BUZZARD: A NUMA-Aware In-Memory Indexing System
Lukas Maas, Thomas Kissinger, Dirk Habich and Wolfgang Lehner

Motivation

Hardware with non-uniform memory access (NUMA) characteristics becomes increasingly common

In-memory index performance is mainly bound by memory access latency and bandwidth

Index performance on NUMA hardware heavily depends on thread and data placement

NUMA-Aware Index Partitioning

Partitioned global index (prefix tree)

Mapped to local memory of NUMA nodes

Requests are handled by worker threads that run on the different NUMA nodes (1 thread per partition).

Benefits for index operations:
- Only local memory accesses necessary
- No index synchronization needed
- Improved cache utilization

Request Processing

1. Find partition
2. Write request to local buffer
3. Flush local buffer
4. Process batch

Request processing in BUZZARD comprises the following general steps:
① Determine the partition of the given request’s key
② Insert request into an unsynchronized thread-local intermediate buffer
③ Periodically flush the local buffers to the main request buffers
④ Worker threads, running on the different NUMA nodes, process multiple requests as a batch, hiding memory latency by interleaving multiple index lookups

Load Balancing

- Most real world data is not uniformly distributed
- Data access patterns may change over time
- Load balancing helps to avoid under-utilization of worker threads (reduced overall throughput)

First Results

BUZZARD performs consistently better than a single global index that uses the same index structure (PTX – a generalized prefix tree with a prefix length of 4 bit) with and without batch processing to hide memory latency

Threads either generate new requests or handle requests on an assigned partition:
- Too many generator threads → worker threads become bottleneck
- Too many worker threads → worker threads wait for new requests

This work is supported by the German Research Foundation (DFG) in the Collaborative Research Center 912 “Highly Adaptive Energy-Efficient Computing”.

http://www.db.inf.tu-dresden.de/research