Sistemas HTC sobre redes P2P

Carlos Pérez-Miguel, Jose Miguel-Alonso y Alexander Mendiburu
Dept. de Arquitectura y Tecnología de Computadores
Universidad del País Vasco / Euskal Herriko Unibertsitatea
{carlos.perezm, j.miguel, alexander.mendiburu}@ehu.es

Resumen

En este artículo hacemos una revisión de los sistemas peer-to-peer, desde el punto de vista de su potencial para dar soporte a sistemas de computación de alta productividad. Presentamos una propuesta que sirve de base para la construcción de un sistema de este tipo, basada en una infraestructura P2P de almacenamiento fiable distribuido, sobre la que se implementa un sistema de colas al que se pueden enviar trabajos.

1. Introducción

El desarrollo durante la última década de los sistemas Peer-to-Peer (P2P) en los que todos sus integrantes cumplen el mismo papel, ha llamado la atención del mundo académico sobre sus posibles aplicaciones prácticas. Desde la creación de los primeros sistemas de intercambio de archivos mediante P2P, se han propuesto diversas aplicaciones, tales como sistemas de almacenamiento masivo, sistemas de indexación de nombres, sistemas de mensajería instantánea e incluso voz sobre IP.

Una de las aplicaciones más interesantes de las redes P2P es el campo de la computación de alta productividad, o High Throughput Computing (HTC). Ya existen sistemas de HTC como Condor [19] o Boinc [1], ampliamente utilizados, pero que adolecen de un problema: al requerir un punto central de gestión, pueden tener problemas de escalabilidad o de tolerancia a fallos. Un sistema de HTC que usara las características de las redes P2P para auto-administrarse y poner en contacto a sus integrantes entre si podría solucionar estos problemas. Por ello, en este artículo proponemos un modelo para la creación de un sistema de HTC sobre redes P2P.

El resto del articulo está estructurado de la siguiente forma: En la Sección 2 presentamos una visión general de los diferentes sistemas Peer-to-Peer que se han propuesto. En la Sección 3 introducimos unas ideas generales sobre sistemas de computación sobre redes P2P, así como de las características que deberían incluir; también presentamos algunos de los sistemas que se han propuesto en la literatura para explotar este paradigma. En la Sección 4 mostramos nuestra propuesta de un sistema de HTC sobre redes P2P. Por ultimo en la Sección 5 exponemos las conclusiones de este trabajo.

2. Redes P2P

Los sistemas P2P son sistemas distribuidos en los cuales no existe ningún tipo de control central ni de estructura jerárquica entre sus integrantes. En un sistema P2P todos los nodos que forman parte de él poseen el mismo rol; estos nodos se conectan entre si a través de algún tipo de red formando una red virtual a nivel de aplicación, también llamada *overlay*, que es usada para encaminar mensajes entre los nodos con el objetivo de buscar información, compartir recursos o permitir que los usuarios se comunique entre sí.

A diferencia de los sistemas de Grid Computing [13], las redes P2P no se basan en la interconexión de grupos organizados, unidos entre si por redes más o menos fiables. A par-

tir de un gran número de usuarios no fiables con roles similares unidos entre si por conexiones no fiables son capaces de ofrecer una serie de características interesantes en un sistema distribuido, como por ejemplo, escalabilidad, tolerancia a fallos, búsqueda eficiente de información, almacenamiento redundante, persistencia o anonimato.

Según Lua et al. [21], existen dos tipos básicos de *overlays*: estructurados y no estructurados. Los overlays no estructurados están formados por nodos conectados entre si de forma aleatoria. Al no existir ninguna estructura, se utiliza un protocolo de encaminamiento denominado *por inundacion*, es decir, haciendo que cada nodo reenvíe cada mensaje recibido a cada uno de sus vecinos excepto aquel que le envió el mensaje. De esta forma es posible alcanzar cualquier punto del sistema, a costa de una penalización en la eficiencia que se agrava con el tamaño del mismo.

Cuando hablamos de overlays estructurados, nos referimos a overlays en los que la topología de la red y el contenido son controladas, para situarlas en ubicaciones no aleatorias; de manera que faciliten su búsqueda. Estos sistemas P2P estructurados basan su funcionamiento en la implementación de una Tabla Hash Distribuida (Distributed Hash Table o DHT) [12, 26], en la cual la información es almacenada de forma determinista en el nodo o nodos cuyo identificador corresponda con la clave del objeto a almacenar, siendo dicha clave única en el sistema.

En un sistema de DHT se considera un espacio de claves de una longitud determinada y se mapea dicho espacio de claves en la red de nodos que forman la red P2P. Dicho mapeo asigna a cada uno de los nodos un identificador perteneciente al espacio de claves para luego dividir el espacio de claves en N fragmentos, asignando cada fragmento al nodo cuyo identificador este contenido en dicho fragmento. De esa forma, cada clave del DHT será mantenida por un sólo nodo en la red.

El uso de una DHT nos permite crear una red que soporte el almacenamiento escalable y distribuido de información en la forma de pares {clave, valor} pudiendo definir una interfaz de uso similar a la de cualquier tabla hash. Dicha interfaz nos proporciona una serie de funciones para acceder a la información a través de una clave, tales como funciones que permitan la inserción/modificación de claves *(put(key, value))* o el acceso a las mismas *(value=get(key))*. Estas funciones implican el uso de un cierto protocolo de encaminamiento, oculto al usuario, para hacer llegar dichas peticiones al nodo correspondiente a dicha clave.

El protocolo de encaminamiento varía en cada sistema, pero todos ellos cumplen una característica en cuanto a la forma de llevar un mensaje hacia un nodo destino: el encaminamiento se realiza de forma progresiva en función de la distancia al destino. De esta forma, cuando un nodo desea enviar, o encaminar, un cierto mensaje a un nodo identificado por la clave k, enviará el mensaje a aquel de entre sus vecinos que más cerca se encuentre del destino del mensaje. Dicha propiedad de cercanía entre un nodo y una clave del DHT varía entre sistemas, ya que dependerá de la forma de organizar el espacio de claves y de la estrategia de encaminamiento. En teoría, los sistemas basados en DHTs garantizan que cualquier objeto puede ser localizado en un numero de saltos de orden $O(logN)$, siendo N el número de nodos que forman el sistema. Uno de los puntos débiles de los sistemas basados en DHTs es su comportamiento ante el ingreso y salida masiva de nodos en la red en un momento dado. La latencia puede variar de forma considerable en estos casos, y por ello se han realizado propuestas para minimizar dichos efectos. En [24] se muestra un algoritmo para alcanzar la latencia casi óptima en grafos que muestran leyes de potencial, como las redes P2P que nos ocupan, sin perder las propiedades de encaminamiento escalable que poseen los DHTs; en [10] se especifica una solución para mantener un cierto equilibrio de carga bajo condiciones adversas.

3. Computación sobre redes P2P

Desde el surgimiento de las redes P2P se ha debatido mucho sobre la posibilidad de usar este paradigma de sistema distribuido como base para montar un sistema de compu-

tación [8]. Un sistema de computación distribuido se puede definir como un conjunto de computadores interconectados entre si por una red de comunicaciones, que intentan unir sus recursos para llevar a cabo algún tipo de tarea computacional. Cada computador conectado al sistema posee sus propios recursos computacionales independientes; sin embargo, desde el punto de vista del usuario, el sistema se percibe como un único ente. En dicho sistema un usuario accedería a los recursos dispersos de la misma forma en que accede a sus recursos locales. Sin embargo, el hecho de que el sistema esté repartido entre múltiples entidades le puede aportar mejoras sustanciales en cuanto a escalabilidad y tolerancia a fallos.

Sistemas que cumplan estas condiciones hay muchos, desde *clusters* hasta sistemas de *Grid Computing*, pasando por sistemas de *Desktop Grid*. Entre todos estos sistemas podemos destacar algunas soluciones ya citadas, como Globus [9] en sistemas de Grid, Condor [19] en sistemas de Clustering, o Boinc [1] en cuanto a sistemas de Desktop Grid. Todos estos sistemas, sin embargo, poseen algo en común: todos están centralizados en algún punto. Esta centralización, que se realiza para facilitar tareas de administración y mantenimiento, o para garantizar el cumplimiento de diversas políticas, debilita al sistema en su comportamiento en cuanto a tolerancia a fallos o escalabilidad, ya que el punto central se convierte en el eslabón más débil de la cadena.

Un sistema de computación mediante redes P2P supera esta desventaja al otorgar a sus integrantes las labores administrativas y al darles iguales derechos ante el resto de los nodos de la red.

En la literatura técnica podemos encontrar diversas propuestas de sistemas de computación usando redes P2P, algunas de las cuales resumimos a continuación.

WaveGrid: Propuesto en [20, 29, 30] consiste en un overlay DHT en el que los nodos se dividen en husos horarios, de tal forma que en cada momento de tiempo, sólo se están usando aquellos nodos que se encuentren en una zona donde sea de noche y los nodos se encuentren inactivos. En el momento en que un huso horario deja de estar en una zona nocturna, se procede a migrar la tarea hacia otra zona horaria que se encuentre en horario nocturno. Cuando un nodo desea lanzar un trabajo, mediante un proceso de planificación elige, de entre los nodos disponibles, aquellos que se ajusten a las condiciones que él imponga para el trabajo, y de entre esos elige el nodo más apropiado según sus criterios. Estos sistemas tienen el problema de que es posible que nunca llegue a encontrarse un nodo libre; tampoco se mantiene ningún tipo de orden entre las tareas que se lanzan en el sistema.

P2P task scheduling in computation grid: Propuesto en [4], es un intento por descentralizar parte de la infraestructura de un sistema de Grid Computing como Globus. Mediante un nuevo módulo llamado Peer in Grid Scheduler (PGS) permite que los nodos del sistema sean los encargados de decidir como ejecutar las tareas que deseen lanzar. Cada nodo es responsable de la planificación de sus tareas y para ello se usa el Grid Peer Information Service (GPIS), unas paginas amarillas centralizadas del sistema de grid con objeto de encontrar nodos disponibles y en caso de encontrarlos lanzar las tareas en dicho nodo. Sin embargo, no existe ningún tipo de acuerdo en común entre los nodos para establecer un orden entre dichas tareas.

CompuP2P: CompuP2P [11] es un sistema P2P basado en DHT que divide el conjunto de los nodos del sistema entre compradores y vendedores de recursos. En este sistema se elige a un nodo encargado del mercado de recursos, sobre el que recaen las tareas de planificación.

CoDiP2P: Desarrollado por la Universidad de Lleida, CoDiP2P [22] está basado en una red P2P estructurada en forma de árbol pero que no usa DHT. Basa su funcionamiento en mapear en dicha estructura en árbol los nodos del sistema, diferenciando por regiones entre nodos maestros y nodos esclavos. Cuando un nodo desea lanzar una tarea, se pone en contacto con el nodo maestro de su zona, el cual

busca los nodos libres de la zona y decide la mejor forma de ejecutar la tarea. En el caso de que no exista ningún nodo libre en su zona, buscará en las zonas adyacentes. Este sistema se separa de las bases de los sistemas P2P debido a la centralización que impone a sus nodos. Aunque en este caso si se asegura la ejecución de todas las tareas que se lancen en el sistema, esto se garantiza mediante la utilización de servidores centralizados.

Super-peer approach for public scientific computation: Propuesto en [23] intenta unificar los conceptos de Grid Computing y P2P Computing. Para ello propone una aproximación ya usada por Gnutella [27]: redes P2P de *Super-Peers* que están al cargo de una serie de nodos Workers conectados mediante el modelo Cliente-Servidor y encargados de ejecutar las tareas que el Super-Peer considere apropiadas dada su carga y características. Por otro lado todos los Super-Peers del sistema conforman un overlay P2P mediante el cual se informan entre ellos de los trabajos y de los nodos disponibles con los que cuentan.

Sin embargo, uno de los problemas de los sistemas P2P propuestos hasta ahora para computación es que en ninguno de ellos se garantiza la ejecución de una tarea, ya que si el sistema se encuentra sobrecargado, no existe ningún mecanismo que asegure que la tarea se terminará por ejecutar en algún momento del tiempo de vida del sistema. En la mayoría de estos sistemas, si el nodo que desea lanzar una tarea es incapaz de encontrar un recurso disponible, nunca se lanzaran las tareas de dicho nodo. Es más, al no existir ningún tipo de orden entre las tareas que se ejecutan en el sistema, nadie garantiza que todas las tareas lanzadas en el sistema se ejecuten tarde o temprano. Por esta razón, en los sistemas de computación distribuida habitualmente existen sistemas de colas encargados de mantener un cierto orden temporal en las tareas que se ejecutan en el sistema, de tal forma que todas las tareas se ejecuten tarde o temprano. La implementación sobre un entorno P2P de un sistema de colas permitiría la ejecución en orden de tareas, así como el uso de diversas técnicas de planificación.

Se podría construir dicho sistema de colas usando las características de almacenamiento de objetos de una DHT. De esta forma la información necesaria para realizar la planificación de tareas en orden estaría distribuida entre todos los integrantes del sistema. Sin embargo, antes de que eso sea posible, es necesario dotar al sistema de persistencia en el almacenamiento de información. En la literatura existen múltiples propuestas [16, 5, 6, 14, 28, 2, 15] para convertir un overlay P2P basado en DHT en un sistema de almacenaje distribuido que asegure la persistencia y la consistencia de los objetos almacenados, usando para ello múltiples replicas de dichos objetos, así como distintos algoritmos para mantener la coherencia entre dichas replicas.

Sobre este sistema de almacenamiento con replicas y consistencia entre las mismas es posible crear grupos en la DHT que mantengan un objeto "cola de procesos" de forma distribuida y mediante la cual se mantenga el orden de ejecución de las tareas que se lancen en el sistema. Los nodos libres del sistema que se ofrezcan voluntarios para ejecutar alguna tarea modificarían la cola de forma atómica, para extraer de la misma la primera tarea que puedan ejecutar; la selección se hará dependiendo de la política de ejecución usada.

Además, este sistema permitiría que un cierto nodo almacenase en la DHT la información de la tarea que desease lanzar en la cola distribuida, conjuntamente con los datos necesarios para lanzarla (ejecutables, librerías, datos de configuración, etc...). Así, aunque el nodo se desconectara del sistema y su tarea sería eventualmente ejecutada sin su colaboración. Cuando este nodo volviese a entrar al sistema encontraría almacenados en la DHT los resultados de la ejecución que solicitó.

4. Propuesta de un sistema de HTC basado en redes P2P

A partir de un sistema de almacenamiento distribuido fiable basado en DHTs, como Oceanstore [17, 25], PAST [7] o Cassandra [18], pro-

ponemos construir un sistema de computación de alta productividad (HTC) basado en un sistema de colas que permita la ejecución en orden de las tareas lanzadas al sistema. Las tareas que se lancen en el sistema serán ejecutadas siguiendo un orden global no estricto, siguiendo una filosofía *best-effort*; es decir, que el orden de extracción de las tareas a ejecutar sera el mismo de entrada en la cola, pero sin garantías estrictas.

En este sistema cada usuario estará identificado mediante un sistema de claves publica/privada [3], lo que permitirá firmar digitalmente los objetos que éstos almacenen en el sistema. Así mismo se implementará un sistema de *karma* o de reputación que permita al usuario puntuar las acciones del resto de los usuarios.

A continuación se detallan los objetos que se almacenarán en el sistema junto con una propuesta para nombrarlos:

- Cola: Puede ser global, es decir, que almacena todas las tareas del sistema; o de cierto tipo, es decir que almacena tareas que cumplen alguna condición, como por ejemplo, tareas que necesitan un sistema operativo determinado o que precisan de cierta arquitectura. Una cola está caracterizada por los siguientes elementos:
  - Nombre, permite localizar la cola en el sistema y acceder a ella. De tipo: "System::Queues::IDCola"
  - Tamaño máximo de la cola: el número de elementos contenidos en la misma, pudiendo ser ilimitado o no.
  - Una lista con el nombre y las características de cada uno de los trabajos de la cola.
- Tareas: Cada tarea contendrá información necesaria para la ejecución de un trabajo, y a su vez apuntará a una serie de objetos necesarios para o producidos por la ejecución del mismo:
  - Nombre del trabajo: "System::Tasks::IDUsuario::IDTarea".
  - Arquitectura requerida.
  - Software requerido.
  - Tiempo límite para ejecutar la tarea en el caso de que éste exista.
  - Tiempo estimado para ejecutar la tarea.
  - Script que guíe la ejecución de la tarea.
  - ID del usuario o usuarios que están ejecutando actualmente el trabajo.
  - Estado de la tarea en el sistema: en espera, en ejecución, parada, cancelada, finalizada.
  - Referencias a los siguientes objetos:
    - Datos necesarios para la tarea.
    - Ejecutables.
    - Resultados.
  - Tipo. Un trabajo puede ser de uno de estos dos tipos:
    - Simple, una sola tarea.
    - Compuesto, una serie de repeticiones de una misma tarea con distintas configuraciones de datos de entrada o una serie de subtareas que deben ser ejecutadas en un cierto orden.
- Ejecutables+librerías: Asociado a un trabajo, este objeto permite la ejecución de una tarea. Nombre: "System::Tasks::IDUser::IDTarea::Binary".
- Datos de configuración: Datos adicionales necesarios para ejecutar una tarea. Nombre: "System::Tasks::IDUser::IDTarea::Data".
- Resultados de una tarea: Nombre: "System::Tasks::IDUser::IDTarea::Result".
- Información de los usuarios:
  - ID del usuario: su clave pública.
  - Karma del usuario.
  - Lista de trabajos en ejecución.

En el sistema se pueden realizar las siguientes acciones:

- Lanzar una tarea: Acción iniciada por el usuario; éste almacenará en el sistema los archivos necesarios para la ejecución de la tarea (objeto tarea) con la información de los objetos necesarios para ejecutarla (incluyendo ejecutables y datos). Todos estos objetos se almacenarán en el DHT y estarán firmados con la clave privada del usuario. Si se introduce alguno de estos objetos en el sistema sin su correspondiente firma, serán eliminados del mismo.

- Administrar una tarea propia: El usuario podrá acceder en todo momento al estado de la tarea y consultar con el nodo que la ejecute el estado de la misma. Así mismo podrá parar, reanudar o cancelar la ejecución de la tarea en cualquier momento.

- Ejecutar una tarea: Todo nodo puede aceptar ejecutar una tarea en cualquier momento, siendo esta decisión tomada automáticamente por el software de planificación del propio nodo, y por lo tanto transparente al usuario. Para ello el nodo cambiará el estado de la misma a "En ejecución", recolectará los objetos necesarios (ejecutables y datos) y ejecutará la tarea siguiendo el script correspondiente, almacenado en el objeto tarea. Una vez termine la ejecución del mismo, creará un objeto con los resultados de la ejecución, firmado con su clave privada, y pasará el estado de la tarea a "Terminada". En este punto se pueden seguir dos estrategias para informar al dueño de la tarea de que su ejecución ha terminado: o bien se le notifica directamente mediante algún sistema de mensajería o bien es el software de planificación del nodo dueño de la tarea el encargado de monitorizar periódicamente mediante encuesta el estado de la tarea.

- Terminar una tarea: Acción iniciada por el software de planificación del nodo dueño de la tarea. Una vez la ejecución haya terminado, el nodo accederá al sistema para eliminar los objetos sobrantes, recolectar los resultados de la ejecución y puntuar al nodo que ejecutó la tarea.

Este modelo cubre solamente la funcionalidad que hemos considerado básica para un sistema de HTC. Sin embargo, creemos que el planteamiento permite la adición de otras características deseables, como son por ejemplo el uso de sistemas de *Checkpointing* que, junto con mecanismos de migración, permitirían que las tareas en ejecución en el sistema pudieran ser suspendidas en un nodo y retomarlas en otro distinto. Esta mejora es deseable en un entorno en el que sus integrantes adolecen de una gran volatilidad. También se podría plantear la ejecución de tareas paralelas mediante MPI, lo cual precisaría de algún mecanismo de co-planificación para encontrar el conjunto de nodos libres más apropiado, por cercanía, para la ejecución de este tipo de tareas paralelas.

5. Conclusiones

En el presente articulo se ha presentado una revisión de las propuestas existentes de sistemas de computación de alto rendimiento sobre redes P2P. Así mismo se hace una propuesta de HTC sobre redes P2P usando para ello las capacidades de almacenamiento distribuido de los sistemas basados en tablas hash distribuidas (DHT). Actualmente estamos implementando un sistema como el aquí descrito sobre un DHT real como Cassandra. Además, estamos trabajando con simuladores de sistemas P2P, lo cual nos permitirá verificar nuestro modelo sobre distintos sistemas de DHT.

Agradecimientos

Este trabajo ha sido posible gracias al apoyo de los programas Saiotek y Research Groups 2007-2012 (IT-242-07) del Gobierno Vasco, los proyectos TIN2007-68023-C02-02, TIN2008-06815-C02-01 y Consolider Ingenio 2010 - CSD2007-00018 del Ministerio Español de Ciencia e Innovación y por la red COMBIOMED de biomedicina computacional (Instituto de la Salud Carlos III). Así mismo, Carlos Pérez-Miguel disfruta de una beca predoctoral concedida por el Gobierno Vasco.

Referencias

[1] D. P. Anderson. Boinc: A system for public-resource computing and storage. In R. Buyya, editor, *GRID*, pages 4–10. IEEE Computer Society, 2004.

[2] S. Bessa, M. E. Correia, and P. Brandao. Storage and retrieval on P2P networks: A DHT based protocol. In *2007 IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, VOLS 1-3*, pages 829–835, 345 E 47TH ST, NEW YORK, NY 10017 USA, 2007. IEEE; IEEE Commun Soc; IEEE Comp Soc; Inst Telecommun; Univ Aveiro; Inovacao; AT&T, IEEE. IEEE Symposium on Computers and Communications, Santiago, PORTUGAL, JUL 01-04, 2007.

[3] N. L. Brisola, A. O. Santin, L. C. Lung, H. B. Ribeiro, and M. H. Vithoft. A Public Keys Based Architecture for P2P Identification, Content Authenticity and Reputation. *2009 International Conference on Advanced Information Networking and Applications Workshops*, pages 159–164, 2009.

[4] J. Cao, O. M. K. Kwong, X. Wang, and W. Cai. A peer-to-peer approach to task scheduling in computation grid. In M. Li, X.-H. Sun, Q. Deng, and J. Ni, editors, *GCC (1)*, volume 3032 of *Lecture Notes in Computer Science*, pages 316–323. Springer, 2003.

[5] B.-G. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, and R. Morris. Efficient replica maintenance for distributed storage systems. In *NSDI*. USENIX, 2006.

[6] F. Dabek, M. F. Kaashoek, D. R. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with cfs. In *SOSP*, pages 202–215, 2001.

[7] P. Druschel and A. Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. *Proc. HotOS VIII*, 2001.

[8] I. T. Foster and A. Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In M. F. Kaashoek and I. Stoica, editors, *IPTPS*, volume 2735 of *Lecture Notes in Computer Science*, pages 118–128. Springer, 2003.

[9] I. T. Foster and C. Kesselman. The globus project: A status report. In *Heterogeneous Computing Workshop*, pages 4–18, 1998.

[10] B. Godfrey, K. Lakshminarayanan, S. Surana, R. M. Karp, and I. Stoica. Load balancing in dynamic structured p2p systems. In *INFOCOM*, 2004.

[11] R. Gupta, V. Sekhri, and A. K. Somani. Compup2p: An architecture for internet computing using peer-to-peer networks. *IEEE Trans. Parallel Distrib. Syst.*, 17(11):1306–1320, 2006.

[12] D. R. Karger, E. Lehman, F. T. Leighton, R. Panigrahy, M. S. Levine, and D. Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *STOC*, pages 654–663, 1997.

[13] C. Kesselman and I. Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, November 1998.

[14] K. Kim and D. Park. Reducing replication overhead for data durability in dht based p2p system. *IEICE Transactions*, 90-D(9):1452–1455, 2007.

[15] K. Kiw and D. Park. Reducing replication overhead for data durability in DHT based P2P system. *IEICE TRANSACTIONS ON INFORMATION AND SYSTEMS*, E90D(9):1452–1455, SEP 2007.

[16] P. Knezevic, A. Wombacher, and T. Risse. Enabling high data availability in a dht. In *DEXA Workshops*, pages 363–367. IEEE Computer Society, 2005.

[17] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels,

R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, and Others. Oceanstore: An architecture for global-scale persistent storage. *ACM SIGARCH Computer Architecture News*, 28(5):190–201, 2000.

[18] A. Lakshman and P. Malik. Cassandra-A Decentralized Structured Storage System. *cs.cornell.edu*, 2009.

[19] M. Litzkow, M. Livny, and M. Mutka. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, June 1988.

[20] V. M. Lo, D. Zappala, D. Zhou, Y. Liu, and S. Zhao. Cluster computing on the fly: P2p scheduling of idle cycles in the internet. In G. M. Voelker and S. Shenker, editors, *IPTPS*, volume 3279 of *Lecture Notes in Computer Science*, pages 227–236. Springer, 2004.

[21] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7(1-4):72–93, 2005.

[22] D. C. Martínez, J. R. Torrento, I. B. Vilardell, F. G. de Sola, and F. Solsona. A new reliable proposal to manage dynamic resources in a computing p2p system. In D. E. Baz, F. Spies, and T. Gross, editors, *PDP*, pages 323–329. IEEE Computer Society, 2009.

[23] C. Mastroianni, P. Cozza, D. Talia, I. Kelley, and I. Taylor. A scalable superpeer approach for public scientific computation. *Future Gener. Comput. Syst.*, 25(3):213–223, 2009.

[24] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory Comput. Syst.*, 32(3):241–280, 1999.

[25] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: the OceanStore prototype. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 1–14, 2003.

[26] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[27] The Gnutella Protocol Specification. http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.

[28] J. Zhao, H. Yu, K. Zhang, W. Zheng, J. Wu, and J. Hu. Achieving reliability through replication in a wide-area network dht storage system. In *ICPP*, page 29. IEEE Computer Society, 2007.

[29] D. Zhou and V. M. Lo. Wave scheduler: Scheduling for faster turnaround time in peer-based desktop grid systems. In D. G. Feitelson, E. Frachtenberg, L. Rudolph, and U. Schwiegelshohn, editors, *JSSPP*, volume 3834 of *Lecture Notes in Computer Science*, pages 194–218. Springer, 2005.

[30] D. Zhou and V. M. Lo. Wavegrid: a scalable fast-turnaround heterogeneous peer-based desktop grid system. In *IPDPS*. IEEE, 2006.