Otto-von-Guericke-University Magdeburg
Max Planck Institute for Dynamics of Complex Technical Systems
Computational Methods in Systems and Control Theory

Dr. Jens Saak, Dipl.-Math. Martin Köhler
Website: http://www.mpi-magdeburg.mpg.de/csc/teaching/18ws/sc1/

Winterterm 2018/2019

Scientific Computing 1
8th Homework

Handout: 29nd Nov. 2018 **Return:** 7th Dec. 2018

Exercise 1: (4 Points)

Consider the matrix-matrix product $C = A \cdot B$ with $A, B, C \in \mathbb{R}^{n \times n}$. An easy pseudo code algorithm to compute the product is

for $i = 1, \ldots, n$ **do**
  for $j = 1, \ldots, n$ **do**
    $C_{ij} = 0$
    for $k = 1, \ldots, n$ **do**
      $C_{ij} = C_{ij} + A_{ik} B_{kj}$
    end for
  end for
end for

Think about how many times the entries of A and B are accessed. Why does this way of evaluating the matrix-matrix product not have a good data locality? Reformulate the matrix-matrix product to get a better data locality in the memory hierarchy.

Exercise 2: (4 Points)

Consider a CPU with a clock rate of 2.13 GHz and a cache bus width of 128 bit (assume `data_per_clock=1`). Compute the bandwidth between CPU and cache. A hardware manufacturer combines this CPU with dual-channel DDR3 main memory (1066 Mhz, approximate bandwidth 34 GB/s).

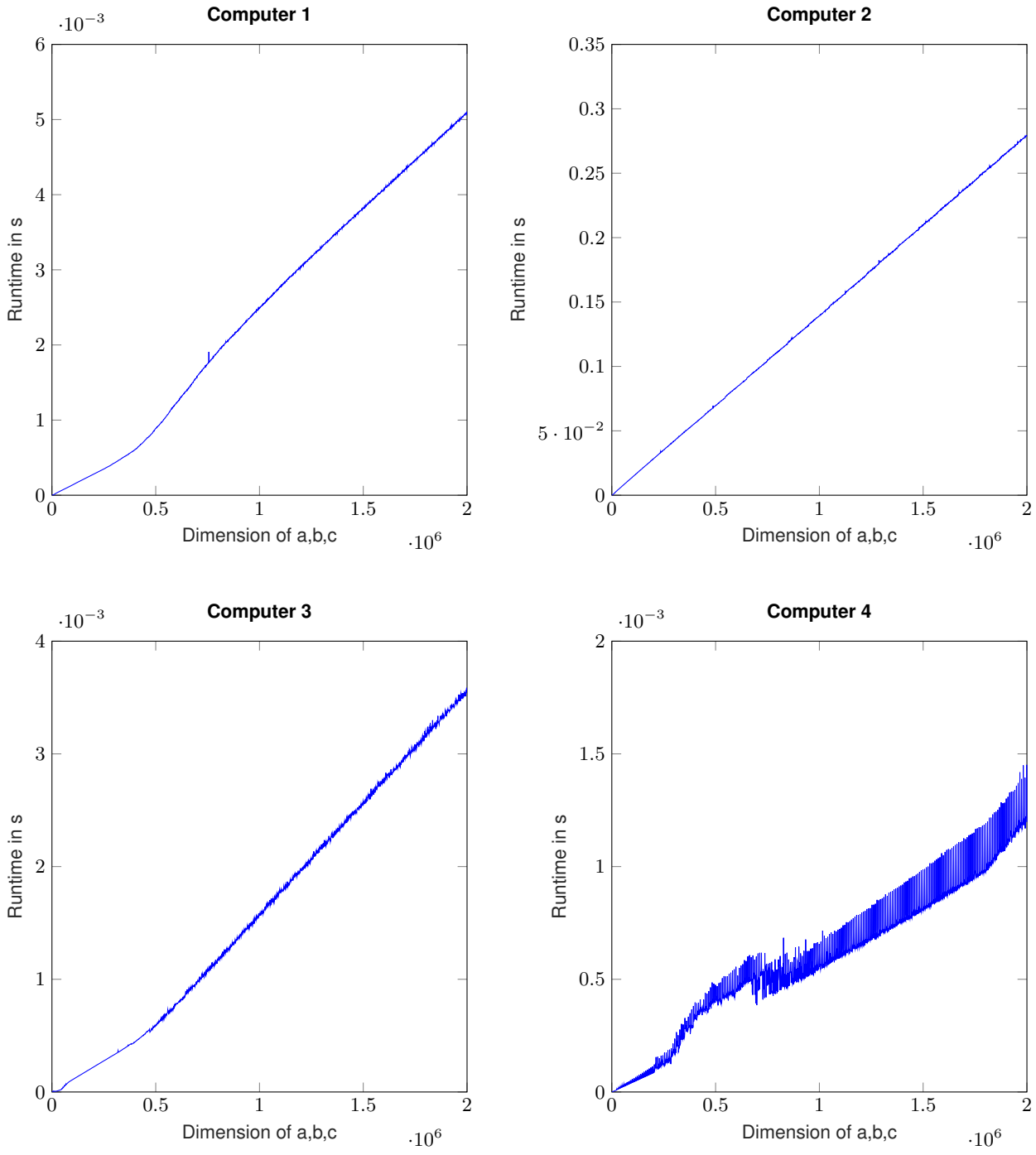
- Compare the cache transfer rate with the main memory transfer rate.
- Assume that the main memory has a CAS-Latency of 3. How many CPU cycles are spent waiting, at least, until a requested dataset arrives from the main memory to the CPU cache? What does this mean in the design of efficient algorithms?
- In which sense does the problem increase when a data portion has to be fetched from local storage, e.g., because it has been swapped?

Exercise 3: (4 Points)

We consider the following C function:

```
void axpy(int n, double *a, double *b, double *c) {
 int i;
 for ( i = 0 ; i < n ; i++){
  c[i]=a[i]+b[i];
 }
}
```

computing $c = a + b$ with $a, b, c \in \mathbb{R}^n$. We ran this for various $n \in \{1000, 2000, \ldots, 2 \cdot 10^6\}$ and took the average time for one vector add operation. On four different computers we got the following plots:



What can you recognize in the plots? Explain this behavior. Is it possible to determine any details of the memory hierarchy? Why is the runtime not linear?

Hint: The complete source of the benchmark program is available at: http://www2.mpi-magdeburg.mpg.de/mpcsc/lehre/2018_WS_SC/tutorial/axpy.c

Exercise 4: **(6 Points)**

Consider an electric signal can move at light speed ($2.997\,10^8 \frac{\mathrm{m}}{\mathrm{s}}$). Compute the distance the electric signal can travel during one clock cycle in the following applications:

- The memory-io clock rate on a mainboard:

RAM - Type	Clock Rate
EDO-RAM	66 MHz
SD-RAM	133 MHz
DDR-RAM	200 MHz
DDR2-RAM	533 MHz
DDR3-RAM	1066 MHz
DDR4-RAM	1600 MHz

- The CPU and L1/2/3 cache clock rate of:

CPU	Clock Rate
Intel Core i7-6700 (Base Frequency)	3.40 GHz
Intel Core i7-6700 (Idle)	900 MHz
Intel Xeon Silver 4110	2.1 GHz
IBM POWER8+	4.023 GHz

- The clock rate of different network techniques:

Technique	Clock Rate
Ethernet 100Mbit	31.25 MHz
Ethernet 10Gbit	417Mhz

Does the covered distance influence the design of electric circuits? Which problems will occur when the clock rates are driven even higher?

Exercise 5: **(5 Points)**

Assume that you are supposed to implement the following algorithm for large scale matrices A:

Input: $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and $\alpha = \{\alpha_1, \ldots, \alpha_{p_{max}} \mid \alpha_i \in \mathbb{R}\}$
Output: $Z := Z_{j_{max}}$ solving $AZZ^T + ZZ^TA^T + BB^T = 0$

1: $W_0 = B, \quad Z_0 = [\,]$
2: **for** $j := 1$ **to** j_{max} **do**
3: $\quad p = ((j-1) \mod p_{max}) + 1$
4: $\quad$ Solve $(A + \alpha_p I)V_j = W_{j-1}$ for V_j.
5: $\quad W_j = W_{j-1} - 2\,\mathrm{Re}(\alpha_p)\,V_j$
6: $\quad Z_j = [Z_{j-1},\ \sqrt{-2\alpha_p}V_j]$
7: $\quad$ **if** $||W_j^T W_j|| < tol \cdot ||B^T B||$ **then**
8: $\quad\quad$ Stop
9: $\quad$ **end if**
10: **end for**

Identify parts of the algorithm where you can run out of memory. Where can techniques like double buffering be used to save memory? Assume that it is cheaper to transfer $\mathcal{O}(n^2)$ many data items to local storage than to compute them with an $\mathcal{O}(n^3)$ method. How can this be used to further reduce the runtime?

Overall Points: 23