

# Рекомендации по тонкой настройке сервера

- Оптимизация нагрузки на сервер и потребления памяти
  - Настройка сборщика мусора
  - Настройка оперативной памяти
  - Настройка REST клиента
  - Ограничение логирования
- Настройка UDP
- Оптимизация загрузки канала
  - Ограничение битрейта при публикации
  - Ограничение битрейта на сервере
- Изменение диапазона динамических портов в Linux
- Изменение максимального количества открытых файлов

Настройки сервера по умолчанию в большой степени универсальны и могут нуждаться в подстройке под определенный сценарий клиента.

## Оптимизация нагрузки на сервер и потребления памяти

### Настройка сборщика мусора

Важной частью Java VM является сборщик мусора (Garbage Collector). При запуске сборщик мусора резко увеличивает нагрузку на сервер и может приостановить выполнение остальных задач, поэтому рекомендуется минимизировать его запуски при помощи следующих настроек в файле [wcs-core.properties](#)

```
#Disable heuristic rules
-XX:+UseCMSInitiatingOccupancyOnly

#Reduce Old Gen threshold
-XX:CMSInitiatingOccupancyFraction=70

# Use System.gc() concurrently in CMS
-XX:+ExplicitGCInvokesConcurrent

# Disable System.gc() for RMI, for 10000 hours
-Dsun.rmi.dgc.client.gcInterval=36000000000
-Dsun.rmi.dgc.server.gcInterval=36000000000
```

### Настройка оперативной памяти

При стриминге в памяти создается и уничтожается много объектов с данными. Поэтому рекомендуется выделять под Java memory heap не менее, чем 1/2 физической памяти сервера. Например, если объем оперативной памяти сервера составляет 32 Гб, рекомендуется выделить 16 Гб при помощи следующих настроек в файле [wcs-core.properties](#)

```
-Xmx16g
-Xms16g
```

Кроме того, если не используется отображение статистики в веб-интерфейсе WCS, можно оптимизировать работу с памятью за счет отключения взаимодействия между ядром и административным модулем при помощи настройки в файле [flashphoner.properties](#)

```
disable_manager_rmi=true
```

### Настройка REST клиента

При использовании [REST hooks](#), на каждое действие WCS сервера (присоединение клиента, публикация и воспроизведение потока, установка звонка и т.д.) создается HTTP REST соединение к бэкэнд-серверу. При большом количестве одновременно публикующих клиентов или подписчиков, при настройках по умолчанию возможно исчерпание пула потоков встроенного REST клиента WCS, что, в свою очередь, может приводить к блокировкам (deadlock). При этом сервер перестает публиковать и воспроизводить потоки.

По умолчанию, максимальное количество одновременных REST соединений установлено в 200 при помощи параметра в файле [flashphoner.properties](#)

```
rest_max_connections=200
```

Для того, чтобы избежать переполнения пула потоков и блокировок, необходимо уменьшить это значение, например

```
rest_max_connections=20
```

Если [REST hooks](#) не используются, REST клиент может быть отключен при помощи настройки

```
disable_rest_requests=true
```

## Ограничение логирования

При использовании [REST hooks](#) в серверный лог записывается работа REST-клиента, работа встроенного бэкенда EchoApp, а также работа REST API сервера. Это приводит к большому количеству записей в лог и, следовательно, увеличивает нагрузку на сервер. При необходимости, объем логирования может быть уменьшен при помощи следующих параметров в файле [log4j.properties](#):

```
log4j.logger.RestClient=WARN
log4j.logger.EchoApp=WARN
log4j.logger.RestApiRouter=WARN
```

## Настройка UDP

При стриминге медиаданные передаются UDP-пакетами. UDP-пакеты могут отбрасываться, если, например, сервер не успевает разобрать очередь пакетов, что ведет к ухудшению качества изображения, фризам. Для того, чтобы этого избежать, необходимо подстроить буферы UDP-сокеты настройками в файле [flashphoner.properties](#)

```
rtp_receive_buffer_size=131072
rtp_send_buffer_size =131072
```

а также настроить системные очереди командой

```
ip link set txqueuelen 2000 dev eth0
```

Для того, чтобы диагностировать проблему с UDP, необходимо отследить сброс UDP-пакетов командой

```
dropwatch -l kas
>start
```

## Оптимизация загрузки канала

Качество картинки при воспроизведении у пользователя зависит от битрейта: чем выше битрейт, тем выше качество. Однако, чем выше битрейт, тем больше загружается канал передачи данных, и, если полоса пропускания между сервером и клиентами ограничена, есть вероятность, что канал будет загружен полностью. Это приводит к сбросу битрейта и резкому снижению качества картинки.

В связи с этим, необходимо ограничивать битрейт для того, чтобы обеспечить достаточное качество картинки при приемлемой загрузке канала.

## Ограничение битрейта при публикации

Для снижения нагрузки на канал от публикующей стороны до сервера можно задать минимальное и максимальное значения битрейта в кбит/с в скрипте публикации при помощи JavaScript API

```
session.createStream({
  name: streamName,
  display: localVideo,
  constraints: {
    video: {
      minBitrate: 500
      maxBitrate: 1000
    }
  }
  ...
}).publish();
```

## Ограничение битрейта на сервере

Минимальное и максимальное значение битрейта в бит/с на сервере устанавливается следующими настройками в файле [flashphoner.properties](#)

```
webrtc_cc_min_bitrate=500000
webrtc_cc_max_bitrate=1000000
```

Для того, чтобы исключить быстрый набор битрейта браузером, необходимо также установить параметр

```
webrtc_cc2_twcc=false
```

и включить декодирование потоков на сервере только по запросу, для снижения нагрузки на сервер

```
streaming_video_decoder_fast_start=false
```

## Изменение диапазона динамических портов в Linux

Динамический или эфемерный порт, — временный порт, открываемый при установке IP-соединения из определённого диапазона программного стека TCP/IP. Многие версии ядра Linux используют в качестве динамических порты 32768 — 61000. Проверить, какой именно диапазон используется на сервере, можно при помощи команды

```
sysctl net.ipv4.ip_local_port_range
```

Если этот диапазон пересекается со стандартными [портами](#), используемыми WCS, его необходимо изменить при помощи команды

```
sysctl -w net.ipv4.ip_local_port_range="59999 63000"
```

## Изменение максимального количества открытых файлов

В скрипте запуска webcallserver, расположенном в подкаталоге bin в каталоге установки WCS, например

```
/usr/local/FlashphonerWebCallServer/bin/webcallserver
```

в функции start() указано значение максимального количества открытых файлов

```
function start() {  
    ...  
    echo -n $"$PRODUCT: starting"  
  
    ulimit -n 20000  
    if [[ "$1" == "standalone" ]]; then  
        ...  
    fi  
    ...  
}
```

По умолчанию, данное значение установлено в 20000, но при необходимости его можно увеличить, учитывая ограничения используемой операционной системы.