МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (СГАУ)

Архитектура современных распределённых систем

Работа выполнена по мероприятию блока 1 «Совершенствование
образовательной деятельности» Программы развития СГАУ
на 2009 – 2018 годы по проекту «Разработка магистерской программы «Программное обеспечение
мобильных устройств по направлению 230100.68 Информатика и вычислительная техника»
Соглашение № 1/12 от 3 июня 2013 г.

САМАРА
2013

УДК 004.272 (075) ББК 32.973.202 я 7
А 878

Автор-составитель: **Востокин Сергей Владимирович**

Архитектура современных распределённых систем [Электронный ресурс] : электрон. учеб.-метод. комплекс по дисциплине в LMS Moodle / Мин-во образования и науки РФ, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т); авт.-сост. С. В. Востокин. - Электрон. текстовые и граф. дан. - Самара, 2013. – 1 эл. опт. диск (CD-ROM).

В состав учебно-методического комплекса входят:

1. Курс лекций.
2. Учебное пособие.
3. Задания на лабораторные работы.
4. Задания на практические работы.
5. Темы для подготовки к экзамену.
6. Тесты для итогового контроля знаний.
7. Рабочая программа.

УМКД «Архитектура современных распределительных систем» предназначен для студентов факультета информатики, обучающихся по направлению подготовки магистров 230100.68 «Информатика и вычислительная техника» в А семестре.

УМКД разработан на кафедре Информационные системы и технологии.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)**

Курс лекций по дисциплине «Архитектура современных распределенных систем»

направление 230100.68 – «Информатика и вычислительная техника»
(магистратура)

Факультет информатики
Кафедра информационных систем и технологий

Разработал:
Востокин С.В.,
профессор кафедры ИСТ

Самара 2013 г.

Лекция 1. Введение в распределённые системы

План лекции

Определение распределенной системы

Задачи распределенных систем

Концепции аппаратных решений

Концепции программных решений

Модель клиент-сервер

Обзор лекции

Компьютерные системы претерпевают революцию. С 1945 года, когда началась эпоха современных компьютеров, до приблизительно 1985 года компьютеры были большими и дорогими. Даже мини-компьютеры стоили сотни тысяч долларов. В результате большинство организаций имели в лучшем случае лишь несколько компьютеров, и, поскольку методы их соединения отсутствовали, эти компьютеры работали независимо друг от друга.

Однако в середине восьмидесятых под воздействием двух технологических новинок ситуация начала меняться. Первой из этих новинок была разработка мощных микропроцессоров. Изначально они были 8-битными, затем стали доступны 16-, 32- и 64-битные процессоры. Многие из них обладали вычислительной мощностью мэйнфреймов (то есть больших компьютеров), но лишь частью их цены.

Скорость роста, наблюдавшаяся в компьютерных технологиях в последние полвека, действительно потрясает. Ей нет прецедентов в других отраслях. От машин, стоивших 100 миллионов долларов и выполнявших одну команду в секунду, мы пришли к машинам, стоящим 1000 долларов и выполняющим 10 миллионов команд в секунду. Разница в соотношении цена/производительность достигла порядка 10^{12}. Если бы автомобили за этот период совершенствовались такими же темпами, «роллс-ройс» сейчас стоил

бы один доллар и проходил миллиард миль на одном галлоне бензина (к сожалению, к нему потребовалось бы 200-страничное руководство по открыванию дверей).

Второй из новинок было изобретение высокоскоростных компьютерных сетей. Локальные сети (Local-Area Networks, LAN) соединяют сотни компьютеров, находящихся в здании, таким образом, что машины в состоянии обмениваться небольшими порциями информации за несколько микросекунд. Большие массивы данных передаются с машины на машину со скоростью от 10 до 1000 Мбит/с. Глобальные сети (Wide-Area Networks, WAN) позволяют миллионам машин во всем мире обмениваться информацией со скоростями, варьирующимися от 64 кбит/с (килобит в секунду) до гигабит в секунду.

В результате развития этих технологий сегодня не просто возможно, но и достаточно легко можно собрать компьютерную систему, состоящую из множества компьютеров, соединенных высокоскоростной сетью. Она обычно называется компьютерной сетью, или распределенной системой (distributed system), в отличие от предшествовавших ей централизованных (centralized systems), или однопроцессорных (single-processor systems), систем, состоявших из одного компьютера, его периферии и, возможно, нескольких удаленных терминалов.

Распределенные системы состоят из автономных компьютеров, которые работают совместно, представая в виде единой связной системы. Их важное преимущество состоит в том, что они упрощают интеграцию различных приложений, работающих на разных компьютерах, в единую систему. Еще одно их преимущество — при правильном проектировании распределенные системы хорошо масштабируются. Их размер ограничивается только размером базовой сети. Платой за эти преимущества часто является очень сложное программное обеспечение, падение производительности и особенно проблемы с безопасностью. Тем не менее, заинтересованность в построении и внедрении распределенных систем наблюдается повсеместно.

Существуют различные типы распределенных систем. Распределенные операционные системы используются для управления аппаратным обеспечением взаимосвязанных компьютерных систем, к которым относятся мультипроцессорные и гомогенные мультикомпьютерные системы. Эти распределенные системы на самом деле не состоят из автономных компьютеров, но успешно воспринимаются в виде единой системы. Сетевые операционные системы, с другой стороны, с успехом объединяют различные компьютеры, работающие под управлением своих операционных систем, так что пользователи с легкостью могут получать доступ к локальным службам каждого из узлов. Однако сетевые операционные системы не создают ощущения работы с единой системой, которое характерно для распределенных операционных систем.

Современные распределенные системы обычно содержат поверх сетевой операционной системы дополнительный уровень программного обеспечения. Этот уровень, называемый промежуточным, предназначен для того, чтобы скрыть гетерогенность и распределенную природу базового набора компьютеров. Распределенные системы с промежуточным уровнем обычно требуют специфическую модель распределения и связи. Известные модели основаны на удаленном вызове процедур, а также на распределенных объектах, файлах или документах.

Для каждой распределенной системы важна схема ее внутренней организации. Широко применяется модель, в которой процессы клиента запрашивают службы у процессов сервера. Клиент посылает на сервер сообщение и ожидает, пока тот вернет ответ. Эта модель тесно связана с традиционным программированием, в котором службы реализуются в виде процедур в отдельных модулях. Дальнейшее уточнение обычно состоит в подразделении на уровень пользовательского интерфейса, уровень обработки и уровень данных. Сервер обычно отвечает за уровень данных, а уровень пользовательского интерфейса реализуется на стороне клиента. Уровень

обработки может быть реализован на клиенте, на сервере или поделен между ними.

В современных распределенных системах для построения крупномасштабных систем такой вертикальной организации приложений модели клиент-сервер недостаточно. Необходимо горизонтальное распределение, при котором клиенты и серверы физически распределены и реплицируются на несколько компьютеров. Типичным примером успешного применения горизонтального распределения является World Wide Web.

Лекция 2. Связь

План лекции

Уровни протоколов

Удаленный вызов процедур

Обращение к удаленным объектам

Связь посредством сообщений

Связь на основе потоков данных

Обзор лекции

Связь между процессами — это суть распределенных систем. Нет смысла изучать распределенные системы, не рассматривая при этом во всех подробностях способы обмена информацией между различными процессами, выполняющимися на разных машинах. Взаимодействие в распределенных системах всегда базируется на низкоуровневом механизме передачи сообщений, предоставляемом базовой сетью. Как мы обсуждали в предыдущей лекции, реализация взаимодействия через передачу сообщений сложнее, чем использование примитивов на базе разделяемой памяти. Современные распределенные системы часто включают в себя тысячи или даже миллионы процессов, разбросанных по ненадежной сети, такой как Интернет. Если не заменить простейшие средства взаимодействия в

компьютерных сетях чем-то иным, разработка масштабных приложений будет достаточно сложной.

Мы начнем эту лекцию с обсуждения правил, которых придерживаются сообщающиеся между собой процессы. Их обычно называют протоколами. Мы сосредоточимся на структурировании этих протоколов в виде уровней. Затем мы рассмотрим четыре широко распространенные модели взаимодействия: удаленный вызов процедур (Remote Procedure Call, RPC), удаленное обращение к методам (Remote Method Invocation, RMI), ориентированный на сообщения промежуточный уровень (Message-Oriented Middleware, MOM) и потоки данных (streams).

Нашей первой моделью взаимодействия в распределенных системах станет удаленный вызов процедур. Механизм RPC нацелен на сокрытие большей части проблем передачи сообщений и идеален для приложений архитектуры клиент-сервер. Усовершенствованный вариант модели RPC имеет вид удаленного обращения к методам, которое основано на представлении распределенных объектов. Механизмы RPC и RMI рассматриваются в отдельных разделах.

Во многих распределенных приложениях связь не ограничивается слегка урезанным шаблоном взаимодействий клиента и сервера. В подобных случаях мыслить категориями сообщений оказывается предпочтительнее. Однако применение разнообразных низкоуровневых средств связи компьютерных сетей приведет к серьезным нарушениям прозрачности распределения. Альтернативой им является высокоуровневая модель очереди сообщений, связь в которой очень напоминает системы электронной почты. Ориентированный на сообщения средний уровень — это достаточно важная тема, чтобы отвести отдельный раздел и на нее.

Что касается мультимедиа в распределенных системах, понемногу становится очевидным, что таким системам недостает поддержки передачи непрерывных потоков, таких как аудио или видео. Им необходимо понятие потока, который позволяет поддерживать непрерывно идущие сообщения в

соответствии с различными ограничениями по синхронизации. Потоки обсуждаются в последнем разделе этой лекции.

Наличие мощных и гибких механизмов взаимодействия между процессами является важным для всякой распределенной системы. В традиционных сетевых приложениях связь часто базируется на низкоуровневых примитивах передачи сообщений, предоставляемых транспортным уровнем. Важной особенностью систем промежуточного уровня является предоставляемая ими высокая степень абстракции, благодаря которой описание взаимодействия между процессами на промежуточном уровне значительно проще, чем если бы мы ограничились только интерфейсами транспортного уровня.

Одной из наиболее широко используемых абстракций является удаленный вызов процедур (Remote Procedure Call, RPC). Сущность RPC в том, что любая служба реализуется посредством вызова процедуры, тело которой выполняется на сервере. Клиент предоставляет только сигнатуру процедуры, то есть имя процедуры и ее параметры. Когда клиент вызывает процедуру, клиентская реализация, называемая заглушкой, упаковывает значения параметров в сообщение и пересылает его на сервер. Последний вызывает собственно процедуру и возвращает результат, снова в виде сообщения. Клиентская заглушка извлекает из этого сообщения значение результата и передает его приложению клиента, инициировавшему вызов.

Механизм RPC ориентирован на обеспечение прозрачности доступа. Однако он относительно слабо поддерживает передачу ссылок. В этом смысле удаленные объекты более прозрачны. Обращение к удаленным методам (Remote Method Invocation, RMI) напоминает RPC, но отражает специфику удаленных объектов. Основная разница между ними состоит в том, что RMI позволяет использовать в качестве параметров ссылки на объекты системы.

RPC и RMI предоставляют механизмы синхронной связи, при которой клиент блокируется до получения ответа от сервера. Несмотря на вариации

существующих механизмов, в которых жесткая синхронная модель смягчена, нередко более удобными оказываются универсальные высокоуровневые модели, ориентированные на передачу сообщений.

В моделях передачи сообщений неважно, является ли связь сохранной, так же как неважно и то, является ли она синхронной. Смысл сохранной связи состоит в том, что посылаемое сообщение хранится в коммуникационной системе до тех пор, пока не будет доставлено по назначению. Другими словами, ни отправитель, ни получатель при передаче сообщения не обязаны быть активными. В случае нерезидентной связи механизмы хранения не предусмотрены, а значит, получатель должен быть готов принять сообщение, когда бы оно ни было послано.

При асинхронной связи отправитель может продолжать работу сразу после установки сообщения в очередь на отправку, возможно, еще до того, как оно будет отправлено. При синхронной связи отправитель блокируется как минимум до момента получения сообщения. В других вариантах отправитель может блокироваться до момента доставки сообщения получателю или до получения от него ответа, как это сделано в RPC.

Модели обмена сообщениями промежуточного уровня обычно предоставляют сохранную асинхронную связь и используются там, где применение механизмов RFC и RMI не оправдано. В первую очередь это интеграция наборов баз данных (сильно распределенных) в крупных информационных системах. Другие области их применения включают в себя электронную почту и рабочие потоки.

Абсолютно иную связь предлагают потоки данных, проблема которых состоит в том, что любые два последовательных сообщения взаимосвязаны по времени. В непрерывных потоках данных максимальная задержка доставки своя для каждого сообщения. Кроме того, необходимо, чтобы сообщения обладали некоей минимальной задержкой доставки. Типичными примерами непрерывных потоков данных являются аудио- и видеопотоки. Часто бывает сложно описать, какими должны быть временные взаимосвязи

или чего мы ожидаем от базовой подсистемы связи (в терминах качества обслуживания). При реализации мы также сталкиваемся с затруднениями. Усложняющим фактором является роль дрожания (максимальное и минимальное значения параметров). Даже если средняя производительность достижима, серьезные колебания времени доставки могут привести к неприемлемой производительности.

Лекция 3. Процессы

План лекции

Потоки выполнения

Клиенты

Серверы

Перенос кода

Программные агенты

Обзор лекции

В предыдущей лекции мы обсуждали взаимодействие в распределенных системах. Взаимодействие происходит между процессами, и в этой лекции мы поближе рассмотрим то, какую роль играют в распределенных системах различные типы процессов. Концепция процесса зародилась в операционных системах, где этим понятием обычно обозначают выполняемую программу. С точки зрения операционных систем наиболее важные для обсуждения вопросы — это управление процессами и планирование процессов. Однако при переходе к распределенным системам другие вопросы становятся столь же или еще более важными.

Так, например, для эффективной организации систем клиент-сервер часто бывает удобно использовать многопоточные технологии. Как мы обсудим в первом разделе этой лекции, основной вклад потоков выполнения в работу распределенных систем заключается в том, что они позволяют

создавать клиенты и серверы так, что взаимодействие и локальная обработка выполняются параллельно, давая выигрыш в производительности.

Как мы говорили в лекции 1, модель клиент-сервер крайне важна для распределенных систем. В этой лекции мы детально рассмотрим основные типы организации клиентов и серверов. Мы также уделим внимание общим вопросам построения серверов. Кроме того, мы обсудим общедоступные серверы объектов, которые предоставляют базовые средства для построения распределенных объектов.

Серьезной проблемой, особенно в глобальных системах, является перенос процессов между различными машинами. Миграция процессов, или, более точно, миграция кода, может помочь повысить масштабируемость и, кроме того, помогает динамически конфигурировать клиенты и серверы. Что на самом деле подразумевается под миграцией кода и как она реализуется, также будет рассмотрено в этой лекции.

Последняя наша тема связана с относительно новым явлением — программными агентами. В отличие от асимметричной программной модели клиент-сервер, мультиагентные системы в основном состоят из набора одинаково важных агентов, совместная работа которых приводит к достижению общей цели. Программные агенты — это еще один тип процессов, которые могут существовать в различных формах. В последнем разделе рассказывается, что такое агенты с точки зрения распределенных систем и каковы принципы их совместной работы.

Процессы играют фундаментальную роль в распределенных системах, поскольку они формируют базис для связи между различными машинами. Важным вопросом является внутренняя организация процессов и в частности, способны ли они поддерживать несколько управляющих потоков выполнения. Потоки выполнения в распределенных системах используются, в частности, для продолжения работы с процессором во время блокирующих операций ввода-вывода. Таким образом, появляется возможность построения более эффективных серверов, в которых несколько потоков выполнения

работают одновременно, причем некоторые из них могут быть блокированы в ожидании выполнения дисковых операций ввода-вывода или операций сетевого взаимодействия.

Возможна организация распределенных приложений в понятиях клиентов и серверов. Клиентский процесс обычно реализует пользовательский интерфейс, который может варьироваться от простого вывода информации до расширенных интерфейсов, способных поддерживать составные документы. Клиентское программное обеспечение, кроме того, способно поддерживать прозрачность распределения, скрывая детали, касающиеся связи с серверами, текущего местоположения серверов и репликации серверов. Кроме того, программное обеспечение клиента способно частично скрыть возникающие сбои и процессы восстановления после сбоев.

Серверы часто сложнее клиентов, но тем не менее при их построении применяется относительно немного архитектурных моделей. Так, например, серверы могут быть итеративными или параллельными, реализовывать одну или несколько служб, сохранять информацию о состоянии или не сохранять. Остальные архитектурные особенности касаются адресации служб и механизмов прерывания серверов после прихода запроса на обслуживание и возможно в ходе его выполнения.

Серверы объектов выделяются в особый класс. Коротко говоря, сервер объектов — это процесс, содержащий размещенные в своем адресном пространстве объекты и готовый принимать направленные к ним обращения. В отдельную категорию серверы объектов выделяются во многом благодаря разнообразию способов обращения к объектам. Так, например, сервер может запустить отдельный поток выполнения для каждого запроса к объекту. С другой стороны, он может выделять каждому объекту собственный поток выполнения или оставить единственный поток выполнения для всех своих объектов. Посредством адаптера объектов в различных серверах может быть реализована разная политика обращения к объектам. Коротко говоря, адаптер

объектов — это компонент, реализующий только одну политику обращения. На сервере может быть несколько адаптеров объектов.

Важной для распределенных систем темой является перенос кода с машины на машину. Для того чтобы поддерживать перенос кода, имеются две веские причины — повышение производительности и мобильность. Если связь дорога, мы можем иногда уменьшить взаимодействие, перенеся вычисления с сервера на клиент и заставив клиент все что можно обрабатывать локально. Гибкость же возрастает, когда клиент имеет возможность динамически загружать программное обеспечение, необходимое для работы с конкретным сервером. Загруженное программное обеспечение может быть уже настроено на взаимодействие с этим сервером, избавляя клиента от необходимости повторно устанавливать его до начала работы.

Перенос кода приносит проблемы использования локальных ресурсов, связанные с тем, что эти ресурсы также необходимо переносить на другие машины, организовывать новые привязки кода к локальным ресурсам целевой машины или задействовать глобальные ссылки. Другая проблема заключается в том, что при переносе кода мы должны принимать во внимание гетерогенность системы. Текущая практика показывает, что, возможно, лучшим средством справиться с гетерогенностью являются виртуальные машины, которые эффективно скрывают гетерогенность с помощью интерпретируемого кода.

И, наконец, программные агенты — специальный вид процессов, работающих как автономные модули, но способных кооперироваться с другими агентами. С точки зрения распределенных систем, отличие агентов от обычных приложений в том, что агенты взаимодействуют друг с другом посредством коммуникационного протокола прикладного уровня, который называется языком взаимодействия агентов (ACL). В ACL имеется четкое разделение между целью сообщения и его содержимым. ACL определяет коммуникационный протокол верхнего уровня: посылка сообщения обычно

предполагает конкретную реакцию получателя на основании исключительно цели сообщения.

Лекция 4. Именование

План лекции

Именованные сущности

Размещение мобильных сущностей

Удаление сущностей, на которые нет ссылок

Обзор лекции

Имена играют важную роль во всех компьютерных системах. Они необходимы для совместного использования ресурсов, определения уникальных сущностей, ссылок на местоположения и т.д. Важная особенность именования состоит в том, что имя может быть разрешено, предоставляя доступ к сущности, на которую оно указывает. Разрешение имени, таким образом, представляет собой процесс доступа к именованной сущности. Для разрешения имен необходимо реализовать систему именования. Разница между именованием в распределенных и нераспределенных системах состоит в способе реализации систем именования.

В распределенных системах реализация системы именования часто сама по себе распределена по нескольким машинам. Способ этого распределения играет ключевую роль для эффективности и масштабируемости системы именования. В этой лекции мы сосредоточимся на трех различных, но одинаково важных способах использования имен в распределенных системах.

Во-первых, до обсуждения некоторых общих вопросов, связанных с именованием, мы поближе рассмотрим организацию и реализацию «человеческих» имен. Типичные примеры подобных имен включают в себя имена файловой системы и World Wide Web. Построение глобальной

масштабируемой системы именования имеет прямое отношение к этим типам имен.

Во-вторых, имена используются для локализации мобильных сущностей. Как оказывается, системы именования на основе «человеческих» имен не особенно подходят для поддержки большого количества мобильных сущностей, которые вдобавок могут быть разбросаны по большой сети. Необходима альтернативная организация, подобная той, что используется в мобильной телефонии, в которой имена (идентификаторы) не зависят от местоположения.

Наша третья и последняя тема будет касаться организации имен. В частности, имена, на которые больше не ссылается ни один объект и которые невозможно более локализовать, чтобы получить к ним доступ, должны автоматически удаляться. Этот процесс также известен как уборка мусора, корни его следует искать в языках программирования. Однако при переходе к крупным распределенным системам автоматическая уборка объектов, на которые нет ссылок, становится особенно важной.

Имена используются для ссылок на сущности. По сути, существует три основных типа имен. Адрес — имя точки доступа, ассоциированной с сущностью, часто называемый просто адресом сущности. Другой тип имени — идентификатор. Он имеет три свойства: каждая сущность имеет только один идентификатор, идентификатор указывает на единственную сущность и не может быть переназначен другой. И наконец, имена, удобные для восприятия, предназначены для использования людьми и представляют собой строку символов.

Имена организованы в пространства имен. Пространство имен может быть представлено в виде графа именования, в узлах которого расположены именуемые сущности, а метки на ребрах представляет собой имена, под которыми эти сущности известны. Узлы, из которых выходит несколько ребер, представляют собой наборы сущностей и известны также под

названием направляющих узлов. Крупномасштабные графы именования часто организуются в виде корневых ациклических направленных графов.

Графы именования подходят для структурирования имен, удобных для восприятия. Доступ к сущности может осуществляться посредством пути. Разрешение имен — это процесс прохода по графу именования в поисках компонентов, входящих в путь, по одному за раз. Крупномасштабный граф именования реализуется путем распределения узлов по нескольким серверам имен. При разрешении пути путем обхода графа именования, если разыскиваемый узел находится на следующем сервере имен, то разрешение имен продолжается на нем.

Системы именования для имен, удобных для восприятия, невозможно использовать для высокомобильных сущностей. Локализация мобильных сущностей более успешно может быть произведена с помощью не зависящих от местоположения идентификаторов. Существует четыре основных подхода к локализации мобильных сущностей.

Первый подход состоит в использовании широковещательных или групповых рассылок. Идентификатор сущности посылается широковещательной посылкой каждому процессу в распределенной системе. Процесс предлагает точку доступа в ответ на предоставление ему адреса этой точки доступа. Очевидно, что этот подход имеет ограниченную масштабируемость.

Второй подход состоит в пересылке указателей. Каждый раз при перемещении сущности в другое место она оставляет за собой указатель, информирующий о том, куда она переместилась. Локализация сущности требует обхода цепочки пересылаемых указателей. Чтобы устранить длинную цепочку указателей, важно уменьшить длину цепи после перемещения.

Третий подход состоит в создании базы сущности. Каждый раз при перемещении сущности в другое место она уведомляет об этом свою базу. При локализации сущности первым делом о текущей локализации запрашивается ее база.

Четвертый подход состоит в построении иерархического дерева поиска. Сеть разбивается на неперекрывающиеся области (домены). Домены могут группироваться в домены верхнего уровня (неперекрывающиеся) и т. д. Существует один домен верхнего уровня, охватывающий всю сеть. Всякий домен на любом уровне имеет ассоциированный с ним направляющий узел. Если сущность находится в домене Z), направляющий узел в домене следующего уровня содержит указатель на D. Направляющие узлы самого нижнего уровня содержат адреса сущностей. Направляющий узел самого верхнего уровня содержит сведения обо всех сущностях.

Сущности, локализация которых больше не нужна, могут быть удалены. Важная цель использования имен в распределенных системах — создание ссылок на сущности так, чтобы сущности, на которые нет ссылок, удалялись автоматически. Такая сборка мусора требует подсчета ссылок или трассировки.

В случае подсчета ссылок сущность просто считает количество созданных на нее ссылок. Когда счетчик достигнет нуля, сущность можно удалять. В отличие от подсчета ссылок можно также поддерживать список ссылок на процессы, ссылающиеся на сущность. Список ссылок более устойчив, чем счетчик ссылок, но имеет проблемы с масштабированием.

В методах трассировки все сущности прямо или косвенно ссылаются на заданный набор корневых сущностей, которые помечаются как доступные. Недоступные сущности удаляются. Распределенная трассировка трудна, поскольку требуется проверить все сущности в системе. Решения разнообразны, но в основном они основаны на традиционных сборщиках мусора, характерных для однопроцессорных систем.

Лекция 5. Синхронизация

План лекции

Синхронизация часов

Логические часы

Глобальное состояние

Алгоритмы голосования

Взаимное исключение

Распределенные транзакции

Обзор лекции

В предыдущих лекциях мы рассматривали процессы и связь между процессами. Хотя связь и очень важна, это еще не все. Не менее важны вопросы взаимодействия и синхронизации процессов друг с другом. Взаимодействие частично обеспечивается именованием, которое позволяет процессам как минимум совместно использовать ресурсы и вообще сущности.

В этой лекции мы в основном сосредоточимся на том, как процессы синхронизируются между собой. Так, например, важно, что несколько процессов не способны одновременно получать доступ к совместно используемым ресурсам, таким как принтеры, а должны взаимодействовать, позволяя друг другу временно получать эксклюзивный доступ к этому ресурсу. Другой пример. Несколько процессов могут время от времени нуждаться в соглашениях о порядке прохождения сообщений, о том, например, что сообщение m_1 от процесса P будет отправлено раньше, чем сообщение m_2 от процесса Q.

Как оказывается, синхронизация в распределенных системах часто значительно сложнее, чем синхронизация в однопроцессорных или мультипроцессорных системах. Проблемы и решения, которые обсуждаются

в этой лекции, являются по своей природе глобальными и случаются во множестве различных ситуаций в распределенных системах.

Мы начнем с обсуждения синхронизации, основанной на реальном времени, продолжим обсуждение синхронизацией, у которой всего один относительный параметр упорядочивания, не считая упорядочивания по абсолютному времени. Мы обсудим также понятие распределенного глобального состояния и то, как это состояние записывается в процессе синхронизации.

Во многих случаях важно, чтобы группа процессов назначила один из процессов координатором. Это обычно происходит в соответствии с алгоритмом голосования. Мы рассмотрим различные алгоритмы голосования в отдельном разделе.

Две отдельные темы, касающиеся синхронизации, — это взаимные исключения в распределенных системах и распределенные транзакции. Распределенные взаимные исключения позволяют совместно использовать ресурсы, предотвратив возможность одновременного доступа нескольких процессов. Распределенные транзакции делают нечто похожее, но посредством совершенствования механизма контроля за параллельным выполнением. Взаимные исключения и транзакции будут обсуждаться в отдельных разделах. Существуют распределенные алгоритмы на любой «вкус и цвет» для распределенных систем самых разных типов.

С взаимодействием между процессами тесно связан вопрос синхронизации процессов в распределенных системах. Синхронизация — это все то, что позволяет нам делать правильные вещи в правильное время. Проблема распределенных систем и вообще компьютерных сетей состоит в том, что для них не существует понятия единых совместно используемых часов. Другими словами, процессы на различных машинах имеют свое собственное мнение о времени.

Существуют различные способы синхронизации часов в распределенных системах, но все они основаны на обмене показаниями часов, а это требует

учитывать задержки на посылку и получение сообщений. Точность алгоритмов синхронизации в значительной мере определяют вариации в задержках доступа и способы учета этих вариаций.

Во многих случаях знания абсолютного времени не требуется. Достаточно, чтобы соответствующие события в различных процессах происходили в правильной последовательности. Лампорт показал, что, введя понятие логических часов, можно заставить набор процессов соблюдать общее соглашение относительно правильной очередности событий. В сущности, каждое событие e, такое как посылка или прием сообщения, получает абсолютно уникальную логическую отметку времени $C(e)$, такую, что если событие a происходит раньше b, то $C(a) < C(b)$. Отметки времени Лампорта могут быть расширены до векторных отметок времени: если $C(a) < C(b)$, то мы всегда знаем, что событие a причинно предшествует событию b.

Поскольку в распределенных системах нет понятия совместно используемой памяти, часто трудно определить текущее состояние системы. Определение глобального состояния распределенной системы можно выполнить путем синхронизации всех процессов так, чтобы каждый из них определил и сохранил свое локальное состояние вместе с сообщениями, которые передаются в этот момент. Сама по себе синхронизация может быть выполнена без остановки процессов и записи их состояния. Вместо этого в ходе работы распределенной системы с нее можно сделать мгновенный слепок — распределенный снимок состояния.

Синхронизация между процессами часто требует, чтобы один из процессов выступал в роли координатора. В этом случае если координатор не фиксирован, необходимо, чтобы процессы в распределенных вычислениях могли решить, который из них будет координатором. Это решение принимается посредством алгоритмов голосования. Алгоритмы голосования, как правило, задействуются при сбое или отключении существовавшего координатора.

Важный класс алгоритмов синхронизации — распределенные взаимные исключения. Эти алгоритмы гарантируют, что в распределенном наборе процессов доступ к совместно используемым ресурсам имеет максимум один процесс. Распределенные взаимные исключения можно легко обеспечить с помощью координатора, который будет следить, чья сейчас очередь. Существуют также и полностью распределенные алгоритмы, но их минус — чересчур большая чувствительность к сбоям процессов и связи.

С взаимными исключениями тесно связаны распределенные транзакции. Транзакция состоит из набора операций с совместно используемыми данными, при этом вся транзакция целиком либо полностью выполняется, либо полностью не выполняется. Кроме того, несколько транзакций могут выполняться одновременно, при этом результат оказывается таким, как если бы эти транзакции выполнялись в некоторой заданной очередности. И наконец, транзакции долговечны, в том смысле, что после завершения их результат неизменен.

Лекция 6. Непротиворечивость и репликация

План лекции

Модели непротиворечивости, ориентированные на данные
Модели непротиворечивости, ориентированные на клиента
Протоколы распределения
Протоколы непротиворечивости

Обзор лекции

Важным вопросом для распределенных систем является репликация данных. Данные обычно реплицируются для повышения надежности и увеличения производительности. Одна из основных проблем при этом — сохранение непротиворечивости реплик. Говоря менее формально, это означает, что если в одну из копий вносятся изменения, то нам необходимо

обеспечить, чтобы эти изменения были внесены и в другие копии, иначе реплики больше не будут одинаковыми. В этой лекции мы детально рассмотрим, что в действительности означает непротиворечивость реплицируемых данных и различные способы, которыми она достигается.

Мы начнем с общего обзора, в ходе которого обсудим, для чего используется репликация и как она влияет на масштабируемость. Особое внимание мы уделим репликации на базе объектов, важность которой для распределенных систем в настоящее время растет.

Чтобы добиться высокой производительности в операциях с совместно используемыми данными, разработчики параллельных компьютеров уделяют большое внимание различным моделям непротиворечивости данных в распределенных системах с разделяемой памятью. Эти модели, одинаково успешно применяемые для различных типов распределенных систем, детально рассматриваются в этой лекции.

Реализация моделей непротиворечивости разделяемой памяти для крупномасштабных распределенных систем обычно представляет собой нелегкую задачу. Однако во многих случаях можно использовать простые модели, которые несложно реализовать. Один из конкретных классов подобных моделей — клиентские модели непротиворечивости, которые ограничиваются непротиворечивостью с точки зрения одного (возможно, мобильного) клиента. Клиентские модели непротиворечивости рассматриваются в отдельном разделе.

Непротиворечивость — это еще не все. Мы должны также обсудить, как эта непротиворечивость реализуется на практике. В поддержании непротиворечивости реплик играют роль два более или менее независимых аспекта. Первый из них — это реальное распространение обновлений. Сюда входят вопросы размещения реплик и того, как по ним расходятся обновления. Мы опишем и сравним несколько протоколов распространения.

Второй аспект касается поддержания непротиворечивости реплик. В большинстве случаев приложениям необходим жесткий вариант

непротиворечивости. Говоря неформально, это означает, что обновления должны расходиться по репликам более или менее быстро. Существует несколько альтернативных реализаций строгой непротиворечивости, которые будут рассмотрены в отдельном разделе. Кроме того, внимание уделяется протоколам кэширования, представляющими собой особую форму протоколов поддержания непротиворечивости.

Мы закончим эту лекцию двумя примерами приложений, в которых активно используются непротиворечивость и репликация. Первый пример относится к области параллельного программирования и упоминается также при обсуждении распределенных систем на базе объектов в лекции 9. Второй пример охватывает вопросы причинной непротиворечивости и того, что называется медленной репликацией.

Существует два основных довода в пользу реплицирования данных — повышение надежности распределенных систем и увеличение их производительности. Репликация порождает проблему противоречивости: при обновлении одной из реплик она становится отличной от остальных. Для сохранения непротиворечивости реплик нам необходимо распространять обновления так, чтобы временная противоречивость оставалась незаметной. К сожалению, это может значительно снизить производительность, особенно в крупных распределенных системах.

Единственное решение этой проблемы — посмотреть, нельзя ли немного ослабить требования к непротиворечивости. В моделях строгой непротиворечивости непротиворечивость определяется для отдельных операций чтения и записи совместно используемых данных. Можно провести разделение этих моделей на строгую непротиворечивость, последовательную непротиворечивость, линеаризуемость, причинную непротиворечивость и непротиворечивость FIFO.

Строгая непротиворечивость предполагает, что операции чтения всегда возвращают последнее записанное значение. Из-за нереальности поддержки глобального времени в распределенных системах реализация строгой

непротиворечивости невозможна. Последовательная непротиворечивость и линеаризуемость, в сущности, имеют ту же семантику, которая используется программистами в параллельном программировании: все операции записи должны наблюдаться отовсюду в одинаковом порядке. Линеаризуемость — немного более строгая модель, она упорядочивает операции в соответствии с глобальными часами (с конечной точностью).

Причинная непротиворечивость отражает тот факт, что операции, потенциально зависящие друг от друга, происходят в порядке, определяемом этой зависимостью. Непротиворечивость FIFO просто определяет, что операции одного процесса происходят в порядке, определяемом этим процессом.

Модели слабой непротиворечивости относятся к последовательностям операций чтения и записи. В частности, они предполагают, что каждая последовательность сопровождается сопутствующими операциями с переменными синхронизации, такими, как блокировки. Хотя это требует дополнительных усилий со стороны программистов, эффективно реализовать модели слабой непротиворечивости обычно проще, чем строгой.

В противовес этим моделям, ориентированным на данных, исследователи распределенных баз данных, предназначенных для мобильных пользователей, предложили множество моделей непротиворечивости, ориентированных на клиента. В этих моделях игнорируется тот факт, что данные могут совместно использоваться различными пользователями, вместо этого обсуждение сконцентрировано на уровне непротиворечивости, необходимом отдельному клиенту. В основу этих моделей легло соображение о том, что клиент время от времени может подсоединяться к различным репликам, при этом различия между ними должны быть незаметны. В сущности, модели непротиворечивости, ориентированные на клиента, обеспечивают такое поведение системы, что при соединении клиента с новой репликой эта реплика быстро приводится в соответствие с данными, с

которыми этот клиент работал раньше и которые могут находиться в других репликах.

Для распространения обновлений применяются различные технологии. Разделение можно провести по тому, что на самом деле распространяется, куда распространяются обновления и кто инициирует распространение. Мы можем рассмотреть распространение уведомлений, операций или состояния. Кроме того, не все реплики следует всегда обновлять немедленно. Какая реплика и в какой момент будет обновлена, зависит от протокола распределения. И наконец, следует решить, могут ли обновления продвигаться другим репликам или каждая реплика сама должна извлекать обновление у других реплик.

Протоколы непротиворечивости описывают конкретные реализации моделей непротиворечивости. Для последовательной непротиворечивости и ее вариантов выбор лежит между протоколами первичной копии и реплицируемой записи. В протоколах первичной копии все операции обновления передаются первичной копии, которая затем убеждается, что обновления правильно упорядочены и переданы. В протоколах реплицируемой записи обновление одновременно передается нескольким репликам. В этом случае правильное упорядочение операций часто затруднено.

Лекция 7. Отказоустойчивость

План лекции

Понятие отказоустойчивости

Отказоустойчивость процессов

Надежная связь клиент-сервер

Надежная групповая рассылка

Распределенное подтверждение

Восстановление

Обзор лекции

Характерной чертой распределенных систем, которая отличает их от единичных машин, является возможность частичного отказа. Частичный отказ происходит при сбое в одном из компонентов распределенной системы. Этот отказ может нарушить нормальную работу некоторых компонентов, в то время как другие компоненты это никак не затронет. В противоположность отказу в распределенной системе отказ в нераспределенной системе всегда является глобальным, в том смысле, что он затрагивает все ее компоненты и легко может привести к неработоспособности всего приложения.

При создании распределенной системы очень важно добиться, чтобы она могла автоматически восстанавливаться после частичных отказов, незначительно снижая при этом общую производительность. В частности, когда бы ни случился отказ, распределенная система в процессе восстановления должна продолжать работать приемлемым образом, то есть быть устойчивой к отказам, сохраняя в случае отказов определенную степень функциональности.

В этой лекции мы познакомимся со способами обеспечения отказоустойчивости распределенной системы. После изложения определенных базовых сведений об отказоустойчивости мы рассмотрим вопросы отказоустойчивости процессов и надежной групповой рассылки. Под отказоустойчивостью процессов мы понимаем методы, при помощи которых отказ одного или более процессов проходит для остальной части системы почти незаметно. С этим вопросом связана проблема надежной групповой рассылки, при которой передача сообщений набору процессов производится с гарантией доставки. Надежная групповая рассылка часто необходима для поддержания синхронности процессов.

Как мы уже говорили в лекции 5, атомарность — это свойство, важное для многих приложений. Так, например, в распределенных транзакциях необходимо гарантировать, что все операции, входящие в транзакцию, либо

происходят, либо нет. Фундаментальным для атомарности в распределенных системах является понятие распределенных протоколов подтверждения, которые обсуждаются в отдельном разделе этой лекции.

И, наконец, мы рассмотрим, как восстанавливать систему после отказов. В частности, мы обсудим, когда и как следует сохранять состояние распределенной системы на тот случай, если позже это состояние потребуется восстанавливать.

Отказоустойчивость — это важная область построения распределенных систем. Отказоустойчивость определяется как способность системы маскировать появление ошибок и исправлять их. Другими словами, система отказоустойчива, если она продолжает функционировать при наличии отказов.

Существуют разные типы отказов. Поломка означает простую остановку системы. Пропуск данных наблюдается в том случае, если процесс не реагирует на входящие запросы. Если процесс отвечает слишком быстро или слишком медленно, мы говорим об ошибке синхронизации. Отклик на входящий запрос с ошибкой — пример ошибки отклика. Сложнее всего обрабатывать ситуации, когда в процессе может случиться ошибка любого типа. Такие ошибки называют произвольными или византийскими.

Избыточность — это стандартный способ обеспечения отказоустойчивости. Если избыточность применяется к процессам, становится важным понятие группы процессов. Процессы в группе тесно взаимодействуют между собой для предоставления некоторой услуги. В отказоустойчивых группах один или более процессов могут отказать, не оказывая при этом заметного влияния на доступность услуги, реализуемой группой. Часто необходимо, чтобы взаимодействие внутри группы было высоконадежным и в плане обеспечения отказоустойчивости поддерживало свойства строгой упорядоченности и атомарности.

Надежное групповое взаимодействие, именуемое также надежной групповой рассылкой, может существовать в различных формах. До тех пор

пока группы относительно малы, достижение надежности вполне возможно. Однако при необходимости поддерживать очень большие группы масштабирование надежной групповой рассылки становится проблематичным. Ключевым моментом в реализации масштабируемости становится снижение числа откликов, возвращающих отчет об удачном (или неудачном) приеме разосланного сообщения.

Проблемы усложняются, когда требуется соблюсти атомарность. В протоколах атомарной групповой рассылки важно, чтобы каждый член группы имел одинаковое представление о том, каким элементам группы доставляется сообщение. Атомарная групповая рассылка должна быть точно сформулирована в понятиях модели виртуально синхронного выполнения. В частности, эта модель вводит границы, внутри которых членство в группах не меняется, а сообщения передаются надежно. Сообщения никогда не пересекают границ.

Механизм изменения членства в группе — это пример того, что каждый процесс должен согласиться с единым списком членов. Такое соглашение может быть заключено при помощи протоколов подтверждения, самым распространенным из которых является протокол двухфазного подтверждения. В условиях протокола двухфазного подтверждения координатор сначала проверяет готовность всех процессов к выполнению одной и той же операции (то есть к ее подтверждению), а на втором этапе рассылает результат этого голосования. Протокол трехфазного подтверждения используется для того, чтобы справиться с отказом координатора без необходимости блокировать все процессы, для достижения соглашения дожидаясь восстановления координатора.

Восстановление в отказоустойчивых системах неизменно организуется на основе регулярного сохранения информации о состоянии системы. Работа с контрольными точками полностью распределена. К сожалению, создание контрольной точки — довольно затратная операция. Для повышения производительности множество распределенных систем сочетают

контрольные точки с протоколированием сообщений. Благодаря протоколированию взаимодействия между процессами после отказа системы можно воспроизвести ход ее функционирования.

Лекция 8. Защита

План лекции

Общие вопросы защиты

Защищенные каналы

Контроль доступа

Управление защитой

Пример — Kerberos

Пример — SESAME

Пример — электронные платежные системы

Обзор лекции

Последняя составная часть распределенных систем, которую мы рассмотрим, — это защита. Хотя мы рассматриваем защиту последней, по значимости она далеко не последняя. Вряд ли кто-нибудь может поспорить с тем, что это одна из наиболее сложных частей, ведь защита должна пронизывать всю систему целиком. Единственная брешь в системе защиты может сделать бесполезными все предпринимаемые меры обеспечения безопасности. В этой лекции мы рассмотрим различные механизмы, которые обычно применяются для защиты данных в распределенных системах.

Мы начнем с разговора о базовых понятиях защиты. Построение механизмов защиты всех типов в системе на самом деле бессмысленно, если неизвестно, как именно должны использоваться эти механизмы и чему они будут противостоять. Для этого нужно знать реализуемые правила защиты. Сначала мы рассмотрим понятие правил защиты, а также дадим некоторое общее представление используемых механизмов реализации этих правил.

Кроме того, мы кратко коснемся вопросов криптографической защиты информации.

Системы защиты в распределенных системах можно разделить на две независимые части. Одна из них — это связь между пользователями или процессами, возможно, расположенными на различных машинах. Принципиальный способ гарантировать защиту взаимодействия — это защищенный канал. Защищенные каналы, и более конкретно, идентификация, целостность сообщений и конфиденциальность, обсуждаются в отдельном разделе.

Другая часть систем защиты — это авторизация, которая позволяет гарантировать, что процессы получат только те возможности доступа к ресурсам распределенной системы, на которые имеют право. Авторизацию и контроль доступа можно рассматривать совместно. В добавление к традиционным механизмам контроля доступа мы рассмотрим также контроль доступа при работе с мобильным кодом, например с агентами.

Защищенные каналы и средства контроля доступа нуждаются в механизмах для работы с криптографическими ключами, а также в механизмах добавления пользователей в систему и удаления их из нее. Эти вопросы входят в то, что называется управлением защитой. В отдельном разделе мы рассмотрим вопросы, связанные с управлением криптографическими ключами, управлением защитой в группах и обработкой сертификатов, удостоверяющих право владельца на доступ к определенным ресурсам.

И, наконец, мы конкретизируем наше обсуждение, разобрав два примера реализации средств защиты в распределенных системах. SESAME — это законченная система, которая может встраиваться в распределенные системы для обеспечения их защиты. Чтобы продемонстрировать абсолютно иной подход к защите распределенных систем, мы кратко рассмотрим электронные платежные системы, позволяющие пользователям и коммерсантам,

находящимся в различных местах, безопасным образом инициировать транзакции, включающие заказ и оплату товаров.

Защита играет в распределенных системах чрезвычайно важную роль. Распределенные системы должны предоставлять пользователям и разработчикам механизмы, обеспечивающие реализацию разнообразных правил защиты. Разработка и правильное применение подобных механизмов обычно делают обеспечение защиты в распределенных системах сложной инженерной задачей.

Следует выделить три важных момента. Первый из них состоит в том, что распределенные системы должны иметь средства для организации защищенных каналов связи между процессами. Защищенный канал в принципе предоставляет средства взаимной аутентификации сторон и защищает сообщения во время пересылки от фальсификации. Защищенный канал обычно предоставляет также и средства поддержания конфиденциальности. Это означает, что никто, кроме связавшихся друг с другом сторон, не в состоянии читать передаваемые по каналу сообщения.

Одним из серьезных вопросов, который следует решить при проектировании, является выбор между исключительно симметричной криптосистемой (основанной на совместном использовании секретных ключей) или сочетанием ее с системой с открытым ключом. В текущей практике криптография с открытым ключом используется для рассылки общих секретных ключей. Последние известны также как сеансовые ключи.

Второй вопрос защиты распределенных систем — это контроль доступа, или авторизация. Авторизация касается защиты ресурсов, чтобы только процессы, имеющие соответствующие права доступа, могли получать доступ к соответствующим ресурсам и использовать их. После аутентификации процесса всегда производится контроль доступа.

Существует два способа реализации контроля доступа. Во-первых, каждый из ресурсов может поддерживать собственный список доступа, в котором перечисляются права доступа всех пользователей или процессов.

Кроме того, процесс может иметь сертификат, точно устанавливающий его права на определенный набор ресурсов. Основное достоинство сертификатов состоит в том, что процесс может с легкостью передать свой талон другому процессу, делегировав свои права доступа. Однако сертификаты имеют и недостаток — их обычно не просто отзывать.

Особое внимание следует уделить вопросам управления доступом в случае мобильного кода. Помимо необходимости защиты мобильного кода от вредоносных хостов, обычно важнее защитить хосты от вредоносного мобильного кода. Для этого существуют различные способы, из которых наиболее часто применяется так называемое сито. Однако сито чрезмерно ограничивает возможности программ, поэтому для решения проблемы были разработаны более гибкие методы на основе реально защищенных доменов.

Третий ключевой вопрос защиты распределенных систем — это управление защитой. Здесь есть два важных аспекта — управление ключами и управление авторизацией. Управление ключами включает распространение криптографических ключей, в котором значительную роль играют сертификаты, выдаваемые доверенным третьим лицом. В деле управления авторизацией важны сертификаты атрибутов и делегирование.

Kerberos — это широко распространенная система защиты, основанная на шифровании общих секретных ключей. Ее основное назначение — аутентификация, хотя она также поддерживает протоколы управления доступом и делегирование прав доступа.

SESAME — это типичный пример системы защиты, которая встраивается в распределенные системы. Она основана на комбинированном использовании шифрования открытых ключей и общих секретных ключей. Она многое взяла от системы Kerberos и совместима с Kerberos.

И, наконец, важной областью использования распределенных систем являются электронные платежные системы. Интерес они представляют с точки зрения тех сторон, которые могут связываться через глобальные сети. Особое внимание часто уделяется обеспечению определенного уровня

анонимности покупателя, который возможен в традиционных расчетных системах на базе наличных денег и в их электронных двойниках.

Лекция 9. Распределенные системы объектов

План лекции

CORBA

DCOM

Globe

Сравнение систем CORBA, DCOM и Globe

Обзор лекции

В этой лекции мы переходим от обсуждения принципов построения распределенных систем к изучению различных парадигм, используемых при их построении. Первая парадигма — это распределенные объекты. В распределенных системах объектов понятие объекта играет ключевую роль для реализации прозрачности распределения. В принципе считать объектами можно все; клиенты получают службы и ресурсы в форме объектов, к которым они могут обращаться.

Распределенные объекты образуют важную парадигму, поскольку все аспекты распределения относительно просто скрыть за интерфейсом объектов. Кроме того, объекты могут присутствовать фактически повсюду, и это также усиливает значение этой парадигмы для построения систем. Распределенные объекты лежат в основе двух важных, широко распространенных распределенных систем, которые мы обсудим в этой лекции.

В качестве первого примера мы рассмотрим CORBA, промышленный стандарт распределенных систем. В настоящее время используется множество систем CORBA, и их разработка и стандартизация развиваются по мере того, как растет число установленных систем.

Нашим вторым примером будет DCOM корпорации Microsoft. DCOM можно рассматривать как систему промежуточного уровня, реализованную поверх различных операционных систем Windows, начиная с Windows 95. Фактически все приложения Windows опираются на предоставляемую DCOM функциональность. Таким образом, DCOM можно считать наиболее широко распространенной распределенной системой промежуточного уровня.

Кроме этих двух коммерческих систем известно также множество исследовательских работ по построению различных распределенных систем объектов. В этой лекции мы рассмотрим экспериментальную систему Globe, которая разрабатывается в настоящее время в рамках исследовательского проекта по глобальным распределенным сетям. Интересной особенностью Globe, которая отличает ее от систем типа CORBA и DCOM, является тот факт, что состояние распределенных объектов Globe может раздельно храниться и реплицироваться на нескольких машинах.

Важно так структурировать наше обсуждение, чтобы иметь возможность сравнивать различные системы. По этой причине каждая из систем рассматривается в отдельном разделе, который начинается с обзора наиболее важных концепций системы, а именно ее объектной модели и основных служб, которые она поддерживает. Затем идет обсуждение семи принципов реализации системы: связи, процессов, именования, синхронизации, непротиворечивости и репликации, отказоустойчивости, защиты.

Преимущество подобной структуры в том, что мы можем провести детальное сравнение различных систем, рассматриваемых в разных разделах. С другой стороны, поскольку основные аспекты этих принципов мы уже обсудили в первой части книги, нам осталось описать лишь некоторые дополнительные моменты, а скорее даже детализацию этих основ. Однако эти детали помогут нам понять, как на самом деле работает каждая из систем и каким образом сравнивать их между собой.

Распределенные системы объектов почти всегда базируются на модели удаленных объектов. Такой подход часто требует дополнительной настройки,

например, для поддержки кэширования или репликации. В CORBA для изменения исходящих и входящих запросов используются перехватчики. В DCOM пользовательский маршалинг требует настройки клиентского заместителя. В Globe настройка выполняется с помощью разных локальных объектов, каждый из которых размещается на собственной машине, а все вместе они образуют единый распределенный объект.

Помимо синхронных обращений к методам распределенные системы объектов, такие как CORBA или DCOM, поддерживают альтернативные средства обращения, включая события и асинхронные вызовы методов. Globe не предоставляет таких альтернатив и, в сущности, поддерживает только синхронные обращения.

Организация серверов объектов в различных системах, за исключением некоторых тонкостей, очень похожа. Во всех случаях сервер способен поддерживать более одного объекта. Различия обнаруживаются в гибкой настройке сервера. В CORBA гибкость обеспечивается при помощи адаптеров объектов. DCOM предоставляет стандартный сервер объектов, который может быть перестроен под конкретное приложение. В Globe серверы объектов относительно просты, поскольку те или иные специальные свойства предполагается реализовывать внутри распределенных объектов.

Различия в механизмах именования распределенных систем объектов имеют место на уровне ссылок на объекты. Поддержка осмысленных имен практически одинакова. И в CORBA, и в DCOM внутрисистемные ссылки на объекты зависят от их местоположения. Эти ссылки содержат информацию о местоположении серверов, на которых располагаются объекты. В противоположность им ссылки в Globe не зависят от местоположения, но для этой системы требуется глобальная служба локализации для разрешения ссылок в контактные адреса. Контактный адрес определяет, где и как можно найти указанный объект.

Репликация в CORBA поддерживается только в плане отказоустойчивости. DCOM вообще не поддерживает репликацию,

разработчик приложения должен при необходимости использовать специализированный сервер репликации или явную программную репликацию. В Globe репликация поддерживается каждым объектом в отдельности посредством подобъекта репликации, который реализует конкретный протокол репликации объекта, частью которого он является.

С репликацией тесно связаны вопросы отказоустойчивости. CORBA обеспечивает отказоустойчивость при помощи службы репликации, в основании которой лежит базовая служба надежной групповой рассылки. В DCOM отказоустойчивость поддерживается при помощи транзакций (автоматических). Globe поддерживает отказоустойчивость только через репликацию, а механизмов восстановления не имеет.

И, наконец, каждая из распределенных систем, обсуждаемых в этой лекции, имеет средства защиты. CORBA предлагает полностью защищенную архитектуру, которая позволяет обеспечить индивидуальную защиту каждого объекта. В DCOM защита тесно связана с организацией доступа к существующим службам защиты, таким как Kerberos. В Globe защита также определяется для каждого объекта в отдельности, при этом объектам предоставляются средства для привязки к существующим службам защиты. Вопросы защиты в Globe продолжают исследоваться и в настоящее время.

Лекция 10. Распределенные файловые системы

План лекции

Сетевая файловая система компании Sun

Файловая система Coda

Другие распределенные файловые системы

Сравнение распределенных файловых систем

Обзор лекции

Совместное использование данных — фундаментальное требование распределенных систем. Поэтому нас не должно удивлять, что в основе множества распределенных приложений лежат распределенные файловые системы. Распределенные файловые системы позволяют нескольким процессам в течение продолжительного времени совместно работать с общими данными, обеспечивая их надежность и защищенность. По этой причине они нередко используются в качестве базового уровня распределенных систем и приложений. В этой лекции мы рассмотрим распределенные файловые системы с точки зрения парадигмы распределенных систем общего назначения.

Мы детально изучим две совершенно разные распределенные файловые системы. В качестве первого примера мы рассмотрим сетевую файловую систему NFS компании Sun Microsystem, имеющую гигантское число установок и в настоящее время в форме версий для Интернета постепенно разрастающуюся до глобальных масштабов. Большинство реализаций NFS основаны на спецификации версии 3. Недавно была описана и выпущена пробная редакция версии 4.

Совершенно другой тип распределенных файловых систем представляет Coda. Coda — это потомок файловой системы AFS, крупномасштабной системы, которая разрабатывалась исключительно с целью обеспечить максимальную масштабируемость. От множества других файловых систем Coda отличает поддержка непрерывных операций, для которых границы сегментов сети — не преграда. Так, в частности, подобная поддержка очень удобна для мобильных пользователей (то есть пользователей переносных компьютеров), которые вынуждены часто отключаться от сети.

Также мы кратко рассмотрим еще три системы. Plan 9 — это распределенная система, в которой все ресурсы рассматриваются как файлы. В этом смысле ее можно считать распределенной системой файлов. Следующая система, которую мы рассмотрим, — это xFS, характерная тем, что в ней отсутствуют серверы, а файловую систему реализуют клиенты. И

наконец, мы познакомимся с системой SFS, которая выделяется среди других распределенных файловых систем масштабируемой системой защиты.

Все семь принципов, о которых мы рассуждали в начале книги, применимы и к распределенным файловым системам, и в этой лекции мы рассмотрим, как реализуется каждый из них на примерах существующих систем. В заключение мы проведем сравнительный анализ различных файловых систем.

Распределенные файловые системы представляют собой одну из важных парадигм распределенных систем. Они строятся, в основном, в соответствии с моделью клиент-сервер с кэшированием на клиенте и поддержкой репликации серверов для получения необходимой масштабируемости. Кэширование и репликация требуются также для повышения доступности этих систем.

Распределенные файловые системы отличаются от нераспределенных семантикой разделения файлов. В идеале файловые системы всегда позволяют клиенту прочесть данные, записанные в файл последними. Такую семантику разделения файлов (семантику UNIX) очень трудно эффективно реализовать в распределенных системах. NFS поддерживает ее «урезанный» вариант, именуемый семантикой сеансов, при которой итоговая версия файла определяется последним клиентом, закрывшим файл, открытый на запись. В Coda разделение файлов происходит в соответствии с семантикой транзакций в том смысле, что клиенты, выполняющие чтение, получат последнюю версию файла, только повторно открыв его. Семантика транзакций в Coda не обладает всеми свойствами ACID настоящих транзакций. В том случае, если управление всеми операциями осуществляет сервер, может использоваться и базовая семантика UNIX, хотя масштабируемость в этом случае останется под вопросом.

Чтобы добиться приемлемой производительности, распределенные файловые системы обычно позволяют клиентам кэшировать файлы целиком. Кэширование целых файлов поддерживается в NFS и Coda, хотя существует

возможность сохранять также и большие фрагменты файлов. После того как файл открыт и передан (частично) клиенту, все остальные операции могут осуществляться локально. Изменения сбрасываются на сервер перед закрытием файла. С другой стороны, такие системы, как Plan 9 и xFS, поддерживают блочное кэширование в комбинации с протоколом кэша обратной записи.

Общепринятая практика состоит в том, что распределенная файловая система не строится поверх транспортного уровня, а использует существующий уровень RPC, так что все операции представляют собой вызовы RPC, адресованные файловому серверу, и необходимости в передаче сообщений не возникает. Такой подход используется в NFS, Coda и SFS. Желательно, чтобы уровень RPC поддерживал семантику обращений «максимум однажды», в противном случае эта семантика должна быть явно реализована в форме части уровня файловой системы, как в случае NFS.

Coda отличается от других распределенных файловых систем тем, что поддерживает работу в автономном (отключенном от сети) режиме. Если гарантировать, что кэш клиента всегда содержит данные, которые могут потребоваться ему в ближайшем будущем, можно позволить клиенту продолжать работу, даже если он временно отключен от файлового сервера. Для поддержки такого режима работы необходимо специальное управление кэшем, именуемое накоплением. Накопление нелегко реализовать, тем не менее, было показано, что его эффективная реализация вполне возможна.

Защита очень важна для любых распределенных систем, включая файловые. Система NFS сама по себе фактически не предоставляет каких-либо механизмов защиты, но, обладая стандартизованным интерфейсом, позволяет использовать в связке с ней разнообразные системы защиты, например Kerberos. В противоположность ей Coda и Plan 9 обладают собственными механизмами защиты. Аутентификация в этих системах осуществляется при помощи защищенных вызовов RFC и часто представляет собой вариации на тему протокола аутентификации Нидхема—Шредера. SFS

отличается от них тем, что информация об открытом ключе сервера включается в имена файлов. Такой подход упрощает управление ключами в крупномасштабных системах. На практике SFS распространяет ключ, включая его в имя файла. Специальные средства помогают делать это прозрачно для клиента.

Лекция 11. Распределенные системы документов

План лекции

World Wide Web

Lotus Notes

Сравнение WWW и Lotus Notes

Обзор лекции

Одной из наиболее важных причин, способствовавших популярности как сетей, так и распределенных систем, стало появление Всемирной паутины (World Wide Web). Сила Web заключается в относительной простоте парадигмы: все вокруг — это документы. В этой лекции мы рассмотрим распределенные системы документов, такие как Web. Системы документов предлагают пользователям представление о документах как простом и мощном средстве обмена информацией. Модель проста для понимания и часто близка тому, с чем пользователи уже сталкивались в своей повседневной деятельности. Так, например, в офисах связь часто осуществляется путем передачи записок, отчетов, заявлений и т.п.

Web в настоящее время является наиболее важной распределенной системой документов и, похоже, останется таковой в ближайшем будущем. Фактически, когда большинство людей говорят об Интернете, они имеют в виду Web. Итак, Web — первый пример системы, который мы обсудим в этой лекции.

Другой важной системой документов, появившейся раньше Web, является Lotus Notes. В противоположность Web в основе системы Notes лежат скорее не файлы, а базы данных. В настоящее время система Lotus Notes остается достаточно популярной и часто интегрируется в технологию Web, предоставляя средства разработки web-служб. Lotus Notes будет вторым примером распределенной системы документов, которую мы обсудим в этой лекции. Завершается лекция разделом, в котором мы сравним эти две системы.

Можно утверждать, что распределенные системы документов, особенно World Wide Web, являются наиболее популярными среди конечных пользователей сетевыми приложениями. Понятие о документе как средстве передачи информации понятно для людей, постоянно имеющих дело с реальными документами. Каждый понимает, что такое бумажный документ, так что распространение этого понятия на электронные документы для большинства людей трудностей не вызывает.

Существуют разные способы моделирования документов, самое важное в которых — способ связывания документов друг с другом. Модель гипертекста, которую поддерживают как Web, так и Lotus Notes, существенно способствовала популярности распределенных систем документов. В этой модели легко можно активизировать ссылку на другой документ. Результатом является получение документа, на который указывает ссылка, и вывод его на экран пользователя.

Документы обычно хранятся на серверах, которые предоставляют клиентам доступ к этим документами. Если ссылки на документы могут пересекать границы между серверами, то есть при наличии глобальной системы ссылок на документы, относительно просто организовать глобальное распределение документов. В простейшем случае ссылка содержит имя сервера, на котором хранится документ, и клиент, желающий получить документ, напрямую обращается к этому серверу.

Важным аспектом архитектуры распределенных систем документов являются механизмы доступа к документам. От того, какой механизм доступа предлагается в данной системе, зависит уровень ее масштабируемости. Документы обычно редактируются их единственным владельцем или небольшой группой пользователей. С другой стороны, читать документ может достаточно большое число пользователей. Такой вариант доступа облегчает кэширование и репликацию, поскольку позволяет задействовать слабые формы распространения изменений. Другими словами, гарантировать хорошую производительность системы относительно просто, поскольку копии документов можно разместить неподалеку от пользователей и при этом не создать серьезных проблем с синхронизацией.

Защиту в рассматриваемых системах нужно обеспечивать, как и в любых других. Меры по обеспечению защиты, как обычно, состоят в организации защищенного канала и последующем контроле доступа.

Лекция 12. Распределенные системы согласования

План лекции

Знакомство с моделями согласования

TIB/Rendezvous

Jini

Сравнение TIB/Rendezvous и Jini

Обзор лекции

В предыдущих лекциях мы рассматривали различные подходы к распределенным системам, в которых в качестве субъекта распределения присутствовал некий тип данных. Этот тип данных — объект, файл или документ — является, по сути, порождением нераспределенных систем, который адаптируется для распределенных систем таким образом, чтобы

многие проблемы, вызываемые распределенной архитектурой, были незаметны ни пользователям, ни разработчикам.

В этой лекции мы рассмотрим новое поколение распределенных систем, которые изначально предполагают распределенность многих компонентов системы, и реальные проблемы таких систем вытекают из необходимости согласовывать взаимодействие разных компонентов. Другими словами, вместо того чтобы решать проблемы прозрачного распределения компонентов, мы сосредоточимся на согласовании этих компонентов.

Некоторые вопросы согласования уже рассматривались в предыдущих лекциях, особенно при обсуждении систем на базе событий. Как можно заметить, во многие традиционные распределенные системы постепенно включаются механизмы, играющие ключевую роль в системах согласования.

Перед тем как рассматривать практические примеры систем, мы познакомимся с моделями согласования и самим понятием согласования в распределенных системах. Затем мы обсудим две системы согласования: Rendezvous Bus компании TIBCO и Jini компании Sun Microsystems. Эти примеры помогут нам понять особенности многих распределенных систем согласования, существующих реально или в виде исследовательских проектов.

Распределенные системы согласования начинают играть важную роль в разработки распределенных приложений. Большинство подобных систем отличаются отсутствием ссылочной связности процессов в том смысле, что процессы для связи между собой не нуждаются в явных ссылках друг на друга. Кроме того, может обеспечиваться и отсутствие временной связности, в этом случае для взаимодействия процессов друг с другом им вовсе не обязательно выполняться одновременно.

Одна из важных групп систем согласования — это системы, основанные на парадигме издателя/подписчика, такие как TIB/Rendezvous. В этой модели сообщения доставляются получателям не в соответствии с их адресами, а в соответствии с темой сообщений. Процессы, желающие получать сообщения,

должны подписаться на определенную тему. За выбор пути сообщений от издателя к подписчику отвечает промежуточный уровень.

Другая группа систем согласования использует модель генеративной связи, впервые предложенную в системе Linda. Генеративная связь реализуется разделяемыми пространствами кортежей. Кортеж — это типовая структура данных, сходная с записью. Чтобы извлечь из пространства кортежей нужный кортеж, процесс отыскивает его при помощи эталонного кортежа. Кортеж, соответствующий эталону, выбирается и возвращается запросившему его процессу. Если совпадений нет, процесс блокируется.

Системы согласования отличаются от большинства других распределенных систем тем, что в них в основном решается задача предоставления удобного способа связи между процессами, которые ничего не знают друг о друге заранее. Связь может и далее осуществляться с сохранением анонимности. Основное преимущество подобного подхода — его гибкость. Можно дополнять и изменять продолжающую работать систему.

Принципы распределенных систем, которые мы обсуждали в первой половине книги, равно применимы и к системам согласования, хотя кэширование и репликация играют в современных их реализациях не столь важную роль. Кроме того, именование в них сильно связано с поиском по атрибутам. Аналогичный подход реализован и в службах каталогов.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)**

Учебное пособие по дисциплине «Архитектура современных распределенных систем»

направление 230100.68 – «Информатика и вычислительная техника»
(магистратура)

Факультет информатики
Кафедра информационных систем и технологий

Разработал:
Востокин С.В.,
профессор кафедры ИСТ

Самара 2013 г.

Основная литература по дисциплине

1. Таненбаум, Э. Распределенные системы. Принципы и парадигмы [Текст] / Э. Таненбаум, М. ван Стеен. - СПб. [и др.] : Питер , 2003. - 877 с. - (Классика computer science). - ISBN 5-272-00053-6 (3 экз.)

2. Гордеев, А.В. Операционные системы [Текст] : [учеб. для вузов по направлению подгот. бакалавров и магистров "Информатика и вычисл. техника" и направлению подгот. дипломир. специалистов "Информатика вычисл.техника"] / А. В. Гордеев. - 2-е изд. - СПб. и др. : Питер : Питер принт, 2004. - 415 с. - (Учебник для вузов). - ISBN 5-94723-632-X (гриф Минобразования России) (20 экз.)

3. Олифер, В.Г. Сетевые операционные системы [Текст] : [учеб. пособие для вузов по направлению подгот. дипломир. специалистов "Информатика и вычисл. техника] / В. Г. Олифер, Н. А. Олифер. - СПб. [и др.] : Питер : Питер Пресс, 2007. - 538 с. - (Учебник для вузов). - ISBN 5-272-00120-6 (гриф Минобразования России) (10 экз.)

4.Востокин, С.В. Графическая объектная модель параллельных процессов и ее применение в задачах численного моделирования [Текст] / С.В. Востокин. Изд-во Самарского научного центра РАН – Самара, 2007. 186 с., ил. – ISBN 978-5-93424-284-9 (37 экз.)

Дополнительная литература по дисциплине

1. Востокин, С.В. Вопросы, задания и упражнения по курсу "Операционные системы" [Текст] : [лаб. практикум] / М-во образования и науки Рос. Федерации, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т) ; [сост. С. В. Востокин]. - Самара : Изд-во СГАУ, 2012. - 29 с. (20 экз.)

2. Востокин, С.В. Операционные системы [Электронный ресурс] : [учеб. для вузов по направления подгот. бакалавров "Инф. и выч. техн.", "Фундам. информатика и информ. технологии", "Прикладная математика и информатика", "Прикладная математика и физика"] / С.В. Востокин ; М-во образования и науки РФ, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т). -

Электрон. текстовые дан. - Самара : Изд-во СГАУ, 2012. – ISBN 978-5-7883-0916-3 (2 эл. опт. Диска)

3. Таненбаум, Э.Современные операционные системы [Текст] / Эндрю Таненбаум. - 2-е изд. - СПб. [и др.] : Питер : Питер принт, 2005. - 1037 с. - (Классика computer science). - ISBN 5-318-00299-4 (5 экз.)

4. Таненбаум, Э. Современные операционные системы [Текст] / Эндрю Таненбаум. - 2-е изд., [перераб. и испр.]. - СПб. [и др.] : Питер : Питер Пресс, 2007. - 1037 с. - (Классика computer science). - ISBN 978-5-318-00299-1 (2 экз.)

5. Солдатова, О. П. Системное программирование [Текст] : Курс лекций / О. П. Солдатова, С. В. Востокин ; Самар. гос. аэрокосм. ун-т им. С. П. Королева. - Самара : [б. и.], 2002. - 123 с. - ISBN 5-7883-0226-9 (94 экз.)

Электронные источники и интернет ресурсы

1. Microsoft Developer Network http://msdn.microsoft.com
2. www.prenhall.com/tanenbaum
3. Проект автоматизации параллельных и распределенных вычислений «Темплет» http://templet.ssau.ru
4. Интуит национальный открытый университет. http://intuit.ru

Методические указания и рекомендации

Текущий контроль знаний студентов завершается на отчетном занятии, результатом которого является допуск или недопуск студента к зачёту по дисциплине. Основанием для допуска к зачёту является выполнение и отчет студента по всем лабораторным работам. Зачёт проводится согласно положению о текущем и промежуточном контроле знаний студентов, утвержденному ректором университета. Зачёт ставится на основании письменного и устного ответов студента по тестовому заданию, а также, при необходимости, ответов на дополнительные вопросы. Тестовое задание включает 18 вопросов. Дополнительно может быть предложен как теоретический вопрос, так и вопрос на понимание программ по листингам.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)**

Задания на лабораторные работы по дисциплине «Архитектура современных распределенных систем»

направление 230100.68 – «Информатика и вычислительная техника»
(магистратура)

Факультет информатики
Кафедра информационных систем и технологий

Разработал:
Востокин С.В.,
профессор кафедры ИСТ

Самара 2013 г.

Лабораторная работа №1. Удаленные объекты Java RMI.

Цель работы: изучить способ написания распределенного приложения с использованием удаленного вызова методов в среде Java

Теоретические сведения:

Распределенная объектная модель, специфицирующая, каким образом производится вызов удаленных методов, работающих на другой виртуальной машине Java.

При доступе к объектам на другом компьютере возможно вызывать методы этого объекта. Необходимо только передать параметры метода на другой компьютер, сообщить объекту о необходимости выполнения метода, а затем получить обратно возвращаемое значение. Механизм RMI дает возможность организовать выполнение всех этих операций.

Типичная реализация модели Java-RMI, использующая объекты 'заглушки'(stub) и 'скелета'(skeleton).

В терминах RMI объект, который вызывает удаленный метод, называется клиентским объектом, а удаленный объект — серверным объектом. Компьютеры выступают в роли клиента и сервера только для конкретного вызова. Вполне возможно, что при выполнении следующей операции эти компьютеры поменяются ролями, то есть сервер предыдущего вызова может сам стать клиентом при обращении к объекту на другом компьютере.

При вызове метода удаленного объекта на самом деле вызывается обычный метод языка Java, инкапсулированный в специальном объекте-заглушке (stub), который является представителем серверного объекта. Заглушка находится на клиентском компьютере, а не на сервере. Она упаковывает параметры удаленного метода в блок байтов. Каждый параметр кодируется с помощью алгоритма, обеспечивающего независимость от аппаратуры. Например, числа всегда передаются в порядке, при котором сначала передается старший байт (big-endian). При этом объекты подвергаются сериализации. Процесс кодирования параметров называется развертыванием параметров (parameter marshaling). Основная цель развертывания параметров — преобразование их в формат, пригодный для передачи параметров от одной виртуальной машины к другой.

Метод, принадлежащий заглушке, создает блок, в который входят следующие элементы:

- идентификатор удаленного объекта;
- описание вызываемого метода;

- развернутые параметры.

Затем метод заглушки посылает эту информацию серверу. Далее объект-получатель выполняет для каждого вызова удаленного метода следующие действия:

- свертывание параметров;
- поиск вызванного объекта;
- вызов заданного метода;
- извлечение и развертывание возвращаемого значения или исключения, сгенерированного данным методом;
- передача пакета, состоящего из развернутых возвращаемых данных, объекту-заглушке на клиентском компьютере.

Клиентский объект-заглушка свертывает возвращаемое значение или исключение, полученное с сервера. Результат свертывания становится возвращаемым значением метода заглушки. Если удаленный метод возвращает исключение, то объект-заглушка повторит его в среде объекта-клиента.

Для вызова удаленного метода используется тот же синтаксис, что и для обращения к локальному методу. Например, чтобы вызвать метод `getQuantity()` объекта-заглушки centralWarehouse центрального хранилища данных на удаленном компьютере, потребуется использовать приведенный ниже код.

```
int q=centralWarehouse.getQuantity("SuperSucker 100 Vacuum Cleaner");
```

Для доступа к удаленным методам клиентский код всегда использует объектные переменные типы interface. Например, с приведенным выше методом может быть связан следующий интерфейс:

```
interface Warehouse {
        int getQuantity(String description) throws RemoteException;
        Product getProduct(Customer cust) throws RemoteException;
        …
}
```

Объявление переменной для объекта, который реализует этот интерфейс, будет выглядеть так:

```
Warehouse centralWarehouse = …;
```

Конечно, интерфейсы представляют собой абстракции и содержат только перечень методов. Переменные типа interface всегда должны быть связаны с фактическим объектом. При вызове удаленных объектов переменная ссылается на объект-заглушку. При этом клиентская программа ничего не знает о типе заглушки, а сами заглушки и связанные с ними объекты создаются автоматически.

При передаче объекта другой программе (он может быть параметром либо

возвращаемым значением удаленного метода) нужен файл класса, соответствующий этому объекту. Например, метод, который возвращает значение типа Product. При компиляции клиентской программы должен быть сгенерирован файл класса Product.class.

При загрузке фрагментов кода по сети всегда возникают сомнения по поводу должного обеспечения безопасности. В связи с этим в приложениях с использованием RMI применяется диспетчер защиты. Он защищает заглушки от проникновения в них вирусов.

Лабораторная работа №2. Система координации Jini

Цель работы: изучить способ написания распределённых приложений с использованием системы координации на примере Jini (JavaSpaces)

Теоретические сведения:

Jini — это распределенная система, состоящая из разных, но взаимосвязанных элементов. Она жестко привязана к языку программирования Java, хотя многие из ее принципов равно могут быть реализованы и при помощи других языков. Важной частью системы является модель согласования генеративной связи.

Модель согласования Jini обеспечивает как временную, так и ссылочную несвязность процессов при помощи Linda-подобной системы согласования JavaSpace. JavaSpace — это разделяемое пространство данных, в котором хранятся кортежи. Кортежи представляют собой типизованные наборы ссылок на объекты Java. В одной системе Jini могут сосуществовать несколько пространств JavaSpace.

Кортежи хранятся в сериализованной форме. Другими словами, когда бы процессу ни потребовалось сохранить кортеж, сначала выполняется маршалинг кортежа, причем также подразумевается маршалинг всех его полей. В результате если кортеж содержит два разных поля, ссылающихся на один и тот же объект, то кортеж, сохраняемый в реализации JavaSpace, будет содержать две подвергнутые маршалингу копии этого объекта.

Кортеж помещается в пространство JavaSpace при помощи операции write, которая сначала выполняет маршалинг кортежа, а затем сохраняет его. Каждый раз при выполнении для кортежа операции write в JavaSpace сохраняется новая подвергнутая маршалингу копия этого кортежа. Мы можем ссылаться на каждую такую копию, как на экземпляр кортежа (tuple instance).

Интересный аспект генеративной связи в Jini — это способ чтения экземпляров кортежа в JavaSpace. Чтобы прочесть экземпляр кортежа, процесс предоставляет другой кортеж и использует его как эталон (template), соответствующий считываемым экземплярам кортежа, хранящимся в JavaSpace.

Как и любой другой кортеж, эталонный кортеж представляет собой типизованный набор ссылок на объекты. В JavaSpace можно прочитать только экземпляры тех кортежей, которые имеют одинаковый с эталоном тип. Поля в эталонном кортеже также содержат либо ссылки на реальные объекты, либо значение NULL.

Чтобы сопоставить экземпляр кортежа в JavaSpace эталонному кортежу, обычным образом выполняется маршалинг эталонного кортежа, включая маршалинг полей со значением NULL. Для каждого экземпляра кортежа того же типа, что и эталон, производится сравнение, поле за полем, с подвергнутыми маршалингу полями эталонного кортежа. Два поля совпадают, если оба они содержат копии одной и той же ссылки или если поле в эталонном кортеже равно NULL. Экземпляр кортежа совпадает с эталонным кортежем, если попарно совпадают соответствующие поля.

Когда обнаруживается экземпляр кортежа, совпадающий с эталонным кортежем (который является частью операции read), выполняется демаршалинг этого экземпляра, и он возвращается процессу, инициировавшему чтение. Для чтения может быть использована также операция take, которая заодно удаляет экземпляр кортежа из пространства JavaSpace. Обе операции блокируют вызвавший их процесс до обнаружения нужного экземпляра кортежа. Максимальное время блокирования можно предсказать. Кроме того, существуют реализации, немедленно возвращающие управление, если нужного кортежа не существует.

По сравнению с моделью публикации/подписки, используемой в TIB/Rendez- VOUS, процессы, применяющие JavaSpace, не должны выполняться одновременно. На самом деле, если пространство JavaSpace реализовано как сохранное хранилище, всю систему Jini можно выключить и запустить снова, не потеряв ни одного экземпляра кортежа. К сожалению, реализация генеративной связи дорогостояща. По этой причине разработка высокоэффективных реализаций JavaSpace или любой другой Linda-подобной системы представляет собой немалую проблему. Обычно при попытке реализовать эту модель в глобальной сети источником проблем масштабируемости является генеративная связь.

Лабораторная работа №3. Распределенные объекты Templet

Цель работы: разработка схемы распределенного приложения с использованием пакета TempletSDK

Теоретические сведения:

Templet - это метод спецификации процессной модели, позволяющий: 1)

описывать протоколы взаимодействия процессов в виде последовательности передаваемых сообщений; 2) представлять логику работы процессов посредством процедур, обрабатывающих сообщения; 3) визуализировать процессы и их взаимодействие при помощи аннотированных графов.

Композиция процессов. Параллельная система в целом моделируется как композиция элементов модели «процесс» и элементов модели «канал». Интерфейсными элементами процессов, участвующих в композиции, являются порты.

Каналы. Диаграмма канала описывает объекты, задающие информационные связи между процессами, показывает, какие именно сообщения передаются между процессами, и определяет возможный порядок их передачи. Во взаимодействии по каналу участвуют два процесса – клиент и сервер. Возможна передача произвольного числа сообщений в обе стороны. В любой момент времени передается не более одного сообщения.

Состояние канала – это переменная, хранящая дугу с передаваемым сообщением. В начальный момент переменная хранит специальный признак, сигнализирующий об отсутствии сообщения.

В начальном состоянии возможна передача сообщений процессом-клиентом по дугам, исходящим из начальной вершины. Новым состоянием канала будет дуга, соответствующая переданному сообщению.

Если состоянием канала является дуга, то возможен прием сообщения, помечающего ее. Принять сообщение может процесс-клиент, если вершина, в которую входит дуга, является вершиной процесса-клиента. Иначе сообщение может принять процесс-сервер. Принимающий сообщение процесс может отправить ответное сообщение из тех, которые помечают дуги, исходящие из рассматриваемой вершины. Аналогично новым состоянием канала будет дуга, соответствующая переданному сообщению.

Процессы. Диаграмма процесса описывает алгоритм обработки поступающих сообщений. Результатом обработки могут стать отправка новых сообщений и/или изменение состояния процесса. Процессы являются пассивными, управляемыми сообщениями: алгоритм обработки запускается только при поступлении сообщения и не может быть прерван другим сообщением до его завершения.

Состояние процесса – это переменная, хранящая значение текущей вершины графа процесса. Когда обработка не выполняется, переменная хранит специальный признак останова. Если имеется сообщение, отправленное процессу, то рано или поздно начнется обработка данного сообщения с вершины-порта. Переменная состояния процесса примет значение соответствующей вершины.

Правило передачи управления для портов: если имеется исходящая дуга с пометкой yes, управление передается по ней; если имеется исходящая дуга, помеченная поступившим сообщением, управление передается по ней; иначе, если имеется дуга с пометкой no, управление передается по ней.

Правило запуска процедуры обработки сообщений: процедура запускается, если одновременно (1) переменная состояния принимает значение вершины,

помеченной данной процедурой; (2) возможен прием сообщений, ведущих в эту вершину; (3) возможна отправка сообщений, исходящих из данной вершины.

Отправка исходящих сообщений выполняется, если одновременно процедура была запущена и вернула признак успешного завершения.

Правило передачи управления для процедур: передача управления по дуге с пометкой yes происходит, если процедура была запущена и вернула признак успешного завершения; иначе происходит передача управления по дуге с пометкой no.

Если нет исходящих из вершин дуг, по которым можно выполнить переход в текущей ситуации, переменная состояния процесса принимает значение признака останова.

Варианты заданий

С использованием технологий RMI, Jini (JavaSpace), Templet реализовать распределенные алгоритмы:

1. Параллельное умножение матриц
   а) итеративный параллелизм (вариант 1)
   б) метод портфеля задач (вариант 2)
   в) круговой конвейер (вариант 3)
2. Вычисление интеграла функции по методу адаптивной квадратуры
   а) рекурсивный параллелизм (вариант 4)
   б) метод портфеля задач (вариант 5)
3. Решение сеточного уравнения методом Якоби
   а) с разделяемыми переменными (вариант 6)
   б) с передачей сообщений (вариант 7)
   в) красное-черное (вариант 8)
4. Гравитационная задача N тел (вариант 9)
5. LU-разложение (вариант 10)

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)**

Задания на практические работы по дисциплине «Архитектура современных распределенных систем»

направление 230100.68 – «Информатика и вычислительная техника»
(магистратура)

Факультет информатики
Кафедра информационных систем и технологий

Разработал:
Востокин С.В.,
профессор кафедры ИСТ

Самара 2013 г.

Задания на практические работы по лекции 1

1. Какова роль программного обеспечения промежуточного уровня в распределенных системах?
2. Объясните, что такое прозрачность (распределения) и приведите примеры различных видов прозрачности.
3. Почему иногда так трудно скрыть наличие в распределенной системе сбоя и восстановление после него?
4. Почему реализация максимально возможной степени прозрачности — это не всегда хорошо?
5. Что такое открытая распределенная система, и какие преимущества дает открытость?
6. Опишите точно, что такое масштабируемая система.
7. Масштабируемости можно добиться, используя различные методики. Что это за методики?
8. Чем мультипроцессорная система отличается от мультикомпьютерной?
9. В чем состоит разница между распределенными и сетевыми операционными системами?
10. Какова причина разработки распределенных систем с совместно используемой памятью? В чем состоит главная трудность их эффективной реализации?

Задания на практические работы по лекции 2

1. Во многих протоколах модели OSI каждый уровень добавляет к сообщению свой заголовок. Несомненно, создавать в начале каждого сообщения единый заголовок, содержащий всю контрольную информацию, было бы более эффективно, чем поддерживать все эти отдельные заголовки. Почему так не делается?
2. Почему коммуникационные службы транспортного уровня обычно не

подходят для построения распределенных приложений?

3. Надежная массовая рассылка требует от отправителя надежной посылки сообщений группе получателей. Может ли подобная служба находиться на промежуточном уровне или должна быть частью нижнего уровня?
5. В языке С существует конструкция, называемая объединением (union), в которой поле записи — в С называемой структурой (struct) — может хранить любое значение из нескольких вариантов. Во время исполнения надежного способа определить, что там находится, не существует. Должна ли эта особенность языка С как-то поддерживаться при удаленном вызове процедур? Поясните свой ответ.
7. Рассмотрим клиент, посылающий на сервер асинхронный вызов RPC и ожидающий возвращения сервером результата, который также будет послан при помощи асинхронного вызова RPC. Будет ли это аналогично использованию клиентом обычного вызова RPC? Что произойдет, если мы заменим асинхронные вызовы RPC синхронными?
8. Вместо регистрации некоего сервера с помощью демона, как это сделано в DCE, мы могли бы навсегда закрепить его за одной из конечных точек. Конечная точка затем может использоваться в качестве ссылки на объект в адресном пространстве сервера. Каково основное неудобство подобной схемы?
9. Приведите пример реализации ссылки на объект, которая позволила бы клиенту выполнить привязку к нерезидентному удаленному объекту.
10. Java и другие языки поддерживают обработку исключений, которые возбуждаются при возникновении ошибки. Каким образом можно применить исключения в RPC и RMI?

Задания на практические работы по лекции 3

1. Имеет ли смысл ограничивать число потоков выполнения серверного процесса?

2. Существуют ли ситуации, в которых однопоточный сервер оказывается лучше многопоточного? Приведите пример.
3. Статически ассоциировать с облегченным процессом единственный поток выполнения — это не самая лучшая идея. Почему?
4. Иметь в процессе только один облегченный процесс — также не лучшая идея. Почему?
5. Опишите простую схему, в которой облегченных процессов столько же, сколько работающих потоков выполнения.
6. Заместители могут поддерживать прозрачность репликации, обращаясь к каждой из реплик. Может ли объект (на стороне сервера) быть предметом реплицированного обращения?
7. Создание параллельных серверов путем порождения вложенных процессов имеет свои преимущества и недостатки по сравнению с многопоточными серверами. Опишите их.
8. Набросайте архитектуру многопоточного сервера, поддерживающего посредством сокетов несколько протоколов, и его интерфейса транспортного уровня с базовой операционной системой.
9. Объясните, что такое адаптер объектов.
10. Опишите особенности адаптера объектов, используемого для поддержания сохранных объектов.

Задания на практические работы по лекции 4

1. Приведите пример ситуации, в которой для реального доступа к сущности ее адрес разрешается в другой адрес.
2. Зависит или нет от локализации URL-адрес http://www.acme.org/index.html? Как насчет адреса http://www.acme.nl/index.html?
3. Приведите несколько примеров правильных идентификаторов.
4. Как найти точку монтирования в большинстве UNIX-систем?
5. Jade — это распределенная файловая система, использующая отдельные

пространства имен для каждого пользователя. Другими словами, каждый пользователь имеет собственное закрытое пространство имен. Могут ли имена из подобных пространств имен при разделении ресурсов совместно использоваться несколькими пользователями?

6. Рассмотрим DNS. Для ссылки на узел N в поддомене, реализованном в другой нежели текущий домен зоне, должен быть определен сервер имен для этой зоны. Всегда ли необходимо включать запись о ресурсах для адреса этого сервера или иногда достаточно указать только его доменное имя?
7. Может ли идентификатор содержать информацию о сущности, которую он идентифицирует?
8. Опишите эффективную реализацию глобально уникальных идентификаторов.
9. Приведите пример того, как должен работать механизм свертывания в URL.
10. Объясните разницу между жесткой и мягкой ссылкой в UNIX-системах.

Задания на практические работы по лекции 5

1. Назовите как минимум три источника задержек между радиостанцией WWV, которая рассылает сигналы точного времени, и процессором в распределенной системе, устанавливающим свои внутренние часы.
2. Рассмотрим поведение двух машин в распределенной системе. Обе они имеют часы, которые считаются выставленными на 1000 тиков в миллисекунду. Одни из часов действительно выдают такую частоту тиков, другие же тикают всего 990 раз в миллисекунду. Если поправки UTC приходят раз в минуту, какое максимальное расхождение может возникнуть между часами?
3. Является ли абсолютно необходимым подтверждение каждого сообщения для полностью упорядоченной групповой рассылки с отметки времени по Лампорту?
4. Рассмотрим коммуникационный уровень, в котором сообщения доставляются

в том же порядке, в котором они были отправлены. Приведите пример, когда даже такое упорядочение излишне строго.

5. Представьте себе, что два процесса одновременно обнаруживают передачу координатора и решают провести голосование по алгоритму забияки. Что произойдет в этом случае?
6. Многие распределенные алгоритмы требуют наличия координирующего процесса. В какой степени такие алгоритмы могут считаться распределенными?
7. Когда для реализации транзакций над файлами используется закрытое рабочее пространство, может случиться так, что в родительское рабочее пространство придется копировать большое количество индексов файлов. Как это сделать, не вводя условий гонок?
8. Приведите полный алгоритм попытки блокирования файла с успешным и неуспешным результатами. Рассмотрите блокировку на чтение и на запись, а также возможность файла быть в момент попытки неблокированным, заблокированным на чтение, заблокированным на запись.
9. Системы, использующие блокировки для управления параллельным выполнением транзакций, обычно отличают блокировку на чтение от блокировки на запись. Что произойдет, если процесс всегда устанавливал блокировку на чтение, а теперь хочет заменить ее блокировкой на запись?
10. Что произойдет в случае замены блокировки на запись блокировкой на чтение?

Задания на практические работы по лекции 6

1. Доступ к совместно используемым объектам в Java можно сериализовать, объявив их методы синхронизуемыми. Достаточно ли этого, чтобы гарантировать сериализацию при репликации этих объектов?
2. Рассмотрим монитор, описанный в лекции 1. Если потоки выполнения в реплицированном мониторе можно блокировать, что нам нужно сделать

для гарантии правильного срабатывания условной переменной?

3. Опишите своими словами, какова главная причина создания моделей слабой непротиворечивости.
4. Опишите, как реализована репликация в DNS и почему она так хорошо работает на практике.
5. В ходе обсуждения моделей непротиворечивости мы часто ссылались на контракт между программой и хранилищем данных. Зачем нужен такой контракт?
6. Линеаризуемость предполагает существование глобальных часов. Однако, как мы видели, подобное требование делает нереальным поддержание в большинстве распределенных систем строгой непротиворечивости. Можно ли реализовать линеаризуемость в физически распределенных хранилищах данных?
7. Мультипроцессорная система имеет одну шину. Можно ли реализовать в такой системе память со строгой непротиворечивостью?
8. Часто утверждается, что модели слабой непротиворечивости — это дополнительная «головная боль» для программистов. В какой степени это высказывание соответствует действительности?
9. Во многих реализациях свободной непротиворечивости в распределенных системах с разделяемой памятью общие переменные синхронизируются при освобождении, а не при захвате. Зачем для них вообще нужен захват?
10. Какую непротиворечивость обеспечивает система Огса, последовательную или поэлементную? Поясните свой ответ.

Задания на практические работы по лекции 7

1. Надежные системы часто требуют обеспечения высокой степени защиты. Почему?
2. Что делает модель аварийной остановки трудной для реализации в случае поломок?

3. Рассмотрим web-браузер, возвращающий устаревшие страницы из кэша, а не обновленные с сервера. Является ли это ошибкой, и если да, какой это тип ошибки?
4. Может ли модель тройного модульного резервирования справиться с византийскими ошибками?
5. Приведите пример, когда для группового взаимодействия вообще не нужно никаких синхронизирующих сообщений.
6. Всегда ли в надежных групповых рассылках есть необходимость в том, чтобы копии сообщений сохранялись на коммуникационном уровне с целью повторной посылки?
7. В какой степени важна масштабируемость для атомарных групповых рассылок?
8. Мы предложили, что при атомарной групповой рассылке можно сохранять дату осуществления изменения согласного на это набора процессов. В какой степени мы можем гарантировать, что каждое из изменений действительно было сделано?
9. Виртуальная синхронность аналогична слабой непротиворечивости распределенных хранилищ данных с изменениями представления групп в качестве точек синхронизации. В этом контексте, что будет аналогом строгой непротиворечивости?
10. Почему в протоколе двухфазного подтверждения невозможно полностью исключить блокировку, даже в случае выбора участниками нового координатора?

Задания на практические работы по лекции 8

1. Какие механизмы могут предоставляться в распределенных системах в качестве служб защиты разработчикам приложений, которые, как говорилось в лекции 5, в проектировании систем доверяют только сквозным аргументам?

2. Можно ли в случае подхода RISSC сосредоточить все службы защиты на защищенных серверах?
3. Предположим, что вас попросили разработать распределенное приложение, которое должно помочь преподавателям принимать экзамены. Укажите как минимум три условия, которые должны стать частью правил защиты такого приложения.
4. Придумайте простой алгоритм аутентификации с использованием подписей для криптосистемы с открытым ключом.
5. Как может быть реализована смена ролей в матрице контроля доступа?
6. Назовите три проблемы, с которыми сталкиваются разработчики интерфейсов, когда для защиты от неавторизованного доступа к локальным ресурсам со стороны мобильных программ они вынуждены, как описано в соответствующем разделе, вставлять инструкции включения и отключения привилегий.
7. Назовите несколько достоинств и недостатков использования централизованного сервера управления ключами.
8. Протокол обмена ключами Диффи—Хеллмана можно использовать также и при создания общего секретного ключа для трех сторон. Объясните, как.
9. В протоколе обмена ключами Диффи—Хеллмана отсутствует аутентификация. Из-за этого третья сторона, Чак, может легко вмешаться в обмен ключами между Алисой и Бобом, после чего проникнуть через механизмы защиты. Объясните, как.
10. Имеет ли смысл ограничение срока жизни сеансового ключа? Если да, приведите пример.

Задания на практические работы по лекции 9

1. Почему для определения интерфейсов объектов используется язык определения интерфейсов (IDL)?
2. Когда могут пригодиться механизмы динамического обращения CORBA?

Приведите пример.

3. Какой из шести видов связи соответствует модели обращений, принятой в CORBA?
4. Опишите простой протокол, реализующий семантику обращений к объектам «максимум однажды».
5. При асинхронном методе обращения нужно ли, чтобы клиентские и серверные объекты в CORBA были сохранными?
6. В сообщении Request протокола GIOP ссылка на объект и имя метода являются частями заголовка сообщения. Почему их нельзя поместить непосредственно в тело сообщения?
7. Поддерживает ли CORBA правило обращений «один поток выполнения — один объект?
8. Пусть имеется две системы CORBA, каждая из которых имеет свою службу именования. Опишите, как можно объединить две службы именования в одну объединенную службу именования.
9. Мы утверждали, что при привязке к объекту CORBA клиентский брокер ORB может выбрать дополнительные службы защиты на основе ссылок на объекты. Как клиентский брокер ORB может узнать о существовании этих служб?
10. Если ORB системы CORBA использует несколько перехватчиков, не имеющих отношение к защите, насколько важен порядок вызова этих перехватчиков?

Задания на практические работы по лекции 10

1. Обязательно ли, чтобы файловый сервер, на котором реализована система NFS версии 3, не хранил информацию о состояния?
2. NFS не имеет глобального разделяемого пространства имен. Существует ли в этом случае способ имитации пространства имен?
3. Укажите простое расширение операции NFS lookup, которое позволяло бы

производить итеративный поиск в ситуации с сервером, экспортирующим каталоги, монтируемые к другому серверу.

4. В UNIX-подобных операционных системах открытие файла с использованием дескриптора файла может производиться только на уровне ядра. Приведите возможную реализацию дескриптора файла NFS для сервера NFS пользовательского уровня в системе UNIX.
5. При использовании автоматического монтировщика, который устанавливает символические ссылки, как было описано в этой лекции, сложнее обеспечить прозрачность монтирования. Почему?
6. Представьте, что текущее состояние запрета на доступ к файлу в NFS — WRITE. Возможна ли ситуация, когда другой клиент сначала успешно открывает файл, а затем запрашивает блокировку на чтение?
7. Если рассматривать согласованность кэшей какой тип протокола согласованности кэшей реализован в NFS?
8. Реализует ли NFS поэлементную непротиворечивость? А свободную непротиворечивость?
9. Мы установили, что NFS реализует модель удаленного доступа к файлам. Можно также считать, что она реализует модель загрузки-выгрузки. Объясните, почему.
10. В NFS атрибуты кэшируются с использованием правил согласованности кэша сквозной записи. Так ли необходимо передавать все изменения атрибутов на сервер немедленно?

Задания на практические работы по лекции 11

1. До какой степени электронная почта соответствует модели документов Web?
2. В Web клиент сначала получает файл, а затем открывает его и выводит на экран. Что можно сказать о таком подходе применительно к файлам мультимедиа?
3. Почему при наличии сохранных соединений производительность

значительно выше по сравнению с несохранными соединениями?

4. Объясните разницу между модулями расширения, апплетами, сервлетами и программами CGI.
5. Опишите структуру службы именования общего назначения, преобразующей URN в URL, то есть службы, которая разрешает имя URN в указатель URL копии документа, идентифицируемого именем URN.
6. Достаточно ли клиенту в системе WebDAV для получения права на запись предъявить серверу только маркер блокировки?
7. Несмотря на то что web-заместитель вычисляет время истечения срока годности документа, это независимо от него делает еще и сервер. В чем достоинство такого подхода?
8. Какому протоколу распространения соответствует сеть CDN Akamai — извлечения или продвижения?
9. Опишите простую схему, при использовании которой сервер CDN Akamai может, не проверяя документ на исходном сервере, обнаружить, что срок годности кэшированного документа истек.
10. Имеет ли смысл вместо использования одной или нескольких глобальных стратегий репликации поддерживать собственную стратегию репликации для каждого web-документа?

Задания на практические работы по лекции 12

1. К какому типу модели согласования вы отнесете системы очередей сообщений?
2. Отсутствие ссылочной связности в TIB/Rendezvous компенсируется адресацией по теме. Какими еще средствами компенсируется отсутствие ссылочной связности в этой системе?
3. Опишите реализацию системы публикации/подписки на основе системы очередей сообщений, такой как MQSeries компании IBM.
4. Во что на самом деле разрешается имя темы в TIB/Rendezvous и как

протекает это разрешение?

5. Опишите простую реализацию полностью упорядоченной доставки сообщений в системе TIB/Rendezvous.
6. Когда в ходе транзакции TIB/Rendezvous сообщение доставляется процессу P, процесс подтверждает доставку. Можно ли на уровне транзакций сделать так, чтобы подтверждение отсылалось демону транзакций автоматически?
7. Предположим, что процесс в системе TIB/Rendezvous реплицируется. Приведите два решения, позволяющие исключить случаи публикации сообщений реплицированными процессами более одного раза.
8. Насколько необходима полностью упорядоченная групповая рассылка при репликации процессов в системе TIB/Rendezvous?
9. Опишите простую схему для PGM, которая позволяла бы получателям обнаруживать пропавшие сообщения, даже если пропадет первое сообщение в последовательности.
10. Могут ли журналы регистрации в TIB/Rendezvous, реализованные в виде файлов, гарантировать, что сообщение, даже при наличии отказавших процессов, не будет потеряно?

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)**

Темы для подготовки к экзамену (зачету) по дисциплине «Архитектура современных распределенных систем»

направление 230100.68 – «Информатика и вычислительная техника»
(магистратура)

Факультет информатики
Кафедра информационных систем и технологий

Разработал:
Востокин С.В.,
профессор кафедры ИСТ

Самара 2013 г.

Список вопросов

1. Определение распределенной системы. Назначение распределенных систем.
2. Основные аппаратные и программные решения, используемые в распределенных системах.
3. Уровни протоколов: низкоуровневые транспортные, верхнего уровня.
4. Модель клиент-сервер. Варианты архитектуры клиент-сервер.
5. Удаленный вызов процедур (RPC): базовые операции, передача параметров, расширенные модели RPC.
6. Обращение к удаленным объектам (RMI): привязка клиента к объектам, статическое и динамическое удаленное обращение к методам, передача параметров.
7. Связь посредством сообщений: сохранность, синхронность, нерезидентная связь.
8. Связь на основе потоков данных: поддержка непрерывных сред, качество обслуживания, синхронизация потоков данных.
9. Внутренняя организация процессов в распределенных системах. Потоки выполнения в распределенных системах.
10. Организация клиентского программного обеспечения: пользовательские интерфейсы, обеспечение прозрачности распределения.
11. Организация серверного программного обеспечения, серверы объектов.
12. Программные агенты, многоагентные системы.
13. Именованные сущности: имена, идентификаторы, адреса. Разрешение имен. Реализация пространства имен.
14. Размещение мобильных сущностей. Именование и локализация сущностей. Иерархические подходы и подходы на основе базовой точки.
15. Удаление сущностей, на которые нет ссылок. Описание проблемы. Подсчет ссылок, организация списка ссылок, идентификация сущностей, на которые нет ссылок.
16. Алгоритмы синхронизации физических часов.
17. Логические часы. Отметки Лампорта, векторные отметки.
18. Снимок глобального состояния в распределенной системе.
19. Алгоритмы голосования: кольцевой алгоритм, алгоритм забияки.
20. Алгоритмы взаимного исключения: централизованный, распределенный, маркерного кольца.
21. Распределенные транзакции: модель транзакции, классификация транзакций, реализация.

22. Модели непротиворечивости, ориентированные на данные.
23. Модели непротиворечивости, ориентированные на клиента.
24. Протоколы распределения и протоколы непротиворечивости.
25. Понятие отказоустойчивости, модели отказов, маскирование ошибок при помощи избыточности.
26. Надежная связь клиент-сервер, семантика RPC при возникновении ошибок.
27. Надежная групповая рассылка: базовые схемы, масштабируемость, атомарная групповая рассылка.
28. Распределенное подтверждение: двухфазный и трехфазный протоколы подтверждения.
29. Восстановление: создание контрольных точек, протоколирование сообщений.
30. Отказоустойчивость процессов. Группы процессов, членство в группе, соглашения в системах с ошибками.
31. Общие вопросы защиты распределенных систем: угрозы, правила, механизмы, криптография.
32. Защищенные каналы: аутентификация, целостность и конфиденциальность сообщений. Защищенное групповое взаимодействие.
33. Контроль доступа, брандмауэры, защита мобильного кода.
34. Управление защитой: управление ключами, защищенными группами, авторизацией.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)**

Тесты для итогового контроля знаний по дисциплине «Архитектура современных распределенных систем»

направление 230100.68 – «Информатика и вычислительная техника»
(магистратура)

Факультет информатики
Кафедра информационных систем и технологий

Разработал:
Востокин С.В.,
профессор кафедры ИСТ

Самара 2013 г.

Дисциплина «Архитектура современных распределенных систем»

Студент _____ группа_____

Вариант 1

Вопрос 1	**LAN используются для:**	**Балл**
1	**Связи между объектами**	
2	**Удаленного вызова методов**	
3	**Соединения близко расположенных компьютеров**	
4	**Для организации глобальных коммуникаций**	

Вопрос 2	**Сетевая операционная система**	**Балл**
1	**Управляет сетевым оборудованием**	
2	**Управляет группой компьютеров**	
3	**Управляет распределением памяти**	
4	**Объединяет группу компьютеров, каждый из которых работает под управлением своей операционной системы**	

Вопрос 3	**В модели клиент-сервер**	**Балл**
1	**Клиент — активный компонент, а сервер - пассивный**	
2	**Клиент — пассивный компонент, а сервер - активный**	
3	**Клиент и сервер — равноправные компоненты**	
4	**Сервер управляет пользовательским интерфейсом**	

Вопрос 4	**Какое допущение *не верно* в децентрализованных алгоритмах**	**Балл**
1	**Ни одна машина не обладает полной информацией о состоянии системы**	
2	**Машины принимают решение на основе локальной информации**	
3	**Сбой на одной машине не вызывает нарушений алгоритма**	
4	**Используется допущение о существовании единого времени**	

Вопрос 5	**Транспортный протокол для интернета называется**	**Балл**
1	**Универсальный протокол датаграмм UDP**	
2	**Протокол Интернета IP**	
3	**Асинхронный способ передачи данных ATM**	
4	**Протокол передачи файлов FTP**	

Вопрос 6	**Реализация интерфейса объекта, вызываемая на машине клиента, называется**	**Балл**
1	**прокси**	
2	**скелетон**	
3	**делегат**	
4	**заместитель**	

Вопрос 7	Ориентированный на сообщения промежуточный уровень МОМ реализует	Балл
1	Нерезидентную синхронную связь	
2	Асинхронную сохранную связь	
3	Нерезидентную синхронную связь с синхронизацией по приему	
4	Нерезидентную синхронную связь с синхронизацией по доставке	

Вопрос 8	В непрерывной среде исполнения важны	Балл
1	Абсолютные скорости доставки	
2	Латентность	
3	Пропускная способность	
4	Временные соотношения между различными элементами данных	

Вопрос 9	В этом режиме необходимо учитывать предельное дрожание	Балл
1	Асинхронный режим передачи	
2	Синхронный режим передачи	
3	Изохронный режим передачи	
4	Комплексный потоковый режим передачи	

Вопрос 10	Миграция в распределенных системах обычно реализуется в форме	Балл
1	Переноса кода	
2	Переноса процессов	
3	Переноса данных	
4	Переноса сообщений	

Вопрос 11	Перенос сегмента кода, возможно с данными для инициализации допускается моделью	Балл
1	Слабой мобильности	
2	Клиент-сервер	
3	Сильной мобильности	
4	Удаленного вызова процедуры	

Вопрос 12	Помогает конечному пользователю:	Балл
1	**Мобильный агент**	
2	**Кооперативный агент**	
3	**Интерфейсный агент**	
4	**Информационный агент**	

Вопрос 13	Имя точки доступа сущности называется	Балл
1	**Адресом**	
2	**Именем сущности**	
3	**Идентификатором**	
4	**Правильным идентификатором**	

Вопрос 14	Сохранение в листовом узле графа именования абсолютного пути реализуется при помощи -	Балл
1	**Псевдоним**	
2	**Символическая ссылка**	
3	**Жесткая ссылка**	
4	**Точка монтирования**	

Вопрос 15	В технологии мобильного IP-адреса обычно используется	Балл
1	**Широковещательная (групповая) рассылка**	
2	**Пересылка указателей**	
3	**Создание базы сущности**	
4	**Иерархическое дерево поиска**	

Вопрос 16	Свойства транзакций ACID – это	Балл
1	**Атомарность, сохранность, изолированность, долговечность**	
2	**Гранулярность, непротиворечивость, изолированность, долговечность**	
3	**Атомарность, непротиворечивость, зависимость, долговечность**	
4	**Атомарность, непротиворечивость, изолированность, долговечность**	

Вопрос 17	Всякое чтение возвращает результат последней записи	Балл
1	**Строгая непротеворечивость**	
2	**Последовательная непротеворечивость**	
3	**Причинная непротеворечивость**	
4	**Непротеворечивость FIFO**	

Вопрос 18	Вероятность того, что система в данный момент времени будет правильно работать	Балл
1	**Доступность**	
2	**Безотказность**	
3	**Безопасность**	
4	**Ремонтопригодность**	

Дисциплина «Архитектура современных распределенных систем»

Студент _____ группа_____

Вариант 2

Вопрос 1	**WAN это**	**Балл**
1	**Глобальная сеть**	
2	**Облачные вычисления**	
3	**Всемирная паутина**	
4	**Локальная сеть**	

Вопрос 2	**Мультипроцессор это**	**Балл**
1	**Много процессоров с разделяемой памятью**	
2	**Много процессоров с распределенной памятью**	
3	**Одновременная обработка**	
4	**Одновременные вычисления**	

Вопрос 3	**Прозрачность доступа**	**Балл**
1	**Скрывает факт перемещения ресурса**	
2	**Скрывает факт репликации ресурса**	
3	**Скрывает отказ и восстановление ресурса**	
4	**Скрывает разницу в представлении к данных и доступе к ресурсу**	

Вопрос 4	**Уровни протоколов определяет**	**Балл**
1	**Модель передачи сообщений**	
2	**Модель уровней**	
3	**Разработчик распределенной системы**	
4	**Эталонная модель взаимодействия открытых систем**	

Вопрос 5	**Удаленный вызов процедур**	**Балл**
1	**Позволяет процессу, запущенному на машине А вызывать процедуру на машине В**	
2	**Позволяет процедуре, запущенной на машине А обращаться к машине В**	
3	**Позволяет реализовать в общую память**	
4	**Позволяет реализовать в глобальную разделяемую память**	

Вопрос 6	**Серверная заглушка в RMI называется**	**Балл**
1	**прокси**	
2	**скелетон**	
3	**делегат**	
4	**заместитель**	

Вопрос 7	Сокеты Беркли реализуют	Балл
1	Нерезидентную связь на основе сообщений	
2	Сохранную связь на основе сообщений	
3	Асинхронную связь на основе RPC	
4	Синхронную связь на основе RPC	

Вопрос 8	Элементы данных передаются в поток последовательно, других ограничений не накладывается. Это ...	Балл
1	Асинхронный режим передачи	
2	Синхронный режим передачи	
3	Изохронный режим передачи	
4	Комплексный потоковый режим передачи	

Вопрос 9	Требования к качеству обслуживания QoS обычно выражают	Балл
1	Временные зависимости и другие нефункциональные требования	
2	Функциональные требования к системе	
3	Стоимость обслуживания	
4	Отказоустойчивость	

Вопрос 10	Часть, содержащая набор инструкций	Балл
1	Сегмент кода	
2	Сегмент ресурсов	
3	Сегмент исполнения	
4	Сегмент потока	

Вопрос 11	Перенос сегмента кода и сегмента исполнения допускается моделью	Балл
1	Слабой мобильности	
2	Клиент-сервер	
3	Сильной мобильности	
4	Удаленного вызова процедуры	

Вопрос 12	Управляет информацией из разных источников:	Балл
1	Мобильный агент	
2	Кооперативный агент	
3	Интерфейсный агент	
4	Информационный агент	

Вопрос 13	Разрешение имени это	Балл
1	Назначение имени сущности	
2	Процесс доступа к именованной сущности	
3	Назначение прав доступа по имени	
4	Снятие запрета с имени сущности	

Вопрос 14	Направляющий узел графа именования во внешнем пространстве имен	Балл
1	Псевдоним	
2	Символическая ссылка	
3	Жесткая ссылка	
4	Точка монтирования	

Вопрос 15	Векторные отметки времени используются для	Балл
1	Синхронизации часов	
2	Определения реального времени	
3	Передачи причинно-следственной связи	
4	Однозначного упорядочивания событий в системе	

Вопрос 16	Серии операций, удовлетворяющая свойствам ACID - это	Балл
1	Плоские транзакции	
2	Вложенные транзакции	
3	Распределенные транзакции	
4	Взаимное исключение	

Вопрос 17	Все процессы видят одно и тоже чередование операций чтения-записи	Балл
1	Строгая непротеворечивость	
2	Последовательная непротеворечивость	
3	Причинная непротеворечивость	
4	Непротеворечивость FIFO	

Вопрос 18	Свойство работать без отказа в течении длительного интервала времени	Балл
1	Доступность	
2	Безотказность	
3	Безопасность	
4	Ремонтопригодность	

Дисциплина «Архитектура современных распределенных систем»

Студент _____ группа_____

Вариант 3

Вопрос 1	**Распределенную систему характеризуют**	**Балл**
1	**Сетевое взаимодействие**	
2	**Распределенное хранение данных**	
3	**Распределенные вычисления**	
4	**Автономность компьютеров и единство системы с точки зрения пользователя**	

Вопрос 2	**Мультикомпьютер это**	**Балл**
1	**Много процессоров с разделяемой памятью**	
2	**Много процессоров с распределенной памятью**	
3	**Одновременная обработка**	
4	**Одновременные вычисления**	

Вопрос 3	**Прозрачность переноса**	**Балл**
1	**Скрывает факт перемещения ресурса**	
2	**Скрывает факт репликации ресурса**	
3	**Скрывает отказ и восстановление ресурса**	
4	**Скрывает разницу в представлении к данных и доступе к ресурсу**	

Вопрос 4	**Основной задачей сетевого уровня является**	**Балл**
1	**Стандартизация электрических, механических и сигнальных интерфейсов**	
2	**Маршрутизация пакетов**	
3	**Обеспечение надежной доставки пакетов**	
4	**Реализация протокола управления передачей**	

Вопрос 5	**Прозрачность RPC реализуется путем применения**	**Балл**
1	**заглушек**	
2	**наследования**	
3	**замещения**	
4	**указателя**	

Вопрос 6	**Сохранный объект**	**Балл**
1	**Объект, который существует, пока сервер управляет им**	
2	**Продолжает существовать вне адресного пространства серверного процесса**	
3	**Это нерезидентный объект**	
4	**Это объект, используемый в RMI**	

Вопрос 7	Интерфейс передачи сообщений реализует	Балл
1	**Нерезидентную связь на основе сообщений**	
2	**Сохранную связь на основе сообщений**	
3	**Асинхронную связь на основе RPC**	
4	**Синхронную связь на основе RPC**	

Вопрос 8	Несколько связанных простых потоков данных используется в этом режиме	Балл
1	**Асинхронный режим передачи**	
2	**Синхронный режим передачи**	
3	**Изохронный режим передачи**	
4	**Комплексный потоковый режим передачи**	

Вопрос 9	Какая архитектура ***не используется*** для построения серверов	Балл
1	**Многопоточный процесс**	
2	**Однопоточный процесс**	
3	**Конечный автомат**	
4	**Легковесный процесс**	

Вопрос 10	Часть, содержащая ссылки на внешние ресурсы	Балл
1	**Сегмент кода**	
2	**Сегмент ресурсов**	
3	**Сегмент исполнения**	
4	**Сегмент потока**	

Вопрос 11	Автономный процесс, способный реагировать на среду исполнения и вызывать изменения в ней это	Балл
1	**Клиент**	
2	**Сервер**	
3	**Объект**	
4	**Программный агент**	

Вопрос 12	Управляет информацией из нескольких источников:	Балл
1	**Мобильный агент**	
2	**Кооперативный агент**	
3	**Интерфейсный агент**	
4	**Информационный агент**	

Вопрос 13	Другое имя той же сущности	Балл
1	**Псевдоним**	
2	**Символическая ссылка**	
3	**Жесткая ссылка**	
4	**Точка монтирования**	

Вопрос 14	Уборка мусора это -	Балл
1	Процесс удаления имен, которые нельзя более локализовать	
2	Процесс разрешения имени	
3	Процесс кэширования указателей	
4	Процесс локализации имени	

Вопрос 15	Отметки времени Лампорта используются для	Балл
1	Синхронизации часов	
2	Определения реального времени	
3	Передачи причинно-следственной связи	
4	Однозначного упорядочивания событий в системе	

Вопрос 16	Транзакции, которые логически разделяются на иерархически организованные транзакции	Балл
1	Плоские транзакции	
2	Вложенные транзакции	
3	Распределенные транзакции	
4	Взаимное исключение	

Вопрос 17	Операции записи, связанные причинно-следственными связями наблюдаются всеми в одинаковом порядке	Балл
1	Строгая непротеворечивость	
2	Последовательная непротеворечивость	
3	Причинная непротеворечивость	
4	Непротеворечивость FIFO	

Вопрос 18	Определяет, насколько катастрофична ситуация временной неработоспособности системы	Балл
1	Доступность	
2	Безотказность	
3	Безопасность	
4	Ремонтопригодность	

Дисциплина «Архитектура современных распределенных систем»

Студент _____ группа_____

Вариант 4

Вопрос 1	**Распределенная операционная система**	**Балл**
1	**Управляет сетевым оборудованием**	
2	**Управляет группой компьютеров**	
3	**Управляет распределением памяти**	
4	**Объединяет группу компьютеров, каждый из которых работает под управлением своей операционной системы**	

Вопрос 2	**В распределенных системах с промежуточным уровнем**	**Балл**
1	**Этот уровень находится к клиентском узле**	
2	**Этот уровень — часть операционной системы**	
3	**Этот уровень распределен между всеми компьютерами**	
4	**Он находится на серверном узле**	

Вопрос 3	**Прозрачность отказа**	**Балл**
1	**Скрывает факт перемещения ресурса**	
2	**Скрывает факт репликации ресурса**	
3	**Скрывает отказ и восстановление ресурса**	
4	**Скрывает разницу в представлении к данных и доступе к ресурсу**	

Вопрос 4	**Транспортный протокол для интернета называется**	**Балл**
1	**Протокол управления передачей TCP**	
2	**Протокол Интернета IP**	
3	**Асинхронный способ передачи данных ATM**	
4	**Протокол передачи файлов FTP**	

Вопрос 5	**Упаковка параметров вызова RPC называется**	**Балл**
1	**Сборкой параметров**	
2	**Передачей параметров**	
3	**Маршалингом параметров**	
4	**Компьютингом параметров**	

Вопрос 6	**Нерезидентный объект**	**Балл**
1	**Объект, который существует, пока сервер управляет им**	
2	**Продолжает существовать вне адресного пространства серверного процесса**	
3	**Это сохранный объект**	
4	**Это объект, используемый в RMI**	

Вопрос 7	Специфика модели очередей сообщений в том, что допускается это состояние при доставке сообщений	Балл
1	Отправитель- активен, получатель-активен	
2	Отправитель- активен, получатель-пассивен	
3	Отправитель- пассивен, получатель-активен	
4	Отправитель- пассивен, получатель-пассивен	

Вопрос 8	Для каждого элемента в потоке данных определяется задержка сквозной передачи. Это ...	Балл
1	Асинхронный режим передачи	
2	Синхронный режим передачи	
3	Изохронный режим передачи	
4	Комплексный потоковый режим передачи	

Вопрос 9	Сервер объектов ориентирован на	Балл
1	Поддержку распределенных объектов	
2	Удаленный вызов процедур	
3	Рассылку сообщений	
4	Управление очередью сообщений	

Вопрос 10	Часть, содержащая текущее состояние процесса	Балл
1	Сегмент кода	
2	Сегмент ресурсов	
3	Сегмент исполнения	
4	Сегмент потока	

Вопрос 11	Может перемещаться между машинами:	Балл
1	Мобильный агент	
2	Кооперативный агент	
3	Интерфейсный агент	
4	Информационный агент	

Вопрос 12	Составляет часть мультиагентной системы:	Балл
1	Мобильный агент	
2	Кооперативный агент	
3	Интерфейсный агент	
4	Информационный агент	

Вопрос 13	Предоставление нескольких абсолютных путей к каждому узлу графа именования	Балл
1	Псевдоним	
2	Символическая ссылка	
3	Жесткая ссылка	
4	Точка монтирования	

Вопрос 14	Служба каталогов - это	Балл
1	Разновидность службы именования	
2	Место хранения файлов	
3	Разновидность файловой системы	
4	Список ресурсов	

Вопрос 15	Основным свойством распределенного глобального снимка состояния является	Балл
1	Состоит из локальных состояний процессов	
2	Является распределенным	
3	Синхронность	
4	Непротиворечивость временного среза	

Вопрос 16	Транзакции, логически представляющие собой неделимые транзакции, которые работают с распределенными данными	Балл
1	Плоские транзакции	
2	Вложенные транзакции	
3	Распределенные транзакции	
4	Взаимное исключение	

Вопрос 17	Операции записи осуществляемые единичным процессом наблюдаются всеми в одном порядке	Балл
1	Строгая непротеворечивость	
2	Последовательная непротеворечивость	
3	Причинная непротеворечивость	
4	Непротеворечивость FIFO	

Вопрос 18	Определяет, насколько сложно исправить неполадки	Балл
1	Доступность	
2	Безотказность	
3	Безопасность	
4	Ремонтопригодность	

Ответы к тестам

Вариант № 1

№ вопроса	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
№ ответа	3	4	1	4	1	1	2	4	3	2	1	3	1	2	3	4	1	1

Вариант № 2

№ вопроса	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
№ ответа	1	1	4	4	1	2	1	1	1	1	3	4	2	4	3	1	1	2

Вариант № 3

№ вопроса	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
№ ответа	4	2	1	2	1	2	1	4	4	2	4	4	1	1	4	2	3	3

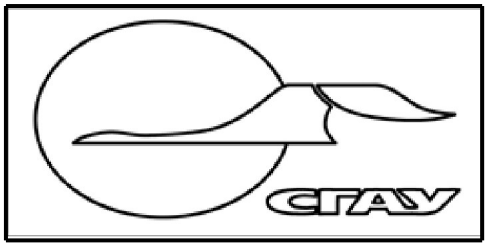
Вариант № 4

№ вопроса	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
№ ответа	2	3	3	1	3	1	4	2	1	3	1	2	3	1	4	3	4	4

Критерии оценивания тестов

Оценка	Количество верных ответов	
	не менее	не более
неудовлетворительно	0	7
удовлетворительно	7	12
хорошо	12	15
отлично	15	18

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)



СОГЛАСОВАНО

Управление образовательных программ

_____/ А.В. Дорошин /

"_____"_____ 20___ г.

УТВЕРЖДАЮ

Проректор по учебной работе

_____/ Ф.В. Гречников /

"____" _____ 20___ г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Наименование модуля (дисциплины)

Архитектура современных распределённых систем

Цикл, в рамках которого происходит освоение модуля (дисциплины)	М2. Профессиональный цикл
Часть цикла	М2.В.ОД Обязательные дисциплины

Код учебного плана

230100_68_2-13-56-3022

Факультет	6
Кафедра	Информационные системы и технологии

Курс	5		
Семестр	А		
Лекции (СЛ)	18		
Семинарские и практические занятия (СП)	0		
Лабораторные занятия (СЛР)	36	Экзамен	
Контроль самостоятельной работы / Индивидуальные занятия (КСР / ИЗ)	0	Зачет	А
Самостоятельная работа (СРС)	18		
Всего (Всего с экзаменами)	72		

Наименование стандарта, на основании которого составлена рабочая программа:

Магистерская программа "Программное обеспечение мобильных устройств" 230100.68 "Информатика и вычислительная техника"

Соответствие содержания рабочей программы, условий ее реализации, материально-технической и учебно-методической обеспеченности учебного процесса по дисциплине всем требованиям государственных стандартов подтверждаем.

Составители:

Востокин С.В., д.т.н., доц. _____
(подпись)

Заведующий кафедрой:

Прохоров С.А., д.т.н., проф. _____
(подпись)

Рабочая программа обсуждена на заседании кафедры

Информационные системы и технологии

Протокол №___ от "____" _____ 20___ г.

Наличие основной литературы в фондах научно-технической библиотеки (НТБ) подтверждаем:

Директор НТБ _____ /_____/
(подпись) (расшифровка подписи)

Согласовано:

Декан _____ /_____/
(подпись) (расшифровка подписи)

1 Цели и задачи модуля (дисциплины), требования к уровню освоения содержания

1.1 Перечень развиваемых компетенций

Коды компетенций из ФГОС-3 " Информатика и вычислительная техника" 230100: ПК-1, ПК-3, ПК-4, ПК-5.

1.2 Цели и задачи изучения модуля (дисциплины)

Целью данного курса является получения общих сведений об архитектурах современных распределённых систем и принципах их функционирования, а также практических навыков их использования.

1.3 Требования к уровню подготовки студента, завершившего изучение данного модуля (дисциплины)

Студенты, завершившие изучение данной дисциплины, должны знать: функциональное назначение, архитектуру, принципы управления ресурсами в современных распределённых системах; уметь: разрабатывать прикладные программы и библиотеки с использованием интерфейсов прикладного программирования распределённых систем, решать конкретные инженерные и исследовательские задачи, требующие применения навыков системного программирования.

1.4 Связь с предшествующими модулями (дисциплинами)

Курс базируется на сведениях из курсов:
- Архитектура современных операционных систем
- Теоретические основы телекоммуникаций.

1.5 Связь с последующими модулями (дисциплинами)

Курс «Архитектура современных распределённых систем» дает необходимые сведение для выполнения проектов и работ, требующих навыков системного программирования для направления подготовки 230100 «Информатика и вычислительная техника», используется в курс
ах: «Методы проектирования и поддержки требований к программному обеспечению», а также при выполнении выпускной квалификационной работы магистра.

2 Содержание рабочей программы (модуля)

Семестр 1		
СЛ 0,1667 18 часов 0,5001 ЗЕТ	Активные 0	
	Интерактивные 0	

	Традиционные 1	1. Введение в распределённые системы.
		2. Связь.
		3. Процессы.
		4. Именование.
		5. Синхронизация.
		6. Непротиворечивость и репликация.
		7. Отказоустойчивость.
		8. Защита.
		9. Распределенные системы объектов.
		10. Распределенные файловые системы.
		11. Распределенные системы документов.
		12. Распределенные системы согласования.
СП 0 0 часов 0 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 0	
СЛР 0,3333 36 часов 0,9999 ЗЕТ	Активные 0	
	Интерактивные 1	Лаб.раб.№1. Удаленные объекты Java RMI.
		Лаб.раб.№2. Система координации Jini
		Лаб.раб.№3. Распределенные объекты Templet
	Традиционные 0	
КСР 0 0 часов 0 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 0	
СРС 0,1667 18 часов 0,5001 ЗЕТ	Активные 1	1.Самостоятельная работа с литературой по предмету.
		2.Изучение примеров приложений по теме лаб.работ.
		3.Подготовка итогового письменного отчета по лаб.работам.
		4.Подготовка к зачету по теоретическим вопросам.
	Интерактивные 0	
	Традиционные 0	

3 Инновационные методы обучения

Для развития профессиональных навыков, необходимых обучающимся, Программа предполагает широкое использование активных и интерактивных форм проведения занятий: дискуссий, презентаций, конференций, проектной работы. При подаче лекционного материала исп
ользуются мультимедиа материалы. Программа предполагает выполнение дополнительных заданий с элементами исследования (разработка подсистем распределённой системы автоматизации параллельного программирования). В лабораторном практикуме ведется работа с электронной технической документацией через сеть Интернет.

4 Технические средства и материальное обеспечение учебного процесса

Компьютерный класс с компьютерами, имеющими подключение к сети Интернет. Программное обеспечение: ОС MS Windows XP/Vista/7 (лицензии СГАУ), MS Visual Studio 2005 Academic Edition или MS Visual Studio 2008 Express (лицензии СГАУ, бесплатно для учебного использования).

5 Учебно-методическое обеспечение

5.1 Основная литература

1. Таненбаум, Э. Распределенные системы. Принципы и парадигмы [Текст] / Э. Таненбаум, М. ван Стеен. - СПб. [и др.] : Питер , 2003. - 877 с. - (Классика computer science). - ISBN 5-272-00053-6 (3 экз.)
2. Гордеев, А.В. Операционные системы [Текст] : [учеб. для вузов по направлению подгот. бакалавров и магистров "Информатика и вычисл. техника" и направлению подгот. дипломир. специалистов "Информатика вычисл.техника"] / А. В. Гордеев. - 2-е изд. - СПб.
и др. : Питер : Питер принт, 2004. - 415 с. - (Учебник для вузов). - ISBN 5-94723-632-X (гриф Минобразования России) (20 экз.)
3. Олифер, В.Г. Сетевые операционные системы [Текст] : [учеб. пособие для вузов по направлению подгот. дипломир. специалистов "Информатика и вычисл. техника] / В. Г. Олифер, Н. А. Олифер. - СПб. [и др.] : Питер : Питер Пресс, 2007. - 538 с. - (Учебник
для вузов). - ISBN 5-272-00120-6 (гриф Минобразования России) (10 экз.)
4.Востокин, С.В. Графическая объектная модель параллельных процессов и ее применение в задачах численного моделирования [Текст] / С.В. Востокин. Изд-во Самарского научного центра РАН – Самара, 2007. 186 с., ил. – ISBN 978-5-93424-284-9 (37 экз.)

5.2 Дополнительная литература

1. Востокин, С.В. Вопросы, задания и упражнения по курсу "Операционные

системы" [Текст] : [лаб. практикум] / М-во образования и науки Рос. Федерации, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т) ; [сост. С. В. Востокин]. - Самара : Изд-во СГАУ, 2012. - 29 с. (20 экз.)
2. Востокин, С.В. Операционные системы [Электронный ресурс] : [учеб. для вузов по направления подгот. бакалавров "Инф. и выч. техн.", "Фундам. информатика и информ. технологии", "Прикладная математика и информатика", "Прикладная математика и физика"] / С.В. Востокин ; М-во образования и науки РФ, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т). - Электрон. текстовые дан. - Самара : Изд-во СГАУ, 2012. – ISBN 978-5-7883-0916-3 (2 эл. опт. диска)
3. Таненбаум, Э.Современные операционные системы [Текст] / Эндрю Таненбаум. - 2-е изд. - СПб. [и др.] : Питер : Питер принт, 2005. - 1037 с. - (Классика computer science). - ISBN 5-318-00299-4 (5 экз.)
4. Таненбаум, Э. Современные операционные системы [Текст] / Эндрю Таненбаум. - 2-е изд., [перераб. и испр.]. - СПб. [и др.] : Питер : Питер Пресс, 2007. - 1037 с. - (Классика computer science). - ISBN 978-5-318-00299-1 (2 экз.)
5. Солдатова, О. П. Системное программирование [Текст] : Курс лекций / О. П. Солдатова, С. В. Востокин ; Самар. гос. аэрокосм. ун-т им. С. П. Королева. - Самара : [б. и.], 2002. - 123 с. - ISBN 5-7883-0226-9 (94 экз.)

5.3 Электронные источники и интернет ресурсы

1. Microsoft Developer Network http://msdn.microsoft.com
2. www.prenhall.com/tanenbaum
3. Проект автоматизации параллельных и распределенных вычислений «Темплет» http://templet.ssau.ru
4. Интуит национальный открытый университет. intuit.ru

5.4 Методические указания и рекомендации

Текущий контроль знаний студентов завершается на отчетном занятии, результатом которого является допуск или недопуск студента к зачёту по дисциплине. Основанием для допуска к зачёту является выполнение и отчет студента по всем лабораторным работам.

Зачёт проводится согласно положению о текущем и промежуточном контроле знаний студентов, утвержденному ректором университета. Зачёт ставится на основании письменного и устного ответов студента по тестовому заданию, а также, при необходимости, ответов на дополнительные вопросы. Тестовое задание включает 16 вопросов. Дополнительно может быть предложен как теоретический вопрос, так и вопрос на понимание программ по листингам.