Towards Soft Optimization Techniques for Parallel Cognitive Applications

Woongki Baek, JaeWoong Chung, Chi Cao Minh
Christos Kozyrakis, Kunle Olukotun

Computer Systems Laboratory
Stanford University
http://tcc.stanford.edu

Outline

- Introduction
- Soft computing properties
- Soft optimizations
- Case study: Loopy Belief Propagation (LBP)
- Conclusions & future works

Introduction

- Cognitive applications are widely used in areas ranging from commercial fields to military and security applications.
- To allow the analysis of ever increasing data sets, development of efficient parallelization techniques is important.
- This talk discusses:
  - A set of soft optimization techniques that build upon soft computing properties of cognitive applications.
  - How the proposed optimizations target common bottlenecks for parallel execution.
  - Case study with Loopy Belief Propagation (LBP).

Soft Computing Properties

- Exact vs. soft computing
  - Conventional applications operate on precise data and have strict accuracy requirements.
  - Cognitive applications process inherently noise inputs, must handle uncertainty, and produce an acceptable approximation of the “correct” answer.
- Soft computing properties:
  - User-defined, relaxed correctness
    - E.g., a small percentage of misclassification rate is tolerable.
  - Redundancy
    - E.g., computations and communications on converged nodes.
  - Inherent adaptivity to errors
    - E.g., noisy data from sensor nodes in online object tracking.

Soft Optimizations (1)

- O1) Reducing computation
  - Target: reducing excessive workload per thread due to large data sets.
  - Possible optimizations:
    - Data dropping
    - Lazy computation
    - Solution pruning
- O2) Mitigating imbalance
  - Target: mitigating performance degradation caused by imbalances
    - Work imbalance
    - Region imbalance
  - Possible optimizations:
    - Adaptive workload discarding
    - Selective barriers

Soft Optimizations (2)

- O3) Reducing communication
  - Target: reducing expensive communications among threads on large-scale parallel systems.
  - Possible optimization:
    - Adaptive communication
- O4) Reducing synchronization
  - Target: avoiding or reducing uses of synchronization primitives.
  - Possible optimizations:
    - Imprecise updates
    - Removing synchronizations

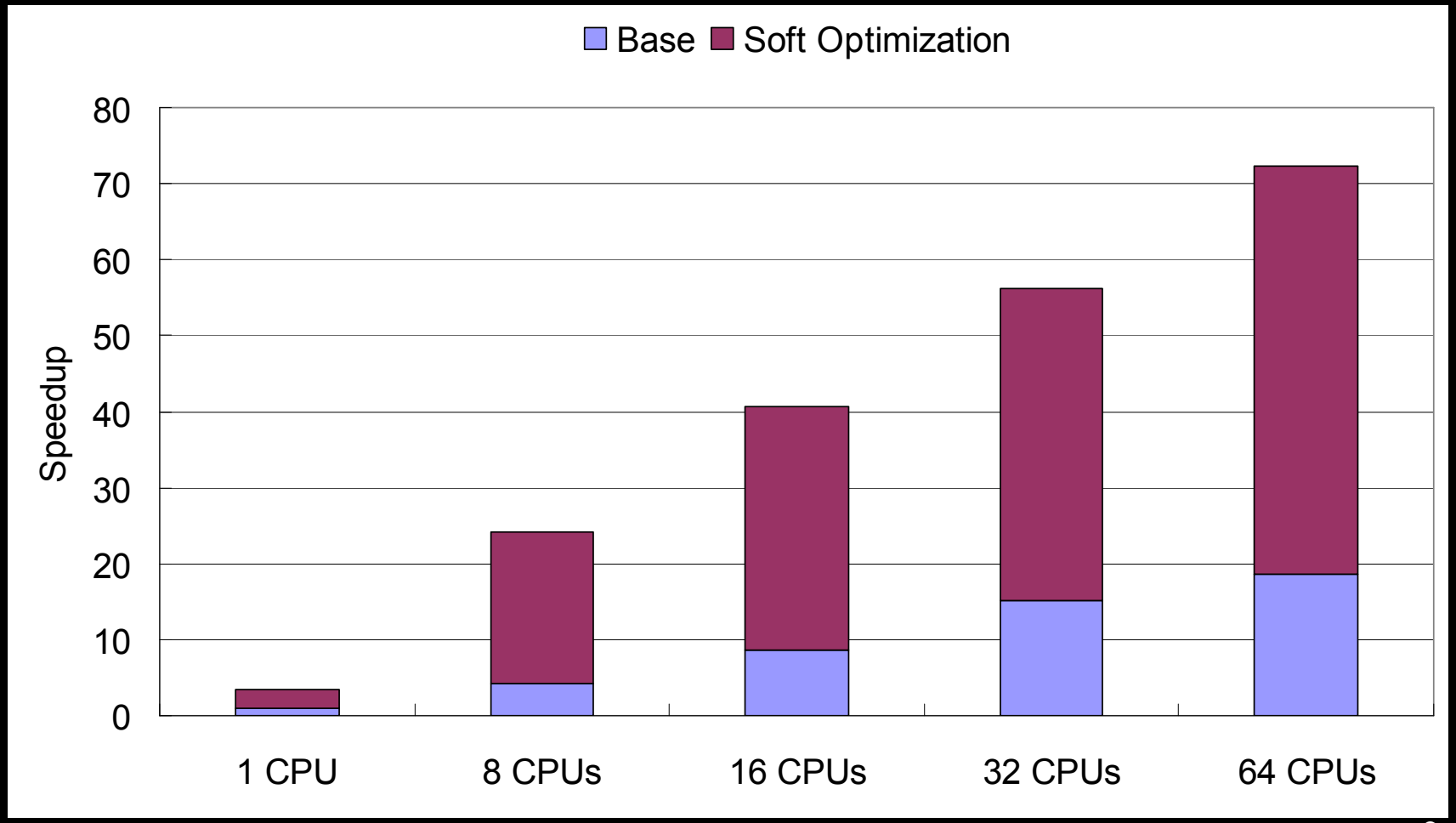
Loopy Belief Propagation (LBP)

- Efficient approximation for probabilistic inference on graphical models
  - Probabilistic inference is known to be NP-hard.
  - Computes “beliefs” (i.e., approximation of exact marginal probabilities) of nodes using local message passing.
  - Time complexity grows linearly with the number of nodes.
  - Not guaranteed to give the correct beliefs on networks with loops, but works well under a wide range of conditions in practice.
- Two key operations to optimize:
  - Message computation
  - Belief computation

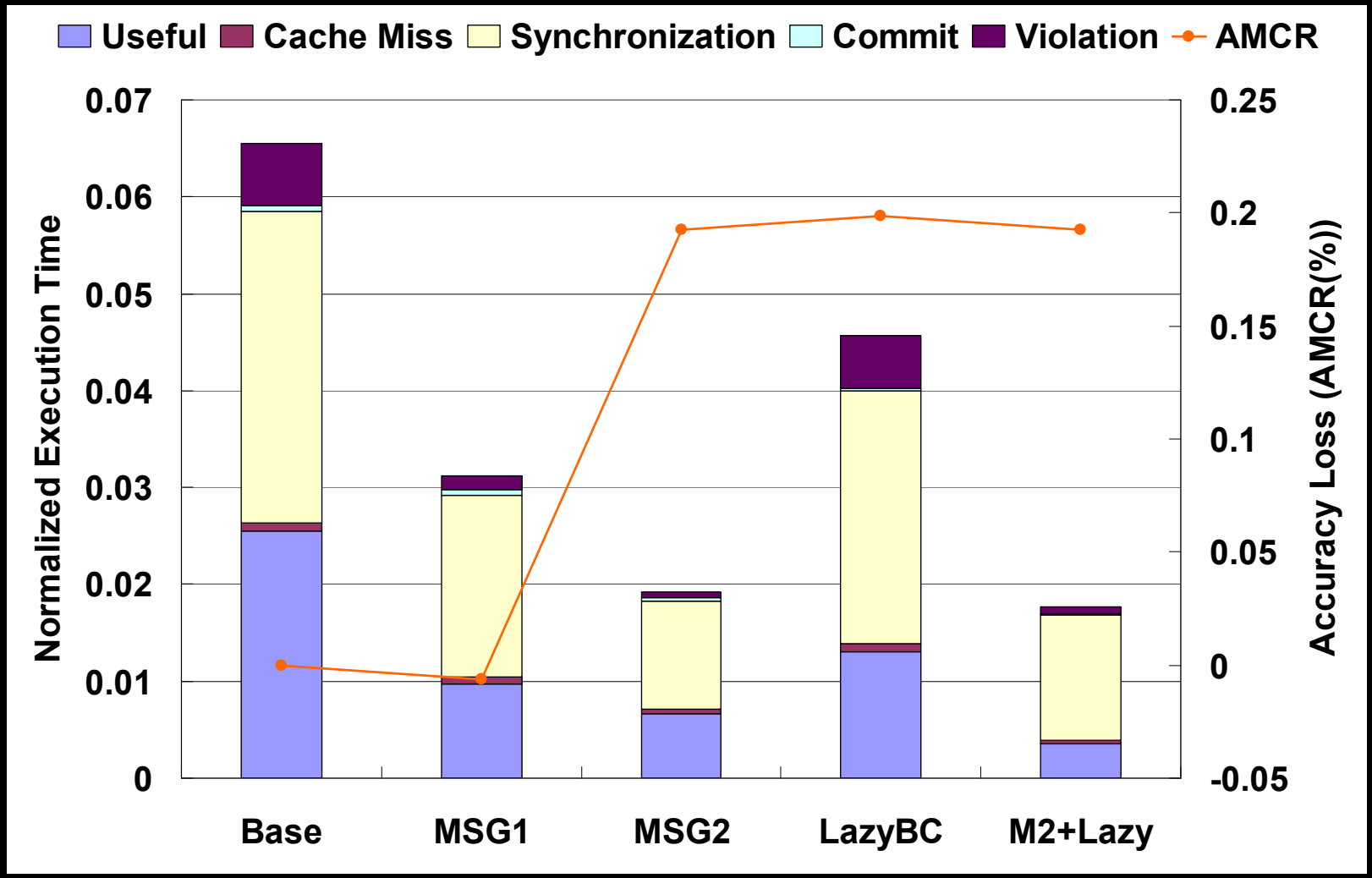
Optimizations & Evaluations

- Soft optimizations on LBP
  - Adaptive message version 1 (MSG1)
    - Does not compute and send messages when both a sender and a receiver have converged.
    - Target: computations, communications, and imbalances
  - Adaptive message version 2 (MSG2)
    - Does not compute and send messages when only a sender has converged.
    - Target: computations, communications, and imbalances
  - Lazy belief computation (LazyBC)
    - Does not compute belief of a node when the difference in beliefs from the previous and the current iterations is within in a threshold.
    - Target: computations and imbalances
- Evaluation metrics (base vs. soft-optimized)
  - Performance: execution time and scalability
  - Accuracy loss: additional misclassification rate (AMCR)
    - AMCR = (# of misses with soft opt - # of misses with base)/# of total classifications * 100

Scalability



Performance and Accuracy Analysis (32 CPUs)



Conclusions & Future Works

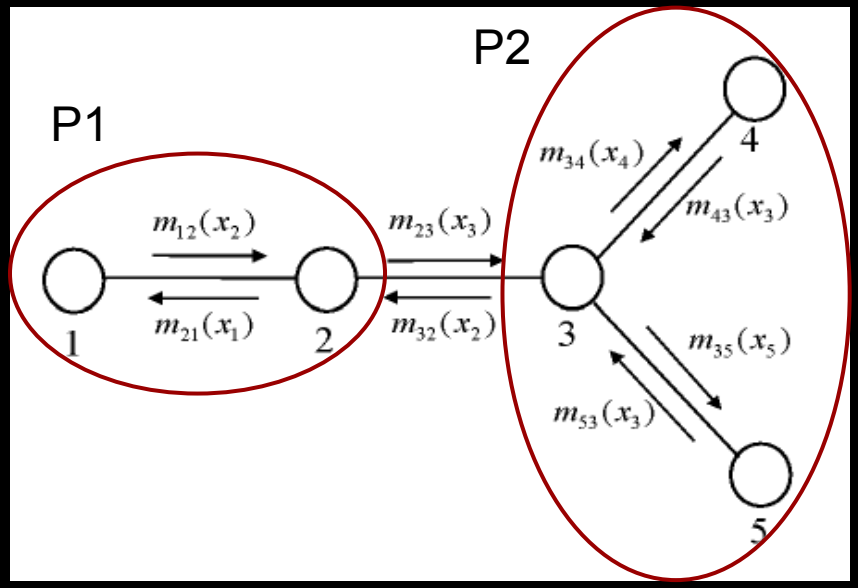
- We investigated soft optimizations for parallel cognitive applications by exploiting soft computing properties.
- We demonstrated soft optimizations lead to large performance improvements (3.7x) with little accuracy impact for LBP.

- Future works
  - Evaluation of wider range of parallel cognitive applications (currently, under review for publication)
  - Theoretical approaches for soft optimizations
    - E.g., how to ensure convergence?, which work is safe to drop?, etc.
  - Further research into parallel programming languages, runtime systems, and architectures for easy use of soft optimizations

Thanks & Questions?

Woongki Baek (wkbaek@stanford.edu)

Backup: Parallel Execution of LBP



- Pseudo code of parallel LBP

```
/* G: entire network */
/* SubV: set of all nodes in each
   sub-group */
if(master_thread){
 partition_and_distribute(G,SubV);
}
while(iter<MAX_ITER){
 /* compute messages */
 for node[i] in SubV[thr_id]
  for e(i,j) in adj(node[i])
   compute_and_send_msg(&m(i,j));
 barrier();
 /* compute beliefs */
 for node[i] in SubV[thr_id]
  compute_bel(&node[i].bel);
 barrier(); iter++;
}
```

Backup: Simulated Architecture

Feature	Description
CPU	1–64 single-issue PowerPC cores
L1 Cache	64KB, 32-byte cache line 4-way associative, 1 cycle latency
L2 Cache	1MB, 32-byte cache line 4-way associative, 16 cycle latency
ICN	2D grid topology, 14 cycle link latency
Main Memory	100 cycle latency up to 8 outstanding transfers
Directory	Full-bit vector sharer list; first touch allocate; Directory cache 10 cycle latency