



TECHNISCHE UNIVERSITÄT CHEMNITZ

---

Fakultät für Informatik

Professur Rechnerarchitektur

# Auswertung

Chic-Übung Parallelrechner

Thomas Rückert

Chemnitz, den 9. Juni 2015

**Thomas Rückert,**  
Chic-Übung Parallelrechner  
Auswertung, Fakultät für Informatik  
Technische Universität Chemnitz, Juni 2015

# 1 Auswertung

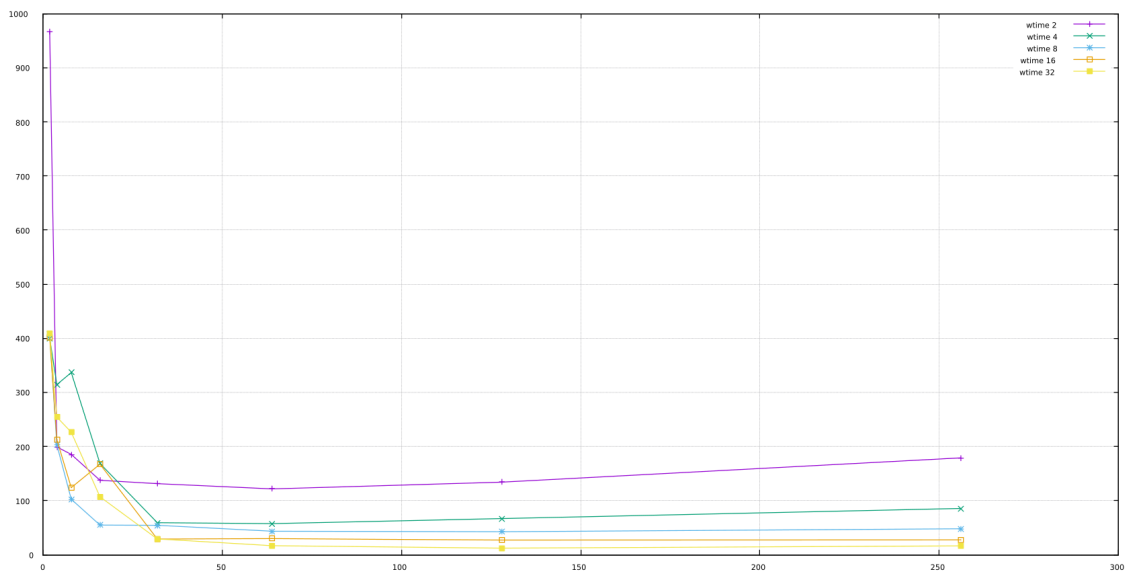
Eine kurze Auswertung zur Praktischen Übung: CHiC.

In Abbildung 1.1 ist ein Vergleich der gesamten Programmlaufzeiten in Abhängigkeit von der Taskmenge dargestellt. Dazu wurde mit verschiedenen vielen Knoten getestet. Der lila Graph repräsentiert dabei den Testlauf auf nur 2 Knoten des CHiC, während der gelbe Graph den Test mit 32 Knoten darstellt.

Bei allen Testläufen ist zu erkennen, dass die Laufzeit zunächst deutlich abnimmt, je mehr Knoten verwendet wurden, bis auf einige Abweichungen. Nach einer gewissen Anzahl ist dann jedoch das Minimum erreicht und die Werte beginnen wieder zu steigen.

Das ist nur logisch, da eine höhere Parallelität der Software ohne Unterstützung durch die Hardware quasi nutzlos ist. Daher erzielt zum Beispiel der Lauf auf 2 CHiC-Knoten über 64 Tasks immer schlechtere Ergebnisse, je länger dieser läuft. Dennoch skaliert die Anwendung sehr gut. Je mehr Knoten man bereitstellt, um so

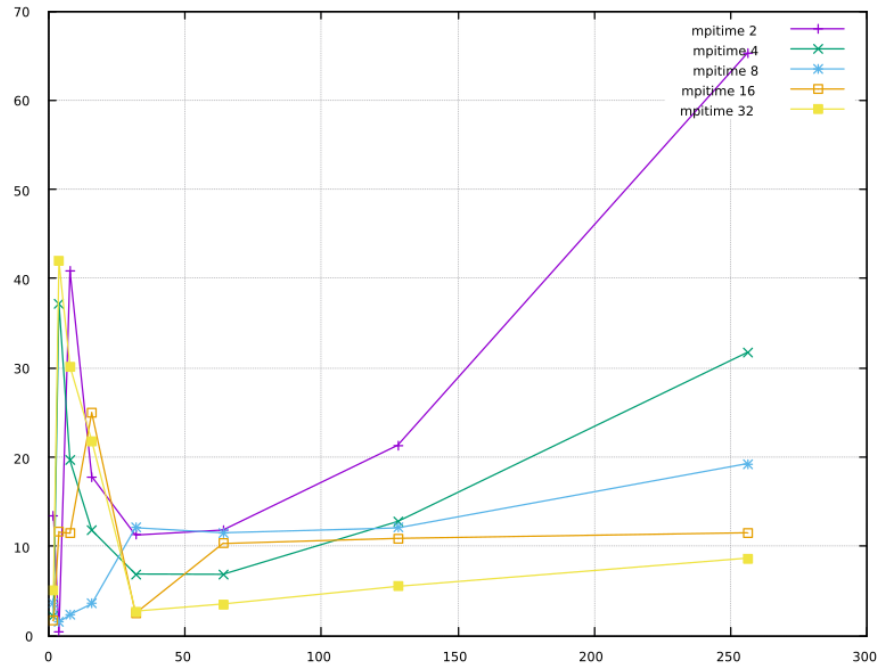
Abbildung 1.1: W-Time



besser (kürzer) wird auch die gesamte Laufzeit.

An Abbildung 1.2 ist der Grund für das Steigen der Laufzeiten bei hoher Taskzahl ebenfalls gut zu erklären. Man sieht hier den Vergleich zwischen den Laufzeiten, die für die MPI-Operationen nötig waren. Besonders im Fall von nur 2 CHiC-Knoten steigt diese Zeit sehr bald erheblich an, während im Falle von 32 Knoten nur ein sehr geringer Anstieg zu verzeichnen ist.

Abbildung 1.2: MPI-Time



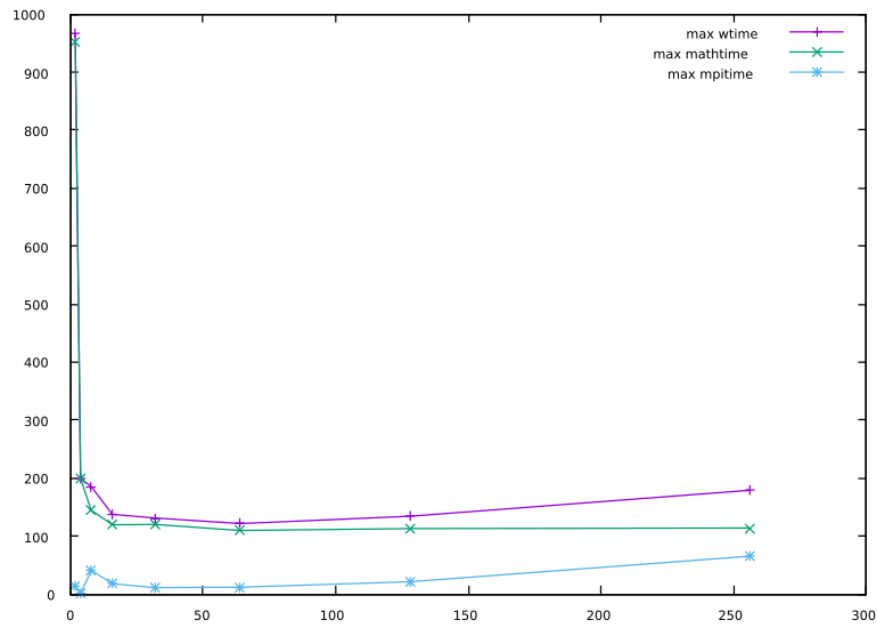
In der Abbildung 1.3 ist noch einmal Fall mit 2 Knoten zu sehen. Dort beschreibt der lila Graph die Gesamtlaufzeit, der Grüne die Zeit für die mathematischen Operationen und der Blaue die Zeit für die MPI-Operationen. Man kann wieder sehr schön sehen, dass die Laufzeit zunächst sinkt und dann nach kurzem stagnieren sogar zu steigen beginnt.

Hier kann man gut ablesen, weshalb die Zeit ab einer bestimmten Knotenzahl zu steigen beginnt. Während die Zeit für die Berechnungen selbst gleich irgendwann einen halbwegs konstanten Wert erreicht beginnt die Zeit für die MPI-Operationen zu steigen. Die Organisation der vielen Tasks auf deutlich weniger physischen Knoten ist irgendwann einfach zu viel overhead, im Vergleich zur Ersparnis bei der Berechnung. Darum kann keine Zeit eingespart werden.

Zuletzt wird in Abbildung 1.4 noch der Speedup verglichen. Man kann sehen, dass dieser deutlich sinkt und damit nicht optimal ist. Dennoch ist er auch bei 32 Knoten noch nicht auf 0 gesunken. Das Programm skaliert also nicht gerade perfekt, aber bei

---

Abbildung 1.3: Maximum von W- Math- und MPI-Time im Vergleich für 2 Knoten



der getesteten Menge von bis zu 32 Knoten sind dennoch immernoch Verbesserungen zu verzeichnen.

Wenn die Kosten für die Knoten keine Rolle spielen würden, dann würde sich die Nutzung von 32 Knoten immernoch auszahlen. Wenn man jedoch die Effizienz betrachtet, dann schneidet das Programm nicht sehr gut ab. Schon eine 2-stellige Anzahl von Knoten kann nicht einmal mehr einen speedup von 0,2 erreichen.

Abbildung 1.4: Speedup in Abhängigkeit von der Knotenanzahl

