Программная инженерия. Лекция 4 Архитектура и архитектурные образцы

Михаил Пожидаев

6 марта 2024 г.

Блоки и образцы

Структурные компоненты архитектуры

Блоки

Подсистемы, зависимости, интерфейсы, кооперации, узлы и активные классы.

Образцы

Брокер, многоуровневая архитектура, клиент-сервер, одноранговая архитектура, монолитная и микросервисная архитектуры.

Составные части

Из чего состоит архитектура?

Архитектура программы включает в себя данные:

- об организации программной системы;
- о структурных элементах, входящих в систему, и их интерфейсах, а также их поведении, которое определяется кооперациями, в которых участвуют элементы;
- о составе структурных элементов и элементов поведения наиболее крупных подсистем;
- о стиле архитектуры, принятом в данной организации, — элементах и их интерфейсах, их кооперации и композиции.

Факторы

Что влияет на архитектуру?

- 1. Системное программное обеспечение, которым необходимо пользоваться для построения системы.
- 2. Программное обеспечение среднего уровня, которым необходимо пользоваться. Например, Apache Kafka в роли брокера.
- 3. Каркас для создания пользовательского графического интерфейса (JavaFX, QT и пр.).
- 4. Унаследованные системы, которые войдут в новую систему.
- 5. Стандарты и правила компании, которые необходимо соблюдать.
- 6. Общие нефункциональные требования (т. е. не привязанные к конкретному варианту использования).
- 7. Требования к распределению, определяющие, как система распределяется по компьютерам.

Построение архитектуры

Как разрабатывается архитектура?

- 1. Архитектура разрабатывается в ходе итераций, в основном *в фазе проектирования*.
- 2. Варианты использования управляют разработкой архитектуры, а архитектура направляет реализуемые варианты использования.
- 3. На каждой итерации мы выбираем и реализуем набор вариантов использования, чтобы подтвердить, а при необходимости и улучшить архитектуру.

Варианты использования помогают по мере продвижения в деле разработки постепенно улучшать архитектуру. Это одно из преимуществ использования методики разработки, управляемой вариантами использования.

Сервер

Центральный узел архитектуры

Сервер — компьютер или виртуальный хост для обработки запросов клиентских систем. Обладает следующими характеристиками:

- предполагает повышенные вычислительные ресурсы,
 т. к. от их количества зависит множество клиентских систем;
- не предполагает работу пользователей, кроме администраторов;
- представлен в единственном варианте или в виде кластера из нескольких соединённых систем.

Клиент

Множество узлов для работы пользователей

Клиент — компьютер или приложение для работы пользователей. Обладает следующими характеристиками:

- функционально содержит реализацию только интерфейса пользователя, перекладывая основную работу на сервер, с которым поддерживает постоянное подключение;
- множество клиентов определяется вычислительными мощностями сервера и пропускной способностью канала подключения к нему;
- может иметь ограниченные вычислительные ресурсы, но становится несамостоятельным при утрате подключения к серверу.

Протокол

Соглашение для связи клиента и сервера

Критический компонент

Протокол выполняет критическую функцию поддержания согласованного взаимодействия всех клиентов и сервера. Нарушения его поддержки немедленно приводят к разрушению всей архитектуры.

- 1. Требуется относиться предельно тщательно к проектированию протоколов, максимально отдаляя момент его обновления.
- 2. Протокол не должен быть перегруженным, поскольку это приводит к исчерпанию ресурсов каналов связи.
- Фиксированный протокол позволяет проводить обновления клиентского и серверного ПО независимо друг от друга.

Микросервисная архитектура

Контроллер, брокер и не только

Блоки рассматриваемой архитектуры:

- 1. Контроллер: центральный узел, получающий команды от внешних клиентских соединений.
- 2. Брокер сообщений: диспетчер обмена сообщениями между контроллером и остальными контейнерами.
- 3. Микросервисы: динамически изменяемое множество одноранговых контейнеров разных типов для выполнения работ, необходимых для обслуживания запросов.

Микросервис

Основные черты

- 1. Слабая связь с остальными микросервисами.
- 2. Изолирование фактической технологии внутри микросервиса (каждый микросервис разрабатывается своей командой).
- 3. Коммуникация путём сетевого взаимодействия (предпочтительно через HTTP).
- 4. Оформление в виде контейнера.
- 5. Могут находиться территориально распределённо.

Проектирование

Как проектировать микросервисную архитектуру

- 1. Формализация ясного функционального назначения микросервисов.
- Максимальное устранение дублирования функциональности между микросервисами.
- Фиксация интерфейсов и связей между микросервисами.
- 4. Фиксация конфигурации для развёртывания и требования к ней.

IO-bound

Варианты реализации конкурентной системы

- 1. Синхронный ввод/вывод: программа разбивается на потоки, каждый из которых ждёт завершения операций ввода/вывода.
- 2. Асинхронный ввод/вывод: всё выполнение происходит в одном потоке без ожидания завершения операций ввода/вывода, которые выполняются в фоновых потоках.

Событийная модель

- 1. Всё приложение разбивается на множество событий как можно меньшего размера.
- 2. События выполняются в одном общем потоке под управлением главного цикла (event loop).
- 3. Операция ввода-вывода обычно разбивается на событие, её запускающее, и событие, обрабатывающее её результат.

Преимущества:

- можно не заботиться о согласовании потоков;
- возрастает общая эффективность исполнения.

Недостатки

Минусы асинхронного программирования

- 1. Нельзя смешивать блокирующие и неблокирующие вызовы.
- 2. Плохо подходит для задач с большой вычислительной нагрузкой (не IO-bound).

Архитектура

Когда применять асинхронный ввод/вывод

- 1. Высокая конкурентность запросов, при которой традиционная архитектура с использованием мьютексов и семафоров может оказаться сложной и запутанной.
- 2. Доминирование операций ввода/вывода над вычислениями.
- 3. Отсутствие необходимости смешивать код с традиционном кодом на основе блокирующего ввода/вывода.

Распределённые системы

Ключевые особенности обобщённого понятия

Под распределёнными системами понимается класс вычислительных систем, которые используют для хранения и обработки информации динамически изменяемое множество узлов.

Не одноранговая архитектура!

Распределённые системы имеют узел-контроллер, которого нет в одноранговых системах.

Не микросервисная архитектура!

Вычислительная модель в распределённых системах обычно однородная и больше ориентирована на однотипные операции сортировки и поиска.

Плюсы

Достоинства распределённых систем

- 1. Способность обрабатывать огромные массивы слабоструктурированных данных.
- 2. Низкая чувствительность к отказу отдельного вычислительного узла, кроме контролирующих.
- 3. Простота решения задач сортировки и поиска информации.
- 4. Возможность географического распределения данных.

Минусы

Недостатки распределённых систем

- 1. Наличие критического узла-контроллера., нарушение работы которого приводит к утрате всей системы.
- 2. Высокая потребность в вычислительных мощностях из-за многократного дублирования хранимой информации.
- 3. Ограниченность вычислительных задач, выполняемых с хранимыми данными.

Одноранговые системы

Основные характеристики и особенности

Общая черта

Все узлы системы играют в ней одинаковую роль, и отсутствует управляющий или координирующий узел. Обычно разрабатываются не от хорошей жизни.

- 1. Обладают повышенной отказоустойчивостью и защищённостью от недобросовестного администрирования.
- 2. Нередко подразумевают многократное дублирование одного и того же функционала (особенно систем хранения).
- 3. Требуют проработки глубоких механизмов согласованной работы.

Блокчейн

Криптовалюты, смарт-контракты и NFT

- 1. Все узлы хранят полную историю операций (цепочку блоков).
- 2. Каждый блок содержит хэш-сумму предыдущего блока.
- 3. Попытка добавления нового блока требует одобрения определённого количества других участников.
- 4. Требуется наличие механизма защиты от добавления несогласованных блоков.

Биткойн

Блоки с «денежными» транзакциями

Блок включает:

- информацию о переводе средств;
- хэш-сумму предыдущего блока;
- специальную «добавку» для корректировки хэш-суммы нового блока.

Майнинг

Вычисление «добавки» требует ресурсов и времени. Введено для ограничения скорости порождения блоков и предполагает вознаграждение за проделанную работу.

DHT

Распределённая хеш-таблица

DHT является примером одноранговой системы со следующими ключевыми характеристиками:

- 1. Практическое отсутствие ограничений масштабирования.
- 2. Выдающаяся отказоустойчивость, практически недостижимая в клиент-серверной и микросервисной архитектурах.

Обычно применяется для отказоустойчивого хранения информации (распределённые СУБД, пиринговые системы и пр.).

CAD

Content addressable storage

Контент вместо адреса

Для построения адреса хранимого блока можно использовать содержимое блока после обработки хеш-функцией.

Свойства:

- неизменяемость данных;
- изменение блока подразумевает получение нового адреса (хотя старый блок можно не удалять);
- ▶ гармоничное сочетание с DHT.

Архитектура

Когда применять DHT

- 1. Требуется особая отказоустойчивость без точек отказа.
- 2. Существует возможность формирования пространства ключей.
- 3. Система ориентирована на хранение информации.
- 4. Доступно большое количество вычислительных ресурсов. Их количество важнее, чем уровень ожидаемой доступности.

Теорема САР

Эмпирическое наблюдение

Любая распределённая система будет удовлетворять только двум любым требованиям из списка:

- согласованность (consistency);
- доступность (availability);
- устойчивость к разделению (partitions tolerance).

Спасибо за внимание!

Всё о курсе: https://marigostra.ru/materials/engineering.html

E-mail: msp@luwrain.org

Канал в Телеграм: https://t.me/MarigostraRu