אילן הינדי

[הקלד את תקציר המסמך כאן. התקציר הוא בדרך כלל סיכום קצר של תוכן המסמך. הקלד את תקציר המסמך כאן. התקציר הוא בדרך כלל סיכום קצר של תוכן המסמך.]

אלגוריתמים מבוזרים

פרק 3 : Snapshots

פרק 3 : Snapshots

תוכן עניינים

[1 הגדרות 2](#_Toc515795737)

[2 שימושים 2](#_Toc515795738)

[3 בעיות 2](#_Toc515795739)

[4 תנאים ל-consistent snapshot 3](#_Toc515795740)

[5 אלגוריתם Lai-Yang 3](#_Toc515795741)

[5.1 דרישות 3](#_Toc515795742)

[5.2 הרעיון המרכזי של האלגוריתם 3](#_Toc515795743)

[5.3 בעיות באלגוריתם 4](#_Toc515795744)

[5.4 פתרון לבעיות 4](#_Toc515795745)

[5.5 ניהול רשימת ההודעות 4](#_Toc515795746)

[5.6 Pseudo code 5](#_Toc515795747)

[5.7 דוגמא 7](#_Toc515795748)

פרק 3 : Snapshots

# הגדרות

Snapshot – מצב המשתנים הלוקליים בכל תהליך וההודעות בערוצים ב-configuration מסוים.

1. Basic message – הודעה של התהליך המרכזי
2. Control message- הודעה של תהליך ה-snapshot
3. Consistent snapshot- snapshot שמתאר קונפיגורציה של התוכנית (כלומר המצב שמתואר ב-snapshot מתואר גם בדיאגרמת המצבים של התוכנית)
4. Inconsistent snapshot – snapshot שמתאר קונפיגורציה שאינה קיימת של התוכנית (כלומר המצב שמתואר ב-snapshot לא מתואר בדיאגרמת המצבים של התוכנית)
5. Presnapshot- אירוע שקרה בתהליך לפני snapshot מקומי
6. Postsnapshot- אירוע שקרה בתהליך אחרי snapshot מקומי

# שימושים

לבחינת תכונות חיצוניות לתהליכים שנשארות נכונות מהרגע שהן נעשות נכונות לדוגמא:

* Deadlock
* Termination
* Garbage
* Restart after failure
* Debugging

# בעיות

נניח שתהליך מעונין לקבל את הסטטוס של הביצוע של אלגוריתם מבוזר הוא צריך לבקש מכל שאר התהליכים לקחת Snapshot מקומי. בנוסף תהליכים צריכים לחשב את המצב בערוצים של הודעות במעבר ברגע הלוגי של ה – snapshot.

הסיבוך הוא שתהליכים לוקחים snapshot מקומי בזמנים שונים לכן snapshot לא בהכרח מציין קונפיגורציה של הביצוע.

1. להבטיח שה-snapshot יהיה של configuration חוקי. כלומר שלא יוצר מצב שבגלל שהחישוב של ה-snapshot מופץ ברשת ולא מתרחש באותו זמן בכל התהליכים לא יקרה מצב שה-snapshots המקומיים לא מרכיבים קונפיגורציה לא קיימת
2. לפתח אלגוריתם שלא יפריע לאלגוריתם שעובד
3. לפתח אלגורית שלא דורש הקפאה של התהליך המרכזי
4. נסתכל על המצב הבא בערוץ PQ:
   1. בתהליך השולח אירוע השליחה postsnapshot כלומר הסדר היה לקיחת snapshot ואח"כ שליחת ההודעה
   2. בתהליך המקבל אירוע הקבלה הוא presnapshot כלומר הסדר היה קודם קבלת ההודעה וא"כ לקיחת snapshot
   3. זה מצב לא הגיוני מאחר ומקבלים סדר כזה : קבלת ההודעה – ביצוע snapshot – שליחת ההודעה שזה סדר לא הגיוני.
5. נסתכל על מצב בערוץ PQ
   1. בתהליך השולח אירוע השליחה הוא presnapshot כלומר הודעה נשלחת לפני לקיחת ה - snapshot
   2. בתהליך המקבל אירוע הקבלה הוא postsnapshot כלומר ההודעה התקבלה אחרי ה-snapshot
   3. זה מצב לא הגיוני מאחר ומקבלים סדר כזה שליחת ההודעה – ביצוע ה – snapshot – קבלת ההודעה (יש להם פה איזה אי הבנה בספר מצב כזה הוא מצב שיש הודעה בערוץ יכול להיות שהם מתכוונים לכך שזה לא חוקי שיש הודעה בערוץ ואז אין קונפיגורציה חוקית ????)

# תנאים ל-consistent snapshot

1. כל אירוע שהוא presnapshot a מקיים שכל אירוע b שמקיים ) b ≺ aכלומר b לפני a ב-casual order) הוא גם presnapshot. תנאי זה מבטיח שהפרש בזמני הדיווח של התהליכים לא יגרום לכך שתהליך אחד "יתקדם" מעבר לתהליך אחר.
2. הודעה נכללת בערוץ ב-snapshot אם ורק אם בתהליך ששולח היא presnapshot ובתהליך המקבל היא postsnapshot

# אלגוריתם Lai-Yang

## דרישות

1. אין דרישה שהערוצים יהיו fifo
2. כל תהליך יכול ליזום את התחלת התהליך

## הרעיון המרכזי של האלגוריתם

1. הנתונים שתהליך צריך ל – snapshot (בכל אלגוריתם):
   1. את המצב הפנימי שלו
   2. את ההודעות שהיו ממוענות אליו לאחר שהתהליך בצד השני של הערוץ ביצע snapshot ועדין לא הגיעו
2. ע"מ לממש את סעיף 1.2 מאחר ואין סדר בערוץ התהליך השולח צריך להפריד בין הודעות שנשלחו לפני שהוא ביצע snapshot (ושהתהליך המקבל צריך לכלול אותן ב – snapshot המקומי) לבין ההודעות שהתהליך השולח שלח לאחר שהוא ביצע snapshot ושאותן (ואת ההשפעה שלהן) צריך לכלול ב – snapshot הבא
3. ע"מ לבצע את ההפרדה הזאת התהליך השולח מוסיף false להודעות שהוא שלח לפני שהוא ביצע snapshot ו – true להודעות אחרי שהוא ביצע snapshot
4. כל עוד מתקבלות הודעות עם סיומת false התהליך המקבל מתנהג כאילו אין בקשה ל – snapshot במערכת
5. ברגע שהתהליך המקבל מקבל הודעה עם true הוא יודע שהתהליך השולח ביצע snapshot ולכן הוא מבצע snapshot. לאחר שהוא ביצע snapshot הוא מייצר רשימה של כל התהליכים עם סיומת false שהגיעו לאחר ביצוע ה – snapshot ע"מ להוסיף אותן ל – snapshot כהודעות בערוצים בזמן שהיה snapshot
6. כל עוד לא הסתיים השלב ההודעות עם סיומת true נצברות ולא מעובדות
7. כאשר כל ההודעות עם סיומתfalse הממוענות אליו הגיעו התהליך המקבל יודע שהסתיים שלב

## בעיות באלגוריתם

קיימות 2 בעיות באלגוריתם כפי שהוא פורט בסעיף הקודם

1. נניח שלאחר ביצוע ה – snapshot התהליך השולח לא שולח עוד הודעות. במקרה כזה התהליך המקבל לא יבצע snapshot
2. איך התהליך המקבל ידע שהוא סיים את השלב (כלומר שהגיעו אליו כל ההודעות עם סיומת false שנשלחו אליו

## פתרון לבעיות

1. כאשר תהליך מבצע snapshot הוא שולחcontrol message רשימה של כל ההודעות עם סיומת false שהוא שלח בערוץ.
2. כאשר התהליך המקבל מקבל control message הוא מבצע snapshot (אם הוא לא ביצע) ויודע כמה הודעות הוא אמור לקבל בערוץ וכך הוא יודע מתי לסיים (ע"מ לסיים צריכים להתקיים 2 תנאים : הגיעו הודעות control מכל השכנים, כל ההודעות עם סיומת false הגיעו)

## ניהול רשימת ההודעות

1. לפני שהתהליך מקבל snapshot התהליך שומר את כל ההודעות עם סיומת false שהוא קיבל
2. לאחר שהתהליך ביצע snapshot הוא מתחיל רשימה שניה של כל ההודעות עם סיומת false שהגיעו לאחר ביצוע ה – snapshot (הודעות בערוצים). כמו כן הוא מתחיל רשימה של ההודעות עם סיומת true שהוא שומר ע"מ לבצע כאשר מסתיים השלב
3. כאשר התהליך מקבל control message הוא יוצר רשימה שלישית של הודעות שהוא אמור לקבל ע"מ לסיים שלב. התהליך מוריד מרשימה זו את ההודעות שהוא כבר קיבל בסעיפים 1,2
4. כאשר מסתיים השלב התהליך מבצע את ההודעות שהגיעו עם סיומת true שהוא שמר. את ההודעות שהן התוצר של העיבוד הוא שולח עם סיומת false

## Pseudo code

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Messages | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | **(m, b)** | | | | | | - | | | base algorithm message with the tag true or false | | | | | | | | | | | | | | | | | |  |
|  | **(presnp: l)** | | | | | | - | | | snapshot algorithm control message. The parameter is the number of messages (including the control message) that where sent from the source process with the tag false | | | | | | | | | | | | | | | | | |  |
|  |  | | | | | | | | | | | | | | | |  | |  | | | | | | | | |  |
| Variables | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | recordedP | | | | | | | | - | | | true if P took local snapshot | | | | | | | | | | | | | | | |  |
|  | sentP[PQ] | | | | | | | | - | | | The number of messages P sent to Q before taking its local snapshot (variable of the sending process) | | | | | | | | | | | | | | | |  |
|  | arrivedP[QP] | | | | | | | | - | | | The total number of messages arrived with false tag from Q to P | | | | | | | | | | | | | | | |  |
|  | expectedP[QP] | | | | | | | | - | | | The number of messages Q sent to P before taking its local snapshot (variable of the receiving process) (This parameter is sent by Q in it's control message) | | | | | | | | | | | | | | | |  |
|  | stateP[QP] | | | | | | | | - | | | A list of false messages arrived to P from Q after P took a local snapshot | | | | | | | | | | | | | | | |  |
|  |  | | | | | | | |  | | |  | | | | | | | | | | | | | | | |  |
| Initialize | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | recordedP | | | | | | | ← | | | | false | | | | | | | | | | | | | | | |  |
|  | sentP[PQ] | | | | | | | ← | | | | 0 for all outgoing channel c | | | | | | | | | | | | | | | |  |
|  | arrivedP[QP] | | | | | | | ← | | | | 0 for all outgoing channel c | | | | | | | | | | | | | | | |  |
|  | expectedP[QP] | | | | | | | ← | | | | ∅ for all outgoing channel c | | | | | | | | | | | | | | | |  |
|  | stateP[QP] | | | | | | | ← | | | | ∅ for all outgoing channel c | | | | | | | | | | | | | | | |  |
|  | **if** P wants to initiate snapshot | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | perform procedure TakeSnapshot | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **end if** | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  |  | | | | |  | |  |  |  |  | |  | |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **if P sends a basic message (m) into outgoing channel c0** | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | send **(m, recordedP)** into **c0** | | | | | | | | | | | | | | | | | | | | | | |  | | |  |
|  | **if recordedP** = false | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | | **sentP[c0] ← sentP[c0]** + 1 | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **end if** | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **if P receives a basic message (m, b) through incoming channel** | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **if b** = true | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | | perform procedure TakeSnapshot | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **else** | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | | **arrivedP[c0] ← arrivedP[c0]** + 1 | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | | **if recordedP** = true | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | |  | | **stateP[c0]** ← **stateP[c0]** ∪ {m} | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | |  | | **if** **arrivedP[c0]** +1 = **expectedP[c0]** for all incomming | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | |  | |  | |  | |  | |  |  |  |  |  |  |  |  |  |  | channels | | | | | |  |
|  | |  | |  | |  | | Terminate | | | | | | | | | | | | | | | | | |  | |
|  | |  | |  | | **end if** | | | | | | | | | | | | | | | | | | | |  | |
|  | |  | | **end if** | | | | | | | | | | | | | | | | | | | | | |  | |
|  | | **end if** | | | | | | | | | | | | | | | | | | | | | | | |  | |
|  | |  | |  | |  | |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
|  | | **if P receives (presnp, l) through an incomming channel c0** | | | | | | | | | | | | | | | | | | | | | | | |  | |
|  | | expectedP[c0] ← l | | | | | | | | | | | | | | | | | | | | | | | |  | |
|  | **if arrivedP[c0]** +1 **= expectedP[c**0**]** for all incomming channels | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | | Terminate | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **end if** | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | |  | |  | |  | |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |
|  | **procedure TakeSnapshot** | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **if** **recordedP** = false | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | | **recordedP**← true | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | | send **(presnp, sentP[c]+1)** into all ougoing channels c | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **end if** | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | | | | | | | | | | | | | | | | | | | | | | | | | |  |

## דוגמא

