

3、代码：

import numpy as np

import matplotlib.pyplot as plt

# 参数设置

n\_values = [10, 20, 40, 80, 160, 320]  # 问题规模

p\_values = [1, 2, 4, 8, 16, 32, 64, 128]  # 处理器数量

alpha = 0.1  # 通信因子

# 初始化结果存储

results = []

# 计算加速比和效率

for n in n\_values:

    for p in p\_values:

        if p == 1:

            T\_parallel = n  # 单处理器时没有通信开销

        else:

            T\_parallel = n / p + alpha \* (p - 1)

        S = n / T\_parallel  # 加速比

        E = S / p  # 效率

        results.append((n, p, S, E))

# 数据分析和可视化

results = np.array(results)

for n in n\_values:

    data = results[results[:, 0] == n]

    plt.plot(data[:, 1], data[:, 2], label=f"Speedup (n={n})")

plt.xscale("log", base=2)

plt.xlabel("Number of Processors (p)")

plt.ylabel("Speedup (S)")

plt.title("Speedup vs Processors for Different Problem Sizes")

plt.legend()

plt.show()

for n in n\_values:

    data = results[results[:, 0] == n]

    plt.plot(data[:, 1], data[:, 3], label=f"Efficiency (n={n})")

plt.xscale("log", base=2)

plt.xlabel("Number of Processors (p)")

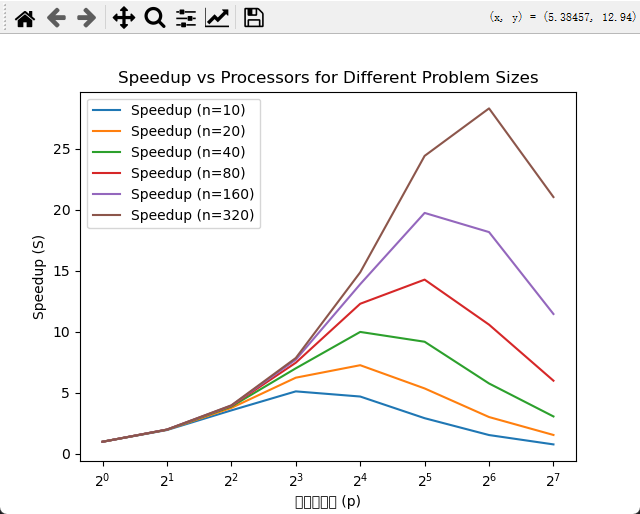
plt.ylabel("Efficiency (E)")

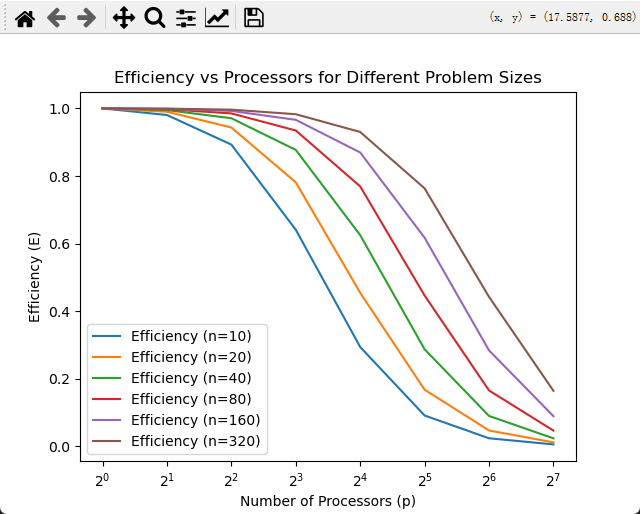
plt.title("Efficiency vs Processors for Different Problem Sizes")

plt.legend()

plt.show()

运行结果：





### ****(a) 加速比和效率的表现****

#### ****当**** n=10****时****：

**小规模处理器 (**p**小)**：

加速比S随着处理器数量p几乎线性增加，因为问题规模n很小，通信开销几乎可以忽略不计。

**处理器数量增加到一定程度 (**p**大)**：

随着p增加，加速比的增加速度减缓。这是因为通信开销逐渐占据主导地位，甚至当p从 64 增加到 128 时，加速比可能反而减少。

效率E的表现也反映了这一点：当p小时，效率接近1；当p增加时，效率迅速下降，表明通信开销显著增加，削弱了并行性能。

#### ****当**** n=320****时****：

**加速比 (**S**)**：

无论p多大，每次将处理器数量翻倍时，加速比几乎都会提高两倍。这是因为问题规模n足够大，通信开销相对于计算成本显得很小。

**效率 (**E**)**：

效率接近1，且随着p增加几乎不下降。这表明在大问题规模下，计算成本占主导地位，而通信开销的影响较小。

#### ****当****p****固定，而****n****增加时****：

* **小规模处理器 (**p**小)**：
  + 加速比S和效率E随着n的增加几乎保持不变。例如，当 p=2时，无论n是10还是320，效率都接近1，因为通信开销相对较小。
* **大规模处理器 (**p**大)**：
  + 随着n增加，加速比S和效率E会提高，因为更大的问题规模可以有效摊薄通信开销。
  + 然而，当n接近其最大值（例如320）时，加速比和效率的提升速度会逐渐减缓，因为通信开销仍然存在，并且已经接近计算性能的极限。

### ****(b) 效率公式及分析****

效率公式为：



其中：

* Tserial 是串行程序的运行时间。
* Toverhead是并行化带来的通信开销或其他额外成本。

#### ****分析：****

#### ****①如果通信开销增长速度比串行计算时间慢****：

随着问题规模n的增加，会变得越来越小。

效率公式中分母 变小，因此效率E随n的增加而提高。

**总结**：问题规模变大，计算时间主导运行开销，并行效率提高。

**②如果通信开销增长速度比串行计算时间快**：

随着问题规模n的增加，会变得越来越大。

效率公式中分母 增大，因此效率E随n的增加而降低。

**总结**：通信开销逐渐主导运行时间，导致并行效率下降。