Event Driven HTTP Server:
Projekt i Nätverksprogrammering (EDA095)

EDA, Institutionen för Datavetenskap, Lund
Tekniska Högskola
Roger Henriksson

Johan Bäversjö, C11 (dic11jba@student.lu.se)
Mikael Gråborg, C11 (dic11mgr@student.lu.se)
Petter Henriksson, C11 (dic11phe@student.lu.se)
Mergim Rama, C11 (dic11mra@student.lu.se)

23 maj 2013

LUNDS
UNIVERSITET
Lunds Tekniska Högskola

Innehåll

1 Bakgrund

I kursen nätverksprogrammering (EDA095) fick vi i uppgift att göra ett slutprojekt som skulle implementera tekniker vi lärt oss under kursens gång. Vi beslöt oss för att göra en HTTP server men av mera komplex typ. Uttöver HTTP servern implementerade vi även funktioner som ska komplettera servern och göra den mer flexibel för filöverföring. HTTP servern är event driven server, vilket medför en mera komplex arkitektur jämfört med en trådbaserad server. Varför vi valde en event driven HTTP server framför en trådbaserad server är för att den kan hantera flera HTTP request samtidigt vilket medför att fler kan nå filerna på servern snabbare.

2 Kravspecifikation

För att få konkreta mål med projektet var vi tvungna att ställa upp krav på vad vår server ska klara av innan implementationen inleddes samt vad den klarade av efter att vi fått den att fungera.

2.1 Krav

- Vi ska lära oss hur HTTP protokollet är uppbyggt och hur vi kan implementera detta.
- Vår server ska vara baserad på asynkrona metoder.
- Implementera SSL.
- Implementera caching.

3 Modell

Vi använder huvudsakligen en klass Server för att hantera anslutningar. I Server initieras i sin tur ett Clientobjekt för varje anslutning. Clientobjekten delegeras till en tråd och lagras sedan i en HashMap lokalt i respektive tråd. För att öka effektiviteten finns det lika många trådar som det finns kärnor. Requesten från en klient parsas i Client och ett response instansieras, exekveras genom alla FileMiddleWare. FileMiddleWare lägger i sin tur till allt nödvändigt i Responseobjektets headers, och slutligen skickar den även ut filen som efterfrågats.
Den mest använda datastrukturen är HashMap som används till att lagra både klienter och headers. För att trådarna ska vara så effektiva som möjligt är även klienterna sorterade i en priorityqueue i vardera tråd. Det som händer när en request skickas till vår webbserver är att requesten behandlas som en instruktion som instuerar kerneln att utföra en viss handling. Detta är en så kallad zero copy - transferTo funktion som gör servern effektiv. När man då ska hämta en fil från servern kommer requesten att instruera kerneln att hämta filen från hårddisken som sedan sparas i serverns RAM-minne för direkt åtkomst. Under denna process kommer servern att använda sig av blocking I/O vilket behövs för att få tillgång till filen. När man använder sig av blocking I/O så blockeras all kommunikation tills dess att filen skickats tillbaka. Detta gör en server långsam men detta är tvunget då man på något sätt måste nå filen första gången. När

filen sedan sparats i RAM-minnet kan resterande anslutningar använda sig av non blocking I/O vilket tillåter flera request (målet med vår server) samtidigt. Med en non blocking I/O blockeras inget vilket betyder att flera anslutningar kan fråga servern efter olika saker. Det som som händer då är att kerneln skickar tillbaka en callback som anropar rätt anslutning så att den kan få det efterfrågade.
För att implementera detta använde vi oss av java 7 och java.nio. Java.nio är en samling av APIs utvecklat för att hantera centraliserade I/O operationer.

4 Användarhandledning

En server är inte något en vanlig användare tänker på när han är ute på Internet och surfar efter webbplatser. Han kommer i kontakt med servern när ett webförmulär ska laddas och endast då. Vi valde därför att skapa vår användarhandlening utifrån en som dagligen arbetar med servrar, det vill säga nätverksadministratörer. Eftersom en nätverksadministratörs arbetsuppgifter handlar om att underhålla och upprätthålla servrar kändes detta som ett väldigt naturligt val.

Det administratören måste göra är att skapa en mapp där alla formulär/ html filer ska finnas tillgängliga i. Detta för att när servern försöker nå filerna måste de finnas på en gemensam plats som man specifierar.

1. Javaprogrammet som ska köra processen måste få rättigheter till webroot, mappen som filerna ligger i. Det bör administratören kunna utföra självmant. För att programmet sedan ska hitta till webroot finns en "environment" variabel kodad i programmet som ger sökvägen till den specifika mappen där html filerna ligger. Det görs enklast i terminalen, men kan variera beroende på vilket operativsystem som körs. Det som man också bör tänka på är att sätta en specifik port på servern, vilket görs på samma sätt som när man ska deklarera en sökväg i environment variabeln för mappen med html filer.

2. Eftersom vi använt bibliotek från java 7 för att skapa servern så måste admininstratören installera detta för att få servern att fungera. Utan Java 7 eller någon nyare version kommer inte servern att fungera eftersom java saknar framåtkompabilitet.

3. Det sista steget är ganska självklart för en erfaren administratör samt datoranvändare, det vill säga exekvera Server.java i en lämplig kompilator som till exempel eclipse.

5 Utvärdering

Vi har uppfyllt samtliga krav förutom att implementera SSL och att servern ska använda sig av asykrona metoder. Vi valde att istället för att använda de asynkrona metoderna, göra vår server eventdriven. Det berodde på att vi inte hade tillgång till transferTo med asynkrona sockets. Vi gjorde därför en avvägning för att behålla hastighet och prestanda. SSL blev aldrig implementerat eftersom

detta inte var möjligt i vår server. Detta på grund av att vi aldrig hämtar filerna till javaprogrammet utan de skickas ut direkt på länken, även det på grund av transferTo. Javaprogrammet kommer endast agera som instruktör då den skickar kommandon som skall utföras. I övrigt fungerar servern mycket bra och kan ta emot och hantera väldigt många anslutningar samtidigt. Självklart finns det fortfarande brister i vår minneshantering och framförallt hårdvara. Vilket gör att servern inte presterar så bra som den skulle kunna göra. Projektet var mycket lärorikt men svårt att få grepp, med mycket fram och tillbaka i arkitekturen, och möjligen var projektet något ambitiöst för den korta tidsperioden. Eftersom så mycket ansvar ligger på studenterna är det svårt att ändra något till det bättre, till nästa kursomgång. Möjligen kan handledarna vara mer involverade i projekten och driva dessa framåt bättre.