אוניברסיטת אריאל בשומרון

קורס מערכות מבוזרות – סמסטר א' 2020 מדעי המחשב

נכתב ע"י צבי מינץ
zvimints@gmail.com

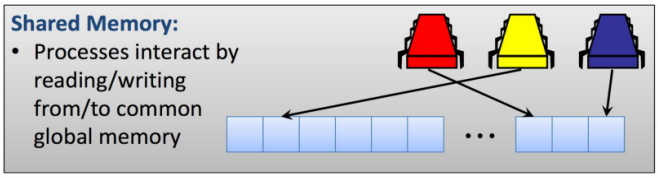
מערכות מבוזרות - Distributed System

אוסף של יחידות חישוב אשר יכולים לתקשר אחד עם השני
הערה: גם מחשב אחד הוא מבוזר, היות ויש כמה מעבדים וכמה תהליכים.

יתרונות של מערכות מבוזרות:

1. לשתף מידע בין קדקודים שונים
2. לטפל בכמות מידע גדולה יותר (במקום שכל המידע יהיה במקום כלשהו, אז ניתן להחזיק מידע בכל קודקוד)
3. עיבוד מקבילי (Parallelism)
4. Redundancy and Resiliency (אם מתוך 100 מכשירים אז 30 נפלו אז נבנה אותם ככה שהם יעבדו רק בעומס)
5. Scaling

מודלים למערכות מבוזרות



מערכות סינכרוניות: לכל המכשירים ישנו "קו שעון" משותף ולכן אפשר לדבר על סיבובים, שליחת ההודעות מתרחשת בתחילת הסיבוב ולקראת סוף הסיבוב ההודעה מגיעה. זמן הסיבוב ידוע מראש

במודל "Message Passing"

בראונד r : בזמן $r-1$, כל תהליך שולח הודעות. ההודעות מגיעות בזמן r

במודל "Shared Memory"

בכל ראונד, כל תהליך יכול לגשת לתא זיכרון בודד.

מערכות אסינכרוניות: אין "קו שעון", זמן הגעת ההודעה הוא מספר **סופי** כלשהו **לא ידוע** אשר יכול להשתנות בין הודעות.

במודל "Message Passing"

ההודעות תמיד מגיעות
זמן הגעת ההודעה הוא לא ידוע מראש ותמיד נסתכל על המקרה הגרוע ביותר

במודל "Shared Memory" (יש זיכרון אחד)

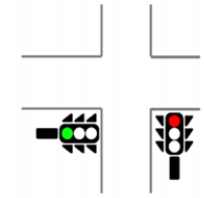
כל התהליכים בסופו של דבר יבצעו את הצעד הבא
זמן העיבוד לא ידוע מראש ותמיד נסתכל על המקרה הגרוע ביותר

תקלות Failures:

- **Crash Failure**
  קודקוד אשר מפסיק לעבוד בזמן מסוים, במערכת סינכרונית זה יכול לקרות באמצע השעון.
- **Byzantine Failure**
  קודקודיים שרוצים לפגוע בחישוב, "האקרים".
- **Omission Failure**
  קודקוד אשר מפסיק לעבוד לזמן מסויים
- **Resilience**
  התמודדות עם מספר מסויים של תקלות (מודל שמתמודד לדוגמא עם 50 תקלות כן ו51 לא)

תכונות נכונות של מערכת מבוזרות:

- Safety - שום דבר רע לא קורה ויקרה
  דוגמא: בכל נקודת זמן, לכל היותר רמזור אחד בצבע ירוק
  על מנת להוכיח Safety נוכיח בעזרת Invariant, המצב ההתחלתי היה בטוח, וכל מצב משאיר את המערכת בטוחה
- Liveness - משהו טוב יקרה למשהו
- דוגמא: יש תמיד אור ירוק אחד דלוק ברמזור
  דוגמא: במערכת אשר מספקת אוכל לאנשים, אז משהו מקבל אוכל.
  דוגמא: במערכת החלפת הקשר, אז משהו יכול לגשת למקור מידע.
- Fairness - משהו טוב יקרה לכולם
  דוגמא: במערכת אשר מספקת אוכל לאנשים, אז כולם מקבלים מספיק אוכל.
  דוגמא: במערכת החלפת הקשר, אז כל אחד יכול לגשת למקור מידע כל הזמן.
  דוגמא: שני האורות ירוקים ברמזור



קורס מערכות מבוזרות – סמסטר א' 2020 מדעי המחשב

נכתב ע"י צבי מינץ
zvimints@gmail.com

מערכת בסיסית מכילה:

1. n קודקודים אשר מיוצגים ב- $v_1, \dots, v_n$
באופן לא מפורש - Implicit יש n קודקודים ממוספרים מ-1 עד n כאשר n הוא מספר ידוע.
2. בכל זמן, לכל קודקוד v_i יש מצב פנימי Q_i

Schedule: רצף של שליחות וקבלות המהוות את האירועים במערכת.
Admissible Schedule: רצף הגיוני, כלומר אין קבלת הודעה לפני שהיא נשלחת.

קונפיגורציה C זה וקטור של n כניסות אשר מתאר את מצב המערכת.
כאשר לאחר סדרת אירועים נקבל את הריצה הבאה:

$$C_0, \Phi_1, C_1, \Phi_2, \dots$$

כאשר Φ_i הוא אירוע כלשהו.

***Local View* – מה כל קודקוד רואה מה-Schedule מסויים.**
בהינתן מתזמן S נגדיר $S|i$ להיות נקודת המבט של קודקוד v_i לאורך המערכת.
לדוגמא, בהינתן רצף $\boldsymbol{S = s_{13}, s_{23}, s_{31}, r_{13}, s_{32}, r_{31}, r_{23}, s_{13}, s_{21}, r_{31}, r_{12}, r_{32}}$

$$S|1 = s_{13}, r_{13}, s_{13}, r_{12}$$
$$S|2 = s_{23}, r_{23}, s_{21}$$
$$S|3 = s_{31}, s_{32}, r_{31}, r_{31}, r_{32}$$

אבחנה: $S|i$ רואה את כל מה ש-i בהתחלה

תרגיל לדוגמא
יהיו $p_1, \dots, p_n$ קודקודים במערכת שליחת הודעות המריצים את האלגוריתם הבא:
אם $i = 1$:
שלח '1' ל- p_2
המתן לקבל הודעה ממנו
אם קיבלת ≤ 5 החזר 1
אחרת, החזר 0.
אחרת:
המתן לקבל הודעה מ– p_{i-1}
אם $i = n$ דלג על השלבים הבאים (תהליך אחרון)
שלח הודעה ל – p_{i+1}
המתן לקבל הודעה מ – p_{i+1}
שלח אותה ל– p_{i-1} בתוספת 1
אם המתנת ל – p_{i+1} יותר מ-$2n \cdot T$ יחידות זמן, החזר 0 ושלח ל – p_{i-1} את '1', אחרת החזר '1'
עבור השאלה הזו: (מערכת סינכרונית)
Safety – קודקוד p_1 מחזיר '1'
Liveness – משהו שאינו p_1 מחזיר '1'
Fairness – כולם מחזירים '1'
זמן הסיבוב – T

מודל הרצה לאלגוריתם:

1. מערכת סינכרונית, הודעה אחת הולכת לאיבוד.

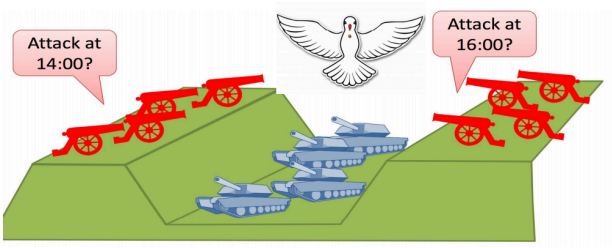
Safety – אם הודעה אבדה בדרך מהקודקוד הראשון לקודקוד החמישי אז הקודקוד הראשון יקבל > 5 ולכן ואין *Safety*
Liveness – אם ההודעה מהקודקוד הראשון לשני אבדה אז כולם תקועים
הערה: אם *Liveness* לא מתקיים אז *Fairness* לא מתקיים

2. מערכת אסינכרונית – כל ההודעות מגיעות.

Safety – לא, כי אם הטיימר של p_2 פקע , אז הוא ישלח לקודקוד הראשון '1' ו- $1 < 5$
Liveness – גם אם הטיימרים של כולם יפקעו אז p_n לא יחכה ולכן יחזיר 1.
Fairness – כמו *Safety*

אוניברסיטת אריאל בשומרון

קורס מערכות מבוזרות – סמסטר א' 2020 מדעי המחשב
נכתב ע"י צבי מינץ
zvimints@gmail.com
בעיית 2 הגנרלים



מודל: שני יחידות חישוב דטרמיניסטיות אשר מדברות במערכת סינכרונית מעל מערכת שליחת הודעות לא אמינה (הודעות יכולות ללכת לאיבוד)
קלט: כל יחידת חישוב מתחילה עם קלט מהקבוצה $\{0,1\}$
פלט: כל אחד מיחידות החישוב צריך להוציא פלט $\{0,1\}$
הסכמה: שני יחידות החישוב צריכות להסכים על **<u>אותו</u>** פלט
נכונות: אם הקלט זהה אז הפלט צריך להיות זהה לקלט במידה והודעות לא הולכות לאיבוד, אחרת, להחזיר 0 או 1 מוסכמים.
תקלות: הודעה יכולה ללכת לאיבוד
סיום: שני יחידות החישוב צריכים להסכים **בזמן סופי**

אבחנה: ללא נכונות, אפשר לבצע אלגוריתם פשוט אשר הינו 'החזר 0'
$\Downarrow$
הבעיה אינה פתירה
שיטת הוכחה: להניח שקיים אלגוריתם ולהראות סדרת הרצה הזהות זו לזו שתביא לסתירה
סימונים:
הרצה E לא ניתנת להבדל מהרצה E' עבור קודקוד v אם v רואה את אותו הקלט ואת אותם ההודעות ב-E ו-E'
אם E לא ניתנת להבדל עבור קודקוד v, אזי v מבצע את אותה סדרה של פעולות (בגלל שזה דטרמניסטי) בשני ההרצאות, ולכן נסמן $E|v = E'|v$ או לחלופין $E \sim_v E'$
אבחנה: אם נשקול סדרה של הרצאות $E_0, E_1, E_2, \dots E_k$ כך ש- $E_{i-1} \sim_v E_i$ עבור $i \in \{1, \dots, k\}$ עבור קודקוד v כלשהו
אזי מתקיים כי כל הקודקודיים מוצאים אותו פלט בכל ההרצאות (בגלל שזה דטרמניסטי)
הוכחה:
נניח בשלילה כי קיים אלגוריתם סופי דטרמיניסטי הפותר את הבעיה ב-T סיבובים
נשקול את רצף ההרצות *Executions* הבא:

Nodes always decide after exactly T rounds

Execution E_0**:** Both inputs are 0, no messages are lost.

Execution E_1**:** One of the messages in round T is lost.

...

Execution E_{2i}**:** Both messages in round T+1-i are lost.

Execution E_{2i+1}**:** One of the messages in round T-i is lost.

...

Execution E_{2T}**:** Both inputs are 0, no messages are delivered. All outputs are 0 due to similarity.

Execution E_{2T}**:** Both inputs are 0, no messages are delivered. All outputs are 0 due to similarity.

Execution E_{2T+1}**:** Input of v_1 is 0 but input of v_2 is 1. No messages are delivered.

Execution E_{2T+2}**:** Both inputs are 1, no messages are delivered.

...

Execution $E_{2T+2i+1}$**:** Exactly one of the messages in round i is delivered.

Execution $E_{2T+2i+2}$**:** Both messages in round i are delivered.

...

Execution E_{4T+2}**:** Both messages in round T are delivered. Decision must be 1 - a contradiction.

נכתב ע"י צבי מינץ
zvimints@gmail.com
לסיכום:

- We start with an execution in which both nodes have input 0 and no messages are lost: by validity both nodes must decide 0.
- We remove messages one by one to obtain a sequence of executions such that consecutive executions are similar.
- From an execution with no messages delivered and both inputs 0, we can get to an execution with no messages delivered and both inputs 1 (in two steps).
- By adding back messages one by one, we obtain an execution in which both nodes have input 1 and no messages are lost: by validity both nodes must decide 1 $\Rightarrow$ contradiction!
- Not hard to generalize to an arbitrary number $n>1$ of nodes.