**Dual locality caching**

Consider the following technique using LRU algorithm and its data structure.

In LRU, newly fetched blocks enter into its stack top, and replaced blocks

leave from its stack bottom. There are two key operations in the DULO scheme.

(1) Forming sequences is one of the key operations. A *sequence* is defined as a

number of blocks whose disk locations are close to each other and have been

accessed continuously without an interruption during a limited time period.

2 Sorting sequences in an LRU stack according

to their recency (temporal locality) and size (spatial locality) with the objective

that sequences of large recency and size are close to the LRU stack bottom.

Block table is the data structure for implementing dual locality.

The block table is analogous in structure to the multilevel page

table used to process address translation.

There are three levels in the example block table: two directory levels and one

leaf level. The table entries at different levels are fit into different memory pages. An entry at the

leaf level is called Block Table Entry (BTE).

the block table covers disk space in the unit of block

where a logical block number (LBN) of a block is the index into the table.

In the system, we set a global variable called a *disk access clock*, which ticks

each time a block is fetched into memory and stamps the block being fetched

with the current clock time . We then record the timestamp in an entry at the

leaf-level of the block table, which is determined by the LBN of the block.

When the sequencing bank is full, it is time to examine blocks in the bank to

aggregate them into sequences.

The DULO algorithm associates each sequence with an attribute *H*, where

a relatively small *H* value indicates its associated sequence should be evicted

earlier. The algorithm has a global inflation value *L*, which is initiated as 0.

When a new sequence *s* is admitted into the eviction section, its *H* value is set

as *H*(*s*) = *L* + 1*/size*(*s*), where *size*(*s*) is the number of the blocks contained in

*s*. When a sequence is evicted, we assign the *H* value of the sequence to *L*. So

*L* records the *H* value of the most recently evicted sequence. The sequences

in the eviction section are sorted by their *H* values with sequences of small *H*

values at the LRU stack bottom. In the algorithm, a sequence of large size tends

to stay at the stack bottom and to be evicted earlier. However, if a sequence of

small size is not accessed for a relatively long time, it would be replaced. This

is because a newly admitted long sequence could have a larger *H* value due to

the *L* value, which keeps being inflated by evicted blocks. When all sequences

are random blocks (i.e., their sizes are 1), the algorithm degenerates into the

LRU replacement algorithm.

.

Evaluation

To demonstrate the performance improvements of DULO on a modern operating

system, we implement it in the recent Linux kernel 2.6.11

Benchmarks

1. *TPC-H* is a decision support benchmark that runs business-oriented queries

against a database system

*2. diff* is a tool that compares two files in a character-by-character fashion

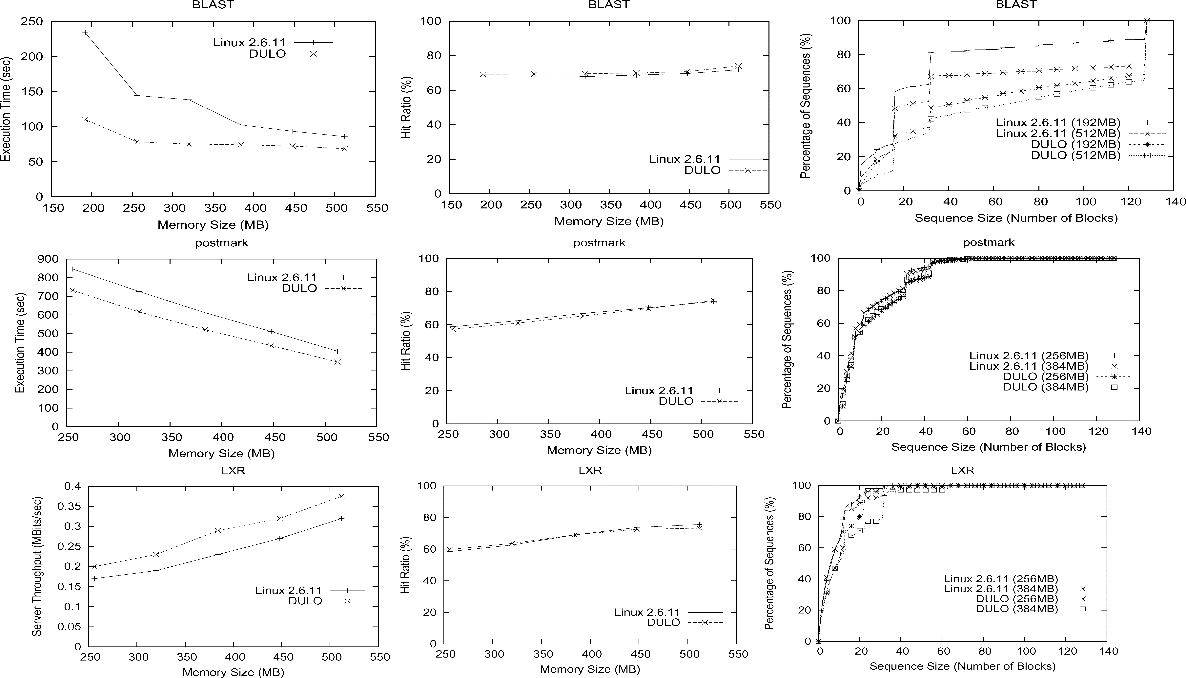
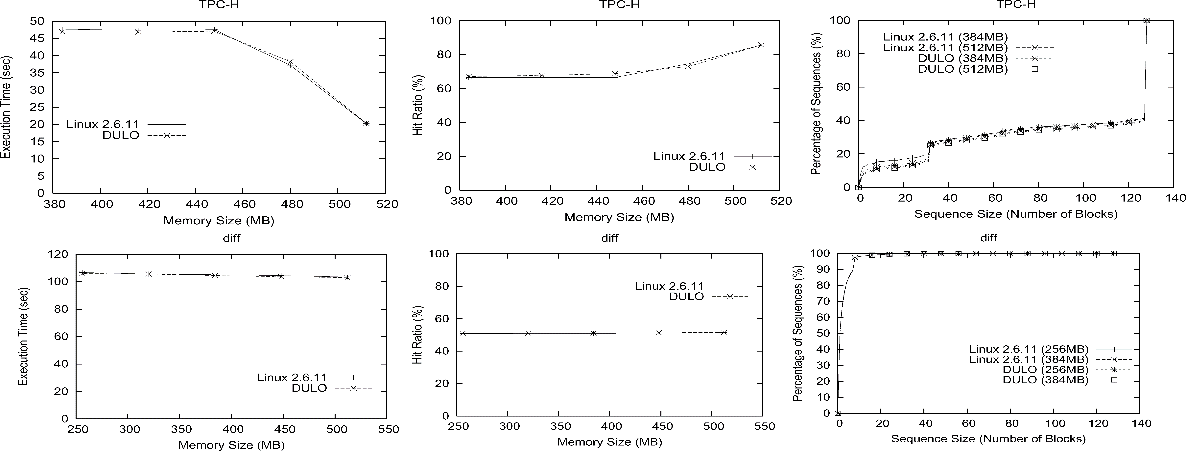
*3. BLAST* (basic local alignment search tool) is software from the National Centre for Biotechnology Information

4. *Postmark* is a benchmark designed by Network Appliance to test performance

of systems,

5. *LXR* (Linux cross-reference) is a widely used source code indexer and crossreferencer

[LXR ].



In Figure 4, the CDF curves show that in workload TPC-H more than 85% of

the sequences are longer than 16 blocks. For this almost-all-sequential workload,

DULO has limited influence on the performance. It can slightly increase

the sizes of short sequences, and accordingly reduce execution time by 2.1%

with a memory size of 384MB. However, for the almost-all-random workload

*diff*, more than 80% of the sequences are shorter than 4 blocks. Unsurprisingly,

DULO cannot create sequential disk requests from application requests consisting

of purely random blocks. As expected, we see almost no improvements

of execution times by DULO.

The other three benchmarks have a considerable amount of both short sequences

and long sequences