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

3 ליולי 2014

פרופ"ח ארז פטרנק
נחשון כהן

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

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: (27 נקודות) שאלה זו מתייחסת לאלגוריתם הרכבת.

א. (2 נק) מדוע המציאו את אלגוריתם הרכבת? (איזה צורך הוא פותר?)
ב. (3 נק) מדוע דרושות רכבות ולא מספיק להשתמש רק בקרונות?
ג. (3 נק) תארו עלות מיוחדת של אלגוריתם הרכבת בהשוואה לאלגוריתמים סטנדרטיים אחרים לאיסוף ה-heap. (אין להסתפק באמירה שהאלגוריתם מחייב write-barrier כי write-barrier קיים כמעט בכל האלגוריתמים שלמדנו. ציינו עלות שהיא ייחודית לאלגוריתם הרכבת.)
ד. (4 נק) תארו (ניתן להיעזר בציור) את הבאג שגילו זליגמן וגראפו. יש לתאר את מבנה ה-heap ואת הסיטואציה שבה יקרה הבאג. האם הבאג פוגע בנכונות של האלגוריתם? במה מתבטאת הבעיה בעת שהבאג קורה?
ה. (5 נק) תכננו תוכנית שתנסה כמיטב יכולתה לגרום לבאג לקרות. לתוכנית מותר לקרוא לאוסף הזבל באופן יזום ואפשר להניח heap גדול כך שלא יקרה איסוף זבל לא יזום (אם יש מספיק איסופים יזומים).
ו. (6 נקודות) נציע תיקון שונה מזה שהוצג ע"י Seligman-Grarup.
התיקון: בעת ריצת התוכנית עם אלגוריתם הרכבת יש write barrier המעדכן את ה- remembered set. נוסיף ל-write barrier בדיקה המסמנת את אובייקט A כ-"מיוחד" אם קורה המקרה הבא: המצביע אותו משנים עתה, p, עובר להצביע אל אובייקט A, וגם p כבר הצביע על האובייקט A באיסוף הלפני אחרון. בעת איסוף הקרון, נעתיק את כל האובייקטים שהוגדרו כמיוחדים אל הרכבת האחרונה ונוציא אותם מרשימת המיוחדים. הניחו שה-write-barrier עובד גם על ה-roots.
האם גם תיקון זה פותר את המצב הבעייתי הספציפי שתיארתם בסעיף ג'? אם כן נמקו ואם לא תנו דוגמא נגדית. אם כתבתם שהתיקון פותר את המצב שתואר על ידיכם בסעיף ג', האם יש מצב אחר שהתיקון הזה אינו פותר? או שפיתרון זה מתקן לחלוטין את הבאג? הסבירו.
ז. (4 נק) תארו מימוש (לאו דווקא יעיל) של ה-write-barrier המוצע בסעיף הקודם. מה העלות במקום ובזמן של ה-write-barrier לפי המימוש שהצעתם?

שאלה 2: (36 נקודות) שאלה זו עוסקת בתכנון Mostly Concurrent Copying GC.

שאלה זו מנסה להרחיב את אלגוריתם ה- concurrent Mostly לעבוד עם copying GC.
כדי לא לעצור את התוכנית לכל משך ההעתקה באלגוריתם ה-copying, נריץ אותו באופן מקבילי. עם זאת, לא נרצה להשתמש ב- read-barrier (כמו באלגוריתם של בייקר) כיוון שהוא מאט את התוכנית. במקום להשתמש באלגוריתם של בייקר, נתכנן בשאלה זו מסגרת ריצה מקבילית שתהיה דומה ל mostly concurrent. כזכור mostly concurrent המקורי עובד עם mark-sweep. אנו נפעיל אותו עם copying לפי השלבים המקוריים הבאים:

I. עצור את התוכנית, טפל בשורשים, ואתחל write-barrier אשר מסמן כרטיסים ב-from-space שמידע עליהם השתנה.
II. המשך את התוכנית (שעדיין רצה על from-space). אבל בו זמנית חוט איסוף זבל מבצע העתקה של הזכרון אל to-space לפי אלגוריתם copying GC (ובפרט האלגוריתם של Chenney).
III. בצע פעולת ניקוי כרטיסים (במקביל לריצת התוכנית).
IV. עצור את התוכנית, נקה כרטיסים, ועבור לעבוד על to-space.

1. (3 נק) תארו מה עושה ה-write-barrier באלגוריתם המקורי של ה-mostly concurrent. (אנו נשתמש בדיוק באותו write-barrier באלגוריתם של שאלה זו).
2. (3 נק) מה עושים עם השורשים בזמן עצירת התוכנית באלגוריתם mostly-concurrent המקורי?

3. (3 נק) מה אתם מציעים לעשות בעת הטיפול בשורשים בזמן עצירת התוכנית באלגוריתם mostly-concurrent החדש?
4. (8 נק) תכננו כיצד לנקות כרטיס באלגוריתם החדש.
ציינו מתי מורידים את סימון הלכלוך של הכרטיס, ציינו מה עושים עם האובייקטים שעל הכרטיס. כמו כן, העתקת זיכרון מ-from-space אל to-space בעת ניקוי כרטיס היא מיידית לרב סוגי השדות (מספרים, מחרוזות, וכיו״ב) אבל לא מיידית לפוינטרים. הסבירו מה צריך לעשות כשמעתיקים פוינטר אל to-space.
5. (4 נק) האם יש צורך בשמירת forwarding-pointers באובייקטים של from-space? אם כן, ציינו את כל השימושים בהם במהלך האלגוריתם. אם לא, הסבירו מדוע הם היו נחוצים באלגוריתם המקורי ומדוע הם אינם נחוצים יותר.
6. (6 נק) הסבירו מה הבעיה שנוצרת אם מריצים את אלגוריתם ה-copying הרגיל במקביל לריצת התוכנית ללא שימוש ב- write-barrier וללא ביצוע שום פעולה של ניקוי כרטיסים. תנו דוגמא לתצורה של ריצה שבה אוספים אובייקט שהתוכנית עדיין צריכה.
7. (6 נק) הציעו פיתרון לאלוקציה של אובייקטים חדשים. איפה נקצה אותם? תארו כל מה שדרוש לעשות כדי שהאלוקציה תתממשק נכון עם האלגוריצם של האיסוף ותאפשר נכונות של האלגוריתם.
8. (3 נק) הסבירו כיצד נקבע סדר האובייקטים שמתקבל ב-to-space.

שאלה 3: (38 נקודות) שאלה זו מתייחסת לאלגוריתם ה-Incremental Copying של Baker

א. (3 נק) מתי אלגוריתם איסוף נקרא incremental? מתי אלגוריתם איסוף נקרא concurrent? מה המטרה המשותפת של שתי השיטות?
ב. (6 נק) מהי השמורה העיקרית של האלגוריתם של Baker? מדוע היא חשובה לנכונות האלגוריתם? כיצד היא נשמרת בזמן שהתכנית רצה?
ג. (5 נק) החיסרון העיקרי של האלגוריתם הוא שצריך להשתמש ב-read barrier. מה לא יעבוד כשורה אם נבצע את העדכונים ב-write barrier? כלומר, אותו קוד שמתבצע לפני כל קריאה יתבצע רק לפני כל כתיבה? תנו דוגמה לכישלון של ריצת תוכנית.
ד. (8 נק) האלגוריתם של בייקר כפי שתואר בכיתה מתאים לחוט אחד של תוכנית. ציינו דוגמה של בעיה שעלולה להיווצר כאשר יש שני חוטים רצים ושניהם מבצעים read-barrier בעת ובעונה אחת. איך הייתם פותרים את הבעיה?
ה. (8 נק) ציינו דוגמה של בעיה שנוצרת כאשר יש שני חוטים רצים אחד מבצע read-barrier ואחד אחר מבצע אלוקציה ועבודת איסוף במסגרת האלוקציה ושניהם מבצעים אלוקציות ובשל כך גם עבודה עבור האיסוף במקביל. (אין להשתמש באותה בעיה שתיארתם בסעיף ד'.) איך הייתם פותרים את הבעיה?
ו. (8 נק) עתה נתבונן במצב שבו שני חוטים רצים, ושניהם מבצעים אלוקציות ובשל כך גם עבודה עבור האיסוף במקביל. האם הפעלת הפיתרונות שהצעתם בשני הסעיפים הקודמים מאפשרים לפעולות האלו לעבוד במקביל ללא חיכוכים? או שנוצרת בעיה נוספת למרות הפתרונות הקודמים? אם כל הבעיות נפתרות, תארו אילו בעיות עולות ומדוע הפתרונות הקודמים מספיקים. אם קיימת בעיה נוספת תארו אותה ותארו מה צריך להוסיף לאלגוריתם כדי לפתור אותה.