A Review to CAM-8 - cellular automata machine[1]

Stephan Richter / 70242

24. Januar 2008

1 Motivation

Wie immer, wenn Wissenschaft betrieben wird, geht es auch bei CAM-8 um Modelle. Cam-8 ist primär ein Modellierungswerkzeug, z.B. für *mikrophysikalische Systeme*. Was aber ist ein Modell? Welche Probleme treten bei den Designaten auf? Wie können diese Probleme gelöst werden? Dies soll in den folgenden Abschnitten umrissen werden.

1.1 Modelle[2]

Im Kontext von Modellen geht es meist darum, dass man ein bestimmtes System, sei es real oder erdacht, (nach)bauen möchte. Dieses nachzubauende System - der Designat - wird in einem System - dem Modell - realisiert, dass seinem Vorbild ähnlich ist. Diese Ähnlichkeit folgt aus dem oft anhaftenden Zweck: In fast allen Fällen dienen Modelle dazu, kausale Zusammenhänge des modellierten Systems aufzudecken und zu erklären, und zum Verständnis grundlegender Mechanismen beizutragen um damit Vorhersagen über das Verhalten zu ermöglichen. In der Regel findet dabei eine Abstraktion statt, es wird sich auf zweckmäßige, *wesentliche* Eigenschaften des Systems beschränkt.

1.2 lokale Interaktionen

Die schon erwähnten mikrophysikalischen Modelle dienen entsprechend der Nachbildung und Vorhersage reeller oder auch künstlicher physikalischer Systeme und werfen einige Probleme auf, denen man mit CAM-8 gerecht werden wollte: Ihnen ist - da physikalische Systeme räumliche Systeme sind - stets eine grundlegende räumliche Struktur zugeodnet. Darüber hinaus werden physikalische Systeme oft so beschrieben, dass Interaktion (Informationsaustausch) lokal begrenzt, z.B. zwischen räumlich benachbarten Molekülen, geschieht. Schwerwiegendstes Problem bei der Simulation ist des Weiteren die auftretende massive Parallelität; jedes Molekül „rechnet“ ja für sich. Aufgrund der

genannten Eigenschaften war das Ziel des CAM-8-Teams die Entwicklung eines Computersystems, welches räumlich strukturiert ist und der den Systemen innewohnende Parallelität möglichst gerecht wird.

1.3 zelluläre Automaten

Geeignete Modellierungswerkzeuge sind schon seit den Anfängen der Computerzeit bekannt: Zelluläre Automaten vermögen eben die geforderte Aufbrechung des Raumes in Untereinheiten - Zellen - welche über lokal begrenzte Interaktionen miteinander kommunizieren. Das Problem bei der Modellierung mittels zellulären Automaten ist die Tatasache, dass herkömmliche Rechner das gewaltige Potential paralleler Berechnungen zellulärer Automaten aufgrund inhärenter Nicht-Parallelität kaum auszuschöpfen vermögen.

2 das Konzept von CAM-8

Aus diesem Grund stellten sich die Entwickler von CAM-8 der Aufgabe, eine neue, besser geeignete Rechnerarchitektur zu entwickeln, die genannten Problemen besser gerecht wird. Es ist naheliegend, dass man die erwähnte Parallelität nur mit einem Multiprozessorsystem erreichen konnte. Der Machbarkeit wegen wurde versucht, das Computersystem aus Komponenten aufzubauen, die auf dem Markt verfügbar waren, wobei sich speziell angepaßte Komponenten natürlich nicht ganz vermeiden ließen. Immerhin hatte man auf Grund früherer Versionen der *cellular automata machine* hohe Ansprüche gestellt: Das System sollte auf möglichst viele CA-Modelle anwendbar und problemlos skalierbar sein.

2.1 Entwicklung von CAM-8

Konsequenterweise erdachte man also ein Computersystem, in dessen Berechnungen der zu simulierende Raum gleichmäßig in kleinere Portionen aufgeteilt wird, wobei jedes Modul des Rechnersystems für eine solche „Raumportion“ zuständig sein sollte. Hier zeigt sich, dass diese Architektur keine massive Parallelität realisiert; immerhin kann jedes dieser Raumteile immernoch tausende „Atome“ (im Sinne von unteilbaren Basiskomponenten des simulierten Systems) umfassen. Diese werden wie bisher sequentiell simuliert, teilen sich also die Rechenzeit eines Rechenmoduls. Die Module selbst werden dazu in einem dreidimensionalen Gitter durch Datenleitungen vernetzt - dies ermöglicht die lokale Weitergabe von Daten zwischen den Knoten. Zusätzlich ist aber auch ein Netzwerk in Baumstruktur vorhanden, welches die Gesamtheit der Module an einen Frontend-Rechner koppelt. Letzterer ist notwendig, um Daten, wie etwa das Programm und Simulationsparameter den einzelnen Modulen zukommen zu lassen. Außerdem übernimmt dieser Computer auch die Erfassung von Zuständen ausgewählter Module und macht damit das Systemverhalten beobachtbar.

3 die Arbeitsweise

Die Module arbeiten gleichgetaktet und ermöglichen somit, dass alle „Atome“ parallel ihren Zustand aktualisieren, wobei wie schon beschrieben, die „Atome“ eines Moduls nur quasi-parallel aktualisiert, in Wirklichkeit jedoch sequentiell bearbeitet werden. Die Beschränkung auf gleichgetaktete Module ist zwar eine kleine Einschränkung der Universalität, sie erlaubt ja nur synchron geupdatete zelluläre Automaten (das sind fast alle), erlaubt dafür die kollisionsfreie Kommunikation der Knoten miteinander: Der synchrone Datenaustausch zwischen den Modulen geschieht über statische Routen und adressierten Zugriff auf RAM-Blöcke benachbarter Module. Innerhalb eines Arbeitstaktes werden also Nutzdaten (Zustände) aus den zum Knoten gehörigen DRAM gelesen, per Table-Lookup die Folgezustände bestimmt und in den DRAM an die gleiche Stelle zurückgeschrieben. Zu festgelegten Takten erfolgt die Kommunikation, bei der jedes Modul in definierter Weise direkt auf den DRAM seiner Nachbarmodule zugreifen kann. Die genannten Lookup-Tables sind in separaten, schnelleren SRAM-Bausteinen gespeichert und stellen das eigentliche (lokale) Programm dar, von welchem jedes Modul eine Kopie besitzt.

4 Umsetzung in reale Hardware

Natürlich sollte dieses System nicht nur reine Theorie bleiben. Für Versuchszwecke wurde ein CAM-8-Rechner gebaut, der aus 8 Modulen bestand. Pro Modul wurde das System mit 64MB DRAM und 2MB SRAM ausgestattet, mit Zugriffzeiten von 70 bzw. 20 Nanosekunden. Die Logik jedes der Module umfasste etwa 2 Millionen Gates - zur Zeit des Baus arbeitete man mit Auflösungen von etwa 1.2 Mikrometern. Der Taktfrequenz lag bei 25MHz. Bemerkenswert ist an dieser Stelle die Tatsache, dass heutiger RAM zwar wesentlich größere Kapazitäten besitzt, die Zugriffszeiten jedoch mit dieser Entwicklung kaum schrittgehalten haben und immer noch in den gleichen Größenordnungen liegen. Und auch die Prozessoren des Systems dürfen als veraltet angesehen werden: Zur Zeit sind 90 Nanometer durchaus üblich, an höheren Auflösungen wird gearbeitet, entsprechend ist auch die Zahl der Gates pro Prozessor enorm gewachsen. Für die Zeit der Realisierung von CAM-8 entsprach die Kapazität dieser Hardware jedoch in etwa der einer low-end workstation oder eines PCs.

4.1 erreichte Leistug

Um so erstaunlicher sind die Leistungen, die gemessen wurden: Man hat auf dem 8-Modul-System unter anderem zelluläre Automaten mit 500 Mio. Sites und je einem Bit pro Site - d.h. jede Zelle kennt nur zwei Zustände - simuliert. Hier konnten etwa 3 Milliarden Site-Updates pro Sekunde erreicht werden. Für einen etwas kleineren Automaten (32 Mio. Sites) mit 16bit pro Site erreichte man immer noch 0.2 Milliarden Zellen-Updates pro Sekunde. Dies entsprach in etwa der Leistung zeitgleich existenter

Supercomputer bei der Simulation von zellulären Automaten! Über diese reinen Rechenleistungen hinaus bringt das System einige weitere Vorteile: es können während der laufenden Simulation Daten vom Frontend erfasst und in Echtzeit zu Video zusammengestellt werden! Außerdem wurde das System vorbereitet für den Anschluß einer Video-Kamera, um Videodaten in Echtzeit mittels CA-Regeln zu verarbeiten.

4.2 Programmierung der Maschine

Bei der Programmierung solcher zellulärer Automaten gilt es dafür zunächst 6 Hauptparameter einzustellen:

- **Dimensionen** sowie **Größe und Form** des Raumes
  War man bei allen Vorgängerversionen der *cellular automata machine* auf nur zwei Dimensionen und eine sehr begrenzte Ausdehnung in diesen Dimensionen beschränkt, so konnte man mit diesem System erstmals auch höherdimensionale Räume modellieren: CAM-8 erlaubt bei der Definition des Raumes beliebig viele Dimensionen mit einer (theoretisch) unbegrenzten Größe in den einzelnen Dimensionen. Praktische Grenzen für die Raumform, aber auch für die Bit pro Zelle, sind nur durch den verfügbaren Speicher gegeben, wobei aufgrund der freien Skalierbarkeit in drei Dimensionen tatsächlich eine Barrierefreiheit erreicht wird. An dieser Stelle sei noch erwähnt, dass, um die Verwendung spezieller Lookuptables für die Außenbereiche zu vermeiden, die Randbedingungen für das Modell zyklisch gewählt wurden, d.h. das zelluläre Raster stellt eine Torus im Hyperraum dar.

- die **Bits pro Site** legen die Zahl der möglichen Zustände einer jeden Zelle fest

- **Anfangszustand** der Zellen

- **Regeln** für die Datenverarbeitung

- Richtung und Schrittweite von **Datenbewegungen**
  Bei allen Vorgängerversionen von CAM-8 konnte jede Zelle nur eine begrenzte Anzahl von Zellen in ihrer unmittelbaren Nachbarschaft „sehen“ und deren Zustände zum Update des eigenen Zustands benutzen. Bei CAM-8 wurde diese Konzept grundlegend verändert:
  Man griff die Idee auf, Daten durch den gesamten Raum „fließen“ zu lassen, indem ganze Datenfelder in beliebige Richtungen geshiftet werden können. Die möglichen Entfernungen für solche Shifts sind sehr variabel und können bis zur Größe eines Moduls reichen - d.h. in einem Arbeitstakt könnten die Zellen eines Moduls ihren Zustand bis zu den entsprechenden Zellen der benachbarten Module senden. Dies entspricht einer effektiven Erweiterung der Nachbarschaft auf viele Millionen Sites! Dabei erfolgt die Aktualisierung des Zustandes der Zellen und die Propagation der Zustände durch RAM-Zugriff alternierend. Durch die derartige Verarbeitung und Weitergabe der Daten in weiteren Takten verbreiten sich diese veränderlichen Datenfelder im gesamten CA-Raum. Dieses Konzept wird der

Idee gerecht, dass sich Zustände wie Gravitation, elektrische Ladung o.Ä. phyischer Teilchen unbegrenzt im Raum ausbreiten, sich jedoch mit der Entfernung vom Ausgangspunkt durch Wechselwirkung mit anderen Teilchen verändern.

5 Anwendung auf verschiedene Probleme

5.1 Partikelströmungen

Genannte Datenflüsse wurden verwendet, um Partikelströmungen in sogenannten *lattice gases* zu simulieren: Solche Partikelbewegungen sind als Hardware-Operationen auf einem niedrigen Level (und somit sehr effektiv) und in einem großen Bereich von Geschwindigkeiten möglich. Dies wurde auch bei verschiedenen geophysikalischen Simulationen verwendet. Solche sind i.d.R. sehr komplex und daher auf klassischem Wege mittels Differentialgleichungen nur schwer zu modellieren. Die Modellierung mit zellulären Automaten hingegen ist einfacher: Aufgrund der Tatsache, dass ein dreidimensionaler zellulärer Automat schon von Hause aus die räumliche Struktur erfasst, verbleibt fast nur die Aufgabe, Interaktion der Teilchen im Sinne von lokalen Interaktionen zu annotieren. Dies wesentlich intuitiver, da man im Grunde nur die realen physikalischen Regeln diskretisiert und programmiert. Mit diesem Werkzeug wurden nun also verschiedene Modelle erzeugt, u.a. Strömungen, in denen auf Grund lokal fixierter Hindernisse Wirbel und sogenannte von-Karman-Straßen auftreten. Auch einfache Modelle der Rayleigh-Bénard-Konvektion von Partikeln zwischen zwei verschieden warmen Platten unter Einwirkung von Gravitation wurden erstellt und simuliert.

5.2 statistische Physik

- thermalized annealing:
  Hierbei wird eine kontrollierte Senkung der Temperatur eines Gases oder einer Flüssigkeit simuliert. Dabei kann bei geeigneten Modellen eine Art Kristallbildung beobachtet werden, welche jener realer Materie nicht unähnlich ist.

- Aggregation diffundierender Teilchen:
  Auch hierbei beschäftigt man sich mit Kristallen, genauer mit Kristallbildung. Die Regeln für einen solches Modell können denkbar einfach sein:
  `Es wird ein endlicher Raum vorgegeben, der viele bewegliche (diffundierende) Teilchen beinhaltet und in welchem zentral ein einzelnes Teilchen fixiert ist. Jedes diffundierende Teilchen, welches ein fixes Teilchen berührt wird fix.`
  Dieses einfache Modell reicht, um ein komplexes Gebilde ähnlich einem Kristalles entstehen zu lassen.

- Simulation von Polymeren:
  Hierbei wird simuliert, wie Polymere (als Ketten von Zellen mit bestimmten Zuständen) im Raum von einem Punkt hoher Konzentration ausgehend diffundieren. Dabei

wurde versucht Eigenschaften wie Erhaltung (Polymere bleiben in ihrer Kettenstruktur gleich, Anzahl der Polymere ist konstant) und Verbot von Durchdringung (an einem Raumpunkt kann stets nur ein Polymerteil sein) umzusetzen.

- andere Methoden, wie zum Beispiel eine Art simulated annealing, die zum Glätten von Bildern und zur Bearbeitung von Texturen Verwendung finden könnte.

5.3 Grafik-Ein- und -Ausgabe

Bei all diesen Methoden wurde die Tatsache ausgenutzt, dass die Systemdynamik durch lokale Eventcounter einfach ausgewertet werden kann. Diese Eventcounter machen im Prinzip nichts anderes, als für definierte Zellen die Anzahl von Durchgängen bestimmter Zustände zu zählen. Statistisch lässt sich somit generell jede beliebige Größe des simulierten Systems, wie z.B. Temperatur, Dichte, Magnetisierung, Druck, Energiedichte usw. erfassen.
Natürlich kann kein Mensch etwas mit den ausgelesenen Datenfeldern anfangen, ohne dass sie in geeigneter Form aufbereitet werden. Deshalb können, wie schon erwähnt, durch den Master-Rechner die erfassten Daten der Eventcounter zu Bildern und Videos zusammengestellt werden.
Umgekehrt ist es möglich über ein gekoppeltes Video-Interface auch Daten von einer Video-Kamera aufzunehmen und in Echtzeit mittels CA-Regeln zu verarbeiten. So ist es etwa möglich, Bilder frei um beliebige Winkel zu rotieren. Für Bilder der Größe 512×512 sind dafür weniger als 10ms nötig, was für zeitgenössische Computer beachtlich schnell ist. Eine weitere Anwendung ist (dank der räumlichen Struktur des Computersystems) die Verarbeitung von 3D Bildern. So können von CAM8 z.B. 3D-Daten von Kernspinresonanz-Bildern in beliebigen Ebenen geschnitten, aber auch zu 2D-Bildern mit beliebigen Blickwinkeln gerendert werden. Durch zweimaliges Rendern jedes Bildes aus leicht versetzten Augpunkten eröffnet sich die Möglichkeit, bewegte Stereobilder in Echtzeit zu erzeugen.

5.4 Spacetime Circuitry

Der Simulation logischer Gatter für bestimmte affine Transformationen entspringt die Idee, solche Transformationen zu beschleunigen, indem eigentlich zeitlich sequentiell abfolgende Operationen parallelisiert werden. Einfach gesagt, geschieht diese Parallelisierung durch Faltung zeilticher Abfolgen in eine zusätzliche Raumdimension. Dies ermöglicht ein effizientes Pipelining der Daten, so dass sequentiell „zufließende“ Daten unverzögert durch das System geleitet und verarbeitet werden können. Beispiel: statt t Updates eines n-dimensionalen Raumes verarbeitet man das Datenfeld in einem Update eines $n+1$-dimensionalen Raumes. Die Daten durchlaufen dann nach wie vor alle einzelnen Operationen der Transformation, nur dass diese eben von Zellen verschiedener Schichten nacheinander abgearbeitet werden, die von den Daten „durchströmt“ werden.

6 die Softwarearchitektur

Natürlich war es für ein solch komplexes Computerarchitektur unumgänglich, eine geeignete Softwarearchitektur aufzustellen. Wie immer bildet die Basis dieser Architektur die Hardware: Die Prozessoren können nur Bytecode ausführen, dieser stellt die unterste Software-Ebene dar. Im Grunde handelt es sich bei diesem low-level Code um die Schnittstelle von Soft- zu Hardware - einfache Treiber-Bibliotheken, die grundlegende Methoden für den Zugriff auf die Hardware zur Verfügung stellen.
Bei Cam-8 ist die Zahl der Schichten zur Programmier-Oberfläche sehr gering, diese liegt fast unmittelbar über der genannten Bytecode-Ebene. Zur Programmierung wurde eine einfache, rudimentäre Programmiersprache entwickelt, die es erleichtern soll, Regeln für den Zellulären Automaten aufzustellen. Ein interessantes Feature der Softwarearchitektur ist, dass für die Entwicklung des Treiber-Levels auch eine Art „Pseudo-Treiber“ entwickelt wurde, der im Grunde nur eine Schnittstelle für die Ausführung eines Cam-8-Programmes auf einem „herkömmlichen“ Rechner darstellt. Es war damit z.B. möglich, die Cam-8-Programme auf einer Sun SPARCstation auszuführen, und damit das CAM-8-System zu *emulieren*. Diese Ausführbarkeit ohne die Anwendung auch nur eines CAM-8-Moduls wird mit „Zero-Module Scalability“ bezeichnet.
Die Vorteile liegen auf der Hand: Forscherteams können noch vor Erhalt eines CAM-8-Systems Programme für dieses implementieren, testen und anschließend praktisch aufwandsfrei portieren.
Natürlich stellte die Entwicklung der Treiber und der Programmierumgebung eine schwere Herausforderung dar:
Es mussten Softwarebibliotheken für die Entwicklung von Programmen zusammengestellt werden, man musste Compiler bauen, die Regeln für zelluläre Automaten automatisch an die gegebene Hardware anpassen. Des Weiteren arbeitete man an der Entwickung von Compilern, die die sequentiellen Operationen von Datentransformationen automatisch zur Pipelines umgestalten sollten, um spacetime cirquitry zu ermöglichen. Für die Entwicklung und Validierung der CA-Programme mussten darüber hinaus High-Level-Debugger erschaffen werden, um den Vergleich des Systemverhaltens mit den Erwartungen zu vergleichen.
Und vielleicht die wichtigste Herausforderung neben der Bereitstellung der Softwarebasis:
für viele mathematische oder physikalische Probleme mussten erst einmal CA-Modelle aufgestellt werden, um sie überhaupt mit dem System verarbeitbar zu machen.

7 Entwicklung und Ende des Projektes[3]

Was ist aus dem Forschungsprojekt geworden? Bis zum August 2001 lässt sich eine stete Entwicklung der Software nachvollziehen, es wurden die Programmierumgebung und die Treiber verbessert und erweitert. Es gab immer wieder Kooperationen mit anderen Arbeitsgruppen, die verschiedene Modelle auf der Hardwareplattform simulierten. Leider fand das Unterfangen am 5. August 2001 ein jähes Ende:

„Sigh... After having our machine abused by some mindless jerks the disk crashed and everything since Nov of 1998 has been lost. This is of course a major setback. Work on CAM8 software has since come to a halt. It is uncertain when it can be resumed. Hopefully we can offer something new this year.“ Dies ist der letzte Eintrag aus dem „Weblog“ des CAM8-Teams - die Arbeit an CAM8 wurde aufgrund der überholten Hardware und dem fehlenden Backup der Software nicht wieder aufgenommen.

8 Zusammenfassung

Die Ideen, die hinter der Entwicklung von CAM8 stecken, dürfen - unter Beachtung der Zeit ihrer Umsetzung - als revolutionär angesehen werden. Obwohl das Konzept zellulärer Automaten schon so alt ist, wie die Informatik selbst[4], war CAM8 in den 1990er Jahren einer der frühen erfolgversprechenden Versuche die Funktionsweise eines zellulären Automaten in Hardware zu gießen. Die Tatsache, dass er schon 7 Vorgängerversionen hatte, zeugt von dem Engagement seines Entwicklerteams bei der Entwicklung eines solchen parallelen Systems. Die Ergebnisse und Leistungen die erreicht wurden zeugen vom hohen Potential, welches im System steckt und mit viel Mühe ausgeschöpft wurde. Im vergleich mit zeitgleich gebauten Home- aber auch Supercomputern wurden bemerkenswerte Rechenkapazitäten von zellulären Automaten offengelegt und neue Wege gefunden Daten parallel zu verarbeiten. Aber es zeigt sich auch ein grundlegendes Problem einer solchen Architektur: offenbar kann die erreichte Leistung den Entwicklungsaufwand im Angesicht der rasant wachsenden Kapazitäten von „normalen“ Rechnern nicht wett machen - als durch den Festplattenabsturz das Team auf den Stand zwei Jahre zuvor zurückgesetzt wurde, hatten die herkömmlichen und alternativen Systeme die Kapazitäten des CAM-8-System längst ein- bzw. überholt.

9 Quellen

1. Norman Margolus: „CAM-8: a computer architecture based on cellular automata“ (1993)

2. Roman Frigg und Stephan Hartmann: „Models in Science“ (http://plato.stanford.edu/entries/models-science)

3. CAM8: a Parallel, Uniform, Scalable Architecture for Cellular Automata Experimentation - die Website zum Projekt: http://www.ai.mit.edu/projects/im/cam8

4. Robert A. Freitas Jr. and Ralph C. Merkle: „Kinematic Self-Replicating Machines“ (2004), http://www.molecularassembler.com/KSRM/2.1.3.htm