# Отчёт по разработке и тестированию алгоритмов параллелизации вычислений

Выполнил Скопецкий Анатолий, Б9118-09.03.04прогин

## 1 Интегрирование. Библиотека OMP

Количество прямоугольников: 10.000.000.

Результат последовательного решения: 0,287 секунды.

### 1.1 Взятие интеграла с использованием локальных переменных суммы

|  |  |  |
| --- | --- | --- |
| Результат вычисления | Время вычисления | Ускорение |
| 0,667 | 0,350 | 1,000 |
| 0,667 | 0,399 | 0,876 |
| 0,667 | 0,254 | 1,376 |
| 0,667 | 0,378 | 0,924 |
| 0,667 | 0,263 | 1,331 |
| 0,667 | 0,258 | 1,355 |
| 0,667 | 0,282 | 1,241 |
| 0,667 | 0,273 | 1,280 |
| 0,667 | 0,201 | 1,744 |
| 0,667 | 0,192 | 1,821 |
| 0,667 | 0,149 | 2,355 |
| 0,667 | 0,148 | 2,359 |
| 0,667 | 0,143 | 2,450 |
| 0,667 | 0,146 | 2,394 |
| 0,667 | 0,139 | 2,523 |
| 0,667 | 0,170 | 2,061 |
| 0,667 | 0,150 | 2,335 |
| 0,667 | 0,166 | 2,104 |
| 0,667 | 0,162 | 2,156 |
| 0,667 | 0,142 | 2,457 |
| 0,667 | 0,159 | 2,196 |
| 0,667 | 0,163 | 2,144 |
| 0,667 | 0,151 | 2,324 |
| 0,667 | 0,149 | 2,346 |
| 0,667 | 0,126 | 2,781 |
| 0,667 | 0,161 | 2,174 |
| 0,667 | 0,166 | 2,111 |
| 0,667 | 0,100 | 3,485 |
| 0,667 | 0,107 | 3,266 |
| 0,667 | 0,137 | 2,550 |
| 0,667 | 0,138 | 2,534 |
| 0,667 | 0,092 | 3,783 |
| 0,667 | 0,114 | 3,063 |
| 0,667 | 0,105 | 3,324 |
| 0,667 | 0,101 | 3,470 |
| 0,667 | 0,111 | 3,163 |
| 0,667 | 0,115 | 3,033 |
| 0,667 | 0,119 | 2,933 |
| 0,667 | 0,099 | 3,521 |
| 0,667 | 0,108 | 3,248 |
| 0,667 | 0,127 | 2,745 |
| 0,667 | 0,102 | 3,426 |
| 0,667 | 0,116 | 3,016 |
| 0,667 | 0,109 | 3,219 |
| 0,667 | 0,106 | 3,297 |
| 0,667 | 0,110 | 3,190 |
| 0,667 | 0,121 | 2,887 |
| 0,667 | 0,114 | 3,068 |

#### Код программы:

double integrate\_omp\_fs(double a, double b, f\_t f) {

double dx = (b - a) / ndx;

double res = 0;

omp\_set\_num\_threads(numOfThreads);

double\* results;

#pragma omp parallel shared(results, numOfThreads)

{

unsigned t = (unsigned)omp\_get\_thread\_num();

#pragma omp single

{

results = new double[numOfThreads];

for (unsigned i = 0; i < numOfThreads; i++)

results[i] = 0;

if (!results)

abort();

}

for (unsigned i = t; i < ndx; i += numOfThreads) {

results[t] += f(dx \* i + a);

}

#pragma omp barrier

}

for (unsigned i = 0; i < numOfThreads; i++)

res += results[i];

return res \* dx;

}

### 1.2 Взятие интеграла с использованием локальных переменных суммы. Разделение кэша

|  |  |  |
| --- | --- | --- |
| Результат вычисления | Время вычисления | Ускорение |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

#### Код программы:

#define CACHE\_LINE 64u

double integrate\_omp\_base(double a, double b, f\_t f) {

double dx = (b - a) / ndx;

double res = 0;

omp\_set\_num\_threads(numOfThreads);

double\* results;

#pragma omp parallel shared(results, numOfThreads)

{

unsigned t = (unsigned)omp\_get\_thread\_num();

#pragma omp single

{

results = (double\*) \_aligned\_malloc(CACHE\_LINE, numOfThreads \* sizeof(double));

for (unsigned i = 0; i < numOfThreads; i++)

results[i] = 0;

if (!results)

abort();

}

for (unsigned i = t; i < ndx; i += numOfThreads) {

results[t] += f(dx \* i + a);

}

#pragma omp barrier

}

for (unsigned i = 0; i < numOfThreads; i++)

res += results[i];

free(results);

return res \* dx;

}

### 1.3 Взятие интеграла с использованием директивы “критическая секция”

|  |  |  |
| --- | --- | --- |
| Результат вычисления | Время вычисления | Ускорение |
| 0,667 | 0,121 | 1,000 |
| 0,667 | 0,061 | 2,001 |
| 0,667 | 0,041 | 2,932 |
| 0,667 | 0,032 | 3,810 |
| 0,667 | 0,025 | 4,774 |
| 0,667 | 0,021 | 5,696 |
| 0,667 | 0,018 | 6,569 |
| 0,667 | 0,016 | 7,472 |
| 0,667 | 0,015 | 8,285 |
| 0,667 | 0,013 | 9,207 |
| 0,667 | 0,012 | 9,975 |
| 0,667 | 0,011 | 10,647 |
| 0,667 | 0,013 | 9,139 |
| 0,667 | 0,013 | 9,104 |
| 0,667 | 0,027 | 4,426 |
| 0,667 | 0,012 | 9,840 |
| 0,667 | 0,013 | 9,349 |
| 0,667 | 0,010 | 11,721 |
| 0,667 | 0,010 | 12,241 |
| 0,667 | 0,010 | 11,677 |
| 0,667 | 0,009 | 13,259 |
| 0,667 | 0,009 | 13,344 |
| 0,667 | 0,009 | 13,656 |
| 0,667 | 0,011 | 11,216 |
| 0,667 | 0,014 | 8,579 |
| 0,667 | 0,014 | 8,815 |
| 0,667 | 0,013 | 9,097 |
| 0,667 | 0,013 | 9,280 |
| 0,667 | 0,013 | 9,360 |
| 0,667 | 0,013 | 9,467 |
| 0,667 | 0,013 | 9,592 |
| 0,667 | 0,013 | 9,530 |
| 0,667 | 0,012 | 9,712 |
| 0,667 | 0,012 | 9,773 |
| 0,667 | 0,012 | 9,787 |
| 0,667 | 0,012 | 9,751 |
| 0,667 | 0,014 | 8,438 |
| 0,667 | 0,016 | 7,668 |
| 0,667 | 0,015 | 8,325 |
| 0,667 | 0,014 | 8,399 |
| 0,667 | 0,014 | 8,571 |
| 0,667 | 0,014 | 8,514 |
| 0,667 | 0,014 | 8,629 |
| 0,667 | 0,014 | 8,787 |
| 0,667 | 0,014 | 8,512 |
| 0,667 | 0,014 | 8,642 |
| 0,667 | 0,014 | 8,468 |
| 0,667 | 0,014 | 8,884 |

#### Код программы:

double integrate\_omp\_cs(double a, double b, f\_t f)

{

double res = 0;

double dx = (b - a) / ndx;

omp\_set\_num\_threads(numOfThreads);

#pragma omp parallel shared(res, numOfThreads)

{

unsigned t = (unsigned)omp\_get\_thread\_num();

double val = 0;

for (unsigned i = t; i < ndx; i += numOfThreads)

{

val += f(dx \* i + a);

}

#pragma omp critical

{

res += val;

}

}

return res \* dx;

}

### 1.4 Взятие интеграла с использованием директивы “атомарная операция”

|  |  |  |
| --- | --- | --- |
| Результат вычисления | Время вычисления | Ускорение |
| 0,667 | 0,147 | 1,000 |
| 0,667 | 0,073 | 2,000 |
| 0,667 | 0,049 | 2,995 |
| 0,667 | 0,070 | 2,090 |
| 0,667 | 0,050 | 2,951 |
| 0,667 | 0,037 | 3,934 |
| 0,667 | 0,043 | 3,417 |
| 0,667 | 0,028 | 5,158 |
| 0,667 | 0,026 | 5,702 |
| 0,667 | 0,023 | 6,355 |
| 0,667 | 0,021 | 6,905 |
| 0,667 | 0,020 | 7,398 |
| 0,667 | 0,028 | 5,248 |
| 0,667 | 0,028 | 5,263 |
| 0,667 | 0,028 | 5,158 |
| 0,667 | 0,024 | 6,018 |
| 0,667 | 0,022 | 6,809 |
| 0,667 | 0,021 | 7,021 |
| 0,667 | 0,020 | 7,449 |
| 0,667 | 0,019 | 7,726 |
| 0,667 | 0,018 | 8,187 |
| 0,667 | 0,016 | 9,048 |
| 0,667 | 0,016 | 9,158 |
| 0,667 | 0,015 | 9,754 |
| 0,667 | 0,015 | 9,750 |
| 0,667 | 0,015 | 10,094 |
| 0,667 | 0,015 | 10,046 |
| 0,667 | 0,014 | 10,500 |
| 0,667 | 0,013 | 10,918 |
| 0,667 | 0,013 | 10,874 |
| 0,667 | 0,013 | 10,871 |
| 0,667 | 0,013 | 11,191 |
| 0,667 | 0,013 | 11,058 |
| 0,667 | 0,013 | 11,233 |
| 0,667 | 0,013 | 10,908 |
| 0,667 | 0,013 | 11,335 |
| 0,667 | 0,015 | 9,705 |
| 0,667 | 0,015 | 9,875 |
| 0,667 | 0,015 | 9,800 |
| 0,667 | 0,015 | 10,071 |
| 0,667 | 0,015 | 9,698 |
| 0,667 | 0,015 | 9,819 |
| 0,667 | 0,014 | 10,119 |
| 0,667 | 0,015 | 9,930 |
| 0,667 | 0,015 | 9,948 |
| 0,667 | 0,015 | 10,028 |
| 0,667 | 0,015 | 9,861 |
| 0,667 | 0,014 | 10,146 |

#### Код программы:

double integrate\_omp\_atomic(double a, double b, f\_t f)

{

double res = 0;

double dx = (b - a) / ndx;

omp\_set\_num\_threads(numOfThreads);

#pragma omp parallel shared(res, numOfThreads)

{

unsigned t = (unsigned)omp\_get\_thread\_num();

double val = 0;

for (unsigned i = t; i < ndx; i += numOfThreads)

{

val += f((double)(dx \* i + a));

}

#pragma omp atomic

res += val;

}

return res \* dx;

}

### 1.5 Взятие интеграла с использованием директивы “OMP for”

|  |  |  |
| --- | --- | --- |
| Результат вычисления | Время вычисления | Ускорение |
| 0,667 | 1,044 | 1,000 |
| 0,667 | 4,053 | 0,257 |
| 0,667 | 6,409 | 0,163 |
| 0,667 | 7,375 | 0,142 |
| 0,667 | 6,408 | 0,163 |
| 0,667 | 8,774 | 0,119 |
| 0,667 | 7,081 | 0,147 |
| 0,667 | 9,214 | 0,113 |
| 0,667 | 7,726 | 0,135 |
| 0,667 | 9,372 | 0,111 |
| 0,667 | 8,308 | 0,126 |
| 0,667 | 8,502 | 0,123 |
| 0,667 | 8,342 | 0,125 |
| 0,667 | 8,498 | 0,123 |
| 0,667 | 8,329 | 0,125 |
| 0,667 | 8,243 | 0,127 |
| 0,667 | 8,594 | 0,121 |
| 0,667 | 8,433 | 0,124 |
| 0,667 | 8,849 | 0,118 |
| 0,667 | 8,322 | 0,125 |
| 0,667 | 8,356 | 0,125 |
| 0,667 | 8,554 | 0,122 |
| 0,667 | 8,948 | 0,117 |
| 0,667 | 9,061 | 0,115 |
| 0,667 | 8,204 | 0,127 |
| 0,667 | 8,155 | 0,128 |
| 0,667 | 8,103 | 0,129 |
| 0,667 | 8,390 | 0,124 |
| 0,667 | 8,494 | 0,123 |
| 0,667 | 8,158 | 0,128 |
| 0,667 | 8,060 | 0,129 |
| 0,667 | 8,448 | 0,124 |
| 0,667 | 8,501 | 0,123 |
| 0,667 | 8,240 | 0,127 |
| 0,667 | 8,127 | 0,128 |
| 0,667 | 8,213 | 0,127 |
| 0,667 | 8,361 | 0,125 |
| 0,667 | 8,013 | 0,130 |
| 0,667 | 8,056 | 0,130 |
| 0,667 | 7,944 | 0,131 |
| 0,667 | 7,929 | 0,132 |
| 0,667 | 8,016 | 0,130 |
| 0,667 | 7,856 | 0,133 |
| 0,667 | 7,916 | 0,132 |
| 0,667 | 7,841 | 0,133 |
| 0,667 | 7,735 | 0,135 |
| 0,667 | 7,813 | 0,134 |
| 0,667 | 7,933 | 0,132 |

#### Код программы:

double integrate\_omp\_for(double a, double b, f\_t f)

{

double res = 0;

double dx = (b - a) / ndx;

omp\_set\_num\_threads(numOfThreads);

int i;

#pragma omp parallel for shared (res)

for (i = 0; i < ndx; ++i)

{

double val = f((double)(dx \* i + a));

#pragma omp atomic

res += val;

}

return res \* dx;

}

### 1.6 Взятие интеграла с использованием директивы “reduce”

|  |  |  |
| --- | --- | --- |
| Результат вычисления | Время вычисления | Ускорение |
| 0,667 | 0,264 | 1,000 |
| 0,667 | 0,126 | 2,099 |
| 0,667 | 0,084 | 3,141 |
| 0,667 | 0,059 | 4,511 |
| 0,667 | 0,045 | 5,928 |
| 0,667 | 0,050 | 5,301 |
| 0,667 | 0,037 | 7,154 |
| 0,667 | 0,033 | 8,115 |
| 0,667 | 0,033 | 7,910 |
| 0,667 | 0,025 | 10,606 |
| 0,667 | 0,023 | 11,636 |
| 0,667 | 0,021 | 12,624 |
| 0,667 | 0,026 | 10,241 |
| 0,667 | 0,025 | 10,585 |
| 0,667 | 0,030 | 8,762 |
| 0,667 | 0,022 | 12,047 |
| 0,667 | 0,021 | 12,806 |
| 0,667 | 0,020 | 13,320 |
| 0,667 | 0,019 | 14,088 |
| 0,667 | 0,018 | 14,954 |
| 0,667 | 0,017 | 15,526 |
| 0,667 | 0,015 | 17,087 |
| 0,667 | 0,014 | 18,638 |
| 0,667 | 0,014 | 18,780 |
| 0,667 | 0,014 | 18,245 |
| 0,667 | 0,014 | 18,738 |
| 0,667 | 0,014 | 18,965 |
| 0,667 | 0,014 | 19,275 |
| 0,667 | 0,013 | 19,653 |
| 0,667 | 0,013 | 19,819 |
| 0,667 | 0,013 | 20,027 |
| 0,667 | 0,013 | 20,248 |
| 0,667 | 0,013 | 20,219 |
| 0,667 | 0,013 | 20,727 |
| 0,667 | 0,013 | 20,184 |
| 0,667 | 0,013 | 20,518 |
| 0,667 | 0,016 | 16,586 |
| 0,667 | 0,016 | 16,321 |
| 0,667 | 0,016 | 16,891 |
| 0,667 | 0,015 | 17,184 |
| 0,667 | 0,015 | 17,443 |
| 0,667 | 0,015 | 17,654 |
| 0,667 | 0,015 | 17,919 |
| 0,667 | 0,015 | 18,038 |
| 0,667 | 0,015 | 18,163 |
| 0,667 | 0,014 | 18,330 |
| 0,667 | 0,015 | 18,067 |
| 0,667 | 0,015 | 17,997 |

#### Код программы:

double integrate\_omp\_reduce(double a, double b, f\_t f)

{

double res = 0;

double dx = (b - a) / ndx;

omp\_set\_num\_threads(numOfThreads);

int i;

#pragma omp parallel for reduction(+: res)

for (i = 0; i < ndx; ++i)

res += f((double)(dx \* i + a));

return res \* dx;

}

### 1.7 Взятие интеграла с использованием директивы “MutEx”

|  |  |  |
| --- | --- | --- |
| Результат вычисления | Время вычисления | Ускорение |
| 0,667 | 0,150 | 1,000 |
| 0,667 | 0,073 | 2,048 |
| 0,667 | 0,049 | 3,066 |
| 0,667 | 0,069 | 2,172 |
| 0,667 | 0,058 | 2,580 |
| 0,667 | 0,043 | 3,471 |
| 0,667 | 0,034 | 4,362 |
| 0,667 | 0,031 | 4,915 |
| 0,667 | 0,027 | 5,518 |
| 0,667 | 0,025 | 5,993 |
| 0,667 | 0,023 | 6,579 |
| 0,667 | 0,027 | 5,660 |
| 0,667 | 0,027 | 5,615 |
| 0,667 | 0,025 | 6,001 |
| 0,667 | 0,025 | 6,026 |
| 0,667 | 0,023 | 6,479 |
| 0,667 | 0,021 | 7,044 |
| 0,667 | 0,019 | 7,779 |
| 0,667 | 0,019 | 8,009 |
| 0,667 | 0,018 | 8,399 |
| 0,667 | 0,017 | 9,045 |
| 0,667 | 0,016 | 9,210 |
| 0,667 | 0,014 | 10,418 |
| 0,667 | 0,014 | 10,860 |
| 0,667 | 0,015 | 10,117 |
| 0,667 | 0,014 | 10,542 |
| 0,667 | 0,014 | 10,643 |
| 0,667 | 0,014 | 10,779 |
| 0,667 | 0,014 | 10,785 |
| 0,667 | 0,013 | 11,378 |
| 0,667 | 0,013 | 11,420 |
| 0,667 | 0,013 | 11,525 |
| 0,667 | 0,013 | 11,679 |
| 0,667 | 0,013 | 11,765 |
| 0,667 | 0,013 | 11,693 |
| 0,667 | 0,013 | 11,693 |
| 0,667 | 0,015 | 10,022 |
| 0,667 | 0,015 | 10,082 |
| 0,667 | 0,014 | 10,404 |
| 0,667 | 0,014 | 10,430 |
| 0,667 | 0,014 | 10,454 |
| 0,667 | 0,014 | 10,663 |
| 0,667 | 0,014 | 10,503 |
| 0,667 | 0,014 | 10,531 |
| 0,667 | 0,014 | 10,825 |
| 0,667 | 0,014 | 10,748 |
| 0,667 | 0,014 | 10,912 |
| 0,667 | 0,014 | 10,613 |

#### Код программы:

double integrate\_omp\_mtx(double a, double b, f\_t f) {

{

double res = 0;

double dx = (b - a) / ndx;

omp\_set\_num\_threads(numOfThreads);

omp\_lock\_t mtxLock;

omp\_init\_lock(&mtxLock);

#pragma omp parallel shared(res, numOfThreads)

{

unsigned t = (unsigned)omp\_get\_thread\_num();

double val = 0;

for (unsigned i = t; i < ndx; i += numOfThreads)

{

val += f((double)(dx \* i + a));

}

omp\_set\_lock(&mtxLock);

res += val;

omp\_unset\_lock(&mtxLock);

}

return res \* dx;

}

}

## 2 Интегрирование средствами C++

### 2.1 Взятие интеграла с использованием локальных переменных суммы

|  |  |  |
| --- | --- | --- |
| Результат вычисления | Время вычисления | Ускорение |
| 0,667 | 0,195 | 1,000 |
| 0,667 | 0,245 | 0,795 |
| 0,667 | 0,124 | 1,566 |
| 0,667 | 0,339 | 0,574 |
| 0,667 | 0,354 | 0,551 |
| 0,667 | 0,277 | 0,703 |
| 0,667 | 0,266 | 0,732 |
| 0,667 | 0,367 | 0,530 |
| 0,667 | 0,326 | 0,598 |
| 0,667 | 0,200 | 0,973 |
| 0,667 | 0,199 | 0,977 |
| 0,667 | 0,201 | 0,971 |
| 0,667 | 0,186 | 1,047 |
| 0,667 | 0,169 | 1,152 |
| 0,667 | 0,154 | 1,264 |
| 0,667 | 0,139 | 1,405 |
| 0,667 | 0,141 | 1,381 |
| 0,667 | 0,147 | 1,326 |
| 0,667 | 0,134 | 1,451 |
| 0,667 | 0,123 | 1,577 |
| 0,667 | 0,143 | 1,366 |
| 0,667 | 0,126 | 1,550 |
| 0,667 | 0,129 | 1,511 |
| 0,667 | 0,124 | 1,566 |
| 0,667 | 0,114 | 1,706 |
| 0,667 | 0,127 | 1,530 |
| 0,667 | 0,169 | 1,153 |
| 0,667 | 0,128 | 1,527 |
| 0,667 | 0,165 | 1,184 |
| 0,667 | 0,135 | 1,443 |
| 0,667 | 0,137 | 1,418 |
| 0,667 | 0,124 | 1,565 |
| 0,667 | 0,145 | 1,340 |
| 0,667 | 0,134 | 1,450 |
| 0,667 | 0,154 | 1,264 |
| 0,667 | 0,155 | 1,260 |
| 0,667 | 0,177 | 1,098 |
| 0,667 | 0,173 | 1,126 |
| 0,667 | 0,155 | 1,254 |
| 0,667 | 0,127 | 1,537 |
| 0,667 | 0,128 | 1,518 |
| 0,667 | 0,118 | 1,657 |
| 0,667 | 0,120 | 1,626 |
| 0,667 | 0,103 | 1,897 |
| 0,667 | 0,127 | 1,536 |
| 0,667 | 0,104 | 1,875 |
| 0,667 | 0,109 | 1,789 |
| 0,667 | 0,114 | 1,712 |

#### Код программы:

double integrate\_cpp\_fs(double a, double b, f\_t f) {

double dx = (b - a) / ndx;

double res = 0;

unsigned T = numOfThreads;

vector<double> results(T);

auto thread\_proc = [=, &results](unsigned t) {

results[t] = 0;

for (unsigned i = t; i < ndx; i += T)

results[t] += f(dx \* i + a);

};

vector<thread> threads;

for (unsigned t = 1; t < T; ++t)

threads.emplace\_back(thread\_proc, t);

thread\_proc(0);

for (auto& thread : threads)

thread.join();

for (unsigned i = 0; i < T; ++i)

res += results[i];

return res \* dx;

}

### 2.2 Взятие интеграла с использованием локальных переменных суммы. Разделение кэша

|  |  |  |
| --- | --- | --- |
| Результат вычисления | Время вычисления | Ускорение |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

#### Код программы:

double integrate\_cpp\_base(double a, double b, f\_t f) {

double dx = (b - a) / ndx;

double res = 0;

//Получаем количество потоков

unsigned T = numOfThreads;

double\* results = (double\*)\_aligned\_malloc(CACHE\_LINE, max\_threads \* sizeof(double));

for (unsigned i = 0; i < numOfThreads; i++)

results[i] = 0;

//Лямбда, содержащая функции, которые затем будут выполнены каждым из потоков

auto thread\_proc = [=, &results](unsigned t) {

results[t] = 0;

for (unsigned i = t; i < ndx; i += T)

results[t] += f(dx \* i + a);

};

//Создаем массив исполняющих потоков

vector<thread> threads;

for (unsigned t = 1; t < T; ++t)

threads.emplace\_back(thread\_proc, t);

thread\_proc(0);

//Инициализируем выполнение каждого из потоков

for (auto& thread : threads)

thread.join();

//Записываем результаты вычислений каждого из потоков в одну общую переменную

for (unsigned i = 0; i < T; ++i)

res += results[i];

\_aligned\_free(results);

return res \* dx;

}

### 2.3 Взятие интеграла с использованием директивы “Критическая секция”

|  |  |  |
| --- | --- | --- |
| Результат вычисления | Время вычисления | Ускорение |
| 0,667 | 0,209 | 1,000 |
| 0,667 | 0,098 | 2,128 |
| 0,667 | 0,066 | 3,178 |
| 0,667 | 0,076 | 2,739 |
| 0,667 | 0,043 | 4,900 |
| 0,667 | 0,046 | 4,556 |
| 0,667 | 0,041 | 5,121 |
| 0,667 | 0,046 | 4,526 |
| 0,667 | 0,032 | 6,474 |
| 0,667 | 0,029 | 7,302 |
| 0,667 | 0,028 | 7,578 |
| 0,667 | 0,026 | 8,129 |
| 0,667 | 0,043 | 4,832 |
| 0,667 | 0,042 | 4,957 |
| 0,667 | 0,040 | 5,234 |
| 0,667 | 0,038 | 5,486 |
| 0,667 | 0,036 | 5,736 |
| 0,667 | 0,034 | 6,101 |
| 0,667 | 0,031 | 6,672 |
| 0,667 | 0,033 | 6,269 |
| 0,667 | 0,029 | 7,095 |
| 0,667 | 0,030 | 6,970 |
| 0,667 | 0,027 | 7,785 |
| 0,667 | 0,027 | 7,809 |
| 0,667 | 0,025 | 8,420 |
| 0,667 | 0,024 | 8,538 |
| 0,667 | 0,022 | 9,465 |
| 0,667 | 0,023 | 9,128 |
| 0,667 | 0,021 | 10,017 |
| 0,667 | 0,020 | 10,413 |
| 0,667 | 0,019 | 11,022 |
| 0,667 | 0,019 | 11,166 |
| 0,667 | 0,019 | 11,212 |
| 0,667 | 0,018 | 11,829 |
| 0,667 | 0,017 | 12,102 |
| 0,667 | 0,020 | 10,249 |
| 0,667 | 0,020 | 10,300 |
| 0,667 | 0,020 | 10,477 |
| 0,667 | 0,019 | 11,019 |
| 0,667 | 0,020 | 10,608 |
| 0,667 | 0,018 | 11,276 |
| 0,667 | 0,018 | 11,559 |
| 0,667 | 0,018 | 11,442 |
| 0,667 | 0,020 | 10,261 |
| 0,667 | 0,018 | 11,668 |
| 0,667 | 0,018 | 11,617 |
| 0,667 | 0,018 | 11,817 |
| 0,667 | 0,017 | 12,251 |

#### Код программы:

double integrate\_cpp\_cs(double a, double b, f\_t f) {

double dx = (b - a) / ndx;

double res = 0;

unsigned T = numOfThreads;

CRITICAL\_SECTION cs;

InitializeCriticalSection(&cs);

auto thread\_process = [=, &res, &cs](unsigned t) {

double l\_res = 0;

for (unsigned i = t; i < ndx; i += T)

l\_res += f((float)(dx \* i + a));

{

EnterCriticalSection(&cs);

res += l\_res;

LeaveCriticalSection(&cs);

}

};

vector<thread> threads;

for (unsigned t = 1; t < T; ++t)

threads.emplace\_back(thread\_process, t);

thread\_process(0);

for (auto& thread : threads)

thread.join();

DeleteCriticalSection(&cs);

return res \* dx;

}

### 2.4 Взятие интеграла с использованием директивы “Атомарная операция”

|  |  |  |
| --- | --- | --- |
| Результат вычисления | Время вычисления | Ускорение |
| 0,667 | 0,285 | 1,000 |
| 0,667 | 0,136 | 2,094 |
| 0,667 | 0,086 | 3,307 |
| 0,667 | 0,059 | 4,799 |
| 0,667 | 0,048 | 5,964 |
| 0,667 | 0,040 | 7,110 |
| 0,667 | 0,044 | 6,498 |
| 0,667 | 0,031 | 9,323 |
| 0,667 | 0,027 | 10,400 |
| 0,667 | 0,025 | 11,516 |
| 0,667 | 0,023 | 12,531 |
| 0,667 | 0,021 | 13,496 |
| 0,667 | 0,040 | 7,168 |
| 0,667 | 0,036 | 7,991 |
| 0,667 | 0,033 | 8,625 |
| 0,667 | 0,030 | 9,368 |
| 0,667 | 0,032 | 9,026 |
| 0,667 | 0,033 | 8,674 |
| 0,667 | 0,031 | 9,171 |
| 0,667 | 0,028 | 10,259 |
| 0,667 | 0,024 | 11,720 |
| 0,667 | 0,023 | 12,201 |
| 0,667 | 0,022 | 12,751 |
| 0,667 | 0,021 | 13,404 |
| 0,667 | 0,018 | 15,557 |
| 0,667 | 0,018 | 16,010 |
| 0,667 | 0,017 | 17,019 |
| 0,667 | 0,016 | 17,572 |
| 0,667 | 0,016 | 18,244 |
| 0,667 | 0,015 | 18,848 |
| 0,667 | 0,015 | 19,433 |
| 0,667 | 0,014 | 20,017 |
| 0,667 | 0,014 | 19,893 |
| 0,667 | 0,014 | 20,334 |
| 0,667 | 0,013 | 21,662 |
| 0,667 | 0,013 | 22,213 |
| 0,667 | 0,019 | 15,170 |
| 0,667 | 0,018 | 15,564 |
| 0,667 | 0,018 | 15,706 |
| 0,667 | 0,018 | 15,916 |
| 0,667 | 0,018 | 16,151 |
| 0,667 | 0,018 | 16,171 |
| 0,667 | 0,017 | 16,416 |
| 0,667 | 0,018 | 16,030 |
| 0,667 | 0,017 | 16,564 |
| 0,667 | 0,017 | 16,778 |
| 0,667 | 0,017 | 16,862 |
| 0,667 | 0,017 | 17,083 |

#### Код программы:

double integrate\_cpp\_atomic(double a, double b, f\_t f) {

using std::atomic;

double dx = (b - a) / ndx;

atomic<double> res = 0;

unsigned T = numOfThreads;

auto thread\_process = [=, &res](unsigned t) {

double l\_res = 0;

for (unsigned i = t; i < ndx; i += T)

l\_res += f((float)(dx \* i + a));

res.store(res.load() + l\_res);

};

vector<thread> threads;

for (unsigned t = 1; t < T; ++t)

threads.emplace\_back(thread\_process, t);

thread\_process(0);

for (auto& thread : threads)

thread.join();

return res \* dx;

}

### 1.6 Взятие интеграла с использованием директивы “reduce”

|  |  |  |
| --- | --- | --- |
| Результат вычисления | Время вычисления | Ускорение |
| 0,667 | 0,264 | 1,000 |
| 0,667 | 0,126 | 2,099 |
| 0,667 | 0,084 | 3,141 |
| 0,667 | 0,059 | 4,511 |
| 0,667 | 0,045 | 5,928 |
| 0,667 | 0,050 | 5,301 |
| 0,667 | 0,037 | 7,154 |
| 0,667 | 0,033 | 8,115 |
| 0,667 | 0,033 | 7,910 |
| 0,667 | 0,025 | 10,606 |
| 0,667 | 0,023 | 11,636 |
| 0,667 | 0,021 | 12,624 |
| 0,667 | 0,026 | 10,241 |
| 0,667 | 0,025 | 10,585 |
| 0,667 | 0,030 | 8,762 |
| 0,667 | 0,022 | 12,047 |
| 0,667 | 0,021 | 12,806 |
| 0,667 | 0,020 | 13,320 |
| 0,667 | 0,019 | 14,088 |
| 0,667 | 0,018 | 14,954 |
| 0,667 | 0,017 | 15,526 |
| 0,667 | 0,015 | 17,087 |
| 0,667 | 0,014 | 18,638 |
| 0,667 | 0,014 | 18,780 |
| 0,667 | 0,014 | 18,245 |
| 0,667 | 0,014 | 18,738 |
| 0,667 | 0,014 | 18,965 |
| 0,667 | 0,014 | 19,275 |
| 0,667 | 0,013 | 19,653 |
| 0,667 | 0,013 | 19,819 |
| 0,667 | 0,013 | 20,027 |
| 0,667 | 0,013 | 20,248 |
| 0,667 | 0,013 | 20,219 |
| 0,667 | 0,013 | 20,727 |
| 0,667 | 0,013 | 20,184 |
| 0,667 | 0,013 | 20,518 |
| 0,667 | 0,016 | 16,586 |
| 0,667 | 0,016 | 16,321 |
| 0,667 | 0,016 | 16,891 |
| 0,667 | 0,015 | 17,184 |
| 0,667 | 0,015 | 17,443 |
| 0,667 | 0,015 | 17,654 |
| 0,667 | 0,015 | 17,919 |
| 0,667 | 0,015 | 18,038 |
| 0,667 | 0,015 | 18,163 |
| 0,667 | 0,014 | 18,330 |
| 0,667 | 0,015 | 18,067 |
| 0,667 | 0,015 | 17,997 |

#### Код программы:

//------------------------------------------------------

class barrier

{

const unsigned m\_T\_max;

unsigned m\_T;

bool barrier\_id = false;

std::condition\_variable cv;

std::mutex mtx;

public:

barrier(unsigned T) :m\_T\_max(T), m\_T(T) {}

void arrive\_and\_wait()

{

std::unique\_lock<std::mutex> lock(mtx);

bool my\_barrier\_id = barrier\_id;

if (--m\_T > 0)

{

while (my\_barrier\_id == barrier\_id)

cv.wait(lock);

}

else

{

cv.notify\_all();

m\_T = m\_T\_max;

barrier\_id = !my\_barrier\_id;

}

}

};

namespace {

//------------------------------------------------------

template <class ElementType, class binary\_fn>

ElementType reduce\_par(ElementType\* V, unsigned count, binary\_fn f, ElementType zero)

{

unsigned j = 1;

constexpr unsigned k = 2;

std::vector<std::thread> threads;

unsigned T =numOfThreads;

for (unsigned t = 1; t < T; t++)

threads.emplace\_back(std::thread{});

while (count > j)

{

auto thread\_fn = [k, count, j, T, zero, V, f](unsigned t)

{

for (unsigned i = t \* j \* k; i < count; i += T \* j \* k)

{

ElementType other = zero;

if (i + j < count)

other = V[i + j];

V[i] = f(V[i], other);

}

};

for (unsigned t = 1; t < T; t++)

threads[t - 1] = std::thread(thread\_fn, t);

thread\_fn(0);

for (auto& thread : threads)

thread.join();

j \*= k;

}

return V[0];

}

template <class binary\_fn, class unary\_fn, class ElementType>

auto reduce\_par\_2(binary\_fn f, unary\_fn get, ElementType x0, ElementType xn, ElementType step, ElementType zero)

{

struct element\_t

{

alignas(std::hardware\_destructive\_interference\_size) ElementType value;

};

unsigned T = numOfThreads;

static std::vector<element\_t> reduction\_buffer(std::thread::hardware\_concurrency(), element\_t{ 0.0 });

std::vector<std::thread> threads;

barrier bar{ T };

auto thread\_proc = [f, get, x0, xn, step, zero, T, &bar](unsigned t)

{

std::size\_t count = (std::size\_t)ElementType((xn - x0) / step);

std::size\_t nt = count / T, it0 = nt \* t;

ElementType my\_result = zero;

if (nt < (count % T))

++nt;

else

it0 += count % T;

std::size\_t it1 = it0 + nt;

ElementType x = x0 + step \* it0;

for (std::size\_t i = it0; i < it1; ++i, x += step)

my\_result = f(my\_result, get(x));

reduction\_buffer[t].value = my\_result;

bar.arrive\_and\_wait();

for (std::size\_t reduction\_distance = 1u, reduction\_next = 2; reduction\_distance < T; reduction\_distance = reduction\_next, reduction\_next += reduction\_next)

{

if (t + reduction\_distance < T && (t & reduction\_next - 1) == 0)

reduction\_buffer[t].value = f(reduction\_buffer[t].value, reduction\_buffer[t + reduction\_distance].value);

bar.arrive\_and\_wait();

}

};

for (unsigned t = 1; t < T; ++t)

threads.emplace\_back(thread\_proc, t);

thread\_proc(0);

for (auto& thread : threads)

thread.join();

return reduction\_buffer[0].value;

}

//------------------------------------------------------

}

### 2.5 Взятие интеграла с использованием директивы “MutEx”

|  |  |  |
| --- | --- | --- |
| Результат вычисления | Время вычисления | Ускорение |
| 0,667 | 0,287 | 1,000 |
| 0,667 | 0,133 | 2,150 |
| 0,667 | 0,098 | 2,918 |
| 0,667 | 0,107 | 2,692 |
| 0,667 | 0,059 | 4,834 |
| 0,667 | 0,054 | 5,304 |
| 0,667 | 0,047 | 6,164 |
| 0,667 | 0,039 | 7,293 |
| 0,667 | 0,034 | 8,498 |
| 0,667 | 0,030 | 9,414 |
| 0,667 | 0,029 | 10,033 |
| 0,667 | 0,041 | 6,943 |
| 0,667 | 0,045 | 6,432 |
| 0,667 | 0,043 | 6,672 |
| 0,667 | 0,040 | 7,111 |
| 0,667 | 0,038 | 7,534 |
| 0,667 | 0,035 | 8,145 |
| 0,667 | 0,039 | 7,358 |
| 0,667 | 0,032 | 9,028 |
| 0,667 | 0,031 | 9,285 |
| 0,667 | 0,030 | 9,649 |
| 0,667 | 0,027 | 10,559 |
| 0,667 | 0,026 | 11,205 |
| 0,667 | 0,024 | 11,842 |
| 0,667 | 0,024 | 12,093 |
| 0,667 | 0,021 | 13,431 |
| 0,667 | 0,020 | 14,106 |
| 0,667 | 0,024 | 11,970 |
| 0,667 | 0,020 | 14,457 |
| 0,667 | 0,018 | 15,680 |
| 0,667 | 0,019 | 15,276 |
| 0,667 | 0,020 | 14,207 |
| 0,667 | 0,017 | 16,847 |
| 0,667 | 0,016 | 17,736 |
| 0,667 | 0,018 | 15,530 |
| 0,667 | 0,016 | 18,250 |
| 0,667 | 0,020 | 14,340 |
| 0,667 | 0,019 | 14,939 |
| 0,667 | 0,019 | 15,312 |
| 0,667 | 0,020 | 14,688 |
| 0,667 | 0,019 | 15,119 |
| 0,667 | 0,018 | 16,016 |
| 0,667 | 0,018 | 15,679 |
| 0,667 | 0,018 | 16,040 |
| 0,667 | 0,018 | 16,172 |
| 0,667 | 0,018 | 16,001 |
| 0,667 | 0,017 | 16,603 |
| 0,667 | 0,018 | 16,363 |

#### Код программы:

double integrate\_cpp\_mtx(double a, double b, f\_t f) {

using std::mutex;

using std::lock\_guard;

double dx = (b - a) / ndx;

double res = 0;

unsigned T = numOfThreads;

mutex mtx;

auto thread\_process = [=, &res, &mtx](unsigned t) {

double l\_res = 0;

for (unsigned i = t; i < ndx; i += T)

l\_res += f((float)(dx \* i + a));

{

lock\_guard<mutex> lck(mtx);

res += l\_res;

}

};

vector<thread> threads;

for (unsigned t = 1; t < T; ++t)

threads.emplace\_back(thread\_process, t);

thread\_process(0);

for (auto& thread : threads)

thread.join();

return res \* dx;

}