# Вопросы организации распределенного хранения данных в системах обработки изображений

*Д.Н. Волков*

## *1. Введение*

Фотограмметрические системы (ФГС) предназначены для построения цифровых моделей местности или поверхности объекта по его изображениям. В настоящее время в большинстве используемых систем этого типа работа осуществляется операторами на рабочих станциях с использованием различных средств автоматизации этого процесса, кроме того на некоторых стадиях обработки могут использоваться полностью автоматические методы. Однако основную долю времени составляет ручная обработка данных, поэтому при проектировании таких систем большое внимание уделяется проблемам организации одновременной обработки различных порций данных разными операторами.

Обработка информации в цифровых фотограмметрических системах (ЦФГС) имеет ряд особенностей, связанных как со значительными объемами обрабатываемых данных, так и с особенностями технологического процесса и его отдельных этапов. Кроме того, к таким системам могут предъявляться особые требования по надежности, используемой аппаратной базе и необходимым вспомогательным функциям.

Использование для хранения данных в ЦФГС файловых систем общего назначения напрямую представляется нецелесообразным, так как при этом:

- невозможно эффективно организовать работу нескольких программных модулей с данными, хранящимися в одном файле (с учетом семантики данных);
- затруднительно обеспечить безопасный доступ к данным по сети;
- отсутствуют средства организации размещения данных с учетом частоты обращений к ним по сети
- повышается трудоемкость администрирования системы и снижается эффективность использования пространства на дисках рабочих станций, а также повышаются затраты на передачу данных по сети.



В связи с этим оптимальным решением является отказ от непосредственного использования файловых систем общего назначения, взамен чего встает задача разработать собственные методики организации хранения данных. При этом становится возможным учесть особенности работы системы и предъявляемые к ней требования, тем самым повысив эффективность хранения и доступа к данным.

## *2. Требования, предъявляемые к системе*

На основе опыта разработки и эксплуатации предыдущих версий фотограмметрической системы, а также с учетом пожеланий пользователей, были выдвинуты следующие требования к модулю распределенного хранения данных (МРХ):

1. Организовать возможность хранения обрабатываемых данных на накопителях различных машин локальной сети.
2. Предусмотреть возможность доступа к данным на работающих компьютерах в то время, когда другие компьютеры могут быть недоступны.
3. Осуществлять доступ к данным на машине после ее включения без дополнительного административного вмешательства.
4. Иметь возможность получения информации о местоположении данных, хранящихся на недоступной в данный момент машине.
5. Организовать защиту данных от их модификации (в том числе по сети) средствами, отличными от системных.
6. Хранить данные так, чтобы затраты на передачу их по сети при обращениях были минимальными.
7. Создать средства автоматизированного управления расположением данных для обеспечения выполнения требований предыдущего пункта.
8. Ввести ряд средств управления системой, в том числе: контроль свободного пространства в сети, выполнение операций резервного копирования и восстановления, переноса данных внутри сети или операции импорта/экспорта с другими системами.

Требование 1 вызвано тем, что обрабатываемые данные могут иметь значительный размер (до десятков и сотен гигабайт), а их обработка должна вестись на различных рабочих станциях. Централизованное хранение подобных данных потребует использование накопителей сверхбольшой емкости (ограничение на аппаратную платформу) и резко увеличит нагрузку на сеть, снизив скорость доступа к данным.

Требования 2 – 5 связаны с обеспечением надежности работы в условиях локальной сети предприятия и сформулированы на основании опыта работы пользователей и по их требованиям. В частности, пункт 5 означает, что следует ограничить доступ к данным по сети через стандартные средства ОС (Windows Explorer, различные навигационные оболочки).

Требование 6 очевидным образом следует из стремления повысить скорость работы на имеющейся аппаратной базе.

Требования 7 – 8 соответствуют запросам системных администраторов по контролю и обслуживанию системы.

## 3. Некоторые сведения о существующих системах

В ЦФГС «AeroSys» и «VirtuoZo» совместная работа с проектом реализована на уровне разделения проекта на независимые фрагменты, работа с которыми проводится раздельно. При этом, в заключительной части работ происходит объединение результатов индивидуальной работы с каждой частью проекта. Недостатками такого подхода является, в первую очередь, невозможность предварительной оценки итогового результата и большие трудозатраты при необходимости коррекции конечного продукта после сборки данных.

В системе «ZIimaging ImageStation» совместная работа организована через общий доступ к файлам по сети. При этом каждый модуль устанавливает ловушки на событие «запись в файл», и по срабатыванию этой ловушки происходит чтение данных, записанных другими модулями. При этом возникают существенные системные расходы времени и ресурсов на организацию вышеупомянутых ловушек, а также снижается надежность системы, поскольку каждый модуль самостоятельно контролирует изменения в общем файле данных, которые ему необходимо внести. Возникают дополнительные накладные расходы на создание и хранение в файле синхроинформации, чтобы каждый модуль мог определить, какие изменения были внесены другими модулями. Ряд операций, которые требуют возможности отката (носят транзакционный характер) при таком подходе труднореализуемы или вообще невозможны.

Все вышеперечисленные системы используют для хранения данных сервис, предоставляемый файловыми системами базовых ОС, расширенные средствами сетевого доступа к файлам ОС. Существующие в настоящее время сетевые файловые системы (ФС) общего назначения организованы по клиент-серверной архитектуре, при которой машина, хранящая на своих накопителях данные (сервер) предоставляет машине, желающей получить доступ к этим данным (клиенту) определенный интерфейс, через который происходит передача команд-запросов и данных. Одной из основных задач при сетевом доступе к данным является унификация интерфейса прикладных программ (API) работы с удаленными файлами. Большинство современных ОС решают ее путем включения каталогов поставляемой сервером части сетевой файловой системы в локальную файловую систему, как часть последней. При этом возникает так называемая «точка привязки» (mount point) для экспортируемой ФС – это каталог локальной ФС, который является корневым каталогом экспортируемой. Как следует из определения, точка привязки своя для каждого клиента, импортирующего одну и ту же удаленную ФС. Для прикладной программы каталоги и файлы удаленной ФС будут подкаталогами точки



привязки и файлами в них. Каждый компьютер может быть и клиентом и сервером, экспортируя элементы своей ФС и импортируя удаленные.

Для передачи данных по сети ФС используют существующие протоколы транспортного уровня (TCP, IPX), а операционная система предоставляет набор функций (API) для открытия и закрытия доступа по сети к каталогам локальной файловой системы, а также для подключения и отключения ресурсов, открытых для доступа с удаленных компьютеров.

Используя эти средства ОС можно организовать доступ модулей ФГС к данным, расположенных на других машинах в сети. Однако этот подход (использовавшийся в некоторых предыдущих версиях системы) имеет ряд недостатков:

- данные становятся доступными для любого приложения, не только системы;
- различные средства разграничения доступа к сетевым ресурсам доступны не во всех версиях ОС и усложняют настройку системы;
- работоспособность системы может быть нарушена, если пользователь изменит настройки доступа или самостоятельно, в обход средств системы, изменит хранящиеся данные (удалит или отредактирует файлы).

Другим подходом может быть использование готового средства распределенного доступа к данным, например распределенной СУБД. Этот способ был отвергнут по следующим причинам:

- для обеспечения функционирования системы на базе СУБД, в соответствии с всеми предъявляемыми требованиями, понадобится разработать надстройку, по объему порядка целого модуля хранения данных;
- затраты на администрирование СУБД существенно повысят трудоемкость (снизят надежность) обслуживания системы;
- за счет использования СУБД повысятся требования к аппаратуре, возрастет стоимость аппаратно-программного комплекса;
- использование дополнительных продуктов может оказаться неприемлемым по лицензионным соображениям или требованиям безопасности.

## 4. Общая структура модуля распределенного хранения данных

МРХ структурно состоит из следующих служб:

**Служба каталогов** – осуществляет хранение информации о расположении данных, позволяя получить некоторую справочную информацию о данных без непосредственного обращения к самим данным. В силу того, что подобная справочная информация имеет относительно малый размер по сравнению с размером данных, ее представляется возможным хранить на каждом

компьютере системы (кэшировать), благодаря чему запрос этой информации будет происходить без сетевых запросов, что повысит скорость выполнения этих запросов. Кроме этого реализуется требование по возможности получения справочной информации о данных, недоступных в текущий момент (выключенный компьютер). Для того, чтобы кэшированная информация поддерживалась в актуальном состоянии, предусмотрены механизмы ее синхронизации.

**Служба сетевого доступа к данным** – позволяет получить доступ к данным, хранящимся на других компьютерах, через собственный сетевой протокол. При этом используется сервис, предоставляемый модулем сетевого доступа, являющимся независимой частью ЦФГС. Последний модуль служит для организации сетевых соединений между модулями ЦФГС, запущенными на различных рабочих станциях, он использует протокол транспортного уровня (TCP) операционной системы, делая все остальные модули независимыми от реализации сетевого протокола на конкретной платформе.

**Служба хранилищ** – организует хранение данных в обособленных объектах – хранилищах. Каждое хранилище имеет собственную структуру каталогов, в котором располагаются файлы с данными, и может быть перенесено на другую машину без изменения конфигурации всей сетевой системы. Каждое хранилище имеет собственный каталог с информацией о хранящихся в нем данных, который используется службой каталогов.

Далее будет подробно разобрана работа каждой из этих служб.

## *5. Организация службы каталогов*

Для доступа к данным необходимо ввести механизмы их идентификации. При этом они, во-первых должны быть удобными для написания программ с их использованием, а во-вторых не приводить к существенным накладным расходам на их работу.

С учетом этого была введена следующая логическая организация данных:

- структурной единицей хранения данных является *ресурс* (аналогичен файлу в ФС) – неинтерпретируемая последовательность байтов;
- каждый ресурс имеет *идентификатор* – строку, уникальную в пределах всей системы;
- вводится однозначное соответствие элементов множества ресурсов и элементов множества идентификаторов: каждому ресурсу соответствует идентификатор и наоборот;
- пространство идентификаторов ресурсов образует дерево (по соображениям иерархической упорядоченности хранимых данных) – Рис. 1.



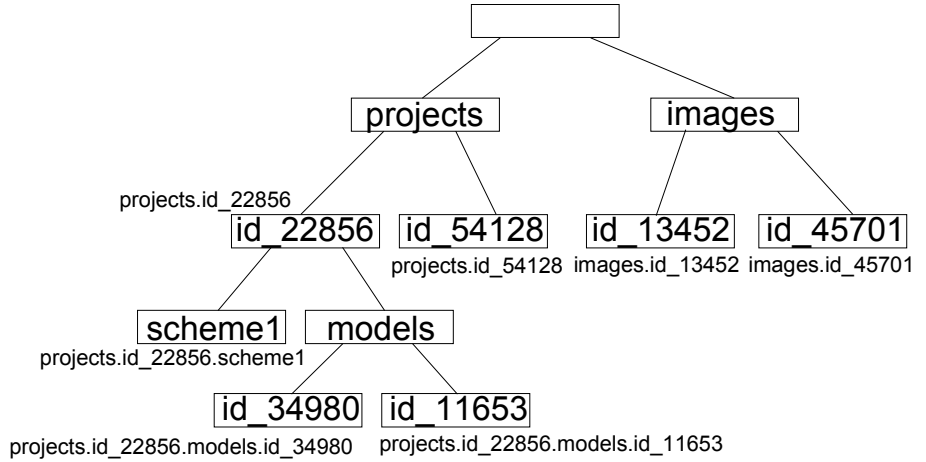


***Рис. 1.*** *Дерево ресурсов.*

Для упорядочивания физического хранения данных вводится понятие хранилища. *Хранилище* – это объект, содержащий данные ресурсов и информацию для обеспечения доступа к этим данным. Каждое хранилище также имеет свой уникальный строковый идентификатор. Собственно данные ресурсов хранятся в виде файлов локальной ФС. Это, правда, не исключает возможности хранения данных на удаленном компьютере с использованием средств сетевой ФС (Рис. 2). Однако при этом теряется ряд преимуществ доступа к данным через сетевые средства МРХ (в первую очередь, связанные с безопасностью), поэтому так следует поступать только при отсутствии других возможностей (на удаленном компьютере не установлены модули ФГС.

Для уменьшения вероятности случайного повреждения пользователем структуры данных все файлы данных хранилища целесообразно поместить в отдельный каталог в корневом каталоге диска, который сделать скрытым (путем установки атрибута или другим образом, в зависимости от ФС). Использование более сильных способов защиты данных, например, путем создания раздела на жестком диске собственного формата, неприемлемо из-за снижения универсальности (более низкий уровень доступа к аппаратным ресурсам потребует разработки значительного объема дополнительных средств, драйверов и т.п.) и надежности (повышается трудоемкость восстановления данных при отказах аппаратуры или ошибках ПО).

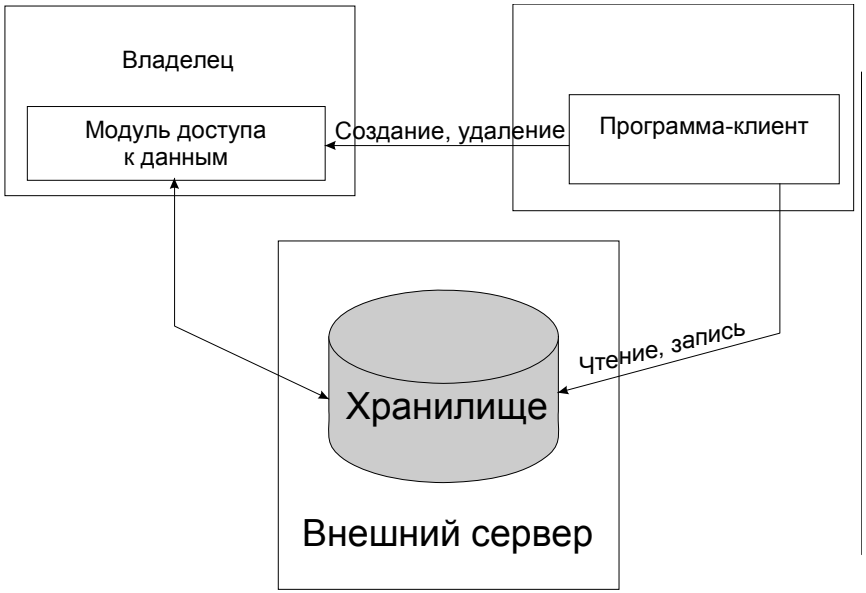


***Рис. 2.*** *Доступ при отсутствии на компьютере модулей ФГС.*

В разрабатываемой версии MPX каждому ресурсу соответствует один отдельный файл. Это позволяет осуществлять доступ к локальным (для приложения) ресурсам напрямую через стандартные функции ОС, без промежуточных обработок данных, максимально повысив производительность. При необходимости разделения некоторого блока данных (например, большого изображения) на несколько ресурсов для хранения их на разных компьютерах, это производится модулями, непосредственно использующими данные, с учетом специфики каждого типа данных.

Для доступа к данным ресурса по его идентификатору необходимо осуществить преобразование идентификатора ресурса в пару «идентификатор хранилища – имя файла». Это преобразование является одной из основных операций в MPX, ее выполняет служба каталогов MPX.

Каждое хранилище имеет *каталог* – таблицу, содержащую всю информацию о хранящихся ресурсах (идентификатор, имя файла с данными, синхронизационные метки, описание для пользователя и др.). Каталог рассылается всем машинам сети, таким образом служба каталогов на каждой машине формирует свое дерево ресурсов, посредством которого осуществляется преобразование идентификатора ресурса для доступа к данным.



Исходя из требования по возможности иметь доступ к информации о ресурсах, хранящихся на компьютерах, к которым нет доступа в настоящий момент, служба каталогов должна хранить последнее состояние каждого хранилища до момента перехода его в недоступное состояние. Это реализуется путем хранения на каждой машине *кэша каталога хранилища* копии каталога, полученного при последнем запросе. При невозможности доступа к хранилищу используются данные из кэша, при этом ресурсы данного хранилища помечаются, как недоступные, а также можно узнать, на какой машине это хранилище было доступным последний раз (какую машину нужно включить для обеспечения доступа к ресурсам).

Такой подход позволяет обеспечить простоту перемещения ресурсов из хранилища в хранилище, и хранилищ с машины на машину, причем эти изменения не отразятся на модулях верхнего уровня, идентификаторы ресурсов не изменятся, и доступ будет осуществляться единообразно, независимо от изменений в расположении данных. Это дает возможность оптимизировать расположение данных по доступу к ним, не испытывая дополнительных накладных расходов по изменению конфигурации модулей верхнего уровня.

## *6. Выполнение операций доступа к ресурсам*

В системе введено понятие «владелец хранилища». Им становится компьютер, который имеет доступ на запись для данного хранилища (локальный или через системные средства сетевого доступа к файлам) и первым произвел операцию захвата хранилища. В первую очередь происходит захват локальных хранилищ (доступных на локальной ФС), затем список захваченных хранилищ рассылается всем остальным машинам, чтобы они имели информацию о расположении того или иного хранилища. В процессе работы владелец хранилища может изменяться, если произошло выключение или сбой в работе предыдущего владельца. Таблица соответствия идентификатора хранилища и машины, ответственной за него, динамически изменяется с учетом информации, приходящей по сети.

Владелец осуществляет полный контроль над каталогом хранилища (все операции чтения или модификации каталога производятся только через него), создание новых ресурсов, рассылку каталога хранилища другим компьютерам и вспомогательные операции. Так достигается синхронизация всех операций с каталогом хранилища, а также осуществляется доступ к данным по сети.

Хранилище может быть доступно через средства ОС сетевого доступа к файлам. В таком случае удаленные машины могут обращаться к ресурсам хранилища через эти средства, открывая соответствующие файлы на чтение и/или запись (Рис. 3).

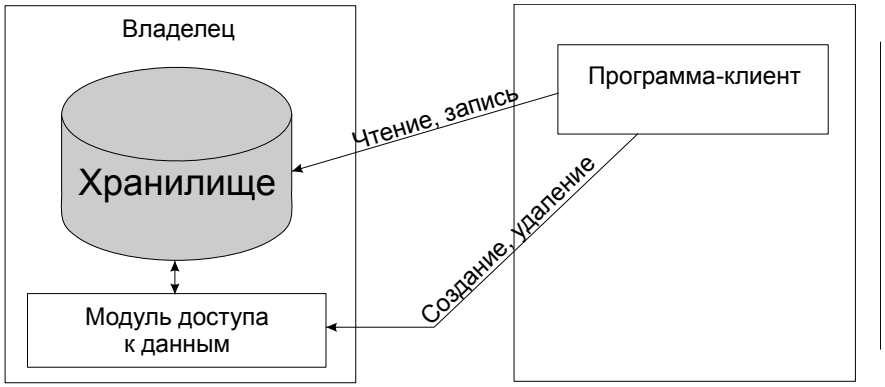


***Рис. 3.*** *Доступ через средства сетевой ФС.*

Возможна ситуация, когда хранилище располагается на диске сервера, на котором не установлена ФГС. Тогда это хранилище делается доступным по сети средствами ОС, один или несколько компьютеров имеют к нему доступ через эти средства. Один из них при старте системы становится владельцем хранилища, управляя каталогом хранилища, а все они обращаются к файлам хранилища через сетевую ФС (Рис. 2).

## *7. Методика уникальности имен. Создание и удаление ресурсов*

Для организации иерархической структуры ресурсов идентификатор ресурса имеет следующий формат: **сегмент_1.сегмент_2.** … **.сегмент_n**, где **сегмент_i** – последовательность алфавитно-цифровых символов. При этом все множество идентификаторов ресурсов системы естественным образом образует дерево. Введя понятие *узла дерева ресурсов*, необходимо заметить, что не каждому узлу соответствует ресурс. Этим данная система именования отличается от традиционной системы каталогов ФС – в нашем случае каталоги отсутствуют, а каждый ресурс именуется своим полным идентификатором без привязки к какому-либо другому ресурсу (Рис. 1 – возле узлов дерева, которым соответствуют ресурсы, указаны их полные идентификаторы). Такая организация используется для группировки ресурсов по функциональному назначению (например, ресурсы, содержащие данные для одного проекта, имеют общий корень, при этом физически могут располагаться на разных машинах), а также для некоторых операций управления ресурсами.

Поскольку для пользователя было бы неудобно ссылаться на ресурсы по идентификатору, который определяется модулем хранения данных и не может



быть произвольно редактируемым, были введены описания ресурсов – произвольные текстовые строки, хранящиеся в каталоге ресурсов хранилища вместе с идентификаторами. Пользователь видит описание при выборе ресурса в диалогах, а также имеет возможность ввести описание для вновь создаваемого ресурса.

Ресурсы можно разделить на две группы по следующему признаку: часть из них должна иметь фиксированный идентификатор (в рамках всей системы или в какой-либо логической группе, например, в поддереве проекта), тогда как другие могут иметь произвольный идентификатор в заданном поддереве. Ресурсы первого типа, как правило, содержат данные, определяемые в единственном экземпляре на проект (например, каталог изображений), а второго – данные, возникающие при обработке в нескольких вариантах (к примеру, разные варианты построения модели рельефа, из которых впоследствии будет выбрана одна). Для создания ресурсов первого типа модулю хранения данных передается заданный фиксированный идентификатор, при создании ресурса второго типа – специальный идентификатор, в котором один из сегментов (обычно последний) задается символом **“*”**. При создании ресурса модуль автоматически заменит звездочку строкой, формируемой по специальным правилам, так, чтобы в результате получился уникальный идентификатор. Этот идентификатор возвращается в код, запросивший создание ресурса, и используется при последующем доступе к нему. Полученный идентификатор можно использовать для создания ресурсов более низких уровней, добавляя в конец строки идентификатора дополнительные сегменты.

Учитывая возможность хранилищ находиться в недоступном состоянии произвольным образом, возникает следующая трудность при организации доступа к ресурсам с фиксированным идентификатором: при поступлении запроса на создание такого ресурса необходимо достоверно знать, существует он (возможно, в хранилище, на данный момент недоступном), или еще не был создан и должен быть создан при обработке текущего запроса. Одним из вариантов метода решения данной задачи можно предложить, например, ведение отдельной таблицы всех ресурсов с фиксированными идентификаторами. Этот вариант неудобен тем, что требует наличия выделенной, всегда включенной, машины и по другим причинам. В рассматриваемой системе был использован другой подход, названный «методом ответственных».

По сути, этот метод реализует вышеупомянутую таблицу уникальных ресурсов, только в распределенном варианте. На каждый узел дерева может быть навешен специальный атрибут «ответственный», имеющий значением идентификатор (имя) компьютера, который является ответственным за создание ресурсов с фиксированными идентификаторами в этом и всех нижележащих узлах дерева. Атрибут на нижележащем узле перекрывает более

верхний. Таким образом, все дерево идентификаторов ресурсов оказывается поделенным на зоны ответственности между компьютерами сети (Рис. 4).

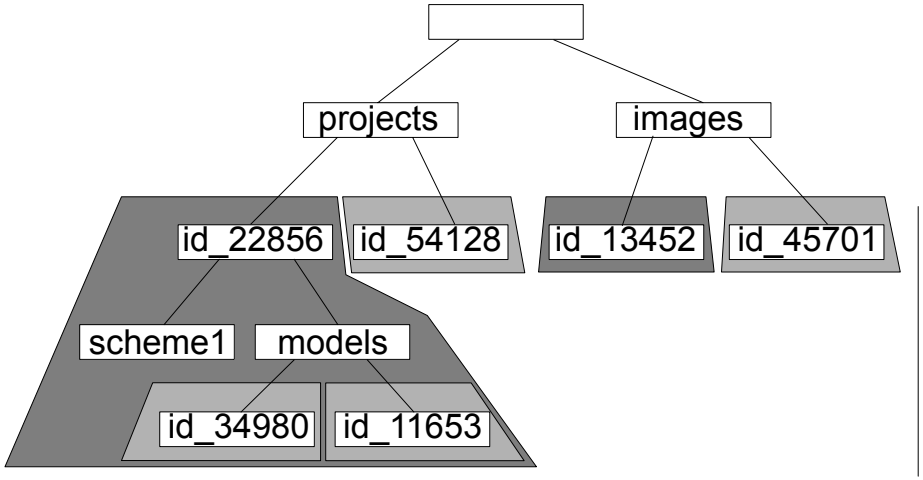


***Рис. 4.*** *Зоны ответственности в дереве ресурсов.*

Рассмотрим алгоритмы создания ресурса и назначения узлу ответственного, поскольку они тесно связаны друг с другом. При запросе на создание ресурса с фиксированным идентификатором, определяется компьютер, ответственный за данную область дерева идентификаторов. Если такой компьютер найден и доступен в данный момент (включен), у него получается разрешение на создание ресурса. В ответ может быть получено разрешение, либо сообщено, что указанный ресурс уже существует и выдан идентификатор хранилища с этим ресурсом (при этом хранилище может быть сейчас недоступно – используется кэш хранилища, находящийся на ответственном компьютере). Если ресурс пока не был создан, он создается в выбранном хранилище, причем операция считается завершенной только после того, как ответственный синхронизирует это хранилище и сохранит его кэш на локальном диске. Такой алгоритм гарантирует, что ресурс с фиксированным идентификатором будет создан в системе в единственном экземпляре.

При создании ресурса с уникальным идентификатором соответствующий ему узел получает атрибут «ответственный» с указанием создающего компьютера или компьютера, который владеет хранилищем ресурса. Это не обязательно, можно использовать и существующую в точке создания ресурса область ответственности, однако такой подход делает дерево идентификаторов разделенным на большее число областей ответственности с их локализацией (ответственным за ресурсы делается компьютер, владеющий хранилищем с этими ресурсами), а следовательно уменьшает среднее число отказов на создание ресурсов по причине отсутствия доступа к какому-либо компьютеру.



Как видно из рисунка, некоторые узлы верхнего уровня не лежат в зоне ответственности никакого компьютера. Это означает, что ресурсы с фиксированными идентификаторами не могут быть созданы в данных узлах. Однако это и не требуется в рамках общей концепции ресурсов рассматриваемой СОИ, а при создании ресурса с уникальным идентификатором автоматически, как следует из только что рассмотренного алгоритма, будет создана новая зона ответственности. Также видно, что все ресурсы верхнего уровня создавались с уникальными идентификаторами.

При удалении ресурса также встает проблема синхронизации (в случае ресурсов с фиксированными идентификаторами). Однако требование – для удаления ресурса, к файлу которого есть физический доступ, необходимо дополнительно иметь доступ к третьему компьютеру (ответственному) – кажется излишне жестким. Поэтому удаление ресурса производится в два этапа: на первом удаляется файл ресурса из хранилища, на втором происходит удаление записей из каталога хранилища, которым не соответствуют файлы в хранилище. Первый этап не требует участия владельца хранилища, что ускоряет данную операцию, особенно при групповом удалении. Рассмотрим подробнее второй этап. Он выполняется владельцем хранилища по запросу (после удаления ресурсов), при начальном запуске и периодически. Для каждого ресурса в хранилище, файл которого отсутствует, определяется ответственный, удаляется запись из каталога, после чего ответственному направляется команда на синхронизацию данного хранилища.

Операция удаления алгоритмически проще операции создания и содержит меньшее число проверок. Это объясняется тем, что при сбоях синхронизации на этапе удаления, до восстановления синхронности каталога хранилища с локальными кэшами, будет невозможно (отказ) создание ресурсов с фиксированными идентификаторами, а при подобных сбоях в процессе создания – образование двух ресурсов с одинаковыми идентификаторами, сохранение в них различных данных и необходимость вмешательства администратора для объединения этих ресурсов в один. Последнее, кстати, не всегда возможно, связано с большими затратами времени и может привести к потере данных. Все это указывает на значительно большую важность синхронизации при создании и необходимости выполнения этой операции более тщательно.

## *8. Синхронизация*

Как уже неоднократно упоминалось, синхронизация, в основном синхронизация каталогов хранилищ, является одной из основных служебных операций в модуле распределенного хранения данных. При синхрооперациях объект хранилища, расположенный на машине-владельце (первичный), взаимодействует с аналогичными объектами (вторичными) на удаленных машинах (Рис. 5). Объекты, соответствующие одному хранилищу, связываются между собой через сетевые службы системы по протоколу TCP. Данные

первичного объекта всегда имеют статус авторизованных, тогда как данные вторичных объектов используются для ускорения справочных запросов и должны быть изменены при приходе данных от первичного объекта.

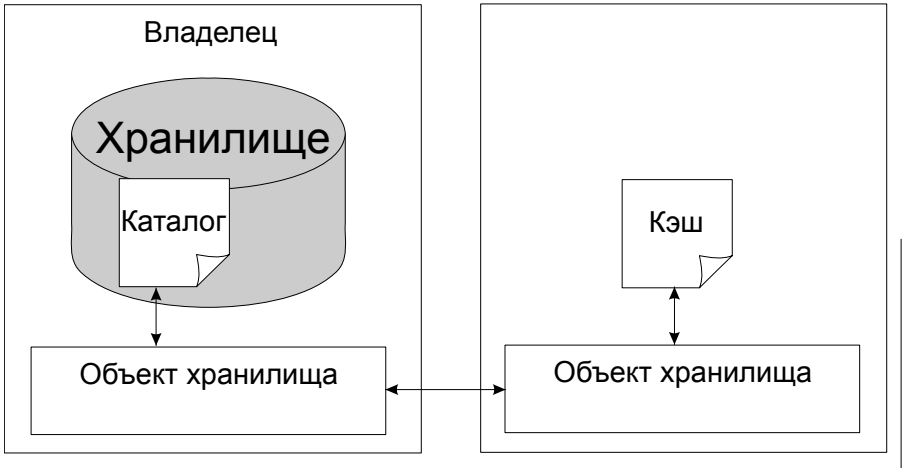


***Рис. 5.*** *Взаимодействие объектов хранилища.*

Основными данным здесь является таблица каталога хранилища. В первичном объекте хранилища она всегда (с точностью до задержек записи на диск) согласована с таблицей каталога хранилища, хранящейся в виде файла в том же каталоге, что и хранилище. При изменениях этой таблицы первичный объект направляет все доступным вторичным объектам сообщение о необходимости пересинхронизировать таблицу ресурсов. Вторичные объекты самостоятельно решают, когда выполнить эту операцию, в зависимости от загрузки системы и необходимости доступа к ресурсам из данного хранилища. Однако в любом случае объект отмечает, что его данные устарели, установкой соответствующего флага.

Для уменьшения объема запросов, передаваемых по сети, вводится понятие временной метки (timestamp) для каталога хранилища. Эти метка представляет собой число, которое каждый раз увеличивается на единицу при соответствующей модификации хранилища. Таких меток две: для последней модификации (создания, удаления, изменения атрибутов ресурса) и для последнего удаления. Эта пара меток существует для каталога хранилища на диске, в памяти, для кэша каталога. Кроме того, для каждого ресурса существует временная метка его последнего изменения его атрибутов (не содержимого!). При модификации хранилища его метка последней модификации увеличивается, после чего полученное значение присваивается метке ресурса, который вызвал изменения в хранилище. В случае удаления какого-либо ресурса из каталога хранилища, временная метка удаления получает значение обновленной метки изменения.



Если метка изменения у каталога на диске или кэша меньше, чем у таблицы в памяти, производится запись на диск (может быть отложена для уменьшения числа дисковых операций). При синхронизации по сети вторичный объект передает в запросе метки модификации и удаления своей таблицы. При этом:

- если метка модификации вторичного объекта *совпадает* с меткой модификации первичного, хранилища синхронны (возвращается пустая таблица изменений);
- если метка модификации вторичного объекта *меньше* метки модификации первичного, а метки удаления *совпадают*, то возвращается таблица изменений, содержащая записи каталога, с метками изменения *большими*, чем метка изменения вторичного объекта;
- если метка удаления вторичного объекта *меньше*, чем у первичного, возвращается вся таблица каталога хранилища.

Программно таблица каталога хранилища отдельно в памяти не хранится. Вместо этого каждый модуль распределенного хранения имеет в памяти общее дерево ресурсов, упорядоченных в соответствии с их идентификаторами, в котором каждый узел, в том случае, если ему соответствует ресурс, имеет указатель на объект (в терминах C++) хранилища, содержащего этот ресурс. Полный обход по дереву позволяет получить полную таблицу ресурсов для заданного хранилища. Эта операция, однако, относительно редка, по сравнению с операциями поиска заданного ресурса (как операция создания нового ресурса редка по сравнению с открытием существующего).

При каждой синхронизации хранилища происходит проход по дереву ресурсов и его модификация в соответствии с полученной таблицей изменений от первичного объекта хранилища. Используемые средства синхронизации позволяют не блокировать дерево на исключительный доступ в процессе этой операции, что дает возможность одновременно получать справочную информацию о ресурсах, необходимую для доступа к ним.

## *9. Заключение*

В данной работе рассматривались особенности хранения данных в цифровых фотограмметрических системах, недостатки использования для решения данной задачи файловых систем общего назначения. В результате был определен набор требований к модулю хранения данных, определена функциональная структура данного модуля и выполнена его реализация, которая в настоящее время проходит тестирование в опытной эксплуатации.

Следующим этапом работы по организации хранения данных будет разработка методики оптимизации размещения данных и создания средств для управления размещением данных в соответствии с этой методикой.