Processing Posting Lists Using OpenCL

Advisor
Dr. Chris Pollett

Committee Members
Dr. Thomas Austin
Dr. Sami Khuri

By
Radha Kotipalli

Agenda

- Project Goal
- About Yioop
- Inverted index
- Compression Algorithms
- Encoding & Decoding
- PHP Extensions
- OpenCL
- Test Results
- Conclusion

Project Goal

- Improve the Posting Lists' performance of Yioop search engine using PHP Extensions (C and OpenCL)
- Identify the resource intensive functions in the existing Yioop's code
    - Replacing them with PHP C-Extensions
    - Modify with OpenCL-Extensions, where the parallelism can benefit most

About Yioop

- Designed and developed by Dr. Pollett
- PHP- based search engine
- Open source
- Search engines use inverted index
- Uses Modified9 compression algorithm to encode and decode posting lists

Inverted Index

- An index data structure (inverted/postings file)
- Maps contents to its locations in document database
- Consists of two components
  - Dictionary: unique words that appear in all documents
  - Posting Lists: [doc_id, index positions of the word]
    - Stored in a compressed binary format

Inverted Index …

- Uncompressed inverted index can be larger than original data
- Compressed
  - Less storage requirement
  - Fast query retrieval time
  - Can accommodate large collections
- Compression Algorithm
  - Encoder
  - Decoder : Lossy & Lossless

Posting Lists

- Larger part of inverted index consists of Posting Lists
- Contains unique large number of elements
- Monotonically increasing index positions
- Replace with Δ-values
  - Difference between consecutive elements
    - Smaller numbers
    - Encoded with fewer bits
- Two compression algorithms
- Non-parametric: Does not consider the Δ-values in a given posting list
  - Ex: Elias's γ-code
- Parametric: considers the specific characteristics of the list to be compressed
  - Ex: Golomb/Rice code

Byte & Word-Aligned codes

- To improve compression and decompression speed
  - Look at codes so that the split between code words falls on byte or word boundaries
- Two types :
  - Byte-Aligned
  - Word-Aligned

Byte-Aligned codes

- vByte (variable-byte coding)
  - Splits the binary representation of each Δ-value into 7-bit chunk + 1 bit continuation flag
- Ex: **L = (1624, 1650, 1876, 1972, 2350, ...)**

  Δ (L) = (1624, 26, 226, 96, 384 ...)

  Binary format of 1624: 11001011000

  1 1011000 **0** 0001100 **0** 0011010 **1** 1100010 **0** 0000001 **0** 1100000 **1** 0000000 **0** 0000011...

  0 at the beginning of the chunk indicates the end of the current code word. (88 + 12* 2^7 = 1624)

Word- Aligned codes

- Inspects the postings list's Δ-values and tries to insert as many consecutive Δ-values as possible into a 32-bit machine word
- Simple-9 algorithm
  - 4 bits for a selector (tells # of Δ-values stored)
  - 28 bits for Δ-values

Selector	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**9**
Number of Δ's	1	2	3	4	5	7	9	14	28
Bits per Δ	28	14	9	7	5	4	3	2	1
Unused bits/word	0	0	1	0	3	0	1	0	0

Word-Aligned codes …

- Ex : L = (1624, 1650, 1876, 1972, 2350, ...)
- Δ (L) : 1624 25 225 95 383 [Δ-value: [1650 - 1624 -1] = 25 ...]
- The above indexes can be saved as 1624 and 25 together as two 14-bits each;
- 225, 95, and 383, together as three 9-bits each, and one unused bit at the end.

Yioop Encoding

- Uses Modified9 algorithm for compression
- Modified9 is similar to Simple-9, inserts as many consecutive Δ-values as possible into a 32-bit machine word
- First 2 bits tell whether or not to look at the next word
  - 11 start of encoded string
  - 10 continue four more bytes
  - 01 end of encoded
  - 00 indicates the whole sequence encoded in one word

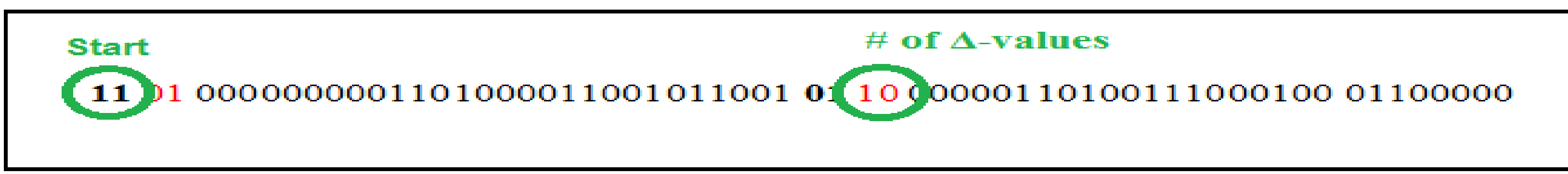
Yioop Encoding …

- Next most significant bits represents selector
- Selector can be 2, 4, 5, or 6 bits

Selector	00 (0)	01 (1)	10 (2)	1100 (12)	1101 (13)	1110 (14)	11110 (30)	111110 (62)	111111 (63)
Number of Δ's	1	2	3	4	5	6	7	12	24
Bits per Δ	28	14	9	6	5	4	3	2	1
Unused bits	0	0	1	0	1	2	4	0	0

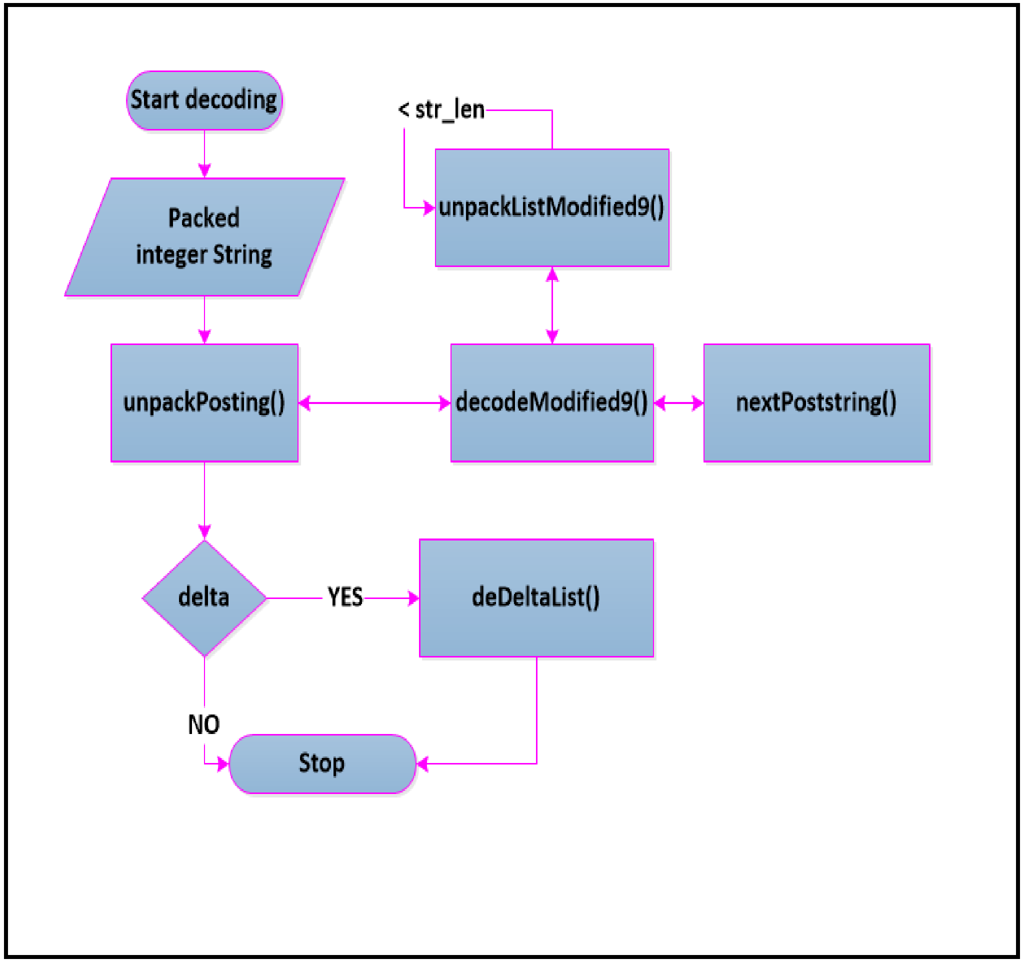
Yioop Encoding

- A typical posting list: [doc_index, index positions]
- **Ex:** L: [25, [1624 1650 1876 1972 ...]] (doc_index: 25)
  Δ-values: [1624, 26, 226, 96, ...] [Δ-value: [1650 -1624] = 26, ...]
- The above indexes can be saved as 26 and 1625 together as two 14-bits each in one 4-byte word
- 26, 226, and 96 together as three 9-bits each, and one unused bit in one 4-byte word
- The final encoded string:

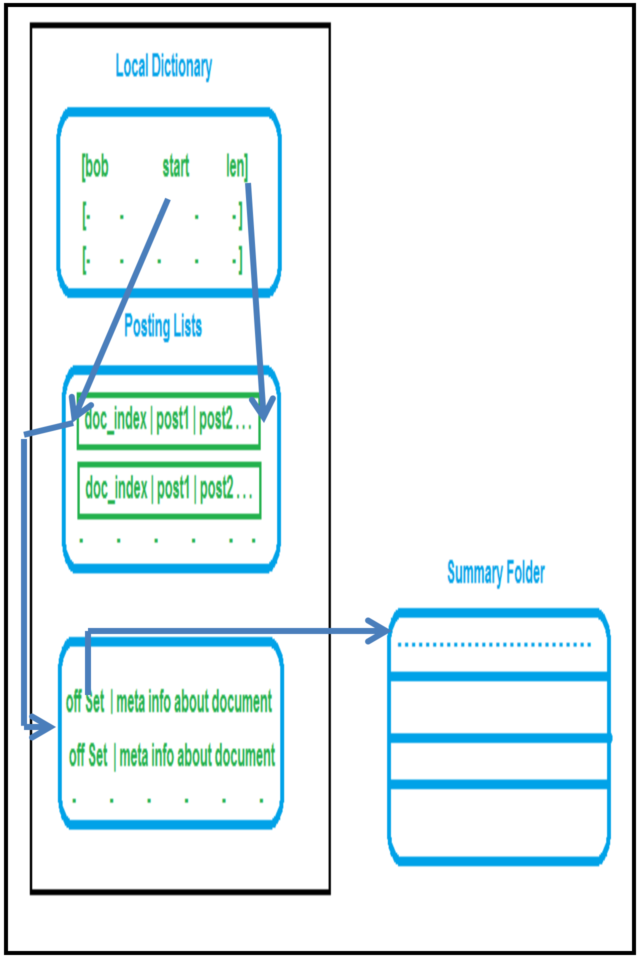


- **Hex String: D0 06 86 5A A0 65 C4 60**

Yioop Decoding



Decoding Flowchart



Index shard

Yioop Decoding …

- The unpackPosting() is the starting point
- The nextPoststring() identifies the complete posting string from the given packed integer string of a posting list by checking first two MSB of each 4-byte string (“11” for start and “01” for the end)
- The unpackListModified9() takes 4-byte string at a time, removes first two MSB bits and then observes the next bits to identify the number of Δ-values in that 4-byte string
- Decodes the string back into its Δ-values according to Modified9.
- Repeats the process until the end of the complete posting string to get back all the Δ-values
- The *deDeltaList()*, converts the values back into the original index positions.
- Returns [doc_index, [index postions]]

PHP Extensions

- A way to customize or extend the default functionality of PHP
- Two types of Extensions
  - Standard Extensions : comes with PHP distribution
    - MySQL, cURL etc.
  - Zend Extensions : Can be written in
    - Java
    - C/C++

Why do we need Extensions?

- Customize or introduce new functionality
- To improve performance
- To hide the proprietary source code
- Reuse of existing code

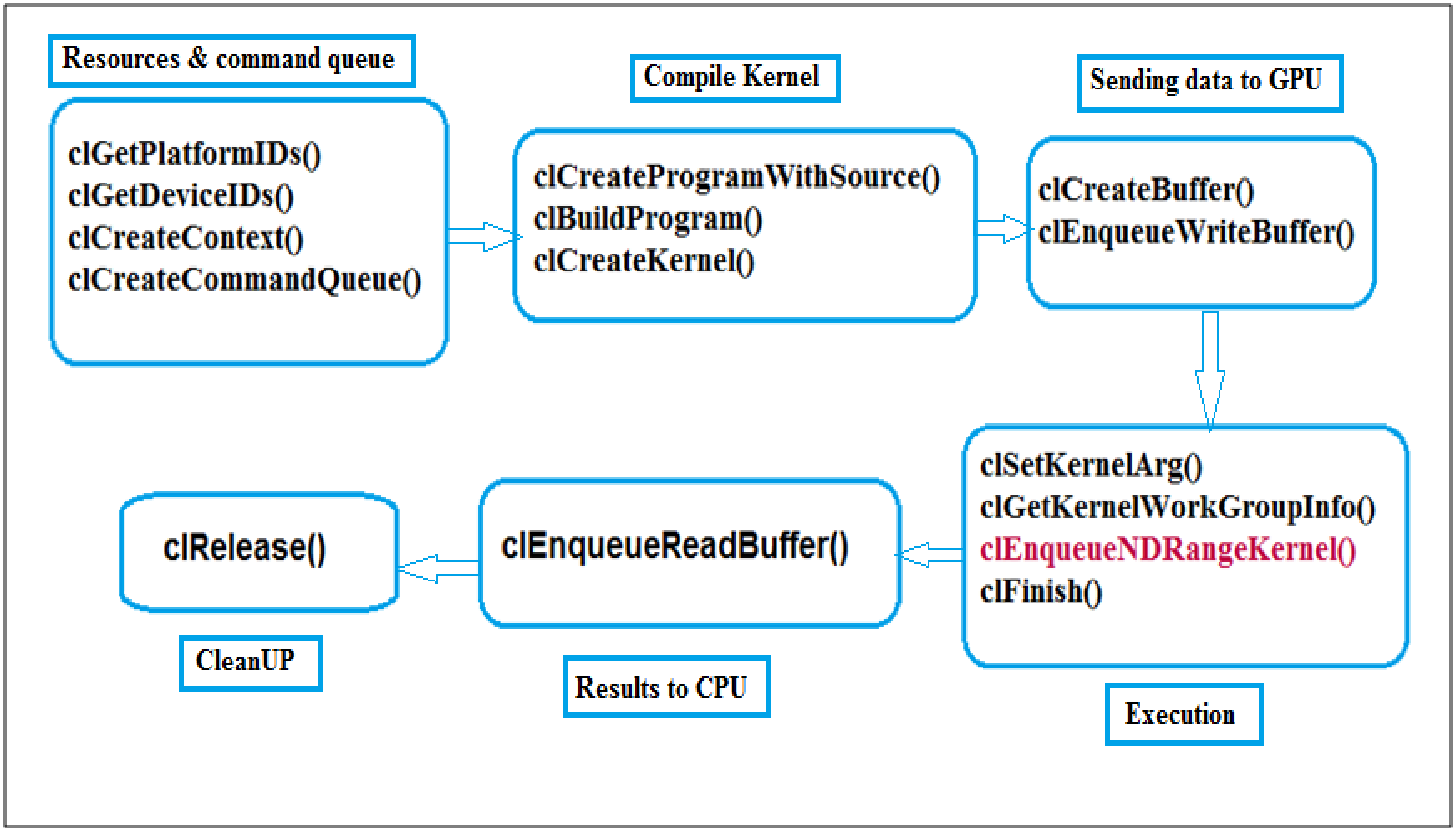
OpenCL (Open Computing Language)

- Framework built specifically for parallel processing over heterogeneous systems
- Parallel programs can be written in C-language and can exploit the power of GPU threads
- Portability across multiple platforms
- Host Code: C/C++ code run on CPU
  - To transfer data between memories (CPU & GPU)
  - To execute device/kernel code (GPU)
- Kernel/Device code: Executes on GPU

OpenCL Program Flow

1. Organize resources, Create command queue
2. Compile Kernel
3. Transfer data from host(CPU) to GPU memory
4. Launch threads running kernels on GPU, Perform main computation
5. Transfer data back to host memory from GPU
6. Free allocated memory

OpenCL Program Flow …



Creating PHP Extensions

Some of the important practices while writing PHP extensions

- Zend function to read input data

```
zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
        "las|b", &doc_index, &position_list, &str, &str_len, &delta)
```

- Letter 'l' represents a variable type long which is used to read an integer or long values.
- Letter 'a' represents a variable type array
- Letter 's' represents a variable type string
- Letter 'b' represents a boolean value type.
- Symbol '|' is given in front of the variable type, if the parameter is an optional.

Creating PHP Extensions …

- To convert *zval** array structure into regular C array

```
for(zend_hash_internal_pointer_reset_ex(arr_hash, &pointer);
    zend_hash_get_current_data_ex(arr_hash, (void**) &data, &pointer)
    == SUCCESS; zend_hash_move_forward_ex(arr_hash, &pointer))
    {
        pos_list.arr[i] = Z_LVAL_PP(data);
        i++;
    }
```

Creating PHP Extensions …

- To send one of the arguments as pass-by-ref, it needs to be declared in the arginfo structure ahead

```
ZEND_BEGIN_ARG_INFO_EX(unpackPosting_arginfo, 0,
                                        ZEND_RETURN_VALUE, 2)
ZEND_ARG_INFO(0, posting) // 0 means "passed by value"
ZEND_ARG_INFO(1, off_set) // 1 means "passed by reference"
ZEND_END_ARG_INFO();
```

PHP Vs C Extensions Functions

- All the functions involved in the Yioop’s encoding and decoding are replaced with PHP C Extensions and additional functionality has been introduced to match with the existing PHP code.
- String length and array length are used frequently in the PHP code, the same was achieved through a struct.
- Same is done for shift functions

PHP Built-in	C
str_len()	struct { chr*, int len}
array len()	struct {int*, int len }
array_shift()	C_arrary_shift()
array_unshift()	C_array_unshift()

OpenCL code

- The deltaList() and unpackListModified() were implemented in OpenCL
- The first two steps shown in OpenCL program flow are needed for each OpenCL function call
- To reduce above overhead, wrote a function using global variables and initialized at startup instead of calling for each function call

Test Variations

- Languages: PHP, C, and OpenCL Extensions
- Processors: *i5* + Intel HD Graphics, *i7* + Nvidia
- Version: PHP 5 and PHP 7
- Bits: 32 and 64 bit
- Documents Size: 10,000 and 100,000
- Rank (term frequency) : 13, 310, and 3000

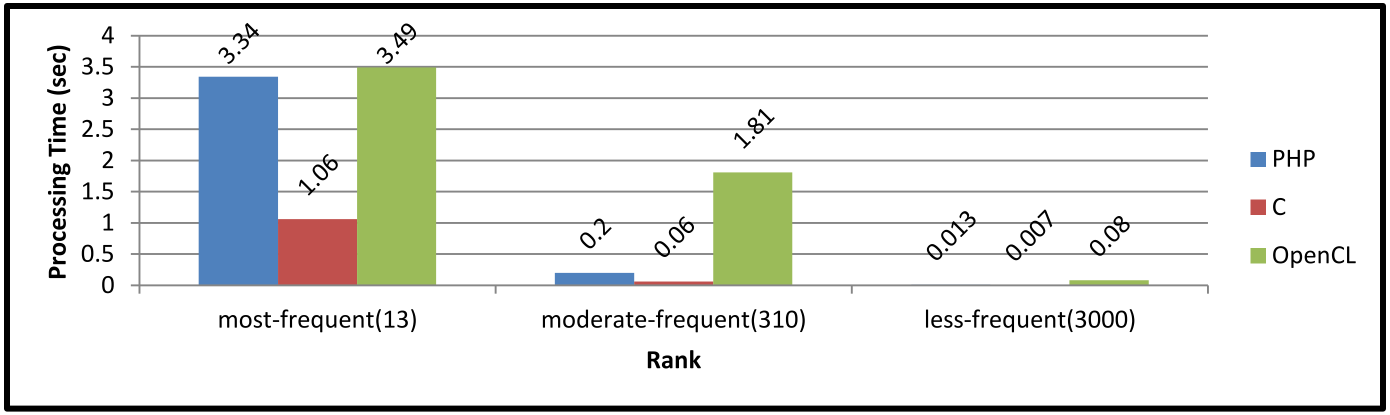
PS: Most frequent rank 13 was chosen between (1 -100), moderate frequent 310 was chosen between (100 -1000), and less frequent 3000 was chosen between (1000 - 10000) randomly through a program

Test Scenarios

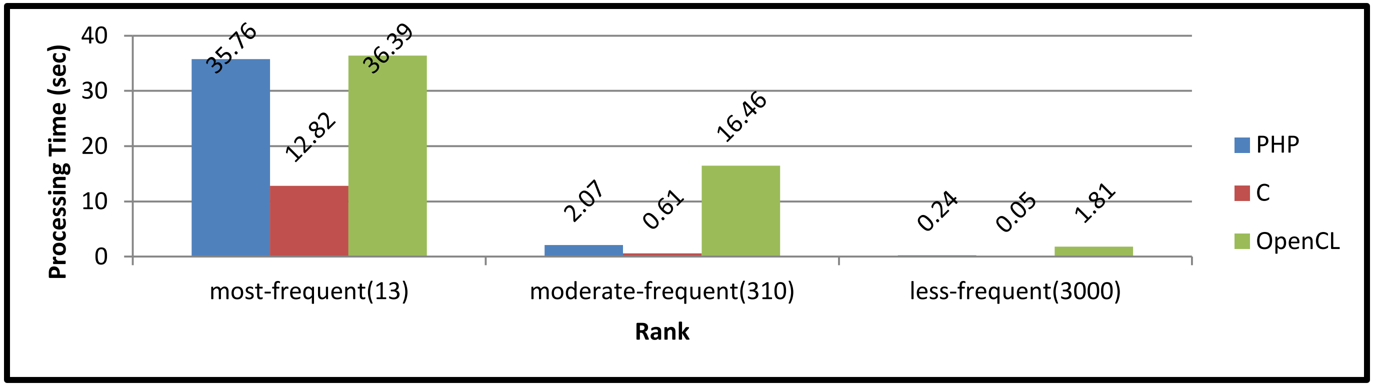
With above combinations of variations the following scenarios were tested

- Encoding: Measures the time taken to encode postings for a given rank
- Decoding: Measures the time taken to decode the above encoded string
- Browser-based testing: Measures the time taken for a search

Encoding Test Results

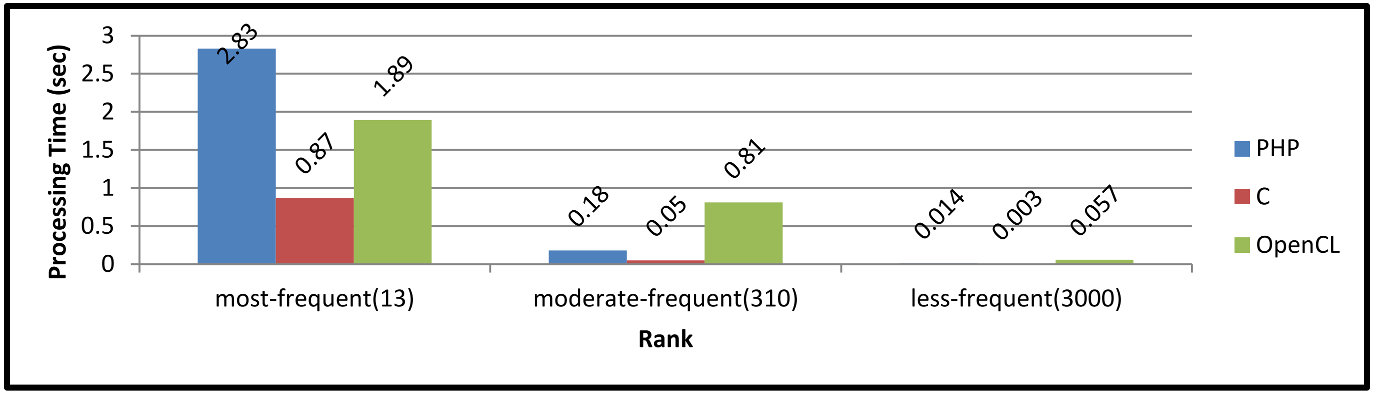


Encoding Test Results for10,000 documents (PHP5, 32 bit, i5+HD GPU)

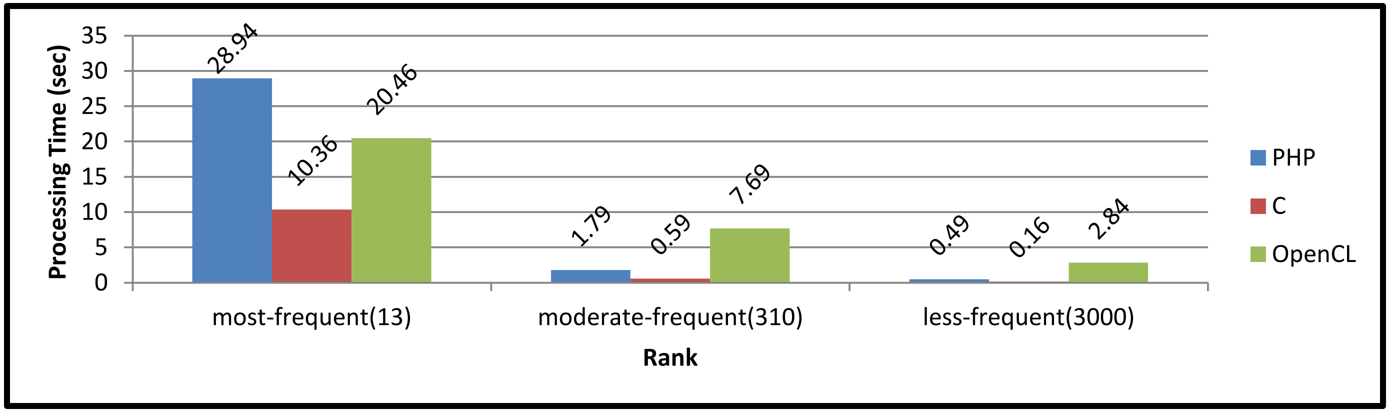


Encoding Test Results for100,000 documents (PHP5, 32 bit, i5+HD GPU)

Encoding Test Results …

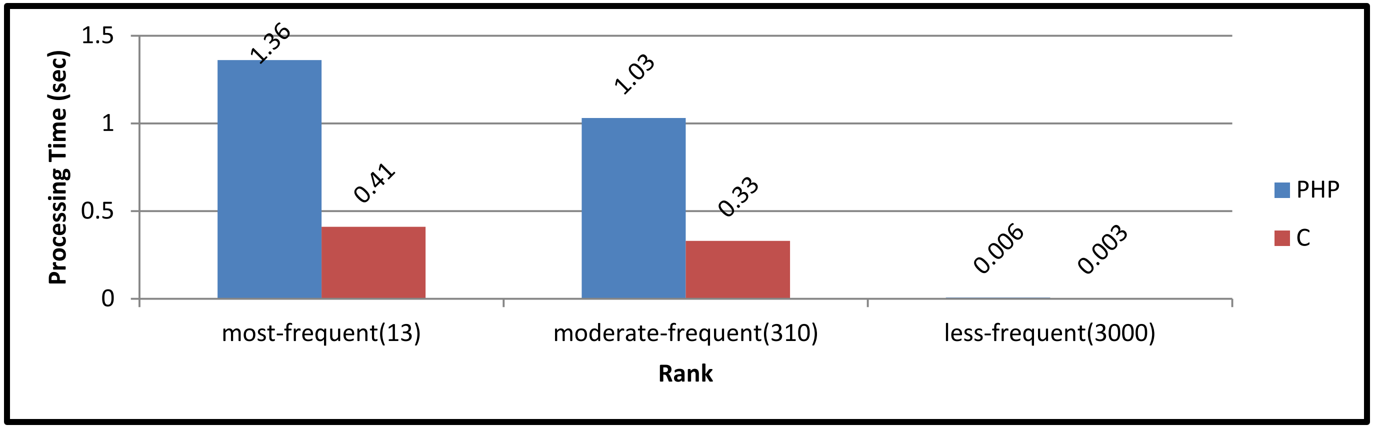


Encoding Test Results for10,000 documents (PHP5, 32 bit, i7+Nvidia GPU)

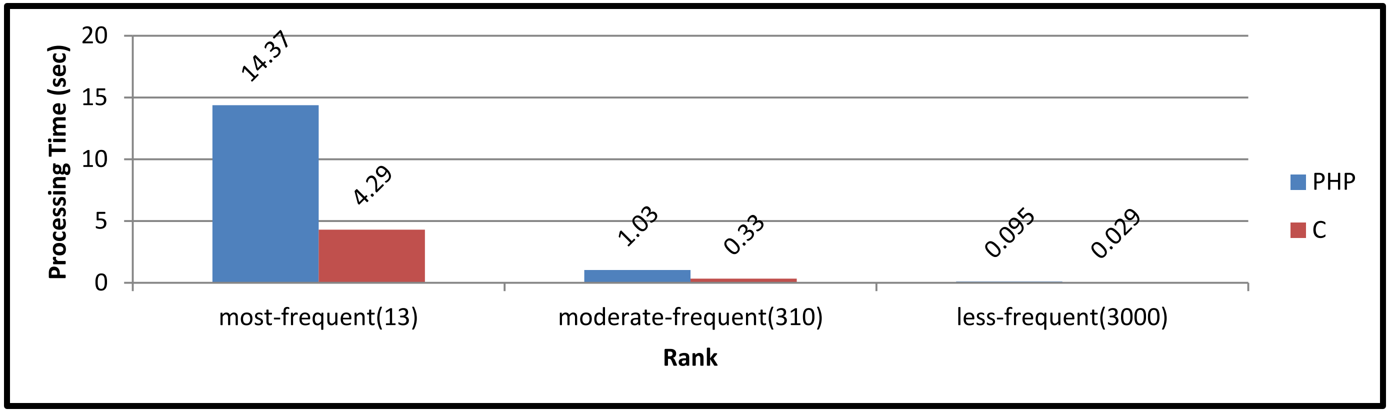


Encoding Test Results for100,000 documents (PHP5, 32 bit, i7+Nvidia GPU)

Encoding Test Results …

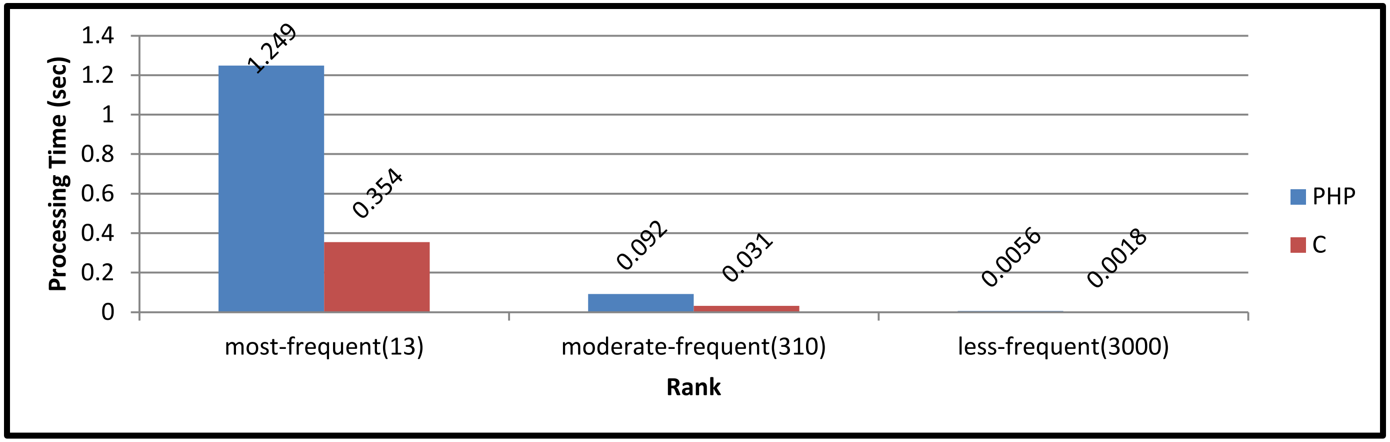


Encoding Test Results for 10,000 documents (PHP7, 32 bit, i5)

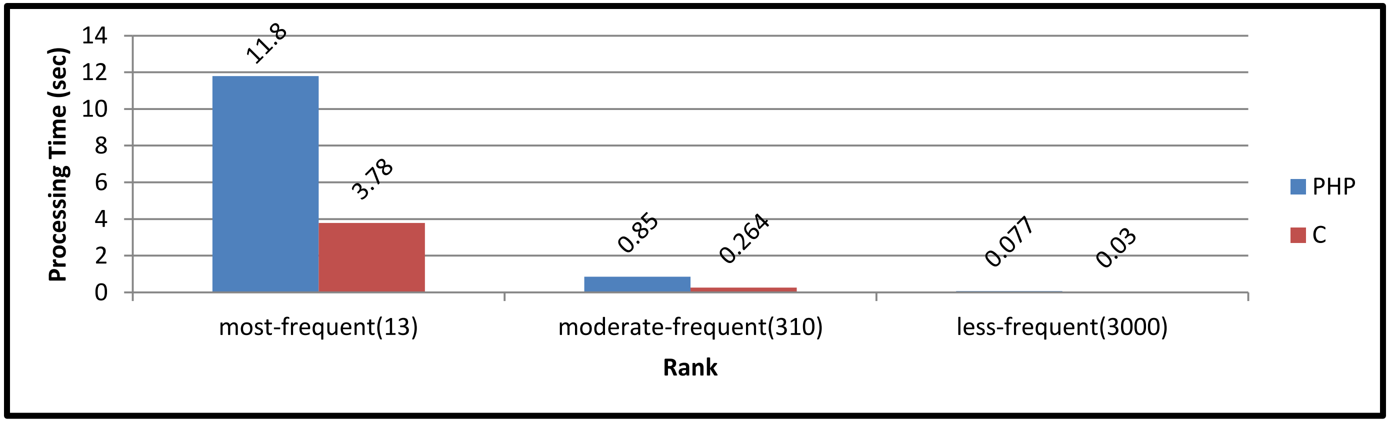


Encoding Test Results for100,000 documents (PHP7, 32 bit, i5)

Encoding Test Results …

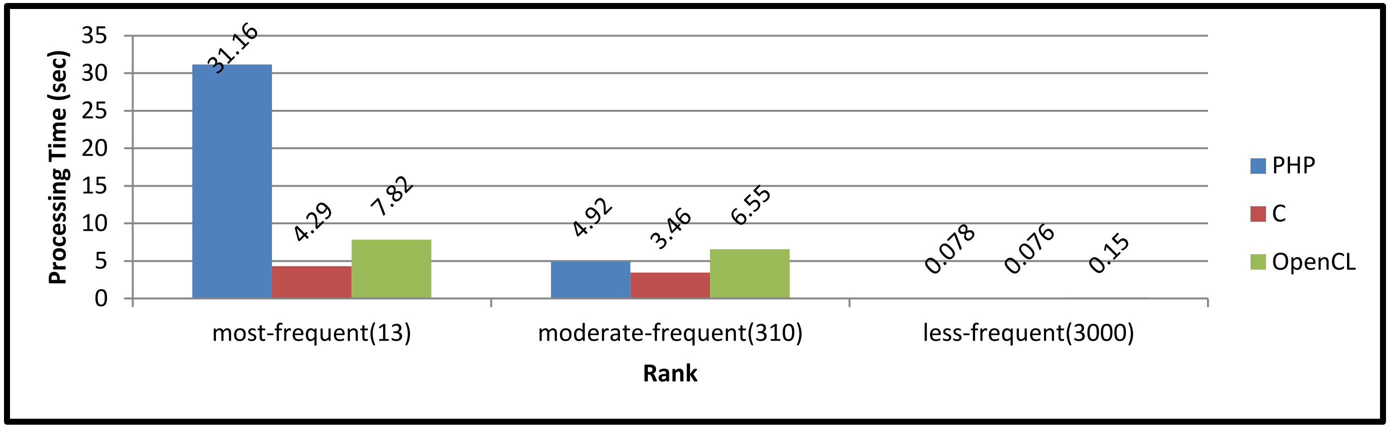


Encoding Test Results for 10,000 documents (PHP7, 32 bit, i7)

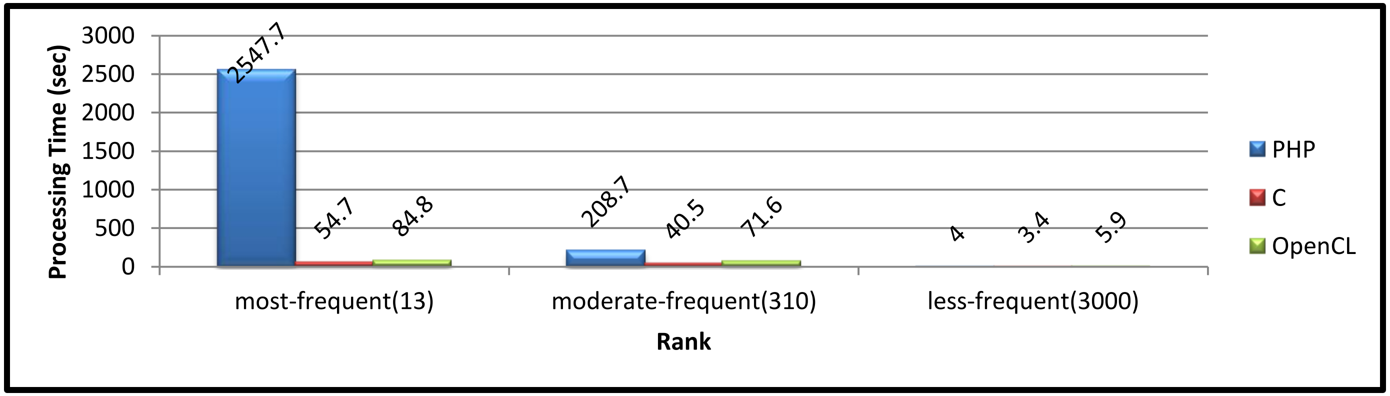


Encoding Test Results for100,000 documents (PHP7, 32 bit, i7)

Decoding Test Results

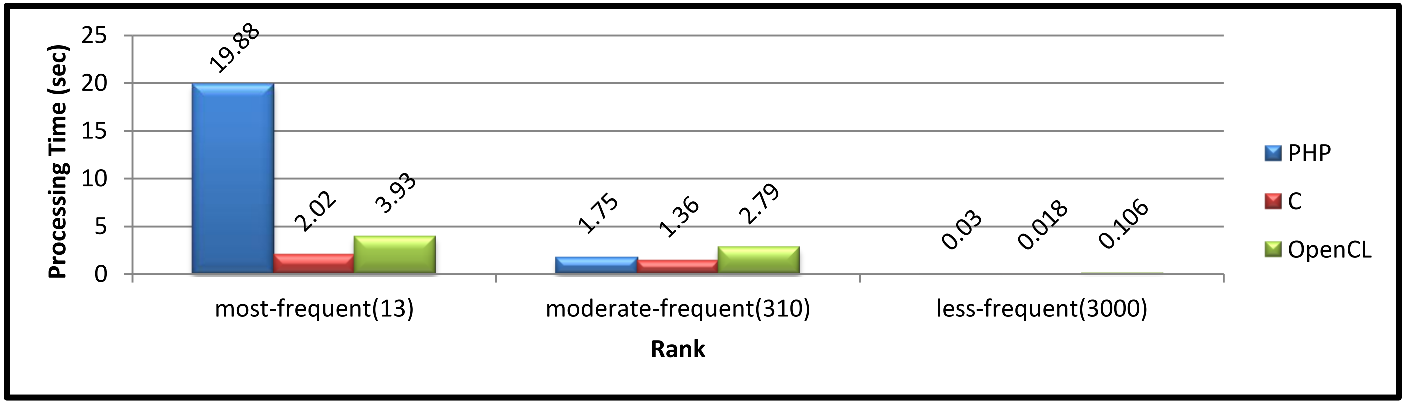


Decoding Test Results for 10,000 documents (PHP5, 32 bit, i5+HD Graphics)

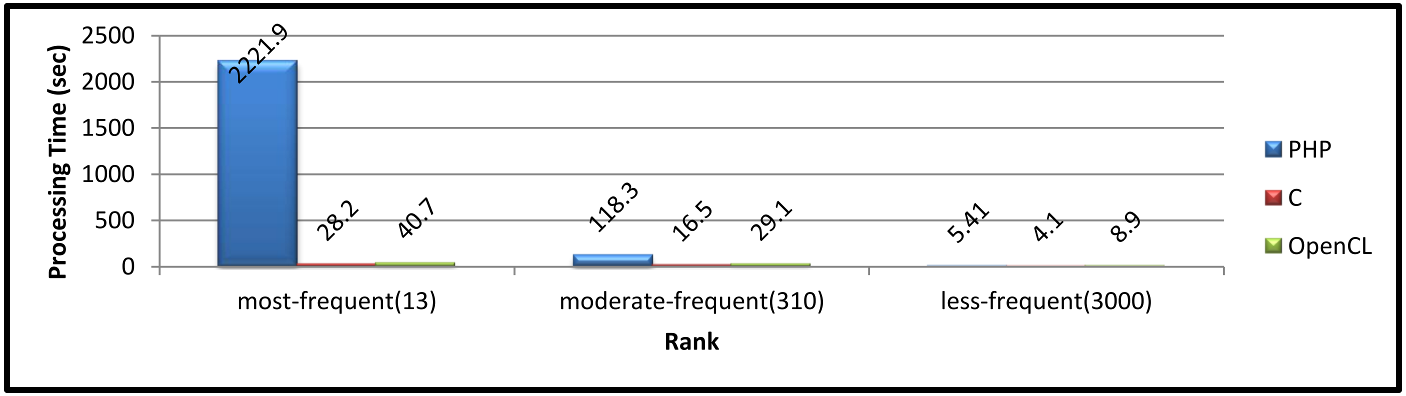


Decoding Test Results for100,000 documents (PHP5, 32 bit, i5+HD Graphics)

Decoding Test Results …

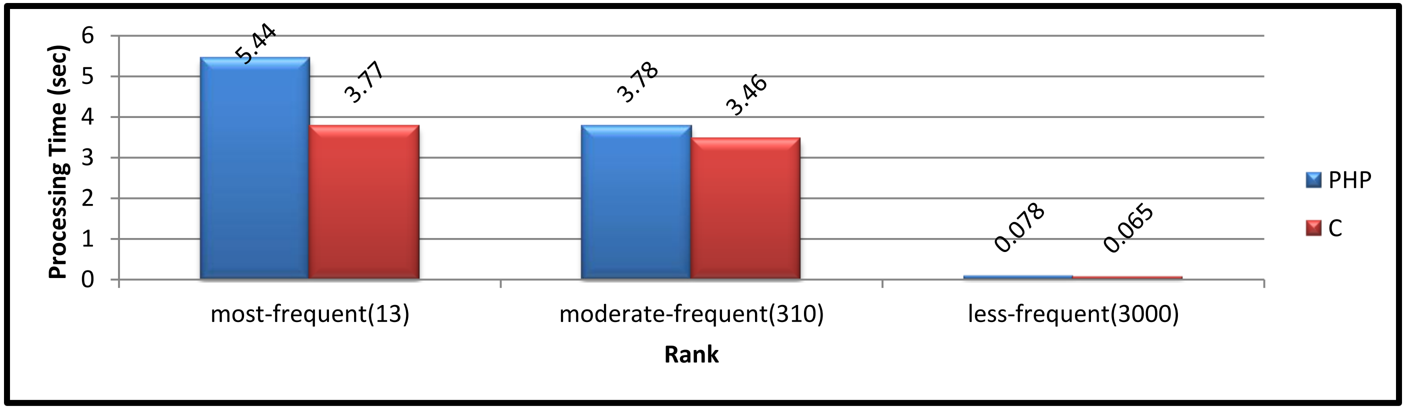


Decoding Test Results for 10,000 documents (PHP5, 32 bit, i7+Nvidia Graphics)

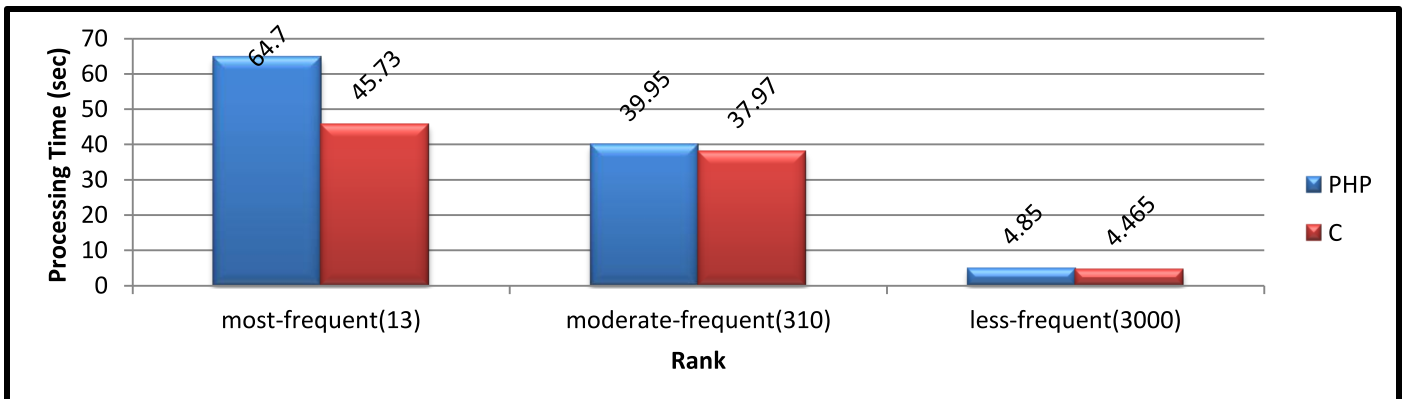


Decoding Test Results for100,000 documents (PHP5, 32 bit, i7+Nvidia Graphics)

Decoding Test Results …

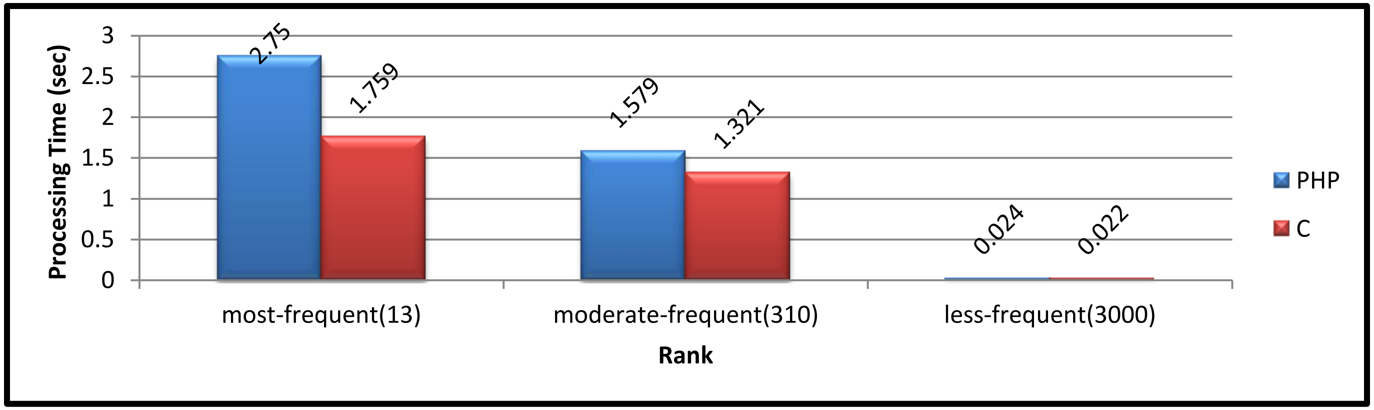


Decoding Test Results for 10,000 documents (PHP7, 32 bit, i5)

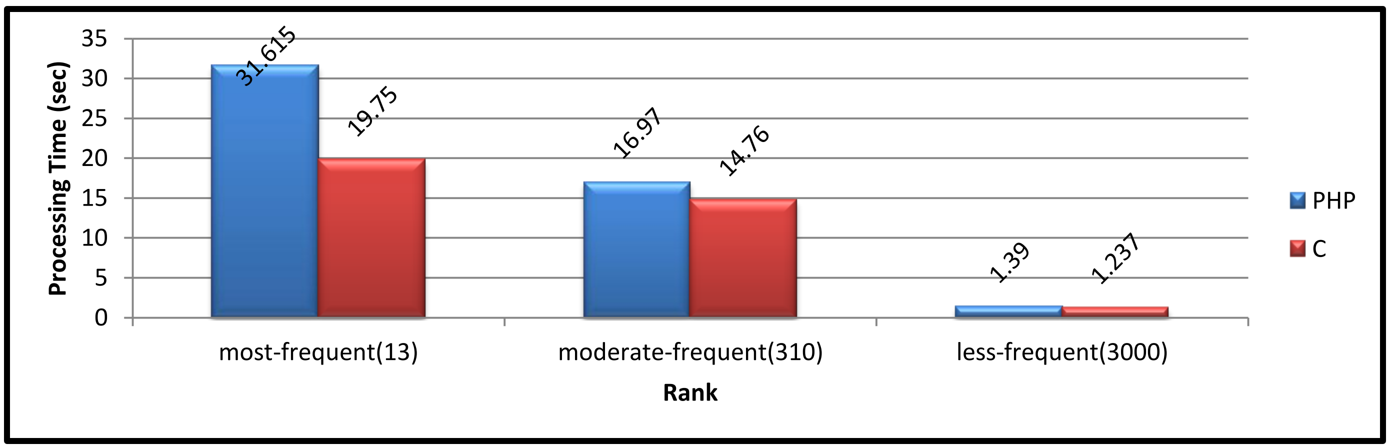


Decoding Test Results for100,000 documents (PHP7, 32 bit, i5)

Decoding Test Results …

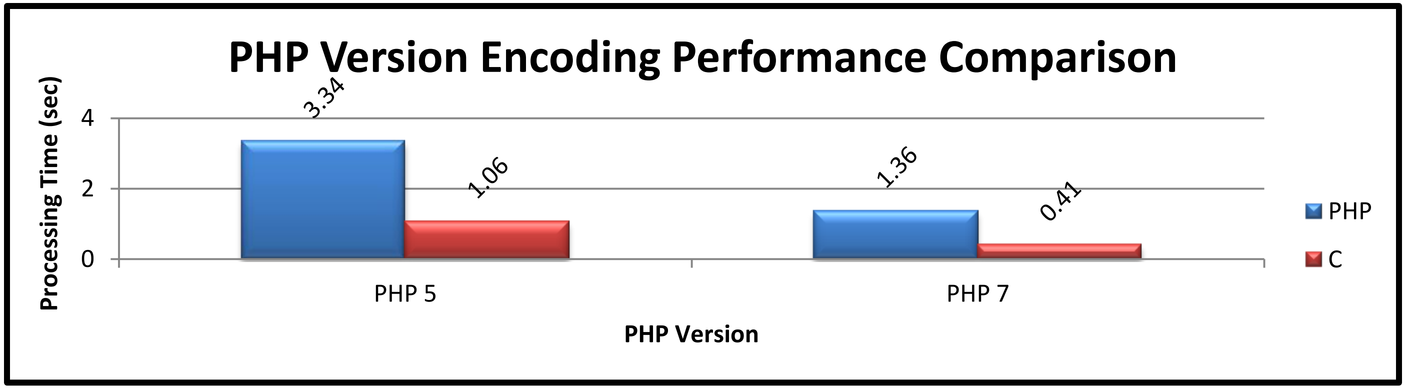


Decoding Test Results for 10,000 documents (PHP7, 32 bit, i7)

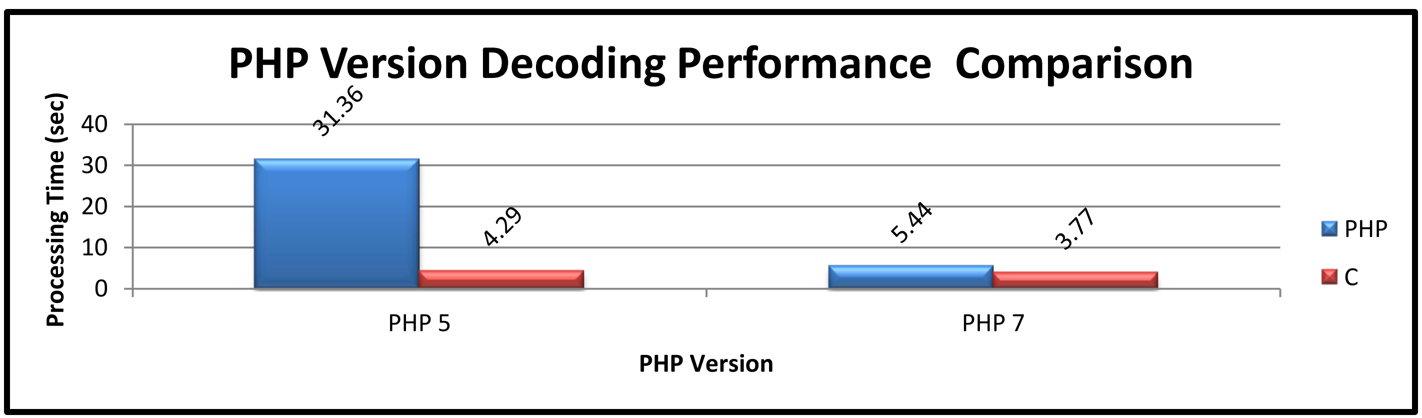


Decoding Test Results for100,000 documents (PHP7, 32 bit, i7)

PHP version comparison

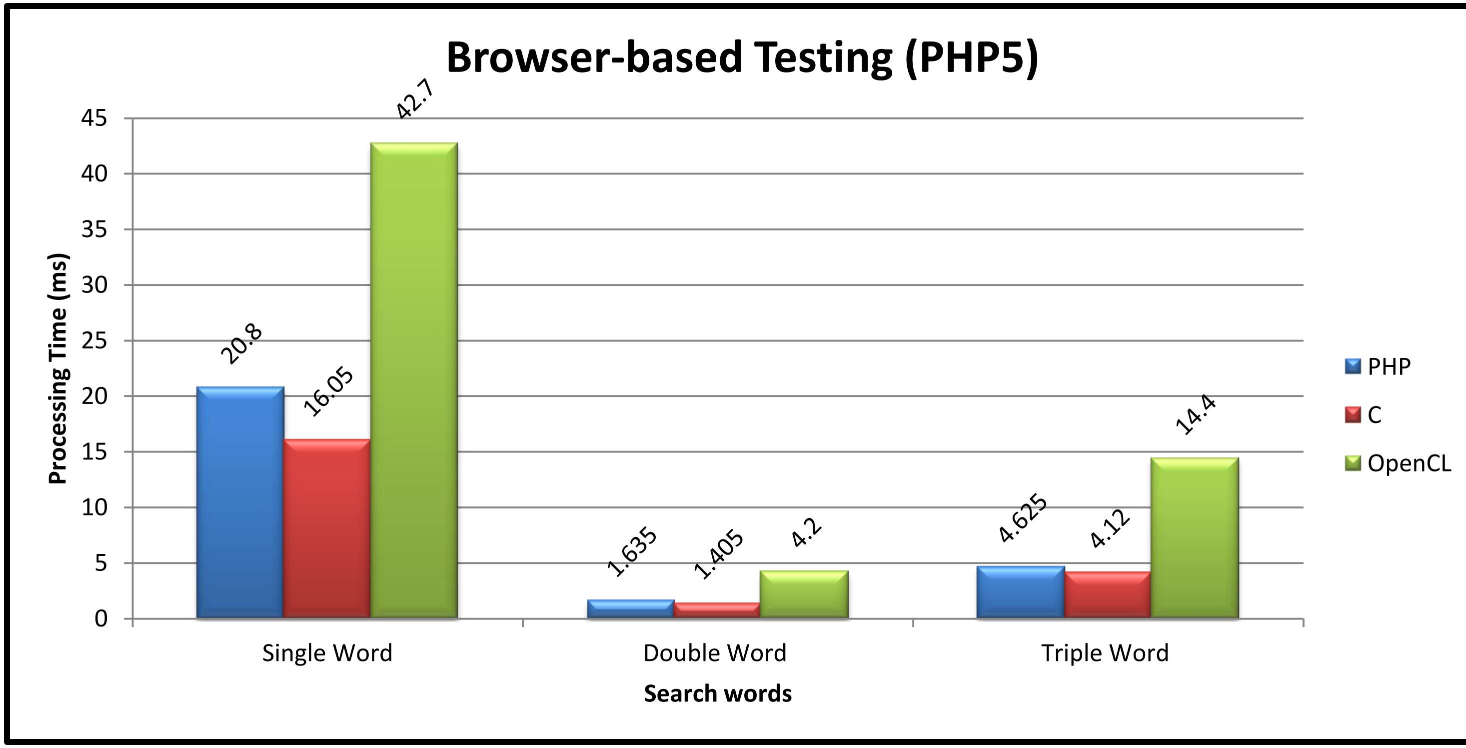


PHP5 Vs PHP7 Encoding Performance Comparison



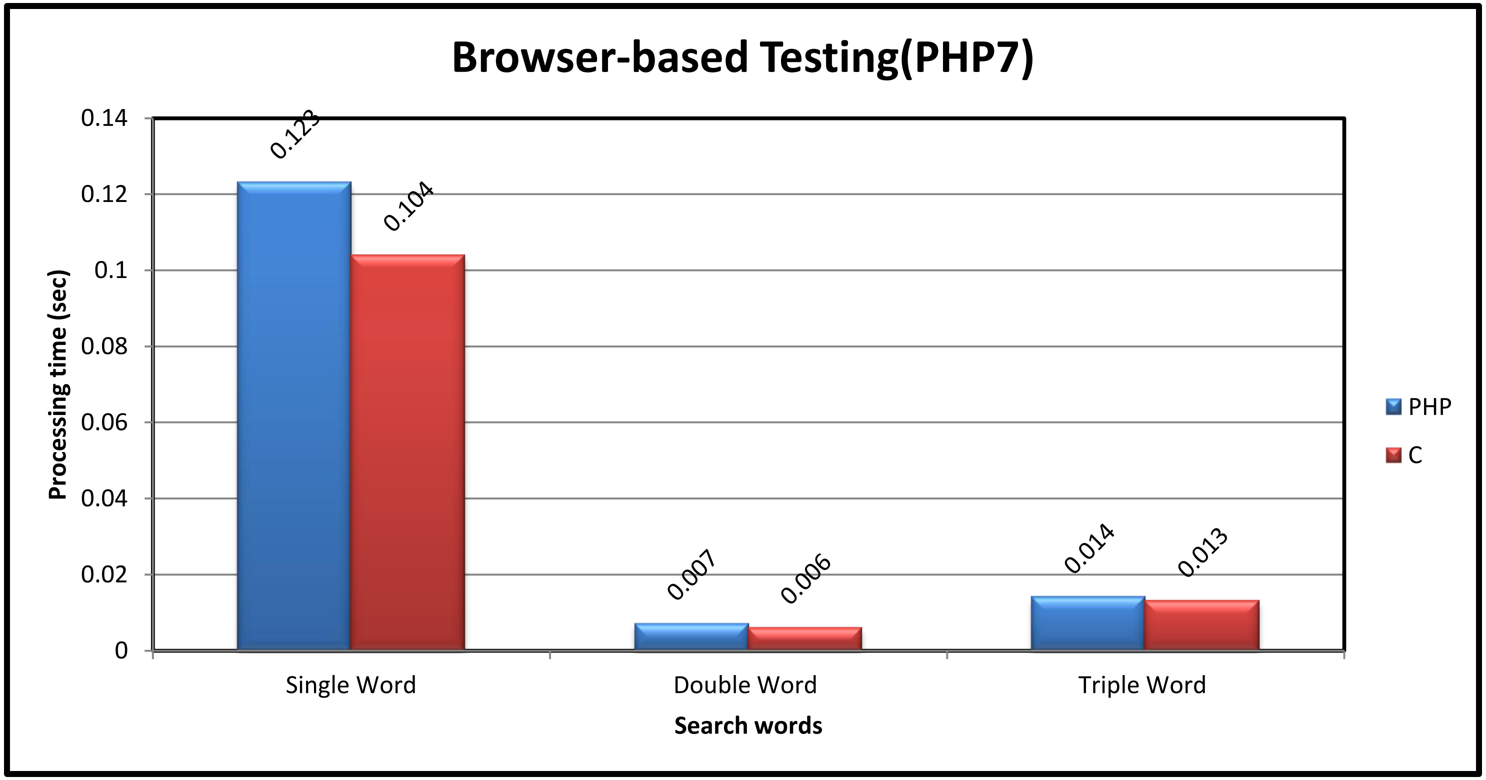
PHP5 Vs PHP7 Decoding Performance Comparison

Browser-based Test Results



Query performance from browser

Browser-based Test Results…



Query performance from browser

Conclusion

- C extensions performed 3 times better when compared to the original PHP code for the encoding test case. However, OpenCL has shown only a 0.4 times improvement for the most ranked term on an Nvidia based GPU
- C extensions performed 5 times better when compared to the original PHP code for the decoding test case. OpenCL has achieved a 4 times improvement for the most ranked term on both GPUs.
- Original PHP code performed about 2.5 times better for encoding case and about 6 times for the decoding test by simply switching to PHP7.
- Another 3 times improvement was observed using C Extensions along with PHP7 for the encoding case and about 0.4 times with the decoding test case.

Conclusion …

- No performance difference observed with 32 bit versus 64 bit software.
- Running tests on an *i7* machine, initial performance numbers were found to be very low, as Windows' defender service was consuming lot of system resources. Stopping the defender service improved the results.
- When the tests were run on a Windows 10-based system, memory compression service was taking up more system resources than the application itself and skewing the results.
- Browser based tests also had shown performance gains when C Extensions were used.

Conclusion …

- Overall, Yioop's original code is already well optimized and achieving further improvements is not a trivial task. However, these experiments proved that the Yioop search engine’s performance can be improved by using a combination of OpenCL and C extensions for most resource and compute intensive functions. The operating system and the right type of GPU and CPU combination will help achieve optimum performance results.

References

- [1] The open standard for parallel programming of heterogeneous systems. (2016). Retrieved on January 15, 2016, from https://www.khronos.org/opencl/
- [2] Extension Writing Part I: Introduction to PHP and Zend. (2015). Retrieved on April 16, 2015, from http://devzone.zend.com/303/extension-writing-part-i-introduction-to-php-and-zend/#Heading2
- [3] Woolley, C. (2010). Introduction to OpenCL. Retrieved on November 5, 2015, from http://www.cc.gatech.edu/~vetter/keeneland/tutorial-2011-04-14/06-intro_to_opencl.pdf
- [4] Stefan, B., Clarke, C., Cormack, G. (2010). Information retrieval-Implementing and Evaluating Search Engines. Cambridge, Massachusetts: MIT Press.
- [5] Benedict, G., Howes, L., Kaeli, D., Mistry, P., Schaa, D. (2011). Heterogeneous Computing with OpenCL. Morgan Kaufmann.
- [6] Golemon, Sara. Extending and Embedding PHP. Indianapolis, Ind.: Sams, 2006. Print.
- [7] Scholer, F., Williams, H. E., Yiannis, J., and Zobel, J. (2002). *Compression of inverted indexes for fast query evaluation. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 222-229. Tampere, Finland.
- [8] Che, S., Jiayuan, M., W. Sheaffer, J., Skadron, K., (). A Performance Study of General Purpose Applications on Graphics Processors. Retrieved on September 2, 2015, from https://pdfs.semanticscholar.org/03aa/649535c7e01ac2b3255f2f44131380dc93c7.pdf
- [9] Keane, A. (2016). "GPUS ARE ONLY UP TO 14 TIMES FASTER THAN CPUS" SAYS INTEL. Retrieved on March 6, 2016, from https://blogs.nvidia.com/blog/2010/06/23/gpus-are-only-up-to-14-times-faster-than-cpus-says-intel/
- [10] Yioop website. Retrieved on September 2, 2015, from http://www.seekquarry.com/

Questions?

Thank you!!