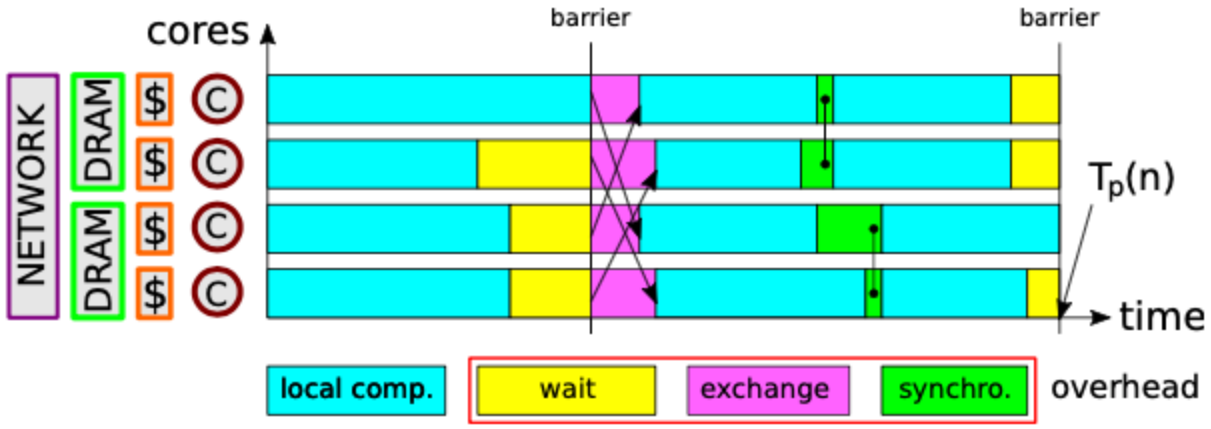
Session #2 Performance Analysis Problems

Problem #1 Cost, Efficiency and Speedup

Consider the following application model as introduced in class with balanced and imbalanced work.



(a) Assume we have 8 processors. The sum of local computations done by these processors is 1000 seconds and is identical to the sequential execution. All processors have a balanced local computations time and there is no waiting time between processors. Synchronisation adds 2 seconds for each processor. The data exchange is also balanced, with each processor executing 10 seconds of data exchange. Calculate the execution time, cost and efficiency.

(b) Assume now that the workload of processors are unbalanced such that the first processor executes 180 seconds of the sequential part. Data exchange time and synchronisation time are the same as in part (a). Calculate again the execution time, cost and efficiency.

Problem #2 Loop Scheduling

Assume a loop of N iterations such that each iteration executes in 10ns. In this problem we consider various types of schedulers. Consider the following overheads:

- For static scheduling there are no per-iteration overheads

- For dynamic scheduling we consider a fixed overhead each time a chunk is assigned, this overhead being 200ns.

Assume N = 1,000,000, and P=8
(a) What are the execution times on the three schedulers: static, self-scheduling and chunk. For the chunk scheduler consider a chunk size of k=1000.

Now assume that to create the parallel loop a fixed overhead is to be paid. This includes the generation of all the threads needed for the parallel execution and the destruction of the threads after the execution. The overhead depends on the number of processors and the time cost is $T = 50us + P \cdot 5us$.
(b) Calculate the execution time for the static, self-scheduling and chunk schedulers considering P=16, 32 and 64

(c) Consider the case of static scheduling: after which value of *p* does adding new processors result in a slowdown instead of a speed-up?

Problem #3 Amdahl's law and Gustafson's law

The execution time of a sequential program is 1000 sec. The program consists of a parallel loop that can be parallelized and which runs in 950 seconds.

(a) Assume a user wants to parallelize this application on a machine with 10, 100 or 1000 processors. What is the maximum speedup the programmer can expect for each case? What is the upper limit?

(b) Now assume that in order to parallelize the kernel, it is necessary to setup new data structures that add a sequential component to the application. This new sequential component adds a time of $T = 10sec + 0.1sec \times P$ where P is the number of processors. Calculate again the speedup for the three machines. Explain the result.

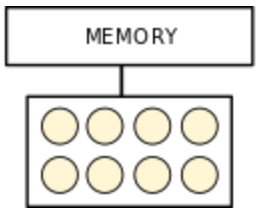
(c) The serial run of the problem has a dependency with the size of the problem n such that $T = 50 + n \cdot 10sec$. For the base case of one processors and n=95 the execution time is the aforementioned 1000 sec, which is satisfactory for the user.

What problem sizes can be addressed with P=10, 100 and 1000 without increasing the execution time? Assume that the overheads of thread creation are negligible. How would you solve it taking into account the overheads? Can you explain the result?

Problem #4 Roofline Model

A team of researchers is considering purchasing a new computing system to run a scientific application in the fastest possible time. The team has analyzed the application and observed that it consists of two phases: (a) a serial phase, which runs for 0.5 second on a single core, and (b) a parallel phase, which -with the current algorithm- consists of 16G ($16 \times 1024 \times 1024 \times 1024$) floating point (FP) operations. All FP operations can be executed in parallel. The parallel phase has a streaming behavior and there is no reuse of data, so caches have no impact. The arithmetic intensity of phase (b) is 0.4 Flops/Byte according to the (simplified) DRAM Roofline model.

The team owns an 8-core machine (shown below) which they use to run this code. The team is wondering how to improve the performance and are considering three options: (a) acquire a more powerful system, (b) further optimize the algorithm, or (c) change to a more high performance algorithm. The problems below explore these options:



The peak performance throughput for each core on the CMP is 0.5GFLOPS. The memory bandwidth of the multicore system is 16GB/s.

(a) Currently, the execution time of the system is 8.5 seconds. What can you say about the level of optimization?

(b) By via of SIMD parallelism, the programmers have been able to double the performance of the parallel part. In this new regime, what is the next bottleneck that should be addressed: Further optimization, Compute FLOPS or Memory (GB/s)?

(c) Assume that the developers have bought a 16-core machine, with each core identical to (b). Repeat problem (b).

(d) One of the developers has devised a new algorithm that reduces the amount of operations from 16G to 4G. But the AI is reduced from 0.4 to 0.2. Given the updated 16-core system, will this new algorithm achieve any improvements?

Problem #5 CPU performance

Consider two methods A and B to simulate the evolution of a galaxy cluster composed of n stars. Method A performs 10 x n^2 floating point operations while method B performs 1000 x n log2(n) operations. The number of control and integer instructions is 1000 x n.

Due to different instruction mixes, the CPI for method A is CPI(A) = 1.0, while for method B it is CPI(B) is 1.5, meaning the processor takes 50% more time in order to execute an instruction for method B. For the remaining non-algorithmic instructions, assume CPI=1.0

(a) What is the execution time (in cycles) for both methods when n=10^3? How about n=10^4

(b) What is the MFLOPS rate achieved in each case? Calculate the MFLOPS according to each method (ie. no fixed flops formula). Assume the cycle time $t_{cycle} = 1ns$

(c) Take the previous exercise and assume a real memory hierarchy with a single-level cache where we assume that cache hits do not take additional cycles. Assume that the number of loads and stores for method A is $3.5 \cdot 10^4 \cdot n$. while for method B there are $5 \cdot 10^4 \cdot n$ loads and stores. Assume also that only the cache miss latency adds to the execution time. The miss rate for method A is 1% and the miss

rate for method B is 2%. Accesses to main memory take 100ns. Repeat subsection (a) of Problem #1. What do you conclude?