Luca Cabibbo

Architettura dei Sistemi Software

Disponibilità

dispensa asw240

marzo 2021

Anything that can go wrong, will go wrong.

Murphy's law

- Riferimenti

- Luca Cabibbo. **Architettura del Software**: Strutture e Qualità. Edizioni Efesto, 2021.
  - Capitolo 10, Disponibilità
- Scott, J. and Kazman, R. Realizing and Refining Architectural Tactics: Availability. Technical report CMU/SEI-2009-TR-006. 2009.
- Abbott, M.L. and Fisher, M.T. **The Art of Scalability**: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise, second edition. Addison-Wesley, 2015.
- Nygard, M. **Release It!** Design and Deploy Production-Ready Software, second edition. Pragmatic Bookshelf, 2018.
- Microsoft. Microsoft Application Architecture Guide: patterns & practices, second edition. Microsoft Press, 2009.

- Obiettivi e argomenti

- Obiettivi
  - presentare la qualità della disponibilità
  - illustrare alcune attività e tattiche per la progettazione per la disponibilità
- Argomenti
  - disponibilità
  - progettare per la disponibilità
  - discussione

* Disponibilità

- Intuitivamente, la disponibilità di un sistema software riguarda
  - la possibilità che, durante l'uso del sistema, si verifichino dei guasti – negli elementi (hardware o software) del sistema
  - in corrispondenza, il modo con cui il sistema risponde a questi guasti
    - il sistema potrebbe continuare a funzionare, oppure si potrebbe verificare un fallimento

Disponibilità

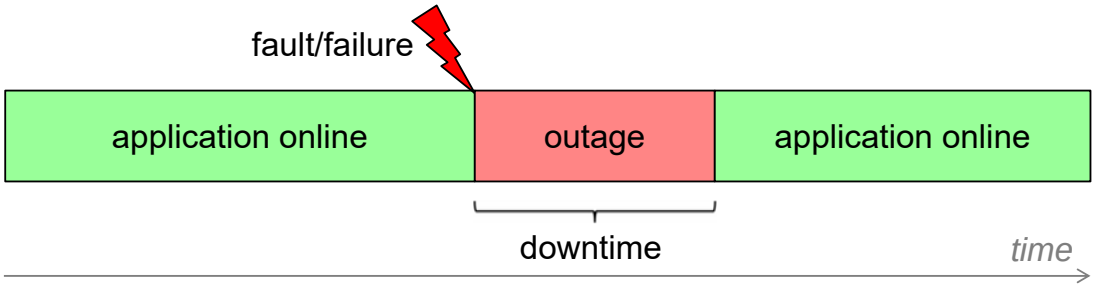
- **Disponibilità** (*availability*)
  - la capacità di un sistema di essere completamente o parzialmente funzionante come e quando richiesto
- **Tolleranza ai guasti**/**Resilienza** (*fault tolerance*/*resilience*)
  - la capacità del sistema di operare come richiesto, anche a fronte di guasti di componenti hardware o software del sistema
- **Capacità di recupero** (*recoverability*)
  - la capacità di recupero di un sistema dai fallimenti – ovvero, di recuperare i dati e rendere il sistema nuovamente operativo dopo un fallimento, entro tempi predefiniti e accettabili
- **Alta disponibilità** (*high availability*, *HA*) disponibilità
  - la combinazione di disponibilità, tolleranza ai guasti e capacità di recupero

Disponibilità

- La disponibilità è la capacità di un sistema software di essere pronto a erogare i propri servizi, quando vengono richiesti dai suoi utenti
  - una qualità complessa, che riguarda il modo con cui il sistema affronta i *guasti* (*fault*) – per evitare o minimizzare i *fallimenti* (*failure*)
    - un *guasto* è una problematica all'interno del sistema
    - un *fallimento* è quando il sistema non eroga più, all'esterno, un servizio in modo coerente con la sua specifica
  - un sistema potrebbe essere *tollerante ai guasti* (*fault tolerance*) oppure potrebbe essere in grado di effettuare un *ripristino* (*recovery*) a fronte di guasti
  - complessivamente, riguarda il tempo in cui il sistema è attivo e disponibile (appunto) a erogare i propri servizi

Interruzioni di servizio e RTO

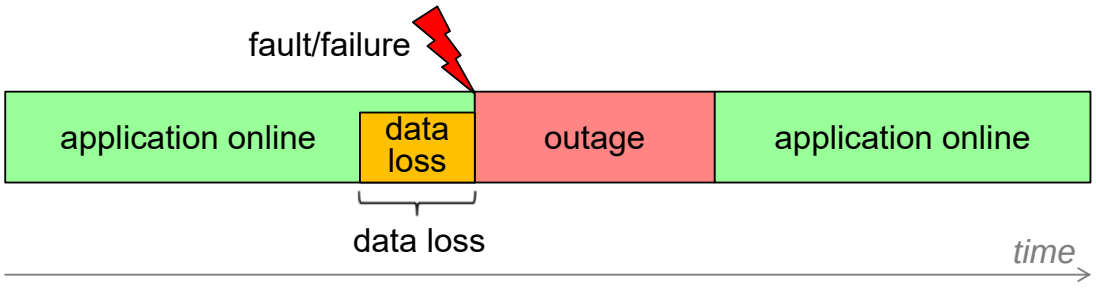
- Un *fallimento* è quando il sistema non eroga più un servizio in modo coerente con la sua specifica
  - può causare un'*interruzione di servizio* (*service outage*)



  - un possibile obiettivo di disponibilità di un sistema
    - minimizzare il tempo delle interruzioni di servizio a fronte dei possibili guasti (*Recovery Time Objective*, *RTO*)

Perdita di dati e RPO

- Un *fallimento* è quando il sistema non eroga più un servizio in modo coerente con la sua specifica
  - può causare una *perdita di dati* (*data loss*)



  - un possibile obiettivo di disponibilità di un sistema
    - minimizzare la perdita di dati a fronte dei possibili guasti (*Recovery Point Objective*, *RPO*)

Che cosa è la disponibilità

- La *disponibilità* può essere definita anche come [SSA]
  - la capacità di un sistema di essere completamente o parzialmente funzionante, come e quando richiesto
  - anche a fronte di guasti di componenti del sistema
    - in modo che eventuali guasti non comportino un fallimento totale dell'intero sistema
    - oppure che comportino un fallimento dal quale è possibile un recupero – preferibilmente entro tempi concordati
    - in alcuni casi può essere accettabile un funzionamento "parziale" del sistema

Che cosa è la disponibilità

- Un'altra caratterizzazione della *disponibilità* [SAP]
  - la capacità di un sistema di mascherare o riparare guasti
  - in modo tale che la durata complessiva delle interruzioni di servizio non ecceda un valore richiesto nell'ambito di un certo intervallo di tempo specificato

Disponibilità come misura

- La “disponibilità” come misura
  - la **disponibilità** (o *uptime*) è una misura della quantità di tempo in cui un sistema (o servizio o componente) è disponibile in un certo periodo di tempo – ovvero, è operativo ed è in grado di erogare i propri servizi

$$\text{disponibilità} = \frac{\text{periodo di tempo in cui il sistema è operativo}}{\text{periodo di riferimento}}$$

  - di solito è espressa come un percentuale o come “numero di 9”
  - attenzione, talvolta include i downtime pianificati, ma talvolta no

Disponibilità come misura

Uptime (%)	Downtime (%)	Downtime per year	Downtime per week
90%	10%	36.5 days	16:48 hours
99%	1%	3.65 days	1:41 hours
99.9%	0.1%	8:45 hours	10:05 minutes
99.99%	0.01%	52:30 minutes	1 minute
99.999%	0.001%	5:15 minutes	6 seconds
99.9999%	0.0001%	31.5 seconds	0.6 seconds

Disponibilità come misura

- La misura della disponibilità di un sistema (o componente) può essere calcolata come la probabilità che esso fornisca i servizi specificati durante un certo periodo di tempo

$$\text{disponibilità} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$

  - in cui
    - *MTBF – Mean Time Between Failures* – è il tempo medio tra guasti
    - *MTTR – Mean/Maximum Time To Repair or Resolve* – è il tempo medio/massimo di riparazione/ripristino

Affidabilità

- L'affidabilità è una qualità correlata alla disponibilità
  - l'*affidabilità* (*reliability*) è la probabilità che un sistema (o componente) fornisca i servizi specificati per tutto un certo periodo di tempo T
    - ovvero, che funzioni senza fallimenti (o guasti) per tutto il periodo di tempo T
  - l'affidabilità è correlata al tempo medio tra guasti (MTBF)
    - per i più curiosi

$$\text{affidabilità}(T) = e^{-\frac{T}{\text{MTBF}}}$$

  - l'affidabilità e la disponibilità sono due qualità correlate ma distinte

- Pianificare per la disponibilità

- La specifica dei requisiti di disponibilità di un sistema software è in genere molto complessa
  - il sistema può offrire molti servizi – che possono fallire in modo indipendente tra loro
  - i requisiti di disponibilità dei diversi servizi possono essere differenti tra di loro – e possono riguardare
    - una limitazione delle interruzioni di servizio
    - la possibilità di fallire in modo parziale – “classi di servizio” e modalità di erogazione “ridotta”
    - requisiti differenti in momenti diversi
    - la possibilità di downtime pianificati – per la manutenzione
    - una limitazione della perdita dei dati

Pianificare per la disponibilità

- La specifica dei requisiti di disponibilità di un sistema software è in genere molto complessa
  - è necessario identificare e analizzare i possibili tipi di guasti
    - errori umani, errori nel software, guasti nell’hardware o nelle reti e disastri naturali
    - per ciascun tipo di guasto va valutata la probabilità, il possibile impatto sui servizi e la gravità
  - per ciascun tipo di guasto, va specificato l’impatto desiderato (per ottenerlo bisognerà prendere delle contromisure)
  - nei sistemi più critici può essere richiesta una soluzione di *disaster recovery*

Pianificare per la disponibilità

- La specifica dei requisiti di disponibilità di un sistema software è in genere molto complessa
  - le considerazioni fatte finora riguardano solo la *disponibilità tecnologica* di un sistema – che però è solo una componente della *continuità del business* – la capacità di un'organizzazione di continuare a esercitare il proprio business a fronte di guasti o altri eventi, anche catastrofici
  - in pratica, la quantità di disponibilità richiesta per un sistema va determinata sulla base di un'opportuna *valutazione economica*
    - valuta il costo delle conseguenze delle possibili interruzioni di servizio – che può avere conseguenze dirette e indirette
    - sulla base di questa informazione, determina quanto sei disposto a spendere per proteggere il sistema da queste possibili interruzioni di servizi

* Progettare per la disponibilità

- Alcune attività nella progettazione per la disponibilità [SSA]
  - identifica i guasti che possono verificarsi
  - per ciascun guasto o combinazione di guasti
    - valuta la gravità e l'impatto sui servizi
    - identifica un'opportuna strategia per la gestione di tale guasto
  - fai anche delle considerazioni complessive sulla disponibilità (sia tecnologica che di business)

Progettare per la disponibilità

- Ci concentriamo soprattutto sulle tattiche architetturali per aumentare la disponibilità di un servizio – a fronte di alcuni possibili guasti del sistema
  - per impedire ai guasti di provocare fallimenti del servizio
  - oppure, per limitare l'effetto del fallimento e rendere possibile il ripristino del servizio
- Tre categorie principali di tattiche
  - rilevare guasti
  - gestire il ripristino da guasti
  - prevenire guasti

- Disponibilità e ridondanza

- La progettazione per la disponibilità è di solito basata sulla *ridondanza*
  - ovvero, sulla presenza di più "copie" o "repliche" di uno o più elementi – per eliminare i *punti di fallimento singoli*
  - gli elementi ridondati possono essere elementi hardware oppure elementi software (componenti o dati)
  - è spesso necessario "ridondare" molti elementi
  - oltre agli elementi ridondati, sono spesso necessari anche degli elementi o delle responsabilità aggiuntive
- Consideriamo soprattutto la ridondanza di nodi (computer usati come server) che implementano servizi
  - i nodi vengono organizzati in cluster e in "gruppi di protezione"

Ridondanza e affidabilità

- Un po' di matematica
  - supponiamo che un elemento E abbia un'affidabilità R – ovvero, ha probabilità 1-R di guastarsi nell'unità di tempo
  - consideriamo poi un sistema composto da N repliche di E (al posto del singolo elemento E), nelle seguenti ipotesi
    - il sistema è disponibile se almeno una delle repliche di E lo è
    - i guasti tra le repliche di E sono indipendenti tra loro
  - sotto tali ipotesi, il sistema si guasta (non è disponibile) se tutte le repliche di E si guastano contemporaneamente
    - questo avviene con una probabilità $(1\text{-}R)^N$ e l'affidabilità complessiva del sistema è dunque $1\text{-}(1\text{-}R)^N$
  - ad es., se l'affidabilità di un singolo elemento non replicato è R=99% (0.99), allora l'affidabilità di un sistema con 2 repliche è 99.99%

Cluster

- Un cluster è un gruppo di più nodi interconnessi, che lavorano assieme per offrire un insieme di servizi come un sistema singolo
  - ad es., un cluster potrebbe offrire due servizi A e B, mediante due nodi X e Y, in cui
    - X è un nodo "primario/attivo" per il servizio A e "secondario/di riserva" per il servizio B
    - Y è un nodo "primario/attivo" per il servizio B e "secondario/di riserva" per il servizio A
  - i cluster hanno in genere configurazioni complesse
  - è più semplice ragionare inizialmente in termini di "gruppi di protezione"

- Gruppi di protezione

- Un **gruppo di protezione** per un servizio S è un insieme di nodi dedicati all'erogazione del servizio S
  - in un gruppo di protezione, in un dato momento
    - uno o più nodi sono *attivi* – ovvero, sono utilizzati per erogare effettivamente il servizio ai client del servizio
    - possono essere anche delle *riserve* ridondanti
    - di solito servono anche altri elementi oltre a questi nodi
  - il caso più semplice di gruppo di protezione è la ridondanza 1+1
  - attenzione, le riserve per un servizio S possono essere utilizzate in modi diversi

Ripristino dai guasti

- In un gruppo di protezione, il ripristino di un nodo attivo (a seguito di un suo guasto) viene in genere gestito come segue
  - rilevamento del guasto del nodo – automaticamente o mediante un intervento manuale
  - preparazione di un nodo di riserva per sostituire il nodo che si è guastato
  - attivazione di questo nodo – l'elemento di bilanciamento del carico gli inizia ad assegnare richieste
  - il tempo richiesto per eseguire queste attività ha influenza sulla disponibilità del servizio (e sul costo della soluzione)

- Tattiche per la disponibilità

- Categorie principali di tattiche per la disponibilità di [SAP]
  - *detect faults*
    - hanno lo scopo di rilevare guasti
  - *recover from faults*
    - hanno a che fare con il ripristino del sistema a fronte di guasti
  - *prevent faults*
    - hanno lo scopo di prevenire guasti
  - *exceptions* – una categoria “trasversale”, che riguarda la gestione dei guasti nel software

Osservazioni

- Queste tattiche per la disponibilità
  - fanno riferimento soprattutto alla disponibilità dei nodi – ma spesso possono essere applicate anche ad altri elementi
  - sono di solito presenti in molti ambienti di esecuzione standard – come sistemi operativi, infrastrutture software e middleware
    - è importante comprendere tali tattiche e il loro effetto
  - il lavoro dell’architetto sarà poi spesso quello di scegliere e valutare (piuttosto che implementare) le tattiche per la disponibilità da applicare nella progettazione di un sistema

- Detect faults

- Prima di poter intraprendere qualunque azione che riguardi un guasto (come un ripristino automatico), la presenza del guasto deve essere rilevata
- Una tattica generale per il rilevamento automatico dei guasti
- *Monitor*
  - un elemento utilizzato per effettuare il monitoraggio dello stato di salute delle altre parti di un gruppo di protezione o di un sistema
  - il monitoraggio può essere effettuato mutuamente dai nodi stessi che appartengono al gruppo di protezione – oppure da un elemento specializzato

Detect faults

- Due ulteriori tattiche principali per il monitoraggio
- *Ping/echo*
  - il monitor invia dei messaggi *ping* al monitorato, che risponde con dei messaggi *echo* – lo scambio di messaggi, di solito asincrono, consente di determinare la raggiungibilità tra elementi e il tempo di roundtrip del canale di comunicazione
- *Heartbeat*
  - l'elemento monitorato emette periodicamente un messaggio *heartbeat* – mentre il monitor è in ascolto

Detect faults

- Esistono anche altre tattiche per il rilevamento dei guasti, per valutare in modo più preciso lo stato di salute del sistema – ad esempio
  - per rilevare tentativi di frodi o attacchi DoS (Denial of Service)
  - usare nella comunicazione checksum, timestamp o timeout
  - verificare la “ragionevolezza” dei messaggi (richieste e risposte) scambiati tra gli elementi
  - richiedere l’esecuzione di una stessa operazione a più elementi software distinti – e confrontare le loro risposte (voting)
    - se gli elementi software sono identici, una risposta differente dalle altre consente di rilevare un problema nell’hardware
    - elementi software sviluppati indipendentemente possono consentire di rilevare errori nel software

- Recover from faults

- Due categorie di tattiche per il ripristino da guasti
  - tattiche che preparano ed eseguono il ripristino di un nodo che si è guastato (tattiche per il *failover*)
  - tattiche relative alla reintroduzione di un nodo che è stato “riabilitato” dopo che si era guastato

Recover from faults

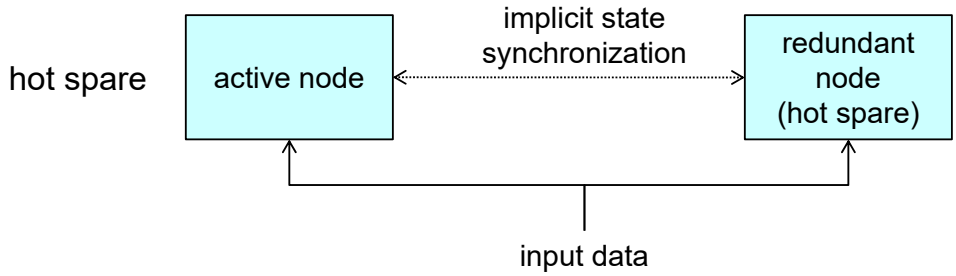
- Nel ripristino da guasti, un nodo di riserva può subentrare a un nodo che si è guastato solo dopo che
  - è stato rilevato il guasto di un nodo
  - il nodo di riserva è stato “preparato” per sostituire questo nodo
    - acquisizione e configurazione del nodo
    - installazione e configurazione del software sul nodo, e suo avvio
    - sincronizzazione dello stato del nodo
  - il nodo preparato è stato reso attivo
  - come si può aumentare la disponibilità? ma a quale costo?

Recover from faults

- Tre tattiche principali per il ripristino da guasti (per preparare ed eseguire il ripristino di un nodo che si è guastato)
  - *hot spare*
  - *warm spare*
  - *cold spare*
  - queste tattiche fanno riferimento a tre livelli diversi del grado di “preparazione” dei nodi di riserva – e portano a tre livelli differenti di disponibilità (e di costo)

Recover from faults

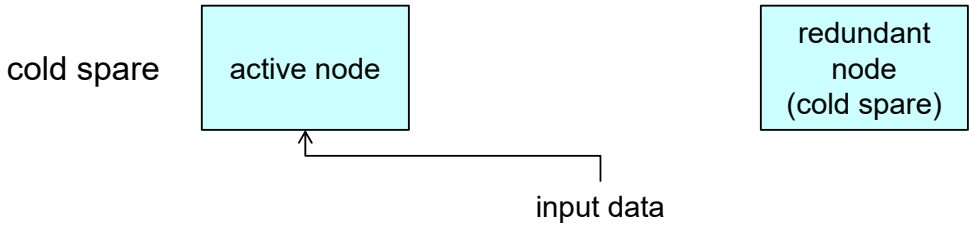
- *Active redundancy (hot spare)*



  - tutti i nodi (attivi e di riserva) in un gruppo di protezione sono preparati e avviati, ed elaborano tutti gli eventi di input – sono sincronizzati in modo continuo
  - in caso di guasto di un nodo attivo, un nodo di riserva lo può sostituire immediatamente
    - il downtime potrebbe essere nell'ordine dei millisecondi
  - la ridondanza può essere anche nel canale di comunicazione (rete) e nelle unità disco (condivise)

Recover from faults

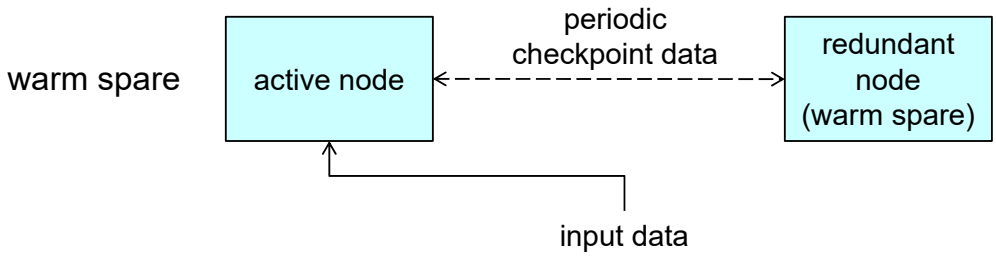
- *Spare (cold spare)*



  - i nodi di riserva di un gruppo di protezione sono tenuti fuori servizio fino a quando non è necessario sostituire un nodo
    - i nodi di riserva sono spesso solo preparati (configurati) per poter sostituire altri nodi – ma non sincronizzati
  - in caso di guasto di un nodo attivo, un nodo di riserva viene configurato e avviato, il suo stato viene sincronizzato (vedi *Rollback*) – poi avviene la sostituzione
    - il downtime potrebbe essere nell'ordine dei minuti
    - ma può essere anche più lungo o più breve – in quali casi?

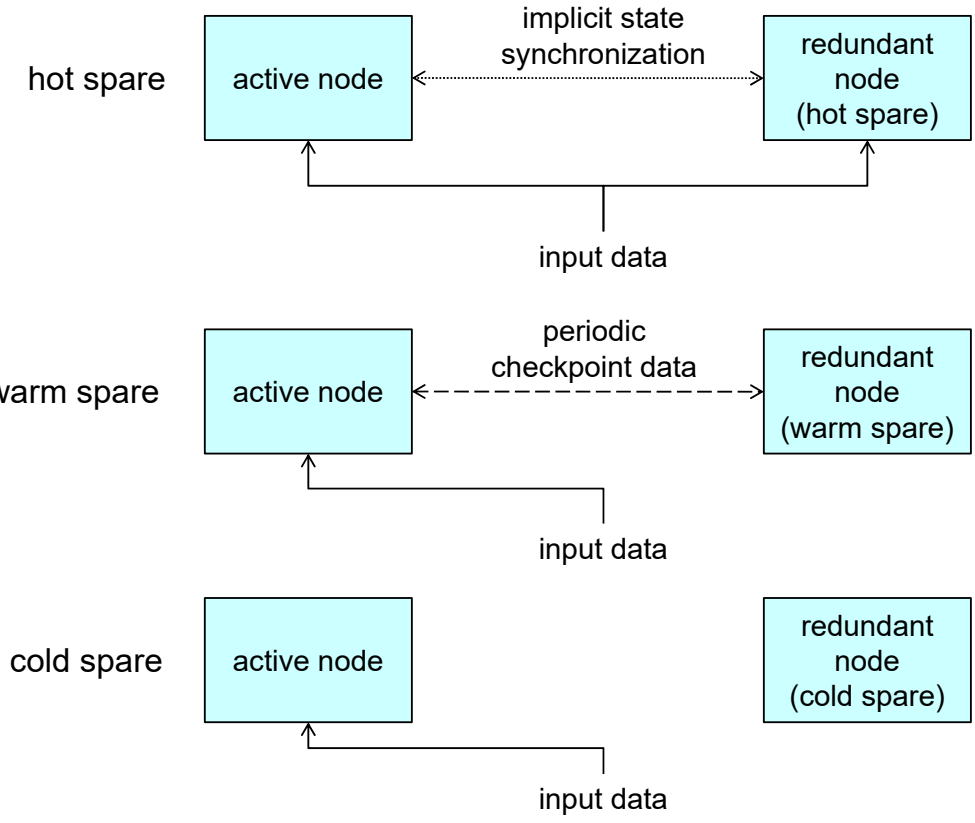
Recover from faults

- *Passive redundancy (warm spare)*



  - solo i nodi attivi del gruppo di protezione ricevono ed elaborano gli eventi di input – i nodi di riserva sono sincronizzati in modo parziale e periodico
  - in caso di guasto di un nodo attivo, un nodo di riserva lo può sostituire, ma solo dopo aver completato la sincronizzazione del suo stato
  - è un compromesso (affidabilità/complessità) tra le due tattiche precedenti – il downtime può essere nell'ordine dei secondi

Hot spare, warm spare, cold spare



Recover from faults e stato del servizio

- Lo stato del servizio di interesse per un gruppo di protezione ha un impatto rilevante nella modalità di ripristino dai guasti
  - ad es., se il servizio è stateless, allora non è necessaria nessuna sincronizzazione tra i nodi
  - se il servizio è stateful e lo stato del servizio è ripartito (ad es., in “shard”) tra i nodi attivi del gruppo di protezione (anziché essere replicato interamente su tutti i nodi attivi), allora la sincronizzazione va gestita “ad hoc”

Recover from faults e clustering

- Le tattiche per il ripristino da guasti possono essere implementate mediante clustering e failover (discussi più avanti)
  - un cluster può essere considerato un pattern architetturale – basato sull’applicazione di un insieme “complesso” di scelte di progettazione – in un cluster è sempre possibile riconoscere l’applicazione di più tattiche per la disponibilità
  - ciascuna tattica corrisponde invece a un’opzione di progettazione “elementare”
  - alcune configurazioni di cluster sono descritte brevemente più avanti

Recover from faults

- Altre tattiche per il ripristino da guasti
  - *Rollback*
    - questa tattica ha lo scopo di sostenere il ripristino dello stato del sistema – in combinazione con altre tattiche per la disponibilità
    - ad es., mediante *checkpoint* e *log* (delle transazioni)
  - *Software upgrade*
    - il periodo richiesto per effettuare l'aggiornamento di un servizio software può corrispondere a un'interruzione di servizio
    - questa tattica ha l'obiettivo minimizzare o anche evitare le interruzioni di servizio relative agli aggiornamenti del software – anche se non si tratta di guasti

Recover from faults

- Altre tattiche per il ripristino da guasti
  - in caso di guasti, prova a mantenere la disponibilità delle funzioni del sistema più importanti e critiche, arrestando invece le funzioni meno importanti (*Degradation*)
  - il ripristino cerca di riassegnare responsabilità agli elementi sopravvissuti ai guasti, sempre cercando di mantenere attive le funzioni più importanti (*Reconfiguration*)
  - prova ad assumere che il guasto sia spurio o transitorio – ignorando il guasto (*Ignore faulty behaviour*), oppure provando a ripetere l'esecuzione dell'operazione in cui si è verificato il guasto (*Retry*)

Recover from faults

- Alcune tattiche relative alla reintroduzione di componenti che sono stati (auspicabilmente) “riparati” – dopo che si erano guastati
  - *Shadow*
    - esegui temporaneamente il nodo da reintrodurre in “modalità ombra” (come riserva) – per verificare se il guasto è stato effettivamente riparato
  - *State resynchronization*
    - verifica la correttezza dello stato dell’elemento da reintrodurre – oppure da ripristinare – prima di reintrodurlo
  - *Escalating restart*
    - consenti di riavviare i servizi in modo granulare – ad es., per evitare di dover riavviare servizi su cui un guasto non ha avuto impatto

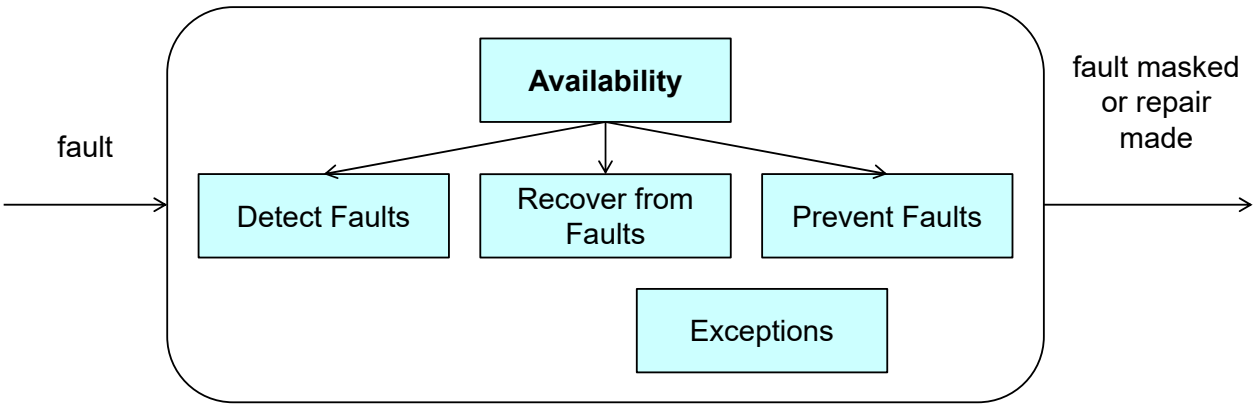
- Prevent faults

- Le tattiche in questa categoria hanno l’obiettivo di favorire la prevenzione dei guasti – per evitare che si verifichino e di doverli rilevare e gestire
  - *Removal from service*
    - rimuovere temporaneamente un componente del sistema per fargli svolgere attività per mitigare possibili fallimenti di sistema
  - *Transactions*
    - una transazione è una sequenza di passi elementari che viene complessivamente considerata un’operazione atomica – da svolgere oppure annullare completamente
    - le transazioni consentono di prevenire inconsistenze nello stato del sistema, nel caso di fallimento di uno dei loro passi elementari

- Exceptions

- Le tattiche per le *eccezioni* riguardano l'applicazione dello strumento linguistico delle eccezioni alle tre precedenti categorie di tattiche per la disponibilità – per gestire guasti nel software
- *Exception detection* – per detect faults
  - il rilevamento delle eccezioni ha lo scopo di identificare le condizioni che devono alterare il normale flusso di esecuzione
- *Exception handling* – per recover from faults
  - la gestione di eccezioni può essere usata per il ripristino da guasti – notificati appunto come eccezioni
- *Increase competence set* – per prevent faults
  - la competenza di un programma è l'insieme di stati in cui il programma ha competenza per operare
- *Exception prevention* – per prevent faults
  - ha lo scopo di prevenire il verificarsi di eccezioni

- Tattiche per la disponibilità



- Discussione

- Riguardo alle tattiche per la disponibilità di [SAP]
  - l'applicazione di una tattica può richiedere anche l'applicazione di altre tattiche – in altre categorie
  - l'applicazione di una tattica non impedisce l'applicazione di altre tattiche – anche nella stessa categoria
  - l'applicazione di una tattica per la disponibilità può essere efficace a fronte di un certo tipo di guasto – ma non lo è necessariamente con tutti i possibili guasti (allo stesso modo)
  - molte configurazioni richiedono delle considerazioni speciali

- Fault isolation

- L'*isolamento dei guasti* (*fault isolation*) [Abbott and Fisher] è un altro importante principio di progettazione per la disponibilità
  - lo scopo è impedire la *propagazione dei guasti* – da un servizio ad altri servizi o all'intero sistema
    - per evitare fallimenti a cascata, e per fare in modo che il sistema sia parzialmente funzionante a fronte di guasti
  - i fallimenti a cascata e le reazioni a catena sono infatti tra i principali problemi alla stabilità e alla disponibilità dei sistemi software [Nygard]
    - la causa sono spesso elementi troppo accoppiati e integrati tra loro anziché isolati

Fault isolation e affidabilità

- Un po' di matematica
  - consideriamo un componente A_1 che dipende (direttamente o indirettamente) da altri componente A_2 … A_N
  - ognuno di questi componenti ha un'affidabilità (in isolamento) R – ovvero, ha probabilità 1-R di guastarsi nell'unità di tempo
  - per fruire di un servizio di A_1, tutti i componenti A_1 … A_N devono essere contemporaneamente attivi – ovvero, non sono isolati
  - qual è l'affidabilità combinata del componente A_1?
    - è R^N – ad es., se R=99.9% e N=10, allora $99.9\%^{10}=99.0\%$
    - in assenza di isolamento dei guasti, l'affidabilità combinata diminuisce in modo esponenziale rispetto a N
  - l'isolamento dei guasti cerca invece di impedire questa propagazione dei guasti

Fault isolation

- L'isolamento dei guasti è utile soprattutto per
  - isolare un servizio di alto valore di business da altri servizi
  - isolare servizi particolarmente soggetti a guasti

Fault isolation

- Progettare per l'isolamento dei guasti
  - identifica nel sistema le zone da isolare rispetto ai guasti – chiamate swim lane, pod, shard o bulkhead
  - linee guida
    - zone distinte non devono comunicare in modo sincrono (che può causare la propagazione dei guasti)
    - zone distinte non devono condividere niente – ad es., una medesima base di dati
  - ecco alcune possibili soluzioni (relative alla comunicazione)
    - comunica in modo asincrono
    - usa timeout stringenti
    - ripeti la comunicazione (utile solo se i guasti sono transienti)
    - fallback – comunica con un componente diverso
    - usa un circuit breaker – combina più tecniche

Fault isolation

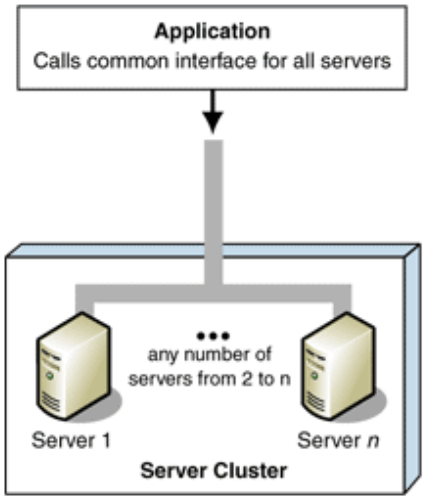
- *Circuit Breaker* [Nygard] è un pattern per l'isolamento dei guasti
  - un "interruttore automatico" o "salvavita"
  - è un intermediario che incapsula la chiamata a un servizio remoto (che potrebbe non essere disponibile)
    - si può trovare, in due stati: "chiuso" o "aperto"
    - se il circuito è "chiuso", allora il circuit breaker chiama (prova a chiamare) il servizio remoto
    - se il circuit breaker rileva dei problemi nella comunicazione, allora (dopo un certo numero di errori) "apre" automaticamente il circuito, in modo che le chiamate successive non provino nemmeno a raggiungere il servizio remoto – il circuit breaker può anche chiamare un servizio alternativo (fallback)
    - dopo un po', il circuito prova a richiudersi automaticamente

- Cluster

- Un **cluster** è un gruppo di due o più nodi interconnessi – chiamati *nodi* o *membri* del cluster – che lavorano assieme per offrire uno o più servizi come un sistema singolo
  - un cluster consente di implementare i gruppi di protezione per uno o più servizi
  - per "sistema singolo" si intende che il gruppo dei nodi interconnessi che forma il cluster viene visto dall'utente/client di un servizio come una singola entità
  - un cluster, oltre ai nodi (server), comprende anche altri elementi, hardware (fisici o virtuali) e software
    - ad es., la rete, un servizio di appartenenza al cluster, un load balancer
  - due motivazioni principali per l'uso dei cluster (anzi tre) – disponibilità e scalabilità
  - qui consideriamo i cluster per l'alta disponibilità

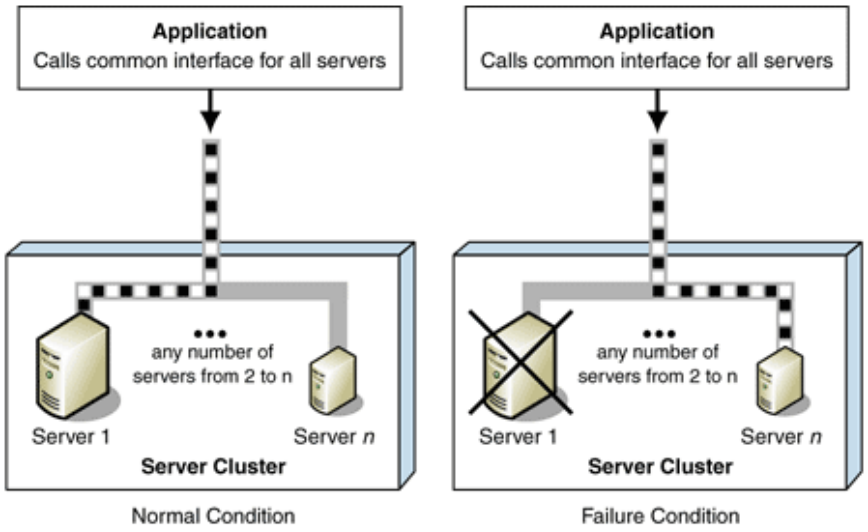
Configurazioni

- Esistono diverse configurazioni comuni per i cluster
  - la configurazione di base è un gruppo di due o più nodi che lavorano insieme per offrire un servizio come un sistema singolo



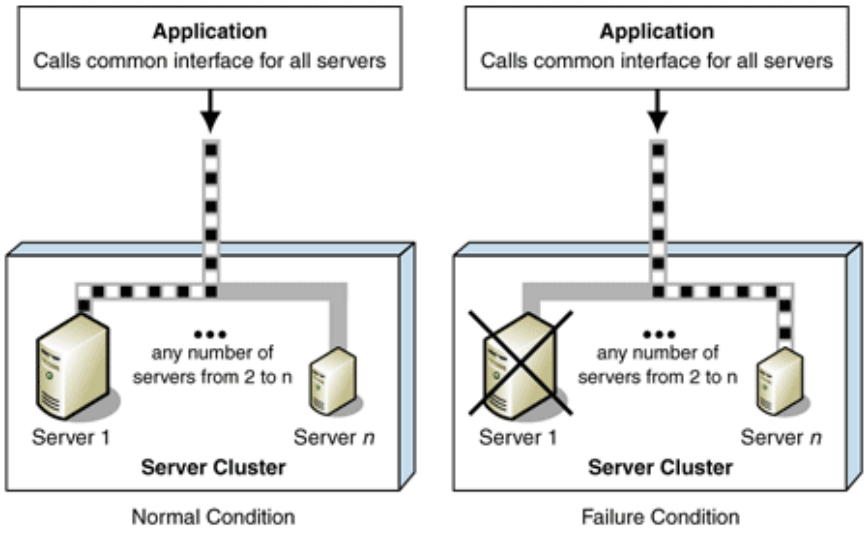
Cluster asimmetrico

- *Cluster asimmetrico*
  - un servizio viene erogato solo dal nodo attivo (*server primario*)
  - il *server secondario* (*standby server*) sostituisce il server primario nell'erogazione del servizio solo nel caso di un suo fallimento



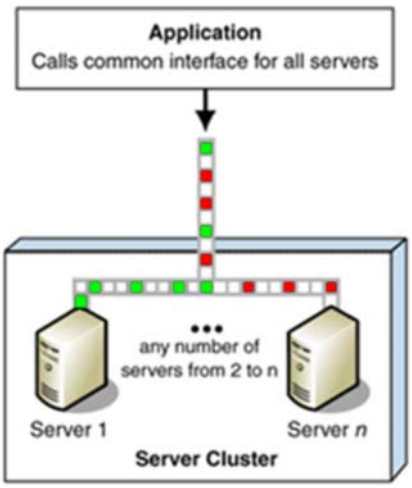
Failover

- *Failover*
  - è la sostituzione del server primario con il server secondario



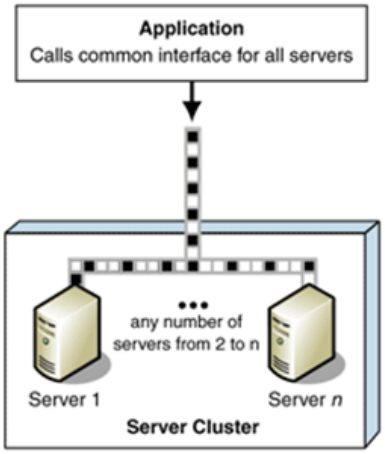
Cluster simmetrico

- *Cluster simmetrico*
  - utile quando un sistema che deve erogare più servizi
  - ciascun nodo del cluster svolge del lavoro utile, erogando uno o più servizi
  - un nodo può essere il server primario per uno o più servizi – ma anche il server secondario (standby server) per altri servizi
  - in caso di fallimento di un nodo, ciascuno dei suoi "servizi primari" viene riassegnato a un server secondario per quel servizio



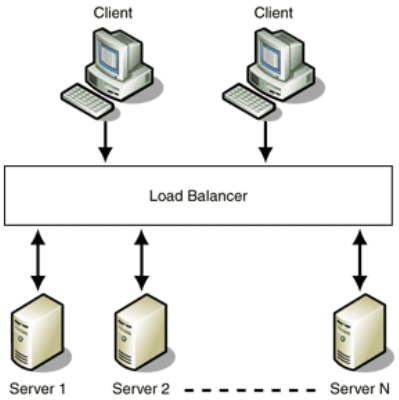
Cluster e load balancing

- *Cluster per load balancing*
  - un cluster di tipo simmetrico, in cui un servizio viene erogato attivamente da più nodi – al limite, tutti i nodi del cluster partecipano attivamente all'erogazione del servizio
  - è necessario un meccanismo aggiuntivo di load balancing
  - può essere necessario anche un meccanismo di sincronizzazione tra i nodi del cluster

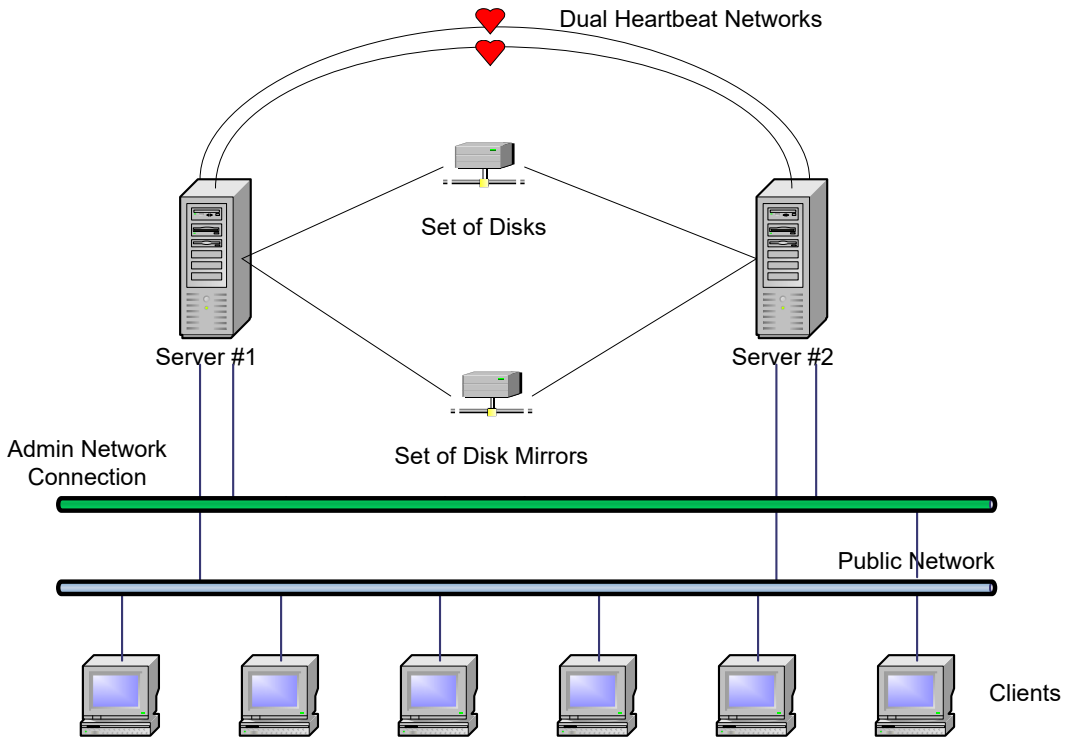


Cluster e load balancing

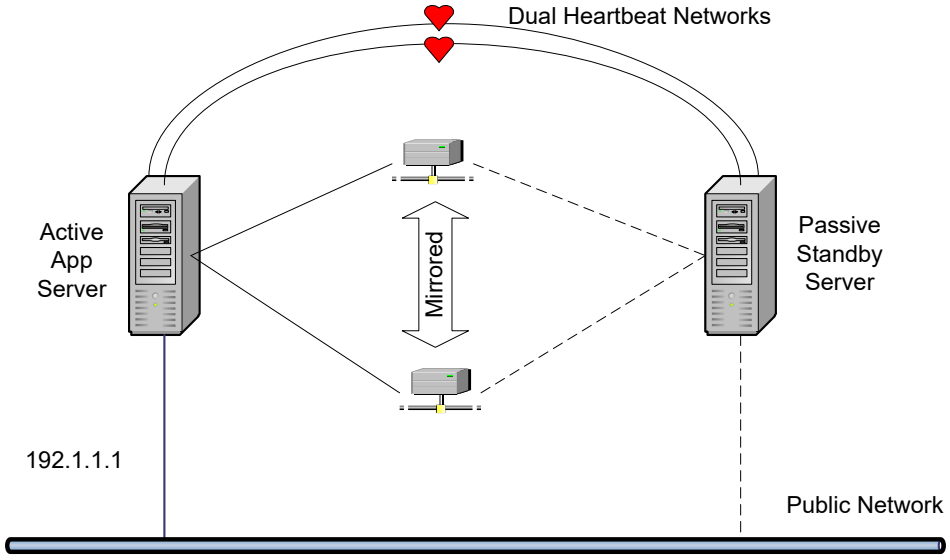
- *Load balancing*
  - le richieste per un servizio vengono ricevute da un load balancer – che le gira ai nodi del cluster che erogano quel servizio sulla base di un'opportuna politica
  - sostiene la scalabilità orizzontale



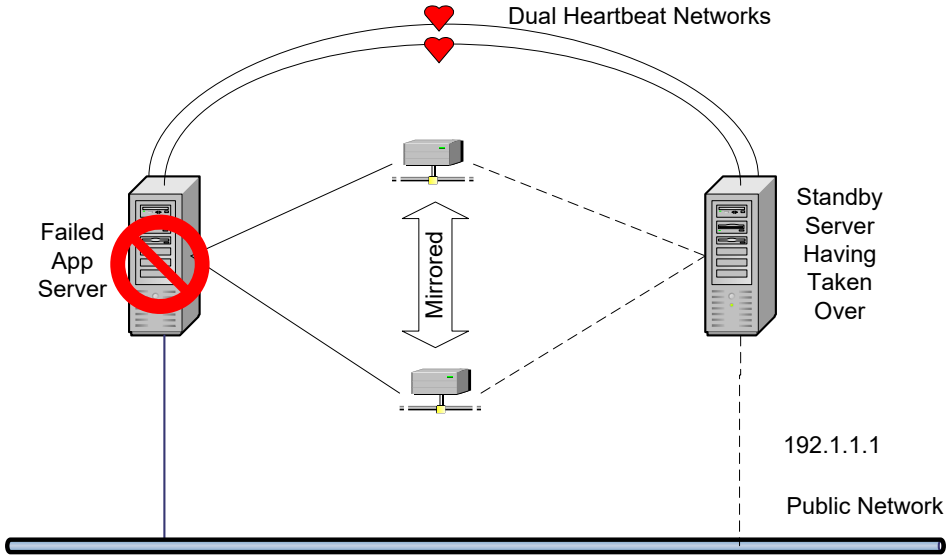
- Esempio - un semplice cluster



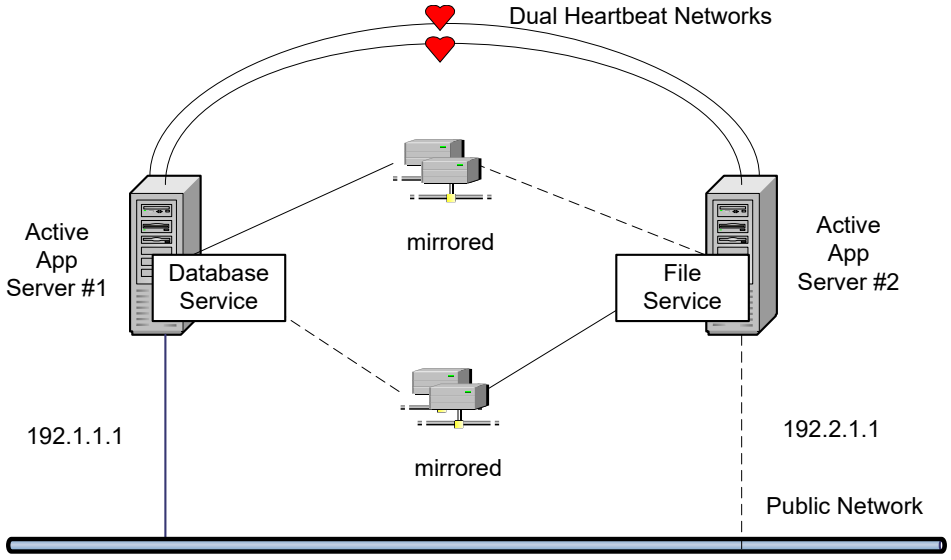
Cluster asimmetrico – prima di un failove



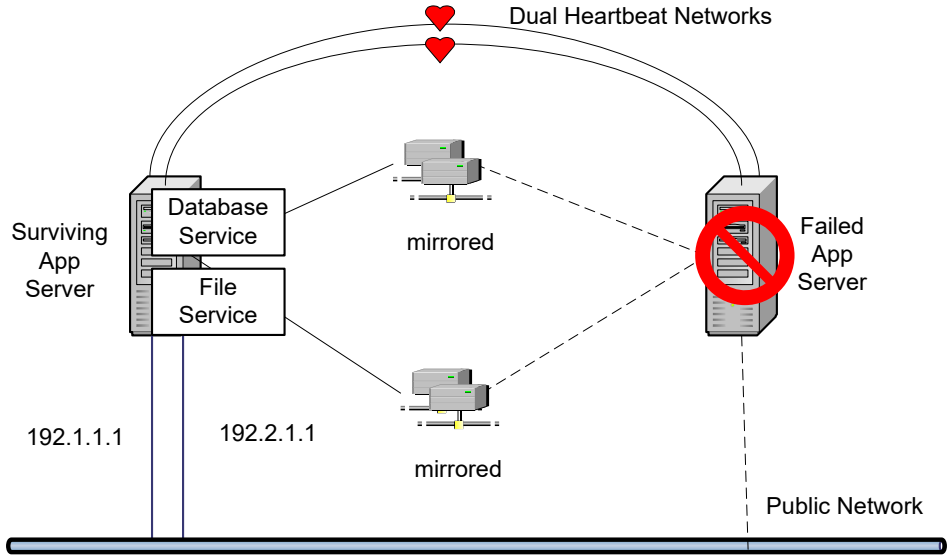
Cluster asimmetrico – dopo un failover



Cluster simmetrico – prima di un failover



Cluster simmetrico – dopo un failover



- Monitoraggio

- Il *monitoraggio* riguarda la capacità di osservare e controllare il comportamento di un sistema software mentre è in esecuzione nell'ambiente di produzione
  - l'obiettivo generale del monitoraggio è rendere più semplice la comprensione del comportamento del sistema e dei suoi componenti
  - il monitoraggio ha delle applicazioni importanti proprio nel contesto della disponibilità – per favorire il rilevamento di problemi e per sostenere la loro diagnosi
    - ad es., per capire come (provare a) ripristinare il sistema a seguito di un guasto o fallimento non previsto
  - ne parleremo meglio nel capitolo sul monitoraggio

- Altre tattiche per la disponibilità

- Altre tattiche (e attività) per la disponibilità, dalla prospettiva della disponibilità e della resilienza [SSA]
  - seleziona e usa tecnologie hardware e software mature – ad es., hardware tollerante ai guasti
  - usa cluster per l'alta disponibilità e meccanismi di bilanciamento dei carico
  - crea o applica soluzioni per la disponibilità del software
  - consenti la replicazione di componenti
  - applica soluzioni di backup ed effettua il log delle transazioni
  - identifica soluzioni di disaster recovery
  - progetta per il fallimento – come se i guasti e i fallimenti fossero la norma e non le eccezioni
  - verifica in modo pragmatico le soluzioni per la disponibilità

* Discussione

- La disponibilità è una qualità importante in molti sistemi software
  - è importante soprattutto nei sistemi che hanno requisiti stringenti in termini di safety (“sicurezza”) per le persone o l’ambiente – oppure che gestiscono informazioni critiche
  - è importante anche nelle situazioni di business in cui le interruzioni di servizio possono causare perdite economiche, danni di immagine e perdita di clienti – e dunque possono avere un impatto significativo per un’organizzazione
  - oggi sono disponibili un numero sempre maggiore di soluzioni tecnologiche per la disponibilità – che però vanno comprese e applicate bene, e vanno spesso anche integrate con l’applicazione di altre opzioni di progettazione e tattiche per la disponibilità
  - torneremo a parlare di disponibilità e affidabilità anche nel contesto della delivery del software