Parallel Automata Processing Review

Jose Chavez·

·Department of Computer Science
University of Engeeniering and Technology

Operating Systems, 2019

Outline

1. Main Concepts
   - Finite State Machines (FSM)
   - The Micron Automata Processor (AP)
2. Problems
3. Solution
   - Theorical Solution
   - Parellel FSM
4. Overall Speedup

Finite State Machines I

FSM definition

- Formally described by a quintuple $\langle$ Q, $\sum$, δ , q_0 , F$\rangle$.
- Q is a set of states.
- $\sum$ is the input symbol alphabet.
- $\delta(Q, \alpha)$ is the transition function
- q_0 is the set of start states.
- F is the set of reporting or accepting states.
-

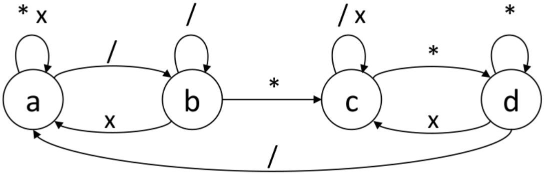
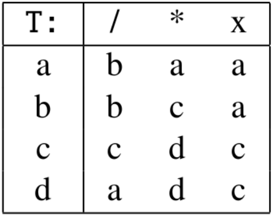


Figure: State representation(NFA)[2]

Finite State Machines II

FSM definition

- 

T:	/	*	x
a	b	a	a
b	b	c	a
c	c	d	c
d	a	d	c

Figure: Transition Table representation[2]

The Micron Automata Processor (AP) I

- Micron
-

The Micron Automata Processor (AP) II

- ”Microns 48-chip evaluation board scales this bandwidth to a ridiculous 38TB/s, which enables Automata to solve problems that traditional processors cannot.” -*Micron*

Problems

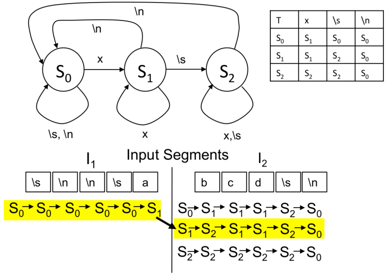
- These embarrassingly sequential applications (FSM) with irregular memory access patterns perform poorly on conventional von-Neumann architectures.
- FSM computation, especially Non-Determinstic Finite Automata (NFA) computation is inherently hard to speedup.
- Modern multi-core processors are limited by the number of transitions they can do per thread in a given cycle, limiting the number of patterns they can identify.

Theorical Solution I

Partitioning the input string into segments and processing these segments concurrently.
The problem with this approach is that starting states for each segment are unknown except the first segment.

1. Represent the NFA with a compact equal form, that is the AutomataNetwork Markup Language (ANML) NFA
2. We look up for destination states by the transition table and we convert them into active states for the next step.
3. We stablish the routing matrix an use it to store the transitions and the function.
4. Determine the set of states that are active in a particular cycle.

Parellel FSM I



T	x	\s	\n
S_0	S_1	S_0	S_0
S_1	S_1	S_2	S_0
S_2	S_2	S_2	S_0

$S_0 \to S_0 \to S_0 \to S_0 \to S_0 \to S_1$

$S_0 \to S_1 \to S_1 \to S_1 \to S_2 \to S_0$

$S_1 \to S_2 \to S_1 \to S_1 \to S_2 \to S_0$

$S_2 \to S_2 \to S_2 \to S_2 \to S_2 \to S_0$

Figure: A FSM example with enumeration[1]

Base Enumerative Technique

Stores a list of states at each step, insted of a single state.

Overall Speedup I

With these implementations we reach a **theoretical 2**x**(number of partitions)** over sequential baseline.
And in a **experimental** result we reached a significant speedup of **25.5**x on average compared again with sequential execution.

Reference I

Aron Subramaniyan
Parallel Automata Processor.
ISCA '17 June.

Todd Mytkowicz.
Data-Parallel Finite-State Machines
ASPLOS'14 March.