Мариэлле, Максу и Элке.
– Мартен ван Стин

Сюзанне, Барбаре, Мэрвину, Арону, Натану, Оливии и Мирте.
– Эндрю С. Таненбаум

Содержание

Предисловие 14

От издательства 15

Глава 1. Введение 16

1.1. Что такое распределенная система? 17
Характеристика 1: совокупность автономных вычислительных элементов 17
Характеристика 2: единая цельная система 19
Промежуточное программное обеспечение и распределенные системы 20
1.2. Цели дизайна 22
Поддержка совместного использования ресурсов 22
Создание прозрачных распределений 23
Типы прозрачности распределений 23
Степень прозрачности распределения 26
Открытость 27
Функциональная совместимость, компоновка и расширяемость 28
Отделение политики от механизма 29
Масштабирование 30
Размерность масштабируемости 31
Техника масштабирования 36
Ловушки 39
1.3. Типы распределенных систем 40
Высокопроизводительные распределенные вычисления 41
Кластерные вычисления 42
Сетевые вычисления 44
Облачные вычисления 46
Распределенные информационные системы 50
Распределенная обработка транзакций 50
Интеграция корпоративных приложений 53
Распространенные системы 56
Повсеместно распространенные вычислительные системы 56
Мобильные вычислительные системы 59
Сенсорные сети 63
1.4. Резюме 68

Глава 2. Архитектуры 70

2.1. Архитектурные стили 71
Многоуровневая архитектура 72
Многоуровневые протоколы связи 73

Уровни приложений ... 75
Объектно-ориентированные и сервис-ориентированные архитектуры ... 77
Ресурсные архитектуры ... 79
Архитектура публикация-подписка ... 82
2.2. Организация промежуточного программного обеспечения ... 87
Упаковщики ... 87
Перехватчики ... 89
Модифицируемое промежуточное ПО ... 90
2.3. Системная архитектура ... 91
Централизованные организации ... 92
Простая архитектура клиент-сервер ... 92
Многоуровневая архитектура ... 93
Децентрализованные организации: одноранговые системы ... 96
Структурированные одноранговые системы ... 97
Неструктурированные одноранговые системы ... 100
Иерархически организованные одноранговые сети ... 103
Гибридные архитектуры ... 105
Системы пограничных серверов ... 106
Совместные распределенные системы ... 107
2.4. Примеры архитектур ... 110
Сетевая файловая система NFS ... 110
Веб ... 114
Простые веб-системы ... 114
Многоуровневые архитектуры ... 116
2.5. Резюме ... 117

Глава 3. Процессы ... 119
3.1. Потоки ... 120
Введение в потоки ... 120
Использование потоков в нераспределенных системах ... 122
Реализация потоков ... 125
Потоки в распределенных системах ... 128
Многопоточные клиенты ... 128
Многопоточные серверы ... 130
3.2. Виртуализация ... 132
Принцип виртуализации ... 133
Виртуализация и распределенные системы ... 133
Типы виртуализации ... 135
Применение виртуальных машин в распределенных системах ... 139
3.3. Клиенты ... 141
Сетевые пользовательские интерфейсы ... 141
Пример: система X Window ... 141
Сетевые вычисления для тонких клиентов ... 143
Клиентское программное обеспечение для прозрачности распространения ... 144
3.4. Серверы ... 146
Общие вопросы дизайна ... 146

Параллельный сервер против итеративного сервера ... 146
Связь с сервером: конечные точки ... 146
Прерывание сервера ... 148
Серверы без сохранения состояния против серверов, сохраняющих состояние ... 148
Объектные серверы ... 150
Пример: веб-сервер Apache ... 157
Кластеры серверов ... 159
Локальные кластеры ... 159
Общая организация ... 159
Глобальные кластеры ... 163
Пример использования: PlanetLab ... 167
V-серверы ... 169
3.5. Миграция кода ... 170
Причины переноса кода ... 171
Миграция в гетерогенных системах ... 176
3.6. Резюме ... 179

Глава 4. Коммуникации ... 182
4.1. Основы ... 183
Многоуровневые протоколы ... 183
Эталонная модель OSI ... 183
Протоколы промежуточного программного обеспечения ... 189
Типы коммуникаций ... 190
4.2. Удаленный вызов процедуры ... 192
Основная операция RPC ... 193
Передача параметров ... 198
Поддержка приложений на основе RPC ... 202
Генерация заглушки ... 202
Языковая поддержка ... 203
Вариации RPC ... 205
Асинхронный RPC ... 205
Многоадресный RPC ... 206
Пример: распределенная вычислительная среда RPC ... 207
Введение в распределенную вычислительную среду DCE ... 208
Цели DCE RPC ... 208
Написание клиента и сервера ... 209
Привязка клиента к серверу ... 211
Выполнение RPC ... 212
4.3. Коммуникации, ориентированные на сообщения ... 212
Простой временный обмен сообщениями с сокетами ... 213
Расширенный переходный обмен сообщениями ... 218
Использование шаблонов обмена сообщениями: ZeroMQ ... 218
Интерфейс передачи сообщений (MPI) ... 223
Постоянная связь, ориентированная на сообщения ... 226
Модель очереди сообщений ... 226
Общая архитектура системы очереди сообщений ... 228

Брокеры сообщений ... 230
Пример: система очереди сообщений IBM WebSphere ... 233
Обзор ... 233
Каналы ... 234
Передача сообщений ... 235
Управление оверлейными сетями ... 237
Пример: расширенный протокол очереди сообщений (AMQP) ... 238
Основы ... 239
AMQP связи ... 239
AMQP обмена сообщениями ... 241
4.4. Многоадресная связь ... 242
Многоадресная рассылка на уровне дерева приложений ... 242
Проблемы с производительностью в оверлеях ... 243
Многоадресная передача сообщений на основе лавинной маршрутизации ... 246
Распространение данных по принципу сплетни ... 250
Модели распространения информации ... 250
Удаление данных ... 254
4.5. Резюме ... 255

Глава 5. Присваивание имен ... 257
5.1. Имена, идентификаторы и адреса ... 258
5.2. Бесструктурное (плоское) наименование ... 261
Простые решения ... 261
Широковещание ... 262
Прямые указатели ... 263
Методы домашнего местоположения ... 265
Распределенные хеш-таблицы ... 267
Общий механизм ... 267
Иерархические методы ... 271
5.3. Структурированное наименование ... 276
Пространства имен ... 277
Разрешение имени ... 279
Механизм закрытия ... 280
Связывание и монтаж ... 281
Реализация пространства имен ... 284
Распределение пространства имен ... 285
Реализация разрешения имен ... 287
Пример: система доменных имен ... 292
Пространство имен DNS ... 292
Реализация DNS ... 294
Пример: сетевая файловая система ... 298
5.4. Наименование на основе атрибутов ... 303
Службы каталогов ... 304
Иерархические реализации: протокол LDAP ... 305
Децентрализованные реализации ... 308
Использование распределенного индекса ... 309

Пространственные кривые 310
5.5. Резюме 315

Глава 6. Координация 317
6.1. Синхронизация часов 318
Физические часы 319
Алгоритмы синхронизации часов 323
Сетевой временной протокол 325
Алгоритм Беркли 326
Синхронизация часов в беспроводных сетях 327
6.2. Логические часы 330
Логические часы Лампорта 331
Пример: полностью упорядоченная многоадресная рассылка 333
Векторные часы 337
6.3. Взаимное исключение 342
Обзор 342
Централизованный алгоритм 343
Распределенный алгоритм 344
Алгоритм кольцо токенов 346
Децентрализованный алгоритм 347
6.4. Алгоритмы выбора 350
Алгоритм хулигана 351
Кольцевой алгоритм 352
Выборы в беспроводной среде 353
Выборы в масштабных системах 356
6.5. Системы локации 357
GPS: система глобального позиционирования 357
Когда GPS не выбор 359
Логическое позиционирование узлов 360
Централизованное позиционирование 361
Децентрализованное позиционирование 363
6.6. Сопоставление распределенных событий 364
Централизованные реализации 364
6.7. Координация на основе сплетен 370
Объединение 370
Служба одноранговой выборки 372
Структура оверлея, основанная на сплетнях 373
6.8. Резюме 374

Глава 7. Согласованность и репликация 377
7.1. Введение 378
Причины репликации 378
Репликация как метод масштабирования 379
7.2. Модели согласованности, ориентированные на данные 381
Непрерывное согласование 382
О понятии конит 383

Согласованный порядок операций ... 386
Последовательная согласованность ... 386
Причинная согласованность ... 391
Группирование операций ... 393
Согласованность и когерентность ... 395
Конечная согласованность ... 395
7.3. Модели согласованности, ориентированные на клиента ... 398
Монотонные чтения ... 400
Монотонные записи ... 402
Чтение собственных записей ... 404
Запись следует за чтением ... 405
7.4. Управление репликами ... 406
Поиск лучшего местоположения сервера ... 406
Репликация и размещение контента ... 408
Постоянные реплики ... 409
Реплики, инициированные сервером ... 409
Реплики, инициированные клиентом ... 411
Распространение контента ... 412
Состояние против операции ... 412
Протоколы извлечения и проталкивания ... 413
Одноадресная и многоадресная рассылка ... 416
Управление реплицированными объектами ... 417
7.5. Согласованность протоколов ... 420
Непрерывная последовательность ... 420
Ограничивающее числовое отклонение ... 420
Граничные отклонения устаревания ... 422
Ограничение отклонений порядка ... 422
Первичные протоколы ... 423
Протоколы удаленной записи ... 423
Протоколы локальной записи ... 424
Протоколы реплицируемой записи ... 426
Активная репликация ... 426
Протоколы на основе кворума ... 426
Протоколы кеширования ... 428
Реализация согласованности, ориентированной на клиента ... 432
7.6. Пример: кеширование и репликация в сети ... 434
7.7. Резюме ... 445

Глава 8. Отказоустойчивость ... 448
8.1. Введение в отказоустойчивость ... 449
Базовые концепции ... 449
Модели отказов ... 452
Маскировка отказов посредством избыточности ... 456
8.2. Устойчивость процесса ... 458
Устойчивость групповых процессов ... 458
Организация групп ... 458
Управление членством ... 459

Маскировка и репликация отказа ... 460
Консенсус в неисправных системах со сбоями ... 461
Пример: Paxos ... 463
Основная идея Paxos ... 464
Понимание Paxos ... 468
Консенсус в неисправных системах с произвольными отказами ... 475
Почему 3k процессов недостаточно ... 476
Почему достаточно 3k + 1 процессов ... 477
Пример: практическая византийская отказоустойчивость ... 481
Некоторые ограничения по реализации отказоустойчивости ... 484
Относительно достижения консенсуса ... 484
Согласованность, доступность и разделение ... 486
Обнаружение отказов ... 487
8.3. Надежная связь клиент-сервер ... 489
Двухточечная связь ... 490
Семантика RPC при наличии отказов ... 490
Клиент не может найти сервер ... 490
Потерянные сообщения запроса ... 491
Сбои сервера ... 491
Потерянные ответные сообщения ... 494
Клиент неисправен ... 495
8.4. Надежное групповое общение ... 496
Атомарная многоадресная рассылка ... 503
Виртуальная синхроннность ... 503
Порядок сообщений ... 505
8.5. Распределенная фиксация ... 509
8.6. Восстановление ... 517
Введение ... 517
Контрольная точка ... 520
Скоординированная контрольная точка ... 521
Независимая контрольная точка ... 521
Регистрация сообщений ... 523
Вычисления, ориентированные на восстановление ... 526
8.7. Резюме ... 526

Глава 9. Безопасность ... 529

9.1. Введение ... 530
Угрозы безопасности, политики и механизмы ... 530
Проблемы дизайна ... 532
Контроль ... 532
Уровни механизмов безопасности ... 533
Распределение механизмов безопасности ... 535
Простота ... 536
Криптография ... 537
9.2. Безопасные каналы ... 541
Аутентификация ... 541
Аутентификация на основе общего секретного ключа ... 542

Аутентификация с использованием центра распределения ключей 545
Аутентификация с использованием криптографии с открытым ключом 548
Целостность и конфиденциальность сообщений 549
Цифровые подписи 549
Сессионные ключи 551
Безопасное групповое общение 553
Конфиденциальное групповое общение 553
Безопасные реплицированные серверы 553
Пример: система Kerberos 556
9.3. Контроль доступа 558
Общие вопросы управления доступом 558
Матрица контроля доступа 559
Брандмауэры 562
Безопасный мобильный код 564
Отказ в обслуживании 568
9.4. Безопасное наименование 570
9.5. Управление безопасностью 571
Управление ключами 571
Установка ключей 572
Распространение ключей 573
Безопасное управление группами 575
Управление авторизацией 577
Возможности и атрибуты 577
Делегирование (прав) 580
9.6. Резюме 582

Предисловие

Это третье издание книги «Распределенные системы». Во многих отношениях она сильно отличается от предыдущих выпусков, и, возможно, самое важное состоит в том, что мы полностью обобщили «принципы» и «парадигмы», включив последние в соответствующие главы, где обсуждаются принципы распределенных систем.

Материал был существенно переработан и дополнен, и в то же время мы были заинтересованы в ограничении общего объема книги. Поэтому он был сокращен более чем на 10 % по сравнению со вторым изданием, в основном за счет удаления материала по парадигмам. Для лучшего понимания материала книги широким кругом читателей мы перенесли некоторую информацию в отдельные выделенные разделы. Эти разделы могут быть пропущены при первом чтении.

Еще одним важным отличием являются примеры кода на языке программирования Python с поддержкой коммуникаций посредством пакета Redis. Примеры в книге опускают много деталей для удобства чтения, но полные примеры доступны на веб-сайте www.distributed-systems.net. Вместе с кодами для запуска, тестирования и расширений алгоритмов сайт предоставляет доступ к слайдам, всем рисункам и упражнениям.

Новый материал был проверен в учебном процессе, за что мы выражаем особую благодарность Тилю Кильманну (Thilo Kielmann) из Амстердамского свободного университета. Его конструктивные и критические замечания помогли нам значительно улучшить книгу.

Наш издатель Pearson Education любезно вернул нам авторские права, и мы должны сказать большое спасибо Трейси Джонсон (Tracy Johnson) за то, что переход прошел гладко. Возврат авторских прав позволил нам начать то, что мы оба хотели сделать: запустить эксперимент. Он заключался в том, чтобы найти способ, обеспечивающий доступность материала, сделать его относительно недорогим и упростить процедуру обновления.

Книга теперь может быть (свободно) загружена, что делает намного более простым использование гиперссылок, где это уместно. В то же время предлагается и печатная версия, доступная через Amazon.com по минимальной цене.

Теперь это полностью электронная книга, что позволяет нам включать обновления, когда это необходимо. Мы планируем выпускать обновления ежегодно, оставляя доступными предыдущие цифровые версии, а также с некоторой периодичностью публиковать печатные версии книги. Часто выпускать обновления не всегда правильно с точки зрения перспектив обучения, но ежегодные обновления и поддержка предыдущих версий кажется нам хорошим компромиссом.

Мартен ван Стин
Эндрю С. Таненбаум

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com на странице с описанием соответствующей книги.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Springer очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Глава 1

Введение[1]

Темпы изменения компьютерных систем были, есть и продолжают оставаться ошеломляющими. С 1945 года, когда началась современная компьютерная эра, примерно до 1985 года компьютеры были большими и дорогими. Более того, из-за ограниченных возможностей их соединения они работали независимо друг от друга.

Однако начиная с середины 1980-х годов два технологических достижения начали изменять ситуацию. Первым достижением была разработка мощных микропроцессоров. Первоначально это были 8-битные машины, но вскоре общедоступными стали 16-, 32- и 64-битные процессоры. С многоядерными процессорами мы сейчас переходим к решению проблемы адаптации и разработки программ для использования параллелизма. Цена компьютеров нынешнего поколения в тысячу раз меньше, чем цена мейнфреймов 30- или 40-летней давности, обладающих такой же вычислительной мощностью.

Вторым достижением было изобретение и разработка высокоскоростных компьютерных сетей. **Локальные сети** (Local-area networks, LAN) позволяют тысячам компьютеров в пределах одного здания быть связанными таким образом, что небольшие объемы информации могут быть переданы в течение нескольких микросекунд. Большие объемы данных могут перемещаться между машинами со скоростью миллиардов *бит в секунду* (бит/с, bps). **Глобальные сети** (Wide-area networks, WAN) позволяют сотням миллионов компьютеров по всему миру быть подключенными друг к другу и обмениваться информацией на скоростях от десятков тысяч до сотен миллионов бит/с.

Параллельно с развитием все более мощных и объединенных сетью машин мы стали свидетелями миниатюризации компьютерных систем, и, возможно, смартфоны – самый впечатляющий результат этого процесса. Снабженные датчиками, большой памятью и мощным процессором, эти устройства стали полноценными компьютерами, и они, конечно, обладают и сетевыми возможностями. Находят свой путь к рынку и так называемые **подключаемые компьютеры** (plug computers). Эти небольшие устройства, часто размером с адаптер питания, просто включаются в розетку и обладают производительностью, сравнимой с настольными компьютерами.

В результате появления этих технологий теперь несложно собрать компьютерную систему, состоящую из большого числа сетевых компьютеров, будь они

---

[1] Версия данной главы была опубликована как «Краткое введение в распределенные системы». Computing, vol. 98 (10): 967–1009, 2016.

большие или маленькие. Компьютеры такой системы, как правило, находятся на расстоянии друг от друга, и потому обычно говорят, что они образуют **распределенную систему** (distributed system). Распределенная система может состоять как из нескольких, так и из миллионов компьютеров. Соединяющая их сеть может быть проводной, беспроводной или сочетанием того и другого. Кроме того, распределенные системы часто очень динамичны, в том смысле, что компьютеры могут присоединяться и отсоединяться, так что топология базовой сети и ее производительность почти непрерывно меняются.

В этой главе мы начнем исследовать распределенные системы и цели их построения, а затем обсудим некоторые известные типы систем.

1.1. Что такое распределенная система?

В литературе приводятся различные определения распределенных систем. Ни одно из них не является удовлетворительным и все они плохо согласуется друг с другом. Для наших целей достаточно дать такое весьма широкое определение:

> *Распределенная система представляет собой совокупность автономных вычислительных элементов, которая выглядит для пользователей единой цельной системой.*

В этом определении отмечается две характерные особенности распределенных систем. Во-первых, распределенная система представляет собой совокупность вычислительных элементов, каждый из которых в состоянии работать независимо от других. Вычислительный элемент, который мы обычно будем называть узлом, может быть аппаратным устройством или программным процессом. Вторая особенность заключается в том, что пользователи (будь то люди или приложения) считают, что они имеют дело с единой системой. Это означает, что в той или иной степени автономные узлы должны сотрудничать. Существо такого сотрудничества лежит в основе разработки распределенных систем. Обратите внимание, что мы не делаем никаких предположений относительно типа узлов. В принципе, даже в пределах одной системы они могут варьироваться от высокопроизводительных мейнфреймов до небольших устройств в сенсорных сетях. Кроме того, мы не делаем никаких предположений относительно того, как узлы взаимосвязаны.

Характеристика 1: совокупность автономных вычислительных элементов

Современные распределенные системы могут и часто состоят из всех видов узлов, начиная от очень больших высокопроизводительных компьютеров и заканчивая маленькими подключаемыми компьютерами или даже еще меньшими устройствами. Основополагающий принцип заключается в том,

что узлы могут действовать независимо друг от друга, хотя очевидно, что если они игнорируют друг друга, то нет смысла помещать их в одну и ту же распределенную систему. На практике узлы запрограммированы для достижения общих целей, которые реализуются путем обмена сообщениями между ними. Узел реагирует на входящие сообщения, которые затем обрабатываются и, в свою очередь, ведут к общению посредством дальнейшей передачи сообщений.

Важным является то, что временна́я последовательность взаимодействия независимых узлов определяется каждым из узлов самостоятельно. Другими словами, не всегда можно предположить, что есть что-то вроде **глобальных часов** (global clock). Этот недостаток приводит к необходимости решения фундаментальных вопросов, касающихся синхронизации и координации внутри распределенной системы, которые мы обсудим подробно в главе 6. Тот факт, что мы имеем дело с *совокупностью* узлов, подразумевает, что нам может понадобиться организация и управление этой совокупностью. Другими словами, нам может понадобиться определять, какие узлы могут принадлежать системе, а какие не могут, а также обеспечивать всех членов системы списком узлов, с которыми они могут общаться напрямую.

Управление **членством в группе** (group membership) может быть чрезвычайно сложным, хотя бы по причине контроля доступа. Проще говоря, мы различаем открытые и закрытые группы. В **открытой группе** любой узел может присоединиться к распределенной системе, что означает, что она может отправлять сообщения любому другом узлу в системе. В **закрытой группе** общаться друг с другом могут только члены этой группы. Необходим также отдельный механизм, позволяющий узлу присоединиться к группе или покинуть ее.

Нетрудно видеть, что контроль доступа может быть сложным. Во-первых, необходим механизм аутентификации узла, и, как мы увидим в главе 9, если он неправильно разработан, управление аутентификацией может легко стать узким местом при масштабировании. Во-вторых, каждый узел должен, в принципе, проверять, действительно ли он общается с другим членом группы, а не, например, со злоумышленником, стремящимся нарушить порядок. Наконец, учитывая, что член группы может легко общаться с участниками, не являющимися членами его группы, то при общении внутри распределенной системы должна обеспечиваться конфиденциальность, иначе можно столкнуться с проблемами доверия.

Что касается организации совокупности узлов, практика показывает, что распределенная система часто организуется в виде **оверлейной сети** (overlay network) [Tarkoma, 2010]. В этом случае узел обычно представляет собой программный процесс, снабженный списком других процессов, которым он может напрямую отправлять сообщения. Может также оказаться, что сначала необходимо найти соседа. Передача сообщений осуществляется через TCP/IP- или UDP-каналы, но, как мы увидим в главе 4, могут быть доступны также средства более высокого уровня. Существует два типа оверлейных сетей.

1) **структурированные:** в этом случае каждый узел имеет четко определенный набор соседей, с которыми он может общаться. Например, узлы организованы в виде дерева или логического кольца;

2) **неструктурированные:** в этом случае каждый узел имеет несколько ссылок на случайно выбранные другие узлы.

В любом случае, оверлейная сеть должна быть в принципе всегда **связна**, то есть между любыми двумя узлами всегда должен существовать канал связи, позволяющий этим узлам отправлять сообщения друг другу. Хорошо известный класс оверлейных сетей формируется **одноранговыми** (peer-to-peer, P2P) сетями. Примеры оверлейных сетей будут подробно обсуждаться в главе 2 и последующих главах. Важно понимать, что организация узлов требует особых усилий и что это иногда одна из самых сложных задач управления распределенными системами.

Характеристика 2: единая цельная система

Как уже упоминалось, распределенная система должна выглядеть как единая согласованная система. В некоторых случаях исследователи даже зашли так далеко, что говорят, что должно быть единое системное представление, то есть конечные пользователи не должны даже замечать, что они имеют дело с процессами, данными и управлением, рассредоточенными по всей компьютерной сети. Достижение единого представления часто требует слишком многого, и по этой причине мы выбрали более слабое определение распределенной системы как *кажущейся* пользователям цельной. Грубо говоря, распределенная система является цельной, если она ведет себя в соответствии с ожиданиями ее пользователей. Более конкретно, в единой связанной системе совокупность узлов в целом работает одинаково, независимо от того, где, когда и как происходит взаимодействие между пользователем и системой.

Часто достаточно сложно представлять распределенную систему как единую цельную систему. Например, это требует, чтобы конечный пользователь не мог точно сказать, на каком компьютере в настоящее время выполняется процесс, или что, возможно, часть задачи была выделена в отдельный процесс, выполняющийся где-то еще. Место, где хранятся данные, также не должно иметь значения, как и то, что система может реплицировать данные для повышения производительности. Эта так называемая **прозрачность распределения** (distributing transparency), о которой мы поговорим подробнее в разделе 1.2, является важной целью проектирования распределенных систем. В некотором смысле это сродни подходу, принятому во многих Unix-подобных операционных системах, в которых доступ к ресурсам осуществляется через единый интерфейс файловой системы, фактически скрывающий различия между файлами, устройствами хранения и основной памятью.

Однако стремление к единой согласованной системе приводит к важному компромиссу. Поскольку мы не можем игнорировать тот факт, что распределенная система состоит из нескольких сетевых узлов, неизбежно, что в любое время какая-то часть системы может отказать. Это означает, что неожиданное поведение системы, при котором, например, некоторые приложения могут продолжать успешно выполняться, в то время как другие полностью останавливаются, – это реальность, с которой приходится иметь дело. Хотя частичные отказы присущи любой сложной системе, в распре-

деленных системах с ними особенно трудно справляться. Это позволило обладателю премии Тьюринга Лесли Лэмпорту (Leslie Lamport) описать распределенную систему как «[...] такую, в которой сбой какого-то компьютера, о существовании которого вы даже не подозревали, может сделать ваш компьютер бесполезным».

Промежуточное программное обеспечение и распределенные системы

Чтобы помочь разработке распределенных приложений, распределенные системы часто организованы так, чтобы иметь отдельный слой программного обеспечения, который логически размещен поверх соответствующих операционных систем компьютеров, являющихся частью системы. Эта организация показана на рис. 1.1 и известна как **промежуточное программное обеспечение** (middleware) [Bernstein, 1996].

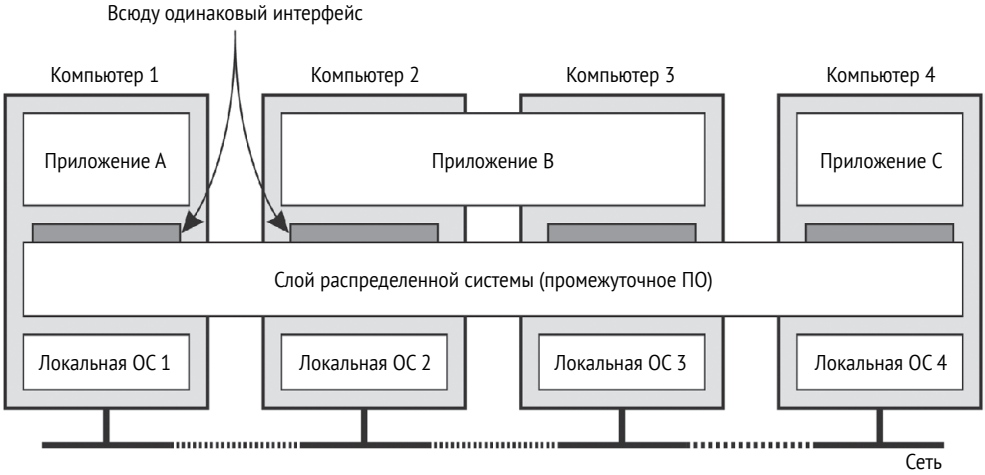

Рис. 1.1 ❖ Распределенная система, организованная на уровне промежуточного программного обеспечения, которая распространяется на несколько машин, предоставляя каждому приложению один и тот же интерфейс

На рис. 1.1 показаны четыре сетевых компьютера и три приложения, причем приложение B распределено между компьютерами 2 и 3. Каждому приложению предоставлен один и тот же интерфейс. Распределенная система предоставляет средства для связи компонентов одного распределенного приложения с каждым другим и позволяет различным приложениям общаться друг с другом. В то же время она скрывает от приложений, насколько это возможно и разумно, различия в оборудовании и операционных системах. В некотором смысле промежуточное ПО распределенной системы подобно операционной системе компьютера – это менеджер ресурсов, предоставляющий своим приложениям возможность эффективно разделять и размещать

эти ресурсы в сети. Помимо управления ресурсами, оно предлагает услуги, которые также можно найти в большинстве операционных систем, в том числе:

- средства для взаимодействия приложений;
- службы безопасности;
- средства учета;
- сокрытие отказов и восстановление после сбоев.

Основное отличие от эквивалентов операционной системы заключается в том, что услуги промежуточного программного обеспечения предлагаются в сетевом окружении. Обратите внимание, что большинство услуг полезно для многих приложений. В этом смысле промежуточное программное обеспечение может также рассматриваться как контейнер часто используемых компонентов и функций, которые не требуется реализовывать в каждом приложении отдельно. Чтобы проиллюстрировать это, давайте кратко рассмотрим несколько примеров типичных услуг промежуточного программного обеспечения.

- **Взаимодействие.** Общепринятой услугой взаимодействия является так называемый **удаленный вызов процедур** (Remote Procedure Call, RPC). Сервис RPC, к которому мы вернемся в главе 4, позволяет приложению вызывать функцию, которая применяется и выполняется на удаленном компьютере, как если бы она была доступна локально. Для этого разработчик должен просто указать заголовок функции, выраженный на специальном языке программирования, который позволяет подсистеме RPC генерировать затем необходимый код, который и выполняет удаленные вызовы.
- **Транзакции.** Многие приложения используют различные сервисы, распределенные по нескольким компьютерам. Промежуточное обеспечение обычно предлагает специальную поддержку для выполнения таких сервисов по правилу «все или ничего», то есть предоставляет атомарные **транзакции** (atomic transaction). В этом случае разработчик приложения должен только указать задействованные удаленные сервисы, и, следуя стандартизированному протоколу, промежуточное программное обеспечение гарантирует, что либо вызываются все службы, либо не вызывается ни одна.
- **Компановка услуг.** Становится все более распространенной разработка новых приложений путем комбинирования существующих программ. Это особенно касается многих веб-приложений, в частности тех, которые известны как **веб-сервисы** [Alonso et al., 2004]. Веб-ориентированное промежуточное программное обеспечение может помочь стандартизировать способ доступа к веб-сервисам и обеспечить средства для генерации их функций в определенном порядке. Простой пример того, как используется компоновка услуг – **машапы** (mashup): веб-страницы, которые объединяют и группируют данные из разных источников. Известными машапами являются машапы, основанные на картах Google, дополненных такой информацией, как планировщики поездок или прогноз погоды в режиме реального времени.
- **Надежность.** В качестве последнего примера: было проведено мно-

жество исследований по расширенным функциям для создания надежных распределенных приложений. Инструментарий Horus [van Renesse et al., 1994] позволяет разработчику создавать приложение как группу процессов, такую что любое сообщение, отправленное одним процессом, гарантированно будет получено всеми или никакими другими процессами. Как оказалось, такие гарантии могут значительно упростить разработку распределенных приложений и, как правило, реализуются как часть промежуточного программного обеспечения.

Примечание 1.1 (историческая справка: термин «промежуточное программное обеспечение»)

Хотя термин «промежуточное программное обеспечение» стал популярным в середине 1990-х годов, впервые он, скорее всего, упоминается в сборнике докладов конференции НАТО по разработке программного обеспечения под редакцией Питера Наура (Peter Naur) и Брайана Рэнделла (Brian Randell) в октябре 1968 года [Peter Naur and Brian Randell, 1968]. В этих докладах промежуточное ПО действительно было размещено как раз между приложениями и подпрограммами обслуживания (эквивалент операционных систем).

1.2. Цели проектирования

Только то, что существует возможность создания распределенных систем, вовсе не обязательно означает, что это хорошая идея. В этом разделе мы обсудим четыре важные цели, которые должны быть достигнуты, чтобы сделать построение распределенной системы стоящим усилий. Распределенная система должна сделать ресурсы легкодоступными; она должна скрывать тот факт, что ресурсы распределены по сети; она должна быть открытой; и она должна быть масштабируемой.

Поддержка совместного использования ресурсов

Важной целью распределенной системы является облегчение пользователям (и приложениям) доступа и совместного использования удаленных ресурсов. Ресурсы могут быть практически чем угодно, но типичными примерами являются периферийные устройства, хранилища, данные, файлы, сервисы и сети, и это лишь небольшая часть. Есть много причин для желания разделять ресурсы. Одна очевидная причина – это экономика. Например, дешевле иметь одно высококлассное надежное общее хранилище, чем покупать и поддерживать хранилище для каждого пользователя в отдельности.

Соединение пользователей и ресурсов также облегчает сотрудничество и обмен информацией, о чем свидетельствует успех интернета с его простыми протоколами для обмена файлами, почтой, документами, аудио и видео. Подключение к интернету позволило географически широко рассредоточен-

ным группам людей работать совместно с помощью всех видов **программного обеспечения для совместной работы** (groupware), то есть программ для совместного редактирования, телеконференций и т. д., как показали многонациональные компании-разработчики программного обеспечения, которые передали многие работы по разработке кода в Азию.

Однако совместное использование ресурсов в распределенных системах, возможно, лучше всего иллюстрируется успешным обменом файлами между одноранговыми сетями, такими как BitTorrent. Эти распределенные системы упрощают обмен файлами между пользователями интернета. Одноранговые сети часто связаны с распределением медиафайлов, таких как аудио и видео. В других случаях технология используется для распространения больших объемов данных, как в случае обновлений программного обеспечения, услуг резервного копирования и синхронизации данных на нескольких серверах.

Примечание 1.2 (дополнительная информация: общий доступ к папкам по всему миру)

Чтобы проиллюстрировать, где мы находимся в настоящее время, когда дело доходит до бесшовной интеграции средств совместного использования ресурсов в сетевой среде, достаточно сказать, что сейчас развернуты веб-службы, которые позволяют группе пользователей размещать файлы в специальной общей папке, которая поддерживается третьей стороной где-то в интернете. При использовании специального программного обеспечения, общая папка почти не отличается от других папок на компьютере пользователя. По сути, эти службы заменяют использование общего каталога в локальной распределенной файловой системе, делая данные доступными для пользователей независимо от организации, которой они принадлежат, и от того, где они находятся. Эта услуга предлагается для разных операционных систем. Где именно хранятся данные, полностью скрыто от конечного пользователя.

Создание прозрачных распределений

Важной целью распределенной системы является скрытие того факта, что ее процессы и ресурсы физически распределены по нескольким компьютерам и, возможно, разделены большими расстояниями. Другими словами, она пытается сделать распределение процессов и ресурсов **прозрачным**, то есть невидимым для конечных пользователей и приложений.

Типы прозрачности распределений

Концепция прозрачности, то есть незаметности внутренней структуры для пользователя, может быть применена к нескольким аспектам распределенной системы, из которых наиболее важные перечислены на рис. 1.2. Мы используем термин «*объект*», имея в виду или процесс, или ресурс.

Прозрачность	Описание

Доступ	Скрыть различия в представлении данных и то, как получен доступ к объекту
Местоположение	Скрыть, где находится объект
Перемещение	Скрыть, что объект может быть перемещен в другое место, когда используется
Миграция	Скрыть, что объект может переместиться в другое место
Репликация	Скрыть, что объект реплицируется
Конкуренция	Скрыть, что объект может разделяться несколькими независимыми пользователями
Отказ	Скрыть сбой и восстановление объекта

Рис. 1.2 ❖ Различные формы прозрачности в распределенной системе (см. ISO[1995]). Объект может быть ресурсом или процессом

Прозрачность доступа (access transparency) связана с сокрытием различий в представлении данных и способа доступа к объекту. На базовом уровне мы хотим скрыть различия в архитектуре машин, но, что более важно, мы достигаем соглашения о том, как данные должны быть представлены различными машинами и операционными системами. Например, распределенная система может иметь в составе компьютеры, на которых работают разные операционные системы со своими соглашениями об именовании файлов. Различия в соглашениях об именовании, различия в файловых операциях или различия в том, как организована низкоуровневая коммуникация с другими процессами, являются примерами проблем доступа, которые желательно скрывать от пользователей и приложений.

Важной группой типов прозрачности являются местоположения процесса или ресурса. **Прозрачность местоположения** (location transparency) означает, что пользователи не могут сказать, где физически объект находится в системе. Именование играет важную роль в достижении прозрачности местоположения. В частности, прозрачность местоположения часто может быть достигнута путем назначения ресурсам только логических имен, то есть имен, в которых не закодировано местоположение ресурса. Примером такого имени является **унифицированный указатель ресурса** (uniform resource locator, URL) http://www.prenhall.com/index.html, который не дает никакой информации о фактическом расположении основного веб-сервера Prentice Hall. URL также не дает подсказки о том, был ли файл index.html всегда в его текущем местоположении или недавно был перемещен туда. Например, весь сайт, возможно, был перемещен из одного центра обработки данных в другой, но пользователи не должны этого замечать. **Прозрачность перемещения** (relocation transparency) становится все более важной в контексте облачных вычислений, к чему мы вернемся позже в этой главе.

Если прозрачность перемещения относится к перемещению самой распределенной системой, то **прозрачность миграции** (migration transparency) обеспечивается распределенной системой, когда она поддерживает мобильность процессов и ресурсов, инициируемых пользователями, без влияния на текущую связь и операции. Типичным примером этого является связь между мобильными телефонами: независимо от того, перемещаются ли два

Конец ознакомительного фрагмента.
Приобрести книгу можно
в интернет-магазине
«Электронный универс»
e-Univers.ru