Tidsglapp hotar Molnet?

Men med god SOA-design kan flexibiliteten i Molnet tillvaratas

Det finns en infrastrukturegenskap som inte förbättrats särskilt mycket de senaste åren. I vissa fall kanske den till och med i praktiken har försämrats: medan bandbredd hela tiden ökat, så har tidsglapp – latens – inte förbättrats. När molnets alla fördelar nu ska utnyttjas kan latens ge stora problem. Men det finns motåtgärder i form av medveten SOA-design.

Latens är generellt uttryckt den tid det tar innan en viss behandling påbörjas. På engelska heter det latency, på svenska kan ordet tidsglapp fungera som alternativ till latinets latens.

Ett exempel: En nätverksrouter måste ta in ett komplett paket med data innan den kan börja avgöra ifall det är korrekt och vart det ska. Vi förlorar denna tid innan paketet kan börja skickas vidare – latenstid. I en kedja av intranet/internet-kommunikation blir ofta antalet sådana mellanstationer högt. Alla dessa nätverkslatenser adderas förstås. Dessutom läggs latenser till som beror på att ljushastigheten sätter tak för hur snabbt data kan börja komma fram. Ifall kommunikation exempelvis sker mellan ett huvudkontor i Sverige och ett lokalkontor i Kina ger det en konkret påverkan. Används satelliter för kommunikationskedjan försvinner hela 0,5 s i satellitlatens för fråga/svar.

Mobil kommunkationslatens kan bli 0,3 s

I ett exempel på kommunikation för mobila användare har jag uppmätt latensen 0,3 s då jag använt mobilens 3G-uppkoppling via Bluetooth för en bärbar dator och nått www.bredbandskollen.se. Kommandot *ping* ger liknande mätetal. Ifall en applikation behöver göra 20 serveraccesser bakom kulisserna för att börja visa ett formulär för användaren med denna latens så förloras alltså 6 s redan utan att någon egentlig serverexekvering eller dataöverföring har börjat förbruka tid.

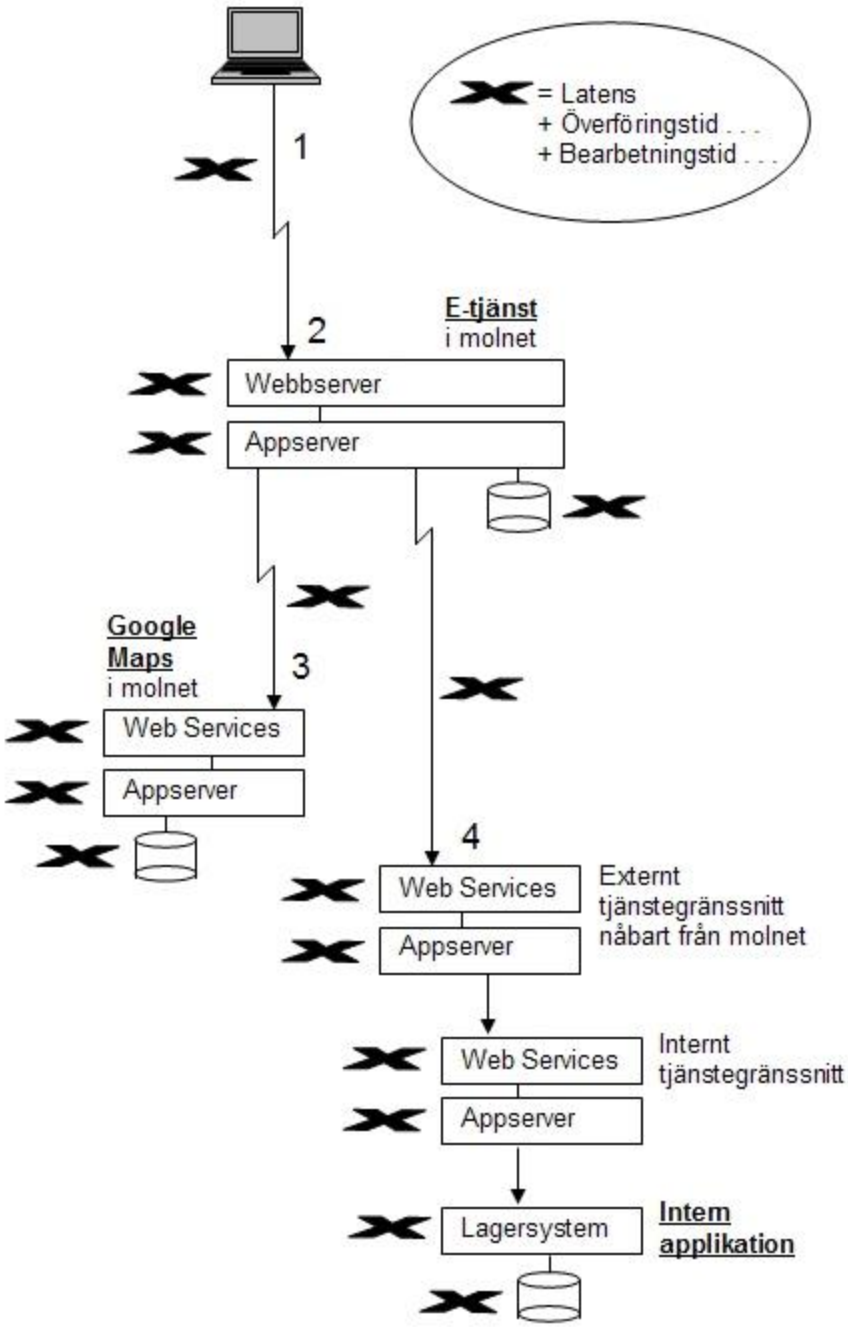
Andra exempel på tidsglapp är applikationsservrar och ESB:er som har en viss fördröjning innan en inkommen begäran har åstadkommit att tillhörande programobjekt är initierat och börjar exekvera denna begäran. EAI-nav har vanligen ännu längre fördröjning, kanske så mycket som 15-30 s. Ytterligare ett exempel är XML-bearbetning,

där oftast en hel XML-struktur, som kanske kan vara i megabyte-storlek, först måste analyseras och delas upp i alla sina beståndsdelar i primärminne innan ett program kan börja nå dataelement inne i strukturen. Allt detta förbrukar också latenstid. I en större SOA-lösning kan ett antal mellanservrar vara inblandade när önskvärd återanvändning sker, och då kan upplevd prestanda definitivt påverkas mycket negativt om inte applikationsarkitekten ser upp.

Många intresserar sig för just nu för att använda det så kallade *Molnet, (Cloud Computing)*. Vad betyder det i det här sammanhanget? Jo, att ett antal SOA-tjänster och servrar som ingår i en avancerad lösning fysiskt sett ligger hos leverantörer som kan finnas på olika ställen runt jorden, Dessutom finns ofta behov av interna tjänster och servrar i helhetslösningen.

Följande exempel på kedja kan vara realistisk för en sammansatt lösning i Molnet:

1. Användarens dator är ansluten till ett företags intranät.
   Latenser uppstår i interna routrar, switchar etc, samt i brandväggen ut till Internet.
2. Webbservern som står för applikationens formulär ligger i Molnet hos en tjänsteleverantör i USA. Nätverkslatenser uppstår i Internets alla routrar på vägen, samt i viss grad på grund av ljushastigheten. Själva webbservern har en latens, bakomliggande applikationsserver adderar ytterligare latens.
3. Applikationsserverna ska kanske i sin tur skicka en begäran bakom kulisserna om data från en tjänst som Google Maps. Här läggs Internetlatens till, samt latens i Googles applikation.
4. Applikationsservern ska kanske också skicka en begäran om information till företagets interna lagersystem. Dessa nätverkspaket drar latenstid genom Internet, sedan tillkommer företagets brandväggskomplex, latens i servern för Web Services samt i bakomliggande applikationsserver. Denna skickar i sin tur en Web Service-fråga till det interna lagersystemet, där samma sorts latenser adderas. Längst bak ska latenser hos en SQL-databas läggas till.



Till alla latenstiderna måste läggas den tid som själva överföringen av informationen tar. Handlar det om enklare lagerdata blir denna tid kort, men är det fråga om bilder, kartmaterial, multimedia etc förbrukas mer överföringstid. Många gånger är det heller inte bara en förfrågan som behövs bakom kulisserna, utan kanske en loop med tjogtals och då multipliceras förstås latenserna.

Till detta kommer vanlig serverexekveringstid.

Först därefter kan webbsidan visas. Risken är överhängande att prestanda upplevs som dåliga. Man kan i och för sig konstruera simpla applikationer som inte har någon integration för att nå resten av informationen som användaren behöver, men detta skulle leda till ökat manuellt arbete och det var väl inte riktigt meningen med it-stödet.

Vad kan vi lära oss av det här?

Jo, egentligen är det kända dygder från god SOA-design som ska tillämpas, men det blir extra viktigt när flexibiliteten hos Molnet ska tas tillvara. Några karaktäristika att eftersträva för en sammansatt lösning som i exemplet ovan:

- Grovkornig informationsöverföring.
  Dvs gör inte en loop för att hämta fakturarader en och en över nätet, utan hämta fakturahuvud och fakturarader i en enda access. Om total latens för en hel kedja är 0,5 s och man slipper en loop på tjugo varv så har man sparat tidsglappet 9,5 s!
- Använd realtidshämtning av information endast när det är riktigt välmotiverat, såsom för ett lagersaldo. I andra fall använder men store-and-forward via en kö (exempelvis för att lagra en faktura) eller replikering av information till en databas nära webbservern (till exempel för artikeldata). Nyttovärdering ska göras när man informationsmodellerar för att ta fram optimala färskhetsbehov för datat!
- Skapa teknikprototyper för att i ett tidigt skede simulera hur både tider för latens, serverexekvering och informationsöverföring för en trolig applikationsarkitektur adderas.

En del Moln-leverantörer har angett att det ska gå för kunden att specificera i vilket land aktuella Moln-serverrreurser ska stå. Detta kan naturligtvis vara av intresse juridiskt sett, men även för att kraftigt reducera latenstidsriskerna.

Slutsatsen blir att applikationsarkitekturen som uppstår när Cloud Computings alla fördelar ska utnyttjas samtidigt kan innebära stora problem med tidsglapp – latens – och att det finns motåtgärder i form av en god och medveten SOA-design!

Förkortningar:
ESB: Enterprise Service Bus
EAI: Enterprise Application Integration
SOA: Service Oriented Architecture

- - - - -

Som publicerat på *www.trendspaning.se* i januari 2009
Sven-Håkan Olsson, Definitivus AB
www.definitivus.se