Caching genellikle verileri ve nesneleri ram’e saklayıp bir dış veri kaynağının okunması gereken toplam sayıyı düşürerek, very tabanı güdümlü web sitelerini hızlandırmak amaçıyla kullanılır.

Memcached

* Sistem bir sunucu-istemci mimarisi kullanmaktadır. Sunucular bir anahtar-değer ilişkisel dizisi (associative array) tutar; istemciler de bu diziyi doldurur ve sorgularlar. **Anahtarlar** en fazla 250 byte, değerler de en fazla 1 mega byte boyutunda olabilir.
* Sunucular değerleri RAM’de tutarlar; eğer sunucu [RAM’i](http://www.bilgisayarkavramlari.com/2008/11/07/rastgele-erisilebilir-bellek-random-access-memory-ram/) dolarsa, sunucu en eski **verileri atar**. Dolayısıyla istemciler Memcached’i geçici bir ön bellek olarak düşünmek zorundadır; bir kez saklanan verinin ne zaman lazım olursa olsun sunucuda bulunacağını düşünmemelidir.
* Memcached kopyalamayı desteklememektedir çünkü veriyi kopyalamak, sahip olunan alanın yarıya inmesi demektir. Veri kopyalanmaz ise kopyalanana göre 2 kat daha fazla alana sahip olunur. Bu durumda, yani 2 kat fazla alanda, daha fazla veri saklanabilir. Bunun dezavantajlı durumu düşünülürse, örneğin 10 sunucu varsa ve 1’i çökerse yüzde 10’luk bir kayıp oranı oluşur. Eğer kayıpsız bir ortam isteniyorsa, Membase veya diğer noSQL veri tabanları kullanılabilir.
* Memcached veri kalıcılığını desteklememektedir
* Memcached kimlik doğrulamayı desteklememektedir.. Network üzerinden erişimi olan bütün istemciler “atama” (set) ve “alma” (get) işlemlerini yapabilirler. Üzerinde memcached çalışan sunucuların güvenlik duvarı ayarlarının dikkatli yapılması gerekir. [6] Memcached, SASL (Simple Authentication and Security Layer) protokolünün kurulmasını desteklemektedir ve bu sayede güvensiz bir ortamda güvenlik sağlanabilir.

Redis

* Redis’in diğer NoSQL bilgi bankalarından en önemli farklarından birisi, yalnızca String’ler ile sınırlı olmamasıdır. String’lere ek olarak Redis, aşağıdaki veri tiplerini de desteklemektedir:
  + [Listeler (List)](http://www.bilgisayarkavramlari.com/2007/05/03/linked-list-linkli-liste-veya-bagli-liste/)
  + Setler (Sets)
  + Sıralı Setler (Sorted Sets)
  + Komut Çizelgeleri (Hash Tables)
  + Atomik İşlemler (Atomic Transactions)
* Redis tek birim olarak maksimum 512 MB veriyi saklayabilmektedir.
* Redis, yalnızca master-slave kopyalamasını desteklemektedir. Herhangi bir master Redis sunucusu, kendinde bulunan veri güncellendiği anda master sunucuya bağlı olan N kadar slave sunucuda da güncelleme yapar. Bir slave, diğer slave’ler için master durumunda olabilir. Böylece bir çizge (graph) yapısı oluşturulabilmektedir.
* Redis aynı zamanda yayınla/abone ol (publish / subscribe) özelliğini de tam olarak desteklemektedir. Bu sayede bir slave’in istemcisi bir kanala abone olup, master’a yayınlanan tüm mesajları alabilir.
* Redis normalde tüm veri setlerini RAM’de tutmaktadır. Kalıcılığı sağlamak için ise 2 yöntem kullanılmaktadır. “Snapshotting” adı verilen birinci yöntemde, belirli zamanlarla snapshot’lar alınır ve bunlar diskte saklanır. Diğer yöntemde ise RAM’de tutulan veri setleri diskte de tutulur. İstenirse bu iki yöntem de devre dışı bırakılabilir. Bir başka seçenek de master yerine slave üzerinden kalıcılığı sağlayarak master üzerindeki yükü hafifletmektir.
* Redis, erişim kontrolü (Access Control) protokolünü implemente etmezken, küçük bir kimlik doğrulama katmanı sağlamaktadır. Kimlik doğrulama katmanı aktif hale getirildiğinde Redis, kimlik doğrulamasından geçemeyen hiçbir istemciyi kabul etmez
* Redis, veri şifrelemesini desteklememektedir. Bu yüzden, sadece güvenli istemcilerin veya güvenilen tarafların Redis’e erişimini sağlamak için, SSL gibi ek güvenlik katmanları implemente edilmelidir.
* Genel olarak denilebilir ki Redis, maksimum güvenlik için değil, maksimum performans ve basitlik üzerine tasarlanmıştır.

Evet Redis çok hızlı peki neden bu kadar hızlı diye sorarsanız o zaman bunu bir örnekle cevaplamak sanırım daha açıklayıcı olur. Bu durumda size **Pipelining**özelliğinden bahsedeceğiz; Diyelim ki cafede garson olarak çalışıyorsunuz ve kalabalık bir arkadaş grubu gelip masaya oturdu hep beraber sipariş verdiler 10 çay 5 kola 8 soda 3 pasta derken düğün menüsü çıktı, her bir müşterinin siparişi için tekrar tekrar mutfağa gitmeniz epey bir zaman ve performans kaybı olacağından tüm siparişleri büyük bir servis masasına koyup götürdükten sonra mutfağa tek gidişle tüm siparişleri almış ve masalara dağıtmış olursunuz işte Redis tam olarak bunu yapıyor **Pipelining**özelliği ile istediğiniz verilerin tamamını tek bir masaya(Pipe) koyup tek seferde size getirdiği için aynı işlem için defalarca getir götür yapmadan hızlıca işlemlerinizi gerçekleştiriyor aynı zamanda performans açısından da çok iyi bir avantaj sağlıyor.

Ama gelin görün ki Redis de diğer NoSQL veri tabanı sistemleri gibi karmaşık sorgulara cevap vermekten kaçınıyor yani relation dblerde bana pazar günü şu saatler arasında kola isteyen müşterilerimin bana ne kadar kazandırdığını getir bana derseniz buna cevap alamazsınız,

Bana pazar günü siparişlerini getir.

Bana şu saatler arasındaki siparişleri getir.

Bana kola olan siparişleri getir.

Bana kazancı getir.

şeklinde teker teker talepte bulunmak gerekiyor.

Hazelcast

* Hazelcast mimarisi P2P’dir, yani master-slave mimarisi yoktur. Dolayısıyla, tek kaynaklı çökme (single point of failure) yaşanmaz. Çok yöne yayını (multicast) veya TCP/IP protokolünü kullanarak dinamik keşfi (dynamic discovery) destekler.
* Hazelcast, eğer
  + Veri birçok sunucu üzerinde dağıtılacaksa (örneğin web oturumu paylaşımı)
  + Veri ön belleğe alınacaksa (dağıtık ön bellek)
  + Uygulama kümelenecekse
  + Sunucular arasında güvenli iletişim sağlanacaksa
  + RAM’deki veri bölünecekse (partitioning)
  + İş yükü birçok sunucuya dağıtılacaksa
  + Paralel işlemenin avantajından faydalanılacaksa ve
  + Veri kaybı olmaksızın veri yönetimi uygulanacaksa
* doğru seçim olacaktır.
* Hazelcast sadece String’ler ile sınırlı olmamakla beraber aşağıdaki veri yapılarını desteklemektedir:
  + [Listeler (List)](http://www.bilgisayarkavramlari.com/2007/05/03/linked-list-linkli-liste-veya-bagli-liste/)
  + Setler (Sets)
  + Sıralar (Queue)
  + Harita (Map)
* Hazelcast için tutabileceği nesne boyutu sınırlaması yoktur; ancak, unutulmamalıdır ki boyu artıkça gecikme (latency) de artacaktır.
* Hazelcast verileri her bir küme üyesine dağıtır. Bir küme üyesi çöktüğü zaman veri kaybı yaşamamk adına Hazelcast belli miktarda yedekleme yapılmasına izin verir. Bu sayede bir küme üyesi üzerinde bulunan veri bir başka üyeye kopyalanmış olur. Kopyalama işlemi asenkron olarak yapılmaktadır ve kaç adet kopyalama yapılacağı yönetici tarafından belirlenebilir. Varsayılan kopyala sayısı 1’dir. Eğer sayı 1’den fazla ise her bir küme üyesi hem kendi hem de bir başka üyenin verisini saklar.
* Hazelcast direkt olarak kalıcılık sağlamamaktadır. Ancak, bu işi uygulama tarafında sağlayan kodla uyumlu çalışarak, uygulama tarafında kalıcılığı sağlayabilmektedir.