(Stochastické) programování a C++

Martin Šmíd

seminář SPA, 8.3.2007

Osnova

Kompilovat nebo interpretovat?

- Interptace: Program je uložen jako sekvence příkazů vyššího programovacího jazyka, do strojového kódu se překládá až za běhu. (kdysi Basic, Matlab, Mathematica, GAMS, php)

Kompilovat nebo interpretovat?

- Interptace: Program je uložen jako sekvence příkazů vyššího programovacího jazyka, do strojového kódu se překládá až za běhu. (kdysi Basic, Matlab, Mathematica, GAMS, php)
- Kompilace: Program je z VPJ do SK přeložen jednou a je pak uložen ve formě příkazů SK (Turbo Pascal, C, C++)

Kompilovat nebo interpretovat?

- Interptace: Program je uložen jako sekvence příkazů vyššího programovacího jazyka, do strojového kódu se překládá až za běhu. (kdysi Basic, Matlab, Mathematica, GAMS, php)
- Kompilace: Program je z VPJ do SK přeložen jednou a je pak uložen ve formě příkazů SK (Turbo Pascal, C, C++)
- $\Longrightarrow$ Kompilace rychlejší, interpretace (někdy) úspornější na paměť, interpretovaný program může bžžet beze změny na více platformách, po změně se rychleji spustí

Kompilovat nebo interpretovat?

- Interptace: Program je uložen jako sekvence příkazů vyššího programovacího jazyka, do strojového kódu se překládá až za běhu. (kdysi Basic, Matlab, Mathematica, GAMS, php)
- Kompilace: Program je z VPJ do SK přeložen jednou a je pak uložen ve formě příkazů SK (Turbo Pascal, C, C++)
- $\Longrightarrow$ Kompilace rychlejší, interpretace (někdy) úspornější na paměť, interpretovaný program může bžžet beze změny na více platformách, po změně se rychleji spustí
- Různé mezistupně: Visual Basic (for Applications), C#, Java

Kompilovat nebo interpretovat?

- Interptace: Program je uložen jako sekvence příkazů vyššího programovacího jazyka, do strojového kódu se překládá až za běhu. (kdysi Basic, Matlab, Mathematica, GAMS, php)
- Kompilace: Program je z VPJ do SK přeložen jednou a je pak uložen ve formě příkazů SK (Turbo Pascal, C, C++)
- $\Longrightarrow$ Kompilace rychlejší, interpretace (někdy) úspornější na paměť, interpretovaný program může bžžet beze změny na více platformách, po změně se rychleji spustí
- Různé mezistupně: Visual Basic (for Applications), C#, Java
- S nepřenositelností se kompilované jazyky vyrovnávají pomocí zavedení standardů (např. ANSI norma pro C++)

$(1 + 2i) * (3 + 4i)$ v BASICu

```
1 GOTO 100
10 LET R3=R1*R2-I1*I2
20 LET I3=R1*I2+I1*R2
30 RETURN

100 LET R1=1
110 LET I1=2
100 LET R2=3
110 LET I2=4
120 GO SUB 10
130 PRINT R3
140 PRINT "+"
130 PRINT I3
140 PRINT "i"
```

$(1 + 2i) * (3 + 4i)$ - v Céčku

```
struct Complex { double re; double im;};

void mult(Complex* a, Complex* b, Complex* r) {
  r->re=a->re*b->re-a->im*b->im;
  r->im=a->re*b->im+a->im*b->re;
}

void main() {
  Complex a={1,2},b={3,4},r;

  mult(&a,&b,&r);
  printf("%f + %f i",b.re,b.im);
}
```

$(1 + 2i) * (3 + 4i)$ v C++

```
class Complex {
    double re; double im;
  public:
    Complex(double r, double i) : re(r), im(i) {}
    Complex operator*(Complex & c )
      { return Complex(re*c.re-im*c.im,
          re* c.im+im*c.re);}
    void display( )
      { cout << re << ", " << im << endl; }
};

void main() {
  Complex a(1,2),b(3,4),r;

  r = a * b;
  r.display();
}
```

Chvála C++

- Umožňuje vytvářet přehledné a robustní programy
- Program je částečně samodokumentující
- Podporuje týmovou práci
- Umožňuje ,,high-level" i ,,low-level" práci
- Výsledný spustitelný program je velmi efektivní
- ...

Kdy optimalizační software a C++?

- Chceme vytvořit uživatelsky přívětivý program pro osobní finance se spoustou grafů a jiných vymyšleností, mezi něž patří i výběr portfolia

Kdy optimalizační software a C++?

- Chceme vytvořit uživatelsky přívětivý program pro osobní finance se spoustou grafů a jiných vymyšleností, mezi něž patří i výběr portfolia
- Chceme počítat kvantitativní stabilitu optimalizační úlohy

Kdy optimalizační software a C++?

- Chceme vytvořit uživatelsky přívětivý program pro osobní finance se spoustou grafů a jiných vymyšleností, mezi něž patří i výběr portfolia
- Chceme počítat kvantitativní stabilitu optimalizační úlohy
- Počítáme složitou funkci, v jejíž definice si objevuje $\min_X(p)\mathbb{E}\ldots$ kde p je parametr.

Kdy optimalizační software a C++?

- Chceme vytvořit uživatelsky přívětivý program pro osobní finance se spoustou grafů a jiných vymyšleností, mezi něž patří i výběr portfolia
- Chceme počítat kvantitativní stabilitu optimalizační úlohy
- Počítáme složitou funkci, v jejíž definice si objevuje $\min_X(p)\mathbb{E}\ldots$ kde p je parametr.

(Dr. Popela: definice úloh vytváří v pomocí programu v Pascalu, který pak vytvoří formu čitelnou pro GAMS a spustí jej)

Projekt SMI

- *SMI: Stochastic Modeling Interface, for optimization under uncertainty* je součástí projektu COIN-OR (zde již podrobně zmiňovaného)

Projekt SMI

- *SMI: Stochastic Modeling Interface, for optimization under uncertainty* je součástí projektu COIN-OR (zde již podrobně zmiňovaného)
- Ze serveru lze stáhnout pouze zdrojové soubory

Projekt SMI

- *SMI: Stochastic Modeling Interface, for optimization under uncertainty* je součástí projektu COIN-OR (zde již podrobně zmiňovaného)
- Ze serveru lze stáhnout pouze zdrojové soubory
- Pokus o kompilaci ve Windows: Celé Coin-OR je určeno spíše pro UNIX, autoři slibují, že projekt SMI půjde kompilovat i pomocí MS Visual Studio (jehož zkušební verzi jsem několik hodin stahoval), ale nejde to, kompilátor nahlásí fatální chyby

Projekt SMI

- *SMI: Stochastic Modeling Interface, for optimization under uncertainty* je součástí projektu COIN-OR (zde již podrobně zmiňovaného)
- Ze serveru lze stáhnout pouze zdrojové soubory
- Pokus o kompilaci ve Windows: Celé Coin-OR je určeno spíše pro UNIX, autoři slibují, že projekt SMI půjde kompilovat i pomocí MS Visual Studio (jehož zkušební verzi jsem několik hodin stahoval), ale nejde to, kompilátor nahlásí fatální chyby
- SMI používá další projekty COIN-OR:
  - `OSI` - Open Solver Interface: knihovna, umožňující jednotně přistupovat k různým solverům
  - `CLP` - simplex solver
  - `BuildTools, CoinUtils, Doxydoc, MSVisualStudio, Data`

Třída SmiScnModel

```
class SmiScnModel {
...
public:
  int readSmps(const char *name, ...);
    // načte soubory <name>.core, <name>.time
    // a <name>.stoch
  void setOsiSolverHandle(OsiSolverInterface & osi);
    // nastaví solver, který se bude používat
  OsiSolverInterface * loadOsiSolverData();
    // vytvoří "deterministický ekvivalent" tak, že
    // vygeneruje scénáře (toto chování třídy lze změnit)
  initialSolve();
    // vyřeší úlohu
...
}
```

Příklad použití

```
SmiScnModel smi;
smi.readSmps(name);
OsiClpSolverInterface *clp = new OsiClpSolverInterface();
smi.setOsiSolverHandle(*clp);
OsiSolverInterface *osiStoch = smi.loadOsiSolverData();
osiStoch->setHintParam(OsiDoPresolveInInitial,true);
osiStoch->setHintParam(OsiDoScale,true);
osiStoch->setHintParam(OsiDoCrash,true);
osiStoch->initialSolve();
printf("Solved stochastic program % s\n", name);
printf("Number of rows: % d\n",osiStoch->getNumRows());
printf("Number of cols: % d\n",osiStoch->getNumCols());
printf("Optimal value: % g\n",osiStoch->getObjValue());
```

SMI: +/-

Plusy

- Možnost, jak používat SP (ale i LP, NLP atd) z jiných programů
- Je zadarmo

SMI: +/-

Plusy

- Možnost, jak používat SP (ale i LP, NLP atd) z jiných programů
- Je zadarmo

Mínusy

- Je složitý a nepřehledný
- Není dokumentace
- Nemá podporu