OBJEKTŲ IR MASYVŲ NAUDOJIMO PHP KALBOJE NAŠUMO TYRIMAS

Tomas Šeirys, Simona Ramanauskaitė
Šiaulių universitetas, Technologijos fakultetas

Įvadas

Nuolat sudėtingėjančios programinės sistemos reikalauja vis daugiau sisteminių resursų, kurių trūkumas gali sukelti kritinių problemų arba visiškai nutraukti programos darbą. Tad norint kuo veiksmingiau išnaudoti sisteminio bloko resursus, pirmiausia reikėtų naudoti kuo mažiau programinės įrangos resursų. Vienas programos resursų naudojimo mažinimo būdų yra atsižvelgti į programos vykdomą kodą ir kaip galima geriau jį optimizuoti. Tai padėtų ne tik veiksmingiau naudoti sistemos resursus, bet ir suteiktų programai stabilumo, paspartėtų programos veikimas.

Tyrimo tikslas – ištirti PHP programavimo kalboje naudojamų objektų ir masyvų savybes resursų naudojimo aspektu.

Pagrindinės PHP kalboje naudojamos duomenų struktūros

PHP (*Hypertext Preprocessor*) – plačiai paplitusi dinaminė interpretuojama programavimo kalba, sukurta 1995 m. ir specialiai pritaikyta interneto svetainėms kurti. PHP sintaksė panaši į daugelį struktūrinių kalbų, ypač į C bei *Perl*. PHP kalba yra atviro kodo ir tai yra viena priežasčių, dėl ko kalba yra nors ir nesudėtinga, bet gana lanksti – veikia daugumoje operacinių sistemų, palaiko nemažai reliacinių duomenų bazių ir veikia su dauguma interneto serverių – CGI, *FastCGI*, ISAPI ir kitais protokolais. PHP yra dažniausiai naudojama interneto puslapiams kurti, ji yra ir labai galingas įrankis atlikti kitas funkcijas komandinėje eilutėje.

Be paprastų kintamųjų, PHP ir kitose programavimo kalbose dažnai tenka saugoti ir sudėtingesnės struktūros kintamuosius. Masyvai ir objektai yra skirti saugoti keletą reikšmių viename kintamajame (Liesis, 2013). Paprastas kintamasis savyje saugo vieną reikšmę, tuo tarpu masyvas ar objektas yra toks kintamojo tipas, kurio viduje galima saugoti keletą reikšmių.

Masyvas – tai indeksuotas kintamasis, galintis turėti ne vieną, bet kelias reikšmes. Kiekvienai masyvo reikšmei priskiriamas indeksas. Pagal nutylėjimą masyvo indeksui yra priskiriama skaitinė reikšmė (*Integer*), tačiau PHP kodo skaitymui palengvinti masyvo indeksui būna priskiriama simbolinė reikšmė (*String*).

Objektas – tai savarankiškas kintamasis, viduje galintis turėti tik tam objektui galiojančių funkcijų ir tik jam būdingą kintamųjų kombinaciją. Objekto kintamieji skiriasi nuo masyvo kintamųjų tuo, kad jie nėra indeksuojami.

Egzistuojantys programų našumo tyrimai

Programinio kodo efektyvumas yra aktuali tema tiek mokslininkams, kurie siekia optimizuoti savo siūlomus algoritmus ar naujai kuriamas kalbas, tiek ir pramonės atstovams, kurie visuomet yra suinteresuoti gauti kuo greičiau veikiančią, mažiau sistemos resursų naudojančią sistemą. J. J. Cohen (2012) pateikia savo mintis, kodėl programos veiksmingumas turėtų būti svarbus visiems, ir pateikia patarimų, kaip reikėtų rašyti kodą, norint pasiekti, kad jis būtų veiksmingai vykdomas. Nors šiame darbe pateikiama patarimų, gali būti naudojami skirtingose programavimo kalbose, bet pats tyrimas atliktas su SAS technologija, todėl kito tipo sistemose galimi visiškai kitokie rezultatai.

Konkrečios programavimo kalbos ir technologijos našumo savybėms ištirti buvo atlikta ir kitų tyrimų, pavyzdžiui, D. Lindbo (2010) pateikia patarimų, kaip efektyviai programuoti *Matlab* paketą, D. G. Moss (2001) savo apskaitų medžiagoje tiria C ir C++ programavimo efektyvumą, G. L. Taboada ir kt. (2009) aprašo, kaip *Java* kalba gali būti taikoma efektyviems skaičiavimams. Tačiau pažymėtina, kad didžioji dalis tyrimų tiria, kaip skiriasi tam tikrų funkcijų atlikimas skirtingose programavimo kalbose: S. Trent ir kt. (2008) tyrė, kokie yra PHP ir JSP skirtumai; R. Eggen ir kt. (2008) lygino *Ruby*, PHP, *Perl* ir *Python* kalbų našumą, o S. Gilmore (2007) – C, *Java*, C#, *Cyclone* ir *Caml* kalbas.

Todėl galima teigti, kad programavimo kalbų našumo vertinimo tyrimų yra nemažai, tačiau tyrimo, kuriame aiškiai būtų įvertinta, ką PHP kalboje geriau naudoti – masyvus ar objektus, nepavyko rasti.

PHP kodo našumo įvertinimo metodika

Siekiant įvertinti masyvų ir objektų naudojimo PHP kalboje našumą, tyrimui pasirinktos 6 pagrindinės situacijos: masyvo kūrimas, objekto kūrimas, įrašymas į masyvą, įrašymas į objektą, kreipimasis į masyvą, kreipimasis į objektą. Šie šeši veiksmai yra pagrindiniai darbui su masyvais ir objektais. Tačiau juos naudojant PHP palaiko skirtingo tipo indeksus ir duomenų tipus, todėl visi testai vykdomi naudojant sveikojo tipo (*Integer*) ir tekstinio tipo (*String*) indeksus. Tokiu būdu tyrimas bus išsamesnis ir atskleis, kada kurio tipo kintamieji yra tinkamesnio kodo našumui padidinti.

Viso tyrimo metu fiksuojami du pagrindiniai našumo parametrai – kodo vykdymo laikas ir jo metu naudojamos atminties kiekis. Šiems parametrams nustatyti nereikalinga papildoma programinė įranga, o tiesiog prieš testuojamą kodo dalį yra įvertinamas dabartinis laikas ir tuo metu išnaudota atmintis, o baigus vykdyti testuojamą kodą vėl fiksuojami tie patys parametrai ir nustatomas jų skirtumas (1 pav.).

Tyrimo metu naudojama PHP 5.3.17 versija. Praktinio testavimo metu testas atliekamas 4 kartus, didinant masyvo elementų ar objekto kintamųjų skaičių (1 pav. naudojamas kintamasis *$valuesCount*) kas 30 000 nuo 10 000 iki 100 000 (10 000, 40 000, 70 000, 100 000). Toks elementų kiekio pasirinkimas leidžia įvertinti ir mažas, ir gana dideles duomenų struktūras.

```
1. $arrayGenerationTime_start = microtime(true);
2. $arrayGenerationMem_start = memory_get_usage();
3.
4. for($i = 0; $i < $valuesCount; $i++) {
5.   $test_array[$i] = NULL;
6. }
7.
8. $arrayGenerationTime = microtime(true) - $arrayGenerationTime_start;
9. $arrayGenerationMem = memory_get_usage() - $arrayGenerationMem_start;
```

1 pav. Masyvo kūrimo pavyzdys su jo kūrimui sugaišto laiko ir sunaudotos atminties nustatymu

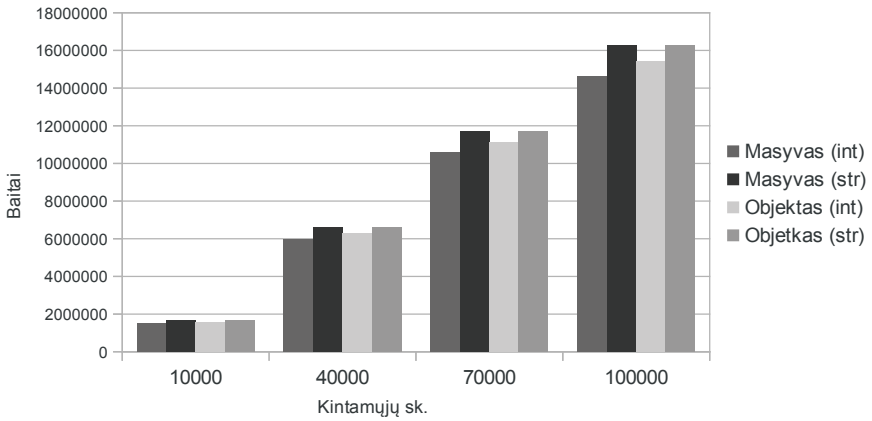
Taip pat tyrimo objektyvumui padidinti kiekvienas testas kartojamas bent po 3 kartus. Tie rezultatai, kurie labai išsiskiria iš kitų to testo rezultatų, yra eliminuojami kaip netinkami dėl per didelės sistemos apkrovos ar kitų priežasčių, o fiksuojamas tik pernelyg nesiskiriančių rezultatų vidurkis.

Našumo tyrimų rezultatai

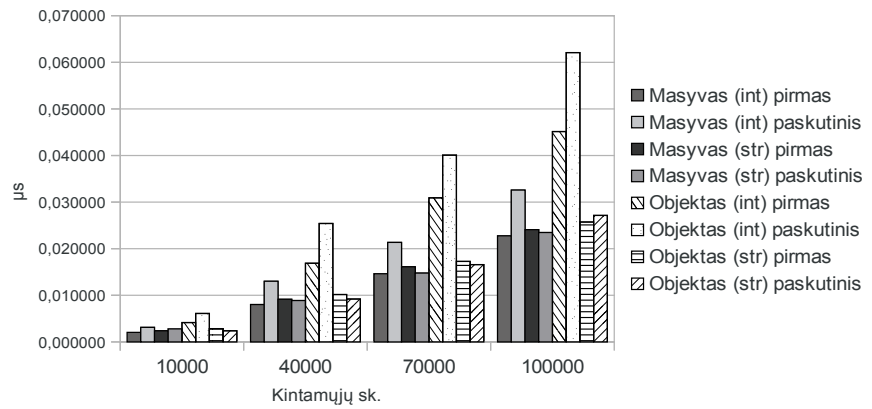
Analizuojant PHP kodo našumą užpildant masyvą tuščiais (NULL) duomenimis pastebėta, kad mažiausiai atminties reikalauja *Integer* tipo indeksus turintis masyvas, *Integer* tipo objektui atminties reikėjo šiek tiek daugiau, o *String* tipo masyvų ir objektų naudojamas atminties kiekis yra beveik identiškas (2 pav.).

Vertinant tos pačios užduoties vykdymo laiką (3 pav.), pastebėtas akivaizdžiai spartesnis masyvo kūrimo greitis, kai indekso tipas yra *Integer*, kai kur net iki 18 kartų sparčiau už *String* tipo indeksą. Tuo tarpu objektų, turinčių *Integer* tipo kintamąjį, kūrimas buvo lėtesnis nei masyvo. Objekto, kurio kintamųjų vardai buvo *String* tipo, ir masyvas, kurio kintamieji buvo indeksuojami *String* tipu, kūrimo greitis buvo beveik identiškas.

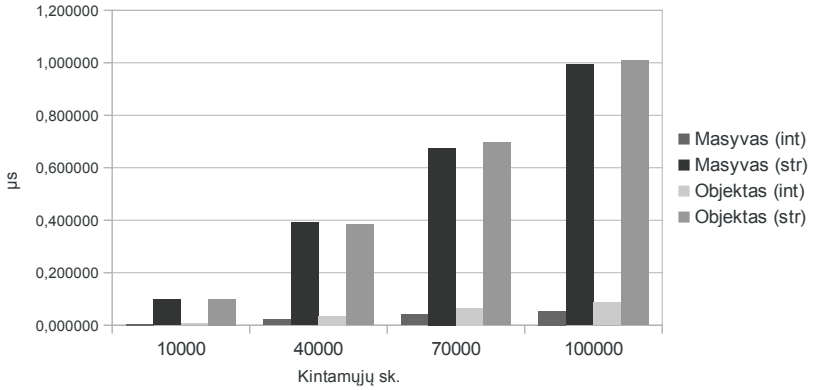
Tirdami užpildyto masyvo ir užpildyto objekto atminties sunaudojimą, gavome rezultatą beveik identišką, kaip ir neužpildytų duomenų struktūrų. Tai leidžia daryti prielaidą, kad tyrime informacijos adresavimas užima daugiau atminties nei pati saugojama



2 pav. Neužpildyto objekto ir masyvo dydžių palyginimas



3 pav. Objekto ir masyvo kūrimo greičio palyginimas



4 pav. Užpildyto objekto ir masyvo nuskaitymo palyginimas

informacija. Tyrimo metu kintamieji saugojo po 20 atsitiktinių simbolių.

Bandydami nuskaityti duomenis iš duomenų struktūrų, pamatėme masyvo su *Integer* tipo indeksu pranašumą (4 pav.). Įdomu tai, kad paskutinį masyvo elementą, turintį *String* tipo indeksą, nuskaito nors ir nežymiai, tačiau greičiau už *Integer* tipo indeksą, nes tai pasikartojo visuose bandymuose, to negalima ignoruoti kaip paklaidos sukeltą anomaliją. Taip pat pažymėtina, kad objekte kintamųjų vardams suteikti *Integer* tipo reikšmes yra labai neveiksminga. O masyvas, indeksuotas *String* tipo reikšmėmis, ir objektas, kurio kintamieji pavadinti *String* tipo vardais, davė panašius rezultatus.

Išvados

1. Neužpildytų objekto ir masyvo atminties sunaudojimas yra beveik vienodas. Masyvas, indeksuotas *Integer* tipo reikšmėmis, naudoja mažiausiai atminties, bet skirtumas yra labai nedidelis, tad teigtina, jog ir masyvo, ir objekto atminties sunaudojimas yra beveik identiškas.
2. Objektų ir masyvų kūrimo greitis taip pat yra beveik identiškas. Tačiau, palyginus spartumą atsižvelgus į indekso ir kintamojo vardo tipą, pastebėtina, kad *Integer* tipo indeksus ir kintamųjų vardus priskiria greičiausiai, vietomis netgi iki 18 kartų sparčiau nei *String* tipo indeksus. Tokiam išsiskyrimui įtakos galėjo turėti tai, kad *String* tipo kintamuosius reikėjo sukurti.
3. Tiriant nuskaitymo veiksmingumą, paaiškėjo, jog objekto kintamiesiems suteikti *Integer* tipo pavadinimus yra labai neveiksminga. Masyvas, indeksuotas *String* tipo indeksu, ir objektas, kurio kintamiesiems suteiktas *String* tipo pavadinimas, davė labai panašius rezultatus. Paskutinis masyvo elementas, kai masyvas indeksuojamas *Integer* reikšmėmis, nuskaitomas lėčiau nei masyvas, indeksuotas *String* tipo reikšmėmis. Šios anomalijos priežasties nustatyti nepavyko.

Literatūra

1. Cohen J. J., 2011, *Why Programming Efficiency Should Matter to All of Us*. Prieiga per internetą: http://www.nesug.org/ Proceedings/nesug11/ds/ds08.pdf.
2. *developerWorks. Writing efficient PHP*. Prieiga per internetą: http://www.stevengould.org/portfolio/developerWorks/efficientPHP/wa-effphp/wa-effphp-a4.pdf.
3. Eggen R., Jones C., Eggen M., 2008, *Ruby, PHP, Perl, Python: A Web Efficiency Compatison.* Prieiga per internetą: http://www.unf.edu/~ree/Eggen-vita.pdf.
4. Gilmore S., 2007, *Advances in Programming Languages: Efficiency*. Prieiga per internetą: http://homepages.inf.ed.ac.uk/stg/ teaching/apl/handouts/efficiency.pdf.
5. Liesis L., 2013, *PHP masyvai*. Prieiga per internetą: http://kodai.manualai.lt/php/pagrindai/masyvai.html.
6. Lindbo D., 2010, *Notes on efficient Matlab programming*. Prieiga per internetą: http://www.csc.kth.se/utbildning/kth/kurser/DN1240/ numi13/matlab_perf.pdf.
7. Moss D. G., 2001, *Efficient C/C++ Coding Techniques*. Prieiga per internetą: http://www.open-std.org/jtc1/sc22/wg21/docs/ESC_ Boston_01_304_paper.pdf.
8. Taboada G. L., Touriño J., Doallo R., 2009, *Java for High Performance Computing: Assessment of Current Research and Practice*. Prieiga per internetą: http://www.des.udc.es/~gltaboada/papers/taboada_pppj09.pdf.
9. Trent S., Tatsubori M., Suzumura T., Tozawa A., Onodera T., 2008, *Performance Comparison of PHP and JSP as Server-Side Scripting Languages*. Prieiga per internetą: http://www.research.ibm.com /trl/people/mich/pub/ 200812_ middleware2008specweb.pdf..

OBJECT AND ARRAY USAGE EFFICIENCY IN PHP LANGUAGE

Tomas Šeirys, Simona Ramanauskaitė

Summary

While developing complex software systems that perform big data computations, program execution time and memory consumption should be taken into consideration . The more efficient utilization of the system characteristic , the faster and more efficient the program execution time and memory occupation will be. The better the code is optimized, the better results of program execution time we will get.

This study determines which data structure is more efficiently to use in PHP language: an array or an object. There is a comparison of array and object execution time as well as memory consumption. The results will help us to optimize and make more efficient use of the system in which a PHP written program will run.

Key words: PHP, optimization, object, array, memory, execution time, string, integer.

OBJEKTŲ IR MASYVŲ NAUDOJIMO PHP KALBOJE NAŠUMO TYRIMAS

Tomas Šeirys, Simona Ramanauskaitė

Santrauka

Kuriant sudėtingas ir daug skaičiavimų atliekančias programines sistemas, programose reikia atkreipti ypatingą dėmesį į vykdymo laiką ir atminties sunaudojimą. Kuo veiksmingiau bus išnaudojama sisteminio bloko charakteristika, tuo spartesnis ir veiksmingesnis bus programos vykdymo laikas ir atminties užimtumas. Vienas iš programos greitaveikos efektyvumo didinimo būdų yra programinio kodo optimizavimas. Šiame tyrime nustatoma, kokią duomenų struktūrą– masyvą ar objektą – PHP kalboje yra veiksmingiau naudoti, lyginama masyvo ir objekto greitaveika, atminties sunaudojimo dydžiai. Tyrimo rezultatai padės labiau optimizuoti ir veiksmingiau išnaudoti sisteminį bloką, kuriame bus vykdoma PHP kalba parašyta programa.

Prasminiai žodžiai: PHP, optimizacija, objektas, masyvas, atmintis, greitaveika.

Įteikta 2013-05-30