שלום רב,

רציתי לסכם את המפגש השני.

מבחינת החוויה, אזי חייב לשתף שאני ממש מרגיש טוב וחלק סופר משמעותי מכך זה האווירה שיש בזכותכם תוך כדי המפגש.

תרגישו חופשי לפנות אליי כל אחד לכל מטרה שהוא מוצא לנכון.

מהמפגש הבא הפוקוס יעבור לעבודת Deployment המתבססת על תשתית NVIDIA בפרט על TensorRT.

כפי שנוכחנו לגלות כולנו כרגע המכונות הווירטואליות לא שמישות לעבודה כזו ואנו נעשה את מירב המאמצים לסדר זאת עד למפגש הבא.

כל הירתמות שלכם לתת יד תתקבל בברכה.

אני למשל מאתמול בערב מנסה להשמיש באמצעות Google colab את היכולת וראינו שאכן לעבודה מעל CUDA  בלבד באמצעות NUMBA זה אכן עובד היטב אך נוכחנו לגלות שכרגע לא מותקנת שם היכולת לעבוד עם TRT.

מהבוקר אני מנסה להשמיש התקנה כזו כי בסה"כ זה התקנה של Deb packages של שתי ספריות CUDNN & TRT.  
גם מחיפוש זריז שביצעתי ברשת נראה שעוד אנשים ניסו להשמיש יכולת זו.

אני ביצעתי התקנה של TRT deb package קינפגתי את המשתנה LD\_LIBRARY\_PATH  כדי שימצא את קבצי ה-so על ה-TRT אך עדיין פעולת ה-import tensorrt as trt מתריעה על כך שהיא לא מוצאת קבצים אלו (למרות שכשאני עושה זאת על מכונת לינוקס לוקאלית שלי זה מיד עובד.

כמו שאתם מבינים כל מי שמכיר Linux וימצא לנכון להצטרף למאמץ הקבוצתי יעזור לנו מאוד.

במידה ועד למפגש הבא לא נצליח בדרך כלשהי לאפשר גישה למכונה וירטואלית נצטרך לאלתר יחד כיצד כל אחת ואחד מאיתנו  מתגבר על האי נוחות הפרטית שלו ומצטרף יחד איתנו למאמץ הקבוצתי כדי למצוא דרך לאפשר גישה כזו.

מתנצל שוב מראש על הסיטואציה אני בטוח שאתם עדים לכך שאני עושים הכל כדי למצוא את האיזון הטוב ביותר לגשר על הפערים בין הרצוי למצוי.

זמין לכל שאלה, עצה, רצון לנסות יחד משהו....

המשך שבוע נעים...

לגופו של עניין,

סיכום טכני:

1. מצרף את שתי המצגות שהוצגו היום.

2. מצרף את התרגולים ופתרונות עליהם עבדנו יחד במפגש.  
כל הקבצי \*.py בקובץ Numba\_Playground.zip שונו ל-\*.py.txt בגלל בעיות סייבר....אנא שנו חזרה כדי שתוכלו להשתמש בהם.

2. נשאלתי שאלות מעניינות שאני חושב שכדאי שכולם ייחשפו אליהן ולתשובות

השאלות היו:

1. מה עושה (#)cuda.grid ולמה צריך אותו?

2. ראיתי שאתה מריץ את הפילטרים למשל ב לולאות. לא אמורים לכתוב ל cuda קוד ללא לולאות?

3. דיברת על בראנצ'ים. מה זה בראנצ'? למה הוא דופק את cuda? ואיך נמנעים ממנו?

4. אני מתכנת ב c++ , יש לך מקבילות של הקוד בפייתון לcpp? הסברת איך מתקינים את cuda ב visual studio?

להלן התשובות המלאות:

1. מעטפת ה-cuda.grid באה לעזור לך ולחסוך לעבוד ישירות עם המשתנים cuda.threadIdx.  
למשל אם אתה הגרדת grid דו ממדי אזי כדי להבין מה האינדקס הדו ממדי של ה-cuda thread שלך, תצטרך לפנות או באמצעות cuda.threadIdx.x & cuda.threadIdx.y או שפשוט תפנה ל-cuda.grid(2) ומיד תקבל tuple שמחזיק את האינדקס (x, y) של ה-cuda thread.

בגדול רק מעטפת לאינדקסים של ה-cuda thread.

2.בגדול זה נכון.  
כל CUDA kernel בהתאם לממד ה-grid שהגדרת מחליף בעצם לולאה.  
למשל הגדרת grid חד ממדי זה אומר שבעצם אתה יכול ליישם במקביל מה שהיית עושה על CPU עם לולאה אחת.

אם הגדרת grid עם שני ממדים, בעצם אתה מקבל מה שקודם היית מיישם באמצעות לולאה מקוננת.

אבל מה אתה עושה אם הלוגיקה שלך בתוך הלולאה המקוננת היא בעצמה מורכבת מעוד לולאות מקוננות?

אזי אין לך ברירה, לעיתים גם CUDA thread צריך ליישם לוגיקה מבוססת לולאות.

זה מצב קלאסי לפילטר.

ה-gird הדו ממדי מחליף את שני הלולאות הראשיות איתן היית אמור לטייל על כל הפיקסלים ואז כשאתה מגיע לפיקסל המיועד אתה צריך עוד שתי לולאות מקוננות שמבצעות את הפילטר לפני השכנים של אותו הפיקסל.

לכן גם ב-CUDA יש לולאות.

3. להימנע מ-Branch זה בעצם להימנע מקוד שהוא לא Sequential כלומר יש לו הסתעפויות.

זה יכול לנבוע משימוש ב-if  else למשל או גם לולאה שהרי בעצם יש לה תנאי בדיקה לעצירתה.  
זו הסיבה שגם לא מומלץ להשתמש בלולאה ב-CUDA, אך כפי שהסברתי קודם לעיתים אין ברירה.  
  
בעבר ה-GPUs לא היו מספקים חזקים להתמודד עם קוד כזה ולכן ההמלצה הייתה מאוד חזקה להימנע מכל שימוש ולכן הכניסו את ההנחיות לקומפיילר כגון #pragma unroll שהייתה אמורה לפרוס קוד שמקודד בלולאה וכך להימנע מ-Branches.

היום ה-GPUs מאוד חזקים ומתמודדים יפה מאוד גם עם Branches אבל עדיין תמיד נכון שאם יש לך אפשרות הגיונית להימנע מהם זה עדיף.

הסיבה קשורה בכך שתזמון CUDA threads ל-CUDA cores פידיים יכול "להיפגע" כאשר cuda threads נתקלים ב-Branch.

התזמון של CUDA threads נעשה לפי יחידה לוגית שנקראת Warp שזו קבוצה המאגדת 32 CUDA threads וקבוצה כזו לא מתחלפת עד אשר כל ה-CUDA threads שלה סיימו את עבודתם.  
אם יש Branch כזה שגורם לכך שחלק מה-CUDA threads יבצעו את מסלול ה-if וחלק את מסלול ה-else זה מייצר Divergence ב-Flow דבר שמאט את כלל עבודת ה-Warp.

4. גם אני מפתח במבצעי שלי רק ב-Cpp.  
כל מה שאני מציג ב-Python זה רק בגלל שקל יותר ללמד עליו.  
המייל סיכום של מפגש קודם יש את כל הנחיות הנדרשות לשם התקנת CUDA Tlk ומעבר לזה גם ההנחיות להשמשת כל ה-TensorRT.  
תעבור עליו, אם יהיה חסר לך משהו צור איתי קשר.

בברכה