Překladač XSL šablon

Marek Běhálek

Katedra informatiky, FEI, VŠB – Technická Univerzita Ostrava
17. listopadu 15, 708 33, Ostrava-Poruba
Marek.Behalek@vsb.cz

Abstrakt. Jazyk XML definuje, jak vytvářet sebepopisující dokumenty. Tento jazyk je dnes obecně znám a hojně využíván. XSLT je komplementární jazyk, popisující jak transformovat XML dokumenty. Díky těmto transformacím lze v XML dokumentech například hledat nebo měnit strukturu dat. XSLT definuje transformace, které by měly být provedeny, ale nedefinuje způsob, jak tyto operace provést. V dnešní době pracuje většina XSLT procesorů (aplikací provádějících tyto transformace) jako interpret. Použijeme-li kompilátor na místo interpretu, můžeme dosáhnout značného zrychlení. Tento příspěvek popisuje postup při vytváření takovéhoto XSLT procesoru a výhody, které tento přístup přináší.

Klíčová slova: kompilační překladač, XML, XSL, XSLT procesor

1 Úvod

XML dokumenty jsou v dnešní době hojně využívány a to nejen k reprezentaci textu, ale celé databáze jsou ukládány jako XML dokumenty. Takovéto dokumenty mohou být rozsáhlé a práce s nimi výpočetně náročná. Tento fakt může být zvláště důležitý, použijeme-li nějaké mobilní zařízení, například PDA.

2 XML

Je několik definic popisujících, co je XML (eXtensible Markup Language). Jedna z nich je tato: XML je meta-značkovací jazyk, který definuje syntaxi využívanou k definici dalších doménově orientovaných, sémantických, strukturovaných značkovacích jazyků. Jinými slovy, XML definuje několik pravidel pro vytváření sémantických tagů, které dělí dokument na části a tyto části identifikují. Tento jazyk byl vytvořen společností W3C. Tento formát je volně dostupný. XML je podporován většinou stávajících programovacích jazyků. Je to flexibilní cesta, jak vytvářet sebepopisující dokumenty a sdílet v jednom souboru data i formát. Dobře utvořený XML dokument má stromovou strukturu.

2.1 XSL

XSL je jazyk pro vytváření šablon použitelných při transformaci nebo formátování XML dokumentů. Tento jazyk se skládá ze dvou částí: jazyka pro transformování XML dokumentů (XSLT) a XML slovníku specifikujícího formátovací sémantiku. V tomto příspěvku se budeme zabývat pouze první částí. Transformace obsažena v XSLT šabloně definuje pravidla, popisující způsob, jakým transformujeme zdrojový XML dokument do nového cílového dokumentu. Výsledkem XSLT transformace může být:

- nový XML dokument – tento dokument může mít rozdílnou strukturu. Tímto způsobem lze vytvářet různé „pohledy“ na stávající data (stejná data seřazená podle různých kritérií, výběr důležitých informací atd.);
- HTML dokument – tímto způsobem můžeme přidat prezentační vrstvu k čistě datově orientovaným XML dokumentům.

XSLT šablona je také XML dokument. Tento dokument využívá elementů z XSLT slovníku k popisu transformace. Kořenem dokumentu je element: <*xsl:stylesheet*>. Tento element pak obsahuje sadu pravidel, popisující konkrétní transformace, které mají být provedeny. Jednotlivá pravidla jsou obsažená v elementech <*xsl:template*>. Důležitým atributem tohoto elementu je atribut *match*. Tento atribut obsahuje cestu v jazyce XPath. Tato cesta definuje množinu uzlů zdrojového dokumentu. Na tuto množinu je pak pravidlo aplikováno. Jednotlivá pravidla jsou označována jako vzory, protože informace obsažené v jejich těle slouží jako model pro konstrukci části výsledného stromu (XML dokumentu).

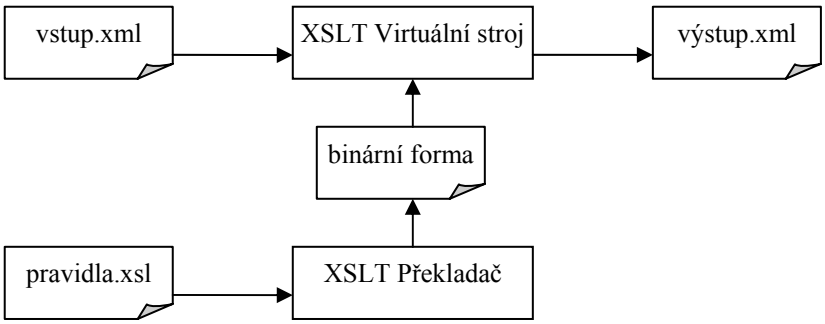
XPath je jazyk sloužící k adresaci uzlů v rámci XML dokumentu. Navíc také poskytuje základní vybavení pro manipulaci s řetězci, čísly, případně hodnotami typu bool. Cesta v jazyce XPath je posloupnost vzorů, oddělených lomítkem, definujících cestu ve stromové struktuře XML dokumentu v závislosti na aktuálním kontextu.

3 Popis problému

Vezměme si modelovou situaci: Aplikace, kterou budujeme, je postavena na architektuře Klient-Server. Server je výkonný počítač s prakticky neomezeným množstvím paměti a velkým výpočetním výkonem. Na druhé straně je klient. Klientem může být zařízení jako PocketPC. Takovéto zařízení má omezenou paměť i výpočetní výkon. Naše aplikace využívá prostředí Internetu. Klient je připojen prostřednictvím mobilního telefonu. Takovýto klient se stará o to, kolik dat je nucen „stáhnout“ a také nechce být připojen pořád. Použít XML a XSLT je v tomto případě velice výhodné. Klient si uloží data jako XML dokument pouze jednou a v případě že potřebuje jiný pohled na stávající data, aplikuje nějakou XSLT šablonu provádějící potřebnou operaci. Další výhodou je, že klient si tyto šablony může připravit předem a může je používat „off-line“. Tento přístup umožňuje zmenšit množství dat, které klient musí uložit a stáhnout, ale teď je to klient, kdo musí provést transformaci a to může být časově velice náročné.

4 Kompilování XSLT šablon

Většina XSLT procesorů, aplikací provádějících transformace, funguje jako interpret. Provádění této transformace lze značně urychlit, využijeme-li výhod kompilačního překladače (obr. 1.). Důvody jsou stejné jako u programovacích jazyků. Provádění předem připravené *binární formy* je rychlejší. Další výhodou je, že provádění binární formy již nepotřebuje původní překladač, ale jen příslušný fyzický nebo virtuální stroj. Kompilační XSLT procesor je tedy rozdělen na dvě části: překladač transformující XSLT šablonu do binárního kódu a příslušný virtuální stroj. V našem modelovém příkladě to tedy znamená, že transformace je připravena na straně serveru a klient si uloží jen výsledek relativně malou binární formu. Také nepotřebuje mít uložen celý XSLT procesor, ale jen příslušný virtuální stroj.



Obr. 1. Struktura XSLT procesoru kompilujícího XSLT šablony.

Použití kompilačního překladače místo interpretu také jasně vymezuje čas kompilace a čas běhu transformace. Hranice mezi těmito dvěmi procesy není pevně daná a závisí na úrovni strojových instrukcí. Rozdělení jedné heterogenní instrukce na několik atomických instrukcí umožňuje provést některé operace již v době překladu. Používáme-li například jednu obecnou CMP instrukci, je typová kontrola provedena až v době běhu. Rozdělíme-li tuto instrukci na několik porovnávacích instrukcí, jednu pro každý typ, umožníme tak překladači, aby se o testování typové korektnosti postaral již v době překladu.

Další výhodou tohoto přístupu je, že je-li XSLT šablona jednou zkompilována, lze ji pak pro danou transformaci využít několikrát. Tato vlastnost může mít zásadní vliv u aplikací, které často provádějí stejné transformace nad měnícími se daty. Příkladem mohou být webové servery. Generuje-li takový server HTML stránku z XML dokumentu s použitím XSLT šablony, může být čas potřebný pro zpracování šablony srovnatelný s dobou trvání vlastní transformace. Je-li tato operace prováděná mnohokrát, může použití kompilačního překladače přinést až několikanásobné zrychlení.

5 Související práce

Koncept kompilování XSLT šablon do binární formy není nový. Jako první ho navrhli A. Novoselsky a K. Karum [1]. V jejich příspěvku byl představen nejen koncept kompilování XSLT šablon, ale také zde navrhují virtuální stroj, simulující imaginární CPU, určené pro provádění XSLT transformací. Tento XSLT CPU byl představen jen jako model, na kterém prezentovali svou myšlenku kompilování XSLT šablon. Neexistuje exaktní specifikace ani není dostupná implementace takovéhoto procesoru, jen krátký článek navrhující toto řešení.
Myšlenka kompilování XSLT šablon je vestavěna do XSLT procesorů některých společností (například Oracle Inc., Ambrosoft Inc. nebo Apache Software Foundation). Všechny tyto aplikace se chovají podobně. Jako model popisující jejich chování lze využít Xalan. Tento produkt Apache Software Fundation je volně dostupný a je k dispozici na webových stránkách výrobce. Tento produkt je úzce svázán s Javou. Binární forma kterou tento XSLT procesor generuj je „class“ soubor Javy.

6 Navrhované řešení problému

Použití „class“ souborů Javy jako binární formy XSLT procesoru svazuje tuto aplikaci příliš úzce s Javou. Je prakticky nemožné přidat podporu pro jinou platformu (například Microsoft .NET). Jsou-li při generování binární formy použity instrukce nějakého imaginárního procesoru pro provádění XSLT šablon, přináší to značnou platformní nezávislost. Binární forma generována XSLT překladačem je pak sekvence těchto instrukcí a běhové prostředí je příslušný virtuální stroj, provádějící tyto instrukce. Kompilátor, vytvářející binární formu (XSLT program), by měl generovat stejný výsledek nezávisle na platformě. Stejně tak by měl virtuální stoj na základě XSLT programu provést stejnou transformaci, ať už byl implementován v Javě, C/C++/C# nebo jiném programovacím jazyce. Tento přístup také umožňuje lépe využít speciální možnosti jednotlivých platforem.

7 Specifikace XSLT procesoru

Komplexní popis XSLT procesoru je mimo možnosti tohoto příspěvku. Proto zde budou prezentovány pouze základní vlastnosti. Vlastní procesor se skládá ze dvou samostatných celků: XSLT překladače a XSLT virtuálního stroje. XSLT překladač je aplikace, která k XSLT šabloně vygeneruje binární reprezentaci (XSLT program). Tento program se skládá z instrukcí imaginárního XSLT CPU navrženého speciálně pro provádění XSLT transformací. Virtuální stroj pak tento imaginární procesor simuluje.
Jak již bylo zmíněno, hranice mezi operacemi prováděnými v době překladu a operacemi prováděnými v době běhu závisí na úrovni strojových instrukcí. Pro potřeby XSLT CPU byly zvoleny instrukce nízké úrovně. Velikost XSLT šablon je

obecně malá a překladač může provést mnoho kontrol již v době překladu. Použité instrukce jsou většinou „zásobníkové“. Berou si hodnoty operandů z vrcholu zásobníku a nahrazují je výsledkem. Instrukce XSLT CPU mohou být rozděleny do tří skupin: systémové instrukce (cykly, instrukce pro větvení, …), instrukce pro zpracovávání XPath elementů a instrukce provádějící vlastní transformace.
Proces kompilování se skládá ze dvou relativně samostatných fází. První je vytváření reprezentace XPath elementů. Tyto instrukce vyberou uzly, na které bude aplikován daný vzor. Další fází je generování instrukcí pro konkrétní vzory. Tyto instrukce provedou vlastní transformaci.
Hlavní paměť virtuálního stroje je zásobník. Na zásobníku jsou uloženy ukazatele na proměnné a částečné výsledky. Další paměť obsažená ve virtuálním stroji je hromada. Zde jsou uloženy hodnoty jednotlivých proměnných případně jiných objektů.

8 Experimentální výsledky a budoucí možnosti dalšího výzkumu

V rámci testování je vytvářen prototyp XSLT překladače. Jako základ je využito částí XSLT procesoru firmy Apache Software Foundation Xalan. Tento produkt je implementován v Javě a je „open source“. Nejdůležitější změnou je proces tvorby binární reprezentace. Místo „class“ souborů Javy je vytvářen soubor s instrukcemi navrženého XSLT CPU. Běhové prostředí (virtuální stroj) bude implementováno v jazyce C# a bude určeno pro platformu Microsoft .NET Compact Framework. Tato restrikce platformy .NET je určena pro mobilní zařízení. Implementace tohoto řešení je stále ve fázi vývoje.
Nejsložitější operací při provádění transformací je nalezení elementů, pro které jsou definované vzory v XSL šabloně. Tyto elementy jsou adresovány pomocí cesty v jazyce XPath. S tímto problémem úzce souvisí, jak vůbec efektivně zpracovávat XML dokument. Další práce se tedy bude zabývat právě optimalizacemi těchto procesů.

9 Závěr

Technika kompilování XSLT šablon má širokou škálu použití zejména ve webových aplikacích a ve chvíli, kdy používáme mobilní zařízení jako PDA. Díky tomuto přístupu můžeme značně urychlit transformování XML dokumentů a tedy efektivněji provádět operace jako vyhledávání dat nebo třídění. Tento přístup také otevírá nové možnosti na poli optimalizace.

Reference:

1. A. Novoselsky, K. Karun. XSLTVM - an XSLT Virtual Machine. http://www.gca.org/papers/xmleurope2000/papers/s35-03.html/.

2. Jacek R. Ambroziak.:Gregor, the next generation XSLT compiler. http://www.ambrosoft.com/.
3. The Apache Software Foundation: Xalan-Java documentation. http://xml.apache.org/xalan-j/.
4. World Wide Web Consortium.: Extensible Markup Language (XML) 1.0. http://www.w3c.org/xml/
5. World Wide Web Consortium.:XSL. See:http://www.w3c.org/xsl/
6. Oracle Corporation.: Transforming XML with XSLT. http://technet.oracle.com/tech/xml/techinfo.html.

Annotation:

XSL Stylesheet compiler

XML is a flexible way how to create a self describing data and share both the format and the data trough the Internet. XSLT provides the complementary language describing how to perform a transformation of a source XML document into a result new XML document. XSLT defines transformations which should be processed but no how to do them. Almost all XSLT processors (applications that perform transformations) that are currently available work like interprets. Similarly like in programing languages also here using a compiler instead of a interpret can dramatically improve performance. This article describe different approaches for creating such a compiler.

.