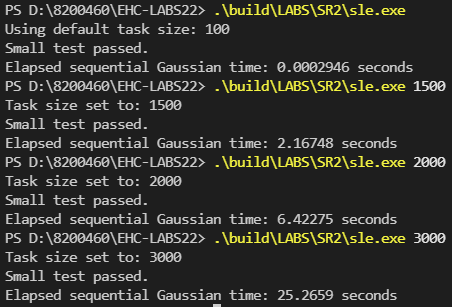
**Отчет по лабораторной работе №2**

*Островского Артема ИВТ-11М*

**Задание 1.**

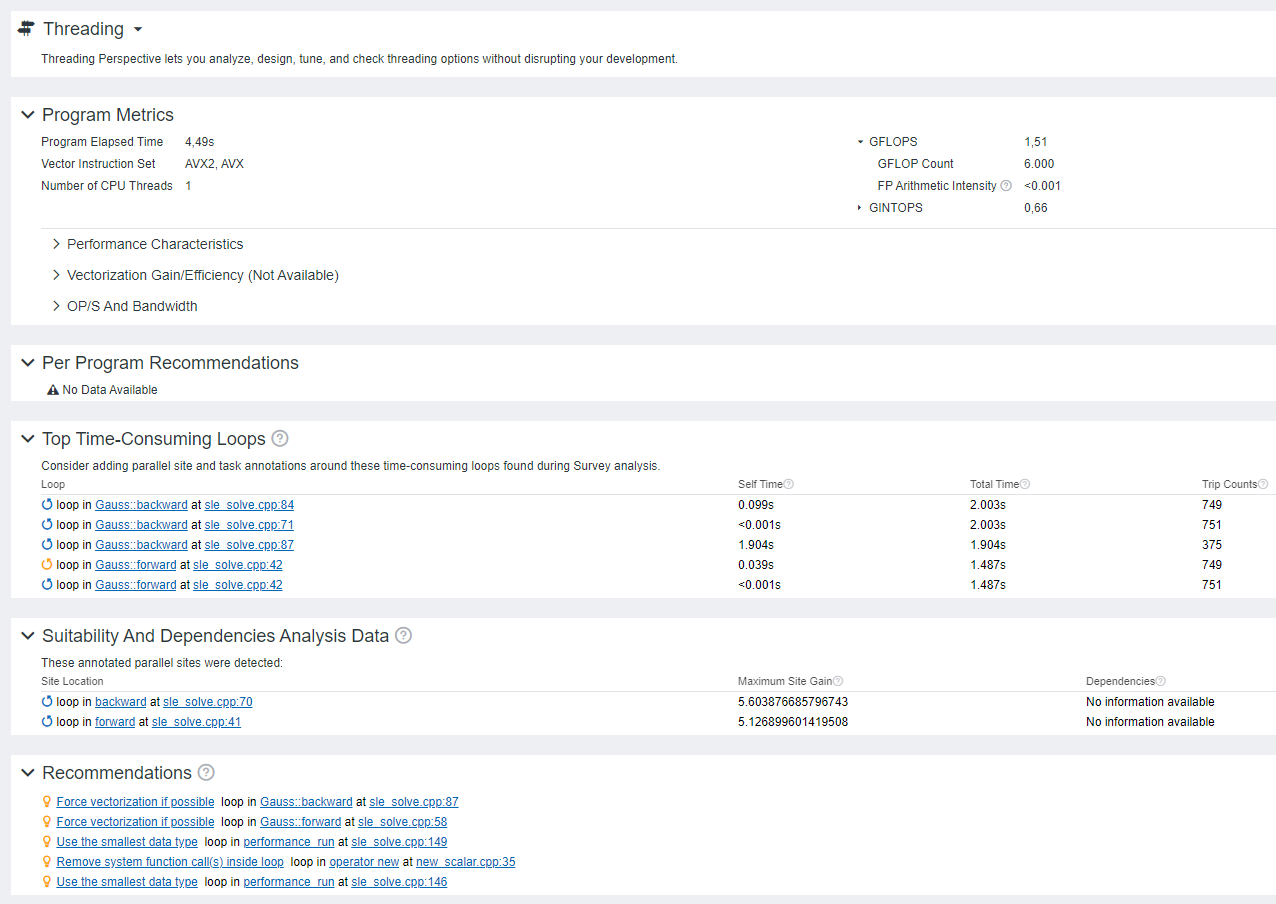
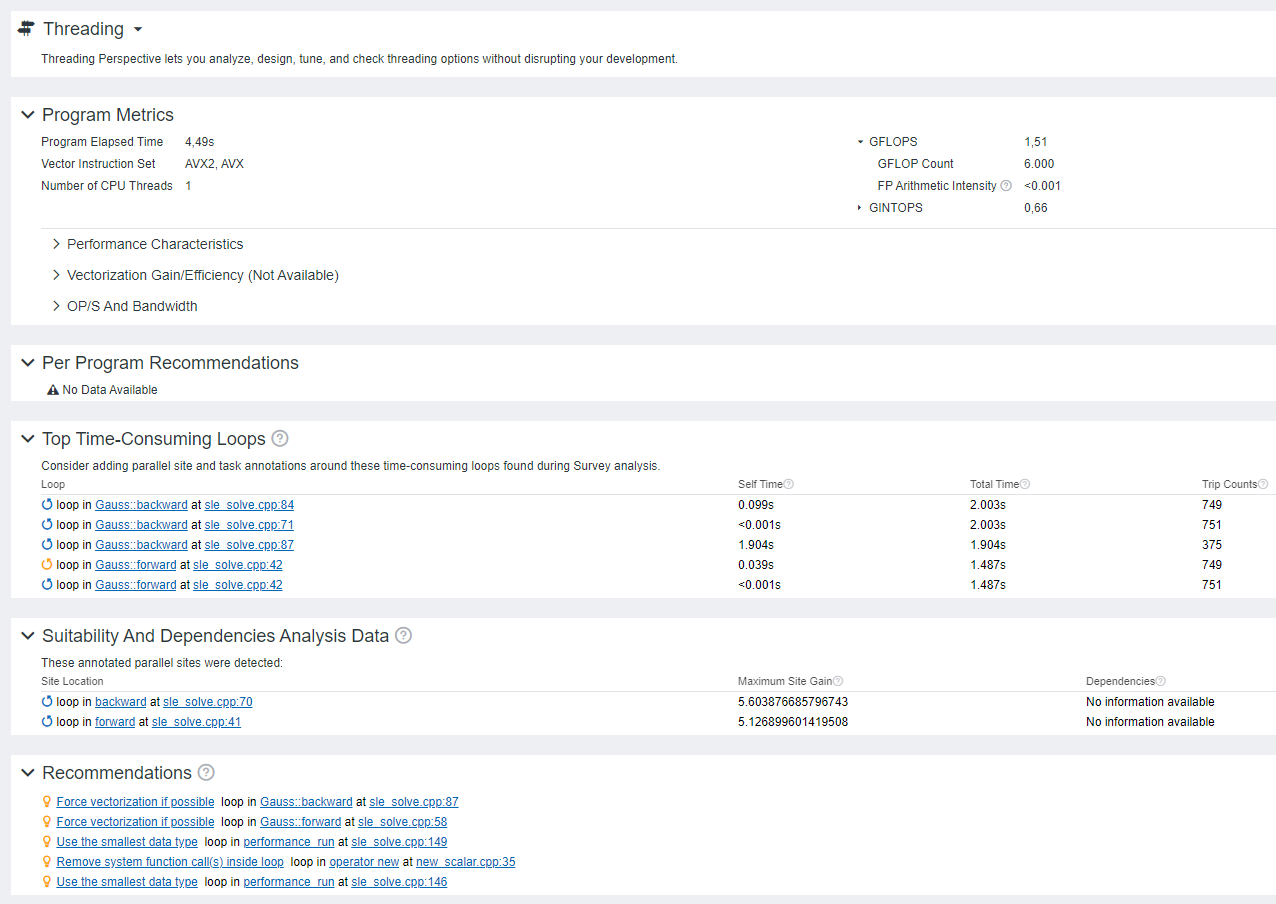
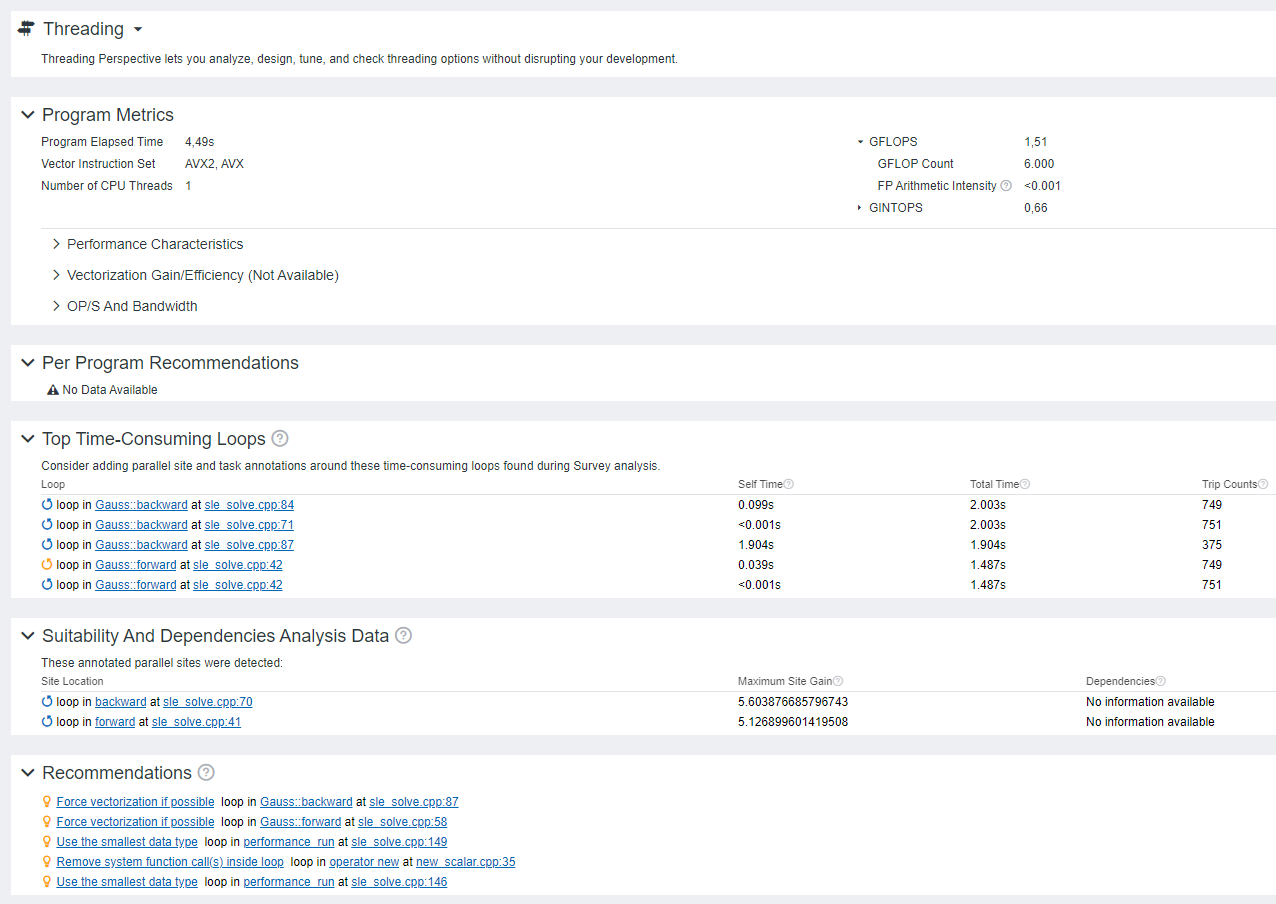
Необходимо собрать проект с исходным файлом sle\_solve.cpp и запустить собранный исполняемый файл. Оценить время работы программы и корректность ее работы. Зафиксируйте это в отчете.

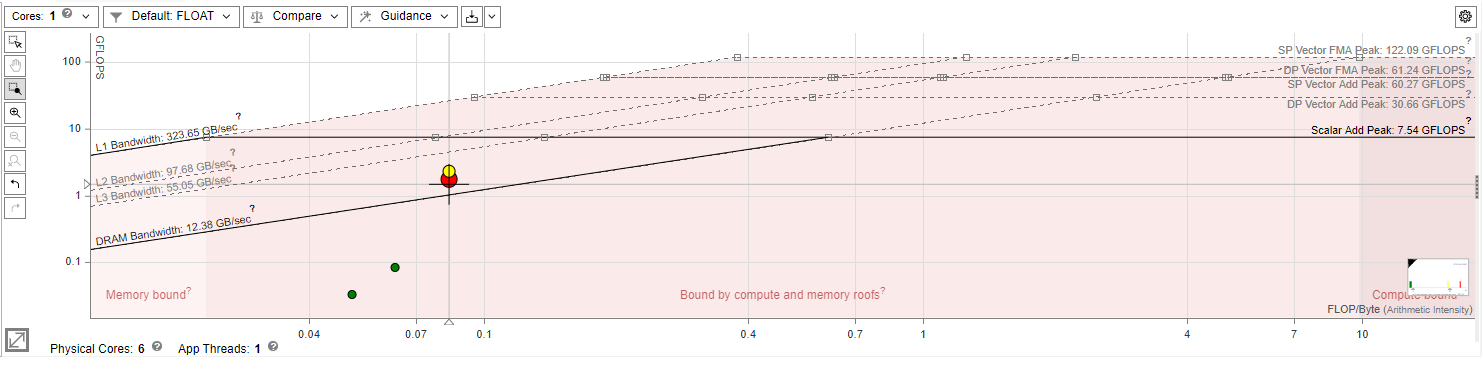


Количество итераций всех случаях никак не отразилось на корректности работы программы. Это видно по выводу **«Small test passed»** во всех трёх запусках программы.

**Задание 2.**

С помощью инструментария Advisor необходимо получить метрики исполнения программы в последовательном режиме: GFLOPs, количество используемых потоков, тип векторных команд, определить хотспоты, построить roofline, сохранить snapshot.





**Задание 3.**

Оцените потенциальный прирост производительности при введении параллелизма через Suitability анализ.

Анализируем два основных ХОТ-спота на потенциальный прирост производительности:

|  |  |
| --- | --- |
| void forward()      {          ANNOTATE\_SITE\_BEGIN(forward);          for (int i = 0; i < m; ++i)          {              type lead\_elem = coefs[i][i];              ANNOTATE\_TASK\_BEGIN(normalize\_row);              for (int j = 0; j < n; ++j)              {                  coefs[i][j] = coefs[i][j] / lead\_elem;              }              free\_term[i] = free\_term[i] / lead\_elem;              ANNOTATE\_TASK\_END();              ANNOTATE\_TASK\_BEGIN(eliminate\_rows);              for (int j = i + 1; j < m; ++j)              {                  type coef = coefs[j][i] / coefs[i][i];                  for (int k = 0; k < n; ++k)                  {                      coefs[j][k] = coefs[j][k] - coefs[i][k] \* coef;                  }                  free\_term[j] = free\_term[j] - free\_term[i] \* coef;              }              ANNOTATE\_TASK\_END();          }          ANNOTATE\_SITE\_END();      } | void backward()      {          ANNOTATE\_SITE\_BEGIN(backward);          for (int i = m - 1; i >= 0; --i)          {              type lead\_elem = coefs[i][i];              ANNOTATE\_TASK\_BEGIN(normalize\_row\_back);              for (int j = n - 1; j >= 0; --j)              {                  coefs[i][j] = coefs[i][j] / lead\_elem;              }              free\_term[i] = free\_term[i] / lead\_elem;              ANNOTATE\_TASK\_END();              ANNOTATE\_TASK\_BEGIN(eliminate\_rows\_back);              for (int j = i - 1; j >= 0; --j)              {                  type coef = coefs[j][i] / coefs[i][i];                  for (int k = n - 1; k >= 0; --k)                  {                      coefs[j][k] = coefs[j][k] - coefs[i][k] \* coef;                  }                  free\_term[j] = free\_term[j] - free\_term[i] \* coef;              }              ANNOTATE\_TASK\_END();          }          ANNOTATE\_SITE\_END();      } |

Потенциальный прирост при введении параллелизма и с увеличением количества итераций в 5 раз, можно добиться прироста по скорости выполнение от 2 до практически 6 раз.

|  |  |
| --- | --- |
|  |  |

Задание 4.