סדנה ברשתות תקשורת – תרגיל 2

בתרגיל זה, נמדוד את ה-throughput של שליחת הודעות בין שני שרתים (במקרה זה, שרתי האוניברסיטה) (את השולח נכנה הלקוח, ואת המקבל נכנה השרת) תוך שימוש בטכנולוגיית RDMA.

נתון קובץ קוד bw\_template.c שמכיל כמה פונקציות שעוטפות את הפונקציות הבסיסיות של RDMA, והן אמורות לעזור לפשט את העבודה.

כמה מילים על הפונקציות בקובץ:

יש את הפונקציות הבסיסיות של שימוש בRDMA, שנמצאות בספריה infiniband/verbs, הן כולן מתחילות ב-ibv\_ (מלשון InfiniBandVerbs). ויש את פונקציות המעטפת בקוד, שמתחילות ב-pp\_ (מלשון PingPong, זו המטרה המקורית של הקובץ, בדיקת bandwidth באמצעות שליחת מידע דו צדדי, פינגפונג).

אנחנו אמורים עקרונית להשתמש רק בפונקציות המעטפת (pp\_). לא צריך ibv

המטרה, כמו בתרגיל 1, זה למדוד את הזמן שלוקח לשלוח הודעות בכל אחד מהגדלים מבית אחד ועד מגה בית אחת, ולחשב את ה-throughput.

בתרגיל אמורים להשתמש בפונקציה send כדי לשלוח את כל ההודעות בין הלקוח לשרת, וכן בין השרת ללקוח.

כפי שלמדנו בכיתה, פונקצית send מצריכה הקצאת מקום מראש ע"י המקבל, והשולח אינו יודע את המיקום המדוייק שאליו תגיע ההודעה (בשונה מ-write).

(ככה עובד send: המקבל מקצה מקום להודעה בודדת, ע"י שימוש בפונקציה post receive, ואז כאשר השולח ישלח ע"י שימוש בפונקציה post send, כרטיס הרשת של השולח ישלח את ההודעה לכרטיס הרשת של המקבל, והוא יעביר את ההודעה למקום שצויין ב-post receive.)

עיקר השינויים בקובץ bw\_template.c צריכים להיות החל מהשורה 809, שם אנחנו צריכים לשנות את השליחות בין הלקוח לשרת. כרגע מה שקורה זה שהלקוח שולח לשרת הודעה אחת, והשרת שולח ללקוח הודעה אחת, והריצה מסתיימת.



מה שאנחנו רוצים שיקרה זה שעבור כל גודל, נפתח שעון, נשלח את ההודעה iters פעמים, וכשהשרת מקבל את הכל, שישלח הודעת send ללקוח, וכשהלקוח יקבל את ההודעה, יעצור את השעון, ויחשב את הthroughput, וידפיס למסך.

לכאורה היינו צריכים להתעסק עם pp\_post\_receive, אבל למזלנו הפונקציה pp\_wait\_completion דואגת לעשות post\_receive בשבילנו, אז אנחנו לא צריכים להוסיף post\_receive בכלל בתרגיל (יש בקוד בשלב האיתחול post\_receive, צריך להשאיר את זה שם).

לכאורה, מה שצריך לעשות זה פשוט

If servername // it’s the client (the sender)

For I in range(21):

Start clock

For iter in iters:

Pp\_post\_send

Pp\_wait\_completion

Stop clock

Calculate

Print

Else // it’s the server (the receiver)

For I in range (21):

Pp\_wait\_completion

Pp\_post\_send

הבלגן שאיתו אתם צריכים להתמודד:

1. צריך לשנות את גדלי ההודעות. בשביל זה צריך לשנות משהו לפני שורה 809.
2. הפונקציה pp\_wait\_completion(ctx, x) מחכה ש(ישלחו או יתקבלו) x הודעות. לכן צריך לחשב איזה ערך x אתם צריכים לתת כדי להמתין בדיוק להודעה שהשרת שולח ללקוח (שאומרת "קיבלתי את כל ההודעות ששלחת").
3. יש הגבלה על כמות בקשות ה-post שיכולות להיות במקביל, לכן יש כבר בקוד לולאה בלקוח שכל כך וכך post\_send מחכה שיסתיימו כל העבודות (ההודעות יישלחו ע"י כרטיס הרשת של הלקוח (לאו דווקא יתקבלו ע"י השרת!)), אתם צריכים לקחת אותה בחשבון כדי לטפל נכון בסעיף הקודם.

הפורמט של ההדפסות צריכות להיות בדיוק כמו בתרגיל 1.

לגבי הכתיבה של הקוד ואיפה מריצים אותו:

פותחים שני טרמינלים, וכותבים באחד מהם:

ssh -l <CS\_username> -J <CS\_username>@bava.cs.huji.ac.il mlx-stud-01

ובשני:

ssh -l <CS\_username> -J <CS\_username>@bava.cs.huji.ac.il mlx-stud-02

מתחברים עם הOTP, כותבים סיסמא, ואתם בפנים. מנווטים לתיקיה שבה אתם רוצים לעבוד. עד כאן לגמרי גנרי, מכאן המלצה שלי על דרך לעבוד בנוחות:

תעבדו על הקוד במחשב שלכם, בעורך הקוד האהוב עליכם, כשאתם רוצים לנסות להריץ, תעתיקו את כל הקוד שלכם (ctrl+a, ctrl+c) עשו כך:

Cat >! bw.c // שומר לתוך קובץ את מה שתיכתבו לטרמינל (ודורס את התוכן הישן)

Ctrl+v // מעתיק את הקוד שלכם לטרמינל

Ctrl+d // שומר את הקובץ

gcc bw\_template.c -libverbs -o server

./server

עוברים לטרמינל השני, וכותבים:

./client mlx-stud-01

והקוד אמור לרוץ.

הערה אחרונה: שורת הקימפול שיש בקובץ תיאור התרגיל הרישמי (gcc bw\_template.c -libverbs -o server && ln -s server client) עושה שני דברים, גם מקמפלת ויוצרת קובץ בשם server, וגם יוצרת אליו קיצור דרך (softLink) בשם client. אם הקיצור דרך כבר קיים זה יתקע לכם את הפקודה השניה, אבל האמת היא שצריך לעשות את זה רק פעם אחת. ברגע שהקיצור דרך קיים, אפשר להריץ רק את הקימפול בלי יצירה של קיצור הדרך:

gcc bw\_template.c -libverbs -o server

בסוף בסוף, צריך לעשות קובץ .make שמקמפל הכל. ולארוז הכל לקובץ

<id1>\_<id2>.tgz

הנה לכם קובץ make מוכן:

<id1>\_<id2> = id1\_id2

all: server client

server: bw.c

gcc bw.c -libverbs -o server

client:

@ln -sf server client

clean:

rm -f client server

tar:

tar czvf $(<id1>\_<id2>).tgz bw.c Makefile

נראה לי שזהו. אנא אל תפציצו אותי בהודעות בפרטי, תשאלו בקבוצה, ואולי תקבלו תשובות גם מאחרים.