

KAPITOLA 2

Hledání úzkých hrdel: testy výkonnosti a profilace

Dříve nebo později nastane chvíle, kdy budete potřebovat od MySQL vyšší výkon. Co byste se ale měli pokusit vylepšit jako první? Konkrétní dotaz? Schéma? Hardware? Jediný způsob, jak se to dozvědět, je změřit, co všechno systém dělá a otestovat jeho výkon za různých podmínek. Nejlepší strategie spočívá v nalezení nejslabšího článku v řetězu aplikačních komponent a jeho následném posílení. Tento přístup je užitečný zejména v situacích, kdy nevíte, co brání vyššímu výkonu (nebo co bude bránit vyššímu výkonu v budoucnu).

Testy výkonnosti (benchmarking) a profilace (profiling) jsou dva hlavní postupy, jimiž se hledají úzká hrdla. Ačkoliv mají podobné rysy, rozhodně neznamenají totéž. Test výkonnosti měří výkon systému. Může vám pomoci určit kapacitu systému, ukázat, na jakých změnách záleží a na kterých ne, nebo demonstrovat, jak se aplikace vypořádává s daty různého druhu. Profilace oproti tomu pomáhá zjistit, kde aplikace stráví nejvíce času nebo kde spotřebovává nejvíce zdrojů. Jinak řečeno – test výkonnosti odpovídá na otázku "Jak dobře se tohle dělá?", kdežto profilace odpovídá na otázku "Proč se tohle dělá tak, jak se to dělá?"

Tuto kapitolu jsme rozdělili na dvě části, první je o testech výkonnosti, druhá o profilaci. Výklad začneme důvody, proč vlastně dělat testy výkonnosti a popíšeme vám příslušné strategie. Poté přejdeme ke konkrétním taktikám používaným při těchto testech. Ukážeme vám, jak plánovat a navrhovat testy výkonnosti. Ukážeme vám, jakým způsobem je navrhnout, aby vám poskytly přesné odpovědi na vaše otázky. Také vám řekneme, jakým způsobem se testy výkonnosti spouštějí a jak se analyzují jejich výsledky. Tuto část kapitoly nakonec zakončíme tím, že si popíšeme několik nástrojů pro testy výkonnosti a ukážeme si příklady, v nichž se používají některé z těchto nástrojů.

Zbývajíc část kapitoly se pak věnuje profilaci aplikací a MySQL. Ukážeme vám profilační kód ze skutečného světa, který jsme používali v ostrém provozu pro usnadnění různých analýz výkonu aplikací. Dále vám předvedeme, jak se dotazy MySQL zaznamenávají do logu, jak se tyto logy analyzují, a jak se používají různé stavové čítače a další nástroje, které vám pomáhají zjistit, co MySQL vlastně dělá a jaké dotazy na něm spouštíte.

K čemu jsou dobré testy výkonnosti?

V mnohých středně velkých a rozsáhlejších vývojářských týmech MySQL často najdeme partu lidí, jejíž hlavní náplní práce jsou testy výkonnosti. Základní principy a postupy, které jsou používány při testech výkonnosti, by ovšem měl dobře znát každý vývojář a databázový administrátor, protože mají v praxi velmi široké uplatnění. Podívejte se nyní na výčet několika věcí, s nimiž vám mohou testy výkonnosti pomoci:

- Měřit, jak si v současné době vede aplikace z hlediska výkonu. Pokud nevíte, jak rychle běhá teď, u žádné změny si nemůžete být jisti, že vám opravdu pomůže. Můžete také použít výsledky dřívějších testů výkonnosti, abyste diagnostikovali problémy, které jste nepředvíдали.
- Prověřovat škálovatelnost (scalability) systému. Prostřednictvím testu výkonnosti můžete nasimulovat mnohem vyšší zátěž, než jakou běžně zpracovávají vaše ostré systémy, například tisícinásobný nárůst počtu uživatelů.
- Plánovat růst. Testy výkonnosti pomáhají odhadnout, kolik hardwaru, síťové kapacity a jiných zdrojů bude zapotřebí pro takové zatížení, jaké plánujete do budoucna. Tohle může snížit riziko během přechodu na novou verzi systému nebo u závažných změn v aplikaci.
- Testovat způsobilost aplikace zvládat měnící se prostředí. Například můžete zjistit, jak se aplikace vypořádává sice se sporadickými, ale velmi silnými nárazovými špičkami v souběžném zpracování, nebo s různými konfiguracemi serverů. Také se můžete podívat, jak vaše aplikace zvládá různá rozložení dat.
- Testovat různý hardware, software a konfigurace operačního systému. Je pro váš systém lepší RAID 5 nebo RAID 10? Jak se mění výkon nesequenčních zápisů, když se přepnete z disků ATA na úložiště SAN? Škáluje kernel Linuxu verze 2.4 lépe než 2.6? Pomůže k lepšímu výkonu upgrade MySQL? A co kdybychom použili pro data jiný úložný engine? Na všechny tyto otázky dokážete odpovědět pomocí speciálních testů výkonnosti.

Testy výkonnosti se ale dají používat i pro mnohé jiné účely, například si můžete vytvořit sadu pro testování jednotek zdrojového kódu aplikace (tzv. unit testing), nicméně v této kapitole se ovšem soustředíme pouze na aspekty vztahující se k výkonu.

Strategie testů výkonnosti

Existují dvě hlavní strategie testů výkonnosti: testovat aplikaci jako celek, nebo izolovat MySQL. Tyto dvě strategie jsou známy pod názvy testování výkonu v úplném rozsahu (full-stack benchmarking) a testování výkonu jediné komponenty (single-component benchmarking). Zde je několik důvodů, proč měřit aplikaci jako celek, nikoliv pouze MySQL:

- Otestujete aplikaci kompletně, tj. včetně webového serveru, kódu aplikace a databáze. To je prospěšné, protože se nestaráte konkrétně o výkon MySQL – jde vám o výkon celé aplikace.

- Úzkým hrdlem aplikace nemusí být vždy MySQL a právě úplný test výkonnosti vám může pomoci v jeho odhalení.
- Pouze při úplném testování aplikace můžete vidět chování cache pro jednotlivé části.
- Testy výkonnosti jsou dobré právě z toho důvodu, že odrážejí skutečné chování celé aplikace. Něčeho takového jen stěží docílíte, když budete testovat pouze část aplikace.

Na druhé straně je ovšem nutné říci, že úplné testy výkonnosti aplikace se vytvářejí obtížně a ještě hůře se správně připravují. Jestliže navrhnete test výkonnosti špatně, můžete na jeho základě učinit špatná rozhodnutí, protože výsledky nebudou odpovídat realitě.

Někdy se však vlastně nechcete dozvídat informace o celé aplikaci. Potřebujete pouze otestovat výkon MySQL, přinejmenším na počátku. Takový test výkonnosti je užitečný, jestliže:

- Chcete porovnat různá schémata nebo dotazy.
- Chcete otestovat výkon konkrétního problému, který vidíte v aplikaci.
- Chcete se vyvarovat dlouhých testů výkonnosti, takže raději dáte přednost kratším testům, čímž zrychlíte opravný cyklus spočívající v měření a následném provádění změn.

Velmi užitečné je také testovat MySQL v případech, kdy můžete zopakovat dotazy aplikace nad nějakou sadou dat, která odpovídá skutečnému světu. Jak data samotná, tak velikost sady dat by měla být realistická. Pokud je to možné, použijte momentku (snapshot) skutečných ostrých dat.

Bohužel – příprava realistického testu výkonnosti může být dosti komplikovanou a časově náročnou záležitostí, a pokud se vám podaří získat nějakou kopii ostré sady dat, můžete se považovat za dítě štěstěny. V některých případech to samozřejmě možné není – například tehdy, když vyvíjíte zbrusu novou aplikaci, která má jen několik uživatelů a málo dat. Pokud se chcete alespoň orientačně dozvědět, jak si povede, až hodně naroste, nemáte jinou možnost než nasimulovat jak větší objem dat, tak i větší pracovní zátěž.

Co měřit

Je nezbytné, abyste identifikovali své cíle ještě předtím, než začnete s testy výkonnosti (a dokonce ještě předtím, než je vůbec začnete navrhovat). Tyto cíle určují, jaké nástroje a techniky použijete, abyste získali přesné smysluplné výsledky. Vaše cíle formulujte ve formě následujících otázek: "Je tento procesor lepší než tamten?", nebo "Pracují nové indexy lépe než stávající?" atd.

Možná to není zřejmé, ale někdy je potřeba použít různé přístupy pro měření různých věcí. Například, latence a propustnost mohou vyžadovat odlišné testy výkonnosti. Zamyslete se nad následujícími mírami výkonu a zvažte, jak zapadají do vašich cílů.

- **Počet transakcí za jednotku času.** Toto je jeden z nejznámějších klasických testů výkonnosti databázových aplikací. Existují i standardizované testy výkonnosti, jako je například test TPC-C (viz <http://www.tpc.org>) a mnozí výrobci databází na nich intenzivně pracují. Tyto testy výkonnosti měří výkon zpracování transakcí online (OLTP, online transaction

processing) a jsou nejvíce užitečné pro interaktivní víceuživatelské aplikace. Obvyklou měrnou jednotkou je počet transakcí za sekundu.

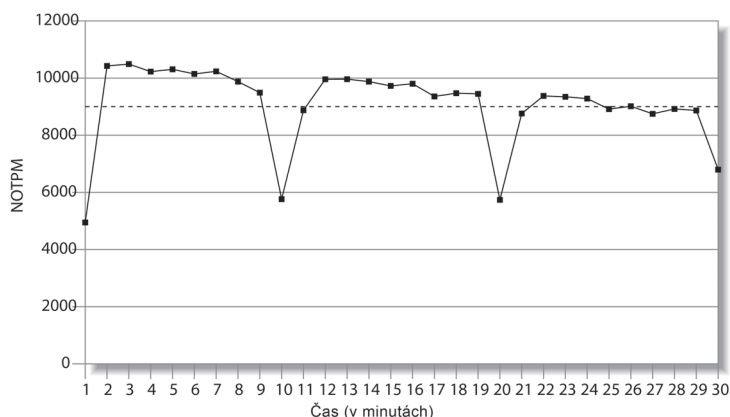
Termín propustnost (throughput) obvykle znamená totéž co počet transakcí (nebo jiné jednotky práce) za jednotku času.

- **Doba odezvy (latence).** Celková doba, po kterou byla požadována nějaká úloha. Podle druhu aplikace se udává v milisekundách, sekundách nebo minutách. Z ní můžete odvodit průměrnou, minimální nebo maximální dobu odezvy.

Maximální možná doba odezvy je málokdy k něčemu užitečná, protože čím déle test výkonnosti běží, tím pravděpodobně bude větší maximální doba odezvy. Navíc se nejedná o zopakovatelnou hodnotu, protože v jednotlivých testech bude tato hodnota s největší pravděpodobností pokaždé odlišná. Z tohoto důvodu mnoho lidí preferuje percentilovou dobu odezvy (percentile response times). Pokud je například 95. percentil doby odezvy 5 milisekund, s 95% pravděpodobností víte, že úloha bude dokončena do 5 milisekund.

Pokud výsledky testů zanesete do nějakého grafu, buď ve formě lomené čáry (například průměrný a 95. percentil), nebo jako histogram, budete schopni na první pohled vidět, jak jsou výsledky rozloženy v čase. Tyto grafy pomáhají výsledovat dlouhodobé chování testů výkonnosti.

Předpokládejme, že váš systém vždy v intervalu jedné minuty provádí pravidelné kontroly. Po dobu této kontroly se systém pozastaví a nebude dokončena žádná transakce. 95. percentil doby odezvy neukáže okamžiky, kdy se zatížení prudce zvyšuje, takže získané výsledky vám nepomohou odhalit daný problém. Na grafu budou ovšem jasně vidět periodické prudké výkyvy doby odezvy. Ilustruje to obrázek 2-1.



Obrázek 2-1. Výsledky třicetiminutového běhu testu výkonnosti dbt2.

V tomto obrázku jsou na ose y vyneseny počty transakcí za minutu (NOTPM). Lomená čára, která je spojuje, ukazuje několik významných výkyvů, jež vám průměrná doba odezvy (čárkovaná rovná čára) neukáže. První vrchol je způsoben tím, že jsou utlumeny cache serveru.

Druhý vrchol pak ukazuje dobu, kdy server tráví spoustu času intenzivním splachováním "ušpiněných stránek" (dirty pages) z bufferů na disk. No, uznejte sami – bez toho grafu byste tyto abnormality mohli odhalit jen velmi obtížně.

- **Škálovatelnost.** Měření škálovatelnosti je užitečné pro systémy, které musí zachovávat dobrý výkon i pod měnící se pracovní zátěží.

"Dobrý výkon i pod měnící se pracovní zátěží" je dost teoretický pojem. Výkon se typicky měří nějakou metrikou, jako je propustnost nebo doba odezvy, přičemž pracovní zátěž může kolísat na základě změn velikosti databáze, počtu simultánních připojení, nebo hardwaru.

Testy škálovatelnosti jsou dobré při plánování kapacit, protože mohou poukázat na takové slabiny ve vaší aplikaci, které by jiné testy výkonnosti nemusely odhalit. Pokud například navrhnete systém, který si dobře povede v testu výkonnosti doby odezvy při jediném připojení (špatná strategie testu výkonnosti), aplikace může mít špatný výkon v okamžiku, kdy dojde k jakémukoli stupni souběžného zpracování. Tento nedostatek v návrhu odhalí pouze takový test, který bude hledat konzistentní doby odezvy při zvyšujícím se počtu připojení.

Některé činnosti, jako jsou dávkové práce, jimiž se vytvářejí souhrnné tabulky z jemně zrnitých dat, prostě potřebují rychlé doby odezvy, tečka. Jistě neuškodí, když otestujete čistě jen jejich doby odezvy, nicméně nezapomínejte na to, v jakých interakcích jsou s jinými činnostmi. Dávkové práce mohou způsobit, že utrpí výkon interaktivních dotazů, nebo naopak.

- **Souběžnost.** Souběžnost (concurrency) je velmi důležitá, ale často chybně používaná a mylně interpretovaná metrika. Populární statistikou je např. údaj, kolik uživatelů si právě nyní prohlíží daný web. HTTP je ovšem bezstavový protokol, takže většina uživatelů jednoduše čte pouze to, co mají zobrazeno ve svých prohlížečích, což nemá se souběžností na webovém serveru nic společného. A obdobně – souběžnost na webovém serveru je něco jiného než souběžnost na databázovém serveru; jediná věc, kterou mají společné, je kolik dat musí zvládnout úložný mechanismus vaší relace. Přesnější mírou souběžnosti na webovém serveru je to, kolik požadavků za sekundu vygenerují uživatelé ve špičce, kdy je zátěž nejvyšší.

Souběžnost se dá také měřit na různých místech v aplikaci. Vyšší souběžnost na webovém serveru může způsobit vyšší souběžnost na úrovni databáze, na to nicméně mají vliv jazyk a používaná sada nástrojů. Například Java s fondem připojení bude mít patrně za následek menší počet souběžných připojení k MySQL serveru, než PHP s trvalými připojeními.

Pro nás je ovšem pořád nejdůležitější počet připojení, na kterých v daném okamžiku běží dotazy. Dobře navržená aplikace může mít například otevřené stovky připojení k MySQL serveru, ale pouze na jejich nepatrném zlomku by měly současně běžet dotazy. To znamená, že web, na kterém "se právě teď nachází 50 000 uživatelů", by měl MySQL server zatěžovat pouze 10 či 15 simultánně běžícími dotazy!

Jinak řečeno – v praxi byste se měli hlavně starat o pracovní souběžnost (working concurrency), neboli o počet vláken či připojení, která pracují simultánně. Také byste měli měřit, zdali hodně klesá výkon, když narůstá souběžnost. Pokud tomu tak je, lze předpokládat, že vaše aplikace patrně nezvládne vrcholy prudce zvyšované zátěže. V tomto případě musíte za-

jistit, aby výkon aplikace příliš neklesal, nebo navrhnout aplikaci tak, aby nevytvářela vysokou souběžnost v těch svých částech, které by nezvládly vrcholnou zátěž. Všeobecně asi budete chtít omezit souběžnost na MySQL serveru pomocí různých návrhů, jako je například aplikační fronta. Více informací k tomuto tématu najdete v kapitole 10.

Uvědomte si, že souběžnost je zcela něco jiného než doba odezvy a škálovatelnost. Není to výsledek nějakého testu výkonnosti, ale spíše vlastnost toho, jak jste test připravili. Neměříte, jaké souběžnosti je vaše aplikace schopna docílit – měříte výkon aplikace při různých úrovních souběžnosti.

Nakonec byste měli otestovat výkon všeho, co je důležité pro vaše uživatele. Testy výkonnosti sice měří výkon, ale "výkon" znamená pro různé lidi různé věci. Nashromáždíte požadavky (ať už formálně nebo neformálně) ohledně toho, jak má být systém schopen škálovat, jaké jsou přijatelné doby odezvy, jaký druh souběžnosti se dá očekávat atd. Poté se pokuste navrhnout testy výkonnosti tak, aby braly všechny tyto požadavky v úvahu, aby neměly klapky na očích, a aby se nesoustřeďovaly pouze na některé věci a opomíjely jiné.

Taktiky testů výkonnosti

Základy máme za sebou, takže přejdeme ke specifickým designu a vykonávání testů výkonnosti. Než ovšem začneme s výkladem, jak dobře dělat testy výkonnosti, podívejme se na několik běžných omylů, které vedou na nepoužitelné nebo nepřesné výsledky.

- Použije se pouze malá podmnožina opravdových dat. Tím máme na mysli situaci, ve které se použije pouze gigabajt dat, zatímco aplikace bude muset zvládnout stovky gigabajtů.
- Použijí se nesprávně rozložená data, například data s rovnoměrným rozložením, zatímco váš systém bude v realitě pracovat s různými shluky dat. (Náhodně vygenerovaná data často neodpovídají rozložení dat ze skutečného světa, s nimiž bude vaše aplikace pracovat.)
- Použijí se nerealistické parametry, například to, že budete předpokládat, že všechny uživatelské profily se budou prohlížet stejně často.
- U víceuživatelské aplikace se použije jednouživatelský scénář.
- Testuje se výkon distribuované aplikace na jediném serveru.
- Opomíjí se soulad s chováním skutečných uživatelů, například "čas na rozmyšlenou" na webové stránce. Uživatel ze skutečného světa požádá o stránku a poté si ji čte. Jinak řečeno: nepředpokládejte, že bude klikat bez rozmyslu na jeden odkaz za druhým.
- V cyklu se spouštějí identické dotazy. Dotazy ve skutečném světě nebývají úplně identické, což znamená, že cache dotazů nebývá využita úplně naplno. Identické dotazy jsou obvykle plně, nebo alespoň částečně, vráceny z cache.
- Zapomíná se kontroly chyb. Pokud výsledky nějakého testu výkonnosti náhle vypadají nesmyslně – například pomalá operace se zčistajasna dokončuje velmi rychle – zkontrolujte,