**TEAM SUMMIT:**

Week 03:

**Number 01: ( monte-carlo approx.)**

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**The C Code is provided. But here a quick overview what is special about every single one.**

**Atomic:**

atomic only supports a restricted set of expressions, all those expressions happen inseparable, either it happens completely

atomic operation has way less overhead than critical,

**Critical:**

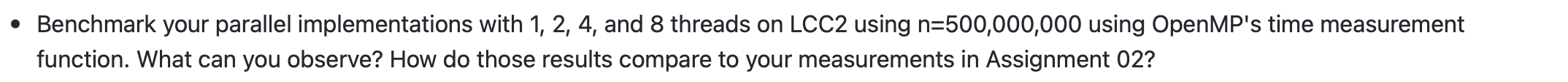
a critical block means that in this specific block, only one thread can be at a time means that, if thread 1 is in the critical block, all the other threads need to wait until it has finished

this synchronization means a lot of overhead, so like you can see later it is way the slowest one

**Reduction:**

reduction gets translated to a private variable which is incremented in the loop

atomic operation of reduction gets done at the end of the loop in an atomic clause



1 Thread:

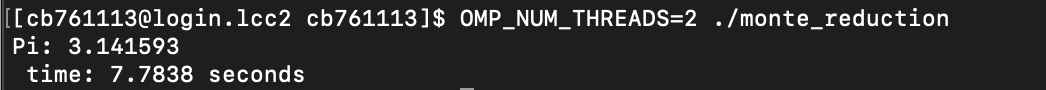
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

2 Threads:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



4 Threads:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

8 threads:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**How do they differ from last time? ( Assignment 02 )**

**Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung**



The yellow marked ones were the results from week02 from the pi approximation.

So, we had the same itertations#. So, this is can be ignored.

**1 Thread:**

**OLD** (55.77) vs

**NEW** 15.5(r) 23,7(a) 31,6(c)

**2 Threads:**

**OLD** (55.78) vs

**NEW** 7.7(r) 41,8(a) 115.2(c)

**4 Threads:**

**OLD** (244.06) vs

**NEW** 3.8 40.1. 222.16

**8 Threads:**

**OLD** (1099.81) vs

**NEW** 2.2(r ) 54.5 (a) 410.8(c)

**ALL different** parallel versions are way faster than the sequential one from the previous sheet. But the critical one is by far the slowest one. At two threads it is even slower than the sequential program from the previous exercise.

**Possible Reason?**

Because the critical section is a **section** so that there is way more payload in there.

**Usr/bin/time**

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**Ein Bild, das Text enthält.

Automatisch generierte Beschreibung**

**How do they differ?**

**User Time differences:**

3051 – Critical (50.85)

15.59 Reduction

412.37 Atomic

**Elapsed Time differences:**

6:50.73 – critical

0:2.04 - reduced

00:54.32 Atomic

**Resume:** The user time is way smaller than the elapsed time. That is because the elapsed time that the CPU gets charged for the expression. User Time is the wall clock time.

**What that means?**

So, if elapsed time > user time would mean that the cpu would have to wait for some other operations. For example, we have 5.85(User Time Critical) vs 6.50 (Elapsed Time critical)

**Does it match with the time measurement from the openmp measurement function?**

**From the usr/bin/time:**

6:50.73 – critical

0:2.04 - reduced

00:54.32 Atomic

Vs.:

**From openMp Function:**

**8 Threads:**

**OLD** (1099.81) vs

**NEW** 2.2(reduced) 54.5 (atomic) 410.8(critical) (== 6.8s )

**Resume:**

It is exactly the same time which was needed before **for 8 threads**. So, they differ in general, but match exactly with the measurement before with 8 threads. But they differ in for example one thread included.