Die Connection Machine

Folien zum Seminarvortrag am 26.01.2001

zum Thema

Unkonventionelle Rechnerarchitekturen

Steffen Schröder

sschroed@cs.uni-frankfurt.de

25. Januar 2001

W. Daniel Hillis

- geboren 1956 in Baltimore, Maryland
  - Jugend in Afrika und Indien
  - Vater Epidemiologe
  - Interesse für Biologie, später Computer
- MIT
  - wollte Neurophysiologie studieren, um das Gehirn genauer kennenzulernen
  - an Marvin Minsky (KI) verwiesen
  - arbeit an LOGO (Computerzugang für Kinder)
  - persönlicher Kontakt mit Minsky
- danach (auch TMC)
  - Bau der „denkenden Maschine“
  - Simulation von Evolution (an Sortieralgorithmen)
  - auch Spielzeugentwickler

Während Hillis noch 1980 dachte, die Organisation des Gehirns könnte helfen, schwierige Computerprobleme zu lösen, so glaubt er heute, dass mathematische Modelle mit Unterstützung durch leistungsfähige Computer zu neuen biologischen Einsichten führen können.

Einsatzgebiete und „Thinking Machines Corp.“

- Ziel: „denkende Maschine“

- spätere Einsatzzwecke (um 1993):
  - Hochenergiephysik
  - globale Klimamodelle (Treibhauseffekt)
  - AmEx Kreditkartendatenbank

- Ende der Connection Machine:
  - 1996 verschwand sie aus den Top500
  - TMC trennt sich vom Hardwaresektor
  - TMC spezialisiert sich auf Data Mining Software
  - TMC später von Oracle aufgekauft (siehe www.think.com)

Idee zur Connection Machine

- konventionelle Rechner:
  - Prozessor aus Silizium
  - Speicher aus Silizium

- Warum getrennt aufgebaut?

  zu Von Neumanns Zeit:
  - Prozessor aus schnellen, teuren Komponenten
  - Speicher langsam und billig

  Trennung heute unnötig!

  $\Rightarrow$: Zusammenlegung auf einen Chip!

Beispiele für paralleles Rechnen

- **Bildverarbeitung**

  jedes Pixel ein Prozessor, Netzwerk: Mesh

- **VLSI-Simulation**

  jeder Transistor ein Prozessor, variables Netzwerk

- **Semantische Netze (KI)**

  jedes Konzept ein Prozessor, variables Netzwerk

- **shortest path Algorithmus**

  jeder Knoten ein Prozessor

dafür benötigen wir:

- **viele Prozessoren**
- ein **anpassbares Kommunikationsnetzwerk**
- eine **erweiterbare Struktur** der Maschine
- **virtuelle Zellen**
- **Fehlertoleranz**

Design der Connection Machine

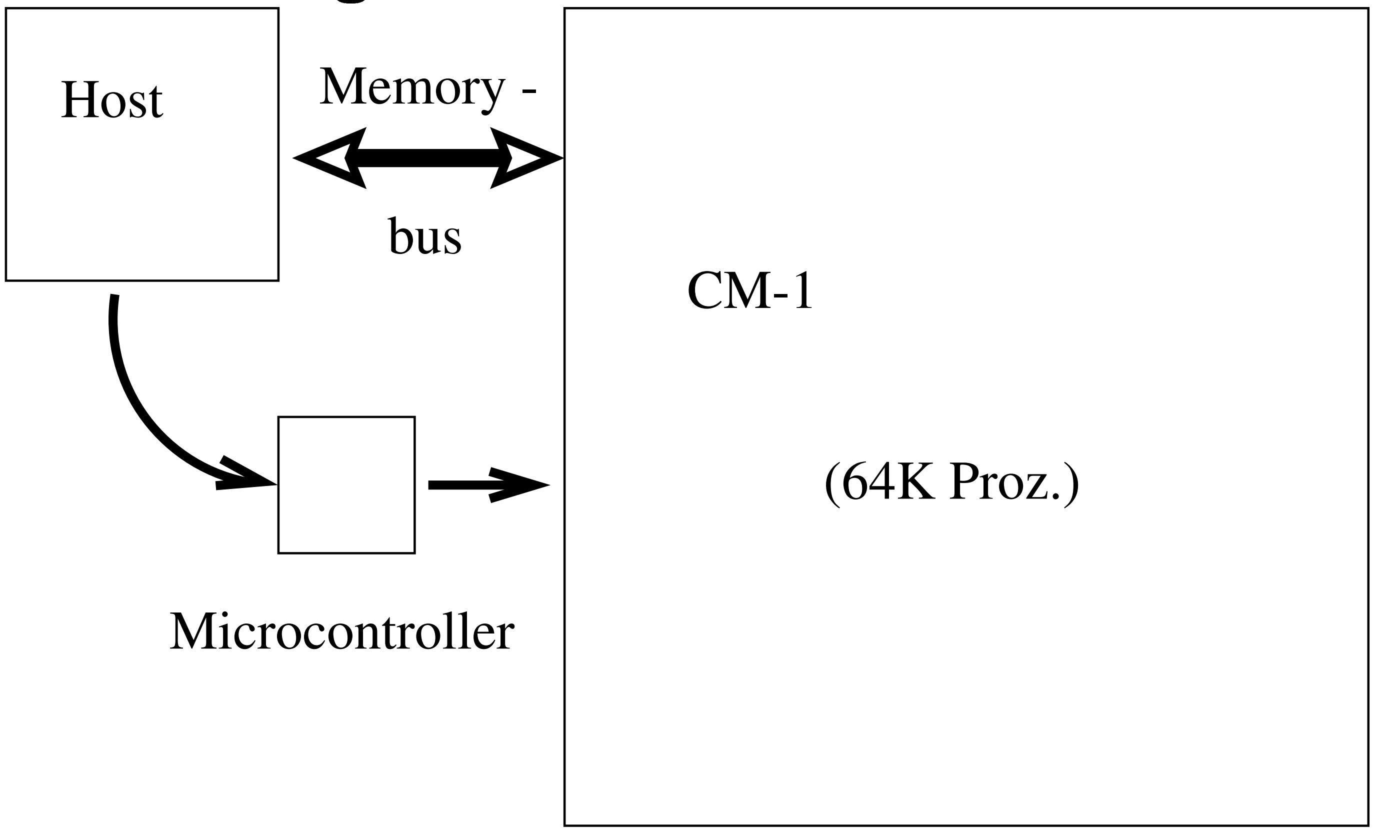

Abbildung 1: Die Connection Machine

Design – Allgemein

- **Prozessoren** — viele kleine oder wenige große?

  abhängig vom Einsatzzweck

- **Kommunikationsnetz** — generell (flexibel) oder starr?

  – generell ist besser, aber teuer
  – muss auf fixer Topologie simuliert werden (Netzwerk mit Routern)

  Bsp.: `Crossbar`, `Mesh`, `Würfel`, `Baum`, `Ring`, ...

- **Kontrollfluss** — SIMD oder MIMD?

  beides kann in erträglicher Zeit auf dem jeweils anderen simuliert werden.

Implementierungen

Forderungen an die Connection Machine waren vor allem:

- viele Prozessoren
- generelle Kommunikation
- Erweiterbarkeit

CM-1 – Zielsetzungen

Die CM-1 wurde 1986 gebaut.

- „Denkende Maschine“
- Designstudie — nicht Geschwindigkeit, sondern Flexibilität und Eleganz zählt
- LISP als einzige unterstützte Programmiersprache.

CM-1 – Implementation

- KI braucht symbolisches Rechnen
  - 1-bittige Datenpfade
  - mächtige ALU (Wahrheitstabellen)
  - einfache Zelle
    - ALU ($f : \{01\}^3 \rightarrow \{01\}^2$ beliebig)
    - 16 Flags (8 general purpose)

- „extremes RISC“, nur ein einziger Befehl

- SIMD

- Konditionalisierung

CM-1 – Organisation

- Kommunikationsnetz: Würfel mit 12 Dimensionen, Routing: Packet switched
- Chips: enthalten 1 Router und 16 Zellen ($16 * 2^{12} = 64K$), sitzen an den „Ecken“ des Würfels (der Chip enthält auch ein Mesh über seine Zellen)
- jeder Chip ist mit vier 4Kx4 Memory Chips verbunden.
- 32 solche Einheiten auf einer Platine (Modul)
- 16 Module auf einer Backplane
- 2 Backplanes - 1 Rack, 4 Racks - Würfel mit Kantenlänge ca. 1,3m

CM-1 – Daten

Zellen	64K
Speicher pro Zelle	4K bits
Speicher insgesamt	$2,5 \cdot 10^8$bits
Speicherbandbreite	$2,0 \cdot 10^{11}$bps
Prozessorbandbreite	$3,3 \cdot 10^{11}$bps
Kommunikationsbandbreite	$3,2 \cdot 10^7 - 5,0 \cdot 10^{10}$bps

Tabelle 1: Kenndaten der CM-1

CM-1 – Router

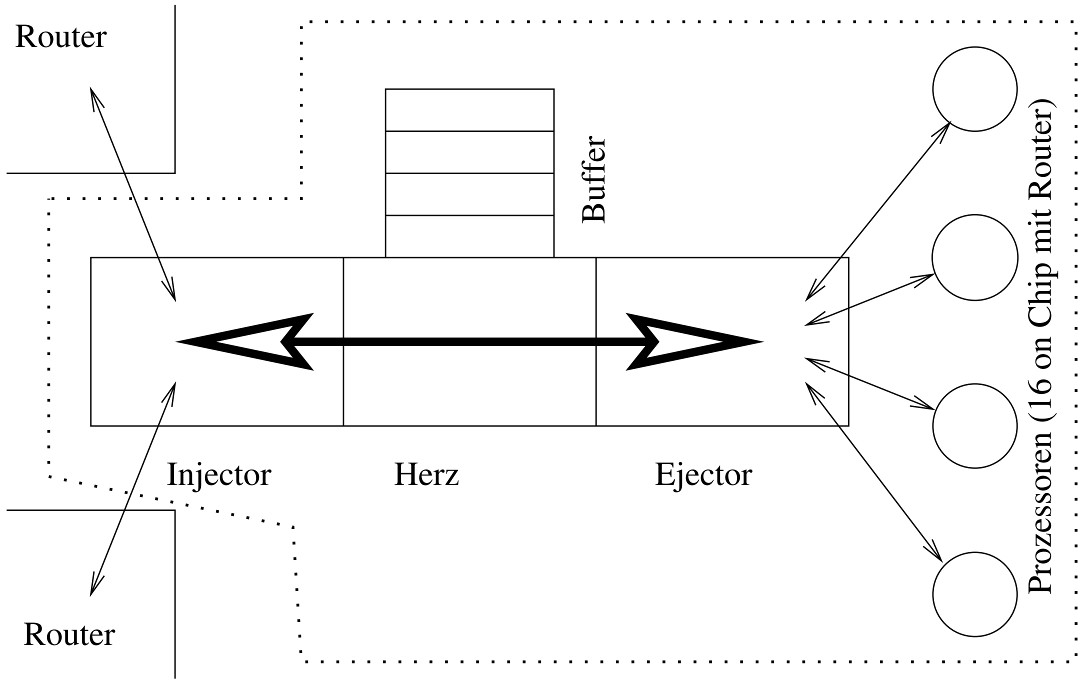

Abbildung 2: Der Router der CM-1

CM-1 – Router (2)

- relative Adressierung
- Puffern
- petit cycles (je 12 Dimensionsrunden)
- Staus ⇒ Umleitung
- Laufzeit: linear
- Fehlertoleranz:

  defekte Router/Chips werden von den Nachbarn erkannt und ausgeblendet.

CM-2

- genau 1 Jahr nach der CM-1 erschienen
- Probleme in der KI
- wenig Nachfrage auf dem Markt

  ⇒ „Abkehr“ von der KI und Einbau von FPUs (2.048 Stück).
- auch FORTRAN und C unterstützt
- bis 1991 in der Top500-Liste

CM-5

- Spezialisierung auf numerisches/wissenschaftliches Rechnen
- weniger, dafür mächtigere Prozessoren
- CHARM (coordinated homogenous array of RISC microprocessors) Architektur:
  - schnelles, breitbandiges Kommunikationsnetz, über das Prozessoren auf die Daten anderer zugreifen können
  - schnelles globales Synchronisationsnetzwerk, das es erlaubt, die gesamte Maschine in einen definierten Zustand zu bringen.

CM-5 – Daten

- 32, 1.024 oder 16k SPARC Prozessoren (32MHz und je 32MB RAM + 46bit Vektoreinheit)
- skaliert linear
- Distributed Shared Memory (über schnelles Kommunikationsnetz)
- MIMD
- 3 Netzwerke (Kommunikationsnetz, Kontrollnetz und Diagnosenetz)
- partitionierbar (bound checking auf Hardwareebene), aber auch time sharing
- I/O über Randzellen (RAID)
- virtuelle Zellen
- kein Host, sondern `control nodes` in den Partitionen
- die voll ausgebaute Maschine benötigt ca. 30x30 Meter Platz.

CM-5 – Der Prozessor

- SPARC mit 32MHz
- Cache nicht benötigt
- plus 4 Gleitkommapfade
- 64bit Vektoreinheit
- nicht auf speziellen Prozessor festgelegt (Zukunft)
- linearer Speedup (Linpack) – 45MFLOPS/node.

CM-5 – Kontrollfluss

- MIMD, weil
  - mächtige Prozessoren
  - schnelles Kontrollnetz $\Rightarrow$ rasche Synchronisation
  - auch Message-Passing Programme möglich
  - Normalbetrieb irgendwo zwischen MIMD und SIMD
  - der Compiler setzt Synchronisationspunkte.
- sowohl Kommunikationsnetz als auch Kontrollnetz werden über Memory-mapped FIFOs angesprochen, Schutzmaßnamen trifft die MMU

CM-5 – Das Kommunikationsnetz – Fat Tree – Bild

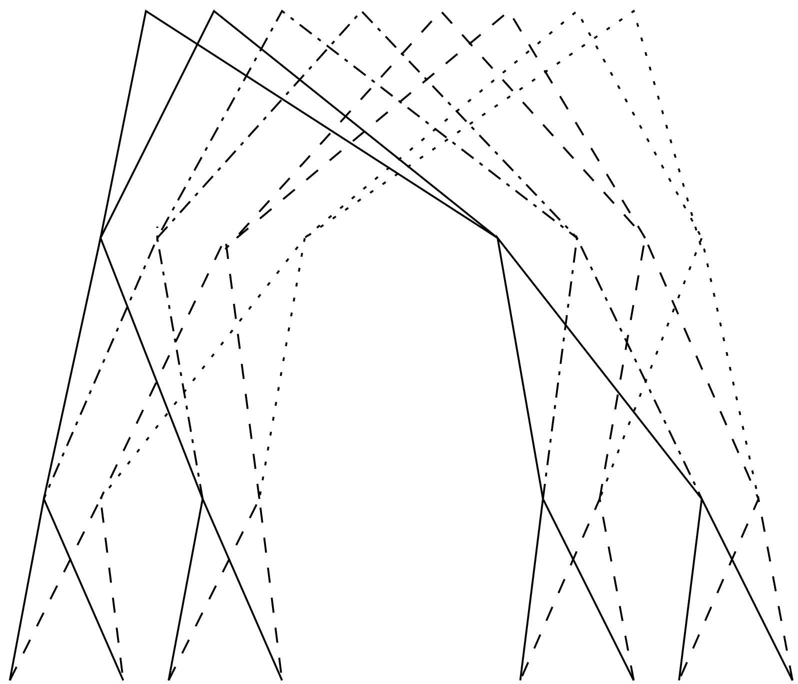

Abbildung 3: Ein Binärer Fat Tree

CM-5 – Das Kommunikationsnetz

- Fat Tree
  - bidirektionale Leitungen zu den Eltern (4 hoch, 4 runter) (20MB/sec)
  - jeder Router hat 4 Kinder
  - erste beide Hierarchieebenen (auf Backplane): zwei Eltern
  - darüber:
    vier Eltern
  - leicht erweiterbar (Verdoppeln)
  - lineares Bandbreitenwachstum
- kleiner Durchmesser
- **Fehlertoleranz**, da mehrfache Auslegung an der Wurzel
- einfache `Partitionierbarkeit` (Nachrichten nach oben werden abgeschnitten)
- Packet switched, Weg des Paketes nach oben über zufällige Eltern, damit keine „schlechten“ Patterns auftreten

CM-5 – Das Kommunikationsnetz(2)

- nur **relative Adressierung** der Knoten für normale Benutzer
- **Partitionierung** gemäß der Baumstruktur
- **Control Nodes** in den Partitionen als „Host“
- **Fehlererkennung** über Prüfsummen
- **garantierte Auslieferung** der Pakete
- **deadlockfrei**

CM-5 – Timesharing

Timesharing einer parallelen Maschine ist kompliziert.

Pakete, die noch im Netz unterwegs sind, könnten den neuen Benutzer behindern.

Lösung: **all-fall-down**

- alle Router (nacheinander) in all-fall-down Modus versetzen
- Pakete markieren und nach unten fallen lassen
- `jeder` Router leitet all-fall-down Pakete `gut verteilt` nach unten weiter.
- Pakete sind nun auf alle Prozessoren zum Puffern verteilt ⇒ keine Verstopfung!
- Ist der Benutzer wieder an der Reihe, werden alle Pakete wieder an den alten Empfänger losgeschickt.

CM-5 – Das Kontrollnetz

- einfache **Baumstruktur**, folgt dem Kommunikationsnetz
- strenges Datenformat (65 Bytes)
- **Synchronisation**
- **Broadcast** von Informationen innerhalb der Partition, z.B. Codestücke zur Ausführung für die Prozessoren
  - immer nur eine Zelle gleichzeitig
  - Paket wandert im Baum nach oben
  - wird an der Spitze herumgedreht
  - auf dem Weg nach unten vervielfacht es jeder Router
  - Kollisionen werden erkannt und gemeldet, Requests sind gepipelined.

CM-5 – Das Kontrollnetz (2)

- globale **Reduktionen** (OR, XOR, MIN, ADD, Präfix, Postfix)
  - zur Unterstützung paralleler Programme
  - z.B. in Datenbanken, Zugehörigkeit zu Mengen,...
  - die Prozessoren können über das Kontrollnetzwerk auch beim Router beantragen, von diesen Operationen nicht „belästigt“ zu werden, dieser sendet dann immer das jeweilige Identitätsdatum.
- teilt auch das **Ende von Datenbewegungen** mit
  - wann ist die Operation abgeschlossen?
  - wichtig für das datenparallele Programmieren

CM-5 – Das Diagnosenetz

- alle Einheiten, Router und alle Netze testbar

- für den normalen Benutzer unsichtbar

- Baumstruktur
  - Wurzel: Diagnoseprozessor
  - Blätter: Pods (= Diagnoseprozessoren an topologischen Einheiten)

- JTAG
  - IEEE Standard (1149.1)
  - serielle Diagnoseleitung über Chips
    - Boundary Scan Register
    - Test einzelner Funktionseinheiten
    - Chip Ein-/Ausgabe im Betrieb ausles-/änderbar
  - alle proprietären Chips
  - nicht Prozessor und RAM

- Adressierung von der Wurzel: (links,rechts,beides) = (01B), kumulierbar

- Fehlereingrenzung über globales AND / OR

Zusammenfassung

Das Konzept der Connection Machine ist sehr flexibel, nicht einengend:

- **viele Prozessoren — generelle Kommunikation**

Die Implementierungen können sehr unterschiedlich sein, gehörten aber einige Zeit zu den stärksten Rechnern der Welt.

Durch die vielen Prozessoren (die auch virtuell sein können) und die generelle Kommunikation sind sogar Simulationen anderer Rechnerarchitekturen im laufenden Betrieb möglich.

Dem Designer sind für seine Maschine je nach Verwendungszweck fast keine Grenzen gesetzt.

Trotzdem ist diese Architektur nach einiger Zeit der Blüte verschwunden, aber immerhin war sie einer der ersten Schritte vom Ein-Prozessor-Großrechner zum massiv-parallelen rechnen.

Auch zeigte sie, dass man viel mehr Anwendungen als gedacht parallelisieren konnte, wenn nur die Datenmengen groß genug sind.