



Luca Cabibbo
**Architettura
dei Sistemi
Software**

Prestazioni

dispensa asw220
ottobre 2024

*An ounce of performance
is worth pounds of promises.*
Mae West



- Riferimenti

- Luca Cabibbo. **Architettura del Software: Strutture e Qualità**. Edizioni Efestò, 2021.
 - Capitolo 8, **Prestazioni**
- Liu, H.H. **Software Performance and Scalability: A Quantitative Approach**. Wiley, 2009.



- Obiettivi e argomenti

□ Obiettivi

- presentare la qualità delle prestazioni
- illustrare alcune attività e tattiche per la progettazione per le prestazioni
- fare delle considerazioni generali sulla progettazione per gli attributi di qualità – poiché questa è la prima qualità che viene discussa

□ Argomenti

- prestazioni
- progettare per le prestazioni
- discussione



* Prestazioni

Stiamo seguendo l'approccio della PROSPETTIVA: se non conosco nulla delle prestazioni, cosa devo sapere? A cosa devo stare attenta? Come posso intraprendere scelte di progetto? Come posso valutarle poi?

□ **Prestazioni** (*performance*)

- la capacità del sistema di eseguire in modo prevedibile entro il profilo di prestazioni temporali richiesto
- Le prestazioni hanno a che fare con il *tempo* – con la capacità del sistema software di soddisfare dei requisiti temporali
 - le prestazioni misurano quanto velocemente un sistema può completare alcuni compiti di elaborazione [Liu]
 - in generale, si vuole che un sistema abbia delle prestazioni adeguate (rispetto a un profilo di prestazioni richiesto)
- Le prestazioni sono importanti anche perché possono avere un impatto su altre qualità del software
 - ad es., su usabilità e scalabilità



Tipi di compiti e metriche per le prestazioni

- Due tipi principali di compiti computazionali – e le relative metriche più rilevanti per le prestazioni
 - **transazioni online** – di tipo interattivo
 - **tempo di risposta** (o **latenza**) – il periodo di tempo richiesto per gestire un evento o per completare l'esecuzione di un'operazione
 - **job** – non interattivi, di tipo batch Mi interessa l'elaborazione dell'intero lotto di richieste
 - **throughput** – il numero di job o transazioni che possono essere completate in un'unità di tempo
 - esistono anche altre metriche (che qui non consideriamo)
- Qui ci concentriamo soprattutto sulla gestione di eventi e sul tempo di risposta richiesto per gestire un evento



Scenari per prestazioni

Uno scenario è una situazione in cui può trovarsi il sistema insieme ad una descrizione di come si può comportare il sistema. Sono utili in sede di descrizione, analisi, valutazione...

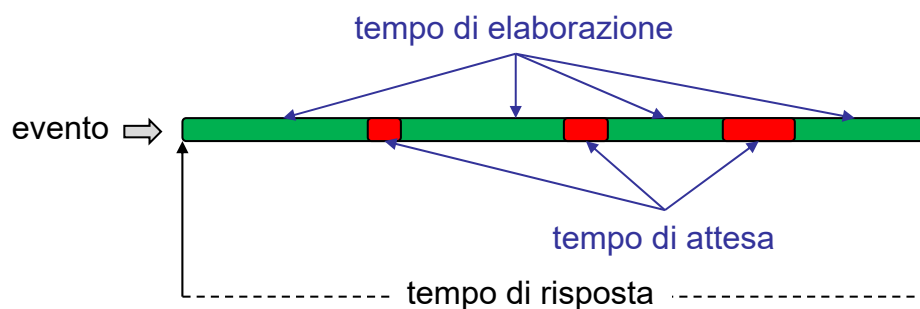
- Uno **scenario** per le prestazioni descrive, in genere, quanto rapidamente il sistema deve rispondere a una certa tipologia di eventi
 - può essere **specificato da**
 - tipo di eventi di interesse
 - distribuzione degli arrivi di questi eventi
 - risposta temporale desiderata
 - **quando vanno considerate le tattiche per le prestazioni?**
Quando l'architettura corrente NON soddisfa delle richieste qualitative per le prestazioni



- Considerazioni sulle prestazioni

elaborazione + attesa

- Due contributi principali al **tempo di risposta** a un evento
 - **tempo di elaborazione**
 - il tempo dedicato a generare la risposta all'evento, in cui vengono effettivamente consumate risorse computazionali
 - **tempo di attesa**
 - il tempo in cui la computazione è bloccata in attesa di qualche risorsa
- intuitivamente, il tempo di risposta può essere ridotto intervenendo su uno, sull'altro o su entrambi questi contributi



7

Prestazioni

Luca Cabibbo ASW



Considerazioni sulle prestazioni

□ Altre considerazioni sulle prestazioni

Ciascuna risorsa risponde anche in base al suo livello di utilizzazione (il peggioramento di prestazioni diventa esponenziale anziché magari lineare dal superamento di una certa soglia di utilizzazione)

- le prestazioni dipendono anche dall'**hardware** utilizzato, dalla sua capacità e quantità, e dal suo livello di **utilizzazione**
- le prestazioni di un sistema dipendono anche dal suo carico – di solito le prestazioni variano in modo non lineare con il carico
- nella progettazione per le prestazioni è necessario considerare la possibilità di eseguire computazioni in modo concorrente
 - la concorrenza ha in genere effetti benefici – ma può avere anche degli **inconvenienti**
 - ad es., richiede di considerare l'accesso e la contesa di risorse condivise e la sincronizzazione tra le computazioni

8

Prestazioni

Luca Cabibbo ASW



* Progettare per le prestazioni

- Alcune attività nella **progettazione per le prestazioni** [SSA]
 - specifica gli scenari e i requisiti più significativi per le prestazioni, e assegnagli delle priorità (Secondo rischio, valore per il cliente, ecc.)
 - crea e valuta modelli per le prestazioni
 - realizza e verifica praticamente dei prototipi – e anche il sistema stesso
 - analizza i risultati della valutazione
 - raffina l'architettura (se necessario) Con le tattiche per le prestazioni



- Tattiche per le prestazioni

- [SAP] propone due categorie principali di tattiche per le prestazioni – per migliorare il tempo di risposta a un evento
 - **tattiche per controllare la richiesta di risorse – *control resource demand*** ① Fanno in modo che gli elementi richiedano meno risorse
 - cercano di ridurre il tempo dedicato alla gestione di un evento, dal lato della richiesta delle risorse che sono necessarie per elaborare l'evento
 - **tattiche per gestire le risorse – *manage resources*** ② Fanno in modo che le risorse del sistema vengano usate meglio
 - operano dal lato dell'offerta delle risorse, per gestire le elaborazioni in modo più efficace



- Control resource demand



❑ Tattiche per ridurre la richiesta di risorse per elaborare un evento

❑ **Increase resource efficiency (Improve algorithm efficiency)**

- l'elaborazione di un evento richiede l'esecuzione di un algoritmo – se ne miglioriamo l'efficienza temporale, soprattutto nelle aree più critiche, il tempo di risposta diminuisce
- talvolta si può migliorare l'efficienza temporale scambiando il tempo con un'altra risorsa

Miglioramento dell'efficienza dell'algoritmo: quando si elabora un evento si usa un algoritmo. Se trovo un algoritmo migliore dal punto di vista dell'efficienza temporale il tempo di risposta potrebbe essere migliorato. Talvolta si può migliorare l'efficienza temporale ad esempio salvando dei risultati per riusarli in altre situazioni.

Nota: se in un algoritmo ci sono parti lineari, quadratiche e cubiche, per migliorarne le prestazioni temporali devo migliorarne le parti cubiche, non serve migliorare le parti lineari



Control resource demand



❑ Tattiche per ridurre la richiesta di risorse per elaborare un evento

❑ **Reduce overhead**

- l'uso di intermediari aumenta le risorse consumate nell'elaborazione di un evento – la loro eliminazione può ridurre il tempo di risposta
- gli intermediari sono però spesso importanti per sostenere altre qualità – pertanto, è spesso necessario trovare dei compromessi sul loro utilizzo

Altra tattica. Nelle nostre elaborazioni per gestire l'evento è possibile che si usino componenti software che fanno elaborazioni aggiuntive per gestire l'evento (es. le componenti software sono più di uno quindi devono comunicare, oppure si fanno passaggi di cifratura ecc) che causano overhead. A volte questi intermediari migliorano alcune qualità per questo servono compromessi sull'utilizzo.



- ❑ Per ridurre la richiesta di risorse, con lo scopo di migliorare il tempo di elaborazione, è talvolta possibile ridurre la quantità di elaborazioni da svolgere
 - limitare il tempo di elaborazione di un evento – ad es., calcolando un risultato approssimato anziché esatto
 - ridurre la frequenza degli arrivi al sistema – ad es., la frequenza di campionamento da un certo sensore
 - ignorare alcuni eventi dalla coda delle richieste – ad es., gestendo una coda degli eventi e limitandone la lunghezza
 - assegnare priorità agli eventi – per poter poi decidere di scartare degli eventi a bassa priorità quando le risorse iniziano a scarseggiare
- attenzione, non sempre è possibile o accettabile applicare queste tattiche



- Osservazioni

- ❑ Prima di andare avanti nell'illustrazione di altre tattiche, è utile fare alcune osservazioni
 - applicabilità delle tattiche
 - non sempre è possibile o accettabile applicare tutte le tattiche a disposizione
 - applicazione delle tattiche
 - l'applicazione di una tattica può avere **effetto** su uno o più elementi architettureali – e/o su una o più relazioni tra elementi architettureali – in una o più viste architettureali

Le tattiche sono OPZIONI, quindi sono tante ma non sono tutte applicabili

Tattica = TRASFORMAZIONE con effetto QUALITATIVO



Osservazioni

- ❑ Prima di andare avanti nell'illustrazione di altre tattiche, è utile fare alcune osservazioni
 - effetto *qualitativo* e *quantitativo* delle tattiche
 - l'applicazione di una tattica può portare a controllare un certo attributo di qualità
 - qui ci limitiamo a dare delle intuizioni circa l'effetto *qualitativo* delle tattiche
 - in alcuni casi è possibile (e utile) fare anche ragionamenti *quantitativi* sull'effetto dell'applicazione di una tattica
 - tattiche ed effetti collaterali – e compromessi
 - quando è necessario applicare una tattica? Se l'architettura corrente non soddisfa alcuni degli scenari prioritari per quanto riguarda la qualità



- Manage resources

②

- ❑ Tattiche per la gestione delle risorse
 - anche una gestione migliore delle risorse può portare a una riduzione del tempo di risposta
- ❑ **Increase resources** Aumentare o migliorare (core, memoria, reti, velocità di trasmissione, dischi ecc) ovviamente molto costosa
 - risorse computazionali addizionali o più veloci hanno il potenziale per ridurre il tempo di risposta
 - è la tattica più semplice – ma è anche la più costosa, e non sempre è efficace



Manage resources

②

- Tattiche per la gestione delle risorse basate sulla concorrenza

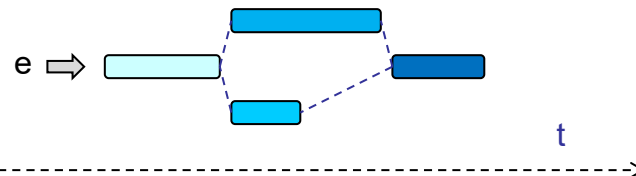
- **Introduce concurrency**

- suddividere l'elaborazione relativa alla gestione di un singolo evento in più compiti distinti, da svolgere in parallelo e in modo concorrente, può ridurre il tempo di risposta per la gestione dell'evento
- in questo caso la concorrenza è usata nella **gestione di un evento individuale** Da algoritmo sequenziale a concorrente, se le parti che vengono rese concorrenti sono ampie di possono suddividere in più rami per migliorare l'efficienza

elaborazione sequenziale di un evento



introduce concurrency



17

Prestazioni

Luca Cabibbo ASW



Manage resources

②

- Tattiche per la gestione delle risorse basate sulla concorrenza

- **Maintain multiple copies of computations**

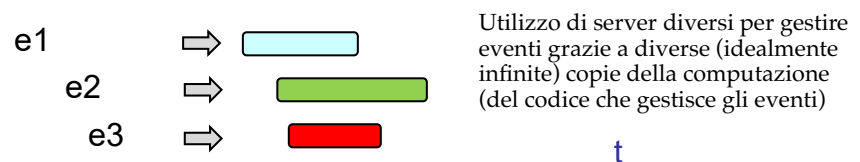
L'obiettivo non è migliorare il tempo di risposta per un SINGOLO evento ma considerare il fatto di dover gestire un FLUSSO di diversi eventi consecutivi

- replicare risorse computazionali per consentire l'elaborazione di più eventi distinti, da svolgere in modo concorrente
 - ad es., replicare un server (serve anche un load balancer)
- la replicazione delle risorse consente anche di ridurre la contesa
- in questo caso la concorrenza è usata per la **gestione di un insieme di eventi concorrenti** tra di loro

elaborazione sequenziale di più eventi



maintain multiple copies of computations



18

Prestazioni

Luca Cabibbo ASW



Manage resources

②


□ Tattiche per la gestione delle risorse basate sulla concorrenza

□ **Maintain multiple copies of computations**

- replicare risorse computazionali per consentire l'elaborazione di più eventi distinti, da svolgere in modo concorrente
 - ad es., replicare un server (serve anche un load balancer)
- la replicazione delle risorse consente anche di ridurre la contesa
 - Le conseguenze negative: ho bisogno di più risorse (es più server = costoso), l'implementazione (e modifica) di un algoritmo concorrente è più complessa, ci sono problemi di contesa su risorse che vanno gestiti tramite sincronizzazione
- in questo caso la concorrenza è usata per la gestione di un insieme di eventi concorrenti tra di loro

In questa ottica è un CONNETTORE

elaborazione sequenziale
di più eventi

e1 e2 e3 → 

maintain multiple copies
of computations

e1 e3 → 

(con un numero limitato di copie)

e2 → 

t

Caso reale: ci possono essere
attese in coda, che però
dovrebbero essere ridotte

Prestazioni

Luca Cabibbo ASW

19



Manage resources

②

□ Tattiche per la gestione delle risorse

□ **Maintain multiple copies of data**

- è anche possibile replicare dei dati, e consentirne l'accesso in modo concorrente
 - ad es., mediante meccanismi di *data replication* o di caching – in questo caso potrebbe servire anche un meccanismo per “sincronizzare” le varie copie dai dati replicati
- “replica” e “copia” non vanno sempre intesi in modo letterale – ma possono essere intesi anche come “dati ridondanti”, anche rappresentati in forme diverse

Si riduce il tempo di accesso in lettura ma ogni volta che scrivo devo scrivere su tutte le copie quindi i tempi in scrittura possono peggiorare.

Si può parlare anche di copie parziali (non complete, non esatte) (rivedi e cerca esempi)

Cache: richiedono meccanismi di validazione e ciò solleva dei problemi sull'utilizzo di questa tattica:

1- ogni volta che ho più copie degli stessi dati devo applicare tecniche di sincronizzazione di essi

2- le copie di dati potrebbero diventare obsolete e quindi devo capire con quale livello di obsolescenza posso accedere alle copie di dati

Prestazioni

Luca Cabibbo ASW

20



Maintain multiple copies...

- ❑ Come vedremo più avanti nel corso, le tattiche *Maintain multiple copies of computations* e *Maintain multiple copies of data* sono tattiche importanti non solo per le prestazioni, ma anche per la disponibilità e la scalabilità



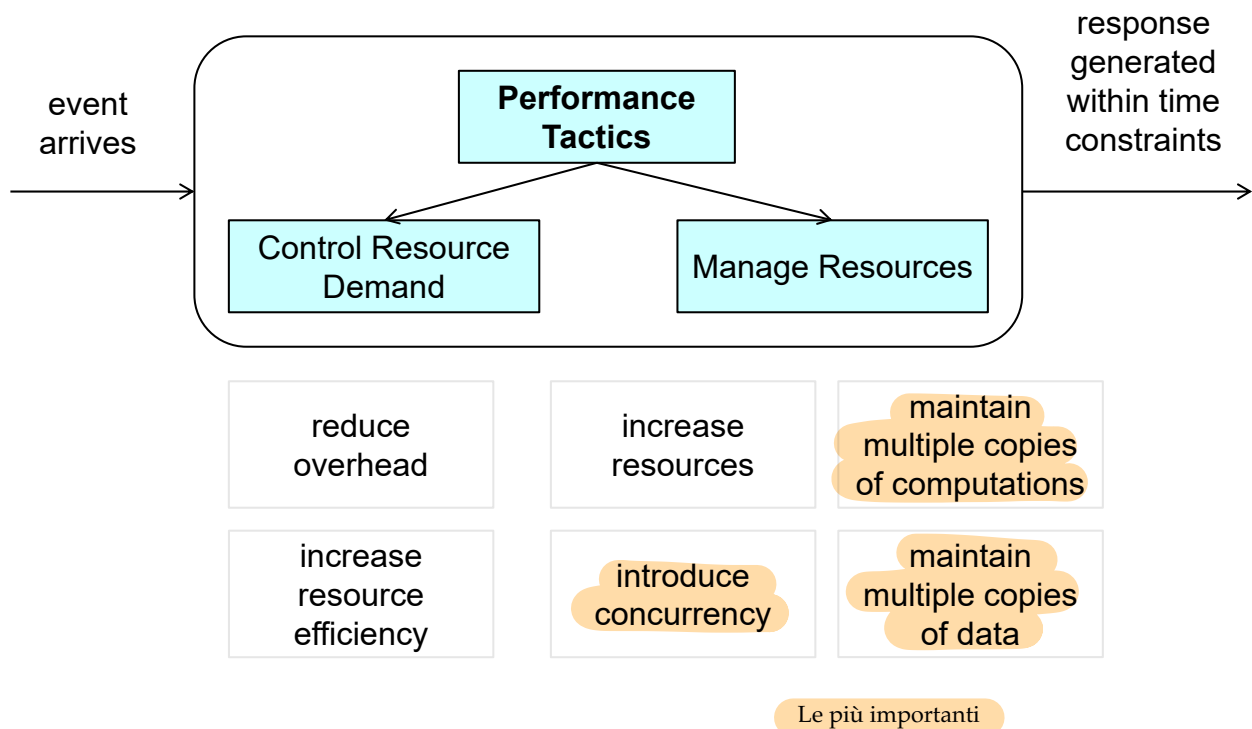
Manage resources



- ❑ Altre tattiche per la gestione delle risorse
 - schedulare l'uso delle risorse
 - limitare la dimensione delle code



Tattiche per le prestazioni



23

Prestazioni

Luca Cabibbo ASW



- Altre opzioni di progettazione per le prestazioni



- ❑ Ecco alcuni ulteriori suggerimenti, tattiche e opzioni di progettazione per le prestazioni – proposti nel contesto della prospettiva delle prestazioni e della scalabilità di [SSA]
 - ottimizza le elaborazioni ripetute – *Increase resource efficiency*
 - riduci la contesa tramite replicazione – *Maintain multiple copies of computations* e *Maintain multiple copies of data*
 - partiziona e parallelizza – *Introduce concurrency*
 - usa elaborazioni asincrone
 - riduci le necessità di sincronizzazione
 - ad es., favorisci transazioni brevi e rilassa i requisiti di consistenza transazionale
 - ridistribuisci l'elaborazione nel tempo
 - ad es., posticipa l'esecuzione di attività meno importanti a momenti di carico minore per il sistema

24

Prestazioni

Luca Cabibbo ASW



* Discussione

- ❑ Le prestazioni riguardano la capacità di un sistema di soddisfare dei requisiti temporali
 - abbiamo discusso alcune attività legate alla progettazione per le prestazioni
 - abbiamo presentato alcune tattiche architetturali e opzioni di progettazione per le prestazioni
 - alcune di queste tattiche e opzioni di progettazione per le prestazioni sono propedeutiche alla progettazione per la scalabilità
 - la presentazione è stata qualitativa e informale
- ❑ Abbiamo anche discusso ed esemplificato i seguenti aspetti
 - ciascuna tattica è una decisione di progetto per controllare un certo attributo di qualità
 - l'applicazione di una tattica consiste in una trasformazione dell'architettura