Sampling-Based Locality Profiling

Ian Christopher and Chen Ding
University of Rochester

Outline

1) Brief motivation and definitions of terms
2) Discussion of sampling methods
3) Locality analysis techniques
4) Introduction to the implementation (SLO-R) and experimental results
5) Conclusion

Motivation

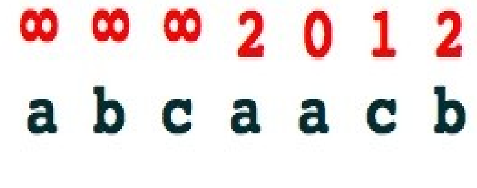
- Locality, both temporal and spatial, has a dramatic effect on prefetching and caching.
- Profiling can point out pieces with poor locality in code or serve as a larger metric for the entire program itself.
- Full trace analysis too slow for most situations, so sampling is commonly necessary.

Definition of Terms

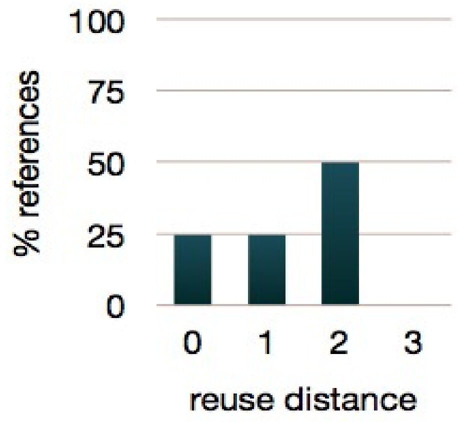
- Reuse distance - the number of unique memory locations referenced between a memory address' use and reuse.
- Reuse signature - histogram of reuse distances.

- Temporal locality can be judged entirely by reuse distances.

- Spatial locality is an equally simple idea but is more complex than temporal locality. It has a few different constituents – inter-block and adjacent-block



(a) reuse distances



(b) reuse signature

Outline

1) Brief motivation and definitions of terms
2) ***Discussion of sampling methods***
3) Locality analysis techniques
4) Introduction to the implementation (SLO-R) and experimental results
5) Conclusion

Sampling Introduction

- Ideally, sampling provides a representative portion of a program.
- Sampling creates a balance between accuracy and execution runtime.
- A full trace based analysis commonly causes a slow down with a factor somewhere in the hundreds.
- Good sampling is difficult.

Types of Samples

- Temporal and spatial locality scores depend on very different access trace measurements.
- Temporal locality can be found using just an approximated reuse distance.
- Spatial locality requires a broader analysis of an access. The access stream around it must be considered.
- The size of these spatial samples is not well defined.

Naive Methods

- Taking every n-th possible sample seen.
- Sampling based on what locations in the code you have already sampled.
- Sampling based on how many samples you have already collected.

Reservoir Sampling

- Consider the following:
  - You have a stream of running water in front of you. You are not sure how long it will remain running and want to collect N representative samples before it stops. What do you do?
  - Reservoir sampling takes its name from this problem.

Reservoir Sampling (continued)

- There are two forms of answers

1) Assign each sample seen a random number. Keep the highest N of them in a reservoir of samples.
2) Use statistical methods to predict how many samples you should throw out before you insert the next in to the reservoir. Prediction is dependent on how many samples you have collected in the past.

Reservoir Sampling Continued

- Algorithm L from "Reservoir-Sampling Algorithms of Time Complexity O(n(1+log(N/n)))" by Kim-Hung Li
  1) [Initialize] Initialize the reservoir Xl,. . . . Xn. to be the first n samples, Set W := exp(log(random( ))/n).
  2) [Generate S] Set S := [log(random( ))/log(l – W)].
  3) [Search for the next potential record] Search for the next (S + l)th sample, say Y. If the file is exhausted before the record is found, deliver Xl, ... Xn. and terminate
  4) [Update X and W] Replace Xrandom, by Y. Replace W by W “exp(log(random( ))\n). Go to step (2).

- Found in : ACM Transactions on Mathematical Software, vol. 20, No. 4, December 1994

Reservoir Sampling Continued

- The last algorithm was very fast with respect to most other sampling algorithms.
- Still have keep the reservoir in memory. The number of samples that can be collected becomes an issue.

Reservoir Locality Analysis

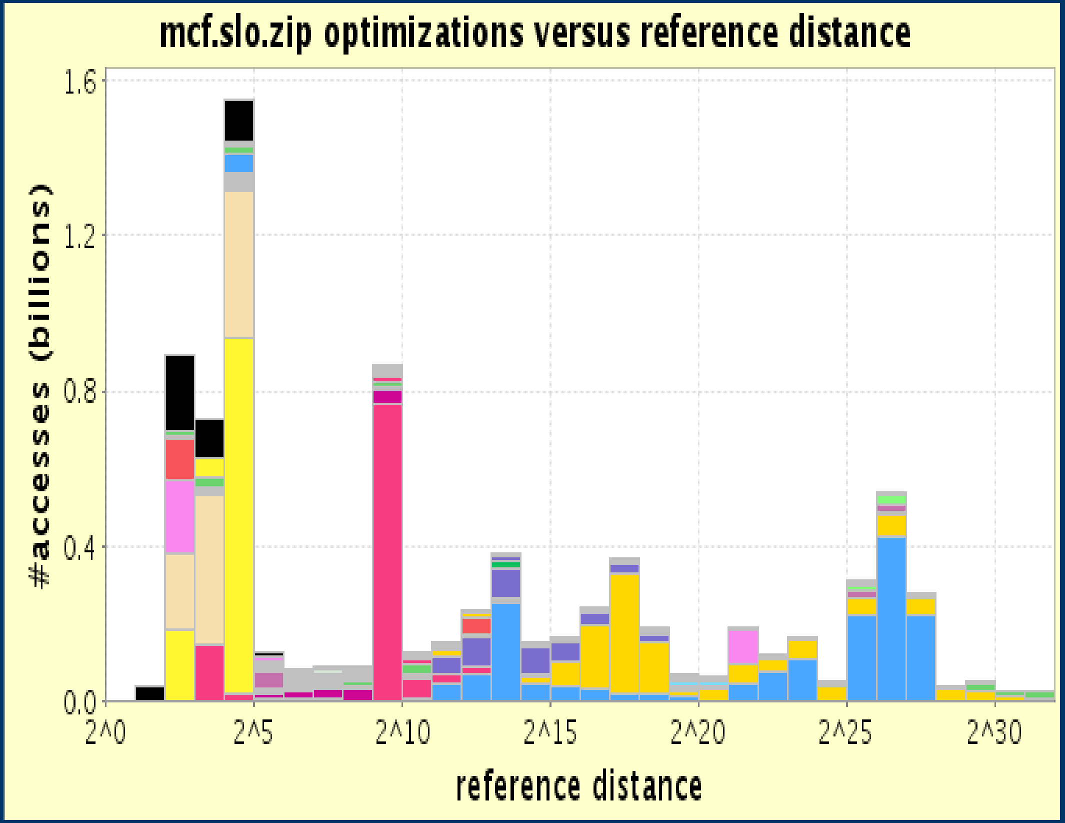
- Now that we have the reservoir, how do we assign locality scores?
- Are there types of analysis that are favorable to this type of sampling? Unfavorable?

Outline

1) Brief motivation and definitions of terms
2) Discussion of sampling methods
3) ***Locality analysis techniques***
4) Introduction to the implementation (SLO-R) and experimental results
5) Conclusion

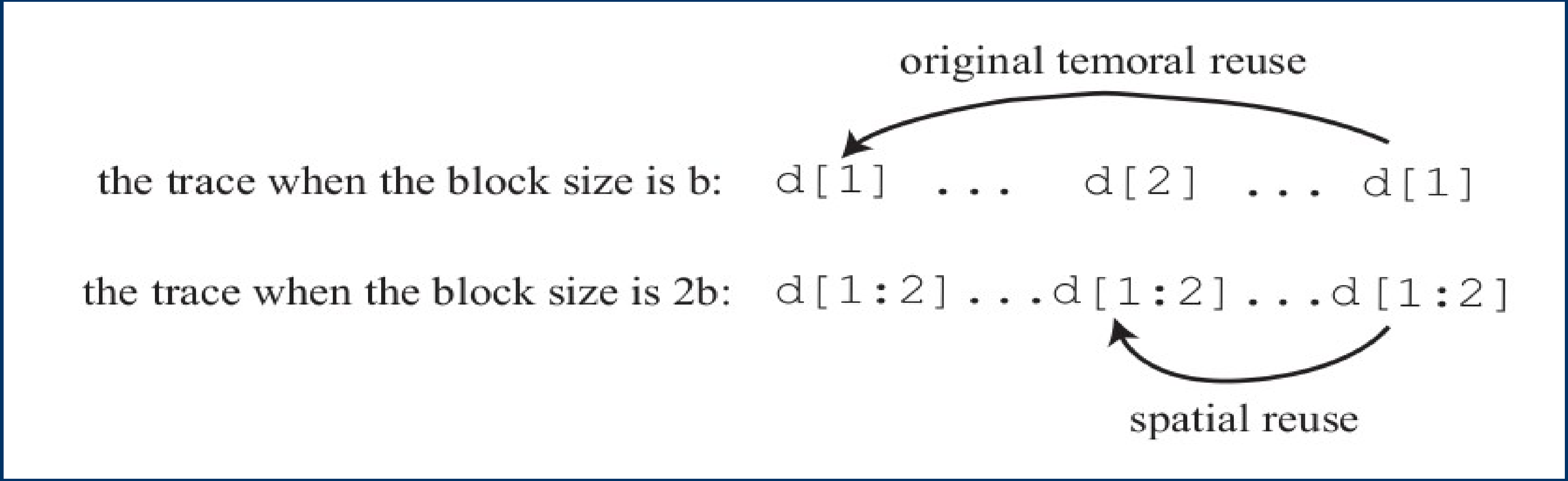
Temporal Analysis

- Temporal samples are relatively straightforward.
- Samples approximate reuse distance with time.
- Defining a metric for reuse distances gives an overall metric for the reservoir.



Spatial Analysis

- A new type of analysis called Intra-block Spatial Reuse Analysis.
- Suppose elements x and y of size b belong to the same 2b block.
  - The original temporal reuses distance is reduced in two ways : larger block sizes and so-called intercepts.



Spatial Analysis Continued

- At a high level, the analysis is a metric on how much the distances changed in the two block size.
- Short original temporal reuses are not analyzed.
- The block sizes where the L1 and L2 cache sizes respectively.
- Full analysis for every reuse is far too costly, so statistical approximations are used.
- Analysis was used to improve a NLP program's runtime roughly seven percent with six lines of code improvement.

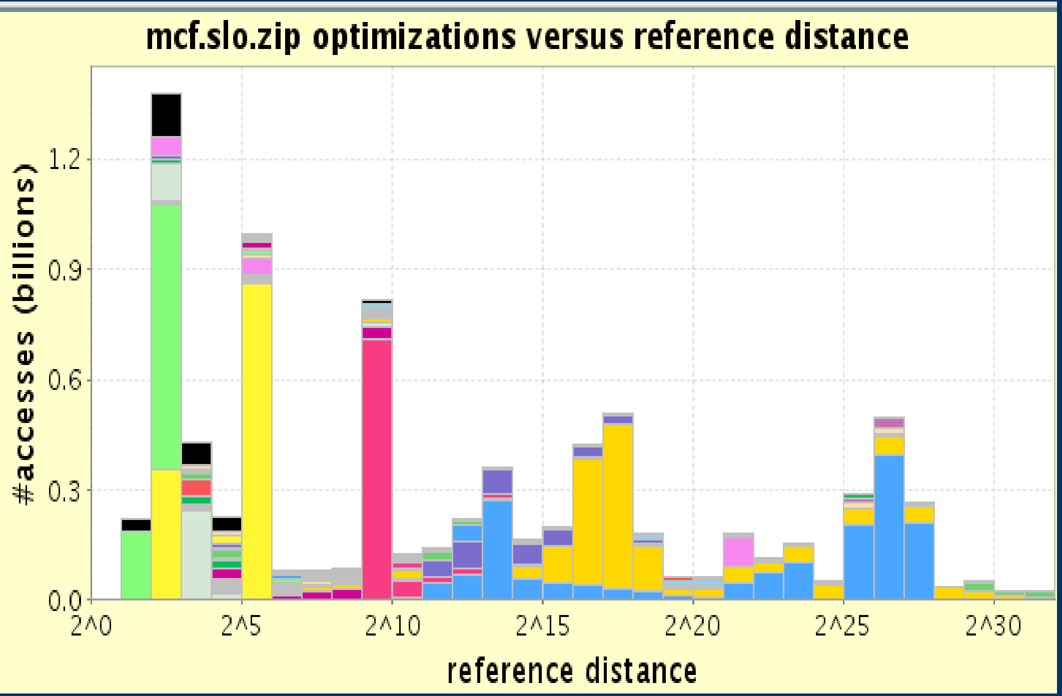
Outline

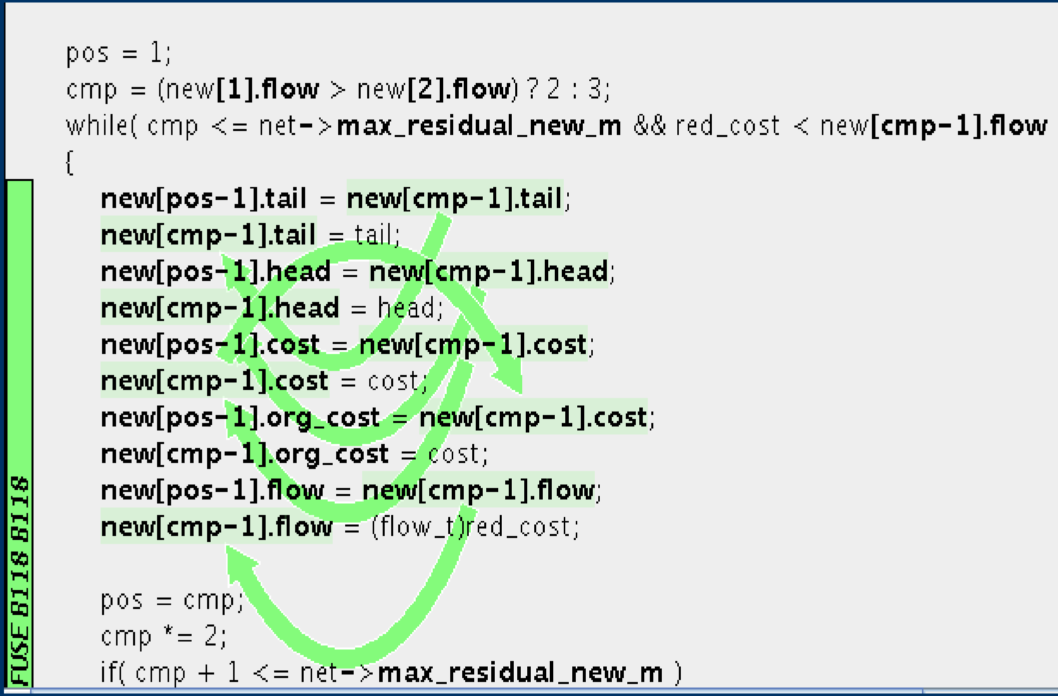
1) Brief motivation and definitions of terms
2) Discussion of sampling methods
3) Locality analysis techniques
4) ***Introduction to the implementation (SLO-R) and experimental results***
5) Conclusion

The Implementation - SLO-R

- SLO-R is a platform-independent locality analysis tool.
- It is an extension of Suggestions for Locality Optimizations (SLO), which will be presented in the next slide. Note: the "R" in SLO-R denotes "Rochester".
- It provides the programmer direct feedback on which parts of code have poor locality and gives suggestions on how to fix them.
- Through GCC, it inserts calls to an external locality library into a compiled program. During execution, result data files are created and analyzed by a Java back end.

SLO – Suggestions for Locality Optimizations





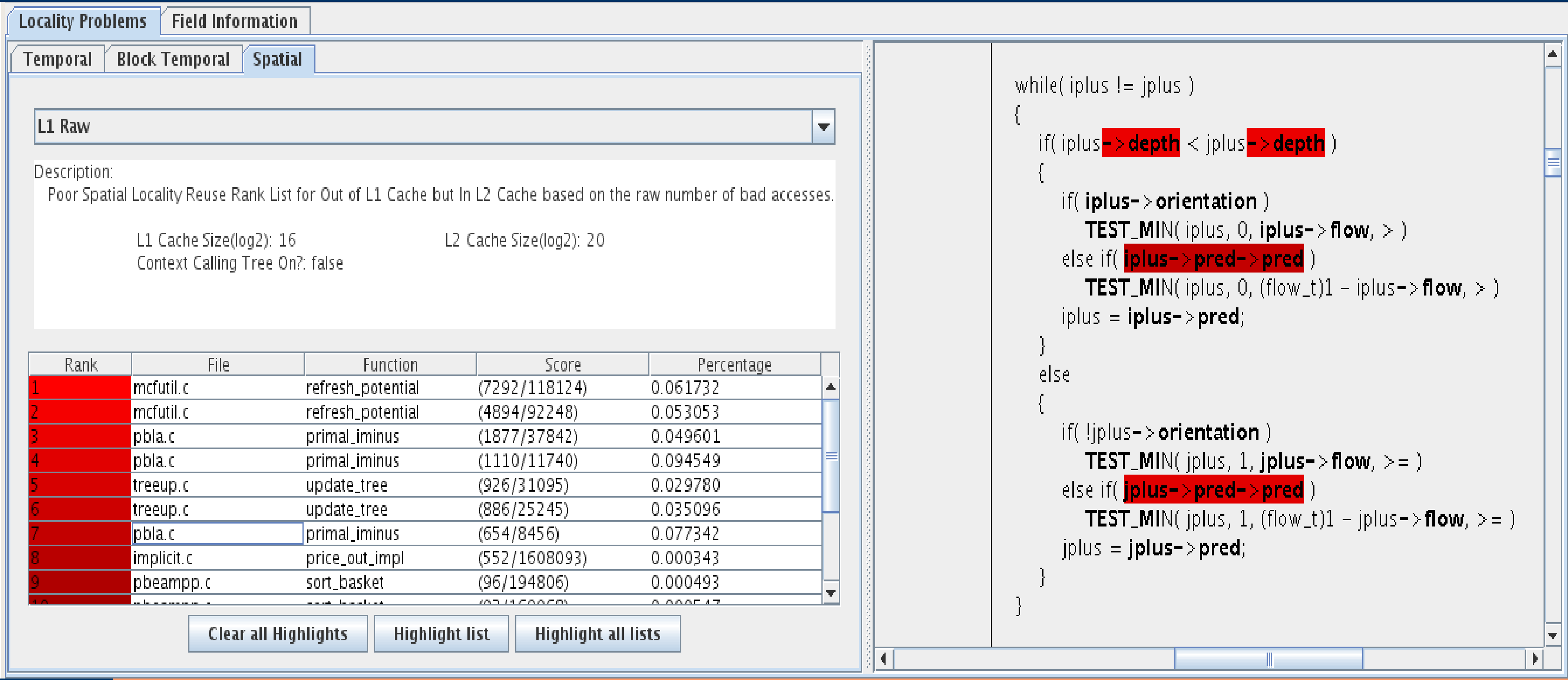
```
pos = 1;
cmp = (new[1].flow > new[2].flow) ? 2 : 3;
while( cmp <= net->max_residual_new_m && red_cost < new[cmp-1].flow
{
    new[pos-1].tail = new[cmp-1].tail;
    new[cmp-1].tail = tail;
    new[pos-1].head = new[cmp-1].head;
    new[cmp-1].head = head;
    new[pos-1].cost = new[cmp-1].cost;
    new[cmp-1].cost = cost;
    new[pos-1].org_cost = new[cmp-1].cost;
    new[cmp-1].org_cost = cost;
    new[pos-1].flow = new[cmp-1].flow;
    new[cmp-1].flow = (flow_t)red_cost;

    pos = cmp;
    cmp *= 2;
    if( cmp + 1 <= net->max_residual_new_m )
```

- Developed by Kristof Beyls and Erik H. D'Hollander at the University of Ghent.
- Deserve the utmost credit – we built on their open source code.
- Collects temporal locality samples through GCC4.1.1 and provides feedback where the worst sections of code are.
- Hosted by Sourceforge

SLO-R vs SLO

- SLO-R has spatial locality considerations and block temporal locality scores.
- SLO-R's spatial reservoir also allowed reference affinities to be estimated.

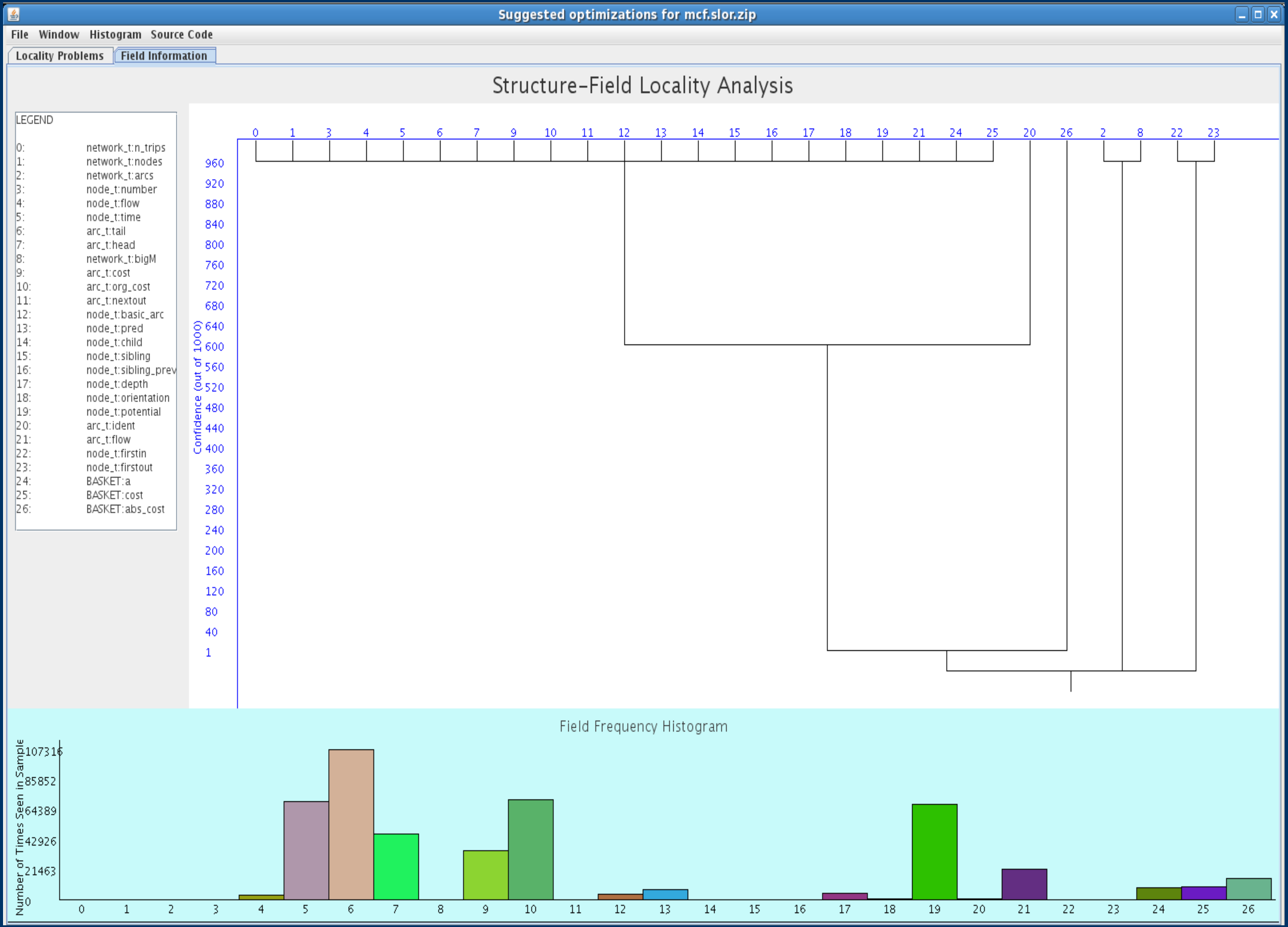


Description:
Poor Spatial Locality Reuse Rank List for Out of L1 Cache but In L2 Cache based on the raw number of bad accesses.

L1 Cache Size(log2): 16 L2 Cache Size(log2): 20
Context Calling Tree On?: false

Rank	File	Function	Score	Percentage
1	mcfutil.c	refresh_potential	(7292/118124)	0.061732
2	mcfutil.c	refresh_potential	(4894/92248)	0.053053
3	pbla.c	primal_iminus	(1877/37842)	0.049601
4	pbla.c	primal_iminus	(1110/11740)	0.094549
5	treeup.c	update_tree	(926/31095)	0.029780
6	treeup.c	update_tree	(886/25245)	0.035096
7	pbla.c	primal_iminus	(654/8456)	0.077342
8	implicit.c	price_out_impl	(552/1608093)	0.000343
9	pbeampp.c	sort_basket	(96/194806)	0.000493

```
while( iplus != jplus )
{
    if( iplus->depth < jplus->depth )
    {
        if( iplus->orientation )
            TEST_MIN( iplus, 0, iplus->flow, > )
        else if( iplus->pred->pred )
            TEST_MIN( iplus, 0, (flow_t)1 - iplus->flow, > )
        iplus = iplus->pred;
    }
    else
    {
        if( !jplus->orientation )
            TEST_MIN( jplus, 1, jplus->flow, >= )
        else if( jplus->pred->pred )
            TEST_MIN( jplus, 1, (flow_t)1 - jplus->flow, >= )
        jplus = jplus->pred;
    }
}
```



Suggested optimizations for mcf.slor.zip

File Window Histogram Source Code

Locality Problems | Field Information

Structure-Field Locality Analysis

LEGEND

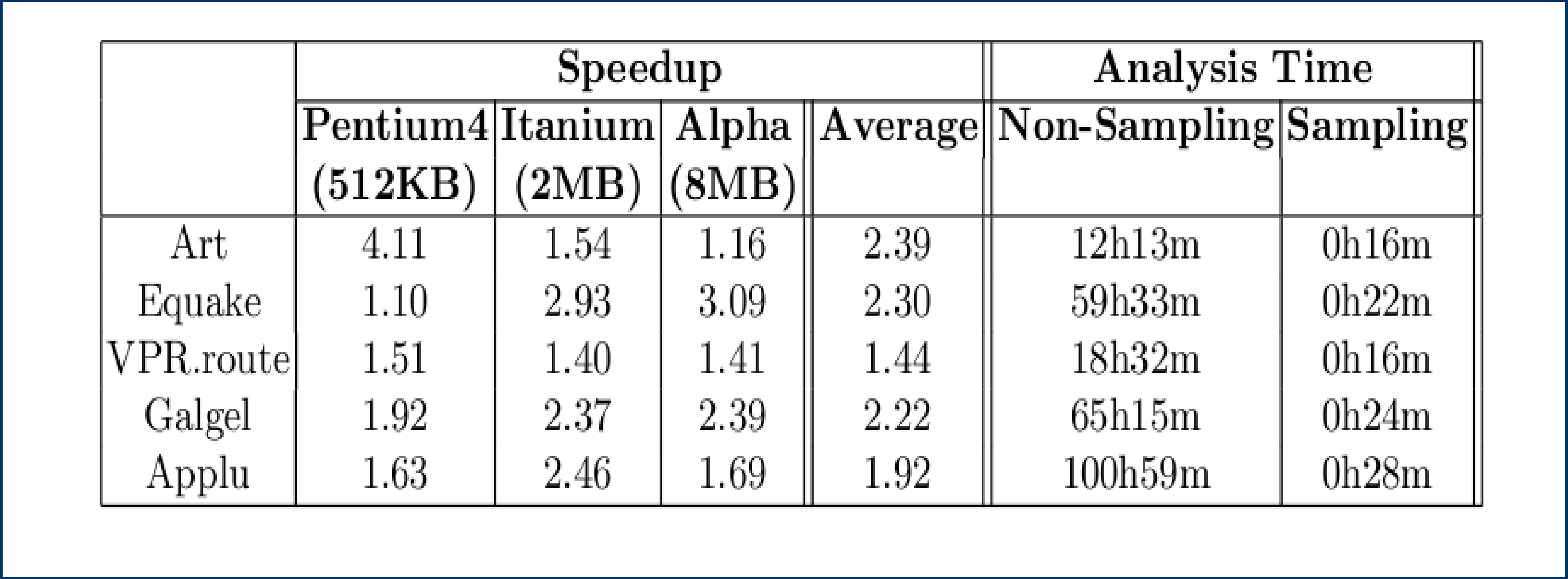
0:	network_t:n_trips
1:	network_t:nodes
2:	network_t:arcs
3:	node_t:number
4:	node_t:flow
5:	node_t:time
6:	arc_t:tail
7:	arc_t:head
8:	network_t:bigM
9:	arc_t:cost
10:	arc_t:org_cost
11:	arc_t:nextout
12:	node_t:basic_arc
13:	node_t:pred
14:	node_t:child
15:	node_t:sibling
16:	node_t:sibling_prev
17:	node_t:depth
18:	node_t:orientation
19:	node_t:potential
20:	arc_t:ident
21:	arc_t:flow
22:	node_t:firstin
23:	node_t:firstout
24:	BASKET:a
25:	BASKET:cost
26:	BASKET:abs_cost

SLO-R results

- Sampling created a slow down of about thirty five. This figure includes a lot of extra execution.
  - Three reservoirs, reference affinity analysis, both types of locality analysis, writing large data files, etc.
- Original slow down was well over three hundred.
- As for code improvement (a metric of sampling quality), there has not been enough testing.
- Small spatial samples resulted in poor results. Larger spatial samples seems to give good results.

SLO's Results

- Beyls and D'Hollander reported these results.
- An average slow down of about seven in long running programs.



	Speedup				Analysis Time	
	Pentium4 (512KB)	Itanium (2MB)	Alpha (8MB)	Average	Non-Sampling	Sampling
Art	4.11	1.54	1.16	2.39	12h13m	0h16m
Equake	1.10	2.93	3.09	2.30	59h33m	0h22m
VPR.route	1.51	1.40	1.41	1.44	18h32m	0h16m
Galgel	1.92	2.37	2.39	2.22	65h15m	0h24m
Applu	1.63	2.46	1.69	1.92	100h59m	0h28m

Outline

1) Brief motivation and definitions of terms
2) Discussion of sampling methods
3) Locality analysis techniques
4) Introduction to the implementation (SLO-R) and experimental results
5) ***Conclusion***

Conclusion

- Reservoir sampling provides good approximations to program access behavior.

- With a relatively small reservoir, program temporal locality can be approximated well.

- Due to sample size and a more ambiguous metric, spatial locality is more difficult to analyze with sampling. Using a large reservoir it can be done, but its size will certainly be an issue.

Question? Comments?

Thank you everyone for coming.