**BTP使用过程中经验小结**

在实施BTP产品过程中，经常会碰到一些莫名其妙的问题，比如端口冲突、性能测试时效果不佳等问题，在此把实施过程中的一些解决方案总结下来，以备借鉴。

1. **在启动BTP应用时，报1000端口冲突**

在启动BTP时，报1000端口已经被占用，启动失败

* 原因分析

BTP产品为支持远程路由功能，会在启动时默认启动1000端口的Jetty服务，如果在同一机器内启动两个BTP应用，则会发生端口占用冲突的异常。

* 解决方案

1. 通过系统参数-Dbtp.remote.server.port进行指定，如果多个BTP应用可以通过-D{appName}.btp.remote.server.port按应用名称进行指定，其中{appName}为对应的BTP应用名称。
2. 通过修改配置文件，停止远程路由功能的Jetty服务启动，把com.primeton.btp.remote.jar 中的META-INF/btp/Module.btp.xml中的

|  |
| --- |
| <module name="com.primeton.btp.remote" activator="com.primeton.btp.impl.remote.Activator" dependency-modules="com.primeton.btp.engine.trans,com.primeton.btp.endpoint"/> |

注释掉即可。

1. **在启动BTP应用时，报6200端口冲突**

在启动BTP时，报6200端口已经被占用，启动失败

* 原因分析

BTP产品在启动时会启动为支持JMX协议访问的MBean Server，其默认端口为6200，如果在同一机器内启动两个BTP应用，则会发生端口占用冲突的异常。

* 解决方案

修改MBean Server端口的配置文件，其路径为{appConfig}/ startup.conf，修改其中的项

|  |
| --- |
| AdminPort=6200 |

即可。

1. **性能测试时，效果不佳，数据库中上下文记录的压力特别大**

在使用BTP在做交易的关于接入、接出的性能测试时，发现交易的执行速度比较慢，TPS比较低，并且数据库的IO非常大，在招行POC做养老金的批量导入时发现导入速度非常缓慢，每秒才几十笔。

* 原因分析

在BTP中，为了支持交易以及主机服务的自动冲正，在交易以及主机服务执行时会把当前交易的上下文记录到冲正相关的数据表中，以备在执行出现错误或者主机服务执行超时时自动进行数据的冲正动作。在大并发时，因记录的上下文都以BLOB的方式存储，将会导致性能的急剧下降，同时对数据库的IO也带来极大的挑战。

* 解决方案

1. 关闭交易执行时记录开关

目前，当交易服务定义为全局事务则会开启上下文的记录，但系统中还有一隐含的条件就是当交易服务为无事务时但配置了记录系统日志时，也会开启上下文的记录，因此在性能测试时最好不要配置全局事务，同时也不要选择记录系统日志的开关。

1. 关闭主机服务执行时记录开关

在目前实现中，为了支持主机服务调用超时后的自动冲正动作，系统在调用主机服务时自动记录其上下文，关闭仅能通过修改配置文件的方式进行。把com.primeton.btp.engine.hosttrans.jar 中的META-INF/services目录下的com.primeton.btp.api.engine.hosttrans.handler.IOutboundHandler文件中的

|  |
| --- |
| com.primeton.btp.impl.engine.hosttrans.handler.SystemLogOutboundHandler |

注释掉即可。

|  |
| --- |
| 在调用主机服务时，由于添加了上下文记录导致POC养老金批量导入场景时，速度缓慢，会有几个数量级的性能损耗。 |

1. **Trace日志中关于事件的记录非常频繁**

在trace.log中发现有如下日志

[@sql][select SERIAL\_NO from BTP\_EVENT\_TRACER where (RESULT = 0)]

其执行的频率非常频繁，一秒钟要执行10次之多

* 原因分析

此语句为查询发送失败的事件信息，在实现中启动了一线程，每隔一定时间从数据库中轮询出，然后再次发送。但系统中的时间间隔默认配置的为100ms，导致了频率过快。

* 解决方案

从目前的实现看，代码中没有提供配置项对此轮询间隔进行设置，只能通过修改代码的方式进行提供，在上海银行项目组已提供了修改补丁。

1. **事件处理命名Sql加载失败**

在启动BTP时，报加载事件处理的命名Sql定义文件失败

* 原因分析

在BTP中事件的保存、查找等数据处理都是通过命名Sql进行操作的，此命名Sql的定义文件与构件包中的命名Sql不同，其定义文件位于事件处理的Jar（\_btp\_eventSubscriber.namingsql、\_btp\_eventTracer.namingsql）中，在启动BTP时会从jar中首先把此文件复制到临时目录中，然后再从临时目录中进行加载，临时目录的位置位于｛java.io.tmpdir｝/temp/namingsql下，如果临时目录没有权限，或者系统中通过不同用户安装了多套应用，则会发生因没有权限导致复制失败的异常。

* 解决方案

把目录｛java.io.tmpdir｝/temp/namingsql赋予对于所有用户都可以读写的权限即可。

1. **性能测试时系统的调优参数**

在通过BTP产品做性能测试时，由于BTP采用了分段的处理模式，因此各段的参数调整对性能的影响将是非常巨大的。在此总结与性能有关的参数配置如下：

1. 与对象池相关的参数

此参数可以配置在渠道和通道上，其功能类似于挡板，可以做流量控制，其说明了能同时接收请求处理的并发个数限制，在配置时尽量大于其请求的并发个数，其配置方式如下：

* 渠道

|  |
| --- |
| <transport-request-handler>  <min-size>50</min-size>  <max-size>50</max-size>  <increase-size>1</increase-size>  <keep-alive-time>60</keep-alive-time>  <checkout-timeout>5</checkout-timeout>  <factory-class></factory-class>  </transport-request-handler> |

* 通道：

|  |
| --- |
| <endpoint-caller>  <min-size>50</min-size>  <max-size>50</max-size>  <increase-size>1</increase-size>  <keep-alive-time>60</keep-alive-time>  <checkout-timeout>5</checkout-timeout>  <factory-class> </factory-class>  </endpoint-caller> |

1. 与线程池相关的参数

在数据的接入、接出以及中间业务逻辑的运行都可能线程池，在TCP的渠道、TCP，Http的通道以及中间逻辑运行的各容器都会包含线程池的配置，其具体配置如下：

* TCP渠道

|  |
| --- |
| <work-threads>  <min-size>50</min-size>  <max-size>50</max-size>  <keep-alive-time>60</keep-alive-time>  <queue-size>0</queue-size> <rejected-policy>java.util.concurrent.ThreadPoolExecutor$AbortPolicy</rejected-policy>  </work-threads> |

* HTTP渠道，其配置在扩展属性中

|  |
| --- |
| <ext-property>  <entry description="" key="MIN\_THREADS" value="5"/>  <entry description="" key="MAX\_THREADS" value="20"/>  <entry description="" key="QUEUE\_SIZE" value="-1"/>  </ext-property> |

* TCP通道、HTTP通道

|  |
| --- |
| <work-threads>  <min-size>50</min-size>  <max-size>50</max-size>  <keep-alive-time>60</keep-alive-time>  <queue-size>-1</queue-size> <rejected-policy>java.util.concurrent.ThreadPoolExecutor$AbortPolicy</rejected-policy>  </work-threads> |

* 各容器（交易、主机服务、事件服务）配置

其文件位于架构构件包目录META-INF\config下，包含多个容器的配置，其配置方式遵循相同的Schema定义，其格式如下：

|  |
| --- |
| <trans-server id="defaultTransServer">  <name>默认交易服务器</name>  <description>这是一个默认的交易服务器</description>  <timeout>0</timeout>  <work-threads>  <min-size>5</min-size>  <max-size>20</max-size>  <keep-alive-time>100</keep-alive-time>  <queue-size>1000</queue-size> <rejected-policy>java.util.concurrent.ThreadPoolExecutor$AbortPolicy</rejected-policy>  </work-threads>  <work-threads-ref/>  <ext-property>  <entry description="" value="testValue" key="test"/>  </ext-property>  </trans-server> |