CACHE MEMORY MANAGEMENT BY SPLAY TREES

KEDAR	(221IT023)
ASHISH R K	(221IT012)
VINAY	(221IT079)
K JAGGARAO	(221IT037)

PROBLEM STATEMENT

REAL SYSTEMS REQUIRE EFFICIENT CACHE MANAGEMENT TO HANDLE RAPID DATA ACCESS. TRADITIONAL SPLAY TREE METHODS STRUGGLE IN MULTI-THREADED ENVIRONMENTS DUE TO CONCURRENCY CONFLICTS AND UNBALANCED TREE STRUCTURES. THIS PROJECT PROPOSES A CACHE MEMORY MANAGEMENT SYSTEM THAT ADDRESSES THESE CHALLENGES BY USING SPLAY TREE DATA STRUCTURE.

OBJECTIVES

- Implementing thread-safe modifications and locking techniques for concurrent access.
- Employing adaptive balancing strategies to optimize performance based on access patterns.
- Utilizing fine-grained locking mechanisms to minimize contention and enhance access efficiency.

The project plans to create a cache memory management system that improves performance and responsiveness in real-time multi-threaded applications by achieving concurrency control, optimized balancing, and reduced contention while ensuring data consistency.

PROPOSED SOLUTION

MULTI THREAD: Thread share the memory and the resources of the process to which they belong.

SYNCHRONIZATION: The multiple threads should be synchronized for avoid conflict

FINE-GRAINED LOCKING: Explore using separate mutex's for different parts of the tree structure.

METHODOLOGY

- Utilize a Splay Tree data structure for fast access and dynamic ordering.
- Exploring the Limitations of Splay Tree By Creating user made Threads and allowing them to do insertion and access the data Thus stimulating multiprocessing environment .
- With the use of critical session problem implementing the mutex to lock and unlock critical session and preventing race condition.

THANK YOU

INTRODUCTION

- Primary memory has faster access times than secondary memory, but it still takes a few microseconds, while the CPU can perform operations in nanoseconds.
- Cache is a fast-access memory used to store frequently accessed data for quicker retrieval, insertion and eviction by the processor.
- Cache memory is smaller, a large amount of data cannot be stored
- It's interesting to know that Splay Trees is a data structure that eliminates the need which require maintaining additional data structures to keep track of access times or access frequencies for each item in the cache unlike traditional cache algorithms like LRU (Least Recently Used) or LFU (Least Frequently Used)

SPLAY TREE IN CACHE MANAGEMENT

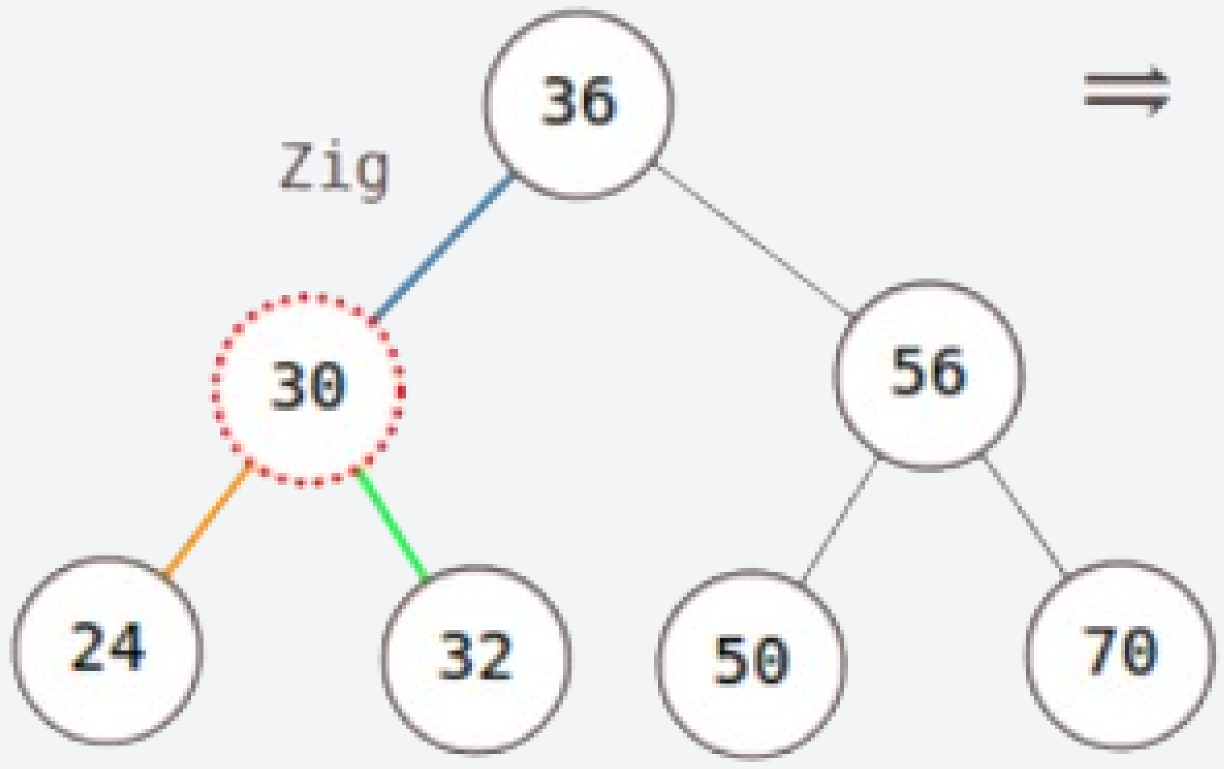
- Splay tree is a self-adjusting binary search tree that automatically reorganizes itself so that frequently accessed or inserted elements are moved closer to the root node.

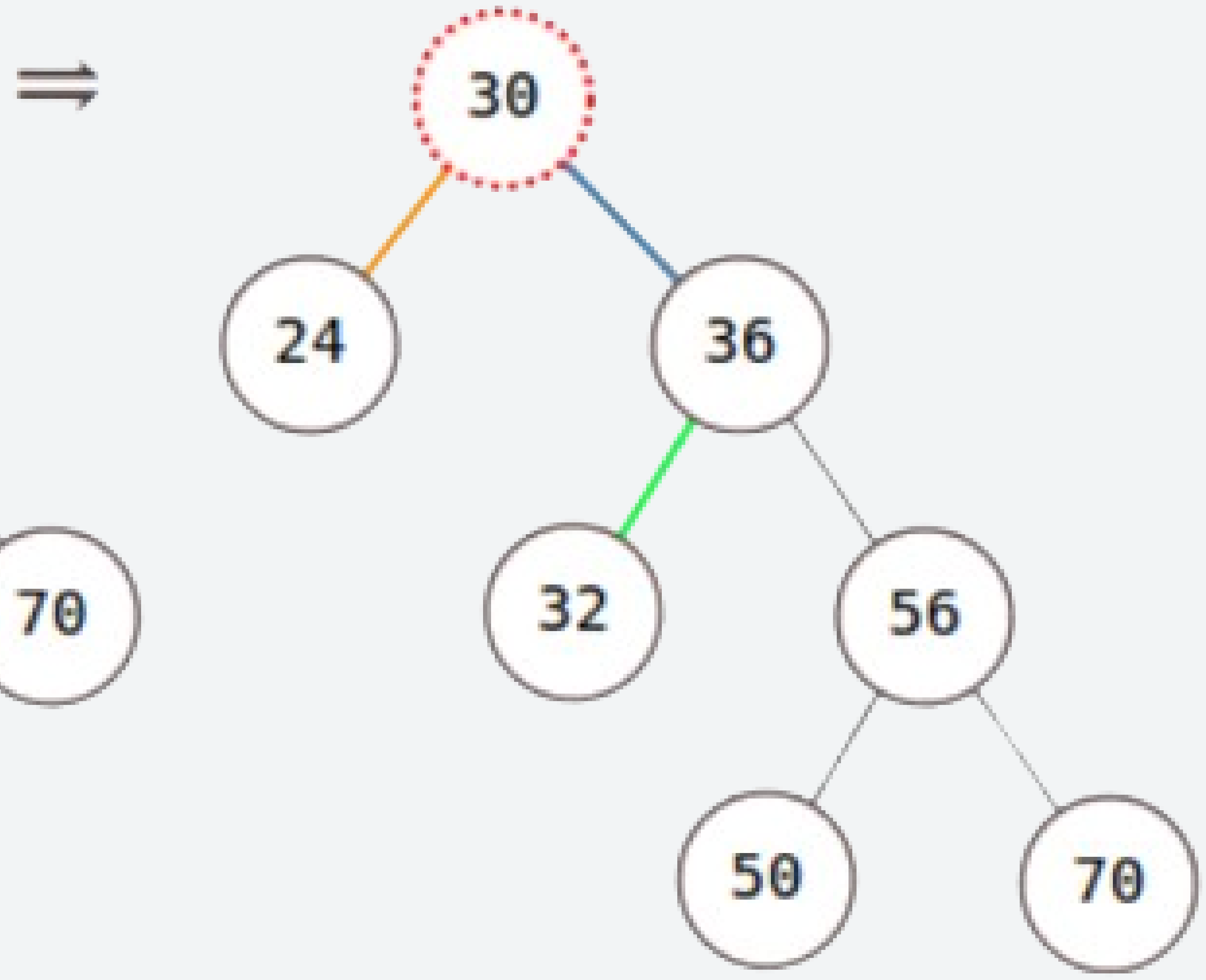
- Same as the cache where we need to faster accessing of a process which we accessed earlier .

- Eviction can be done easily using splay trees as the process that is least recently used which would typically be the item at the bottom of the splay tree (the farthest from the root).

Splay Tree Demonstration

After search for node 30 ,it become root node





Before

After

Expected Outcomes

- Enabling concurrent access to the Splay Tree Cache from multiple threads resulting in efficient key-value storage.
- Ensuring that data is consistent and preventing race conditions.
- Achieve efficient performance and scalability in multi-threaded scenarios.