|  |
| --- |
| #include <stdio.h> |
|  | #include <omp.h> |
|  | #include <stdlib.h> |
|  | #include <time.h> |
|  | #include <string.h> |
|  |  |
|  | void histogram(int \* data, int n, int \* bins, int k){ |
|  | int num\_bins = 800 / k; |
|  | int number\_of\_threads = omp\_get\_max\_threads(); |
|  | int local\_bins[number\_of\_threads + 1][1024]; //for false sharing |
|  | memset(bins, 0, sizeof(bins)); |
|  | memset(local\_bins, 0, sizeof(local\_bins)); |
|  | #pragma omp parallel |
|  | { |
|  | int id = omp\_get\_thread\_num(); |
|  | int i, j; |
|  |  |
|  | #pragma omp for |
|  | for(i = 0 ; i < n ; ++i) |
|  | local\_bins[id][data[i] / k] ++; |
|  |  |
|  | #pragma omp for |
|  | for(i = 0 ; i < num\_bins; ++i) |
|  | { |
|  | for(j = 0 ; j < number\_of\_threads; ++j) |
|  | { |
|  | bins[i] += local\_bins[j][i]; |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void naive\_histogram(int \* data, int n, int \* bins, int k){ |
|  | int num\_bins = 800 / k; |
|  | int i; |
|  | memset(bins, 0, sizeof(bins)); |
|  | for(i=0;i<n;i++)bins[data[i] / k] ++; |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | int n, k, \*array, \*bins, i, \*bins2; |
|  | long c; |
|  |  |
|  | n = 4 \* 10000 \* 10000; // \* 10000; |
|  | k = 10; |
|  | srand(time(NULL)); |
|  | c = clock(); |
|  | array = (int \*)malloc(sizeof(int) \* n); |
|  | bins = (int \*)malloc(sizeof(int) \* 800 / k); |
|  | bins2 = (int \*)malloc(sizeof(int) \* 800 / k); |
|  | for(i=0;i<n;++i) |
|  | array[i] = rand() % 800; |
|  | printf("%lf\n", ((double)clock() - c) / CLOCKS\_PER\_SEC); |
|  | histogram(array, n, bins, k); |
|  | naive\_histogram(array, n, bins2, k); |
|  | for(i=0;i<800/k;++i) |
|  | if(bins[i] != bins2[i]) |
|  | { |
|  | fprintf(stderr, "%d: %d %d WRONG!!\n",i, bins[i], bins2[i]); |
|  | break; |
|  | } |
|  | if(i== 800 / k) |
|  | fprintf(stderr, "OK!\n"); |
|  |  |
|  | return 0; |
|  | } |