Characterizing the Performance of Value Prediction using Statistical Simulations

CMPE 382/510 Project

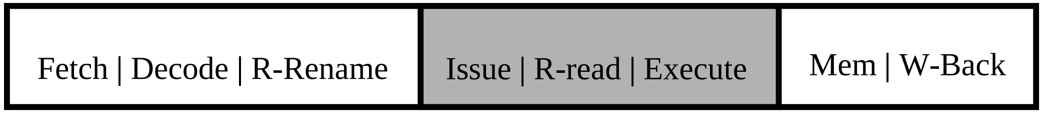
Peter Giese

9 December 2005

Outline

- **Objective**
- **Overview:**
  - Statistical Simulation
  - Value Prediction
- **Simulated Processor**
- **Results & Conclusions**
- **Further Research**

Value Prediction (VP)



Fetch | Decode | R-Rename

Issue | R-read | Execute

Mem | W-Back

Inst. Cache

Increase fetch bandwidth

Value Predication

Reduce data dependency and overcome latencies

Pre-fetch

Hide memory latency

- **VP is capable of overcoming true data dependencies**
- **Despite the advantage of VP as a speculative execution approach, there is still no commercial processor that employs this technique**
- **Why not …. what is the potential of VP?**

This report ….

- Characterizes the upper limit of VP performance
- Uses a statistical simulation approach
- Simulates a simple 5-stage pipelined processor with register forwarding and separate FUs for prediction validation

Factors Considered

- **Factors influencing VP performance:**

  i) Frequency of long latency instructions

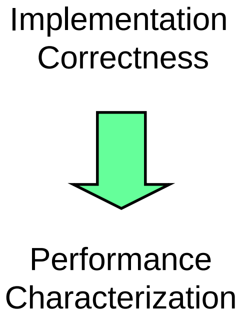
  ii) Level of value locality within instructions

  iii) Level of data dependency between instructions

  iv) Accuracy of value predictions

- **All of the above are studied using statistical models and random event generation**

Statistical Simulation


Implementation Correctness

Performance Characterization

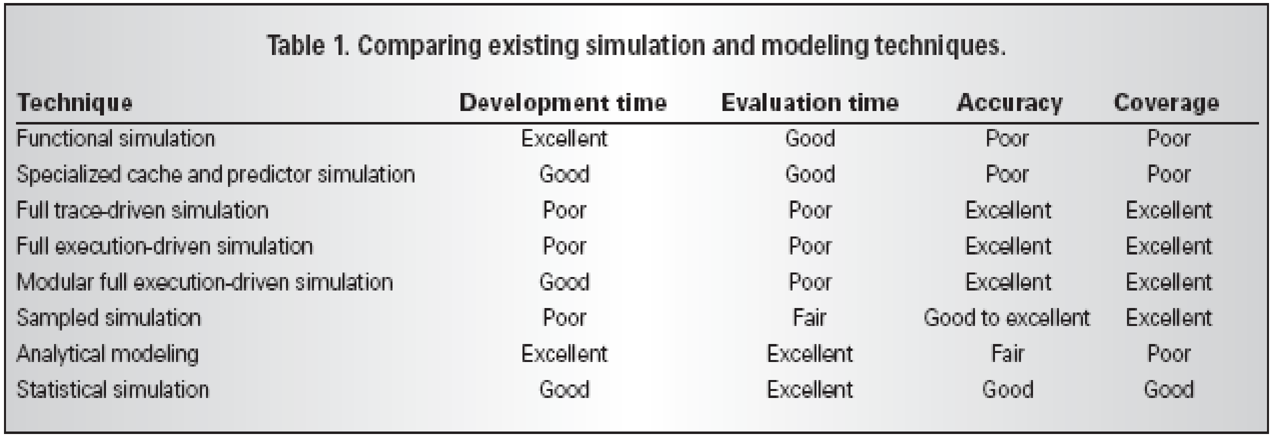


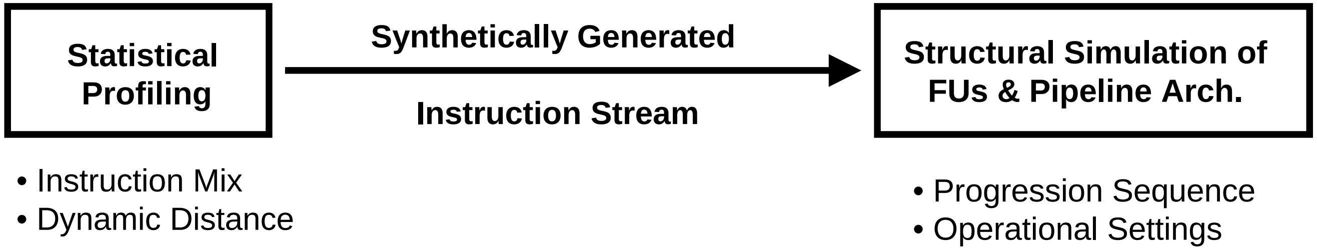
Table 1. Comparing existing simulation and modeling techniques.

Technique	Development time	Evaluation time	Accuracy	Coverage
Functional simulation	Excellent	Good	Poor	Poor
Specialized cache and predictor simulation	Good	Good	Poor	Poor
Full trace-driven simulation	Poor	Poor	Excellent	Excellent
Full execution-driven simulation	Poor	Poor	Excellent	Excellent
Modular full execution-driven simulation	Good	Poor	Excellent	Excellent
Sampled simulation	Poor	Fair	Good to excellent	Excellent
Analytical modeling	Excellent	Excellent	Fair	Poor
Statistical simulation	Good	Excellent	Good	Good

Source: L. Eeckhout, et al, 2003

- **Lies between detailed simulation & analytical models**
- **Provides a first-order performance estimate early in the design cycle**
- **Allows the design potential to be evaluated over a wide range of parameter values**

Synthetic Trace Approach



Statistical Profiling

- Instruction Mix
- Dynamic Distance

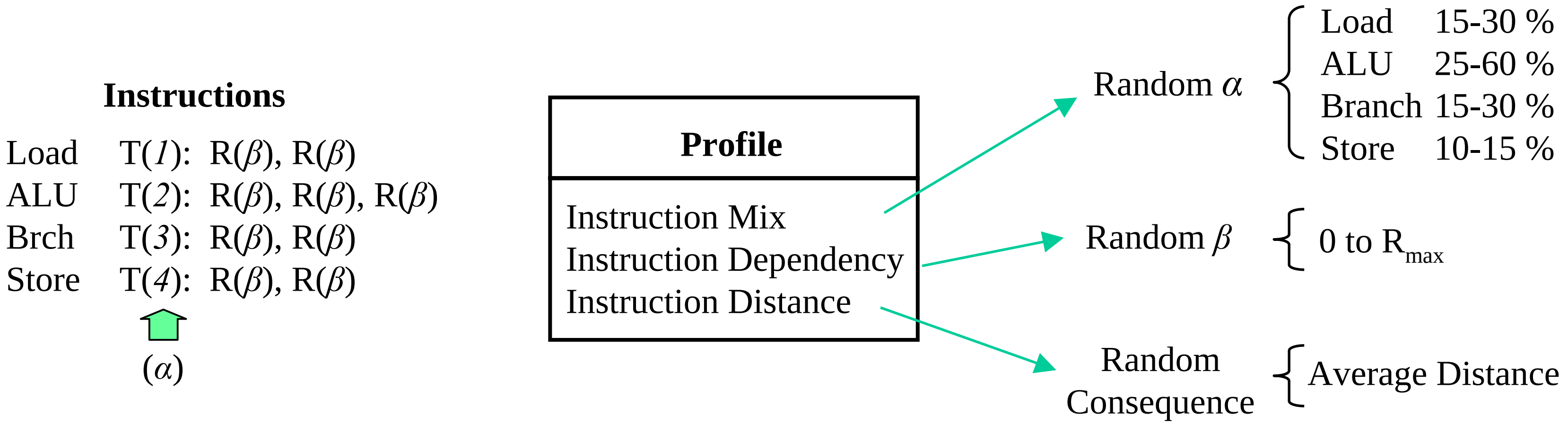
Synthetically Generated Instruction Stream

Structural Simulation of FUs & Pipeline Arch.

- Progression Sequence
- Operational Settings

- **Replaces actual code with synthetic instruction stream and synthetic functional behavior**
- **Accurate to within 4-8% of cycle-by-cycle simulations**

Instruction Profiles



Instructions

Load	T(*1*):	R(β), R(β)
ALU	T(*2*):	R(β), R(β), R(β)
Brch	T(*3*):	R(β), R(β)
Store	T(*4*):	R(β), R(β)

(α)

Profile

Instruction Mix
Instruction Dependency
Instruction Distance

Random α

Load	15-30 %
ALU	25-60 %
Branch	15-30 %
Store	10-15 %

Random β

0 to R_{max}

Random Consequence

Average Distance

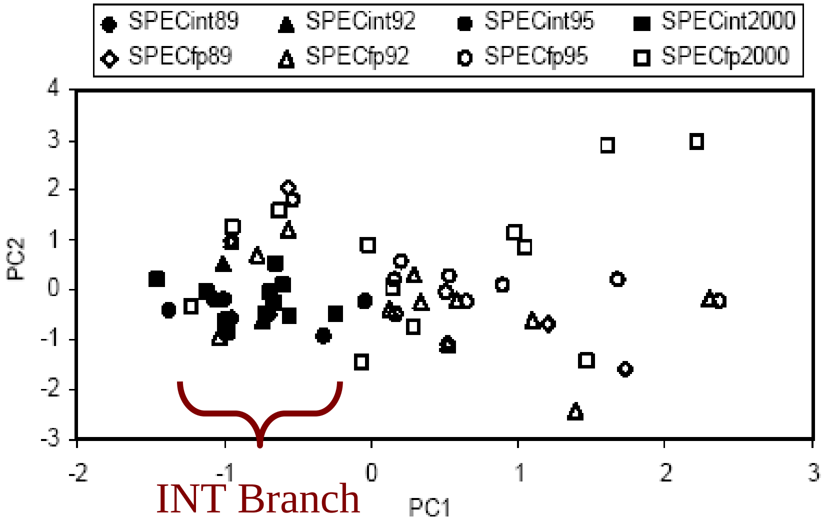
- **First-order approximations:**
  - Four generic instruction types
  - Random register dependencies (RAW)
  - Average block size

Instruction Distance

	Load	ALU	Brch	Store
Load			🟩	
ALU			🟩	
Brch	🟩	🟩	4-10	
Store				

Distance Matrix

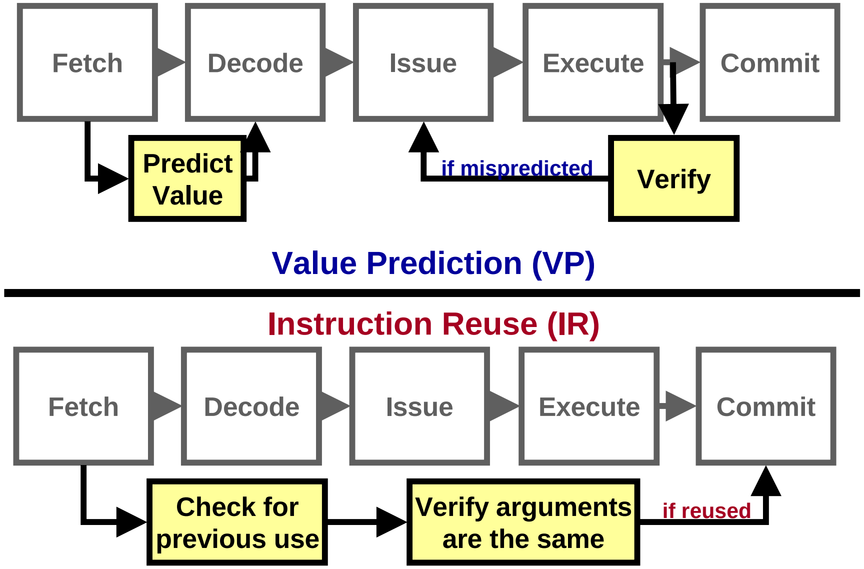
Branch PCA (Source: A. Phansalkar, et al, 2005)



INT Branch
Distance 4 –10 Inst.

- **Profiling distance between instructions is key to achieving accurate results**
  - Definition of distance matrix entries is an open research topic
- **This work … branch distance set to 6**

Different Approaches



Source:
A. Sodani & Sohi 1998

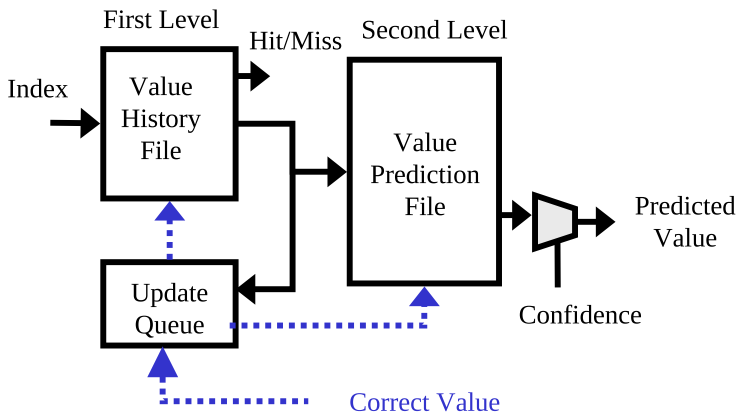
This work … focus is on Value Prediction

Predictions for Load and ALU instructions

Value Predictors

- **History-based**
  - Last value predication
  - Last-n value predictions
  - Finite Context Method (FCM)
- **Computational Predictors**
  - Detect stride, compute future values
- **Hybrids**
  - Combination of FCM and Computational

Predictor Accuracy



Two Stage Table Look-up
PC + Register Value + Stride + Last Five Values

- **Best predictors are Hybrid FCM**
  - Accuracy ~80% (90% max)
  - Requires large tables & complex lookup logic

Exploiting Value Locality

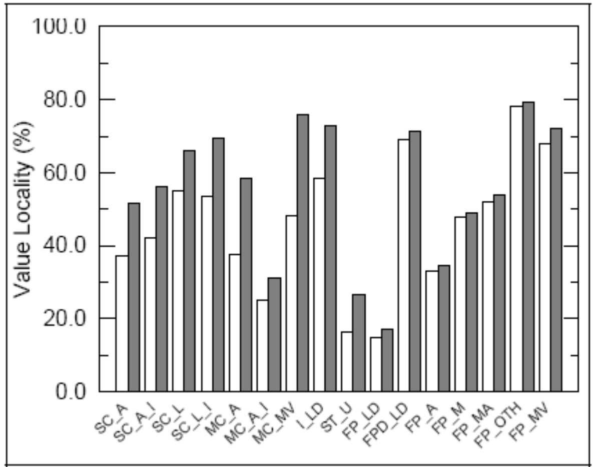


FIGURE 3. Register Value Locality by Instruction Type.

Source: M. Lipasti & J. Shen, 1996

Range of Locality Studied

Perfect: 100% ALU & 100% Load

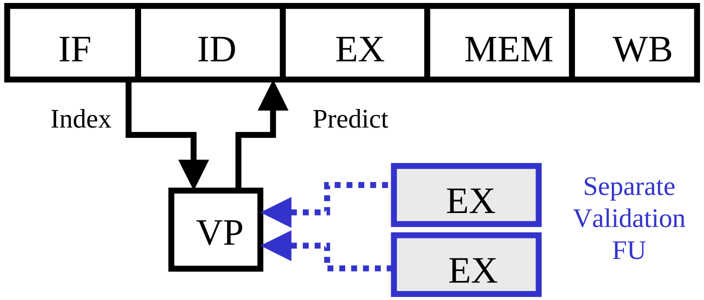
Upper limit: 60% ALU & 90% Load

Lower level: 20% ALU & 70% load

- **Locality varies per instruction**
  - Load writes (80%)
  - ALU writes (20-60%)

Simulation: 5-Stage Pipeline

- Single Issue & forwarding
- Speculative Execution
- Rollback Window



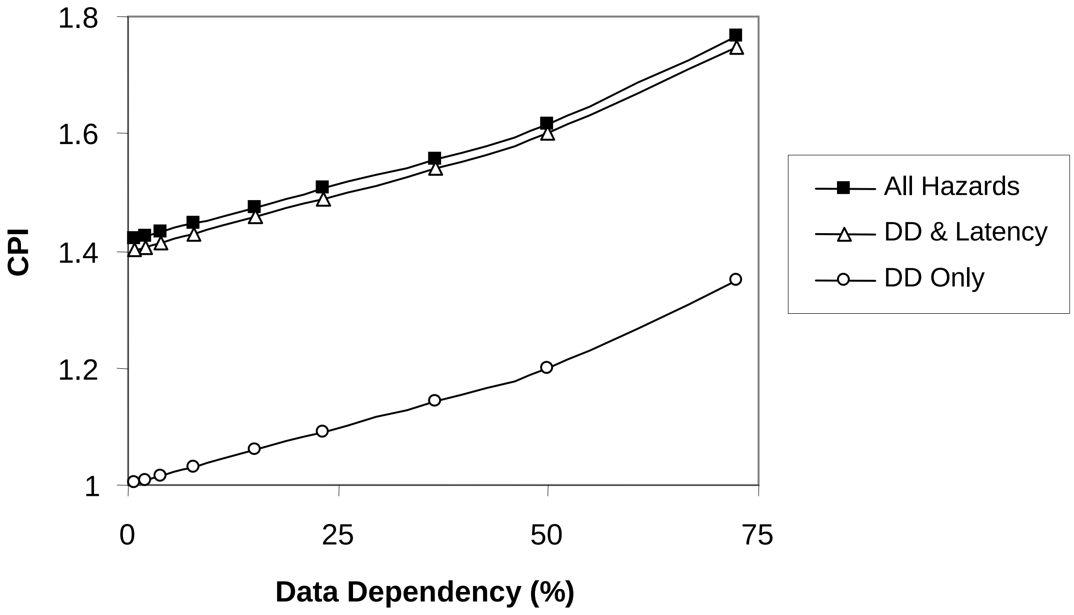
- **C-program used to simulate pipelined execution**
- **Idealized model:**
  - Predicted instructions are verified using separate FUs
  - No prediction delay or table size limitations
  - Rollback scenarios: 0 and 1 cycle delay

Simulation Parameters

- Target is a generic integer-based application
- 10 million synthetic instructions

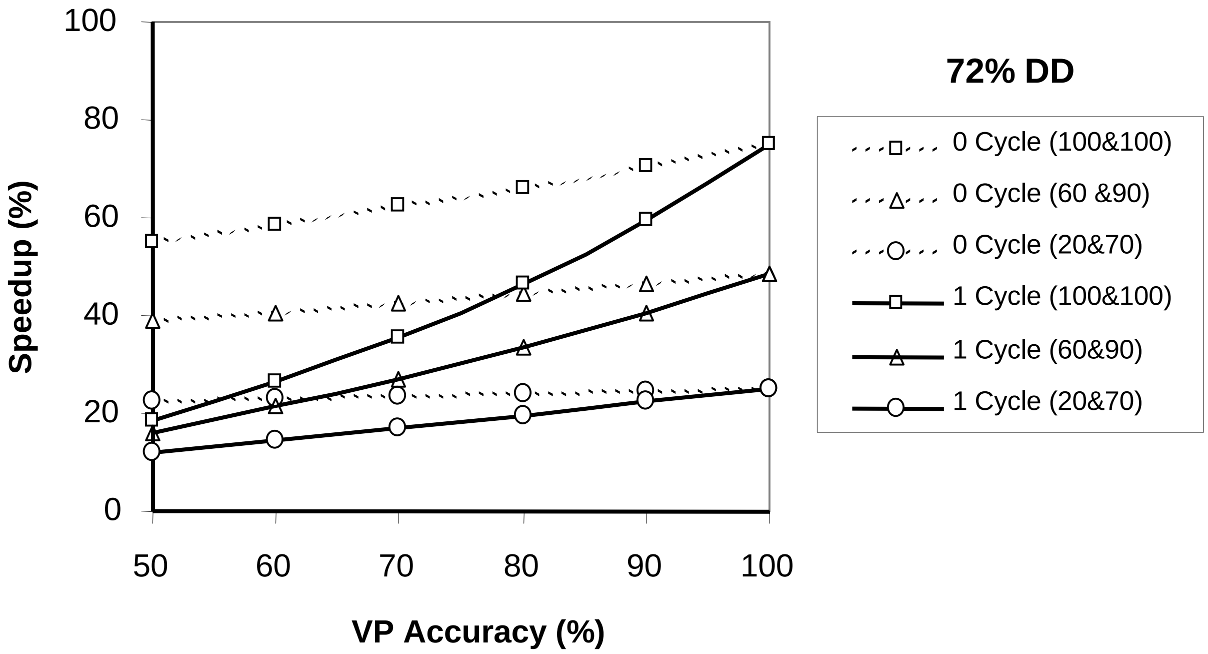
Profile	Characteristic
Inst. Occurrence	Random Distribution
Load	22%
ALU	50%
Branch	17%
Store	11%
ALU Latency	1 Cycle
	Mult/Div: 5 Cycles Aver.
	Latency Occurrence 15%
Inst. Dependency	Register RAW
Data Dependency	Variable:72% to 0%
Inst. Distance	6 Between Branches
Branch Behavior	Delay Slot
Data Stream Behavior	No miss 1 cycle
	L2/Mem: 10 Cycles Aver.
	Miss Rate 5%
Inst. Stream Behavior	Nil

Results: Baseline without VP



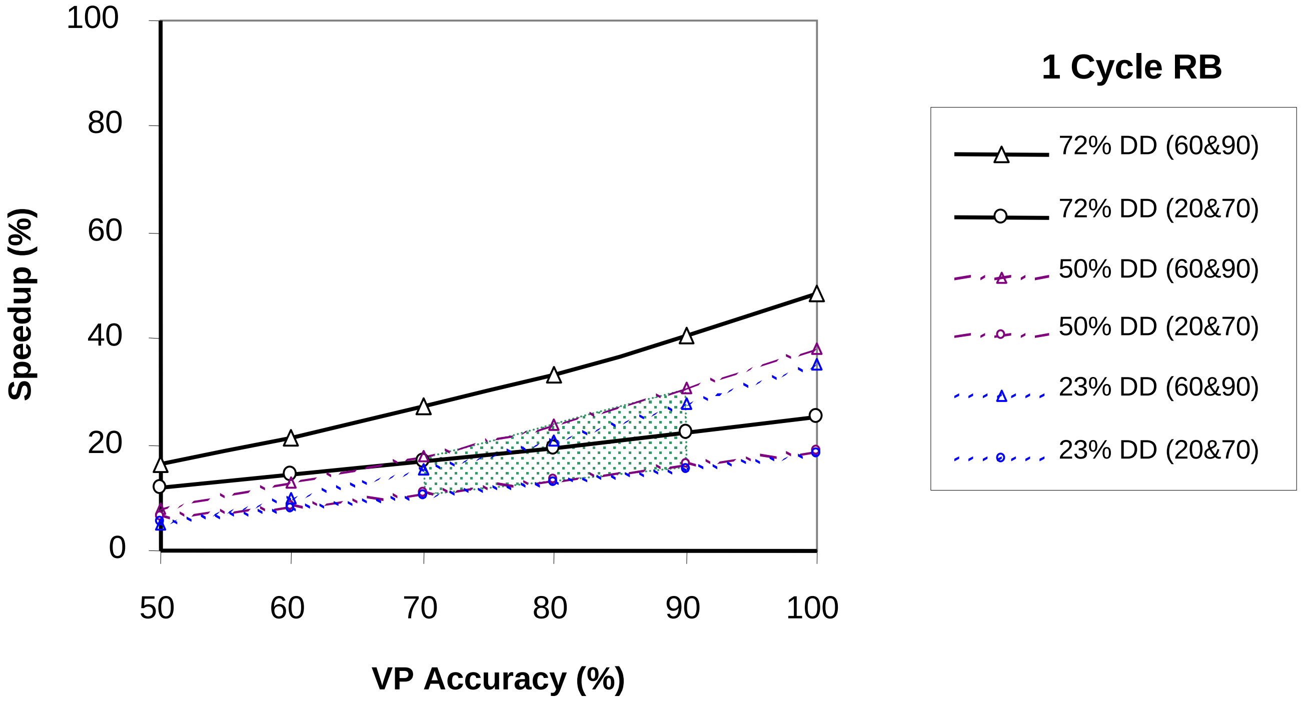
- **Data Dependency (DD) varied - 72% (max) to 0% (min)**
- **Constant control hazard - Excluded from VP experiments**

VP With &Without Rollback Penalty



- **Unrealistic: 100% locality and 0 cycle rollback**
- **Also, 72% DD is extreme**

VP versus DD versus Accuracy



VP Characterization

- Upper bound < 40 %
- 50% DD: 10% < $\text{VP}_{\text{Speedup}}$ < 30%

Other VP Results

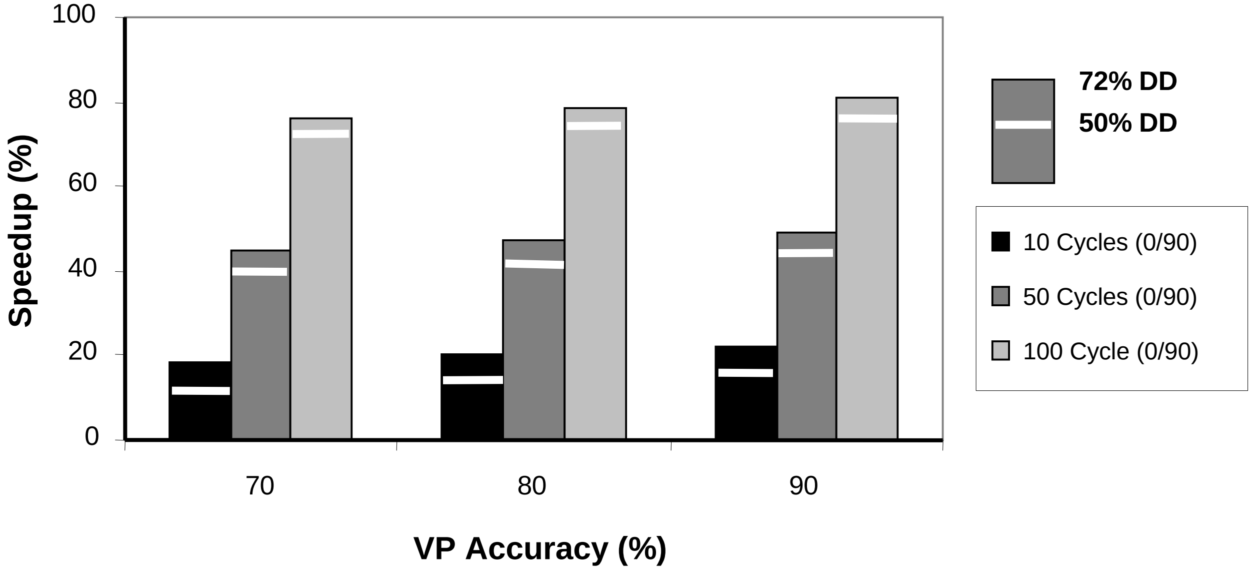
- **Other researches report VP performance improvements of 5-20%***
  - Across integer and floating-point applications
  - Across simplescalar and superscalar architectures
- **Why the difference (10-30%)?**
  - No limitations imposed by hardware and tables sizes
  - Near-ideal value locality and DD is modeled
  - No limitations imposed by control hazards
  - Integer versus floating-point specific applications

[*] J. Markovski & M. Gusev, 2004

VP for Load Instructions

- **Overcoming long memory latencies is an attractive scenario for the application of VP**
  - Overcome CPU/Memory performance gap
  - Reduces number of predictions (smaller tables)
- **Statistical simulation experiments also performed for larger memory latencies:**
  - L2/Mem average of 10, 50, 100 cycles

Load Only VP



VP Characterization

- Upper bound < 80 %
- 50% DD: 12% < $\mathrm{VP}_{\mathrm{Speedup}}$ < 76%

Summary

- **Predicated upper limits of VP performance:**
  - 10 to 30% for generic integer-based applications
  - <80% for load only VP with large latencies
- **Benchmark experiments required to confirm accuracy of statistical simulations results (+/-10%)**

Conclusion(s)

- **Upper bound of VP performance is not significant, even under near-ideal conditions**
  - Not enough to justify building generic VP into a general-purpose processors
- **Load only prediction may be applicable for specialized applications involving long memory latencies**

Further Research

- **Study integer versus floating point applications using specific distance matrix profiles**
- **Apply VP simulation approach to superscalar and speculative multithreaded processors**
- **Test a thesis:**
  - Upper limit of VP performance reported in this study is representative of behavior found in other pipeline architectures
  - Upper bound of VP performance will not very significantly in superscalar and multithreaded processors

Thank You