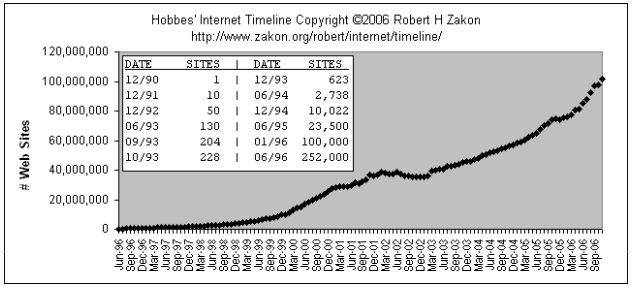
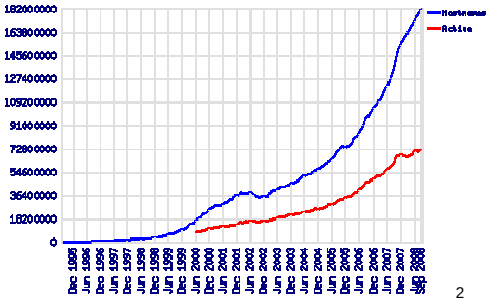
World Wide Web: introduzione e componenti

I segnali del successo del Web



Dal 2007 aumento esponenziale del numero di siti presso fornitori di servizi di blogging e social networking (MySpace, Live Spaces, Blogger, ..)

Fonte: Netcraft Web Server Survey (http://news.netcraft.com/archives/web_server_survey.html)



I segnali del successo del Web (2)

- Fino all'introduzione dei sistemi P2P, il Web è stata l'applicazione killer di Internet (75% del traffico di Internet nel 1998)

Event	*Period*	*Peak day*	*Peak minute*
NCSA server (Oct. 1995)	-	2 Million	-
Olympic Summer Games (Aug. 1996)	192 Million (17 days)	8 Million	-
Nasa Pathfinder (July 1997)	942 Million (14 days)	40 Million	-
Olympic Winter Games (Feb. 1998)	634.7 Million (16 days)	55 Million	110,000
Wimbledon (July 1998)	-	-	145,000
FIFA World Cup (July 1998)	1,350 Million (84 days)	73 Million	209,000
Wimbledon (July 1999)	-	125 Million	430,000
Wimbledon (July 2000)	-	282 Million	964,000
Olympic Summer Games (Sept. 2000)	-	875 Million	1,200,000

[Carico misurato in contatti]

- Inoltre: google.com, msn.com, yahoo.com (> 200 milioni di contatti al giorno)

I motivi alla base del successo del Web

- Digitalizzazione dell'informazione
  - Qualsiasi informazione rappresentabile in binario come sequenza di 0 e 1
- Diffusione di Internet (dagli anni 1970)
  - Trasporto dell'informazione ovunque, in tempi rapidissimi e a costi bassissimi
- Diffusione dei PC (dagli anni 1980)
  - Accesso, memorizzazione ed elaborazione dell'informazione da parte di chiunque a costi bassissimi
- Semplicità di utilizzo e trasparenza dell'allocazione delle risorse
  - Uso di interfacce grafiche

Una definizione di World Wide Web

- "L'universo dell'informazione globale accessibile tramite rete" (T. Bernars-Lee, l'inventore del WWW o Web)
- Il Web è un sistema *ipermediale*, *distribuito globalmente* che supporta accessi interattivi a risorse e servizi
  - ipermediale: diverse forme di rappresentazione delle risorse (testo, immagini, audio, video, …) tra loro collegate
    - *ipertesto*: sistema non lineare per la strutturazione di informazioni
  - distribuito globalmente: risorse distribuite e scalate su l'intera Internet
    - La ragnatela (web) di collegamenti è di ampiezza mondiale (world-wide)
    - risorse Web amministrate indipendentemente → possibile perdita di consistenza nel tempo

Una definizione di World Wide Web (2)

- Il Web è un sistema *aperto*
  - Può essere esteso e implementato con nuove modalità senza alterare le sue funzionalità esistenti
  - Il suo funzionamento è basato su standard di comunicazione e di documenti che sono pubblicamente disponibili ed ampiamente implementati
- Il Web è un sistema aperto rispetto al tipo di risorse che possono essere pubblicate e condivise
- Aspetto più interessante per gli utenti: a differenza di altri mezzi informativi, il Web è *on demand*
- Il Web non è sinonimo di Internet, ma rappresenta un'applicazione dell'infrastruttura Internet!

Evoluzione del Web

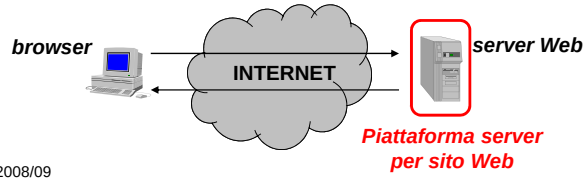
- Una classificazione "ufficiale" dell'evoluzione del Web
  - Web 1.0
  - Web 1.5
  - Web 2.0
- Una classificazione "alternativa" dell'evoluzione del Web
  - Prima generazione
  - Seconda generazione
  - Terza generazione
  - Servizi Web e Web semantico
  - Non solo evoluzione delle applicazioni ma anche prestazioni

Da Web 1.0 a Web 2.0

- Non è un'evoluzione di tipo tecnologico ma sociale
  - Approccio con il quale gli utenti si rivolgono al Web
- Web 1.0: modello di interazione statico
  - Semplice consultazione e fruizione di contenuti statici
- Web 1.5: modello di interazione dinamico
  - Consultazione e fruizione di contenuti dinamici
  - Tecnologie per la generazione di contenuti dinamici client-side e server-side
- Web 2.0: applicazioni online che permettono uno spiccato livello di interazione sito-utente
  - Termine coniato da T. O'Reilly e D. Dougherty nel 2004
  - Blog, forum, chat, sistemi quali Wikipedia, YouTube, Flickr, Facebook, Myspace, Gmail, …
  - Fruizione e creazione/modifica di contenuti multimediali

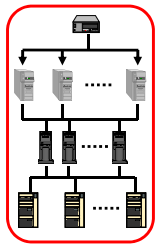
La prima generazione del Web

- Prima generazione (*ieri*) = Web publishing
  - Un ulteriore canale di comunicazione
  - 95% dell'informazione costituita da testo ed immagini
  - Siti Web prevalentemente **statici**, con alcune tecnologie (ad es. CGI) per la generazione di contenuti dinamici
  - Manutenzione ed aggiornamenti occasionali
  - Prestazioni molto variabili
  - Affidabilità non garantita
  - Sicurezza non indispensabile



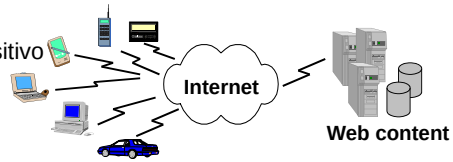
La seconda generazione del Web

- Seconda generazione (*ieri-oggi*) = Web-based information systems
  - Canale di informazione critica, divenuto un mezzo di informazione privilegiato per molti utenti
  - "Vetrina" importante per industrie e organizzazioni
  - Contenuti **dinamici** ed attivi in continuo aumento
  - Servizi di streaming audio e video
  - Servizi personalizzati, servizi a pagamento (diretto o indiretto)
  - Interfaccia di accesso per molti altri servizi informatici usufruiti via rete (anche se non propriamente servizi di rete)
    - E-mail, trasferimento di file
    - Accesso ad archivi, basi di dati, banche dati, …
  - Necessaria la **qualità di servizio** (QoS)
    - Prestazioni garantite (*regola degli X secondi*)
      - X=8, poi X=4, …
    - Affidabilità come capacità di tollerare guasti
    - Sicurezza



La terza generazione del Web

- Terza generazione (*oggi-domani*) = Ubiquitous Web
  - Web for Everyone and Web on Everything
  - Possibilità di usufruire di tutti i servizi Web della seconda generazione in modalità AAA
    - *Anytime:* sempre (24/7)
    - *Anywhere*: ovunque
    - *Anyway*: da qualunque dispositivo



  - In più: servizi time-aware, location-aware, device-aware e personalizzazione dei servizi
  - Prima fase:
    - Si accettano prestazioni ed affidabilità *variabili*
  - Seconda fase:
    - Prestazioni ed affidabilità *garantiti*
  - La sicurezza è comunque un requisito indispensabile

Ulteriori evoluzioni del Web

- Servizi Web (Web services):
  - Il Web come infrastruttura per la comunicazione e l'interazione tra applicazioni distribuite
  - Sistemi eterogenei possono lavorare insieme per realizzare il service oriented computing (SOC)
    - Programmazione con componenti distribuite sul Web
- Web semantico (Semantic Web):
  - Non più solo documenti, ma informazioni e dati relativi ai documenti stessi (metadati) in un formato adatto all'interrogazione, interpretazione e, più in generale, all'elaborazione automatica
  - Tale possibilità trasformerà il Web da *machine-readable* a *machine-understandable*, permettendo la nascita di applicazioni sofisticate in grado di interpretare ed elaborare dati in maniera semi-intelligente

Organismi di standardizzazione per il Web

- Internet Engineering Task Force (IETF, http://www.ietf.org/)
  - Realizzazione e discussione degli standard in ambito Internet
  - Pubblicazione delle specifiche approvate in forma di Request for Comments (RFC, http://www.rfc-editor.org/)
- World Wide Web Consortium (W3C, http://www.w3.org/)
  - Organizzazione fondata e diretta da T. Berners-Lee al fine di "sviluppare protocolli comuni per migliorare l'interoperabilità e guidare l'evoluzione del World Wide Web"
  - Produzione di specificazioni di interoperabilità e codice d'esempio
  - Pubblicazione di technical report (http://www.w3.org/TR/): W3C Note e Recommendation Track

Stack tecnologico del Web (secondo il W3C)

Fonte: http://www.w3.org/Consortium/technology

Ingredienti basilari del Web

- Informazione digitalizzata
- Architettura client-server
  - Componenti, funzionamento ed architettura del Web client (browser)
  - Componenti, funzionamento ed architettura del Web server
- Meccanismi di comunicazione e naming delle risorse (a livello di nodi) di Internet
  - Protocollo TCP/IP e Domain Name System (DNS)
- Tre principali componenti tecnologici standard alla base del Web
  - **URI**: meccanismo di naming universale per identificare le risorse
  - **HTTP**: protocollo per il trasferimento delle risorse
  - **HTML**: linguaggio di markup ipertestuale

Architettura client-server

- Browser (client Web)
  - Agente dell'utente per il Web
  - Guida l'interazione client/server nei confronti di un server per volta
  - Visualizza la pagina Web richiesta dall'utente; fornisce molte caratteristiche per la navigazione e configurazione
  - Implementa il client nel protocollo HTTP
- Server Web
  - Fornisce su richiesta le risorse Web
    - Memorizzate su disco o generate dinamicamente
  - Implementa il server nel protocollo HTTP

Macro-componenti del Web: client

- Diversi tipi di Web client (user agent nel protocollo HTTP)
- Browser Web
  - Forma più popolare di Web client
- Spider o robot (bot)
  - Applicazione automatica usata dai motori di ricerca per ottenere informazioni sulle risorse fornite dai server Web a diversi scopi (indicizzazione, catalogazione)
- Agente software
  - Client in grado di svolgere compiti specifici (ad esempio, motori di meta-ricerca, agenti per auction, verifica di correttezza sintattica)

Browser

- Applicazione software che svolge il ruolo di interfaccia fra l'utente ed il Web, mascherando la complessità di Internet e del Web
- Principali servizi del browser
  - Consente di specificare le richieste (URL)
  - Implementa il client del protocollo HTTP (query al DNS per la risoluzione dell'hostname nell'URL, gestione delle connessioni TCP, invio dei messaggi di richiesta HTTP)
  - Visualizza in modo appropriato il contenuto delle risposte sullo schermo (ad es., attivando programmi esterni per elaborare specifiche parti di una pagina Web) e consente la navigazione
  - Invia informazioni al server per la generazione di risorse dinamiche (codificate nell'URL o inviate con il metodo POST dell'HTTP), gestisce informazioni di stato (cookie)
  - Caching locale delle risorse
  - Altri servizi (preferiti, stampa, salva, ricerca nel testo, …)
- Browser diversi, diverse compatibilità

Breve storia

- *FrameMaker* **e** *Acrobat PDF*
  - consentono generazione e incorporazione di hyperlink
- *Gopher* (University of Minnesota)
  - primo browser con hyperlink verso siti remoti
  - uso di standard, quali testo ASCII e socket Unix
  - difetti: link separati dal testo, immagini non gestite

HTML Mosaic

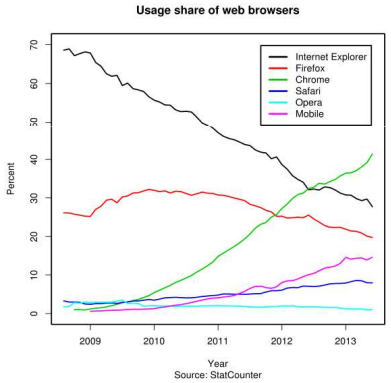
- *Mosaic* (NCSA, 1993)
  - *Netscape Navigator*
  - *Microsoft Explorer*

Principali browser

- Microsoft Internet Explorer
  - Leader di mercato
  - Trident come motore di rendering (o layout engine)
- Mozilla Firefox
  - Il maggiore competitor di IE
  - Gecko come layout engine
- Opera
  - Presto come layout engine
- Google Chrome
- Browser per dispositivi mobili
  - Es.: Nokia browser, Opera Mobile Browser, Internet Explorer Mobile per pocket PC e handheld PC
- Altri browser
  - Safari, Konqueror (browser di KDE), lynx (solo testuale)

Principali browser (2)

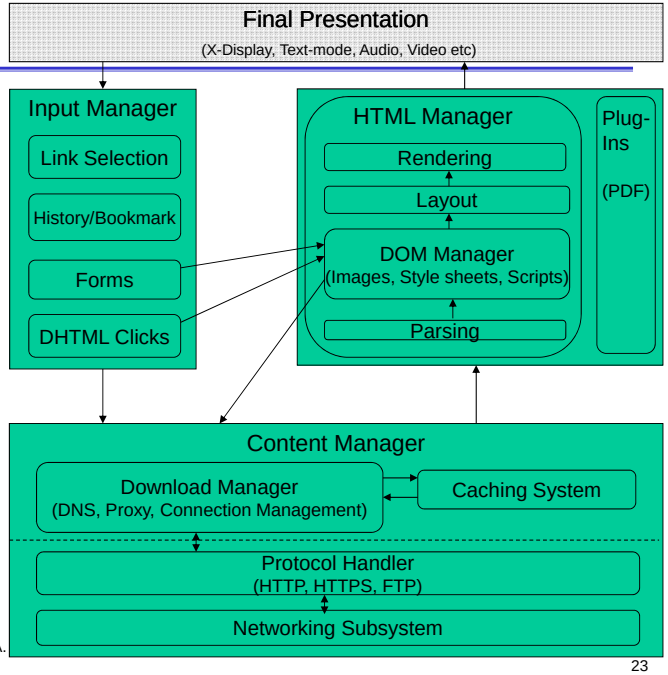
- Una stima approssimativa sulla diffusione dei principali browser



Componenti di un browser

- Il browser, di per sé, è in grado solo di visualizzare correttamente file HTML e immagini GIF e JPEG
- Può ricorrere ad applicazioni esterne (*plug-in* ed *helper*) per visualizzare o comunque per elaborare specifiche parti di una pagina Web; ad esempio:
  - Acrobat Reader per visualizzare file in formato pdf
  - RealAudio per video e video streams
  - Flash per animazioni
  - Java Virtual Machine per applet Java

Componenti dell'architettura di un browser



Componenti di un sito Web

- Piattaforma hardware
- Software di base
- Parte informativa ed applicativa del sito Web
  - Insieme di risorse Web che possono essere richieste dai client tramite messaggi di richiesta HTTP
  - Applicazioni per la generazione delle risorse
- Server Web
  - Il processo server ed il relativo software che viene eseguito sulla piattaforma (hardware - software di base)

Parte informativa

- Organizzata generalmente in pagine ipermediali con collegamenti verso altre pagine (interne o esterne al sito)
- L'organizzazione delle pagine è tipicamente gerarchica (*ad albero*), in cui vi è un punto di partenza e tutte le altre pagine sono poste al di sotto della radice dell'albero
- L'albero delle pagine Web è organizzato in modo simile ad un file system gerarchico con una radice e directory che contengono altre directory o file
- Ogni pagina Web ha un nome unico, che è il cammino assoluto dalla radice "/" dell'albero delle pagine
  - Questo cammino è quello specificato nella parte path dell'URL richiesto dal client
- La parte informativa del sito è creata e mantenuta dal *content provider*

Memorizzazione delle pagine

- L'albero delle pagine Web non riflette necessariamente la vera organizzazione dei file all'interno del file system
- L'organizzazione fisica che rispecchia fedelmente quella logica è solo una delle possibili alternative
- Per motivi di efficienza organizzativa (gruppi differenti possono creare o fornire le informazioni per le pagine) o di efficienza nella risposta, l'albero delle pagine Web può essere partizionato tra due o più dischi della stessa piattaforma o addirittura tra piattaforme differenti, utilizzando o meno meccanismi di Network File System

Memorizzazione delle pagine (2)

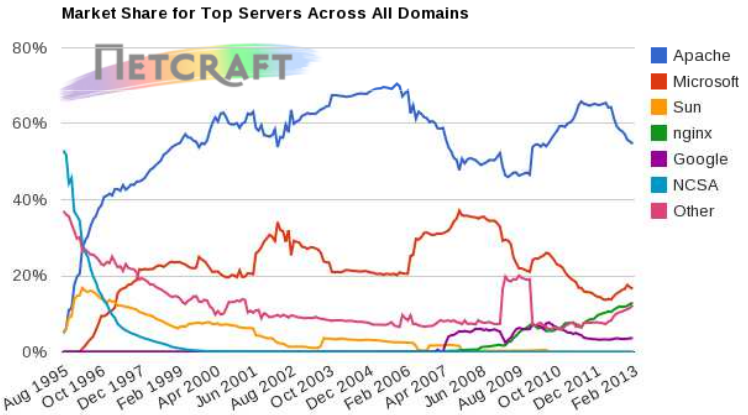
- *Mirroring*
  - l'intero albero è replicato su più dischi o piattaforme
- *Soluzioni intermedie*
  - le parti superiori dell'albero, ovvero le pagine più frequentemente richieste, sono replicate
  - le altre pagine possono essere o meno partizionate
- Ciascuna di queste organizzazioni deve essere trasparente per l'utente, che deve poter navigare e richiedere le pagine nello stesso identico modo, *indipendentemente* dall'organizzazione fisica dei file e dei servizi

Tipi di risorse (*classificazione funzionale*)

- Risorse statiche
  - risorse il cui contenuto è relativamente stabile nel tempo
- Risorse volatili
  - risorse il cui contenuto viene modificato da eventi in corso
    - Es.: ultime notizie, avvenimenti sportivi, titoli in borsa
- Risorse dinamiche
  - risorse il cui contenuto è creato dinamicamente sulla base della richiesta del client
- Risorse attive
  - risorse contenenti codice che viene eseguito dal client

Software per server Web

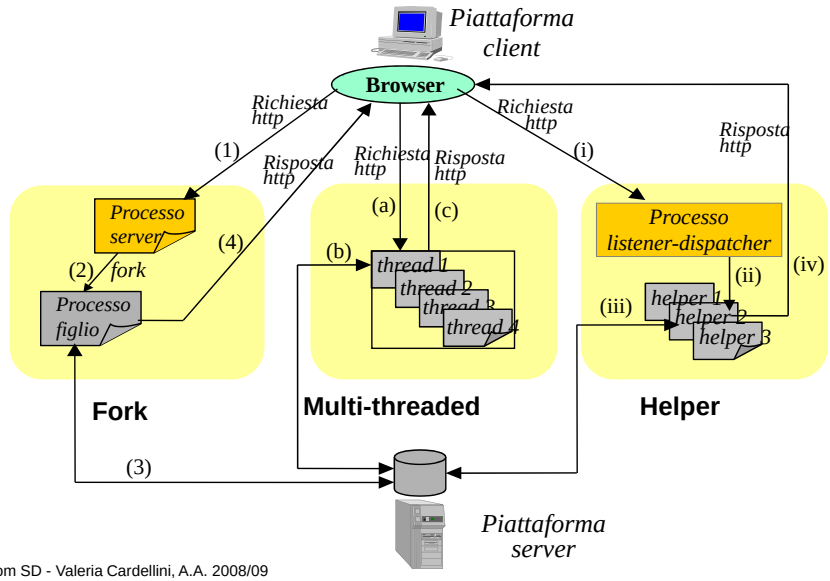
- I server Web più diffusi sono:
  - Apache (http://httpd.apache.org/)
  - Microsoft Internet Information Server (http://www.microsoft.com/)
  - Google Web server



Modelli architetturali di server Web

- Diversi approcci per l'architettura del server
  - Basato su processi
    - Fork e preforking
    - Esempio: Apache 1.X e Apache 2.X con mpm_prefork
  - Basato su thread
    - Esempio: Microsoft IIS
  - Ibrido (processi e thread)
    - Apache 2.X con mpm_worker
  - Basato su eventi
    - Esempio: Flash, Zeus, nginx
  - Interno al kernel
    - Esempio: Tux
- Nella scelta del modello tradeoff tra:
  - Prestazioni, robustezza, protezione, estensibilità, …

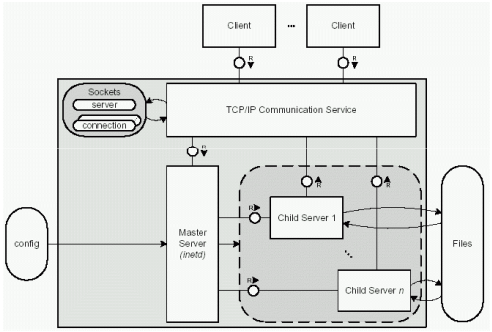
Principali approcci

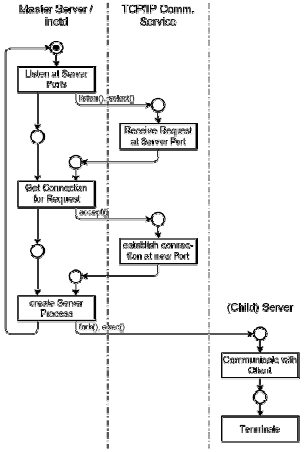


Server multi-process: fork

- Server basato su processi
- Per ogni nuova richiesta che arriva il processo server (padre):
  - Crea una copia di se stesso (un processo child), alla quale affida la gestione della richiesta tramite fork()
  - Si pone in attesa di nuove richieste
  - Il processo child si occupa di soddisfare la richiesta e poi termina
    - Un processo child per ogni client
  - Con fork():
    - Copia di dati, heap e stack; condivisione del segmento testo
    - No copia completa ma *copy-on-write*: è una ottimizzazione

Server multi-process: fork (2)





Server multi-process: fork (3)

- Vantaggi:
  - Il codice del server rimane semplice, poiché la copia è demandata in toto al sistema operativo
- Svantaggi:
  - Overhead di fork() può penalizzare l'efficienza del sistema
    - Il tempo di generazione del processo child può non essere trascurabile rispetto al tempo di gestione della richiesta
  - In mancanza di un limite superiore al numero di richieste che possono essere gestite concorrentemente, nel caso di un elevato numero di richieste i processi child possono esaurire le risorse del sistema
  - Meccanismo di IPC per la condivisione di informazioni tra padre e figli successivamente a fork()
    - Ad esempio memoria condivisa

Server multi-process: helper

- Server basato su processi
- Un processo *dispatcher* (o listener) ed alcuni processi per il servizio delle richieste, detti processi *helper* (o worker)
  - All'avvio del servizio, il dispatcher effettua il *preforking* dei processi helper (*pool* di processi helper)
  - Il dispatcher rimane in ascolto delle richieste di connessione
  - Quando arriva una richiesta di connessione, il dispatcher la trasferisce ad un helper per la gestione
    - Occorre usare una forma di passaggio di descrittori tra processi distinti
  - Quando l'helper termina la gestione della richiesta, si rende disponibile per gestire una nuova richiesta
  - A regime, il dispatcher svolge compiti di supervisione e controllo

Server multi-process: helper (2)

- Vantaggi
  - Processi helper creati una sola volta e poi riusati
    - Si evita l'overhead dovuto alla fork() all'arrivo di ogni nuova richiesta
  - Maggiore robustezza (separazione dello spazio di indirizzi) e portabilità rispetto al server multi-threaded
  - Maggiore semplicità rispetto al server basato su eventi (linearità nel modo di pensare del programmatore)
- Svantaggi
  - Processo dispatcher potenziale collo di bottiglia
  - Gestione del numero di processi helper nel pool
  - Maggior uso di memoria rispetto a server multi-threaded
  - Gestione della condivisione di informazioni tra i processi helper (uso di lock)

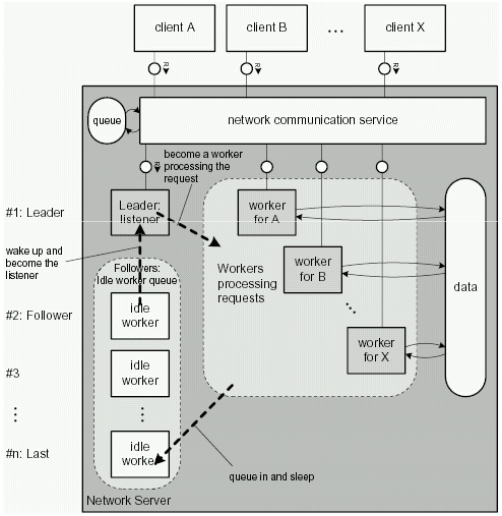
Schemi per preforking

- Schema con preforking considerato finora:
  - Il dispatcher effettua accept() e passa il descrittore del socket di connessione ad un helper
  - Anche noto come schema *job-queue*
    - Il dispatcher è il produttore; gli helper sono i consumatori
    - Il dispatcher accetta le richieste e le pone in una coda
    - Gli helper leggono le richieste dalla coda e le servono
- In alternativa, si può usare lo schema *leader-follower*

Schema leader-follower

- Dopo il preforking, ogni helper chiama accept() sul socket di ascolto
- Soluzione 1: nessuna forma di locking per accept
  - Problema a: *thundering herd*
  - Problema b: funziona correttamente su kernel Unix derivati da Berkeley (accept implementata nel kernel), ma non su kernel Unix derivati da System V (accept come funzione di libreria)
    - Il socket d'ascolto è una risorsa a cui accedere in mutua esclusione
- Soluzione 2: file locking per accept
  - Dopo il preforking, ogni helper chiama accept(), effettuando un *file locking* prima dell'invocazione di accept()
  - Vedi esempio: file locking Posix con funzione fcntl()
- Gli helper idle competono per accedere al socket d'ascolto
  - Al più uno (detto *leader*) si può trovare in ascolto, mentre gli altri (detti *follower*) sono accodati in attesa di poter accedere alla sezione critica per il socket d'ascolto

Schema leader-follower (2)



Preforking con file locking

```
static int      nchildren;
static pid_t    *pids;
int main(int argc, char **argv)
{
    int         listenfd, i;
    socklen_t   addrlen;
    void        sig_int(int);
    pid_t       child_make(int, int, int);

    …           /* creazione del socket di ascolto, bind() e listen() */
    pids = calloc(nchildren, sizeof(pid_t));
    my_lock_init("/tmp/lock.XXXXXX"); /* un file di lock per tutti i processi child */
    for (i = 0; i < nchildren; i++)
        pids[i] = child_make(i, listenfd, addrlen);
    …
}

pid_t child_make(int i, int listenfd, int addrlen)
{
    pid_t pid;
```

Preforking con file locking (2)

```
    if ( (pid = fork()) > 0)
        return(pid);                    /* processo padre */
    child_main(i, listenfd, addrlen);   /* non ritorna mai */
}

void child_main(int i, int listenfd, int addrlen)
{
    int     connfd;
    socklen_t clilen;
    struct sockaddr *cliaddr;

    cliaddr = malloc(addrlen);
    printf("child %ld starting\n", (long) getpid());
    for ( ; ; ) {
        clilen = addrlen;
        my_lock_wait();             /* my_lock_wait() usa fcntl() */
        connfd = accept(listenfd, cliaddr, &clilen);
        my_lock_release();
        web_child(connfd);          /* processa la richiesta */
        close(connfd);
    }
}
```


Preforking con file locking (3)

```
#include <fcntl.h>
#include "basic.h"
static struct flock  lock_it, unlock_it;
static int  lock_fd = -1;
    /* fcntl() will fail if my_lock_init() not called */

void my_lock_init(char *pathname)
{
  char  lock_file[1024];  /* must copy caller's string, in case it's a constant */

  strncpy(lock_file, pathname, sizeof(lock_file));
  if ( (lock_fd = mkstemp(lock_file)) < 0) {
    fprintf(stderr, "errore in mkstemp");
    exit(1);
  }
  if (unlink(lock_file) == -1) {            /* but lock_fd remains open */
    fprintf(stderr, "errore in unlink per %s", lock_file);
    exit(1);
  }
```


Preforking con file locking (4)

```
  lock_it.l_type = F_WRLCK;
  lock_it.l_whence = SEEK_SET;
  lock_it.l_start = 0;
  lock_it.l_len = 0;

  unlock_it.l_type = F_UNLCK;
  unlock_it.l_whence = SEEK_SET;
  unlock_it.l_start = 0;
  unlock_it.l_len = 0;
}
```


Preforking con file locking (5)

```
void my_lock_wait()
{
  int rc;
  while ( (rc = fcntl(lock_fd, F_SETLKW, &lock_it)) < 0) {
    if (errno == EINTR) continue;
    else {
      fprintf(stderr, "errore fcntl in my_lock_wait");
      exit(1);
    }
  }
}

void my_lock_release()
{
  if (fcntl(lock_fd, F_SETLKW, &unlock_it) < 0) {
    fprintf(stderr, "errore fcntl in my_lock_release");
    exit(1);
  }
}
```

Server multi-threaded

- Server basato su thread
- Una sola copia del server che genera thread multipli di esecuzione
  - Il thread principale rimane sempre in ascolto delle richieste
  - Quando arriva una richiesta, esso genera un nuovo thread (un *request handler*), che la gestisce e poi (eventualmente) termina
  - Ogni thread possiede una copia privata della connessione gestita, ma condivide con gli altri thread uno spazio di memoria (codice del programma e variabili globali)
- In alternativa alla creazione del thread all'arrivo della richiesta, il pool di thread può essere pre-creato (*prethreading*)

Server multi-threaded (2)

- Vantaggi
  - Creazione di un thread più veloce della creazione di un processo
    - Da 10 a 100 volte
  - Minore overhead per il context switching
  - Condivisione delle informazioni per default
  - Mantiene una maggiore semplicità rispetto a server basato su eventi
- Svantaggi
  - Maggiore complessità del codice del server (gestione della sincronizzazione tra thread)
  - Minore robustezza rispetto al server multi-process: i thread non sono protetti uno dall'altro
  - Supporto da parte del sistema operativo al multithreading (ad es., Linux/Unix, Windows)
  - Maggiori limitazioni sul numero di risorse rispetto al preforking (ad es. numero di descrittori aperti)

Preforking e prethreading

- Riassumiamo le possibili alternative per realizzare un server con preforking o prethreading
- Preforked server
  - Dispatcher/listener
    - Il processo padre invoca accept; occorre passare da padre a figlio il descrittore del socket di connessione
  - Leader/follower
    - Nessuna forma di locking per accept
    - File locking per accept
    - Mutex per proteggere accept
- Prethreaded server
  - Dispatcher/listener
    - Il thread principale invoca accept; non occorre passare il descrittore da un thread all'altro, perché i thread condividono tutti i descrittori
  - Leader/follower
    - Mutex per proteggere accept

Dimensione del pool

- Comportamento della dimensione del pool di processi o thread
  - Scelta significativa nell'architettura software di un server basato su processi o thread
- Alternative: dimensione statica o dinamica
- Pool di dimensione *statica* (ad es. *p*)
  - Carico alto: se i *p* processi o thread del pool sono occupati, una nuova richiesta deve attendere
  - Carico basso: la maggior parte dei processi o thread sono idle (spreco di risorse)
- Pool di dimensione *dinamica*
  - Il numero di processi o thread varia con il carico: cresce se il carico è alto, diminuisce se il carico è basso
  - Tipicamente, c'è un minimo numero di processi o thread idle
  - Esempio: Apache

Server ibrido

- Server basato su processi e thread
- Molteplici processi, ciascuno dei quali è multi-threaded
  - Un singolo processo di controllo (processo padre) lancia i processi figli
  - Ciascun processo figlio crea un certo numero di thread di servizio ed un thread listener
  - Quando arriva una richiesta, il thread listener la passa ad un thread di servizio che la gestisce
- Combina i vantaggi delle architetture basata su processi e basata su thread, riducendo i loro svantaggi
  - In grado di servire un maggior numero di richieste usando una minore quantità di risorse rispetto all'architettura basata su processi
  - Conserva in gran parte la robustezza e stabilità dell'architettura basata su processi

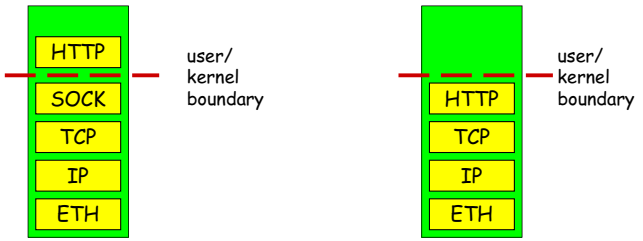
Server basato su eventi

- Un solo processo che gestisce le richieste in modo event-driven
  - Anziché servire una singola richiesta nella sua interezza, il server esegue una piccola parte di servizio per conto di ciascuna richiesta
- Uso di select(), opzioni non bloccanti sui socket, gestione asincrona dell'I/O
  - Il server continua l'esecuzione mentre aspetta di ricevere una risposta alla chiamata di sistema da parte del sistema operativo

Server basato su eventi (2)

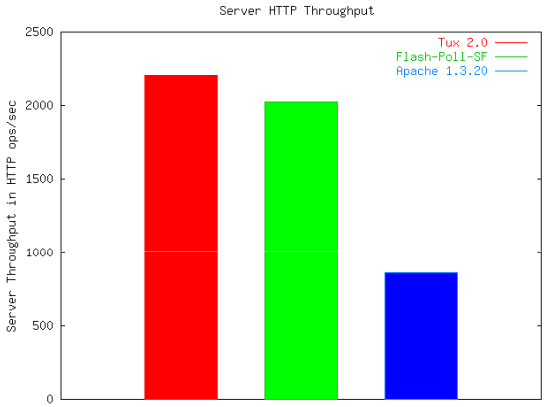
- Vantaggi
  - Molto veloce
  - La condivisione è intrinseca: un solo processo
  - Non c'è bisogno di sincronizzazione come nel server multi-threaded
  - Non ci sono overhead dovuti al context switching o consumi extra di memoria
- Svantaggi
  - Maggiore complessità nella progettazione ed implementazione
  - Meno robusto: una failure può fermare l'intero server
  - Limiti delle risorse per processo (es. descrittori di file)
  - Supporto in tutti i sistemi operativi di I/O asincrono

Server interno al kernel



- Thread del kernel dedicato per richieste HTTP:
  - Prima alternativa: tutto il server nel kernel
  - Seconda alternativa: gestione delle richieste GET statiche nel kernel, mentre richieste dinamiche gestite da un server (es. Apache) nello spazio utente
- Tux rimosso dal Linux kernel 2.6
  - Tuttavia, alcune distribuzioni (es. Fedora) lo hanno rimesso nel kernel 2.6

Confronto delle prestazioni



Tux: interno al kernel
Flash: basato su eventi
Apache 1.3: preforking

Fonte: E. Nahum, “Web servers: Implementation and performance”

- Il grafo mostra il throughput per Tux, Flash e Apache
- Esperimenti su P/II 400 MHz, gigabit Ethernet, Linux 2.4.9-ac10, 8 macchine client, WaspClient come generatore di carico