

Static Prefetch Insertion via Program Analysis

Course Project – CS6843: Program Analysis

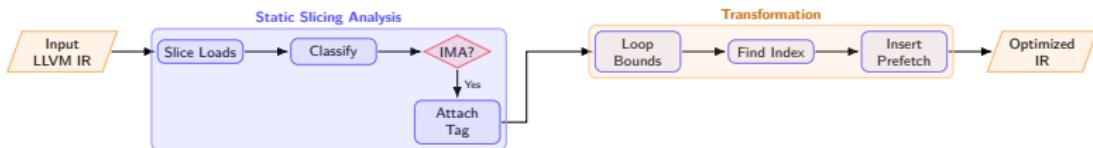
Rahul Utkoor (CS25D005)

Department of Computer Science and Engineering
Indian Institute of Technology Madras (**IITM**)

Instructor: Prof. Rupesh Nasre

A static analysis-driven approach for identifying and prefetching indirect memory accesses.

Overall Workflow: Static Prefetch Insertion Pipeline



Two phase pipeline: (i) Static slicing and classification (ii) Guarded prefetch generation

LLVM IR Snapshot: Before vs After Transformation

Before Transformation

```
; Indirect access in loop  
%i = phi i64 [0, %entry],  
[%i.next, %loop]  
%idx = load i32, ptr %A[%i]  
%val = load float, ptr %B[%idx]  
br label %loop
```

After Transformation

```
; Guarded prefetches inserted  
%cond = and i1 %deg, %range  
br i1 %cond, label %pf.then, label %pf.cont  
pf.then:  
%idx = load i32, ptr %A[%i+d]  
call void llvm.prefetch.p0(ptr %B[%idx])  
br label %pf.cont
```

Prefetch calls are inserted before the loop body and guarded by runtime range checks.

Pattern-specific Prefetch Transformations

Vector Pattern

```
%p1 = load ptr, ptr @A, align 8  
%idx = call @_ZNSt6vectorIiSaIiEEixEm(%p1,)  
%p2 = getelementptr ... , i64 %idx  
%val = load float, ptr %p2, align 4  
call llvm.prefetch.p0(...)
```

Array Pattern

```
%p1 = load ptr, ptr @A, align 8  
%p2 = getelementptr ... , ptr %p1, i64 %iv  
%idx = load i32, ptr %p2, align 4  
%p3 = getelementptr ... , ptr @B, i64 %idx  
%val = load float, ptr %p3, align 4  
call llvm.prefetch.p0(...)
```



Prefetching Impact on PageRank Performance

Key Observations:

- ▶ Positive $\Delta \Rightarrow$ improvement.
- ▶ **web-Google** and **LiveJournal** show slight execution-time gains ($\approx 0.7\%$ and 0.08% respectively).
- ▶ Other datasets experience marginal slowdowns ($< 2\%$).
- ▶ L1 cache misses reduce significantly for LiveJournal (+8.8%) and Slashdot0811 (+17.6%).
- ▶ Cache behavior worsens for web-Google (-75%) and wiki-Vote (-454%), showing prefetch inefficiency.

Execution Time (s)

Dataset	Base	Pref.	Δ (%)
soc-LiveJournal1	36.71	36.68	+0.08
soc-Slashdot0811	0.25	0.254	-1.60
soc-Slashdot0902	0.268	0.270	-0.75
web-Google	2.93	2.91	+0.68
web-Stanford	2.18	2.21	-1.38
wiki-Vote	0.009	0.0091	-1.11

L1 Cache Miss (%)

Dataset	Base	Pref.	Δ (%)
soc-LiveJournal1	3.05	2.78	+8.85
soc-Slashdot0811	3.36	2.77	+17.56
soc-Slashdot0902	3.497	3.47	+0.77
web-Google	1.95	3.42	-75.38
web-Stanford	3.08	3.51	-13.96
wiki-Vote	0.50	2.77	-454.00