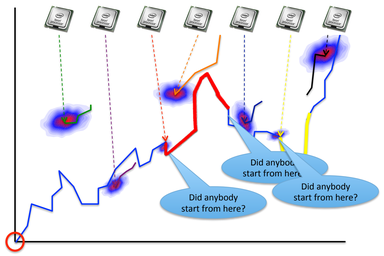
ASC: Automatically Scalable Computation — A Bridge between Worlds

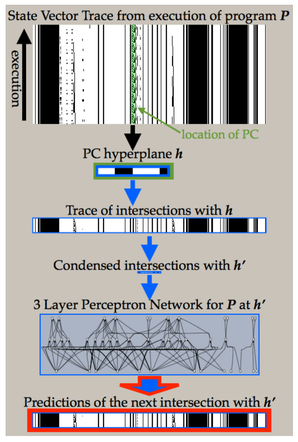
Sequential Execution
Simple Programming
Deterministic
Frequency and Space Limited

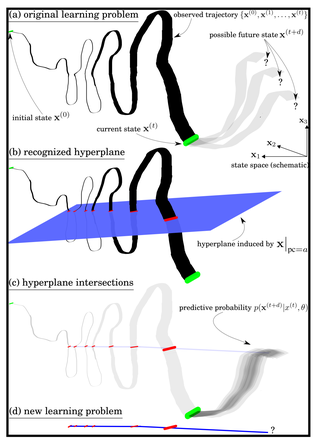
Scalable Technologies
Massive Multi-cores
Probabilistic
Neural Network Accelerators
Sub-threshold
Data-Centers

System

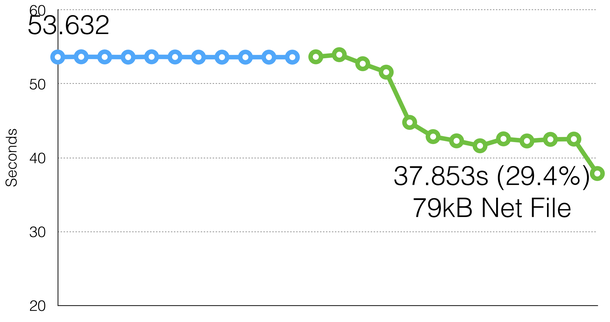
Execution : learning, recall, speculation and caching



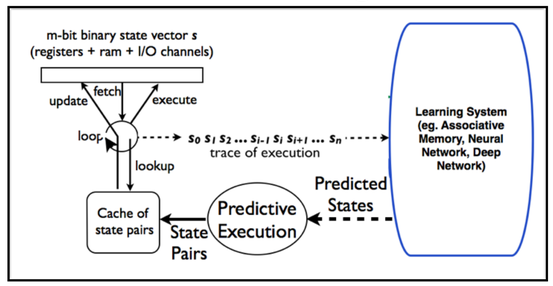




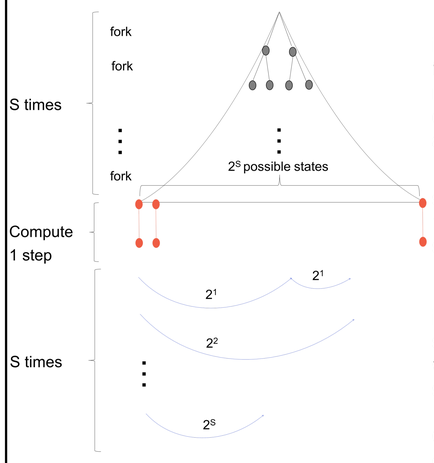

```
$ time ./factor > out
$ for ((i=0;i<10;i++)); do time ./factor > out; done
$ time ./asc factor > out
$ time ./asc -a 40013f -i 23068684 factor > out
$ for ((i=0;i<10;i++)); do
    time ./asc -a 40013f -i 23068684 factor.net factor > out; done
$ ./asc factor.* > out.train &  sleep 30; kill %1
./asc:main.c:233: batch training `factor.net' on `factor.train' ...
$ time ./asc -a 40013f -i 23068684 factor.net factor  > out
```



Theory



Chandra-Stockmeyer Parallelization (1976)



CONSTRUCTION

Phase 1: fork S times in parallel to obtain 2^S threads/sequential machine configurations/ addresses

Phase 2: simulate sequential machine for 1 step to obtain successor's address

Phase 3: for S iterations replace successor's address with successor's successor address

Conclusion: each state either has a pointer to a halting state or the sequential machine doesn't halt from this state

Result: PARALLELTIME(S) ⊆ DPACE(S^2)⊆PARALLELSPACE(S^2)

Extension of C-S Parallelization

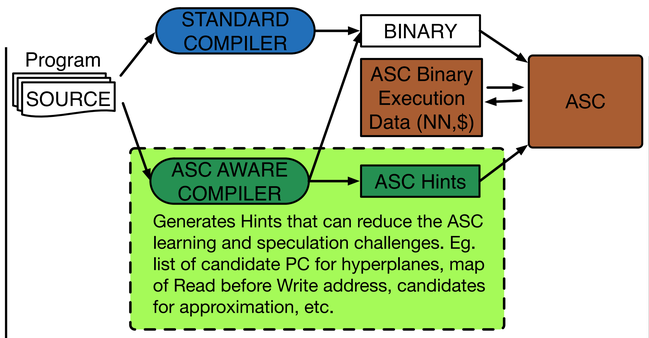
We assume the existence of a small, efficiently computable set S of states, where S is repeatedly visited during a program execution. We let t_{max} be the maximum time between two visits to states in S during the computation, and we let l_{max} be the maximum length of a configuration in S.

Generalizing the techniques of the C-S parallelization above, we obtain a similar relationship between the time and space used by our system when the assumptions above hold.

Result : If the function computing the configurations in S takes time linear in l_{max} then the computation is in PARALLELTIME ($O(\log(|S|)+(t_{max})(l_{max}))$.

Corollary: If the function computing the configurations in S takes constant time then the computation is in PARALLELTIME ($O(\log(|S|)+t_{max})$.

Compiler

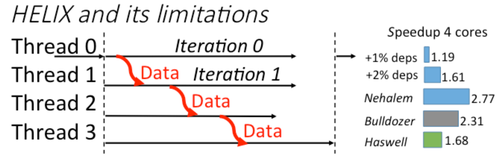


HELIX-UP: the Unleashed Parallelizer

- HELIX: automatic code parallelizer
- HELIX limitations due its conservativeness
  - Performance saturates at 4 cores
  - Inconsistent performance across architectures
  - Sensitive to dependence analysis accuracy
- HELIX-UP allows users to trade output distortion to remove HELIX's limitations
  - It reduces conservativeness to gain performance

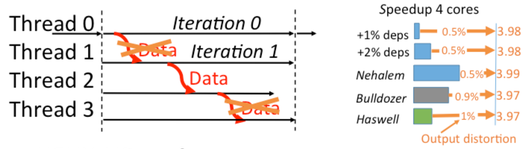
No output distortion *Baseline performance* — *Max output distortion* *Max performance*

HELIX and its limitations



HELIX performance:

- Lower than we would like
- Inconsistent across architectures
- Sensitive to dependence analysis accuracy



HELIX-UP performance:

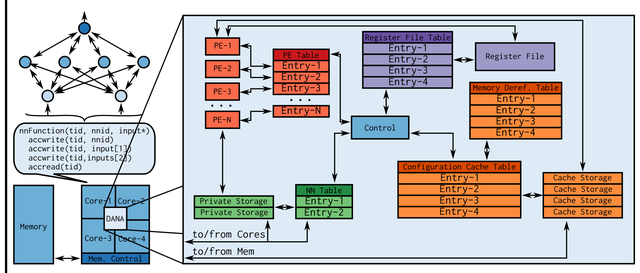
- Scale with number of cores
- Consistent across architectures
- Insensitive to dependence analysis accuracy

Compiler + runtime selectively remove unlikely dependences

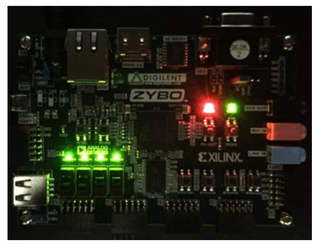
No output distortion *Baseline performance* — *Max output distortion* *Max performance*

Hardware

Dynamically Allocated Neural Network Accelerator (DANA)



FPGA DANA Prototype
Can store and evaluate software defined neural networks



ASC + DANA Prototype

To our knowledge first use of neuromorphic hardware to transparently accelerate a deterministic computation — binary

BOSTON UNIVERSITY
Jonathan Appavoo, Steve Homer, Ajay Joshi
Tommy Unger, Tomislav Petrovic, Schuyler Eldridge

Margo Seltzer, Ryan P. Adams, David Brooks
Amos Waterland, Yu (Emma) Wang
VERITAS HARVARD