**Chapter 2. Architectural Overview**

JanusGraph is a graph database engine. JanusGraph itself is focused on compact graph serialization, rich graph data modeling, and efficient query execution. In addition, JanusGraph utilizes Hadoop for graph analytics and batch graph processing. JanusGraph implements robust, modular interfaces for data persistence, data indexing, and client access. JanusGraph’s modular architecture allows it to interoperate with a wide range of storage, index, and client technologies; it also eases the process of extending JanusGraph to support new ones.

Between JanusGraph and the disks sits one or more storage and indexing adapters. JanusGraph comes standard with the following adapters, but JanusGraph’s modular architecture supports third-party adapters.

* Data storage:
  + [Apache Cassandra](http://docs.janusgraph.org/latest/cassandra.html)
  + [Apache HBase](http://docs.janusgraph.org/latest/hbase.html)
  + [Oracle Berkeley DB Java Edition](http://docs.janusgraph.org/latest/bdb.html)
* Indices, which speed up and enable more complex queries:
  + [Elasticsearch](http://docs.janusgraph.org/latest/elasticsearch.html)
  + [Apache Solr](http://docs.janusgraph.org/latest/solr.html)
  + [Apache Lucene](http://docs.janusgraph.org/latest/lucene.html)

Broadly speaking, applications can interact with JanusGraph in two ways:

* Embed JanusGraph inside the application executing [Gremlin](http://tinkerpop.apache.org/docs/3.2.6/reference#graph-traversal-steps) queries directly against the graph within the same JVM. Query execution, JanusGraph’s caches, and transaction handling all happen in the same JVM as the application while data retrieval from the storage backend may be local or remote.
* Interact with a local or remote JanusGraph instance by submitting Gremlin queries to the server. JanusGraph natively supports the Gremlin Server component of the [Apache TinkerPop](http://tinkerpop.apache.org/) stack.

Chapter 4. Configuration

A JanusGraph graph database cluster consists of one or multiple JanusGraph instances. To open a JanusGraph instance, a configuration has to be provided which specifies how JanusGraph should be set up.

一个JanusGraph图数据库集群由一个或者多个JanusGraph 实例构成,创建一个JanusGraph实例需要提供一个配置文件指定这个图应该怎么样建立.

A JanusGraph configuration specifies which components JanusGraph should use, controls all operational aspects of a JanusGraph deployment, and provides a number of tuning options to get maximum performance from a JanusGraph cluster.

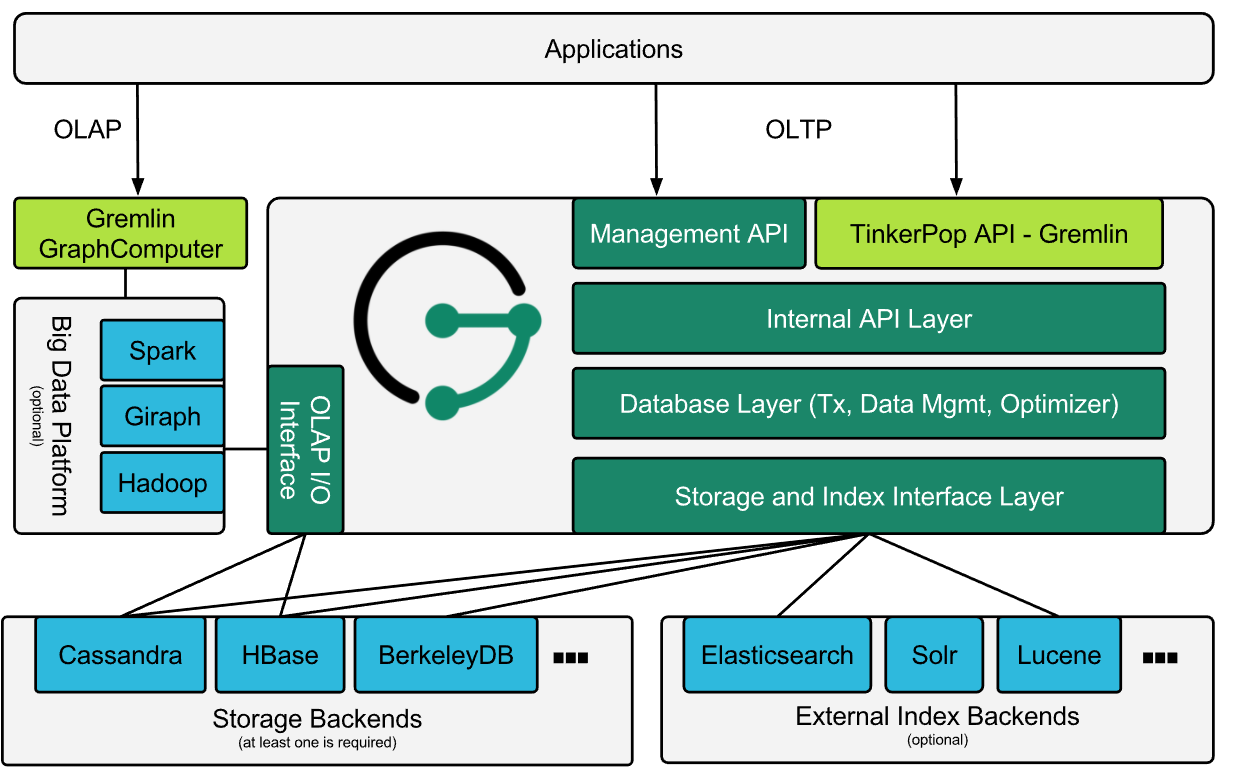
At a minimum, a JanusGraph configuration must define the persistence engine that JanusGraph should use as a storage backend. [Part III, “Storage Backends”](http://docs.janusgraph.org/latest/storage-backends.html) lists all supported persistence engines and how to configure them respectively. If advanced graph query support (e.g full-text search, geo search, or range queries) is required an additional indexing backend must be configured. See [Part IV, “Index Backends”](http://docs.janusgraph.org/latest/index-backends.html) for details. If query performance is a concern, then caching should be enabled. Cache configuration and tuning is described in [Chapter 11, *JanusGraph Cache*](http://docs.janusgraph.org/latest/caching.html).

JanusGraph

JanusGraph graph database implementation of the Blueprint's interface.这是一个接口, 定义了图数据库的接口实现,  
\* Use {**@link** JanusGraphFactory} to open and configure JanusGraph instances.

通过JanusGraphFactory来打开或者配置一个JanusGraph实例.

JanusGraphTransaction



单词

persistence engine 存储引擎

storage backend 后端存储

tuning 调优