Quantbox

III Всероссийская конференция по алгоритмической торговле

Немного о нас

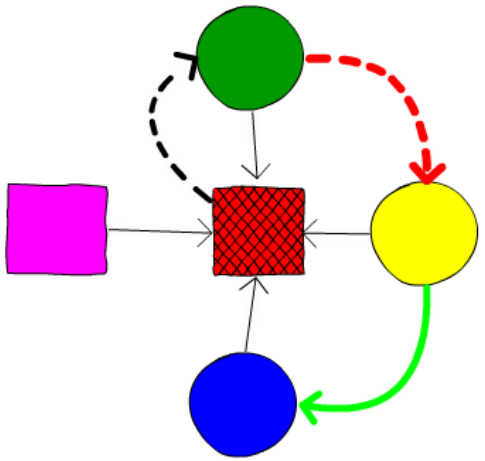
- Quantbox LLC – 2015
- Сфера деятельности:
  - Разработка торговой инфраструктуры (подключение к биржам, протоколы трансляции биржевой информации, протоколы исполнения биржевых заявок)
  - Алгоритмы для брокеров (TWAP, VWAP, POV, Iceberg, etc)
  - Other stuff: market makers (options, equities), hedgers, rollovers, spreader
  - HFT алгоритмы

Технологии

- Linux (kernel >= 3.9). Основной дистрибутив Ubuntu 14.04
- Языки разработки: C (gcc), Python
- Юнит-тестирование: Google Test/Mock
- Инструментарий непрерывной интеграции: Jenkins
- Система сборки: cmake
- Система контроля версий: git

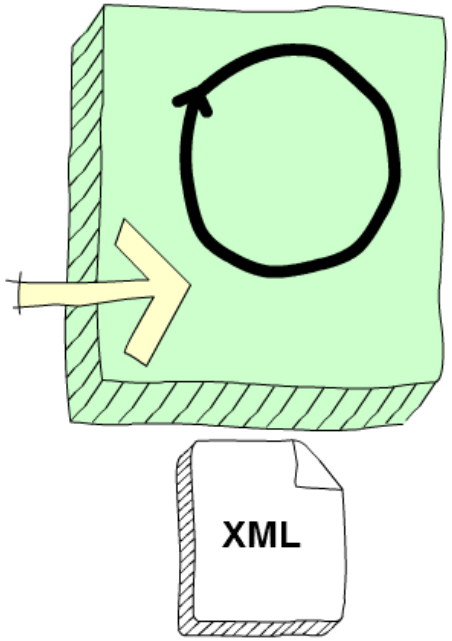
Требования к системе

- Проблема: много независимых сервисов, много общих подсистем, много разных потоков данных
- Корпоративные требования:
  - Отказоустойчивость
  - Слабые взаимосвязи между сервисами
  - Высокая степень переиспользования функционала
  - Работа в режиме 24x7
  - Масштабируемость
  - Простота поддержки



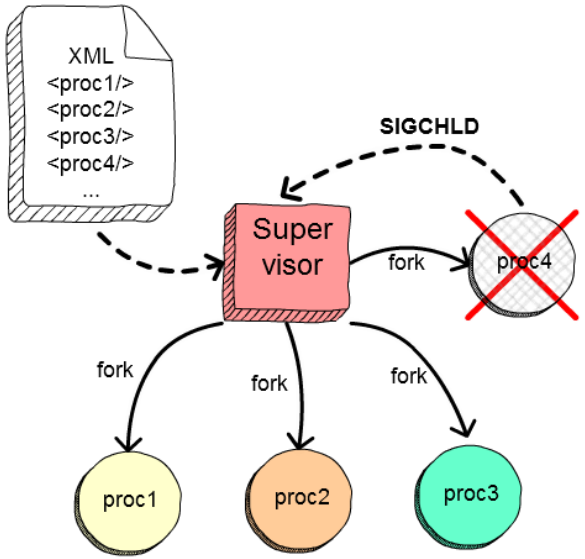
Требования к микросервису

- Один сервис – один процесс. Личное адресное пространство
- Однопоточное решение
- Нет блокирующих операций
- Каждый процесс имеет свою входящую очередь сообщений
- Простой управляющий интерфейс
- Независимая и простая конфигурация
- Каждый сервис выполняет свою микро-бизнес функцию



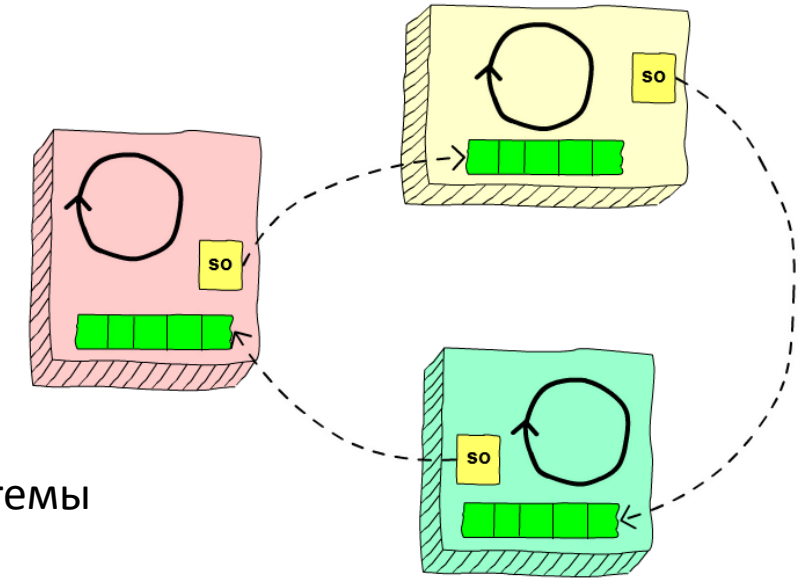
Управление микросервисами

- Проблема: много процессов (от двух-трех до нескольких десятков)
- Supervisor – parent процесс для всех процессов системы
- Функции supervisor’a:
  - Порождает все остальные процессы по расписанию
  - Расписание – свой внутренний crontab
  - Выполняет функцию watchdog’a
  - Активирует/деактивирует процессы по расписанию
- Supervisor: одна точка входа – одна точка выхода
- Каждый процесс: стандартный набор управляющих сообщений (activate, deactivate, reconfigure, etc)



IPC

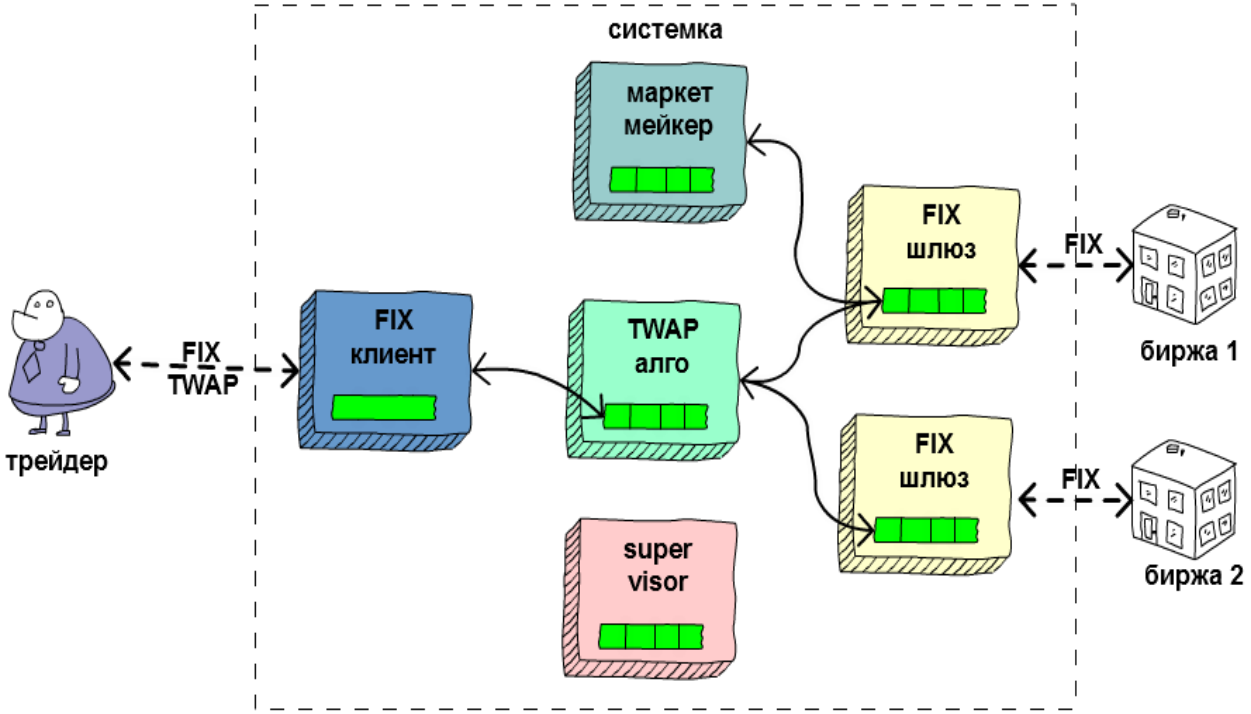
- IPC: POSIX Message Queue
- Каждому микросервису своя очередь сообщений
- P2P: нет единой шины или брокера сообщений
  - Нет брокера – нет единой точки отказа
  - Отсутствие overhead’a при передачи сообщений
- So: подгружаемая библиотека роутинга:
  - Собирается непосредственно перед запуском системы
  - Код библиотеки открыт для потребителя
  - Возможность менять настройки роутинга не перезагружая микросервисы
  - Одна библиотека для всех сервисов



Пример простой системы

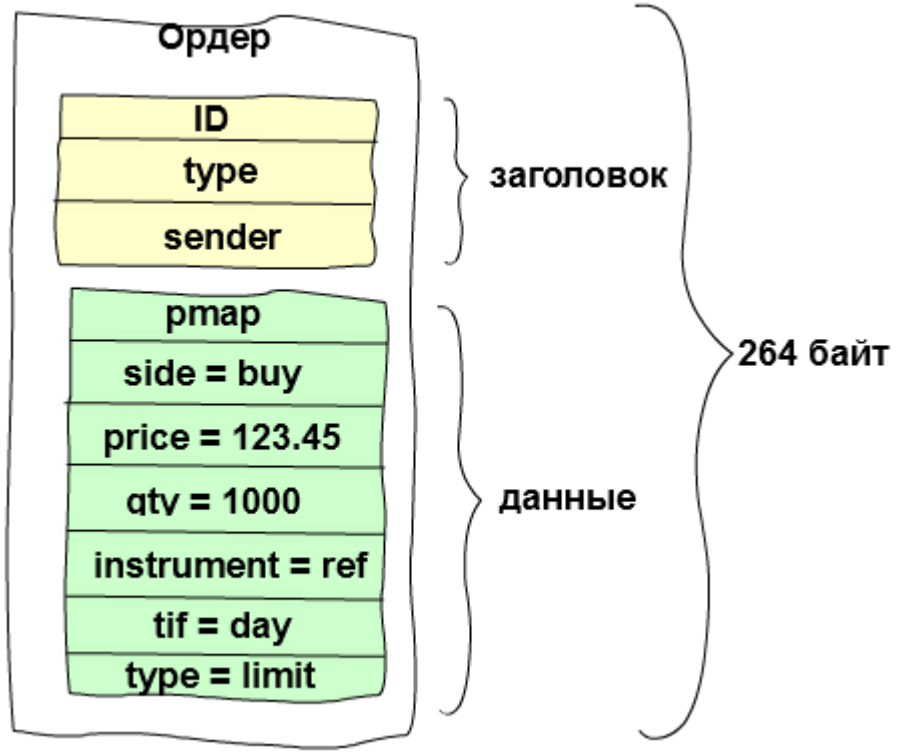
- POSIX mqueue performance
  - Ping-ping test
  - Intel Core i7-5820k

Size (byes)	Latency (usec)
32	**1.1**
64	1.2
128	1.2
256	1.2



Протокол обмена

- Различные типы передаваемых и получаемых сообщений (объектов)
  - Управляющие сообщения
  - Бизнес объекты (ордер, сделка и т.д.)
  - Объекты для хранения информации (описание инструмента, статистические данные, данные FIX сессии и т.д.)
- ***Все*** эти сообщения могут передаваться
- Бинарный протокол: сообщения (объекты) это С структуры с полями фиксированной длинны
- C, Python API для работы с объектами (getters/setters, send, check, clone, …)

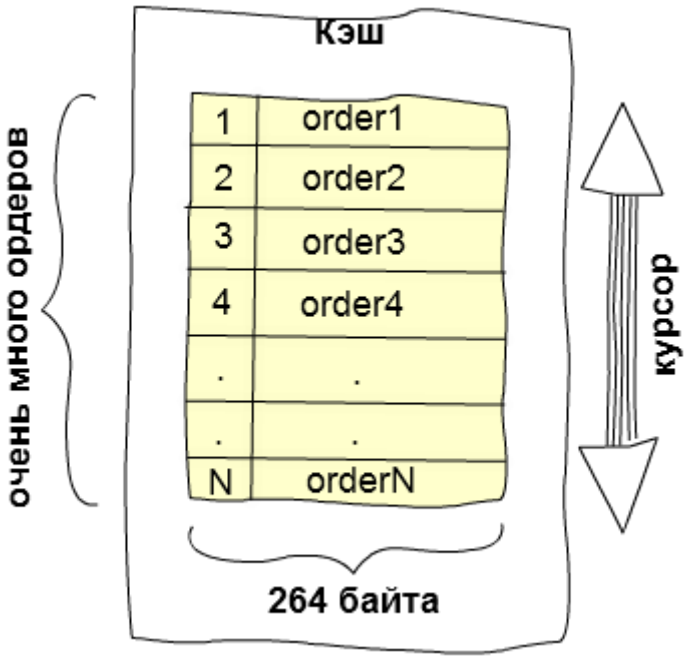


Кэширование данных 1

- Проблема: какие-то объекты надо где-то сохранять (ордера, сделки)
- SQL, NoSQL БД? – очень медленно
- Объекты могут храниться как в shared memory так и в memory mapped фалах
- Создание объекта – это выделение области в общей памяти
- Уникальный ID объекта (reference) – это смещение в общей памяти
- Передача объекта – это передача всего лишь reference на объект (zero-copy)
- Все микросервисы имеют прямой доступ к кэшу объектов

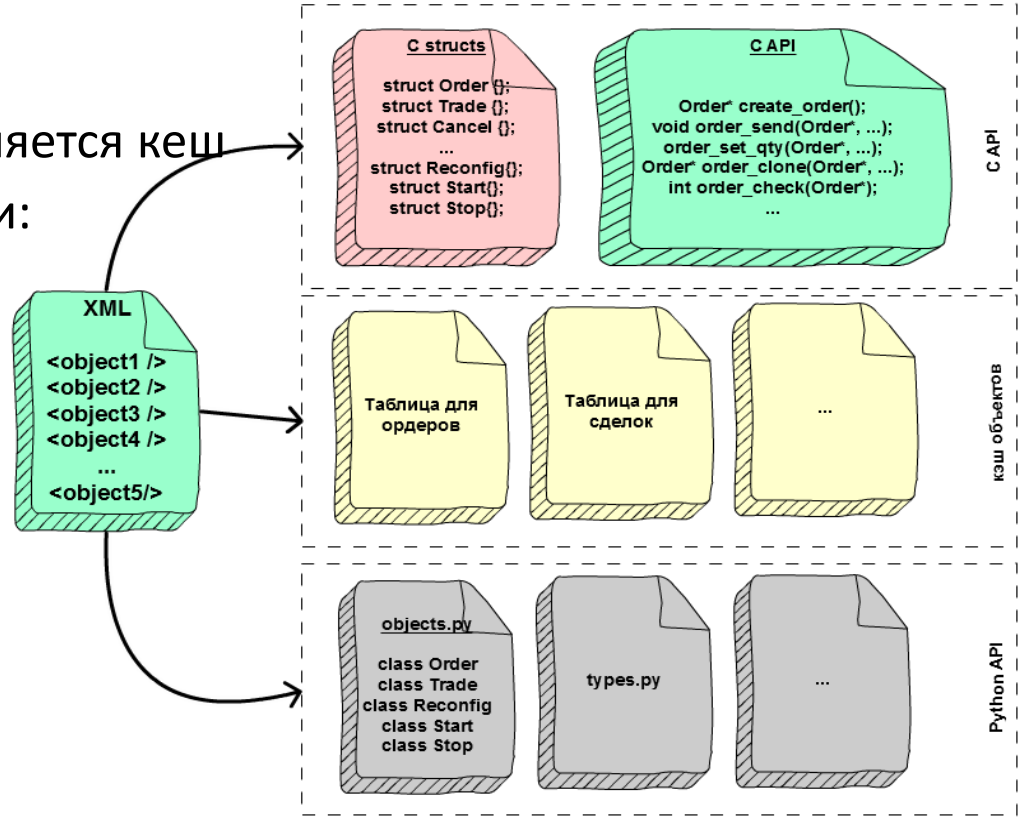
Кэширование данных 2

- Резюме
  - Кэш – key-value, где key это смещение
  - Хранилище это либо shared memory, либо memory mapped фалы
  - Создание объекта в кэше < 1 микросекунды
  - Курсор – есть возможность перебрать все объекты в кэше
  - Модификация объекта –модификация памяти
  - Минимум блокировок – только при создании объекта
  - Не все объекты нужно хранить в кэше – управляющие сообщения копируются при передаче



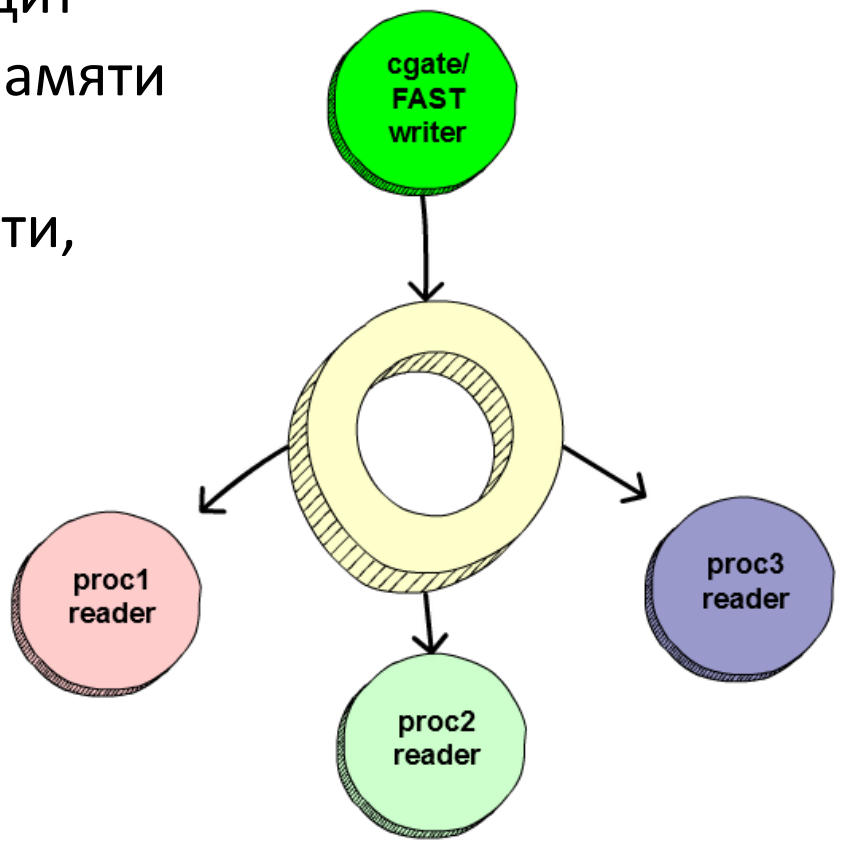
Генерация объектов

- Проблема
  - объектов много (десятки)
  - Объекты меняются – меняется API, меняется кеш
  - Объекты имеют разные характеристики: сохраняемые к кэше, не сохраняемые, сохраняемые в файлах, и т.д.
- Все объекты и их типы описаны в XML схеме
- Python + Jinja2 генератор – создает все необходимые файлы (API, кэш)

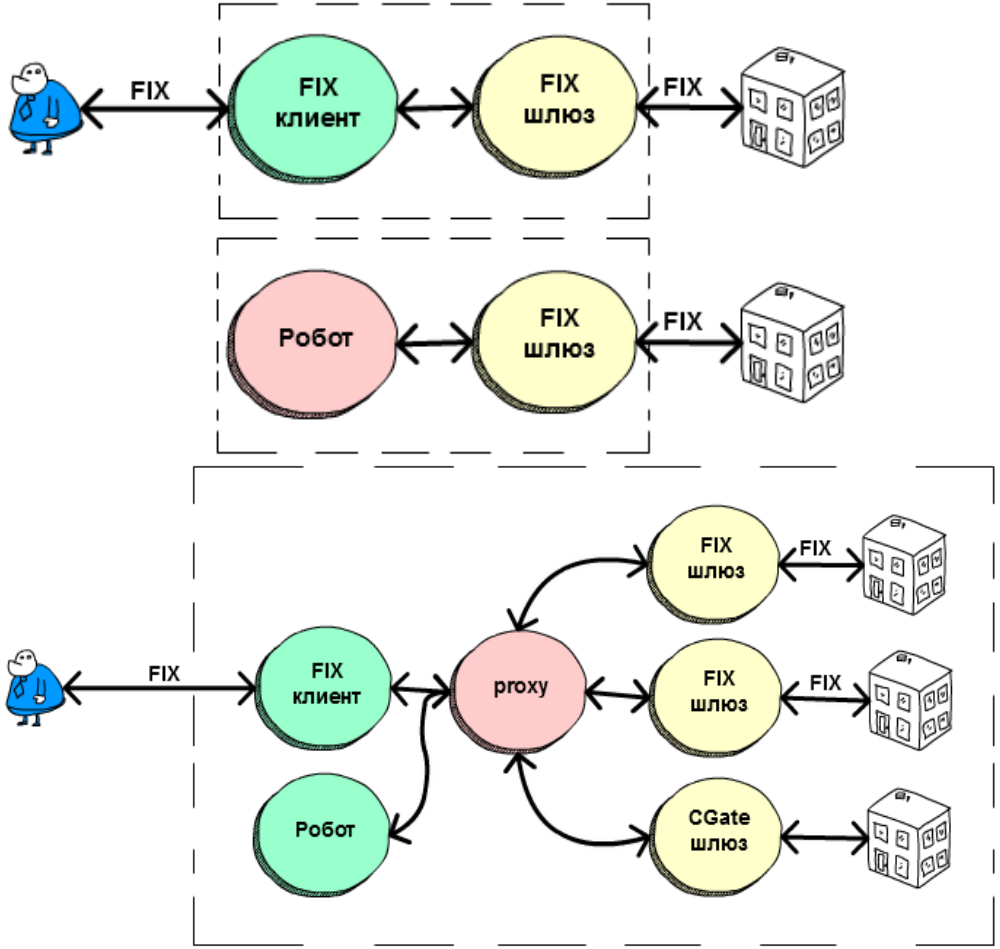


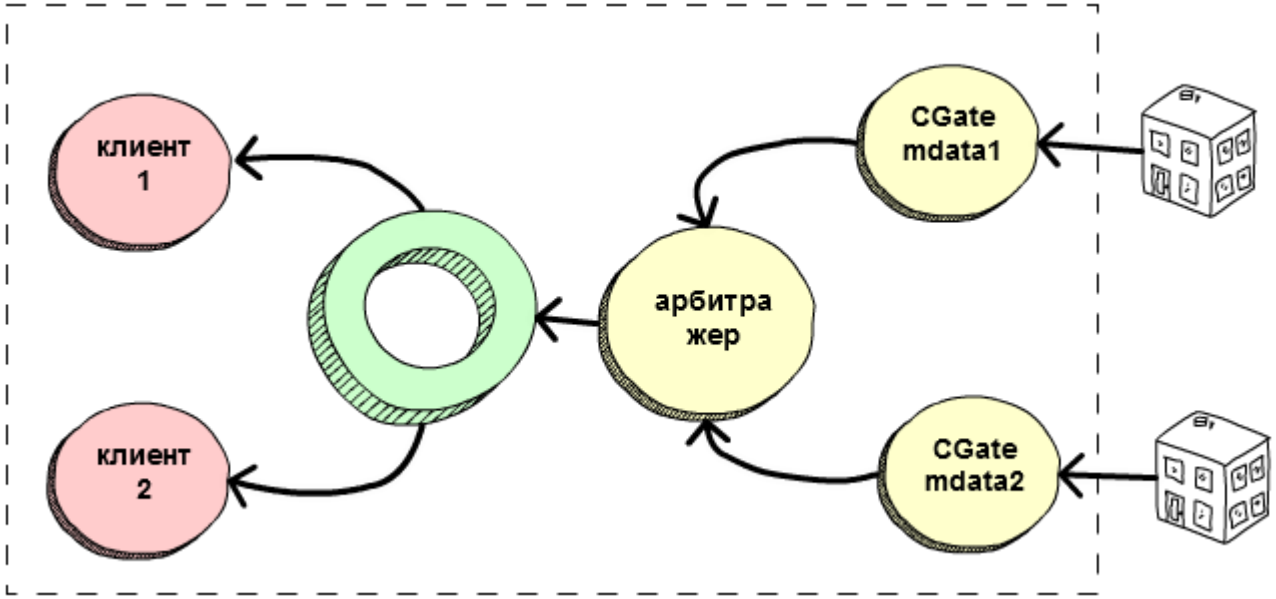
Маркет дата

- Проблема: POSIX Message Queue не подходит
- Кольцевой не блокирующийся RW буфер памяти
- Потребители имеют возможность как читать последние данные по необходимости, так и получать нотификации, что пришли новые данные
- Одновременный и равнозначный доступ для всех потребителей

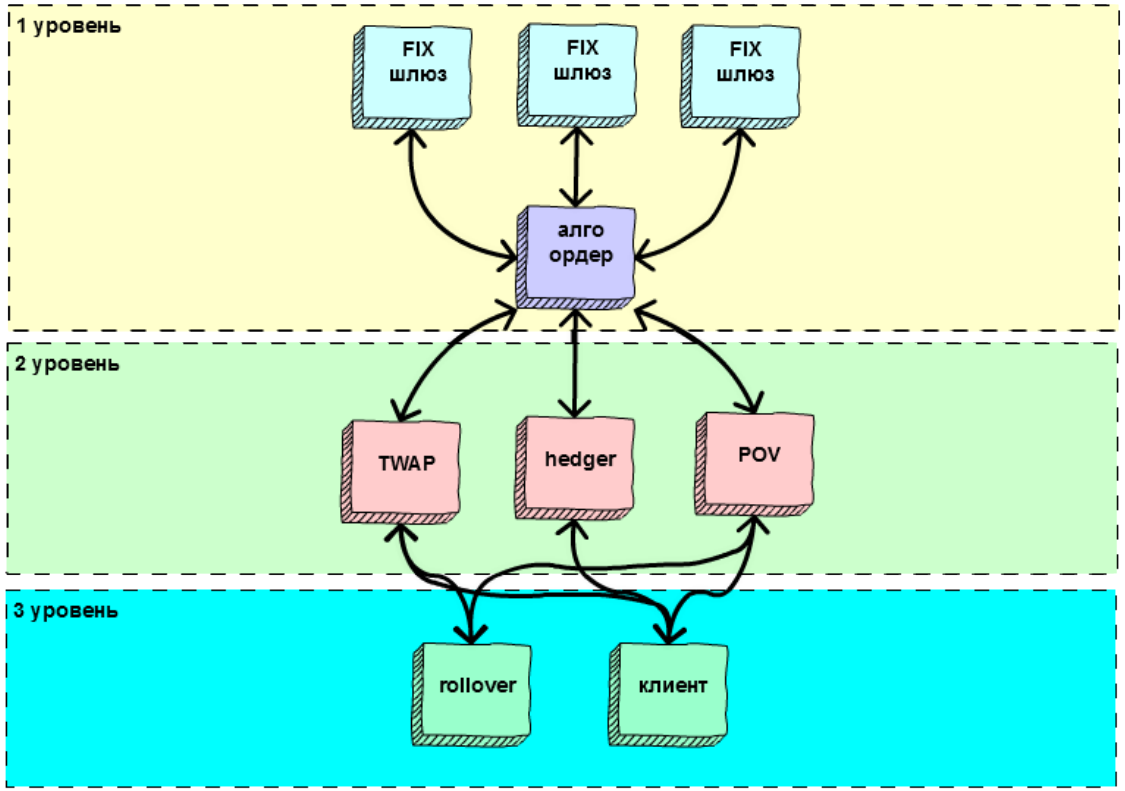


Примеры систем



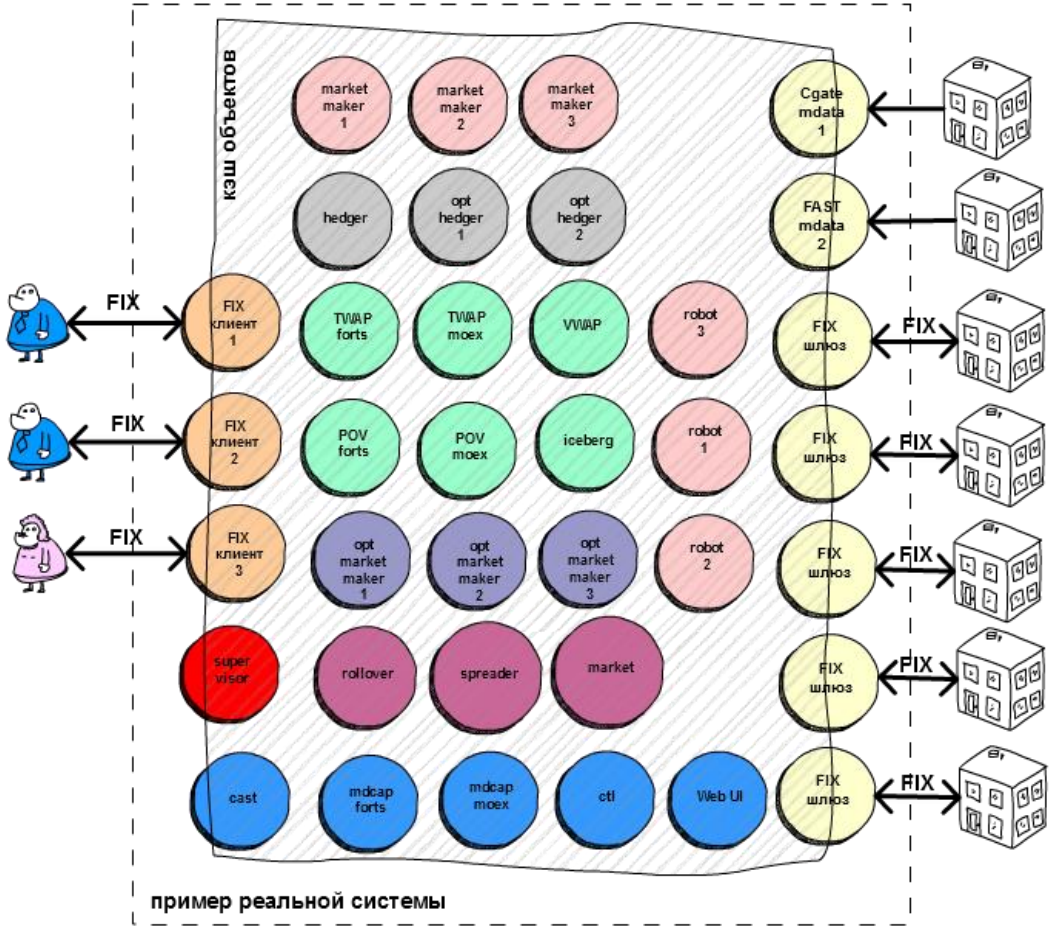


Комбинирование алгоритмов



Большие системы

- Десятки микросервисов
- У каждого своя бизнес задача
- Единое пространство данных
- Сбор биржевых данных
- Сервис для мониторинга системы
- Сервис интеграции с Zabbix & Nagios
- Возможность биндинга микросервиса к изолированному CPU
- 24x7



Будущее

- Развитие Python API
- Развитие Web-UI: web-service’ы на основе xroad
- Реализация **TWIME** (скоро)
- Бэктестинг
- Распределенная система – RDMA
- Развитие бизнес функционала
- Интеграция с Excel