TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing (ZIH)

Localization of Performance Problems with Gprof

Linux/x86 Performance Practical, 17.06.2009

Zellescher Weg 12

Willers-Bau A106

Tel. +49 351 - 463 - 31945

Ulf Markwardt (ulf.markwardt@tu-dresden.de)

Matthias Lieber (matthias.lieber@tu-dresden.de)

ZIH
Center for Information Services & High Performance Computing

Profiling

- Collect statistics of a program run
  - Timing of functions / source code lines
  - Number of function calls
  - Call graph
- Timings are collected by statistical sampling (100/1000 Hz rate)
  - Small run times may give inaccurate results
- Measurement influences run time
  - Impact on compiler optimization
  - Overhead due to interruption of normal program flow (cache, registers, pipes etc.)
  - Program will run slower with profiling

Gprof Usage

- Compile and Link complete program with **`-pg`** and optimization:
  - Consider turning off inlining to avoid loss of information

```
icc -pg -O3 -g -fno-inline -o myprog main.c utils.c
```

- Run as usual
  - Use small but representative input data
  - Profiling information is written to file **`gmon.out`**
- Extract the profile using the **`gprof`** command:

```
gprof myprog
```

- Gprof online manual: http://sourceware.org/binutils/docs/gprof/

Gprof Usage

```
mlieber@phobos:~/heat-c
mlieber@phobos:~/heat-c> icc -pg -O3 -g -fno-inline exampleC.c gettimec.c -o exampleC
mlieber@phobos:~/heat-c> ./exampleC
iteration steps:         20   dthetamax:         0.2981828027
   3.52502 s

= Energy Conservation Check =
 initial Energy:          1646938.000
   final Energy:          1646938.000
     Difference:               -0.000
mlieber@phobos:~/heat-c> ls -l
total 80
-rwxr-xr-x  1 mlieber zih 45724 2009-06-15 11:12 exampleC
-rw-r--r--  1 mlieber zih  8889 2009-06-15 11:12 exampleC.c
-rw-r--r--  1 mlieber zih  1196 2009-06-15 11:12 exampleC.h
-rw-r--r--  1 mlieber zih   834 2009-06-15 11:12 gettimec.c
-rw-r--r--  1 mlieber zih   197 2009-06-15 11:12 gettimec.h
-rw-r--r--  1 mlieber zih  4962 2009-06-15 11:13 gmon.out
mlieber@phobos:~/heat-c> gprof ./exampleC
```

Gprof Output: Flat Profile

mlieber@phobos:~/heat-c

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 62.31     22.32    22.32       20     1.12     1.12  heatTimestep
 25.71     31.53     9.21                             fmax.A
  3.74     32.87     1.34        1     1.34     1.34  heatAllocate
  2.51     33.77     0.90        1     0.90     0.90  heatInitialize
  2.15     34.54     0.77                             fmax
  1.31     35.01     0.47        2     0.24     0.24  gettimec
  1.20     35.44     0.43        2     0.22     0.22  heatTotalEnergy
  0.89     35.76     0.32                             fmax.L
  0.25     35.85     0.09       20     0.00     0.00  heatBoundary
  0.00     35.85     0.00        1     0.00     0.00  heatDeallocate

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
lines 1-35
```

`gprof ./exampleC`

10 times higher runtime is a known bug in older Linuxes

Gprof Output: Call Graph

mlieber@phobos:~/heat-c

```
index % time    self  children    called     name
                                                 <spontaneous>
[1]     71.3    0.00   25.55                 main [1]
               22.32    0.00      20/20          heatTimestep [2]
                1.34    0.00       1/1           heatAllocate [4]
                0.90    0.00       1/1           heatInitialize [5]
                0.47    0.00       2/2           gettimec [7]
                0.43    0.00       2/2           heatTotalEnergy [8]
                0.09    0.00      20/20          heatBoundary [10]
                0.00    0.00       1/1           heatDeallocate [11]
-----------------------------------------------
               22.32    0.00      20/20          main [1]
[2]     62.3   22.32    0.00      20          heatTimestep [2]
-----------------------------------------------
                                                 <spontaneous>
[3]     25.7    9.21    0.00                 fmax.A [3]
-----------------------------------------------
                1.34    0.00       1/1           main [1]
[4]      3.7    1.34    0.00       1          heatAllocate [4]
-----------------------------------------------
                0.90    0.00       1/1           main [1]
[5]      2.5    0.90    0.00       1          heatInitialize [5]
-----------------------------------------------
                                                 <spontaneous>
[6]      2.1    0.77    0.00                 fmax [6]
-----------------------------------------------
                0.47    0.00       2/2           main [1]
[7]      1.3    0.47    0.00       2          gettimec [7]
-----------------------------------------------
                0.43    0.00       2/2           main [1]
[8]      1.2    0.43    0.00       2          heatTotalEnergy [8]
-----------------------------------------------
                                                 <spontaneous>
[9]      0.9    0.32    0.00                 fmax.L [9]
-----------------------------------------------
lines 49-83
```

```
gprof ./exampleC
```

Gprof Output: Line-by-line Profile

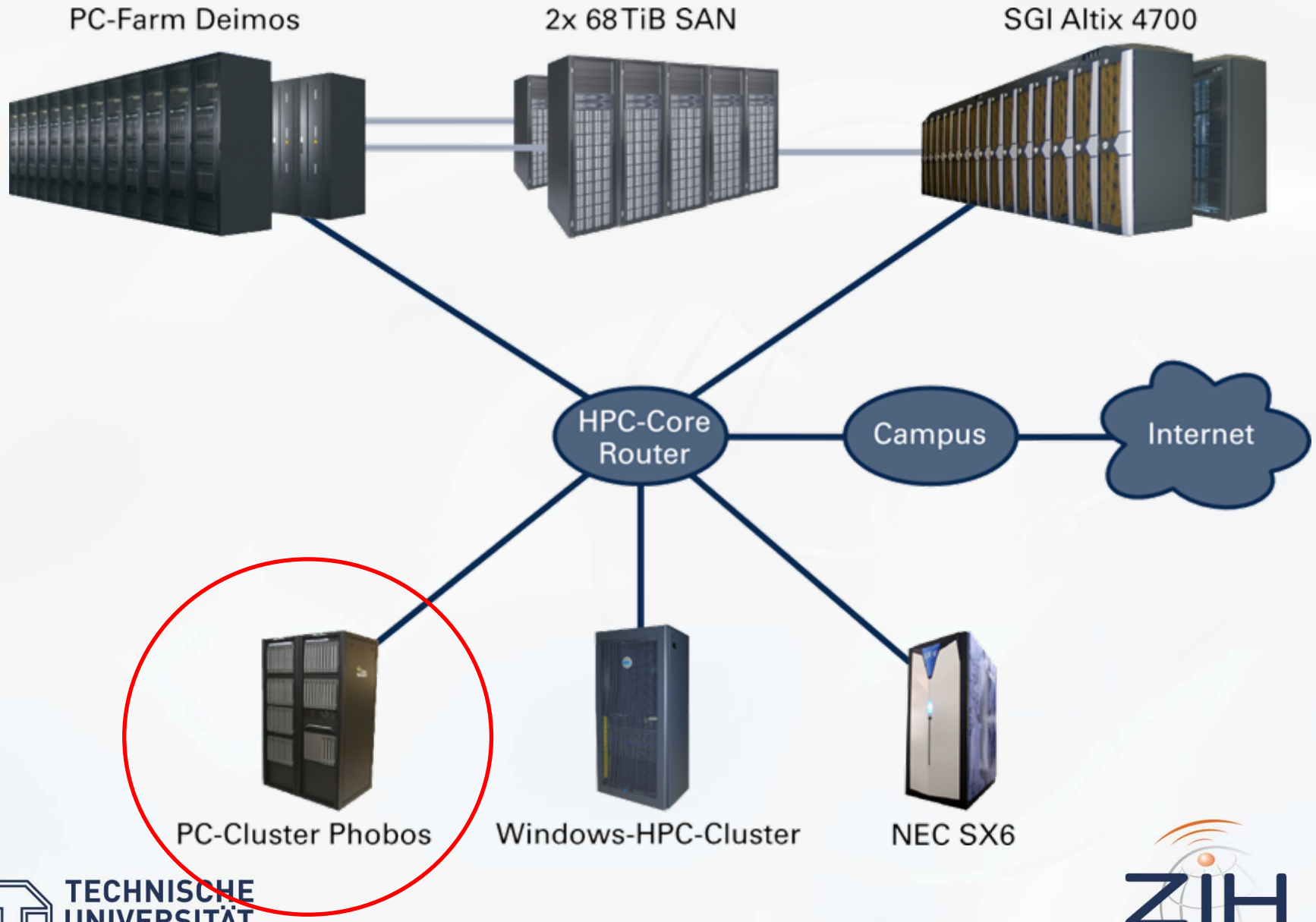
mlieber@phobos:~/heat-c

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
 27.97     10.02    10.02                             heatTimestep (exampleC.c:169 @ 4010aa)
 25.71     19.23     9.21                             fmax.A
 10.30     22.91     3.69                             heatTimestep (exampleC.c:156 @ 40103c)
  4.95     24.69     1.77                             heatTimestep (exampleC.c:154 @ 400fec)
  2.79     25.69     1.00                             heatTimestep (exampleC.c:160 @ 40104b)
  2.15     26.46     0.77                             fmax
  1.87     27.13     0.67                             heatAllocate (exampleC.c:38 @ 4012da)
  1.84     27.79     0.66                             heatTimestep (exampleC.c:168 @ 4010c2)
  1.70     28.40     0.61                             heatTimestep (exampleC.c:152 @ 40105c)
  1.62     28.98     0.58                             heatAllocate (exampleC.c:37 @ 4012cb)
  1.54     29.53     0.55                             heatTimestep (exampleC.c:154 @ 400fc9)
  1.45     30.05     0.52                             heatTimestep (exampleC.c:160 @ 401035)
  1.36     30.54     0.49                             heatTimestep (exampleC.c:155 @ 401025)
  1.31     31.01     0.47                             gettimec (gettimec.c:9 @ 4018c1)
  1.23     31.45     0.44                             heatTimestep (exampleC.c:155 @ 401002)
  1.23     31.89     0.44                             heatTimestep (exampleC.c:154 @ 40101d)
  1.23     32.33     0.44                             heatTimestep (exampleC.c:168 @ 40109d)
  1.20     32.76     0.43                             heatInitialize (exampleC.c:81 @ 401791)
  1.16     33.17     0.42                             heatTimestep (exampleC.c:156 @ 401031)
  1.01     33.53     0.36                             heatInitialize (exampleC.c:71 @ 4016c4)
  0.99     33.89     0.36                             heatTimestep (exampleC.c:156 @ 401010)
  0.89     34.21     0.32                             fmax.L
  0.89     34.53     0.32                             heatTimestep (exampleC.c:155 @ 400fd5)
  0.84     34.83     0.30                             heatTotalEnergy (exampleC.c:248 @ 401156)
  0.60     35.04     0.22                             heatTimestep (exampleC.c:154 @ 400fde)
  0.57     35.25     0.21                             heatTimestep (exampleC.c:155 @ 400fe8)
  0.47     35.42     0.17                             heatTimestep (exampleC.c:155 @ 400fc1)
  0.22     35.50     0.08                             heatBoundary (exampleC.c:196 @ 400ec9)
  0.22     35.58     0.08                             heatTotalEnergy (exampleC.c:247 @ 401149)
  0.15     35.64     0.06                             heatInitialize (exampleC.c:82 @ 4017a5)
lines 1-35/223 20%
```

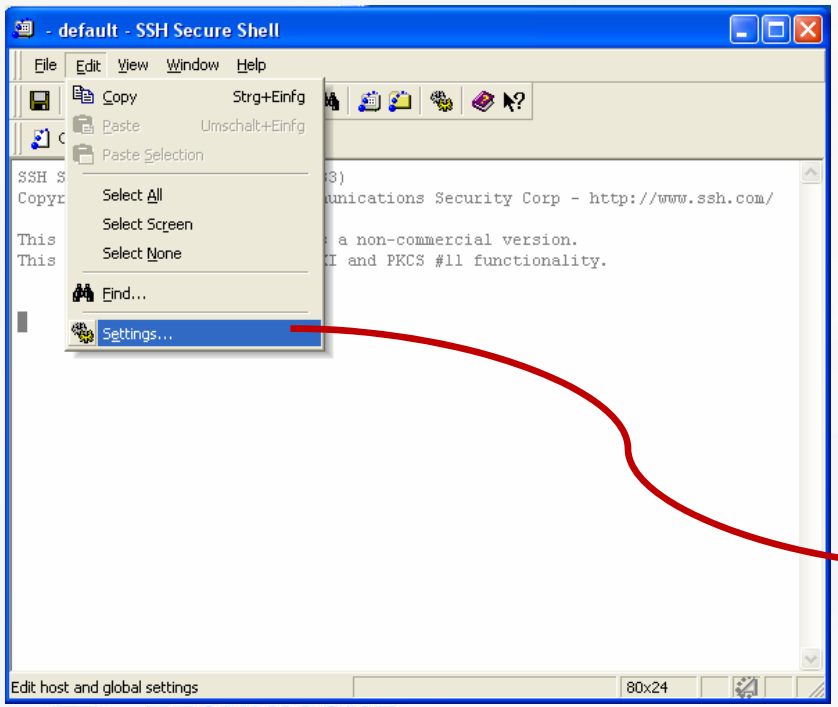
Compile with **`-g -pg`**
`gprof -l ./exampleC`

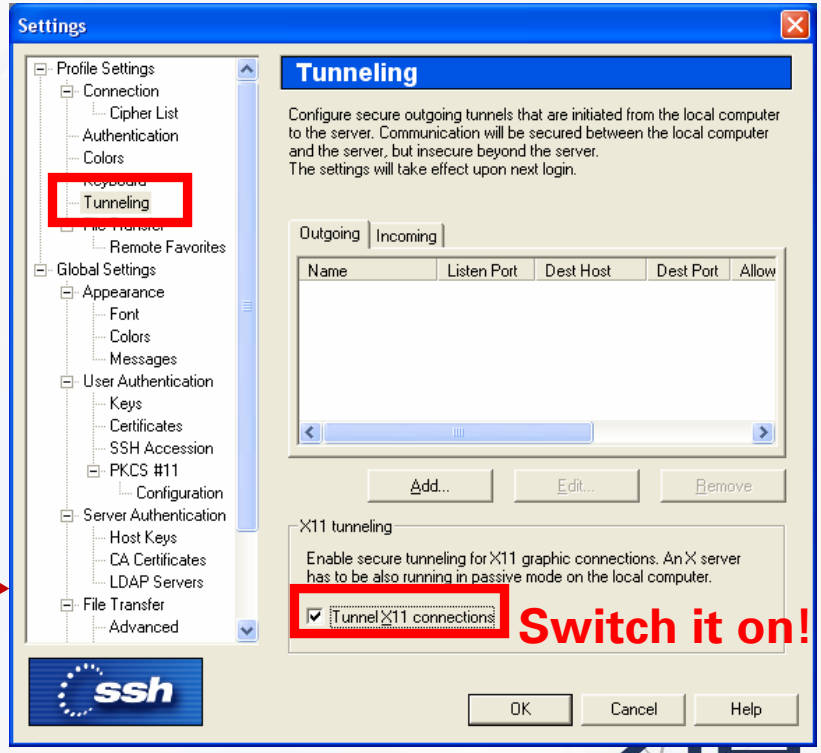
Exercise on Phobos



Login to Phobos

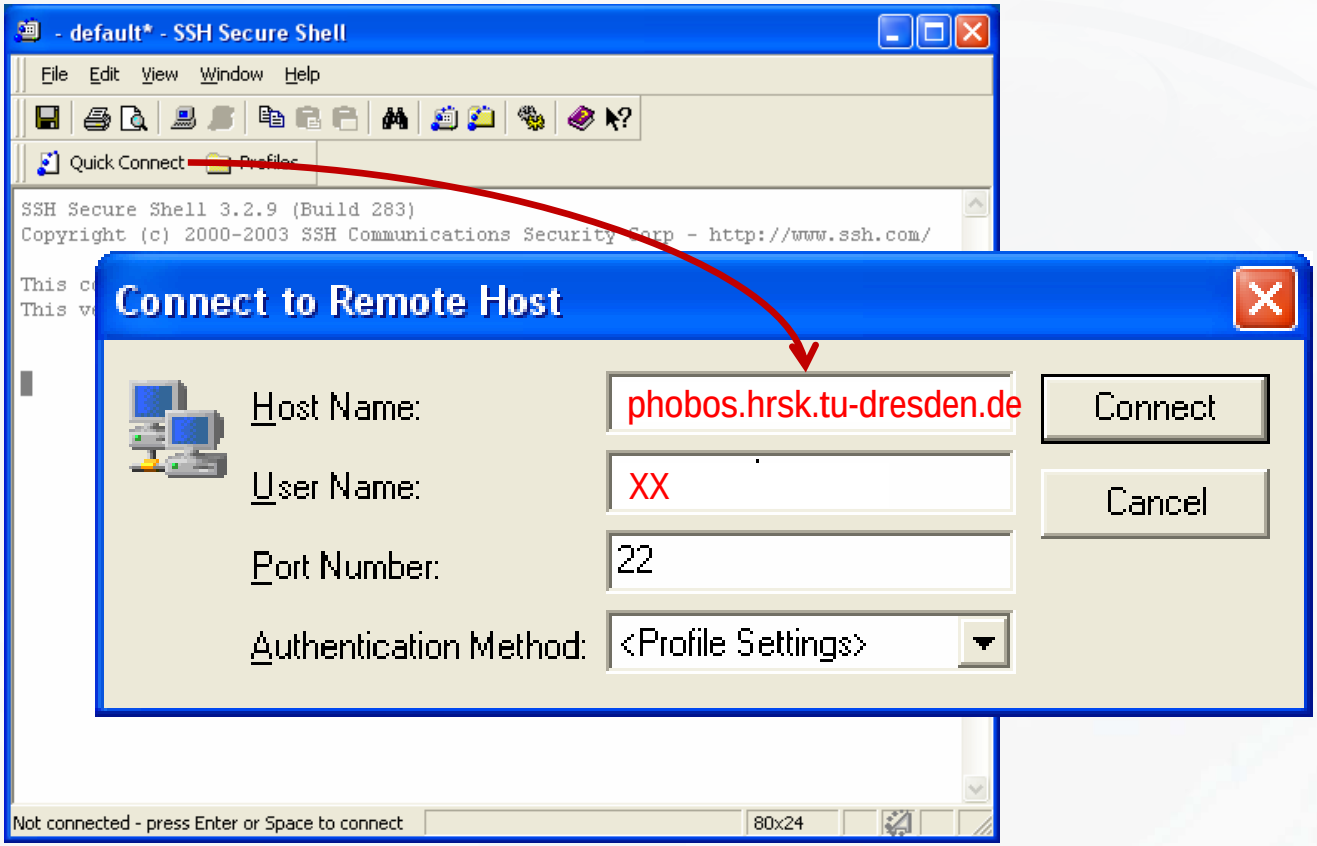
- Login into your PC with
- Open a SSH connection to phobos.hrsk.tu-dresden.de
- Start -> Programme -> SSH Secure Shell -> Secure Shell Client
- Enable X11 forwarding:





Login to Phobos

- Start X Server: Start -> Programme -> XWin32 -> XWin32
- Connect to phobos:



First Steps after Logging in

- All necessary modules are already loaded via .bash_login
- If you need an editor use: vi, mcedit, nedit, kwrite, or kate
- Start an interactive batch session
  - **`bsub -U perfprac#0 -n 1 -Is -W 8:00 bash`**

- We will use GNU and Intel Compilers
  - Intel (version 11): **`icc, ifort`**
  - GNU (version 3.3): **`gcc`**
  - GNU (version 4.3): **`gcc-4.3, gfortran-4.3`**

Exercise

- Profile a small scientific application
  - C: **`cd heat-c`**
  - Fortran90: **`cd heat-f90`**
- Use the Makefile (edit **`CFLAGS`** / **`FFLAGS`** first) or call compiler at the shell
- Required flags for profiling: **`-pg -g`**
- Turn on basic optimization: **`-O2`**
- C example requires **`-std=c99`** and **`-lm`** when compiled with GNU
- Call Gprof after program run: **`gprof exampleC`** / **`gprof exampleF`**
- Try GNU vs. Intel compiler
- Try with and without inlining: **`-finline`** / **`-fno-inline`**
- Find the most expensive lines of code: **`gprof -l`**