원격교육봉사체계 요청응답특성개선의 한가지 방법

김철혁, 어남일

선행연구[1]에서는 아파치HTTP봉사기에서 표준적으로 제공하는 mod_proxy모듈의 리용방법을 제안하였다.

선행연구[2]에서는 Tomcat과 아파치HTTP봉사기의 호상련결방법을 제안하였다.

론문에서는 원격교육운영체계의 요청응답특성을 분석하고 부하분산방법을 리용하여 요청부하를 분산시켜 봉사기요청처리능력을 개선하는 방법을 제안하였다.

1. 원격교육봉사체계의 요청부하특성분석

원격교육봉사체계의 구성과 요청응답특성, 망통신량세기는 다음과 같다.

① 원격교육봉사체계의 구성

봉사프로그람의 측면에서 볼 때 학습관리모듈(LMS), 업무자료기지(MySQLDB), 학습활동리력자료기지(MongoDB)로 구성되였다.

학습관리모듈은 Tomcat응용봉사프로그람, 업무자료기지는 MySQL관계형자료기지관리프로그람, 학습활동리력관리기지는 Mongo비관계형자료기지관리프로그람을 기반으로 하고있다.

학습자들과 업무관계자말단으로부터 들어오는 모든 요청은 학습관리모듈에서 처리하며 이 과정에 학습관리모듈은 학습활동리력자료기지와 업무자료기지를 호출한다.

② 원격교육봉사체계의 요청응답특성

현재의 웨브페지는 페지를 완전히 현시하는데 100개이상의 요청을 요구한다.

원격교육봉사기는 수천개의 학습자말단들로부터 들어오는 강의, 자체평가, 시험과 관련한 요청들과 업무관계자, 관리자말단에서 들어오는 요청들을 실시간적으로 중단없이 처리하여야 한다.

원격교육봉사체계의 요청응답시간을 평가하기 위하여 Google Chrome웨브열람기에서 가입 및 학습페지요청응답시간, 가입 및 시험페지요청응답시간을 측정하였다.

가입 및 학습페지요청응답시간을 표 1에 보여주었다.

표 1. 가입 및 학습페지요청응답시간

No.	요청형태	요청/수	응답시간/ms
1	첫 페지호출	52	1 200
2	가입	72	1 860
3	학습페지호출	42	897
4	록화형강의열기	24	140～930
5	평가내용물열기	22	80～658
Σ_0	합계	212	4 120～5 500
	평균요청응답시간		23.5

가입 및 시험페지요청응답시간을 표 2에 보여주었다.

표 2. 가입 및 시험페지요청응답시간

No.	요청형태	요청/수	응답시간/ms
1	첫 페지호출	52	1 200
2	가입	72	1 800
3	시험페지호출	31	3 000～9 600
4	시험과목선택	24	700
Σ_0	합계	179	7 000～13 300
	평균요청응답시간		12.8

③ 망통신량세기평가

리틀(little)의 규칙에 의하면 안정상태에서 특정망령역에 남아있는 평균통보문의 수 N은 통보문의 평균도착률 λ에 통보문들의 망령역평균경과시간 T를 곱한것과 같다. 즉 $N = \lambda \times T$ 이다.

$M/M/1$모형에 따라 시간간격 T내에 k개의 통보문도착확률을 P라고 할 때

$$P = (\lambda T)^k e^{-\lambda T} / k!, k = 0, 1, 2, \cdots$$

과 같이 표시된다.

6h사이에 평균가입회수에 따르는 요구도착확률을 표 3에 보여주었다.

표 3. 6h사이에 평균가입회수에 따르는 요구도착확률

가입/수	시간/h	λ	T/s	N	k	P
1 200	6	0.056	5	0.28	1	0.212
2 400	6	0.11	5	0.55	1	0.313

평균봉사률을 μ라고 하면 평균봉사시간은 $1/\mu$이다.

통신량세기는 다음과 같이 표시된다.

$$\rho = \lambda_0 / \mu, 0 \le \rho \le 1$$

여기서 λ_0은 요청도착률(통보문은 여러개의 부분요청으로 이루어짐.)이다.

6h사이에 평균가입회수에 따르는 통신량세기를 표 4에 보여주었다.

표 4. 6h사이에 평균가입회수에 따르는 통신량세기

가입/수	$\lambda_0(\lambda \times \Sigma_0 / 5)$	μ	ρ
1 200	2.36	42.4	0.056
2 400	4.72	42.4	0.11

표 4에서 Σ_0은 학습페지평균요청응답시간(5s)동안 1개 사용자말단에서 발생한 요청개수이고 안정상태에서 $\lambda_0 \le \mu$이다. 그리고 $T = 1/\mu \times 1/(1-\rho)$이다. 이것은 통보문의 평균지연이 평균봉사시간의 $1/(1-\rho)$ 배라는것을 보여준다. ρ가 1에 가까와갈수록 T는 무한대로 커진다.

$\rho - T$ 관계를 보면 $\rho > 0.7$에서 T가 급격히 증가된다. 즉 봉사률이 클수록, 성능이 높을수록 통신량세기에 따르는 지연이 줄어든다는것을 알수 있다.

$\rho - \lambda$ 관계곡선을 그림 1에 보여주었다.

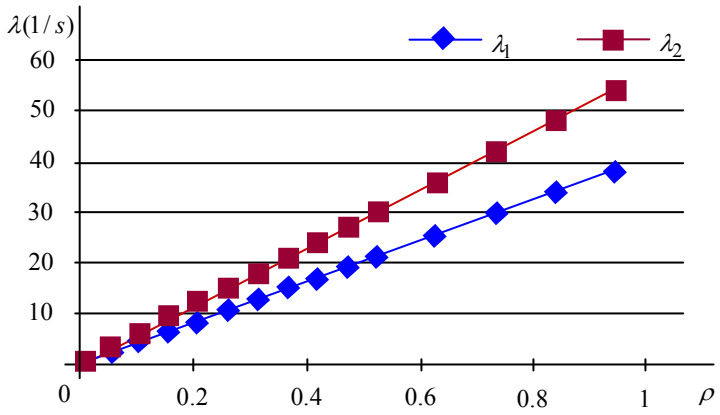

그림 1. $\rho-\lambda$ 관계곡선

그림 1에서 λ_1, λ_2는 각각 $\mu=42.4, \mu=60$일 때의 도착률곡선이다. $\rho=0.9$일 때 도착률과 가입회수와의 관계를 표 5에 보여주었다.

표 5. $\rho=0.9$일 때 도착률과 가입회수와의 관계(6h동안)

가입/수	$\lambda_0(\lambda\times\sum_0/5)$	λ	μ/s^{-1}	ρ
19 440	38.16	0.9	42.4	0.9
27 432	54	1.27	60	0.9

$\rho=0.5$일 때 가입회수와 μ관계를 그림 2에 보여주었다.

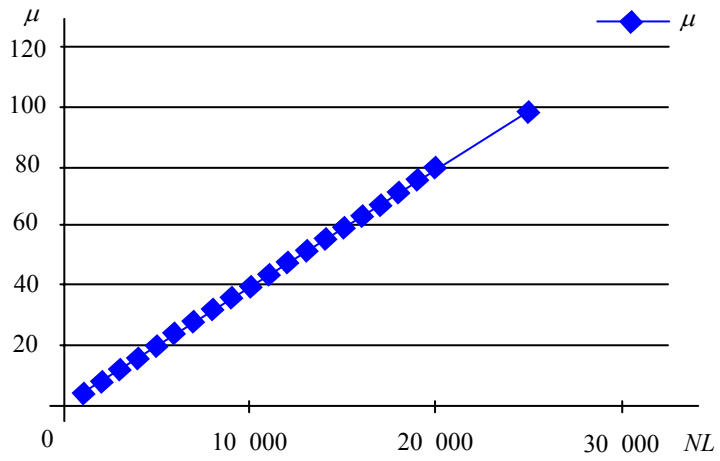

그림 2. $\rho=0.5$일 때 가입회수와 μ관계

$\mu-T$ 관계에서 $\mu=43.2$일 때 $T=4.9$s이다. 이것은 1개 학습자말단으로부터 학습실 방문요구에 대한 응답시간이다. 이때 가능한 가입회수는 11 000회/6h이다. 성능을 1.5배 높일 때 즉 $\mu=66.7, T=3.17$s에서 가입회수가 17 000회/6h정도로 늘어나도 체계는 안정상태를 유지할수 있다.

2. 요청응답특성개선방법

봉사기의 요청응답특성은 응용프로그람준위와 봉사프로그람준위에서 개선할수 있다. 여기서는 아파치봉사기를 부하조종기로 하고 들어오는 요청을 여러개의 Tomcat실체에 분산시켜 처리하는 방법에 대하여 론의한다.

요청흐름은 다음과 같다.

① 열람기에서 URL을 지적하면 요청은 HTTP봉사기에서 접수한다.

② HTTP봉사기는 요청을 접수하고 업무론리를 처리하기 위하여 Tomcat에 요청처리를 넘긴다.

③ Tomcat은 자료기지봉사기와 련계하여 필요한 처리를 진행하고 요청접수통로를 통하여 응답을 귀환시킨다.

아파치HTTP봉사기를 리용하여 부하조종을 실현하는 우점은 다음과 같다.

image, JS, CSS, HTML화일과 같은 정적내용에 대하여 Tomcat에 비하여 10%이상 높은 속도로 처리될수 있다.

아파치HTTP봉사기는 여러개의 Tomcat실체와의 련결을 적은 비용과 높은 안정성을 가지고 실현하도록 하며 봉사체계의 전반적인 확장성과 유연성을 높여준다.

요청부하는 여러 Tomcat실체들로 분산되며 부하처리능력은 실체개수에 비례하여 개선된다.

부하분산기구성도를 그림 3에 보여주었다.

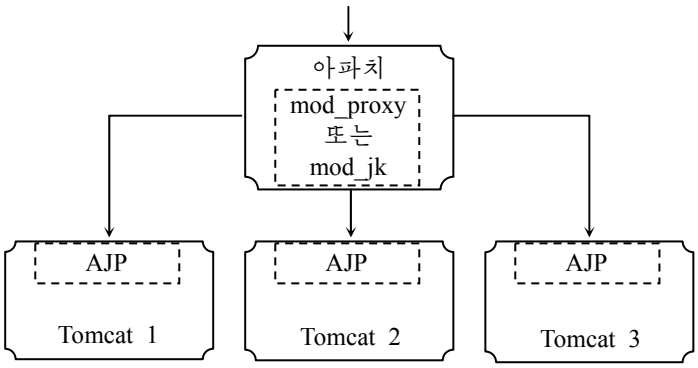

그림 3. 부하분산기구성도

아파치봉사기와 Tomcat실체들과의 련계는 TCP파케트에 기초한 HTTP 혹은 AJP(Apache Jserv Protocol)를 리용한다.

AJP는 일반본문형식대신 2진형식으로 망자료전송을 보장하기 위해 개발되였다.

mod_jk와 mod_proxy는 아파치봉사기와 Tomcat을 결합시키는 AJP련결기이다.

3개의 Tomcat실체를 리용하는 mod_jk련결기설정은 다음과 같다.

아파치웨브봉사기설정으로 workers.properties와 mod_jk.conf를 창조한다.

· mod_jk.conf내용

LoadModule jk_module modules/mod_jk.so

JkWorkersFile conf.d/workers.properties

JkLogFile logs/mod_jk.log

JkLogLevel info
JkMount /RNS/* bal1
JkMount /jkstatus/ stat1
· workers.properties 내용
worker.list = bal1, stat1
worker.NodeA.port = 8009
worker.NodeA.host = rnsServer
worker.NodeA.type = ajp13
worker.NodeA.lbfactor = 10
worker.NodeB.port = 8019
worker.NodeB.host = rnsServer
worker.NodeB.type = ajp13
worker.NodeB.lbfactor = 10
worker.NodeC.port = 8029
worker.NodeC.host = rnsServer
worker.NodeC.type = ajp13
worker.NodeC.lbfactor = 10
worker.bal1.type = lb
worker.bal1.sticky_session = 1
worker.bal1.balance_workers = NodeA, NodeB, NodeC
worker.stat1.type = status
Tomcat실체들에 대한 설정
Tomcat실체설치경로에서 conf/server.xml을 편집
<Server port = "8005" shutdown = "SHUTDOWN">부분에서 port값을 실체별로 8005, 8015, 8025로 설정한다.

<Connector port = "8009" enableLookups = "false"redirectPort = "8443" protocol = "AJP/1.3"/> 부분에서 port값을 실체별로 8009, 8019, 8029로 설정한다.

3개 Tomcat실체의 port설정을 표 6에 보여주었다.

표 6. 3개 Tomcat실체의 port설정

실체이름	수정할 화일이름	TCP port(Shutdown, Ajp Connector)
NodeA	../NodeA/conf/server.xml	8005, 8009
NodeB	../NodeB/conf/server.xml	8015, 8019
NodeC	../NodeC/conf/server.xml	8026, 8029

Engine부분에서 jvmRoute값을 수정한다.

<Engine name = "Catalina" defaultHost = "ocalhost" jvmRoute = "NodeA">

3개 Tomcat실체의 jvmRoute값설정을 표 7에 보여주었다.

표 7. 3개 Tomcat실체의 jvmRoute값설정

실체이름	수정할 화일이름	jvmRoute 값
NodeA	../NodeA/conf/server.xml	NodeA
NodeB	../NodeB/conf/server.xml	NodeB
NodeC	../NodeC/conf/server.xml	NodeC

3. 실험결과분석

실험환경을 표 8에 보여주었다.

표 8. 실험환경

No.	환 경	기술적지표
1	실험봉사기장치환경	HP Pro CPU: intel corei3－3240 3.4GHz 주기억: 1737936KB 체계기동후 free: 458896KB buffer: 121824KB 캐쉬: 801964KB 망포구 1개, 1Gbps
2	망하브(1대)	Mercury s108c(8포구)100Mbps
3	실험봉사기조작체계	Linux－2.6.32－358－el6_x86_64
4	응용봉사프로그람	Apache Tomcat/7.0.56 Java－openjdk－1.7.0.9.x86_64
5	자료기지봉사프로그람	Mysqld v5.6.25 Mongod v3.0.4
6	웨브응용프로그람	RNS.war, LRStore.war

아파치성능분석도구 ab를 리용하여 진행한 실험결과를 표 9에 보여주었다.

표 9. 아파치성능분석도구 ab를 리용하여 진행한 실험결과

실험항목	1개 실체	2개 실체	3개 실체
처리시간/ms	53	14	14
대기시간/ms	13	12	12
50%요청처리시간/ms	11	2	3
80%요청처리시간/ms	19	3	3
95%요청처리시간/ms	281	4	5
99%요청처리시간/ms	1 337	214	31

실험은 ab선택파라메터(c=1 000, n=28 000), 호출페지(RequestInfoExample)에서 진행하였다.

맺 는 말

리론적으로는 부하분산에 리용되는 봉사실체개수에 비례하여 요청처리능력은 높아지지만 실제적으로 체계절환시간과 장치자원의 제약으로 하여 리론값과 차이난다. 실험환경에서 볼 때 부하분산기술을 2개의 봉사실체에 적용할 때 1개 실체를 리용할 때에 비하여 요청평균처리시간이 대략 1/3로 줄어든다는것을 확증하였다.

참 고 문 헌

[1] Marko Maslac; Apache HTTP Server Introduction, Autun Peicevic, 32～38, 2016.
[2] Tanuj Khare; Apache Tomcat7 Essentials, Packt, 91～123, 2012.

주체109(2020)년 5월 5일 원고접수

A Method to Improve Request-Response Performance in E-Learning Server System

Kim Chol Hyok, *O Nam Il*

In this paper, we analyse some of request-response characters of e-learning server system and propose a method to improve these characteristics based on load sharing.

Keywords: request-response character, load sharing, e-learning server system