DISTRIBUOVANÉ VÝPOČTY V MATLABE

Martin Foltin

Fakulta elektrotechniky a informatiky,
Slovenská technická univerzita,
Ilkovičova 3, 831 02 Bratislava, Slovenská republika

1. Úvod

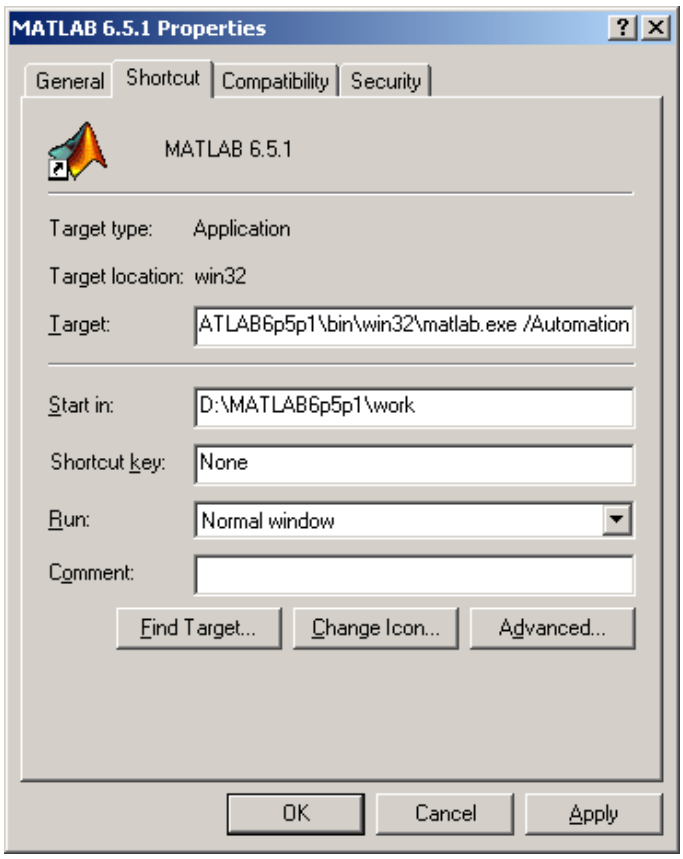
V predkladanom článku sa zaoberáme problematikou využitia technológie distribúcie zložitého výpočtu na viacero počítačov. Pre tento zámer sme si zvolili programový balík Matlab 6.5.1 určený pre operačný systém Windows. Voľba operačného systému je v tomto prípade dôležitá nakoľko budeme využívať technológiu DCOM, ktorá je charakteristická práve pre operačné systémy Windows. Rozdelením zložitého problému na niekoľko počítačov je možné významne znížiť čas výpočtu. Nebudeme sa však venovať rozdeľovaniu samotnej výpočtovej úlohy, ale ukážeme postup ako vytvoriť prostredie pre takýto spôsob výpočtov. Hlavným cieľom je zabezpečiť komunikačné spojenie medzi jednotlivými počítačmi jednak na fyzickej úrovni (počítačová sieť), ale aj na softvérovej úrovni (technológia DCOM). Technológiu DCOM je možné úspešne využívať aj v Matlabe v prípade, že je na počítači nainštalovaný operačný systém Windows 2000, resp. Windows XP.

2. Komunikačný kanál s využitím technológie DCOM

Na to aby sme mohli vôbec uvažovať o distribuovaných výpočtoch, musíme mať k dispozícii viacero počítačov. Jeden z nich, ktorý bude mať na starosti koordináciu ostatných, označme písmenom **A**. Ostatné označme písmenami **B**, **C**, **D** ... Predkladaným postupom môžeme vytvoriť komunikačný kanál medzi počítačmi **A** a **B**. Je zrejmé, že komunikácia medzi viacerými počítačmi by sa riešila analogicky. Na to aby sme mohli otvoriť komunikačný kanál medzi počítačmi **A** a **B** musia byť splnené tieto podmienky

- musia byť pripojené do počítačovej siete
- použitý OS musí byť na oboch počítačoch Windows 2000 alebo Windows XP
- musia byť v spoločnej doméne
- v prípade že nie sú v spoločnej doméne musia mať vytvorené kontá s rovnakým prihlasovacím menom a heslom
- na oboch musí byť nainštalovaný Matlab

V prípade, že chceme mať možnosť sledovať čo sa deje na počítači **B** počas vykonávania úloh, musíme spúšťať Matlab ako aplikáciu typu *Automation* (na počítači **B**). To zabezpečíme vytvorením ďalšieho zástupcu existujúcej ikony Matlabu. Následne naň klikneme pravým tlačidlom myši a v položke *Properties*, záložka *Shortcut*, položka *Target* pripíšeme na koniec /Automation (Obr. 1). Z praktických dôvodov je vhodné zmeniť aj ikonu, aby v budúcnosti nedochádzalo k mýleniu si oboch ikon.



Obr. 1 Nastavenie Matlabu na typ Automation

Takto nastavený Matlab spustíme na počítači **B** a maximalizujeme si jeho okno.

Na počítači **A** zadefinujeme a otvoríme komunikačný kanál príkazom actxserver. [1]

```
e1=actxserver('Matlab.Application','názov_počítača')
```

kde objekt e1 predstavuje komunikačný kanál.

Ako prvý parameter príkazu actxserver nastavíme *Matlab.Application* (s touto aplikáciou chceme komunikovať). Druhý parameter predstavuje názov počítača **B**. Tiež je možné namiesto mena počítača **B** zadať jeho IP adresu. Ak sme pred otvorením komunikačného kanála spustili na počítači **B** *Matlab /Automation*, tak komunikačný kanál komunikuje priamo s touto aplikáciou. Ak sme ho na **B** nespustili, automaticky sa spustí (je možné postrehnúť bežiacu aplikáciu Matlab v spustených procesoch), ale nebudeme mať možnosť sledovať dianie počas výpočtu priamo na počítači **B**. O správnej komunikácii a správnom zapnutí *Matlab /Automation* sa presvedčíme zadaním príkazu

```
e1.visible=1
```

a následne

```
e1.visible=0
```

na počítači **A**.

Matlab by sa mal na počítači **B** maximalizovať a následne minimalizovať. Ak všetko prebehlo bez problémov máme funkčný komunikačný kanál medzi počítačmi **A** a **B**. Z počítača **A** môžeme zadávať úlohy pre počítač **B** využitím príkazu Execute.

```
Execute(e1,'bench')
```

spustíme na B benchmark. Syntax príkazu Execute je :

Execute(*meno komunikačného kanála*, '*matlabovský príkaz*')

Ak budeme pozorne sledovať priebeh vykonávania *bench* testu na **B** tak si všimneme že počas behu sa počítač **A** zablokuje. Problém je v tom že počítač **A** čaká na odpoveď od **B**. Tá však príde až po vykonaní daného príkazu. Tento nedostatok je možné odstrániť pomocou objektu *timer*.

3. Definovanie objektu *timer* na počítači B

Na počítači **B** si zadefinujeme *timer*. Tento objekt bude spúšťať funkciu ktorú potrebujeme vykonať (zložitý výpočet ktorý je časovo náročný). *Timer* využívame na to aby funkcia bežala na pozadí a tým sa riadenie vráti späť počítaču **A** a nezablokuje sa. Dôležité je *timer* nastaviť s určitým oneskorením, aby sa riadenie stihlo vrátiť na počítač **A**. Ak by sme tak neurobili a StartDelay nezadefinovali (default=0) tak sa riadenie nestihne vrátiť a **A** sa zablokuje. *Timer* zadefinujeme nasledovným spôsobom :

```
t1=timer('StartDelay',1,'TimerFcn','mftimer')
```

Presný opis definície objektov typu timer je uvedený [2, 3] Ako TimerFcn sme zadefinovali funkciu **mftimer.m**. Táto funkcia musí byť samozrejme v aktuálnom pracovnom adresári počítača **B**. Príkazom start(t1) aktivujeme *timer* t1. Je vhodné súbor týchto príkazov napísať ako *m-file*, aby ich bolo možné jednoducho zavolať z počítača **A**. Je tiež dôležité zadefinovať globálne premenné ktoré budeme chcieť čítať v priebehu výpočtu z počítača **A**, kde sa môžu ďalej spracúvať. Keby sme tak neurobili, tak nemáme možnosť čítať žiadne dáta, nakoľko premenné v rámci funkcie sú zapúzdrené.

4. Spustenie úlohy z počítača A

Keď sme si na počítači **B** vytvorili *m-file* (v našom prípade **runtimer.m**) v ktorom sa definuje *timer* a tiež *funkciu* ktorá sa má vykonávať, môžeme pristúpiť k spusteniu úlohy z počítača **A**. Príkazom

```
Execute(e1,'runtimer')
```

zadaným na počítači **A**, spustíme výpočet na počítači **B**.

K premenným môžeme pristupovať príkazom GetFullMatrix. Presné použitie a syntax sa nachádza v manuáli [1]

```
X=GetFullMatrix(e1, 'X', 'base', zeros(2,2), zeros(2,2))
```

prečíta premennú X (musí byť globálna a musí mať rozmer 2x2) z počítača **B**. Takto teda môžeme sledovať stav výpočtu z počítača **A** na počítači **B**.

5. Výpis jednotlivých programov pre počítač B

zadefinovanie timera (**runtimer.m**)

```
%funkcia na vytvorenie a spustenie timeru
global X
t1=timer('StartDelay',1,'TimerFcn','mftimer')
disp('timer t1 inicializovany')
start(t1)
```

výpočtová funkcia (**mftimer.m**)

```
%funkcia v ktorej prebieha cast zloziteho vypoctu
function a=mftimer
global X
for i=1:1:300000   % tu moze byt lubovolna casovo narocna funkcia
   X=i*ones(2)
   pause(0);
end
a=i;
```

6. Záver

Uvedeným postupom je možné zapojiť do komplikovaného výpočtu viacero počítačov a tým skrátiť čas výpočtu. Tento postup bol použitý pri návrhu parametrov PSS pre jadrovú elektráreň Mochovce [4]. Návrh bol realizovaný pomocou genetických algoritmov, ktoré sú vhodné na riešenie tohto typu úloh.

7. Literatúra

[1] External Interfaces Reference Version 6, The MathWorks 2003
[2] Matlab Function Reference Version 6, The MathWorks 2003
[3] Using Matlab Version 6, The MathWorks 2003
[4] Sekaj I., Murgaš J., Foltin M., PSS Parameters Design Using Genetic Algorithms, 6th International Conference, Control of Power Systems ’04, June 16-18 2004, Štrbské Pleso High Tatras, Slovak Republic

kontakt
Tel : ++421 2 602 91 506
E-mail : foltin@kasr.elf.stuba.sk