Worum soll es heute gehen? Wir befinden uns im Abschnitt Parallelrechner und wir wollen heute mal versuchen, die Parallelverarbeitung mathematisch ein bisschen zu fassen. Um dieses Kapitel zu verstehen, brauchen wir ein paar Grundkenntnisse in Mathematik, also so ein bisschen Wahrscheinlichkeitsrechnung, was die meisten oder was alle von Ihnen eigentlich können müssten und einfach so die Kapitelvorher, wo wir schon Grundlagen zur Parallelverarbeitung mal gelegt haben. Worum geht es in dem Kapitel? Es geht darum, ein mathematisches Modell zu entwickeln und wir wollen mit Hilfe dieses Kenngrößen, wie zum Beispiel eben die zu erwartende Leistung, der zu erwartende Speedup und daraus resultierend abgeleitet die zu erwartende Effizienz und Wirkungsgrad unserer Verarbeitung auf einer Zielarchitektur. Das Ganze mit Hilfe von mathematischen Formen. Das heißt, wir schauen uns nochmal an, was ist denn überhaupt Parallelarbeit? Welche Grundbegriffe gibt es da? Wie ist Parallelarbeit definiert? mathematisches Modell, das heißt wir machen für eine Anwendung ein Parallelitätsprofil für einen idealen Parallelrechner, der so viel Prozessoren hat, wie wir ihn gerade brauchen, wo die Kommunikation auch sau schnell ist, also das heißt die nicht wehtut und versuchen das dann schrittweise in ein reales Bearbeitungsprofil überzuführen, um dann Aussagen treffen zu bei der parallelverarbeitung natürlich quantifizieren und wollen dann ganz zum schluss abschätzungen machen wie kann mein programm best case laufen und wie kann mein programm worst case laufen ja und in diesem korridor der sich zwischen best case und worst case aufspannt da liegt dann immer irgendwo die wahrheit ok schauen wir uns an was ist eigentlich parallelarbeit parallelarbeit ist ganz normal im täglichen leben das ist also eine übliche arbeitstechnik und ja das komplett die Problematik an sich, Parallelverarbeitung ist unabhängig vom Spezialfall, ja Parallelverarbeitung mit Rechnern. Deshalb machen wir auch mal so ein Beispiel nicht aus der Informatik. Irgendwann sind sich die Engländer und Schotten ziemlich auf den Keks gegangen und haben sich gegenseitig immer ein bisschen die Köpfe eingeschlagen, so dass man dann beschlossen hat, okay wir bauen eine Mauer zwischen den beiden, den sogenannten Hadrian's Wall. der wurde irgendwann gebaut zwischen England und Schottland. So, die Aufgabe ist jetzt relativ einfach zu definieren, natürlich sprachlich. Wir bauen eine Mauer. Heutzutage könnte man natürlich das Ganze auch zwischen Mexiko und USA aufbauen oder betrachten als Beispiel. Aber wir gehen da in der Geschichte lieber ein bisschen weiter zurück. Egal. Wir bauen eine Mauer. Wir haben N Mauer zur Verfügung und wir wollen diese Mauer so schnell wie möglich bauen. So ein Maurer, der ist dann ein äquivalent zu einem Processing Element und unsere Problemdomäne, das ist einfach eine Mauer, die hat die Länge L gleich 120 km und die Höhe H gleich 2 m. So, damit wir diese Mauer möglichst schnell bauen mit möglichst vielen Maurern, müssen wir unser Problem irgendwie aufteilen, damit die sich nicht nur gegenseitig auf den Füßen stehen, sondern damit die möglichst unabhängig voneinander möglichst schnell arbeiten können. Und dafür machen wir erstmal eine sogenannte Domain Decomposition, das heißt eine Problemaufteilung. Wir legen fest, okay, wenn wir n Maurer haben, dann machen wir für jeden Maurer, gehen wir mal davon aus, dass die alle gleich schnell sind, was natürlich ein bisschen unrealistisch schon wieder

ist, aber wir gehen davon aus und wir sagen, okay, dann machen wir n gleich lange vertikale Segmente mit der Länge klein l und wir haben einen Maurer pro Segment vorgesehen. diesen einfachen Fall, den zu erwartenden Speed Up. Ideal wäre es natürlich, wenn unser Speed Up, das heißt unsere Beschleunigung, unser Beschleunigungsfaktor, der gleich der Anzahl der Maurer ist. Aber damit das eintrifft, haben wir natürlich ein paar Voraussetzungen. Eine Voraussetzung ist, dass eben alle Maurer gleich schnell oder gleich langsam arbeiten und dass ich keinen Verlust durch irgendwelche Kommunikation oder Synchronisation habe. definiert durch die zeit die ein maurer benötigen würde um diese ganze mauer zu bauen dividiert durch die zeit die n maurer benötigen um diese mauer zu bauen so jetzt ist es natürlich normalerweise so dass im berg sind problems in der natur kaum existieren ja im berg sind parallel heißt die da ist steckt so viel Kommunikation erforderlich ist oder Synchronisation erforderlich ist, normalerweise muss ich an irgendeiner Stelle kommunizieren. Und bei unserem Mauerbeispiel ist es natürlich so, dass ich die Mauer stabil bauen möchte, das heißt die Steine liegen verzahnt und dann muss ich natürlich im Überlappbereich dafür sorgen, dass ich diese Verzahnung auch auflegen kann. werden. Wo ich aufpassen muss ist eben der Überlappbereich zwischen zwei benachbarten Mauerstücken. Da ist eine Zusammenarbeit erforderlich, das heißt da müssen in der richtigen Reihenfolge die Steine aufeinander gelegt werden. Ansonsten schaffe ich das mit der Verzahnung. Wir haben also hier so einen Überlappbereich und in diesem Überlappbereich findet Kommunikation statt. Muss natürlich berücksichtigt werden. Jetzt können wir sagen, okay, wer sagt denn, dass wir unsere Problemdomäne so aufteilen müssen, dass kleines Stück Mauer kriegt und wir dann diesen Überlappbereich haben, wir können doch unsere Mauer auch anders aufteilen. Wenn Sie sich das Bild anschauen, es erinnert natürlich ein bisschen an Pipeline-Verarbeitung auch. Wir können es so machen, dass der erste Maurer startet mit der untersten Reihe an Steinen, der nächste dann mit der zweiten Reihe, mit der dritten Reihe und so weiter. Das heißt, jeder Maurer bekommt so eine Steindreihe und wenn diese Pipeline mal eingeschwungen Allerdings ist das unter Umständen natürlich ungünstig, weil ja die Mauer bloß zwei Meter hoch ist, aber 120 Kilometer lang. Das heißt, ich habe nicht allzu viele parallele Schichten, die ich da verwenden kann. Ich könnte aber auch natürlich rein theoretisch beide Methoden miteinander verbinden. Das heißt, ich mache eine Problemaufteilung wie auf der Folie vorher mit relativ langen Einzelstücken und innerhalb der Einzelstücke mache ich dann diese Pipeline-Verarbeitung. Also es gibt Parallelarbeit zu organisieren und je nach Anwendung kann die eine oder andere vorteilhaft sein. Gut, der tatsächliche Speedup berechnet sich ja durch Zeit von N Maurern und wenn ich jetzt diesen tatsächlichen Speedup vergleiche mit dem idealen Speedup von N, dann habe ich so etwas wie eine Effizienz. Das heißt, die Effizienz ist ganz einfach der tatsächliche Speedup durch die Da der maximale Speed ab N sein kann, ist also das Beste, was ich an Effizienz erreichen kann, 1, also entspricht 100%. Und ansonsten werde ich immer einen Bruchteil davon erreichen. Dann gibt es eine andere Größe, die immer wieder auftaucht. Das ist, habe ich coarse grain, also habe ich groben Parallelismus oder habe ich fine grain Parallelismus. Das bezieht sich ein bisschen auf die Arbeitshappen,

die einem Maurer zugeordnet sind. wenn ich wie auf der folie vorher so ein ganz großes mauerstück an einen maurer verteile dann würde ich mal eher von den kors grain parallelism sprechen und wenn ich feingranular zusammenarbeitet das heißt nach dem stein kommt dann der nächste stein in der nächsten ebene dann ist es eher fine grain parallelism und natürlich muss diese korngröße im vergleich zum überlapp ja darf nicht zu Das heißt, ich darf nicht mehr Zeit im Überlapp verbringen als mit der eigentlichen Arbeit, weil sonst ist eh klar, produziere ich warme Luft und werde nicht in der Lage sein, meinen Rechner halbwegs auszulasten. So, wir können auch eine Formel aufstellen. Der Speedup ist n Arbeiter mal Länge des Teilstücks des Zugeordneten zum Beispiel, effizienz effizienz haben wir hier es durch n das heißt ich setze es einfach ein und dann habe ich eine effizienz und die effizienz kann ich umformen mathematisch und dann steht da dass die obst natürlich um 1 durch 1 plus l durch überlappte entschuldigen sie ein bisschen die darstellung sie müssen ein bisschen genau hinschauen also das nicht kursive das in einzelnen und das kursive das zu halten. Okay, ja wenn wir das jetzt mal so in ganz allgemeine Kurven auftragen, dann kann man sagen, okay, wenn wir hier unten auf der x-Achse die Anzahl der Prozessoren auftragen, dann wird unsere Effizienzkurve in etwa einen solchen Verlauf nehmen. Das heißt, wenn ich einen Prozessor verwende, dann habe ich automatisch 100% Effizienz, weil ich eben keine Synchronisation habe, keine Kommunikation, keine Loadbalancing-Probleme. Aber in dem Moment, wo ich Prozessoren dazufüge, kommt eben Synchronisation und Kommunikation als Verlustarbeit dazu und dementsprechend sinkt meine Effizienz und je größer die Anzahl der Prozessoren sind, desto niedriger wird meine Effizienz sein. Die Gegenkurve dazu ist der Speedup. Wir haben hier wieder die Anzahl der Prozessoren, wir haben hier den Speedup, also bei einem Prozessor habe ich einen Speedup von 1, ist klar. Und dann möchte ich natürlich bei 2 Prozessoren einen Speedup von 2 haben, bei 4 Prozessoren einen Speedup von 4 und so weiter. mein Ziel ist eigentlich einen linearen Speedup zu erreichen. In der Realität, aufgrund der dargelegten Verluste, wird es eben keinen linearen Speedup geben, sondern die Speedup-Kurve wird abknicken. Das ist soweit relativ ungefährlich, je nachdem wie stark sie abknickt. Gefährlich wird es dann, wenn es wieder nach unten geht, das heißt, wenn ich keinen Speedup dann mehr fahre, sondern eigentlich einen Slowdown fahre. Auch das ist in der Parallelverarbeitung durchaus üblich und passiert immer wieder, von noch mehr prozessoren dazu führt dass die ausführungszeit wieder größer wird das ist natürlich ganz schlecht weil dann haben wir auch mit der warme luft erzeugt aber die dann auch noch teuer erzeugt durch großen energieverbrauch das heißt das gilt es zu vermeiden und einfach um solche situationen vorher ein bisschen besser abschätzen zu können machen wir das mit diesen mathematischen modellen gut also fassen wir nochmal wir bauen das Hedwinswald durch N-Maurer in kürzestmöglicher Zeit. Wir haben einen Overhead, den berechnen wir aus dem Überlapp, und zwar ist der einfach das Verhältnis des Überlappbereichs