עיבוד מבוזר

בתרגיל זה נבנה מערכת שמצליחה לפענח תוצאת חישוב של MD5. במקרה שלנו זיהוי סיסמה המורכבת מספרות בלבד. אורך הסיסמה היא בת 10 ספרות. מ- 0000000000 עד 9999999999

תזכורת: MD5 היא פעולה המקבלת רצף (מערך, buffer) של בתים ומחזירה מחרוזת של 32 תווי acsii (המייצגים 128 ביטים).

המערכת שלנו תקבל בשורת הפקודה (argv) את המחרוזת בת 32 תווי ה- ascii אשר התקבלו כתוצאה מהפעלת פעולת MD5 על מחרוזת בת 10 תווים שהם ספרות.

לדוגמא עבור **F18E3AE3DE669B865C76C87C87867FC8** המערכת שלנו תצטרך להחזיר **0012300456**

המערכת שלנו צריכה לפענח את המחרוזת: **EC9C0F7EDCC18A98B1F31853B1813301**

שרת

המשימה מוטלת על השרת. היות והפעולה עשויה לקחת הרבה זמן אז השרת יחלק את המשימה ללקוחות שיתחברו אליו (בתקשורת socket). השרת יחליט איך לחלק את המשימה למנות. לדוגמא יחלק למנות של מליון אופציות.

מנה אחת 0000000000 – 0000999999, מנה שנייה 0001000000 – 0001999999, מנה שלישית 0002000000 – 0002999999, וכך הלאה...

גודל המנה יהיה גמיש בהתאם לקבוע. החלוקה למנות יהיו רשומים בשרת כפרמטר/משתנה גלובאלי שאפשר יהיה לשנות.

כל לקוח שמתקשר לשרת אומר לו כמה CPU יש לו ואחוז שמוכן להיות עסוק (25 50 75 100) ולפי זה הוא שלוח לו מנות (מספר אחד שמתחיל את המנה וגודל המנה חשוב לתת שתי משימות לפחות לכל CPU). נניח שלקוח אומר שיש לו 4 מעבדים והוא מוכן להיות עסוק 75% אז השרת שולח ללקוח 4*2*0.75=6 מנות. המנות לא חייבים להיות רציפים.

עם סיום פעולת הלקוח, (אם לא מצא) יקבל מנות נוספות בהתאם לכמה שנשאר. אם מצא, נגמרה כל העבודה ויש להודיע לכל הלקוחות שהמשימה הסתיימה.!!

הרעיון הוא שאם יתחברו הרבה לקוחות ולכל לקוח יהיו כמה שיותר CPU הבעיה תיפתר מהר.

הלקוחות מתחברים בזמנם החופשי לשרת. השרת לא יודע כמה לקוחות יתחברו אליו. אם יתחברו אליו לקוחות אחרי שנמצאה השימה, אז הוא ייתן להם משימות. הוא יכול לא לתת להם להתחבר או להחזיר להם תשובה שאין משימות לביצוע.

לקוח

מקבל בשורת הפקודה (argv) כמה CPU יש לו "בכאילו" וכמה מוכן להיות עסוק מתחבר לשרת ומודיע לו כמות שמוכן לקבל. מקבל מנות ומטרה (את המחרוזת שיש לחשב) ומתחיל לחפש. הלקוח מפעיל thread עבור כל מנה. רצוי שכל thread יפעיל תהליך אמיתי (process) ולא thread כדי לנצל באמת יותר מ CPU אחד (כי threads בפייתון לא באמת רצים במקביל)

הנחיות נוספות

כללי

1. בתרגיל זה ישנם 3 שחקנים: שרת – לקוח – ופועלים (threads)
2. השרת מתקשר עם לקוחות. הלקוחות יכולים לרוץ על מחשבים שונים. הלקוחות מפעילים תהליכים שרצים על אותו מחשב של הלקוח.
3. ראשית וודאו שאתם מבינים איך מחשבים MD5 ואיך בודקים אם התוצאה נכונה. שימו לב איך מתקבלת התוצאה של MD5, אותיות גדולות או קטנות. נסו להריץ על מחרוזות שלכם וודאו שהאלגוריתם שלכם מוצא נכון על מחרוזת שלכם.
4. את הפיתוח אפשר לחלק למשימות בלתי תלויות. יש משימות מסוימות שניתן להתחיל לפתח לפני התקשורת. לדוגמא אפשר לפתח רק את הלקוח ללא תקשורת עם השרת ולראות שהוא אכן מפעיל מספר פועלים (threads or process) ומוצא את המחרוזת. ללא קשר לחלק הזה ניתן לפתח שרת לקוח שמבקש משימה לביצוע ומקבל אותה. כמו כן הוא יודע להפסיק את המשימה באמצע אם התקבלה הודעה מהשרת שיש להפסיק את המשימות. בכל מקרה, לפני שמתחילים לעבוד על התקשורת (בין אם זו משימה ראשונה ובין אם זה אחרי שכבר עבדת על משימות החישוב, יש לתכנן את הפרוטוקול על דף (מסמך word).
5. יש להשתמש בקבועים בתוכנית ולמקם אותם בתחילת המודול (בראש הקובץ/סקריפט).
6. יש לשאוף ולהשתמש במודולים נפרדים ובמחלקות. חלק מהמודולים והמחלקות ניתן להשתמש בהם גם בצד השרת וגם בצד הפועל (לקוח). לדוגמא מודול הפרוטוקול שאחראי לבנות ולפרק את ההודעות. מודול התקשורת. אלו מודולים זהים לצד השרת והלקוח ואין טעם (גם לא נכון) לכתוב אותם פעמיים.
7. כל שורות הקוד נמצאות בפונקציות. אין לכתוב שורות קוד בסקריפט שהן לא עטופות בפעולות.
8. יש להשתמש ב- "__main__" == __name__
9. שימו לב: יכול להיות שלקוחות שקיבלו משימה (מספר מנות) יתנתק הקשר איתם ולא יחזירו תשובה. השרת צריך לזהות מקרים כאלו ולהעביר את המשימות שלא הסתיימו ללקוחות אחרים. הסדר של חלוקת המנות ללקוחות לא חשוב. כלומר נניח שיש 100 משימות (מנות) ונתנו ללקוחות כבר את משימות 1-20, נניח שמשימות 10, 11, 15, 17 הלכו לאיבוד. השרת יחליט מתי הוא נותן משימות אלו ולמי. הוא יכול להחליט שקודם הוא יסיים לחלק את כל ה- 100 משימות ואח"כ יטפל שמשימות שללא הסתיימו והוא יכול להכניס מנות אלו לעבודה בהקדם האפשרי.

שרת

1. ינהל רשימה של משימות שכרגע בעבודה.
2. ינהל רשימה של משימות ממתינות לתחילת עבודה.
3. יקבל בקשה לעבודה מלקוח ויקצה לו טווח עבודה. יקח משימה מרשימת המשימות הממתינות ויעביר לרשימת המשימות שבעבודה.
4. יש לדאוג ולקשר בין לקוח למשימה שכרגע הוא עובד עליה.
5. השרת יתמוך בלקוחות חדשים שמצטרפים ולקוחות שנעלמים בפתאומיות.
6. במידה ולקוח התנתק (או לא חזר עם תשובה אחרי פרק זמן מסוים, timeout), יש להחזיר את המשימה שהוא עבד עליה, חזרה לרשימת המשימות שממתינות לעבודה.
7. במידה והעבודה הסתיימה, כלומר אחד הלקוחות חזר עם תשובה, לקוחות חדשים שיבקשו עבודה יקבלו תשובה שאין עבודה בשבילם והשרת יסגור את הערוץ (socket) איתם.
8. לאחר שהמשימה הסתיימה (המספר שממנו נוצרה מחרוזת ה- hash נמצאה) השרת יסמן ללקוחות שהם יכולים להפסיק את העבודה (במקום שהשרת יחכה עד שכל ההלקוחות יסיימו את העבודה שהם עובדים עליה כרגע). דבר זה ניתן למימוש בתקשורת אסינכרונית עם הלקוחות. לדוגמא ע"י זה שהלקוח ישלח הודעה לשרת כל זמן מסוים "האם להמשיך לעבוד".
9. שימו לב, רשימת המשימות שכרגע בעבודה הן לא סתם רשימת טווחים. יש צורך לשמור גם איזה לקוח עובד עליו כרגע. זה נדרש למקרה שהלקוח יתנתק באמצע ושיהיה מסוגל לדעת איזו משימה יש להחזיר לרשימה "ממתין לעבודה".
10. השתמשתי פה במסמך במונח רשימה, אבל אתם תחליטו מה מבנה הנתונים שהכי יתאים לכם (רשימה, מילון או מבנה אחר). כמו כן יש להחליט מה סוג האיברים שישמרו ברשימות האלו. מספרים פשוטים, מחרוזות, מחלקה עם מספר תכונות ועוד.

לקוח ופועלים

1. מקבל מ argv כמה CPU יש לו "בכאילו" מתחבר לשרת ומודיע לו כמות CPU. מקבל מנות ומטרה ומתחיל לחפש כל מנה ב thread אחר.
2. לקוח שסיים את המשימה שלו, יחזיר תשובה לשרת וישאל האם יש לשרת משימות נוספות עבורו.
3. ראשית כתוב פעולה שמקבלת טווח ומחרוזת hash ומחזירה תשובה 1- אם החישוב לא נמצא בטווח או את המספר שקיים את החישוב.
4. כתוב לקוח שמקבל טווח גדול לעבודה ומפעיל מספר פועלים (תהליכים, threads) ברקע כאשר כל תהליך מקבל טווח שונה לעבוד עליו. התהליך מריץ את הפעולה שנכתבה ב-1.
5. שפר את הלקוח שיבצע תקשורת עם השרת. יקבל ממנו טווח עבודה. הלקוח יודיע בזמן הבקשה מהשרת את עוצמת החישוב שיש לו (לדוגמא מספר הליבות/cpu שיש ברשותו) והשרת יחזיר לו טווח עבודה יותר גדול.
6. בכדי לשפר ביצועים (את מהירות החישוב במקביל) הלקוח יפעיל תהליכים (process) במקום תהליכונים (thread). ניתן לעשות זאת באמצעות המודול subprocess שלמדנו ולהשתמש בפעולה check_output או להשתמש במודול multiprocessing ובאוביקט מסוג queue
7. כאשר אחד הפועלים סיים את העבודה שלו על הטווח ולא מצא את הסיסמה, הלקוח ישלח לשרת הודעה שהעבודה על מנה זו הסתיימה ויבקש מנה נוספת.

דוגמא להפעלת השרת והלקוח משורת הפקודה:

```
python distributed_calc_server.py    EC9C0F7EDCC18A98B1F31853B1813301

python distributed_calc_client.py    <server_ip>   <number of cpu>    <load percent>

python distributed_calc_client.py    192.168.0.12    6    75
```

עזרה ב md5:

שימו לב שכל פעם יש להשתמש ב- 3 השורות הבאות מחדש. כי update מוסיף למחרוזת הקודמת.

וודאו טוב טוב שאתם מבינים איך לעבוד עם הפעולות המאות

```
import hashlib

md5_item = hashlib.md5()
md5_item.update(str(num).zfill(10))
curr = md5_item.hexdigest()
```

בהצלחה,

אופיר שביט.