# Branch Trace Compression for Snapshot-Based Simulation

Kenneth Barr

Krste Asanović

Massachusetts Institute of Technology

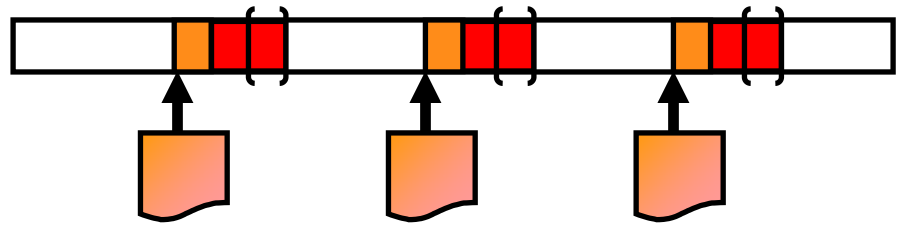
CSAIL

ISPASS

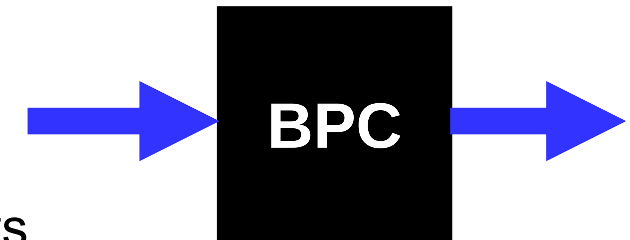
March 20, 2006

# Branch Trace Compression for Snapshot-Based Simulation

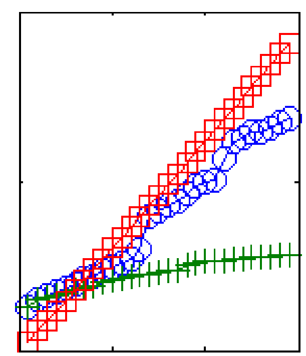
1. Motivation, simulation context, vocabulary



2. Branch Predictor-based Compression **(BPC)**
   - Compress branch traces instead of storing predictor snapshots
   - Goal: reduce storage requirements, increase flexibility, high speed



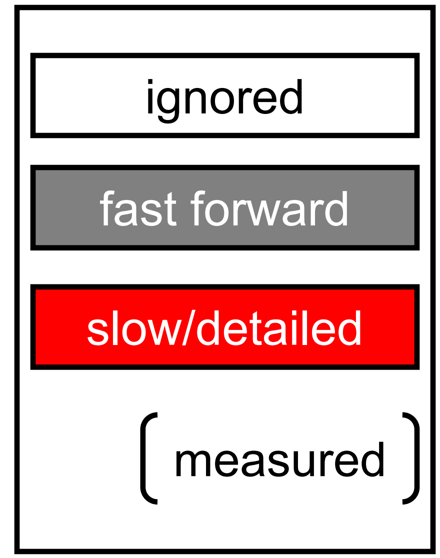

3. Preview of results
   - Size
   - Scalability
   - Speed



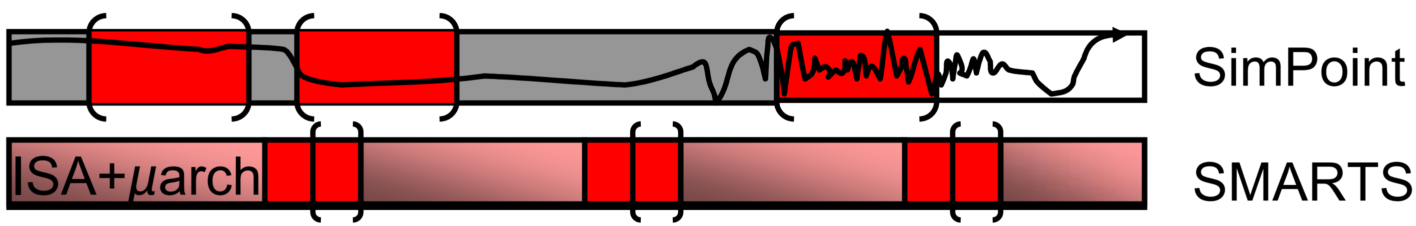
# Intelligent sampling gives best speed-accuracy tradeoff for uniprocessors (Yi, HPCA `05)

Run benchmark entirely
in detailed mode: slow!

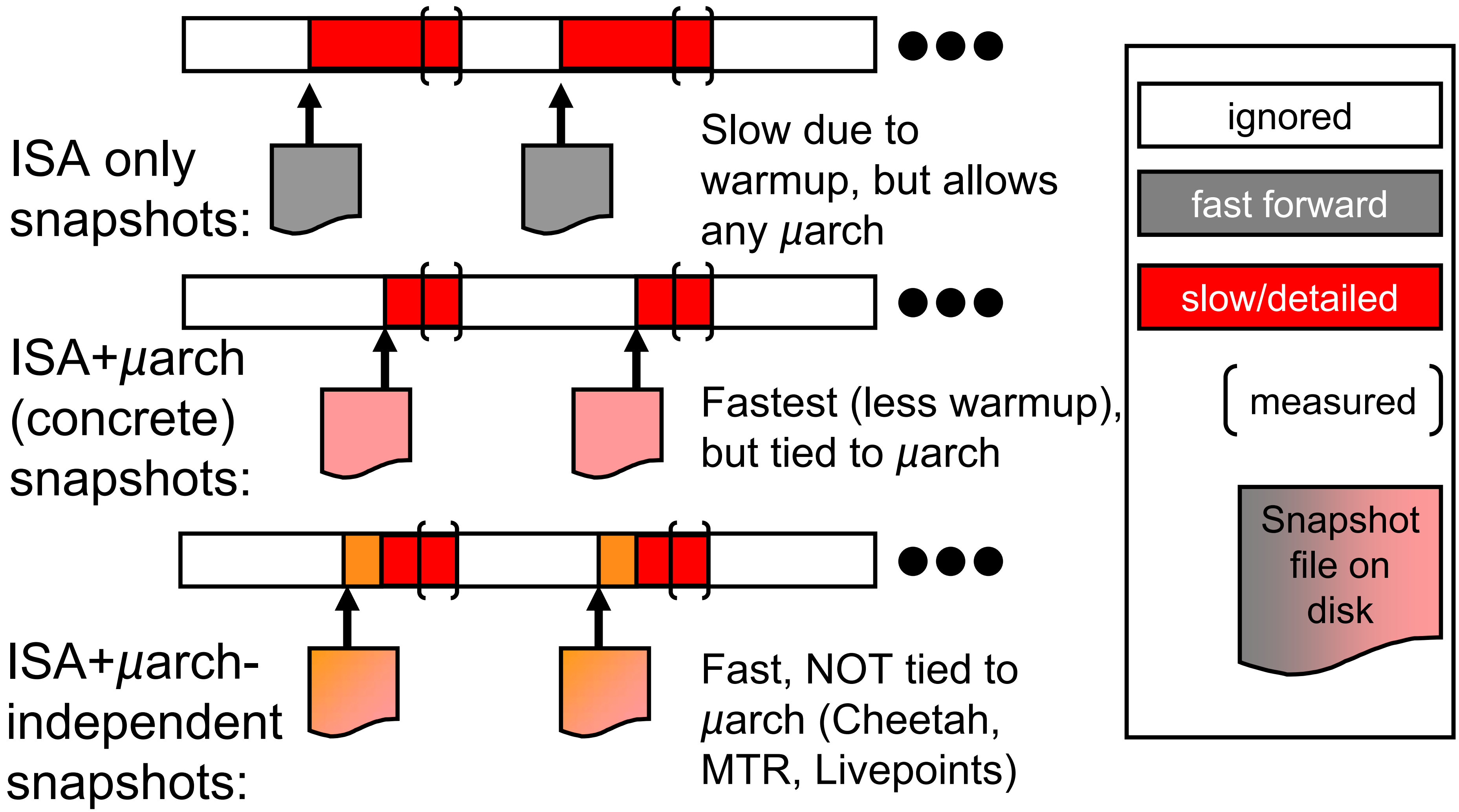
Aggregate detailed samples




Improvements

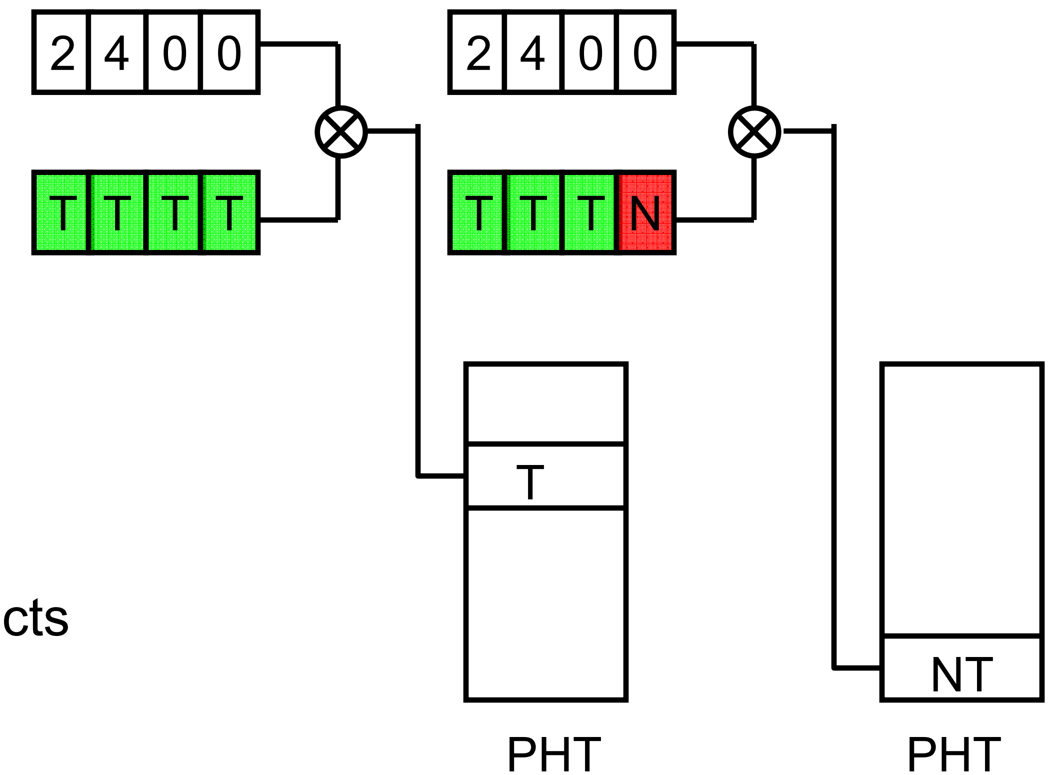


# <u>Snapshots</u> amortize fast-forwarding, but imply slow warming or binding to a particular µarch.



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

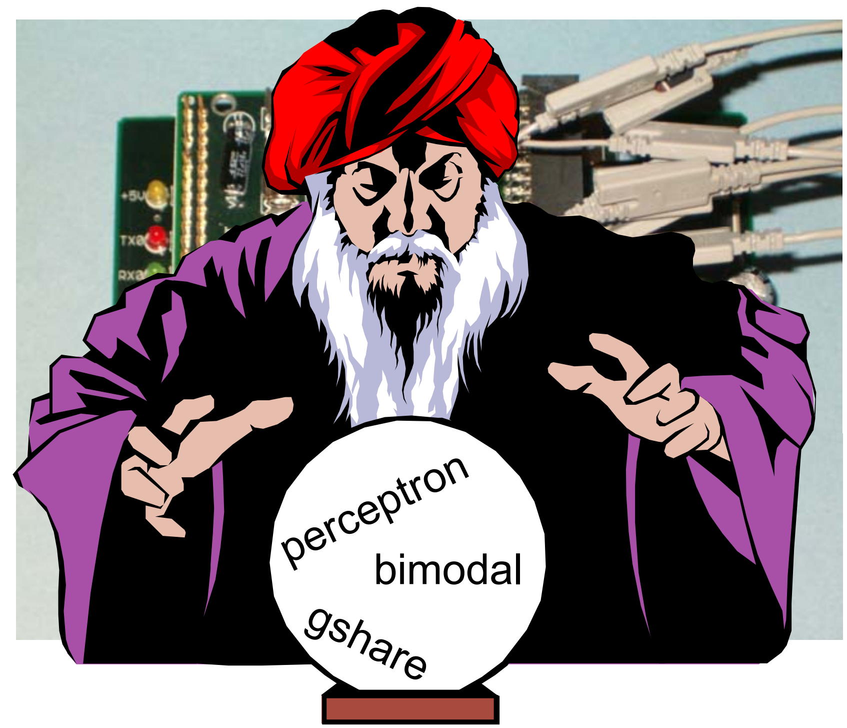
- In cache, an address maps to a particular cache set.
- In branch predictor, an address maps to **many** locations. We combine address with **history** to reduce aliasing and capture context.
  – Same branch address…………..
  – In a different context……………




- 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, can we store several branch predictor tables?

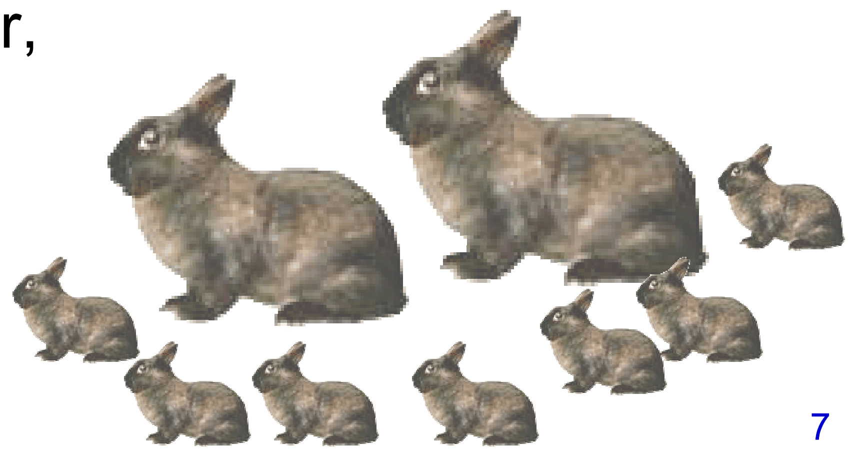
- Suggested by
  - TurboSMARTS / Livepoints SIGMETRICS ’05 / ISPASS ’06
  - SimPoint Group: HiPEAC ‘05
- Not always an option
  - If you generate snapshots via hardware dumps, you can’t explore other microarchitectures
- Requires predicting the future
  - If it takes two weeks to run a non-detailed simulation of a real workload you don’t want to guess wrong
- “Several branch predictor tables” aren’t as small as you think!



# One predictor is small, but we need many. Example: 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 |

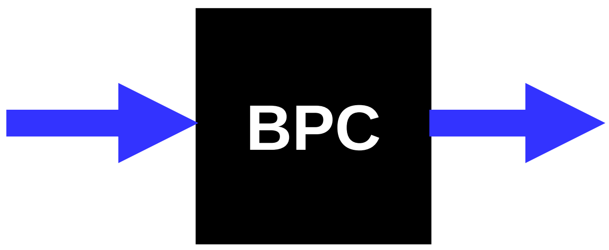


# Don’t store collection of concrete snapshots!

# Don’t store collection of concrete snapshots! Store entire branch trace… with BPC

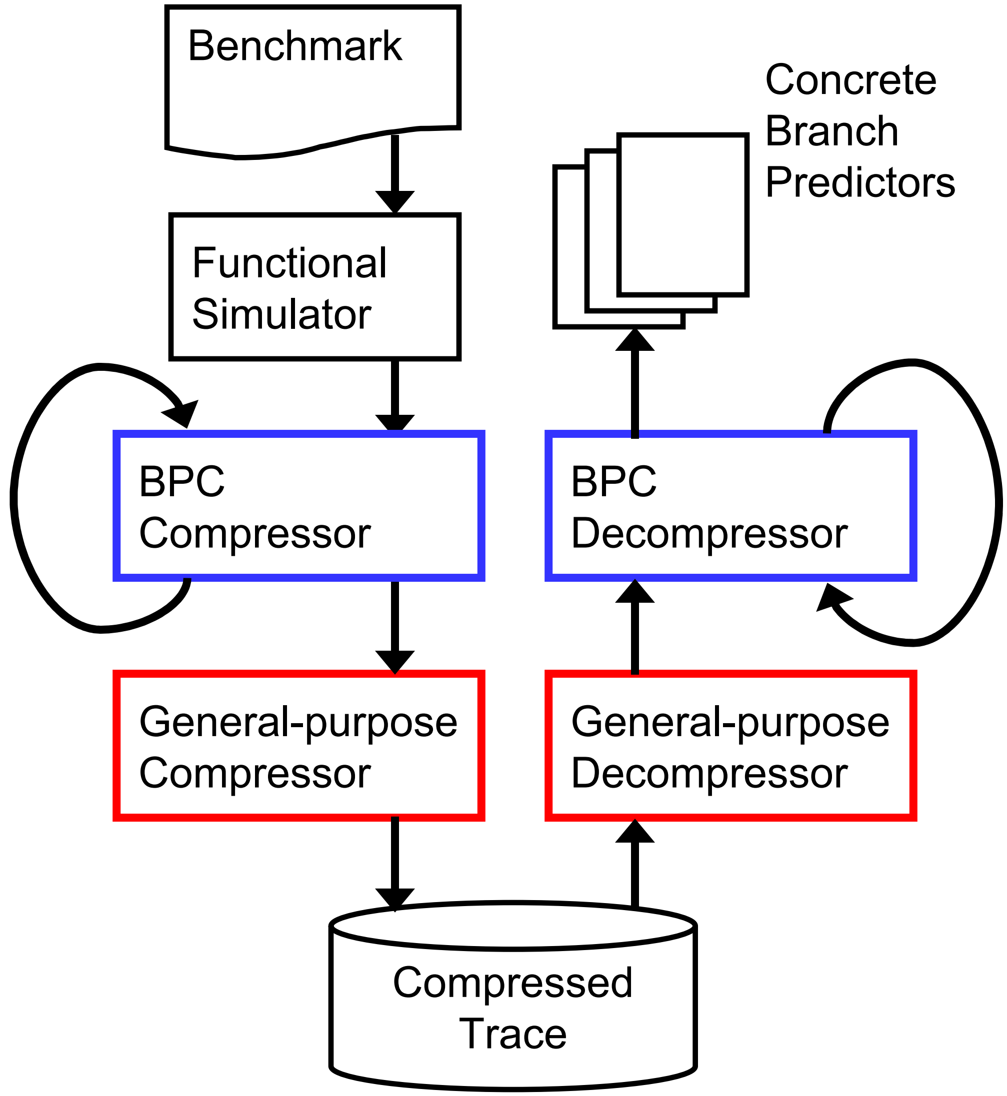
BPC = **B**ranch **P**redictor-based **C**ompression

- Flexible
  - Store entire branch trace: inherently microarchitecture independent
- Fast:
  - Simple algorithm allows fast decompression
- Small Footprint:
  - Compresses to < 0.5 bits/branch



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

1. Simulator decodes branches
2. BPC Compresses trace
   – Chaining if necessary
3. General-purpose compressor shrinks output further
   – PPMd
4. Reverse process to fill concrete predictors, one branch at a time

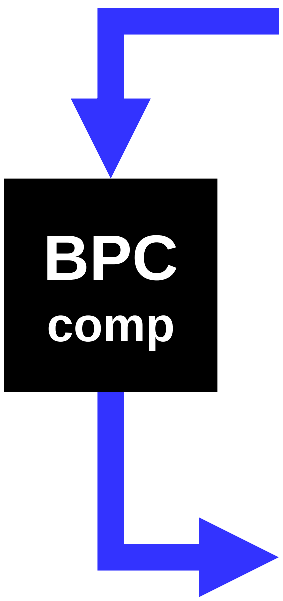


# BPC uses branch predictors to <u>model</u> a branch trace. Emits only unpredictable branches.

BPC

- Contains the branch predictors from your wildest dreams! Hurrah for software!
  - 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)
    - PAg structure

# BPC Compression




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
**< >**

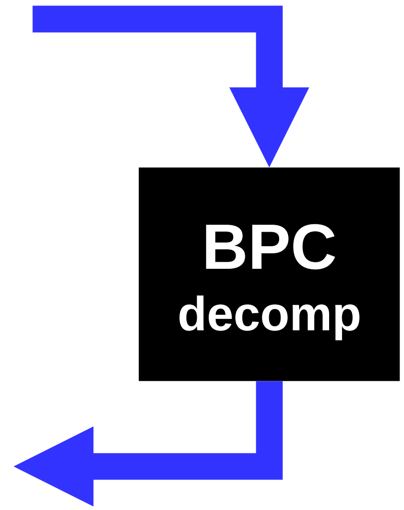
– If BPC says “I didn’t expect *that* branch record!”
**< skip N, branch record >**

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 >

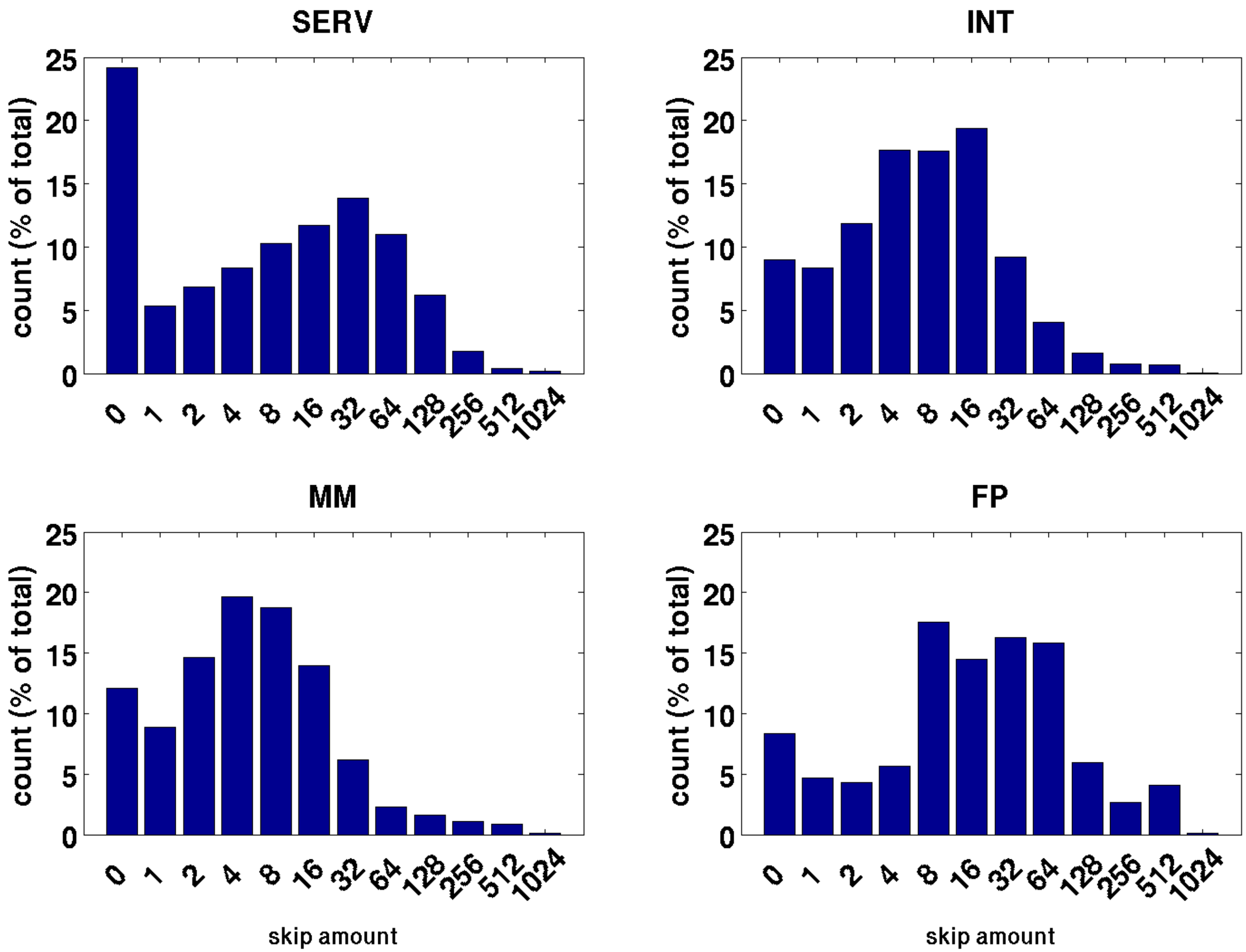



Output:

if (skip==0)
**emit branch record**
**// update predictors**

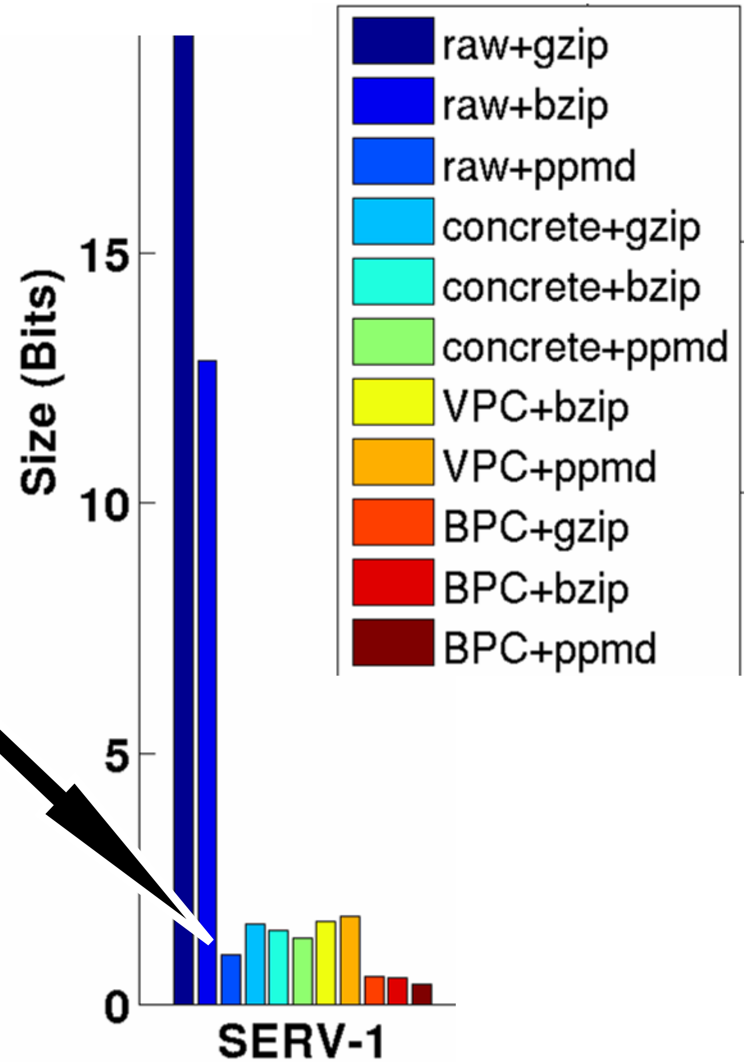
while(skip > 0)
**BPC says “let me guess!”**
**emit prediction – guaranteed correct**
**// update predictors**
**// decrement skip**

# We produce long chains of good predictions represented by single <skip, branch record>.



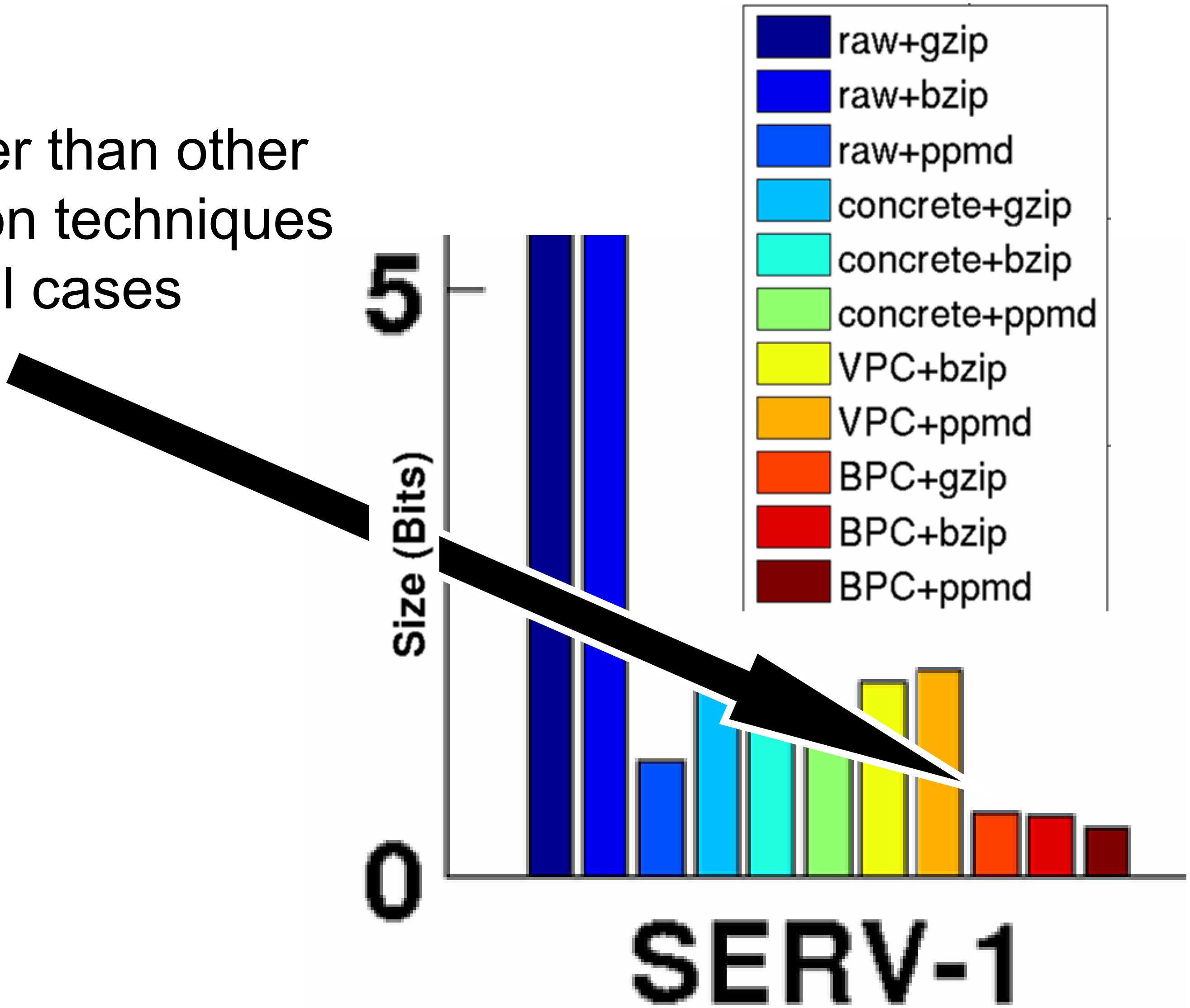
# Results: Size. BPC-compressed traces smaller than snapshots in all cases

Choose the right general-purpose technique
gzip > bzip2 > PPMd



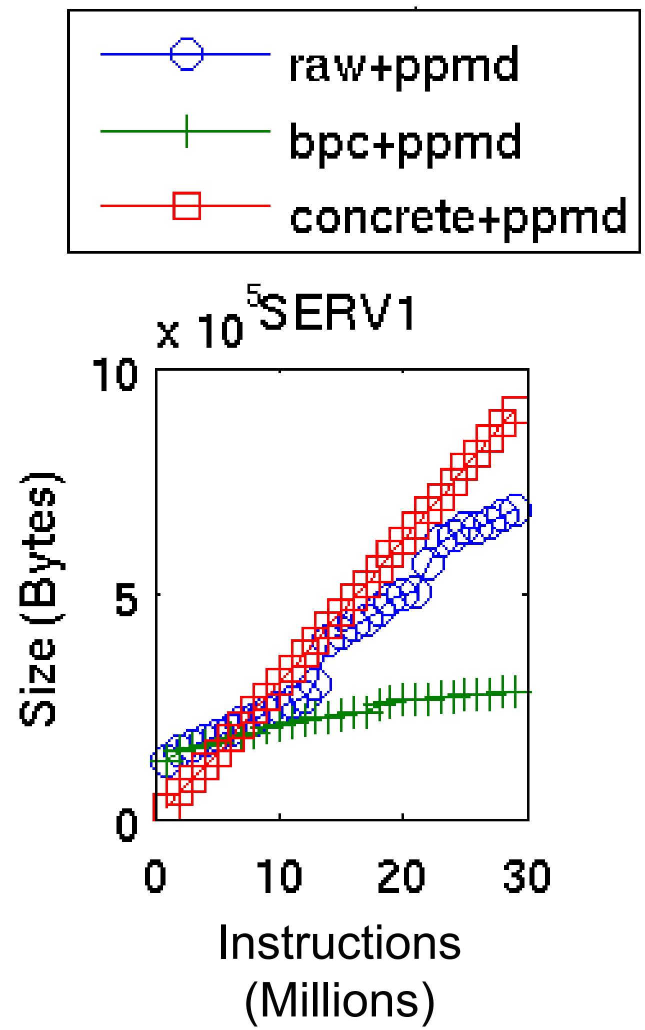
# Results: Size. BPC-compressed traces smaller than snapshots in all cases

BPC smaller than other compression techniques in almost all cases



# Results: Scaling. BPC-compressed traces grow slower than concrete snapshots

- We compare against **one** stored Pentium 4 style predictor: 2.7x smaller (avg)
- Growth
  - BPC has shallow slope, B
  - concrete scales with *mnP*
  - m=10 predictors → 27x smaller
- Example (SERV-1)
  - P=31002, B=9972, m=10, n=1000
  - 9.5 MB for BPC+ppmd
  - 295 MB for concrete snapshot+ppmd
- Both grow with number of benchmarks and cores



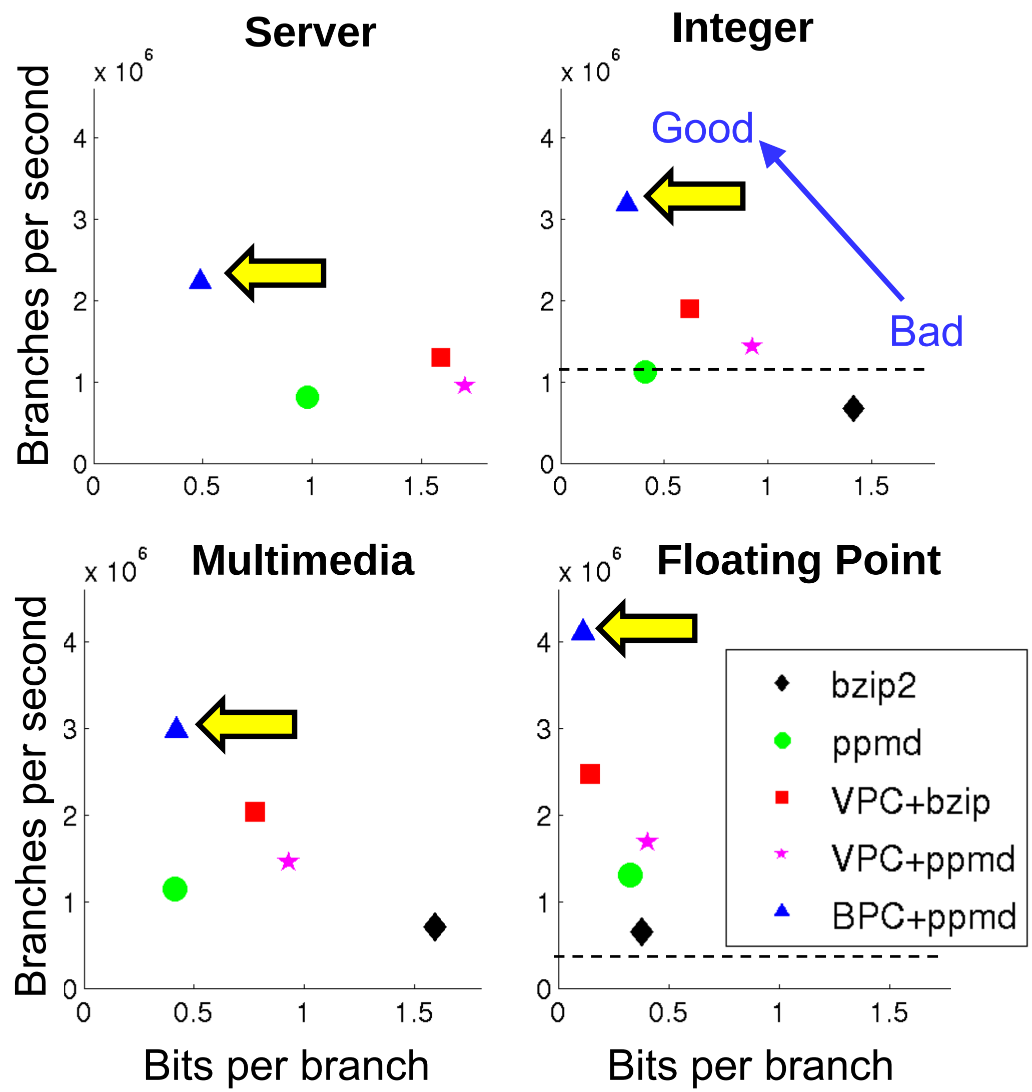
# Results: Speed. BPC is faster than other decompressors and sim-bpred

- Millions of branches/second
- Harmonic means
- 3GHz Pentium 4

| | SERV | INT | MM | FP | AVG |
|---|---|---|---|---|---|
| gzip | 7.27 | 17.71 | 15.68 | 20.23 | 13.02 |
| bzip2 | 0.79 | 0.67 | 0.71 | 0.65 | 0.70 |
| PPMd | 0.81 | 1.12 | 1.14 | 1.30 | 1.06 |
| VPC+bzip2 | 1.29 | 1.90 | 2.03 | 2.47 | 1.82 |
| VPC+PPMd | 0.95 | 1.43 | 1.46 | 1.68 | 1.32 |
| BPC+PPMd | 2.23 | 3.18 | 2.98 | 4.10 | 2.98 |
| sim-bpred | | 1.09 | | 0.34 | 0.50 |

# Summary:
# BPC compresses well and decompresses fast

- Best region: upper left
  fast and small
- BPC is faster than other decompressors
- …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.

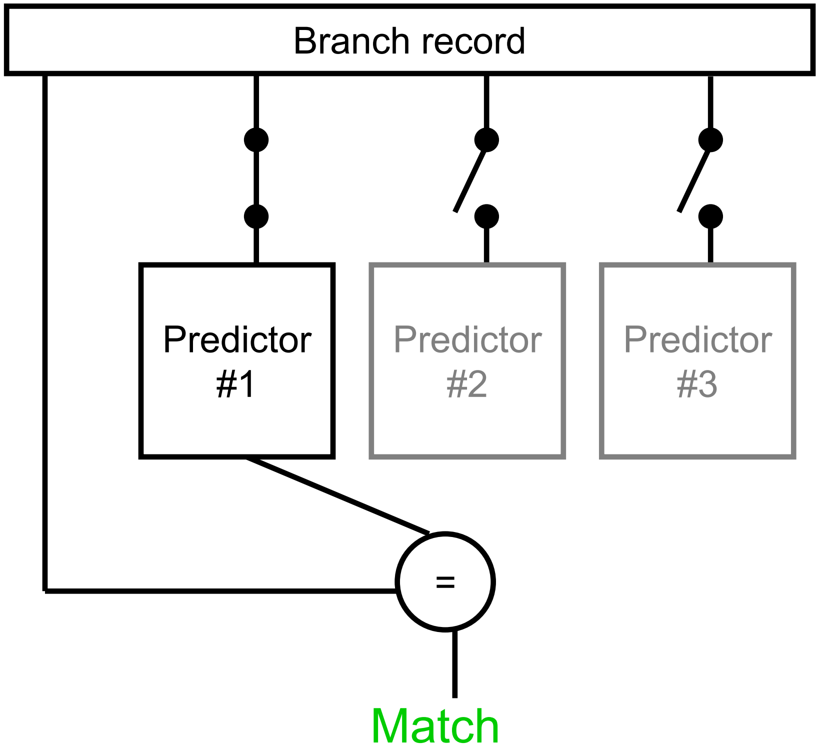
1. Value-predictor based compression (VPC)
   - Prof. Martin Burtscher at Cornell
   - Trans on Computers, Nov 2005
2. Championship Branch Prediction Contest (CBP)
   - Stark and Wilkerson, Intel
   - MICRO workshop, Jan 2005
   - Provided traces used a technique with similar spirit

Our Branch Prediction-based Compression (BPC) paper identifies application to snapshot-based simulation

- Barr and Asanović, MIT
- ISPASS, Mar 2006

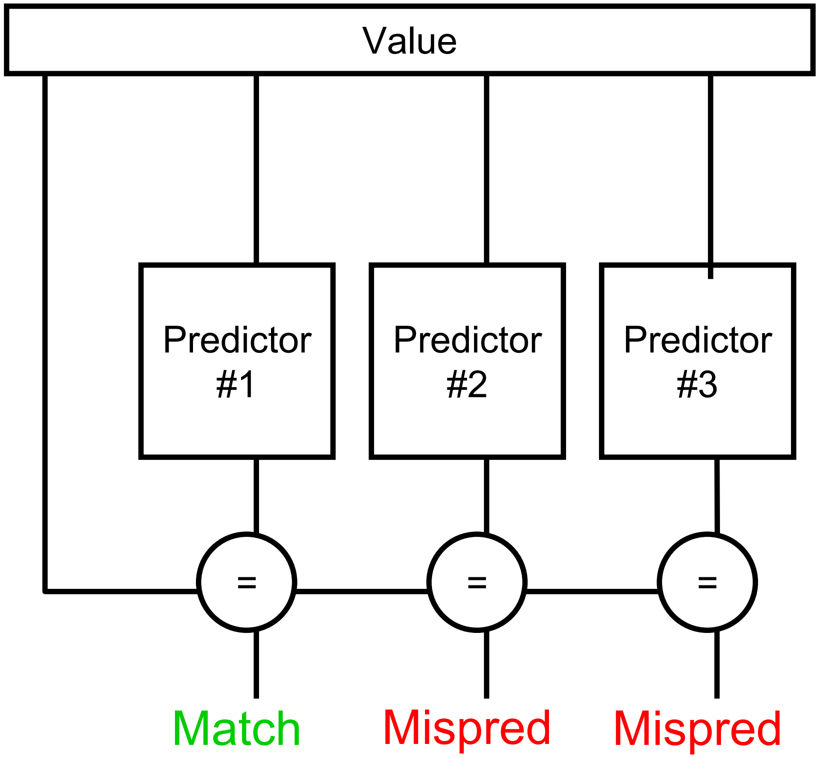
# With BPC, choice of predictor is implicitly provided, not included in output stream.

**BPC:**




Output: <>

**VPC/CBP:**




Output: <P1>

# Conclusion

- Compressed branch traces are smaller and more flexible 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.

**http://www.mit.edu/~kbarr**