema de optimización de restricciones tolerante a fallos.
3. Simular la arquitectura y algoritmo heurístico con flujos de trabajo científicos aleatoriamente para poder revisar que estos cumplen con las restricciones.
4. Desplegar y correr la arquitectura y algoritmo heurístico en una nube pública para poder terminar de validar que las restricciones se cumplen.

1.6 Organización del documento

El documento se compone por tres capítulos. El primer capítulo contiene una recopilación breve de los antecedentes, objetivos, alcance y justificación del trabajo y un breve resumen de la metodología que se continuará implementando el proyecto. Posteriormente, tenemos un capítulo de marco teórico en donde nos enfocamos en definir para el lector los flujos de trabajo junto con sus clasificaciones más comunes por la comunidad científica y explicar cuáles son las líneas de investigación más relevantes. Finalmente, presentamos un capítulo de la metodología que se está aplicando para poder validar los objetivos del trabajo.

Capítulo II.

2.1 Marco Teórico

2.1.1 Flujos de trabajo

Se entiende como flujos de trabajo o *workflows* a la secuencia estructurada de tareas que pueden o no ser dependientes entre ellos que permiten resolver un problema. Los flujos de trabajo normalmente se representan como grafos acíclicos dirigidos (DAG). Se entienden los nodos como las tareas de los grafos y las relaciones como las dependencias que estos tienen. Cuando un nodo termina una tarea, pasa automáticamente a la siguiente tarea. En la figura 1 tenemos un ejemplo de un DAG que para este caso es secuencial. Podemos usar un ejemplo simple para entender mejor un flujo de trabajo, digamos que queremos abrir una puerta, la primera tarea debería ser buscar las llaves en mi bolsillo, la segunda tarea debería ser introducir la llave en el cerrojo de la puerta y abrirla y la tercera tarea debería ser cruzar la puerta. Estas tareas necesariamente tienen que ser secuenciales, no podemos introducir la llave si no la tenemos y tampoco podemos cruzar la puerta si no la hemos abierta.

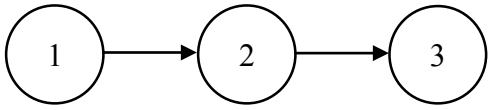

Fig. 1 Grafo acíclico dirigido

Los flujos de trabajo no siempre se presentan como un DAG, en muchos sistemas se espera que el usuario introduzca el flujo como en ese tipo de diagrama, existen algunos trabajos que se hicieron para poder analizar programas y construir DAGs automatizados [27] o para poder refactorizarlos y minimizarlos [28].

Cada tarea puede tener una necesidad específica y estar enfocada a atacar cierto aspecto. Siguiendo el ejemplo anterior, la primera tarea utiliza mi mano y mi bolsillo como recursos, la segunda utiliza mi mano y la puerta y la tercera utiliza mi cuerpo. De esa misma manera, podemos dividir las tareas dentro de los flujos de trabajo según el tipo de recurso o la razón de ser de la tarea. Una de las divisiones más utilizadas por la comunidad es la de flujos de trabajo empresariales (intensivos en instancias) y flujos de trabajo científicos (intensivos en computación o en datos).

Los flujos de trabajo empresariales son en su mayoría intensivos en instancias, normalmente tienen la finalidad de mantener un orden y procesamiento de los datos de la empresa. Usualmente estos flujos son implementados como servicios los cuales se comunican a través de mensajes, por esta razón, son llamados intensivos en instancias puesto que estos servicios normalmente no hacen procesamientos de datos o cálculos computacionales muy complejos, sin embargo, se necesitan varios para poder mantener muchas tareas simples concurrentes que son importantes para una empresa. Dentro de las aplicaciones más citadas sobre estos flujos de trabajo, está el de procesamiento de boletos de tren, en verano y en fechas de temporada alta, estos sistemas llegan a tener una gran demanda de boletos de tren que deben procesar al mismo tiempo. Estas tareas son normalmente fáciles de procesar desde un punto de vista computacional, pero al ser tantas y casi al mismo tiempo, se necesita un proceso sistemático que permita hacerlo sin errores, en la figura 2 se presenta un flujo de trabajo para este caso particular [29].

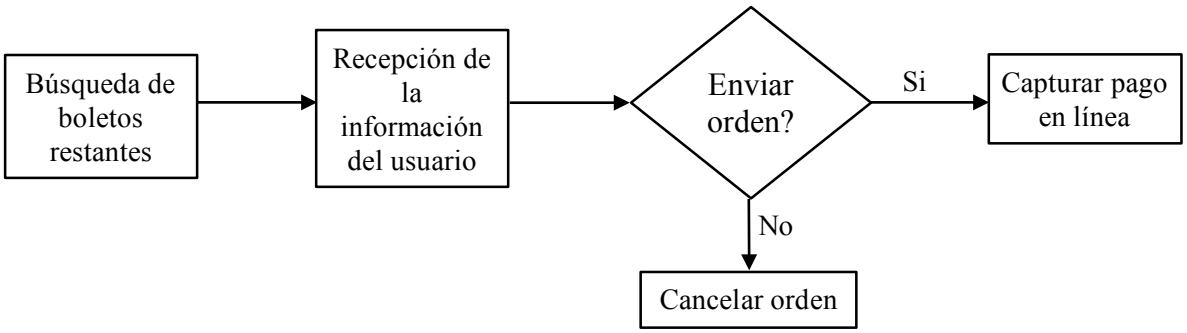

Fig. 2 Flujo de trabajo de un sistema de boletos electrónicos

Los flujos de trabajo científicos son los que más atención llamaron a la comunidad académica, esto es porque los mismos son en su mayoría intensivos en datos y computación y sus aplicaciones pueden llegar a ser mucho más diversas y complejas. Para poder diferenciar los flujos de trabajo intensivos en datos e intensivos en computación, se utiliza la proporción de computación-comunicación (CCR). Los flujos intensivos en datos no llegan a tener un CCR alto, puesto que mucho tiempo lo pasan pasando varios mensajes entre los diferentes flujos y/o la memoria interna de la tarea, mientras que los flujos intensivos en computación se enfocan más en computar resultados sin mucha utilización de mensajes o memoria interna. Un ejemplo de esto puede ser un flujo de un modelo de aprendizaje de máquina que necesita una gran cantidad de datos y procesamiento en la nube [30] comparado con un flujo de trabajo que se utiliza para aproximar métodos numéricos que permiten modelar la cuantificación de incertidumbre [31]. Si bien el primero depende mucho del algoritmo, se puede decir que requiere una mayor cantidad de datos comparado con el segundo ejemplo.

Otra clasificación aceptada por la comunidad científica se basa en la diferenciación de flujos de trabajo de control y de datos [12], esta caracterización no es excluyente. Los flujos de trabajo de control normalmente llegan a ser más pequeños que los de datos y generalmente representan lo que determinado sistema debe

hacer. Sin embargo, los flujos de trabajo de datos llegan a ser más grandes. En la figura 2 por ejemplo, podemos ver un flujo de trabajo que contiene nodos de datos (los rectangulares) y nodos de control (los romboideos).

Los sistemas de flujos de trabajo se encargan en manejar todo el procesamiento del flujo de trabajo. Como procesamiento en este caso podemos entender como la recepción, estimación, planificación, ejecución, monitoreo y finalización del flujo de trabajo. Usualmente, podemos encontrar a estos sistemas de flujo de trabajo casados con computación en la nube, ya que normalmente es la herramienta más utilizada por los sistemas de manejo de flujos de trabajo. Según Poola *et al.* [16], los sistemas de manejo de flujos de trabajo se pueden componer por:

1. *Portal del flujo de trabajo:* Utilizado para poder modelar y definir un flujo de trabajo.
2. *Motor de procesamiento del flujo de trabajo:* En esta podemos tener un *parser* o estimador de tareas, planificador, despachador de tareas, un sistema para el control y manejo de fallos, servicios asignación y negociación.
3. *Interfaz de recursos:* Esta parte es independiente al motor y se encarga de la comunicación del mismo con los recursos.

Dentro del motor de procesamiento, las tres tareas más importantes del flujo de trabajo son la estimación, planificación y despachador de tareas, como mostramos en la imagen 3.

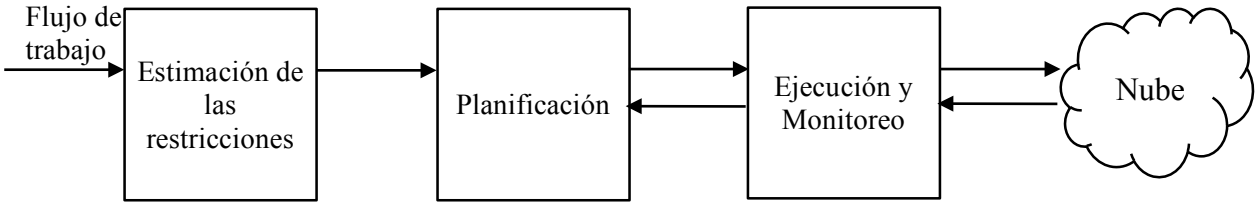

Fig. 3 Arquitectura básica flujo de trabajo.

2.1.2 Estimación de flujos de trabajo

La estimación de flujos de trabajo es la tarea que se encarga de poder calcular el tiempo y/o costo que puede llegar a tener cada tarea que compone el flujo. En muchas ocasiones, no tenemos información del tiempo de ejecución de un flujo de trabajo, para esto, se recurren a distintas estrategias que puedan predecir y estimar el tiempo según información histórica, detalles del flujo o información sobre las entradas. Podemos dividir en dos los tipos de estimaciones, los que lo hacen *a priori* y los que hacen la estimación *in situ.* Normalmente, los trabajos presentan soluciones que mezclan estos dos tipos de estimaciones, sin embargo, podemos notar que predomina una de estas en los trabajos que permite clasificarlos en uno de los dos grupos no excluyentes.

Entre los algoritmos *a priori*, que hacen el cálculo antes de la ejecución, tenemos a [9] que puede predecir el tiempo de ejecución de tareas de un flujo de trabajo a partir de las entradas y el tamaño de la tarea, para poder hacer esto, el autor explica que necesitó caracterizar cinco flujos de trabajo reales tomando información de la utilización hardware y el tiempo para poder crear perfiles que permitan predecir el tiempo de ejecución y utilización del hardware. Posteriormente, el autor en [9] intenta ajustar las estimaciones cuando el flujo de trabajo se corre. En [10] se hace una caracterización de varios flujos de trabajo para generar las predicciones iniciales a partir de sus entradas y se agrega una función de error que va ajustando las predicciones según el flujo de trabajo, esto es para hacerlo más flexible.

Entre los algoritmos que son *in situ* tenemos a [11] que hace el cálculo a partir del rendimiento de la tarea y las características del hardware que están utilizando. En [32] el autor hace un ensamble de estrategias de

predicción del tiempo de ejecución de tareas y toma la predicción que considere más acertada tomando en cuenta la probabilidad que el nodo asignado provea los recursos necesarios para la tarea y pueda experimentar un retraso en su ejecución.

Para los sistemas de planificación, normalmente se necesita un algoritmo que haga estimaciones del tiempo antes de correr un flujo de trabajo en la nube para poder hacer planificaciones más exactas, se puede tomar en cuenta también que el usuario conoce estos valores y los ingresa con las tareas del flujo de trabajo. En concreto para el trabajo que desarrollaremos, no necesitaremos de este sistema puesto que tomaremos en cuenta que esta información ya fue aproximada o entregada por el usuario.

2.1.3 Planificación de flujos de trabajo

Un precepto muy importante del planificador de tareas es que debe mapear las tareas a recursos sin violar las dependencias que existan entre las tareas, es decir, en caso que el flujo de trabajo tenga dependencias, se debe cumplir con las mismas al momento de ser ejecutado para poder generar un resultado aceptable. Las bolsas de trabajo (*Bag-of-tasks*) son tareas que no tienen dependencias entre ellas, esto hace que puedan llegar a ser altamente paralelizables y aplicar diferentes estrategias de planificación [33][34] que normalmente son bastante rápidas. En nuestro caso en particular, nos encargaremos de resolver problemas con tareas que pueden o no tener dependencias entre ellas.

Al tener dependencias, planificar para llegar a un óptimo se considera un trabajo NP-completo [35], lo que hace que se utilicen algoritmos heurísticos o metaheurísticos. Los algoritmos heurísticos son los que aproximan una solución en un tiempo reducido, generalmente, estos puntos son subóptimos: el algoritmo no sabe cuál es el óptimo global, pero puede dar una solución aceptable (dependiendo la definición de aceptable según el problema y las restricciones). Por lo mismo, los algoritmos heurísticos están atados al problema y no se los puede sacar de ese contexto, es decir, son especialistas. Por su parte, los algoritmos metaheurísticos están diseñados para resolver una gran variedad de problemas de optimización difícil, son generales, sin embargo, normalmente no son tan rápidos como los algoritmos específicos. Se llaman "meta" heurísticos porque son de un nivel superior a los algoritmos heurísticos en cuanto a encontrar soluciones que con mayor probabilidad caen en el óptimo global [36]. La aplicación de uno u otro depende de que queramos optimizar, el tiempo que tengamos para optimizar y cuál es la definición de una solución aceptable.

Dentro de los trabajos heurísticos que se han desarrollado en los últimos años, BaRRS [17] presenta una solución con un algoritmo heurístico que divide el flujo en subflujos que posteriormente serán paralelizados, además, intenta aplicar técnicas de reúso de datos que permitan optimizar la cantidad de datos que se transfieren en contextos intensivos de datos. HCOC [14] presenta otro algoritmo heurístico que permite reducir costos utilizando nubes públicas y privadas: la política está en usar recursos potentes de nubes públicas que son personalizables, como Amazon Web Services en las tareas que son más pesadas e importantes, y por otra parte, utilizar la nube privada para tareas que no tengan tanto impacto. IC-PCP [37] por su parte es otro algoritmo heurístico que también busca optimizar tiempo y costo como los dos anteriores, sin embargo, lo hace a partir de una distribución de tiempo máximo entre las tareas y determinando el menor recurso que pueda ser utilizado por la tarea. Fard [38] presenta un algoritmo heurístico multiobjetivo que permite optimizar a partir de más de dos restricciones entre las que tenemos tiempo, costo, consumo de energía y confianza. El algoritmo calcula un rango valido mínimo y máximo para cada restricción. Adicional a esto, se le asigna una importancia a cada restricción para que el algoritmo pueda tomar decisiones de cuál debería ser la solución subóptima.

En la parte de los trabajos metaheurísticos, Yu y Buyya [15] presenta un algoritmo genético que permite planificar optimizando costo y tiempo. Para la optimización de costo, los autores proponen que la función de costo estimule decisiones que sean más baratas y para la optimización de tiempo se usa el algoritmo genético que aproxima la solución con menor tiempo. Por su parte Pandey *et al.* [22] presenta un algoritmo de enjambre de partículas, que llega a ser muy parecido a un algoritmo genético pero que utiliza el comportamiento social de sus partículas para optimizar una solución. Para poder hacer esta optimización,

los autores se enfocan en maximizar los ahorros que se pueden generar al planificar tareas sin precedencia. Aprovechando los beneficios del mercado abierto de proveedores de servicios en la nube, Wu *et al.* [39] presenta un algoritmo que se basa en dos partes: el algoritmo busca entre diferentes proveedores cuales cumplen con las restricciones impuestas por el flujo, para esto, el autor utiliza tres algoritmos metaheurísticos: colonia de hormigas, colonia de partículas y algoritmos genéticos, posteriormente, el autor crea subflujos a los cuales les aplica los mismos algoritmos heurísticos para encontrar una aproximación al óptimo global.

El principal problema de todos estos algoritmos es que no toman en cuenta la probabilidad de que existan errores y fallas que generen tiempos, costos y energía gastada de más en las tareas por tener que repetir tareas, o peor aún, entregar resultados erróneos. El trabajo de investigación que se hará permitirá tomar en cuenta estos problemas haciendo que la penalización por los mismos sea mínima, esto conllevará a que la planificación del sistema sea más real y segura.

2.1.4 Ejecución y Monitoreo de flujos de trabajo

Normalmente, la mayoría de los flujos de trabajo no contienen un sistema de monitoreo actual del flujo de trabajo. Para un sistema de manejo completo, como se presentó anteriormente, es necesario tener una parte que se encargue de la ejecución y el monitoreo del flujo. Esta parte de la arquitectura básica del flujo de trabajo se puede ver como una interfaz, puesto que normalmente se encarga de las siguientes tareas:

- **Comunicarse con el proveedor de la nube para pedir un recurso**: En el caso de tener más de un servidor, el despachador de tareas debe poder saber comunicarse, en su mayoría, se hace por servicios web. Para nubes que son privadas, esta comunicación debe ir directa al manejador de recursos de la nube.
- **Encargarse que la tarea en el recurso tenga lo necesario para ejecutarse:** En el caso de datos, se pueden subir los datos o especificar una base de datos centralizada a la cual la tarea se puede conectar, por ejemplo. En el caso que deba existir comunicación entre tareas, el despachador debe crear las maneras para que estos se puedan comunicar.
- **Monitorear anomalías y comunicárselas al planificador:** Al existir una desviación en el desempeño del flujo de trabajo, se debe comunicar al planificador para tomar las medidas paliativas necesarias.

Para poder detectar anomalías, el sistema debe primero definir qué clase de anomalías son los que afectarían al flujo de trabajo. Así también, estas se deben jerarquizar para saber cuál es el impacto que tienen en las restricciones de optimización. Según una recopilación en PANORAMA [19], el sistema de monitoreo de Pegasus, toma en cuenta dos clases de anomalías:

- **Anomalías de infraestructura:** Estas anomalías son todas las que tienen que ver con todos los problemas que son atribuibles al hardware y/o al proveedor, por ejemplo, bajo desempeño de dispositivos E/S, fallas en el disco, corrupción de datos, mala conexión por paquetes perdidos, firewalls, limitaciones en la red, entre otros.
- **Anomalías de aplicación:** Estas anomalías tienen que ver con todos los problemas que son imputables al programa que corre en los recursos, por ejemplo, errores a nivel aplicación, imposibilidad de acceder/entender datos de entrada, entre otros.
- **Anomalías del flujo de trabajo:** Este tiene que ver con los problemas imputables al sistema del flujo de trabajo, por ejemplo, lentas transferencias de datos entre tareas, poco uso de los recursos, mala planificación/estimación de las tareas, mala configuración, duplicado de tareas, entre otros.

Así también, se pueden tomar medidas preventivas y de contingencia para cada uno de las anomalías en el funcionamiento del flujo de trabajo. Entre las más utilizadas por la comunidad según Poola *et al.* [16] están:

- **Checkpointing:** Es una buena manera de ir guardando el avance de lo que se calcula o lleva del flujo de trabajo en checkpoints en caso de pérdida/corrupción de datos.
- **Duplicado de tareas:** Cuando se aplica esta política como medida de prevención, se espera que las anomalías imputables a la infraestructura no actúen en dos recursos a la vez.
- **Flujos de rescate:** Son normalmente utilizados como medida de contingencia cuando una tarea falla, se continua con el flujo hasta donde se pueda para poder guardar ese estado y continuar una vez que se arregle el problema.
- **Tareas alternas:** Cuando una tarea falla, entonces se usa una tarea alterna definida previamente que sea segura.
- **Tiempos flojos:** Se le agrega un tiempo extra a la tarea para que pueda terminar, pasado este tiempo, se entiende que habrá un problema en el tiempo de ejecución total.

En el trabajo que se desarrollará, se buscará implementar tanto medidas preventivas como paliativas, lo que diferencia a la mayoría de trabajos en esta rama puesto que normalmente sólo utilización uno de estos métodos.

Capítulo III.

3.1 Metodología

La metodología necesaria para validar los objetivos de este trabajo se puede definir de la siguiente manera:

1. Definir que se quiere optimizar en el flujo de trabajo.
2. Definir bajo qué aspectos el sistema será tolerante a fallas.
3. Definir el algoritmo que logra implementar una optimización que pueda ser dinámica.
4. Diagramar una arquitectura que permita optimizar el flujo de trabajo.
5. Pruebas y simulaciones necesarias que permitan validar la arquitectura y el algoritmo o regresar al paso tres.
6. Implementar y desplegar el sistema utilizando un proveedor de nube pública para su posterior evaluación
7. Evaluar y validar que los objetivos se cumplan o regresar al paso tres.

Primeramente, es importante definir que queremos optimizar en el flujo de trabajo. Como comenta el autor en [8], los sistemas de manejo de flujos de trabajo optimizan sobre una restricción o varias. En nuestro caso, la optimización se hará sobre tiempo y costo, que llega a ser el más común de todos los sistemas implementados por su practicidad y aplicación.
Posteriormente, para poder desarrollar un sistema que sea tolerante a fallos, como explica Poola *et al.* [16], debemos definir formalmente cuáles serán los factores que le generen inseguridad al flujo de trabajo. En los alcances hemos definido la serie de eventos que pueden afectar la planificación del sistema. Posteriormente, debemos definir como cuantificaremos la tolerancia a fallos. Para nuestro caso particular, hemos definido que el sistema se considerara tolerante a fallos siempre y cuando cumpla con las restricciones de tiempo y de costo que sean dadas por el usuario.
Una vez llegado al tercer punto, el trabajo se tornará iterativo puesto que el algoritmo heurístico que se defina necesitará de arreglos y modificaciones para poder cumplir con los objetivos. El siguiente punto, de la arquitectura, se definirá como en algunos trabajos sobre el área [5]. Simularemos la arquitectura y algoritmo definido con distintos flujos de trabajo para posteriormente, implementarlo y desplegar en un proveedor de nube pública. Finalmente, se realizarán pruebas con los flujos de trabajo más usados como se

presentó por [17], se los comprara introduciéndoles fallas que permitan demostrar que el sistema que se desarrolló pueda trabajar mejor en ambientes inseguros y cercanos a la realidad.

Referencias

[1] W. Tan, W. Xu, F. Yang, L. Xu, and C. Jiang, "A framework for service enterprise workflow simulation with multi-agents cooperation," *Enterprise Information Systems*, vol. 7, no. 4, pp. 523–542, 2013.

[2] M. A. Shweta, N. John, and S. Shenoy, "Improving Enterprise Build Process Using a Workflow Driven Approach in a Distributed Environment," *Advances in Computing and Communications (ICACC), 2014 Fourth International Conference on*, pp. 214–217, 2014.

[3] J. R. Medverd, N. M. Cross, F. Font, and A. Casertano, "Advanced medical imaging protocol workflow - A flexible electronic solution to optimize process efficiency, care quality and patient safety in the national VA enterprise," *Journal of Digital Imaging*, vol. 26, no. 4, pp. 643–650, 2013.

[4] LIGO Scientific Collaboration *et al.*, "All-sky Search for Periodic Gravitational Waves in the O1 LIGO Data," pp. 1–33, 2017.

[5] E. Deelman *et al.*, "Pegasus, a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015.

[6] P. Ewels, F. Krueger, M. Käller, and S. Andrews, "Cluster Flow: A user-friendly bioinformatics workflow tool," *F1000Research*, vol. 5, p. 2824, 2016.

[7] F. Marozzo, D. Talia, and P. Trunfio, "A Workflow Management System for Scalable Data Mining on Clouds," *IEEE Transactions on Services Computing*, pp. 1–1, 2016.

[8] L. Liu, M. Zhang, Y. Lin, and L. Qin, "A survey on workflow management and scheduling in cloud computing," in *Proceedings - 14th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2014*, 2014, pp. 837–846.

[9] R. F. da Silva, G. Juve, M. Rynge, E. Deelman, and M. Livny, "Online Task Resource Consumption Prediction for Scientific Workflows," *Parallel Processing Letters*, vol. 25, no. 3, p. 1541003, 2015.

[10] A. M. Chirkin, A. S. Z. Belloum, S. V. Kovalchuk, and M. X. Makkes, "Execution Time Estimation for Workflow Scheduling," *2014 9th Workshop on Workflows in Support of Large-Scale Science*, pp. 1–10, 2014.

[11] I. Pietri, G. Juve, E. Deelman, and R. Sakellariou, "A Performance Model to Estimate Execution Time of Scientific Workflows on the Cloud," *2014 9th Workshop on Workflows in Support of Large-Scale Science*, pp. 11–19, 2014.

[12] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528–540, 2009.

[13] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso, "A Survey of Data-Intensive Scientific Workflow Management," *Journal of Grid Computing*, vol. 13, no. 4, pp. 457–493, 2015.

[14] L. Fernando, B. Edmundo, and R. Mauro, "HCOC - A Cost Optimization Algorithm for Workflow Scheduling in Hybrid Clouds," *Journal of Internet Services and Applications*, vol. 2, no. 3, pp. 207–227, 2011.

[15] J. Yu and R. Buyya, "Scheduling Scientific Workflow Applications with Deadline and Budget Constraints using Genetic Algorithms," *Scientific Programming Journal*, vol. 14, no. 3, pp. 217–230, 2006.

[16] D. Poola, M. A. Salehi, K. Ramamohanarao, and R. Buyya, "A Taxonomy and Survey of Fault-Tolerant Workflow Management Systems in Cloud and Distributed Computing Environments," in *Software Architecture for Big Data and the Cloud*, 2017, pp. 285–320.

[17] I. Casas, J. Taheri, R. Ranjan, L. Wang, and A. Y. Zomaya, “A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems,” *Future Generation Computer Systems*, vol. 74, pp. 168–178, 2017.

[18] A. Benoit, M. Hakem, and Y. Robert, “Multi-criteria scheduling of precedence task graphs on heterogeneous platforms,” *Computer Journal*, vol. 53, no. 6, pp. 772–785, 2010.

[19] E. Deelman *et al.*, “PANORAMA: An Approach to Performance Modeling and Diagnosis of Extreme Scale Workows,” *International Journal of High Performance Computing Applications*, vol. 31, no. 1, pp. 4–18, 2015.

[20] N. J. Malawki M., Juve G., Deelman E, “Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds.,” *Future Generation Computer Systems*, vol. 48, pp. 1–18, 2015.

[21] N. Chopra and S. Singh, “Deadline and cost based workflow scheduling in hybrid cloud,” *Proceedings of the 2013 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2013*, pp. 840–846, 2013.

[22] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, “A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments,” in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 400–407.

[23] I. Foster, Y. Zhao, I. Raicu, and S. Lu, “Cloud computing and grid computing 360-degree compared,” *Grid Computing Environments Workshop*, 2008.

[24] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. Ullah Khan, “The rise of ‘big data’ on cloud computing: Review and open research issues,” *Information Systems*, vol. 47, pp. 98–115, 2015.

[25] J. Varia and S. Mathew, “Overview of Amazon Web Services,” *Amazon Web Services*, no. January, p. 22, 2014.

[26] X. Ye, J. Liang, S. Liu, and J. Li, “A Survey on Scheduling Workflows in Cloud Environment,” *Proceedings - 2015 International Conference on Network and Information Systems for Computers, ICNISC 2015*, pp. 344–348, 2015.

[27] L. Virine and J. Mcvean, “Visual Modeling of Business Problems: Workflow and Patterns,” *Proceedings of the 2004 Winter Simulation Conference, 2004.*, pp. 760–765, 2004.

[28] J. Vanhatalo, H. Völzer, F. Leymann, and S. Moser, “Automatic workflow graph refactoring and completion,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, vol. 5364 LNCS, pp. 100–115.

[29] H. Li, S. Ge, and L. Zhang, “A QoS-based scheduling algorithm for instance-intensive workflows in cloud environment,” *26th Chinese Control and Decision Conference, CCDC 2014*, pp. 4094–4099, 2014.

[30] C. Chen, Y. Yan, L. Huang, and X. Dong, “A scalable and productive workflow-based cloud platform for big data analytics,” *Proceedings of 2016 IEEE International Conference on Big Data Analysis, ICBDA 2016*, 2016.

[31] G. Guerra *et al.*, “Uncertainty Quantification in Computational Predictive Models for Fluid Dynamics Using a Workflow Management Engine,” *International Journal for Uncertainty Quantification*, vol. 2, no. 1, pp. 53–71, 2012.

[32] M. Tao, S. Dong, and L. Zhang, “A multi-strategy collaborative prediction model for the runtime of online tasks in computing cluster/grid,” *Cluster Computing*, vol. 14, no. 2, pp. 199–210, 2011.

[33] D. Silva, W. Cirne, and F. Brasileiro, “Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids,” *Euro-Par 2003 Parallel Processing SE - Lecture Notes in Computer Science*, vol. 2790, pp. 169–180, 2003.

[34] J. O. Gutierrez-Garcia and K. M. Sim, “A family of heuristics for agent-based elastic Cloud bag-of-tasks concurrent scheduling,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1682–1699, 2013.

[35] J. J. Durillo and R. Prodan, “Workflow scheduling on federated clouds,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8632 LNCS, pp. 318–329.

[36] I. Boussaïd, J. Lepagnot, and P. Siarry, “A survey on optimization metaheuristics,” *Information Sciences*, vol. 237, pp. 82–117, 2013.

[37] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, “Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds,” *Future Generation Computer Systems*, vol. 29, no. 1, pp. 158–169, 2013.

[38] J. Yu, M. Kirley, and R. Buyya, “Multi-objective Planning for Work ow Execution on Grids,” *Search*, pp. 10–17, 2007.

[39] Z. Wu, X. Liu, Z. Ni, D. Yuan, and Y. Yang, “A market-oriented hierarchical scheduling strategy in cloud workflow systems,” *Journal of Supercomputing*, vol. 63, no. 1, pp. 256–293, 2013.