Functional Programing in Concurrent Distributed Systems

Final Project Report File

פרויקט 7

עיבוד מקבילי ומבוזר של Big Data מ-IMDB באמצעות אלגוריתם MapReduce

עומר לוכסמבורג 205500390

עילי נוריאל 312538580

# **תיאור המערכת**

# **מטרת הפרויקט**

שליטה בשפת ארלנג, עבודה בצורה מקבילית ומבוזרת וכן עבודה עם מידע מבוזר. בנוסף, מטרת הפרויקט היא להבין את אלגוריתם ה-Map-Reduce, שימומש על נתונים מתוך אתר [IMDb](http://www.imdb.com).

רעיון המערכת הוא שכל לקוח (client) יוכל להרכיב שאלה כללית אודות נתונים הרלוונטיים לסרטים מהאתר ולקבל תוצאה רלוונטית לפי הפרמטרים המבוקשים.

# **רעיון כללי**

מערכת IMDb Map-Reduce מחולקת לשלושה חלקים – מאסטר, שרתים, ולקוחות, כאשר המאסטר הוא המחבר בין כולם.

המאסטר מנהל את התקשורת מול השרתים ומול הלקוחות. תפקידו הוא לשמש כמוניטור על עליה ונפילה של שרתים – ובהתאם לכך שליחת המידע לביזור מחדש, וכן מענה ללקוחות בזמן אמת, באופן מקבילי, על ידי שליחת הבקשה לכל השרתים המחוברים למערכת.

השרתים מקבלים את המידע הרלוונטי אליהם ושומרים אותו במחשב המריץ את התהליך. כאשר תגיע שאילתה מהמאסטר, יוקצה תהליך שיממש אלגוריתם Map-Reduce רלוונטי על המידע המקומי. את התוצאה השרת ישלח למאסטר והמאסטר יאחד את המידע ויחזיר אותו ישירות ללקוח.

הלקוח, שולח את הבקשה ומקבל מענה. בנוסף למענה היבש לפי הבקשה ששלח, יתווספו עוד סטטיסטיקות רלוונטיות לתצוגת הלקוח.

# **מבנה המערכת**

מטרת הפרויקט הראשונית הייתה לפתח את המערכת הנ"ל בעזרת מערכת ה-[[1]](#footnote-1)Disco.

מאחר והמערכת לא פעלה, מימשנו מערכת דומה ואיכותית בשפת ארלנג בלבד המושתת על אותם קווי ייסוד:

כל לקוח מתקשר עם המאסטר, והמענה לו יתבצע באופן נפרד ומקבילי משאר הלקוחות. כך יתבצעו גם הבקשות שמתקבלות בשרתים, כדי לא ליצור צוואר בקבוק של מענה בודד ללקוח מתוך המאסטר.

המערכת פותחה בשפת ארלנג (Erlang 22, OTP22, erts- 10.7).

1. **מאסטר** – סינגלטון – מריץ את קובץ master.erl, ואת dataDistributor.erl, parse\_csv.erl. ממומש ב-gen\_server.

* אחראי על התקשורת בינו לבין הלקוחות בזמן אמת – מקבל שאילתה, מחזיר את המגולם.
* אחראי על התקשורת בינו לבין השרתים בזמן אמת – מעביר פקודות לביצוע, מקבל תשובה מכל אחד.
* מנהל את ביזור המידע הדינאמי – מתוך רשימת serverslist.txt יזהה אילו שרתים מחוברים. אל שרתים אלו ישלח המידע. כאשר שרת מתוך הרשימה עולה, יבוצע ביזור מחדש של המידע באופן שווה. כאשר שרת המחזיק מידע נופל, המידע יבוזר מחדש כדי לא לאבד אף מידע המוחזר ללקוח. כל הביצוע הנ"ל קורה באופן אוטומטי ע"י קובץ input ידני serverslist.txt בלבד.

טיפול בהודעות – צד המאסטר

המאסטר מקבל הודעת query מהלקוח, ומעביר אותה הלאה. בנוסף המאסטר מקבל לתהליך שנוצר עבור שרת ספציפי את ההודעה Reply/table\_error – הודעה זו היא התשובה של השרת לשאילתה שהמאסטר שלח, את התשובה הזאת נשלח עם כל ההודעות מכלל השרתים ישירות ללקוח.

בנוסף המאסטר מקבל הודעות nodeup, nodedown הרלוונטיות רק ל-node-ים בשם server#. אם התקבלה הודעה כזו, המאסטר מבצע בהתאם ביזור מחדש של המידע לצורך שמירה על איזון בין המידע ומניעת איבוד מידע שאוחסן בשרת שנפל.

במידה ונפל שרת בזמן עבודה על שאילתה, כלומר בזמן שהמאסטר מחכה לתשובה ממנו, המאסטר יקבל הודעת nodedown בתהליך ייעודי לניטור שרת זה – שיעביר הודעה לתהליך הראשי כי אין עליו להמשיך לחכות לתשובה משרת זה – והמאסטר יחזיר ללקוח את התוצאות משאר השרתים.

1. **שרת (server)** – מרובה – מריץ את קובץ server.erl ואת mapreduce.erl. ממומש ב-gen\_server.

* אחראי על שמירת המידע הלוקלי במחשב המארח.
* אחראי על פיענוח שאילתה, וביצוע אלגוריתם Map-Reduce בהתאם לדרישה.
* אחראי על שליחת המידע חזרה בצורה קומפקטית ככל הניתן, אל המאסטר.
* שולח הודעות nodeup, nodedown למאסטר, לצורך ביזור דינאמי של המידע (כפי שפורט לעיל).

טיפול בהודעות – צד שרת

שרת מקבל את ההודעה store – הודעה זו נשלחת מהמאסטר לצורך שמירת המידע שבהודעה בתור מידע לוקלי במחשב המארח. לאחר שמירה זו השרת מוכן לקבל הודעות שאילתה לצורך ביצוע Map-Reduce.

השרת מקבל בנוסף הודעות query – הודעות מסוג זה שולחות שאילתה לביצוע. השרת פותח תהליך חדש המבצע את אלגוריתם ה-Map-Reduce הרלוונטי לפי השאילתה, על המידע המאוחסן במחשב המארח, ומחזיר תשובה ישר כאשר היא התקבלה.

1. **לקוח (client)** – מרובה – מריץ את קובץ wxclient.erl. ממומש בממשק GUI ע"י wxwidgets.

* ניהול התצוגה הגרפית והממשק למשתמש.
* אחראי על ניתוח השאילתה ושליחתה למאסטר.
* את המידע מציג באופן גנרי בחלון חדש, לפי הנתונים שהוחזרו.
* מציג עם התוצאות סטטיסטיקות נוספות אודות השאילתה.

טיפול בהודעות – צד לקוח

ללקוח הודעה הנשלח למאסטר – הודעת query. בעקבות הודעה זו נקבל את המידע הרלוונטי אל הלקוח, מהמאסטר. שאילתה זו מכילה מידע אודות הקטגוריה בה אנו מחפשים והערך שהמשתמש הכניס (string), בנוסף לכך היא מכילה את הקטגוריות הרצויות להצגה. לפני שליחת הבקשה, נוכל לקבוע את אופי מיון התוצאות כרצוננו.

במידה והמאסטר נפל טרם שליחת השאילתה, או במהלכה – הקליינט יקבל הודעת nodedown בתהליך ייעודי לניטור המאסטר – שיעביר הודעה לתהליך הראשי כי אין עליו להמשיך לחכות לתשובה וכי המאסטר לא זמין.

# **רצף התוכנית**

כאשר המערכת עולה (הכוונה למאסטר) המאסטר מבזר את המידע ההתחלתי המוצג כקובץ CSV לפי המחשבים הרשומים לו ברשימת *serverslists.txt,* אשר מחוברים לרשת. לאחר גילוי הסרברים, המאסטר מבזר את המידע (לאחר shuffle - כדי למנוע עומס על סרבר מסוים בלבד) ושולח לכל סרבר chunk-ים של מידע בגודל זהה. מאותה עת, כל פעם שיעלה או ייפול node של שרת, המערכת תבצע ביזור מחדש של המידע לצורך איזון ושלמות המידע.

כעת ניתן להעלות ולהוריד שרתים באופן נטול חריגות. לצורך חיבור לקוח, נריץ את ממשק ה-GUI שרשום ב-wxclient.erl. מממשק זה, ניתן לבצע שאילתות ישירות למאסטר. את השאילתות המאסטר יעביר לכל השרתים הזמינים, וימתין להודעה. בזמן ההמתנה להודעה, המאסטר ממשיך לקבל בקשות באופן בלתי תלוי (לפי יכולת המעבד המריץ אותו).

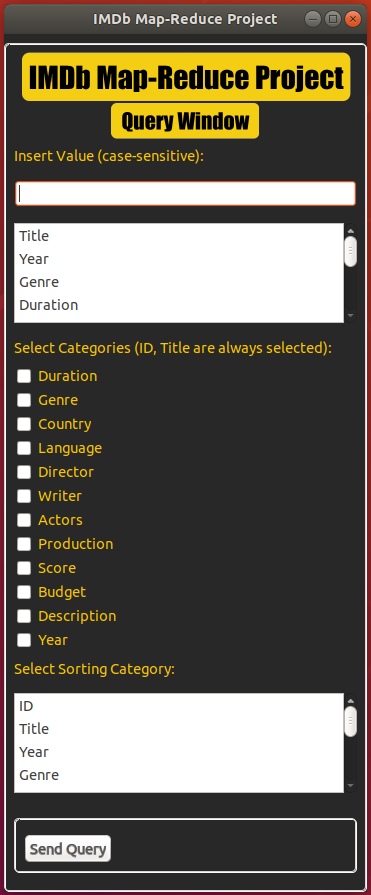
כל שאילתה תענה ללקוח בחלון חדש המציג את כל הנתונים הרצויים. סגירת חלון הממשק, תהווה ניתוק של הלקוח באופן אוטומטי מהמאסטר.

# **ממשק המשתמש**

ממשק המשתמש בנוי משני חלונות.

הראשון הוא חלון השאילתה (Query Window) בו המשתמש ממלא שאילתות הנשלחות אל המאסטר. חלון זה בעל מנגנון תקינות הקלט, על מנת שחישובים שיעשו בהמשך האלגוריתם לא יחזירו חריגה.

החלון השני הוא חלון התוצאות. בחלון זה מרוכז כל המידע אודות הסרטים העונים על השאילתה שנשלחה. בנוסף לכך נתונים עוד סטטיסטיקות שונות בנוגע למידע המוחזר.

1. **חלון בקשת השאילתה:**

שליחת השאילתה

בחירת קטגוריה לפיה ימוינו הרשומות בחלון התוצאות

בחירת המידע שיוצג בחלון התוצאות על כל הסרטים התואמים לחיפוש

הקטגוריה של הערך אותו מחפשים

הכנסת הערך לפיו מחפשים מידע

# **חלון התוצאות**



נתונים על השאילתה, שהוכנסה על ידי המשתמש

נתונים סטטיסטיים כלליים על ביצוע השאילתה וכן נתונים נוספים על תוצאות החיפוש

טבלה המציגה את תוצאות החיפוש ממוינים לפי השדה שנבחר (מהקטן לגדול) – כאן ניתן לראות שהמידע מסודר לפי Year.

# **מבני נתונים**

אנו משתמשים בשמירת המידע ע"י שמירת טבלת ets לקובץ. שימוש זה מאפשר לנו לשלוף ערכים כמשתני ארלנג שהוגדרו בביזור המידע באופן יעיל ומהיר. כל chunk של טבלה נשמר במחשב המארח של השרת, וכאשר יידרש, השרת ישלוף את מידע זה וייבצע עליו פעולות.

שימוש זה אמנם שומר בזיכרון המחשב המארח את הטבלה, אך מפני שכל תהליך המבצע את אלגוריתם Map-Reduce יצטרך להשתמש בטבלה זו באופן בלתי תלוי בשרת עצמו, העתקה הייתה מבצעת בכל מקרה. לכן, כתוצאה מכך, בחרנו לשמור את הנתונים כ-hard copy, על מנת לא להכביד על תהליך שרת ה-gen\_server, בשמירת הטבלה ב-server\_state וכן לחסוך זיכרון מזיכרון ה-RAM.

בכל טבלה יופיעו **רשומות של סרטים** המכילים את המידע הבא (מוגדר כ-record בארלנג):

movie\_data = {id, title, original\_title, year, date\_published, genre, duration, country, language, director, writer, production\_company, actors, description, avg\_vote, votes, budget, usa\_gross\_income, worlwide\_gross\_income, metascore, reviews\_from\_users, eviews\_from\_critics}

מאחר וכל רשומה מכילה הרבה פרמטרים, השימוש הנ"ל עוזר לנו בניתוח המידע באופן ישיר. את המידע הזה אנו שולחים מהשרתים למאסטר ומהמאסטר ללקוח.

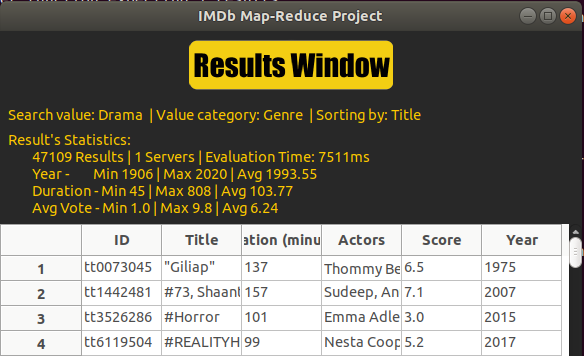
את **השאילתות** אנו שולחים במערכת ב-record אחר המכיל את המידע הבא:

query = {type, searchVal, searchCategory, resultCategory}

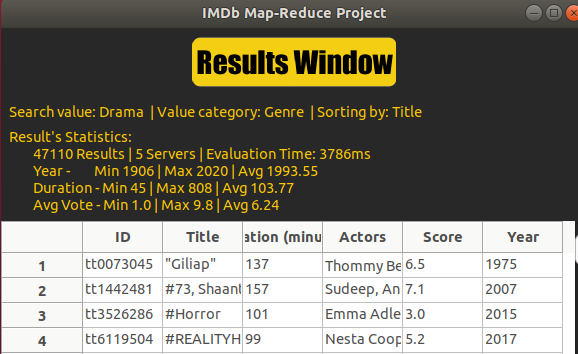
כאשר ה-type נועד לשם להרחיב את אופציות המענה לשאילתות, searchVal הוא הערך המוזן מהמשתמש כ-string, searchCategory הוא הפרמטר אותו אנו מחפשים (אותו חיפשנו ע"י ה-string) ולבסוף resultCategory הוא למעשה מסוג movie\_data המציג לנו במערך אינידקציה (true/false) אילו קטגוריות לשלוח בחזרה.

# **ביצועים**

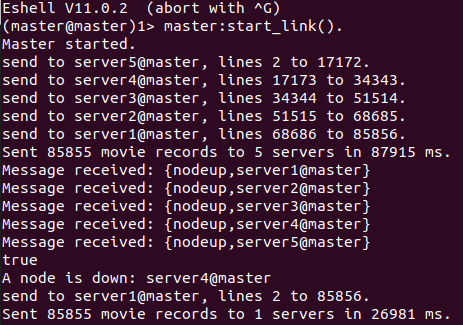
ביצועי המערכת נמדדים בזמן הערכת המידע. יש לשים לב שמכיוון שאנו משתמשים בממשק המציג את **כל** המידע, זמן ייקח עד שהוא יוצג לפנינו כטקסט.

* בדקנו את המערכת כאשר לרשותך 5 שרתים מול שרת בודד, התוצאות מוצגות כאן:

ניתן לראות כי עבור ביצוע של שרת בודד קיבלנו תוצאות גרועות. עבור 47k תוצאות לקח למערכת 7.5 שניות ומעלה לעבד את המידע.

 זמן השגת המידע

לעומת זאת, עבור 5 שרתים, ניתן לראות כי הביצועים נחתכו לפחות מ-4 שניות. למרות שהשיפור אינו לינארי בכמות השרתים, ניתן לראות כי הדבר אפקטיבי.

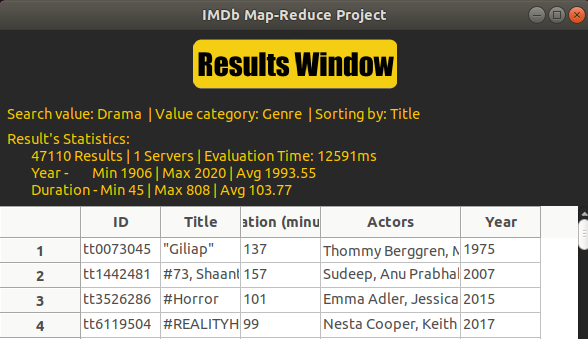
* כאשר שרת מתנתק או מתחבר, המאסטר מזהה זאת ומבצע ביזור מידי מחדש:

בדוגמה הבאה, חיברנו 5 שרתים אל המאסטר ולאחר מכן ניתקנו 4 מהם וקיבלנו שינוי במאסטר.

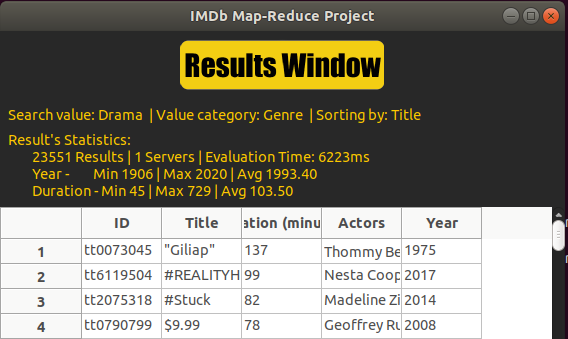
רישום מידע ל-5 השרתים

שליחת המידע לשרת הבודד שנשאר

* כאשר שרת מתנתק בזמן בקשת תשובה, המאסטר מחזיר ללקוח תשובה רק מהשרתים שהוא הספיק להשיג מהם את המידע. לאחר מכן, המאסטר יהיה לא זמין, לשם ביזור מחדש – כך לא נאבד מידע ונוכל כבר בתשאול הבא לקבל את כלל התוצאות.



כאן ניתן לראות שבעבור שרת אחד שמכיל את כל המידע, חזרו כ-47k תוצאות.



לעומת זאת, עבור המקרה המתואר, עבדנו עם 2 שרתים בזמן ביצוע השאילתה. כאשר שלחנו אותה, הפלנו את אחד השרתים והמתנו לתשובה. סך התשובות הן כ-23k וזאת בגלל שאלו התשובות שהגיעו רק משרת בודד.

בדוגמה הבאה, חיברנו 5 שרתים אל המאסטר ולאחר מכן ניתקנו 4 מהם וקיבלנו שינוי במאסטר.

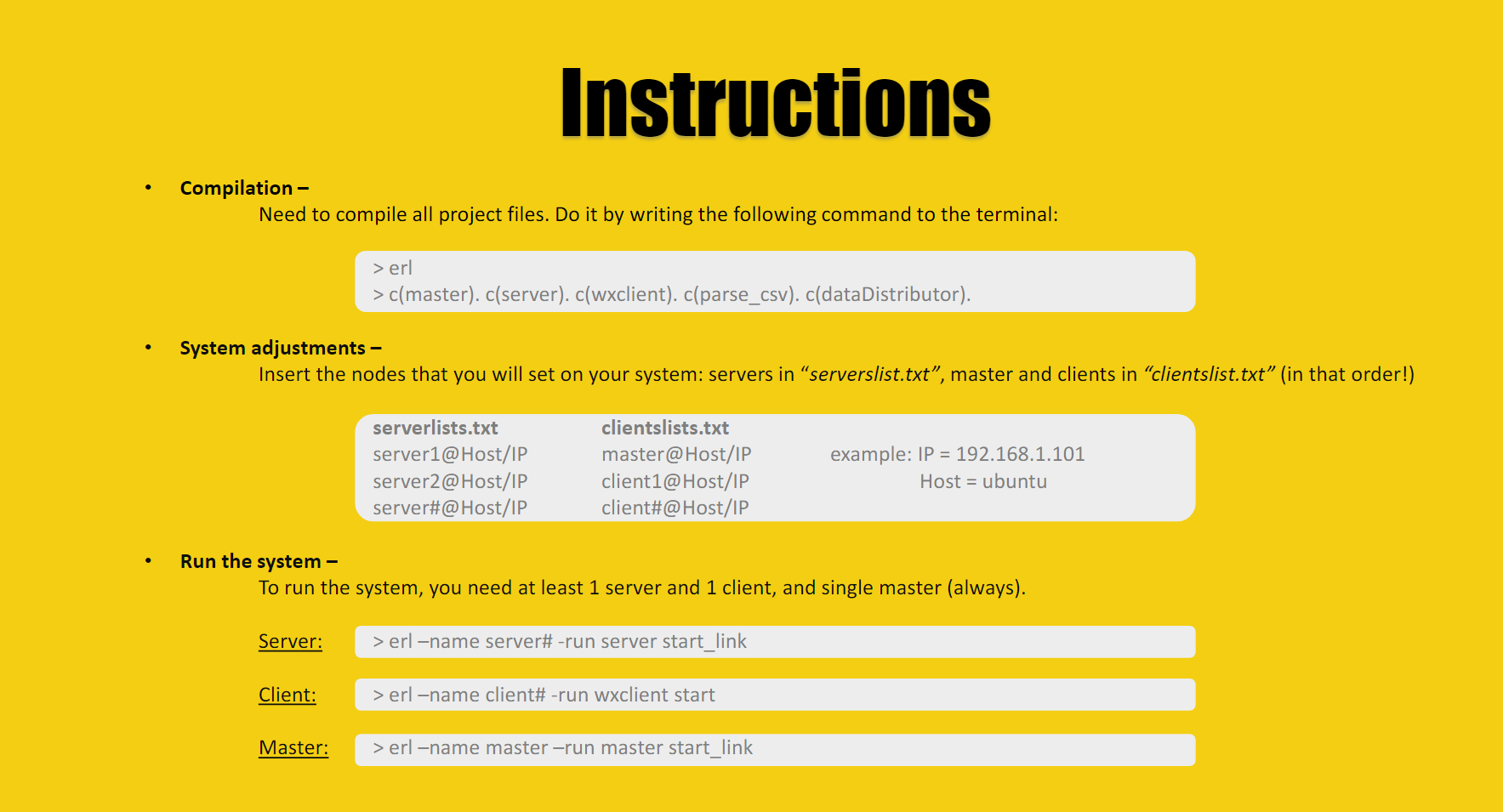
# **מסקנות**

* כאשר מפתחים מערכת מבוזרת ומקבילית חשוב לתכנן מראש את הארכיטקטורה בצורה מפורטת שתמפה עבורנו את כלל הבעיות הנפרסות בפנינו.
* חשוב לשים לב למקרים בהם ייתכנו צווארי בקבוק במערכת –המאסטר למשל מנהל את כל התקשורת בין השרתים ובין הלקוחות, לכן חשוב לשים לב לביצוע הטיפול בבקשות בצורה יעילה ובלתי תלויה אחד בשני.
* גודל המידע ומספר השרתים לביזור משפיעים על תוצאות ביצוע השאילתה. ככל שנגדיל את מספר השרתים כך זמן התגובה יהיה קטן יותר. ( \*\*\*\* לצרף תמונה של הניסוי \*\*\*\* )

הערה: בגלל מגבלות טכניות לא יכלנו לבדוק את המערכת עם כמה סרברים הממוקמים במחשבים שונים, אך ע"פ התיאוריה הדבר היה נותן ביצועים מצויינים.

* ארלנג כשפת תכנות נותנת כלים לתכנות מקבילי ומבוזר בצורה קלה ויעילה, לכן על מנת ליצור מערכת מבוזרת המעבדת כמויות גדולות של מידע מכמה יעדים מרוחקים, כדאי להשתמש בה כשפת התכנות הראשית.

# **הוראות הרצה**



\*

\* for client you may need to open shell “erl –sname client”, and write “wxclient:start()”

# **נספחים**

* [סרטון הדגמה ב-YouTube](https://youtu.be/u_CJGBQcP_M)
* [קישור לפרויקט ב-GitHub](https://github.com/ilaynuriel/Project-Ilay-Omer)

1. מתוך: <http://discoproject.org/> [↑](#footnote-ref-1)