**Отчет по лабораторной работе №2**

*Островского Артема ИВТ-11М*

**Задание 1.**

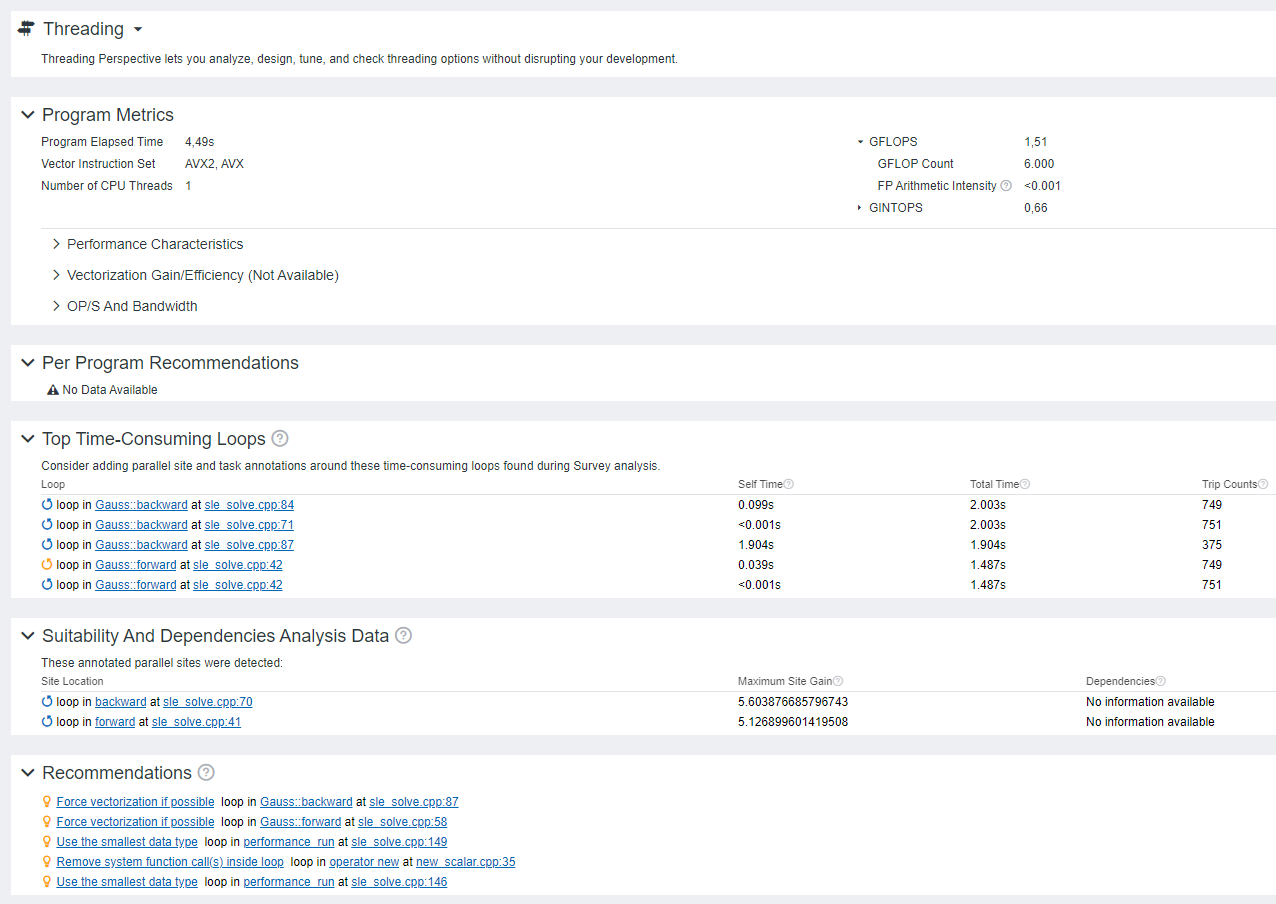
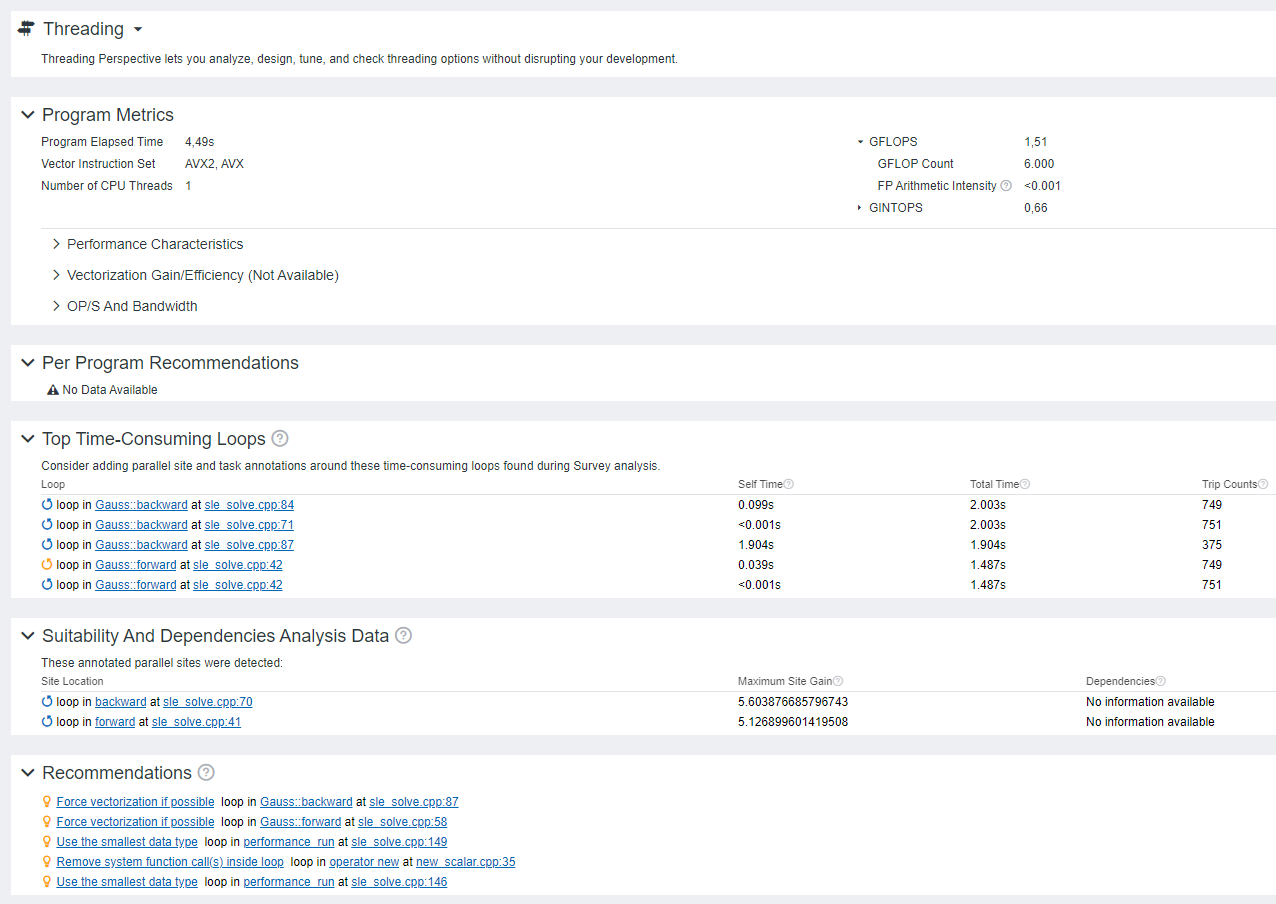
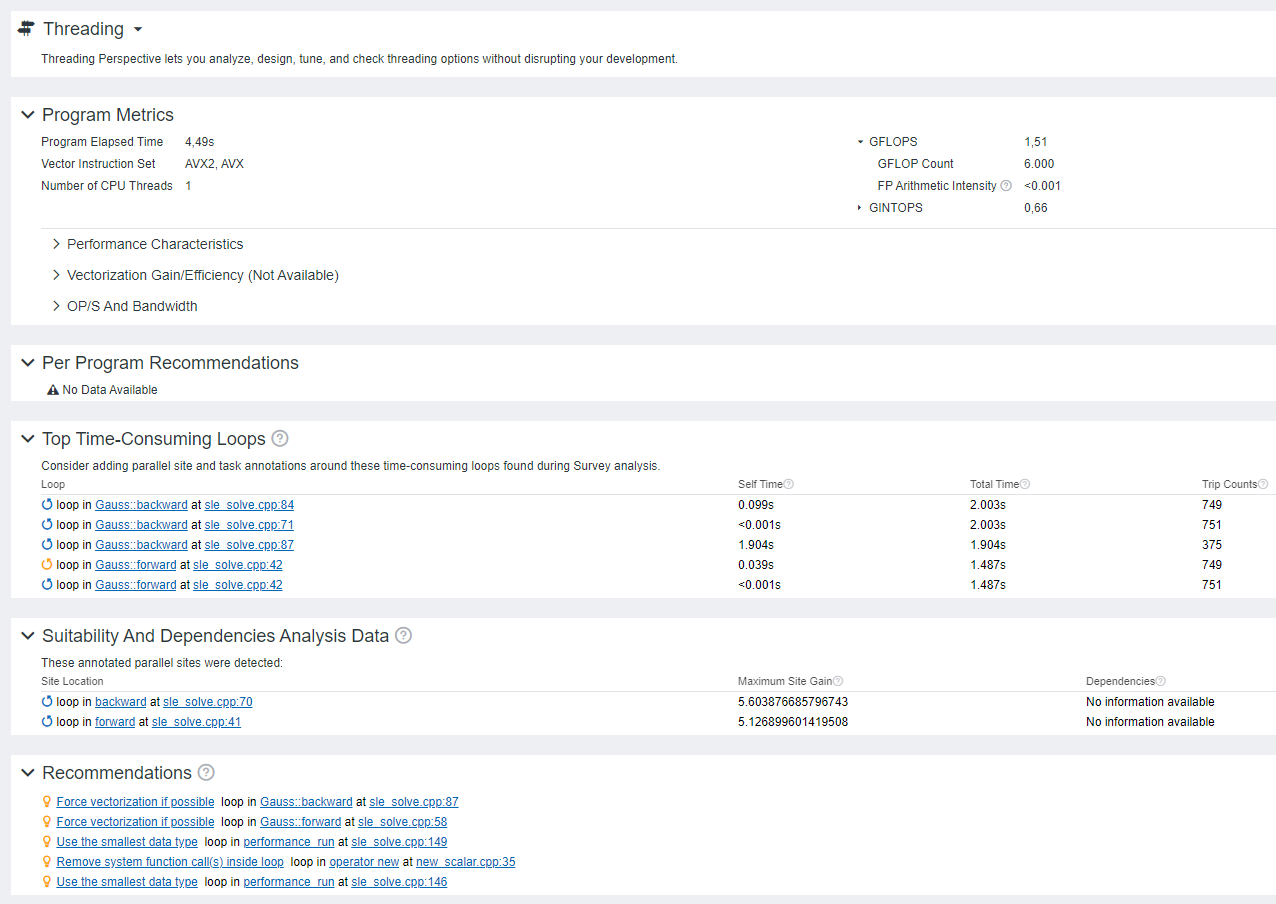
Необходимо собрать проект с исходным файлом sle\_solve.cpp и запустить собранный исполняемый файл. Оценить время работы программы и корректность ее работы. Зафиксируйте это в отчете.

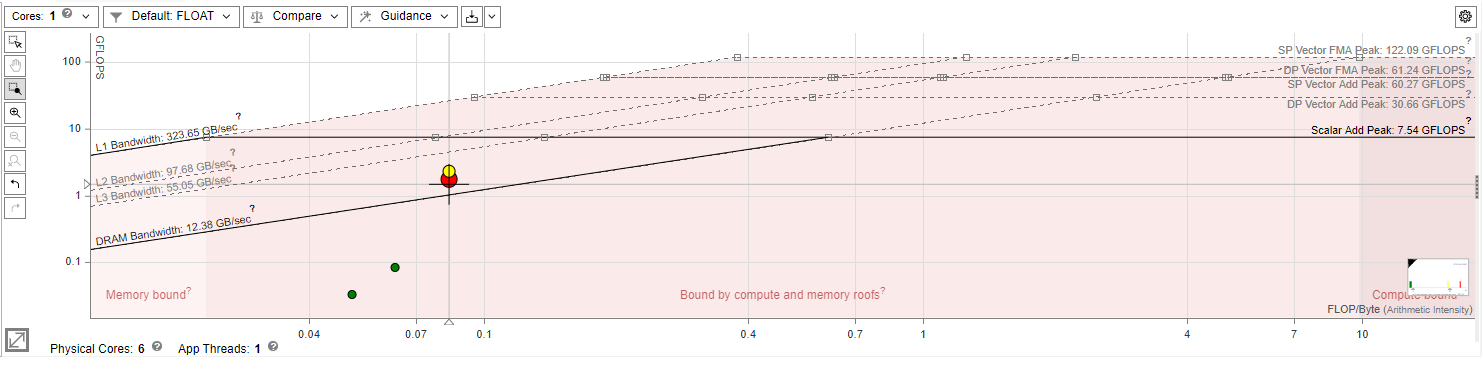
|  |  |  |
| --- | --- | --- |
| Elapsed sequential Gaussian time: 0.0440707 seconds  Elapsed sequential Gaussian time: 0.0489429 seconds  Elapsed sequential Gaussian time: 0.047664 seconds  Elapsed sequential Gaussian time: 0.0471572 seconds  Elapsed sequential Gaussian time: 0.0485878 seconds | Elapsed sequential Gaussian time: 0.408661 seconds  Elapsed sequential Gaussian time: 0.414637 seconds  Elapsed sequential Gaussian time: 0.423543 seconds  Elapsed sequential Gaussian time: 0.447739 seconds  Elapsed sequential Gaussian time: 0.428404 seconds | Elapsed sequential Gaussian time: 2.19916 seconds  Elapsed sequential Gaussian time: 2.26897 seconds  Elapsed sequential Gaussian time: 2.84567 seconds  Elapsed sequential Gaussian time: 2.53275 seconds  Elapsed sequential Gaussian time: 2.3177 seconds |

Количество итераций во 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.**

|  |  |
| --- | --- |
|  | Elapsed sequential Gaussian time: 0.0372814 seconds  Elapsed sequential Gaussian time: 0.0394884 seconds  Elapsed sequential Gaussian time: 0.0385043 seconds  Elapsed sequential Gaussian time: 0.0386812 seconds  Elapsed sequential Gaussian time: 0.0358365 seconds |
| Elapsed sequential Gaussian time: 0.264645 seconds  Elapsed sequential Gaussian time: 0.268775 seconds  Elapsed sequential Gaussian time: 0.255553 seconds  Elapsed sequential Gaussian time: 0.270667 seconds  Elapsed sequential Gaussian time: 0.267122 seconds |
| Elapsed sequential Gaussian time: 2.06753 seconds  Elapsed sequential Gaussian time: 1.92792 seconds  Elapsed sequential Gaussian time: 1.92066 seconds  Elapsed sequential Gaussian time: 2.57824 seconds  Elapsed sequential Gaussian time: 2.38607 seconds |

При изменении алгоритма выигрыш в производительности удаётся увеличить до 4-х раз.

|  |  |  |
| --- | --- | --- |
|  | Elapsed sequential Gaussian time: 0.0238677 seconds  Elapsed sequential Gaussian time: 0.0207765 seconds  Elapsed sequential Gaussian time: 0.0247634 seconds  Elapsed sequential Gaussian time: 0.0240692 seconds  Elapsed sequential Gaussian time: 0.0243284 seconds |  |
| Elapsed sequential Gaussian time: 0.116392 seconds  Elapsed sequential Gaussian time: 0.109253 seconds  Elapsed sequential Gaussian time: 0.112218 seconds  Elapsed sequential Gaussian time: 0.11313 seconds  Elapsed sequential Gaussian time: 0.126135 seconds |  |
| Elapsed sequential Gaussian time: 0.694425 seconds  Elapsed sequential Gaussian time: 0.73187 seconds  Elapsed sequential Gaussian time: 0.745461 seconds  Elapsed sequential Gaussian time: 0.788513 seconds  Elapsed sequential Gaussian time: 0.762283 seconds |  |

|  |
| --- |
| Elapsed sequential Gaussian time: 0.105266 seconds  Elapsed sequential Gaussian time: 0.12073 seconds  Elapsed sequential Gaussian time: 0.103154 seconds  Elapsed sequential Gaussian time: 0.128897 seconds  Elapsed sequential Gaussian time: 0.115669 seconds |