Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Analýza metod pro komprimaci XML dat

Bc. Dominik Spiral

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Dominik Spiral

Informatika

Název práce

Analýza metod pro komprimaci XML dat

Název anglicky

Analysis of XML data compression methods

Cíle práce

Cílem práce je analyzovat známé techniky pro komprimaci XML dat a provést jejich srovnání.
Dílčí cíle jsou:
- analyzovat odbornou literaturu a vybrat aktuální kompresní metody,
- stanovit hodnotící kriteria a stupnice,
- provést měření dle zvolených kriterií.

Metodika

Diplomová práce je založena na studiu odborných článků, vědecké literatury a dalších relevantních publikací. Komprimační techniky budou zkoumány na základě zvolených metrik, podle kterých bude vyhodnocena jejich efektivita a využitelnost. Na základě dosažených výsledků komparace a získaných poznatků z odborné literatury bude zformulován závěr.

Doporučený rozsah práce

60 – 80 stran

Klíčová slova

XML, komprimace, analýza, metoda, komparace

Doporučené zdroje informací

Friesen, Jeff. Java XML and JSON. Berkeley, CA New York, NY: Apress, 2016. ISBN 978-1484219157.

Changqing, and Tok W. Ling. Advanced applications and structures in XML processing : label streams, semantics utilization, and data query technologies. Hershey, PA: Information Science Reference, 2010. ISBN 978-1-61520-727-5.

Joshi, Bipin. Beginning XML with C# 7 : XML processing and data access for C# developers. Berkeley, CA: Apress, 2017. ISBN 9781484231043.

McAnlis, Colt, and Aleks Haecky. Understanding compression : data compression for modern developers. Sebastopol, CA: O'Reilly Media, 2016. ISBN 978-1491961537.

Sayood, Khalid. Introduction to data compression. Elsevier Science, 2017. ISBN 978-0-12809474-7.

Předběžný termín obhajoby

2018/19 LS – PEF

Vedoucí práce

Ing. Alexandr Vasilenko, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 15. 2. 2019

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 20. 2. 2019

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 20. 02. 2019

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Analýza metod pro komprimaci XML dat" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 28.3.2019

Poděkování

Tímto děkuji panu Ing. Alexandru Vasilenkovi, Ph.D. za vedení mé diplomové práce, ochotu a rady, které mi pomohly při její kompletaci.

Analýza metod pro komprimaci XML dat

Abstrakt

Tato diplomová práce se zabývá analýzou a srovnáním známých metod pro komprimaci XML dat.

V teoretické části práce je nejprve charakterizován jazyk XML a s ním související pojmy. Dále jsou popsány vlastnosti datové komprese a představeny základní bezztrátové slovníkové a statistické kompresní algoritmy. Poslední část teorie je věnována definici rozdělení XML komprimačních metod a následné charakteristice vybraným z nich.

Praktická část práce se věnuje testování sedmi dostupných kompresních technik nad sestavenou testovací kolekcí skládající se z XML dokumentů s různou strukturou, datovým obsahem a velikostí, přičemž je měřen kompresní poměr a časová náročnost komprese a dekomprese. Na základě vyhodnocení výsledků měření je provedeno srovnání testovaných kompresních metod a následně jsou doporučeny vhodné techniky pro optimální kompresi XML dat.

Klíčová slova: XML, komprimace, analýza, metoda, komparace

Analysis of XML data compression methods

Abstract

This Master's thesis deals with the analysis and comparison of known methods of XML data compression.

In the theoretical part, firstly the XML language and related concepts are characterized. Furthermore, the properties of data compression and basic lossless dictionary and statistical compression algorithms are described. The last part of the theory is devoted to the classification of XML compression methods and the subsequent characteristic of the ones selected.

The practical part deals with the testing of seven available compression techniques over an assembled test collection consisting of XML documents with different structure, data content and size, while the compression ratio and the time required for compression and decompression are measured. Based on the evaluation of the measurement results, a comparison of the tested compression methods is carried out, following which suitable techniques for optimal compression of XML data are recommended.

Keywords: XML, compression, analysis, method, comparison

Obsah

Seznam obrázků

Seznam tabulek

Seznam grafů

Seznam použitých zkratek

API – Application Programming Interface
ASCII – American Standard Code for Information Interchange)
BOM – Byte order mark
BWT – Burrows-Wheeler Transformation
CDATA – Character data
CSS – Cascading Style Sheets
DOM – Document Object Model
DTD – Document Type Definition
DTDPPM – DTD Prediction by partial matching
Gzip – GNU zip
HTML – HyperText Markup Language
kB – kilobyte
LZ77 – Lempel-Ziv 77
LZ78 – Lempel-Ziv 78
LZMA –Lempel-Ziv-Markov-Chain Algorithm
LZSS – Lempel-Ziv-Storer-Szymanski
LZW – Lempel-Ziv-Welch
MB – Megabyte
MHM – Multiplexed Hierarchical Modeling
MTF – Move-to-front
NDATA – Notation Data
PCDATA – Parsed Character Data
PDF – Portable Document Format
PHP – Hypertext Preprocessor
PI – processing instruction
PPM – Prediction by partial matching
SAX – Simple API for XML
SCMPPM – Structural Context Modeling
SGML – Standard Generalized Markup Language
URI – Uniform Resource Identifier
URL – Uniform Resource Locator
UTF – Unicode Transformation Format
W3C – World Wide Web Consortium
WAP – Wireless Application Protocol
WBXML – WAP Binary XML
XHTML – Extensible Hypertext Markup Language
XML – Extensible Markup Language
XMLPPM – XML Prediction by partial matching
XPath – XML Path Language
XSD – XML Schema Definition
XWRT – XML Word Replacing Transform

1 Úvod

Technologie XML (Extensible Markup Language) je dnes široce rozšířený, obecný a platformově nezávislý značkovací jazyk, který umožňuje kódování zejména textových dat způsobem, který je čitelný člověkem i strojově. Díky své jednoduchosti a obecnosti jsou formáty XML od svého vzniku jednou z nejrozsáhlejších technologií především pro standardizované a strukturované skladování a přenos dat mezi různými výpočetními systémy.

Značky v XML, na rozdíl například od HTML, nejsou předdefinované, ale definované uživatelem, přičemž jsou často sebepopisné – jazyk XML je tedy v rámci možností definice významu a struktury dat velmi flexibilní. Pro zachování kompatibility a přenositelnosti formátu ovšem vyžaduje striktní dodržování syntaktických pravidel. Tyto vlastnosti vedou k největší nevýhodě XML oproti jiným datovým formátům – syntaxe může být velmi rozsáhlá a vykazuje vysokou míru verbozity, dokumenty tudíž často dosahují značných velikostí. Volba efektivní techniky komprimace souborů formátu XML je tedy pro účinnou použitelnost této technologie stěžejní, zvláště při přenosu velkých objemů dat či limitovaných kapacitách úložišť rozsáhlých systémů, kdy je potřeba objem dat co nejvíce snížit.

První kapitola teoretické části diplomové práce se bude věnovat popisu a charakteristice samotného jazyka XML. Budou vysvětleny vlastnosti jazyka, jeho principy a definice potřebné k pochopení problematiky. Následovat bude kapitola věnující se samotné kompresi XML dat, v rámci které budou popsané vlastnosti datové komprese a charakterizovány vybrané metody komprese textových dat vhodné pro použití při komprimaci XML souborů, které se budou skládat jak z obecných kompresních technik, tak specifických kompresorů specializovaných na XML.

V rámci praktické části diplomové práce budou vybrané aktuálně využívané metody pro kompresi XML dat podrobeny analýze vycházející z měření. Po stanovení metodiky měření, definici hodnotících kritérií a výběru vhodné testovací sady dat bude provedeno samotné měření, jehož výsledky budou použity pro srovnání a vyhodnocení efektivity jednotlivých kompresorů. Závěrem praktické části práce bude vyhodnocení výsledků

komparace jednotlivých komprimačních technik, na základě kterého by měly být určeny nejúčinnější a nejvhodnější metody pro komprimaci XML dat.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem práce je analyzovat známé techniky komprimace XML dat a provést jejich srovnání.

Ke splnění hlavního cíle práce byly stanoveny dílčí cíle, kterých je potřeba dosáhnout. Prvním dílčím cílem je provedení analýzy odborné literatury, na základě které budou vybrány aktuálně relevantní metody komprese XML dat. Dále budou stanoveny hodnotící kritéria, na základě kterých budou jednotlivé komprimační techniky hodnoceny. Poté bude provedeno měření zvolených kompresních metod dle vybraných kritérií, jehož výsledkem bude jejich přímá komparace.

2.2 Metodika

Metodika řešené diplomové práce je založena na studiu a následné analýze odborných článků, vědecké literatury a dalších relevantních publikací. V praktické části budou vybrané komprimační techniky zkoumány dle zvolených kritérií, na základě kterých bude vyhodnocena jejich efektivita a využitelnost. Na základě dosažených výsledků komparace kompresních metod a teoretických poznatků nabytých z odborné literatury bude zformulován závěr.

3 Teoretická východiska

3.1 XML

Extensible Markup Language (česky rozšířitelný značkovací jazyk, nejčastěji uváděn ve zkrácené formě XML) je obecný značkovací jazyk umožňující popis a reprezentaci převážně textových dat standardizovaným způsobem. XML je takzvaný metajazyk, tedy jazyk, kterým lze definovat vlastní značkovací jazyky, které lze použít ke strukturalizaci a popisu dat určitým způsobem. Tato vlastnost definovat vlastní značky představuje „rozšiřitelnost“ v názvu XML a jeden z hlavních důvodů jeho důležitosti a dlouholeté oblíbenosti, zejména v oblasti výměny dat. [1]

XML, stejně jako například HTML, vychází z jazyka SGML (Standard Generalized Markup Language), což je původní univerzální značkovací metajazyk umožňující definování dalších značkovacích jazyků jakožto svých podmnožin. XML formát je tedy na první pohled HTML velmi podobný, přičemž hlavní rozdíl představuje možnost uživatelem definovaných značek a pravidel, kdežto značky v HTML jsou pevně dány.

Zásadně se pak obě technologie liší ve svém využití – zatímco primárním účelem HTML je způsob prezentace dat, XML, jak již bylo zmíněno, je zaměřeno na strukturu a reprezentaci dat (přidává datům metadata, která popisují jejich význam). Zatímco tedy značky v HTML upravují způsob, jakým jsou data prezentována, XML vzhled žádným způsobem neupravuje, ale dává datům strukturu a definuje jejich význam. [1], [2]

3.1.1 Vznik a vývoj

První funkční draft XML specifikace byl publikován skupinou W3C[1] koncem roku 1996, přičemž ke schválení, jakožto W3C doporučení (standardu) došlo v únoru 1998 pod označením XML 1.0. [2]

Hlavní motivací pro vývoj XML bylo řešení problému, jak zajistit jednoduchou výměnu dat napříč různými systémy a aplikacemi. Tento problém se nejvíce projevoval s akcelerujícím vývojem informačních technologií, přičemž data mají většinou mnohem

[1] World Wide Web Consortium – mezinárodní konsorcium, jehož primárním účelem je vývoj webových standardů s cílem jeho dlouhodobého rozvoje.

delší životní cyklus než softwarové a hardwarové systémy, na kterých se zobrazují a zpracovávají.

Jedním z uvažovaných příkladů je například letecký průmysl, kde mohou produkty (tedy například i jejich technická dokumentace) dosahovat životních cyklů o mnoho desetiletích. V takovém časovém rozpětí se výpočetní systémy několikrát zcela změní a vyžadovala-li data ke zpracování specializovaný či proprietární software, byla by starší data na novějších systémech již nečitelná. Častým problémem byla i výměna dat, která nebyla normalizována – dvě strany, které si potřebovaly data vyměnit, často každá používala formát s vlastní reprezentací dat a sadou instrukcí, pro který existoval proprietární parsovací nástroj. Pro výměnu dat si tedy strany musely vyměnit i programy či mít nástroj umožňující konverzi mezi formáty, což bylo značně neefektivní.

S rychlým nástupem a rozšířením webu objemy vyměňovaných dat značně narostly, z čehož vzešla iniciativa vytvořit jednoduchý standardizovaný jazyk pro popis a výměnu strukturovaných dat, který nebude závislý na systému, na kterém byl vytvořen a na proprietárních technologiích (specifikace XML je volně dostupná, jeho podporu tedy může implementovat kdokoliv). Ve W3C tudíž vznikla speciální pracovní skupina vedena Jonem Bosakem, která vyvinula XML jakožto aplikační profil SGML. [3]

Při návrhu a tvorbě XML specifikace bylo vytyčeno deset základních cílů [4]:

1. XML bude přímočaře použitelné na internetu.
2. XML bude podporovat širokou škálu aplikací.
3. XML bude kompatibilní s jazykem SGML.
4. Psaní programů zpracovávajících XML musí být jednoduché.
5. Počet volitelných vlastností XML musí být minimální, v ideálním případě nulový.
6. XML soubory budou čitelné člověkem a v rámci možností pochopitelné.
7. Návrh XML by měl být zhotoven rychle.
8. Návrh XML bude formální a stručný.

9. Tvorba XML dokumentů bude snadná.

10. Stručnost má v XML značkování minimální důležitost.

Z výše definovaných cílů je zřejmý zejména důraz na jednoduchost, obecnost a použitelnost napříč internetem.

Od publikace v roce 1998 prošel standard XML pouze minimálními revizemi, přičemž poslední změny byly přijaty v listopadu 2008 v páté edici XML 1.0. Paralelně s verzí 1.0 je v platnosti XML 1.1, uvedeno v roce 2014, nyní ve druhé edici ze srpna 2006.

Hlavní rozdíl mezi oběma verzemi představuje podpora Unicode[2] 4.0 znaků, které nebyly v Unicode 2.0, v názvech XML prvků, atributů atd. (v datech jsou nové znaky podporovány i v XML 1.0). Zároveň bylo ve verzi 1.1 v názvech povoleno vše, co nebylo explicitně zakázáno, čímž se XML vyvaruje nutnosti dalších úprav specifikace s novými verzemi Unicode. Poslední větší změnou ve verzi 1.1 je podpora znaků pro ukončení řádku používaných na některých IBM kompatibilních platformách.

Primárním všeobecně akceptovaným a W3C doporučovaným standardem je stále verze 1.0, kdy verze 1.1 není příliš rozšířena a je používaná pouze pokud je například potřeba použít nově přidané znaky v názvech prvků. Většina dokumentů vytvořených v souladu s verzí XML 1.0 je platná i ve verzi 1.1. [5]

3.1.2 **Syntax jazyka XML**

Jak již bylo zmíněno, XML patří mezi značkovací jazyky, XML dokumenty tedy tvoří řetězce Unicode znaků, které lze rozdělit na dvě základní skupiny:

- značkování (markup),
- obsah (textová/znaková data).

Značkování je při parsování dokumentu rozpoznáno pomocí speciálních znaků jako například „<“ a „>“. Patří do něj zahajovací a konečné tagy (značky) ohraničující elementy a další prvky jazyka XML, XML deklarace, komentáře, procesní instrukce a další prvky

[2] Standard pro konzistentní kódování, reprezentaci a zpracování textu obsahující většinu písem světa. V poslední verzi Unicode 11.0 obsahuje 137 439 znaků.

jazyka XML blíže vysvětleny v následujících podkapitolách. Značkování dává dokumentu strukturu a popisuje znaková data. Všechny zbývající znaky, které nejsou součástí značkování, tvoří samotný obsah dokumentu, někdy také nazývaný znaková data z anglického character data.

Přesto, že XML bylo navrženo tak, aby psaní dokumentu bylo co nejjednodušší, syntaktická pravidla pro tvorbu validního dokumentu jsou přísnější, než u jiných značkovacích jazyků. Primárním důvodem je nutnost správného zápisu pro další zpracování XML dokumentu různými aplikacemi. Tato podkapitola popisuje základní prvky XML, ze kterých se skládá XML dokument a syntaktická pravidla, která je při jejich užívání potřeba dodržet. [6]

3.1.2.1 **XML deklarace**

Začátek XML dokumentu by měl obsahovat takzvaný prolog, který obsahuje XML deklaraci. Jedná se o speciální nepárový element, který informuje parser, že se jedná o XML dokument. XML deklarace není povinnou součástí XML, nicméně pokud je deklarace v dokumentu uvedena, nesmí být žádný obsah před ní – vyskytuje se nad kořenovým elementem. V minimální formě deklarace obsahuje atribut „version“, kterým je identifikována verze XML specifikace, podle které je dokument zhotoven (1.0 nebo 1.1). [6]

Deklarace může obsahovat další dva volitelné atributy. Atribut „encoding“ specifikuje, podle jakého standardu jsou znaky v XML dokumentu kódovány. Nejčastěji je používáno UTF-8 a UTF-16[3]. V případě absence atributu encoding či celé deklarace parser rozpozná kódování dokumentu dle speciální sekvence znaků na jeho začátku vytvořené textovým procesorem při jeho uložení nazývané BOM[4]. Pokud dokument neobsahuje ani tuto sekvenci, je předpokládáno UTF-8.

Třetím atributem, který se může objevit v XML deklaraci, je „standalone“, tento atribut je relevantní pouze pokud je použito DTD (viz kapitola 3.1.3) a určuje, zda existují deklarace v externím souboru. Defaultní hodnotou je „no“ a tudíž není potřeba ji

[3] Nejrozšířenější způsoby kódování Unicode znaků s proměnnou délkou znaku jeden až čtyři byty u UTF-8 a dva nebo čtyři u UTF-16, přičemž jedno bytové znaky UTF-8 se kódují stejným způsobem jako v ASCII, čímž je zaručena zpětná kompatibilita.
[4] Byte order mark, neboli označení pořadí bytů je Unicode znak, který slouží k rozpoznání použitého kódování a případně pořadí ukládání bytů.

specifikovat, zatímco hodnota „yes“ označuje, že dokument není závislý na externích deklaracích entit či defaultních hodnot atributů. Dnes se již atribut standalone v XML deklaraci příliš neobjevuje z důvodu nízké využávanosti DTD. XML deklarace se všemi atributy může mít například následující zápis [1]:

<?xml version=“1.1“ encoding=“UTF-16“ standalone=“yes“?>

3.1.2.2 **Elementy a atributy**

Element (neboli prvek) je nejzákladnější složkou XML dokumentu. Elementem lze označit vše od počátečního po ukončovací tag včetně. Každý element může obsahovat kombinaci následujícího:

- textový obsah,
- atributy,
- komentáře a procesní instrukce,
- další vnořené elementy. [9]

XML element musí mít název, který je obsažen v počátečním a ukončovacím tagu. Název elementu musí začínat alfanumerickým znakem nebo podtržítkem, nesmí obsahovat mezery a je nutné dodržovat shodu velkých a malých písmen, v XML nejsou žádná rezervovaná slova kromě „xml“ na začátku názvu elementu. Značky se vymezují pomocí znaků „<“ a „>“, ukončovací značka je odlišena od počáteční symbolem lomítka před názvem elementu, oblast elementu se tedy v dokumentu vymezí následujícím způsobem [2]:

<nazev_elementu></nazev_elementu>

Kromě názvu elementu může počáteční značka obsahovat také jeden nebo více atributů. Atributy poskytují dodatečné upřesňující informace o daném elementu. Hodnoty atributů se zapisují do uvozovek (případně apostrofů, pokud samotná hodnota obsahuje uvozovky) a více atributů jednoho elemetu lze oddělit mezerou:

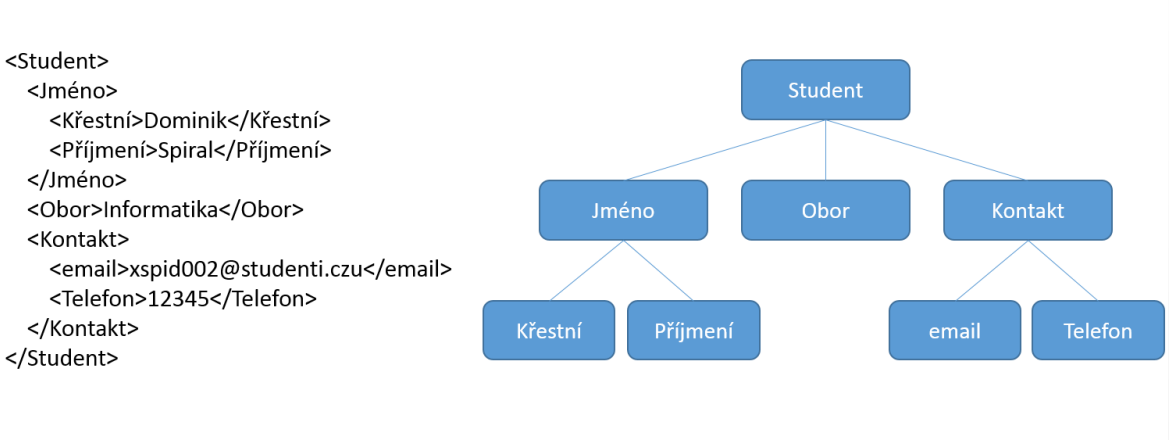
<nazev_elementu atribut1=“hodnota1“ atribut2=“hodnota2“>...

Element v XML také nemusí mít žádný obsah mezi značkami. Prázdné elementy mohou obsahovat atributy a využití mají například pro zachování pořadí značek definované schematem. Zápis takového elementu pak lze zkrátit následovně:

<prazdny_element />

Elementy v XML jsou rozšiřitelné, autor XML dokumentu tedy může například přidat do elementu další doplňující vnořené elementy bez vlivu na funkci aplikace, která zpracovává pouze původní informace. [1], [9]

Jak již bylo zmíněno, XML elementy lze vnořovat jeden do druhého, počet těchto vnoření je neomezený, musí být pouze dodrženo, že na nejvyšší úrovni je pouze jeden kořenový element (viz syntaktická pravidla). XML dokument tedy tvoří hierarchickou stromovou strukturu, která se větví od kořenového elementu, který je předkem všech ostatních elementů. Obrázek 1 obsahuje ukázku stromové reprezentace XML dokumentu, ve kterém kořen stromu (root) tvoří entita Student. [1]



Obrázek 1- Ukázka XML dokumentu a jeho stromové struktury, vlastní tvorba.

3.1.2.3 Speciální znaky

XML syntax obsahuje několik speciálních znaků, které není možné vložit mezi počáteční a konečné značky elementu nebo do hodnoty atributu. Jedná se například o znaky „<“ a „>“ používané pro ohraničení tagů. Pokud během procházení dokumentu parser narazí na znak < v textu, bude jej interpretovat jako začátek dalšího tagu a zpracování skončí chybou.

Pro vložení speciálních znaků do obsahu se tedy používají číselné znakové reference a reference znakových entit. Číselná znaková reference odkazuje na číslo pozice znaku v tabulce Unicode, zatímco reference znakové entity odkazuje na jméno entity, která požadovaný speciální znak zastupuje.

<	&lt;
>	&gt;
&	&amp;
'	&apos;
"	&quot;

Tabulka 1 - Znakové entity předdefinované v XML.

Pokud se v datech objevuje více speciálních znaků, například při vkládání zdrojového kódu do obsahu, je možné použít sekci CDATA (zkráceně character data) s prefixem „*<![CDATA*“ a sufixem „*]]>*“. Obsah sekce CDATA parser ignoruje, dokud nenarazí na ukončovací znaky *]]>*, do této sekce je tedy možné vkládat vše bez použití referencí, včetně částí nebo celých XML dokumentů, aniž by docházelo k mylným interpretacím značkování. [1]

3.1.2.4 **Komentáře a procesní instrukce**

XML umožňuje do dokumentů vložit komentáře a procesní instrukce. Komentář se může vyskytovat v libovolné části dokumentu za XML deklarací, mimo jiné značky. Komentáře umožňují například vysvětlení následujícího kódu či poskytnutí dodatečných informací k dané sekci. Dle XML specifikace komentáře netvoří součást znakového obsahu XML dokumentu a většinou bývají při zpracování ignorovány, tudíž bývají viditelné pouze ve zdrojovém kódu dokumentu. XML parsery však mohou volitelně textový obsah komentářů aplikacím předat. Syntax komentářů je v XML stejný jako u HTML [4]:

<!-- text komentáře -->

Aby byla dodržena konformita s W3C doporučením a nedošlo k chybné interpretaci konce komentáře parserem, nesmí tělo komentáře obsahovat dvě po sobě jdoucí pomlčky, pro zajištění kompatibility se však doporučuje nepoužívat v obsahu komentáře znak pomlčky vůbec. [1]

Dalším speciálním značkováním, které XML umožnuje, jsou procesní instrukce. Procesní instrukce XML dokumentům umožňují obsahovat instrukce, které parser předá aplikaci ke zpracování. Stejně jako komentáře se mohou vyskytovat kdekoliv v XML dokumentu kromě jiných značek a jsou vymezeny posloupností znaků „<?“ na začátku a „?>“ na konci. Na začátku procesní instrukce je takzvaný target, který identifikuje aplikaci, které jsou instrukce určeny, za ním jsou samotné instrukce ve formátu čitelném pro danou aplikaci. [4]

```
<dokument>
  <datum>Dnešní datum je <?php echo Date("d.m.Y")?></datum>
  <para>Nějaké důležité informace.</para>
</dokument>
```

Obrázek 2 - Integrace PI pro zpracování PHP interpreterem v těle XML elementu. [6]

3.1.2.5 **Namespaces**

Jelikož názvy elementů v XML definuje vývojář a nejsou předurčeny, může často docházet ke konfliktům při kombinaci XML dokumentů z různých zdrojů. Pokud nastane situace, kdy jsou názvy elementů či jiných XML prvků v jednom dokumentu shodné s názvy v jiném, přičemž oba dokumenty obsahují jiný formát XML, dojde při jejich spojení ke konfliktu, jelikož mezi shodně pojmenovanými prvky parser nedokáže rozlišit, přestože budou mít odlišný obsah a význam.

Namespaces, neboli jmenné prostory, těmto konfliktům zamezují. Jmenný prostor je určen svým URI (Uniform Resource Indentifier)[5]. Deklarace se provádí pomocí vyhrazeného atributu „*xmlns*“, jehož hodnotou je daná URI:

<element xmlns=“URI“> ... </element>

Na specifikované URI adrese se nemusí nacházet reálný obsah, jelikož slouží pouze pro přidělení unikátního jména danému jmennému prostoru, nicméně často se jedná o URL vedoucí na stránky popisující daný namespace nebo XML formát. Pokud je pro element definován namespace výše popsaným způsobem, spadají do něj automaticky i všichni

[5] Řetězec znaků s definovanými syntaktickými pravidly, který jednoznačně identifikuje konkrétní internetový zdroj. Nejznámějším příkladem je URL.

potomci tohoto elementu, nejsou-li explicitně přiřazeni do jiného namespacu, jedná se tedy o takzvaný defaultní namespace.

U jmenných prostorů se také často využívají prefixy – za atribut *xmlns* se vloží dvojtečka s uživatelem definovaným prefixem.

<prefix:element xmlns:prefix="URI"> ... </element>

V takovém případě se namespace nevztahuje automaticky na element, u kterého byl deklarován, ani jeho potomky, ale pouze na elementy, před jejichž název je explicitně vložen daný prefix. Při tomto způsobu definice namespaců lze také deklarovat u jednoho elementu více než jeden namespace a na základě jejich prefixů přiřadit elementu a jeho potomkům různé z nich.

Oba způsoby deklarace lze kombinovat, je tedy možné definovat defaultní namespace a zároveň prefixované u jednoho elementu, toto ilustruje příklad níže, ve kterém první element spolu s vnořeným druhým elementem spadají do defaultního jmenného prostoru, zatímco vnořený element tři, u kterého je stanoven prefix, do jiného. [10]

```
<element1 xmlns:="URI1" xmlns:prefix="URI2">
   <element2> ... </element2>
   < prefix:element3> ... </element3>
</element1>
```

3.1.2.6 **Well-formed XML dokument**

Syntaktická pravidla XML jsou jednoduchá a logická. Při jejich dodržení se XML dokument nazývá „well-formed" a tudíž je v souladu se specifikací. Chyby v XML syntaxi nejsou povoleny – v případě, že parser zpracovávající XML dokument detekuje chybu, měl by jeho další zpracování dle XML specifikace ukončit (toto je rozdíl například oproti HTML, kdy prohlížeče zobrazují i dokumenty s chybným syntaktickým zápisem). Pravidla pro well-formed XML dokument lze shrnout následovně[8]:

- XML dokument musí obsahovat jeden kořenový (root) element, který je rodičem (parent) všech ostatních prvků.

- Pokud se v dokumentu vyskytuje volitelný prvek prolog, musí se vyskytovat na začátku, před kořenovým prvkem.
- Všechny XML elementy musí mít ukončovací tag (výjimku tvoří speciální element prolog, který není součástí XML dokumentu).
- Elementy v XML musí být správně vnořeny jeden do druhého (pokud se v rámci jednoho elementu vyskytuje počáteční značka druhého, pak musí první element obsahovat i ukončovací značku druhého, není tedy možné následující křížení:

  <element_a><element_b></element_a></element_b>
- Počáteční a ukončovací tagy elementů musí dodržet stejnou velikost písmen, jelikož jsou case-sensitive.
- Hodnoty atributů musí být vždy uvedeny v uvozovkách.
- Znaky se speciálním významem vyskytující se v obsahu musí být psány pomocí referencí (znakových entit nebo číselných znakových referencí).
- Uvnitř komentáře nesmí být dvě pomlčky za sebou, zároveň tělo komentáře nesmí končit pomlčkou. [2], [7]

XML parsery vnímající jmenné prostory vynucují další dvě pravidla definovaná ve W3C doporučení o jmenných prostorech v XML. Při jejich splnění se dokument nazývá „namespace-well-formed“:

- Názvy všech elementů a atributů obsahují maximálně jeden znak dvojtečky.
- Targety procesních instrukcí, názvy entit a notace[6] nesmí obsahovat znak dvojtečky. [11]

[6] Entity a notace souvisí s DTD (definicí typu dokumentu) vysvětlenou v kapitole 3.1.3.1.

3.1.3 Validita XML dokumentů

Aby byl XML dokument validní, musí být well-formed, tedy splňovat syntaktická pravidla sumarizovaná výše, a zároveň vyhovovat pravidlům a omezením, která definují jeho výsledný formát a strukturu. Může se jednat například o dodržení pořadí elementů a vymezení jejich vzájemných vztahů, stanovení atributů, které mohou být pro daný element použity atd. Právě tato omezení, která zmenšují množinu validních dokumentů pro danou sadu pravidel, vymezují konkrétní formáty (jazyky) se specifickým využitím vycházející z obecného XML. [6]

Tento soubor pravidel tvoří gramatiku dokumentu a může být deklarován přímo v daném XML dokumentu, většinou se však nachází v externím souboru, který obsahuje pouze gramatiku. Při validaci parser kontroluje shodu mezi XML dokumentem a jeho gramatikou a v případě nalezení odchylky od pravidel (například chybějící povinná entita či atribut) reportuje aplikaci chybu, v takovém případě dokument není validní. Poté už je na samotné aplikaci, zda bude chybu ignorovat, pokusí se ji opravit či zda XML dokument odmítne – parser při nalezení chyby ve validitě na rozdíl od kontroly „well-formedness“, nemusí zastavit a může pokračovat v parsování.

Ne každý XML parser validitu kontroluje a i validující parser tak činní většinou až na explicitní vyžádání, jelikož se jedná o relativně náročnou operaci, při které je potřeba zkontrolovat každý element dokumentu vůči výchozímu schématu.

Gramatiky XML dokumentů nejsou přímou součástí XML a jsou psané ve speciálních jazycích – nejrozšířenějšími z nich jsou původní DTD (Document Type Definition neboli definice typu dokumentu) vycházející ze SGML, které je referencované přímo ve W3C XML doporučení a XML Schema, které bylo napsáno v XML. Tyto dva jazyky jsou blíže popsány v následujících dvou podkapitolách. [1]

3.1.3.1 Document Type Definition

Definice typu dokumentu DTD je nejstarší jazyk, který se používá pro specifikaci gramatiky jazyků z rodiny SGML včetně XML. Pomocí DTD lze definovat, jaké elementy se mohou v dokumentu vyskytovat na jakém místě, co je jejich obsahem a jaké mohou mít atributy, případně jaké budou defaultní hodnoty atributů.

Deklarace typu dokumentu

Pro připojení DTD k XML dokumentu musí dokument obsahovat deklaraci typu dokumentu DOCTYPE, která informuje parser, že je s dokumentem spjata definice typu dokumentu. Tato deklarace se uvádí před kořenovým elementem dokumentu, za XML deklarací. Jak již bylo zmíněno, po deklaraci lze DTD umístit přímo do daného XML dokumentu či na něj externě odkázat. Možnost lokálního umístění definice typu dokumentu se příliš nepoužívá, jelikož pak DTD nelze sdílet mezi více dokumenty, oba způsoby je však možné kombinovat, čímž je umožněno takzvanou interní podsadou doplnit či upravit části externího DTD pro daný dokument, přičemž syntax může být následovný:

```
<!DOCTYPE korenovy_element SYSTEM "URI odkaz na externí DTD" [
   <!-- interní podsada DTD-->
]>
```

Jak ukazuje syntax výše, po klíčovém slovu DOCTYPE se v deklaraci uvádí název kořenového elementu, od kterého má parser na dokument aplikovat DTD. V případě čistě interního DTD pak už následují samotné deklarace entit vymezeny hranatými závorkami. V případě, že je odkazován externí soubor obsahující DTD, následuje po specifikaci kořenového elementu klíčové slovo SYSTEM nebo PUBLIC, které určuje, zda je DTD soukromé či veřejně přístupné. Za klíčovým slovem SYSTEM se vyskytuje systémový identifikátor, například ve formě URI, za PUBLIC se pak uvádí speciální veřejný identifikátor, za kterým může být volitelně systémový identifikátor pro případ, že veřejný nebude parserem rozpoznán. [6], [12]

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/2002/REC-xhtml1-20020801/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html;charset=ISO-8859-2"/>
    <title>Zpráva</title>
  </head>
  <body>Moje zpráva</body>
</html>
```

Obrázek 3 - Deklarace typy dokumentu pro DTD XML formátu XHTML [12] s úpravami.

Deklarace elementu

Samotná definice typu dokumentu umožňuje deklaraci elementů, atributů, entit a notací. V rámci deklarace elementu je stanoven název elementu a jeho obsah.

<!ELEMENT nazev_elementu obsah_elementu>

Základní pravidla pro stanovení obsahu elementů lze definovat pomocí klíčových slov – například EMPTY označuje, že element musí být prázdný, ANY povoluje libovolný obsah a #PCDATA udává, že obsahem elementu jsou parsovatelná znaková data. Pro definici složitější vnitřní struktury elementu lze pak použít regulární výrazy, například svislici „|“ pro výběr mezi dvěma alternativy, asterisk „*“ pro libovolný počet opakování daného prvku či plusové znaménko „+“ označující nutnost alespoň jednoho opakování prvku. [12]

Deklarace atributu

V rámci deklarace atributů elementu, pro kterou se používá klíčové slovo ATTLIST, lze definovat název atributu, jeho typ, povinnost a implicitní hodnotu. Nejčastějším typem jsou znaková data CDATA, lze ovšem například stanovit typ ID, tedy unikátní textový řetězec v rámci daného dokumentu, či typ Enumerated, který umožňuje zvolit jednu ze seznamu definovaných hodnot. Dále lze určit povinnost atributu (#REQUIRED – povinný, #IMPLIED – atribut může být vynechán, případně lze také uvést před stanovenou defaultní hodnotu atributu klíčové slovo #FIXED, v takovém případě defaultní hodnotu v dokumentu nelze měnit). Poslední položkou deklarace atributu je samotná výchozí hodnota atributu. [12]

<!ATTLIST nazev_elementu nazev_atributu typ_atributu povinnost_atributu hodnota>

Deklarace entit a notací

Jak již bylo zmíněno, kromě deklarací elementů a atributů lze prostřednictvím DTD také deklarovat entity a notace. V kapitole o speciálních znacích byly popsány znakové entity, které jsou pro XML předdefinované. DTD umožňuje deklaraci vlastních entit, které je pak možné využít v XML dokumentu použitím referencí na entitu stejným způsobem, jako tomu je u předdefinovaných znakových entit (*&nazev_entity;*).

Entity se dle způsobu deklarace dělí na interní a externí. V případě interních entit je text, který entita zastupuje, uveden přímo v deklaraci dané entity. Interní entity se primárně používají pro nahrazení komplexnějších opakujících se textů v dokumentu, ve kterém se místo vypisování celého textu uvede reference na entitu.

<!ENTITY nazev_entity "zastupovaný text">

Pokud deklarace entity odkazuje na externí soubor, jedná se o externí entitu. U externí entity je třeba za jejím názvem uvést klíčové slovo PUBLIC/SYSTEM a příslušný systémový či veřejný identifikátor. Výhodou externích entit je možnost vytvoření společné reference sdílené více dokumenty, kdy v případě změny externího souboru budou všechny dokumenty odkazující se na danou entitu aktualizovány.

V případě, že deklarace externí entity obsahuje klíčové slovo NDATA (notation data) následované datovým typem (odkazem na notaci), data načtená z adresy dané identifikátorem nejsou XML textovými daty nýbrž binárními, tudíž nedochází k jejich parsování. Notace aplikaci pracující s XML dokumentem poskytuje informace o datovém typu na adrese odkazované v entitě. Identifikaci datového typu souboru notace umožňuje pomocí veřejného identifikátoru považovaného za standard (například „JPG 1.0“) nebo systémového identifikátoru (například „image/jpeg“, cesta k aplikaci, která umí daný formát číst atd.). Příklad níže znázorňuje možný syntax externí netextové entity a příslušné notace. [13]

<!ENTITY nazev_entity SYSTEM "URI souboru“ NDATA nazev_notace>

<!NOTATION nazev_notace PUBLIC "verejne_ID" "systemove_ID">

3.1.3.2 XML Schema

XML Schema (také označováno XSD – XML Schema Definition, z důvodu zaměnitelnosti názvu s obecným označením jazyků pro definici pravidel a omezení XML dokumentů) je gramatický jazyk pro deklaraci struktury a povoleného obsahu XML dokumentu, který byl přijat jakožto W3C doporučení v roce 2001. Umožňuje stejně jako DTD definovat elementy, jejich pořadí, strukturu a vzájemný vztah či atributy. Dále například umožňuje stanovit datové typy obsahu elementů, což v DTD není podporováno.

Srovnání s DTD

XML Schema vzniklo z důvodu nutnosti vyřešit limitace DTD. Patří mezi ně zejména chybějící podpora jmenných prostorů u DTD, nemožnost využití dědičnosti při deklaracích (XML Schema poskytuje objektově orientovaný přístup k deklaracím) či špatná čitelnost a s tím spojená komplikovaná údržba rozsáhlých DTD souborů. XML Schema je také kompletně vytvořeno v XML a schémata (gramatické dokumenty) jsou samy o sobě XML dokumenty využívající speciální elementy, není tedy třeba znát jiný syntax a je rozšiřitelné. Komplexita a širší funkcionalita XML Schema však vede i k poměrně složitější syntaxi oproti DTD.

Velkým rozdílem je také již zmíněná možnost definice omezení nad textovým obsahem pomocí mnoha jednoduchých datových typů – lze specifikovat například double, boolean, date, string, mnoho variací typu integer či odvodit vlastní datový typ, zatímco DTD nabízí pouze velmi obecná CDATA a PCDATA, která neumožňují omezení textového obsahu. [1]

Základy tvorby schématu

Každý XML dokument, který tvoří schéma, musí obsahovat kořenový element <*schema*>, v němž je deklarován standardní XML Schema namespace, kterému bývá nejčastěji přidělen prefix *xs* či *xsd* – tento prefix se pak udává před speciální XML Schema elementy a atributy, které se ve schématu používají pro typovou definici. Element <*schema*> může obsahovat i další atributy, jako například *targetNamespace*, který udává, do jakého jmenného prostoru patří elementy definovány daným schématem. Deklarace kořenového elementu schématu tedy může vypadat například následovně:

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
TargetNamespace="URI" xmlns="URI"> ... </xs:schema>

Prostřednictvím XML Schema lze definovat primárně jednoduché a komplexní elementy, přičemž jednoduchý element nese pouze textová data a obsahuje povinné atributy *name* a *type* pro stanovení názvu a datového typu, který dále vymezuje, jakého typu mohou textová data být. Další atributy jednoduché elementy obsahovat nemohou.

<xs:element name="nazev_elementu" type="datovy_typ" />

Komplexní elementy mohou obsahovat libovolné atributy, další elementy, textová data či mohou být prázdné. Komplexní element lze definovat pomocí elementu *xs:complexType*, který tvoří kontejner pro definice elementů, ze kterých se definovaný komplexní element skládá. Potomky elementu *xs:complexType* mohou být také například takzvané kompozitory, které umožňují stanovení pravidel pro pořadí výskytu svých potomků v XML dokumentu – v případě použití *xs:sequence* musí být elementy v dokumentu v pořadí deklarovaném ve schématu, *xs:choice* značí, že pouze jeden z elementů se může v dokumentu objevit a *xs:all* umožňuje libovolné pořadí.

Definice komplexního elementu:

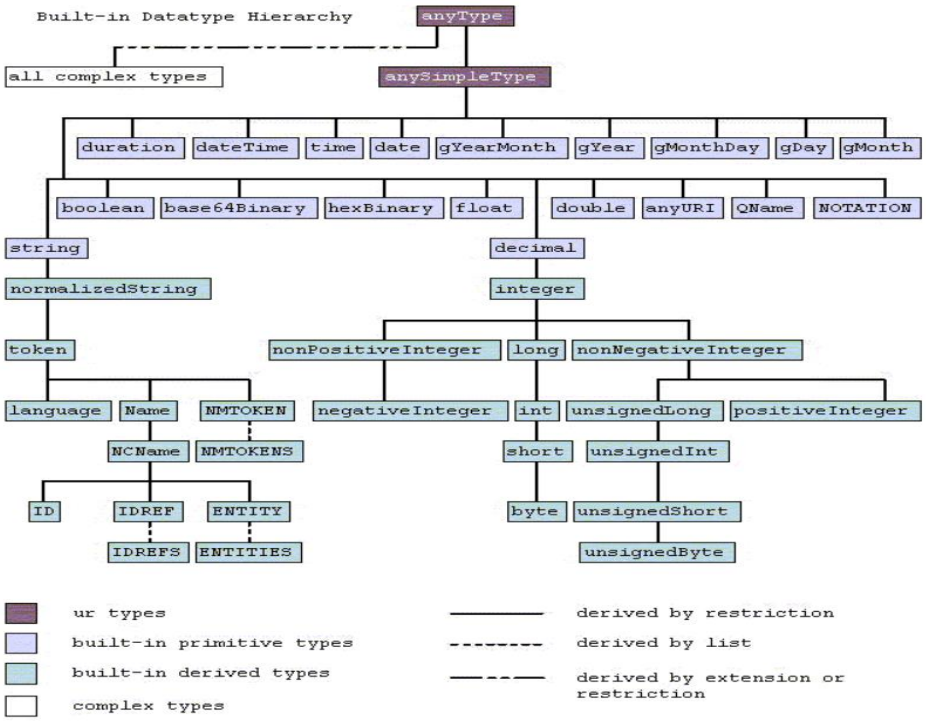
```
<xs:element name="Supplier">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Phone" type="xs:integer" />
            <xs:element name="Address" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Přislušné validní XML:

```
<Supplier>
    <Phone>0123987654</Phone>
    <Address>Ulice 123, Město, Česká republika 12345</Address>
</Supplier>
```

Obrázek 4 - Ukázka definice komplexního typu v XSD [14] s úpravami.

U elementů lze dále například specifikovat maximální či minimální počet opakování každého z nich pomocí indikátorů *minOccurs* a *maxOccurs*, kontrolovat unikátnost hodnot, seskupovat elementy do skupin, nastavovat výchozí hodnoty elementu atd. Lze také definovat restrikce elementů, kdy výchozí element je omezen na podmnožinu původního rozsahu povoleného obsahu. Restrikce je možné použít i u jednoduchých elementů pro dodatečné omezení povolených hodnot datového typu (lze například užitím speciálních atributů nebo regulárních výrazů omezit předdefinovaný datový typ integer na určitý interval či jinou podmnožinu celých čísel). Užitím restrikcí je odvozena i většina předdefinovaných datových typů, jak ukazuje obrázek 5 (dále lze odvodit jednoduché typy pomocí seznamu a sjednocení více typů).



Obrázek 5 - Odvození předdefinovaných datových typů XSD. [15]

Elementy komplexního typu lze také rozšířit pomocí konstruktu *xs:extension*, který umožňuje využít objektových principů dědění a specializace – nový element pak bude obsahovat všechny elementy výchozího rozšířené o nově dodefinované.

Kromě elementů lze pomocí XML Schema také definovat jejich atributy, stejně jako u DTD lze kromě názvu a typu volitelně stanovit jejich povinnost a výchozí, případně fixní hodnotu. [14]

<xs:attribute name="nazev_atributu" type="datovy_typ" use="required" default="vychozi_hodnota" />

Pro asociaci XML dokumentu s konkrétním schématem se standardně používá atribut *xsi:schemaLocation*, jehož obsahem je seznam dvojic hodnot cílový jmenný prostor schématu – umístění schématu. Prefix *xsi* odkazuje na standardní namespace XML Schema Instance. Reference na schéma se uvádí v počáteční značce kořenového elementu dokumentu, od kterého má parser provádět kontrolu validity [1]:

```
<korenovy_element xmlns="URI_jmenneho_prostoru"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="URI_jmenneho prostoru schema.xsd">
```

3.1.4 XML parsery

XML parsery (procesory) byly v dřívějších kapitolách již několikrát zmíněny – jedná se o softwarové knihovny či moduly, které poskytují aplikacím rozhraní pro přístup k obsahu a struktuře XML dokumentu. XML parsery jsou například zabudovány ve všech webových prohlížečích, jelikož XML je hlavním formátem pro přenos dat na internetu. Parser prochází každou část dokumentu a převádí je do struktur v paměti, se kterými může koncová aplikace manipulovat. Parsery jsou k dispozici i ve formě samostatných programů, použitelných například na kontrolu tvořeného dokumentu. [6]

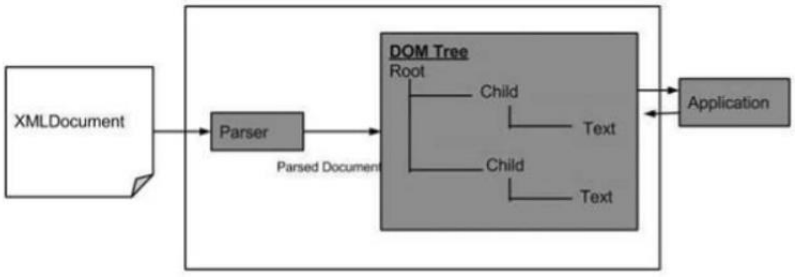
Parsery se dle XML specifikace dělí na validující a nevalidující. Aby bylo zajištěno, že struktura a význam obsahu XML dokumentu budou při jeho přenosu mezi systémy vždy interpretovány správně, musí každý parser kontrolovat jeho syntaktickou správnost (well-formedness) dle pravidel stanovených XML specifikací. Validující parsery kromě toho také kontrolují validitu dokumentu vůči patřičnému DTD či schématu. [4]

Většina parserů je založena na rozhraních DOM a SAX, přičemž existuje mnoho různých implementací v mnoha programovacích jazycích.

3.1.4.1 **DOM**

DOM (Document Object Model, neboli objektový model dokumentu) je platformově a jazykově nezávislé API a doporučení W3C, které nahlíží na XML dokument jakožto stromovou strukturu, ve které je jeho veškerý obsah reprezentován uzly – kromě každého elementu tvoří samostatné uzly i textový obsah elementů, atributy elementů či například komentáře. Soubor je načten do paměti, ve které DOM parser vytvoří stromovou reprezentaci celého dokumentu a až poté může klientská aplikace použít funkce DOMu umožňující vyčíst, změnit, smazat či přidat jakýkoliv uzel stromu. [6]

Jelikož je v paměti načten strom celého dokumentu, je možné jím procházet a vyhledávat ve všech směrech a manipulovat s libovolným uzlem, ze stejného důvodu může být ovšem DOM paměťově i časově velmi náročný, v závislosti na velikosti parsovaného XML dokumentu. Obecně vzato je DOM vhodný pro relativně menší XML dokumenty, nad kterými může být potřeba rozsáhlejšího dotazování a manipulace. Výhodou rozhraní DOM je možnost ne pouze parsovat, ale i tvořit XML dokumenty. [1]



Obrázek 6 - Schéma funkce rozhraní DOM. [16]

3.1.4.2 **SAX**

SAX (Simple API for XML) je rozhraní založeno na událostech pro sériové parsování XML dokumentu od jeho začátku do konce. SAX parsuje XML dokument sekvenčně uzel po uzlu a ve chvíli, kdy nastane takzvaná událost, je zavolána funkce aplikace spojená s obsluhou dané události a jsou ji předány informace o aktuálně procházeném prvku XML dokumentu. Událostí může být například detekce startovací značky elementu, atributu, textového obsahu v elementu, koncové značky a podobně. Informace o vazbách mezi uzly a struktuře dokumentu si musí v případě potřeby udržovat klientská aplikace.

Velkou výhodou při použití parseru postaveném na rozhraní SAX je jeho rychlost a paměťová nenáročnost, jelikož nenačítá celý dokument do paměti jako DOM, ale pouze klientské aplikaci postupně zpřístupňuje jeho malé části pomocí událostí. Nevýhodou tohoto přístupu ovšem je nemožnost zpětného procházení dokumentu, jelikož SAX jej zpracovává proudově od začátku do konce. Ze stejného důvodu SAX dále neumožňuje přidávat, odstraňovat či editovat jednotlivé uzly, možné je pouze čtení. SAX je tedy výhodné použít v případě paměťových limitací u velkých XML dokumentů či pokud je požadováno rychlé zpracování XML dokumentu, přičemž není potřeba modifikovat jeho strukturu. [6]

3.2 Komprimace XML

Jak již bylo zmíněno v úvodu, jednou z hlavních nevýhod formátu XML je relativně velká velikost XML dokumentů oproti jiným formátům nesoucím stejný obsah. XML syntax obsahuje mnoho redundantního značení a vykazuje vysokou verbozitu – textová data jsou sebepopisná, tudíž každý element obsahuje kromě samotných dat i metadata, která je popisují (např. název elementu) a toto značkování se opakuje pro každý výskyt daného typu dat. Zároveň, syntax jazyka je velmi striktní a vyžaduje například pro každou počáteční značku elementu i značku ukončovací, ve které se opakuje název elementu. Tato verbozita přispívá čitelnosti a tudíž usnadňuje práci s XML dokumentem, nicméně zároveň značně navyšuje jeho velikost. Datová úložiště a přenosové kanály často disponují omezenou kapacitou či jsou s jejich využíváním spojeny zvýšené náklady, schopnost komprimovat XML dokumenty je tedy z hlediska efektivního využití formátu velmi důležitá.

Tato kapitola v první části představuje obecnou charakteristiku datové komprese, v další části jsou představeny možnosti komprese XML dat a v poslední části kapitoly jsou popsány konkrétní komprimační techniky použitelné pro XML.

3.2.1 Datová komprese

Obecně vzato, datová komprese je postup kódování dat způsobem, který umožnuje redukci bitů potřebných pro reprezentaci informací, která nesou. Výstupem komprese je tedy datový soubor menší velikosti než do komprese vstupující originál. Komprimace dat umožňuje úsporu kapacity datových úložišť a urychlení datových přenosů, s určitou formou komprese se tedy lze setkat téměř ve všech sférách informačních a telekomunikačních technologií.

Jedním ze základních principů realizace komprese je vyhledání a odstranění redundantních informací v datech – může se jednat například o nahrazení opakujících se řetězců kratším řetězcem či znakem nebo úplné odstranění informací nepotřebných pro daný účel. Detekce redundance v datech je realizována pomocí takzvaných modelů, které se ji snaží popsat. Datové modely mohou být buďto statické, nebo adaptivní, přičemž statické modely se nepřizpůsobují dle vstupních dat, vycházejí například z obecně předpokládaných frekvencí výskytů jednotlivých znaků či řetězců, kompresor pak kóduje veškerá data stejným způsobem. Statické modely často necharakterizují konkrétní data s dostatečnou přesností

(určitý text může například obsahovat časté opakování obecně málo používaných znaků), čímž dochází k méně efektivní komprimaci. Adaptivní modely jsou utvářeny na základě průchodu a analýzy vstupních dat určených ke kompresi, lze tedy na jejich základě docílit efektivnějšího kódování. Jelikož se při komprimaci i dekomprimaci používá pro tvorbu adaptivního modelu stejná sada pravidel, není potřeba ho ke komprimovaným datům přikládat – při dekompresi vznikne model identický, na základě kterého lze data dekódovat. [17], [20]

Jak je naznačeno v odstavci výše, kompresní techniky zahrnují dva algoritmy – kromě samotného kompresního algoritmu, jehož výstupem je reprezentace vstupních dat o menší velikosti, také algoritmus dekompresní, který komprimovaná data zpětně rekonstruuje. Dle charakteristiky této rekonstrukce se kompresní techniky dělí na dvě základní skupiny – ztrátové a bezztrátové. [17]

3.2.1.1 **Bezztrátová a ztrátová komprese**

Bezztrátová komprese

Techniky bezeztrátové komprese umožňují rekonstrukci originálních dat z komprimované formy v jejich původní podobě, bez ztráty jakýchkoliv informací, nedochází tedy k degradaci kvality dat. Tento způsob komprese je nutné použít v případech, kdy vzhledem k účelu dat není akceptovatelná jakákoliv odchylka mezi původními daty a jejich rekonstruovanou podobou po jejich komprimaci a následné dekomprimaci. Primárně se tedy bezztrátové metody aplikují na spustitelné soubory, zdrojové kódy či textová data, ve kterých je každý znak důležitý pro správnou interpretaci nesené informace. K datům, které je potřeba komprimovat bezztrátově, se tudíž řadí i XML dokumenty.

Ztrátová komprese

U mnoha typů dat není při kompresi potřeba zachovávat veškeré informace, ale lze je dále využívat s drobnými chybami či chybějícími méně potřebnými informacemi oproti originálu. Pokud charakter dat nevyžaduje přesnou rekonstrukci původních dat, ale stačí pouze jejich aproximace, lze docílit mnohem většího (většinou řádově násobného) snížení velikosti původního souboru než u komprese bezeztrátové, jelikož při komprimaci dochází k úplnému a nenávratnému odstranění části dat.

Ztrátové metody se nejčastěji aplikují na obrazová a zvuková data, u kterých bývá odstranění některých dat z původního záznamu pro lidské smysly téměř nebo zcela nedetekovatelné. Například u zvukového záznamu určeného pro poslech člověkem nemá význam uchovávat informace o vyšších frekvencích zachycených pořizovacím zařízením, které jsou pro člověka neslyšitelné. Po aplikaci ztrátové komprese se na zbývající data většinou aplikuje bezeztrátový algoritmus. [17], [18]

3.2.1.2 **Výkonnost komprese**

Kromě ztrátovosti a bezztrátovosti lze u komprimačních metod sledovat řadu dalších vlastností, pomocí kterých je možné zkoumat a hodnotit jejich výkonnost či vhodnost pro konkrétní účel. Nejčastěji se sledují následující kritéria:

- U ztrátových komprimačních metod se často sleduje, jak přesná je rekonstrukce původním datům po dekompresi (míra zkreslení).
- Komplexita komprimační techniky.
- Kapacita operační paměti potřebná pro spuštění a běh komprimační techniky.
- Časová náročnost komprimační techniky na daném počítači – komprese a dekomprese mohou být časově náročné, přičemž je třeba zhodnotit výpočetní čas investovaný do komprese/dekomprese oproti ušetřené úložné kapacitě. Zvláště rychlost dekomprese je v některých případech kritická, například pokud je potřeba rychlý přístup k datům v reálném čase.
- Symetrie/asymetrie komprimační techniky – z hlediska časové náročnosti lze také sledovat, zda se čas běhu kompresního algoritmu značně liší od dekompresního (například pro audiovizuální data, která se kódují a dekódují v reálném čase, jsou vhodné symetrické techniky).
- Míra komprese původního souboru. [19]

Pro určení míry komprese, tedy o kolik menší je reprezentace původního souboru po aplikaci komprimačního algoritmu, se nejčastěji používá kompresní poměr. Jedná se o poměr velikosti nekomprimovaných dat ke komprimovaným datům:

$$kompresní\ poměr = \frac{velikost\ nekomprimovaných\ dat}{velikost\ komprimovaných\ dat}$$

Kompresní poměr se zapisuje ve formě poměru (například 3:1 v případě komprese souboru o velikosti 3 kB na 1 kB).

Kompresní poměr může být také reprezentován jako procentuální hodnota redukce velikosti dat oproti nekomprimované velikosti – v tomto případě se používá označení úspora místa (v příkladu výše by se jednalo o 66,67%).

$$úspora\ místa = (1 - \frac{velikost\ komprimovaných\ dat}{velikost\ nekomprimovaných\ dat}) \times 100[\%]$$

Další metoda, kterou lze použít pro hodnocení dosažené míry komprese, spočívá ve stanovení průměrného počtu bitů potřebného pro reprezentaci jednoho vzorku dat po komprimaci. [17]

3.2.2 Rozdělení komprimačních technik

Z hlediska principu funkce komprimačního algoritmu lze komprimační techniky rozdělit zejména do dvou kategorií – statistické komprimační techniky a slovníkové komprimační techniky. Obsahem této kapitoly je stručné shrnutí nejznámějších komprimačních algoritmů zastupujících obě kategorie.

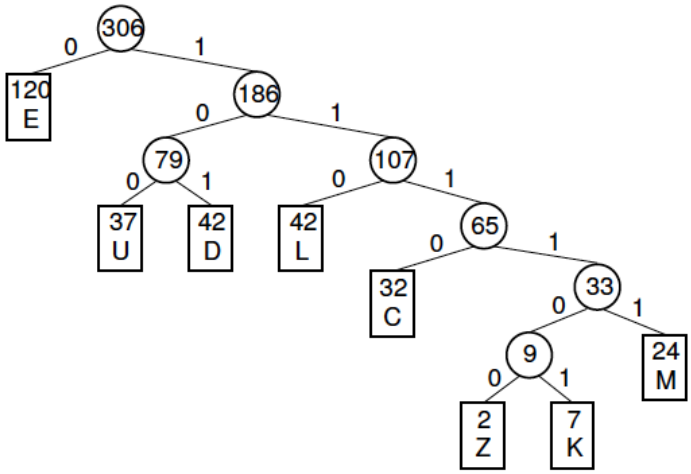
3.2.2.1 Statistické techniky

Statistické metody využívají znalosti pravděpodobnosti výskytu určitého znaku pro snížení počtu bitů potřebných pro jeho reprezentaci. Nejznámějšími statistickými metodami komprese jsou Huffmanovo a aritmetické kódování.

Huffmanovo kódování

Do této kategorie spadá například technika Huffmanova kódování, ve které jsou jednotlivé znaky v datech konvertovány do binárního kódu variabilní délky, přičemž nejčastěji se vyskytující znaky jsou nahrazeny nejkratšími kódy, zatímco znaky, jejichž pravděpodobnost výskytu je nižší, jsou nahrazeny kódem delším. Technika spočívá v sestavení binárního stromu, jehož uzly mají hodnotu frekvence výskytu a listy představují jednotlivé znaky dané abecedy. Strom je budován od konce dle pořadí frekvence výskytu

znaků, kdy listy v nejhlubší úrovni stromu reprezentují dva znaky s nejnižší frekvencí výskytu. Průchodem stromu po hranách, které jsou po jedné straně hodnoceny *1* a po druhé *0*, vzniká výsledný kód. [24]



Obrázek 7 - Příklad Huffmanova binárního stromu. [24]

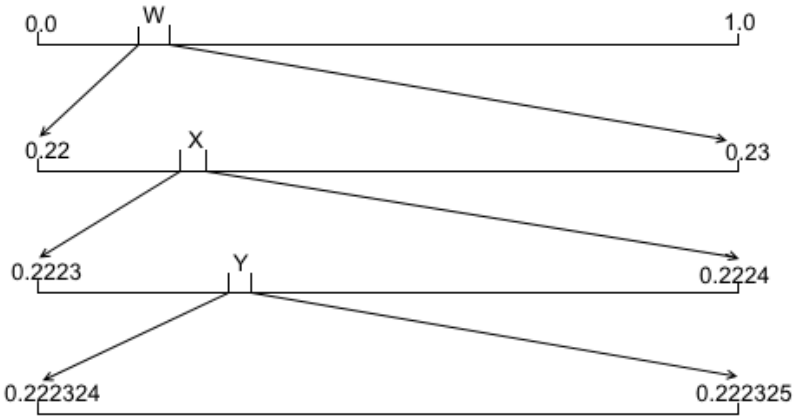
V případě neznalosti statistického rozložení znaků v datech musí při Huffmanově kódování nejprve dojít k prvnímu průchodu daty za účelem jejich analýzy a až při druhém průchodu dojde k jejich zakódování – z tohoto důvodu vznikla adaptivní varianta Huffmanovy techniky, kdy je pravděpodobnost dynamicky kalkulována na základě statistiky již prošlých znaků. [19]

Aritmetické kódování

Mezi statistické komprimační techniky patří také například metoda aritmetického kódování, která stejně jako Huffmanovo kódování umožňuje znakům s častým výskytem být reprezentováno méně bity. Aritmetické kódování nepřiděluje kódy jednotlivým znakům, nýbrž kóduje celý řetězec vstupních znaků do jediného zlomkového čísla (čísla s pohyblivou řádovou čárkou) v intervalu <0; 1).

Každému symbolu abecedy je v modelu aritmetického kódování přiřazen unikátní segment intervalu (čím větší pravděpodobnost výskytu, tím větší část intervalu je danému symbolu přiřazena). Při zahájení aritmetického kódování pak kompresor interval <0; 1) s každým načteným symbolem postupně zužuje výpočtem nové spodní a horní hranice intervalu, které poměrově odpovídají původnímu vymezení symbolu na intervalu <0;1). Po

průchodu všech vstupních symbolů je řetězec kódován libovolným číslem z konečného intervalu. [25]



Obrázek 8 - Vizualizace aritmetického kódování s modelem rovnoměrné pravděpodobnosti výskytu. [25]

Obecně platí, že aritmetické kódování poskytuje lepší kompresní poměry než Huffmanovo, je ovšem výpočetně náročnější, tudíž komprese trvá déle. Algoritmus Huffmanova kódování je také méně komplexní a tudíž jednodušší pro implementaci. [26]

3.2.2.2 **Slovníkové techniky**

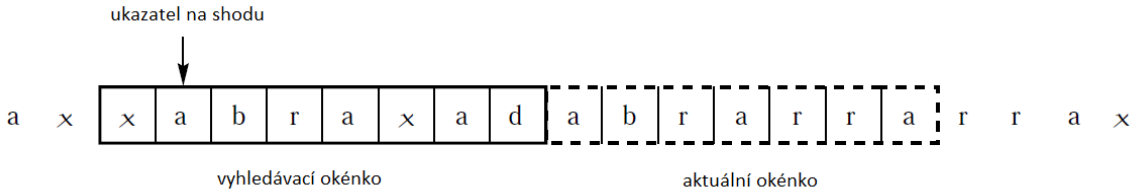
Slovníkové komprimační techniky využívají výskytu opakujících se posloupností symbolů v datech (například opakování slov v textu). Kompresor si při průchodu daty vytváří seznam (slovník) a v případě, že narazí na shodnou posloupnost symbolů se záznamem již se vyskytujícím ve slovníku, provede její zakódování vytvořením odkazu (ukazatele) na daný slovníkový záznam.

Výhodou oproti statistickým metodám je, že před komprimací není potřebná znalost statistického rozložení symbolů v datech či jiných vlastností vstupních dat, není tedy třeba data předběžně procházet a analyzovat či provádět náročné dynamické propočty během samotné komprimace.

Mezi nejpoužívanější slovníkové komprimační algoritmy patří LZ77 (Lempel-Ziv, pojmenovaný podle svých autorů a publikovaný v roce 1977) a stejnými autory o rok později publikovaný LZ78. Tyto metody mají mnoho variací, přičemž většina dnes používaných komprimačních technik vychází z jednoho či druhého z nich.

LZ77

Metoda LZ77 používá pro průchod vstupními daty takzvané klouzavé okénko skládající se ze dvou částí – vyhledávacího okénka (search buffer), obsahujícího část již prošlých dat, a aktuálního okénka (look-ahead buffer), které obsahuje následující symboly ke kódování. Pro zakódování sekvence symbolů v aktuálním okénku algoritmus prochází vyhledávací okénko, dokud nenarazí na shodu s prvním symbolem v aktuálním okénku, načež jsou prověřeny následující symboly, zda se také neshodují. Po nalezení nejdelší shody ve vyhledávacím okně je provedeno zakódování aktuálního řetězce symbolů pomocí přímého odkazu na předešlý shodný řetězec. Odkaz se skládá z trojice hodnot – vzdálenosti začátku předešlého shodného řetězce od aktuálního okna, délky shodného řetězce a prvního symbolu následujícího po řetězci v aktuálním okně (nutný pro případ nenalezení shody pro vstupní znak ve vyhledávacím okénku). [17], [19]



Obrázek 9 - ukázka klouzavého okénka metody LZ77 [17] s úpravami.

LZSS

Mezi nejznámější modifikace algoritmu LZ77 patří LZSS (Lempel-Ziv-Storer-Szymanski). Algoritmus LZSS řeší nevýhodu LZ77, kdy jsou pomocí odkazu zmíněného výše kódovány všechny řetězce, včetně jednotlivých znaků bez nalezené shody, přičemž zakódování jednotlivých symbolů či krátkých řetězců může vést k expanzi namísto komprese. Je tedy nastavena minimální délka nalezené shody, při jejímž nedosažení je ponechán vstupní znak bez zakódování do podoby reference. Před každý zpracovaný vstup je také zapsán příznakový bit, aby mohl dekodér rozlišit, zda se jedná o nezakódovaný znak či zakódovaný řetězec ve formě odkazu. V případě LZSS již zároveň není třeba v odkazu zapisovat první znak za shodným řetězcem, čímž je kód zkrácen. [17], [18]

Deflate

Zřejmě nejpoužívanější a velmi efektivní metodou založenou na LZ77 je algoritmus Deflate, který kombinuje optimalizované řešení slovníkové metody LZ77 se statistickou metodou Huffmanova kódování. Vstupní data jsou rozdělena do bloků různých velikostí, přičemž každý blok je komprimován jedním ze tří módů. Blok může být buďto ponechán bez komprese (vhodné například pro komprimovaná data, kde by pokus o další kompresi vedl k expanzi) či komprimován metodou LZ77 a následně Huffmanovým kódováním staticky, za užití stromů předdefinovaných ve specifikaci Deflate či dynamicky, kdy stromy vytváří kompresor. Pro každý komprimovaný blok tedy algoritmus nejprve za užití klouzavého okénka vyhledá duplicitní řetězce a nahradí je referencemi na předchozí výskyt. Pokud pro řetězec není shoda nalezena v posledních 32 KB (velikost okénka – odkazováno může být i na řetězec v předchozím bloku), je poslán nezakódovaný znak. Následně jsou v daném bloku nekódované znaky a délky shodných řetězců kódovány jedním Huffmanovým stromem a vzdálenosti shodných řetězců druhým. [27]

LZ78

Oproti LZ77 o rok později vydaný algoritmus LZ78 nevyhledává shodné fráze pomocí klouzavého okénka, nýbrž buduje slovník jakožto samostatnou stromovou datovou strukturu. Vstupní řetězce jsou kódovány jako dvojice indexu nejdelšího shodného záznamu ve slovníku a prvního vstupního znaku nenacházejícího se v daném záznamu, přičemž se tato dvojice zároveň stává dalším záznamem ve slovníku. Velkou výhodou oproti algoritmu LZ77 je skutečnost, že hledání shodné sekvence pomocí LZ78 není omezeno pouze na vyhledávací okno klouzavého okénka, nicméně v závislosti na velikosti komprimovaných dat může být budování a udržování slovníku paměťově velmi náročné. [17]

LZW

Jednou z nejznámějších modifikací metody LZ78 je algoritmus LZW (Lempel-Ziv-Welch), který umožňuje kódovat řetězce pouze hodnotou indexu odkazujícího na záznam ve slovníku, lze tedy zkrátit kód řetězce o druhý element dvojice používané metodou LZ78. Toho je docíleno tím, že slovník před začátkem komprimace již obsahuje všechny znaky abecedy komprimovaného obsahu. Při nalezení nejdelšího slovníkového záznamu shodujícího se se vstupním řetězcem je vstupní řetězec kódován pouze indexem záznamu ve

slovníku a odstraněn ze vstupu, daný záznam je pak zřetězen s následujícím znakem na vstupu a přidán do slovníku. [19]

3.2.3 Komprimační metody XML

V této kapitole jsou nejprve představeny možné způsoby rozdělení komprimačních technik zaměřených na kompresi XML dat na základě jejich vlastností a zaměření. Dále jsou představeny jednotlivé vybrané techniky, které jsou v současnosti ke kompresi XML dat využívány.

3.2.3.1 Klasifikace XML kompresorů

XML kompresory lze klasifikovat mnoha různými způsoby. Mezi nejpoužívanější patří dělení dle schopnosti vnímat strukturu XML dokumentu či klasifikace na základě podpory dotazování nad komprimovanými daty.

Z hlediska schopnosti vnímat strukturu XML dokumentu lze kompresní techniky dělit na dvě základní skupiny – obecné textové kompresory a takzvané XML-aware, nebo XML-conscious (přeložitelné jako XML-vnímající) kompresory. [21]

Obecné textové kompresní techniky

XML dokumenty jsou tvořeny posloupnostmi Unicode znaků, jedná se tedy o čistě textové soubory, na které lze aplikovat standardně používané bezztrátové techniky pro kompresi textu. Tyto techniky nevnímají vnitřní strukturu XML dokumentu, nicméně bývají vyvíjeny mnohem déle, než XML-conscious kompresory, lze tedy předpokládat vyšší míru optimalizace.

XML-conscious kompresní techniky

Jedná se o kompresory, které jsou schopny vnímat vnitřní strukturu XML dokumentu. Tyto kompresní techniky jsou specializovány na kompresi XML dat, a tudíž mohou využít znalosti jejich vlastností k dosažení vyšších kompresních poměrů, než obecné kompresní metody. XML-conscious metody se dále rozlišují dle závislosti na znalosti schématu:

- **Závislé na schématu** – tyto kompresní techniky využívají znalosti DTD nebo schématu XML dokumentu, což v určitých případech může vést k dosažení lepších kompresních poměrů. Kompresní techniky vyžadující znalost

schématu obecně nejsou příliš využívány, jelikož pro jejich správnou funkci musí mít kodér i dekodér k dispozici schéma, jehož dostupnost není vždy garantována.

- **Nezávislé na schématu** – tyto kompresory při aplikaci standardních komprimačních technik berou v potaz strukturu XML dokumentu k dosažení vyšších kompresních poměrů oproti obecným textovým technikám, přičemž nevyžadují znalost schématu. V praxi jsou mnohem rozšířenější, než závislé kompresory. [28]

Klasifikace dle podpory dotazování

Druhým způsobem klasifikace XML kompresních metod je rozdělení na techniky podporující či nepodporující dotazování nad komprimovaný daty. Kompresní techniky, které nepodporují dotazování, jsou primárně určeny k archivaci dat, jejich záměrem je tedy docílit co nejvyšší míry komprese. V této kategorii jsou zároveň všechny obecné textové komprimační techniky, jelikož strukturu XML neuvažují.

Kompresory s podporou dotazování umožňují nad komprimovanými daty práci s jednotlivými elementy či atributy dokumentu prostřednictvím dotazovacích jazyků, jakým je například XPath. Jedná se o velmi důležitou vlastnost pro umožnění práce s XML dokumenty v aplikacích či na zařízeních, která jsou paměťově omezena, tudíž není možné provést dekompresi celého XML dokumentu a sestavení jeho DOM stromové struktury v paměti. Kompresní poměry technik v této skupině jsou obecně horší než u kompresorů bez podpory dotazování. [28], [29]

Klasifikace dle homomorfismu

XML vnímající komprimační techniky lze také klasifikovat dle přístupu ke zpracování XML dat během komprese na homomorfní a nehomomorfní. Homomorfní kompresní techniky zachovávají původní strukturu XML dokumentu. Výsledný komprimovaný formát dokumentu lze parsovat a nad daty se dotazovat stejným způsobem, jako nad původním nekomprimovaným dokumentem, lze tedy například i editovat jednotlivé uzly. Dále je například možné komprimovaná data validovat pomocí schématu.

Nehomomorfní komprimační techniky rozdělují strukturu XML dokumentu (značkování) a jeho znaková data (obsah). Data jsou dále rozdělena dle svého typu do sémanticky podobných kontejnerů, které jsou poté každý komprimován samostatně. Na základě znalosti typu dat v každém kontejneru lze zvolit optimální komprimační techniku pro každý zvlášť, čímž lze maximalizovat míru komprese celého dokumentu. Nevýhodou nehomomorfních metod ovšem je, že výsledný komprimovaný formát dokumentu svou strukturou neodpovídá původní nekomprimované verzi. [29]

Online/offline komprese

Dalším způsobem dělení XML komprese je na online a offline kompresní techniky. Online XML kompresory jsou schopny zpracovávat XML dokument proudově v reálném čase, lze tedy data kódovat a dekódovat tak, jak po částech přicházejí na vstup, zatímco offline kompresory vyžadují pro zahájení kódování či dekódování celý dokument. Online komprese například umožnuje zkrácení zpoždění při přenosu dat mezi systémy po síti, kdy dekodér na straně příjemce může okamžitě začít s dekompresí a nemusí čekat na přenesení celého komprimovaného dokumentu. Online kompresory mají také využití v případech, kdy zařízení, na kterém je dokument zpracováván, disponuje limitovanými zdroji, a tudíž není možná dostupnost celého dokumentu. Offline komprese obecně dosahuje vyšší efektivity oproti online technikám z hlediska kompresního poměru. [30], [31]

3.2.3.2 Obecné textové komprimační techniky

Jak již bylo zmíněno v přechozí kapitole, nejjednodušší přístup ke komprimaci XML dat je jejich komprese jakožto textu, bez rozlišování znakových dat a značkování dokumentu, pomocí standardních statistických a slovníkových kompresních metod. Mezi nejrozšířenější bezztrátové kompresní techniky textu, na kterých jsou založeny i mnohé specializované XML kompresory, patří Gzip, bzip2 a PPM. [28]

Gzip

Gzip (GNU Zip) je rozšířená proudová bezztrátová komprimační technika, která je postavena na metodě Deflate, tudíž využívá kombinace slovníkové metody LZ77 pro nahrazení opakovaných výskytů řetězců odkazy na předchozí výskyt a Huffmanova kódování pro přidělení nejkratších kódů nejčastěji se vyskytujícím znakům.

Původně Gzip vzniknul za účelem nahrazení Unixového nástroje compress, který byl v té době ohrožen patenty na algoritmus LZW, který využívá. Jelikož Gzip dosahuje vyšších kompresních poměrů oproti compress, je dodnes velmi rozšířený. [32]

Bzip2

Bzip2 je kompresní program vyvinutý Julianem Sewardem a poprvé představen v roce 1996. Ke kompresi používá několik vrstev různých bezztrátových kompresních technik. Bzip2, na rozdíl od metody Gzip, komprimuje data po na sobě nezávislých blocích o velikostech 100 až 900 kB, což může být výhodné, zvláště pokud dojde k poškození části dat.

Data jsou nejprve kódována velmi jednoduchou metodou RLE (Run-length encoding), která nahradí sekvence čtyř až dvě stě padesáti pěti po sobě jdoucích opakujících se znaků čtveřicemi opakujících se znaků a počtem jejich dalších opakování za první čtveřicí. Následně je aplikována Burrows-Wheeler transformace (BWT), která permutuje symboly vstupního řetězce způsobem, kdy opakující se podřetězce vstupního řetězce jsou na výstupu konvertovány do několika sekvencí po sobě jdoucích stejných symbolů. Burrows-Wheeler transformace vytváří výhodnější pořadí symbolů pro komprimaci a je plně reverzibilní, přičemž jedinou přidanou informací ke vstupním datům je původní pozice symbolů.

Výsledný blok je následně dále transformován metodou MTF (Move-to-front), která proudově kóduje znaky řetězce jejich pozičními indexy z řazeného seznamu znaků příslušné abecedy, načež je daný znak posunut na začátek seznamu a ten je celý přeindexován. Premisou je, že pokud se některé znaky lokálně často opakují, výstup bude tvořen malými čísly menšího rozsahu. Výstup transformací s nižší entropií je následně opět kódován metodou RLE. V další fázi je pak aplikováno Huffmanovo kódování (původní verze bzip používala aritmetické kódování).

Pro dekompresi jsou jednotlivé vrstvy procházeny v opačném pořadí. Bzip2 je asymetrický kompresor, jelikož dekomprese je mnohem rychlejší než komprese. Bzip2 obecně dosahuje lepších kompresních poměrů, než kompresory založené na metodách Deflate nebo LZW, jako například Gzip, nicméně je z důvodu své výrazné komplexity a aplikace transformací pomalejší. [33]

PPM

Prediction by partial matching (PPM) je statistická kontextová metoda komprese dat, která používá adaptivní modely a predikce a tvoří základ mnoha XML vnímajících kompresorů. PPM bylo poprvé představeno již v roce 1984, od té doby však vzniklo mnoho variací jeho implementace.

PPM předpovídá nadcházející znak vstupního proudu na základě množiny předchozích znaků, k čemuž využívá kontextové modely. Kontextový model PPM udržuje statistiku počtu výskytů všech znaků, které se vyskytly za každou sekvencí určité délky, která již byla zaznamenána na vstupu. Na základě udržované statistiky je pro každou sekvenci v kontextu počítána pravděpodobnost výskytů jednotlivých znaků, které ji následují. Délka sekvence předcházející vstupnímu znaku udává stupeň kontextového modelu, přičemž PPM využívá kombinaci několika modelů od nultého do maximálního stanoveného stupně.

PPM kombinuje vypočítané pravděpodobností rozdělení modelů všech stupňů pro určení výsledného pravděpodobnostního rozdělení, na základě kterého je znak vyskytující se na vstupu zakódován aritmetickým kódováním. Výchozím modelem pro aritmetické kódování je kontextový model nejvyššího stupně, přičemž pokud znak na vstupu zatím nebyl v daném kontextu zaznamenán, je zakódován únikový znak, který signalizuje přechod do modelu následujícího nižšího stupně. Tento proces pokračuje, dokud není nalezen kontext, který obsahuje daný znak a pravděpodobnost jeho výskytu, podle kterého jej lze zakódovat. V případě, že vstupní znak není doposud obsažen v žádném kontextu, je kódován podle modelu, který udává rovnoměrné rozložení pravděpodobnosti výskytu všem znakům použité abecedy.

V závislosti na implementaci je buďto stanoven maximální stupeň kontextového modelu nebo je délka neomezena. Zvláště při použití vyšších stupňů kontextových modelů se PPM stává časově a paměťově velmi náročné oproti jiným kompresním metodám, dosahuje však nejlepších kompresních poměrů. [17], [34]

3.2.3.3 XML komprimační techniky bez podpory dotazování

Z hlediska funkcionality lze považovat za primární způsob klasifikace XML vnímajících kompresorů na podporující a nepodporující dotazování nad komprimovanými daty. Jak již bylo zmíněno výše, komprimační techniky bez podpory dotazování vyžadují před zpracováním dotazu nad XML daty jejich úplnou dekompresi, což je náročné na paměť a výpočetní čas. Tyto techniky však obecně dosahují lepších kompresních poměrů a tudíž jsou primárně využívány pro datovou archivaci, která by měla dosahovat vyšší efektivity než obecné textové kompresory. Mezi kompresní techniky v této kategorii patří XMill, XMLPPM, DTDPPM, SCMPPM či Millau. [22]

XMill

XMill je jeden z prvních kompresorů specializovaných na XML, přičemž jeho primárním cílem je docílení maximální komprese pro využití v datové výměně a archivaci. Dle svých autorů dosahuje až dvounásobné míry komprese při podobných rychlostech jako obecné textové kompresory typu Gzip. [35]

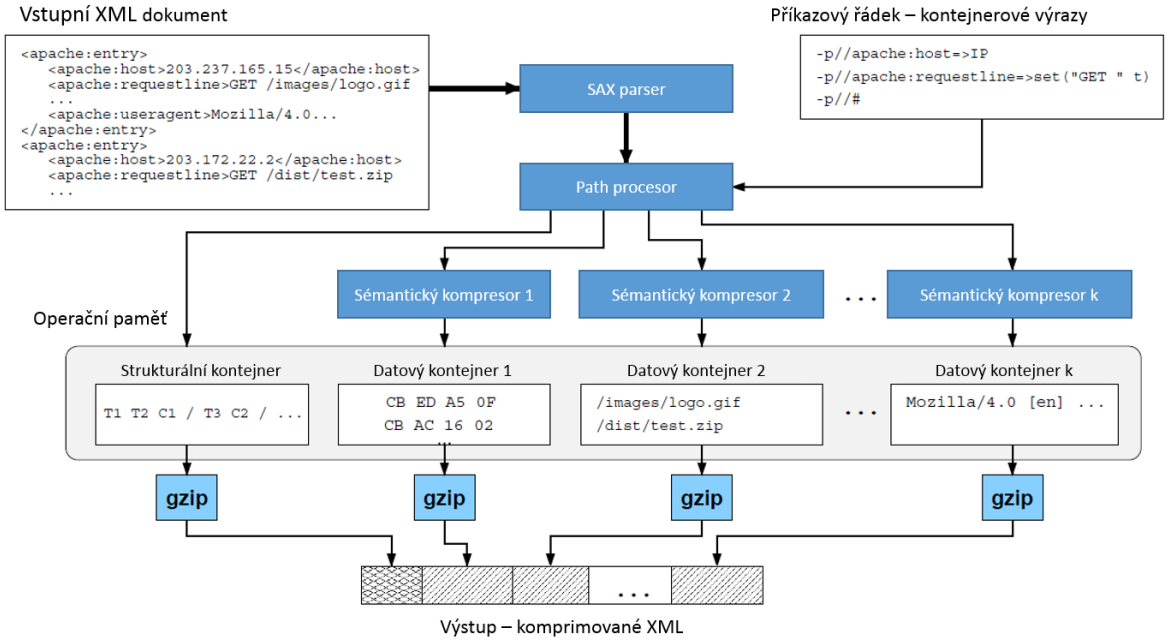
XMill patří do kategorie nehomomorfních kompresorů, separuje tedy strukturu XML dokumentu, která se skládá z názvů značek a atributů, od datového obsahu elementů a atributů, přičemž obě části komprimuje zvlášť. Pro samotnou kompresi XMill využívá primárně zlib (knihovna, kterou používá Gzip), specializované kompresní techniky pro určité datové typy a umožňuje i rozšíření o uživatelem definované kompresory. Pro provedení komprese XMill nevyžaduje DTD či schéma dokumentu, patří tedy do skupiny kompresorů nezávislých na schématu.

Proces, kterým XMill data komprimuje, je složen ze tří základních částí. Nejprve jsou XML data parsována pomocí SAX parseru, který odesílá tokeny (značky, atributy, datové hodnoty) do takzvaného path procesoru, který jednotlivé tokeny rozděluje do různých kontejnerů. Strukturální data jsou odeslána do strukturálního kontejneru a zakódována slovníkovým způsobem, ve kterém jsou názvy elementů a atributů nahrazeny slovníkovými indexy. Zbývající data jsou rozdělena do různých kontejnerů na základě předpokládané sémantické podobnosti.

Rozdělení obsahových dat do jednotlivých kontejnerů XMill provádí dvěma způsoby. Výchozím způsobem je seskupování dat se stejným názvem značky či atributu do jednoho

kontejneru. Druhý způsob umožňuje nastavení způsobu alokace dat do jednotlivých kontejnerů uživatelem, k čemuž XMill využívá takzvané kontejnerové výrazy, které lze použít jako parametry v příkazové řádce při spuštění XMill komprese. Prostřednictvím těchto výrazů lze také nastavit specifické sémantické kompresory pro komprimaci jednotlivých kontejnerů, kterými je možné pro daný typ dat dosáhnout lepší komprese (lze například použít sémantickou kompresi pro kladná celá čísla). Dále lze odkázat na vlastní uživatelem definované kompresory, jak bylo zmíněno výše, pro optimální kompresi vysoce specializovaných dat. [35]

V poslední fázi jsou data v každém kontejneru separátně kódována metodou Gzip (novější verze XMill však přidala i podporu bzip2 a PPM) a následně na výstupu opět zřetězena dohromady. Vysoká míra uživatelské přizpůsobitelnosti, unikátní pro XMill, umožňuje výrazné zlepšení efektivity komprese, zvláště u vysoce specializovaných dat, nicméně po uživateli vyžaduje dobrou znalost komprimovaného obsahu XML dokumentu. [22], [28]



Obrázek 10 - Architektura a příklad použití XMill [35] s úpravami.

XMLPPM

XMLPPM je proudový XML kompresor založený na obecné statistické kompresní metodě PPM, metodě modelování dat MHM (multiplexované hierarchické modelování) a kódování výstupu SAX parsování. Jedná se o nehomomorfní kompresní metodu nevyžadující znalost schématu.

XML dokument je nejprve parsován SAX parserem. Pro každou generovanou událost jsou počáteční a koncové značky elementů, názvy atributů či počátek a konec znakových dat kódovány jednotlivými byty, přičemž pokud se jedná o první výskyt sekvence znaků, jsou za kódové označení přidány nekódované znaky sloužící jako reference při dalším výskytu. Takto kódovaný proud událostí SAX parseru se nazývá ESAX (encoded SAX).

V následující fázi se využívá metody MHM. Zakódované ESAX tokeny jsou předávány jednomu ze čtyř multiplexovaných PPM modelů na základě jejich syntaktické struktury (modely jsou udržovány pro názvy elementů a atributů, strukturu elementů, hodnoty atributů a řetězce obsažené v elementech), čímž je docíleno přesnější predikce, než při použití jediného modelu. Každý model si udržuje vlastní vnitřní stav, všechny čtyři ovšem sdílí stejný aritmetický kodér, který generuje finální proud komprimovaných dat. Jelikož separace tokenů do různých modelů narušuje jejich závislosti mezi různými syntaktickými skupinami (například data ohraničena určitou značkou s ní v XML často silně korelují), jsou do modelů před zakódované elementy, atributy a řetězce vloženy indexy tokenů značek, pod které spadají, čímž jsou PPM modely informovány o existujících závislostech umožňujících další zpřesnění predikcí. [36]

Z důvodu využívání PPM dosahuje XMLPPM obecně vyšší míry komprese, než XMill při použití Gzip komprimace a ve výchozím nastavení komprese, nicméně ze stejného důvodu je XMLPPM výpočetně náročné a tudíž pomalejší. XMLPPM také na rozdíl od XMill umožňuje online zpracování komprimovaných dat. [22]

DTDPPM

V roce 2005 představil autor XMLPPM rozšíření této komprimační techniky s názvem DTDPPM. Jedná se o optimalizaci vyžadující znalost DTD komprimovaného XML dokumentu, která přinesla zlepšení efektivity komprese primárně na malých vysoce

strukturovaných dokumentech, u větších dokumentů byl benefit však méně znatelný, přičemž DTDPPM vyžaduje komplexnější implementaci a dostupnost DTD. DTDPPM přineslo celkem čtyři DTD-specifické optimalizace, mezi které patří například možnost vynechání nadbytečných bílých znaků v obsahu elementu či schopnost sestavit tabulku referencí na názvy elementů a atributů z DTD namísto zapisování textu těchto značek na výstup za jejich kódy při prvním výskytu. [37]

SCMPPM

Dalším rozšířením XMLPPM je například SCMPPM (Structural context modeling PPM). Jedná se o variantu, která používá metodu strukturálního kontextového modelování – namísto přepínání mezi několika modely na základě syntaktické třídy dat, SCMPPM používá samostatný model pro textový obsah každého elementu určitého názvu. Premisou tohoto rozdělení je, že znaková data ohraničena stejnou značkou (stejný název elementu) budou mít podobnou sémantiku, čímž lze docílit lepší míry komprese (stejného předpokladu využívá MHM vkládáním indexů značek před kódovaná data v modelech či XMill rozdělováním dat do kontejnerů). Z důvodu udržování mnoha modelů, zvláště u větších vysoce strukturovaných dokumentů, je SCMPPM velmi paměťově náročné. [38]

Millau

Millau je technika pro komprimaci XML dokumentů, která využívá znalost DTD pro dosažení lepšího kompresního poměru (dostupnost DTD však není nutností). Hlavním principem této kompresní techniky je kódování pouze těch informací XML dokumentu, které nelze odvodit z asociovaného DTD. Mezi tyto informace patří datové hodnoty (znaková data elementů a hodnoty atributů) a strukturální informace v případě použití operátorů v rámci DTD pro volitelnost, výběr či nespecifikovaný počet opakování, při jejichž použití danou část struktury dokumentu nelze určit. Millau je nehomomorfní metoda, která parsuje DOM strom vstupního XML dokumentu a DTD strom zároveň, načež je výstup parsování rozdělen na strukturální a obsahový proud. Tato kompresní technika dosahuje při dostupnosti DTD dobrých kompresních poměrů u větších dokumentů, nicméně je u velkých XML dokumentů zároveň velmi náročná na paměť z důvodu generování DOM stromové struktury. [22]

Millau je nadstavbou WBXML (WAP Binary XML), což je slovníková technika pro kompaktní binární reprezentaci XML určená primárně pro datový přenos, která XML značky, atributy a hodnoty atributů nahrazuje bytovými tokeny. [29]

XML-WRT

XML Word Replacing Transform (zkráceně XWRT nebo XML-WRT) je další kompresor, který využívá podobného principu jako XMill. V prvním průchodu daty je sestaven slovník, který obsahuje všechny nepřekrývající se sekvence znaků písmen o minimální délce dvou znaků, které se v dokumentu vyskytují aspoň šestkrát. Slovník obsahuje také například počáteční značky, před kterými může být jeden či více znaků mezer, adresy URL a e-mailové adresy. Vytvořený slovník je následně zapsán na začátek výstupního souboru. Dále jsou časté fráze vyskytující se například v okolí atributů jako =" a "> či koncové značky elementů (případně společně s označením nového řádku) nahrazeny jednobytovými kódy. Další úpravou je odstranění jednotlivých znaků mezer před kódovanými slovy v textovém obsahu, které jsou následně při dekompresi opětovně automaticky vloženy. V neposlední řadě je použit samostatný model pro kompaktnější kódování číslic.

XML-WRT také od verze 2.0 rozděluje data XML dokumentu do různých kontejnerů podle názvů značky a další specializované kontejnery jsou vyhrazeny pro kalendářní data, časy či zlomková čísla, přičemž je následně každý kontejner komprimován separátně jako tomu je u XMill. Transformace vstupních dat provedená XWRT je primárně optimalizována pro následnou kompresi zlib, nicméně kompresor umožňuje také nastavení konečné kompresní techniky na LZMA[7], PPM či PAQ[8]. [49], [51]

EXI

EXI (Efficient XML Interchange) je W3C doporučením, které představuje binární XML formát a jednu z nejnovějších metod komprimace XML dat, jehož primárním cílem je urychlení přenosu XML dat po síti. EXI umožňuje kombinaci binárního kódování a obecné Gzip (Deflate) komprese k docílení vyšších kompresních poměrů. EXI také může využít

[7] Bezeztrátový kompresní algoritmus využívají slovníkovou kompresi podobnou LZ77, původně vyvinutý pro archivační program 7-Zip.
[8] Kompresor podobný PPM využívající aritmetické kódování.

informací o datových typech a omezeních definovaných v DTD či schématu k docílení dalšího zlepšení komprese, nicméně jejich přítomnost není vyžadována.

Obsah XML dokumentu je reprezentován prostřednictvím EXI jako takzvaný EXI proud, který se skládá z hlavičky a těla, přičemž hlavička obsahuje informace o nastavení použitém pro kódování, potřebné k dekódování těla. Tělo EXI proudu se skládá ze sekvence zakódovaných událostí (podobných SAX událostem), které tvoří dokument. Kódování EXI probíhá na základě formálních gramatik, které jsou odvozeny od schématu dokumentu, v případě nepřítomnosti schématu jsou použity defaultní gramatiky. Na základě gramatiky jsou určeny pravděpodobnosti výskytu událostí v daných místech proudu, přičemž nejpravděpodobnější alternativy jsou kódovány méně bity. Pro další zvýšení komprese může být EXI proud heterogenních dat multiplexován do kanálů sémanticky podobných dat – elementy a atributy jsou seskupovány na základě kvalifikovaných názvů[9] (mále kanály jsou následně opět sdruženy pro minimalizaci organizačních informací) a struktura (kódy událostí) zvlášť, podobně jako tomu je například u XMill kontejnerů. Následně je na jednotlivé kanály aplikován algoritmus Deflate. [41]

EXI nabízí další nastavení, jako například volbu mezi striktní a nestriktní interpretací schématu, kdy striktní vyžaduje validitu kódovaných instancí a deviace od schématu způsobí chybu kódování, umožňuje však na základě schématu přesné predikce následujících elementů a jejich datových typů, vedoucí k efektivnějšímu kódování. Nevýhodou EXI však je, že k dosažení vyšší míry komprese, než umožňují jiné dostupné XML vnímající kompresory, je potřebná plná značně komplexní implementace a validita kódovaného dokumentu, jejíž ověření je další procesně náročnou operací. [29], [41]

3.2.3.4 **XML komprimační techniky s podporou dotazování**

Tato kapitola se věnuje XML vnímajícím kompresorům, které umožňují zpracování dotazů nad XML daty v komprimované podobě. Hlavním cílem těchto kompresorů není docílení maximální míry komprese, nýbrž umožnění práce s XML dokumentem bez nutnosti jeho plné dekomprese, což je stěžejní, jak již bylo zmíněno, zejména pro zařízení

[9] Kombinace prefixu a lokálního jména elementu/atributu tvořící jejich jednoznačné označení.

s limitovanou paměťovou kapacitou či pro urychlení práce s XML daty. Mezi nejznámější kompresory v této kategorii patří například XGrind, XMLZip či XPress.

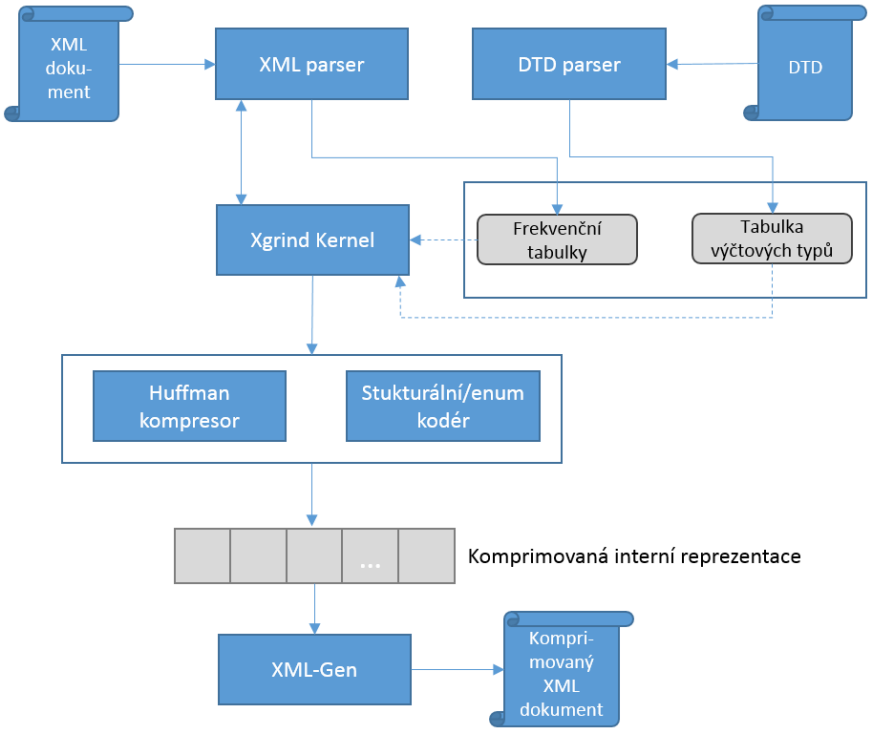
XGrind

XGrind byl prvním XML-conscious kompresorem, který umožňuje zpracovávání jednoduchých dotazů bez nutnosti úplné dekomprese dat. Jedná se o homomorfní kompresní metodu, kdy výsledný komprimovaný formát zachovává syntaktickou strukturu původního XML dokumentu. Komprimovanou verzi dokumentu tedy lze také například parsovat DOM či SAX parsery bez provedení dekomprese.

XGrind kóduje strukturu dokumentu slovníkovým způsobem velmi podobně, jako tomu je u XMill – počáteční značky elementů a názvy atributů jsou kódovány jako *T*, respektive *A* následované číselným indexem a koncové značky lomítkem. Dále XGrind na základě DTD dokumentu sestrojí slovník pro atributy výčtového typu, které kóduje samostatně od ostatních dat. Komprimace znakových dat obsažených v elementech a hodnot ostatních typů atributů je prováděna statickým bezkontextovým Huffmanovým kódováním, přičemž tabulky frekvence výskytů pro jednotlivé elementy a atributy jsou inicializovány na základě DTD.

Samotný proces komprimace XGrind realizuje ve dvou krocích – nejprve parsuje XML dokument pro získání statistik o četnosti výskytů znaků, na základě kterých jsou naplněny modely pro statické Huffmanovo kódování nezávislé na kontextu (pro možnost dotazování nad komprimovanými daty musí být použito kompresní schéma, ve kterém kód přiřazený řetězci není závislý na jeho pozici v dokumentu). V této fázi XGrind stejně jako předešlé kompresory využívá předpokladu sémantické podobnosti a podobné distribuce znaků v jednotlivých elementech a atributech a používá separátní tabulku frekvence výskytů pro každý element a nevýčtový atribut. Následně XGrind provede druhé parsování dokumentu, ve kterém jsou značky a názvy atributů zakódovány indexy odkazujícími na slovníkové záznamy dle metody popsané výše, zakóduje výčtové atributy podle tabulky sestrojené na základě DTD a podle příslušných Huffmanových stromů zakóduje jednotlivá znaková data elementů a hodnoty ostatních atributů. Výstup kódování se slovníkem a všemi potřebnými tabulkami tvoří komprimovanou interní reprezentaci kompresoru, která je za běhu předávána modulu, který z ní generuje finální komprimovaný dokument. Jelikož je

komprimovaný dokument homomorfní transformací vstupního XML dokumentu, lze jím navigovat stejným způsobem, jako originálním dokumentem. Dotazy na přesnou shodu a shodu prefixu mohou být prováděny přímo nad komprimovaným dokumentem – cesta a predikát dotazu jsou konvertovány do odpovídající komprimované podoby. Pro dotazy na rozsah či částečnou shodu je pro vyhodnocení dotazu nutná dekomprese hodnot elementů či atributů, které jsou součástí predikátu dotazu, přičemž komprimovaná je pouze cesta dotazu. Komprimovaný dokument lze také například aktualizovat či validovat oproti komprimované verzi jeho DTD. [22], [39]



Obrázek 11 - Architektura XGrind, vlastní tvorba.

XMLZip

Hlavním principem kompresní techniky XMLZip je rozdělení XML DOM stromové struktury v určité hloubce do několika podstromů. Jednotlivé větve pod stanovenou hloubkou rozdělení jsou samostatně komprimovány metodou Gzip, zatímco část stromu nad danou hloubkou, označována jako kořenová, je ponechána bez komprimace. Hloubka rozdělení je uživatelsky nastavitelná. Výstupem XMLZip je archiv, který obsahuje kódovaný soubor ve formě nekomprimované kořenové části XML dokumentu s odkazy na jednotlivé komprimované větve v archivu ve formě elementů s různými identifikátory,

samotné soubory obsahující komprimované podstromy a soubor obsahující mapování odkazujících elementů ke komprimovaným souborům. [23]

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <Book><Title><xmlzip id="1"/></Title>
3    <Info author="B. A. Writer">
4      <xmlzip id="2"/>
5    </Info>
6    <Chapter><xmlzip id="3"/>Chapter>
7    <Chapter><xmlzip id="4"/>Chapter>
8  </Book>
```

Obrázek 12 – XMLZip kódování XML dokumentu při nastavení hloubky rozdělení 2. [23]

V případě potřeby komprimované části XML dokumentu k vyhodnocení dotazu, umožňuje XMLZip dekomprimaci dané části, pro provedení dotazů tedy není nutná úplná dekomprese. Nicméně XMLZip ve většině případů dosahuje horší míry komprese, než samotný Gzip, navíc pro správnou funkci vyžaduje odstranění komentářů z dokumentu a na každém řádku by měla být pouze jedna značka. Dále, nutnost převádění XML dokumentu na DOM stromovou strukturu a naopak jej činí pomalým. [22], [23]

XPress

XPress je další komprimační technikou, která umožňuje přímé dotazování nad komprimovanými daty. Xpress je stejně jako XGrind homomorfní a dvouprůchodový, kdy při prvním průchodu probíhá statistická analýza dat a ve druhém samotná komprimace. Nicméně XPress přináší několik vylepšení oproti metodě XGrind v samotné komprimaci i dotazování.

První optimalizací je použití nové metody nazvané reverzní aritmetické kódování pro kódování struktury. Na rozdíl od kompresorů jako XMill a XGrind, které kódují jednotlivé značky slovníkovými identifikátory, XPress kóduje celou cestu ke každému elementu jako interval v rozmezí <0; 1). Přidělená rozsah intervalů je dán frekvencí výskytů jednotlivých elementů v dokumentu. Důležitou vlastností těchto intervalů je, že pokud je cesta (například *P*) sufixem jiné cesty (*Q*), interval reprezentující *P* bude obsahovat interval reprezentující *Q*. Takto jsou při zpracování dotazu efektivně vybrány pouze ty elementy, jejichž interval je obsažen v intervalu cesty dané výrazem dotazu.

Dalším vylepšením oproti XGrind je automatická aplikace různých metod kódování na základě datového typu bez nutnosti intervence uživatelem. Na základě rozpoznaného

datového typu elementu na vstupu XPress používá jeden z šesti sémantických kodérů, přičemž tři jsou pro celočíselné typy, jeden pro čísla s pohyblivou řádovou čárkou, jeden pro slovníkové kódování výčtových textových dat a jeden pro Huffmanovo kódování obecných textových řetězců. Použitím specializovaných kodérů lze docílit lepší komprese, přičemž výhodou například oproti XMill je, že výběr kodéru je automatizovaný. [40]

4 Praktická část

Praktická část této diplomové práce se věnuje analýze XML kompresorů z výběru výše popisovaných. Nejprve jsou stanovena kritéria pro výběr kompresorů a představeny konkrétní testované implementace. Následně jsou specifikovány metodika a metriky měření, prostředí testování a korpus testovacích dat. V druhé části kapitoly jsou pak představeny výsledky samotného měření.

4.1 Přípravná část měření

4.1.1 Testované kompresory

V předešlé kapitole bylo představeno celkem třináct známých technik pro komprimaci XML dat, přičemž byly rozděleny primárně na základě schopnosti rozlišení XML struktury. XML vnímající kompresní techniky byly dále rozděleny dle jejich hlavní funkční diference - na archivační a podporující dotazování nad komprimovanými daty. Jedná se o následující techniky:

- **Obecné textové** – Gzip, bzip2, PPM.
- **XML vnímající bez podpory dotazování (archivační)** – XMill, XMLPPM, DTDPPM, SCMPPM, Millau, XML-WRT, EXI.
- **XML vnímající s podporou dotazování** – XGrind, XMLZip, XPress.

Předpokladem pro zařazení kompresní techniky do testu je přístupnost veřejně dostupné implementace či zdrojových kódů a kompatibilita s alespoň jedním operačním systémem použité měřící stanice (viz tabulka 2 – primárním testovacím prostředím je Linux Mint 13, přičemž Microsoft Windows Home 10 je použit pouze v případě, že se daný kompresor nezdařilo zprovoznit v Linuxovém prostředí). Těmto podmínkám odpovídají následující techniky – Gzip, bzip2, PPM, XMill, XMLPPM, XML-WRT a EXI.

Pro obecnou textovou kompresi byly využity standardní balíčky Gzip, bzip2 a ppmd obsažené v repozitáři Linuxové distribuce Debian. Pro testování komprese XMill byly použity zdrojové kódy ve verzi 0.7 původních tvůrců Dana Suciu a Harmuta Liefke používající kompresi Gzip a novější rozšířené verze 0.9 od autora Bena Colvera pro otestování XMill s využitím bzip2 a PPM [47]. XMill byl zároveň z důvodu problémů blíže popsaných v kapitole 4.2 testován v prostředí Windows. Pro XMLPPM byla použita verze

0.98.2 [49]. Kompresor XML-WRT pak byl testován na základě zdrojových kódů ve verzi 3.4 od jeho tvůrce Przemyslawa Skibinskiho [49]. Finálním testovanou metodou je Efficient XML Interchange (EXI), pro jehož testování byla zvolena implementace EXIficient v programovacím jazyku Java [50].

Webové adresy, na kterých byly dříve přístupné zdroje pro kompresory XMLZip a SCMPPM jsou již nedostupné, od ostatních kompresorů pak nebyly nalezeny ani dřívější odkazy na jejich lokaci, tudíž lze předpokládat, že nedošlo k jejich zveřejnění. Například kompresor XPress je komerční produkt, tudíž jeho otestování není možné. V případě kompresní metody XGrind došlo po doplnění potřebných knihoven k úspěšnému sestavení z dostupných zdrojů, nicméně aplikace při pokusu o kompresi jakéhokoliv souboru, včetně testovacího, distribuovaného v samotném XGrind balíčku, skončila chybou, její testování tedy nebylo možné.

4.1.2 Metriky a prostředí měření

Sledované metriky

V rámci kapitoly 3.1.2 byly představeny možnosti, kterými lze obecně hodnotit různé aspekty celkové výkonnosti kompresních technik. Bezesporu nejdůležitější z těchto metrik, která bude primární sledovanou v tomto měření, je samotná míra komprese ve formě kompresního poměru. Maximalizace kompresního poměru vede k úspoře kapacity úložiště, snížení vytížení přenosových kanálů a také s tím spojeným finančním úsporám. V tomto měření bude kompresní poměr použit ve dříve představeném formátu úspory místa, který vyjadřuje komprimací docílenou procentuální redukci velikosti původních dat, tedy vyšší hodnota udává lepší výsledek:

$$KP = \left(1 - \frac{velikost\ komprimovaných\ dat}{velikost\ nekomprimovaných\ dat}\right) \times 100[\%]$$

Druhou sledovanou metrikou měření bude čas komprese, respektive dekomprese jednotlivých kompresních technik. Tento údaj je z praktického hlediska zvláště důležitý, jelikož se přímo podílí na celkovém času zpracování dat. Tento údaj je závislý na použitém zařízení, nicméně poměrově lze jednotlivé techniky při měření za stejných podmínek na jedné stanici srovnat.

Paměťová náročnost jednotlivých kompresních technik bude v tomto měření zanedbána, jelikož se stává, se zvyšující se kapacitou paměti i mobilních zařízení, méně závažným problémem.

Měřící stanice

Veškerá měření budou provedena na stanici s následující specifikací:

MODEL	Dell Inspiron N5110
PROCESOR	Intel Core i7-2670QM @ 2,2 GHz
OPERAČNÍ PAMĚŤ	8082 MB
PEVNÝ DISK	Seagate ATA ST500LM012 HN-M5 Oddíl 85,9 GiB, 76,5 GiB volných
OPERAČNÍ SYSTÉM	Linux Mint 13 Cinnamon 64-bit Microsoft Windows 10 Home 64-bit

Tabulka 2 - HW specifikace měřící stanice.

4.1.3 Testovací korpus

Pro srovnávací testování XML dosud neexistuje univerzálně standardizovaný korpus, je tedy třeba provést vhodný výběr z veřejně dostupných XML dokumentů. Jednotlivé vybrané kompresní techniky používají různé postupy k docílení komprese, lze tedy očekávat, že povedou k odlišným výsledkům při zpracování dokumentů různých struktur a velikostí. Například lze předpokládat, že některé kompresní techniky vysoce specializované na oddělení a kompresi struktury XML mohou u méně strukturovaných dokumentů dosahovat i horších výsledků, než obecné textové kompresory.

Pro otestování efektivity jednotlivých kompresních technik na komprimaci XML dokumentů s různými vlastnostmi tedy bude vybrána kolekce obsahující vysoce strukturované dokumenty s vysokou mírou zanoření, jejichž obsah je tvořen převážně značkováním, textové dokumenty s jednoduchou strukturou, jejichž obsah je tvořen

převážně znakovými daty a standardní XML dokumenty tvořeny značkováním proloženým krátkými datovými obsahy. Zvolené dokumenty jsou následovné:

XMark Benchmark

XMark je projekt umožňující výkonnostní porovnání XML databází, zejména z hlediska efektivity zpracování různých dotazů. V rámci tohoto projektu je volně dostupný zdrojový kód pro nástroj xmlgen umožňující libovolně škálovatelné generování XML dokumentů ve velikostech od desítek kilobytů po mnoho gigabytů. Vygenerované soubory simulují aukční databázi s relativně hlubokým zanořením elementů. Dokumenty obsahují plný rozsah XML prvků, skládají se jak z relativně strukturovaných pravidelných částí, tak z elementů obsahujících vysoce odlišné vnitřní struktury potomků a délky textů. Pro simulaci přirozeného jazyka v delších textových částech je generovaný text skládán ze 17 000 nejčastěji se vyskytujících slov Shakespearových her. [42]

Pomocí nástroje xmlgen byly vygenerovány následující XML dokumenty:

- XMark-small – 11,7 MB
- XMark-medium – 116,5 MB
- XMark-large – 585,5 MB

SwissProt

SwissProt je databáze obsahující sekvence aminokyselin v proteinech. XML dokument obsahuje mnoho anotací a má minimální úroveň redundance. Neobsahuje DTD. Velikost – 114.8 MB. [43]

Wikipedia abstrakty

Otevřená online encyklopedie Wikipedie pravidelně zálohuje obsah své databáze, který je volně poskytován. V této práci je použit XML dokument cswiki_abstract obsahující úvody všech článků české Wikipedie. Jednotlivé abstrakty obsahují regulární strukturu o čtyřech úrovních zanoření a mnoho URL odkazů. Velikost – 420,3 MB. [44]

Treebank

Jedná se o kolekci zašifrovaných textů deníku Wall Street Journal, které jsou proloženy značkováním dle slovník druhů. Struktura dokumentu je velmi neregulární, rekurzivní a hluboká – maximální hloubka zanoření dosahuje hodnoty třicet šest. Velikost - 86,1 MB. [43]

TPC-H Benchmark

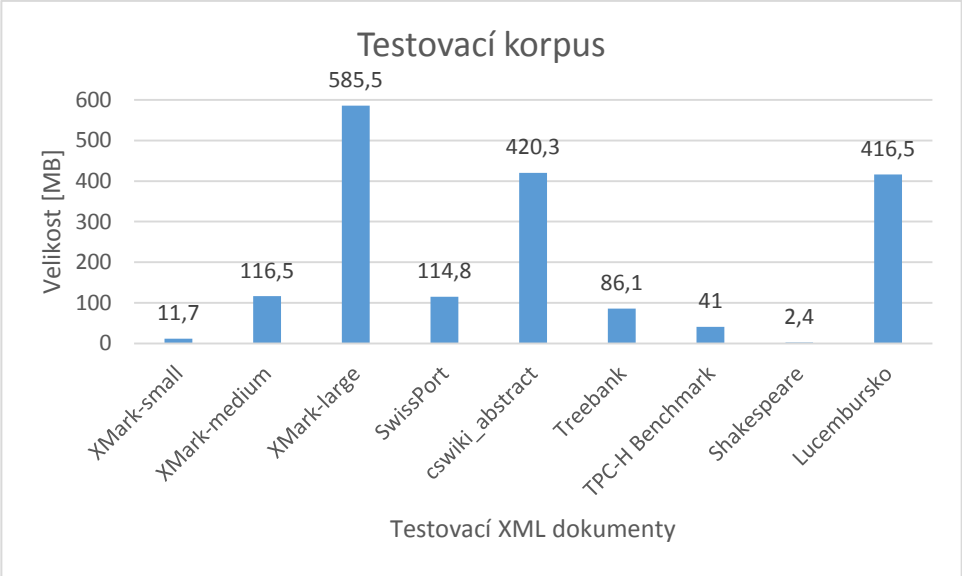
Testovací databáze pro hodnocení výkonu zpracování dat v relační databázi. Databáze je naplněna širokou škálou dat z obchodního prostředí. Osm tabulek databáze konvertovaných do XML bylo spojeno do jednoho souboru, struktura dosahuje hloubky třetí úrovně. Velikost – 41 MB. [43]

Shakespearovy hry

Kolekce jedenácti her Williama Shakespeara značkovaných v XML spojena do jednoho souboru. Jedná se o XML dokument skládající se převážně z dlouhých textových sekcí a s minimalistickou strukturou. Velikost – 2,4 MB. [45]

Lucembursko

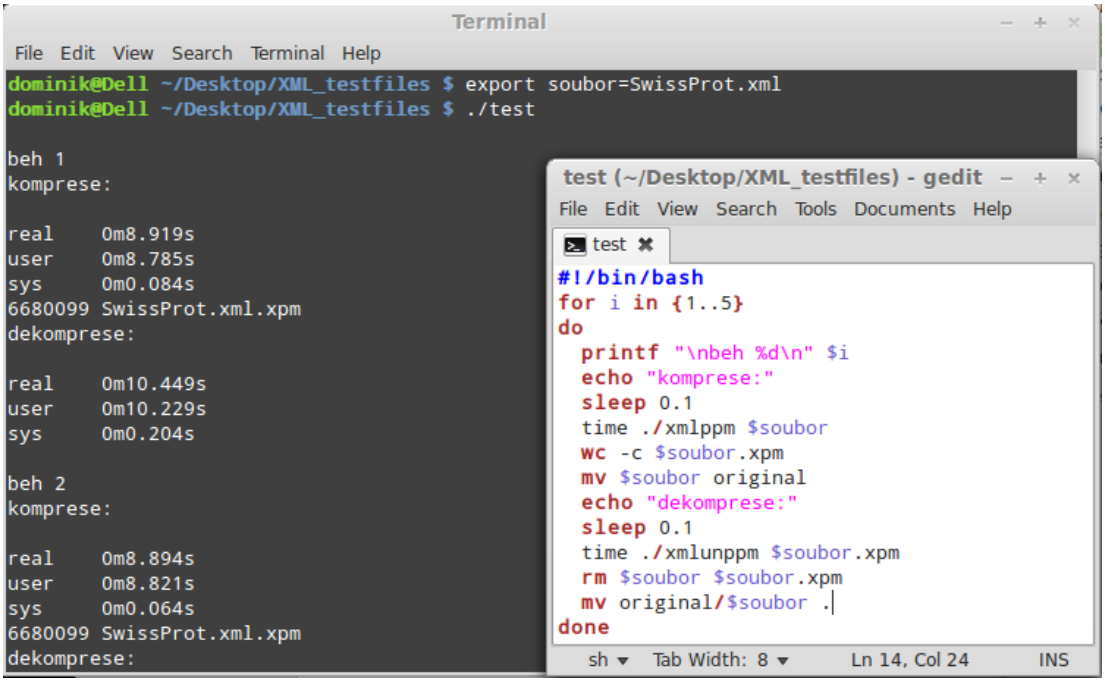
OpenStreetMap model Lucemburska využívající XML. Hlavními elementy jsou uzly popisující bod v prostoru, cesty tvořené uspořádanými seznamy uzlů, relace vysvětlující závislosti elementů a značky obsahující doplňující informace ke konkrétním elementům. Maximální zanoření je do hloubky druhé úrovně a všechna data (souřadnice, identifikátory, text atd.) jsou obsažena v atributech. Velikost – 416,5 MB. [46]



Graf 1 - Přehled XML dokumentů testovací kolekce

4.2 Měření XML komprese

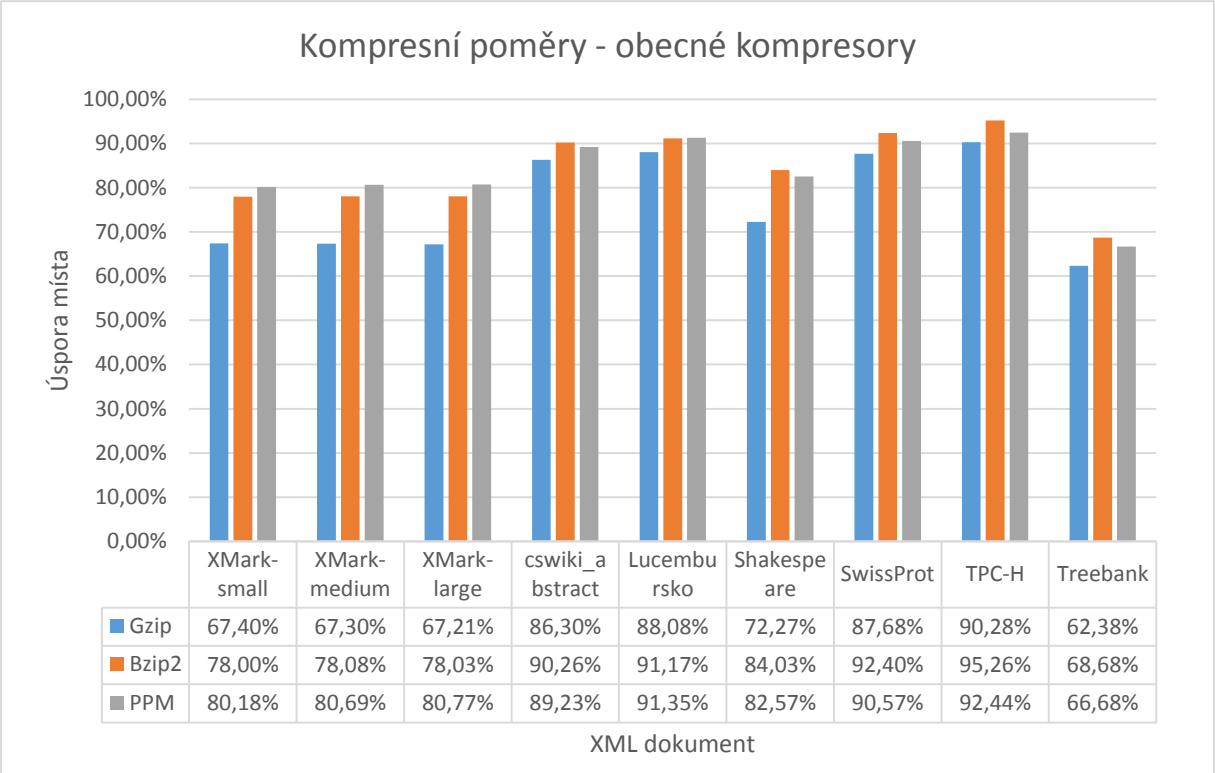
Kompresory byly po kompilaci testovány ve formě konzolových aplikací, přičemž měření času běhu komprese a dekomprese bylo v prostředí Linux měřeno standardním příkazem operačních systémů Unixového typu, zabudovaným v příkazovém procesoru Bash, *time*, pro výpis doby běhu programu. V případě testování v prostředí Microsoft Windows byl pro měření času použit Windows PowerShell příkaz *Measure-Command* s obdobnou funkcí. Pro minimalizaci vlivu různých efektů způsobujících mírné časové fluktuace v době běhu kompresoru byla komprese a dekomprese každého XML dokumentu s daným kompresorem zopakována pětkrát, načež byl zapsán průměr. Tabulky s naměřenými hodnotami testovaných kompresorů (velikost komprimovaných souborů, kompresní poměr, průměrné časy) jsou obsaženy v příloze číslo 1.



Obrázek 13 - Ukázka testování XML komprese, vlastní tvorba.

4.2.1 Obecné textové kompresory

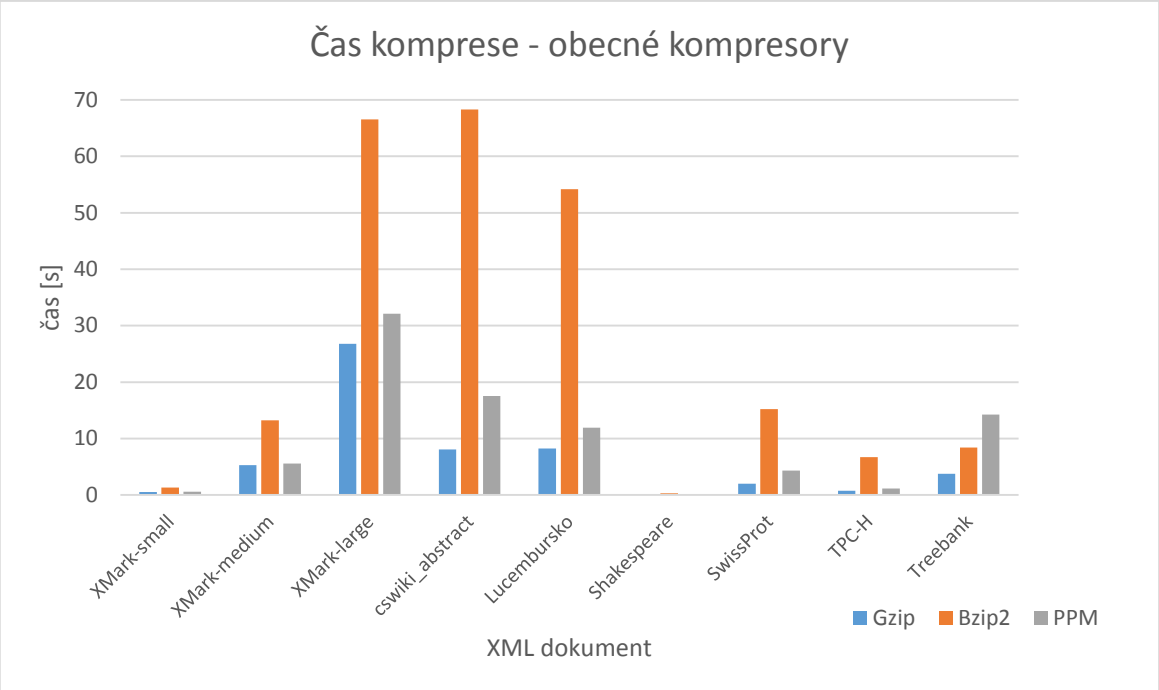
Obecné textové kompresory Gzip, bzip2 a PPM dosahovaly na testovaných XML dokumentech procentuální úspory místa v rozpětí 62,38% až 95,26% (viz graf 2), přičemž průměrné hodnoty napříč kolekcí byly v pořadí Gzip – 76,55%, PPM – 83,83% a bzip2 – 83,99%. Největším rozdílem v rámci jednoho komprimovaného dokumentu bylo 13,56% na souboru XMark-large mezi metodami Gzip (67,21%) a PPM (80,77%), což poukazuje na důležitost závislosti výskytu znaku na jeho kontextu.



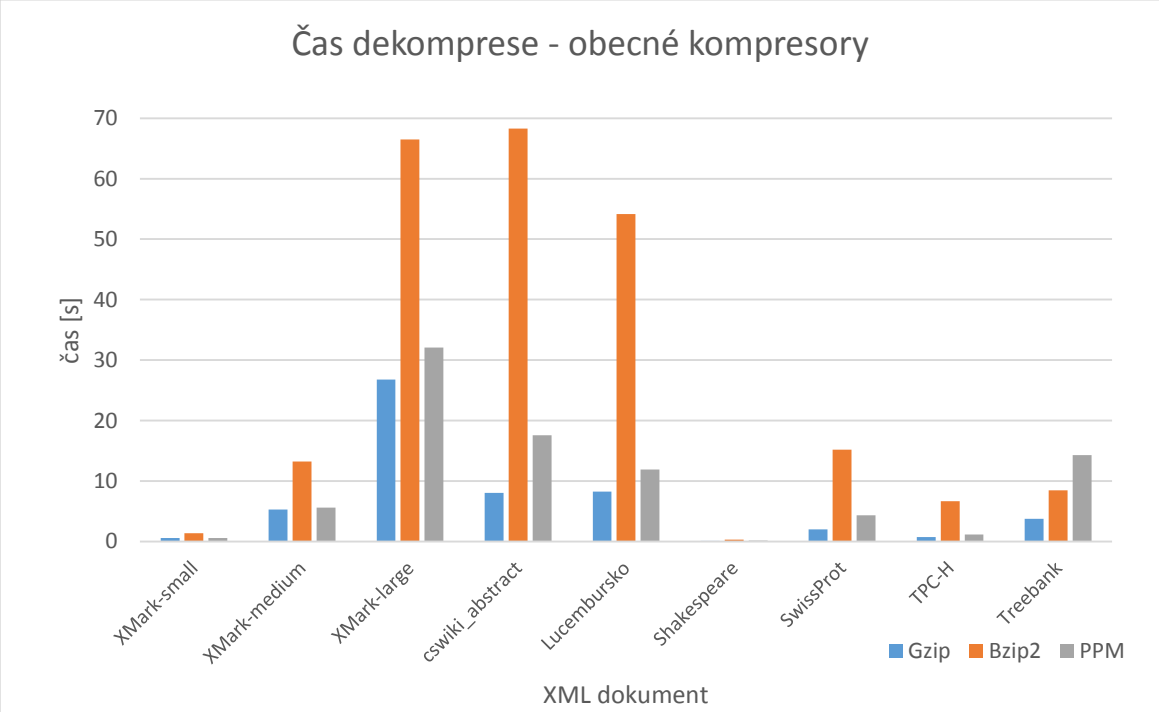
	XMark-small	XMark-medium	XMark-large	cswiki_abstract	Lucembursko	Shakespeare	SwissProt	TPC-H	Treebank
Gzip	67,40%	67,30%	67,21%	86,30%	88,08%	72,27%	87,68%	90,28%	62,38%
Bzip2	78,00%	78,08%	78,03%	90,26%	91,17%	84,03%	92,40%	95,26%	68,68%
PPM	80,18%	80,69%	80,77%	89,23%	91,35%	82,57%	90,57%	92,44%	66,68%

Graf 2 - Kompresní poměry obecných textových kompresorů.

Z hlediska míry komprese nejméně efektivní Gzip byl zároveň dle očekávání nejrychlejší při kompresi i dekompresi. Bzip2 je zdaleka nejpomalejší obecnou metodou, v některých případech mnohonásobně pomalejší, než PPM, což se negativně projevuje především u největších dokumentů. Jelikož PPM dosahuje téměř totožné míry komprese jako bzip2, lze jej považovat za z celkového hlediska nejefektivnější obecnou metodu komprese textu.



Graf 3 - průměrné časy komprese obecných textových kompresorů.



Graf 4 - Průměrné časy dekomprese obecných textových kompresorů.

4.2.2 XMill

Překážky testování

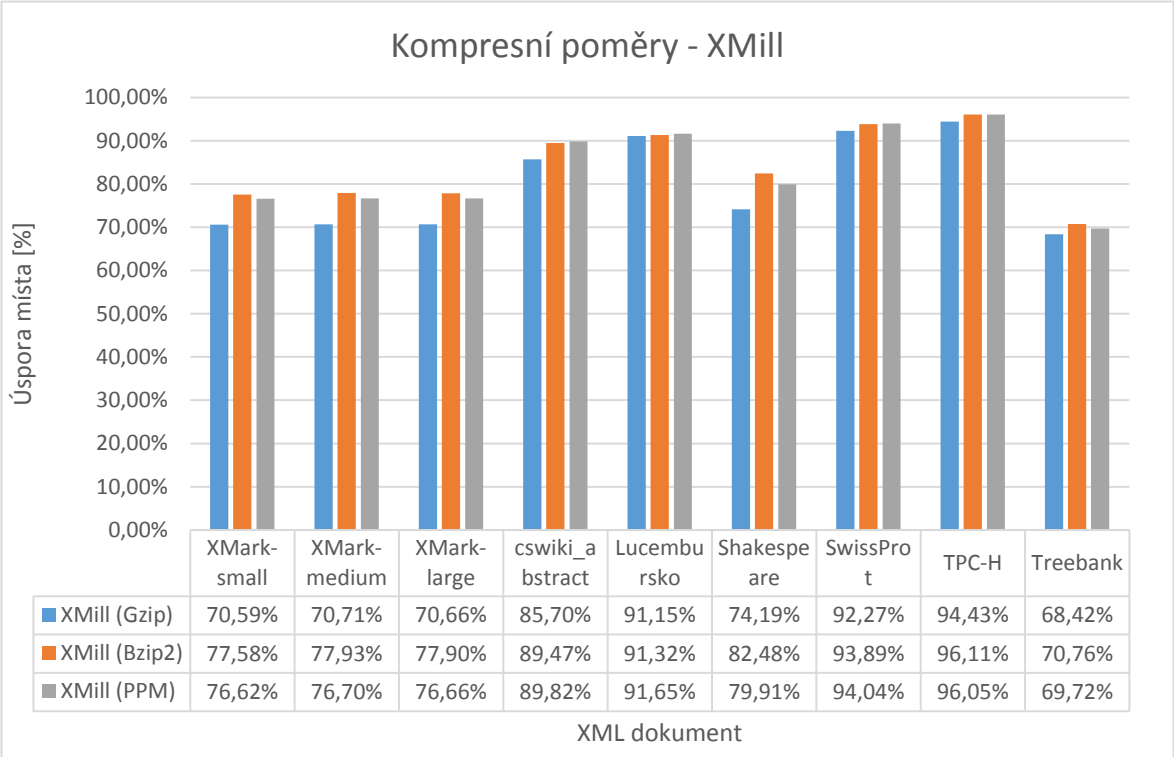
Jak již bylo zmíněno, kompresor XMill byl testován na základě zdrojových kódů v původní verzi 0.7 používající Gzip pro finální textovou kompresi a verzi 0.9 pro otestování kompresoru s využitím metod bzip2 a PPM při finální kompresní fázi. Důvodem nevyužití novější verze 0.9 také pro variantu s Gzip kompresí je, že verze 0.9 byla při testech s Gzip kompresí ve stejné konfiguraci pomalejší, než původní verze 0.7, přičemž stabilita a efektivita kompresoru s využitím Gzip zůstala totožná. Důvodem zřejmě bude celkové přepsání XMill ve verzi 0.8 (nově je možné například původně čistě konzolový nástroj kompilovat jako knihovnu, přibylo mnoho možností nastavení, komprese atd.). XMill byl také jako jediný testován v prostředí operačního systému Microsoft Windows 10, jelikož v Linuxovém prostředí se nezdařila kompilace ani jedné z verzí 0.7, 0.8 a 0.9. Hlavní problém po vyřešení nekompatibility novějších verzí překladačů představovalo primárně využívání starých již nepodporovaných knihoven, v nichž bylo po jejich nalezení a instalaci odkazováno na další soubory, které již nebylo možné dohledat. V systému Windows byl XMill úspěšně sestaven v Microsoft Visual Studiu.

Výsledky testování XMill

Kompresor XMill byl pro měření ponechán ve výchozím nastavení a výjimkou volby možností koncové komprese a vynucení zachování nepodstatných bílých znaků, jelikož bez tohoto parametru docházelo k nežádané změně původního souboru a snížení čitelnosti po opětovné dekomprimaci (v případě permanentního odstranění nepodstatných bílých znaků bylo možné v průměru zvýšit úsporu místa oproti uvedeným hodnotám až o 1%).

XMill dosáhnul v porovnání s odpovídajícími samotnými textovými kompresory zlepšení průměrného uspořeného místa o 3,24% v případě Gzip, 0,17% v případě bzip2 a 0,37% v případě PPM (viz graf 5), přičemž pořadí efektivity komprese jednotlivých kompresních metod zůstalo stejné, jako při použití samostatné textové komprese. Výsledek zvlášť u bzip2 a PPM je značně pod očekáváním a při velmi vysokém času zpracování (viz graf 6 a 7) nenese XMill při nastavení bzip2 či PPM komprese objektivní výhodu oproti běžné textové kompresi. Při použití bzip2 komprese dokonce XMill dosáhnul horšího

výsledku, než samotný bzip2 při kompresi pěti z devíti testovaných dokumentů, u PPM tomu tak bylo u čtyř dokumentů.



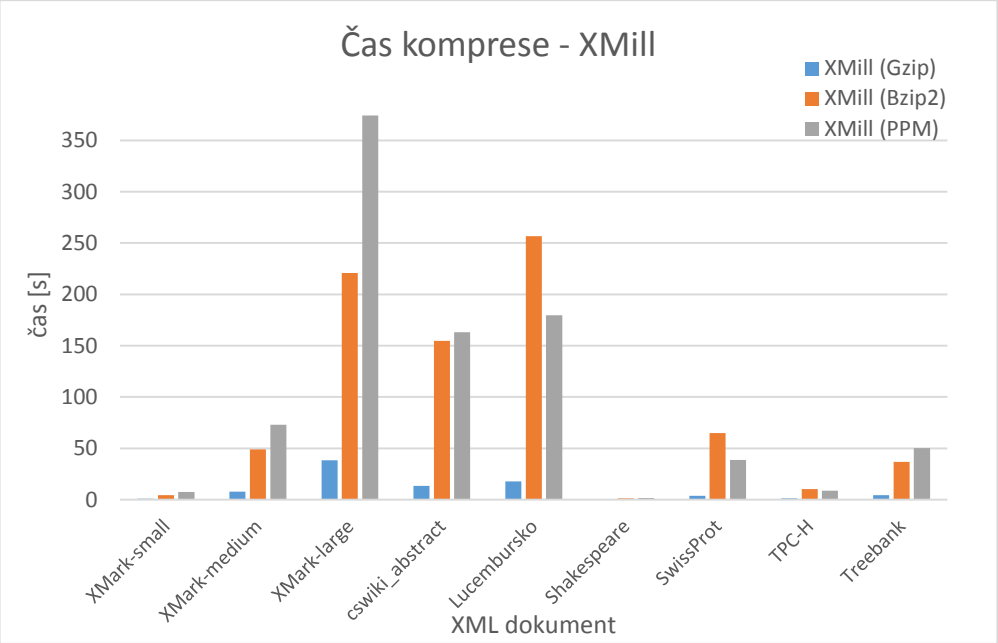
	XMark-small	XMark-medium	XMark-large	cswiki_abstract	Lucembursko	Shakespeare	SwissProt	TPC-H	Treebank
XMill (Gzip)	70,59%	70,71%	70,66%	85,70%	91,15%	74,19%	92,27%	94,43%	68,42%
XMill (Bzip2)	77,58%	77,93%	77,90%	89,47%	91,32%	82,48%	93,89%	96,11%	70,76%
XMill (PPM)	76,62%	76,70%	76,66%	89,82%	91,65%	79,91%	94,04%	96,05%	69,72%

Graf 5 - Kompresní poměry XML kompresoru XMill.

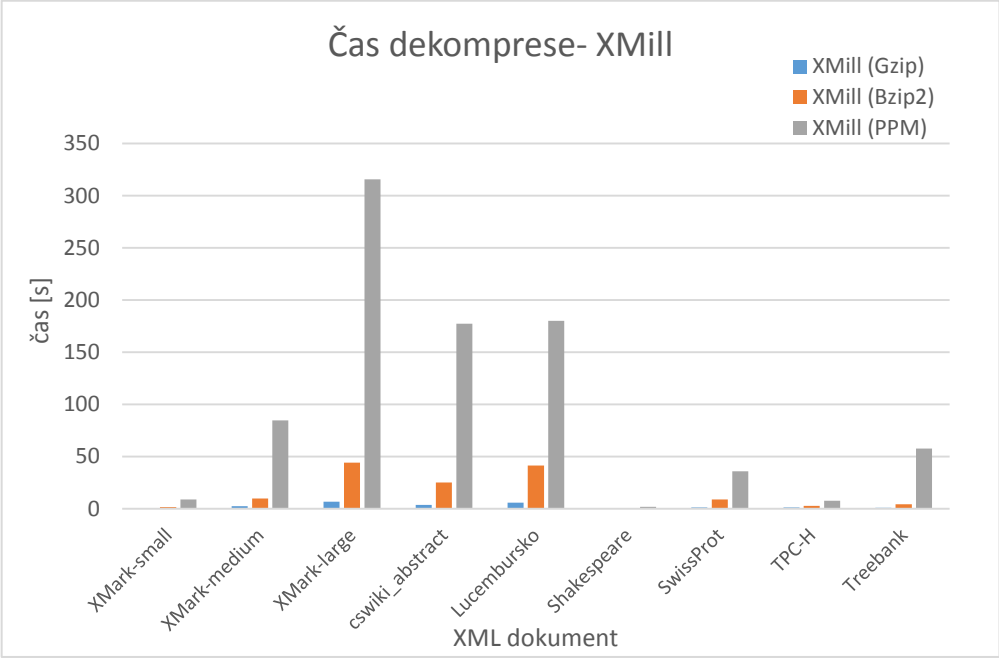
V případě Gzip komprese, s kterou byl XML původně stvořen a je pro ni tudíž nejlépe optimalizován, byly výsledky komprese konzistentně lepší oproti použití samotného Gzip kompresoru (mírně horší komprese pouze u Wikipedia abstraktů), zároveň byl XMill s Gzip při kompresi v průměru pouze o 58,75% (rozdíl 32,59 sekund na celé kolekci) pomalejší než samotný Gzip (například v případě PPM se u rozdílu samotného PPM kompresoru a XMill s PPM jednalo o časový nárůst 923,66% představující celkový nárůst času zpracování testovací kolekce o 809,03 s).

Nejlepších výsledků XMill dosahoval při použití Gzip a PPM komprese na pravidelně strukturovaných dokumentech s konzistentním a předvídatelným obsahem SwissProt a TPC-H, kde byla úspora místa přibližně o 4 % větší, než u standardní textové komprese. Lze předpokládat, že při nastavení vhodných sémantických kompresorů na jednotlivé kontejnery by bylo docíleno dalšího mírného zvýšení kompresního poměru, XMill by byl tedy vhodný pro archivaci velkých XML dokumentů podobného typu. Naopak převážně horší výsledky měly všechny konfigurace kompresoru XMill oproti samotné textové kompresi v případě

většinově textových dokumentů s menším podílem značkování Shakespeare a cswiki_abstract. Data v jednotlivých kontejnerech těchto dokumentů jsou sémanticky velmi podobná, přičemž udržování informací o jejich rozdělení pouze navyšuje velikost souboru.



Graf 6 - Průměrný čas komprese XML kompresoru XMill.



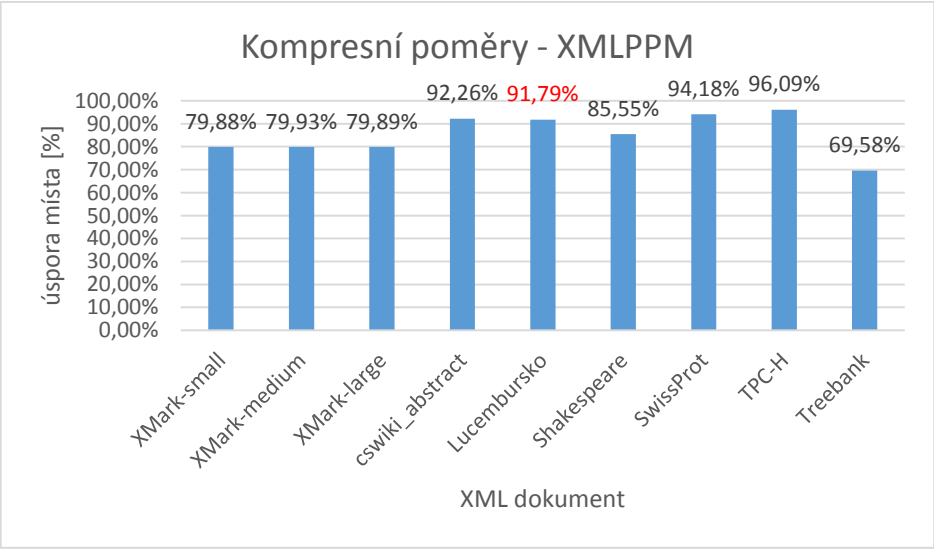
Graf 7 - Průměrný čas dekomprese XML kompresoru XMill.

4.2.3 XMLPPM

Testování kompresoru XMLPPM proběhlo ve verzi 0.98.2, přestože nejnovější verze je číslována 0.98.3. Důvodem použití verze 0.98.2 je, že novější verze po kompilaci skončila při každém běhu chybou porušení ochrany paměti (Segmentation Fault), kterou se nepodařilo vyřešit – nicméně verze 0.98.3 je dle dokumentace téměř totožná. XMLPPM dosáhlo průměrné úspory místa 85,46%, což je rozdíl pouze 1,63% oproti samotné PPM kompresi, nicméně výsledek je ovlivněn kolekcí XMark s částmi nepravidelného obsahu a velkým počtem různých značek, u které XMLPPM dosáhlo snahou o třídění dat horšího výsledku, než samotná PPM komprese. U zbývajících šesti dokumentů je průměrná úspora necelá tři procenta, přičemž největší úspory oproti samotné kompresi bylo opět docíleno u dokumentů SwissProt a TPC-H.

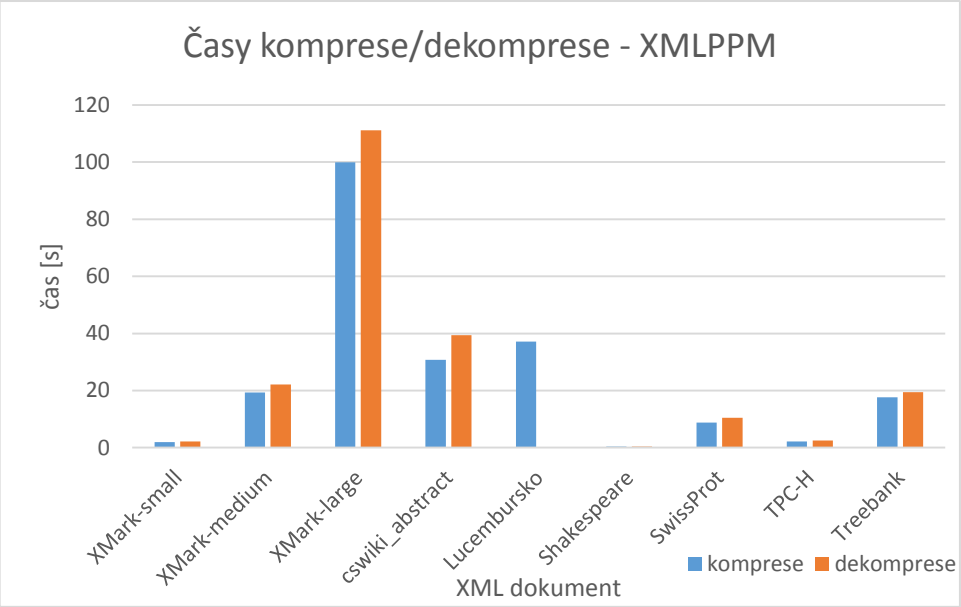
XMLPPM při kompresi odstraňuje nepodstatné bílé znaky, načež je při dekompresi vkládá samo automaticky, zároveň expanduje prázdné elementy označeny na <a>, při dekompresi tedy může docházet k nechtěné mírné expanzi oproti originální velikosti dokumentu.

U souboru Lucembursko (jehož syntaktická správnost byla nezávisle ověřena), došlo při každém pokusu o dekompresi k neočekávanému ukončení běhu programu s chybou – jedná se tedy o velmi závažnou chybu, kdy se stává obnovení původního dokumentu a jeho další použití nemožným. XMLPPM je jediným kompresorem z testovaných, u kterého došlo k nenávratné ztrátě dat.



Graf 8 - Kompresní poměry XML kompresoru XMLPPM.

Z hlediska časové náročnosti je XMLPPM oproti PPM 2,48x pomalejší při kompresi a 2,18x při dekompresi, což jej činí mnohem použitelnější, než XMill PPM komprese.

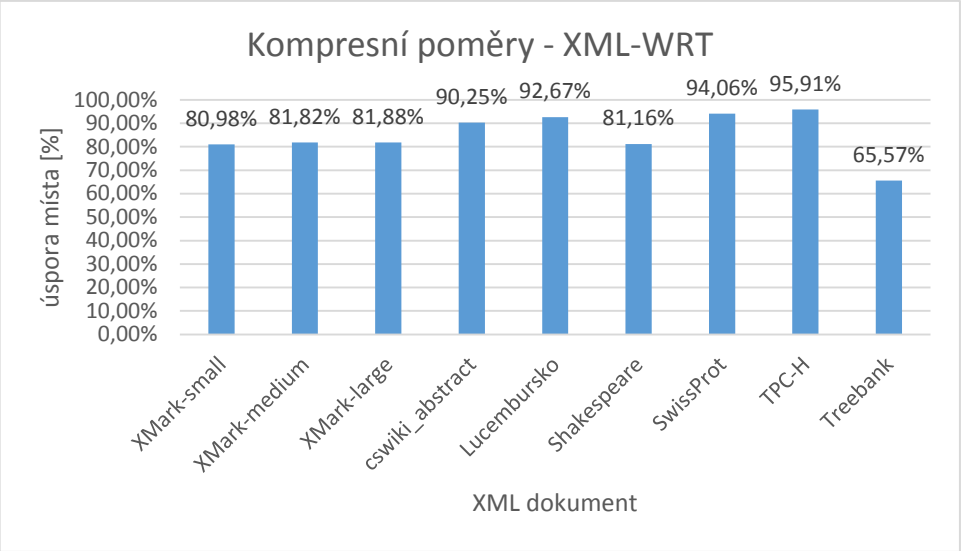


Graf 9 - Průměrné časy komprese/dekomprese XML kompresoru XMLPPM.

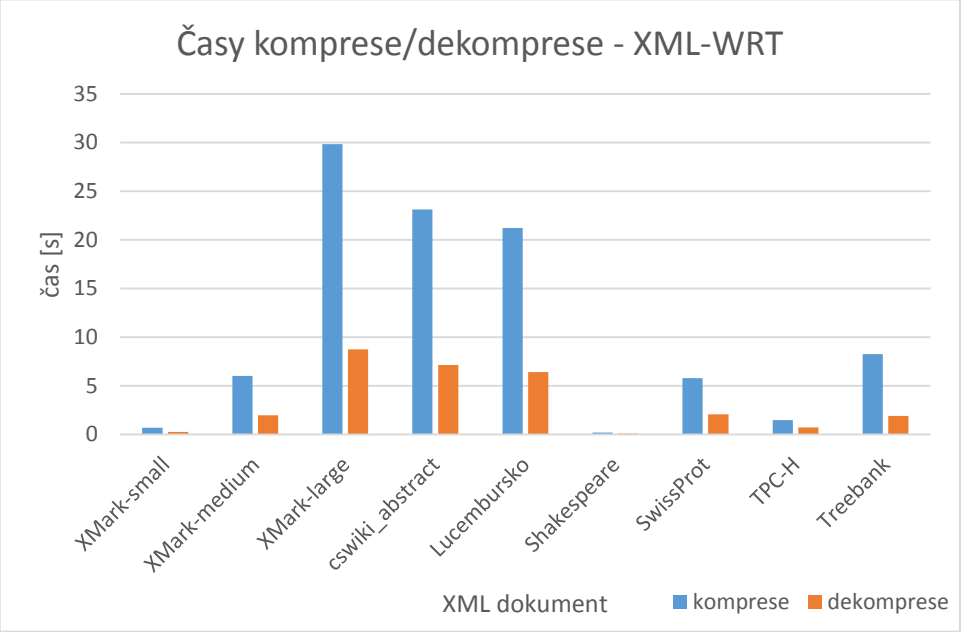
4.2.4 XML-WRT

Kompresor XWRT při nastavení zlib (Gzip) pro finální kompresi dosáhnul průměrné úspory místa v kolekci 84,92%, svou efektivitou komprese se tudíž blíží XMLPPM, přičemž je více než dvakrát rychlejší při kompresi a více než sedmkrát rychlejší při dekompresi, zároveň při jeho běhu nedošlo k žádným chybám. XWRT je pouze o 74,09% pomalejší při kompresi kolekce a o něco více než dvakrát pomalejší při její dekompresi, než Gzip, což jej činí druhým nejrychlejším testovaným XML specifickým kompresorem, přičemž průměrná úspora místa oproti Gzip je o 8,37% vyšší a oproti XMill s Gzip o 5,13% vyšší.

Další nabízené možnosti volby finálního komprese – PPM, LZMA/BWT a PAQ nebyly do testování zahrnuty, jelikož ve všech případech dosahovaly až čtyřikrát horších kompresních poměrů oproti výchozímu nastavení zlib, zřejmě z důvodu neoptimalizované implementace.



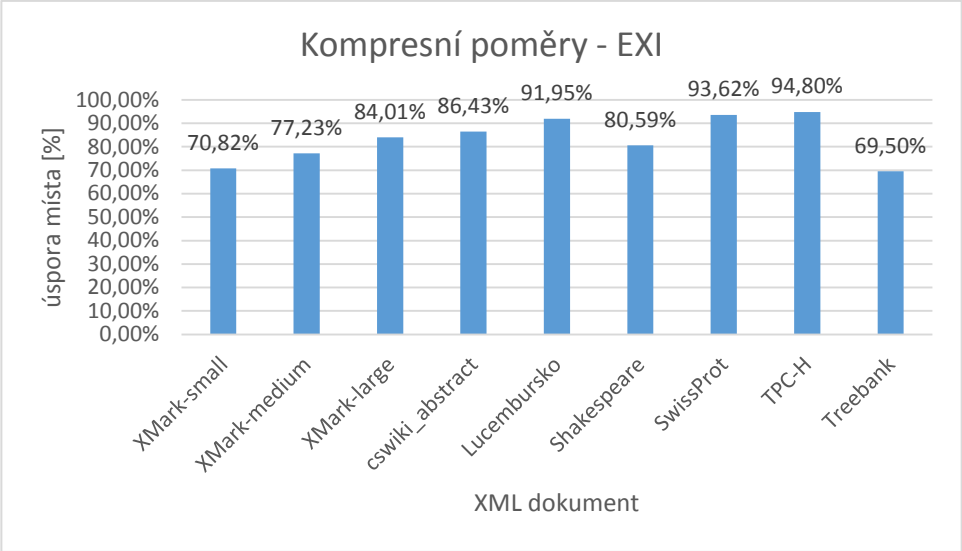
Graf 10 - Kompresní poměry XML kompresoru XML-WRT.



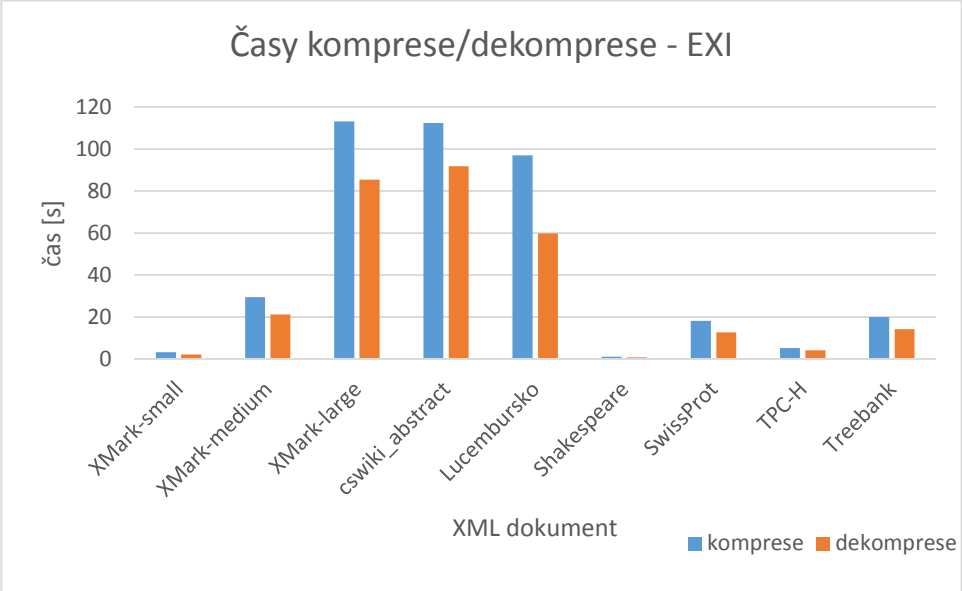
Graf 11 - Časy komprese a dekomprese XML kompresoru XML-WRT.

4.2.5 EXI

Komprese EXI dosáhla průměrné úspory místa 83,22%, což je třetí nejnižší výsledek po Gzip a XMill s Gzip, přičemž komprese celé kolekce byla časově více než sedmkrát náročnější a dekomprese třiadvacetkrát náročnější než u Gzip. EXI dosahuje horších výsledků, než obecná textová komprese pomocí bzip2 a PPM jak po stránce úspory místa, tak časové, při relativně vysoké implementační složitosti a nemožnosti dotazování nad komprimovanými daty je tedy jedinou výhodou jeho využívání větší dostupnost a rozšíření, jelikož se jedná o W3C doporučení.



Graf 12 - Kompresní poměry XML kompresoru EXI.

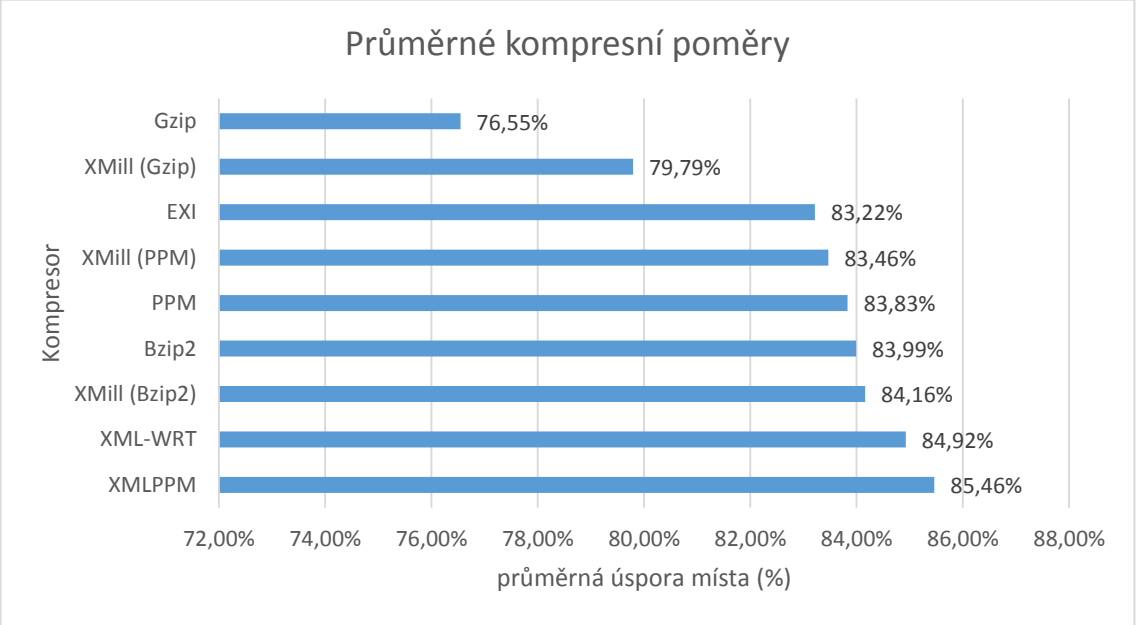


Graf 13 - Časy komprese/dekomprese XML kompresoru EXI.

5 Zhodnocení výsledků

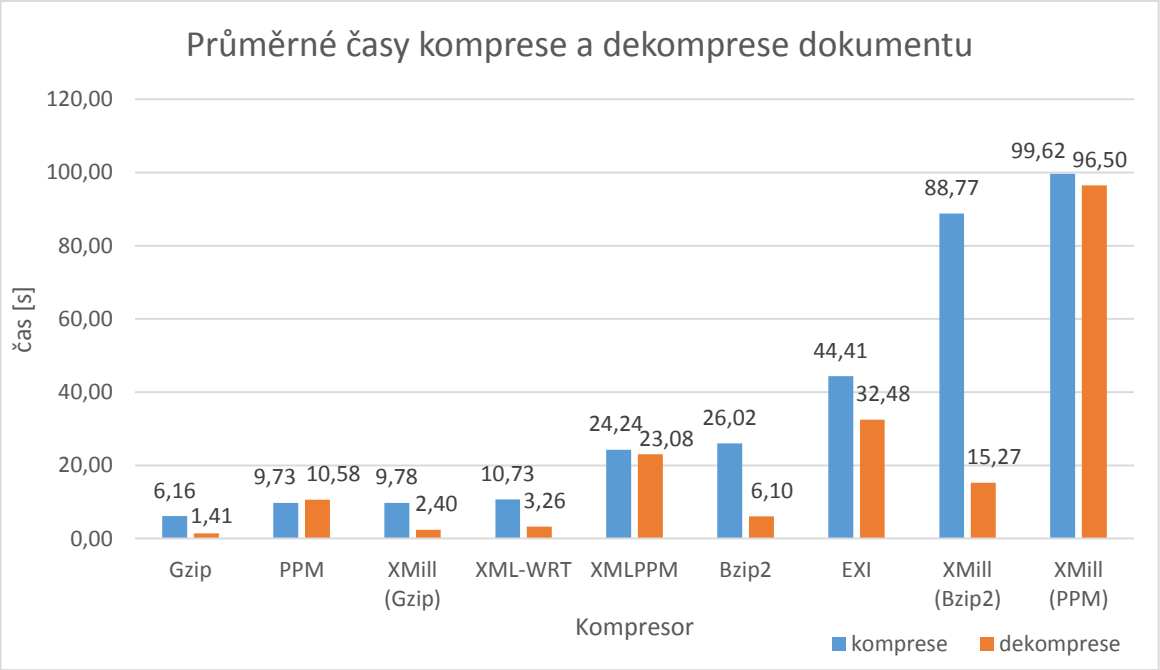
5.1 Porovnání kompresorů

V kapitole výše bylo provedeno měření sedmi různých kompresorů – z toho tří obecných textových a čtyř zaměřených na XML, přičemž kompresor XMill byl testován ve třech různých konfiguracích. Průměrné kompresní poměry všech kompresorů za celou testovací kolekci se pohybují v rozmezí pouhých 8,91%, přičemž nejnižší průměrné úspory místa docílil kompresor Gzip (76,55%) a nejvyšší XML specifický kompresor XMLPPM (85,46%) – kompresory jsou seřazeny dle dosaženého průměrného kompresního poměru v grafu 14.



Graf 14 - Průměrné kompresní poměry.

Porovnání kompresorů z časového hlediska za pomoci průměrného času komprese a dekomprese (viz graf 15) potvrzuje, že parsování a úpravy XML dokumentu, které XML vnímající kompresory před samotnou kompresí provádí, znatelně zvyšují časovou náročnost – nejrychlejšími kompresory jsou Gzip a PPM, z daleka nejpomalejším je pak XMill v konfiguraci používající Bzip2 či PPM kompresi. Překvapivého výsledku docílil kompresor XML-WRT s průměrnou dobou komprese 10,73 sekund a dekomprese 3,26 sekund, což jej umisťuje z hlediska časové náročnosti na čtvrté místo s minimálním rozdílem v době běhu oproti kompresně mnohem méně efektivnímu XMill v konfiguraci Gzip.



Graf 15 - průměrné časy komprese a dekomprese jednoho souboru kolekce.

Každý XML specifický kompresor dosáhnul v průměru lepších kompresních výsledků, než samotná textová komprese, kterou využívá, čímž se potvrzuje efektivita předzpracování dokumentu a využití jeho znalosti pro docílení vyššího kompresního poměru. Výsledky XML specifických kompresorů však neprokázaly příliš výrazné zlepšení oproti obecným textovým kompresorům, což potvrzuje zmíněné rozpětí procentuální úspory místa mezi všemi testovanými kompresory pouze 8,91%.

V některých případech dokonce, v závislosti na komprimovaném dokumentu, dosahovaly XML specifické kompresory minimálního zlepšení či dokonce zhoršení komprese oproti samotné textové kompresi stejného typu – tento efekt se projevoval nejvíce u dokumentů obsahujících Wikipedia abstrakty a Shakespearovy hry, jelikož jak již bylo zmíněno, jejich textový obsah je sémanticky velmi podobný napříč všemi elementy, rozdělení dat a separátní komprimace v takovém případě nepřináší zlepšení komprese. Stejně tak bylo pro všechny kompresory náročné zpracování tří XMark dokumentů, které obsahují velké množství elementů s různým datovým obsahem. Nejnáročnějším dokumentem pro všechny testované obecné i specializované kompresory pak byl vysoce iregulárně strukturovaný dokument s vysokým zanořením Treebank, u něho bylo dosaženo

průměrné úspory místa napříč všemi kompilátory pouze 67,92% oproti celkovému průměru všech kompilátorů napříč celou kolekcí dokumentů 82,82%.

Nejlepších výsledků naopak dosáhla komprese dokumentu TPC-H s průměrnou redukcí velikosti o 92,66%, u kterého docházelo spolu s dokumentem SwissProt také k největšímu zlepšení u skupiny XML specifických kompresorů oproti obecným (důvodem je jejich pravidelná struktura s konzistentním sémantickým rozdělením dat v jednotlivých elementech, jak bylo zmíněno již v kapitole 4.2.2).

5.2 Výběr optimálního XML kompresoru

Volba univerzálně optimálního XML kompresoru je komplikovanou záležitostí, jelikož pro různé konkrétní aplikace mohou být důležitá odlišná kritéria – pro účely dlouhodobé archivace je například podstatný téměř výhradně kompresní poměr, zatímco při pravidelném průběžném zálohování či přenosu dat po síti je stěžejní čas komprese a dekomprese, zvláště pokud je po přenosu nutné s daty okamžitě pracovat. Výběr vhodného kompresoru mohou ovlivňovat také další, v tomto experimentu ne přímo měřené faktory, jako například komplexita implementace, která je u XML specifických kompresorů vždy vyšší než u obecných, což při přípravě a během měření vedlo k mnoha komplikacím s jejich zprovozněním. Hodnocení kompresorů lze ovšem zkráceně shrnout následovně:

- XMill je v konfiguraci PPM a Gzip2 špatně optimalizovaný – kompresní poměr je oproti samotné textové kompresi téměř bez změny, v některých případech horší, přičemž doba běhu je oproti ostatním kompresorům extrémní (viz 4.2.2). XMill v původní konfiguraci s kompresorem Gzip je velmi rychlý a dosahuje konzistentně lepších, na větších souborech znatelně vyšších kompresních poměrů oproti samotnému Gzip, nicméně je jeho efektivita značně překonána kompresorem XML-WRT a zprovoznění s moderními nástroji a systémy problematické, přičemž na aktualizaci již zřejmě nikdo pracovat nebude – původní XML specifický kompresor tedy dnes již nelze doporučit.
- EXI komprese je časově náročná, implementace EXI netriviální a formát nepřináší žádné výhody oproti bzip2 či PPM kompresi, přičemž dosahuje i

mírně horšího kompresního poměru oproti zmíněným – EXI tedy též nelze doporučit.

- XMLPPM při průměrné/spíše vyšší časové náročnosti dosahuje nejlepšího kompresního poměru, v poslední dostupné podobě je však stále označeno za beta verzi, přičemž je projekt původními autory opuštěný. XMLPPM v případě jednoho dokumentu způsobilo nenávratnou ztrátu dat chybou při dekompresi (viz 4.2.3) – jedná se o kritickou chybu, tudíž kompresor nelze doporučit z důvodu nestability.
- Kompresor XML-WRT dosahuje téměř stejných kompresních výsledků jako XMLPMM, je více než dvakrát rychlejší (v případě dekomprese mnohonásobně rychlejší a celkově druhý nejrychlejší) a zároveň je plně stabilní. Jedná se také o jediný XML specifický kompresor, při jehož zprovoznění se nevyskytly žádné problémy – tento kompresor lze tedy plně doporučit.
- Z obecných textových kompresorů je na kompresi XML možné doporučit především PPM, které při snadné dostupnosti a téměř zaručené kompatibilitě dosahuje podobných či lepších výsledků, než mnohé XML specifické kompresory, zároveň se jedná o jeden z nejrychlejších testovaných kompresorů. V případě nutnosti velmi rychlé komprese a především dekomprese s nízkými nároky na úsporu místa lze také použít standardní Gzip (který je dnes pro kompresi XML používán nejčastěji). Jelikož kompresor Bzip2 dosahuje téměř totožných kompresních poměrů jako PPM při více než dvounásobné době komprese, není důvod tento kompresor využívat (výjimkou může být potřeba rychlé dekomprese při zachování velké úspory místa – zde je ovšem lepší při kompresi XML použít XML-WRT).

6 Závěr

Cílem této diplomové práce byla analýza známých komprimačních technik XML a provedení jejich komparace. K dosažení cíle byla analyzována odborná literatura za účelem výběru relevantních komprimačních technik, načež byly definovány sledované metriky a provedeno měření, na základě jehož výsledků byla komparace realizována.

V teoretické části je práce zahájena kapitolou, která uvádí do problematiky jazyka XML a definuje s ním související pojmy. Druhá polovina teorie je pak věnována samotné kompresi XML dat a kromě obecné charakteristiky datové komprese, základních kompresních algoritmů či definici způsobů dělení XML kompresorů také obsahuje popis celkem třinácti vybraných komprimačních technik používaných pro kompresi XML dat, z nichž tři představují obecné textové kompresory a deset je XML specifických.

Praktická část práce se skládá primárně z realizace testování kompresorů a měření zvolených metrik pro jednotlivé kompresní techniky – kompresního poměru ve formě procentuální úspory místa docílené kompresí a časovou náročností komprese a dekomprese, na základě vyhodnocení, jehož výsledků byly vybrané kompresory srovnány. V přípravné fázi byly zvoleny konkrétní implementace kompresních technik, které byly testovány – z důvodu nedostupnosti veřejně přístupných zdrojových kódů a problémům se zprovozněním a funkčností mnohých kompresorů specializovaných na XML bylo možné podrobit testování pouze celkem sedm z třinácti popisovaných kompresních technik. Dále byl před samotným měřením sestaven testovací korpus celkem devíti XML dokumentů sestávajících z různých velikostí, vnitřních struktur a datového obsahu.

Výsledky provedeného měření ukázaly, že kompresní techniky specializované na XML v průměru dosahovaly lepších kompresních poměrů, než obecné textové kompresory, což prokazuje efektivitu znalosti struktury XML dokumentu a jejího zpracování či rozdělení před provedením samotné komprese. Nicméně, zlepšení kompresního poměru mezi standardně používanými obecnými textovými kompresními technikami a XML specifickými nebylo markantní, jelikož výsledky všech testovaných kompresorů byly v rozmezí pouhých 8,91% průměrné úspory místa, přičemž u dokumentů s větším poměrem textového obsahu dosahovaly XML specifické kompresory často i horších kompresních poměrů, než obecné textové kompresory jako Gzip, bzip2 a PPM. Zároveň, ze stejného důvodu parsování a

předzpracování dokumentu byly XML specifické kompresní techniky vždy pomalejší, než samotný textový kompresor, který využívají.

Kromě malého zlepšení míry komprese a znatelného nárůstu doby běhu brání rozšíření kompresorů vnímajících XML strukturu další faktory zjištěné během testování, mezi které patří nízká míra dostupnosti a podpory, kdy například mnoho kompresorů využívá již nepodporované knihovny a je po letech stále v beta verzi, značná chybovost, která například v případě z pohledu efektivity komprese nejlepšího kompresoru XMLPPM vedla v určitém případě k nenávratné ztrátě komprimovaných dat či vyšší komplexita oproti obecným textovým kompresorům, které dosahují mírně horších výsledků komprese, nicméně jsou snadno dostupné, plně podporované, jednodušší a plně stabilní. Při finálním zhodnocení bylo tedy možné doporučit mimo velmi efektivního obecného textového kompresoru PPM pouze XML-WRT, jelikož tento XML kompresor je relativně podporovaný, stabilní, rychlý a dostupný.

I přes zmíněné problémy může komprese XML dat specializovanými XML strukturu vnímajícími kompresory vést, v případech, kdy je potřebné skladování a práce s velmi velkými kolekcemi XML dokumentů v řádu stovek gigabytů až terabytů, jako tomu je například u projektu OpenStreetMap či stáhnutelnému obsahu Wikipedie, k značné úspoře kapacity datového úložiště či času přenosu dat po síti, oproti použití běžných kompresních metod jako Gzip či bzip2. V tomto ohledu by byly největším přínosem XML kompresory s podporou dotazování, které by pro určité případy umožnily práci s dokumentem bez nutnosti dekomprese a další komprese, nicméně veřejná míra dostupnosti této podskupiny XML specifických kompresorů je velmi malá a nepodařilo se takový kompresor zařadit do testu.

Na poznatky ohledně jednotlivých kompresních technik zjištěné v této diplomové práci by bylo možné navázat aktualizací a optimalizací otevřeného kódu perspektivních kompresorů jako XMLPPM, v současnosti neefektivního XMill v konfiguraci PPM, nezprovozněného XGrind či XML-WRT při jiném nastavení než zlib, které by vedly ke stabilizaci, kompatibilitě s moderními systémy či optimalizaci efektivity komprese, čímž by bylo umožněno rozšíření XML strukturu vnímajících kompresorů ze současné pozice velmi okrajového odvětví datové komprese.

7 Seznam použitých zdrojů

1. **Friesen, Jeff**. *Java XML and JSON.* New York, NY : Apress, 2016. ISBN 978-1484219157.

2. **Joshi, Bipin**. *Beginning XML with C# 7 : XML processing and data access for C# developers.* Berkeley, CA : Apress, 2017. ISBN 9781484231043.

3. **Genevès, Pierre.** *Course: Introduction to XML.* Grenoble : University of Grenoble, 2013.

4. *Extensible Markup Language (XML) 1.0 (Fifth Edition).* [Online] W3C, 26. listopad 2008. [Citace: 5. leden 2019.] Dostupné z: https://www.w3.org/TR/REC-xml/.

5. *Extensible Markup Language (XML) 1.1 (Second Edition).* W3C. [Online] 2006. srpen 16. [Citace: 6. leden 2019.] Dostupné z: https://www.w3.org/TR/xml11/#sec-intro.

6. **Kosek, Jiří**. *XML pro každého.* Praha : Grada Publishing, 2000. ISBN 80-7169-860-1.

7. *XML Syntax Rules.* W3schools.com. [Online] [Citace: 7. leden 2019.] Dostupné z: https://www.w3schools.com/xml/xml_syntax.asp.

8. *XML Validator.* W3schools.com. [Online] [Citace: 2019. leden 7.] Dostupné z: https://www.w3schools.com/xml/xml_validator.asp.

9. *XML Elements.* W3schools. [Online] [Citace: 7. leden 2019.] Dostupné z: https://www.w3schools.com/xml/xml_elements.asp.

10. *XML pro web aneb od teorie k praxi, 8. díl - jmenné prostory a XHTML.* zive.cz. [Online] 24. leden 2003. [Citace: 8. leden 2019.] Dostupné z: https://www.zive.cz/clanky/xml-pro-web-aneb-od-teorie-k-praxi-8dil---jmenne-prostory-a-xhtml/sc-3-a-110128/default.aspx.

11. *Namespaces in XML 1.0 (Third Edition).* W3C. [Online] 8. prosinec 2009. [Citace: 10. leden 2019.] Dostupné z: https://www.w3.org/TR/xml-names/.

12. **Irena Mlýnová, Martin Nečaský, Jaroslav Pokorný, Kamil Toman, Vojtěch Toman, Karel Richta**. *XML technologie.* Praha : Grada Publishing a.s., 2008. ISBN 978-80-247-2725-7.

13. *DTD Entity*. Liquid Technologies. [Online] [Citace: 16. leden 2019.] Dostupné z: https://www.liquid-technologies.com/DTD/Structure/ENTITY.aspx.

14. *XML schema (XSD) Overview.* Liquid Technologies. [Online] [Citace: 16. leden 2019.] Dostupné z: https://www.liquid-technologies.com/xml-schema-tutorial/xsd-elements-attributes#complex-types.

15. *XML Schema Part 2: Datatypes Second Edition*. W3C. [Online] [Citace: 17. leden 2019.] Dostupné z: https://www.w3.org/TR/xmlschema-2/#built-in-datatypes.

16. *XML DOM - Overview.* tutorialspoint. [Online] [Citace: 16. leden 2019.] Dostupné z: https://www.tutorialspoint.com/dom/xml_dom_quick_guide.htm.

17. **Sayood, Khalid**. *Introduction to Data Compression 5th Edition.* Cambridge, MA : Elsevier, 2017. ISBN 978-0-12-809474-7.

18. **Colt McAnlis, Aleks Haecky**. *Understanding Compression: Data Compression for Modern Developers.* Sebastopol, CA : O'Reilly Media, Inc., 2016. ISBN 978-1491961537.

19. **Doa'a Saad El-Shora, Nabil Aly Lashin, Ehab Rushdy Mohamed, Ibrahim Mahmoud El-Henewy.** *Performance Evaluation of Data Compression Techniques versus Different Types of Data.*. 12, Sangdo : International Journal of Computer Science and Information Security, 2013, Sv. 11. ISSN 1947-5500.

20. *Data compression.* Encyclopædia Britannica. [Online] Encyclopædia Britannica, inc. , 23. duben 2013. [Citace: 5. únor 2019.] Dostupné z: https://www.britannica.com/technology/data-compression.

21. **Sakr, Sherif.** *Investigate state-of-the-art XML compression techniques.* místo neznámé : IBM Corporation, 2011.

22. **Wilfred Ng, Lam Wai Yeung, James Cheng.** *Comparative Analysis of XML Compression Technologies.* 1, Hong Kong : World Wide Web, 2006, Sv. 9. ISSN 1386-145X.

23. **Christopher J. Augeri, Dursun A. Bulutoglu, Barry E. Mullins, Rusty O. Baldwin, Leemon C. Baird.** *An Analysis of XML Compression Efficiency.* 7, San Diego, CA : Proceedings of the 2007 workshop on Experimental Computer Science, 2007. ISBN: 978-1-59593-751-3.

24. *Huffman coding.* Indiana University Bloomington - Sc. [Online] [Citace: 9. únor 2019.] Dostupné z: http://homes.sice.indiana.edu/yye/lab/teaching/spring2014-C343/huffman.php.

25. Nelson, Mark. *Data Compression with Arithmetic Encoding*. Dr. Dobb's - The World of Software Development. [Online] 4. listopad 2004. [Citace: 10. únor 2019.] Dostupné z: http://www.drdobbs.com/cpp/data-compression-with-arithmetic-encodin/240169251.

26. **Asadollah Shahbahrami, Ramin Bahrampour, Mobin Sabbaghi Rostami, Mostafa Ayoubi Mobarhan.** *Evaluation of Huffman and Arithmetic Algorithms for Multimedia.* 4, místo neznámé : International Journal of Computer Science, Engineering and Applications (IJCSEA), 2011, Sv. 1.

27. **Savan Oswal, Anjali Singh, Kirthi Kumari.** *Deflate Compression Algorithm.* 1, místo neznámé : International Journal of Engineering Research and General Science, 2016, Sv. 4. ISSN 2091-2730.

28. **Sakr, Sherif.** *XML compression techniques: A survey and comparison.* 5, Sydney : Journal of Computer and System Sciences, 2009, Sv. 75.

29. **Kheirkhahzadeh, Antonio D.** *On The Performance of Markup Language Compression.* London : University of West London, 2015.

30. **Tomasz Müldner, Tyler Corbin, Jan Krzysztof Miziolek, Christopher Fry.** *Design and Implementation of an Online XML Compressor for Large XML files.* 3, místo neznámé : International Journal on Advances in Internet Technology, 2012, Sv. 5.

31. **Tyler Corbin, Tomasz Müldner, Jan Krzysztof Miziolek.** *Pre-order Compression Schemes for XML in the Real Time Environment.* místo neznámé : WEBIST, 2013.

32. *gzip.* gzip.org. [Online] 2018. [Citace: 14. únor 2019.] Dostupné z: https://www.gzip.org/.

33. *bzip2.org.* [Online] 21. září 2018. [Citace: 16. únor 2019.] Dostupné z: http://www.bzip.org/.

34. **J. G. Cleary, W. J. Teahan.** *Unbounded Length Contexts for PPM.* 2 a 3, Swindon : The Computer Journal, 1997, Sv. 40.

35. **Hartmut Liefke, Dan Suciu.** *XMill: An efficient compressor for XML data.* 2, Dallas, TX : SIGMOD Record (ACM Special Interest Group on Management of Data), 2000, Sv. 29.

36. **Cheney, James.** *Compressing XML with Multiplexed Hierarchical PPM Models.* Washington, DC : IEEE Computer Society, 2001.

37. **Cheney, James.** *An Empirical Evaluation of Simple DTD-Conscious Compression Techniques.* Baltimore, MD : Eight International Workshop on the Web & Databases, 2005.

38. **Cheney, James.** *Tradeoffs in XML Database Compression.* Snowbird, UH : IEEE Computer Society - Data Compression Conference, 2006.

39. **Pankaj M. Tolani, Jayant R. Haritsa.** *XGRIND: A Query-Friendly XML Compressor.* San Jose, CA : IEEE International Conference on Data Engineering (ICDE), 2002.

40. **Jun-Ki Min, Myung-Jae Park, Chin-Wan Chung.** *XPRESS: a queriable compression for XML data.* New York, NY : Proceedings of the 2003 ACM SIGMOD Management of data (SIGMOD '03), 2003.

41. *Efficient XML Interchange (EXI) Primer.* w3c.org. [Online] W3C, 24. duben 2014. [Citace: 20. únor 2019.] Dostupné z: https://www.w3.org/TR/exi-primer/.

42. **Albrecht Schmidt, Florian Waas, Martin Kersten, Michael J. Carey, Ioana Manolescu, Ralph Busse.** *XMark: a benchmark for XML data management.* Hong Kong : Proceedings of the 28th international conference on Very Large Data Bases, 2002.

43. *XML Data Repository.* Washington.edu. [Online] [Citace: 24. únor 2019.] Dostupné z: aiweb.cs.washington.edu/research/projects/xmltk/xmldata/www/repository.html.

44. *Wikimedia Downloads*. [Online] 23. únor 2019. [Citace: 25. únor 2019.] Dostupné z: https://dumps.wikimedia.org/backup-index.html.

45. **Bosak, Jon.** *The Plays of Shakespeare in XML*. [Online] 20. červenec 1999. [Citace: 25. únor 2019.] Dostupné z: http://xml.coverpages.org/bosakShakespeare200.html.

46. *Planet.osm.* OpenStreetMap. [Online] 27. únor 2019. [Citace: 27. únor 2019.] Dostupné z: https://wiki.openstreetmap.org/wiki/Planet.osm.

47. **Suciu, Dan**. *XMILL*. University of Washington Computer Science & Engineering community. [Online] [Citace: 26. únor 2019.] Dostupné z: https://homes.cs.washington.edu/~suciu/XMILL/.

48. **Cheney, James**. *XML Compression Tools*. Sourceforge. [Online] 17. duben 2013. [Citace: 3. březen 2019.] Dostupné z: https://sourceforge.net/projects/xmlppm/.

49. **Skibinski, Przemyslaw.** *XWRT*. GitHub. [Online] 16. srpen 2016. [Citace: 8. březen 2019.] Dostupné z: https://github.com/inikep/XWRT.

50. *Java Implementations*. EXIficient. [Online] 8. listopad 2018. [Citace: 9. březen 2019.] Dostupné z: https://exificient.github.io/java/.

51. **Przemyslaw Skibinski, Szymon Grabowski, Jakub Swacha.** *Fast transform for effective XML compression.* místo neznámé : 9th International Conference - The Experience of Designing and Applications of CAD Systems in Microelectronics, 2007.

8 Přílohy

Příloha č. 1 – Naměřené hodnoty

Příloha 1 – Naměřené hodnoty

Gzip					
XML dokument	velikost dokumentu [B]	velikost po kompresi [B]	kompresní poměr	průměrná rychlost komprese [s]	průměrná rychlost dekomprese [s]
XMark-small	11 669 761	3 804 595	67,40%	0,54	0,12
XMark-medium	116 517 364	38 095 435	67,30%	5,27	0,99
XMark-large	585 540 615	192 017 468	67,21%	26,8	5,18
cswiki_abstract	420 340 124	57 587 358	86,30%	8,04	2,57
Lucembursko	416 510 304	49 631 411	88,08%	8,23	2,32
Shakespeare	2 386 450	661 805	72,27%	0,11	0,03
SwissProt	114 820 211	14 140 327	87,68%	2,01	0,61
TPC-H	40 997 869	3 983 569	90,28%	0,73	0,21
Treebank	86 082 517	32 388 297	62,38%	3,74	0,66

Bzip2					
XML dokument	velikost dokumentu [B]	velikost po kompresi [B]	kompresní poměr	průměrná rychlost komprese [s]	průměrná rychlost dekomprese [s]
XMark-small	11 669 761	2 567 817	78,00%	1,34	0,44
XMark-medium	116 517 364	25 546 036	78,08%	13,22	4,54
XMark-large	585 540 615	128 650 887	78,03%	66,51	22,55
cswiki_abstract	420 340 124	40 956 676	90,26%	68,3	11,09
Lucembursko	416 510 304	36 779 152	91,17%	54,17	8,93
Shakespeare	2 386 450	380 999	84,03%	0,3	0,1
SwissProt	114 820 211	8 723 104	92,40%	15,2	2,85
TPC-H	40 997 869	1 943 155	95,26%	6,68	0,96
Treebank	86 082 517	26 958 932	68,68%	8,43	3,42

PPM					
XML dokument	velikost dokumentu [B]	velikost po kompresi [B]	kompresní poměr	průměrná rychlost komprese [s]	průměrná rychlost dekomprese [s]
XMark-small	11 669 761	2 312 856	80,18%	0,58	0,66
XMark-medium	116 517 364	22 504 640	80,69%	5,59	6,06
XMark-large	585 540 615	112 619 851	80,77%	32,09	32,63
cswiki_abstract	420 340 124	45 250 632	89,23%	17,55	19,99
Lucembursko	416 510 304	36 044 911	91,35%	11,9	14,37
Shakespeare	2 386 450	415 967	82,57%	0,13	0,15
SwissProt	114 820 211	10 828 410	90,57%	4,32	4,95
TPC-H	40 997 869	3 097 786	92,44%	1,17	1,37
Treebank	86 082 517	28 679 224	66,68%	14,26	15,06

XMill (Gzip)					
XML dokument	velikost dokumentu [B]	velikost po kompresi [B]	kompresní poměr	průměrná rychlost komprese [s]	průměrná rychlost dekomprese [s]
XMark-small	11 669 761	3 432 469	70,59%	0,82	0,39
XMark-medium	116 517 364	34 130 582	70,71%	7,81	2,33
XMark-large	585 540 615	171 776 153	70,66%	38,52	6,54
cswiki_abstract	420 340 124	60 095 601	85,70%	13,48	3,68
Lucembursko	416 510 304	36 879 749	91,15%	17,81	5,65
Shakespeare	2 386 450	615 977	74,19%	0,21	0,15
SwissProt	114 820 211	8 879 681	92,27%	3,77	1,06
TPC-H	40 997 869	2 284 123	94,43%	1,28	0,98
Treebank	86 082 517	27 183 497	68,42%	4,36	0,83

XMill (Bzip2)					
XML dokument	velikost dokumentu [B]	velikost po kompresi [B]	kompresní poměr	průměrná rychlost komprese [s]	průměrná rychlost dekomprese [s]
XMark-small	11 669 761	2 616 930	77,58%	4,54	1,29
XMark-medium	116 517 364	25 711 550	77,93%	48,93	9,83
XMark-large	585 540 615	129 414 046	77,90%	220,8	44,15
cswiki_abstract	420 340 124	44 254 924	89,47%	154,57	24,89
Lucembursko	416 510 304	36 168 560	91,32%	256,6	41,25
Shakespeare	2 386 450	418 213	82,48%	1,22	0,33
SwissProt	114 820 211	7 012 988	93,89%	64,96	8,78
TPC-H	40 997 869	1 596 516	96,11%	10,45	2,66
Treebank	86 082 517	25 169 275	70,76%	36,9	4,28

XMill (PPM)					
XML dokument	velikost dokumentu [B]	velikost po kompresi [B]	kompresní poměr	průměrná rychlost komprese [s]	průměrná rychlost dekomprese [s]
XMark-small	11 669 761	2 728 598	76,62%	7,39	8,66
XMark-medium	116 517 364	27 149 707	76,70%	72,9	84,44
XMark-large	585 540 615	136 645 799	76,66%	374,28	315,8
cswiki_abstract	420 340 124	42 775 891	89,82%	163,01	177,13
Lucembursko	416 510 304	34 767 034	91,65%	179,76	179,87
Shakespeare	2 386 450	479 375	79,91%	1,52	1,69
SwissProt	114 820 211	6 844 303	94,04%	38,67	35,9
TPC-H	40 997 869	1 620 268	96,05%	8,72	7,55
Treebank	86 082 517	26 062 227	69,72%	50,37	57,46

XMLPPM					
XML dokument	velikost dokumentu [B]	velikost po kompresi [B]	kompresní poměr	průměrná rychlost komprese [s]	průměrná rychlost dekomprese [s]
XMark-small	11 669 761	2 347 778	79,88%	1,98	2,23
XMark-medium	116 517 364	23 379 734	79,93%	19,35	22,14
XMark-large	585 540 615	117 747 885	79,89%	99,97	111,17
cswiki_abstract	420 340 124	32 545 026	92,26%	30,79	39,37
Lucembursko	416 510 304	34 184 204	91,79%	37,16	NELZE - CHYBA
Shakespeare	2 386 450	344 822	85,55%	0,34	0,39
SwissProt	114 820 211	6 680 099	94,18%	8,78	10,47
TPC-H	40 997 869	1 603 681	96,09%	2,15	2,5
Treebank	86 082 517	26 188 143	69,58%	17,63	19,46

XML-WRT					
XML dokument	velikost dokumentu [B]	velikost po kompresi [B]	kompresní poměr	průměrná rychlost komprese [s]	průměrná rychlost dekomprese [s]
XMark-small	11 669 761	2 219 061	80,98%	0,68	0,28
XMark-medium	116 517 364	21 180 200	81,82%	6,01	1,97
XMark-large	585 540 615	106 105 819	81,88%	29,83	8,76
cswiki_abstract	420 340 124	40 984 504	90,25%	23,13	7,15
Lucembursko	416 510 304	30 550 130	92,67%	21,2	6,41
Shakespeare	2 386 450	449 515	81,16%	0,21	0,11
SwissProt	114 820 211	6 817 679	94,06%	5,78	2,06
TPC-H	40 997 869	1 676 605	95,91%	1,48	0,71
Treebank	86 082 517	29 634 424	65,57%	8,25	1,92

EXI					
XML dokument	velikost dokumentu [B]	velikost po kompresi [B]	kompresní poměr	průměrná rychlost komprese [s]	průměrná rychlost dekomprese [s]
XMark-small	11 669 761	3 405 131	70,82%	3,31	2,23
XMark-medium	116 517 364	26 529 149	77,23%	29,5	21,21
XMark-large	585 540 615	93 613 892	84,01%	113,04	85,38
cswiki_abstract	420 340 124	57 035 586	86,43%	112,37	91,73
Lucembursko	416 510 304	33 535 344	91,95%	96,94	59,78
Shakespeare	2 386 450	463 275	80,59%	1,13	0,82
SwissProt	114 820 211	7 328 466	93,62%	18,22	12,72
TPC-H	40 997 869	2 132 379	94,80%	5,19	4,14
Treebank	86 082 517	26 257 392	69,50%	20,03	14,29