http://qks.cqu.edu.cn

文章编号:1000-582X(2013)02-056-07

云计算环境下的 SWRL 规则分布式推理框架

李 韧[a],杨 丹[b],胡海波[b],谢 娟[b],吴云松[a],傅 鹂[b]
(重庆大学 a.计算机学院;b.软件学院,重庆市 400044)

摘 要:为解决传统推理引擎在进行大规模 OWL 本体数据的 SWRL 规则推理时存在的计算性能和可扩展性不足等问题,提出了云计算环境下的 SWRL 规则分布式推理框架 CloudSWRL。根据 SWRL 规则语义,并以 Hadoop 开源云计算框架为基础,设计了 OWL 本体在 HBase 分布式数据库中的存储策略,定义了 SWRL 规则解析模型和相关推理中间数据模型,提出了在 DL-safe 限制下基于 MapReduce 的 SWRL 规则分布式推理算法。实验结果表明,在对大规模 OWL 本体进行 SWRL 规则推理时,CloudSWRL 框架在计算性能和可扩展性方面均优于传统推理引擎。

关键词:语义 Web;云计算;本体;并行推理;SWRL
中图分类号:TP311 **文献标志码**:A

A cloud computing based SWRL distributed reasoning framework

LI Ren[a], *YANG Dan*[b], *HU Hai-bo*[b], *XIE Juan*[b], *WU Yun-song*[a], *FU Li*[b]
(a. College of Computer Science; b. School of Software Engineering,
Chongqing University, Chongqing 400044, China)

Abstract: With the explosion of semantic web technologies, large amounts of OWL ontologies are common place. Conventional rule engines inevitably meet the bottleneck of computing performance and scalability. A cloud computing based SWRL distributed reasoning framework named CloudSWRL is proposed. Based on the Hadoop open-source framework and SWRL semantics, the storage schema for OWL ontologies is designed to implement efficient data retrieving from HBase. Some novel data models for SWRL rules and intermediate data are defined. At last, a MapReduce paradigm based distributed SWRL reasoning algorithm is proposed under DL-safe restriction. An experiment on a simulation environment shows our framework is more efficient and scalable than conventional rule engines when reasoning over large-scale of OWL data.

Key words: Semantic Web; cloud computing; ontology; parallel reasoning; SWRL

随着语义 Web 的不断发展,建立在资源描述框架(RDF)[1]之上的 Web 本体描述语言(OWL)已被广泛地应用于各种领域的本体建模和推理[2]。由于基于描述逻辑的 OWL 无法对一般形式的规则进行描述,W3C 将 OWL 子语言 OWL DL、OWL Lite 与 RuleML 相结合,提出了具备更强逻辑表达能力的语义 Web 规则描述语言 SWRL[3],但其推理不可判定[4]。目前,业界还没有一个专用的 SWRL 规则推理引擎,研究人员主要使用 protégé 工具的 SWRLTab 插件将 OWL 本体映射入 Jess 事实库,将 SWRL 规则映射入 Jess 规则库,并调用 Jess 规则引擎对已知 OWL 个体进行可判定的 SWRL 规则推

收稿日期:2012-01-20
基金项目:国家自然科学基金重点资助项目(91118005);国家自然科学基金青年科学基金资助项目(51005260);重庆市自然科学基金重点项目(CSTC-2011BA2022)
作者简介:李韧(1985-),男,重庆大学博士,主要从事语义 Web、云计算及软件工程等方向研究,(Tel)13594154026;(E-mail)renli@cqu.edu.cn。

http://qks.cqu.edu.cn

理[5]。同时,Pellet 和 KAON2 等本体推理引擎也提供了在 DL-safe 限制[6]下的可判定 SWRL 推理支持。但随着 OWL 本体数据量的不断增长,上述单机环境下运行的推理引擎由于需要将本体数据和规则载入内存,在对大规模本体数据进行 SWRL 推理时,存在计算性能和可扩展性不足等问题[7]。另外,在语义 Web 数据管理研究领域,传统关系型数据库作为本体数据存储载体被广泛应用,并主要采用基于三元组的策略对本体数据进行存储。虽然学术界相继提出了垂直存储、水平存储和模式生成等优化方案,但依然存在数据存储可扩展性不足和查询效率低等问题[8]。

自 2007 年 Google 和 IBM 提出云计算概念以来,云计算因其具备高性能、可扩展的海量数据计算和存储能力已经成为产业界和学术界在信息技术领域的最新研究方向[9]。目前,基于 GFS 分布式文件系统模型[10]、MapReduce 分布式计算模型[11]和 BigTable 分布式数据库模型[12]实现的开源 Hadoop 平台[13]已成为云计算研究领域最广泛使用的数据密集型计算和存储模型[14]。

近年来,为解决传统语义 Web 数据管理和推理工具在处理海量本体数据时存在的不足,学术界逐步形成了以语义 Web 和云计算研究领域相结合的新研究方向[15]。文献[16]提出了一个云计算环境下的分布式 SPARQL 本体数据查询框架,并证明了在处理大规模 RDF 本体数据查询时,该框架在可扩展性和查询效率等方面均优于传统工具和方法。为实现高性能和可扩展的语义 Web 本体推理,文献[17]提出了基于 MapReduce 的分布式本体推理引擎 WebPIE,实现了云计算环境下的 RDF 图闭包计算及 OWL Horst 推理。文献[18]提出了一个基于 HBase 的分布式语义 Web 数据管理框架,通过与 MySQL 集群数据库进行对比实验,证明了云计算环境下的语义 Web 数据管理方法在可扩展性和数据查询效率等方面优于传统关系型数据库存储模式。但到目前为止,在应对大规模语义 Web 本体数据的 SWRL 规则推理方面,业界仍然缺少一种高性能、易扩展的解决方案。

因此,以 Hadoop 平台为基础,研究并提出了云计算环境下的 SWRL 规则分布式推理框架 CloudSWRL。首先描述了框架的功能模块架构,设计了 OWL 本体在 HBase 中的存储策略,然后阐述了 CloudSWRL 框架中相关术语和解析模型定义,并以此为基础,提出了基于 MapReduce 的推理计划生成算法和在 DL-safe 限制下 SWRL 规则分布式推理算法。最后,通过进行对比实验和可扩展性实验,证明了在处理大规模 OWL 本体数据的 SWRL 规则推理时,CloudSWRL 框架在计算性能和可扩展性方面均优于传统推理引擎。

1　框架功能模块架构设计

CloudSWRL 框架以开源 Hadoop 云计算平台为基础,设计了数据预处理模块、规则解析模块、数据存储模块、推理计划生成模块和分布式规则推理模块。其功能模块架构设计如图 1 所示。

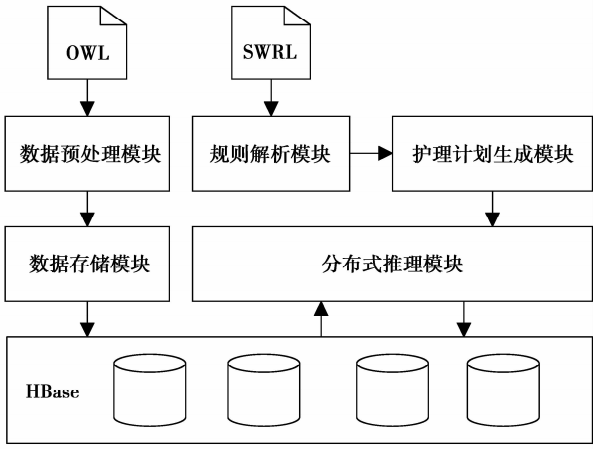

图 1　CloudSWRL 功能模块架构设计

以用户输入的 OWL 本体数据和 SWRL 规则作为数据源,数据预处理模块首先调用 Jena API[19]将基于 XML 格式的本体文件转换为三元组格式的 N-Triple 文件,并将其以数据流的方式导入 HDFS[13]。然后,根据研究的 OWL 本体存储策略,数据存储模块将三元组格式的本体数据并行地存入 HBase 数据库。规则解析模块将 SWRL 规则转换为定义的 SWRL 规则解析模型,并以此为输入,计算生成推理计划模型。最后,分布式规则推理模块根据推理计划和 SWRL 分布式推理算法,在 MapReduce 中计算并输出推理结果。

2　基于 HBase 的 OWL 本体存储策略

作为 BigTable 模型的开源实现,HBase 数据库以基于列的方式对数据进行分布式存储。HBase 数据表使用行键(row key)对每行数据进行唯一性标识,并提供了类似 B+树的高效索引。每个数据表中可定义多个列族(column family),每个列族可由多个列(column)构成,并且列名及列数目可由程序动态任意指定[20]。

在 CloudSWRL 框架中,OWL 本体数据以三元

http://qks.cqu.edu.cn

组模式进行存储。根据SWRL推理规则的特点,框架将OWL个体断言集存储于数据表A_SP_O和A_PO_S中,将OWL术语集存储于表T_SP_O中,其存储模型如图2和图3所示。

RowKey	Column Family				Timestamp
	O (1)	O (2)	...	O (N)	
<S, P>	null	null	...	null	Time T

图2 T_SP_O表和A_SP_O表数据存储模型

RowKey	Column Family				Timestamp
	S (1)	S (2)	...	S (N)	
<P, O>	null	null	...	null	Time T

图3 A_PO_S表数据存储模型

T_SP_O表的行键由本体术语集三元组中的主语和谓语联合构成,对应的三元组的宾语值作为列名存储在列族中,以此减少数据存储空间。A_SP_O表与T_SP_O表具有相同的存储结构。A_PO_S表的行键由本体个体断言集三元组中谓语和宾语联合组成,所对应的主语以列名的形式存储。基于上述存储策略,通过结合HBase提供的Scan区域检索和Get数据检索机制,可实现本体数据的高效检索。

3 相关术语及解析模型

根据W3C对SWRL语言的定义[3],每条SWRL规则由规则前件(Body)和规则后件(Head)两部分组成。其中前件和后件均可由零或多个规则原子(Atom)的合取组成。SWRL规则的形式化描述如式(1)

$$(\wedge_{i=1}^{n}\mathrm{Atom}_i)\rightarrow(\wedge_{j=1}^{m}\mathrm{Atom}_j), \quad (1)$$

其中$n\geqslant 0$,$m\geqslant 0$。规则原子可由$C(x)$、$P(x, y)$、same$As(x, y)$、differentFrom(x, y)或SWRL内置函数构成,其中x和y可为变量、OWL个体或数据常量,C代表OWL类描述,P表示OWL属性。在本体库Ω中,对规则原子$C(x)$,若有个体a为类C的实例,即$C(a)$成立,称a为$C(x)$的一个解释。同理,对规则原子$P(x, y)$,若有实例组(a, b)满足$P(a, b)$,称(a, b)为$P(x, y)$的一个解释。

为实现可判定的SWRL规则推理,CloudSWRL框架在DL-safe限制下对本体库中已知OWL个体和公理进行推理。目前,框架支持$C(x)$和$P(x, y)$类型的SWRL规则原子,并规定SWRL规则前件至少包含一个规则原子,规则后件中有且仅有一个规则原子。

由于SWRL规则原子与OWL类和属性存在一一对应关系,而基于RDF的OWL可用三元组形式描述,因此SWRL规则原子可用如表1所示的映射关系表示为三元组形式,且具备相同的语义,其中?x和?y表示变量,a和b表示OWL个体或常量。

表1 SWRL规则原子的三元组表示

SWRL规则原子	三元组表示
$C(?\ x)$	$(?\ x, P(\mathrm{rdf:type}), C)$
$P(?\ x, ?\ y)$	$(?\ x, P, ?\ y)$
$P(a, ?\ y)$	$(a, P, ?\ y)$
$P(?\ x, b)$	$(?\ x, P, b)$

表2描述了框架在进行SWRL规则原子的本体数据检索时,各类规则原子对应的HBase数据表和检索命令。

表2 规则原子及其对应的数据表和检索命令

SWRL规则	数据表	查询命令
$C(?\ x)$	A_PO_S	Get(rdf:type, C)
$P(?\ x, b)$	A_PO_S	Get(P, b)
$P(a, ?\ x)$	A_SP_O	Get(a, P)
$P(?\ x, ?\ y)$	A_PO_S	Scan(P)

基于如表1所示的映射关系,给出SWRL规则及其规则原子在CloudSWRL框架中的解析模型。

定义1 SWRL规则原子Atom,可用如表1所示的映射关系,将Atom解析为规则原子解析模型$A=(A(s), A(p), A(o))$,其中$A(p)$为OWL本体属性,$A(s)$和$A(o)$可为变量、OWL个体或常量。

定义2 任一SWRL规则可用规则解析模型$R=(B, H)$表示,其中B为前件规则原子解析模型集合$B=\{A_1, \cdots, A_n\}$,$(n\geqslant 1)$;H为后件规则原子解析模型集合$H=\{A'\}$。

根据SWRL规则语义,对于任一SWRL规则,其含义为:如果前件中的所有规则原子同时成立,则后件一定成立[7]。即当在本体库中存在至少一个解释使得SWRL前件规则原子同时成立时,那么也存在至少一个解释使得*SWRL*后件规则原子成立。

http://qks.cqu.edu.cn

定义 3　对规则原子解析模型 A,当 A 中仅存在一个变量 v 时,A 在本体库 Ω 中的解释模型 $I(A)=(v, a)$;当 A 存在变量 v_1 和 v_2 时,$I(A)=((v_1, a), (v_2, b))$。其中 a, b 为在本体库 Ω 中满足 A 中变量的 OWL 个体或常量,即 $A(a)$ 或 $A(a, b)$ 在 Ω 中成立。

定义 4　规则解析模型 R 各规则原子解析模型存在变量 v_1 和 v_2,若 v_1 同时存在于 2 个或多个规则原子解析模型 $A_1,\cdots, A_n,(n\geqslant 2)$,称 v_1 为 $A_1,\cdots, A_n$ 的共同变量;若 v_2 仅存在于一个规则原子解析模型 A 中,称 v_2 为非共同变量。

对 SWRL 规则原子解析模型 A_1 和 A_2,在本体库 Ω 中,若 A_1 存在解释模型的集合 $U_1=\{I_1(A_1),\cdots, I_j(A_1)\},(j\geqslant 1)$,$A_2$ 存在解释模型的集合 $U_2=\{I_1(A_2),\cdots, I_k(A_2)\},(k\geqslant 1)$,当 A_1 和 A_2 存在共同变量 v 时,使得 A_1 和 A_2 同时成立的解释模型 $I(A_1, A_2)\in(U_1\bowtie_v U_2)$,即对集合 U_1 和 U_2 进行以共同变量 v 为基准的连接操作;当 A_1 和 A_2 之间仅存在非共同变量时,$I(A_1, A_2)\in(U_1\times U_2)$,即对集合 U_1 和 U_2 取笛卡尔集。因此,对规则解析模型 $R=(B, H),B=\{A_1,\cdots, A_n\},(n\geqslant 1),H=\{A'\}$,前件规则原子解析模型集合 B 的解释模型 $I(B)\in(\{I(A_1)\}\cdots\{I(A_j)\}\times\{I(A_{(j+1)})\}\times\cdots\times\{I(A_n)\})$,其中,$A_1,\cdots, A_j$ 之间存在共同变量,$A_{(j+1)},\cdots, A_n$ 中仅存在非共同变量。推理结果为 H 中规则原子解析模型 A' 变量对应的解释模型 $I(A')\in\{I(B)\}$。

由于 MapReduce 模型以键值对的形式对数据进行分布式计算,当规则中存在多个不同的共同变量时,SWRL 推理需要由多个 MapReduce 任务(Job)组合完成。

定义 5 推理计划模型 $L=\{\text{job}_1,\cdots,\text{job}_n\},(n\geqslant 1)$,其中 $\text{job}=(DS, \{\text{subjob}_1,\cdots,\text{subjob}_m\}),(m\geqslant 1)$,$DS$ 为标识符,当推理任务以 HBase 为数据源时,DS 标识为 D;当以 HDFS 为数据源时,DS 标识为 F。模型 $\text{subjob}=(\text{Tag}, \{A_1,\cdots, A_j\}),(j\geqslant 1)$,$A$ 为规则原子解析模型,Tag 为标识符,当 $A_1,\cdots, A_j$ 中存在共同变量 v 时,Tag 标识为 v;当 $A_1,\cdots, A_j$ 中仅存在非共同变量时,Tag 标识为 Neg。

定义 6　推理中间数据模型 IKV 为键值对,$\text{IKV}=<(v, I[v]), (\{A\}, I(\{A\}))>$,其中 v 为变量名,$I[v]$ 为 $I(\{A\})$ 中变量 v 的值,$I(\{A\})$ 为满足规则原子解析模型集合 $\{A\}$ 的解释模型。

4　基于 MapReduce 的 SWRL 规则分布式推理算法

MapReduce 编程模型中每个任务由在各个计算节点中运行的 Map 和 Reduce 函数组成。为尽可能减少任务数,提升推理效率,提出 SWRL 规则推理计划生成算法,并以此为基础,在提出 MapReduce 模型下的 SWRL 分布式推理算法。

4.1　推理计划生成算法

为实现云计算环境下基于 SPARQL 的本体数据分布式查询,文献[16]中给出对查询语句中变量 v 取 E-count(v)值的定义,并基于贪心选择策略提出了查询计划的生成方法。笔者对该方法加以改进,提出一种针对 SWRL 规则的推理计划生成算法。该算法执行步骤如下

1)输入 SWRL 规则解析模型 $R=(B, H),B=\{A_1,\cdots, A_n\},(n\geqslant 1),H=\{A'\}$。计算 B 中共同变量 $cv_1,\cdots, cv_i,(i\geqslant 0)$,$B$ 中非共同变量 $nv_1,\cdots, nv_j,(0\leqslant j\leqslant 2n-i)$。设集合 CV、NV 和 U,对 $k=1,\cdots, i$,计算 $N_k\leftarrow E\text{-count}(cv_k)$,再执行 $CV\leftarrow CV\cup\{<cv_k, N_k>\}$。对 $l=1,\cdots, j$,执行 $NV\leftarrow NV\cup\{nv_l\}$。执行 $U\leftarrow B$。

2)计算 $cl\leftarrow\text{length}(CV)$,$cl$ 为集合 CV 中元素个数。当 $cl=0$ 时,执行步骤 4;当 $cl\geqslant 1$ 时,对 CV 按 E-count 值进行升序排列,但若存在 $<cv_s, N_s>\in CV$ 和 $<cv_t, N_t>\in CV$,使得 $cv_s\neq cv_t, N_s=N_t$,则计算 cv_s 和 cv_t 在 B 中出现次数 T_s 和 T_t,若 $T_s>T_t$,将 $<cv_s, N_s>$ 在 CV 中排序优先于 $<cv_t, N_t>$,若 $T_s=T_t$,再计算 cv_s 和 cv_t 在 $A_p(s)$ 出现次数 S_s 和 $S_t,(p=1,\cdots, n)$,如果 $S_s>S_t$,则将 $<cv_s, N_s>$ 在 CV 中排序优先于 $<cv_t, N_t>$。

3)设集合 Temp,参数 $e\leftarrow 1$。设 job_e 标识符 DS 为 D。对 $x=1,\cdots, cl$,取 $cv_x\in CV, A_p\in U,\cdots, A_{(p+q)}\in U,(p\geqslant 1, q\geqslant 1)$,且满足 $cv_x\in A_p,\cdots, cv_x\in A_{(p+q)}$。再执行 $\text{subjob}_x\leftarrow(cv_x, \{A_p,\cdots, A_{(p+q)}\})$,$x\leftarrow x+1, U\leftarrow U-\{A_p,\cdots, A_{(p+q)}\}$,对 $z=p,\cdots,(p+q)$,若 A_z 中存在除 cv_x 以外的共同变量 v_z 且 $v_z\notin\text{Temp}$,执行 $\text{Temp}\leftarrow\text{Temp}\cup\{v_z\}$。最后,对生成的 x 个 subjob 模型,执行 $\text{job}_e\leftarrow(DS, \{\text{subjob}_1,\cdots, \text{subjob}_x\}), L\leftarrow L\cup\{\text{job}_e\}, e\leftarrow e+1$。

4)设集合 W,计算 $nl\leftarrow\text{length}(NV)$,$nl$ 为集合 NV 中元素个数,并设 job_e 标识符 DS 为 D。对 $y=1,\cdots, nl$,取 $nv_y\in NV, A_y\in B$,满足 $nv_y\in A_y$,执行 $W\leftarrow W\cup\{A_y\}, U\leftarrow U-\{A_y\}$,计算 $e\leftarrow e+1$。

5)计算 $c\leftarrow\text{length}(\text{Temp})$,$c$ 为集合 Temp 中元素个数。当 $c\geqslant 1$ 时,对 $r=1,\cdots, c$,取 $v_r\in\text{Temp}, A_d\in B,\cdots, A_{(d+g)}\in B,(d\geqslant 1, g\geqslant 1)$,满足 $v_r\in A_d,\cdots, v_r\in A_{(d+g)}$。执行 $\text{subjob}_r\leftarrow(v_r, \{A_d,\cdots, A_{(d+g)}\})$,对 $u=d,\cdots,(d+g)$,若 A_u 中存在除 v_r

http://qks.cqu.edu.cn

以外的共同变量 v_u 且 $v_u \notin$ Temp，执行 Temp←Temp$\cup\{v_u\}$，c←$c+1$。设 job$_e$ 标识符 DS 为 F，对生成的 r 个 subjob 模型，执行 job$_e$←(DS，{subjob$_1$，…，subjob$_r$})，L←$L\cup$ {job$_e$}，计算 e←$e+1$。

6)设 job$_e$ 标识符 DS 为 F，对 $A' \in H$，执行 subjob←(Neg，H)，job$_e$←(DS，{subjob})，L←$L\cup${job$_e$}。

4.2 基于 MapReduce 的 SWRL 规则分布式推理算法

以推理计划模型 L 和 HBase 中 OWL 本体数据为输入，分布式规则推理模块实现基于 MapReduce 的 SWRL 规则分布式推理算法。该算法由 5 个函数构成：Main、DB_Map、File_Map、Reduce 和 Result_Map。其中，在 Hadoop 平台的 Master 节点中运行的 Main 函数负责推理任务的生成和调度，运行于各个计算节点的 DB_Map、File_Map、Reduce 和 Result_Map 函数执行具体的推理计算。上述函数的算法描述如下

Main：输入推理计划模型 L，执行

1)对 L 中所有规则原子解析模型 A，当 $A(p)$ 为 rdf:type 时，对 $A(o)=C$，从 HBase 数据库 T_SP_O 表中查询出以 C 为根的子类层次结构树 T，并对 T 按深度优先策略排列为{C，C_1，…，C_h}，($h\geqslant 0$)，设 $A_1=(A(s),\ A(p),\ C_1)$，…，$A_h=(A(s),\ A(s),\ C_h)$，执行 A←{A，A_1，…，A_h}；当 $A(p)$ 为 OWL 属性 P 时，从 HBase 数据库 T_SP_O 表中查询出以 P 为根的子属性层次结构树 T'，并对 T' 按深度优先策略排列为{P，P_1，…，P_l}，($l\geqslant 0$)，设 $A_1=(A(s),\ P_1,\ A(o))$，…，$A_l=(A(s),\ P_l,\ A(o))$，执行 A←{A，A_1，…，A_l}。

2)计算 ln←length(L)，ln 为 L 中模型 job 个数，设 HDFS 文件夹 JoinOut、TempOut、和 DcarIn，设分布式缓存 DC。

3)对 $i=1$，…，($ln-1$)，job$_i\in L$，执行 DC←job$_i$，当 job$_i$ 标识符 DS 为 F 时，设置 File_Map 数据源为 TempOut，执行 File_Map 和 Reduce 函数；当 job$_i$ 标识符 DS 为 D 时，计算 sn←length(job$_i$)，sn 为 job$_i$ 中 subjob 个数，对 $j=1$，…，sn，计算 an←length(subjob$_j$)，an 为 subjob$_j$ 中规则原子解析模型个数，对 $x=1$，…，an，$A_x\in$subjob$_j$，生成 A_x 对应的数据表 Table$_x$ 及查询命令 Comd$_x$，执行 Q←$Q\cup$ {(Table$_x$，Comd$_x$)}。设 DB_Map 数据源为 Q，当 subjob$_j$ 的 Tag 值为任一共同变量时，执行DB_Map 和 DB_Reduce 函数，当 subjob$_j$ 的 Tag 值为 Neg 时，仅执行 DB_Map 函数。

4)对 $i=ln$，设 Result_Map 函数数据源为 HDFS 文件夹 JoinOut 和 DcarIn，并执行 Result_Map 函数。

DB_Map：根据集合 Q，检索得到规则原子解析模型在本体库中的解释数据集 DL={data$_1$，…，data$_n$}，其中 data 为 OWL 个体或常量的 a 或二元组(a，b)，执行

1)当 $n=0$ 时，终止计算；当 $n\geqslant 1$ 时，读取 DC 中 job 模型，对 $i=1$，…，n，data$_i\in DL$，根据 data$_i$ 对应的 Table 和 Comd，取 data$_i$ 在 job 中对应模型 A_i 和 subjob$_i$，当 data$_i=a$ 时，执行 $I_i(A_i)$←(v，a)，当 data$_i=(a,\ b)$ 时，执行 $I_i(A_i)$←((v_1，a)，(v_2，b))，当 subjob$_i$ 标识符 Tag 为共同变量名 v 时，执行 IKV$_i$←<(v，$I_i[v]$)，(A_i，$I_i(A_i)$)>，当 subjob$_i$ 标识符 Tag 为 Neg 时，执行 IKV$_i$←<(v，$I_i(A_i)$)，(A_i，$I_i(A_i)$)。

2)对 $i=1$，…，n，当 subjob$_i$ 标识符 Tag 为共同变量名 v 时，将 IKV$_i$ 传递给 Reduce 函数；当 subjob$_i$ 标识符 Tag 为 Neg 时，将 IKV$_i$ 输出到 DcarIn 文件夹中 $A_{i,\text{out}}$ 文件。

File_Map：输入 TempOut 中 IKV 模型的集合 KL={IKV$_1$，…，IKV$_n$}，执行

1)若 $n\geqslant 1$，读取 DC 中 4 模型，计算 sl←length(job)，sl 为 job 模型中 subjob 个数。

2)对 $i=1$，…，n，若 IKV$_i$ 键中存在变量 $v\in$subjob$_k$，($1\leqslant k\leqslant sl$)，将 IKV$_i$ 传递给 Reduce 函数。

Reduce：输入具有相同键的推理中间数据模型 IKV$_1$，…，IKV$_u$，执行

1)若 $u\geqslant 1$，读取 DC 中模型 job，根据 IKV$_1$，…，IKV$_u$ 键中变量名 v，取 job 中对应 subjob 模型，计算 cnt←length(subjob)，cnt 为 subjob 中规则原子解析模型个数，设集合 E_1，…，E_{cnt}，分别用于存放 A_1，…，A_{cnt} 的解释模型。

2)对 $i=1$，…，u，取 IKV$_i$ 中解释模型 $I(\{A_i\})$ 的变量 v 或(v_1，v_2)，当 v 或(v_1，v_2)$\in A_j$ 时，执行 E_j←$E_j\cup\{I(\{A_i\})\}$，($1\leqslant j\leqslant$cnt)。

3)计算 $E_1\bowtie E_2\bowtie\cdots\bowtie E_{\text{cnt}}$，得同时满足 subjob 中各规则原子解析模型的解释模型 $I_1(E_1$，…，$E_{\text{cnt}})$，…，$I_r(E_1$，…，$E_{\text{cnt}})$。当 $r=0$ 时，终止运行 Reduce 函数；当 $r\geqslant 1$ 时，对 $k=1$，…，r，若 $I_k(E_1$，…，$E_{\text{count}})$ 中存在仍未进行连接操作的变量 v_k，构造推理中间数据模型 IKV$_k$=<(v_k，$I(v_k)$)，({A_1，…，A_{cnt}}，$I_k(E_1$，…，$E_{\text{cnt}})$)>，当 job 标识符 DS 为 D 时，输出 IKV$_k$ 到 TempOut；当 job 标识符 DS 为

http://qks.cqu.edu.cn

F 且 $I_k(E_1,\cdots,E_{cnt})$中不存在仍未进行连接操作的变量时,将 $I_k(E_1,\cdots,E_{cnt})$输出到 JoinOut。

Result_Map:输入 JoinOut 和 DcarIn 文件夹中解释模型,执行

1)设集合 Result,从 JoinOut 中读取解释模型集合 $JI=\{I_1,\cdots,I_n\}$。从 DcarIn 各个文件中读取解释模型集合 $DI_1,\cdots,DI_m$,若 $n\geqslant 1$ 且 $m\geqslant 1$,计算 Result$\leftarrow JI\times DI_1\times\cdots\times DI_m$;若 $m=0$ 且 $n\geqslant 1$,执行 Result$\leftarrow JI$;若 $n=0$ 且 $m\geqslant 1$,计算 Result$\leftarrow DI_1\times\cdots\times DI_m$。

2)读取 DC 中模型 job,取 job 中后件规则原子解析模型 A'。根据 A'中变量,从 Result 解释模型中取得对应的变量值,得到推理结果 $I(A')$,并输出到 HDFS 文件 ResultOut。

5　实　验

由于目前没有专用于 SWRL 规则引擎性能测试的 SWRL 规则集和 OWL 本体数据集,采用被学术界广泛应用于本体推理和查询性能测试的 LUBM 本体数据集[21],并根据 LUBM 的术语本体,设计了如表 3 所示的测试规则。为验证 CloudSWRL 框架的推理性能,进行了 CloudSWRL 与 Jess 和 Pellet 推理引擎的对比实验,以及通过递增计算节点来验证 CloudSWRL 框架可扩展性的实验。

表 3　测试 SWRL 规则

ID	测试规则
R_1	Professor(? x) ∧ Department(? y) ∧ works-For(? x, ? y) ∧ TeachingAssistant(? z) ∧ advisor(? z, ? x)→worksFor(? z, ? y)
R_2	ResearchAssistant(? x) ∧ GraduateCourse(? y) ∧ takesCourse(? x, ? y)→GraduateStudent(? x)

5.1　对比实验

在对比实验中,使用 LUBM 工具分别了生成了如表 4 所示的本体数据集。将 Jess 71p2 和 Pellet 2.3.0 推理引擎部署于配置为双核 64 位 E7200 CPU, DDR2 6 GB 内存,4TB 硬盘存储空间,安装 64 位 Windows 操作系统的计算机之上,并为 Java 虚拟机分配 5 GB 内存空间。同时,将 CloudSWRL 框架及 Hadoop 0.20.2 平台 Master 节点部署于 Pentium 4 CPU、1.5 GB 内存、320 GB 硬盘空间,并安装 Ubuntu10.10 操作系统计算机上,设置了 8 台分布式计算和 HBase 数据存储节点,每台节点配置为 Pentium 4 CPU、1.5 GB 内存和 80 GB 硬盘空间,分配 Java 虚拟机内存 512 MB。Master 节点及八台计算节点通过 100 Mb/s 的局域网互连。通过分析对比实验推理输出,对测试规则 R_1 和 R_2,各测试推理工具均生成相同推理结果。统计由本体加载用时、规则加载用时、推理用时和输出用时组成的计算总用时,如表 5 和表 6 所示。

表 4　LUBM 测试本体数据集

ID	OWL 文件数	数据集大小/MB
D_1	90	50
D_2	140	75
D_3	190	100
D_4	240	125
D_5	290	150

表 5　测试规则 R_1 对比实验结果

ID	Jess/s	Pellet/s	CloudSWRL/s
D_1	83.36	45.46	65.54
D_2	121.87	68.05	67.33
D_3	197.48	91.49	69.59
D_4	376.92	116.56	71.34
D_5	内存溢出	148.85	74.19

表 6　测试规则 R_2 对比实验结果

ID	Jess/s	Pellet/s	CloudSWRL/s
D_1	90.21	69.53	66.26
D_2	326.67	76.76	69.12
D_3	633.59	99.22	75.07
D_4	内存溢出	152.92	78.86
D_5	内存溢出	222.52	79.77

对比实验结果表明,在单机环境和分布式环境的硬件配置总体相当的前提下,当输入本体数据量较小时,基于 Tableau 算法的 Pellet 推理引擎具有较好的计算性能,而 Jess 规则引擎由于需要进行本体和规则的映射,在占用大量内存空间的同时推理性能与 Jess 和 CloudSWRL 存在差距。但随着输入本体数据量不断增加,CloudSWRL 框架较 Jess 和 Pellet 推理引擎在处理大规模本体数据时,计算性能上存在明显优势。

5.2　框架可扩展性实验

可扩展性实验以 1 GB 的 LUBM 本体数据集及

http://qks.cqu.edu.cn

R_1 和 R_2 测试规则作为输入，实验硬件环境与5.1节中对比实验相同。计算节点数分别设置为2、4、6和8台，其实验结果如图4所示，其中横轴表示计算节点数，纵轴表示推理总用时，单位为S。

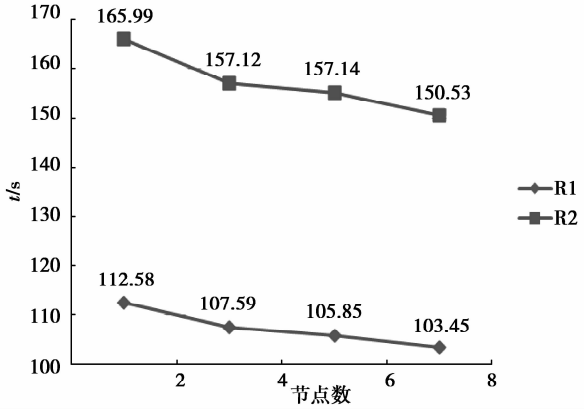

图4 CloudSWRL框架可扩展性实验结果

框架可扩展性实验结果表明，对相同输入数据，当云计算环境中计算节点数逐步递增时，框架具备更强的计算能力，推理用时逐步降低。即当需要处理海量本体输入数据时，可通过扩展计算节点数来获得更佳的推理性能。

综合对比实验和可扩展性实验结果，不难得出，CloudSWRL框架在对大规模本体数据进行SWRL规则推理时，在计算性能和可扩展性方面优于传统SWRL推理工具。

6 结 论

由于传统单机环境下运行的SWRL推理工具在计算性能和扩展性等方面存在瓶颈，提出了云计算环境下的SWRL规则分布式推理框架CloudSWRL。以Hadoop平台的MapReduce编程模型和Hbase分布式数据库为基础，设计了OWL本体数据在HBase中的存储策略，定义了框架中相关解析模型，提出了基于MapReduce的推理计划生成算法和在DL-safe限制下SWRL规则分布式推理算法。对比实验和可扩展性实验结果表明，CloudSWRL框架较传统的Jess和Pellet推理引擎在处理大规模语义Web本体的SWRL规则推理时，具备更好的计算性能和可扩展性。

在后续研究工作中，除了进一步对CloudSWRL框架推理性能进行优化以外，还将引入SWRL规则sameAs、differentFrom和内置函数，以实现对更复杂SWRL规则推理的支持。

参考文献：

[1] W3C. Resource description framework [EB/OL]. [2011-11-30]. http://www.w3.org/RDF/.

[2] Grau B C, Horrocks I, Motik B, et al. OWL 2: the next step for owl [J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2008, 6(4):309-332.

[3] Horrocks I, Patel-Schneider P F, Boley H, et al. SWRL: a semantic web rule language combining OWL and RuleML [EB/OL]. [2011-11-30]. http://www.w3.org/Submission/SWRL/.

[4] Horrocks I, Patel-Schneider P F, Bechhofer S, et al. OWL rules: a proposal and prototype implementation[J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2005, 3(1): 23-40.

[5] ProtégéWiki. SWRLLanguageFAQ [EB/OL]. [2011-11-30]. http://protege.cim3.net/cgi-bin/wiki.pl?SWRLLanguageFAQ.

[6] Motik B, Sattler U, Studer R. Query answering for OWL-DL with rules [J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2005, 3(1): 41-60.

[7] Husain M F, Khan L, Kantarcioglu M, et al. Data intensive query processing for large RDF graphs using cloud computing tools [C]// Proceeding of the 2010 IEEE 3rd International Conference on Cloud Computing, July 5-10, 2010, Miami, Florida. Piscataway: IEEE Press, 2010:1-10.

[8] 杜小勇，王琰，吕彬. 语义Web数据管理研究进展[J]. 软件学报，2009，20(11):2950-2964.
DU Xiaoyong, WANG Yan, Lü Bin. Research and development on semantic web data management [J]. Journal of Software, 2009, 20(11): 2950-2964.

[9] 冯登国，张敏，张妍，等. 云计算安全研究[J]. 软件学报，2011，22(1):71-83.
FENG Dengguo, ZHANG Min, ZHANG Yan, et al. Study on cloud computing security [J]. Journal of Software, 2011, 22(1):71-83.

[10] Ghemawat S, Gobioff H, Leung S T. The google file system[C]// Proceedings of the 19th ACM Symposium on Operating Systems Principles, October 19-22, 2003, Bolton Landing, USA. New York, USA: ACM, 2003: 29-43.

（下转第84页）

http://qks.cqu.edu.cn

an integrated starter generator parallel hybrid electric car [J]. Journal of Chongqing University, 2008, 31(5): 499-504.

[7] 朱道伟，谢晖，严英，等. 基于道路工况自学习的混合动力城市客车控制策略动态优化[J]. 机械工程学报，2010，46(6)：33-38.
ZHU Daowei, XIE Hui, YAN Ying, et al. Control strategy dynamic optimization of the hybrid electric bus based on driving cycle self-learning[J]. Chinese Journal of Mechanical Engineering, 2010, 46(6): 33-38.

[8] Kheir N A, Salman M A, SchoutenN J. Emissions and fuel economy trade-off for hybrid vehicles using fuzzy logic[J]. Mathematics and Computer in Simulation, 2004, 66(2/3): 155-172.

[9] Baumann B M, Washington G, Glenn B C, et al. Mechatronic design and control of hybrid electric vehicles [J]. IEEE/ASME Transactions on Mechatronics, 2000, 5(1): 58-72.

[10] Ehsani M, Gao Y M, Butler K L. Application of electrically peaking hybrid (ELPH) propulsion system to a full-size passenger car with simulated design verification [J]. IEEE Transactions on Vehicular Technology, 1999, 48(6): 1779-1787.

[11] Zimmermann H J. Fuzzy programming and linear programming with several objective functions [J]. Fuzzy Sets and Systems, 1978, 1(1): 45-55.

[12] Li S Y, Hu C F. Two-step interactive satisfactory method for fuzzy multiple objective optimization with preemptive priorities[J]. IEEE Transactions on Fuzzy Systems, 2007, 15(3): 417-425.

[13] Banvait H, Anwar S, Chen Y B. A rule-based energy management strategy for plug-in hybrid electric vehicle (PHEV)[C]// Proceedings of the 2009 Conference on American Control, June 10-12, 2009, St. Louis, MO. Piscataway: IEEE Press, 2009: 3938-3943.

[14] Sun H, Yang L F, Jing J Q, et al. Control strategy of hydraulic/electric synergy system in heavy hybrid vehicles [J]. Energy Conversion and Management, 2011, 52(1): 668-674.

[15] Pisu P, Rizzoni G. A comparative study of supervisory control strategies for hybrid electric vehicles[J]. IEEE Transactions on Control Systems Technology, 2007, 15(3): 506-518.

[16] 戴一凡，罗禹贡，边明远，等. 一种新型强混合动力结构的控制策略[J]. 汽车工程，2009(10)：919-923,951.
DAI Yifan, LUO Yugong, BIAN Mingyuan, et al. The control strategy for a new full hybrid powertrain structure[J]. Automotive Engineering, 2009(10): 919-923,951.

（编辑 张小强）

（上接第 62 页）

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

[12] Chang F, Dean J, Ghemawat S, et al. BigTable: a distributed storage system for structured data[J]. ACM Transactions on Computer Systems, 2008, 26(2): 1-14.

[13] Apache Software Foundation. Hadoop [EB/OL]. [2011-11-30]. http://hadoop.apache.org/.

[14] 王鹏，孟丹，詹剑锋，等. 数据密集型计算编程模型研究进展[J]. 计算机研究与发展，2010，47(11)：1993-2002.
WANG Peng, MENG Dan, ZHANG Jianfeng, et al. Review of programming models for data-intensive computing [J]. Journal of Computer Research and Development, 2010, 47(11): 1993-2002.

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

[16] Husain M, McGlothlin J, Masud M M, et al. Heuristics-based query processing for large RDF graphs using cloud computing [J]. IEEE Transactions on Knowledge and Data Engineering, 2011, 23(9): 1312-1327.

[17] 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, 2011, 10: 59-75.

[18] Franke C, Morin S, Chebotko A, et al. Distributed semantic web data management in HBase and MySQL cluster [C] // Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing, July 4-9, 2011, 2011: 105-112.

[19] Apache Software Foundation. Apache jena [EB/ OL]. [2011-11-30]. http://jena.apache.org/.

[20] Sun J L, Jin Q. Scalable RDF store based on HBase and mapreduce [C] // Processings of the 2010 3rd International Conference on Advanced Computer Theory and Engineering, August 20-22, 2010, Chengdu, China. Piscataway: IEEE Press, 2010,1: 633-636.

[21] Guo Y B, Pan Z X, Heflin J. LUBM: a benchmark for OWL knowledge base systems [J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2005, 3(2/3):158-182.

（编辑 侯 湘）