**Tipologia accessi:**

Reference data: accesso principalmente in lettura, scrittura molto rara. Es catalogo prodotti aggiornato una volta al giorno (o al mese)

Transactional Data: Cambiano con la stessa frequenza con cui vengono letti.

**Accessi scalablilita/performances:**

1. Accesso Lettura:  
   Ridondanza migliora performances accessi lettura:  
   Accesso in lettura piu veloce quanti meno nodi bisogna attraversare per ottenere il dato richiesto. Se il dato non e’ su tutti i servers, per raggiungerlo ogni server fara’ un forward al prossimo nodo fino a trovare quello giusto.
2. Transazionalita’:  
   Numero nodi da aggiornare e sincronicita’ updates influenzano le performances.

**Tipologie:**

Idee di performances con Ncache:

**Mirrored Cache:**Reference data: Good. Max 30000 reads/sec (1k size).  
Transactional data: Good. Max 20000 writes/sec (1k size).

**Replicated Cache:**Reference data: Good. 30000 reads/sec per server. Grow linearly by adding servers.  
Transactional data: Not so good. Max 2500 writes/sec. Drops if you add 3rd server.

**Partitioned Cache (no replication):**Reference data: Good. 30000 reads/sec per server. Grow linearly by adding servers.  
Transactional data: Good. 25000 writes/sec per server. Grow linearly by adding servers.

**PartitionReplica**Cache (with replication):  
Reference data: Good. 30000 reads/sec per server. Grow linearly by adding servers.  
Transactional data: Good. 20000 writes/sec per server. Grow linearly by adding servers.

**Mirroring:**

Active/Passive cluster. (Master/Slave)  
  
Un solo server attivo. Tutte le richieste lettura/scrittura vengono fatte su un’unico nodo.  
L’altro riceve in modo asincrono una copia dei dati e viene usato solo per recovery e failover.  
Problemi:  
- Manca di scalabilita’ (non ha senso aggiungere altri nodi passivi)  
- Manca di load balancing

**Replicated Cache:**

N Nodi contenenti ciascuno l’intera cache.  
Usati sia in lettura che in scrittura.  
Ogni update viene propagato in modo sincrono a tutti gli altri nodi.  
Ogni update e’ associato ad un sequence, per cui la transazione si completa se tutti i nodi completano una certa sequence. Se ricevono un update con una sequence maggiore, aspettano prima di completare quella precedente.  
Performances:  
Transactional: Pessime performances al crescere del numero dei nodi.   
Reference: Linear scalability.  
Molto veloce perche’ tutti i dati sono replicati ovunque, quindi ogni cache e’ in grado di rispondere senza forward alle richieste dei clients.

**Partitioned Cache (no replication):**La cache e’ suddivisa (senza replica) in un numero di nodi (partizioni).  
Ogni dato e’ scritto/letto in base ad una funzione di hash che lo assegna alla partizione data dalla formula: hashEntry MOD numeroPartizioni.  
In tal modo gli UPDATE avvengono sempre su un unico nodo.  
I GET vengono eseguiti sempre sull unico nodo (visto che e’ prevedibile la sua collocazione)

Vantaggi: Scalabilita’ lineare per GET e UPDATE.  
Svantaggi: Manca supporto per failover.

**Partioned Cache REPLICA:**Uguale a Partitioned ma gli oggetti sono replicati su 2 (o piu) nodi.  
Ogni nodo funziona da nodo attivo per i suoi oggetti, ed in piu contiene la replica (passiva) di oggetti gestiti da altri nodi.  
Vantaggi:uguali a Partitioned e Replica: Scalabilita lineare, failover, recovery.

Synch e asynch replica write behind.   
Se sincrono piu sicuro (in caso il nodo attivo fallisce) ma piu lento.  
Gli updates asincroni vengono immessi in una coda e processati in background da un thread separato.  
Asincrono piu veloce, ma a rischio di perdita di dati nel caso di crash del nodo attivo che ha ricevuto l update.