CDP תרגיל בית 2, OpenMP

מגישים :

יוסף רייסמן	315835355	yosef@campus
גל יהודה	206016834	gal2016@gmail

תיאור הפיתרון:

מימוש simple_parallel_walsh

השתמשנו ב-omp parallel for כדי להתאים לכל חוט תא מתוך הווקטור שעליו לחשב. החישוב בחוט נעשה בעזרת כפל מטריצות רגיל כאשר הפונקציה hamming_weight() שימשה לחישוב האיבר הנחוץ בעמודת המטריצה (מימושה בהתאם לאשכול ב-StackOverflow).

מימוש fast_parallel_walsh

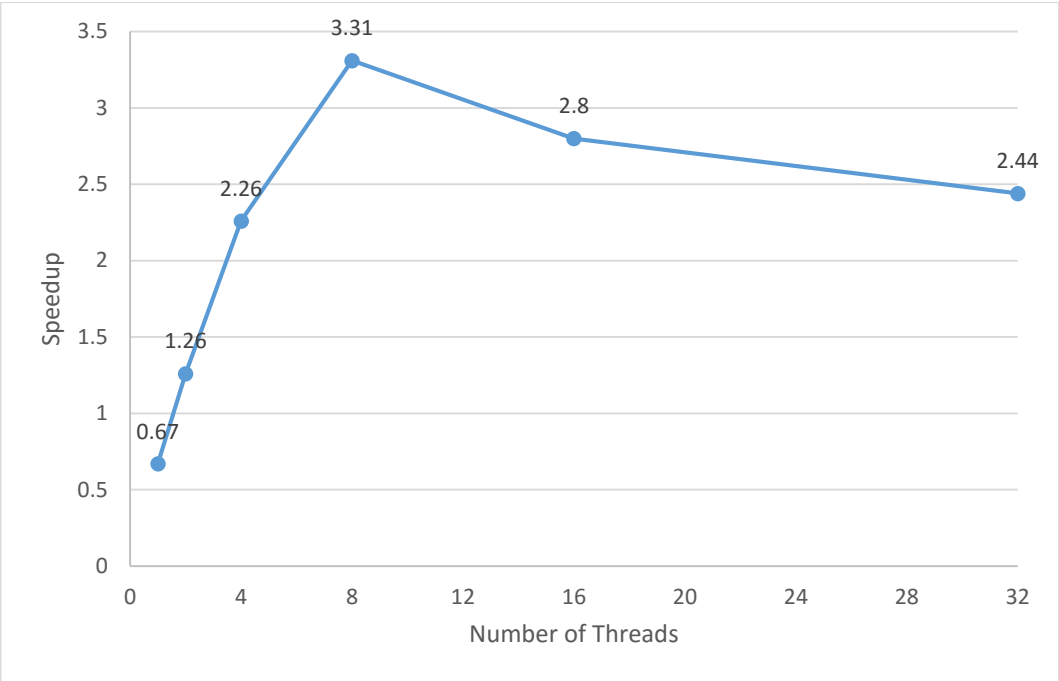
השתמשנו במספר החוטים (omp_get_num_threads())), כדי להגיע למספר צעדי הפיתוח הנחוצים לפני מעבר לפיתרון רקורסיבי בחוטים לא תלויים. בלולאה מסונכרנת באופן מקבילי, פיתחנו בכל חוט את החתך שלו (בגודל parallel_size) צעד יחיד, ובכל צעד שמרנו את תוצר החוטים אל הווקטור המקורי. כאשר סיימנו לקדם את הווקטור בצעדים הנחוצים, השתמשנו בפונקציה recursive_fft() שמממשת רקורסיבית את האלגוריתם תוך הנחה שאין השפעה הדדית בין החוטים (כלומר, הם יכולים לרדת לעומק המימוש בלי להמתין לחוטים האחרים), וקראנו לה מכל חוט על הבלוק המתאים לו.

להלן תוצאות הרצת המימוש המהיר על השרת:

פקודה	OMP_NUM_THREADS	Time (msec)	Speedup
./hw2 16777216 2	1 (base)	2358	1
	2	1269	1.86
זמן סריאלי (msec):	4	709	3.33
~1600	8	474	4.97
	16	557	4.23

חלק יבש:

שאלה 1:



א. הגרף מתאים לחוק אמדל, ומדגים כיצד האצת התוכנית משתפרת כאשר נעשה שימוש במספר מעבדים גדול יותר ($\text{Speedup}(\#\text{CPUs}) = \frac{1}{\left(\frac{A}{\#\text{CPUs}}+(1-A)\right)}$). ניתן לראות בגרף שההאצה בריצה (לעומת הפתרון הסריאלי הנתון) טובה יותר, ככל שמוקצים לתוכנית יותר חוטים (נסביר למה זה לא נכון לגמרי בסעיפים הבאים).

ב. עם זאת, ההאצה שאנחנו רואים לא עולה ליניארית לכמות החוטים, שכן החלק הסדרתי תמיד רץ באותו מחיר זמן, ללא תלות בשיפור שמאפשרת המקביליות. כלומר רק חלק מהתוכנית משתפר, כך שהמשקל היחסי של החלק הסדרתי הולך ועולה. בנוסף, חלק מהירידה ניתן לייחס לענייני סנכרון בין החוטים, שנעשה יקר יותר ככל שנוצרים יותר חוטים (נדרש סנכרון בחלק הראשון של הפתרון המקבילי, שכן לא ניתן להתקדם דור כל עוד שאר החוטים לא סיימו להעביר את המידע שלהם אל new_vector).

ג. בניסוי הנוכחי 16 ו-32 חוטים הציגו ביצועים טובים פחות מ-8 חוטים. להלן שלוש סיבות שמסבירות פער זה:

a. כפי שכתבנו בסעיף הקודם, בלולאה המקבילית הראשונה ב-fast_walsh, החוטים לא יכולים לעבור לאיטרציה הבאה עד שכל שאר החוטים מסיימים את האיטרציה הנוכחית, שכן קיימות תלויות מידע בין החתכים השונים של הווקטור המוקצים להם. המתנה זו ממומשת ב-barrier בשורה 82. כאמור, ככל שמספר החוטים גבוה יותר, עולה הסיכוי שחוט יצטרך להמתין למשנהו לפני שיוכלו להתקדם לצעד הפיתוח הבא.

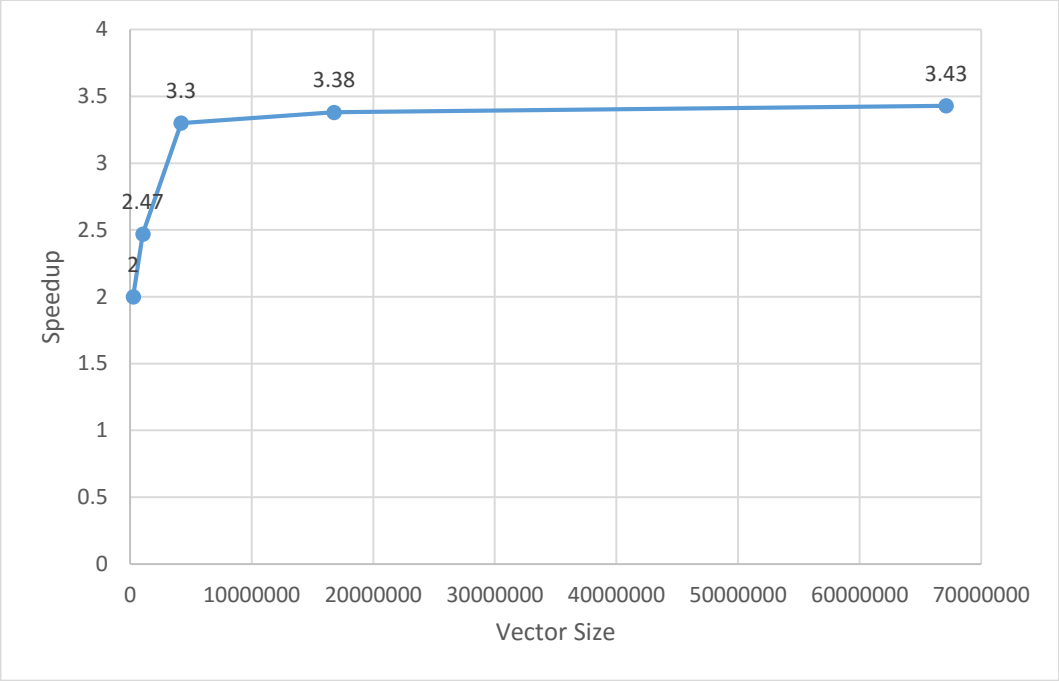
b. התכנית מורצת על שרת שמספק לה (לפי ה-Technical Details) לכל היותר 10 מעבדים בו זמנית, מה שאומר שתחת הנחת אופטימליות על מנגנון OpenMP החוטים יזומנו למעבדים שונים וירוצו במקביליות מלאה כאשר מספרם קטן מ-10. כאשר מספר

החוטים גבוה יותר, יש משמעות גבוהה למדיניות ה-scheduling של מערכת ההפעלה שכן לא כל התהליכים יכולים לפעול יחד וצריך להחליף ביניהם (מה שמייצר overhead).

c. ציינו את עניין ההמתנה לחוטים שכנים כדי לדאוג לנכונות התוכנה (כלומר, לא לאפשר לחוטים לחקור צעדים מתקדמים יותר לפני שהמידע הרלבנטי להם מגיע מהחוטים האחרים). מחיר נוסף נשלם בעניין דומה: ככל שנשתמש ביותר חוטים הגישות לזכרון המקביליות ידרשו התערבות רבה יותר כדי לוודא שלא מתבצעת דריסת מידע, לעדכן מטמונים שונים, או להוסיף למטמון נתונים שנמחקו ממנו בשל ריבוי הגישות (אם מידע אחר תפס את מקומם).

הירידה במימוש שלנו אינה קריסה של ממש, וההאצה נשארת טובה בשני המקרים מההאצה של שימוש ב-4 חוטים. בניסוי הנוכחי ההבדל עשוי להיות לא משמעותי שכן החוטים מבצעים עבודה דומה וכך זמני ההמתנה שלהם אינם גבוהים במערכת תיזמון שוויונית. כמו כן, המימוש שלנו דואג לכך שכאשר אין צורך בסנכרון המידע בין החוטים, החוטים יוכלו להתקדם ללא תלות זה בזה, ובכך מצטמצם זמן ההמתנה שלהם.

<u>שאלה 2:</u>



א. הגרף מתאים ל-Gustafson's law, בכך שהוא מראה שגודל הקלט קשור למידת ההאצה של התוכנית המקבילית: ככל שעולה גודל הקלט, התוכנית רצה מהר יותר יחסית לתוכנית הסדרתית שסופקה. $\text{Speedup} \leq \#\text{CPUs} - (1 - A) \cdot (\#\text{CPUs} - 1)$; נראה שההאצה יחסית לגודל החלק הסריאלי - חלק סדרתי קטן יותר מעלה את הרף לתאוצה תחת מספר מעבדים קבוע. לכן בתוכנית כמו שלנו בה גודל החלק הסריאלי קבוע וגודל החלק המקבילי עולה עם גודל הקלט, החלק הסריאלי פחות משמעותי לזמן הסופי של התוכנית ובכך למידת ההאצה.

ב. השינוי בהאצה לפי גודל הקלט תלוי ביחס בין החלק הסדרתי (1-A) לבין החלק המקבילי; אם החלק הסדרתי קטן יחסית לחלק המקבילי נקבל האצה בתוכנית (כפי שאפשר לראות בניסוי הנוכחי), אחרת נקבל האטה (שכן כמות הקוד המקבילי תלך ותקטן יחסית לגדילת הקלט).

שאלה 3 :

א. נשערך את החלק המקבילי להיות סדר הגודל של הפעולות שיובצעו בריצת מקבילית לחלוטין חלקי כמות הפעולות המוערכת לתוכנית כולה. הגענו לסדר גודל זה באמצעות התבוננות : כמות הקוד שמתבצעת בלולאה העיקרית ובהמשכה הרקורסיבי היא ממוקבלת לחלוטין מבחינתנו כמתכנתים. החלק שאין בו מקבול הוא לולאת ה- log שעולה עבור קלט בגודל 2^x, x פעולות, וכן הקצאת הזיכרון ושחרורו שעולים לכל היותר 2^x. החלק המקבילי עובר על כל תא בווקטור (log2(num_threads פעמים, ולאחר מכן מספר הפעמים הנוסף שדרוש להשלמת החישוב (ידוע לנו שמדובר סה״כ ב-x פעמים. זהו החלק אליו הגענו, אם כן :

$$\frac{2^x \cdot x}{2^x(x+1)+x}$$

נציב *x=18,22,26*

$$A(2^{18}) \approx 0.9474$$

$$A(2^{22}) \approx 0.9565$$

$$A(2^{26}) \approx 0.963$$

ב. חוק אמדל נותן את הנוסחה הבאה ל-Speedup :

$$Speedup(\#CPUs) = \frac{1}{\left(\frac{A}{\#CPUs} + (1-A)\right)} = \frac{T_s}{T_p}$$

נשתמש בקלט בגודל 2^{18} ונגיע לחישוב החלק המקבילי בתוכנית לפי נוסחה זו. מהירות התוכנית עבור חוט אחד : 29msec, 2 חוטים : 17msec, 4 חוטים : 11msec, 8 חוטים : 9msec.

$$Speedup(2) = \frac{29}{17} = \frac{1}{\frac{A}{2} + (1-A)} \Longrightarrow A = \frac{24}{29} \approx 0.83$$

$$Speedup(4) = \frac{29}{11} = \frac{1}{\frac{A}{4} + (1-A)} \Longrightarrow A = \frac{24}{29} \approx 0.83$$

$$Speedup(8) = \frac{29}{9} = \frac{1}{\frac{A}{8} + (1-A)} \Longrightarrow A = \frac{160}{203} \approx 0.79$$

התוצאות עקביות בשתי ההרצות הראשונות, אך לא בשלישית. חוסר העקביות נובעת בין השאר משימוש בנתון לא מדויק בתור זמן ריצת התוכנית לחוט אחד (החישובים המתבצעים שונים בין הגבלה לחוט אחד ובין גרסה סדרתית ממש), וכן בשינויים שלא תלויים במימוש אלא בסביבת ההרצה (הזמן המדווח למשתמש). כמו כן, כפי שנכתב בהערה, החוקים של אמדל מהווים מגבלה עליונה אך לא מדויקת.

ג. ככל הנראה הערכים שהתקבלו בסעיפים הקודמים אינם מהווים מדד מדויק למידת המקביליות של הקוד שלנו, שכן הם אינם עקביים. כמו כן, אין מקבילות מוחלטת בתוכנה (אפילו שהקוד כביכול, בתיאוריה, מקבילי לחלוטין) שכן מימושה מוסדר על ידי OpenMP ומערכת ההפעלה שמייצרות תקורה.

שאלה 4:

א. ידוע כי האצה בתלות בגודל הליבה היא $Speedup = \frac{1}{\frac{1-A}{perf(r)}+\frac{A\cdot r}{perf(r)\cdot n}}$

נציב בביטוי את הנתונים ונגזור לפי r, נשווה ל-0:

$$A = 0.9, n = 32, per\ f(r) = \sqrt{r}$$

$$Speedup = \frac{1}{0.028125\sqrt{r} + \frac{0.1}{\sqrt{r}}}$$

$$\frac{d}{dr} \cdot Speedup = 0 = \frac{63.2099 - 17.7778r}{\sqrt{r} \cdot (r + 3.55556)^2}$$

$$\Longrightarrow \sqrt{r} = 1.88562 \Rightarrow r \approx 3.56,$$

$$\Longrightarrow r_{optimal} = 4$$

ב. כמו כן: $Speedup = \frac{1}{\frac{1-A}{perf(r)}+\frac{A}{perf(r)+n-r}}$

ושוב:

$$A = 0.9, n = 32, per\ f(r) = \sqrt{r}$$

$$Speedup = \frac{1}{\frac{0.9}{-r+\sqrt{r}+32} + \frac{0.1}{\sqrt{r}}}$$

$$\frac{d}{dr} Speedup = 0 \Rightarrow \frac{1}{0.141358 - 0.0112637\sqrt{r}} + \sqrt{r} = 13.0163$$

$$\Longrightarrow r \approx 11.275, r \approx 493.21$$

$$\Rightarrow r_{optimal} = 11$$

ג. בשימוש ב-*Dynamic Multicore Chip*, החלק המקבילי רץ על יחידות *BCE* ששייכות לליבה אחת, שתריץ את החלק הסדרתי כליבה גדולה. במקרה האופטימלי כל יחידות ה*BCE* (32) ישמשו את התוכנית בריצת החלק הסדרתי על הליבה (כי ביצועי הליבה הסדרתיים ביחס ישר למספר היחידות שמרכיבות אותה), והן ישמשו בנפרד לכשירוץ החלק המקבילי. כלומר קונפיגורציה אופטימלית $r_{optimal} = 32$.