# CHERI-picking: Leveraging capability hardware for pre-fetching

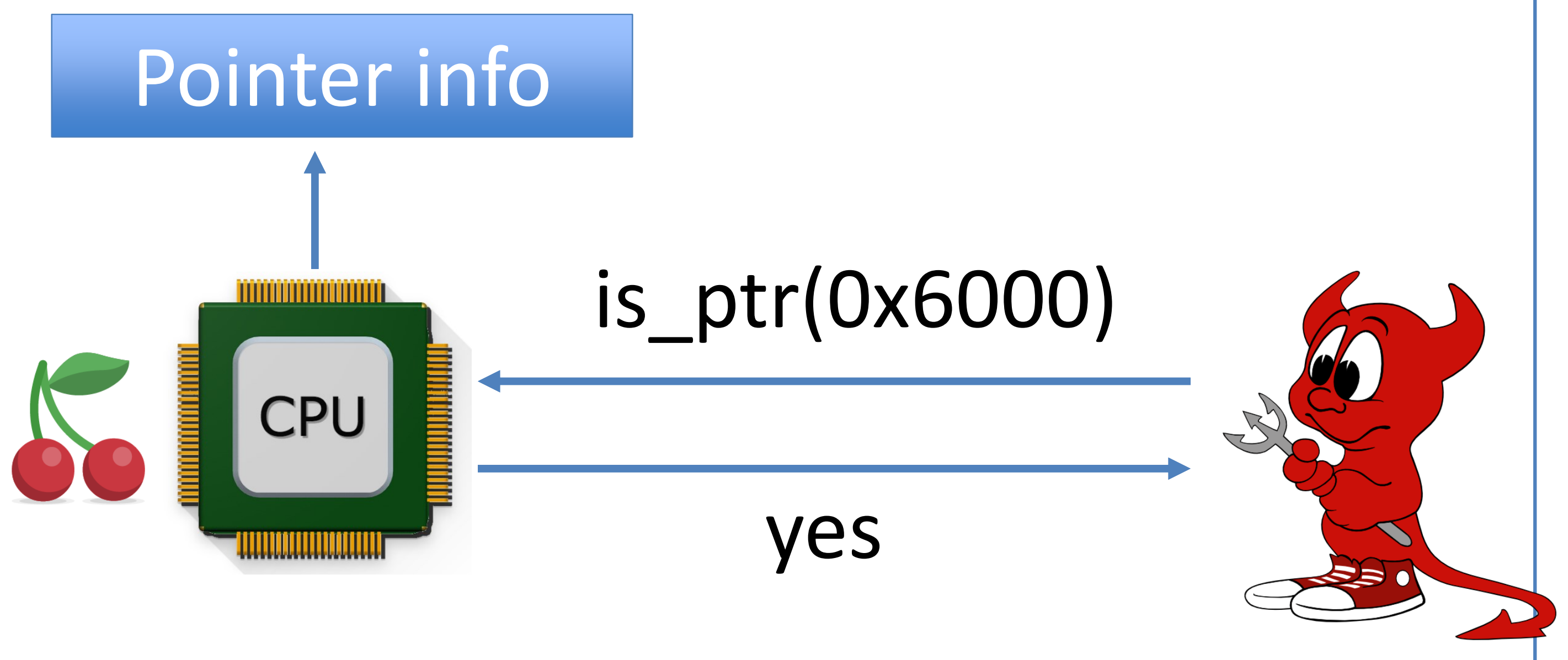
Shaurya Patel, Sidhartha Agrawal, Alexandra Fedorova, Margo Seltzer

## Problem

- ❑ DRAM is **expensive** and **memory demands** of datacenter applications **are increasing**.
- ❑ Far memory **is cheap** but **degrades application performance**, prefetchers are important to hide latency of memory access.
- ❑ Applications with **Linked data structures** are difficult to prefetch because of their pointer-chasing behavior.

## Insightful data from hardware

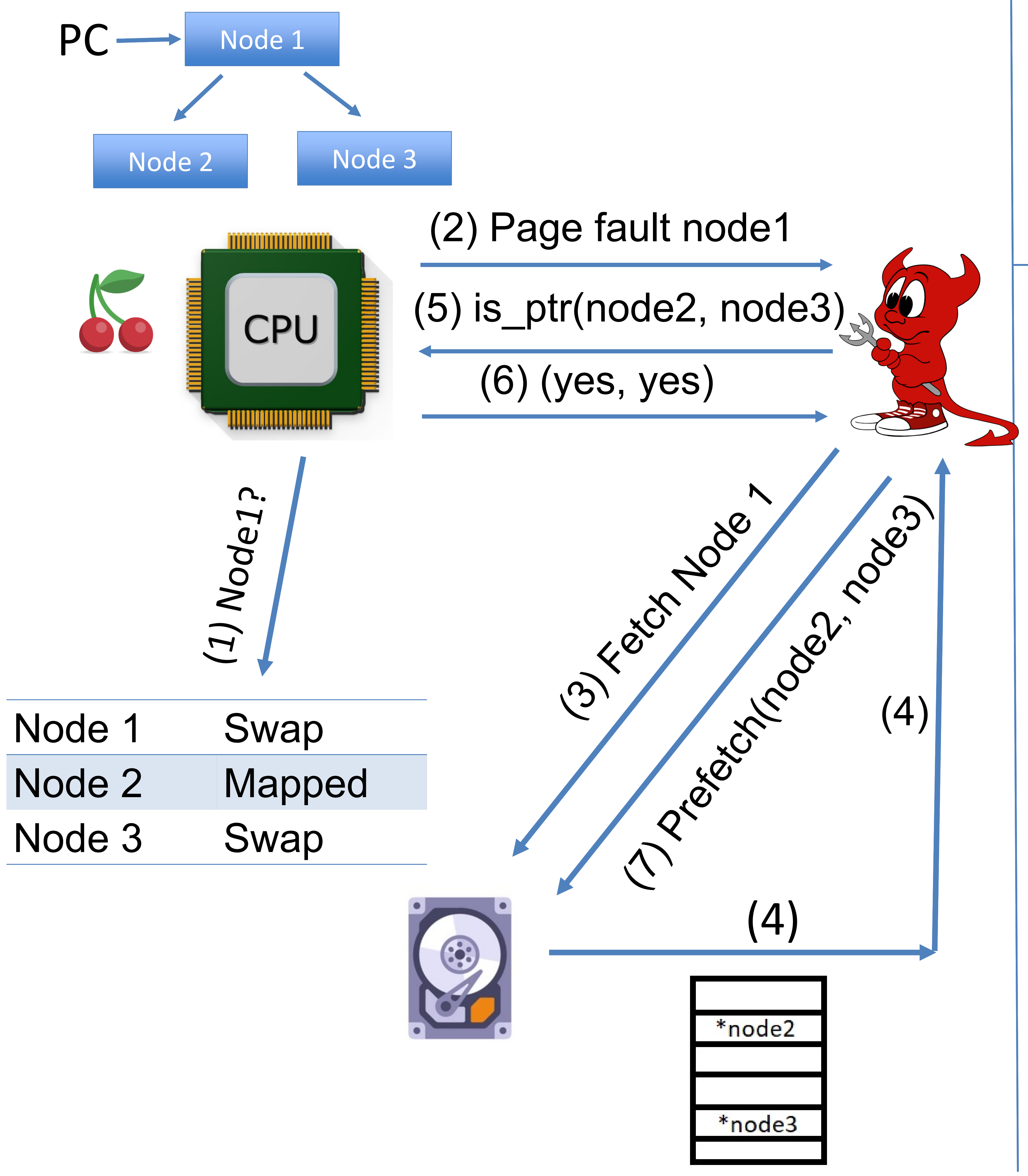
- ❑ Capability hardware systems such as CHERI makes it possible to identify pointers at runtime.



## Our Approach

- ❑ Analyze application traces to classify percentage of pagefaults caused by pointer accesses.
- ❑ Leverage CHERI data at runtime to make OS pre-fetching decisions.

## Prefetching system for pointer data



| | |
|---|---|
| Node 1 | Swap |
| Node 2 | Mapped |
| Node 3 | Swap |

## Preliminary results

- ❑ We benchmark the system by using a linked list traversal.
- ❑ We used a random permutation of numbers to assign linked list nodes, each node is page size.
- ❑ Total memory consumption is 4GB and we constrained it at 50%.

| | Default prefetcher | Pointer based prefetcher | Improvement |
|---|---|---|---|
| Cache hits | 46749 | 655345 | 12.9X ↑ |
| #major faults | 1.86m | 1.41m | 24% ↓ |
| Execution time | 21.85 minutes | 20.56 minutes | 5.91% ↑ |

## Future work

- ❑ Use application execution traces to identify programs for which pointer-based prefetching can be effective.
- ❑ Evaluate the pre-fetcher CHERI-Morello hardware using CheriBSD.
- ❑ Develop real-time monitoring to identify programs for which we should do pointer-based prefetching.