**Caching e scalability:**

Scalability is another benefit of object caching. Since cached data is accessed across multiple sessions and Web applications, object caching can become a big part of a scalable Web application's design. Cached data objects directly addresses latency, reduces operating costs, and eliminates bottlenecks.  
  
In a distributed server environment, the main shortfall of object caching is

1. not capable of offering linear scalability.
2. The other difficulty is synchronization complexity. The complexity increases because consistency between the cached data's state and the data source must be ensured. Otherwise, the cached data can fall out of sync with the actual data, which leads to data inaccuracies.

Linear scalability (memoria sommata)

Distributed cache technology using data grid solves both shortfalls. Most important is he linear scalability through data grid partitioning. In the data grid, the data is spread out over all the servers in such a way that no two servers are responsible for the same piece of cached data. This means that the size of the cache and the processing power associated with the management of the cache can grow linearly with the size of the cluster.

Navigazione:

When there is a request for cached data, the response can be accomplished with a "single hop" to another server, if the data object is not found in local cache. This means when more servers are adding to the grid, the performance of the response does not decrease.

**Features Caches:**

In additional these, I summarize cache related features into three categories:  
  
**Transaction Related Features:**(1) Read-Through Caching: automatically delegate to the Cache and ask it to load object from the underlying data source.  
  
(2) Write-Through Caching: when the application updates a piece of data in the cache, the transaction will not complete until the process has gone through the cache and successfully stored the data to the underlying data source.  
  
(3) Write-Behind Caching: modified cache entries are asynchronously written to the data source after a configurable delay.

**Data Set Related Features:**

(1) Near/local Cache: an application local vs in-line cache when used in a client/server topology.  
  
(2) Sparse and Complete Cache:   
A sparse cache only keeps a subset of the data and can be populated lazily, on-demand.   
A complete cache contains all of the required data.   
A complete cache is preloaded with data prior to applications' using it. Once loaded, it can be treated similarly to a database.

**Data Loading Related Features:**

(1) Refresh-Ahead Caching: allows a developer to configure a cache to automatically and asynchronously reload (refresh) any recently accessed cache entry from the cache loader prior to its expiration.  
  
(2) Periodic refresh: The cache can be automatically invalidated or updated periodically

**CAMBIAMENTI ARCHITETTURALI:**

It brings an unique cache layer to all servers in the grid. Following are the changes we can foresee to use this new technology.  
  
Cache vs HttpSession  
As most of the objects can be cached, there are no needs to design application particular cache. The usage of cache can be simpler as well. For example, for web application, developers tend to cache some session related objects into session object for caching. All these can be simplified to use the unique cache. And all these cached objects will be automatically distributed to other server on the grid.  
  
Unico livello (Business-Cache)  
An unique cache can also be implemented as either unique data layer or business object layer. The unique business objects in both runtime and model is a true business service in local memory.   
There is no need to design transfer object, client side business object and server side facada pattern and etc. All business object layer can be accomplished by design one class model and its runtime objects are distributed automatically to all servers’ local memory.

2.2.3 The Important Timing of Accessing the Data Source  
Under this new technology, the timing to commit the transaction to the data source is especially important in the application, as this could significantly impacts the overall application performance. The timing of loading the large volume data from data source will also affects the application performance as well.   
In the distributed cache technologies, all these can be configurable and adjustable. In fact, these have been the key factors in existing application as well. Careful testing and measure still required to fine tune the application.  
  
  
Vantaggi:

1) The most important is the linear scalability provided by this technology. As a result, the overall application performance increases as more servers are putted into grid.  
  
2) It will be not necessary to design the application as stateless in order to leverage the simple clustering technology.   
  
This could be another key point to adopt distributed cache technology. As all objects and status are saved in cached memory, and can be distributed to other server, simple loading balance technology like DNS-base approach or TCP/IP-level loading balance can be leveraged. Application level loading balance is fully covered by this new technology.  
  
3) This new technology also enhance clustering in several other areas like high availability, reduced single point failure and traffic bottle neck and  
4) increased the low-latency high performance by doing in memory persistence instead of keeping the persistence information into file and database.