הטכניון - מכון טכנולוגי לישראל **הפקולטה למדעי המחשב**

18 לספטמבר 2014

פרופ"ח ארז פטרנק
נחשון כהן

236780 - אלגוריתמים לניהול זיכרון
אביב תשע"ד – מועד ב'

הנחיות:

- הבחינה עם חומר סגור.
- אסור להחזיק פלאפונים ברשות הנבחנים בזמן הבחינה.
- כתבו בצורה מסודרת ונקייה ובכתב ברור.
- בבחינה 3 שאלות, משך הבחינה שלוש שעות.
- נמקו את כל תשובותיכם.
- כאשר מבקשים להציע פיתרון אלגוריתמי לאיזושהי בעיה, יש לנסות לתת פיתרון יעיל ככל היותר במקום ובזמן.

בהצלחה !

שאלה 1: (31 נקודות) שאלה זו מתייחסת לאלגוריתם ה snapshot.

שאלה זו מתייחסת לאלגוריתם ה-snapshot. כזכור, בתחילת האלגוריתם עוצרים את כל החוטים, סורקים ומסמנים את השורשים ומפעילים את ה-write-barrier. הנחה סמויה היא שאף חוט לא נמצא באמצע כתיבה בעת עצירת החוטים. נזכיר עתה איך נראה ה-write-barrier ונוסיף גם דגל write_barrier_flag שיאמר לתוכנית מתי לבצע את ה-write-barrier.

```
Update( ptr, obj ){
        If (write_barrier_flag) {
                saved = copy(O's pointers)     // Where ptr is a pointer in object O
                if (O is not dirty) {
                        log( saved )
                        SetDirty(O)
                }
        }
        *ptr = obj
}
```

שימו לב שאם מדליקים את דגל ה-write-barrier בזמן שחוט נמצא באמצע כתיבה אז הכתיבה עצמה עלולה להתרחש ללא שמירת הערך הישן לאחר זמן העצירה.

1. (6 נק) תארו את האלגוריתם snapshot mark-sweep המבוסס על ה-write-barrier הזה. הסבירו מה עושים בעת עצירת החוטים ואחר כך איך סורקים כל אובייקט במהלך הסריקה.
2. (6 נק) מדוע נחוצה השורה השלישית (ששומרת את הפוינטרים של O במקום פרטי של החוט)? מה קורה אם מורידים אותה ופשוט שומרים את הפוינטרים של O בשורה החמישית? תארו ריצה שבמהלכה קורה מאורע שגורם בילבול לאלגוריתם ה-snapshot.
3. (6 נק) אם עוצרים את החוטים ומדליקים את דגל ה-write-barrier בזמן שחוט נמצא באמצע כתיבה, אז הכתיבה עצמה עלולה להתרחש ללא שמירת הערך הישן לאחר זמן העצירה. הסבירו כיצד יכולה להתרחש כתיבה של פוינטר בזיכרון ללא שמירת הערך הישן, לאחר איתחול ה-write-barrier רק משום שהכתיבה התחילה לפני העצירה והסתיימה לאחריה. הסבירו באיזו שורת קוד נעצר המעבד ומה קורה אחרי שהוא חוזר לפעילות.
4. (6 נק) מדוע מצב כזה (שתארתם בסעיף הקודם) מסכן את הנכונות של האיסוף? תארו באופן ברור מהלך דברים שיגרום לטעות באיסוף.
5. (7 נק) הציעו write-barrier מורחב שמאפשר נכונות של האיסוף גם אם העצירה התבצעת באמצע כתיבה. רמז: ניתן להניח ש-roots נסרקים נכון גם אם החוט ישן. הציעו write-barrier שדואג שה-roots יצביעו (בזמן הנכון) על האובייקט אותו אתם חוששים לפספס בסריקה. הסבירו למה התיקון פותר את הבעיה.

שאלה 2: (37 נקודות) שאלה זו מתייחסת להשוואה ושילוב של אלגוריתם reference counting עם אלגוריתם mark-sweep.

בתוכניות מסויימות ישנם שני מבני נתונים. נקרא להם "המבנה האנרגטי" ו-"המבנה העייף". במבנה האנרגטי התוכנית משנה את הפויינטרים באופן תכוף. במבנה העייף הפויינטרים משתנים באופן נדיר. אפשר להניח שהאובייקטים בשני המבנים שונים אחד מהשני, כך שניתן להבחין ביניהם בזמן קומפילציה (עפ"י אובייקט מאיזה class מקצים ואובייקט מאיזה class משנים). כמו כן ידוע ששני מבני הנתונים הם אציקלים (כלומר לא מכילים מעגלים).

1. (5 נק) תארו את החיסכון במקום המושג עבור reference counting ע"י שימוש במונה מחוונים שגודלו קטן. מתי קורה overflow? מה עושים בעת overflow? איך ניתן להשתחרר מהמגבלה שנוצרת כאשר overflow קורה על שדה ספירת מחוונים ספציפי?
2. (4 נק) נניח שתוכנית משתמשת רק במבנה נתונים אנרגטי (גדול) ואין שום אובייקט ב-heap שאינו חלק מהמבנה. האם עדיף להשתמש עבור התוכנית הזו ב reference counting או ב tracing? הסבירו את היתרונות והחסרונות בכל שיטת איסוף עבור המבנה הזה.
3. (4 נק) נניח שהתוכנית משתמשת רק במבנה נתונים עייף (גדול) ואין שום אובייקט ב-heap שאינו חלק מהמבנה. האם עדיף להשתמש ב reference counting או ב tracing? הסבירו את היתרונות והחסרונות בכל שיטת איסוף עבור מבנה כזה.
4. (3 נק) האם תשובתכם לסעיף הקודם היתה משתנה אם המבנה העייף היה קטן?
5. (7 נק) כעת נניח שתוכנית משתמשת בשני מבני הנתונים גם יחד. כלומר, בשני מבני נתונים גדולים שאחד מהם עיף ואחד אנרגטי וב-heap אין שום דבר מלבדם. נניח גם שנתון שאין שום פוינטר במבנה נתונים אחד המצביע על אובייקט במבנה הנתונים השני. תארו שילוב של שתי שיטות האיסוף באותה ריצה כך שהאיסוף הוא concurrent. האיסוף יריץ את גירסת ה-snapshot של שתי שיטות האיסוף. תארו מה יהיה ה- write barrier, מה יקרה בעת עצירת החוטים, ומה יקרה בעת הריצה המקבילה אחר כך.
6. (7 נק) כעת המצב כמו ב (5), אולם יכולים להיות פויינטרים בודדים מאובייקטים במבנה האנרגטי אל אובייקטים במבנה העיף. תכננו אלגוריתם יעיל ככל יכולתכם המשלב את שתי שיטות הסריקה (בגירסת ה-snapshot). האם דרוש שינוי של ה-write-barrier? כיצד יתבצע האיסוף?
7. (7 נק) כעת המצב כמו ב (5), אולם יכולים להיות פויינטרים בודדים מהמבנה העייף אל המבנה האנרגטי. תכננו אלגוריתם יעיל ככל יכולתכם המשלב את שתי שיטות הסריקה (בגירסת ה-snapshot). האם דרוש שינוי של ה-write-barrier? כיצד יתבצע האיסוף?

שאלה 3 (32 נקודות) שאלה זו מתייחת ל-concurrent mark-sweep.

בשאלה זו נתבונן באלגוריתם ה concurrent mark-sweep שבו מופעלים השלבים הבאים

1. עצירת כל החוטים, סריקת שורשים והפעלת write-barrier
2. סריקה מקבילית
3. כיבוי ה write-barrier ו-sweep.

ה-write-barrier שבו נשתמש יסמן באפור (או יכניס ל-markstack) את האובייקט שאיבד פוינטר במהלך הכתיבה. האלגוריתם יבצע write-barrier רק על מצביעים ב-heap, ולא יבצע את ה-write-barrier על השורשים.

1. (5 נק) הסבירו למה מתקבלת סריקה מסוג snapshot. כלומר, הסריקה מתבצעת כאילו ה-heap הוקפא בנקודה מסויימת בזמן. מהי הנקודה בזמן שעבורה מצב ה-heap מוקפא ולפיהו אנו סורקים? מדוע מחוייב שאנו סורקים כל פוינטר שהיה קיים בזמן שהגדרתם?
2. (4 נק) מדוע נשמרת נכונות הסריקה למרות שאין write-barrier על השורשים?
3. (3 נק) אם אובייקט נהפך לזבל לאחר עצירת החוטים, לא ניתן לאסוף אותו. הסבירו מדוע.
4. (5 נק) נרצה לתכנן אלגוריתם שיכול לאסוף גם חלק מהאובייקטים שנהפכים ללא נגישים במהלך ריצת האיסוף. לשם כך נשתמש ב-write-barrier שמסמן באפור דווקא את האובייקט שמקבל פוינטר במהלך הכתיבה. כלומר, נסמן באפור את האובייקט המוצבע ע"י הערך החדש שנכתב לפוינטר. בסעיף זה נניח שה-write-barrier פועל גם על השורשים (ה-stack והרגיסטרים). האם האלגוריתם המתקבל הוא נכון? נמקו.
5. (6 נק) עתה נתבונן באלגוריתם של הסעיף הקודם כאשר ה-write-barrier פועל רק על מצביעים ב-heap אבל לא על מצביעים בשורשים. תארו מצב שבו הסריקה לא תסמן אובייקט למרות שהוא נגיש.
6. (9 נק) הציעו שינוי לאלגוריתם שיהפוך את האלגוריתם לתקין ללא הפעלת write-barrier על השורשים. רמז: מותר לסרוק את השורשים יותר מפעם אחת. נסו להציע אלגוריתם יעיל ככל האפשר.