Лабораторна робота №3

# Дослідження оптимізації коду з використанням векторних розширень CPU

<https://github.com/comradeFreeman/csc.git> ( директорія lab3 )

sc.sh – bash cценарій, що здійснює обрахунок затраченого часу.

директорія **matriX** – більш наглядний приклад оптимізації

log1…log5 – файл результатів вимірювання часу обрахунку.

integral.cpp, matrix.cpp – вихідний код програми.

Integral, matrix – скомпільований файл програми.

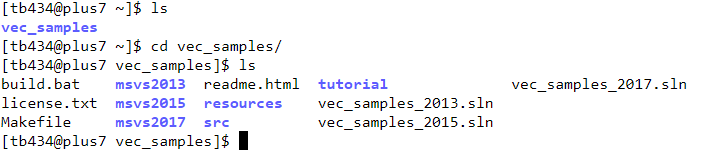
Multiply.optrpt.**\***, Driver.optrpt.**\*** - файли, сгенеровані в процесі оптимізації під час виконання туторіала.

## **Хід роботи**

### Отримайте доступ на обчислювальний кластер для роботи з Intel Compiler



### Завантажте файли Intel® C++ Compiler - Using Auto-Vectorization Tutorial (<https://software.intel.com/en-us/product-code-samples?topic=20813>) на свій комп’ютер та в домашню директорію користувача обчислювального кластеру.



### Використовуючи інструкції в readme.html ознайомтесь та виконайте Tutorial на обчислювальному кластері

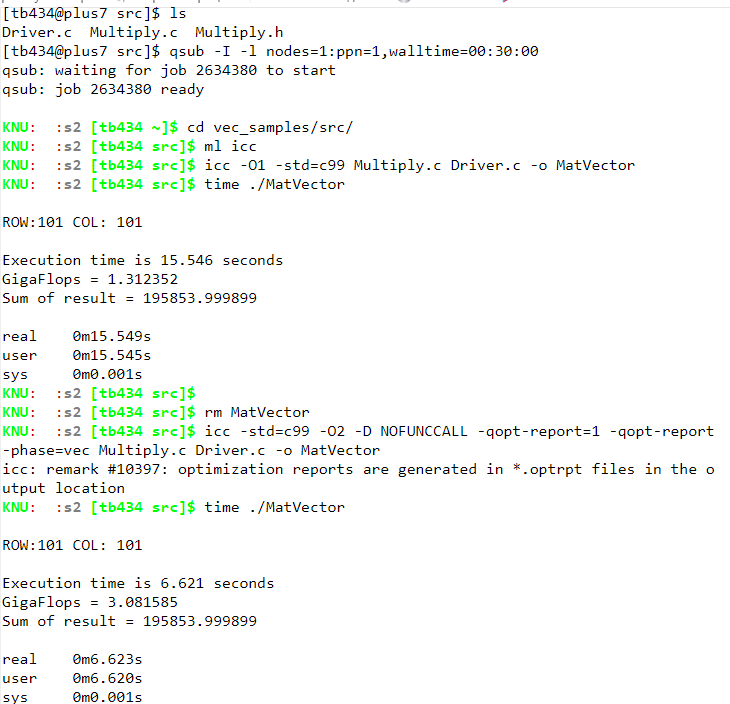
* 1. Замість інструкцій в пункті “Setting the Environment Variables” завантажте оточення компілятора шляхом виконання команди: ml icc

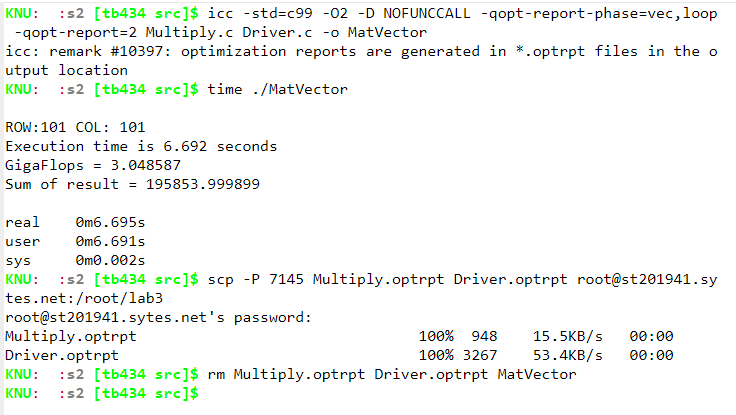


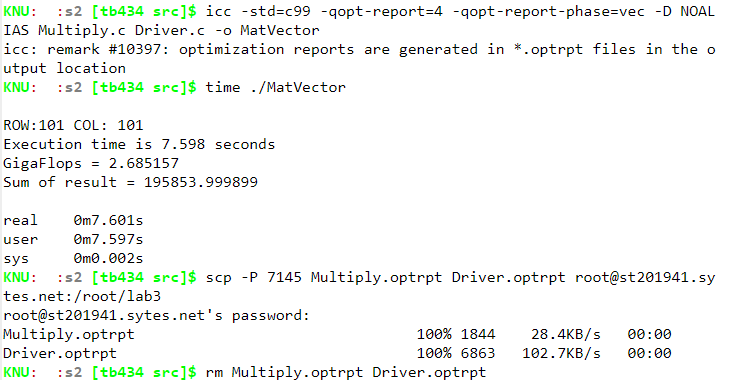
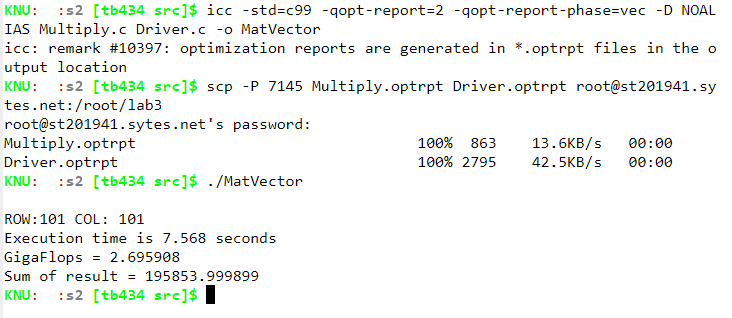
* 1. Виконуйте завдання на робочих вузлах кластеру замість вхідної ноди. По-перше процесори робочих вузлів мають набагато більше розширень. По-друге виконання компіляції та запуску на вхідній ноді заважає іншим користувачам, що призведе до **блокування вашого аккаунту та автоматичного незарахування лабораторної роботи**. Рекомендований варіант виконання роботи - використання інтерактивних задач в системі планування:

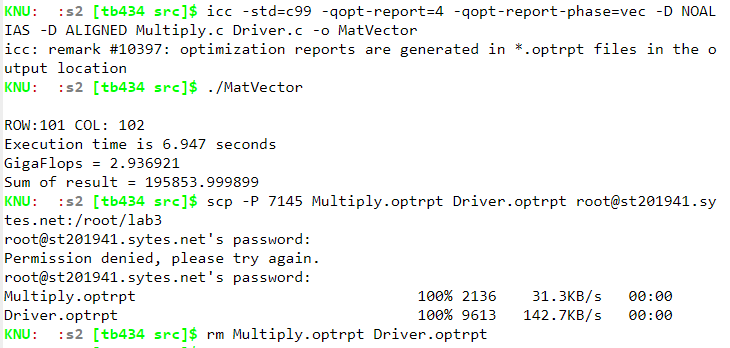
[manf@plus7 ~]$ qsub -I -l nodes=1:ppn=1,walltime=00:30:00

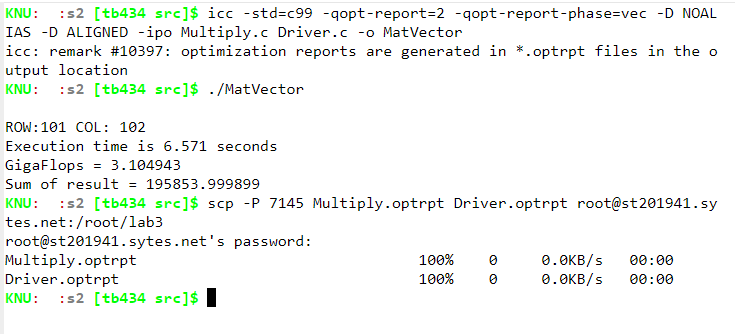
KNU:WN:s5 [manf ~]$ ml icc





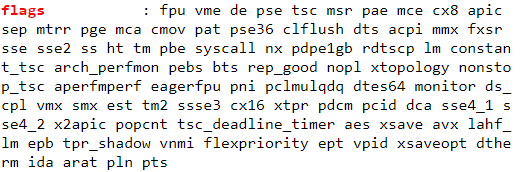






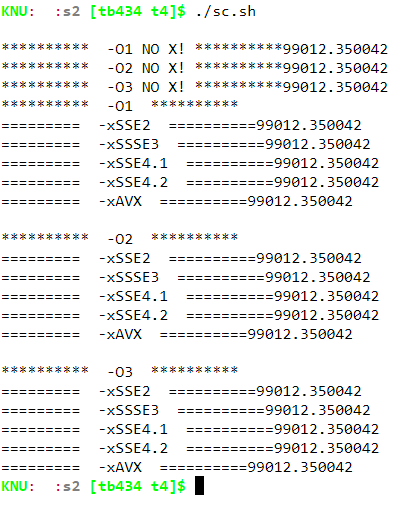
### Оберіть будь-яку неінтрерактивну консольну програму мовою С/C++ (унікальну в межах групи, в гуглі більше ніж 50 програм)

* 1. Напишіть сценарій, що:
     1. Компілює програму з різними оптимізаціями (-O) та виміряйте час її роботи. Якщо час досить малий - вимірюйте час роботи 1000 (чи 1000000) запусків алгоритму в циклі. Час роботи можна виміряти утилітою time.
     2. Отримує перелік всіх розширень процесору що підтримуються



* + 1. Для кожного розширення компілює Intel-компілятором окремий варіант оптимізованого коду (наприклад -x SSE2)
    2. Вимірює час виконання кожного варіанта оптимізованої програми
  1. Запустіть задачу **в планувальник** обчислювального кластеру 5 разів (для статистики на різних нодах)

[manf@plus7 ~]$ qsub -N MyJob -l nodes=1:ppn=1,walltime=00:30:00 script.sh



* 1. **Побудуйте графіки** залежності часу від різних варіантів компіляції.

## Виконання всіх попередніх пунктів оцінюється в 8 балів. Для отримання 10 балів виконайте наступне:

* 1. Встановіть програмний продукт Intel® Parallel Studio (<https://software.intel.com/en-us/intel-parallel-studio-xe/>)
  2. Використовуйте 30-day Tiral або отримайте студенську ліцензію, перейшовши за посиланням: <https://software.intel.com/en-us/qualify-for-free-software/student>
  3. Ознайомтесь з роботою за допомогою EVALUATION GUIDE: <https://software.intel.com/sites/default/files/managed/3d/a6/Boost_Performance.pdf> та інших джерел на сайті Intel (докуметація, відео-матеріал).
  4. Оберіть будь-який зі створених вами програмних продуктів та виконайте його оптимізацію з використання Intel® Parallel Studio.

**В звіті наведіть**:

* знімки екрану, що ілюструють виконання лабораторної роботи
* створений сценарій
* та результат його виконання,
* побудовані графіки часу виконання різних варіантів оптимізованої програми,
* а також процес та результат оптимізації вашої програми в Intel® Parallel Studio.