Datalogi

An optimal and practical cache-oblivious algorithm for computing multiresolution rasters

Lars Arge, Gerth Stølting Brodal, Jakob Truelsen, Constantinos Tsirogiannis

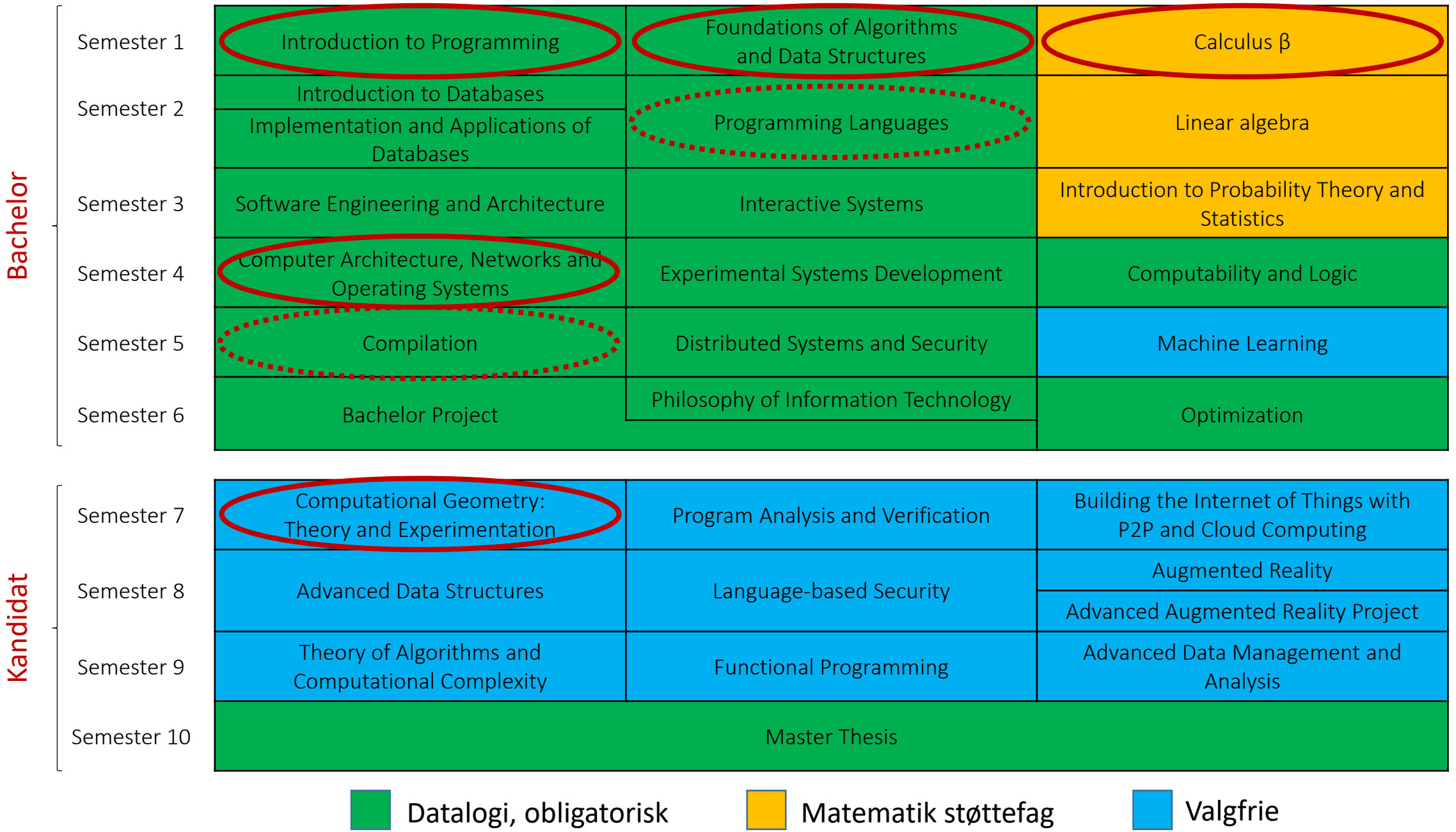
MADALGO*, Department of Computer Science, Aarhus University, Denmark
{large, gerth, jakobt, constant}@madalgo.au.dk

Abstract. In many scientific applications it is required to reconstruct a raster dataset many times, each time using a different resolution. This leads to the following problem; let $\mathcal{G}$ be a raster of $\sqrt{N} \times \sqrt{N}$ cells. We want to compute for every integer $2 \leq \mu \leq \sqrt{N}$ a raster $\mathcal{G}_\mu$ of $\lceil \sqrt{N}/\mu \rceil \times \lceil \sqrt{N}/\mu \rceil$ cells where each cell of $\mathcal{G}_\mu$ stores the average of the values of $\mu \times \mu$ cells of $\mathcal{G}$. Here we consider the case where $\mathcal{G}$ is so large that it does not fit in the main memory of the computer.

We present a novel algorithm that solves this problem in $O(\text{scan}(N))$ data block transfers from/to the external memory, and in $\Theta(N)$ CPU operations; here $\text{scan}(N)$ is the number of block transfers that are needed to read the entire dataset from the external memory. Unlike previous results on this problem, our algorithm achieves this optimal performance without making any assumptions on the size of the main memory of the computer. Moreover, this algorithm is cache-oblivious; its performance does not depend on the data block size and the main memory size.

We have implemented the new algorithm and we evaluate its performance on datasets of various sizes; we show that it clearly outperforms previous approaches on this problem. In this way, we provide solid evidence that non-trivial cache-oblivious algorithms can be implemented so that they perform efficiently in practice.

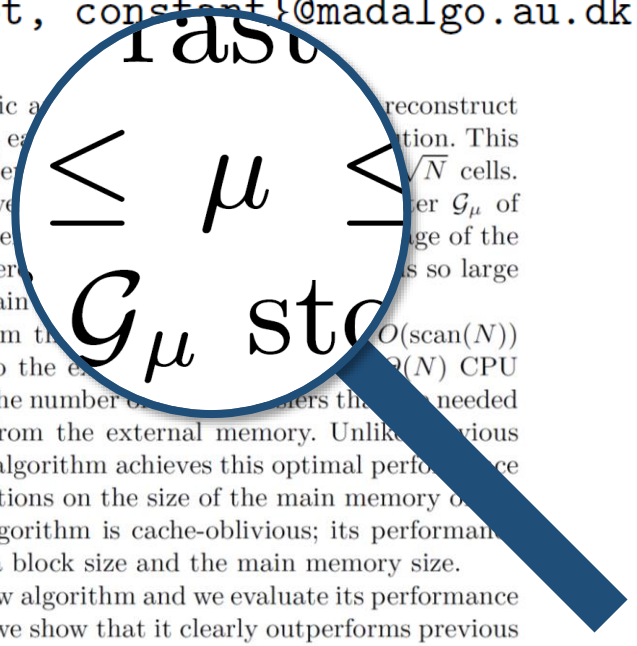
Proc. 21st Annual European Symposium on Algorithms, Sophia Antipolis, France, volume 8125 of *Lecture Notes in Computer Science*, pages 61-72, Springer Verlag, Berlin, 2013, doi 10.1007/978-3-642-40450-4_6


Bachelor
Semester 1
Introduction to Programming
Foundations of Algorithms and Data Structures
Calculus β
Semester 2
Introduction to Databases
Implementation and Applications of Databases
Programming Languages
Linear algebra
Semester 3
Software Engineering and Architecture
Interactive Systems
Introduction to Probability Theory and Statistics
Semester 4
Computer Architecture, Networks and Operating Systems
Experimental Systems Development
Computability and Logic
Semester 5
Compilation
Distributed Systems and Security
Machine Learning
Semester 6
Bachelor Project
Philosophy of Information Technology
Optimization
Kandidat
Semester 7
Computational Geometry: Theory and Experimentation
Program Analysis and Verification
Building the Internet of Things with P2P and Cloud Computing
Semester 8
Advanced Data Structures
Language-based Security
Augmented Reality
Advanced Augmented Reality Project
Semester 9
Theory of Algorithms and Computational Complexity
Functional Programming
Advanced Data Management and Analysis
Semester 10
Master Thesis
Datalogi, obligatorisk
Matematik støttefag
Valgfrie

An optimal and practical cache-oblivious algorithm for computing multiresolution rasters

Lars Arge, Gerth Stølting Brodal, Jakob Truelsen, Constantinos Tsirogiannis

MADALGO*, Department of Computer Science, Aarhus University, Denmark
{large, gerth, jakobt, constant}@madalgo.au.dk



Abstract. In many scientific a reconstruct a raster dataset many times, e tion. This leads to the following proble $\sqrt{N}$ cells. We want to compute for eve er $\mathcal{G}_\mu$ of $\lceil\sqrt{N}/\mu\rceil \times \lceil\sqrt{N}/\mu\rceil$ cells whe ge of the values of $\mu \times \mu$ cells of $\mathcal{G}$. Her s so large that it does not fit in the main We present a novel algorithm t $O(\text{scan}(N))$ data block transfers from/to the e $O(N)$ CPU operations; here $\text{scan}(N)$ is the number ers th needed to read the entire dataset from the external memory. Unlik vious results on this problem, our algorithm achieves this optimal perfo ce without making any assumptions on the size of the main memory o computer. Moreover, this algorithm is cache-oblivious; its performan does not depend on the data block size and the main memory size.
We have implemented the new algorithm and we evaluate its performance on datasets of various sizes; we show that it clearly outperforms previous approaches on this problem. In this way, we provide solid evidence that non-trivial cache-oblivious algorithms can be implemented so that they perform efficiently in practice.

Proc. 21st Annual European Symposium on Algorithms, Sophia Antipolis, France, volume 8125 of *Lecture Notes in Computer Science*, pages 61-72, Springer Verlag, Berlin, 2013, doi 10.1007/978-3-642-40450-4_6

An optimal and practical cache-oblivious algorithm for computing multiresolution rasters

Lars Arge, Gerth Stølting Brodal, Jakob Truelsen, Constantinos Tsirogiannis

MADALGO*, Department of Computer Science, Aarhus University, Denmark
{large, gerth, jakobt, constant}@madalgo.au.dk

I er her

Aarhus
2004-07 B.Sc.
2007-10 M.Sc.
2008-15 Ph.d.

SCALGO

2001-07 B.Sc. Athen
2006-07 M.Sc. Athen
2007-11 Ph.d. Eindhoven
2011-17 Postdoc Aarhus
(2013-14 Græsk militær)

teklatech

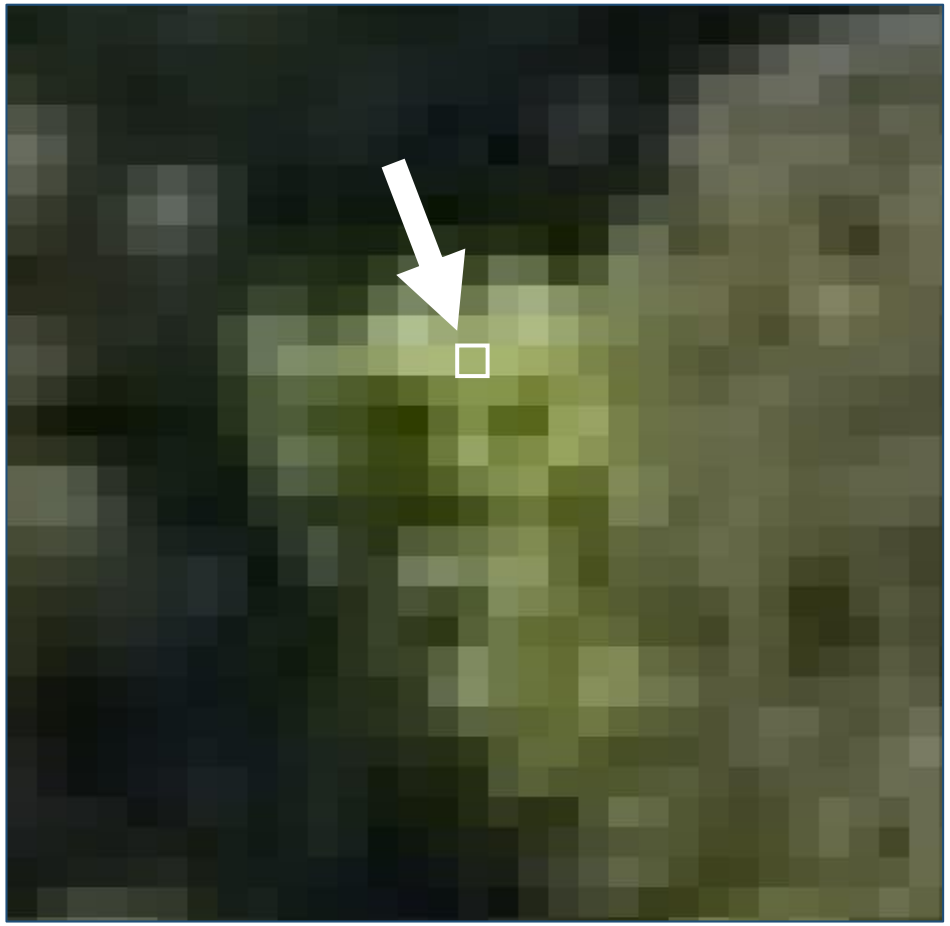
Aarhus
1989-94 M.Sc.
1993-97 Ph.d.
1997-98 Postdoc (Saarbrücken)
1998-04 Postdoc
2004-15 Lektor
2016- Professor

Aarhus
1986-90 B.Sc.
1990-93 M.Sc.
1992-96 Ph.d.
1996-98 Postdoc (Duke)
1998-03 Adjunkt (Duke)
2004- Professor
2007-17 Centerleder

MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

Samarbejde med Økoinformatik og Biodiversitet gruppen ved Bioscience

An optimal and practical cache-oblivious algorithm for computing multiresolution rasters

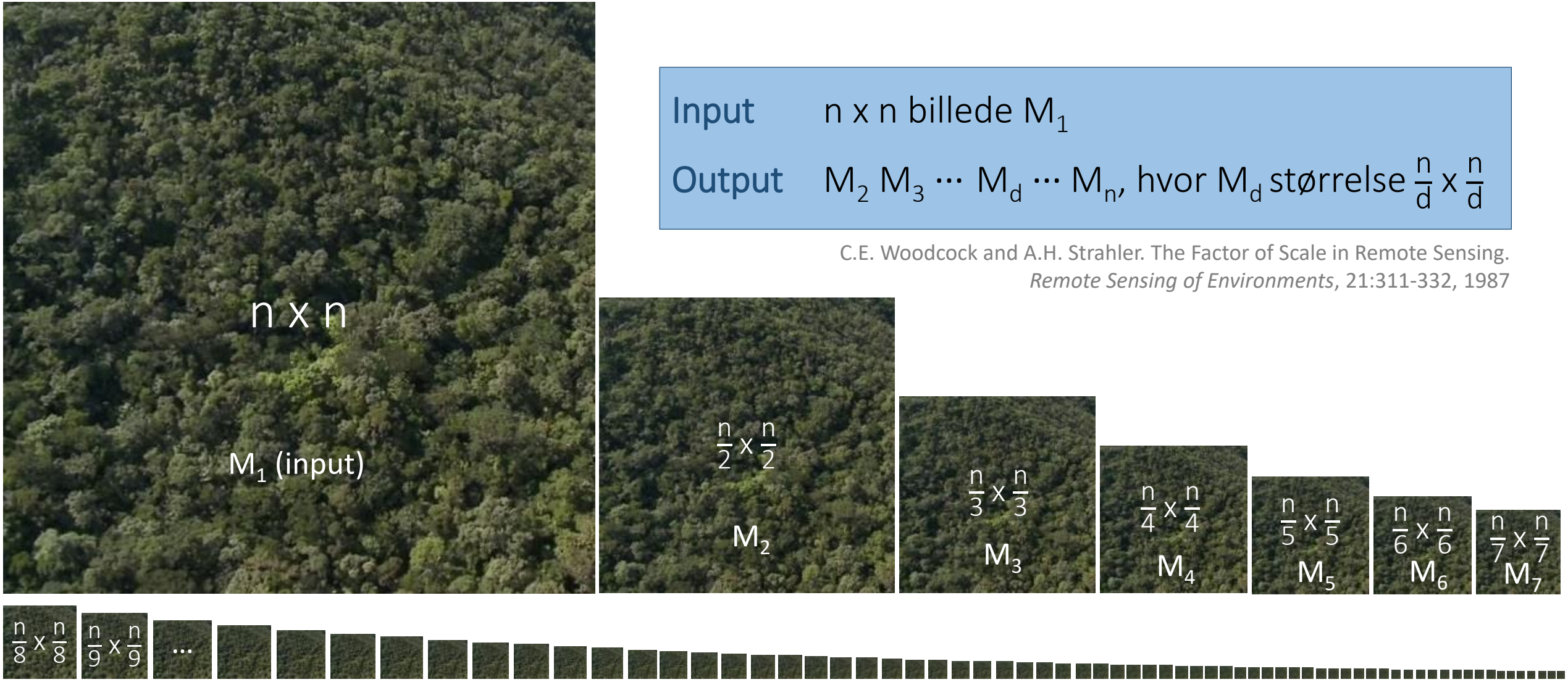


Rød: 64%
= Grøn: 70%
Blå: 44%

An optimal and practical cache-oblivious algorithm for computing multiresolution rasters

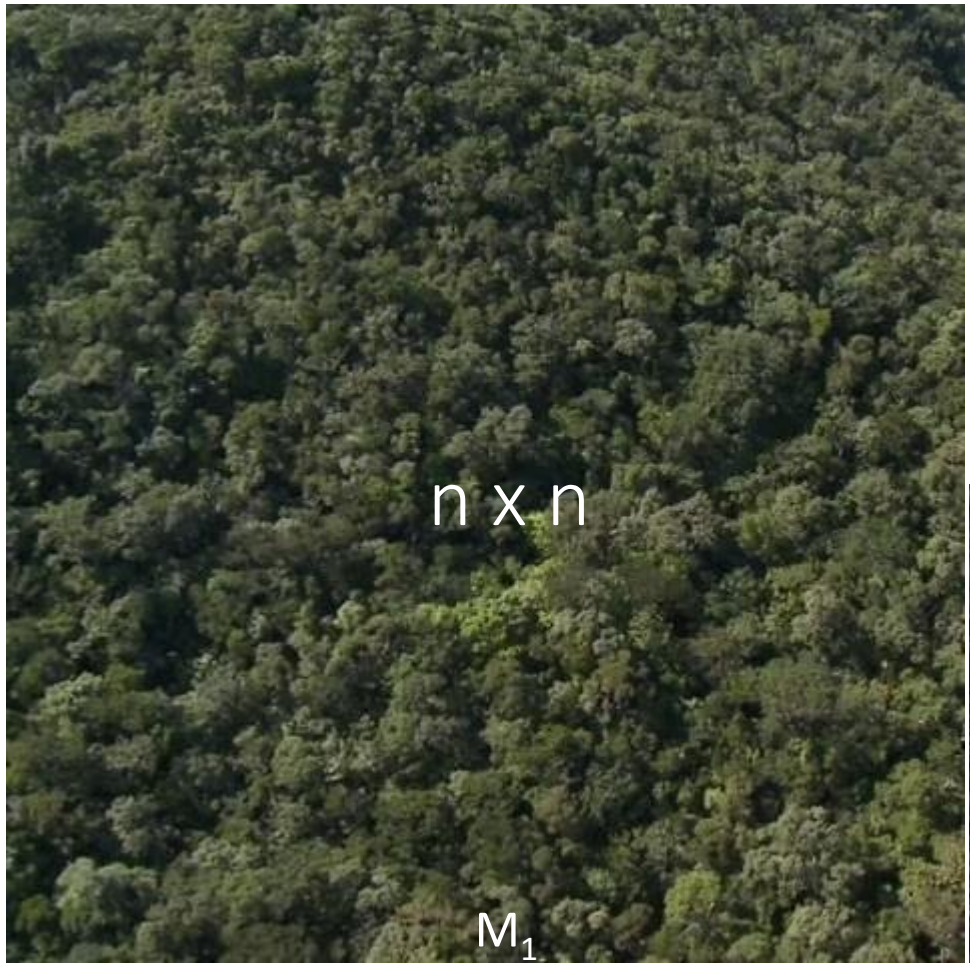
Input	n x n billede M_1
Output	M_2 M_3 ⋯ M_d ⋯ M_n, hvor M_d størrelse $\frac{n}{d}$ x $\frac{n}{d}$

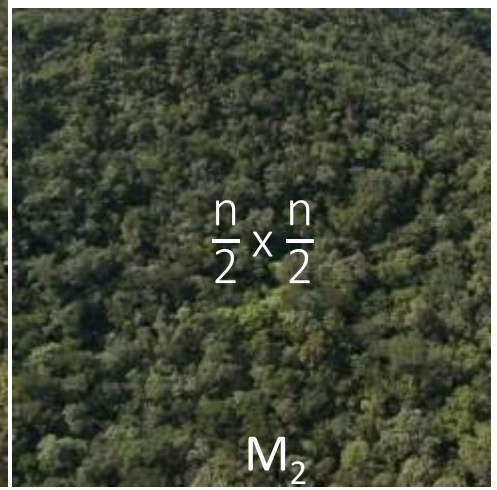
C.E. Woodcock and A.H. Strahler. The Factor of Scale in Remote Sensing. *Remote Sensing of Environments*, 21:311-332, 1987

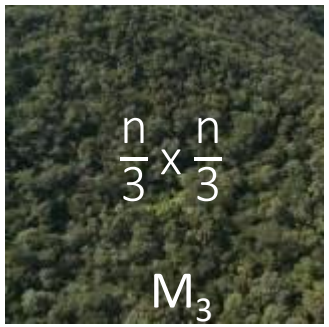


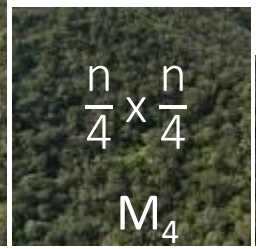
Samlet bredde $= \frac{n}{1} + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{d} + \cdots + \frac{n}{n-1} + \frac{n}{n} =$

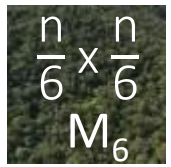
$$\sum_{d=1}^{n} \frac{n}{d} = n \sum_{d=1}^{n} \frac{1}{d} \leq n\left(1 + \int_{1}^{n} \frac{1}{x}\,dx\right) = n\left(1 + [\ln(x)]_{1}^{n}\right) = n(1+\ln(n))$$

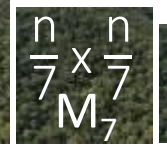


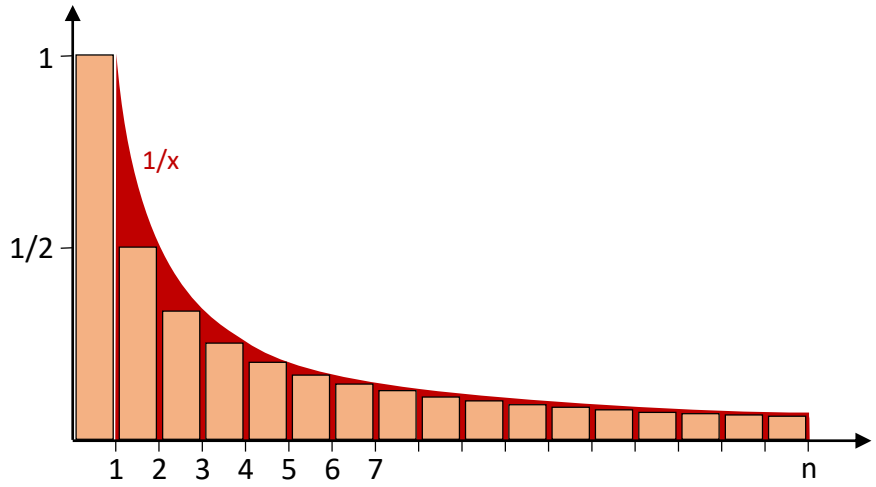






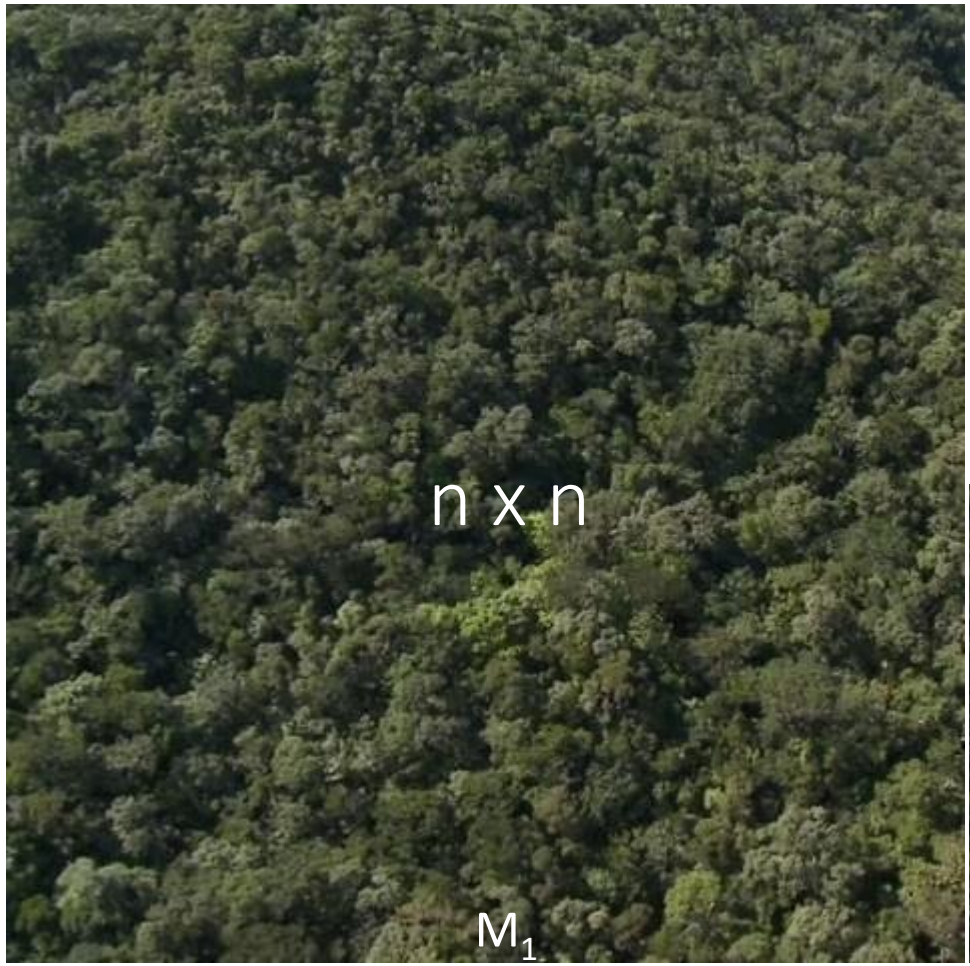


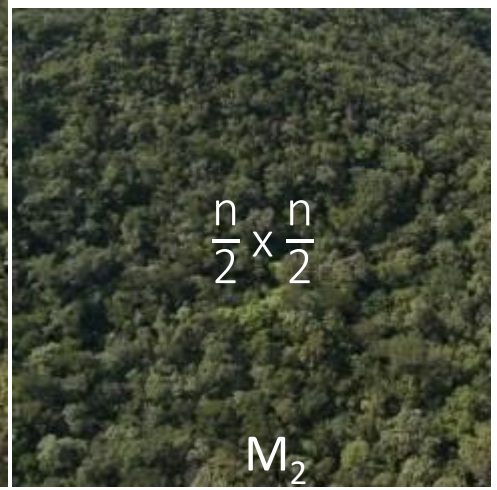


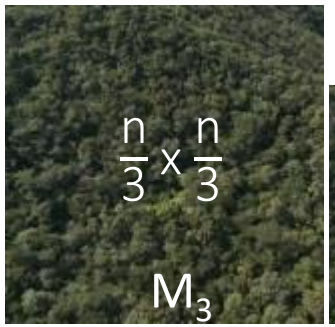


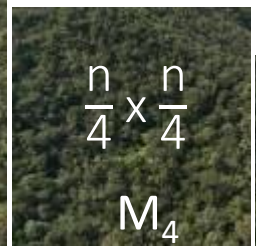
$$\text{Samlet størrelse} = \left(\frac{n}{1}\right)^2 + \left(\frac{n}{2}\right)^2 + \left(\frac{n}{3}\right)^2 + \cdots + \left(\frac{n}{d}\right)^2 + \cdots + \left(\frac{n}{n-1}\right)^2 + \left(\frac{n}{n}\right)^2$$

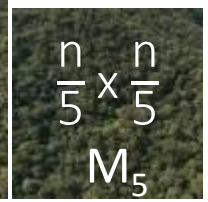
$$= \sum_{d=1}^{n} \left(\frac{n}{d}\right)^2 = n^2 \sum_{d=1}^{n} \frac{1}{d^2} \leq n^2 \left(1 + \int_1^n \frac{1}{x^2}\,dx\right) = n^2 \left(1 + \left[-\frac{1}{x}\right]_1^n\right) \leq 2n^2$$

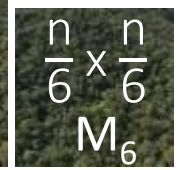


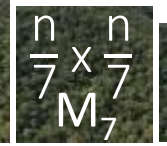


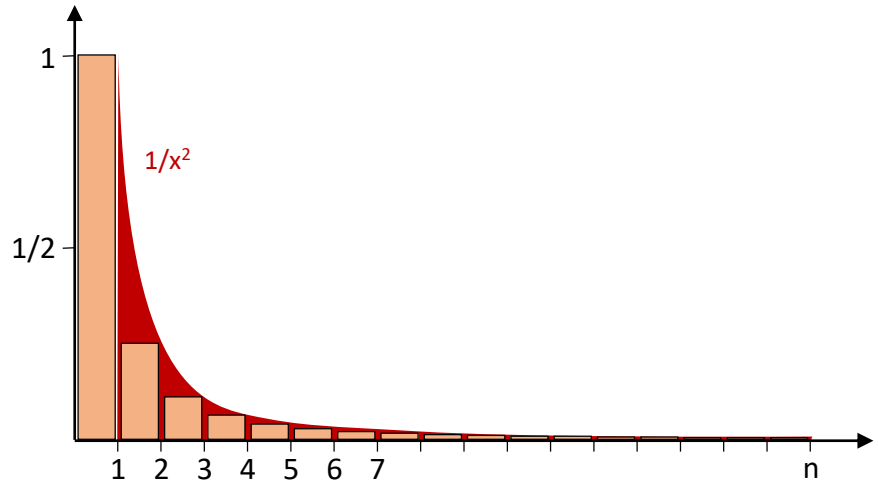




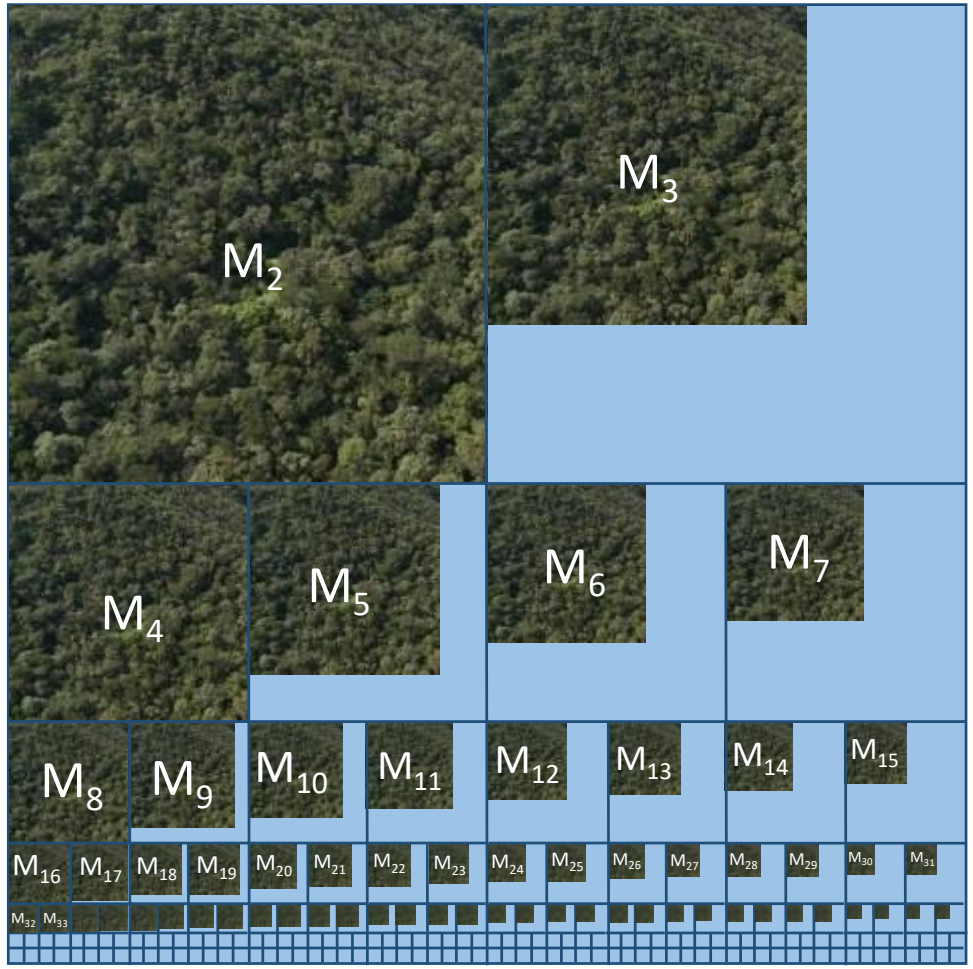






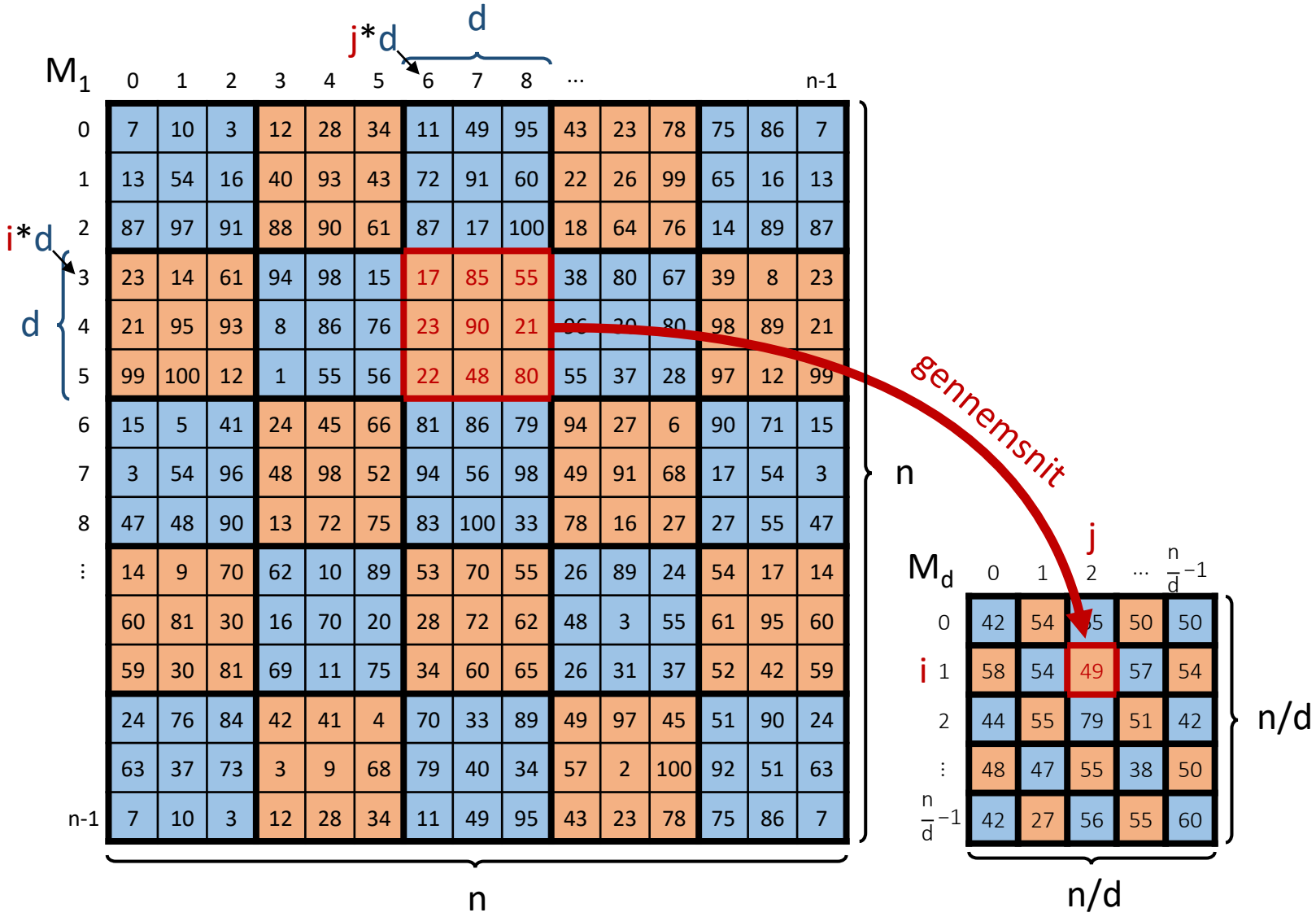


Input n x n



Output n x n

An optimal and practical cache-oblivious algorithm for computing multiresolution rasters



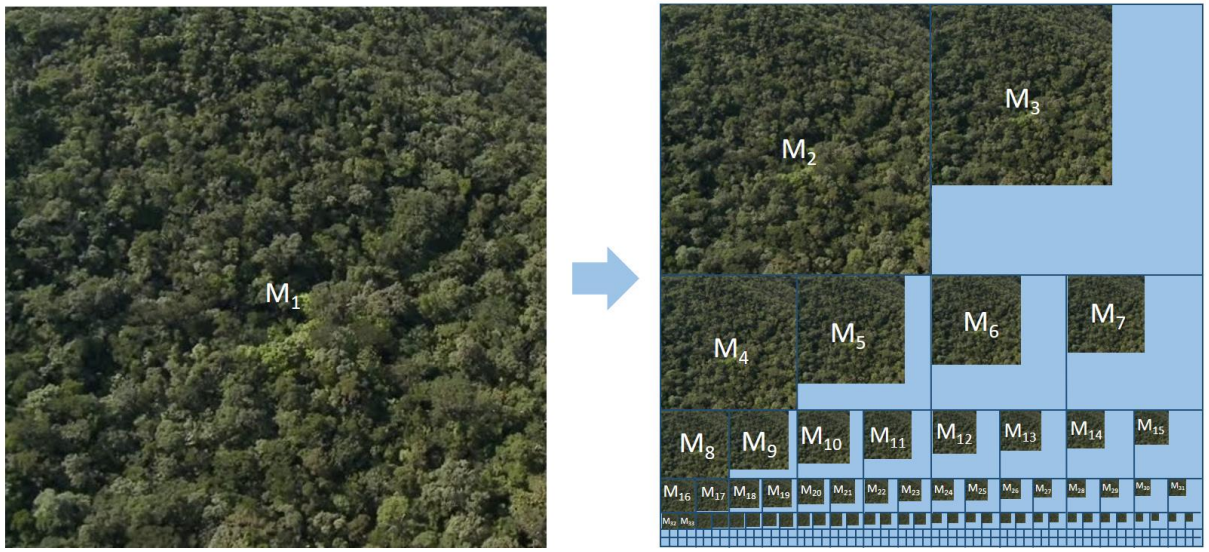
Algoritme 1

```
for d = 2 to n
   for i = 0 to ⌊n/d⌋-1
      for j = 0 to ⌊n/d⌋-1
         sum = 0
         for r = 0 to d-1
            for k = 0 to d-1
               sum = sum + M1[i*d+r, j*d+k]
         Md[i, j] = ⌊sum/d²⌋
```

M_1 tilgås ≈ $\mathbf{n^3}$ gange

$$\left\lfloor \frac{17+85+55+23+90+21+22+48+80}{9} \right\rfloor = 49$$

An optimal and practical cache-oblivious algorithm for computing multiresolution rasters



Input n x n

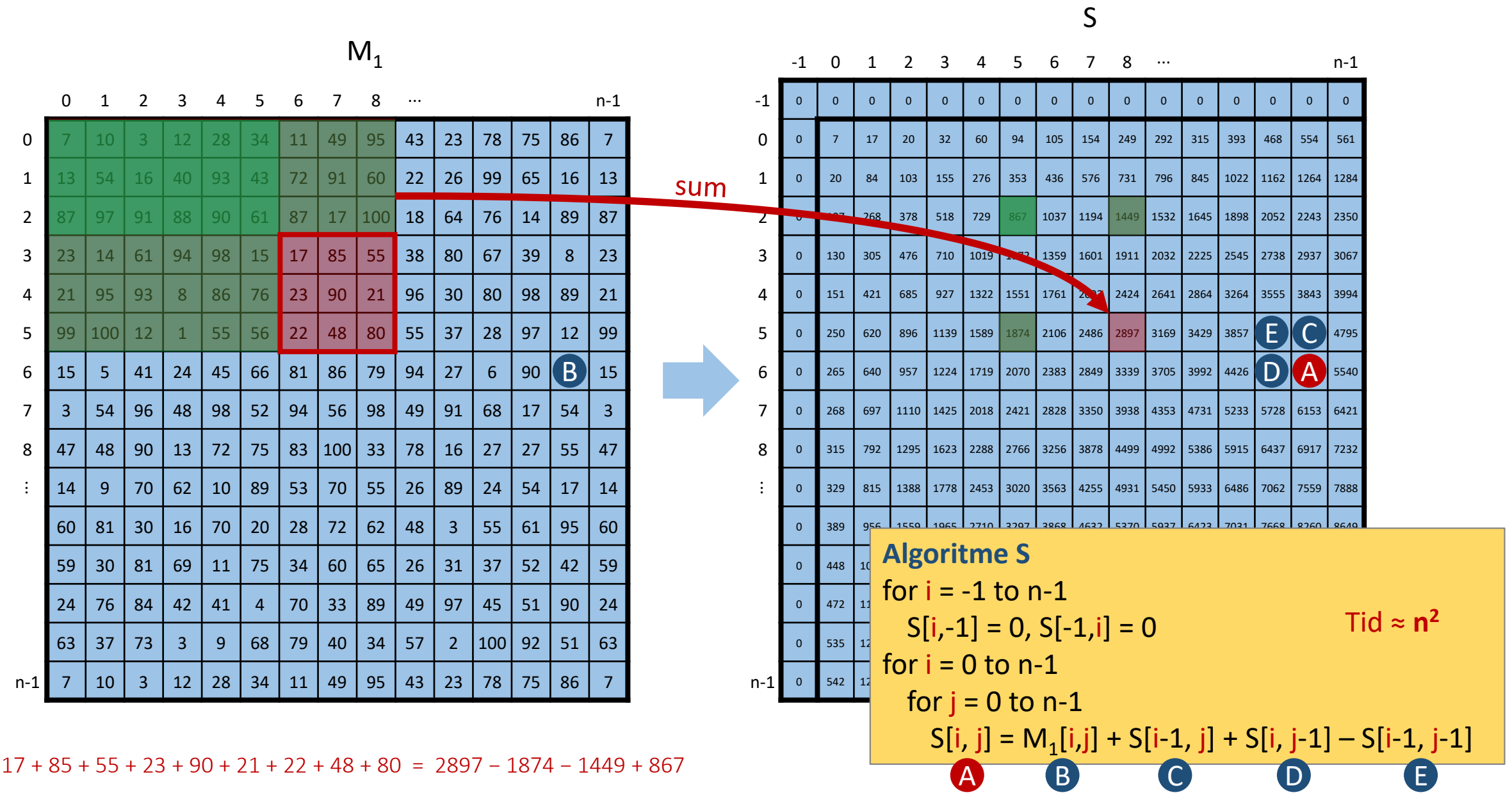
Output n x n

Algoritme 1

```
for d = 2 to n
   for i = 0 to ⌊n/d⌋-1                    Tid ≈ n³
      for j = 0 to ⌊n/d⌋-1
         sum = 0
         for r = 0 to d-1
            for k = 0 to d-1
               sum = sum + M1[i*d+r, j*d+k]
         Md[i, j] = ⌊sum/d²⌋
```

Input og output størrelse n^2, men Algoritme 1 tager tid n^3.

Kan man løse problemet hurtigere ?


M1
S
sum
Algoritme S
for i = -1 to n-1
S[i,-1] = 0, S[-1,i] = 0
for i = 0 to n-1
for j = 0 to n-1
S[i, j] = M1[i,j] + S[i-1, j] + S[i, j-1] – S[i-1, j-1]
Tid ≈ n²
A
B
C
D
E
17 + 85 + 55 + 23 + 90 + 21 + 22 + 48 + 80 = 2897 – 1874 – 1449 + 867

An optimal and practical cache-oblivious algorithm for computing multiresolution rasters

S

r \ k	-1	0	1	2	3	4	5	6	7	8	⋯	n-1
-1	0	0	0	0	0	0	0	0	0	0	⋯	0
0	0	7	17	20	32	60	94	105	154	249	⋯	561
1	0	20	84	103	155	276	353	436	576	731	⋯	1284
2	0	107	268	378	518	729	867	1037	1194	1449	⋯	2350
3	0	130	305	476	710	1019	1172	1359	1601	1911	⋯	3067
4	0	151	421	685	927	1322	1551	1761	2093	2424	⋯	3994
5	0	250	620	896	1139	1589	1874	2106	2486	2897	⋯	4795
6	0	265	640	957	1224	1719	2070	2383	2849	3339	⋯	5540
7	0	268	697	1110	1425	2018	2421	2828	3350	3938	⋯	6421
8	0	315	792	1295	1623	2288	2766	3256	3878	4499	⋯	7232
⋮	0	329	815	1388	1778	2453	3020	3563	4255	4931	⋯	7888
	0	389	956	1559	1965	2710	3297	3868	4632	5370	⋯	8649
	0	448	1045	1729	2204	2960	3622	4227	5051	5854	⋯	9380
	0	472	1145	1913	2430	3227	3893	4568	5425	6317	⋯	10199
	0	535	1245	2086	2606	3412	4146	4900	5797	6723	⋯	10970
n-1	0	542	1262	2106	2638	3472	4240	5005	5951	6972	⋯	11531

Algoritme 2

for d = 2 to n

  for i = 0 to ⌊n/d⌋-1

    for j = 0 to ⌊n/d⌋-1

      r = i * d + d + 1, k = j * d + d + 1

      $M_d[i,j] = \lfloor (S[r,k] - S[r-d,k] - S[r,k-d] + S[r-d,k-d])/d^2 \rfloor$

Tid ≈ $\mathbf{n^2}$

S tilgås ≤ $4\mathbf{n^2}$ gange

M_d

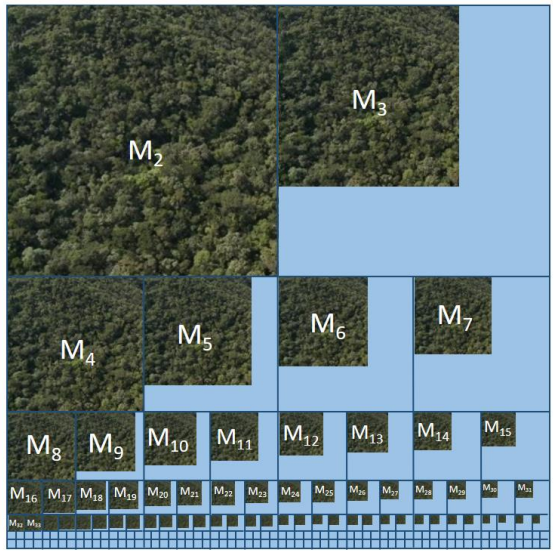
i \ j	0	1	2	⋯	$\frac{n}{d}-1$
0	42	54	65	50	50
1	58	54	49	57	54
2	44	55	79	51	42
⋮	48	47	55	38	50
$\frac{n}{d}-1$	42	27	56	55	60

n/d

n/d

An optimal and practical cache-oblivious algorithm for computing multiresolution rasters

Input n x n



Output n x n

Algoritme 2

for d = 2 to n
  for i = 0 to ⌊n/d⌋-1
    for j = 0 to ⌊n/d⌋-1
      r = i * d + d + 1, k = j * d + d + 1
      $M_d[i,j] = \lfloor (S[r,k] - S[r-d,k] - S[r,k-d] + S[r-d,k-d])/d^2 \rfloor$

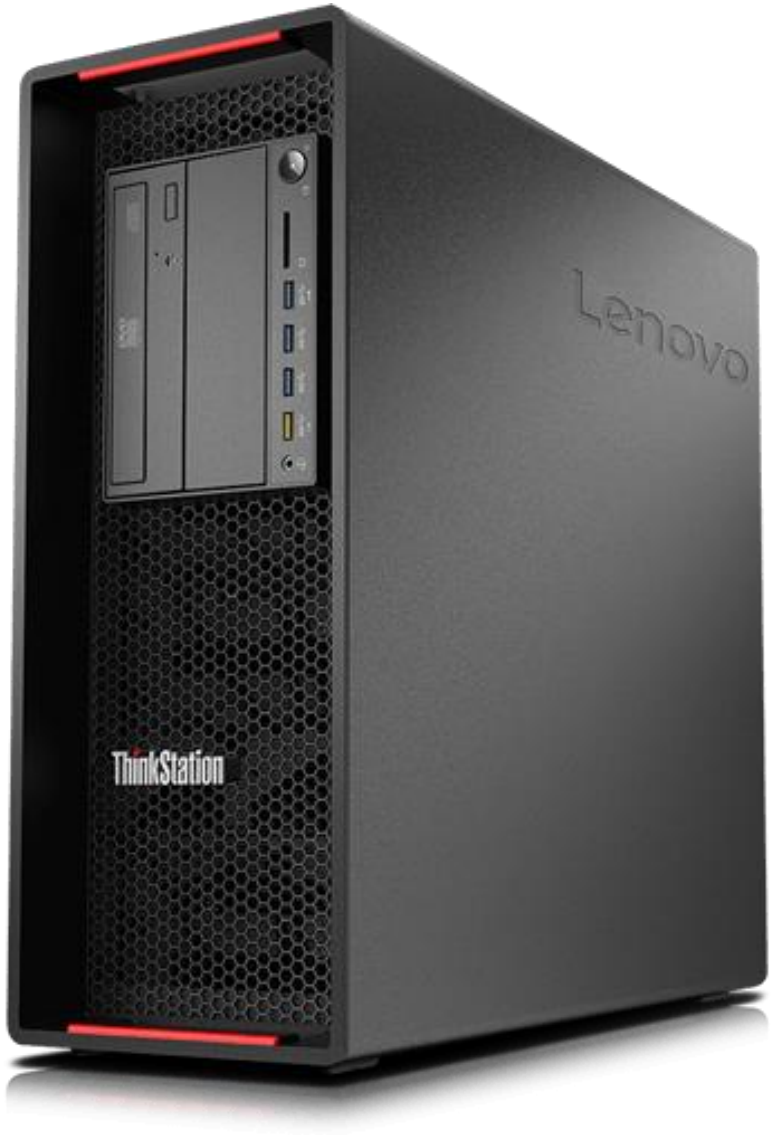
Tid ≈ **n^2**

Algoritme 2 tager optimal tid ≈ n^2

An optimal and practical cache-oblivious algorithm for computing multiresolution rasters

Algoritme 2 bruger den **fysiske hukommelse** uhensigtsmæssigt

Lenovo ThinkStation P710



lenovo.com

2 x CPU

2 x Intel Xeon
E5-2643 v4-processor

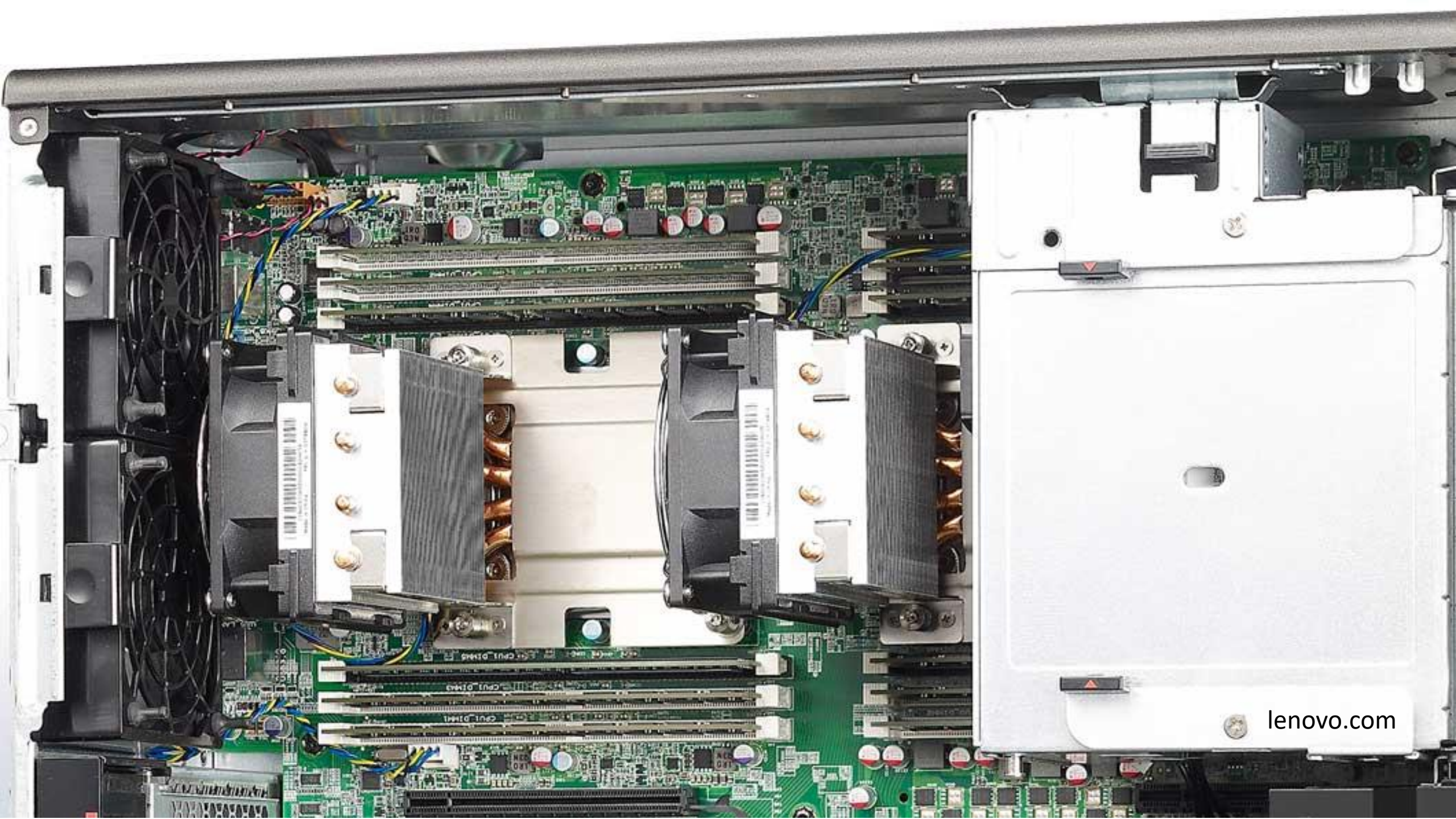
12 x hukommelse

12 x 32 GigaBytes
= 384 x 10^9 bytes

4 x harddiske

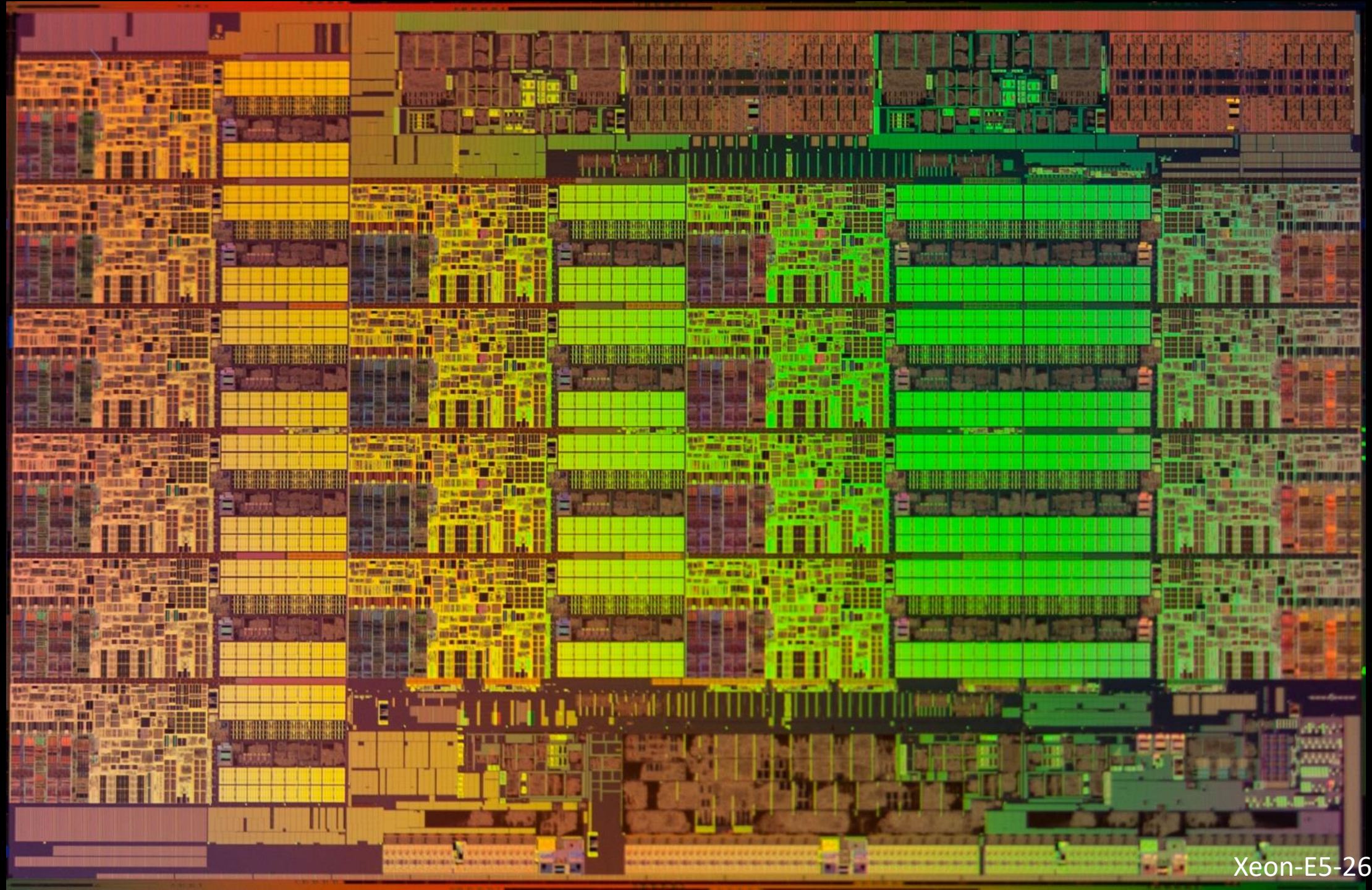
4 x 4 TeraBytes
= 16 x 10^{12} bytes

lenovo.com

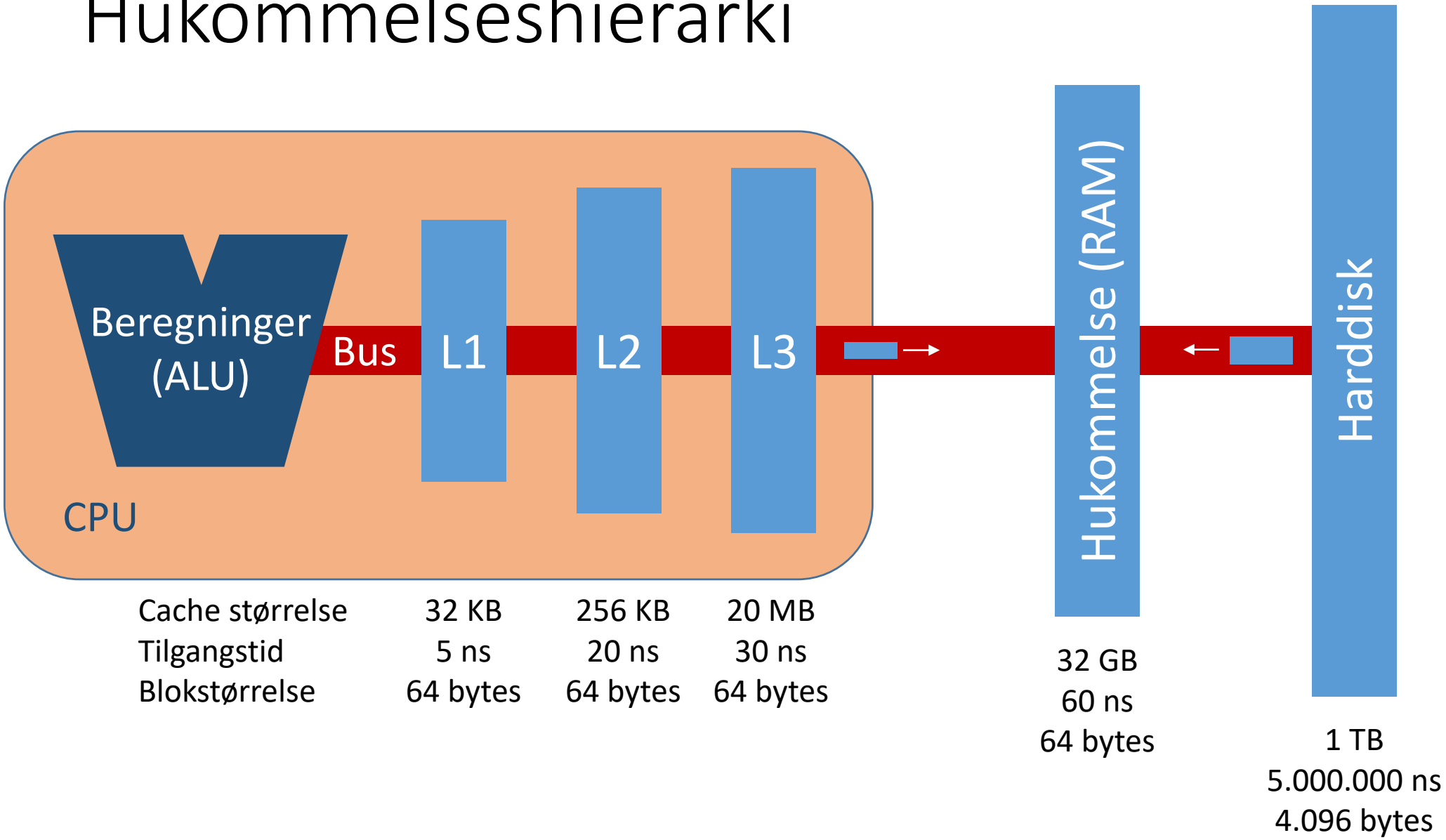

lenovo.com

Intel Xeon E5-2643 v4-processor

Transistors	3,200,000,000
The number of CPU cores	6
The number of threads	12
Level 1 instruction cache	6 x 32 KB 8-way set associative
Level 1 data cache	6 x 32 KB 8-way set associative
Level 2 cache	6 x 256 KB 8-way set associative
Level 3 cache	20 MB 20-way set associative shared
Cache line size	64 bytes
Data width	64 bit


Xeon-E5-2600-V3

Hukommelseshierarki



Algoritme 2 og Cache

Cache blokstørrelse = 4

S	-1	0	1	2	3	4	5	6	7	8	⋯					n-1
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	7	17	20	32	60	94	105	154	249	292	315	393	468	554	561
1	0	20	84	103	155	276	353	436	576	731	796	845	1022	1162	1264	1284
2	0	107	268	378	518	729	867	1037	1194	1449	1532	1645	1898	2052	2243	2350
3	0	130	305	476	710	1019	1172	1359	1601	1911	2032	2225	2545	2738	2937	3067
4	0	151	421	685	927	1322	1551	1761	2093	2424	2641	2864	3264	3555	3843	3994
5	0	250	620	896	1139	1589	1874	2106	2486	2897	3169	3429	3857	4245	4545	4795
6	0	265	640	957	1224	1719	2070	2383	2849	3339	3705	3992	4426	4904	5275	5540
7	0	268	697	1110	1425	2018	2421	2828	3350	3938	4353	4731	5233	5728	6153	6421
8	0	315	792	1295	1623	2288	2766	3256	3878	4499	4992	5386	5915	6437	6917	d
⋮	0	329	815	1388	1778	2453	3020	3563	4255	4931	5450	5933	6486	7062	7559	7888
	0	389	956	1559	1965	2710	3297	3868	4632	5370	5937	6423	7031	7668	8260	8649
	0	448	1045	1729	2204	2960	3622	4227	5051	5854	6447	6964	7609	8298	8932	9380
	0	472	1145	1913	2430	3227	3893	4568	5425	6317	6959	7573	8263	9003	9727	10199
	0	535	1245	2086	2606	3412	4146	4900	5797	6723	7422	8038	8828	9660	10435	10970
n-1	0	542	1262	2106	2638	3472	4240	5005	5951	6972	7714	8353	9221	10128	10989	11531

Algoritme 2

```
for d = 2 to n
   for i = 0 to ⌊n/d⌋-1
      for j = 0 to ⌊n/d⌋-1
         r = i * d, k = j * d
         M_d[i,j] = ⌊(S[r,k] – S[r–d,k] – S[r,k–d] + S[r–d,k–d])/d²⌋
```

Tid ≈ **n²**

Hver tilgang til S kan være i en ny blok ☹

M_6

	0	1
0	52	55
1	49	56

An optimal and practical cache-oblivious algorithm for computing multiresolution rasters

Forskningsområde

- algoritmedesign
- udnytter at data kommer i blokke
- hardware parametrene må *ikke* indgå i algoritmen

Matteo Frigo, Charles E. Leiserson, Harald Prokop, Sridhar Ramachandran. Cache-Oblivious Algorithms. *ACM Transactions on Algorithms*, 8(1), Article No. 4, 2012.

Primtal

Definition Et heltal p er et primtal hvis og kun hvis 1 og p går op i p

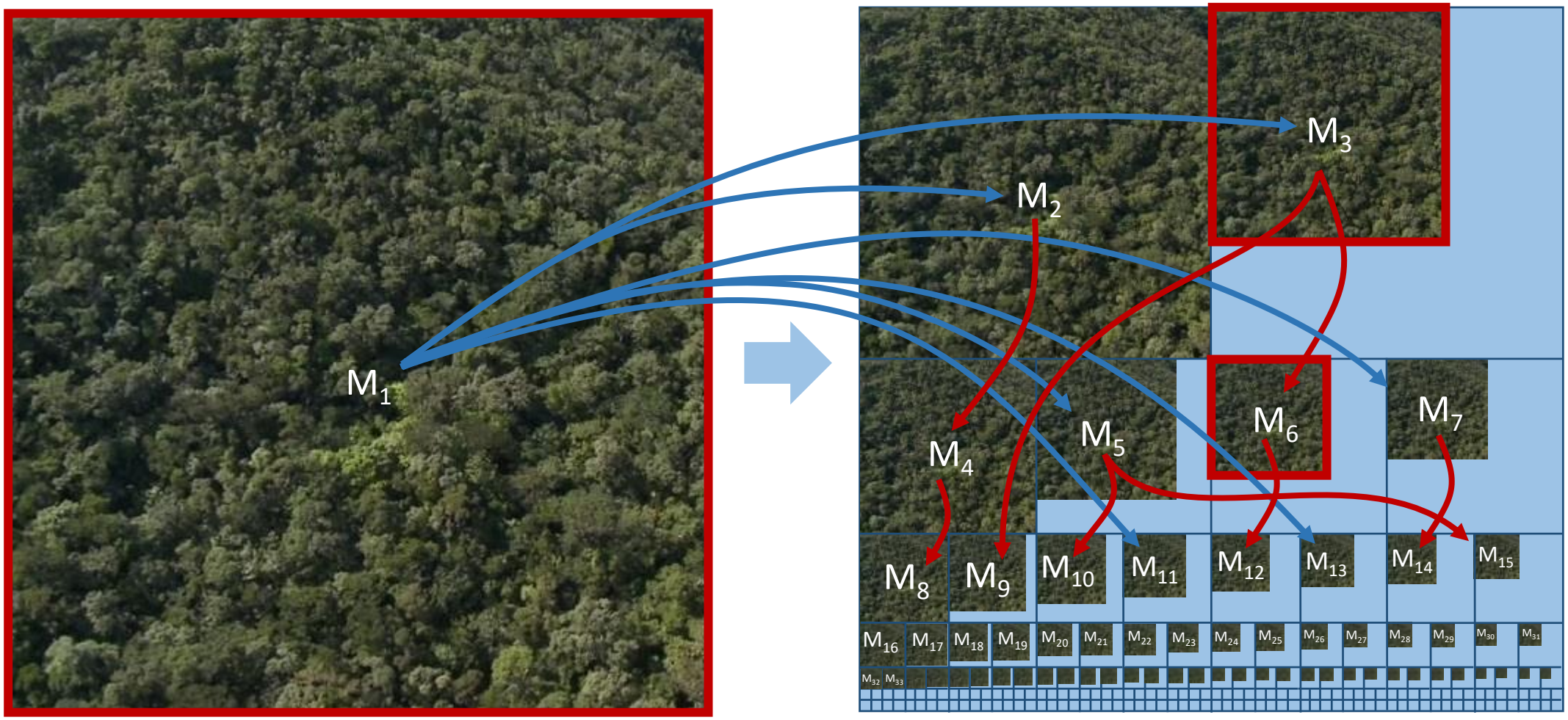
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137,
139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, …

Primtalssætningen Der er $< \frac{1.26\ n}{\ln n}$ primtal mellem 2 og n

Primtalsfaktorisering

Definition Opskrivning af et heltal x som produkt af primtal

$$75460 = 2 * 2 * 5 * 7 * 7 * 7 * 11 = 2^2 * 5 * 7^3 * 11$$



Input n x n

Output n x n

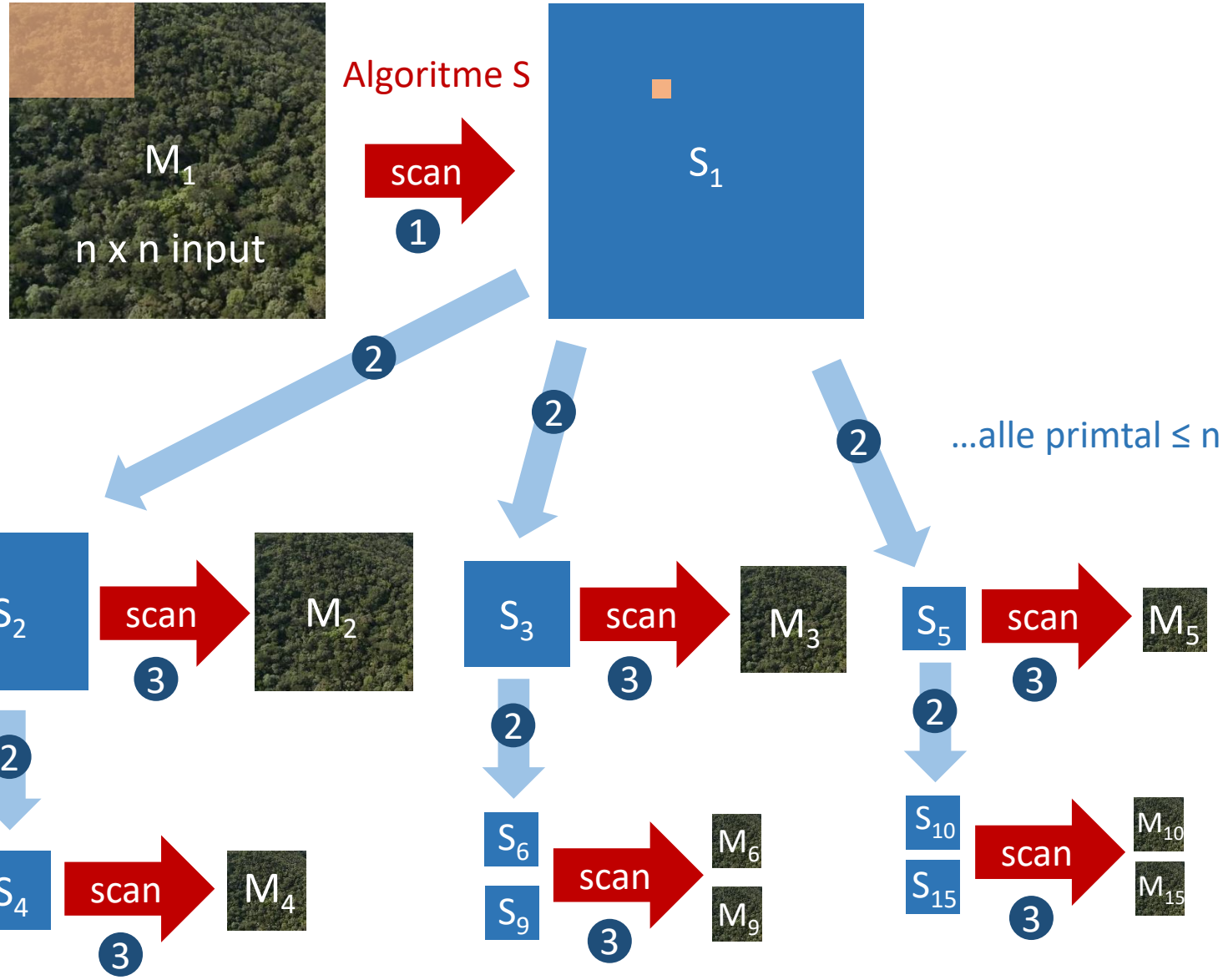
Ide Konstruer M_d ud fra $M_{d/p}$, hvor p mindste primfaktor i d

Cache Oblivious Algoritme

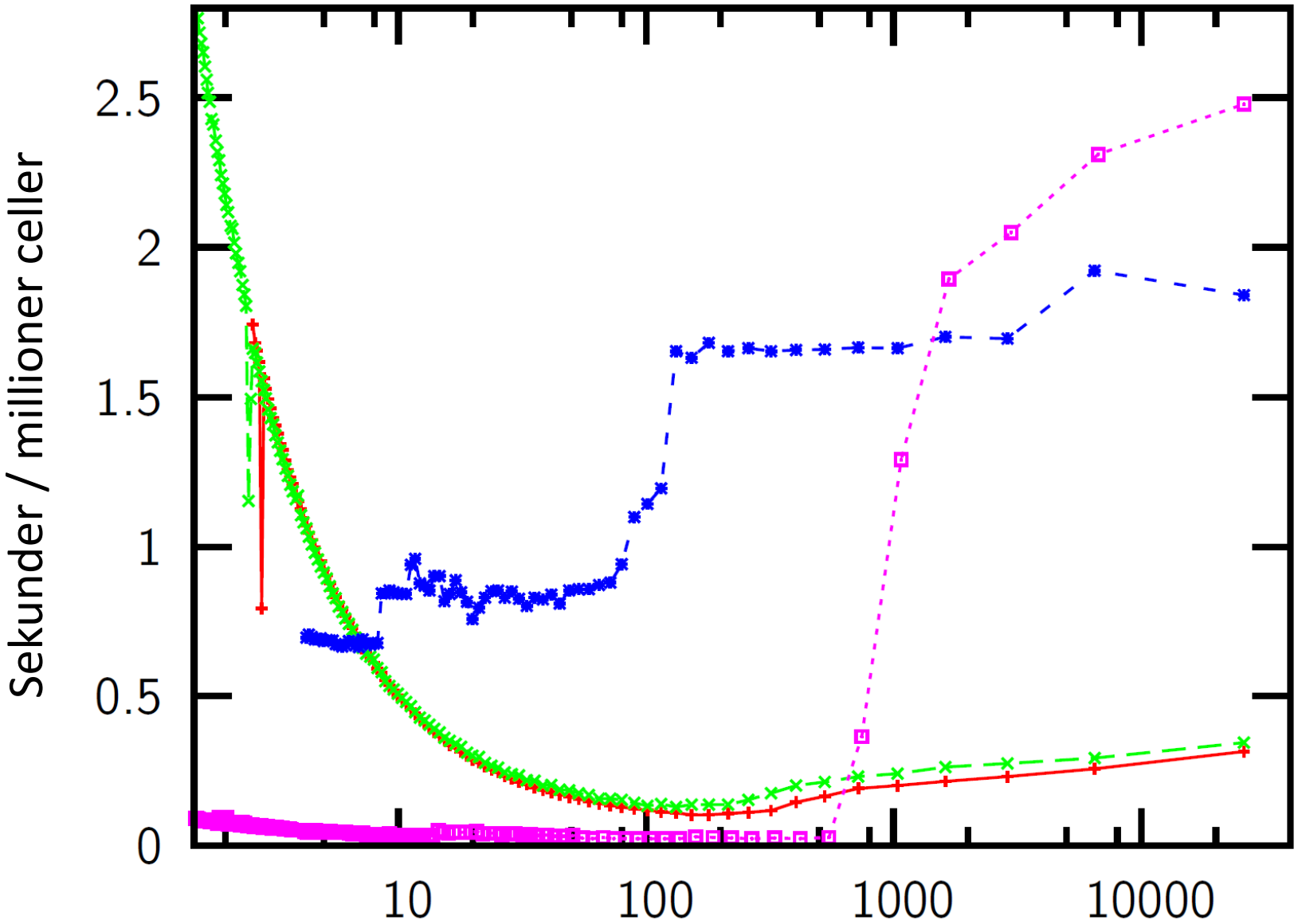
- Input n^2
- Output $\leq n^2$
- Sum $|S_i| \leq 2n^2$
- 1 + 3 scanner $\leq 3n^2$
- 2 gentagne scanninger

$$\sum_{\text{primtal } p} \left(\frac{n}{p}\right)^2 \pi(p) \leq cn^2$$

$\pi(p)$ er antal primtal $\leq p$
Primtalssætningen



Eksperimentelle Resultater



Algoritme 2

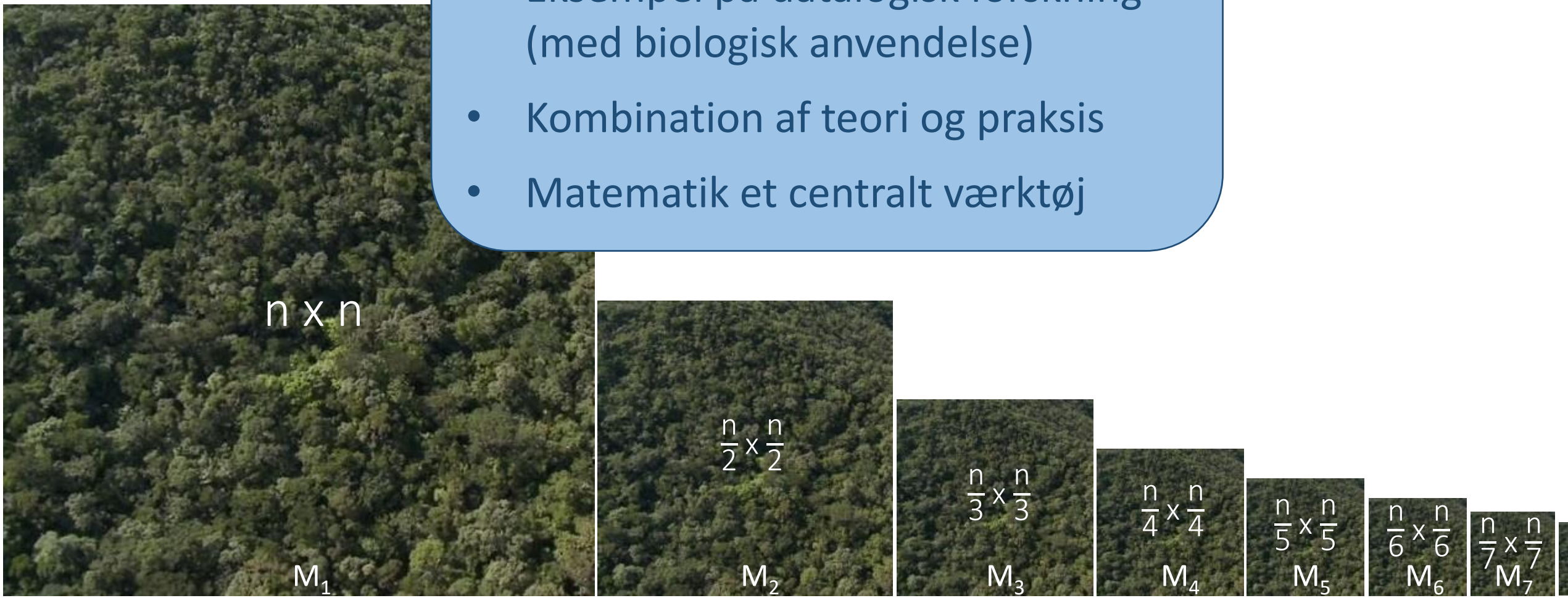
Cache aware

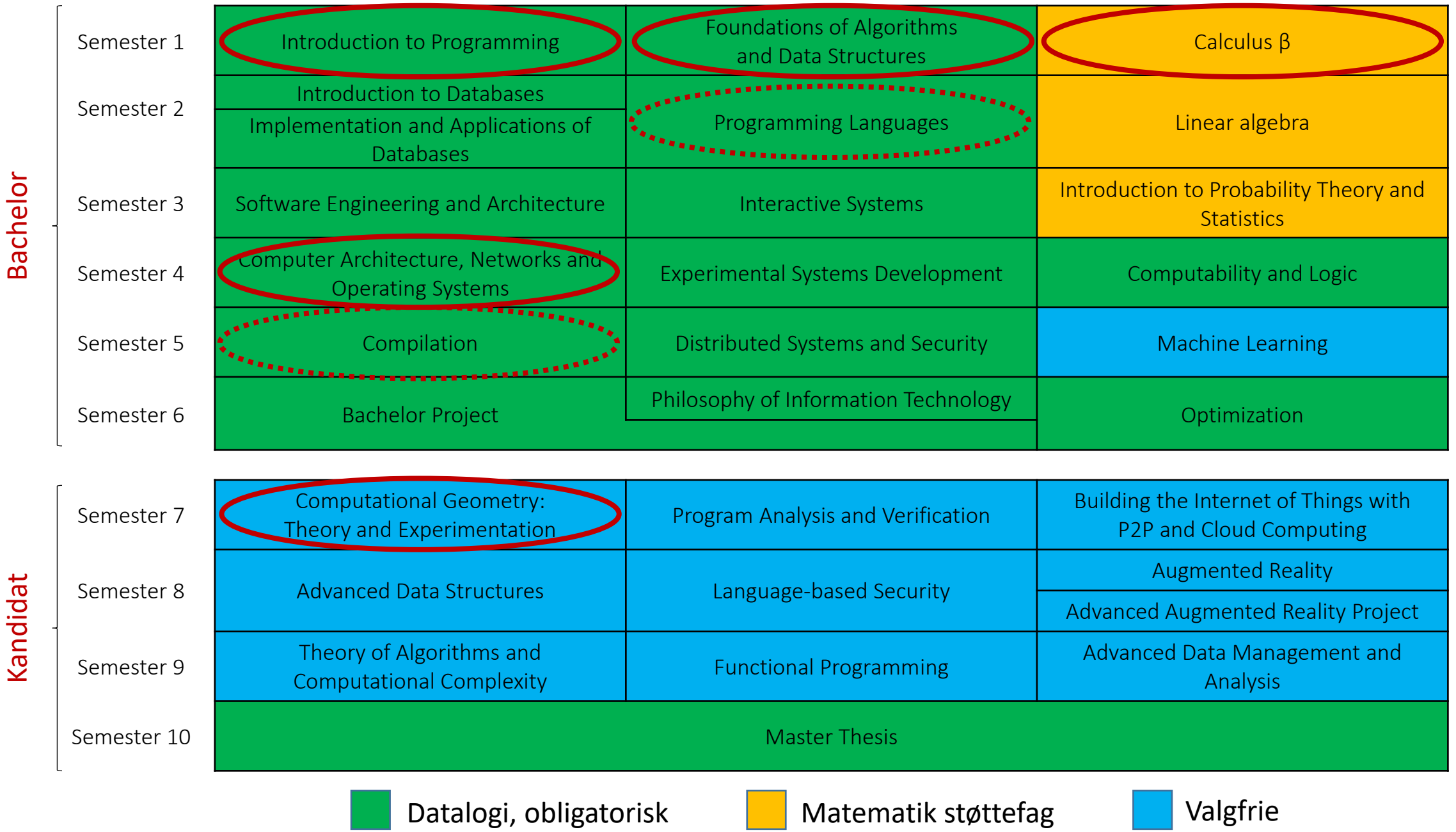
Arge, Haverkort, Tsirogiannis 2012

Cache oblivious

Opsummering

- Eksempel på datalogisk forskning (med biologisk anvendelse)
- Kombination af teori og praksis
- Matematik et centralt værktøj




Bachelor
Semester 1
Introduction to Programming
Foundations of Algorithms and Data Structures
Calculus β
Semester 2
Introduction to Databases
Implementation and Applications of Databases
Programming Languages
Linear algebra
Semester 3
Software Engineering and Architecture
Interactive Systems
Introduction to Probability Theory and Statistics
Semester 4
Computer Architecture, Networks and Operating Systems
Experimental Systems Development
Computability and Logic
Semester 5
Compilation
Distributed Systems and Security
Machine Learning
Semester 6
Bachelor Project
Philosophy of Information Technology
Optimization
Kandidat
Semester 7
Computational Geometry: Theory and Experimentation
Program Analysis and Verification
Building the Internet of Things with P2P and Cloud Computing
Semester 8
Advanced Data Structures
Language-based Security
Augmented Reality
Advanced Augmented Reality Project
Semester 9
Theory of Algorithms and Computational Complexity
Functional Programming
Advanced Data Management and Analysis
Semester 10
Master Thesis
Datalogi, obligatorisk
Matematik støttefag
Valgfrie

Tak for nu