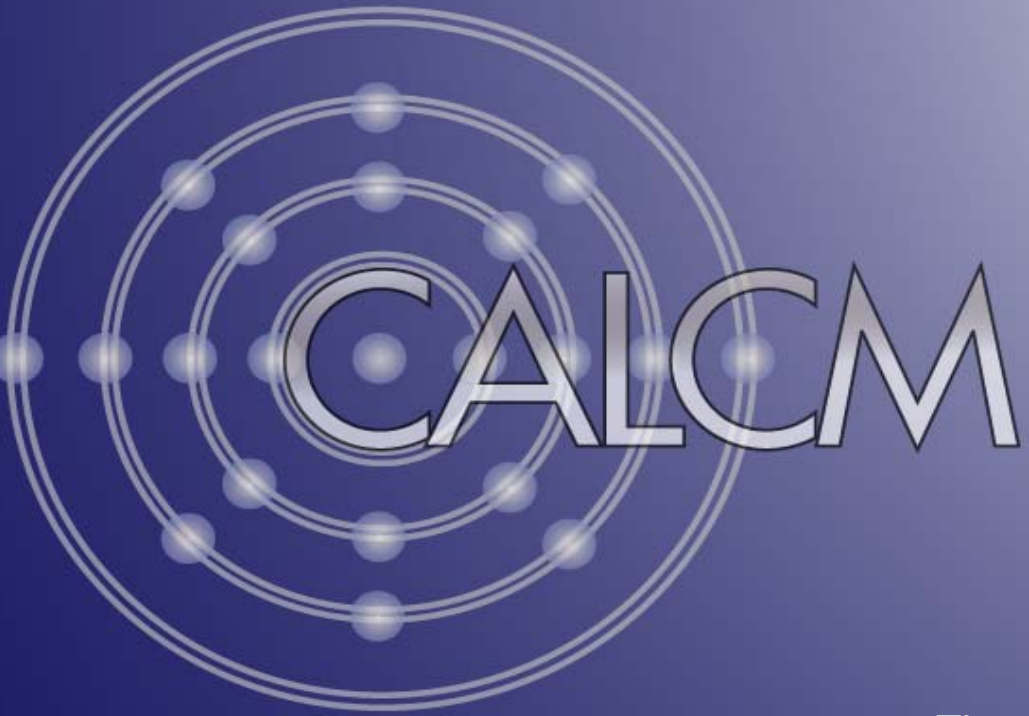
Execution Slack and Criticality

Nikos Hardavellas



The Computer Architecture Lab at Carnegie Mellon
http://www.ece.cmu.edu/CALCM

Execution Slack & Criticality: Context

- Out of order processors are highly parallel
  - overlap computation
- Technological constraints
  - wire delay, power, complexity
- Non-uniform designs
  - clustering, multi-frequency FU's, memory hierarchies
- Criticality & slack
  - what is it?

Execution Slack & Criticality: Why do I care?

- Exploit to guide control policies
- Examples here do :
  1. Resource Arbitration
  2. Misspeculation Reduction
  3. Power Management
  4. Cache Management

Papers

1. Focusing processor policies via critical-path prediction
   - Framework to identify instruction criticality and design to increase instruction scheduling locality in a cluster
2. Slack: maximizing performance under technological constraints
   - Framework to identify variability in instruction execution criticality and application in slowing down execution to save power
3. Locality vs. Criticality
   - Is sacrificing locality for criticality a good idea?
4. Non-vital loads
   - Identify instructions that can tolerate longer cache hit latencies and remove them from higher-level caches

Evaluation/Outline

- Overview of each technique
- Present results
- Present pros & cons

Papers

1. Focusing processor policies via critical-path prediction
   - Framework to identify instruction criticality and design to increase instruction scheduling locality in a cluster
2. Slack: maximizing performance under technological constraints
   - Framework to identify variability in instruction execution criticality and application in slowing down execution to save power
3. Locality vs. Criticality
   - Is sacrificing locality for criticality a good idea?
4. Non-vital loads
   - Identify instructions that can tolerate longer cache hit latencies and remove them from higher-level caches

Paper 1: Context

- Out of order processors are highly parallel
- Critical-path analysis identifies dominating instructions
  - Micro-architectural critical path
- Focus on optimizing critical instructions
  - Better arbitration of scarce resources
  - Reduce misspeculation

Paper 1: Contributions

- Dependence-graph model of micro-architectural critical path
- Hardware predictor to approximate instruction criticality
- Resource arbitration
- Misspeculation reduction

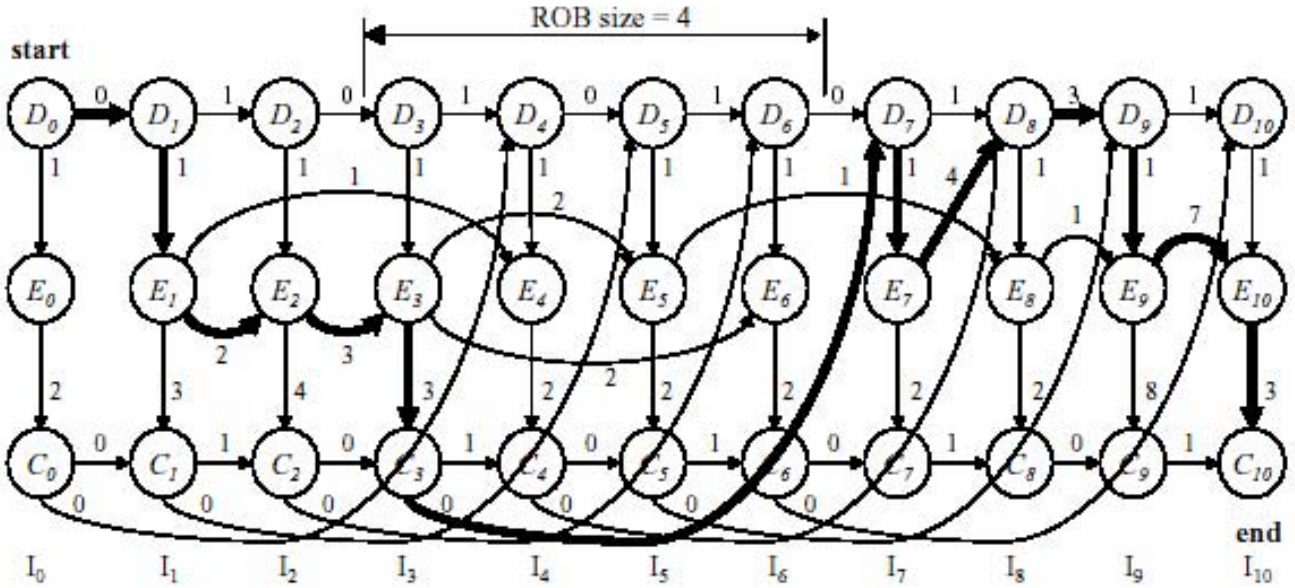
Paper 1: Critical Path Model

name	constraint modeled	edge	
DD	In-order dispatch	$D_{i-1} \rightarrow D_i$	
CD	Finite re-order buffer	$C_{i-w} \rightarrow D_i$	w = size of the re-order buffer
ED	Control dependence	$E_{i-1} \rightarrow D_i$	inserted if $i-1$ is a mispredicted branch
DE	Execution follows dispatch	$D_i \rightarrow E_i$	
EE	Data dependences	$E_j \rightarrow E_i$	inserted if instruction j produces an operand of i
EC	Commit follows execution	$E_i \rightarrow C_i$	
CC	In-order commit	$C_{i-1} \rightarrow C_i$	

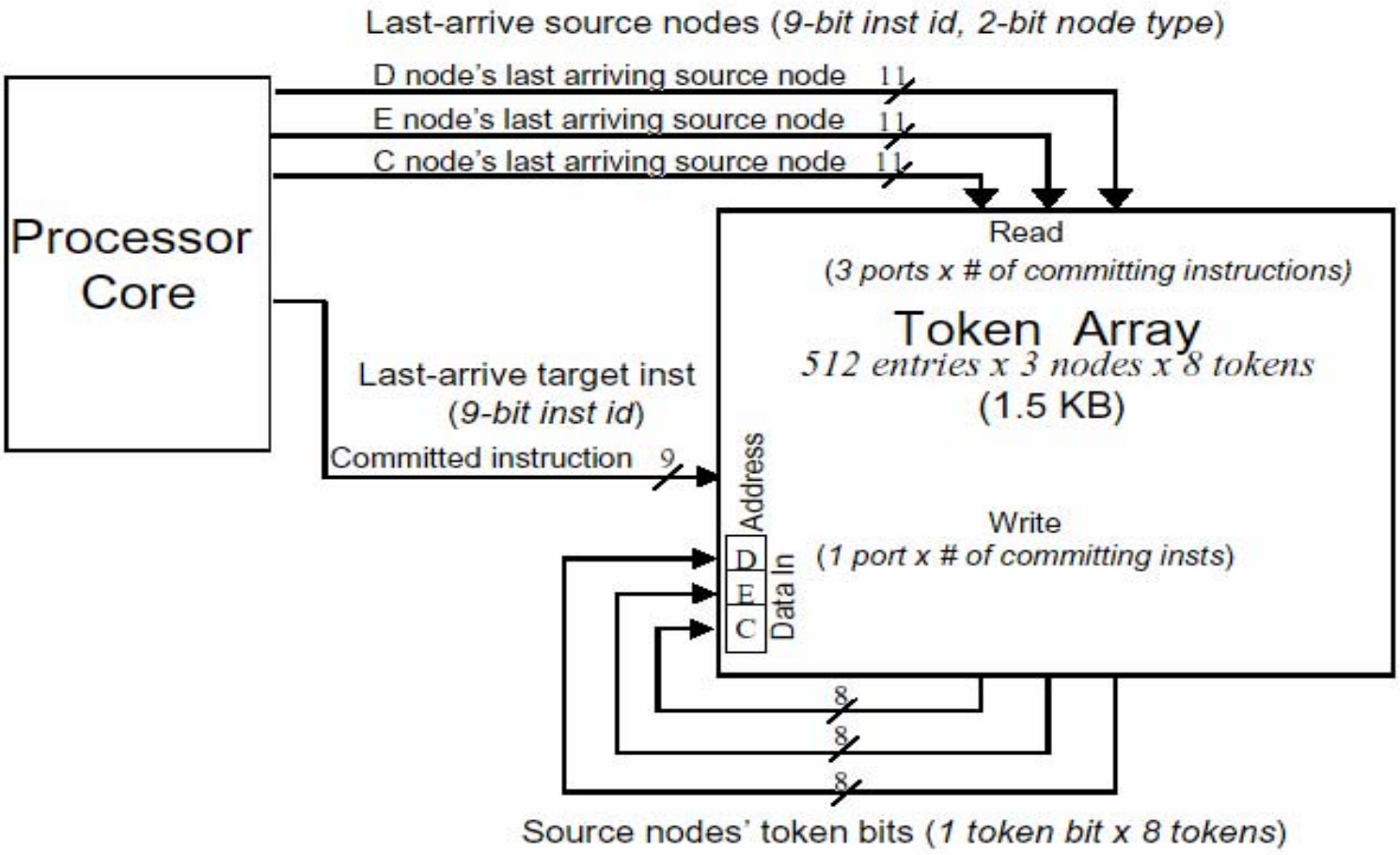
Table 1: **Dependences captured by the critical-path model**, grouped by the target of the dependence.

```
    I0: r5=0
    I1: r3=ld[r2]
L1: I2: r1=r3*6
    I3: r6=ld[r1]
    I4: r3=r3+1
    I5: r5 = r6 + r5
    I6: cmp R6,0
    I7: br L1
    I8: r5 = r5+100
    I9: r0 = r5 / 3
   I10: Ret r0
```

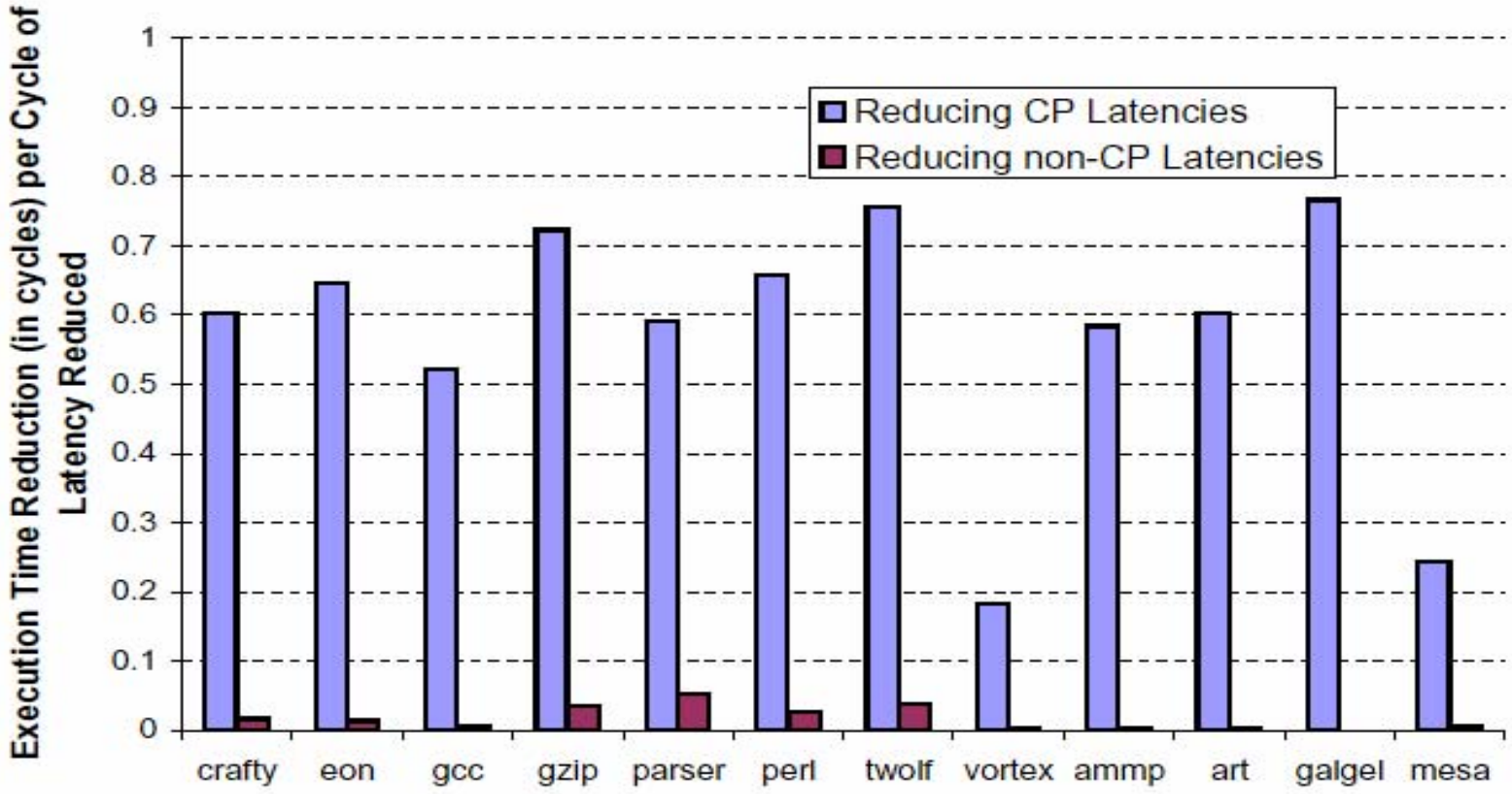
Dynamic Inst. Trace



Paper 1: Hardware

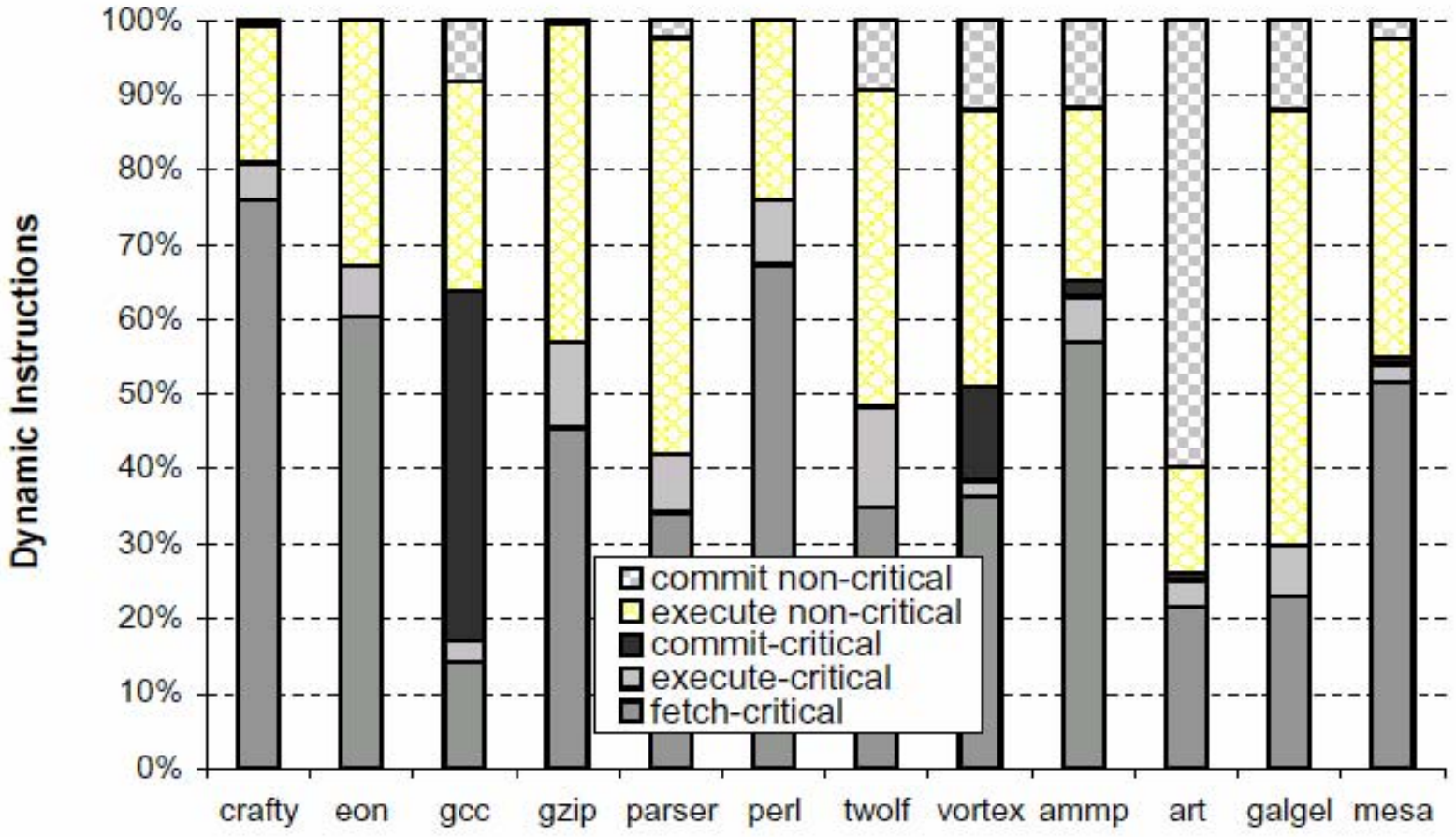


Paper 1: Model Evaluation



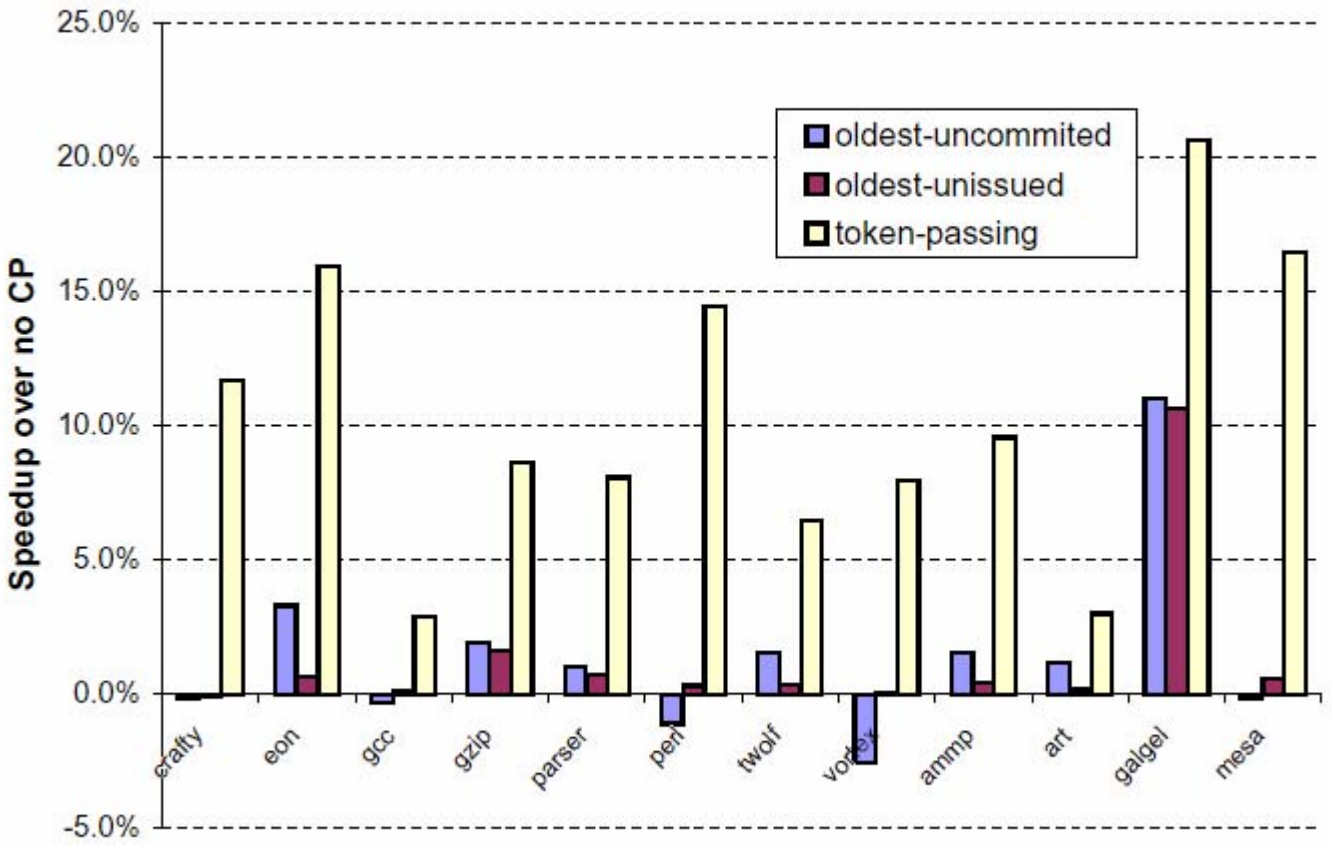
(a) Validation of the critical-path model.

Paper 1: Model Evaluation



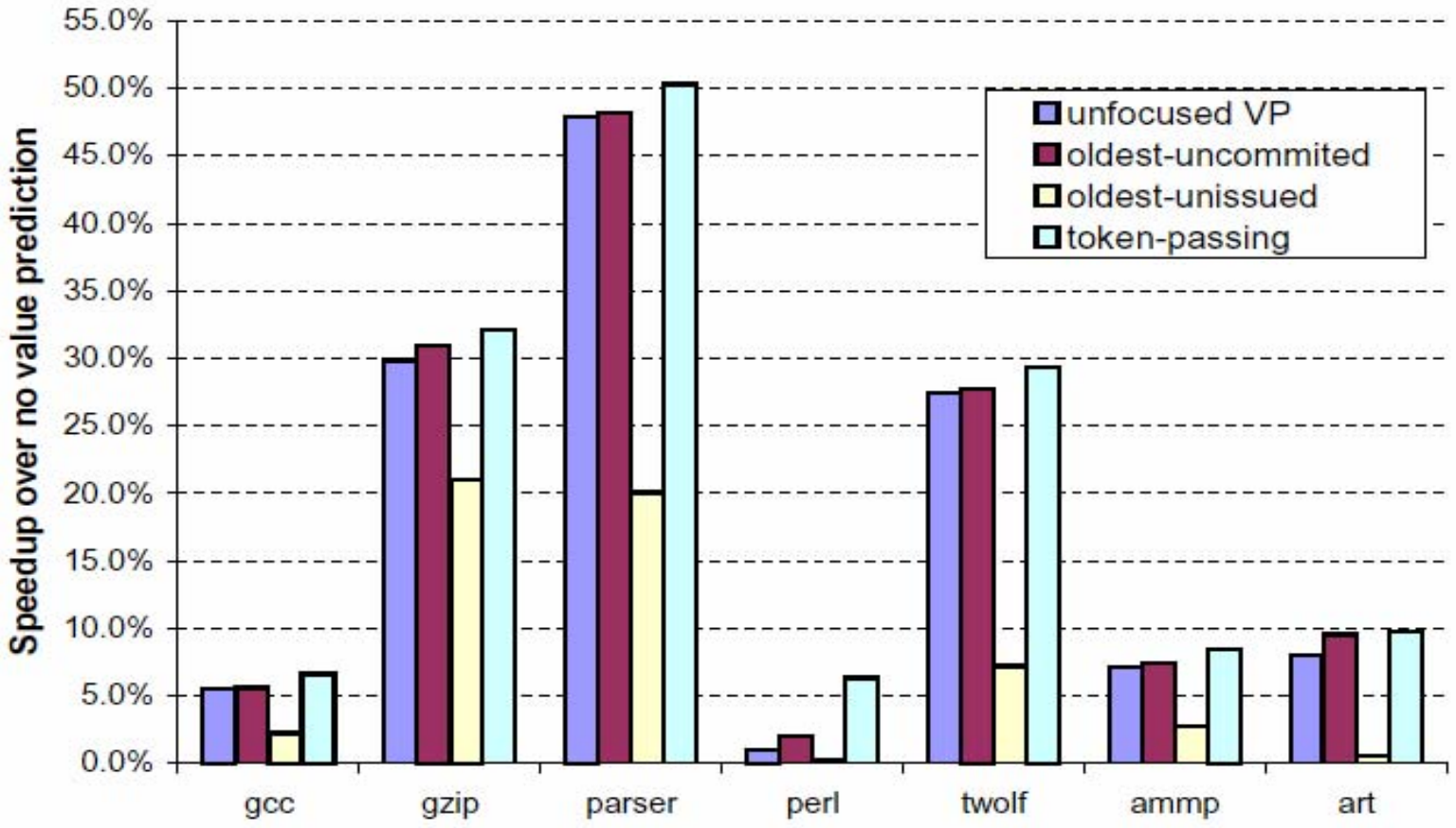
(b) Breakdown of the dynamic instruction count.

Paper 1: Instruction Scheduling and Steering



(b) Comparison to heuristics-based predictors.

Paper 1: Value Prediction



(b) Speedup of focused value prediction.

Paper 1: Critique

Prons

- Multipurpose framework
- Adaptive
- Practical algorithmic approach
- Yields interesting results

Cons

- Wrong path instructions, criticality at memory system ?
- Coarse
- Classification imbalance
- Practical hardware realization

Papers

1. Focusing processor policies via critical-path prediction
   - Framework to identify instruction criticality and design to increase instruction scheduling locality in a cluster
2. Slack: maximizing performance under technological constraints
   - Framework to identify variability in instruction execution criticality and application in slowing down execution to save power
3. Locality vs. Criticality
   - Is sacrificing locality for criticality a good idea?
4. Non-vital loads
   - Identify instructions that can tolerate longer cache hit latencies and remove them from higher-level caches

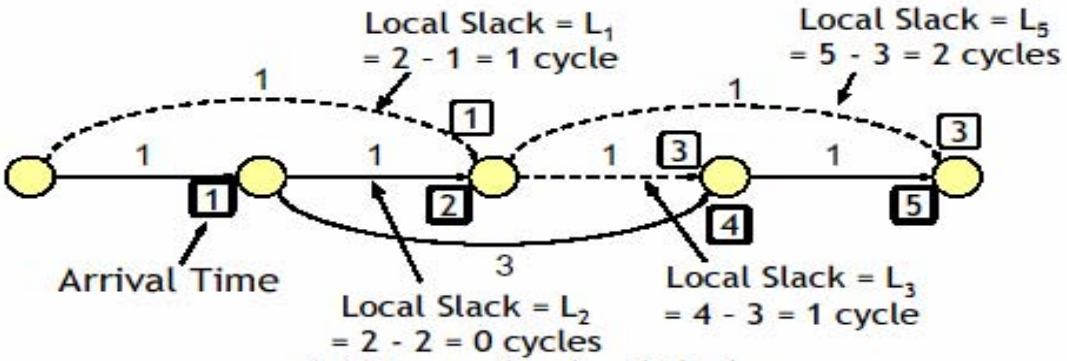
Paper 2: Context

- Technological constraints
  - wire delay, power, complexity
- Non-uniform designs
  - clustering, multi-frequency FU's
- Control policies
- Slack

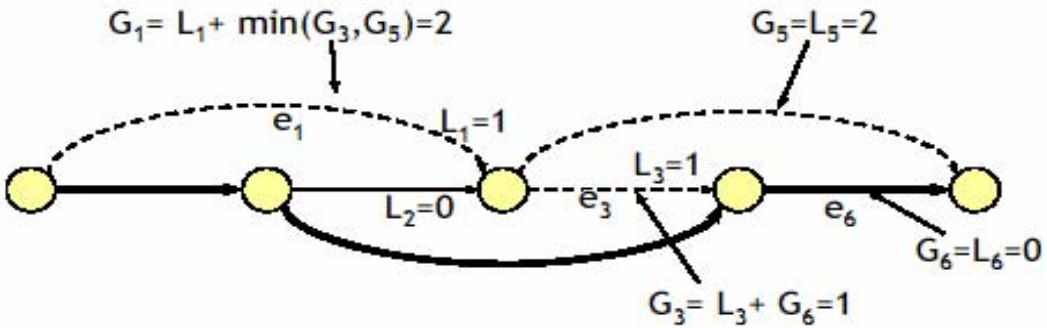
Paper 2: Contributions

- Model and characterize slack
  - local, global, apportioned
- Slack prediction
  - explicit, implicit
- Application in power saving hardware
  - fast/slow pipeline

Paper 2: Computing Slack

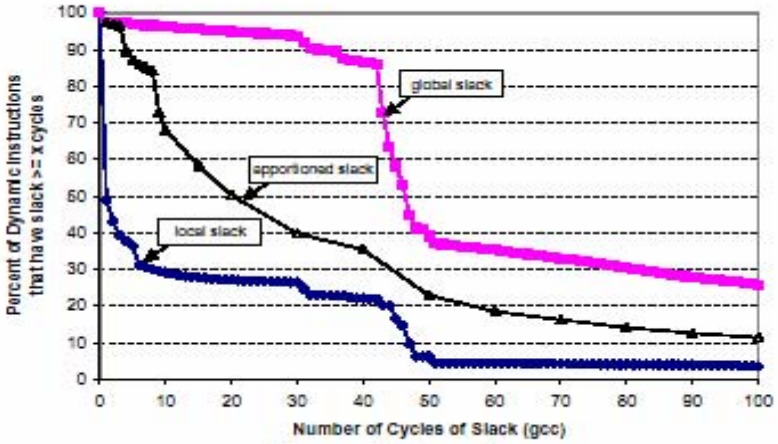


(a) Computing local slack.

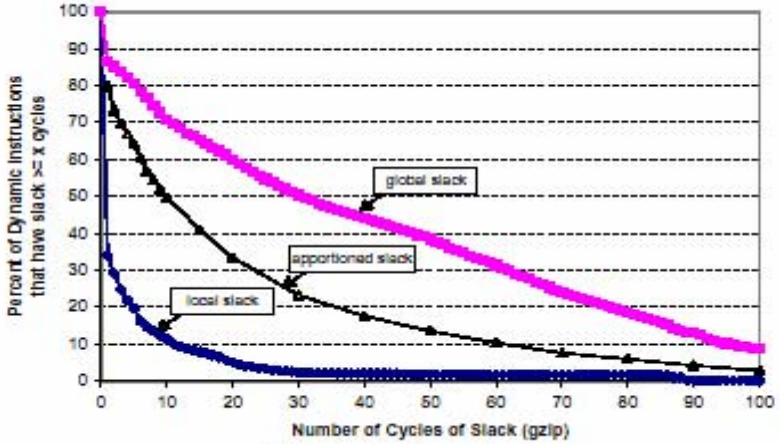


(b) Computing global slack.

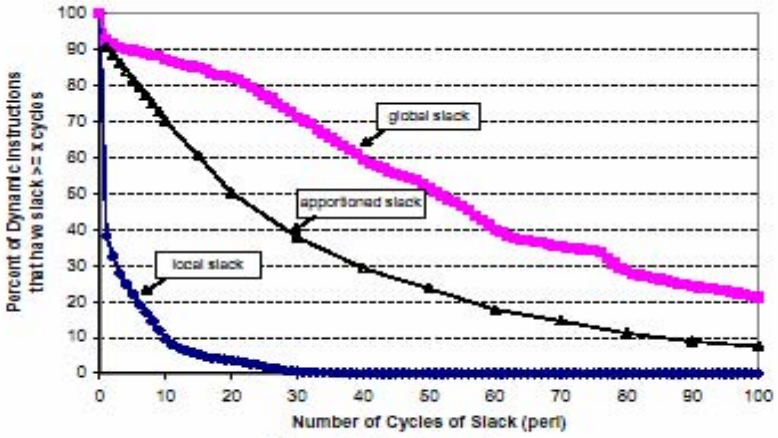
Paper 2: Slack Potential



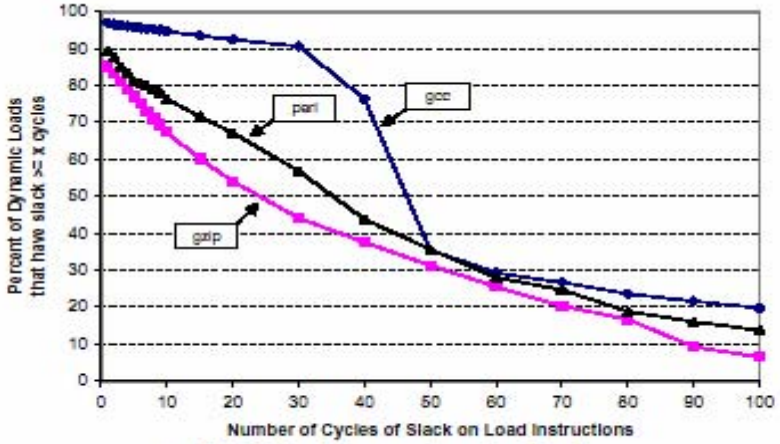
(a) *gcc* slack results.



(b) *gzip* slack results.

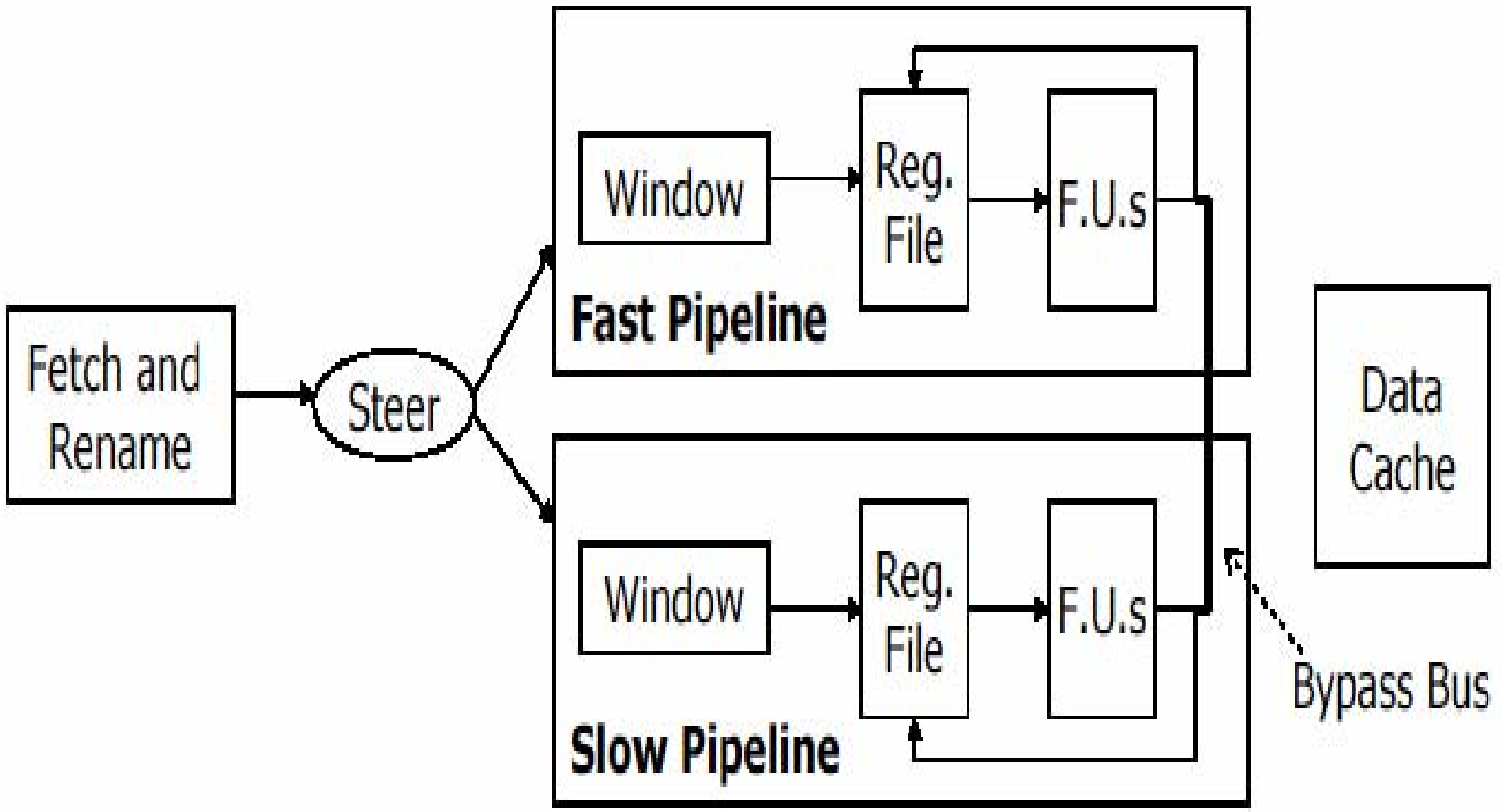


(c) *perl* slack results.

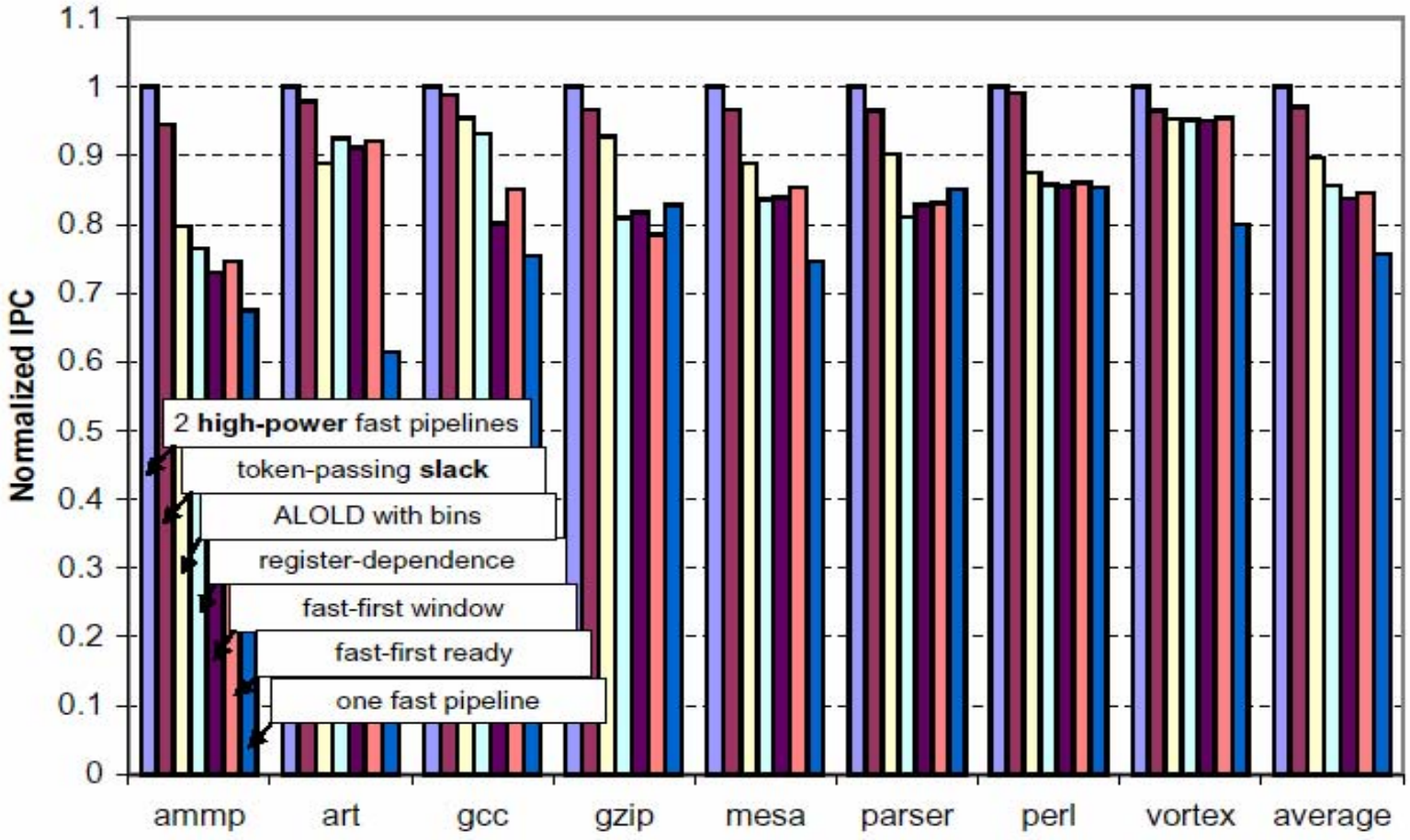


(d) All slack apportioned to loads.

Paper 2 : Microarchitecture



Paper 2: Performance Impact



Paper 2: Critique

Prons:

- Easy to reason
- Adaptive
- Fine grained

Cons

- Sampling constraints
- Criticality at memory system ?
- Weakly defined apportioned slack

Papers

1. Focusing processor policies via critical-path prediction
   - Framework to identify instruction criticality and design to increase instruction scheduling locality in a cluster
2. Slack: maximizing performance under technological constraints
   - Framework to identify variability in instruction execution criticality and application in slowing down execution to save power
3. Locality vs. Criticality
   - Is sacrificing locality for criticality a good idea?
4. Non-vital loads
   - Identify instructions that can tolerate longer cache hit latencies and remove them from higher-level caches

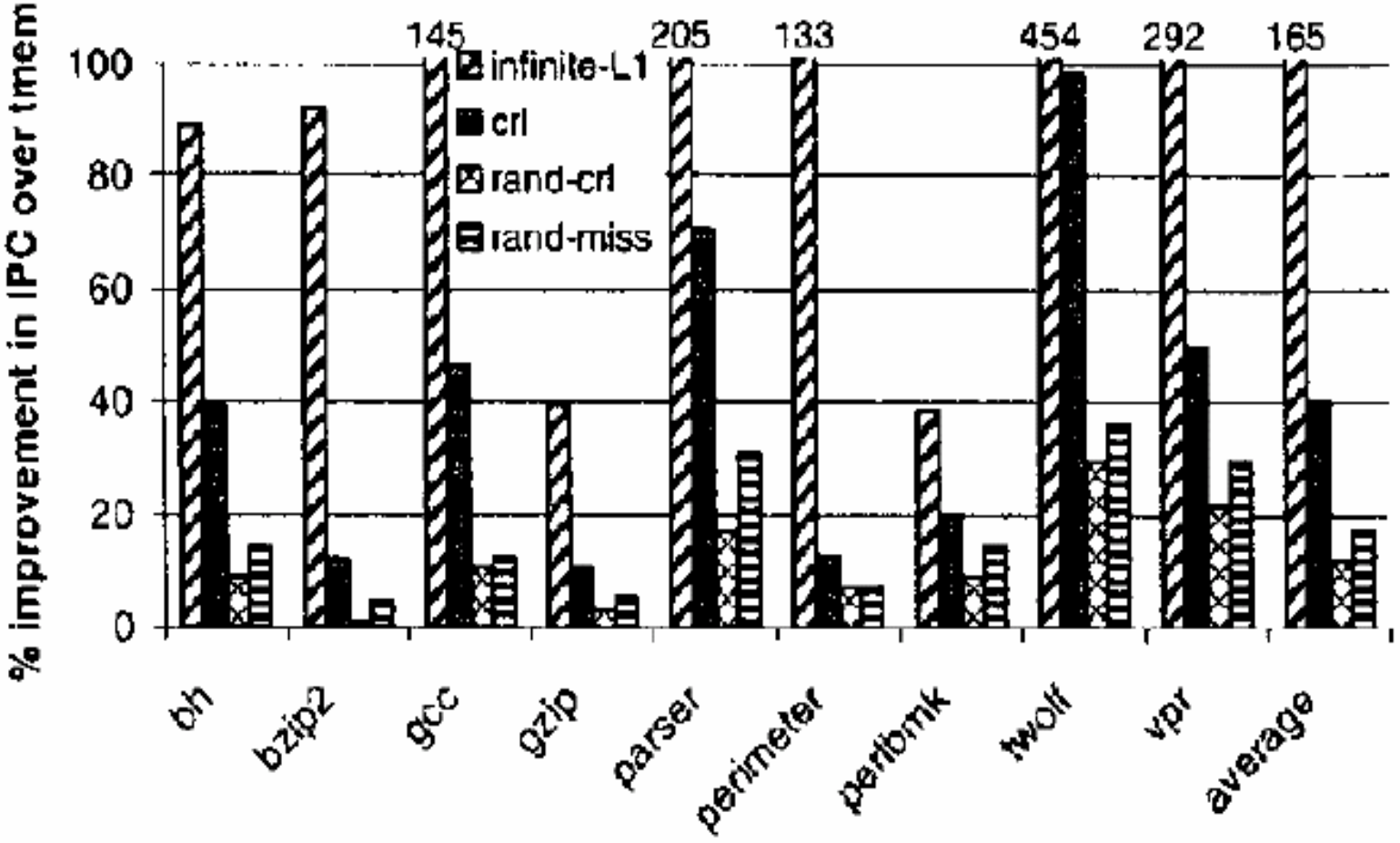
Paper 3: Context

- Caches exploit spatial and temporal locality
- Locality based schemes ignore nature of misses
- Is criticality a strong enough property to warrant a change in memory hierarchies?
- Classify loads as critical
  - feed mispredicted branch, feed into missed load, few independent instructions following

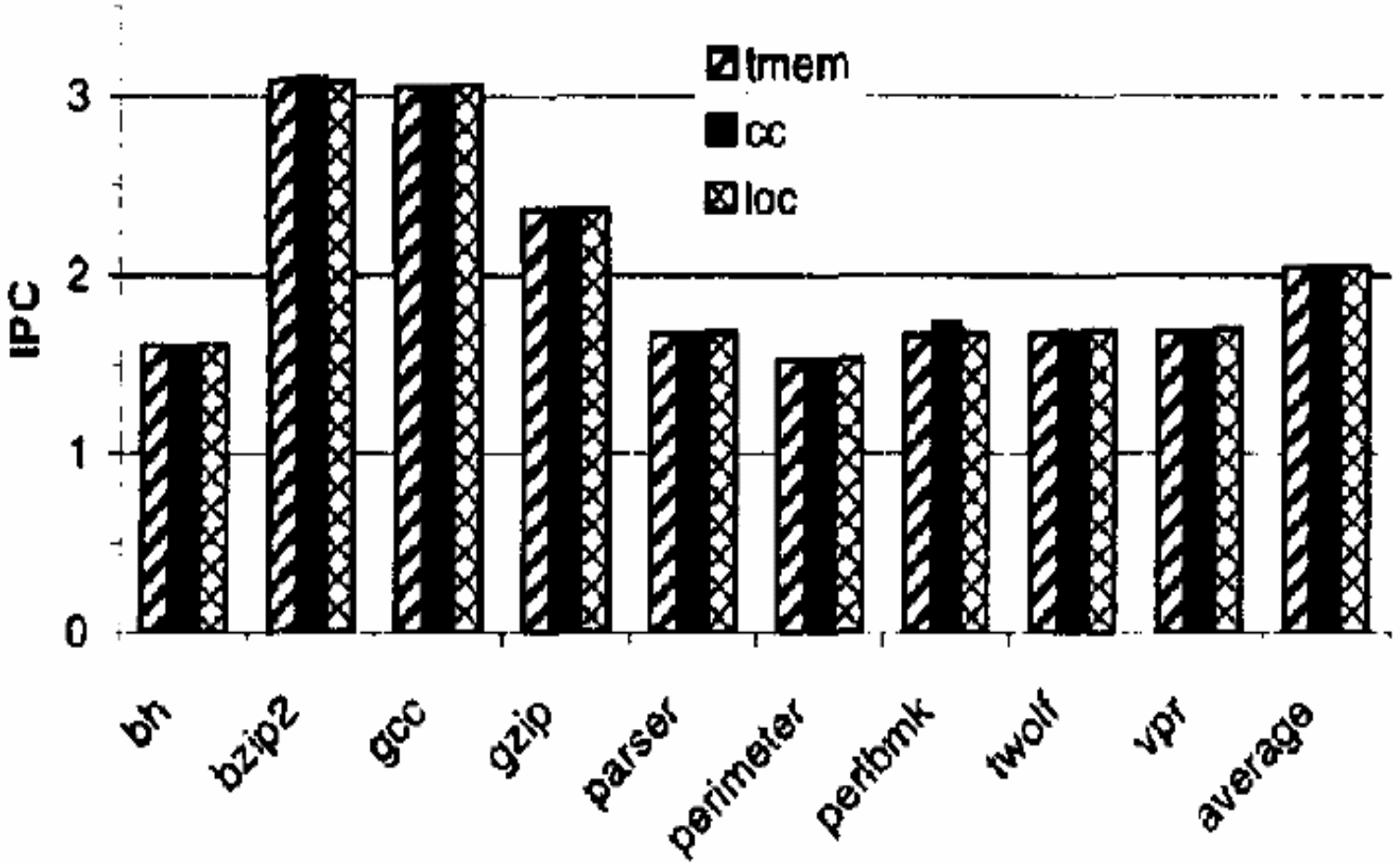
Paper 3: Contributions

- Provides an answer to the question posed
  - Potential is there
  - Implementations fail

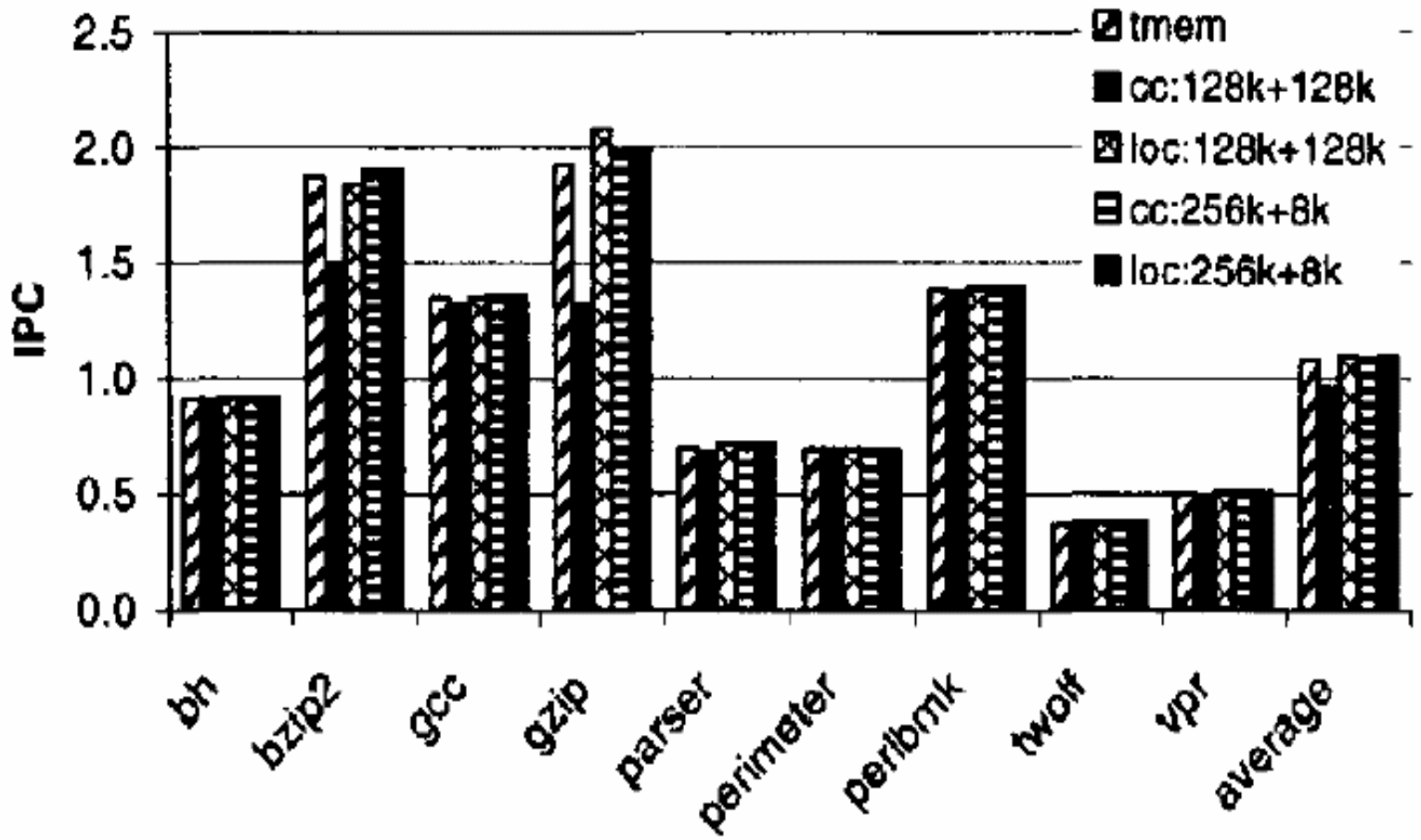
Paper 3: L1 Criticality Potential



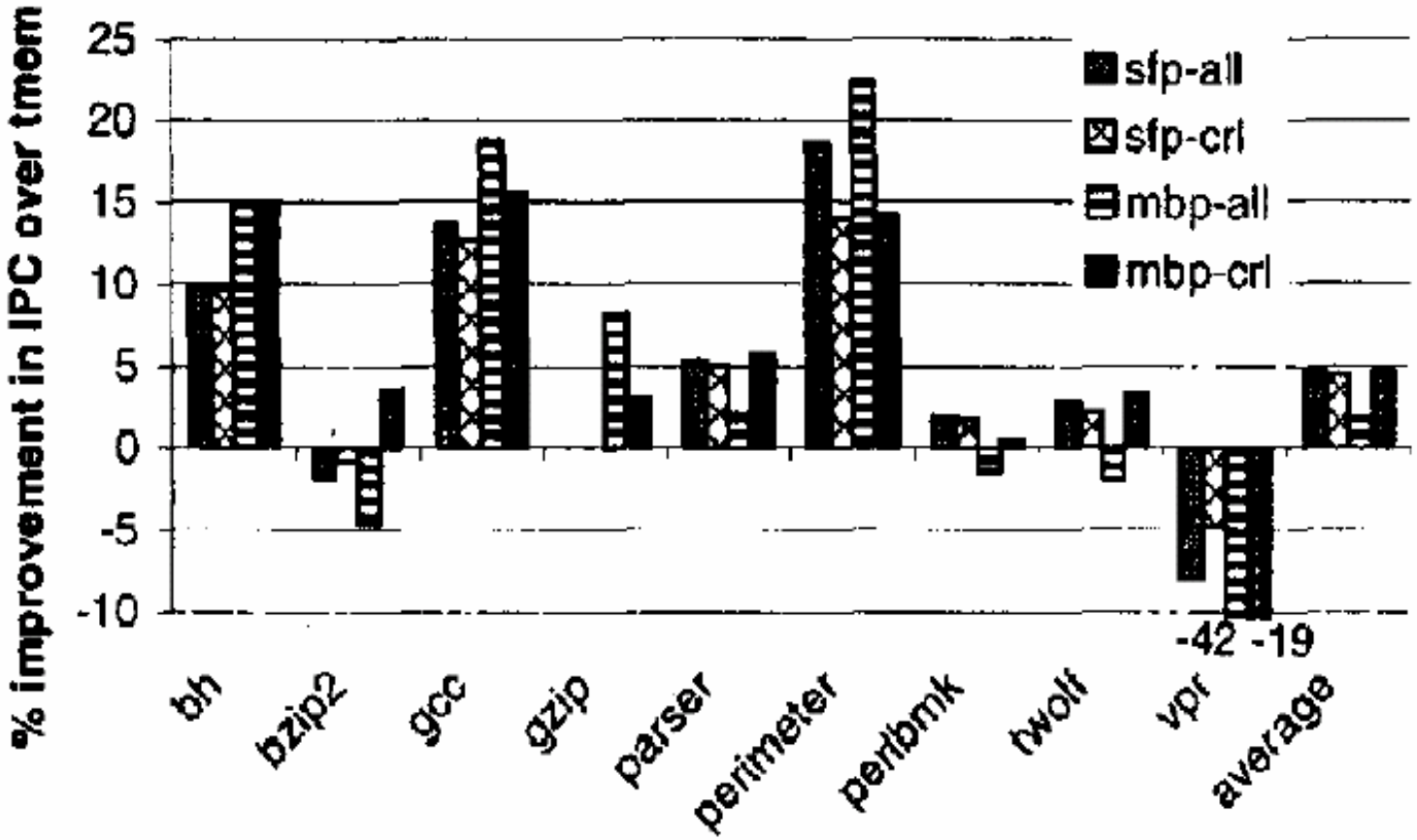
Paper 3: Exploiting L1 Criticality



Paper 3: Exploiting L2 Criticality



Paper 3: Criticality Guided Prefetching



Paper 3: Critique

Prons

- Raises important question
- Provides a limit study: the potential is there

Cons

- Is classification really complete?
- Coverage?
- Accuracy?

1. Focusing processor policies via critical-path prediction

   Framework to identify instruction criticality and design to increase instruction scheduling locality in a cluster

2. Slack: maximizing performance under technological constraints

   Framework to identify variability in instruction execution criticality and application in slowing down execution to save power

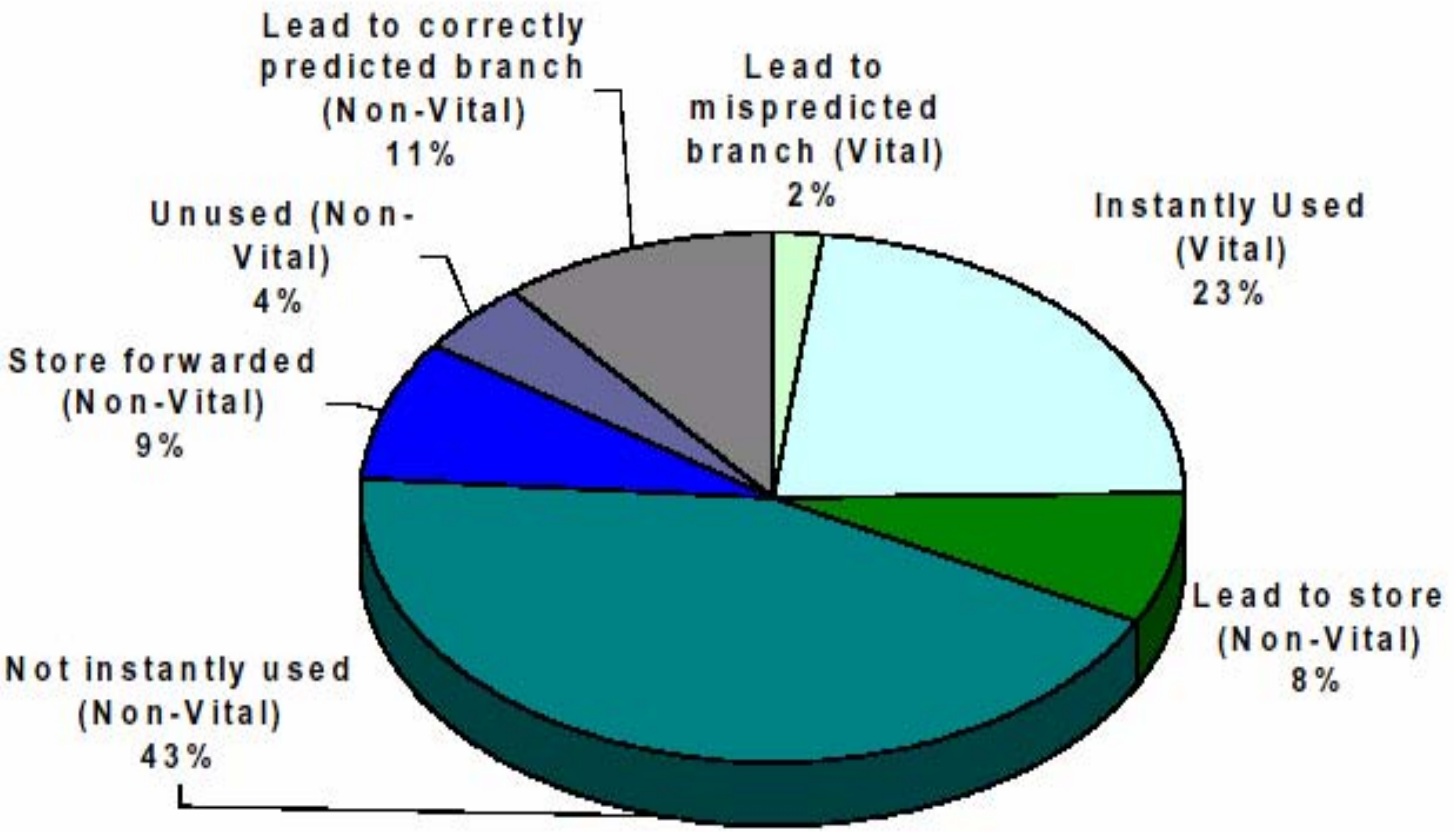
3. Locality vs. Criticality

   Is sacrificing locality for criticality a good idea?

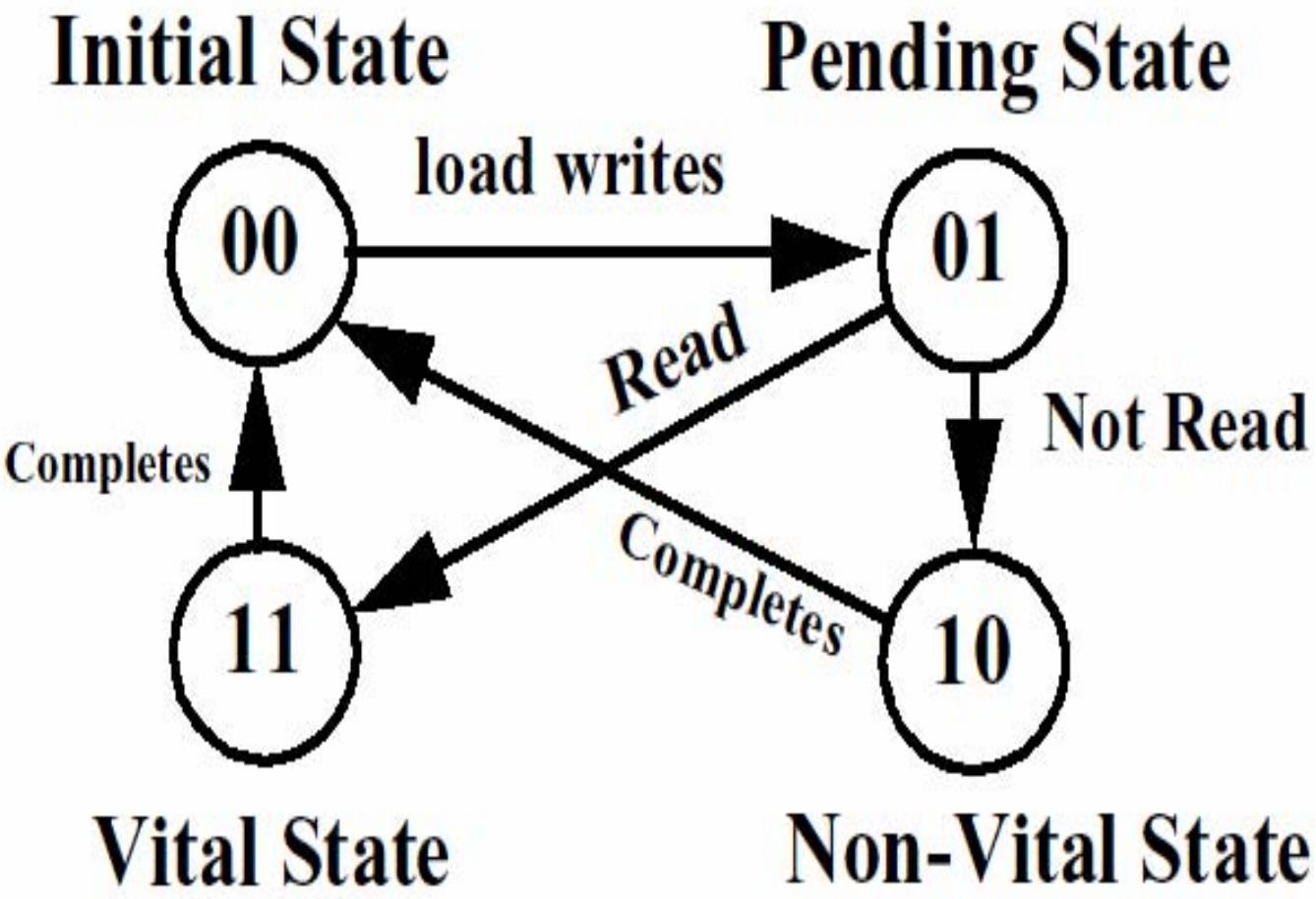
4. Non-vital loads

   Identify instructions that can tolerate longer cache hit latencies and remove them from higher-level caches

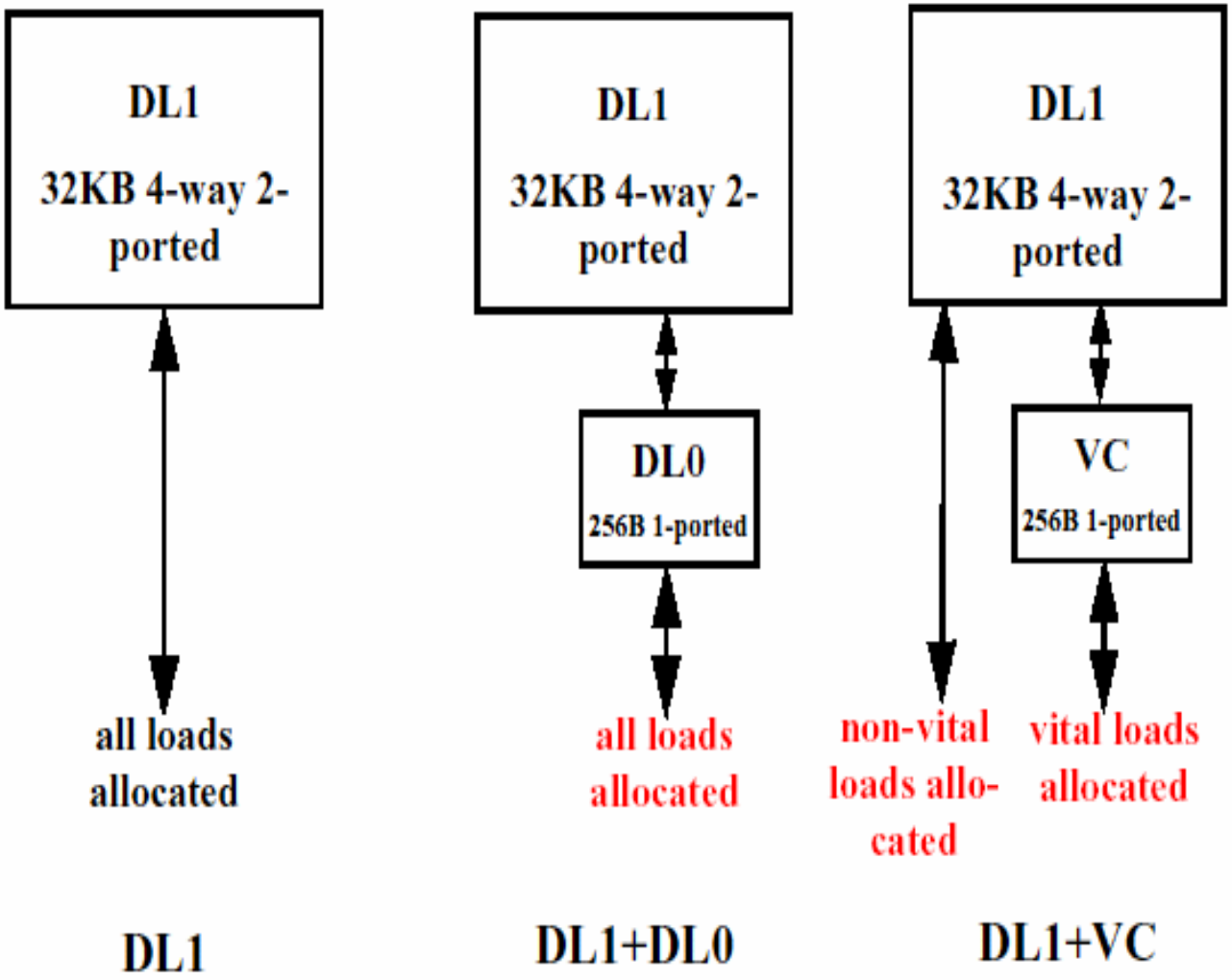
Paper 4: Load Classification



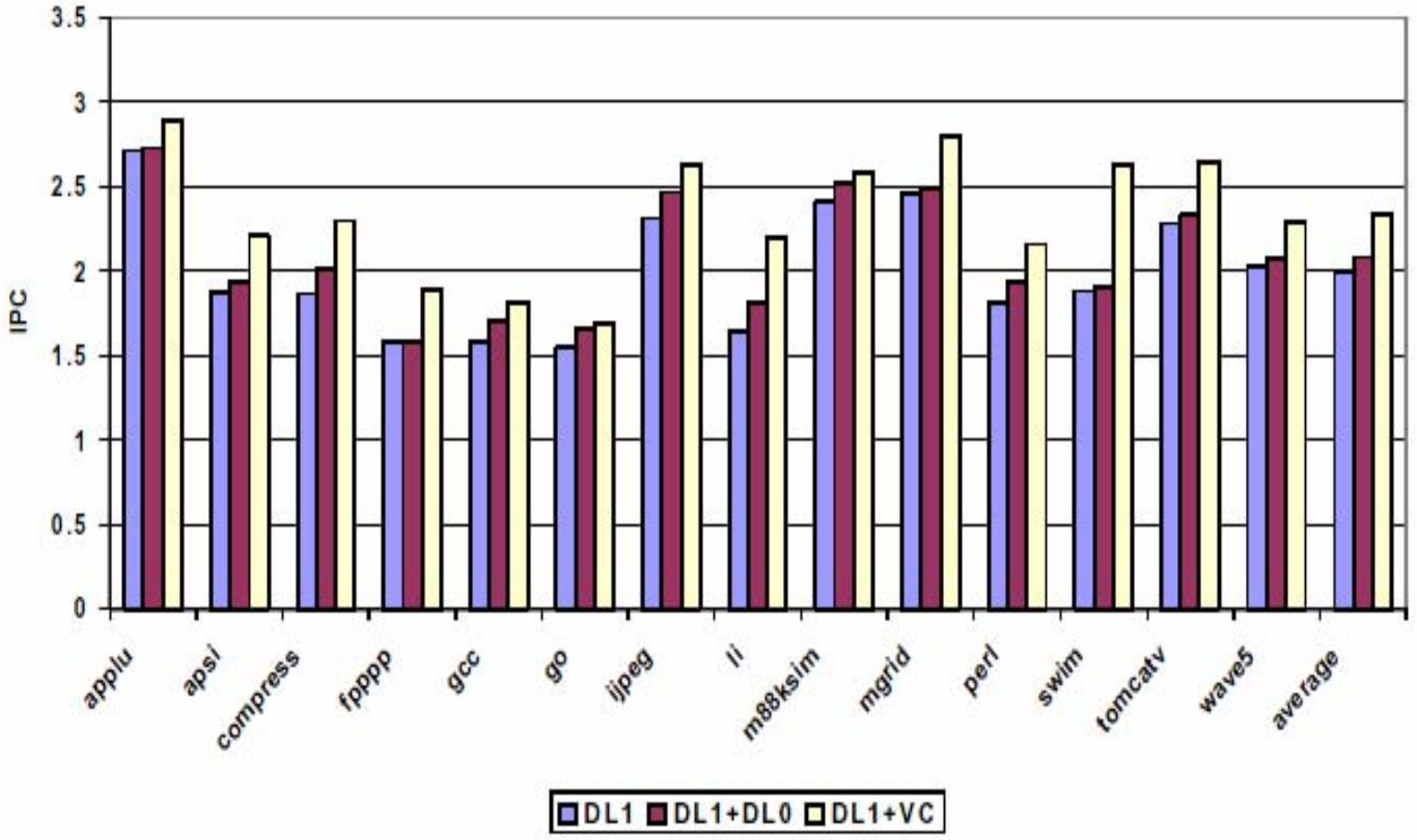
Paper 4: Classification FSM



Paper 4: Cache Hierarchy



Paper 4: Evaluation



Paper 4: Critique

Prons

- Vital Cache improves performance
- Shows more promise than previous paper

Cons

- Does not explore slack
- Still need an optimality study

Summary

- Criticality and Slack can be exploited to guide policies
- Correct classification is key
- Are hardware implementations practical?
- Criticality at the memory system is still challenging
  - Should we give up hope?