International symposium on the contribution of Europeans to the evolution and the achievements of computer technology "Computers in Europe. Past, Present and Future", Kyiv, October 5-9, 1998
http://www.icfcst.kiev.ua/Symposium/Proceedings2/Bigdan.pdf

ВЗАИМОДЕЙСТВИЕ РАСПРЕДЕЛЕННЫХ СИСТЕМ ХРАНЕНИЯ НАУЧНО-ТЕХНИЧЕСКОЙ ИНФОРМАЦИИ

Бигдан В.Б., Малашок Т.И., Малиновский Л.Б., Почекайлов В.В.
ICFCST[1], Київ, Україна

Излишне говорить о значении обмена научно-технической информацией между учеными разных стран. Сеть Internet стала достоянием миллионов ученых в различных странах мира. Однако, если транспортные средства сети справляются с возложенной на них задачей и основные их компоненты изменяются сравнительно редко, то информационная структура сети находится в процессе постоянных преобразований и главная цель таких преобразований — обеспечить средства доставки пользователю нужной информации в нужном формате. Образно, основную проблему взаимодействия пользователя с информационной системой можно описать так. Сетевые средства могут хранить огромные объемы данных и доставлять их с большой скоростью в любую точку Земли, но они не знают, *что хранить* и *куда доставлять*. Пользователи, с другой стороны хорошо представляют сферу своих интересов, но не знают какие данные могут иметь отношение к интересующим их вопросам и где эти данные могут храниться. Другими словами, пользователи знают, что они хотят получить, но не знают *что искать* и *где искать*. Практически каждый пользователь, имеющий достаточно большой опыт работы в Internet, сталкивался с ситуацией, при которой он тратил много времени на поиск данных и совершенно случайно обнаруживал, что нужная информация располагается на сервере, который он неоднократно посещал.

Для решения проблемы автоматизации сбора и поиска данных создаются многочисленные специализированные системы, Одной них, стала распределенная система хранения научно-технической информации "Интеллект" [1], разработанная авторами данной статьи. Одной из проблем развития данной системы стала организация ее взаимодействия с другими системами подобного назначения.

Успех деятельности каждой системы, состоит ли она из одного сервера или объединяет в себе серверы, расположенные на значительном расстоянии один от другого, зависит от объема и качества хранящейся в ней информации. Сбор данных организуется двумя взаимодополняющими способами. Во-первых, информация поставляется владельцами (в частности, авторами оригинальных разработок), заинтересованными в ее распространении. Во-вторых, производится целенаправленный сбор свободно распространяемых данных из Internet.

Каждый из указанных способов порождает специфические проблемы. Владелец информации, представив ее на одной из информационных систем, как правило, не обращается к другим системам подобного назначения, справедливо предполагая, что дальнейшее распространение его данных должно происходить автоматически. При сборе информации из Internet количество и качество данных зависит от стратегии поиска и применяемых при этом программных средств.

В результате, на различных системах собирается различная информация и администраторам имеет смысл наладить взаимодействие и вести активный обмен

---

[1] International charity foundation for history and development of computer science and technique
http:\\www.icfcst.kiev.ua
e-mail: icfcst@icfcst.kiev.ua

хранящимися в системах данными. Подход, при котором в системе хранится не сама информация, а сообщение о её наличии и условиях доступа к ней, облегчает достижение договоренностей об обмене и гарантирует сохранение авторских прав на интеллектуальную собственность. Однако, в процессе обмена возникают значительные технические и организационные трудности.

Разные системы могут иметь различную структуру хранения и представления информации, форматы хранения данных могут различаться и т. д. Так как узлы информационных систем обслуживаются небольшим количеством специалистов, вопрос об обновлении баз данных вручную не может даже рассматриваться. При наличии соглашений об обмене информацией между различными системами, этот обмен может осуществляться лишь автоматически.

Традиционно, при обмене данными между различными системами используется некий промежуточный формат представления информации. В процессе преобразования данные, хранящиеся на одной системе преобразуются в промежуточный формат, затем выполняется повторное преобразование — в формат, используемый другой системой. Однако, в данном случае применению традиционного подхода препятствуют следующие причины.

- Необходимость разработки промежуточного формата, что при небольшом штате сотрудников может занять значительное время.
- Необходимость разработки и отладки программ преобразования.
- Сложность преобразования из-за различий в структурах систем.
- Излишняя загрузка каналов связи, неизбежная при обмене между различными системами.

Чтобы исключить данные недостатки и минимизировать затраты времени специалистов, обслуживающих систему, разработчиками системы был использован следующий подход.

После достижения договоренности об обмене информацией, программы, организующие взаимодействие с пользователями модифицируются. Функции, используемые для представления информации пользователю, дублируются аналогичными функциями, обращение к которым происходит через прикладной программный интерфейс (API). Таким образом организуется дополнительный интерфейс взаимодействия с системой, при котором роль пользователя выполняет программа (рис. 1).

Благодаря этому, клиент-программа, обращающаяся к API, получает возможность оперировать с те же данными, что и пользователь, представленными в том же формате. (Понятно, что функции API не могут быть полностью аналогичными функциям пользовательского интерфейса, в частности, API исключает визуальное представление данных поэтому информация, используемая для оформления экрана не передается).

При проектировании клиент-программы, возникает ряд противоречивых требований. Для подробного рассмотрения этих требований необходимо уточнить терминологию. Несмотря на то, что доступ должен быть симметричным, т. е. каждая система в разные моменты времени выполняет роль как источника, так и приемника информации, будем рассматривать передачу информации только в одну сторону. Назовем систему-источник данных *удаленной* системой, а систему-приемник данных *локальной* системой.

Итак, требования к клиент-программе можно сформулировать следующим

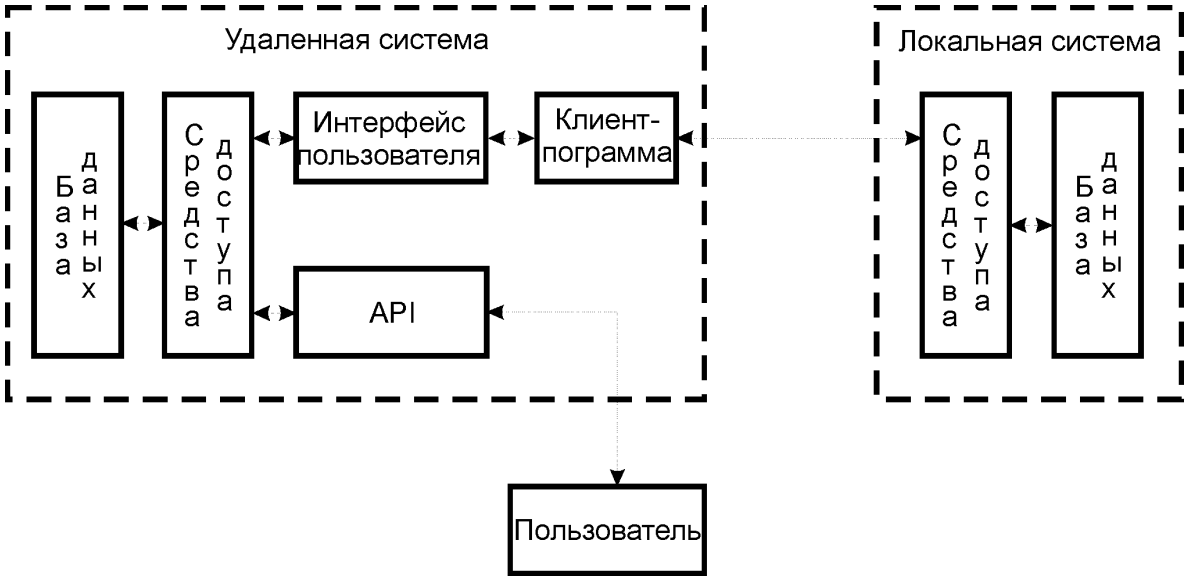

Рис.1.

образом.

- Программа должна взаимодействовать через API, следовательно должна находиться на удаленной системе и, управляться персоналом удаленной системы.
- Программа должна "представлять интересы" локальной системы, т. е. принимать информацию и преобразовывать ее в формат локальной системы. Следовательно, программа должна быть разработана специалистами локальной системы.
- Информация, полученная клиент-программой должна быть сохранена в локальной системе, т. е. желательно, чтобы она находилась на локальной системе.

Как видно, требования к программе, выполняющей взаимодействие с удаленным узлом противоречивы. Один из способов исключить противоречия, реализовать программу силами специалистов, обслуживающих локальную систему (в этом случае наилучшим образом будут учтены особенности представления данных) и разместить ее на удаленной системе. На удаленной системе программа может быть размещена по-разному. Например, она может быть запущена как специализированный сервер работающий в фоновом режиме и ожидающий обращения через TCP-порт. Другое решение состоит в том, чтобы выполнить программу в виде CGI-сценария, вызывающегося при обращении к Web-странице.

Ни одно из этих решений нельзя назвать оптимальным. Запуск в режиме сервера создает для персонала системы ряд неудобств, связанных с обслуживанием "чужой" программы, не являющейся частью системы. Взаимодействие с CGI-сценарием было бы удобно для пользователя, находящегося на локальном узле, Web-страница выполняла бы роль удобнго интерфейса, используемого при обращении к системе. Однако, программный вызов HTML-страницы и управление формой из программы неудобно и создает для разработчиков системы ряд трудностей.

Гораздо лучшее решение — реализовать клиент-программу, взаимодействующую с удаленной системой в виде мобильного агента [2]. Взаимодействие систем с использованием мобильного агента показано на рис. 2.

Работа мобильного агента построена ни использовании концепции удаленного программирования. Мобильная программа переписывает свой код на удаленный узел и запускается там на выполнение. Чтобы подобная миграция программы могла осуществиться, на удаленном узле постоянно поддерживается специальная управляющая структура, которая в терминах мобильных агент-программ называется "место". Эта структура предназначена для приема мобильного агента и запуска его на узле.

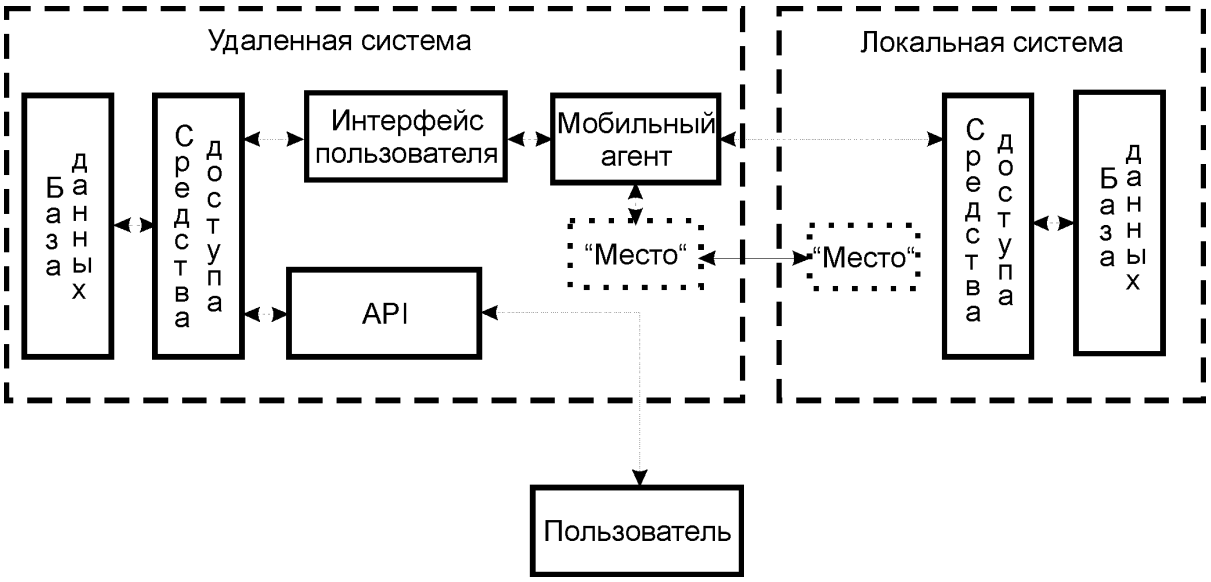

Рис. 2

Начиная работу с удаленной системой, мобильный агент запускается на локальной машине и переписывается на удаленный узел. Чтобы упростить управление программой и исключить необходимость поддержки взаимодействия между несколькими копиями одной программы, копия, расположенная на локальном узле уничтожается. Агент, расположенный на удаленной системе начинает взаимодействие с ней через API. Информация, полученная от системы, анализируется, оценивается и, при необходимости, копируется на локальную систему.

Так как копирование производится по инициативе клиент-программы, "представляющей" локальный узел, это позволяет устранить избыточность передаваемой информации и загрузка сети снижается.

По окончании работы копия мобильного агента на удаленном узле уничтожается. В обратном переходе агента на локальную систему нет необходимости, так как свою основную работу программа выполняет на удаленной системе и все сообщения о результатах своей работы передает по сети.

При использовании данного подхода достигается ряд преимуществ.

- В процессе работы клиент-программа находится на удаленном сервере. В результате, она имеет возможность вести как угодно длинные диалоги с системой поддержки данных, анализируя предлагаемую информацию. Уровень взаимодействия при этом ограничивается лишь "интеллектом" программ, участвующих в диалоге.
- Клиент-программа разработана специалистами, обслуживающими локальную систему. Следовательно, она может учитывать все особенности хранения данных

на локальной системе и использовать оптимальные методы сжатия при передаче по сети.

- Персонал удаленного узла избавляется от необходимости поддерживать "чужое" приложение, так как клиент программа посещает сервер только в те моменты, когда обе системы готовы к обмену данными. По окончании сеанса, экземпляр клиент-программы автоматически уничтожается. Единственное, что должны сделать специалисты системы для организации обмена — поддерживать на своем сервере "место", обеспечивающее запуск агента. Место позволяет контролировать работу агента (в частности, проверять, не являются ли его действия потенциально опасными для системы).

Информационные структуры, подобные системе "Интеллект" создаются для того, чтобы пользователь не *искал* информацию, а *получал* её. Эта цель достигается специализацией системы. Использование нового подхода, допускающего автоматическую синхронизацию содержимого различных узлов, позволяет вплотную приступить к построению систем нового класса — *взаимодействующих информационных систем*.

1. Бігдан В. Б., Малашок Т. І., Малиновський Л. Б., Почекайлов В. В. Досвід створення та теоретичні аспекти побудови системи обміну науково-технічними даними. Праці міжнародного симпозіуму "Комп'ютери в Європі минуле, сучасне та майбутнє. Стор. 421.
2. Jim White. Mobile Agents White Paper. (http://www.genmagic.com/agents/Whitepaper/whitepaper.html).

International symposium on the contribution of Europeans to the evolution and the achievements of computer technology "Computers in Europe. Past, Present and Future", Kyiv, October 5-9, 1998
http://www.icfcst.kiev.ua/Symposium/Proceedings2/Bigdan.pdf