Introduction

Recently, approximate storage emerges in the area of computer architecture. It trades off precision for better performance and/or energy. Previous studies have demonstrated the benefits for applications that are tolerant to imprecision such as image processing. However, it is still an open question whether and how approximate storage can be used for applications that do not expose such intrinsic tolerance.

In this paper, we study one of the most basic operations in database, sorting, on a hybrid storage system with both precise storage and approximate storage.

Contributions:
1. The first to leverage precise computing on approximate storage.
2. A novel approx-refine mechanism to guarantee precise sorting on hybrid-memory machines.
3. Experimental results show that approximate storage can improve performance of sorting by up to 11%.

Background

We study:
- Mergesort
- Quicksort
- LSD Radixsort
- MSD Radixsort

Hardware

Address translation and allocation are enhanced with OS and assembly support.

Architecture

Unmodified

Memory bus

CPU

Cache Hierarchy

Precise PCM

Approximate PCM

I/O bus

Storage System

Algorithm

We study:
- Mergesort
- Quicksort
- LSD Radixsort
- MSD Radixsort

Approx-Refine Mechanism

Warm-up

Approx Preparation

Approx Stage

Refine Preparation

Find LISREMID

Refine Stage

Merge REMID/LIS

• Sorting algorithms are accelerated on approximate memory in the approx stage. Most algorithms get almost sorted sequences after the approx stage.
• Nearly sorted sequences are refined to be strictly sorted on precise memory in the refine stage.

Challenges of the refine stage:
- overhead of copying data between approximate and precise memory is not negligible
- must introduce as few memory writes as possible

Solutions of a lightweight refine algorithm:
- fully make use of presortedness of the sequence Key
- use heuristics to find an approximate LIS in O(n)
- only introduce 2n+Rem(Key) memory writes in total

Evaluation

1. We showcase that approximate storage can be not only be used for approximate computing, but also improving performance and/or energy efficiency of precise computing.
2. We evaluate four sorting algorithms on hybrid storage systems. We demonstrate great potential of approximate storage to provide speedup with handful inaccuracies.
3. It is a nontrivial task to refine imperfect results. Refine algorithms should make fully use of the marginal unsortedness and introduce negligible overhead.

Conclusion

1. We showcase that approximate storage can be not only be used for approximate computing, but also improving performance and/or energy efficiency of precise computing.
2. We evaluate four sorting algorithms on hybrid storage systems. We demonstrate great potential of approximate storage to provide speedup with handful inaccuracies.
3. It is a nontrivial task to refine imperfect results. Refine algorithms should make fully use of the marginal unsortedness and introduce negligible overhead.