Scheduling Queries Across Replicas

Ana Freire[1], Craig Macdonald[2], Nicola Tonellotto[3], Iadh Ounis[2], Fidel Cacheda[1]

[1] University of A Coruña, Campus de Elviña s/n, 15017 A Coruña, Spain
[2] University of Glasgow, G12 8QQ Glasgow, UK
[3] National Research Council of Italy, Via G. Moruzzi 1, 56124 Pisa, Italy

{ana.freire, fidel.cacheda}@udc.es[1], {craig.macdonald, iadh.ounis}@glasgow.ac.uk[2], {nicola.tonellotto}@isti.cnr.it[3]

ABSTRACT

For increased efficiency, an information retrieval system can split its index into multiple shards, and then replicate these shards across many query servers. For each new query, an appropriate replica for each shard must be selected, such that the query is answered as quickly as possible. Typically, the replica with the lowest number of queued queries is selected. However, not every query takes the same time to execute, particularly if a dynamic pruning strategy is applied by each query server. Hence, the replica's queue length is an inaccurate indicator of the workload of a replica, and can result in inefficient usage of the replicas. In this work, we propose that improved replica selection can be obtained by using query efficiency prediction to measure the expected workload of a replica. Experiments are conducted using 2.2k queries, over various numbers of shards and replicas for the large GOV2 collection. Our results show that query waiting and completion times can be markedly reduced, showing that accurate response time predictions can improve scheduling accuracy and attesting the benefit of the proposed scheduling algorithm.

Categories & Subject Descriptors: H.3.3 [Information Storage & Retrieval]: Information Search & Retrieval

General Terms: Performance, Experimentation

Keywords: Query Scheduling, Simulation

1. INTRODUCTION

A distributed information retrieval (IR) system consists of several query servers, each of them storing the index *shard* for a subset of the documents in the corpus. New queries arrive at the *broker*, which routes them to each shard, before collating and merging the results for presentation to the user. The efficiency of each query server can be improved by deploying a dynamic pruning strategy, such as WAND [1], which aims to avoid the scoring of postings for documents that cannot make the top K retrieved set.

While multiple shards increase efficiency compared to a monolithic ("single shard") retrieval system, the throughput of a distributed retrieval system can be further enhanced by *replicating* shards, so that one of multiple query servers can provide the results for a single shard [6]. The problem tackled in this work is how a broker should select (*schedule*) the most suitable replica of a given shard in order to reduce the queue waiting time. For example, the replica with the minimum number of queued queries can be selected.

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.
ACM 978-1-4503-1472-5/12/08.

However, the response time for different queries can vary widely, particularly if dynamic pruning is employed [5]. Hence, the accurate choice of replica is made more difficult, as the number of queries queued by a given query server does not accurately predict the processing backlog of the server. A recently proposed technique for *query efficiency prediction* [5] offers a plausible manner to estimate the workload of a replica. Hence, we hypothesise that query efficiency prediction [4] can permit accurate query scheduling in a distributed/replicated IR system. Indeed, to the best of our knowledge, this work contributes the first study to applying query efficiency predictors for scheduling in a distributed/replicated IR system. Using a simulated distributed/ replicated search environment, based on actual query response times, we experiment to determine how different scheduling algorithms can be deployed for replica selection. Our results show that by using query efficiency prediction, we can improve the selection of replicated query servers, and hence the average query completion times are reduced. For instance, using predicted response time to select between 4 replicas of a 2 shard index results in a 42% reduction in mean completion time compared to selecting replicas by considering only the length of their queues.

2. SIMULATING REPLICATION

In comparing different scheduling algorithms, we experiment with a various numbers of shards and replicas. To facilitate such experiments without exhaustive hardware resources, we build a simulation framework that supports different distributed settings. Indeed, Cacheda et al. [2] showed that a simulation framework could accurately model the efficiency of a real distributed IR system, including the network delays, the queue waiting and processing time for queries and the time for merging the results. Following this, we implement a simulation framework[1], extended to encapsulate the presence of multiple shards each with multiple replicas as well as several scheduling algorithms for selecting replicas. The constants for network delays follow [2], in order to achieve a proven realistic simulation environment.

The baseline scheduling algorithms implemented for selecting replicas are: Random (the replica is chosen randomly); Round Robin (modulo the number of replicas, if replica i was selected for this query, replica $i+1$ is used for the next query); Queue Length (the replica with the fewest queries waiting to be processed is selected). In addition to these baselines, we propose Predicted, where the replica with the current shortest queue in terms of predicted response times is selected. We use query efficiency predictions [5]

[1]Built using JavaSim: `http://javasim.codehaus.org/`

Replicas	Random		Round Robin		Queue Length		Prediction		*Oracle*	
	ACT	AWT	ACT	AWT	ACT	AWT	ACT	AWT	*ACT*	*AWT*
2 Shards										
2	9617	9382	10061	9826	8897	8662	**613**	**362**	*610*	*359*
4	902	667	409	174	434	199	**253**	**3**	*253*	*3*
8	410	175	263	28	428	193	**250**	**0**	*250*	*0*
5 Shards										
2	375	237	241	103	247	109	**158**	**4**	*159*	*5*
4	265	126	155	16	231	93	**154**	**0**	*154*	*0*
8	192	54	**140**	2	231	93	154	**0**	*154*	*0*
10 Shards										
2	168	69	120	22	145	47	**114**	**1**	*114*	*1*
4	139	41	**101**	3	144	46	114	**0**	*114*	*0*
8	123	25	**98**	**0**	144	46	114	**0**	*114*	*0*

Table 1: ACTs and AWTs (in milliseconds) for different settings and scheduling algorithms.

for estimating the response time of a query. Moreover, as the predicted response time is dependent on statistics of the query terms on that index shard, our framework accounts for the time to calculate the prediction at the selected replica and transmit it back to the broker, such that the expected workload of the replica can be updated.

Finally, as the selection of replicas is based on predicted response times, we additionally implement an Oracle scheduling algorithm, which knows the actual response time of a query before it is executed, but still accounts for the calculating the predicted response time. In this way, Oracle represents a best-case scenario for Predicted scheduling.

3. EXPERIMENTAL SETUP

We hypothesise that using predicted response times can increase overall efficiency compared to other scheduling algorithms. To address this hypothesis, we conduct experiments by indexing TREC GOV2 corpus using Terrier[2], applying Porter's English stemmer and removing standard stopwords. We experiment with three different index configurations: 2, 5 and 10 shards. For retrieval on each query server, we use a set of 2200 queries of the TREC 2005 Terabyte track Efficiency task. We sample real arrival times of a set of queries from an Excite query log and assign them to our TREC queries (query arrival rates vary from 20 to 180 per second). We use the WAND dynamic pruning strategy [1] to retrieve $K = 1000$ documents, scored by the DPH Divergence from Randomness document weighting model. Timings are made using an Intel Xeon 2.66GHz.

To obtain the response time predictions, we follow Tonellotto et al. [5], by calculating various term-level statistics, such as the IDF, maximum score, number of postings, number of postings with scores $> 95\%$ maximum score. These are then aggregated across terms by sum, max, min, mean, median, stddev, variance and range functions, to form a total of 113 features (14 statistics * 8 aggregations + query length). Predicted response times are obtained by gradient boosted regression trees [3], trained on a separate subset of 2500 Efficiency task queries. Finally, to compare the five scheduling algorithms, we use two measures: average waiting time (AWT) and average completion time (ACT) over all the queries, in milliseconds (ms). Note that the average completion time is inclusive of the average waiting time.

4. EXPERIMENTAL RESULTS

From Table 1, we note that increasing both the numbers of shards and the number of replicas reduces both ACTs and AWTs. Indeed, in general, 2 shards with only 2 replicas is insufficient for a low completion time for this query workload, as queries can spend 8 seconds waiting for an available query server. For 5 or more shards, more than 4 replicas is sufficient for eliminating any *contention* for query servers (i.e. AWTs close to 0).

Next, comparing the scheduling algorithms, we note that Random obtains the highest ACTs and AWTs, because it can choose replicas that are busy, whist other replicas for that shard are idle. Queue Length is superior to Round Robin under high contention (i.e. 2 shards, 2 replicas). In other settings, Round Robin appears to better balance load than Queue Length. However, across different numbers of shards and replicas, Prediction always achieves the smallest AWT. For instance, with 4 replicas of the 2 shard index, Prediction can reduce AWT to 3ms, compared to 199ms for Queue Length and 174ms for Round Robin. Under settings with very little contention (e.g. 10 shards, 4 or 8 replicas), Round Robin has slightly lower ACTs than Prediction and even Oracle, due to the expense of predicting the response time (typically 6-40ms, depending on query length). Finally, Prediction obtains ACTs and AWTs that are almost identical to the best-case Oracle algorithm, based on actual response times. Overall, we find that using predicted response times to select the suitable replica for each query results in improved efficiency.

5. CONCLUSIONS

We proposed that using the predicted response time (obtained using query efficiency prediction) could enhance replica selection within a distributed/replicated IR system, compared to other scheduling algorithms. Indeed, experiments using the GOV2 corpus showed that the proposed Prediction algorithm could attain marked reductions in the query waiting times, across different number of shards and replicas. In future work we will investigate if query response times within a shard are correlated, and hence if the number of replicas required for a given shard can be predicted in advance.

6. ACKNOWLEDGEMENTS

Ana Freire acknowledges the support from the Spanish Government (Project TIN2009-14203). Craig Macdonald and Iadh Ounis acknowledge the support of EC-funded project SMART (FP7-287583).

7. REFERENCES

[1] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien. Efficient query evaluation using a two-level retrieval process. In *Proc. CIKM 2003*.

[2] F. Cacheda, V. Carneiro, V. Plachouras, and I. Ounis. Performance analysis of distributed information retrieval architectures using an improved network simulation model. *Information Processing and Management*, 43:204–224, 2007.

[3] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[4] C. Macdonald, N. Tonellotto, and I. Ounis. Learning to Predict Response Times for Online Query Scheduling. In *Proc. SIGIR 2012*.

[5] N. Tonellotto, C. Macdonald, and I. Ounis. Query efficiency prediction for dynamic pruning. In *Proc. LSDS-IR 2011*.

[6] F. Cacheda, V. Carneiro, V. Plachouras and I. Ounis. Performance Comparison of Clustered and Replicated Information Retrieval Systems. In *Proc. ECIR 2007*.

[2] `http://terrier.org`