Ingegneria del Software
15. Stili e QoS

Dipartimento di Informatica
Università di Pisa
A.A. 2014/15

scale up, scale out

- Application scalability can be defined as the ability to increase the application throughput in proportion to the hardware that is being used to host the application. In other words, if an application is able to handle 100 users on a single CPU hardware, then the application should on be able to handle 200 users when the number of processors are doubled

- Vertical scalability is adding more memory and CPUs to a single box, or scaling up. Vertical scalability or scaling up is well suited for database tier. DBs have the following needs

  - large shared memory space
  - many dependent threads
  - Tightly-coupled internal interconnect

- Horizontal scalability is adding more boxes of similar memory and CPU, or scaling out. Scale out is ideal for web-tier, and has some of the following characteristics

  - small non-shared memory space
  - many independent threads
  - loosely-coupled external interconnect [important point]
  - possibly many OSs

- “To scale horizontally (or scale out) means to add more nodes to a system, such as adding a new computer to a distributed software application. […] To scale vertically (or scale up) means to add resources to a single node in a system, typically involving the addition of CPUs or memory to a single computer.” [wiki]

n-tier

Disponibilità	I server di ogni tier (ordine, fila) possono essere replicati, quindi se uno fallisce c’è solo una minor QoS
Fault tolerance	Se un cliente sta comunicando con un server che fallisce, la maggior parte dei server reindirizza la richiesta a un server replicato in modo trasparente all’utente
Modificabilità	Il disaccoppiamento e la coesione tipici di questa architettura favoriscono la modificabilità
Performance (efficienza)	Performance ok, ma da tenere d’occhio: numero di thread paralleli su ogni server, velocità delle comunicazioni tra server, volume dati scambiato
Scalabilità	Basta replicare i server in ogni tier. Unico collo di bottiglia la base di dati

publish-subscribe

Disponibilità	Si possono creare cluster di dispatcher
Fault tolerance	Si cerca un dispatcher replica
Modificabilità	Si possono aggiungere publisher e subscriber a piacere. Unica attenzione al formato dei messaggi
Performance (efficienza)	Ok, ma compromesso tra velocità e altri requisiti tipo affidabilità e/o sicurezza
Scalabilità	Con un cluster di dispatcher si può gestire un volume molto elevato di messaggi

comunicazione asincrona

Disponibilità	Basta replicare le code
Fault tolerance	Se una coda fallisce, si cerca una replica
Modificabilità	Unico vincolo il formato dei messaggi
Performance (efficienza)	Si possono spedire migliaia di messaggi al secondo. Compromesso tra affidabilità/sicurezza e performance
Scalabilità	Le code possono essere ospitate presso origine e destinazione della comunicazione, ma anche da cluster grandi a piacere di server

coordinatore di processi

- Il coordinatore di processi conosce la sequenza di passi necessari per realizzare un processo aziendale (PA)
- Riceve la richiesta, chiama i server secondo l'ordine prefissato, fornisce una risposta
- Normalmente usato per realizzare processi aziendali complessi
- Disaccoppiamento: i server non conoscono il loro ruolo nel PA complessivo né l'ordine dei passi del processo, e semplicemente definiscono un insieme di servizi
- Comunicazione flessibile: sincrona o asincrona

coordinatore di processi

Disponibilità	Il coordinatore è un punto critico: deve essere replicato se si vuole garantire disponibilità
Fault tolerance	Occorre specificare compensazione. Occorre ridirigere su un coordinatore replica
Modificabilità	Posso modificare liberamente i server purché non cambino le funzionalità esportate
Performance (efficienza)	Il coordinatore deve essere in grado di servire più richieste concorrenti. La performance del processo è limitata dal server più lento. Se non tutti i server sono necessari, si usa un time-out
Scalabilità	Replicando il coordinatore. Scala sia up che out

p2p

Disponibilità	Dipende dal numero di nodi in rete, ma si assume di si
Fault tolerance	Gratis
Modificabilità	Si, se dell'architettura interessa solo la parte di comunicazione
Performance (efficienza)	Dipende dal numero di nodi connessi, dalla rete, dagli algoritmi. Per esempio, BitTorrent ottimizza scaricando per primo il file/pezzo più raro
Scalabilità	Gratis

pipe & filter

Disponibilità	Avendo “pezzi di ricambio” (componenti e possibilità di connetterle) sufficienti a ricostruire una catena
Fault tolerance	Occorre riparare una catena interrotta usando componenti replica
Modificabilità	Si, se le modifiche interessano una o comunque poche componenti
Performance (efficienza)	Dipende dalla capacità del canale di comunicazione e dalla performance del filtro più lento
Scalabilità	Ok anche scale out