Impacto de las topologías Grid en el monitoreo y descubrimiento automático de recursos

David González Márquez
Directores: Dr. Diego Fernández Slezak, Dr. Esteban Mocskos, Dr. Pablo Turjanski
Jurados: Lic. Cristian Rosa (INRIA) y Ing. Alejandro Furfaro (UBA-UTN)

Departamento de Computación - FCEN - UBA

Categoría: Trabajos Finales de Carreras de Grado

Resumen

Las tecnologías Grid afloran como un nuevo paradigma en la computación de alto rendimiento, proveyendo la infraestructura para compartir recursos geográficamente distribuidos de forma transparente. Para utilizar eficientemente estos recursos es necesario conocer su estado y disponibilidad de forma actualizada. En estos escenarios, la dinámica de los sistemas es muy compleja, y resulta prácticamente imposible capturarla por medio de una estructura estática. En este trabajo se estudian algunos de los mecanismos por los cuales se obtiene y distribuye información de recursos en un Grid. GridMatrix, provee un entorno de ejecución para simular el comportamiento de políticas de distribución de información en entornos Grid, utilizando para esto el motor de simulación SimGrid2.

En esta tesis se extendió la aplicación GridMatrix con herramientas para la generación automática de topologías de red Grid y la generación automática de configuraciones, permitiendo la creación de una amplia gama de combinaciones de parámetros, tanto para la generación de redes, como para la configuración de las simulaciones. La configuración de parámetros para las simulaciones se realiza de forma automática para todas las políticas implementadas: Jerárquica, Super-Peer y Random.

Además fue necesario desarrollar nuevas herramientas metodológicas para el estudio propuesto. Puntualmente se introdujeron métricas para cuantificar la cantidad de información y su validez en la red, se encontraron relaciones entre parámetros de la topología de red y las políticas de distribución de información, y se trabajó en la estimación de resultados por medio de aproximaciones sobre el comportamiento de determinadas políticas.

Se realizó un estudio preliminar del comportamiento de las distintas políticas en las topologías básicas obteniendo interesantes resultados basados en las propiedades estructurales de las redes.

La política Super-Peer mostró resultados muy buenos y estables en diversos escenarios. Se estudió el comportamiento de esta política variando la cantidad de nodos super-peer, mostrando una variación suave a medida que se aumenta la cantidad de super-peers, empezando de un comportamiento prácticamente centralizado, llegando finalmente a un comportamiento similar a Random. A partir de los resultados promisorios conseguidos, se estudió su escalabilidad, analizando redes del orden de 10000 nodos, con un comportamiento muy estable de esta política. Por último, se realizó un experimento para determinar el valor óptimo de tiempo de refresco de la información en función de tiempo de expiración de los recursos. Para redes de gran tamaño, esta relación debe ser tomada con particular cuidado, pudiendo caer en configuraciones con

muy bajos valores de información actualizada en los nodos, o en ciclos de refresco más seguido de lo necesario con el consiguiente mal gasto de recursos.

Los resultados obtenidos, demuestran el poder de análisis disponible y la versatilidad tanto en el desarrollo de topologías como en la configuración de políticas de propagación de información, abriendo un gran número de caminos de estudio.

1. Introducción

Durante las últimas dos décadas se han producido cambios sustanciales en la forma en que percibimos y usamos los recursos y servicios computacionales. Dos décadas atrás, era normal esperar que las necesidades de cómputo fueran satisfechas por la infraestructura tecnológica limitada a un centro de cómputos. Esta situación ha cambiado, y este cambio surge, fundamentalmente, por la creciente transformación de las computadoras y los componentes de red, generando hardware cada vez más poderoso y software más sofisticado. Como consecuencia de estos cambios se abrió la posibilidad de utilizar recursos distribuidos geográficamente de forma eficiente para responder a ciertas necesidades en campos de aplicaciones que así lo requieren[1].

Grid Computing aflora como un nuevo paradigma, proveyendo la infraestructura básica para compartir recursos de forma geográficamente distribuida.

En este contexto, se propone el estudio del impacto de las topologías Grid en el monitoreo y descubrimiento de recursos. Para ello, se extiende la herramienta GridMatrix, que ataca específicamente la problemática de los servicios de información. Esta herramienta permite, mediante una moderna interfaz gráfica y scripts implementados en *Python*, la simulación de diferentes políticas de distribución de información, utilizando las funcionalidades provistas por SimGrid2[2].

1.1. Grid Computing

Grid Computing[1] es un nuevo paradigma en la computación de alto rendimiento que provee la infraestructura básica para compartir recursos de forma transparente y geográficamente distribuida.

Esta tecnología ha incursionado en prácticamente todos los centros de cómputos importantes del mundo siendo, GT4 (Globus Toolkit 4)[3,4] una de las implementaciones más utilizadas para crear estos entornos sobre diferentes sistemas operativos y arquitecturas de computadoras.

GT4 consiste en un conjunto básico de servicios y aplicaciones para la administración de diferentes características del Grid siguiendo un diseño de capas.

Una de las características más importantes en Grid Computing, es la necesidad de mantener información actualizada referente a los recursos y servicios que el Grid provee[5].

El componente responsable de obtener, distribuir, indexar y almacenar la información acerca de la configuración y estado de los servicios y recursos es el GRIS(Grid Resource Information Service).

El enfoque estándar proporciona una organización centralizada para el GRIS, donde las consultas de recursos son enviadas a una máquina central distinguida, la cual provee el servicio de indexación.

Este enfoque centralizado presenta algunos inconvenientes[6] como, por ejemplo, es propenso a un único punto de falla, carece de escalabilidad y posee altos costos de comunicación en los enlaces que conducen información al servidor.

La nueva generación de sistemas de información, como MDS(Monitoring and Discovery System)[7] y Ganglia[8], se basan en una organización jerárquica donde la información de los recursos es enviada entre los diferentes nodos siguiendo una jerarquía.

Estos temas resultan clave frente a un futuro no muy lejano en el que se debe estar preparado para un escenario en el cual los recursos compartidos en la red podrían alcanzar un número tal que todo el planeta podría ser considerado un único Grid. En tal entorno, la dinámica de la aparición o estado de los recursos no puede ser capturada con una jerarquía estática [9] o administrada manualmente.

1.2. Grid Matrix

El estudio en el campo de las arquitecturas Grid puede llegar a ser inabordable utilizando recursos reales, ya que requeriría la puesta en marcha de un desmesurado número de equipamiento en sitios geográficamente distribuidos y bajo diversos dominios de administración. Por esta razón surgieron varios simuladores de procesos distribuidos orientados a arquitecturas Grid. A pesar de esto, ninguno ha sido orientado para estudiar políticas de monitoreo y descubrimiento de recursos. GridMatrix es una aplicación que tiene como objetivo cubrir el vacío existente en este sentido.

Para el desarrollo de GridMatrix se optó por implementar las funcionalidades necesarias para el estudio de políticas de monitoreo y descubrimiento de recursos, sobre un motor de simulación ya existente llamado SimGrid2[2].

SimGrid2 es una herramienta que proporciona funcionalidades para la simulación de aplicaciones distribuidas en entornos heterogéneos. Facilita la programación de aplicaciones en plataformas que van desde simples redes de computadoras a complejas arquitecturas grid. En este último punto, es donde se sustenta la decisión de utilizarlo como motor para GridMatrix.

El comportamiento de los nodos es programado mediante el uso de una interfaz de scripting en *Python*, dando acceso a un entorno muy potente y completo de funcionalidades. La interfaz permite, de manera sencilla, implementar diferentes políticas de propagación de información, incluyendo la capacidad de visualizar el progreso de la ejecución.

Sin embargo, GridMatrix en su versión original, posee un conjunto de limitaciones en relación a los requerimientos necesarios para el estudio propuesto.

En esta tesis se propone abordar el estudio de políticas de propagación de información en una variedad de topologías de red, solucionando las diferentes limitaciones de la herramienta GridMatrix extendiendola en los siguientes puntos:

1. Mejorar la escalabilidad en GridMatrix.
2. Mejorar la interfaz gráfica a fin de permitir una mejor visualización de las redes.
3. Mejorar la compatibilidad con SimGrid2.
4. Implementar la generación automática de topologías de redes.
5. Implementar la generación automática de esquemas de propagación de información con parámetros de entrada particulares para las diferentes políticas y topologías disponibles.
6. Implementar la medición de parámetros de las redes generadas.

2. Metodología

Esta sección incluye un detalle de generadores de redes y políticas de distribución de información implementadas en GridMatrix.

2.1. Generación de redes

Uno de los aspectos más importantes en este trabajo es la generación automática de topologías de red. Se examinaron soluciones preexistentes utilizadas por la comunidad científica.

Finalmente se decidio basar la implementación de los algoritmos en *SIMULACRUM* (SIMUlated pLAtform CReation and User Modification) debido a que implementa una vasta cantidad de algoritmos de generación de topologías de red: *Clique*, *Line*, *Ring*, *Star*, *Uniform*, *Exponential*, *Waxman*, [10], *Zegura* [11] y *Barabasi* [12].

2.2. Políticas de monitoreo y descubrimiento de recursos

El monitoreo y descubrimiento de recursos es fundamental en un sistema distribuido, permitiendo el diagnóstico y la detección de nuevas incorporaciones de recursos. Ambas tareas requieren la recolección de datos desde múltiples fuentes de información distribuidas a lo largo de la red. Estas fuentes, representadas por nodos en la red, incluyen toda la información relacionada al recurso.

Los diferentes nodos están interconectados por medio de una red física a través de canales de comunicación lógicos (rutas) entre los routers. La estructura de comunicación entre nodos es lo que denominamos política de distribución de información.

La comunicación entre nodos puede ser en cualquiera de los dos sentidos; es válida tanto la solicitud como el envío de información a otros nodos. La elección de cuál es el nodo destino, también depende de la política de distribución de la información. En GridMatrix se implementan las políticas Jerárquica y Peer-to-Peer, mencionadas en el trabajo de Mastroiani[13].

Política Jerárquica: Esta política se basa en la estructuración de una jerarquía estática, establecida a priori, entre los nodos de la red. Cada nodo es asignado en algún nivel de la jerarquía, conociendo quién es su padre (nivel

superior) y quiénes sus hijos (nivel inferior). De esta forma, el intercambio de información se realiza entre los nodos padre e hijos, fluyendo entre niveles. La política jerárquica es la que actualmente utiliza MDS para la propagación de información.

2.3. Algoritmos Peer to Peer (P2P)

Las siguientes políticas se engloban como algoritmos P2P(Peer to peer). Las redes P2P son redes lógicas entre nodos que se basan en el intercambio de información entre nodos vecinos que no necesariamente se encuentran cercanos físicamente.

Política Random: Esta política carece de todo tipo de estructura, cada nodo elige un vecino al azar al cual enviar o solicitar información. Una de las ventajas más importante es que no se requiere almacenar información acerca de los nodos vecinos con los que se comunica ya que se escogen aleatoriamente siguiendo una distribución uniforme[14]. La enorme ventaja es la simplicidad del algoritmo, muy fácil de implementar, requiere poca capacidad de cómputo y almacenamiento. En redes muy grandes con vecindarios extensos, la elección al azar de nodos puede ser muy deficiente, llevando a un aumento en la latencia de los mensajes, o incluso creando una congestión innecesaria en la red[14].

Política Super-Peer : Una red P2P pura utiliza parte del ancho de banda en funciones que podrían ser realizadas por una cache local, ahorrando tráfico de red. Es por esto que surgieron, por ejemplo, las redes Super-Peer como punto intermedio entre sistemas totalmente distribuidos y servicios basados en cache[15]. Estas redes requieren identificar a un conjunto de nodos como super-peers, entre los cuales se dispondrá la interacción usando una política P2P pura. El resto de los nodos actuará de forma diferente; como clientes, ya que tendrán asignado un Super-Peer al cual realizarle consultas, es decir una distribución jerárquica de un nivel.

2.4. Métricas de la calidad de la información

Para poder dar cuenta del rendimiento de las políticas de distribución de información se debe poder medir la calidad y cantidad de la información proporcionada por cada una de ellas. Esta métrica debe estar enfocada a cuantificar la información, tanto de forma global en toda la red, como de forma particular en un solo nodo de la red. Para realizar esta tarea se plantean principalmente dos índices. Estos sirven para comparar las diferentes políticas. A continuación se describe cada uno de ellos.

- Local Information Rate (LIR): Este coeficiente está comprendido entre 0 y 1. Captura la cantidad de información que un nodo particular tiene de toda la red completa en un solo momento, ponderando el tiempo de expiración de esta información. El valor 1, significaría que el nodo conoce cada fragmento

de información sobre la red entera, y ésta es tan nueva como pueda serlo. La definición de LIR para el nodo k es la siguiente:

$$LIR_k = \frac{\sum_{h=1}^{N} \frac{(expiration_h - age_h)}{expiration_h} \cdot resourceCount_h}{totalResourceCount} \quad (1)$$

Donde N es el número de nodos en todo el sistema, $expiration_h$ es el tiempo de expiración de los recursos del nodo h en el nodo k, age_h es el tiempo desde que la información fue obtenida por ese nodo, $resourceCount_h$ es la cantidad de recursos en el nodo h y $totalResourceCount$ es el total de recursos en toda la red. La expresión $expiration_h - age_h$ es similar al parámetro conocido como *time-to-live* en el protocolo `IP`. Es importante aclarar que age_h puede valer como máximo $expiration_h$.

- Global Information Rate (GIR): Este coeficiente también es un valor entre 0 y 1. Captura la cantidad de información que toda la red conoce sobre sí misma. Es calculado como el promedio de los valores LIR de todos los nodos.
- Global Information Variability (GIV): Este coeficiente es el desvío estándar del valor GIR. Mide la variabilidad sobre GIR.

El cálculo de estos índices está integrado a GridMatrix. Como resultado de una simulación se obtienen estos índices, tanto a nivel global, como indicando particularmente los nodos sobre los cuáles se desea obtener estos valores.

3. Desarrollo de GridMatrix2

El objetivo principal del presente trabajo consiste en estudiar el comportamiento de las políticas de distribución de información sobre diferentes topologías de red. Para abordar este problema, se extendió la herramienta GridMatrix con dos características de manera independiente: por un lado, (1) la generación automática de topologías de red, y por el otro, (2) la generación de esquemas de propagación de información con parámetros de entrada particulares para las diferentes políticas disponibles.

Adicionalmente, mediante el uso de las herramientas desarrolladas es posible caracterizar los parámetros de redes existentes, posibilitando la generación de escenarios sintéticos con características similares a redes reales.

3.1. GridMatrix2

La extensión de GridMatrix consta de tres nuevos módulos. El primero, encargado de la generación de redes, construye a partir de una determinada configuración un archivo de descripción de red (`.net`). El segundo, el generador de archivos *platform*, transforma el archivo de descripción de red (`.net`) a un archivo *platform*, necesario como entrada de SimGrid2. El tercer y último módulo se encarga de generar archivos *deploy*, los cuales describen la configuración de una simulación. En la figura 1 se ilustra la interacción entre los archivos y las nuevas prestaciones de GridMatrix2.

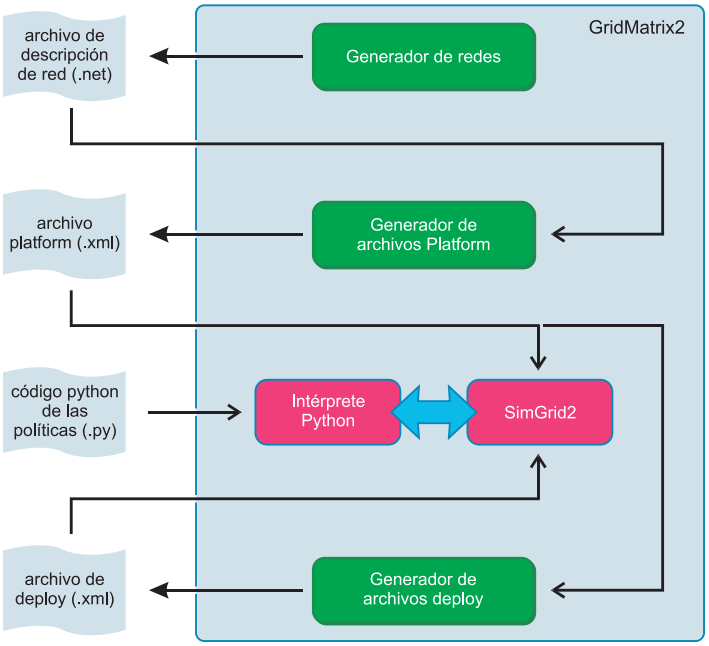


Figura 1: Esquema de interacción entre archivos en GridMatrix2. En verde se muestran las nuevas prestaciones implementadas.

4. Resultados

En esta sección se muestran algunos resultados obtenidos usando las nuevas características implementadas en GridMatrix2.

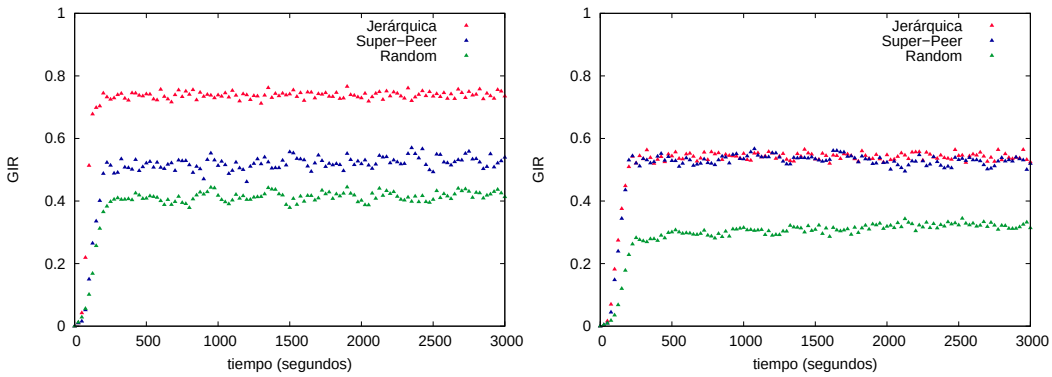
4.1. Topologías básicas de red

Se comenzó con el estudio del comportamiento de algunas topologías de red básicas. Para todos los casos, se generaron redes de 50 routers y una cantidad de nodos variable (100, 200 y 300 nodos). El objetivo es mantener la estructura subyacente de la red, pero aumentar su tamaño por medio del incremento de nodos.

Cliqué:

En esta topología todos los routers se encuentran conectados entre sí, los nodos se conectan de forma balanceada, es decir, todos los routers tienen conectados aproximadamente la misma cantidad de nodos.

La figura 2 muestra el comportamiento a nivel global en esta topología de cada una de las políticas en escenarios en los cuales se va aumentando gradualmente la cantidad de nodos del sistema entre 100 y 300 nodos.



(a) 100 nodos (b) 300 nodos

Figura 2: Topología cliqué: GIR para las diferentes políticas. Se usan 50 routers y se varía la cantidad de nodos utilizando un tiempo de expiración distribuido uniformemente entre 200 y 300 segundos. Super-Peer presenta el resultado más estable en todos los casos, mientras que jerárquico con tres niveles solo la supera en los sistemas de 100 nodos. Random pierde rápidamente eficiencia con el aumento del tamaño del sistema, pero no deja de ser una opción interesante dado que el GIR cae solo un 50 % cuando el sistema crece un 300 % en cantidad de nodos.

En este tipo de topología el camino más largo entre dos nodos es de dos *hops* y el más corto de tan solo uno, por lo que el camino no afectaría provocando que cualquier política se comporte relativamente bien.

La política Random muestra una caída gradual del rendimiento a medida que aumenta el tamaño del sistema. Esto se debe a que, al aumentar la cantidad de nodos y la cantidad de recursos existentes, la política Random se muestra menos eficiente para recolectar la información. De todos modos, constituye una alternativa interesante dado que un aumento de 300 % en la cantidad de nodos del sistema repercute en una caída de aproximadamente 50 % en el GIR.

En los escenarios considerados, la política Super-Peer se comportó de una manera muy estable y con buen rendimiento a pesar del aumento en el tamaño del sistema.

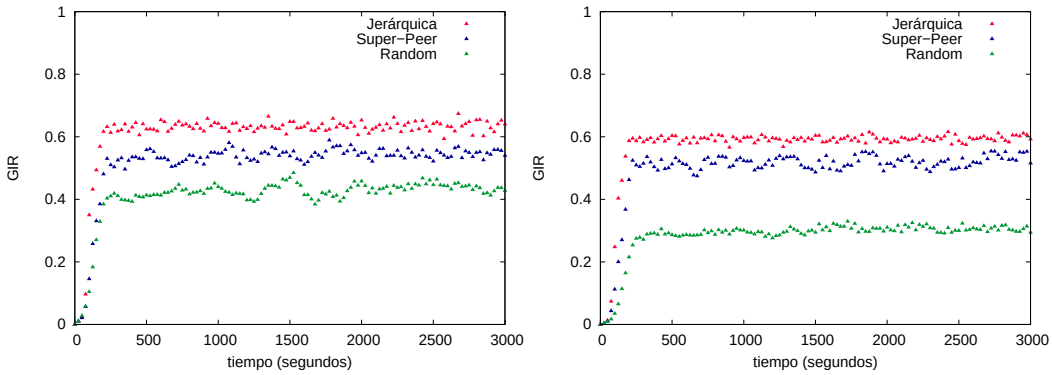
La política Jerárquica presenta resultados interesantes, ya que si bien su comportamiento para 100 nodos supera al del resto de las políticas evaluadas (figura 2a), para los sistemas de mayor tamaño aparece una caída en el rendimiento que lo empareja a Super-Peer. La jerarquía que se contruye sobre el grafo cliqué tiene tres niveles:

- el primer nivel está compuesto únicamente por el nodo *root*.
- en el segundo nivel aparecen todos los nodos que están conectados al mismo router que el nodo *root*, es decir, que el camino para llegar al *root* es de un *hop*.
- en el nivel tres quedan todos los nodos restantes de la red.

Esta forma de organización no aprovecha las características de la topología cliqué. Tener tres niveles significa que, para intercambiar información entre dos nodos hoja (pertenecientes al tercer nivel), es posible que el camino a seguir sea mucho más largo que si directamente fuera enviada entre ellos.

Línea:

En esta topología los routers están conectados entre sí uno a continuación de otro, con aproximadamente la misma cantidad de nodos conectados a ellos. El camino más largo entre un par de nodos en esta topología está directamente relacionada con la cantidad de routers.



(a) 100 nodos (b) 300 nodos

Figura 3: Topología línea: GIR para las diferentes políticas. Se usan 50 routers y se varía la cantidad de nodos utilizando un tiempo de expiración distribuido uniformemente entre 200 y 300 segundos. Tanto la política Jerárquica como la Super-Peer se comportan de manera muy estable en todos los casos estudiados, siendo la Jerárquica la que mejor desempeño presenta.

Intuitivamente, esta topología no parece ser propicia para el intercambio de información entre nodos. Si la comunicación entre nodos superara el ancho de banda disponible en alguno de los links, se observaría congestión y por ende una disminución en la eficiencia de los índices de información. Este inconveniente, es de particular relevancia en la política Jerárquica ya que la elección del nodo *root* y la jerarquía utilizada puede provocar la congestión si esta se contruyera incorrectamente. El resto de las políticas, si bien podrían sufrir el mismo síntoma, poseen una porción aleatoria que distribuye (al menos, en parte) la congestión.

Por este motivo, se eligieron redes con abundante ancho de banda, y se generaron las jerarquías tomando distintos nodos *root* para corroborar que no se produjera este fenómeno. Se ejecutaron las diferentes políticas variando la cantidad de nodos en la red, manteniendo fija la cantidad de routers. Los resultados de estos escenarios se presentan en la figura 3.

De manera muy similar a la topología cliqué, la política Random muestra una caída en el desempeño a medida que aumenta el tamaño del sistema, mientras que las políticas Jerárquica y Super-Peer mantienen estable su desempeño en todos los tamaños de sistema estudiados. La política Jerárquica es la que presenta mejores resultados, mientras que, levemente por debajo, Super-Peer sigue representando una buena alternativa en infraestructuras de esta escala.

4.2. Impacto de la cantidad de Super-Peers

En esta sección se estudia la influencia de la cantidad de super-peers en el comportamiento de esta política.

Es importante destacar, que la política Super-Peer presenta dos casos extremos que son interesantes de remarcar: por un lado si todos los nodos de la red fueran super-peers sin nodos asociados, se estaría en presencia de una política Random pura. Por otro lado, si en esta forma de organización se eligiese un único super-peer para toda la red, es decir, si todos los nodos estuvieran asociados a un único super-peer, se estaría en presencia de una política Jerárquica con dos niveles o esquema centralizado.

El escenario de pruebas consta de un sistema en el que se mantiene constante la cantidad de nodos, pero se aumenta gradualmente la cantidad de super-peers y de routers. El aumento de routers impacta directamente en la longitud de los caminos entre nodos que quedan definidos en la red. Nuevamente, las redes contruidas son similares a internet, siguiendo la ley de potencias, con 500 nodos y utilizando 100 y 300 routers. Las simulaciones fueron realizadas con configuraciones de 10, 30 y 50 super-peers.

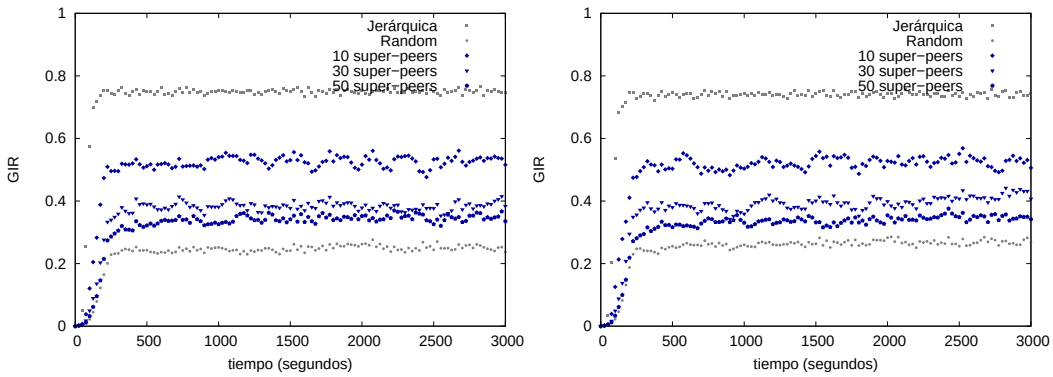
Es importante notar que el número promedio de hijos de los nodos super-peer dependerá la cantidad de super-peers utilizados en la simulación. Por ejemplo, para 20 super-peers corresponderá que cada uno tenga $500/20 = 25$ hijos. Al aumentar la cantidad de nodos super-peers, la cantidad de hijos de cada nodo disminuirá.

En la figura 4 se presentan los resultados de esta simulación. Para poder comparar el comportamiento de Super-Peer, se incluyen en la figura los gráficos de las políticas Random y Jerárquica de dos niveles (esquema centralizado).

En las figuras 4a y 4b se observan las políticas Random y Jerárquica de dos niveles en los dos extremos de valor de GIR. De forma consistente con los resultados de las secciones anteriores, la política Jerárquica presenta valores de GIR cercanos a 0,8, mientras que Random se mantiene alrededor de 0,2. Es interesante observar que Random no disminuye su valor con el aumento de routers, sino que esto depende particularmente del aumento en la cantidad de nodos, en este caso constante.

El valor de GIR de Super-Peer se encuentra entre los valores de las dos políticas mencionadas, y al aumentar la cantidad de nodos super-peer el valor de GIR disminuye, acercándose cada vez más a los valores obtenidos por Random.

Como se observa en los resultados, el comportamiento en promedio de esta política actuará de forma que nunca caiga por debajo del desempeño de la



(a) 100 routers (b) 300 routers

Figura 4: GIR de las políticas Random, Jerárquica de dos niveles y super-peer en un sistema con 500 nodos, con 100 routers (panel izquierdo) y 300 routers (panel derecho), variando la cantidad de super-peers seleccionados.

política Random y además no supere el valor dado por la política Jerárquica de dos niveles.

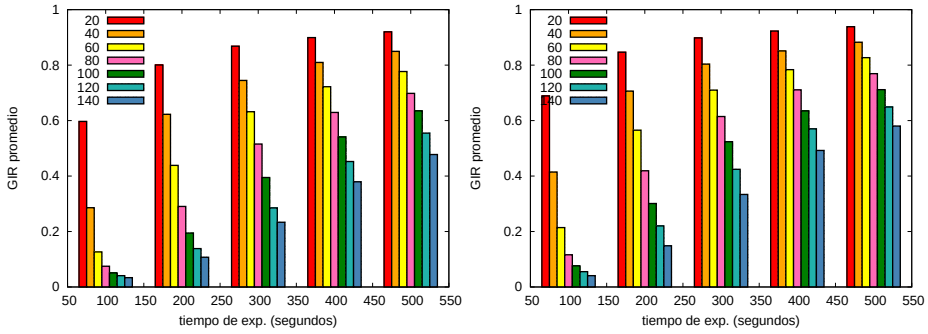
Vale la pena aclarar que, como se dijo en la sección anterior, el comportamiento de la política estudiada con pocos nodos super-peer (en comparación con la cantidad total de nodos) es prácticamente igual al visto en la política Jerárquica de tres niveles (no graficado por claridad). Sin embargo, Super-Peer presenta varias ventajas por sobre la política Jerárquica. Por un lado, la subdivisión de nodos a lo largo de la red se puede realizar de forma independiente, pudiendo separar por entidad de administración independientes. Luego, la interconexión de estas entidades se realiza por una política P2P pura (aleatoria), lo que requiere muy poco esfuerzo de implementación. Además, no posee punto único de falla; al menos, no de forma global.

4.3. Impacto de la relación entre tiempo de expiración y ciclo de actualización

En esta sección se busca profundizar el estudio de la relación entre el tiempo de expiración de la información y el ciclo de actualización entre los nodos.

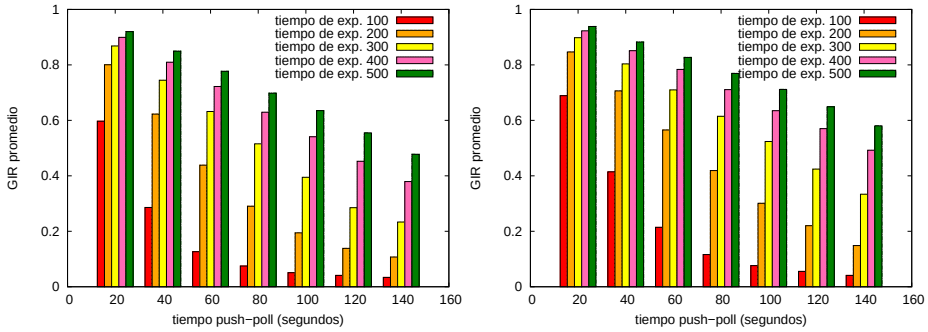
Las figuras 5 y 6 muestran el GIR promedio de una corrida completa en las distintas políticas descartando los instantes iniciales, dado que en ese periodo inicial todas las políticas presentan un tiempo durante el cual la información empieza a fluir hasta que se estabiliza, por lo que no aporta demasiado a fin del análisis que se desea realizar.

En todos los casos, una disminución en el ciclo de actualización (es decir, aumento de la frecuencia con que los nodos se comunican entre sí) y un aumento en el tiempo de expiración producen mejores resultados sin importar la política que se analice.



(a) Jerárquico (b) Super-Peer

Figura 5: GIR promedio de las distintas políticas para distintos valores de tiempo de expiración y ciclo de actualización fijo



(a) Jerárquico (b) Super-Peer

Figura 6: GIR promedio en las distintas políticas para distintos valores de tiempo de actualización dejando fijo el tiempo de expiración de la información. En todos los valores de ciclo de actualización, el GIR promedio obtenido aumenta con el tiempo de expiración. Esto ocurre para todas las políticas.

En las figuras 5 y 6 se estudia la relación entre el ciclo de actualización y el tiempo de refresco, para lo cual, en ambos casos se dejó uno de los dos valores fijo y se varió el otro. A pesar de la valiosa información que muestran estos gráficos, para analizar con mayor profundidad las distintas políticas entre sí, introducimos un nuevo parámetro R:

$$R = \frac{tiempo\_expiracion}{ciclo\_actualizacion} \tag{2}$$

De acuerdo a esta definición, cuando R se acerque a 1, el tiempo de expiración de la información y el ciclo de refresco serán muy parecidos entre sí. Esto significa que la información expirará rápidamente y no dará tiempo a los no-

dos a comunicar información válida al resto del sistema. En este caso extremo, estaríamos ante una situación en la cual todos los nodos sólo contarían con la información de sus propios recursos o, a lo sumo, con la de sus vecinos directos.

Por otro lado, el crecimiento de R significa que la información existente en el sistema tarda muchos ciclos de actualización en expirar, lo que puede permitir que viaje trayectos muy largos y aumente fuertemente la cantidad de información válida con que cuentan los nodos del sistema.

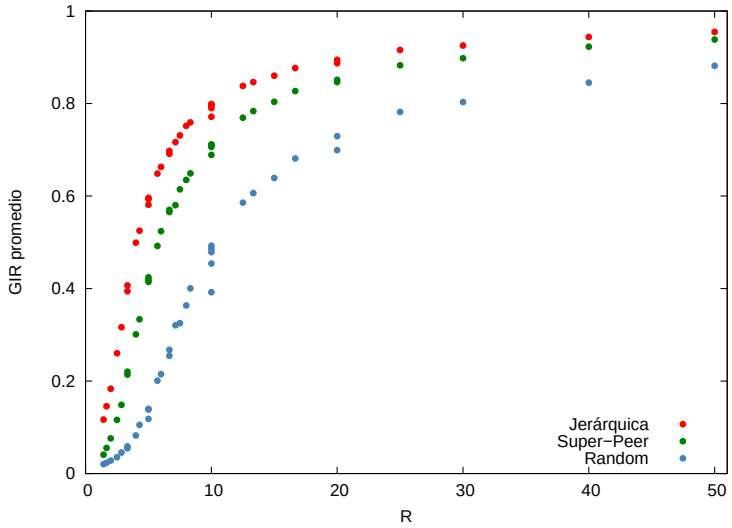


Figura 7: GIR obtenido por distintas políticas en función de la relación entre tiempo de expiración y frecuencia de actualización (R).

Como base para la ejecución de las simulaciones cuyos resultados se resumen en la figura 7 se usa un escenario con un sistema que cuenta con 100 nodos y 100 routers. En los límites de los valores de R analizados, todas las políticas suelen tener un comportamiento semejante: cerca de 0, todas tienen valores muy bajos de GIR, mientras que al alejarse hacia los valores altos, todas presentan una pendiente suave de mejora que las termina acercando entre sí.

La política Jerárquica (esquema centralizado) es la que mejor resultados muestra en los rangos intermedios. Por otro lado, la política Super-Peer se comporta de manera más deficiente que la Jerárquica, aunque con un buen comportamiento general en todo el rango de R estudiado. Finalmente la política Random presenta el peor desempeño inclusive para valores altos de R.

5. Conclusiones

En esta tesis se abordaron los estudios de políticas de propagación de información en una variedad de topologías de red, extendiendo la herramienta

GridMatrix para subsanar sus limitaciones. Para ello se utilizaron los índices GIR, LIR y GIV que miden la información disponible tanto a nivel global como local (en cada nodo).

La implementación de la nueva versión de GridMatrix consistió en implementar dos módulos. El primero, encargado de la generación de redes, construye a partir de una determinada configuración un archivo de descripción de red y su traducción para el simulador. El segundo, el generador de archivos, describe la configuración de una simulación particular, es decir las políticas a utilizar en cada uno de los nodos.

A partir de este desarrollo se pudo estudiar el comportamiento de las distintas políticas de propagación de información en diferentes topologías de red. En primer lugar, se analizó el comportamiento en las topologías más básicas. Como era de esperar, el esquema jerárquico de dos niveles (esquema centralizado) mostró los valores más altos de eficiencia. En segundo lugar, Super-Peer y Jerárquico de tres niveles mostraron comportamientos similares, con la ventaja del primero de no poseer punto único de falla y administración más simple.

Con los datos recabados, se decidió ahondar en el análisis de la política Super-Peer ya que en los ensayos mostró una excelente relación entre eficiencia y costo de mantenimiento. Se estudió esta política en algunos escenarios predefinidos, en los cuales se realizó un barrido de la cantidad de super-peers. Se obtuvieron resultados alentadores, promoviendo a esta política como una excelente alternativa, presentando resultados similares (incluso, mejores) que aquellos obtenidos por la política jerárquica de tres niveles. Para obtener estos resultados se calculó el límite de nodos super-peer en la red para el mejor funcionamiento, manteniendo un alto valor de GIR y cierto tamaño de cluster de nodos que facilite el mantenimiento y sea tolerante a fallas.

Por último, se profundizó en el estudio del comportamiento de las políticas en función de tiempo de vida de la información de recursos. Para ello se definió un nuevo índice(R) que relaciona el tiempo de expiración de los recursos con el tiempo del ciclo de actualización entre nodos. La política Jerárquica con dos niveles (esquema centralizado) mostró los mejores resultados en los rangos intermedios. Por el contrario, es muy interesante que la política Super-peer se comportó consistemente mejor que la Jerárquica de tres niveles en todos el rango de R estudiado.

La herramienta desarrollada en esta tesis permitió un estudio profundo del comportamiento de las distintas políticas en una diversidad de topologías de red. Los resultados demuestran un gran poder de análisis disponible, con versatilidad tanto en el desarrollo de topologías como en la configuración de políticas de propagación de información. Además propone un fuerte aporte metodológico, plantenado distintos escenarios, definiendo índices para cuantificar la propagación de información y obteniendo conclusiones generales.

Referencias

1. D. De Roure, M.A. Baker, N.R. Jennings, and N.R. Shadbolt. *Grid computing - making the global infrastructure a reality*, chapter The evolution of the Grid, pages

65–100. John Wiley & Sons Ltd, 2003.
2. Henri Casanova, Arnaud Legrand, and Martin Quinson. Simgrid: A generic framework for large-scale distributed experiments. In *10th IEEE International Conference on Computer Modeling and Simulation*, pages 126–131, Los Alamitos, CA, USA, March 2008. IEEE Computer Society.
3. Ian Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. In H. Jin, D. Reed, and W. Jiang, editors, *IFIP International Conference on Network and Parallel Computing 2005*, volume 3779, pages 2–13. Springer Berlin / Heidelberg, 2005.
4. Globus Alliance web page. last visited on 02/11/2011.
5. Giovanni Aloisio, Massimo Cafaro, Italo Epicoco, Sandro Fiore, Daniele Lezzi, Maria Mirto, and Silvia Mocavero. Resource and service discovery in the igrid information service. In *Computational Science and Its Applications - ICCSA*, pages 1–9, 2005.
6. R. Ranjan, A. Harwood, and R. Buyya. Peer-to-peer-based resource discovery in global grids: a tutorial. *Communications Surveys & Tutorials, IEEE*, 10(2):6–33, 2008.
7. K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *HPDC '01: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, pages 181–194, Washington, DC, USA, 2001. IEEE Computer Society.
8. Federico D. Sacerdoti, Mason J. Katz, Matthew L. Massie, and David E. Culler. Wide area cluster monitoring with ganglia. In *Proceedings of the IEEE International Conference on Cluster Computing*, pages 289–298. IEEE Computer Society, December 2003.
9. P. Trunfio, D. Talia, C. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi. Peer-to-Peer resource discovery in Grids: Models and systems. 23(7):864–878, August 2007.
10. Bernard M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, December 1988.
11. Kenneth L. Calvert, Matthew B. Doar, and Ellen W. Zegura. Modeling Internet Topology. *IEEE Communications Magazine*, 35(6):160–163, June 1997.
12. Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.
13. Carlo Mastroianni, Domenico Talia, and Oreste Verta. Designing an information system for grids: Comparing hierarchical, decentralized P2P and super-peer models. 34(10):593–611, 2008.
14. A. Iamnitchi, I. Foster, and D. Nurmi. A peer-to-peer approach to resource discovery in grid environments. In *Proceedings of the 11 th IEEE International Symposium on High Performance Distributed Computing HPDC-11 (HPDC'02)*, page 419, Edinbourgh, UK, Jul 2002. IEEE.
15. Diego Puppin, Stefano Moncelli, Ranieri Baraglia, Nicola Tonellotto, and Fabrizio Silvestri. A grid information service based on peer-to-peer. In José C. Cunha and Pedro D. Medeiros, editors, *Euro-Par 2005 Parallel Processing*, volume 3648 of *Lecture Notes in Computer Science*, pages 454–464. Springer, 2005.

6. Apendice: Desarrollo de GridMatrix2

En este apendice se presenta el desarrollo de (1) la generación automática de topologías de redes (sección 6.3) y (2) la generación de esquemas de propagación de información con parámetros configurables (sección 6.4).

En primer lugar se abordará la descripción del estado actual de GridMatrix.

6.1. GridMatrix

Como se mencionó en la introducción, GridMatrix es una aplicación desarrollada para el estudio de las políticas de propagación de información en arquitecturas grid. Para el desarrollo de GridMatrix se utilizó SimGrid2 como motor de simulación, el cual posee diferentes interfaces de alto nivel. De todas ellas, GridMatrix utiliza **MSG** de su sigla original "Meta-SimGrid" correspondiente a la interfaz denominada "Simple programming environment"

MSG fue originalmente desarrollada para el estudio de algoritmos de scheduling, que luego, gracias a posteriores actualizaciones, se convirtió en una de las API más utilizadas de SimGrid2. Esta API permite la creación de aplicaciones distribuidas, a través de un prototipado de funciones que se ejecutan como procesos en los diferentes nodos de una red.

Una aplicación consiste en procesos que pueden ser creados, suspendidos o abortados dinámicamente. Estos procesos pueden ser sincronizados mediante el intercambio de mensajes. Para este intercambio de mensajes entre procesos, SimGrid2 tiene en cuenta una estructura de red y por lo tanto los mensajes son sincronizados en base a los retardos de los diferentes links dentro de la red.

Para ejecutar una simulación en SimGrid2 se requiere de dos archivos de configuración XML. El primero de ellos representa la red (*platform file*), conteniendo una descripción de la distribución e interconexión de los diferentes equipos, en términos de nodos y links. El segundo archivo describe qué procesos serán ejecutados en cada nodo (*deploy file*).

Una vez que ambos archivos XML son definidos, se deben implementar los programas que correrán en cada nodo. Estos programas son registrados en SimGrid2 como funciones. En el archivo *deploy file* se indicará a cada nodo qué función registrada será la que ejecute, desencadenando un proceso asociado a dicha función. La registración de funciones en SimGrid2 es un punto clave a tener en cuenta. En vistas a la implementación de GridMatrix, esta cuestión será explicada en detalle más adelante.

Una vez configurados estos archivos se puede proceder a la ejecución de la simulación. En la figura 8 se ilustra el esquema de interacción entre archivos en GridMatrix. Notar que el *platform file* es generado automáticamente por la aplicación GridMatrix a partir de un archivo de descripción de la red (.net). Este último posee la descripción de toda la red junto con características adicionales que el *platform file* no posee, por ejemplo la posición de los elementos dentro de la interfaz gráfica.

GridMatrix interactúa con SimGrid2 de forma de proveer una interfaz para asistir en la configuración de la red. Como se mencionó en la introducción, el com-

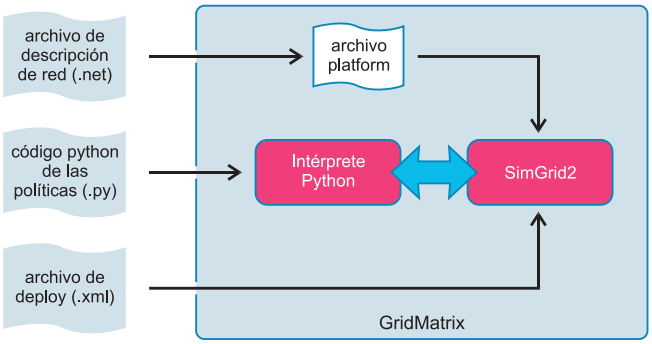


Figura 8: Esquema de interacción entre archivos en GridMatrix

portamiento de los nodos es programado mediante el uso de scripts en *Python*. Este detalle es fundamental, ya que el uso de *Python* abre un gran abanico de posibilidades. Además de la sencillez en el uso y la facilidad para la modificación del comportamiento programado. Es posible construir una política o realizar un cambio a una política existente en poco tiempo. Incluso, calcular parámetros en la simulación sin tener que construir ningún tipo de código adicional relevante.

Para poder correr simulaciones utilizando código *Python*, fue desarrollado un conjunto de *bindings* que hacen posible el intercambio de datos entre la interfaz gráfica en *C++* y el código *Python* que implementa las políticas. Desde un punto de vista práctico, en el código *Python* se incluye una biblioteca que provee la forma de acceder a las funcionalidades de la interfaz gráfica.

SimGrid2 necesita tener registradas las funciones que ejecutará en cada nodo. Las mismas están escritas en *Python*, y es un requerimiento que SimGrid2 ejecute este código. Pero esto último no es posible porque para registrar una función en SimGrid2, ésta debe estar programada en *C*. Para solucionar dicho problema, se optó por generar una única función en código *C*, encargada de ejecutar una función en *Python* distinguida. Con este mecanismo, SimGrid2 ejecuta un solo proceso correspondiente a la función *config*, encargada de ejecutar en todos los nodos las tareas configuradas desde el archivo *deploy*. En el archivo *deploy* se describe un solo proceso que ejecuta la función *config*. Los parámetros pasados a esta función son los que conforman la configuración para cada uno de los nodos.

Otro punto importante a mencionar corresponde a una decisión de diseño que será modificada en la segunda versión de GridMatrix. Cuando se ejecuta una simulación, como primera fase de la misma, es generado de forma automática el archivo *platform*. El archivo se construye temporalmente para la actual simulación. El período de tiempo invertido en la costosa tarea de generar este archivo es inútil para posteriores simulaciones, ya que no puede ser almacenado ni reutilizado. Para redes del orden de 500 nodos, esta tarea puede llevar un tiempo considerable (del orden de minutos en las pruebas realizadas).

6.2. GridMatrix2

Como fue explicado en la introducción, principalmente se busca implementar la generación automática de topologías de red y la configuración automática de escenarios de simulación. Principalmente se busca implementar la generación automática de topologías de red y la configuración automática de escenarios de simulación. Para estas dos tareas fue necesario implementar tres módulos.

La extensión de GridMatrix consta de tres nuevos módulos. El primero, encargado de la generación de redes, construyendo a partir de una determinada configuración un archivo de descripción de red (.net). El segundo, el generador de archivos *platform*, encargado de transformar el archivo de descripción de red (.net) a un archivo *platform*, necesario como entrada de SimGrid2. En secciones posteriores se describe este proceso y la relación entre ambos archivos. El tercer y último módulo se encarga de generar archivos *deploy*, los cuales describen la configuración de una simulación.

En la figura 1 en el cuerpo principal, se ilustra la interacción entre los archivos y las nuevas prestaciones de GridMatrix2. Es especialmente interesante la comparación con la figura 8 que muestra el funcionamiento de la versión anterior de GridMatrix.

6.3. Generación de redes

Previo a abordar el problema de la generación de redes, se analizarán las cuestiones referentes al almacenamiento de éstas por parte de SimGrid2. El proceso de generación de redes terminará construyendo un archivo de descripción de red (.net).

Como fue mencionado con anterioridad, este archivo posee un formato diferente al utilizado por SimGrid2 (archivo *platform*). Por esta razón se realiza un proceso posterior correspondiente a la generación del archivo *platform*. Para comprender los fundamentos de esta transformación y la inherente necesidad de su existencia, debemos interiorizarnos en cómo se almacena la información en los archivos *platform*.

SimGrid2 almacena tres elementos en sus archivos de descripción de red.

- Nodos: Elementos de cómputo de la red.
- Links: Conexiones entre elementos de la red, sean nodos o routers.
- Rutas: Camino de Links entre dos Nodos. Están conformados por dos nodos (fuente y destino), y una lista de Links intermedios.

Las rutas poseen sólo información acerca de los links que las conforman. No tienen ninguna información acerca de los routers que conectan estos links. Un ejemplo de este concepto se ilustra en la figura 9.

En base a esta representación, se puede observar que la topología almacenada no es la topología *real* de la red, sino que es un posible conjunto de rutas entre todos los nodos. Si se parte de este conjunto de rutas sólo será posible reconstruir una fracción de la red original, ya que si ninguna ruta pasa por un determinado

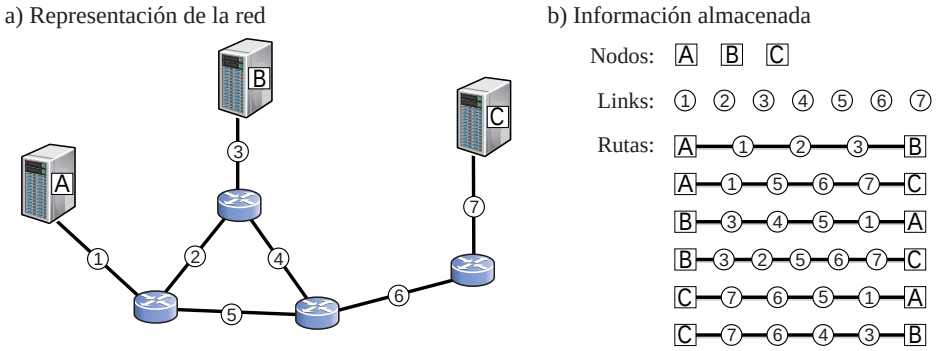


Figura 9: Ejemplo de la información almacenada en un archivo *platform* de SimGrid2, en términos de nodos, links y rutas.

link, este link no será almacenado. Por otro lado, las rutas se construyen a partir de los links, no dando cuenta de los routers que interconectan estos links. Puede observarse que a partir de esta representación se puede reconstruir, en términos de grafos, solo una fracción del grafo de líneas que se induce de la red original. Siguiendo el resultado clásico de teoría de grafos, no siempre se puede reconstruir el grafo original a partir de un grafo de líneas, por ende tampoco la red original. Esto es debido exclusivamente a que se almacena las rutas en términos de aristas en vez de hacerlo en términos de vértices, es decir, en función de links en vez de routers respectivamente.

Dado que Simgrid2, no guarda información total acerca de la red física subyacente en el archivo *platform*, se decidió utilizar otro formato de archivo para el almacenamiento de la misma. En este contexto surge la necesidad de contar con un archivo de descripción de red propio de GridMatrix (archivo de extensión `.net`), que guarda la descripción de la red física. A partir de este archivo es posible generar de manera automática el archivo *platform* (necesario para la ejecución de una simulación en Simgrid2).

El archivo de descripción de red (`.net`) almacena nodos, routers y conexiones entre ellos. En términos de grafos, los nodos y routers corresponden a vértices, y las conexiones corresponden a aristas entre vértices. En la figura 10 se ilustra un ejemplo de la forma en que es almacenada esta información en GridMatrix2.

A continuación se detallan los aspectos más relevantes de la generación de redes.

Interfaz gráfica para la generación de redes

A fin de integrar la funcionalidad de generación automática de redes en GridMatrix2, se diseñó una interfaz gráfica que se ilustra en la figura 11.

Las opciones disponibles pueden ser clasificadas en tres grupos. En primer lugar (a) las opciones referentes a las características de la topología de red (Network Options). En segundo lugar (b) la información referente a las características de los elementos propios de la red (SimGrid Options). Por último, (c) opciones de carácter general.

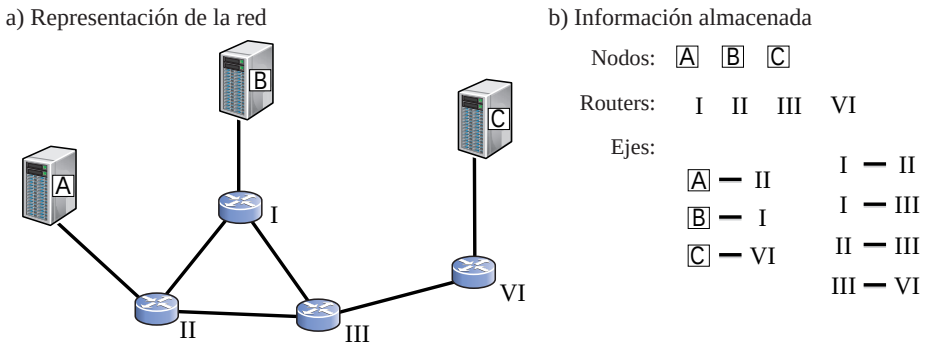


Figura 10: Ejemplo de la información almacenada en un archivo de GridMatrix2 en términos de nodos, routers y conexiones entre ellos.

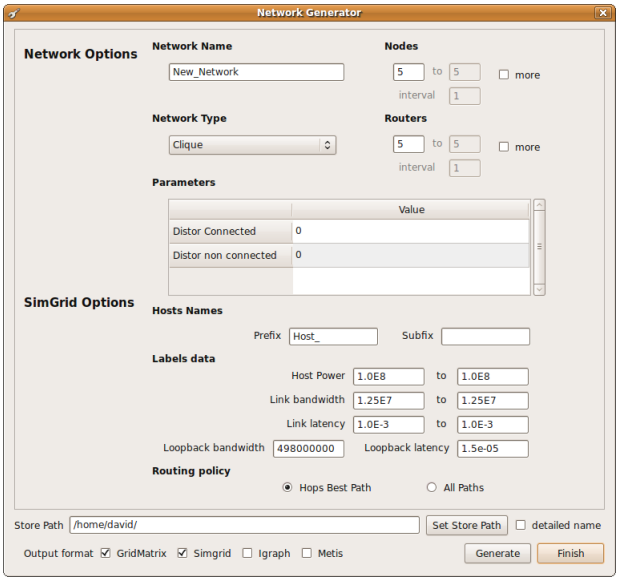


Figura 11: Interfaz Gráfica del Generador de Redes de GridMatrix2

(a) Dentro de las opciones de red, se debe indicar un nombre, que será tanto el nombre de la red, como el nombre del archivo a generar. A continuación se debe seleccionar el tipo de topología de la red.

Una vez configurada la topología y el nombre de la red, se deberá indicar el tamaño de la misma. Este último punto es medido en términos de cantidad de nodos y routers. Adicionalmente, es posible generar una serie de redes variando el tamaño automáticamente. Para esto, se puede optar por la opción *more*. En este caso se deberá indicar los valores límite y el intervalo, tanto para routers como para nodos.

Finalmente, si para generar alguna topología es necesario ingresar algún parámetro específico, se provee la lista de ellos con la posibilidad de editarlos desde la interfaz. Por ejemplo, en el caso de la topología Exponencial, posee un parámetro propio: el exponente de la red.

(b) Dentro del bloque de opciones referentes a SimGrid, se puede observar tres grupos. El primero permite indicar los nombres que tendrán los nodos, indicando para esto un prefijo y un sufijo. Los nombres de los nodos se forman a partir de estos valores y números correlativos. El segundo grupo configura las características de los elementos de la red, tanto links como nodos. Proporcionando valores uniformes para ancho de banda, latencia y capacidad de cómputo. La capacidad de cómputo (power) se mantiene por cuestiones de compatibilidad, ya que sólo es utilizada para construir el archivo *platform* para SimGrid2. Por último, se encuentra la opción de ruteo. Como fue explicado con anterioridad, el ruteo en SimGrid2 es estático. A partir de esto, se decidió generar dos clases de ruteos: el primero generando solo los caminos mínimos entre todos los nodos; el segundo, listando todos los caminos posibles entre los nodos.

(c) Dentro de las opciones de carácter general, en el extremo inferior de la pantalla, se cuenta con cuatro items que permiten indicar el formato en que se desea generar la red. Las opciones posibles son, GridMatrix, SimGrid, Igraph y Metis. Las primeras dos opciones corresponden a archivos en formato XML, utilizados como entrada para GridMatrix2 (`.net`) y como archivo *platform* para SimGrid2 respectivamente. Las dos opciones restantes corresponden a archivos en texto plano que pueden ser utilizados como entradas para Igraph y Metis. Ambos formatos pueden ser utlizados para exportar las redes generadas y permitir utilizarlas en otros programas, por ejemplo, con el fin de caracterizarlas.

Además, se puede apreciar un campo de texto. En dicho campo es obligatorio ingresar el directorio donde se guardarán las redes generadas. A su derecha se presenta la opción *detailed name*, que permite al momento de guardar el resultado de la generación, adicionar en el nombre del archivo los parámetros de cantidad de routers y cantidad de nodos con los que cuenta la red generada.

En la siguiente sección se detalla todo lo referente a la generación de las distintas topologías de red.

Topologías de red

En términos generales, la política para la generación se basa en construir una red entre los routers de forma adecuada, respetando la topología y los parámetros especificados. Una vez construida dicha red, se procede a agregar los nodos a la misma.

La adición de nodos a la red se puede realizar ordenadamente o de forma aleatoria. En el primer caso, se ordenan todos los routers y se agrega a cada uno, en orden, todos los nodos de la red. De esta manera pueden darse tres situaciones: a) si la cantidad de routers es menor que la cantidad de nodos,

entonces habrá routers que tengan más nodos que otros; b) si la cantidad de routers es mayor que la cantidad de nodos, existirán routers que no tengan ningún nodo asignado; y c) si la cantidad de routers es la misma que la de nodos, cada router tendrá un solo nodo asociado.

La otra forma en la que pueden ser agregados nodos a una red, es de forma aleatoria. En este caso, por cada nodo se selecciona uniformemente un router a cual asociarlo.

Las topologías soportadas en esta implementación son Clique, Line, Ring, Star, Uniform, Exponential, Waxman, Zegura y Barabasi. En el capítulo anterior fueron descriptas las características de cada una de ellas. A continuación resta algunos detalles referentes al desarrollo.

Para las topologías básicas, Clique, Line, Ring y Star se optó por, una vez generadas, los nodos sean agregados de forma ordenada. Esta decisión se debe a que este tipo de topologías se utilizan de forma teórica y se desea poder conocer exactamente cuál es el resultado de la generación de la red. Para las restantes topologías implementadas, la opción fue que los nodos sean agregados de forma aleatoria a la red. Adicionalmente se provee un mecanismo para alterar las redes básicas, asignando un porcentaje de links que serán agregados o borrados.

Por otro lado, se busca que la red generada sea conexa, es decir que exista un camino entre todo par de nodos de la red. Los algoritmos Uniform, Exponential, Waxman, Zegura y Barabasi no garantizan este punto. Por esta razón se decidió iterar una cantidad finita de veces hasta encontrar una red conexa; solución que se comparte con el generador de redes Simulacrum. A nivel de interfaz, en el caso de no encontrar una red adecuada, se le indica al usuario que se iteró una cantidad determinada de veces y que aún así no se pudo encontrar una red satisfactoria. Además, en el mismo mensaje, se le indica que los parámetros para la generación no son seguros.

La generación de la red se podría dar por terminada, si no fuera porque la misma debe ser presentada en una interfaz gráfica. Para analizar este punto se invita a leer la siguiente sección.

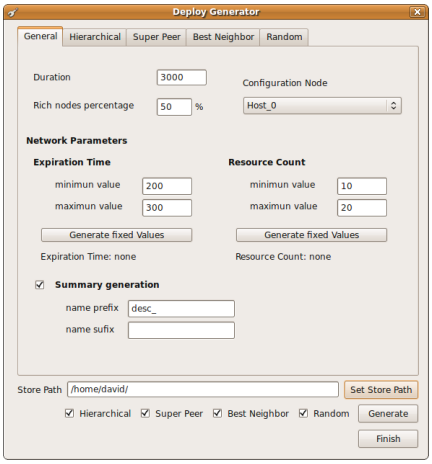
6.4. Configuración de simulaciones

El segundo archivo requerido para SimGrid2 es el archivo *deploy* cuya función principal es la de definir los procesos que serán ejecutados en cada nodo, los resultados que deberán ser generados y el tiempo total de ejecución.

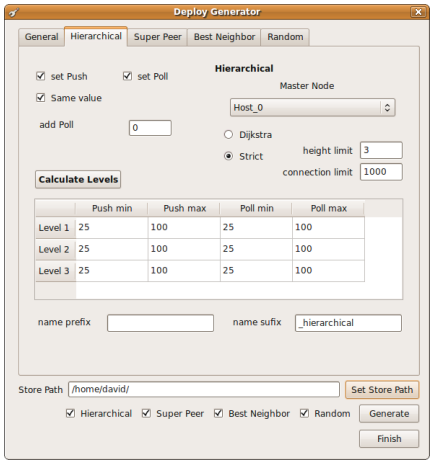
Interfaz gráfica de generación de Deploy

A fin de integrar la funcionalidad de generación automática de archivos de deploy en GridMatrix2, se desarrolló la interfaz gráfica que puede verse en la figura 12.

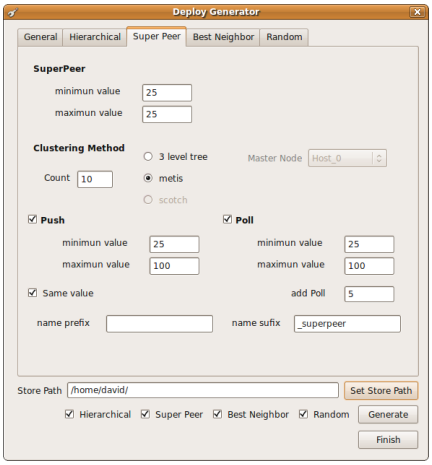
En el gráfico se observan varias pantallas, las cuales se encuentran integradas en la pantalla general de generación de archivos de *deploy*. Cada una de éstas se corresponde con un conjunto de opciones relacionadas a las políticas implementadas de distribución de información. Seleccionando cualquiera de las cuatro opciones posibles en el extremo inferior de cada una de las pantallas (Hierarchical, Super Peer, Best Neighbor y Random), se define finalmente cuáles serán



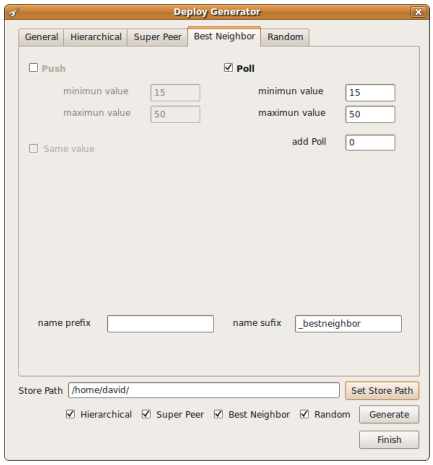
(a) General



(b) Jerárquico



(c) Super-Peer



(d) Best Neighbor

Figura 12: Interfaz Gráfica del Generador de Deploy. Se omite la pantalla destinada a la política *random* dado que la misma es equivalente a la que se muestra en la política *best-neighbor*

los archivos a generar. En el cuadro de texto "Store Path" se debe indicar, de manera obligatoria, cuál será el directorio donde se almacenarán los resultados.

En la primer pantalla, denominada *General* (figura 12a), se encuentran los parámetros generales para todas las políticas. Se debe completar el tiempo de duración de la corrida y el nombre del nodo que será asignado para ejecutar la configuración inicial. Cabe recordar, tal como se detallara en la primer sección del presente capítulo, que en GridMatrix2 existe un solo proceso que se encarga de configurar la política de distribución de información en todos los nodos.

Además se deben completar dos parámetros: *resource count* y *expiration time*. Ambos son intervalos a partir de los cuales se determinan los valores, de manera uniforme, del parámetro a utilizar en la política de propagación. Para el caso de *resource count*, sólo se aplica a un porcentaje de nodos, determinado mediante el parámetro *rich nodes percentage*. El resto de los nodos serán configurados con la mínima cantidad de recursos posible.

Cabe destacar que tanto el parámetro *resource count* como *expiration time*, son definidos de forma independiente y por única vez. Esto permite generar un conjunto de instancias de archivos de *deploy* con exactamente los mismos valores para cada nodo, siempre y cuando no sean regenerados dichos valores.

El resto de las pantallas permiten seleccionar parámetros específicos de cada una de las políticas. Si bien hay parámetros que figuran simultáneamente en todas, pueden ser configurados de forma diferente. Un ejemplo de esto es el prefijo (*name prefix*) y sufijo del nombre de salida de los archivo (*name sufix*).

Las políticas Jerárquica y Super-Peer son las que requieren especial atención, ya que necesitan información adicional ligada a la topología de red.

Para la política Jerárquica se requiere crear una jerarquía entre los nodos. Este trabajo no es para nada trivial y será explicado en detalle más adelante. Dentro de la pantalla de la figura 12b puede visualizarse un campo que indica la manera en que será creada la jerarquía y cuál será el nodo utilizado como raíz de la misma. La interfaz permite sólo dos opciones para crear la jerarquía: a) libre (es el caso de *Dijkstra*) o limitada (el caso de *strict*). En el caso que sea limitada, se deben completar dos parámetros: *altura* y *conectividad límite*. La *altura* se refiere a la cantidad de niveles totales que tendrá el árbol de la jerarquía. A modo de ejemplo, si el nodo más lejano se encuentra a 5 niveles del nodo raíz, y el límite seleccionado es 4, entonces todos los nodos pertenecientes al nivel 5 no serán configurados en la jerarquía, sino que figurarán como nodos que no corren ningún proceso. En el caso del parámetro *conectividad límite*, a un padre no se le podrá asociar en la jerarquía más de la cantidad de hijos que sea indicado como límite.

Dentro de esta pantalla, también deberá configurase los valores de *push* y *poll* para cada uno de los nodos. En el caso de la política jerárquica, poder configurar de forma diferente estos valores para cada uno de los niveles de la jerarquía, es algo casi indispensable. Por esta razón se cuenta con una tabla donde se enumeran cada uno de los niveles de la jerarquía. Para generarla es necesario hacer clic sobre el botón *Calculate Levels*. La generación se realiza teniendo en cuenta el nodo seleccionado como *Master Node*; en el caso de cambiar la elección de dicho nodo, se deberá regenerar. Esta decisión fue tomada en base a que calcular la cantidad de niveles que tendrá una jerarquía es muy costoso para redes de gran tamaño.

La pantalla correspondiente a la política Super-Peer se puede observar en la figura 12c. Los valores de *push* y *poll* son configurados de forma genérica para todos los nodos de la red. A través del parámetro *SuperPeer* se define el rango a partir del cual se hará una distribución uniforme y se asignará a cada super-peer el tiempo de intercambio de información entre ellos. La selección de cuáles nodos

son designados como super-peers se configura a través de las opciones *clustering method*. Figurando tres opciones: a) árbol de tres niveles (*3 level tree*), b) *metis* y c) *scotch*. La primer opción aplica un algoritmo que será descripto más adelante; el mismo se encarga de armar un árbol jerárquico y tomar los primeros nodos de esa jerarquía como super-peers. Las dos opciones restantes del menú permiten indicar el programa que será utilizado como particionador. Los detalles referentes a este mecanismo serán descriptos más adelante.

Las configuraciones para las políticas *random* y *best-neighbor* son muy similares. Sólo se deben configurar los valores de *push* y *poll*. Un punto a destacar sobre todas las políticas, es que la configuración de los valores de *push* y *poll* puede ser o no habilitadas, de forma que una política solo responda a uno de los dos eventos. Adicionalmente, puede configurarse que los valores de *push* y *poll* tengan el mismo valor, mediante la opción *same value*. La opción *add Poll* permite sumar un valor constante al valor asignado a *Poll*. Todas estas opciones abren un abanico de posibilidades para la configuración de los valores en que se generarán para los eventos de *push* y *poll*.