

Collezione di slides presentate al corso di informatica Industriale

A. Fantechi

1. Duplicazione Configurabile
 - Esempi
 - Valutazione disponibilità
2. Dischi RAID
3. Checkpointing in ambito distribuito
4. Memoria stabile
5. Two phase commit
6. Paradosso dei generali bizantini

• *Duplicazione configurabile*

- Una macchina *master* e una *slave (primary/secondary)*
- La macchina slave subentra a quella master in caso di guasto del master (*failover*)
- La rilevazione dell'errore avviene mediante codifica dell'errore o time-out: il fallimento è un crash della macchina, sempre rilevabile (*assunzione sui guasti*)
- con *riserva fredda (cold spare)*: lo slave viene tenuto normalmente spento
 - Problema della copia dei dati sensibili
- con *riserva calda (hot spare)*: lo slave funziona in parallelo al master, ricevendo gli stessi input; i suoi output non vengono forniti all'esterno.
 - la riserva calda permette di non attendere il tempo di start-up della riserva, e mantiene sempre una copia aggiornata dei dati sensibili.
 - richiedendo che anche la riserva sia sempre in funzione, diminuisce l'affidabilità complessiva

A configurable duplication solution from the open source world

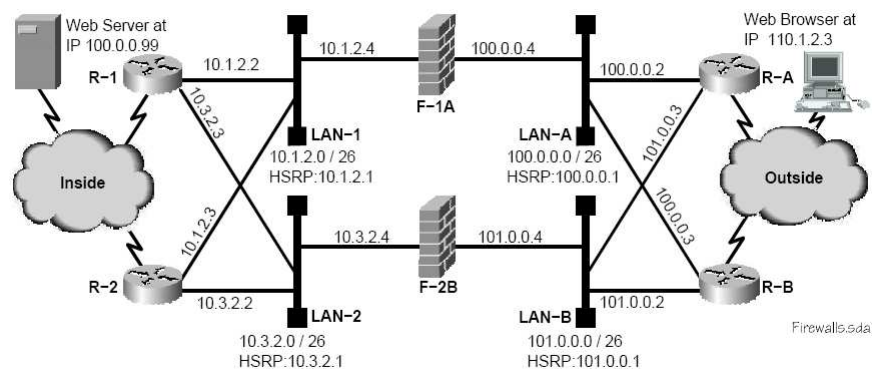
- Build a redundant installation of OpenBSD firewalls against hardware failures. (OpenBSD is stable enough: no need to worry about software crashing, but concern about hard disks and other possibly failing hardware)

- 1) Install a working firewall to be the "primary" firewall that operates continually under normal conditions.
 - 2) Install a "secondary" firewall with the same configuration, but have the interfaces set up with different addresses.
 - 3) Create a process on the "secondary" firewall to monitor the "primary" firewall.
 - 4) Install a software-controlled power switch on the primary firewall power line, and have it controlled by the secondary firewall.
- 5) Create a script on the secondary firewall to reassign the interfaces with the same addresses as the primary firewall, and to start the packet filter with the same rules as the primary firewall.

So, when the primary firewall fails, the secondary will kill the power to the secondary firewall, and reconfigure itself to run as the firewall for the network. Since the primary firewall is powered off, it can't interfere with the secondary firewall.

- → filtering function of firewall is assumed *stateless (static filtering)*: no need of recovering the state of the killed firewall

An industrial solution (Networking Unlimited)



Quattro router e due firewall per coprire tutti i possibili guasti singoli
Riconfigurazione basata sull'aggiornamento automatico delle tabelle di routing a livello del protocollo IP.

Esempio di Valutazione disponibilità

- Consideriamo un sistema in cui un computer funzionante è affiancato da un altro identico che agisce come “riserva fredda”, subentrando nel funzionamento in caso di guasto del primo; l’interruzione del servizio dovuta allo start-up del computer freddo risulta essere di mezz’ora. (MTTR)
- Il tasso di fallimento di ognuno dei due computer è di $1 \cdot 10^{-5}$ ore (MTTF = circa 11 anni), e si suppone che il computer guasto venga rimpiazzato in tempo utile ad evitare un guasto doppio
- Come si valuta la disponibilità del sistema?

Esempio di Valutazione disponibilità

$$\begin{aligned} \text{Disp} &= \text{MTTF}/\text{MTBF} = \\ &= \text{MTTF}/(\text{MTTR}+\text{MTTF}) = \\ &= 100000/100000,5 = \\ &= 0,99999500002499.... \end{aligned}$$

Non è “quasi uguale” a 1, ma l’indisponibilità è pari a 0,5 su 100000, cioè mezz’ora di mancato servizio ogni 11 anni (ovvio....), o circa 3 minuti di mancato servizio in un anno

→ (accettabile???) *vedi SLA...*

La disponibilità spesso si esprime in termini di “nines”,

cioè il numero di nove dietro lo zero

In questo caso si parla di una disponibilità di 5 nines

Guasti dei supporti di memorizzazione: Strutture RAID

- Uso di dischi in parallelo
 - Maggiori Prestazioni
 - Minore Affidabilità (l'uso di dischi in parallelo aumenta la probabilità di guasto)
- Affidabilità
 - Si migliora introducendo ridondanza
- Copiatura speculare (mirroring)

Dischi RAID

Redundant Arrays of Independent (Inexpensive) Disks

- molti dischi indipendenti visti come un unico grande disco logico ad elevate prestazioni
- i dati sono distribuiti (striped) su più dischi che sono acceduti in parallelo ottenendo:
 - **elevato transfer rate** per accessi a grandi quantità di dati (operazioni di I/O pesanti)
 - **elevato I/O rate** per accessi a piccole quantità di dati (operazioni di I/O leggere)
 - **load balancing** tra i vari dischi in modo automatico

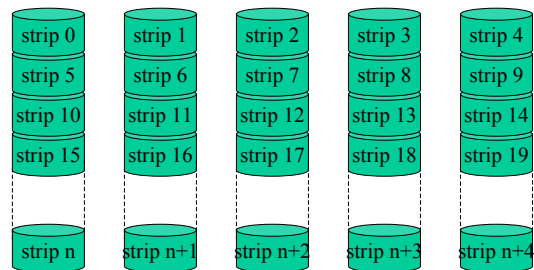
RAID: aumentata vulnerabilità

- array di 100 dischi: probabilità di guastarsi superiore di 100 volte quella di un disco solo
- se un disco ha un MTTF (Mean Time To Failure) di 200000 ore (~23 anni) un array di 100 dischi avrà un MTTF di 2000 ore (~ 3 mesi)
- **ridondanza** dei dati scritti: possibilità di correzione di eventuali errori/perdite di dati (disco guasto) con tecniche di codici a correzione di errore che utilizzano dati ridondanti scritti su dischi diversi da quelli sui quali vengono scritti i dati

data striping

- i dati sono distribuiti, in modo trasparente all'utente, su più dischi 'visti' come un unico disco veloce di grande capacità
- **striping**: suddivisione di dati che devono essere scritti sequenzialmente (un vettore, un file, ...) in segmenti (unità di stripe) che vengono scritti su più dischi fisici con un algoritmo round robin
- unità di stripe: quantità di dati che vengono scritti su un solo disco (~2÷128KB)
- **stripe width**: numero di dischi usati dall'algoritmo di striping (numero di dischi nell'array)

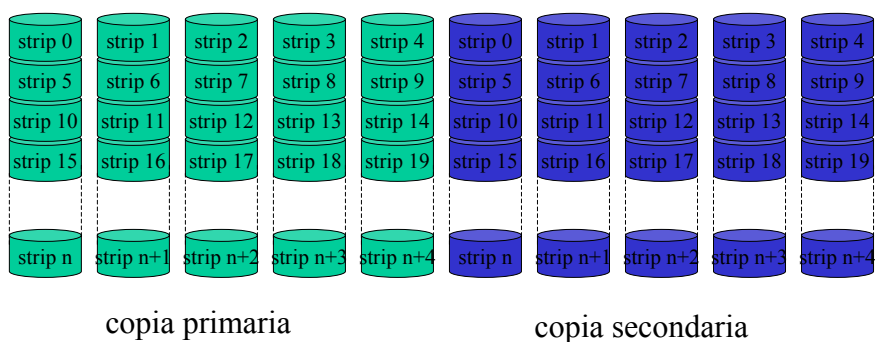
RAID-0



Le strip vengono scritte in parallelo:
maggiore velocità di accesso

Non c'è ridondanza

RAID - 1

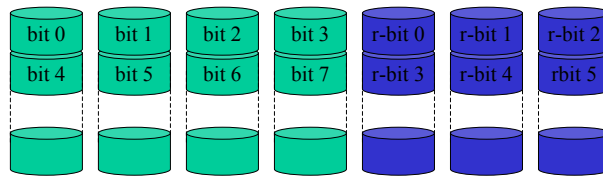


Ridondanza (duplicazione) - mirroring

A seguito di un guasto, si può ripristinare
(una volta riparato/sostituito) il contenuto del disco guasto,
prelevandolo dalla copia

(si utilizza solo il 50% della capacità fisica)

RAID-2



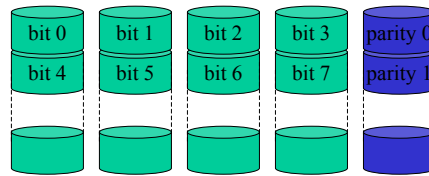
codifica dei dati usando un codice Hamming per ogni strip di dati. Questo codice può rivelare e correggere errori e permette il recupero dei dati senza una totale duplicazione.

(Codice Hamming)

| Dimensione di parola | Bit di controllo | Dimensione totale | Percentuale assorbimento |
|----------------------|------------------|-------------------|--------------------------|
| 8 | 4 | 12 | 50% |
| 16 | 5 | 21 | 31% |
| 32 | 6 | 38 | 19% |
| 64 | 7 | 71 | 11% |
| 128 | 8 | 136 | 6% |
| 256 | 9 | 265 | 4% |
| 512 | 10 | 522 | 2% |

Numero di bit di controllo necessari per un codice di correzione per un solo errore (SEC: Single Error Correction)

RAID-3



La stessa word/byte e` distribuita su diversi dischi.

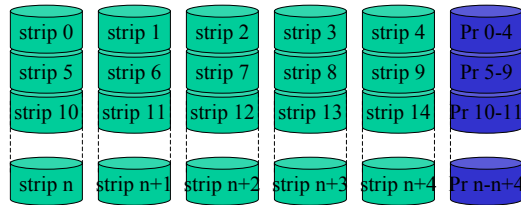
Il bit di parità consente di recuperare il contenuto di un disco guasto, e quindi ripristinare l'array una volta riparato il guasto.

Recupero del contenuto tramite parità

(somme modulo 2, cioè XOR)

- $\text{parity} = \sum_i \text{bit}_i$
- in caso di scrittura in cui il bit_i diventa bit'_i , basta eseguire la scrittura:
 $\text{parity} := \text{parity} + (\text{bit}_i - \text{bit}'_i)$, cioè:
 $\text{parity} := \text{parity} + \text{bit}_i + \text{bit}'_i$
- in caso di guasto al disco j , si recupera il valore del bit j -esimo eseguendo:
 $\text{bit}_j := \text{parity} - \sum_{i \neq j} \text{bit}_i$, cioè:
 $\text{bit}_j := \text{parity} + \sum_{i \neq j} \text{bit}_i$

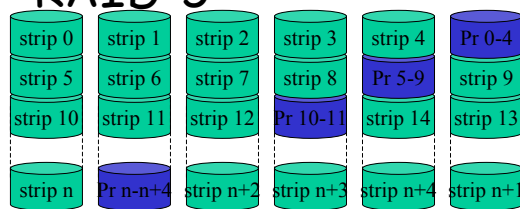
RAID-4



I dati sono distribuiti in blocchi, non in bit

Poiche' la parita' e' sullo stesso disco, tutte le transazioni richiedono un accesso ad esso, che quindi diventa un collo di bottiglia. La soluzione proposta dal RAID-5 consiste nel distribuire la parita' sui dischi

RAID-5



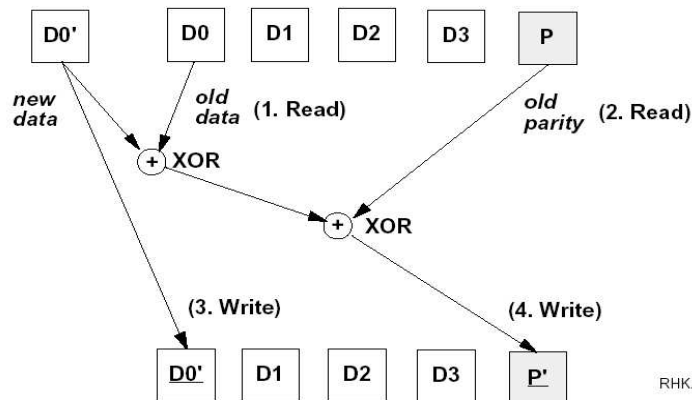
Soluzione più diffusa: maggiori vantaggi in termini di affidabilità e prestazioni, con minori costi per la bassa ridondanza

(Esiste anche un RAID-6 che e' un potenziamento del RAID-5: maggiore ridondanza per uso di codifica)

Problems of Disk Arrays: Small Writes

RAID-5: Small Write Algorithm

1 Logical Write = 2 Physical Reads + 2 Physical Writes



RHK.S96 15

Forward e Backward Error Recovery

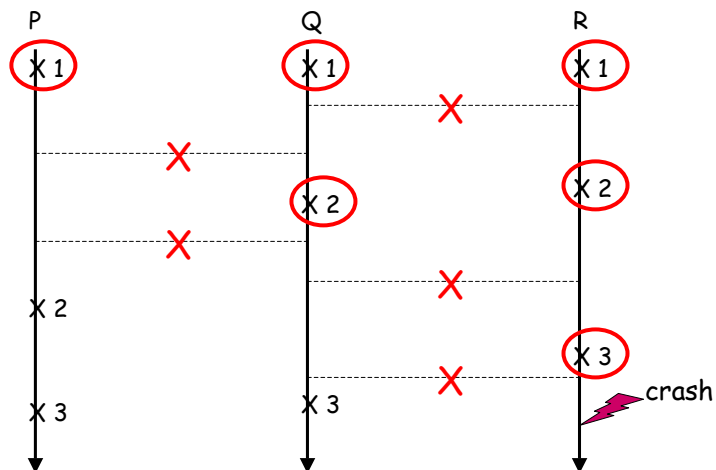
- Il reset è un caso particolare delle procedure di **recovery** (recupero) a fronte di un guasto di una macchina singola.
- In caso di guasto (**crash**) la macchina viene fatta ripartire, ma nel caso di computazioni "stateful" dovremo essere in grado di ripristinarne lo stato.
- **Forward error recovery**: il sistema viene riportato ad uno stato noto come "safe" (il reset è un caso particolare: stato iniziale)
- **Backward error recovery**: il sistema viene riportato ad uno stato "corretto" in cui è già transitato.
→ occorre perciò memorizzare lo stato di tanto in tanto su memoria non volatile (**Checkpoint**) (→ **memoria stabile**)

La granularità del checkpointing dipende dai valori di disponibilità attesi (quanta computazione possiamo permetterci di perdere)
La tecnica del checkpointing può essere utilizzata anche per inizializzare lo stato di una riserva fredda

Azioni atomiche e transazioni

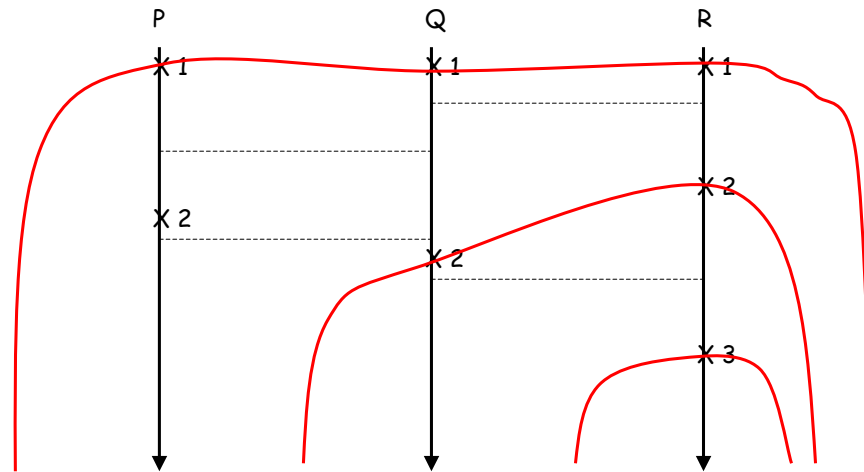
- Il lavoro svolto tra un checkpoint e il successivo, viene considerato, per quanto riguarda i guasti, un'azione atomica, che gode della proprietà *tutto o niente*
- Concetto simile a quello di transazione su una base dati.
- *ACID transaction*:
 - Atomic
 - Consistent
 - Isolated
 - Durable

Checkpointing in ambito distribuito



EFFETTO DOMINO

Azioni atomiche distribuite



= lavoro svolto dai nodi tra due *linee di recovery*

Meccanismo per garantire azioni atomiche: Two-phase commit protocol

Memoria stabile: un meccanismo di base per protezione dei dati dai crash

- Astrazione **memoria reale:**

```
type dati = .....;
  pagina = 0 .. Maxpag;
  risultato = (OK, BAD);
Procedure Read (P: pagina; var D: dati; var R: Risultato);
Procedure Write (P: pagina; D: dati);
```

Non viene influenzata da crashes del processore, eccettuata la pagina su cui, eventualmente, si scriveva al momento del crash (procedura di write non atomica)

Memoria stabile

- Ogni pagina stabile P_S è realizzata tramite due (o più) pagine reali P_1 e P_2 tali che non decadano mai contemporaneamente (appartengono a cilindri diversi dello stesso disco, o a dischi diversi)
- Le procedure read-stabile, write-stabile, restart-da-crash, garantiscono, per ogni pagina stabile P_S , che valga il seguente invariante:

$(OK(P_1) \mid OK(P_2)) \ \&$
 $(OK(P_1) \ \& \ OK(P_2)) \Rightarrow \text{content}(P_1) = \text{content}(P_2)$

Validity

Agreement

(l'invariante vale sempre, tranne che durante l'esecuzione di una write-stabile)

Memoria stabile - operazioni stabili






```
Procedure Read-stabile (Ps: pagina-stabile; var Ds: dati);
begin
  Read(P1,d,Ris)*;
  if Ris = OK then Ds:=d;
  else begin
    Read(P2,d,Ris)*; Ds :=d;
  end;
end;

Procedure Write-stabile (Ps: pagina-stabile; var Ds: dati);
Begin
  repeat
    Write(P1, Ds); Read(P1,d,Ris);
  until Ris = OK and Ds = d;
  repeat
    Write(P2, Ds); Read(P2,d,Ris);
  until Ris = OK and Ds = d;
end;
```

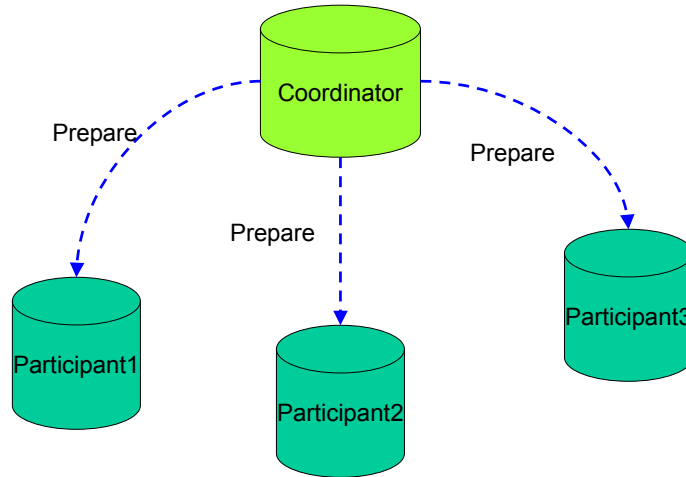
Memoria stabile - operazioni stabili

```
Procedure restart-da-crash;  
Begin  
  Read(P1,d1,Ris1);  
  Read(P2,d2,Ris2);  
  if Ris1 = BAD then  
    repeat  
      Write(P1, d2); Read(P1,d1,Ris1);  
    until Ris1 = OK and d1 = d2;  
  else if Ris2 = BAD or d1 != d2 then  
    repeat  
      Write(P2, d1); Read(P2,d2,Ris2);  
    until Ris2 = OK and d1 = d2;  
end;
```

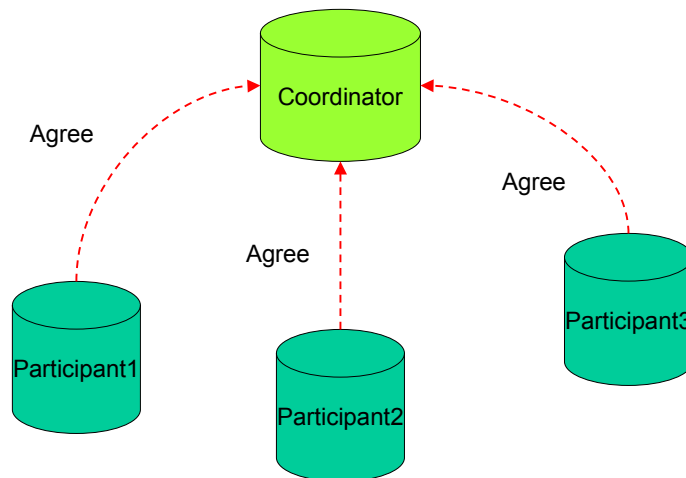
Validity and Agreement in case of crash

```
Procedure Write-stabile (Ps: pagina-stabile; var Ds: dati);  
Begin  crash 1  
  repeat  
    Write(P1, Ds);  crash 2  
    Read(P1,d,Ris);  
  until  crash 3 Ris = OK and Ds = d;  
  repeat  
    Write(P2, Ds);  crash 4  
    Read(P1,d,Ris);  
  until  crash 5 Ris = OK and Ds = d;  
end;
```

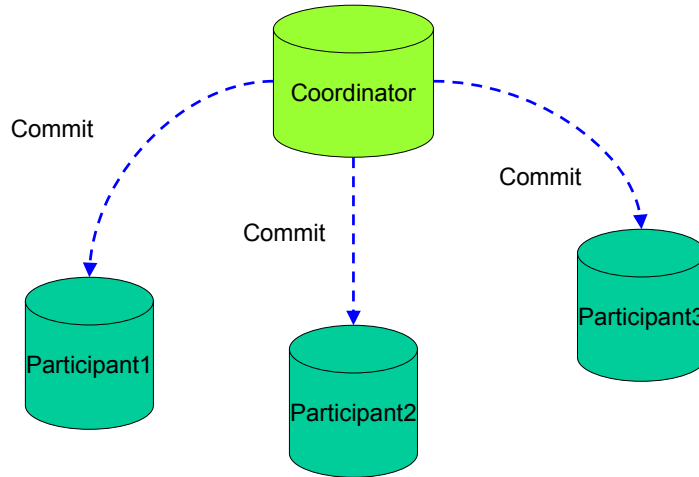
Two-Phase Commit Protocol (Commit) - 1 (fase 1)



Two-Phase Commit Protocol (Commit) - 2 (fase 1)

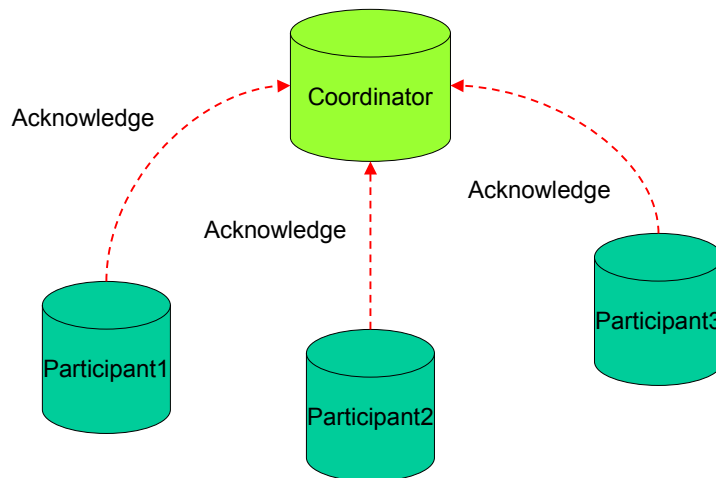


Two-Phase Commit Protocol (Commit) - 3 (fase 2)

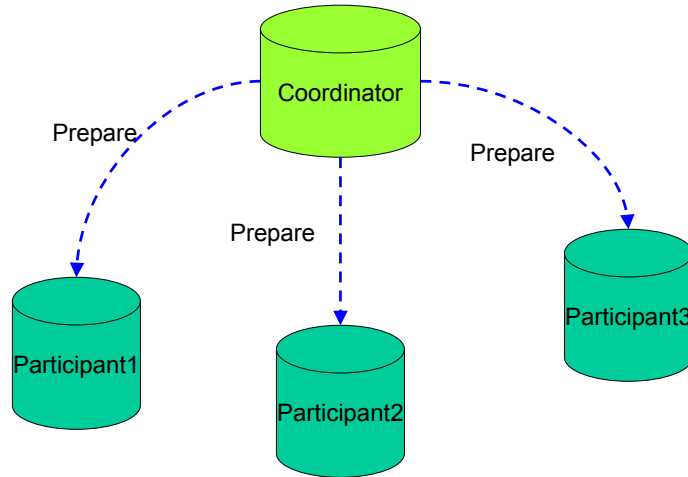


Two-Phase Commit Protocol (Commit) - 3 (fase 2)

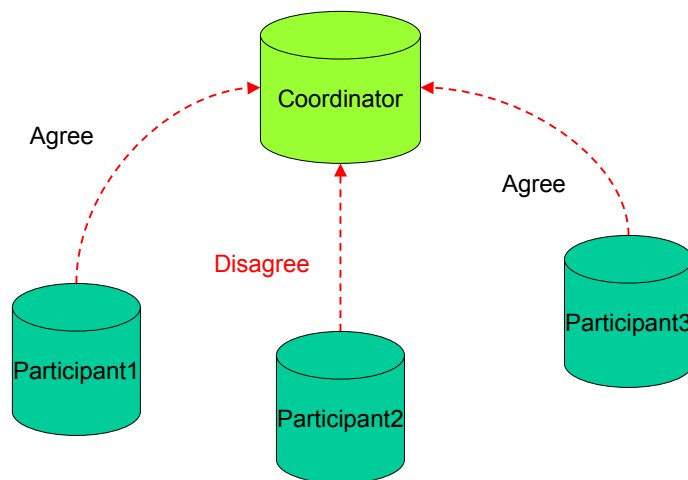
→ *(consenso a terminare con successo l'azione atomica)*



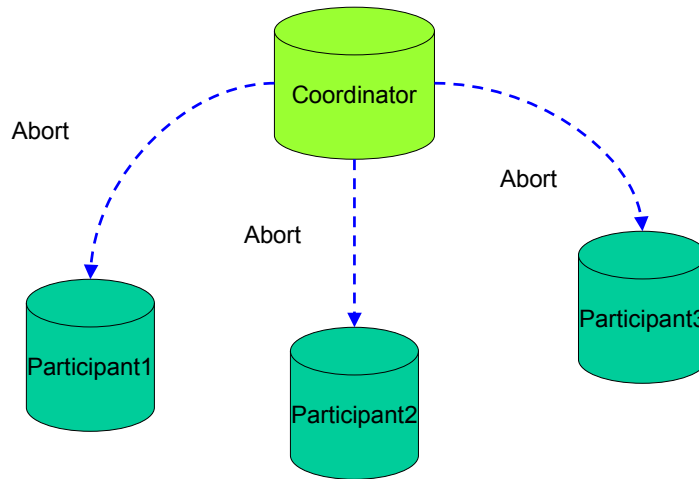
Two-Phase Commit Protocol (Abort) - 1 (fase 1)



Two-Phase Commit Protocol (Abort) - 2 (fase 1)

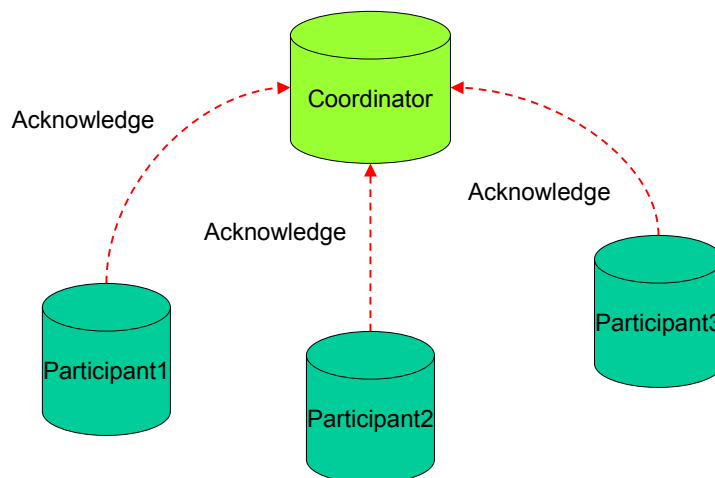


Two-Phase Commit Protocol (Abort) - 3 (fase 2)

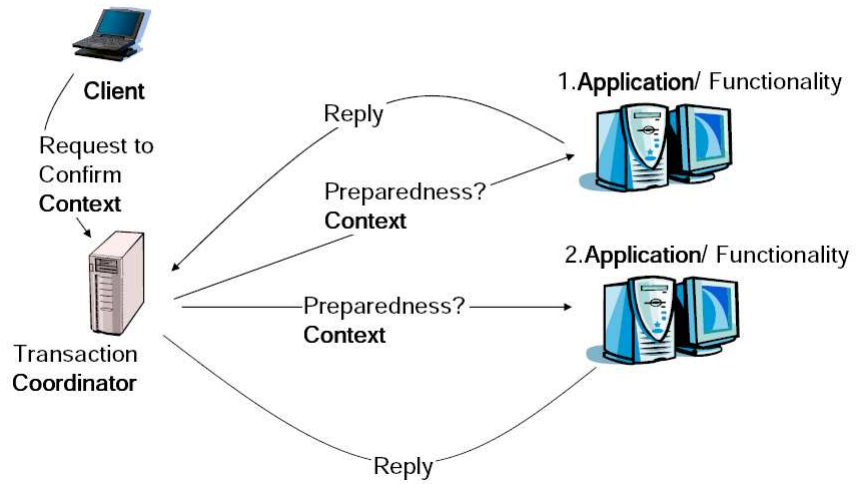


Two-Phase Commit Protocol (Abort) - 3 (fase 2)

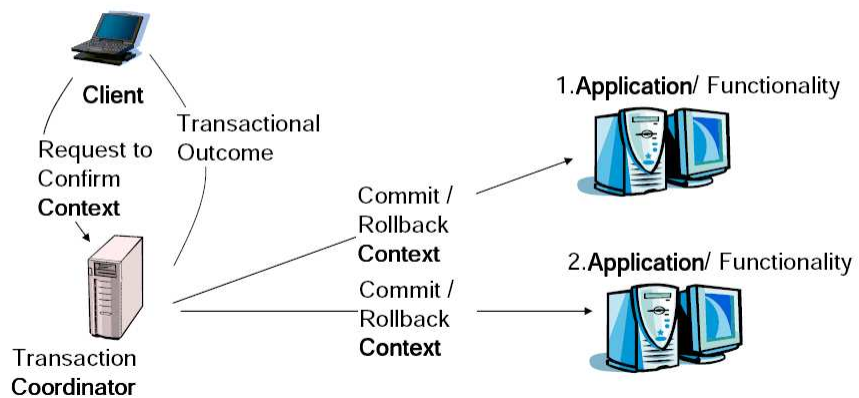
→ (consenso ad abortire l'azione atomica, tornando alla situazione definita dagli ultimi checkpoint)



Phase one



Phase two



Linear two phase commit protocol

- *Nel 2PC il numero dei messaggi è $4N$, dove N è il numero dei partecipanti*
- **Variante lineare:**
 - Ad ogni partecipante è assegnato un numero progressivo
 - Ogni partecipante conosce il nome del successivo e l'ultimo sa di essere l'ultimo
 - I partecipanti comunicano, dal primo verso l'ultimo, di essere arrivati al punto di commit (fase 1)
 - Se tutto va bene l'ultimo risponde OK e il messaggio fluisce in senso inverso (fase 2)
 - Se durante la fase 1 qualcuno decide di abortire il messaggio "Abort" fluisce nei due sensi.
- **Numero totale dei messaggi $2N$.**

Linear two phase commit protocol

- *Il protocollo lineare è appropriato se:*
 - Il meccanismo di comunicazione è costoso e in assenza di broadcast, oppure la tipologia della rete è ad anello
 - Non è necessario prevedere concorrenza durante le fasi 1 e 2
 - La struttura dei partecipanti è conosciuta staticamente
- *Il protocollo generale è appropriato se:*
 - Esiste il meccanismo di broadcast (in questo caso il totale dei messaggi è $2N+2$)
 - Si vuol privilegiare il parallelismo tra i partecipanti
 - La struttura e il numero dei partecipanti variano dinamicamente

Algoritmi distribuiti

2PC come primo esempio di algoritmo distribuito per garantire una nozione di consistenza su una computazione distribuita, anche a fronte di guasti.

Diversi altri algoritmi:

Leader election protocol:

si sceglie un "leader" tra i nodi non guasti, ad esempio per svolgere il ruolo di coordinatore nel 2PC

Group membership protocol: (Clustering)

In caso di guasto di un nodo appartenente a un gruppo (o cluster), se ne sceglie un altro da un pool di nodi disponibili e non guasti.

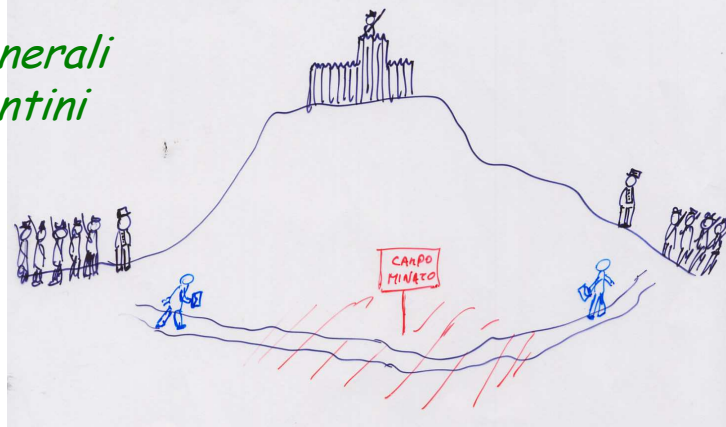
..... Applicazioni: quando si voglia garantire la consistenza di dati in tutto o parzialmente replicati in vari server

In generale, algoritmi che garantiscono proprietà di *Validity* e *Agreement* a fronte di possibili guasti...

Algoritmi distribuiti

- Molti di questi algoritmi sono limitati dal **principio di incertezza**:
- *Un nodo che ha richiesto un'azione remota non può sempre determinare, in un tempo finito, se l'azione è stata eseguita o no.*
- Paradosso dei generali bizantini

I generali bizantini



- **Protocollo per sincronizzare gli attacchi**

Non esiste un protocollo di durata massima fissata.

Dimostrazione: Supponiamo che esistano protocolli di questo tipo, e sia P quello più corto (in termini di numero di messaggi scambiati). Se l'ultimo messaggero salta su una mina, allora:

- o era inutile, ma allora P non era il più corto
- o era utile, ma allora P non funziona