חישוב מקבילי ומבוזר

פרויקט סיום הקורס

שם הסטודנט	ת.ז הסטודנט
אבו גמל מעאד	_
אדקידק חאלד	_

❖ שיטת המקבול:

את הארכיטקטורה שבחרנו בה כדי למקבל את החישובים היא ארכיטקטורת ה- Master וה- Worker, כך שהעבודה של ה- Master היא לקרוא את הנתונים מקובץ ה- input ולשלוח אותם ל- Worker שמבצע בתורו את החיפושים והחישובים בצורה מקבילית שמנצלת את כל משובים המחשב על ידי שימוש במנגנוני ה- CUDA Multi Streaming וה- OpenMP Tasks, כאשר ה- Worker מסיים את העבודה הוא מחזיר את התוצאות ל- Master ואז ה- Master יכתוב אותן בקובץ ה- output.

שלבי העבודה והמקבול בצורה מפורטת:

1- מתחיל ה- Master בלקרוא את האובייקטים מקובץ ה- input ולאחר שהוא קורא אובייקט הוא שולח אותו ל- Worker. קבלת האובייקטים בצד ה- Worker מתבצעת ב- Parallel Region על ידי Single Thread שמקבל את האובייקט ומאחסן אותו ב- Object Struct, ואז הוא מייצר Task כדי להעתיק את האובייקט ל- GPU בצורה מקבילית על ידי אחד ה- Threads של ה- Parallel Region, ההעתקה ל- GPU מתבצעת על ידי שימוש במנגנון ה- CUDA Multi Streaming, כך שה- Thread שמבצע את ההעתקה ממיצר Stream משלו והוא גם מחכה להעתקה שתסתיים על ידי השימוש בפונקציית ה- cudaStreamSynchronize. לאחר קבלת כל האובייקטים ה- Single Thread מבצע taskwait כדי לוודא שהעתקת האובייקטים ל- GPU הסתיימה.

2- לאחר שהסתתים תהליך העברת האובייקטים ולאחר שה- Worker העתיק אותם ל- GPU הוא מתחיל לקבל את התמונות מה- Master עם אותו הרעיון שהזכרתי למעלה. קבלת התמונות בצד ה- Worker מתבצעת ב- Parallel Region על ידי Single Thread שמקבל את התמונה ומאחסן אותו ב- Picture Struct, ואז הוא מייצר Task כדי לבצע את החיפוש על האובייקטים בתמונה בצורה מקבילית על ה- GPU על ידי אחד ה- Threads של ה- Parallel Region, החישובים והחיפוש על האובייקטים בתמונה ב- GPU מתבצעת על ידי שימוש במנגנון ה- CUDA Multi Streaming, כך שה- Thread שמבצע את החיפוש מיצר Stream משל.

את החישובים והחיפוש על האובייקטים ב- GPU ממוקבלים בצורה הבאה:

את ה- Thread שמבצע את ה- Task מעתיק את התמונה ל- GPU ואז הוא מתחיל לעבור בלולאה על האובייקטים ולכל אובייקט הוא קורא לפונקציית GPU Kernel שבודקת בצורה מקבילית את כל המקומיים (Positions) האפשריים שיכול להופיע בהם את האובייקט כך שכל מקום אפשרי נבדק על ידי GPU Thread אחד, זאת אמרת שמספר ה- GPU Threads שאנחנו צריכים לחיפוש על האובייקט בתמונה הוא:

$$\text{Working Area Dimension} = \text{Picture Dimension} - \text{Object Dimension} + 1$$

את ה- GPU Threads אנחנו ממקמים או מארגנים בתוך Grid שמכיל בלוקים בגודל 32 x 32 כדי לנצל את כל הגודל של ה- Warp, אז אם אנחנו רוצים לדמיין את זה יש מעל כל מיקום שאפשר להתחיל ממנו בלחפש על האובייקט GPU Thread שאחראי לבצע את חישובי ההתאמה.

אם האובייקט נמצא בתמונה אנחנו שומרים את המופע הראשון שלו בתמונה בתוצאת החישוב לתמונה, לאחר שה- CPU Thread מוצא שלושה אובייקטים בתמונה הוא מסיים את ה- Task ומתחיל לעבוד על Task חדש אם יש אחד. כמובן ה- CPU Thread מבצע את כל הפעולות שצריך לאחר שהוא מסיים את כל שלב משלבי העבודה כמו: free CPU/GPU, cudaStreamSynchronize, streamDestory and others.

לאחר קבלת כל התמונות ה- Single Thread שייצר את ה- Tasks כדי לבצע את החיפושים בתמונות מבצע taskwait כדי לוודא שהחיפושים על ה- GPU הסתיימו.

3- לאחר שסיים ה- Worker את העבודה על התמונות הוא שולח את התוצאות ל- Master ועושה את כל ה- frees שנדרשים ומסתיים, ה- Master כותב את התוצאות בקובץ ה- output ומסתיים.

❖ צורת העברת הנתונים:

כדי להעביר את הנתונים בין ה- Master וה- Worker במקום לייצר סוג משתנה חדש (Struct) ב- MPI ולהתחיל להתעסק ולתפל בהעברת ה- "Pointers" והצורך לכמה שליחות כדי לסיים העברת הנתונים לאובייקט או תמונה אחת החלטנו לבנות Formats משלנו כדי להעביר את הנתונים שכל לפענח אותם בשתי הצדדים. את ה- Formats האלה הם רצפי Bytes כמו מערך והם בנויים בצורה הבאה:

1- את הנתונים של האובייקטים אנחנו מעבירים בצורה הבאב:

Object ID (int)	Object Dimension (int)	Object Matrix (Sequence of int)

2- את הנתונים של התמונה אנחנו מעבירים בצורה הבאב:

Picture ID (int)	Picture Dimension (int)	Picture Matrix (Sequence of int)

3- את תוצאת החיפוש בתמונה אנחנו מעבירים בצורה הבאב:

Found Object Info = < Object ID (int), X (int), Y (int) >

Picture ID (int)	Found Objects Num (int)	Found Object Info * Found Objects Num (Sequence of int)

❖ תוצאות ההרצה:

בדקנו את ההבדל בזמנים בין הפתרון הסדרתי (Sequential) לבין הפרטון המקבילי (Parallel) בעזרת הקובץ שהעלה המרצה לאתר.

Solution Type	Running Time (sec)
Sequential	9.090775e+00
Parallel	1.166644e-01

הפתרון המקבילי (Parallel) הוא יותר מהר פי 77.922 פעמים מהפתרון הסדרתי (Sequential).

❖ שיטת הקמבול, ההרצה והניקוי:

To Compile: make build

To Run: make run arg1=<input-file-path> arg2=<output-file-path>

To Clean: make clean