ISBN: 978–972–8924–72-0 © 2008 IADIS

EXPERIMENTOS SOBRE LA CARGA DE RED EN SISTEMAS DISTRIBUIDOS DE RECUPERACIÓN DE INFORMACIÓN

Fidel Cacheda Seijo, Diego Fernández Iglesias y Rafael López García
Departamento de Tecnologías de la Información y las Comunicaciones, Universidade da Coruña
Facultad de Informática, Campus de Elviña s/n, 15071 A Coruña, Spain

RESUMEN

El empleo de sistemas distribuidos aporta un importante beneficio en cuanto al rendimiento en el ámbito de la Recuperación de Información. En dichos sistemas, existen diversas estrategias y arquitecturas para repartir la carga computacional entre varias máquinas, de forma que cada una de estas soluciones suele mejorar ciertos aspectos de rendimiento, pero también presenta factores negativos y cuellos de botella. No obstante, gran parte de estas soluciones han sido estudiadas desde un punto de vista teórico y sólo han sido probadas mediante experimentos basados en simulación.

En el presente artículo se emplea un sistema distribuido de Recuperación de Información (RI) para la realización de un experimento sobre la carga de red generada por los diversos elementos del sistema en base a varias configuraciones, con el objetivo de determinar si existe un impacto en el tiempo de ejecución al modificar el número de servidores de consultas o el número de resultados parciales generados por los mismos.

PALABRAS CLAVES

Sistemas de información distribuidos, recuperación de información.

1. INTRODUCCIÓN

El tamaño de las colecciones de datos manejadas por los buscadores actuales y el alto nivel de concurrencia que éstos deben soportar hacen que estos sistemas resulten no ser escalables y se sobrecarguen [10]. Sin embargo, el número de documentos que forman parte de las colecciones de datos aumenta día a día, a la vez que el usuario se vuelve más exigente en cuanto al tiempo de respuesta del buscador. Por ende, se hace necesario buscar una solución a dicho problema.

En los sistemas de búsqueda distribuidos la carga se reparte entre todos los nodos que los conforman. Generalmente, en dichos sistemas existen dos protagonistas: los mediadores o *brokers* y los servidores de consultas (SCs). Los primeros se encargan de recibir las consultas de los usuarios y distribuirlas a los distintos SCs, mientras que los segundos son los responsables de procesar las consultas y enviar sus resultados parciales a los *brokers*. Otra función de los *brokers* es combinar y reordenar dichos resultados parciales para generar un resultado final que es el que se le devuelve al usuario.

Los ficheros de índice se pueden distribuir entre los distintos SCs del sistema mediante dos estrategias clásicas: una conocida como *term partitioning* o ficheros invertidos globales, y otra conocida como *document partitioning* o ficheros invertidos locales [16] [19]. La segunda estrategia es empleada con mayor frecuencia en las soluciones comerciales, ya que es muy sencillo distribuir las colecciones de documentos de manera uniforme entre todos los SCs y generar índices locales en los mismos. A la hora de procesar una consulta, cada uno de los SCs recibe todos y cada uno de los términos que forman parte de la misma y selecciona el conjunto de documentos resultante, que será disjunto con respecto a los conjuntos generados por el resto de SCs.

Los trabajos previos de Cacheda et al. [6] [5] [7] identificaban dos cuellos de botella en los sistemas distribuidos con ficheros invertidos locales: el *broker* y la red de comunicaciones. Estos trabajos se basan en simulaciones en las cuales se presupone un entorno ideal. En dichos artículos también se proponen ciertas

soluciones, como la reducción de resultados parciales enviados por cada SC o la arquitectura de *brokers* distribuidos.

No obstante, Badue et al. [7] concluyen que si el entorno no es ideal (y en la práctica la mayoría de las veces no lo es), los tiempos de respuesta de cada uno de los SCs pueden diferir bastante entre sí aunque los SCs repartan los documentos de una forma homogénea. A esta variación se le conoce como *imbalance* o desequilibrio en el rendimiento de los SCs.

El presente artículo tiene como objetivo describir el experimento escogido para determinar si el número de resultados parciales enviados por cada SC repercute en el rendimiento global de sistema, y para comprobar si existe o no un *imbalance* o desequilibrio en el tiempo de ejecución de los diferentes servidores de consultas. Aparte de la medición de los propios tiempos de ejecución con una cantidad variable de SCs, el recuento del número de bytes de datos trasmitidos por la red permite la realización de un estudio más detallado.

Este artículo está estructurado de la siguiente forma: en la Sección 2 se mencionan los trabajos más importantes de la materia. En la Sección 3 se describe el entorno empleado para los experimentos. En la Sección 4 se describen los experimentos realizados en nuestro entorno real y en la Sección 5 se discuten los resultados obtenidos. Por último, en la Sección 6, se hace una presentación de las conclusiones obtenidas y se enumeran aquellas ideas que pueden dar lugar a futuros trabajos.

2. ESTADO DEL ARTE

Nuestro interés en el área de Recuperación de Información (RI) se centra en los sistemas distribuidos, de forma que a continuación presentaremos los trabajos más relevantes en la materia.

Para empezar, la escalabilidad de este tipo de sistemas ha sido estudiada mediante simulación por Cahoon y McKinley [8] llegando a conclusiones muy alentadoras de cara a una posible demostración práctica.

En cuanto a las estrategias de distribución de índices, es posible encontrar un análisis comparativo en trabajos con cierta madurez temporal, como los de Tomasic y García-Molina [19] e incluso en otros más modernos como los de MacFarlane et al. [13] y Badue et al. [4]. Por su parte, Moffat et al. presentan en un artículo más actual [14] ciertas técnicas para la mejora del rendimiento de sistemas distribuidos basados en *term partitioning*. El trabajo de Ribeiro-Neto y Barbosa [16] evalúa estas estrategias, pero también muestra otros parámetros que afectan al rendimiento en la resolución de consultas en sistemas distribuidos.

Como ya hemos mencionado con anterioridad, varios artículos de Cacheda et al. [6] [5] [7] identifican dos cuellos de botella en este tipo de sistemas: el *broker* y la red de comunicaciones. Sus trabajos proponen además una serie de soluciones para mejorar el rendimiento en ambos puntos, tales como la reducción de resultados parciales o los *brokers* distribuidos.

Por su parte, en Badue et al. [3] se realiza un análisis del *imbalance* que se puede encontrar en los sistemas distribuidos de RI para demostrar que en entornos reales también afecta al rendimiento. Los tres factores básicos que provocan este *imbalance* son el número de SCs, la cantidad de memoria de los mismos y el uso de caché de disco. Dicho desequilibrio fuerza al sistema a reducir su velocidad conforme al más lento de los SCs.

Por último, Lin y Zhou [12] y Hawking [9] han sido de los primeros en publicar trabajos en base a experimentos en entornos no simulados.

Las principales diferencias entre nuestro artículo y los anteriormente mencionados son que los experimentos que relatamos han sido llevados a cabo en un entorno de prueba con hardware real, que empleamos colecciones públicas de datos y de consultas (la colección de consultas es además gratuita), y que hacemos especial incidencia en el cuello de botella de la red de comunicaciones, identificado en los trabajos de Cacheda et al. [6] [5] [7].

3. ENTORNO DE PRUEBAS

3.1 El sistema de Recuperación de Información

El sistema de RI distribuido que hemos diseñado e implementado para realizar los experimentos es una extensión del motor de búsqueda Terrier [17], desarrollado en la Universidad de Glasgow [21]. Entre las características más destacables de Terrier hay que mencionar que es Open Source, está escrito en Java, soporta un lenguaje de consulta bastante convencional, incorpora un buen número de modelos de ranking de documentos y está preparado para trabajar con los formatos de fichero más comunes y con colecciones de datos especialmente creadas para investigación, como las de TREC Terabyte Track [20].

A la vez que es imprescindible conservar la mayoría de características de Terrier, también es importante cumplir dos objetivos en el diseño que extiende el sistema: modificar la menor parte posible del código original y facilitar la adición de nuevas funcionalidades al entorno distribuido.

Para cumplir el primer objetivo, hemos optado por emplear una distribución de índices conforme a la estrategia de ficheros invertidos locales, ya que la estrategia de ficheros invertidos globales nos forzaría a modificar la estructura de índices de Terrier. Las únicas modificaciones que ha sufrido el código original de Terrier han sido las necesarias para la medición de rendimiento.

El segundo objetivo se ha cumplido a través de un diseño en capas (figuras 1 y 2). En cada capa pueden existir diversas implementaciones que resuelven el mismo problema, diferenciándose éstas únicamente en ciertas características. Una factoría se encarga de instanciar la implementación más adecuada, de forma que el usuario puede elegirla simplemente cambiando un parámetro de configuración. Como ejemplo más notable, señalaremos que es posible seleccionar el protocolo de transporte que empleará el sistema distribuido, de manera que el usuario escoge entre TCP o UDP, y en el segundo caso es posible sacar provecho del Multicast o bien renunciar a esta posible ventaja.

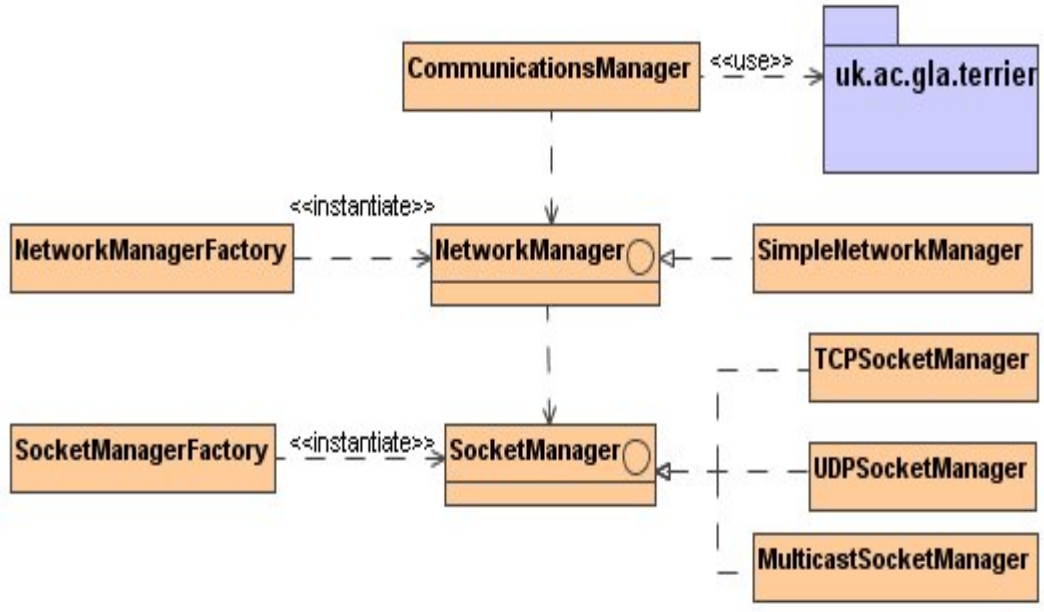

Figura 1. Arquitectura de los SCs.

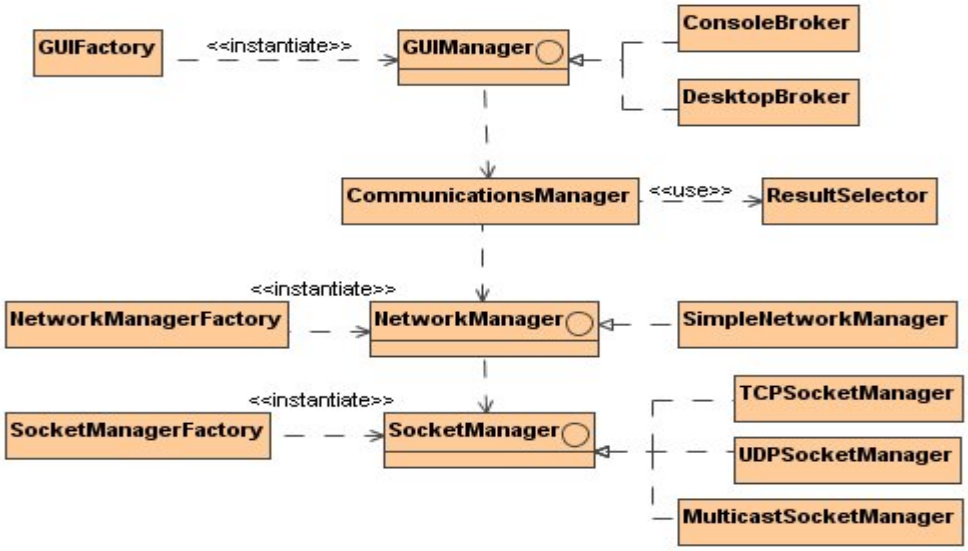

Figura 2. Arquitectura del broker.

Ambas partes del sistema comparten 3 capas. La primera es *CommunicationsManager*, encargada de la lógica general de la aplicación. La segunda es *NetworkManager*, concebida para posibles operaciones de red, como la partición de paquetes en la capa de aplicación, aunque por el momento sólo se ha realizado una implementación trivial. Por último, *SocketManager* se encarga de la transferencia de paquetes a través de sockets. El *broker* dispone a mayores de una interfaz gráfica de usuario.

El sistema también permite establecer una jerarquía de *brokers*. Para ello, simplemente hay que cubrir un parámetro de configuración en cada uno para indicarle quién es su nodo padre. No obstante, es necesario indicar en este punto que los experimentos que hemos realizado no explotan esta característica, de forma que los diversos SCs empleados están conectados a un mismo *broker*. Sin embargo, esperamos emplear esta característica en futuros trabajos.

3.2 Colecciones de Documentos y de Consultas

Los estudios de Cacheda et al. [6] [5] [7] se apoyan en una simulación de la colección SPIRIT (94.552.870 documentos y 1 TB de texto) [11]. Por otra parte, el estudio realizado por Badue et al. [3] emplea una colección de documentos WBR03, compuesta de 10 millones de páginas, coleccionadas de la web brasileña por el motor de búsqueda TodoBR [18] en 2003. El conjunto de consultas empleado está compuesto por 100.000 consultas extraídas de un log parcial enviado al motor de búsqueda TodoBR en septiembre de 2003.

En el caso que concierne al presente artículo, la colección de documentos sobre la que hemos realizado los experimentos es .GOV [1], la cual forma parte del TREC Terabyte Track [20], está compuesta por 1.247.753 documentos pertenecientes a dominios .gov y es distribuida por la Universidad de Glasgow [21]. Estos documentos fueron coleccionados a principios del 2002 empleando el hardware y la red del NIST [15].

En cuanto a las consultas, hemos utilizado la colección 2006 Efficiency Task Topics de TREC 2006 Terabyte Track [2]. Esta colección está compuesta de 100.000 consultas, pero sólo hemos considerado un subconjunto de 10.000.

3.3 Entorno Hardware

Los experimentos se han realizado en una red de comunicaciones Fast Ethernet a 100 Mbps.

El *broker* se ha instalado en un PC con procesador Intel Pentium 4 a 3,2 GHz con HyperThreading, 1 GB de memoria RAM y sistema operativo Linux Ubuntu.

Las máquinas que soportan los SCs tienen las siguientes características técnicas: procesador Intel Pentium 4 a 2,6 GHz con HyperThreading, 512 MB de memoria RAM y sistema operativo Linux Ubuntu Server.

4. EXPERIMENTOS

El experimento consiste en ejecutar cada una de las 10.000 consultas escogidas de la colección 2006 Efficiency Task Topics [2] y tomar una serie de tiempos en el *broker* y en cada uno de los SCs. A la vez, se contará el número de bytes de datos enviados por la red, así como el número de resultados parciales emitidos por cada SC y los que el *broker* guarda finalmente. Los documentos serán distribuidos uniformemente entre los diversos SCs siguiendo la estrategia de *document partitioning.*

Para cada consulta, se asume que el sistema distribuido no sólo recupera los *n* mejores resultados con igual precisión que su contrapartida centralizada, sino que según el caso peor, cada SC también recupera los *n* mejores resultados parciales. Éstos se combinan en el *broker* de forma que está garantizado que finalmente se ofrecerán los *n* mejores resultados al usuario.

Todo este procedimiento se repetirá para un número variable de SCs: 1, 2 y 4 respectivamente y para TCP y UDP con y sin Multicast. Además se programarán las consultas 2 veces: una indicando al sistema que cada SC debe devolver un máximo de 10 resultados parciales y otra indicando un máximo de 100 resultados.

5. RESULTADOS

A continuación se muestran unas tablas con los resultados de la ejecución de las consultas en las distintas configuraciones del sistema (tablas 1, 2 y 3). Los tiempos se expresan en milisegundos y los tamaños en bytes.

Tabla 1. Mediciones para 1 SC.

	Máximo 10 resultados parciales			Máximo 100 resultados parciales		
	TCP	UDP	Multi-cast	TCP	UDP	Multi-cast
Media del tiempo de ejecución en el broker (ms)	277,4	271,57	269,52	277,05	272,22	280,03
Desviación típica del tiempo de ejecución en el broker (ms)	177,67	177,02	177,07	177,75	177,94	184,98
Media del tiempo de ejecución en el SC 1 (ms)	277,09	271,28	269,27	276,55	271,74	279,59
Desviación típica del tiempo de ejecución en el SC 1 (ms)	177,67	177,02	177,07	177,73	177,92	184,96
Media del tiempo de envío de resultados (ms)	0,31	0,28	0,25	0,5	0,48	0,44
Desviación típica del tiempo de envío de resultados (ms)	0,08	0,08	0,07	0,16	0,16	0,15
Media del mínimo tiempo de red (ms)	0,31	0,28	0,25	0,5	0,48	0,44
Desviación típica del mínimo tiempo de red (ms)	0,08	0,08	0,07	0,16	0,16	0,15
Media de bytes enviados por el broker (ms)	38,24	38,19	38,14	38,27	38,2	38,22
Desviación típica de bytes enviados por el broker	12,04	12,05	12,07	12,07	12,11	12,06
Media bytes enviados por el SC 1	134,1	134,17	134,22	915,06	914,89	914,87
Desviación típica de bytes enviados por el SC 1	34,16	34,07	33,99	565,37	565,44	565,47
Media de resultados enviados por el SC 1	9,01	9,01	9,02	64,79	64,78	64,78
Desviación típica de resultados enviados por el SC 1	2,44	2,43	2,43	40,38	40,39	40,39

Las filas en las cuales se indique número de bytes enviados se refieren a número de bytes de datos, contabilizados desde el propio código fuente y, por tanto, libres de cabeceras de red y transporte. En el caso de TCP y UDP se considera que hay un envío por cada SC, mientras que con Multicast el mismo envío se dirige a todos los SCs.

Se entiende por tiempo de envío de resultados aquél comprendido entre el comienzo del primer envío y la finalización del último envío. A su vez, el mínimo tiempo de red es el tiempo empleado por el servidor de consultas más rápido en enviar sus datos.

Para cada una de las configuraciones se han calculado los percentiles 2 y 98 con el objetivo de eliminar resultados desproporcionados que introduzcan un sesgo en la evaluación de los datos obtenidos, por ejemplo, los afectados aspectos de la máquina virtual Java (carga de clases bajo demanda, etc.). Esta criba también se realiza en base a la proporción entre los tiempos de red y los bytes enviados, de forma que se eliminan aquellos resultados en los cuales la red ha tardado mucho tiempo en pasar una cantidad de bytes muy baja o viceversa. Por ello, es lógico que el número de bytes enviados sea ligeramente diferente en protocolos como TCP y UDP, ya que aunque el número de bytes enviados es el mismo en la misma consulta, el tiempo de red no lo es, por lo que podemos estar eliminando un cierto resultado para un protocolo de transporte pero no para otro.

Tabla 2. Mediciones para 2 SCs.

	Máximo 10 resultados parciales			Máximo 100 resultados parciales		
	TCP	UDP	Multi-cast	TCP	UDP	Multi-cast
Media del tiempo de ejecución en el broker (ms)	140,81	141,88	149,94	142,15	148,04	143,27
Desviación típica del tiempo de ejecución en el broker (ms)	95,2	95,94	98,13	95,05	94,03	97,14
Media del tiempo de ejecución en el SC 1 (ms)	140,34	141,33	149,27	141,52	147,33	142,41
Desviación típica del tiempo de ejecución en el SC 1 (ms)	95,16	96,01	98,33	95	94,03	97,22
Media del tiempo de ejecución en el SC 2 (ms)	127,49	127,36	132,11	127,4	128,62	127,23
Desviación típica del tiempo de ejecución en el SC 2 (ms)	11,1	12,08	16,21	11,65	14,06	16,15
Media del tiempo de envío de resultados (ms)	13,55	14,83	18,3	14,95	19,71	16,32
Desviación típica del tiempo de envío de resultados (ms)	11,19	12,17	16,32	11,64	14,01	13,81
Media del mínimo tiempo de red (ms)	0,25	0,25	0,21	0,44	0,44	0,58
Desviación típica del mínimo tiempo de red (ms)	0,08	0,08	0,07	0,18	0,17	0,94
Media de bytes enviados por el broker	38,26	38,2	38,22	38,14	38,09	38,15
Desviación típica de bytes enviados por el broker	12,37	12,34	12,35	12,35	12,32	12,32
Media bytes enviados por el SC 1	129,87	129,87	129,87	856,8	856,8	856,8
Desviación típica de bytes enviados por el SC 1	39,08	39,08	39,08	583,2	583,2	583,2
Media bytes enviados por el SC 2	129,9	129,9	129,9	854,55	854,55	854,55
Desviación típica de bytes enviados por el SC 2	39,09	39,09	39,09	581,89	581,89	581,89
Media de resultados enviados por el SC 1	8,87	8,92	8,9	62,71	62,9	61,89
Desviación típica de resultados enviados por el SC 1	2,58	2,52	2,54	40,8	40,75	41,14
Media de resultados enviados por el SC 2	8,82	8,8	8,87	61,27	61,08	61,34
Desviación típica de resultados enviados por el SC 2	2,64	2,66	2,57	41,05	41,1	41,15
Imbalance (%)	4,80	5,20	6,10	5,25	6,78	5,63

Tabla 3. Mediciones para 4 SCs.

	Máximo 10 resultados parciales			Máximo 100 resultados parciales		
	TCP	UDP	Multi-cast	TCP	UDP	Multi-cast
Media del tiempo de ejecución en el broker (ms)	69,81	70,3	70,42	73,05	73,84	73,07
Desviación típica del tiempo de ejecución en el broker (ms)	35,93	36,72	36,64	48,88	49,82	49,01
Media del tiempo de ejecución en el SC 1 (ms)	66,25	68,52	68,54	71,38	71,36	71,34
Desviación típica del tiempo de ejecución en el SC 1 (ms)	32,7	35,72	35,51	48,08	48,11	48,15
Media del tiempo de ejecución en el SC 2 (ms)	67,15	65,67	66,51	70,43	71,92	70,75
Desviación típica del tiempo de ejecución en el SC 2 (ms)	34,78	33,5	33,9	47,69	49,28	47,99
Media del tiempo de ejecución en el SC 3 (ms)	64	62,36	62,53	67,81	68,04	67,69
Desviación típica del tiempo de ejecución en el SC 3 (ms)	32,05	30,55	30,54	45,14	45,38	45,26
Media del tiempo de ejecución en el SC 4 (ms)	55,62	55,47	55,49	60,37	60,26	60,11
Desviación típica del tiempo de ejecución en el SC 4 (ms)	26,89	26,89	26,94	39,87	39,99	39,92
Media del tiempo de envío de resultados (ms)	4,64	4,82	4,62	5,8	6,33	5,88
Desviación típica del tiempo de envío de resultados (ms)	4,42	4,73	4,59	5,73	6,34	5,78
Media del mínimo tiempo de red (ms)	0,25	0,28	0,21	0,44	0,47	0,37
Desviación típica del mínimo tiempo de red (ms)	0,07	0,03	0,05	0,22	0,19	0,19
Media de bytes enviados por el broker	38,31	38,15	38,24	38,16	38,18	38,17
Desviación típica de bytes enviados por el broker	12,35	12,3	12,33	12,23	12,33	12,32
Media bytes enviados por el SC 1	131,96	130,71	132,2	811,1	810,72	817,76
Desviación típica de bytes enviados por el SC 1	36,63	37,95	36,19	587,57	587,58	586,56
Media bytes enviados por el SC 2	131,43	131,83	132,04	837,15	837,23	840,46
Desviación típica de bytes enviados por el SC 2	37,2	36,76	36,41	582,28	582,17	581,64
Media bytes enviados por el SC 3	128,49	129,03	128,73	833,33	833,68	833,67
Desviación típica de bytes enviados por el SC 3	39,83	39,21	39,52	583,16	582,88	583,04
Media bytes enviados por el SC 4	128,41	128,66	128,67	825,08	824,99	824,17
Desviación típica de bytes enviados por el SC 4	39,98	39,62	39,59	582,95	582,87	583,13
Media de resultados enviados por el SC 1	8,85	8,77	8,87	57,36	57,34	57,84
Desviación típica de resultados enviados por el SC 1	2,62	2,71	2,58	41,97	41,97	41,9
Media de resultados enviados por el SC 2	8,82	8,85	8,86	59,22	59,23	59,46
Desviación típica de resultados enviados por el SC 2	2,66	2,63	2,6	41,59	41,58	41,55
Media de resultados enviados por el SC 3	8,61	8,65	8,62	58,95	58,98	58,98
Desviación típica de resultados enviados por el SC 3	2,84	2,8	2,82	41,65	41,63	41,65
Media de resultados enviados por el SC 4	8,6	8,62	8,62	58,36	58,36	58,3
Desviación típica de resultados enviados por el SC 4	2,86	2,83	2,83	41,64	41,63	41,65
Imbalance (%)	12,07	11,95	12,29	10,56	11,25	10,91

Un punto a destacar es que el *imbalance* se acentúa a medida que crece la cantidad de servidores.

A pesar de que hemos empleado solamente 4 servidores, se puede comparar cómo la distribución de documentos dio lugar al decremento del tiempo global de ejecución a medida que se incrementó el número de máquinas. Esto se puede deducir de la tabla 4, cuyos tamaños están expresados en megabytes.

Tabla 4. Índices.

	1 SC	2 SCs	4 SCs
Gramática (MB)	114,39	75,09	47,30
Índice directo (MB)	441,60	223,91	106,61
Índice invertido (MB)	454,09	233,71	112,76
Índice de documentos (MB)	46,17	24,37	12,55

La carga de red generada con un número tan reducido de servidores de consultas no parece suficiente para impactar en el rendimiento global del sistema, puesto que independientemente de si el sistema gestiona uno, dos o cuatro SCs, el tiempo de red permanece constante. Ésta parece ser la misma razón por la que los datos obtenidos según los diferentes protocolos de transporte son tan semejantes entre sí.

6. CONCLUSIÓN

En este artículo hemos presentado un sistema distribuido de recuperación de información para la elaboración de diversos experimentos en la materia. Este sistema podría ser instalado en una red de área local o un clúster con un buen número de máquinas, con la finalidad de corroborar mediante experimentos realistas que ciertas técnicas de optimización de RI, tales como la eliminación de resultados parciales o la distribución de *brokers*, realmente son efectivas en la práctica y que los resultados de ciertos experimentos (i.e. los trabajos de Cacheda et al. [6][5][7]) son similares a los obtenidos en un entorno real.

El modelo también se ha diseñado pensando en su posible extensión y adición de nuevas funcionalidades, a fin de estudiar otras vías de optimización, tales como un mejor aprovechamiento de la red de comunicaciones (por ejemplo, particionando los paquetes en la capa de aplicación, modificando o suprimiendo cabeceras, creando un protocolo de red ad-hoc, etc.).

El número medio de resultados enviados por cada SC sólo decrece ligeramente cuanto mayor es el número de SCs. Para 1 SC ronda los 65 resultados, para 2 SCs ronda los 61 resultados y para 4 SCs ronda los 58 resultados. Este hecho se debe a que, incluso disponiendo de 4 SCs, en gran parte de las consultas casi todos devuelven 100 resultados, que es el número máximo permitido. Como cabe esperar, el mismo aumento se produce en el número de bytes de datos enviados por cada servidor.

También hemos corroborado que sea cual sea el número de resultados parciales obtenidos y el número de servidores de consultas empleados, el *imbalance* y los tiempos totales de ejecución en el broker y en los SCs (independientemente del protocolo de transporte empleado) siguen siendo similares.

Los principales datos que han cambiado entre 10 y 100 resultados parciales son el número de bytes enviados por los SCs y el tiempo de envío de datos por la red, en el que se nota un ligero aumento con 4 SCs.

Otro factor que hay que destacar es que, sea cual sea el número de resultados parciales obtenidos, para un número reducido de SCs existe un desequilibrio entre los tiempos de ejecución de los mismos, y éste va en aumento a medida que el número de SCs se incrementa, tal y como se señalaba en el trabajo que previamente habían realizado Badue et al. [3].

REFERENCIAS

GOV collection. http://ir.dcs.gla.ac.uk/test_collections/govinfo.html.

2006 Efficiency Task Topics (TREC 2006 Terabyte Track), a.k.a. TREC Efficiency Track http://trec.nist.gov/data/terabyte/06/06.efficiency_topics.tar.gz.

Badue, C.S. et al. 2007. Analyzing imbalance among homogeneous index servers in a web search system. *Information Processing and Management Journal,* Vol. 43 No. 3, pp. 592-608.

Badue, C.S. et al. 2005. Basic Issues on the Processing of Web Queries. *Proc. of the 28th International ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'05)*, pp. 577–578.

Cacheda, F. et al. 2005. A Case Study of Distributed Information Retrieval Architectures to Index One Terabyte of Text. *Information Processing and Management Journal*, Vol. 41 No. 5, pp. 1141-1161.

Cacheda, F. et al. 2004. Performance Analysis of Distributed Architectures to Index One Terabyte of Text. *Proc. Of 26th European Conf. on Information Retrieval Research (ECIR'04)*, Lecture Notes on Computer Science No. 2997, pp. 394-408.

Cacheda, F. et al. 2007. Performance Analysis of Distributed Information Retrieval Architectures Using an Improved Network Simulation Model. *Information Processing and Management Journal*, Vol. 43 No. 1, pp. 204-224.

Cahoon, B. and McKinley, K.S. 1996. Performance evaluation of a distributed architecture for information retrieval. *Proc. of 19th ACM-SIGIR International Conf. on Research and Development in Information Retrieval*, pp. 110-118.

Hawking, D. 1997. Scalable text retrieval for large digital libraries. *Lecture notes in computer science*, No. 1324, pp. 127–146.

Hawking, D. and Thistlewaite, P. 1999. Methods for Information Server Selection. *ACM Transactions on Information Systems*, Vol. 17 No. 1, pp. 40-76.

Jones, C.B. et al. 2002. Spatial information retrieval and geographical ontologies an over-view of the SPIRIT project. *Proc. of the 25th ACM-SIGIR Conf. on Research and Development in Information Retrieval*, pp. 387-388.

Lin, Z. and Zhou, S. 1993. Parallelizing I/O intensive applications for a workstation cluster: a case study. *ACM SIGARCH Computer Architecture News*, Vol. 21 No. 5, pp. 15–22.

MacFarlane, A. et al. 2000. Parallel Search using Partitioned Inverted Files. *Proc. of the 7th International Symposium on String Processing and Information Retrieval (SPIRE'00)*, pp. 209–220.

Moffat, A. et al. 2006. A pipelined architecture for distributed text query evaluation. *Information Retrieval*, published on-line.

National Institute of Standards and Technology (NIST). http://www.nist.gov.

Ribeiro-Neto, B. and Barbosa, R. 1998. Query performance for tightly coupled distributed digital libraries. *Proceedings of the 3rd ACM Conference on Digital Libraries*. New York: ACM Press, pp. 182-190.

Terrier (TERabyte RetrIEveR). http://ir.dcs.gla.ac.uk/terrier/.

TodoBR (2003). http://www.todobr.com.br.

Tomasic, A. and García-Molina, H. 1993. Performance of inverted indices in shared-nothing distributed text document information retrieval systems. *Proc. of the 2nd International Conf. on Parallel and Distributed Information Systems*, pp. 8-17.

TREC Terabyte Track. http://www-nlpir.nist.gov/projects/terabyte/.

University of Glasgow. http://www.gla.ac.uk/.