Język programowania Erlang

Piotr Pałach

PWSZ w Tarnowie

Tarnów, 19 stycznia 2009

Spis treści

Informacje podstawowe

Opis Języka

Erlang został stworzony w laboratorium firmy Ericsson i jest z powodzeniem stosowany w projektach wymagających wysokiej skalowalności i odporności na awarie. Jest to język funkcyjny, dynamiczny, o ścisłym typowaniu, w którym główny nacisk położono na współbieżność oraz możliwości przetwarzania rozproszonego. Erlang był własnościowym językiem programowania używanym przez Ericssona, jednak od roku 1998 kiedy kod źródłowy został otwarty, zyskuje z dnia na dzień na popularności. Z powodzeniem używany jest w komercyjnych produktach na całym świecie, które wykraczają niejednokrotnie poza jego telekomunikacyjne korzenie.

Nazwa

Nazwa Erlang została nadana na cześć pierwszego naukowca w dziedzinie telekomunikacji: Agnera Krarupa Erlanga (1878-1929), matematyka dunskego, pioniera teorii ruchu telekomunikacyjnego oraz teorii kolejek. Czasami nazwa języka bywa interpretowana jako Ericsson LANGuage.

Cechy Erlanga

Współbieżność

Procesy są podstawowymi elementami w strukturze aplikacji napisanych w Erlangu. Implementacja współbieżnych procesów jest niezależna od systemu operacyjnego i nie używa specyficznych dla danego środowiska mechanizmów jak na przykład wątki. Tworzenie i zarządzanie erlangowymi procesami jest bardzo proste, podczas gdy w innych językach współbieżność dostarcza wielu problemów oraz jest źródłem nie małej ilości różnego rodzaju błędów. Erlangowe procesy są dużo lżejsze od systemowych(ok. 300 bajtów na jeden proces), dzięki czemu operacje tworzenia i niszczenia procesów są relatywnie tanie obliczeniowo i pamięciowo. W pojedynczym erlangowym systemie możliwe jest utworzenie nawet milion procesów bez znaczącego obniżenia wydajności.

W Computer Language Shootout w konkursie na przesyłanie wiadomości pomiędzy tysiącami wątkow (thread-ring) rozwiązanie napisane w Erlangu zajmuje pierwsze miejsce pod względem wydajności oraz objętości kodu.

Cechy Erlanga

Komunikacja pomiędzy procesami w Erlangu(z racji tego, że procesy nie dzielą między sobą wspólnej pamięci) odbywa się poprzez przesyłanie asynchronicznych komunikatów. Z każdym procesem powiązana jest kolejka FIFO, służąca do odbioru komunikatów. Proces odbiorczy decyduje o tym, czy dany komunikat przetworzyć, czy nie. Wszystko to jest Każdy proces ma swoją skrzynkę (ang. mailbox), w postaci kolejki, w której są przechowywane wiadomości wysłane przez inne procesy dopóki nie zostaną odczytane. Odbieranie wiadomości odbywa się przez mechanizm dopasowania wzorca. Po odczytaniu wiadomości proces Erlanga wraca do wykonywania. Do utworzenia wiadomości może zostać użyta dowolna struktura danych(liczby całkowite, zmiennoprzecinkowe, znaki, atomy), a nawet funkcje.

Cechy Erlanga

Rozposzoność

W język wbudowany jest mechanizm rozproszenia. Można tworzyć procesy z dowolnego węzła na dowolnym węźle. Proces może zostać utworzony na zdalnym węźle, a komunikacja z nim jest przezroczysta (tzn. komunikację z zdalnym procesem przeprowadza się w dokładnie takim sam sposób jak z procesem lokalnym).

Obsługa błędów

Obsługa błędów odbywa się na ogół przez nadzór jednych procesów nad innymi. Kiedy proces się zepsuje, wychodzi i wysyła wiadomość do procesu kontrolnego, który może podjąć odpowiednią akcje - np. restart bądź zakończyć zepsuty proces i uruchomić następny. Takie podejście do programowania zwiększa niezawodność i czyni kod mniej skomplikowanym. Ponadto proces kontrolny i proces nadzorowany nie muszą znajdować się na tej samej fizycznie maszynie, dzięki czemu stosunkowo łatwo daje się programować systemy wysokiej dostępności, w których poszczególne funkcje wykonywane są przez sprawne w danym momencie węzły.

Cechy Erlanga

Gorąca wymiana kodu(Hot code upgrade)

Wiele systemów nie może być zatrzymana w celach wymiany oprogramowania. W Erlangu można wymieniać kod w uruchomionym systemie(w systemie koegzystują przez chwile oba kody), a tym samym instalować poprawki błędów oraz uaktualnienia bez zatrzymywania działania systemu.

Obsługa systemów wieloprocesorowych

Od wersji R11 środowisko Erlanga potrafi automatycznie wykorzystać potencjał ukryty w procesorach wielordzeniowych (multi-core) oraz systemach wieloprocesorowych. Wcześniej aby wykorzystać taki system należało uruchomić kilka kopii maszyny wirtualnej, co niestety pociągało zwiększenie zużycia pamięci, oraz zwiększało niepotrzebnie koszty przesyłania wiadomości pomiędzy procesami na jednym komputerze.

Zastosowania języka Erlang

Najpopularniejsze aplikacje napisane w erlangu

- Mnesia, rozproszona, odporna na błędy baza danych czasu rzeczywistego, rozprowadzana wraz z Erlangiem
- Ejabberd, serwer Jabber napisany w Erlangu, który jest uznawany za bardzo stabilny i wysoko skalowalny
- Yet another web server, wysoko wydajny serwer stron Web który w wielu testach generowania stron dynamicznych przegania Apacha
- Tsung, wysoko wydajne narzędzie diagnostyczne do przeprowadzania testów wydajności
- oraz wiele innych aplikacji zarówno klienckich jak i również serwerowych

Zastosowania języka erlang pokazują, że ten język jest bardzo dobrze sprawdzoną i stabilną platformą do rozwoju oprogramowania. Obecnie Erlang używany jest przez kilka ogólnoświatowych firm telekomunikacyjnych takich jak: Nortel, T-Mobile i Telia.

Typy danych

Erlang jest językiem z dynamicznym typowaniem oraz statycznym pojedynczym przypisaniem. Oznacza to, iż nie określamy jawnie typu zmiennej, jednak wartość zmienne musimy już jawnie określić. Ponadto w przeciwieństwie do wielu języków jak np. C++, czy Java raz przypisana wartość do zmiennej nie może zostać później zmieniona. Nową zmienną tworzy się poprzez jej pierwsze użycie, kiedy zostaje przypisana jej wartość.

Podstawowe typy danych

- Liczba (całkowita lub zmiennoprzecinkowa)
- Atom
- Tuple
- Lista
- pozostałe typy

Typy danych

Liczba całkowita lub zmiennoprzecinkowa

Liczby w Erlangu są konstruowane jak w wielu innych językach, przy pomocy cyfr. Liczby całkowite mają nieograniczony zakres(jedyne ograniczenie - wielkość pamięci).

```
-15
123.1233
```

Atom

Atom jest to literał, którego nazwa zaczyna się z małej litery. Jeżeli atom zaczyna się z dużej litery, lub zawiera spacje musi być ujęty w pojedynczy apostrof '.

```
pierwszy
drugi_atom
'Trzeci atom'
```

Typy danych

Tupte

Tuple składa się ze stałej liczby elementów otoczonych parą nawiasów klamrowych ({ i }) . Element tupla może się składać z liczby, bądź cyfry. Dowolny elementy tupla może być łatwo dostępny, szczególnie przy użyciu wzorców. Tuple możemy manipulować za pomocą funkcji BIF - Build In Function(funkcje które zostały wbudowane w wirtualną maszynę Erlang).

```
1> T1 = {piotr,23,{luty,29}}.
{piotr,23,{luty,29}}
2> element(1,T1}.
piotr
3> element(3,T1).
{luty,29}
4> {Imie,Lata,Data_urodz} = T1.
{piotr,23,{luty,29}}
5> Imie.
piotr
```

Typy danych

Lista

Lista może zawierać zmienną liczbę elementów oddzielonych przecinkami, które otoczone są parą kwadratowych nawiasów [i]. Ciąg znaków otoczonych cudzysłowem również jest listą, a jej elementami są litery. Możemy utworzyć listę z elementami różnych typów, ze względu na to że Erlang jest językiem dynamicznym, w którym występuje nieścisłe typowanie. Dostęp do elementów listy jest realizowany przy pomocy dostępu do głowy listy (head), oraz ogona (tail).

```
1>  L1 = [b,1,{d,2 }].
[b,1,{d,2 }].
2> H.
b
3> T.
[1,{d,2 }].
4> L2 = [d|T].
[d,b,1,{d,2 }].
```

Typy danych

Filtrowanie danych listy

```
1> L = [1,2,3,4,5].
[1,2,3,4,5]
2> [E +1 || E <-  L].
[2,3,4,5,6]
3> [E + 1000 || E <-  L, E =/= 1, E < 3 orelse E > 4].
[1002,1005]
4> [E + 1 || E <- [1,"Kitten","Batman",5], is_integer(E) ].
[2,6]
5> [ A || {A,B} <- [{a,b},{c,d},1,tomato,{e,f}] ].
[a,c,e]
```

Dwie pierwsze operacje na listach sprowadzają się do wykonania akcji dla każdego E w liście L gdzie warunek jest spełniony. Ostatnie dwie instrukcje pokazuja że filtrowanie danych na liście, może być wykonywane dla różnych typów elementów.

Typy danych

Pozostałe typy

- pid procesu - potrzebny do przekazywania komunikatów, albo sprawdzania stanu procesu
- port -służy do komunikacji przy pomocy komunikatów ale z elementami poza językiem Erlang, np. programem C, potokiem UNIX, czy socketem TCP
- rekord - tuple o specjalnej konstrukcji
- boolean - wartość logiczna true lub false

Prosty przykład - Silnia

Wzór silni

```
n! = 1,  if n = 0,
n! = n * (n - 1)!,  if n > 0
```

Zapis silni w erlangu

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

Forma klauzulowa

W erlangu do zapisu wyrażenia warunkowego wykorzystano tak zwaną formę klauzulową. Składa się ona z pewnej liczby klauzul opisujących przypadki związane z wyznaczaniem wartości funkcji. Poszczególne klauzule pisane są jedna pod drugą, rozdzielone są znakiem średnika a ostatnia zakończona jest kropką. Kiedy funkcja o takiej postaci jest wykonywana wybierana jest jedna klauzula, pierwsza z góry, która spełnia warunek. Wybór następuje na podstawie związanych z nimi warunków. Liczy się przy tym kolejność definicji klauzul.

Prosty przykład - Silnia

Przykład formy klazulowej

```
funkcja(argumenty1) -> ciało1; % klauzula1
funkcja(argumenty2) -> ciało2; % klauzula2
...
funkcja(argumentyN) -> ciałoN. % klauzulaN
```

Dopasownie do wzorca

Podstawowym sposobem dostarczania warunków jest mechanizm dopasowywania do wzorca. Dopasowanie do wzorca polega na sprawdzeniu czy oba typy argumentów do siebie pasują. W rozważanym przypadku, liczby pasują gdy są sobie równe, zmienne zaś pasują do każdej liczby przejmując jednocześnie jej wartość. W przypadku silni jeśli argumentem wywołania funkcji będzie 0 zostanie ono dopasowane do argumentu formalnego pierwszej klauzuli i zwrócona zostanie wartość 1. W przeciwnym przypadku zadziała klauzula 2.

Prosty przykład - Silnia

Wszystkie funkcje w Erlangu powinny być definiowane w modułach. Moduł prosteFunk.erl zawiera zdefiniowaną silnię wraz z odpowiednimi nagłówkami

Definicja modułu prosteFunk oraz funkcji silnia

```
-module(prosteFunk).
-export([silnia/1]).

silnia(0) -> 1;
silnia(N) when N>0 -> silnia(N-1) * N.
```

module(prosteFunk) - nazwa modułu zgodną z nazwą pliku
export([silnia/1]) - nazwa funkcji, oraz liczba argumentów(funkcja silnia posiada jeden argument)

Prosty przykład - Quicksort

Defnicja funkcji Quicksort

```
-module(prosteFunk).
-export([quicksort /1]).

quicksort ([]) -> [];
 quicksort([Pivot|T]) -> quicksort ([X||X <- T, X < Pivot]) ++
    [Pivot] ++  quicksort ([X||X <- T, X >= Pivot]).
```

Powyższa definicja funkcji Quicksort wywołuje rekurencyjnie funkcje quicksort dopóki wszystkie elementy nie zostały posortowane.
Wyrażenie `[X||X <- T, X < Pivot]` oznacza: dla każdego X który należy do ogona listy, który nie zawiera już pierwszego elementu, i takich X które są mniejsze od elementu Pivot(element podziału) wykonaj rekurencyjnie sortowanie tego zbioru.
Operator ++ ma za zadanie połączyć powstałą po sortowaniu listę

Procesy w Erlangu

Kilka słów o procesach

Większość równocześnie działających funkcji w Erlangu tworzy się bardzo prosto za pomocą funkcji:
`spawn(Module,Fun,Args)` `spawn(fun() -> loop() end)`
która zwraca identyfikator procesu, używany do komunikacji z nowo utworzonym procesem.
Wyrażenie `Pid ! Wiad` wysyła wiadomość do konkretnego procesu za pomocą identyfikatora procesu - Pid. Wiadomość jest odbierana używając konstrukcji `receive ... end`, która rozpoznaje rodzaj wiadomości otrzymanej za pomocą klazul które zostały zadeklarowane.

Prosty przykład - wypisywanie informacji, gorąca wymiana kodu, czat

Wypisywanie informacji

Proces wypisujący informację w zależności od dopasowania wzorca.

Gorąca wymiana kodu

Podmiana kodu programu bez zatrzymywania systemu.

Chat

Program chat wypisujący informacje przesłane przez jednego użytkownika każdemu użytkownikowi który aktualnie znajduje się w pokoju.

Źródła prezentacji

- `http://erlang.org/`
  Strona domowa języka Erlang
- `http://erlang.org/doc/pdf/`
  Pełna Documentacja
- `http://erlang.org/white_paper.html`
  Wprowadzenie do Erlanga z przykładami
- `http://www.erlang.pl/doku.php`
  Główna polska strona języka Erlang

Dziekuję za uwagę!!!