|  |
| --- |
| ИнтерТраст |
| Настройка параметров серверов приложений, РСУБД и ОС |
|  |

|  |
| --- |
| Митавский Д. В. |

Содержание

[Настройка JBoss 3](#_Toc418003299)

[Настройка Postgres 6](#_Toc418003300)

[Настройка Linux 7](#_Toc418003301)

# Настройка JBoss

1. Выделить примерно 40% свободной памяти Java-машине. MaxPermSize установить в 1024М. Если на сервере, будет только JBoss, то выделяем ему 90% свободной оперативной памяти. Для этого в standalone.conf (для Linux), standalone.conf.bat (для Windows) переопределить строку:

JAVA\_OPTS="-Xms1303m -Xmx1303m -XX:MaxPermSize=256m -Djava.net.preferIPv4Stack=true"

1. Если JBoss-у выделяется более 32 гигабайт памяти, необходимо в standalone.conf/bat удалить опцию -XX:+UseCompressedOops (она там прошита намертво для 64битных машин).
2. Настроить таймаут транзакции по согласованию с заказчиком. По умолчанию 300 секунд.

<**subsystem xmlns="urn:jboss:domain:transactions:1.1"**>  
 <**core-environment**>  
 <**process-id**>  
 <**uuid**/>  
 </**process-id**>  
 </**core-environment**>  
 <**recovery-environment socket-binding="txn-recovery-environment" status-socket-binding="txn-status-manager"**/>  
 <**coordinator-environment default-timeout="300"**/>  
</**subsystem**>

Параметр **default-timeout** определяется в секундах.

1. Настроить пул соединений. Пример настройки:

<**datasource jndi-name="java:jboss/datasources/CM5" pool-name="CompanyMedia" enabled="true" use-ccm="false"**>  
 <**connection-url**>jdbc:postgresql://localhost:5432/sochi</**connection-url**>  
 <**datasource-class**>org.postgresql.xa.PGXADataSource</**datasource-class**>  
 <**driver**>postgresql-9.3-1103.jdbc41.jar</**driver**>  
 <**pool**>  
 <**min-pool-size**>40</**min-pool-size**>  
 <**max-pool-size**>40</**max-pool-size**>  
 <**prefill**>true</**prefill**>  
 </**pool**>  
 <**security**>  
 <**user-name**>postgres</**user-name**>  
 <**password**>password</**password**>  
 </**security**>  
 <**statement**>  
 <**prepared-statement-cache-size**>200</**prepared-statement-cache-size**>  
 <**share-prepared-statements**>true</**share-prepared-statements**>  
 </**statement**>

<**timeout**>  
 <**use-try-lock**>60</**use-try-lock**>

</**timeout**>

</**datasource**>

Минимальный и максимальный размер пула установить равным количеству ядер (если 32 ядра, то 32, если 1000, то 1000) **сервера СУБД**. Позже нужно провести эксперименты, увеличив размер пула вдвое, вчетверо и т.д. Это может привести к улучшению производительности, в зависимости от накладных расходов (сетевых задержек, задержек жёсткого диска и т.п.).

**prepared-statement-cache-size** – вещь, потенциально кушающая много памяти, поэтому с ней нужно обращаться аккуратно. С параметром нужно поэкспериментировать. Общее количество памяти, требующееся для этого кэша, по приблизительным прикидкам равно

**pstm\_cache\_mem ~ 2 \* avg\_query\_size \* pool\_size \* cache\_size**.

То есть при размере пула в 1000 соединений, кэша 100 и среднем размере запроса в 1Кб (~500-1000 символов), память под этот кэш составит ~200Мб. Чем больше размер кэша, тем лучше, но даже если позволит память, я бы его ограничил 10000 (при размере пула 1000, это 20 Гб оперативной памяти) или 20% памяти, выделенной Java-машине JBoss:

**pstm\_cache\_size ~ min(10000, java\_mem / (10000 \* pool\_size)**,

где java\_mem – в байтах.

Например, если Java-машине выделено 16 гигабайт, а размер пула 1000, то размер кэша выбиарем так:

pstm\_cache ~ min(10000, 17179869184 / (10000 \* 1000)) ~ min(10000, 1718) ~ **1700**

**timeout/use-try-lock** – время ожидания соединения из пула, если все заняты в секундах. Если соединение так и не удалось получить за данное время, будет выброшено исключение и транзакция откатится. Значение лучше установить меньшим или равным таймауту транзакции.

1. Настроить пул EJB (внутри настройки подсистемы EJB - <subsystem xmlns="urn:jboss:domain:ejb3…):

<**pools**>  
 <**bean-instance-pools**>  
 <**strict-max-pool name="slsb-strict-max-pool" max-pool-size="20" instance-acquisition-timeout="5" instance-acquisition-timeout-unit="MINUTES"**/>  
 <**strict-max-pool name="mdb-strict-max-pool" max-pool-size="20" instance-acquisition-timeout="5" instance-acquisition-timeout-unit="MINUTES"**/>  
 </**bean-instance-pools**>  
</**pools**>

Рекомендуемое значение max-pool-size для slsb-strict-max-pool – минимум 100, но лучше 10000, если позволяет память. Замечено, что иногда и нерегулярно от очень больших значений JBoss сходит с ума и отказывается разворачивать приложение, в таких случаях размер лучше снизить до приемлемого уровня. Закономерность неизвестна.

Для mdb-string-max-pool – 5-10 достаточно.

1. Настроить пул потоков для асинхронных EJB (внутри настройки подсистемы EJB - <subsystem xmlns="urn:jboss:domain:ejb3…):

<**thread-pools**>  
 <**thread-pool name="default"**>  
 <**max-threads count="200"**/>  
 <**keepalive-time time="100" unit="milliseconds"**/>  
 </**thread-pool**>  
</**thread-pools**>

Максимальное количество потоков установить равным количеству ядер, но не меньше 50.

1. Настроить пул потоков для веб-котейнера:

<**subsystem xmlns="urn:jboss:domain:threads:1.1"**>  
 <**unbounded-queue-thread-pool name="JBossWeb"**>  
 <**max-threads count="1000"**/>  
 <**keepalive-time time="75" unit="minutes"**/>  
 </**unbounded-queue-thread-pool**>  
</**subsystem**>

Количество потоков установить равным количеству ядер, но не меньше 50.

1. Настроить веб-контейнер:

<**subsystem xmlns="urn:jboss:domain:web:1.1" default-virtual-server="default-host" native="true"**>  
 <**connector name="http" protocol="HTTP/1.1" scheme="http" socket-binding="http" enable-lookups="false" executor="JBossWeb" max-connections="3260"**/>

Важно, чтобы **native** было установлено в true, а **executor** должен указывать на пул потоков из предыдущего пункта. **max-connections** установить немного б***о***льшим, чем максимальное число пользователей.

# Настройка Postgres

Свободную память для Postgres считаем так:

PGMem = (0.9 \* FreeMem – JBossMem).

При выделении JBoss 40% свободной памяти, это получается 50% свободной памяти после старта операционной системы.

Необходимо установить следующие значения параметров (после этого перезапустить Postgres):

1. **checkpoint\_segments**

Установить в 32.

1. **checkpoint\_completion\_target**

Установить в 0.8

1. **shared\_buffers**

Установить значение в 1/4 \* PGMem. «Плоское» значение в файле измеряется в страницах по 8Кб. Такими образом, чтобы трансформировать значение в гигабайтах, нужно их умножить на 131072, а значение в килобайтах – разделить на 8.

PGAdmin позволяет указать читабельное значение, самостоятельно преобразуя его.

1. **max\_connections**

Должно быть б**о**льшим, чем число соединений в пуле (скажем, на 20%). Например, если число соединений в пуле 1000, то этот параметр следует установить в 1200.

1. **effective\_io\_concurrency**

*Под Windows и Solaris не работает. Возможно, под какими-то другими ОС тоже.* Установить в 10 *и убедиться, что СУБД стартует после этого; если нет – закомментировать обратно.*

1. **effective\_cache\_size**

Установить в 1/2 \* PGMem. «Плоское» значение в файле измеряется в страницах по 8Кб. Такими образом, чтобы трансформировать значение в гигабайтах, нужно их умножить на 131072, а значение в килобайтах – разделить на 8.

PGAdmin позволяет указать читабельное значение, самостоятельно преобразуя.

1. **maintenance\_work\_mem**

Установить в 0.05 \* PGMem.

Размерность параметра в «плоском» файле – килобайты, PGAdmin позволяет задавать значение в удобной размерности.

1. **work\_mem**

Параметр, к которому нужно отнестись очень внимательно или не настраивать вовсе. Неплохое приближение рассчитывается так:

**work\_mem = 1/8 \* PGMem / max\_connections**

Размерность параметра в «плоском» файле – килобайты, PGAdmin позволяет задавать значение в удобной размерности.

1. **default\_statistics\_target**

Установить в 10000.

Этот параметр регулирует количество «корзин» при сборе статистики. Чем их больше, тем точнее статистика и тем более эффективные планы запросов, Postgre может построить. При увеличении, увеличивается время сбора статистики, но это не критично, так операция сбор статистики производится раз-два (максимум) в день и при этом не тормозит работу базы.

Параметры 10 – 14 призваны «заставить» Postgre более уважительно относиться к использованию индексов планировщиком запросов.

1. **seq\_page\_cost**

Установить в 1.0

Относительная стоимость последовательного сканирования.

1. **random\_page\_cost**

Установить в 1.0 на обычных жёстких дисках и 0.5 – на SSD.

Относительная стоимость произвольного доступа.

1. **cpu\_tuple\_cost**

Установить в 0.01

Относительная стоимость обработки записи процессором.

1. **cpu\_index\_tuple\_cost**

Установить в 0.0005

Относительная стоимость обработки записи процессором.

1. **cpu\_operator\_cost**

Установить в 0.0002

Относительная стоимость обработки оператора или функции.

# Настройка Linux

В ОС Linux необходима настройка максимального количества процессов, работающих с файлами. Минимальное количество необходимо рассчитать, исходя из числа пользователей, так как они, не производя непосредственных операций сохранения могут загружать файлы как минимум во временное файловое хранилище и скачивать из него по HTTP.

В качестве первого приближения можно указать количество конечных пользователей плюс 20% (если 20% меньше 50, то хотя бы 50) для различных системных нужд. При необходимости – увеличить.

**САМИ НАСТРОЙКИ ВПИСАТЬ ЗДЕСЬ.**