Branch Trace Compression for Snapshot-Based Simulation

Kenneth Barr

Krste Asanović

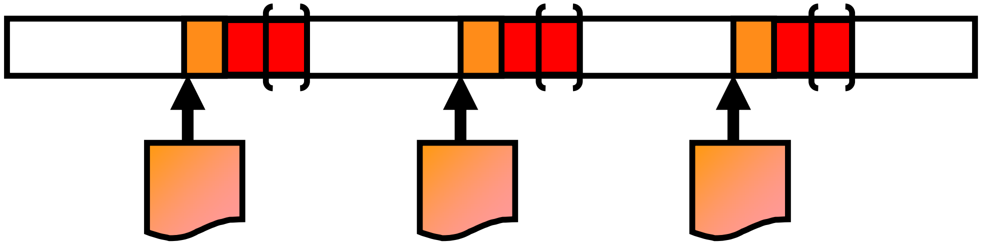
Massachusetts
Institute of
Technology

CSAIL

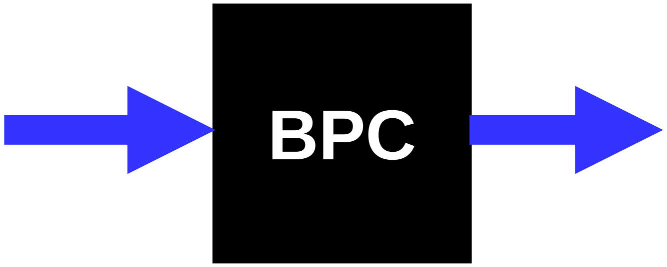
BARC
February 3, 2006

BPC: compact, fast, flexible warming of branch predictors for snapshot-based simulation.

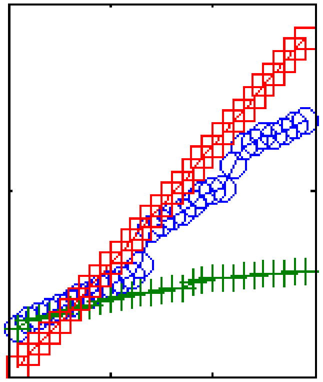
1. Motivation, simulation context, vocabulary



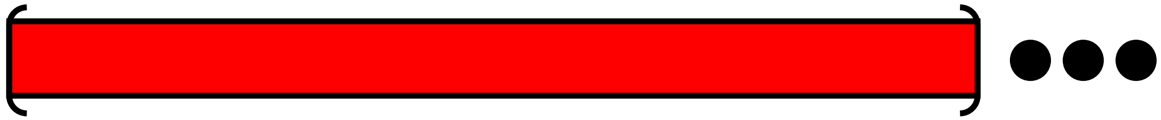
2. Branch Predictor-based Compression **(BPC)**
   - Compress traces instead of storing snapshots



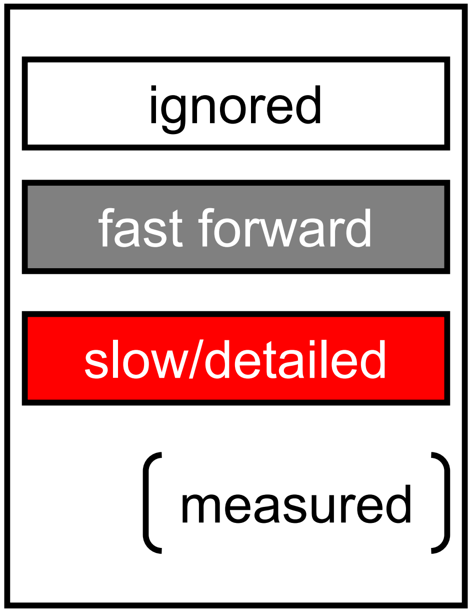
3. Preview of results
   - Size
   - Scalability
   - Speed

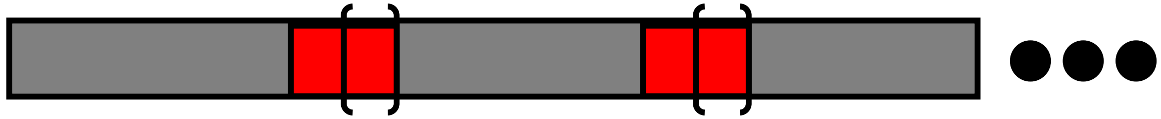


Intelligent sampling gives best speed-accuracy tradeoff for uniprocessors (Yi, HPCA `05)



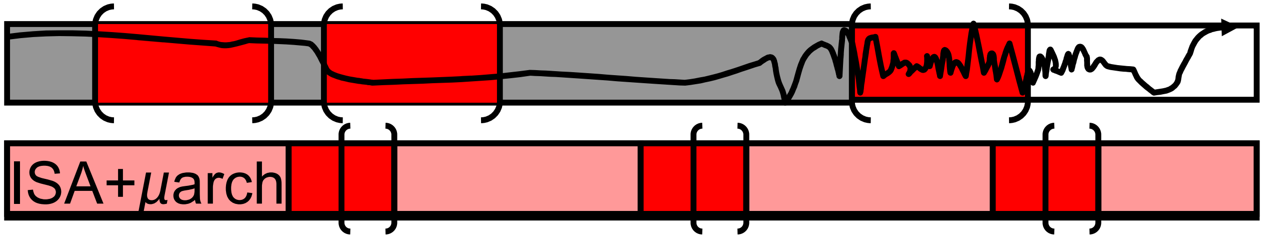
Run benchmark entirely
in detailed mode: slow!



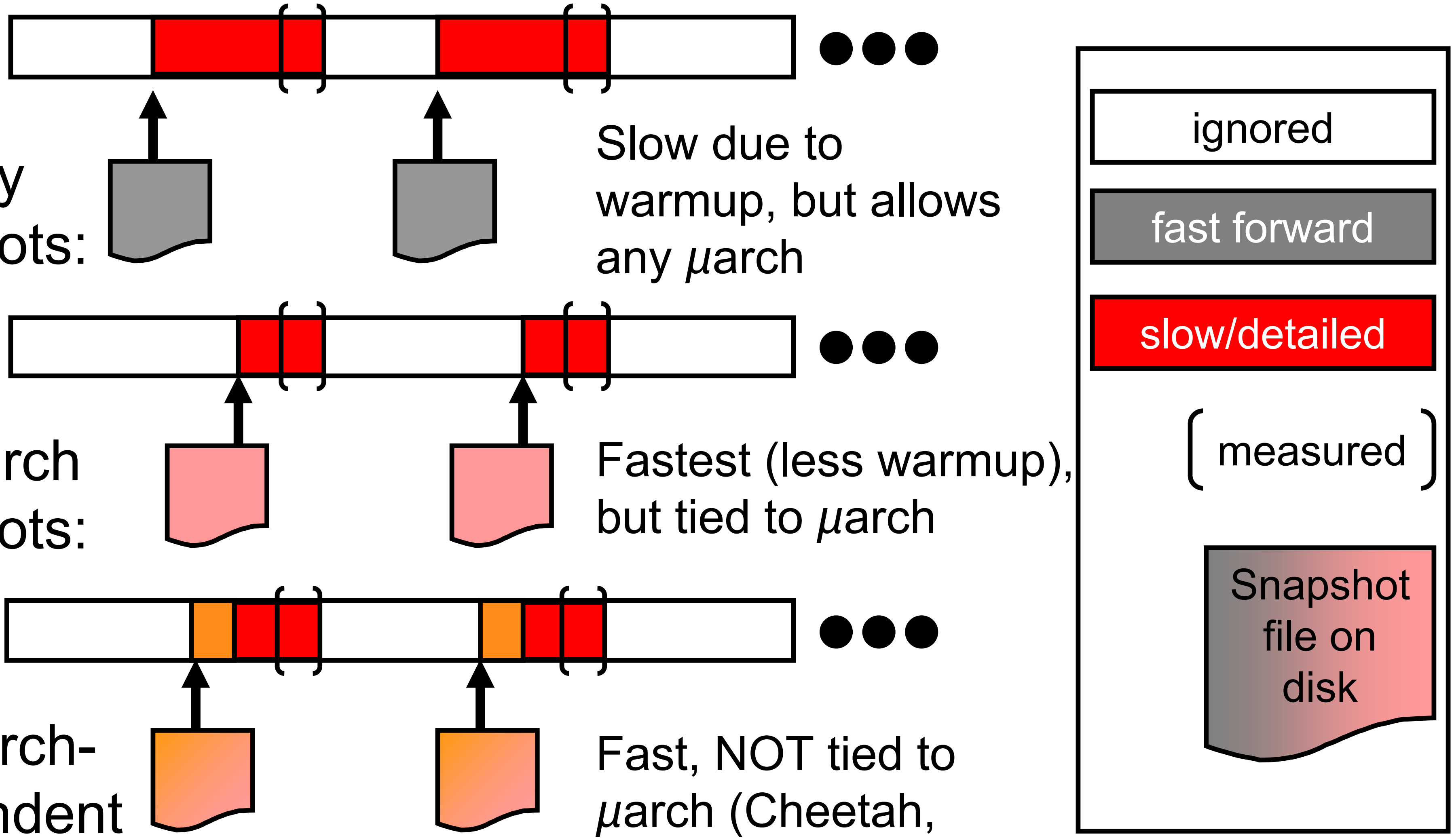


Aggregate detailed samples

Variations



Snapshots amortize fast-forwarding, but require slow warming or bind us to a particular *µ*arch.



ISA only snapshots:

Slow due to warmup, but allows any *µ*arch

ISA+*µ*arch snapshots:

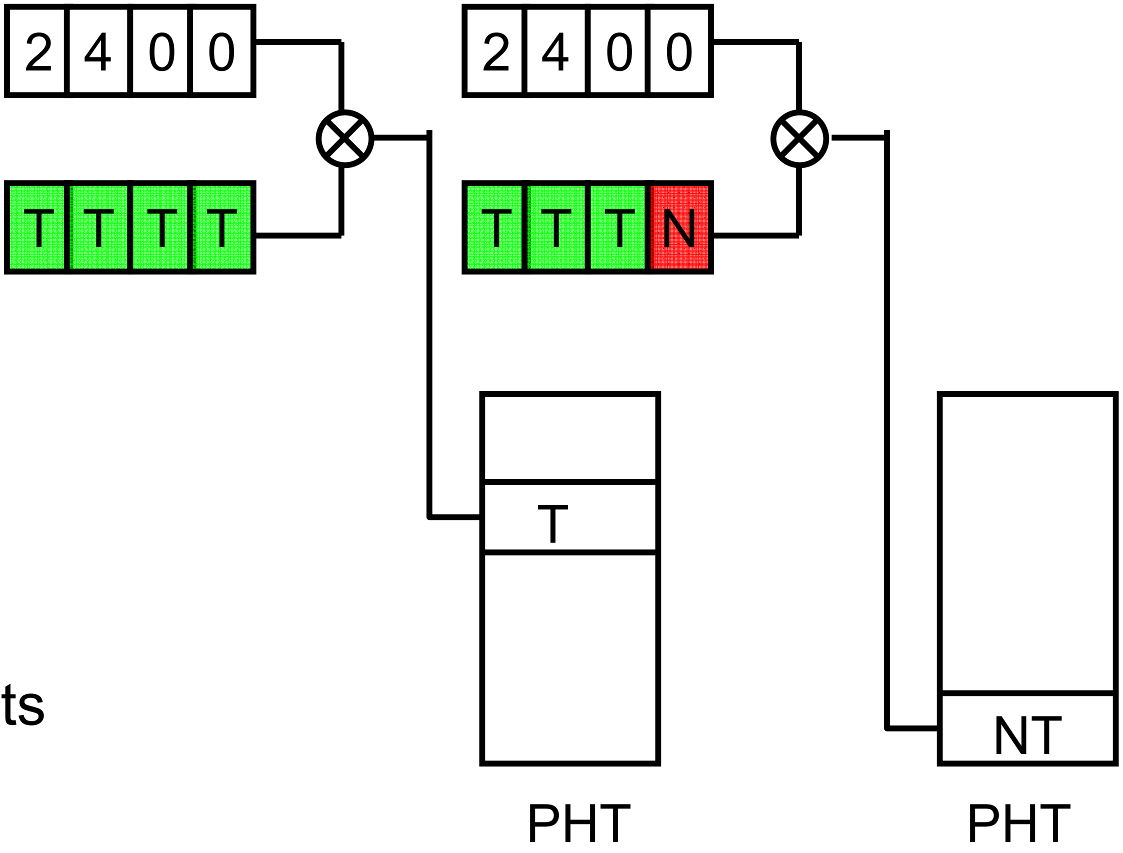
Fastest (less warmup), but tied to *µ*arch

ISA+*µ*arch-independent snapshots:

Fast, NOT tied to *µ*arch (Cheetah, MTR)

Why can't we create *μ*arch-independent snapshot of a branch predictor?

- In cache, an address maps to a particular cache set
- Branch history (global or local) "smears" static branch across the pattern history table
  - Same branch address…………..
  - In a different context……………



- In a cache, we can throw away LRU accesses
- In a branch predictor, who knows if ancient branch affects future predictions?!

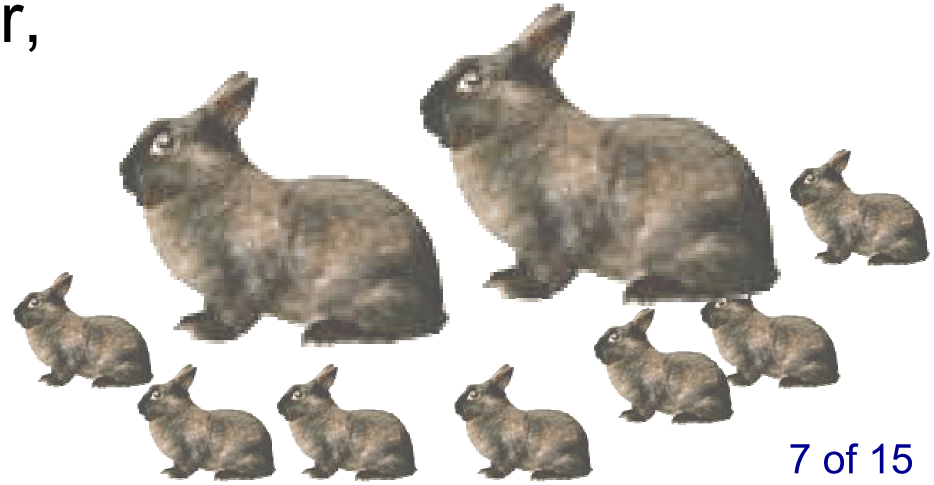
If a *μ*arch independent snapshot is tricky, let's try to store several predictor tables?

- Suggested by [SMARTS, SimPoint]
- Is this an option?
  - If you generate snapshots via hardware dumps, you can't explore other microarchitectures
- Which ones?
  - If it takes two weeks to run a non-detailed simulation of a real workload you don't want to guess wrong
- Those branch predictors aren't as small as you think!

Branch predictors are small, but multiply like rabbits! 8KB quickly becomes 1000’s of MB.

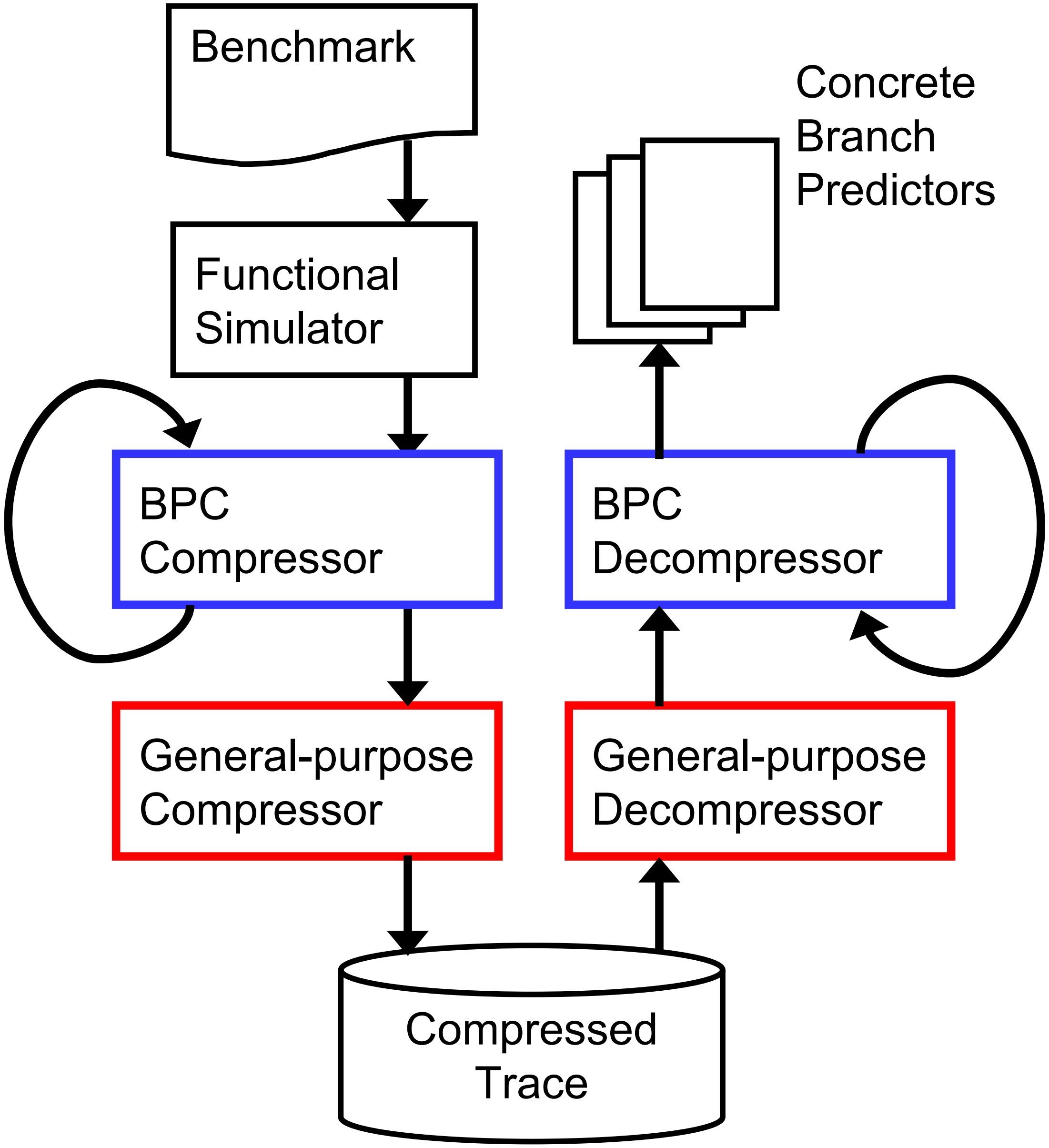
- ***P:*** gshare with 15 bits of global history
- ***n***: 1 Billion instructions in trace sampled every million insts requires 1000 samples
- ***m***:10 other tiny branch predictors
- 26 benchmarks in Spec2000
- 16 cores in design?
- Now, add BTB/indirect predictor, loop predictor…
- Scale up for industry: 100 benchmarks, 10s of cores

	8	KBytes
x 1000	**=** 8	MBytes
x 10	= 78	MBytes
x 26	= 2.0	GBytes
x 16	= 32	GBytes

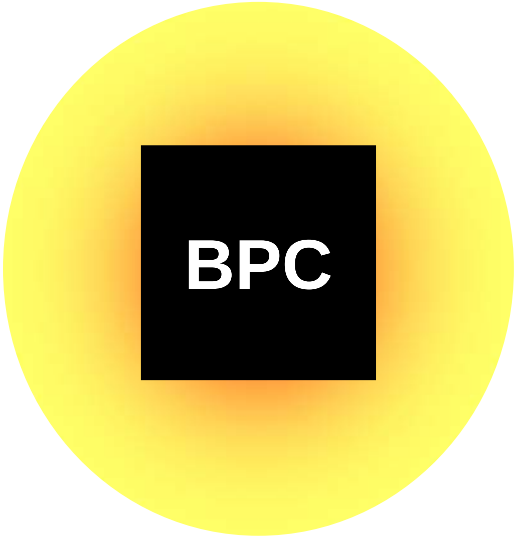


BPC compresses branch traces well and quickly warms up any concrete predictor.

- Simulator decodes branches
- BPC Compresses trace
  - Chaining if necessary
- General-purpose compressor shrinks output further
  - PPMd
- Reverse process to fill concrete predictors

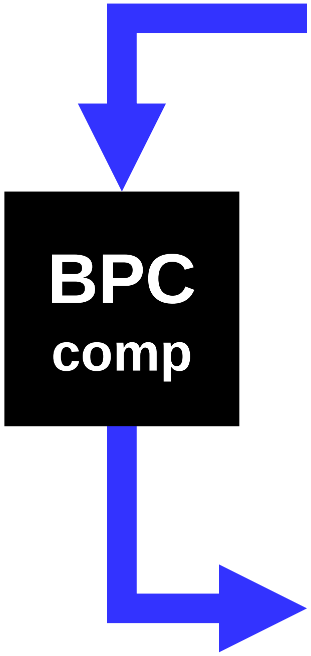

Benchmark
Functional Simulator
Concrete Branch Predictors
BPC Compressor
BPC Decompressor
General-purpose Compressor
General-purpose Decompressor
Compressed Trace

BPC uses branch predictors to model a branch trace. Emits only unpredictable branches.



- Contains the branch predictors you always dreamed about!
  - Large global/local tournament predictor
    - 1.44Mbit
    - Alpha 21264 style
  - 512-deep RAS
  - Large hash tables for static info
    - Three 256K-entry
  - Cascaded indirect predictor
    - 32KB leaky filter
    - path-based (4 targets)
    - 2 entries
    - PAg structure

BPC Compression


BPC
comp

Input: branch trace from functional simulator

```
0x00:  bne    0x20  (NT)
0x04:  j      0x1c  (T)
0x1c:  ret          (T to 0xc4)
```

Output:

– If BPC says “I could have told you that!” (Common case): no output
  < >

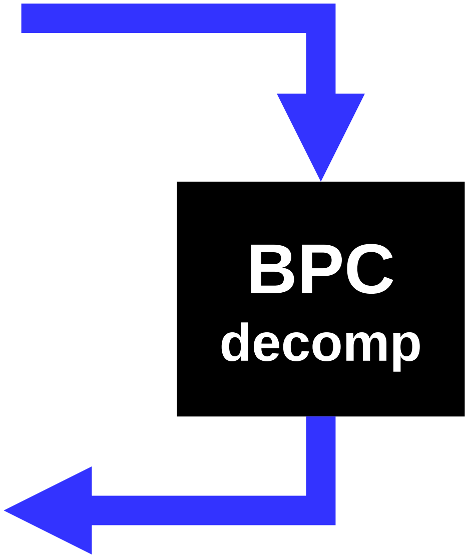
– If BPC says “I didn’t expect *that* branch record!”
  < skip N, branch record >

Update internal predictors with every branch.

BPC Decompression

Input: list of pairs < skip N, branch record >

```
< 0,     0x00:  bne 0x20 (NT) >
< 0,     0x04:  j   0x1c (T)  >
< 13,    0x3c:  call 0x74     >
```



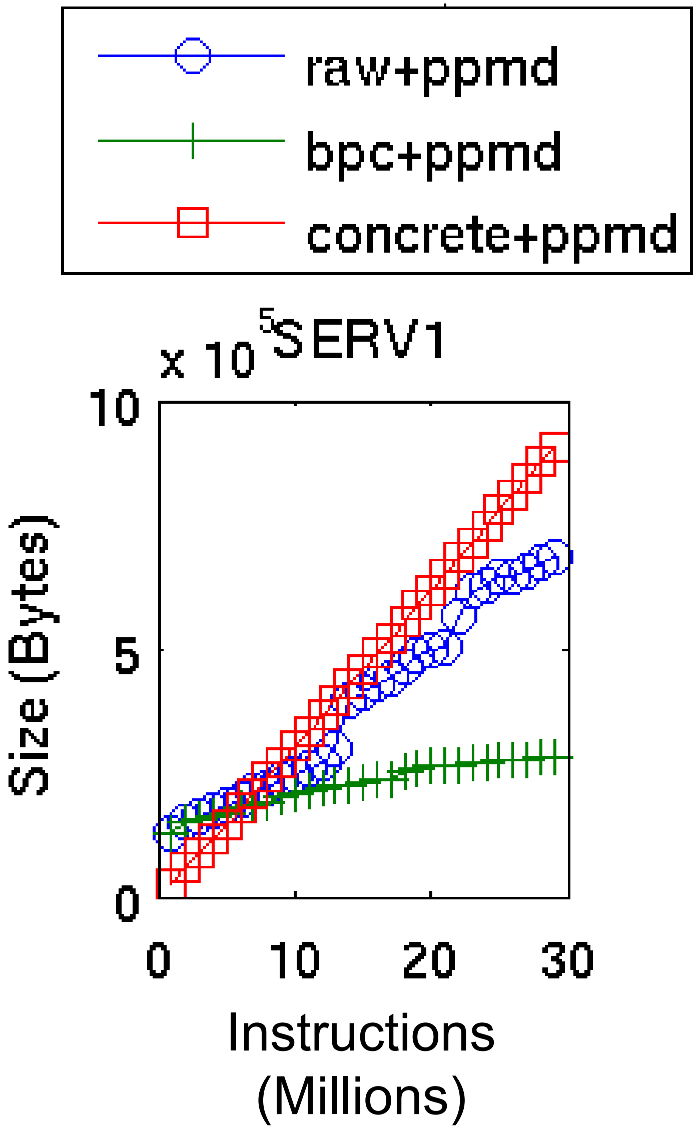
Output:

```
if (skip==0)
    branch record
    // updates predictors

while(skip > 0)
    BPC says “let me guess!”
    // updates predictors
    // decrement skip
```

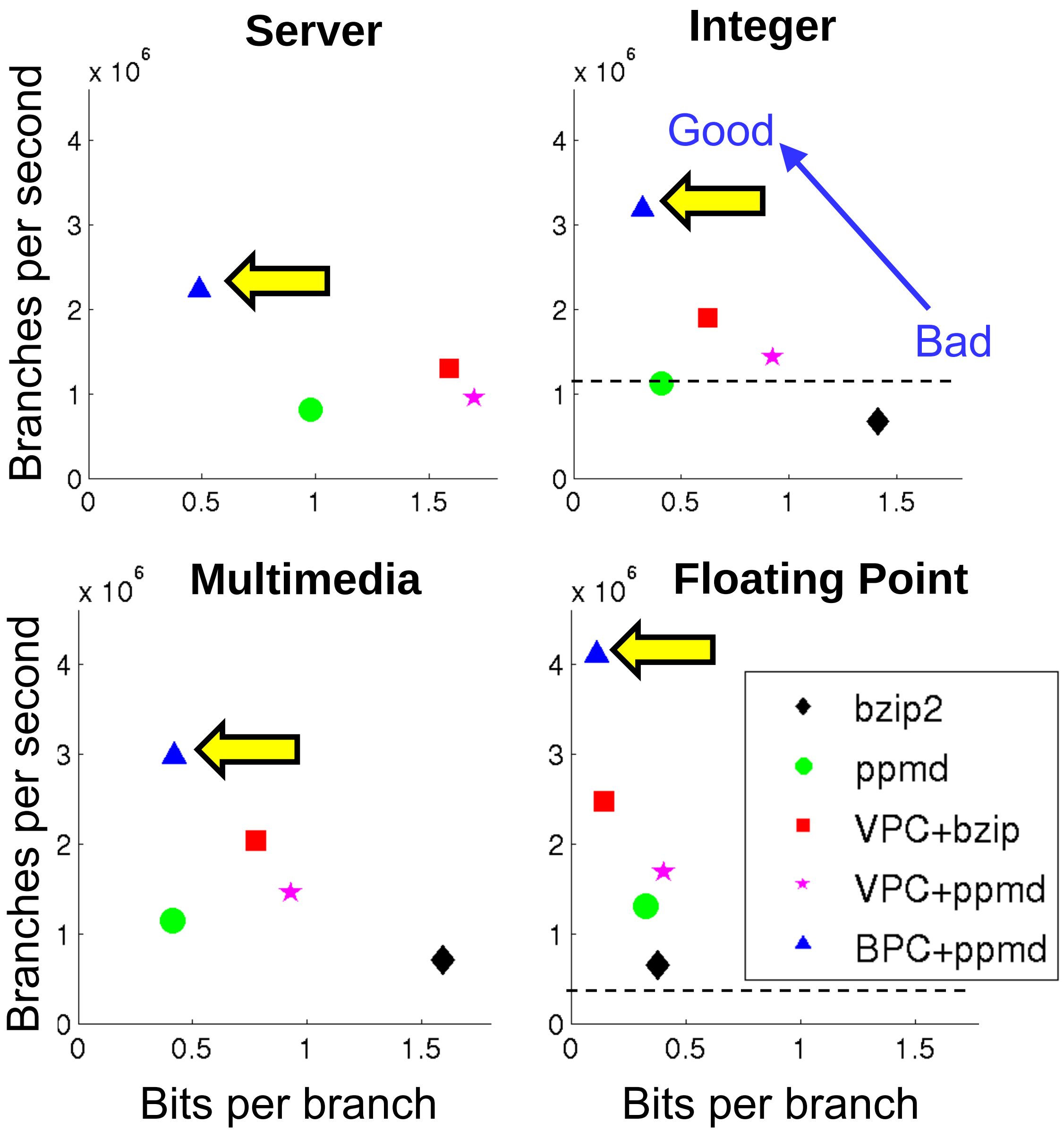
BPC-compressed traces grow slower than concrete snapshots

- We compare against **one** stored Pentium 4 style predictor: 2.7X smaller (avg)
- If you store 1000 samples, 10 predictors…
  - 11 MB for BPC
  - 310 MB for concrete snapshot
- Growth
  - BPC has shallow slope
  - concrete scales with *mnP*
  - Both grow with number of benchmarks and cores



Summary: BPC decompresses faster, compresses as good or better than others.

- BPC+PPMd faster than other compressors **and** sim-bpred
- Know your general-purpose compressors: gzip’s too big bzip2 is too slow
- Biggest help for phase-changing Server code



Related work: BPC is a specialized form of VPC or a modified version of CBP.

- Value-predictor based compression (VPC)
  - Prof. Martin Burtscher at Cornell
  - Trans on Computers, Nov 2005
- Championship Branch Prediction Contest (CBP)
  - Stark and Wilkerson, Intel
  - MICRO workshop, Jan 2005
  - Provided traces used a technique with similar spirit
- Our Branch Prediction-based Compression (BPC) paper identifies application to snapshot-based simulation
  - Barr and Asanović, MIT
  - ISPASS, Mar 2006

Conclusion

- Compressed branch traces are smaller than concrete branch predictor snapshots
  - 2.0–5.6x smaller than a **single**, simple predictor snapshot
  - Improvement multiplies for each predictor under test, size of those predictors, and each additional sample
- We introduce Branch Predictor-based Compression
  - Better compression ratios than other compressors
  - Faster than other decompressors; and 3-12X faster than functional simulation. Slower than µarch snapshots, but infinitely more flexible.

- Full-length paper: ISPASS, March 2006
- **`http://cag.csail.mit.edu/scale`**