Datové proudy

Komunikace mezi funkčními bloky

Technická zpráva - FIT - VG20102015006 - 2011 - 07

Ing. Jiří Zuzaňák

Faculty of Information Technology, Brno University of Technology

1.12.2011

Contents

Abstract

Tato zpráva pojednává o nástroji umožňujícím kombinování implementací řešících dílčí výpočetní úlohy do robustního řešení daného problému. Výhodou tohoto přístupu je možnost využití již existujících řešení pocházejících z různých zdrojů, bez potřeby detailní znalosti jejich implementace, a také bez omezení platformou, na které fungují. Popisované nástroje v sobě zahrnují postupy, kterými je možné jednotlivé funkční bloky (implementace dílčích řešení) zkombinovat na úrovni vstupních a výstupních dat jednotlivých algoritmů. K dosažení tohoto cíle je využito množství nástrojů pro určených pro mezi-procesovou komunikaci, ať už na úrovni operačního systému, nebo až na úrovni TCP/IP protokolu.

1 Motivace

Pro většinu problémů řešených v počítačové grafice, zpracování obrazu a počítačovém vidění existuje nepřeberné množství softwarových řešení, která jsou více či méně volně dostupná, ať už na celosvětové sítí, nebo z jiných zdrojů. Tyto řešení (funkční bloky) je možné získat v různých formách, může to být funkce zapsaná v určitém programovacím jazyce, binární knihovna použitelná jako součást vlastního programu, zdrojové kódy, nebo spustitelný binární soubor. Navíc je možné že jednotlivá použití hodná řešení kladou navzájem neslučitelné požadavky na systémové zdroje, kterými mohou být typ operačního systému, architektura hardware, instrukční sada procesoru, typ grafické karty a jiné.

Z důvodu snadné a rychlé implementace robustního řešení složitějších problémů je vhodné využít již existujících dostupných implementací, které se pomocí vhodného nástroje spojí do komplexního celku. Tato zpráva popisuje nástroj, který je konstruován za účelem snadného spojování dílčích řešení (funkčních bloků) do uceleného robustního řešení daného problému.

2 Návrh řešení

Jak bylo uvedeno v úvodním textu, předpokládáme že máme k dispozici funkční bloky, kde každý tento blok přijímá jako vstup data pevně daného typu a produkuje jako výstup také data pevně daného typu. Jiná data neakceptuje a pokud tyto nebudou splňovat požadavky funkčního bloku, tak tento blok pravděpodobně zhavaruje. Proto je nutné pro jednoduché předávání takových dat nadefinovat funkce, umožňující snadnou konverzi mezi jednotlivými datovými typy. Není nutné aby si základní datové typy jednotlivých implementačních jazyků odpovídaly.

Datové proudy pro spojování funkčních bloků se snaží řešit výše uvedené problémy pomocí rozhraní umožňujícího snadnou komunikaci mezi různými implementacemi funkčních bloků. Rozhraní se skládá z množiny konfiguračních skriptů popisujících prostředí, ve kterém se spouští jednotlivé funkční bloky. Konfigurační skripty popisují: datové typy které se používají pro komunikaci pomocí komunikačních kanálů, datové typy akceptované jednotlivými funkčními bloky a typy komunikačních kanálů, které je možné využít pro přenos dat mezi funkčními bloky.

3 Komunikační rozhraní

Jednotlivé funkční bloky jsou ve výsledném datovém toku reprezentovány spustitelným souborem, který se postará o zpracování parametrů příkazové řádky a navázání komunikace se vstupními a výstupními datovými kanály. Těmto blokům budeme říkat moduly z pohledu datového toku, nebo funkční jednotky z pohledu jejich funkce (operace, kterou provádějí nad vstupními daty).

Aby mohl systém generující datový tok snadno vytvořit takovéto moduly musí mít dostatečnou znalost o reprezentaci datových typů z pohledu funkčního

bloku a musí znát způsob, kterým takovéto data vytvořit z datových paketů, které používá pro komunikaci. Tyto informace jsou generujícímu systému zprostředkovány pomocí konfiguračních skriptů několika různých typů. Konfigurační skript je přiřazen každému programovacímu jazyku, datovému paketu a v neposlední řadě každé funkční jednotce, ze které se takto vytvoří modul datového toku.

Následuje seznam základních typů používaných konfiguračních skriptů:

- `DS_BASE` - popisuje základní proměnné prostředí, a definice všech tříd objektů vytvářených na základě konfiguračních skriptů.
- `DS_LANG` - definuje jazyk, který je možné použít jako implementační jazyk funkčního bloku. Popisuje funkce pro zpracování argumentů příkazové řádky (propojující komunikační kanály), funkce pro obsluhu signálů operačního systému a funkce sloužící k zaznamenávání informačního logu.
- `DS_CHANNEL` - definuje typ komunikačního kanálu, funkce pro otevření a zavření kanálu, a funkce pro rozpoznání, čtení a zápis datového paketu. Komunikačním kanálem může být například soubor operačního systému, sdílená paměť, TCP/IP protokol či jiné prostředky mezi-procesové komunikace. Funkce pro identifikaci, čtení a zápis datového paketu musí být definovány pro všechny programovací jazyky, pro které je komunikační kanál určen.
- `DS_DT` - definuje datový typ v rámci jednoho implementačního jazyka definovaného jako instance třídy `DS_LANG`. Popisuje strukturu přenášených dat ve vnitřní reprezentaci tohoto jazyka.
- `DS_PKT` - definuje datový paket (jednotku dat přenášenou přes komunikační kanál). Obsahuje funkce popisující konverzi z a do datových typů jazyka odpovídajících typu paketu pro všechny jazyky, pro které je paket nadefinován. Zajišťuje transformaci přečteného paketu do vnitřní reprezentace použitého jazyka a zpět. Je definován pro jeden či více jazyků.
- `DS_MODULE` - definuje jednu procesní jednotku (modul). Ten v implementaci datových proudů reprezentuje jeden funkční blok. Je definován pro jeden jazyk a slouží k zavedení informace umožňující začlenění funkčního bloku do datového toku. Obsahuje informace o použitém jazyce, požadovaném počtu vstupních a výstupních kanálů, očekávaných paketech na jednotlivých vstupních kanálech a generovaných paketech na výstupních kanálech modulu. Dále identifikuje zdrojové soubory funkční jednotky, které jsou v rámci generovaní datového proudu modifikovány na základě výše popsaných parametrů a zkompilovány na základě typu použitého jazyka.
- `DS_GENERATOR` - existuje pouze jedna instance této třídy, ta po spuštění zajistí prohledání celého prostoru sestavovaného proudu, nalezne instance jednotlivých výše uvedených tříd a na jejich základě vygeneruje upravené zdrojové soubory, ze kterých se sestaví výsledné moduly (funkční bloky).

Jednotlivé konfigurační skripty obsahují kód v jazyce `Python`, který vytváří objekt dané třídy, reprezentující jednu jednotku datového toku. Po slinkování do jednoho programu vznikne kompletní reprezentace datových modulů včetně

možností jejich vzájemného propojení. Program reprezentuje pouze pohled na funkční jednotky a jimi zpracovávaná data, samotný funkční kód je reprezentován spustitelnými binárními soubory, které nejsou součástí tohoto programu.

4 Začlenění funkčního bloku

V této části dokumentu je stručně popsán postup začlenění funkční jednotky do datového toku.

4.1 Funkční blok ve formě zdrojového kódu

Pokud je začleňovaný funkční blok ve formě zdrojových kódů je nutné pro integrování tohoto bloku do datového toku splnit následující podmínky.

1. Je nutné aby byl definován odpovídající jazyk (tj. instance třídy `DS_LANG`.), který zajistí správné zpracování argumentů příkazové řádky a následné otevření potřebných komunikačních kanálů pro daný modul.
2. Pro všechny vstupní a výstupní datové typy v reprezentaci zvoleného jazyka musí být nadefinovány instance třídy datový typ `DS_DT`.
3. Je podmínkou aby pro vstupní a výstupní datové typy funkčního bloku existoval vhodný typ paketu (instance třídy `DS_PKT`), který umožní jejich přenos pomocí komunikačního kanálu. Není nutné aby struktura paketu odpovídala struktuře datového typu, musí však existovat funkce, která zvládne tyto data zkonvertovat oběma směry.

Pokud jsou výše uvedené podmínky splněny stačí pro začlenění funkčního bloku vytvořit instanci třídy `DS_MODULE` a nadefinovat v ní podstatné informace o použitém jazyce, vstupních a výstupních kanálech a použitých vstupních a výstupních paketech. Dále je nutné upravit samotný zdrojový kód funkčního bloku, a to vložením řídících (`DS_INJECT`) sekvencí, mezi které se vloží kód vygenerovaný při sestavování datového toku viz sekce 6.

4.2 Funkční blok ve formě spustitelného souboru

Pro zaintegrování spustitelného binárního souboru do datového toku je nutné vytvořit tzv. obalový modul, který zajistí vhodnou konverzi vstupních a výstupních dat. Existuje velké množství způsobů, kterými může spustitelný program přijímat data pro zpracování. Může data číst z argumentů příkazové řádky, načítat je z předem daných souborů, může data číst ze standardního vstupu a jiné. Velká variabilita zůstává i u komunikačních možností, které může program využívat pro zápis výstupních dat. Program reprezentující obalový modul musí načítaná data přemístit na pozici, na které je očekává spuštěný obalený program a výstupní data programu opět zapojit do datového toku. Také se musí starat o správné ukončení programu, který se z tohoto důvodu často spouští jako jeho dětský proces.

Pro začlenění obalového programu do datového toku musí být splněny stejné podmínky jako pro začlenění funkčního bloku reprezentovaného zdrojovým kódem, viz 4.1.

5 Výsledný modul datového toku

Jak už bylo uvedeno výše, pro každý nadefinovaný modul vznikne jeden spustitelný binární program, nebo skript. Ten umí na základě předaných parametrů příkazové řádky otevřít komunikační kanály zadaného typu a z nich načítat či do nich zapisovat pakety definované v jeho konfiguračním skriptu.

Moduly akceptují následující seznam parametrů:

- `-i`, `--input` - nastaví jeden vstupní kanál. Vstupních kanálu může být nadefinováno neomezené množství. Za argumentem následuje identifikace typu a parametrů vstupního kanálu.
- `-o`, `--output` - nastaví jeden výstupní kanál. Obdobně jako vstupních může být výstupních kanálů nadefinováno neomezené množství. Za argumentem následuje identifikace typu a parametrů kanálu.
- `-h`, `--help` - vytiskne obecné informace o modulu a detailní informace popisující individuální vlastnosti modulu (ty jsou součástí konfiguračního skriptu modulu).
- `-u`, `--unrecognized` - nastaví pravidla pro manipulaci s nerozpoznanými typy paketů. Ty je možné kopírovat na cílový výstupní kanál, v závislosti na kterém kanále byly přečteny, nebo je ignorovat.

Pro snadnější zápis toků využívajících unixových rour se kanály reprezentované standardním vstupem a výstupem programu otevírají automaticky, za předpokladu že nebyl pomocí argumentů otevřen dostatečný počet kanálů. Díky této vlastnosti konstruovaných datových toků je snadné do nich zapojit standardní unixové příkazy.

Příklad datového toku, který uloží snímky načtené z videa do souboru komprimovaného pomocí programu `gzip`.

```
user$ ./load_video video.avi | gzip > stream.bin.gz
```

Příklad datového toku, který načte datový tok ze souboru a jednotlivé snímky v něm obsažené uloží do samostatných souborů.

```
user$ ./invert < stream.bin | ./save_image target png
```

Příklad datového toku, který načte video z první kamery nalezené v systému, to poté zaznamená do souboru a zároveň zobrazí na displeji.

```
user$ ./load_video | tee stream.dat | ./show_video -i file:↩
    stdio
```

Jednotlivé moduly datového toku jsou navrženy takovým způsobem, aby bylo možné popsat datový tok v jednoduchém grafickém nástroji (nadefinování modulů a jejich vzájemného propojení), a na základě takového popisu vygenerovat příkaz, či sekvenci příkazů, která by takto popsaný datový tok realizovala.

6 Příklad zdrojových dat

Následuje příklad zdrojového souboru v jazyce `c`, který obsahuje výše zmíněné řídící sekvence pro generování kódu datových toků. Vygenerovaný kód se vloží mezi dvojici formátovacích příkazů `DS_INJECT<>`, kde parametr prvního příkazu z této dvojice identifikuje generující instrukci. Řádky mezi těmito příkazy jsou ignorovány, čímž se zajistí možnost opětovného zpracování již zpracovaného souboru. Úroveň odsazení je odvozena od odsazení prvního příkazu `DS_INJECT<>`, což je podstatné v některých programovacích jazycích.

```
#include <stdio.h>

//DS_INJECT<code()>
//DS_INJECT<END>

int main(int argc,char **argv)
{
   //DS_INJECT<init("argc","argv")>
   //DS_INJECT<END>

   while (
         //DS_INJECT<test()>
         //DS_INJECT<END>
         ) {

      unsigned length = 0;
      char *data = NULL;

      //DS_INJECT<read(DS_DT_c_string,0)>
      //DS_INJECT<END>

      fwrite(data,sizeof(char),length,stdout);
      free(data);
   }

   //DS_INJECT<clear()>
   //DS_INJECT<END>

   return 0;
}
```

Výše uvedený kód popisuje triviální modul, který čte řetězce ze vstupního komunikačního kanálu a tyto zapisuje na standardní výstup programu. V kódu je použito následujících pět generujících příkazů:

- `DS_INJECT<code>` - generuje kód funkcí používaných při běhu funkčního bloku. Obsahuje zpracování parametrů, otevírání a zavírání komunikačních kanálů a obsluhu signálů operačního systému.
- `DS_INJECT<init("argc","argv")>` - generuje kód, který inicializuje datový modul a zpracuje parametry příkazového řádku, na základě kterých otevře požadované komunikační kanály. Parametry příkazové řádky jsou předány jako argumenty generujícího příkazu.
- `DS_INJECT<test()>` - vygeneruje podmínku testující zda má modul pokračovat ve své práci. Pokud je tato negativní (`False`) modul musí ukončit svou činnost.
- `DS_INJECT<read(DS_DT_c_string,0)>` - generuje kód pro čtení řetězce v jazyce `c` z prvního vstupního komunikačního kanálu. Požadavky na datový typ a komunikační kanál jsou předány jako parametry generujícího příkazu.
- `DS_INJECT<clear()>` - generuje kód, který uvolní všechny dynamicky alokované proměnné a v neposlední řadě uzavře všechny komunikační kanály.

Tyto generující příkazy představují jedinou úpravu, kterou projdou zdrojové kódy funkčního bloku, který je zamýšlený k integraci do datového toku.

6.1 Implementace

Základním implementačním jazykem (jazykem, ve kterém jsou zapsány jednotlivé konfigurační skripty) je jazyk `Python`. Další použité nástroje a jazyky jsou omezeny jen velikostí množiny konfiguračních skriptů popisujících implementační jazyky, kterých z tohoto důvodu může být neomezené množstcí.

7 Závěr

Velkou výhodou oproti jiným stramovacím nástrojům je více méně snadná integrace již existujících řešení a implementací funkčních bloků. Zatímco některé podobné projekty předpokládají že bude funkční blok vytvořen v jednom daném jazyce a základní šablona zdrojového kódu je poskytnuta jako součást projektu, uvedené řešení datových toků umísťuje svůj řídící kód ve vhodné formě do kódu již existujícího funkčního bloku.

7.1 Aktuální stav řešení

Jsou implementovány šablony popisující jednotlivé konfigurační skripty, včetně několika konfiguračních skriptů vytvořených za účelem testování funkce datových toků. Byl vytvořen generující nástroj, který na základě konfiguračních skriptů upravuje zdrojové kódy jednotlivých funkčních bloků. Aktuální stav řešení umožňuje vytvořit jednoduchý datový tok z několika popsaných modulů.

Pro reálné nasazení je nutné vytvořit konfigurační skripty popisující větší množství dostupných komunikačních kanálů, kterými mohou být například, roury operačního systému, sdílená paměť procesů, TCP/IP protokol, apod. Jejich absence omezuje možnosti komunikace mezi jednotlivými moduly datového toku.