COMPILER OF EQUATION IN ENERGETIC INDUSTRY

Petr Pehal
Bachelor Degree Programme (3), FIT VUT
E-mail: xpehal00@stud.fit.vutbr.cz

Supervised by: Martin Čermák
E-mail: icermak@fit.vutbr.cz

ABSTRACT

This paper describes specialized programming language used for specific calculations in energetic industry. First, a quick language description with short example is given. Then we discuss current language state along with its usage and finally its extensions and optimizations and further development.

1 ÚVOD

Energetický průmysl patří bezesporu mezi nejvýznamnější odvětví světového průmyslu. Firma Elektrosystem a.s. vyvíjí pro energetické společnosti informační a řídící systémy. V současnosti převádí svůj systém Ris do z real-time operačního systému QNX [1] na multiplatformní. Součástí tohoto převodu je i úprava a rozšíření jazyka dopočtových rovnic.

Řízení energetických sítí je velmi komplexní záležitost. Informace jsou získávány z různých zdrojů. Jedním z nich je výpočet aktuálně nedostupných hodnot z hodnot známých. Za tímto účelem využívá systém Ris jazyk pro definici dopočtových rovnic.

2 DOPOČTOVÝ JAZYK

Jedná se o jazyk interpretovaný. Zdrojový kód je nejdříve přeložen do bytecodu, který je po zavedení dopočtu interpretován dopočtovým aparátem. Dopočty jsou dvojího typu, periodické a změnové. Periodické dopočtové rovnice jsou interpretovány vždy jednou za danou periodu jako celek. Změnové dopočty jsou interpretovány pouze při změně některého ze vstupních parametrů, přičemž jsou interpretovány pouze rovnice obsahující daný parametr jako vstupní. Tento výpočet může spustit další změnový dopočet, pokud jsou výstupní parametry z právě vypočtených rovnic vstupními parametry rovnic jiných. Interpret přitom kontroluje cyklické závislosti.

Jazyk je ze syntaktického hlediska inspirován jazykem C. Obsahuje velké množství vestavěných funkcí a operátorů. Kromě operátorů známých z jazyka C přidává jazyk speciální operátory, např. *energetické and* nad energetickými signály. Vestavěné funkce se opět dělí na dvě skupiny. Matematické funkce, jako integral či sinus, a funkce specifické pro energetiku, např. funkce *alr_rc* (test dalkového ovládání veličiny).

Zásadní odlišností od procedurálních jazyků jsou proměnné. Proměnné reprezentují konkrétní položky z databáze. Systém Ris používá vlastní real-time databázový engine. Jedná se o relační

databázi, proměnné jsou tedy obecně *n*-rozměrné vektory, které představují řádek dané tabulky. V praxi se ale využívají vektory 4-rozměrné případně 5-rozměrné: jméno (název veličiny, přes který se k ní přistupuje), hodnota (aktuální hodnota veličiny), kvalita (kvalita hodnoty veličiny, například telemetrická), případně alarm (příznak určující různé stavy veličiny) a čas (čas zjištění hodnoty veličiny). Před použitím proměnných je třeba definovat tzv. zdroj, který určuje tabulku a její sloupce. Zdroj vymezí umístění a rozměr proměnných. Po definovaní zdroje je možné transparentně přistupovat k databázovým datům pomocí proměnných, což zajišťuje perzistenci dopočtených hodnot.

Pro představu přikládáme krátký výsek zdrojového kódu periodických dopočtů podílejících se na řízení energetických sítí v České Republice.

```
// nastaveni periody dopoctu
def_period = 1;
// definice zdroje
source A = '/rdb/analog', name, 'value quality ~alarm';
// konkretni dopoctova rovnice
A:gAGC:SL:t_st = iif(!qvalid(A:gAGC:SL_M:P),_N(A:gAGC:SL_M:P));
```

3 SOUČASNÝ STAV JAZYKA

Současný dopočtový jazyk řeší v systému Ris velmi závažné úkoly a uživatelské zdrojové kódy jsou značně rozsáhlé. Samotný překladač a interpret jsou však ve velmi neuspokojivém a nadále obtížně udržovatelném stavu, protože během používání byly obě části značně nastavovány a upravovány podle přání uživatelů. Tyto úpravy s sebou přinesly problém mnohem závažnější. Kvůli úpravám utrpěla rychlost interpretu a nyní pro objemnější dopočty místy přestává dostačovat. Bylo tedy rozhodnuto o kompletní rekonstrukci překladače i interpretu. V důsledku tohoto kroku je možné současně jazyk rozšířit o další chybějící prvky.

4 ÚPRAVY A ROZŠÍŘENÍ

Základním nedostatkem jazyka je nemožnost definice uživatelských funkcí, konstant a pomocných proměnných, což často vede k nepřehledným rovnicím. Kvůli problémům s kompatibilitou diskutovaným dále bylo rozhodnuto tento nedostatek řešit přidáním preprocesoru.

Pro implementaci překladače byly využity programy *flex* a *bison*, které přispějí k lepší udržovatelnosti kódu a ušetří čas, který je nutné věnovat optimalizacím interpretu a kvalitnímu testování. Chyby v interpretu by vzhledem k závažnosti úloh, za které dopočtový jazyk v současnosti odpovídá, mohly mít velmi nepříjemné následky.

4.1 OMEZENÍ SPOJENÉ S KOMPATIBILITOU

Vzhledem k tomu, že úpravy jazyka jsou pouze rozšiřujícího a optimalizačního charakteru, zajištění zpětné kompatibility s původní verzí nepředstavuje problém. Problémem se ukázala kompatibilita při spolupráci s ostatními programy systému Ris, především s programem pro práci se statistikami. Ve chvíli zavedení dopočtu si dopočtový aparát nahraje bytecode a pomocí interpretu jej interpretuje. Zároveň se ale bytecode nahraje i do programu pro práci se statistikami, který s ním pracuje bez využití interpretu. Jedná se o zásadní problém, který znemožňuje

rozšíření bytecodu bez úpravy programu pro práci se statistikami. Tyto úpravu v současné situaci nejsou možné. Proto bylo rozhodnuto přidat preprocesor, který umožní jazyk rozšířit bez zásahu do bytecodu. Pomocné proměnné jsou zavedeny opět bez zásahu do bytecodu. Každý dopočtový soubor má implicitně definovaný zdroj *W*, který zpřístupňuje tabulku sloužící pro pomocné proměnné.

4.2 PREPROCESOR

Nově zavedený preprocesor je implementovaný pomoci utility *flex*. Umožňuje textové vkládání externích dopočtových souborů, definici konstant a maker, včetně maker s parametry, která alespoň částečně nahradila uživatelské funkce.

4.3 OPTIMALIZACE INTERPRETU

Zásadním úkolem bylo zvýšení rychlosti interpretu. Při implementaci byl kladen důraz na efektivitu (bitové operace, příkaz goto, ...), srozumitelnost kódu byla až druhořadá.

Rozpracovaná verze interpretu využívá v maximální míře vysoce optimalizavaný real-time databázový engine systému Ris. Databázový engine umožňuje s tabulkami pracovat přímo v operační paměti. Zároveň je možné mezi paměťovými tabulkami různých procesů komunikovat prostřednictvím volání registrovaných uživatelských funkcí. Meziprocesová komunikace využívá proprietární protokol postavený nad TCP/IP. Celý bytecode je složitá struktura tabulek zapouzdřená v jedné tabulce tzv. *headru*. Součástí bytecodu je seznam odkazů do všech tabulek, které v dopočtových rovnicích vystupují. Při zavedení dopočtů registruje interpret u těchto tabulek uživatelské funkce. Pomoci nich při interpretaci detekuje změny, které při změnových dopočtech aktuvují výpočet.

Výrazným optimalizačních krokem byla specializace interpretu na specifický formát vektorů zmiňovaný již dříve (jméno, hodnota, kvalita, alarm, čas). Obecné vektory nejsou a ani nebudou v energetických dopočtech využívány a přitom značně omezují rychlost interpretace. Současná verze interpretu zatím nároky na rychlost splňuje, ovšem interpret zatím není zcela kompletní. Jak bude dostačovat rychlost po úplném dokončení zatím nelze odhadnout. Případně by bylo nutné zavést nějakou formu paralelního zpracování dopočtů.

5 ZÁVĚR A BUDOUCÍ VÝVOJ

Po dokončení interpretu bude velmi důležité kvalitní testování. Vhodné se jeví testování pomocí srovnávání s původní verzí interpretu. Tedy nechat oba interprety pracovat se stejnými dopočtovými rovnicemi nad stejným obsahem databáze a průběžně srovnávat stavy obou testovacích databází. Za tímto účelem bude nutné navrhnout automatizovaný testovací nástroj a test tohoto typu provést nejlépe pro všechny aktuálně využívané dopočty.

Poděkování: Tato práce vznikla částečně za podpory grantu VUT FIT, FIT-S-10-2 a specifického výzkumu MSM0021630528.

REFERENCE

[1] QNX Neutrino RTOS Overview [on-line], [cit. 2010-02-20], URL: <http://www.qnx.com/products/neutrino_rtos/>