

The background features a perspective view of a grid of squares, with the left side being a darker blue and the right side fading into a lighter blue. A bright, circular light source is visible in the upper right quadrant, creating a lens flare effect. The text 'IL WEB' is centered in the lower half of the image.

IL WEB

World Wide Web (WWW)

Sistema che permette la condivisione di documenti ipertestuali multimediali (costituiti da un insieme di contenuti testuali, visuali e audio/video) sfruttando l'infrastruttura WAN della rete Internet.

WEB

- Nato come **sistema di gestione di ipertesti in ambiente distribuito**, il **Web** ha poi assunto i più svariati ruoli, da veicolo di contenuti e flussi multimediali ad'interfaccia per applicazioni interattive.
- Nasce al CERN nel 1989 e viene “regalato” al pubblico nel 1991.
- Si basa sul protocollo **HTTP** (**HyperText Transfer Protocol**).
- Poggia su di un'architettura **Client / Server**.
- Servizio più diffuso di Internet (Internet è il Web?!?).

HTTP

- La prima versione del protocollo **HTTP (1.0)** prevedeva che, in una sessione, il **client** (browser) potesse fare una singola richiesta di una risorsa (un `pathname`) al **server**, e che quest'ultimo potesse solo rispondere a tali richieste.
- Il protocollo **HTTP 1.1** introduce la possibilità di avere più richieste per connessione, di avere connessioni permanenti e, all'interno di queste, di mandare/ricevere richieste/risposte asincrone.
- Il protocollo obbliga inoltre i client a specificare, nella richiesta, qual è l'**hostname** dal quale si vuole ottenere la risorsa.
 - Questo permette di implementare il **virtual hosting**, in cui un server può ospitare più siti internet raggiungibili usando nomi diversi, sebbene quest'ultimi corrispondano tutti allo stesso ip.

WEB Server

- È un'applicazione software che, in esecuzione su un (host) server, è in grado di gestire le richieste di trasferimento di pagine web verso un client, di solito un web browser.
- La comunicazione tra server e client avviene tramite il protocollo **HTTP**, che utilizza la porta **TCP 80**, o eventualmente la versione sicura **HTTPS**, che utilizza invece la **TCP 443**.

WEB Server

- Esistono molti programmi che fungono da **Web Server**...
- Tutti i **web server** che vedremo forniscono pagine web standard allo stesso modo (**streaming di caratteri**): per intenderci sono tutti in grado di gestire richieste **HTML** e **CSS**.

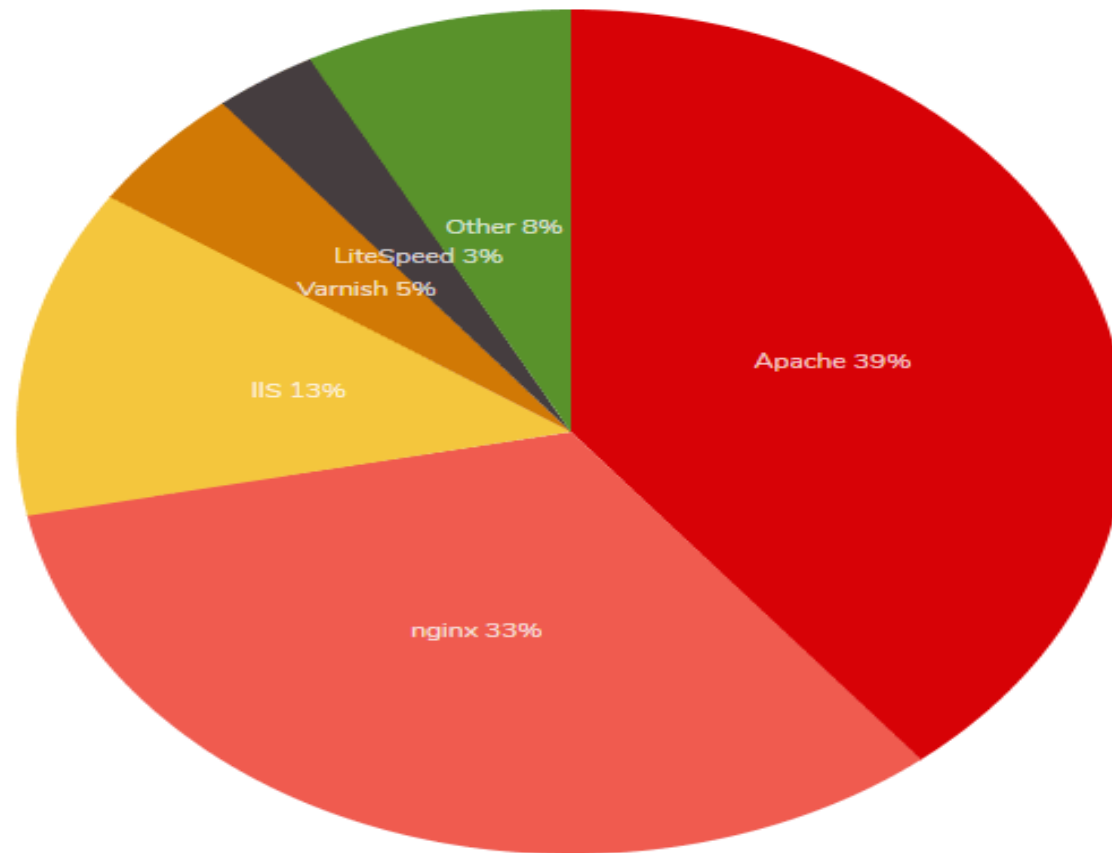
L'interpretazione del codice inviato è fatta dal browser, che riceve dal server un semplice **file di testo**.

- Ma ognuno di essi ha delle peculiarità che gli permettono di gestire applicazioni dinamiche diverse.

WEB Server

- **NCSA HTTP**: il più diffuso prima di Apache.
- **Apache HTTP Server**: fino a qualche tempo fa l'unico che, grazie ad una tecnologia modulare, è in grado di gestire applicazioni di tutti i tipi: php, python, java,..... è tra i più diffusi!
- **Apache Tomcat**:(sviluppato dalla Apache Software Foundation) fornisce un sistema di gestione delle servlet java.
- **Light httpd**: web server leggero adatto a piccole applicazioni (come ad esempio `phpmyadmin`).
- **Internet Information Services**: (**IIS**, sviluppato da Microsoft) permette la gestione di applicazioni .NET, nelle nuove versioni dovrebbe funzionare anche php, python ecc....
- **Engine-x**:(**nginx**) è un web server leggero che ha preso piede molto rapidamente, adatto a servire (velocemente) pagine statiche (e dinamiche), può fungere da reverse proxy e load balancer.

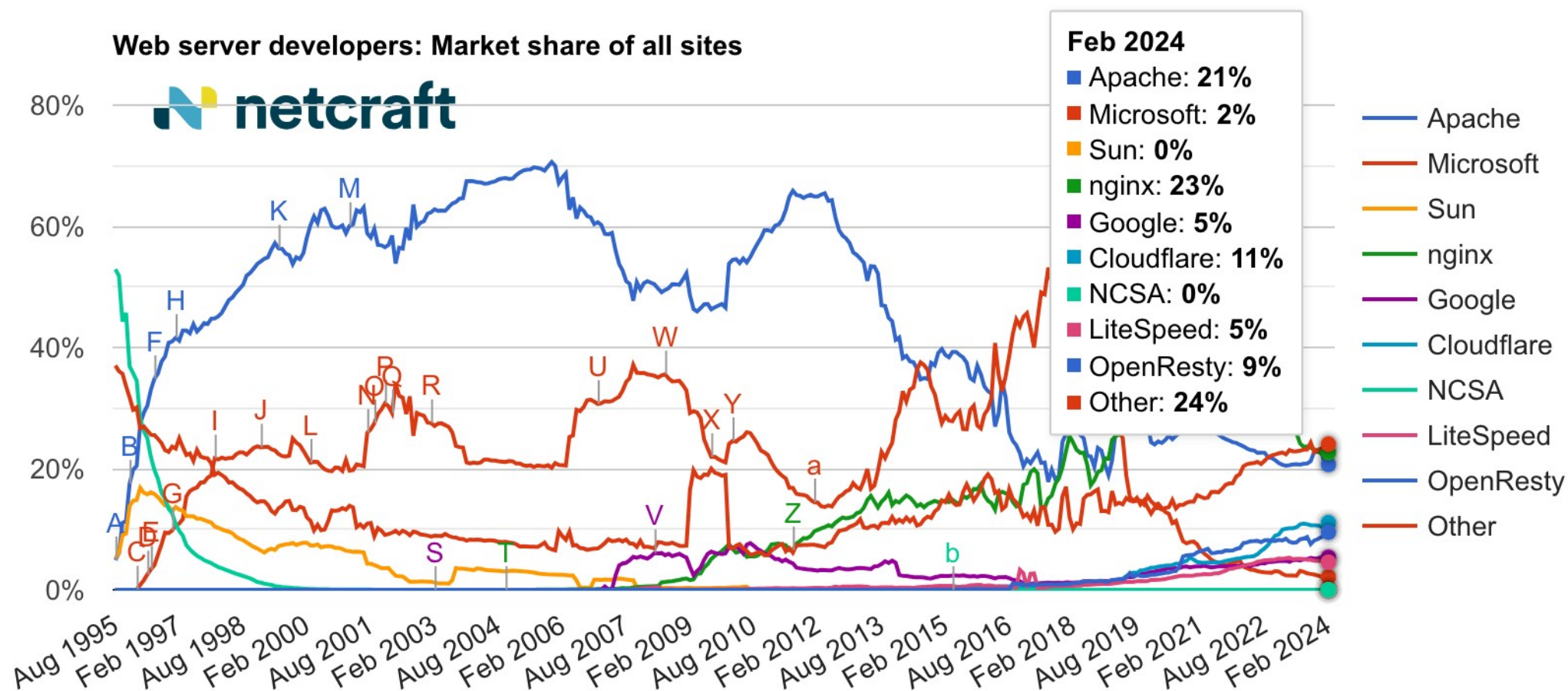
Diffusione dei vari Web Server (2019)



Top In Web Server Usage Distribution in the Top 1 Million Sites

Technology	Websites	%
 Apache	401,286	40.13
 nginx	334,870	33.49
 IIS	128,835	12.88
 Varnish	49,123	4.91
 LiteSpeed	30,849	3.06
 Apache Tomcat Coyote	15,298	1.53
 Microsoft	1,000	1.00

Fonte: <https://www.netcraft.com/blog/march-2024-web-server-survey/>



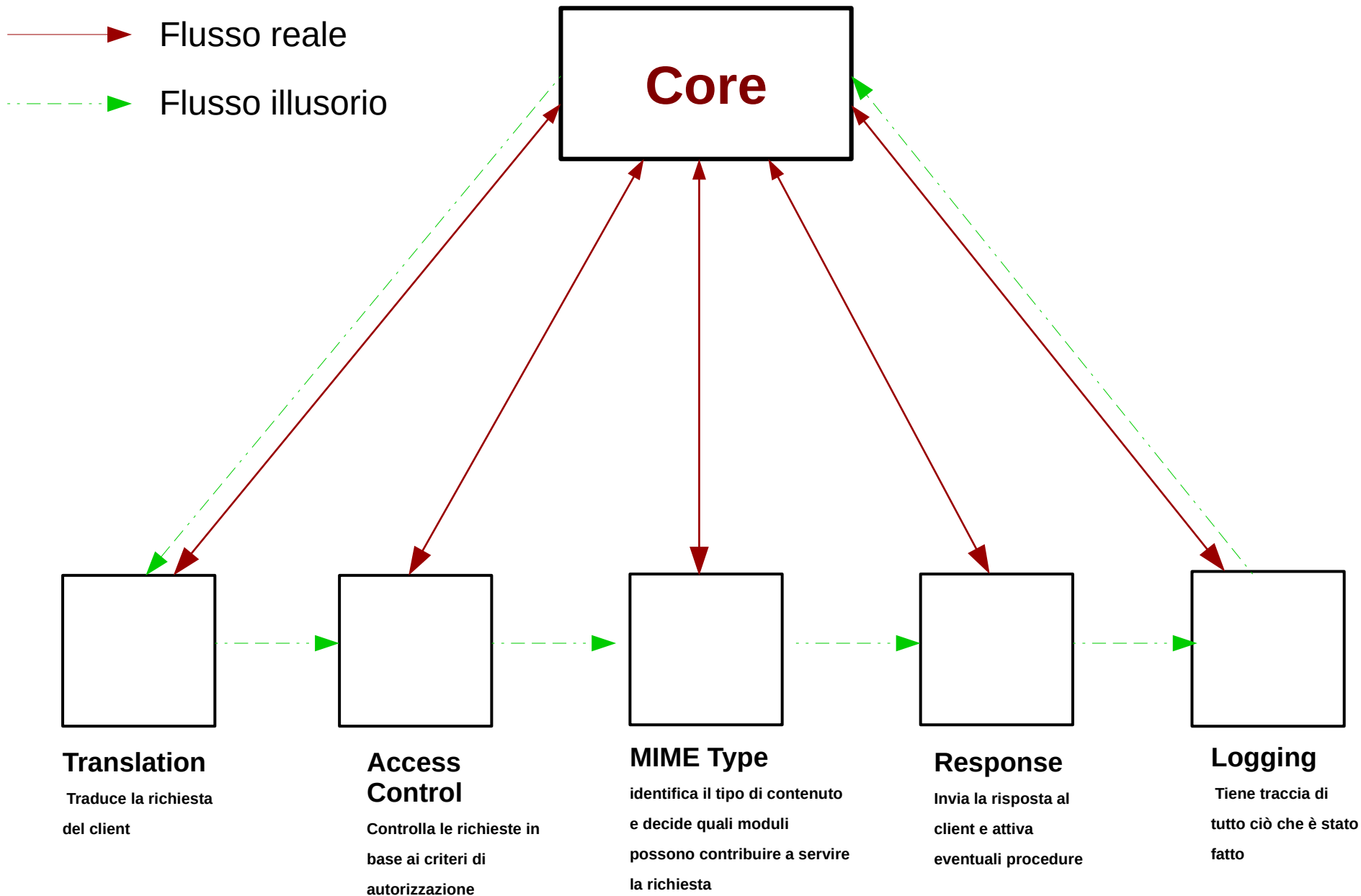
Apache HTTP Server

- In grado di operare su una grande varietà di sistemi operativi, tra cui UNIX/Linux, Microsoft Windows e OS X.
- L'architettura è **composta da un demone (servizio) che, sulla base delle impostazioni contenute nel file di configurazione `httpd.conf` (o `apache.conf`), permette l'accesso a uno o più siti, gestendo varie problematiche di sicurezza e potendo ospitare diverse estensioni per pagine attive (o dinamiche), come PHP o Jakarta/Tomcat o Python o Ruby o CGI.**

Apache HTTP Server

- Presenta **un'architettura modulare**, quindi ad ogni richiesta del client vengono svolte funzioni specifiche da ogni modulo di cui è composto, come unità indipendenti. Ciascun modulo si occupa di una funzionalità, ed il controllo è gestito dal **core**.
- Il **core** è sostanzialmente un *demone che esegue un ciclo di polling*, attraverso il quale vengono interrogate continuamente le linee logiche da cui possono pervenire messaggi di richiesta (ad esempio la porta 80 e/o 443).
- Il **core** passa poi la richiesta ai vari moduli in modo **sequenziale**, usando i parametri di uscita di un modulo come parametri di accesso per il successivo, creando così l'illusione di una *comunicazione orizzontale fra i moduli*.

Apache HTTP Server



Apache HTTP Server 2.X – Moduli MPM

- Apache >= 2.0 estende questo design modulare alle funzioni più basilari di un server web: viene fornito con una selezione di moduli **multi-processing (MPM)** che sono responsabili della connessione alle porte di rete della macchina, dell'accettazione delle richieste e dell'invio ai figli per la gestione delle richieste.
- I moduli **MPM** rappresentano l'evoluzione di **Apache** e i diversi modi in cui il web server è stato modificato **per gestire le richieste HTTP entro i vincoli di elaborazione del tempo nel corso della sua lunga storia.**
- **Sono moduli che** controllano la gestione dei processi di Apache.
- Apache è progettato per gestire più richieste contemporaneamente e gli **MPM** determinano come queste richieste vengono elaborate.
- Diversi **MPM** utilizzano strategie diverse per gestire più richieste e ognuna ha i suoi vantaggi e svantaggi.
- Sono essenzialmente tre:
 - **prefork**
 - **Worker (thread)**
 - **Event (thread)**

Apache HTTP Server 2.X – Moduli MPM

- **mpm_prefork:** Esegue una serie di processi figlio per soddisfare le richieste e i processi figlio servono **solo una richiesta alla volta**.
- Poiché ha più processi in attesa e non gestendo le richieste creando nuovi thread, è il più veloce dei moduli **MPM threaded** quando si ha a che fare con una singola richiesta alla volta.
- Nel caso di richieste simultanee che eccedano il numero di processi, esse saranno accodate in attesa di un processo libero che le gestisca.
- **Si utilizza per moduli non thread-safe come php.**
- Può richiedere molta RAM.

Apache HTTP Server 2.X – Moduli MPM

- **mpm_worker:** utilizza i thread, per migliorare la gestione delle richieste concorrenti.
- In avvio vengono creati dei processi figlio che restano in attesa di richieste. Sostanzialmente ad ogni **richiesta il processo che la riceve genera un thread che si occupa di gestirla.**
- Gestisce le richieste concorrenti molto più facilmente, poiché le connessioni devono solo attendere un thread libero (che di solito è disponibile) invece di un processo di riserva in **prefork.**
- Richiede meno RAM.
- Non adatto alla gestione di moduli dinamici non **thread safe.**

Apache HTTP Server 2.X – Moduli MPM

- **mpm_event:** è simile al modulo **worker**.
- La differenza principale rispetto al modulo **worker** è che utilizza un thread dedicato per gestire le connessioni mantenute attive e passa le richieste ai thread figlio solo quando è stata effettuata una richiesta.
- Presente da **APACHE 2.4**.
- Più performante del modulo **worker**, meno esoso in termini di ram del modulo **prefork**, non adatto a moduli dinamici non **thread-safe**.
- Ottimo per la gestione di accessi concorrenti di client che non sono necessariamente tutti attivi contemporaneamente, ma che fanno richieste occasionali.

Apache HTTP Server - Configurazione

- Apache normalmente va a leggere i file che compongono il sito nella cartella `/var/www/html`. Questa posizione viene determinata tramite la clausola **DocumentRoot**.
- **DocumentRoot** può essere cambiata! Una volta si usava mettere `/srv/www`.
- In Debian/Ubuntu, tutti i file e le directory di configurazione di **apache** risiedono in `/etc/apache2/`.
- In particolare:
 - apache2.conf**: file con la configurazione iniziale di apache.
 - ports.conf**: file specifica le porte di ascolto del demone apache.
 - conf-available**: directory che contiene configurazioni aggiuntive disponibili.
 - conf-enabled**: directory che contiene configurazioni aggiuntive attive.
 - mods-available**: directory che contiene moduli disponibili.
 - mods-enabled**: directory che contiene moduli attivi.
 - sites-available**: directory che contiene siti disponibili.
 - sites-enabled**: directory che contiene siti attivi.

Apache HTTP Server - Modulo mpm-prefork

- Si configura tramite il file:
`/etc/apache2/mods-available/mpm-prefork.conf`.
- Particolarmente importante perchè contiene alcuni parametri di start up, tra cui il **numero di istanze di apache** da lanciare all'avvio!

```
<IfModule mpm_prefork_module>
```

```
    StartServers 5
```

```
    MinSpareServers 5
```

```
    MaxSpareServers 10
```

```
    MaxRequestWorkers 150
```

```
    MaxConnectionsPerChild 0
```

```
</IfModule>
```

Apache HTTP Server -Modulo mpm-prefork

- La configurazione di questo modulo permette il tuning di **apache**, inserendo i valori corretti si possono aumentare le prestazioni (per i test si utilizza il comando `ab`: Apache Benchmark).
- A seconda della dotazione hardware del server, si imposta un numero prefissato di processi di avvio: `StartServers=N`, di conseguenza `MinSpareServers=StartServers=N` e `MaxSpareServers=2*StartServers=2*N`. Gli altri parametri servono per un tuning più fine e di solito non vengono toccati.
- Ad esempio, se `N=20`

```
<IfModule mpm_prefork_module>
```

```
    StartServers    20
```

```
    MinSpareServers 20
```

```
    MaxSpareServers 40
```

```
    MaxRequestWorkers 150
```

```
    MaxConnectionsPerChild 0
```

```
</IfModule>
```

Apache HTTP Server – Configurazione sito di default

- In sites-enabled troverete un link simbolico al sito di default:
`/etc/apache2/sites-enabled/000-default.conf`

```
<VirtualHost *:80>
```

```
ServerAdmin webmaster@localhost
```

```
DocumentRoot /var/www/html
```

```
<Directory /var/www/html>
```

```
Options Indexes FollowSymLinks MultiViews
```

```
AllowOverride all
```

```
Order allow,deny
```

```
allow from all
```

```
</Directory>
```

```
ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
CustomLog ${APACHE_LOG_DIR}/access.log combined
```

```
</VirtualHost>
```

Apache HTTP Server – Configurazione

- Questa è la configurazione di base di **apache**, se create una pagina html (`mypage.html`) e la copiate in `/var/www/html` potete puntare il browser verso `http://<ipvostravm>/mypage.html`.
Come abbiamo fatto prima puntando a `http://<ipvostravm>/` In tal caso, non abbiamo specificato nessun documento html... di default apache risponde con la pagina di nome `index.<html|php>` se esiste!
- Se puntate il browser su `http://<ipvostravm>/phpmyadmin` trovate l'interfaccia di gestione di MySQL.
- Nella directory `/var/log/apache2` trovate i file di log di **apache**, dateci un'occhiata con i comandi `cat` e/o `tail -f`.

VirtualHost

- **Permette ad un server web di ospitare più di un dominio.**
- Utile in ambienti condivisi di web hosting poichè permette di collocare centinaia di siti web in un unico server fisico.
- Implementato nei moderni software di web server: **Apache, Nginx, IIS, Lighttpd.**
- In **Apache** è implementato dalla direttiva **VirtualHost.**

VirtualHost - Esempio

- Il server web del DAIS ospita svariati siti web.
- Un esempio di questi è collegato ad un progetto denominato *copernicus*.
- Si vuole che il server Web risponda alle richieste per `copernicus.dais.unive.it` e `copernicus.dsi.unive.it`
- Anzitutto è necessario definire una entry nel server **DNS** che faccia puntare sanitaveneto al server web:

```
copernicus A 157.138.20.11 #in questo caso \  
bisogna inserire anche il reverse
```

oppure

```
copernicus CNAME www  
www A 157.138.20.11
```

VirtualHost - Esempio

- Successivamente bisogna creare la directory dove il sito sarà ospitato e i relativi file di log:

```
# sudo mkdir /var/www/copernicus  
# sudo mkdir /var/log/apache2/copernicus
```

- Ricordiamoci di dare gli accessi alla directory del sito all'utente `www-data`:

```
# sudo chown -R www-data:www-data /var/www/copernicus
```

- Creiamo il file di configurazione del VirtualHost:

```
# sudo [vi|nano] /etc/apache2/sites-available/copernicus.conf
```


VirtualHost – Esempio

- E ci copiamo dentro questa configurazione:

```
<VirtualHost *:80>
    ServerAdmin webmaster@dsi.unive.it
    ServerName copernicus.dais.unive.it
    DocumentRoot /var/www/copernicus/
    ErrorLog /var/log/apache2/copernicus.error.log
    LogLevel warn
    CustomLog /var/log/apache2/copernicus.access.log combined
    ServerSignature Off
</VirtualHost>
```

- Ora non resta che attivare il sito e riavviare apache:

```
# sudo a2ensite copernicus.conf

# sudo service apache2 restart
oppure
# sudo systemctl restart apache2
```

Apache comandi utili

- `# sudo a2ensite <nomesito>` abilita un sito web presente in `sites-available`.
- `# sudo a2dissite <nomesito>` disabilita un sito web presente in `sites-enabled`.
- `# sudo a2enmod <modulo>` abilita un modulo di **apache** disponibile in `mods-available`.
- `# sudo a2dismod <modulo>` disabilita un modulo di **apache**
- `# ab apache benchmark`.
- `# apachectl configtest` **apache** configuration test.
- `# sudo service apache2 <start|restart|stop>`
- `# sudo systemctl <start|restart|stop> apache2`

Apache comandi utili (sunto)

Apache2 Service Commands	Apache2 Important Files
Starting Apache2 \$ sudo /etc/init.d/apache2 start \$ sudo service apache2 start \$ sudo apachectl -k start	Apach2 Syntax check \$ apachectl configtest \$ apachectl -t
Restarting Apache2 \$ sudo /etc/init.d/apache2 restart \$ sudo service apache2 restart \$ sudo apachectl -k restart	Apache2 Web root \$ /var/www/html - default \$ /var/www/ - New domain location
Stopping Apache2 \$ sudo /etc/init.d/apache2 stop \$ sudo service apache2 stop \$ sudo apachectl -k stop	Enable / Disable Virtual Hosts \$ sudo a2ensite xxxx.conf \$ sudo a2dissite xxxx.conf
Status Apache2 \$ sudo /etc/init.d/apache2 status \$ sudo service apache2 status	Loaded apache2 Modules \$ apachectl -M \$ apache2ctl -M
Reload Apache \$ sudo /etc/init.d/apache2 reload \$ sudo service apache2 reload \$ sudo apachectl -k reload	Apache2 Config file's \$ /etc/apache2/apache2.conf \$ /etc/apache2/ports.conf \$ /etc/apache2/sites-available/xxx.conf
Apache2 Graceful \$ sudo apachectl -k graceful \$ sudo apachectl -k graceful-stop	Available apache2 Modules \$ /usr/lib/apache2/modules/
	Apache2 log file's \$ /var/log/apache2/error.log \$ /var/log/apache2/access.log

SSL

- **La crittografia è importante per la sicurezza, deve essere usata in tutte quelle applicazioni che richiedono autenticazione (come ad esempio una form web) e che trasmettono dati sensibili.**
- Non protegge al 100% da attacchi informatici ma aiuta a rendere difficile il lavoro dei Cracker.
- Può essere usata per attaccare gli utenti (**Ransomware**). Ad esempio **Cryptolocker**.

SSL

- **Transport Layer Security (TLS)** e il suo predecessore **Secure Sockets Layer (SSL)**, sono dei protocolli crittografici usati nel campo delle telecomunicazioni e dell'informatica che permettono una comunicazione sicura dalla sorgente alla destinazione (end-to-end) su reti TCP/IP fornendo autenticazione, identità (tramite certificati), integrità dei dati e cifratura, operando al di sopra del livello di trasporto.
- Un esempio di applicazione di **SSL/TLS** è nel protocollo **HTTPS**.
- Ma anche **smtps**, **pop3s**, **imaps**, **ldaps** ecc...
- Vi sono varie implementazioni di **SSL**, quella più famosa è probabilmente **OpenSSL** implementata per tutti i sistemi Linux...

OpenSSL

- Nato nel 1998, il progetto **OpenSSL** è un'implementazione open source del protocollo **SSL/TLS**.
- La libreria principale è scritta in **linguaggio C** e implementa funzioni di crittografia basilari fornendo strumenti per l'attivazione di varie funzionalità avanzate.
- Ne esistono versioni per la gran parte dei sistemi operativi UNIX (Solaris, Linux, OS X e alcune versioni open source di BSD) e Microsoft Windows.
- Ideato come un set di strumenti open source per la crittografia del codice e dello scambio di dati, oggi è utilizzato da circa il 70% dei server della rete.

A close-up of Gene Wilder as Willy Wonka, wearing his signature purple velvet suit, a brown bow tie, and a brown top hat. He has a slight, knowing smile and is resting his head on his right hand.

**TI PREGO
RACCONTAMI DI COME**

**TI SENTI AL
SICURO CON OPENSSL...**

OpenSSL

- **OpenSSL** ha conquistato, qualche anno fa, suo malgrado, l'attenzione della stampa internazionale – e non solo – a causa di una sua falla di sicurezza sfruttata dall'attacco **Heartbleed**.
- **OpenSSL** permette di utilizzare sia certificati rilasciati da **Certification Authority** valide, sia autogenerati (in questo caso i browser avvertiranno l'utente che il certificato può non essere sicuro).
- Per non dover pagare a tutti i costi un certificato rilasciato da una CA si può utilizzare **Let's Encrypt!** (<https://letsencrypt.org/>)

HTTPS

- Un esempio dell'uso di **OpenSSL** è la combinazione di **http + ssl = https.**
- **https** è un protocollo simile ad **http** che permette connessioni crittografate per garantire la sicurezza dei dati.
- Un server **https** si può realizzare tramite il modulo **ssl** di **Apache** e risponderà sulla porta **443**.
- Anche per **Apache Tomcat**, il web server per le servlet Java, esiste una modalità SSL, che risponde sulla porta **8443**, invece che sulla tradizionale porta **8080**.

HTTPS - Esempio

- Attiviamo il modulo `ssl` e riavviamo apache:
`sudo a2enmod ssl`
`sudo service apache2 restart`
- Creiamo una directory dove mettere il certificato appena generato:
`sudo mkdir /etc/apache2/ssl`
- Creiamo il certificato:
`sudo openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt`
- Configuriamo apache creando un nuovo sito:
`sudo nano /etc/apache2/sites-available/default-ssl.conf`

HTTPS- Esempio

- E inseriamo la seguente configurazione nel file:

/etc/apache2/sites-available/default-ssl.conf:

```
<IfModule mod_ssl.c>

<VirtualHost _default_:443>

    ServerAdmin webmaster@localhost

    DocumentRoot /var/www/html

    ErrorLog ${APACHE_LOG_DIR}/error.log

    CustomLog ${APACHE_LOG_DIR}/access.log combined

    SSLEngine on

    SSLCertificateFile /etc/apache2/ssl/apache.crt

    SSLCertificateKeyFile /etc/apache2/ssl/apache.key

    BrowserMatch "MSIE [2-6]" \
                nokeepalive ssl-unclean-shutdown \
                downgrade-1.0 force-response-1.0

    BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown

</VirtualHost>

</IfModule>
```

HTTPS - Esempio

- Abilitiamo il sito:

```
# sudo a2ensite default-ssl.conf
```

- Riavviamo apache:

```
# sudo service apache2 restart
```

- A questo punto puntiamo il browser verso `https://<ipvostravm>/phpmyadmin` Dovremo vedere il nostro sito protetto con SSL....



Sistemi di monitoring

- Un esempio di utilizzo di un **web server** sono le **applicazioni di monitoring** di un sistema.
- Esistono diversi sistemi di monitoring, classificabili a seconda del livello in cui operano:
 - **Sistemi di basso livello:**
 - Controllo dello stato di salute dei dischi, Controllo dello stato del RAID / Controller, Controllo delle memorie ECC, Sensori di temperatura / umidità Indicatori di velocità delle ventole....
 - **Sistemi a livello SO:**
 - **lm-sensors**, **hddtemp**, **smartd** che controllano i sistemi di basso livello.
 - **watchdog** che controlla errori del sistema operativo.

Sistemi di Monitoring

- Ad un livello più alto (**Applicazione**) troviamo i software di monitoring e di raccolta dati del sistema:
 - **Nagios, ZABBIX**: permettono di controllare un grande numero di host, monitorando hardware e software dei vari host e lo stato dei servizi ospitati.
 - **Mrtg, mailgraph, cacti, ganglia**: raccolgono statistiche e le interpretano sotto forma di grafici per descrivere l'uso del sistema.
- I software sopra citati possono acquisire i dati da una varietà di fonti eterogenee, spesso usando metodi ad hoc.
- Esiste un protocollo per il monitoring di dispositivi, il **Simple Network Management Protocol (SNMP)**.

Nagios - Screenshot

Nagios®

General

[Home](#)
[Documentation](#)

Current Status

[Tactical Overview](#)
[Map \(Legacy\)](#)
[Hosts](#)
[Services](#)
[Host Groups](#)
 [Summary](#)
 [Grid](#)
[Service Groups](#)
 [Summary](#)
 [Grid](#)
[Problems](#)
 [Services \(Unhandled\)](#)
 [Hosts \(Unhandled\)](#)
 [Network Outages](#)

Quick Search:

Current Network Status

Last Updated: Wed Jan 24 12:19:45 CET 2018
Updated every 90 seconds
Nagios® Core™ 4.3.2 - www.nagios.org
Logged in as *nagiosadmin*

[View History For all hosts](#)
[View Notifications For All Hosts](#)
[View Host Status Detail For All Hosts](#)

Host Status Totals

Up	Down	Unreachable	Pending
54	86	0	0
All Problems		All Types	
86		140	

Service Status Totals


Ok	Warning	Unknown	Critical	Pending
115	3	0	92	0
All Problems		All Types		
95		210		

Service Status Details For All Hosts


Limit Results: 100 ▾


Host ↕	Service ↕	Status ↕	Last Check ↕	Duration ↕	Attempt ↕	Status Information
ammsis	PING	OK	01-24-2018 12:18:28	0d 3h 31m 16s	1/3	PING OK - Packet loss = 0%, RTA = 0.89 ms
arf	HTTP	OK	01-24-2018 12:14:28	11d 6h 4m 59s	1/3	HTTP OK: HTTP/1.1 301 Moved Permanently
	PING	OK	01-24-2018 12:14:28	27d 23h 56m 14s	1/3	PING OK - Packet loss = 0%, RTA = 0.46 ms
banale	PING	OK	01-24-2018 12:14:28	27d 23h 59m 9s	1/3	PING OK - Packet loss = 0%, RTA = 0.32 ms
bau	DISK	CRITICAL	01-24-2018 12:16:15	28d 0h 16m 10s	3/3	(Service check timed out after 60.01 seconds)
	HTTP	OK	01-24-2018 12:14:28	27d 23h 59m 9s	1/3	HTTP OK: HTTP/1.1 200 OK - 970 bytes in 0,0
	LOAD	CRITICAL	01-24-2018 12:16:15	28d 0h 16m 6s	3/3	(Service check timed out after 60.01 seconds)
	MEM	CRITICAL	01-24-2018 12:18:26	227d 14h 26m 41s	3/3	(Service check timed out after 60.01 seconds)
	PING	OK	01-24-2018 12:14:28	27d 23h 56m 2s	1/3	PING OK - Packet loss = 0%, RTA = 1.35 ms
	UPTIME	CRITICAL	01-24-2018 12:18:26	227d 14h 14m 8s	3/3	(Service check timed out after 60.01 seconds)


Zabbix - Screenshot


ZABBIX << 


dell



 Monitoraggio ▾

 Inventario ▾

 Report ▾

 Configurazione ▴

Gruppo host

Template

Host

Manutenzione

Azioni

<input type="checkbox"/>	Nome ▲	Applicazioni	Item	Trigger	Grafici	Discovery	Web	Interfaccia	Proxy	Template	Stato	Disponibilità	Agent encryption
<input type="checkbox"/>	arf	Applicazioni 5	Item 27	Trigger 8	Grafici 7	Discovery	Web	157.138.20.111: 10050		Template App HTTP Service , Template App HTTPS Service , Template App SSH Service , Template Module Linux CPU by Zabbix agent , Template Module Linux memory by Zabbix agent	Abilitato	ZBX SNMP JMX IPMI NESSUNO	
<input type="checkbox"/>	broot	Applicazioni 10	Item 124	Trigger 104	Grafici 12	Discovery 3	Web	157.138.22.12: 10050		Template Module ICMP Ping , Template Module Windows CPU by Zabbix agent , Template Module Windows filesystems by Zabbix agent , Template Module Windows memory by Zabbix agent , Template Module Windows physical disks by Zabbix agent , Template Module Windows services by Zabbix agent active	Abilitato	ZBX SNMP JMX IPMI NESSUNO	
<input type="checkbox"/>	budspencer	Applicazioni 4	Item 28	Trigger 9	Grafici 7	Discovery	Web	157.138.20.24: 10050		Template App SSH Service , Template Module ICMP Ping , Template Module Linux CPU by Zabbix agent , Template Module Linux memory by Zabbix agent	Abilitato	ZBX SNMP JMX IPMI NESSUNO	
<input type="checkbox"/>	coccode	Applicazioni 5	Item 27	Trigger 8	Grafici 7	Discovery	Web	157.138.20.11: 10050		Template App HTTP Service , Template App HTTPS Service , Template App SSH Service , Template Module Linux CPU by Zabbix agent , Template Module Linux memory by Zabbix agent	Abilitato	ZBX SNMP JMX IPMI NESSUNO	

Progetto

- Realizzare un web server apache che ospiti almeno 2 virtualhost con https. Realizzare e proteggere una piccola form php. Potete tralasciare i controlli SQL Injection...
- Provare ad utilizzare Let's Encrypt per realizzare un web server con https. Realizzare e proteggere una piccola form php. Potete tralasciare i controlli SQL Injection...
- Effettuare dei benchmark di apache e nginx usando `tsung`: <http://tsung.erlang-projects.org/>
- Usare nagios per monitorare 4 host:
<https://www.nagios.org/>

Avete un Server WEB yeah!

