**236370 תכנות מקבילי ומבוזר לעיבוד נתונים ולמידת מכונה**

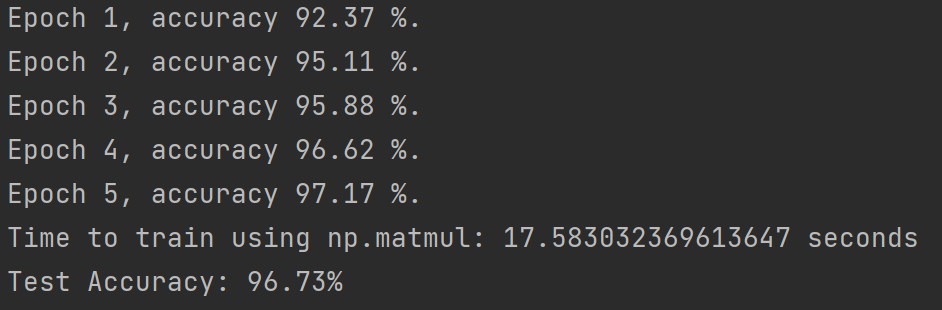
**דוח תרגיל בית 1**

מרינה ינובסקי 324515659

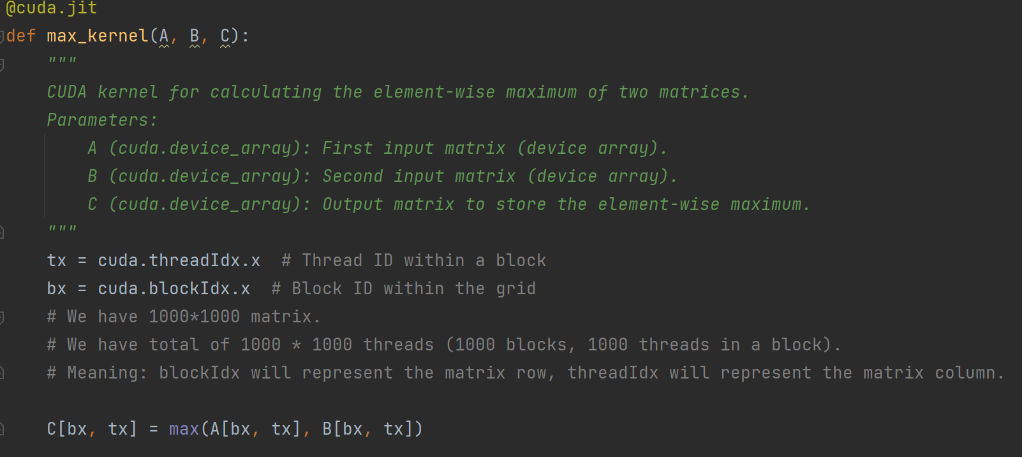
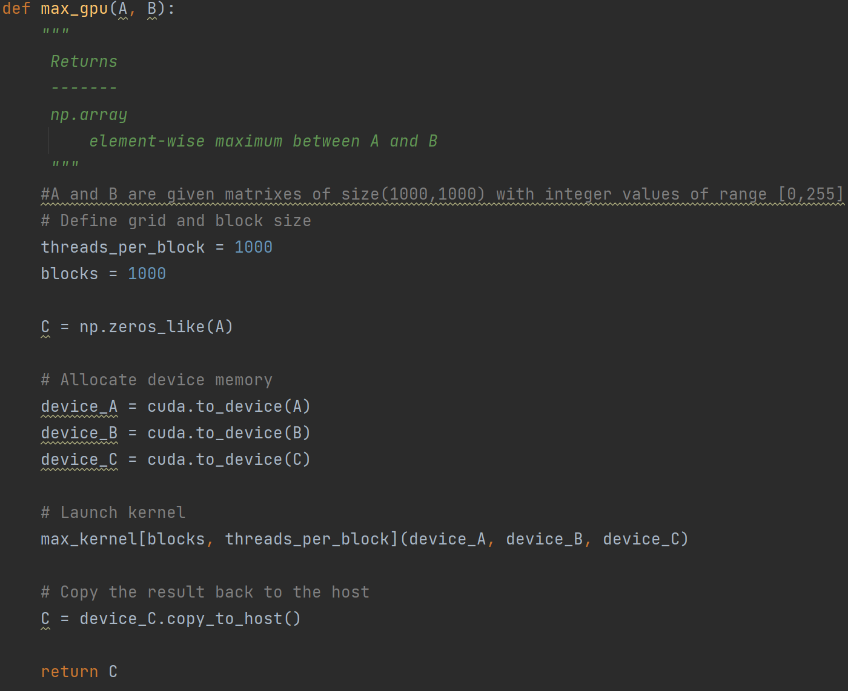
קורן מועברי 207987314

**חלק 1:**

תוצאות האימון, עם דיוק סופי גבוה מ95% כנדרש:



**חלק 2:**



מימוש הgpu\_max- הסבר:

נקצה מטריצת פלט.

נבצע העתקה של הפרמטרים הדרושים ל-device.

נקרא לקרנל: נפעילו עם 1000 בלוקים, כאשר בכל בלוק 1000 חוטים. הפרמטרים לקרנל אלו הפרמטרים שהעתקנו לזיכרון הdevice בשלב הקודם.

בסיום עבודת הקרנל, נעתיק את התוצאה שנכתבה לאחד הפרמטרים בזיכרון של הdevice חזרה לזיכרון הhost, כדי שהhost יכול לגשת אליה ולקרוא אותה.

מימוש הקרנל- הסבר:

נתון שגודל מטריצות הקלט הוא 1000\*1000. ישנם 1000 בלוקים, ובכל אחד מהם 1000 חוטים שמריצים את הקרנל.

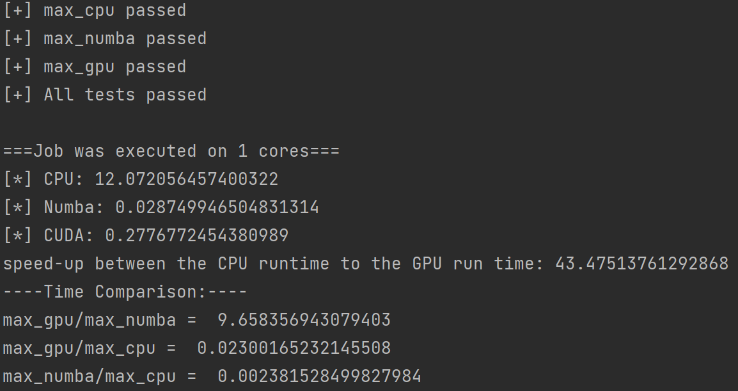
לכן ניתן לסתכל על זה בצורה הבאה: לחישוב של כל שורה, יהיה אחראי בלוק אחד. לחישוב תא בתוך שורה, יהיה אחראי חוט בבלוק. סך הכל- בלוק x בgrid מתאים לשורה הx, וחוט y בבלוק מתאים לעמודה הy, כלומר על ביצוע מקסימום בין אינדקסים x,y במטריצות B,A אחראי החוט הy בבלוק x.

נקבל את האינדקס של הבלוק ע"י הפקודה cuda.blockIdx.x (אנחנו משתמשים בגריד בעל מימד אחד ולכן מעניין אותנו הערך .x בלבד).

נקבל את האינדקס של החוט בבלוק ע"י הפקודה cuda.threadIdx.x (אנחנו משתמשים בבלוקים בעלי מימד אחד ולכן מעניין אותנו הערך .x בלבד).

סך הכל- נכתוב במטריצת הפלט במיקום הרצוי המקסימום בין האינדקסים המתאימים, לפי ההסברים לעיל.

הרצה על core אחד:



אכן ניתן לראות שאנו מקבלים האצה של פי 40 לפחות בין זמן הביצוע סדרתית על הCPU לזמן הביצוע מקבילית על הGPU.

מסופקים בצילום המסך כמו כן חישובים של הspeedup בין ההרצות השונות, כנדרש.

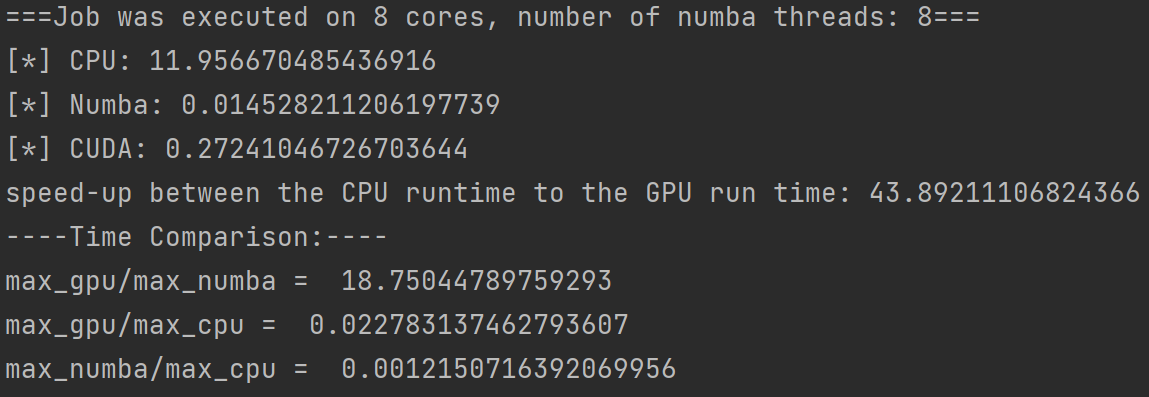
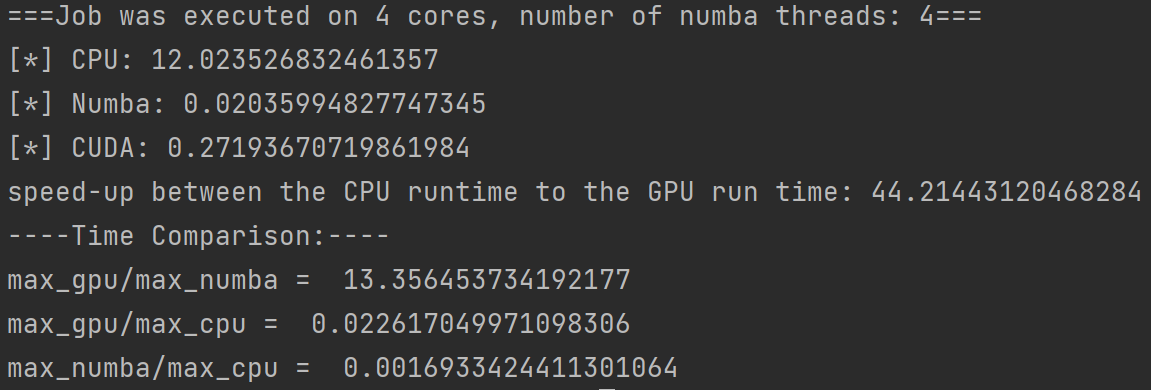
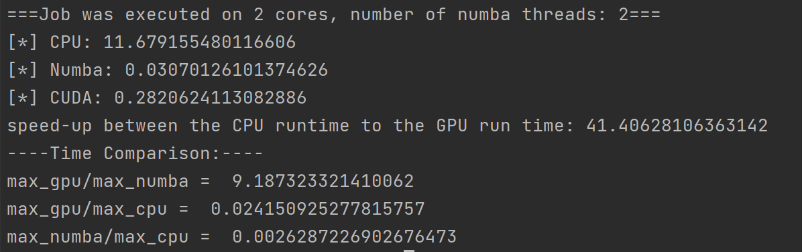
הסבר של התוצאות:

הCPU מבצע הרצה רגילה של קוד פייתון: כלומר כל שורה מתורגמת על ידי הinterpreter, ומורצת. במימוש הנאיבי שלנו, יהיו 1000\*1000 איטרציות סדרתיות, דבר שיקח זמן רב.

הGPU מבצע הרצה במקביל, הוא יבצע את הפעולה באופן מקבילי ולכן יתן תוצאות טובות יותר ביחס לCPU. התוכנית בCPU היא compute-bound, ומבצעת את אותה הפעולה הרבה פעמים. לכן, המקבול (בפרט מקבול שמאפשר ביצוע מספר כה רב של פעולות במקביל) נותן תוצאה טובה יותר מאשר הרצה על CPU.

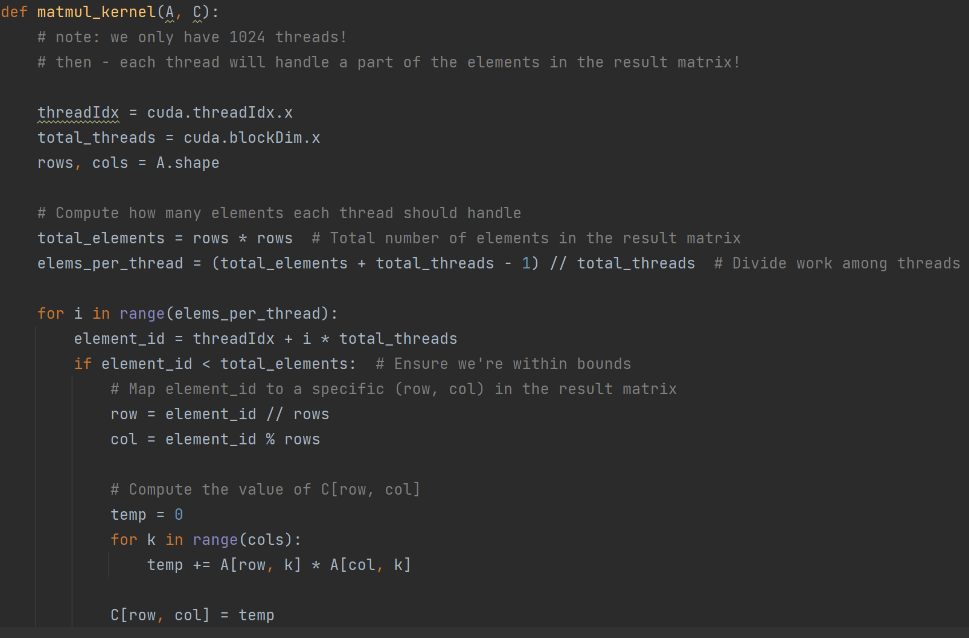
הרצה עם numba: מכיוון שהקוד עובר באיטרציה ה1 קמפול (JIT), בשאר האיטרציות הוא ירוץ מהר יותר בהשוואה לinterpreter ולהרצת הCPU הרגילה. בנוסף, ישנה תקורת העברת מידע לGPU וחזרה ממנו, ותקורת הניהול שלו, ולכן הרצה זו תהיה מהירה יותר גם כן בהשוואה לGPU.

הרצה על כמות משתנה של cores:



נשים לב כי ההבדל המהותי הוא עבור ההרצה לגרסה של הnumba. הסיבה להבדלים בהרצות אלו לכמות משתנה של cores היא שהקוד מתבצע בparallel loop, כלומר יש כמה תהליכים שמריצים את הקוד שבלולאה בצורה מקבילית, דבר שמספק שיפור בביצועים ככל שיש יותר cores שמאפשרים הרצה עליהם (מספקים יותר מקבול).

עבור גרסת הCPU התוכנית מבוצעת סדרתית, ועבור גרסת הCUDA התוכנית רצה על הGPU ולא מושפעת מהגדלת הליבות. לכן הגדלת כמות הcores לא משפיעה על מהלך הביצוע של התוכנית ברגסאות אלו ובפרט גם לא משפיעה על התוצאות שלהן.

**חלק 3:**

מימוש הקרנל matmul\_kernel:

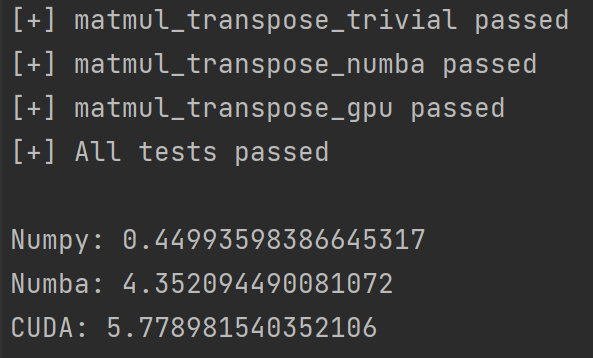
אין לנו הנחות כלשהן על גדלי המטריצות שאנחנו מכפילים, ולכן נרצה שכל thread יבצע כמות עבודה זהה (נרצה לחלק את הworkload באופן שוויוני). העבודה שתבוצע: הכפלת שורה בעמודה המתאימה לה במטריצה הtranspose.

תחילה נחשב את גודל הchunk עליו כל thread יבצע עבודה (גודל chunk הינו מספר האלמנטים שנחשב עבור מטריצת התוצאה). גודלו יחושב על ידי כמות האלמנטים הכוללת במטריצת התוצאה הרצויה (כמות איברי המטריצה) לחלק במספר הthreads.

כעת, בלולאה על גודל הchunk:

נחשב את המיקום היחסי עליו מעוניינים לבצע את העבודה. נוודא בפרט שהמיקום הנ"ל לא חורג מגבולות הבעיה (שכן תיתכן שארית לחישוב).

לאחר מכן, נבצע הכפלה של וקטור השורה בוקטור העמודה המתאים במטריצת הtranspose. הערך המתקבל הוא התוצאה הרצויה לאלמנט אחד בchunk. נשמור את התוצאה במטריצת הפלט, אליה יכתבו כל החוטים במקביל. נציין כי אין צורך בביצוע סנכרון, שכן אמנם כל החוטים כותבים לאותו האוביירט בזיכרון, אך האינדקסים אליהם החוטים כותבים הם ייחודיים.

ניתוח התוצאות:

לספריה numpy- להכפלת המטריצות הספריה משתמשת בקוד לו בוצעו אופטימיזציות רבות עבור הפעולה.

השוואה בין ביצועי numba לביצועי cuda-

* שניהם מריצים קוד בצורה מקבילית.
* Cuda מריץ קוד על GPU, ודורש העתקה של הקלט לdevice ושל הפלט מהdevice, דבר שדורש זמן ומשפיע על המדידות.
* הGPU לא מנוצל בצורה המיטבית ולכן השימוש בו לא משתלם:
  + גודל הבעיה הנבדקת לא מספיק גדול כדי לנצל את חומרת הGPU.
  + אנו מריצים את הקרנל רק עם בלוק אחד ו1024 חוטים, דבר שגורם להרבה cores של הGPU להיות במצב idle ולא מנוצלים. אם נרצה לשפר ביצועים- נצטרך להגדיל את סך החוטים הכולל שמריץ את הבעיה כדי לבצע יותר utilization לGPU.
  + תקורה גבוהה של העתקת המידע ביחס לרווח מההרצה המקבילית על הGPU.

לפיכך, הרצה מקבילית על הCPU שמספקת numba, בשילוב עם הקומפיילר של JIT, מספקים לנו ביצועים טובים יותר😊