אוניברסיטת חיפה
החוג למדעי המחשב
מערכות הפעלה – תרגול

תרגול מס׳ 10

Network File System (NFS)

1 מבנה מערכת NFS

המטרות שהוצגו ע״י מתכנני המערכת היו:

1. חוסר תלות בארכיטקטורת מחשב ומערכת הפעלה – הגדרת מערכת שתהיה ברת מימוש על ארכיטקטורות שונות ומערכות הפעלה שונות. המערכת צריכה להיות כזו שתאפשר ליצור סביבה הטרוגנית (מחשבים עם ארכיטקטורות שונות ומערכות הפעלה שונות) בה שרת יוכל לתת אותם שירותים ללקוחות מסוגים שונים (שאת UNIX נותן שירותים ללקוחות Windows).
2. התאוששות מתקלות – המערכת במקרה של תקלה (נפילת שרת או בעיות תקשורת) תוכל להתאושש בצורה פשוטה ומהירה.
3. שקיפות – המערכת תאפשר לתוכנית גישה לקבצים מרוחקים באותה צורה בה היא ניגשת לקבצים לוקליים. השקיפות צריכה להיות ברמה כזו שהתוכנית לא תוכל להבדיל בין קובץ לוקלי לקובץ מרוחק.
4. שמירה על סמנטיקה של UNIX (פעולת קריאה מחזירה את הערך האחרון שנכתב) – היות ואנו דורשים שהמערכת תהיה שקופה לתוכניות, אנו צריכים להבטיח שאותה סמנטיקה על קבצים לוקליים תמומש עבור קבצים מרוחקים. היות וייתכן מצב בו הן השרת והן הלקוח יהיו מחשבי UNIX ובמערכת UNIX מובטח עבור קבצים לוקליים סמנטיקה של UNIX – גם עבור קבצים מרוחקים יש לממש סמנטיקה זהה.
5. ביצועים סבירים – כדי שהמערכת תהיה נפוצה ושאנשים ישתמשו בה עלינו להבטיח שביצועי המערכת יהיו טובים (או לפחות דומים) למערכות הקיימות. עלינו להבטיח גם שגישה לקובץ מרוחק יחסית לגישה לקובץ לוקלי תהיה סבירה.

1.1 סיווג שרתים

ניתן לסווג שרתים לשני סוגים: Stateless Servers ו-State Servers.

1.1.1 Stateless Servers

שרתים ללא מצב. שרתים אלו אינם שומרים נתונים לגבי לקוחות המחוברים אליהם. היות והשרת אינו שומר מצב של לקוחות המחוברים אליו, על כל פעולה שהוא מקבל לכלול את כל המידע הדרוש לביצוע.

יתרונות

התאוששות פשוטה מתקלות – אם שרת נופל הלקוח יכול להמשיך לשלוח הודעות לשרת עד שיקבל תגובה (כאשר השרת יפעל שוב) היות ואין הוא מחזיק מידע על לקוחות שמחוברים אליו. מאחר והבקשה כוללת את כל המידע הדרוש לביצועה, השרת יוכל לענות לבקשה מיידית. אם הלקוח נופל השרת לא צריך לדעת מכך היות ואין הוא שומר מידע על הלקוח.

חסרונות

על כל בקשה הנשלחת ברשת לכלול את כל המידע הדרוש לביצועה ולכן גודלה של כל בקשה יחסית גדול.

State Servers 1.1.2

שרתים עם מצב. שרתים אלו שומרים נתונים לגבי לקוחות המחוברים אליהם. היות והשרת שומר מידע לגבי הלקוחות שמחוברים אליו הבקשות (בדרך כלל) יכללו פחות נתונים.

יתרונות

היות והשרת שומר נתונים לגבי לקוחות – גודל הבקשות הנשלחות על פני הרשת קטן יחסית.

חסרונות

התאוששות מסובכת מתקלות. אם שרת נופל הלקוח צריך לזהות שהשרת נפל כדי שכשהוא יעלה שוב הוא ישלח את המידע שאבד (מידע שהיה אצל השרת בעת הנפילה). אם לקוח נופל השרת צריך לזהות זאת כדי שיוכל למחוק את כל המידע השמור עבור לקוח זה.

- היות ואחת המטרות שהציבו להם מתכנני המערכת היתה התאוששות פשוטה מתקלות, שרתי NFS הינם Stateless Servers.

2 פרוטוקולים של מערכת NFS

אחת התכונות החשובות של מערכת NFS היא יכולת תמיכה בארכיטקטורות שונות של מחשבים ומערכות הפעלה שונות. כדי לממש את התכונה הנ"ל הפרידו את NFS לשני פרוטוקולים בין השרת והלקוח.

שני הפרוטוקולים מגדירים פעולות שלקוח יכול לשלוח לשרת. כל פעולה בפרוטוקולים מורכבת מ:

1. שם פעולה
2. פרמטרים
3. ערך מוחזר

מרבית הפעולות מקבלות כפרמטר / מחזירות ערך מסוג fhandle.

מהו אותו fhandle?

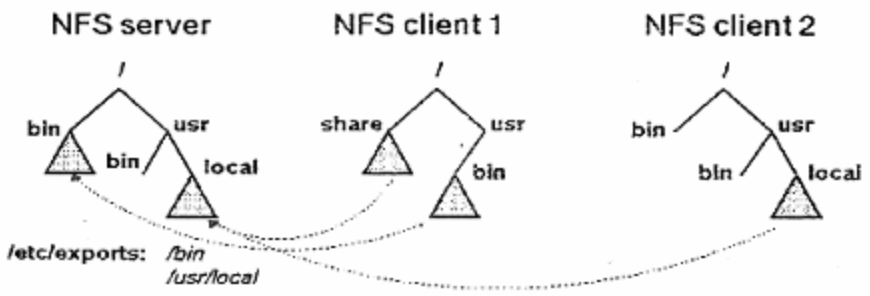
פעולות על קבצים לוקליים שלמדנו בתחילת הקורס משתמשות ב-File Descriptor שהינו אינדקס לתוך טבלת הקבצים של התהליך. בטבלה זו ישובים נתונים שמערכת הפעלה צריכה כדי לבצע פעולות

על קובץ. ה-fhandle מכיל נתונים מקבילים לנתונים אלו. מערכת ההפעלה שרצה על השרת זקוקה לנתונים אלו כדי לבצע את הפעולות על הקובץ. היות והשרת הינו Stateless Server (אין הוא שומר נתונים אצלו על קבצים שנפתחו ע"י לקוחות) ה-fhandle נשלח בין השרת ללקוח. לקוחות אינם קוראים את ה-fhandle ואינם משנים אותם – נתונים אלו הינם למעשה נתונים שהשרת זקוק להם שמרביתם היו נשמרים אצלו במידה והיה State Server.

2.1 פרוטוקול ראשון

פרוטוקול זה מטפל ב-mount של מדריך מרוחק. כזכור פקודת mount במחשב לוקלי מחברת File System נוסף לעת המדריכים תחת שם של ספריה. אנו רוצים שפקודת ה-mount המרוחקת תבצע את אותו הדבר – תחבר File System מרוחק לעץ המדריכים הלוקלי תחת שם של ספריה. מחשב הלקוח שולח בקשת mount ומסלול למדריך במחשב השרת. אם הבקשה היתה תקינה, השרת מחזיר fhandle בו מצויים נתונים של המדריך במערכת הקבצים המרוחקת.

2.1.1 דוגמא



<u>הערות</u>

- בכל Server נמצאת רשימה של כל המדריכים (אצלו) שהוא מרשה ל-Client לגשת אליהם.
- הרשימה נמצאת בקובץ לוקלי /etc/exports כך שכל המדריכים ניתנים לשימוש ברשת מיד עם אתחול המחשב הלוקלי (הדבר כולל גם את תתי ההיררכיות).
- אם ל-Client אין דיסק לוקלי (Diskless Station), יכולים לבצע mount לשורש מערכת הקבצים בזמן האתחול (boot). כך יווצר מצב שכל הקבצים מרוחקים, כולל קבצים מערכת ההפעלה.
- אם 2 או יותר מחשבי Client מחברים ע"י mount את אותו המדריך המרוחק למערכת הקבצים שלהם, הם יכולים לשתף קבצים במדריך זה ביניהם. למשל, תוכנית במחשב אחד יכולה לייצר קובץ מסוים ותוכנית במחשב אחר יכולה לקרוא מהקובץ.

2.2 פרוטוקול שני

פרוטוקול זה מגדיר את כל הפעולות עבור גישה לקבצים ומדריכים. לדוגמא:

- קריאה של מס' בתים מקובץ המוצבע ע"י fhandle מתבצעת ע"י הפקודה:

```
status NFS_READ(fhandle, count)
```

- כתיבה של נתונים לקובץ המוצבע ע"י fhandle מתבצעת ע"י הפקודה:

```
status NFS_WRITE(fhandle, data)
```

מרבית הפעולות המוגדרות על מערכת קבצים רגילה מוגדרות גם עבור קבצים מרוחקים. שני יוצאי דופן הם open ו-close. הסיבה לכך שפקודות open ו-close חסרות משמעות ב-NFS הינה משום שפעולות אלו אינן משנות את הקובץ (כפעולת write) או מחזירות נתונים על הקובץ (כפעולה read). פקודות אלו קיימות כדי לאתחל/לשחרר משתנים המאפשרים פניה לקוב. היות והשרת הינו Stateless Server משתנים אלו אינם נשמרים אצלו.

את פקודת open מחליפה פקודת LOOKUP: fhandle LOOKUP(fhandle_dir, name). פקודה זו מחזירה fhandle לקובץ בעל שם name המצוי בתוך ספריה המוצבעת ע"י fhandle_dir.

הסיבה להפרדה בין שני הפרוטוקולים היתה שפרוטוקול ה-mount דורש נתונים ספציפיים עבור מערכת ההפעלה, כגון: פענוח שם של שרת (כדי למצוא אותו ברשת), אימות זהותו של המשתמש ובדיקת הרשאות עבור משתמש.
ע"י הפרדה זו ניתן לשנות את פרוטוקול ה-mount מבלי לשנות את הפרוטוקול השני. למשל: שינוי מנגנון אימות זהות המשתמש או בדיקת ההרשאות של משתמש.

3 בעיות שהתגלו בעת מימוש NFS

1. הרשאות לקבצי ריצה – בעיה זו מתעוררת כאשר עובדים עם קבצי הרצה במערכת Demand Paging. בשיטה זו אין טוענים את כל התוכנית מיד אלא בקטעים עפ"י דרישה.
   בהרצת קובץ לוקלי – מערכת ההפעלה בודקת שלמשתמש יש הרשאת ריצה על הקובץ ואז מבצעת קריאה של תוכן הקובץ כדי להריץ אותו.
   בהרצת קובץ מרוחק – מערכת ההפעלה של ה-Client בודקת תחילה שלמשתמש יש הרשאת ריצה על הקובץ, ולאחר מכן תשלח בקשות קריאה של תוכן הקובץ אל השרת המרוחק. השרת המרוחק אינו יכול להבדיל בין קריאה רגילה של קובץ לבין קריאה הנוצרת עקב הרצה של קובץ.
   פתרון: השרת יאפשר למשתמש לקרוא את תוכן הקובץ אם יש לו הרשאת קריאה או הרצה.
2. פעולות שניתן לחזור עליהן ופעולות שלא ניתן לחזור עליהן: פעולות מהסוג הראשון הן פעולות שאם נבצע אותן מס' פעמים נקבל תמיד אותה תוצאה (לדוגמא קריאה). קיימות

פעולות NFS שלא ניתן לחזור עליהן (לדוגמא מחיקת קובץ). חלק מהפעולות שלא ניתן לחזור עליהן הם פעולות בעלי אופי הרסני במקרה ונבצע אותן מס׳ פעמים (לדוגמא פעולה קיצוץ אורכו של קובץ).

באילו מקרים יכול שרת לקבל את אותה בקשה לפעולה יותר מפעם אחת:

א. כאשר לקוח NFS מבקש פעולה מהשרת, הוא מאפשר לשרת לבצע את הפעולה במשך זמן t. אם הלקוח לא קיבל אישור על ביצוע הפעולה כעבור הזמן הזה הוא מניח שהפעולה אבדה ולכן יכול לבקשה שוב.

ב. בסיום הפעולה השרת שולח את תוצאת הפעולה אל הלקוח – ייתכן שהודעה זו תלך לאיבוד ואז הלקוח ישלח את הפעולה שוב.

היות שחלק מהפעולות בעלות אופי הרסני, מרבית המימושים של שרתי NFS משתמשים ב-Cache של פעולות אחרונות ותוצאתן – כאשר מתקבלת אצל השרת בקשה לפעולה פעם שניה, השרת שולח את תוצאת הפעולה הראשונה ובצורה זו אין הוא חוזר על הפעולה.

3. מנגנון נעילה של קבצים – במערכת ההפעלה UNIX קיים מנגנון של נעילת קבצים. תוכנית הפותחת קובץ יכולה להבטיח שאף תוכנית אחרת לא תשתמש באותו הקוב. ע״י כך שתנעל את הקובץ (התוכנית תשחרר את הקובץ כאשר היא תסיים להשתמש בו). מנגנון הנעילה של קבצים ממומש במערכת ההפעלה ע״י טבלאות פנימיות. אין אנו יכולים לממש בצורה טריוויאלית את מנגנון הנעילה על הקבצים בשרת ה-NFS היות ואין הוא שומר מידע על לקוחות המחוברים אליו.

4 שיפור ביצועים במערכות NFS

אחת המטרות שהיו למגדירי המערכת היתה ביצועים סבירים. כל פעולת NFS מורכבת למעשה משלוש פעולות:

1. התקשרות הלקוח אל השרת והעברת הבקשה.
2. ביצוע הבקשה אצל השרת.
3. התקשרות השרת אל הלקוח והעברת תוצאת הבקשה.

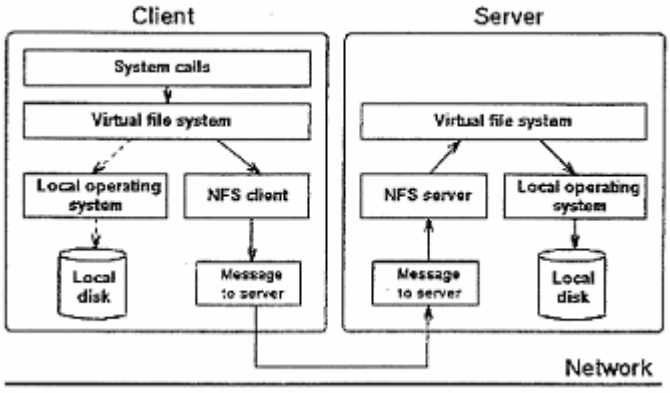
ברור אם כן שפעולת NFS איטית יותר מאשר פעולה לוקלית היות ופעולה לקולית דורשת רק את שלב 2. כל שיטת שיפור הביצועים משתמשות ב-Cache אצל הלקוח. שיפור הביצועים נובע מכך שחסכנו את התעבורה ברשת (שלבים 1, 3) עבור חלק מהפקודות.

1. מטמון fhandle – הרבה מאוד תוכניות המורצות זו אחר זו יוצרות קבצים אחת עבור השנייה (לדוגמא קומפיילר שיוצר בשלבי הביניים קבצים). פעולת ה-LOOKUP המחזירה fhandle עבור קובץ מרוחק מקבלת כל פעם שם של קובץ אחר לחפש בספריה. דבר זה גורר שלקוח המנסה לגשת לקובץ המצוי בשרת תחת מס׳ ספריות יצטרך לשלוח מס׳ פעולות LOOKUP. כדי לחסוך

בתקשורת זו מחזיקים לקוחות NFS מטמון של fhandle אחרונים בהם השתמשו. בצורה זו חוסכים בפעולות LOOKUP.

2. מטמון של קבצים זמניים – כשליש מהקבצים שנוצרים במערכת ההפעלה UNIX הם קבצים המשתמשים בהם פעם אחת. קבצים אלו בד״כ נוצרים ע״י תוכנית שמוחקת אותם בגמר השימוש. אם ניצור קבצים אלו תחילה אצל הלקוח ונמתין עם העתקה אל השרת מס׳ שניות (עד אשר התוכנית שיצרה אותם תגמור להשתמש בהם) נוכל לחסוך הרבה העתקות והיות וחלק ניכר מהקבים יימחקו.
3. מטמון של כתיבת – צוואר הבקבוק הגדול ביותר של מערכת NFS הינו ביצוע כתיבות לקובץ מרוחק. כדי לשפר ביצועים אלו, קיימים לקוחות NFS המממשים מטמון של כתיבות הנשמר אצל הלקוח ונכתב לשרת כל 30 שניות. דבר זה משפר את הביצועים היות וכתיבה אחת גדולה מהירה בהרבה מסדרת כתיבות קטנות. צורה זו של Cache אינה מבטיחה סמנטיקה של UNIX עפ״י הגדרה, אך הינה נפוצה היות והיא משפרת מאוד את הביצועים.

5 מימוש NFS ע"י SUN



המימוש מורכב מ-3 רמות:

1. הרמה העליונה היא קריאות מערכת. היא מטפלת בקריאות כמו open, read, write או close.
2. הרמה השנייה היא VFS או Virtual File System שמחזיקה ב-Client טבלה עבור על קובץ פתוח – טבלת v-nodes (Virtual I-Nodes).
   - לכל קובץ לקולי נשמר מדע רגיל (שם של התקן ו-I-Node).
   - לקובץ מרוחק נשמר מצביע ל-r-node שנמצא ברצה של NFS Client.
3. ברמה השלישית (NFS Client) נשמרים כל ה-r-nodes (Remote I-Node) של כל הקבצים הפתוחים (כאלו שבוצע להם LOOKUP) עם מידע על הקובץ כמו File Handle, מחוונים עבור קריאה או כתיבה וכתובת השרת. בפנייה לקובץ מרוחק מופעל פרוטוקול תקשורת עם השרת עליו יושב הקובץ ומועברת הבקשה לפעולה בצירוף ה-File Handle.