

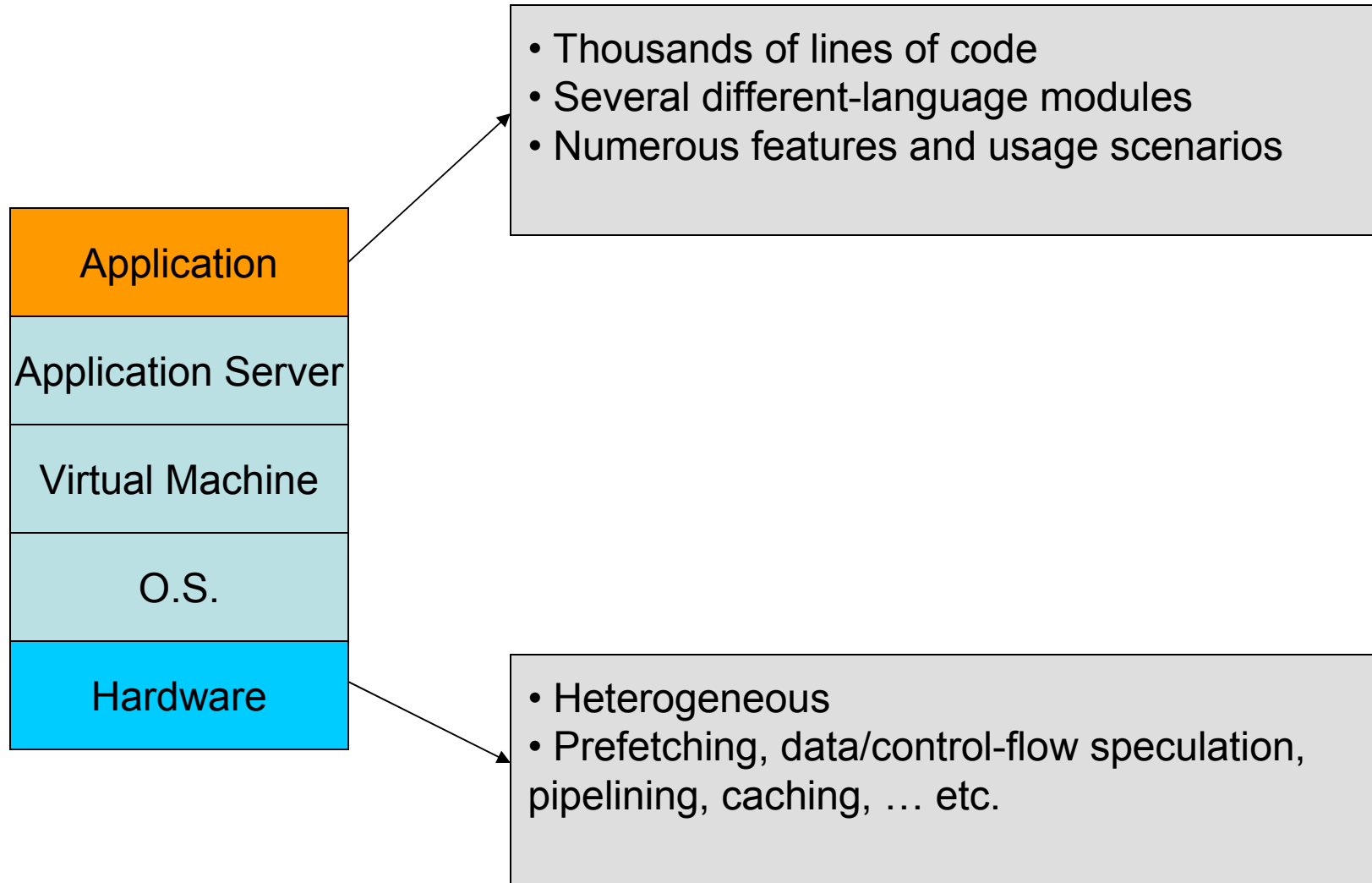
Software Performance Profiling

Major Area Presentation
Nagy Mostafa
6-5-2008

Committee Members
Chandra Krintz (chair)
Tim Sherwood
Tevfik Bultan



Motivation



Motivation

- Static analysis fails to capture runtime-behavior because
 - S/W and H/W are complex
 - S/W – H/W interaction is hard to understand
 - Application usage scenarios are hard to predict
- Solution:
Dynamic analysis based on **run-time** information

Profiling

- Profiling: investigation of program behavior using run-time information
- Profiler: conceptual module that collects/analysis run-time data
- Profile: a set of frequencies associated with run-time events
- Profile usage examples:
 - Feedback-directed optimization [Calder '99] [Burrows '00]
 - Debugging and bug-isolation [Liblit '03]
 - Coverage testing [Tikir '02]
 - Understanding program/architecture interaction [Hauswirth '04]

Profiling

- Desired qualities of a profiler:
 - Accurate
 - Low-overhead
 - Fast convergence
 - Flexible
 - Portable
 - Transparent
 - Low storage overhead

Outline

- Types of profiles
- Profiling techniques and implementations
- Hybrid profiling systems
- Software profiling systems
- Path profiling

Types of profiles

- Point profile

Events are **simple** and **independent**

e.g.

basic block, edge, method, cache-misses, ...

- Context profile

Events are composed of simpler ordered events

e.g.

call-context is a **sequence** of method invocations

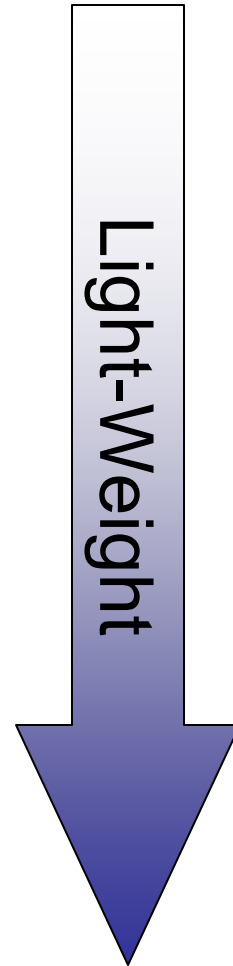
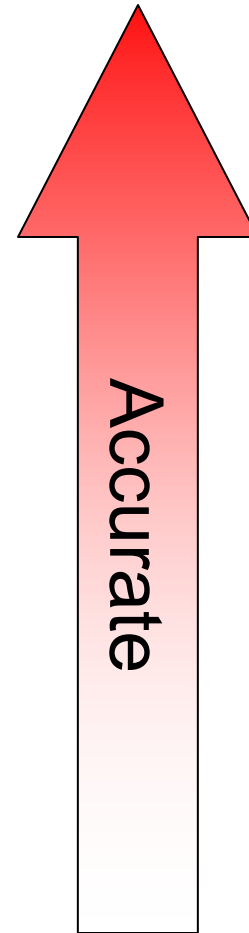
execution paths is a **sequence** of edges in a CFG

Profiling techniques

Exhaustive instrumentation

Instrumentation sampling,
temporary instrumentation, ...

Sampling



Profiler implementations

- Hybrid (HW-assisted)
 1. Hardware Performance Monitors (HPMs)
 2. Dedicated HW collectors that deliver data to SW module
 - Fixed, low-overhead
- Software
 - Pure software implementations
 - Portable, flexible, high-overhead

Outline

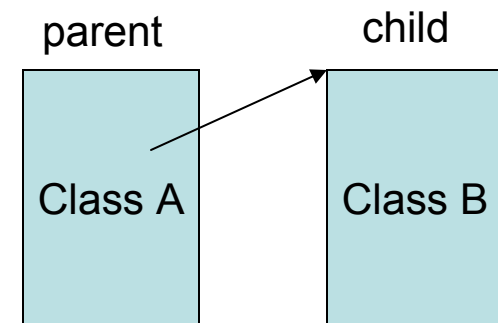
- Types of profiles
- Profiling techniques and implementations
- Hybrid profiling systems
- Software profiling systems
- Path profiling

Hybrid profiling systems

- HPM-based system-wide profilers
 - Event-based sampling
 - Support multiple events
 - Profile unmodified binaries
 - Low overhead
 - May be used in production environment
 - E.g. DCPI [Anderson '97] , OProfile, VTune

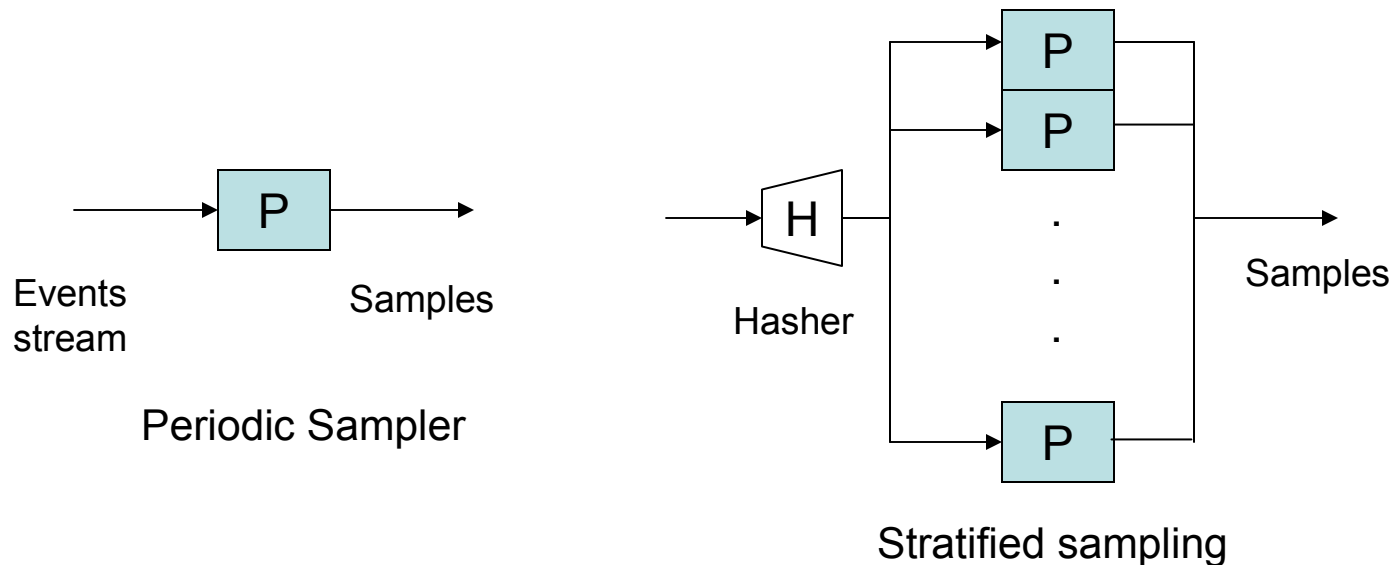
Hybrid profiling systems

- HPM-sampling for dynamic compilation [Buytaert '07]
 - HPM-sampling for JikesRVM
 - More accurate, converges faster, less overhead
 - Speed-up by 5 - 18%
- Online optimizations using HPMs [Schneider '07]
 - Cache-misses profiling
 - Co-allocation of objects
 - O.H. < 1%, speed-up 14%

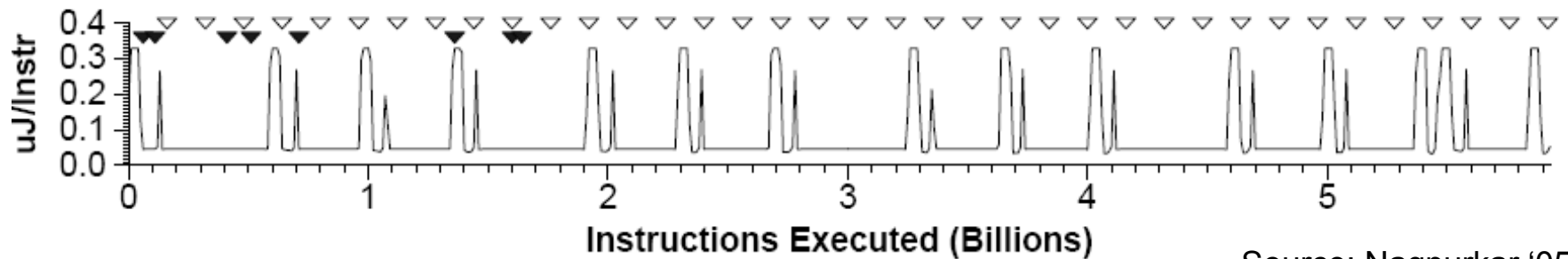


Hybrid profiling systems

- Rapid profiling via stratified sampling [Sastry '01]
 - Data stream is divided into disjoint strata
 - Reduces size of output stream and improves accuracy
 - O.H. 4.5%, accuracy 97%



Hybrid profiling systems



Source: Nagpurkar '05

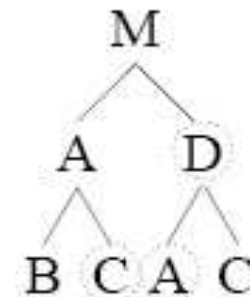
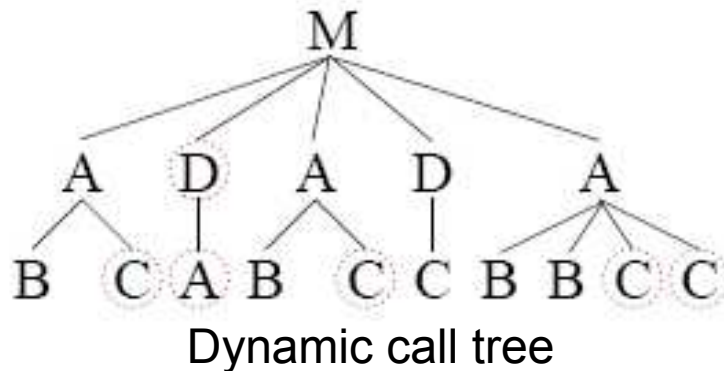
- Phase-aware profiling [Nagpurkar '05]
 - Programs exhibit repeating patterns of execution (phases)
 - Profiling a representative of each phase approximates full profile
 - Phase-change detector is in HW
 - O.H. reduction by 58% over periodic sampling, accuracy 95%

Outline

- Types of profiles
- Profiling techniques and implementations
- Hybrid profiling systems
- **Software profiling systems**
- **Path profiling**

Software profiling systems

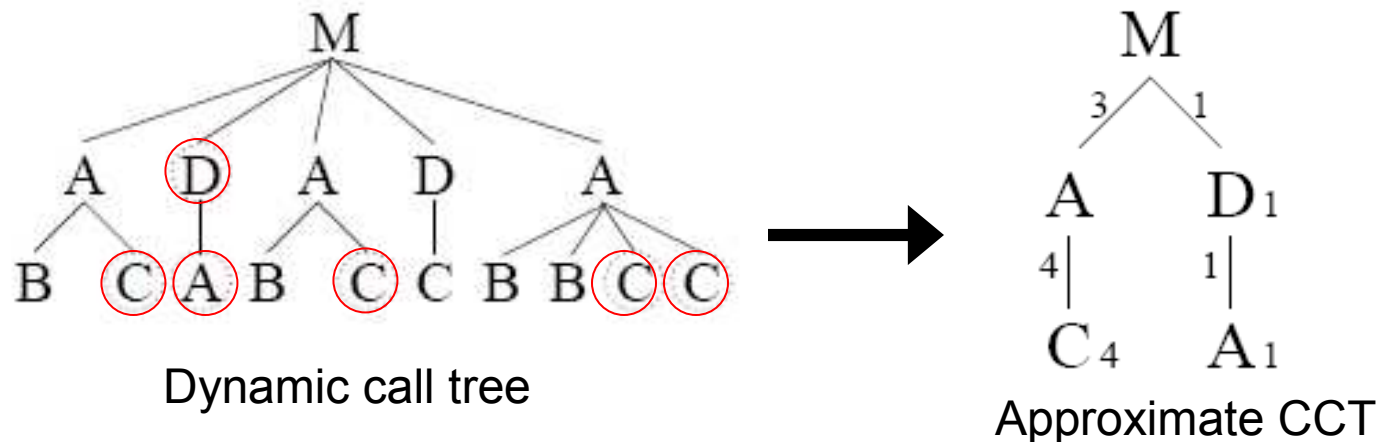
- Dynamic call tree
 - Fully describes method invocation during execution
- Calling context tree (CCT)
 - Aggregates calls with same context



Source: [Whaley '00]

Software profiling systems

- Calling-context profiling
 - Goal: build an approximate calling context tree (CCT)
 - Expensive to collect via instrumentation



Source: Whaley '00

Software profiling systems

- Sampling-based approach [Whaley '00]
 - Builds a partial call-context tree (PCCT)
 - Walks the stack on each interrupt up to k frames
 - Overhead 2-4%, Precision more than 90%
 - Disadv:
 - Precision is the correlation of PCCTs for different runs of benchmark (not accuracy)
 - Relies on O.S. timer interrupt (Low frequency, slower on faster architectures)

Software profiling systems

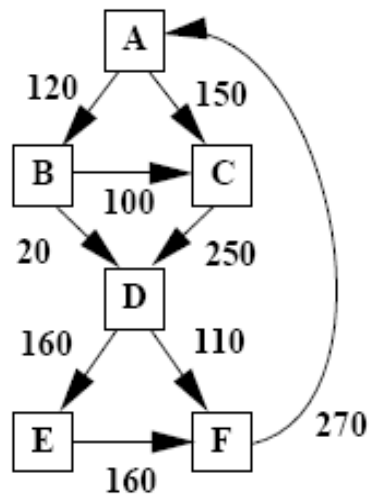
- Approximating CCT [Arnold '00]
 - Uses event-based sampling (method counter)
 - No bound on stack depth sampled
 - High accuracy, O.H. not analyzed (expected to be high)
- Timer/event-based sampling [Arnold '05]
 - Stride-enabled (mix of timer-based and event-based sampling), O.H. < 0.3%, Accuracy \approx 60%
- Probabilistic calling context (PCC) [Bond '07]
 - Usage: Residual testing, debugging, anomaly detection
 - Instrumentation-based (Sampling is not suitable)
 - No CCT is maintained \rightarrow O.H. 3%

Outline

- Types of profiles
- Profiling techniques and implementations
- Hybrid profiling systems
- Software profiling systems
- Path profiling

Path profiling

- Frequency of execution paths in CFG
- Exponential in the number of CFG nodes
- Hard to collect by sampling
- Path profile is NOT edge profile

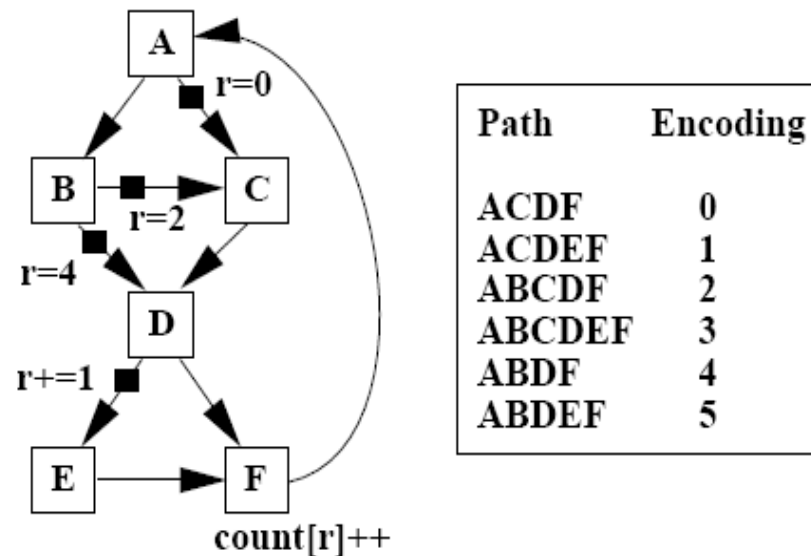


Path	Prof1	Prof2
ACDF	90	110
ACDEF	60	40
ABCDF	0	0
ABCDEF	100	100
ABDF	20	0
ABDEF	0	20

Source: Ball '96

Path profiling

- Efficient path profiling [Ball '96]
 - Compact enumeration of path (Path IDs)
 - O.H. average 31%, maximum 97%
 - Disadv: High overhead, suitable only for offline setting



Source: Ball '96

Path profiling

- Online hot path prediction scheme [Duesterwald '00]
 - Only path head is instrumented instead of full path
 - Next Executing Trail (NET) predicted as hot
 - Hit rate average: 97% when 10% of execution profiled
 - Disadv: Cannot distinguish hot from warm paths (false positives)

Path profiling

- Selective path profiling (SPP) [Apiwattanapong '02]
 - Instrument only paths of interest
 - Usage: Residual testing, profiling different subsets over multiple copies of deployed software
 - Disadv: the set of path is not totally arbitrary
- Preferential path profiling [Vasawni '07]
 - Similar to SPP but paths can be specified arbitrarily
 - O.H. average 15%

Path profiling

- Variational path profiling [Perelman '05]
 - Offline profiling scheme
 - Collects execution time variability for paths
 - Paths with high variability are good candidates for optimization
 - O.H. average: 5%, speedup via simple optimizations: 8.5%
 - Disadv: Doesn't report constantly slow paths.

Path profiling

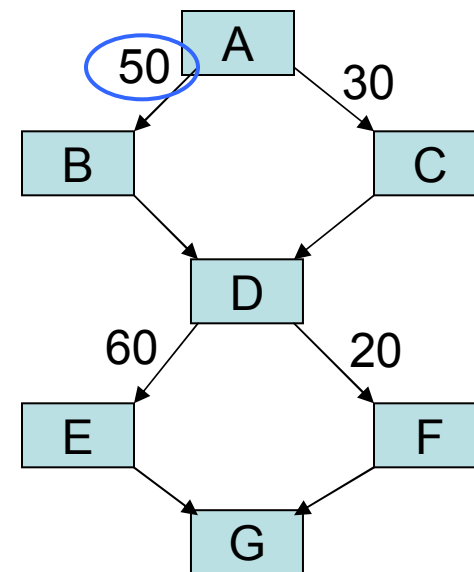
- From edge profile we can find
 - total flow (frequency)
 - upper bound on the flow of each paths
 - lower bound on the flow of each path (definite flow)

Consider ABDEG:

Max flow = 50

Let $ACDFG = 0$, $ACDEG = 30$, $ABDFG = 20$

→ Min flow = 30 (Definite Flow)

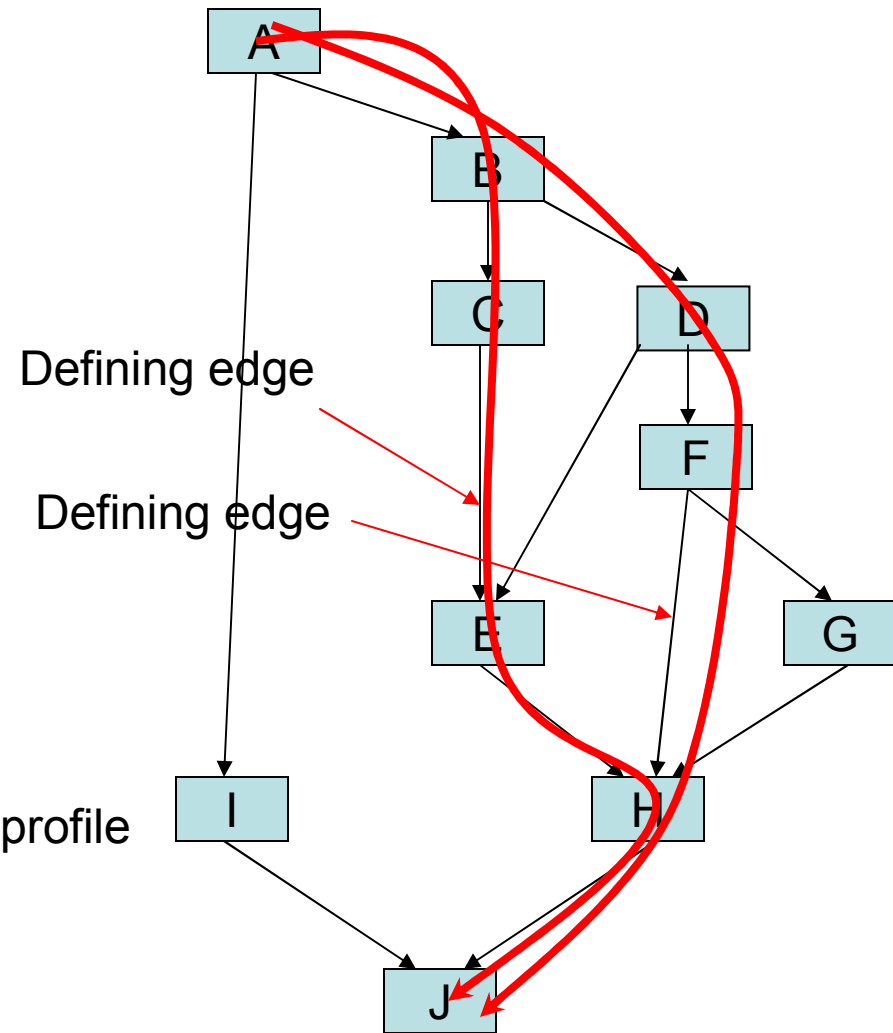


Path profiling

Defining Edge	Obvious path
AI	AIJ
CE	ABCEHJ
DE	ABDEHJ
FH	ABDFHJ
FG	ABDFGHJ

Obvious path:

Path whose flow can be driven from edge profile



Path profiling

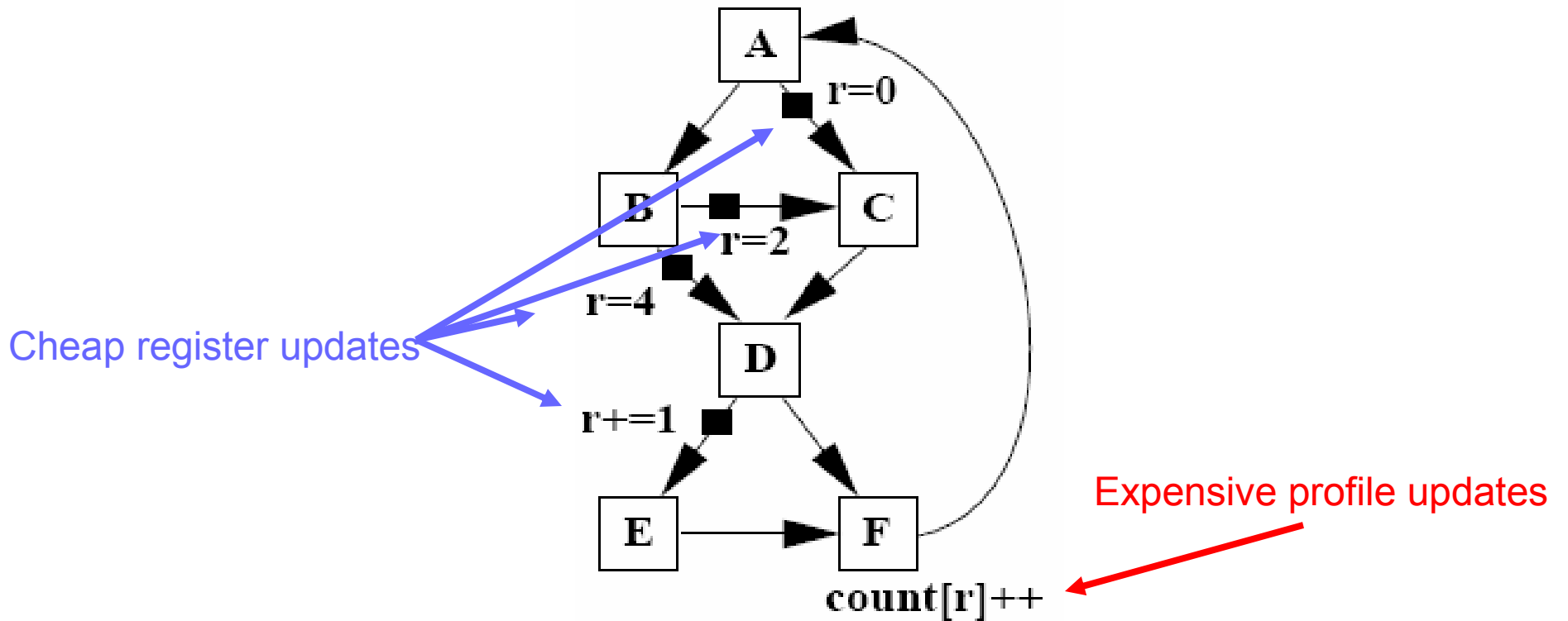
- Targeted path profiling (TPP) [Joshi '04]
 - Suitable for staged dynamic optimizers
 - Relies on edge profile to:
 - Remove obvious paths
 - Remove cold edges
 - O.H. 14%, Acc > 97%
 - Disadv: Compilation O.H. 74%

Path profiling

- Practical path profiling (PPP) [Bond '05]
 - Reduces amount of instrumentation than TPP
 - Reduce instrumentation overhead
 - Smart path numbering (avoid instrumenting hot edges)
 - O.H. average 5%, accuracy average 96%
 - Disadv: Compilation overhead not reported (expected to be high)

Path profiling

Two types of instrumentation:



Path profiling

- Path and edge profiling (PEP) [Bond'05]
 - Orthogonal approach that samples profile updates
 - Piggybacks on JikesRVM sampling profiler
 - Edge profile guides instrumentation placing
 - O.H. average 1.2%, 94% accuracy
 - Disadv: VM-specific

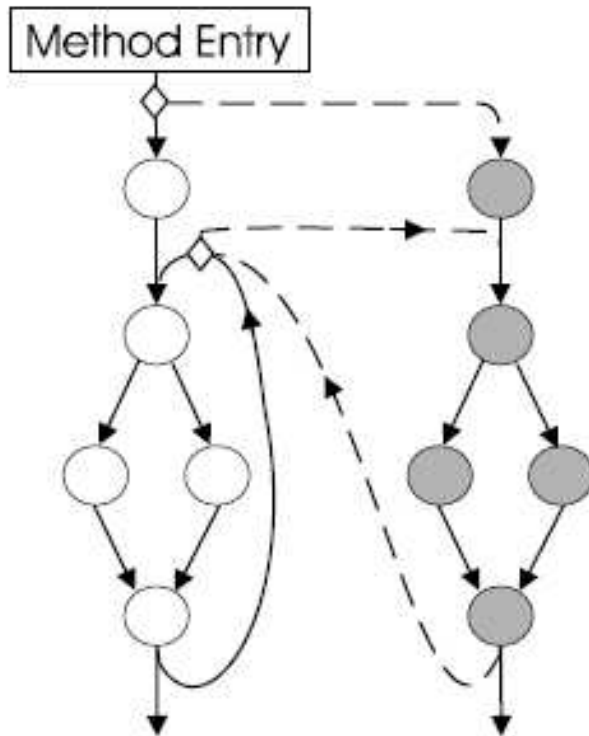
Conclusion

- Programs behavior are hard to understand statically
- Dynamic analysis based on runtime data is needed
- Performance profiling investigates runtime behavior
- Profiling techniques range from exhaustive instrumentation to sampling
- Implementation can be in hardware, software or hybrid
- Context and path profiling techniques

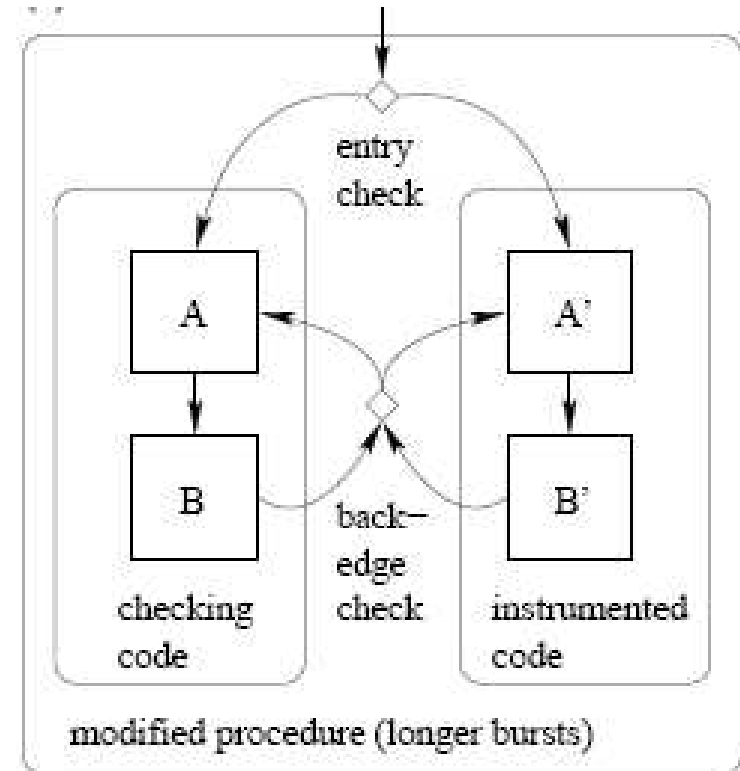
Reducing profiling cost

- Ephemeral instrumentation [Traub '00]
 - Temporary instrumentation
- Shadow profiling [Moseley '07]
 - O.H. < 1%, Accuracy 94% (value-profiling)
- Profile over adaptive ranges [Mysore '06]
 - Adaptively reducing profile storage overhead

Reducing profiling cost



Source: Arnold '01

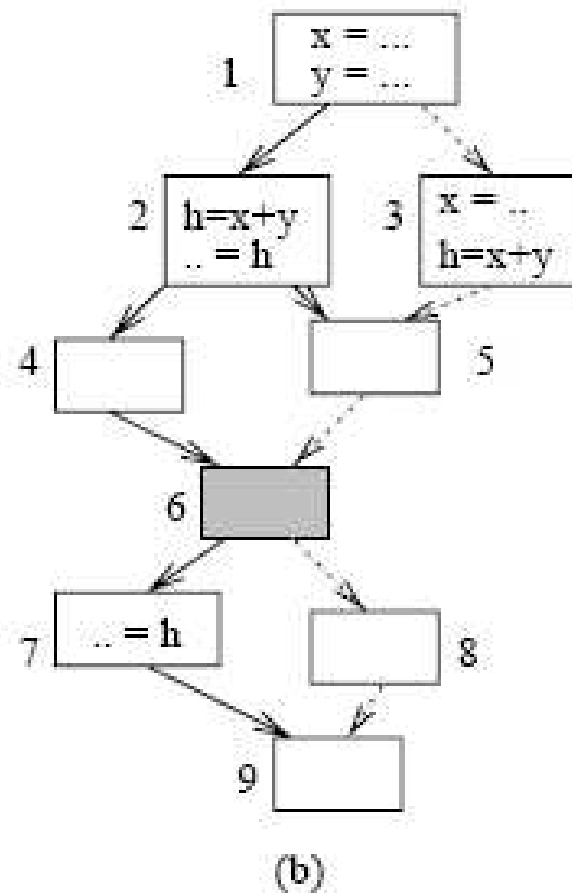
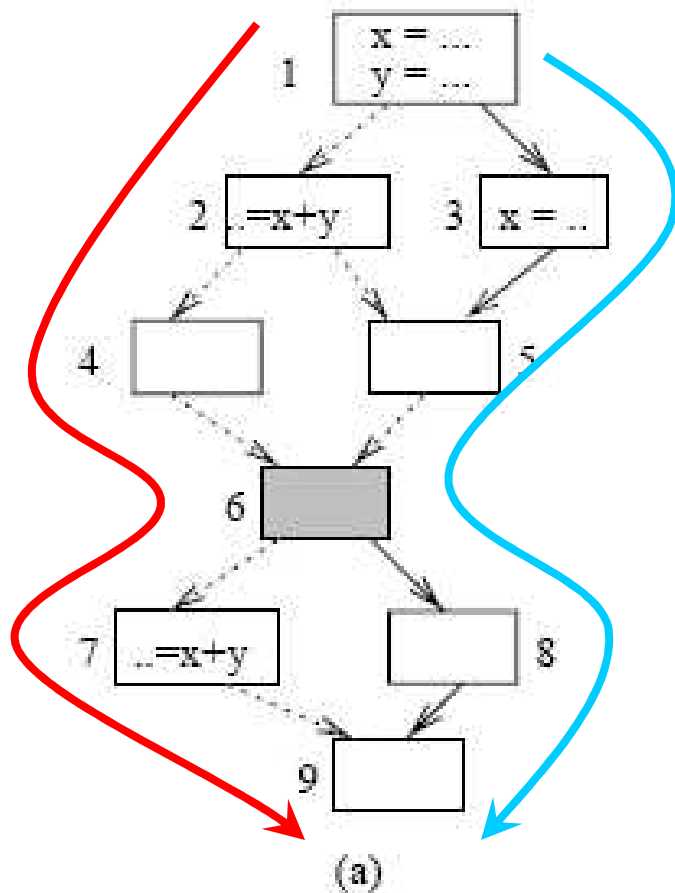


Source: Hirzel '01

Usage examples

- Value profiling and optimization [Calder '99] [Burrows '00]
- Code coverage testing [Tikir '02]
- Bug isolation [Liblit '03]
- Understand OO applications [Hauswirth '04]
- Offline/Online feed-back directed optimization

Path profiling usage example



Source: Gupta et al. 1998

Sampling: Triggering mechanism

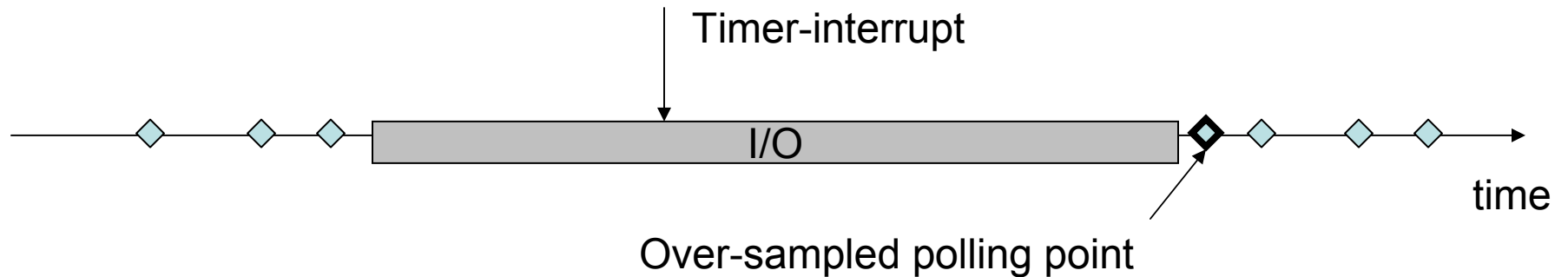
- Timer-based
 - Easy to implement, Relies on O.S. timer interrupt (e.g. 4 ms on Linux 2.6 = 250 samples/sec)
 - **Disadv.:** Low sampling frequency, independent of processor speed, cannot always correlate with program events
- Event-based
 - Relies on event counting in SW or HW (e.g. HPMs)
 - Correlates with program events
 - Adapts to processor speed

Sampling: Collection mechanism

- Polling-based
 - Samples are taken only at polling points (e.g. method entries, back-edges, ... etc.)
 - Disadv.:
 - Not timely
 - Overhead: flag checking
 - Limited accuracy: biased sampling (e.g. polling points after long I/O operations)
- Immediate
 - Sample is taken right after interrupt is raised

Sampling: Collection mechanism(2)

- Problem with polling sampling



- Arnold-Grove sampling [Arnold '05]

