IETE TECHNICAL REVIEW, 2016
VOL. 33, NO. 4, 415–427
http://dx.doi.org/10.1080/02564602.2015.1117399

Evaluating Large-Scale Biomedical Ontology Matching Over Parallel Platforms

Muhammad Bilal Amin[a], Wajahat Ali Khan[a], Shujaat Hussain[a], Dinh-Mao Bui[a], Oresti Banos[a], Byeong Ho Kang[b] and Sungyoung Lee[a]

[a]Ubiquitous Computing Lab, Department of Computer Engineering, Kyung Hee University, Yongin-si, South Korea; [b]School of Computing and Information Systems, University of Tasmania, Hobart, Australia

ABSTRACT

Biomedical systems have been using ontology matching as a primary technique for heterogeneity resolution. However, the natural intricacy and vastness of biomedical data have compelled biomedical ontologies to become large-scale and complex; consequently, biomedical ontology matching has become a computationally intensive task. Our parallel heterogeneity resolution system, i.e., SPHeRe, is built to cater the performance needs of ontology matching by exploiting the parallelism-enabled multicore nature of today's desktop PC and cloud infrastructure. In this paper, we present the execution and evaluation results of SPHeRe over large-scale biomedical ontologies. We evaluate our system by integrating it with the interoperability engine of a clinical decision support system (CDSS), which generates matching requests for large-scale NCI, FMA, and SNOMED-CT biomedical ontologies. Results demonstrate that our methodology provides an impressive performance speedup of 4.8 and 9.5 times over a quad-core desktop PC and a four virtual machine (VM) cloud platform, respectively.

KEYWORDS

Biomedical informatics; Multithreading; Biomedical ontologies; Ontology matching; Parallel processing; Parallel programming; Semantic web

1. INTRODUCTION

Over the recent years, semantic web technologies have started penetrating in biomedical systems for greater benefits. Among these technologies, ontologies are extensively used in biomedical information systems [1]. This usage is largely contributed for annotation of medical records [2], standardization of medical data formats [3], medical knowledge representation and sharing, clinical guidelines (CG) management [4], clinical data integration, and medical decision-making [5]. These vast usages of ontologies in the biomedical field have compelled researchers to invest more in development of newer ontologies and provide continuity to the already created ones. Therefore, biomedical ontologies like the Gene Ontology (GO) [6], the National Cancer Institute (NCI) Thesaurus [7], the Foundation Model of Anatomy (FMA) [8], and the Systemized Nomenclature of Medicine (SNOMED-CT) [9] have emerged and maintained over the years. There exist several service-oriented infrastructures encouraging the development and usage of ontologies in biomedicine including, BioPortal [10] and OBO Foundry [11]. BioPortal currently hosts a repository of 384 biomedical ontologies. The Open Biomedical Ontologies (OBO) consortium worked on introducing strategies for evolving ontologies [11]; however, the design, development, and management of biomedical ontologies become challenging due to continuous evolution of medical data. Consequently, biomedical ontologies are becoming larger in size and their growing use is making them increasingly available.

Large biomedical ontologies are complex in nature, containing overlapping information. Utilization of this information is necessary for the integration, aggregation, and interoperability; for example, NCI ontology defines the concept of "Myocardium" related to the concept "Cardiac Muscle Tissue", which describes the muscles surrounding the human heart. Concept "Cardiac Muscle Tissue" is defined in FMA ontology; therefore, a biomedical system or a professional, integrating knowledge regarding human heart, requires correspondence between candidate ontologies FMA and NCI [1]. Likewise, finding correspondence between GO ontology and FMA ontology can be used by molecular biologist in understanding the outcome of proteomics and genomics in a large-scale anatomic view [12]. Moreover, correspondence between ontologies has also been used for heterogeneity resolution among various health standard [13]. This correspondence between candidate ontologies is called mappings or alignments and the process of discovering these mappings is termed as ontology matching.

The active research community has acknowledged the importance of matching large-scale biomedical ontologies. Subsequently, initiatives like Unified Medical Language system (UMLS) by National Library of Medicine

[14] and Ontology Alignment Evaluation Initiative (OAEI) [15] are now mainstream biomedical ontology matching research campaigns. In complement to these initiatives, our motivation also lies in the use of ontology matching for the integration of biomedical information; however, ontology matching over large-scale biomedical ontologies is a computationally intensive task with quadratic computational complexity [16]. Ontology matching is a Cartesian product of two candidate ontologies, which requires resource-based element-level (string-based, annotation-based, language-based, and label-based) [17] and structural-level (child-based, graph-based, and property-based) [17] matching algorithms to be executed over candidate ontologies for the generation of the required mappings. In our experiments, executing these matching algorithms over large-scale biomedical ontologies, whole FMA with whole NCI has taken 3 days to generate desirable results. This delay in mapping results makes ontology matching of large-scale biomedical ontologies ineffective for biomedical systems and professionals with in-time processing demands. Ontology matching problem is formally defined as following:

For given two ontologies $O_S = \langle C_S, R_S, I_S, A_S \rangle$ and $O_T = \langle C_T, R_T, I_T, A_T \rangle$, different type of inter-ontology relationships called mappings or alignments can be defined. These mappings are derived by a set of matching algorithms with a similarity degree $\delta \in [0, 1]$. A matching is a quadruple $m = \langle \mathrm{id}, x_S, x_T, \delta \rangle$, x_S, and x_T are aligned ontology terms, and δ is the similarity degree of m.

Over the years, ontology matching systems and techniques have taken large-scale biomedical ontologies into consideration and proposed various resolutions. However, these resolutions are matching effectiveness-centric, i.e., accuracy of the matching algorithms. The performance aspect of matching these ontologies is concentrated on optimization of the matching algorithms and partitioning of larger ontologies into smaller chunks for performance benefits [18]. Due to the trade-off between performance and accuracy, optimizing the matching algorithms can take the performance-gain in ontology-matching only to a certain degree. Furthermore, the performance improvement based on exploitation of newer hardware technologies has greatly been missed. Among these technologies are affordable parallelism-enabled systems, which are easily available as stand-alone (desktop) and distributed platforms (cloud) [19].

Parallelism has long been associated with high performance computing (HPC); however, with the advent of virtualization over multicore processors, performance-oriented computing environments are ubiquitously available as cloud platforms [20]. Moreover, cloud computing with its limitless yet affordable computational power can be exploited for applications with higher complexity [21]. Our ontology matching system SPHeRe [22,23] avails this opportunity and provides a performance-based ontology matching resolution, which exploits multicore platforms, i.e., desktop and particularly cloud for parallel ontology matching.

To contribute in the performance aspect of large-scale biomedical ontology matching, we have enabled SPHeRe to execute parallel matching over these complex and comprehensive ontologies. Therefore, we have evaluated the performance measures by incorporating it with the interoperability engine of a clinical decision support system (CDSS) and deploying it over a quad-core desktop PC and four VM cloud platform. We have been able to achieve an impressive performance speedup of 4.8 times over the desktop and 9.5 times over the cloud platform on matching requests for large-scale biomedical ontologies FMA, NCI, and SNOMED-CT. Furthermore, we have compared this method with GOMMA's parallel matching techniques called inter- and intra-matching [18] used for matching biomedical ontologies. Our method outperforms inter- and intra-matching technique by 50% in performance speedup and 16% in scalability over multi-node platform, proving this method to be more performance efficient and effective in utilizing available computational resources.

The rest of the paper is structured as follows. In Section 2, we provide the methodology used by SPHeRe for matching large-scale biomedical ontologies. Section 3 describes the experimentation performed over the multicore desktop PC and the cloud platform, and discusses the results. Section 4 briefly discusses the related work in the area of biomedical ontology matching from the perspective of performance. Section 5 concludes this paper.

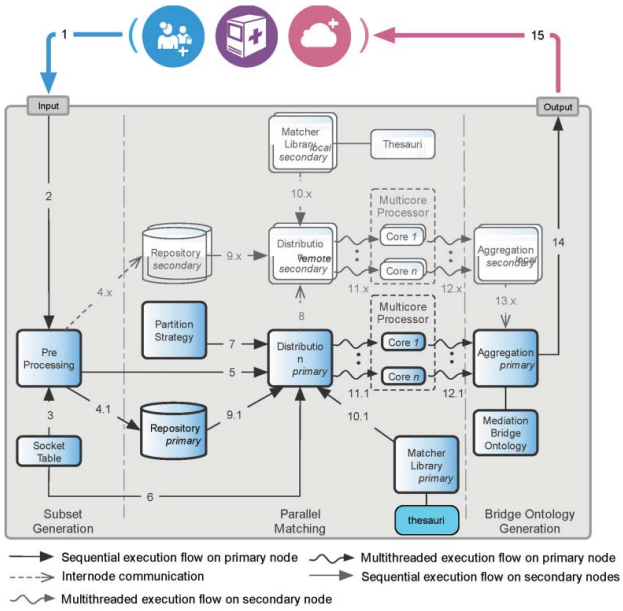
2. METHODOLOGY OVERVIEW

This section provides the overview of the methodology used by SPHeRe for large-scale biomedical ontology matching. The intrinsic technical details of SPHeRe's implementation are already provided in [22]; furthermore, the finer details of the methodology are comprehensively covered in [24].

The primary goal of the methodology is to exploit parallelism-enabled platforms for large-scale biomedical ontology matching by distributing the matching process of candidate ontologies, i.e., source ontology and target

ontology over individual cores as independent matching tasks. A matching task is the unit of matching process defined as a single independent execution of matching algorithm over a resource from source and target ontologies (e.g., matching the concept "Abdominal skin" from NCI ontology with the concept "Skin of abdomen" from FMA ontology by using an element-level synonym matching algorithm is a single matching task).

The execution flow of our proposed methodology is described in Figure 1. As illustrated in the figure, to complete the whole matching process, a request is processed through subset generation, parallel matching, and bridge ontology generation stages. In subset generation stage, candidate ontologies are divided into smaller and simpler subsets based on the ontology matching algorithms. In parallel matching stage, ontology subsets are distributed as matching tasks among available computing resources. In bridge ontology generation stage, matched results are aggregated and transformed into a formal representation for utilization. These stages are explained in following subsections.



Control Flow	Description
1	Ontology matching request
2	Candidate ontologies for matching {Source ontology, Target ontology}
3 & 6	Connectivity information of participating nodes
4.x	Serialized subsets or Change Instances of candidate ontologies
5	Request for ontology distribution for parallel matching
7	Partitioning algorithm for ontology as data distribution
8	Task assignment for secondary nodes {Control message}
9	Deserialization of ontology subsets from primary repository
10	Matching algorithm instances
11	Matcher threads
12	Matched results
13	Aggregated matched results
14	Bridge ontology
15	Ontology matching response

Figure 1: Execution flow

2.1 Subset generation

As illustrated in Figure 1, requests for matching large-scale biomedical ontologies are submitted by providing either the actual candidate ontologies or uniform resource identifier (URI). The candidate ontologies encapsulated by the matching request are fed to the subset generation stage where preprocessing creates smaller, simpler, and scalable subsets of these ontologies. There are two primary motivations for the subset approach. First, large biomedical ontologies are complex in nature and are not scalable. On the other hand, subsets are built on linear data structures (e.g., list, arrays) that can be easily partitioned among multiple threads for efficient scalability. Second, these subsets are generated depending upon the needs of the matching algorithms; for example, an element-level synonym-based matcher for concept names only require a linear data-structure of concepts and an annotation-based matcher will only require concepts names, their associated labels, and comments. By this method, subsets of candidate ontologies become independent of each other; thus, in parallel matching stage, matching threads will only load those subsets that required by the matching algorithm(s) they are going to execute. Consequently, the memory stress during parallel matching is reduced. In our experiments, we have recorded as much as 8 times smaller memory footprint during matching [22]. Furthermore, there is no inter-thread communication during parallel matching due to this method.

Parsing large-scale biomedical ontologies is an expensive process, as it requires preprocessor to iterate over every ontology resource. To preserve this parsing effort, subsets are serialized to ontology repositories by the preprocessor. This method facilitates our methodology to avoid re-processing of processed ontologies for future matching requests. Moreover, loading an ontology subset is faster than loading a whole ontology due to its simple data structures and smaller size. In our experiments, we have recorded as much as four times faster ontology loading from subsets than from a whole ontology [22].

To facilitate parallel matching in a distributed environment like cloud, ontology subsets are replicated over participating nodes. Furthermore, due to the dynamic and ever-evolving nature of medical data, evolution in ontologies is inevitable. To support the continuous change, instead of re-processing any updated ontology, preprocessing only implements the updated instances in the primary repository and propagates the changes across the repositories in secondary nodes.

2.2 Parallel matching

The goal of parallel matching stage is to distribute candidate ontology subsets as matching tasks over available computing cores and invoke matching algorithms on them. To achieve this goal, total numbers of matching tasks are determined from serialized subsets of ontologies and size-based partitioning is performed. Multicore and multi-node distribution is responsible for the distribution of matching process over computational resources for invoking data parallelism on candidate ontologies in parallel matching stage.

In case of multi-node distribution, primary node receives the whole matching request and divides it among the participating computing nodes as per their individual computational resources. These matching requests are further subdivided by individual nodes into matching jobs such that each job on a node contains equal number of matching tasks. Subsequently, a matching job is assigned to execute over a processing core available on a participating node. For multi-node matching request distribution, primary node requires the communication information of all the secondary nodes. This information is instantiated as socket objects containing information of IP addresses and number of computing cores available on all secondary nodes. These socket objects are persisted in a configuration table called a socket table. Socket table generation is a passive process done once per distributed matching environment warm-up. Multi-node matching request distribution receives the information regarding the available computational resource of secondary nodes from the socket table. Size-based partitioning strategy is performed on the larger size ontology, resulting in determination of partitioning slab required for the distribution of equal number of matching tasks. Depending upon this distribution, matching requests are created and sent as control messages to the secondary nodes.

Control messages are received by multicore distribution of secondary nodes. Matching requests extracted from the control messages provide each node with the information regarding their matching jobs, which include the information about candidate ontologies and the portion of matching tasks to be performed. Consequently, every secondary node loads their respective candidate ontologies and creates matching jobs by assigning matching request attributes with matching algorithms. Furthermore, each job is exploited as a thread over a core.

Multicore and multi-node distribution benefits the matching methodology of SPHeRe by better scalability, as each core is assigned with equal number of matching tasks (no idle cores) and executes a single matching job (single-thread per core). Furthermore, the independent nature of matching tasks ensures that no communication takes place among the cores during parallel matching, thus contributing more in the matching performance.

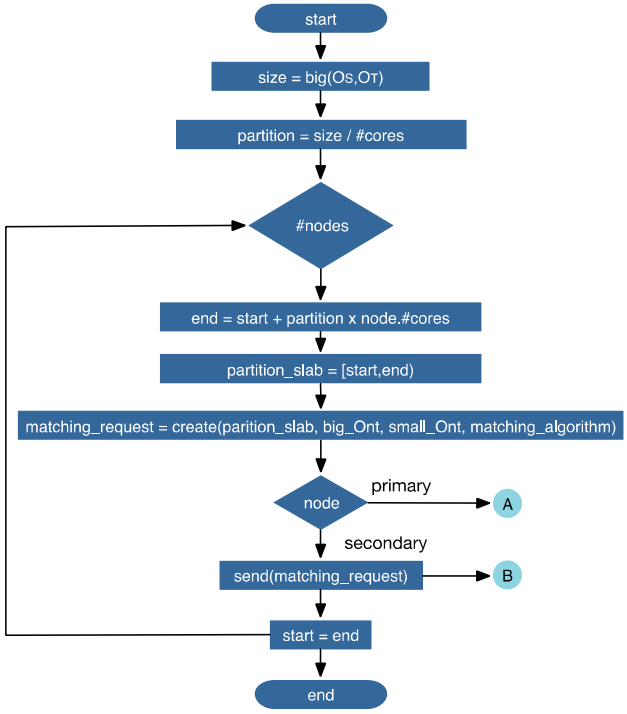


Figure 2: Flowchart for multi-node distributor of matching tasks

Flowcharts from Figures 2 and 3 illustrate the multi-node and multicore distribution algorithms. For explanation, consider the example of a matching problem of two real-world ontologies MA (2744 concepts) and NCI (3304 concepts) being matched over a distributed environment of four nodes each with dual cores. For data parallel distribution, the essential computation is the derivation of partitioning slab. This partition slab is calculated by first taking the size of the bigger ontology, which in this case is NCI with its 3304 concepts, and dividing it with the total number of cores present in the distributed environment (3304/8 = 413). Partitioning slab enables the execution to determine the starting and ending indexes of matching tasks for each matching request. For primary node, two matching jobs are created with first 826 concepts of NCI to be matched with all the concepts of MA. In parallel, three matching requests are also created for the three secondary nodes with rest of the 2478 concepts of NCI i.e., [826, 1652), [1652, 2478), and [2478, 3304) respectively. At node level, multicore distributor further classifies each request into matching jobs. Each job is assigned with their partitioning slab containing their respective 413 concepts of NCI to be matched with all

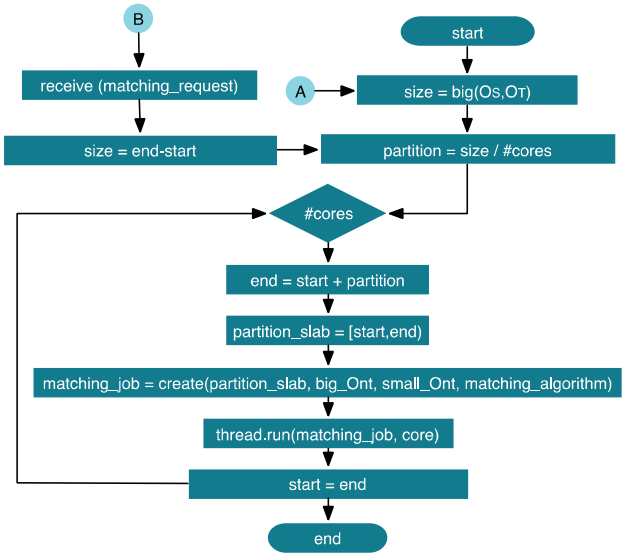


Figure 3: Flowchart for multicore distributor of matching tasks

the concepts of MA ontology achieving equal distribution of matching tasks. Executions illustrated in Figures 2 and 3 work hand-in-hand to accomplish the complete distribution of the ontology matching problem over available computing resources.

Distribution process also provides an interface to matcher library, which encapsulates biomedical ontology matching algorithms with variety of complexity, accuracy measures. Instances of these matching algorithms are assigned to the individual matching tasks during parallel matching stage.

2.3 Bridge ontology generation

The goal of bridge ontology generation is to aggregate matched results from participating node(s) and generate bridge ontology. Depending upon the deployment platform (single- or multi-node), aggregation of primary node accumulates matched results from local and remote distributions, and creates a formal representation of mappings called mediation bridge ontology [25,26]. This bridge ontology can be of various patterns depending upon their usage and biomedical client [26]. The finalized mediation bridge ontology is delivered to the biomedical client as the matching response, and subsequently persisted in participating repositories for the same requests in future.

3. EVALUATION AND DISCUSSION

In this section, we describe the evaluation performed on SPHeRe by incorporating it with a running instance of a CDSS [27]. To evaluate our system during this integrated execution, three of the most complex and large-scale real-world biomedical ontologies FMA (78,989 concepts), NCI (66,724 concepts), and SNOMED-CT (122,464 concepts) are used for experimentation. We have executed a matcher library consisting upon two element-level (string-based and annotation-based) and one structural-level (child-based) [25,26] matching algorithms provided to us by semantic web experts from biomedical field. The computational complexity of all these matching algorithms is $\geq O(n^2)$. Similar to [24], our matching methodology is evaluated over two widely available parallel platforms, i.e., multicore desktops and multi-node cloud platform. Our test scenarios execute over: (1) a single-node quad-core desktop PC, equipped with 3.4 Ghz Intel(R) Core i7(R) Hyper-Threaded (Intel (R) HT Technology) [28] CPU (two threads/core) with 16 GB memory and (2) a medium-sized instance of a multi-node cloud platform providing four virtual machines (VM); each VM is provided with 4 GB memory and two cores. Hence, this cloud platform cumulatively provides a test-bed of eight cores and 16 GB of distributed memory for our system's evaluation.

3.1 Scenario I: multicore desktop PC

For the first scenario (illustrated in Figure 4), matching requests are generated from the interoperability engine. Candidate ontologies FMA (O_S) and NCI (O_T) are requested for matching by the interoperability engine. Subsets of these ontologies O_s and O_t are loaded by preprocessing and provided to multicore distribution. With the knowledge of available computing resources (two cores, eight threads) and ontology subsets (O_s, O_t) required by matching algorithms, multicore distributor creates eight independent matching jobs {MJ_1, MJ_2, … MJ_8}. Each job is allocated with a set of equal numbers of independent matching tasks ($FMA_{classes} \times NCI_{classes}$)/8. As string- and annotation-based matching algorithms execute on the same subsets of the respective ontologies {O_s, O_t}, multicore distributor assigns these algorithms to every matching job. Subsequently, multicore distributor allocates each matching job to a single thread (matcher thread) for the execution of the matching process. All matcher threads are executed in parallel as independent matching jobs. After completion of all jobs, results from all matching threads {R_1, R_2, … R_8} are accumulated and an intermediate bridge ontology (O_{b0}) is generated. Thereafter, preprocessing loads the subsets of FMA and NCI required for child-based matching algorithm {O'_s, O'_t} and follows the same procedure of distribution and accumulation. After completion of matching jobs executing child-based matching algorithm, matching results are accumulated and another

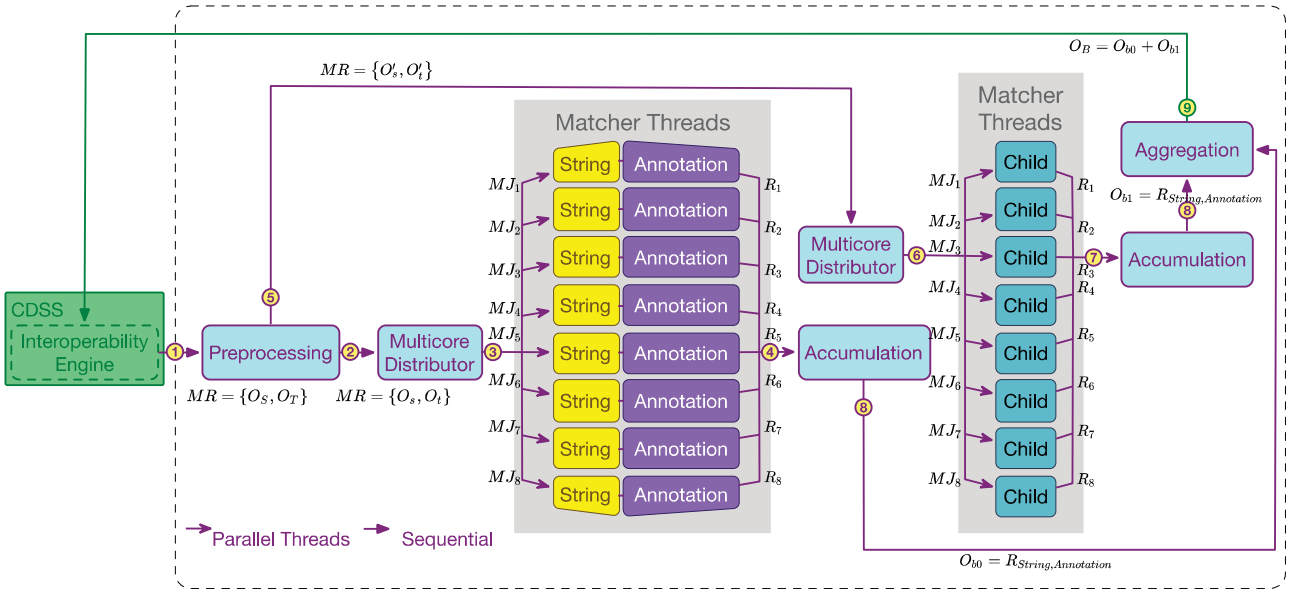


Figure 4: Scenario I: parallel flow on a single-node quad-core (two threads/core) desktop PC

intermediate bridge ontology (O_{b1}) is generated. Aggregation combines the two intermediate ontologies (O_{b1} and O_{b2}) and generates a formal mediation bridge ontology (O_B). This bridge ontology is finally delivered to the interoperability engine of the CDSS. The same scenario is repeated with FMA (O_S) and SNOMED-CT (O_T), and NCI (O_S) and SNOMED-CT (O_T) ontologies.

3.2 Scenario II: multi-node cloud platform

For the second scenario (illustrated in Figure 5), primary node receives the matching request for candidate ontologies, FMA (O_S) and NCI (O_T), from interoperability engine. Subsets of candidate ontologies are loaded by preprocessing of primary node, which consequently invokes the multi-node distributor for distributed matching. Socket table at the primary node provides the multi-node distributor with socket objects for all secondary nodes. With the information on available computing resources (four nodes, one primary and three secondary, each with two cores available) and ontology subsets (O_s, O_t) required by matching algorithms, multi-node distribution of primary node creates four independent yet equal size matching requests. The local multicore distributor handles the first matching request and creates two independent matching jobs with equal number of independent matching tasks. In parallel, multi-node distributor sends control messages to all three secondary nodes with their respective matching requests. At receiving nodes, these matching requests are forwarded to their local multicore distributors. All four participating nodes load serialized subsets containing class names, annotations and labels, and class relationships of FMA {O_s} and NCI {O_t} from their respective ontology repositories. Onwards, all the nodes execute in multicore execution mode until an intermediate bridge ontology is generated by every node (O_{b0}, O_{b1}, O_{b2}, O_{b3}). These intermediate bridge ontologies are accumulated by aggregation at the primary node and finally delivered to the interoperability engine of the CDSS as the formal mediation bridge ontology (O_B). The same scenario is repeated with FMA (O_S) and SNOMED-CT (O_T), and NCI (O_S) and SNOMED-CT (O_T) ontologies.

Results from both these scenarios are compared with conventional sequential flow of the same matching requests. These results are based on total matching time, which includes loading time for serialized subsets of candidate ontologies, parallel matching time taken by all the matching requests, and aggregation of matched results to generate the bridge ontology. These results are discussed in the following subsection.

3.3 Results and discussions

We have evaluated each scenario with three matching requests. The first test is performed over a matching request generated by the interoperability engine for the whole FMA with whole NCI ontology. For the first scenario, this request executes over quad-core desktop and the results are described in Figure 6(a). The sequential process takes a little over 8 h (8.17 h) to complete the matching request; however, with the use of our data parallel methodology over multiple cores, total matching

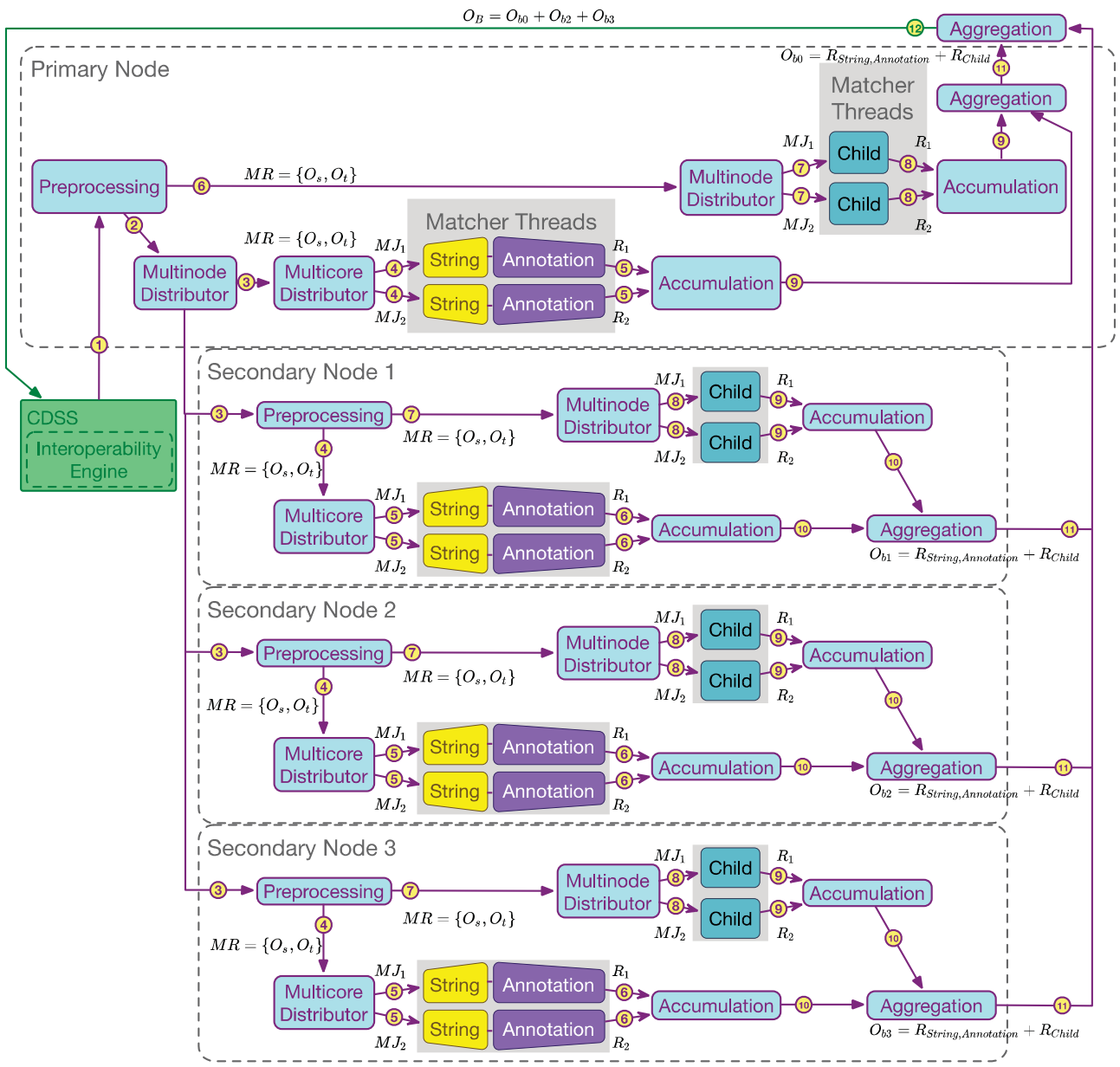


Figure 5: Scenario II: parallel flow on multi-node cloud platform

time starts improving significantly as more cores are introduced. With data parallel methodology, the matching process is completed in less than 2 h (1.78 h) over four cores (= 8 threads) with an performance speedup of 4.58. The same matching request is executed for the second scenario over four-node cloud platform and the results are described in Figure 6(b). The sequential process takes over 10 h (10.52 h) to complete; however, by using our methodology, this time starts improving as more nodes are added. The time improves to 2.4 times by executing the request over single node parallel (two threads). This time further improves by scaling the matching process over all four nodes. With data parallel methodology, the matching process is completed in a little over an hour (1.08 h) with an impressive performance speedup of 9.74.

The second test is performed over a matching request generated by the interoperability engine for the whole FMA with a large fragment of SNOMED-CT ontology. For the first scenario, the request executes over quad-core desktop and the results are described in Figure 7(a). The sequential process takes 16 h (16.09 h) to complete the matching request. With our methodology, the matching process over four cores is completed in 3 h (3.13 h) with an performance speedup of 5.14. The same matching request is executed for the second scenario over four-node cloud platform and the results are

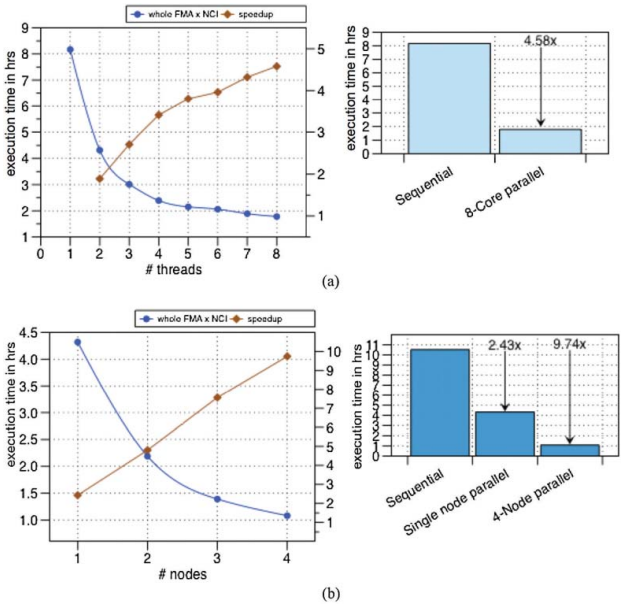


Figure 6: Whole FMA with NCI matching results. (a) Multicore desktop scenario results; (b) multi-node cloud platform scenario results

described in Figure 7(b). The sequential process takes over 18 h (18.91 h) to complete. With our data parallel methodology, this time starts improving as more nodes are added. The matching time improves to 2.5 times by executing the request over dual-core single node. This time further improves by scaling the matching process to all four nodes. Our methodology completes the matching process in less than 2 h (1.92 h) with an impressive performance speedup of 9.84.

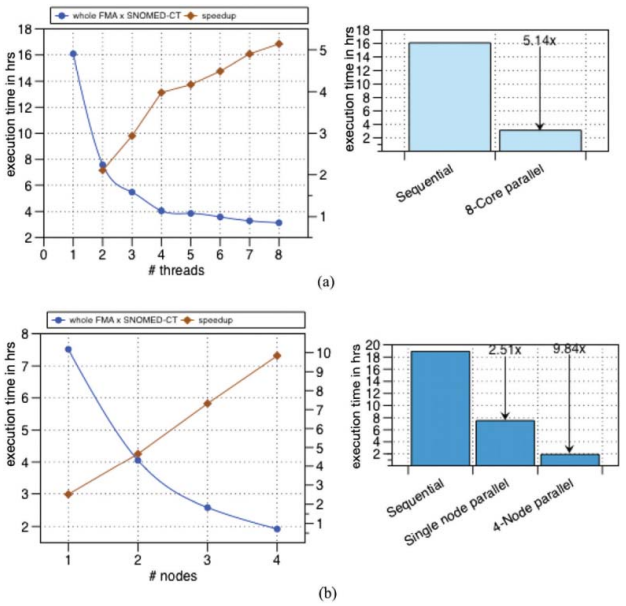


Figure 7: Whole FMA with SNOMED-CT matching results. (a) Multicore desktop scenario results; (b) multi-node cloud platform scenario results

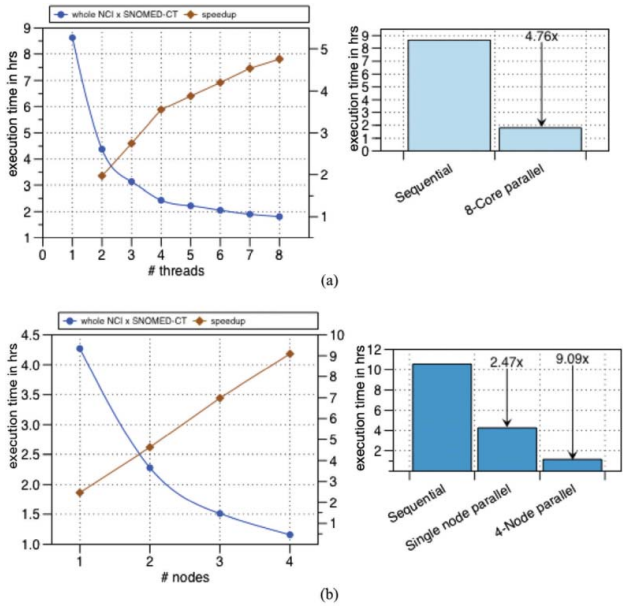


Figure 8: Whole NCI with SNOMED-CT matching results. (a) Multicore desktop scenario results; (b) multi-node cloud platform scenario results

Third test is performed over a matching request generated by the interoperability engine for whole NCI with a large fragment of SNOMED-CT ontology. Similar to the previous two test runs, the first scenario is executed over multicore desktop and its results are presented in Figure 8(a). The sequential process takes over 8 h (8.63 h) to complete; however, our methodology over four cores completes the same matching process and generates the bridge ontology in less than 2 h (1.16 h) with an performance speedup of 4.76. The same matching request is executed for the second scenario over four-node cloud platform and the results are described in Figure 8(b). The sequential process takes over 10 h (10.55 h) to complete; however, our data parallel methodology takes 4 h over a single node, which is 2.4 times faster. When scaled over four nodes, our methodology completes the matching process in a less than 2 h (1.81 h) with an impressive performance speedup of 9.09. Table 1 summarizes the results from our six test cases.

Table 1: Result summary

Test cases	Platform	Speedup
FMA × NCI	Quad-core desktop	4.58x
FMA × NCI	4-node cloud	9.74x
FMA × SNOMED-CT	Quad-core desktop	5.14x
FMA × SNOMED-CT	4-node cloud	9.84x
NCI × SNOMED-CT	Quad-core desktop	4.76x
NCI × SNOMED-CT	4-node cloud	9.09x

3.4 Parallelism efficiency

The execution of the SPHeRe is a three-stage process, i.e., preprocessing, parallel matching, and bridge ontology generation stage. However, from the runtime perspective, parallel matching is the stage that takes most of the execution time. Depending upon the type of ontology matching problem, our results have recorded that around 90% of the execution time is consumed by the matching phase which SPHeRe has parallelized for performance benefits. By using the Amadahl's Law [29], we can theoretically calculate the performance speedup of very large-scale biomedical ontology matching problems with computing cores beyond eight. For example, in case of 16 cores, with 90% of total execution being in parallel, the theoretical performance speedup is: $\frac{1}{0.1+\frac{0.9}{16}} = 6.41$. Similarly, in case of 24 and 32 cores, with 90% of total execution being in parallel, the theoretical performance speedup is 7.29× and 7.8×, respectively.

4. RELATED WORK

In this section, we overview the related tools and techniques built for matching biomedical ontologies. Some of the discussed systems are particularly built for biomedical ontologies; however, few are generic ontology matching systems with matching support for biomedical ontologies.

SAMBO [30] is a pioneering system for aligning and merging biomedical ontologies. Its implementation uses WordNet [31], UMLS as meta-thesaurus, and PubMed abstracts for higher accuracy in alignments. These third-party thesauri and resources are slow in nature and create performance bottlenecks while matching over millions of concepts. Besides that, SAMBO's sequential nature of execution limits its abilities to overcome its performance bottlenecks with better and parallel platforms. For SAMBO's evaluation, the authors have used very small subsets of biomedical ontologies GO (57 and 73 terms) with SigO (10 and 17 terms) [32] and MeSH (15, 39, and 45 terms) [33] with MA (18, 77, and 112 terms) [34] and have not provided any benchmarks regarding large-scale biomedical ontologies. However, from the results of OAEI 2008 [35], SAMBO took 12 h to complete the anatomy track of biomedical ontologies NCI and MA.

ASMOV [36] is another ontology matching system with the motivation of producing alignments for biomedical ontologies. With its computational performance being dependent on the comprehensiveness of its matching algorithms, the authors of [36] acknowledged that, effort is required to improve the computational complexity of the system. Due to its sequential execution, ASMOV is unable to avail any performance benefits from parallel platforms. Evaluation of ASMOV over anatomy parts of NCI (3304 classes) with Adult Mouse Anatomy (2744 classes) is provided in [36]. Even for such a small matching task, ASMOV took 3 h to complete the matching process.

ServOMap [37] is another ontology matching system built particularly for matching large-scale biomedical ontologies. ServOMap has been able to record a better performance over large-scale biomedical ontologies FMA, NCI, and SNOMED-CT; however, from [37], it is understood that this performance gain is due to the absence of third-party resources and thesauri. ServOMap does not implement any performance gain techniques that can exploit parallelism for the benefit of matching.

In current state-of-the-art generic ontology matching systems, i.e., AgrMaker [38], LogMap [39], and GOMMA [40], performance has been given a vital consideration. Performance of AgrMaker depends upon the iterative execution of matching algorithms. However, with no parallelism at all, baseline performance of AgrMaker depends upon the complexity of the first iteration of matching algorithms. From OAEI 2011.5 campaign, AgrMaker scored the highest precision over matching biomedical ontologies, but lagged over performance. It did not take part in any further OAEI campaigns.

LogMap is a generic ontology matching system with a step-by-step matching process. For performance reasons, it uses highly optimized data structures for lexical and structural indexing; however, similar to AgrMaker, the whole matching process is sequential in nature.

Among the state-of-the-art ontology matching systems, GOMMA is considered the most performance efficient ontology-matching tool. The authors of GOMMA have proposed inter- and intra-matcher parallelism techniques, which uses parallel and distributed infrastructure for ontology matching to achieve a better performance [40]. Inter-matcher parallelism processes independent matchers on a parallel platform. However, as mentioned by the authors of [18], due to its high memory requirements, inter-matcher creates memory strains during execution. On the other hand, intra-matcher parallelism deals with decomposition of ontology resources into several finer parts with limited complexity. However, the definition of granularity has not been provided. Furthermore, neither inter- nor intra-matcher guarantees the optimal computational resource utilization, and the ontologies used for

Table 2: Performance speedup comparison

Platform	Intra- and inter-matcher speedup	SPHeRe speedup	Improvement
Single-node	4.5−5.0	4.5−5.1	2%
Multi-node	6.5	9.0−9.8	50%

their evaluation are far smaller in size, i.e., Adult Mouse Anatomy MA (2737 concepts) with anatomical part of NCI Thesaurus (3289 concepts) and two GO sub-ontologies Molecular Function (9395 concepts) with Biological Processes (17,104 concepts). We have evaluated SPHeRe's methodology of matching biomedical ontologies by comparing the results with intra- and inter-matcher (Intra&Inter) technique. Although the evaluation setup used by [18] is far superior to our commodity hardware and the ontologies used are far smaller in size, SPHeRe has been able to outperform Intra&Inter matcher in performance speedup and scalability.

Performance speedup comparison between Intra&Inter matcher and methodology of SPHeRe is described in Table 2. Results have been drawn over eight cores single- and multi-node configuration, where a single thread exploits each core. Due to our explicit distribution of matching request to independent matching jobs and matching tasks over individual cores by thread-level parallelism, SPHeRe was able to outperform Intra&Inter matcher, even with ontologies larger in size and platform with weaker computation power. Due to the magnitude of the large-scale biomedical ontology-matching problem, it requires a distributed environment for performance-based resolution; thus, efficiency of multi-node distribution is the key. As shown in Table 2, on a multi-node configuration, our methodology's performance speedup is far ahead (50%) of Intra&Inter matcher. Therefore, SPHeRe is more effective for parallel large-scale biomedical ontology matching than Intra&Inter matcher.

Reduction score described in Table 3 measures the scalability of both methodologies. Results are calculated by using OAEI's formula for reduction score, which is a ratio between execution time on n cores and execution time on a single core. For example, in case of four cores:

$$R_{\text{score}} = \frac{T_4}{T_1} \times 100 \mid T_x \rightarrow \text{execution time on } x \text{ cores.}$$

Table 3: Reduction score comparison

Platform	Intra- and inter-matcher (variation from 25%)	SPHeRe
Single-node	30% (+5)	27% (+2)
Multi-node	31% (+6)	26% (+1)

System with the best scalability will score around 25%. Due to our distribution that ensures reduced chances of idle cores, our methodology outperforms Intra&Inter matcher again by scoring closer to 25%. This score shows that SPHeRe's methodology utilizes computational resources far more efficiently than Intra&Inter matcher.

5. CONCLUSION

In this paper, we presented methodology of our ontology-matching tool, i.e., SPHeRe for matching large-scale biomedical ontologies equipped with parallel matching for performance benefits. Ontology matching is a widely used technique for integration and interoperability among biomedical systems; however, size and complexity of these ontologies require solutions that are built from a performance aspect. With the availability of affordable multicore platforms like desktop and cloud, SPHeRe exploits their performance benefits by thread-level parallelism for ontology matching. Its methodology decomposes the large complex biomedical ontologies into simple and scalable resource-based ontology subsets, based on matching algorithms. These subsets are further used by distribution components for parallel matching over available computing nodes and their cores. Results from parallel matching are aggregated from all computing resources and delivered as a bridge ontology mapping file. We have benchmarked our system by incorporating it with an interoperability engine of a CDSS. Its methodology has shown substantial performance speedup over multicore platforms (desktop and cloud) while matching large biomedical ontologies, i.e., FMA, NCI, and SNOMED-CT. It has also outperformed Intra&Inter matcher technique in scalability and performance speedup.

Our presented methodology benefits biomedical professionals, researchers, and systems, which rely on large-scale biomedical ontologies for biomedical resource annotation to provide integration, aggregation, and interoperability. Due to the complex and increasing size of these ontologies, a biomedical client either has to wait or may acquire expensive computational platforms (e.g., high-end servers and HPC clusters) for in-time results. Our methodology provides the resolution to these medical clients by taking care of large and complex biomedical ontologies and performs matching operations in parallel over affordable platforms. The distribution method is built to scale from multicore desktop PCs to ubiquitous and affordable distributed multi-node platforms like clouds for a better performance. For higher accuracy, biomedical ontology matching algorithms may require utilization of external thesauri like UMLS,

PubMed, and WordNet; however, these thesauri are slow in performance due to their high complexity search algorithms. Although SPHeRe cannot intervene in the internal implementation of these algorithms, nevertheless, it can contribute in overcoming this slowness by performance exploitation from parallel hardware.

In the current implementation of SPHeRe, multi-node distribution presumes the distributed environment to be homogenous. However, in case of heterogeneous distributed environment where different nodes have varying computational abilities, current even distribution can lead to idle cores. In future, this limitation can be resolved by intelligent distribution of the matching tasks with consideration of nodes' computational abilities. Furthermore, we plan on extending the implementation of SPHeRe from an application to a platform with service-based access. This extension will facilitate the biomedical experts to integrate their own algorithms for matching large-scale biomedical ontologies over a parallel platform.

ACKNOWLEDGMENTS

Microsoft Azure cloud platform was provided for the development of this research under Microsoft Azure4Research initiative.

DISCLOSURE STATEMENT

No potential conflict of interest was reported by the authors.

FUNDING

This research was supported by Microsoft Research Asia, Beijing, China, under the research grant provided as MSRA Project Award 2013–2014 and MSIP (Ministry of Science, ICT&Future Planning), Korea, under IT/SW Creative research program supervised by the NIPA (National IT Industry Promotion Agency) [grant number NIPA-2013-(H0503-13-1010)]. This work was also supported by the Industrial Core Technology Development Program (Develop of mining core technology exploiting personal big data) [10049079] funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea) and by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) [grant number NRF-2014R1A2A2A01003914].

REFERENCES

1. E. Jiménez-Ruiz, C. Meilicke, B. C. Grau, and I. Horrocks, "Evaluating mapping repair systems with large biomedical ontologies," in *Informal Proceedings of the 26th International Workshop on Description Logics*, Ulm, Germany, 2013.
2. H. López-Fernández, M. Reboiro-Jato, D. Glez-Peña, F. Aparicio, D. Gachet, M. Buenaga, and F. Fdez-Riverola, "BioAnnote: A software platform for annotating biomedical documents with application in medical learning environments," *Comput. Methods Programs Biomed.*, Vol. 111, no. 1, pp. 139–47, Jul. 2013.
3. J. J. Cimino, and X. Zhu, "The practical impact of ontologies on biomedical informatics," *Yearb. Med. Inform.*, Vol. 1, pp. 124–35, 2006.
4. D. Isern, D. Sánchez, and A. Moreno, "Ontology-driven execution of clinical guidelines," *Comput. Methods Programs Biomed.*, Vol. 107, no. 2, pp. 122–39, Aug. 2012.
5. P. De Potter, H. Cools, K. Depraetere, G. Mels, P. Debevere, J. De Roo, C. Huszka, D. Colaert, E. Mannens, and R. Van de Walle, "Semantic patient information aggregation and medicinal decision support," *Comput. Methods Programs Biomed.*, Vol. 108, no. 2, pp. 724–35, Nov. 2012.
6. G. O. Consortium, "The Gene Ontology (GO) database and informatics resource," Vol. 32, no. 1, pp. D258–61, Jan. 2004.
7. J. Golbeck, G. Fragoso, F. Hartel, J. Hendler, J. Oberthaler, and B. Parsia, "The National Cancer Institute's thésaurus and ontology," *Web Semantics Sci. Serv. Agents World Wide Web*, Vol. 1, no. 1, pp. 75–80, Dec. 2003.
8. C. Rosse, and J. L. V. Mejino Jr., "A reference ontology for biomedical informatics: The foundational model of anatomy," *J. Biomed. Inform.*, Vol. 36, no. 6, pp. 478–500, Dec. 2003.
9. S. Schulz, R. Cornet, and K. Spackman, "Consolidating SNOMED CT's ontological commitment," *Appl. Ontol.*, Vol. 1, pp. 1–11, 2011.
10. P. L. Whetzel, N. F. Noy, N. H. Shah, P. R. Alexander, C. Nyulas, T. Tudorache, and M. A. Musen, "BioPortal: Enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications.," Vol. 39, suppl., pp. 541–5, 2011.
11. B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S.-A. Sansone, R. H. Scheuermann, N. Shah, P. L. Whetzel, and S. Lewis, "The OBO Foundry: Coordinated evolution of ontologies to support biomedical data integration," *Nat. Biotechnol.*, Vol. 25, no. 11, pp. 1251–5, Nov. 2007.
12. J. H. Gennari, and A. Silberfein, "Leveraging an alignment between two large ontologies: Fma and go," in *Seventh International Protege Conference*, Bethesda, MD, 2004.
13. W. A. Khan, M. Hussain, M. Afzal, M. B. Amin, M. A. Saleem, and S. Lee, "Personalized-detailed clinical model for data interoperability among clinical standards," *Telemed. e-Health*, Vol. 19, no. 8, pp. 632–42, Jul. 2013. doi:10.1089/tmj.2012.0189.
14. M. Tuttle, D. Sherertz, M. Erlbaum, N. Olson, and S. Nelson, "Implementing meta-1: The first version of the UMLS metathesaurus," in *Proceedings of the Annual Symposium on Computer Application in Medical Care*, Washington, DC, Nov. 1989, p. 483.
15. B. C. Grau, Z. Dragisic, K. E. 0001, J. Euzenat, A. Ferrara, R. Granada, V. Ivanova, E. Jiménez-Ruiz, A. O. Kempf, P. Lambrix, A. Nikolov, H. Paulheim, D. Ritze, F. Scharffe, P. Shvaiko, C. T. dos Santos, and O. Zamazal, "Results of the ontology alignment evaluation initiative 2013," in *Proceedings of the 8th ISWC Workshop on Ontology Matching (OM)*, Sydney, Australia, 2013, pp. 61–100.

16. W. R. van Hage, S. Katrenko, and G. Schreiber, "A method to combine linguistic ontology-mapping techniques," in *The Semantic Web – ISWC 2005*, Vol. 3729, no. 52, Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, Eds. Berlin: Springer, 2005, pp. 732–44.
17. J. Euzenat, and P. Shvaiko, *Ontology Matching*, 2nd ed., Heidelberg (DE): Springer-Verlag, 2013, p. 520.
18. A. Groß, M. Hartung, T. Kirsten, and E. Rahm, "On matching large Life Science ontologies in parallel," in *Data Integration in the Life Sciences*, Vol. 6254, no. 4, P. Lambrix and G. Kemp, Eds. Berlin: Springer, 2010, pp. 35–49.
19. P. Shvaiko, and J. Euzenat, "Ontology matching: State of the art and future challenges," *IEEE Trans. Knowl. Data Eng.*, Vol. 25, no. 1, pp. 158–76, 2013.
20. W. D. Mulia, N. Sehgal, S. Sohoni, J. M. Acken, C. L. Stanberry, and D. J. Fritz, "Cloud workload characterization," *IETE Tech. Rev.*, Vol. 30, no. 5, p. 382, Sep. 2014.
21. S. Kaur, "Cloud computing is like having an infinite credit line!," *IETE Tech. Rev.*, Vol. 29, no. 6, p. 438, 2012.
22. M. B. Amin, R. Batool, W. A. Khan, S. Lee, and E.-N. Huh, "SPHeRe – A performance initiative towards ontology matching by implementing parallelism over cloud platform," *J. Supercomput.*, Vol. 68, no. 1, pp. 274–301, 2014.
23. W. A. Khan, M. B. Amin, A. M. Khattak, M. Hussain, and S. Lee, "System for parallel heterogeneity resolution (SPHeRe) results for OAEI 2013," in *Proceedings of the 8th International Workshop on Ontology Matching co-located with the 12th International Semantic Web Conference {(ISWC} 2013)*, Sydney, Australia, 2013. pp. 184–9.
24. M. Amin, W. Khan, S. Lee, and B. Kang, "Performance-based ontology matching," *Appl. Intell.*, Vol. 43, no. 2, pp. 1–30, 2015.
25. W. A. Khan, A. M. Khattak, M. Hussain, M. B. Amin, M. Afzal, C. Nugent, and S. Lee, "An adaptive semantic based mediation system for data interoperability among health information systems," *J. Med. Syst.*, Vol. 38, no. 8, pp. 1–18, 2014.
26. W. A. Khan, M. B. Amin, A. M. Khattak, M. Hussain, M. Afzal, S. Lee, and E. S. Kim, "Object-oriented and ontology-alignment patterns-based expressive Mediation Bridge Ontology (MBO)," *J. Inform. Sci.*, Vol. 41, no. 3, pp. 296–314, Jun. 2015.
27. M. Hussain, A. M. Khattak, W. A. Khan, I. Fatima, M. B. Amin, Z. Pervez, R. Batool, M. A. Saleem, M. Afzal, M. Faheem, M. H. Saddiqi, S. Y. Lee, and K. Latif, "Cloud-based smart CDSS for chronic diseases," *Health Technol.*, Vol. 3, no. 2, pp. 153–75, 2013.
28. A. Valles, M. Gillespie, and G. Drysdale, *Performance Insights to Intel® Hyper-Threading Technology*. Intel Software Network, Nov. 20, 2010.
29. A. Michalove. (2006). *Amdahls Law*. Available: http://home.wlu.edu/~whaleyt/classes/parallel/topics/amdahl.html
30. P. Lambrix, and H. Tan, "SAMBO – A system for aligning and merging biomedical ontologies," *Web Semantics Sci. Serv. Agents World Wide Web*, Vol. 4, no. 3, pp. 196–206, Sep. 2006.
31. G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, Vol. 38, no. 11, pp. 39–41, Nov. 1995.
32. "SIGNAL-ONTOLOGY: Ontology for cell signaling," *Genome Inform.*, Vol. 11, no. 3, p. 440, Jul. 2011.
33. C. E. Lipscomb, "Medical Subject Headings (MeSH)," *Bull. Med. Library Assoc.*, Vol. 88, no. 3, p. 265, Jul. 2000.
34. T. F. Hayamizu, M. Mangan, J. P. Corradi, and J. A. Kadin, "The adult mouse anatomical dictionary: A tool for annotating and integrating data," *Genome Biol.*, Vol. 6, no. 3, 2005.
35. P. Lambrix, H. Tan, and Q. Liu, "SAMBO and SAMBOdtf: Results for the ontology alignment evaluation initiative 2008," in *Proceedings of the 3rd International Workshop on Ontology Matching (OM-2008) Collocated with the 7th International Semantic Web Conference (ISWC-2008)*, Karlsruhe, Germany, 2008.
36. Y. R. Jean-Mary, E. P. Shironoshita, and M. R. Kabuka, "Ontology matching with semantic verification," *Web Semantics Sci. Serv. Agents World Wide Web*, Vol. 7, no. 3, pp. 235–51, 2009.
37. M. Ba, and G. Diallo, "Large-scale biomedical ontology matching with ServOMap," *IRBM* (*Ingénierie et Recherche Biomédicale*), Vol. 34, no. 1, pp. 56–9, Feb. 2013.
38. I. F. Cruz, F. P. Antonelli, and C. Stroe, "AgreementMaker: Efficient matching for large real-world schemas and ontologies.," *Proc. Very Large Database*, Vol. 2, no. 2, pp. 1586–9, 2009.
39. E. Jiménez-Ruiz, and B. C. Grau, "LogMap: Logic-based and scalable ontology matching," *Int. Semantic Web Conf.*, Vol. 7031, no. 18, pp. 273–88, 2011.
40. T. Kirsten, A. Groß, M. Hartung, and E. Rahm, "GOMMA: A component-based infrastructure for managing and analyzing life science ontologies and their evolution," *J. Biomed. Sem.*, Vol. 2, no. 1, p. 6, 2011.

Authors

Muhammad Bilal Amin received his PhD in computer engineering from Kyung Hee University, South Korea. He did his MS from DePaul University, Chicago, IL, USA in 2006. He is currently a postdoctoral research fellow at Ubiquitous Computing Lab, Department of Computer Engineering, Kyung Hee University, South Korea. He has a working experience of more than 10 years in software industry working for Fortune 500 companies in USA. His research interest include, cloud computing, parallel programming, distributed systems, software architecture, semantic web, and performance-based ontology matching.

E-mail: mbilalamin@oslab.khu.ac.kr

Wajahat Ali Khan received his PhD degree in computer engineering from Kyung Hee University, South Korea. He is currently serving as a postdoctoral research fellow at Ubiquitious Computing Lab, Kyung Hee University, South Korea. He has numerous publications in the area of biomedical healthcare standardization and interoperability in highly reputable journals. His current focus of research includes, ontology mappings, healthcare standards compatibility, SOA, and semantic web services.

E-mail: wajahat.alikhan@oslab.khu.ac.kr

Shujaat Hussain received his BS with distinction from Mohammad Ali Jinnah University, Pakistan in 2005. He got MS from Mohammad Ali Jinnah University, Pakistan in 2009. Since September 2011, he has been working on his PhD degree at the Department of Computer Engineering at Kyung Hee University, Korea. His research interests include cloud computing, health analytics in big data and distributed systems.

E-mail: shujaat.hussain@oslab.khu.ac.kr

Dinh-Mao Bui received his BS degree in computer science from the Computer Engineering Department at Ton Duc Thang University, Vietnam, in 2009 and the MS degree in data communication and networking from the Posts and Telecommunications Institute of Technology, Vietnam, in 2012. He is now working toward the PhD degree in the Department of Computer Engineering at Kyung Hee University, Korea. His research interests include convex optimization, stochastic process and big data.

E-mail: mao.bui@khu.ac.kr

Oresti Banos received his MSc degree in telecommunications engineering in 2009, the MSc degree in computer network engineering in 2010, the MSc in electrical engineering in 2011 and the PhD degree in computer science in 2014, all from the University of Granada, Granada, Spain. He is currently working as a postdoctoral research fellow with the Department of Computer Engineering, Kyung Hee University, Korea. He is a former research associate at the Research Center for Information and Communications Technologies and lecturer with the Department of Computer Architecture and Computer Technology, University of Granada. He has been visiting researcher at several prestigious institutions such as the Swiss Federal Institute of Technology Zurich, Switzerland, 2011; the University of Alabama, USA, 2011; and the Technical University of Eindhoven, The Netherlands, 2012. His main research is on wearable, ubiquitous, pervasive and mobile computing with a particular focus on digital health and wellbeing applications. His expertise also covers pattern recognition and machine learning for probabilistic modeling of human behavior from multi-modal sensor data, information fusion and context-awareness, with a special interest in robust, adaptive and opportunistic expert systems.

E-mail: oresti@oslab.khu.ac.kr

Byeong Ho Kang is an associate professor at Department of Information and Communication Technology, University of Tasmania, Australia. He has numerous years of experience working on industrial and academic projects with many highly cited and reputable publications. His research interests include artificial intelligence, expert systems, knowledge acquisition, and Internet applications.

E-mail: byeong.kang@utas.edu.au

Sungyoung Lee received his BS from Korea University, Seoul, Korea. He got his MS and PhD degrees in computer science from Illinois Institute of Technology (IIT), Chicago, USA in 1987 and 1991 respectively. He has been a professor in the Department of Computer Engineering, Kyung Hee University, Korea since 1993. He is a founding director of the Ubiquitous Computing Laboratory, and has director of Neo Medical ubiquitous-Life Care been affiliated with a Information Technology Research Center, Kyung Hee University since 2006. Before joining Kyung Hee University, he was an assistant professor in the Department of Computer Science, Governors State University, Illinois, USA from 1992 to 1993. His current research focuses on ubiquitous computing and applications, wireless ad-hoc and sensor networks, context-aware middleware, sensor operating systems, real-time systems and embedded systems, activity and emotion recognition. He is a member of ACM and IEEE.

E-mail: sylee@oslab.khu.ac.kr