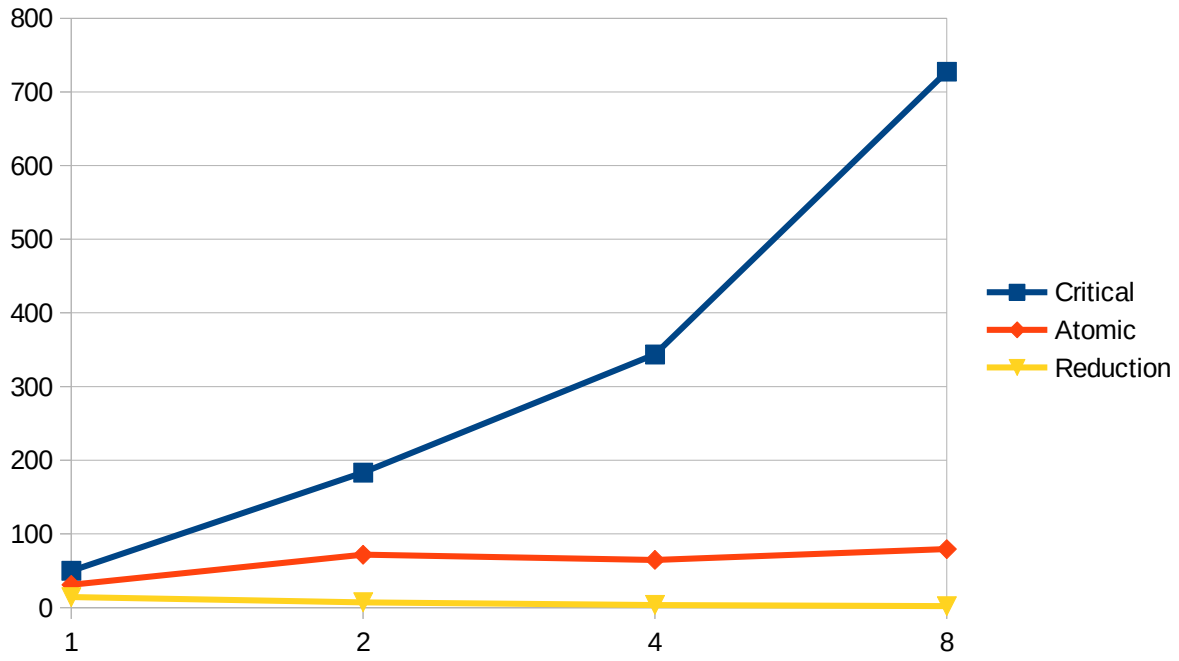
Aufgabe 1

Files:

mc_critical.c, mc_atomic.c, mc_reduction.c

Analyse:

Cores	Critical	Atomic	Reduction
1	49.780253	30.925873	14.198414
2	183.290659	71.774624	7.10845
4	343.681689	64.742035	3.555786
8	727.43277	79.363695	1.79246



x-Achse: Anzahl Cores
y-Achse: Zeit für Ausführung des Programms in Sekunden

Im Vergleich zu Woche 2: Es sind bedeutend schnellere Ausführungen möglich als bei der Benutzung von POSIX Threads. Vor allem bei der Ausführung mit einem Core sind alleder dieswöchigen Programme deutlich schneller.

Critical:
Mit einer steigenden Anzahl an Cores stegit auch die Zeit, die da Programm benötigt stark an. Das liegt daran, dass alle aktiven Threads jeweils darauf warten müssen, dass die kritischen Zonen von keinem anderen Thread passiert werden. Bei einer Problemgröße von 500 Millionen entsteht hier also viel Wartezeit.

Atomic:
Auch bei dieser Variante steigt die benötigte Zeit zur Ausführung des Programmes an, wenn es von mehreren Cores ausgeführt wird.

Reduction:
Bei dieser Variante ist eine viele schnellere ausführungszeit zu beobachten, als bei den beiden anderen Varianten. Bei einer Reduction werden alle Berechnungen im Thread lokal gespreichert und erst am Ende zusammengerechnet. Dies erspart sehr viel Wartezeit (im Vergleich zur Critical Region). Diese Variante ist auch die einzige, bei der hier eine Parallelisierung Sinn macht, da sich ihre Ausführungszeit bei einer ansteigenen Anzahl an Cores beschleunigt, bei dne anderen Varianten verlangsamt sie sich.