Использование протокола LDAP для поддержки распределенности гетерогенных информационных систем

А.Н. Бездушный bezdushn@ccas.ru
Д. А. Ковалев mit@openldap.org
А.А. Филиппова filipova@ccas.ru

Аннотация

В статье изучаются и предлагаются технологии обеспечения распределенного функционирования в системах, интегрирующих разнородные электронные библиотеки. Предлагаемые решения иллюстрируются и апробируются в рамках проекта ИСИР РАН, краткое описание которого также приводится.

KEYWORDS

поиск в распределенной гетерогенной информационной системе, LDAP, query routing, forward knowledge, centroids

ВВЕДЕНИЕ

На сегодняшний день накоплены большие объемы цифровой информации, и на первое место выходят задачи организации ее интеграции и эффективного использования. Одним из основных направлений здесь является проблема интеграции электронных библиотек, библиотечных систем, баз данных научных и других организаций - то есть хранилищ *долгоживущих* информационных ресурсов, качество и актуальность которых находится под контролем. Именно такие ресурсы (научные публикации, стандарты, экспериментальные и другие знания) представляют наибольший интерес с точки зрения их интеграции и эффективного использования. Важно, что речь идет не о попытке создания распределенной ИС, которая предоставляла бы качественный поиск по заведомо некачественной информации, каковая в основном имеется в сегодняшнем Интернете (по большей части - домашние страницы частных лиц или имиджевые сайты компаний, и не так много качественно структурированной информации), а о средстве интеграции существующих специализированных информационных систем и методах создания новых.

На текущий момент многие системы имеют шлюзы для публичного доступа в Интернет, что позволяет говорить об их доступности в целом, однако сводит ситуацию к индексированию HTML-страниц. Теряются основные преимущества - качество, возможность структурированного поиска и доступа, также не решается проблема интеграции (например, единого механизма поиска). Обычные поисковые машины и каталоги интернет-ресурсов страдают одним из двух недостатков - либо, при хорошем качестве, они имеют ограниченные темпы обновления из-за большого процента человеческого участия (как Yahoo), либо, за счет использования роботов, не могут гарантировать качество индексируемой информации.

Как при интеграции существующих систем, так и при проектировании новых (распределенных), в первую очередь возникает круг проблем связанный с объемом, географической распределенностью информации, разными административными принципами управления в каждой из участвующих организаций.

Эти проблемы учтены в архитектурах, подобных DNS, где выделяется иерархия зон ответственности за информацию. В пределах каждой "подзоны"задача по структурированию и дальнейшему сопровождению информации по технологиям обычного каталога (библиотеки) становится обозримой, в сумме же эти зоны дают легко масштабируемую распределенную коллекцию информации. Например, зона может соответствовать библиотеке, которая может выделять подзоны своим филиалам.

В системах с такой распределенной архитектурой на первое место выходит уже не проблема качества и объема хранимой информации, а проблема организации эффективного поиска. Поверхностные решения, как то рассылка запроса по всей иерархии узлов и суммирование ответа, не позволяют рассчитывать на приемлемое время отклика (потенциально огромное количество узлов, а также неизвестные условия связи).

Решение, найденное в этой ситуации и примененное в системах типа WHOIS++, Harvest [1, 2, 3, 5, 34, 35], состоит в сужении множества узлов-получателей поискового запроса на основе предварительной (индексной) информации. В случае иерархии узлов можно говорить о маршрутизации поисковых запросов (каждый из узлов, получивших запрос, в свою очередь, использует индек-

сы с соседних узлов для принятия решения о дальнейшем распространении запроса). Компактная форма представления индексов (центроиды или SOIF-записи и т.п.), а также другие аспекты этой технологии (структура и управление потоками индексов), более подробно освещаются далее в работе (также см. [1, 2, 3, 5, 6, 7, 8, 9, 34, 35]), здесь же следует упомянуть о возникающих ограничениях.

Проблемы сохранения качества информации при интеграции в распределенную систему.

Все информационные системы так или иначе оперируют (мета)данными - характеристиками, описывающими свойства информационного ресурса и позволяющими его идентифицировать. Информационный ресурс характеризуется набором атрибутов - свойств данного объекта - например, публикация (название, ключевые слова, рубрикатор, ФИО авторов и т.п.). Более подробное и строгое определение информационного ресурса см. [37].

Поиск осуществляется с помощью задания условий на значения этих полей у искомых объектов. Основная потеря качества в *гетерогенных* распределенных системах состоит в отсутствии поддержки связей между ресурсами. Причем нужно сразу заметить, что речь не идет о технологиях выражения связей между отдельными объектами, или даже их ограниченными совокупностями (таких технологий достаточно - например, CORBA [36]). Речь идет о методах поиска связанных между собой объектов среди потенциально огромной, распределенной коллекции. Также важно, что рассматриваются именно *гетерогенные* системы, т.е системы, объединяющие существенно разнородные узлы (как в смысле платформ, конкретного ПО, так и в смысле технологий), т.к. распределенные информационные системы с однородными узлами известны и реализованы всеми ведущими поставщиками СУБД.

По сути, атрибуты типа ФИО автора не описывают собственно ресурс (публикацию), а *приближенно* выражают его связь с другим ресурсом "персона", для которого метаинформацией являются атрибуты типа "ФИО", "контактная информация"и т.п.

В утрированной форме, это означает что, например, заинтересовавшись некоторой публикацией, мы не можем перейти непосредственно на информацию об ее авторе, а лишь инициировать поиск ресурсов типа "персона"с ФИО, совпадающим с полем "ФИО автора"этой публикации.

Такую проблему относительно легко решить, введя уникальную идентификацию для каждого ресурса. Действительно, в каждой организации, участвующей в системе, в том или ином виде решается задача идентификации собственных ресурсов (номера, классификаторы и проч.); для вновь создаваемых узлов может существовать рекомендуемая система идентификации. Каждой из организаций соответствует, как предлагалось выше, "зона ответственности"за информацию. В пределах этой зоны организация отвечает не только за актуальность и полноту ресурсов, но и за их идентификацию. Выделяя подзоны, она также следит за уникальностью их названий. Последовательно дописывая названия объемлющих зон к идентификатору объекта, мы получим идентификатор, уникальный в пределах всей распределенной системы. Далее, этот уникальный идентификатор можно использовать в поисковых запросах наравне с другими атрибутами, что позволит избежать проблемы дублирования. В нашем примере, мы можем в качестве атрибута публикации хранить не ФИО автора, а идентификатор соответствующего объекта.

Однако, если попытаться ввести в поисковый язык возможность задавать условия на связи, то в чистом виде технология центроидов и суммирования ответов окажется уже недостаточной. Например, удобно было бы иметь возможность найти документы, удовлетворяющие определенному набору условий на атрибуты (например, тематика), авторами которых являются люди с определенными свойствами. Условия на атрибуты объектов каждого типа дают два множества ресурсов, распределенных по многим узлам, пары этих ресурсов нужно проверить на связанность. Потенциальный объем пересылок, которые понадобятся для этого, не позволит надеяться на приемлемое время отклика в условиях сильной распределенности и использования каналов связи Интернет.

Но поисковые запросы обычно затрагивают далеко не все атрибуты информационного ресурса, а лишь их (сравнительно небольшую по объему) часть - назовем ее *поисковой* метаинформацией. Вместо того, чтобы пересылать поисковую метаинформацию каждый раз при исполнении запроса, можно реплицировать ее на постоянной основе, так, чтобы на каждом узле поддерживалась копия поисковой информации для всех ресурсов, непосредственно связанных с его собственными. Это позволит в большинстве случаев избежать пересылки данных в процессе выполнения запросов.

Конечно, несмотря на сравнительно небольшой (по отношению к общему) объем реплицируемых данных, такая технология возможна только при разумном количестве связей. Поэтому в целом система должна комбинировать технологии репликации поисковых данных и суммирования ответов, в случае каждого конкретного узла принимая во внимания баланс между качеством связи, объемом информации и т.п. Некоторые мощные узлы могут вообще полностью реплицировать метаданные с подчиненных, и отвечать на поисковые запросы к поддереву самостоятельно. В других же случаях, когда мощностей не хватает даже для поддержки метаданных связанных ресурсов, можно прибегнуть к пересылке во время выполнения.

Кроме того, для повышения эффективности можно использовать планирование запросов на основе метаинформации другого рода - статической или накапливаемой информации о качестве связи, скорости ответа от разных узлов, их доступности и т.п.

Естественно, что реально масштабные информационные системы не могут создаваться в рамках единственной инициативы, а только лишь за счет взаимной интеграции множества существующих и заново создаваемых систем и узлов. Поэтому технологии, претендующие на возможность межсистемной интеграции, должны предусматривать множество разноуровневых средств, начиная от непосредственного импорта всех поисковых данных, и кончая организацией более или менее полноценного взаимодействия. Эти средства должны включать возмож-

ность не только преобразования форматов и протоколов связи, но также и возможность установления взаимного соответствия рубрикаторов, моделей данных (в разных системах набор ресурсов и их атрибутов может быть разным).

Именно разнородностью узлов разрабатываемая технология существенно усложняется по сравнению, например, с распределенными реляционными СУБД, в которых подобные технологии минимизации вычислительных затрат при распределенном поиске уже применяются. Данными обмениваются узлы (системы), интегрируемые на гораздо более высоком уровне абстракции.

Кроме прочего, это означает необходимость по возможности использовать открытые технологии, стандарты, протоколы, потенциально облегчая будущую интеграцию и учитывая также доступность тех или иных технологий.

Именно эти соображения учитывались при выборе конкретных механизмов решения задачи распределенного функционирования.

Система ИСИР РАН

Технологии, описываемые в этой статье, изучаются и реализуются в рамках проекта ИСИР РАН (Интегрированная система информационных ресурсов РАН).

Основной целью проекта ИСИР является разработка концептуальной основы и инфраструктуры для интеграции разнородных информационных и вычислительных ресурсов в единое информационное пространство. Этот базис должен обеспечить объединение в единое пространство всевозможных цифровых библиотек, информационных и вычислительных систем, использующих как собственные принципы организации, так и технологию открытой архитектуры ИСИР или непосредственно ее ПО (в первую очередь речь идет об информационных ресурсах РАН). Гибкая организация информации, ее интегрированное представление, открытая архитектура системы являются ключевыми моментами в проектировании ИСИР.

Более полная информация по ИСИР РАН содержится в [37], в этой же статье рассматривается один из ключевых аспектов архитектуры системы - поддержка распределенного функционирования.

Далее приведено краткое описание протокола LDAP, дающее основу для изложения предлагаемых решений на его основе.

LDAP - КРАТКОЕ ОПИСАНИЕ

Протокол LDAP, возникший как сравнительно легковесный протокол доступа к сервисам X.500 [12, 13, 14, 15] (и поэтому изначально представлявший сильно упрощенную модель X.500 - см. [16]), позднее получил развитие и по сей день развивается как самостоятельный набор стандартов, по существу определяющий модель данных, применимые операции и протокол удаленного доступа к иерархической объектной БД. Строгое описание протокола см. в [17], здесь же мы приводим лишь его краткое описание.

Модель данных. Схема

Протокол LDAP предполагает, что есть один или больше серверов, которые совместно обеспечивают доступ к "информационному дереву директории"(Directory Information Tree - DIT). В узлах дерева находятся *записи*.

Основа информационной модели LDAP - запись (entry), содержащая информацию о некотором объекте (например, о персоне). Записи состоят из значений атрибутов, которые имеют тип. Тип атрибута задает метод хранения (текстовый/бинарный), текстовое описание семантики атрибутов данного типа, правила сравнения (matching rules) двух значений атрибутов данного типа, и т.д.

Определяется также иерархия классов узлов (objectclasses). Класс узла определяет набор обязательных и возможных типов атрибутов для узлов, принадлежащих к этому классу, а также описание семантики таких узлов и предполагаемых способов использования соответствующих данных. Кроме того, можно указать, может ли атрибут данного типа иметь в узле данного класса много значений, или значение должно быть одно.

Иерархия классов имеет смысл наследования - новый класс может быть создан на основе описания уже существующего класса, наследуя в дополнение к собственным атрибутам обязательные атрибуты предка (без изменений), и имея возможность изменить статус унаследованных необязательных атрибутов. Узел может принадлежать к нескольким классам, при этом наборы обязательных и необязательных атрибутов сливаются естественным образом (т.е. соответствующие множества объединяются, если в одном из классов некоторый тип атрибута указан как необязательный, а в другом - как обязательный, для узла наличие такого атрибута обязательно). Принадлежность к тому или иному классу определяется наличием соответствующего значения (имени класса) у специального атрибута objectclass, каковой должен присутствовать в каждом узле.

Вместе набор определений attribute types и objectclasses дает так называемую *схему*, дающую представление об отображении предметной области в модель данных LDAP, и одновременно обеспечивающую возможность довольно жесткого автоматического контроля целостности (на наличие необходимых атрибутов, синтаксис значений и пр.).

Идентификация

Схема именования записей в LDAP очень похожа на много раз упоминавшуюся ранее доменную модель. Рассмотрим множество непосредственных потомков любого отдельно взятого узла. Каждая из таких записей имеет уникальное относительно своих соседей имя (Relative Distinguished Name - RDN), состоящее из одной или нескольких пар «имя атрибута[1] >=<значение>"записи (напомним, что запись LDAP, собствен-

[1]Точнее, имя *типа* атрибута. Как ранее указывалось, имена имеют именно типы атрибута, и описание класса содержит именно указание, что в узле могут содержаться атрибуты данного типа, а не "название"атрибута в составе узла.

но, и состоит из множества таких пар). Самое большее, одно значение каждого атрибута может использоваться для формирования RDN. Уникальность RDN среди непосредственных потомков контролируется автоматически. Примеры возможных RDN для записи о персоне, состоящей из атрибутов типа "имя"(shortName, sn), "фамилия"(commonName, cn), "инн"(userID, uid): "uid=241541245"или "cn=Иванов+sn=Иван".

Далее, полный идентификатор записи в информационном дереве в целом (Distinguished Name - DN) получается последовательным приписыванием к ее RDN относительных имен всех узлов вверх по дереву до корня. По традициям X.500 ранее организация узлов в дерево на верхнем уровне производилась по геополитическому принципу (страна -> организация -> отдел -> данные), и в нашем примере запись о Иване Иванове была бы потомком записи об отделе, где он работал, та, в свою очередь - потомком записи об организации, далее - запись о стране. Поэтому DN этой записи имел бы вид "cn=Иванов,ou=отдел X,o=ВЦ РАН,c=RU". Подробнее о литеральном представлении DN и принципах именования см. [18, 19, 20].

Каждая организация занимается поддержкой информации в поддереве соответствующей записи, и мы имеем для любой записи уникальный в пределах всего дерева идентификатор.

Распределенность

Предполагается, что информационное дерево, о котором идет речь, может поддерживаться несколькими распределенными по разным машинам серверами. Каждый из серверов поддерживает некоторое поддерево (например, в каждой организации есть собственный сервер, поддерживающий соответствующее поддерево, начиная с некоторого узла этого поддерева может начинаться компетенция сервера подчиненной организации и т.д.), и отвечает на запросы к данным в этом поддереве.

В ситуации, когда полученный запрос относится к данным вне компетенции данного сервера, предусмотрены две возможности: либо на основе имеющейся у сервера информации сделать запрос к нужному серверу и вернуть результат прозрачно для клиента, либо вернуть ему ссылку на нужный сервер (в последнем случае прозрачность распределенного характера операции может быть достигнута на уровне клиентского LDAP API - см. [21, 25]). На данный момент имеется стандарт de-facto (находящийся на пути к официальному документу - см. [26]), определяющий способ представления таковой информации (достаточной для переадресации запроса к соответствующему серверу) в виде LDAP-записей в составе собственно информационного дерева.

Вкратце этот способ состоит в следующем. Если объект с запрашиваемым DN отсутствует в БД сервера, но среди его предков есть объекты класса "referral", то содержимое атрибута "ref"объекта, наиболее близкого к искомому, считается ссылкой на нужный сервер. Если при поиске объект, удовлетворяющий условиям, имеет среди прочих класс "referral", то ссылки, содержащиеся в его атрибуте "ref", обрабатываются соответствующим образом. Подробнее см. [26].

Кроме referrals, имеется еще один специализированный класс - alias, имеющий смысл, аналогичный soft link в файловой системе Unix (объект-ссылка, на место которого в большинстве случаев прозрачно подставляется объект, с DN из атрибута aliasedObjectName). Использование таких объектов-ссылок в сочетании с referrals, плюс механизм их прозрачной обработки на стороне сервера или клиентского API, дает возможность представить клиенту работу с распределенным деревом как с единым целым.

Протокол удаленного доступа определен, в основном, для случая постоянного соединения (хотя есть и спецификация для Connectionless LDAP - CLDAP, см. RFC 1798), однако все операции могут совершаться асинхронно, предполагают возможность разрыва связи, задание ограничений на время выполнения запросов, размер результата и т.д. Протокол рассчитан на использование в Интернет (имеются в виду характеристики типичных линий связи и стек TCP/IP). Подробнее об этом см. [17, 21, 25].

Модель взаимодействия, краткое описание основных операций

Протокол определяет следующую модель взаимодействия между клиентом и сервером:

1. Открытие соединения с LDAP сервером. Выполнение аутентификации к LDAP серверу (операция **Bind**)

2. Выполнение некоторой серии операций LDAP и получение результатов (операции **Search, Modify, Add, Delete, ModifyRDN, CompareDN,Abandon**)

3.Закрытие соединения (операция **Unbind**).

За полным описанием протокола и операций необходимо обратиться к [17], здесь же мы кратко приводим ключевые моменты, уточняя их далее по мере необходимости.

Операции добавления, модификации и удаления применяются к одному узлу за один вызов. Добавление и модификация могут работать на любом уровне, вплоть до отдельных значений атрибутов, т.е. можно оперировать как объектами целиком, так и изменять состав атрибутов отдельного узла (в пределах ограничений схемы), и отдельных значений каждого атрибута, причем модификации атрибутов и значений можно проводить сериями за один запрос для данной записи. Контроль схемы производится автоматически, если какое-то из серии запрошенных изменений противоречит схеме, то откатывается вся операция над данным объектом.

Операция поиска предусматривает три режима (области действия - scope) - BASE, ONELEVEL, SUBTREE. Кроме режима, в качестве параметров операции передаются: базовый узел (search base DN), поисковый фильтр (комбинация условий на атрибуты искомых узлов - описание см. ниже), лимиты времени и размера и т.д.

Режим BASE, по существу, осуществляет загрузку записи с заданным DN. В режимах ONELEVEL и SUBTREE базовый DN интерпретируется как начальная вершина поиска, в первом случае поиск ограничивается непосредственными потомками данной вершины, во втором - всем ее поддеревом.

Необходимо отметить, что предусмотрена возможность указывать какие именно атрибуты загружать (в случае, если интересуют не полные записи, а лишь их конкретные атрибуты).

Поисковый язык (фильтры)

Поисковый язык LDAP позволяет задать комбинацию условий на значения атрибутов, имея в распоряжении, кроме логических, следующие операции: сравнение на равенство, приблизительное равенство (семантика определяется типом атрибута), вхождение подстроки (в начале, в конце или в любом месте строки). Формальные описания фильтров и их строкового представления даны в [17, 22], здесь же мы дадим лишь компактное описание и примеры.

Строковое представление описывается следующей БНФ:

<filter> ::= '(' <filtercomp> ')'
<filtercomp> ::= <and> | <or> | <not> | <item>
<and> ::= '&' <filterlist>
<or> ::= '|' <filterlist>
<not> ::= '!' <filter>
<filterlist> ::= <filter> | <filter> <filterlist>
<item> ::= <simple> | <present> | <substring>
<simple> ::= <attr> <filtertype> <value>
<filtertype> ::= <equal> | <approx> | <greater> | <less>
<equal> ::= '='
<approx> ::= '~='
<greater> ::= '>='
<less> ::= '<='
<present> ::= <attr> '=*'
<substring> ::= <attr> '=' <initial> <any> <final>
<initial> ::= NULL | <value>
<any> ::= '*' <starval>
<starval> ::= NULL | <value> '*' <starval>
<final> ::= NULL | <value>

<attr> - строковое представление AttributeType (формат описан в [23]).

<value> - строковое представление AttributeValue, либо его части (формат описан в [17])

Если <value> должно содержать '*', или '(', или ')', то эти символы следует предварять символом '\'.

Дадим несколько примеров фильтров, написанных с использованием этой нотации, из [22].

(cn=Babs Jensen)
(!(cn=Tim Howes))
(&(objectClass=Person)(|(sn=Jensen)(cn=Babs J*)))
(o=univ*of*mich*)

Заметим, что при отсутствии условий на атрибут objectclass, фильтру могут удовлетворять (и, соответственно, клиенту будут выданы), записи совершенно различных классов, в описании которых допускается содержание атрибутов данного типа.

Область применения, перспективы

Многие не упомянутые здесь аспекты работы с LDAP, как то API, стандартные способы применения и соответствующие схемы, поддержка языковых тэгов (включение в запрос информации о желаемом языке ответа) и многое другое, находятся в стадии активного обсуждения и стандартизации (см. документы рабочей группы IETF LDAPext), многие уже широко распространены и приняты de facto.

Область применения LDAP постоянно и стремительно расширяется. На текущий момент это - справочники white pages (шлюзы из административных справочников X.500, административные справочники организаций, служб Yahoo, BigFoot и др., которые используются по умолчанию в популярных почтовых клиентах типа Microsoft Outlook, Netscape Messenger и проч. - см. RFC 2798, RFC 1274, RFC 2218), управление интранет в организациях (как замена NIS, средство интеграции и центр управления гетерогенными корпоративными сетями - см. RFC 2307), и многие другие, менее общие применения.

Реализации LDAP имеются у ведущих производителей, таких как iPlanet (Sun/Netscape Alliance), Netware Directory Server и др. Недавно представленный и рекомендованный Microsoft сервис Active Directory тоже являет из себя LDAP-сервер.

Широкое признание и распространение, а также универсальность модели данных позволяют говорить о LDAP как о серьезном претенденте при решении поставленных выше задач по поддержке распределенного поиска в информационной системе, претендующей на возможность широкого распространения.

Далее мы детально рассмотрим возможности LDAP в контексте решаемой нами задачи, и конкретно обсудим уже явно наметившиеся аналогии в модели данных и проч.

ПЕРЕЧЕНЬ ЗАДАЧ, ВХОДЯЩИХ В ПОДДЕРЖКУ РАСПРЕДЕЛЕННОГО ФУНКЦИОНИРОВАНИЯ

Еще раз обсудим предлагаемую архитектуру распределенной системы, четко выделив аспекты, относящиеся к поддержке ее распределенного функционирования.

Итак, система состоит из множества узлов, в общем случае неоднородных в смысле систем хранения данных, используемых платформ и интерфейсов. Объединяющим фактором для этих узлов является представимость хранимых на них информационных ресурсов в виде атрибутированных объектов. Это принципиально позволяет, формализовав и классифицировав наборы ресурсов на каждом из узлов (например, какие атрибуты содержит описание персоны на каждом узле), согласовать единый набор классов, или правила автоматизированного преобразования этих схем.

Единый интерфейс

Соответственно, первая из задач, которая встает при объединении отдельных узлов в единую систему, это предоставление на каждом из них ***единообразного, плат-***

формонезависимого интерфейса манипуляции с данными в терминах атрибутированных объектов. Причем, для возможности преобразования схем при передаче данных между узлами, этот интерфейс должен предусматривать манипуляции на уровне отдельных атрибутов и их значений (например, преобразование ФИО персоны из трех отдельных атрибутов на одном узле в одну строчку на другом).

Решив эту задачу, мы получаем возможность обмениваться данными между узлами системы, а также получать данные для передачи клиенту, независимо от их типов (будь то СУБД организации, библиотечная система, интернет-каталог и т.д.). Вопросы преобразования схем при передаче между узлами, и учета их специфики при других операциях, представляют собой отдельную тему для обсуждения. Далее при изложении мы будем считать, что узлы нашей системы имеют одинаковую схему данных, и лишь обозначать трудности, возникающие на разных этапах при необходимости преобразования.

Проблемы поиска

Остальные задачи возникают при необходимости обеспечить эффективный поиск в распределенной системе. Решение, предлагающее направить запрос каждому из узлов системы, а затем суммировать ответ, не подходит по целому ряду причин:

• во-первых, потенциально огромное количество этих узлов не позволяет надеяться на приемлемое время ответа;

• во-вторых, неизвестные условия связи между ними, и разные вычислительные мощности узлов могут еще более усугубить проблему;

• в-третьих, в зависимости от требований к возможностям языка запросов, один узел не всегда в состоянии ответить на запрос самостоятельно (ему могут понадобиться данные с других узлов).

Индексирование, маршрутизация запросов

Первая из перечисленных выше проблем (большое количество узлов) решается следующим образом. Узлы организовываются в иерархию, соответствующую зонам ответственности за информацию. Например, в рамках структуры Академии Наук вершиной иерархии может быть сервер Президиума, непосредственные подчиненные - сервера отделений, далее некая иерархия узлов различных организаций. На каждом из серверов накапливается в относительно компактном виде информация с подчиненных серверов, позволяющая предварительно оценить, каким из подчиненных серверов стоит передавать поисковый запрос. Образно говоря, получив запрос по тематике Computer Science, сервер Президиума может отсечь большую часть дерева, относящуюся к гуманитарным наукам, административным структурам и т.п. Технология отсечения ветвей при прохождении распределенного запроса называется *маршрутизацией запросов* (query routing, см. [34]).

Компактной формой представления подобной индексной информации являются *центроиды*. Выделим несколько совокупностей (коллекций) данных, которые мы собираемся индексировать (это могут быть совокупности данных, хранящихся на каждом из подчиненных узлов, либо более узкие совокупности, например, разбиение по рубрикатору). Определим, по каким атрибутам объектов осуществляется поиск (ранее вводился тезис, согласно которому эти атрибуты составляют сравнительно небольшую по объему часть хранимых ресурсов - например, библиографическая информация по сравнению с полным текстом). Далее для каждой такой совокупности нам нужно составить компактное представление того, какие значения каждого из интересующих нас атрибутов содержатся в объектах коллекции. Для текстовых атрибутов, поиск по которым происходит как задание условий на содержание тех или иных слов и подстрок, решение достаточно просто - нужно составить такое значение, в котором по одному разу встречались бы все слова из значений данного атрибута в объектах коллекции.

Например, при индексации названий книг из двух значений "Java: справочник программиста"и "Пособие для программиста на Perl"получится "Java справочник программиста пособие Perl". Еще большей компактности можно достичь приведением слов в каноническую форму при индексации и поиске.

Объект, полученный для каждой коллекции и содержащий подобные значения каждого из индексируемых атрибутов, позволяет предварительно оценить, содержатся ли в коллекции объекты, удовлетворяющие запросу. Такой объект называют центроидом. Более подробно об этой технологии см. [3, 35].

После построения центроида он пересылается вверх по иерархии узлов, встает также задача поддержки его актуальности.

Итак, вторая группа задач, входящая в поддержку распределенного поиска, это ***построение центроидов, обмен ими и поддержание их актуальности***.

Репликация данных

Вторая проблема - неоднородность вычислительных и коммуникативных возможностей в разных узлах системы, решается путем перебалансировки нагрузки. Именно, если какой-то из узлов, ответственный за часто запрашиваемые данные, имеет плохой канал связи и/или небольшие вычислительные мощности, то ожидание ответа от этого узла задерживает ответ системы в целом (не говоря уже о случае, когда связь с узлом вообще ненадежна).

В таких случаях целесообразно на регулярной основе поддерживать полную копию ресурсов узла на вышестоящем (предположительно более мощном) узле. Например, сервер федеральной библиотеки может поддерживать копию данных о своих филиалах. При этом ответственность за данные целиком лежит на филиалах, центральный же сервер лишь поддерживает актуальность копий и отвечает на клиентские запросы на их основании.

Итак, еще одна задача поддержки распределенного поиска - это ***репликация данных и поддержка актуальности копий*** в целях балансировки нагрузки.

Поддержка условий целостности

Третья из поставленных проблем - необходимость поддержать задание условий на связи в поисковом языке (что на первый взгляд делает невозможным получение ответа системы как простой суммы ответов некоторых ее узлов) - тоже имеет решение.

Именно, если некий ресурс A на одном из узлов X непосредственно связан с ресурсом B на другом узле Y, то поисковая информация A должна иметься на Y (с пометкой о том, что мастер-копия хранится на X), а соответствующая информация о ресурсе B - на узле X. В этом случае каждый из узлов может ответить на вопрос о связанности этих ресурсов без обращений вовне.

Заметим, что реплицируются не ресурсы целиком, а только поисковые данные, причем только для ресурсов, *непосредственно* связанных между собой (т.е. если реплицированный ресурс, в свою очередь, связан с третьим, такая связь не ведет к репликации). Кроме того, соблюдение этих условий имеет смысл только на тех узлах, которые участвуют в обработке клиентских запросов (т.е. на узлах с достаточной вычислительной мощностью). Все эти соображения позволяют снизить объем данных, которые необходимо передавать для поддержки непосредственных связей в поисковом языке, до приемлемого уровня. Сопровождение данных по-прежнему остается задачей узла-владельца ресурса, актуальность копий должна поддерживаться автоматически.

Итак, третья задача поддержки распределенного поиска - это ***репликация в целях поддержки связанности***.

Далее мы подробно рассмотрим решение этих задач на основе протокола LDAP, предлагаемое в ИСИР.

LDAP КАК СРЕДСТВО ПОДДЕРЖКИ РАСПРЕДЕЛЕННОГО ФУНКЦИОНИРОВАНИЯ

Единый интерфейс

Прежде всего надо отметить, что определение информационного ресурса как набора именованных атрибутов и определение LDAP-объекта практически совпадают (с точностью до терминологии). Действительно, мы можем хранить данные ресурса в атрибутах LDAP-записи. Тип атрибута определяет семантику операций сравнения и пр., некоторые атрибуты могут содержать DN связанных объектов, осуществляя таким образом связи. Единственным серьезным ограничением является атомарность значений атрибутов в LDAP. Многие системы существенно используют структуру значений атрибутов, таких как, например, адрес (отдельно используются индекс, название улицы, города и т.д.), чего сложно достигнуть, поместив такой структурированный атрибут в качестве одной строки в атрибуте LDAP-объекта. Далее мы вернемся к решению этой проблемы на примере ИСИР.

Следующий аспект, в котором наблюдается сходство концепций - это структура распределения информации. И в предложенной архитектуре распределенной информационной системы, и в модели данных LDAP предлагается иерархия серверов, которые вместе поддерживают распределенную иерархию данных в виде атрибутированных объектов. Также совпадает доменная концепция именования ресурсов.

Выше упоминалась необходимость формализации набора классов информационных ресурсов на каждом узле системы. Основой такой формализации вполне может служить схема LDAP, которая также обеспечит автоматический контроль.

Принципиальные возможности по преобразованию схем в LDAP также налицо - мы имеем возможность выбрать интересующие нас атрибуты из разнотипных объектов на одном сервере, и сформировать из них объект на другом, не выходя за рамки протокола.

Все эти соображения становятся существенно более интересными с точки зрения практики, если принять во внимание следующие факты.

Платформонезависимая реализация, система backend'ов.

Существует в виде свободно распространяемых исходных кодов реализация сервера и клиентского API для протокола LDAP (OpenLDAP foundation, http://www.openldap.org). Эта реализация относительно ОС-независима (ПО собирается на большинстве UNIX-подобных систем, а также на платформах Win32 - Win95/98/NT/2000). Кроме того, внутренняя организация сервера такова, что части, реализующие собственно сетевой протокол, контроль схемы, разграничение доступа и другие не зависящие от способа хранения объектов аспекты, выделены в общую часть (frontend), собственно же манипуляции с данными производятся с помощью внутреннего API, допускающего разные способы реализации (backend'ов).

Это позволяет позиционировать LDAP-сервер как универсальный шлюз к различного рода узлам (будь то организация или каталог с СУБД, другая система или др.) - для обеспечения взаимодействия достаточно реализовать соответствующие backend'ы. Эта задача представляется более легкой, чем реализация "с нуля"всех аспектов обмена, начиная с сетевых коммуникаций и кончая представлениями ресурсов и их конвертацией.

Для апробирования работоспособности LDAP-сервера в качестве интерфейса к узлу информационной системы в рамках ИСИР РАН был реализован SQL-backend к серверу OpenLDAP, в настоящее время включенный в состав версии 2.0-beta и доступный вместе со всем пакетом на публичном FTP-сервере OpenLDAP (подробности см. http://www.openldap.org).

Backend позволяет представлять информацию, хранимую в реляционной СУБД, в виде LDAP-дерева, и дает возможность организовать обмен между узлами, использующими различные ОС, СУБД разных производителей, разные реляционные схемы.

Опишем кратко принцип функционирования этого ПО, проблемы, возникшие при его использовании в рамках ИСИР, узлы которой используют РСУБД, и их решение.

LDAP-сервер как интерфейс к узлу ИСИР

Более конкретное описание, инструкции по установке и примеры использования back-sql можно получить в со-

ставе пакета OpenLDAP 2.0, здесь же мы приведем его описание на концептуальном уровне.

Концепция, лежащая в основе работы back-sql, заключается в том, что любая реляционная схема представляет собой модель некоторой предметной области, и в конечном используется именно в терминах этой предметной области. а не отдельных таблиц, составляющих ее. Стандартный цикл проектирования подразумевает на первом этапе создания ER-диаграммы, которая как раз и представляет из себя схему классов объектов предметной области, содержащих атрибуты, и связей между ними. На следующих этапах из соображений эффективности эти отношения нормализуются, давая в результате несколько связанных таблиц.

В результате очевидна по крайней мере одна схема LDAP, с помощью которой данная реляционная схема достаточно эффективно представима. Сопоставим каждой сущности в ER-диаграмме objectclass LDAP, атрибутам этой сущности - один или несколько атрибутов LDAP. Каждой сущности соответствует после нормализации как минимум одна таблица, хранящая записи, соответствующие экземплярам этой сущности. Атрибуты представляются либо как колонки той же таблицы, либо как содержимое некоторого количества связанных таблиц. Связи с другими сущностями также представляются некоторыми внешними ключами и дополнительными таблицами. Для каждого из этих атрибутов, соответственно, существует эффективный SQL-запрос, получающий его значения по идентификатору записи в главной таблице.

Имея метаинформацию о том, какая таблица соответствует какому objectclass'у, в каких колонках каких таблиц содержатся значения его атрибутов, и по каким условиям они объединяются с главной таблицей, мы получаем возможность автоматически транслировать LDAP-запрос на загрузку ресурса в серию SQL-запросов.

Еще один вид метаинформации - это соответствие первичных ключей записей в таблице DN'ам LDAP-объектов, а также иерархия этих объектов - дерево (понятно, что подобная информация в реляционной схеме полностью отсутствует).

Итого, back-sql требует добавления в существующую реляционную схему трех таблиц с фиксированными именами и структурой, которые содержат метаинформацию об objectclass'ах, их атрибутах и их соответствии SQL-запросам, а также о структуре информационного дерева и его соответствии первичным ключам соответствующих реляционных отношений.

Запросы на добавление, удаление и модификацию ресурсов сложно транслировать автоматически, т.к. связи между таблицами могут быть достаточно сложными. Однако в ИСИР поддержка работы с ресурсами выполнена в виде пакета хранимых процедур СУБД, поэтому эти типы запросов LDAP успешно транслируются в вызовы этих процедур (достаточно предоставить метаинформацию о том, какие процедуры и с какими параметрами нужно вызывать для добавления, удаления, или модификации отдельных атрибутов каждого objectclass'а).

Для эффективного выполнения поисковых запросов этого недостаточно, т.к. перебирать все объекты и проверять их на соответствие фильтру нереально.

Полностью транслировать поисковый фильтр LDAP в стандартный SQL невозможно, т.к. набор типов в SQL ограничен, в отличие от типов атрибутов LDAP, для которых могут быть определены собственные правила сравнения. Однако для большинства стандартных типов (числа, строки и т.п.) можно сгенерировать SQL-условия, соответствующие условиям фильтра (равенство, существование, наличие подстроки и т.п.). Поэтому можно сгенерировать запрос, выбирающий "кандидатов"на соответствие по частичному фильтру, а затем уже средствами LDAP frontend проверить их на соответствие полному фильтру.

Представление связей, которые в реляционной схеме осуществляются внешними ключами и дополнительными таблицами, а со стороны LDAP должны выглядеть как атрибуты, содержащие DN связанного объекта (типа authorDN у публикации), возможно следующим образом. Благодаря тому, что используемая метаинформация находится в той же СУБД, что и собственно данные, мы можем сконструировать запрос, который по ID связанного объекта (по внешнему ключу) получит из соответствующей служебной таблицы DN объекта.

Добавление или удаление связи со стороны LDAP выглядит как добавление или удаление соответствующего DN к значениям нужного атрибута, следовательно, для реализации этих операций соответствующие хранимые процедуры должны по DN найти ID связанных объектов, и добавить или удалить запись из связывающей их таблицы.

Были произведены работы по настройке этого ПО на схему ресурсов ИСИР для СУБД Oracle, MS SQL server, ОС Solaris,Linux,WinNT,Win2000, организован обмен данными между тестовыми узлами ИСИР, использующими разные платформы.

Детали обмена будут обсуждены ниже, здесь же следует сказать о двух встреченных при настройке проблемах представления.

Первая - структурированные атрибуты типа "адрес". Дело в том, что ИСИР предоставляет широкие возможности поиска по разным частям, составляющим адрес - как то, по городу, индексу и т.д. Поэтому оформление адреса как одного строкового атрибута означает потерю важной структурной информации при обмене. Поэтому было принято решение выделить адрес в самостоятельный objectclass, и вместо литерального адреса выводить ссылку на DN соответствующего объекта.

Вторая - поддержка разных языков для одних и тех же атрибутов. Вообще говоря, существуют предложения по введению в LDAP-запросы тэгов языка, определяющий желаемый язык ответа, или язык вновь добавляемого значения (см. [24]). Однако на данный момент эти предложения находятся на этапе стандартизации, и опыт их использования нам не известен.

На уровне реляционной схемы ИСИР данные на разных языках хранятся в разных записях с соответствующими тэгами. Поэтому было принято временное решение на уровне запросов и хранимых процедур, занимающихся преобразованием реляционных данных в LDAP и обратно, дописывать к значениям префиксы вида «language tag>: ...". При извлечении для LDAP к значению приписывается тэг (и для LDAP-клиента оно имеет вид "ru:текст значения"). При добавлении и изменении LDAP-клиент предоставляет значение, предваренное таким префиксом, а соответствующие процедуры выделяют его, и

размещают в БД значение без префикса, но в запись с нужным тэгом.

Итак, решение задачи единого платформонезависимого интерфейса на основе LDAP реализовано и апробировано в ИСИР.

Индексирование

Второй задачей, обозначенной ранее, является построение и обмен центроидами. Здесь тоже очевидно удобство использования LDAP, с учетом того, что имеется представление данных на каждом узле системы в виде LDAP-поддерева.

Именно, для создания центроида, как обсуждалось выше, необходимо выделить описываемую совокупность объектов и далее для каждого из интересующих атрибутов загрузить все значения. Поисковые запросы LDAP вполне подходят для этой цели - мы имеем возможность задать совокупность как некоторое поддерево (что может соответствовать данным одной организации, либо какой-то ветви рубрикатора в этой организации), и при необходимости далее ограничить совокупность путем задания условий на атрибуты. Например, выделить все персоны в пределах одной организации не составляет труда. Далее, поисковый запрос предлагает возможность указать, какие атрибуты нам нужны, и не загружать объекты целиком. Это все, что нужно при выборке данных для построения центроида.

Кроме того, уже построенный центроид также целесообразно оформить, хранить и транспортировать в виде LDAP-объекта, т.к. те же поисковые возможности LDAP пригодятся в процессе маршрутизации клиентских поисковых запросов для просмотра центроид-индексов.

Еще один большой пласт вопросов, связанный с заданием конфигурации потоков обмена индексами, в этой работе не рассматривается. Действительно, описанное выше дает лишь принципиальную возможность обмена индексами и транспортный протокол для этого. Каким образом администратор конфигурирует и инициирует автоматический обмен индексами между узлами, как и какие совокупности объектов, какие наборы атрибутов индексируются для достижения максимальной отдачи от применения маршрутизации запросов - все эти вопросы здесь не освещаются. Заметим, что в этой области также существуют попытки стандартизации схем обмена и протоколов управления (см. [8, 9, 10, 11]), в том числе и с помощью LDAP (во всяком случае, LDAP уже широко используется для управления корпоративными сетями, существуют стандартные схемы и ПО, использующее их). Эта часть ИСИР находится в стадии проектирования.

Репликация, поддержка связей

Другие две задачи - репликация данных для балансировки нагрузки и для поддержания связей в поисковых запросах - также имеют базовое решение на LDAP.

В рамках ИСИР реализована и апробирована низкоуровневая компонента, которая, используя LDAP-интерфейс, осуществляет обмен между узлами под управлением специализированного языка заданий. Предложение языка заданий описывает команду по переносу, в которой указывается, какую часть поддерева (начальный узел, область - ONELEVEL или SUBTREE, фильтр, список атрибутов) взять, и куда перенести. Очевидно, что этого достаточно для организации и того, и другого вида репликации. Кроме того, язык предусматривает режим обновления, когда в целевом дереве создаются только новые объекты, а существующие лишь обновляются, т.е. существует базис для механизмов поддержки актуальности.

Здесь опять возникает большой пласт вопросов, аналогичных управлению индексированием. Каким образом автоматизируется составление заданий для этой компоненты низкого уровня, как описать, между какими узлами и в какую сторону нужно организовать обмен, и т.д. Ситуация в этой области точно такая же - существует много предложений, находящихся на разных стадиях стандартизации, нуждающихся в анализе и апробации. В их числе, опять же, есть предложения и на основе LDAP. В рамках ИСИР проводится работа по изучению существующих предложений и проектированию максимально открытой архитектуры, т.к. это позволило бы существенно облегчить плотную интеграцию различных существующих систем (на уровне взаимодействия, а не импорта данных).

Дополнительные применения LDAP

Кроме описанных выше применений, использование LDAP дает дополнительные полезные возможности.

Существуют стандартные схемы LDAP для его использования в целях администрирования корпоративных сетей, разграничения доступа и т.д. (см. RFC 2307 и др.). Можно рассмотреть возможность использования этих стандартов для решения аналогичных задач в распределенной информационной системе.

Широкое распространение и доступность LDAP (наличие клиентов в популярных почтовых программах и броузерах,standalone-клиенты) делают осмысленным позиционирование LDAP-интерфейса к системе не только в качестве внутреннего протокола поддержки функционирования, но и в качестве самостоятельного клиентского интерфейса. Например, кроме служебной схемы, средствами backend'ов можно представить данные в соответствии со стандартами типа RFC 2798, RFC 1274, RFC 2218, что позволит использовать систему как глобальный справочник white pages.

ДРУГИЕ ПОДХОДЫ И ТЕХНОЛОГИИ. КРАТКИЙ ОБЗОР И СРАВНЕНИЕ

Соблазнительная особенность использования LDAP в контексте распределенных информационных систем описываемого типа состоит в том, что он дает почти готовые и стандартизованные решения на всех уровнях, начиная с модели данных, набора операций, средств сетевого обмена, разграничения доступа, API, распределенного функционирования, расширения, обработки результатов и т.д.

Кроме существенного уменьшения затрат на разработку, это дает также надежду на доступность и открытость полученной системы за счет того, что не изобретаются никакие специфичные для системы протоколы.

Конечно, и сама модель, и связанные с ней стандарты определенным образом ограничивают возможности по сравнению с другими технологиями, однако это происходит за счет проработанности и стандартизации. Ни одна другая технология не стандартизует такое количество аспектов функционирования распределенной системы сразу (этот тезис мы попытаемся проиллюстрировать на примерах ниже).

Кроме того, ничто не заставляет при использовании LDAP отказываться от других технологий, заменяющих или дублирующих LDAP в соответствующих аспектах (альтернативные интерфейсы доступа, форматы и способы обмена данными). Неслучайно везде в тексте описываемая технология позиционируется как технология *поддержки* распределенного функционирования, и ни в коей мере не ограничивает его только рамками LDAP (конкретные аспекты, в которых применение LDAP видится целесообразным, суммированы выше).

Итак рассмотрим кратко некоторые другие технологии и возможные области их применения для решения задач, поставленных в начале работы (организация единого интерфейса узла, синхронизация моделей данных, индексирование и распределенный поиск, обработка результатов).

CORBA+CORBAservices

Собственно CORBA (см. [36]) предоставляет возможность разработки распределенных приложений в обычной концепции ООП - набора объектов, взаимодействующих посредством вызовов методов друг друга (теперь объекты могут находиться на различных узлах распределенной системы). Вопросы определения интерфейсов их взаимодействия, проектирования этих конкретных интерфейсов с точки зрения устойчивости по отношению к условиям связи в Интернет (разрывам и проч.) никак не специфицируются.

В случае распределенной информационной системы обсуждаемого типа использование CORBA, скорее всего, означает определение CORBA-интерфейса объекта "узел системы", и планирование функционирования системы как набора взаимодействующих экземпляров такого объекта. Вопрос о попытке представить в виде объекта каждый из хранимых в системе информационных ресурсов обсуждался в самом начале, там же был сделан вывод, что таким образом максимум можно представить заведомо ограниченное множество ресурсов, полученных в результате поиска. Ни этот интерфейс (который должен включать средства создания и обмена индексами, служебной информации, миграции данных для балансировки нагрузки и проч., и проч.), ни язык, на котором можно было бы запросить эти ресурсы, нигде не стандартизован.

К важным наработкам в области стандартизации использования CORBA относится набор рекомендаций OMG (группы-разработчика CORBA) под названием CORBAServices [36]. В нем предлагается масса рекомендованных объектов и способов их взаимодействия для решения типичных задач распределенных систем. В том числе, многое из этих рекомендаций имеет отношение к информационным системам обсуждаемого типа (коллекции объектов, рекомендованный язык запросов OQL и проч.). Однако охватывается далеко не весь круг поставленных выше задач, предлагаемые решения громоздки в плане архитектуры и реализации, по фактору доступности и наличию конкретных реализаций они далеки от желаемого. Скорее, они относятся к более высокому уровню функционирования системы, не затрагивая вопросы поддержки распределенного функционирования на более низком уровне, которые, в основном, и обсуждаются в работе.

Тема данной работы не позволяет обсуждать этот вопрос более подробно, можно лишь сказать, что CORBA несомненно является одной из наиболее перспективных технологий в данной области, и в планы развития ИСИР входит реализация вышеупомянутых интерфейсов на CORBA по мере стандартизации решений на этой платформе. Возможно (и даже скорее всего), внутренняя реализация этих интерфейсов будет использовать уже готовые механизмы на основе LDAP.

XML, RDF - обмен

Пакет стандартов вокруг XML [27] покрывает другой аспект функционирования распределенной системы - мало занимаясь вопросами транспорта и вообще механизмов обмена, он прорабатывает вопросы представления структурированных данных, разбора и обработки таких представлений.

Однако полностью стандартизованы, опять, только самые общие технологии, позволяющие задавать синтаксис подмножеств XML для конкретной предметной области. В изучаемой же предметной области еще нет никакой ясности по поводу представления ресурсов.

Множество предложений рассматриваются и применяются в различных системах, начиная от форматов метаописаний произвольных ресурсов (Dublin Core, RDF и т.д. - см. [28]), и кончая традиционными форматами библиотечных систем типа MARC и т.д.

Тем более неясно дело со стандартами записи всех этих описаний на XML, которые позволили ли бы надеяться на реальную интероперабельность систем, работающих с этим языком разметки, без дополнительных преобразований, кроме того, за исключением языка запросов XQL, проблемы поиска в огромной распределенной коллекции ресурсов также не прорабатываются.

Более подробное обсуждение этих вопросов также выходит за рамки данной работы. В качестве направления использования XML и сопутствующих технологий в ИСИР можно указать следующее. Разрабатываются средства импорта и экспорта данных в виде RDF/XML-документов (наряду с другими стандартами), в дальнейшем, по мере разработки этой концепции, эти данные будут сопровождаться выгрузкой метаописания схемы данных в некотором стандартизованном XML-виде, что позволит автоматизировать преобразования схем при обмене между разнородными узлами. Протоколы обмена также находятся в стадии обсуждения (см. [30] и др.).

Harvest и подобные системы

В отличие от обсужденных выше технологий, затрагивающих разные отдельные аспекты построения распределенных систем, Harvest [35] и подобные представляют собой законченные, уже функционирующие и продолжающие развиваться системы, в которых в числе прочего

осуществляется и поддержка распределенного функционирования (индексация и пр.).

Существует также инициатива, использующая потокол LDAP как основной протокол распределенной системы (см. [31, 32, 33]).

Однако они изначально рассчитаны на существенно более слабую интеграцию распределенной информации (отсутствие пресловутых связей в поисковых языках).

В ИСИР планируется реализовать как можно более плотные схемы интеграции с такими системами (полный сервис для них со стороны ИСИР, и ограниченный сервис с их стороны, как отдельных узлов).

ЗАКЛЮЧЕНИЕ

Предложенная в данной статье технология позволяет интегрировать в единую информационную среду разнотипные информационные системы, цифровые библиотеки, и обеспечить логическую связанность данных этой среды. Хотя в различных случаях степень интеграции может быть разной, в целом качество информации в подобной распределенной системе существенно превышает уровень текущих веб-технологий (поисковые системы в Интернет и т.п.).

Использование для реализации протокола LDAP позволило уменьшить затраты на разработку, не ограничивая перспектив развития, открытости или функциональности проектируемой системы, и выявило несколько дополнительных преимуществ. Именно легковесность в смысле затрат на разработку делает использование данной технологии целесообразной во многих случаях, так как позволяет приступить к отлаживанию высокоуровневых аспектов функционирования на раннем этапе, и предлагает хороший уровень функциональных возможностей при использовании.

Список литературы

Рабочие документы IETF (RFCs)
Электронные версии этих документов доступны по адресу
http://www.rfc-editor.org/rfc.html

WHOIS++

[1] *J. Gargano, K. Weiss.* RFC1834: Whois and Network Information Lookup Service, Whois++. August 1995 (INFORMATIONAL)

[2] *P. Deutsch, R. Schoultz, P. Faltstrom, C. Weider.* RFC1835: Architecture of the WHOIS++ service. August 1995 (PROPOSED STANDARD)

[3] *C. Weider, J. Fullton, S. Spero.* RFC1913: Architecture of the Whois++ Index Service. February 1996 (PROPOSED STANDARD)

[4] *P. Faltstrom, R. Schoultz, C. Weider.* RFC1914: How to Interact with a Whois++ Mesh. February 1996 (PROPOSED STANDARD)

[5] *S. Williamson, M. Kosters, D. Blacka, J. Singh, K. Zeilstra.* RFC2167: Referral Whois (RWhois) Protocol V1.5. June 1997 (INFORMATIONAL)

CIP

[6] *J. Allen, M. Mealling.* RFC2651: The Architecture of the Common Indexing Protocol (CIP). August 1999 (PROPOSED STANDARD)

[7] *J. Allen, M. Mealling.* RFC2652: MIME Object Definitions for the Common Indexing Protocol (CIP). August 1999 (PROPOSED STANDARD)

[8] *J. Allen, P. Leach, R. Hedberg.* RFC2653: CIP Transport Protocols. August 1999 (PROPOSED STANDARD)

[9] *R. Hedberg, B. Greenblatt, R. Moats, M. Wahl.* RFC2654: A Tagged Index Object for use in the Common Indexing Protocol. August 1999 (EXPERIMENTAL)

[10] *T. Hardie, M. Bowman, D. Hardy, M. Schwartz, D. Wessels.* RFC2655: CIP Index Object Format for SOIF Objects. August 1999 (EXPERIMENTAL)

[11] *R. Hedberg.* RFC2657: LDAPv2 Client vs. the Index Mesh. August 1999 (EXPERIMENTAL)

X.500

[12] *C. Weider, J. Reynolds.* RFC1308: Executive Introduction to Directory Services Using the X.500 Protocol. March 1992 (FYI13)

[13] *C. Weider, J. Reynolds, S. Heker.* RFC1309: Technical Overview of Directory Services Using the X.500 Protocol. March 1992 (FYI14)

[14] *S. Hardcastle-Kille, E. Huizer, V. Cerf, R. Hobby, S. Kent.* RFC1430: A Strategic Plan for Deploying an Internet X.500 Directory Service. February 1993 (INFORMATIONAL)

[15] *P. Barker, S. Kille, T. Lenggenhager.* RFC1617: Naming and Structuring Guidelines for X.500 Directory Pilots May 1994 (INFORMATIONAL)

LDAP

[16] *W. Yeong, T. Howes, S. Kille.* RFC1777: Lightweight Directory Access Protocol. March 1995 (DRAFT STANDARD)

[17] *M. Wahl, T. Howes, S. Kille.* RFC2251: Lightweight Directory Access Protocol (v3). December 1997 (PROPOSED STANDARD)

[18] *S. Kille, M. Wahl, A. Grimstad, R. Huber, S. Sataluri.* RFC2247: Using Domains in LDAP/X.500 Distinguished Names. January 1998 (PROPOSED STANDARD)

[19] *A. Grimstad, R. Huber, S. Sataluri, M. Wahl.* RFC2377: Naming Plan for Internet Directory-Enabled Applications. September 1998 (INFORMATIONAL)

[20] *M. Wahl, S. Kille, T. Howes.* RFC2253: Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names. December 1997 (PROPOSED STANDARD)

[21] *T. Howes, M. Smith.* RFC1823: The LDAP Application Program Interface. August 1995 (INFORMATIONAL)

[22] *T. Howes.* RFC2254: The String Representation of LDAP Search Filters. December 1997 (PROPOSED STANDARD)

[23] *M. Wahl, A. Coulbeck, T. Howes, S. Kille.* RFC2252: Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions. December 1997 (PROPOSED STANDARD)

[24] *M. Wahl, T. Howes.* RFC2596: Use of Language Codes in LDAP. May 1999 (PROPOSED STANDARD)

Рабочие документы IETF (internet drafts)

Электронные версии этих документов доступны по адресу http://www.ietf.org/ietf/1id_abstracts.txt. Однако эти документы действительны в течение ограниченного времени, и могут быть удалены или заменены на другие.

LDAP

[25] *M. Smith, T. Howes, A. Herron, M. Wahl, A. Anantha.* <draft-ietf-ldapext-ldap-c-api-04.txt> The C LDAP Application Program Interface. October 1999

[26] *Christopher Lukas, Tim Howes, Michael Roszkowski, Mark C. Smith, Mark Wahl.* <draft-ietf-ldapext-namedref-00.txt> Named Referrals in LDAP Directories. June 1999

Рабочие документы W3C

XML, RDF, ICE

[27] *Tim Bray, Jean Paol and C. M. Sperberg-McQueen.* Extensible Markup Language (XML) 1.0 http://www.w3.org/TR/REC-xml

[28] *Lassila, Ora, Ralph R. Swick.* Resource Description Framework (RDF) Model and Syntax. http://www.w3.org/TR/REC-rdf-syntax

[29] *Dan Brickley, R.V. Guha, Andrew Layman.* Resource Description Framework, (RDF) Schemas. http://www.w3.org/TR/WD-rdf-schema

[30] *Neil Webber, Conleth O'Connell, Bruce Hunt, Rick Levine, Laird Popkin, Gord Larose.* The Information and Content Exchange (ICE) Protocol. 26 October 1998. http://www.w3.org/TR/NOTE-ice

Другое

Isaac project

[31] *C. Lukas, M. Roszkowski.* The Isaac Network: LDAP and Distributed Metadata for Resource Discovery. IEEE Metadata 1999 Conference. http://computer.org/conferen/proceed/meta/1999/papers/46/cl

[32] *C. Lukas, M. Roszkowski.* A Distributed Architecture for Resource Discovery Using Metadata. June 1998, D-Lib Magazine. http://www.dlib.org/dlib/june98/scout/06roszkowski.html

[33] 33. Project Isaac Architecture Overview for Collaborators. http://scout.cs.wisc.edu/research/arch/index.html

Другое

[34] *Kirriemuir J., Brickley D.* Cross-searching Subject Gateways: the Query Routing and Forward Knowledge Approach. D-Lib Magazine, January 1998.

[35] *Bowman C.M. etc.* Harvest: A Scalable, Customizable Discovery and Access System. University of Colorado, technical report CU-CS-732-94, March 1995. http://www.tardis.ed.ac.uk/harvest/

[36] OMG documents. http://www.omg.org

[37] *Бездушный А.Н.* Подход к созданию интегрированных цифровых библиотечных систем. ОСМО ВЦ РАН, ТО-1999-4