gridgain.com

From Cache To In-Memory Data Grids

Nikita Ivanov November 19, 2013

4-5 minutes

I would like to clarify definitions for the following technologies:

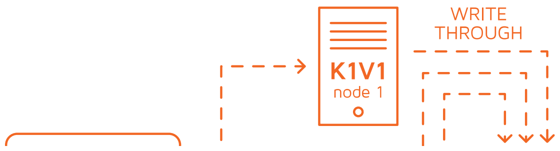
- In-Memory Distributed Cache
- In-Memory Data Grid

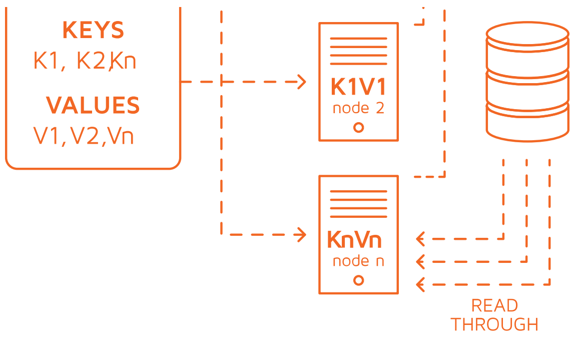
These terms are, surprisingly, often used interchangeably and yet technically and historically they represent very different products and serve different, sometimes very different, use cases.

It's also important to note that there's no specifications or industry standards on what cache, or data grid should be (unlike java application servers and JEE, for example). There was and still is an attempt to standardize caching via JSR107 and JSR 347 but it has been years (almost a decade for JSR 107) in the making and they are both hopelessly outdated by now (I'm on the expert group for JSR 107).

Tricycle vs. Motorcycle

First of all, let me clarify that I am discussing caches and data grids in the context of *in-memory, distributed architectures.* Traditional disk-based databases and non-distributed in-memory caches or databases are out of scope for this article.





Chronologically, caches and data grids were developed in that order from simple caching to more complex data grids. The first distributed caches appeared in the late 1990s, and data grids emerged around 2003-2005.

Both of these technologies are enjoying a significant boost in interest in the last couple years thanks to explosive growth in-memory computing in general fueled by 30% YoY price reduction for DRAM and cheaper Flash storage.

Despite the fact that I believe that distributed caching is rapidly going away, I still think it's important to place it in its proper historical and technical context along with data grids and databases.

In-Memory Distributed Caching

The primary use case for caching is to keep frequently accessed data in process memory to avoid constantly fetching this data from disk, which leads to the High Availability (HA) of that data to the application running in that process space (hence, "in-memory" caching).

Most of the caches were built as distributed in-memory key/value stores that supported a simple set of 'put' and 'get' operations and optionally some sort of read-through and write-through behavior for writing and reading values to and from underlying disk-based storage such as an RDBMS. Depending on the product, additional features like ACID transactions, eviction policies, replication vs.

partitioning, active backups, etc. also became available as the products matured.

These fundamental data management capabilities of distributed caches formed the foundation for the technologies that came later and were built on top of them such as **In-Memory Data Grids.**

In-Memory Data Grid

The feature of data grids that distinguishes them from distributed caches the most was their ability to support co-location of computations with data in a distributed context and consequently provided the ability to move computation to data. This capability was the key innovation that addressed the demands of rapidly growing data sets that made moving data to the application layer increasing impractical. Most of the data grids provide some basic capabilities to move the computations to the data.

Another uniquely new characteristic of in-memory data grids is the addition of distributed MPP processing based on standard SQL and/or MapReduce, that allows to effectively compute over data stored in-memory across the cluster.

Just as distributed caches were developed in response to a growing need for data HA, in-memory data grids were developed to respond to the growing **complexities** of data processing. It was no longer enough to have simple key-based access. Distributed SQL, complex indexing and MapReduce-based processing across TBs of data in-memory are necessary tools for today’s demanding data processing.

Adding distributed SQL and/or MapReduce type processing required a complete re-thinking of distributed caches, as focus has shifted from pure data management to hybrid data and compute management.

This new and very disruptive capability of in-memory data grids also marked the start of the in-memory computing revolution.