# Onchain benchmarks

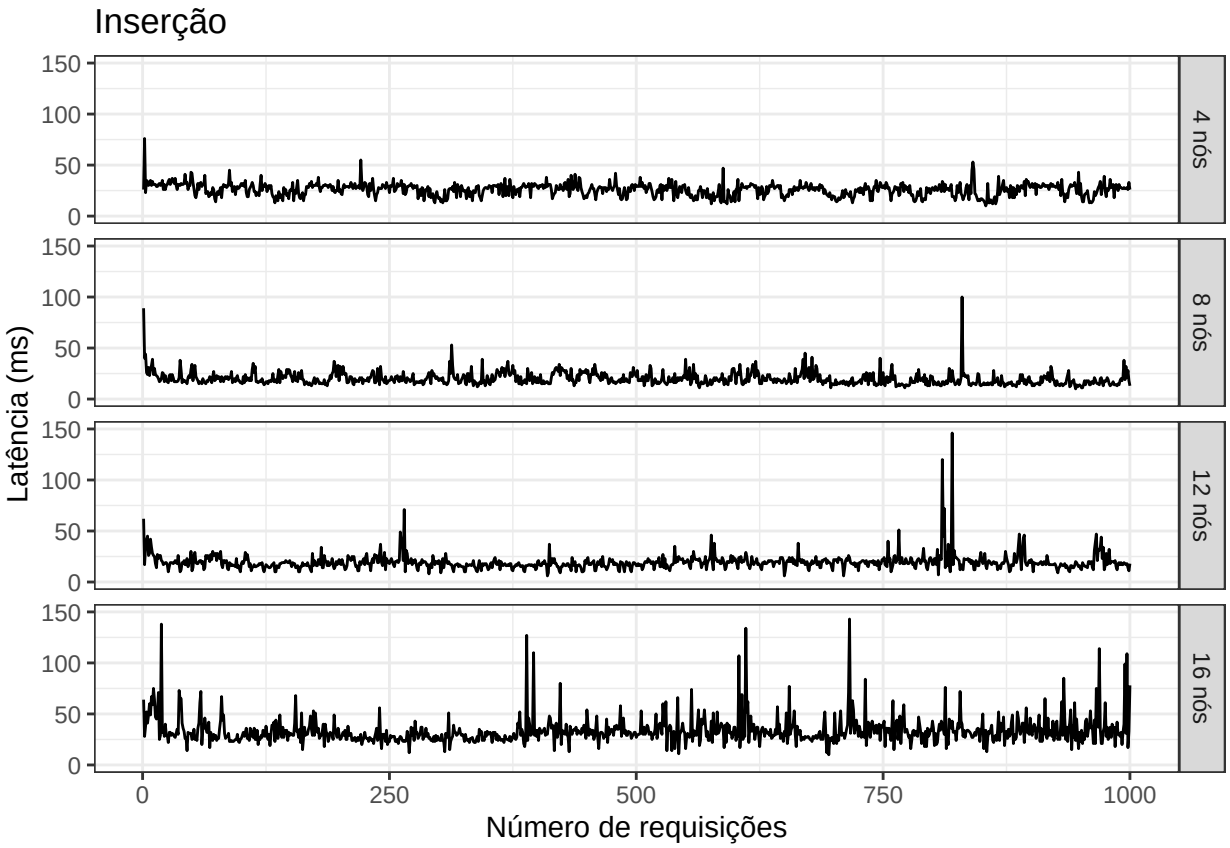
## leitura de todos os datasets para onchain e setup

## Grafico comparativo entre as latencias por grupo de nós.

- Avaliar se é necessario utilizar talvez algum outro tipo de grafico como em linha

## Latencia para envio:

```
# length(benchSend4nodes$Latency)  # e.g., 1000
# length(benchSend8nodes$Latency)  # e.g., 1000
# length(benchSend12nodes$Latency) # e.g., 1000
# length(benchSend16nodes$Latency) # e.g., 1000

df_plotter <- data.frame(
  time = rep(1:1000, times = 4),
  latency = c(benchSend4nodes$Latency, benchSend8nodes$Latency, benchSend12nodes$Latency, benchSend16no
  scenario = rep(c("4 nós", "8 nós", "12 nós", "16 nós"), each = 1000)
)

df_plotter$scenario <- factor(df_plotter$scenario, levels = c("4 nós", "8 nós", "12 nós", "16 nós"))

df_plotter <- df_plotter %>%
  mutate(case = "Envio")

p <- ggplot(df_plotter, aes(x = time, y = latency)) +
  # geom_point() +
  geom_line() +
  facet_grid(rows = vars(scenario)) +
  labs(title = "Inserção",
      x = "Número de requisições",
       y = "Latência (ms)") +
  theme_bw() +
  coord_cartesian(ylim = c(0, 150))

print(p)
```

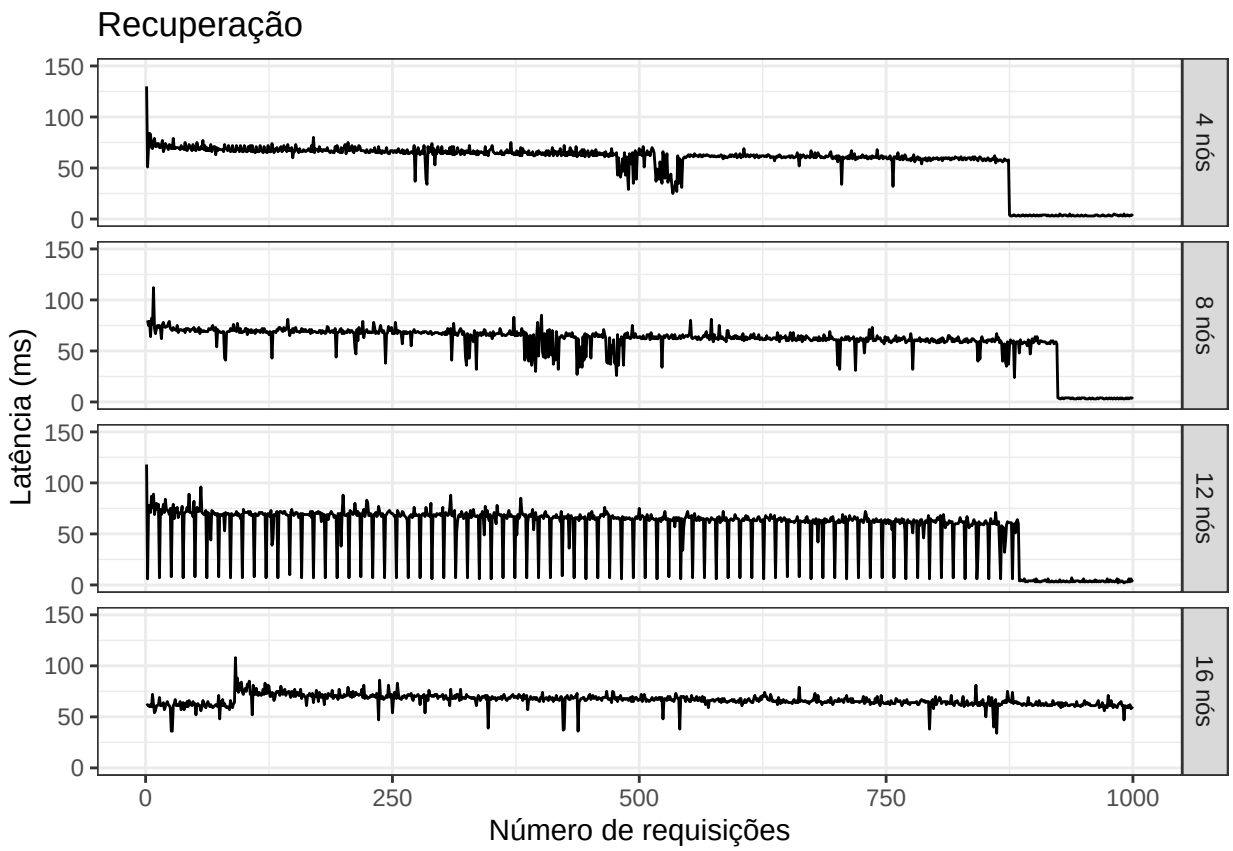



```
# ggsave("send-nodes-latency-cmp.png")
```

## Latência para recuperação:

```
# length(benchRecv4nodes$Latency)  # e.g., 1000
# length(benchRecv8nodes$Latency)  # e.g., 1000
# length(benchRecv12nodes$Latency) # e.g., 1000
# length(benchRecv16nodes$Latency) # e.g., 1000


df_plotter2 <- data.frame(
  time = rep(1:1000, times = 4),
  latency = c(benchRecv4nodes$Latency, benchRecv8nodes$Latency, benchRecv12nodes$Latency, benchRecv16no
  scenario = rep(c("4 nós", "8 nós", "12 nós", "16 nós"), each = 1000)
)

df_plotter2$scenario <- factor(df_plotter$scenario, levels = c("4 nós", "8 nós", "12 nós", "16 nós"))

df_plotter2 <- df_plotter2 %>%
  mutate(case = "Recuperação")


ggplot(df_plotter2, aes(x = time, y = latency)) +
  # geom_point() +
  geom_line() +
  facet_grid(rows = vars(scenario)) +
```

```
  labs(
      title = "Recuperação",
      x = "Número de requisições",
       y = "Latência (ms)") +
  theme_bw() +
  coord_cartesian(ylim = c(0, 150))
```

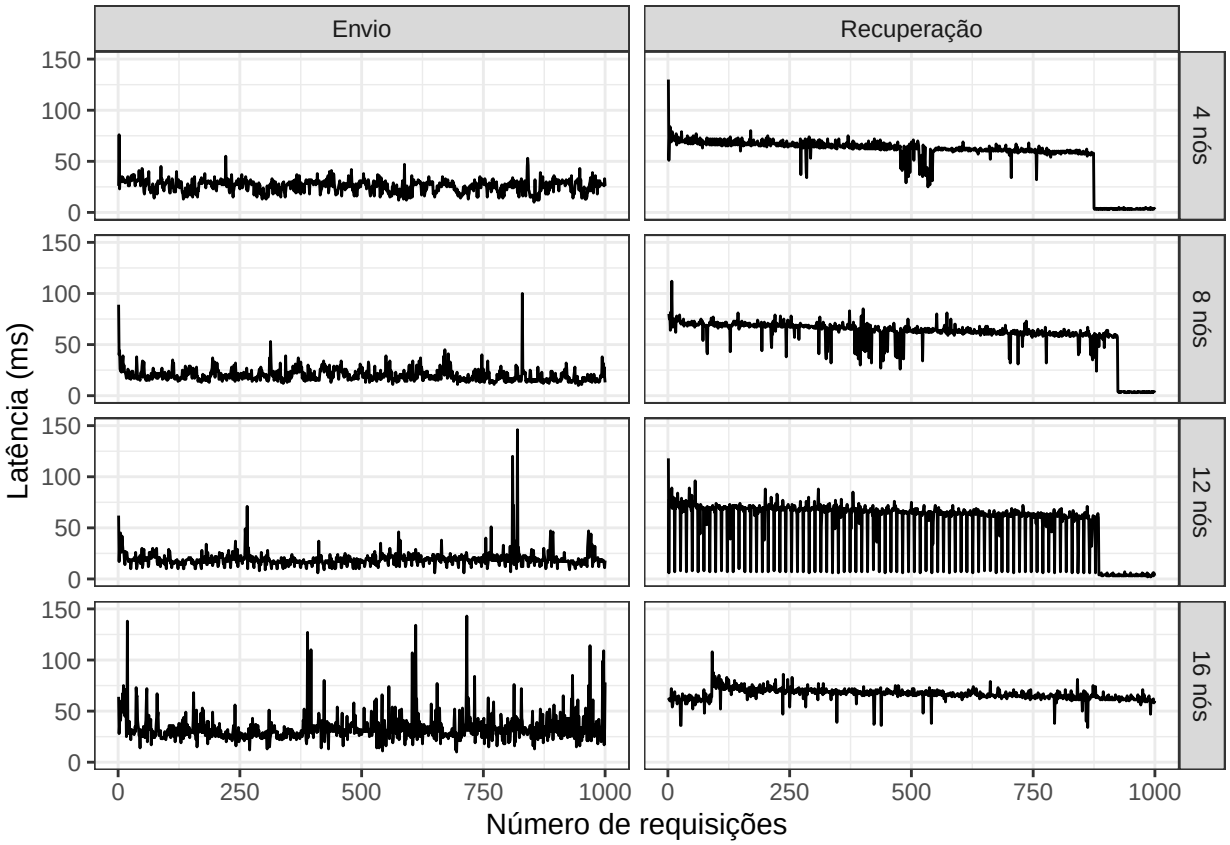



```
# ggsave('./recv-nodes-latency-cmp.png')
```

## Gráfico agrupado para onchain

```
df_plotter1 <- df_plotter

combined_df <- rbind(df_plotter1, df_plotter2)


# Tentativa de fazer um gráfico agrupado

p <- ggplot(combined_df, aes(x = time, y = latency)) +
  # geom_point() +
  geom_line() +
  facet_grid(scenario ~ case) +
  labs(x = "Número de requisições",
       y = "Latência (ms)") +
  theme_bw() +
```

```
  coord_cartesian(ylim = c(0, 150))

print(p)
```

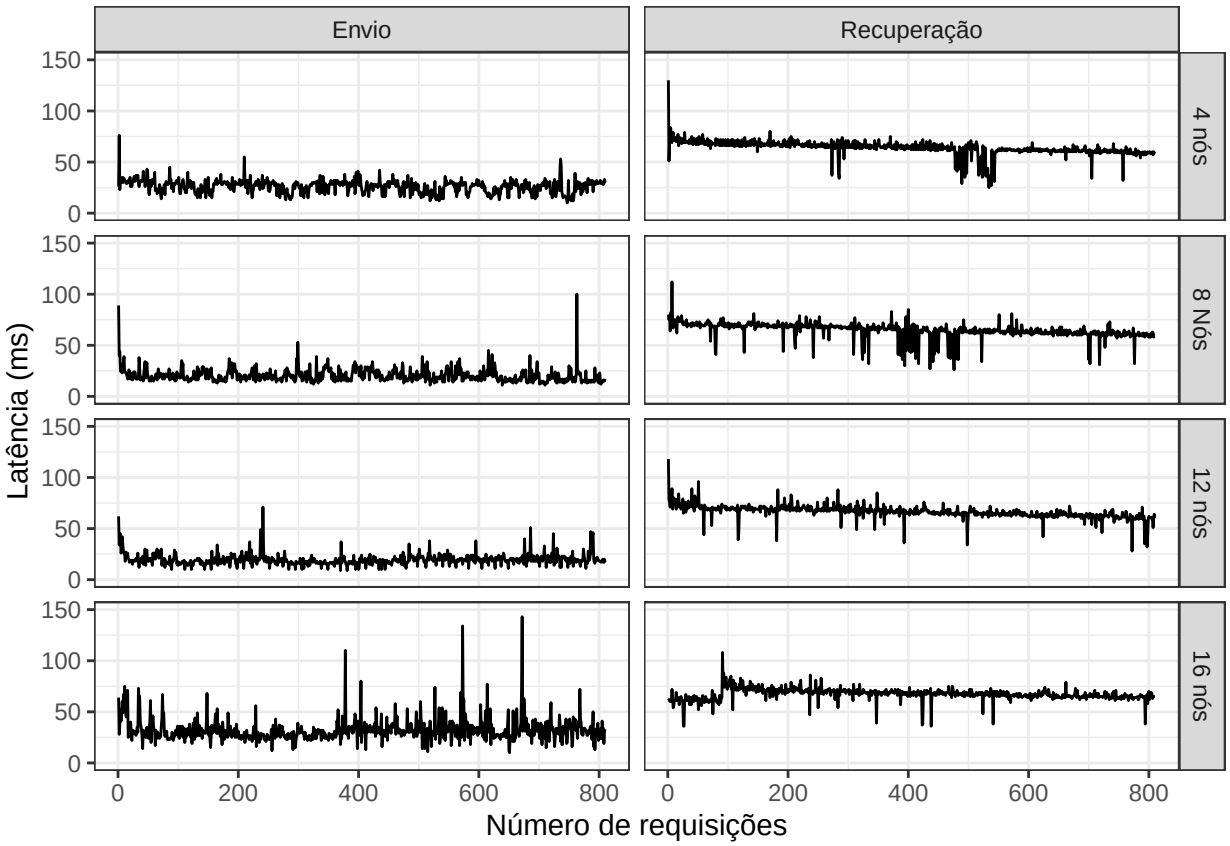



```
# ggsave('./figures/onchain-group-1k.png')
```

## Sem erros nos requests, apenas respondeCode 200 (OK)

```
# Read: bs4n - bench send 4..16 nodes
bs4n <- subset(benchSend4nodes, responseCode == 200)
bs8n <- subset(benchSend8nodes, responseCode == 200)
bs12n <- subset(benchSend12nodes, responseCode == 200)
bs16n <- subset(benchSend16nodes, responseCode == 200)
# Read: bs4n - bench recv 4..16 nodes
br4n <- subset(benchRecv4nodes, responseCode == 200)
br8n <- subset(benchRecv8nodes, responseCode == 200)
br12n <- subset(benchRecv12nodes, responseCode == 200)
br16n <- subset(benchRecv16nodes, responseCode == 200)


min_rows <- min(
  nrow(br4n), nrow(br8n),
  nrow(br12n), nrow(br16n),
  nrow(bs4n), nrow(bs8n),
  nrow(bs12n), nrow(bs16n)
```

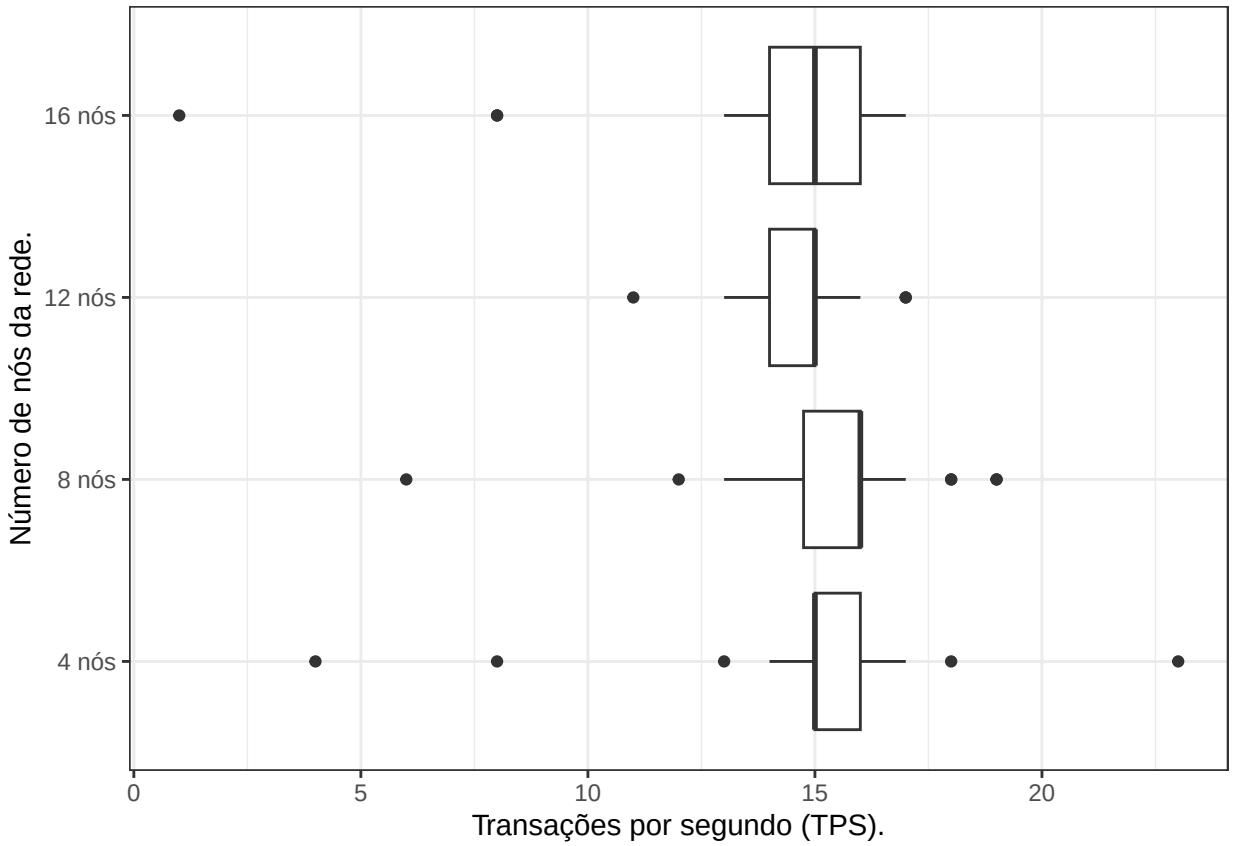
```
)

# Truncate all Latency columns to the minimum number of rows
nbr4n <- br4n[1:min_rows, ]
nbr8n <- br8n[1:min_rows, ]
nbr12n <- br12n[1:min_rows, ]
nbr16n <- br16n[1:min_rows, ]
nbs4n <- bs4n[1:min_rows, ]
nbs8n <- bs8n[1:min_rows, ]
nbs12n <- bs12n[1:min_rows, ]
nbs16n <- bs16n[1:min_rows, ]

df_plotter1 <- data.frame(
  time = rep(1:min_rows, times = 4),
  latency = c(nbs4n$Latency, nbs8n$Latency, nbs12n$Latency, nbs16n$Latency),
  scenario = rep(c("4 nós", "8 Nós", "12 nós", "16 nós"), each = min_rows),
  case = rep("Envio", each = min_rows)
)

df_plotter1$scenario <- factor(df_plotter1$scenario, levels = c("4 nós", "8 Nós", "12 nós", "16 nós"))


df_plotter2 <- data.frame(
  time = rep(1:min_rows, times = 4),
  latency = c(nbr4n$Latency, nbr8n$Latency, nbr12n$Latency, nbr16n$Latency),
  scenario = rep(c("4 nós", "8 Nós", "12 nós", "16 nós"), each = min_rows),
  case = rep("Recuperação", each = min_rows)
)

df_plotter2$scenario <- factor(df_plotter2$scenario, levels = c("4 nós", "8 Nós", "12 nós", "16 nós"))


combined_df <- rbind(df_plotter1, df_plotter2)


# Tentativa de fazer um gráfico agrupado

p <- ggplot(combined_df, aes(x = time, y = latency)) +
  # geom_point() +
  geom_line() +
  facet_grid(scenario ~ case) +
  labs(x = "Número de requisições",
       y = "Latência (ms)") +
  theme_bw() +
  coord_cartesian(ylim = c(0, 150))

print(p)
```




```
#ggsave('./figures/onchain-group-1k-cleaned.png')
```

## Gráfico para request/s realizados RECUPERAÇÃO;

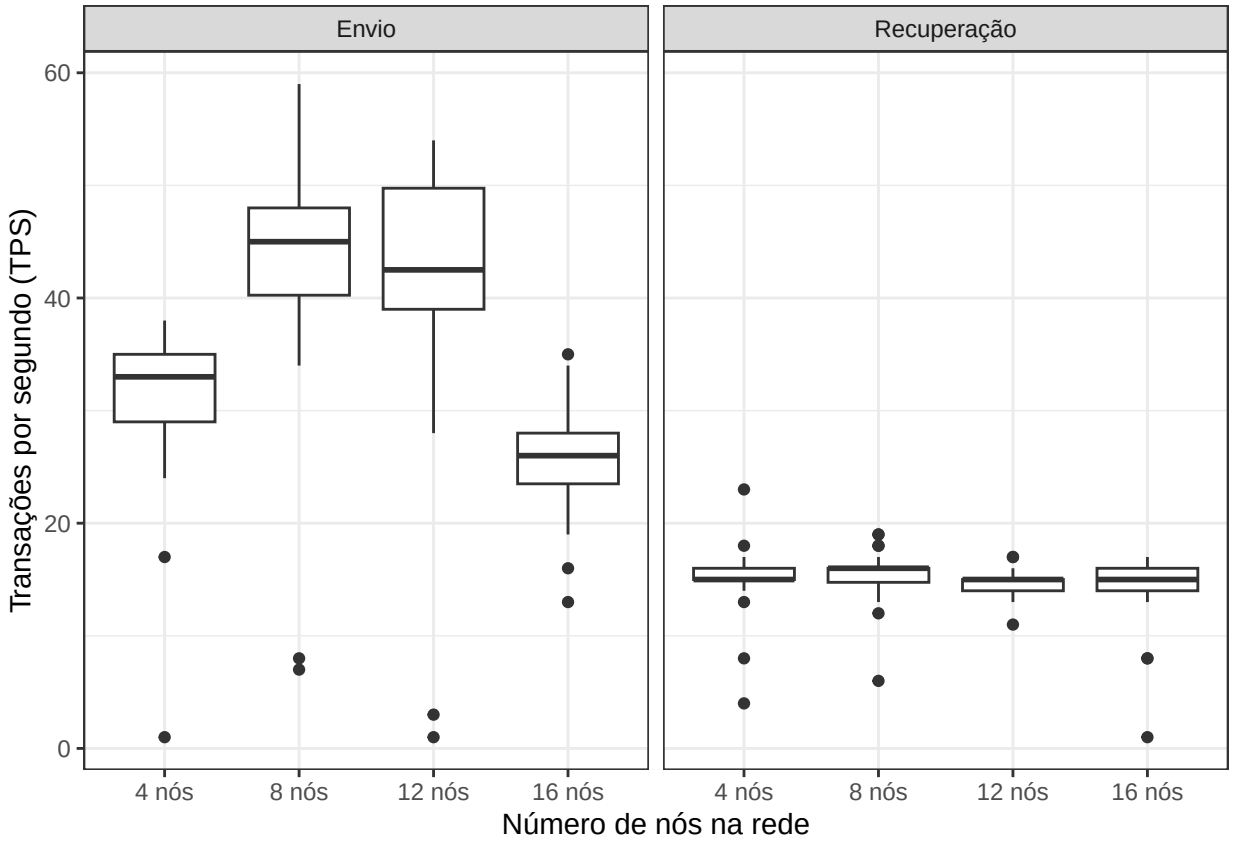
```
tpsbr4n <- getTps(br4n)
tpsbr8n <- getTps(br8n)
tpsbr12n <- getTps(br12n)
tpsbr16n <- getTps(br16n)

df_tps_recv_onchain <- data.frame(
  tps = c(tpsbr4n$transactions, tpsbr8n$transactions, tpsbr12n$transactions, tpsbr16n$transactions),
  scenario = rep(c("4 nós", "8 nós", "12 nós", "16 nós"), time = c(length(tpsbr4n$transactions), length
  )


df_tps_recv_onchain$scenario <- factor(df_tps_recv_onchain$scenario, levels = c("4 nós", "8 nós", "12 n


p <- ggplot(df_tps_recv_onchain, aes(x=tps, y=scenario)) +
    theme_bw()+
    labs(x = "Transações por segundo (TPS).", y = "Número de nós da rede.") +
    geom_boxplot()

print(p)
```




## Gráfico para request/s realizados INSERÇÃO (apenas requests com sucesso);

```
# Read: bs4n - bench send 4..16 nodes
bs4n <- subset(benchSend4nodes, responseCode == 200)
bs8n <- subset(benchSend8nodes, responseCode == 200)
bs12n <- subset(benchSend12nodes, responseCode == 200)
bs16n <- subset(benchSend16nodes, responseCode == 200)
# Read: bs4n - bench recv 4..16 nodes
br4n <- subset(benchRecv4nodes, responseCode == 200)
br8n <- subset(benchRecv8nodes, responseCode == 200)
br12n <- subset(benchRecv12nodes, responseCode == 200)
br16n <- subset(benchRecv16nodes, responseCode == 200)


tpsbs4n <- getTps(bs4n)
tpsbs8n <- getTps(bs8n)
tpsbs12n <- getTps(bs12n)
tpsbs16n <- getTps(bs16n)


df_tps_send_onchain <- data.frame(
  tps = c(tpsbs4n$transactions, tpsbs8n$transactions, tpsbs12n$transactions, tpsbs16n$transactions),
  scenario = rep(c("4 nós", "8 nós", "12 nós", "16 nós"), time = c(length(tpsbs4n$transactions), length
  )
```

```r
df_tps_send_onchain$scenario <- factor(df_tps_send_onchain$scenario, levels = c("4 nós", "8 nós", "12 nó

merged_df <- rbind(df_tps_recv_onchain, df_tps_send_onchain)

df_tps_send_onchain <- df_tps_send_onchain %>%
  mutate(case = "Envio")

df_tps_recv_onchain <- df_tps_recv_onchain %>%
  mutate(case = "Recuperação")

merged_df_onchain <- rbind(df_tps_recv_onchain, df_tps_send_onchain)

ggplot(merged_df_onchain, aes(x = scenario, y = tps)) +
  geom_boxplot() +
  labs(x = "Número de nós na rede",
       y = "Transações por segundo (TPS)") +
  facet_grid(~case) +
  theme_bw()
```




```r
# ggsave('./figures/onchain-tps-grouped.png')
```

## Uso de CPU / ram / disco para ENVIO

```r
# Alguns problemas com undefined não sendo transformado em NA, força via as.integer
metricSend12nodes$value <- as.integer(metricSend12nodes$value)
```

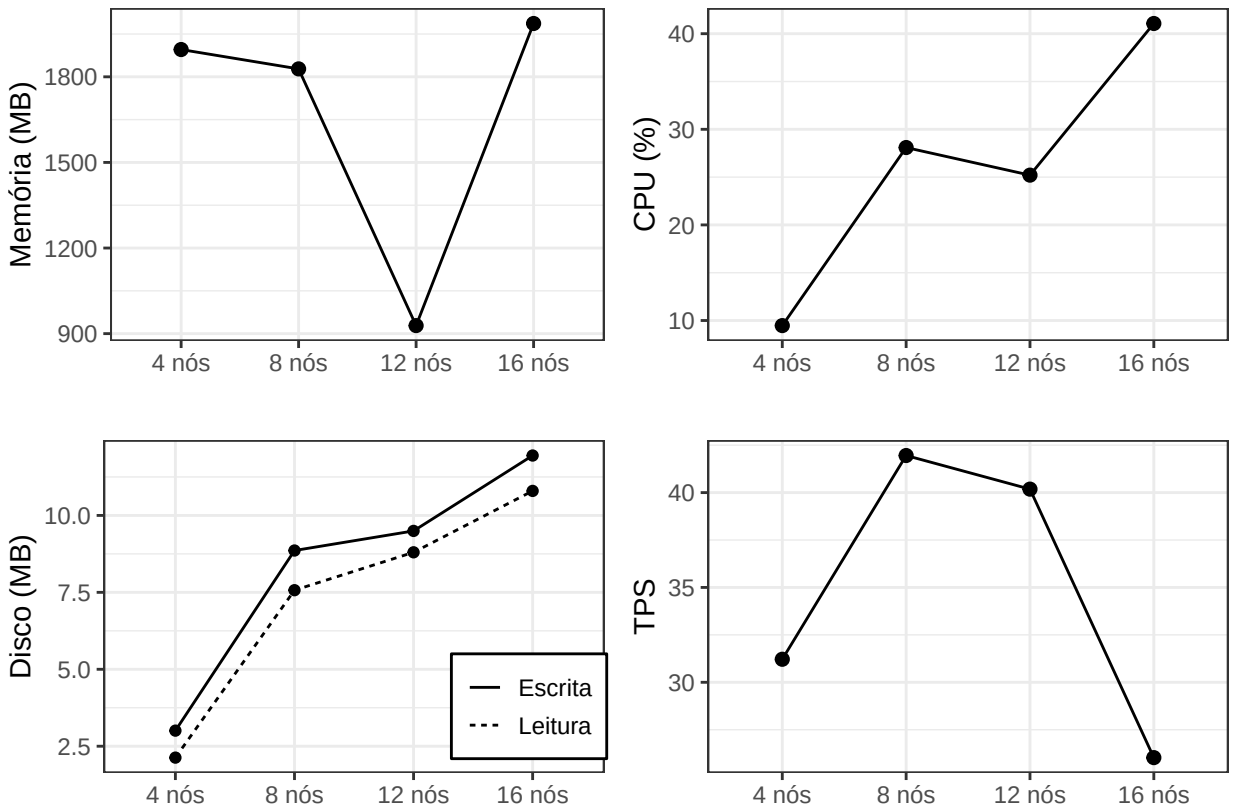
```
## Warning: NAs introduced by coercion
```

```
## Warning: NAs introduced by coercion to integer range
```

```
ams4n <- getAvgOnchain(metricSend4nodes)
ams8n <- getAvgOnchain(metricSend8nodes)
ams12n <- getAvgOnchain(metricSend12nodes)
ams16n <- getAvgOnchain(metricSend16nodes)

ams4n <- ams4n %>%
  mutate(scenario = "4 nós")
ams8n <- ams8n %>%
  mutate(scenario = "8 nós")
ams12n <- ams12n %>%
  mutate(scenario = "12 nós")
ams16n <- ams16n %>%
  mutate(scenario = "16 nós")

df_metrics_send_join <- rbind(ams4n, ams8n, ams12n, ams16n)

library(ggplot2)

# Reorder levels of scenario
df_metrics_send_join$scenario <- factor(df_metrics_send_join$scenario, levels = c("4 nós", "8 nós", "12

# Plot for system_memory_used
plot_memory <- ggplot(df_metrics_send_join[df_metrics_send_join$name == "system_memory_used", ], aes(x =
  geom_line(aes(group = name)) +
  geom_point(size = 2) +
  labs(y = "Memória (MB)", x = "") +
  theme_bw()

disk_data <- df_metrics_send_join %>%
  filter(name %in% c("system_disk_readbytes", "system_disk_writebytes")) %>%
  mutate(name = recode(name,
                       "system_disk_readbytes" = "Leitura",
                       "system_disk_writebytes" = "Escrita"))

plot_disk_rw <- ggplot(disk_data, aes(x = scenario, y = avg_value, linetype = name)) +
  geom_line(aes(group = name)) +
  geom_point() +
  labs(y = "Disco (MB)", linetype = "Metric", x = "") +
  theme_bw() +
  guides(linetype = guide_legend(title = NULL)) +
  theme(legend.position = c(0.85, 0.2),
        legend.key.size = unit(0.5, "cm"),
        legend.background = element_rect(fill = "white", color = "black"))
```

```
## Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
## 3.5.0.
## i Please use the `legend.position.inside` argument of `theme()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
# Plot for system_cpu_sysload
plot_cpu_sysload <- ggplot(df_metrics_send_join[df_metrics_send_join$name == "system_cpu_sysload", ], ae
  geom_line(aes(group = name)) +
  geom_point(size = 2) +
  labs(y = "CPU (%)", x = "") +
  theme_bw()


df_avg_tps_send_onchain <- df_tps_send_onchain %>%
    group_by(scenario) %>%
    summarise(avg_value = mean(tps, na.rm = TRUE))

df_avg_tps_send_onchain <- df_avg_tps_send_onchain %>%
    mutate(type = "send")

plot_tps <- ggplot(df_avg_tps_send_onchain, aes(x = scenario, y = avg_value)) +
  geom_line(aes(group = type)) +
  geom_point(size = 2) +
  labs(y = "TPS", x = "") +
  theme_bw()


# Arrange plots

g <- grid.arrange(plot_memory, plot_cpu_sysload, plot_disk_rw, plot_tps, nrow = 2, ncol = 2)
```



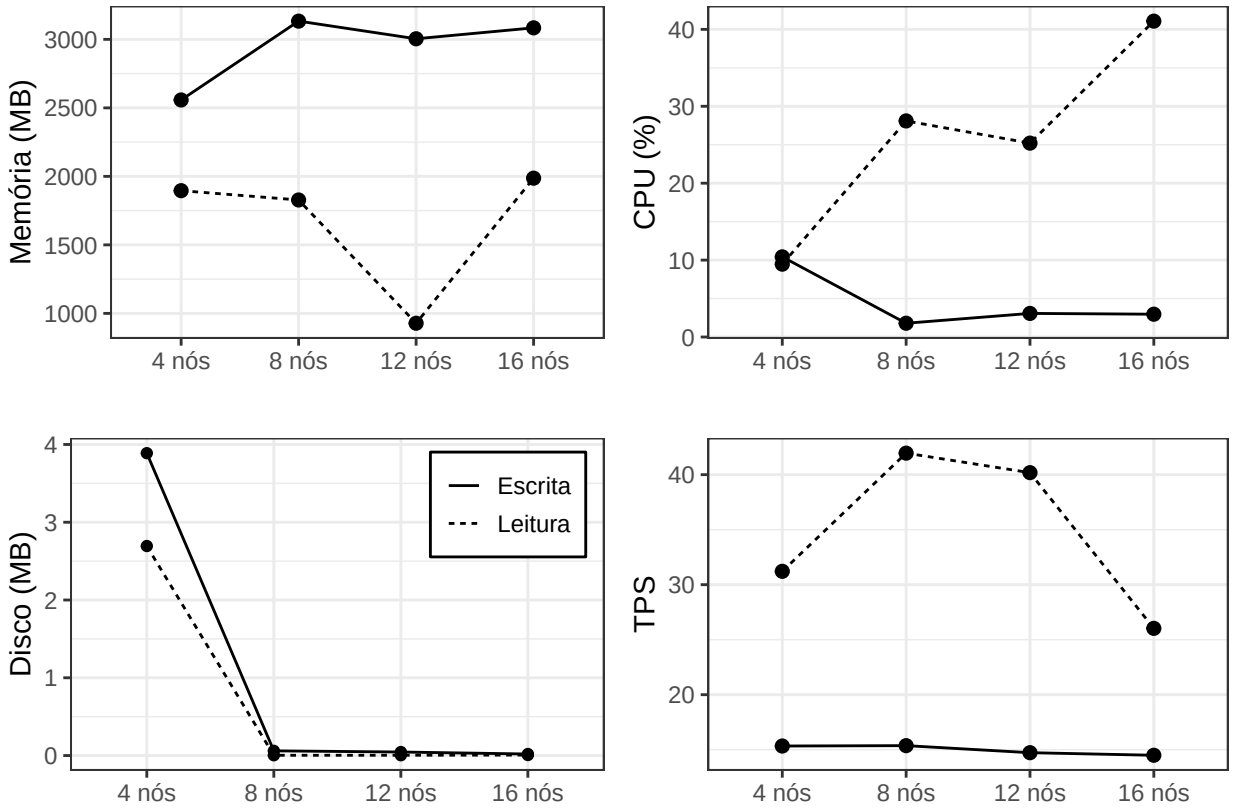
```
#ggsave('./figures/onchain-review-metrics-send.png', g)
```

## Onchain metricas agrupadas send e recv

```
# amr4n <- metricRecv4nodes %>%
#   group_by(name) %>%
#   summarise(avg_value = mean(value, na.rm = TRUE))
#
# amr8n <- metricRecv8nodes %>%
#   group_by(name) %>%
#   summarise(avg_value = mean(value, na.rm = TRUE))
#
# amr12n <- metricRecv12nodes %>%
#   group_by(name) %>%
#   summarise(avg_value = mean(value, na.rm = TRUE))
#
# amr16n <- metricRecv16nodes %>%
#   group_by(name) %>%
#   summarise(avg_value = mean(value, na.rm = TRUE))

amr4n <- getAvgOnchain(metricRecv4nodes)
amr8n <- getAvgOnchain(metricRecv8nodes)
amr12n <- getAvgOnchain(metricRecv12nodes)
amr16n <- getAvgOnchain(metricRecv16nodes)

amr4n <- amr4n %>%
  mutate(scenario = "4 nós")
amr8n <- amr8n %>%
  mutate(scenario = "8 nós")
amr12n <- amr12n %>%
  mutate(scenario = "12 nós")
amr16n <- amr16n %>%
  mutate(scenario = "16 nós")

df_metrics_recv_join <- rbind(amr4n, amr8n, amr12n, amr16n)
df_metrics_recv_join$scenario <- factor(df_metrics_recv_join$scenario, levels = c("4 nós", "8 nós", "12
df_metrics_send_join$scenario <- factor(df_metrics_recv_join$scenario, levels = c("4 nós", "8 nós", "12


df_avg_tps_recv_onchain <- df_tps_recv_onchain %>%
    group_by(scenario) %>%
    summarise(avg_value = mean(tps, na.rm = TRUE))

df_avg_tps_recv_onchain <- df_avg_tps_recv_onchain %>%
    mutate(type = "recv")

df_metrics_recv_join <- df_metrics_recv_join %>%
  mutate(type = "recv")

df_metrics_send_join <- df_metrics_send_join %>%
  mutate(type = "send")
```

```r
merge_df_metrics <- rbind(df_metrics_recv_join, df_metrics_send_join)

# Remove as métricas de disco
merge_df_metrics <- merge_df_metrics %>%
  filter(!(name %in% c('system_disk_readbytes', 'system_disk_writebytes')))


disk_data1 <- df_metrics_recv_join %>%
  filter(name %in% c("system_disk_readbytes", "system_disk_writebytes")) %>%
  mutate(name = recode(name,
                       "system_disk_readbytes" = "Leitura",
                       "system_disk_writebytes" = "Escrita"))

plot_disk_rw1 <- ggplot(disk_data1, aes(x = scenario, y = avg_value, linetype = name)) +
  geom_line(aes(group = name)) +
  geom_point() +
  labs(y = "Disco (MB)", linetype = "Metric", x = "") +
  theme_bw() +
  guides(linetype = guide_legend(title = NULL)) +
  theme(legend.position = c(0.82, 0.8),
        legend.key.size = unit(0.5, "cm"),
        legend.background = element_rect(fill = "white", color = "black"))


# Plot for system_memory_used
plot_memory1 <- ggplot(
  merge_df_metrics[merge_df_metrics$name == "system_memory_used", ],
  aes(x = scenario, y = avg_value, linetype = type)) +
  geom_line(aes(group = type)) +
  geom_point(size = 2) +
  labs(y = "Memória (MB)", x = "") +
  theme_bw() +
  theme(legend.position = "none")

# Plot for system_cpu_sysload
plot_cpu_sysload1 <- ggplot(merge_df_metrics[merge_df_metrics$name == "system_cpu_sysload", ], aes(x = s
  geom_line(aes(group = type)) +
  geom_point(size = 2) +
  labs(y = "CPU (%)", x = "") +
  theme_bw() +
  theme(legend.position = "none")

df_avg_tps_send_onchain <- df_tps_send_onchain %>%
    group_by(scenario) %>%
    summarise(avg_value = mean(tps, na.rm = TRUE))

df_avg_tps_send_onchain <- df_avg_tps_send_onchain %>%
    mutate(type = "send")

df_avg_tps_recv_onchain <- df_avg_tps_recv_onchain %>%
    mutate(type = "recv")

tps_joined <- rbind(df_avg_tps_recv_onchain, df_avg_tps_send_onchain)
```

```
plot_tps1 <- ggplot(tps_joined, aes(x = scenario, y = avg_value, linetype = type)) +
  geom_line(aes(group = type)) +
  geom_point(size = 2) +
  labs(y = "TPS", x = "") +
  theme_bw() +
  theme(legend.position = "none")


# Arrange plots

pp <- grid.arrange(plot_memory1, plot_cpu_sysload1, plot_disk_rw1, plot_tps1, nrow = 2, ncol = 2)
```

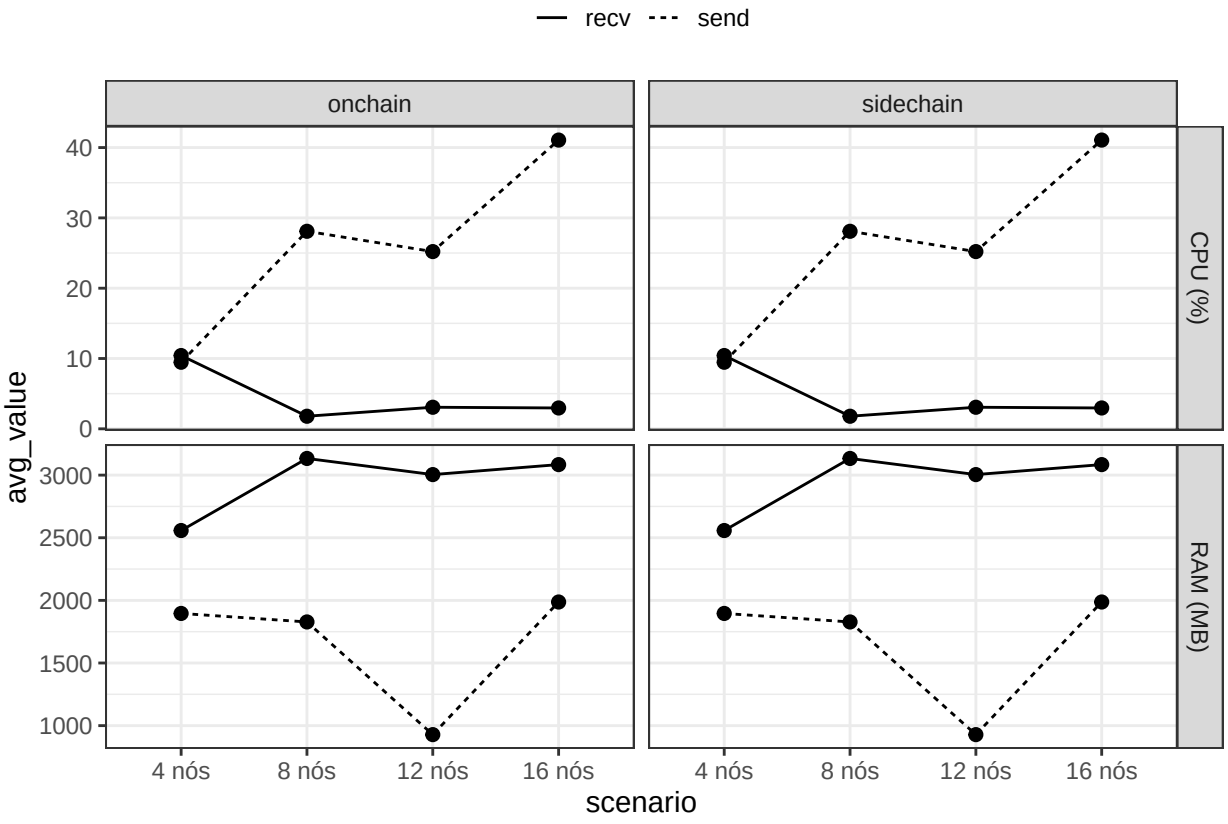



```
# ggsave("./figures/onchain-review-metrics-recu.png", pp)
```

## Agrupamento de todas as métricas onchain CPU e RAM (TESTE!!)

```
merge_df_metrics_onchain <- merge_df_metrics %>%
  filter(name %in% c("system_cpu_sysload", "system_memory_used")) %>%
  mutate(name = recode(name,
                       "system_cpu_sysload" = "CPU (%)",
                       "system_memory_used" = "RAM (MB)")) %>%
  mutate(system = "onchain")


merge_df_metrics_sidechain <-  merge_df_metrics %>%
  filter(name %in% c("system_cpu_sysload", "system_memory_used")) %>%
```

```
  mutate(name = recode(name,
                       "system_cpu_sysload" = "CPU (%)",
                       "system_memory_used" = "RAM (MB)")) %>%
  mutate(system = "sidechain")

mm_metrics <- rbind(merge_df_metrics_sidechain, merge_df_metrics_onchain)

ggplot(mm_metrics, aes(x = scenario, y = avg_value, linetype = type)) +
  geom_line(aes(group = type)) +
  geom_point(size = 2) +
  # labs(y = "CPU (%)", x = "") +
  facet_grid( name ~ system, scale = "free") +
  theme_bw() +
  guides(linetype = guide_legend(title = NULL)) +
  theme(legend.position = "top", legend.key.size = unit(0.5, "cm"),)
```




## Quantidade de erros no envio e recuperação

```
# Quantidade de erros nos requests, ou seja que não foram bem sucessidos (statusCode == 200)
errors <- benchSend12nodes$responseCode[benchSend12nodes$responseCode != 200]
length(errors) / length(benchSend12nodes$responseCode) * 100 # Porcentagem de erros em relação aos requ
```

```
## [1] 11.6
```