University POLITEHNICA of Bucharest
Computer Science & Engineering Department

Performance Optimization in the Age of Shifting Bottlenecks

Laura Mihaela Vasilescu, Costin Raiciu
University POLITEHNICA of Bucharest, Romania
laura.vasilescu@cs.pub.ro, costin.raiciu@cs.pub.ro

- I/O devices have historically lagged behind the CPU
- rigid assumptions about bottlenecks (caching, compression, etc.)
- CPU cores struggles to keep up with I/O rates

Hypothesis: software performance optimizations should explicitly account for shifting bottlenecks during application deployment.

1. Web Proxy

We implemented a customizable web proxy app that allows us to serve HTML files from disk or memory; it also supports compression, if the client supports it (as indicated in the HTTP request). The proxy is multi-threaded. To increase performance, our app stores recently loaded web pages in memory in a fixed size cache and uses the LRU algorithm to evict old entries. When there is a LRU miss, our app can be configured to:

- load the file from **disk**, whether in plain text or compressed format.
- **decompress**, if the compressed version is already in memory; cache the results.
- **compress** the file and serve it if the raw file is already in memory; cache the result.

2. Experimental Setup

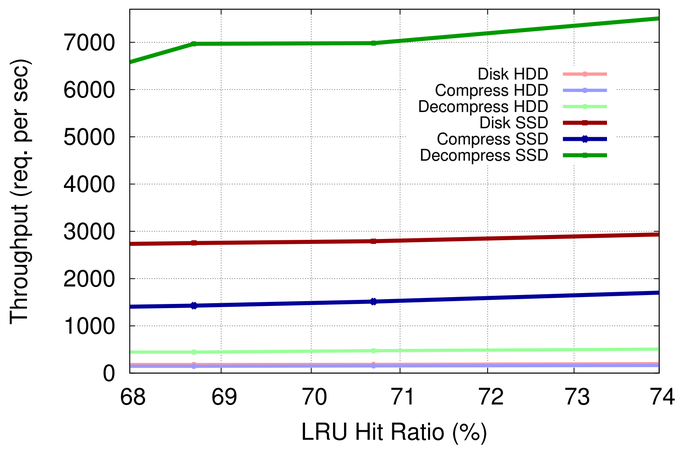
- Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz, 62GB RAM
- one magnetic disk and one solid-state disk
- server and clients communicate through the loop-back interface
- server and clients run on different NUMA nodes
- server threads are equally distributed among the available cores
- clients are equally distributed among the available cores
- dataset contains Wikipedia pages with size above 500 KB

3. Initial Results

We used ten clients that maintain long running connections and repeatedly request a random file, randomly selecting between plain text or compressed format.
The table below shows the total server throughput (req. per sec) in different scenarios.

HDD (.html)	HDD (.gz)	SSD (.html)	SSD (.gz)	LRU=∞	Compress	Decompress
45	241	634	3665	26785	672	3405

4. HDD vs. SSD

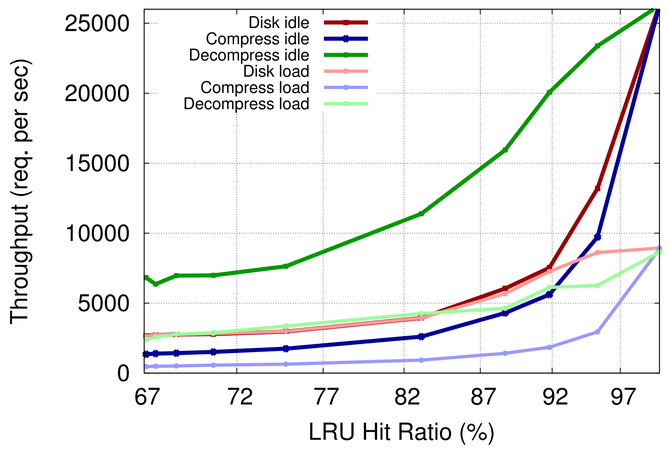


- requests are randomly generated with the ZIPF distribution:

$$P(page_{id}) = \frac{[num_files]}{\rho^{id}}, \rho = 1.5$$

- Decompress > Disk > Compress
- magnetic disk highly degrades performance even when LRU hit ratio is big

5. Idle vs. Load



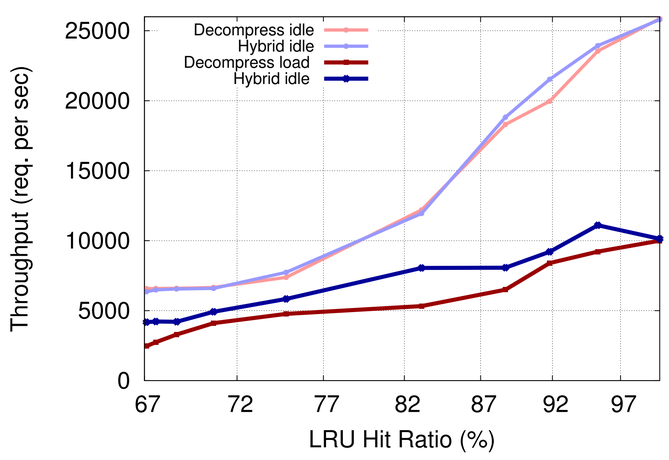
- exclusive access (idle):

  Decompress > Disk > Compress
- non-exclusive access (load), with LRU hit ratio > 82%:

  Disk > Decompress > Compress
- non-exclusive access (load), with LRU hit ratio < 82%:

  Disk $\approx$ Decompress > Compress

6. Hybrid



- double number of clients
- **hybrid** - if another thread running on the same core is doing decompression (CPU intensive), change the policy to read the .gz file from disk
- Hybrid > Decompress
- performance of the hybrid algorithm is with 68% better if the LRU hit ratio is smaller than 82%