sar-?????????CPU???????PDF??????????mailx?

你这个需求可以通过以下组合来实现：

? ???????`sysstat`??? sar ???

- 默认安装后已自动收集网卡、CPU、内存等历史数据，保存在 `/var/log/sysstat/saXX`。
- 确保服务启动：

  ```
  sudo systemctl enable sysstat
  sudo systemctl start sysstat
  ```

? ?????? + ?????Python ???

用 Python 自动读取 sar 数据，生成图表，并导出 PDF 报告：

?????

- 使用 `sar` 提取：
  - 网卡流量（rx/tx）
  - CPU 使用率（idle, user, system）
  - 内存使用（free, used, available）
- 使用 `matplotlib` 生成图表（折线图）
- 用 `reportlab` / `matplotlib.backends.backend_pdf` 导出 PDF 报告
- 最终用 `mailx` 自动发邮件

? ??Python ??????????????

? ????

```
sudo apt install sysstat mailx
pip install matplotlib pandas
```

? Python ??????????

```
import os
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
import subprocess
from datetime import datetime

# □□□CPU□□□□□□□□□
today = datetime.now().strftime("%d")
safile = f"/var/log/sysstat/sa{today}"
iface = "eth0"

def run_sar(cmd):
    result = subprocess.run(cmd, shell=True, stdout=subprocess.PIPE,
stderr=subprocess.DEVNULL)
    return result.stdout.decode('utf-8').splitlines()

# □□□□
def get_net_data():
    lines = run_sar(f"sar -n DEV -f {safile} | grep {iface}")
    data = []
    for line in lines:
        parts = line.split()
        if len(parts) >= 8 and parts[1] != 'IFACE':
            data.append([f"{parts[0]} {parts[1]}", float(parts[4]), float(parts[5])])
    df = pd.DataFrame(data, columns=["Time", "rx_kB", "tx_kB"])
    return df

# CPU □□□
def get_cpu_data():
    lines = run_sar(f"sar -u -f {safile}")
    data = []
    for line in lines:
        parts = line.split()
```

```
        if len(parts) >= 8 and parts[1] != 'CPU':
            data.append([f"{parts[0]} {parts[1]}", 100 - float(parts[7])])  # □□□ = 100 - idle
    df = pd.DataFrame(data, columns=["Time", "CPU_Usage"])
    return df

# □□□□□
def get_mem_data():
    lines = run_sar(f"sar -r -f {safile}")
    data = []
    for line in lines:
        parts = line.split()
        if len(parts) >= 5 and parts[1] != 'kbmemfree':
            total = float(parts[3]) + float(parts[1])
            used_pct = 100 * float(parts[3]) / total
            data.append([f"{parts[0]} {parts[1]}", used_pct])
    df = pd.DataFrame(data, columns=["Time", "Mem_Used"])
    return df

# □□□□□□□□□ PDF
def generate_pdf():
    pdf_file = "/tmp/system_report.pdf"
    with PdfPages(pdf_file) as pdf:
        for title, df, ycol in [
            ("Net Usage (kB/s)", get_net_data(), ["rx_kB", "tx_kB"]),
            ("CPU Usage (%)", get_cpu_data(), ["CPU_Usage"]),
            ("Memory Usage (%)", get_mem_data(), ["Mem_Used"]),
        ]:
            df.set_index("Time", inplace=True)
            df[ycol].plot(figsize=(10, 4), title=title)
            plt.xticks(rotation=45)
            plt.tight_layout()
            pdf.savefig()
            plt.close()
    return pdf_file

# □□□□
def send_mail(pdf_path):
    subject = f"Daily System Report - {datetime.now().strftime('%Y-%m-%d')}"
    cmd = f'echo "Please find the attached system usage report." | mailx -a {pdf_path} -s
"{subject}" you@example.com'
```

```
    subprocess.run(cmd, shell=True)


# 主逻辑
pdf = generate_pdf()
send_mail(pdf)
```

? ?????????crontab?

编辑定时任务：

```
crontab -e
```

添加：

```
0 8 * * * /usr/bin/python3 /opt/daily_report.py
```

每天 8 点自动发送日报邮件。

? ???????PDF ???

每页一个图表（如折线图）：

- **Page 1**: eth0 接收 /发送流量（ rx_kB/s, tx_kB/s）
- **Page 2**: CPU 使用率 %
- **Page 3**: 内存使用率 %

如需支持多网卡（ eth0, ens33, lo 等）、加上磁盘 IO、 TCP连接数、服务运行状态（如 nginx、 mysql ），都可以继续拓展脚本。

是否需要我帮你生成完整脚本并打包？或者加一个多主机日报支持？

Revision #1
Created 25 July 2025 03:01:42 by Admin
Updated 25 July 2025 03:05:56 by Admin