



**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**  
**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ**  
**ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт      компьютерных наук  
Кафедра      автоматизированных систем управления

Практическая работа №1  
по дисциплине  
«Программно-аппаратные средства интеллектуальных систем»

Студент М-РИТ-25-1

\_\_\_\_\_  
(подпись, дата)

Станиславчук С. М.

Руководитель

Ст. преподаватель

\_\_\_\_\_  
(подпись, дата)

Болдырихин О. В.

Липецк 2025

## Содержание

1. Задание, конкретизированное вариантом
2. Описание исследуемого показателя
3. Исходные данные
4. Формулы для построения гистограмм
5. Используемые программные средства
6. Результаты расчёта: таблицы, графики
7. Выводы

### 1. Задание, конкретизированное вариантом

Построить статистические оценки (гистограммы) функций распределения и плотностей распределения вероятностей основных технико-эксплуатационных показателей компьютеров рейтинга top500 для нескольких заданных списков по отдельности и для всех вместе в совокупности.

Вариант 17: Энергоэффективность, Energy Efficiency (33, 32, 31)

### 2. Описание исследуемого показателя

Энергоэффективность вычислительной машины — это показатель, отражающий **соотношение между объемом выполненной работы и потребленной энергией**. Чем выше энергоэффективность, тем меньше энергии тратит машина на выполнение задачи.

Формула расчёта:

$$EnergyEfficiency = R_{max} / Power,$$

$R_{max}$  - наивысший результат, полученный при использовании системы тестов [Linpack](#) (измеряется в [PFLOPS](#))

Power — потребляемая мощность суперкомпьютера во время выполнения теста [Linpack](#) (измеряется в киловаттах)

### 3. Исходные данные

В качестве исходных данных используются результаты рейтингов TOP500 под номерами 31, 32 и 33, соответствующие выпускам июнь 2008 года, ноябрь 2008 года и июнь 2009 года соответственно.

Данные этих рейтингов представлены в приложении 2.

### 4. Формулы для построения гистограмм

Для построения гистограмм распределения показателя энергоэффективности суперкомпьютеров использовались стандартные статистические определения:

$$h = (x_{\max} - x_{\min}) / k$$

где

$h$  — ширина интервала (шага гистограммы),

$x_{\max}$ ,  $x_{\min}$  — максимальное и минимальное значения показателя.

Высота столбца гистограммы для интервала  $i$  равна

$$f_i = n_i / (N * h)$$

где

$n_i$  — количество наблюдений, попавших в интервал  $i$ ,

$N$  — общее число наблюдений.

Выполняемое условие нормировки:

$$\sum_{i=1}^k f_i \cdot h = 1$$

Для построения оценки плотности распределения вероятности (KDE) использовалась гауссовская аппроксимация:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right), \quad K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$$

где

$x_i$  — наблюдаемое значение энергоэффективности,

$K(u)$  — ядро Гаусса,

$h$  — параметр сглаживания.

## 5. Использованные программные средства

Для выполнения анализа использовался язык программирования Python

3.

Необходимые библиотеки устанавливались с помощью команды:

```
pip install requests lxml matplotlib pandas numpy scipy tabulate
```

С помощью библиотеки requests выполнялось автоматическое скачивание XML-файлов с сайта [top500.org](http://top500.org)

Модуль `lxml.etree` применялся для разбора структуры XML. Из каждого документа извлекались значения:

- `rank` — место системы в рейтинге,
- `r-max` — достигнутая производительность (в FLOPS),
- `power` — энергопотребление (в киловаттах),
- `system` — название суперкомпьютера.

После извлечения вычислялся показатель энергоэффективности.

Затем все записи объединяются в единую таблицу (`DataFrame`) с помощью библиотеки `pandas` и сохраняются в файл `top500_efficiency.csv`.

Для каждой выборки (отдельного списка TOP500) строились гистограммы распределения энергоэффективности.

По оси X — реальные значения энергоэффективности.

По оси Y — одновременно количество систем (`Count`) и плотность распределения (`Density`).

Для оценки плотности применялся метод ядровой оценки плотности (KDE) из пакета `scipy.stats.gaussian_kde`.

С помощью модуля `tabulate` формировалась таблица из 20 наиболее энергоэффективных систем с указанием их ранга, названия, производительности, энергопотребления и вычисленной эффективности.

## 6. Результаты расчёта: таблицы, графики.

Первые 20 вычислительных машин, отсортированных по энергоэффективности (по убыванию) представлены на рисунке 1. Гисторграммы эффективности для 2008(1), 2008(2) и 2009 представлены на рисунках 2, 3, 4 соответственно. Общая гистограмма представлена на рисунке 5.

```

stanik@archlinux: /home/stanik/programmer/++Programmer/5_1/PASAI/pract/1 > ./venv/bin/python gist.py
[SKIP] xml/TOP500_200906.xml already exists
Rank 33, year 2009, month 6: parsed 239 records
[SKIP] xml/TOP500_200811.xml already exists
Rank 32, year 2008, month 11: parsed 253 records
[SKIP] xml/TOP500_200806.xml already exists
Rank 31, year 2008, month 6: parsed 247 records
[CSV] Saved tops500_efficiency.csv (739 rows)
[PLOT] tops500_efficiency_plots/top500_efficiency_List#31.png saved
[PDF] tops500_efficiency_plots/top500_efficiency_cdf_List#31.png saved
[PLOT] tops500_efficiency_plots/top500_efficiency_List#32.png saved
[PDF] tops500_efficiency_plots/top500_efficiency_cdf_List#32.png saved
[PLOT] tops500_efficiency_plots/top500_efficiency_List#33.png saved
[PDF] tops500_efficiency_plots/top500_efficiency_cdf_List#33.png saved
[PLOT] tops500_efficiency_plots/top500_efficiency_summary.png saved

```

```

=== TOP500 Energy Efficiency Table ===

```

	list_rank	rank	system	rmax	power	efficiency
694	33	422	BladeCenter QS22 Cluster, PowerXCell 8i 4.0 Ghz, Infiniband	18570.000	34.630	536.240
352	32	220	BladeCenter QS22 Cluster, PowerXCell 8i 4.0 Ghz, Infiniband	18570.000	34.630	536.240
474	32	429	BladeCenter QS22 Cluster, PowerXCell 8i 3.2 Ghz, Infiniband	13990.000	26.380	530.326
476	32	431	BladeCenter QS22 Cluster, PowerXCell 8i 3.2 Ghz, Infiniband	13990.000	26.380	530.326
475	32	430	BladeCenter QS22 Cluster, PowerXCell 8i 3.2 Ghz, Infiniband	13990.000	26.380	530.326
149	31	324	Monte Capanne - BladeCenter QS22 Cluster, PowerXCell 8i 3.2 Ghz, Infiniband	11110.000	22.760	488.137
223	31	464	Cell - BladeCenter QS22 Cluster, PowerXCell 8i 3.2 Ghz, Infiniband	9259.000	18.970	488.086
539	33	61	Cerrillos - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 Ghz, Infiniband	63250.000	138.000	458.333
280	32	42	BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 Ghz, Infiniband	63250.000	138.000	458.333
279	32	41	Cerrillos - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 Ghz, Infiniband	63250.000	138.000	458.333
540	33	62	BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 Ghz, Infiniband	63250.000	138.000	458.333
500	33	1	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 Ghz, Voltaire Infiniband	1105000.000	2483.470	444.942
247	32	1	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 Ghz, Voltaire Infiniband	1105000.000	2483.470	444.942
0	31	1	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 Ghz, Voltaire Infiniband	1026000.000	2345.500	437.433
608	33	277	GRAPE-DR cluster - GRAPE-DR accelerator Cluster, Infiniband	23860.000	51.200	428.906
146	31	306	Blue Gene/P Solution	11710.000	31.500	371.746
145	31	305	Schrödinger - Blue Gene/P Solution	11710.000	31.500	371.746
144	31	304	Blue Gene/P Solution	11710.000	31.500	371.746
565	33	124	Blue Gene/P Solution	35123.000	94.500	371.672
27	31	51	Blue Gene/P Solution	35123.000	94.500	371.672

... (only first 20 rows shown)

Рисунок 1 — 20 самых энергоэффективных ВМ (Тор500 2008-2009(июнь), 31-33 рейтинги)

TOP500 Energy Efficiency — List#31 (N=247)

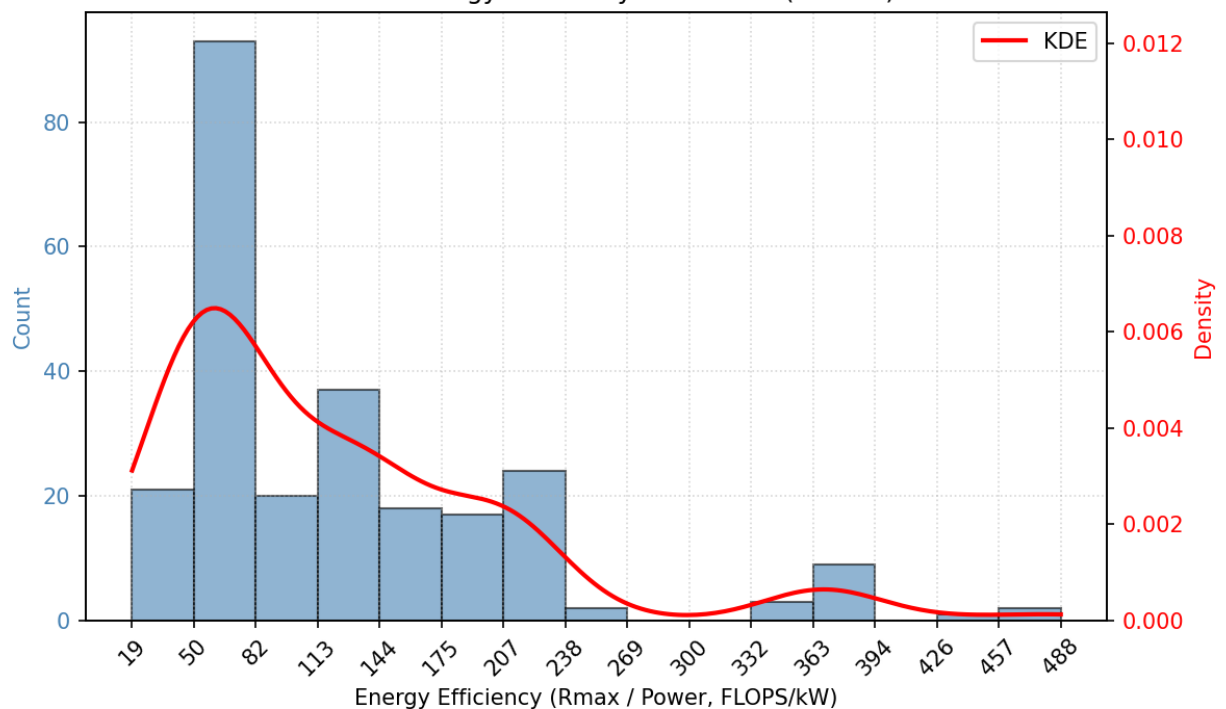


Рисунок 2 — Гистограмма энергоэффективности (Тор500, июнь 2008, 31 рейтинг)

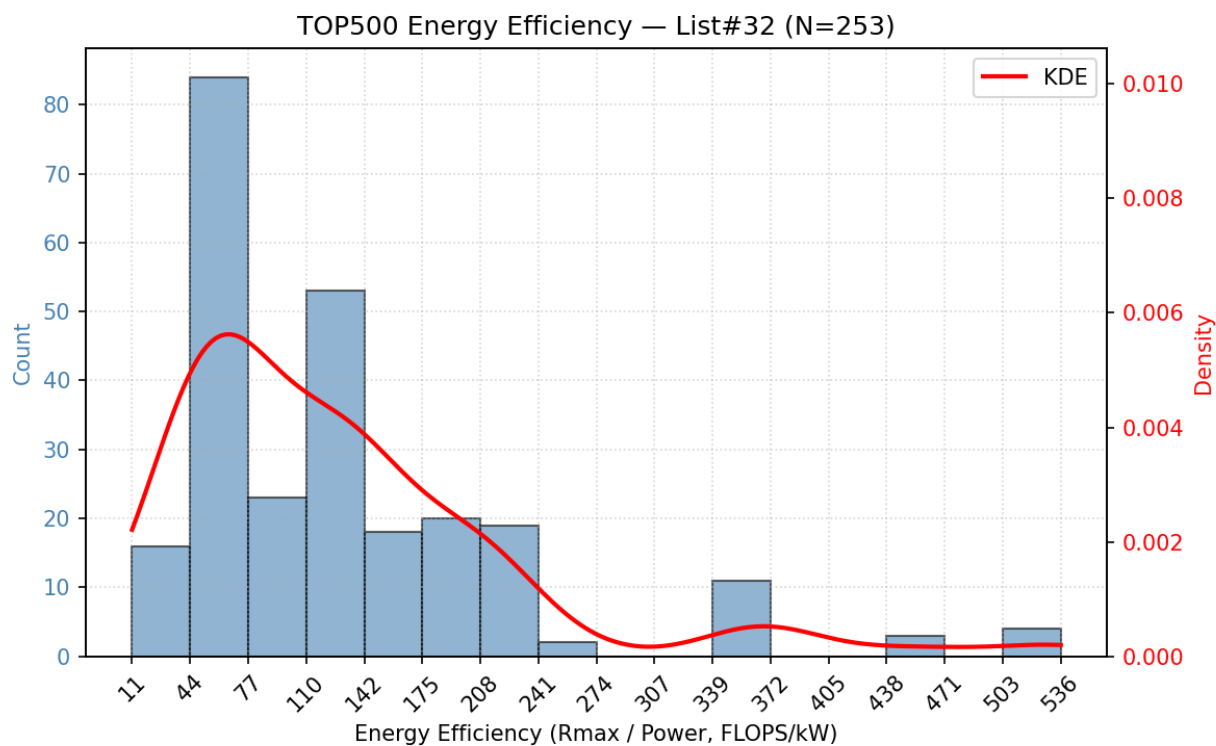


Рисунок 3 — Гистограмма энергоэффективности (Тор500, октябрь 2008, 32 рейтинг)

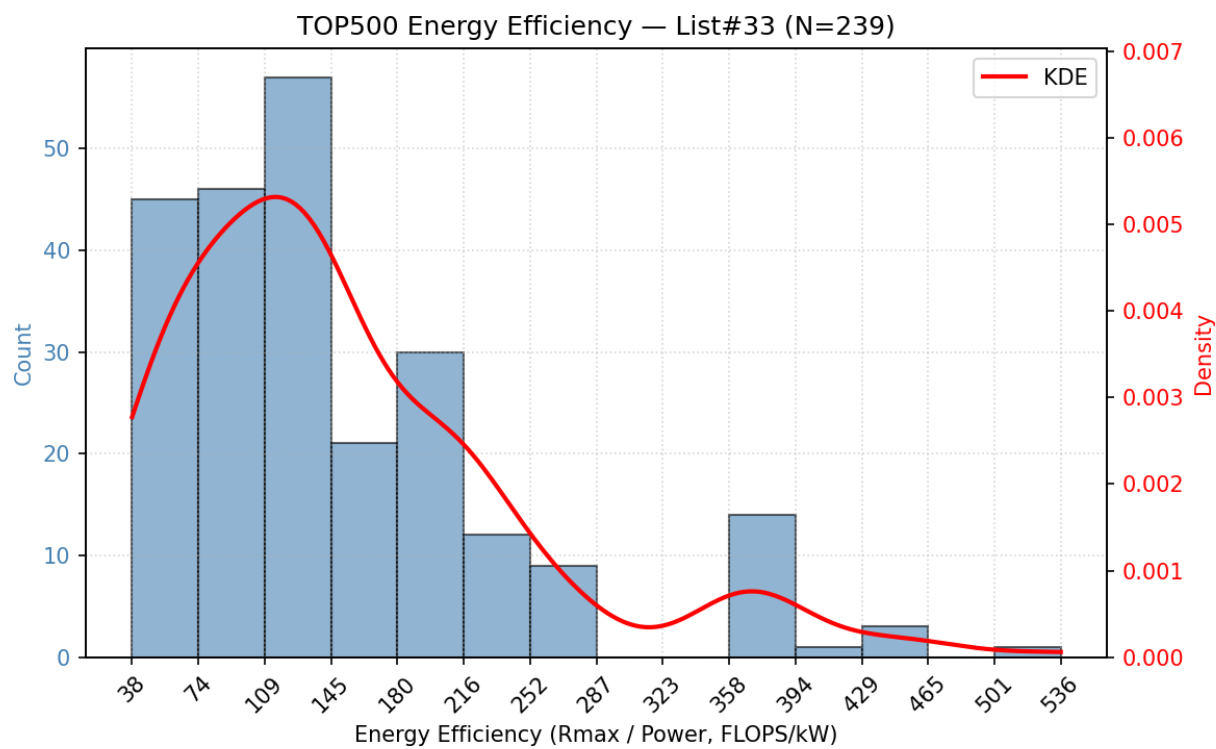


Рисунок 4 — Гистограмма энергоэффективности (Тор500, октябрь 2009, 33 рейтинг)

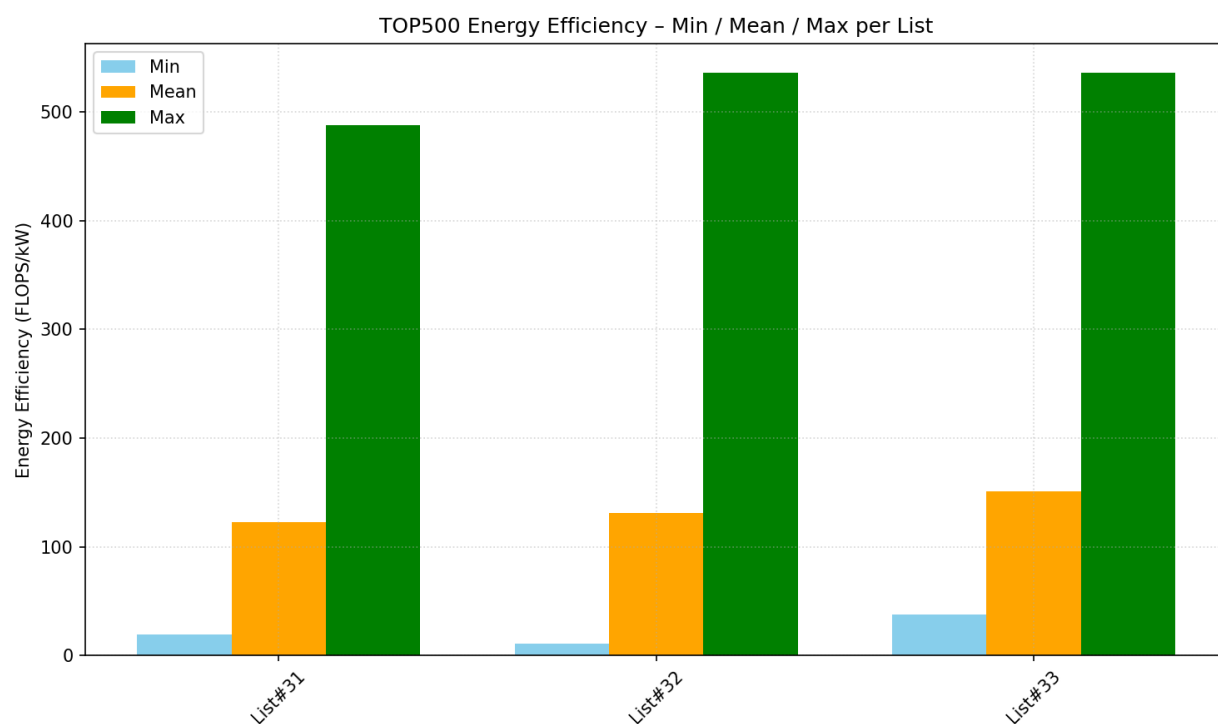


Рисунок 5 — Гистограмма энергоэффективности с минимальным, средним и максимальным значениями в каждом (31-33) рейтинге

Оценки функции распределения (Cumulative Distribution Function) представлены на рисунках 6-8.

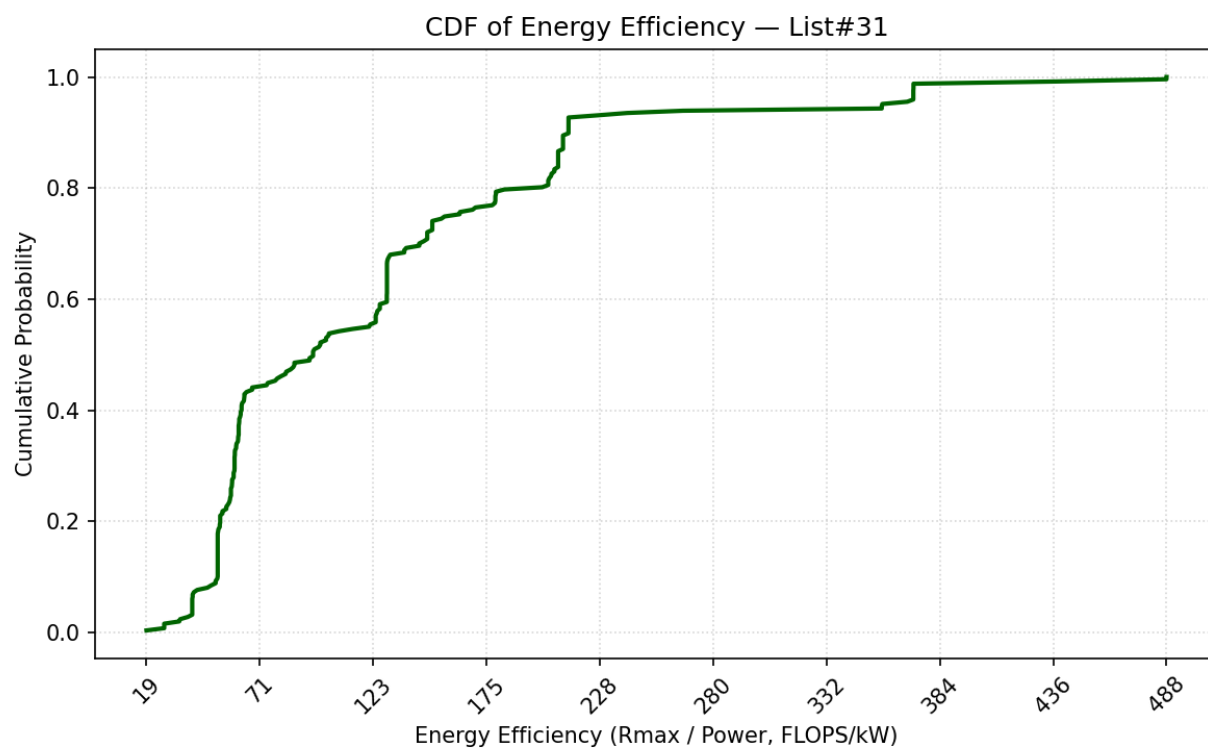


Рисунок 6 — Оценка функции распределения (Тор500, июнь 2008, 31 рейтинг)

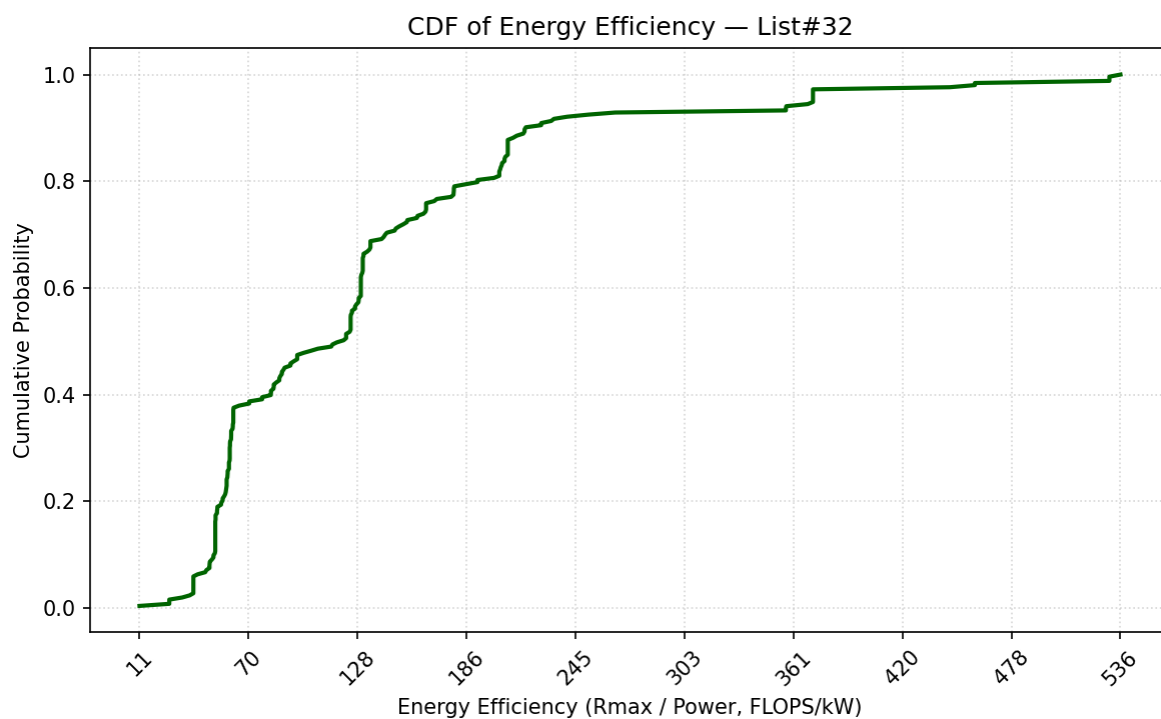


Рисунок 7 — Оценка функции распределения (Тор500, октябрь 2008, 32 рейтинг)

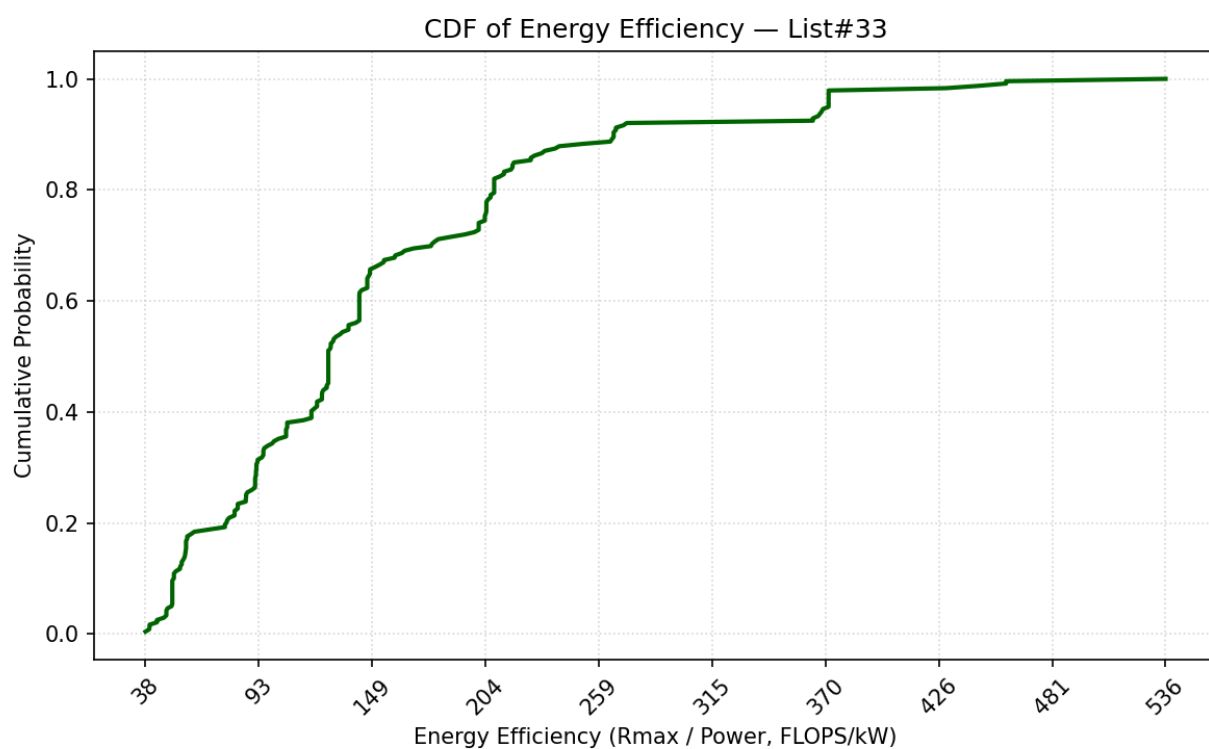


Рисунок 8 — Оценка функции распределения (Тор500, июнь 2009, 33 рейтинг)

Python-программа представлена в приложении 1. Входные данные (31, 32, 33 рейтинги Тор-500) представлены в приложении 2.

## 7. Выводы

Судя по полученным результатам, средняя энергоэффективность постепенно увеличивается. Максимальная энергоэффективность также растёт, появляются отдельные машины, значительно превосходящие остальные по эффективности. Минимальные значения остаются примерно на том же уровне. Прогресс идёт неравномерно — лидирующие системы (BladeCenter QS22 с PowerXCell 8i) заметно вырываются вперёд, что отражается в широком диапазоне значений энергоэффективности.

CDF: наклон кривой в диапазоне от ~50 до ~200 FLOPS/кВт довольно крутой — это значит, что большинство суперкомпьютеров (около 70%) находятся именно в этом диапазоне энергоэффективности. Основная масса систем имеет эффективность в пределах 50–200 FLOPS/кВт. После ~300 FLOPS/кВт кривая становится пологой, что говорит о том, что систем с более высокой энергоэффективностью намного меньше

Также нужно добавить, что результаты основываются лишь на исходных данных, которые были предоставлены в открытом доступе. Таких машин, не скрывающих свои показатели энергопотребления и Rmax, примерно половина в каждом рейтинге.

## Приложение 1. Python-программа

```
#!/usr/bin/env python3
"""
TOP500 Energy Efficiency Analysis for ratings 31, 32, 33 (2008-2009)

Dependencies:
    pip install requests lxml matplotlib pandas numpy scipy tabulate
"""

import os
import requests
from lxml import etree
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import gaussian_kde
from tabulate import tabulate

# ----- CONFIG -----
RANK_TO_DATE = {
    33: (2009, 6),
    32: (2008, 11),
    31: (2008, 6),
}
BASE_URL =
"https://top500.org/lists/top500/{year}/{month:02d}/download/TOP500_{year}
{month:02d}_all.xml"
OUT_XML = "xml"
OUT_CSV = "top500_efficiency.csv"
OUT_PLOTS = "top500_efficiency_plots"
TIMEOUT = 20
# -----

os.makedirs(OUT_XML, exist_ok=True)
os.makedirs(OUT_PLOTS, exist_ok=True)

# ----- Download -----
def download_xml(year, month):
    url = BASE_URL.format(year=year, month=month)
    fname = os.path.join(OUT_XML, f"TOP500_{year}{month:02d}.xml")
    if os.path.exists(fname):
        print(f"[SKIP] {fname} already exists")
        return fname, True
    try:
        r = requests.get(url, timeout=TIMEOUT)
        if r.status_code == 200:
            with open(fname, "wb") as f:
                f.write(r.content)
            print(f"[OK] {fname} downloaded")
            return fname, True
        else:
            print(f"[MISS] {url} ({r.status_code})")
            return None, False
    except Exception as e:
        print(f"[ERR] {url} -> {e}")
        return None, False

# ----- Parse XML -----
```

```

def parse_xml(path):
    """Return list of dicts with system, rank, power, rmax, efficiency"""
    with open(path, "rb") as f:
        xml_bytes = f.read()
    root = etree.fromstring(xml_bytes)
    ns_uri = root.nsmap.get("top500")
    if ns_uri is None:
        print(f"[WARN] Namespace not found in {path}")
        return []
    ns = {"top500": ns_uri}

    records = []
    for site in root.findall("top500:site", namespaces=ns):
        try:
            name = site.findtext("top500:system-name", namespaces=ns)
            if not name:
                name = site.findtext("top500:computer", namespaces=ns)
            rank = site.findtext("top500:rank", namespaces=ns)
            power = site.findtext("top500:power", namespaces=ns)
            rmax = site.findtext("top500:r-max", namespaces=ns)

            if name and rank and power and rmax:
                p = float(power)
                r = float(rmax)
                if p > 0 and r > 0:
                    records.append({
                        "system": name.strip(),
                        "rank": int(rank),
                        "power": p,
                        "rmax": r,
                        "efficiency": r / p # FLOPS per kW
                    })
        except Exception:
            continue
    return records

# ----- Plot Histogram + KDE -----
def plot_hist(df, label):
    arr = df["efficiency"].values
    fig, ax1 = plt.subplots(figsize=(8,5))

    counts, bins, _ = ax1.hist(
        arr,
        bins="auto",
        alpha=0.6,
        color="steelblue",
        edgecolor="black"
    )

    ax1.set_xticks(bins)
    ax1.set_xticklabels([f"{b:.0f}" for b in bins], rotation=45)

    ax1.set_xlabel("Energy Efficiency (Rmax / Power, FLOPS/kW)")
    ax1.set_ylabel("Count", color="steelblue")
    ax1.tick_params(axis='y', labelcolor="steelblue")

    ax2 = ax1.twinx()
    ax2.hist(arr, bins=bins, density=True, alpha=0.0)
    kde = gaussian_kde(arr)

```

```

xs = np.linspace(min(arr), max(arr), 200)
ax2.plot(xs, kde(xs), color="red", lw=2, label="KDE")
ax2.set_ylabel("Density", color="red")
ax2.tick_params(axis='y', labelcolor="red")

plt.title(f"TOP500 Energy Efficiency – {label} (N={len(arr)})")
ax2.legend(loc="upper right")
ax1.grid(True, linestyle=":", alpha=0.5)
plt.tight_layout()

fname = os.path.join(OUT_PLOTS, f"top500_efficiency_{label}.png")
plt.savefig(fname, dpi=150)
plt.close()
print(f"[PLOT] {fname} saved")

# ----- Summary plot -----
def plot_summary(df):
    df_valid = df[(df["efficiency"].notna()) & (df["efficiency"] > 0)]
    stats = df_valid.groupby("list_rank")["efficiency"].agg(["min", "mean",
"max"])
    labels = [f"List#{r}" for r in stats.index.values]

    plt.figure(figsize=(10,6))
    width = 0.25
    x = np.arange(len(labels))
    plt.bar(x - width, stats["min"], width, label="Min", color="skyblue")
    plt.bar(x, stats["mean"], width, label="Mean", color="orange")
    plt.bar(x + width, stats["max"], width, label="Max", color="green")

    plt.xticks(x, labels, rotation=45)
    plt.ylabel("Energy Efficiency (FLOPS/kW)")
    plt.title("TOP500 Energy Efficiency – Min / Mean / Max per List")
    plt.legend()
    plt.grid(True, linestyle=":", alpha=0.5)
    plt.tight_layout()

    fname = os.path.join(OUT_PLOTS, "top500_efficiency_summary.png")
    plt.savefig(fname, dpi=150)
    plt.close()
    print(f"[PLOT] {fname} saved")

# ----- CDF -----
def plot_cdf(df, label):
    """Plot empirical cumulative distribution function (CDF) for
efficiency."""
    arr = np.sort(df["efficiency"].values)
    cdf = np.arange(1, len(arr) + 1) / len(arr)

    plt.figure(figsize=(8,5))
    plt.plot(arr, cdf, color="darkgreen", lw=2)
    plt.xlabel("Energy Efficiency (Rmax / Power, FLOPS/kW)")
    plt.ylabel("Cumulative Probability")
    plt.title(f"CDF of Energy Efficiency – {label}")
    plt.grid(True, linestyle=":", alpha=0.5)

    # X ticks – реальные значения энергоэффективности
    xticks = np.linspace(min(arr), max(arr), 10)
    plt.xticks(xticks, [f"{x:.0f}" for x in xticks], rotation=45)

```

```

plt.tight_layout()
fname = os.path.join(OUT_PLOTS, f"top500_efficiency_cdf_{label}.png")
plt.savefig(fname, dpi=150)
plt.close()
print(f"[CDF] {fname} saved")

# ----- Main -----
def main():
    all_records = []
    for rank, (year, month) in RANK_TO_DATE.items():
        path, ok = download_xml(year, month)
        if not ok:
            continue
        recs = parse_xml(path)
        print(f"Rank {rank}, year {year}, month {month}: parsed {len(recs)}
records")
        for r in recs:
            r["year"] = year
            r["month"] = month
            r["list_rank"] = rank
        all_records.extend(recs)

    if len(all_records) == 0:
        print("[ERROR] No data collected! Check XML parsing.")
        return

    df = pd.DataFrame(all_records)
    df.to_csv(OUT_CSV, index=False)
    print(f"[CSV] Saved {OUT_CSV} ({len(df)} rows)")

    # Histogram per list
    for rank in sorted(df["list_rank"].unique()):
        df_rank = df[df["list_rank"] == rank]
        if len(df_rank) > 0:
            plot_hist(df_rank, f"List#{rank}")
            plot_cdf(df_rank, f"List#{rank}")

    plot_summary(df)

    # ----- Display table -----
    print("\n=== TOP500 Energy Efficiency Table ===")
    table = df[["list_rank", "rank", "system", "rmax", "power",
"efficiency"]]
    table_sorted = table.sort_values(["list_rank",
"rank"]).reset_index(drop=True)
    table_sorted_eff = table_sorted.sort_values("efficiency",
ascending=False)
    print(tabulate(table_sorted_eff.head(20), headers="keys",
tablefmt="github", floatfmt=".3f"))
    print("... (only first 20 rows shown)\n")

if __name__ == "__main__":
    main()

```

## Приложение 2. Входные данные

[https://top500.org/lists/top500/2008/06/download/TOP500\\_200806\\_all.xml](https://top500.org/lists/top500/2008/06/download/TOP500_200806_all.xml)

[https://top500.org/lists/top500/2008/11/download/TOP500\\_200811\\_all.xml](https://top500.org/lists/top500/2008/11/download/TOP500_200811_all.xml)

[https://top500.org/lists/top500/2009/06/download/TOP500\\_200906\\_all.xml](https://top500.org/lists/top500/2009/06/download/TOP500_200906_all.xml)