Управление памятью в Java

- Настройка оперативной памяти
- Настройка сборщика мусора
  - Настройка Concurrent Mark Sweep (CMS) Collector
  - НастройкаZ Garbage Collector (ZGC)
- Настройка выделения и освобождения физической памяти на уровне системы
- Известные проблемы

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 следующихнастроек в файлеwcs-core.properties

```
-Xmx16g
-Xms16g
```

Настройка сборщика мусора

Важной частью Java VM является сборщик мусора (Garbage Collector, GS). 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

```
# 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 Concurrent Mark Sweep (CMS) Collector

Сборщик мусора Concurrent Mark Sweep (CMS) предназначен для приложений, которым требуются короткие паузы при сборе мусора и которые могут предоставить ресурсы процессора сборщику мусора, не снизив существенно производительность системы.

1. Включить CMS GC можноследующими настройками в файле wcs-core.properties(например, выделяем 24G под heap и указываем NewSize иMaxNewSize равными значениями для регулировки размеранижней и верхней границы для молодого поколения; она должна составлять от 1/4 до 1/3 максимального размера heap):

```
# Used CMS GC
-XX:+UseConcMarkSweepGC -Xms24g -Xmx24g -XX:NewSize=6144m -XX:MaxNewSize=6144m

# Disable heuristic rules
-XX:+UseCMSInitiatingOccupancyOnly

# Reduce Old Gen threshold
-XX:CMSInitiatingOccupancyFraction=70

# Log
-Xloggc:/usr/local/FlashphonerWebCallServer/logs/gc-core-
-XX:ErrorFile=/usr/local/FlashphonerWebCallServer/logs/error%p.log
```

2. После перезапуска WCS в файлах логов gc-core.log видим результат работы сборщика мусора. Вывод можетотличатьсяв зависимости от установленной версии Java. Например:

openjdk version "1.8.0_222":

```
1815.658: [GC (Allocation Failure)  1255412K->28934K(16623872K), 0.0207810 secs]
1893.220: [GC (Allocation Failure)  1255750K->29037K(16623872K), 0.0178801 secs]
1975.637: [GC (Allocation Failure)  1255853K->29715K(16623872K), 0.0176784 secs]
```

openjdk version "12.0.2":

```
[1100.631s][info][gc] GC(28) Pause Young (Allocation Failure) 104M->24M(1014M) 6.243ms
[1143.706s][info][gc] GC(29) Pause Young (Allocation Failure) 104M->24M(1014M) 3.582ms
[1198.943s][info][gc] GC(30) Pause Young (Allocation Failure) 104M->24M(1014M) 3.868ms
```

НастройкаZ Garbage Collector (ZGC)

Для Java 12 доступна экспериментальная реализации сборщика мусора Z Garbage Collector (ZGC), позволяющаяобеспечить низкую задержку при сборке мусора, не останавливая выполнения потоков приложения более чем на 10 миллисекунд, даже при работе с очень большими memory heap. Стоит обратить внимание, что ZGCтребует больше ресурсов процессора по сравнению с CMS GC.

Пример установки ZGC с версиейOpenJDK 12:

1. Установите OpenJDK 12, как описано здесь

2. Проверяем корректность установкиOpenJDK 12:

```
java -version

openjdk 12.0.2 2019-07-16
OpenJDK Runtime Environment (build 12.0.2+10)
OpenJDK 64-Bit Server VM (build 12.0.2+10, mixed mode, sharing)
```

3. Устанавливаем WCS (если требуется).

4. Если WCS был установлен ранее, комментируем или удаляем следующие строки в файле wcs-core.properties

```
-XX:+UseConcMarkSweepGC
-XX:+UseCMSInitiatingOccupancyOnly
-XX:CMSInitiatingOccupancyFraction=70
-XX:+PrintGCDateStamps
-XX:+PrintGCDetails
```

Заменяем строку

```
-Xloggc:/usr/local/FlashphonerWebCallServer/logs/gc-core-
```

на

```
-Xlog:gc*:/usr/local/FlashphonerWebCallServer/logs/gc-core-:time
```

5. Добавляем настройки в wcs-core.properties (например, выделяем 24G под heap):

- в JDK 12-15

```
# ZGC
-XX:+UnlockExperimentalVMOptions -XX:+UseZGC -Xms24g -Xmx24g
```

- в JDK 16 и выше

```
# ZGC
-XX:+UseZGC -Xms24g -Xmx24g
```

6. Если планируется использование больших страниц памяти (hugepages), добавляем настройки:

- в JDK 12 или 14

```
-XX:+UseLargePages -XX:ZPath=/hugepages
```

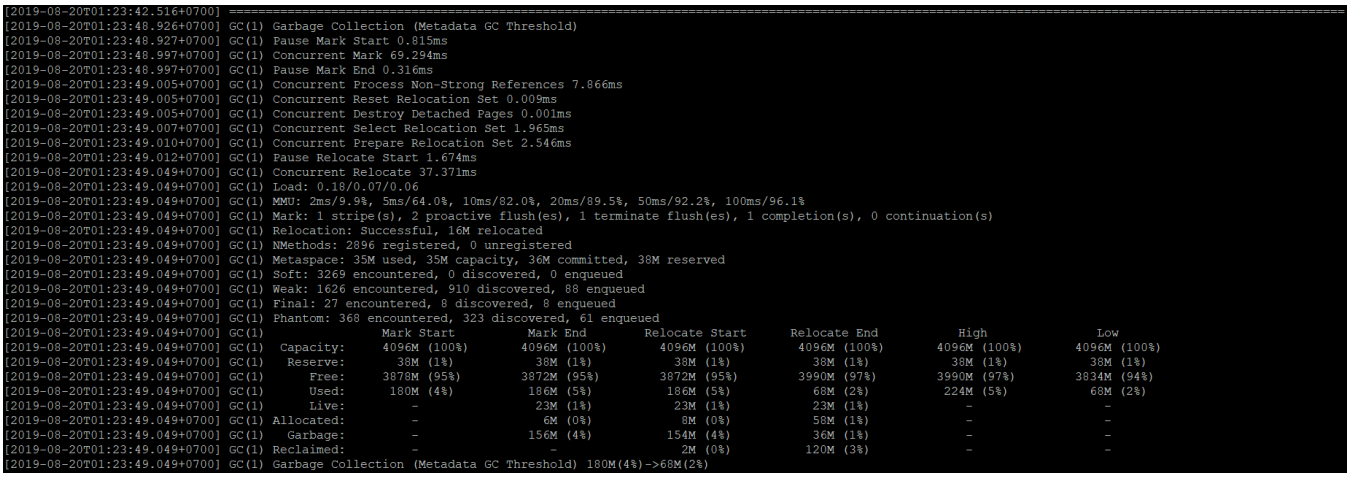
- в JDK 15 и выше

```
 -XX:+UseLargePages -XX:AllocateHeapAt=/hugepages
```

Затем настраиваем hugepages порекомендациям(число страниц памяти (по 2048K каждая) с запасом к выделенной памяти под heap (1,125 * 24G *1024 / 2M)) и добавляем необходимые параметры в автозагрузку сервера (пример для Centos):

```
sudo mkdir /hugepages
sudo echo "echo 13824 >/sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages" >>/etc/rc.local
sudo echo "mount -t hugetlbfs -o uid=0,mode=0757 nodev /hugepages" >>/etc/rc.local
sudo chmod +x /etc/rc.d/rc.local
sudo systemctl enable rc-local.service
sudo systemctl restart rc-local.service
sudo chmod o+w /hugepages
```

7. После перезапуска WCS, в файлах логов gc-core.log видна периодическая операция работы сборщика мусора. Для понимания модели работы Z Garbage Collector можно ознакомиться с презентацией.



Настройка выделения и освобождения физической памяти на уровне системы

При большой нагрузке на сервер, может оказаться недостаточно областей маппинга физической памяти (memory map areas), выделяемых процессу системой по умолчанию, что может приводить к завершению работы JVM по нехватке нативной памяти. При этом лог ошибки будет содержать следующий комментарий

```
#
# There is insufficient memory for the Java Runtime Environment to continue.
# Native memory allocation (mmap) failed to map 12288 bytes for committing reserved memory.
# Possible reasons:
#   The system is out of physical RAM or swap space
...
```

Для того, чтобы предотвратить такое поведение, необходимо увеличить количество областей маппинга памяти

```
sysctl -w vm.max_map_count=131072
```

и перезапустить WCS.

Известные проблемы

1. При использовании ZGC нагрузка на процессор сервера выше, особенно в JDK 15

Симптомы: при обновлении JDK с 12 или 14 до 15 и использовании ZGCвозрастает средняя загрузка процессора, измеренная на уровне системы (например, утилитой htop)

Решение: для высоконагруженных серверов, если требуется минимизация задержек при сборке мусора, использовать ZGC в JDK 12 или 14

2. Логи ZGC с настройками вывода по умолчанию могут занимать много места на диске

Симптомы: файлы логов `gc-core*.log` занимают много места на диске

Решение: в файле настроек wcs-core.properties ограничить набор сообщений, выводимых в лог сборщика мусора

```
-Xlog:gc,gc+start,gc+phases:/usr/local/FlashphonerWebCallServer/logs/gc-core-:time
```