Wir stehen bei der Vorlesung im Kapitel oder im großen Abschnitt Parallelrechner und hier im Kapitel Geschichte der Parallelrechner am Beispiel, kennenlernen wollen und weil wir natürlich auch aus geschichte lernen wollen beziehungsweise wissen wollen wann welche technologien entwickelt wurden die wir auch heute noch in parallel rechner sehen wir stehen im jahr 1971 wenn sie sich so an rechner strukturen vorlesungen erinnern 1971 das vorher waren die rechner ja noch eher größere schränke und ein standard rechner der damaligen zeit jede uni hat es umgerät war die pdp 11 auch wenn es damals schon sehr früh war also 71 hat man sich damals schon gedanken gemacht okay vielleicht kann man durch zusammenschalten mehrere pdp 11 vielleicht kann man da ein parallel rechner bauen eine parallel rechner bauen der eben als eine einzelne PDP-11 und hat dementsprechend ein Projekt gestartet. Dieses Projekt wurde bezeichnet als Carnegie Mellon Multi-Mini-Prozessor, also nicht Mikro-Prozessor, so wie wir das kennen, sondern eben diese PDP-11s wurden als Mini-Rechner bezeichnet, im Gegensatz vielleicht zu einem großen Mainframe, der dann eben nochmal deutlich teurer und leistungsstärker ist. Man hat erstmal die Idee verfolgt, dass ein Parallelrechner einen global gemeinsamen Speicher haben muss. Das heißt, jegliche Kommunikation bei der Berechnung findet über diesen Speicher statt und dementsprechend hat man also an diese PDBLs einen riesenglobalen gemeinsamen Speicher hingepflanscht. Verbinden wollte man diesen Speicher über Caches und einen Crossbar Switch, also einen Kreuzschienenverteiler, dass alle Prozessoren, und man hat von 16 Prozessoren gesprochen, man hat für 16 Prozessoren geplant, dass alle Prozessoren an eine beliebige Stelle dieses Speichers schreiben konnten. Das erste Problem war, dass dieser Speicher sollte für die damaligen Verhältnisse riesengroß werden, also ein Modul 2 MB, das heißt 16 Module insgesamt 32 MB. Im Gegensatz dazu hatte die einzelne PDP-11 nur einen Speicher von 8 KB. riesen Faktor, der dazwischen ist und der Adressraum der PDB11, also die Speicher, die sie direkt adressieren konnte, der war nur 64 KB groß. Und deshalb befindet sich auch hier eine Schicht, die da bezeichnet wird als Adress Translator, das heißt heutzutage würde man natürlich MMU dazu sagen, also Memory Management Unit, man musste irgendwie versuchen, diesen großen Adressraum von 32 MB hier vernünftig zu adressieren, Adressraum von 64 KB, den diese einzelnen PDB11s hatten. Gut, dazu wurde noch ein gemeinsamer Interprozessor Interrupt Bus angeschaltet, also die Interrupts kamen über so einen gemeinsamen Bus aus Steuerungsgründen und jede dieser PDB11s verfügte natürlich noch über I.O. und Kommunikationsschnittstelle. So, was haben wir jetzt also aus diesem Shared Memory MIMD Rechner gelernt? Wir haben erst mal gelernt, dass es nicht so einfach ist Caches zu bauen für so eine Maschine. Also wenn 16 Prozessoren gleichzeitig auf Caches, unterschiedliche Caches, also jeder hat seinen eigenen Cache zugreifen, dann führt das zu Kohärenzproblemen. Darüber haben wir ja in der Vorlesung schon gesprochen und die sollten möglichst per Hardware gelöst werden und dieses Problem hat dieses Projekt nicht geschafft zu lösen. wurden nie realisiert, sondern die Prozessoren wurden direkt an den Crossbar angeschlossen. Man hat natürlich auch seine Erfahrungen mit so einem Crossbar gesammelt, also so ein Crossbar ist natürlich toll, weil alle gleichzeitig mit irgendwelchen Speichermodulen kommunizieren können, also im Idealfall 16

Prozessoren mit 16 Speichermodulen, wenn es da keine Überschneidungen gibt, das heißt es ist relativ leistungsstark das Ganze und der Speicher war natürlich groß geplant aber da haben wir eben das nächste problem mit der adressierung was wir darüber gesprochen haben ich muss also so ein adress translator bauen der ist mir überhaupt erst mal ermöglicht 32 megabyte zu adressieren ansonsten war das ganze sehr positiv weil der speicherzugriff der lag bei einer millisekunde und die befehls ausführungszeit einer pdb 11 bei 2,5 millisekunden das heißt wir haben hier deutlich genügend zeit die prozessoren Befehlen und mit Daten zu versorgen. Also das Problem, was wir heute haben, dass die Prozessoren so schnell sind und der Speicher so langsam ist, das stellte sich damals noch nicht. Deshalb war es auch ohne weiteres möglich, so eine Hierarchie zu bauen. Ein Problem war, wie schließe ich jetzt Peripherie an, an so ein System? Und das Einfachste ist, dass ich eben meine Peripheriegeräte den einzelnen Processing-Elements zuordne, hat halt dann den Drucker oder irgendein anderes Peripheralgerät zugeordnet. Und das ist natürlich ein bisschen schlecht, wenn ich dann auf einem anderen Processing-Element sitze und dann diese Hardware nicht direkt nutzen kann. Das heißt, man hat eine Betriebssystem-Erweiterung geschaffen, das sogenannte Hydra-Betriebssystem, das es eben ermöglichte, dass jeder Prozessor zum Beispiel den Drucker oder eine Ausgabe nutzen konnte. So, natürlich hat man sehr viel gelernt aus diesem System. ist und dementsprechend auch die Hardwareausfälle wesentlich häufiger auftreten und insbesondere bei einem parallelen System, wenn es jetzt an einem Rechner zum Beispiel eine kleine Störung gibt oder wenn da ein Fehler einprogrammiert wurde, dann führt es häufig zu Deadlocks. Das heißt, das ganze System bei einer globalen Synchronisation zum Beispiel wartet auf einen Prozessor und zur Not wartet es ewig und ewig warten, das ist natürlich Wärmeproduktion, eingeführt. Das heißt, die Prozessoren, die haben so alle regelmäßige Zeiteinheiten eine I'm still alive Meldung gegeben, der Watchdog hat es überprüft und wenn eben diese live Meldung nicht gekommen ist, dann hat der Watchdog Alarm geschlagen und hat gesagt, okay, einer der Prozessoren hängt möglicherweise und wir müssen da was tun. Man hat auch positive Erfahrungen gesammelt, nämlich es ist ja so am Anfang gerade ein bisschen in Frage gestellt worden, ob ich denn überhaupt ein Speedup schalte und man hat sogar linearen speedup erreicht für bestimmte anwendungen und man hat auch erkannt dass der adressraum von 64 kilobyte für den rechner der damaligen zeit schon auch zu klein war dass man also daran arbeiten musste größeren adressierungsraum brauchte und da das programmieren auf so einem parallelen system doch ein bisschen drücke ist hat man eben erkannt dass ich eine software umgebung schaffen muss die und Checkpointing erlaubt. Also das ist genau der Grund, warum wir uns die Geschichte der Parallelrechner anschauen. Man lernt eben mit jedem System dazu und verbessert sich. Die nächste Generation war wieder ein Multiprozessor mit gemeinsamen Speicher. Also gerade in den USA war man ja am Anfang schwer davon überzeugt, dass der normale Programmierer zu dumm ist, einen Parallelrechner mit verteiltem Speicher zu programmieren und deshalb kam eigentlich nur in Frage erstmal Multiprozessoren mit gemeinsamen Speicher aufzubauen. Die Idee bei der Sequent war dann gewesen, okay, wir nehmen Components of the Shelf, also wir

befinden uns jetzt im Jahr 1984 ungefähr, es gab schon erste Mikroprozessoren und das Konzept war, okay, man nimmt einfach Standard-Mikroprozessoren aus dem Regal und macht ein bisschen Software und Hardware dazu und baut sich dann seinen Parallelrechner, global gemeinsamen Speicher aufzubauen, auf den man relativ zügig, relativ schnell zugreifen kann. Wir haben damals schon einen 64-Bit-Datenbus und 32-Bit-Adressbus aufgebaut, also Harvard-Architektur, getrennte Busse für Daten und Adressen. Die Processing Elements, also die Prozessoren, waren 80386er Prozessoren mit 16 MHz getaktet. Dazu kam so ein Co-Prozessor für die schnellere eine Floating Point Beschleuniger, also Floating Point wirklich in Hardware gerechnet und nicht eben emuliert, was unendlich viel Zeit kostet. Man musste natürlich auch das Betriebssystem des Multiprozessors ein bisschen erweitern vom Monoprozessor Betriebssystem. Wir wollten, wir wollen ja, dass viele oder alle Prozessoren an einem Programm arbeiten. Das heißt, ich muss mir irgendwie so ein Laufzeitsystem oder so ein Loader bauen, der versetzt, mein Programm und in der Regel war es ein identisches Programm, auf alle Prozessoren zu laden und dann eben auch gleichzeitig zu starten und dann auch am Programmende, wenn die alle beendet sind, dass man das alles vom Betriebssystem her mitbekommt. Das muss also im Betriebssystem passieren, dann muss ich natürlich meine Programmiersprachen erweitern, ich muss ja es ermöglichen auf diesem global gemeinsamen Speicher zu arbeiten und Private Daten. Shared Daten sind die Daten, auf die jeder beteiligte Prozess zugreifen kann und Private Daten sind eben Prozesseigene Daten, wo nur der Prozess selbst drauf zugreifen kann. Und diese Erweiterungen muss ich natürlich, ich erfinde also keine komplett neue Sprache, sondern ich nehme eine gängige Sprache Vortran, Pascal, Lisp, Ada, Kobold, das war halt damals die Sprachen der Wahl und hat die entsprechend erweitert, um Konstrukte angelegt werden konnten. Was damals wirklich schon toll war, man hat sich Gedanken gemacht, wie man automatisch parallelisieren kann, also wenn ich zum Beispiel eine Duschleife habe, dass ich dann diese Duschleife automatisch parallelisiere, wenn keine Datenabhängigkeiten passieren. dann habe ich ein Konstrukt, das es mir erlaubt, dass alle Prozessoren gleichzeitig diese Duschleife abarbeiten. Gut, der Rechner war kommerziell halbwegs erfolgreich, wurde also in der Industrie eingesetzt, an den Unis eingesetzt als Forschungsrechner und Anwendungen ganz normal im kommerziellen Bereich, bei Transaktionssystemen und bei Datenbanken hat man damit gearbeitet. aus dass ich hier unten habe ich meine cpu alles ist an diesem sequenz systembus angeschlossen also diesen leistungsstarken schnellen bus und da hängen auch meine speicher erweiterungen dran die ich also modular aufbauen kann je nachdem wie viel geld mit zur verfügung steht konnte ich da einen relativ großen speicher also bis zu 240 megabytes aufbauen und ich konnte je nach und bei der S81 bis zu 30 Mikroprozessoren eben hier anschließen. Ich habe damit also ein skalierbares, modular erweiterbares System, das mit einem global gemeinsamen Speicher ausgestattet ist. Dazu gibt es natürlich noch ein bisschen Adapter. Ich muss ja auch irgendwie mit dem System rechnen können. ich muss Festplatten anschließen und da gibt es dann unterschiedliche Interfaces dafür. Gut, man hat dann aber schon angefangen, also Mitte, Ende der 80er Jahre gemerkt, dass die Systeme, wie sie jetzt hier gezeigt sind, dass

die nur bedingt skalierbar sind. Bedingt skalierbar, warum? Ja, ganz einfach, wir sehen hier als zentrales Element diesen Sequenzsystembus jegliche Form von Datenverkehr über diesen Systembus läuft, dann kann man sich vorstellen, dass der eben irgendwann auf jeden Fall dicht macht. Und so diese Anzahl von 30 Prozessoren, die man hier schon angeschlossen hat, das ist schon eine relativ hohe Prozessoranzahl. Eigentlich hat man auch damals schon gesagt, naja, so ein Bus verträgt vielleicht maximal 16 Prozessoren und dann wird es schon ziemlich schnell sehr eng auf diesem Bus. Wenn man jetzt also dann an größere Prozessoranzahlen denkt, auf die Idee, dass ein verteilter Speicher, das ist also ein Distributed Memory Rechner, möglicherweise die bessere Entscheidung ist. Man hat sich also jetzt dann Gedanken gemacht, okay, wir gehen jetzt mal nicht von 20, 30 Prozessoren aus, sondern wir gehen jetzt mal von 500 bis 1000 Prozessoren aus. Wie kann ich die denn vernünftig im Parallelbetrieb betreiben? So, die eine Idee, die man dann hatte erstmal, dass man sagt, okay, allen diesen Knoten das gleiche Programm läuft und wir betreiben dieses System als schnellen Co-Prozessor. Das heißt, der eigentliche Parallelrechner, der steckt jetzt hier unten, das ist also zum Beispiel ein 6-dimensionaler Hypercube, also 2 hoch 6 Prozessoren als Hypercube miteinander verbunden, die betreibe ich als schnellen Co-Prozessor, also hier laufen dann die Programme. Die Programmentwicklung treibe ich aber auf einem anderen System, also zum werden die Programme entwickelt, die sequentiellen Programme zumindest getestet und dann werden sie über so einen Intermediate Host, der nichts anderes als die Aufgabe eines Laders hat, also der lädt dann die Programme auf die Maschine, hier findet die Verarbeitung statt, die Ergebnisse werden zurückgemeldet und das Post-Processing findet hier wieder auf der Host-Maschine da oben statt. aufzubauen. Bekannte Uni ist die Caltech, also California, die dann einen Cosmic Cube aufgebaut hat, also so ein 6-dimensionaler Hypercube, 2 hoch 6 ist 64, also 64 Processing Elements und mit einer Werkstatt. Das wurde dann von Intel ja, Intel Personal Supercomputer praktisch von 1985 bis 1990 vertrieben. Das war auch eben ein mehrdimensionaler Hypercube. Ich weiß nicht mehr genau, wie viele Dimensionen da möglich waren, aber mindestens 10 Dimensionen. Ich glaube bis zu 12 Dimensionen, also bis zu 2000 Prozessoren, die ich da verbinden kann. In unserem Institut in Hannover am SRA, da gibt es den N-Cube. Hypercube mit eben vielen Prozessoren, mit 128 Prozessoren damals und entsprechendem Speicherausbau, gleiche Architektur, bloß als Hostrechner hatten wir damals eine SAN-Maschine, dieser Intermediate Host fiel aus, also wir haben direkt von der SAN-Maschine auf der SAN-Maschine cross-kompiliert und dann eben unsere Programme auf den Hypercube geladen. jetzt sind wir dann schon in der Gegenwart. Sie können sich im Internet unter top500.org können Sie sich jederzeit anschauen, was zurzeit weltweit die größten, schnellsten Parallelrechner sind. Diese Liste wird alle halbe Jahre aktualisiert und die Zahlen, die Sie jetzt hier sehen, sind die drei schnellsten Rechner weltweit vom Juni 2020, also ungefähr Da sehen wir, dass einmal vertreten ist ein Rechner in Japan. Dieser Rechner hat über 7 Millionen Cores, hat entsprechend viele Teraflop-Leistungen, also 415.000 Teraflops als erreichte Maximalleistung, also die tatsächlich von Programmen erreicht wurde in der Parallelverarbeitung. die ist natürlich deutlich höher, das

ist also die theoretisch erreichbare Höchstleistung, die wäre bei 513.000 Teraflop und Sie sehen, dass Sie da wirklich ein Kraftwerk brauchen, um so einen Teil zu betreiben. Wir sind nämlich bei 28 Megawatt Leistungsaufnahme, die ja dann auch wieder irgendwie weggekühlt werden muss. Industrie, Unternehmen, Forschungsanstalter dahinter stecken, um so ein Teil überhaupt betreiben zu können. Das heißt, da reden wir landesweit von einer Installation und solche Rechner gibt es eben nicht allzu häufig. Das sind ja auch die weltschnellsten. Dann kommen zwei Rechner in Amerika, die auch die Plätze 2 und 3 belegen. Hier die Kürzel DOE, das steht immer Department of Energy, also die haben natürlich sehr viel Geld und dann außerdem ist da National Laboratory dabei, das ist also sowas ähnliches wie die DFG in Deutschland, also Großforschungseinrichtung in Amerika und ja, Oak Ridge ist vielleicht auch deshalb bekannt, weil die waren damals am Manhattan Project beteiligt also in den 40er, 50er Jahren des letzten Jahrhunderts an der Entwicklung der Atombombe, also da steckt auf jeden Fall über das Department of Energy viel Geld, damit sie sich solche großen Rechner leisten Wir haben hier 2 Millionen Cores, das heißt ein Viertel der Cores, dementsprechend auch ein Viertel der Leistung, ungefähr Leistungsaufnahme ist auch noch bei 10 Megawatt. Und dann gibt es die nächste Installation, das ist auch um Sierra. Also die Systeme basieren alle auf, also die hier unten auf IBM Power Mikroprozessoren, die da in Japan stehen, da weiß ich gar nicht, welcher Mikroprozessor genau ist, der da drin steht. Gut, wir haben hier natürlich auch jede Menge Grafikkarten immer dabei bei den Boards, sonst komme ich nicht auf die viele Anzahl von Kernen. Und wir haben natürlich ein schnelles Verbindungsnetzwerk, also ein InfiniBand-Verbindungsnetzwerk, da kommen wir dann gleich noch dazu, das sehr schnell ist. Und hier haben wir wieder das Department of Energy, dann die NSA steckt ein bisschen dahinter, und LLNL. LLNL steht für Lawrence Livermore National Lab, also das ist auch wieder so eine, ähnlich wie Euclid, eine nationale Großforschungseinrichtung in den USA und die betreibt also den im Jahr 2020, im Juni 2020, drittschnellsten, weltweit drittschnellsten Rechner. Diese Spitzenplätze, das ist ganz interessant, das ist auch immer so ein bisschen ein Wettbewerb, wer hat jetzt den dicksten, ab und zu ist da China vorn dran, das mal war Japan vorn dran, also natürlich ist das auch so ein bisschen Prestige, dass man sagen kann, okay, ich habe jetzt den weltweit schnellsten Rechnen. Deshalb spielt an der Stelle dann ab und zu Geld auch nicht die tragende Rolle, sondern in solchen Ländern ist dann dieses Geld einfach zur Verfügung. Ja, Deutschland gibt es auch ein paar schnelle Installationen, aber deutlich weiter hinten. Ich weiß gar nicht, ob wir momentan in den Top Ten vertreten sind. Beispiel, wir haben am SRA uns auch Gedanken gemacht über so einen Parallelrechner. Im Jahr 2012 haben wir da einen beschafft und das ist vielleicht insofern interessant, als das Teil zwar klein ist, aber letztendlich die Architektur aller heutigen Hochleistungsrechner widerspiegelt. haben wir zwei 6-Core-Hochleistungs-Mikroprozessoren, also zwei Prozessoren mit jeweils 6 Cores. Und wir haben pro Core 4 GB Hauptspeicher, das heißt pro Compute Node dementsprechend 12 x 4 sind 48 GB. Wir haben eine schnelle Festplatte angeschlossen für temporäre Daten und für Swappen. hatten die M2075 waren damals eben aktuell als graphical processing unit also

wir reden vom jahr 2012 die daten waren für die damaligen zeit doch schon beeindruckend von dieser grafikkarte wir haben hier 515 gigaflops bei double precision floating points und bei single precision ist doppelte Zugriff auf diesen Grafikkarten selber mit 150 GB pro Sekunde und eben auch sehr große Speicher dann in diesen Grafikkarten mit 6 GB und die Anzahl von Kernen waren 448 Rechenkerne auf diesen Grafikkarten. Was brauche ich, um dann so ein System zu betreiben? Ich brauche ein schnelles Verbindungsnetzwerk. Ähnlich mit dem Gigabit Ethernet, allerdings ist die Latenz deutlich besser. Das heißt, die Zeit, die ich brauche, um eine Nachricht auf den Bus zu bringen und zu verschicken, die ist deutlich schneller. Hierfür benutzt das InfiniBand einen bidirektionalen Bus, Serienbus, und schafft es also in unter zwei Mikrosekunden so eine Datenverbindung aufzubauen, also so eine Sendoperation zu starten. Die Bandbreiten liegen im Gigabit pro Sekunde Bereich pro Kanal 2,5 und wenn ich zwei Kanäle habe, dann bin ich eben bei 5 Gigabit pro Sekunde. Das InfiniBand erlaubt es mir Kanäle zu bündeln und kann dann entsprechend eben diese Bandbreite erhöhen. Je nachdem wie viel ich bündel, also entweder vier Kanäle, dann bin ich bei 10 bis 20 Gigabit. dann bis zu 60 Gigabit pro Sekunde erreiche an Bandbreite. Dann gibt es immer, um dieses System zu betreiben, einen speziellen Knoten, das ist der Verwaltungsknoten. Der Knoten unterscheidet sich jetzt nicht großartig von allen anderen Knoten, also es sind im Wesentlichen die gleichen Prozessoren drauf. Ab und zu ist da keine Grafikkarte drauf, ab und zu ist eine Grafikkarte drauf. Das ist von System zu System unterschiedlich. 190.000 Euro gekostet und die Leistungsaufnahme, die liegt bei ungefähr 17 Kilowatt pro Stunde und natürlich erzeugt das Ding sehr viel Wärme, das heißt ich brauche nochmal in etwas so viel, um das Ding wieder runter zu kühlen. Und man muss natürlich bei so einem System eine Ausschreibung machen und muss dann sich entscheiden, was ist das beste System und angebote bekommen und hat dann zum kriterienkatalog entwickelt und was interessant war dass diese acht angebote sehr ähnlich waren also man hat im wesentlichen die gleichen prozessoren verwendet auf den boards man hat so die die gleiche ausbaustufe an speicher gehabt an anzahl der kurs an grafikkarten und die haben sich also alle nicht großartig unterschieden die rechnen in diesem bereich trotzdem muss man die miteinander vergleichen und dann Erstmal den Masterknoten, also das ist dieser eine Knoten, der mehr oder weniger nur für die Verwaltung des ganzen Systems eingesetzt wird, wo die Programmentwicklung drauf stattfindet und von wo aus die Programme dann alle aufs System geladen werden. Und dann hat man eben die einzelnen Compute Nodes, wie die aufgebaut sind. Da sehen Sie also die Mikroprozessoren, die drauf sind, die Anzahl der Kerne, die die haben, den Hauptspeicher, die die haben und die Peripherie, also Festplatte, die angebunden ist. findet man auch noch hier pro Compute-Knoten eine bestimmte Anzahl von Grafikkarten, also entweder eine oder zwei, in dem Fall von unserem Rechner, das ist das System, also für das wir uns entschieden haben, sind da zwei Grafikkarten drauf mit entsprechender Anzahl von Nodes, von GPU-Cores. Ja und so fasst man das Ganze halt zusammen, man berechnet dann daraus die theoretische Leistung, Markung noch realistische Leistungen, dann berechnet man die Energieaufnahme, berechnet so eine Leistung pro Watt und dann rankt

man das ganze System und am Ende trifft man dann seine Kaufentscheidung. Also bei uns war es eben dieses TransTech-System. Die wichtigste Erfahrung aus dem Prozedere, die ich gemacht habe, ist, dass diese Systeme alle sehr, sehr ähnlich sind. Und auch wenn Sie sich jetzt in den Top 500 Listen umschauen, dann sehen dass die Systeme alle eine sehr ähnliche Architektur haben und wie diese Architektur ausschaut, das wollen wir im folgenden Kapitel nun besprechen.