SISTEMI DISTRIBUITI

Al posto di comprare un PC grande si prendono n microprocessori
Un sistema distribuito è:

- network di workestation
- distribuzione manufatturiera
- network di P.C. d'ufficio

(ANALISI DATI per x@home)

Praticamente tutto il web si basa su reti di sistemi distribuiti

Alcuni aspetti

Vantaggi dei sistemi distribuiti rispetto a centralizzati:

- Più economico
- Più veloce, meno delay
- Migliore distribuzione di dati
- Reliability
- Crescita incrementale

Svantaggi:

- Software e sincronizzazione, e coordinazione
- Saturazioni di rete
- Sicurezza: accesso facile e protetto ai dati

Obbiettivo Primario

Scambio di risorse e dati
- Coordinazione
- Sincronizzazione

Coordinazione

Alcune considerazioni
- serve concorrenza spaziale e temporale
- Non c'è clock GLOBALE
- Latenze non controllabili
- Problemi

Per discgnare un sistema distribuito si devono gestire:

- Eterogeneità su più punti
- Apertura (interfacce pubbliche API)
- Sicurezza (C.I.A)
- Scalabilità (growth esponenziale accesso risorse centrali)
- Affidabilità (più affidabili sistemi singoli)
- Concorrenza
- Flessibilità (se dawn uno funzione tutto)
- Performance (performance gain)
- Trasparenza

In generale NON mi devo accorgere di alla macchina del sistema sto usando!

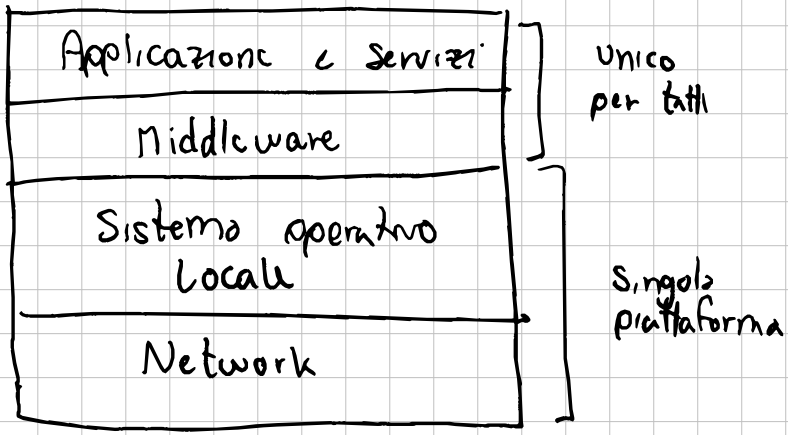
Sicurezza: (C.I.A)
- Confidenzialità: limitare accesso a risorse a chi non è autorizzato
- Integrità: protezione da alterazione e corruzione dei dati.
- Availability: protezione da interferenze per accedere a dati.

2 MODELLI DI INTERAZIONE

- Client-server
- Peer-to-peer

Layer di hw e sw

Se si vogliono progettare sistemi distribuiti è necessario fare cambiamenti..
SISTEMA UNICO di comunicazione



Alcuni problemi del middleware

- Estrema eterogeneità da gestire: OS, tipi di dati, clock..
- Non c'è conoscenza globale di tutti i node
- Asincronia locale e di rete

↳ Il middleware deve saper gestire queste situazioni attraverso un set di API

COMPUTING CLIENT-SERVER

- Client sono computer single client che lasciano un' interfaccia user-friendly
- Server rilascia una serie di risorse condivise ai client
- il server permette l'accesso simultaneo a più client per accedere a stesso database

La logica applicativa è fatta lato client.
La comunicazione passa attraverso, spesso, TCP-IP

Alcune componenti dell' applicazione

- logica di presentazione
- I/O di processing
- Business Processing
- Data storage

4 Classi di elaborazione

HOST BASED
- mainframe tradizionale. Tutto gestito da server, client solo mouse+tastiera + dumb!

COOPERATIVE PROCESSING
- logica applicativa e shared
- complessa da mantenere

SERVER BASED
- server fa tutto processing
- client fa logica presentazione

CLIENT-BASED PROCESSING
- client fa logica di presentazione, applicazione e database
- server fa logica DBMS e DB

ARCHITETTURA a 3 LIVELLI

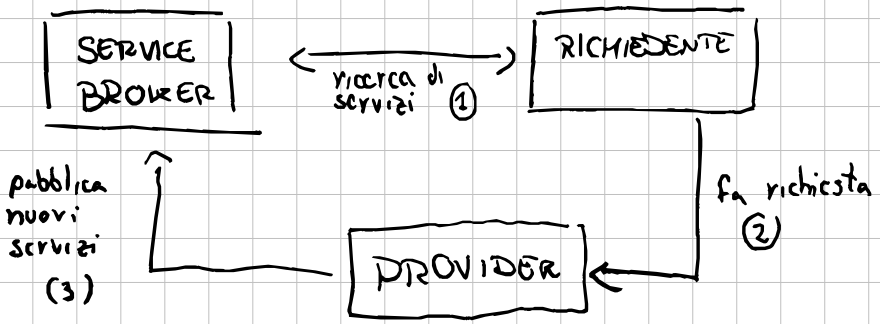
La logica software dell'applicazione è a 3 tipi di macchine

- USER MACHINE
- MIDDLE TIER SERVER
  - Gateway
  - Server di smistamento
- BACKEND SERVER

SERVICE-ORIENTED ARCHITECTURE

Usato in sistemi enterprise, organizza funzioni in struttura modulare, più che app. monolitica ad ogni dipartimento.

Insieme di servizi e appl. client che li usano, che comunicano attraverso interfacce standardizzate (e.g. XML)



REMOTE PROCEDURE CALL

- Fa in modo di avere interazioni tra diverse macchine attraverso statement e return
- STANDARDIZZATO
  - Il codice di comunicazione può essere generato automaticamente
  - Modularizzabile
- Server che contiene funzioni e procedure.

Passaggio parametri

- per valore è semplice
- per puntatore è difficile:
  - necessario sistema di puntamento universale
  - not worth.

BINDING

Abbinamento (anche persistente) tra macchine.
[come con TCP e UDP]
Nel caso del binding persistente è comodo perché basta una sola connessione e quella può essere usata per più procedure call

Nel binding temporaneo può essere comodo se si usa la RPC una o due volte.

RPC SINCRONE vs ASINCRONE

Synchronous:

- Sistema si blocca fino all'arrivo del risultato
- più facile da gestire, ma performance peggiori!
- Non sfrutta parallelismo

Asynchronous:

- Non blocca il caller
- client procede in parallelo con il server

ERRORI

Ci possono essere tre tipi di errori:

(1) Invio messaggio
(2) Elaborazione
(3) Ricezione risposta

Bisogna gestire ogni tipo di errore!

Gestione:

CLUSTER

gruppo di computer interconnessi che si comportano come una sola macchina.

Ogni pc che appartiene al cluster è detto NODO

failure management

Il failover è quello che si fa quando si passano dati e applicazioni da una macchina fail a una funzione

Il fallback è invece il restore di app e dati nel pc dopo che è stato messo a posto.

controllo carichi

Il cluster ha bisogno di un sistema di monitoraggio, avvio e trasloco di task

Middleware tiene traccia dei programmi, del carico