SEMANTIC LOCALITY & CONTEXT BASED PREFETCHING

Leeor Peled[1], Shie Mannor[1], Uri Weiser[1], Yoav Etsion[1,2]

[1]Electrical Engineering and [2]Computer Science
Technion – Israel Institute of Technology

ICRI-CI
Intel Collaborative Research Institute
Computational Intelligence

ICRI-CI 2015 Retreat, May 2015

TECHNION
Israel Institute of Technology

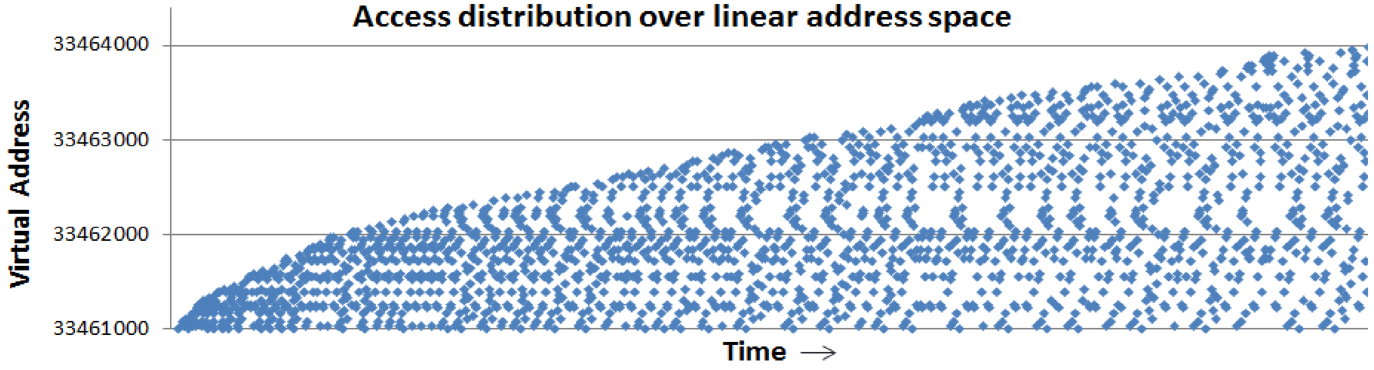
TCE
Technion
Computer
Engineering

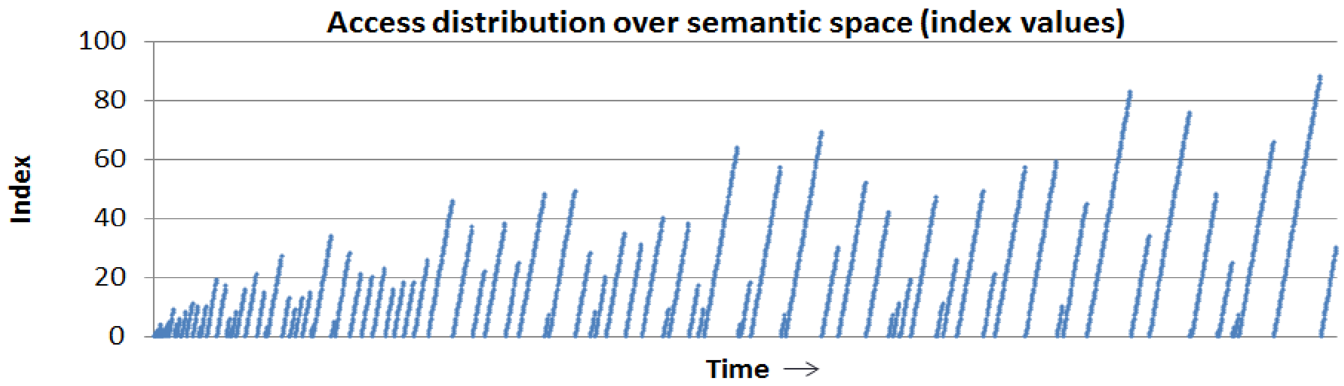
Motivation

- Where does spatial-locality come from?
- Many workloads do not naturally manifest the familiar spatial locality, and are optimized for grid-structured memories
- We argue that spatial-locality is an artifact of grid-structured memory systems.
- Can we identify locality at the algorithm level, in a manner that is agnostic to implementation details?

Semantic Locality

- Data recurrence (locality) is inherent in program semantics and is also common in irregular data structures
- It is just a question of perspective:

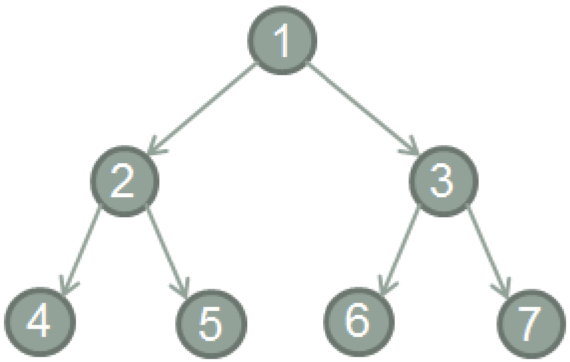




Semantic Locality

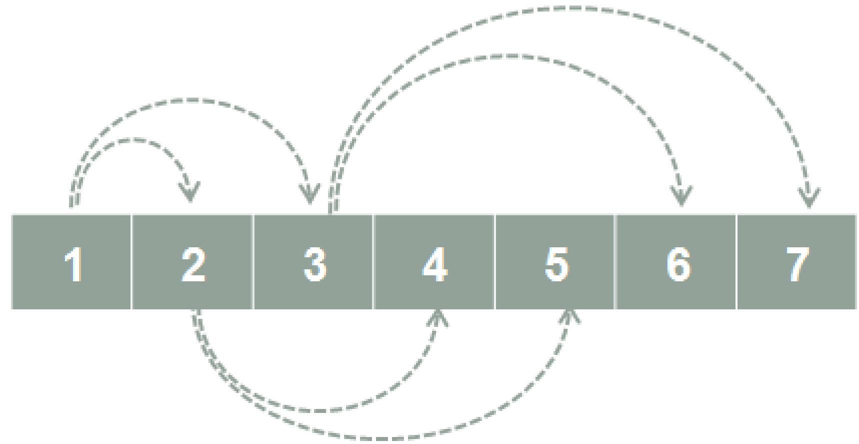
- Locality should be defined by the semantic correlation between memory objects
  - Dictated by program semantics
  - Accesses are semantically adjacent if they are related through a given sequence of actions
- Semantic locality describes the data connectivity at a level that transcends implementation details

Semantic Locality: Binary Tree




```
if (val == node->val)
    return node;
else if (val < node->val)
    node = node->left;
else
    node = node->right;
```

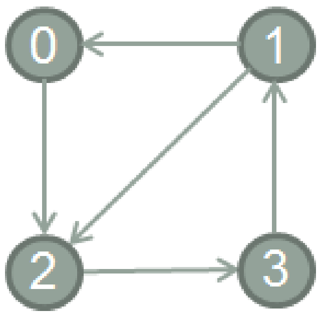
(a) Linked graph




```
if (val == array[index])
    return index;
else if (val < array[index])
    index = index * 2;
else
    index = index * 2 + 1;
```

(b) Array

Semantic Locality: BFS traversal




```
while (!Q.empty) {
   node = Q.pop();
   for (v:neighbours(node)){
      if (!v->visited)
         Q.push(v);
   }
}
```

(a) Linked graph

	0	1	2	3
0			1	
1	1		1	
2				1
3		1		

```
while (!Q.empty) {
   index = Q.pop();
   for (i: 1 .. N-1) {
      if (adjMat[index][i]>0
            && !visited[i])
         Q.push(i);
   }
}
```

(b) Adjacency matrix

Vals	1	1	1	1	1
Cols	2	0	2	3	1
RowIndex	0	1	3	4	

```
while (!Q.empty) {
   index = Q.pop();
   for (j: RowIndex[i] ..
              RowIndex[i+1]-1){
      if (!visited[Cols[j]])
         Q.push(Cols[j]);
   }
}
```

(c) Compressed sparse row (CSR)

Identifying Semantic Locality

- Identifying semantic locality can improve performance
  - Our case study: data prefetching

- The problem:

  How can we dynamically identify Semantic Locality?

- Deterministic tools are not available
  - So use statistics…

- Our approach: **Reinforcement Learning**

Machine Learning Approach

- Reinforcement learning model based on "Contextual Bandits".

  Associate the state of the program and
  the machine with potential future addresses

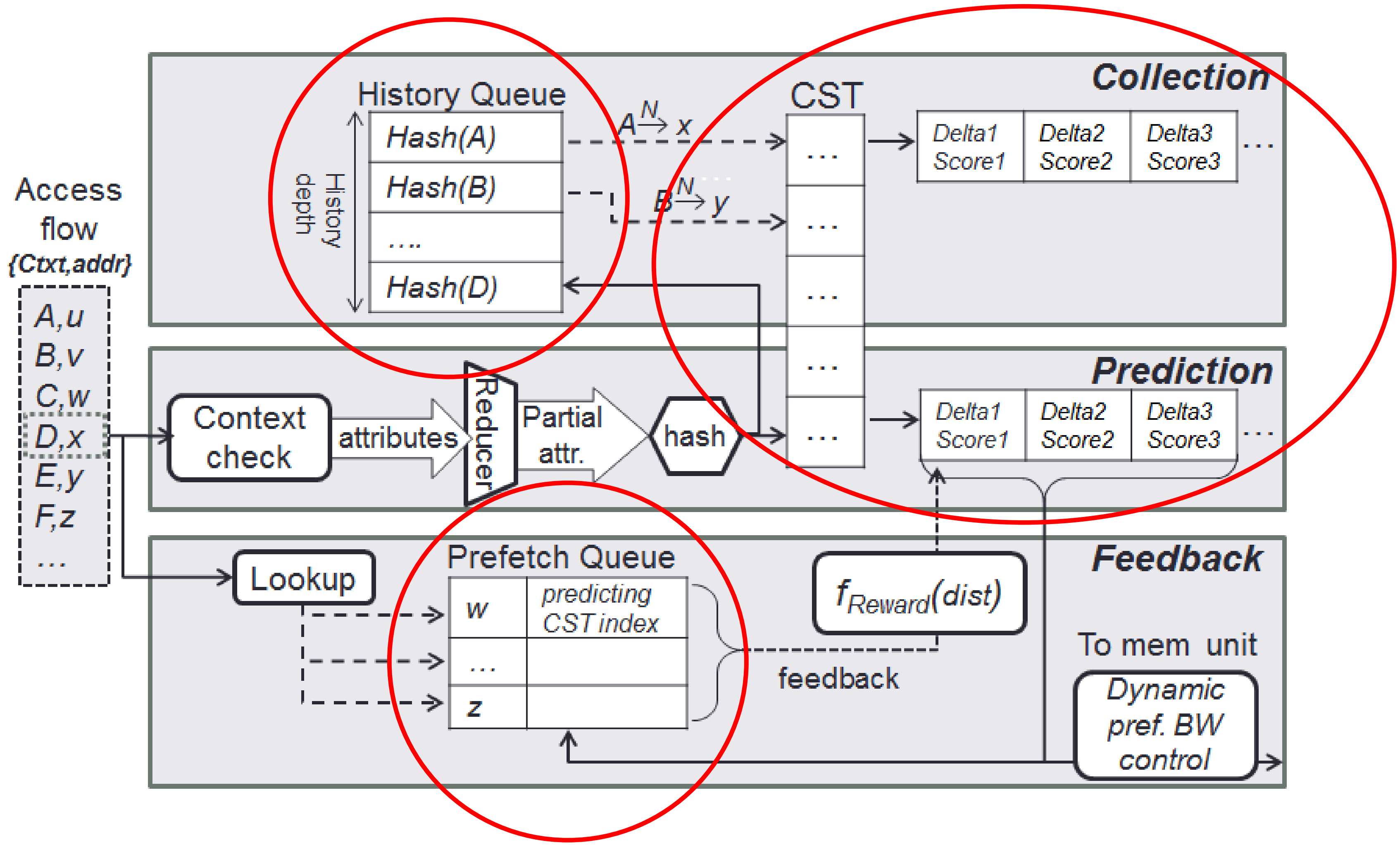
- Machine learning parlance:
  - **State**: program and machine attributes
  - **Action:** which virtual address should be prefetched?
- The system balances *exploration* (learning by taking unknown actions), and *exploitation* (making use of existing knowledge).
- Shadow prefetches: learning with no caching/BW penalty.

Context-based Prefetching

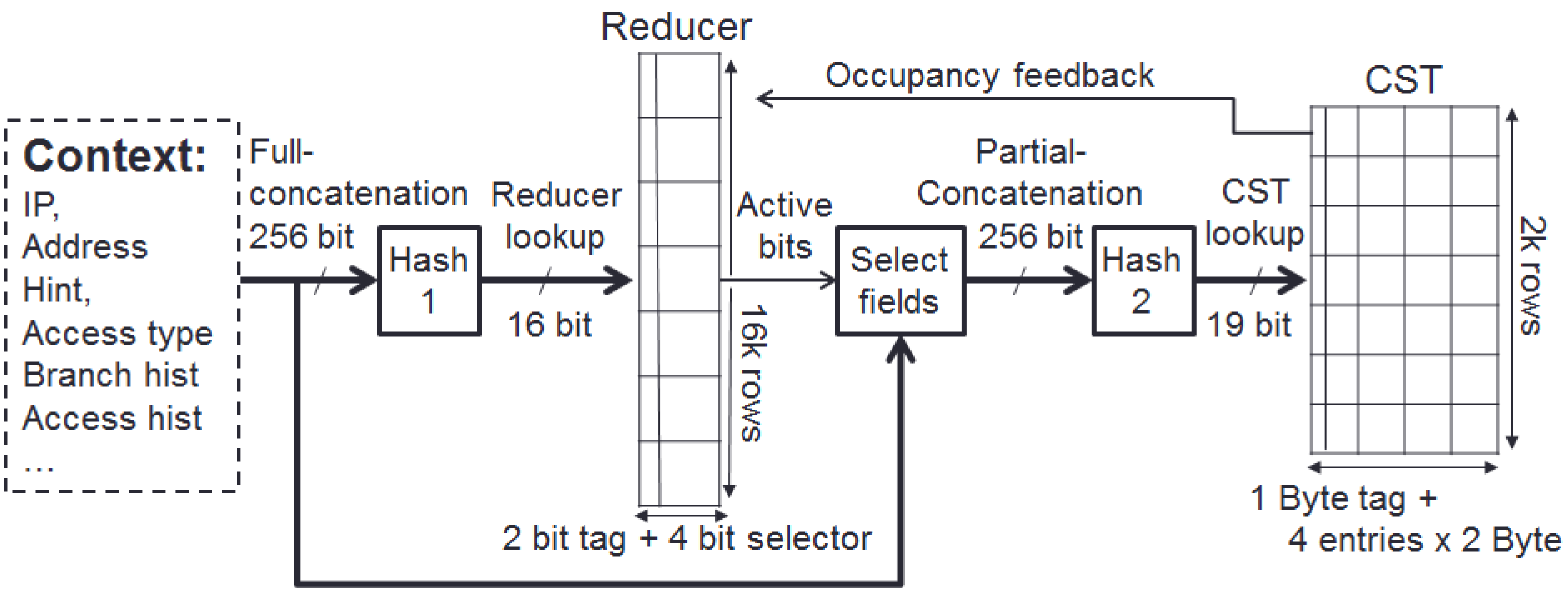
- In order to capture semantic behavior, we identify unique program context-states using a set of SW/HW attributes:
  - Compiler: Type enumeration and structure info.
  - HW: IP, branch history, data values, reference method, etc
- Useful attributes are selected dynamically to avoid overfitting / underfitting
- Accesses are correlated with preceding context states within the prediction range.
- When observing a known context, trigger best scoring prediction (based on accumulated reward).

Prefetcher schematic flow

Context State Table

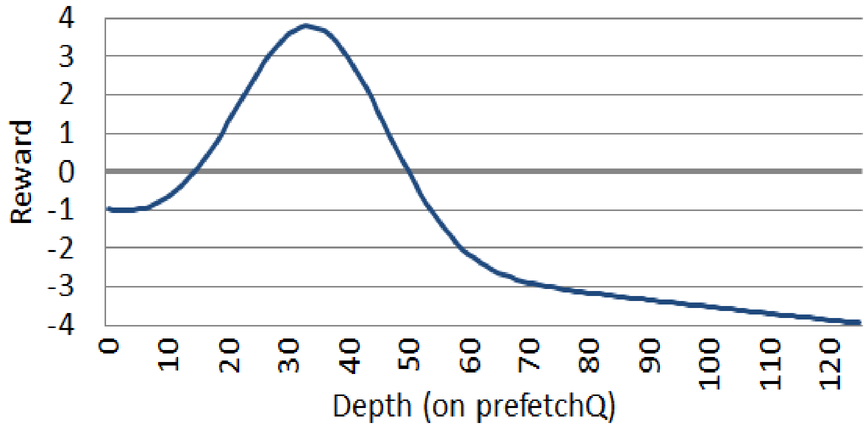


Indexing and Feature Selection



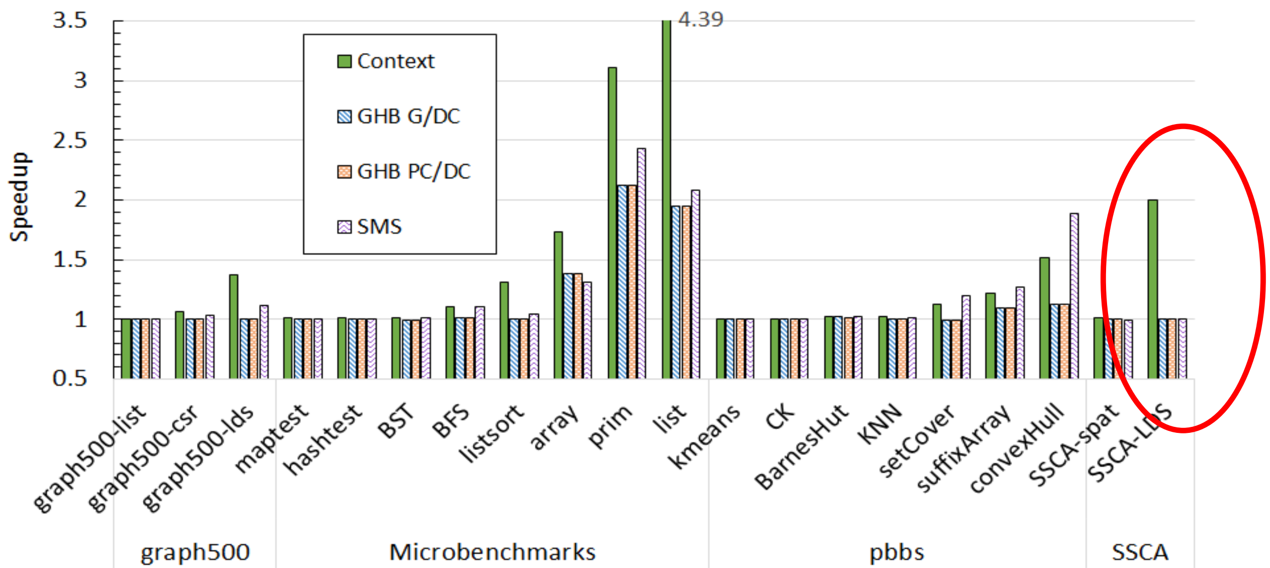
Reward Mechanism

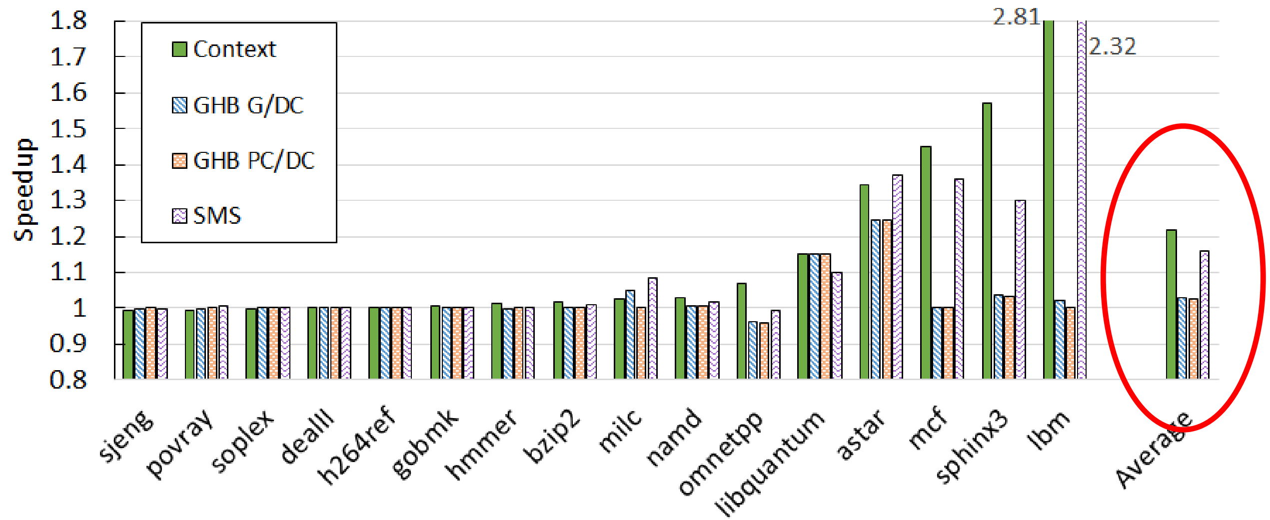
- Feedback is based on prediction accuracy and timeliness, awarded when a prediction is deemed successful or not.
- Bell-shape reward function to accommodate varying depths (OOO effects, changing control flows)



- Target distance:
  - L1 miss penalty = L2 lat. + L2 miss rate × DRAM latency
  - Prefetch distance = L1 miss penalty × IPC × P(mem.op.)

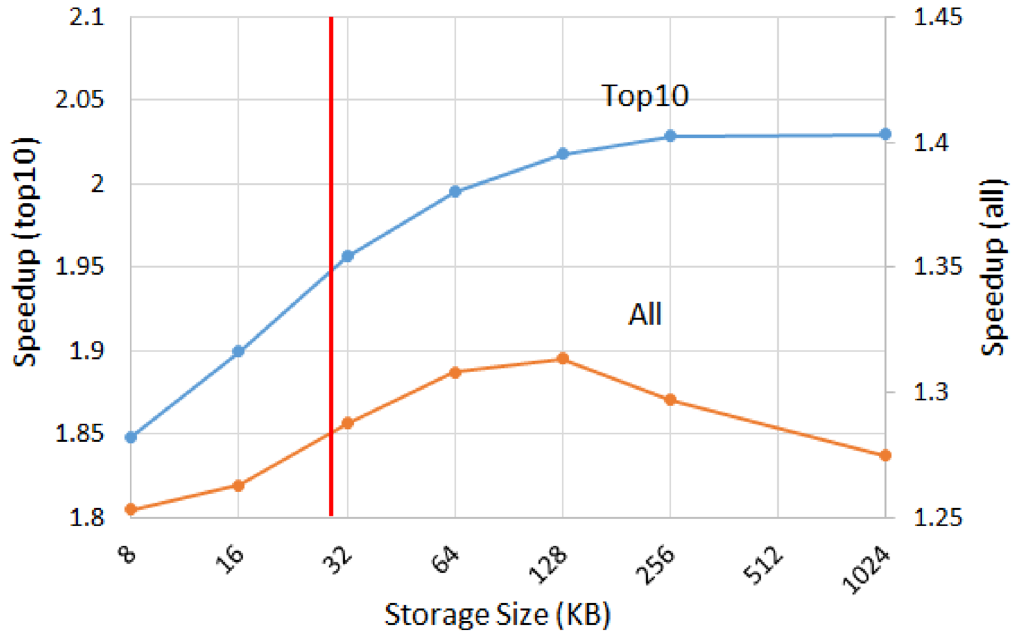
Prefetcher speedup





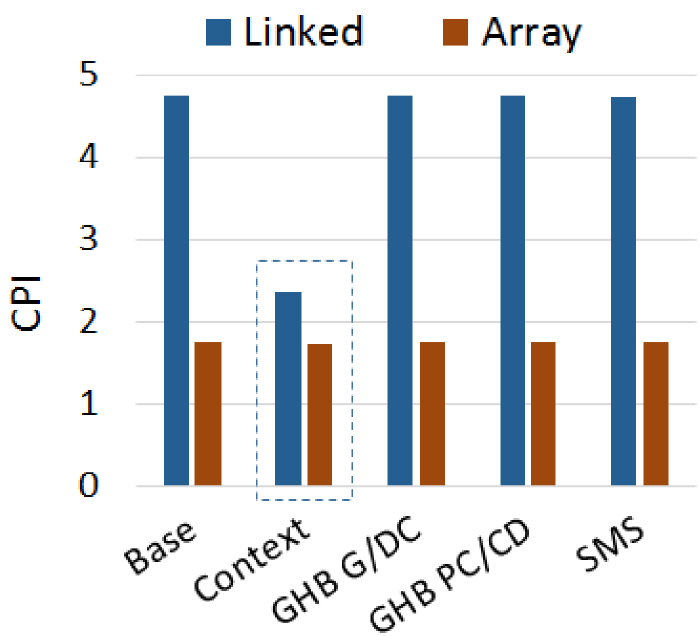
Storage size analysis

- CST and Reducer sizes reflect a tradeoff.
  - On one hand, bigger tables allow better learning history and more potential addresses. They also reduce the chance of forgetting useful but infrequent associations.
  - On the other hand, bigger tables increase the learning time and harm convergence.

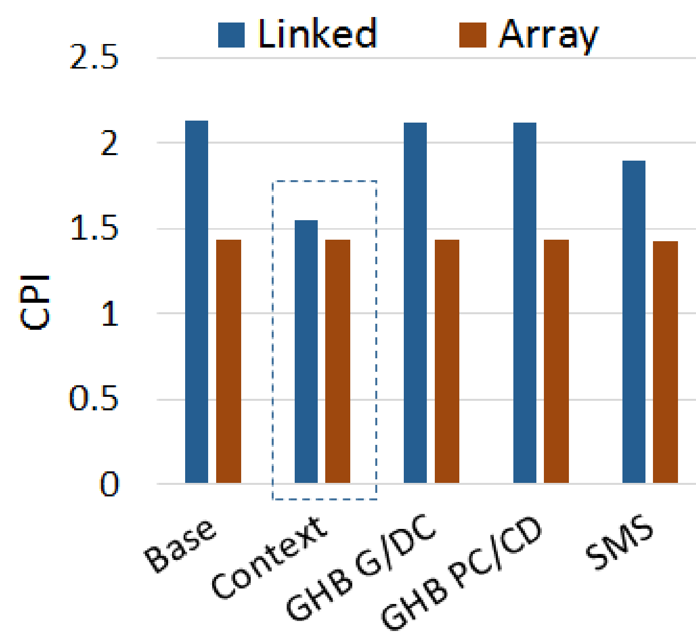


Layout agnostic programming

- We implemented several algorithms in both a “naïve” and spatially optimized ways (linked data structures vs. arrays).
- Our contextual prefetcher provides almost the same optimized performance.



SSCA2



graph500

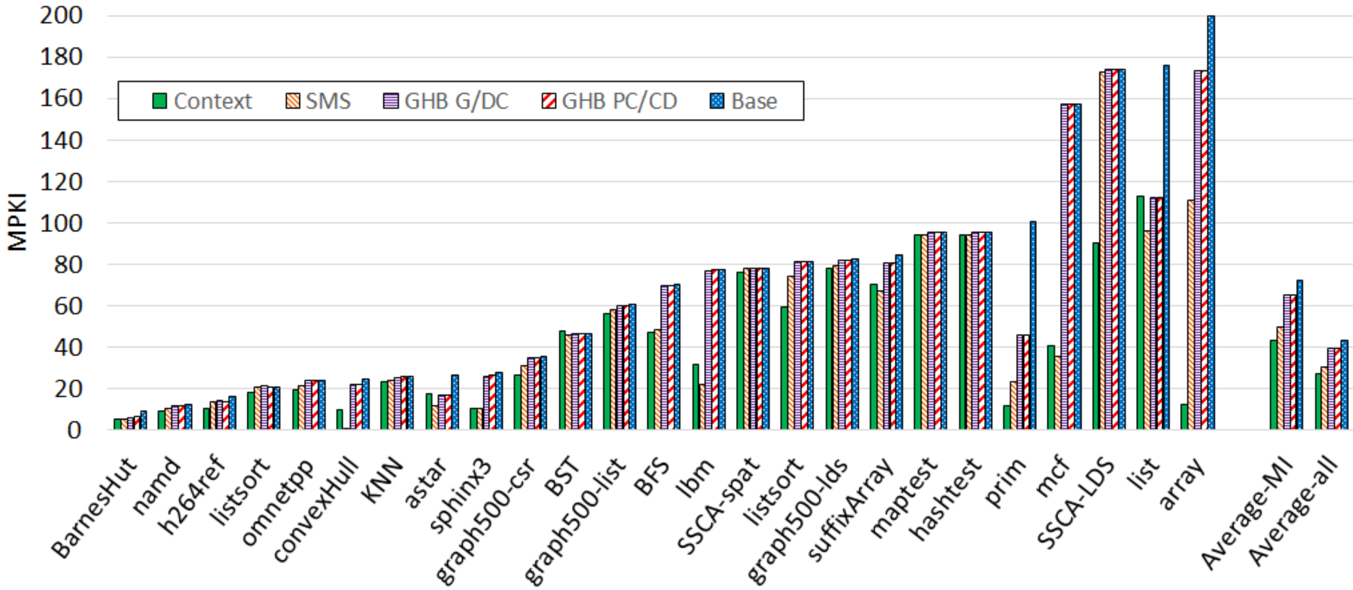
Conclusion

- We argue that locality is an attribute of program semantics
- Program + machine attributes can represent a semantic execution context
- Machine learning can approximate semantic locality and be used to prefetch data
- What else can Semantic Locality + ML be used for?
  - Value prediction, branch prediction, …

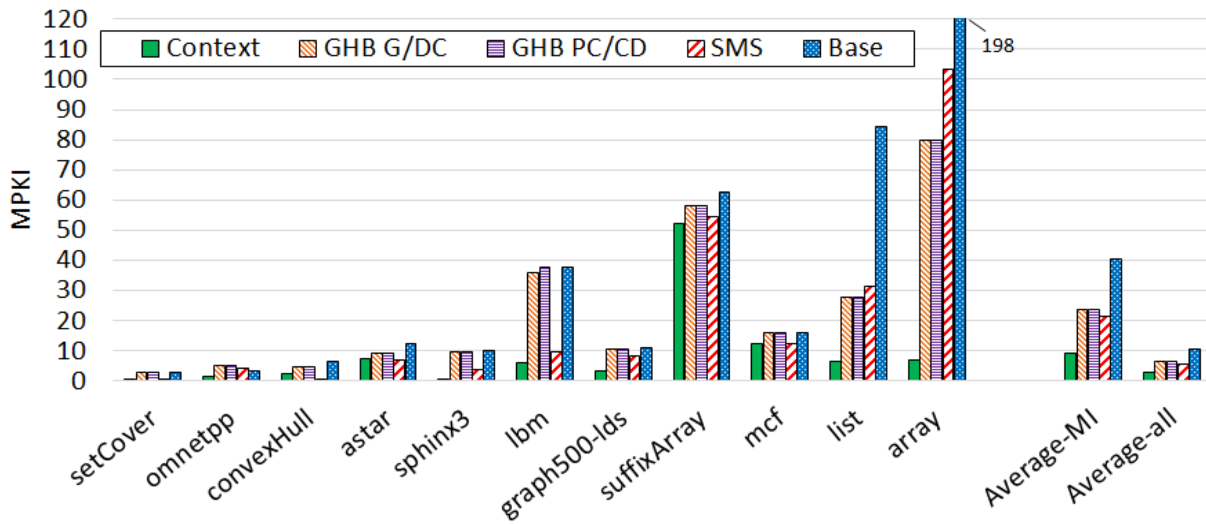
BACKUP FOILS

MPKI comparison

Level1 data cache
(cutoff - MPKI > 5)



Level2 cache
(cutoff - MPKI > 1)



Prefetcher accuracy (bucketing)