Monolith vs. Microservices: CPU, Latency & Throughput Analysis

Projeto de Arquitetura de Software

1. Pacotes e utilitários

```
library(dplyr)
library(tidyr)
library(ggplot2)
library(readr)
library(here)
library(lubridate)
library(broom)

# Pacotes opcionais para testes estatísticos
if(!require(agricolae, quietly = TRUE)) {
  message("Pacote 'agricolae' não encontrado. Testes post-hoc não estarão disponíveis.")
}
```

2. Carregamento dos dados

```
# ajuste o path dos results
df_cpu <- read_csv(here("merged_all.csv"))
df_locust <- read_csv(here("merged_locust.csv"))

# nomes fixos das colunas
cpu_metric_col <- "cores"   # ajuste para o nome correto no merged_all
latency_col    <- "total_median_response_time" # ajuste para o nome correto no merge_locust
throughput_col <- "req_s"
```

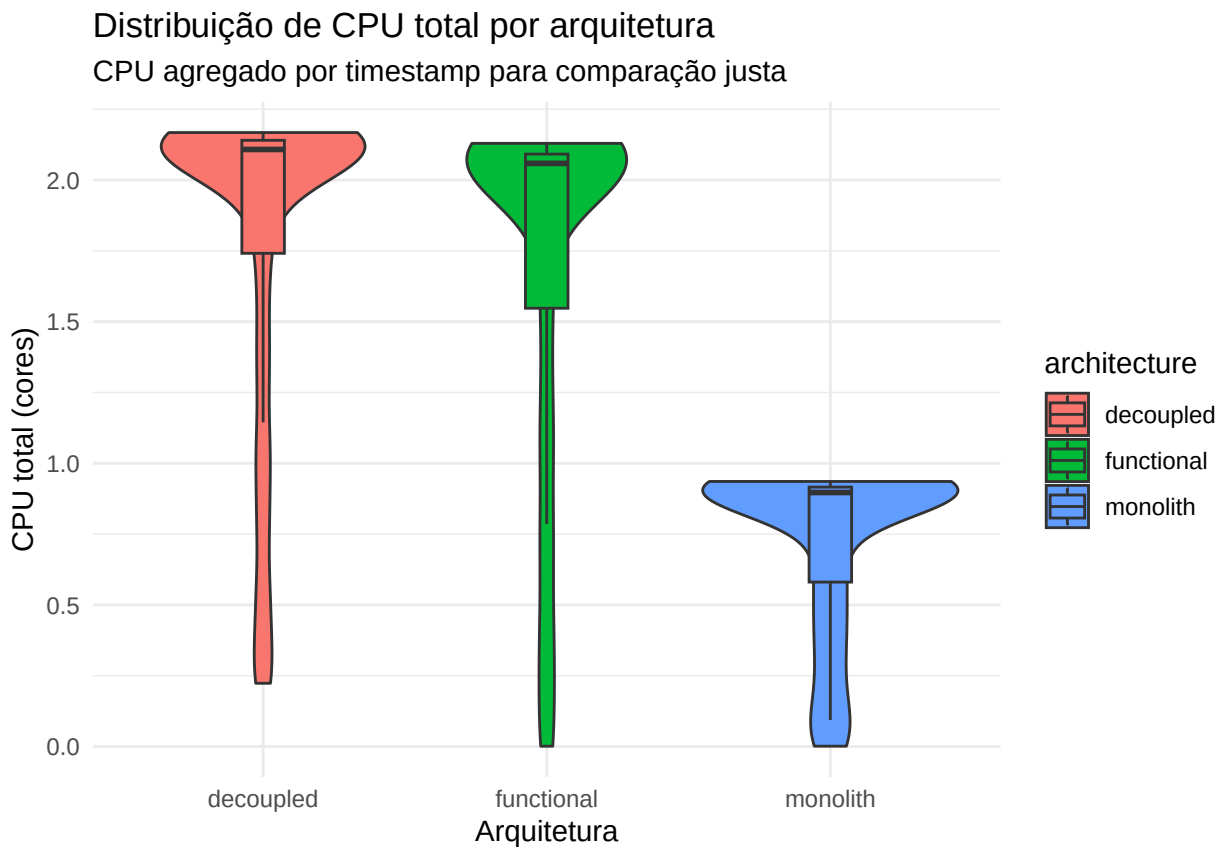
3. CPU

3.1 Entre arquiteturas (comparação justa - CPU agregado por timestamp)

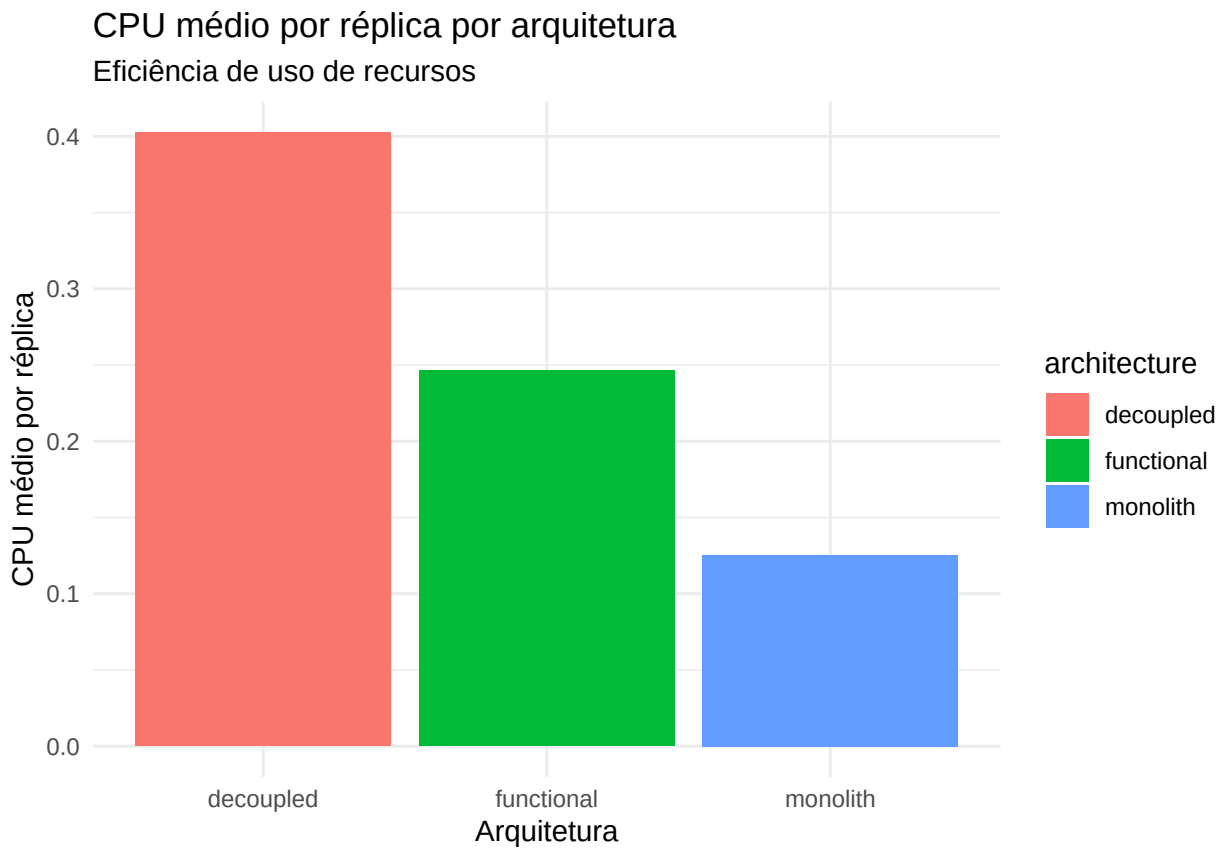
```r
# Para uma comparação justa, agregamos o uso de CPU por timestamp
# Monólito: já representa o uso total do sistema
# Microserviços: soma do uso de todos os serviços em cada timestamp
df_cpu_aggregated <- df_cpu %>%
  group_by(architecture, timestamp) %>%
  summarise(
    total_cpu = sum(.data[[cpu_metric_col]], na.rm = TRUE),
    .groups = "drop"
  )

# Estatísticas descritivas por arquitetura
df_cpu_aggregated %>%
  group_by(architecture) %>%
  summarise(
    cpu_mean   = mean(total_cpu, na.rm = TRUE),
    cpu_median = median(total_cpu, na.rm = TRUE),
    cpu_p95    = quantile(total_cpu, 0.95, na.rm = TRUE),
    cpu_max    = max(total_cpu, na.rm = TRUE)
  )
```

```
## # A tibble: 3 x 5
##   architecture cpu_mean cpu_median cpu_p95 cpu_max
##   <chr>           <dbl>      <dbl>   <dbl>   <dbl>
## 1 decoupled       1.79       2.11    2.15    2.17
## 2 functional      1.70       2.06    2.11    2.13
## 3 monolith        0.721      0.897   0.930   0.936
```

```r
# Visualização da distribuição de CPU agregado
df_cpu_aggregated %>%
  ggplot(aes(x = architecture, y = total_cpu, fill = architecture)) +
  geom_violin(trim = TRUE) +
  geom_boxplot(width = 0.15, outlier.shape = NA) +
  labs(title = "Distribuição de CPU total por arquitetura",
       subtitle = "CPU agregado por timestamp para comparação justa",
       x = "Arquitetura", y = "CPU total (cores)") +
  theme_minimal()
```




```r
# Comparação adicional: CPU por réplica (para entender eficiência)
df_cpu_per_replica <- df_cpu %>%
  mutate(cpu_per_replica = .data[[cpu_metric_col]] / pmax(current_replicas, 1)) %>%
  group_by(architecture, timestamp) %>%
  summarise(
    total_cpu_per_replica = sum(cpu_per_replica, na.rm = TRUE),
    .groups = "drop"
  )

df_cpu_per_replica %>%
  group_by(architecture) %>%
  summarise(
    cpu_per_replica_mean = mean(total_cpu_per_replica, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  ggplot(aes(x = architecture, y = cpu_per_replica_mean, fill = architecture)) +
  geom_col() +
  labs(title = "CPU médio por réplica por arquitetura",
       subtitle = "Eficiência de uso de recursos",
       x = "Arquitetura", y = "CPU médio por réplica") +
  theme_minimal()
```



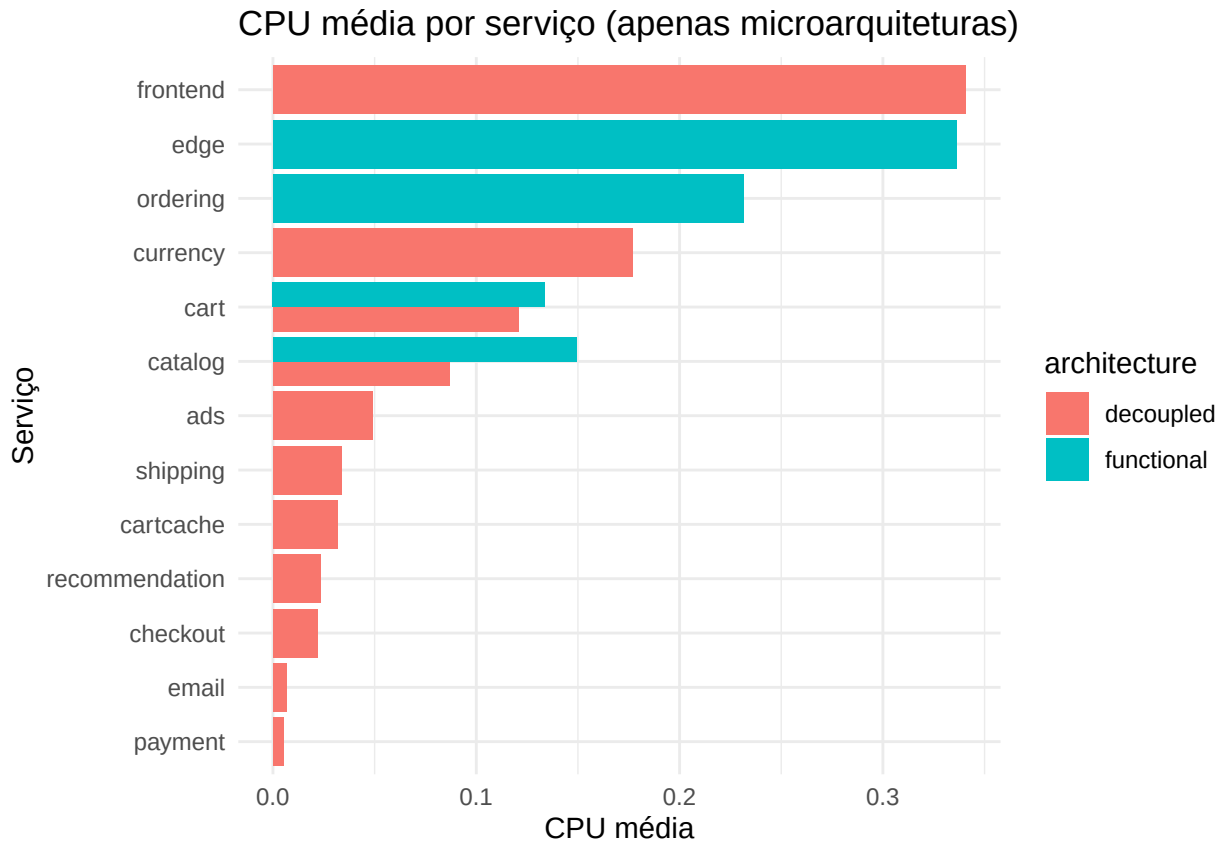
3.2 Entre serviços (somente microarquiteturas)

```
df_cpu %>%
  filter(architecture %in% c("decoupled", "functional"),
         service != "all_services") %>%
  group_by(architecture, service) %>%
  summarise(
    cpu_mean = mean(.data[[cpu_metric_col]], na.rm = TRUE),
    .groups = "drop"
  ) %>%
  arrange(architecture, desc(cpu_mean))
```

```
## # A tibble: 15 x 3
##    architecture service        cpu_mean
##    <chr>        <chr>             <dbl>
##  1 decoupled    frontend         0.341
##  2 decoupled    currency         0.177
##  3 decoupled    cart             0.121
##  4 decoupled    catalog          0.0869
##  5 decoupled    ads              0.0491
##  6 decoupled    shipping         0.0338
##  7 decoupled    cartcache        0.0319
##  8 decoupled    recommendation   0.0236
```

```
##  9 decoupled    checkout        0.0218
## 10 decoupled    email           0.00676
## 11 decoupled    payment         0.00508
## 12 functional   edge            0.336
## 13 functional   ordering        0.231
## 14 functional   catalog         0.149
## 15 functional   cart            0.134
```

```r
df_cpu %>%
  filter(architecture %in% c("decoupled", "functional"),
         service != "all_services") %>%
  group_by(architecture, service) %>%
  summarise(cpu_mean = mean(.data[[cpu_metric_col]], na.rm = TRUE), .groups = "drop") %>%
  ggplot(aes(x = reorder(service, cpu_mean), y = cpu_mean, fill = architecture)) +
  geom_col(position = "dodge") +
  coord_flip() +
  labs(title = "CPU média por serviço (apenas microarquiteturas)",
       x = "Serviço", y = "CPU média") +
  theme_minimal()
```



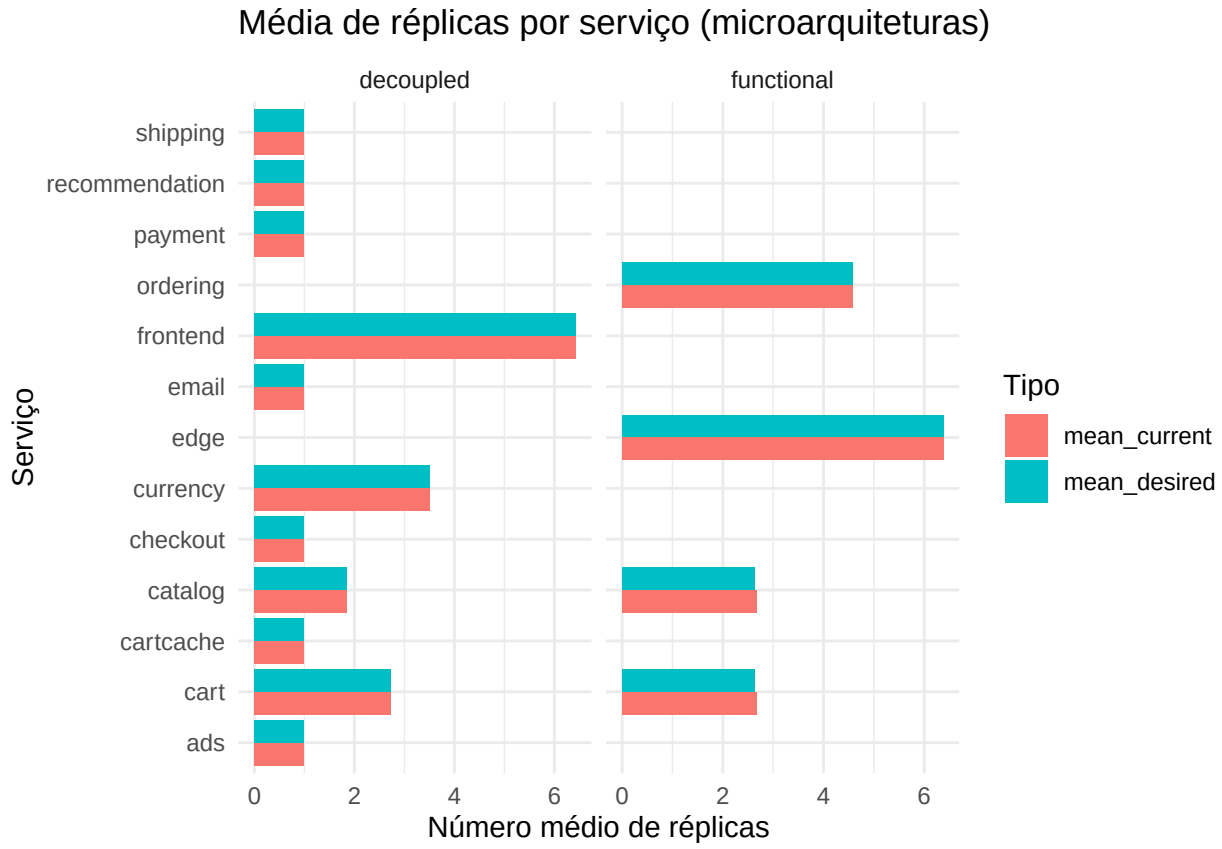
3.3 Réplicas — comparação entre serviços (microarquiteturas)

```
df_cpu %>%
  filter(architecture %in% c("decoupled", "functional"),
         service != "all_services") %>%
  group_by(architecture, service) %>%
  summarise(
    max_repl     = max(max_replicas, na.rm = TRUE),
    mean_desired = mean(desired_replicas, na.rm = TRUE),
    mean_current = mean(current_replicas, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  arrange(architecture, desc(mean_current))
```

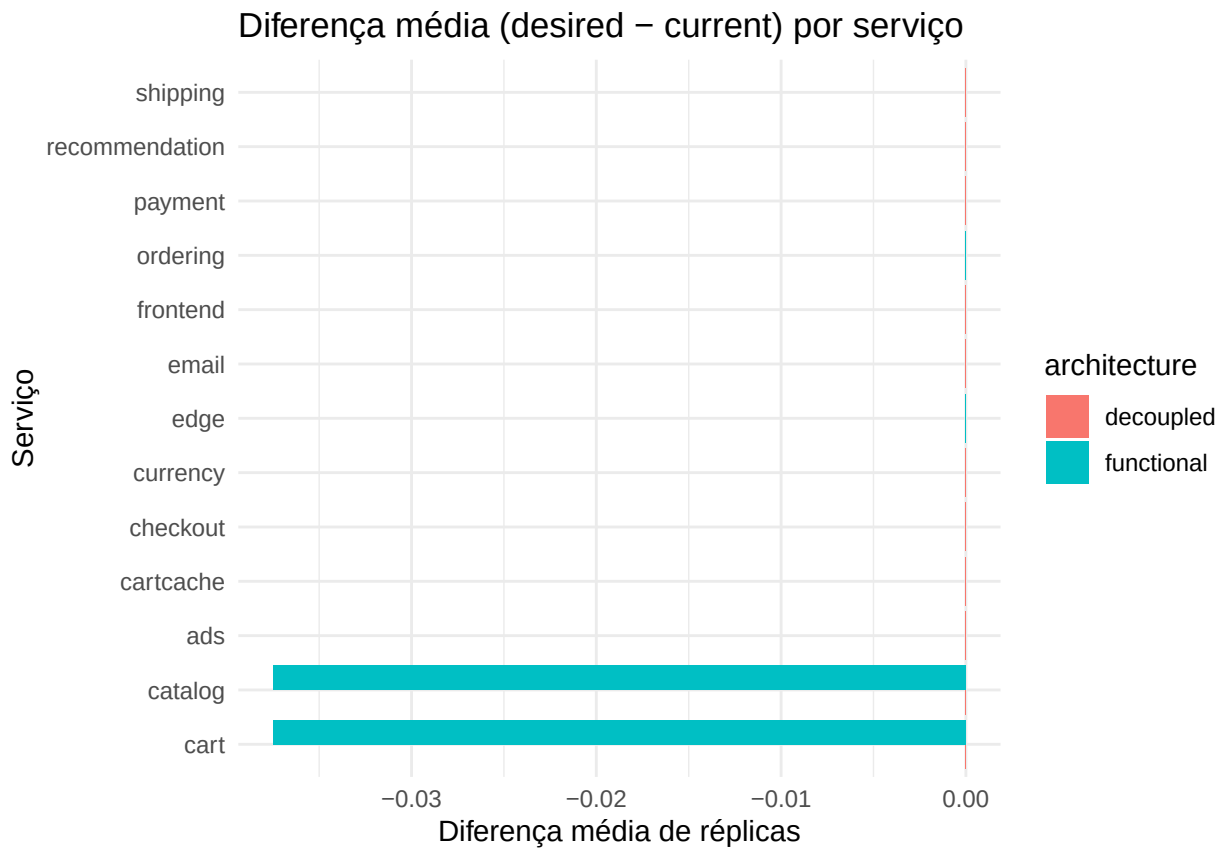
```
## # A tibble: 15 x 5
##    architecture service        max_repl mean_desired mean_current
##    <chr>        <chr>             <dbl>        <dbl>        <dbl>
##  1 decoupled    frontend             10         6.42         6.42
##  2 decoupled    currency             10         3.52         3.52
##  3 decoupled    cart                 10         2.73         2.73
##  4 decoupled    catalog              10         1.85         1.85
##  5 decoupled    ads                  10         1            1
##  6 decoupled    cartcache            10         1            1
##  7 decoupled    checkout             10         1            1
##  8 decoupled    email                10         1            1
##  9 decoupled    payment              10         1            1
## 10 decoupled    recommendation       10         1            1
## 11 decoupled    shipping             10         1            1
## 12 functional   edge                 10         6.38         6.38
## 13 functional   ordering             10         4.58         4.58
## 14 functional   cart                 10         2.64         2.68
## 15 functional   catalog              10         2.64         2.68
```

```
df_cpu %>%
  filter(architecture %in% c("decoupled", "functional"),
         service != "all_services") %>%
  group_by(architecture, service) %>%
  summarise(
    mean_desired = mean(desired_replicas, na.rm = TRUE),
    mean_current = mean(current_replicas, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  pivot_longer(cols = c(mean_desired, mean_current),
               names_to = "replica_type", values_to = "replicas") %>%
  ggplot(aes(x = service, y = replicas, fill = replica_type)) +
  geom_col(position = "dodge") +
```

```
  facet_wrap(~ architecture, scales = "free_x") +
  coord_flip() +
  labs(title = "Média de réplicas por serviço (microarquiteturas)",
       x = "Serviço", y = "Número médio de réplicas",
       fill = "Tipo") +
  theme_minimal()
```




```
df_cpu %>%
  filter(architecture %in% c("decoupled", "functional"),
         service != "all_services") %>%
  mutate(diff_desired_current = desired_replicas - current_replicas) %>%
  group_by(architecture, service) %>%
  summarise(
    mean_diff = mean(diff_desired_current, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  ggplot(aes(x = reorder(service, mean_diff), y = mean_diff, fill = architecture)) +
  geom_col(position = "dodge") +
  coord_flip() +
  labs(title = "Diferença média (desired - current) por serviço",
       x = "Serviço", y = "Diferença média de réplicas") +
  theme_minimal()
```



3.4 Análise de eficiência e overhead das arquiteturas

```
# Comparação de overhead entre arquiteturas
cpu_summary <- df_cpu_aggregated %>%
  group_by(architecture) %>%
  summarise(
    cpu_mean = mean(total_cpu, na.rm = TRUE),
    cpu_sd = sd(total_cpu, na.rm = TRUE),
    cpu_cv = cpu_sd / cpu_mean,  # Coeficiente de variação
    .groups = "drop"
  )

print("Resumo de uso de CPU por arquitetura:")
```

```
## [1] "Resumo de uso de CPU por arquitetura:"
```

```
print(cpu_summary)
```

```
## # A tibble: 3 x 4
##   architecture cpu_mean cpu_sd cpu_cv
```

```
##   <chr>           <dbl>  <dbl>  <dbl>
## 1 decoupled       1.79   0.586  0.326
## 2 functional      1.70   0.650  0.382
## 3 monolith        0.721  0.301  0.418
```

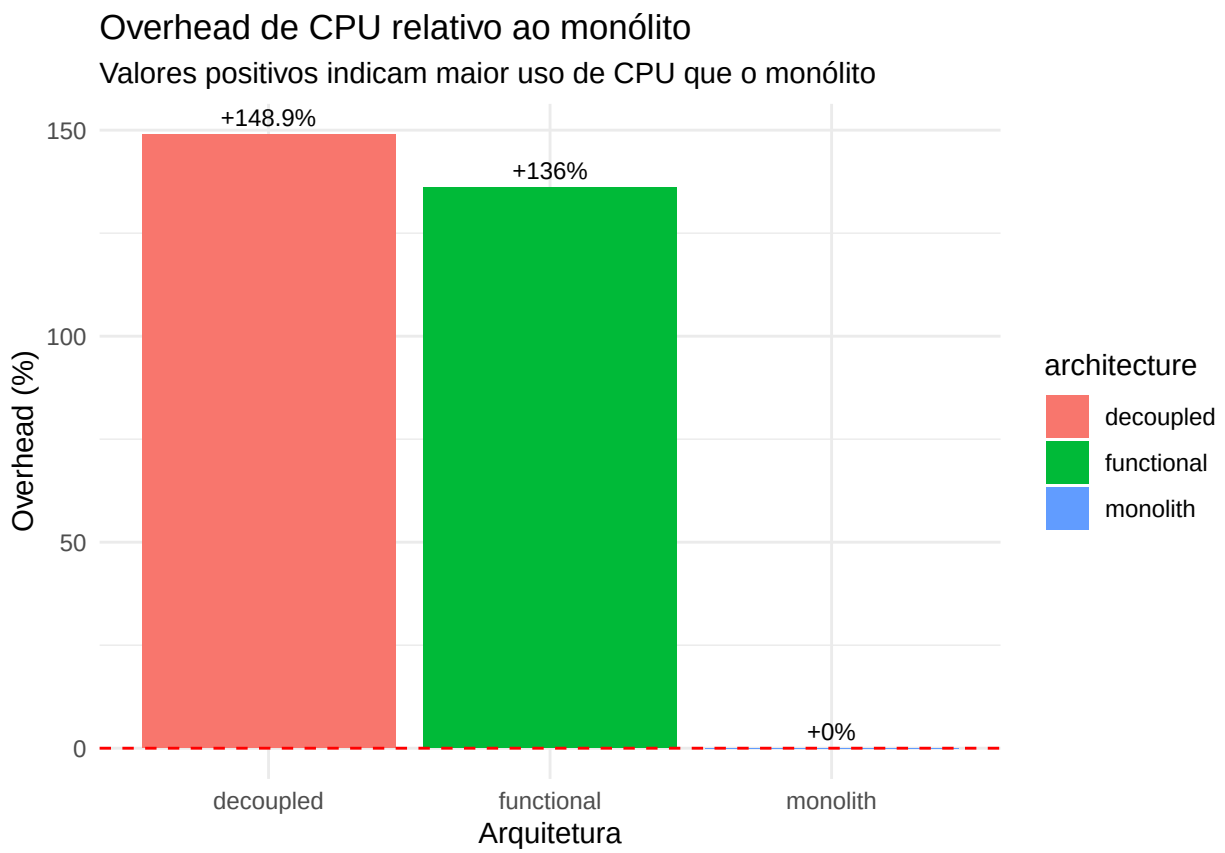
```r
# Overhead relativo (usando monólito como baseline)
monolith_mean <- cpu_summary %>%
  filter(architecture == "monolith") %>%
  pull(cpu_mean)

cpu_overhead <- cpu_summary %>%
  mutate(
    overhead_factor = cpu_mean / monolith_mean,
    overhead_percent = (cpu_mean - monolith_mean) / monolith_mean * 100
  )

print("Overhead relativo ao monólito:")
```

```
## [1] "Overhead relativo ao monólito:"
```

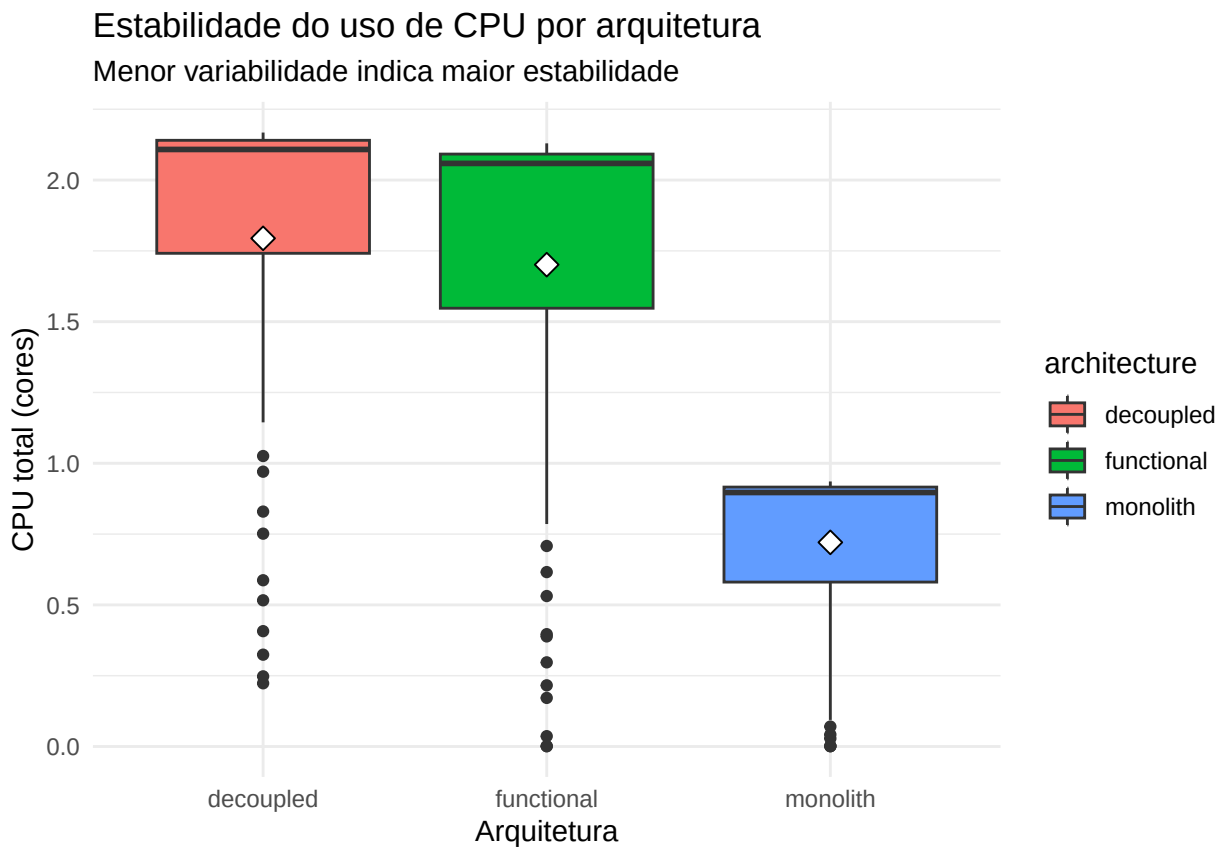
```r
print(cpu_overhead %>% select(architecture, overhead_factor, overhead_percent))
```

```
## # A tibble: 3 x 3
##   architecture overhead_factor overhead_percent
##   <chr>                  <dbl>            <dbl>
## 1 decoupled               2.49             149.
## 2 functional              2.36             136.
## 3 monolith                1                  0
```

```r
# Visualização do overhead
cpu_overhead %>%
  ggplot(aes(x = architecture, y = overhead_percent, fill = architecture)) +
  geom_col() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Overhead de CPU relativo ao monólito",
       subtitle = "Valores positivos indicam maior uso de CPU que o monólito",
       x = "Arquitetura", y = "Overhead (%)") +
  theme_minimal() +
  geom_text(aes(label = paste0("+", round(overhead_percent, 1), "%")),
            vjust = -0.5, size = 3)
```




```
# Análise de estabilidade (variabilidade do uso de CPU)
df_cpu_aggregated %>%
  ggplot(aes(x = architecture, y = total_cpu, fill = architecture)) +
  geom_boxplot() +
  stat_summary(fun = mean, geom = "point", shape = 23, size = 3, fill = "white") +
  labs(title = "Estabilidade do uso de CPU por arquitetura",
       subtitle = "Menor variabilidade indica maior estabilidade",
       x = "Arquitetura", y = "CPU total (cores)") +
  theme_minimal()
```



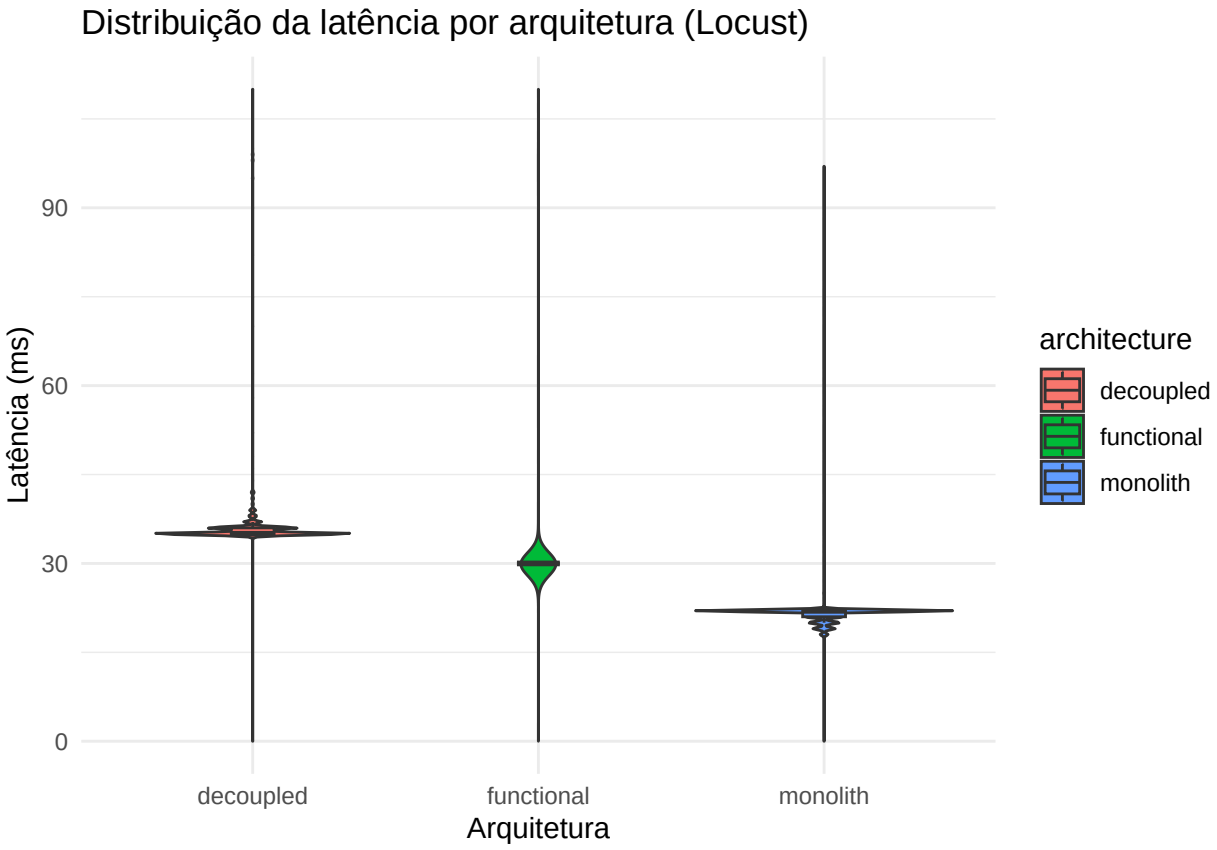
4. Latência (Locust)

```
df_locust %>%
  group_by(architecture) %>%
  summarise(
    latency_mean   = mean(.data[[latency_col]], na.rm = TRUE),
    latency_median = median(.data[[latency_col]], na.rm = TRUE),
    latency_p95    = quantile(.data[[latency_col]], 0.95, na.rm = TRUE)
  )
```

```
## # A tibble: 3 x 4
##   architecture latency_mean latency_median latency_p95
##   <chr>               <dbl>          <dbl>       <dbl>
## 1 decoupled            36.6             35          39
## 2 functional           31.0             30          30
## 3 monolith             22.0             22          22
```

```
df_locust %>%
  ggplot(aes(x = architecture, y = .data[[latency_col]], fill = architecture)) +
  geom_violin(trim = TRUE) +
```

```
  geom_boxplot(width = 0.15, outlier.shape = NA) +
  labs(title = "Distribuição da latência por arquitetura (Locust)",
       x = "Arquitetura", y = "Latência (ms)") +
  theme_minimal()
```



5. Throughput (Locust)

```
df_locust %>%
  group_by(architecture) %>%
  summarise(
    thr_mean   = mean(.data[[throughput_col]], na.rm = TRUE),
    # thr_median = median(.data[[throughput_col]], na.rm = TRUE),
    thr_p95    = quantile(.data[[throughput_col]], 0.95, na.rm = TRUE)
  )
```

```
## # A tibble: 3 x 3
##   architecture thr_mean thr_p95
##   <chr>           <dbl>   <dbl>
## 1 decoupled        59.3    61.5
## 2 functional       59.6    61.7
## 3 monolith         59.9    61.9
```

```r
df_locust %>%
  ggplot(aes(x = architecture, y = .data[[throughput_col]], fill = architecture)) +
  geom_violin(trim = TRUE) +
  geom_boxplot(width = 0.15, outlier.shape = NA) +
  labs(title = "Distribuição do throughput por arquitetura (Locust)",
       x = "Arquitetura", y = "Requests/s") +
  theme_minimal()
```