The research problem is to develop an assembly code search engine for reverse engineering. Assembly code analysis is one of the critical processes for detecting and proving software plagiarism and software patent infringements when the source code is unavailable. It is also a common practice to discover exploits and vulnerabilities in existing software. However, it is a manually intensive and time-consuming process.

Assembly code analysis is one of the key processes for detecting and proving software plagiarism and software patent infringements when the source code is unavailable. It is also a common practice to discover exploits and vulnerabilities in existing software. However, it is a manually intensive and time-consuming process.

The KamlnO engine is designed for general key-value storage and the Apache Spark computational framework. Its solution stack, as shown in Figure 2, consists of three layers. The data storage layer is concerned with how the data is stored and indexed. The distributed/focal execution layer manages and executes the jobs submitted by the engine. The KamlnO engine splits a search query into multiple jobs and coordinates their execution flow.

Figure 8. The index structure for the Adaptive Locality Sensitive Hashing (ALSH).

The search process consists of the three steps: Preprocessing: After parsing the input (either a binary file or assembly functions) into control flow graphs, this step normalizes assembly code into a general form. Find basic blocks clone pairs: Given a list of assembly basic blocks from the previous step, it finds all the clone pairs of blocks using adaptive locality sensitive hashing. Search the subgraph clones: Given the list of clone block pairs, the MapReduce module searches and constructs the subgraph clones. Note that this clone search process does not require any source code.

We construct a new labeled one-to-many assembly code clone dataset that is available to the research community by linking the source code and assembly function level clones.

We benchmark the performance of twelve existing state-of-the-art solutions on the dataset using three typical information retrieval metrics (Table 1). KamlnO boosts the clone search quality and yields stable results across different datasets and metrics.

We also setup a mini-cluster (4 nodes) to evaluate the scalability of KamlnO (Figure 8).

KamlnO is open-source and available on GitHub. Scan the code to check out KamlnO and the paper.