1. א. היתרון של מדיניות זו יהיה בגישה מהירה לדיסק ועדיפות גבוהה לתהליך הנוכחי שרץ ברקע, כלומר תהליך נוכחי שרץ ומבקש בקשה שמגיעה לדיסק מקבל מענה מיידי ומטופל מיד.

ב. החיסרון הוא אפשרות להרעבה.

במידה ויש תהליך שמבקש משאבים מהדיסק ומיד אחריו מגיע עוד תהליך שמבקש משאבים מהדיסק (ואם המערכת מוצפת בפניות היא יכולה לקבל עוד ועוד בקשות) הבקשה הראשונה תצטרך לחכות לכל הבקשות שהגיעו אחריה ומקבלות עדיפות מידית עליה – מה שיכול לקחת זמן רב.

1. למערך הדיסק RAID LEVEL 2 יתרון משום שהוא יכול לתקן שגיאה כאשר יש הפרעה שבעקבותיה מתקבלת שגיאה כך שלא ידוע באיזה מהדיסקים התרחשה ההפרעה, במקרה זה בגלל השימוש עם אלגוריתם קוד המינג שבעזרתו ניתן לבדוק את זוגיות הסיביות השונות בדיסקים וכך לגלות את השגיאה בדיסק מסוים ולהפוך את הסיבית. ניתן לתקן את המידע גם כשלא ידוע באיזה מהדיסקים התרחשה השגיאה.

לעומת זאת בRAID LEVEL 3 לא ניתן לדעת באיזה מהדיסקים נמצא החלק המשובש, ויכולים לתקן אותו רק כשמתבצעת קריסה בדיסק ולא הפרעה, או בהנחה שאנו יודעים איזו סיבית היא המשובשת, כלומר ניתן לזהות מידע משובש אך לא ניתן לדעת איפה – באיזה מהדיסקים המידע השתבש.

1. נבדוק את האפשרויות אחת אחת:

הקצאה רציפה – במדיניות זו כל קובץ מקבל הקצאה, אך במידה והקובץ גדל, תתבצע העתקה שלו למקום אחר בדיסק והקצאה מחדש.

בנוסף, במקרה מחיקה או הזזה יכולים להתקבל אזורים בזיכרון שלא יהיה ניתן להקצות בהם מקום, וכך עלול להתקבל ריסוק שיגרום לניצול גרוע של שטח הדיסק, ולכן פחות נעדיף את המדיניות הזו.

FAT – במדיניות זו נוכל להגדיר את גודל הבלוק לדוגמא כ-4KB ואז במקרה שלנו כל קובץ יתפוס בין בלוק ל1000 בלוקים, הבלוקים לא בהכרח יסודרו באופן רציף על הדיסק ולכן יכול להיות שהקריאה תהיה מעט איטית מהקצאה רציפה.

בנוסף במידה והקובץ מפוזר על הדיסק ונרצה לכתוב כמות גדולה ייתכן שהכתיבה תהיה מעט איטית.

נצטרך גם להגדיר מקום בדיסק לטבלת הFAT, ותהיה מגבלה על גודל הקובץ המירבי.

אבל היתרון במדיניות זו בא לידי ביטוי בהגדלה של הקובץ ללא פגיעה משמעותית בביצועים.

INODE – מבחינת ביצועים של הקריאה והכתיבה עבור קבצים קטנים הם יהיו דומים לFAT, אך עבור קבצים גדולים ייתכן שזמן הגישה יגדל והקריאות יהיו איטיות מFAT בגלל השימוש בindirect blocks.

יתרון של שימוש במדיניות זו הוא תמיכה בדיסק גדול יותר ובגדלי קובץ גדולים יותר, אך במקרה שלנו זה קצת פחות רלוונטי משום שהוגדר שגודל קובץ מקסימלי הוא 4MB

חיסרון של שימוש במדיניות זו היא בזבוז מקום על המצביעים, במיוחד במקרה שבו יש קבצים רבים במערכת תיתכן תקורה גבוהה מהצפוי.

לכן לסיכום, בהנחה שיש כמות יחסית גדולה של קבצים שעושים בהם שימוש יחסית דחוף, המדיניות העדיפה תהיה לדעתי FAT.

הקצאה רציפה עלולה ליצור ריסוק שיגרום לניצול רע של שטח הדיסק.

מדיניות inode עלולה לתת ביצועים נחותים עקב בזבוז מקום רב למצביעים.

לעומת זאת במדיניות FAT למרות שנצטרך מקום לטבלת הקבצים, מדובר בגודל קבוע וידוע מראש שאינו משתנה ולא תלוי במספר הקבצים, ולכן בשילוב פוטנציאל לביצועים טובים, נעדיף את מדיניות זו.

1. נציג 3 שיטות להגנה על ה-capabilities:
2. הגנה ע"י הצפנה: כתיבה של ה-capabilities בצורה מוצפנת בזיכרון, כך שרק מערכת ההפעלה תוכך לפענח ולשנות את הרשימה, שיטה זו לא מונעת את דריסת הרשימה על ידי ערכים אחרים, אך מונעת שינויים ספציפיים.
3. הגנה ע"י חומרה: הגבלת כתיבה על האזור/המילים בזיכרון שמייצגות את הרשימה, כך שמשתמש ברמת user mode לא יוכל לכתוב שם (מתבצע על ידי הוספת סיבית הרשאות לכל חלק בזיכרון).
4. הגנה ע"י מערכת ההפעלה: שמירה של כל הcapabilities באזורים פנימיים שמורשים לגישה ושינוי רק על ידי מערכת ההפעלה וכל גישה אחרת תחסם על ידי מערכת ההפעלה, כך שרק היא תוכל לכתוב מה שתרצה.