Erlang is een hogere programmeertaal voor concurrent soft real-time systemen

De taal van Ericsson

Net als Java is Erlang afkomstig uit het ontwikkellab van een grote onderneming, in dit geval Ericsson. En evenals Java is Erlang een open-source taal, hoewel Ericsson ook commerciële licenties heeft met bijbehorende ondersteuning. Erlang is een functionele programmeertaal met ingebouwde primitieven voor concurrency en distributed computing en een geavanceerde foutafhandeling. Het is gebruikt voor de Ericsson AXD 301/305 MultiService Switch en Erlang is beschikbaar voor Solaris, Linux, Windows en VxWorks.

HANS VAN THIEL

Geen assignments? Geen assignments: een statement als 'x = x + 1' is niet mogelijk in Erlang en er zijn ook geen 'for' en 'while' lussen. In plaats daarvan gebruikt Erlang recursie (zie listing 1), met 'pattern matching' als het basismechanisme voor evaluatie van termen.

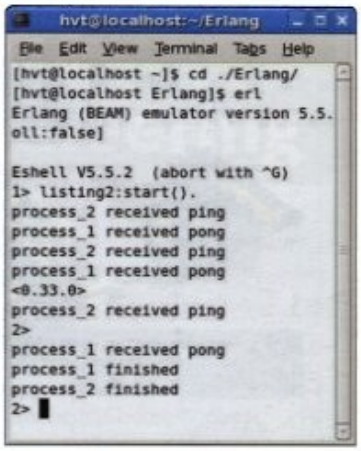
Erlang is een functionele programmeertaal, wat wil zeggen dat het resultaat van een bewerking alleen afhangt van de invoer, zonder neveneffecten (side effects). Als een variabele X (hoofdletter in Erlang) eenmaal een waarde heeft gekregen, dan kan diezelfde variabele overal weer gebruikt worden, met gegarandeerd diezelfde waarde. In samenhang met een streng modulaire opbouw moet dit de kans op programmeerfouten verkleinen en de menselijke leesbaarheid vergroten.

Anders dan andere functionele programmeertalen is Erlang echter speciaal ontworpen voor het programmeren van real-time software, met name in telecommunicatiesystemen. De taal heeft dan ook ingebouwde voorzieningen voor parallellisme en gedistribueerde verwerking. Erlang en de bijbehorende standaardbibliotheken hebben zich bewezen in industriële toepassingen van Ericsson zoals de AXD 301/305 MultiService Switch met honderden tot duizenden afzonderlijke processen. Context switching tussen Erlangprocessen is tot tweemaal goedkoper dan wisseling van vergelijkbare C-threads, aldus de open source faq.

Geschiedenis

De ontwikkeling van Erlang is al begonnen in de jaren '80 bij het Ericsson Computer Science Laboratory (ref.1). De naam 'Erlang' staat overigens niet voor 'Ericsson Language'. Dit was wel een min of meer gewenste toevalligheid, maar Erlang is genoemd naar de wiskundige Agner Erlang, wiens theorieën over waarschijnlijkheidsrekening veel worden gebruikt in telecommunicatie.
Het doel van het project was om te onderzoeken of declaratief programmeren ook geschikt zou zijn voor grote industriële telecomsystemen. Hierbij werd vooral gekeken naar Prolog, een taal die destijds populair was als taal voor kunstmatige intelligentie.

Een declaratieve taal kent geen assignments en expliciete control-lussen, maar is gebaseerd op recursie. Recursie is het herhaald aanroepen van dezelfde functie, maar met steeds eenvoudiger argumenten, terminerend in een basisdefinitie. Een voorbeeld is de definitie van 'faculteit' in Listing 1; fac(0) is 1 en fac(N) is gelijk aan N maal fac(N-1).
De nieuwe taal moest geschikt zijn voor soft real-time systemen, waarbij 'soft-real-time' opgevat moet worden in de trant van: een database look-up moet in 97% van alle gevallen binnen 20 ms zijn afgehandeld.
Het onderzoek werd uitgevoerd in nauwe samenwerking met hardwareontwikkelaars en uiteindelijk werd Erlang binnen Ericsson ook gebruikt voor productiesystemen. Er kwam een collectie standaardbibliotheken onder de naam OTP (Open Telephony Platform) met onder meer een ASN.1 compiler, een SDL-vertaler en een gedistribueerde real-time database. Abstract Syntax Notation 1 wordt in telecom veel gebruikt om datastructuren te beschrijven en de Specification and



Figuur 1. Het resultaat van Listing 2 op de Erlang shell.

```
-module(listing1).
-export([hello_world/0, fac/1]).

hello_world() -> io:fwrite("Hallo, World\n").

fac(0) -> 1; fac(N) -> N * fac(N-1).
```

Listing1: 'Hello World' in Erlang en de faculteit van een positief geheel getal. Recursie, zonder assignments en loops, lijkt op de mathematische definitie. De eerste twee regels illus-treren de modulaire opbouw van Erlang. De '-export' regel noemt de functies, elk met het aantal van hun argumenten. Een definitie in Erlang wordt afgesloten met een punt.

```
-module(listing2).
-export([start/0, process_1/2, process_2/0]).

process_1(0, Process_2_PID) ->
    Process_2_PID ! finished,
    io:format("process_1 finished~n", []);

process_1(N, Process_2_PID) ->
    Process_2_PID ! {ping, self()},
    receive
        pong -> io:format("process_1 received pong~n",[])
    end,
    process_1( N-1, Process_2_PID).

process_2() ->
   receive
       finished -> io:format("process_2 finished~n",[]);
       {ping, Process_1_PID} ->
             io:format("process_2 received ping~n",[]),
             Process_1_PID ! pong,
             process_2()
   end.

start() ->
       Process_2_PID = spawn(listing2, process_2, []),
       spawn(listing2, process_1, [3, Process_2_PID]).
```

Listing 2: Message passing tussen twee processen in Erlang. De uitdrukking 'PID ! Msg' betekent: 'stuur de boodschap Msg naar het proces met identifier PID'. De 'receive' clausule leest boodschappen uit de proces postbus; de zender is klaar zodra de boodschap is verzonden. Variabelen beginnen altijd met een hoofdletter; identifiers met een kleine letter zijn 'atoms'. Uitdrukkingen tussen {} zijn 'tuples' met een vast aantal elementen en vierkante haken omsluiten een 'list', met een variabel aantal elementen. (De io:format functie dient hier alleen om in de Erlang shell te kunnen controleren dat er inderdaad drie maal ping en pong 'atoms' worden verstuurd.) Bron: http://www.erlang.org/doc/doc-5.5.4/doc/getting_started/part_frame.html

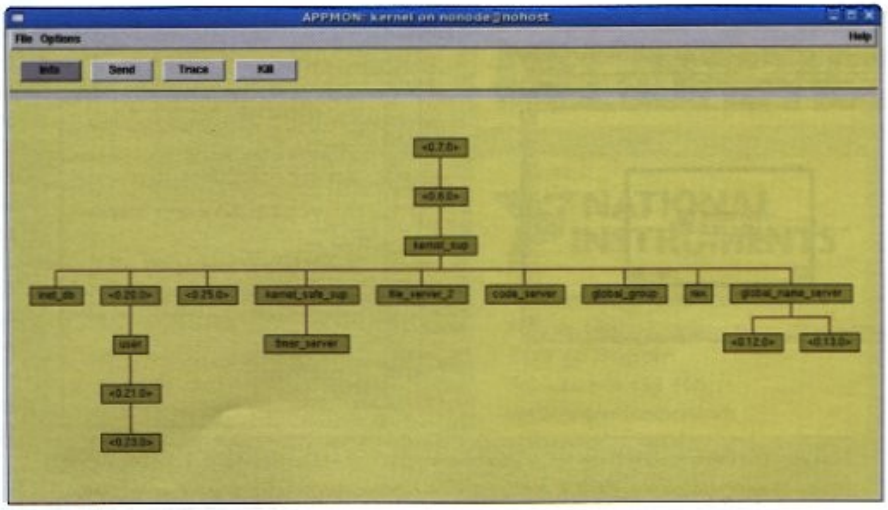
Description Language is ook een internationale (telecom) standaard.
Vervolgens werd geprobeerd Erlang zelfstandig op de markt te brengen als ontwikkeltaal en -omgeving maar dat had minder succes. De redenen zijn nog steeds niet helemaal duidelijk (zie ref. 1) maar als bedrijfseigen taal van Ericsson was het waarschijnlijk minder interessant voor andere telecombedrijven, terwijl anderzijds misschien onzekerheid bestond over de ondersteuning. Tenslotte is Ericsson primair een telecombedrijf en geen leverancier van ontwikkelsoftware.
In 1998 werd Erlang vrijgegeven als open-source product, naast de commerciële versie van Ericsson, en die situatie bestaat nog steeds.

Concurrent

Het bijzondere van Erlang is dat het declaratief programmeren combineert met 'light-weight' processen die kunnen communiceren en samenwerken door 'message passing'. Het is in Erlang heel eenvoudig een proces te implementeren (zie Listing 2), al dan niet terminerend en al dan niet met timeouts. Deze processen worden intern afgehandeld en elk proces heeft zijn eigen 'garbage collection'. Bovendien ondersteunt Erlang processen die gedistribueerd zijn over verschillende 'nodes', waardoor een programma op twee processoren ook echt tweemaal zo snel kan zijn, volgens de documentatie. Processen kunnen als client-server functioneren en kunnen een toezichthoudende ('supervisory') rol hebben.
Het derde onderscheidende kenmerk is de ingebouwde foutafhandeling in Erlang. Een proces kan normaal of abnormaal termineren en kan, als het termineert, een signaal sturen naar andere processen die er door een 'link' mee verbonden zijn. De programmeur kan die links eenvoudig aangeven, en de verbonden processen kunnen worden geprogrammeerd om actie te ondernemen. Een abnormaal terminerend proces kan, bijvoorbeeld door een toezichthoudend proces, opnieuw worden opgestart.

Erlang ondersteunt ook 'hot code replacement' - vervanging van oude code door nieuwe zonder dat het systeem hoeft te worden stilgelegd.
Er zijn, buiten Ericsson, vijftien ondernemingen die Erlang gebruiken en zeven onderzoeksinstellingen en universiteiten. Op de websites zijn diverse projecten en applicatievoorbeelden te vinden.
Erlang is, evenals Java, een geïnterpreteerde taal en daardoor (min of meer) onafhankelijk van het besturingssysteem. De open-source versie is op dit moment beschikbaar voor Solaris, Windows, Linux en VxWorks (zie www.erlang.org). Over de door Ericsson ondersteunde commerciële versie is meer informatie beschikbaar op www.erlang.se. ■

Referenties
1. Bjarne Däcker, Thesis, Stockholm, 2000
2. J. Armstrong et al., 'Programming Erlang, Software for a Concurrent World', (Prentice Hall, deel 1 beschikbaar (pdf) op www.erlang.org)
3. www.erlang.se



Figuur 2. Een van de opties van de grafische toolbar, de 'Application Monitor'.