Dynamic Hot Data Stream Prefetching for General-Purpose Programs

Authors:

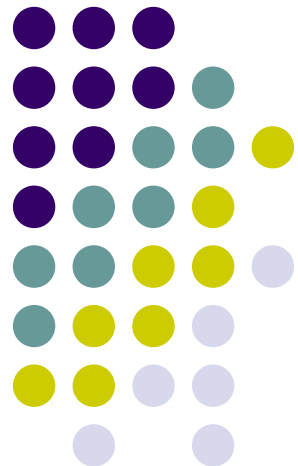
Trishul M. Chilimbi, Microsoft

Martin Hirzel, U Colorado

Presented by:

Weifeng Zhang

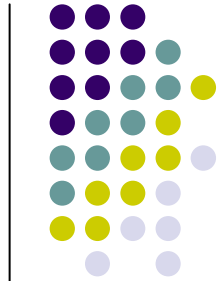
CSE 231, Fall 2003



Why prefetching?

- Tolerate the long memory latency by overlapping memory accesses with useful computations
  - Prefetch the data into cache before it is used
  - How accurate?
  - How timely?
- Static software prefetching is only successful in a limited domains
  - Relying on the compiler to predict the access pattern and insert prefetch instructions
  - Handling complex pointer-chasing?
  - What about the existing binaries?

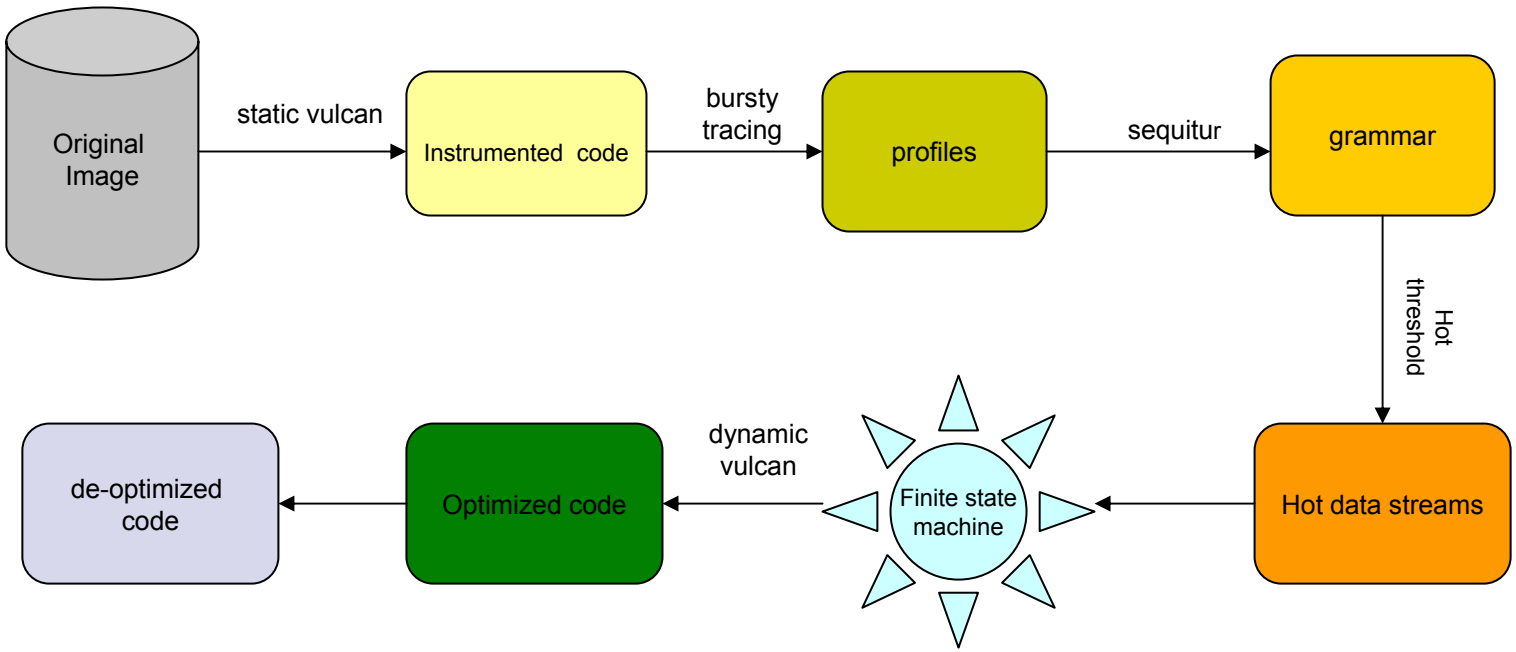
Proposed Method



- Profiling-based dynamic prefetching scheme
- Why profiling?
  - Irregular access
  - Accurate and general-purpose
- Three Phases:
  - Profiling: collection of data access traces
  - Digestion: analysis and code injection
  - Hibernation: de-optimization
    - Why?
- Efficiency:
  - Low overhead online sampling
  - Alternating on/off of phases

Proposed Method (cont)

- Dynamic prefetching flow

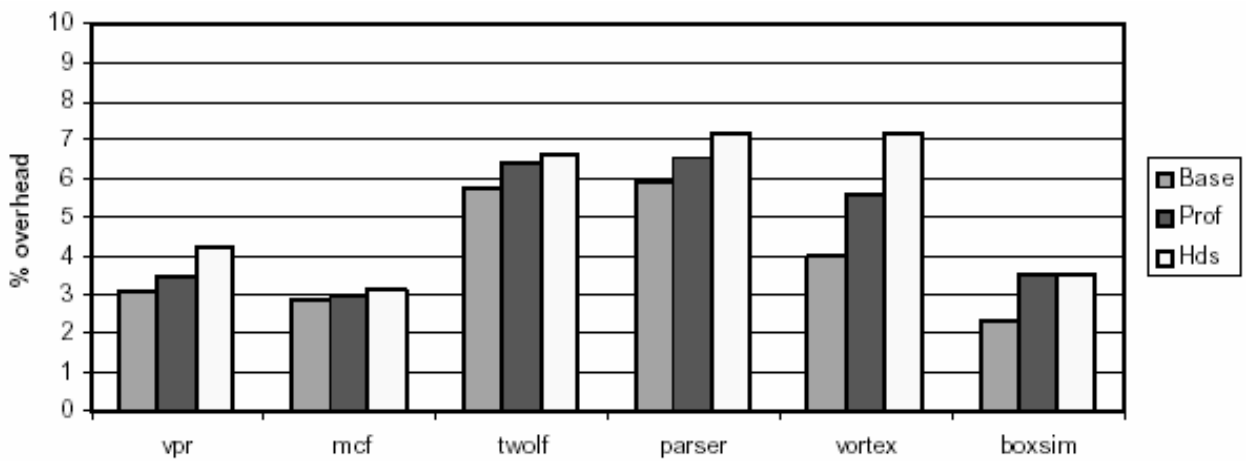


Method Details

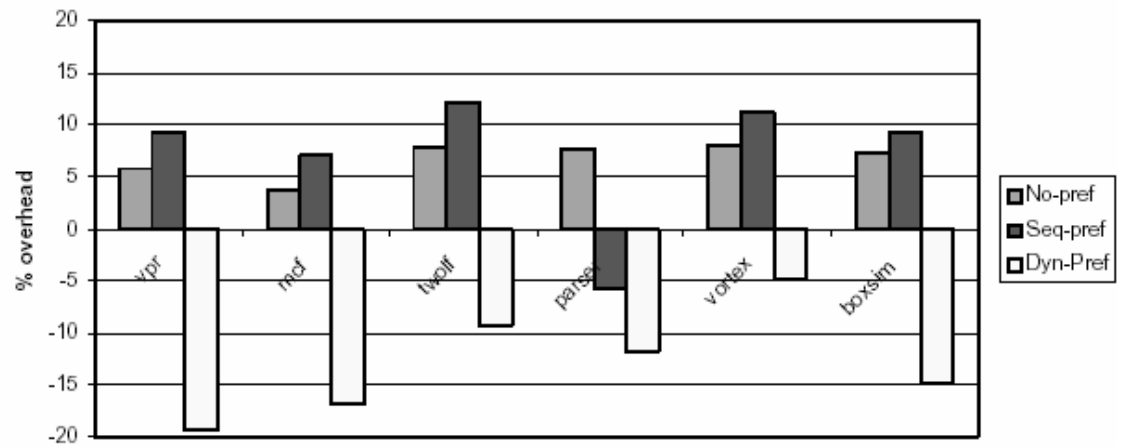
- Low overhead profiling:
  - Bursty tracing: checking code ⇔ instrumented code
  - Profiles sent to Sequitur incrementally
- Hot data stream detection:
  - Generate CFG
  - Identify streams with heat magnitude ≥ threshold
- Code injection
  - Construct DFSM for stream prefix matching
  - Inject code for prefix checking and prefetching
- De-optimization
  - Remove checks and prefetching instructions

Results

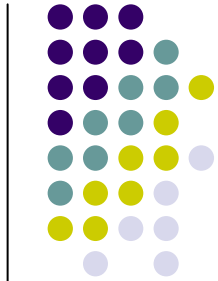
- Up to 7% overhead
- Between 5% - 19% speedup even with overhead



- Fair comparison with Seq-pref?



Questions



- Dynamo?
- Hardware prefetching mechanisms:
  - Pattern-based: stride
    - Local
    - Global
  - Predictor-based stream buffer
  - Speculative pre-computation with SMT