Case study: Dynamic Instruction Distribution

Fu-Ching Yang

Thursday, October 06, 2016

Introduction

After finishing the NTHU Statistics course, I'm ready to apply this on the instruction execution frequency problem.

First, load the partial PC trace file of pattern-40766.

```
data <- read.table("40766_pc_trace_partial.log", header = FALSE, sep = ",")
colnames(data) <- "pc"
```

Data Summary

There are 10,000 dynamic instructions in this trace.

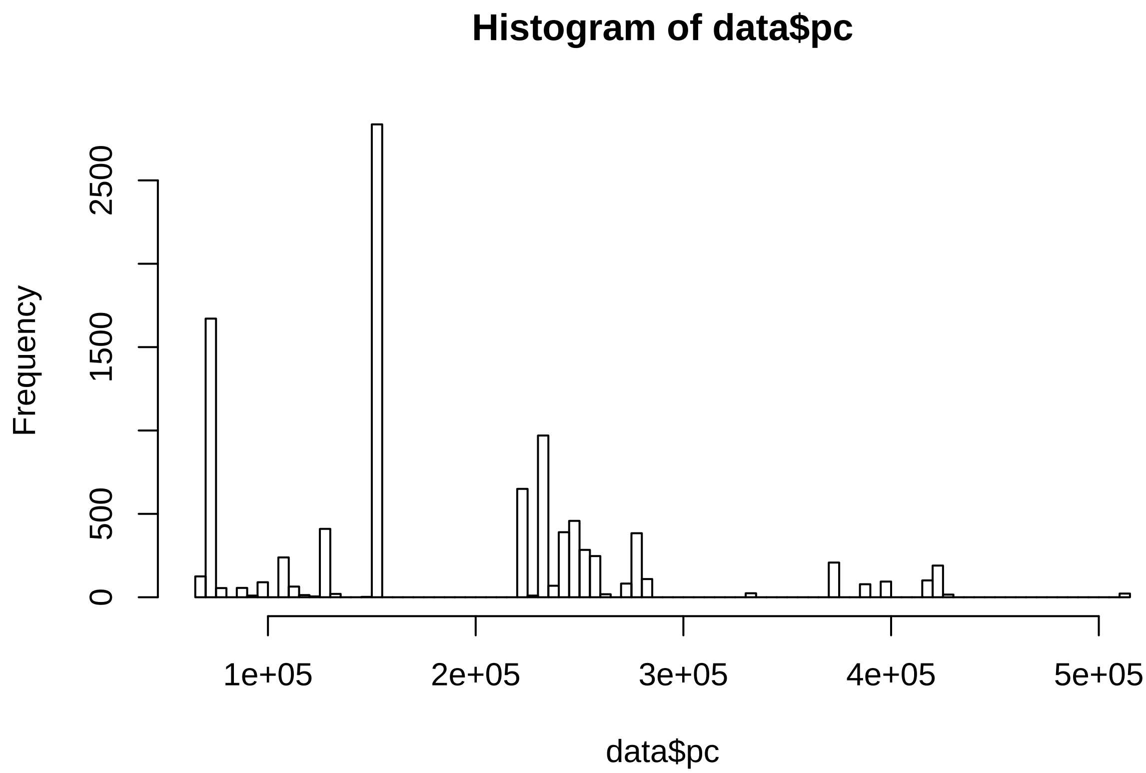
```
str(data)
```

```
## 'data.frame':    10000 obs. of  1 variable:
##  $ pc: num  68608 68612 68616 68620 68624 ...
```

I wonder how the execution amount of instructions varies. Histogram shows that the execution amount is contributed by only a few instructions.

You can see the bars are sparse, since x-axis is the instruction identity (PC).

```
hist(data$pc, breaks=100)
```



Let's ignore the instruction identity (PC) and sort the execution amount in decreasing order.

table() calculates the execution amount per PC.

```
tb <- table(data$pc)
head(tb)
```

```
##
## 65856 65860 65864 65868 65872 65876
##     1     1     1     3     3     3
```

Then, we only retrieve the execution amount (ignore PC), and assign it with serial numbers (instr_sn) just for plotting reason. The results are kept in exe_dist (execution amount distribution).

You can see this instr_sn as order statistics X(0), X(1), . . . , X(n).

```
exe_amount <- sort(as.vector(tb), decreasing=TRUE)
head(exe_amount, n=50)
```

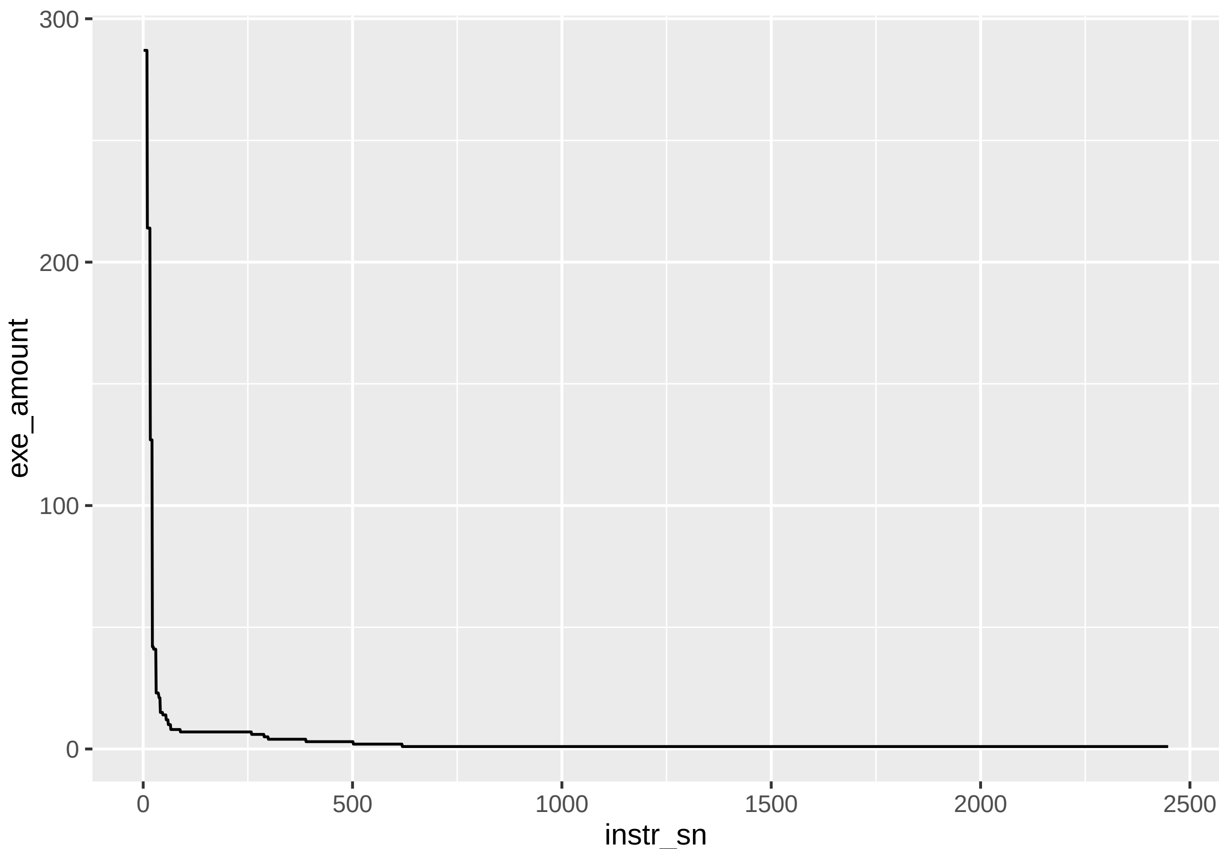
```
##  [1] 287 287 287 287 287 287 287 287 287 214 214 214 214 214 214 214 127
## [18] 127 127 127 127  42  42  42  41  41  41  41  41  41  23  23  23  23
## [35]  23  23  22  21  21  21  15  15  15  15  15  15  14  14  14  14
```

```
exe_dist <- data.frame(instr_sn=c(1:length(exe_amount)), exe_amount=exe_amount)
head(exe_dist)
```

```
##   instr_sn exe_amount
## 1        1        287
## 2        2        287
## 3        3        287
## 4        4        287
## 5        5        287
## 6        6        287
```

Then, we plot the execution amount distribution, as follows. It is very clear that, the execution amount is extremely contributed by only a few instructions.

```
ggplot(exe_dist, aes(x=instr_sn, y=exe_amount)) + geom_line()
```



Statistical modeling & estimation

From the execution distribution plot, since it looks like Exponential distribution, I use this as the statistical modeling.

Using fitdistr() to perform MLE as the point estimation, we can have the lambda estimate 0.1966581. The standard error is 0.003562512.

```
fit <- fitdistr(exe_amount, "geometric")
fit$estimate
```

```
##      prob
## 0.1966581
```

```
fit$sd
```

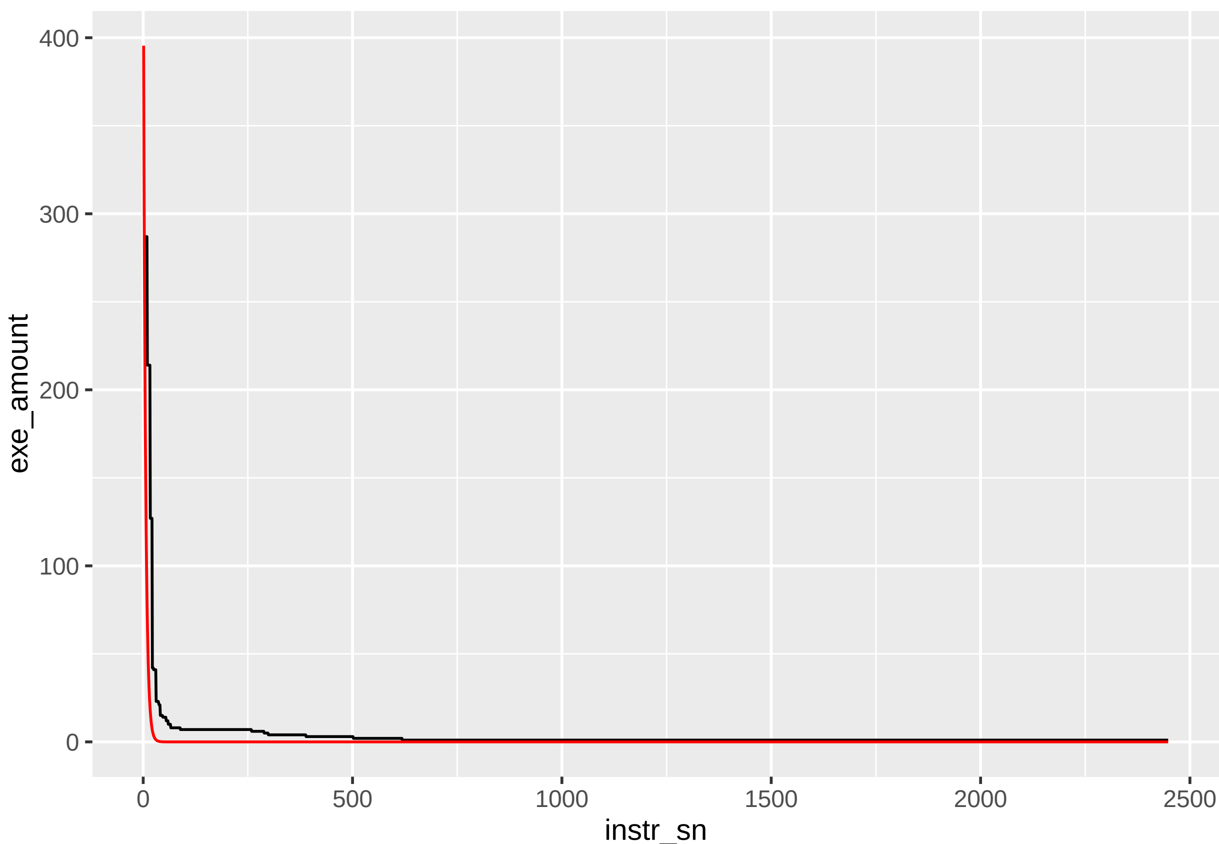
```
##        prob
## 0.003562512
```

Goodness of fit

To see how closely Exponential fits, I bi-plot the original execution amount distribution (black) and the Exponential distribution (red).

Unfortunately, they are not alike.

```
ggplot(exe_dist, aes(x=instr_sn, y=exe_amount)) + geom_line() + geom_line(aes(y=length(exe_amount)*dexp
```



Let's use chi-square test to see exactly what numbers tell.

H0: Execution amount distribution is the Exponential distribution with lambda 0.1966581. (dim=0) HA: Execution amount distribution is the Exponential distribution with other lambda. (dim=1)

Unfortunately, the p-value is 0, I need to reject the hypothesis that execution amount distribution follows Exponential distribution given the estimated lambda. Since the estimated lambda is the best I got, I believe Exponential distribution might not be the good distribution for execution amount distribution.

```
results <- chisq.test(exe_amount, p=dexp(exe_dist$instr_sn, rate=fit$estimate), rescale.p=TRUE, simulat
results$statistic
```

```
##     X-squared
## 3.081785e+206
```

```
pchisq(results$statistic, df=1, lower.tail=FALSE)
```

```
## X-squared
##         0
```

Even if I reduce the segment number, the goodness of fit does not improve. (Not sure if this is a reasonable move)

```
rediv <- tapply(exe_amount,cut(1:length(exe_amount),100),FUN=sum)
results <- chisq.test(rediv, p=dexp(c(1:100), rate=fit$estimate), rescale.p=TRUE, simulate.p.value=TRUE
results$statistic
```

```
## X-squared
## 539542977
```

```
pchisq(results$statistic, df=1, lower.tail=FALSE)
```

```
## X-squared
##         0
```

```
#hist(rediv)
#hist(length(exe_amount) * dexp(c(1:100), rate=fit$estimate))
```

I also try other common distribution, but no luck.

Hypothesis testing

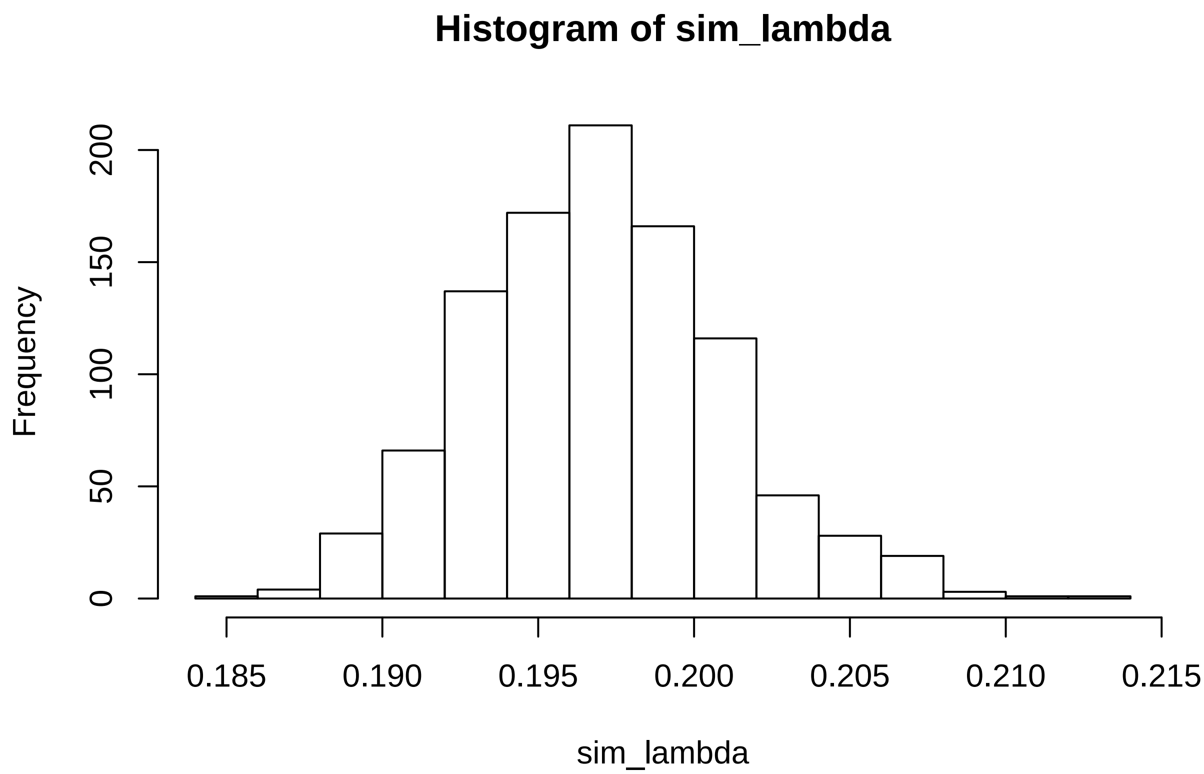
Since the goodness of fit fails, there is no reason to use Exponential distribution to perform futher investigation on the data. However, to make this statistical inference complete, I pretend the Exponential distribution with the estimated lambda is ok.

To get the sampling distribution of lambda, I use bootstrap method. The lambda is set to the estimated one from fitdistr(). The sample size is set to the static instruction amount. Results are saved in sim_lambda.

```
sim_lambda <- NULL
for(i in c(1:1000)) {
    sim_lambda <- c(sim_lambda, 1/mean(rexp(length(exe_amount), rate=fit$estimate)))
}
```

I can plot sim_lambda to see this sampling distribution.

```r
hist(sim_lambda)
```



As long as I have the sampling distribution, I can provide the confidence interval. Here, a 95% confidence interval is [0.1894, 0.2049]

The standard error from such sampling distribution is closed to the one reported by fitdistr().

```r
quantile(sim_lambda, c(0.025, 0.975))
```

```
##      2.5%     97.5%
## 0.1896270 0.2058085
```

```r
sd(sim_lambda)
```

```
## [1] 0.004005605
```

```r
fit$sd
```

```
##        prob
## 0.003562512
```

Then, I can answer question like "will lambda be bigger than 0.20".

By looking for the sampling distribution, the probability bigger than 0.20 is 22%, which is the p-Value. Thus, if we set type-I error to 5% (one-sided), we cannot reject this null hypothesis. In other words, lambda is not bigger than 0.20.

```r
p_value <- sum(sim_lambda[sim_lambda>0.2])/sum(sim_lambda)
p_value
```

```
## [1] 0.2200534
```