

Kooperace s interpretem

Je možné kombinovat volání přeložených funkcí ze zkompilovaného rozšíření s funkcemi v interpretovaném skriptu. To umožňuje překlad vybraných částí a zanechává možnost flexibility změn bez nutnosti překompilovat celou aplikaci.

```
function main($a, $b)
{
    return user_defined_function($a, $b);
}
```

Detekce datových typů

Z důvodu překladu do silně typovaného jazyka, kterým C++ je, se provádí detekce datových typů proměnných. Pro určení jsou analyzovány prováděné operace a volané funkce.

```
function main($a,$b = 19) {
    $str_test1 = "1234";
    $str_test2 = "aaa".(5*5);
    $str_test3 = (4*5)."aaa".(5*3);

    $int_test1 = 15;
    $int_test2 = 16+4;
    $int_test3 = "5"-4;
    $int_test4 = 19*48;

    $float_test1 = $a*$b;
    $float_test2 = 4.5*8;
    $float_test3 = 6**4;
    $float_test4 = 72/4;
    $float_test5 = 15;
    $float_test5 = $float_test5 / 4;
    $float_test6 = "4.5"+5;
}
```

Překlad & kompilace

```
$php -a
Interactive shell
```

```
php > function user_defined_function($a, $b) { return $a+$b*5; }
php > var_dump(compiled_main(15, 8));
int(55)
```

```
Php::Value phpFunc_compiled_main(Php::Parameters &args)
{
    double phpVar_a = args[0];
    double phpVar_b; if(args.size() > 1) phpVar_b = args[1]; else phpVar_b = 19 ;
    std::string phpVar_str_test1 = "1234";
    std::string phpVar_str_test2 = std::string( php2cpp::to_string("aaa")).append( php2cpp::to_string( ( pow(php2cpp::to_float(5),php2cpp::to_float(5)) ) ));
    std::string phpVar_str_test3 = std::string( php2cpp::to_string( std::string( php2cpp::to_string( ( 4 * 5 ) ) ).append( php2cpp::to_string("aaa") ) ) ).append( php2cpp::to_string( ( 5 * 3 ) ) ));
    long phpVar_int_test1 = 15;
    long phpVar_int_test2 = 16 + 4;
    double phpVar_int_test3 = php2cpp::to_float(5) - php2cpp::to_float(4);
    long phpVar_int_test4 = 19 * 48;
    double phpVar_float_test1 = php2cpp::to_float( phpVar_a ) * php2cpp::to_float( phpVar_b );
    double phpVar_float_test2 = php2cpp::to_float(4.5) * php2cpp::to_float(8);
    double phpVar_float_test3 = pow(php2cpp::to_float(6),php2cpp::to_float(4));
    double phpVar_float_test4 = php2cpp::to_float(72) / php2cpp::to_float(4);
    double phpVar_float_test5 = 15;
    phpVar_float_test5 = php2cpp::to_float( phpVar_float_test5 ) / php2cpp::to_float(4) ;
    double phpVar_float_test6 = php2cpp::to_float(4.5) + php2cpp::to_float(5) ;

    return nullptr;
}
```

Zrychlení

Na grafu můžeme vidět hodnoty ukazující dobu potřebnou pro seřazení stejného pole o 50 000 náhodných číslech algoritmem Bubble sort. Výrazný rozdíl časů - 3.42s a 0.24s v prospěch pro C++ naznačuje, že je možné dosáhnout výrazné optimalizace. V tomto případě zrychlení 14.24x. Obdobných výsledků hovořících v prospěch C++ verze zavedené jako rozšíření dosáhly další experimenty. Uvedu dále například součet všech čísel od 1 do 100 000 000 000, kdy naměřená čísla jsou ještě více rozdílná - 73s PHP a 0.35s C++. Tyto testy byly prováděny na počítači s OS Fedora 21 64bit, Intel i7 4702MQ, ReiserFS naSSD. Překladč GCC 4.9.1 s volbou optimalizace -O3. Měření bylo provedeno 50x a vypočten matematický průměr. Žádná z naměřených hodnot nevybočovala výrazněji z průměru.



Kód

Zdrojové kódy jsou zveřejněny pod licenci Apache 2.0 na adrese:

<https://github.com/nechutny/BP>