Hierarchical Cluster Engine (HCE) project

v1.0 general description and architecture basics

IOIX Ukraine 2013

HCE project aim and main idea

This project became the successor of Associative Search Machine (ASM) full-text web search engine project that was developed from 2006 to 2012 by IOIX Ukraine.

The main idea of this new project – to implement the solution that can be used to: construct custom network mesh or distributed network cluster structure with several relations types between nodes, formalize the data flow processing goes from upper node level central source point to down nodes and backward, formalize the management requests handling from multiple source points, support native reducing of multiple nodes results (aggregation, duplicates elimination, sorting and so on), internally support powerful full-text search engine and data storage, provide transactions-less and transactional requests processing, support flexible run-time changes of cluster infrastructure, have many languages bindings for client-side integration APIs in one product build on C++ language...

HCE application area

As a network infrastructure and messages transport layer provider – the HCE can be used in any big-data solution that needs some custom network structure to build distributed high-performance easy scalable vertically and horizontally data processing or data-mining architecture.

As a native internally supported full text search engine interface provider – the HCE can be used in web or corporate network solutions that needs smoothly integrated with usage of natural target project specific languages, fast and powerful full text search and NOSQL distributed data storage. Now the Sphinx (c) search engine with extended data model internally supported.

AS a Distributed Remote Command Execution service provider – the HCE can be used for automation of administration of many host servers in ensemble mode for OS and services deployment, maintenance and support tasks.

Hierarchical Cluster as engine

- **Provides hierarchical cluster infrastructure – nodes connection schema, relations between nodes, roles of nodes, requests typification and data processing sequences algorithms, data sharding modes, and so on.**
- **Provides network transport layer for data of client application and administration management messages.**
- **Manages native supported integrated NOSQL data storage** (Sphinx (c) search index and Distributed Remote Command Execution)**.**
- **Collect, reduce and sort results of native and custom data processing.**
- **Ready to support transactional messages processing.**

HCE key functional principles

- *Free* network cluster structure architecture. Target applied project can construct specific schema of network relations that succeeds on hardware, load-balancing, file-over, fault-tolerance and another principles on the basis of one simple Lego-like engine.
- *Stable* based on ZMQ sockets reversed client-server networking protocol with connection heart-beating, automated restoration and messages buffering.
- *Easy* asynchronous connections handling with NUMA oriented architecture of messages handlers.
- *Unified* I/O messages based on json format.
- Ready to have client APIs bindings for many programmer languages covered by ZMQ library. Can be easily integrated and deployed.

HCE-node application

The heart and main component of the HCE project it is hce-node application. This application integrates complete set of base functionality to support network infrastructure, hierarchical cluster construction, full-text search system integration and so on, see “Hierarchical Cluster as engine” main points.

Implemented for Linux OS environment and distributed in form of source code tarball archive and Debian Linux binary package with dependencies packages.

Supports single instance configuration-less start or requires set of options that used to build correspondent network cluster architecture.

Supposes usage with client-side applications or integrated IPI.

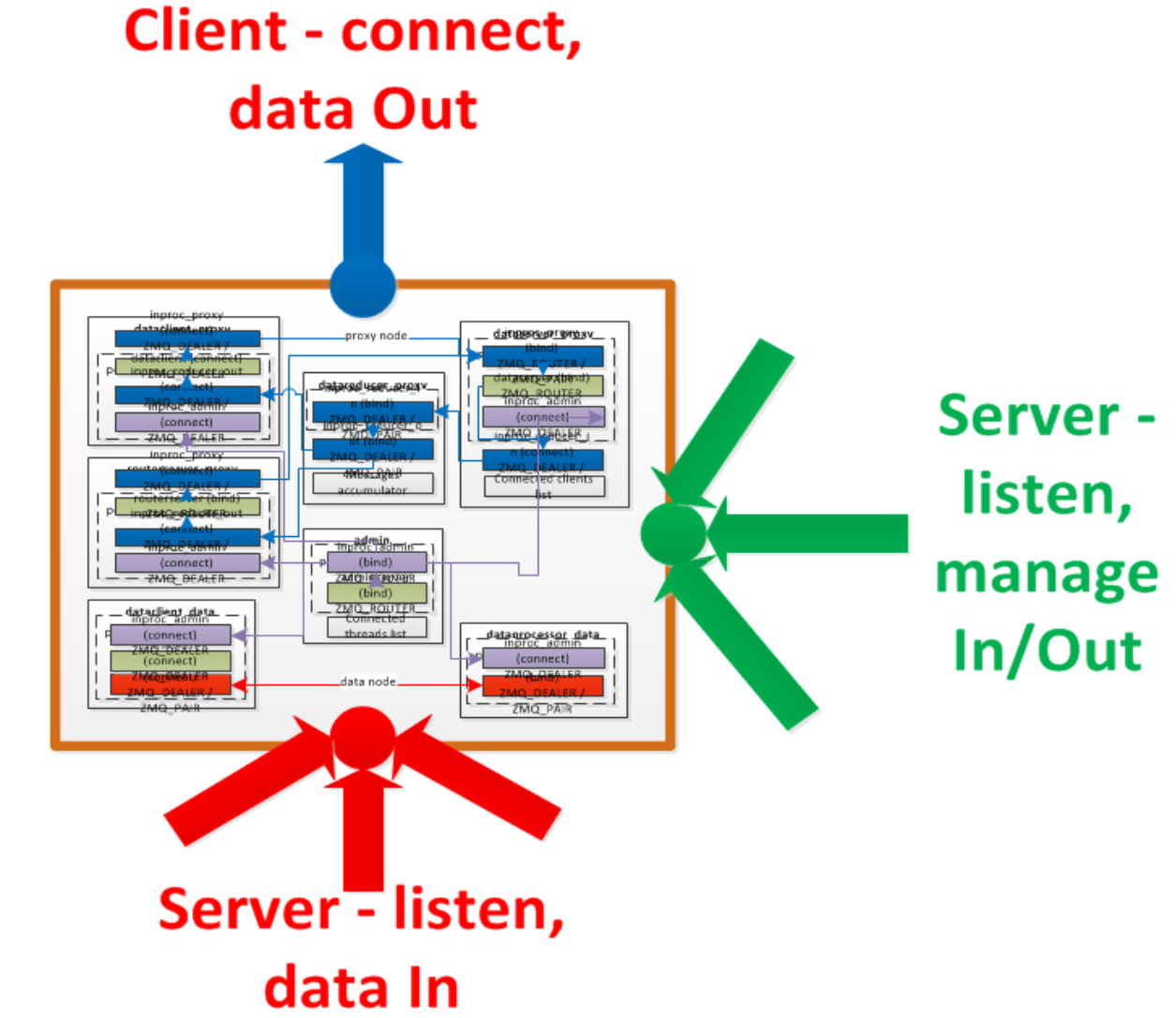
First implementation of client-side API and cli utilities bind on PHP.

Hce-node roles in the cluster structure

Internally HCE-node application contains seven basic handler threads. Each handler acts as special black-box messages processor/dispatcher and used in combination with other to work in one of five different roles of node:

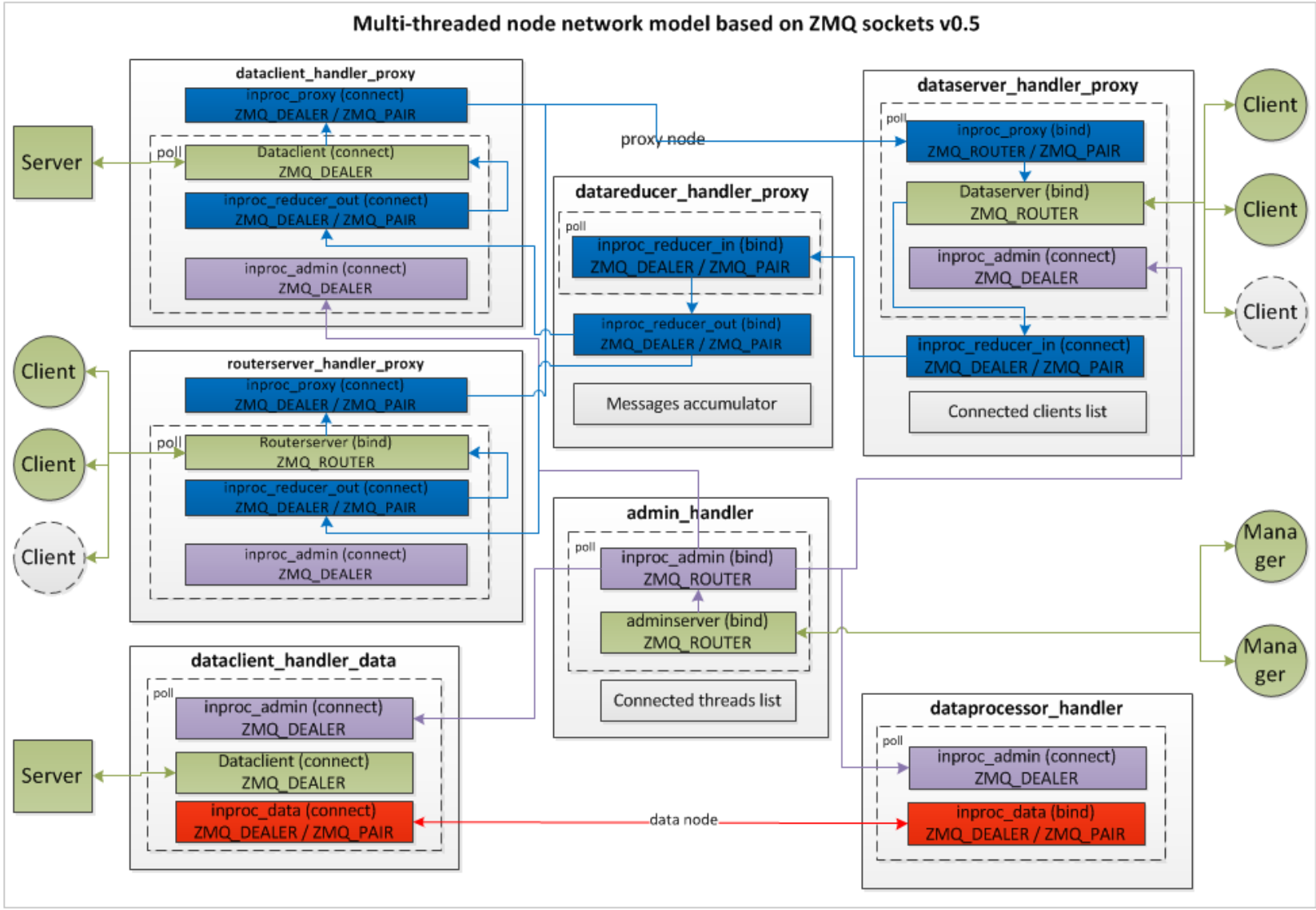
- **Router** – upper end-point of cluster hierarchy. Has three server-type connections. Handles client API, any kind of another node roles instances (typically, shard or replica managers) and admin connections.
- **Shard manager** – intermediate-point of cluster hierarchy. Routes messages between upper and down layers. Uses data **sharding** and messages multicast dispatching algorithms. Has two server-type and one client connections.
- **Replica manager** – the same as shard manager. Routes messages between upper and down layers uses data **balancing** and messages round-robin algorithms.
- **Replica** – down end-point of cluster hierarchy. Data node, interacts with data storage and/or process data with target algorithm(s), provides interface with fill-text search engine, target host for Distributed Remote Commands Execution. Has one server- and one client-side connections used for cluster infrastructure Also can to have several data storage-dependent connections.

hce-node typical connection points

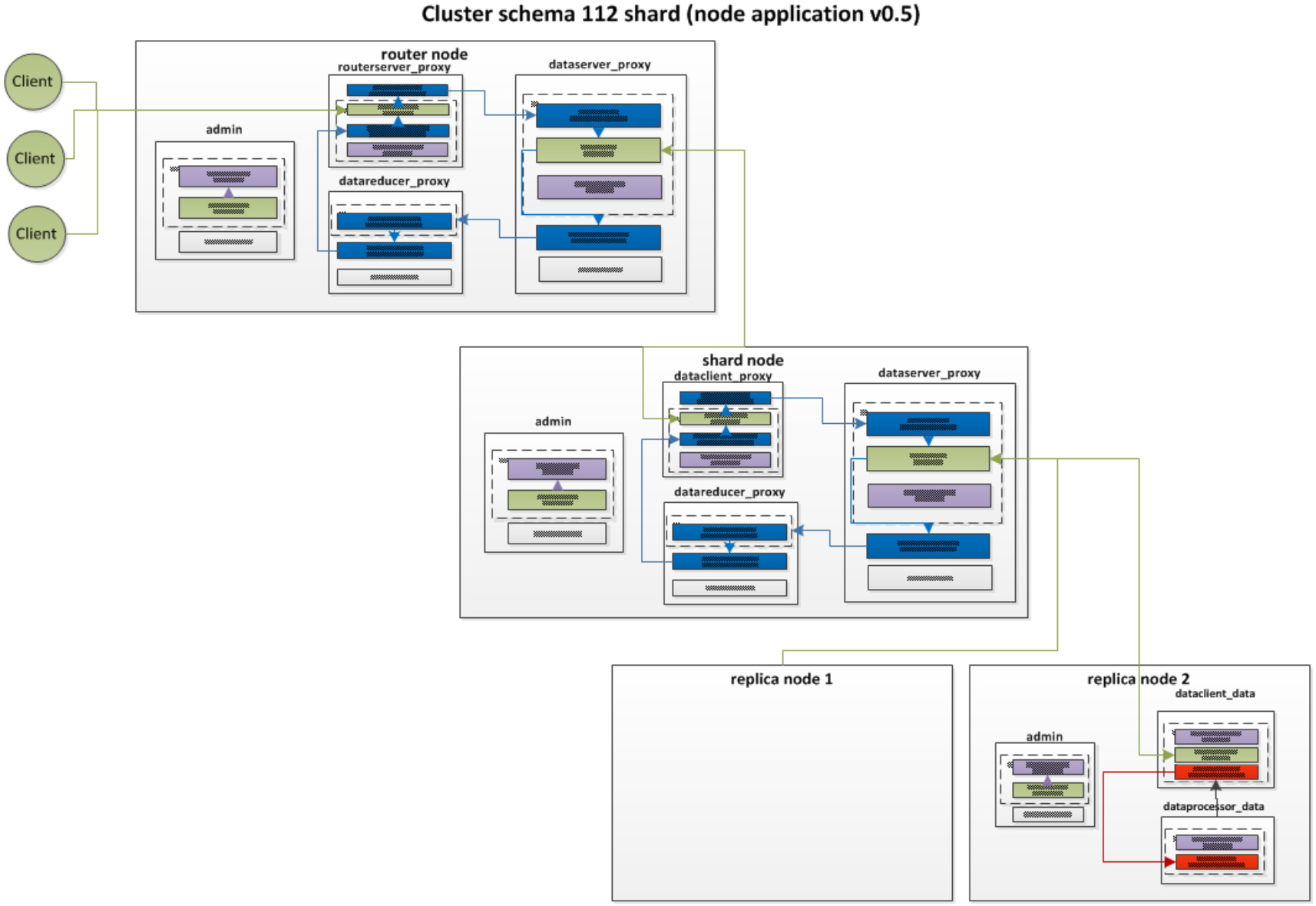


Hce-node internal architecture

seven handlers objects in relations

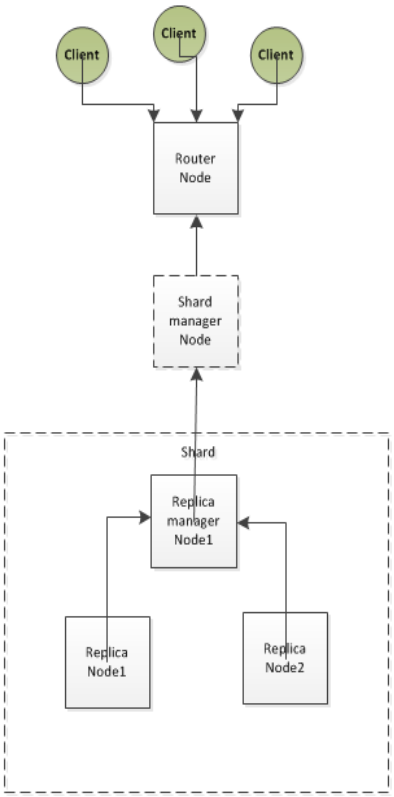


hce-node in cluster structure relations



Simple cluster structures comparison

Simple cluster structure schema 1-1-2 replica



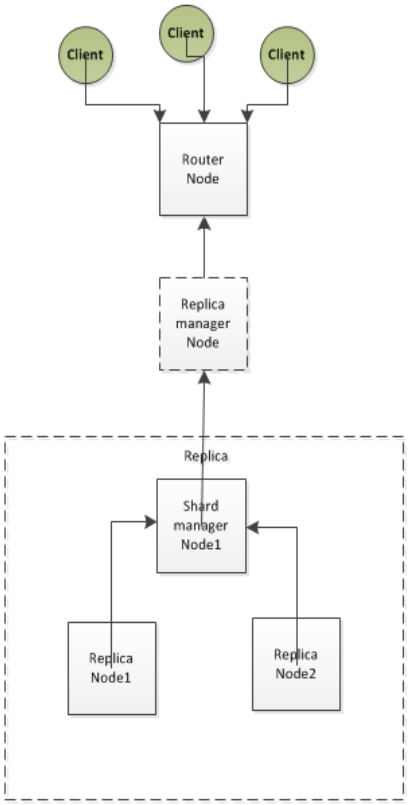
Final reducing, sort, merge messages from shard or replica manager, prepare response to client application

Reducing, sort, merge messages (search results, NLP results, DRCE, etc...) returned from replica manager or shard/replica node.

Reducing, sort, merge messages (search results, NLP results, etc...) returned from shard manager or replica node.

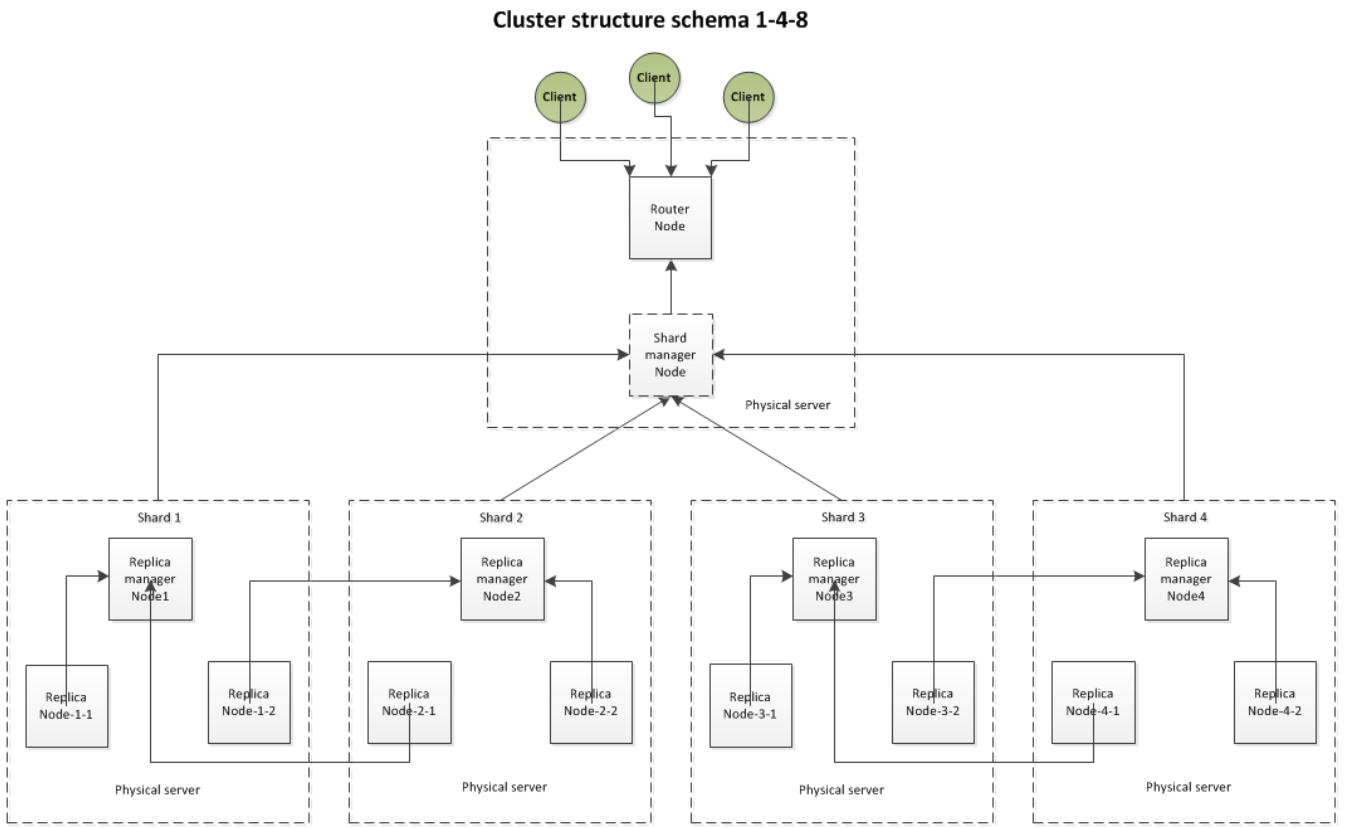
Request messages processing (Sphinx search, NLP task, etc...). Create response message with data processing results. Data storage (Sphinx index, NLP data files, etc). Algorithms storage (Scripting languages, Java or binary applications and libraries).

Simple cluster structure schema 1-1-2 shard



Hierarchical cluster structure example

This is just one example of four level cluster structure. This structure can be used to create four hosts distributed data model solution to split one huge documents storage on three physical host servers and utilize all CPU and I/O resources in effective way.



Each level uses different data distribution modes (shard or replica). Two replica nodes of each shard balances requests and have complete mirrored data. Shards are multicasted and requests results accumulated and reduced by shard manager node.

Extended Sphinx index support

- Smooth transparent aggregation of several different indexes and search algorithms in federated search.

- Native support by file operation the run-time indexes attach, copy, replace, swap, mirror, merge.

- Native support of indexes sharding and mirroring according different data schema.

- Extended full text search queries including full match, pattern match, set of fields match, logical operators, proximity limits, stop-words lists and many other provided by regular Sphinx search (c) engine.

- Filters for numeric fields including exact match and range comparisons.

- Multi-value numeric fields, bit-set and logical operations simulation in search query.

- Custom order including Sphinx full text search weight and custom weightier calculation algorithms definitions like SQL ORDER BY prioritized fields list or more complex logic.

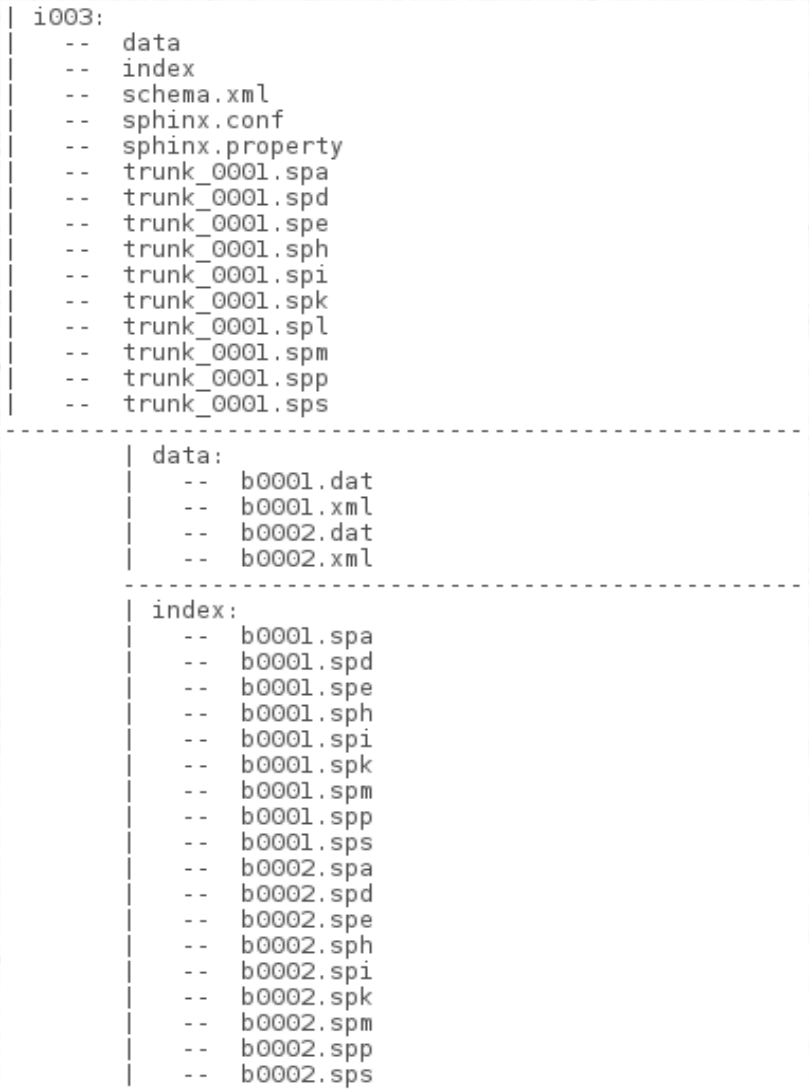
HCE Sphinx Index Extension - SIE

hce-node application provides solution named HCE Sphinx Index Extension (HCE SIE) that add to the Sphinx index nature more functionality and two high level structures:

Index – local FS folder contains complete set of sub-folders and files including native Sphinx indexes (trunk and branches), configuration files for searchd, source xml data files of branches, prepared xml data files of branches, schema xml file and other...
Index supports operations: create, copy, rename, delete, start, stop, rebuild, merge branches, delete document, connect, and other...

Branch – local FS files like source xml data file, prepared xml data file and native Sphinx Index files created from prepared xml data file. All branches of index uses the same schema and represents some part of ***index***. Set of branches can be merged in trunk that is used in search process. Branch supports operations: create, delete, rebuild, insert document, delete document and other...

HCE SIE folder structure



HCE technologies basics

- ZMQ (c) sockets for inter-process and in-process interactions.
- POCO C++ framework (c) for application architecture and algorithms including json and xml parsers, objects serialization, configuration settings and many other.
- Sphinx search (c) engine client native integration with extensions.
- Mutex-less multithread application architecture and messages handling.
- Strict C++ OOP design, including elements of 11 standard.

HCE distribution and deployment

- Open source distribution in source code of tarball archive.
- Binary Debian Linux native packages separated on HCE-node and utility packages as well as complete full meta-package. Dependencies libraries packaging and maintenance to complete cover dependencies and make installation smooth.
- Public repositories accessible for contributors community.
- Upgrade dependency packages and libraries as part of product.
- Box ready to start distributive with Demo Test Suit (DTS) with examples of simple cluster structure to start full text Sphinx-based search right after installation.
- Open documentation.

HCE project todo...

- Additions and extensions to support transparent configurable transactional model for all kind of actions.
- Extension of data node with native support of another index types and data storage like SQLite.
- Extensions with set of API and utility tools for creating typical cluster structures and sharding models.
- Extensions with set of API for general management and tracking of statistics, including web-UI for state visualization and visual management.
- Extensions for statistics and logging data analysis and visualization.