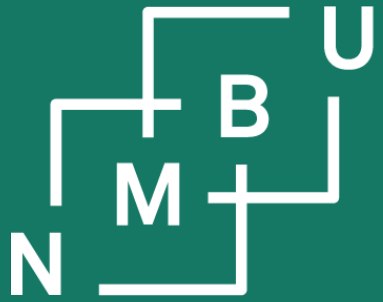
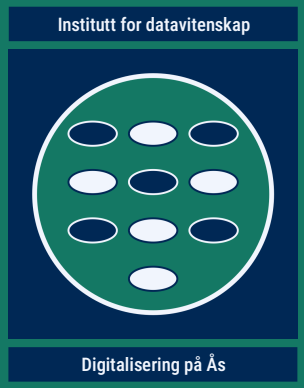


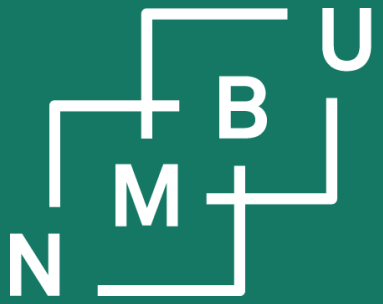
Norges miljø- og biovitenskapelige universitet



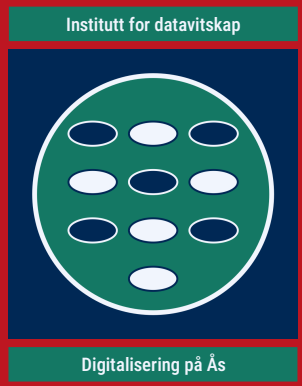
INF205
Resource-efficient programming

5 Parallel data

5.5 Resource efficiency metrics
5.6 Requirements modelling
5.7 Load balancing



Noregs miljø- og biovitskaplege universitet



5 Parallel data

5.5 Resource-efficiency metrics

INF205

30th November 2022

Requirements: The traditional view

In most cases, discussion of computational resources limits itself to "**space**" and "**time**." This is also motivated by tradition in theoretical computer science. In practice, then, *time usually becomes the main performance metric*, whereas *space becomes the main bottleneck* (memory access, communication, file I/O).

Strong scaling (**Amdahl**, *constant* problem size) on parallel architectures:

- Runtime reduction as number of processes increases (ideally, linear).
- Total CPU time increase as there are more processes (ideally, none).
- Rate of CPU operations (*e.g.*, FLOP/s) as fraction of peak performance.
- *Amdahl's law: Deterioration of performance at some point is inevitable.*

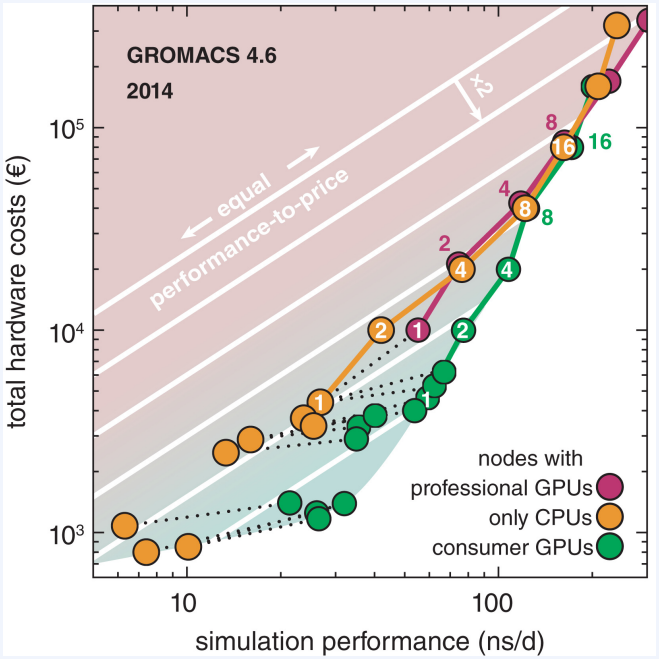
Weak scaling (**Gustafson**, *proportional* problem size) on parallel architectures:

- CPU time per problem size as problem and core usage are scaled up.
- Runtime increase during the scale-up.
- Rate of CPU operations (*e.g.*, FLOP/s) as fraction of peak performance.
- *Some algorithms and codes don't show a major decay in these metrics.*

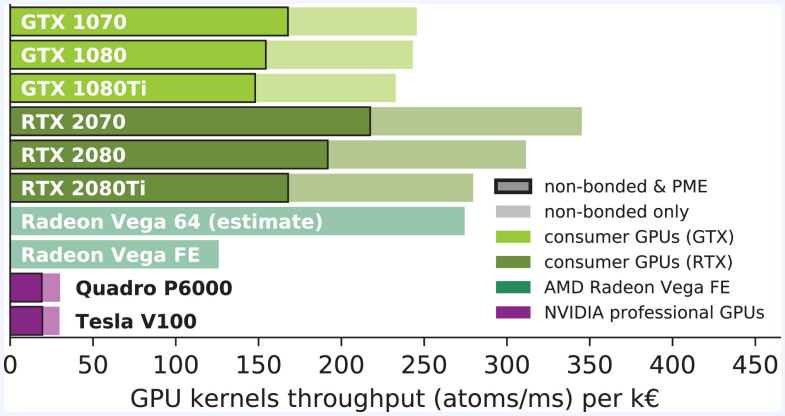
Economic metrics

Investment and operational costs can be considered. For an analysis of investment costs[1] to be reasonably actionable, it must include multiple representative use cases.

Operational costs usually also have a major ecological aspect (computational and cooling electricity cost). They might be taken into account for scheduling/workflow management.



Example on the right: C. Kutzner *et al.*, "More bang for your buck: Improved use of GPU nodes for GROMACS 2018," *J. Comp. Chem.* **40**(27): 2418–2431, doi:10.1002/jcc.26011, **2019**.



Ecological and socioeconomic metrics

How about bitcoins, and maybe similar blockchain technologies such as NFTs?

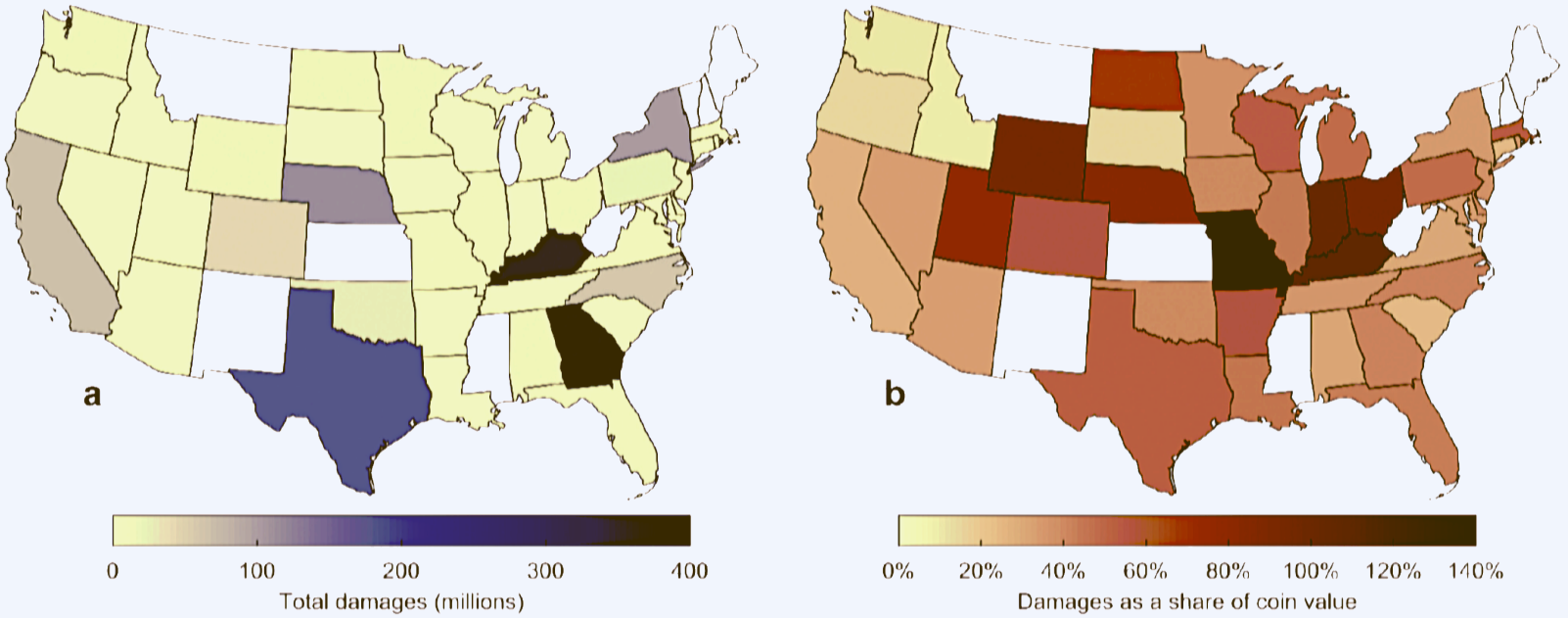


Figure 2. BTC mining damages (climate plus human health) across US states. Panel **a** (left): total damages of BTC mining between 1 September 2019 to 31 December 2021. Panel **b** (right): average damages per BTC mined as a share of the market price of the coin. States in white did not mine BTC over this period according to the CBECI dataset.

From A. L. Goodkind *et al.*, *Appl. Econ. Lett.*, doi:10.1080/13504851.2022.2140107, **2022**.

Ecological and socioeconomic metrics

Norwegian University of Life Sciences

Key performance indictators (KPIs) for ecological performance[1] can include:

- Abiotic-resource depletion potential (ADP)
- Cumulative energy consumption
- Greenhouse warming potential, including from use of refrigerants
- Water consumption
- *DCiRE: Contribution of building infrastructure to these categories*

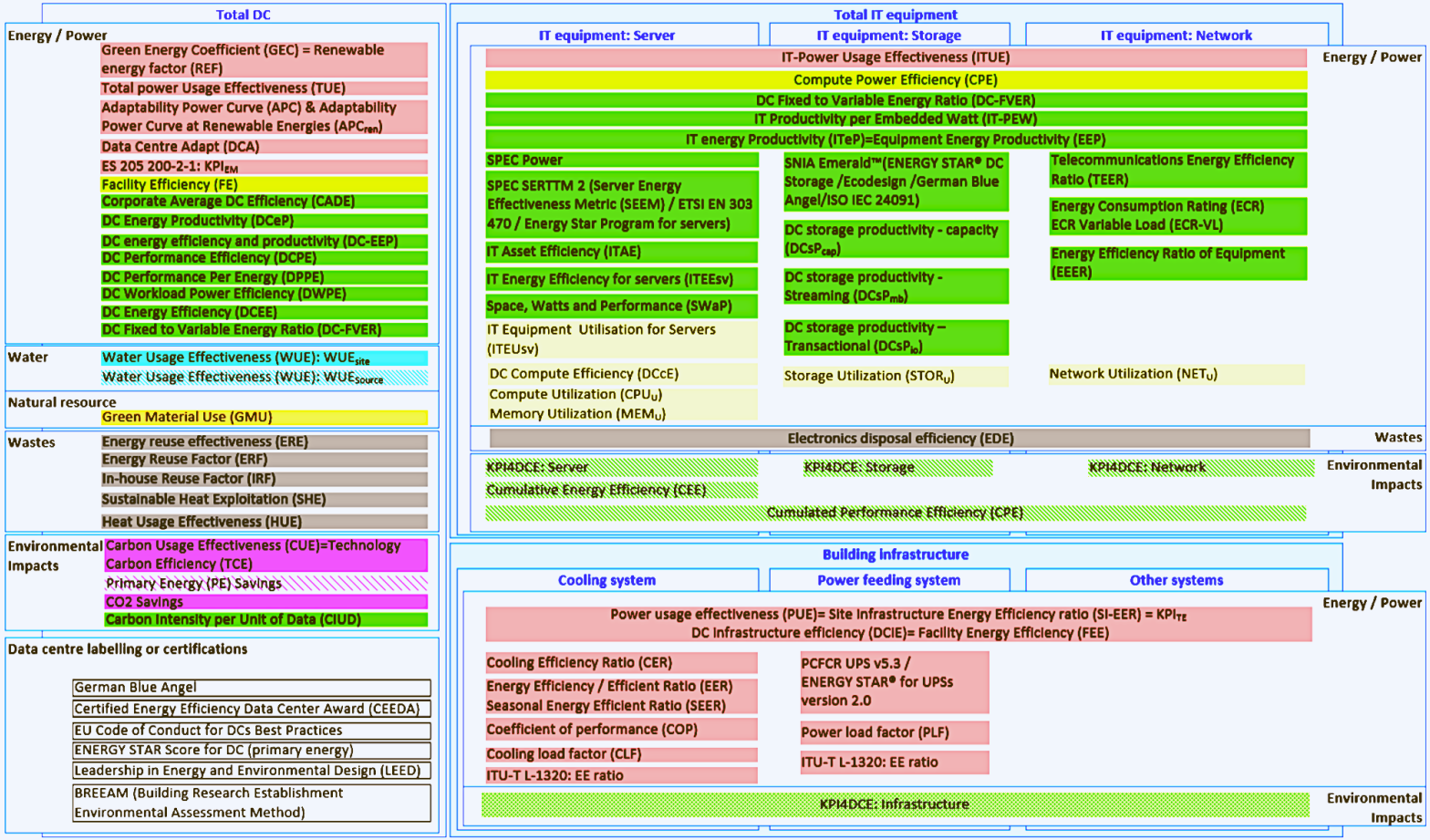
Blue Angel: "Type I environmental label" (ISO 14024) based on full life-cycle-analysis, targeting customers.[2]

EC explores introduction of additional Type II and III labels.[3]

[1]B. Schödwell, R. Zarnikow, KPI4DCE report, UBA no. 19/2018, *pp.* 25, 154, **2018**.
[2]See *e.g.* https://www.hlrs.de/about/certifications.
[3]EC report "Study on Greening Cloud Computing […]," doi:10.2759/116715, *pp.* 128, 289, **2022**.

Ecological and socioeconomic metrics

Table 16: Overview of 71 selected metrics and 6 DC-relevant labelling or certification scheme



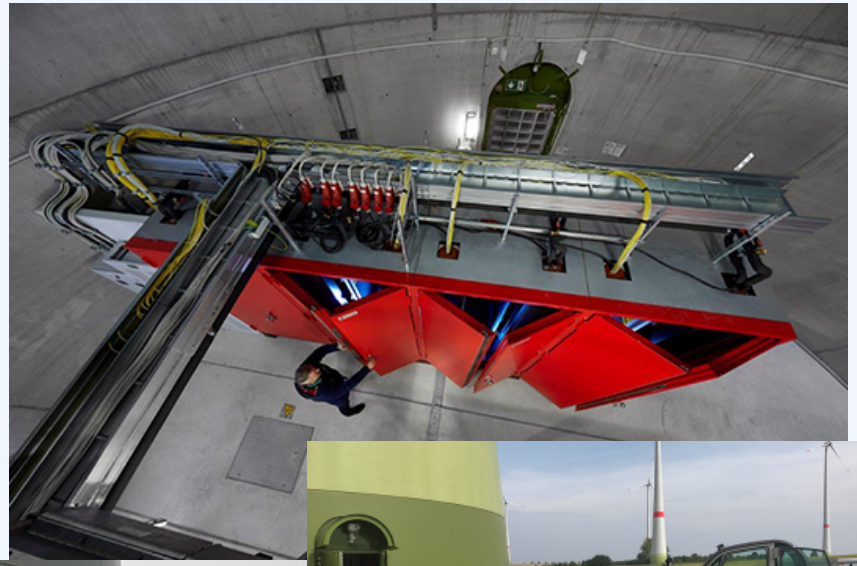
p. 98, "Study on Greening Cloud Computing […]," EC report, doi:10.2759/116715, **2022**.

Example: Ideas for resource efficiency

Norwegian University of Life Sciences

WindHPC based on pre-existing solution WindCores[1] from WestfalenWind GmbH: Computing nodes (and data centre nodes) integrated into the structure of wind turbines. Slogan: "Power to bytes."

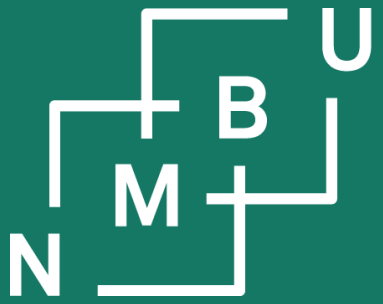
(BMBF project "In Windkraft-anlagen integrierte Second-Life-Rechencluster")



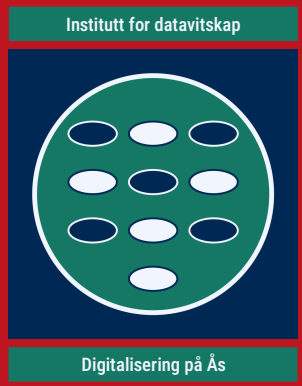
windCORES
WestfalenWIND IT

Federal Ministry of Education and Research

[1] https://www.windcores.de/



Noregs miljø- og biovitskaplege universitet



5 Parallel data

5.5 Resource efficiency metrics

5.6 Requirements modelling

INF205

30th November 2022

Requirements: The traditional view

Norwegian University of Life Sciences

Assume that algorithms A and B both solve the same problem (problem size: n).

Algorithm A

Runtime: ~~200 µs · n^2 + 1 s~~ **O(n^2) time!**

Memory: ~~8 KB · n~~ **O(n) space!**

Algorithm B

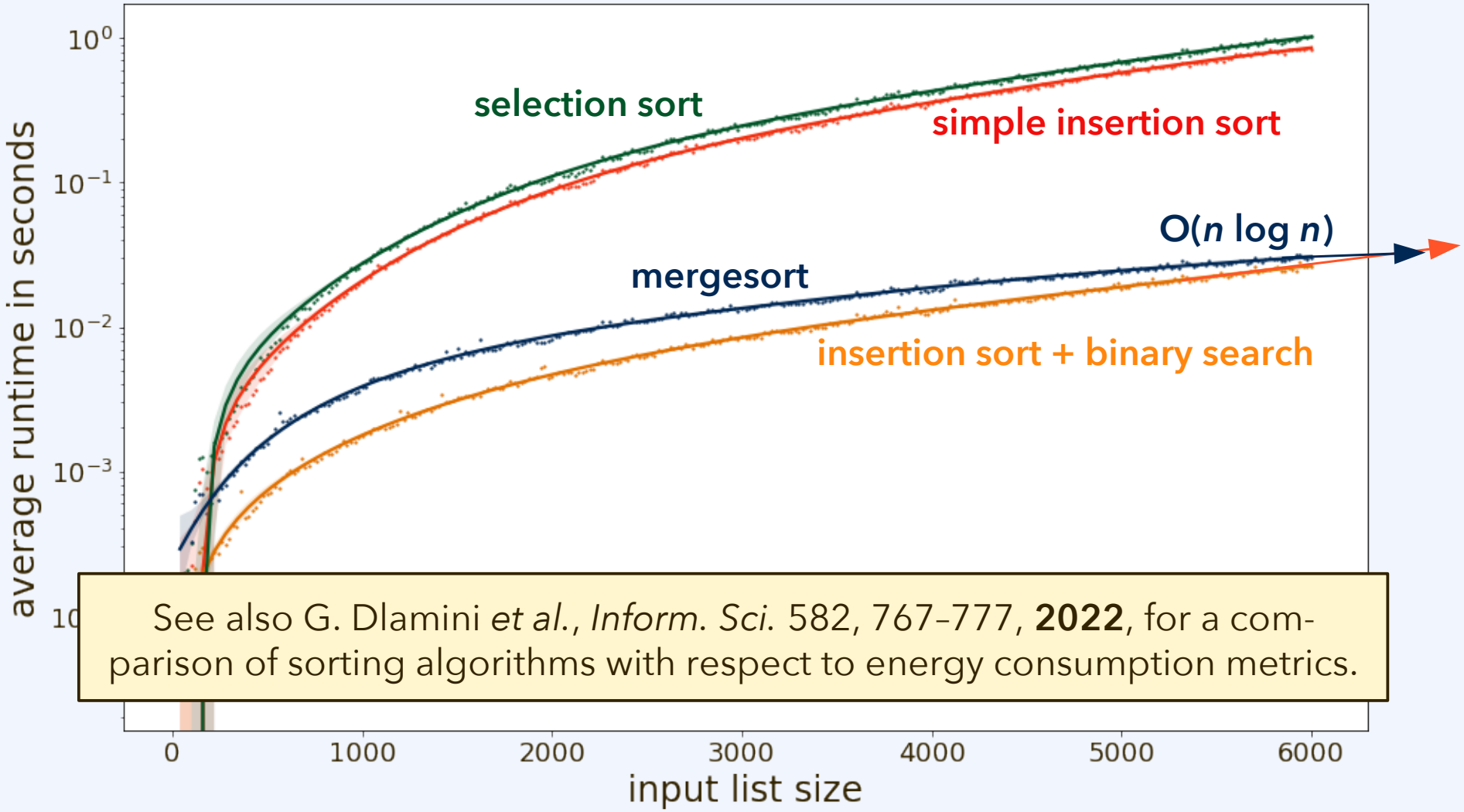
Runtime: ~~1 µs · n^3 + 100 µs · n^2 + 1 s~~ **O(n^3) time!**

Memory: ~~64 byte · n^2~~ **O(n^2) space!**

The main purpose of asymptotic efficiency/performance analysis with O(n) notation is *not even about algorithms*. It is done to *classify problems into complexity classes*: **Problem complexity = Best algorithm's asympotic efficiency**.

Yes: It is important to know, for example, whether a problem is NP-complete.
But: That knowledge alone does not help when we need to solve the problem.

Limitation of the traditional view



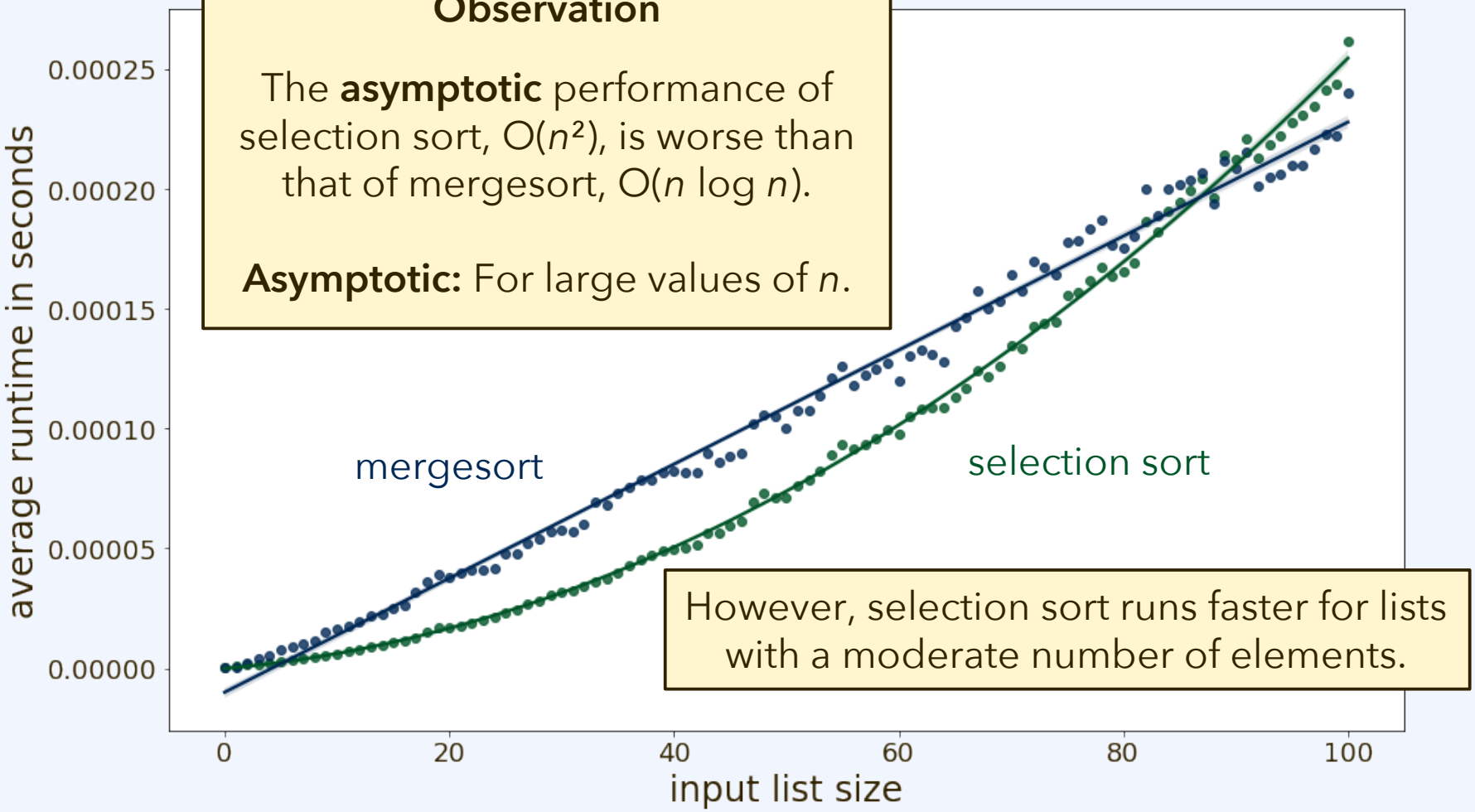
See also G. Dlamini *et al.*, *Inform. Sci.* 582, 767–777, **2022**, for a comparison of sorting algorithms with respect to energy consumption metrics.

Limitation of the traditional view

Observation

The **asymptotic** performance of selection sort, $O(n^2)$, is worse than that of mergesort, $O(n \log n)$.

Asymptotic: For large values of n.



However, selection sort runs faster for lists with a moderate number of elements.

Quantitative requirements modelling

Idea and prerequisites:

- The parameter space or **domain of the requirements model** is well defined, accounting for type and size of the input or use case and the execution conditions of the code (such as number of processes).
- We build a correlation or closed expression that serves as a model of the code, predicting its computational resource requirements. This can be:
  - Purely predictive, based on a theoretical analysis of the code. (Can always be done for a simplified model, if no data are available.)
  - Regression/parameterization of a model, known to be qualitatively right, to **performance data**. (Counts as supervised machine learning.)
  - Unsupervised machine learning from **performance data**.

Discussion: For what purpose can it be helpful to have a quantitative requirements model? In what ways might we use it in practice?

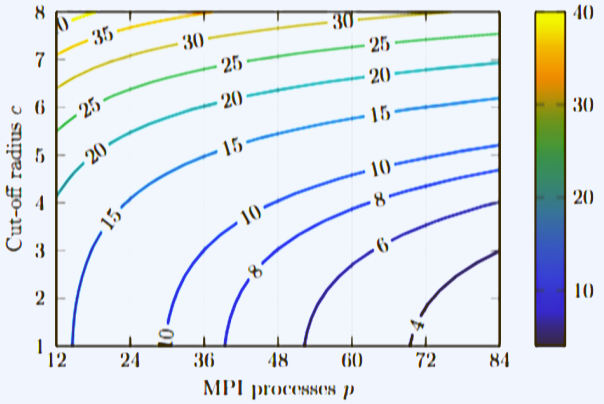
Quantitative requirements modelling

Table 2: 2-parameter models for the execution time of the *ms2* application.

Model	Fixed parameters	Model	$\bar{R}^2$
$T(n,m)$	$d=0.84,\ c=2.0,\ p=72$	$4.41+8.03\cdot10^{-5}\cdot m\cdot n\cdot\log n$	0.99
$T(p,m)$	$n=4{,}000,\ d=0.84,\ c=2.0$	$6.6+3.21\cdot m^2-0.42\cdot m^2\cdot\log p$	0.92
$T(p,d)$	$n=4{,}000,\ m=1,\ c=2.0$	$20.67-2.2\cdot\log p$	0.88
$T(p,c)$	$n=4{,}000,\ m=1,\ d=0.84$	$33.83+0.05\cdot c^3-4.89\cdot\log p$	0.79
$T(n,c)$	$m=1,\ d=0.84,\ p=36$	$-0.99+0.06\cdot c^3+1.81\cdot10^{-5}\cdot\log^2 n$	0.95
$T(m,c)$	$n=4{,}000,\ d=0.84,\ p=36$	$-23.49+10.09\cdot m+0.22\cdot c^3\cdot m$	0.95

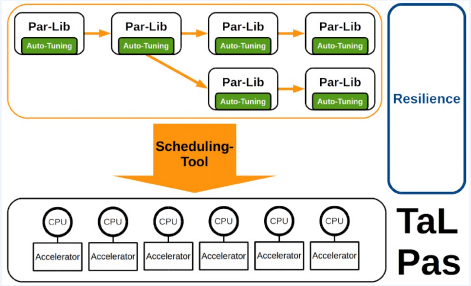
Table 3: 3-parameter models for the execution time of the *ms2* application.

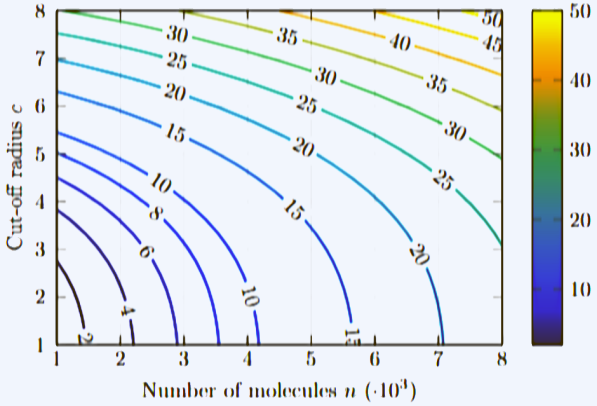
Model	Fixed parameters	Model	$\bar{R}^2$
$T(p,n,m)$	$d=0.84,\ c=2.0$	$62.28+2.03\cdot10^{-8}\cdot m^2\cdot n^{1.5}\cdot\log^2 n-9.63\cdot\log p$	0.83
$T(n,m,c)$	$d=0.84,\ p=72$	$9.24+5.71\cdot10^{-6}\cdot n\cdot\log n\cdot c^2\cdot\log c\cdot m$	0.88



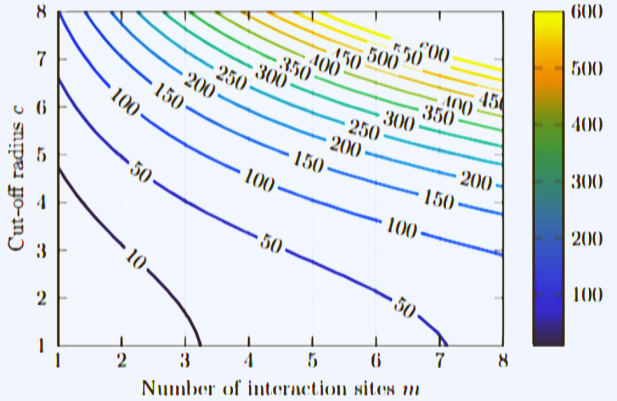
(d) $T(p,c)$

Federal Ministry of Education and Research





(e) $T(n,c)$

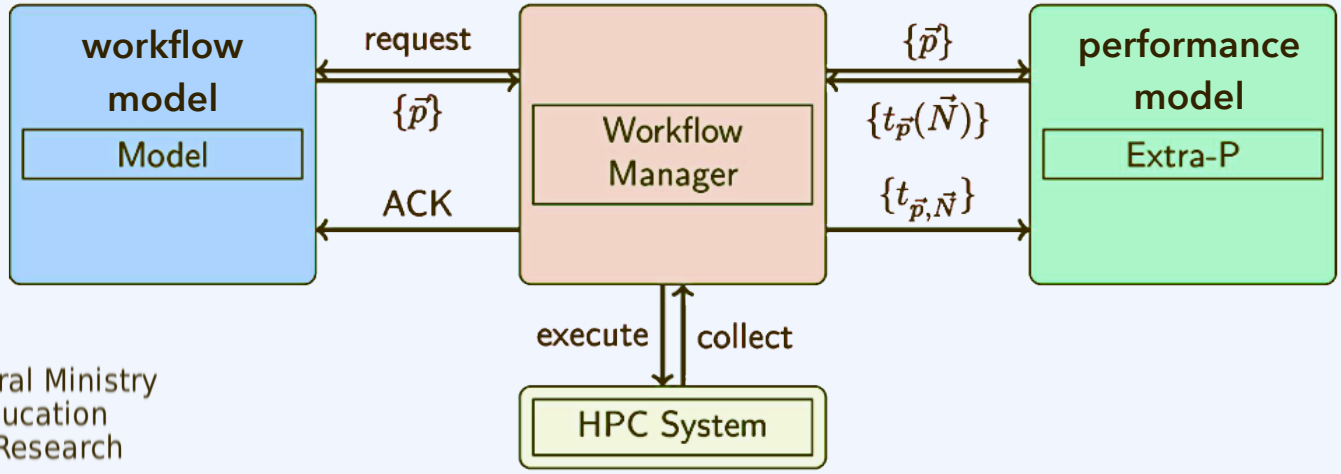


(f) $T(m,c)$

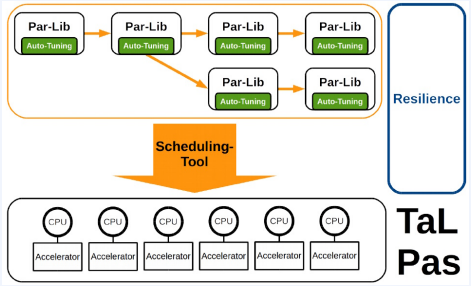
[1]S. Shudler *et al.*, in *Proc. ESPT-VPA 2017&18*, doi:10.1007/978-3-030-17872-7_8, **2019**.

Use in workflow management systems

TaLPas WMS for task-based load balancing, scheduling, and autotuning:[1]



Federal Ministry of Education and Research

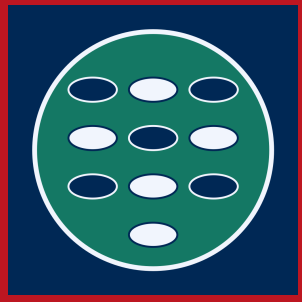


Workflow model suggests what to simulate next (p).
Performance model predicts runtime as function of N.
N is chosen by workflow manager, simulation is run.
Runtime data used to improve performance model.

[1] *J. Chem. Eng. Data* **65**(3): 1313–1329, doi:10.1021/acs.jced.9b00739, **2020**.

Noregs miljø- og biovitskaplege universitet

Institutt for datavitskap



Digitalisering på Ås

5 Parallel data

5.5 Resource efficiency metrics

5.6 Requirements modelling

5.7 Load balancing

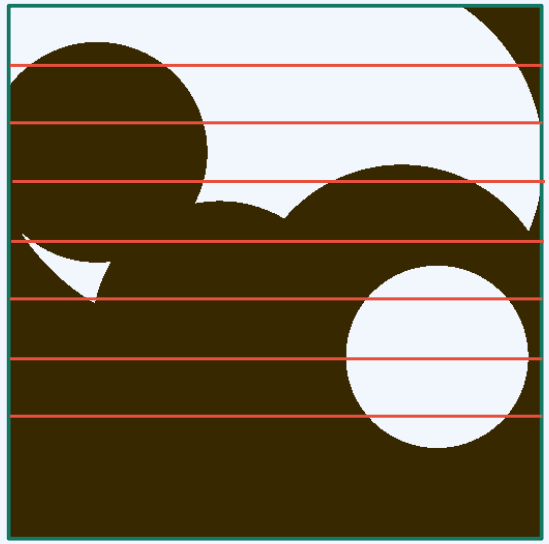
INF205

30th November 2022

The slowest process decides

The aim of load balancing is to make the slowest process as fast as possible. Ideally, all processes take the same time for their task: There is no idle time.

***a·a* pixel image**

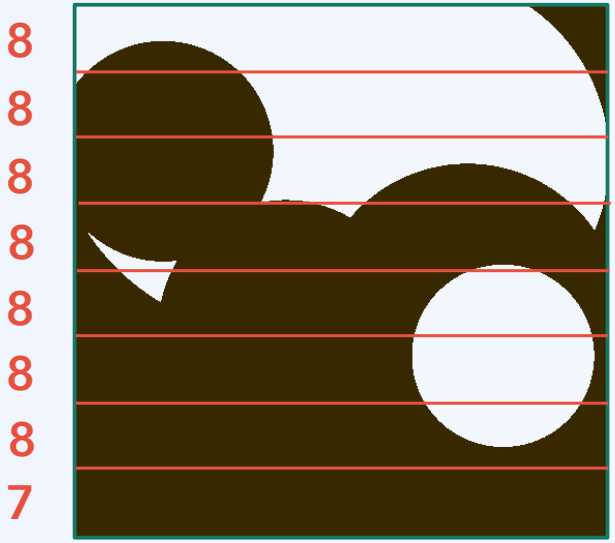


charmap-output

Ranks 0 to 6 are idle 50% of the time, all waiting for rank 7.

Viable load balance.

***a·a* pixel image**




```
int ny = a/size;
int yoffset = rank*ny;
if(rank == size-1) ny = a - yoffset;
```

```
int ny = 1 + (a-1)/size;
int yoffset = rank*ny;
if(rank == size-1) ny = a - yoffset;
```

Load-balanced domain decomposition

Norwegian University of Life Sciences

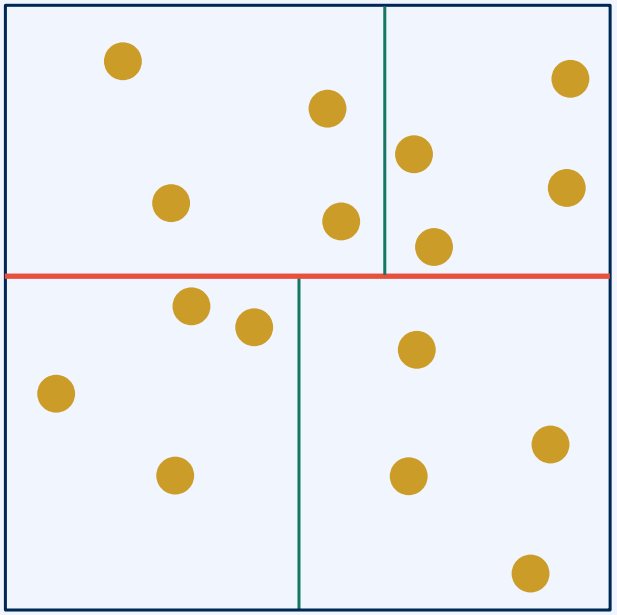
Requirements modelling can be used to predict how the way in which the domain is decomposed, for a given input case/scenario, influences the load of each of the parallel processes.

Usually, no quantitatively accurate requirements model exists. Even then, a rough approximation can be used as guidance for distributing the load.

Example scheme: **Recursive bisection** (with "*k*-dimensional tree," $k = 3$).

From the top (whole domain) down to the bottom (single process), split the volume recursively into parts such that processes will receive a similar load.

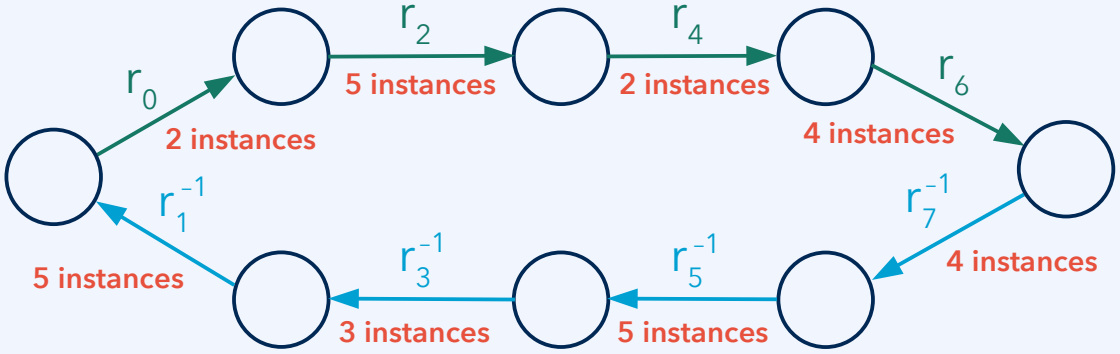
Alternate between spatial dimensions.



Load-balanced domain decomposition

Norwegian University of Life Sciences

How could this be applied to our graph querying problem?
(In absence of a model, we might approximate runtime ~ number of edges.)



$p = \{r_0, r_2, r_4, r_6\}$

$q = \{r_1, r_3, r_5, r_7\}$

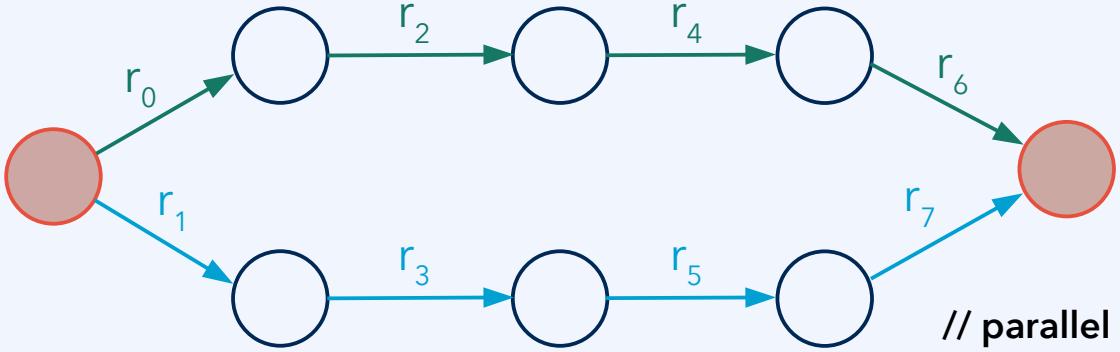
<r0> <r2> <r4> <r6> <>
<r1> <r3> <r5> <r7> <>

Can be reduced to the problem of finding a cycle, all edges have equal status.
We can split up the graph according to edge labels.

Discussion: How might the domain above be split up among four processes?

Practical findings: OpenMP sections

On average, we can expect the two paths to procude the same load. If we split the domain in two, using OpenMP sections, *parallel speedup 2.0 is expected*.



$q_1 = \{r_0, r_2, r_4, r_6\}$

$q_2 = \{r_1, r_3, r_5, r_7\}$

Observation:

Speedup is much lower than expected. In many cases, the code even becomes slower just by turning on OpenMP.

```
// parallel query search
#pragma omp parallel sections num_threads(2)
{
  #pragma omp section
  g.query(&q1, &res1);

  #pragma omp section
  g.query(&q2, &res2);
}
```

Practical findings: OpenMP sections

Challenge no. 1: Cores with shared-memory access can compete on L1 cache.

```
int main(int argc, char** argv){
  ...
  int64_t* counted_primes = new int64_t[num_threads];  // shared memory!
  omp_set_num_threads(num_threads);  // default would be to create $OMP_NUM_THREADS threads
  #pragma omp parallel {
    int thread_id = omp_get_thread_num();  // corresponds to MPI_Comm_rank in the MPI code

    counted_primes[thread_id] = 0;
    for(int64_t n = 6*(thread_id+1) - 1; n < limit; n += 6*num_threads)
      if(is_prime(n)) counted_primes[thread_id]++;
    for(int64_t n = 6*(thread_id+1) + 1; n < limit; n += 6*num_threads)
      if(is_prime(n)) counted_primes[thread_id]++;
  }
  ...
  int64_t overall_primes = 0;
  for(int i = 0; i < num_threads; i++) overall_primes += counted_primes[i];  // shared memory!
    ...
  }
```

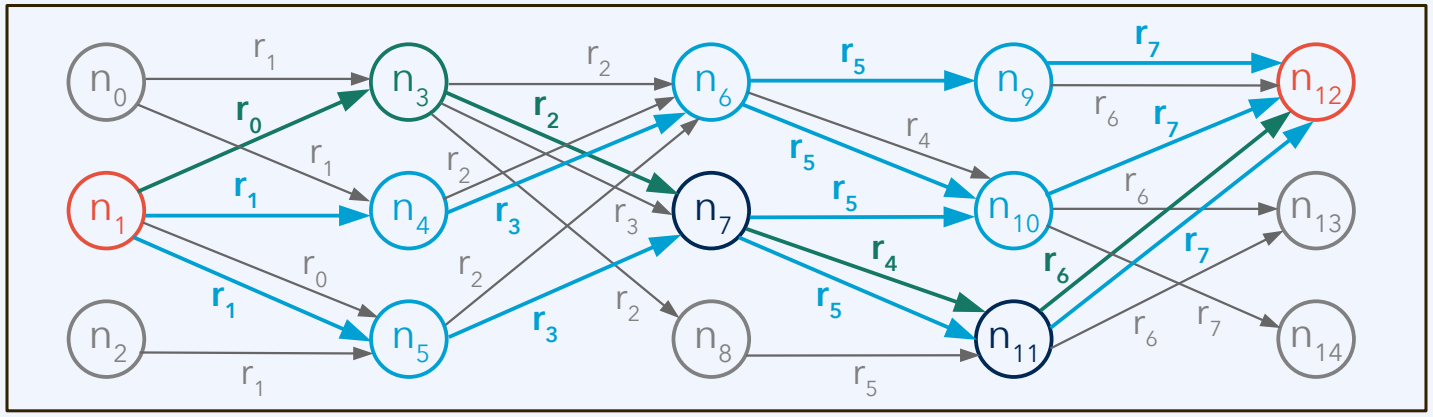
Attention: Risk of "**false sharing**" due to L1 cache line overlap.

(Compare code **omp-primes-padding**.)

In **omp-primes-padding** from week 44, we left large parts of an array empty to deal with the case where multiple cores try to load the same memory into their L1 cache.

Practical findings: OpenMP sections

Challenge no. 2: Equal load *on average* does not mean equal load *every time*.



<n0>	<r1>	<n3>	<n3>	<r2>	<n6>	**<n6>**	**<r5>**	**<n9>**	<n9>	<r6>	<n12>	<>
<n0>	<r1>	<n4>	**<n3>**	**<r2>**	**<n7>**	<n6>	<r4>	<n10>	**<n9>**	**<r7>**	**<n12>**	
<n1>	**<r0>**	**<n3>**	<n3>	<r3>	<n7>	**<n6>**	**<r5>**	**<n10>**	**<n10>**	**<r7>**	**<n12>**	
<n1>	**<r1>**	**<n4>**	<n3>	<r2>	<n8>	**<n7>**	**<r5>**	**<n10>**	<n10>	<r6>	<n13>	
<n1>	<r0>	<n5>	<n4>	<r2>	<n6>	**<n7>**	**<r4>**	**<n11>**	<n10>	<r7>	<n14>	
<n1>	**<r1>**	**<n5>**	**<n4>**	**<r3>**	**<n6>**	**<n7>**	**<r5>**	**<n11>**	**<n11>**	**<r6>**	**<n12>**	
<n2>	<r1>	<n5>	<n5>	<r2>	<n6>	<n8>	<r5>	<n11>	**<n11>**	**<r7>**	**<n12>**	
			<n5>	**<r3>**	**<n7>**				<n11>	<r6>	<n13>	

Dynamic load balancing

Observations:

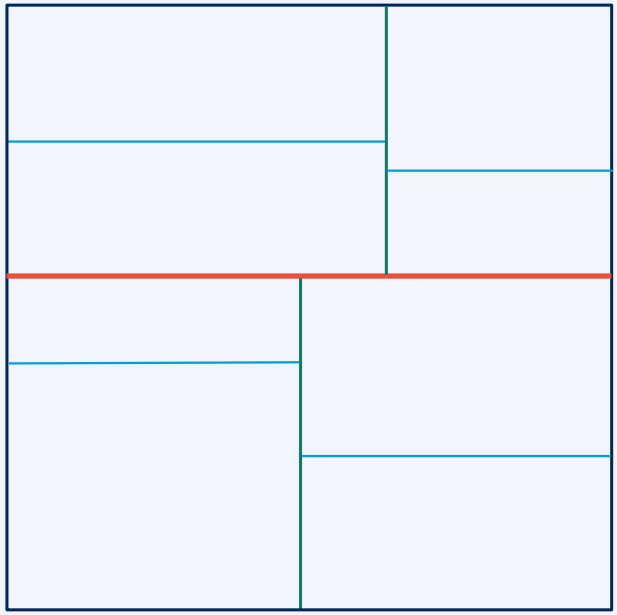
1) *Performance models* are not completely accurate. Moreover, they will usually neglect some of the parameters that influence the runtime.
2) The actual resource requirements will not always be the same for given parameter values. There can be a non-negligible *statistical uncertainty*.
3) Load can *change over runtime*, *e.g.*, from a changing density profile.
4) Anything can occur *on a node in the background, or at the hardware level* (poor cooling, needs to be clocked down, *etc.*). This cannot be reflected in the performance model, and it can change at runtime.

Execution times on HPC infrastructures are of the order of hours to days. The value of the consumed resources is substantial.

Therefore, it can be worth the effort to readjust the decomposition dynamically.

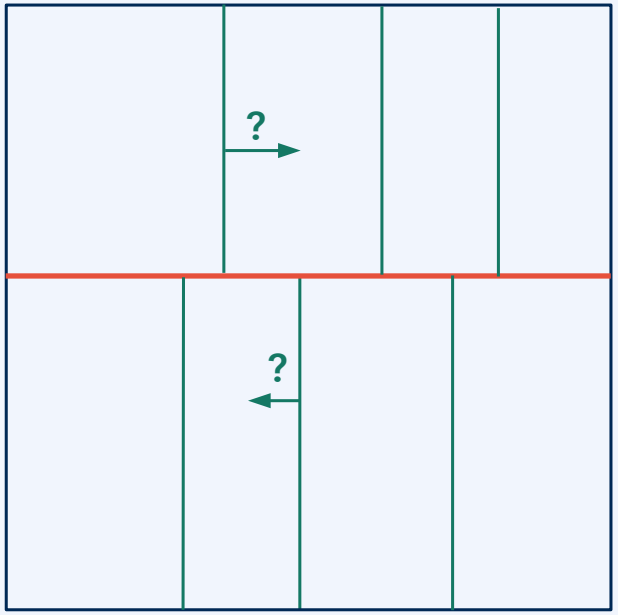
Dynamic load balancing: Example

recursive bisection



Reconstruct decomposition, *e.g.*, every 10000 steps in a molecular dynamics simulation.

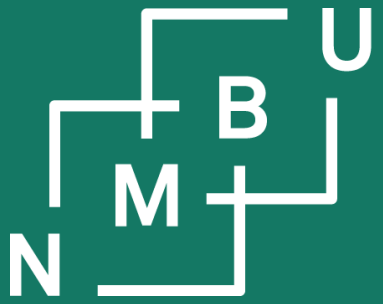
diffusive multisection[1, 2]



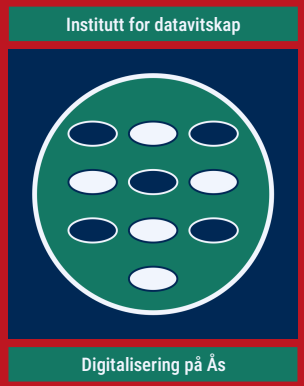
Gradual ("diffusive") changes can be implemented to adjust to configuration and performance.

[1]S. Seckler, *J. Computat. Sci.* **50**: 101296, doi:10.1016/j.jocs.2020.101296, **2021**.

[2]J. Sablić, E-CAM project deliverable 4.6, **2020**.



Noregs miljø- og biovitskaplege universitet



Conclusion

INF205

30th November 2022

Group presentations

You do **not** need to attend all presentations!
Suggestion: Come to those about your topic.

The main aims are:

- Discussion of the problems and exchange of ideas
  - Could be more useful if we can extend the deadline, *e.g.*, to 15.12.
  - Will in any case improve the way these (or similar) problems will be discussed in INF205 in the future
- Avoid misunderstandings of what you did and what your intentions were
  - Reduces mistakes during grading

1.12., 10.00–11.30
TF1-105
disk graphics, ANNs

#9, #23, #26

1.12., 12.00–14.00
TF1-105
graph querying

#8, #10, #17, #20

7.12., 14.00–16.00
TF1-102
spherical particles

#1, #6, #12, #19

9.12., 12.30–13.30
tba
various topics

#2, #3

15.12., 10.30–12.00
tba
various topics

#4, #5, #7

15.12., 12.30–14.00
tba
disk graphics

#13, #16, #25

Submission deadline

The proposal has come up (not unusual) to extend the submission deadline which is so far set to Monday, 5th December, 24.00 CET.

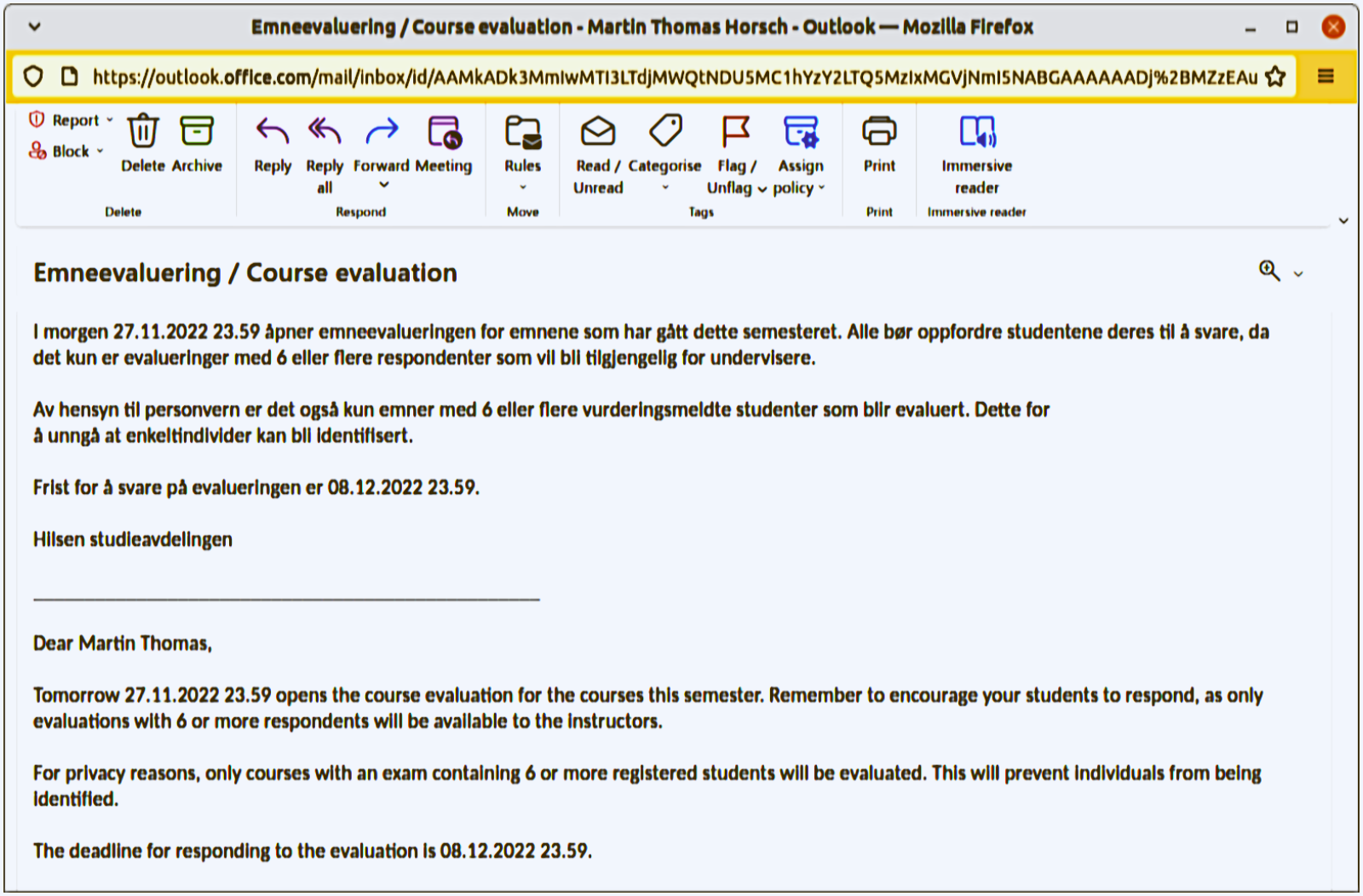
- Right now, the deadline is before the presentation for some groups, but after the presentation for others, creating slightly unlike conditions.
- The purpose of the project as an examination is to demonstrate that you have acquired competencies. *When* that is done should not matter.

On the other hand:

- With an extended deadline, people with another activity scheduled for week 49 (such as the project from INF250 “Image Analysis”) could experience themselves as being disadvantaged.
- People may have made their schedule with 5.12. as the INF205 deadline, but then they could need to contribute to their group’s work.

We can extend the deadline **if nobody objects**. Discussion may be needed.

Module evaluation



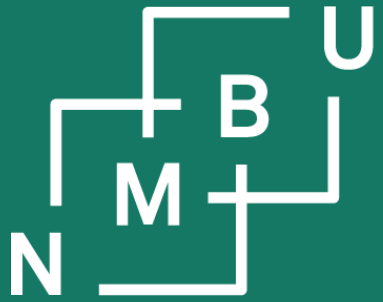
Where to go with INF205 …

See below some ideas that we have discussed internally.
They are not officially decided so far (to my knowledge).

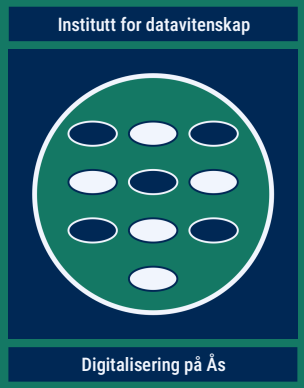
These could be worth commenting on in the module evaluation. And do think of your own ideas, small or large, how to improve INF205 for the future. Concrete, actionable suggestions are always the most useful kind of feedback.

- INF205 to be moved to the Spring term (next time is then Spring 2024).
- Lecture module INF205 to be split from programming project INF206. The project (in June) would then become optional, but require INF205.
- Consequently, INF205 would need a normal exam instead of a project.

We should also think about rearranging content, what parts of the module would benefit from a stronger focus, and what we need less (or not at all).



Norges miljø- og biovitenskapelige universitet



INF205
Resource-efficient programming

5 Parallel data

5.5 Resource efficiency metrics
5.6 Requirements modelling
5.7 Load balancing