Usando PIN framework instrumentation no SPEC 2006

Paulo H. Ribeiro[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Campinas – SP – Brazil

paulohsilvar@gmail.com

Abstract. *This report documentation is to describe the Project1 of MO601 discipline UNICAMP. The work consists of four stages, the first two are to install and understand the SPEC 2006 tools and PIN framework instrumentation, the third step aims to use a pintool already established PIN and run it on all SPEC benchmarks. The last step calls for the establishment a new pintool, and run it at least on five benchmarks choice. The results need to be stored and displayed.*

Resumo. *Esta documentação de reporte vem para descrever o projeto1 da disciplina MO601 do IC da UNICAMP. O trabalho consiste em quatro etapas, as duas primeiras são para instalar e compreender as ferramentas SPEC 2006 e PIN framework instrumentation, a terceira etapa visa usar uma pintool já estabelecida do PIN e executar sobre todos os benchmarks do SPEC, a última etapa solicita a criação de uma nova pintool para o PIN e que ele seja executada em pelo menos cinco benchmarks a escolha. Os resultados precisam ser guardados e apresentados.*

1. Introdução

Uma ferramenta utilizada será o SPEC 2006. Ela visa fornecer um conjunto de padrões sob benchmarks relevantes para sistemas computacionais. A partir dela podemos avaliar o desempenho dos sistemas que buscam oferecer o mesmo serviço, tentando assim realizar uma comparação entre eles. Para realizar essas avaliações, o SPEC utiliza de diferentes benchmarks, que são programas em diferentes linguagens, com diferentes propósitos, tipos de entradas variados.

O PIN é a segunda ferramenta que utilizamos no projeto. Ela consiste de um framework para instrumentação, já vem disponibilizado várias tools desenvolvidas (conhecidas por pintools), um exemplo deles é a inscount, que tem como objetivo contar o número de instrução de um programa.

A infra-estrutura utilizada, foi um notebook, com sistema operacional linux ubuntu 12.04, processador intel i3, 32 bits e 4 núcleos.

O projeto1 tem como objetivo utilizar as duas ferramentas em conjunto. Inicialmente usando a pintool inscount (capaz de contar o número de instrução de um programa) para contar o número de instrução de todos os benchmarks disponibilizados pela SPEC 2006 e seus números de execuções. Para um segundo momento, deve-se desenvolver uma nova pintool e executá-la sob cinco benchmarks.

2. Pintool e SPEC 2006

Quando o PIN é instalado, já vem um conjunto de pintools programadas, no formato .cpp, para usar alguma é preciso gerar o executável delas. Para a pintool a ser utilizada nesse projeto, mais de uma versão são descrita, a de escolha foi a inscount2.cpp. Para gerar o executável da pintool, o PIN disponibiliza um arquivo de Makefile para esse função, a imagem a seguir apresenta como fazê-lo.

```
paulo@Carla:~/Documentos/PauloH/pin-3.0-76991-gcc-linux/source/tools/ManualExamples$ make obj-ia32/inscount2.so
g++ -DBIGARRAY_MULTIPLIER=1 -Wall -Werror -Wno-unknown-pragmas -D__PIN__=1 -DPIN_CRT=1 -fno-stack-protector -fno-exceptions -funwind-tables -fas
ynchronous-unwind-tables -fno-rtti -DTARGET_IA32 -DHOST_IA32 -DTARGET_LINUX -fabi-version=2  -I../../../source/include/pin -I../../../source/inc
lude/pin/gen -isystem /home/paulo/Documentos/PauloH/pin-3.0-76991-gcc-linux/extras/stlport/include -isystem /home/paulo/Documentos/PauloH/pin-3.
0-76991-gcc-linux/extras/libstdc++/include -isystem /home/paulo/Documentos/PauloH/pin-3.0-76991-gcc-linux/extras/crt/include -isystem /home/paul
o/Documentos/PauloH/pin-3.0-76991-gcc-linux/extras/crt/include/arch-x86 -isystem /home/paulo/Documentos/PauloH/pin-3.0-76991-gcc-linux/extras/cr
t/include/kernel/uapi -isystem /home/paulo/Documentos/PauloH/pin-3.0-76991-gcc-linux/extras/crt/include/kernel/uapi/asm-x86 -I../../../extras/co
mponents/include -I../../../extras/xed-ia32/include -I../../../source/tools/InstLib -O3 -fomit-frame-pointer -fno-strict-aliasing   -c -o obj-ia
32/inscount2.o inscount2.cpp
g++ -shared -Wl,--hash-style=sysv ../../../ia32/runtime/pincrt/crtbeginS.o -Wl,-Bsymbolic -Wl,--version-script=../../../source/include/pin/pinto
ol.ver -fabi-version=2    -o obj-ia32/inscount2.so obj-ia32/inscount2.o  -L../../../ia32/runtime/pincrt -L../../../ia32/lib -L../../../ia32/lib-
ext -L../../../extras/xed-ia32/lib -lpin -lxed ../../../ia32/runtime/pincrt/crtendS.o -lpin3dwarf  -ldl-dynamic -nostdlib -lstlport-dynamic -lm-
dynamic -lc-dynamic
paulo@Carla:~/Documentos/PauloH/pin-3.0-76991-gcc-linux/source/tools/ManualExamples$
```

Figure 1. Gerando executável da pintool inscount2.cpp

Para executar a pintool basta chamar o comando pin que se encontra no diretório raiz de onde o PIN foi instalado, passando a pintool que se deseja usar e o programa. Um arquivo com o resultado da instrumentação e gerado com a extensão .out, conforme o exemplo a seguir:

```
paulo@Carla:~/Documentos/PauloH/pin-3.0-76991-gcc-linux/source/tools/ManualExamples/obj-ia32$ ls inscount*.so
inscount0.so  inscount2.so
paulo@Carla:~/Documentos/PauloH/pin-3.0-76991-gcc-linux/source/tools/ManualExamples/obj-ia32$ ../../../../pin -t inscount2.so -- /bin/ls
inscount0.o  inscount0.so  inscount2.o  inscount2.so  inscount.out  mediainst.o  mediainst.so  pin.log
paulo@Carla:~/Documentos/PauloH/pin-3.0-76991-gcc-linux/source/tools/ManualExamples/obj-ia32$ cat inscount.out
Count 511936
paulo@Carla:~/Documentos/PauloH/pin-3.0-76991-gcc-linux/source/tools/ManualExamples/obj-ia32$
```

Figure 2. Exemplo de uso do PIN

A SPEC 2006 é a outra ferramenta que estamos utilizando, após sua instalação para usá-la é necessário carregar o seu kit, através do comando a seguir no diretório raiz onde foi realizada a instalação:

```
paulo@paulo:~/programs$ . ./shrc
paulo@paulo:~/programs$
```

Figure 3. Carregando kit SPEC 2006

Após carregarmos o kit do SPEC 2006, podemos utilizar o comando runspec, que é capaz de executar os benchmarks. O uso dele corresponde de chamá-lo passando os parâmetros de execução e a lista de benchmarks (que pode ser de apenas um até todos) que serão executados. Como temos um conjunto grande dos parâmetros, existe a possibilidade de passarmos um arquivo de configuração para o runspec contendo nele todo o ajuste que queremos usar e também a configuração de compilação desejada.

```
28 ############################################################################
29 # ia32 (32-bit) gcc 4.3, 4.4 and 4.6 config file
30 # Sample Config file for CPU2006
31 ############################################################################
32
33 tune          = base
34 basepeak      = yes
35 # Select and/or modify as needed
36 #ext           = gcc43-32bit
37 #ext           = gcc44-32bit
38 ext           = gcc46-32bit
39 output_format = asc,csv,html
40 flagsurl0     = $[top]/config/flags/Example-gcc4x-flags-revA.xml
41 flagsurl1     = $[top]/config/flags/Example-linux-platform-revA.xml
42
43 hw_avail      = Dec-9999
44 license_num   = 0
45
46 submit = /home/paulo/Documentos/PauloH/pin-3.0-76991-gcc-linux/pin -t /home/paulo/Documentos/PauloH/pin-3.0-76991-gcc-linux/inscount2.so -
   o /home/paulo/programas/count_numbers_fp/${benchmark}_${workload}.out -- $command
47
48 use_submit_for_speed = 1
```

Figura 4. Parâmetros definidos pelo arquivo de configuração

```
70 default=default=default=default:
71 ############################################################################
72 #
73 # Compiler selection
74 #
75 ############################################################################
76 # NOTE: The path may be different if you use the compiler from
77 #       the gnu site.
78 CC                  = /usr/bin/gcc
79 CXX                 = /usr/bin/g++
80 FC                  = /usr/bin/gfortran
```

Figura 5. Setando compiladores pelo arquivo de configuração

```
156 __MD5__
157 400.perlbench=base=gcc46-32bit=default:
158 # Last updated Thu Sep 15 00:48:11 2016
159 optmd5=8c4687abad10f0fe52588da97420afb8
160 baggage=
161 compile_options=\
162 @eNqtUU9PgzAUv/dTNL1X4ryRsQRK3VCghEGiXghDWKrQmgImfnvL2BjOqBebJn197/X9/jSUAjf5\
163 a1nxuoTyreNStCZoO8WLLlO9eOYqey8Vrz4sdI0AYUFkQqNvlbHjwtgXBcR6y9PTKwmxu40oyUiU\
164 6jB0qZOudRDR2M8IiymEELMFxJWQeITBec3zlos9xM3NAh7XeUzme2H6cMovW9mrolwBYkJCLDTn\
165 gg5J5tyxKLHQF2IIaN562K1vr7e69p3kocOhIdlkU9PEGgE2TI4SL/CeqC79JuGIxeLEdjzfSx7n\
166 eKMaBPRxf+Hk3878YMpS7l7KomtXQ0vdTBZO6gdffPfSrP+WpEECO9noqzO4VzdoxGXp8Buzr/gE\
167 XAu1wQ==
168 exemd5=e7be182acdb6388dccf40c51da22d529
```

Figura 6. MD5 gerado de um benchmark

Quando um benchmark é executado, o arquivo de configuração que foi usado, recebe no seu final uma codificação de MD5.

Através do submit passado no arquivo, podemos alterar qual o comando de execução será usado no benchmark. Como o objetivo desse projeto é usar a ferramenta PIN para calcular o número de instruções de cada benchmark, passamos o comando do PIN por meio dessa variável, assim seu comando será chamado. A variável use_submit_for_speed=1 também precisa ser definida, pois por default, o runspec já tem qual comando ele deve executar para os benchmarks, usando essa configuração, informa-se para ele que se deseja utilizar outro comando, o que se encontra no submit.

Vamos apresentar um exemplo de como usar o comando runspec passando o arquivo de configuração e chamando a classe de benchmark de ponto flutuante.

```
paulo@Carla:~/programas$ runspec --config=paulo_gcc_fp.cfg --size=ref --noreportable --iterations=1 --action=build fp
runspec v6674 - Copyright 1999-2011 Standard Performance Evaluation Corporation
Using 'linux-suse10-ia32' tools
Reading MANIFEST... 22481 files
Loading runspec modules.................
Locating benchmarks...found 31 benchmarks in 6 benchsets.
Reading config file '/home/paulo/programas/config/paulo_gcc_fp.cfg'
Running "specperl /home/paulo/programas/Docs/sysinfo" to gather system information.
Retrieving flags file (/home/paulo/programas/config/flags/Example-gcc4x-flags-revA.xml)...
Retrieving flags file (/home/paulo/programas/config/flags/Example-linux-platform-revA.xml)...
Benchmarks selected: 410.bwaves, 416.gamess, 433.milc, 434.zeusmp, 435.gromacs, 436.cactusADM, 437.leslie3d, 444.namd, 447.dealII, 450.soplex, 453.povray, 454.calculix, 459.GemsFDTD, 465.tonto, 470.lbm, 481.wrf, 482.sphinx3, 998.specrand
Compiling Binaries
  Up to date 410.bwaves base gcc46-32bit default
  Up to date 416.gamess base gcc46-32bit default
  Up to date 433.milc base gcc46-32bit default
```

Figura 7. Uso do comando runspec

Para esse projeto foi utilizado o conjunto de entradas para cada benchmark contido no diretório “ref” de cada um, por isso, utilizado o parâmetro –size=ref.

3. Resultados do inscount

A pintool utilizada inscount tem como funcionalidade contar o número de instruções de um programa, conforme mencionamos anteriormente. O comando de runspec apresentado na secção anterior, foi rodado para todos os benchmarks da classe inteiros e pontos flutuantes, salvando os seus resultados em diretórios diferentes para melhor gerenciamento, como exemplo, podemos ver pela imagem a seguir, o resultado da execução dos benchmarks da classe inteiros e o conteúdo do arquivo de saída.

```
paulo@Carla:~/programas$ cd count_numbers
paulo@Carla:~/programas/count_numbers$ ls
400.perlbench_0.out  401.bzip2_5.out  403.gcc_7.out     445.gobmk_4.out       471.omnetpp_0.out
400.perlbench_1.out  403.gcc_0.out    403.gcc_8.out     456.hmmer_0.out       473.astar_0.out
400.perlbench_2.out  403.gcc_1.out    410.bwaves_0.out  456.hmmer_1.out       473.astar_1.out
401.bzip2_0.out      403.gcc_2.out    429.mcf_0.out     458.sjeng_0.out       483.xalancbmk_0.out
401.bzip2_1.out      403.gcc_3.out    445.gobmk_0.out   462.libquantum_0.out  999.specrand_0.out
401.bzip2_2.out      403.gcc_4.out    445.gobmk_1.out   464.h264ref_0.out
401.bzip2_3.out      403.gcc_5.out    445.gobmk_2.out   464.h264ref_1.out
401.bzip2_4.out      403.gcc_6.out    445.gobmk_3.out   464.h264ref_2.out
paulo@Carla:~/programas/count_numbers$ cat 400.perlbench_0.out
Count 1096503046051
paulo@Carla:~/programas/count_numbers$
```

Figure 8. Resultado do inscount

Como podemos ver, todos os benchmarks da classe inteira foram executados e estão apresentados de acordo com o seu número de execuções, cada arquivo de resultado apresenta o número de instrução da execução determinada do seu benchmark. No diretório /src do repositório, todos os arquivos de resultados estão disponibilizados.

4. Criação de uma nova pintool

A ferramenta PIN disponibliza uma API que permite a criação de novas pintools, possibilitando assim aumentar o conjunto de ferramentas de instrumentação ou se preciso, ajustar alguma já existente para determinada necessidade. Dando continuidade ao projeto 1, uma nova pintool foi criada, com o objetivo de calcular o tempo médio de cada instrução por segundo, o nome para ela é **mediainst**.

Para realizar sua tarefa, a pintool conta o tempo que o programa executou e o número de instruções que ele teve, realiza em seguida o calculo da medida de tempo médio de instrução na unidade de segundo.

Para a nova pintool, também há a necessidade de se gerar seu executável, apresentamos isso na imagem a seguir:

```
paulo@Carla:~/Documentos/PauloH/pin-3.0-76991-gcc-linux/source/tools/ManualExamples/obj-ia32$ ls
inscount0.o  inscount0.so  inscount2.o  inscount2.so  inscount.out  mediainst.o  mediainst.so  pin.log
paulo@Carla:~/Documentos/PauloH/pin-3.0-76991-gcc-linux/source/tools/ManualExamples/obj-ia32$
```

Figura 9. Executável do mediainst pintool

5. Resultado do mediainst

Foi utilizado a nova pintool mediainst para executar sobre 5 diferentes benchmarks do SPEC 2006, um novo arquivo de configuração foi criado, para apontar a nova pintool diferentemente do que foi realizado nas secções anteriores que apontavam para a

inscount. Os resultado foram salvos em um diretório a parte, onde o nome dos arquivos consiste do nome do benchmark rodado e o número de sua execução. O resultado pode ser visto na figura a seguir, como também toda a base de dados também se encontra no diretório /src no repositório.

```
paulo@Carla:~/programas$ cd count_media/
paulo@Carla:~/programas/count_media$ ls
400.perlbench_0.out  401.bzip2_1.out  401.bzip2_5.out  403.gcc_3.out  403.gcc_7.out     450.soplex_1.out
400.perlbench_1.out  401.bzip2_2.out  403.gcc_0.out    403.gcc_4.out  403.gcc_8.out
400.perlbench_2.out  401.bzip2_3.out  403.gcc_1.out    403.gcc_5.out  434.zeusmp_0.out
401.bzip2_0.out      401.bzip2_4.out  403.gcc_2.out    403.gcc_6.out  450.soplex_0.out
paulo@Carla:~/programas/count_media$ cat 403.gcc_1.out
Mediainst 1.32851e-09
paulo@Carla:~/programas/count_media$
```

Figure 10. Resultado do mediainst

6. References

Pin documentation,
https://software.intel.com/sites/landingpage/pintool/docs/67254/Pin/html/

Pin tutorial,

http://www.cs.du.edu/~dconnors/courses/comp3361/notes/PinTutorial

SPEC 2006 documentation,
https://www.spec.org/cpu2006/docs/