

Project on CPU, GPU, and TPU

Lixian Chen, Zhanhao Zhang, Jingbin Cao

4/14/2021

Preparing Required Packages

Read Data

```
data <- read.csv("../data/Runtime.csv")
head(data)
```

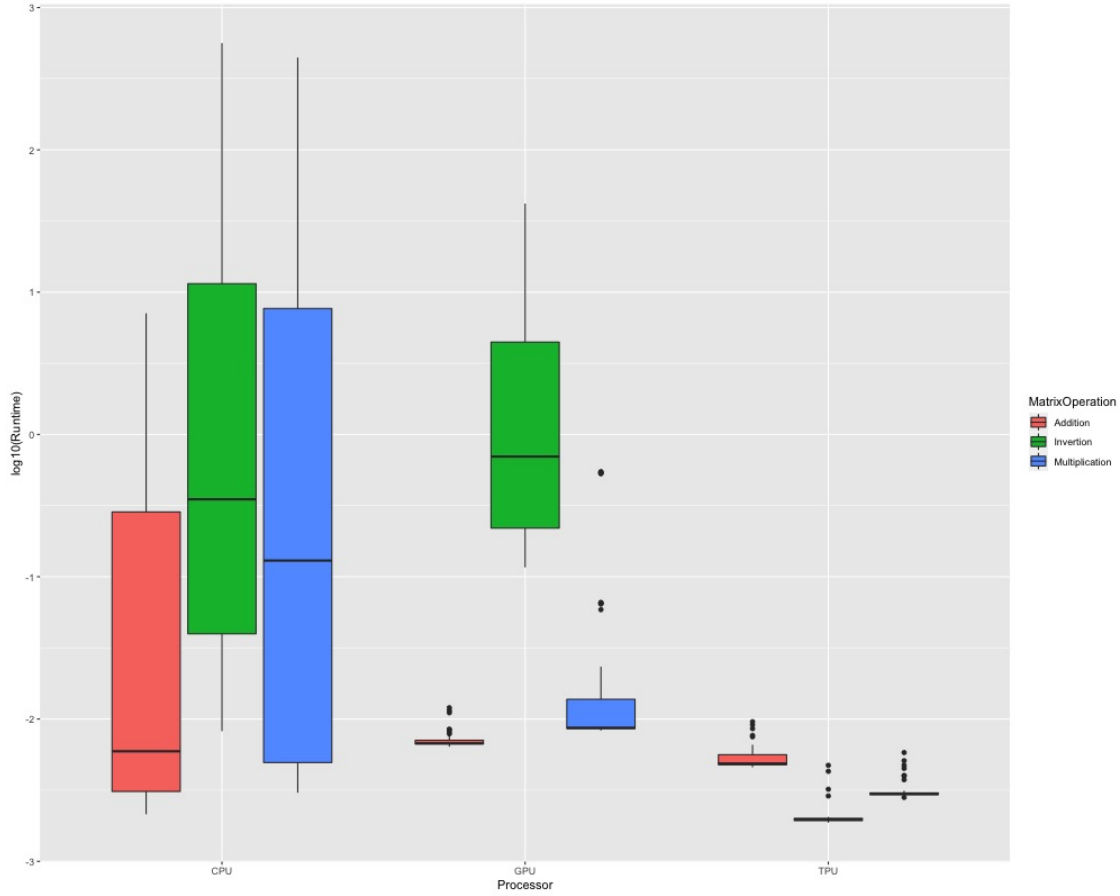
```
##      Runtime Processor MatrixSize MatrixOperation Trial
## 1 0.003411770      CPU         10      Addition     1
## 2 0.006412983      CPU         10      Addition     2
## 3 0.003450394      CPU         10      Addition     3
## 4 0.003098965      CPU         10      Addition     4
## 5 0.002490997      CPU         10      Addition     5
## 6 0.002594948      CPU         20      Addition     1
```

We tested three types of processors CPU, GPU, and TPU for three kinds of matrix operation, addition, multiplication, and inversion, with the matrix from size 10 to size 2160. We repeat each test for five times. **We measured $\log_{10}(\text{run-time})$ for each trial, and we use that as the evaluation of the performances.**

Simple Plots

Here is the general visualization for the performances of each processor under three matrix operations:

```
jpeg(filename = "../figs/overview.jpeg", width = 1000, height = 800, quality = 10000)
ggplot(data = data, aes(x = Processor, y = log10(Runtime))) +
  geom_boxplot(aes(fill = MatrixOperation))
while (!is.null(dev.list())) dev.off()
```



Jingbin Cao Part I

One Way Anova for different matrix sized for each pair of processor and matrix operation: $\mu_0 = Matrix_{Size} = 320$ $\mu_1 = Matrix_{Size} = 640$ $\mu_2 = Matrix_{Size} = 1280$ $\mu_3 = Matrix_{Size} = 2160$

Getting Data

```
cpu_add <- data[data$Processor == "CPU" & data$MatrixOperation=="Addition" & data$MatrixSize >= 320,]
cpu_mult <- data[data$Processor == "CPU" & data$MatrixOperation=="Multiplication" & data$MatrixSize >= 320,]
cpu_inv <- data[data$Processor == "CPU" & data$MatrixOperation=="Inversion" & data$MatrixSize >= 320,]
gpu_add <- data[data$Processor == "GPU" & data$MatrixOperation=="Addition" & data$MatrixSize >= 320,]
gpu_mult <- data[data$Processor == "GPU" & data$MatrixOperation=="Multiplication" & data$MatrixSize >= 320,]
gpu_inv <- data[data$Processor == "GPU" & data$MatrixOperation=="Inversion" & data$MatrixSize >= 320,]
tpu_add <- data[data$Processor == "TPU" & data$MatrixOperation=="Addition" & data$MatrixSize >= 320,]
tpu_mult <- data[data$Processor == "TPU" & data$MatrixOperation=="Multiplication" & data$MatrixSize >= 320,]
tpu_inv <- data[data$Processor == "TPU" & data$MatrixOperation=="Inversion" & data$MatrixSize >= 320,]
```

Anovas

```
summary(aov(Runtime ~ MatrixSize, data=cpu_add))
```

```
##              Df Sum Sq Mean Sq F value    Pr(>F)
## MatrixSize   1 152.37  152.37   338.2 4.08e-13 ***
## Residuals   18   8.11    0.45
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(aov(Runtime ~ MatrixSize, data=cpu_mult))
```

```
##              Df Sum Sq Mean Sq F value    Pr(>F)
## MatrixSize   1 619133  619133   187.8 5.8e-11 ***
## Residuals   18  59356    3298
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(aov(Runtime ~ MatrixSize, data=cpu_inv))
```

```
##              Df Sum Sq Mean Sq F value    Pr(>F)
## MatrixSize   1 984766  984766   203.7 2.95e-11 ***
## Residuals   18  87022    4835
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(aov(Runtime ~ MatrixSize, data=gpu_add))
```

```
##              Df      Sum Sq    Mean Sq F value Pr(>F)
## MatrixSize   1 1.200e-07 1.250e-07   0.038  0.847
## Residuals   18 5.845e-05 3.247e-06
```

```
summary(aov(Runtime ~ MatrixSize, data=gpu_mult))
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## MatrixSize   1 0.0003 0.00025   0.005  0.947
## Residuals   18 0.9873 0.05485
```

```
summary(aov(Runtime ~ MatrixSize, data=gpu_inv))
```

```
##              Df Sum Sq Mean Sq F value    Pr(>F)
## MatrixSize   1   4896    4896   476.7 2.11e-14 ***
## Residuals   18    185     10
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(aov(Runtime ~ MatrixSize, data=tpu_add))
```

```
##           Df      Sum Sq   Mean Sq F value Pr(>F)
## MatrixSize  1 1.000e-08 9.800e-09   0.006   0.94
## Residuals  18 2.993e-05 1.663e-06
```

```
summary(aov(Runtime ~ MatrixSize, data=tpu_mult))
```

```
##           Df      Sum Sq   Mean Sq F value Pr(>F)
## MatrixSize  1 1.200e-08 1.210e-08   0.028  0.869
## Residuals  18 7.745e-06 4.303e-07
```

```
summary(aov(Runtime ~ MatrixSize, data=tpu_inv))
```

```
##           Df      Sum Sq   Mean Sq F value Pr(>F)
## MatrixSize  1 3.300e-07 3.301e-07   1.13  0.302
## Residuals  18 5.258e-06 2.921e-07
```

Zhanhao Zhang Part I

Pros & Cons of Each Processor

When matrix size is the same, is there any processor or operation effects? Or is there any interactive effect?

```
df_cpu <- data[data$Processor == "CPU",]
lm(Runtime ~ MatrixSize + as.factor(MatrixOperation), data = df_cpu) %>%
  summary()
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixSize + as.factor(MatrixOperation),
##     data = df_cpu)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -234.810  -41.042   -6.483   48.307  248.752
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -67.641331   13.699933  -4.937 2.37e-06
## MatrixSize      0.120877    0.009097  13.287 < 2e-16
## as.factor(MatrixOperation)Inversion    71.726373   17.944905   3.997 0.000107
## as.factor(MatrixOperation)Multiplication 55.821041   17.944905   3.111 0.002291
##
## (Intercept)          ***
## MatrixSize           ***
## as.factor(MatrixOperation)Inversion    ***
## as.factor(MatrixOperation)Multiplication **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 85.12 on 131 degrees of freedom
## Multiple R-squared:  0.5971, Adjusted R-squared:  0.5879
## F-statistic: 64.73 on 3 and 131 DF,  p-value: < 2.2e-16
```

```
df_tpu <- data[data$Processor == "TPU",]
lm(Runtime ~ MatrixSize + as.factor(MatrixOperation), data = df_tpu) %>%
  summary()
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixSize + as.factor(MatrixOperation),
##     data = df_tpu)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.0008794 -0.0003316 -0.0002244 -0.0000993  0.0041166
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.460e-03  1.389e-04  39.296  <2e-16
## MatrixSize     -2.635e-08  9.226e-08  -0.286    0.776
## as.factor(MatrixOperation)Inversion  -3.320e-03  1.820e-04 -18.243  <2e-16
## as.factor(MatrixOperation)Multiplication -2.238e-03  1.820e-04 -12.296  <2e-16
##
## (Intercept)                ***
## MatrixSize
## as.factor(MatrixOperation)Inversion      ***
## as.factor(MatrixOperation)Multiplication ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0008633 on 131 degrees of freedom
## Multiple R-squared:  0.7256, Adjusted R-squared:  0.7193
## F-statistic: 115.4 on 3 and 131 DF,  p-value: < 2.2e-16
```

```
df_gpu <- data[data$Processor == "GPU",]
lm(Runtime ~ MatrixSize + as.factor(MatrixOperation), data = df_gpu) %>%
  summary()
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixSize + as.factor(MatrixOperation),
##     data = df_gpu)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4178  -3.7532   0.9807   2.6766  24.7084
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -2.9430365  1.0021808  -2.937  0.00392
```

```
## MatrixSize                0.0051962  0.0006655   7.808 1.64e-12
## as.factor(MatrixOperation)Inversion    6.7906886  1.3127100   5.173 8.42e-07
## as.factor(MatrixOperation)Multiplication 0.0669200  1.3127100   0.051 0.95942
##
## (Intercept)                **
## MatrixSize                  ***
## as.factor(MatrixOperation)Inversion    ***
## as.factor(MatrixOperation)Multiplication
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.227 on 131 degrees of freedom
## Multiple R-squared:  0.4237, Adjusted R-squared:  0.4105
## F-statistic: 32.1 on 3 and 131 DF, p-value: 1.273e-15
```

Lixian Chen Part I

When the processor is the same, is there any operation and matrix size effect? We want to answer the following question: in each scenario, which processor should we use?

Getting Data Ready

```
# Read Data
df <- fread("../data/Runtime.csv", header=TRUE)
attach(df)

MatrixOperation<-factor(MatrixOperation)
Processor<-factor(Processor)

CPUdata <- df %>% filter(Processor=="CPU")
GPUdata <- df %>% filter(Processor=="GPU")
TPUdata <- df %>% filter(Processor=="TPU")

describeBy(df$Runtime,group = df$MatrixSize, mat = TRUE) %>% #create dataframe
  select(MatrixSize=group1, N=n, Mean=mean, SD=sd, Median=median, Min=min, Max=max,
         Skew=skew, Kurtosis=kurtosis, SEM=se)
```

##	MatrixSize	N	Mean	SD	Median	Min	Max
## X11	10	45	0.02117285	0.04190229	0.005617142	0.001893044	0.1475253
## X12	20	45	0.02285517	0.04503445	0.004884481	0.001916409	0.1613362
## X13	40	45	0.03386522	0.07211264	0.005309582	0.001902819	0.2506576
## X14	80	45	0.06023479	0.12131721	0.006777525	0.001910210	0.4039948
## X15	160	45	0.13455488	0.22988237	0.006753206	0.001870394	0.7055206
## X16	320	45	0.54802946	0.77266887	0.008684158	0.001933098	1.8301501
## X17	640	45	2.65963363	4.07807665	0.064900875	0.001965523	11.5108950
## X18	1280	45	16.76243327	28.75264487	0.538181782	0.001962900	80.1931298

```
## X19      2560 45 117.16431474 210.69591026 0.008469582 0.001975775 562.2822752
##      Skew      Kurtosis      SEM
## X11 2.2815817 3.52258988 0.006246425
## X12 2.2598210 3.47569251 0.006713340
## X13 2.3149204 3.63635579 0.010749918
## X14 2.1514343 3.03908564 0.018084903
## X15 1.6153802 1.18296414 0.034268841
## X16 0.7633051 -1.35170085 0.115182675
## X17 1.1600905 -0.21560402 0.607923773
## X18 1.3579285 0.07252682 4.286191230
## X19 1.3393614 -0.10016629 31.408691862
```

```
#boxplot(Runtime~Processor*MatrixOperation)
#tapply(Runtime,list(Processor, MatrixOperation),mean)
#tapply(Runtime,MatrixOperation,mean)
```

```
group_by(df, Processor) %>%
  summarise(
    count = n(),
    mean = mean(Runtime, na.rm = TRUE),
    sd = sd(Runtime, na.rm = TRUE)
  )
```

```
## # A tibble: 3 x 4
##   Processor count      mean      sd
##   <chr>      <int>    <dbl>    <dbl>
## 1 CPU          135  43.5     133.
## 2 GPU          135   2.29      8.11
## 3 TPU          135  0.00359  0.00163
```

```
group_by(df, MatrixOperation) %>%
  summarise(
    count = n(),
    mean = mean(Runtime, na.rm = TRUE),
    sd = sd(Runtime, na.rm = TRUE)
  )
```

```
## # A tibble: 3 x 4
##   MatrixOperation count      mean      sd
##   <chr>          <int>    <dbl>    <dbl>
## 1 Addition          135  0.334     1.35
## 2 Inversion          135  26.5    107.
## 3 Multiplication    135  19.0     84.5
```

```
#detach(data)
```

```
size320data <- df %>% filter(MatrixSize==320)
size640data <- df %>% filter(MatrixSize==640)
size1280data <- df %>% filter(MatrixSize==1280)
size2560data <- df %>% filter(MatrixSize==2560)
```

Analysis

Anova

#320

```
fit2<-lm(Runtime~Processor+MatrixOperation, data = size320data)
summary(fit2)
```

```
##
## Call:
## lm(formula = Runtime ~ Processor + MatrixOperation, data = size320data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.63252 -0.44075  0.05662  0.39766  0.58682
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.5072     0.1496   3.389  0.00159 **
## ProcessorGPU     -0.4163     0.1639  -2.540  0.01508 *
## ProcessorTPU     -1.0252     0.1639  -6.254 2.08e-07 ***
## MatrixOperationInversion  1.1525     0.1639   7.031 1.70e-08 ***
## MatrixOperationMultiplication  0.4116     0.1639   2.511  0.01619 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4489 on 40 degrees of freedom
## Multiple R-squared:  0.6931, Adjusted R-squared:  0.6624
## F-statistic: 22.59 on 4 and 40 DF,  p-value: 8.149e-10
```

```
fit1<-lm(Runtime~Processor*MatrixOperation, data = size320data)
summary(fit1)
```

```
##
## Call:
## lm(formula = Runtime ~ Processor * MatrixOperation, data = size320data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.0259142 -0.0004408 -0.0000341  0.0006995  0.0137136
##
## Coefficients:
##              Estimate Std. Error t value
## (Intercept)      0.067873   0.003147   21.57
## ProcessorGPU     -0.060596   0.004450  -13.62
## ProcessorTPU     -0.062968   0.004450  -14.15
## MatrixOperationInversion  1.647123   0.004450  370.15
## MatrixOperationMultiplication  1.234854   0.004450  277.50
## ProcessorGPU:MatrixOperationInversion  0.165856   0.006293   26.36
## ProcessorTPU:MatrixOperationInversion -1.649847   0.006293 -262.17
## ProcessorGPU:MatrixOperationMultiplication -1.233006   0.006293 -195.93
## ProcessorTPU:MatrixOperationMultiplication -1.236834   0.006293 -196.54
```



```
##                                Pr(>|t|)
## (Intercept)                   < 2e-16 ***
## ProcessorGPU                  9.04e-16 ***
## ProcessorTPU                  2.80e-16 ***
## MatrixOperationInversion      < 2e-16 ***
## MatrixOperationMultiplication < 2e-16 ***
## ProcessorGPU:MatrixOperationInversion < 2e-16 ***
## ProcessorTPU:MatrixOperationInversion < 2e-16 ***
## ProcessorGPU:MatrixOperationMultiplication < 2e-16 ***
## ProcessorTPU:MatrixOperationMultiplication < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.007036 on 36 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 6.633e+04 on 8 and 36 DF,  p-value: < 2.2e-16
```

```
anova(fit2, fit1)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ Processor + MatrixOperation
## Model 2: Runtime ~ Processor * MatrixOperation
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1      40 8.0610
## 2      36 0.0018  4    8.0592 40700 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod320<-aov(Runtime~Processor*MatrixOperation, data = size320data)
#summary(mod320)
Anova(mod320,type="III")
```

```
## Anova Table (Type III tests)
##
## Response: Runtime
##               Sum Sq Df F value    Pr(>F)
## (Intercept)    0.0230  1  465.30 < 2.2e-16 ***
## Processor      0.0127  2   128.65 < 2.2e-16 ***
## MatrixOperation 7.3464  2 74200.69 < 2.2e-16 ***
## Processor:MatrixOperation 8.0592  4 40700.27 < 2.2e-16 ***
## Residuals      0.0018 36
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modF<-lm(Runtime~Processor+MatrixOperation, data = size320data)
modA<-lm(Runtime~Processor, data = size320data)
modB<-lm(Runtime~MatrixOperation, data = size320data)

anova(modA, modF)
```

```
## Analysis of Variance Table
```

```
##
## Model 1: Runtime ~ Processor
## Model 2: Runtime ~ Processor + MatrixOperation
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1      42 18.293
## 2      40  8.061  2    10.232 25.387 7.62e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(modB, modF)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ MatrixOperation
## Model 2: Runtime ~ Processor + MatrixOperation
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1      42 16.036
## 2      40  8.061  2    7.9754 19.788 1.061e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#640
fit2_640<-lm(Runtime~Processor+MatrixOperation, data = size640data)
summary(fit2)
```

```
##
## Call:
## lm(formula = Runtime ~ Processor + MatrixOperation, data = size320data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.63252 -0.44075  0.05662  0.39766  0.58682
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.5072     0.1496   3.389  0.00159 **
## ProcessorGPU     -0.4163     0.1639  -2.540  0.01508 *
## ProcessorTPU     -1.0252     0.1639  -6.254 2.08e-07 ***
## MatrixOperationInversion  1.1525     0.1639   7.031 1.70e-08 ***
## MatrixOperationMultiplication  0.4116     0.1639   2.511  0.01619 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4489 on 40 degrees of freedom
## Multiple R-squared:  0.6931, Adjusted R-squared:  0.6624
## F-statistic: 22.59 on 4 and 40 DF,  p-value: 8.149e-10
```

```
fit1_640<-lm(Runtime~Processor*MatrixOperation, data = size640data)
summary(fit1_640)
```

```
##
## Call:
```

```
## lm(formula = Runtime ~ Processor * MatrixOperation, data = size640data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.070744 -0.001994 -0.000012  0.000669  0.055012
##
## Coefficients:
##                                Estimate Std. Error t value
## (Intercept)                   0.282343   0.008379   33.70
## ProcessorGPU                  -0.275710   0.011849  -23.27
## ProcessorTPU                  -0.277395   0.011849  -23.41
## MatrixOperationInversion      11.186470   0.011849  944.06
## MatrixOperationMultiplication  7.371087   0.011849  622.07
## ProcessorGPU:MatrixOperationInversion -6.741354   0.016758 -402.29
## ProcessorTPU:MatrixOperationInversion -11.189419   0.016758 -667.73
## ProcessorGPU:MatrixOperationMultiplication -7.313893   0.016758 -436.45
## ProcessorTPU:MatrixOperationMultiplication -7.373071   0.016758 -439.99
##                                Pr(>|t|)
## (Intercept)                   <2e-16 ***
## ProcessorGPU                  <2e-16 ***
## ProcessorTPU                  <2e-16 ***
## MatrixOperationInversion      <2e-16 ***
## MatrixOperationMultiplication <2e-16 ***
## ProcessorGPU:MatrixOperationInversion <2e-16 ***
## ProcessorTPU:MatrixOperationInversion <2e-16 ***
## ProcessorGPU:MatrixOperationMultiplication <2e-16 ***
## ProcessorTPU:MatrixOperationMultiplication <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01874 on 36 degrees of freedom
## Multiple R-squared:  1, Adjusted R-squared:  1
## F-statistic: 2.606e+05 on 8 and 36 DF, p-value: < 2.2e-16
```

```
anova(fit2_640, fit1_640)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ Processor + MatrixOperation
## Model 2: Runtime ~ Processor * MatrixOperation
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1      40 184.706
## 2      36   0.013  4    184.69 131541 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod640<-aov(Runtime~Processor*MatrixOperation, data = size640data)
Anova(mod640,type="III")
```

```
## Anova Table (Type III tests)
##
## Response: Runtime
##                                Sum Sq Df    F value    Pr(>F)
```

```
## (Intercept)          0.40  1   1135.52 < 2.2e-16 ***
## Processor            0.25  2    363.15 < 2.2e-16 ***
## MatrixOperation      323.38  2 460630.58 < 2.2e-16 ***
## Processor:MatrixOperation 184.69  4 131541.31 < 2.2e-16 ***
## Residuals            0.01 36
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modF<-lm(Runtime~Processor+MatrixOperation, data = size640data)
modA<-lm(Runtime~Processor, data = size640data)
modB<-lm(Runtime~MatrixOperation, data = size640data)

anova(modA, modF)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ Processor
## Model 2: Runtime ~ Processor + MatrixOperation
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1      42 388.42
## 2      40 184.71  2    203.71 22.058 3.496e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(modB, modF)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ MatrixOperation
## Model 2: Runtime ~ Processor + MatrixOperation
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1      42 528.04
## 2      40 184.71  2    343.33 37.176 7.521e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#1280
fit2_1280<-lm(Runtime~Processor+MatrixOperation, data = size1280data)
summary(fit2)
```

```
##
## Call:
## lm(formula = Runtime ~ Processor + MatrixOperation, data = size320data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.63252 -0.44075  0.05662  0.39766  0.58682
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.5072     0.1496   3.389  0.00159 **
## ProcessorGPU     -0.4163     0.1639  -2.540  0.01508 *
```

```
## ProcessorTPU          -1.0252      0.1639  -6.254 2.08e-07 ***
## MatrixOperationInversion  1.1525      0.1639   7.031 1.70e-08 ***
## MatrixOperationMultiplication 0.4116      0.1639   2.511 0.01619 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4489 on 40 degrees of freedom
## Multiple R-squared:  0.6931, Adjusted R-squared:  0.6624
## F-statistic: 22.59 on 4 and 40 DF,  p-value: 8.149e-10
```

```
fit1_1280<-lm(Runtime~Processor*MatrixOperation, data = size1280data)
summary(fit1_1280)
```

```
##
## Call:
## lm(formula = Runtime ~ Processor * MatrixOperation, data = size1280data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32048 -0.00256 -0.00001  0.00333  0.51330
##
## Coefficients:
##                                Estimate Std. Error t value
## (Intercept)                   1.50458    0.05293   28.43
## ProcessorGPU                  -1.49482    0.07485  -19.97
## ProcessorTPU                  -1.49793    0.07485  -20.01
## MatrixOperationInversion       78.17525    0.07485 1044.41
## MatrixOperationMultiplication  56.11665    0.07485  749.71
## ProcessorGPU:MatrixOperationInversion -66.68873    0.10586 -630.00
## ProcessorTPU:MatrixOperationInversion -78.17992    0.10586 -738.56
## ProcessorGPU:MatrixOperationMultiplication -55.58840    0.10586 -525.14
## ProcessorTPU:MatrixOperationMultiplication -56.11973    0.10586 -530.16
##                                Pr(>|t|)
## (Intercept)                   <2e-16 ***
## ProcessorGPU                   <2e-16 ***
## ProcessorTPU                   <2e-16 ***
## MatrixOperationInversion       <2e-16 ***
## MatrixOperationMultiplication  <2e-16 ***
## ProcessorGPU:MatrixOperationInversion <2e-16 ***
## ProcessorTPU:MatrixOperationInversion <2e-16 ***
## ProcessorGPU:MatrixOperationMultiplication <2e-16 ***
## ProcessorTPU:MatrixOperationMultiplication <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1183 on 36 degrees of freedom
## Multiple R-squared:  1, Adjusted R-squared:  1
## F-statistic: 3.246e+05 on 8 and 36 DF,  p-value: < 2.2e-16
```

```
anova(fit2_1280, fit1_1280)
```

```
## Analysis of Variance Table
##
```

```
## Model 1: Runtime ~ Processor + MatrixOperation
## Model 2: Runtime ~ Processor * MatrixOperation
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1      40 9812.3
## 2      36    0.5  4    9811.8 175129 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modF<-lm(Runtime~Processor+MatrixOperation, data = size1280data)
modA<-lm(Runtime~Processor, data = size1280data)
modB<-lm(Runtime~MatrixOperation, data = size1280data)

anova(modA, modF)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ Processor
## Model 2: Runtime ~ Processor + MatrixOperation
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1      42 16666.1
## 2      40 9812.3  2    6853.7 13.97 2.505e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(modB, modF)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ MatrixOperation
## Model 2: Runtime ~ Processor + MatrixOperation
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1      42 29521.7
## 2      40 9812.3  2    19709 40.173 2.708e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod1280<-aov(Runtime~Processor*MatrixOperation, data = size1280data)
Anova(mod1280,type="III")
```

```
## Anova Table (Type III tests)
##
## Response: Runtime
##


|                           | Sum Sq  | Df | F value   | Pr(>F)        |
|---------------------------|---------|----|-----------|---------------|
| (Intercept)               | 11.3    | 1  | 808.11    | < 2.2e-16 *** |
| Processor                 | 7.5     | 2  | 266.44    | < 2.2e-16 *** |
| MatrixOperation           | 16245.1 | 2  | 579906.51 | < 2.2e-16 *** |
| Processor:MatrixOperation | 9811.8  | 4  | 175128.60 | < 2.2e-16 *** |
| Residuals                 | 0.5     | 36 |           |               |


## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

#2560

```
fit2_2560<-lm(Runtime~Processor+MatrixOperation, data = size2560data)
summary(fit2)
```

```
##
## Call:
## lm(formula = Runtime ~ Processor + MatrixOperation, data = size320data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.63252 -0.44075  0.05662  0.39766  0.58682
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.5072     0.1496   3.389  0.00159 **
## ProcessorGPU     -0.4163     0.1639  -2.540  0.01508 *
## ProcessorTPU     -1.0252     0.1639  -6.254 2.08e-07 ***
## MatrixOperationInversion  1.1525     0.1639   7.031 1.70e-08 ***
## MatrixOperationMultiplication  0.4116     0.1639   2.511  0.01619 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4489 on 40 degrees of freedom
## Multiple R-squared:  0.6931, Adjusted R-squared:  0.6624
## F-statistic: 22.59 on 4 and 40 DF,  p-value: 8.149e-10
```

```
fit1_2560<-lm(Runtime~Processor*MatrixOperation, data = size2560data)
summary(fit1_2560)
```

```
##
## Call:
## lm(formula = Runtime ~ Processor * MatrixOperation, data = size2560data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.85301 -0.00118 -0.00007  0.00026  1.23586
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      7.0363     0.1765   39.87 <2e-16 ***
## ProcessorGPU     -7.0296     0.2496  -28.17 <2e-16 ***
## ProcessorTPU     -7.0315     0.2496  -28.18 <2e-16 ***
## MatrixOperationInversion  554.0101     0.2496 2219.89 <2e-16 ***
## MatrixOperationMultiplication  437.5216     0.2496 1753.13 <2e-16 ***
## ProcessorGPU:MatrixOperationInversion -512.2048     0.3529 -1451.25 <2e-16 ***
## ProcessorTPU:MatrixOperationInversion -554.0125     0.3529 -1569.71 <2e-16 ***
## ProcessorGPU:MatrixOperationMultiplication -437.5188     0.3529 -1239.64 <2e-16 ***
## ProcessorTPU:MatrixOperationMultiplication -437.5234     0.3529 -1239.65 <2e-16 ***
##
## (Intercept)      <2e-16 ***
## ProcessorGPU     <2e-16 ***
## ProcessorTPU     <2e-16 ***
## MatrixOperationInversion  <2e-16 ***
```

```
## MatrixOperationMultiplication          <2e-16 ***
## ProcessorGPU:MatrixOperationInversion  <2e-16 ***
## ProcessorTPU:MatrixOperationInversion  <2e-16 ***
## ProcessorGPU:MatrixOperationMultiplication <2e-16 ***
## ProcessorTPU:MatrixOperationMultiplication <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3946 on 36 degrees of freedom
## Multiple R-squared:      1, Adjusted R-squared:      1
## F-statistic: 1.568e+06 on 8 and 36 DF, p-value: < 2.2e-16
```

```
anova(fit2_2560, fit1_2560)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ Processor + MatrixOperation
## Model 2: Runtime ~ Processor * MatrixOperation
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1      40 541548
## 2      36      6  4    541542 869482 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod2560<-aov(Runtime~Processor*MatrixOperation, data = size2560data)
Anova(mod2560,type="III")
```

```
## Anova Table (Type III tests)
##
## Response: Runtime
##               Sum Sq Df    F value    Pr(>F)
## (Intercept)      248  1    1589.81 < 2.2e-16 ***
## Processor        165  2     529.08 < 2.2e-16 ***
## MatrixOperation  853203  2 2739748.40 < 2.2e-16 ***
## Processor:MatrixOperation 541542  4 869481.88 < 2.2e-16 ***
## Residuals          6 36
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modF<-lm(Runtime~Processor+MatrixOperation, data = size2560data)
modA<-lm(Runtime~Processor, data = size2560data)
modB<-lm(Runtime~MatrixOperation, data = size2560data)

anova(modA, modF)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ Processor
## Model 2: Runtime ~ Processor + MatrixOperation
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1      42 859034
## 2      40 541548  2    317486 11.725 9.829e-05 ***
```



```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(modB, modF)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ MatrixOperation
## Model 2: Runtime ~ Processor + MatrixOperation
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1      42 1635796
## 2      40  541548  2   1094248 40.412 2.501e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Mean of Runtime

```
tapply(Runtime,list(Processor, MatrixOperation),mean, data = size320data)
```

```
##           Addition      Inversion Multiplication
## CPU 0.989863337 72.716236062  56.810904413
## GPU 0.007234912  6.797923491   0.074154954
## TPU 0.005444972  0.002124808   0.003207064
```

```
tapply(Runtime,list(Processor, MatrixOperation),mean, data = size640data)
```

```
##           Addition      Inversion Multiplication
## CPU 0.989863337 72.716236062  56.810904413
## GPU 0.007234912  6.797923491   0.074154954
## TPU 0.005444972  0.002124808   0.003207064
```

```
tapply(Runtime,list(Processor, MatrixOperation),mean, data = size1280data)
```

```
##           Addition      Inversion Multiplication
## CPU 0.989863337 72.716236062  56.810904413
## GPU 0.007234912  6.797923491   0.074154954
## TPU 0.005444972  0.002124808   0.003207064
```

```
tapply(Runtime,list(Processor, MatrixOperation),mean, data = size2560data)
```

```
##           Addition      Inversion Multiplication
## CPU 0.989863337 72.716236062  56.810904413
## GPU 0.007234912  6.797923491   0.074154954
## TPU 0.005444972  0.002124808   0.003207064
```

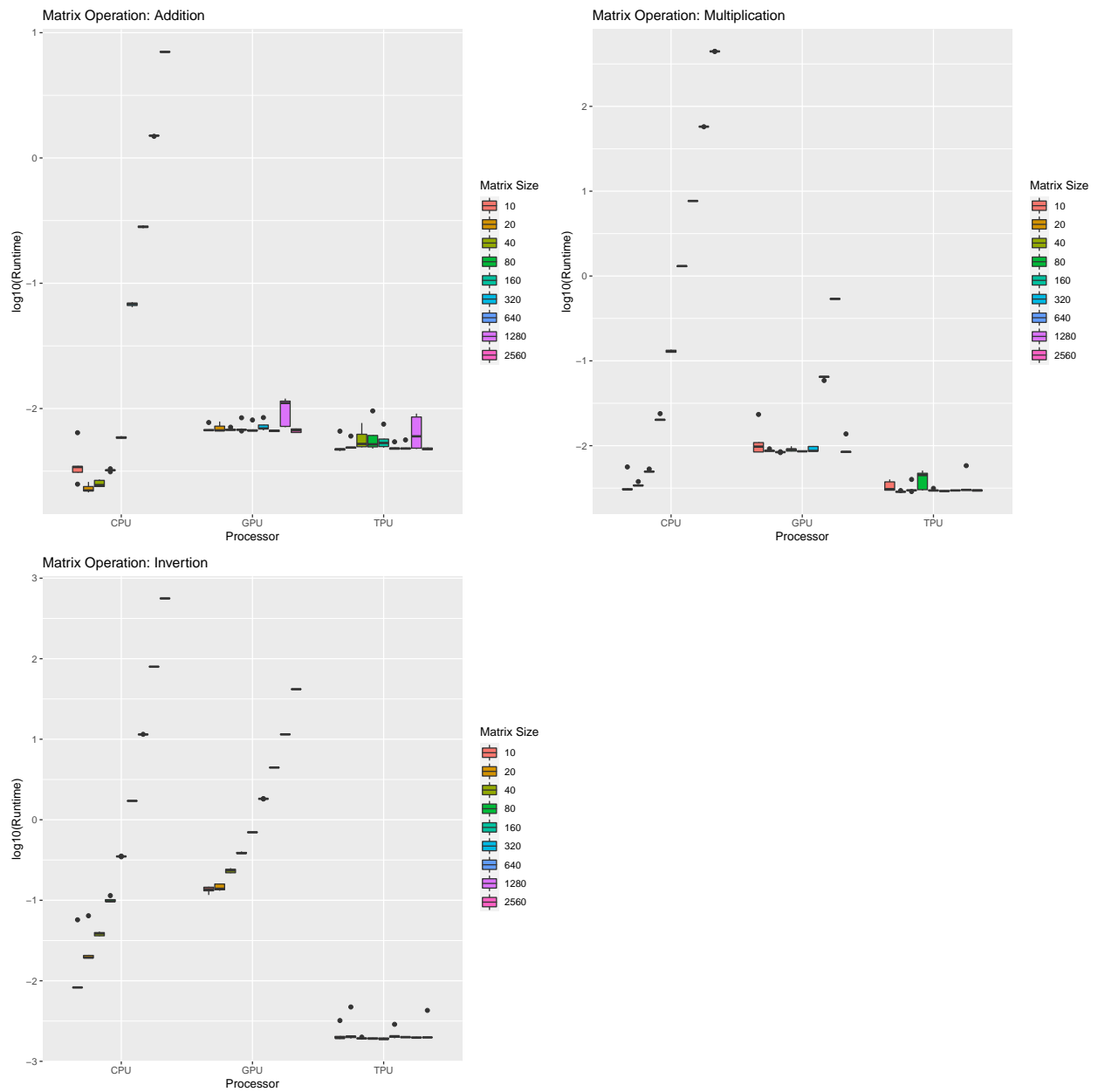


Figure 1: Operation v.s. Processors for Each Matrix Size

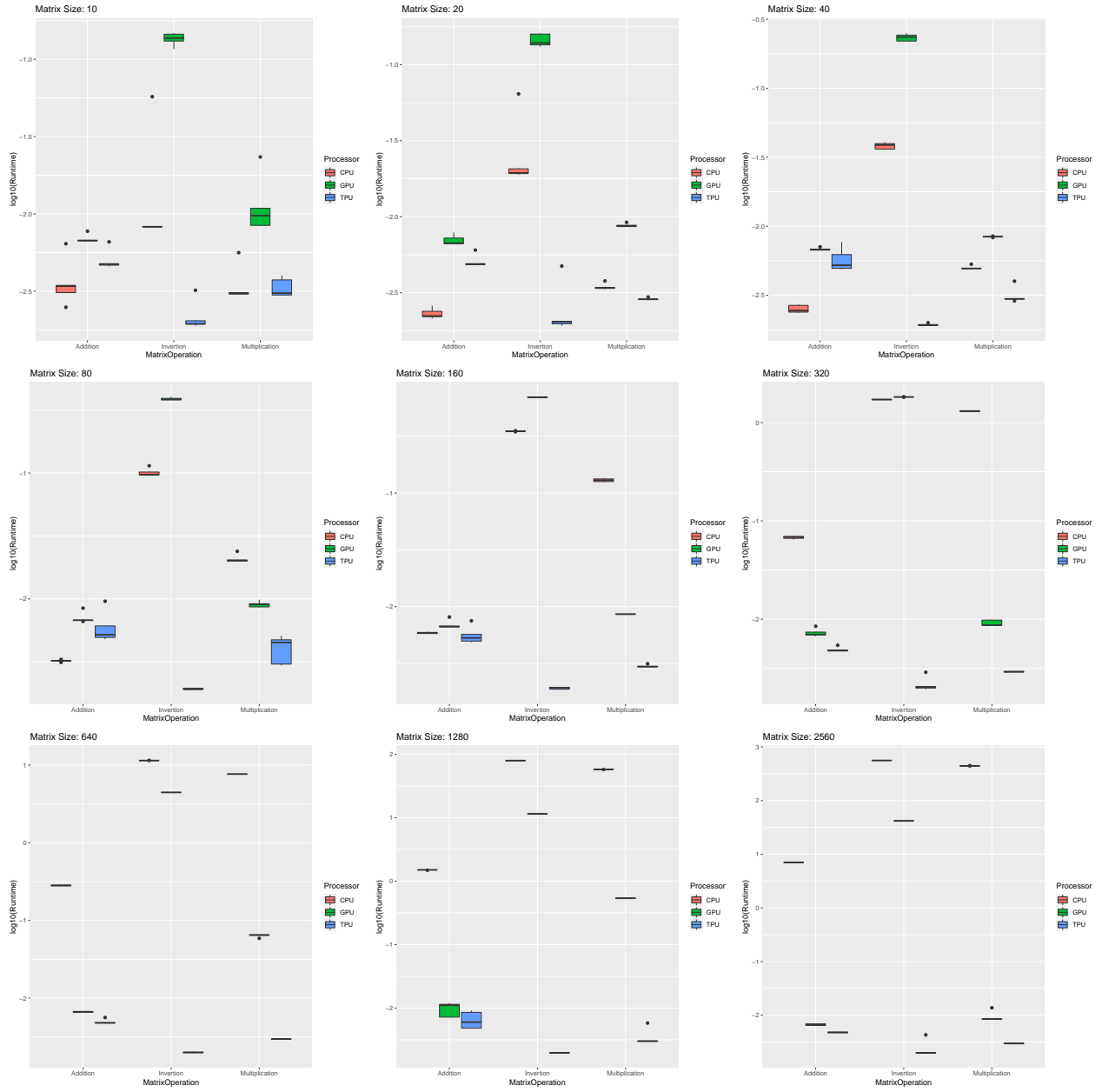


Figure 2: Matrix Size v.s. Operations for Each Processor.

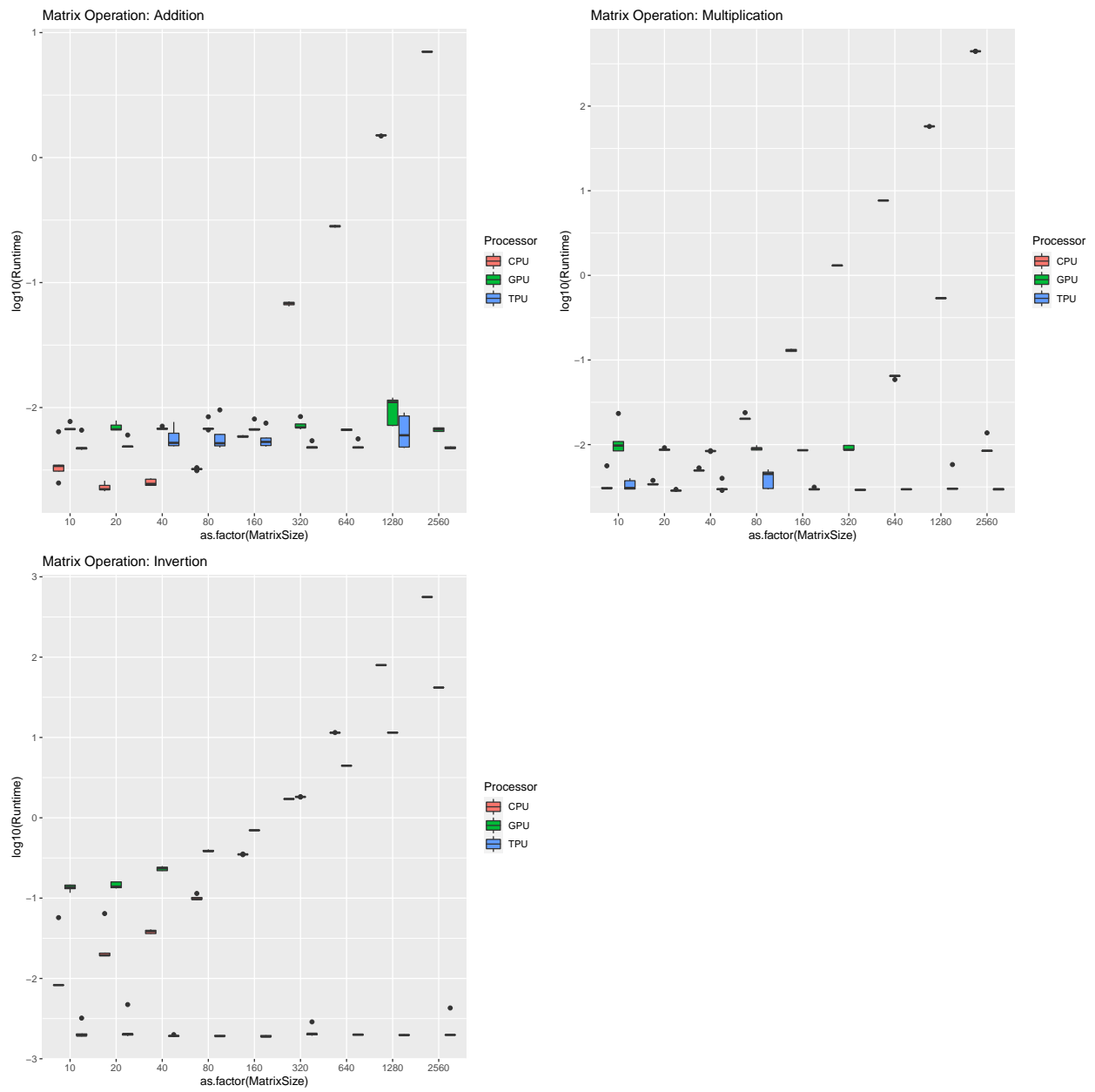


Figure 3: Operation v.s. Matrix Size for Each Processor

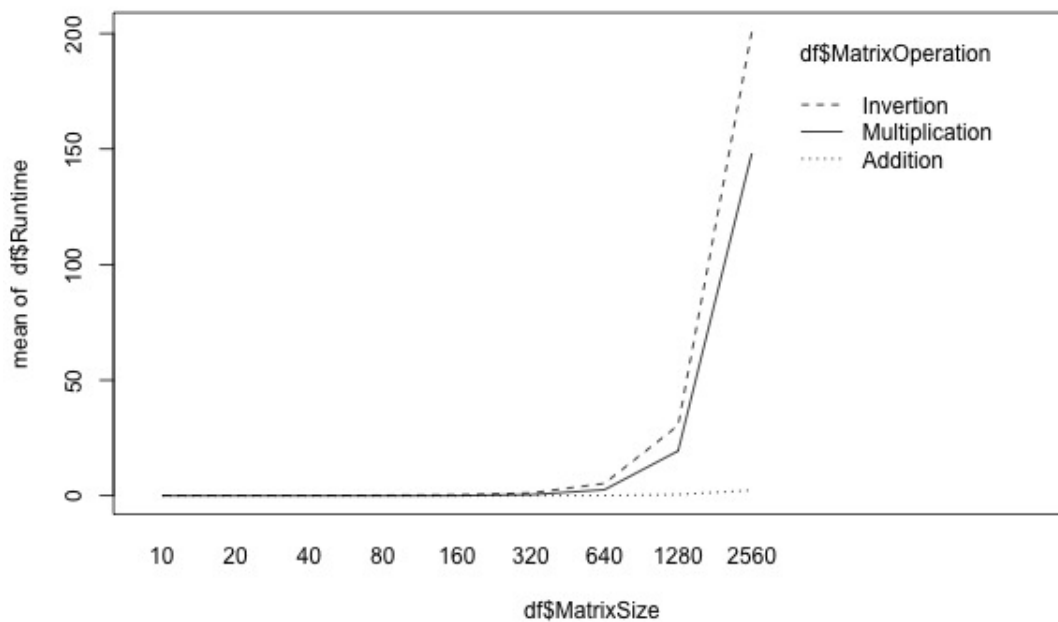
Zhanhao Zhang Part II

General Visualization

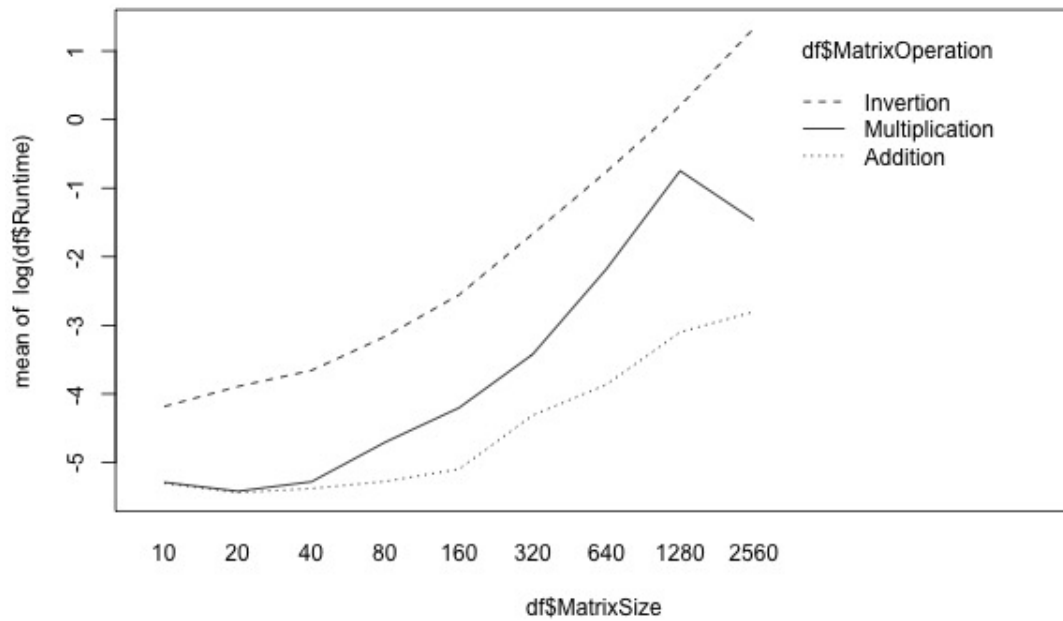
Lixian Chen Part II: Plots

Interaction Plots

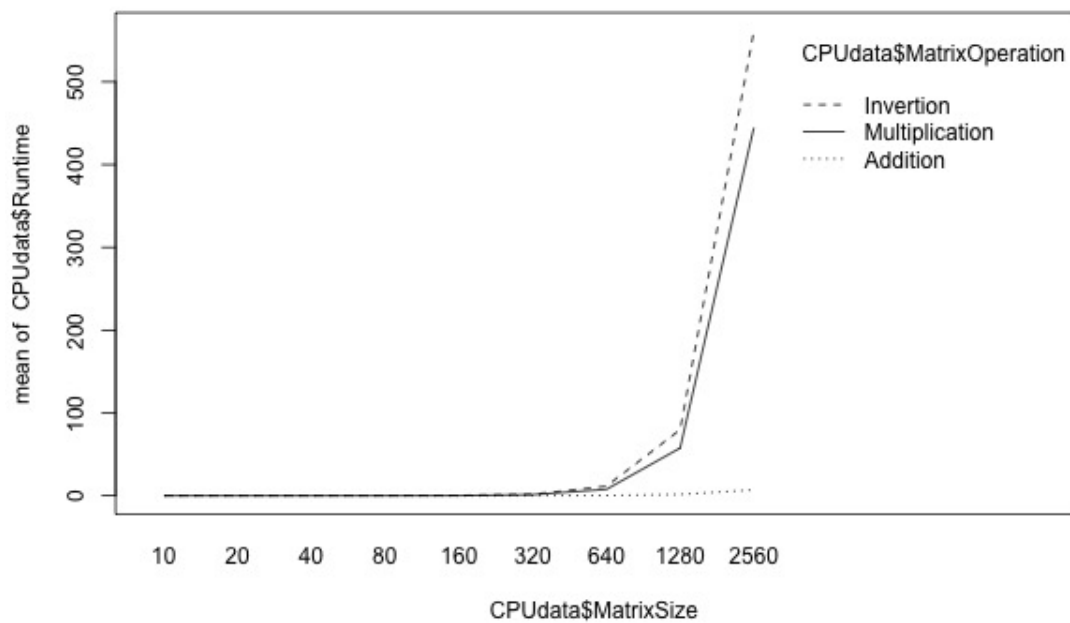
```
jpeg(filename = "../figs/interaction_size_time.jpeg", width = 600, height = 400, quality = 10000)
interaction.plot(df$MatrixSize, df$MatrixOperation, df$Runtime)
while (!is.null(dev.list())) dev.off()
```



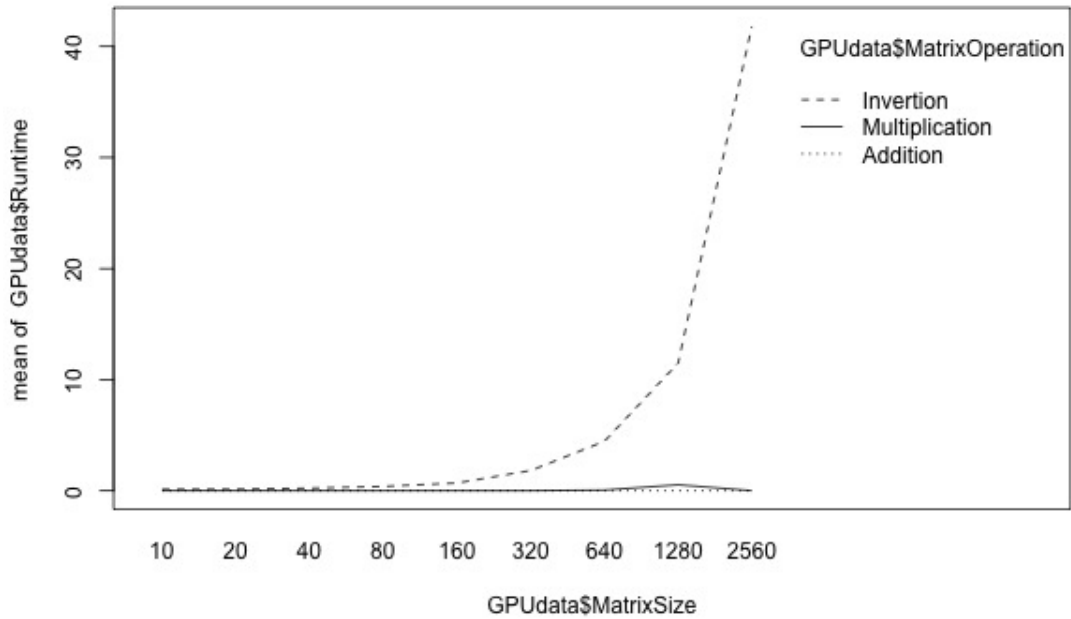
```
jpeg(filename = "../figs/interaction_size_log_time.jpeg", width = 600, height = 400, quality = 10000)
interaction.plot(df$MatrixSize, df$MatrixOperation, log(df$Runtime))
while (!is.null(dev.list())) dev.off()
```



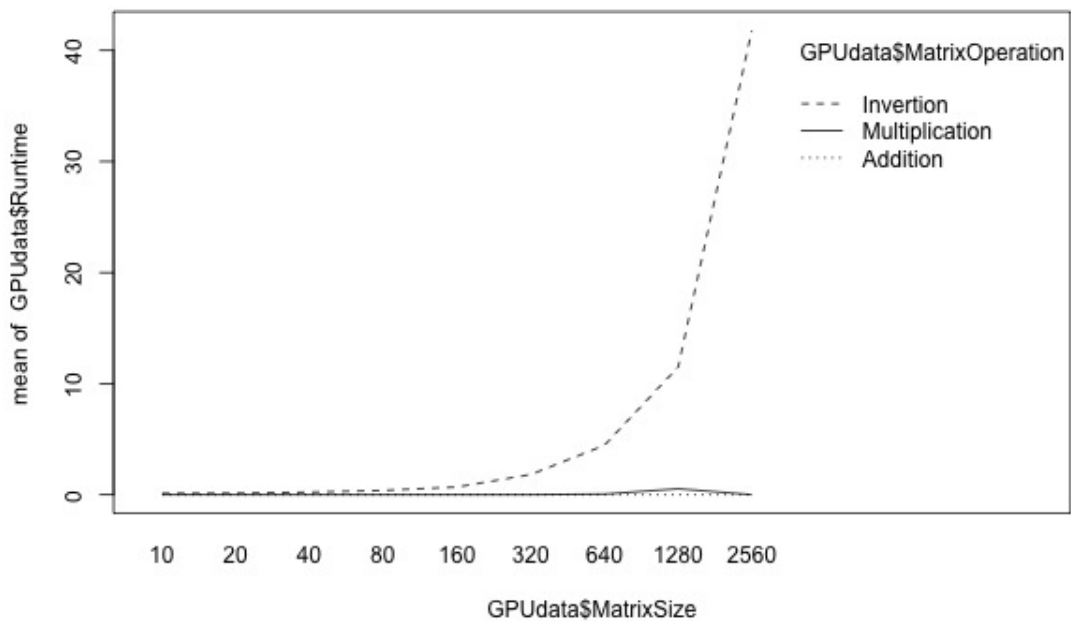
```
jpeg(filename = "../figs/interaction_CPU_size_time.jpeg", width = 600, height = 400, quality = 10000)
interaction.plot(CPUdata$MatrixSize, CPUdata$MatrixOperation, CPUdata$Runtime)
```



```
jpeg(filename = "../figs/interaction_GPU_size_time.jpeg", width = 600, height = 400, quality = 10000)
interaction.plot(GPUdata$MatrixSize, GPUdata$MatrixOperation, GPUdata$Runtime)
```

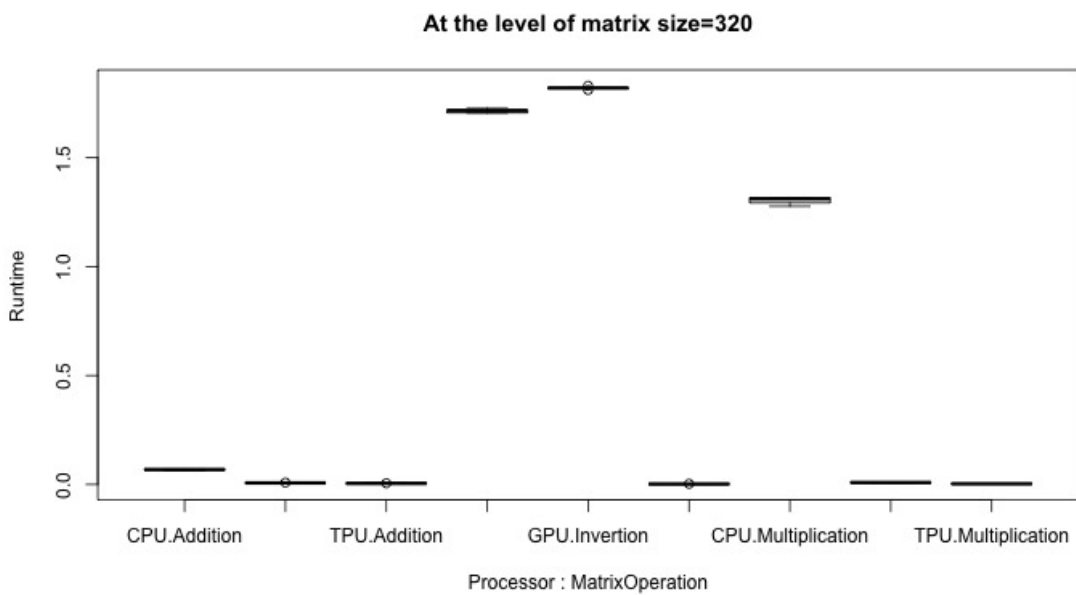


```
jpeg(filename = "../figs/interaction_TPU_size_time.jpeg", width = 600, height = 400, quality = 10000)
interaction.plot(TPUdata$MatrixSize, TPUdata$MatrixOperation, TPUdata$Runtime)
```

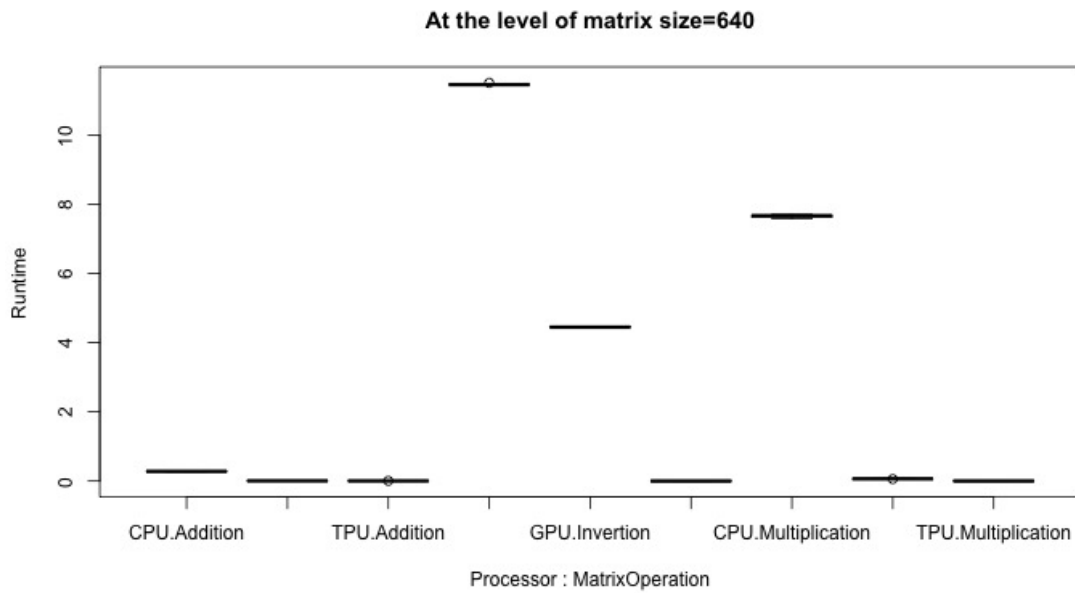


Boxplots

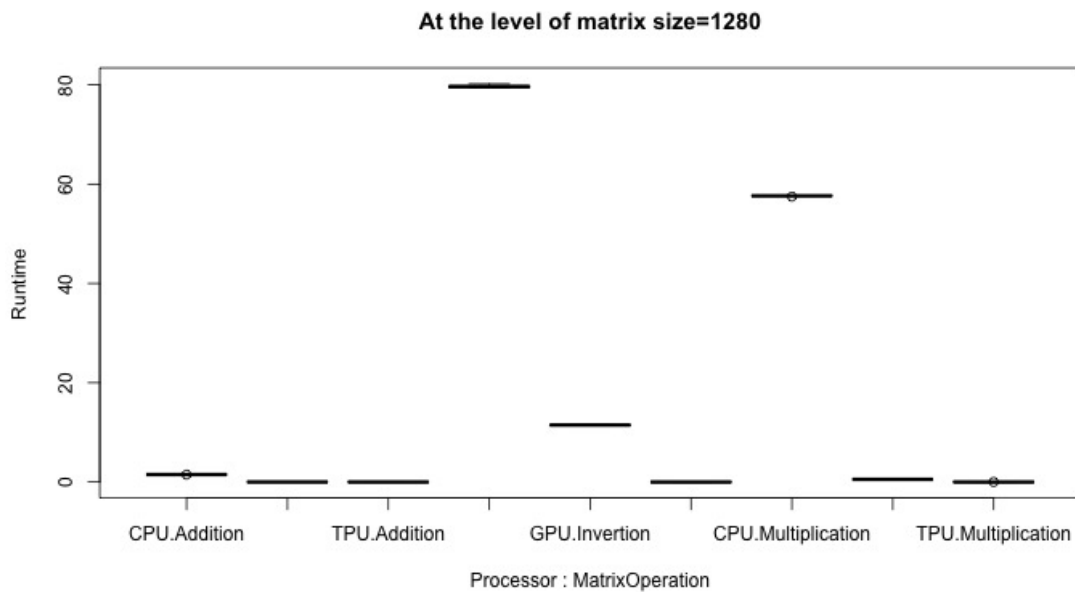
```
jpeg(filename = "../figs/Operation_vs_runtime_size320.jpeg", width = 700, height = 400, quality = 10000)
boxplot(Runtime~Processor*MatrixOperation, data = size320data, main="At the level of matrix size=320")
```



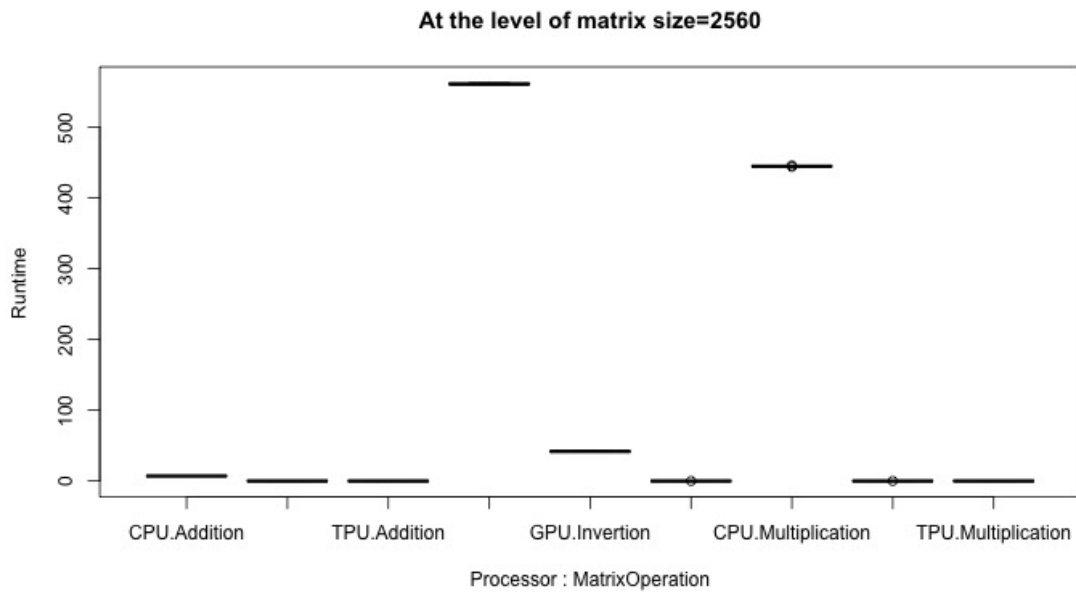

```
jpeg(filename = "../figs/Operation_vs_runtime_size640.jpeg", width = 700, height = 400, quality = 10000)
boxplot(Runtime~Processor*MatrixOperation, data = size640data, main="At the level of matrix size=640")
```



```
jpeg(filename = "../figs/Operation_vs_runtime_size1280.jpeg", width = 700, height = 400, quality = 10000)
boxplot(Runtime~Processor*MatrixOperation, data = size1280data, main="At the level of matrix size=1280")
```



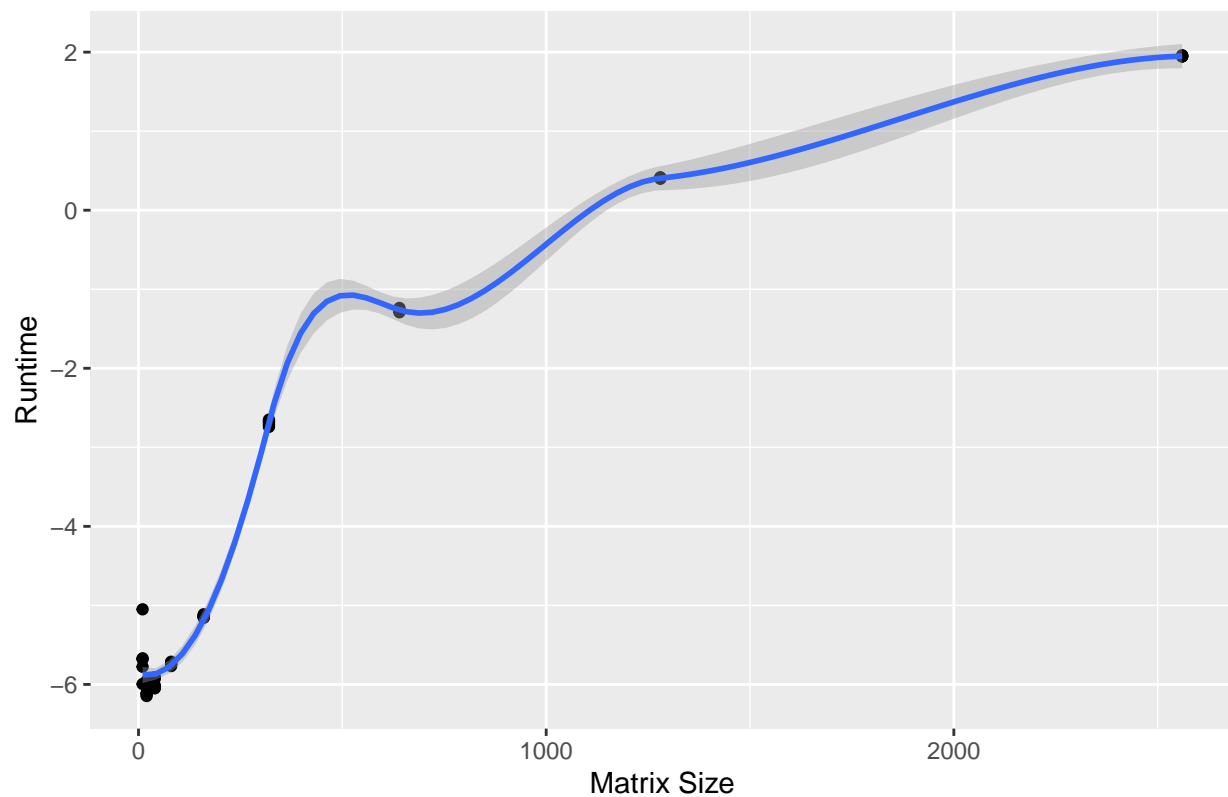
```
jpeg(filename = "../figs/Operation_vs_runtime_size2560.jpeg", width = 700, height = 400, quality = 10000)
boxplot(Runtime~Processor*MatrixOperation, data = size2560data, main="At the level of matrix size=2560")
```



At the level of matrix size=2560, avoid using CPU for inversion and multiplication because its run-times are much bigger.

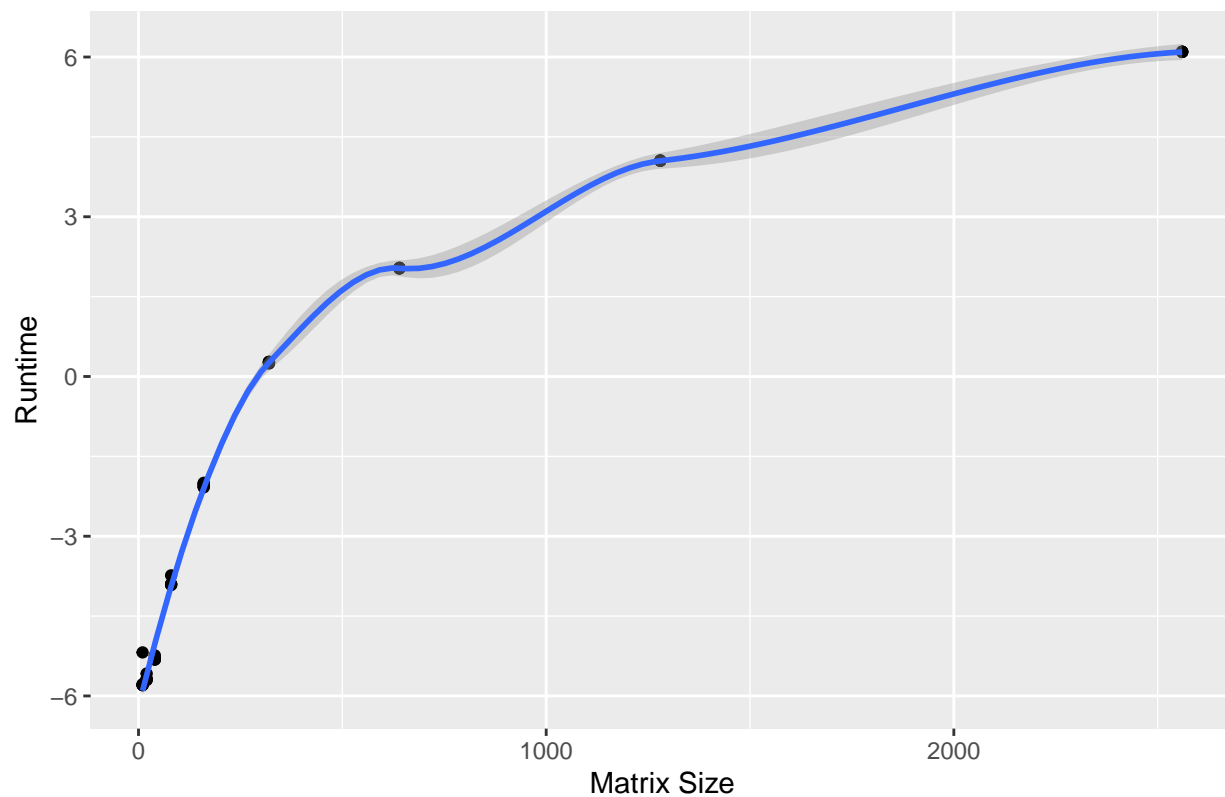
```
# Addition & CPU
data[data$Processor == "CPU" & data$MatrixOperation=="Addition",] %>%
ggplot() +
geom_point(mapping = aes(x = MatrixSize, y = log(Runtime))) +
geom_smooth(mapping = aes(x = MatrixSize, y = log(Runtime))) +
labs(title = "For CPU and Addition with Different Matrix Size", x = "Matrix Size", y = "Runtime")
```

For CPU and Addition with Different Matrix Size



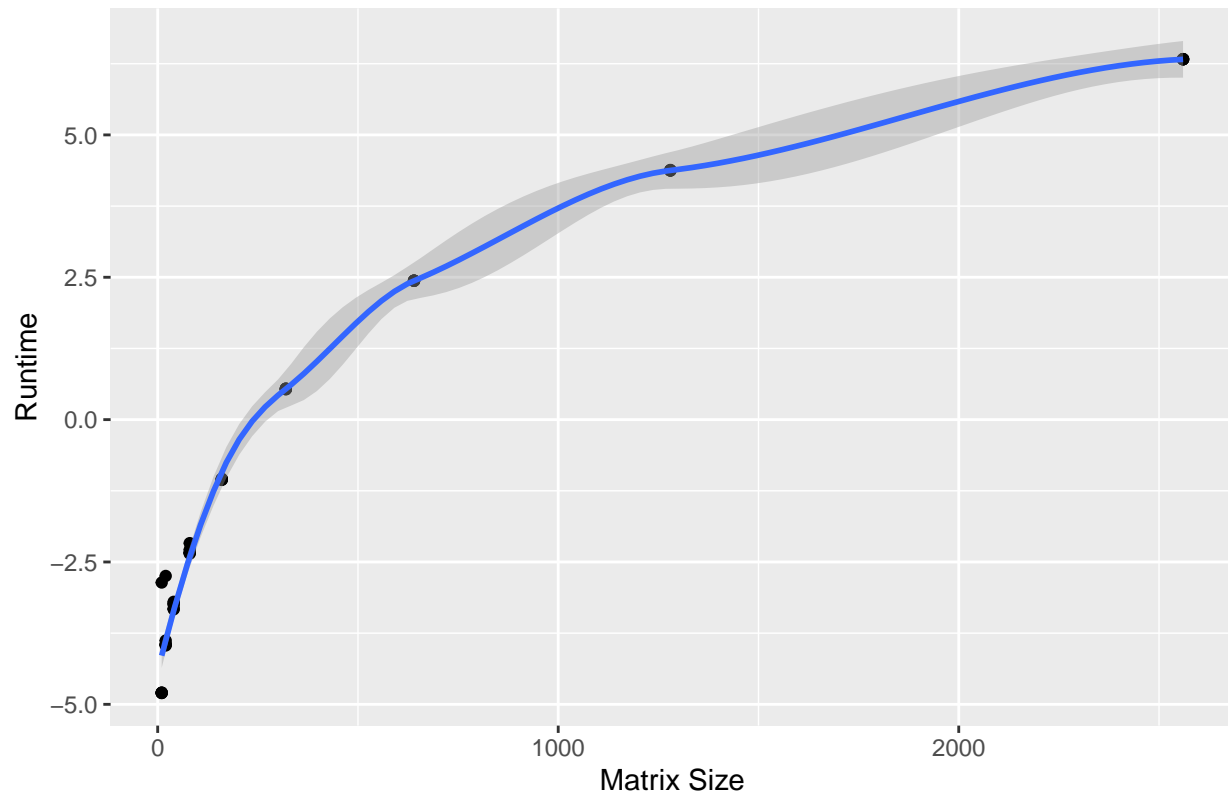
```
# Multiplication & CPU
data[data$Processor == "CPU" & data$MatrixOperation=="Multiplication",] %>%
ggplot() +
geom_point(mapping = aes(x = MatrixSize, y = log(Runtime))) +
geom_smooth(mapping = aes(x = MatrixSize, y = log(Runtime))) +
labs(title = "For CPU and Multiplication with Different Matrix Size", x = "Matrix Size", y = "Runtime")
```

For CPU and Multiplication with Different Matrix Size



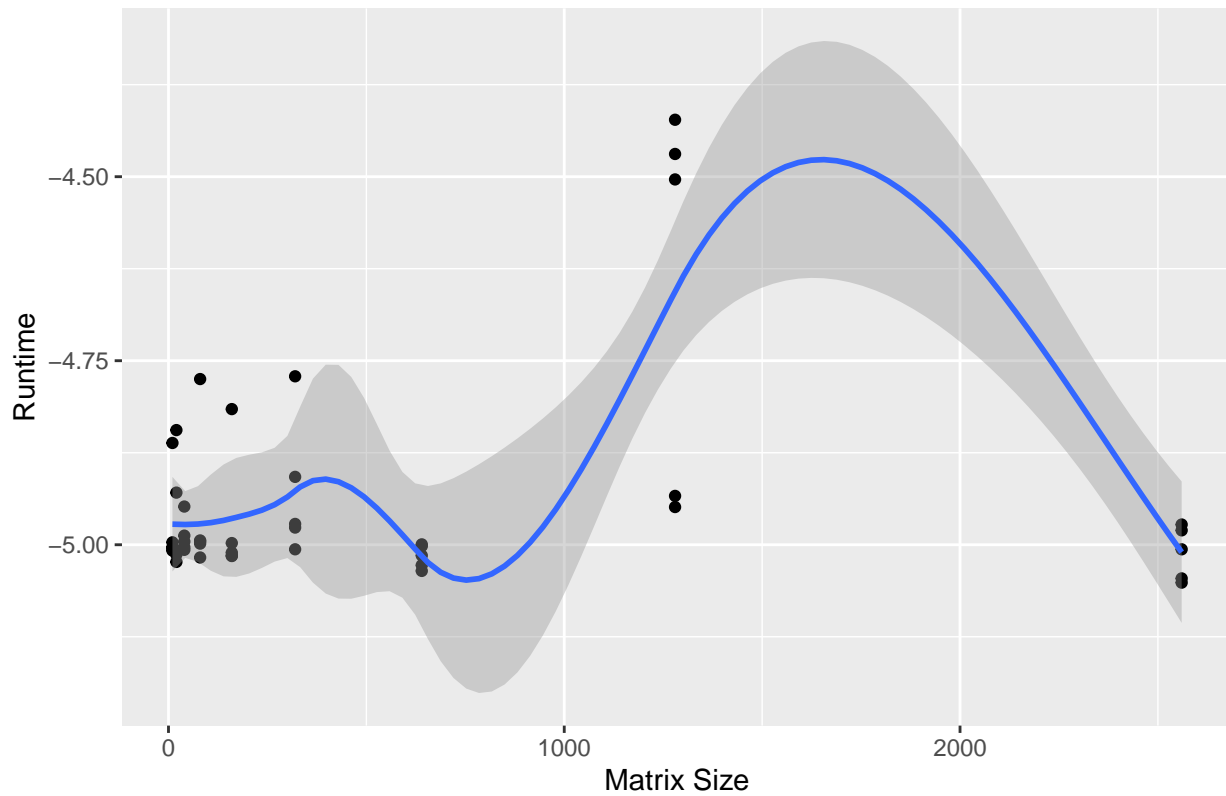
```
# Inversion & CPU
data[data$Processor == "CPU" & data$MatrixOperation=="Inversion",] %>%
ggplot() +
geom_point(mapping = aes(x = MatrixSize, y = log(Runtime))) +
geom_smooth(mapping = aes(x = MatrixSize, y = log(Runtime))) +
labs(title = "For CPU and Inversion with Different Matrix Size", x = "Matrix Size", y = "Runtime")
```

For CPU and Inversion with Different Matrix Size

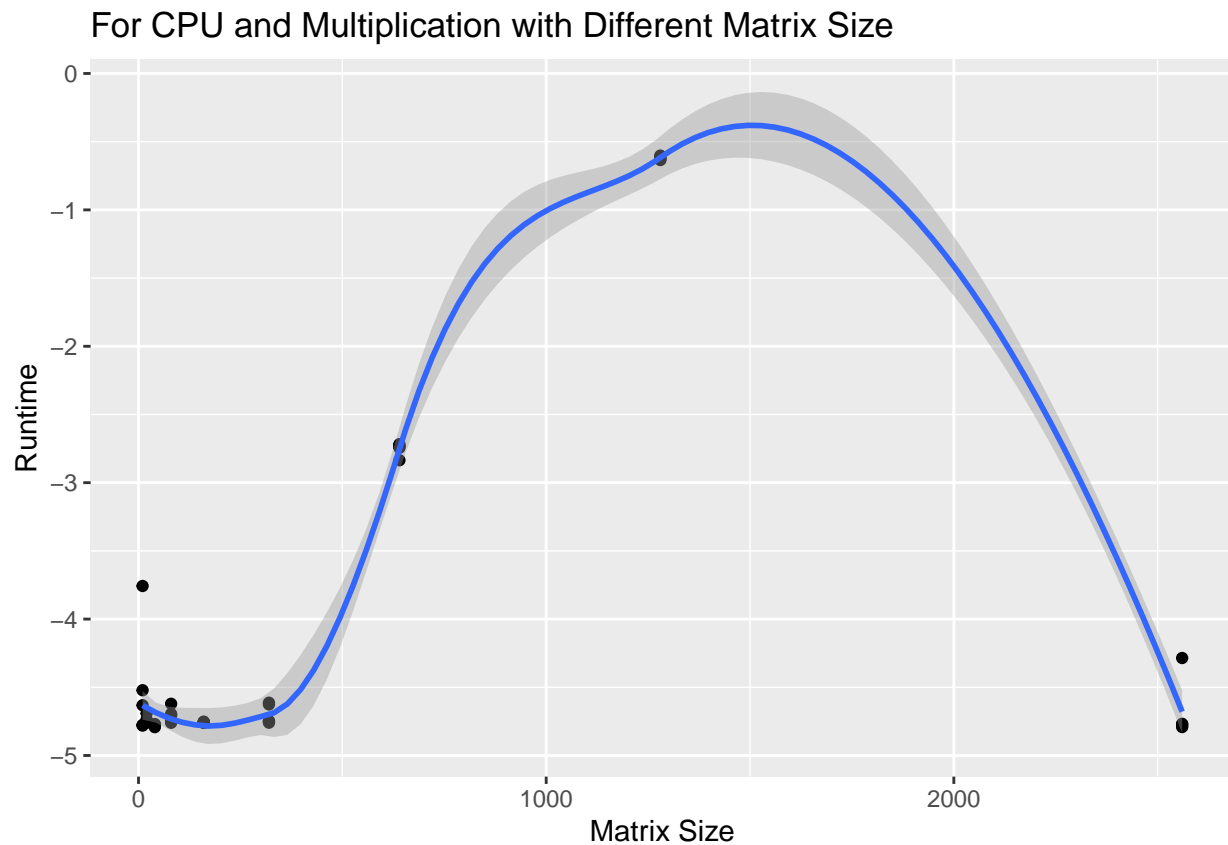


```
# Addition & GPU
data[data$Processor == "GPU" & data$MatrixOperation=="Addition",] %>%
ggplot() +
geom_point(mapping = aes(x = MatrixSize, y = log(Runtime))) +
geom_smooth(mapping = aes(x = MatrixSize, y = log(Runtime))) +
labs(title = "For GPU and Addition with Different Matrix Size", x = "Matrix Size", y = "Runtime")
```

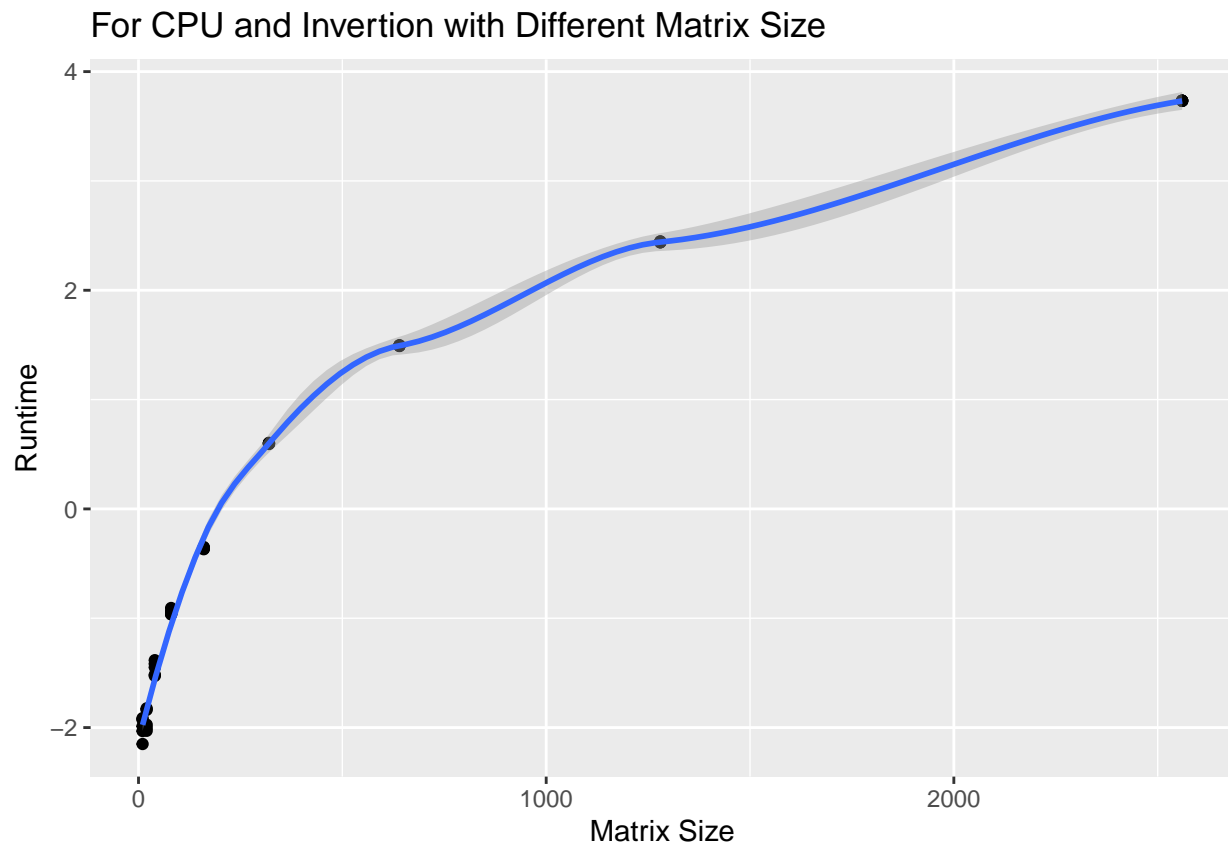
For GPU and Addition with Different Matrix Size



```
# Multiplication @ GPU
data[data$Processor == "GPU" & data$MatrixOperation=="Multiplication",] %>%
ggplot() +
geom_point(mapping = aes(x = MatrixSize, y = log(Runtime))) +
geom_smooth(mapping = aes(x = MatrixSize, y = log(Runtime))) +
labs(title = "For CPU and Multiplication with Different Matrix Size", x = "Matrix Size", y = "Runtime")
```

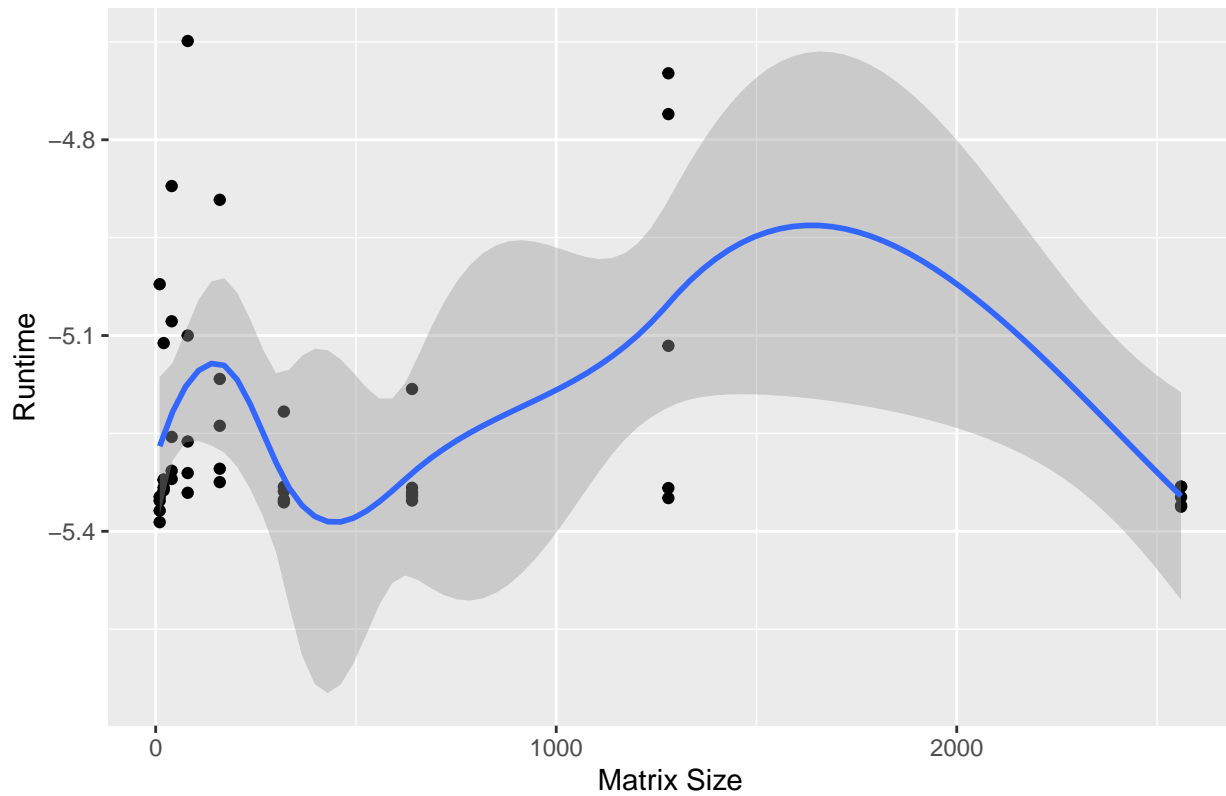


```
# Inversion & GPU
data[data$Processor == "GPU" & data$MatrixOperation=="Inversion",] %>%
ggplot() +
geom_point(mapping = aes(x = MatrixSize, y = log(Runtime))) +
geom_smooth(mapping = aes(x = MatrixSize, y = log(Runtime))) +
labs(title = "For CPU and Inversion with Different Matrix Size", x = "Matrix Size", y = "Runtime")
```



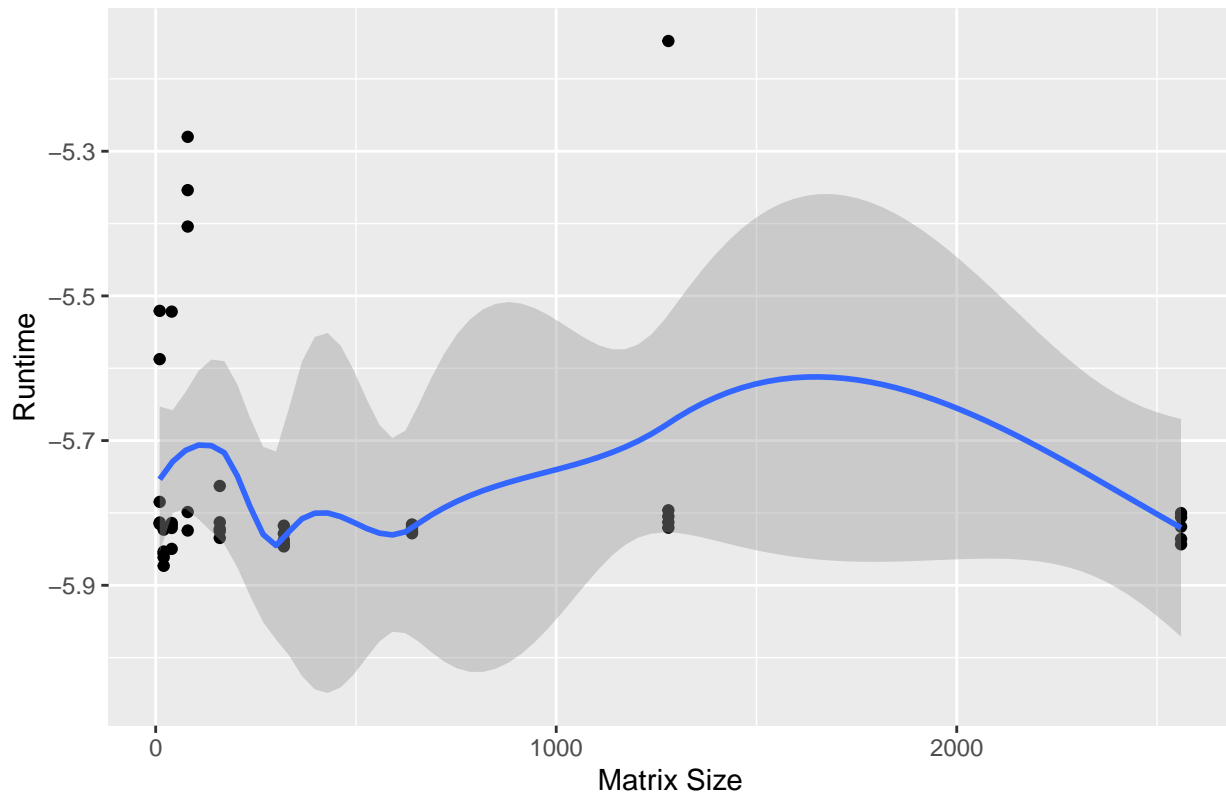
```
# Addition & TPU
data[data$Processor == "TPU" & data$MatrixOperation=="Addition",] %>%
ggplot() +
geom_point(mapping = aes(x = MatrixSize, y = log(Runtime))) +
geom_smooth(mapping = aes(x = MatrixSize, y = log(Runtime))) +
labs(title = "For TPU and Addition with Different Matrix Size", x = "Matrix Size", y = "Runtime")
```


For TPU and Addition with Different Matrix Size



```
# Multiplication & TPU
data[data$Processor == "TPU" & data$MatrixOperation=="Multiplication",] %>%
ggplot() +
geom_point(mapping = aes(x = MatrixSize, y = log(Runtime))) +
geom_smooth(mapping = aes(x = MatrixSize, y = log(Runtime))) +
labs(title = "For TPU and Multiplication with Different Matrix Size", x = "Matrix Size", y = "Runtime")
```

For TPU and Multiplication with Different Matrix Size



```
# Inversion & TPU
data[data$Processor == "TPU" & data$MatrixOperation=="Inversion",] %>%
ggplot() +
geom_point(mapping = aes(x = MatrixSize, y = log(Runtime))) +
geom_smooth(mapping = aes(x = MatrixSize, y = log(Runtime))) +
labs(title = "For TPU and Inversion with Different Matrix Size", x = "Matrix Size", y = "Runtime")
```

For TPU and Inversion with Different Matrix Size

