UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS

LICENCIATURA EN CIENCIAS COMPUTACIONALES

EVALUACIÓN DE ESTRATEGIAS DE CALENDARIZACIÓN BASADAS EN RUTAS CRÍTICAS Y CLUSTERIZACIÓN DE TAREAS COMPUESTAS EN UN AMBIENTE GRID DE DOS NIVELES

TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN CIENCIAS COMPUTACIONALES

Presenta:

CRISTAL AZALIA DENA FLORES

JOSÉ FELICIANO DÍAZ BELTRÁN

ENSENADA, BAJA CALIFORNIA, MÉXICO, MAYO DEL 2011

UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS

Evaluación de Estrategias de Calendarización Basadas en Rutas Críticas y Clusterización de Tareas Compuestas en un Ambiente Grid de Dos Niveles

T E S I S P R O F E S I O N A L

QUE PRESENTA

Cristal Azalia Dena Flores

José Feliciano Díaz Beltrán

APROBADO POR:

M.C. Adán Hirales Carbajal
Presidente del Jurado

Dr. José Ignacio Ascencio López
Secretario

Dr. José Ángel González Fraga
1er. Vocal

Resumen de la tesis de **Cristal Azalia Dena Flores y José Feliciano Díaz Beltrán,** presentada como requisito parcial para obtener el grado de LICENCIADO EN CIENCIAS COMPUTACIONALES. Ensenada, Baja California, México. Marzo del 2011.

EVALUACIÓN DE ESTRATEGIAS DE CALENDARIZACIÓN BASADAS EN RUTAS CRÍTICAS Y CLUSTERIZACIÓN DE TAREAS COMPUESTAS EN UN AMBIENTE GRID DE DOS NIVELES

El presente trabajo se centra en el estudio de dos estrategias para la calendarización de trabajos con relaciones de precedencia, propiamente, estrategias de agrupamiento de tareas y de construcción de rutas críticas estáticas. El propósito de la investigación es evaluar el desempeño de dichas estrategias.

El algoritmo CPOP pertenece a la clase de estrategias de rutas críticas y el algoritmo PCH pertenece a la clase de estrategias de agrupamiento. Ambas estrategias han sido evaluadas sobre contextos de ejecución con multiprocesadores heterogéneos, sin embargo, las cargas de trabajo utilizadas son sintéticas. La presente investigación extiende los resultados presentados en los anteriores trabajos mediante: el uso de cargas de trabajo reales; la utilización de una abstracción de un modelo de ejecución real, propiamente, un Grid computacional compuesto de dos sitios computacionales; y la realización de un diseño experimental de un factor.

La configuración del Grid consiste de dos sitios computacionales con un total de 228 procesadores, el tamaño de los sitios se mantiene relativamente pequeño, con el fin de experimentar alta utilización o tiempo de espera grandes. Una alta utilización podría complicar la colocación de la ruta crítica.

El análisis de datos se realizó en base a tres objetivos, Makespan, Speedup y Tiempos de Espera. A partir del análisis de los datos se demostró que la estrategia PCH tuvo mejor desempeño que CPOP, considerando las tres métricas que se evaluaron, también encontramos que los calendarios locales y las agrupaciones por ruta crítica no ayudan a mejorar los resultados de las estrategias de calendarización para CPOP y PCH.

Resumen aprobado:

M.C. Adán Hirales Carbajal

Palabras clave: Calendarización de tareas, calendarización de flujos de trabajo, Grid computacional.

Agradecimientos

Agradecemos primeramente a Dios por permitirnos culminar esta etapa de nuestras vidas.

Al M.C. Adán Hirales Carbajal, por ser nuestro director de tesis, por ser un gran maestro y apoyarnos a lo largo de este trabajo, muchas gracias.

A nuestros padres por su gran apoyo incondicional, por enseñarnos el valor del estudio, el respeto y a vivir con esmero y dedicación. Esto es por y gracias a ustedes.

Al Dr. José Ángel González Fraga y al Dr. José Ignacio Ascencio López por aceptar ser nuestros sinodales y por todos sus comentarios realizados durante la etapa de revisión de la tesis.

A nuestros familiares y amigos que estuvieron con nosotros dándonos su apoyo y su aliento durante la realización de este trabajo.

Contenido

Lista de Figuras

Lista de Tablas

Capítulo I. Introducción

El presente trabajo se centra en el estudio de dos estrategias para la calendarización de trabajos con relaciones de precedencia, propiamente, estrategias de agrupamiento de tareas y de construcción de rutas críticas estáticas. El propósito de la investigación es evaluar el desempeño de dichas estrategias, empleando un Grid computacional con dos niveles de calendarización como contexto de ejecución.

Las estrategias de calendarización tienen como común denominador la estimación de agrupamiento de tareas, se diferencian por los mecanismos de asignación de recursos y los criterios que buscan optimizar. Las estrategias de agrupamiento emplean políticas de etiquetado para la priorización de tareas, posteriormente, construyen un conjunto de agrupamientos. La asignación de cada agrupamiento puede estar sujeta a una política de asignación, por ejemplo, lo primero es asignar el agrupamiento que contenga la ruta crítica, posteriormente, asignar el agrupamiento con la segunda ruta crítica, esto de manera contínua hasta agotar los agrupamientos. Las estrategias basadas en la identificación de rutas críticas estáticas o dinámicas emplean políticas de etiquetado para la asignación de prioridades. Las tareas con prioridad equivalente son agrupadas y asignadas a un recurso. Tal recurso suele ser reservado exclusivamente para la ejecución de las tareas pertenecientes a la ruta crítica. Posteriormente, las tareas que se encuentren fuera de la ruta crítica son asignadas a un recurso siempre y cuando minimicen algún criterio de optimización y que no extienda la ruta crítica. En este trabajo restringimos el estudio a la evaluación de estrategias que construyen agrupaciones cuya ruta crítica es estática.

Cada estrategia de calendarización requiere de herramientas de propósito específico que brinden soporte a procesos de toma de decisiones, por ejemplo, las estrategias basadas en la identificación de rutas dinámicas requieren de mecanismos de etiquetado, estimación de rutas críticas y predicción de tiempos de ejecución. Las estrategias basadas en la construcción de agrupamientos, requieren de políticas de etiquetado, agrupación y colocación de grupos.

En este trabajo evaluamos el desempeño de las estrategias Ruta Critica en un Procesador (CPOP) y Heurística de Clusterización de Rutas propuestas en (Topcuoglu et al. 2002), (Bittencourt & Madeira, 2006). El algoritmo CPOP pertenece a la clase de estrategias de rutas críticas y el algoritmo PCH pertenece a la clase de estrategias de agrupamiento. Ambas estrategias han sido evaluadas sobre contextos de ejecución con multiprocesadores heterogéneos, sin embargo, las cargas de trabajo utilizadas son sintéticas. La presente investigación extiende los resultados presentados en los anteriores trabajos mediante: el uso de cargas de trabajo reales; la utilización de una abstracción de un modelo de ejecución real, propiamente, un Grid computacional compuesto de dos sitios computacionales; y la realización de un diseño experimental de un factor.

Esta investigación forma parte del proyecto de investigación “Optimización de los recursos y calendarización de tareas en un Grid computacional”. Contribuye con tal trabajo, mediante la implementación y evaluación de estrategias de calendarización en línea, sobre un Grid computacional estático. Esta investigación busca contribuir mediante la generación de estrategias de calendarización, conocimiento empírico y la creación de un calendarizador prototipo.

I.1 Objetivo general

El objetivo de esta investigación es evaluar el desempeño de estrategias de calendarización de tareas con precedencias basadas en agrupamiento, rutas críticas estáticas y políticas de asignación sobre un Grid computacional de dos niveles.

I.1.1 Objetivos específicos:

1. Implementar componentes de estrategias de calendarización basadas en agrupamiento
2. Implementar componentes de estrategias de calendarización basadas en rutas estáticas
3. Evaluar el desempeño de estrategias de calendarización basadas en agrupamiento de tareas
4. Evaluar el desempeño de estrategias de calendarización basadas en rutas críticas estáticas

No es un objetivo específico implementar estrategias de calendarización de flujos de trabajo basadas en políticas de asignación de trabajos paralelos, dado que estas ya se encuentran implementadas en el simulador.

I.2 Alcance de la investigación

Con el objetivo de simplificar el proceso de experimentación, los múltiples factores que se conocen y que inciden en el desempeño de estrategias de calendarización se limitaran a un nivel:

- El número de recursos o procesadores es considerado constante, esto se refiere que no tendrán cambios durante su ejecución.
- La topología de comunicación es completa y no existe latencia de comunicación.
- La calendarización de tareas con precedencias está restringida al meta-calendarizador.
- Los sitios computacionales solo tienen la capacidad de procesar tareas independientes.
- No se permite que los sitios computacionales generen carga de trabajo local.
- Los trabajos asignados por el meta-calendarizador son ejecutados por el sitio computacional hasta su completa terminación, no se permite co-asignación de trabajos, ni migración de tareas.

Aunque las metas contemplan el desarrollo de actividades afines a la ingeniería de software, no es el objetivo de este trabajo desarrollar un diseño arquitectónico, ni presentar modelos de software derivados del análisis y diseño de la infraestructura de simulación. El objetivo del proceso de investigación se centra en el desarrollo de un estudio cualitativo experimental.

I.3 Estructura de la tesis

Esta tesis está estructurada de la siguiente manera, en el capítulo 2 iniciamos con una introducción a los términos que se estarán utilizando durante todo el trabajo. La orientación de tal capítulo está encaminada a la definición de Grid, flujos de trabajo y sus aplicaciones. Concluimos el capítulo presentando el modelo computacional y una breve explicación de las estrategias de calendarización basadas en rutas críticas y clusterización, así como la herramienta de simulación que se utilizó.

En el capítulo 3 se presenta el ambiente de ejecución donde se describe brevemente los componentes del simulador así como las características de implementación de las estrategias. En el capítulo 4 se presenta el análisis de desempeño, el cual tiene como objetivo evaluar la calidad de ejecución de la ruta crítica generada por las estrategias de calendarización. La calidad de la ejecución de la ruta crítica es evaluada en términos de las métricas de aceleración (speedup) y tiempo de espera. Este capítulo está encaminado en la definición del diseño experimental. Los resultados experimentales son presentados y discutidos en el capítulo 5. Finalmente, en el capítulo 6 se muestran las conclusiones que se obtuvieron en esta investigación.

Capítulo II. Marco Teórico

II.1 Introducción

En el desarrollo de este trabajo investigamos múltiples fuentes de información con el objetivo de delimitar el objeto de estudio. Por lo tanto, respondemos a las siguientes preguntas:

1. ¿Qué es un Grid computacional?
2. ¿Qué es un flujo de trabajo? ¿Qué tipo de problemas están siendo modelados con flujos de trabajo?
3. ¿Qué estrategias contemporáneas son utilizadas para la calendarización de flujos de trabajo sobre Grids?
4. ¿Cómo modelamos el problema de calendarización matemáticamente?
5. Finalmente, ¿Qué herramienta utilizaremos para modelar el problema de calendarización abordado en esta investigación?

II.2 Grid computacional

El término Grid Computacional fue acuñado a mitad de los 90's para denotar una propuesta de infraestructura de cómputo que permitiera utilizar recursos computacionales geográficamente distribuidos para ejecutar aplicaciones de alto rendimiento o también llamados *problemas de gran reto*, para las cuales la infraestructura de cómputo instalada (incluso supercomputadoras) no era suficiente (Berman & Hey, 2004),(Foster & Kesselman, 2003),(Foster & Kesselman, 1999). Dentro del tipo de problemas de gran reto encontramos los abordados en la física de

alta energía, meteorología, astronomía, biología, medicina, genética, farmacología y economía (CERN, 2008a).

Se pueden distinguir tres tipos de tendencias en el desarrollo de Grids, *centrados en cómputo*, *centrados en datos*, y *centrados en comunidad* (CERN, 2008a). Los primeros son del dominio del cómputo de alto rendimiento, donde el usuario requiere alta capacidad de procesamiento; la segunda tendencia concierne a problemas que manejan grandes cantidades de datos (del orden de los Petabytes, PB); y la tercera tendencia, también llamada *aplicaciones colaborativas*, intentan reunir comunidades o usuarios para la implementación de estrategias de colaboración sobre medios electrónicos. En esta última clasificación se encuentran principalmente los proyectos @home (California, 2009a) los cuales pueden ser considerados como precursores del Grid, por ejemplo SETI@home (California, 2009b), Rosetta@home (Washington, 2009) y predicción del clima (Oxford, 2009). Los Grids han sido creados principalmente para sufragar las dos primeras tendencias, y en base a esto, podemos diferenciar los Grids Computacionales (C-Grid) y los Grids de Datos (D-Grid).

Un Grid es considerado como una colección distribuida de recursos de cómputo y almacenamiento, mantenidos para servir las necesidades de alguna Organización Virtual (VO) (Foster & Kesselman, 1999),(Foster *et al.*, 2001),(Ranganathan & Foster, 2003). Una organización virtual se define como un conjunto de individuos y/o instituciones afines en sus actividades y que han establecido reglas para compartir sus recursos entre ellos (Foster et al., 2001).

Aún cuando el nacimiento de Grids fue originado por necesidades en el ámbito de la investigación, éstos están siendo adoptados por organizaciones y empresas que buscan aprovechar las ventajas que este tipo de plataformas les ofrecen.

II.2.1 Arquitectura del Grid

Existen algunas variantes en el modelo arquitectónico Grid. La más ampliamente aceptada y que ha sido la base de la mayoría de las contribuciones al Grid es la propuesta por (Foster et al., 2001). El cual presenta al Grid con un diseño arquitectónico en capas. La funcionalidad de cada capa se resume a continuación:

- F*ábrica,* provee recursos cuyo acceso compartido es mediado por protocolos del Grid. Estos recursos pueden clasificarse en físicos y lógicos. Los recursos físicos se definen como dispositivos de hardware, como por ejemplo: procesadores, sistemas de almacenamiento, catálogos, recursos de red y sensores. Los recursos lógicos se definen como servicios o aplicaciones que permiten la colaboración entre sistemas, como por ejemplo: un sistema de archivos distribuido o protocolo de administración de recursos.
- C*onectividad,* la constituyen protocolos de comunicación y autenticación. Los protocolos de comunicación permiten el intercambio de datos entre los recursos de la capa de fábrica. Los protocolos de autenticación permiten verificar la identidad de usuarios y recursos.
- R*ecurso,* la constituyen protocolos que permiten la negociación, iniciación, monitoreo, control, contabilidad, y pago por el uso de recursos individuales compartidos.

- C*olectiva*, a diferencia de la capa de *recurso*, contiene protocolos y servicios que no están asociados con ningún recurso específico sino que más bien son globales. Captura interacciones a través de colecciones de recursos. Algunos ejemplos de estos servicios son: servicios de directorio, co-asignación, calendarización, y servicios de intermediación (brokering), servicios de diagnóstico y monitorización, servicio de replicación de datos y sistemas de programación habilitados para Grid.
- La capa de *aplicación* comprende las aplicaciones de usuario que operan dentro de una VO (por sus siglas en inglés, Virtual Organization).

Del planteamiento de la infraestructura de Grid se desprenden diversas líneas de investigación, las cuales buscan atacar problemas que surgen al intentar compartir recursos geográficamente distribuidos y presentarlos como un recurso unificado. Problemas en seguridad, descubrimiento, agregación, y calidad de servicio son algunos tópicos de interés tratados en investigaciones contemporáneas (Foster & Kesselman, 2003).

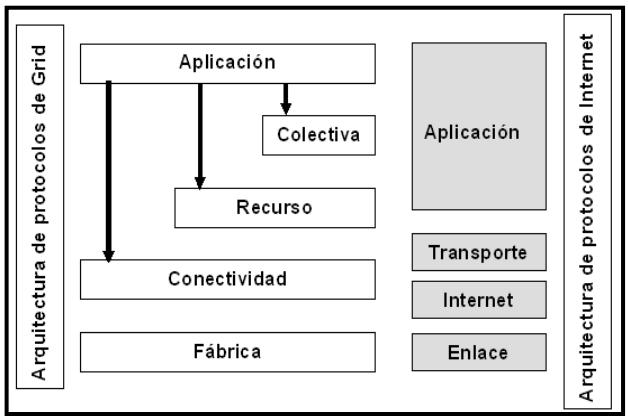

Figura 1.Arquitectura del Grid.

II.2.2 Aplicaciones Grid

Un buen ejemplo de un proyecto que emplea un Grid de datos es el proyecto LHC Computing Grid o LCG. Tal fue implementado con el objetivo de analizar los datos generados por el Gran Colisionador de Hadrones (LHC, por sus siglas en Ingles Large Hadron Collider) (CERN, 2006),(CERN, 2008c). El LHC es un acelerador de partículas que mide casi 27 Km (26,659 metros) de circunferencia y se encuentra dentro de un túnel que está entre 50 y 175 metros bajo tierra.

Se calcula que 3,000 haces de 100 millones de protones viajan continuamente en direcciones opuestas encontrándose cada 25 ns en el lugar exacto donde se encuentran los detectores, produciendo hasta 20 colisiones en cada haz. En operación normal, el LHC produciría alrededor de 15 PB de datos por año, equivalente a una pila de CDs de 20 Km de altura. Para el almacenamiento, procesamiento y análisis de tal cantidad de datos se decidió construir el LCG (CERN, 2009). El LCG une diversas universidades y centros de investigación de muchas partes del mundo y permite distribuir la información generada por LHC a las instituciones participantes.

Otro ejemplo de una aplicación de Grid, es el proyecto EGEE (Enabling Grids for E-sciencE) (EGEE, 2004). Provee una infraestructura de cómputo a más de 13,000 investigadores alrededor del mundo que trabajan en diversas áreas de la ciencia, como la física de alta energía, ciencias de la tierra, ciencias de la vida, astronomía, astrofísica, química computacional, y ciencias de la computación.

Un ejemplo más de un proyecto Grid es el TeraGrid (Teragrid-NSF, 2001), fundado en 2001 por la NSF (National Science Foundation) en Estados Unidos. En él

participan actualmente 11 instituciones conectadas mediante redes de alto rendimiento. TeraGrid integra computadoras de alto rendimiento, recursos y herramientas de datos así como instalaciones experimentales de alto nivel. Los recursos actuales del TeraGrid alcanzan actualmente más de 1 PetaFLOPS de capacidad de cómputo, más de 30 PB de almacenamiento, recursos de visualización y más de 100 bases de datos de disciplinas específicas.

En (CERN, 2008b) y (OGF, 2006) se puede ver una lista completa de los proyectos de Grid en producción alrededor del mundo.

II.3 Flujos de trabajo

En la actualidad, nuevas representaciones para el modelado de trabajos derivados del E-comercio y la E-Ciencia están siendo propuestos, tales son los casos de los flujos de trabajo y los trabajos sujetos a prestaciones de servicio (SLA, Service Level Agreements).(G.C. Fox & D. Gannon, 2006) definen un flujo de trabajo como la automatización de un proceso el cual involucra la administración de un conjunto de servicios, agentes y actores para la solución de un problema u oferta de un nuevo servicio.

II.3.1 Problemas modelados como flujos de trabajos

Actualmente hay una gran variedad de problemas en diferentes disciplinas que están siendo modelados como flujos de trabajo. La evaluación de los flujos de trabajo con frecuencia se realiza mediante una simulación. La simulación es efectiva cuando se trabaja con problemas a gran escala, por ejemplo: en la bioinformática, los investigadores utilizan flujos de trabajo para entender los fundamentos de las enfermedades complejas; en la astronomía, los científicos están utilizando flujos de trabajo para generar mosaicos de grado en la ciencia del cielo, con la finalidad de examinar la estructura de las galaxias, y en general, para conocer la estructura del universo; en la sismología, los flujos de trabajo son utilizados para predecir la magnitud de los temblores dentro de un área geográfica sobre un periodo de tiempo; en la física, los flujos de trabajo se utilizan en la búsqueda de ondas gravitacionales para modelar la estructura de átomos (Juve & Deelman, 2010). Y así como estas, existen muchas más aplicaciones en las que se utilizan flujos de trabajo. Hoy en día, las aplicaciones con flujos de trabajo son ejecutadas en infraestructuras nacionales e internacionales tales como OSG,TeraGrid y EGEE (EGEE:Online).

II.4 Estrategias para la calendarización de flujos de trabajo sobre Grids

Dentro de los problemas principales que surgen en la implementación de un Grid se encuentra la *administración de recursos*. Debido a que el rendimiento del Grid depende principalmente de las estrategias utilizadas para planificar y distribuir trabajos a los diferentes recursos que lo conforman. Entre mejor sea la calidad de la información utilizada para auxiliar los procesos de planificación y distribución de trabajos, mejor será el desempeño del sistema (ej. mejor utilización) o mayor la

calidad del servicio al usuario (ej. menor tiempo de espera) (Ranganathan & Foster, 2003),(Shnizler *et al.*, 2004).

En general, la *administración de recursos* es utilizada para describir todos los aspectos del proceso de localización de recursos o servicios, disponerlos para su uso, utilizarlos y monitorizar su estado (Nabrzyski et al., 2003). Una de las actividades esenciales de la *administración de recursos* es la *calendarización*, la cual refiere la asignación de recursos limitados a tareas, con el objetivo de optimizar una o más criterios de optimización. En otras palabras, la *calendarización* es una planificación de cuándo y dónde se ejecutarán los programas de usuario de tal manera que se optimice algún criterio común (ej. utilización, tiempo de espera). La *calendarización* ha sido estudiada intensamente en las últimas seis décadas, en diferentes áreas como son la administración, la ingeniería industrial, la investigación de operaciones y las ciencias de la computación (Leung, 2004).

La calendarización con restricciones de precedencia de trabajos en un Grid distribuido es compleja. De hecho, en la teoría de calendarización clásica, algunos modelos de problemas con restricciones de precedencia han sido comprobados NP-completos (Leung, 2004) y la complejidad de la calendarización de flujos de trabajo ha aumentado significativamente. Varios factores nuevos como: el consumo de energía, la capacidad limitada de almacenamiento, fracasos, indisponibilidad de recursos y contextos heterogéneos de ejecución, han contribuido a diseñar nuevas formulaciones de los problemas de calendarización.

Existe una extensa literatura referente a las heurísticas para la calendarización de DAG's (o por sus siglas Grafo acíclico dirigido) y flujos de trabajo han sido publicadas. (Topcuoglu et al. 2002) han estudiado el problema $Q_m|prec|C_{max}$. En su estudio, usaron estimaciones de tiempo de ejecución de trabajos basadas en una política que busca minimizar el tiempo de finalización y latencias de comunicación, con el objetivo de identificar procesadores adecuados para la colocación de trabajos. Trabajos sin precedencia o trabajos que forman parte de la ruta crítica utilizan la política de colocación anterior. En diferentes investigaciones se ha estudiado el problema de calendarización Grid $GQ_m|prec|C_{max}$, una variedad de problemas tales como: la calendarización incremental contra total de flujos de trabajo (Wieczorek,2005), calendarización sujeta a variaciones de comunicación y procesamiento (Bittencourt & Madeira, 2006),la calendarización de calidad de servicios (QoS) sujeta a restricciones de tiempo y cotas económicas (Jia,2006), la calendarización de flujos que generan o utilizan grandes volúmenes de información (Ramakrishnan,2007), la calendarización de flujos de trabajo de datos empleando estrategias de clasificación de tareas en base a niveles o etiquetas (Singh,2008), han sido abordadas en estudios existentes durante la última década.

Estudios contemporáneos de calendarización se han diversificado en diferentes líneas de investigación, enfocándose en: el estudio de la estructura de flujos de trabajo con el objetivo de agrupar tareas, tal que la ruta crítica sea minimizada; en la selección de recursos confiables en un ambiente compartido de ejecución; o localización de recursos adecuados para la colocación de flujos de trabajo con altos volúmenes de datos, o cuando las restricciones de QoS están

presentes. Un común denominador en varios estudios es el uso de estimaciones de tiempo de ejecución de tareas, específicamente, la minimización del tiempo de terminación. Tal criterio de selección de recursos se utiliza tanto para la colocación de trabajos independientes o agrupados.

El estudio de estrategias para selección de recursos es muy extenso, sin embargo, la mayoría de los resultados son analíticos, por lo que su desempeño sobre contextos de producción es desconocido. Adicionalmente, los algoritmos propuestos en la literatura clásica de calendarización suelen simplificar el contexto de ejecución. Por lo que es necesario evaluar algunas heurísticas clásicas y contemporáneas con el objetivo de cuantificar su desempeño. En este estudio nos hemos enfocado en dos heurísticas de calendarización de tareas basadas en clusterización y rutas críticas: Ruta Critica en un Procesador (CPOP) y Heurística de Clusterización de Rutas (PCH). Elegimos las anteriores heurísticas considerando como criterio de selección: su buen desempeño reportado en trabajos previos, la frecuencia de trabajos que lo citan, y si la heurística es contemporánea (con al menos una década de antigüedad). Las características principales de las heurísticas que evaluamos son resumidas en párrafos subsecuentes.

II.5 Modelo del problema de calendarización

Estudiamos el problema de calendarización offline (fuera de línea) donde n trabajos del flujo de trabajo $J_1, J_2, \ldots, J_n$ debe ser calendarizado en m recursos (sitios) paralelos $N_1, N_2, \ldots, N_m$. Sea m_i el numero de procesadores idénticos de una maquina paralela N_i. Suponga sin pérdida de generalidad que las máquinas paralelas están ordenadas en orden no descendiente de sus tamaños $m_1 \leq m_2 \leq \cdots \leq m_m$.

II.5.1 Propiedades del trabajo

Dos tipos de trabajos son distinguidos: flujos de trabajo y trabajos independientes. Un flujo de trabajo es una composición de trabajos independientes sujetos a restricciones de precedencia, mientras que un trabajo independiente modela aplicaciones secuenciales o paralelas. Un trabajo con restricciones de precedencia, también llamado flujos de trabajo, puede ser modelado por un grafo acíclico dirigido, $G_j = (V_j, E_j)$, donde V_j es el conjunto de tareas independientes y $E_j = \{(T_u, T_v) \mid T_u, T_v \in V_j, u \neq v,\ no\ hay\ ciclos\ T_u \rightsquigarrow T_v \rightsquigarrow T_u\}$ es el conjunto de aristas entre trabajos en V_j . Cada arista $(T_u, T_v) \in E_j$ representa la restricción de precedencia entre las tareas T_u y T_v, tal que T_u debe finalizar su ejecución previo al inicio de ejecución de T_v. En este estudio los costos de comunicación y dependencias de datos no son considerados.

Cada trabajo J_j es descrito por la tupla $(r_j,\ prec,\ size_j,\ p_j,\ \dot{p}_j)$: donde $r_j \geq 0$ es el tiempo de liberación; $prec \in \{chain,\ tree,\ DAG\}$ es el tipo de trabajo; el tamaño de la tarea se encuentra acotado por $1 \leq max\{size_{j_u}\} \leq m_m$, con $T_u \in V_j$. Donde $max\{size_{j_u}\}$ modela el requerimiento máximo de recursos o el grado de paralelismo; p_j es el tiempo de ejecución; y $\dot{p}_j$ es la estimación de tiempo de ejecución realizada por el propietario del trabajo. Dado que el modelo de calendarización es fuera de línea, el tiempo de liberación $r_j = 0$ es constante.

El tiempo de procesamiento se desconoce hasta que el trabajo ha finalizado su ejecución. p_j modela el tiempo de procesamiento de la ruta crítica dinámica o estática. El costo de la ruta crítica dinámica es relevante cuando comunicación,

heterogeneidad de recursos, contexto poco fiable de ejecución u otros factores que afecten el costo de la ruta crítica son considerados en el problema de calendarización. En este trabajo, el tiempo de ejecución p_j modela los costos de ejecución de la ruta crítica estática.

Usuarios de grandes ordenadores paralelos normalmente deben proporcionar una estimación de tiempo de ejecución $\dot{p}_j$ para los trabajos que someten. Tal es empleada por calendarizadores en sistema de producción para acotar el tiempo de ejecución del trabajo, evitando desperdiciar consumo de recursos. Así, cuando un trabajo llega a su cota, es típicamente abortado. Para nuestro conocimiento, el tiempo estimado de ejecución de un flujo de trabajo normalmente no es dado. Por lo tanto, fijamos $\dot{p}_j$ a p_j para cada $T_v \in V_j$. Finalmente, sea $CP_j = \{T_v \,|T_v \in V_j$ donde T_v pertenece a la ruta critica$\}$ el conjunto de tareas pertenecientes a la ruta critica. Las tareas en CP_j pertenecen a la ruta crítica estática. El costo de la ruta crítica del trabajo J_j es definido como $cpc_j = \sum_{T_k \in CP_j} p_k$.

Cada flujo de trabajo J_j es evaluado en términos de los siguientes criterios: makespan, tiempo de espera, y speedup. Tiempo de espera $tw_j = c_j - cpc_j - r_j$ considera los tiempos de espera de las tareas en la ruta crítica. Speedup $SU_j = \frac{cpc_j + r_j}{c_j}$ (aceleración) evalúa la calidad del tiempo de ejecución de la ruta crítica. Una aceleración de cero indica tiempos de espera nulos para las tareas en la ruta crítica. Mientras que un valor menor a uno, indica que el tiempo de ejecución de la ruta crítica incremento, aumentando así el tiempo de espera de las tareas que pertenecen a la ruta crítica.

II.5.2 El Modelo de calendarización

Denotamos nuestro Grid como GP_m. De acuerdo a la notación de tres campos $(\alpha|\beta|\gamma)$ de Graham (Leung, 2004) empleada para modelar problemas de calendarización teórica, nuestro problema de calendarización es caracterizado como $GP_m|prec|c_{max}$ para el criterio de optimización c_{max}. La notación (α) describe las características de la máquina, (β) describe las características del trabajo, y (γ) la función objetivo. Las funciones objetivo se basan en la evaluación de los criterios descritos anteriormente. Utilizamos la notación MWPS (por su siglas en Ingles, Multiple Workflow Parallel Scheduling) para referirnos a este problema, mientas el término PS (Calendarización Paralela) describe la calendarización de trabajos en una máquina paralela.

El proceso de calendarización Grid puede ser dividido en la parte de etiquetado, la parte de colocación global, y la parte de calendarización local. Por lo tanto, consideramos MWPS como una estrategia de calendarización de tres etapas: MWPS = MPS_Etiquetado + MPS_Asignación + PS. En la primera etapa, tareas de un flujo de trabajo son etiquetadas mediante el uso de una estrategia de etiquetado, tal como: rango superior (UR, Upward Rank), rango inferior (DR, Downward Rank), ó tan tarde como sea posible (LAP, as Late As Possible) (Yu-Kwong Kwok, 1999). Una vez etiquetada una tarea, esta permanece estática. En la segunda etapa, para cada trabajo una máquina adecuada es seleccionada empleando un criterio de selección dado. Solo trabajos independientes son considerados para la colocación. Un trabajo es independiente -listo para ser ejecutado- si no tiene predecesores o sus predecesores han finalizado su ejecución. En la tercera etapa, un algoritmo

independiente PS es aplicado en cada máquina sobre los trabajos colocados durante la etapa previa. La finalización de trabajos con precedencias puede provocar la asignación de trabajos recientemente liberados.

El modelo de calendarización descrito anteriormente es empleado por PWAS. Otras estrategias de calendarización evaluadas en esta investigación, podrían no seguir la misma secuencia de etapas de calendarización. Sin embargo, hemos procurado que en su implementación su diseño original resida.

II.6 Una política basada en estrategias de calendarización de flujos de trabajo

Las estrategias de etiquetado y colocación han sido modeladas como políticas. Las políticas de etiquetado son responsables de estimar las prioridades de de las tareas de todos los flujos de trabajo, considerando cada flujo de trabajo independientemente. Tres estrategias de etiquetado se distinguen: rango superior (UR), estima la longitud de la trayectoria máxima dada una tarea de entrada T_k. La longitud de la trayectoria es la suma de los tiempos de procesamiento de las tareas en la ruta crítica. Puede existir más de una ruta crítica con igual costo; rango inferior (DR) estima la longitud de la trayectoria máxima a partir de una tarea T_k a una tarea terminal; tan tarde como sea posible (AL) estima el tiempo en que una tarea puede aplazar su ejecución, sin incrementar la longitud del calendario. En este trabajo se utilizóó la política UR para el etiquetado de tareas. La calendarización se realiza en orden decreciente del DR, colocando primero las tareas que pertenecen a la ruta crítica (Yu-Kwong Kwok, 1999). Las políticas de colocación son responsables de seleccionar un sitio adecuado para cada tarea. Utilizamos estrategias de colocación

de trabajos paralelos como políticas de colocación de trabajos independientes (Alcaraz, 2010) ,(Tchernykh et al 2008).

(Tchernykh et al 2010) clasifican estrategias de colocación de trabajos paralelos en términos de la cantidad de información que requiere para la toma de decisiones. Se distinguen tres niveles de información. En el primer nivel, una vez que un trabajo ha sido sometido, sus requerimientos de procesamiento son conocidos. No existe información sobre el tiempo de procesamiento de los trabajos, por lo tanto, consideramos una calendarización non-clairvoyant. Sin embargo, puede utilizar la información de trabajos asignados a cada máquina. En el segundo nivel, se tiene acceso a la información del nivel 1, a las estimaciones del tiempo de ejecución de trabajos o a las predicciones del sistema. En el nivel 3, tenemos acceso a toda la información del nivel 2, a todos los calendarizadores locales y estados de los sitios. Note que el número y el tamaño de los sitios son conocidos. En la Tabla I, se listan las estrategias de colocación de trabajos paralelos reportados en (Arun Ramakrishnan, 2007).

Tabla I.
Estrategias Rígidas de Colocación de Trabajos Paralelos

Nivel	Información disponible	Estrategia	Descripción
1	El número y tamaño de los sitios se conocen. No hay información sobre el tiempo de procesamiento de los trabajos, estos están disponibles una vez que se ha sometido un trabajo.	***Rand,*** asigna aleatoriamente trabajos a los sitios disponibles.	
		MLp, selecciona el sitio con la carga mínima del procesador.	$min_{i=1\cdots m}\left(\frac{n_i}{m_i}\right)$
		MPl, selecciona el sitio con la mínima carga paralela. Donde $g(J_k) = M_i$ denota que el trabajo J_k fue asignado al sitio M_i	$min_{i=1\cdots m}\left\{\sum_{g(J_k)=M_i}\frac{size_k}{m_i}\right\}$
		MaxAR, selecciona el sitio con mayor número de recursos disponibles $avail_i$. No se considera requerimientos de trabajos en cola.	$max_{i=1\cdots m}\left(\frac{avail_i}{m_i}\right)$
2	Además de la información en el nivel 1, tenemos acceso a las estimaciones del tiempo de ejecución, $\dot{p}_j$	***MLB***, selecciona el sitio con el trabajo mínimo por procesador en el tiempo r_j.	$min_{i=1\cdots m}\left\{\sum_{g(J_k)=M_i}\frac{size_k \cdot 1}{m_i}\right.$
3	Además de la información en el nivel 2, se tiene acceso a todos los calendarios locales.	***MCT***, selecciona el sitio con el tiempo mínimo de terminación. Donde $C^i_{max} = max_{g(J_k)=M_i}(C^i_k)$, y C^i_k es el tiempo de finalización del trabajo J_k	$min\{C^i_{max}\}$
		MWT, selecciona el sitio con el mínimo tiempo de espera del trabajo. Si n_i es cero n_i se establece en 1	$min\left\{\sum_{g(J_k)=M_i}\frac{C^i_k - r_k - \dot{p}_k}{n_i}\right.$

tareas tal que las restricciones de precedencia sean respetadas. La ejecución de

tareas paralelas es posible, siempre y cuando una tarea predecesora tenga dos o más sucesores y el controlador tenga dos o más recursos.

PCH tiene como objetivo optimizar la colocación de tareas y minimizar costos de comunicación. En este trabajo consideramos costos de comunicación nulos. Evaluamos PCH en términos de la calidad de los calendarios estimados, cuando la política de agrupamiento y colocación de trabajos son empleadas.

El proceso de calendarización realizado por PCH es MWPS = MPS_Etiquetado + MPS_agrupación + MPS_Asignación + PS. Donde MPS_Etiquetado = UR + DR, tal suma genera etiquetas con costo cero para las tareas pertenecientes a la ruta crítica. MPS_agrupación, selecciona las tareas con costo cero, las agrupa y poda el flujo de trabajo removiendo las tareas agrupadas. El proceso MPS_Etiquetado y MPS_agrupación se realiza hasta que el grafo no contenga tareas. MPS_Asignación elige una agrupación y la coloca sobre el recurso que minimice el tiempo absoluto tardío de inicio. La primera agrupación colocada contiene la ruta crítica del flujo de trabajo, agrupaciones posteriores son elegidas arbitrariamente.

Bittencourt en (Bittencourt & Madeira, 2006), evalúa el desempeño de PCH empleando una carga de trabajo sintética compuesta de 15 flujos de trabajo generados aleatoriamente. El tiempo de ejecución de cada tarea se encontraba dentro del intervalo de 10,000 a 310,000 segundos. La información transferida entre tareas se encontraba dentro de 250 a 780 MB. El contexto de ejecución consistía de una arquitectura de calendarización de dos capas, con una topología completa compuesta

de 5 sitios con a lo sumo 7 recursos. Tanto el contexto de ejecución como la carga de trabajo no modelan características de infraestructuras y trabajos contemporáneos.

II.7.2 Estrategia de calendarización basada en rutas críticas (Algoritmo CPOP)

El algoritmo Ruta Crítica en un Procesador (CPOP) es una estrategia para la calendarización de flujos de trabajos sobre multiprocesadores heterogéneos. Sin embargo, ha sido empleado para la administración de recursos en Grid computacionales (Topcuoglu et al. 2002). CPOP agrupa tareas pertenecientes a la ruta crítica y las coloca sobre un recurso que minimice su ejecución. Tareas no pertenecientes a la ruta crítica son colocadas en cualquier otro recurso, siempre y cuando la longitud de la ruta crítica no incremente.

Para obtener las tareas que conforman la ruta crítica y decidir que procesador las atenderá, primero obtenemos la prioridad de cada tarea, la cual es el resultado de sumar el valor de *rango superior* (UR) y *rango inferior* (DR) de cada tarea que conforma el flujo de trabajo. Una vez obtenida la ruta crítica, se asignan las tareas al procesador que minimice el tiempo de ejecución.

CPOP tiene como objetivo calendarizar las tareas críticas sobre recursos heterogéneos; a este recurso lo llamamos procesador de ruta crítica (PRC); si la tarea seleccionada no es crítica, entonces la selección de procesador se basa en un criterio de minimización del tiempo de ejecución de la tarea.

El proceso de calendarización realizado por CPOP es MWPS = MPS_Etiquetado + MPS_agrupación + MPS_Asignación + PS. Donde al igual que PCH, MPS_Etiquetado = UR + DR, tal suma genera etiquetas con costos iguales

para las tareas pertenecientes a la ruta crítica, estos criterios son definidos en la Tabla II. MPS_Agrupación, selecciona las tareas con costo igual y las agrupa. MPS_agrupación se realiza una sola vez, las tareas restantes son tomadas como independientes. MPS_Asignación elige la agrupación de la ruta crítica y la coloca sobre el recurso que minimice el tiempo absoluto tardío de inicio; posteriormente son elegidas las demás tareas como independientes y son colocadas en el recurso que minimice el tiempo absoluto tardío de inicio pero sin aumentar el tamaño del calendario.

Tabla II.
Cálculos de los Atributos del Grafo

Métrica	
Costo computacional, w_i es el costo computacional de la tarea i en el recurso r, donde $Power_r$ es la capacidad de procesamiento de un recurso r, en instrucciones por segundo.	$w_i = \frac{instrucciones_i}{power_r}$
Rango superior. Donden_j es un predecesor de n_i	$rank_u(n_i) = w_i + \max\left(rank_u(n_j)\right)$
Rango inferior	$rank_d(n_i) = max\left(rank_d(n_j)\right) + w_j$
Tiempo Absoluto de Inicio más Temprano, es el tiempo en el que la tarea puede empezar más temprano.	$AEST(n_i) = max\left(rank_d(n_j)\right) + w_j$
Tiempo Absoluto de inicio más Tarde, es el tiempo en el que la tarea puede empezar mas tarde.	$ALST(n_i) = \sum_{T_k \in CP_j} p_k - \left(w_i + \max\left(rank_u(n_j)\right)\right)$

aleatoriamente. Así como flujos de trabajo que fueron derivados de problemas numéricos reales (Eliminación Gaussiana, Transformada de Fourier, y un código molecular dinámico). CPOP fue evaluado empleando las cargas de trabajo sintéticas y reales, los resultados muestran que CPOP tiene un desempeño cercano al de HEFT (Tiempo más temprano de finalización heterogenia) en términos de la longitud del flujo de trabajo y eficiencia. Los cuales a su vez superan el desempeño de algoritmos basados en la calendarización de listas.

II.7.3 Estrategia de calendarización Tiempo más Temprano de Finalización Heterogenia (Algoritmo HEFT)

El algoritmo de Tiempo más Temprano de Finalización Heterogenia (HEFT, por sus siglas en ingles Heterogeneous Erliest Finish Time) es una estrategia de calendarización de aplicaciones para un limitado número de procesadores heterogéneos, el cual tiene dos fases: la fase de etiquetado de tareas que calcula todas las prioridades de las tareas y la fase de asignación que selecciona las tareas en el orden de sus prioridades y calendariza cada tarea seleccionada en el mejor procesador, el cual minimice el tiempo de finalización de la tarea (Topcuoglu et al. 2002).

Para decidir que procesador atenderá las tareas, primero obtenemos la prioridad de cada tarea, la cual tiene como valor el *rango superior* (UR) de cada tarea que conforma el flujo de trabajo. Una vez obtenida la tarea con la mayor prioridad, se asigna al procesador que minimice el tiempo de ejecución.

El algoritmo HEFT busca un espacio vacío apropiado para una tarea n_i en un procesador p_j , la búsqueda comienza cuando n_i esta lista para ser colocada en el procesador. La búsqueda continua hasta que se encuentra el primer espacio vacío y la tarea n_i es colocada.

El proceso de calendarización realizado por HEFT es MWPS = MPS_Etiquetado + MPS_Asignación + PS. MPS_Etiquetado = UR, tal valor genera etiquetas con costos diferentes para las tareas pertenecientes al flujo de trabajo, si existen empates entre tareas, se elige al azar cualquiera de ellas. MPS_Asignación elige la tarea y la coloca sobre el recurso que minimice el tiempo absoluto tardío de

inicio; posteriormente es colocada en el recurso que minimice el tiempo absoluto tardío de inicio pero sin aumentar el tamaño del calendario.

Capítulo III. Ambiente de Simulación

III.1 Herramientas de simulación

Teikoku Grid Scheduling Framework (tGSF) es un simulador basado en estándares de modelado de trabajos y arquitectónicos. En sus versión inicial (1.0) fue empleado para la calendarización de tareas paralelas (Grimme et al., 2009). tGSF es una aplicación codificada en Java desarrollada por el grupo de investigación en arquitecturas de calendarización (GSA-RG, Grid Scheduling Research Group) del foro abierto Grid y por un grupo de investigación de igual nombre dentro del CoreGrid (CoreGrid, 2009). Fue diseñado inicialmente para el intercambio de trabajos paralelos entre sitios en un Grid computacional, emplea políticas de aceptación, distribución y localización de trabajos para lograr balanceo de trabajos. Su diseño arquitectónico es en capas, tales se resumen a continuación:

- *Capa de cimiento,* es un núcleo que administra eventos y el tiempo global. Los eventos son connotados con la unidad de tiempo en que ocurren y son registrados en el núcleo. El ambiente de ejecución permite emular el tiempo real, realizar simulaciones, o depurar los pasos de una calendarización.
- *Capa común,* proporciona una abstracción para el modelado de trabajos, métricas y persistencia. Un trabajo es una agregación de una descripción, ciclo de vida e información procedencia. La descripción está constituida por atributos que caracterizan y vinculan al trabajo con un usuario. Algunos atributos incluyen pertenencia a grupos y requerimientos de procesamiento. El ciclo de vida contiene información que describe los estados actuales y pasados del trabajo. Procedencia almacena la ruta de ejecución del trabajo, tomando

como punto de partida la ubicación –dirección- del sitio que libero el trabajo hasta llegar al sitio que satisfizo los requerimientos de procesamiento del trabajo. Las métricas proporcionan mecanismos para evaluar el desempeño del trabajo. Por último, persistencia proporciona mecanismos para acceder a almacenamiento permanente. Por ejemplo, bases de datos relacionales y archivos de texto.

- *Capa sitio,* administra los recursos de procesamiento de un sitio por medio de un calendarizador abstracto. Las estrategias son usadas por el calendarizador para la colocación de trabajos. El calendarizador puede evaluar múltiples estrategias y seleccionar la más apropiada. Estrategias de calendarización de tareas paralelas incluyen Easy Backfilling y FCFS. Esta capa también proporciona mecanismos que permiten consultar el estado del sitio anfitrión. Información de estado puede ser utilizada por estrategias de calendarización.
- *Capa Grid,* permite intercambio de trabajos entre sitios empleando políticas de aceptación, ubicación y distribución de trabajos. El tomador de decisiones puede ser operado bajo una configuración centralizada o distribuida. Bajo una configuración centralizada, su responsabilidad reside en aceptar trabajos y delegárselos a los sitios locales. En una configuración descentralizada, la delegación de trabajos se realiza mediante una política de distribución.

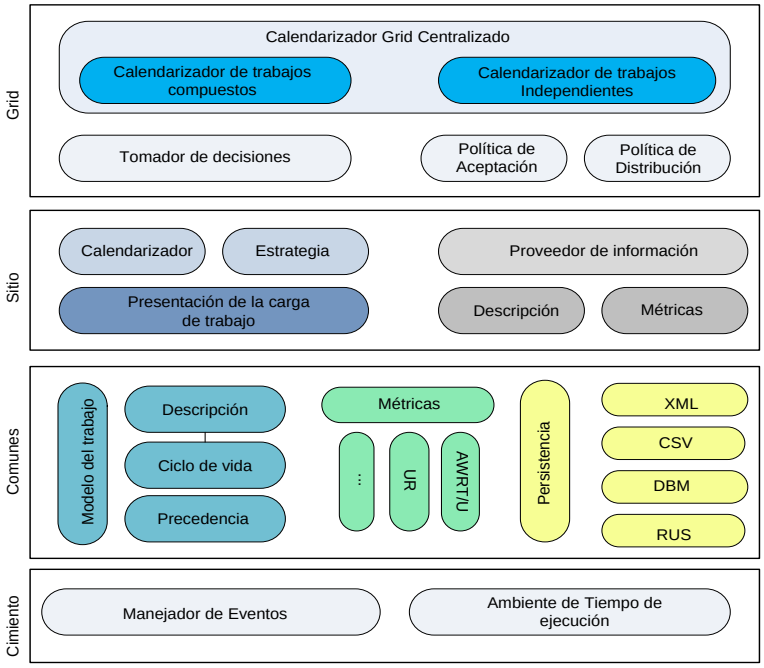

Figura 2.Arquitectura de tGSF.

III.2 Proceso de calendarización

Los calendarizadores a nivel sitio computacional reciben trabajos que arriban eventualmente, y los calendarizan sin tener conocimiento de arribos futuros. Por ende, no es posible generar calendarios óptimos. Por lo tanto, tGSF emplea un modelo de calendarización en línea.

Dado que numerosas estrategias de calendarización de flujos de trabajo comparten funcionalidad, hemos procurado el diseño de una arquitectura de calendarización de bajo acoplamiento. A partir de la revisión de la literatura citada en este trabajo, hemos identificado que las estrategias de calendarización de flujos de trabajo comparten la siguiente funcionalidad:

- **Etiquetado**, asigna una etiqueta o prioridad a cada tarea del flujo de trabajo. Las etiquetas son empleadas para ordenar las tareas, posteriormente, son calendarizadas en orden del etiquetado. Estrategias de etiquetado conocidas incluyen: rango superior (upper-rank), rango inferior (downward-rank) y lo antes posible (as late as posible).

- **Selección de trabajos**, los trabajos son elegidos de acuerdo a un criterio de selección. Trabajos con mayor o menor prioridad son seleccionados primero, pasándolos subsecuentemente a la fase de asignación.

- **Asignación de trabajos**, una función de optimización se emplea para determinar la mejor colocación de un trabajo dado. Las funciones típicas de optimización incluyen: el tiempo más rápido de inicio (earliest start time) y el tiempo absoluto tardío de inicialización (absolute latest start time).

Estas tres etapas han sido utilizadas en múltiples diseños de estrategias estáticas y dinámicas de calendarización de flujos de trabajo. Por ende, tales etapas son modeladas como políticas de etiquetado, selección y asignación de trabajos. El orden u forma en que se emplean las políticas, se define por el mecanismo o estrategia de calendarización.

III.3 Componentes del calendarizador

Hirales (Hirales, 2010), propone una arquitectura extensible y altamente cohesiva para la calendarización de trabajos paralelos y compuestos. La arquitectura de calendarización Grid es una agregación de calendarizadores, estrategias, proveedores de información, y agentes (brokers). Los componentes que intervienen en el proceso de calendarización se resumen a continuación:

- **Broker abstracto Grid,** (Abstract Grid Broker, AGB) recibe trabajos sometidos por los usuarios Grid y los reenvía al calendarizador de trabajos compuestos. Actualmente, tGSF solo permite la definición de un AGB centralizado (CGB). La topología de interconexión entre el CGB y los sitios computacionales es completa.

- **Calendarizador abstracto**, permite la administración de tareas a través de mecanismos concretos. Dos calendarizadores concretos se distinguen: el calendarizador de trabajos paralelos (CTP) y el calendarizador de flujos de trabajo (CFT).
- **Estrategia**, es una interfaz extensible utilizada para definir políticas de colocación de trabajos (ver tabla I).
- **Sistema de Información (SI)**, permite consultar el estado y solicitar estimaciones de tiempo de ejecución de trabajos en sitios computacionales.
- **Distribuidor**, envía trabajos a los destinos elegidos por alguna estrategia de colocación.

La secuencia de eventos para la calendarización de un flujo de trabajo no es única, depende de la lógica de la estrategia de calendarización. Sin embargo, a pesar de las diferencias, Hirales en (Hirales, 2010) propone un modelo flexible de colaboración que permite modelar múltiples estrategias de calendarización de trabajos paralelos y compuestos. La figura 3 ejemplifica un posible flujo de colaboración, tal se describe a continuación.

1. El proceso de calendarización inicia cuando flujos de trabajo son sometidos al CGB a través de un componente de comunicación. Un trabajo es asociado a una sesión. Si la sesión expira, el trabajo es abortado. Note que el CGB es una implementación concreta de AGB.
2. Los trabajos compuestos son almacenados en el buffer de trabajos compuestos (Composite Job Queue, CJQ). El CGB multiplexa el trabajo al CTP o CFT basado en el atributo tipo de trabajo.
3. El CFT administra un flujo de trabajo creando o actualizando una estructura de datos que alberga información de control y estado, tal estructura es denominada Bloque de Control del Trabajo (BCT). BCT incluye el estado del trabajo e información estadística. Posteriormente, una política de etiquetado es empleada para marcar los trabajos. Una o más políticas de selección se emplean para elegir un conjunto de trabajos independientes, los cuales son ordenados de acuerdo a alguna política de ordenamiento ó priorización.
4. Los trabajos independientes son procesados de acuerdo al ordenamiento relativo. Para cada trabajo independiente, CFT selecciona un recurso adecuado

aplicando una o más estrategias de colocación.

5. Una estrategia puede requerir información de los sitios para la toma de decisión. En caso que así sea, el SI solicita estimaciones de tiempo de ejecución o información estado, por ejemplo: número de trabajos en espera, número de trabajos en ejecución, número de recursos disponibles, etc. De acuerdo al objetivo de la estrategia (minimización, maximización o Pareto) un sito destino es seleccionado. Finalmente, la dirección de sitio destino es anexada al BCT.
6. Los trabajos independientes son colocados en la Cola de Trabajos Independientes (CTI), conforme se determina la colocación del trabajo. CFT crea un evento de liberación para cada trabajo en CTI. El evento actúa como un temporizador, el cual indica cuando un trabajo debe ser removido de la cola.
7. En la ocurrencia del evento de liberación, CTF extrae el trabajo correspondiente de CTI. Recupera del BCT la dirección del anfitrión destino y proporciona tanto el trabajo como la dirección destino al distribuidor. El distribuidor tiene la función de migrar el trabajo al anfitrión destino.
8. El distribuidor agrega información de precedencia a los trabajos y los envía a los sitios de destino.

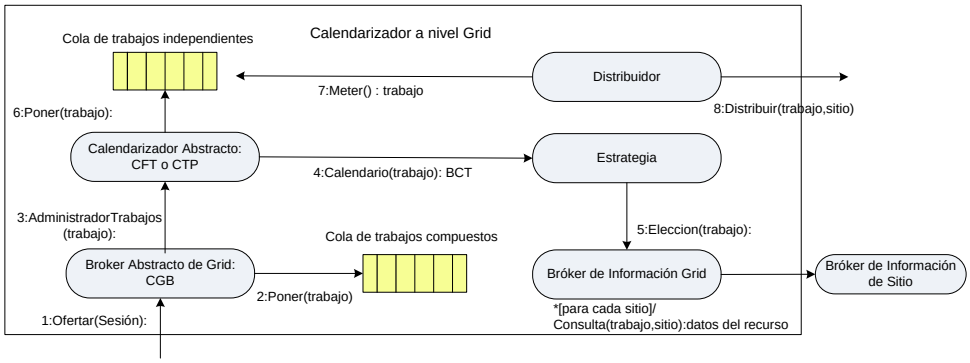

Figura 3.Trabajos paralelos y compuestos del tGSF extendido para soportar calendarización Grid.

En este trabajo extendemos la funcionalidad de Teikoku implementado las estrategias de colocación de trabajo CPOP y PCH. Lo realizamos extendiendo la interfaz estrategia, implementando CPOP y PCH como clases concretas.

Múltiples componentes conforman el modelo de calendarización de tGSF. Esta sección no tiene como objetivo profundizar en los detalles de análisis y diseño de Teikoku. La complejidad del simulador se encuentra fuera del alcance de este trabajo. Se refiere al lector a (Rajkumar., 2006) para detalles de diseño de Teikoku.

III.4 Programación basada en eventos

Los algoritmos de calendarización contemplados en la presente investigación, fueron codificados empleando el paradigma de programación basado en eventos. Tales son empleados en el desarrollo de aplicaciones de interfaces gráficas ó GUI (Graphical User Interface). Una aplicación basada en eventos es muy simular a un autómata de estado finito, el cual está compuesto por estados, eventos y acciones.

En Teikoku los eventos son concentrados en el Administrador de Eventos (AE). Tal componente tiene conocimiento global de todos los eventos que ocurren

durante la simulación y provee sincronización entre componentes mediante la administración del tiempo global. Los componentes interesados en recibir alguna notificación del AE registran los eventos y la información que permita ubicar los componentes destinatario.

PCH y CPOP fueron implementados empleando el paradigma basado en eventos, sus seudo algoritmos se presentan en secciones subsecuentes. Los comentarios en los seudo algoritmos representan eventos, mientras que las líneas posteriores a los comentarios modelan el estado y las acciones ejecutadas por él.

PCH y CPOP transforman el estado de los trabajos. Esto es, pueden modificar la información interna del trabajo. Tal información es almacenada en una estructura de datos denominada Bloque de Control de Trabajo (BCT). Los algoritmos de calendarización de flujos de trabajo no almacenan el estado, esto es que no guardan el estado de trabajos atendidos en el pasado. Toda transformación sobre un trabajo es almacenada en su BCT correspondiente.

Un algoritmo de calendarización inicia cuando el AE notifica la ocurrencia de un evento de liberación de trabajo al calendarizador Grid (CGB). El CGB multiplexa el evento al componente destinatario e identifica el estado apropiado y ejecuta las acciones correspondientes. Posiblemente causando una transición de estado en la tarea y generando un nuevo evento el cual es registrado nuevamente en el AE. Tal flujo de ejecución continúa mientras la tarea no llegue a un estado terminal.

III.4.1 PCH

El algoritmo PCH fue adaptado para un modelo de calendarización offline. La implementación de dicha estrategia requirió de extender la funcionalidad del simulador para que modele flujos de trabajo y tareas independientes. A continuación se describe el pseudo algoritmo de PCH. Dado que la complejidad de tGSF es alta por la gran cantidad de componentes que lo conforman, se omitieron aspectos de diseño e implementación.

Alg.: PCH
Entrada: El trabajo compuesto *j*
Inicio

1. ***//Evento I: En la inicialización, no existen tareas en j calendarizadas***
2. cluster ← {∅}
3. **hacer**
4. *//Fase de etiquetado (MPS_Etiquetado)*
5. Estima rango superior de j y almacena en UR
6. Estima rango inferior de las tareas de j y almacena en AEST
7. $\text{TamañoRutaCritica}_j \leftarrow$ Tamaño de la ruta critica de j
8. **para** cada tareas n ∈ V(j) **hacer**
9. ALST[n] ← $\text{TamañoRutaCritica}_j$ – UR[n]
10. **para** cada tareas n ∈ V(j) **hacer**
11. etiqueta[n] ← ALST [n] – AEST [n]

12. *//Fase de agrupación (MPS_Agrupación)*
13. cluster = {n| etiqueta[n] = 0 y n ∈ CP_j con costo máximo}

14. *//Podado del trabajo compuesto j*
15. $\text{V(j)} = \text{V(j)} - \text{V(cluster)}$
16. $\text{E(j)} = \text{E(j)} - \text{E(cluster)} -$ aristas que tegan un predecesor en la rutacritica

17. *//Identificar un procesador donde puede ser colocado el clúster (MPS_Asignacion)*
18. Encontrar el procesador p que minimice el EFT del cluster. Para cada tarea n ∈ $cluster$sea
19. $\text{p(n)} = \text{p}$ es procesador donde la tarea n será colocada.
20. **mientras** $|\text{V(j)}| \neq 0$

21. *//Colocación de trabajos independientes sobre los procesadores (MPS_Asignación)*
22. I = {n | no tiene predecesores o sus predecesores finalizaron su ejecución}
23. Para cada tarea n ∈ I, colocar n en $p(\text{n})$

24. ***//Evento II: Finaliza la ejecución de una tarea, entonces colocar nuevas tareas independientes***
25. *// (MPS_Asignación)*

26. I = {n | no tiene predecesores o sus predecesores finalizaron su ejecución}
27. Para cada tarea n ∈ I, colocar n en p(n)

Fin

En el seudo algoritmo de PCH se distinguen dos eventos: *inicialización* y *finalización de ejecución de una tarea*. En la inicialización, PCH crea agrupaciones de tareas e identifica que recurso p puede procesarlas mejor. Todas las tarea que pertenecen a una agrupación son designas al recurso p. Cuando un evento de finalización ocurre, nuevas tareas pueden ser liberadas. Sin embargo, la fase de localizar un recurso óptimo se omite dado que durante el evento de inicialización todas las colocaciones fueron predeterminadas. Detallamos las acciones que se desprenden de la ocurrencia de los eventos previamente descritos en los siguientes párrafos.

En el caso del evento de inicialización, PCH clasifica las tareas en agrupaciones iterativamente. Una agrupación se conforma por las tareas que pertenecen a la ruta crítica. El proceso para la creación de agrupación e identificación del recurso adecuado para su colocación se describe a continuación:

1. **Etiquetado de tareas (MPS_Etiquetado)**, las políticas de etiquetado superior e inferior son empleadas para etiquetar al trabajo compuesto. Por ende, una tarea tendrá dos etiquetas. Las tareas pertenecientes a la ruta crítica se distinguen al computar la diferencia entre el costo de la ruta crítica, el rango superior y el rango inferior. Las tareas con costo cero serán las que pertenecen a la ruta crítica. Esto se realiza en las líneas 4 a 11.

2. **Agrupación (MPS_Agrupación)**, las tareas que tienen costo igual a cero son seleccionadas y colocadas en el conjunto clúster. Esto se ilustra en la línea 13.
3. **Podado del trabajo compuesto *j***, durante esta fase del algoritmo, las tareas en clúster junto con todas las aristas que tengan un predecesor en la ruta crítica son removidas de j. El grafo resultante es nombrado grafo residual. Al remover cluster de j , el grafo residual resultante es un bosque. Un vértice tonto con costo de procesamiento cero se aumenta como raíz del bosque, generando un árbol enraizado. Haciendo posible identificar en la siguiente iteración la siguiente ruta crítica. Líneas 15 y 16.
4. **Identificación del recurso óptimo**, el criterio de optimización TFT (Tiempo de Finalización Temprana) es empleado para determinar que recurso minimiza el TFT. Una vez identificado el procesador p que minimice el TFT. Toda tarea en cluster deberá ser colocado en p. Distinguimos $p(n) = p$ el procesador donde las tareas n en cluster deben ser colocadas.
5. En las líneas 22 y 23, solo un evento de liberación es creado por cada tarea independiente en j. La colocación de la tarea n debe realizarse exclusivamente sobre el procesador $p(n)$ (líneas 22 y 23). Las fases 1 a 5 se realizan mientras el grafo residual j tenga vértices.

En el caso de un evento de finalización, las tareas independientes de cada clúster son seleccionadas y enviadas a los recursos predeterminados durante el

evento de inicialización. Recordando que el proceso de envío realmente involucra la creación de un evento de liberación para cada tarea independiente seleccionada. Conforme nuevos eventos de liberación ocurren, nuevas tareas independientes son sometidas a recursos. Este proceso continua hasta que no existen tareas independientes en j.

III.4.2 CPOP

CPOP difiere de PCH en el número de agrupaciones que crea. Mientras que PCH crea tantas agrupaciones como rutas críticas existan en los grafos residuales. CPOP solo crea una sola agrupación. Esta agrupación es colocada sobre el recurso que minimicé el tiempo absoluto de finalización. Las tareas que no pertenecen a la agrupación, son calendarizadas como independientes siempre y cuando se respeten las condiciones y su colocación no incremente el costo de la ruta crítica.

Alg. CPOP
Entrada: El trabajo compuesto j
Inicio

1. ***// Evento I. Identificación de la ruta crítica y su colocación en un procesador crítico***
2. *// Etiquetado* (MPS_Etiquetado)
3. Estima rango superior de j y almacena en UR
4. Estima rango inferior de las tareas de j y almacena en DR

5. *// Calcular prioridad para cada trabajo n en j.*
6. prioridad $[n] \leftarrow$ UR(n) + DR(n)

7. *// Establece la agrupación de la ruta crítica* (MPS_Agrupación)
8. CP_j= {n| prioridad $[n]$ = Valor máximo}

9. *// Identificar el procesador donde la ruta crítica CP_jserá colocador (procesador crítico)*
10. Encontrar el procesador p que minimice el EFT de CP_j. Para cada tarea $n \in CP_j$ sea
11. $p(n) = p$ el procesador donde la tarea n será colocada.
12. *//Colocación de trabajos independientes sobre los procesadores (MPS_Asignación)*
13. I = {n | no tiene predecesores}
14. **para** cada tarea n ∈ V(I) **hacer**
15. **si** $n \in CP_j$ **entonces**
16. Colocar la tarea n en p(n)

17. **sino**
18. Encontrar el procesador p que minimice el EFT de n.
19. Colocar la tarea n en p

20. ***//Evento II: Finaliza la ejecución de una tarea, entonces colocar nuevas tareas independientes***
21. *//Colocación de trabajos independientes sobre los procesadores (MPS_Asignación)*
22. I = {n | no tiene predecesores o sus predecesores finalizaron su ejecución}
23. **para** cada tarea n ∈ V(I) **hacer**
24. **si** $n \in CP_j$ **entonces**
25. Colocar la tarea n en p(n)
26. **sino**
27. Encontrar el procesador p que minimice el EFT de n.
28. Colocar la tarea n en p

Fin

En el seudo algoritmo de CPOP se distinguen dos eventos: *identificación/colocación de ruta crítica* y *finalización de ejecución de una tarea*. En la ocurrencia del primer evento, CPOP agrupa las tareas pertenecientes a la ruta crítica e identifica que recurso p puede procesarlas mejor. El proceso de recalendarización que se desprende de la ocurrencia de un evento de finalización, es similar al realizado por PCH. A excepción de que las tareas que no pertenecen a la ruta crítica emplean MPS_Asignación para seleccionar un recurso adecuado para su colocación. Mientras que las tareas en la ruta crítica debe ser colocadas en el procesador designado. Detallamos las acciones que se desprenden de la ocurrencia de los eventos previamente descritos en los siguientes párrafos.

En el caso del evento de identificación/colocación de la ruta crítica, CPOP crea una sola agrupación e identifica el procesador más adecuado para su colocación, tal procedimiento se describe a continuación:

1. **Etiquetado de tareas (MPS_Etiquetado)**, las políticas de etiquetado superior e inferior son empleadas para etiquetar al trabajo compuesto.

Las tareas pertenecientes a la ruta crítica se distinguen al computar la suma del rango superior e inferior. Las tareas con costo máximo pertenecen a la ruta crítica (líneas 3 a 6).

2. **Agrupación (MPS_Agrupación)**, las tareas que tienen costo máximo son seleccionadas y colocadas en el conjunto CP_j (línea 8).
3. **Identificación del recurso óptimo,** el criterio de optimización TFT (Tiempo de Finalización Temprana) es empleado para determinar que recurso p minimiza el TFT (líneas 10 y 11). Toda tarea en el clúster deberá ser colocado en p.
4. **Colocación de tareas independientes,** todas las tareas independientes pertenecientes a V(I) son colocadas en p. Si no pertenecen a V(I) entonces MPS_Asignación es empleado para seleccionar el procesador optimal p, dode subsecuentemente la tarea es enviada (líneas 10 a 19).

Cuando un evento de finalización ocurre, nuevas tareas pueden ser liberadas. Sin embargo, si las nuevas tareas liberadas pertenecen a V(I) entonces son colocadas en $p(n)$. De lo contrario, MPS_Asignación es empleado para seleccionar el procesador optimo p, donde la tarea independiente es subsecuentemente colocada. El proceso de calendarización continúa hasta que no existan tareas independientes.

III.4.3 Sistema de información

Los sistemas de información local tienen la función de proveer estimaciones de tiempo de ejecución y el estado del sistema computacional. Hasta la versión 1.0, tGSF solo permitía consultar estimaciones de trabajos paralelos. Con la incorporación de flujos de trabajo, fue necesario reescribir el sistema de información para que proporcionara estimaciones de tiempo de ejecución de cadenas de tareas.

La metodología empleada para proporcionar estimaciones de tiempo de ejecución consistía de los siguientes pasos:

- **Clonar el calendario local,** sobre la copia se efectúan en pasos subsecuentes inserciones de tareas prospectas a ser ejecutadas. La duplicación del calendario local es una operación costosa, dado que requiere clonar tareas que están en espera y en ejecución en cada sistema de calendarización local.
- **Colocación de tareas en espera,** en esta fase las tareas que se encuentran en espera son insertadas en el calendario clonado empleando la estrategia de calendarización local, por ejemplo: FIFO o EASY.
- **Colocación de la tarea para la cual se solicita la estimación (tarea prospecta)**, durante esta fase se desprenden una serie de escenarios, enumeramos algunos de ellos a continuación:
    1. Si no existe una tarea prospecta que haya solicitado una estimación en tiempo pasado, entonces se inserta la tarea prospecta empleando la estrategia de colocación local en el calendario clonado local.
    2. Si existe alguna tarea prospecta que solicitó alguna estimación en tiempo pasado y que no ha sido colocada en algún recurso, entonces

se requiere ordenar las tareas prospectas en orden no-descendiente al tiempo de llegada e insertarlas en el calendario clonado local empleando la estrategia de calendarización local.

El cómputo de una estimación de tiempo de ejecución es costoso. Cada que llega una nueva solicitud de estimación de tiempo de ejecución, es necesario la reconstrucción del calendario temporal.

Con la inclusión de trabajos compuestos la metodología para estimar predicciones de tiempo de ejecución cambia, ya que es necesario tratar tareas independientes y cadenas de tareas. Los pasos 1 y 2 de la anterior metodología se preservan, el paso 3 se elimina y se sustituye por las siguientes consideraciones:

- Si una cadena de trabajo A arriba y existe uno o más trabajos independientes, entonces ordenar los trabajos en orden no-descendiente al tiempo de llegada e insertar en el calendario clonado local.
- Si llegaron dos o más cadenas de un trabajo A, entonces realizar un ordenamiento topológico sobre las cadenas de A y reordenar las tareas en la cola de trabajos prospectos. Durante la reordenación todas las tareas de A son insertadas en la ubicación de la cadena con tiempo de llegada mínimo, desplazando tareas que llegaron antes.
- Bajo la presencia de dos o más cadenas de diferentes grafos, se realizan ordenamientos topológicos sobre las cadenas de cada grafo y se colocan las tareas de cada grafo en la ubicación de la cadena que llego primero en la cola de trabajos prospectos.

El costo computacional para computar predicciones de tiempo de ejecución es alto, dado que requiere del uso de algoritmos para el tratamiento de grafos. Específicamente de DFS con orden $O(V + E)$, ordenamiento topológico con orden $O(V + E)$ y Bellman-Ford con orden $O(VE)$.

Capítulo IV. Diseño experimental

El objetivo del análisis de desempeño es evaluar la calidad de la ejecución de la ruta crítica generada por las estrategias de calendarización, específicamente, cuando la calendarización es realizada sobre contextos de ejecución donde la reservación de recursos no es permitida. La calidad de la ejecución de la ruta crítica es evaluada en términos de la métrica speedup. Otras métricas utilizadas en el análisis son definidas posteriormente.

Los factores considerados en el diseño experimental se resumen en la Tabla III. Los factores describen atributos del contexto de ejecución Grid, por ejemplo: las estrategias de calendarización a nivel sitio, las estrategias de calendarización de flujo de trabajo a nivel Grid, el tamaño de la máquina, y la composición de la carga de trabajo. Los niveles corresponden a las alternativas de cada factor.

IV.2 Entorno de simulación

Todos los experimentos se realizaron usando el simulador de calendarización Grid tGSF (Teikoku Grid Scheduling Framework) (Christian Grimme, 2007). tGSF es un simulador basado en estándares que ha sido utilizado para el estudio de problemas de gestión de recursos GRID. Hemos extendido Teikoku para que incluya soporte para la calendarización de trabajos paralelos y con precedencias. Para detalles de diseño de la infraestructura de simulación vea (Hirales, 2010). Las estrategias de calendarización de flujo de trabajo que fueron integradas en tGSF son: HEFT, CPOP y PCH (Bittencourt & Madeira, 2006),(Topcuouglu et al, 2002).Aparte de las estrategias basadas en clusterización y rutas críticas, Hemos

evaluado cinco nuevas estrategias de calendarización de flujos de trabajo basadas en políticas de colocación de trabajos paralelos, propuestas en (Hirales et al. 2010). Todas las estrategias usan una política de etiquetado DR. Las políticas de colocación evaluadas incluyen: DR+MCT, DR+MLB, DR+MLp, DR+MPL, y DR+MaxAR.

Tabla III.
Diseño Experimental

Factores	Niveles
Estrategia de calendarización por sitio	Backfilling
Estrategia de calendarización de flujo de trabajo en Grid	HEFT, CPOP, PCH, DR+MaxAR, DR+MCT, DR+MLB, DR+MLp, and DR+MPL
Tipo de Carga de trabajo	Flujos de trabajo
Número de sitios computacionales	2
Número de recursos por sitio	228 procesadores en total. Sitio uno tiene 100 y dos tiene 128 procesadores
Variables de respuesta	Tiempo promedio de espera, longitud del calendario y la aceleración
Replicaciones	30 repeticiones
Diseño	Factor único
Unidad experimental	Estación de trabajo
Interacción	Ninguna
Modelo del problema	Offline (determinístico)

IV.3 Composición de cargas de trabajo

Los flujos de trabajo Genoma y Cybershake fueron utilizados para preparar las cargas de trabajo (S. Bharathi, 2008). Estos flujos son de acceso público a través del portal del proyecto Pegasus (Pegasus, 2010). Con el fin de utilizarlos en el entorno de simulación, los flujos de trabajo se modificaron de su formato original DAX a un formato extendido SWF (Standard Workload Format) (Feitelson., 2010). El formato extendido SWF incluye cuatro campos que permiten el modelado de restricciones de precedencia, propiamente: un identificador de flujo de trabajo, el tipo de flujo de trabajo, un conjunto de identificadores de trabajos predecesores y un conjunto de identificadores de trabajos sucesores (véase apéndice A).

Una carga de trabajo se compone de 40 flujos de trabajo con 50 tareas cada uno, para un total de 2000 tareas. Dos clases de flujos de trabajo se utilizaron para construir la carga de trabajo de base: 20 flujos de trabajo del Genoma y 20 flujos de trabajo Cybershake. Puesto que una permutación de la carga de trabajo base puede generar resultados diferentes cuando se calendariza, 30 permutaciones aleatorias de la carga de trabajo base se utilizaron para 30 experimentos.

IV.4 Configuración del Ambiente de ejecución

La configuración del Grid consiste de dos sitios computacionales con un total de 228 procesadores, el tamaño de los sitios es definido en la tabla III. El tamaño de los sitios se mantiene relativamente pequeño, con el fin de experimentar alta utilización o tiempo de espera grandes. Una alta utilización podría complicar la colocación de la ruta crítica.

Con el objetivo de delimitar el análisis, se mantienen constantes los factores que se conocen que afectan el desempeño de las estrategias de calendarización Grid (mirar tabla II); Otras restricciones incluyen las siguientes: los flujos de trabajo son etiquetados por el calendarizador Grid; el calendarizador Grid asigna trabajos a sitios computacionales. Los sitios computacionales solo ejecutan tareas independientes. No se permite que los sitios computacionales generen trabajos.

IV.5 Procedimiento

Treinta experimentos fueron ejecutados en una estación de trabajo. Los datos fueron recolectados y analizados mediante la estimación de medidas de tendencia central de las variables dependientes (Arun Ramakrishnan, 2007).

Un buen algoritmo de calendarización debe distribuir de manera adecuada los trabajos para lograr un alto rendimiento del GRID, al tiempo que requiere satisfacer las demandas de varios usuarios de manera equitativa. Comúnmente, los proveedores de recursos y los usuarios tienen diferentes objetivos de desempeño y a menudo presentan conflictos. Estos conflictos van desde minimizar el tiempo de respuesta hasta optimizar la utilización de los recursos. El gestor de recursos GRID puede emplear múltiples criterios en apoyo a la toma de decisión. La metodología multi-criterio basada en la optimización de Pareto puede ser aplicada. Sin embargo, es muy difícil estimar rápidamente las soluciones que requiere el gestor Grid usando dominancia de Pareto. El problema a menudo es simplificado transformando el problema multi-objetivo a un problema mono-objetivo. Sin embargo, la pregunta que surge es ¿Cómo cuantificamos la preferencia de un objetivo? Existen múltiples maneras de modelar preferencias, por ejemplo, pueden ser dadas explícitamente por

los usuarios, expresando la importancia de cada criterio o la importancia relativa entre los criterios.

Con el objetivo de proporcionar un criterio para la elección de la mejor estrategia, se aplicó la metodología propuesta por (Alcaraz et al., 2010). Ellos Utilizan un enfoque para el análisis multi-criterio asumiendo la misma importancia de cada métrica.

El análisis se lleva acabo de la siguiente manera. Primero, evaluamos la degradación en desempeño de cada estrategia para cada uno de los tres parámetros (Tabla II). Esto se hace en relación con la mejor estrategia para cada métrica, como se muestra a continuación: 100* estrategia_metrica/mejor_metrica-100. Por lo tanto, cada estrategia se caracteriza ahora por tres números, reflejando su degradación de rendimiento. En el segundo paso, promediamos estos tres valores (asumiendo la igual importancia de métricas), y estimamos el rango de cada estrategia. La mejor estrategia, con grado de degradación bajo, tendrá el rango 1, la peor estrategia tendrá mayor rango. Note que intentamos identificar qué estrategias tienen un buen desempeño bajo diversos escenarios; Esto es, tratamos de encontrar un acuerdo que considera todos nuestros casos de prueba.

Tabla IV.
Métricas

Métrica	
Makespan	$C_{max} = \max_{j=1,\cdots,n}\{c_j\}$
Tiempo Promedio de Espera	$\bar{t}_w = \frac{1}{n}\sum_{j=1}^{n} tw_j$
Promedio SpeedUp	$\overline{SU} = \frac{1}{n}\sum_{j=1}^{n} SU_j$

Capítulo V. Resultados

En la figura 4 se muestra la media y la desviación estándar del makespan. Las variaciones de los makespans de la gran media son mostradas en la Figura 5. El valor cero en el eje *y* representa la gran media. El promedio del makespan (la gran media) es 428879 segundos. DR+MLB, DR+MPL, DR+MLp, DR+MaxAR y PCH produjeron estos resultados que tienen 366340, 366311, 366183, 365800, y 247126 segundos por debajo de la gran media. Las estrategias DR+MCT, HEFT, DR+MTA, DR_MWT y CPOP muestran peores resultados con 131483, 153166, 394612, 494879 y 537620 segundos sobre la gran media. DR+MLB se muestra que es la mejor estrategia.

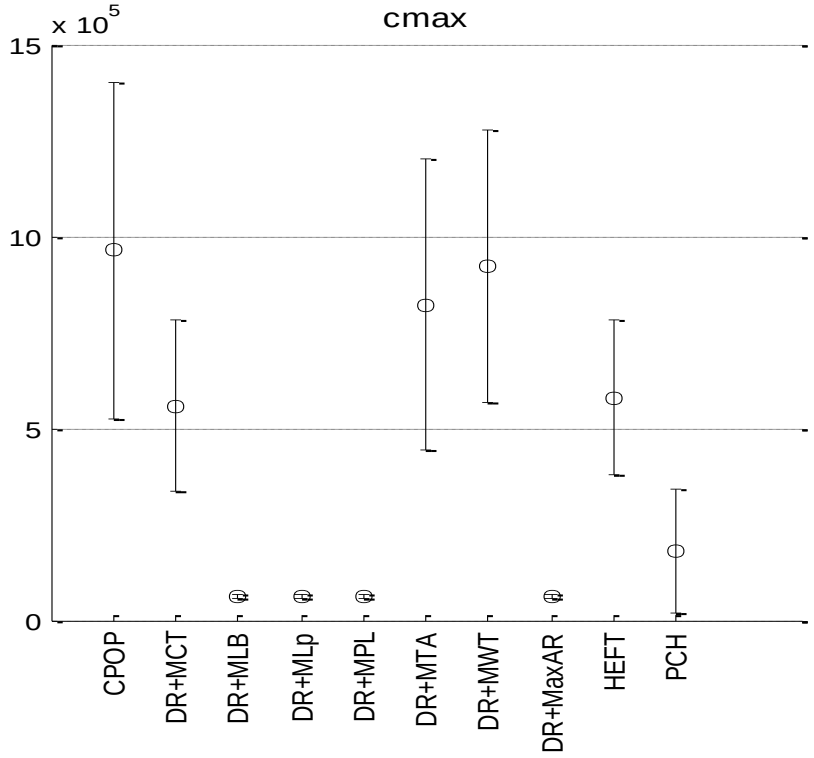

Figura 4. La media y la desviación estándar de Makespan.

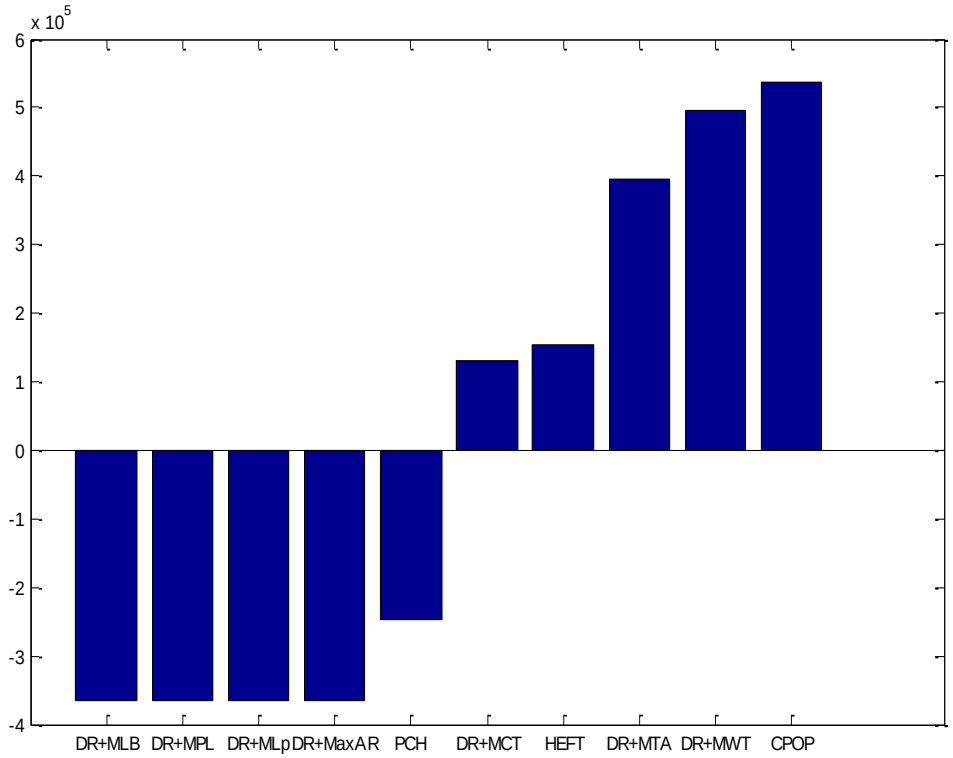

Figura 5. Efecto de makespan sobre la gran media

El porcentaje de variación explicada por el factor de C_{max} es de 70%. El porcentaje restante se deben a errores experimentales, que en este caso se pueden atribuirse a la calidad de la información empleada por las estrategias MPS_Asignación.

La figura 6 se muestra la media y desviación estándar del tiempo de espera del flujo de trabajo. Las estrategias DR+MaxAR, DR+MLB, DR+MLp, DR+MPL tienen el tiempo promedio de espera cercano a cero. Las estrategias CPOP, DR+MCT, DR+MTA, DR+MWT, HEFT y PCH producen resultados con un tiempo de espera significativamente alto.

CPOP y PCH emplean estimaciones de tiempo de ejecución para determinar el tiempo de finalización temprana (TFT) en cada sitio computacional. Sin embargo, las estimaciones o predicciones de tiempo de ejecución normalmente son incorrectas. Esto se debe a que se emplea el tiempo de ejecución proporcionado por el usuario para el cómputo de la estimación de una tarea dada. Normalmente, usuarios en sistemas de producción proporcionan estimaciones muy grandes en comparación con los tiempos reales de ejecución.

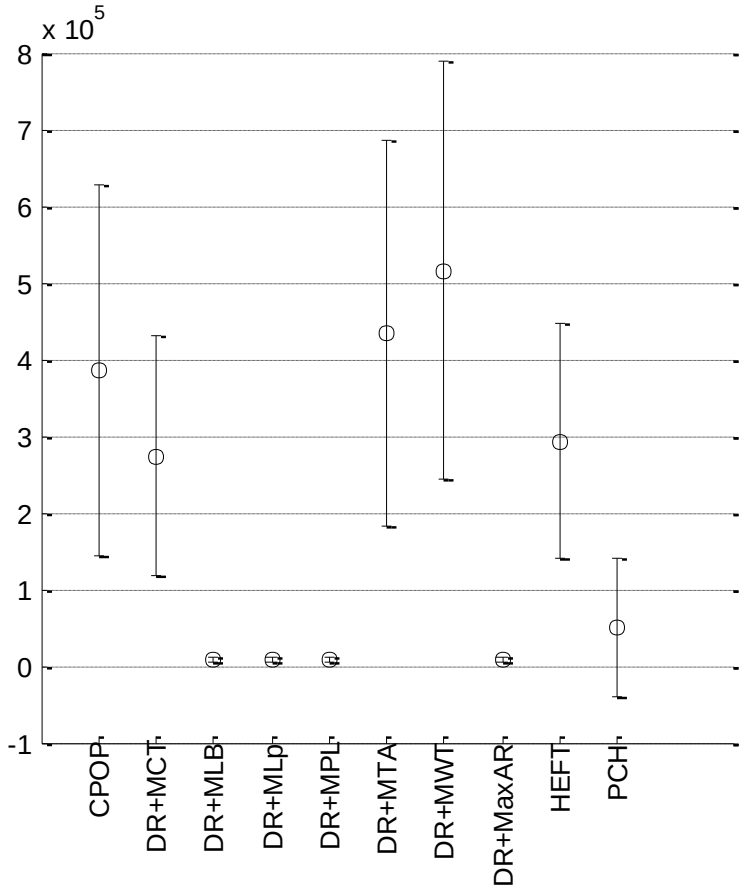

Figura 6. Media y desviación estándar del tiempo de espera.

Una política de asignación de tareas basada en estimaciones de tiempo de ejecución, utiliza estimaciones proporcionadas por el usuario $\dot{p}_j$ para determinar el tiempo de liberación de la tarea que atiende. Al ser incorrecta la estimación, el tiempo de liberación de la tarea a colocar será incorrecto. Ahora, si consideramos una cadena de tareas donde cada tarea tiene una estimación incorrecta, los tiempos de liberación asignados por la heurística de colocación serán incorrectos, este problema ocurre cuando fijamos p_j a $\dot{p}_j$.

La comparación de la gran media de las estrategias de asignación de flujos de trabajo se muestra en la Figura 7. El tiempo promedio de espera (gran media) es 199997 segundos. DR + { MLp, MLB, MPL, MaxAR } y PCH tienen un tiempo promedio de espera de 190582, 190558, 190119, 190055 y 148250 segundos por debajo de la gran media. Los tiempos de espera de DR+MCT, HEFT, CPOP, DR+MTA y DR_MWT son 75375, 94797, 187489, 235313 y 316590 segundos sobre la gran media.

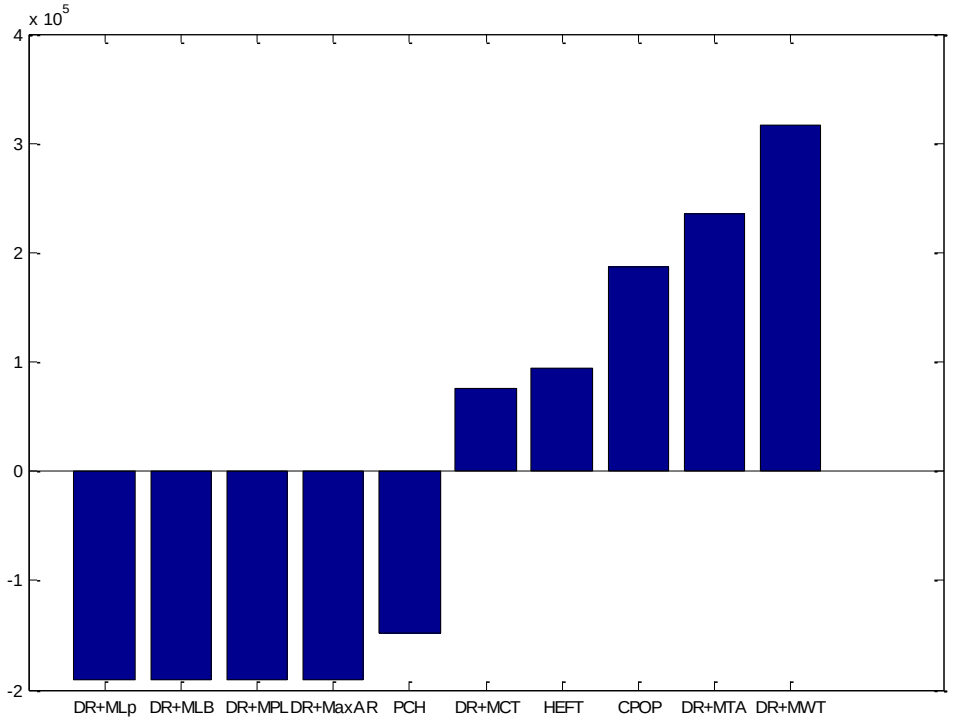

Figura 7. Efectos del tiempo de espera sobre la gran media.

La Figura 8 muestra la media y la desviación estándar del speedup del flujo de trabajo. Las estrategias DR + {MLB, MLP, MPL y MaxAR} tienen un speedup promedio muy cercano a 0.40. PCH también muestra un speedup cercano a 0.25. Todas las demás estrategias tienen un speedup cercano a cero.

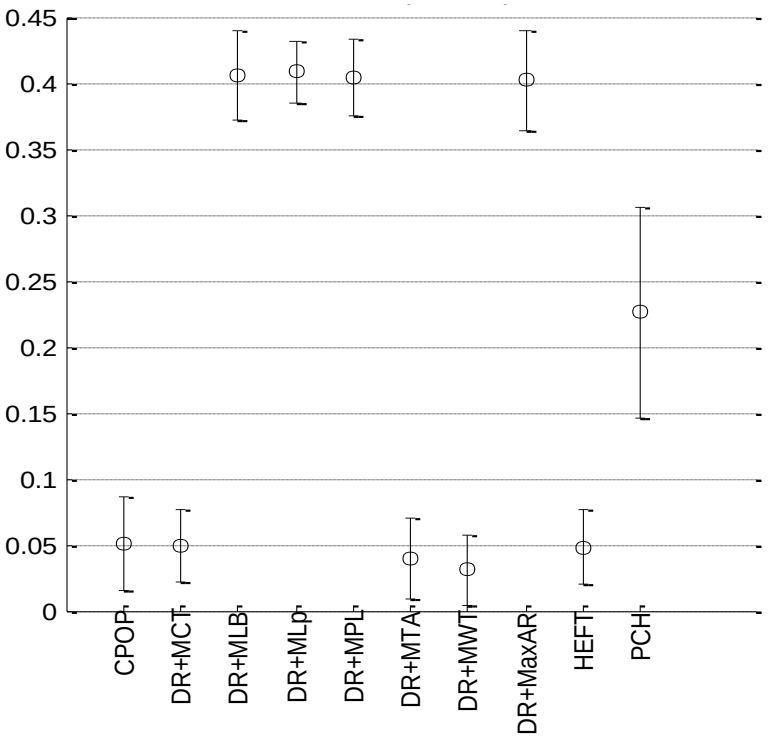

Figura 8. Media y desviación estándar de Speedup.

Recapitulando, una aceleración cercana a uno indica que no ocurrieron tiempos de espera durante la ejecución del flujo de trabajo. La aceleración máxima fue de aproximadamente 0.40, lo cual indica que ocurrieron tiempos de espera en las tareas pertenecientes a la ruta crítica. Las estrategias que lograron apenas una aceleración de .05 experimentaron tiempos de espera muy grandes. Estas observaciones se confirman en los resultados de las figuras 6 y 7.

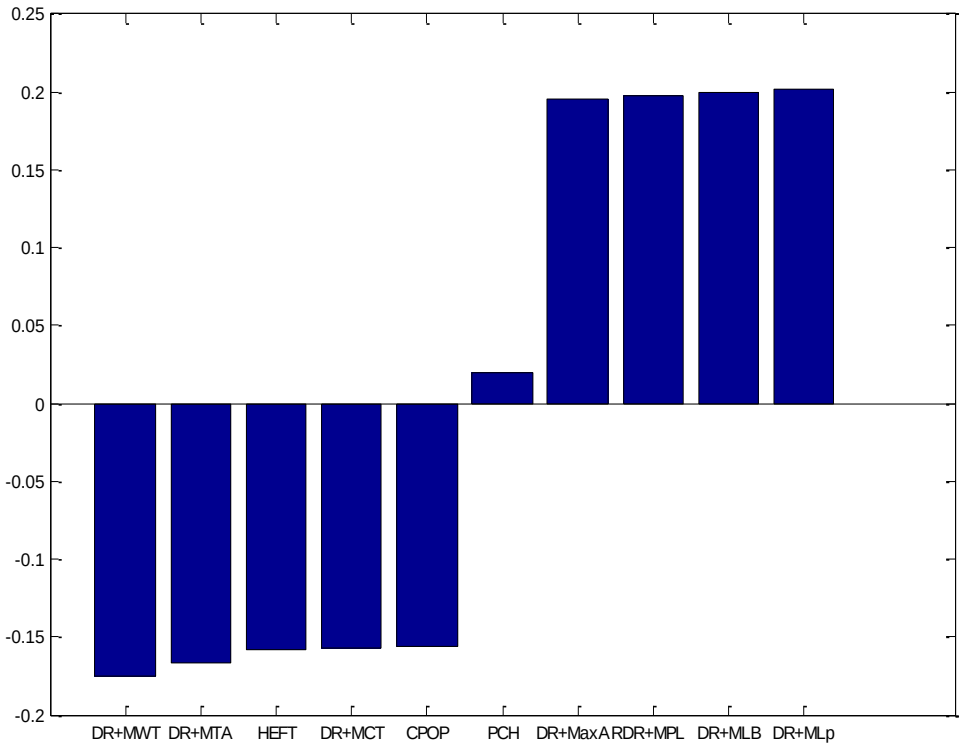

Figura 9. Efectos de speedup sobre la gran media.

Los datos de rendimiento de varias estrategias de colocación con el algoritmo de calendarización local *Backfilling* son mostrados en las Figuras 10 a 12, muestra la degradación del desempeño de cada métrica. El rendimiento promedio de degradación de todas las métricas se ilustra en la Figura 13.

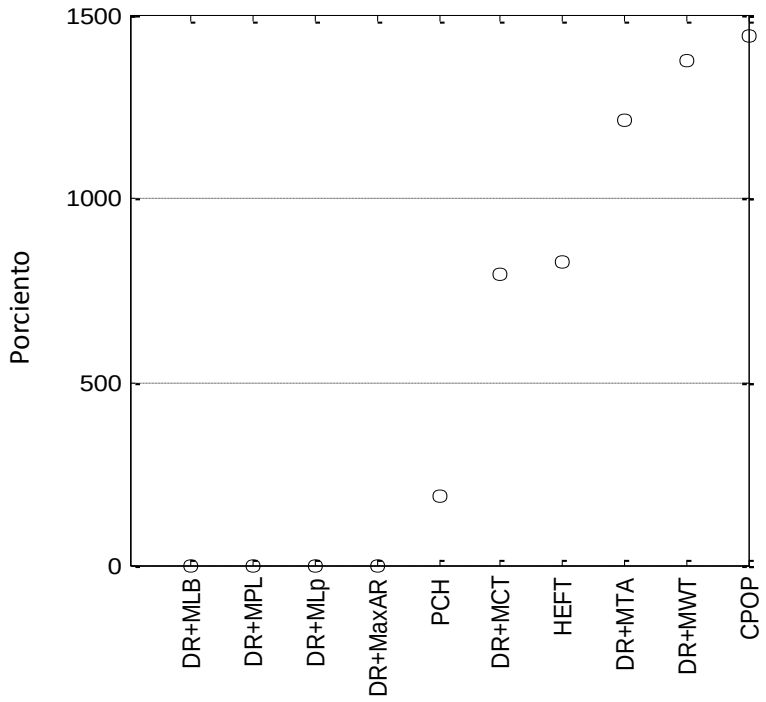

Figura 10. Degradación de desempeño de makespan.

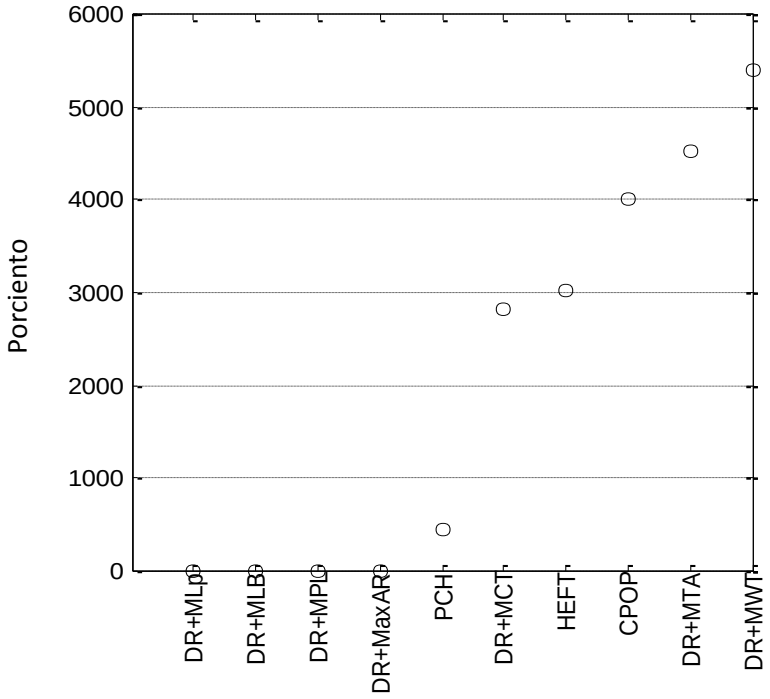

Figura 11. Degradación de desempeño de tiempo de espera.

Dado que speedup es un problema de maximización, un alto rango es una indicativa de buen desempeño. Las estrategias de colocación PCH y DR + {MaxAR, MPL, MLB y MPL} muestran un rango significativamente alto (ver Figura 12).

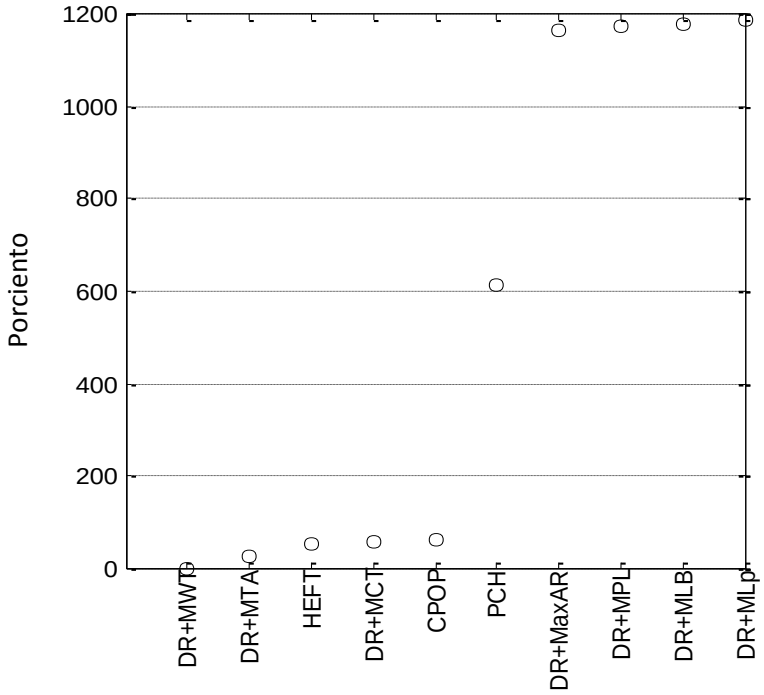

Figura 12. Degradación de desempeño de speedup.

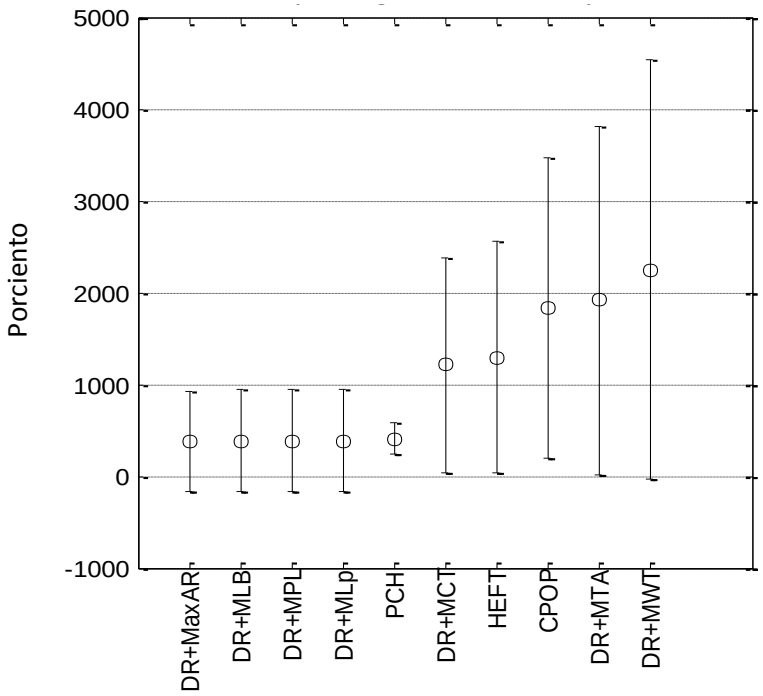

Figura 13. Degradación promedio considerando todas las métricas

V.1 Discusión

El análisis de datos se realizó en base a tres objetivos, Makespan, Speedup y Tiempos de Espera. Los resultados obtenidos mostrarón que el makespan es minimizado por las políticas MPS_Asignación que utilizan información de estado. Las estrategias DR+ {MaxAR, MLB, MLp y MPL} caen en tal caso. La política de colocación con mejor desempeño fue MLp, con un makespan de 78% menor que la gran media. Mientras mayor información tengan las estrategias MPS_Asignación, el makespan crecerá. Tal comportamiento, puede ser explicado al considerar los resultados de tiempo de espera y speedup. De la Figura. 6, es claro que el tiempo de espera de DR+{MaxAR, MLB, MLp, yMPL} es cercano a cero. Tales resultados, sugieren que los trabajos en la ruta crítica tuvieron tiempos de espera pequeños. Los resultados de speedup apoyan las anteriores declaraciones (ver Figura 8 y 9). Un speedup de aproximadamente 40% fue obtenido por el mismo conjunto de estrategias de colocación. Sin embargo, a pesar de que DR+{MaxAR, MLB, MLp, y MPL} tuvieron en promedio un factor de a lo sumo 0.93 menor que la gran media del tiempo de espera, el speedup de los trabajos en la ruta crítica no alcanzaron el valor óptimo de uno. La pérdida en speedup podría ser atribuida a una alta utilización del sistema. Esperábamos obtener tales resultados, dado que el tamaño del contexto de ejecución Grid se mantuvo pequeño. Las estrategias MPS_Asignación que utilizan información del nivel 3, es decir que tienen acceso a todos los calendarios locales, experimentan pérdida de rendimiento en comparación a las estrategias MPS_Asignación basadas en información de estado. Un speedup de aproximadamente 5% fue obtenido por las estrategias DR+MCT, CPOP y HEFT

(ver la Figura 8), estos resultados muestran que los trabajos de la ruta crítica sufrieron tiempos de espera grandes. Tal comportamiento podría ser explicado por el hecho de que las estimaciones de tiempo de ejecución no garantizan la reservación de recursos. Las estimaciones de tiempo de ejecución se crean utilizando una copia clonada del calendario del sitio local, la copia es posteriormente empleada para calcular las estimaciones de tiempo de ejecución de los trabajos. Los recursos que son utilizados en tal cálculo no son reservados en el calendario real. Por lo tanto, cuando un trabajo es colocado en un tiempo posterior, su tiempo de ejecución estimado podría ser inválido.

Independientemente de que nivel de información sea utilizado por la estrategia MPS_Asignación, se requiere consultar el sitio cada vez que un trabajo va a ser calendarizado. La consulta es intrusiva y requiere de acceso a información del sitio. El procedimiento empleado para recabar información de cada sitio puede inducir cargas de procesamiento adicionales. Sin duda, el administrador del sistema puede elegir minimizar el número de consultas, proporcionando información antigua. El efecto de información obsoleta en el desempeño de estrategias MPS_ Asignación queda aún por estudiar. Las estrategias del nivel 3, requieren información del sitio que no sea obsoleta, por lo tanto, las estimación de tiempo de ejecución de los trabajos siempre son calculados.

Los rangos confirman que las estrategias DR+ {MaxAR, MLB, MLp, y MPL} superaron a las estrategias de calendarización restantes. Como se discutió en la capitulo metodología, el rango asume igual importancia de cada métrica. Un

enfoque alternativo es permitir a los interesados ponderar la importancia de cada métrica.

Capítulo VI. Conclusiones

Se implementaron dos estrategias de calendarización: por agrupamiento y rutas críticas. Para lograr tales objetivos (I.1.1-2), fue necesario extender la funcionalidad de tGSF mediante la incorporación de nuevos modelos arquitectónicos y algoritmos. Por ejemplo: el modelo de trabajos compuestos (flujos de trabajo), estrategias para el cómputo de estimaciones de tiempos de ejecución de tareas paralelas y agrupaciones (cadenas), desarrollo de nuevas métricas, implementación de estrategias de etiquetado y el desarrollo de algoritmos auxiliares para el tratamiento de grafos.

Se desarrollo una investigación cuantitativa, la cual consistió en el uso de un diseño experimental de un factor para la evaluación de desempeño de las estrategias propuestas. El proceso de investigación requirió adicionalmente de la elaboración de otras labores, tales como: la preparación del contexto de simulación, el desarrollo de herramientas para automatizar el proceso de experimentación, el desarrollo de herramientas para transformar las salidas de las simulaciones y uso de herramientas para el análisis de datos.

A partir del análisis de los datos se demostró que las estrategias DR+MaxAR, DR+MLB, DR+MPL, DR+MLp y PCH tuvieron buen desempeño, considerando las tres métricas que reflejan tanto los intereses de los usuarios como del sistema. Contrario a lo esperado, encontramos que los calendarios locales y las agrupaciones por ruta crítica no ayudan a mejorar los resultados de las estrategias de calendarización para CPOP y PCH.

Al evaluar el Grid empleando datos reales, se encontró que una distribución en términos del número de procesadores requeridos por los trabajos o el número de trabajos asignados realizado por las estrategias DR+MPL, DR+MLp y términos de los requerimientos de consumo $(size_j * \dot{p}_j)$ del trabajo realizado por la estrategia DR+MLB es más beneficioso. Es decir, entre menos información de los calendarios locales requiera la heurística de asignación, mejor será el desempeño al calendarizar flujos de trabajo.

Referencias

Berman, F., & Hey, T.(2004), The scientific imperative. In I. Foster & C. Kesselman (Eds.), *The grid 2. Blueprint for a new computing infrastructure* (Vol. 2). San Francisco, CA, USA.: Morgan Kaufmann.

Bharathi S., Chervenak, Deelman, Mehta, Mei-Hui Su, & K. Vahi. (Noviembre 2008). Characterization of scientific workflows. pages 1 –10.

Bittencourt Luiz F. & M. Madeira Edmundo R.. (2006). A dynamic approach for scheduling dependent tasks on the xavantes grid middleware. In *MCG '06: Proceedings of the 4th international workshop on Middleware for grid computing*, page 10, New York, NY, USA,

California, U. (2009ª), Boinc. Berkeley open infrastructure for network computing. Consultado el día 9 de Octubre del 2009, de {"http://boinc.berkeley.edu/"}.

California, U. (2009b), Seti@home. Consultado en Agosto del 2009, de {"http://setiweb.ssl.berkeley.edu/"}.

CERN. (2006). Lhc machine outreach. Consultado el día 5 de Octubre del 2009, de { "http://lhc-machine-outreach.web.cern.ch/lhc-machine-outreach/"}.

CERN. (2008a). Grid café. Consultado en Agosto del 2009, de {"http://www.gridcafe.org"}.

CERN. (2008b). Grid café: Grid-powered projects. Consultado el día 8 de Octubre del 2009, de {"http://www.gridcafe.org/grid-powered-project.html"}.

CERN. (2008c). Lhc - the large hadron collider. Consultado el día 5 de Octubre del 2009, de {"http://public.web.cern.ch/Public/fr/LHC/LHC-fr.html"}.

CERN. (2009.). Worldwide lhc computing grid. Consultado el día 5 de Octubre del 2009, de {“http://lcg.web.cern.ch/LCG/public/default.htm”}.

CoreGrid.(Diciembre 2009). Consultado el día 6 de Octubre del 2010, de{“http://www.coregrid.net/”}.

EGEE, P. (2004), Egee-enabling grids for e-science. Consultado el día 11 de Septiembre del 2009, de {“ http://www.eu-egee.org/”}.

Feitelson Dror.(Agosto 2010). Standard workload archive. {“http://www.cs.huji.ac.il/labs/parallel/workload/index.html”}.

Foster, I., & Kesselman, C. (1999). (Eds.), *The grid: Blueprint for a new computing.*
Foster, I., & Kesselman, C. ,The grid in a nutshell. In J. Nabrzyski, J. M. Schopf & J. Weglarz.(2003). *Grid resource management, state of the art and future trends* (1a ed.): Kluwer Academic Publisher.

Fox, G. C. and Gannon, D.(Agosto 2006). Special Issue: Workflow in Grid Systems: Editorials. Concurr. Comput. : Pract. Exper. 18, 10, 1009-1019.

Grimme Christian, Lepping Joachim, Papaspyrou , and Fölling Alexander. (Diciembre 2009). Consultado en Octubre del 2010, en {“http://forge.it.irf.tu-dortmund.de/trac/teikoku/wiki/TeikokuArchitecture#Architecture”}

Grimme Christian, Lepping Joachim, Papaspyrou Alexander, Wieder Philipp, Yahyapour Ramin, Oleksiak Ariel, Wäldrich Oliver, and Ziegler Wolfgang. (Diciembre 2007). Towards a standards-based Grid Scheduling Architecture. CoreGRID Technical Report TR-0123, Institute on Resource Management and Scheduling.

Hirales-Carbajal, Tchernykh A., Roblitz T., & Yahyapour R.. (Abril 2010). A grid simulation framework to study advance scheduling strategies for complex workflow applications. Pp. 1–8.

ISI Pegasus. (Agosto 2010). Workflow generator. {"https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator,"}

Jia Yu & Rajkumar Buyya. (2006). Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming*, 14(3):217–230.

Juve Gideon & Deelman Ewa. (2010). *Scientific Workflows and Clouds.*

Kwok Yu-Kwong & Ahmad Ishfq. (Diciembre 1999). Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors. In ACM computing survey, vol.31, No. 4.

Leung Y-T J..(2004). *Handbook of Scheduling: Algorithms, Models and Performance Analysis*.

OGF.(2006). Open grid forum-grid projects. Consultado el día 9 de Octubre del 2009, de { "http://www.ogf.org/UnderstandingGrids/grid_projects.php"}.

Oxford, U. (2009), Climateprediction.Net. Consultado en Septiembre del 2009, de {"http://climateprediction.net/"}.

Ramakrishnan Arun , Singh Gurmeet , Zhao Henan, Deelman Ewa, Sakellariou Rizos, Vahi Karan, Blackburn Kent, Meyers David, and Samidi Michael. (2007). Scheduling data-intensive workflows onto storage-constrained distributed resources. In *In proceedings of the 7th IEEE Symposium on Cluster Computing and The Grid (CCGrid*, pages 14–17

Ramírez-Alcaraz Juan Manuel, Tchernykh Andrei , Yahyapour Ramin, Schwiegelshohn Uwe, Quezada-Pina Ariel, González-García José Luis & Hirales-Carbajal Adán. (2010). *User run time estimate unaware online job scheduling in hierarchical grids.* Submitted to the Journal of Grid Computing,

Ranganathan, K., & Foster, I.(2003). Computation scheduling and data replication algorithms for data grids. In J. Nabrzyski, J. M. Schopf & J. Weglarz (Eds.), *Grid resource.*

Teragrid-NSF. (2001). Teragrid. Consultado el día 8 de Octubre del 2009, de {“ http://www.teragrid.org/”}.

Topcuouglu, Hariri Salim, & Wu Min-you. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.*, 13(3):260–274.

Washington, U. (2009), Rosetta@home. Consultado en Septiembre 2009, de {“http://boinc.bakerlab.org/rosetta/”}.

Wieczorek Marek , Prodan Radu, & Fahringer Thomas. (2005). Scheduling of scientific workflows in the askalon grid environment. *SIGMOD Rec.*, 34(3):56–62.

Apéndice A

Entradas y salidas del simulador

Una carga de trabajo es una composición de múltiples rastreos de flujos de trabajo reales, tales como: Montage fue creado por NASA/IPAC el cual crea un mosaico del cielo, Cybershake creado por el Centro de Terremotos del sur de California para caracterizar el impacto de terremotos en el área, Epigenomics creado por USC Centro de Epigenome, Ligo empleado para estudiar problemas gravitaionales, y SIPHT empleado para estudiar problemas en el contexto de la bio-informática.

En nuestro estudio empleamos una carga de trabajo conformada por flujos de trabajo Cybershake y Montage. Dada que modelaban trabajos con bifurcaciones grandes y longitud de rutas críticas no homogéneas. Los flujos de trabajo fueron transformados de formato DAX a SWF extendido. Las bitácoras originales pueden accederse a través del projecto Pegasus.

La tabla V ilustra una abstracción de los campos de una carga de trabajos con tareas con precedencias. El simulador tGSF toma la carga de trabajo y las procesa trabajo por trabajo.

Tabla V.
Ejemplo de una carga de trabajo

Numero de tarea	Tiempo de Inicio	Tiempo de procesamiento	Tipo de trabajo	Id del trabajo compuesto	Sucesores	predecesores
1	5	2	DAG	1	2,3,4,5	-1
2	5	3	DAG	1	6,7	1
3	5	3	DAG	1	8	1
4	5	4	DAG	1	8	1
5	5	5	DAG	1	-1	1
6	5	4	DAG	1	9	2
7	5	4	DAG	1	9	2
8	5	4	DAG	1	9	3,4
9	5	1	DAG	1	-1	6,7,8

En total la bitácora consiste de 22 campos en total, los cuales incluyen otras características de las tareas tales como: memoria empleada, memoria solicitada, tiempo de procesamiento estimado por el usuario, entre otros atributos.

La primera columna en la tabla V representa el identificador de la tarea compuesta 1, con tiempo de liberación 5. Note que todas las tareas del flujo de trabajo son liberadas en la misma unidad de tiempo, esto no implica que su ejecución inicie en ese momento. Lo que implica es que el trabajo se encuentra disponible desde esa unidad de tiempo. Sin embargo, su ejecución está sujeta a restricciones de precedencia. Los trabajos adicionalmente están clasificados por tipo. En el ejemplo de la tabla V, solo se resume el tipo DAG. Actualmente, se cuenta con soporte para el modelado de grafos, cadenas, arboles y trabajos independientes. La columna cinco y seis resumen el conjunto de trabajos sucesores y predecesores respectivamente.

La figura 14 ilustra el grafo de la tabla V mediante un grafo acíclico dirigido. Todos los trabajos modelados en tGSF son acíclicos no dirigidos. No existe restricción en cuanto al número de vértices raíz y de finalización.

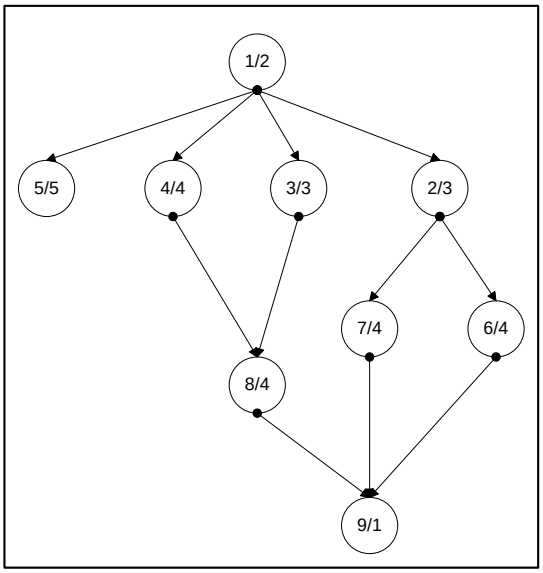

Figura 14. El grafo de la carga de trabajo

El grafo de la figura 14 tiene nueve nodos. Cada nodo esta etiquetado con dos valores: identificador/tiempo estimado de procesamiento. Note que los tiempos estimados de procesamiento son empleados por los algoritmos de etiquetado, y no los tiempos de ejecución real. El vértice de salida tiene identificador nueve y tiempo estimado de procesamiento uno. El algoritmo de etiquetado DR estimaría etiqueta 11 para el vértice con identificador nueva, esto es el valor de la ruta crítica.

```
Job 1 c: 7000
Job 3 c: 10000
Job 5 c: 12000
Job 4 c: 14000
Job 2 c: 15000
Job 8 c: 18000
Job 7 c: 19000
Job 6 c: 22000
Job 9 c: 23000
terminating runtime system on Sun Jan 16 22:09:36 PST 2011,
```

Figura 15. Muestra la salida que nos da el simulador con la carga de trabajo utilizando el algoritmo de calendarización PCH.

tGSF genera bitácoras de texto que almacenan los resultados de las métricas aplicadas a cada trabajo. Al menos una bitácora de salida es creada para cada sitio que conforma el Grid computacional. Para propósitos de ilustración la figura 15 ilustra los tiempos de terminación correspondientes a la métrica makespan a nivel Grid para la estrategia de colocación de trabajos PCH. La figura 16 muestra el diagrama de Gantt estimado. Cabe mencionar que el simulador no construye automáticamente el diagrama.

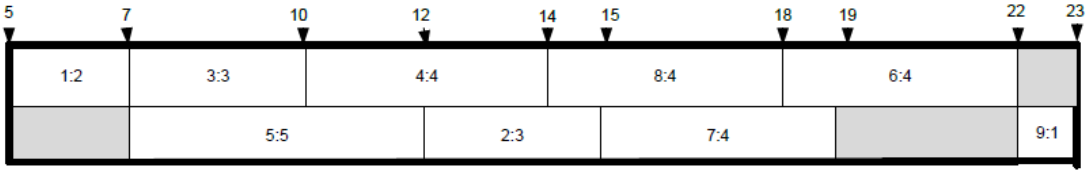

Figura 16. El calendario del algoritmo PCH.

Utilizando los mismos datos de la carga de trabajo de la tabla V, pero con la estrategia de calendarización CPOP. La figura 17 resume los tiempos de finalización de cada tarea.

```
Job 1 c: 7000
Job 3 c: 10000
Job 5 c: 12000
Job 2 c: 13000
Job 4 c: 16000
Job 7 c: 17000
Job 6 c: 20000
Job 8 c: 21000
Job 9 c: 22000
```

Figura 17. Resultados finales utilizando la estrategia de calendarización CPOP.

La figura 18 ilustra el diagrama de Gantt construido por CPOP. El algoritmo genera mejores resultados para el problema de calendarización de un único grafo. Sin embargo, en pruebas de calendarización con una carga compuesta de múltiples flujos de trabajo, el desempeño de PCH es superior.

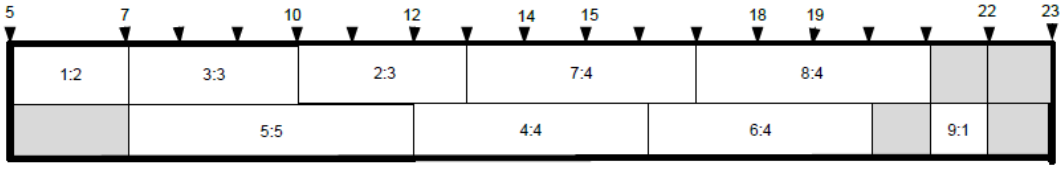

Figura 18. El calendario del algoritmo CPOP