Large Scale Computing

Lab 3

1. Job script

```
#!/bin/bash
#SBATCH --job-name=blender_flower
#SBATCH --output=blender.out
#SBATCH --error=blender.err
#SBATCH --array=1-100
#SBATCH --time=00:30:0
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1GB
#SBATCH -p plgrid
#SBATCH -A plglscclass24-cpu

module load blender

BLEND_FILE="repeat_zone_flower_by_MiRA.blend"
OUTPUT_DIR="$SCRATCH/rendered_frames"

mkdir -p $OUTPUT_DIR

blender -b "$BLEND_FILE" -o "$OUTPUT_DIR/frame_" -F PNG -f $SLURM_ARRAY_TASK_ID

# Notify completion
echo "Rendering completed for frame $SLURM_ARRAY_TASK_ID. Results are in
$OUTPUT_DIR"
```

2. History

ID	Name	Partition	Nodes	Cores	Decl._mem	Mem._%_usage	Eff.	CPU._used	Wall._Used	Wall._Req.	End_Time
15402315_8	blender_flower	plgrid	1	4	4.0GiB	5.5%	94.9%	00:06:52	00:01:43	00:30:00	2025-03-26 15:06:24
15402315_2	blender_flower	plgrid	1	4	4.0GiB	5.6%	93.8%	00:07:00	00:01:45	00:30:00	2025-03-26 15:06:26
15402315_3	blender_flower	plgrid	1	4	4.0GiB	5.6%	94.1%	00:07:04	00:01:46	00:30:00	2025-03-26 15:06:27
15402315_7	blender_flower	plgrid	1	4	4.0GiB	5.4%	94.4%	00:07:08	00:01:47	00:30:00	2025-03-26 15:06:28
15402315_5	blender_flower	plgrid	1	4	4.0GiB	5.6%	94.2%	00:07:12	00:01:48	00:30:00	2025-03-26 15:06:29

To perform calculations I used a python script:

```python
import re


def parse_time_to_hours(time_str):
    h, m, s = map(int, time_str.split(':'))
    return h + m / 60 + s / 3600

file_path = "hpc-jobs-history.txt"

total_cpu_hours = 0
total_efficiency = 0
job_count = 0

with open(file_path, "r") as file:
    for line in file:
        # ID   Name   Partition   Nodes   Cores   Decl._mem   Mem._%_usage   Eff.
CPU._used   Wall._Used   Wall._Req.   End_Time
        match =
re.search(r"(\d+)\s+\S+\s+\S+\s+\d+\s+(\d+)\s+\S+\s+\S+\s+(\S+)%\s+(\d+:\d+:\d+)\s+(\d+:\d+
:\d+)", line)
        if match:
            cores = int(match.group(2))
            efficiency = float(match.group(3))
            cpu_time = parse_time_to_hours(match.group(4))
            print(match.group(1), match.group(2), match.group(3), match.group(4))
            total_cpu_hours += cpu_time
            total_efficiency += efficiency
            job_count += 1

average_efficiency = (total_efficiency / job_count) if job_count else 0


print(f"Total CPU-Hours Used: {total_cpu_hours:.2f} hours")
print(f"Average Efficiency: {average_efficiency:.2f}%")
```

3. Efficiency

Average efficiency ≈ 97.53%

Highest efficiency = 99.4%

4. CPU-hours

Estimated time for the whole animation ≈ 61.92 h