Assignment # 1

Submitted To

Mr. Farhad M. Riaz

Submitted By

Tayyaba Ejaz

Roll No

2751

Course

Parallel & Dist. Comp.

Due Date

October 16th, 2025

BSCS - 8th A, Morning

QUESTION # 1

A research team . . . . were chosen.

SOLUTION

ANALYSIS OF SUITABLE FLYNN'S ARCHITECTURE

For a real-time satellite image processing system, each image frame goes through multiple independent filtering operations such as edge detection, denoising and segmentation. These operations are highly data parallel, as the same computations are applied to many pixels simultaneously.

Therefore, choosing the right Flynn's architecture is critical to achieve high throughput, energy, efficiency and manageable hardware complexity.

JUSTIFICATION OF CHOOSING SIMD

Among all models, SIMD offers the best trade-off between throughput and hardware complexity.

In satellite image filtering, most operations perform identical mathematical functions on each pixel. SIMD efficiently executes the same instruction across multiple data points in parallel, maximizing throughput while keeping power consumption and control overhead low.

It provides excellent energy efficiency and computational density, making it ideal for real-time and power-sensitive image-processing applications.

Comparison of Flynn's Architectures

Architecture	Data Parallelism	Inst. Control	Power-Performance
SISD (Single Instruction Single Data)	- no parallelism - processes one pixel at a time	- Single control unit - very simple	- Low power but extremely slow - can't handle real time image streams
SIMD (Single Instruction Multiple Data)	- high data parallelism - same operation on multiple pixel simultaneously	- One control unit manages many processing elements.	- Excellent throughput per watt - Ideal for uniform & repetitive filtering tasks
MISD (Multiple Instruction Single Data)	- Little or no practical data parallelism	- Multiple inst. streams acting on same data.	- Very complex, rarely used. - inefficient for image filtering
MIMD (Multiple Inst. Multiple Data)	- Supports task-level parallelism. - Each core can process different data	- Each core has its own inst. control - Flexible but complex	- High performance but power-hungry. - less efficient for identical pixel-wise tasks

Potential Bottlenecks

SISD (Single Instruction Single Data)

- Processes one pixel at a time leading to very slow throughput
- Can't meet real time requirements
- Not scalabe for parallel workloads.

MISD (Multiple Instructions Single Data)

- Rarely used because not suitable for independent filters.
- Adds unnecessary control complexity
- Wastes hardware resources

MIMD (Multiple Instructions Multiple Data)

- High hardware & synchronization overhead.
- Higher power usage due to seperate control units.
- Less efficient for identical pixel operations

Possible memory contention between cores

CONCLUSION

So in summary, SIMD is the most suitable architecture for the given scenario because it efficiently handles the parallel, pixel-based operations in satellite image processing. It achieves high throughput with minimal control and power, offering the best balance between performance and hardware complexity as compared to other models.

QUESTION # 2

A data scientist's. . . . . .improve performance.

SOLUTION

GIVEN DATA

Data preparation = 15%

Training = 60%

Parallelizable training = 80%

training parallel = 0.60 x 0.80

= 0.48

Evaluation = 25%

Synchronization overhead = 10%

Number of cores = N = 8

1. Effective parallel fraction

Parallel portion of training = 0.48 {before sync}

Applying 10% synchronization

$$P_{eff} = 0.48(1 - 0.10)$$

$$= 0.48 \times 0.90$$

$$\boxed{P_{eff} = 0.432}$$

So 43.2% of whole workload is effectively parallel

2. Theoretical Speedup

Amdahl's law $\Rightarrow$ $$S = \frac{1}{(1 - P_{eff}) + \frac{P_{eff}}{N}}$$

$$= \frac{1}{(1 - 0.432) + \frac{0.432}{8}}$$

$$= \frac{1}{0.568 + 0.054}$$

$$= \frac{1}{0.622}$$

$$\boxed{S = 1.608x}$$

3. Effective Execution Time (relative to single-core)

Relative runtime $= \frac{1}{S}$

$= \frac{1}{1.608}$

$\boxed{\text{Relative runtime} = 0.622}$

So 8-core run takes about 62.2% of single-core time

4. Interpretation

Theoretical max speedup as $N \to \infty$

$$S_\infty = \frac{1}{(1 - P_{eff})}$$

$$= \frac{1}{1 - 0.432}$$

$$\boxed{S_\infty = 1.761}$$

So absolute ceiling is 1.761x

=> Doubling no. of cores to 16 give us:-

$$S_{16} = \frac{1}{(1-0.432) + \frac{0.432}{16}}$$

$$= \frac{1}{0.568 + 0.027}$$

$$= \frac{1}{0.595}$$

$$\boxed{S_{16} = 1.681}$$

Efficiency ~~of~~ at 8 cores

$$E = \frac{S}{N}$$

$$= \frac{1.608}{8}$$

$$\boxed{E = 0.201}$$

So, 20.1% utilization per core.

Therefore, even with 8 cores, speedup is limited to 1.6x because a large portion of task remains serial. Adding more cores would bring only minor performance gains due to remaining serial and synchronization overheads.

QUESTION #3

The execution. . . . . during training.

GIVEN DATA

Data preparation = 20% (non-parallel)

Training = 55% (90% is parallelizable) => training parallel = 0.55×0.90
$= 0.495$

Evaluation = 25% (10% is parallelizable) => 0.25×0.10
$= 0.025$

Cores = $N = 8$

Communication latency during training = 5%

1. Total Parallel fraction

Parallel part of training = 0.495

Parrallel part of evalution = 0.025

Total parallel fraction:

$$P_{total} = 0.495 + 0.025$$

$$\boxed{= 0.520}$$

So ~~55~~ 52.0% total work is parallelizable

2. Adjust for 5% training latency

Communication latency = $P_{eff} = 0.495 \times (1 - 0.05) + 0.025$

$$= 0.495 \times 0.95 + 0.025$$

$$= 0.47025 + 0.025$$

$$\boxed{P_{eff} = 0.4952}$$

So 49.52% is effective parallel after accounting for latency

CamScanner

3. Speedup with Amdahl's law

$$S = \frac{1}{(1 - P_{eff}) + \frac{P_{eff}}{N}}$$

$$= \frac{1}{(1-0.4952) + \frac{0.4952}{8}}$$

$$= \frac{1}{0.50475+0.06191}$$

$$\boxed{S = 1.76x}$$

4. Execution Efficiency

$$\text{Efficiency } E = \frac{S}{N}$$

$$= \frac{1.76}{8}$$

$$\boxed{E = 0.2206}$$

The system achieves an overall speedup of about 1.76x with 8 cores, but efficiency drops to around 22% due to limited parallelism & communication latency. This shows that adding more cores would result in minimal performance gains with additional cores.