**תכנות מערכות מבוזרות - עבודה 1 - READ ME:**

**מגישים:** אוראל חג'ג' 204501704, כפיר רימברג 311374185

**AMI used**: ami-0aaf300e7e9b2a831

**instance type**: InstanceType.*T2\_MICRO*

**כיצד להריץ את הפרוייקט?**

java -jar Application.jar <input file> <output file> <numofworkers> <terminate\*> [\*-optional]

**זמני ריצה:**

עבור קובץ הקלט עם 2500 הקבצים – כולל יצירת מנהל, הקצאת 250 קבצים לעובד (כלומר 10 עובדים) ואפליקציה נוספת שמטופלת במקביל – כ-**5 דקות**.

עבור קובץ קלט עם 50 קבצים ו-25 קבצים לעובד (2 עובדים סה"כ) – כ-**40 שניות**.

**אופן המימוש:**

הפרויקט מחולק ל3 ישויות – האפליקציה, שמשמשת את משתמשי הקצה ומגישה בקשות עיבוד קבצים. העובדים, שמעבדים את קבצי הPDF שמבקשות האפליקציות לתמונה, טקסט או HTML, והמנהל, שמנהל את כל העבודה – בקשות האפליקציות והטיפול בהן באמצעות העובדים.

התקשורת בין הישויות מתבצעת באמצעות מערכת תורים. ישנם שני תורים מרכזיים – תור בקשות האפליקציות ותור תשובות המנהל – שמשמשים לתקשורת בין האפליקציות למנהל. התורים הנוספים הם תורים בין המנהל לעובדים – שני תורים לכל אפליקציה – תור משימות העובדים ותור תשובות העובדים.

**אפליקציה:**

האפליקציה היא החלק שמשמש את משתמשי הקצה. לאפליקציה המשתמש מזין את קובץ הקלט וממנה הוא מקבל את פלט התכנית. האפליקציה מגישה בקשה למנהל התכנית (Manager) ומחכה לתשובה ממנו. כאשר מגיעה תשובה מהמנהל, האפליקציה בודקת אם התשובה חיובית או שלילית – במידה ושלילית מחזירה הודעת שגיאה, במידה וחיובית – מורידה את קובץ הפלט, ממירה אותו לקובץ HTML ומסיימת את ריצתה. בסיום הריצה המשתמש יכול לצפות בקובץ הHTML המכיל את פלט התוכנית כפי שהוגדר בעמוד הפרויקט.

בתחילת ריצתה יוצרת האפליקציה מזהה ייחודי לעצמה לפי הtimestamp הנוכחי. מזהה זה משמש את האפליקציה בשירותים המשותפים – המנהל, מערכת הקבצים S3 (שם הסל במערכת נקבע לפי מזהה האפליקציה) ומערכת התורים. האפליקציה יוצרת סל במערכת הקבצים ששמו הוא המזהה הייחודי, ומעלה אליו את קובץ הקלט תחת השם המוסכם input.txt. לאחר מכן בודקת האפליקציה האם קיים מנהל שרץ במערכת. אם כן, מושכת את המזהה שלו ועובדת איתו, אם לא – יוצרת מנהל חדש. לאחר מכן מנסה האפליקציה ליצור את תורי התקשורת של האפליקציות והמנהל (שהוזכרו למעלה). אם התורים כבר קיימים, האפליקציה מושכת את הכתובות שלהם.

האפליקציה מגישה בקשה למנהל דרך תור בקשות האפליקציות. האפליקציה כותבת בהודעה לתור את מזהה האפליקציה הייחודי שלה ואת מספר הקבצים לעובד. לאחר הגשה הבקשה האפליקציה בודקת את תור תשובות המנהל לאפליקציות – היא מושכת את כל ההודעות שנמצאות בתור ללא visibility timeout – כלומר ההודעות לא נעלמות מהתור. האפליקציה עוברת על כל הודעה ומחפשת בה את המזהה שלה. אם מצאה, היא מוחקת את ההודעה מהתור. כך מובטח לנו שאפליקציות לא יעכבו אחת את השניה – כולן מושכות את כל ההודעות בו זמנית – וכל אפליקציה תמחק רק את ההודעה הרלוונטית אליה. האפליקציה בודקת את תשובת המנהל – תשובה שלילית גוררת הדפסת הודעה שגיאה, ותשובה חיובית גוררת קריאה לפונקציית יצירת הקלט. כדי למנוע מהאפליקציה להיתקע במקרה והמנהל הפסיק לעבוד בצורה לא צפויה (קריסת המחשב שמריץ את המנהל לדוגמא), האפליקציה בודקת אם המנהל עדיין במצב ריצה במידה ולא היו הודעות בתור. אם לא, האפליקציה תציג הודעת שגיאה ותמחק את הבקשה שלה מתור הבקשות במידה ועדיין קיים. בכל מקרה של שגיאה בטיפול בבקשה, האפליקציה מוחקת את הסל שיצרה יחד עם כל הקבצים שבו.

האפליקציה מורידה את קובץ הפלט (עם השם המוסכם output.txt) שנמצא בסל שלה וממירה אותו לקובץ HTML. הקובץ נשמר תחת השם שביקש המשתמש, אם הוזנה האפשרות terminate, שולחת האפליקציה למנהל הודעת “terminate” דרך תור בקשות האפליקציות ומסיימת את ריצתה.

**עובד:**

העובד הוא מחשב שמבצע את עיבוד קבצי הPDF שמבקשות האפליקציות. העובדים נוצרים וכבים ע"י המנהל. במהלך ריצתו העובד מושך בקשות עיבוד קבצים מהתור הרלוונטי אליו, מעבד את הקבצים ומחזיר הודעות סיום למנהל.

העובד נוצר עם מזהה האפליקציה שבקבצים שלה הוא הולך לטפל, יחד עם כתובות התורים הרלוונטיים לאפליקציה – תור משימות העיבוד ותור המשימות שהושלמו. העובד מתחבר לתור המשימות ומתחיל למשוך משם הודעה אחת בכל פעם. כאן מוגדר visibility timeout של 15 שניות – זהו הזמן המירבי שמצופה מהעובד לטפל בקובץ. כך אנו מבטיחים שלא ייווצר מצב ששני עובדים מטפלים באותו קובץ, אך במידה ועובד משך קובץ וקרס באמצע – הבקשה תחזור חזרה לתור ועובד אחר ימשוך אותה – וכך לא תהיה בקשה שלא טופלה. המשיכה של הודעה אחת בכל פעם מאפשרת לעובדים "מהירים" יותר לטפל ביותר משימות – אין בהכרח חלוקה אחידה של המשימות לעובדים – וכך מושגים הביצועים המיטביים.

ההודעה מכילה את מספר הקובץ, את הלינק לקובץ ואת הפעולה שנדרש לבצע עליו (תמונה/טקסט/HTML). העובד מוריד את הקובץ מהרשת ומנסה לבצע עליו את פעולת העיבוד המבוקשת. במידה והצליח – קובץ הפלט מועלה למערכת הקבצים לסל של האפליקציה, כך ששם הקובץ הוא מספר הקובץ שהגיע בהודעה. העובד מייצר הודעת סיום בהצלחה, בה הוא כותב את הלינק לקובץ הקלט, את הפעולה שבוצעה ולינק לקובץ הפלט מS3. במידה והייתה שגיאה במהלך העיבוד מייצר העובד הודעת סיום בה הוא כותב את הלינק לקובץ הקלט, את הפעולה שבוקשה ואת תיאור השגיאה שקרתה (לדוגמא read time-out, page not found).

לאחר מכן העובד מוחק את הודעת הבקשה מתור המשימות (הבקשה טופלה) ומנסה למשוך הודעה נוספת. התהליך חוזר על עצמו עד שהעובד מכובה ע"י המנהל.

**מנהל:**

המנהל הוא הישות שמנהלת את הטיפול באפליקציות ואת העובדים. המנהל הוא יחיד במערכת – בכל זמן נתון רץ לא יותר ממנהל אחד. כדי לאפשר סקלביליות של המערכת המנהל בנוי משתי תתי ישויות – המנהל הראשי ותתי המנהלים. המנהל הראשי הוא ת'רד יחיד שאחראי על ניהול התקשורת מול האפליקציות. הוא מושך בקשות טיפול מאפליקציות וכותב הודעות סיום אליהן. תתי המנהלים הם ת'רדים (מספר קבוע) בThread Pool האחראים על ניהול הטיפול בכל אפליקציה (יורחב בהמשך). מספר קבוע של ת'רדים שמטפלים במשימות במקטעים כאשר ת'רד אחד אחראי על התקשורת עם האפליקציות מאפשרת מערכת סקלבילית בעל תגובה מהירה. הת'רד הראשי מוסיף משימות לתתי המנהלים, ותתי המנהלים כותבים הודעות סיום טיפול באפליקציות לתור אותו קורא המנהל הראשי.

כאשר המנהל נוצר הוא מושך את כתובות התורים של התקשורת עם האפליקציות (בהכרח קיימים לפני שנוצר) ויוצר את הת'רד פול. הת'רד הראשי מבצע לולאה קבועה – הוא מושך 5 בקשות (לכל היותר) מתור בקשות האפליקציות (ללא visibility timeout מכיוון שהוא היחיד שמושך הודעות מהתור הזה). לכל בקשה יוצר המנהל הראשי משימת עיבוד קלט ומוסיף אותה למשימות של תתי המנהלים, מוסיף את האפליקציה לתור המשימות בטיפול ומוחק את הבקשה מהתור. לאחר מכן מושך 5 הודעות סיום טיפול (לכל היותר). לכל הודעת סיום מייצר המנהל הודעת סיום טיפול בהצלחה ומוסיף אותה לתור סיום הבקשות של האפליקציות. במידה וקיבל המנהל הראשי הודעת סיום (“terminate”), מתחיל נוהל סיום מסודר – המנהל מחכה שתור האפליקציות בטיפול יתרוקן – כך נדע שכל האפליקציות שכבר בטיפול טופלו, ואז מושך את הודעות סיום הטיפול וכותב אותן לאפליקציות. לאחר מכן המנהל מושך את כל הבקשות שהגיעו לאחר הודעת הסיום מאפליקציות ומחזיר להן הודעת שגיאה (כך מובטח שטיפול באפליקציה לא ילך לאיבוד). המנהל מוחק את תורי האפליקציות ומסיים את ריצתו.

לתתי המנהלים יש שתי משימות אפשריות לביצוע: **עיבוד קלט** ו**עיבוד פלט**.

**עיבוד קלט:** זו המשימה הראשונה שנוצרת לטיפול באפליקציה והיא מטופלת בבת אחת. תת המנהל שמטפל במשימה הזו מוריד את קובץ הקלט מהסל של האפליקציה (לפי המזהה שלה). הוא יוצר את תורי התקשורת עם העובדים לאפליקציה הזו, ואז מעבד את הקובץ ומייצר ממנו משימות לעובדים, אותן הוא שם בתור המשימות. תוך כדי המנהל סופר את מספר הקבצים לעיבוד. בסיום הטיפול בקובץ הקלט המנהל מחשב את כמות העובדים הדרושה (לפי מספר הקבצים לחלק למספר הקבצים לעובד שביקשה האפליקציה), ומייצר עובדים בהתאם. בסיום המשימה יוצר המנהל משימת עיבוד פלט לאפליקציה ומוסיף אותה לפול. הסיבה שמשימה זו מטופלת בפעם אחת היא שעד סיום הטיפול בקובץ לא נוכל לדעת את כמות העובדים הדרושה ולכן לא נוכל להפעילם – וזה בזבוז זמן מיותר.

**עיבוד פלט:** במשימה הזו תת המנהל מושך הודעות סיום טיפול בקבצים שמייצרים העובדים ומייצר מהן קובץ קלט. המשימה הזו מתבצעת **בשלבים** – המנהל שמטפל במשימה יטפל בלכל היותר 10 הודעות הסיום שנמצאות בתור. כך אנו מבטיחים שיהיה טיפול שווה בכל האפליקציות ות'רד לא יתעכב על אפליקציה מסוימת. המנהל מושך הודעות מתור סיום הטיפול במשימות של העובדים, ולכל בקשה יוצר שורה שתהיה בקובץ הקלט. יש שתי אופציות לאחסון השורה – כדי למנוע אחסון של קבצי פלט גדולים או הורדה והעלאה של קובץ הפלט בכל טיפול במשימה הזו, ההודעות נשמרות בbuffer שמוקצה למשימה. תתי המנהלים ממלאים את הbuffer במידע. כאשר הbuffer מלא או שנכתבו כל התשובות, תת המנהל מוריד את קובץ הפלט ממערכת הקבצים וכותב אליו את הbuffer. במידה וקובץ הפלט נמצא מקומית ברגע זה – המנהל כותב אליו את המידע ישירות. בסיום הריצה על המשימה (מקטע אחד של ריצה) הקובץ מועלה בחזרה לS3.

המנהל יודע מתי לסיים את משימת הטיפול בפלט לפי מספר ההודעות – ביצירת המשימה מועבר מספר הבקשות ממשימת עיבוד הקלט (שם הבקשות נספרות), ובכל משיכה של בקשות המספר מעודכן בהתאם. המשימה מסתיימת כאשר מספר התשובות שנשאר לקבל הוא 0. בשלב זה יוצר המנהל הודעת סיום טיפול ומוסיף אותה לתור התקשורת עם המנהל הראשי, מסיר את האפליקציה מרשימת המשימות בטיפול ולאחר מכן מכבה את כל העובדים שהוקצו לאפליקציה (המזהים שלהם מועברים ממשימת הקלט).

כדי למנוע מצב של תקיעת הטיפול בעקבות קריסת העובדים, במידה ולא היו הודעות בתור סיום המשימות המנהל בודק שכל העובדים רצים. במידה ועובד נפל המנהל יוצר עובד אחר במקומו.

**בטיחות:**

כדי שפרטי ההתחברות לא יהיו חשופים, קבצי הjar מכווצים לקבצי zip עם סיסמא.

**הפעלת הinstances:**

האפליקציה בודקת בתחילת ריצתה שהסל שמכיל את הקוד של המנהל והעובד (תחת השם המוסכם kfirorel) קיים. במידה ולא האפליקציה יוצרת את הסל ומעלה לשם את קבצי הzip של המנהל והעובד.

כאשר המנהל/עובד מופעלים, הם מורידים את את קובץ הzip הרלוונטי להם, מחלצים אותו עם הסיסמא ומריצים את קובץ הjar. לאחר סיום ההרצה הם מתכבים (ע"י shutdown -h now). בכך הinstances יתכבו במקרה של סיום לא צפוי של הריצה.