Udržování konzistentnosti cachovaných dat

Pavel Bžoch[1]

1 Úvod

Potřeba uchovávat a sdílet data v posledních letech roste. Uchovávaná data mohou být různých formátů (např. multimédia, dokumenty, produkty vědeckých výpočtů a další). Takto vyprodukovaná data můžeme uchovávat na lokálním systému souborů nebo na vzdáleném úložišti, které ovšem vyžaduje síťovou komunikaci pro přístup k těmto datům.

Výhodou lokálního uložení je rychlost přístupu k datům. S příchodem SSD disků tato rychlost dosahuje hodnot až 500MB/s pro sekvenční zápis i čtení dat [1]. Nevýhoda lokálního ukládání dat je v omezené kapacitě těchto úložišť a omezené možnosti přistupovat tato data vzdáleně.

Vzdálené úložiště, které může využívat moderní technologie (např. distribuce dat na více uzlů), poskytuje rozsáhlé možnosti pro ukládání dat a dále má vlastnosti jako vyšší spolehlivost, dostupnost, šifrování, rozšiřitelnost a další. Data z nich mohou být přistupována odkudkoli, kde je síťová konektivita. Nevýhodou takto uložených dat je rychlost přístupu, kterou je ovšem možné zvýšit použitím tzv. cache.

2 Cache jako komponenta a její nevýhody

Pro urychlení opakovaného přístupu ke vzdáleně uloženým datům lze použít mezilehlé komponenty cache. Cache uchovává již přistupovaná data lokálně v rychlé paměti (obecně musí poskytovat data rychleji než vzdálený přístup) a poskytuje je okamžitě bez nutnosti vzdáleného přístupu. U dat uložených v cache se předpokládá, že budou v budoucnosti opět používána.

První nevýhodou cache je její omezená kapacita. Při návrhu cache je tedy nutné s tímto omezením počítat a implementovat algoritmy, které v případě plné cache rozhodnou, jaký obsah cache se má nahradit novým obsahem. Tyto algoritmy jsou souhrnně nazývány caching policies. Těmito algoritmy se zde podrobněji nebudeme zabývat.

Druhou nevýhodou, kterou lze u cache najít, je konzistentnost cachovaných dat. Jakmile jsou data uložena v cache, mohou se stát nekonzistentními (lišícími se od zdrojových dat). K tomuto jevu dochází, pokud jsou data na původním úložišti změněna. Tento jev je nežádoucí, protože jsou uživateli předkládána data, která jsou zastaralá. V textu se dále budeme zabývat algoritmy, které se snaží tomuto stavu předcházet.

3 Přístupy pro udržování konzistentní cache

Pro udržování konzistentnosti dat lze vytipovat tři hlavní přístupy, které se snaží tento nežádoucí stav eliminovat - update souborů inicializovaný serverem, ptaní se na stav souborů ze strany klienta a hybridní přístup.

3.1 Update souborů inicializovaný serverem

V tomto přístupu jsou aktualizovaná data zasílána ze serveru přímo klientským aplikacím. Server v tomto případě používá tzv. call-back pro informování klientské aplikace. Server si proto musí pro každý soubor pamatovat odkazy (IP adresy) na všechny klienty, kteří mají daný soubor ve své cache.

[1] Ing. Pavel Bžoch, student doktorského studijního programu Inženýrská informatika, obor Informatika a výpočetní technika, e-mail: pbzoch@kiv.zcu.cz

Problémy tohoto přístupu spočívají v nutnosti udržovat aktuálnost tohoto seznamu a ve způsobu jeho uložení. Seznam se totiž běžně neukládá do perzistentní paměti, což může způsobit ztrátu této informace při výpadku serveru. Dalším velkým problémem tohoto přístupu je používání privátních IP adres a NATu (Network Address Translation) na straně klientské aplikace, kdy není možné navázat spojení ze strany serveru.

Tento přístup používají např. distribuované systémy souborů AFS [2] nebo CODA [3].

3.2 Ptaní se na stav souborů ze strany klienta

V tomto přístupu se klientská aplikace periodicky ptá na stav souborů, které má uloženy v cache. Jakmile zjistí, že některý z cachovaných souborů byl na straně serveru změněn, je jeho nová verze stažena do cache pro případné budoucí použití.

Prvním problémem, který lze v tomto přístupu spatřit, je zjištění stavu souboru. U každého souboru se na straně serveru uchovává časový otisk jeho poslední modifikace. Podle této informace lze zjistit, zda se liší cachovaný a na serveru uložený soubor. Nicméně pokud je soubor uložen na více serverech, mohou být tyto servery rozsynchronizovány a čas uložení nelze v tomto případě použít. Druhou možností zjištění konzistence je používání logických hodin - verzování souborů. Jakmile je na server nahrána nová verze souboru, je zvětšen čítač verze. Následným porovnáním verzí lze zjistit nekonzistentnost cachovaného souboru.

Druhým problémem tohoto přístupu je frekvence zjišťování nových verzí. Jakmile se na nové verze souborů ptáme moc často, dochází k vyššímu zatěžování sítě a serveru. Nicméně máme jistotu, že novou verzi souboru zjistíme rychle. Když se na nové verze souborů ptáme méně často, může dojít k tomu, že zjistíme novou verzi souboru až po dlouhé době a mezitím již mohlo dojít k použití staré verze z cache.

Tento přístup používá např. NFS [2].

3.3 Hybridní přístup

Tento přístup kombinuje oba dva předchozí přístupy. Server v tomto přístupu periodicky posílá tzv. invalidační zprávu. Tato zpráva obsahuje seznam všech souborů, které byly změněny od poslední zprávy. Součástí zprávy mohou být i celé změněné soubory. Tato zpráva je posílána broadcastem na všechny počítače v daném segmentu sítě. V tomto je i vidět nevýhoda tohoto přístupu, kdy jej nelze obecně použít, protože všechny počítače s klientskými aplikacemi nemusí ležet na stejném síťovém segmentu.

4 Závěr

V tomto příspěvku byly představeny algoritmy pro udržování konzistentnosti cachovaných dat. Všechny tyto algoritmy mají své klady i zápory. Při výběru vhodného algoritmu je nutné brát zřetel na podmínky, ve kterých bude algoritmus fungovat. Na základě těchto poznatků bude v budoucnu navržen algoritmus pro udržování konzistentní cache, který se bude snažit výše zmíněné nedostatky eliminovat.

Literatura

[1] Jeremy Laird. (2013, January) Best SSD: 10 of the top SSDs on test. [Online]. http://www.techradar.com/news/computing-components/storage/best-ssd-10-of-the-top-ssds-on-test-994095/

[2] Rainer Többicke, "Distributed File Systems: Focus on Andrew File System/Distributed File Service (AFS/DFS)," in *Mass Storage Systems, 1994. 'Towards Distributed Storage and Data Management Systems.' First International Symposium. Proceedings., Thirteenth IEEE Symposium on*, Annecy, France, 1994, pp. 23-26.

[3] A. Boukerche and R. Al-Shaikh, "Towards Building a Fault Tolerant and Conflict-Free Distributed File System for Mobile Clients," in *Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 02, AINA 2006.*, Washington, DC, USA, 2006, pp. 405-412.