

Fortsetzung Rechnerarchitekturen

GPU; Spezielle Hardware

Wichtig insbes. speziell GPU um zu Reduzieren,
oder XeonPhi

Hauptunterschied zu CPU

sehr viele Kerne, spezialisiert auf bestimmte Operationen;
geringer Takt: Beispiel Pascal Architektur von Nvidia
(chip für Supercomputer (GP100))

56 x SM (multiprozessor): 32 double precision
und 64 single precision
shades

(Bei Grafikkarte 128 single precision
4 double precision)

= 1792 double precision und 3584 single precision.

Jede SM teilt sich L1 Cache für Texture (Array)
und 64 kB Shared Memory
Sowohl L2 Cache 4096 kB



Transfer zu CPU langsam

Rechen auf GPU durch nur am Anfang und Ende
(oder selten) mit CPU Interaktion (sonst langsam)

Wichtig: Präzision von Fließkommazelle geringer als bei CPU (Situation bzw bei GPGPU chips)
(Generell an den Datentypen, z.B. 32-bit)

Wegen massiver Parallelität => Programmierung und Befehlendes

Viele Recheneinheiten führen dieselben Operationen durch aber an andere Daten parallel. (Andere Arten von Befehlen)

Neuer Trend: Auch Rechenkerne speziell optimiert für Tensorberechnung auf der Grafikkarte
(Nvidia oder als externe Beschleuniger (Google in den Cloud Rechenzentren))

Damit die Shader nicht aus dem Takt kommen gibt es keine Sprünge (z.B.)

CPU

```

if (Bedingung) {
  Code mit viel Befehlen
}
  
```

Falls die Bedingung unwahr ist Sprung

GPA:

```

if (Bedingung) {
  Code mit viel Befehlen
}
  
```

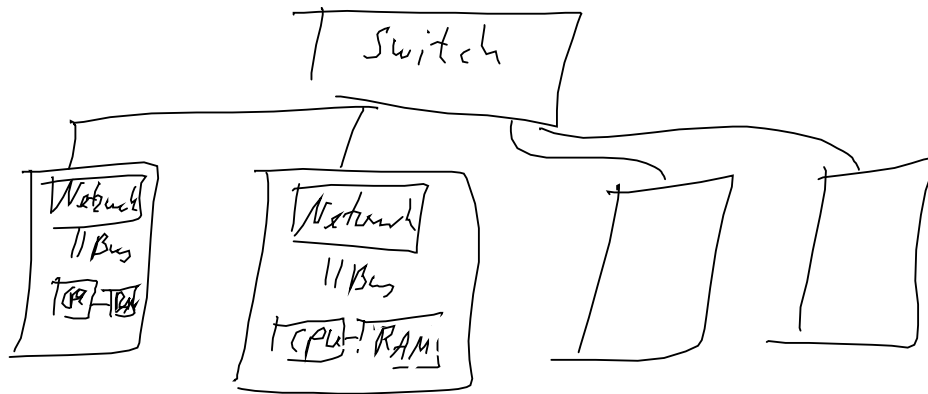
Falls Bedingung unwahr (Code wird trotzdem ausgeführt)

Ein Flag verhindert das Daten verändert werden.

Vorteil: Synchronisierung nicht so schwierig

Auch hier gibt es Möglichkeit zur Synchronisierung aber auch gilt es dies zu vermeiden.

Mehrere Rechner in Cluster



Bei großen Anwendungen Kopplung verschiedener Rechner:
 mehrere Möglichkeiten • 1 Prozess je Rechner
 • mit viel Threads (OMP)
 • mehrere Prozesse je Rechner mit
 Single Thread

Kommunikation in der Regel mit MPI Library
 (verschiedene Implementierungen openmpi, mpich2)

Vernetzung: • Herkunft Gigabit Ethernet
 (Übertragung zu langsam, Ethernet,
 IP Protokoll zu hoher Overhead)
 wichtiger (Lasten =) oft: Infiniband (z.B. 20 Gbit und
 mehr)

RDMA Protokoll
 (Remote Direct Memory Access)
 Direkter Transfer von Speicher zu Speicher
 ohne viel Overhead
 (Wenn korrekt konfiguriert macht MPI den
 Hintergrund)

Frage:

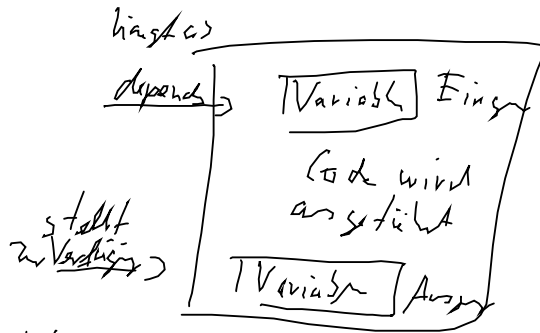
→ Knoten mit viel CPUs und viel Speicher
 versus mehr Knoten mit weniger CPUs und
 wenig Speicher

Vorteil bei CPU lastigen Anwendungen:
geringer Preis, geringe Anforderung
an Skalierbarkeit

Höhere Speicherbandbreite möglich, da Speicherbus Flusches
mehr Bus besser Transfer

Wichtig: Vermeide es zu viel Daten über das Netzwerk
zu schicken lassen.

Neue Programmiermöglichkeit Rechenoperation in einzelnen
Aufgaben zerlegen in tasks:



Problem in einzelne Task zerlegen.

Anwendung sowohl CPU wie auch GPU oder Tensorcores.

Frameworks

- OpenMP kann das
- Tensorframework z.B. tensorflow
für Tensor specialtip, Spatillisten
etc.

I.3 (Nicht-) Lineare Algebra für parallel programming

Vektorisierung der Probleme

Warum Vektorisierung? Viele Hardwarearchitekturen,
Software-Bibliotheken etc. stehen zu Verfügung,
um optimierte Matrix multiplication und
Vektor operation durchzuführen.

Beispiel für typische lineare Problem (formuliert):
 Lösung linear Gleichung:

$$\underline{A} \cdot \underline{x} = \underline{b}$$

\uparrow Matrix \uparrow Vektor

Lösung von zeitabhängigen Problem

$$\frac{d}{dt} \underline{v} = \underline{M} \cdot \underline{v}$$

\uparrow Vektor \uparrow Matrix

Eigenwertproblem

$$\underline{A} \cdot \underline{x} = \lambda \underline{x}$$

\uparrow Matrix \uparrow Zahl \uparrow Vektor

(und viele Probe spectral transform, singular Value Decomposition)

Beispiel für physikalische Problem

Schrödinger G.

$$H |\psi\rangle = E |\psi\rangle \Rightarrow$$

\uparrow Matrix \uparrow Vektor \uparrow Zahl

Wichtig ist es eine geeignete Basis zu finden!
 (s. Finite Differenzen und Quantenchemie)

$$|\psi\rangle = \sum_i c_i |i\rangle$$

\swarrow Spaltenvektor

Liouville von Master Gleichung

$$\frac{d}{dt} \rho = -\frac{i}{\hbar} [H, \rho]$$

(Hier ist die Dichtematrix ein Vektor

(z.B. $\langle i|\rho|j\rangle = \rho[i+Nxj]$ und

$-\frac{i}{\hbar} [H, \cdot]$ wird auf die Matrix abgebildet.

Helmholtz Gleichung

$$\Delta \underline{A}(\underline{v}, \omega) + \frac{\omega^2}{c^2} \tau(\omega) \underline{A}(\underline{v}, \omega) = 0 \rightarrow \text{Randwert}$$

\uparrow Matrix \rightarrow \leftarrow \uparrow Vektor

Besondere Form des Eigenwertproblems (nicht hermitisch)
 z. B. Wichtig für quantisiertes Lichtfeld.

Entwicklung $\underline{A}(\underline{v}, \omega)$ über Sitter oder Finite Element

Beispiel für Vektor und Matrixformate

Sequentielle Formate

Vektorformat: Vektor ist ein array von Fließkommazahlen:

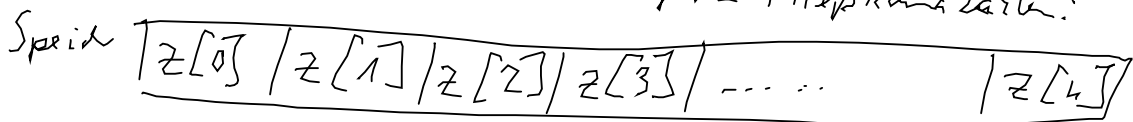
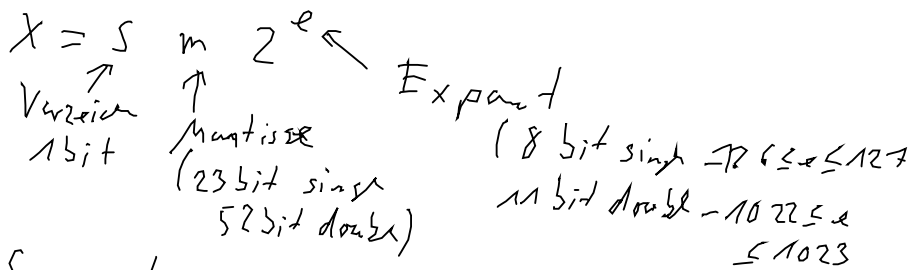


Abbildung Speicher als single [32bit] oder double [64bit]
 precise Fließkommazahlen.

z. B. IEEE 754



Wichtig: Genauigkeit beschränkt (Multiplikation und
 Addition sehr großem und
 sehr kleinen Zahlen schwierig)

Bei komplexen Zahlen besteht $z[i]$ aus zwei Werten.
 real und imaginär Teil (Ist ein c_i (HT) schon fertig
 implementiert ...)

Matrixformate

↳ ~~un~~ mögliche Möglichkeit (w) Block

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{pmatrix}$$

Zwei Möglichkeiten das in Speicher abzulegen:

1) Spaltenweise (column major) (z.B. Fortran, Matlab) BLAS

$$\boxed{\begin{array}{|c|c|c|c|c|c|} \hline m_{11} & m_{21} & m_{12} & m_{22} & m_{13} & m_{23} \\ \hline \end{array}}$$

Schnell bei Matrix Vektor multiplikation
(oder umgekehrt) (langsam)

$$\underline{M \cdot x} \text{ bzw. } \underline{M^T \cdot x}$$

2) Zeilenweise (Row major) (z.B. Standard bei C, ++)

$$\boxed{\begin{array}{|c|c|c|c|c|c|} \hline m_{11} & m_{12} & m_{13} & m_{21} & m_{22} & m_{23} \\ \hline \end{array}}$$

Schnell bei transponierter Matrix Vektor $M^T \cdot x$

(Bei den meisten Anwendungen / Algorithmen wird meist nur die
Anwendung der Matrix oder der transponierten Matrix gebraucht)

Wichtig, die innere Schleife muss immer die am meisten
variablen Größe ansprechen sonst langsam!

Low-level Routine für die Matrix format finden sich im
BLAS

An die Architektur optimierten linear Algebra Pakete,
bei Supercomputern von Herstellern...

Dieser Format ideal (gute Speichernutzung) für dünn besetzte
Matrizen.

$$\begin{pmatrix} 2 & 2 & 4 \\ 1 & 0 & 7 \\ 9 & 5 & 3 \end{pmatrix} \leftarrow \text{Matrix mit wenig } 0$$