

Apache Server

Insegnamento di Sistemi Operativi di Rete
Master Universitario in Tecnologie Internet

Domenico De Guglielmo

E-mail: domenicodegu@virgilio.it Telefono: 050 2217468
Pervasive Computing & Networking Lab (PerLab) <http://www.perlab.it>
Dipartimento di Ingegneria dell'Informazione, Università di Pisa

- **Aspetti preliminari**
 - architettura di riferimento
 - cenni al protocollo HTTP
 - Uniform Resource Identifier (URI)
- **Configurazione del server web Apache**
 - file di configurazione
 - invocazione del server
 - processi di Apache
 - corrispondenza richieste/filesystem (mapping)
- **Virtual Host**
 - definizione e classificazione
 - name-based virtual host

Aspetti preliminari

- La maggioranza dei web server su Internet usano attualmente Apache HTTP Server.
- Il server Apache è implementato su piattaforma UNIX da un processo che prende il nome di demone `httpd`

Il protocollo HTTP

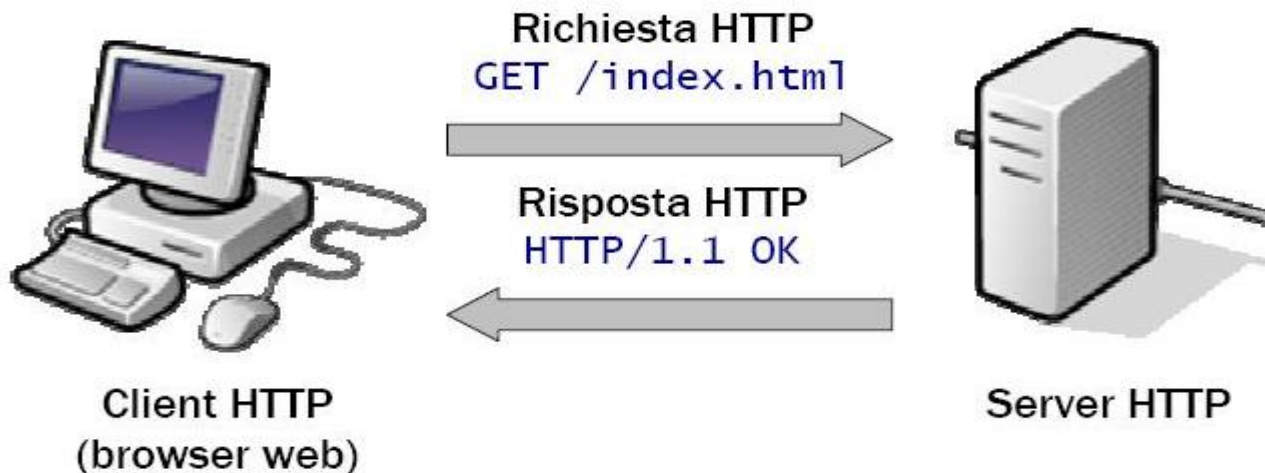


■ Architettura di riferimento

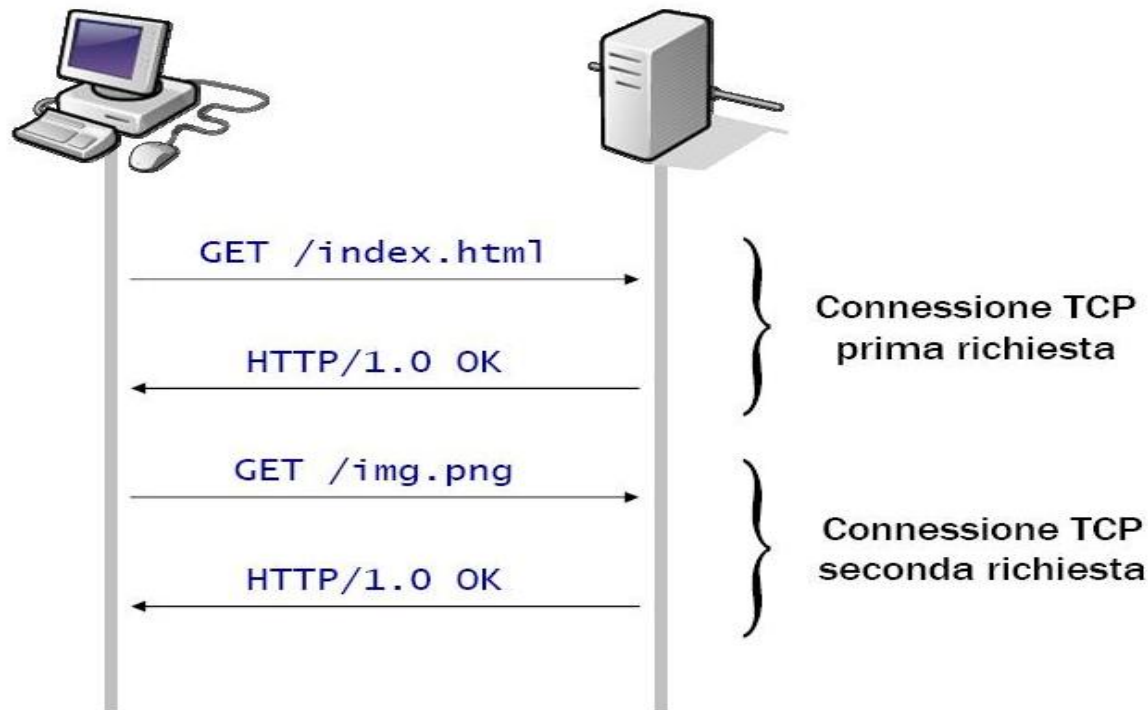
- client/server
- interazione richiesta/risposta

⇒ protocollo **stateless**

- tra una connessione e l'altra, il server non tiene nota della connessione precedente, e quindi tutte le connessioni sono trattate alla stessa maniera, come se si trattasse ogni volta di un nuovo client.

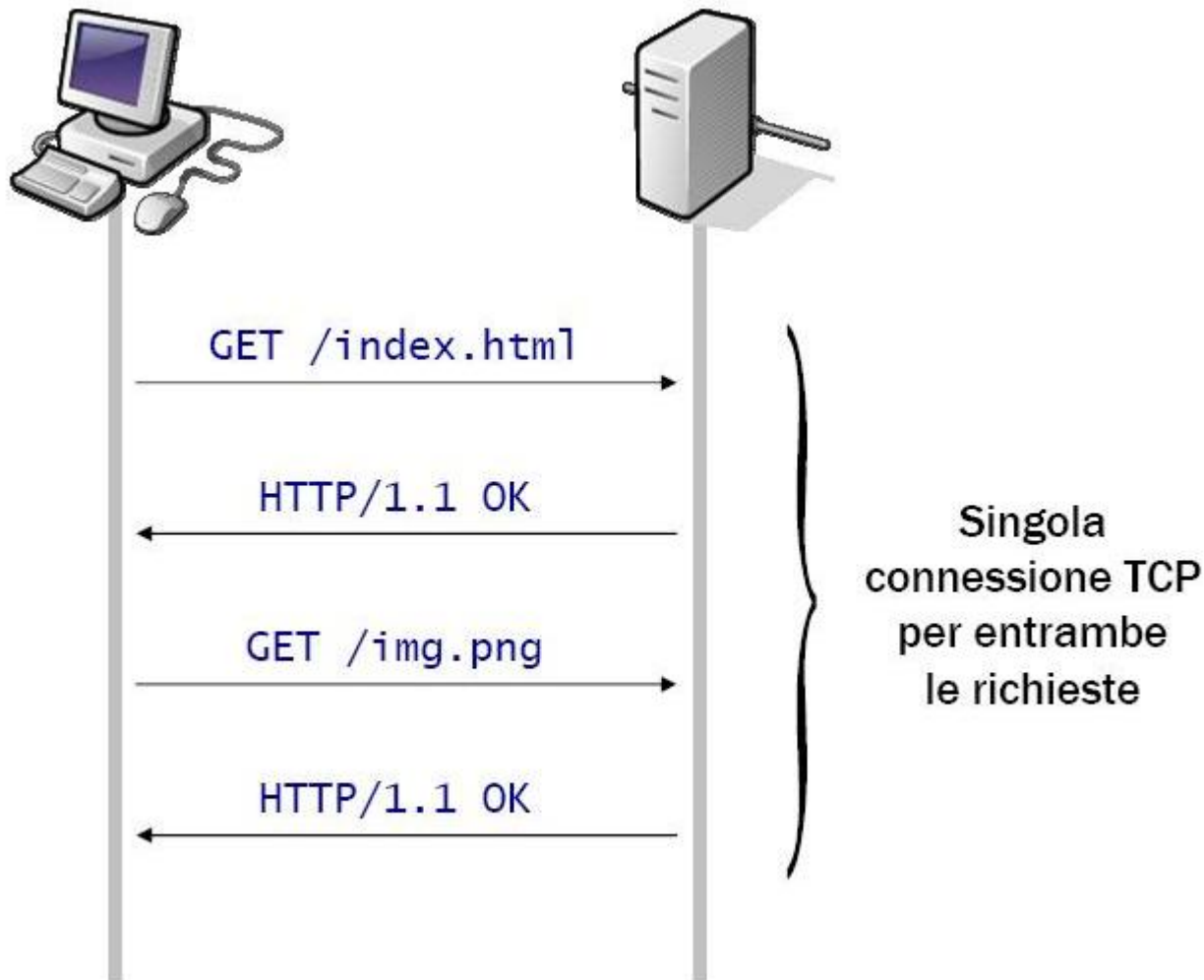


HTTP 1.0 e connessioni



- Client e server chiudono la loro connessione TCP al termine di ogni singola richiesta e la ristabiliscono per la richiesta successiva.

HTTP 1.1 e connessioni permanenti



Esempio di richiesta/risposta HTTP



■ Richiesta HTTP

GET /index.html HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0 (compatible; Konqueror/3.2; Linux)

Connection: Keep-Alive

■ Risposta HTTP

HTTP/1.1 200 OK

Date: Mon, 23 May 2005 22:38:34 GMT

Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)

Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT

Content-Length: 438

Connection: close

Content-Type: text/html; charset=UTF-8

Uniform Resource Identifier (URI)



- L'URI richiesto può riferirsi ad un file fisico, ad una risorsa dinamica prodotta da uno script esterno, oppure ad una risorsa generata da un modulo interno
- Il server deve sapere come individuare la risorsa, prima di poter effettuare decisioni successive: necessita la conversione da URI a risorsa presente sul server
- Le direttive standard `Alias`, `ScriptAlias` e `DocumentRoot` permettono di tradurre l'URI nel nome di un file presente nell'albero dei documenti

Configurazione del server web Apache

Processo padre e processi figli



- Inizialmente viene attivato un processo `httpd` **PADRE** che è sempre in ascolto di eventuali richieste per essere pronto a riceverle ma che non si occupa di gestirle.
- Il processo padre genera dei processi **FIGLI** (uguali a lui nel codice) ai quali affida l'effettiva gestione della richiesta.
- Il processo padre genera, appena lanciato, un certo numero di figli prima che arrivino le richieste e fa in modo che ci siano sempre almeno un determinato numero di figli liberi.
- Per questo motivo se osserviamo i processi attivi su una certa macchina subito dopo aver lanciato il server non vedremo un unico processo `httpd` ma una serie di processi `httpd`, che sono stati lanciati dal padre.

Invocazione del server



- L'interazione con il server può avvenire in diversi modi
 - invocazione manuale dell'eseguibile
 - invocazione dello script di gestione del server
 - utilizzo del comando `apachectl`
- In genere si preferisce non invocare manualmente l'eseguibile perché le altre due soluzioni sono
 - più comode
 - più generali

- **Comando ad hoc per operazioni sul server**
- **Lanciando lo script `apachectl` si può eseguire il demone `httpd`**
- `apachectl` **configura alcune variabili dipendenti dal SO**
- **Sono necessari i privilegi di `root`**
- **Sintassi**

`apachectl` *command*

apachectl: *command*



- `start`
 - **Avvia il server**
- `stop`
 - **Ferma il server**
- `restart`
 - **Riavvia il server**
 - **Usato dopo aver fatto modifiche al file di configurazione**
- `configcheck`
 - **Controlla il file di configurazione (senza ricaricarlo)**
 - **Utile in fase di test delle modifiche**
- `graceful`
 - **Riavvia Apache senza mandare in `abort` le connessioni correnti**

- Quando il web server è in esecuzione si può navigare il proprio sito web

`http://localhost/`

- La pagina di default che viene mostrata è

`/var/www/index.html`

Multi-Processing Module (MPM)

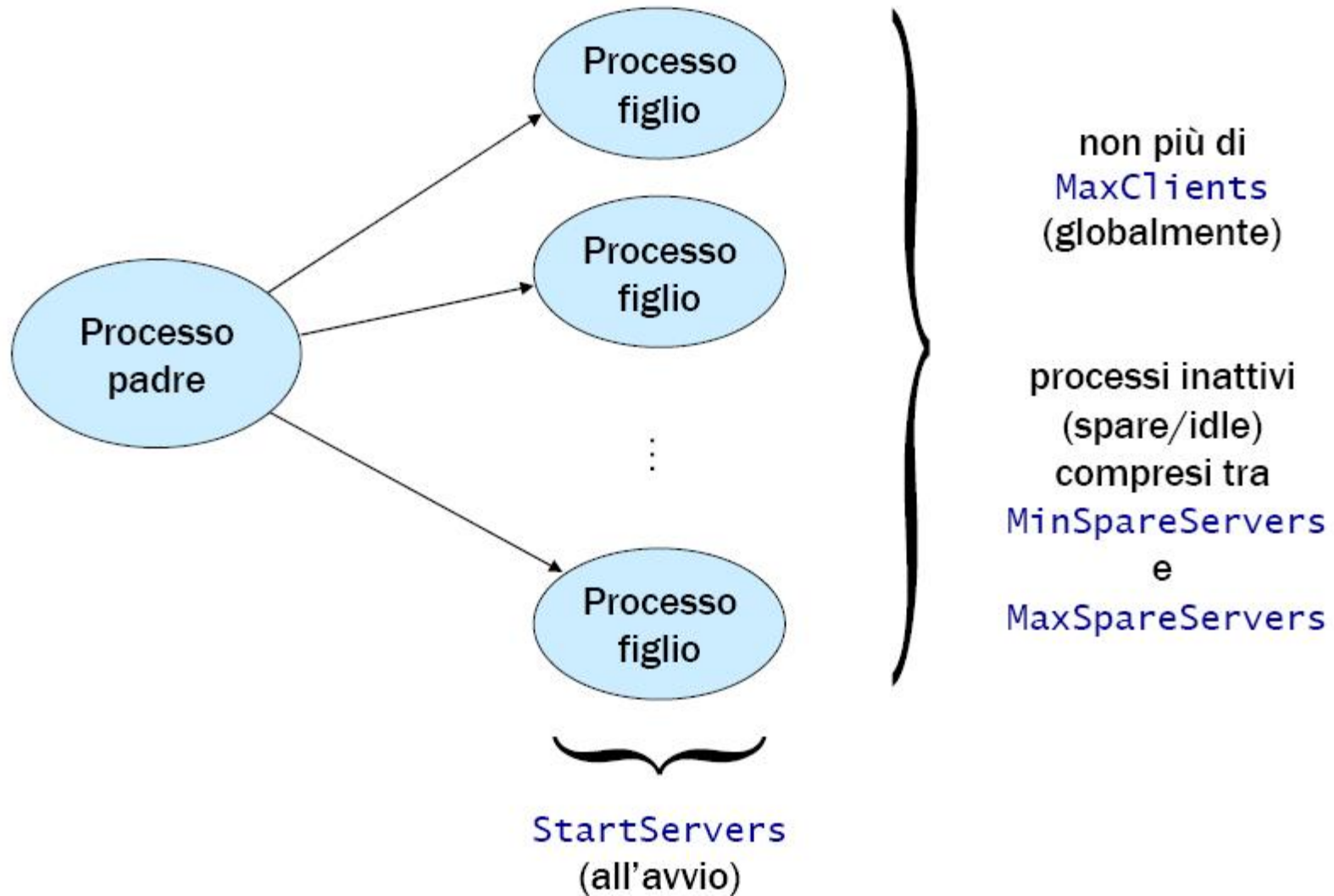


- Apache è stato progettato per essere flessibile su ogni tipo di piattaforma e con ogni configurazione d'ambiente
- È possibile aggiungere moduli MPM per la gestione di operazioni quali il binding, la gestione di processi/thread e le connessioni
- Un modulo MPM alla volta può essere caricato nel server
- Gli MPM definiscono il modo in cui il server realizza la concorrenza

- **prefork**
 - default
 - massima stabilità
 - server multi-processo con processi creati prima di servire le richieste
- `mpm_winnt`
 - default per windows
 - basato sui thread
- **worker**
 - massime prestazioni
 - server ibrido multi-processo/multi-thread
- `event`
 - variante sperimentale di worker

- **Esistono un processo principale (padre) ed alcuni processi ausiliari (figli) per il servizio delle richieste**
 - Il padre manda in esecuzione i figli (pre-forking dei figli)
 - I figli attendono le connessioni e le servono quando arrivano
- **Vantaggi**
 - I figli creati una sola volta e poi riutilizzati
 - ⇒ Non ho overhead causata dalle `fork()`
 - Maggiore semplicità, stabilità e portabilità rispetto ad un server multi-thread puro
- **Svantaggi**
 - Gestione del numero di figli
 - ⇒ presenza di processi figli idle
 - Come gestire dinamicamente il pool di processi figli?

MPM prefork



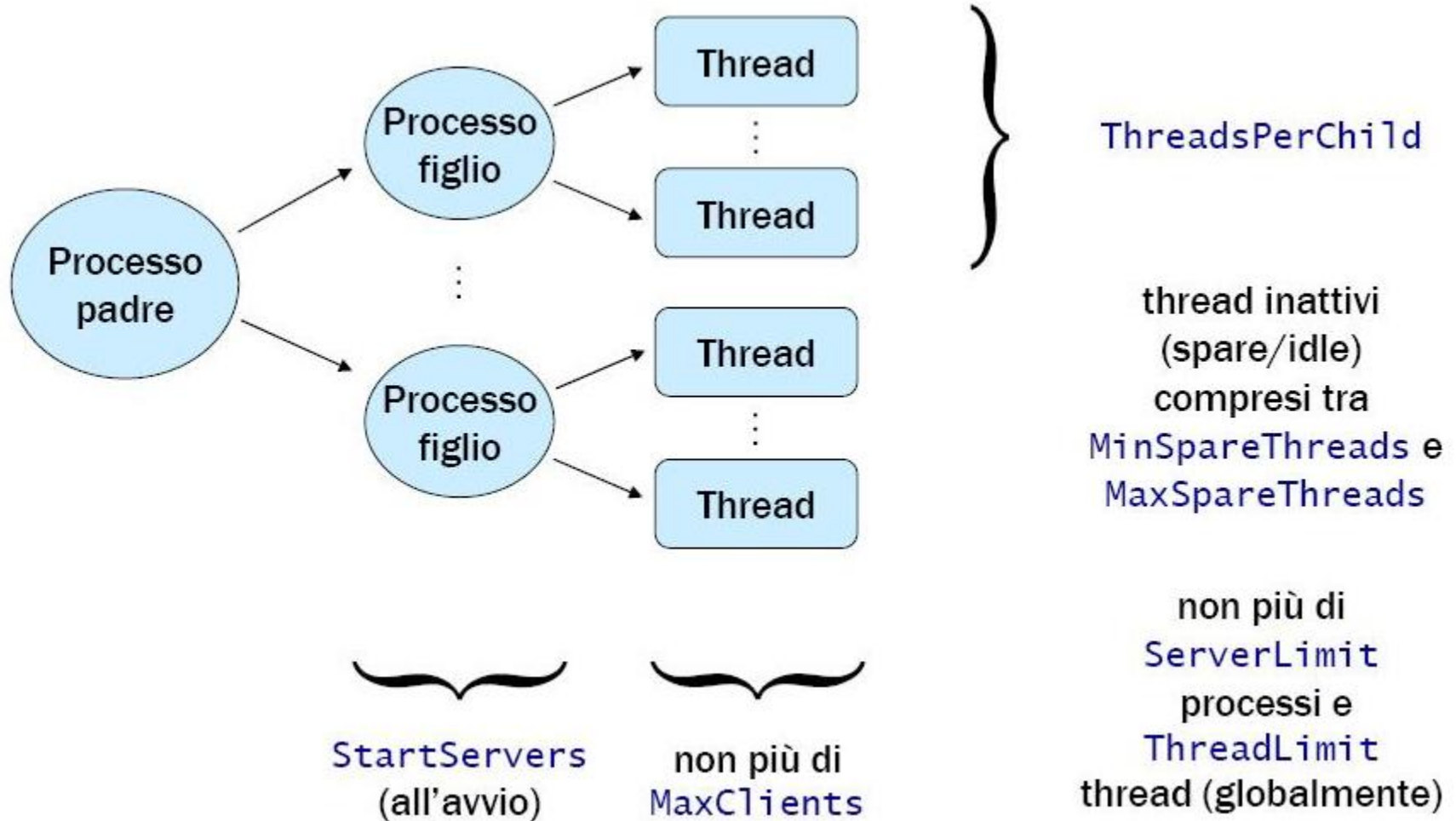
MPM prefork: direttive



- `StartServers` (**default 5**)
 - Numero dei processi figli in preforking
- `MaxClients` (**default 256**)
 - Limite sul numero di processi figli
 - Numero massimo di richieste servite contemporaneamente
- `MinSpareServer` (**default 5**) e `MaxSpareServers` (**default 10**)
 - Limite sul numero minimo e massimo di processi figli idle
- `MaxRequestsChild` (**default 10000**)
 - Numero massimo di richiesta HTTP servite da ciascun processo figlio
 - Allo scadere del numero di richieste il processo figlio termina
 - Può essere impostato a 0 cioè il processo figlio non termina

- **Architettura di server ibrida**
- **Un processo padre, molteplici processi figli, ciascuno dei quali genera multipli thread di esecuzione**
 - Il padre manda in esecuzione i figli
 - Ciascun figlio crea un numero fissato di thread server ed un thread listener
 - Quando arriva una richiesta, il listener la assegna ad un thread worker che la gestisce
- **Vantaggi**
 - Maggiore scalabilità e minor consumo di risorse del sistema
 - Stabilità simile (ma inferiore) ad un server multi-process puro
- **Svantaggi**
 - Maggiore complessità del codice del server (gestione dei thread)
 - Supporto del multi-threading da parte del SO

MPM worker



MPM worker: direttive



- **StartServers (default 3)**
 - **Preforking dei processi figli**
- **MinSpareThreads (default 75) e MaxSpareThreads (default 250)**
 - **Limite sul numero minimo e massimo di thread idle (complessivo per tutti i processi)**
- **ThreadsPerChild (default 25)**
 - **Numero di thread creati da ciascun processo figlio**
- **MaxClients (default $\text{ServerLimit} * \text{ThreadsPerChild}$)**
 - **Limite sul numero totale di thread**
 - **Numero massimo di richieste servite contemporaneamente**
- **ServerLimit (default 16)**
 - **Limite sul numero di processi figli attivi**
 - $\text{ServerLimit} \geq \text{MaxClients} / \text{ThreadsPerChild}$
- **ThreadLimit (default 64)**
 - **Limite sul numero di thread creati da ogni processo figlio**
 - $\text{ThreadLimit} \geq \text{ThreadPerChilds}$

Modifica del file di configurazione

File di configurazione



- Directory `/etc/apache2`
 - **`apache2.conf`**: principali opzioni di configurazione.
 - **`httpd.conf`**: vuoto e riservato alla personalizzazione della configurazione di Apache.
 - **`ports.conf`**: porte su cui il server sarà in ascolto all'avvio.

Altre directory



- File di log sugli accessi
 - `/var/log/apache2`
- Siti Web
 - `/var/www`

Direttive: impostazioni fondamentali



- `ServerType`
- `Listen`
- `ServerRoot`
- `KeepAlive`
- `KeepAliveTimeout`
- `ServerName`

- I servizi di rete possono venir avviati in due modi:
 - standalone
 - ⇒ processi in attesa di richiesta
 - ⇒ avviati al boot
 - ⇒ crea un processo figlio per ogni richiesta
 - inetd
 - ⇒ avvia servizi di rete solo quando un utente remoto li richiede permettendo di risparmiare risorse
 - ⇒ il server viene eseguito su richiesta da `inetd` che avvia un nuovo processo `httpd` per ogni richiesta

■ Sintassi

`ServerType standalone|inetd`

- In generale è preferibile usare `standalone` perché i server Web sono caratterizzati per la maggior parte del tempo in cui sono in esecuzione da numerose richieste; meglio quindi avviarli subito al boot e lasciarli attivi.

Listen



- Ad ogni macchina è associato un indirizzo IP che la identifica univocamente.
- Su una stessa macchina però possono essere attive diverse applicazioni
- Ai processi in esecuzione viene associata una porta
- È possibile fare in modo che il server stia in ascolto, cioè accetti, le richieste provenienti dalla porta specificata oppure da una combinazione di indirizzo IP-porta ben precisa
- La direttiva `Listen` è **sempre richiesta** in quanto la sua assenza dal file di configurazione provoca il fallimento dell'avvio del server
- Esempio: il server accetta connessioni su entrambe le porte 80 e 8080

`Listen 80`

`Listen 8080`

- **Sintassi:**

`Listen [IP-address:]portnumber`

- Specifica la gerarchia di directory di default per l'installazione di Apache.
- Ogni volta che all'interno del file `apache2.conf` si trova un path relativo come opzione di una direttiva, si può ricostruire il path assoluto antepoendovi la stringa che si trova nel `ServerRoot`.
- E' necessario avere i permessi adeguati sulla directory specificata
- Sintassi:

`ServerRoot directory-path`

- Default:

`ServerRoot /usr/local`

KeepAlive



- Nel protocollo di comunicazione HTTP 1.0, client e server chiudevano la loro connessione TCP al termine di ogni singola richiesta e la ristabilivano per la richiesta successiva.
- Creare una connessione richiede un tempo sufficientemente lungo, la direttiva `KeepAlive` offre la possibilità di sfruttare la stessa connessione TCP per più richieste successive.
- Riduzione del tempo necessario a scaricare un documento HTML con molte immagini.
- `On` : la connessione viene mantenuta attiva
 - Tramite `MaxKeepAliveRequest` è possibile stabilire un numero massimo di richieste permesse per connessione
- `Off` : la connessione viene chiusa ogni volta
- Sintassi:
`KeepAlive on | off`
- Default:
`KeepAlive on`

- Specifica quanto tempo aspettare la successiva richiesta sullo stesso canale di connessione prima di chiuderlo.
- Impostare un `KeepAliveTimeout` troppo alto potrebbe causare problemi alle prestazioni dei server molto carichi di lavoro
- Più alto sarà il timeout, più saranno i processi server occupati ad aspettare richieste su connessioni di client già sconnessi.

ServerName



- Definisce il nome simbolico e la porta che il server utilizza per identificare se stesso e che usa nel creare l'URL di risposta alle richieste dei client.
- Se non specificato il server tenta di dedurre il nome dell' host su cui sta girando facendo un'operazione di lookup inverso sul suo indirizzo IP.
- Se invece è solo la porta a non essere specificata, il server utilizzerà la stessa porta da cui è arrivata la richiesta.
- Esempio:
 - Se il nome della macchina che ospita il nostro server Web è `new.hostname.com`, ma la stessa macchina possiede anche un **DNS alias** `www.hostname.com`.
 - Se desideriamo che il server venga identificato con quest'ultimo nome, dobbiamo impostare la direttiva

`ServerName www.hostname.com:80`

▪ Sintassi

`ServerName fully-qualified-domainname[:port]`

- **Definisce l'indirizzo di posta elettronica del server Web che il server stesso include nei messaggi di errore, dovuti a problemi di funzionamento, che invia ai client.**
- **Sintassi**

```
Serveradmin email_address
```

Direttive: impostazioni di mapping



- DocumentRoot
- Alias
- UserDir
- DirectoryIndex

DocumentRoot



- **Specifica la directory a partire dalla quale si trovano tutti i documenti HTML e che il server utilizza per mappare le richieste**
- **Di default tutte le richieste sono girate a questa directory.**
- **Esempio**
 - `DocumentRoot /usr/local/www/data`
 - **Un client invia la richiesta**
`GET /ricerca/rmi.html http 1.0 accept text/html`
 - **Il server può ricostruire il path assoluto del file richiesto in:**
`/usr/local/www/data/ricerca/rmi.html`
- **Sintassi:**
`DocumentRoot directory-path`
- **Di default:**
`DocumentRoot /var/www`

Alias & ScriptAlias



- Mappano gli URL in modo alternativo rispetto al meccanismo definito da `DocumentRoot`.
- Tali direttive si trovano nella forma:

`Alias <directory passata> <directory reale>`

`ScriptAlias <directory passata> <directory reale>`

- *directory passata*

⇒ quella che il client scrive nell' URL all'atto della richiesta

- *directory reale*

⇒ quella che il server sostituisce nel momento in cui deve cercare il documento all'interno del file system.

Esempio: Alias



`Alias /icons/ "usr/local/www/icons"`

- **Quando il server incontra un URL del tipo**

`GET /icons/pagina http 1.0`

- 1. estrae la URI dalla GET**
- 2. esegue la sostituzione**
- 3. cerca il file**

`/usr/local/www/icons/pagina`

- Permette agli utenti di gestire le proprie pagine direttamente nelle loro home directory.
- Gli utenti creano una sottodirectory all'interno della loro home e vi depositano le loro pagine html, le immagini, gli script e tutti gli altri files che vogliono mettere a disposizione.
- Il nome di tale sottodirectory deve essere lo stesso per tutti gli utenti e deve coincidere con quello riportato dalla direttiva `UserDir`.
- Tutto quello che gli utenti depositeranno in queste sottodirectory sarà automaticamente accessibile da web all' URL:

`http://< indirizzo>/~<nome utente>`

- Affinché questo meccanismo funzioni è necessario che il server Apache possieda i diritti necessari per accedere alle sottodirectory delle home personali

- **Default**

`UserDir public_html`

DirectoryIndex



- Quando all'interno dell'URL richiesto dal client non viene specificato il nome di un file ma quello di una directory, il server restituirebbe l'elenco del suo contenuto se per quella directory è specificata l'opzione `Indexes`.
- E' possibile, sia per motivi di sicurezza sia per evitare che l'utente si perda tra un'infinità di file, mascherare tale contenuto grazie alla direttiva `DirectoryIndex`
- Ogni volta che all'interno dell'URL compare solo il nome di una directory, il server cerca se al suo interno esiste il file `index.html`, se lo trova restituisce al client tale file.
- Esiste la possibilità di specificare nella direttiva più di un file, in questo caso il server restituirà il primo che trova.
- Se non esiste nessuno dei file specificati e l'opzione `Indexes` per quella directory è attiva, il server genererà da solo la lista del contenuto della directory.

- **Sintassi:**

```
DirectoryIndex local_URL [local_URL]...
```

- **Default:**

```
DirectoryIndex index_html
```

Direttive: impostazioni di logging



- `ErrorLog`
- `LogLevel`

- I logfile di un server web sono utili per monitorare gli accessi e lo stato del server.
- Le informazioni memorizzabili nel logfile sono quelle che viaggiano negli header HTTP di messaggi di richiesta e risposta
- I server web permettono di definire quali campi dei messaggi devono essere memorizzati
- Le informazioni che maggiormente si vogliono trarre da un logfile riguardano:
 - Numero di utenti del sito e loro provenienza geografica
 - Browser utilizzati
 - Giorni e orari di maggiore affluenza
 - Pagine più popolari
 - Errori verificatisi per determinare la presenza di link sbagliati all'interno del sito
 - Siti che fanno riferimento al proprio sito

- **Definisce qual è il nome e la locazione del più importante file di log del server.**
- **Esempio**

```
ErrorLog /var/log/httpd-error.log
```

- Permette di stabilire una soglia di priorità che seleziona i messaggi da registrare nell'error log
- Solo i messaggi che hanno livello di priorità superiore o uguale a tale soglia sono memorizzati.
- Gli otto possibili livelli di priorità, in ordine crescente di priorità, sono:
 - debug
 - info
 - warn
 - error
 - crit
 - alert
 - emerg
- **Esempio:**
LogLevel crit
- Il server scriverà solo quei messaggi di errore che hanno una priorità: crit, alert, emerg.

Direttive: gestione dei processi



- MinSpareServers
- MaxSpareServers
- StartServers
- MaxClient
- MaxRequestsPerChild
- User
- Group

MinSpareServers, MaxSpareServers, StartServer



- MinSpareServers
 - **Numero minimo di figli che devono essere mantenuti sempre liberi**
 - **Ogni volta che il padre accetta una nuova richiesta controlla se deve generare o meno un nuovo figlio per mantenere costante questo numero.**
- MaxSpareServers
 - **Specifica il numero massimo di figli che possono rimanere inattivi.**
 - **Se tale numero viene superato il padre termina i processi figli in eccesso.**
- StartServers
 - **Numero di processi che il padre crea appena viene lanciato.**

MaxClients & MaxRequestsPerChild



- `MaxClients`
 - Numero massimo di figli che il server tiene attivi contemporaneamente
 - Numero massimo di clienti che possono essere serviti insieme.
 - Le richieste che superano questo numero vengono messe in coda finché il server non è di nuovo disponibile.
- `MaxRequestPerChild`
 - Numero massimo di richieste che un processo figlio può gestire prima di venire forzatamente terminato dal padre.

- I processi figli gestiscono effettivamente le richieste quindi sono loro ad accedere alle risorse richieste dai clienti
- Non è opportuno che essi abbiano i permessi di root.
- Le direttive `User` e `Group` permettono di definire l'utente e il gruppo fittizi da assegnare all'insieme di processi `httpd` secondari che si occupano di servire direttamente le richieste dei client.
- Tra le possibili opzioni di queste direttive si preferisce scegliere:

`User www`

`Group www`

Direttive: gestione dei thread



- `ThreadsPerChild`
- `MinSpareThreads`
- `MaxSpareThreads`
- `ServerLimit`

ThreadsPerChild



- **Numero di thread creati da ciascun processo figlio**
- **Default 25**

MinSpareThreads & MaxSpareThreads



- MinSpareThreads
 - **Limite sul numero minimo di thread idle**
 - **Complessivo per tutti i processi**
 - **Default 75**
- MaxSpareThreads
 - **Limite sul numero massimo di thread idle**
 - **Complessivo per tutti i processi**
 - **Default 250**

- **Limite sul numero di processi figli attivi**
- `ServerLimit >=`
`MaxClients/ThreadsPerChild`
- **Default 16**

Virtual Host

- **Situazione tipica**
 - Un sito corrisponde ad un singolo server web in esecuzione su un certo indirizzo
 - Nel caso di siti multipli
 - ⇒ tante istanze del server web quanti siti
 - ⇒ ogni sito corrisponde ad un indirizzo IP distinto
- **Problemi**
 - Elevata richiesta di risorse
 - Ogni server web richiede
 - ⇒ un suo indirizzo IP
 - ⇒ un suo file di configurazione

Più siti sullo stessa macchina server



■ Vantaggi

■ maggiore praticità

⇒ la centralizzazione semplifica le operazioni di amministrazione

■ maggiore flessibilità

⇒ non è necessario avere un indirizzo IP per ogni sito

■ prestazioni migliori

⇒ non essendoci istanze multiple del server si riduce l'occupazione delle risorse

■ Usi tipici

■ intranet aziendale

■ Internet provider

- Un elemento dell'insieme di siti in esecuzione su una singola macchina
- I virtual host permettono agli utenti di ospitare infiniti siti sul proprio PC.
- Due architetture possibili:
 - Molteplici demoni `httpd`
 - Un singolo demone `httpd` (Apache)

Tipi di Virtual Hosting



- IP-based virtual host
 - Il sito è individuato dall'indirizzo IP a cui è diretta la richiesta
 - Ogni sito ha un diverso indirizzo IP
 - Il server è dotato di uno o più indirizzi IP (reali o virtuali)
 - Una o più NIC a cui sono associati uno o più indirizzi IP
- name-based virtual host
 - Il sito è individuato dal suo nome (indirizzo simbolico)
 - Usa gli header HTTP/1.1 per scoprire l'hostname
 - Molti domini diversi possono condividere lo stesso indirizzo IP
 - Al singolo indirizzo IP sono associati più nomi di dominio a livello DNS
 - Una o più NIC a cui sono associati uno o più nomi logici

Virtual host (operazioni)



1. Per attivare il Virtual Hosting basato sui nomi:

- Creare il file `/etc/apache2/conf.d/virtual.conf` e al suo interno scrivere **NameVirtualHost ***

2. Vogliamo creare 2 siti: *www.sito1.com* e *www.sito2.com*

- Creiamo i file `/etc/apache2/sites-available/www.sito1.com` e `/etc/apache2/sites-available/www.sito2.com`
- Al loro interno digitare:

```
<VirtualHost *>  
    ServerName www.sito1.com  
    DocumentRoot /var/www/sito1  
</VirtualHost>
```

```
<VirtualHost *>  
    ServerName www.sito2.com  
    DocumentRoot /var/www/sito2  
</VirtualHost>
```

Virtual host (operazioni)



3. Digitare i seguenti comandi:

⇒ `$ a2ensite www.sito1.com`

⇒ `$ a2ensite www.sito2.com`

4. Reload del server:

`$ /etc/init.d/apache2 reload`