클라우드 환경에서의 고가용성 시스템 구현 및 결함 내성 시험

고석주[1] · 손종영[*1] · 김선희[1] · 조현경[1] · 배근령[1] · 지호준[2]

[1]경북대학교 · [2](주)범일정보

Implementation and Fault-tolerance Tests of High Availability System in Cloud Environment

Seok-ju Koh[1] · Jong-young Son[*1] · Sun-hee Kim[1] · Hyun-kyung Cho[1] · Keun-ryeong Bae[1]· Ho-jun Ji[2]

[1]KyungPook National University · [2]Bumil Information Corp

E-mail : sjkoh@knu.ac.kr / sondahum@gmail.com / enter2558@gmail.com / ruddl_s2@naver.com / ssnvan@naver.com / jjunaon@gmail.com

요 약

많은 기업체와 공공기관 들이 자체적으로 서버 안정화를 위한 시스템을 구축하고 있지만 아직까지 일부 공공기관을 포함한 다수의 웹 서버에서는 트래픽 집중 등의 예상치 못한 원인으로 인한 장애가 빈번하게 일어난다. 이런 장애시마다 소실되는 정보들과 업무의 마비는 큰 손실을 가져올 수 있다. 본 연구에서는 다중화 구조의 웹 서버에서 로드밸런싱을 적용하여 각 서버로 들어오는 부하를 분산해 과부하로 인한 장애를 대비하고, 세션 클러스터링을 통해 일부 서버의 장애시에도 사용자의 세션정보를 유지하여 연속적인 서비스를 제공하는 고가용성 웹 시스템을 제안한다.

ABSTRACT

Many enterprises and public institutions are building their own server stabilization system. Nevertheless, many web servers including some public institutions have frequent obstacles due to the unexpected causes such as traffic concentration. Data loss during each of these situations and business interruption can cause a huge loss. In this study, we propose high-availability Web system that uses load balancing and session clustering. Each deals with the failure due to the over traffic by distributing incoming requests in the multiplexed web server and provides the continuous services by maintaining session information of the users in case of the failure of some servers.

키워드

Session Clustering, Load Balancing, 고가용성, WEB, WAS

Ⅰ. 서 론

현재 대부분의 웹 시스템은 정적 요청을 처리하는 Web Server(이하 WEB)와 동적 요청을 처리하는 Web Application Server(이하 WAS)가 연동된 구조이다. 규모가 크고 이용자가 많은 웹 서비스의 경우 많은 트래픽과 시스템 장애에 대한 대비가 철저히 이루어져야 한다. 이를 위해 여러 대의 서버를 병렬로 연결하는 구조로 시스템을 구축하면 성능과 결함 내성을 향상할 수 있다.

Load Balancing은 웹 서비스에 들어오는 트래픽을 여러 대의 서버로 부하 분산하여 많은 트래픽을 처리할 수 있게 한다. 또한, Session Clustering은 서버들이 서로 통신하여 일부 서버가 중단되는 경우에도 해당 서버가 가졌던 정보를 다른 서버가 넘겨받아 처리할 수 있게 한다.

본 논문에서는 Load Balancing과 Session

Clustering을 통해 많은 트래픽의 처리가 가능하고 장애 시에도 서비스 중단이나 정보의 소실 없이 없는 고가용성 웹 시스템을 제안한다.

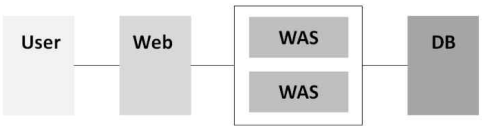

그림 1. 다수의 서버가 연결된 웹 시스템 구조

II. 관 련 연 구

2.1 Load Balancing

웹 서비스에서 발생하는 트래픽을 병렬로 연결된 여러 대의 서버가 부하 분산하여 처리하게 하는 기능이다. 네트워크 상단에 트래픽을 분배하는 load balancer가 존재하여 동시에 들어오는 수많은 요청을 각 노드로 부하 분산시킨다. 이를 통해 서버의 가용성과 응답시간을 최적화시킬 수 있다.

2.2 Sticky Session

load balance 적용 시 클라이언트와 요청을 처리하는 서버 간의 일관적인 대응을 보장할 수 없다. Sticky Session은 해당 세션 정보를 가지고 있는 서버가 일관적으로 해당 클라이언트의 요청을 처리하여 연속성을 보장할 수 있다.

2.3 Session Clustering

다수의 서버가 특정 목적을 위해 서로 소통하며 마치 하나처럼 동작하게 하는 기능이다. 이때, 임의의 서버에 장애가 발생하여 중단되면 다른 서버로 세션 정보를 전달한다. 이를 통해 사용자에게 세션이 끊기지 않고 연속적인 서비스가 가능하다.

III. 구 현 방 법

본 논문에서는 하나의 WEB에 두 개의 WAS를 연결하여 WAS간의 트래픽 분산과 장애 시 fail-over에 대한 시험을 진행하였다.

시스템이 실행되는 운영체제는 Amazon Web Service에서 제공하는 Amazon Linux2 AMI로 구성하였으며 오픈소스 미들웨어 Apache HTTP server (ver. 2.4.39)와 Wildfly(ver. 16.0.0.Final)를 사용해 WEB과 WAS를 구성하였다.

3.1 WAS 구축

Wildfly는 domain과 standalone 두 가지 모드로 실행이 가능하며 본 논문에서는 standalone모드로 인스턴스를 구성하였다. standalone모드는 인스턴스별로 개별관리 및 프로필 설정이 가능하다.

먼저 두 개의 standalone 인스턴스 이미지를 만들기 위해 /opt/wildfly/standalone 디렉토리를 복제하여 각각 node1, node2로 이름 붙인다.

표 1. 인스턴스 실행 스크립트

```
#!/bin/sh
./standalone.sh  - Djboss.node.name=인스턴스 이름
 - Djboss.server.base.dir=/opt/wildfly/인스턴스 이름
 — server-config=standalone-ha.xml
-Djboss.socket.binding.port-offset=오프셋 값
```

편의를 위해 각 인스턴스의 설정 경로와 두 개의 인스턴스가 충돌하지 않기 위해 port-offset 옵션을 추가한 인스턴스 실행 스크립트 파일을 만든다.

표 2. standalone-ha.xml 파일

```
<server name="default-server">
    <ajp-listener name="ajp" socket-binding="ajp"/>
                                .
                                .
<socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
<socket-binding name="http" port="${jboss.http.port:8080}"/>
```

ajp는 WEB으로 들어오는 요청을 WAS로 위임할 때 필요한 프로토콜이다. Apache HTTP Server와 Wildfly를 연동하기 위해 standalone-ha.xml 파일에 ajp프로토콜에 대한 리스너와 연결할 포트 설정을 해준다.

3.2 WEB server 구축 및 WAS 연동

Apache HTTP Server와 Wildfly의 연동에 필요한 mod_jk.so 모듈을 설치하고 httpd.conf 에 이에 대한 내용을 추가한다. JkMount에 로드밸런서 역할을 하는 WAS의 이름을 명시하고 JkWorkersFile을 통해 그 WAS의 정보를 설정할 수 있다.

표 3. httpd.conf 파일

```
#AJP13 Setting
LoadModule jk_module /etc/httpd/modules/mod_jk.so
JkWorkersFile /etc/httpd/conf/workers.properties
JkShmFile /var/run/mod_jk.shm
JkLogFile /var/log/mod_jk.log
JkLogLevel info
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"
JkMount /* loadbalancer
JkMountFile /etc/httpd/conf/uriworkermap.properties
```

3.3 Load Balancing 및 Sticky Session구현

workers.properties에서 WAS 구성에 대한 정보를 설정한다.

표 4. workers.properties 파일

```
# Define 1 real worker using ajp13
worker.list=loadbalancer

worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1, node2
worker.loadbalancer.method=Session # 트래픽 분산 방법
worker.loadbalancer.sticky_session=true

worker.node1.type=ajp13
worker.node1.host=127.0.0.1
worker.node1.port=8009
worker.node1.lbfactor=1
worker.node1.route=node1

worker.node2.type=ajp13
worker.node2.host=127.0.0.1
worker.node2.port=8109
worker.node2.lbfactor=1
worker.node2.route=node2
```

표 3 에서 명시한 loadbalancer의 type을 lb로 지정하고 세션 정보를 기준으로 부하를 분산하도록 설정한다. sticky session 설정을 true로 활성화 시켜주면 세션 정보를 기반으로 load balancing이 이루어진다.

3.4 Session Clustering 구현

Wildfly web application은 표 5 와 같은 구조를 가진다. WEB-INF/web.xml 파일은 웹 어플리케이션의 설정을 위한 deployment descriptor이다.

표 5. Wildfly web application project 구조

```
$ APP_HOME
    > test.jsp
    > WEB-INF
        > web.xml
```

표 6 과 같이 web.xml파일에 <distributable/> 태그를 추가하여 세션 정보가 복제될 수 있도록 한다.

표 6. web.xml 파일

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0"
                                .
                                .
    <session-config>
        <session-timeout>30</session-timeout>
<!--tracking-mode>URL</tracking-mode-->
    </session-config>
  <distributable/>
</web-app>
```

IV. 실 험 결 과

node1과 node2에 각 인스턴스의 이름과 세션 정보를 표시하는 웹 페이지를 만들어 배포하고 테스트 해보았다.

그림 2. 테스트 페이지 최초 접속 결과

그림 2 와 같이 구축한 웹 서비스에 처음 접속하면 로드밸런서에 의해 임의로 node1 인스턴스에 접속하게 되고 새로운 브라우저로 접속했을 때는 node2로 접속하게 된다.

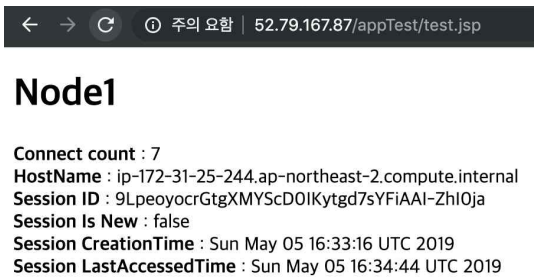

그림 3. 같은 세션정보로 여러번 재접속한 결과

node1로 접속했던 세션 정보로 여러 번 재접속하더라도 일관적으로 node1로 접속하게 된다.

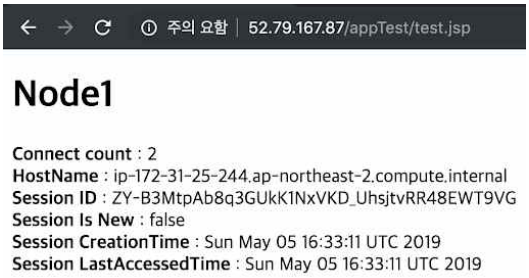

그림 4. node2를 중단시키고 재접속한 결과

node2 인스턴스를 중단시킨 뒤 node2에 접속해있던 세션 정보로 재접속을 했을 때, 기존의 세션 정보가 그대로 유지되면서 node1로 접속하여 서비스가 지속되는 것을 확인할 수 있다.

Ⅴ. 결 론

시스템을 확장 시키기 위해 대용량의 메모리 자원이나 고성능의 컴퓨팅 자원을 사용하는 데에는 물리적 한계와 비용적 부담이 따른다. 다수의 서버를 연결한 수평적 시스템 확장은 비교적 적은 비용으로 이러한 한계를 보다 유연하게 뛰어넘을 수 있다.

본 논문에서는 여러 대의 서버를 병렬로 연결하고 이를 효율적으로 운용하기 위한 기술들을 적용하여 시스템을 구축했다. Session Clustering을 통해 다수의 서버가 서로 통신하며 마치 하나의 서버처럼 동작함으로써, Load Balancing 적용 시에 트래픽이 분산되어 클라이언트의 요청이 어떤 서버에서 처리되더라도 그 세션 정보를 공유하여 일관적인 서비스 제공이 가능했다. 또한, Sticky Session을 통해 서비스로 들어오는 요청이 세션 정보를 기반으로 각 서버에 분배되어 항상 동일한 서버가 같은 사용자에 대한 요청을 처리하도록 보장하였다. 또한, 임의의 서버에 장애가 발생하였을 때, 해당 서버가 관리하던 세션 정보를 다른 서버가 넘겨받아 유지하면서 연속적인 서비스가 가능하였다. 이러한 방법으로 웹 시스템을 설계하면 많은 트래픽을 효율적으로 처리하고 서버 장애 위험을 분산시킬 수 있는 고가용성 시스템 구축할 수 있다.

이와 같은 고가용성 시스템에는 각 서버에 대한 병렬성과 처리할 데이터에 대한 정합성의 보장이 매우 중요하다. 후행 연구로서 추후 이러한 시스템의 도커(Docker) 이미지를 생성하여 컨테이너 기반의 웹 시스템으로 구축한다면 보다 간편하고 효율적인 버전 관리 및 자원 관리가 가능하고 새로운 환경으로의 이식성 및 호환성도 향상될 것이다.

References

[1] Jae-Won Choi, “Implementation and Fault-tolerance Tests of Load Balanced and Duplicated Active-Active Web Servers” The Journal of Korea Institute of Information and Communication Engineering, Vol. 18, No. 1, pp. 63-71, 2014. Jan.

[2] Red Hat Wildfly [Internet]. Available : https://wildfly.org/.

[3] Apache HTTP Server Project Documentation [Internet]. Available : https://httpd.apache.org.

Acknowledgment

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음"(2015-0-00912)

"This research was supported by the MSIP(Ministry of Science, ICT & Future Planning), Korea, under the National Program for Excellence in SW(2015-0-00912) supervised by the IITP(Institute for Information & communications Technology Planning&Evaluation)"