



# **Review of previous examinations**

## TMA4280—Introduction to Supercomputing

NTNU, IMF

April 20. 2018

# Examination



The examination is usually comprised of:

- one problem related to linear algebra operations with calculation of complexity and parallelism.
- one problem related to parallelization of solver for partial differential equations.
- a series of short questions covering all the topics of the course.

to be solved in four hours.

## Important points



The course covered a broad range of topics, but a few points are important:

- Differentiate between the different types of parallelism (data, functional/task, and fine-/coarse-grained).
- Be able to categorize hardware architectures, e.g with Flynn's taxonomy and understand their benefits/drawbacks.
- Understand the memory hierarchy of computers and how it affects the performance of solvers.
- Be able to define clearly and use the distributed and the shared-memory model.
- Have a basic knowledge of the different specifications (OpenMP, MPI, BLAS) and libraries (LAPACK, PETSc) discussed.

## Notions used in Problems

Study of parallel performance of algorithms.

**Amdahl's law:** speed-up on  $p$  processor w.r.t serial

$$S_p = \frac{T_1}{T_p} = \frac{p}{f(p-1) + 1}$$

with  $f$  fraction of time spent in serial sections of the code.

The fraction  $f$ :

—  $\rightarrow 1$  for purely serial case

—  $\rightarrow 0$  for ideal parallel case

Linear strong scaling:  $f = 0$

$$S_p = \frac{T_1}{T_p} = p$$

ideal speed-up when solving a fixed problem on  $p$  processors.



## Notions used in Problems



Some notions are important and will be used during the examination.

Linear model for transmission:

$$(1) \quad \tau_c(k) = \tau_s + \gamma k$$

with  $\tau_s$  a constant that is the start-up phase and  $\gamma$  the inverse bandwidth.

**Amdahl's law:** speed-up on  $p$  processor w.r.t serial

$$S_p = \frac{p}{p \frac{T_c}{T_1} + 1}$$

with  $T_c$  the time spent in communications, a function of  $\tau_c$ .

# Challenges of parallel computing

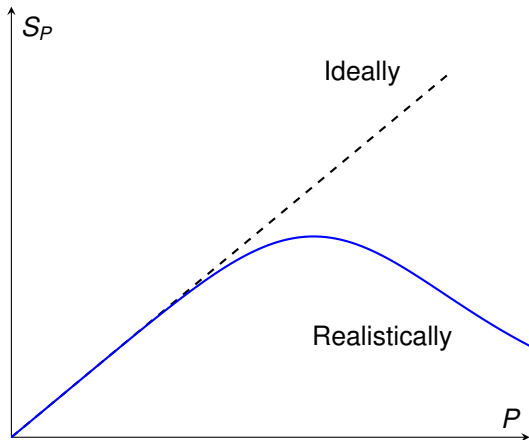


Figure: Ideal speedup ( $S_P = P$ ) and realistic speedup.

- strong scaling should be interpreted carefully!
- weak scaling should also be considered.

## Questions: Parallel computing

### Question (2016)

*A good strategy for reducing communication overhead is to increase the number of processes. Answer: true or false*

It is crucial to remember Amdahl's Law, and for instance the practical effect observed in your project when studying the strong scaling.

### Question (2015)

*A code with a large parallel efficiency typically has much network traffic.*

The same arguments apply.

### Question (2015)

*A code with a large parallel speedup has a large parallel efficiency.*

The same arguments apply.

## Questions: Computer architecture

### Question (2016)

*The following loops are compiled with an optimizing compiler:*

- Which of them will likely have the highest performance (more FLOPS)?*
- The following loops are compiled with an optimizing compiler. Which of them will likely have the highest performance (more FLOPS)?*

*Answer: A, B or none. Briefly explain why.*

A:

```
for (int i = 0; i < N ; i ++)  
    a[i] = a[i] + b [i];
```

B:

```
for (int i = 0; i < N ; i ++)  
    a[i] = a[i] + c * b [ i ];
```

Remember the definition of FLOPS and the capabilities of processor when it comes to floating-point operations.



## Questions: Computer architecture

### Question (2015)

*A ccNUMA machine can always do multiple additions in parallel. Answer: true or false*

Remember Flynn's taxonomy and the discussion on different computer architectures.

### Question (2015)

*An LFU cache replacement policy is typically the best for solving partial differential equations*

The question was briefly discuss in the description of different caching policies: First-In First-Out (FIFO), Least-Frequent Used (LFU), Least-Recently Used (LRU) and the pseudo-LRU. Even without remembering the details of the implementation, you can think about the memory access pattern.

## Questions: Computer architecture

### Question (2015)

*A SIMD processor can perform a multiplication and an addition simultaneously. Answer: true or false.*

Remember Flynn's taxonomy to motivate your answer.

### Question (2015)

*Cancellation is a concern when subtracting floating point numbers. Answer: true or false.*

Remember the binary representation of floating-point numbers and how it affects the decimal precision.

### Question (2014)

*Floating point numbers of a given precision can only represent a fixed range of numbers.*

Remember the binary representation of floating-point numbers.

## Questions: Computer architecture



### Question (2014)

*A modern processor typically has a cache hierarchy. These are designed as levels, where a higher level is given to faster memory.*

Remember for instance the architecture of modern CPUs and the different memory levels.

### Question (2014)

*A superscalar processor can perform two additions simultaneously.*

Remember the discussion about pipelining.

## Questions: Distributed memory

### Question (2016)

- *It is always OK to call MPI library functions from different threads.  
Answer: true or false.*
- *It is never OK to call MPI library functions from different threads.  
Answer: true or false.*

`MPI_THREAD_SINGLE`

Only one thread will execute.

`MPI_THREAD_FUNNELED`

The process may be multi-threaded, but only the main thread will make MPI calls (all MPI calls are funneled to the main thread).

`MPI_THREAD_SERIALIZED`

The process may be multi-threaded, and multiple threads may make MPI ca

`MPI_THREAD_MULTIPLE`

Multiple threads may call MPI, with no restrictions.

In practice `MPI_THREAD_SINGLE` only is portable, see Balaji (2010) for a discussion.

## Questions: Distributed memory

### Question (2011)

*Consider the MPI-function below: the amount of data sent corresponds to 128 floating point numbers in double precision. Answer: true or false.*

```
MPI_Send(buffer, 1024, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);
```

Writing the code for the projects should make this question easy.

### Question (2011)

*The most efficient implementation of the `MPI_Allreduce` operation on  $P$  processors completes in a certain number of communication stages. Which of the 4 alternatives is correct: (i) 1 stage; (ii)  $\log_2(P)$  stages; (iii)  $2 \log_2(P)$  stages; (iv)  $P$  stages*

Remember your implementation in Project I.

## Questions: Distributed memory



### Question (2011)

*The functions `MPI_Send` and `MPI_Recv` are appropriate to use for the exchange of an interprocess boundary. Is it possible to experience "deadlock" when using these functions? Answer: yes or no.*

Knowledge of point-to-point communication is sufficient.

## Questions: Shared-memory model



### Question (2016)

*Since threads do not need to communicate (like processes do), there is no penalty incurred by using more of them. Answer: true or false.*

Remember how OpenMP works and the difference between concurrency and parallelism.

### Question (2014)

*OpenMP is usable from all programming languages.*

Just enough to know about the OpenMP specification.

# Questions: Numerical Linear Algebra



## Question (2016)

*BLAS is a high performance library for solving linear systems of equations. Answer: true or false.*

Remember how the specification was introduced during the lecture. Hint: different level of algebraic operations.

## Question (2015)

*In code utilizing dense linear algebra, you have to choose between using BLAS or LAPACK. Answer: true or false.*

Similar question, the relation between BLAS and LAPACK is important to remember.



# Questions: Numerical Linear Algebra



## Question (2015)

*Using PETSc there is no point in pre-declaring the sparsity pattern of your operator. Answer: true or false.*

Review how the storage for a sparse matrix is handled and how it may affect the performance.

## Question (2014)

*You typically get performance closer to the theoretical peak performance of a machine when you do level 1 operations (vector operations) compared to level 3 (matrix-matrix operations).*

Think about the ratio between operations and data movement.

## Questions: Parallel Input/Output



### Question (2015)

*MPI-I/O is often used to do post-mortem data assembly. Answer: true or false.*

The exact term as not been mentioned during the lecture but remember how the MPI-I/O implementation works.

### Question (2014)

*MPI-I/O always writes multi-dimensional arrays in Fortran order.*

Just remember the purpose of MPI-I/O.