PhD. Dissertation Presentation

Performance-based Ontology Matching

An Effectiveness-independent Approach for Performance-gain

M. Bilal Amin
mbilalamin@oslab.khu.ac.kr
Dept. of Computer Engineering
Kyung Hee University

Advisor : Prof. Sungyoung Lee
sylee@oslab.khu.ac.kr

Contents.

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Background.(1/2)

- Semantic Heterogeneity
  - The progress of information and communication technologies have created abundance of dissimilar information [1]
  - Semantic Heterogeneity, handling of information variation in meanings and ambiguity is an open challenge [2]

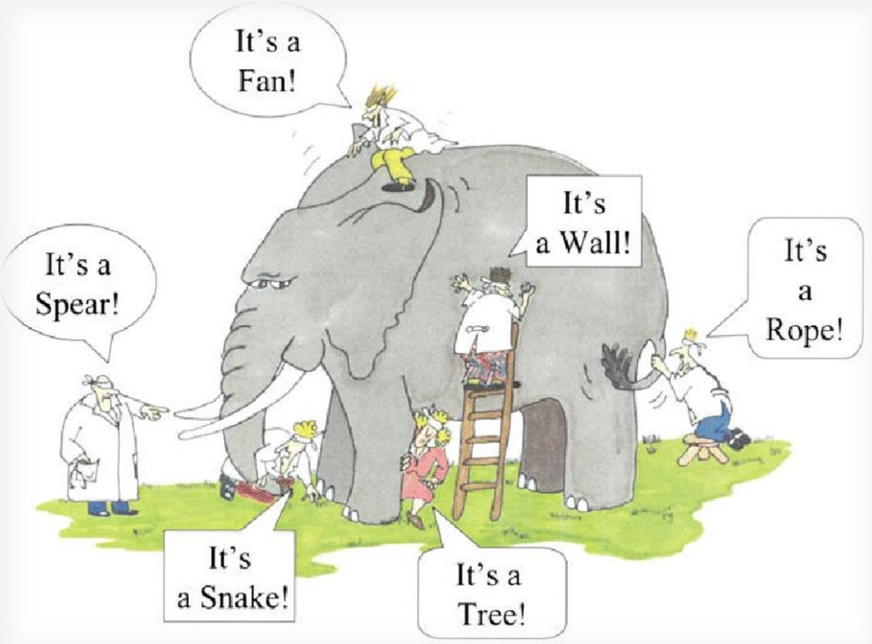
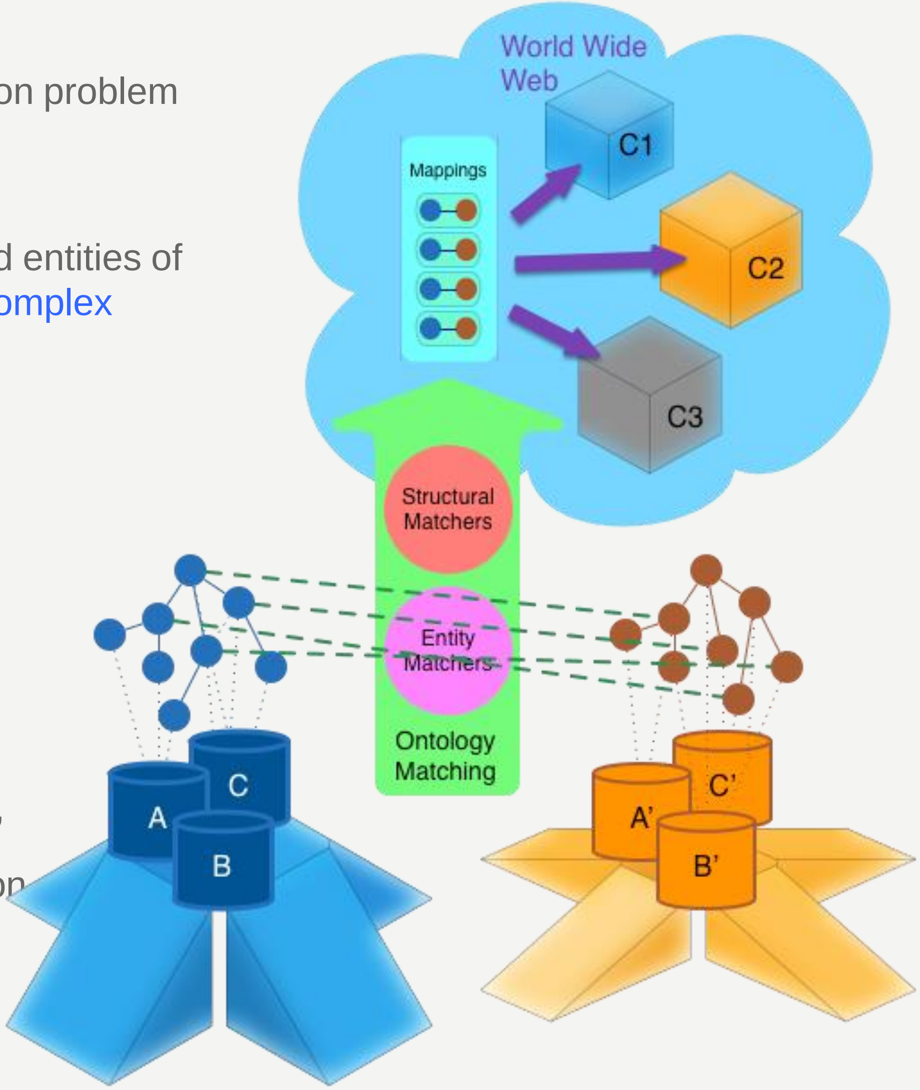


Image from: **André Freitas,**
Crossing the Vocabulary Gap for Querying Complex and Heterogeneous Databases
http://www.slideshare.net/andrenfreitas/crossing-the-vocabulary-gap-for-querying-complex-and-heterogeneous-databases

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Background.(2/2)

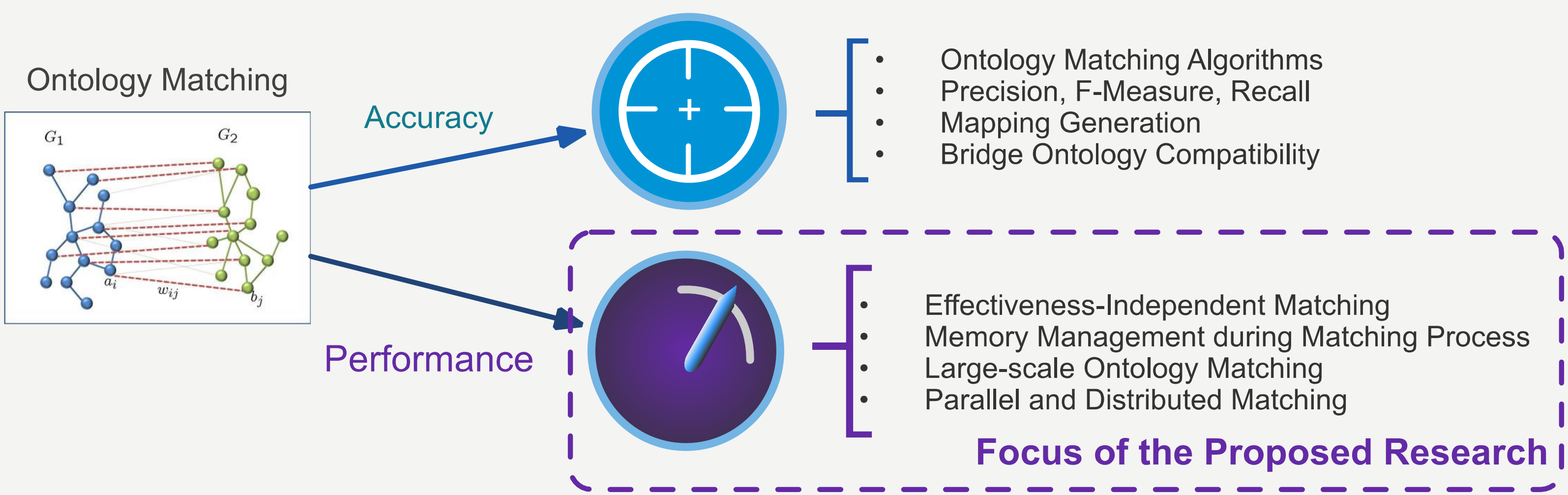
- Ontology Matching
  - Primary solution to the heterogeneity resolution problem heterogeneity resolution problem [1]
  - Resources are annotated by ontologies and correspondence between semantically related entities of these ontologies is determined by library of complex ontology matching algorithms [3]
  - Correspondences are further used for [5][6]
    - Information and e-Commerce systems,
    - Database integration,
    - Semantic-web services,
    - Medical knowledge-bases,
    - Clinical guidelines and Decision making,
    - Medical data formats and Standardization
    - Social networks,
    - Data interoperability,
    - Information translation.



mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Motivation.

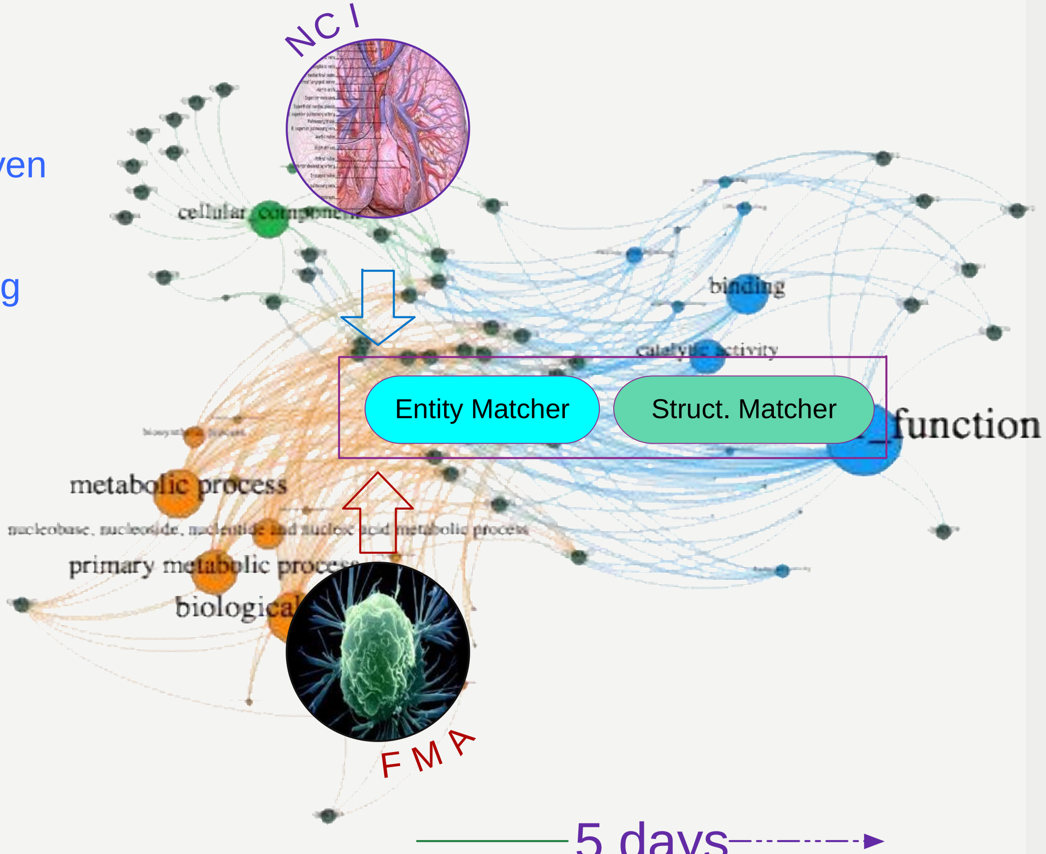
- Due to excess of data, size of the Ontologies have grown and become complex; Consequently, the Ontology Matching has become a computationally intensive task with complexity quadratic or higher [4]
- Shvaiko et. al, "Ontology Matching: State of the Art and Future Challenges"*. **IEEE Transaction on Knowledge and Data Engineering (2013)***, for the first time discussed ontology matching as two-fold problem which requires explicit performance efficiency resolutions for in-time results
- The core techniques for achieving better performance are either related to the optimization of matching algorithms or the fragmentation of ontologies, Parallel and distributed ontology matching is largely unaddressed so far [1]
- Design time nature and delay caused by current monolithic matching techniques makes ontology matching ill-equipped for dynamic systems with in-time result needs [1][6][9]



mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Motivation.(example)

- FMA, NCI Matching problem
  - Two large-scale ontologies with 78 Mb, and 66 Mb owl file size
  - Two matching algorithms
  - Quad-core commodity machine, 8 Gb Memory
  - Impulsive shut-down due to no result even after 5 days
  - Java Heap blow up errors during parsing



mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Problem Statement.

- Ontology matching is the most efficient and used methodology for Semantic Heterogeneity resolution
- Abundance of data has caused Ontologies to grow and become complex; Consequently, matching algorithms have become complex ($> O(n^2)$). As a result, ontology matching is now a computationally intensive task
- Current state-of-the-art resolutions talk about performance in regards with optimization of matching algorithms (effectiveness-dependent resolution), They fail to engage approaches where performance-gain can be achieved without compromising the accuracy (effectiveness-independent performance-gain)
  - For high accuracy, compromise on performance, delay in results making current techniques ill-equipped for clients and systems with in-time requirements
- Current approaches are monolithic, with no collaboration and sharing at service and platform level

- Goal

"To devise one such methodology that identifies the possible bottlenecks of the ontology matching process from end-to-end and provides explicit performance measures for the matching process in a shareable environment such that through out the performance gain, accuracy of the matching process is preserved, thus achieving an effectiveness-independent performance-gain resolution"

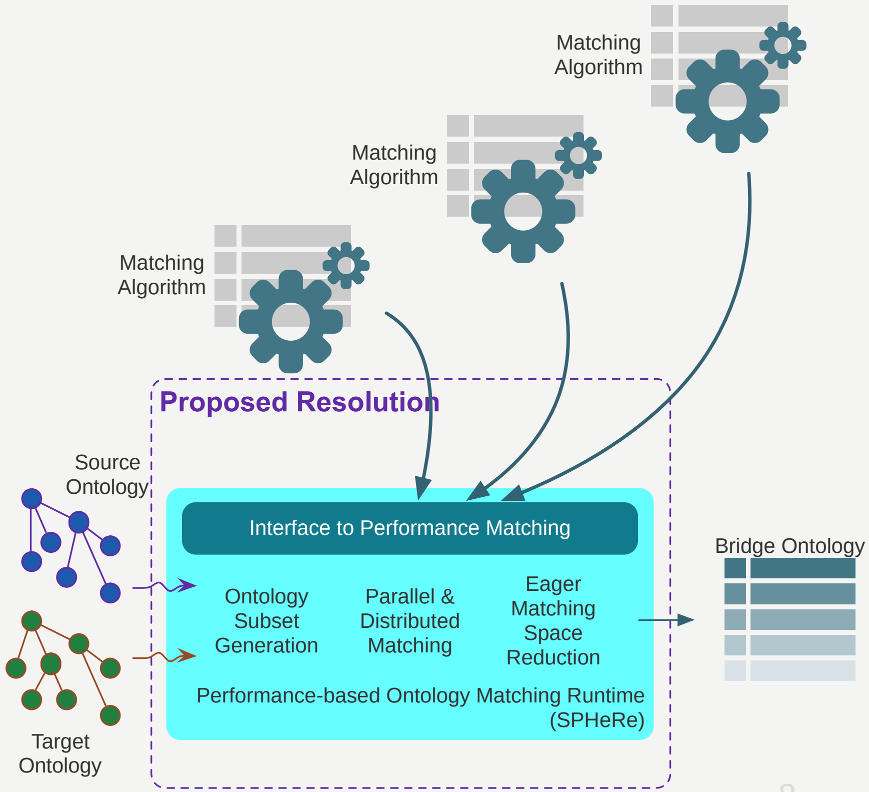
mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Objectives.

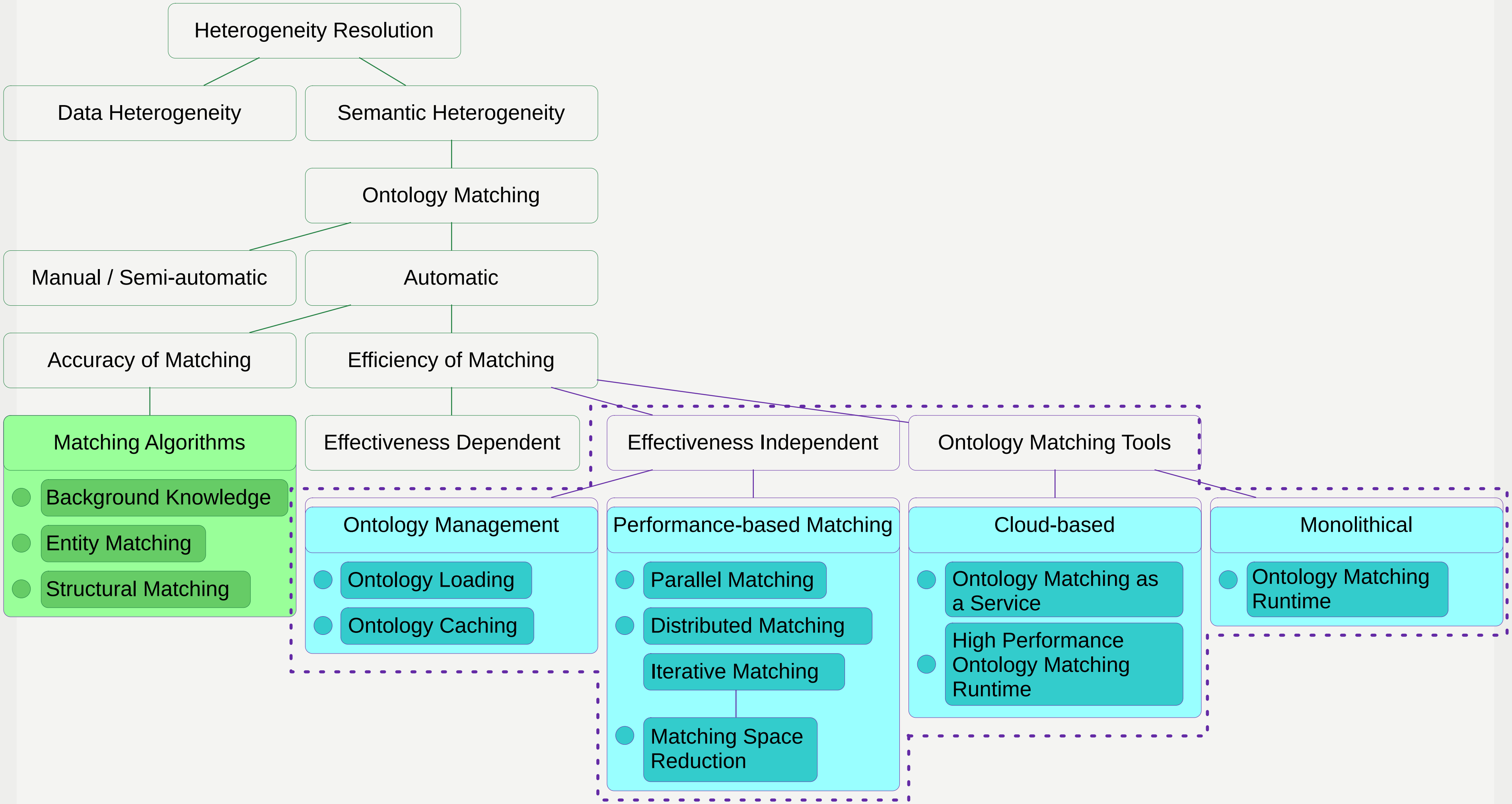
- A performance-efficient solution for accessing ontology resources in the memory without memory stress
- Optimal exploitation of available computational resources for the matching process
- Avoid redundant computationally expensive matching operations through out the matching process
- Presented resolution must be sharable for mapping generation and decoupled matching library execution

• Challenges

- Completion of whole matching process with-in optimal Heap size
- Scalability over available computing cores
- Large-scale ontology matching problems
- Accuracy Preservance through-out the performance-gain (Effectiveness-Independent Resolution)



Research Taxonomy.



mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Related Work.(1/2)

Performance-Requirement Matrix

Proposed Methodology
in comparison with
OAEI Ranked System
(2006-2014)

	1. Domain Independent	2. Accuracy Preservance	3. Design Time Support	4. Soft-real-time Support	5. Matching Library	6. Large-scale Ontology Matching Support	7. Monolithic Runtime	8. Shareable as a Service	9. Parallel and Distributed Matching	10. Scalability	11. Memory Stress and Footprint Reduction
1. AgrMaker [10]	✓	✗	✓	✗	✓ coupled	✓	✓	✗	✗	▼	✗
2. AROMA [11]	✓	✗	✓	✗	✓ coupled	?	✓	✗	✗	−	✗
3. ASMOV [12]	✗	✗	✓	✗	✓ coupled	✗	✓	✗	✗	▼	✗
4. CODI [13]	✓	✗	✓	✗	✓ coupled	?	✓	✗	✗	▼	✗
5. CSA [14]	✓	✗	✓	✗	✓ coupled	?	✓	✗	✗	▼	✗
6. Falcon-AO [15]	✓	✗	✓	✗	✓ coupled	✓	✓	✗	✗	▼	✗
7. GOMMA [16]	✓	✗	✓	✗	✓ coupled	✓	✓	✗	✓ !	▲	✗
8. Hadoop-MapReduce	✓	✓	✓	✗	✗	✓ !	✗	✓ platform	✓ !	▲	!
9. Lily [17]	✓	✗	✓	✗	✓ coupled	?	✓	✗	✗	▼	✗
10. LogMap [18]	✓	✗	✓	✗	✓ coupled	✓	✓	✗	✗	−	✗
11. MAPSSS [19]	✓	✗	✓	✗	✓ coupled	?	✓	✗	✗	▼	✗
12. MassMtch [20]	✓	✗	✓	✗	✓ coupled	?	✓	✗	✗	−	✗
13. SAMBO [21]	✗	✗	✓	✗	✓ coupled	✓	✓	✗	✗	▼	✗
14. ServOMap [22]	✗	✗	✓	✗	✓ coupled	✓	✓	✓ platform	✗	▼	✗
Proposed Methodology [5]	✓	✓	✓	✓	✓ decoupled	✓	✓	✓ software ✓ platform	✓	▲	✓

The sequence numbers do not reflect the chronological order of ranking

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Related Work.(2/2)

- The performance aspect of the current ontology matching systems is tightly coupled with the accuracy and complexity of matching algorithms
- Their implemented resolutions are more focused on optimization of the matching algorithms and partitioning of larger ontologies into smaller chunks for performance benefits
- Increase the Heap-Memory for Large-scale matching problems
- A clear distinction between the resolutions for accuracy and performance does not exist
- Redundant matching operations with no workflow-based execution
- An explicit and decoupled runtime has not been proposed yet which can improve the performance factors without inflicting any changes in the effectiveness of matching algorithms
- These resolutions fall into the category of effectiveness-dependent solutions where a trade-off between matching effectiveness (accuracy measures, precision, recall, and F-Measure) and execution time (performance) exists
- The performance improvement based-on exploitation of newer hardware technologies has largely been missed

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

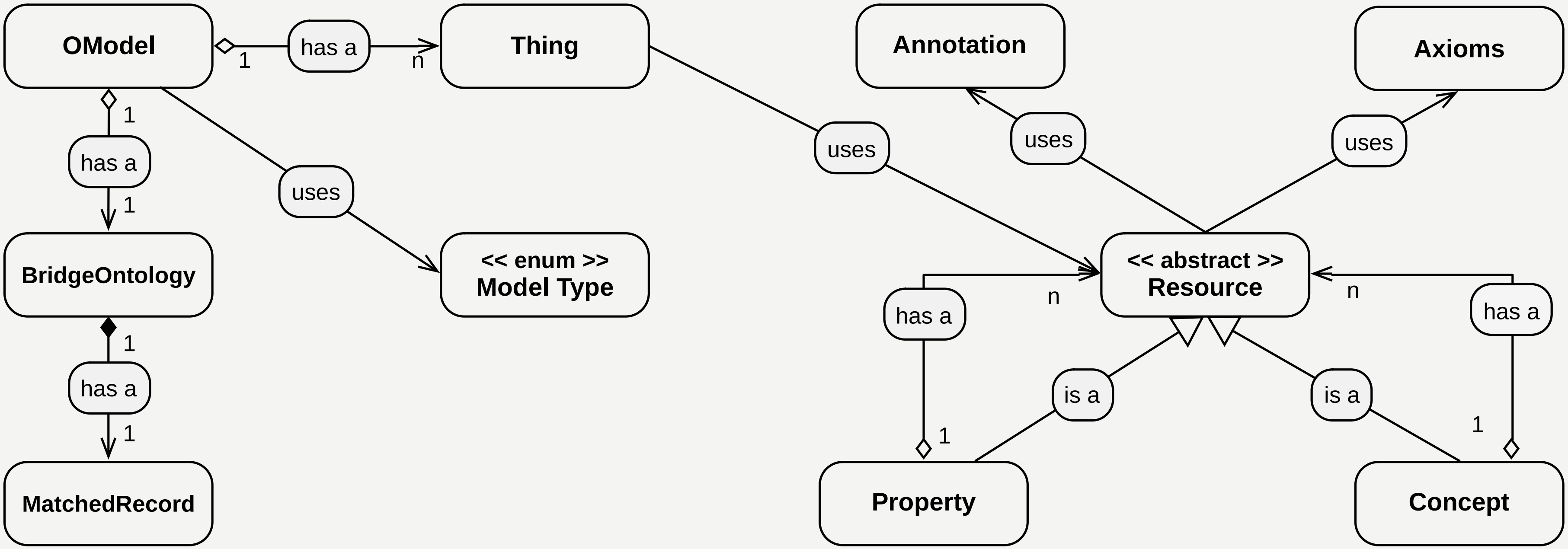
Proposed Methodology.

	Limitations	Proposed Solution	Objectives
1.	• Lack of Performance-efficient Ontology Model, (Jena and OWLApi are used) • Whole Ontology Load with Memory stress and Heap Blowups	• Generic, yet concise Ontology Model with Caching, re-usability, and multi-threading support • Matching Algorithm-based Ontology Subsets Creation and Loading for Parallel Matching	A performance-efficient solution for accessing ontology resources in the memory without memory stress
2.	• Subtle increase in performance with better hardware • Ill-equipped to perform Parallel and Distributed Matching for effectiveness independent performance-gain	• Parallel and Distributed Ontology Matching platform with abstractions defined from grainer to finer level of Matching Process	Optimal exploitation of available computational resources for the matching process
3.	• Late checking for redundant bridge instances	• Aligned execution workflow for Eager Matching Space Reduction	Avoid redundant computationally expensive matching operations through out the matching process
4.	• Monolithic implementations with no sharing at service or platform level • Effectiveness-dependent solutions	• Cloud-based runtime, Ontology Matching as a Service, as a Platform • Effectiveness-independent resolution	Presented resolution must be sharable for mapping generation and decoupled matching library execution

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Solution 1(1/4) : Matching Algorithm-based Ontology Subset generation.

UML Conceptual Representation of Ontology Model



Differences from Existing Approaches

- Current Systems and approaches use Jena and OWLApi for Ontology Models
- Reduced structural complexity of the ontology
- Supports Multithreading by Mutable and Immutable Objects
- Caching of Serialized objects for faster ontology loading

Salient Features and Benefits

1. Performance-oriented data structures
3. Supports thread-safety and Parallel Ontology read
4. Evaluated by experts for accuracy and comprehensiveness
5. No re-parsing for pre-cached Ontologies

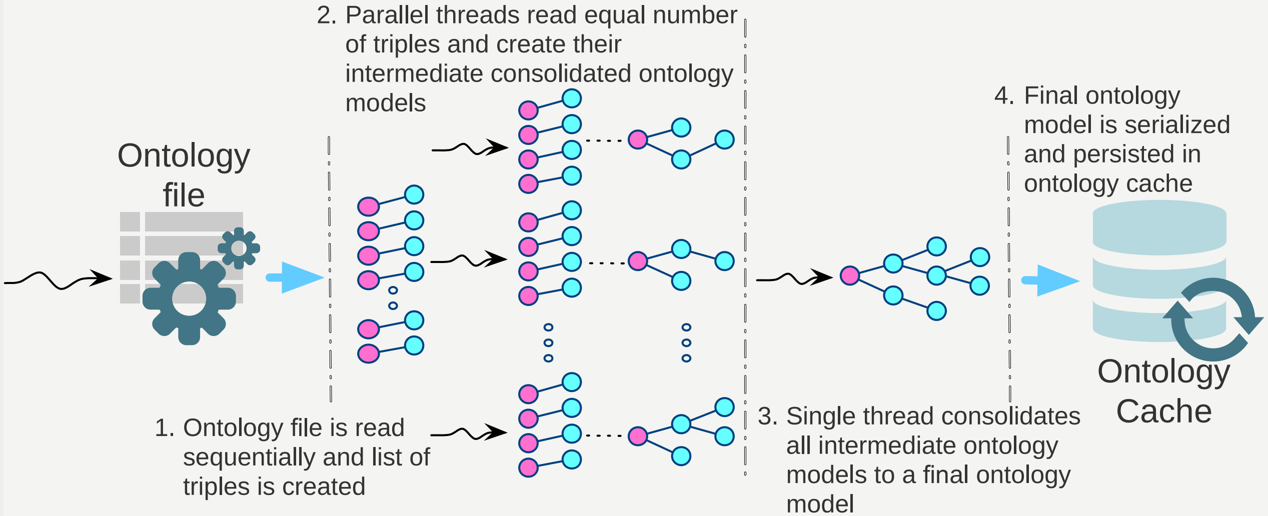
Related Publication

- Muhammad Bilal Amin, Rabia Batool, Wajahat Ali Khan, Sungyoung Lee, Eui-Nam Huh. (2014). SPHeRe: A Performance Initiative Towards Ontology Matching by Implementing Parallelism over Cloud Platform, Journal of Supercomputing, 68(1), 274–301.

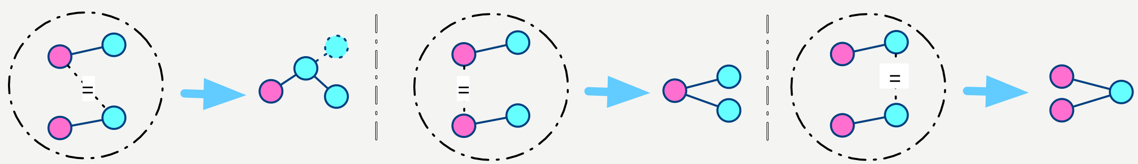
mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Solution 1(2/4) : Matching Algorithm-based Ontology Subset generation.

Bottom-up Ontology Parsing and Hierarchy Consolidation Algorithm



Consolidation Conditions



$$i \quad \epsilon \quad Algorithms \quad : \quad Algorithms = \{String, Label, Properties, \ldots, Child\}$$

$$O_x^i \leftarrow f^i(x) \quad : \quad x \quad \epsilon \quad \{source, target\}$$

$$O_x = \bigcup_{j=1}^{n} O_x^j \quad : \quad n = NumberOfAlgorithms$$

Differences from Existing Approaches

- Current Systems and approaches use Jena and OWLApi for Ontology Access
- Subsets based on Matching Algorithm instead of fragmentation or late-binding queries
- Ontology subsets are cached by custom serializers and deserializers for faster load

Salient Features and Benefits

1. Subsets based on executing algorithms represented as Ontology Model and serialized into Ontology Cache
2. Subsets are generated without redundancy
3. Parallel deserialization of subsets for Parallel and distributed matching
4. Subsets required are only loaded reducing the memory stress
5. Completion of matching process without Heap Overflow (within 2Gb of Heap Size)

Related Publication

- Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong-Ho Kang (2015). Performance-based Ontology Matching, A data-parallel approach for an effectiveness-independent performance-gain in ontology matching. Applied Intelligence DOI 10.1007/s10489-015-0648-z
- Muhammad Bilal Amin, Rabia Batool, Wajahat Ali Khan, Sungyoung Lee, Eui-Nam Huh. (2014). SPHeRe: A Performance Initiative Towards Ontology Matching by Implementing Parallelism over Cloud Platform, Journal of Supercomputing, 68(1), 274–301.

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Solution 1$_{(3/4)}$: Matching Algorithm-based Ontology Subset generation.

Algorithm 1 Method owlLoad

```
Require: O_s ≠ NULL and O_t ≠ NULL
  Hash_s ← Utility.calculateHash(O_s)
  Hash_t ← Utility.calculateHash(O_t)
  ontologyCache ← OntologyCache.getInstance()
  parser ← Parser.createInstance()
  if Hash_s, Hash_t !in ontologyCache then
    parser.parse(O_s, O_t)
    parser.serialize(O_s, O_t)
  else
    if Hash_t !in ontologyCache and Hash_s in ontologyCache then
      parser.parse(O_t)
      parser.serialize(O_t)
    else if Hash_s !in ontologyCache and Hash_t in ontologyCache then
      parser.parse(O_s)
      parser.serialize(O_s)
    end if
  end if
  deserialize(Hash_s, Hash_t)
  return
```

Algorithm 2 Method nameParser

```
Require: O_x ≠ NULL, x ∈ {s, t}
  thing ← Thing.createInstance(url)
  while O_x has classes do
    concept ← OClass.createInstance(currentClassName)
    thing.addConcept(&concept)
  end while
  return thing
```

Algorithm 3 Method labelParser

```
Require: O_x ≠ NULL, x ∈ {s, t}
  thing ← Thing.createInstance(url)
  while O_x has classes do
    concept ← OClass.createInstance(currentClassName)
    while currentClass has labels do
      label ← Annotation.createLabel(labelName)
      concept.addAnnotation(&label)
    end while
    thing.addConcept(&concept)
  end while
  return thing
```

Related Publication

- Muhammad Bilal Amin, Rabia Batool, Wajahat Ali Khan, Sungyoung Lee, Eui-Nam Huh. (2014). SPHeRe: A Performance Initiative Towards Ontology Matching by Implementing Parallelism over Cloud Platform, Journal of Supercomputing, 68(1), 274–301.

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Solution 1$_{(4/4)}$: Matching Algorithm-based Ontology Subset generation.

Algorithm 4 Method propertyParser

Require: $O_x \neq NULL, x \in \{s, t\}$

```
thing ← Thing.createInstance(url)
while O_x has classes do
  concept ← OClass.createInstance(currentClassName)
  while currentClass has properties do
    property ← OProperty.createInstance(propertyName)
    concept.addProperty(&property)
  end while
  thing.addConcept(&concept)
end while
return thing
```

Algorithm 5 Method hierarchyParser

Require: $O_x \neq NULL, x \in \{s, t\}$

```
thing ← Thing.createInstance(url)
while O_x has classes do
  concept ← OClass.createInstance(currentClassName)
  while currentClass has parents do
    if !thing.exists(parent) then
      parent ← OClass.createInstance(parentName)
      thing.addConcept(&parent)
    else
      parent ← thing.getConcept(parentName)
    end if
    concept.addConcept(&parent)
  end while
  thing.addConcept(&concept)
end while
return thing
```

Related Publication

- Muhammad Bilal Amin, Rabia Batool, Wajahat Ali Khan, Sungyoung Lee, Eui-Nam Huh. (2014). SPHeRe: A Performance Initiative Towards Ontology Matching by Implementing Parallelism over Cloud Platform, Journal of Supercomputing, 68(1), 274–301.

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Solution $2_{(1/4)}$: Parallel and Distributed Ontology Matching.

Matching Task (MT) Definition

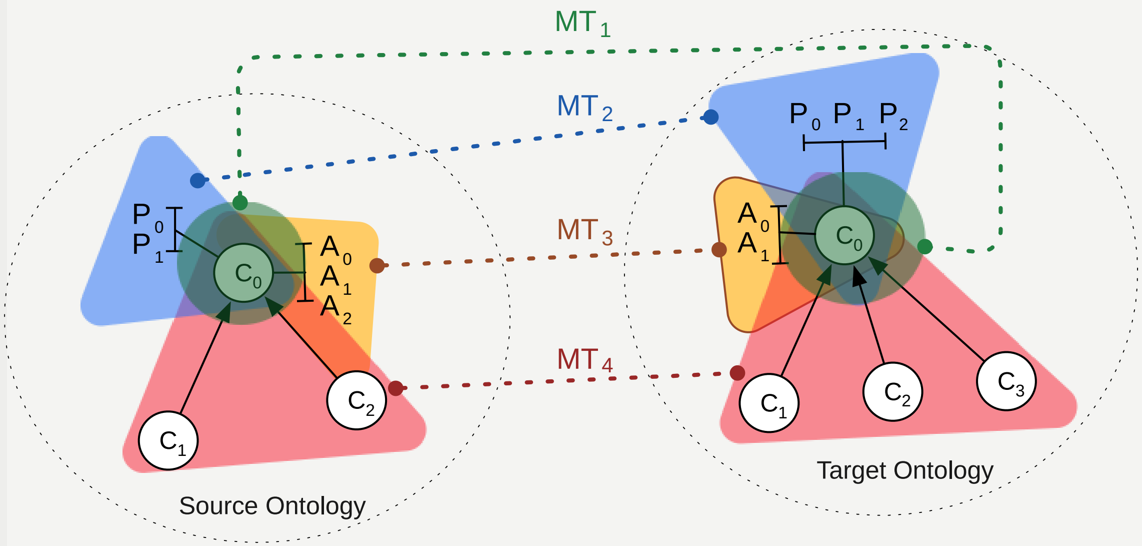
MT is the unit of matching process; defined as, a single independent execution of a matching algorithm over a resource from source (O_S) and target ontologies (O_T)

$$MT = f(m, n, MatchingAlgorithm_i)$$

$$MT_i \bigcap MT_{i+1} \bigcap MT_{i+2} \ldots\ldots \bigcap MT_n = \phi$$

$$MT_{Total} \geq m \times n \quad \forall \quad m \;\epsilon\; O_s \;\; \& \;\; n \;\epsilon\; O_t$$

$$MT_{Core} \leftarrow \frac{MT_{Total}}{Cores_{Total}}$$



Differences from Existing Approaches

- Current Systems and approaches do not implement any parallel and distributed ontology matching methodologies
- Adding more computational resources directly impacts the overall performance of the matching process

Salient Features and Benefits

1. Highly efficient for medium to large-scale ontology matching problem
2. Independent Matching Task, leading to no communication overhead during matching process
3. Data parallelism implementation by thread-level parallelism
4. Size-based partitioning of matching tasks at finer-level for optimal computing resource utilization

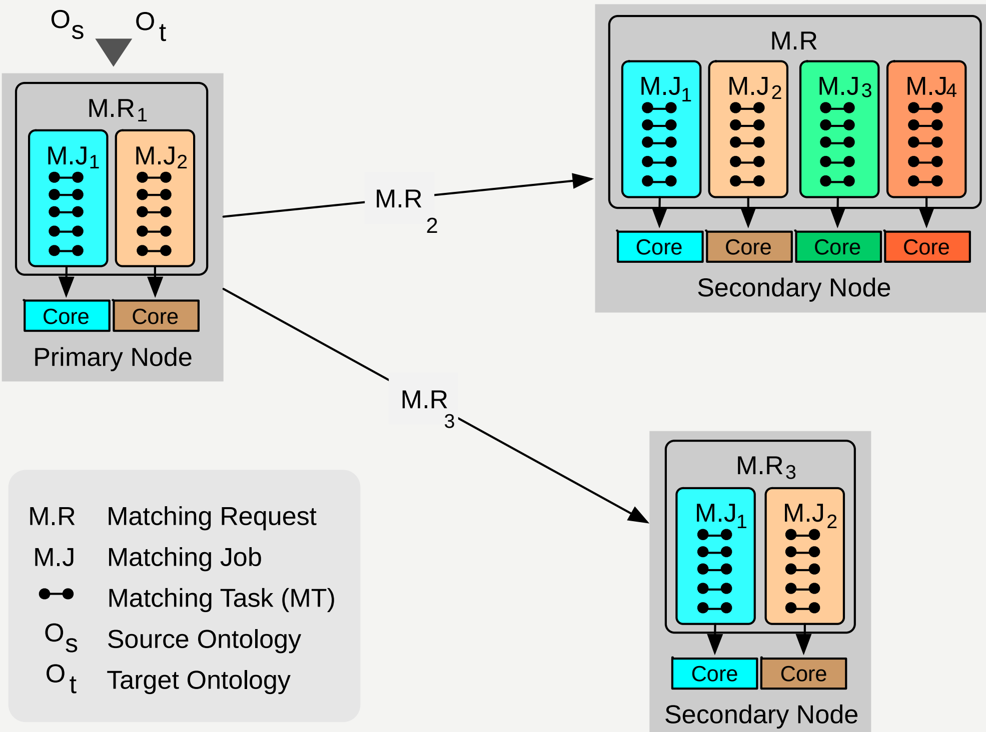
Related Publications

- Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong-Ho Kang (2015). Performance-based Ontology Matching, A data-parallel approach for an effectiveness-independent performance-gain in ontology matching. Applied Intelligence DOI 10.1007/s10489-015-0648-z
- Muhammad Bilal Amin, Rabia Batool, Wajahat Ali Khan, Sungyoung Lee, Eui-Nam Huh. (2014). SPHeRe: A Performance Initiative Towards Ontology Matching by Implementing Parallelism over Cloud Platform, Journal of Supercomputing, 68(1), 274–301. doi:10.1007/s11227-013-1037-1

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Solution 2$_{(2/4)}$: Parallel and Distributed Ontology Matching.

Distribution Abstractions



$$MR \leftarrow \sum_{i=1}^{n} MR_i \quad : \quad n = TotalNodes$$

$$MR_i \leftarrow \sum_{i=1}^{c} MJ_i \quad : \quad c = TotalCoresPerNode$$

$$MJ_i \leftarrow \left\{ \bigcup_{i=1}^{t} MT_i \right\} \quad : \quad t = TotalTasksPerCore$$

Differences from Existing Approaches

- Current Systems and approaches do not implement any parallel and distributed ontology matching methodologies
- Adding more computational resources directly impacts the overall performance of the matching process

Salient Features and Benefits

1. Highly efficient for medium to large-scale ontology matching problem
2. Independent Matching Task, leading to no communication overhead during matching process
3. Data parallelism implementation by thread-level parallelism
4. Size-based partitioning of matching tasks at finer-level for optimal computing resource utilization

Related Publications

- Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong-Ho Kang (2015). Performance-based Ontology Matching, A data-parallel approach for an effectiveness-independent performance-gain in ontology matching. Applied Intelligence DOI 10.1007/s10489-015-0648-z

- Muhammad Bilal Amin, Rabia Batool, Wajahat Ali Khan, Sungyoung Lee, Eui-Nam Huh. (2014). SPHeRe: A Performance Initiative Towards Ontology Matching by Implementing Parallelism over Cloud Platform, Journal of Supercomputing, 68(1), 274–301. doi:10.1007/s11227-013-1037-1

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Solution 2$_{(3/4)}$: Parallel and Distributed Ontology Matching.

Algorithm 1 Generate socket table

```
Require: node ≥ 2,
  temp = 1
  uuidMsg ← generateUUID()
  cores ← Runtime.getNumberofCores()
  rank ← getRankforThisNode()
  while temp < ShiftLeft(1, nodes) do
     if temp ≥ nodes then
        stop
     end if
     sender = rank
     receiver ←XOR(rank, temp)
     socket ←getSocket(receiver)
     if sender > receiver then
        sendMessage(socket, uuidMsg, cores)
        socketTable.add(socket,receiveMessage(socket))
     else
        socketTable.add(socket,receiveMessage(socket))
        sendMessage(socket, uuidMsg)
     end if
     temp = temp + 1
  end while
```

Algorithm 2 Distributor algorithm

```
Require: nodes > 0
  if nodes=1 then
     MulticoreDistributor(O_S, O_T)
  else
     Multi-nodeDistributor(O_S, O_T)
  end if
```

Related Publications

- Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong-Ho Kang (2015). Performance-based Ontology Matching, A data-parallel approach for an effectiveness-independent performance-gain in ontology matching. Applied Intelligence DOI 10.1007/s10489-015-0648-z

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Solution 2$_{(4/4)}$: Parallel and Distributed Ontology Matching.

Algorithm 3 Multicore distributor algorithm

```
Require: nodes > 0
  cores ←Runtime.getNumberOfCores()
  if nodes=1 then
      start=0
      bigOnt ← (size_S ≥ size_T)?O_S : O_T
      smallOnt ← (size_S < size_T)?O_S : O_T
      Partition_slab = ⌈bigOnt.size/cores⌉
      SPAWN MATCHER THREADS:
      for doi = 1 to cores do
          end = start + Partition_slab
          if end ≤ bigOnt.size then
              end = bigOnt.size
          end if
          MatchingJob.create(MatchingTasks[start,
          end), big, small, matcher)
          thread.run(matchingJob)
          start = end
      end for
  else
      RECEIVE MATCHING REQUEST:
      controlMessage.receive(matchingRequest)
      Partition_slab = (end − start)/cores
      GOTO SPAWN MATCHER THREADS
  end if
```

Algorithm 4 Multi-node distributor algorithm

```
Require: nodes > 1
  nodes ←initDaemon.getNoOfNodes()
  participatingCores = ∑ node.#cores
  start=0
  end=0
  bigOnt ← (size_S ≥ size_T)?O_S : O_T
  smallOnt ← (size_S < size_T)?O_S : O_T
  Distribution_slab = ⌈bigOnt.size/
  participatingCores⌉
  for node ← nodes do
      end = start + Distribution_slab × node.#cores
      if end ≤ bigOnt.size then
          end = bigOnt.size
      end if
      MatchingRequest.create([start, end), big, small,
      matcher)
      if node.isLocal then
          local.MulticoreDistributor(matchingRequest)
      else
          controlMessage.send(matchingRequest)
      end if
      start = end
  end for
```

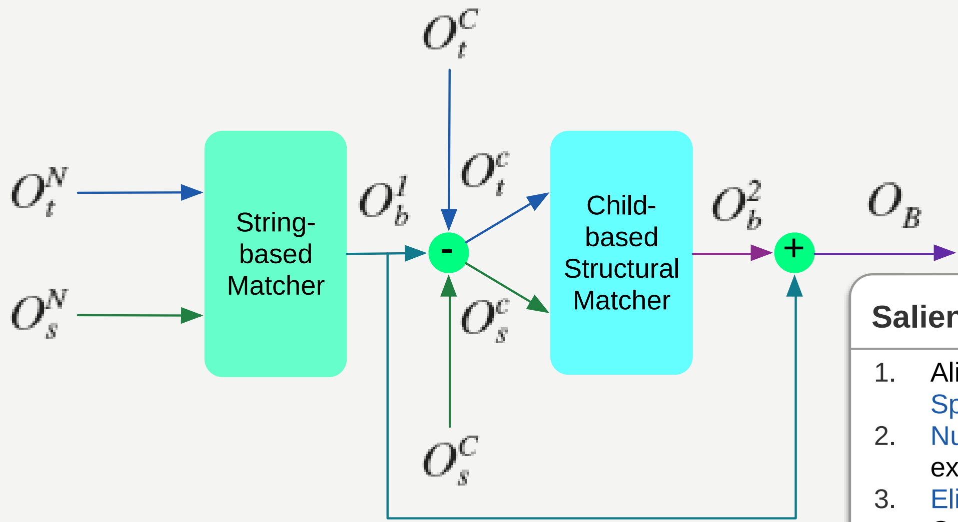
Related Publications

- Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong-Ho Kang (2015). Performance-based Ontology Matching, A data-parallel approach for an effectiveness-independent performance-gain in ontology matching. Applied Intelligence DOI 10.1007/s10489-015-0648-z

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Solution 3 : Eager Matching Space Reduction.

Algorithm Sequence to Minimize the Matching Space



$$f^n(O_s, O_t) \Longrightarrow f^n\left(O_s, O_t, O_b^{n-1}\right)$$

$$O_b^1 \leftarrow (m \times n)_{i=1} \quad \forall \quad i \quad \epsilon \quad Algorithms, \quad m \quad \epsilon \quad O_x^i \quad \& \quad n \quad \epsilon \quad O_t^i$$

$$O_B \leftarrow \bigcup_{i=2}^{t}\left(\left(m^i - \left(m^i \bigcap O_b^{i-1}\right)\right) \times \left(n^i - \left(n^i \bigcap O_b^{i-1}\right)\right)\right) \quad | \quad O_b^i \quad \geq \quad O_b^{i+1}$$

Differences from Existing Approaches

- Eager Matching Space Reduction vs. Late Redundant Bridge checking
- Aligned Matching Algorithm execution. Algorithm with most of the candidate bridge instance executes first and the sequence follows

Salient Features and Benefits

1. Aligns the execution of Matching Algorithms to minimize the Matching Space
2. Number of expensive Matching Operations is reduced as they only execute on ontology resources that are still unmatched
3. Eliminates the chances of redundant matches in the final Bridge Ontology
4. Overall Matching Performance during run-time is improved

Related Publication

- Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong-Ho Kang (2015). Performance-based Ontology Matching, A data-parallel approach for an effectiveness-independent performance-gain in ontology matching. Applied Intelligence DOI 10.1007/s10489-015-0648-z

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Solution 4 (1/2) : Ontology Matching Runtime as a Service and a Platform.

Differences from Existing Approaches

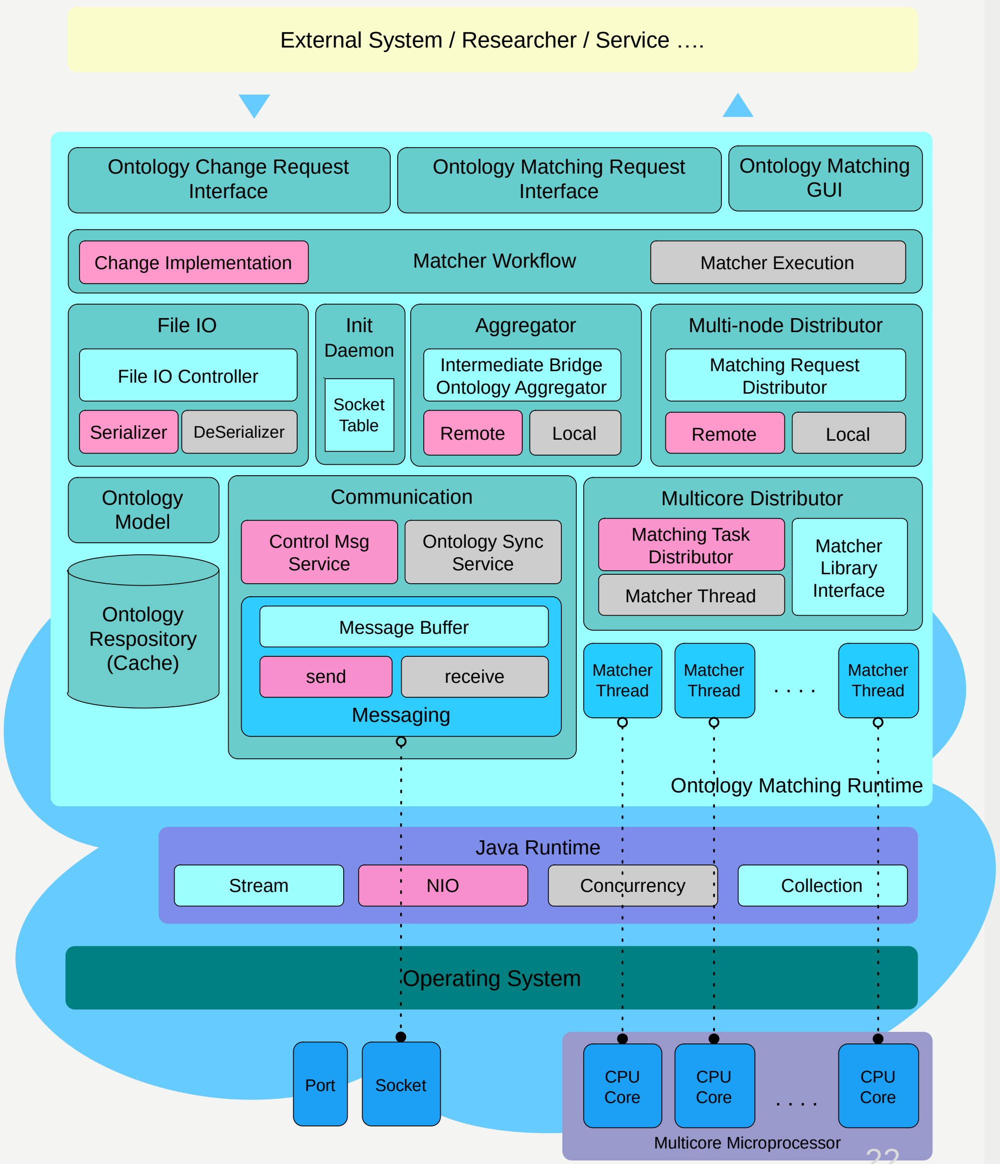
- Decoupled platform and runtime built for performance aspects of ontology matching
- Support for parallel and distributed matching
- Can work as a monolithic implementation and dedicated ontology matching platform
- Built with Cloud aspects (Virtual Machines) in consideration

Salient Features and Benefits

1. Decoupled Performance Platform from Ontology Matching Algorithms
2. Parallel serializers and deserializers for ontology subset loading and persistence
3. Support for local parallel matching by multicore distributors
4. Support for distributed parallel matching by multi-node distributors
5. High Performance Socket-based communication for Candidate Ontology subset replications and repository synchronization
6. Thread-level parallelism for parallel matching
7. Interface for Ontology Matching Request via Ontology Matching as a Service (SaaS)
8. Share-ability by Service, Platform, and Results

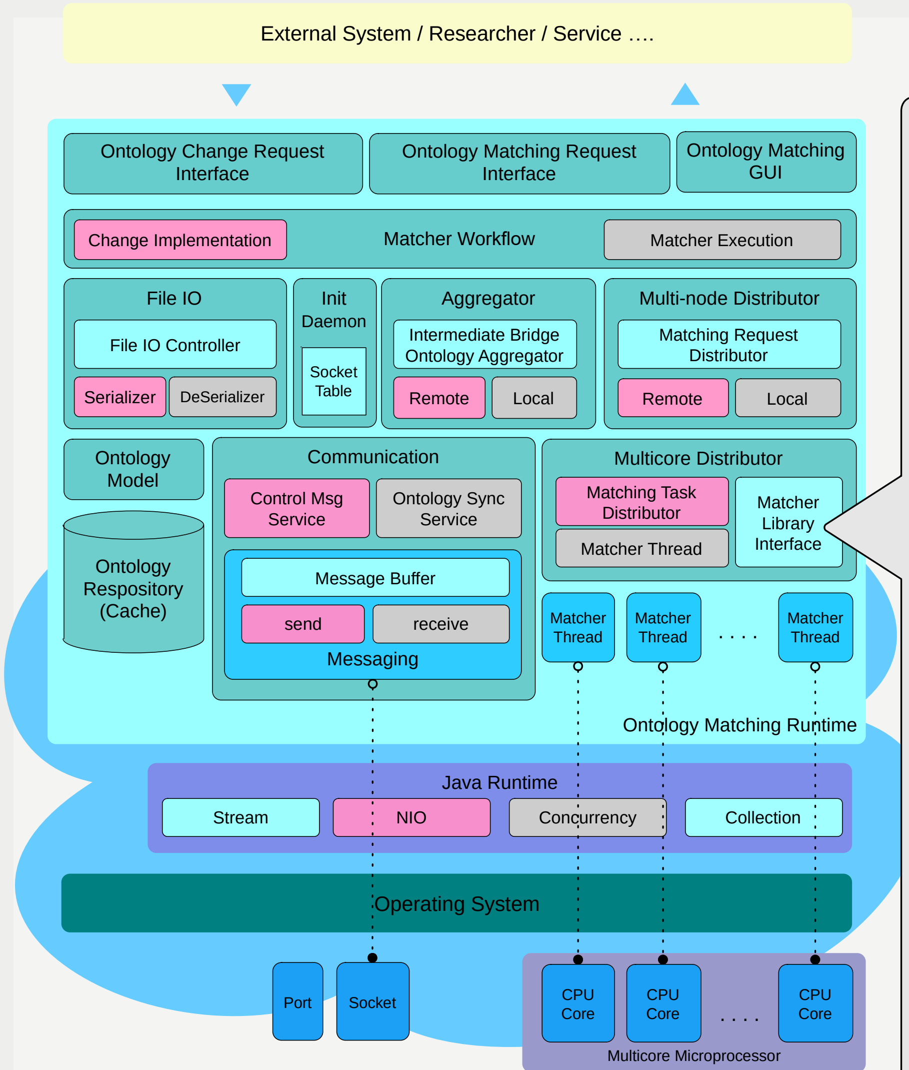
Related Publications

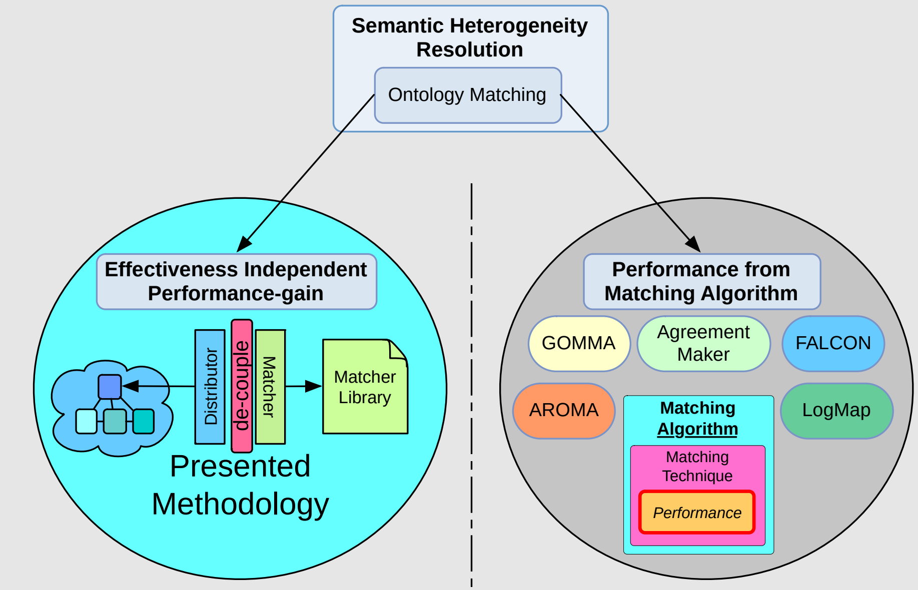
- Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong-Ho Kang (2015). Performance-based Ontology Matching, A data-parallel approach for an effectiveness-independent performance-gain in ontology matching. Applied Intelligence DOI 10.1007/s10489-015-0648-z
- Muhammad Bilal Amin, Rabia Batool, Wajahat Ali Khan, Sungyoung Lee, Eui-Nam Huh. (2014). SPHeRe: A Performance Initiative Towards Ontology Matching by Implementing Parallelism over Cloud Platform, Journal of Supercomputing, 68(1), 274–301. doi:10.1007/s11227-013-1037-1
- Muhammad Bilal Amin, Aamir Shafi, Shujaat Hussain, Wajahat Ali Khan, Sungyoung Lee, High Performance Java Sockets for Scientific Health Clouds, 14th International Conference on e-Health Networking, Applications and Services (Healthcom 2012), Beijing, China
- Muhammad Bilal Amin, Wajahat Ali Khan, Asad Masood Khattak, Maqbool Hussain, Sungyoung Lee, System for Parallel Heterogeniety Resolution (SPHeRe) 2013 OAEI results, ISWC Ontology Matching Workshop, Sydney 2012.



mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Solution 4(2/2) : Ontology Matching Runtime as a Service and a Platform.





Algorithm Matching Interface Methods

```
1. match ( sourceOntologyURI/File, targetOntologyURI/File )

2. match ( sourceOntologyURI/File, targetOntologyURI/File,
   returnEmail)

3. match ( sourceOntologyURI/File, targetOntologyURI/
   File,matchingAlgorithms [ ])

5. match ( sourceOntologyURI/File, targetOntologyURI/File,
   matchingAlgorithms [ ], returnEmail)
```

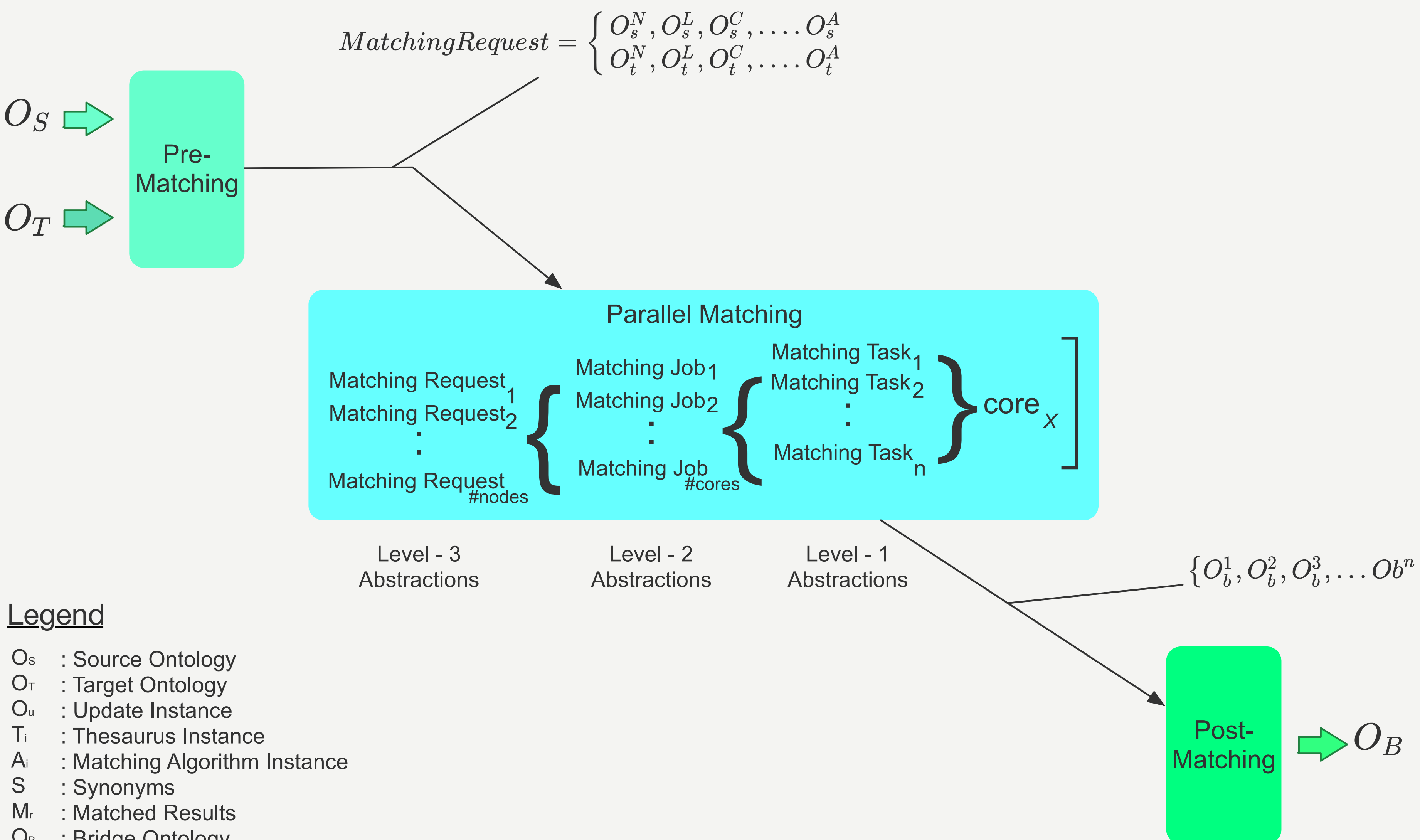
Related Publications

- Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong-Ho Kang (2015). Performance-based Ontology Matching, A data-parallel approach for an effectiveness-independent performance-gain in ontology matching. Applied Intelligence DOI 10.1007/s10489-015-0648-z
- Muhammad Bilal Amin, Mahmood Ahmad, Wajahat Ali Khan, Sungyoung Lee, Biomedical Ontology Matching as a Service, ICOST 2014, Advances in Cognitive Technologies, Denver Colorado USA; 06/2014

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Over-all Execution flow.

High Level Representation



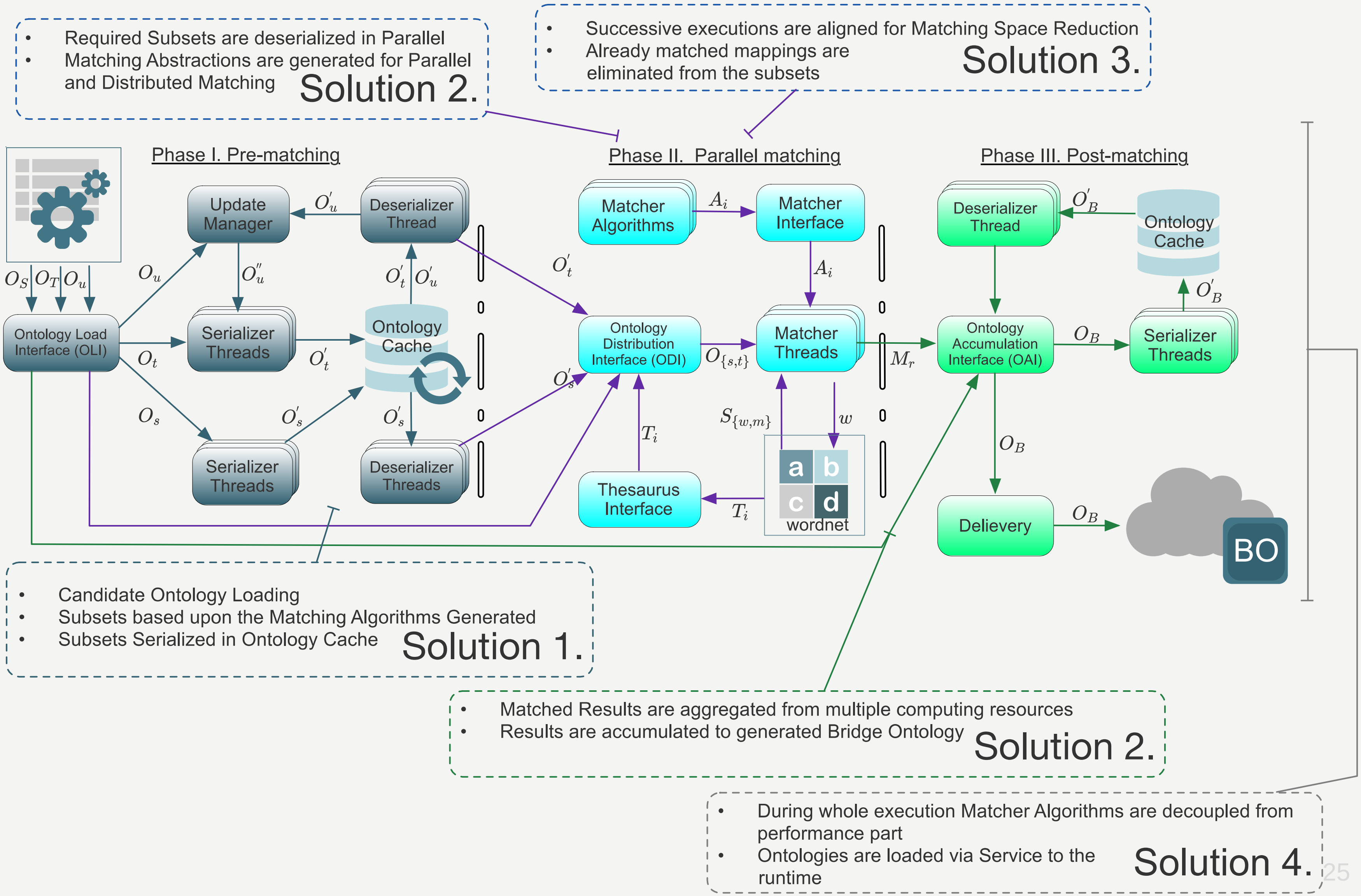
Legend

O_S : Source Ontology
O_T : Target Ontology
O_u : Update Instance
T_i : Thesaurus Instance
A_i : Matching Algorithm Instance
S : Synonyms
M_r : Matched Results
O_B : Bridge Ontology

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Over-all Execution flow.

Detail Level Representation



mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Loading Time Comparison

Dataset Source

- OAEI 2012-2013 Standard Evaluation Dataset of Real-world Ontologies
- Candidate Ontologies:
  FMA = 78,989 concepts
  NCI = 66,724 concepts
  SNOMED = 49,622 concepts

Testbed

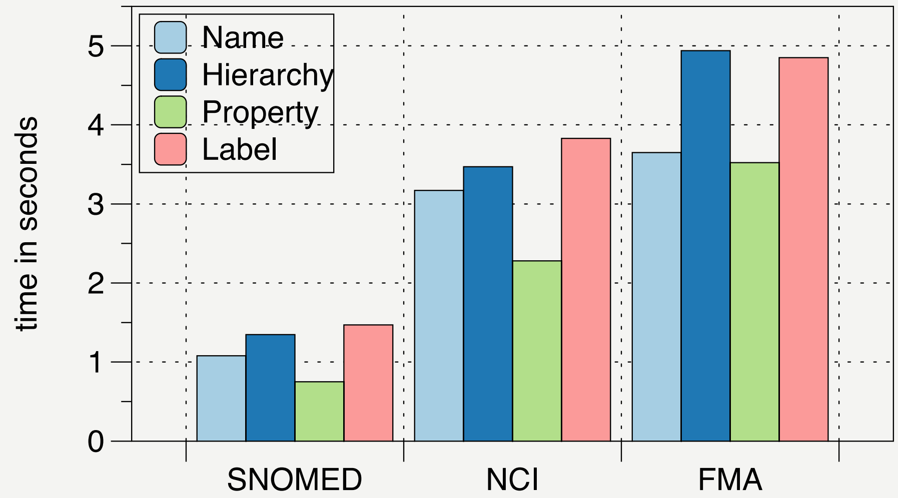
- Cloud:
  Tri-node private cloud equipped with Intel(R) Core(TM) i7 CPU, 16 GB memory with Xen Hypervisor, Java 1.6

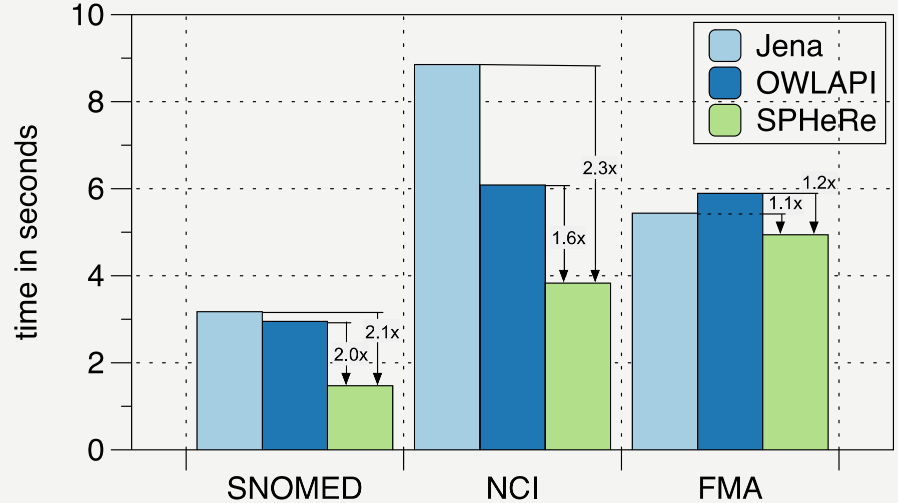
Description

- This evaluation is in Regards to Solution 1
- ~2.5x fast Ontology Loading from Serialized Subsets as compared to Jena and OWL Api

Published In

- Muhammad Bilal Amin, Rabia Batool, Wajahat Ali Khan, Sungyoung Lee, Eui-Nam Huh, SPHeRe, A Performance Initiative Towards Ontology Matching by Implementing Parallelism over Cloud Platform, Jr. of Supercomputing (SCI, IF:0.95) (2014) 68:274-301





mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Evaluation Results.(2/19)

Memory Stress Comparison

Dataset Source

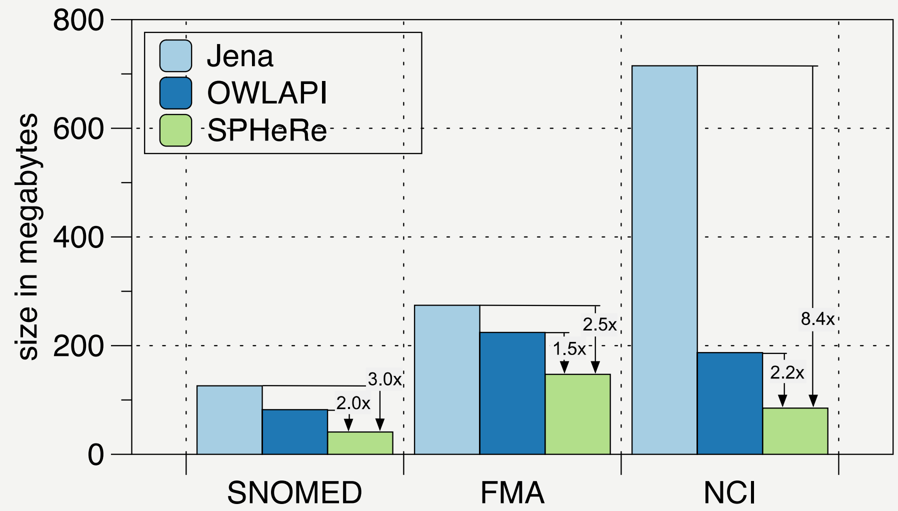
- OAEI 2012-2013 Standard Evaluation Dataset of Real-world Ontologies
- Candidate Ontologies:
  FMA = 78,989 concepts
  NCI = 66,724 concepts
  SNOMED = 49,622 concepts

Testbed

- Cloud:
  Tri-node private cloud equipped with Intel(R) Core(TM) i7 CPU, 16 GB memory with Xen Hypervisor, Java 1.6

Description

- This evaluation is in Regards to Solution 1
- Memory stress varies from Ontology's complexity, reduced memory foot print from 1.5x to 8.4x as compared to Jena and OWLApi



Published In

- Muhammad Bilal Amin, Rabia Batool, Wajahat Ali Khan, Sungyoung Lee, Eui-Nam Huh, SPHeRe, A Performance Initiative Towards Ontology Matching by Implementing Parallelism over Cloud Platform, Jr. of Supercomputing (SCI, IF:0.95) (2014) 68:274-301

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

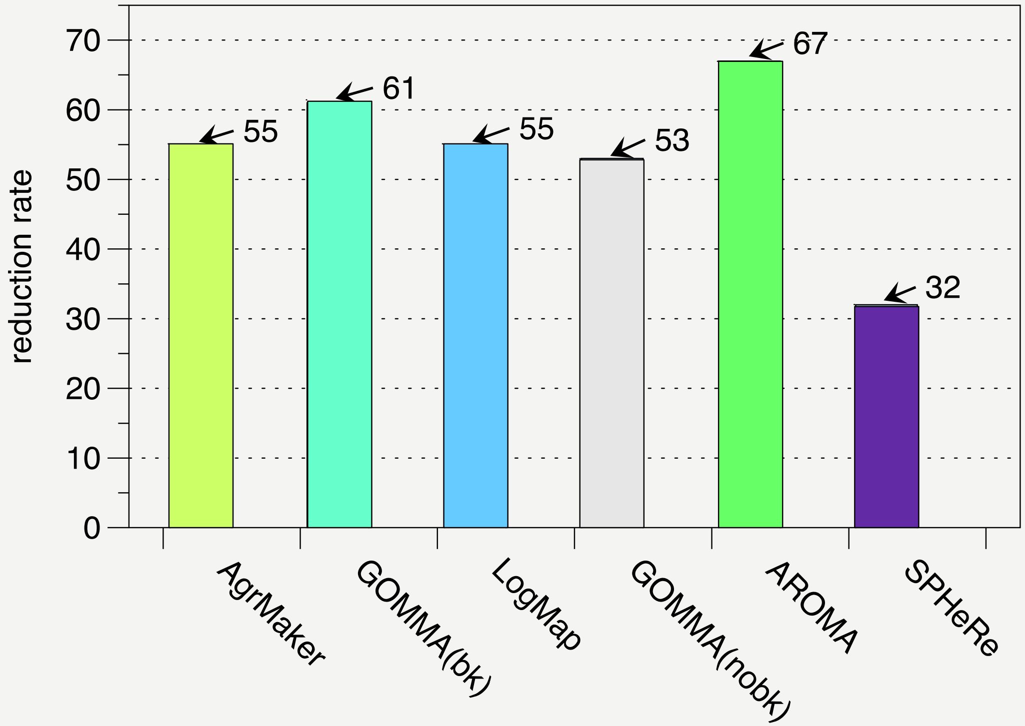
Reduction Rate

Dataset Source

- OAEI 2012-2013 Standard Evaluation Dataset of Real-world Ontologies
- Candidate Ontologies:
  MA = 2,000 concepts
  FMA = 78,989 concepts
  NCI = 66,724 concepts
  SNOMED = 49,622 concepts

Testbed

- Cloud:
  Tri-node private cloud equipped with Intel(R) Core(TM) i7 CPU, 16 GB memory with Xen Hypervisor, Java 1.6



Description

- This evaluation is in Regards to Solution 2
- Measures the scalability aspect of the system
- Measures the optimal resource utilization by a system
- OAEI standard reduction formula is used
- ~40% better reduction score than GOMMA

Published In

- Muhammad Bilal Amin, Rabia Batool, Wajahat Ali Khan, Sungyoung Lee, Eui-Nam Huh, SPHeRe, A Performance Initiative Towards Ontology Matching by Implementing Parallelism over Cloud Platform, Jr. of Supercomputing (SCI, IF:0.95) (2014) 68:274-301

Evaluation Results.(4/19)

Performance Comparison with GOMMA

Dataset Source

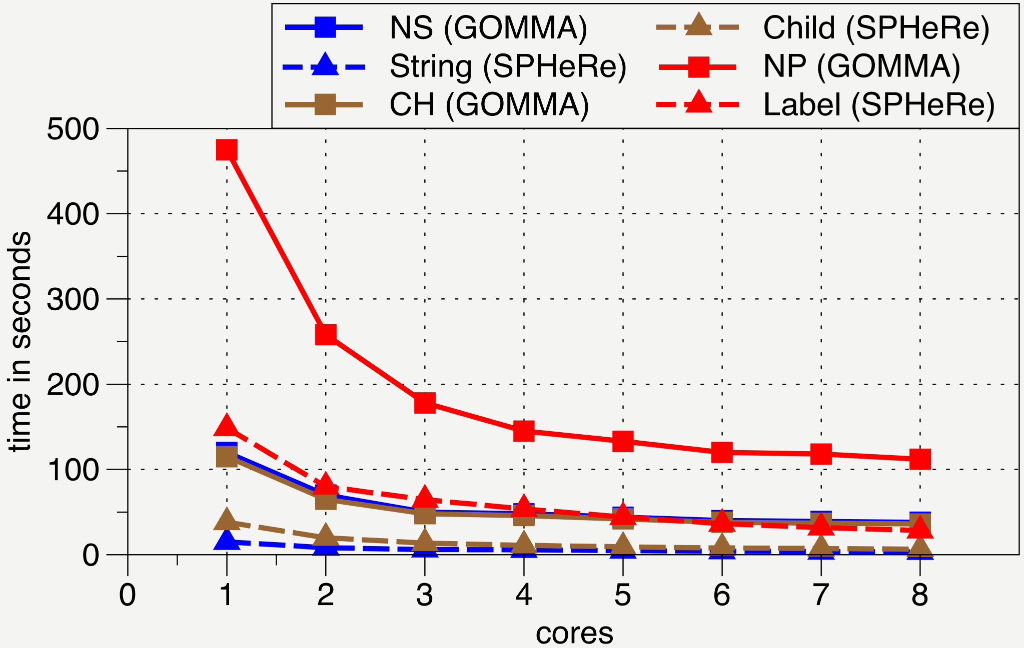
- Candidate Ontologies:
  MA = 2,737 concepts (OAEI 2012)
  NCI = 3,298 concepts (OAEI 2012)
  MF = 9,395 concepts (GO, 2009)
  BP = 17,104 concepts (GO, 2009)

Testbed

- Cloud:
  Tri-node private cloud equipped with Intel(R) Core(TM) i7 CPU, 16 GB memory with Xen Hypervisor, Java 1.6

Description

- This evaluation is in Regards to Comparison with most performance efficient ontology matching system GOMMA
- ~4 times better performance than GOMMA



Published In

- Muhammad Bilal Amin, Rabia Batool, Wajahat Ali Khan, Sungyoung Lee, Eui-Nam Huh, SPHeRe, A Performance Initiative Towards Ontology Matching by Implementing Parallelism over Cloud Platform, Jr. of Supercomputing (SCI, IF:0.95) (2014) 68:274-301

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

OAEI Anatomy Track

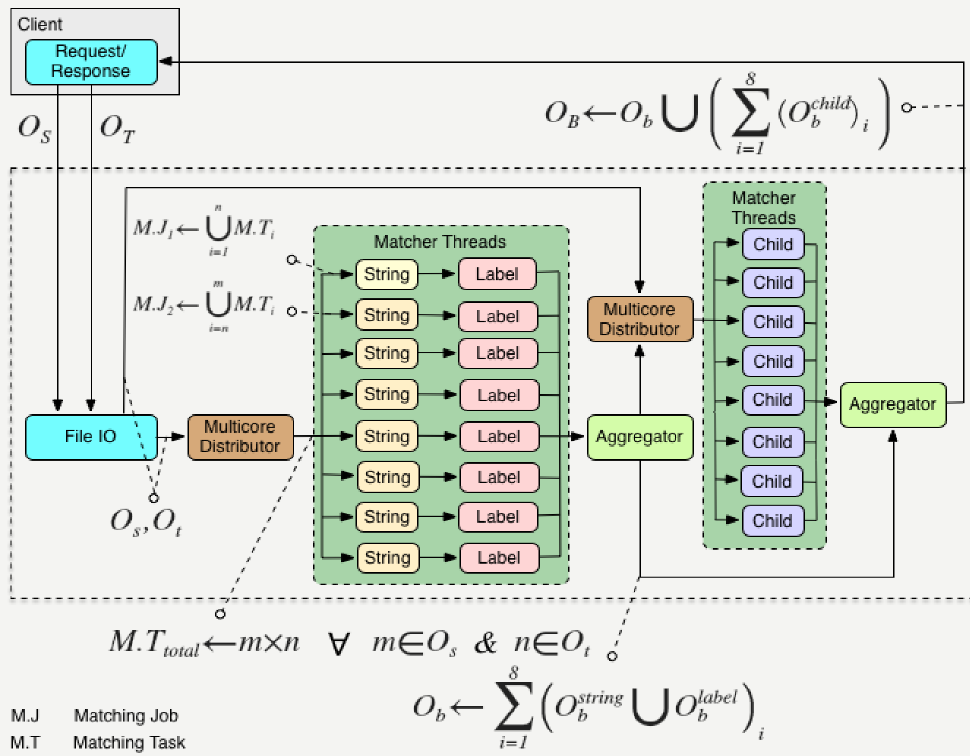
Dataset Source

- OAEI 2013-2014 Standard Evaluation Dataset of Real-world Ontologies
- Matching Library: String-Label-ChildBased
- Magnitude: Medium-scale (MT > 27 Million)
- Candidate Ontologies:
  Adult Mouse Anatomy = 2,744 concepts
  NCI Thesaurus = 3,304 concepts

Testbed

- Multicore Desktop:
  3.4 GHz Intel(R) Core i7(R) Hyper-Threaded (Intel(R) HT Technology) CPU (2 threads/ core) with 16 GB memory, Java 1.8 and Windows 7 64 bit OS
- Cloud:
  Microsoft Azure standard A4 VM instances with 8 cores, 14 GB of memory, Java 1.8, and Windows 2012 R2 Guest OS running over an AMD Opteron(TM) 2.1 GHz CPU

Execution Scenario



Description

- This evaluation is in Regards to Overall Performance particularly Solution 3, 4
- Designed by OAEI experts for trivial and non-trivial mappings
- 4x performance-gain over desktop
- 5.5x performance-gain over cloud node
- Accuracy measures stay preserved through-out the process

Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

OAEI Anatomy Track

Dataset Source

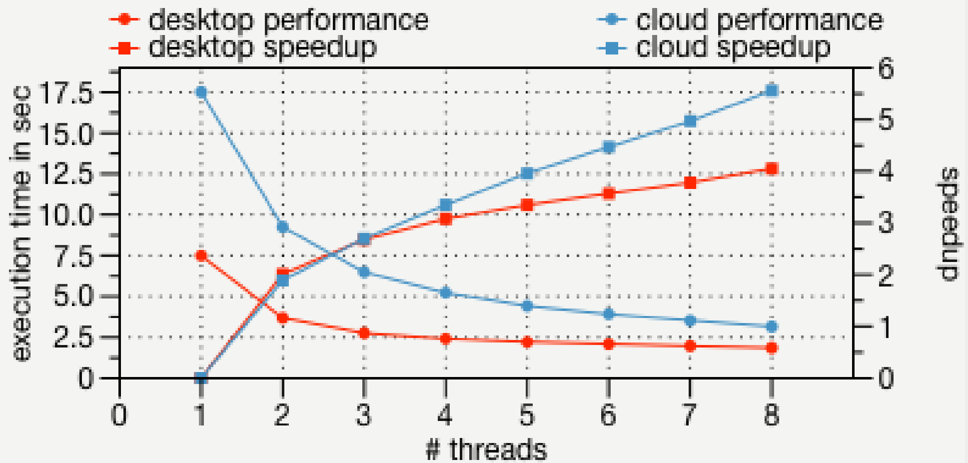
- OAEI 2013-2014 Standard Evaluation Dataset of Real-world Ontologies
- Matching Library: String-Label-ChildBased
- Magnitude: Medium-scale (MT > 27 Million)
- Candidate Ontologies:
  Adult Mouse Anatomy = 2,744 concepts
  NCI Thesaurus = 3,304 concepts

Testbed

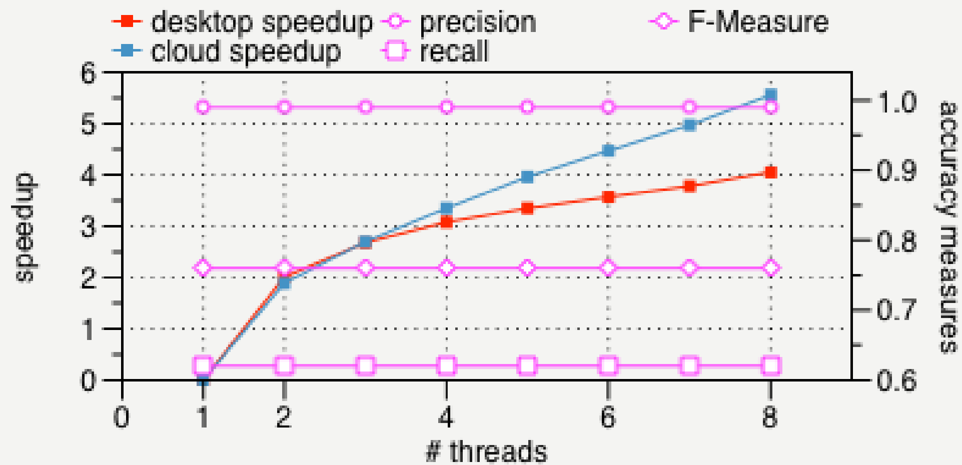
- Multicore Desktop
  3.4 GHz Intel(R) Core i7(R) Hyper-Threaded (Intel(R) HT Technology) CPU (2 threads/ core) with 16 GB memory, Java 1.8 and Windows 7 64 bit OS
- Cloud:
  Microsoft Azure standard A4 VM instances with 8 cores, 14 GB of memory, Java 1.8, and Windows 2012 R2 Guest OS running over an AMD Opteron(TM) 2.1 GHz CPU

Description

- This evaluation is in Regards to Overall Performance particularly Solution 3, 4
- Designed by OAEI experts for trivial and non-trivial mappings
- 4x performance-gain over desktop
- 5.5x performance-gain over cloud node
- Accuracy measures stay preserved through-out the process



(a) Performance-speedup



(b) Speedup-matching effectiveness

Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

Evaluation Results.(7/19)

OAEI Library Track

Dataset Source

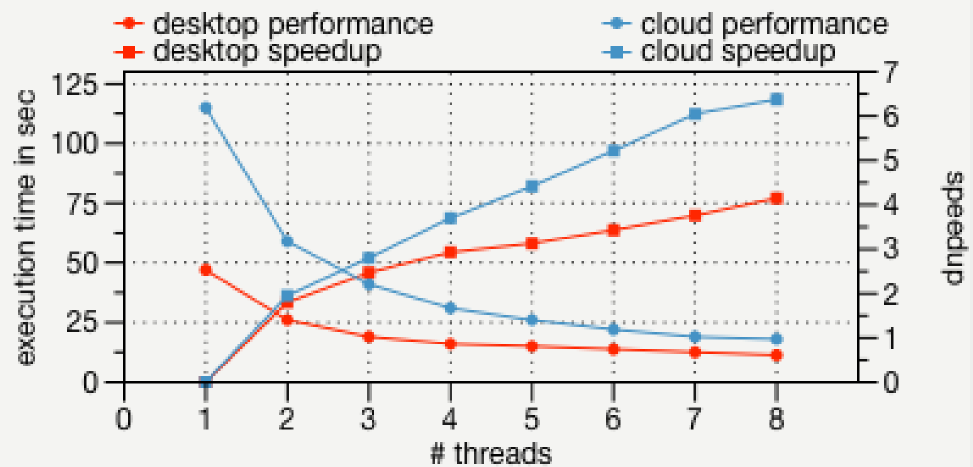
- OAEI 2013-2014 Standard Evaluation Dataset of Real-world Ontologies
- Matching Library: String-Label-ChildBased
- Magnitude: Medium to Large scale (MT > 165 Million)
- Candidate Ontologies:
  STW = 8,376 concepts
  TheSoz = 6,575 concepts

Testbed

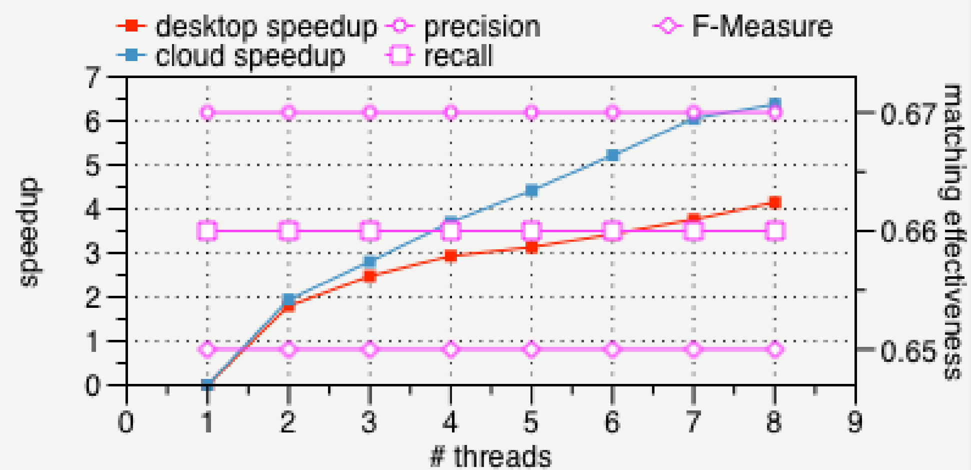
- Multicore Desktop:
  3.4 GHz Intel(R) Core i7(R) Hyper-Threaded (Intel(R) HT Technology) CPU (2 threads/ core) with 16 GB memory, Java 1.8 and Windows 7 64 bit OS
- Cloud:
  Microsoft Azure standard A4 VM instances with 8 cores, 14 GB of memory, Java 1.8, and Windows 2012 R2 Guest OS running over an AMD Opteron(TM) 2.1 GHz CPU

Description

- This evaluation is in Regards to Overall Performance particularly Solution 3, 4
- Used for Library Indexation and retrieval
- 4.15x performance-gain over desktop
- 6.38x performance-gain over cloud node
- Accuracy measures stay preserved through-out the process



(a) Performance-speedup



(b) Speedup-matching effectiveness

Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Evaluation Results.(8/19)

OAEI Large-scale Biomedical Track: task 1

Dataset Source

- OAEI 2013-2014 Standard Evaluation Dataset of Real-world Ontologies
- Matching Library: String-Annotation-ChildBased
- Magnitude: Medium to Large scale (MT > 71 Million)
- Candidate Ontologies:
  FMA = 3,696 concepts
  NCI = 6,488 concepts

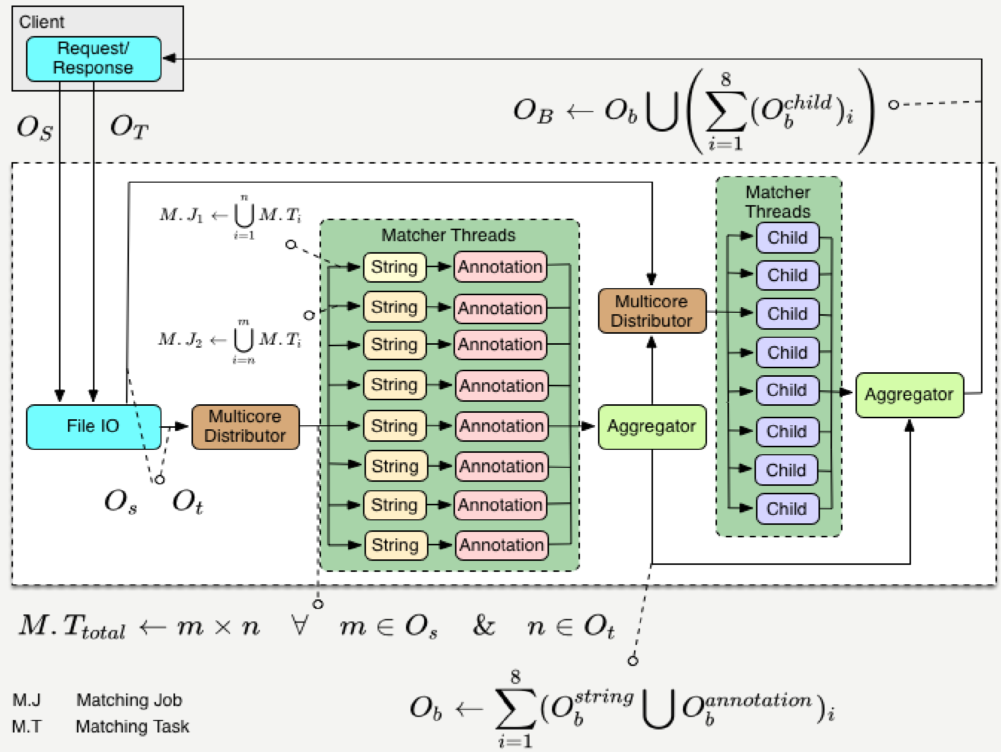
Testbed

- Multicore Desktop:
  3.4 GHz Intel(R) Core i7(R) Hyper-Threaded (Intel(R) HT Technology) CPU (2 threads/ core) with 16 GB memory, Java 1.8 and Windows 7 64 bit OS
- Cloud:
  Microsoft Azure standard A4 VM instances with 8 cores, 14 GB of memory, Java 1.8, and Windows 2012 R2 Guest OS running over an AMD Opteron(TM) 2.1 GHz CPU

Description

- This evaluation is in Regards to Overall Performance particularly Solution 3, 4
- 4.27x performance-gain over desktop
- 6.53x performance-gain over cloud node
- Accuracy measures stay preserved through-out the process

Execution Scenario



Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

OAEI Large-scale Biomedical Track: task 1

Dataset Source

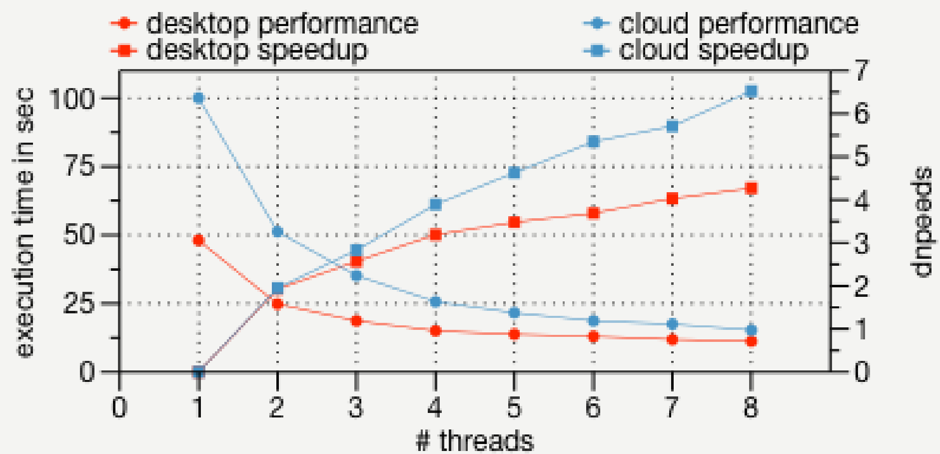
- OAEI 2013-2014 Standard Evaluation Dataset of Real-world Ontologies
- Matching Library: String-Annotation-ChildBased
- Magnitude: Medium to Large scale (MT > 71 Million)
- Candidate Ontologies:
  FMA = 3,696 concepts
  NCI = 6,488 concepts

Testbed

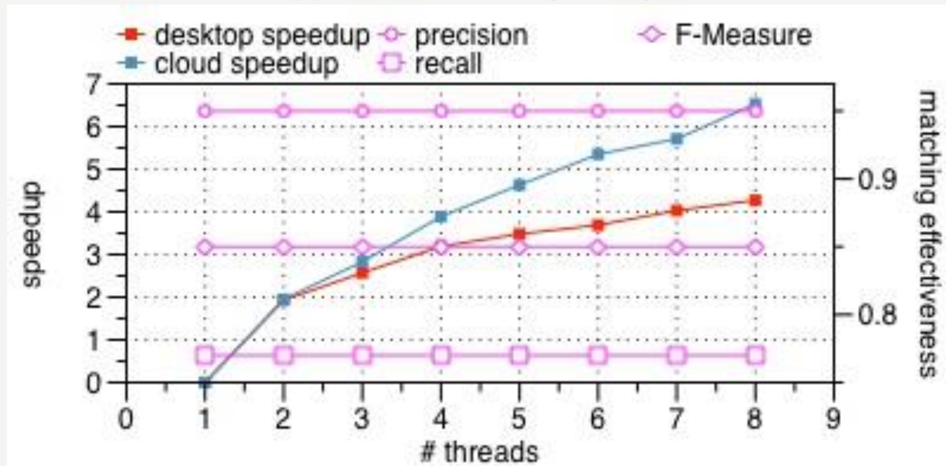
- Multicore Desktop:
  3.4 GHz Intel(R) Core i7(R) Hyper-Threaded (Intel(R) HT Technology) CPU (2 threads/ core) with 16 GB memory, Java 1.8 and Windows 7 64 bit OS
- Cloud:
  Microsoft Azure standard A4 VM instances with 8 cores, 14 GB of memory, Java 1.8, and Windows 2012 R2 Guest OS running over an AMD Opteron(TM) 2.1 GHz CPU

Description

- This evaluation is in Regards to Overall Performance particularly Solution 3, 4
- 4.27x performance-gain over desktop
- 6.53x performance-gain over cloud node
- Accuracy measures stay preserved through-out the process



(a) Performance-speedup



(b) Speedup-matching effectiveness

Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

Evaluation Results.(10/19)

OAEI Large-scale Biomedical Track: task 2

Dataset Source

- OAEI 2013-2014 Standard Evaluation Dataset of Real-world Ontologies
- Matching Library: String-Annotation-ChildBased
- Magnitude: Very Large scale (MT > 15 Billion)
- Candidate Ontologies:
  FMA = 78,989 concepts
  NCI = 66,724 concepts

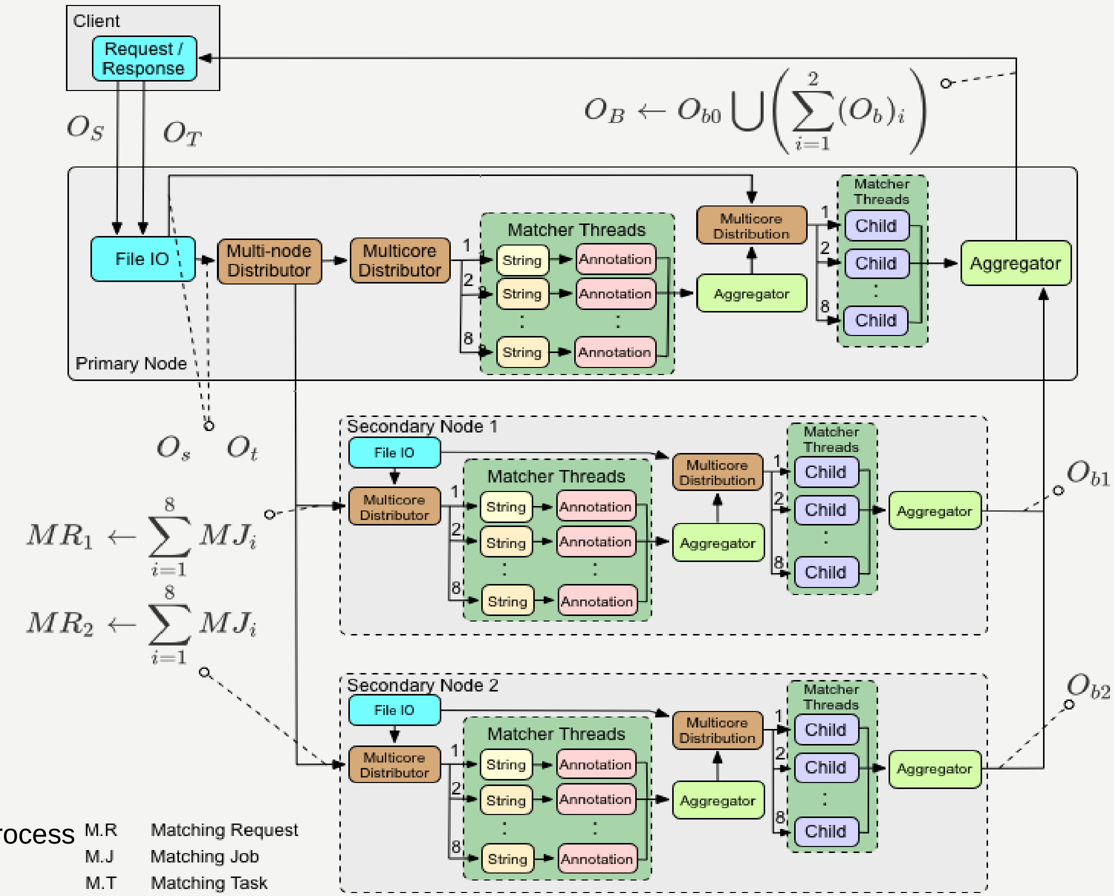
Testbed

- Multicore Desktop:
  3.4 GHz Intel(R) Core i7(R) Hyper-Threaded (Intel(R) HT Technology) CPU (2 threads/ core) with 16 GB memory, Java 1.8 and Windows 7 64 bit OS
- Cloud:
  Microsoft Azure standard A4 VM instances with 8 cores, 14 GB of memory, Java 1.8, and Windows 2012 R2 Guest OS running over an AMD Opteron(TM) 2.1 GHz CPU

Description

- This evaluation is in Regards to Overall Performance particularly Solution 3, 4
- 14.7x performance-gain over desktop
- 21.8x performance-gain over cloud node
- Accuracy measures stay preserved through-out the process

Execution Scenario



Published In

Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Evaluation Results.(11/19)

OAEI Large-scale Biomedical Track: task 2

Dataset Source

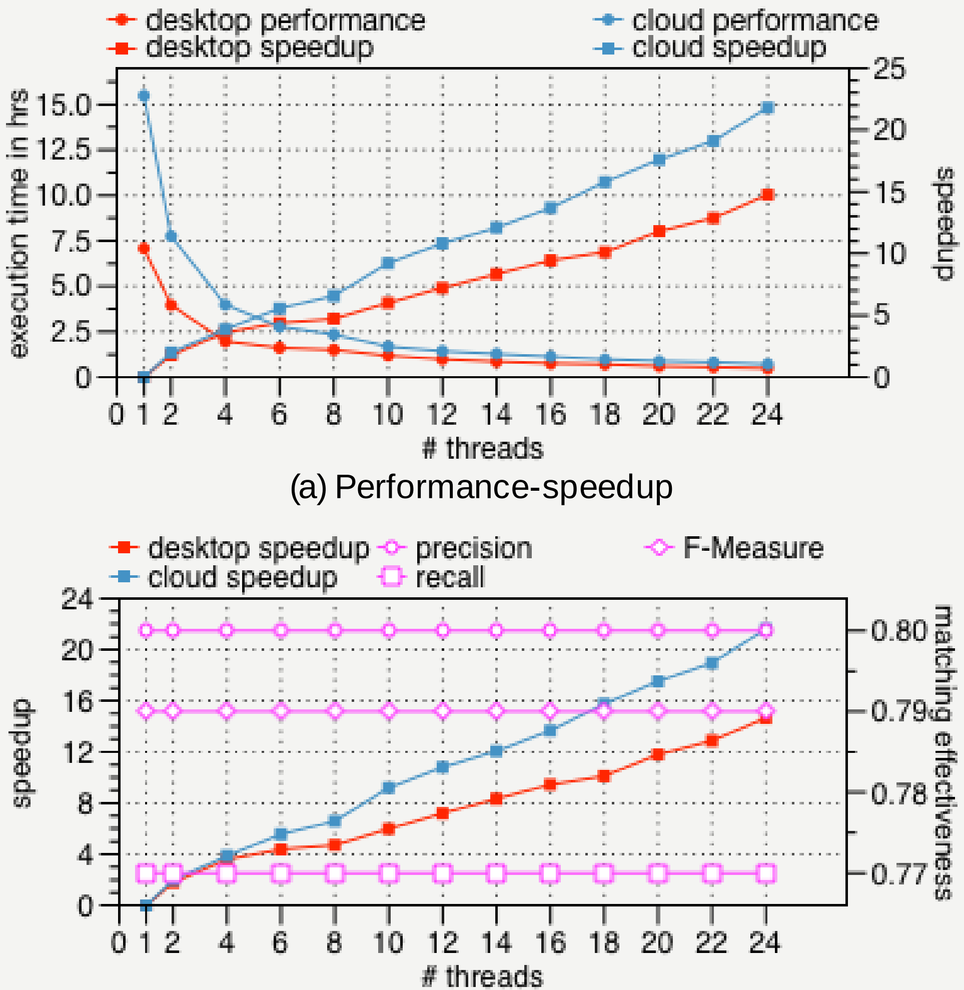
- OAEI 2013-2014 Standard Evaluation Dataset of Real-world Ontologies
- Matching Library: String-Annotation-ChildBased
- Magnitude: Very Large scale (MT > 15 Billion)
- Candidate Ontologies:
  FMA = 78,989 concepts
  NCI = 66,724 concepts

Testbed

- Multicore Desktop:
  3.4 GHz Intel(R) Core i7(R) Hyper-Threaded (Intel(R) HT Technology) CPU (2 threads/ core) with 16 GB memory, Java 1.8 and Windows 7 64 bit OS
- Cloud:
  Microsoft Azure standard A4 VM instances with 8 cores, 14 GB of memory, Java 1.8, and Windows 2012 R2 Guest OS running over an AMD Opteron(TM) 2.1 GHz CPU

Description

- This evaluation is in Regards to Overall Performance particularly Solution 3, 4
- 14.7x performance-gain over desktop
- 21.8x performance-gain over cloud node
- Accuracy measures stay preserved through-out the process



(a) Performance-speedup

(b) Speedup-matching effectiveness

Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Evaluation Results.(12/19)

OAEI Large-scale Biomedical Track: task 3

Dataset Source

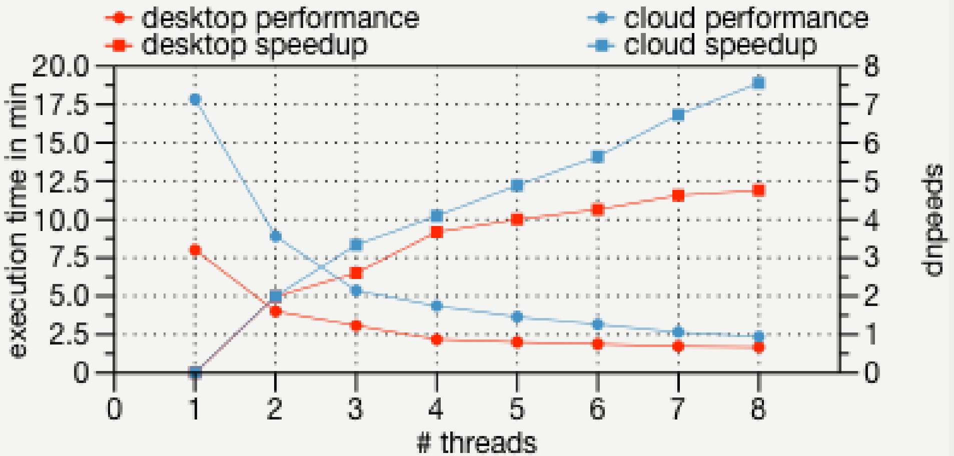
- OAEI 2013-2014 Standard Evaluation Dataset of Real-world Ontologies
- Matching Library: String-Annotation-ChildBased
- Magnitude: Large scale (MT > 400 Million)
- Candidate Ontologies:
  FMA = 10,157 concepts
  NCI = 13,412 concepts

Testbed

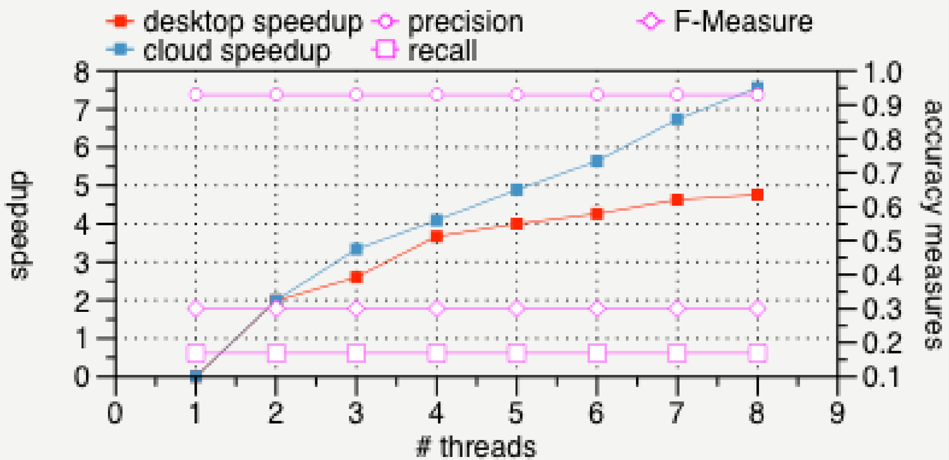
- Multicore Desktop:
  3.4 GHz Intel(R) Core i7(R) Hyper-Threaded (Intel(R) HT Technology) CPU (2 threads/ core) with 16 GB memory, Java 1.8 and Windows 7 64 bit OS
- Cloud:
  Microsoft Azure standard A4 VM instances with 8 cores, 14 GB of memory, Java 1.8, and Windows 2012 R2 Guest OS running over an AMD Opteron(TM) 2.1 GHz CPU

Description

- This evaluation is in Regards to Overall Performance particularly Solution 3, 4
- 4.76x performance-gain over desktop
- 7.56x performance-gain over cloud node
- Accuracy measures stay preserved through-out the process



(a) Performance-speedup



(b) Speedup-matching effectiveness

Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Evaluation Results.(13/19)

OAEI Large-scale Biomedical Track: task 4

Dataset Source

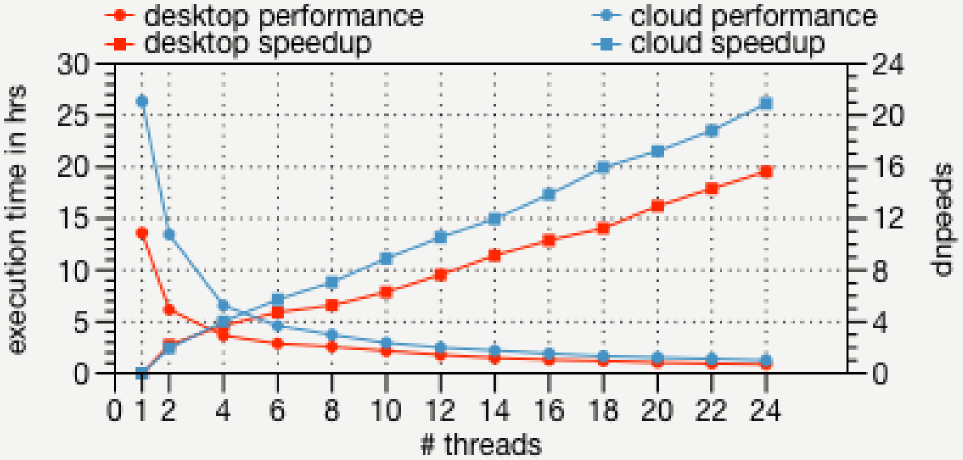
- OAEI 2013-2014 Standard Evaluation Dataset of Real-world Ontologies
- Matching Library: String-Annotation-ChildBased
- Magnitude: Very Large scale (MT > 29 Billion)
- Candidate Ontologies:
  FMA = 78,989 concepts
  SNOMED = 122,464 concepts

Testbed

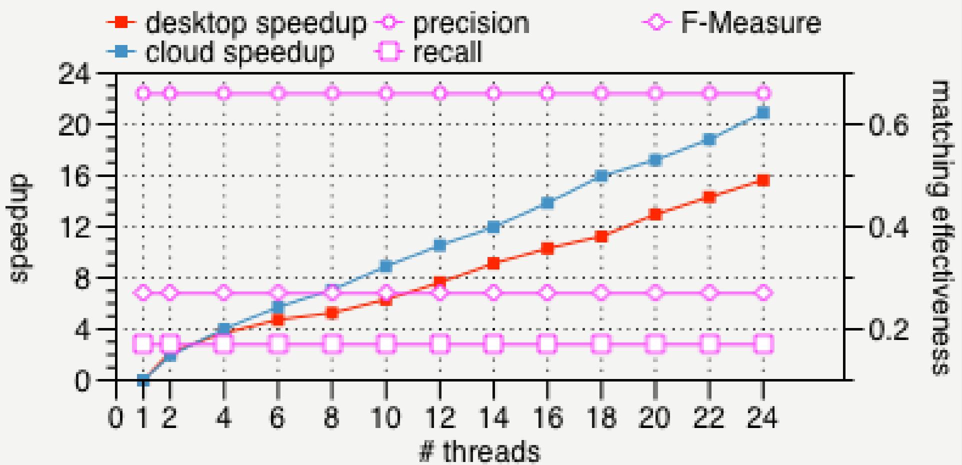
- Multicore Desktop:
  3.4 GHz Intel(R) Core i7(R) Hyper-Threaded (Intel(R) HT Technology) CPU (2 threads/ core) with 16 GB memory, Java 1.8 and Windows 7 64 bit OS
- Cloud:
  Microsoft Azure standard A4 VM instances with 8 cores, 14 GB of memory, Java 1.8, and Windows 2012 R2 Guest OS running over an AMD Opteron(TM) 2.1 GHz CPU

Description

- This evaluation is in Regards to Overall Performance particularly Solution 3, 4
- 15.64x performance-gain over desktop
- 21x performance-gain over cloud node
- Accuracy measures stay preserved through-out the process



(a) Performance-speedup



(b) Speedup-matching effectiveness

Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Evaluation Results.(14/19)

OAEI Large-scale Biomedical Track: task 5

Dataset Source

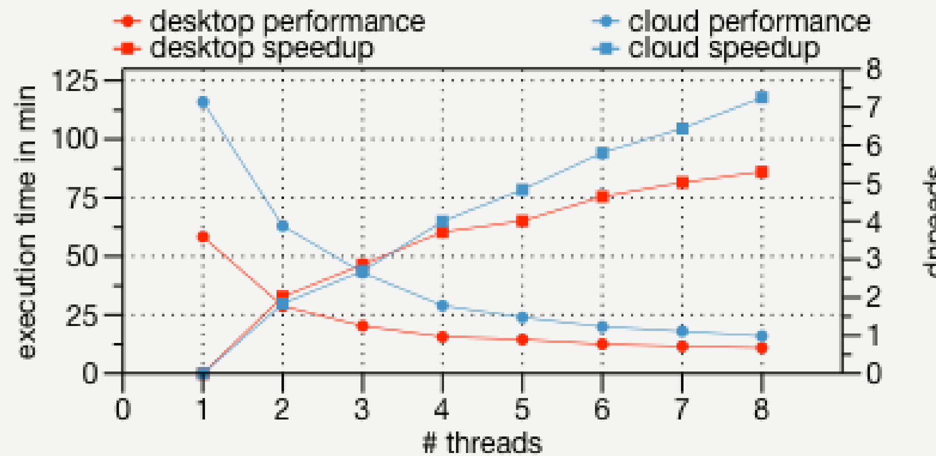
- OAEI 2013-2014 Standard Evaluation Dataset of Real-world Ontologies
- Matching Library: String-Annotation-ChildBased
- Magnitude: Large scale (MT > 3 Billion)
- Candidate Ontologies:
  FMA = 51,128 concepts
  NCI = 23,958 concepts

Testbed

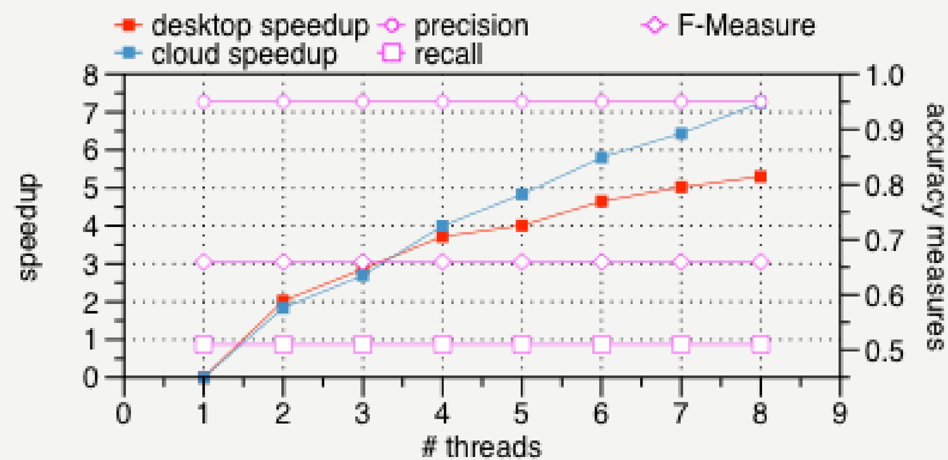
- Multicore Desktop:
  3.4 GHz Intel(R) Core i7(R) Hyper-Threaded (Intel(R) HT Technology) CPU (2 threads/ core) with 16 GB memory, Java 1.8 and Windows 7 64 bit OS
- Cloud:
  Microsoft Azure standard A4 VM instances with 8 cores, 14 GB of memory, Java 1.8, and Windows 2012 R2 Guest OS running over an AMD Opteron(TM) 2.1 GHz CPU

Description

- This evaluation is in Regards to Overall Performance particularly Solution 3, 4
- 5.31x performance-gain over desktop
- 7.25x performance-gain over cloud node
- Accuracy measures stay preserved through-out the process



(a) Performance-speedup



(b) Speedup-matching effectiveness

Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Evaluation Results.(15/19)

OAEI Large-scale Biomedical Track: task 6

Dataset Source

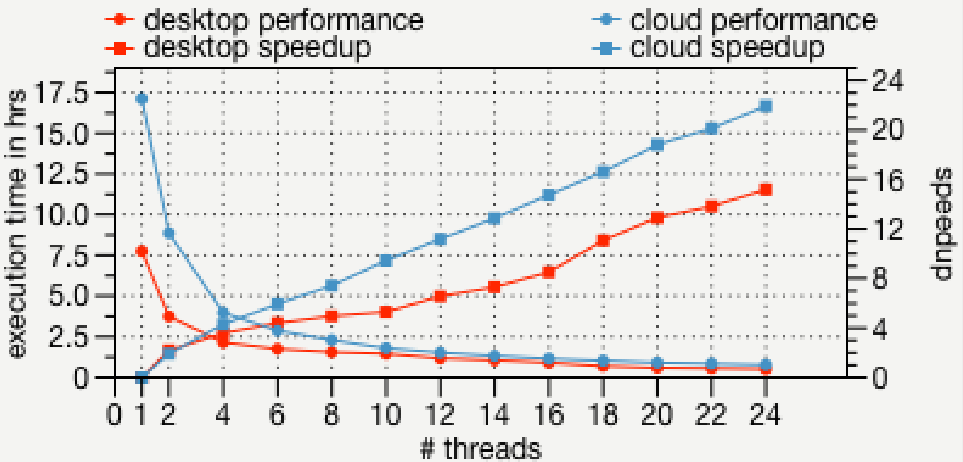
- OAEI 2013-2014 Standard Evaluation Dataset of Real-world Ontologies
- Matching Library: String-Annotation-ChildBased
- Magnitude: Very Large scale (MT > 24 Billion)
- Candidate Ontologies:
  NCI = 66,724 concepts
  SNOMED = 122,464 concepts

Testbed

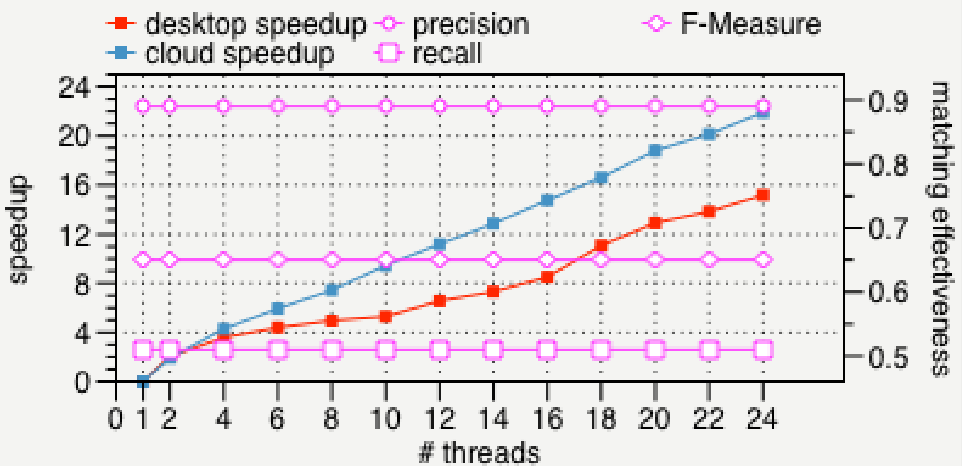
- Multicore Desktop:
  3.4 GHz Intel(R) Core i7(R) Hyper-Threaded (Intel(R) HT Technology) CPU (2 threads/ core) with 16 GB memory, Java 1.8 and Windows 7 64 bit OS
- Cloud:
  Microsoft Azure standard A4 VM instances with 8 cores, 14 GB of memory, Java 1.8, and Windows 2012 R2 Guest OS running over an AMD Opteron(TM) 2.1 GHz CPU

Description

- This evaluation is in Regards to Overall Performance particularly Solution 3, 4
- 15.19x performance-gain over desktop
- 22x performance-gain over cloud node
- Accuracy measures stay preserved through-out the process



(a) Performance-speedup



(b) Speedup-matching effectiveness

Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Evaluation Results.(16/19)

OAEI Conference Track

Dataset Source

- OAEI 2013-2014 Standard Evaluation Dataset of Real-world Ontologies
- Matching Library: String-Annotation-ChildBased-Synonym
- Magnitude: Small scale

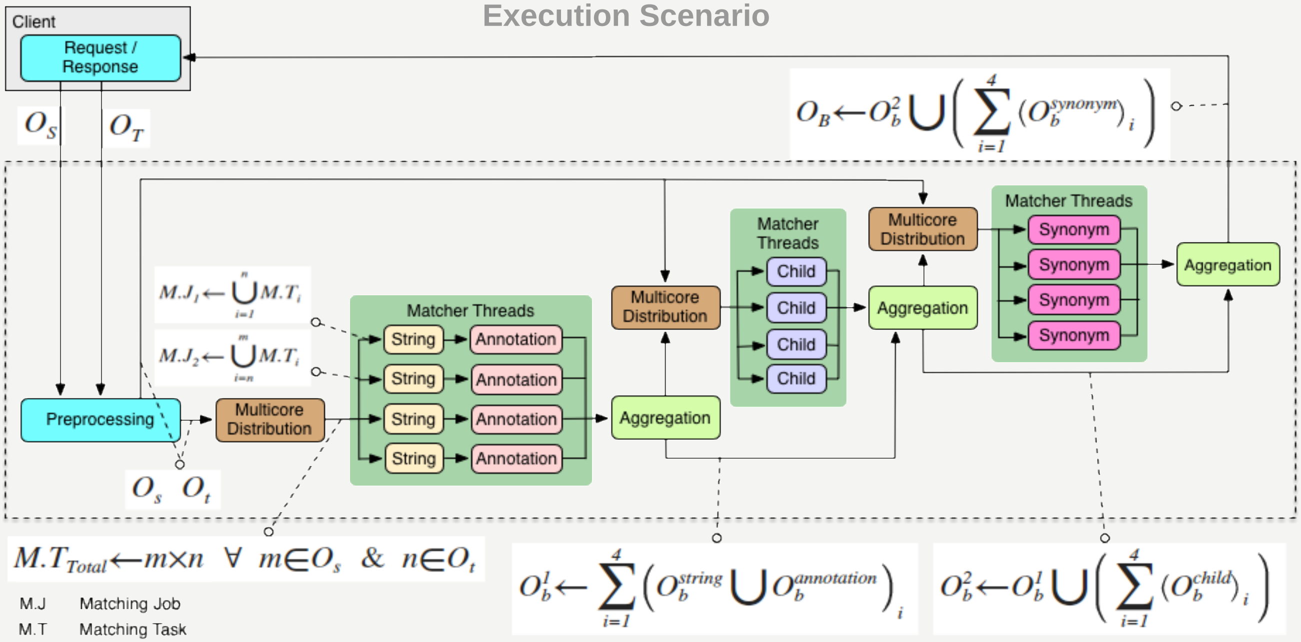
Testbed

- Cloud:
  Microsoft Azure standard A2 VM instances with 2 cores, 1.5 GB of memory, Java 1.8, and Windows 2012 R2 Guest OS running over an AMD Opteron(TM) 2.1 GHz CPU

Description

- This evaluation is in Regards to Overall Performance particularly Solution 3, 4
- 1.2x performance-gain over cloud node
- Accuracy measures stay preserved through-out the process

Execution Scenario

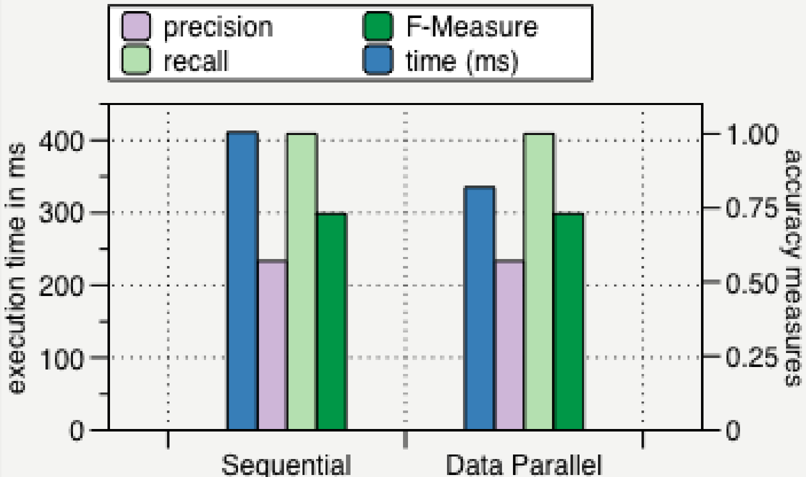


Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

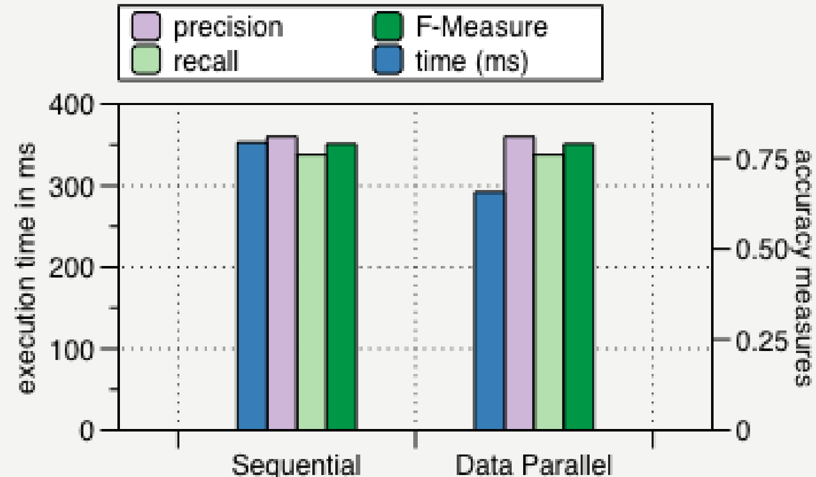
mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Evaluation Results.(17/19)

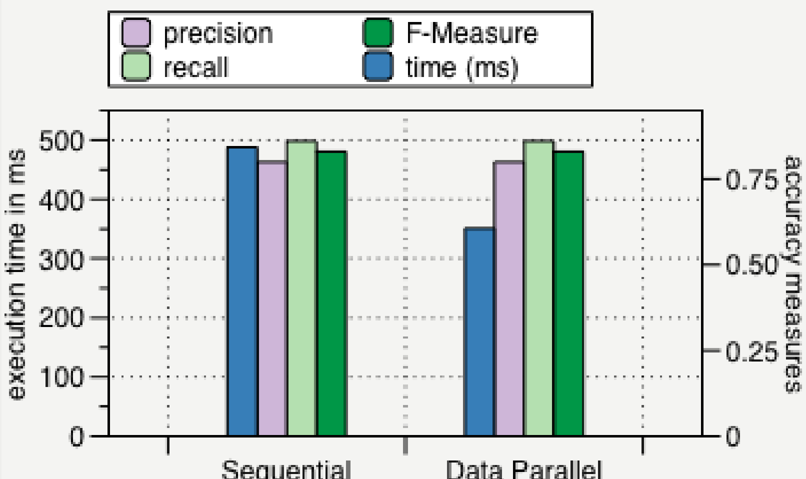
OAEI Conference Track



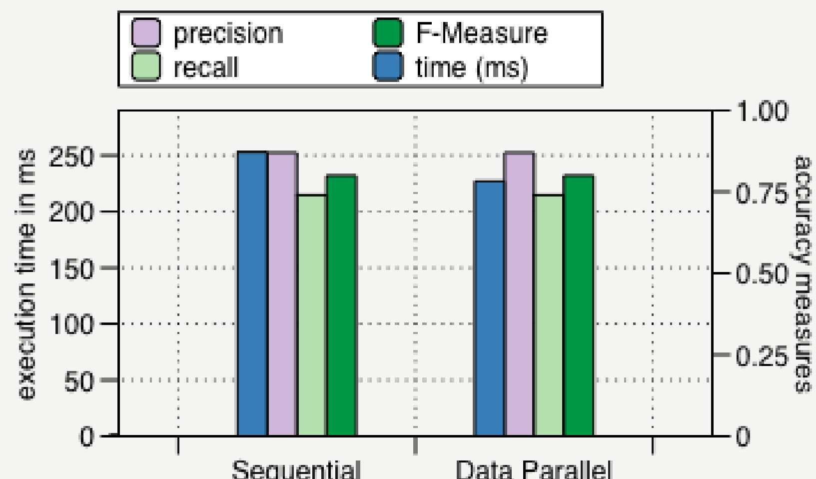
(a) cmt-iasted



(b) conference-edas



(c) conference-iasted

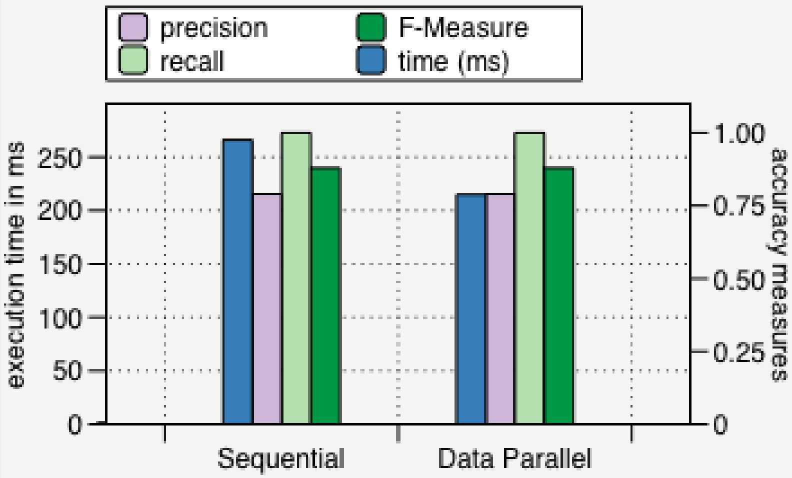


(d) confOf-edas

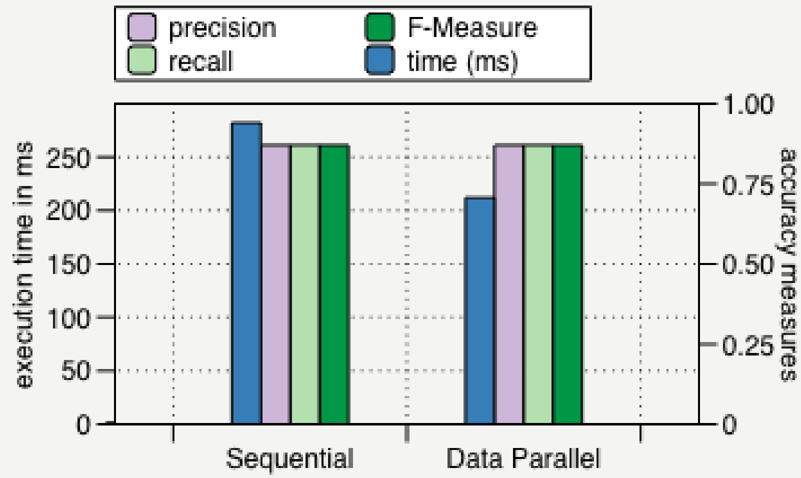
Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

Evaluation Results.(18/19)

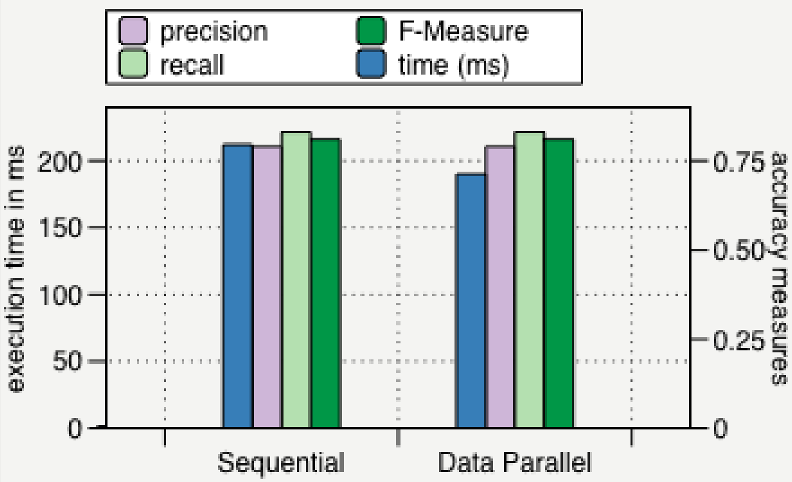
OAEI Conference Track



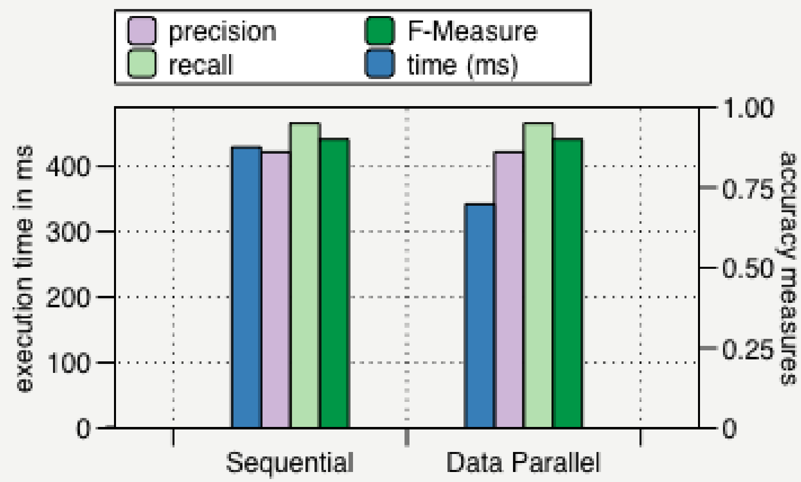
(a) ekaw-sigkdd



(b) iasted-sigkdd



(c) edas-ekaw

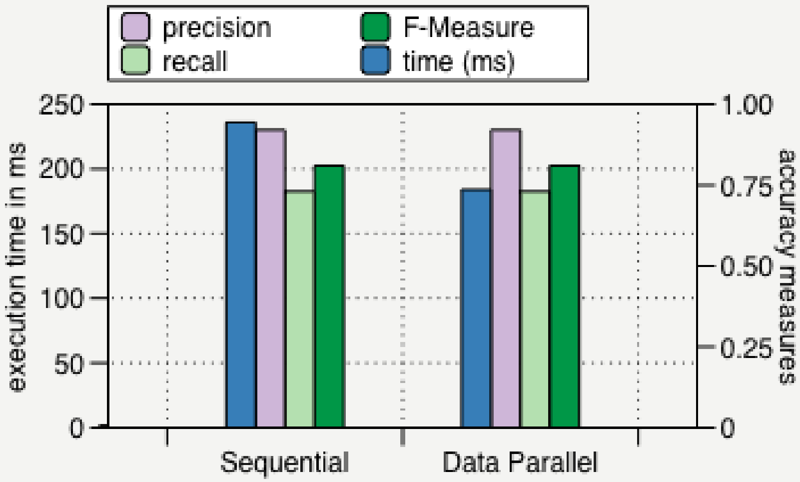


(d) edas-iasted

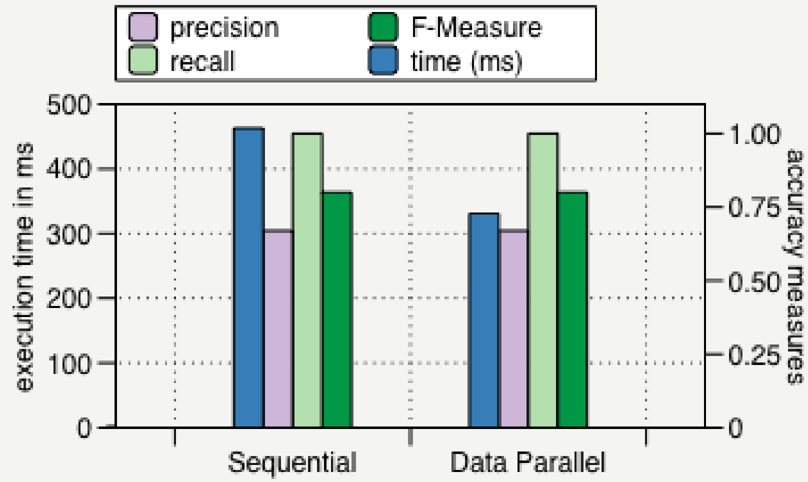
Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

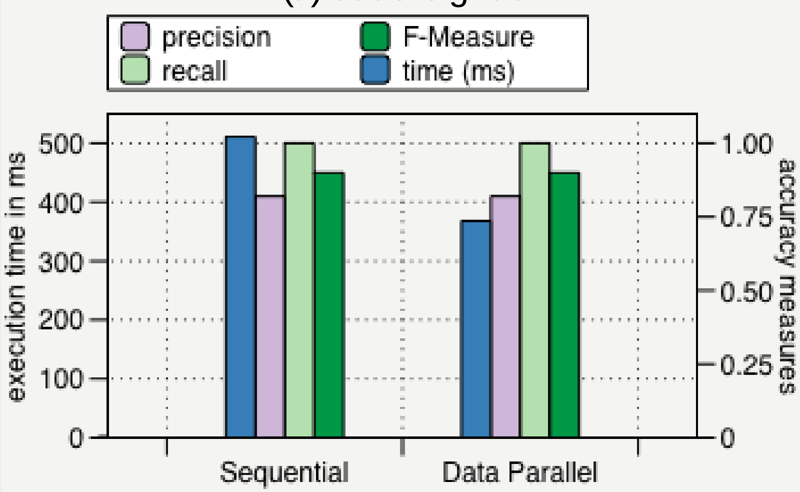
OAEI Conference Track



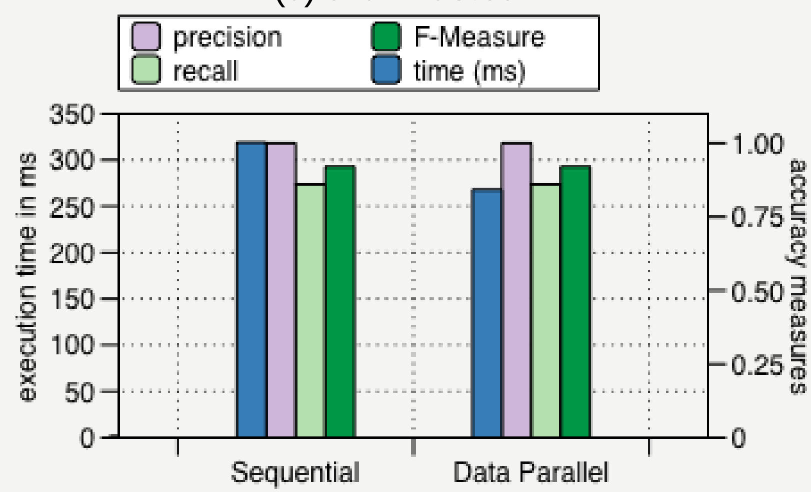
(a) edas-sigkdd



(b) ekaw-iasted



(c) confOf-iasted



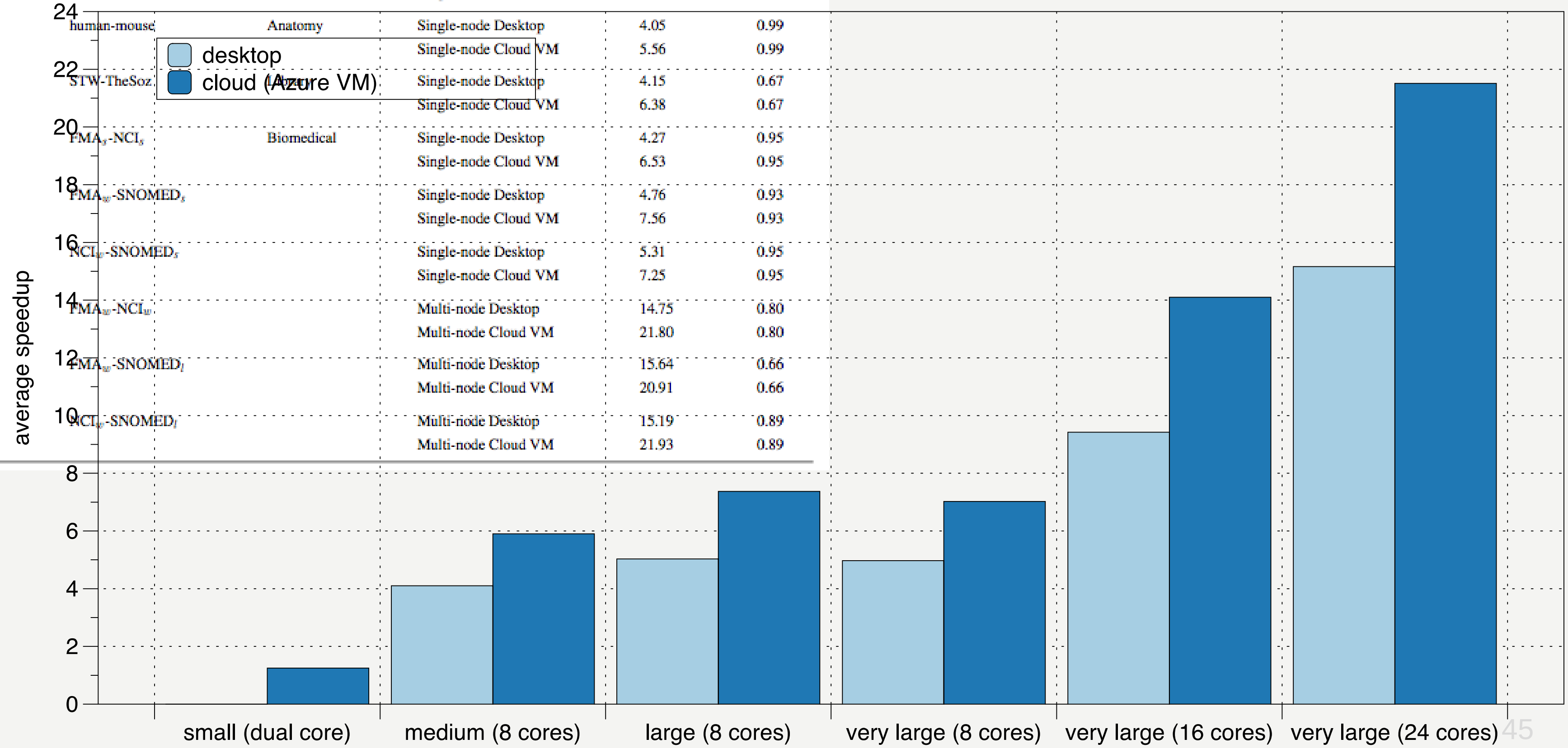
(d) confOf-sigkdd

Published In Muhammad Bilal Amin, Wajahat Ali Khan, Sungyoung Lee, Byeong Ho Kang, Performance-based ontology matching, A data parallel approach for an effectiveness-independent performance-gain in ontology matching, Applied Intelligence (SCI, IF:1.85) (2015)

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Result Summary.

	Matching problem	Domain	Platform	Speed up	Precision
Small	cmt-iasted	Conference	Single-node Cloud VM	1.22	0.57
	conference-edas		Single-node Cloud VM	1.25	0.81
	conference-iasted		Single-node Cloud VM	1.39	0.80
	confof-edas		Single-node Cloud VM	1.11	0.87
	confof-iasted		Single-node Cloud VM	1.38	0.82
	confof-sigkdd		Single-node Cloud VM	1.19	1.00
	edas-sigkdd		Single-node Cloud VM	1.28	0.92
	ekaw-iasted		Single-node Cloud VM	1.39	0.67
	ekaw-sigkdd		Single-node Cloud VM	1.23	0.79
	iasted-sigkdd		Single-node Cloud VM	1.33	0.87
	edas-ekaw		Single-node Cloud VM	1.11	0.79
	edas-iasted		Single-node Cloud VM	1.25	0.86
Medium	human-mouse	Anatomy	Single-node Desktop	4.05	0.99
			Single-node Cloud VM	5.56	0.99
	STW-TheSoz	[illegible]	Single-node Desktop	4.15	0.67
			Single-node Cloud VM	6.38	0.67
	FMA_s-NCI_s	Biomedical	Single-node Desktop	4.27	0.95
			Single-node Cloud VM	6.53	0.95
Large	FMA_w-$SNOMED_s$		Single-node Desktop	4.76	0.93
			Single-node Cloud VM	7.56	0.93
	NCI_w-$SNOMED_s$		Single-node Desktop	5.31	0.95
			Single-node Cloud VM	7.25	0.95
Very Large	FMA_w-NCI_w		Multi-node Desktop	14.75	0.80
			Multi-node Cloud VM	21.80	0.80
	FMA_w-$SNOMED_l$		Multi-node Desktop	15.64	0.66
			Multi-node Cloud VM	20.91	0.66
	NCI_w-$SNOMED_l$		Multi-node Desktop	15.19	0.89
			Multi-node Cloud VM	21.93	0.89



mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

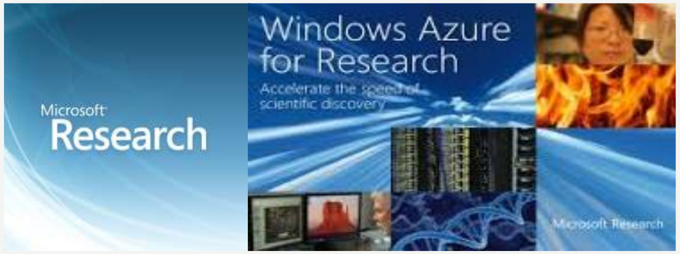
Contribution and Uniqueness.

- Ontology Loading and Memory Stress
  - An Ontology model based on scalable data structures, provides thread-safety and supports parallelism. (2.5 times better performance)
  - Ontology subset creation , substantially reduces the memory load within the 2Gb of Heap (8 times smaller than Jena and OWLApi) as System only loads the required ontology resources
- Parallel and distributed Ontology Matching
  - Provides a 3 layer abstraction over ontology matching process, matching from grainer to finer level with the help of thread level parallelism (40% better Reduction Score)
  - Exploits the multi-core desktops and cloud platforms for its benefit in performance gain (From 4 to 22 times performance-gain depending upon the size of the ontologies and execution environment)
- Aligned Execution for Zero Redundant matching operations (Eager Matching Space Reduction)
- A non-monolithically runtime, platform is sharable by services for clients and by platform for semantic web experts
  - 9 Matching Libraries ported for execution without any change in the ontology matching algorithms
- Accuracy Preservance through-out the performance-gain (Effectiveness-Independent Resolution), No loss of accuracy with the performance-gain in the matching process

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Achievements.

- Accepted Proposals
  - Microsoft Research Asia 2013-2014, Beijing, China
    - Semantic Heterogeneity Resolution by Implementing Parallelism over Multicore Cloud Platform, Muhammad Bilal Amin and Sungyoung Lee
  - Azure4Research Award 2014, Microsoft Research, Redmond, USA
    - Enabling Data Parallelism for large-scale Biomedical Ontology Matching over Multicore Cloud Instances, Muhammad Bilal Amin and Sungyoung Lee [6]



- Ontology Alignment Evaluation Initiative (OAEI 2013-2014)
  - Proposed Methodology as SPHeRe's Runtime for Ontology Matching
  - Evaluation of all 6 task over 9 real-world ontology matching problems of vario complexities
  - Only 23 from 54 participating systems completed all the tasks in-time
  - Ranked among top 12 Ontology Matching Systems of 2013-2014 [7]

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Conclusion and Future Work

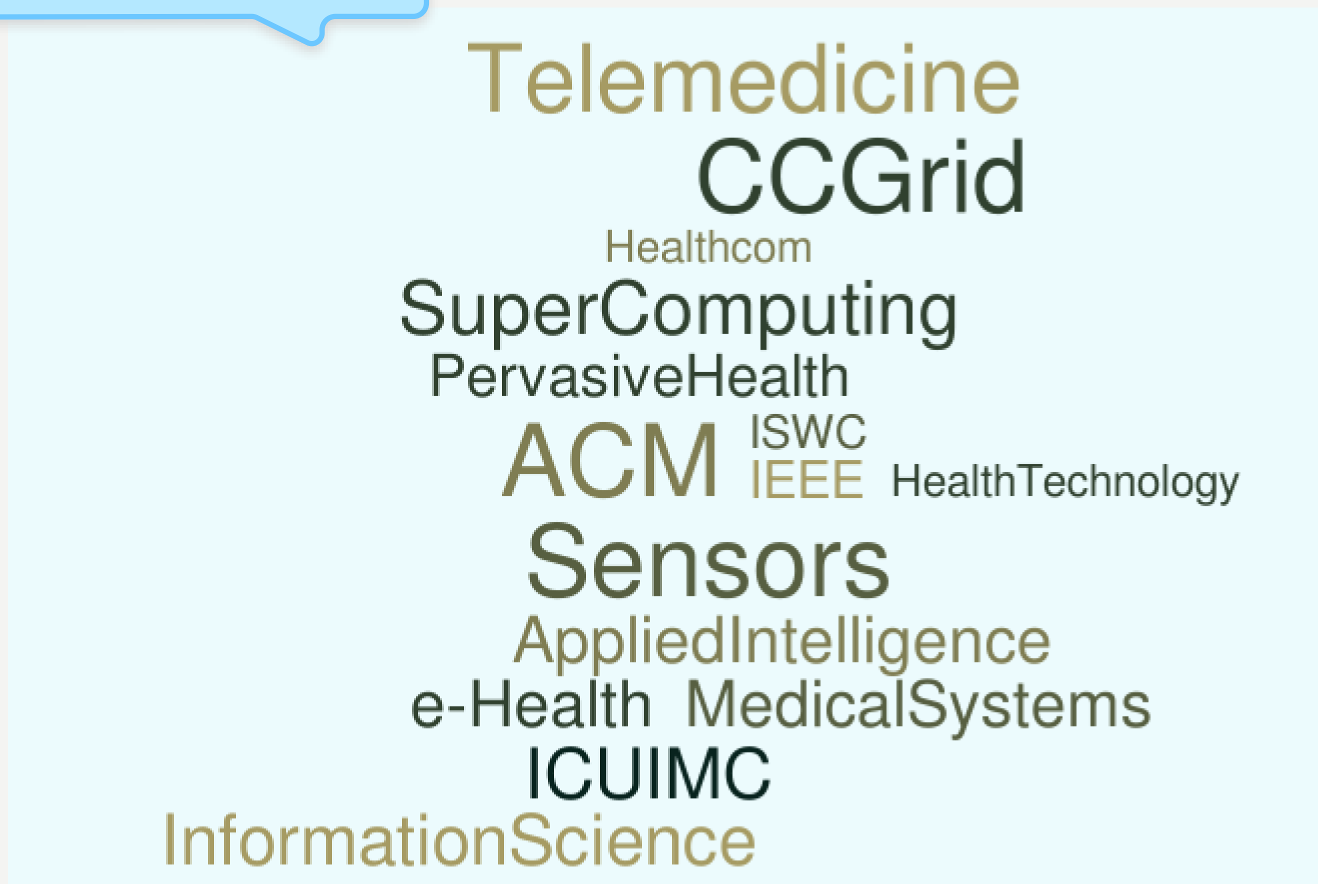
- This thesis explicitly discusses the performance issues and bottlenecks of the ontology matching problem
- Present methodology provides end-to-end resolution by catering performance from ontology loading, memory management, matching and delivery
- Results have shown a substantial gain in performance with Accuracy preservance by adopting the presented proposal
  - 2.5x faster Ontology Loading
  - 8x smaller Memory Footprint with No Heap Issues
  - 40% better reduction score due to abstraction based parallelism
  - 4 – 21x overall performance-gain depending upon the size of the matching problem and provided environment
- Future Work & Research
  - Presents and opportunity for the semantic-web and cloud community to use proposed implementation as a platform for heterogeneity resolution and matching algorithm evaluation
  - Cloud-based High Performance Ontology Matching and Algorithm evaluation portal

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Publications.

- Patents (6)
  - Domestic : 5
  - International : 1
- Journals (7)
  - SCI :
    - First Author (2)
    - Co-author (1)
  - SCI(E) :
    - Co-authors (3)
  - Non-SCI :
    - Co-authors (1)
- Conferences (21)
  - International :
    - First Author (4)
    - Co-author (12)
  - Domestic :
    - First Author (5)

34 Publications
1 Major Revision
1 Under review



mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Selected References.

1. Pavel Shvaiko and Jerome Euzenat. Ontology Matching: State of the Art and Future Challenges. (2013). IEEE Transaction on Knowledge and Data Engineering, 25(1), 158–176. doi:10.1109/TKDE.2011.25
2. Agrawal, R., Ailamaki, A., Bernstein, P. A., Brewer, E. A.,Carey, M. J., Chaudhuri, S., et al. (2009). The Claremont report on database research. Communications of the ACM, 52(6), 56–65. doi:10.1145/1516046.1516062
3. Ontology Matching - Springer. (2007). doi:10.1007/978-3-642-38721-0
4. Anika Groß, Michael Hartung, Toralf Kirsten, Erhard Rahm . (2010), On Matching Large Life Science Ontologies in Parallel, 1–16. Proceedings of the 7th International Conference on Data Integration in the Life Sciences
5. M. Bilal Amin, Rabia Batool, Wajahat Ali Khan, Sungyoung Lee, Eui-Nam Huh. (2014). SPHeRe: A Performance Initiative Towards Ontology Matching by Implementing Parallelism over Cloud Platform, Journal of Supercomputing, 68(1), 274–301. doi:10.1007/s11227-013-1037-1
6. M. Bilal Amin, Wajahat Ali Khan, Byeong Ho Kang, Sungyoung Lee (2015). Performance-based Ontology Matching, A data-parallel approach for an effectiveness-independent performance-gain in ontology matching. Applied Intelligence DOI 10.1007/s10489-015-0648-z
7. Latest recipients of Windows Azure for Research Awards announced, http://blogs.msdn.com/b/msr_er/archive/2014/01/16/latest-recipients-of-windows-azure-for-research-awards-announced.aspx
8. Bernardo et. al. Results of the ontology alignment evaluation initiative 2013. (2013). ISWC workshop on Ontology Matching (OM-2013)
9. Stoilos, G., Stamou, G. & Kollias, S. (2005). A String Metric For Ontology Alignment. In Y. Gil, E. Motta, V. R. Benjamins & M. A. Musen (eds.), Proceedings of the 4rd International Semantic Web Conference (ISWC) (p./pp. 624--637), November, Berlin, Heidelberg: Springer.
10. Cruz, I. F., Antonelli, F. P. & Stroe, C. (2009). AgreementMaker: efficient matching for large real-world schemas and ontologies. Proceedings of the VLDB Endowment, 2, 1586–1589.
11. David, J., Guillet, F. & Briand, H. (2006). Matching directories and OWL ontologies with AROMA. In P. S. Yu, V. J. Tsotras, E. A. Fox & B. L. 0001 (eds.), CIKM (p./pp. 830-831), : ACM. ISBN: 1-59593-433-2
12. Jean-Mary, Y. R., Shironoshita, E. P. & Kabuka, M. R. (2009). Ontology Matching with Semantic Verification. Web Semantics, 7, 235--251.
13. Combinatorial optimization for data integration (CODI). http://code.google.com/p/codi-matcher/
14. Tran QV, Ichise R, Ho BQ (2011) Cluster-based similarity aggregation for ontology match- ing. In: OM, CEUR workshop proceedings, vol. 814. CEUR-WS.org. http://dblp.uni-trier.de/ db/conf/semweb/om2011.html#TranIH11
15. Hu, W. & Qu, Y. (2008). Falcon-AO: A practical ontology matching system. Web Semantics, 6, 237–239.
16. Generic ontology matching and mapping management (GOMMA). http://dbs.uni-leipzig.de/GOMMA
17. Wang P, Xu B (2009) Lily: Ontology alignment results for oaei 2009
18. LogMap: logic-based methods for ontology mapping. http://www.cs.ox.ac.uk/isg/projects/LogMap/
19. Cheatham M (2011) MAPSSS results for oaei 2011. In: OM, CEUR workshop proceedings, vol 814. CEUR-WS.org. http://dblp.uni-trier.de/db/conf/semweb/om2011.html#Cheatham11
20. Schadd FC, Roos N (2011) Maasmatch results for oaei 2011. In: OM, CEUR workshop proceedings, vol. 814. CEUR-WS.org. http://dblp.uni-trier.de/db/conf/semweb/om2011.html#SchaddR11
21. Lambrix P, Tan H (2006) SAMBO-A system for aligning and merging biomedical ontologies. Web Semant 4:196–206
22. Ba M, Diallo G (2011) Large-scale biomedical ontology matching with ServOMap. IRBM 34:56–59

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Thank you

Appendix

Ontology Loading Algorithms.(1/2)

Algorithm 1 Method owlLoad

Require: $O_s \neq NULL$ and $O_t \neq NULL$
  $Hash_s \leftarrow$ Utility.calculateHash(O_s)
  $Hash_t \leftarrow$ Utility.calculateHash(O_t)
  ontologyCache $\leftarrow$ OntologyCache.getInstance()
  parser $\leftarrow$ Parser.createInstance()
  if $Hash_s$, $Hash_t$ *!in ontologyCache* **then**
    parser.parse(O_s, O_t)
    parser.serialize(O_s, O_t)
  else
    if $Hash_t$ *!in ontologyCache* **and** $Hash_s$ *in ontologyCache* **then**
      parser.parse(O_t)
      parser.serialize(O_t)
    else if $Hash_s$ *!in ontologyCache* **and** $Hash_t$ *in ontologyCache* **then**
      parser.parse(O_s)
      parser.serialize(O_s)
    end if
  end if
  deserialize($Hash_s$, $Hash_t$)
  return

Algorithm 2 Method nameParser

Require: $O_x \neq NULL, x \in \{s, t\}$
  thing $\leftarrow$ Thing.createInstance(url)
  while O_x *has classes* **do**
    concept $\leftarrow$ OClass.createInstance(*currentClassName*)
    thing.addConcept(&*concept*)
  end while
  return thing

Algorithm 3 Method labelParser

Require: $O_x \neq NULL, x \in \{s, t\}$
  thing $\leftarrow$ Thing.createInstance(url)
  while O_x *has classes* **do**
    concept $\leftarrow$ OClass.createInstance(*currentClassName*)
    while *currentClass has labels* **do**
      label $\leftarrow$ Annotation.createLabel(*labelName*)
      concept.addAnnotation(&*label*)
    end while
    thing.addConcept(&*concept*)
  end while
  return thing

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Ontology Loading Algorithms.(2/2)

Algorithm 4 Method propertyParser

Require: $O_x \neq NULL, x \in \{s, t\}$

```
thing ← Thing.createInstance(url)
while O_x has classes do
    concept ← OClass.createInstance(currentClassName)
    while currentClass has properties do
        property ← OProperty.createInstance(propertyName)
        concept.addProperty(&property)
    end while
    thing.addConcept(&concept)
end while
return thing
```

Algorithm 5 Method hierarchyParser

Require: $O_x \neq NULL, x \in \{s, t\}$

```
thing ← Thing.createInstance(url)
while O_x has classes do
    concept ← OClass.createInstance(currentClassName)
    while currentClass has parents do
        if !thing.exists(parent) then
            parent ← OClass.createInstance(parentName)
            thing.addConcept(&parent)
        else
            parent ← thing.getConcept(parentName)
        end if
        concept.addConcept(&parent)
    end while
    thing.addConcept(&concept)
end while
return thing
```


mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Barrier Read Algorithm

Algorithm 1 Method owlLoad

Require: $O_s \neq NULL$ and $O_t \neq NULL$

$Hash_s \leftarrow$ Utility.calculateHash(O_s)
$Hash_t \leftarrow$ Utility.calculateHash(O_t)
$ontologyCache \leftarrow$ OntologyCache.getInstance()
$parser \leftarrow$ Parser.createInstance()
if $Hash_s$, $Hash_t$ $!in\ ontologyCache$ **then**
  parser.parse(O_s, O_t)
  parser.serialize(O_s, O_t)
else
  if $Hash_t$ $!in\ ontologyCache$ **and** $Hash_s$ $in\ ontologyCache$ **then**
    parser.parse(O_t)
    parser.serialize(O_t)
  else if $Hash_s$ $!in\ ontologyCache$ **and** $Hash_t$ $in\ ontologyCache$ **then**
    parser.parse(O_s)
    parser.serialize(O_s)
  end if
end if
deserialize($Hash_s$, $Hash_t$)
return

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Distribution Algorithms.

Algorithm 2 Distributor algorithm

Require: $nodes > 0$
  if $nodes$=1 **then**
    MulticoreDistributor(O_S,O_T)
  else
    Multi-nodeDistributor(O_S,O_T)
  end if

Algorithm 3 Multicore distributor algorithm

Require: $nodes > 0$
  $cores$ ←Runtime.getNumberOfCores()
  if $nodes$=1 **then**
    $start$=0
    $bigOnt \leftarrow (size_S \geq size_T)?O_S : O_T$
    $smallOnt \leftarrow (size_S < size_T)?O_S : O_T$
    $Partition_{slab} = \lceil bigOnt.size/cores \rceil$
    SPAWN MATCHER THREADS:
    for do$i = 1$ **to** $cores$ **do**
      $end = start + Partition_{slab}$
      if $end \leq bigOnt.size$ **then**
        $end = bigOnt.size$
      end if
      MatchingJob.create($MatchingTasks[start, end), big, small, matcher$)
      thread.run($matchingJob$)
      $start = end$
    end for
  else
    RECEIVE MATCHING REQUEST:
    controlMessage.receive($matchingRequest$)
    $Partition_{slab} = (end - start)/cores$
    GOTO SPAWN MATCHER THREADS
  end if

Algorithm 4 Multi-node distributor algorithm

Require: $nodes > 1$
  $nodes$ ←initDaemon.getNoOfNodes()
  $participatingCores = \sum node.\#cores$
  $start$=0
  end=0
  $bigOnt \leftarrow (size_S \geq size_T)?O_S : O_T$
  $smallOnt \leftarrow (size_S < size_T)?O_S : O_T$
  $Distribution_{slab} = \lceil bigOnt.size/participatingCores \rceil$
  for $node \leftarrow nodes$ **do**
    $end = start + Distribution_{slab} \times node.\#cores$
    if $end \leq bigOnt.size$ **then**
      $end = bigOnt.size$
    end if
    MatchingRequest.create($[start, end), big, small, matcher$)
    if $node.isLocal$ **then**
      local.MulticoreDistributor($matchingRequest$)
    else
      controlMessage.send($matchingRequest$)
    end if
    $start = end$
  end for

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Class Diagrams and Conceptual Models.(1/2)

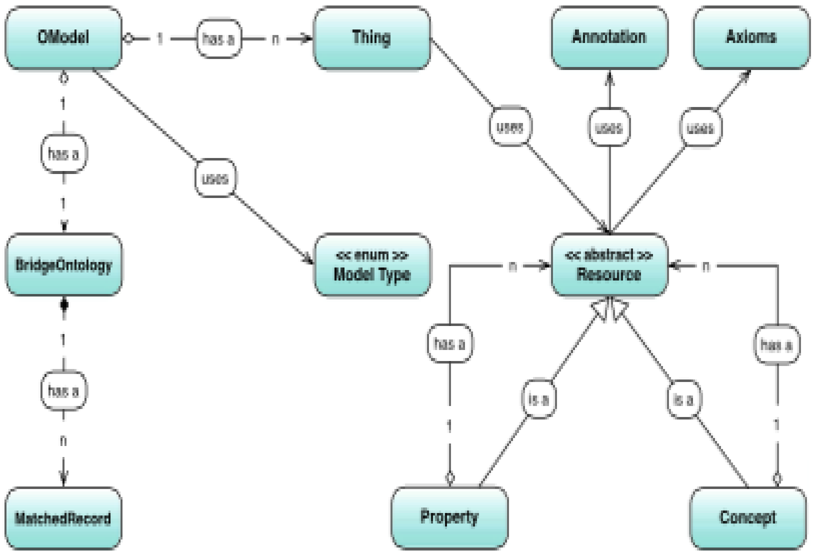


Fig. 2 Ontology Model class diagram

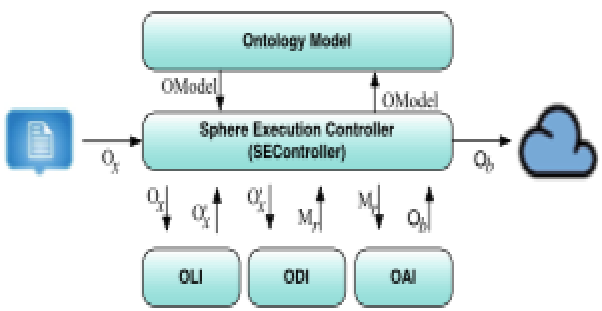


Fig. 3 SPHeRe's execution controller

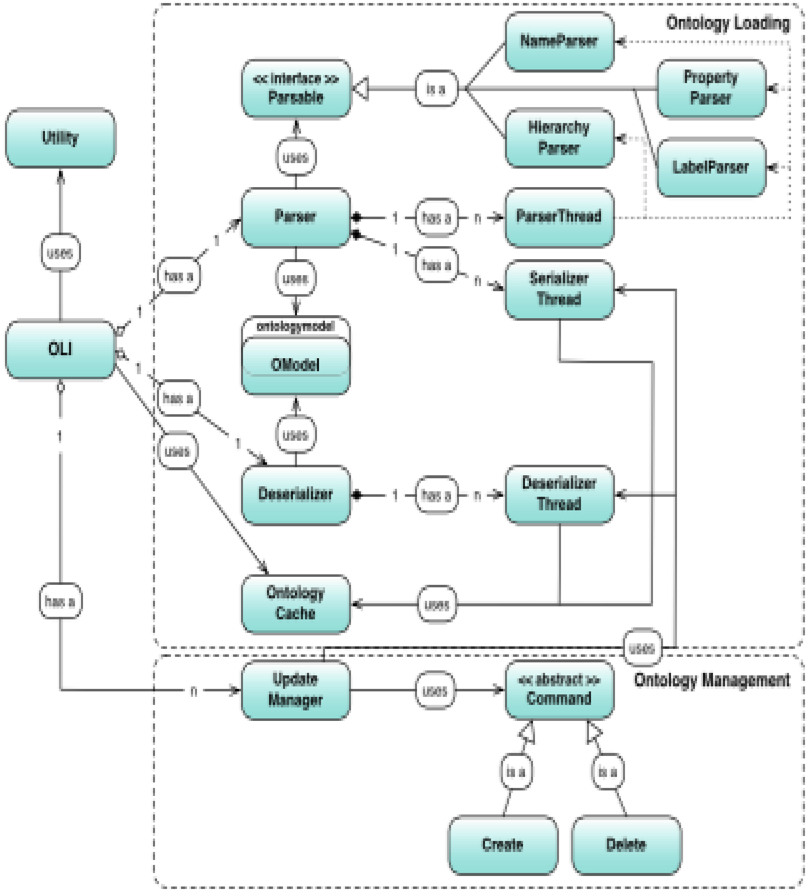


Fig. 4 Class diagram for loading and management component

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Class Diagrams and Conceptual Models.(2/2)

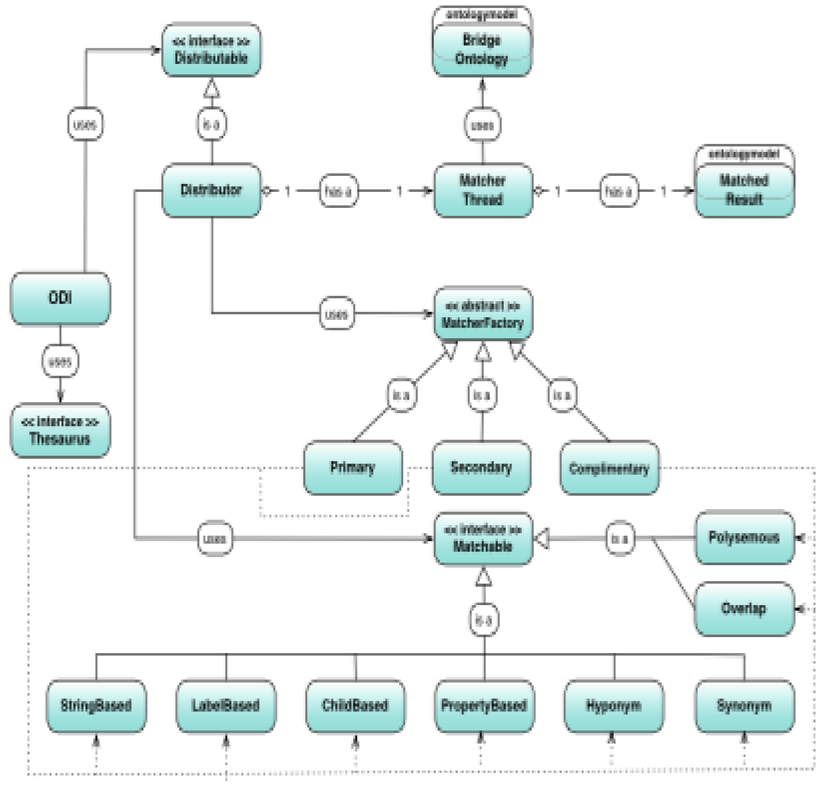


Fig. 5 Class diagram for distribution and matching component

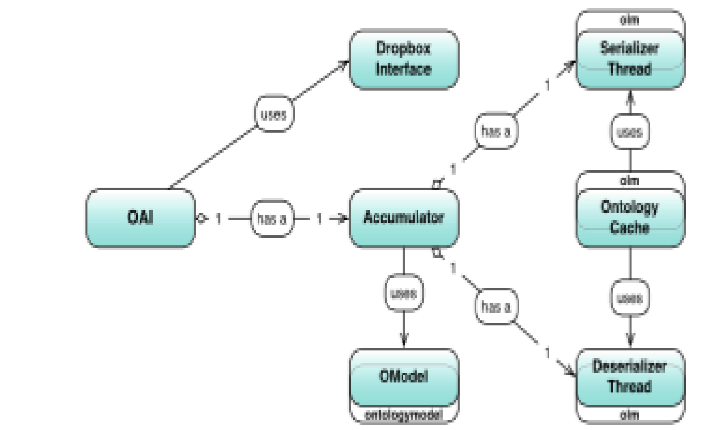


Fig. 6 Class diagram for accumulation and delivery component

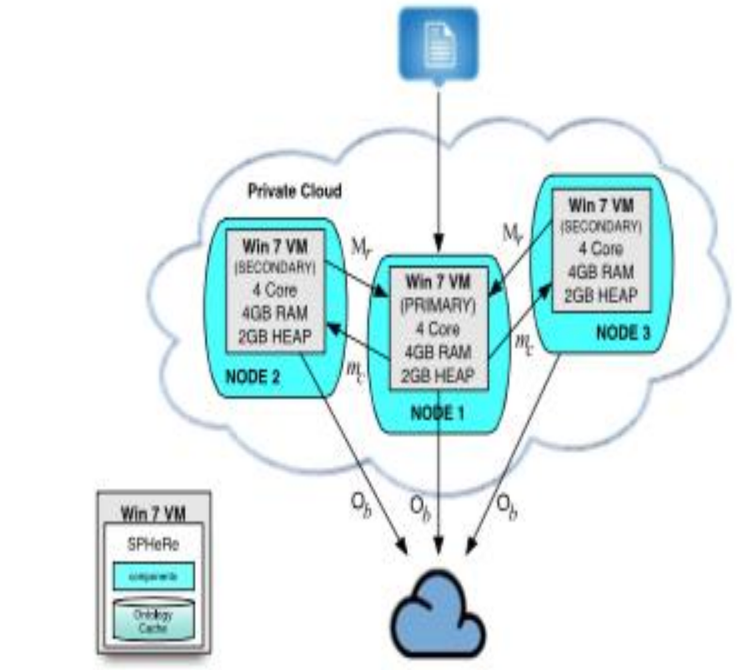


Fig. 7 SPHeRe's deployment setup

Communication and Sequence Diagrams.(1/2)

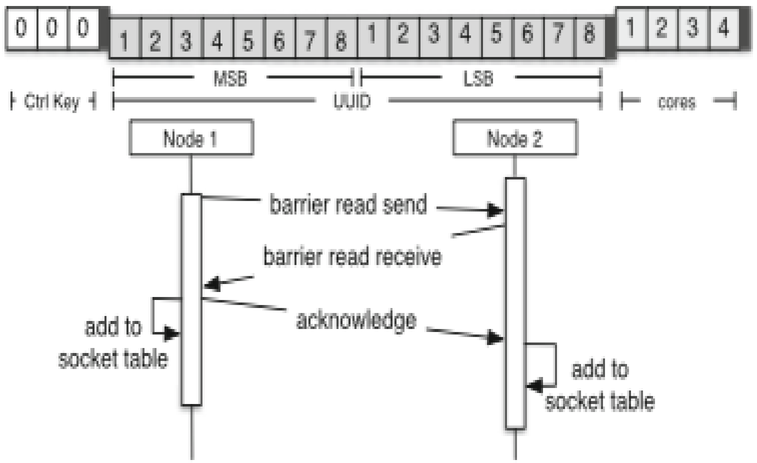


Fig. 5 Barrier read sequence diagram

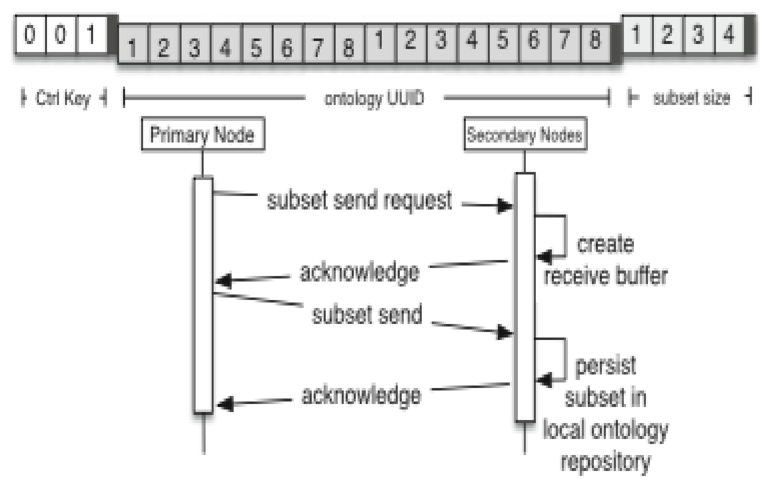


Fig. 7 Ontology subset replication sequence diagram

mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea

Communication and Sequence Diagrams.(2/2)

Fig. 8 Ontology change request sequence diagram

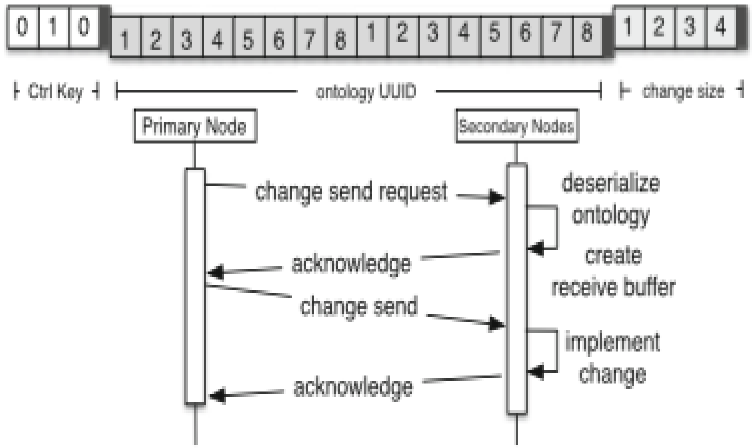
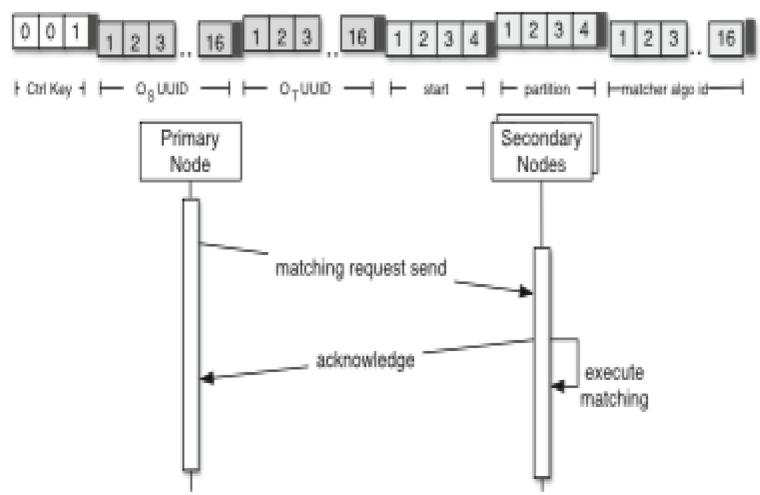


Fig. 10 Ontology matching request sequence diagram



mbilalamin@oslab.khu.ac.kr, Ubiquitous Computing Lab, Dept. of Computer Engineering, Kyung Hee University, South Korea