Chybně implementovaná komprese

Cílem komprese je snížit nároky na místo na disku a na síťovou konektivitu. Při přenosu (otevřených) dat tak jistě svou roli (až na ty úplně nejmenší soubory) má, obzvláště proto, že většina otevřených formátů je textových, a texty se komprimují dobře. Dá se ovšem implementovat různě šikovnými způsoby a jednotlivé způsoby mají souvislosti i s metadatovým popisem distribucí datových sad a dokonce i se samotným dělením dat na datové sady. Jednotlivé případy si nyní rozebereme, a postupně budeme volbu komprese vylepšovat od nejméně vhodné až po nejvhodnější.

Více souborů v ZIP (nebo jiném) archivu

Tento případ je doslovné zneužití komprese k obejití toho, že nevím, jak správně dělit data na datové sady a distribuce. Je třeba důsledně oddělovat kompresi a spojení více souborů do jednoho. Na spojení více souborů do jednoho slouží i nástroj TAR, který nic nekomprimuje. Problém spočívá v tom, že jednotlivé soubory v takovém archivu již nelze jednotlivě popsat pomocí metadat, obzvláště pokud se jedná o různorodé soubory, nebo dokonce o adresářovou strukturu. Nedá se pak tedy správně popsat, v jakém formátu jaký soubor je, k čemu slouží, jaké má schéma, případně jaké má podmínky užití.

Tuto situaci je tedy třeba řešit rozdělením takové datové sady na více datových sad, kde co jeden soubor, jedna datová sada. Výjimku tvoří situace, kdy máme stejná data ve více datových formátech, například CSV a RDF, pak máme jednu datovou sadu se dvěma distribucemi, kde jedna bude CSV a druhá RDF, v obou případech bude obsahovat jeden soubor ke stažení.

Pokud by toto vedlo k neúnosně velkému počtu datových sad, je třeba zvážit, zda taková úroveň detailu dává smysl. Každopádně je třeba přidat datové sady s nižší granularitou, a tedy nižším počtem. To lze provést v různých formátech různě, pro XML lze zavést nový kořenový element a jednotlivé záznamy dát do něj, u CSV sloučit více tabulek do jedné a přidat rozlišující sloupec, u RDF stačí soubory slít.

Každopádně je třeba se dostat do situace, že distribuce datové sady má pouze jeden soubor, který není archivem jiných souborů, a pak teprve řešit kompresi.

Soubor ke stažení komprimovaný pomocí ZIP, 7z, RAR nebo jiné neproudové metody

Neproudová metoda komprese je taková, kde pro započetí dekomprese je třeba mít k dispozici celý soubor. To je v pořádku, pokud si takový soubor přinesete na USB flashce, Blu-Ray, DVD, CD. Otevřená data jsou ale poskytována přes Internet, kde nejužším hrdlem je kapacita síťového připojení. Tedy soubor se stahuje delší dobu, a může být užitečné vidět jeho obsah ještě než ho stáhnu celý.

To se běžně používá u webových stránek, kdy samotné HTML začne prohlížeč zobrazovat co nejrychleji, dříve, než je celá stránka stažena. Je totiž pravděpodobné, že uživatel začne číst odshora dolu, a čte pomaleji, než se stahuje zbytek stránky. Vidí tedy obsah dříve, než je celá stránka stažena, a často se také stane, že stránku opustí dříve, než se vůbec celá dostahuje. Tohoto efektu se ale s neproudovou kompresí nedá dosáhnout, jelikož vždy musíme čekat na stažení celého souboru, než

vůbec můžeme začít s dekompresí. Proto se tyto kompresní metody do prostředí Internetu nehodí.

Soubor ke stažení explicitně komprimovaný pomocí gzip, bzip2 nebo jiné proudové metody

Proudová (streamová) metoda komprese je taková, kde mohu obsah dekomprimovat tak, jak ho načítám, jak mi přichází ze sítě, a nemusím čekat až ho dostahuju celý. Takový soubor je pak poskytnut ke stažení, a má pak typicky za jménem ještě další příponu `.gz` nebo `.bz2`, například `.xml.gz`, `.csv.gz`, `.nt.gz` a podobně. Pokud je ale soubor vystaven takto, klient (člověk nebo aplikace) se z hlavičky `Content-Type` protokolu HTTP dozví, že se jedná o data typu `GNU zip`. V prostředí webu se pro tyto účely používá tzv. MIME typ, v tomto případě tedy `application/gzip`. Nedozví se tedy už, co za datový formát je uvnitř, a když soubor stáhne, musí ho před použitím nejprve dekomprimovat správnou metodou. Takový soubor tedy lze postupně rozbalovat, ale nelze ho přímo zpracovávat. Zejména nelze takto komprimovaný datový soubor validovat vůči jeho schématu, protože validátory obvykle nedetekují kompresi a neimplementují dekompresi.

Soubor ke stažení volitelně komprimovaný pomocí gzip

Ideálním řešením v prostředí webu je využít možností, které už dlouho poskytuje protokol HTTP, a poskytovat soubor jak v komprimované, tak v nekomprimované podobě. HTTP hlavička `Accept-Encoding: gzip` umožňuje klientovi říct, že umí přijímat komprimovaná data. Ta jsou pak klientovi poslána v komprimované podobě, a klient si je u sebe rovnou dekomprimuje. Tentokrát je ale v odpovědi serveru jasně popsáno, kterou metodou komprimujeme `Content-Encoding: gzip`, a co za data se přenáší, například `Content-Type: text/csv`, což pak skutečně odpovídá i záznamu v katalogu jako je NKOD. Tato metoda se běžně používá pro webové stránky v HTML, CSS styly a JavaScriptové soubory, a uplně stejně lze použít i pro otevřená data. V této variantě jsou tedy data na serveru v nekomprimované podobě, a pokud klient požádá o komprimovaný přenost, server použije proudovou kompresi a klient proudovou dekompresi. Zdrojová data jsou tedy nekomprimována, cílová také a komprimovaný je pouze přenos po síti, což plní původní cíl komprese na Internetu - šetří síťovou kapacitu. V dnešní době je již disková kapacita levná, čili to, že se zdrojová data na serveru nachází v nekomprimované podobě by vadit nemělo. Tato metoda může mít jednu nevýhodu, a to že pokud k datům přistupuje více uživatelů najednou, může komprese zatěžovat procesor serveru.

Soubor ke stažení volitelně komprimovaný pomocí gzip, s předkomprimovanou verzí

Jedná se o variantu předchozí metody s tím, že na serveru jsou uloženy jak nekomprimované verze souborů, tak komprimované verze souborů. Na požadavek klienta se zašle požadovaná verze, klient nic nepozná a pracuje s konečnou, dekomprimovanou verzí dat. Soubory na serveru jsou tedy předkomprimovány, a kompresí není třeba server zatěžovat v okamžik příchodu požadavku. Například webový server nginx toto podporuje pomocí nastavení `gzip_static`.