CATEGORY: **DEVELOPER - ALGORITHMS - DA01**

POSTER
P5103

CONTACT NAME
Akiyoshi Wakatani: wakatani@konan-u.ac.jp

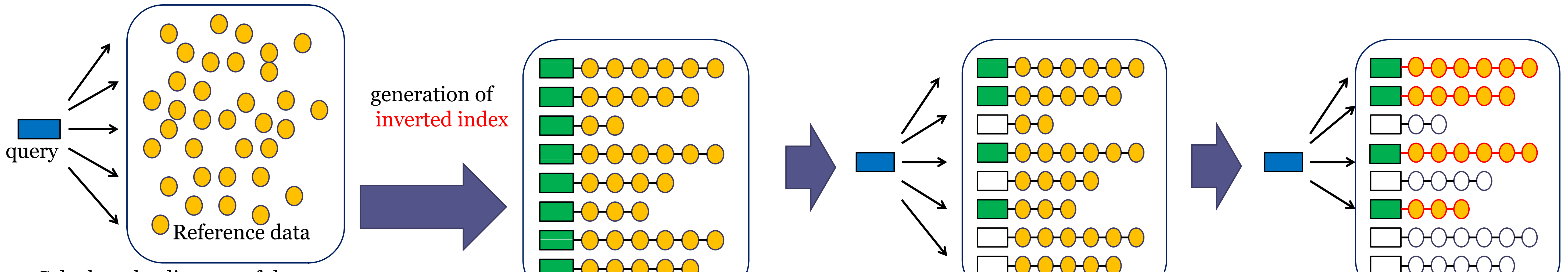
GPU TECHNOLOGY CONFERENCE

Implementation of Nearest Neighbor Search on GPGPU systems

Akiyoshi Wakatani (Konan University, JAPAN)

Summary: A nearest neighbor search with product quantization is a prominent method that achieves a high-precision search with less memory consumption than an exhaustive way. In order to accomplish a large size search with a large reference data, the search method have to be accelerated by using parallel systems such as multicore processors and GPGPU (General Purpose computing on GPU) systems. The distance calculation between a query and a reference data is an independent operation that is easily parallelized, but the reduction computation of distances after that is not completely parallel, so this leads to performance degradation. Therefore, in order to maximize a speedup, the adequate parameter selection is required in terms of parallelism. In this paper, the baseline of parallelization of the nearest neighbor search with product quantization is described, and the validity of our approach (Optimistic Search), which utilizes small number of candidates of nearest neighbors, is discussed with experiments. We also show the effectiveness of pseudo matrix transposition for the sake of the efficient search. In addition, the method for autotuning is proposed and its effectiveness is empirically confirmed.

Nearest neighbor search with product quantization and its parallelization



Calculate the distance of the query and reference data and find the nearest neighbor

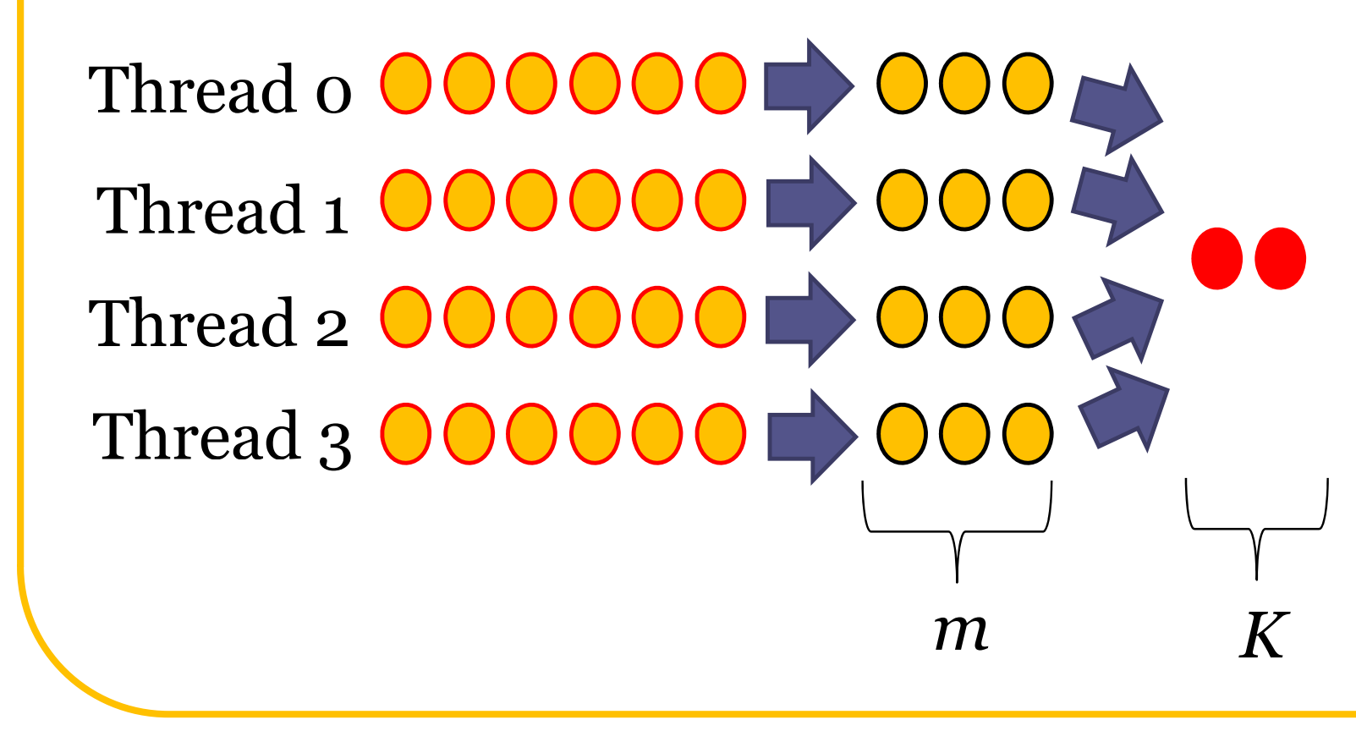
but, very large computation

Each data is quantized by product quantization method.

Select the nearest indices for the search by the coarse quantization.

Select the nearest neighbors linked to the selected indices.

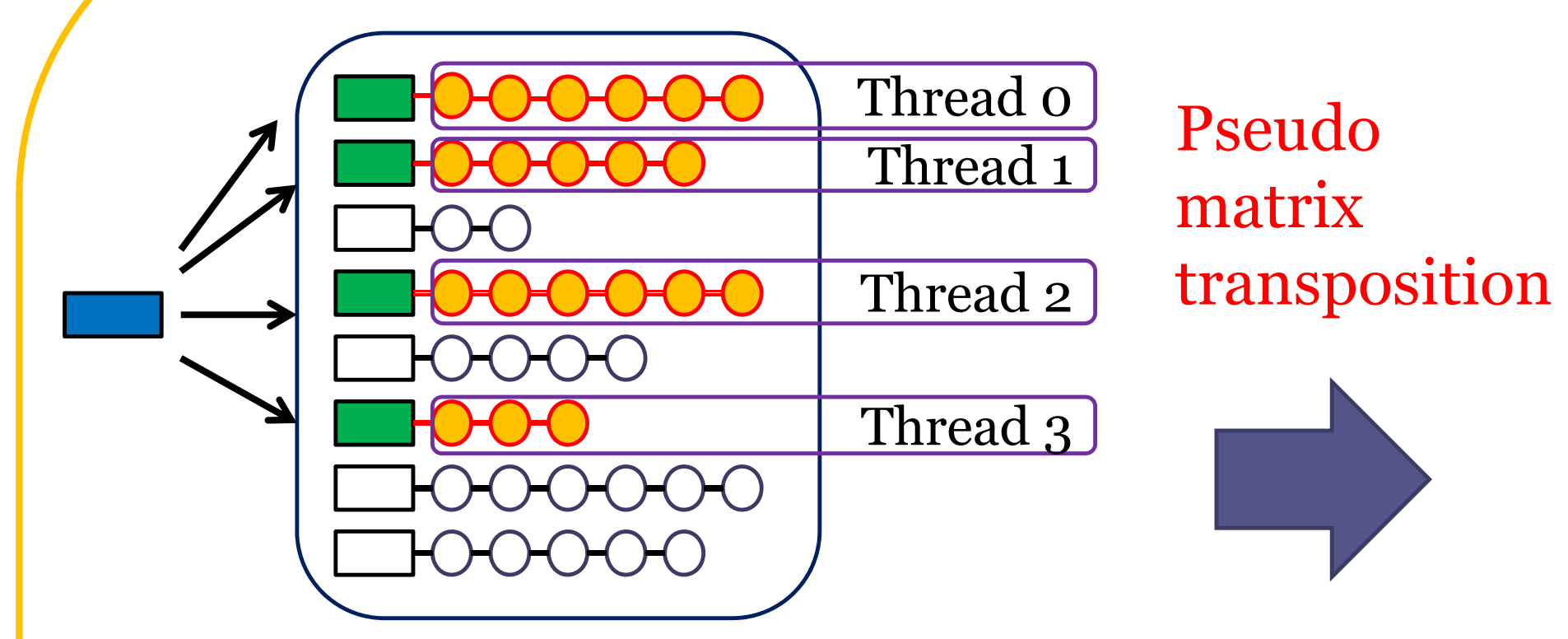
Parallelization

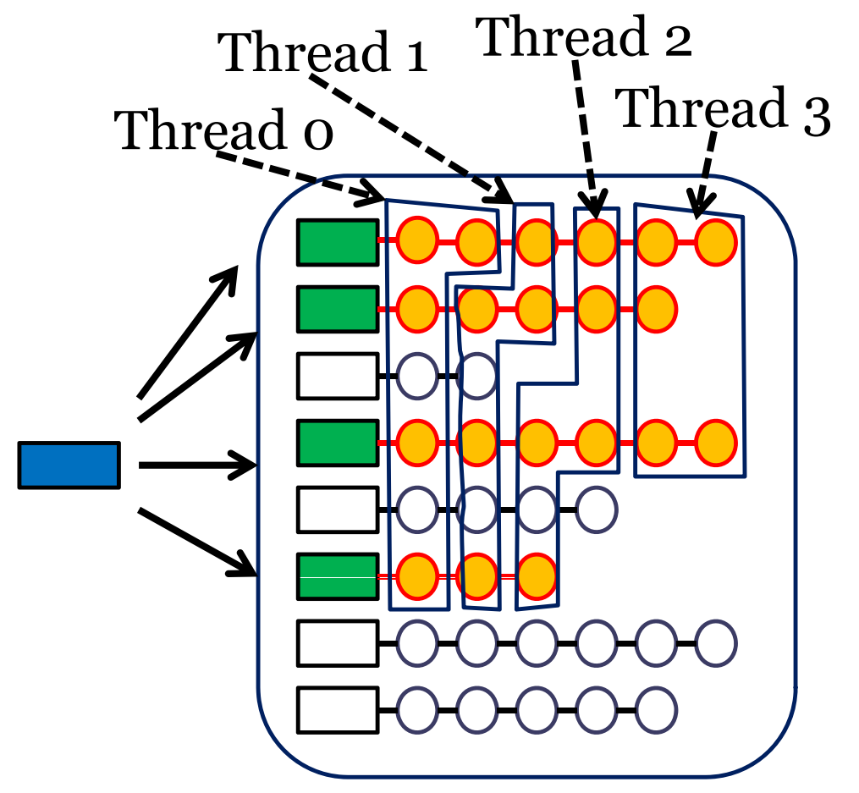


On GPU, in order to find *K* nearest neighbors, each thread (totally, *P* threads) find *m* nearest neighbor candidates, and then *K* nearest neighbor are found among *P*m* candidates. So, m must be as small as possible to enhance the parallelism ($K/P<m<K$). Note that each thread block of a GPU is in charge of each query, and all the threads in a block collectively determine the nearest neighbors of the query.

Results and discussion

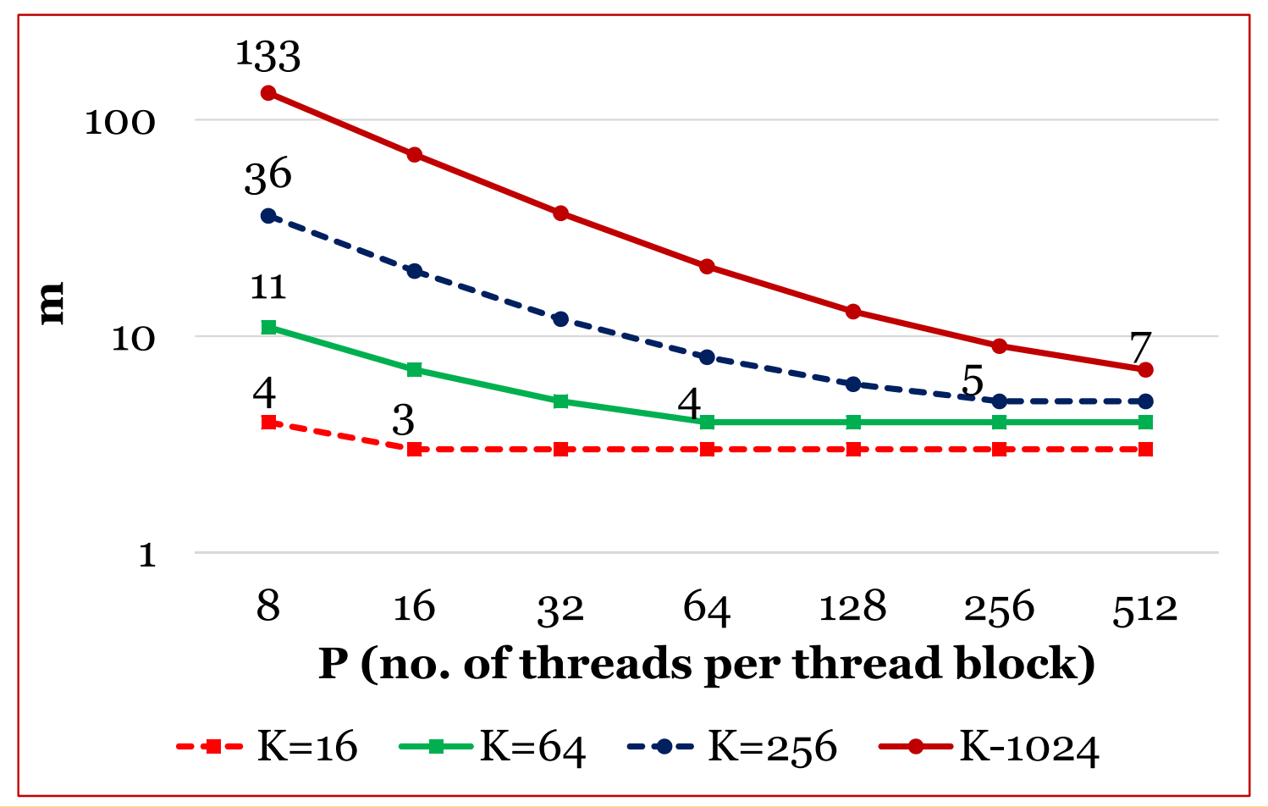
Optimal number of nearest neighbor candidates (m) and pseudo matrix transposition



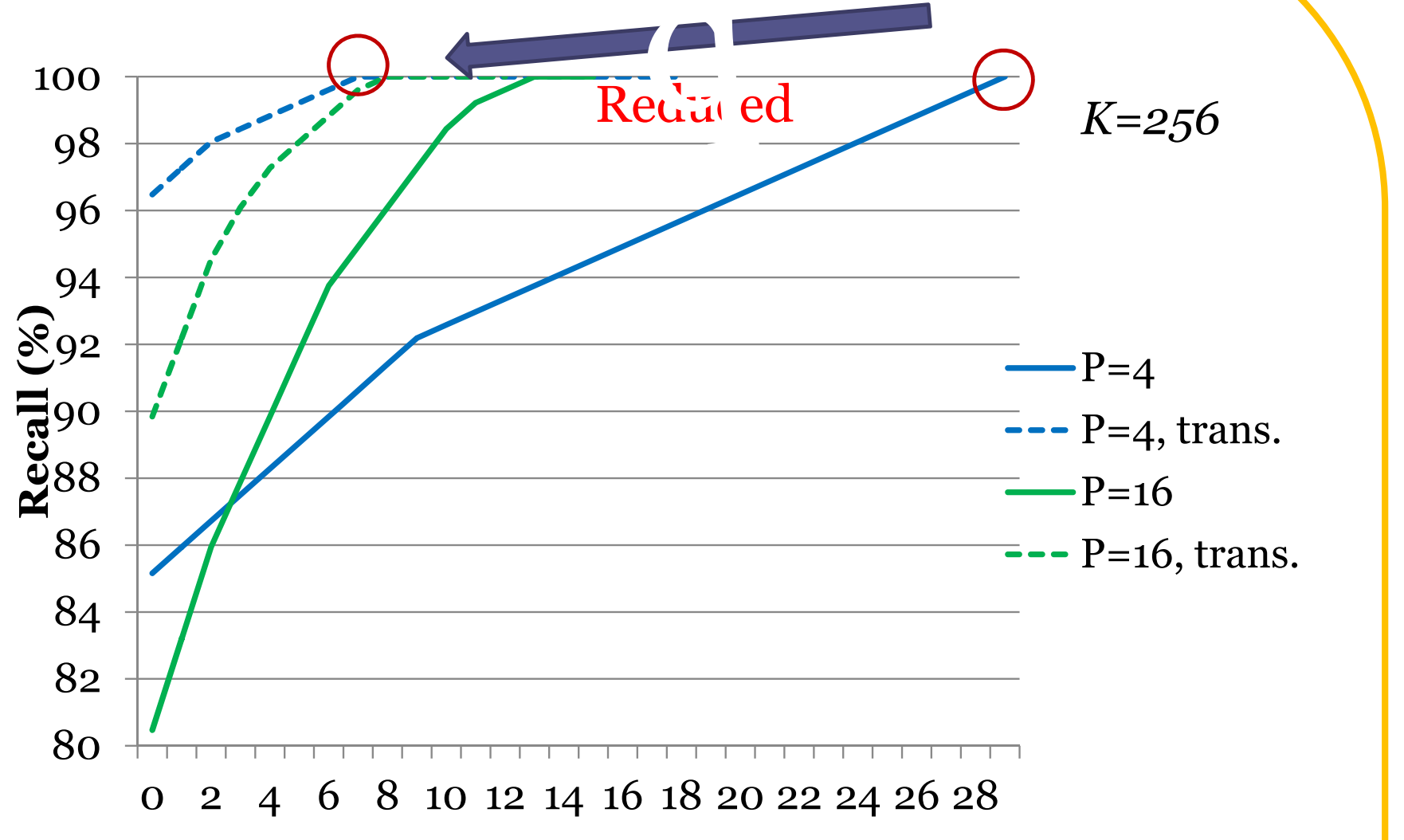


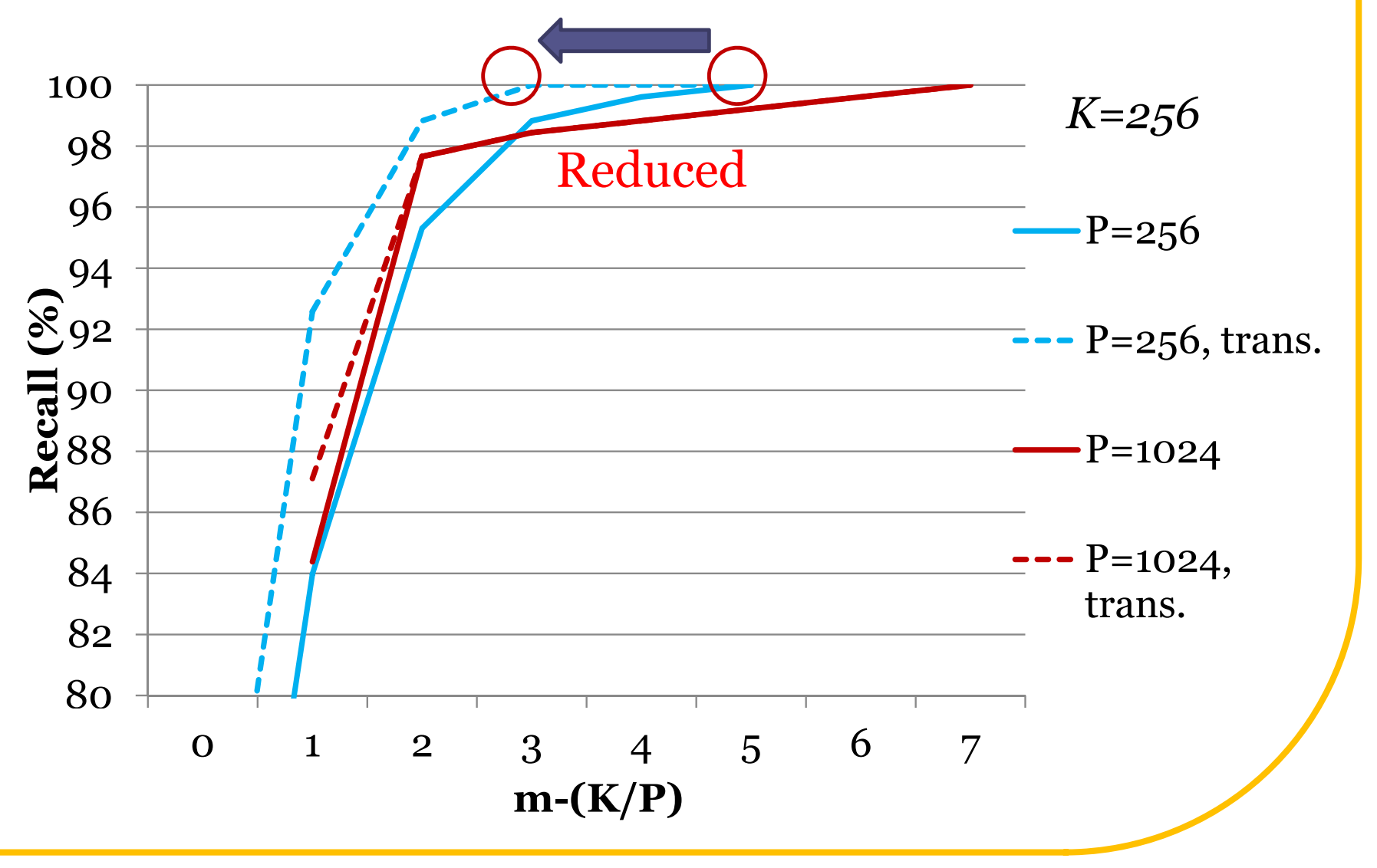
It is very likely that the nearest neighbor is a reference data that is linked to the nearest index to the query. So, this assignment leads to the increase of nearest neighbor candidates .

This assignment guarantees that there is the nearest neighbor in the reference data for each thread with an equal probability.



The selection of m is determined.





Autotuning

The optimal parallelism and the usage of LUT (lookup table) should be determined in order to enhance the performance. The prediction of the execution time is as follows:

T_1: the execution time without LUT

T_2: the execution time using LUT

$$T_1 = \alpha_1 \cdot \frac{N'Dm}{P} + \beta_1 \cdot (\frac{N'}{P} \cdot \frac{m}{2} + \frac{m^2}{4} + \frac{m}{4}) + \beta_2 \cdot (mP \cdot \frac{K}{2} + \frac{K^2}{4} + \frac{K}{4})$$

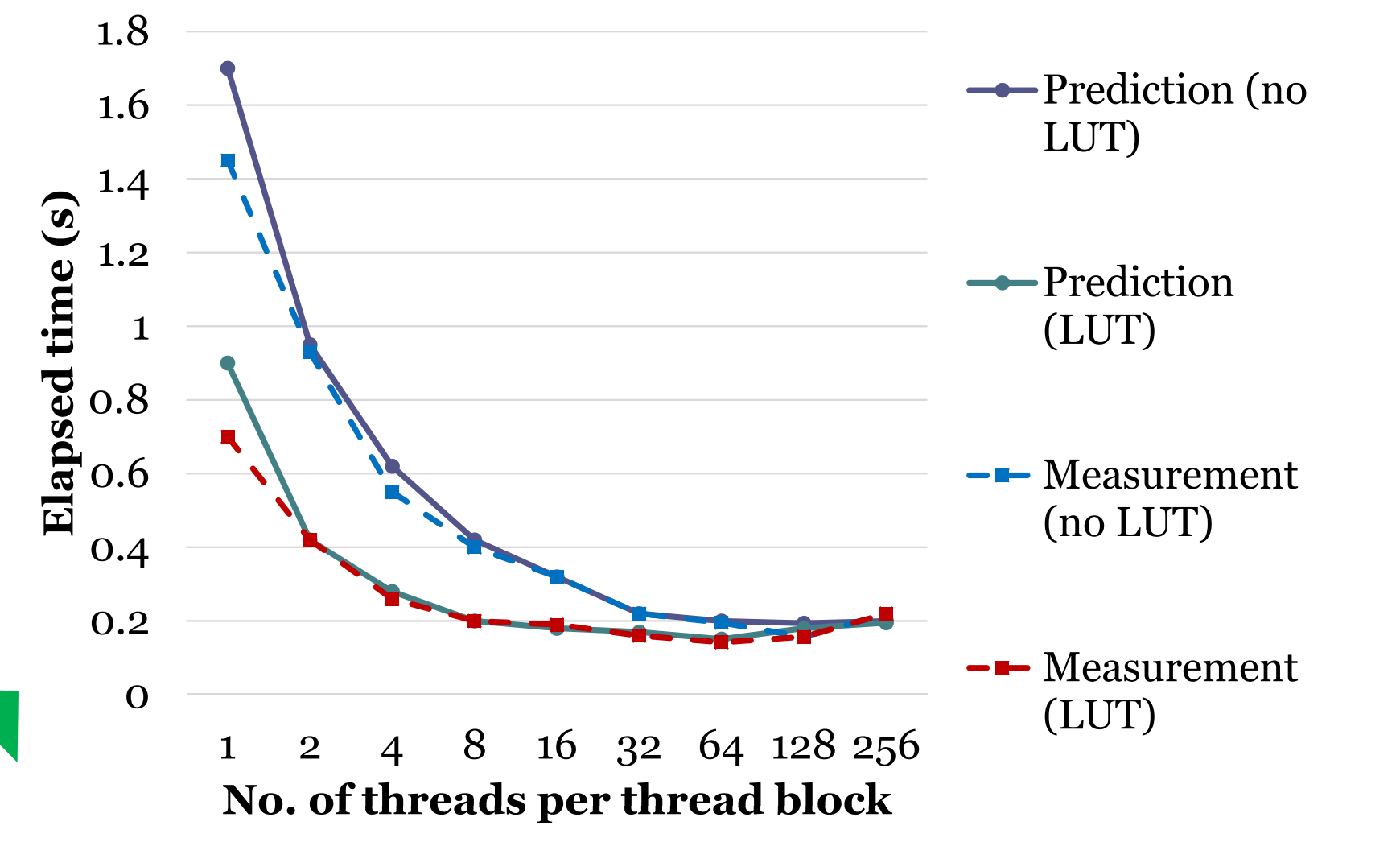
$$T_2 = \alpha_2 \cdot \frac{N'Dm}{P} + \beta_1 \cdot (\frac{N'}{P} \cdot \frac{m}{2} + \frac{m^2}{4} + \frac{m}{4}) + \beta_2 \cdot (mP \cdot \frac{K}{2} + \frac{K^2}{4} + \frac{K}{4}) + \alpha_3 WDC_p$$

N': the number of reference data
D: the dimensions of vectors
C_p: the size of codebook of product quantization
$\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2$: constant parameters that are determined by the least squares method

$$P_{opt1} = \sqrt{\frac{(2\alpha_1 D + \beta_1)N'}{\beta_2 K}}$$

$$P_{opt2} = \sqrt{\frac{(2\alpha_2 D + \beta_1)N'}{\beta_2 K}}$$

The autotuning using the optimal parameters



The absolute values of predicted execution times are not completely identical to the measured execution times. However, the trend of the results are generally similar to each other. Therefore, the validity of our autotuning approach is empirically confirmed.

GPGPU system : PC withIntel Core i7-3770K CPU, 8 GB memory and Nvidia GTX 680 GPU under CentOS 6.4.

Conclusions: In this paper, the baseline of parallelization of the nearest neighbor search with product quantization is described. Our implementation on Nvidia GTX 680 systems achieves a speedup of about 10 times compared with intel Core i7-3770K. In order to enhance the efficiency of the search, we propose Optimistic Search, which utilizes a small number of candidates of nearest neighbors. We also discuss the effectiveness of pseudo matrix transposition for the search and show that the number of candidates of nearest neighbors is reduced by over 50%. In addition, we propose an autotuning method that consists of the preliminary executions and the least squares method. By using parameters that are determined by the preliminary executions, the trend of the predicted execution times is identical to the trend of the measured execution times. Thus, the validity of our approach of the autotuning is also empirically confirmed. In the near future, we will implement our autotuning method on other GPGPU systems, and evaluate the effectiveness of our approach by experiments.

甲南大学
KONAN UNIVERSITY

Contact:
Akiyoshi Wakatani (Konan University, Faculty of Intelligence and Informatics) Email: wakatani @konan-u.ac.jp FAX +81-78-435-2540