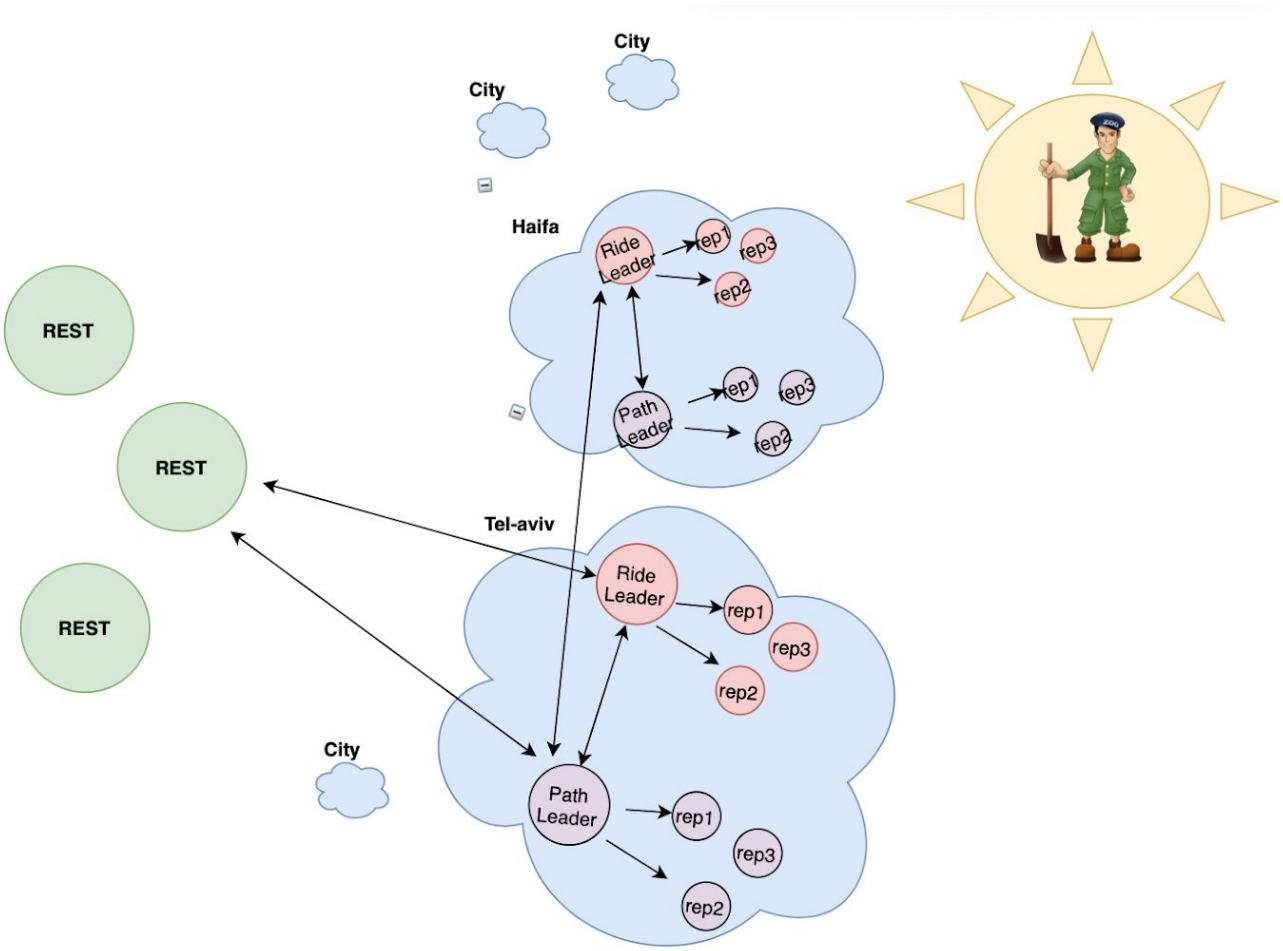
מערכות מבוזרות
תרגיל בית 2

מגישים:

גיא שפירא 208136820

גל סידי 315467753

https://github.com/Guy-Shapira/236351_HW2



תכנון המערכת:

לוגיקה:

כל עיר מיוצגת ע"י קבוצת שרתים במערכת. השרתים של עיר מחולקים לשתי תתי קבוצות - קבוצה אחת אשר אחראית על רישום טרמפים חדשים והקבוצה השניה אחראית על בקשות למסלולי נסיעה והצטרפות לטרמפים קיימים.
לכל אחת מהקבוצות עבור כל עיר, קיים מנהיג יחיד והבקשות לקבוצה מטופלות ע"י המנהיג.

בעת בקשה של משתמש עבור מסלול *P = p1-p2-p3-....-pn* כלומר מסלול המורכב מ: n תתי מסלולים, הבקשה תטופל ע"י המנהיג שאחראי על קבוצת תכנון המסלולים של עיר המקור של p1 (נסמנו S).

עבור כל pi יפנה S למנהיגים של קבוצות הטרמפים הקיימים מערים אחרות במערכת ויחפש נסיעות שעושות את המלסול pi בין אם באופן ישיר ובין אם בעיקוף.
אם המנהיג מצא נסיעה מתאימה לpi שביקש S, ישמור בשבילו מקום בנסיעה.
במידה וקיימים טרמפים עם מקום פנוי לכל תתי המסלולים בP , השרת S יאשרר את המקומות שנתפסו בשבילו בכל אחד מתתי המסלולים (יזמין את הנסיעה).
במידה ולא קיימת נסיעה מתאימה (או שכל הנסיעות המתאימות למקטע מסויים תפוסות לחלוטין, או שלא פורסם טרמפ מתאים כלל) יוחלט על כישלון טוטאלי ואף קטסטרופאלי של הטיול.
אם שוריינו בהצלחה מקומות לכל תתי הנסיעות במסלול יוחזר למשתמש הצלחה סופית על המסלול, נוסף על פרטי המסלול עצמו.

תקשורת וניהול בין הישויות:

שרתי REST:

כל תקשורת בין משתמש קצה (לקוח) לבין המערכת מתבצעת דרך שרתי הREST. המתמשים מופנים לשרתי הREST באמצעות תקשורת מול הZOOKEEPER שמחזיק את אוסף שרתי הREST הקיימים ופעילים. כאשר משתמש פונה לשרת REST עם בקשה מסויימת, שרת הREST יפנה את הבקשה למנהיג בקבוצה המתאימה בעיר המתאימה לטיפול בבקשה. כדי להתמודד עם עומסים מוקצים כמה שרתי REST לתקשורת ישירה עם המשתמשים, ומשתמש מופנה אקראית לאחד משרתי הREST בתקשורת אל מול המערכת על מנת לפזר את העומס בצורה יעילה.
אם שרת REST מסויים מגיש בקשה מסויימת לטיפול והיא לא חוזרת אחרי timeout מסויים הבקשה תוגש מחדש.

ZOOKEEPER :

כל ניהול הcity clusters מתבצע באמצעות הZOOKEEPER . הרישום והשיוך של שרת לקבוצה מסויימת בעיר מסויימת מנוהל באמצעות ZK, והתקשורת בין השרתים (חיפוש המנהיג המתאים, חיפוש הרפליקות של מנהיג כלשהו) מתבצעת באמצעות המידע שZK מחזיק על מבנה המערכת. נוסף על כך, באמצעות ZK אנחנו מתשאלים ומגלים מתי שרתים לא זמינים (נפלו).

בנוסף, בעת כל הפניית בקשה לעיר מסויימת- משתמשים ב ZK על מנת לוודא כי לעיר הרלוונטית קיים מנהיג עבור הפעולה הרצויה. במידה ולא קיים, ממנים את השרת "הותיק ביותר" (חי הכי הרבה זמן) שקיים בcluster הנוכחי ובקבוצה המתאימה כמנהיג.

מבנה הCITY CLUSTER:

כל עיר מיוצגת ע"י cluster במערכת. הcluster מחולק לשתי תתי קבוצות של שרתים -

- Ride Posting Group **:** אחראית על רישום טרמפים חדשים.
- Path Planning Group: אחראית על בקשות למסלולי נסיעה, והצטרפות לטרמפים קיימים.

לכל אחת מהקבוצות קיים מנהיג יחיד ושאר השרתים בקבוצה מהווים רפליקה של המנהיג של אותה הקבוצה.

RIDE POSTING GROUP:

קבוצה זו אחראית לשמור מידע אודות הנסיעות הקיימות במערכת ויוצאות מהעיר שעליה היא אחראית. כל הבקשות לרישום טרמפ חדש מגיעות למנהיג של הקבוצה. בסיום הרישום של הטרמפ הוא מופץ לשאר הרפליקות.

כאשר מתבצע ניסיון להזמנת מסלול שהמקור שלו הוא העיר של cluster מסווים, השרת שהזמין את המסלול (שייך לpath planning group) יפנה אל המנהיג של קבוצת ride posing group על מנת לחפש מקומות פנויים בטרמפים מהעיר. במידה וקיים מקום פנוי המנהיג ישריין אותו ויודיע לשרת המבקש כי קיים טרמפ בשבילו, ולאחר מכן לכל הרפליקות בקבוצה. השרת המבקש יחזיר ACK על מנת להודיע כי אכן הוא נרשם לטרמפ הזה, ובכך הפעולה של שמירת מקום מאושרת. במידה ולא התקבל ACK לאחר treshold זמן (= המנהיג המבקש נפל) המקום שנשמר בשבילו ישוחרר.

במידה והשרת המבקש מתחרט על הבקשה (הוחלט לא לנסוע בטרמפ הנוכחי כי המסלול כולו לא מסתפק) הוא ישלח בקשת ביטול אל המנהיג ששמר לו את המקום, והמקום שנשמר בשבילו ישוחרר.

PATH PLANNING GROUP:

קבוצה זו אחראית לשמור מידע ולטפל בבקשות למסלולי נסיעה. כל הבקשות לתכנון נסיעה חדשה היוצאת מעיר מסויימת מגיעות למנהיג של הקבוצה של אותה העיר.

כאשר המנהיג מקבל את הבקשה הוא מפיץ אותה לשאר הרפליקות שלו והיא מסומנת כpending בכל אחת מהן.

לאחר מכן המנהיג מטפל בבקשה, כלומר עבור כל אחד מתתי המסלולים המקיימים במסלול המבוקש הוא ינסה להזמין אליו נסיעה מתאימה. כאשר כל הנסיונות להזמנה של המסלול השלם הסתיימו והוחלט האם נמצאו טרמפים לכל הlegs במסלול, או לחילופין המסלול כולו מבוטל, הבקשה מסומנת כ completed. המנהיג מפיץ לרפליקות שלו שהבקשה הושלמה ואת המידע עליה, ומיד לאחר מכן שולח ACK לשרתי הטרמפים במידה ונמצאו טרמפים מתאימים לכל הlegs ובכך משריין סופית את המקומות.

אם אחד משרתי הטרמפים אליהם שולחים ACK נפל והשליחה נכשלה, נאלץ לוותר על הטרמפים ששריינו סופית ולכן נשלח הודעת ביטול לשרתים שכבר שלחנו להם ACK. במידה וכל שליחות הACK עברו בהצלחה הנסיעה משוריינת סופית סופית סופית למשתמש.

התוצאה של הבקשה מוחזרת למשתמש.

הצטרפות nodes חדשים והתאוששות מנפילות:
כאשר בקשה חדשה מתקבלת בשרת REST כלשהו, היא מופנית למנהיג הרלוונטי של העיר ממנה הבקשה צריכה להיות מטופלת. בעת הפניית הבקשה מתבצעת בדיקה האם המנהיג של העיר הרלוונטית חי. במידה והמנהיג לא קיים נבחר מנהיג חדש לקבוצה על ידי שרת הREST והוא לוקח את מקומו של המנהיג הישן.

כאשר מצטרף node חדש לעיר מסויימת (לא משנה לאיזו קבוצה), או לחילופין שרת מסויים "מתעורר" מנפילה- הוא מבקש מהמנהיג של אותה הקבוצה snapshot של הstate הנוכחי, וכך הוא יכול להתעדכן במידע ולהפוך גם הוא לרפליקה מעודכנת של המנהיג.

נפילה של מנהיג בקבוצת PATH PLANNING GROUP:
כאשר המנהיג נופל השרת שיחליף את מקומו יפעל באופן הבא:
עבור כל בקשה למסלול שמסומנת כpending המנהיג הנבחר יבצע ביטול לבקשה באמצעות תקשורת אל מול השרתים של הערים במסלול, כלומר עליו לבטל את הנסיעות שהוזמנו עבור כל הlegs במסלול הנ"ל בכל אחד מהשרתים. פעולה זו מתבצעת על מנת לחסוך בזמן ולהתאושש במהירות (לפני שעובר הtreshold שהוגדר במידה ומתאפשר). לאחר שיעבור timeout ושרת ה REST יראה שהבקשה לא טופלה, הבקשה תוגש מחדש.

נפילה של מנהיג בקבוצת RIDE POSTING GROUP:
כאשר המנהיג נופל השרת שיחליף את מקומו יפעל באופן הבא:
עבור כל הנסיעות ששוריין אליהן מקום ולא התקבל ACK בפרק הזמן הרצוי שחרר מקום מנסיעה זו.

SNAPSHOT
כפי שהתבקשנו בהוראות התרגיל, נוסף על האפשרות לבקש תמונה של הstate של המנהיג בכל רגע מאחד מחברי הקבוצה שלו, אפשרנו גם לבקש תמונת מצב של המערכת כולה ברגע נתון. תמונת מצב זו כוללת את האיחוד של תמונות המצב הנלקחות מכל אחד מהמנהיגים במערכת.

בדיקות
על מנת לוודא את נכונות המערכת, זמינותה והיכולת שלה להתמודד עם נפילות ביצענו בדיקות מכמה סוגים.

- נכונות: ביצענו טסטים בסיסיים עבור נכונות פרסומי נסיעות ובקשות מסלול ממשתמשים במערכת.
- התמודדות עם עומסים: בדיקות דומות לבדיקות הנכונות הנ"ל ביצענו במהירות גבוהה ומספר רב של פעמים על מנת לראות שהמערכת לא נופלת והביצועים לא נפגעים באופן משמעותי.
- ניהול המידע בין שרתים : בדקנו שאכן ההודעות המועברות בין שרתים באותה קבוצה מתאימות למצב המערכת ומאפשרות התאוששות מהירה ונכונה בעת נפילת מנהיג.

- נפילה של מנהיג : ביצענו בדיקות בהן הפלנו את אחד המנהיגים של הקבוצות בעת טיפול בבקשות. בדיקה זו באה לוודא התאוששות מתאימה של הקבוצה, הכוללת בחירה מחודשת של מנהיג יחיד וטיפול בבקשות שלא הסתיימו.