SC26
Chicago, IL | hpc unites.

Temporal Modeling of Soft Error Propagation Using Dynamic Bayesian Networks

Ahmer Jamil

Md Hasanur Rahman · Bingzhe Li · Xiaoyi Lu · Guanpeng Li

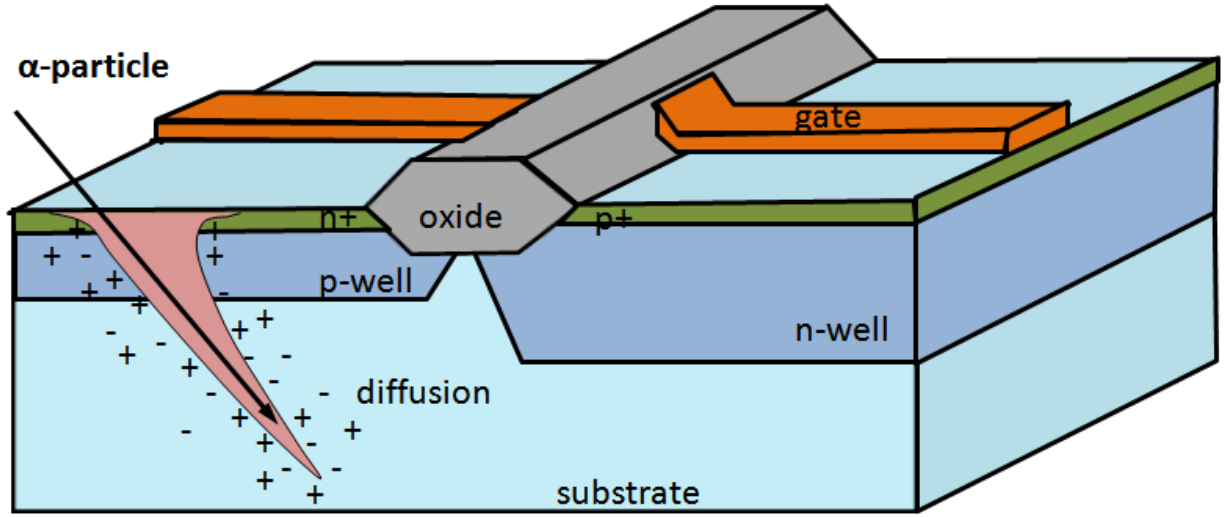
UF | UNIVERSITY of FLORIDA

THE UNIVERSITY OF TEXAS AT DALLAS

SC26 · Chicago, Illinois

From a particle strike to a bit flip

A charge disturbance can change a stored or computed bit.



Particle interaction in a sensitive device

One computed value

24 0 0 0 1 1 0 0 0

152 1 0 0 1 1 0 0 0

Illustrative unsigned 8-bit value

Hardware event → Activated bit error → **What does the program do next?**

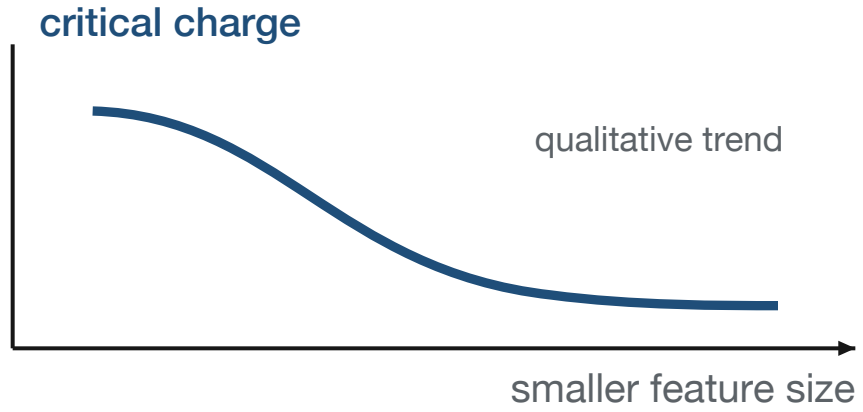
Hydra: our model of error propagation
Estimates the risk of a silent wrong answer.

Physical mechanism: Johnston, NASA/JPL | Illustration: MPS Lab, Arizona State University

Why the problem remains relevant

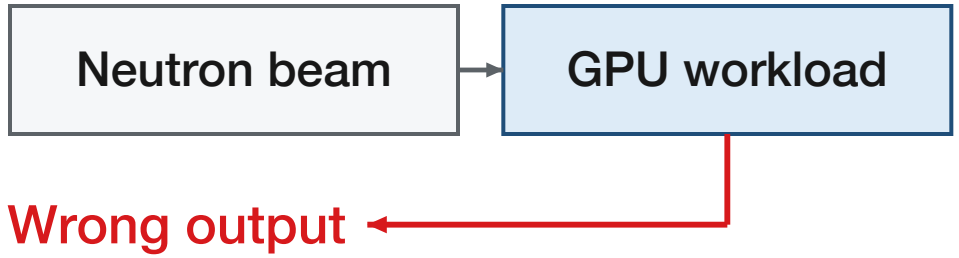
Device susceptibility

Smaller SRAM cells generally need less charge to upset a bit. [1]



Application consequences

GPU neutron-beam experiments measured silent output errors. [2]



Controlled exposure; not a field estimate

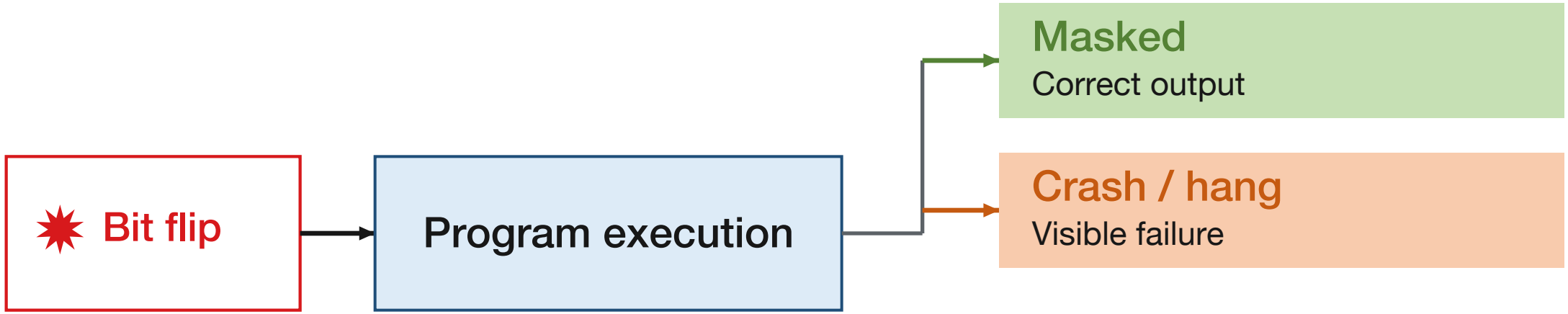
Hardware susceptibility alone does not tell us the SDC risk.

The instructions that use the corrupted value determine the outcome.

[1] D. Kobayashi, IEEE TNS 68(2), 2021, Fig. 4 | [2] F. F. dos Santos et al., IPDPS 2021

What follows a transient bit flip?

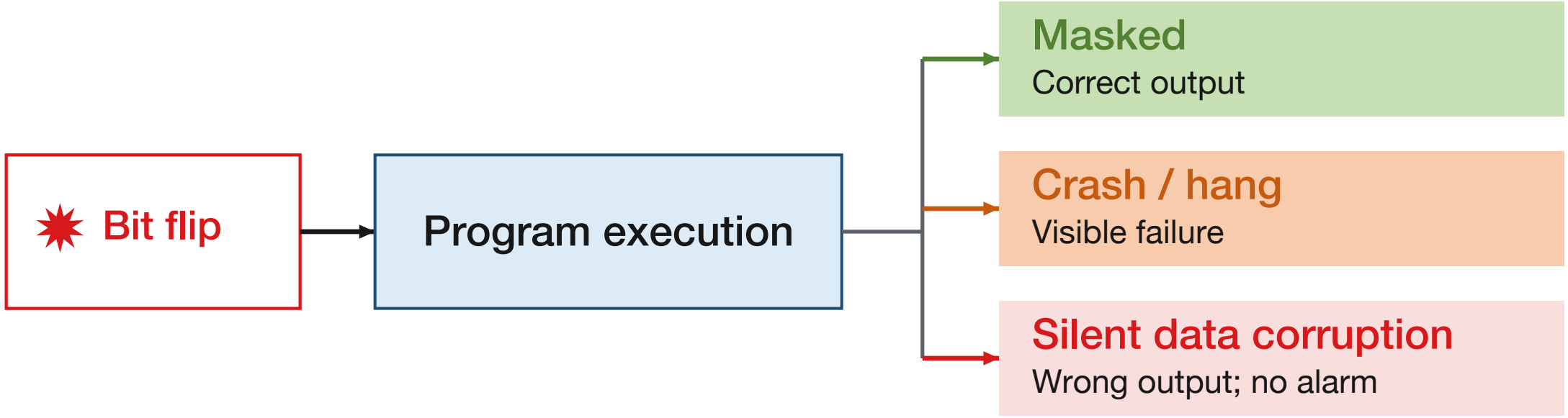
The same fault model can produce very different outcomes.



> **A corrupted value may be masked, or fail visibly.**
>
> But neither outcome explains the silent case.

What follows a transient bit flip?

The same fault model can produce very different outcomes.



A production example Wrong file size → Rows skipped **Months to diagnose**

Meta case: a repeatable silicon defect, illustrating the consequence of silence.
Hydra focuses on transient faults in computed values.

SDC consequence example: Dixit et al., Silent Data Corruptions at Scale, 2021

A changed comparison can skip a store

A predicate controls whether memory is updated.

One comparison result

Consider a visit where c4 is true.

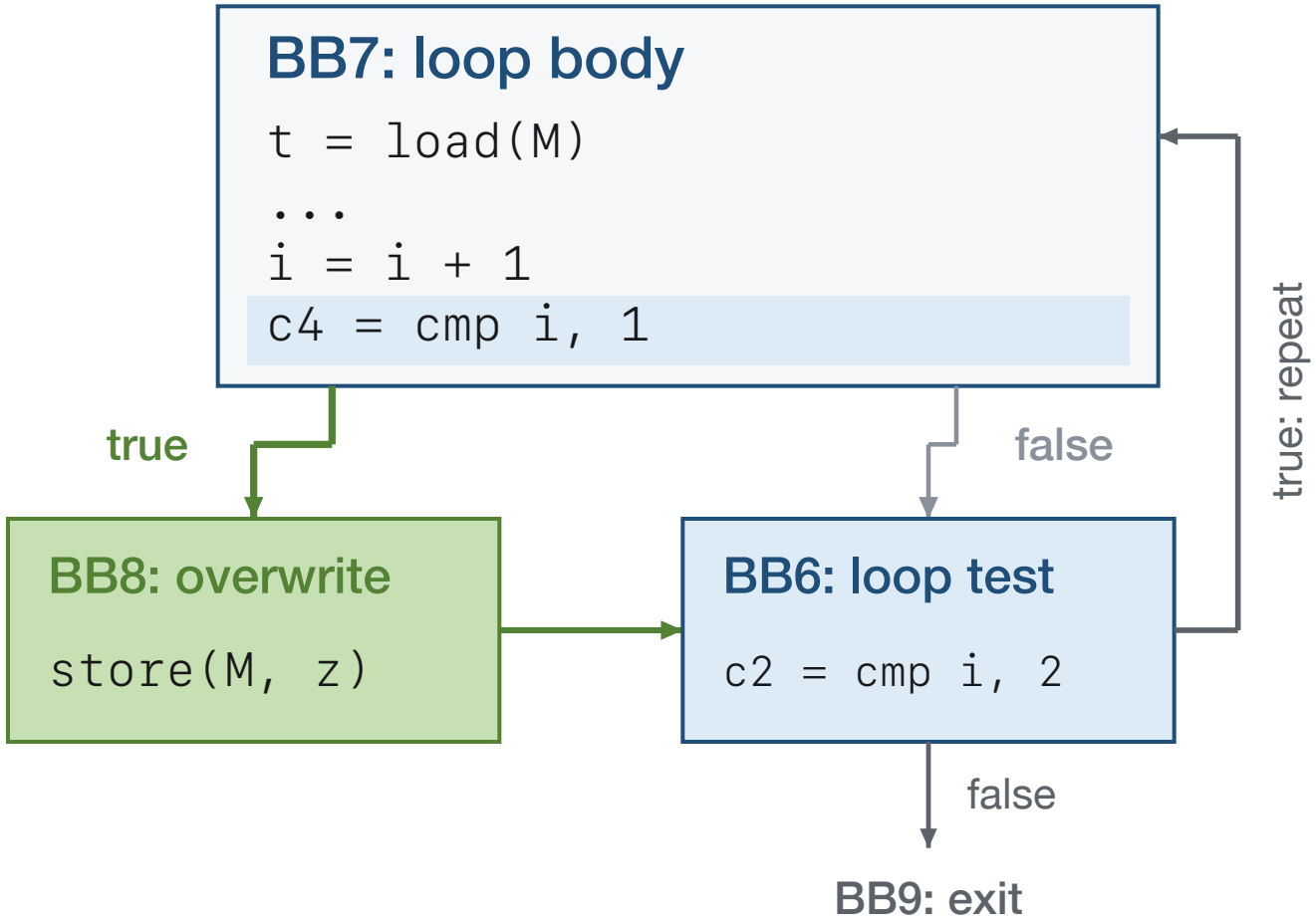
1

true

BB8 writes z to M.

Execution then reaches BB6.

The write happens before the next loop test.



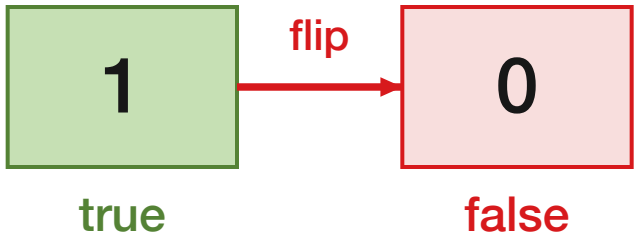
Selected instructions; arrows show control transfers

A changed comparison can skip a store

A predicate controls whether memory is updated.

One comparison result

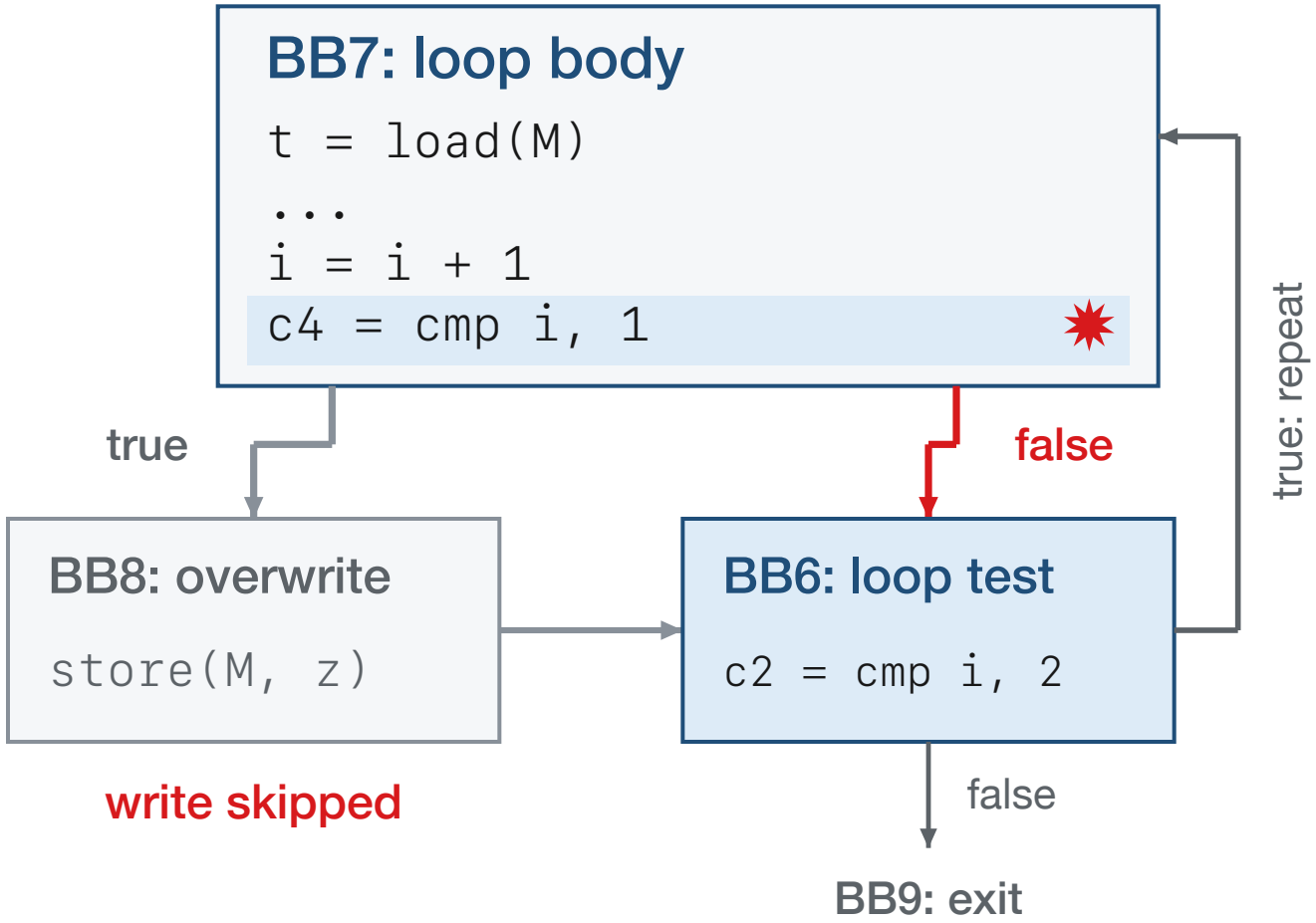
Consider a visit where c4 is true.



BB8's write is skipped.

M keeps its previous value.

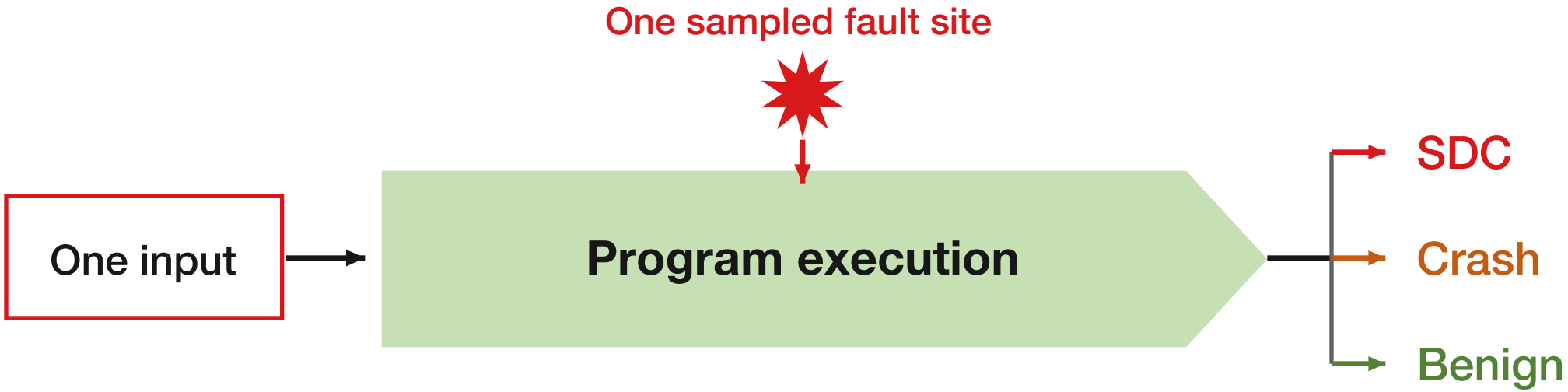
A later load may therefore read a different value.



Selected instructions; arrows show control transfers

Fault injection observes one outcome per run

Choose a dynamic instruction, flip a result bit, then finish the run.



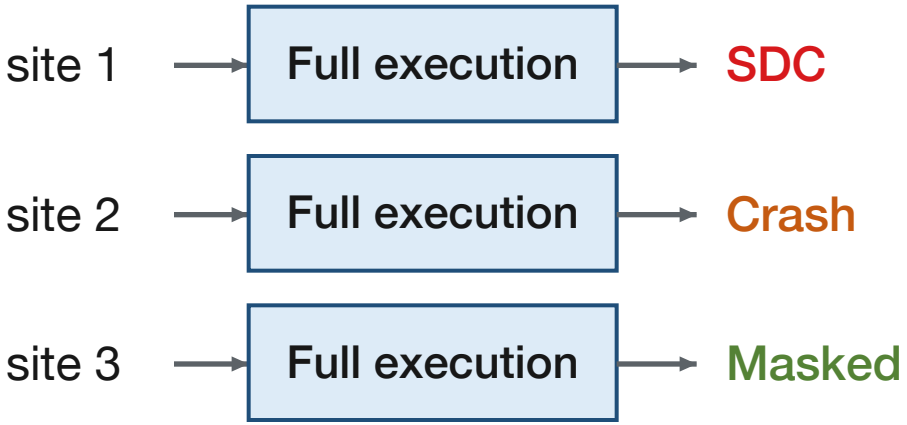
Repeat Across fault sites and bit positions

Calculate SDC probability from the fraction of trials with SDC

The gap: preserve when dependencies matter

Fault injection

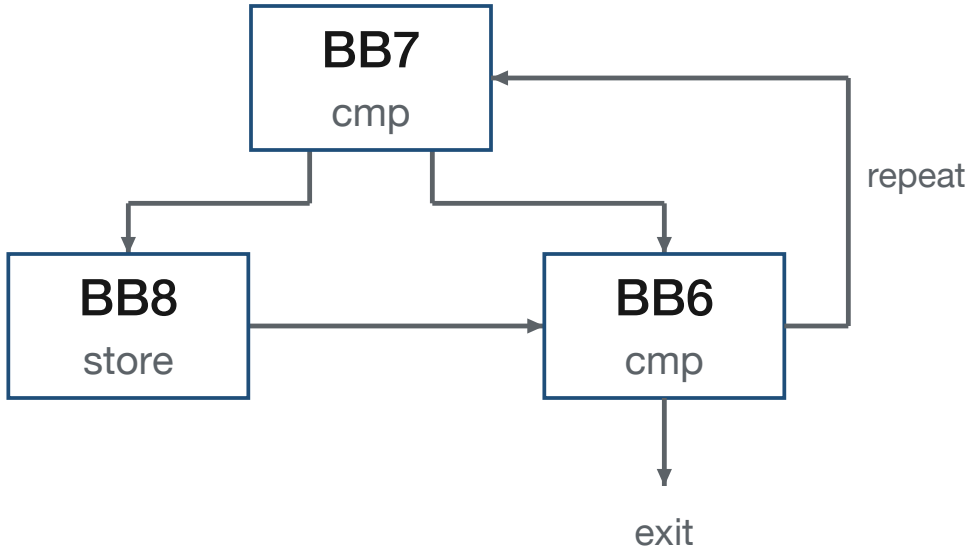
Observe repeated executions



Cost grows with repeated runs.

Prior analytical work

Estimate from program relationships



Static dependencies do not distinguish dynamic instances.

Changed decisions and changing memory sources make timing essential.

Prior analytical modeling: Li et al., Modeling Soft-Error Propagation in Programs, DSN 2018

The same load can have a different source

Follow the memory location M through one execution.



First BB7 instance

Its load reads the value written by BB5.

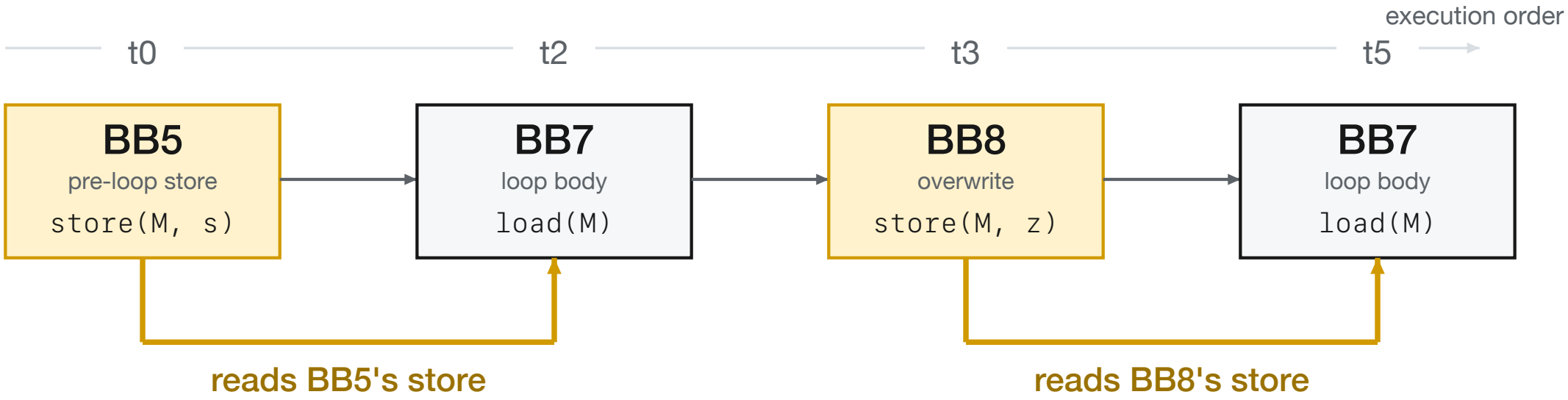
BB7 also contains a predicate that can enable BB8's store.

> **A corrupted store can affect a load much later.**

Selected dynamic instances; BB6 loop tests omitted. Gray arrows indicate execution order.

The same load can have a different source

Follow the memory location M through one execution.



After the overwrite

The later BB7 load no longer reads BB5 directly.

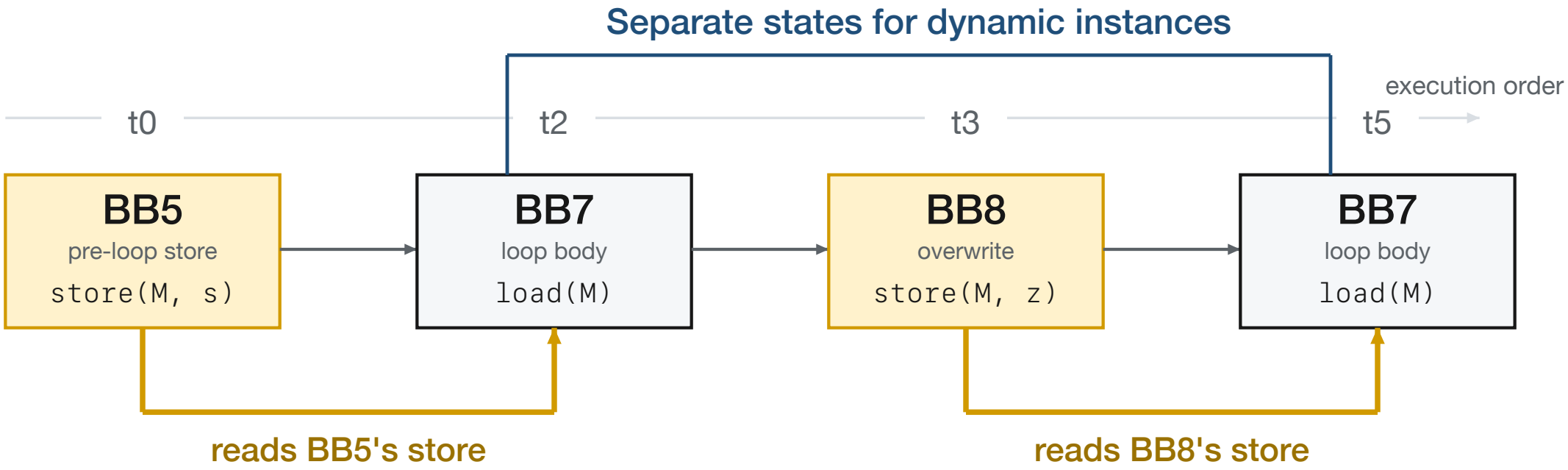
BB8 may still write a value influenced by the earlier error.

> **Repeated dynamic instances have different memory dependencies.**

Selected dynamic instances; BB6 loop tests omitted. Gray arrows indicate execution order.

What must the model remember?

Keep the execution history that changes the error's route.



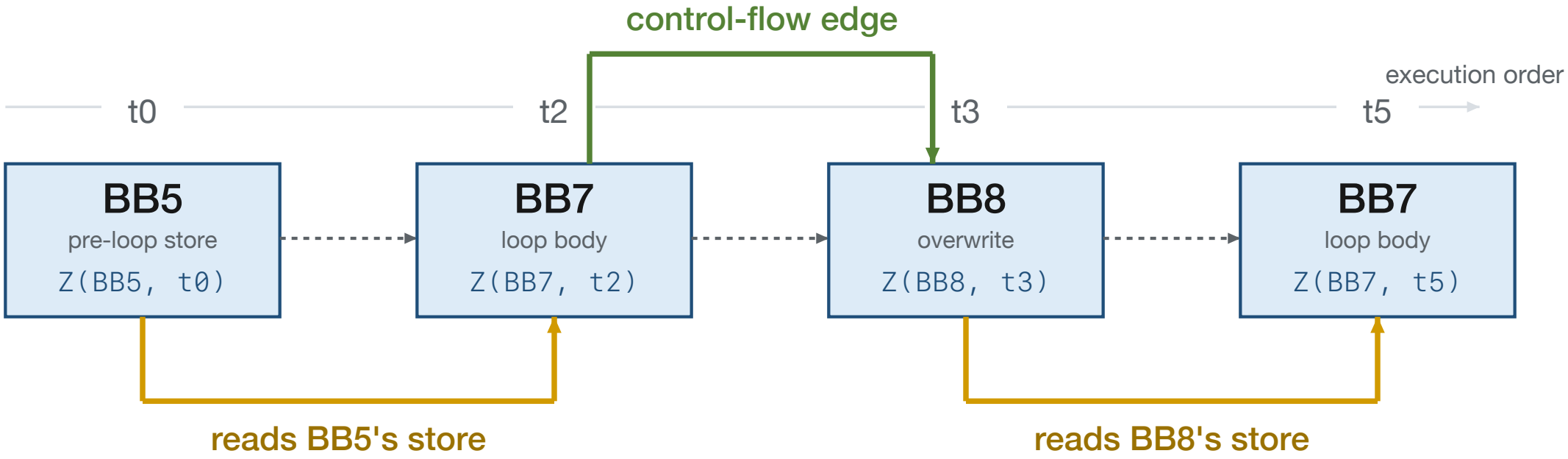
■ Dynamic instances ■ Control dependencies ■ Memory dependencies

Estimate SDC without enumerating every faulty execution.

Selected dynamic instances; BB6 loop tests omitted. Gray arrows indicate execution order.

Hydra models propagation over time

Dynamic Bayesian network (DBN): one node per dynamic basic-block instance.

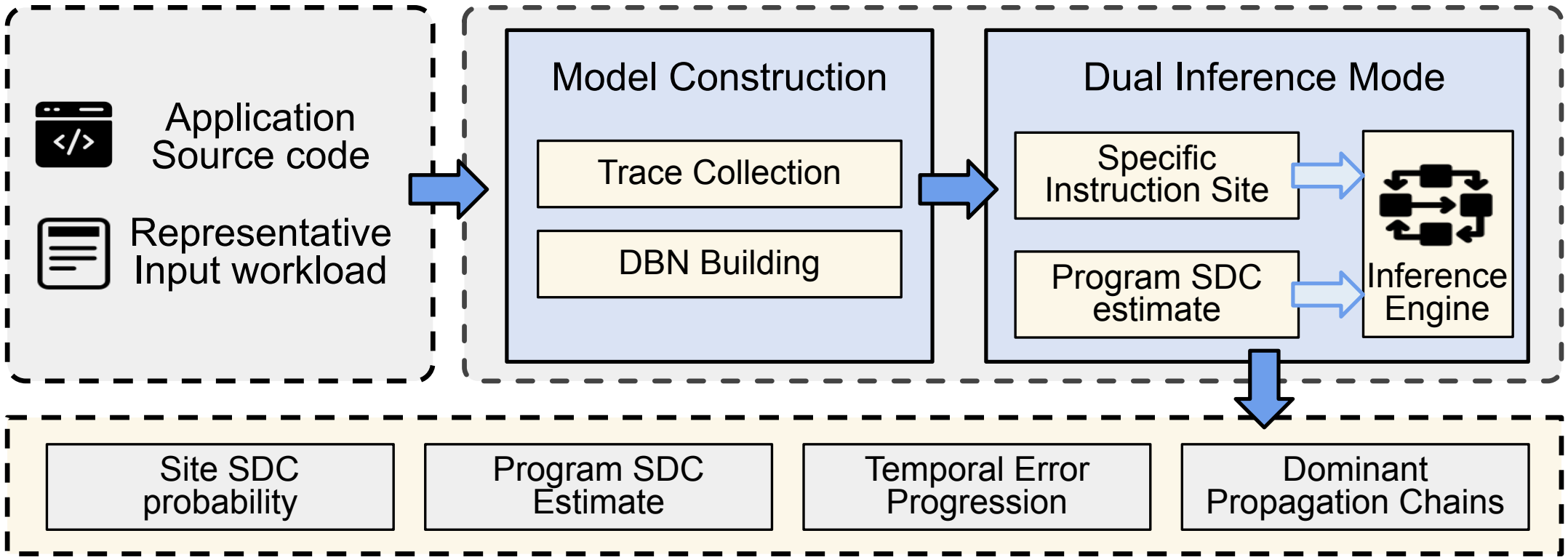


- DBN nodes
- Control-flow edges
- Memory-dependency edges

Estimate SDC without enumerating every faulty execution.

Selected dynamic instances; BB6 loop tests omitted. Gray arrows indicate execution order.

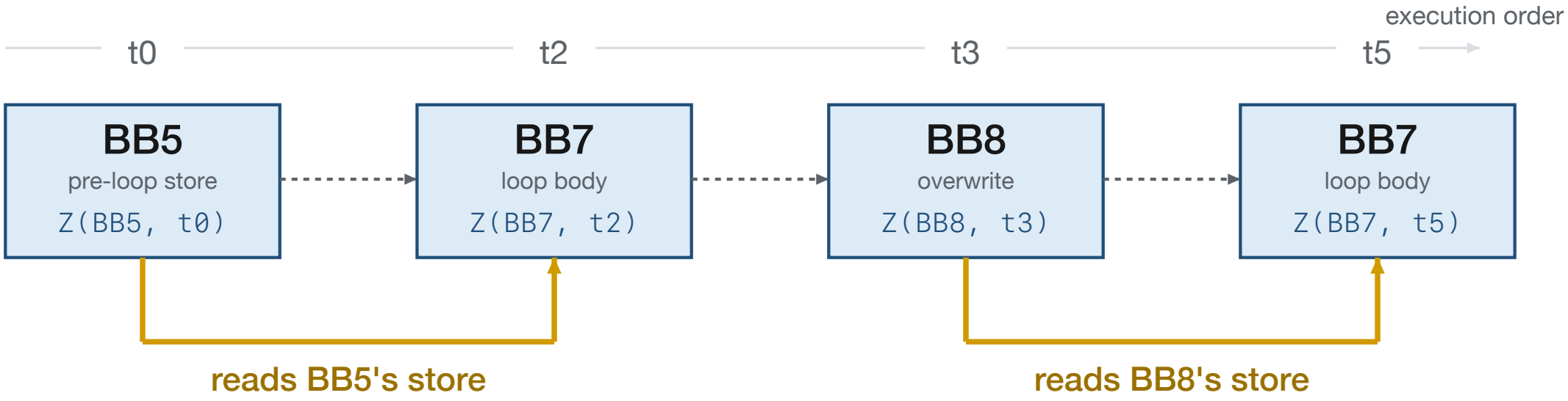
Hydra: model construction and inference



Build once for a program and input, then query many fault sites.

Each dynamic basic-block instance is a DBN node

Each node reuses a DDS template but has its own corruption probability.



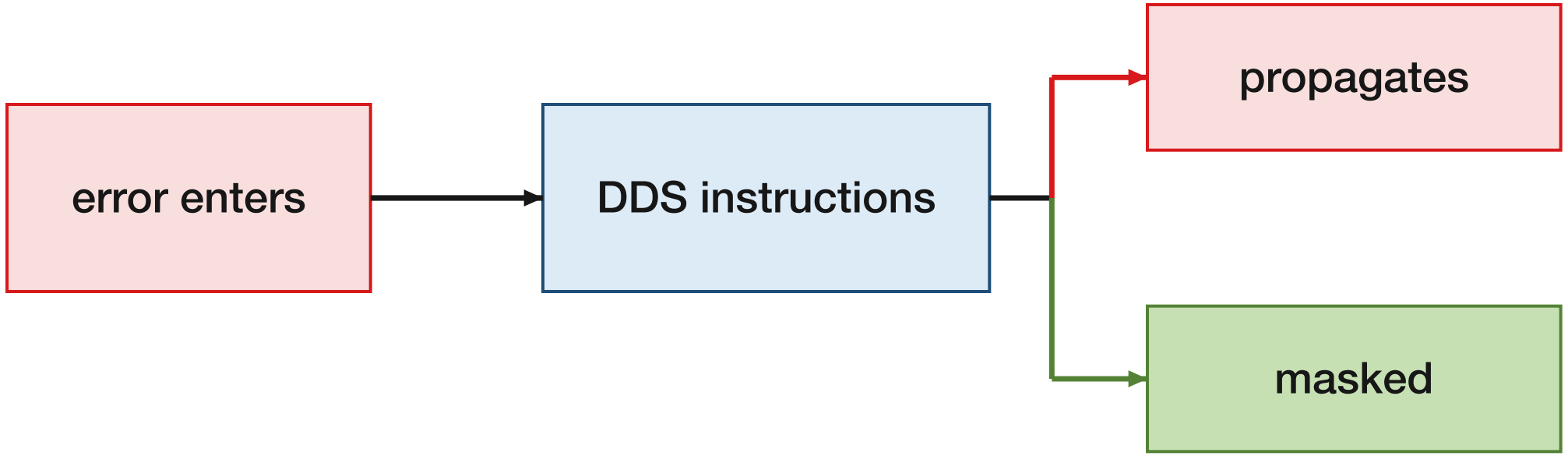
Same static code — Shared data-dependent sequence (DDS) template

Distinct DBN nodes — Z(BB7, t2) and Z(BB7, t5) have different parents.

Selected dynamic instances; BB6 loop tests omitted. Gray arrows indicate execution order.

Intra-block propagation: DDS and masking

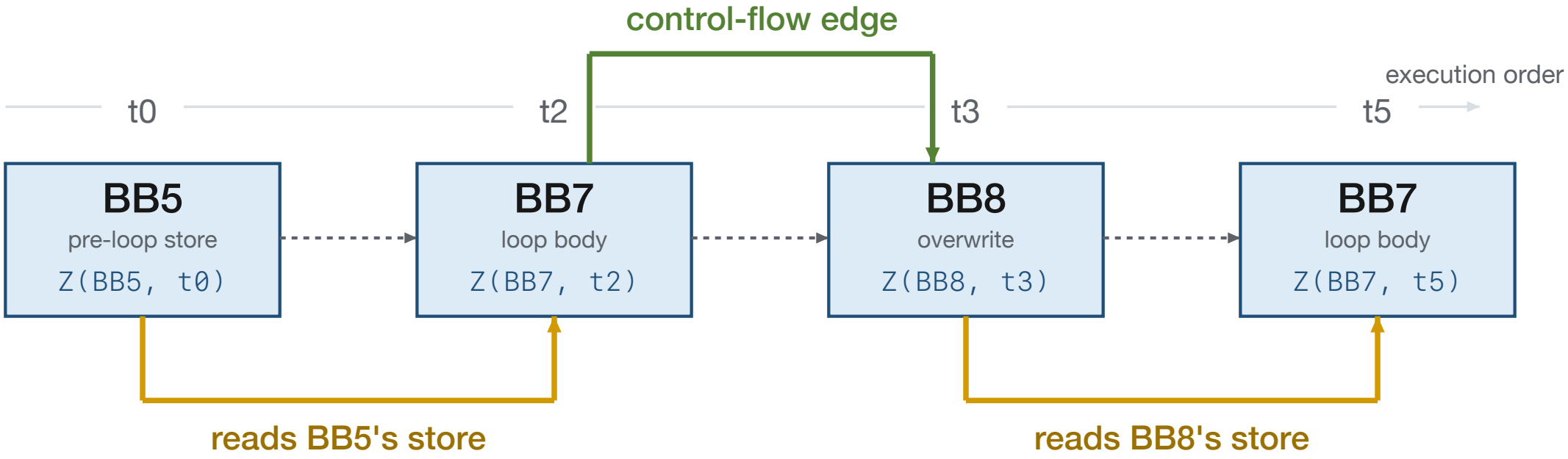
Intra-block propagation models masking within a DBN node.



Instruction-level propagation probabilities combine along each DDS.

Inter-block propagation: control-flow edges

BB7's c4 comparison can add or skip the store in BB8.



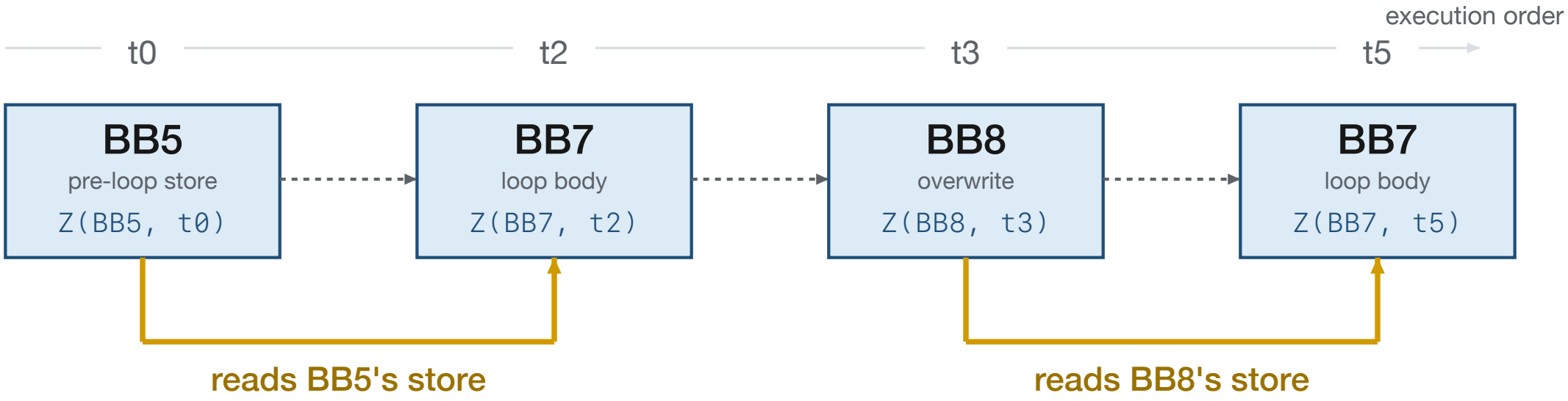
Control-flow edge weight How likely is the downstream store to be affected?

A profile-based estimate on the observed execution trace.

Selected dynamic instances; BB6 loop tests omitted. Gray arrows indicate execution order.

Inter-block propagation: memory-dependency edges

Connect each dynamic load to the most recent store at that address.



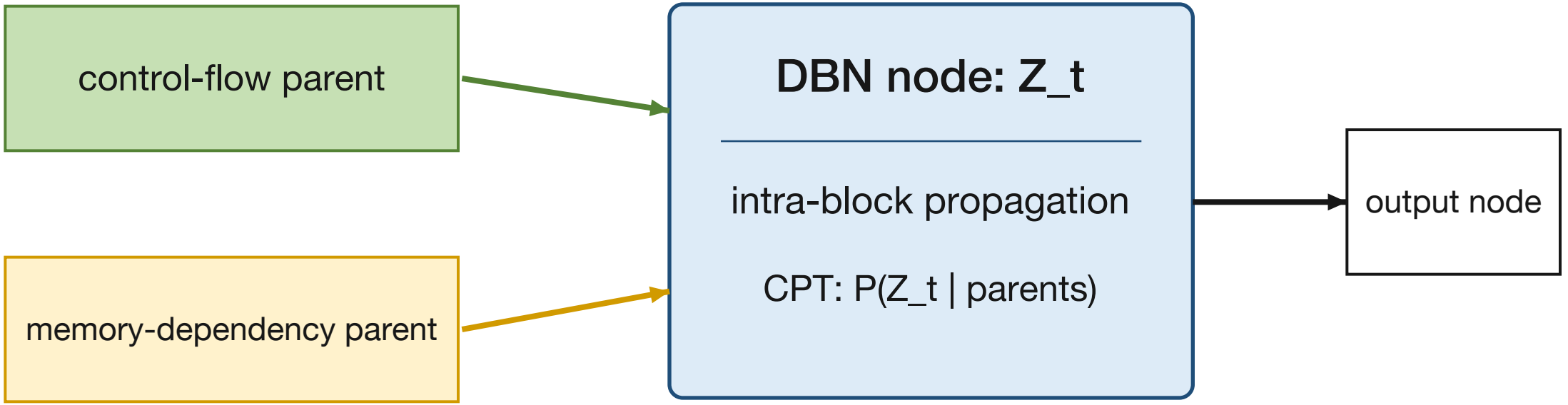
Memory-dependency edge — Address match + no intervening overwrite

Intra-block propagation — Loaded errors may be masked along the receiving DDS.

Selected dynamic instances; BB6 loop tests omitted. Gray arrows indicate execution order.

CPT generation: combining incoming propagation

Conditional probability table (CPT): node corruption given parent states.



Inter-block edges connect time-indexed DBN nodes.

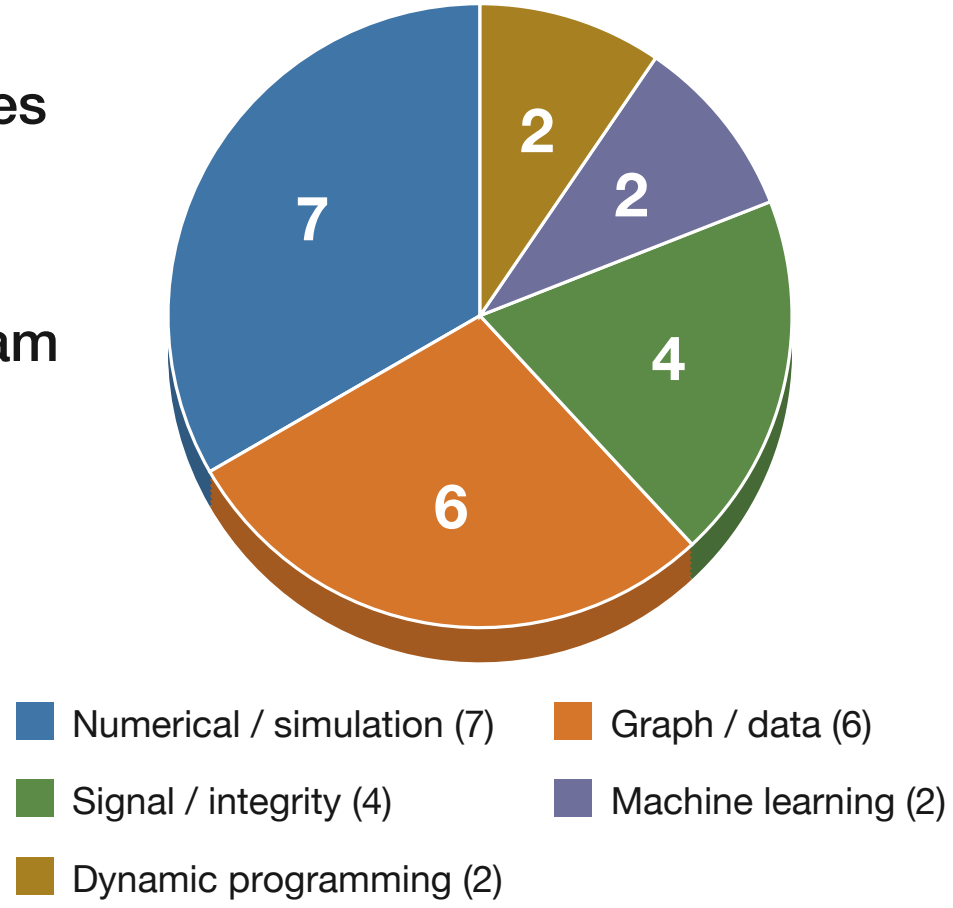
Experimental setup

21 programs across five domain families

3,000 single-bit injections per program

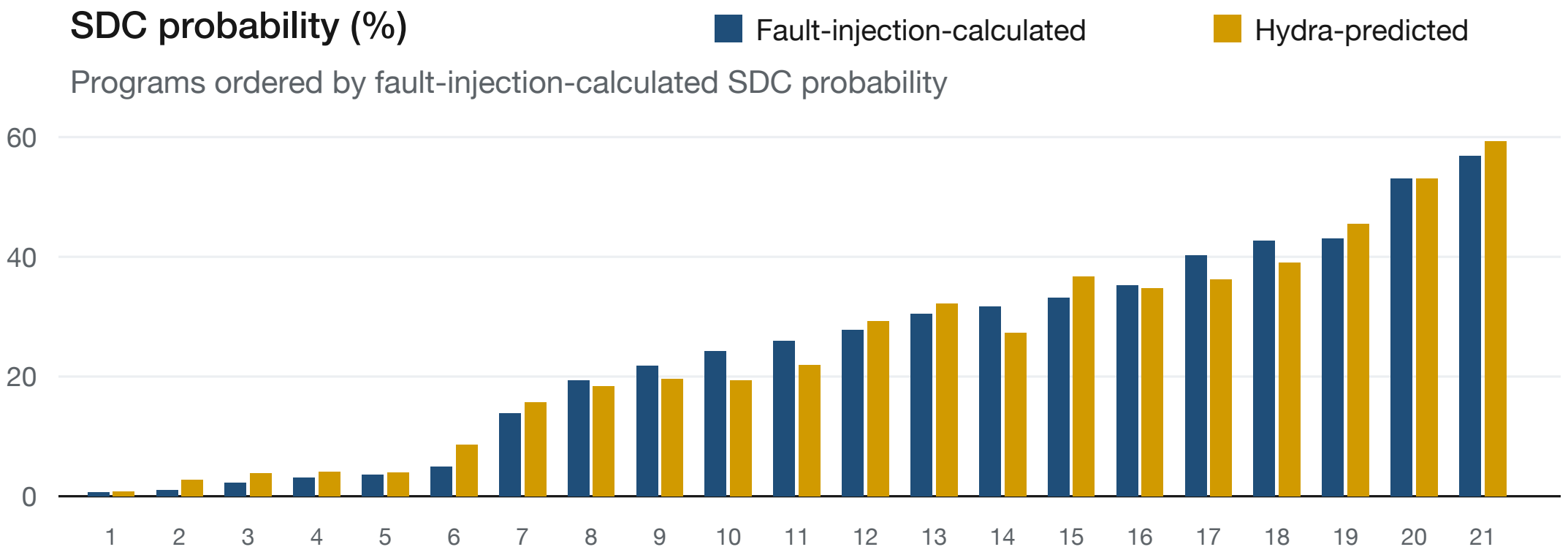
One sampled dynamic instruction per run

Compare Hydra-predicted with
fault-injection-calculated SDC probability



95% confidence intervals for fault-injection-calculated SDC probability: ±0.30 to ±1.77 pp.

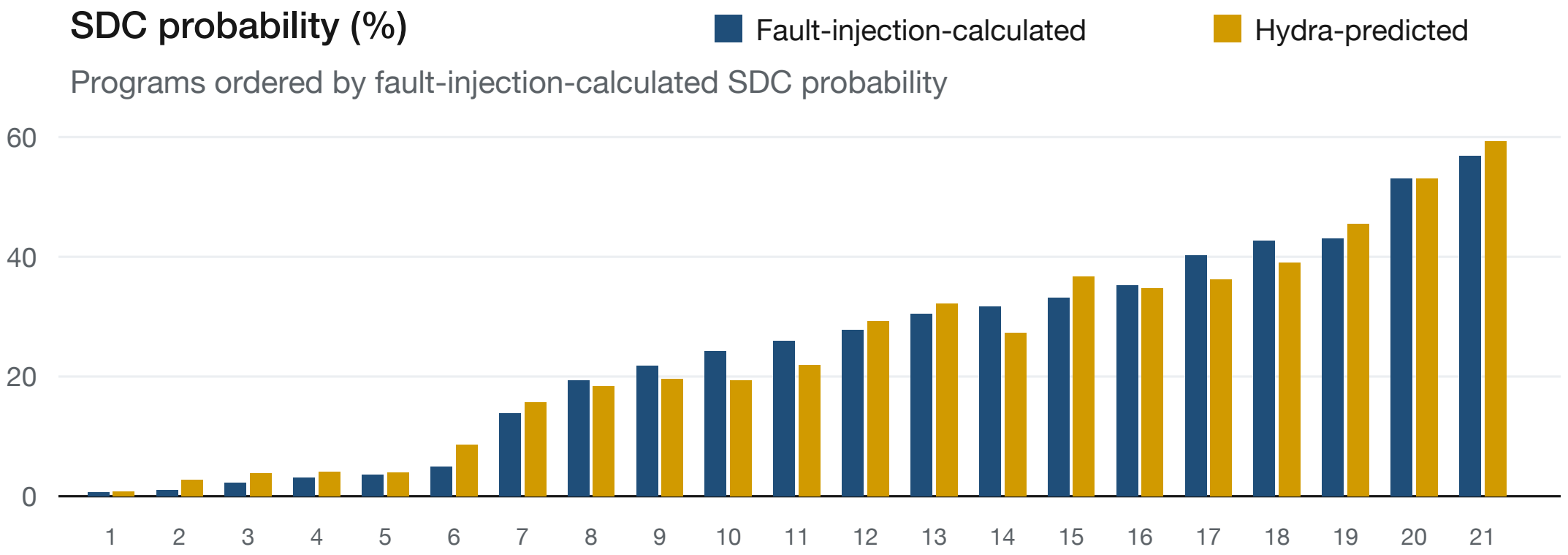
Hydra predicts SDC probability across 21 programs



Each pair compares fault-injection-calculated and Hydra-predicted SDC probability.

EVALUATION

Hydra predicts SDC probability across 21 programs



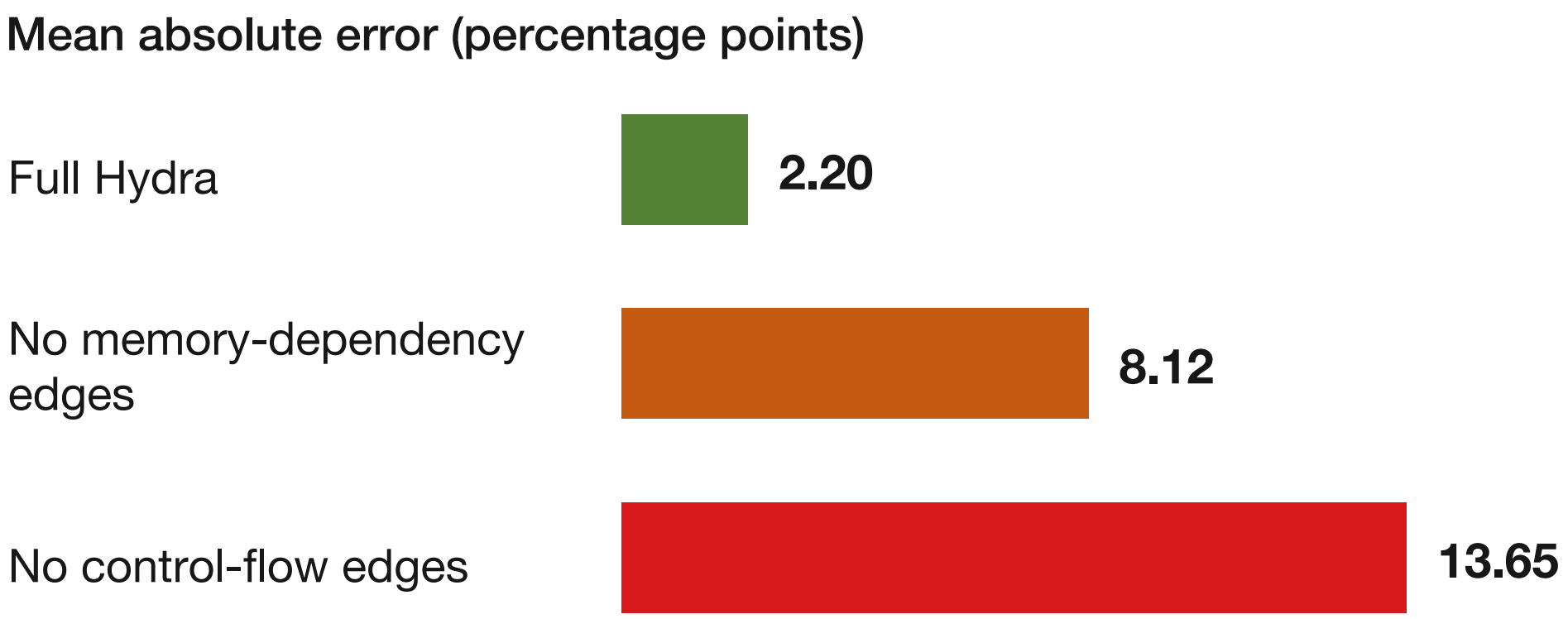
2.2 pp
mean absolute error

21 / 21
within 5 percentage points

24.4% / 24.5%
mean: Hydra / fault injection

Seven shared workloads: Hydra 1.4 pp MAE; earlier models 5.0–5.3 pp.

Both temporal edge types improve accuracy

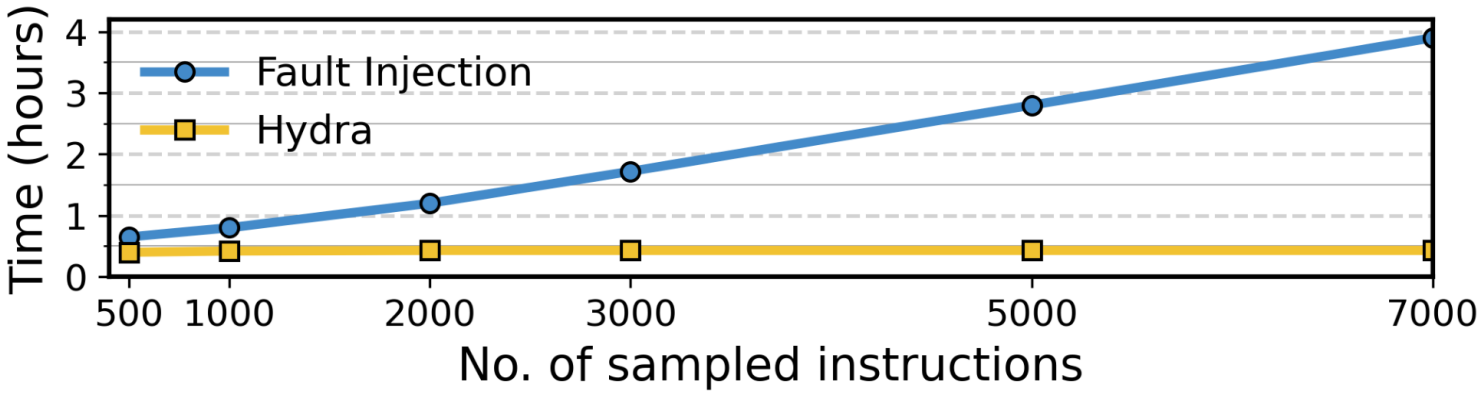


Lower error is better; both edge types contribute

Control-flow edges also carry forward propagation; edge effects interact.

Runtime and memory tradeoff

Runtime as sampled instructions increase



At 7,000 samples: Hydra < 0.4 h; fault injection nearly 4 h

Average memory

9.39 GB

Hydra model

110 MB

Fault injection

The time savings require substantially more memory.

Hydra makes propagation inspectable

Inference retains node corruption probabilities and inter-node edge flows.

WHERE	Where does corruption persist, fade, or return?
HOW	Which control-flow or memory-dependency edge carried the corruption?
SITE RISK	Which static instructions have higher estimated SDC risk?

Summary

- A transient datapath bit flip can silently change the final answer.
- Control flow, dynamic instances, and memory reuse determine propagation.
- Hydra represents those dependencies in a time-indexed DBN.

2.2 pp mean absolute error across 21 programs

Thank you

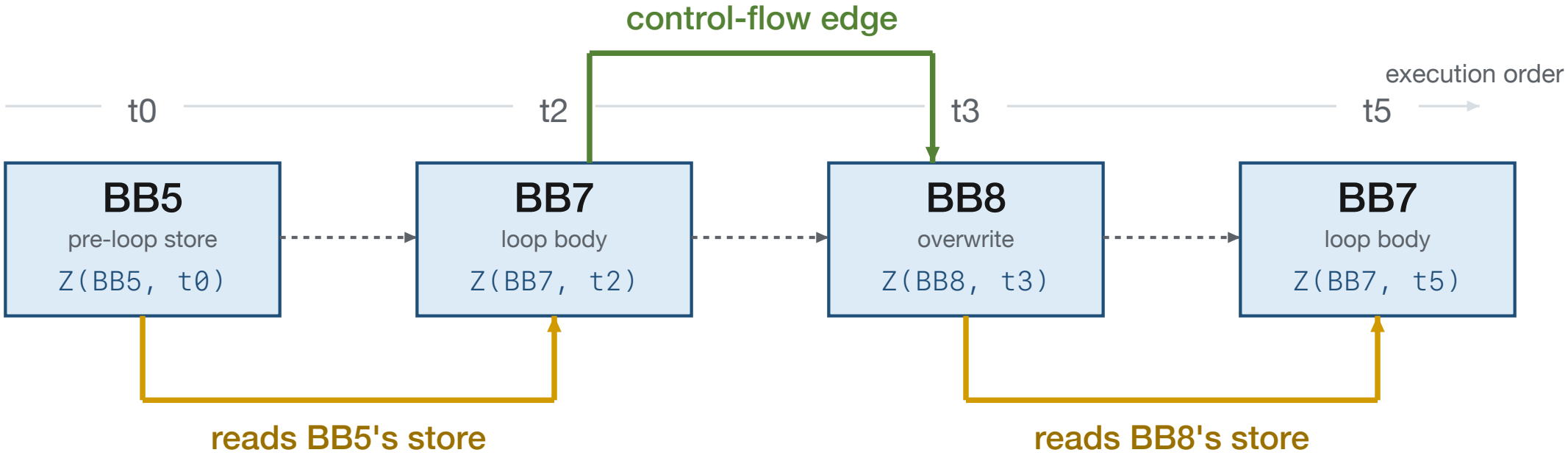
Questions?

Temporal Modeling of Soft Error Propagation
Using Dynamic Bayesian Networks

Backup slides

DBN nodes represent dynamic basic-block instances

The same loop example, now expressed as random variables.



Z = 0 No corruption reaches the node boundary.

Z = 1 Corruption reaches the node boundary; its CPT conditions on parent states.

Selected dynamic instances; BB6 loop tests omitted. Gray arrows indicate execution order.

Intra-block and inter-block propagation

A separate numerical example: source node S and receiving node R.



1.00 x 0.95 x 1.00 x 0.94 = 0.893

0.893 is the probability delivered along this route; other routes may also contribute.

What does a CPT contain?

A conditional probability table gives node corruption given its parent states.

The same isolated S-to-R route

Fault site in S activated?	P(R corrupt) via this route
0	0*
1	**0.893**

With two parents, the table expands

C	M	P(Z=1 | C,M)
0	0	0*
1	0	qC
0	1	qM
1	1	combine(qC,qM)

qC = control-flow edge weight x sC qM = 1 x sM

sC and sM: intra-block propagation probabilities from the corresponding entry instructions

* No other active parent or seeded fault. Multi-parent rows show the general case symbolically.

Instruction-level propagation: the icmp example

Illustrative 8-bit operands: a = 0, b = 4; flip one uniformly chosen bit of a.

`icmp ult i8 a, b`

Green bits change a < b

	7	6	5	4	3	2	1	0
a	0	0	0	0	0	0	0	0
b	0	0	0	0	0	1	0	0

a XOR b = 00000100; highest differing bit k = 2

Flip a[1]: 2 < 4, still true

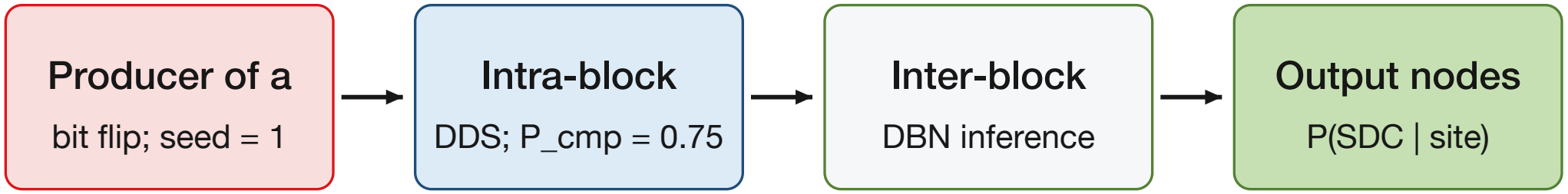
Flip a[2]: 4 < 4, now false

P_cmp ~ (8 - 2) / 8 = 0.75

Instruction-level propagation probability, not SDC probability

From intra-block propagation to site SDC probability

Here the activated fault is in a before it reaches the comparison.

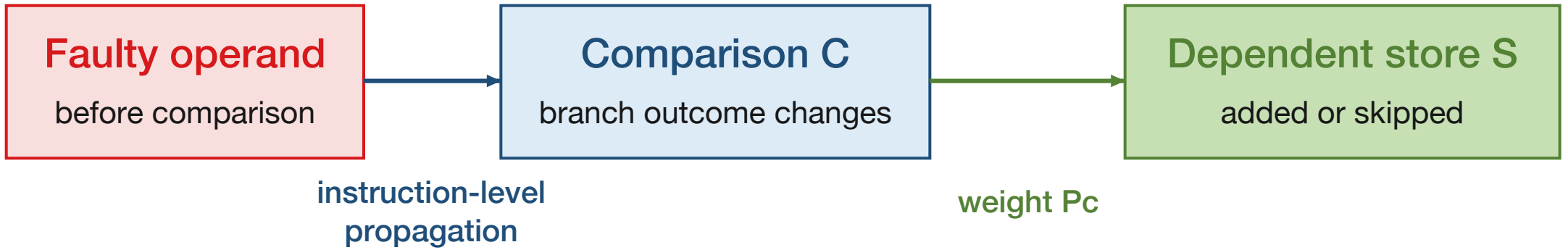


Intra-block	Multiply along a DDS; noisy-OR approximates multiple causal paths.
Inter-block	Infer through control-flow and memory-dependency edges to the DBN's output nodes.
Static view	Aggregate dynamic site estimates for each static instruction.

A fault in icmp's output starts after its own masking step.

What does a control-flow edge weight mean?

Instruction masking and control dependence are separate steps.



1. Reach the predicate

Operand features estimate whether the corrupted input changes the comparison.

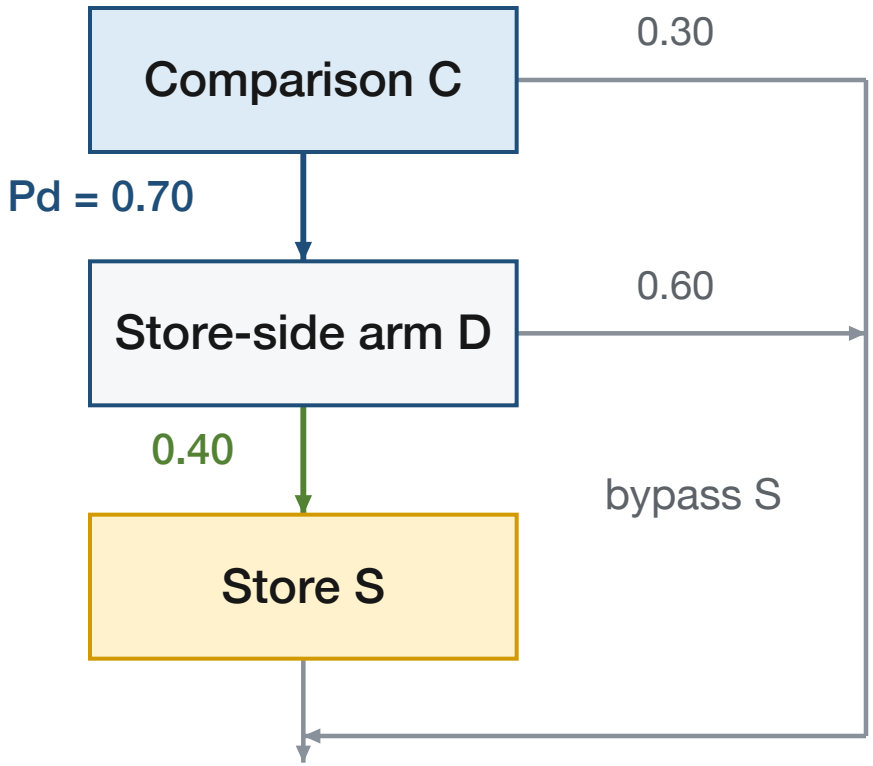
2. Affect a later store

Control dependence identifies candidate stores; profiled branches set Pc.

The final SDC estimate also depends on propagation after the store.

Non-loop controlling comparisons

Illustrative profiled frequencies; C selects between two forward paths.



Of 100 branch encounters

70 take arm D

28 reach store S

Given arm D: 28 / 70 = 40%

```
Pc = Pe / Pd
= 0.28 / 0.70 = 0.40
```

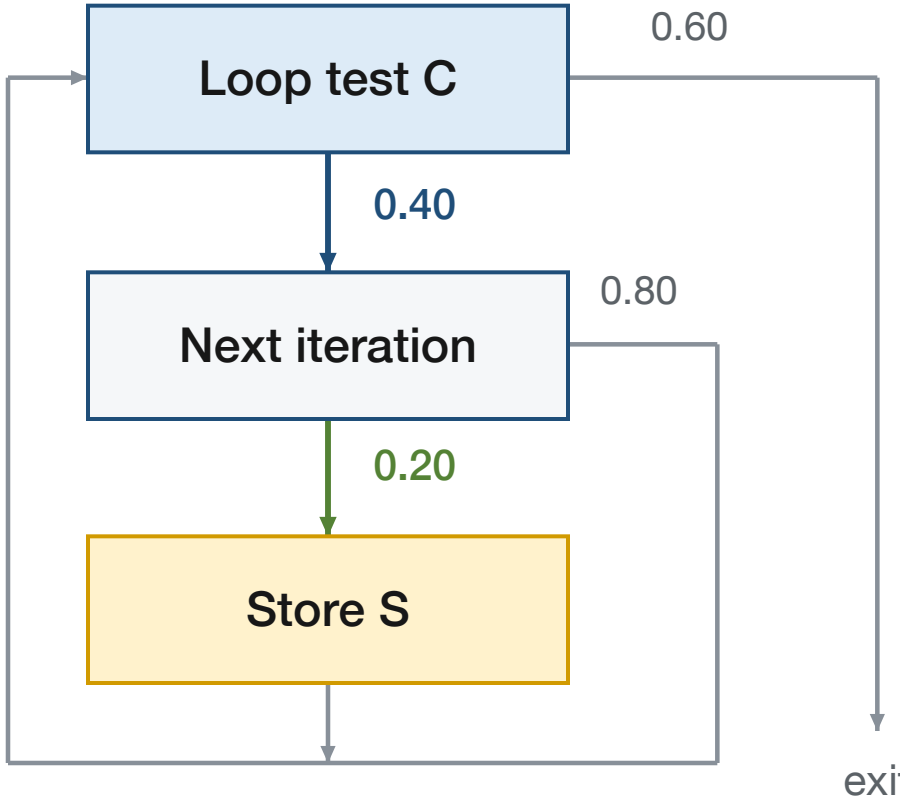
Divide out the arm probability to condition on taking that arm.

Illustrative counts, not fault-injection outcomes. Pe is the full-path store probability.

Related derivation: Li et al., Modeling Soft-Error Propagation in Programs, DSN 2018, Sec. IV-D

Loop controlling comparisons

Continuation and store reachability must both be accounted for.



`Pb = 0.40`

probability of continuing

`Pe = 0.20`

P(store S | continuation)

`Pc = Pb x Pe = 0.08`

Hydra's loop control-flow edge weight

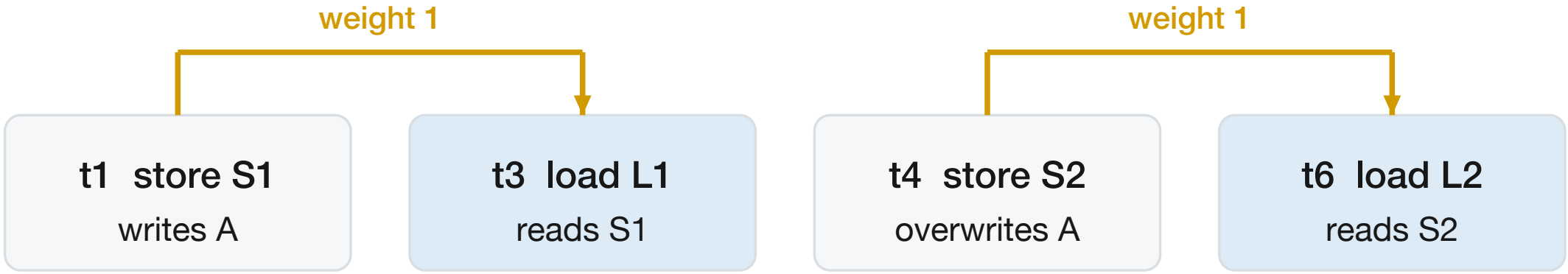
Pe is conditional on continuing; multiply by Pb to weight that route.

Prior derivation assumes rare exits; this is not an exact probability over all predicate flips.

Related derivation and approximation: Li et al., DSN 2018, Sec. IV-D (Trident)

How are memory-dependency edges assigned?

Follow the observed read-from relation at one address A.



S2 ends S1's direct memory-dependency edge to later loads of A.

Matching address + no intervening overwrite -> edge

After the load, masking is handled by intra-block propagation along the receiving DDS.

Ablation detail: both edge types matter

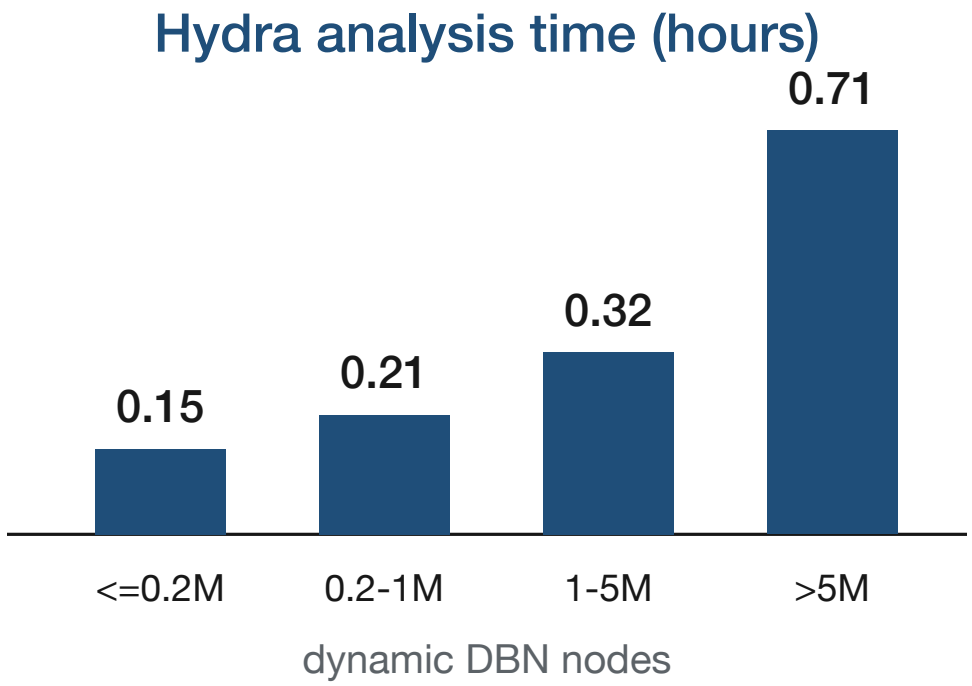
Mean absolute inaccuracy (SDC percentage points); lower is better.

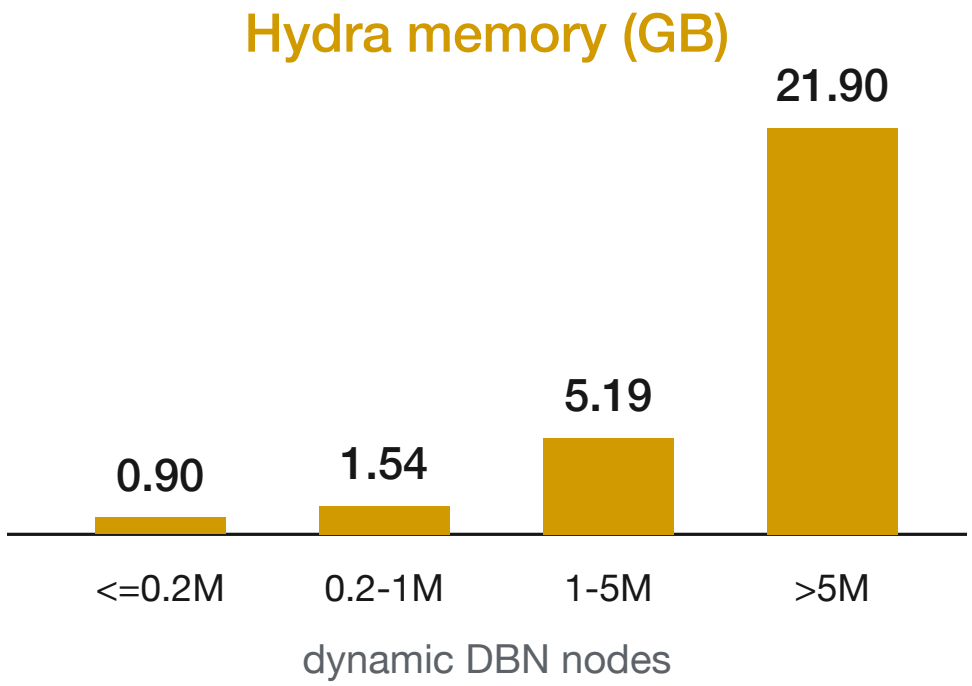
Workload group	Full Hydra	No memory-dependency edges	No control-flow edges
Dense math	2.25	8.27	14.90
Irregular memory	2.01	9.80	14.17
Graph / pointer	2.60	6.13	10.00
Overall	**2.20**	**8.12**	**13.65**

Intra-block DDS propagation remains; inter-block edge effects interact and are not additive.

What grows with DBN node count?

Group averages across four bins of dynamic basic-block instances.





These are size-group averages; the sample-count plot averages across programs.

How does Hydra compare with prior models?

Seven matched benchmarks and input workloads; lower error is better.

Model	Approach	MAE (pp)	Mean RI	RMSD (pp)
Hydra	Temporal analytical DBN	**1.4**	**12.3%**	**1.6**
Trident	Analytical propagation	5.3	25.8%	7.1
PARIS	Learned FI-based features	5.0	36.7%	6.5

Hydra: 3.8x lower MAE than Trident; 3.6x lower than PARIS on this subset.

Runtime: Hydra 0.40 h; Trident 0.36 h. Competitors lead on some individual rows.

Trident: Li et al., DSN 2018 | PARIS: Guo, Li and Laguna, JPDC 152, 2021

What are the model's practical limits?

Input-specific trace

One fault-free run with a representative input supplies the timeline. A new input needs a new profile.

Bounded divergence

Hydra models path changes probabilistically on the profiled execution trace. It does not enumerate every path.

Memory cost

The dynamic trace and DBN require 9.39 GB on average; the largest node bin averages 21.90 GB.

Report predictions for the profiled workload and its model size