Speeding up Batch Alignment of Large Ontologies Using MapReduce

Uthayasanker Thayasivam
THINC Lab, Dept. of Computer Science
University of Georgia, Athens, GA 30602, USA
Email: uthayasa@cs.uga.edu

Prashant Doshi
THINC Lab, Dept. of Computer Science
University of Georgia, Athens, GA 30602, USA
Email: pdoshi@cs.uga.edu

***Abstract*—Real-world ontologies tend to be very large with several containing thousands of entities. Increasingly, ontologies are hosted in repositories, which often compute the alignment between the ontologies. As new ontologies are submitted or ontologies are updated, their alignment with others must be quickly computed. Therefore, aligning several pairs of ontologies quickly becomes a challenge for these repositories. We project this problem as one of batch alignment and show how it may be approached using the distributed computing paradigm of MapReduce. Our approach allows any alignment algorithm to be utilized on a MapReduce architecture. Experiments using four representative alignment algorithms demonstrate flexible and significant speedup of batch alignment of large ontology pairs using MapReduce.**

I. Introduction

We are witnessing a growing number of ontology repositories hosting several ontologies on specific domains [1], [2]. Simultaneously, ontologies in these repositories are significantly large (more than 1,000 concepts). Because many of these ontologies overlap in their scope, aligning ontologies is important to the success and usefulness of the repositories [3].

Although ontology alignment is traditionally perceived as an offline and one-time task, issues of scaling to large ontologies and performing the alignment in a reasonable amount of time without much qualitative compromise are gaining importance. As new ontologies are submitted or ontologies are updated, their alignment with others must be quickly computed. As existing algorithms find it difficult to scale up for very large ontologies, aligning several pairs of ontologies quickly becomes a challenge for these repositories.

A prevalent way of managing the alignment complexity posed by large ontologies is to simply dissect the ontologies into smaller pieces and align some of the ontology parts [4], [5]. Parallelizing the alignment process is another way of approaching scalability. Intra-matcher parallelization introduces parallelization within the alignment algorithm. On the other hand, inter-matcher parallelization aligns several ontology parts in parallel using ontology alignment algorithms [6].

In the context of a general absence of inter-matcher parallelization, our primary contribution in this paper is a novel and general method for batch alignment of large ontology pairs using the distributed computing paradigm of MapReduce [7]. As distributed computing clusters including cloud computing proliferate, the significance of this approach is that it allows us to exploit these parallel computing resources toward automatically aligning several ontologies whose scale takes them out of the reach of many of the current algorithms, and simultaneously align in a reasonable amount of time. In contrast to simply dividing the batch of ontology pairs into mutually exclusive subsets and aligning each set on different nodes, we partition each ontology to obtain similar-sized subontology pairs and align them in a distributed manner. This provides improved and flexible speedup compared to the other approach.

In order to demonstrate the efficiency that MapReduce brings in general, we utilize Falcon-AO [8], Logmap [9] Optima+ [5], and YAM++ [10] as representative algorithms and the open-source Hadoop implementation [11] of MapReduce. Using batches of several ontology pairs spanning multiple domains, we show: (a) our formulation of distributed alignment using MapReduce demonstrates more than an order of magnitude in speedup for aligning multiple ontology pairs; (b) small changes in the quality of the alignment when using some of the algorithms while no change for others; (c) batch alignment of large ontologies using scalable algorithms such as Logmap may be further speeded up through distributed computing despite the overhead.

II. Distributed Ontology Alignment Using MapReduce Paradigm

MapReduce [7] is a popular programming framework for processing large data sets in parallel using a distributed computing environment. MapReduce involves two steps: **Map** and **Reduce**. The *map* function maps the input data to an intermediate data-set which is processed by a *reduce* function. The *reduce* function reads the output of map, processes it and generates the final output. MapReduce defines a *master* node and several *worker* nodes. The master node manages the distribution of tasks and data to worker nodes. A worker node is a *mapper* if it performs the map step or is labeled as a *reducer* if it is assigned a reduce task.

Input to the MapReduce framework is a list of data records, where each record has a unique key and a value. The master node splits the input and assigns each part to a mapper. The mapper reads each record in the given part and generates intermediate key-value pairs. The master node then processes the intermediate output from mappers and assigns a set of

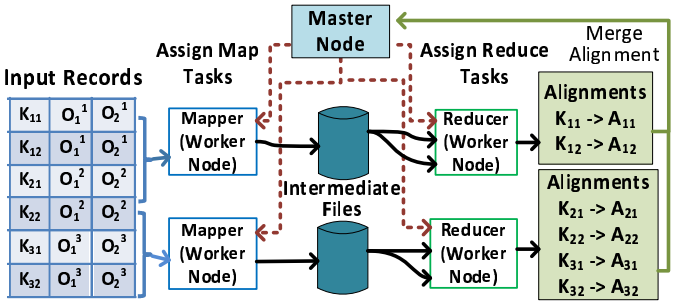


Fig. 1. The MapReduce framework for ontology alignment. The input is a list of key-value pairs, which is split. A mapper reads a record and writes intermediate key-value pairs to the different nodes' local file systems. A reducer reads the allocated intermediate output and aligns the subontologies. Finally, output alignments between the subontology pairs are merged.

keys and for each key the list of all associated values to a reducer. For each key, reducer processes the set of values and writes out the output in key-value pair format. Distributed implementations of MapReduce such as Hadoop [11] provide functionalities such as a simple partitioning of the input data, managing node failures, and administering communications, while expecting users to program the *map* and *reduce* steps. Approaches adopting this functional model may be naturally parallelized and executed on a large cluster of commodity machines. In this distributed setup, several mappers and reducers could be independently working in parallel. MapReduce provides a simple programming framework for tasks to scale up to large data while keeping the overhead of distributed computation transparent. Below we provide our approach for batch alignment using MapReduce. [1]

A. Identifying Alignment Subproblems

We formulate alignment subproblems by partitioning each pair of ontologies, $\mathcal{O}_1$ and $\mathcal{O}_2$ from the batch, and aligning pairs of parts. Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be partitioned into k_1 subontologies, $\{\mathcal{O}_1^1, \mathcal{O}_1^2, \ldots, \mathcal{O}_1^{k_1}\}$, and k_2 subontologies, $\{\mathcal{O}_2^1, \mathcal{O}_2^2, \ldots, \mathcal{O}_2^{k_2}\}$, respectively. Among the few existing partitioning approaches, Falcon-AO generates structurally cohesive subontologies using clustering [4]. Hamdi et al. [12] noted that in this approach each ontology is decomposed independently of the other without considering the alignment objective. This limitation is mitigated by first identifying anchors, which are entities in the two ontologies that have identical names or labels, followed by forming subontologies around these anchors based on the structural neighborhood. In this paper, we utilize this technique to cluster the concepts. To possibly avoid loosing relationships between entities in different clusters, we duplicate one of the participating entities in the other cluster and add the relationship. Note that this step may lead to overlapping subontologies, and therefore the parts do not technically form a partition. Given the subontologies, we formulate alignment subproblems, $(\mathcal{O}_1^i, \mathcal{O}_2^j)$ such that parts i and j have a correspondence between their anchors.

[1]We provide an implementation of our algorithm based on the alignment API provided by OAEI at http://tinyurl.com/mxsyq4f for reuse.

B. Aligning Ontologies Using MapReduce

An alignment subproblem, S_{ij}, is defined as, $S_{ij} = \langle K_{ij}, (\mathcal{O}_1^i, \mathcal{O}_2^j)\rangle$ where, K_{ij} is the unique key for the subproblem, and $\mathcal{O}_1^i$ and $\mathcal{O}_2^j$ are the subontologies that have a correspondance between their anchors, as discussed in the previous subsection. As shown in Fig. 1, the input to MapReduce is a set of key-value pairs such that the key uniquely identifies a subproblem and the value is a pair of subontologies associated with that subproblem. This list is split by the master node and the parts are sent to the mapper nodes. The map function reads in a data record, say S_{ij}, and writes out two intermediate key-value pairs, one for each subontology – $\langle K_{ij}, \mathcal{O}_1^i\rangle$ and $\langle K_{ij}, \mathcal{O}_2^j\rangle$. An instance of the reducer node will get these new key-value pairs, and possibly more with other keys. The reduce function aligns the subontologies associated with the same key, and writes out the output as another key-value pair where the key remains the same, K_{ij}, and the value is the alignment between the corresponding blocks. Alignments for all subontology pairs from all reducers are transferred to the master node where they are merged. The overhead of distributed execution is usually transparent.

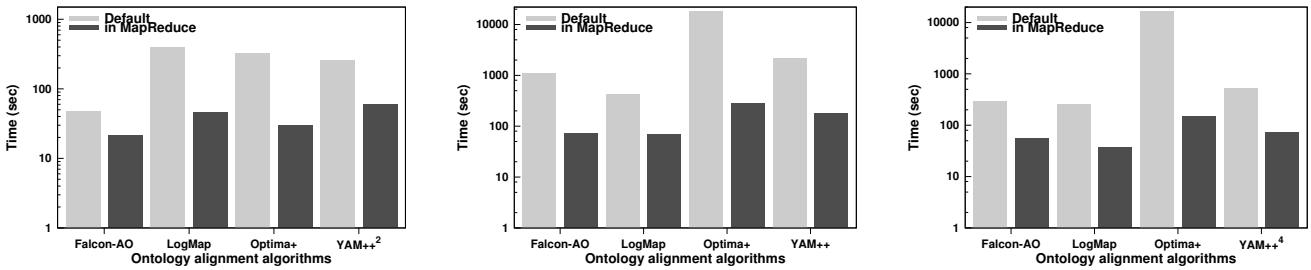
C. Merging Subproblem Alignments

Alignment algorithms may process the correspondences. The goal of this postprocessing is to remove inconsistent and duplicate correspondences. Despite this post processing, we may need to postprocess them further to remove specific inconsistencies. In addition to removing duplicate correspondences, we identify two inconsistencies, which must be resolved:

(a) Crisscross correspondence (b) Redundant correspondence

Fig. 2. Two types of inconsistent correspondences, which must be resolved while merging subproblem alignments.

1) *Crisscross mappings*, as illustrated in Fig. 2(a). While merging alignments from two subproblems, let there exist correspondence, $\langle x_a, y_\beta, =, c_{a\beta}\rangle$, in one alignment and, $\langle x_b, y_\alpha, =, c_{b\alpha}\rangle$, in the other, where x_a and x_b are entities in ontology, $\mathcal{O}_1$, y_α and y_β are entities in ontology, $\mathcal{O}_2$, and $c_{a\beta}$ and $c_{b\alpha}$ are confidence scores in the equivalence correspondences. If x_b is a subclass of x_a and y_β is a subclass of y_α then these crisscross correspondences are inconsistent. We remove the one with the lower confidence score while merging.
2) *Redundant mappings* are illustrated in Fig. 2(b). In order to keep the alignment minimal, we remove those correspondences which may be inferred from another. Let there exist correspondence, $\langle x_a, y_\alpha, \subseteq, c_{a\alpha}\rangle$, in one subproblem alignment and, $\langle x_a, y_\beta, =, c_{a\beta}\rangle$, in the other alignment. Here, x_a is an entity of ontology, $\mathcal{O}_1$, and



(a) (b) (c)

Fig. 3. Average execution times of Falcon-AO , Logmap, Optima+, and YAM++ in their original form on a single node and using MapReduce, for aligning the (a) *conference* track ontologies, (b) large OAEI ontologies and (c) biomedical ontologies from NCBO. Note, YAM++ has difficulties in aligning some ontology pairs in conference track and biomedical track[4]. Note that the time axes are in log scale.

TABLE I

THE PRECISION (P), RECALL (R) AND F-MEASURE (F) OF THE OUTPUT ALIGNMENTS BY FALCON-AO, LOGMAP, OPTIMA+, AND YAM++ IN MAPREDUCE SETUP FOR THE LARGE ONTOLOGY PAIRS FROM OAEI. THE DEFAULT PERFORMANCE OF LOGMAP AND YAM++ ARE ALSO PRESENTED IN THE TABLE FOR COMPARISON[4]. FALCON-AO AND OPTIMA+ PRODUCES SAME OUTPUT IN BOTH THE SETUP.

Ontology Pairs	MapReduce/Default Falcon-AO			MapReduce Logmap			MapReduce/Default Optima+			MapReduce YAM++			Default Logmap			Default YAM++		
	P%	R%	F%	P%	R%	F%	P%	R%	F%	P%	R%	F%	P%	R%	F%	P%	R%	F%
(mouse,human)	73	74	73	96	75	84	78	73	76	95	77	85	92	85	88	94	86	90
(STW,TheSoz)	57	50	53	57	51	54	18	40	25	55	52	53	69	64	67	60	75	66
(fma,nci)	95	81	88	95	83	89	96	83	89	97	84	90	95	86	90	98	85	91
(fma,snomed)	85	63	72	85	63	72	84	61	71	86	63	73	97	66	78	97	70	81
(snomed,nci)	69	58	63	67	58	62	70	58	63	71	58	64	90	64	75	95	60	74

y_α and y_β are entities of ontology, $\mathcal{O}_2$. If y_β is a subclass of y_α, then we may remove the correspondence, $\langle x_a, y_\alpha, \subseteq, c_{a\alpha}\rangle$, which can be inferred.

Though these inconsistencies are similar to those previously discussed [13], and can be resolved using the same techniques, they are not obtained in a similar manner. Importantly, we do not seek inconsistencies within an alignment of a subproblem, but address the inconsistencies between alignments of two different subproblems while *merging* them. We may enrich this postprocessing further using the techniques detailed in [13].

III. PERFORMANCE EVALUATION

We study the impact of distributing ontology alignment using MapReduce in terms of average speedup of the alignment time and its impact on the quality of the alignment. For this study, we utilize the four representative alignment algorithms: Falcon-AO, Logmap, Optima+, and YAM++. Specifically, we compare the total execution time when using MapReduce with the time required by the default setup of each alignment algorithm for aligning batches of ontology pairs. In order to evaluate the scalability of our formulation we measure the speedup obtained as we allocate an increasing number of nodes to Hadoop for each algorithms. We use a Hadoop cluster of 12 CentOS 6.3 systems, each with 24 2.0GHz Intel Xeon processors and memory limited to a maximum of 2GB per task in each node. All timing results are averages of 3 runs; we observed very small variances in the execution times.

We use three comprehensive batches of several ontology pairs spanning multiple domains. The first batch, labeled *conference testbed* consists of 120 medium-sized ontology pairs from the *conference* track of OAEI 2012, all of which structure knowledge related to conference organization. OAEI provides reference alignments for only 21 pairs in this track. The second batch includes large ontology pairs from *anatomy*, *library* and *large biomedical ontologies* tracks of OAEI 2012 along with their reference alignments. We call this batch as, *large OAEI testbed*. Finally, we utilized a recently created batch of 50 large ontology pairs using ontologies from the NCBO, available at http://tinyurl.com/n4t2ns3.

In Fig. 3, we show the average execution time consumed by Falcon-AO, Logmap, Optima+, and YAM++ in batch aligning ontology pairs from the three testbeds mentioned previously, in their default form on a single node and with MapReduce in the Hadoop framework [2]. We observe an order of magnitude reduction in average execution time brought about by MapReduce for all four algorithms in aligning large ontology pairs from OAEI. Importantly, while Logmap is designed to be the scalable, distributed alignment of a set of pairs using Logmap demonstrates significant speedup.

We tabulate the precision (P), recall (R) and F-measure (F) of the output alignments by all four algorithms in MapReduce setup for the large ontology pairs from OAEI in Table I. Because, both Falcon-AO and Optima+ by default employ partitioning, the performance metrics do not change between their default setup and MapReduce. We observed a significant reduction in F-measure when aligning subontology pairs in

[2]YAM++ in conference batch aligned only 21 ontology pairs and fails for rest of the pairs (similar difficulties were observed in OAEI 2012 [14]). Because its source-code is not available we could not investigate its failure. Also, it is not able align our large biomedical testbed without partitioning. Subsequently, we compare the performance of default YAM++ aligning ontology parts of the biomedical testbed with its MapReduce setup.

MapReduce using both Logmap and YAM++. A maximum of 13% reduction in F-measure is observed on using Logmap, for the *(snomed,nci)* ontology pair and on using YAM++ for the *(STW,TheSoz)* ontology pair. We believe that with improved partitioning techniques we may reduce this impact.

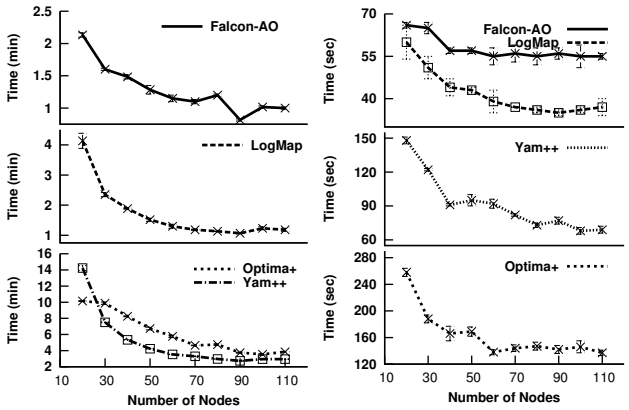
As an aside, partitioning is not mandatory for our approach. For example, we also observe a significant speedup in aligning the medium-sized conference ontology pairs in MapReduce without partitioning. Batch alignment of conference testbed using MapReduce and Falcon-AO, Logmap, Optima+, and YAM++ obtained 59%, 63%, 61% and 71% F-measure. Since we do not partition these medium-sized ontologies there is no change in output using MapReduce. For our biomedical testbed Falcon-AO generated a recall of 49% while with Optima+ a recall of 58% is obtained. Logmap and YAM++ produced 51% and 56% recall respectively.

To analyze the maximum speedup the MapReduce approach could offer for batch aligningnment and the minimum number of nodes required to achieve it, we gradually increased the number of nodes allocated and measured the average execution time of alignment. The average execution time to align, (a) large OAEI testbed and (b) biomedical testbed using MapReduce with increasing number of nodes is shown in Fig. 4. The execution time decreases exponentially with an increasing number of nodes until it reaches a minimum. We observed that the minimum number of nodes required to reach the minimum execution time varies between using different algorithms and data-sets. This is because, execution times required for aligning subproblems vary between algorithms.

IV. Conclusion

This paper showed how automated ontology alignment may be performed in a distributed manner using the popular distributed computing model, MapReduce, thereby allowing ontology alignment to exploit the proliferating cloud computing paradigm. MapReduce demonstrated significant speedup when aligning three different batches of ontology pairs using four representative alignment algorithms. This included recent efficient algorithms such as Logmap, whose alignment time for performing batch alignment reduced when deployed in MapReduce compared to its default execution on a single node. This paper represents an important step toward making alignment techniques computationally more scalable.

As additional analysis, we note that MapReduce also decreases the execution time of aligning a *single* ontology pair when used with any of the representative algorithms other than Logmap. For example, alignment using MapReduce with Falcon-AO gained speedup by a factor of 3.8 while with Optima+, it offered a speedup factor of 58, for aligning the very large ontology pair, *(mouse,human)*, from OAEIs anatomy track. MapReduce with YAM++ achieved a speedup of 22 for the same ontology pair. However, Logmap designed to be scalable from the ground up when used in MapReduce consumed 22 seconds more. Here, we note that YAM++ provides the best F-measure on this and many other ontology pairs, so its improved scalability is of import.



(a) Large OAEI Ontologies (b) Biomedical Ontologies

Fig. 4. The plot depicts the exponential decaying of average total execution time with increasing number of nodes by Falcon-AO, Logmap, Optima+, and YAM++ for (a) large ontologies from OAEI and (b) biomedical ontologies. Note, the average execution time gradually converges to a minimum time.

Acknowledgment

This research is supported in part by grant number R01HL087795 from the NHLBI. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NHLBI and NIH.

References

[1] K. Viljanen, J. Tuominen, E. Makela, and E. Hyvonen, "Normalized access to ontology repositories," in *Proceedings of the 2012 IEEE Sixth International Conference on Semantic Computing*, 2012, pp. 109–116.

[2] M. A. Musen, N. F. Noy, N. H. Shah, P. L. Whetzel, C. G. Chute, M.-A. D. Storey, and B. Smith, "The National Center for Biomedical Ontology," *Journal of the American Medical Informatics Association*, vol. 19, no. 2, pp. 190–195, 2012.

[3] G. Amir, N. Natalya, and M. Mark, "Creating mappings for ontologies in biomedicine: simple methods work." *AMIA*, pp. 198–202, 2009.

[4] W. Hu, Y. Zhao, and Y. Qu, "Partition-Based Block Matching of Large Class Hierarchies," in *Proceedings of the First Asian conference on The Semantic Web*, 2006, pp. 72–83.

[5] P. Doshi, R. Kolli, and C. Thomas, "Inexact Matching of Ontology Graphs Using Expectation-Maximization," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 2, pp. 90–106, 2009.

[6] A. Gross, M. Hartung, T. Kirsten, and E. Rahm, "On matching large life science ontologies in parallel," in *7th international conference on Data integration in the life sciences*, 2010, pp. 35–49.

[7] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[8] N. Jian, W. Hu, G. Cheng, and Y. Qu, "Falcon-AO: Aligning Ontologies with Falcon," in *K-Cap Workshop on Integrating Ontologies*, 2005, pp. 87–93.

[9] E. Jiménez-Ruiz and B. Cuenca Grau, "LogMap: Logic-Based and Scalable Ontology Matching," in *Proc. of the 10th International Semantic Web Conference (ISWC'11)*, vol. 7031, 2011, pp. 273–288.

[10] D. Ngo and Z. Bellahsene, "YAM++ : A Multi-strategy Based Approach for Ontology Matching Task," in *Knowledge Engineering and Knowledge Management*, vol. 7603, 2012, pp. 421–425.

[11] GlenMazza, "Hadoop Wiki," http://wiki.apache.org/hadoop/ProjectDescription, 2012.

[12] F. Hamdi, B. Safar, C. Reynaud, and H. Zargayouna, "Alignment-based Partitioning of Large-scale Ontologies," in *Advances in Knowledge Discovery And Management*. Springer, 2010, vol. 292, pp. 251–269.

[13] C. Meilicke, "Alignment Incoherence in Ontology Matching," Ph.D. dissertation, University of Mannheim, 2011.

[14] P. Shvaiko, J. Euzenat, A. Kementsietsidis, M. Mao, N. Noy, and H. Stuckenschmidt, Eds., *International Workshop on Ontology Matching*, vol. 946. CEUR-WS.org, 2012.