FDPR-Pro - Usage Feedback Directed Program Restructuring

July 2005

Vadim Eisenberg (vadimi@il.ibm.com)

Moshe Klausner (klausner@il.ibm.com)
Agenda

- Objectives
- FDPR-Pro operation (black box)
- Working with FDPR-Pro
  - Instrumentation
  - Profile collection
  - Optimization
- Working with FDPR
- Additional Info
- Homework
Objectives of the presentation

- Become familiar with:
  - FDPR-Pro operation (black box) – the user view
  - FDPR-Pro main command line options
  - FDPR – the FDPR-Pro wrapper
FDPR-Pro operation

1. Instrumentation

Profile

2. Running the instrumented profile collection

3. Optimization
FDPR-Pro operation

1. Instrumentation

Input executable

Profile

3. Optimization

2. Running the instrumented profile collection
FDPR-Pro operation

1. Instrumentation

Input executable

Profile

2. Running the instrumented profile collection

3. Optimization
FDPR-Pro operation

1. Instrumentation

Input executable

Instrumented executable

Profile

2. Running the instrumented profile collection

3. Optimization
FDPR-Pro operation

1. Instrumentation
   → Instrumented executable
   → Profile
   → 2. Running the instrumented profile collection

Input executable

3. Optimization
FDPR-Pro operation

1. Instrumentation

Input executable

Instrumented executable

Profile

2. Running the instrumented profile collection

3. Optimization
FDPR-Pro operation

1. Instrumentation

Input executable

Instrumented executable

Profile

2. Running the instrumented profile collection

Optimized executable

3. Optimization
Running FDPR-Pro from command line – typical example
Running FDPR-Pro from command line – typical example

> fdprpro –a instr myexe –f myexe.prof –o myexe.instr
> myexe.instr
> fdprpro –a opt myexe –f myexe.prof –o myexe.fdpr
FDPR-Pro options format

- Unix style
- All options have a short and long version. For example, `-a` and `--action`
- The short one starts with one ‘-’ sign, the long one starts with two ‘-’ signs
- Some options have subparameters, such as `-f input.prof` and some have no subparameters, such as `--killed-regs`
- The options and input file can come in any order
FDPR-Pro actions
FDPR-Pro actions

- Defined with the –a option
  - -a instr – instrumentation
  - -a opt – optimization
  - -a anl – analysis
  - -a check_sign – check signature of the fdprpro
- The -a option is mandatory
Running FDPR-Pro: instrumentation
Running FDPR-Pro: instrumentation

- `fdprpro myexe -a instr -f {pwd}/myexe.prof -o myexe.instr`
Running FDPR-Pro: instrumentation

```bash
fdprpro myexe -a instr -f {pwd}/myexe.prof -o myexe.instr
```

The input file
Running FDPR-Pro: instrumentation

diamond fdprpro myexe -a instr –f {pwd}/myexe.prof -o myexe.instr
Running FDPR-Pro: instrumentation

- fdprpro myexe -a instr –f {pwd}/myexe.prof -o myexe.instr

- The input file

- The profile file

-a instr action
Running FDPR-Pro: instrumentation

- fdprpro myexe -a instr –f {pwd}/myexe.prof -o myexe.instr

The input file

-a instr action

The profile file

The output instrumented file
Collecting profiling (training)

- In this phase, the user runs the instrumented executable
- Users run it with the usual invocation command, in the same way they would run the original executable
- fdprpro does not run in this phase
- Users should choose a representative workload to receive good optimization results
Collecting profile – how it works

- ProfFD
- MemorySegment
- ProfileFileName
- Instrumentation
  - shared library

Disk

Profile

Updated the attached profile

Open profile with profFD

The profile is detached on exit

Attach the profile to shared memory to MemorySegment

Shared memory
Collecting profile – how it works

**Instrumented executable**

- ProfFD
- MemorySegment
- ProfileFileName

**Instrumentation shared library**

- Attach the profile to shared memory to MemorySegment

**Disk**

- Open profile with profFD
- Updating the attached profile

**Profile**

- The profile is detached on exit

**Shared memory**
Collecting profile – how it works

Instrumented executable

ProfFD
MemorySegment
ProfileFileName

Updating the attached profile

Open profile with profFD

The profile is detached on exit

Attach the profile to shared memory to MemorySegment

Shared memory

Disk

Profile

Instrumentation shared library
Collecting profile – how it works

- Instrumented executable
- ProfFD
- MemorySegment
- ProfileFileName

Disk

Profile

- Updating the attached profile
- Open profile with profFD
- The profile is detached on exit

Instrumentation shared library

- Attach the profile to shared memory to MemorySegment

Shared memory
Collecting profile – how it works

Instrumented executable

ProfFD
MemorySegment
ProfileFileName

Updating the attached profile

Open profile with profFD
The profile is detached on exit

Attach the profile to shared memory
to MemorySegment

Profile

Disk

Instrumentation shared library

Shared memory
Collecting profile – how it works

Instrumented executable

ProfFD
MemorySegment
ProfileFileName

ProfFDMemorySegmentProfileFileName
Open profile with profFD
Updating the attached profile
The profile is detached on exit

Attach the profile to shared memory
to MemorySegment

Instrumentation shared library

Disk
Profile
Shared memory
Collecting profile

- **ProfileFileName** – is a name (including path) of the profile file
  - Default is `{input file full path}.nprof`
  - Override the default by:
    - Using –f option in instrumentation phase
    - Using FDPR_PROF_DIR environment variable in profile collection phase
      - Overrides only the path to the profile file
      - Important when creating “cross instrumentation”

- **ProfFD** is the file descriptor for opening the profile file
  - Has different default values for each OS
  - Override the default by:
    - Using –fd option in the instrumentation phase
    - Using FDPR_PROF_FD environment variable during profile collection
    - Important to avoid collision with the executable used FDs
Collecting profiling
Collecting profiling

- MemorySegment is the location in memory to which the profile file will be attached
  - Default value 0 – memory location is chosen by the OS
  - Override default by
    - Using –M option in the instrumentation phase
    - Using FDPR_MEMORY_SEGMENT environment variable in training phase

- Instrumentation shared library
  - Location must be defined in the library path
  - AIX
    - libfdprinst.a and libfdprinst64.a
    - Default location /usr/lib/perf/
    - Use LIB_PATH to set library path
  - Linux
    - libfdprinst32.so and libfdprinst64.so
    - Default location /opt/ibm/fdprpro/lib
    - Use LD_LIBRARY_PATH to set library path
Running FDPR-Pro: optimization
Running FDPR-Pro: optimization

fdprpro myexe -a opt -f {pwd}/myexe.prof -O3 -o myexe.fdpr
Running FDPR-Pro: optimization

diamond fdprpro myexe -a opt –f {pwd}/myexe.prof -O3 -o myexe.fdpr

The input file
Running FDPR-Pro: optimization

- fdprpro myexe -a opt -f {pwd}/myexe.prof -O3 -o myexe.fdpr
Running FDPR-Pro: optimization

- fdprpro myexe -a opt -f {pwd}/myexe.prof -O3 -o myexe.fdpr
Running FDPR-Pro: optimization

- fdprpro myexe -a opt -f {pwd}/myexe.prof -O3 -o myexe.fdpr
Running FDPR-Pro: optimization

diamond fdprpro myexe -a opt –f {pwd}/myexe.prof -O3 -o myexe.fdpr

- The input file
- The profile file
- Optimization options
- The output optimized file
Optimization
Optimization

- The are four levels of optimization, -O is the basic one, -O4 is the most aggressive
  - -O
  - -O2
  - -O3
  - -O4

- The basic optimizations are
  - Code reordering
  - NOOP removal
  - Branch prediction bit setting
  - Branch folding
Code reordering optimization

- Reduce the number of I-cache misses
- Reduce the number of I-TLB misses
- Reduce the number of page faults
- Reduce the branch penalty
- Improve branch prediction
FDPR-Pro - analysis
FDPR-Pro - analysis

- FDPR-Pro can be used to produce disassembly files of an input executable
- `myexe.fdpr.dis_text`, `myexe.fdpr.dis_data` and `myexe.fdpr.dis_bss` are produced
- Usage: `fdprpro myexe -a anl -f {pwd}/myexe.prof -d -dd -db`
FDPR-Pro - analysis

- FDPR-Pro can be used to produce disassembly files of an input executable
- myexe.fdpr.dis_text, myexe.fdpr.dis_data and myexe.fdpr.dis_bss are produced
- Usage: fdprpro myexe -a anl –f {pwd}/myexe.prof –d –dd -db

The input file
FDPR-Pro - analysis

- FDPR-Pro can be used to produce disassembly files of an input executable
- myexe.fdpr.dis_text, myexe.fdpr.dis_data and myexe.fdpr.dis_bss are produced
- Usage: fdprpro myexe -a anl -f {pwd}/myexe.prof -d -dd -db
FDPR-Pro - analysis

- FDPR-Pro can be used to produce disassembly files of an input executable
- myexe.fdpr.dis_text, myexe.fdpr.dis_data and myexe.fdpr.dis_bss are produced
- Usage: fdprpro myexe -a anl -f {pwd}/myexe.prof -d -dd -db
FDPR-Pro - analysis

- FDPR-Pro can be used to produce disassembly files of an input executable
- myexe.fdpr.dis_text, myexe.fdpr.dis_data and myexe.fdpr.dis_bss are produced
- Usage: fdprpro myexe -a anl -f \{pwd\}/myexe.prof -d -dd -db

Disassembly options: text, data and bss
Disassembly example

```plaintext
--- crt0main.s { safe csect .__start} ( size = 197 ) ---
--- __start { function } ( size = 197 ) ---
--- safe bb size = 56 func = __start ( prolog ) count = 1 orig addr : 0x10000128 ---
--- __C_runtime_pstartup */
--- 0x10000128: 0x82220028: lwzr17,40(r2)->( lnsym) /* .__start */, ( lnsym) /* .__start */, ( lnrel) 0x20000a04(DataBU) /* __C_runtime_pstartup */
---
--- 0x1000012c: 0x7c6e1b78: orr14,r3,r3
--- 0x10000130: 0x7c8f2378: orr15,r4,r4
--- 0x10000134: 0x7cb02b78: orr16,r5,r5
--- 0x10000138: 0x7c1ef800: cmpcr0,0x0,r30,r31
--- 0x1000013c: 0x91c20000: stwr14,0(r2)->( lnrel) 0x200009dc(DataBU) /* p_xargc */
--- 0x10000140: 0x91e20004: stwr15,4(r2)->( lnrel) 0x200009e0(DataBU) /* p_xargv */
--- 0x10000144: 0x82310000: lwzr17,0(r17)
--- 0x10000148: 0x81420000: lwzr17,0(r17)
--- 0x1000014c: 0x39200000: lir9,0
--- 0x10000150: 0x65290403: orisr9,r9,0x403
--- 0x10000154: 0x2d110000: cmpicr2,0x0,r17,0
--- 0x10000158: 0x912a0000: stwr9,0(r10)
--- 0x1000015c: 0x912a0000: stwr9,0(r10)
--- 0x10000160: 0x48001d41: bl0x10001e98/* __threads_init */
---
--- 0x10000164: 0x60000000: orir0,r0,0x0
--- 0x10000168: 0x48000014: b0x1000017c
---
--- 0x1000016c: 0x39200000: li r9,0
--- 0x10000170: 0x65290403: orisr9,r9,0x403
--- 0x10000174: 0x2d110000: cmpicr2,0x0,r17,0
--- 0x10000178: 0x912a0000: stwr9,0(r10)
--- 0x1000017c: 0x40820000: bne cr0,0,0x1000016c
---
--- profiled targets : 0x10000160 #1
--- safe bb size = 4 func = __start count = 1 orig addr : 0x10000160
--- 0x10000160: 0x480001d4: bl 0x100001e98/* __threads_init */
---
--- profiled targets : 0x100001e98 #1
--- safe bb size = 8 func = __start count = 1 orig addr : 0x100001e98
--- 0x100001e98: 0x60000000: ori r0,0x0
--- 0x100001ec: 0x48000014: b 0x100001f0
---
--- profiled targets : 0x100001f0 #1
--- safe bb size = 8 func = __start count = 0 orig addr : 0x100001f0
```
FDPR (FDPR-Pro wrapper)
FDPR (FDPR-Pro wrapper)

- Interface to FDPR-Pro for the novice-users
- Enables activation of all FDPR-Pro operations in single command line
- Usage:
  - AIX: `fdpr -p myexe -O3 -o myexe.fdpr -x myexe myexe_arg1`
  - Linux: `fdpr --instr --train test --opt -O3 -o myexe.fdpr myexe`
  - test is: `$1 myexe_arg1`
FDPR (FDPR-Pro wrapper)

- Interface to FDPR-Pro for the novice-users
- Enables activation of all FDPR-Pro operations in single command line
- Usage:
  - AIX: fdpr –p myexe -O3 -o myexe.fdpr -x myexe myexe_arg1
  - Linux: fdpr –instr –train test —opt -O3 -o myexe.fdpr myexe
    - test is: $1 myexe_arg1

The input file
FDPR (FDPR-Pro wrapper)

- Interface to FDPR-Pro for the novice-users
- Enables activation of all FDPR-Pro operations in single command line
- Usage:
  - AIX: fdpr -p myexe -O3 -o myexe.fdpr -x myexe myexe_arg1
  - Linux: fdpr -instr -train test --opt -O3 -o myexe.fdpr myexe
- test is: $1 myexe_arg1

Optimization options

The input file
FDPR (FDPR-Pro wrapper)

- Interface to FDPR-Pro for the novice-users
- Enables activation of all FDPR-Pro operations in single command line
- Usage:
  - AIX: fdpr –p myexe -O3 -o myexe.fdpr -x myexe myexe_arg1
  - Linux: fdpr –instr –train test --opt -O3 -o myexe.fdpr myexe
  - test is: $1 myexe_arg1
FDPR (FDPR-Pro wrapper)

- Interface to FDPR-Pro for the novice-users
- Enables activation of all FDPR-Pro operations in single command line

Usage:
- AIX: fdpr –p myexe -O3 -o myexe.fdpr -x myexe myexe_arg1
- Linux: fdpr –instr –train test --opt -O3 -o myexe.fdpr myexe
  - test is: $1 myexe_arg1
Training exercises

- Run three phases of FDPR-Pro on some executable on AIX or Linux
- Create disassembly of text
  - Find Functions, Basic Blocks and Const Areas
  - Find “main” function in the executable
    - What is its size?
    - What is its address?
    - What is the size of the prolog BB of the “main” function?