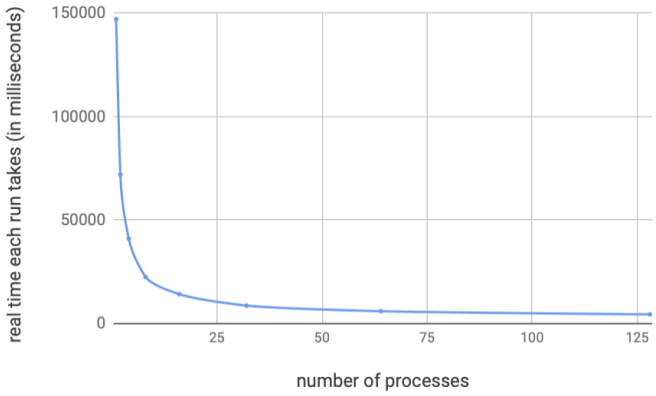
מערכות הפעלה – שיעורי בית 2

חלק 2 –

3. הגרף הוא -



4. אנו רואים שיפור בגרף למרות שאנו משתמשים בcore יחיד משום שאנו מריצים מספר proccessים במקביל. במקום שבכל פעם לאחר ביצוע פעולה בודדת נצטרך להמתין לתשובה מהשרת, חילקנו את העבודה בין מספר תהליכים שרצים במקביל כלומר מתבצעות מספר פעולות, מספר בקשות מהשרת וכך כאשר תהליך אחד ממתין השני יכול להמשיך לפעולה הבאה – הם עובדים במקביל ומשפרים את זמן הפעולה.
5. במידה ונבצע רק בדיקה של תקינות ה-URL הכתובת שנשלחת הגרף ישתנה בצורה משמעותית. בדיקת תקינות אינה דורשת ביצוע system calls ולכן, אין בתוכנית זמן בו התהליך ממתין לתגובה. כלומר, שימוש במספר תהליכים במקביל לא ייעל את זמן הריצה אלא יאט אותו. זאת משום שקריאה ויצירת תהליכים חדשים היא גם כן פעולה הלוקחת זמן ולכן הגרף יעלה ולא ירד.
6. במידה ונבצע שימוש ב-threadים במקום בתהליכים זמן הריצה ישתפר ויהיה מהיר יותר. Threadים חולקים זיכרון ולכן, מעבר מידע ביניהם מתבצע בצורה קלה יותר, זמן היצירה של thread קצר יותר משל process בדר"כ, ביצוע context switch בין תהליכים הוא תהליך ארוך יותר מאשר בין threadים כלומר ניתן לבצע פניות למערכת ההפעלה בצורה קלה יותר. לכן, זמן הריצה יתקצר.

חלק 3 –

א. הקוד בשאלה ידפיס את המספרים בין 1-101 מספר פעמים מסוים. בעבור כל מספר x בין 1 ל101 המספר x יודפס 2^x פעמים. כלומר, סה"כ יודפסו $\sum_{x=1}^{101} 2^x$ מספרים ושורות. זאת משום שבכל פעם כל התהליכים הרצים מתפצלים לשניים ומדפיסים את המספר. כלומר במידה והיו שני תהליכים כלומר כל תהליך ייצור 2 תהליכים ולכן יהיו לאחר ביצוע פעולת ה-fork 4 תהליכים אשר ידפיסו את המספר 2.

ב.

חלק 4 –

1. When using fork(), if the parent process exits before the child process then the child process terminates immediately (use Google...).

לא נכון, כאשר ההורה של תהליך מסוים מפסיק לרוץ הילד ממשיך לרוץ. השינוי היחידי המתבצע הוא במספר הסידורי של האב של הילד. הילד מקבל מספר סידורי שונה בעבור אבא שלו. כלומר הוא אינו מפסיק לרוץ.

2. When using fork(), the child process and the parent process cannot communicate through named pipes. Named pipes are only used for non-related processes.

   לא נכון, תהליכים אשר קשורים וגם תהליכים שאינם קשורים יכולים להשתמש ב- Named pipes.

3. Consider the following code:

```
public class Worker implements Runnable {
    @Override
    public void run() {
        ...
    }
}
...
public static void main(String[] args){
    Thread t = new Thread(new Worker());
    ??
}
```

   Replacing ?? with t.run() and t.start() results exactly in the same effects (Google is your best friend).

   לא נכון, החלפת ה?? ב-t.run( ) תריץ את ה-tread הנוכחי – כלומר לא יהיה שימוש במספר threadים במקביל, בעוד הרצת הפעולה t.start( ) תיצור tread חדש ותריץ אותו (פעולה פנימית) וירוצו מספר threadים במקביל.