Steganography – coding and intercepting the information from encoded pictures in the absence of any initial information

Steganografia – kodowanie oraz przechwytywanie informacji z zakodowanych obrazów przy braku jakichkolwiek informacji początkowych

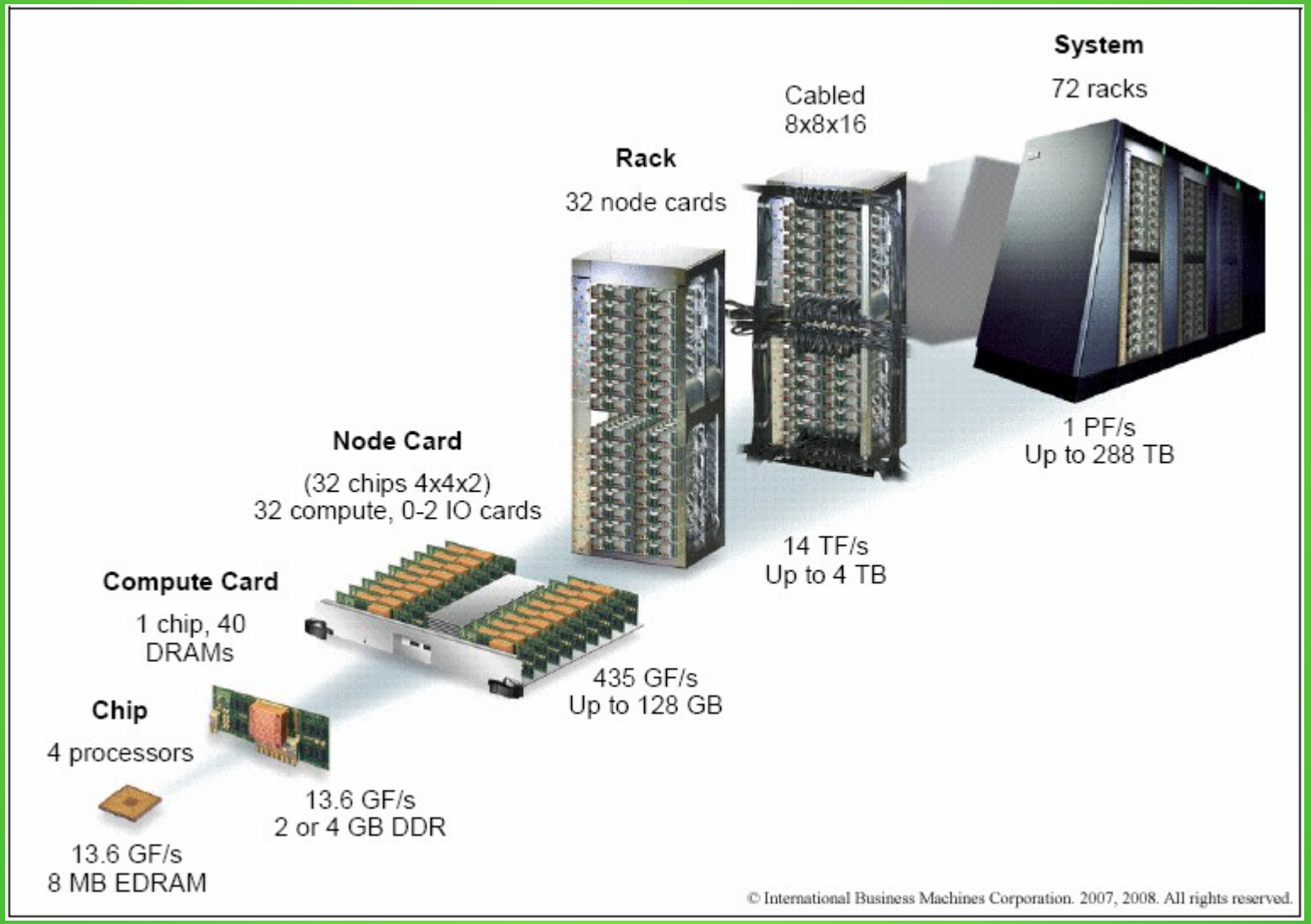
Monika Kwiatkowska, Łukasz Świerczewski
Lublin, Poland, Łomża, Poland

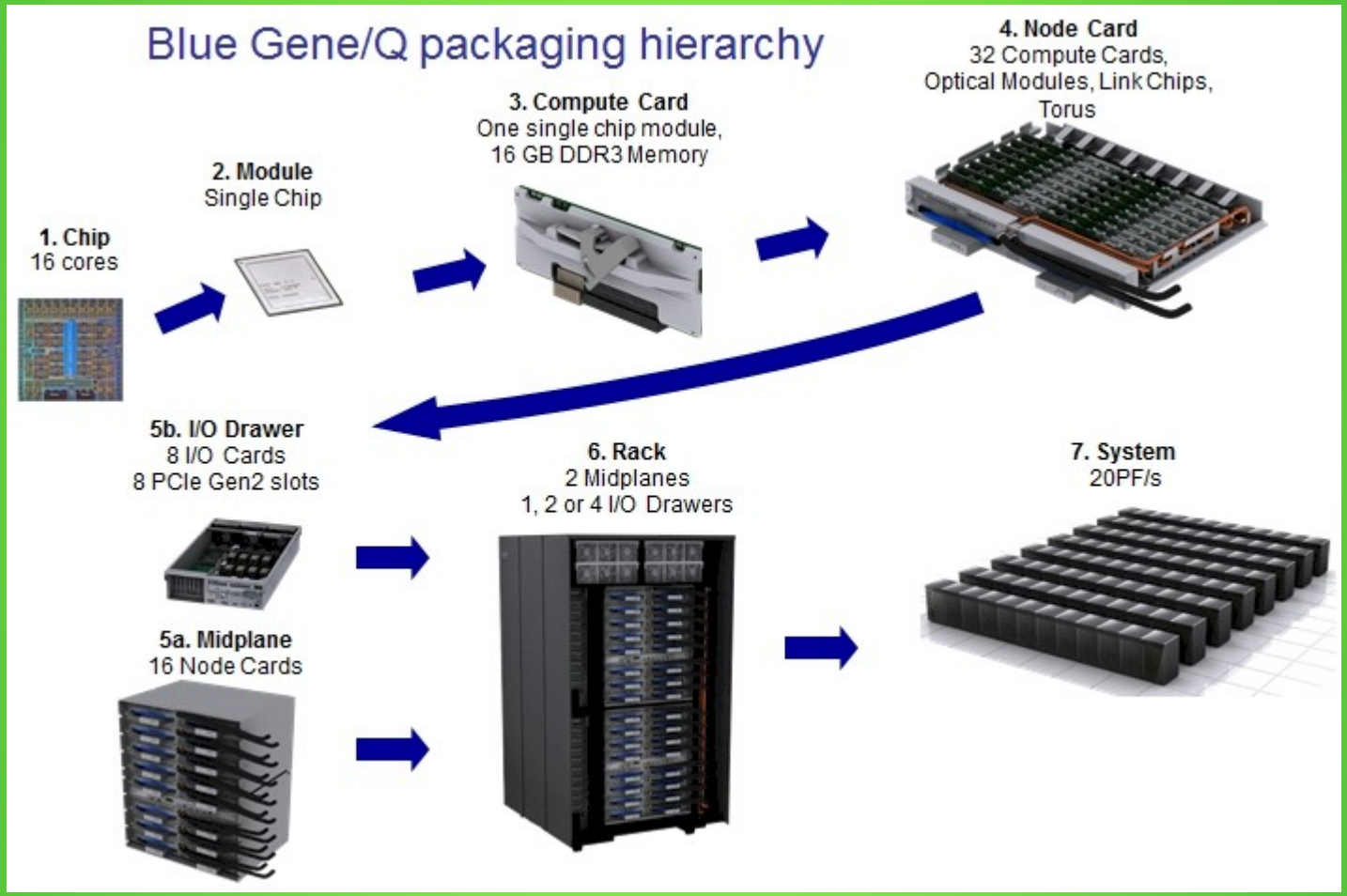
kwiatkowska2305@wp.pl
luk.swierczewski@gmail.com

15.02.2014

- Used supercomputers – BlueGene /P and /Q
- Introduction to steganography
  - **Traditional Steganography**
  - **Digital Steganography**
- Performance results

BlueGene/P


System
72 racks
Cabled
8x8x16
Rack
32 node cards
1 PF/s
Up to 288 TB
Node Card
(32 chips 4x4x2)
32 compute, 0-2 IO cards
14 TF/s
Up to 4 TB
Compute Card
1 chip, 40
DRAMs
435 GF/s
Up to 128 GB
Chip
4 processors
13.6 GF/s
2 or 4 GB DDR
13.6 GF/s
8 MB EDRAM
© International Business Machines Corporation. 2007, 2008. All rights reserved.


Blue Gene/Q packaging hierarchy
1. Chip
16 cores
2. Module
Single Chip
3. Compute Card
One single chip module,
16 GB DDR3 Memory
4. Node Card
32 Compute Cards,
Optical Modules, Link Chips,
Torus
5b. I/O Drawer
8 I/O Cards
8 PCIe Gen2 slots
5a. Midplane
16 Node Cards
6. Rack
2 Midplanes
1, 2 or 4 I/O Drawers
7. System
20PF/s


IBM
Blue Gene/P

Traditional Steganography

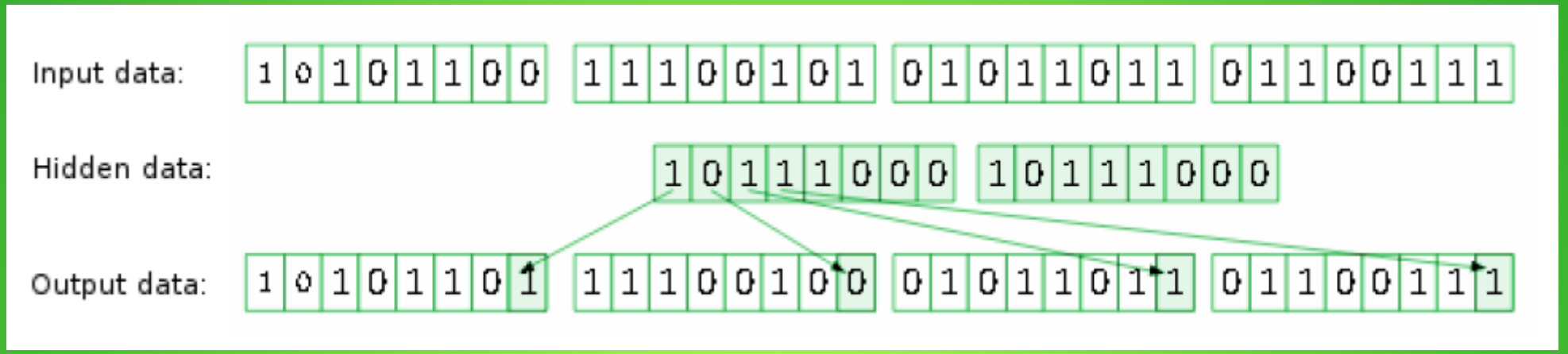
The use of steganography dates back to the time of Herodotus, the fifth century BC. Examples of traditional steganography can be tattooing the scalp (after the hair grew back information remains invisible). One of the best solutions of this kind applied by the Germans during World War II - microdots technique. It was based on minimazing pictures to such scale so that you can paste them into the text as a dot.

Image of a tree with a steganographically hidden image. The hidden image is revealed by removing all but the two least significant bits of each color component and a subsequent normalization. The hidden image is shown below.

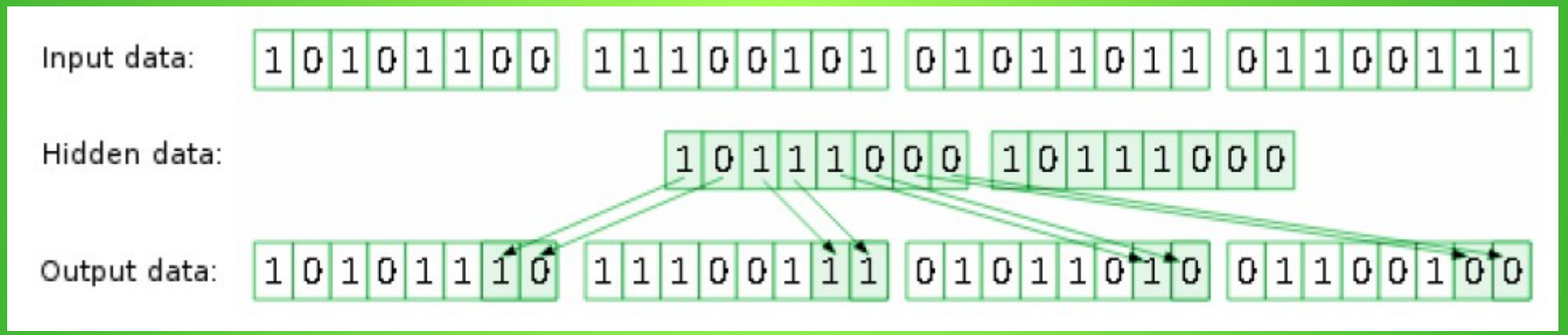
Logo of the conference LVEE with hidden text „LVEE – the best conference”.

	Cryptography	Steganography
Transforming information into a form incomprehensible to third parties	Yes	Yes
Hiding information	No	Yes
Key usage	Yes	Yes
Hiding the fact of communication	No	Yes
Ensuring anonymity of communicating parties	No	Yes
The amount of information transmitted in the communication process	Comparable to the amount of encrypted information	Much greater than the amount of encrypted information
Additional carrier needed	No	Yes

Differences between steganography and cryptography.



Applying the LSB using one least significant bit.



Application of the LSB method using the two least significant bits.

```
int crypt_space_1bit(char *buffer, char *open_text, long int start_position, long int
open_text_size, long int space_size)
{
  long int i;
  for(i=0; i < (open_text_size * 8); i++)
  {
    if( check_bit(open_text[i/8], i%8) )
    {
      set_bit(buffer[start_position+i*space_size], 0);
    }
    else
    {
      clear_bit(buffer[start_position+i*space_size], 0);
    }
  }
}
```

The function encrypting the data in the image using one least significant bit.

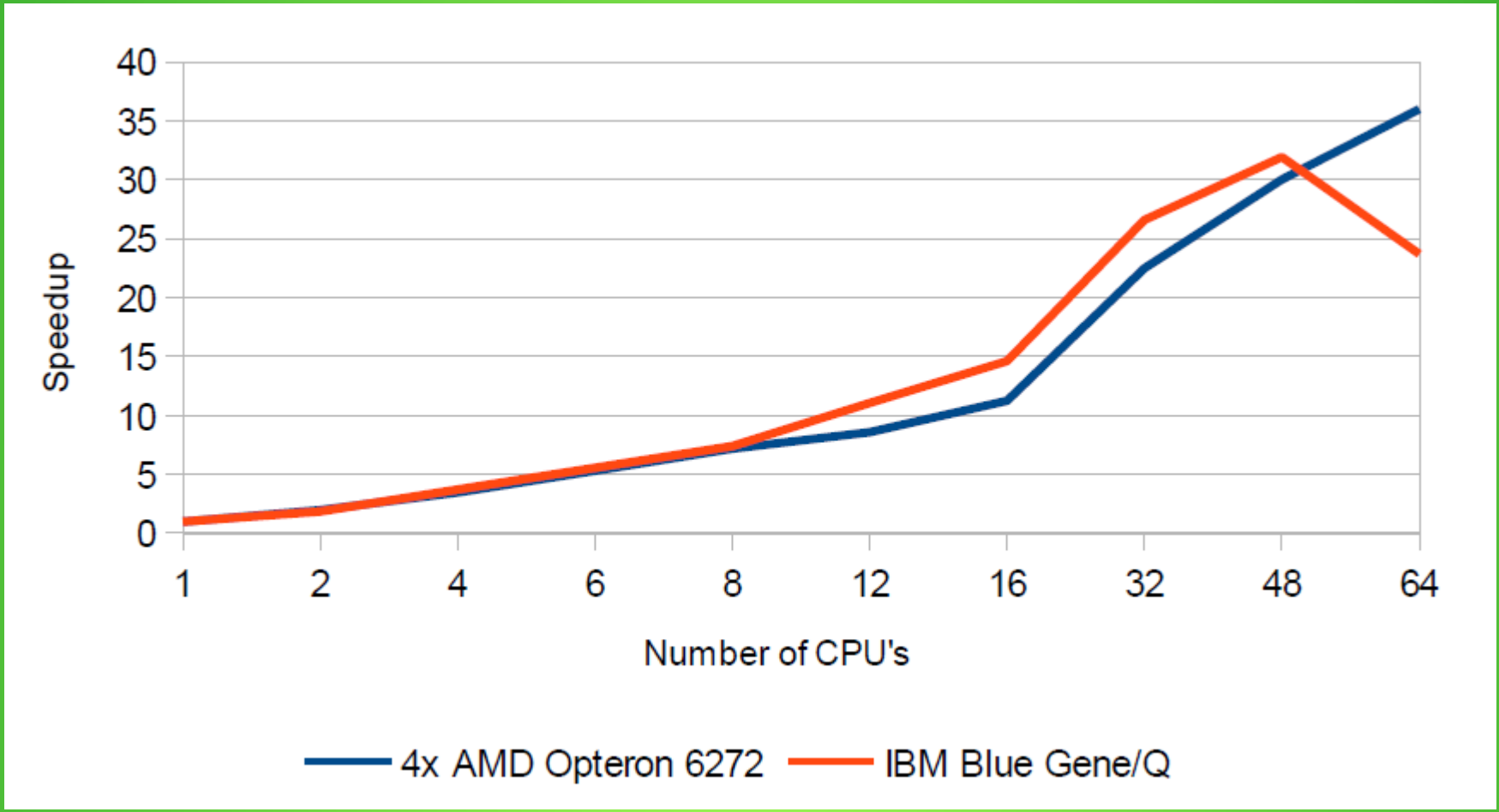
```
long int breaker_standard_1bit_unknown_size_openmp(char *buffer, long int buffer_size,
long int size_down, long int size_upper, result_structure *result, int number_of_threads)
{
  char *open_text;

  long int start_position;

  long int index_result;
  index_result = 0;

  long int i;
  long int j;
  j = size_down;

  #pragma omp parallel for shared(buffer, buffer_size, size_down, size_upper, result,
index_result) private(i, j, start_position, open_text) num_threads(number_of_threads)
  for(i=0; i < (buffer_size - j); i++)
  {
    open_text = (char*) malloc( (size_upper+1) * sizeof(char));

    start_position = i;

    for(j=size_down; j <= size_upper; j++)
    {
      encrypt_standard_1bit(buffer, open_text, start_position, j);
```

```
      if ( ascii_veryfi(open_text, j) == 1 )
      {
        open_text[j] = '\0';

        #pragma omp critical
        {
          result[index_result].start_position = i;
          strcpy(result[index_result].text, open_text);
          index_result++;
        }
      }
    }

    free(open_text);
  }

  return index_result;
}
```

Function to extract information from the least significant bits without the initial information.

	Execution time		
Number of threads	IBM Blue Gene/P	IBM Blue Gene/Q	AMD Opteron 6272
1	560.47 s	193.11 s	180.00 s
2	282.78 s	97.71 s	92.00 s
4	149.23 s	48.38 s	52.00 s
6	-	32.28 s	34.00 s
8	-	24.27 s	25.00 s
12	-	16.29 s	21.00 s
16	-	12.32 s	16.00 s
32	-	6.77 s	8.00 s
48	-	5.64 s	6.00 s
64	-	7.60 s	5.00 s

Performance results obtained during scan of the image with a resolution of 1600x1200 (searching one least significant bit using length of the search phrase in the range of 10 to 25).



The speedup obtained on platforms AMD Opteron 6272 and IBM Blue Gene/Q while searching an image with a resolution of 1600x1200 (search only one least significant bit of the length of the text to search in the range from 10 to 25).

	Execution time		
Number of threads	IBM Blue Gene/P	IBM Blue Gene/Q	AMD Opteron 6272
1	4769.00 s	1309.42 s	1282.08 s
2	2482.00 s	651.12 s	652.00 s
4	1311.00 s	308.89 s	401.00 s
6	-	201.24 s	265.00 s
8	-	147.00 s	200.00 s
12	-	97.25 s	132.00 s
16	-	74.19 s	107.00 s
32	-	36.90 s	59.00 s
48	-	25.76 s	43.00 s
64	-	22.49 s	35.00 s

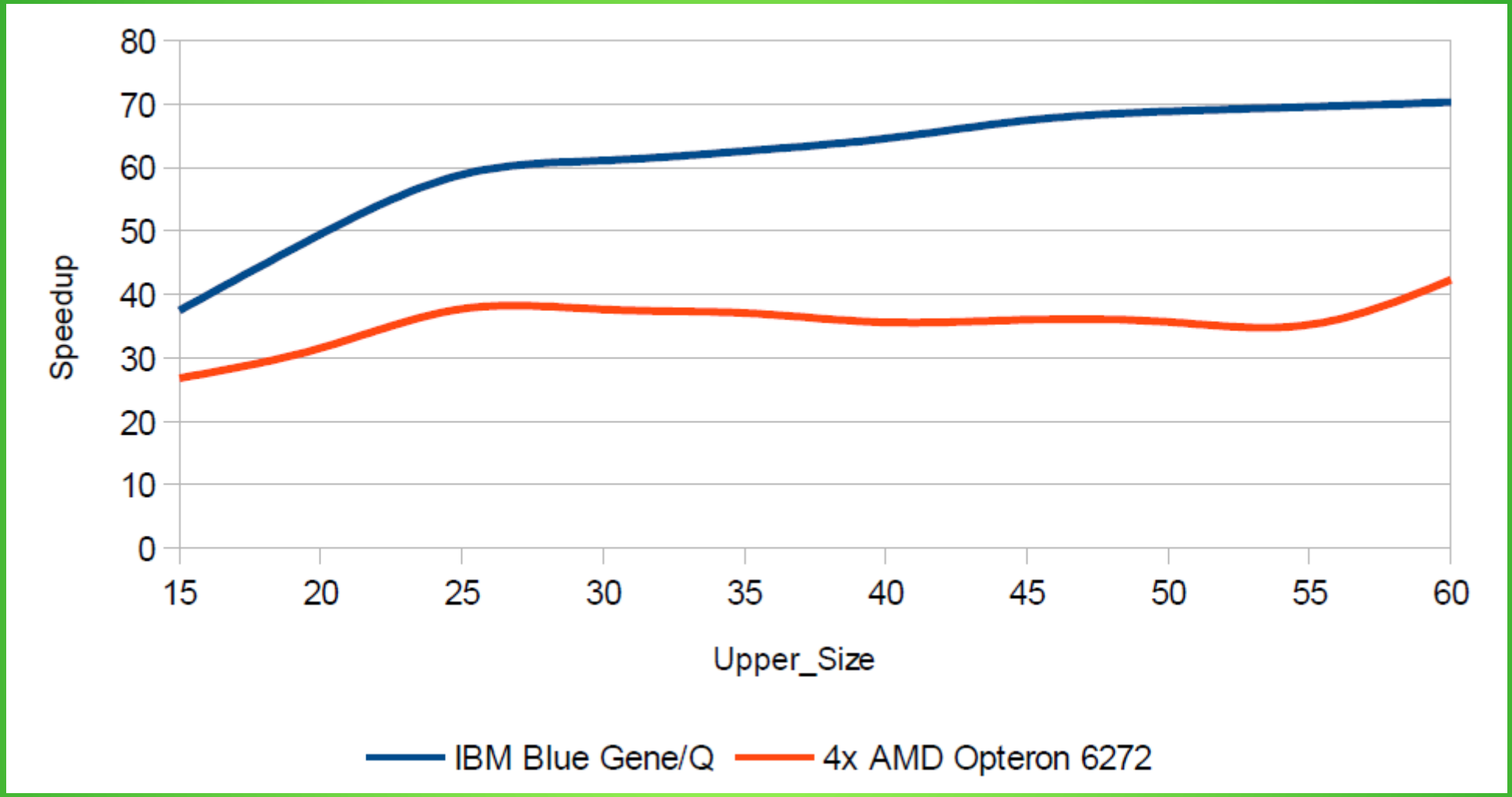
Performance results obtained during scan of the image with a resolution of 4096x4096 (searching one least significant bit using length of the search phrase in the range of 10 to 25).

Number of CPU's	*Down_Size = 10; Upper_Size =*									
	15	20	25	30	35	40	45	50	55	60
1	348.47	757.28	1282.40	1919.38	2668.21	3524.63	4501.62	5631.80	6799.42	8122.23
64	13.00	24.00	34.00	51.00	72.00	99.00	125.00	158.00	193.00	192.00
Speedup	26.805	31.553	37.718	37.635	37.064	35.602	36.013	35.664	35.230	42.303

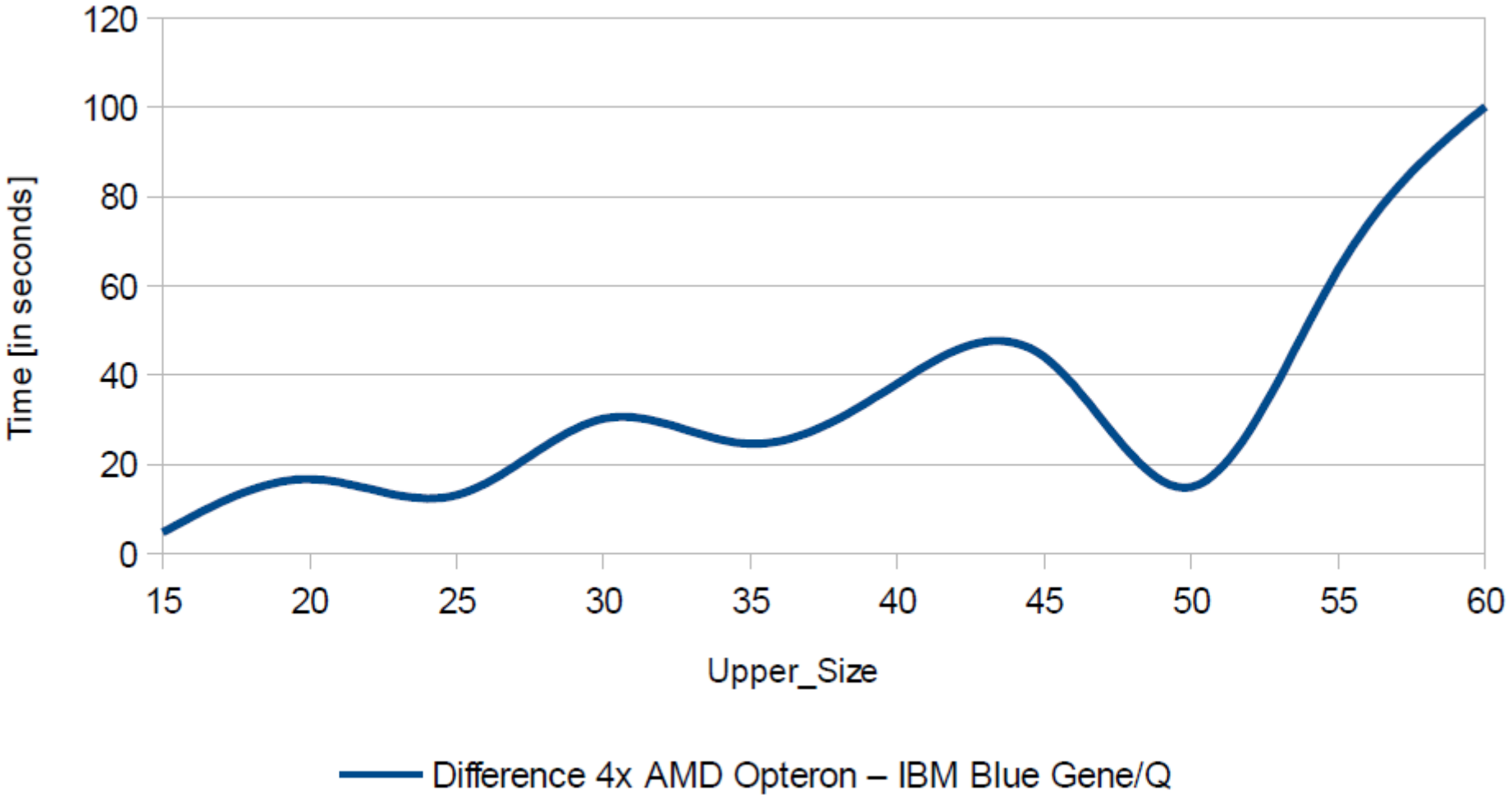
The results of the analysis of performance for different sizes of searched hidden string [Down_Size; Upper_Size] (the ranges of [10, 15] to [10,60]) and platform based on AMD Opteron 6272.

Number of CPU's	*Down_Size = 10; Upper_Size =*									
	15	20	25	30	35	40	45	50	55	60
1	353.18	773.93	1295.46	1949.57	2692.76	3562.74	4545.64	5646.70	6863.10	8222.39
64	9.43	15.64	22.00	31.90	43.01	55.16	67.39	81.99	98.64	116.93
Speedup	37.452	49.484	58.885	61.115	62.608	**64.589**	**67.452**	**68.871**	**69.577**	**70.319**

Table 5. The results of the analysis of performance for different sizes of searched hidden string [Down_Size; Upper_Size] (the ranges of [10, 15] to [10,60]) and platform based on processors installed in the Blue Gene/Q.



Graph showing the resulting speedup on AMD Opteron 6272 and IBM Blue Gene/Q in the analysis of performance for different sizes of searched hidden string.



Graph showing the time difference obtained on a platform consisting of the AMD Opteron 6272 and obtained on the Blue Gene/Q in the analysis of performance for different sizes of searched hidden string.

Acknowledgment

Interdisciplinary Centre for Mathematical and Computational Modeling (ICM), Warsaw University, Poland is acknowledged for providing the computer facilities under the Grant No. G55-11.

Thank you for your attention!

kwiatkowska2305@wp.pl

luk.swierczewski@gmail.com

www.lib.oproject.info