一种基于 Spark 的大规模语义数据分布式推理框架

陈 恒
(大连外国语大学软件学院 大连 116044)

摘 要 随着大规模语义数据的涌现,研究高效的并行化语义推理成为热点问题之一。现有推理框架大多存在可扩展性方面的不足,难以满足大规模语义数据的需求。针对现有推理框架的不足,提出一种基于 Spark 的大规模语义数据分布式推理框架。该框架主要包括语义建模、规则提取和基于 Spark 的并行推理机等 3 个模块。通过过程分析和推理实例验证,提出的分布式并行推理的计算性能($T(n)=O(\log_2 n)$)远远优于顺序式推理的计算性能($T(n)=O(n)$)。

关键词 Spark,并行化语义推理,分布式框架,语义大数据
中图法分类号 TP181 **文献标识码** A

Spark Based Large-scale Semantic Data Distributed Reasoning Framework

CHEN Heng
(School of Software, Dalian University of Foreign Languages, Dalian 116044, China)

Abstract With the emergence of large-scale semantic data, the study of efficient parallel semantic reasoning has already become a hot topic. In terms of scalability, most of the existing reasoning frameworks still have deficiencies, so it is hard to meet the needs of large-scale semantic data. To solve this problem, a distributed reasoning framework for large-scale semantic data based on Spark was proposed in this paper, which is composed of 3 modules, including semantic modeling, rule extraction and Spark-based parallel reasoning. The result of the process analysis and reasoning instance reveals that computing performance of the proposed distributed parallel reasoning ($T(n)=O(\log_2 n)$) is far better than that of sequential reasoning ($T(n)=O(n)$).

Keywords Spark, Parallel semantic reasoning, Distributed framework, Semantic big data

1 引言

当前,随着语义 Web、物联网(IoT)、云计算、移动互联等 IT 技术的突飞猛进,业内数据的迅速膨胀成了多数行业共同面对的大挑战和大机遇,因而,数据成为话语权的大数据时代已经悄悄地到来。在数据爆炸的同时,语义 Web 数据尤其是 RDF(Resource Description Framework)[1]三元组正以十亿级乃至百亿级的规模产生。例如,表 1 是近几年 W3C 的 SWEO (Semantic Web Education and Outreach)研究小组统计的 RDF 语义数据规模[2]。由于海量语义数据的涌现,针对语义的分布式推理框架、机制和方法成为行业迫切需要解决的必要问题。

表 1 RDF 语义数据规模(个)

截止时间	数据集	RDF 三元组
2010 年 9 月	203	250 亿+
2011 年 9 月	295	310 亿+
2012 年 3 月	325	520 亿+
2014 年 9 月	570	未知

由谷歌倡导的 MapReduce 模式成为当前大数据处理中最为成功、最易使用的并行计算技术之一[3]。目前已有一些推理机制是基于 MapReduce 的并行推理机制[4-7]。文献[4]优化了执行次序,提高了执行效率,但没有为跨本体的关联数据实现分布式推理。文献[5]提供了跨本体的复杂关联语义数据推理功能,但只适用于静态语义数据的推理。文献[6,7]将以往的推理技术完全运用到 MapReduce 模式下,每次推理任务需要多个 MapReduce 作业协作完成,因此效率较低。

Apache Spark 是一个快速的大规模数据处理引擎,在内存中的运行速度比 Hadoop MapReduce 快 100 倍以上,可运行在 Hadoop, Mesos, standalone 以及云平台上,可访问 HDFS, Cassandra, HBase 和 S3 等数据源[8]。为对海量语义数据进行高效推理,本文在分析语义推理有关内容的基础上,提出一种基于 Spark 的大规模语义数据分布式推理框架。

本文第 2 节详细介绍语义推理的相关概念;第 3 节介绍语义推理的相关工作;第 4 节主要介绍语义推理问题的形式化定义;第 5 节提出一种基于 Spark 的大规模语义数据分布式推理框架;第 6 节利用一个实例验证了该推理框架的性能;最后对全文进行总结。

2 语义推理相关概念

RDF 是一种数据模型(RDF 图),是谓词逻辑的一种特殊形式,具有形式化的语义表示,以便计算机理解它所表达的语

本文受国家自然科学基金项目(61371090),辽宁省自然科学基金项目(2015020017),大连外国语大学校级科研项目(2014XJQN09)资助。
陈 恒(1982-),男,硕士,讲师,主要研究方向为智能信息处理,E-mail:chenhengdl@126.com。

义信息。RDF 的基本构造为一个三元组，表示为主谓宾的形式。其中主语是资源，资源以唯一的统一资源标识符（Uniform Resource Identifiers，URI）来表示，可以是网络上的虚拟概念、信息，也可以是现实中的事和物；谓语是属性，用来描述资源及它们的关系，也由 URI 表示；宾语是资源或文本。

RDFS 即 RDF Schema，是针对 RDF 模型的一个基本类型系统，是一种本体语言。可以使用 RDFS 定义被描述资源的类，定义属性的特征，并约束类和关系的可能组合。RDF 结合 RDFS 语义定义的规则具有表示隐含数据的能力。这种表示隐含数据的过程称为推理过程，即在已知一个 RDF 三元组集合的条件下，依据 RDFS 规则推理后产生新的三元组。例如，有如下三元组｛〈Student，rdfs：subClassOf，Person〉，〈Jenny，rdf：type，Student〉｝，根据规则（s rdf：type x & x rdfs：subClassOf y=＞ s rdf：type y）可知，三元组集合中还隐含着三元组〈Jenny，rdf：type，Person〉。

语义推理所解决的主要问题是如何利用推理机从已知的语义数据中求解出一些未知语义数据，从而使未知的语义数据表现出来。推理可分为两种：1）前向推理，即从一个已知的条件或事实开始，连续结合推理规则求解出相关结论；2）后向推理，即从目标或结论开始，不断利用推理规则进行目标或结论消解来完成推理。前向推理的优点是查询速度快；缺点是推理隐含数据的时间较长，并且占用较多的存储空间。当前有一些前向推理系统，如 WebPIE[9]，Sesame[10] 和 YARM[4]。后向推理的优点是不仅不需要大容量的空间存储无用的结果，而且也不需要数据事先处理的额外时间；缺点是查询性能较低。本文研究的推理均指前向推理。

3 相关工作

目前在并行化推理方面存在众多研究，按照实现方法的不同，主要分为以下几种：

（1）基于 Hash 技术的分布式推理方法[11,12]。此类方法把数据保存在分布式 Hash 表中，推理时需要不停地访问 Hash 表中的数据以免重复。该类方法主要面临着负载不均衡的问题，在处理大规模数据时易产生性能瓶颈。

（2）基于数据划分的推理方法[13,14]。此类方法主要是通过对语义规则或数据进行划分来实现语义推理的并行化，但这些方法没有提供大数据集下的实验结果，也不能处理复杂关联语义数据的推理问题。

（3）基于 P2P 的推理方法[15-17]。此类方法以“divide-conquer-swap”的方式执行前向推理，即为基于 P2P 自组织网络的并行推理方法。该方法在负载均衡、可扩展性、推理速率、加速比等方面表现较好，但因没有全局控制器而存在控制方面的缺陷。

（4）基于 MapReduce 的推理方法[4-7,9,18-20]。此类方法在底层架构上采用 Hadoop MapReduce，具有良好的综合性能。MapReduce 的推理算法由文献[7]第一次提出，但没有给出实质性实验结果。该算法提出后，得到了业界的强烈关注，众多研究人员开始探索基于 MapReduce 的特定应用领域的推理算法。如文献[4]结合 RDFS 规则特征和 MapReduce 框架特点实现了一种面向大规模 RDF 语义数据的推理引擎 YARM；文献[18]提出一种基于Map-Reduce的可扩展的语义数据的 Horn-like 推理方法；文献[19]提出一种基于 MapReduce 的并行化规则推理方法；文献[20]提出一种基于Map-Reduce的面向大规模本体的增量并行化推理方法。这些工作的研究为大规模语义数据的分布式并行化推理做出了极大贡献。

本文提出的框架是一种基于 Spark 的分布式推理框架。目前已有研究者开始探讨基于 Spark 的并行化语义推理方法[21,22]。

4 大规模语义数据推理

本文不仅关注语义数据的规模，而且将重点关注语义数据的复杂关联关系。复杂关联语义数据是指那些存在关联关系但又分布在不同领域本体中的实体。复杂关联语义数据推理是指采用语义推理的方法提取复杂关联语义数据之间隐藏的关联信息，如图 1 所示。

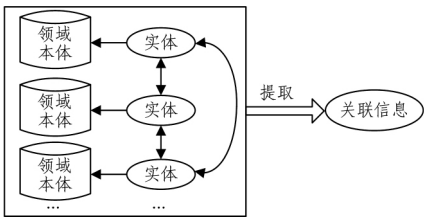

图 1 复杂关联语义数据推理

经典的推理方法，如 RDFS，SWRL，OWL 都是依据其内部本体词汇的规则来进行推理。而当我们从多个不同领域本体中推理复杂关联语义数据时，这些经典方法就不再适用。本文采用文献[5]提到的基于属性链的推理方法，该方法可以有效地推理不同领域本体之间的语义数据。

4.1 语义推理问题的形式化定义

为了清晰描述推理问题，首先介绍下面几个概念。

定义 1（推理规则链，Reasoning Rule Chain，RRC） 由一组有序的三元组组成。一个基本规则 R 的三元组可形式化为〈C_1，P，C_2〉，它表示类 C_1 与类 C_2 通过属性 P 关联起来。例如，一个长度为 n 的推理规则链可形式化为 $RRC_0=\{R_0\langle C_0,P_0,C_1\rangle,R_1\langle C_1,P_1,C_2\rangle,R_2\langle C_2,P_2,C_3\rangle,\cdots,R_{n-1}\langle C_{n-1},P_{n-1},C_n\rangle\}$。

定义 2（推理属性链，Reasoning Property Chain，RPC） 是由一个或多个基本规则的属性依次合并运算（⊕）得到。随着推理的进行，多个连续的 RPC 将依次通过⊕运算合并成一个新的 RPC。例如：如果存在长度为 n 的推理规则链 $RRC_0=\{R_0\langle C_0,RPC_0,C_1\rangle,R_1\langle C_1,RPC_1,C_2\rangle,\cdots,R_{n-1}\langle C_{n-1},RPC_{n-1},C_n\rangle$，那么可推出另外一个基本规则 $R_{new}\langle C_0,RPC_{new},C_n\rangle$，其中 RPC_{new} 为 $RPC_0\oplus RPC_1\oplus\cdots\oplus RPC_{n-1}$。

定义 3（属性链集合，Property Chain Set，PCS） 由一组 RPC 构成。初始条件 $PCS_0=\{RPC_0,RPC_1,\cdots,RPC_{n-1}\}$。在推理过程中，$RPC$ 不停合并计算，因此，PCS 也随之不断改变。

定义 4（属性 ID（PID）） 即为 PCS 的每个 RPC 分配一个 ID。第 1 个 RPC 的 ID 为 0，第 2 个为 1，…，最后一个为 $n-1$。缺省状态下，某个基本规则三元组的属性不属于 PCS，则它的属性 ID 为 -1。

根据以上的定义，可将推理问题形式化定义为问题 1。

问题 1 将一个四元组〈S_0，PCS_0，C_0，C_{n-1}〉输入到推理系统，推理系统需要计算出解集 $X=\{\langle O_0,RPC,O_{n-1}\rangle\mid O_0\in$

$C_0, O_{n-1} \in C_{n-1}, RPC = RPC_0 \oplus RPC_1 \oplus \cdots \oplus RPC_{n-1}\}$。其中，$S_0$ 表示一个初始的实例三元组集合，S_0 中的一个三元组可表示为 $O_k\langle C_k, P_k, D_k\rangle$，$C_k$ 代表类 C 的一个实体或实例，D_k 代表类 D 的一个实体或实例，P_k 是 PID 为 k 的 RPC；PCS_0 表示长度为 n 的初始属性链集合 $PCS_0 = \{RPC_0, RPC_1, \cdots, RPC_{n-1}\} = \{P_0, P_1, \cdots, P_{n-1}\}$；$C_0$ 和 C_{n-1} 依次表示需要推理出具有联系的两类实体对象。

4.2 推理实例

考虑如下推理情景：事先已知一个由大量位于不同本体库中的实例三元组组成的集合 S_0，又已知这些本体之间的语义规则 PCS_0，根据已知条件推理获得其中类 C 与类 H 之间具有关联关系的语义数据。

基于 4.1 节的形式化定义，可将此情景的输入形式化为一个四元组 $Q_0\langle S_0, PCS_0, C, H\rangle$，其中 $S_0 = \{R_0\langle C_0, P_0, D_0\rangle, R_1\langle D_0, P_1, E_0\rangle, R_2\langle E_0, P_2, F_0\rangle, R_3\langle F_0, P_3, G_0\rangle, R_4\langle G_0, P_4, H_0\rangle, R_5\langle C_1, P_0, D_0\rangle, R_6\langle D_0, P_6, E_1\rangle\}$，$PCS_0 = \{P_0, P_1, P_2, P_3, P_4\}$。

4.3 推理思路

整体过程：通过多次迭代操作，推理得到 4.2 节中类 C 与类 H 之间具有关联关系的语义数据。在每次迭代中，完成符合连接条件的三元组的推理连接，经过多次迭代或转换，得到最终解集。具体过程：利用已知的两个三元组的连接操作推理得到一个新的三元组，在一次迭代中首先需要找出所有符合连接条件的三元组对，然后通过指定的推理规则得到新的三元组对，新三元组对又将作为下一次迭代的输入，直至推理出合理的结果。这里需要定义连接候选集。

定义 5(连接候选集) 由符合连接条件的所有三元组构成的集合。在每次迭代中，推理机首先根据连接条件得到对应的候选集，其次只需对候选集的元素进行连接合并操作。假如 4.2 节例子的 R_1 和 R_2 符合条件，那么在候选集中就需要添加一个二元组$\langle R_1, R_2\rangle$，继而通过此元素推理出一个新的元组 $R_K\langle D_0, P_1 \oplus P_2, F_0\rangle$。

一个比较简洁的推理思路(见图 2)就是依次顺着属性链在每次迭代中进行前 2 类实体的连接，即首先进行类 C 和类 D 所有符合条件的三元组的推理，其次将推理结果与类 E 进行连接，依次执行，直至推理出与类 H 相关联的语义信息。

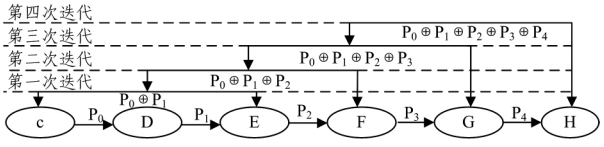

图 2 基于属性链的顺序推理

上述顺序化推理思路与并行化推理完全相悖，它的运行性能较低，并需要 $n-1$ 次迭代过程，同时浪费资源，为此需要一种高效率的并行化分布式推理框架来加速推理。

4.4 选择 Spark 的理由

新的分布式计算技术日益成熟，像 MapReduce 一样越来越有可能在服务器集群上实现大规模的运算。然而，MapReduce 的性质使得它进行迭代运算时效率低下。为了解决这个问题，已经提出了一些替代技术。Spark[23] 是一个并行计算平台，它支持迭代算法在分布式内存中高效执行，该分布式内存抽象命名为弹性分布式数据集(Resilient Distributed Datasets，RDDs)。

鉴于 Spark 擅长迭代运算的特点，本文设计一种基于 Spark 的并行化推理框架对海量语义数据进行高效推理。

5 基于 Spark 的推理框架

基于 Spark 的语义推理框架主要由语义建模、规则提取和基于 Spark 的并行推理机等 3 个子模块组成，如图 3 所示。

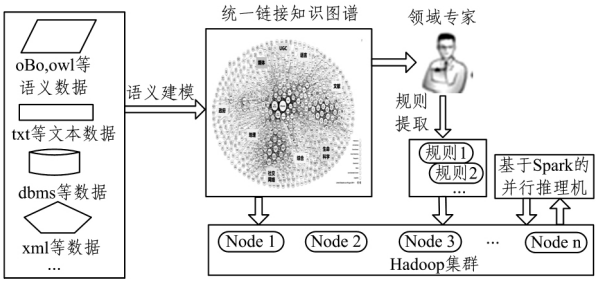

图 3 基于 Spark 的语义推理框架

语义建模子模块的功能是将来源于多领域的、大规模的、异构的(obo，owl，txt，dbms，xml 等)数据通过语义和文本处理的方法构建成一个统一链接知识图谱。构建统一链接知识图谱需要使用巨大的领域本体来定义相应的词汇，包含类(概念)、属性以及关系等。

规则提取子模块的功能是领域专家参考领域本体，从统一链接知识图谱中根据语义概念模型抽取语义推理规则。语义推理的核心理念是通过复杂关联关系将不同领域的数据链接起来，进而可以通过语义规则的导航进行智能化更高的关联分析或者数据挖掘。因此，规则是语义推理的基础。

基于 Spark 的并行推理机是以统一链接知识图谱中的实例三元组为原始数据集，以语义推理规则中的语义关系为介质，在 Spark 中进行迭代运算而且输出推理结果。

6 推理实例解析

图 2 中顺序式推理效率低下的原因是推理条件比较严格，每次迭代推理所使用的连接候选集仅能选择前 2 类 PID 为 0 或者 1 的实体三元组，其他的三元组并没有得到并行处理。因此需要发掘一种灵活性更高的连接策略，目的是在每次迭代中能够推理出尽可能多的结果，减少总的迭代次数，提高推理效率。根据以上设想，对于输入的实例三元组需要分两种情况制定推理连接条件：1)PID 值相邻的三元组；2)PID 值非相邻的三元组。

对于 PID 值相邻的两个三元组，连接条件如下：

(1)PID 值较小的三元组的宾语是另一个三元组的主语。

(2)假如一个三元组的 PID 值为奇数 k，那么该三元组只能和它前面相邻的三元组(即 PID 值为 $k-1$)连接；假如一个三元组的 PID 值为偶数 k，那么该三元组只能和它后面相邻的三元组(即 PID 值为 $k+1$)连接。

对于 PID 值非相邻的两个三元组，一个三元组的宾语是另一个三元组的主语即可连接。

基于以上推理连接条件的制定规则，推理系统首先在一次迭代中将所有的实例三元组分为 $(n+1)/2$ 组，其次在每组实例中找出候选集合，然后依据候选集进行并行推理，推理得到的结果又将作为下次推理的输入，循环迭代直至推理机推理出最后的结果，因此 4.2 节中推理实例整个算法的时间复

杂度由 $O(n)$ 降为 $O(\log_2 n)$，其迭代过程如图 4 所示。

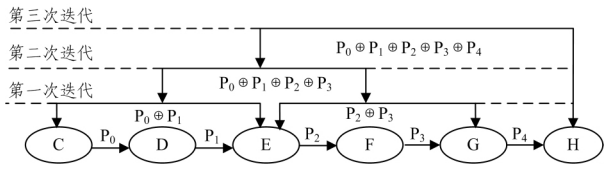

图 4　基于属性链的并行推理

对于 4.2 节的实例，基于属性链的并行推理过程如下：

1）第 1 次迭代

S_0 中符合连接条件的候选集为 $\{\langle R_0, R_1\rangle, \langle R_1, R_5\rangle, \langle R_2, R_3\rangle\}$，依据候选集推理出新的三元组 $\{R_7\langle C_0, P_0 \oplus P_1, E_0\rangle, R_8\langle E_0, P_2 \oplus P_3, G_0\rangle, R_9\langle C_1, P_0 \oplus P_1, E_0\rangle\}$。此时，输入三元组更新为 $S_1 = \{R_4\langle G_0, P_4, H_0\rangle, R_7\langle C_0, P_0 \oplus P_1, E_0\rangle, R_8\langle E_0, P_2 \oplus P_3, G_0\rangle, R_9\langle C_1, P_0 \oplus P_1, E_0\rangle\}$，属性链集合 PCS_0 更新为 $PCS_1 = \{P_0 \oplus P_1, P_2 \oplus P_3, P_4\}$，输入四元组更新为 $Q_1\langle S_1, PCS_1, C, H\rangle$。

2）第 2 次迭代

第 1 次迭代结果 G_1 中符合条件的候选集为 $\{\langle R_7, R_8\rangle, \langle R_8, R_9\rangle\}$，依据候选集推理出新的三元组 $\{R_{10}\langle C_0, P_0 \oplus P_1 \oplus P_2 \oplus P_3, G_0\rangle, R_{11}\langle C_1, P_0 \oplus P_1 \oplus P_2 \oplus P_3, G_0\rangle\}$。此时，输入三元组更新为 $G_2 = \{R_4\langle G_0, P_4, H_0\rangle, R_{10}\langle C_0, P_0 \oplus P_1 \oplus P_2 \oplus P_3, G_0\rangle, R_{11}\langle C_1, P_0 \oplus P_1 \oplus P_2 \oplus P_3, G_0\rangle\}$，属性链集合 PCS_1 更新为 $PCS_2 = \{P_0 \oplus P_1 \oplus P_2 \oplus P_3, P_4\}$，输入四元组更新为 $Q_2\langle G_2, PCS_2, C, H\rangle$。

3）第 3 次迭代

第 2 次迭代结果 G_2 中符合条件的候选集为 $\{\langle R_4, R_{10}\rangle, \langle R_4, R_{11}\rangle\}$，依据候选集推理出新的三元组 $\{R_{12}\langle C_0, P_0 \oplus P_1 \oplus P_2 \oplus P_3 \oplus P_4, H_0\rangle, R_{13}\langle C_1, P_0 \oplus P_1 \oplus P_2 \oplus P_3 \oplus P_4, H_0\rangle\}$。此时，$PCS = \{P_0 \oplus P_1 \oplus P_2 \oplus P_3 \oplus P_4\}$，推理结束。

结束语　为了实现大规模语义数据的高效推理计算，本文在分析语义推理有关知识的基础上，提出一种基于 Spark 的分布式推理框架，并通过一个实例验证了顺序式推理与分布式并行推理的计算性能。

在将来研究工作中，我们将继续对基于 Spark 的分布式推理框架进行改进，并实现高效的分布式推理引擎。

参 考 文 献

[1] W3C. Resource Description Framework[EB/OL]. [2015-01-25]. http://www.w3.org/RDF

[2] W3C. Linking open data[EB/OL]. [2015-01-25]. http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/Linking-OpenData

[3] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113

[4] 顾荣，王芳芳，袁春风，等. YARM：基于 MapReduce 的高效可扩展的语义推理引擎[J]. 计算机学报，2015，38(1)：74-85

[5] 陈曦，陈华钧，顾珮嵚，等. 一种基于 Hadoop 的语义大数据分布式推理框架[J]. 计算机研究与发展，2013(2)：103-113

[6] Urbani J, Kotoulas S, Oren E, et al. Scalable distributed reasoning using mapreduce[C]// Proceedings of the 8th International Semantic Web Conference. Washington, USA, 2009: 634-649

[7] Mika P, Tummarello G. Web semantics in the clouds[J]. Intelligent Systems, IEEE, 2008, 23(5): 82-87

[8] The Apache Software Foundation. Spark [EB/OL]. [2016-02-01]. https://spark.apache.org

[9] Urbani J, Kotoulas S, Maassen J, et al. WebPIE: A Web-scale parallel inference engine using MapReduce[J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2012, 10: 59-75

[10] Broekstra J, Kampman A, Van Harmelen F. Sesame: A generic architecture for storing and querying RDF and RDF schema[C]// Proceedings of the 1st International Semantic Web Conference on The Semantic Web. Sardinia, Italy, 2002: 54-68

[11] Kaoudi Z, Koubarakis M. Distributed RDFS reasoning over structured overlay networks[J]. Journal on Data Semantics, 2013, 2(4): 189-227

[12] Fang Q, Zhao Y, Yang G, et al. Scalable distributed ontology reasoning using DHT-based partitioning[C]// Proceedings of the 3rd Asian Semantic Web Conf. Berlin: Springer, 2008: 91-105

[13] Soma R, Prasanna V K. Parallel inferencing for OWL knowledge bases[C]// Processing ofthe 37th International Conference on Parallel Processing. Portland, USA, 2008: 75-82

[14] Weaver J, Hendler J A. Parallel materialization of the finite rdfs closure for hundreds of millions of triples[C]// Proceedings of the 8th International Semantic Web Conference. Washington, USA, 2009: 682-697

[15] Kotoulas S, Oren E, Van Harmelen F. Mind the data skew: distributed inferencing by speeddating in elastic regions[C]// Proceedings of the 19th International Conference on World Wide Web. ACM, 2010: 531-540

[16] Tsatsanifos G, Sacharidis D, Sellis T. On enhancing scalability for distributed RDF/S stores[C]// Proceedings of the 14th International Conference on Extending Database Technology. Uppsala, Sweden, 2011: 141-152

[17] Muhleisen H, Dentler K. Large-scale storage and reasoning for semantic data using swarms[J]. IEEE Computational Intelligence Magazine, 2012, 7(2): 32-44

[18] Wu H, Liu J, Ye D, et al. Scalable Horn-Like rule inference of semantic data using MapReduce[C]// Proceedings of the 7th International Conference (KSEM 2014). Sibiu, Romania, 2014: 270-277

[19] Wu H, Liu J, Ye D, et al. A distributed rule execution mechanism based on MapReduce in sematic web reasoning[C]// Proceedings of the 5th Asia-Pacific Symposium on Internetware. ACM, 2013: 6

[20] Liu B, Huang K, Li J, et al. An incremental and distributed inference method for large-scale ontologies based on MapReduce paradigm[J]. IEEE Transactions on Cybernetics, 2015, 45(1): 53-64

[21] Gu R, Wang S, Wang F, et al. Cichlid: Efficient Large Scale RDFS/OWL Reasoning with Spark[C]// 2015 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE, 2015: 700-709

[22] Kim J M, Park Y T. Scalable OWL-Horst ontology reasoning using SPARK[C]// 2015 International Conference on Big Data and Smart Computing (BigComp). IEEE, 2015: 79-86

[23] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: cluster computing with working sets[J]. HotCloud, 2010(10): 10