**Tomcat**  **Guide**



**■ Table of Contents**

**1. Tomcat configuration**

1.1 설치 디렉토리 구성

**2. Tomcat 운영관리**

2.1 Tomcat 기동

2.2 Tomcat 정지

2.3 Tomcat log 설정

2.3.1 Tomcat Access Log 디렉토리 경로 변경

2.3.2 Tomcat Catalina.yyyy-mm-dd.log 디렉토리 경로 변경

2.3.3 Tomcat Catalina.out 디렉토리 경로 변경

2.4 Application deploy

2.5 JNDI를 이용한 DBCP DB Pool 설정 (Oracle, mysql, altibase)

2.5.1 Oracle

2.5.2 mysql

2.5.3 Altibase

2.6 Clustering 설정

2.7 서비스 점검을 위한 프로세스 및 포트 체크

2.8 포트 변경

**3. Tomcat 보안설정**

* 1. Tomcat 데몬관리

3.2 디렉토리 권한관리

3.3 소스/설정 파일 권한관리

3.4디렉토리 쓰기 권한관리

* 1. 로깅 디렉토리/파일 권한관리
  2. 에러페이지 관리
  3. HTTP Method 제한
  4. 인코딩 설정

**4. 운영체제 환경설정**

5.1 커널파라미터

5.2 적용한 커널파라미터 값

5.3 사용자 limit 값

**5. monitoring & troubleshooting**

5.1 장애 발생시 진단 및 조치

5.2 장애 발생시 대처

**1. Tomcat configuration**

1.1 설치 디렉토리 구성

|  |  |
| --- | --- |
| 디렉토리 구성 | |
| 설치 디렉토리 | /tomcat/tomcat8 |
| bin 디렉토리 | /tomcat/tomcat8/bin : Tomcat 실행 파일들이 위치한 디렉토리 |
| conf 디렉토리 | /tomcat/tomcat8/conf : Tomcat 설정 파일들이 위치한 디렉토리 |
| logs 디렉토리 | /tomcat/tomcat8/logs : Tomcat 로그 파일 기본 디렉토리 |
| lib 디렉토리 | /tomcat/tomcat8/lib : Apache에서 기본적으로 사용하는 모듈들이 위치한 디렉토리 |

**2. Tomcat 운영관리**

2.1 Tomcat 기동

# pwd

/app/was/tomcat8/bin

#./startup.sh

Using CATALINA\_BASE: /app/was/tomcat8

Using CATALINA\_HOME: /app/was/tomcat8

Using CATALINA\_TMPDIR: /app/was/tomcat8/temp

Using JRE\_HOME: /usr

Using CLASSPATH: /app/was/tomcat8/bin/bootstrap.jar:/app/was/tomcat8/bin/tomcat-juli.jar

Using CATALINA\_OPTS:

Tomcat started.

2.2 Tomcat 정지

# ./shutdown.sh

Using CATALINA\_BASE: /app/was/tomcat8

Using CATALINA\_HOME: /app/was/tomcat8

Using CATALINA\_TMPDIR: /app/was/tomcat8/temp

Using JRE\_HOME: /usr

Using CLASSPATH: /app/was/tomcat8/bin/bootstrap.jar:/app/was/tomcat8/bin/tomcat-juli.jar

Using CATALINA\_OPTS:

2.3 Tomcat log 설정

톰캣 로그 종류

(1). catalina.out : 서버상에서 발생한 모든 내용(서버의 기동,정지,개시 등)을 기록한 파일이다.

그렇기 때문에 생기는 로그의 양이 방대하다.

날짜별, 혹은 시간별로 로그를 분리하는것은 일반적으로 catalina.out에 사용한다.

(2). catalina.yyyy-mm-dd.log : 톰캣에서 생기는 로그만을 기록한다.

Standard output(표준 스트림), Standard error(표준에러)의 로깅은 제외

(3). host-manager.log : Tomcat Host Manager Web app 로그 (가상호스트 매니저)

(4). manager.log : Tomcat Manager Web App 로그 (웹콘솔)

(5). localhost.log : host(특정 가상호스트 대상)한정 로그

2.3.1 Tomcat Access Log 디렉토리 경로 변경

#vi /app/was/tomcat8/conf/server.xml

수정 전 : default 설정

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

prefix="localhost\_access\_log." suffix=".txt"

pattern="%h %l %u %t &quot;%r&quot; %s %b" />

수정 후: /data/tomcat\_logs로 경로 설정

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="/data/tomcat\_logs"

prefix="localhost\_access\_log." suffix=".txt"

pattern="%h %l %u %t &quot;%r&quot; %s %b" />

2.3.2 Tomcat Catalina.yyyy-mm-dd.log 디렉토리 경로 변경

#vi /app/was/tomcat8/conf/logging.properties

# 수정 전 ${catalina.base}/logs

1catalina.org.apache.juli.AsyncFileHandler.level = FINE

1catalina.org.apache.juli.AsyncFileHandler.directory = ${catalina.base}/logs

1catalina.org.apache.juli.AsyncFileHandler.prefix = catalina.

1catalina.org.apache.juli.AsyncFileHandler.encoding = UTF-8

# 수정 후: /data/tomcat\_logs

1catalina.org.apache.juli.AsyncFileHandler.level = FINE

1catalina.org.apache.juli.AsyncFileHandler.directory = /data/tomcat\_logs

1catalina.org.apache.juli.AsyncFileHandler.prefix = catalina.

1catalina.org.apache.juli.AsyncFileHandler.encoding = UTF-8

2.3.3 Tomcat Catalina.out 디렉토리 경로 변경

#vi /app/was/tomcat8/bin/catalina.sh

# 수정 전 ${catalina.base}/logs

228 if [ -z "$CATALINA\_OUT" ] ; then

229 CATALINA\_OUT="$CATALINA\_BASE"/logs/catalina.out

# 수정 후: /data/tomcat\_logs

228 if [ -z "$CATALINA\_OUT" ] ; then

229 CATALINA\_OUT=/data/tomcat\_logs/catalina.out

2.4 Application deploy

간단한 index.jsp 를 가지고 /app 라는 폴더를 자동배포가 가능한 “webapps” 폴더로 이동한다.

# pwd

/app/prod/app

# ls

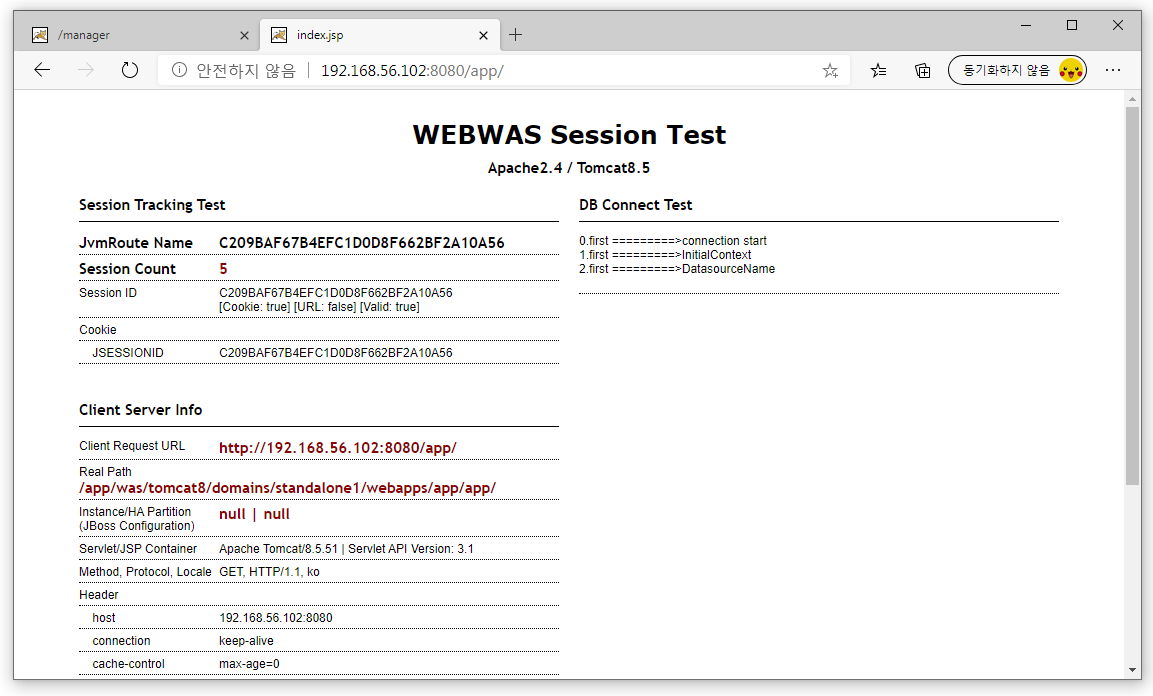
META-INF WEB-INF index.jsp

# cd ..

# cp -R app /app/was/tomcat8/webapps/

폴더명이 /app 이므로 기본적으로 context도 /app 로 지정된다.

디플로이 된app 서비스 페이지 호출 : <http://serviceIP:8080/app/>



2.5 JNDI를 이용한 DBCP DB Pool 설정 (Oracle, mysql, altibase)

비교적 리소스를 많이 소요되는 데이터베이스와의 연결은 풀형태로 관리하여 데이터베이스 연결에 대한 부하를 최소화하여 안정적으로 운영하는데 도움이 된다.

2.5.1 Oracle

Oracle(dirverClassName, url, username, password 등의 정보를 수정해야 합니다.)

#vi /app/was/tomcat8/conf/server.xml

<GlobalNamingResource> </GlobalNamingResource>

# 태그 안에 아래의 내용을 추가합니다.

<Resource auth="Container" driverClassName="oracle.jdbc.OracleDriver" initialSize="10" loginTimeout="10" maxActive="40" maxIdle="-1" maxWait="-1" name="jdbc/oraDS" password="passwod" username="username"

testOnBorrow="true"

type="javax.sql.DataSource”

poolPreparedStatements="true"

testWhileIdle="true"

maxOpenPreparedStatements="10"

removeAbandoned="true"

removeAbandonedTimeout="60"

logAbandoned="true"

url="jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS\_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=XXX.XXX.XXX.XXX)(PORT=1521))(ADDRESS=(PROTOCOL=TCP)(HOST=XXX.XXX.XXX.XXX)(PORT=1521))(FAILOVER=on)(LOAD\_BALANCE=off))(CONNECT\_DATA=(SERVER=DEDICATED)(SERVICE\_NAME=GTP)))"

validationQuery="SELECT 1 FROM DUAL" />

<Context> </Context>

# 태그 안에 아래의 내용을 추가합니다.

<ResourceLink name="jdbc/oraDS" global="jdbc/oraDS " type="javax.sql.DataSource" />

2.5.2 mysql

MySQL(driverClassName, url, username, password 등의 정보를 수정해야 합니다.)

#vi /app/was/tomcat8/conf/server.xml

<GlobalNamingResource> </GlobalNamingResource>

# 태그 안에 아래의 내용을 추가합니다.

<Resource name="jdbc/mysql" auth="Container" type="javax.sql.DataSource"

driverClassName="com.mysql.jdbc.Driver"

url="jdbc:mysql://XXX.XXX.XXX.XXX:3306/mysql"

username="user" password="password"

initialSize="20"

maxActive="100"

maxIdle="-1"

maxWait="-1"

validationQuery="SELECT 1 "

testOnBorrow="true"

poolPreparedStatements="true"

maxOpenPreparedStatements="10"

removeAbandoned="true" removeAbandonedTimeout="60" logAbandoned="true" />

<Context> </Context>

# 태그 안에 아래의 내용을 추가합니다.

<ResourceLink name="jdbc/mysql" global="jdbc/mysql " type="javax.sql.DataSource" />

2.5.3 Altibase

Altibase(driverClassName, url, username, password 등의 정보를 수정해야 합니다.)

#vi /app/was/tomcat8/conf/server.xml

<GlobalNamingResource> </GlobalNamingResource>

# 태그 안에 아래의 내용을 추가합니다.

<Resource name="jdbc/altibase" auth="Container" type="javax.sql.DataSource"

driverClassName="Altibase.jdbc.driver.AltibaseDriver" url="jdbc:Altibase:// XXX.XXX.XXX.XXX:20300/hmmdb?AlternateServers=(XXX.XXX.XXX.XXX:20300)&amp;ConnectionRetryCount=3&amp;ConnectionRetryDelay=1&amp;SessionFailOver=off&amp;LoadBalance=off"" username="user" password="password"

initialSize="20"

maxActive="100"

maxIdle="-1"

maxWait="-1"

validationQuery="SELECT 1 FROM DUAL"

testOnBorrow="true"

poolPreparedStatements="true"

maxOpenPreparedStatements="10"

removeAbandoned="true" removeAbandonedTimeout="60" logAbandoned="true" />

<Context> </Context>

# 태그 안에 다음 내용을 추가합니다.

<ResourceLink name="jdbc/altibase" global="jdbc/btvtoon" type="javax.sql.DataSource" />

2.6 Clustering 설정

톰캣에서 세션을 공유하는 방식은 아래 2가지 방법이 있다.

DeltaManager : All-to-All 방식 Session Replication 기능 구현

BackupManager : Primary-Secondary Session Replication 기능을 구현

Clustering 기능을 사용하려면 Server.xml 에 아래의 설정부분 추가한다.

UDP/Multicast 방식

<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"

channelSendOptions="6">

<Manager className="org.apache.catalina.ha.session.BackupManager"

expireSessionsOnShutdown="false"

notifyListenersOnReplication="true"

mapSendOptions="6"/>

<!--

<Manager className="org.apache.catalina.ha.session.DeltaManager"

expireSessionsOnShutdown="false"

notifyListenersOnReplication="true"/>

-->

<Channel className="org.apache.catalina.tribes.group.GroupChannel">

<Membership className="org.apache.catalina.tribes.membership.McastService"

address="228.0.0.4"

port="45564"

frequency="500"

dropTime="3000"/>

<Receiver className="org.apache.catalina.tribes.transport.nio.NioReceiver"

address="auto"

port="5000"

selectorTimeout="100"

maxThreads="6"/>

<Sender className="org.apache.catalina.tribes.transport.ReplicationTransmitter">

<Transport className="org.apache.catalina.tribes.transport.nio.PooledParallelSender"/>

</Sender>

<Interceptor className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"/>

<Interceptor className="org.apache.catalina.tribes.group.interceptors.MessageDispatch15Interceptor"/>

<Interceptor className="org.apache.catalina.tribes.group.interceptors.ThroughputInterceptor"/>

</Channel>

<Valve className="org.apache.catalina.ha.tcp.ReplicationValve"

filter=".\*\.gif|.\*\.js|.\*\.jpeg|.\*\.jpg|.\*\.png|.\*\.htm|.\*\.html|.\*\.css|.\*\.txt"/>

<Deployer className="org.apache.catalina.ha.deploy.FarmWarDeployer"

tempDir="/tmp/war-temp/"

deployDir="/tmp/war-deploy/"

watchDir="/tmp/war-listen/"

watchEnabled="false"/>

<ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener"/>

</Cluster>

TCP 방식

<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"

channelSendOptions="8"

channelStartOptions="3">

<Manager className="org.apache.catalina.ha.session.DeltaManager"

expireSessionsOnShutdown="false"

notifyListenersOnReplication="true"/>

<Channel className="org.apache.catalina.tribes.group.GroupChannel">

<!-- 클러스터 전송 -->

<Sender className="org.apache.catalina.tribes.transport.ReplicationTransmitter">

<Transport className="org.apache.catalina.tribes.transport.nio.PooledParallelSender" />

</Sender>

<!-- 수신 설정 -->

<Receiver className="org.apache.catalina.tribes.transport.nio.NioReceiver"

address="auto"

port="4055"

autoBind="0"

selectorTimeout="5000"

maxThreads="6"/>

<!-- PING 체크 여부

staticOnly: 고정멤버

-->

<Interceptor className="org.apache.catalina.tribes.group.interceptors.TcpPingInterceptor" staticOnly="true"/>

<Interceptor className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"/>

<Interceptor className="org.apache.catalina.tribes.group.interceptors.StaticMembershipInterceptor">

<Member className="org.apache.catalina.tribes.membership.StaticMember"

host="192.168.0.50"

port="4055"

uniqueId="{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1}" />

</Interceptor>

<Interceptor className="org.apache.catalina.tribes.group.interceptors.MessageDispatchInterceptor"/>

</Channel>

<!-- 데이터 복제 시기 설정

filter: 필터에 속하는 자원의 경우 세션 변경 여부 공유 불필요

-->

<Valve className="org.apache.catalina.ha.tcp.ReplicationValve"

filter="" />

<!-- 동일 서버로 요청이 들어오도록 설정 -->

<Valve className="org.apache.catalina.ha.session.JvmRouteBinderValve"/>

<!-- 메시지 수신 -->

<ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener"/>

</Cluster>

2.7 서비스 점검을 위한 프로세스 및 포트 체크

Tomcat 설정 변경 후 재기동 또는 서비스 이상이 있는 경우 다양한 방법으로 와스서버 상태를 체크 할 수 있다. 주로 사용하는 방법은 프로세스 및 HTTP 포트 확인을 통해 서비스 기동 유무를 확인하며 카탈리나 로그를 통해 서비스 이상상태를 확인한다.

# ps -ef | grep java

# netstat -na | grep 8080

2.8 포트 변경

server.xml에서 http 8080 포트, AJP 8009 포트를 변경해줄 수 있다.

# vi server.xml

<Connector port="8080" protocol="HTTP/1.1"

<Connector protocol="AJP/1.3"

address="0.0.0.0"

port="8009"

secretRequired="false"

redirectPort="8443" />

**3. Tomcat 보안설정**

3.1 Tomcat 데몬관리

- Tomcat 서버 데몬이 root 권한으로 운영되지 않도록 관리해야 함.

WAS 서버 데몬이 root 권한으로 운영될 경우 WAS Application의 취약점이나 Buffer Overflow 시 공격자에게 root권한을 유출할 수 있음.

start script를 기동시킬 때 사용되는 계정의 권한으로 서버 데몬이 운영 (tomcat등 데몬 기동을 위한 계정을 별도로 관리)

3.2 디렉토리 권한관리

- 웹 사이트 변조를 예방함.

일반 사용자가 웹 서버 홈 디렉토리에 임의의 파일을 생성, 삭제, 변경 할 수 있으면, 홈페이지 변조, 작업 실수로 인한 파일 삭제, 백도어 삽입 등의 피해가 발생할 수 있음

# vi /app/was/tomcat8/conf/server.xml

….

<Host name="localhost" appBase="webapps"

unpackWARs="true" autoDeploy="true"

xmlValidation="false" xmlNamespaceAware="false">

3.3 소스/설정 파일 권한관리

- 비인가 사용자에 의한 소스 변경을 예방함.

일반 사용자가 웹 서버의 설정 파일을 삭제, 변경할 수 있으면, 시스템이 오작동하여 사용 불능 상태에 빠질 우려가 있음

3.4 디렉토리 쓰기 권한관리

디렉토리 검색 기능(Directory Indexing)이 설정되어 있는 경우, Web서버 구조 노출 및 설치 파일의 유출 가능성이 있음.

# vi /app/was/tomcat8/conf/web.xml

<servlet>

<servlet-name>default</servlet-name>

<servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>

<init-param>

<param-name>debug</param-name>

<param-value>0</param-value>

</init-param>

<init-param>

<param-name>listings</param-name>

<param-value>false</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

3.5 로깅 디렉토리/파일 권한관리

- 시스템 정보 노출을 방지함.

로그 파일에는 공격자에게 유용한 정보가 들어있을 수 있으므로 권한 관리가 필요함. 일반 사용자에 의한 정보 유출이 불가능 하도록 권한 설정을 강화함.

3.6 에러페이지 관리

공격자가 대상 시스템의 정보를 획득하기 위해 고의적으로 다양한 에러를 유발하여 돌아오는 에러 페이지를 통해 프로그램의 구조 및 환경 설정을 추정할 수 있고 별도의 에러 페이지를 생성하여 불필요한 정보의 유출을 막음.

설정파일에서 에러 페이지 설정 확인 (필수 설정 : 400, 401, 403, 404, 500)

# vi /app/was/tomcat8/conf/web.xml

…

<welcome-file-list>

<welcome-file>index.html</welcome-file>

<welcome-file>index.htm</welcome-file>

</welcome-file-list>

<error-page>

<error-code>404</error-code>

<location>에러페이지명.jsp</location>

</error-page>

3.7 HTTP Method 제한

- 웹 사이트 변조를 예방함.

OPTIONS, GET, POST 이외의 다른 HTTP Method 를 지원하는 경우, 악의적인 공격자가 임의의 파일을 삭제하거나 업로드하여 서버의 정상 운영에 지장을 줄 수 있음.

- HTTP Method 를 일부 항목으로 제한

Tomcat 에서는 GET, POST, PUT, DELETE, OPTIONS, TRACE 등의 Method 를 지원하는데, 이 Method 들은 WebDAV 나 telnet 을 이용해 해당 Method 를 요청하는 경우 서버에 임의의 파일 생성 혹은 삭제

# vi /app/was/tomcat8/conf/web.xml

…

<security-constraint>

<display-name>Forbidden</display-name>

<web-resource-collection>

<web-resource-name>Forbidden</web-resource-name>

<url-pattern>/\*</url-pattern>

<http-method>PUT</http-method>

<http-method>DELETE</http-method>

<http-method>OPTIONS</http-method>

</web-resource-collection>

<auth-constraint>

<role-name></role-name>

</auth-constraint>

</security-constraint>

3.8 인코딩 설정

웹 서브시스템의 URI 인코딩이나 Java의 파일 인코딩 등 애플리케이션에서 사용하는 인코딩을 설정하여야 한다. 기본적으로 애플리케이션에서 사용할 UTF-8 인코딩을 위한 설정들이 되어 있다. 이외에도 Java 시스템 프로퍼티로 사용할 값을 추가하여도 된다.

<Connector executor="tomcatThreadPool" port="8080" protocol="HTTP/1.1"

maxPostSize="-1" maxKeepAliveRequests="2000" connectionTimeout="30000"

enableLookups="false" acceptCount="2000"

redirectPort="8443" address="192.168.56.102"

URIEncoding="UTF-8" />

<!-- A "Connector" using the shared thread pool-->

**4. 운영체제 환경 설정**

4.1 커널파라미터

웹 서버와 웹 기반 미들웨어 서버는 모두 네트워크를 통해 서비스를 제공하는 시스템이다. 네트워크를 통해 데이터를 전달하기 때문에, 운영체제의 TCP/IP에 대한 튜닝은 필수적이다. 아래 표에서 설명한 핵심적인 파라미터를 적용하는 것이 좋다. 특히 TCP의 수신, 송신 버퍼의 크기는 운영체제가 기본적으로 제공하는 것보다 크게 설정해야 서버의 성능을 향상할 수 있다. 다음 설정을 웹 서버와 JBoss 운영 서버에 대해 모두 적용한다.

|  |  |  |
| --- | --- | --- |
| **파라미터** | **권장값** | **설명** |
| net.ipv4.tcp\_keepalive\_time | 30 | keep-alive 시간을 줄인다. |
| net.ipv4.tcp\_fin\_timeout | 10 | FIN 타임아웃 시간을 줄여 FD를 빨리 확보할 수 있도록 한다. |
| net.core.netdev\_max\_backlog | 2500 | 백로그에 들어오는 소켓 개수를 늘린다. |
| net.ipv4.tcp\_retries1 | 3 | TCP 연결에 문제가 있을 때 연결을 재시도하는 횟수(최솟값은 3이다) |
| net.ipv4.tcp\_retries2 | 3 | TCP 연결을 끊기 전에 재시도하는 횟수를 줄인다. |
| net.ipv4.ip\_local\_port\_range | 1024 65000 | 사용할 수 있는 로컬 포트 범위를 늘린다. |
| net.core.rmem\_max | 56777216 | TCP 수신 버퍼크기 최댓값을 늘린다. |
| net.core.rmem\_default | 16777216 | TCP 수신 버퍼크기 기본값을 늘린다. |
| net.core.wmem\_max | 56777216 | TCP 전송 버퍼크기 최댓값을 늘린다. |
| net.core.wmem\_default | 16777216 | TCP 수신 버퍼크기 기본값을 늘린다. |
| net.ipv4.tcp\_window\_scaling | 1 | 65kb 이상의 큰 TCP 윈도우 스케일링을 사용한다. |
| net.ipv4.tcp\_orphan\_retries | 0 | 서버 측에서 닫은 TCP 연결을 끊기 전에 확인하는 횟수를 줄인다. 기본값은 7로 50초~16분 정도 걸린다. |
| net.ipv4.tcp\_sack | 0 | SYNC 패킷을 전송한 후 일부 ACK를 받지 못했을 경우 선택적으로 받지 못한 ACK 패킷을 받도록 설정할 수 있다. 0은 받지 않는 설정이다. 패킷 유실이 많은 네트워크에서는 1로 설정한다. |

4.2 적용한 커널파라미터 값

/etc/sysctl.conf

# Updates

net.ipv4.neigh.default.unres\_qlen=100

net.ipv4.tcp\_keepalive\_time = 30

net.ipv4.tcp\_fin\_timeout = 10

net.core.netdev\_max\_backlog = 2500

net.ipv4.tcp\_retries1 = 2

net.ipv4.tcp\_retries2 = 3

net.ipv4.ip\_local\_port\_range = 1024 65000

net.core.rmem\_max = 56777216

net.core.rmem\_default = 16777216

net.core.wmem\_max = 56777216

net.core.wmem\_default = 16777216

net.ipv4.tcp\_window\_scaling = 1

net.ipv4.tcp\_timestamps = 0

net.ipv4.tcp\_sack = 0

net.ipv4.tcp\_orphan\_retries = 0

5.3 사용자 limit 값

# /etc/security/limits.conf

#

#This file sets the resource limits for the users logged in via PAM.

#It does not affect resource limits of the system services.

#

#Each line describes a limit for a user in the form:

#

#<domain> <type> <item> <value>

#

#Where:

#<domain> can be:

# - an user name

# - a group name, with @group syntax

# - the wildcard \*, for default entry

# - the wildcard %, can be also used with %group syntax,

# for maxlogin limit

#

#<type> can have the two values:

# - "soft" for enforcing the soft limits

# - "hard" for enforcing hard limits

#

#<item> can be one of the following:

# - core - limits the core file size (KB)

# - data - max data size (KB)

# - fsize - maximum filesize (KB)

# - memlock - max locked-in-memory address space (KB)

# - nofile - max number of open files

# - rss - max resident set size (KB)

# - stack - max stack size (KB)

# - cpu - max CPU time (MIN)

# - nproc - max number of processes

# - as - address space limit (KB)

# - maxlogins - max number of logins for this user

# - maxsyslogins - max number of logins on the system

# - priority - the priority to run user process with

# - locks - max number of file locks the user can hold

# - sigpending - max number of pending signals

# - msgqueue - max memory used by POSIX message queues (bytes)

# - nice - max nice priority allowed to raise to values: [-20, 19]

# - rtprio - max realtime priority

#

#<domain> <type> <item> <value>

#

#\* soft core 0

#\* hard rss 10000

#@student hard nproc 20

#@faculty soft nproc 20

#@faculty hard nproc 50

#ftp hard nproc 0

#@student - maxlogins 4

**wasAdmin hard nofile 65536**

**wasAdmin soft nofile 65536**

**wasAdmin soft nproc 2047**

**wasAdmin hard nproc 16384**

# End of file

**5. monitoring & troubleshooting**

1. 장애 발생시 진단 및 조치

▣ Hang UP (Instance Run, Service Stop)

* WAS 및 플러그인 리소스문제 (튜닝: Thread 및 각종 Pool, Heap, JVM option)
* Data Source Resource 반환
* Commit / Rollback 명시 미비로 의한 DB Lock
* Backend / legacy 시스템문제
* Application Synchronize
* Resource 문제
* CPU (과도한 CPU 사용 = 무한루프 | 잦은 CPU 점유 | CPU 리소스 부족)
* Memory (과도한 메모리 사용, Leak 또는 부족, 잘못된 메모리 튜닝)
* Disk (파티션 풀 또는 부족)
* Slow Response (Response Time 급격히 낮아짐)

TimeZone (서버 | 스토리지 미스매치), duplex 문제 등

* Network

용량 또는 잘못된 구성 및 장비 제약 사항 (클러스터링, EC2)

2. 장애 발생시 대처

▣ 장애 발생 시점 데이터 수집

* 장애 현상에 대한 자세한 파악 및 확인 (주기, 특정기능)
* 장애 발생 전 작업 내역 확인
* CPU 상태, 메모리 상태, 네트워크 상태, 디스크 상태
* top, prstat, sar, free, vmstat, netstat, iostat
* WAS 프로세스의 상태
* native stack trace, java stack trace, heap dump, lsof(List open file)
* thread별 CPU 사용율
* WAS thread 동작 상태
* network connection 상태
* Web Server 프로세서의 상태
* native stack trace
* network connection 상태
* Web server 모니터링 데이터
* OS 로그, WAS로그, Web Server 로그, jk로그 (각 WAS의 plug-in log)

▣ Hang

* 전체 서비스가 hang에 걸린 것인지 특정 서비스만 hang에 걸린 것인지 확인
* 웹 서버와 연동했다면 직접 WAS로 접근했을 때도 hang이 발생하는지 확인
* 웹 서버에 접속된 연결 수 확인

- netstat –n | grep EST | grep ‘:80’ | wc -l

* WAS에 접속된 연결 수 확인

- netstat –n | grep EST | grep ‘:8009’ | wc –l

* Database로 나가는 연결 수 확인

- netstat –n | grep EST | grep ‘:1521’ | wc –l

* Web Server / WAS native stack trace

- pstack <pid>

* WAS java stack trace

- kill -3 <pid> or jstack –m <pid>

