ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Corso di Laurea in Ingegneria e Scienze Informatiche

B-Tree File System per piattaforme IoT

Tesi di laurea in:
Supervisor's course name

Relatore
Marco Antonio Boschetti

Candidato
Leonardo Marcaccio

I Sessione di Laurea
Anno Accademico 2023-2024

Abstract

Questa Tesi esplora il processo di studio, progettazione e implementazione di un prototipo di File System (FS) basato sulla struttura dati B-Tree, sviluppato per la piattaforma IoT IOtto di Onit S.p.A.

L'obiettivo principale è migliorare l'efficienza nella gestione e nel recupero dei dati, affrontando le problematiche tipiche di un FS come la scalabilità e l'ottimizzazione delle risorse.

Il lavoro comprende un'analisi dello stato dell'arte sui moderni FS, con particolare attenzione alla loro interazione con i sistemi IoT. Viene approfondito il principio di funzionamento del B-Tree, dimostrando come questa struttura dati possa essere sfruttata per realizzare una struttura performante e affidabile.

Inoltre, vengono descritte le fasi di progettazione e implementazione, evidenziando le soluzioni adottate per adattare il prototipo alle esigenze specifiche della piattaforma IOtto.

I risultati preliminari mostrano che il prototipo proposto garantisce significativi miglioramenti in termini di velocità di accesso ai dati e utilizzo delle risorse rispetto alle alternative tradizionali.

Optional. Max a few lines.

Contents

List of Figures

List of Listings

Chapter 1

Introduction

1.1 Background

1.1.1 Onit

1.1.2 IOtto

AGGIUNGI PRESENTAZIONE DELL'AZIENDA E DEL PRODOTTO

1.2 Descrizione del problema

L'attuale FS utilizzato dall'applicativo risulta non essere più ottimale nel contesto attuale, data la crescita e la mole dei dati gestiti. Risulta dunque necessario trovare un'alternativa che soddisfi le necessità attuali e future, in ambito di scalabilità e ottimizzazione.

1.3 Obiettivi di Tesi

L'obbiettivo della Tesi risulta essere quello di realizzare un prototipo di FS che sia scalabile, in grado di effettuare rapide letture e scritture, strutturalmente solido e ottimizzato nell'allocazione di spazio.

In particolare, il lavoro si concentra sulla progettazione e realizzazione di un prototipo alternativo al sistema preesistente, con l'intento non solo di mantenere

le prestazioni attuali, ma, ove possibile, di migliorarle.

1.3.1 Domande

- Quali tipi di dati devono essere memorizzati e gestiti?
- Qual è la quantità stimata di dati da gestire a regime?
- Quali saranno le dimensioni massime dei file e la granularità delle operazioni sui dati?
- Come garantire un basso consumo di risorse?
- Qual è la struttura di archiviazione più adatta?
- Il FS dovrà scalare per gestire un numero crescente di dispositivi IoT?
- Qual è la strategia per l'espansione dello spazio di archiviazione?
- Come gestire guasti hardware o interruzioni improvvise di alimentazione?
- È necessario prevedere meccanismi di backup o snapshot per i dati?
- Come si possono introdurre nuove funzionalità senza compromettere i dati esistenti?

1.3.2 Scopo

L'obiettivo finale è quello di apportare un significativo miglioramento al software esistente, sfruttando l'hardware già disponibile per ottenere prestazioni più elevate. Questo intervento mira a incrementare le capacità del prodotto, garantendo una maggiore efficienza e un'esperienza d'uso più soddisfacente per i clienti.

In particolare, il miglioramento si traduce nell'ottimizzazione delle funzionalità attuali e nell'ampliamento delle possibilità offerte dal sistema, lavorando direttamente sul software. Questo approccio consente di massimizzare il valore del prodotto, andando incontro alle crescenti aspettative degli utenti e mantenendo un vantaggio competitivo sul mercato.

1.4 Struttura della Tesi

Questa Tesi è strutturata in tre capitoli principali.

Il primo capitolo fornisce le basi teoriche del lavoro, offrendo una panoramica sui moderni FS e sulla loro connessione con il mondo dell'Internet of Thing (IoT). Viene inoltre approfondita la struttura dati B-Tree, evidenziandone le proprietà principali e la sua rilevanza rispetto al problema affrontato.

Il secondo capitolo si concentra sulla progettazione e sull'implementazione del prototipo di FS. Vengono descritte le scelte effettuate durante il processo di sviluppo e spiegato come il B-Tree sia stato utilizzato per soddisfare le esigenze specifiche della piattaforma IOtto.

Infine, il terzo capitolo discute i risultati ottenuti dalla valutazione del prototipo. Vengono inoltre presentate le conclusioni tratte dallo studio e delineate le possibili direzioni per future modifiche e sviluppi relativi al prodotto.

Chapter 2

State of the art

I suggest referencing stuff as follows: **??** or **??**

2.1 L'Internet of Things

Il concetto di Internet of Things, o in italiano Internet delle Cose, rappresenta un'evoluzione nell'utilizzo della rete Internet, in cui gli oggetti, o "cose", diventano riconoscibili e acquisiscono la capacità di comunicare dati attraverso la rete e di accedere a informazioni aggregate da altre fonti.

Uno dei primi utilizzi di questo termine risale al 2001, in un documento del centro Auto-ID relativo alla creazione del network EPC, concepito per tracciare automaticamente il flusso di beni nella catena di fornitura.

Tuttavia, non esiste un'origine univoca o un creatore effettivo del termine, poiché il concetto si è sviluppato progressivamente attraverso contributi di diversi ricercatori e innovatori nell'arco di tempo che va dai primi anni '90 fino al 2010.

Gli oggetti connessi, che costituiscono il cuore pulsante dell'IoT, sono definiti "oggetti intelligenti" (Smart Objects) e si caratterizzano per proprietà distintive come la capacità di identificarsi, connettersi, localizzarsi, elaborare dati e interagire con l'ambiente circostante.

Attraverso tecnologie come le etichette RFID (Identificazione a radiofrequenza) o i codici QR, questi oggetti e luoghi possono comunicare informazioni tramite la

rete o dispositivi mobili.

L'obiettivo principale dell'IoT risulta essere dunque il tentativo di creare una mappatura elettronica del mondo fisico, attribuendo un'identità digitale agli oggetti e ai luoghi che lo compongono.

Sono nati inoltre organizzazioni come EPCglobal e Open Geospatial Consortium (o OGC) che lavorano per creare standard globali per la visibilità dei dati e l'utilizzo di sensori connessi via Web.

In conclusione, le applicazioni dell'IoT sono molteplici e variano dai processi industriali alla logistica, dalla logistica all'efficienza energetica, fino all'assistenza remota e alla salvaguardia ambientale.
[Uck11][HBE11][Cha13]

2.2 File System

Tutti i processi e le applicazioni informatiche necessitano di archiviare e recuperare informazioni, ed è possibile, durante la loro esecuzione di memorizzare una quantità limitata di informazioni all'interno del proprio spazio di indirizzamento.
Questo metodo di gestione dei dati però non viene sprovvisto di problematiche:

- Capacità limitata, in quanto lo spazio di indirizzamento virtuale limita la quantità di dati archiviabili, rendendolo inadeguato per applicazioni che richiedono un'ampia archiviazione di dati, come sistemi bancari o database.
- Le informazioni memorizzate nello spazio di indirizzamento di un processo sono altamente volatili e vengono perse al termine del processo stesso.
- Non è permesso l'accesso concorrente. Spesso è necessario che più processi possano accedere contemporaneamente alle stesse informazioni, ma il confinamento dei dati all'interno di un unico processo lo impedisce.

Per rispondere a queste sfide, l'archiviazione a lungo termine richiede il rispetto di tre requisiti fondamentali:

- La possibilità di memorizzare grandi quantità di informazioni.
- La persistenza delle informazioni, anche dopo la terminazione del processo.
- L'accesso simultaneo ai dati da parte di più processi.

Per affrontare queste problematiche, è stato sviluppato il concetto di File System, una componente fondamentale dei sistemi informatici progettata per permettere la gestione completa dei dati, inclusa la loro creazione, modifica, archiviazione e recupero all'interno di un sistema.

Un file system, al suo livello più alto, fornisce dunque un modo organizzato per archiviare, recuperare e gestire informazioni su supporti di archiviazione permanenti come i dischi.

Esistono diversi approcci alla gestione dell'archiviazione permanente, poiché il problema è ampio e offre numerose soluzioni possibili. [Gia98][TB15]

2.3 Categorizzazione dei File System

I file system possono essere suddivisi in diverse categorie in base a specifiche caratteristiche, tra cui il livello di organizzazione gerarchica. Di seguito, analizziamo due tipologie principali di file system in relazione alla loro struttura gerarchica.

2.3.1 Single-Level Directory Systems

Questo tipo di file system rappresenta la forma più basilare e semplice di organizzazione dei dati.

Consiste in una singola directory, spesso denominata root, che contiene tutti i file del sistema.

Non esistono ulteriori livelli o sottodirectory: tutti i file sono archiviati in un unico spazio condiviso.

L'approccio a livello unico offre alcuni vantaggi distinti. In primo luogo, la semplicità della struttura lo rende facile da implementare e utilizzare, poiché non sono necessari meccanismi complessi per navigare tra directory o sottodirectory. Inoltre, la localizzazione dei file è immediata, poiché tutto è contenuto all'interno di una singola cartella, rendendo inutile un sistema di ricerca avanzato.

Tuttavia, questo modello ha limitazioni significative, soprattutto per applicazioni più complesse o ambienti che richiedono la gestione di un gran numero di file.

Per questo motivo, i sistemi a directory singola sono utilizzati principalmente in dispositivi embedded o sistemi con esigenze limitate, come fotocamere digitali, lettori MP3 o dispositivi elettronici di consumo di base. In tali contesti, la semplicità e la leggerezza dell'architettura superano le necessità di una maggiore organizzazione.

2.3.2 Hierarchical Directory Systems

A differenza del modello a singolo livello, i sistemi a directory gerarchica sono progettati per rispondere alle esigenze dei moderni ambienti informatici, caratterizzati dalla gestione di grandi volumi di dati e file.

In questo tipo di file system, è possibile creare una struttura organizzata composta da directory e sottodirectory. Questa struttura consente di suddividere i file in categorie o gruppi logici, favorendo una gestione più efficiente e un accesso più rapido alle informazioni.

L'organizzazione gerarchica risulta particolarmente utile quando il sistema deve gestire elevate quantità di file. La possibilità di classificare i file in directory dedicate consente di mantenere ordine e chiarezza, riducendo la complessità che deriverebbe dall'avere tutto in un'unica directory.

Un ulteriore vantaggio di questo approccio emerge nei contesti multiutente, dove più utenti condividono lo stesso file server. In questi casi, ogni utente può disporre di una directory root personale, che funge da punto di partenza per una gerarchia dedicata. Questa configurazione garantisce non solo la separazione dei dati tra gli utenti, ma anche un livello di personalizzazione che si adatta alle esigenze individuali.

In conclusione, mentre i sistemi a livello singolo sono ideali per applicazioni semplici e specifiche, i sistemi gerarchici rappresentano una soluzione robusta e scalabile per ambienti complessi e moderni.

[TB15]

2.4 Data Structures

Una struttura dati è un modo per memorizzare e organizzare i dati al fine di facilitare l'accesso e le modifiche.

Sono elementi fondamentali nella progettazione e nell'implementazione dei file system, poiché determinano come i dati vengono organizzati, memorizzati e recuperati, influenzando direttamente l'efficienza e le prestazioni delle operazioni sui dati.

Nessuna struttura dati singola è ottimale per tutti gli scopi, quindi è importante conoscere i punti di forza e le limitazioni delle diverse strutture.

La scelta della struttura dati appropriata dipende da vari fattori, tra cui le esigenze specifiche del sistema, le prestazioni richieste e le limitazioni hardware.

Tra le strutture dati più comuni utilizzate nei file system, si possono citare:

- Linked List: una sequenza di nodi collegati tra loro, in cui ogni nodo contiene un riferimento al nodo successivo.
- Array: una collezione di elementi disposti in una sequenza lineare, in cui ogni elemento è accessibile tramite un indice numerico.
- Tree: una struttura gerarchica composta da nodi collegati tra loro, in cui ogni nodo può avere uno o più nodi figli.
- Hash Table: una struttura dati che associa chiavi a valori, consentendo un accesso rapido e diretto ai dati.

2.5 B-Tree

Citando

> B-trees are balanced search trees designed to work well on disk drives or other direct-access secondary storage devices. B-trees are similar to red-black trees, but they are better at minimizing the number of operations that access disks.

2.6 B-Tree nella realizzazione dei File System

Chapter 3

Contribution

You may also put some code snippet (which is NOT float by default), eg: chapter 3.

3.1 Fancy formulas here

```
public class HelloWorld {
  public static void main(String[] args) {
    // Prints "Hello, World" to the terminal window.
    System.out.println("Hello, World");
  }
}
```

Bibliography

[Cha13] Hakima Chaouchi. *The internet of things: Connecting objects to the web*. John Wiley & Sons, 2013.

[CLRS22] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[Gia98] Dominic Giampaolo. *Practical file system design with the Be file system*. Morgan Kaufmann Publishers Inc., 1998.

[HBE11] Olivier Hersent, David Boswarthick, and Omar Elloumi. *The internet of things: Key applications and protocols*. John Wiley & Sons, 2011.

[TB15] Andrew S Tanenbaum and Herbert Bos. *Modern operating systems*. Pearson Education, Inc., 2015.

[Uck11] D Uckelmann. *Architecting the Internet of Things*. Springer, 2011.

Acknowledgements

Optional. Max 1 page.