Знания

- О наличии инженерной практики определения основных индикаторов жизнеспособности сервиса и наблюдения за уровнем обслуживания через проектирования "целей" для индикаторов
- О критический необходимости в практике работы сервисов фиксации уровня обслуживания и применения описанной выше инженерной практики
- О корнях формирования практики и ее практического применения в рамках промышленной эксплуатации сервисов

Умения

- Способность анализа контекста и формирования индикаторов для наблюдения
- Способность нахождения нужного инструмента для решения прикладной задачи
- Будут закреплены умения полученные в рамках семинаров 2,3 в рамках домашней работы

Навыки

- Будет выработан навык самостоятельного принятия решения выбора уровня обслуживания сервиса, формирования индикаторов и целей для индикаторов

Теоретическая вводная

Что такое метрика? Определений может быть много:

- Набор чисел, которые описывают информацию о конкретных процессах или активностях (Dictionary Cambridge)
- Измерения количественной оценки, которые обычно используют для оценки, сравнения и отслеживания производительности производства (Investopedia)
- Мера, позволяющая получить численное значение некоторого свойства программного обеспечения

Когда мы проектируем любую систему (простую или сложную), в том числе и программное обеспечение - мы понимаем что у нее есть параметры, характеристики и свойства, которые характеризуют нашу систему и процессы, которые система обслуживает/симулирует. Чем

сложнее система - тем больше в себе она может содержать параметров, процессов, свойств и характеристик. Нам же требуется возможность оценивать интересующие нас измерения.

Уже в случае standalone сервиса - это по умолчанию может быть сложная система (сложная бизнес логика + внутренние механизмы ЯП + механизмы операционной системы)

Даже при довольно высоком уровне абстракции в современной Cloud-парадигме, когда мы можем оперировать уровнем вычислительных ресурсов (CPU, RAM) - без измерений не обойтись, так как мы говорим о конечных ресурсах.

В прошлый раз мы говорили о сборе метрик операционной системы. Это очень важный пласт, но они описывают процессы уровня ОС + железа и мало могут сказать о том, правильно ли работает наш сервис. Потому что метрики показывают измерения определенного процесса - планировщик задач, работа vmm системы, производительность ОС.

Детализация может быть колосальной. И в то же время избыточной. Также метрики могут показывать ничего. Наша задача, как инженеров - определить/ограничить тот набор параметров, процессов за которыми мы должны следить. В случае проектирования систем - мы должны определить как можно измерить интересующий нас аспект. В случае эксплуатации - какой набор может сказать нам о жизнеспособности, надежности и часто стоимости сервиса.

Переизбыток информации, как и недостаток может привести к падению качества принимаемых решений.

Проведем мыслиный эксперимент. Есть сервис, который установлен на N хостов. По мониторингу все хорошо - все системы работают, приложения работают. При этом нет проблем с saturation (перевести) на нодах. Но к вам обращаются пользователи и говорят, что сервис не работает. При повторном наблюдении за системными метриками - вы находите следующую ситуацию - кол-во 4XX ответов на данный момент в 3 раза больше чем каждый день за прошлую неделю (естественно в то же временное окно). По логам вы находите затем ошибки обращения к внешнему сервису авторизации и понимаете, что пользователи просто не могут попасть в вашу систему.

Давайте представим что в обоих случаях у нас было информирование нас о проблемах. В первом случае мы получаем ложно положительное срабатывание, что может привести к стрессу в команде, потраченному времени, а иногда даже неправильно выбранному пути при поиске причины (сработала системная метрика на одном из хосте и мы побежали искать черную кошку в комнате без света, при условии что ее нет). Во втором случае наши системы были в порядке, но при этом пользователи не получали сервис.

Подходы к принятию решения о выборе метрик бывают разные - RED (Rate/Error/Duration), USE (Usage/Saturation/Errors), 4GS (Latency, Traffic, Error, Saturation)
Также не стоит забывать о подходах к процессам - сверху-вниз (когда мы проходим все слои вплоть до механизмов ОС и поведения на уровне железа), слева-направо (когда мы изучаем как проходят данные через различные участки нашей системы).

Когда мы описали тот набор процессов и выбрали нужный уровень абстракции - мы можем добавить еще больше системного подхода и начать проектирование SLI/SLO

SLI/SLO/SLA

Соответственно возникает необходимость в формировании метрик, которые были бы связаны с нашим бизнес контекстом. То есть, объявить SLI для нашего сервиса. Обратимся к определению SLI из Google SRE Book:

- An SLI is a service level indicator—a carefully defined quantitative measure of some aspect of the level of service that is provided. (Вольный перевод: выверенное количественное измерение аспекта предоставляемого сервиса, тут я специально опустил перевод уровня, потому что на старте сложно рассуждать в целом каким либо уровнем)

Что же мы понимаем под выверенным измерением. А то, что диктует ваш контекст. Давайте обратимся к Oncall сервису. Что он позволяет нам сделать:

- Завести команду
- Завести в команду специалистов
- Назначить расписание для дежурств
- Автоматически распределять дежурных по ростеру

При этом для разных команд важный контекст будет разниться.

Обратимся опять к фантазии. У вас есть несколько сервисов на аутсорсе и вам важно, чтобы к каждому сервису была своя команда и при этом в этих командах был бы хотя бы один дежурный в рабочие часы вашей фирмы. На этом завязан допустимый уровень поддержки.

Индикаторы стоит выбирать для той функциональности, что критично на данный момент времени и, как и многие процессы, проводить раунды ревью.
От индикаторов переходим к "показателям" (вольный перевод objectives).

Снова обратимся к SRE Book - a target value or range of values for a service level that is measured by an SLI. То есть мы знаем что мы мерим, но нам также требуется определить какие показатели для каждого индикатора мы имеем.

Очень часто SLO расчитывается исходя из суммарного влияния для всего сервиса. К примеру мы говорим, что сервис будет доступен 99% времени и это означает, что он будет 99% времени находится в стабильном состоянии по всем индикаторам.

И вот наступает момент, когда мы определили метрики, установили порог и выбрали цель для SLO. Стоит обратить внимание, что в SLO лучше всего по максимуму осветить все этапы обработки запроса. Можете снять метрику с клиента сколько занимает резолв имени? Вперед. Есть время обработки запроса на проксях и LB? Да, дайте парочку. Но опять не стоит забывать, что с каждым новым индикатором - обслуживать ваш SLO будет все дороже и дороже. Поэтому добавлять их нужно по мере взросления проекта.

В случае если вообще нет ни 1 индикатора - даже индикатор жизни (сервис отвечает 200 при обращении) - уже хороший старт.

Провели какое-то время в наблюдении и итогом знаем что SLO у нас считается правильно и не "шумит" (к примеру нет ложно-позитивных срабатываний). Именно на этом этапе мы готовы предоставлять SLA, не раньше. И он не может быть больше фактического значения SLO на момент времени.

Почему? Потому что SLA это соглашение о нарушении определенного уровня SLO, которое мы нашли достоверным для нашего сервиса и который мы можем выдержать. То есть какие пенальти будут, если мы нарушим SLO.

Но в жизни SLA и SLO часто путают между собой. Поэтому надо привыкнуть что SLA == SLO в обсуждении, а под SLO будут понимать внутренний вариант контракта.

А теперь как именно мы можем получить нужные нам метрики под индикаторы?

Прежде всего стоит обратить внимание на то, что наш сервис может быть как "черным" ящиком, так и "серым" или даже "белым" (мы можем сами активно разрабатывать проекты).

В случае с "черным" ящиком можно использовать:

- Хелсчекеры (к примеру blackbox_exporter)
- Системные метрики
- Метрики с промежуточных узлов (с балансеров, проксей, сторонних сервисов)

Если мы более менее знакомы с контрактом сервиса и можем говорить о нем, как о "черном" ящике, то в дополнении:

- Можем написать свой экспортер (как примеру haproxy_exporter)
- Можем написать пробер с отдачей метрик (условный e2e тест, который мы могли бы запускать и получать информацию о работоспособности с точки зрения пользователя)

В случае с белым ящиком:

- Мы можем выставлять метрики сами (и это нам поможет намного лучше, так как мы будем смотреть с точки зрения самого сервиса и сможем октазаться от мониторинга сторонних сервисов)

Ставим blackbox_exporter и проверяем работоспособность своего сервиса

Краткий алгоритм:

- Забираем исполняемый/инсталлянционный файл под нужную систему

```
wget "https://github.com/prometheus/blackbox_exporter/releases/download/v0.22.0/blackbo
```

- Производим установку (распаковка, сборка, укладка в нужные директории)

```
# Unarchive blackbox exporter
tar xvf /tmp/blackbox_exporter-0.22.0.linux-amd64.tar.gz -C /tmp/
# Copy executable into FHS standard directory
cp /tmp/blackbox_exporter-0.22.0.linux-amd64/blackbox_exporter /usr/local/bin/blackbox_e
```

- Производим дополнительные настройки и конфигурирование сервиса (отдельный пользователь, специальная директория, прописывание по иерархии FHS в нужное место, заводим демона/сервис)

```
# Create config directory
mkdir -p /etc/blackbox_exporter
# Create user and group for blackbox_exporter
useradd --no-create-home --shell /bin/false blackbox_exporter
# Prepare configuration file
cat << EOF > /etc/blackbox_exporter/blackbox.yml
modules:
    http_2xx:
        prober: http
        timeout: 1s
        http:
            valid_http_versions: ["HTTP/1.1", "HTTP/2.0"]
            method: GET
            headers:
                Host: localhost:8080
            no_follow_redirects: false
EOF
# Change permissions
chown -R blackbox_exporter:blackbox_exporter /etc/blackbox_exporter

# Create systemd service and start
cat << EOF > /lib/systemd/system/blackbox.service

[Unit]
Description=Blackbox exporter service

[Service]
User=blackbox_exporter
Group=blackbox_exporter

ExecStart=/usr/local/bin/blackbox_exporter --config.file /etc/blackbox_exporter/blackbo

[Install]
WantedBy=multi-user.target

EOF

systemctl daemon-reload
systemctl enable blackbox.service
systemctl start blackbox.service

# Remove archive
rm -rf /tmp/blackbox_exporter-0.22.0.linux-amd64
```

- Проверяем установку (ручной/автоматический режим)
- Наслаждаемся

Проектируем свой экспортер

Бизнес-контекст: как сервису по предоставлению SRE услуг, нам критически важно, чтобы у каждого пользователя системы был предоставлен хотя бы 1 контакт. Наличие пользователей без номера > 15% может привести с учетом ротации к выходу такого пользователя на линию, что в свою очередь приведет к репутационным рискам, так как мы не сможем нормально проинформировать специалиста о критических ситуациях для обслуживаемых сервисов.

SLI: Кол-во пользователей без контакных данных
SLO: <15% от общей базы пользователей
SLA: При нарушении вышелижащего SLA - мы предоставляем от 1 до 1 мес бесплатных услуг. Вес в SLA находится на уровне 10% от общего SLA. При нарушении вводятся маротории на новые релизы и назначается стабилизация по бизнес-потоку обслуживания.

В данном случае индикаторов может быть нескольлко. Мы спроектировали только одну метрику - которая следит, что у нас нет пользователей без номера.

По коду:

- Функция опроса низлижащего сервиса:

```
def populate(self) -> None:
    logging.debug("Making a request to Oncall installation")
    EXPORTER_API_REQUESTS_TOTAL.inc()
    resp = requests.get("%s/users" % (self.oncall_api_url))
    if resp.status_code != 200:
        EXPORTER_API_REQUESTS_FAILED_TOTAL.inc()
    failed_users = 0
    for user in resp.json():
        if not user['contacts']:
            logging.debug(f"Found user without contacts: {user['name']}")
            failed_users += 1
    EXPORTER_USERS_WITHOUT_CONTACTS_GAUGE.set(failed_users)
```

- Объявление метрик

```
EXPORTER_API_REQUESTS_TOTAL = Counter(
    "exporter_api_requests_total", "Total count of requests to oncall API"
)
EXPORTER_API_REQUESTS_FAILED_TOTAL = Counter(
    "exporter_api_requests_failed_total", "Total count of faled requests to oncall API"
)
EXPORTER_USERS_WITHOUT_CONTACTS_GAUGE = Gauge(
    "exporter_users_without_contacts_gauge", "Check if we have users without contact da
)
```

- Хорошие практики (напомним о 12factor):

1. Конфигурационный блок с инициализацией через переменные окружения

```
class Config(object):

    oncall_exporter_api_url = env("ONCALL_EXPORTER_API_URL")
    oncall_exporter_scrape_interval = env.int("ONCALL_EXPORTER_SCRAPE_INTERVAL", 60)
    oncall_exporter_log_level = env.log_level("ONCALL_EXPORTER_LOG_LEVEL", logging.INFO
    oncall_exporter_metrics_port = env.int("ONCALL_EXPORTER_METRICS_PORT", 9091)
```

2. Обработка системных сигналов

```
def terminate(signal, frame):
    print("Terminating")
    sys.exit(0)
```

Список литературы (в нашей случае без разделения семинарист/студент)

- Betsy Beyer, Niali Richard Murphy, David K. Rensin, Kent Kawahara & Stephen Thorne, The Site Reliability Workbook, 2018 - Chapter 2, Chapter 3
- SLA vs. SLO vs. SLI: What's the difference?, доступно на: https://www.atlassian.com/incident-management/kpis/sla-vs-slo-vs-sli
- SLOs and SLIs best practices for system, доступно на: https://newrelic.com/blog/best-practices/best-practices-for-setting-slos-and-slis-for-modern-complex-systems
- What are SLOs? How service-level objectives work with SLIs to deliver on SLAs, доступно на: https://www.dynatrace.com/news/blog/what-are-slos/
- The RED Method: key metrics for microservices architecture, доступно на: https://www.weave.works/blog/the-red-method-key-metrics-for-microservices-architecture/

Примеры требований к сервису:

- Сервис должен отвечать < 2с. в 95% случаев
- Сервис должен быть доступен хотя бы на половине нод
- Кол-во внутренних ошибок не должно быть выше 2%
- Дополнительные требования из бизнес-контекста =3

Домашнее задание

- (Творческое) Придумать контекст для которого будем готовить метрики
- Для указанного контекста придумать хотя бы 2 индикатора за которыми вы будете следить
- Спроектируйте exporter с отловом указанных метрик
- Спроектируйте SLO для нашего сервиса (референс - SRE Workbook там хорошо описано)

Критерии оценки

- Экспортер спроектирован, в отчете присутсвует запись их захвата/изменения + пояснение почему именно указанные метрики - 5 баллов
- Спроектирован SLO только по спроектированным метрикам + пояснение почему именно такой уровень - 7 баллов
- Спроектирован SLO по спроектированным бизнес метрикам + по системным метрикам - 9 баллов

В видео отчете должны присутствовать:

- Работа экспортера (удачный кейс/ошибочный кес)
- Рассказ на 3-5 минут почему был выбран такой SLO и метрики
- Ссылка на репозиторий с экспортером