Әдебиеттер тізімі:

1. Қазақстан Республикасының ауылдық аумақтарын дамытудың 2023 – 2027 жылдарға арналған тұжырымдамасын бекіту туралы» Қазақстан Республикасы Үкіметінің 2023 жылғы 28 наурыздағы № 270 қаулысына өзгерістер мен толықтырулар енгізу туралы Қазақстан Республикасы Үкіметінің 2023 жылғы 1 шілдедегі № 539 қаулысы
2. Қазақстанда АӨК-ті қаржыландыру бес жыл ішінде 65%-ға өсті ELDALA.KZ Bas AGRARYO SITY 2022 ж. https://eldala.kz/kz/specproekty/18338- finansirovanie-apk-v -kazahstane-za-pyat-let-vyroslo-na-65, 1-2 беттер.
3. Ауыл шаруашылығы өнімдерін өңдеу саласы жыл сайын өсім көрсетуде Қазақстан республикасы Премьер министрлігінің ресми ақпараттық ресурсы 08 қараша 2024, 1-3 беттер https://primeminister.kz/news/reviews/auyl-sharuashylygy-onimderin-ondeu-salasy-zhyl-sayyn-osim-korsetude-293244. 1
4. Қазақстанда агроөнеркәсіп кешені қалай дамып жатыр/ Егеменді Қазақстан 15 қараша, 2024 ж. 1-2 беттер https://egemen.kz/article/376379-qazaqstanda-agroonerkasip-kesheni-qalay-damyp-dgatyr

ЭКОНОМИКА

УДК 004

Надежность финансовых приложений: причины сбоев, последствия и современные методы повышения устойчивости

Senior Software Engineer АО "Kaspi Bank" Сычев Е. А.

м.э.н. преподаватель кафедры "Финансы и учет" Международный Таразский университет имени Ш.Муртазы Ким Т.А.

Резюме

В статье рассматриваются основные аспекты надежности приложений в финансовом секторе, включая такие ключевые понятия, как сбой, отказ и точка отказа. Подчеркивается, что даже кратковременные сбои способны вызывать серьезные финансовые, репутационные и операционные потери. В качестве примеров приводятся инциденты Knight Capital, TSB Bank и Citibank, понесшие многомиллионные убытки вследствие ошибок в программном обеспечении. Статья предлагает микросервисную архитектуру, комплексное тестирование, мониторинг и алерты как эффективные методы повышения отказоустойчивости, акцентируя внимание на практике Chaos Engineering. Дополнительно рассматриваются ключевые метрики (SLA, MTBF, MTTR, MTTD), позволяющие количественно оценивать стабильность системы.

Summary

The article examines the primary aspects of application reliability in the financial sector, covering key concepts such as malfunction, failure, and single point of failure. It emphasizes that even brief service disruptions can lead to serious financial, reputational, and operational losses. Knight Capital, TSB Bank, and Citibank serve as notable examples, having incurred multimillion-dollar damages due to software errors. The article proposes microservice architecture, comprehensive testing, monitoring, and alert systems as effective methods to enhance fault tolerance, highlighting the importance of Chaos Engineering. Additionally, it

reviews essential metrics (SLA, MTBF, MTTR, MTTD) that facilitate quantitative assessment of system stability.

Түйіндеме

Мақала қаржы секторындағы қосымшалардың сенімділігінің негізгі қырларын қарастырады, соның ішінде істен шығу, ақау және барлық жүйенің істен шығуына әкелетін негізгі әлсіз буын сияқты маңызды ұғымдар қамтылған. Уақытша ғана қызметтің үзілуі де ауқымды қаржылық, беделдік және операциялық шығындарға әкелуі мүмкін екені атап көрсетіледі. Knight Capital, TSB Bank және Citibank компаниялары бағдарламалық қамтамасыз етудегі қателіктерден миллиондаған АҚШ долларына тең шығынға ұшыраған нақты мысалдар ретінде талданады. Мақалада қауіпке төзімділікті арттырудың тиімді әдістері ретінде микросервис сәулетін, кешенді тестілеуді, мониторинг пен ескерту жүйелерін енгізу ұсынылады, сондай-ақ Chaos Engineering тәжірибесінің маңыздылығына ерекше назар аударылады. Бұған қоса, SLA, MTBF, MTTR, MTTD сынды негізгі көрсеткіштерді пайдалану жүйенің тұрақтылығын сандық тұрғыдан бағалауға мүмкіндік беретіні көрсетілген.

Ключевые слова: надежность, финансовые приложения, отказоустойчивость, микросервисы, SLA (Service Level Agreement), тестирование, мониторинг и алерты, Chaos Engineering, репутационные риски, банковский сектор

Keywords: кeliability, financial applications, fault tolerance, microservices, SLA (Service Level Agreement), testing, monitoring and alerts, Chaos Engineering, reputational risk, banking sector

Значимость надежности приложений

Современная финансовая сфера характеризуется непрерывным ростом объемов транзакций и повышенными требованиями к доступности сервисов. Каждая минута простоя в финансовых приложениях способна приводить к миллионным убыткам. При этом даже кратковременные неполадки не только снижают финансовые показатели, но и подрывают доверие клиентов, что усиливает репутационные риски.

Рассмотрим основные понятия:

Сбой – отклонение работы функционала от изначально заданных характеристик

Отказ – вся система прекращает предоставления функционала пользователям.

Точка отказа – компонент при сбое которого становится недоступна вся система.

Надежность приложений – способность работы системы корректно даже при неблагоприятных факторах а также способность предотвратить перетекание сбоя участка системы в отказ всей системы [1, с. 27].

Надежность приложений это ключевой фактор в критических сферах таких как финансовые, банковские сферы. Нарушение стабильности работы приложения способно обусловить финансовые, репутационные и операционные потери.

Рассмотрим основные потери:

1. Финансовые потери.
    a. Прямые убытки. В финансовых системах, платежных системах сбои, отказы, ошибки в транзакциях могут привести к потерям денежных средств пользователей.

   b. Простой бизнеса. Недоступность приложения вызывает прямое сокращение выручки компании.
   c. Штрафы регуляторов и судебные иски. В финансовых и банковских сферах существует большое количество нормативных требований, сбой системы часто приводит к их нарушениям, как следствие, ведет к большим штрафам для компании.
2. Репутационные потери.
   a. Потеря доверия клиентов. Регулярные сбои в приложении формируют у клиента обоснованное беспокойство о своих денежных средствах на счетах в банке.
   b. Негативный PR. Сбои в крупных компаниях часто освещаются в СМИ и социальных сетях что влияет на имидж компании и может привести к снижению стоимости акций на фондовых рынках.
   c. Снижение конкурентноспособности. Высокая надежность может быть конкурентным преимуществом для бизнеса, в свою очередь проблемы со стабильностью приложения приводят к оттоку клиентов к конкурентам.
3. Операционные потери.
   a. Снижение эффективности труда сотрудников. Сотрудники компании вместо развития бизнеса вынуждены работать с решением инцидентов и решения их последствий.
   b. Увеличение затрат на поддержку системы и клиентов. Повышенная частота инцидентов приводит к возрастанию расходов на мониторинг и логирования работы системы.
   c. Безопасность данных. Сбои в приложениях могут приводить к утечки клиентских данных и проявить уязвимости системы для дальнийших хакерских атак.

Крупнейшие сбои в финансовых и банковских системах и их последствия:

1. В 2012 году Knight Capital, одна из крупнейших фирм-маркетмейкеров в США, обновила программное обеспечение с критической ошибкой в результате чего алгоритм начал некорректно исполнять торговые ордера что привело к потере $440 млн. в течение 45 минут. Компания была поглощена.
2. В 2018 году TSB Bank попытался перенести данные 5.2 млн клиентов на новую платформу. В результате около миллиона клиентов не могли воспользоваться услугами интернет-банкинга в течение недели, кроме того некоторые клиенты сообщали что могли видеть информацию о счетах других клиентов. Совокупные убытки превысили $500 млн.
3. В 2020 году сотрудники Citibank случайно отправили кредиторам $900 млн. платежей вместо запланированных $7.8 млн. из-за неудобного интерфейса банковского ПО. В дальнейшем Citibank не смог добиться в суде возврата $500 млн от клиентов которые отказались возвращать денежные средства.

Современные методики повышения надежности приложений

1. Отказоустойчивая архитектура и микросервисы.

Проектирование отказоустойчивой архитектуры должно включать анализ потенциальных точек отказа и предпринимать меры для их сокращения.

При разработке информационной системы важно выделить ключевой и второстепенный функционал. Основная идея - дать пользователю возможность продолжать пользоваться нашим приложением, даже если наблюдаются проблемы с второстепенным функционалом.

Рассмотрим пример процесса оформления онлайн кредита через мобильное приложение банка.

Основной функционал в данном случае - чтобы пользователь смог подать заявку на кредит и получить решение. Существуют также второстепенные компоненты, которыми можно пренебречь при деградации сервисов. Главное - чтобы их неисправность не влияла на ключевой функционал. В нашем примере к таким компонентам относятся – персональная рекомендация других банковских продуктов, форма оценки удобства пользованием и т д.

Представляется целесообразным разделить функционал рекомендаций и формы обратной связи на отдельные микросервисы чтобы изолировать их сбои. В свою очередь в основном сервисе выдачи кредита отказ второстепенных микросервис мы должны сделать ожидаемым и корректно обработать его.

Способы обработки отказа второстепенного микросервиса:

- Обработка исключений вызова. Если при вызове сервиса возникло исключение оно перехватывается и логируется.
- Установка минимального таймаута вызова. Из-за деградации второстепенного сервиса пользователь не должен долго ждать загрузки ключевого функционала.
- Сделать возможность отключения вызова сервиса «на горячую». При выявлении избыточной задержки ответа или некорректной передачи данных сервис может быть отключен вручную без ожидания истечения таймаута.
- Шаблон «размыкания цепи» (circuit breaker). Существуют библиотеки которые при недоступности сервис автоматически ограничивают его вызов не дожидаясь таймаута а при восстановлении продолжат вызывать его в штатном режиме.

Таким образом, при выбросе исключения, срабатывании таймаута или ручном отключении сервиса система вернет пустой список предложений банковских продуктов а на стороне фронтедна такой блок скроется.

При возникновении исключений, превышении времени ожидания или ручном отключении сервиса система возвращает пустой список рекомендаций, а на уровне интерфейса пользователя соответствующий блок скрывается.

Дополнительные преимущества микросервисов:

- Гибкое масштабировние - существует возможность увеличить мощность только необходимых сервисов.
- Упрощение обновлений – микросервис можно обновлять по отдельности без остановки всей системы.

2. Тестирования приложений

Тестирование – один из ключевых элементов обеспечения надежности систем. Оно помогает выявлять ошибки и потенциальные точки отказа еще на этапе разработки, существенно снижая риск сбоев.

Основные виды тестирования для повышения надежности

- Юнит-тесты. Проверяют отдельные модули приложения, изолируя их от остальных модулей и компонентов системы. Отлично подходят для тестирования различных бизнес алгоритмов таких как калькуляция банковских комиссий, проверка лимитов, валидация данных [2, с. 188].

- Интеграционное тестирование. Проверяет взаимодействие между модулями и сервисами. В финансовых приложениях критически важно тестировать взаимодействие с базами данных, платежными шлюзами и т. д
- Нагрузочное тестирование. Проверяет как система работает под высокой нагрузкой. Данный аспект особенно важен для платежных платформ, систем интернет-банкинга и и биржевых систем, обрабатывающих многомиллионные параллельные сессии. Важно проводить нагрузочное тестирование как отдельных микросервисов так и комплексное тестирование всех сервисов участвующих в процессе.
- Тестирование на отказоустойчивость. Исскуственно создаются сбои в инфраструктуре системы чтобы проверить как приложение реагирует на аварии. Данная методика тестирования (Chaos Engineering) считается одной из наиболее современных и была популяризована компанией Netflix [3].

Одной из современной практики является автоматический запуск тестов перед развертыванием приложения в боевой среде.

3. Мониторинг и алерты

Система мониторинга и алертов позволяет оперативно выявить проблемы с приложением, проанализировать их и принять соответствующие меры для восстановления и минимизации потерь.

Виды мониторинга:

- Инфраструктурный. Отслеживание серверов, баз данных. Основные метрики: загрузка CPU, потребление памяти, доступность портов.
- Мониторинг приложений. Анализирует производительность кода и внутренних сервисов. Метрики: время отклика, ошибки кода, время выполнения запросов.
- Бизнес-мониторинг. Контроль за бизнес-процессами позволяет выявить аномалии без явных ошибок систем, например резкое снижение банковских транзакций, нетипичное для такого времени суток.

Мониторинг не будет достаточно эффективен без настроенной системы алертов. Алерты позволяют оперативно реагировать на сбои и отказы и предотвратить более серьезные последствия.

Виды алертов:

- Критические алерты. Система перестала работать – требуется немедленное вмешательство. Примеры: недоступность критических сервисов, отказ базы данных, высокая загрузка CPU. В таких случаях уведомления отправляются по SMS, мессенджерам, осуществляются звонки в том числе автоматические ответственным лицам.
- Предупреждающие алерты. Может произойти сбой в системе если не принять меры. Примеры: увеличение времени откликов сервисов или запросов к базе данных, не тепичное снижение бизнес показателей. Каналы оповещения: рабочие чаты, электронная почта.
- Информационные алерты. Дополнительные полезные сведения которые не требуют срочной реакции. Примеры: успешный деплой приложений, рост количества активных сессий.

Показатели надежности приложения

Надежность приложения можно измерить с помощью ключевых метрик, которые помогают оценить стабильность работы приложения, предсказуемость отказов и быстроту восстановления.

- Доступность (Availability, Uptime) – это способность системы долго и непрерывно находится в рабочем состоянии. Часто используется такой термин как SLA (service level agreement). Фактически это отношение времени доступности приложения к времени всего [4, с.43].

SLA	Суточный простой	Недельный простой	Месячный простой	Годичный простой
99 %	14,40 мин	1,68 час	7,31 час	3,65 дня
99.9 %	1,44 мин	1,08 мин	43,83 мин	8,77 час
99.99 %	8,64 сек	1,01 мин	4,38 мин	52,60 мин
99.999 %	864 мс	6,05 сек	26,30 сек	5,26 мин
99.9999 %	86 мс	604,8 мс	2,63 сек	31,56 сек

- Среднее время безотказной работы (MTBF, Mean Time Between Failures). Отношение общего времени работы к количеству отказов.
- Среднее время восстановления (MTTR, Mean Time To Repair). Отношение времени простоя (восстановления) к количеству сбоев.
- Среднее время обнаружения проблемы (MTTD, Mean Time to Detect). Отношение времени обнаружения проблем к количеству инцидентов.

Помимо SLA, все более широкое применение находит концепция SLO (Service Level Objectives). В отличие от SLA, задающего публично декларируемый уровень доступности для клиентов, SLO фокусируется на внутренних целевых показателях надежности и производительности, которые команды разработки и эксплуатации используют для отслеживания и улучшения качества сервиса [5, с. 78].

Список литературы:

1. Клепман Мартин. Высоконагруженные приложения. Программирование, масштабирование, поддержка. Астана: Спринт Бук, 2024 – 640 с.
2. Хориков Владимир. Принципы юнит-тестирования. Спб.: Питер, 2022 – 320 с.
3. Netflix. Chaos Monkey. [Электронный ресурс]. URL: https://github.com/Netflix/chaosmonkey
4. Сюй Алекс. System Design. Подготовка к сложному интервью. Спб.: Питер, 2022 – 304 с.
5. Бетси Бейер, Крис Джоунс, Дженнифер Петофф, Нейл Ричард Мерф. Site Reliability Engineering. Надежность и безотказность как в Google. — СПб.: Питер, 2019. — 592 с.