Testování webových aplikací

Jakub Vavřík

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

K čemu to je?

- Testování výkonu je spíše umění než věda
- Opakovaný proces
  - Webová aplikace příliš komplexní
  - Mnoho ovlivňujících faktorů
    - Nutnost postupně modifikovat
    - Zjistit hlavní vlivy a dle potřeby modifikovat
    - Provést nová měření s modifikovanými podmínkami
    - ! to celé stále dokola

Není test jako test – testujeme zátěž

- Výkonostní testování jako široký pojem
  - **Performance test** – white box testing
    - Stanoveni hodnot pro regresni testing
  - **Load test –** black box testing
    - Pozdější fáze pro regresni testovani
  - **Stress test –** destruktivní testování
    - Koncová fáze vývoje

Performance testing

- měříme, monitorujeme, zkoumáme, divíme se<
- Neslouží pro odhalení chyb ale pro
  - Nalezení úzkých míst
  - Určení hodnot pro pozdější regresní testování
- Je třeba mít definovaný cíl – jaký výkon je dostačující
  - Nutné alespoň základní metriky jako
    - Předpokládaný počet uživatelů – ve smyslu očekávaných http konexí
    - Přijatelná odezva serveru
- Postupným navyšováním zátěže hledáme na více úrovních
  - !plikace, databáze, operační systém, síu
  - Nejsou-li výsledky dobré ladíme a opakujeme

Load testing

- měříme odezvu a chybovost v čase postupným navyšováním zátěže pomocí automatizovaných nástrojů
- Slouží k
  - odhalení chyb jež se jindy neprojeví
    - typicky memory leak, buffer overflow
  - srovnání s předešlými výsledky pro regresní testování
- Provádí se předem danou zátěží jež systém zvládne obsloužit
- Je důležité provádět nad stejně objemnými daty jako produkční prostředí
  - Objemnější databáze podá menší výkon
  - !lgoritmy nad menšími daty pracují rychleji atp;

Stress testing

- Cílem je dostat systém do přetíženého stavu
- Slouží k určení tzv; zotavitelnosti
- Na rozdíl od předchozích testů je pro tento typický spíše chaos a nepředvídatelnost
- Způsobů je mnoho
  - Znásobit počet uživatelů
  - Vypnout či restartovat databázi
  - Zatížit cpu serveru jinými procesy
- Co z toho kromě potěšení z destrukce?
  - Umře systém hned nebo si vše stihne uložit a vypne se?
  - Jenom zamrzne nebo úplně selže?
  - Naběhne po restartu do správného stavu?
  - Zaloguje rozumné chybové hlášení nebo nesmysly?

Vybrané nástroje pro tesování

- **Pěkný seznam** http://www.opensourcetesting.org/performance.php
- **Ab – apache http server benchmarking tool**
  - Velmi skromná aplikace pro jednoduché testování
- **OpenLoad – placené free trial dostupný**
  - Podle marketingu po funkční stránce velmi vybavená aplikace nabízející tzv; skript free testing, umožňující široké nastavení uživatelů včetně user-think-times apod; Nicméně starší kousek co si moc neporadil s nahráváním user akcí z browseru(poslední update 2007);
- **jMeter**
  - Velmi šikovná pure java utilitka; Umožňuje distribuované spouštění testů z více strojů – zajímavá, hojně používaná utilitka.
  - Podpora integrace do mnoha nástrojů(junit, maven,<)

- **Grinder 3**
  - Umožňuje distribuované spouštění testů, nicméně je třeba poměrně rozsáhlé skriptování v jythonu což nepovažuji za šuastné nicméně zajímavá možnost; Nejdůležitější je asi otázka zda nevyplatí investice do vývoje takovýchto testů;
- **OpenSTA**
  - Umožňuje jak tvorbu aktivních testů, tak i odposlech u produkčního nasazení aplikace pasivním monitorováním jejího běhu a výkonu za cenu poměrně složitého a těžkopádného ovládání a nutnosti naučení se skriptovacímu jazyku pro dotazování se webové aplikace (Tato funkcionalita je zjednodušena přiloženým recorderem – který nahraje vaše kliky a převede na skript nicméně tento nefunguje až tak dobře a výsledné skripty jsou nepoužitelné);
- **Httperf (linux only)**
  - Poměrně šikovná aplikace dovolující zahrnout také tzv; user-think-time, nicméně velmi jednoduchá, taktéž neumí brát více URL ze souboru;

- Příklad výsledku puštění Httperf na stroj:

Total: connections 30000 requests 29997 replies 29997 test−duration 299.992 s
Connection rate: 100.0 conn/s (10.0 ms/conn, <=14 concurrent connections)
Connection time [ms]: min 1.4 avg 3.0 max 163.4 median 1.5 stddev 7.3
Connection time [ms]: connect 0.6
Connection length [replies/conn]: 1.000
Request rate: 100.0 req/s (10.0 ms/req)
Request size [B]: 75.0
Reply rate [replies/s]: min 98.8 avg 100.0 max 101.2 stddev 0.3 (60 samples)
Reply time [ms]: response 2.4 transfer 0.0
Reply size [B]: header 242.0 content 1010.0 footer 0.0 (total 1252.0)
Reply status: 1xx=0 2xx=29997 3xx=0 4xx=0 5xx=0
CPU time [s]: user 94.31 system 205.26 (user 31.4% system 68.4% total 99.9%)
Net I/O: 129;6 KB/s (1;1*10^6 bps)
Errors: total 3 client−timo 0 socket−timo 0 connrefused 3 connreset 0
Errors: fd−unavail 0 addrunavail 0 ftab−full 0 other 0

- **Siege (linux only)**
  - Umí načíst URL z konfigurace, umí je náhodně či v daném pořadí pustit na server;

Příklad výsledku puštění na stroj:

The server is now under siege...done
Transactions: 250 hits
Elapsed time: 14.67 secs
Data transferred: 448000 bytes
Response time: 0.43 secs
Transaction rate: 17.04 trans/sec
Throughput: 30538.51 bytes/sec
Concurrency: 7.38
Status code 200: 250
Successful transactions: 250
Failed transactions: 0

Dostupné java knihovny

- Httpunit
  - Velmi nízká úroveň – spousta programování
  - Pouze základní podpora javascriptu

```
ServletRunner sr = new ServletRunner( "web.xml" );
ServletUnitClient client = sr.newClient();
client.setAuthorization( "aUser", „test-admin" );
WebResponse response = client.getResponse("http://localhost/test" );
WebForm form = response.getFormWithID( „fooForm" );
assertNotNull( „Nenalezen formular s ID ‘fooForm’", form );
```

HtmlUnit

- Dá se říci „Prohlížeč bez GUI“
- Slušně interpretuje javascripty(slušná podpora pro ajax)
- Využíval se v projektu WebDriver(nyní Selenium)
- Dost pomalý – zejména interpret javascriptu(implementace rhino)

```
final WebClient webClient = new WebClient();
final HtmlPage page = webClient.getPage(" http://localhost/test ");
final HtmlDivision div = page.getHtmlElementById("some_div_id");
final HtmlAnchor anchor = page.getAnchorByName("anchor_name");
final HtmlDivision div = (HtmlDivision) page.getByXPath("//div[@name='JohnDoe']").get(0);
final HtmlForm form = page.getFormByName(„fooForm");
final HtmlTextInput textField = form.getInputByName("userid");
textField.setValueAttribute("root");
final HtmlPage page2 = button.click();
webClient.closeAllWindows();
```

Z praxe - jmeter

- http://jakarta.apache.org/jmeter
- distribuovaný test engine(server+klient) – využití java RMI
- GUI – chytřejší editor XML testů
  - Z počátku nepřátelský a dokumentace strohá
  - Často používaný
  - Dobrý mailinglist – většinu potíží již někdo řešil
- Podpora pro BeanShell – skriptovací jazyk ala groovy
- Prakticky použito pro testy projektu Oxyonline

Jak interpretovat získaná data

- Výsledkem jmeteru je strohý řádek v souboru
  - Nevhodné pro srovnávání – nutný processing dat
  - GUI jmeteru nabízí listenery
    - Zpracují soubor a zobrazují různé výstupy
    - Slušné ale nevhodné pro srovnávání
  - Jmstats
    - Projekt apache foundation
    - Momentálně beta

Poznatky z praxe

- Pozor na limity TCP spojení u windows XP
  - Myslíte si, že zkoušíte xy uživatelů – reálně jen 10
  - Patch na systémová dll odstraňující limit
- Zjistěte strop klienta
  - Závislé na konkretním stroji a testu
  - Složitější test -> méně simulovatelných klientů současně
- Magie nastavení Garbage Collectoru
  - !plikace mývají různé požadavky a defaultní nastavení není pro výkon serveru to nejlepší
- Ujistěte se, že vám odpovídají všechny servery
  - Běží-li jmeter jako služba windows, má tendence zamrzat