- Affidabilità:
  - Inaffidabilità: è la probabilità che il componente si guasti nell'intervallo di tempo 0..t sapendo che per t=0 funzionava correttamente (funzione che parte da 0)
  - Affidabilità(RELIABILITY): è la probabilità che il componente non si guasti nell'intervallo 0..t sapendo che per t=0 funzionava (funzione che parte da 1)
  - MTBF = MTTF + MTTR = Mean Time Between Failures.
  - MTTF = Mean Time To Failure.
  - MTTR = Mean Time To Repair.
  - Una stima per il MTTR del software è il tempo necessario al suo riavvio.
  - A = Availability = $\frac{MTTF}{MTTF+MTTR}$ è la % di tempo in cui il sistema funziona bene
  - R(t) = Reliability = $\lim_{n\to\infty}\frac{n(t)}{n_0}$ con n(t) elementi in funzione all'istante t e n(0) = n_0;

    detti X = istante di guasto dell'elemento; F(t) = P(X < t); R(t) = 1 – F(t):

    $\lambda(t)=\lim_{\Delta t\to 0}\frac{P(X\in(t,t+\Delta t)|X>t)}{\Delta t}=\lim_{\Delta t\to 0}\frac{P(X\in(t,t+\Delta t)\cap X>t)}{\Delta t * P(X>t)}=$

    $=\lim_{\Delta t\to 0}\frac{P(X\in(t,t+\Delta t))}{\Delta t*(1-F(t))}=\frac{F(t)}{R(t)}=-\frac{dR(t)}{R(t)dt}=$ failure rate in t.

    supponendo $F(t)=1-e^{-\frac{t}{MTTF}}\simeq\frac{t}{MTTF}$ se $F(t)\ll 1\Rightarrow\lambda=\frac{1}{MTTF}$

    $R(t)=e^{(-\lambda t)}$ $con\,\lambda=cost$ il failure rate è sempre >= di f(t),il F.R non è una densità di probabilità,infatti può essere maggiore di 1

    Se $\mu=cost$ $M(t)=1-e^{(-\mu t)}$
  - $MTTF=\frac{1}{\lambda}$ $MTTR=\frac{1}{\mu}$ $con\,\lambda\,e\,\mu\,costanti$
  - Affidabilità di un sistema con n comp. in parallelo: $R_S(t)=1-\prod_{i=0}^{n}(1-R_i(t))$
  - Affidabilità di un sistema con n comp. in serie: $R_S(t)=\prod_{i=0}^{n}R_i(t)$
  - $MMTF_{SERIE}=\frac{1}{\frac{1}{MTTF1}+...+\frac{1}{MTTFn}}$,

    $MTTF_{PARALLELO}=MTTF(\frac{1}{n}+\frac{1}{n-1}+...+1)$

CLASSIFICAZIONE GUASTI:

- Componenti elettronici: il failure rate è a vasca da bagno. Primo periodo di mortalità infantile dove il failure rate decresce.Secondo periodo,vita utile,il failure rate è basso e costante.Terzo periodo,usura,il F.R cresce,progressivo indebolimento.
- Componenti meccanici:Primo periodo,mortalità infantile in cui il failure rate decresce ma parte da un valore più basso rispetto ai componenti elettronici.Secondo periodo,vita utile,il F.R parte da un valore basso e aumenta quasi linearmente.Terzo periodo,usura,il F.R cresce velocemente.
- Componenti Software: Esiste sempre la mortalità infantile dove il FR decresce.Non esiste il fenomeno di usura;nella vita utile il FR continua a decrescere ma più lentamente;più a lungo è in grado di operare senza errori,più è probabile che in futuro non manifesti problemi.

- LIFE TEXT EXPERIMENT
- SENZA RIMPIAZZO,Failure terminated (si sono rotti tutti):

  $MTTF=\frac{\sum_{i=1}^{n_0}t_i}{n_0}$ n0=numero totale componenti
- SENZA RIMPIAZZO,time terminated(termina dopo un tempo t_d):

  $MTTF=\frac{(\sum_{i=1}^{n_0}t_i+(n_0-M)td)}{m}$ m= componenti guaste fino al tempo t_d
- RIMPIAZZO,failure term:

  $MTTF=\frac{(n_0 t_m)}{m}$ termina dopo che si sono guastate m componenti al tempo t_m
- RIMPIAZZO,time term:

  $MTTF=\frac{(n_0 t_d)}{m}$ t_d è un tempo fissato e in questo intervallo si sono guastate m comp.

WEB APPLICATION

- MOM(message oriented middleware): consentono di coordinare i diversi elementi della logica applicativa in modo affidabile e flessibile. Un messaggio è una richiesta,una risposta,un evento,asincroni,generati da un applicativo. E' un sistema peer-to-peer; per ricevere ed inviare MSG ci si collega ad un agente che fa da intermediario.
  2 tipi: POINT TO POINT, PUBLISH AND SUBSCRIBE:
  PTP: lavorano come un sistema di code,è simile ad una casella mail.
  PAS: concetto di topic (nodo). I client pubblicano i messaggi su questo nodo e si sottoscrivono ad esso per ricevere quelli inviati da altri (newsgroup).
- JAVABEAN: sono una pubblica API,sono classi java realizzate seguendo uno specifico standard: da la possibilità di aggiungere funzionalità senza dover riscrivere il codice.
  STANDARD: le property determinano le caratteristiche comunque modificabili,*gli eventi* vegono usati dai bean per comunicare con altri bean. I bean devono essere *persistenti*,cioè si può salvare e ripristinare lo stato corrente; la persistenza è supportata dalla serializzazione.
  Tutti i *metodi* pubblici vengono esportati.*L'introspezione* è il processo tramite il quale vengono scoperte le caratteristiche del bean: 2 modi,design pattern e metodo che permette la creazione di una classe che implementi BeanInfo.

Reti di code

- **s** = tempo di servizio;
- **Tq** = tempo di coda;
- **λ** = frequenza d'arrivo;
- **R** = tempo di risposta = **Tq** + **s**;
- **N** = numero di utenti nella stazione = **Nq** + **Ns**, con **Ns** = 1 nel caso di un solo servente.
- X= throughput
- U= utilizzo,ovvero il tempo in cui il server è impegnato a processare le richiste.
- Q=lunghezza della coda

Si definiscono tre **grandezze di base**:

- **T** = tempo d'osservazione della stazione [sec];
- **A** = numero di arrivi [job];
- **C** = numero di completamenti [job];

E' chiaro che sarà sempre $A(t)\geq C(t)$ ed entrambe saranno funzioni monotone non decrescenti.
Il numero di utenti del sistema in un dato istante è, intuitivamente: $N(tI)=A(tI)-C(tI)$.

La frequenza di arrivo sarà: $\lambda=\frac{A}{T}$ [job/sec]; il throughput: $X=\frac{C}{T}$ [job/sec].

Si definisce la grandezza **U** = percentuale d'utilizzo = $\frac{B}{T}$ ≤ 1, dove:

$B=\sum_{i=1}^{n}t_i$ è detto busy time, con ti tutti gli intervalli di tempo in cui la stazione è occupata.

Legge Operazionale: $U=\frac{B}{C}\frac{C}{T}=s\,X$ $X=\frac{C}{T}$ $S=\frac{B}{C}$. s= tempo di servizio

Legge di Little:
definiamo $W=\sum t_n n$, con n numero di utenti presenti nell'intervallo di tempo t_n.

Si deduce innanzitutto che $N=$ numero medio di utenti $=\frac{W}{T}$ e che $R=\frac{W}{C}$ da cui: $N=XR$.

Legge del bilanciamento del flusso: $\lambda=X$ deriva dal fato che se osserviamo il sistema per un periodo T sufficientemente lungo,il numero dei richieste arrivate e di quelle completate è uguale.
Modello di un impianto informatico.
Per parlare di un impianto informatico ci resta ancora la componente umana:
si definisce **Z** = think time il tempo medio per pensare di un essere umano.
L'utente è visto come una stazione ad infiniti servitori perchè introduce un ritardo ma non un tempo di coda perchè aspetta la risposta del servente.

Legge di Little per sistemi con utenti interattivi: $R=\frac{N}{X}-Z$

Distinguiamo inoltre tra modello aperto, in cui è noto N, e modello chiuso, in cui N è variabile.
Calcolo tempi di risposta: $R_i=D_i+W_i$,W è il tempo speso ad aspettare che vengano servite tutte le richeste in coda $W_i=D_i+Q_i=D_i X\,R_i=D_i\lambda\,R_i$ $R_i=\frac{D_i}{(1-\lambda D_i)}=\frac{D_i}{(1-V_i)}$

Calcolo lunghezza coda: $Q_i=\lambda R_i=\frac{(\lambda D_i)}{(1-V_i)}=\frac{U_i}{(1-V_i)}$ $Q_i(\lambda)=U_i\frac{(\lambda)}{(1-U_i(\lambda))}$

Legge del flusso forzato.
In modelli complessi, anche un servente è a sua volta utilizzatore di una risorsa.
Di conseguenza un suo job richiederà un numero medio di operazioni.
Definito C_i = numero di operazioni eseguite per compiere un numero C_0 di job, chiamiamo

$V_i=\frac{C_i}{C_0}=$ numero medio di operazioni fatte per job $=\frac{C_i}{T}\frac{T}{C_0}=\frac{X_i}{X_0}$ e

$D_i=V_i S_i=$ richiesta totale di tempo ad una risorsa. U= X D ; D=domanda di servizio,
Q= X V R ; N= X V R

$$X_0=\frac{U_i}{U_j}=\frac{D_i}{D_j}=funz(N)$$

Dischi RAID (storage: high performance high reliability)

- Caratteristiche generali:
  La potenza di calcolo (CPU) e il tasso di velocità della rete crescono molto più rapidamente nel tempo del tasso di trasferimento e del tempo di ricerca dei dischi.
  I tassi di crescita diversi possono dare luogo a due problemi:
  - diminuisce la densità di accessi (se la capacità cresce più rapidamente dei tempo d'accesso);
  - starvation: non completo utilizzo della CPU.

  Si basano sul fatto che le memorie veloci sono più costose e devo dunque essere più piccole.
  I parametri di misura sono:
  - tempo di accesso al dato (fissato dal costruttore);
  - miss rate m(i) (dipende dalla dimensione della memoria e dall'applicazione, dato negli es.);
  - p(i), probabilità di accesso al livello i, equivalente a: p(i) = p(i-1)*m(i-1);
  - tempo medio di accesso al livello i: p(i)*t(i);
  - **tempo medio di accesso al dato**: $\sum_{i=reg}^{dov'è il dato} p(i)*t(i)$ chiaramente calcolato dalla CPU a dove è situato il dato.

  Un disco magnetico è composto da più dischi fisici (piatti) ognuno dei quali è letto da una testina governata da un braccio meccanico.
  I dati sono memorizzati su più tracce per ogni piatto a loro volta suddivise in settori (unità elementare di lettura/scrittura) tipicamente da 512 byte.
  Più la velocità di rotazione dei dischi è alta (tra i 7200 e 15000 rpm) più il tranfert rate e il seek time crescono.
  Il modello di comunicazione tra periferiche e processore è ancora quello degli anni '70 basato sul concetto di bus che al giorno d'oggi si sono però specializzati ed adattati alle molteplici esigenze:
  - tipi: CPU – Memoria; I/O; backplane;
  - comunicazione: sincrona (tra memorie e cpu); asincrona (basato sugli interrupt);
  - traferimento: seriale; parallelo.
- Le prestazioni:
  Si definisce un tempo di servizio come il tempo che si impiega a recuperare il dato richiesto:
  S_{disk} = seek_time + rotation_latency + data_transfert_time + overhead_controller
  - seek_time: posizionamento delle testine (dato);
  - rotation_latency: tempo richiesto affinchè il settore passi sotto le testine (½ giro);
  - data_transfert_time: tempo di trasferimento del dato;
  - overhead_controller: gestione di traferimento dati dal buffer.

  Si definisce un tempo di risposta come il tempo che si impiega a recuperare il dato richiesto considerando il tempo di attesa dato dalla coda nel disco:
  $r = S_{disk} + T_{attesa}$.
- Tecniche per migliorare le prestazioni di I/O:
  Le prestazioni si misurano tramite due parametri: response time e throughput.
  - Read caching: consiste nel tenere in una memoria più veloce i dati LRU.
    Non porta benefici rilevanti se la chache supera il 4% della memoria.
  - Prefetching: consiste nel prevedere e a caricare precedentemente i dati necessari.
    Sono importanti: accuratezza della previsione; costo; tempestività.
    Large Fetch Unit: vengono caricati dei blocchi precedenti e successivi a quello richiesto. Il problema è che si crea un response time penality, cioè il tempo di prefetching può ostacolare degli accessi utili.
    Si possono leggere comunque un certo numero di blocchi successivi (Read Ahead), oppure leggere solo se il disco non ha richieste (Preemptible Read Ahead): la soluzione migliore è una combinazione dei due approcci.
  - Write Buffering: consiste nel mantenere i blocchi scritti in memoria prima di fare un destaging sulla memoria permanente. Per evitare perdita di dati la memoria dev'essere non volatile oppure trasferita su disco periodicamente.
    Il processo di destage inizia quando il numero di blocchi modificati è superiore alla soglia di high mark (0.8) e finisce quando si scende sotto il limite low mark (0.2).
    Il blocco scritto meno di recente viene scaricato (LRW) oppure è selezionata per il destage la traccia con più blocchi modificati.
- I dischi RAID (Redundant Arrays of Indipendent Disks):
  Striping: dati distribuiti in maniera trasparente all'utente su due dischi attraverso un algoritmo ciclico (round robin) per aumentare le performance.Grazie al parallelismo si riduce il tempo di coda e aumenta il trafert rate.E' detta unità di stripe la quantità di dati che vengono scritti su un solo disco.
  Ridondanza:
  cresce la probabilità di guasto => scrittura d'informazioni ridondanti => lentezza delle write.
  Quasi tutti i raid possono essere calssificati in base a:
  - Granularità di interleaving dei dati (unità di stripe):
    - fine grained: piccole unità.
      aumentano il trasfert rate ma si perde tempo in posizionamento.
    - coarse grained: unità relativamente grandi. Richieste piccole possono essere eseguite concorrentemente, quelle grandi usano più dischi.
  - Tipologia di ridondanza: su tutti i dischi o solo su alcuni?
- Tipologie di RAID:
  - Level 0: striping, non c'è ridondanza.Massime prestazioni, costi minimi, scarsa affidabilità.
  - Level 1: mirroring.Alte prestazioni, costi elevati.
  - Level 0+1: prima striping, poi mirroring.
  - Level 1+0: prima mirroring, poi striping.
  - Level 2: bit interleaved parity (ma non usato perchè si basa su un'ipotesi sbagliata). Non si conosce qual'è il dato errato. Dischi ridondanti proporzionali al log dei dischi.
  - Level 3: byte interleaved parity.
    Si conosce dov'è il guasto. Un solo disco dedicato alla parità, più dischi per i dati.
  - Level 4: block interleaved parity.
    Come il livello 3 ma i dati sono gestiti a blocchi, non a byte.
    Si possono usare gli hot spares (dischi sostituibili a caldo).
  - Level 5: block interleaver distributed parity.
    Massime prestazioni/affidabilità con minimi costi.
    I blocchi di parità sono distribuiti (no bottle neck) dunque c'è load balancing.
  - Level 6:
    Estende Level 5 offrendo più tolleranza all'errore.
    Ci sono due sistemi di parità distribuiti (il che peggiora la scrittura).

  Misure dei guasti di un disco:
- RAID 0:
  $MTTF_n = \frac{MTTF_1}{n}$ MTTF=MTTDL; $MTTDL = 1 - e^{(\lambda n t)}$ $MTTF_n = \frac{1}{(n\lambda)}$
- RAID 1:
  P[array RAID 1° guasto]=P[1°guasto] P[2°guasto<MTTR] $MTTDL_{raid1} = \frac{MTTF^2}{(2\,MTTR)}$
- RAID 1+0:
  P[1°guasto]= $n\lambda t = \frac{n}{MTTF} t$ P[2°guasto stesso mirror<MTTR]= $\frac{1}{MTTF} MTTR$
  P[1°guasto] P[2guasto<MTTR]= $\frac{(n\,t\,MTTR)}{MTTF^2}$ -> $MTTDL_{(RAID1+0)} = \frac{MTTF^2}{(n\,MTTR)}$
- RAID 0+1:
  P[1°guasto]= $\frac{n}{MTTF_1} t$ P[2°guasto nel mirror rimanente<MTTR]= $\frac{n}{(2\,MTTF_1)} MTTR$

  $MTTDL_{(RAID0+1)} = 2\frac{MTTF^2}{(n^2\,MTTR)}$
- RAID 5
  M= # dischi G=#gruppi N=# dischi per gruppo =M/G
  P[1gruppo guasto]=P[1°disco guasto] P[2°disco guasto<MTTR]= $N(N-1)\frac{MTTR_t}{MTTF^2}$
  P[array 5+0]=P[1° gruppo guasto] G= $GN(N-1)\frac{MTTR_t}{MTTF^2}$
  $MTTDL_{(RAID5+0)} = \frac{MTTF^2}{(M(N-1)MTTR)}$
- RAID 6
  $MTTDL_{(RAID6+0)} = \frac{MTTF^2}{(M(n-1)(n-2)MTTR^2)}$

Sistemi di Storage

- Direct Attached Storage (DAS):
  Su di un unico host l'applicazione che lavora con la granularità del file si interfaccia col SO (file system, volume manager e poi storage driver) che accede tramite controller con la granularità del blocco alla periferica di storage.
  La scalabilità di quest'architettura è scarsa perchè posso collegare più host attraverso una LAN ma questi host hanno un costo superiore alla singola unità di storage e tra l'altro possono creare problemi in caso di sistemi eterogenei.
- Network Attached Storage (NAS):
  L'host su cui risiede l'applicazione richiede il file al SO che indirizza la richiesta al NAS server (attraverso network file protocol, TCP/IP stack e NIC driver) sulla NIC.
  La richiesta arriva al NAS server che in quanto tale conosce il protocollo di rete e recupera, come se fosse un DAS, il file richiesto.
  Il NAS file access handler si occupa ricevere le richieste dei files e inoltrarle al richiedente.
  La scalabilità di quest'architettura è elevata perchè ogni aggiunta non perturba i sistemi. Può inoltre crescere lungo due dimensioni: aggiungendo periferiche di storage al NAS oppure aggiungendo ulteriori NAS.L'applicazione esegue una richiesta di I/O -> File system-> NFS-> TCP/IP->Driver scheda rete->ethernet->NAS->File system-> volume manager-> storage driver->controller disco
- Storage Area Network (SAN):
  L'idea è quella di avere una rete dedicata all'accesso di periferiche di storage.
  La soluzione SAN attached prevede che ogni host su cui risiede l'applicazione supporti il protocollo SAN (tipicamente Fiber Channel stack), abbia un driver per l'interfaccia e l'interfaccia stessa. Il server SAN è dotato di interfaccia, driver e supporto al protocollo che indirizza direttamente la richiesta alla periferica di storage.
  La differenza col NAS è che in questo caso il trasferimento dati avviene tramite rete dedicata che supporta il trasferimento di singoli blocchi di I/O e non tramite TCP/IP che è un protocollo di rete generico non ottimizzato.Questa soluzione è però molto costosa, motivo per cui sono state introdotte le architetture SAN + NAS head: gli host si interfacciano tramite rete ethernet al NAS server, l'unico ad avere l'interfaccia dedicata per accedere alla SAN e i software necessari per gestirla.Questa soluzione è chiaramente meno performante ma sicuramente più economica.
- Protocollo Fiber Channel:
  Il protocollo TCP/IP su interfaccia ethernet non è adatto allo scambio di grandi moli di dati:
  - gestione del frame ethernet (più è grande, più overhead sulla rete);
  - dimensione del frame ethernet (più piccoli, più overhead sulla CPU);
  - routing dei pacchetti IP (overhead sugli apparati di rete).

  è necessario definire un protocollo dedicato agli Storage Network che unisce i benefici del bus locale (veloce e semplice), con quelli dei protocolli di rete (flessibilità, orientato al pacchetto): full-duplex 1600Mbps, 10km di raggio, componenti standard.
  Architettura: Applicazione; UPPER LAYER {Mapping; Common service};
  PHYSICAL LAYER {Framing protocol; Transmission protocol; Physical}.
  Tipicamente ci si appoggia alla fibra ottica (1Km/5µs) per sfruttare appieno le potenzialità del protocollo.La differenza fondamentale sta nel fatto che frammentazione e trasmissione (routing) sono fatti a livello molto basso (firmware) e sono dunque più veloci.Elementi: viene detto Fabric l'apparato che si occupa di routing, switching e buffering.Vantaggi: prestazioni, affidabilità.Svantaggi: non diffuso come ethernet, costoso.
- Protocolli alternativi a FC: iSCSI: protocollo a livello di sessione che si appoggia su TCP/IP (economico ma scarso); utilizza le stesse ottimizzazioni di FC ed accede a reti SAN.
- FCIP: è lo stesso protocollo FC che si appoggia però su di una rete IP;ha senso di esistere per interfacciarsi a reti ibride oppure per sfruttare l'opportunità di fare backup di una rete FC a distanze maggiori di 10Km.

Sicurezza Informatica

- Definizioni:
  La protezione del patrimonio informativo di un'azienda si basa sul paradigma CIA:
  Confidenzialità: solo chi è autorizzato può avere accesso logico alle informazini;
  Integrità: solo chi è autorizzato può modificare le informazioni;
  Disponibilità: chi è autorizzato può sempre ricevere le informazioni in tempo ragionevole.
  L'implementazione più classica consiste nel definire relazioni tra UTENTI e RISORSE.
  Rischio: esposizione di un'organizzazione a perdite o danni;
  Minaccia: elemento esterno che rappresenta un pericolo per l'impresa;
  Vulnerabilità: un exploit che può essere utilizzato da una minaccia (che diventa aggressore).
  Disastro: attuazione di una minaccia non intenzonale, si previene con:
  - Contingency management plan (chi individua il disastro e attiva le risposte);
  - Business continuity plan (come garantire la continuità del business);
  - Disaster recovery plan (come ripristinare le condizioni normali).

  Attacco: violazione del paradigma CIA (interno/esterno, generico/specifico).
- Gli attacchi:
  Dov'è il nemico: Fuori, Dentro (volontario, involontario), Partner (policy non corrette).
  Tipologie:
  - Attacchi DOS;
  - Penetrazione nel sistema: sottrazione informazioni, utilizzo improprio del sistema;
  - Social Engineering;
  - Attacchi a livello di rete:
    - Sniffing (lettura abusiva dei pacchetti);
    - Spoofing (falsificazione dell'IP del mittente);
    - Hijacking (persone non autorizzate prendono il controllo del canale di com.);
  - Infezione da parte di virus/worm;
  - Online Phishing.
- Difesa:
  A livello di SO è prevista un'identificazione dell'utente e un'assegnamento di privilegi.
  La sicurezza della password può essere aumentata tramite:
  aging (costringere a cambiare la pw) o limitando il numero di tentativi con insuccesso.

CRITTOGRAFIA

- Crittografia: è la scienza che si occupa dello studio delle scritture segrete.
  Garantisce la confidenzialità di un messaggio plaintext rendendolo un ciphertext illeggibile attraverso un algoritmo crittografico pubblico ed una chiave privata (e lunga).
  Ci sono due famiglie di sistemi crittografici:
- Cifrari simmetrici: utilizzano la stessa chiave per cifrare e decifrare il messaggio.
  Il grosso limite è la trasmissione della chiave segreta attraverso una rete insicura.
  Si può risolvere attraverso la trasmissione tramite un canale sicuro (Out Of Band).
  Due algoritmi semplici alla base di quelli moderni: Sostituzione (Cesare) e Trasposizione.
  Il keyspace è la dimensione in bit della chiave segreta.
  Quanto più è lungo, tanto più è difficile forzare l'algoritmo (brute force).
  Vi sono inoltre due approcci crittografici:
  Block cipher: processa il plaintext a blocchi di dimensione fissa con eventuale padding;
  DES: blocchi di 64bit; chiavi di 56bit;
  3DES: usa tre volte il DES con 2 chiavi (cifra con K1, decifra con K2, cifra con K1);
  AES: sostituto del DES; efficiente in sw e hw; standard;
  blocchi da 128, 192, 256 bit; chiavi da 128, 192, 256 bit.
  Stream cipher: processa il plaintext bit per bit.
- Crittografia asimmetrica: si basa sul concetto che ognuno deve avere una coppia di chiavi.
  Il testo cifrato con la chiave pubblica può essere decifrato con quella privata.
  Non c'è il problema di comunicare la chiave.
  Tuttavia sono algoritmi complessi che introducono molto overhead, dunque si preferisce usarli solo per scambiarsi una chiave simmetrica attraverso un canale sicuro.
  - Diffie Hellman: serve per concordare una chiave segreta su un canale insicuro.
    Noti p (numero primo molto alto) ed a (radice primitiva di p);
    decisa la chiave privata X_A di A, la chiave pubblica è: $Y_A = a^{X_A} \bmod p$;
    Allo stesso modo B calcola Y_B e lo trasmette ad A;
    $K_A = (Y_B)^{X_A} \bmod p$, si dimostra essere uguale a K_B !
    A e B sono in possesso di una chiave comune che solo loro possono calcolare.
  - RSA:dati p, q molto grandi e primi è facile calcolare:
    n = p*q e z = (p – 1)*(q – 1) detto toziente di Eulero;
    si sceglie d più piccolo e coprimo (o relativamente primo) di z (esponente privato);
    si trova una e tale che e*d = 1 (mod z) (esponente pubblico)
    Sedobbiamo spedire m ad A, basta spedire $c = m^{e_A} (\bmod n)$, decifrabile solo da A.
    Infatti A farà: $m = c^{d_A} = m (\bmod n)$.
  - Message Digest: impronta digitale di un messaggio che ne garantisce l'autenticità.
    Si ottiene applicando una funzione one way hash tale che:
    - la probabilità che due messaggi generino lo stesso digest sia molto bassa;
    - sia impossibile ricostruire il messaggio a partire dal digest (one way);
    - ogni messaggio abbia uno ed un solo digest (determinismo);
    - la dimensione del digest non dipenda dalla dimensione del messaggio (cost.);
    - sia facile da calcolare.

    MD5: processa il testo in blocchi di 512 bit, lunghezza del digest 128 bit.
    SHA1: algoritmo che migliora MD5, processa blocchi da 512 bit,
    lunghezza del digest di 160bit di default ma variabile (fino a 512 bit).
    Ogni digest di un blocco dipende dal suo contenuto e del digest precedente.
    Il digest del blocco finale è quello dell'intero messaggio.
- Scenari di utilizzo:Criptare un messaggio con la chiave privata assicura autenticazione del mittente ed integrità.Criptarlo con la chiave pubblica del destinatario assicura confidenzialità.La firma digitale è la criptazione del digest del messaggio con chiave pubblica dell'autore.Per garantire le quattro proprietà (autenticazione mittente, integrità messaggio, riservatezza messaggio e non ripudio da parte dell'autore) è necessario combinare questi tre singoli scenari.

Certificati:

- Problematiche e requisiti:
  Differente dal commercio tradizionale:dati importanti viaggiano su Internet, venditore ed acquirente non si conoscono.E' necessario garantire:l'identità e la rintracciabilità del venditore; al venditore la transazione di pagamento; assicurare l'atomicità delle operazioni di pagamento e di evasione dell'ordine.
  Problemi tecnologici: IP, TCP, HTTP non sono autenticati; IP non garantisce riservatezza.
  E' necessario: garantire confidenzialità ed integrità dei dati; autenticare mutuamente le due parti coinvolte. Si vuole rendere trasparente ed interoperabile.
- Architettura di un sistema di pagamento elettronico:
  Il cliente cardholder accede tramite Internet al sito web del merchant e sceglie un prodotto.
  Una certification authority garantisce l'identità del carholder e del merchant.
  Il sito web accede al payment gateway che processa le richieste di pagamento mettendo in comunicazione la banca del cliente, detta issuer, con quella del venditore, detta acquirer.
- Certificati digitali e PKI:
  Problema dell'identità:
  La firma digitale ci dice che un documento è stato generato da una certa chiave privata.
  Non abbiamo la certezza dell'associazione: chiave pubblica <--> persona fisica.
  La Public Key Infrastructure ha lo scopo di garantire quest'associazione.
  Un terzo ente detto Certification Authority (CA) fornisce le informazioni sulla persona associata ad una determinata chiave pubblica.
  La sua comunicazione è a sua volta firmata, per garantire l'autenticità della CA.
  Rilascio di un certificato:L'utente prova la propria identità alla CA (punto debole);
  L'utente genera le sue due chiavi e le consegna alla CA in modo sicuro (punto debole);
  La CA crea un certificato digitale che contiene le informazioni sull'utente, la sua chiave pubblica e la firma digitale della CA.
  Certificato standard X.509:
  Associa un Distinguished Name ad una chiave pubblica e contiene:
  - Numero di versione del certificato;
  - Serial Number del certificato;
  - Chiave pubblica;
  - DN della CA;
  - Periodo di validità;
  - DN del proprietario del certificato;
  - Tipo del certificato;
  - Firma digitale della CA.

  Bisogna stabilire l'identità della CA:
  Autorità di livello superiore: autorità pubbliche, Web Of Trust PGP, standard de facto.
- Secure Socket Layer (SSL):
  Garanzie: confidenzialità & integrità; autenticazione del server e volendo del client.
  Rischi: la comunicazione è sicura ma non è detto che i dati rimangano protetti;
  se fallisce la PKI non si può fare nulla; non è invulnerabile.
  E' necessario che ognuno definisca un cipher setting: insieme degli algoritmi crittografici supportati (simmetrici, autenticazione, integrità, certificazione e scambio di chiavi).
- Fase di Handshake:
  - --> Connection request: contiene cipher setting e dati random per evitare replay;
  - <-- Connection Response: contiene cipher setting dati random concatenati ai precedenti, il certificato del server e volendo l'autenticazione del client;
  - --> Client Authentication: il client invia il suo certificato se richiesto, altrimenti invia la sua chiave pubblica (non certificata);
  - --> Session Key Exchange: il client genera una chiave simmetrica che viene criptata con la chiave pubblica del server.
- Conn. sicura:grazie alla chiave simm. segreta client e server instaurano un canale sicuro.
- HTTPS: è HTTP sopra un canale sicuro costruito con SSL; Utilizza la porta 443.

- Autenticazione:
  Chi si *identifica* dev'essere *autenticato* ed eventualmente *autorizzato* a compiere determinate operazioni.
  Paradigmi di autenticazione:
  - One way: il client si autentica presso un server dato per fidato;
  - Two way: autenticazione reciproca tra client e server;
  - Trusted third party: un ente esterno affidabile autentica sia client che server.

  Tecnologie di autenticazione:
  - Una conoscenza esclusiva (password);
  - Un oggetto esclusivo (smart card, one time password, certificato digitale);
  - Una caratteristica biometrica.

  I problemi della password riguardano:
  - Furto della password;
  - Uso improprio della password (condivisione, cattiva segretezza);
  - Può essere indovinata se il keyspace è breve (brute force, dictionary based).

Sicurezza Architetturale:

- Firewall: sistema di controllo degli accessi che verifica tutto il traffico che transita attraverso di lui, consentendone o negandone il passaggio basandosi su una security policy.
  Funzioni: IP Filtering, NAT (mascheramento indirizzi interni), Blocco dei pacchetti pericolosi o non autorizzati.
  Limiti: si applica ad un collegamento ad Internet dunque non può bloccare intrusioni interne, né gli attacchi alla rete da parte di collegamenti che non sono di sua competenza, può essere forzato. E' l'organo vitale per la sicurezza.
  Policy di Sicurezza: da loro dipende il buon funzionamento del Firewall; tipicamente è un tradeoff tra sicurezza e bisogni dei dipendenti (politica aziendale).
  Tipologie di Firewall:
  - Network Layer:
    - Packet Filtering: filtra i pacchetti in base all'header (ip:porta di sorgente e destinazione, protocollo e sue opzioni); E' basato su ACL e nient'altro;
    - Stateful Packet Filtering: riproduce la macchina a stati del TCP offrendo una maggiore espressività, ispezione avanzata, NAT; può però comportare problemi di memoria e performance.

    Sono più veloci ed efficienti delle altre tipologie e sono trasparenti.
  - Application Proxy Firewall: controlla la validità dei dati a livello applicativo; offre caching, sicurezza a livello applicativo (autenticazione o antivirus...), logging, politiche personalizzate, filtraggio sui contenuti;
    gli svantaggi sono la scalabilità problematica (nuovi servizi richiedono nuovi proxy) e la progettazione non banale.
- Architettura a due zone: serve per far accedere utenti remoti a risorse pubbliche dell'azienda; si crea una zona demilitarizzata (DMZ) attraverso due firewall oppure un firewall a tre vie; gli utenti remoti possono accedere a risorse della DMZ il che la rende una zona pericolosa;
- Virtual Private Network (VPN): è un canale di comunicazione crittografato su rete pubblica; serve per far accedere utenti aziendali alla rete interna da remoto mantenendo confidenzialità ed integrità (risparmiando su linee dedicate);
  ci sono due politiche possibili:
  - tutto il traffico nel tunnel: moltiplica il traffico, impatta sulla CPU del firewall e del VPN server;
  - split tunneling:il traffico per la rete va nella VPN, quello di Internet va sulla rete insicura; questo è un rischio perchè collega la rete aziendale ad una rete non sicura.

Architetture Enterprise:

- Cluster: sistema di computer che cooperano in modo tale da essere visti come un singolo.
  Si possono fare due utilizzi:
  - High Availability cluster: se un server si guasta viene sostituito automaticamente da un altro attivato per l'occasione (attivo / passivo) oppure già attivato (attivo / attivo);
  - Load Balancing Cluster: più server concorrono per allegerire il carico di ognuno (occorre load balancing).

  Ogni nodo del cluster deve avere due interfacce: una di rete pubblica, una privata che serve per comunicare il proprio stato agli altri nodi (heartbeat,è una versione sofisticata del ping e viene utilizzata dal nodo passivo per verificare se il nodo attivo sta operando correttamente) e per compiti di sincronizzazione.
  Se i nodi sono due si può usare interfaccia seriale o cavo cross link; altrimenti si usano hub o switch dedicati o persino interfacce dedicate a questi compiti (backplane InfiniBand).
- Load Balancer & SSL Accelerator:
  Si preferisce il load balancer che garantisce prestazioni elevate e maggiore robustezza.
  La scalabilità è garantita dalla possibilità di escludere momentaneamente dal load balancer dei server a scopo di manutenzione o di aggiungerne di nuovi.
  I client devono collegarsi ad un Virtual IP che corrisponde a quello del load balancer il quale reindirizza, secondo algoritmi appositi, la richiesta al server meno carico.
  Questo sistema implica che il Load Balancer debba agire anche da NAT (+ sicurezza).
  Inoltre il Load Balancer dev'essere stateful, ovvero indirizzare le richieste di una stessa sessione allo stesso server; per fare ciò dispone di una tabella di sessione.
  Per questioni di affidabilità il Load Balancer può essere a sua volta affiancato da un'altro LB che lavora insieme a lui sulla stessa tabella di sessione (synchronized).
  Il Load Balancer si occupa anche di health checking sia della macchina (attraverso ping), sia dell'applicativo (attraverso http ad esempio).
  Il Load Balancer può inoltre essere fornito (o anche disponibile come apparecchio esterno) di SSL Accelerator che svolge in HW e dunque molto velocemente le fasi di creazione di un canale sicuro SSL (criptazione e decriptazione) in modo che i server risultino sgravati da questo carico e vi sia comunque una connessione sicura sopra Internet.
- Componenti:
  - Web Server;
  - Application Server: fornisce logica applicativa, middleware, pooling delle risorse;
  - Database;
  - Gestore dei domini;
  - Mainframe e sistemi legacy.

Grid computing e architetture parallele:

- Grid computing: paradigma/infrastruttura che abilita la condivisione, la selezione e l'aggregazione di risorse distribuite.
- Risorsa distribuita: è un concetto molto generale che può indicare risorse HW, software, dati, periferiche, persone (virtua organization) che cooperano allo stesso problema pur essendo distribuite geograficamente oppure eterogenee.
- Obiettivo: risoluzione di problemi di larga scala attualmente non fattibile con le macchine a disposizione.
  Availability, reliability;Prestazioni;Modularità e facilità di espansione;Condivisione di risorse;Load balancing;Buone prestazioni anche in caso di overload.
- Peculiarità:
  Molteplicità di risorse eterogenee;Interconnessioni;Unità di controllo (coordinamento);
  Trasparenza di programmazione e di utenza;Autonomia di componenti
- Problematiche:
  Eterogeneità delle risorse;Indirizzamenti / Nomenclatura;Ownership: chi controlla la risorsa e problemi di sicurezza;Accesso concorrente;Coerenza della cache;
  Parallelizzazione del problema (scomposizione del dominio).
- Applicazioni principali:
  Simulazioni fisiche (previsioni del tempo, dinamica molecolare);Filtraggio filmati.

Parallel Computing:

- Speed-Up: incremento di prestazioni all'aumentare del numero di CPU;
  fissata la macchina e l'applicazione e definito P = #CPU e T = tempo di esecuzione:

$$S_P = \frac{T_1}{T_P}$$

- Efficienza: capacità di sfruttare la potenza di calcolo: $E_P = \frac{S_P}{P}$;

  si vorrebbe un'efficienza lineare (= 1) o anche più che lineare ma è difficile da ottenere.
- Legge di Amdahl: spiega la difficoltà di ottenere Speed-Up lineari;
  $T_1 = T_S + T_P$, ovvero il tempo d'esecuzione di un programma è scomponibile in una componente seriale ed una parallelizzabile;

  La % di codice sequenziale si esprime con $f_S = \frac{T_S}{T_S + T_P}$;

  Il tempo di esecuzione con un numero N di processori è: $T_N = T_S + \frac{T_P}{T_N}$;

  Dunque si ricava che: $S_N = \frac{N}{N f_S + (1 - f_S)}$.

  Ciò significa che solo se $f_S = 0$ lo Speed-Up può essere lineare, altrimenti si giunge ad un livello di saturazione.
  In realtà non è neanche così perchè aumentare il numero di CPU aumenta inevitabilmente i tempi di comunicazione e coordinamento.
  Dunque all'aumentare di N non c'è saturazione ma persino una decrescita di Speed-Up dopo un livello di massimo.
  L'approccio multiprocessore parrebbe non aver senso...
  ma se consideriamo $T(k, N) = T_S + \frac{T_P}{N} + k log_2 N$
  dove l'ultima è la parte che indica la complessità del problema si scopre che k dipende da N;
  che ci porta a scrivere la formula dello Speed-Up scalato: $S_N = \frac{T(1, k(n))}{T(n, k(n))}$
  che prova che l'approccio multiprocessore può essere utile.

Modelli Hardware:

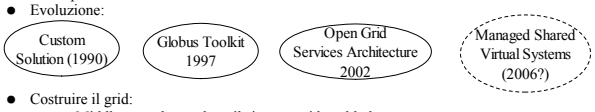
- Macchine a memoria condivisa: Symmetric MultiProcessor
  perchè non hanno percorsi preferenziali di accesso alla memoria.
  - Uniform Memory Access;
  - Non Uniform Memory Access: sfruttano la località spaziale dei dati grazie ad un raggruppamento a "grappoli" delle CPU e di una chache di terzo livello.

  Il collo di bottiglia è il BUS della memoria condiviso.
  E' inoltre importante gestire il problema della coerenza della cache.
  L'algoritmo più usato per affrontare questo problema è lo snooping algorithm:
  - letture multiple dello stesso dato non creano problemi;
  - in caso di scrittura introducono uno snoop tag che monitora il BUS condiviso:
    - Write Invalidate: prima di aggiornare il dato si manda un segnale di "invalidate" e NON si aggiorna la memoria;
    - Write Update: non si invalida la cache ma si manda il dato aggiornato (pesantissimo).
- Macchine a memoria distribuita: ogni CPU ha la sua allocazione di memoria
  l'utente / programmatore deve gestire in maniera esplicita la distribuzione dei dati e la loro allocazione
- Tipologie di elaborazione parallela:
  - [SISD]: Single Instruction, Single Data (seriale);
  - SIMD: Single Instruction, Multiple Data (calcolatori vettoriali);
  - MIMD: Multiple Instruction, Multiple Data (ogni CPU esegue istruzioni differenti);
    - SPMD: Single Program, Multiple Data (stesso programma su dati differenti).
- Programmazione parallela:
  Occorre sincronizzazione, è più ottimale parallelizzare tutti gli step, spesso cambia l'algoritmo al fine di sfruttare al meglio tutti i processori.
- Paradigmi di programmazione a memoria condivisa:
  - HPF: deriva dal Fortran90 ed aggiunge lo statement forall più alcune direttive di decomposizione dei dati (approccio SPMD); parallelizzazione in compilazione;
  - OpenMP: librerie per il C che parallelizzano alcune istruzioni (cicli);
  - Skeleton: insieme di strutture / modelli tipici delle applicazioni parallele:
    - pipeline: scompone una funzione in sottofunzioni eseguibili in parallelo;
    - farm: frazionamento del problema in sottoproblemi gestiti dal farmer ed eseguiti dai worker;
    - loop: parallelizzazione dei cicli;
    - sequenziale ed altri...

  ha il problema delle basse performance.
- Paradigmi di programmazione a memoria distribuita:
  - Parallel Virtual Machine: fornisce supporto per un OS distribuito;
    l'utente genera e gestisce i processi;
  - Message Passing Interface: è un insieme di specifiche API (diverse implementazioni) per gestire Cluster anche eterogenei;codice parallelo riusabile e portabile;
    sper sviluppatori di librerie o programmi;indipendente dalla piattaforma;
    rende possibile creare sottogruppi di CPU (communicators);offre comunicazioni point to point e collettive;

Infrastrutture Grid Computing:

- Cluster: insieme di PC (o Workstation) interconnessi da una rete e collaboranti tra loro.
  La banda sempre maggiore ha permesso di renderli competitivi (soprattuto per i costi) coi supercomputer: la banda cresce di più della potenza di calcolo.
- Pros: performance, scalabilità, throughput, availability, low cost.
- Cons: la rete è il collo di bottiglia, le performance sono inferiori rispetto ai supercomputer.
- Classificazione di Cluster:
  - Applicazione: High Performance; High Availability.
  - Ownership: Dedicated; Non Dedicated.
  - Hardware: PC, Workstation, SMP.OS: Linux, Solaris, ..., Microsoft.
  - Node Configuration: Homogeneous; Heterogeneous.
  - Levels Of Clustering: Group, Departmental, Organizational, National, International.
- I parametri del grid computing possono essere l'HPC (High Performance Computing) o potenza istantanea e l'HTC (High Throughtput Computing) o throughput.
  L'idea è quella di creare delle comunità di condivisione di risorse simili ad una rete elettrica dove ognuno utilizza, quando gli serve, la potenza di calcolo di cui ha bisogno.
- Nascono così le Virtual Organization: aggregati di risorse / persone che collaborano allo stesso problema.
- Evoluzione:



- Costruire il grid:
  - Middleware che rendano il sistema grid enabled;
  - Security Mechanism:
    - capire chi è resource owner / user;
    - flessibilità per gestire molte risorse;
  - Programming tools: astazioni e modelli per aggiungere velocità / robustezza allo sviluppo (ricerca di bug e condivisione di codice comuni).
  - Tools: per tradurre i requisiti di un'applicazione nei requisiti di risorse; per monitorare, scoprire, concordare, comporre job.
- Layer:
  Application; Collective: coordina multiple risorse; Resource: controlla singole risorse;
  Connectivity: comunicazione ed autenticazione;Fabric: accesso locale alle risorse.
- La sicurezza (in Globus Toolkit):
  Autenticazione sicura:
  - X.509: tramite mutual authentication;
  - Single sign-on e delegation: un sito delega degli utenti, gli utenti delegano i servizi;
  - User Proxy: anche lui ha un certificato X.509.
- Sulla sicurezza si appoggiano tre "servizi":
  - Execution Management: GRAM controlla l'esecuzione dei job;
  - Information Services: accesso uniforme allo stato delle risorse;
  - Data Management: trovare tutta l'informazione rilevante.
- MDS è un insieme di servizi per il monitoring e il discovery delle risorse e dei servizi della griglia,è utile per avere un accesso uniforme a quali risorse sono disponibili,qual'è il loro livello di utilizzo così da controllare il sistema.
- GridFtp è un protocollo basato su FTP,l'estensione riguarda l'integrazione di protocolli sicuri,usando GSI(Grid security infrastructure) per garantire autenticazione,integrità e confidenza.
- RLS (replica location service) è un registro pubblico che memorizza dove si trovano le copie dei dati e ne permette il discovery,la replica è fondamentale per garantire fault tollerance e assicurare disponibilità al sistema,migliorare le prestazioni,evitare latenze e consentire il bilanciamento del carico.