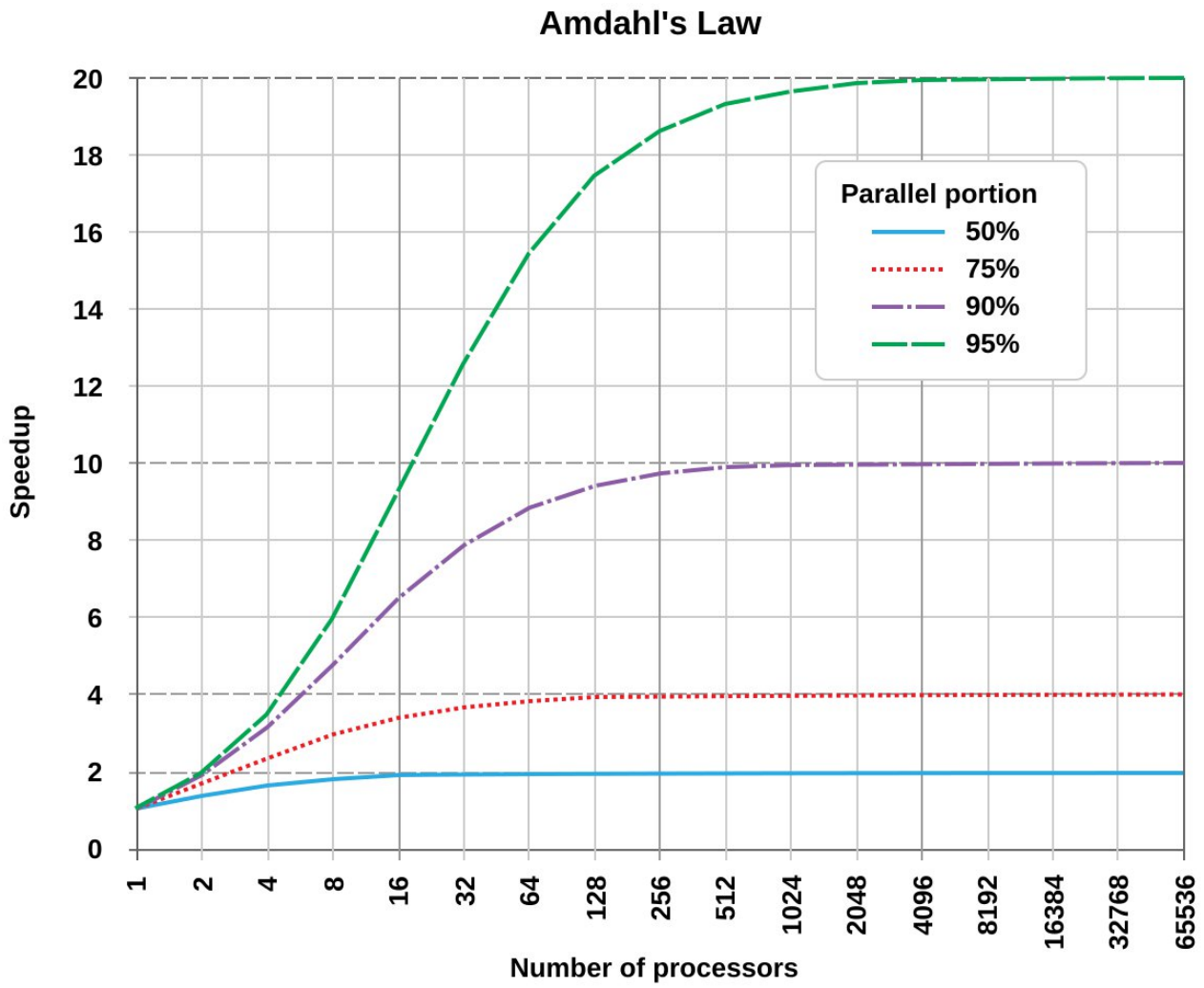
С помощью разработанного приложения была реализованна многопоточная обработка данных и продемонстрированы некоторые особенности изменения эффективности обработки данных в зависимости от сложности задачи и числа используемых потоков.

Многопоточная обработка может увеличить производительность, однако даже в идеальном случае существует ограничение роста производительности, которое определяется с помощью закона Амдала.



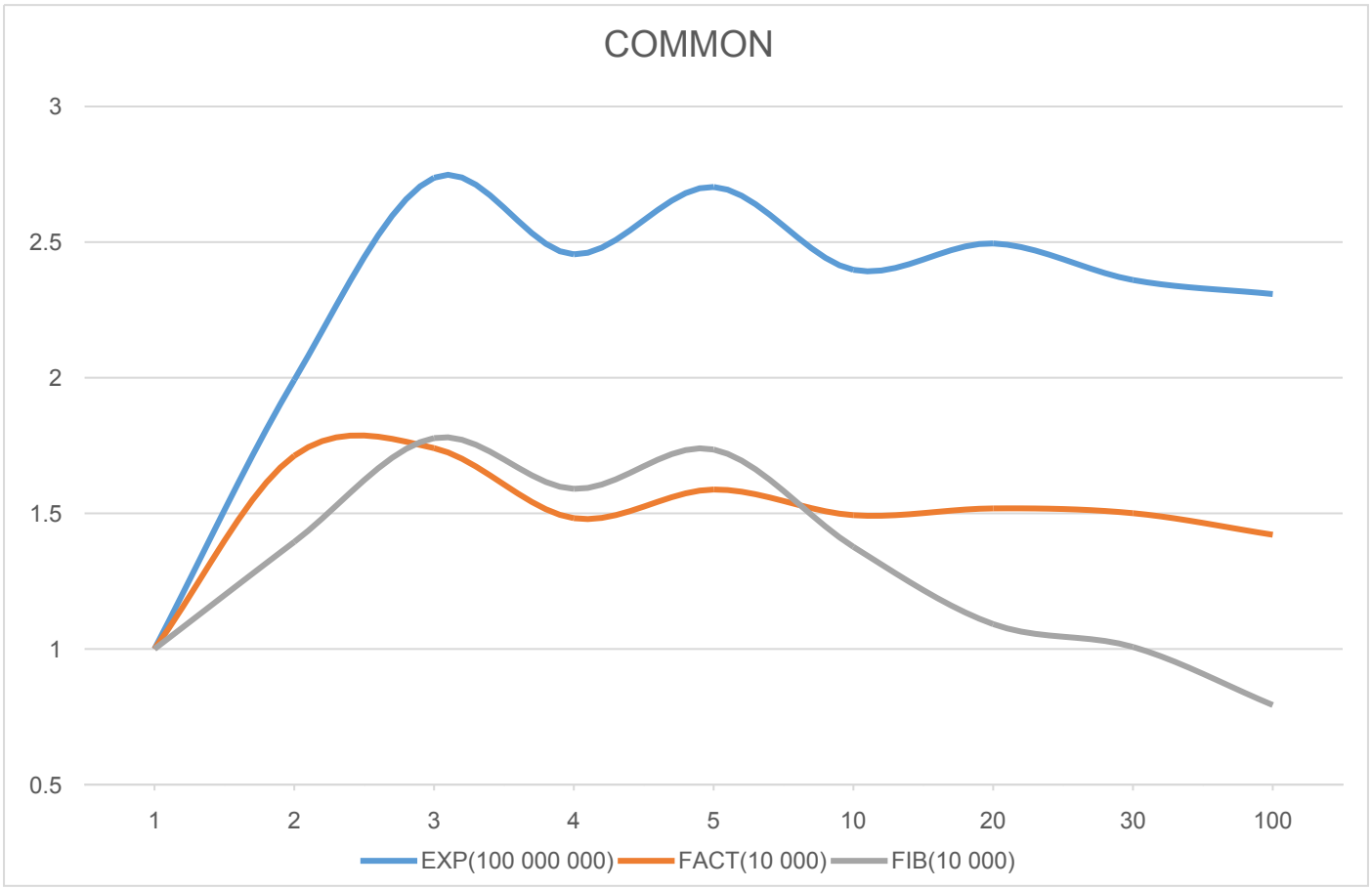
В реальности (неидеальных условиях)ожидается следующее поведение многопоточного обработчика данных:

1. При небольшом объёме обрабатываемых данных увеличение производительности за счёт многопоточности не будет наблюдаться или производительность даже снизиться. Это связано с затратами на обслуживание потоков, большим объёмом используемой памяти и, как следствие, большими затратами на манипуляции с памятью, например сборку мусора. Пока рост этих затрат превышает выгоду от использования нескольких потоков, будет наблюдать снижение производительности.
2. При усложнении обработки элементов данных польза от многопоточных вычислений станет более очевидной, так как при простых операциях

процессор будет тратить больше своих мощностей и времени на обслуживание созданных потоков, чем на сами вычисления.

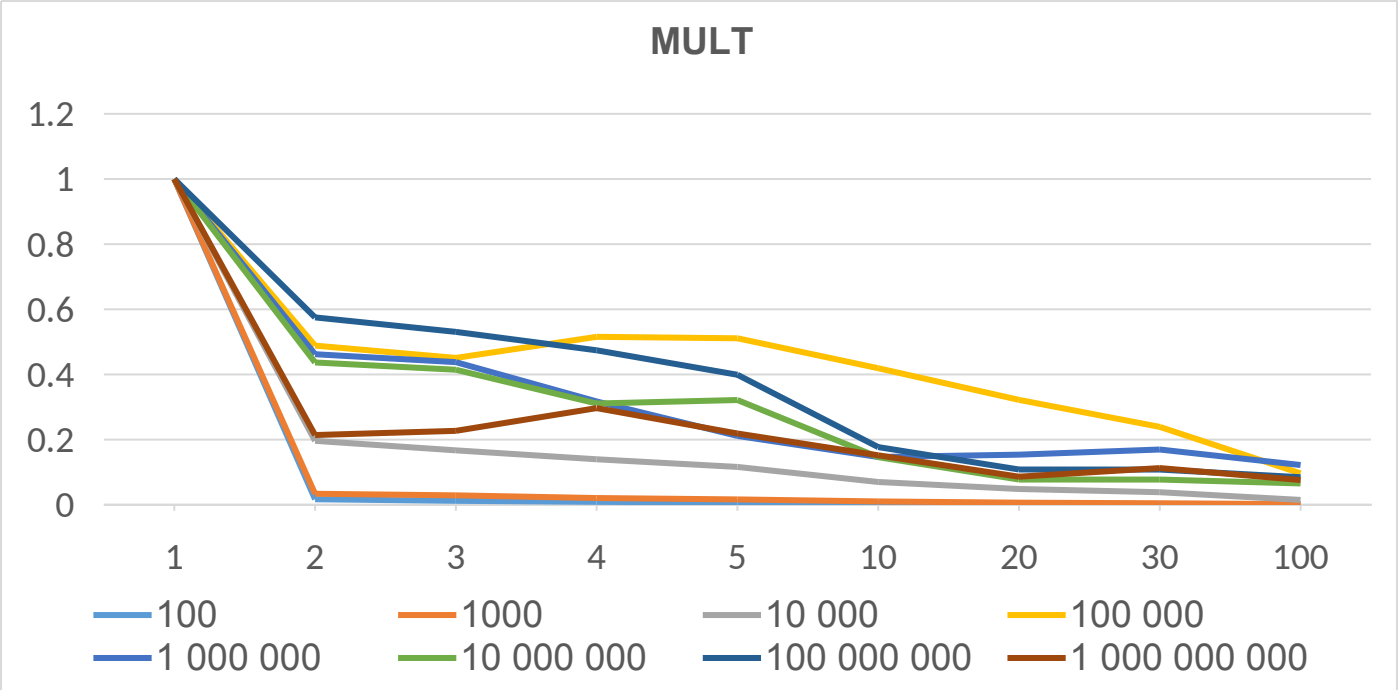
3. Увеличение числа потоков свыше некоторого числа не будет давать прирост производительности, так как «узким местом» станет число реальных потоков процессора. Создание большего числа программных потоков лишь увеличит издержки.

Ниже представлен сводный график, на котором отображена зависимость ускорения от числа потоков для вычислений на основе возведения в степень, получения факториала и нахождения числа Фибоначчи. Использовались данные, полученные для максимальной обсчитанной длины задачи: 100 000 000 для возведения в степень и 10 000 для двух других.



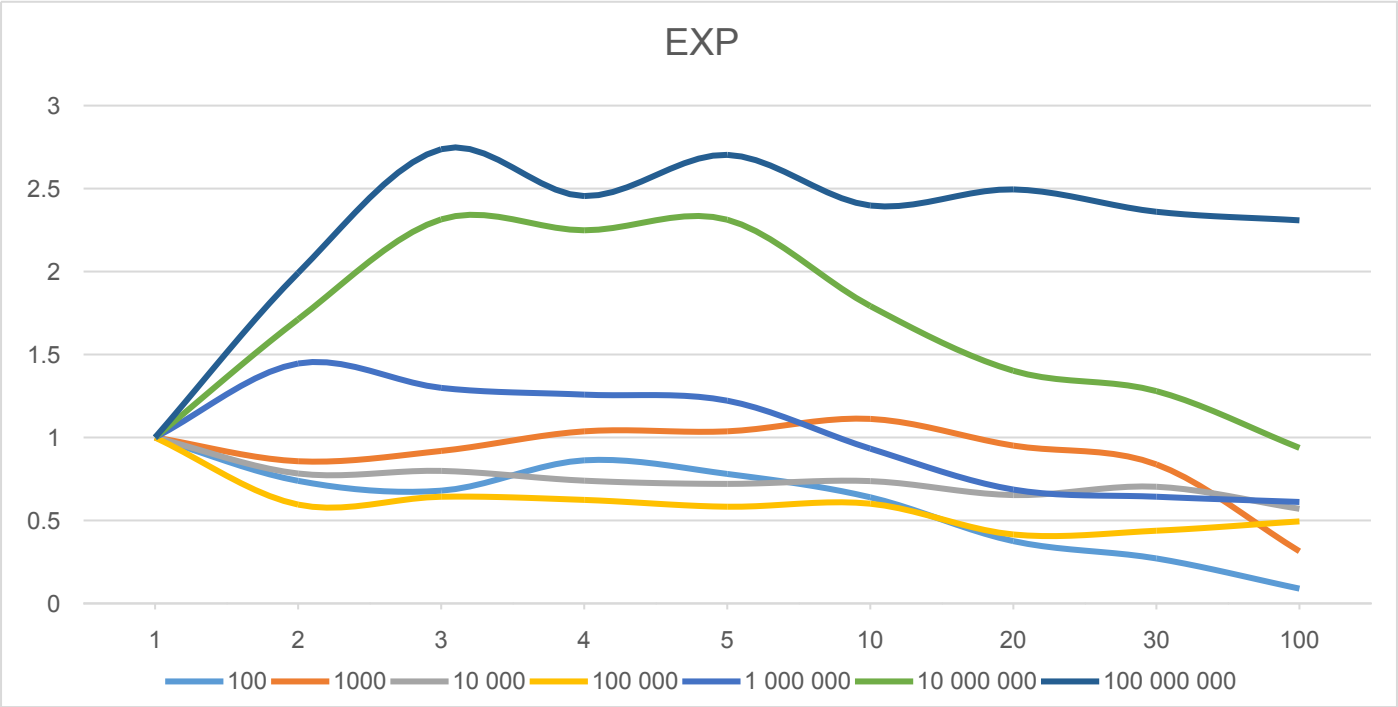
Данные, отображённые на этом графике, подтверждают высказанное выше предположение 3 о поведении системы многопоточной обработки данных. Как видно на графике, сначала наблюдается значительный прирост производительности при добавлении потоков обработки, но как только их число превышает 2-3 потока рост производительности престаёт наблюдаться, а при дальнейшем увеличении числа потоков производительность и вовсе начинает снижаться.

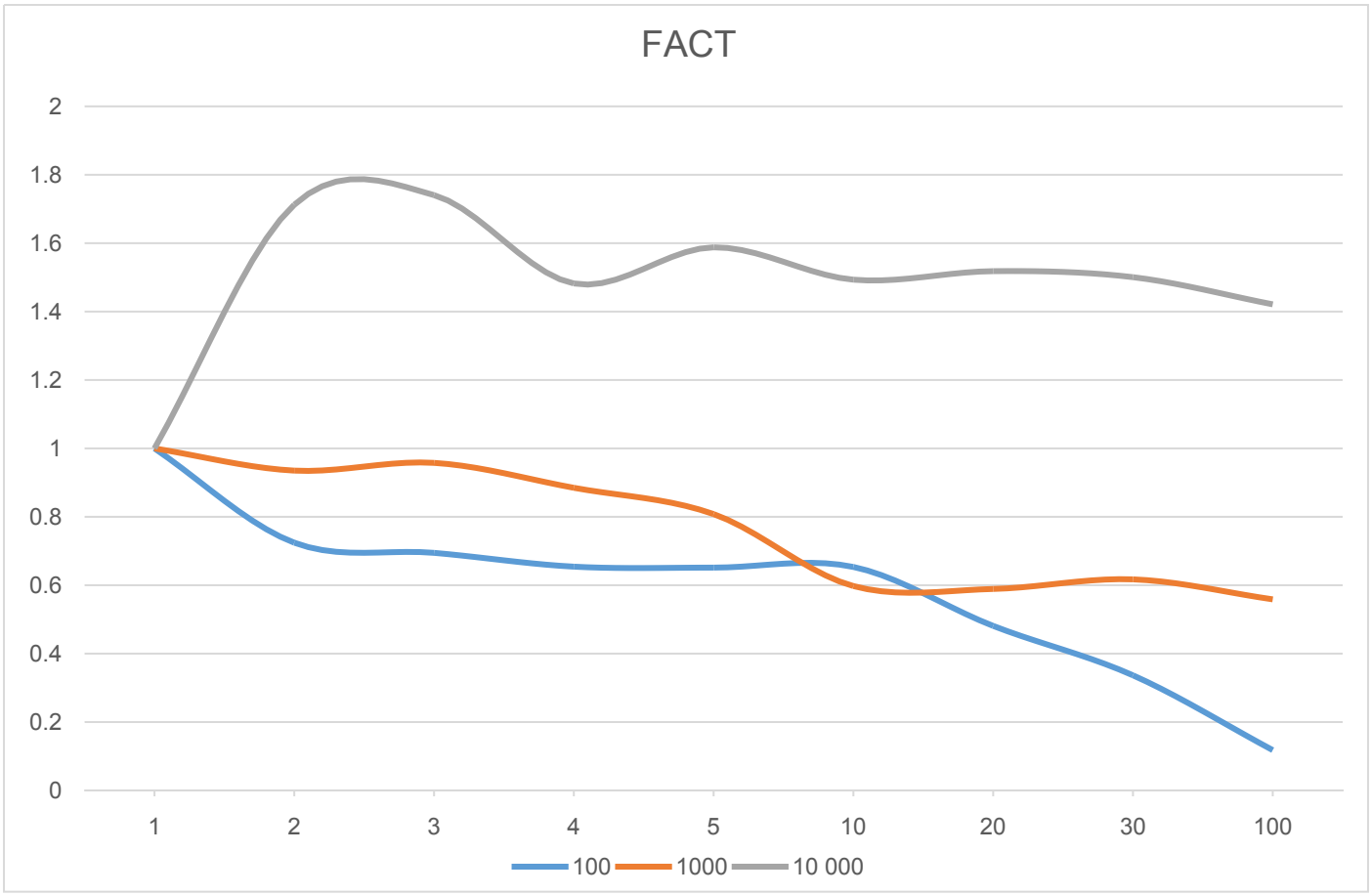
Для операции умножения был получен следующий график. Он наглядно демонстрирует, что издержки многопоточной обработки могут превышать пользу в том случае, если каждое отдельное вычисления слишком просто (как операция умножения на число).

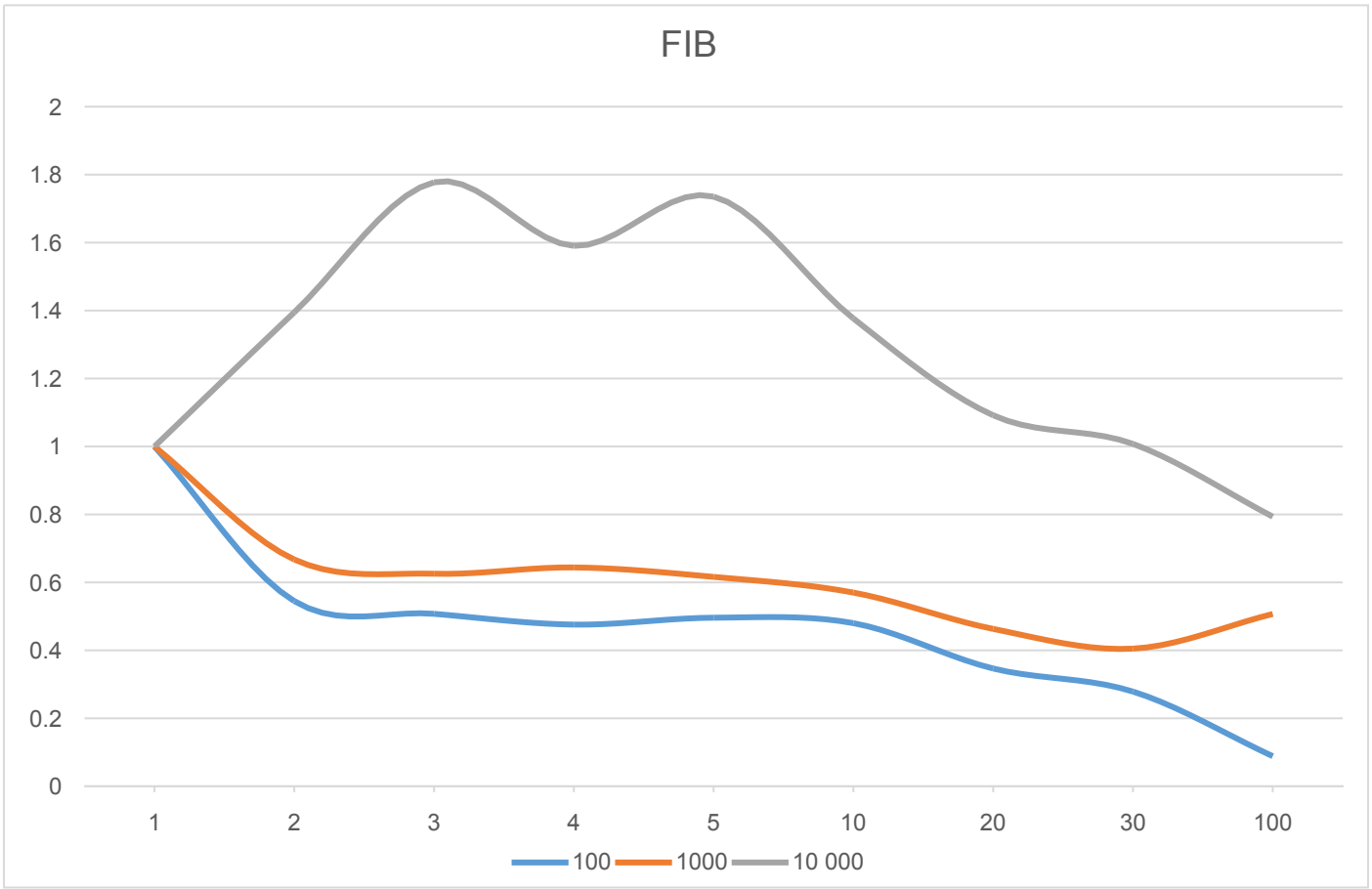


На данном графике видно, что при любом числе потоков и любом объёме данных последовательная обработка была эффективнее многопоточной.

Ниже представлены графики для трёх других операций. В совокупности с графиков умножения они подтверждают предположения 1 и 2. Многопоточная обработка действительно оказывалась более эффективной при большем размере и сложности задачи.




FACT
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
2
3
4
5
10
20
30
100
100
1000
10 000


FIB
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
2
3
4
5
10
20
30
100
100
1000
10 000