Что делать если JIRA работает недостаточно быстро?

Немного из жизни Джировода в отставке

Яндекс

- С 2008 года в течении четырех лет моей основной работой было развитие, оптимизация и расширение Atlassian JIRA в компании Яндекс.
- За это время я досконально изучил кодовую базу JIRA с версии 3.13 до 5.2
- На тот момент это была одна из самых больших инсталляций JIRA по объему данных

ALM Works

- Консультировал компанию ALMWorks при создании их знаменитого плагина Structure
- На сегодняшний день я в рамках сотрудничества с ALMWorks разрабатываю мобильное приложение PocketDesk

Каковы же основные причины проблем?

- Thread Pool
- Plugin System
- Garbage Collector
- Cache
- Lock Contention
- Bugs

Thread Pool

- Нередко, запрос прежде чем начать обрабатываться, ожидает пока закончит обрабатываться другой запрос
- Большой пул – плохо
- Маленький пул – плохо

Plugin System

За гибкость надо платить

Garbage collector

- Некоторые данные «живут» дольше чем нужно
- Некоторые данные «живут» меньше чем хотелось бы
- Полная сборка мусора иногда создает лавиноподобную нагрузку
- Полная сборка мусора останавливает работу JIRA целиком

Cache

- Что бы лишний раз не обращаться к хранилищу данных, мы поместим редко изменяющиеся данные в память
- Зачем усложнять себе жизнь сложной схемой инвалидации, если можно опустошить кэш целиком?

Lock Contention

- Иногда потоки обработки выстраиваются в очередь
- Иногда это оправдано
- Иногда это не оправдано

Что делать?

- Бесплатно
- Дешево и сердито
- Дорого
- Очень дорого

Бесплатно

- Обновляться до последней доступной версии
- Писать в Support

Как писать в Support

- Быть готовым к идиотским вопросам
- Быть готовым к тому, что это может занять очень много времени
- Максимально конкретизировать запрос
- Быть настойчивым

Дешево и сердито

- Закэшировать статику «намертво»
- Купить максимально производительное железо
- Отключить расширения в которых вы не уверены

Кэширование статики

```
proxy_cache_path  /var/cache/nginx/jira levels=1:2 keys_zone=jira:32m max_size=1000m inactive=60m;

location ~* ^/s/(.*)/_/(.*)(gif|jpg|jpeg|js|css|png)$ {
    expires     max;
    add_header  Cache-Control public;
    more_clear_headers 'Last-Modified' 'ETag';
    proxy_next_upstream error timeout http_404 http_500;
    proxy_pass http://jira-server;
    proxy_cache jira-bugs;
    proxy_cache_key  "$request_uri";
    proxy_cache_valid 200;
}
```

Дорого

- Выделить человека с хорошим знанием Java для исследования причин низкой производительности
- Найти причину и разработать решение, устраняющее ее
- Поддерживать это решение

Очень дорого

Нанять эксперта и попросить сделать «хорошо»

Как это делаю я

- Сбор данных
- «Танк»
- Мишень

Сбор данных

- Access логи веб-сервера
- Логи работы GC
- Thread Dumps
- Профайлер

Формат Access лога

```
log_format  jira  '$remote_addr - $remote_user [$time_local] "$request" '
□$status $body_bytes_sent "$http_referer" "$http_user_agent" "$host,$server_port" '
□"$http_x_forwarded_for" "$http_cookie" 0 "$request_time" "$gzip_ratio" '
□"$upstream_addr" $upstream_status $upstream_response_time';
access_log  /var/log/jira/jira.access.log jira;
□
```

Логи работы GC

```
GCLOG_OPTION="-XX:+PrintGCTimeStamps \
-XX:+PrintGCDetails \
-verbose:gc \
-XX:+PrintGCApplicationStoppedTime\
-XX:+PrintGCApplicationConcurrentTime\
-Xloggc:$CATALINA_HOME/logs/jiragc-$(date +%Y%m%d-%H%M%S).log"

JAVA_OPTS="$GCLOG_OPTION $JAVA_OPTS"
```

Thread Dumps

```
#!/bin/bash
if [ -f /var/run/jira/jira.pid ] ; then
    LOG_FILE=/var/log/jira/threads_dump.log.gz
    DATE=$(date +%Y%m%d-%H%M%S)
    (  if flock -ne 200; then
        PID=$(cat /var/run/jira/jira.pid)
        JSTACK=$(cat /proc/${PID}/cmdline | strings | head -1 | xargs -I {} dirname {})/jstack
        echo "=== threads dump: pid=$PID timestamp=$DATE ===" | gzip >> $LOG_FILE
        $JSTACK $PID 2>&1 | gzip >> $LOG_FILE
    else
        echo "[$DATE] thread dump process already running" >> /var/log/jira/threads_dump_run.log
    fi
    ) 200>>$LOG_FILE
fi
```

Профайлер

```
YJP_VERSION=11.0.8
YJP_HOME=$CATALINA_HOME/bin/yjp-$YJP_VERSION
YJP_JAVA_OPTS="-agentlib:yjpagent=dir=/tmp/ -agentlib:yjpagent=disableexceptiontelemetry"

# Specify path to proper version of profiler agent library, depending on the OS
if [ `uname` = 'Linux' ] ; then
  if [ "`uname -a | grep x86_64`" ] ; then
    # Assume Linux AMD 64 has 64-bit Java
    export LD_LIBRARY_PATH="$YJP_HOME/linux-x86-64:$LD_LIBRARY_PATH"
  else
    # 32-bit Java
    export LD_LIBRARY_PATH="$YJP_HOME/linux-x86-32:$LD_LIBRARY_PATH"
  fi
else
  YJP_JAVA_OPTS=""
  echo "Unsupported platform for YourKit profiler: `uname`"
fi
JAVA_OPTS="$YJP_JAVA_OPTS $JAVA_OPTS"
```


«Танк»

Как все это использовать?

1. Выбираем интересующий нас URL или группу
2. Развертываем мишень и танк. Генерируем патронную ленту для танка
3. Стреляем первый раз, что бы оценить пиковую производительность
4. Стреляем под сэмплирующим профайлером
5. Стреляем под трасирующим профайлером
6. Вносим измеения
7. Повторяем и сравниваем

Спасибо за внимание

AMA