Lezione: Sistemi Distribuiti

Introduzione ai Sistemi Distribuiti

I **sistemi distribuiti** sono una tipologia di sistema informatico in cui componenti software e hardware, dislocati su diversi nodi di una rete (come computer, server, dispositivi mobili, ecc.), collaborano e comunicano tra loro per raggiungere un obiettivo comune. L'idea alla base di un sistema distribuito è di permettere l'elaborazione e la condivisione di risorse in modo efficiente e affidabile, superando le limitazioni di un singolo sistema isolato.

A Cosa Servono i Sistemi Distribuiti?

I sistemi distribuiti sono utilizzati in molte applicazioni pratiche per migliorare l'efficienza, l'affidabilità, la scalabilità e la disponibilità dei servizi informatici. Alcuni esempi di utilizzo includono:

- **Server Web e Cloud Computing**: Distribuzione delle risorse su più server per bilanciare il carico e garantire alta disponibilità.
- **Reti di Sensori**: Utilizzate per monitorare ambienti e raccogliere dati in tempo reale.
- **Sistemi di File Distribuiti**: Come HDFS (Hadoop Distributed File System), che consentono l'archiviazione e l'accesso a grandi quantità di dati in modo distribuito.
- **Applicazioni di E-commerce**: Sistemi che gestiscono transazioni e dati di utenti distribuiti su più server per migliorare la sicurezza e la velocità.
- **Blockchain e Sistemi Finanziari**: Utilizzano nodi distribuiti per garantire l'integrità e la trasparenza delle transazioni.

Classificazione dei Sistemi Distribuiti

I sistemi distribuiti possono essere classificati in base a diversi criteri:

1. **In base all'Architettura:**
    - **Client-Server**: Architettura tradizionale dove i client richiedono servizi a uno o più server centralizzati.
    - **Peer-to-Peer (P2P)**: Tutti i nodi hanno lo stesso ruolo e possono agire sia come client che come server. Questo modello è utilizzato in molte applicazioni di condivisione di file e criptovalute.
    - **Sistemi Ibridi**: Combinano aspetti di entrambe le architetture, utilizzando una struttura client-server per alcune operazioni e un modello P2P per altre.
2. **In base alla Topologia di Rete:**
    - **Bus**: Tutti i nodi condividono un unico canale di comunicazione.
    - **Anello (Ring)**: I nodi sono collegati in una configurazione ad anello, dove ogni nodo è connesso esattamente a due altri nodi.
    - **Stella (Star)**: Un nodo centrale agisce come hub per la comunicazione con altri nodi.
    - **Maglia (Mesh)**: Ogni nodo è connesso direttamente a uno o più altri nodi, migliorando la ridondanza e l'affidabilità.

3. **In base alla Coerenza dei Dati:**
    - **Sistemi Fortemente Consistenti**: Garantisce che tutti i nodi abbiano una visione identica dei dati in ogni momento.
    - **Sistemi Eventualmente Consistenti**: Consentono che le copie dei dati su nodi diversi possano temporaneamente essere incoerenti, ma alla fine convergono verso uno stato consistente.

Classificazione dei Sistemi Distribuiti

I sistemi distribuiti possono essere classificati in diverse categorie in base alle loro caratteristiche, architetture e finalità. Le principali classificazioni includono:

1. **Sistemi di Calcolo con Cluster**
2. **Sistemi di Calcolo Grid**
3. **Sistemi Informativi Distribuiti**
4. **Sistemi Distribuiti Pervasivi**

Ecco un approfondimento su ciascuna di queste categorie:

1. Sistemi di Calcolo con Cluster

Definizione:
I sistemi di calcolo con **cluster** sono costituiti da un insieme di computer (nodi) collegati tra loro tramite una rete ad alta velocità. Ogni nodo in un cluster è una macchina fisica indipendente, ma i nodi lavorano insieme per eseguire compiti complessi come se fossero un unico sistema.

Caratteristiche Principali:

- **Prestazioni Elevate**: I cluster sono progettati per fornire alte prestazioni tramite il parallelismo. I compiti vengono suddivisi tra i vari nodi per essere eseguiti simultaneamente.
- **Affidabilità e Tolleranza ai Guasti**: Se un nodo in un cluster fallisce, altri nodi possono prendere il suo posto, garantendo la continuità del servizio.
- **Scalabilità**: È possibile aggiungere più nodi per aumentare la capacità di calcolo.

Esempi:

- **Cluster HPC (High-Performance Computing)**: Utilizzati per calcoli scientifici intensivi, simulazioni, analisi di big data, e applicazioni di intelligenza artificiale.
- **Cluster Web Server**: Utilizzati da grandi organizzazioni per gestire il traffico elevato dei siti web, bilanciando il carico di lavoro tra diversi server.

2. Sistemi di Calcolo Grid

Definizione:
I **sistemi di calcolo grid** (o "griglie") sono costituiti da un insieme di risorse di calcolo distribuite geograficamente, che collaborano per risolvere problemi complessi. A differenza dei cluster, i grid possono includere risorse eterogenee, come supercomputer, server e persino dispositivi personali.

Caratteristiche Principali:

- **Geograficamente Distribuiti**: I nodi di un grid sono spesso sparsi su un'area geografica vasta e non sono necessariamente collegati da una rete ad alta velocità.
- **Condivisione delle Risorse**: Consentono la condivisione di risorse eterogenee (potenza di calcolo, storage, dati) tra organizzazioni e utenti.
- **Eterogeneità e Autonomia**: Le risorse in una griglia possono avere diverse configurazioni hardware e software e possono essere gestite in modo autonomo dalle loro organizzazioni.

Esempi:

- **Progetti di Calcolo Distribuito per la Ricerca Scientifica**: Come il progetto SETI@home, che utilizza i computer di volontari per analizzare segnali radio dallo spazio.
- **Grid Computing per l'Analisi di Dati in Fisica delle Particelle**: Utilizzato dal CERN per l'analisi dei dati prodotti dagli esperimenti con il Large Hadron Collider (LHC).

3. Sistemi Informativi Distribuiti

Definizione:
I **sistemi informativi distribuiti** sono progettati per la gestione e la distribuzione di informazioni attraverso più nodi. Sono utilizzati principalmente per archiviare, gestire e processare dati su una rete di computer, fornendo accesso distribuito a tali dati.

Caratteristiche Principali:

- **Accesso e Condivisione di Dati Distribuiti**: Consentono a utenti e applicazioni di accedere e condividere dati distribuiti su diversi nodi in modo trasparente.
- **Coerenza dei Dati**: Devono gestire la coerenza dei dati tra diversi nodi, utilizzando tecniche come la replica e la sincronizzazione.
- **Affidabilità e Disponibilità**: Progettati per garantire l'accesso continuo ai dati anche in caso di guasti o disconnessioni di nodi.

Esempi:

- **Sistemi di Database Distribuiti**: Come Apache Cassandra, che memorizza e gestisce grandi volumi di dati distribuiti su molti nodi.
- **Sistemi di File Distribuiti**: Come Hadoop Distributed File System (HDFS), utilizzato per archiviare e gestire grandi quantità di dati su cluster distribuiti.
- **Applicazioni di Enterprise Resource Planning (ERP)** distribuite**: Come SAP o Oracle ERP, che gestiscono i dati aziendali tra vari reparti e sedi.

4. Sistemi Distribuiti Pervasivi

Definizione:
I **sistemi distribuiti pervasivi** (noti anche come **computazione ubiqua**) si riferiscono a reti di dispositivi connessi che interagiscono in modo continuo con l'ambiente circostante per fornire servizi e informazioni agli utenti in tempo reale. Questi sistemi sono integrati nell'ambiente fisico e si adattano dinamicamente alle esigenze dell'utente.

Caratteristiche Principali:

- **Dispositivi Eterogenei e Connessi**: Comprendono una vasta gamma di dispositivi, come smartphone, sensori IoT (Internet of Things), dispositivi indossabili, ecc., che comunicano e collaborano tra loro.
- **Context-Awareness (Consapevolezza del Contesto)**: Questi sistemi possono raccogliere e analizzare informazioni dal contesto dell'utente (posizione, attività, preferenze) per fornire servizi personalizzati.
- **Interazione Continuativa e Adattativa**: Forniscono servizi e informazioni in modo continuo e adattativo, spesso senza richiedere l'intervento dell'utente.

Esempi:

- **Smart Home Systems**: Sistemi di automazione domestica che utilizzano sensori e dispositivi intelligenti per controllare luci, termostati, elettrodomestici e sistemi di sicurezza.
- **Wearable Health Monitors**: Dispositivi indossabili che monitorano parametri vitali (come battito cardiaco, livello di ossigeno nel sangue) e inviano dati a sistemi sanitari distribuiti per l'analisi e il monitoraggio continuo.
- **Sistemi di Trasporto Intelligente**: Come veicoli autonomi e infrastrutture di traffico intelligenti che utilizzano sensori e reti distribuite per migliorare la sicurezza e l'efficienza del traffico.

Benefici dei Sistemi Distribuiti

1. **Scalabilità**: I sistemi distribuiti permettono di aggiungere nuove risorse (nodi) facilmente per aumentare la capacità di elaborazione o di archiviazione.
2. **Affidabilità e Tolleranza ai Guasti**: In un sistema distribuito, la perdita o il malfunzionamento di un singolo nodo non comporta necessariamente il fallimento dell'intero sistema. Ridondanza e replica dei dati possono essere utilizzate per garantire la continuità del servizio.
3. **Disponibilità**: I sistemi distribuiti possono essere progettati per garantire alta disponibilità, riducendo il tempo di inattività grazie alla ridondanza e alla distribuzione delle risorse.
4. **Prestazioni**: I sistemi distribuiti possono migliorare le prestazioni suddividendo le attività di elaborazione su più nodi, riducendo così il carico su ciascun nodo individuale.
5. **Flessibilità e Modularità**: È possibile sviluppare e aggiornare componenti del sistema indipendentemente, senza interrompere l'intero servizio.

Svantaggi dei Sistemi Distribuiti

1. **Complessità**: La progettazione, implementazione e gestione di sistemi distribuiti è molto più complessa rispetto ai sistemi centralizzati. Devono essere affrontate problematiche come sincronizzazione, coerenza dei dati, tolleranza ai guasti e sicurezza.
2. **Problemi di Comunicazione**: Poiché i nodi di un sistema distribuito comunicano attraverso una rete, sono soggetti a ritardi di trasmissione, perdita di pacchetti, congestione della rete e altri problemi legati alla comunicazione.
3. **Sicurezza**: Aumenta la superficie di attacco, poiché ogni nodo rappresenta un potenziale punto di ingresso per malintenzionati. La sicurezza dei dati e delle comunicazioni diventa cruciale.

4. **Gestione della Consistenza dei Dati**: Garantire che tutti i nodi abbiano una vista coerente dei dati può essere complesso, soprattutto in presenza di guasti o ritardi di rete.
5. **Costi di Implementazione**: Richiede risorse hardware e software significative, oltre a un'infrastruttura di rete affidabile e sicura.

Approfondimento: Concorrenza e Programmazione Parallela nei Sistemi Distribuiti

Concorrenza e Programmazione Parallela

La **concorrenza** e la **programmazione parallela** sono concetti chiave nel funzionamento dei sistemi distribuiti. Entrambi riguardano l'esecuzione simultanea di più operazioni, ma con alcune differenze fondamentali:

- **Concorrenza**: Si riferisce alla gestione di più compiti (tasks) che possono essere eseguiti nello stesso periodo di tempo. Non implica necessariamente che i compiti vengano eseguiti nello stesso momento, ma piuttosto che il sistema possa passare rapidamente da un compito all'altro (multitasking). In un sistema distribuito, la concorrenza si manifesta quando diversi nodi gestiscono contemporaneamente richieste o operazioni indipendenti, ottimizzando così l'uso delle risorse.
- **Programmazione Parallela**: Coinvolge l'esecuzione simultanea di più operazioni in modo che vengano eseguite contemporaneamente. Questo richiede l'uso di più processori o nodi, come nei sistemi distribuiti. La programmazione parallela è comune in applicazioni che richiedono elaborazione ad alte prestazioni, dove diverse parti di un compito possono essere suddivise e distribuite su più nodi per accelerare il completamento.

Esempi Pratici di Concorrenza e Programmazione Parallela nei Sistemi Distribuiti

1. **Motori di Ricerca e Crawler Web**:
    - I motori di ricerca come Google utilizzano sistemi distribuiti per eseguire **crawler web** che scandagliano miliardi di pagine su Internet. In questo contesto, la concorrenza si manifesta quando più crawler lavorano simultaneamente su diverse parti del web per raccogliere e indicizzare contenuti. Ogni crawler è un nodo distribuito che opera indipendentemente ma in coordinazione con gli altri.
    - La programmazione parallela è utilizzata per elaborare e analizzare grandi volumi di dati raccolti dai crawler. Ad esempio, l'indicizzazione di contenuti può essere suddivisa in sotto-compiti che vengono eseguiti simultaneamente su più server, migliorando significativamente la velocità di aggiornamento dell'indice.
2. **Rendering di Grafica in Cloud**:
    - Servizi di rendering grafico come quelli utilizzati nei film d'animazione o nei videogiochi ad alta risoluzione spesso fanno uso di **sistemi di calcolo distribuito**. In questo caso, un'immagine complessa o una scena 3D viene suddivisa in "fotogrammi" o "blocchi" più piccoli, ognuno dei quali viene elaborato simultaneamente da diversi nodi distribuiti.
    - Qui, la programmazione parallela permette di dividere il carico di lavoro su molteplici nodi che lavorano contemporaneamente, riducendo significativamente il tempo totale di rendering rispetto a un sistema singolo.
3. **Elaborazione di Big Data**:
    - I framework di Big Data come **Apache Hadoop** e **Apache Spark** sfruttano sia la concorrenza che la programmazione parallela. Hadoop utilizza un approccio di programmazione

parallela per suddividere grandi dataset in blocchi più piccoli, che vengono poi elaborati contemporaneamente su diversi nodi del cluster.

- Spark, d'altra parte, gestisce la concorrenza distribuendo compiti (tasks) su vari nodi che operano simultaneamente. Ad esempio, un'analisi di log file di grandi dimensioni per trovare pattern di comportamento degli utenti può essere eseguita in parallelo su diversi nodi, con ciascun nodo responsabile di una porzione del dataset.

4. **Sistemi di Trading ad Alta Frequenza (High-Frequency Trading, HFT)**:
   - Nei mercati finanziari, i sistemi HFT utilizzano algoritmi avanzati per eseguire operazioni di trading in microsecondi. Questi sistemi sono distribuiti su più server situati vicino ai centri di scambio per ridurre la latenza. La **concorrenza** si manifesta quando più algoritmi operano contemporaneamente su diverse fonti di dati di mercato per identificare opportunità di trading.
   - La **programmazione parallela** viene utilizzata per analizzare simultaneamente grandi volumi di dati storici e in tempo reale, calcolare indicatori finanziari e prendere decisioni di trading in frazioni di secondo.
5. **Servizi Cloud e Microservizi**:
   - I servizi cloud come Amazon Web Services (AWS) o Microsoft Azure utilizzano un'architettura basata su **microservizi**, dove diverse componenti di un'applicazione sono distribuite su più server. Ogni microservizio può gestire diverse richieste simultaneamente (concorrenza), mentre le operazioni intensive (come l'elaborazione di immagini o video) possono essere suddivise in più task paralleli.
   - Un esempio pratico è un sistema di gestione di contenuti (CMS) basato su microservizi: il servizio di autenticazione gestisce simultaneamente richieste di login da parte di utenti (concorrenza), mentre un altro servizio elabora e comprime in parallelo i video caricati.
6. **Sistemi di Raccomandazione**:
   - Piattaforme come Netflix, Amazon, e Spotify utilizzano **sistemi di raccomandazione** che si basano su modelli di machine learning per suggerire contenuti agli utenti. Tali modelli vengono addestrati su grandi dataset distribuiti su più nodi.
   - La concorrenza entra in gioco quando più modelli di raccomandazione vengono eseguiti contemporaneamente per diversi utenti, mentre la programmazione parallela è utilizzata per addestrare questi modelli su cluster di computer in cloud, accelerando il processo di apprendimento.