



## **Erlang får nytt liv**

Text: Tomas Mannerstedt

### ***Inledning***

Utvecklingen inom processorbranchen har de senaste åren gått mer och mer åt fler processorkärnor i stället för högre klockfrekvens. För att kunna utnyttja denna form av arkitektur tvingas utvecklare till att skriva multitrådade program. Den som någon gång försökt skriva ett multitrådat feltolerant program i C, C++ eller Java är säkerligen medveten om de problem detta innebär. Borde det inte kunna vara enklare? Borde det inte finnas ett programmeringsspråk där man från början tänkt på multitrådade och distribuerade program? Svaret är självklart ja och en av lösningarna är Erlang.

### ***Historik***

"Erlang? Har inte Ericsson lagt ner det än?" Det var den kommentar jag fick höra första gången jag berättade för en kollega i branchen om att jag jobbade med Erlang. Faktum är att Erlang är gammalt men nu hetare än någonsin.

Erlang utvecklades på Ericsson under 1980-talet. Målet var att ta fram ett språk eller en utvecklingsmetodik som mappade väl mot den parallellism man hade i de system man utvecklade. Man hade också höga krav på feltolerans samt krav på mjuk upgradering.

**VI ÄR ALLA KONSULTER PÅ COMBITECH**, ett av Sveriges största konsultföretag som kombinerar teknik, miljö och säkerhet. Vi är ca 800 medarbetare på 20 orter.

1991 släpptes den första releasen till användare och 1998 den första kommersiella produkten, AXD301, utvecklad av Ericsson. Samma år bannades Erlang från Ericsson och släpptes som Open Source. Numera är dock Erlang tillbaka som utvecklingspråk i vissa delar av organisationen.

## ***Erlang som koncept***

Vad är det då som gör att Erlang lämpar sig så väl för multitrådade applikationer? Tanken med Erlang är enkel. Konceptuellt saknar man trådar. All form av multitrådning görs i stället med processer. Att starta en process i Erlang är en billig operation. Två (eller flera) processer delar inget minne och därmed saknas även kritiska sektioner som på något vis måste skyddas med lås i form av semaforer. Kommunikation mellan processer sker med asynkrona meddelanden. Detta är vad som brukar kallas för *share nothing concurrency*.

Processerna som man skapar i Erlang mappar inte mot operativsystemets processer eller trådar. Erlang körs i en virtuell maskin, precis som Java, och skapar en exekveringstråd med schemaläggare för Erlang-processer för varje processorkärna. På så vis fördelas Erlang-processerna jämnt mellan olika processorkärnor. Processorns kapacitet utnyttjas därmed maximalt.

En viktig del i ett system med extremt många parallella händelser är feltolerans. Systemet måste kunna upptäcka om andra delar av systemet slutat fungera. Detta är något man tänkt på från början i Erlang. Processer är ordentligt isolerade från varandra, och en krasch i en process får inga obehagliga bieffekter för de andra processerna. Processer kan även övervaka varandra, och en process kan på så vis få kännedom om att en annan process kraschat och då vidta de åtgärder som behövs. En inte alltför ovanlig lösning är att skriva sin kod på ett sådant vis att den kraschar direkt när något fel upptäcks (ett ogiltigt paket i ett telekomsystem, en ogiltig transaktion i ett banksystem etc.). En övervakningsprocess kan då upptäcka att operationen misslyckats och signalera detta vidare.

Erlang är funktionellt vilket innebär att man saknar for-loopar och while-slingor som är vanligt förekommande i procedurrella språk. Man använder i stället rekursion. Liksom flera andra funktionella språk använder Erlang single-assignment, vilket innebär att en variabel bara kan tilldelas ett värde en gång. Därefter fungerar variabeln som en konstant.

En annan annorlunda detalj med Erlang är att det är strikt men dynamiskt typat. Man skapar typer "on the fly" och kan

mönstermatcha på typ, struktur eller innehåll.

Hos många funktionella språk saknar man sidoeffekter. Med Erlang gäller detta dock inte helt. Vad man däremot kan säga är att det i allmänhet är enklare att hålla reda på de sidoeffekter man har än i andra procedurella eller objektorienterade språk. Detta är en viktig egenskap vid parallellisering.

Låter det svårt? Inte alls, Erlang är faktiskt ett enkelt språk att lära sig så länge de egna, tidigare programmeringserfarenheterna inte präglats alltför hårt av procedurella eller objektorienterade språk. Programmerare tenderar att, av de språk de använder, begränsas i sitt sätt att tänka.

## **Användning idag**

De flesta som idag använder Erlang gör detta för att konstruera system som hanterar extremt många parallella händelser. En sådan tillämpning är telekomsystem. En annan tillämpning som kommit starkt de senaste åren är webbapplikationer. Man skulle även kunna tänka sig olika former av banksystem, vilket faktiskt är något som har praktiserats en tid.

Historiskt är Ericsson den stora användaren av Erlang. Som "success story" brukar ofta nämnas Ericssons AXD301. Den senaste

tiden har även ett antal andra projekt utvecklade i Erlang presenterats. Ett av dessa är den chattjänst som Facebook använder sig av. Ett annat exempel är Amazons SimpleDB, vilket är en distribuerad databas.

Not: Tomas Mannerstedt har arbetat på Combitech sedan han tog civilingenjörsexamen i datateknik på KTH. På uppdrag hos Ericsson har han bland annat jobbat med tester och testmiljöutveckling.



**VI ÄR ALLA KONSULTER PÅ COMBITECH**, ett av Sveriges största konsultföretag som kombinerar teknik, miljö och säkerhet. Vi är ca 800 medarbetare på 20 orter.