ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

XÁC SUẤT THỐNG KÊ (MT2013)

Bài tập lớn

Đánh giá hiệu năng GPU

GVHD: Nguyễn Thị Kiều Dung

	Nguyễn Văn A	**1234567**
Mã nhóm:	**Trần Văn B**	**1234568**
	Lê Văn C	**1234569**

HỒ CHÍ MINH, THÁNG 11/ 2025

Mục lục

Danh sách hình vẽ

Danh sách bảng

Danh sách đoạn mã

1 Tổng quan dữ liệu

Trong thời đại công nghệ phát triển nhanh chóng, bộ xử lý đồ họa (GPU – Graphics Processing Unit) đã trở thành một trong những thành phần quan trọng nhất của máy tính hiện đại. Ban đầu, GPU được thiết kế với mục đích chính là xử lý hình ảnh và đồ họa trong các trò chơi điện tử, phần mềm thiết kế và dựng phim. Tuy nhiên, cùng với sự tiến bộ của công nghệ, vai trò của GPU đã vượt xa khỏi phạm vi đồ họa thuần túy. Ngày nay, GPU đóng vai trò cốt lõi trong các lĩnh vực như trí tuệ nhân tạo (AI), học sâu (Deep Learning), mô phỏng khoa học, và xử lý dữ liệu lớn. Nhờ vào cấu trúc song song mạnh mẽ với hàng nghìn lõi xử lý, GPU có thể thực hiện hàng loạt phép tính phức tạp cùng lúc, giúp rút ngắn đáng kể thời gian xử lý so với CPU truyền thống. Dưới đây là tập dữ liệu khảo sát các yếu tố về GPU cung cấp những thông tin chi tiết như tốc độ xung nhịp, nhiệt độ tối đa, mức tiêu thụ điện năng, kích thước khuôn chip, ngày phát hành, giá bán, và nhiều đặc trưng kỹ thuật khác. Việc phân tích các dữ liệu này giúp ta hiểu rõ hơn về sự phát triển của công nghệ GPU qua các giai đoạn, từ đó đánh giá xu hướng tiến hóa về hiệu năng, giá thành, và mức độ tối ưu năng lượng. Bên cạnh đó, người nghiên cứu có thể khám phá mối quan hệ giữa giá thành và hiệu suất hoạt động, tìm hiểu xem liệu có nhà sản xuất nào nổi bật trong một phân khúc nhất định hay không. Thông qua việc khai thác và phân tích dữ liệu GPU, chúng ta có thể dự đoán xu hướng của các thế hệ GPU tương lai, phục vụ cho các ứng dụng như học máy, đồ họa máy tính, và tính toán hiệu năng cao.

Tập dữ liệu All_GPUs, bao gồm 34 thông số của 3406 bộ xử lý đồ hoạ (GPU) khác nhau đến từ 3 nhà sản xuất chính là NVIDIA, AMD và Intel. Dữ liệu được quan sát và thu thập từ trang web Kaggle. Tập dữ liệu này có tương đối nhiều thông số, trong đó có thể kể đến một số thông số được nêu ra trong bảng 1.1 dưới đây:

STT	Tên biến	Đơn vị	Mô tả
1	Manufacturer		Hãng của sản xuất GPU
2	Release_Date		Năm sản xuất của GPU
3	Memory_Bandwidth	Gigabyte/giây (GB/s)	Lượng dữ liệu tối đa mà bộ nhớ GPU có thể truyền tải trong mỗi giây
4	Memory_Speed	MHz	Độ rộng của bus bộ nhớ, ảnh hưởng đến tốc độ truy cập và hiệu suất của bộ nhớ GPU
5	L2_Cache	KB	Bộ nhớ đệm cấp 2 giúp GPU truy cập nhanh hơn vào dữ liệu được sử dụng thường xuyên, tối ưu hóa hiệu suất
6	Memory_Bus	Bit	Độ rộng của kênh truyền dữ liệu trong RAM (Memory).
7	Memory		Dung lượng bộ nhớ đồ họa (VRAM) của GPU, quyết định khả năng xử lý và lưu trữ dữ liệu hình

Bảng 1.1: Bảng mô tả một vài thông số quan trọng của tập dữ liệu GPU

Như đã đề cập ở trên, tập dữ liệu có tổng cộng 34 thông số khác nhau, tuy nhiên nếu liệt kê hết ở bảng 1.1 thì sẽ rất dài. Ngoài ra, trong các thông số đó, có nhiều thông số mang tính kỹ thuật cao và không phổ biến, những dữ liệu này sẽ được sử lý sau để dễ dàng hơn trong việc phân tích và trực quan hoá dữ liệu.

2 Tiền xử lý dữ liệu

2.1 Đọc dữ liệu vào R

Ta đọc dữ liệu từ file dữ liệu đã cho dưới định dạng CSV vào R bằng hàm `read.csv()` như sau:

```
df <- read.csv("./data_sets/All_GPUs.csv")
head(df, 5)
```

Listing 2.1: Đọc dữ liệu từ file trong R

Sau khi đọc dữ liệu xong, ta có thể sử dụng hàm `head()` để hiển thị 5 dòng đầu tiên của dữ liệu nhằm kiểm tra xem dữ liệu đã được đọc đúng chưa (Hình 2.1).

```
> df ← read.csv("./data_sets/All_GPUs.csv")
> head(df, 5)
  Architecture Best_Resolution Boost_Clock Core_Speed DVI_Connection Dedicated Direct_X
1  Tesla G92b                             738 MHz              2       Yes  DX 10.0
2     R600 XT       1366 x 768                 \n-              2       Yes    DX 10
3    R600 PRO       1366 x 768                 \n-              2       Yes    DX 10
4       RV630       1024 x 768                 \n-              2       Yes    DX 10
5       RV630       1024 x 768                 \n-              2       Yes    DX 10
  DisplayPort_Connection HDMI_Connection Integrated L2_Cache Manufacturer Max_Power   Memory Memory_Bandwidth
1                     NA               0         No      0KB       Nvidia 141 Watts 1024 MB          64GB/sec
2                     NA               0         No      0KB          AMD 215 Watts  512 MB         106GB/sec
3                     NA               0         No      0KB          AMD 200 Watts  512 MB        51.2GB/sec
4                     NA               0         No      0KB          AMD            256 MB        36.8GB/sec
5                     NA               0         No      0KB          AMD  45 Watts  256 MB        22.4GB/sec
  Memory_Bus Memory_Speed Memory_Type                          Name Notebook_GPU Open_GL
1   256 Bit      1000 MHz       GDDR3                GeForce GTS 150          No     3.3
2   512 Bit       828 MHz       GDDR3           Radeon HD 2900 XT 512MB          No     3.1
3   256 Bit       800 MHz       GDDR3                Radeon HD 2900 Pro          No     3.1
4   128 Bit      1150 MHz       GDDR4 Radeon HD 2600 XT Diamond Edition          No     3.3
5   128 Bit       700 MHz       GDDR3                 Radeon HD 2600 XT          No     3.1
                PSU  Pixel_Rate Power_Connector Process ROPs   Release_Date Release_Price Resolution_WxH
1 450 Watt & 38 Amps 12 GPixel/s          None     55nm   16 \n01-Mar-2009                     2560x1600
2 550 Watt & 35 Amps 12 GPixel/s          None     80nm   16 \n14-May-2007                     2560x1600
3 550 Watt & 35 Amps 10 GPixel/s          None     80nm   16 \n07-Dec-2007                     2560x1600
4                     3 GPixel/s          None     65nm    4 \n01-Jul-2007                     2560x1600
5 400 Watt & 25 Amps  3 GPixel/s          None     65nm    4 \n28-Jun-2007                     2560x1600
  SLI_Crossfire Shader TMUs Texture_Rate VGA_Connection
1           Yes      4   64  47 GTexel/s              0
2           Yes      4   16  12 GTexel/s              0
3           Yes      4   16  10 GTexel/s              0
4           Yes      4    8   7 GTexel/s              0
5           Yes      4    8   6 GTexel/s              0
```

Hình 2.1: Hiển thị 5 dòng đầu tiên của dữ liệu sau khi đọc vào R

Như vậy, ta đã hoàn thành việc đọc dữ liệu từ file CSV vào R và có thể tiến hành các bước tiền xử lý dữ liệu tiếp theo.

2.2 Làm sạch dữ liệu

Vì trong file dữ liệu ban đầu, có thể tồn tại các giá trị bị thiếu (NA) hoặc các giá trị không hợp lệ, ta cần thực hiện các bước làm sạch dữ liệu để đảm bảo tính chính xác và độ tin cậy của phân tích sau này.

Trước tiên, ta sẽ phải thay thế tất cả các giá trị rác, không hợp lệ thành giá trị NA trong R. Ví dụ, nếu một ô bất kì có giá trị là chuỗi rỗng `""` hoặc ký tự đặc biệt như `"N/A"`, ta sẽ thay thế chúng bằng NA như sau:

```
df <- df %>%
  mutate(across(where(is.character), trimws))
df[df == ""] <- NA
df[df == "N/A"] <- NA
df[df == "NA"] <- NA
df[df == "-"] <- NA
df[df == "Unknown Release Date"] <- NA
# Chi lay nam san xuat, khong lay ngay cu the
df$Release_Date <- as.Date(df$Release_Date, format = "%d-%b-%Y")
df$Release_Date <- format(df$Release_Date, "%Y")
```

Listing 2.2: Thay thế giá trị rác thành NA

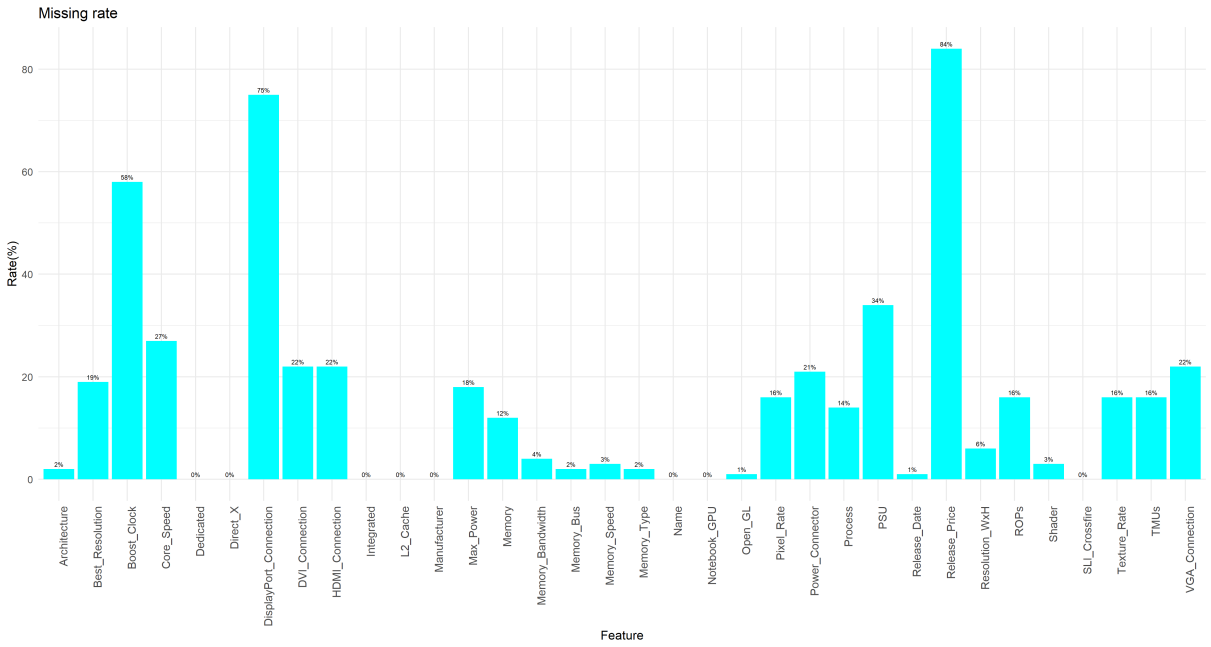
Sau khi thay thế các giá trị rác, nhóm nhận thấy rằng có rất nhiều yếu tố có số lượng NA lớn, điều này buộc nhóm phải lựa chọn giữa loại bỏ và chuẩn hoá. Trong nội dung bài báo cáo này, nhóm sẽ loại bỏ những đặc điểm (cột) có số lượng giá trị NA vượt quá 10% tổng số dòng dữ liệu. Để thực hiện việc này, ta có thể sử dụng đoạn mã sau:

```
# Dem so luong gia tri NA trong moi cot
missing_counts = freq.na(df)

# Ve do thi ty le du lieu khuyet
ggplot(missing_counts, aes(x = rownames(missing_counts), y = missing_counts[,2], ))
    ↪ +
  geom_bar(stat = "identity", fill = "cyan") +
  geom_text(aes(label = paste0(missing_counts[,2], "%")), vjust = -0.5, size = 2) +
  labs(title = "Missing rate", x = "Feature", y = "Rate (%)") +
  theme_minimal() +
  theme(axis.text.x = element_text(
    size = 10,
    angle = 90,
    hjust = 1
  ))
```

Listing 2.3: Loại bỏ các cột có tỷ lệ NA $>$ 10%

Kết quả trực quan hóa tỷ lệ dữ liệu khuyết thiếu được thể hiện trong Hình 2.2.



Hình 2.2: Tỷ lệ dữ liệu khuyết thiếu trong các đặc trưng

Tiếp theo sau đó, ta sẽ loại bỏ các cột có tỷ lệ giá trị NA vượt quá 10% tổng số dòng dữ liệu như sau:

```
missing_counts_df <- data.frame(
  feature = rownames(missing_counts),
  percent = missing_counts[,2]
)

cols_to_keep <- missing_counts_df$feature[missing_counts_df$percent ⩽ 10 & missing_
    ↪ counts_df$feature != "Architecture" & missing_counts_df$feature != "Name"]
df_filtered <- df[, cols_to_keep, drop = FALSE]

head(df_filtered, 5)
df_filtered <- na.omit(df_filtered)
```

Listing 2.4: Loại bỏ các cột có tỷ lệ NA > 10%

Bằng câu lệnh `print(names(df_filtered))`, ta có thể kiểm tra lại các cột còn lại sau khi đã loại bỏ các cột có tỷ lệ giá trị NA vượt quá 10% (Hình 2.3).

```
> print(names(df_filtered))
 [1] "Resolution_WxH"    "Memory_Bandwidth" "Shader"           "Memory_Speed"     "Memory_Bus"
 [6] "Memory_Type"       "Open_GL"          "Release_Date"     "Dedicated"        "Integrated"
[11] "Direct_X"          "L2_Cache"         "Manufacturer"     "Notebook_GPU"     "SLI_Crossfire"
```

Hình 2.3: Các cột còn lại sau khi loại bỏ các cột có tỷ lệ NA > 10%

Mặc dù đã làm sạch dữ liệu bằng cách loại bỏ các cột có tỷ lệ khuyết hoặc số lượng giá trị NA cao, vẫn còn một yếu tố khiến việc phân tích dữ liệu trở nên khó khăn, đó là các đơn vị đo, do đó ta cần chuẩn hoá các đơn vị đo trong dữ liệu bằng cách loại bỏ chúng.

```r
# Chuan hoa don vi do trong cac cot
remove_unit_cols <- c("Memory_Bandwidth", "Memory_Speed", "Memory_Bus", "Direct_X")

main_df <- df_filtered

# print(df_filtered[remove_unit_cols])
main_df[remove_unit_cols] <- lapply(df_filtered[remove_unit_cols], function(x) {
  as.numeric(gsub("[^0-9.]", "", x))
})

clean_cache <- function(x) {
  main <- as.numeric(sub("KB.*", "", x))      # 2304
  mult      <- as.numeric(sub(".*\\(x([0-9]+)\\)", "\\1", x))  # 2
  if (is.na(mult)) mult <- 1
  return(main * mult)
}

main_df$L2_Cache <- sapply(main_df$L2_Cache, clean_cache)
```

Listing 2.5: Chuẩn hoá các đơn vị đo trong dữ liệu