Homework №2

Klauchek Maria

group M01-407

.

1 Part 1

1.1 Estimate the work and parallelism using θ-notation

1.1.1 Work

The work for parallel merge sort can be derived similarly to sequential merge sort, as it performs the same amount of operations in total.

1. Divide Step: Splits the array in half recursively, requiring $\log n$ levels of recursion for an input size n
2. Merge Step: Each level requires $\theta(n)$ operations to merge two sorted halves of size $\frac{n}{2}$

Since each level of recursion (from the split to the final merge) performs $\theta(n)$ work, and there are $\theta(logn)$ evels, the total work $W(n)$ is:

$$W(n) = \theta(n \log n) \tag{1}$$

1.1.2 Span

The span is the longest chain of dependent tasks, which limits the parallel speed-up.

1. Divide Step: In parallel merge sort, the two halves are sorted in parallel. Therefore, each level of the recursion requires only the time for one half (i.e., $\theta(\log n)$) levels of recursion, where each level only takes the time of a single split operation).
2. Merge Step: he merge operation at each level still takes $\theta(\log n)$ time in parallel (with concurrent merges).

Thus, the span $S(n)$ is $\theta(\log^2 n)$:

$$S(n) = \theta(\log^2 n) \tag{2}$$

1.1.3 Parallelism

Parallelism is defined as the ratio of work to span:

$$Parallelism = \frac{W(n)}{S(n)} = \frac{\theta(n \log n)}{\theta(\log^2 n)} = \theta(\frac{n}{\log n}) \tag{3}$$

2 Part 2

2.1 Using results of part 1, quantify real-life scaling effects. Answer the questions: how theoretical parallelism estimation correlates to measurement.

Theoretical parallelism estimation gives an idealized view of the parallel speed-up possible, assuming perfect conditions and no overhead. However, real-life scaling effects often deviate from these estimates due to various factors. Here's a breakdown of how theoretical and practical parallelism correlate, with a focus on factors that impact real-life scaling:

2.1.1 Hardware Constraints

1. Theoretical Model: Theoretical parallelism calculations assume an unlimited number of cores and no contention for resources.

2. Real-Life Scaling: Physical hardware limitations, such as the number of cores and memory bandwidth, cap the maximum achievable parallelism. For example, if you have 8 cores, the speed-up will plateau around an 8x factor (or even lower due to overhead). Theoretical models often predict near-linear scaling, which is not achievable once the number of tasks exceeds available cores.

2.1.2 Amdahl's Law and Sequential Bottlenecks

1. Theoretical Model: Theoretical parallelism does not consider the sequential portions of code that cannot be parallelized.

2. Real-Life Scaling: According to Amdahl's Law, the presence of any sequential portion limits the maximum speed-up. For example, if 95% of the code can be parallelized, the maximum speed-up is limited to 20x, regardless of how many cores are added. For parallel merge sort, the merging process itself may have sequential dependencies (like final merges), which reduce achievable speed-up as the array size grows.

2.1.3 Memory Bandwidth and Cache Effects

1. Theoretical Model: Theoretical models assume no limitations in data transfer rates or cache sizes.

2. Real-Life Scaling: In practice, memory bandwidth becomes a bottleneck as the number of cores accessing memory simultaneously increases. Furthermore,

cache hierarchies and limited cache size can affect scaling when data movement becomes excessive. When data doesn't fit into the cache, more cache misses and memory fetches can slow down the algorithm, impacting speed-up.

2.1.4 Establish a benchmarking system (use any available existing tools or create your own). Measure the real scaling factor and compare it to your estimation in part 1.

See code.
Results:

1. Close matching at lower thread counts: For 2–4 threads, the real speed-up might closely match the theoretical speed-up, as the overhead remains low.

2. Plateauing at higher thread counts: As the number of threads increases (e.g., 8 or 16), the speed-up likely begins to plateau due to overhead, limited memory bandwidth, and diminishing returns from task parallelism.

3. Theoretical overestimate: The theoretical speed-up may increasingly overestimate the achievable speed-up as threads increase, illustrating limitations in real-world parallel scaling (e.g., Amdahl's Law).

3 Describe the measurement methodology. Quantify observer effects (caches, overheads, HW-modes like hyper-threading, etc)

See code.
And observer effects:

1. Cache Effects: Quantified by increased cache misses at higher thread counts, visible in profiling tools.

2. Threading Overheads: Measured by increased context-switch rates and diminishing speed-up beyond physical core counts.

3. Hyper-Threading Effects: Observed through changes in cache usage, memory contention, and varying speed-up gains.

4. Memory Bandwidth: Limits scaling as thread count grows, measurable by monitoring memory usage and latency

4 Estimate the potential for improvement on the same HW (headroom study; use high-level parameters like TOPS)

1. Cache misses: Use more cache friendly algorithms. Analyze task synchronization overhead and cache misses. If these are high, reducing task granularity and improving data locality could free up additional headroom.

2. TOPS = Clock Speed (GHz) * Operations per cycle * Num of cores. For example, on a CPU with 3.5 GHz, 4 operations per cycle, and 8 cores: $TOPS = 112$. If observed performance is significantly below TOPS, there may be compute headroom achievable by optimizing CPU utilization, parallelism, or thread management.