CSE-498-011-SP21

Generated by Doxygen 1.8.14

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

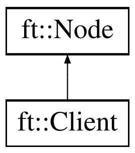
Chapter 4

Class Documentation

4.1 ft::Client Class Reference

```
#include <client.h>
```

Inheritance diagram for ft::Client:



Public Member Functions

- int initialize (std::string cfg_file)
- ft::Shard ∗ getShard (unsigned long long key)

Additional Inherited Members

4.1.1 Detailed Description

Client Node definition

4.1.2 Member Function Documentation

4.1.2.1 getShard()

```
ft::Shard* ft::Client::getShard (
            unsigned long long key )
```

Get the primary server storing a key

Parameters

key	- key whose primary server to search for

Returns

Server storing key

4.1.2.2 initialize()

```
int ft::Client::initialize (
            std::string cfg_file )  [virtual]
```

Initialize client

Returns

status. 0 on success, non-zero otherwise.

Reimplemented from ft::Node.

The documentation for this class was generated from the following file:

- /root/cjdambro/grad-school/CSE498/gits/fault-tolerance/include/faulttolerance/client.h

4.2 KVCGConfig Class Reference

```
#include <kvcg_config.h>
```

Public Member Functions

- int parse_json_file (std::string filename)
- std::size_t get_checksum ()
- std::vector< ft::Server * > getServerList ()
- cse498::ProviderType getProvider ()
- int getServerPort ()
- int getClientPort ()

4.2.1 Detailed Description

Class to parse config file and store data

4.2.2 Member Function Documentation

4.2.2.1 get_checksum()

```
std::size_t KVCGConfig::get_checksum ( )
```

Calculate and return a checksum for the configuration.

Returns

hash of config file

4.2.2.2 getClientPort()

```
int KVCGConfig::getClientPort ( )  [inline]
```

Get the port for server-client communication

Returns

int for client port

4.2.2.3 getProvider()

```
cse498::ProviderType KVCGConfig::getProvider ( )  [inline]
```

Get the provider from config.

Returns

ProviderType for servers.

4.2.2.4 getServerList()

```
std::vector<ft::Server*> KVCGConfig::getServerList ( )  [inline]
```

Get list of servers parsed from config.

Returns

vector of Servers

4.2.2.5 getServerPort()

```
int KVCGConfig::getServerPort ( )  [inline]
```

Get the port for server-to-server communication

Returns

int for server port

4.2.2.6 parse_json_file()

```
int KVCGConfig::parse_json_file (
            std::string filename )
```

Parse JSON input file

Parameters

filename	- name of JSON file to parse

Returns

status. 0 on success, non-zero otherwise.

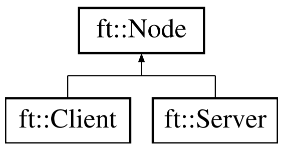
The documentation for this class was generated from the following file:

- /root/cjdambro/grad-school/CSE498/gits/fault-tolerance/include/faulttolerance/kvcg_config.h

4.3 ft::Node Class Reference

`#include <node.h>`

Inheritance diagram for ft::Node:



Public Member Functions

- virtual int initialize (std::string cfg_file)
- void setName (std::string n)
- void setAddr (std::string a)
- std::string getName ()
- std::string getAddr ()
- int getClientPort ()
- cse498::ProviderType getProvider ()
- bool **operator<** (const ft::Node &o) const

Public Attributes

- bool **alive** = true

Protected Attributes

- std::string **hostname**
- std::string **addr** = ""
- int **clientPort**
- cse498::ProviderType **provider**
- size_t **cksum**

4.3.1 Detailed Description

Base class for Server and Client

4.3.2 Member Function Documentation

4.3.2.1 getAddr()

```
std::string ft::Node::getAddr ( )  [inline]
```

Get the address of the node

Returns

Addres of node

4.3.2.2 getClientPort()

```
int ft::Node::getClientPort ( )  [inline]
```

Get the client port of the node

Returns

The client port of node

4.3.2.3 getName()

```
std::string ft::Node::getName ( ) [inline]
```

Get the name of the node

Returns

Name of the node

4.3.2.4 getProvider()

```
cse498::ProviderType ft::Node::getProvider ( ) [inline]
```

Get the provider of the node

Returns

The provider of node

4.3.2.5 initialize()

```
virtual int ft::Node::initialize (
            std::string cfg_file ) [inline], [virtual]
```

Initialize node data

Reimplemented in ft::Server, and ft::Client.

4.3.2.6 setAddr()

```
void ft::Node::setAddr (
            std::string a ) [inline]
```

Set the address of the node

Parameters

a	- Address to set for node

4.3.2.7 setName()

```
void ft::Node::setName (
             std::string n )  [inline]
```

Set the name of the node

Parameters

n	- Name to set for node

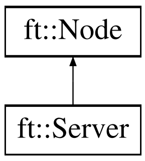
The documentation for this class was generated from the following file:

- /root/cjdambro/grad-school/CSE498/gits/fault-tolerance/include/faulttolerance/node.h

4.4 ft::Server Class Reference

```
#include <server.h>
```

Inheritance diagram for ft::Server:



Public Member Functions

- **Server** (const std::function< void(std::vector< RequestWrapper< unsigned long long, data_t *>>)> &commitFn)
- Server & **operator=** (const ft::Server &&src)
- int initialize (std::string cfg_file)
- void shutdownServer ()
- void printServer (const LogLevel lvl)
- std::vector< std::pair< unsigned long long, unsigned long long > > getPrimaryKeys ()
- bool addKeyRange (std::pair< unsigned long long, unsigned long long > keyRange)
- bool addPrimaryServer (ft::Server *s)
- std::vector< ft::Server * > getBackupServers ()
- bool addBackupServer (ft::Server *s)
- bool isPrimary (unsigned long long key)
- bool isBackup (unsigned long long key)
- int logRequest (unsigned long long key, data_t *value)
- int logRequest (std::vector< unsigned long long > keys, std::vector< data_t *> values)
- int logRequest (std::vector< RequestWrapper< unsigned long long, data_t *>> batch, std::vector< RequestWrapper< unsigned long long, data_t *>> *failedBatch=nullptr)
- std::size_t getHash ()

Public Attributes

- cse498::Connection ∗ **primary_conn**
- cse498::Connection ∗ **backup_conn**

Additional Inherited Members

4.4.1 Detailed Description

Server Node definition

4.4.2 Member Function Documentation

4.4.2.1 addBackupServer()

```
bool ft::Server::addBackupServer (
            ft::Server ∗ s )
```

Add server who is backing this one up

Parameters

s	- Server to add to backup server list

Returns

true if added successfully, false otherwise

4.4.2.2 addKeyRange()

```
bool ft::Server::addKeyRange (
            std::pair< unsigned long long, unsigned long long > keyRange )
```

Add key range to primary list

Parameters

keyRange	- pair of min and max key

Returns

true if added successfully, false otherwise

4.4.2.3 addPrimaryServer()

```
bool ft::Server::addPrimaryServer (
            ft::Server * s )
```

Add server who this one is backing up

Parameters

s	- Server to add to primary server list

Returns

true if added successfully, false otherwise

4.4.2.4 getBackupServers()

```
std::vector<ft::Server*> ft::Server::getBackupServers ( )  [inline]
```

Get list of servers acting as this one's backup

Returns

vector of backup servers

4.4.2.5 getHash()

```
std::size_t ft::Server::getHash ( )
```

Get a hash value of this server configuration

Returns

hash of the server

4.4.2.6 getPrimaryKeys()

```
std::vector<std::pair<unsigned long long, unsigned long long> > ft::Server::getPrimaryKeys (
) [inline]
```

Get vector of primary key ranges

Returns

vector of min/max key range pairs

4.4.2.7 initialize()

```
int ft::Server::initialize (
             std::string cfg_file ) [virtual]
```

Initialize server

Returns

status. 0 on success, non-zero otherwise.

Reimplemented from ft::Node.

4.4.2.8 isBackup()

```
bool ft::Server::isBackup (
             unsigned long long key )
```

Check if server is backing up a given key

Parameters

key	- key to check if backing up

Returns

true if backing, false otherwise

4.4.2.9 isPrimary()

```
bool ft::Server::isPrimary (
             unsigned long long key )
```

Check if server is running as primary for a given key

Parameters

key	- key to check if primary

Returns

true if primary, false otherwise

4.4.2.10 logRequest() [1/3]

```
int ft::Server::logRequest (
            unsigned long long key,
            data_t * value )
```

Log a PUT transaction to all backup servers.

Parameters

key	- value of key in table
value	- data to store in table at key

Returns

0 on success, non-zero on failure

4.4.2.11 logRequest() [2/3]

```
int ft::Server::logRequest (
            std::vector< unsigned long long > keys,
            std::vector< data_t *> values )
```

Log a batch of PUT transactions to backup servers.

Parameters

keys	- vector of keys to update
values	- vector of data to store at keys

Returns

0 on success, non-zero on failure

4.4.2.12 logRequest() [3/3]

```
int ft::Server::logRequest (
            std::vector< RequestWrapper< unsigned long long, data_t *>> batch,
            std::vector< RequestWrapper< unsigned long long, data_t *>> * failedBatch =
nullptr )
```

Log a batch of RequestWrapper transactions to backup servers.

Parameters

batch	- batch of backup requests
failedBatch	- will be populated with any requests that were not backed up. This could be due to the backup server being unavailable, or the request being invalid.

Returns

0 on success, KVCG_EUNAVAILABLE if all requests were valid, but no backup server was available. KVC↩ G_EINVALID if any request was not valid.

4.4.2.13 printServer()

```
void ft::Server::printServer (
            const LogLevel lvl )
```

Print server configuration if log level $>$ lvl

Parameters

lvl	- log level to start printing. Will print more at higher levels.

4.4.2.14 shutdownServer()

```
void ft::Server::shutdownServer ( )
```

Shutdown server

The documentation for this class was generated from the following file:

- /root/cjdambro/grad-school/CSE498/gits/fault-tolerance/include/faulttolerance/server.h

4.5 ft::Shard Class Reference

Public Member Functions

- **Shard** (std::pair< unsigned long long, unsigned long long > kr)
- void addServer (ft::Server ∗s)
- ft::Server ∗ getPrimary ()
- void setPrimary (ft::Server ∗s)
- int discoverPrimary ()
- bool containsKey (unsigned long long key)
- unsigned long long getLowerBound ()
- unsigned long long getUpperBound ()
- std::vector< ft::Server ∗ > getServers ()

4.5.1 Member Function Documentation

4.5.1.1 addServer()

```
void ft::Shard::addServer (
            ft::Server ∗ s )  [inline]
```

Add a server to this Shard

Parameters

s	- server to add

4.5.1.2 containsKey()

```
bool ft::Shard::containsKey (
            unsigned long long key )  [inline]
```

Determine if this Shard contains a given key

Parameters

key	- key to check in Shard

Returns

true if contains key, false otherwise

4.5.1.3 discoverPrimary()

```
int ft::Shard::discoverPrimary ( )
```

Discover primary server for a shard to be cached in that shard

Returns

status. 0 on success, non-zero otherwise.

4.5.1.4 getLowerBound()

```
unsigned long long ft::Shard::getLowerBound ( ) [inline]
```

Get the lower bound on this Shard's key range

Returns

lower key value

4.5.1.5 getPrimary()

```
ft::Server* ft::Shard::getPrimary ( ) [inline]
```

Get the cached primary for this Shard

Returns

current primary server

4.5.1.6 getServers()

```
std::vector<ft::Server*> ft::Shard::getServers ( ) [inline]
```

Get the list of servers in this Shard

Returns

list of servers

4.5.1.7 getUpperBound()

```
unsigned long long ft::Shard::getUpperBound ( ) [inline]
```

Get the upper bound on this Shard's key range

Returns

upper key value

4.5.1.8 setPrimary()

```
void ft::Shard::setPrimary (
             ft::Server * s ) [inline]
```

Set the current primary in this Shard's cached data

Parameters

s	- Server pointer to primary

The documentation for this class was generated from the following file:

- /root/cjdambro/grad-school/CSE498/gits/fault-tolerance/include/faulttolerance/shard.h

Chapter 5

File Documentation

5.1 /root/cjdambro/grad-school/CSE498/gits/fault-tolerance/include/faulttolerance/fault↩_tolerance.h File Reference

Public API for KVCG Fault Tolerance protocol.

```
#include <faulttolerance/node.h>
#include <faulttolerance/server.h>
#include <faulttolerance/shard.h>
#include <faulttolerance/client.h>
```

5.1.1 Detailed Description

Public API for KVCG Fault Tolerance protocol.

Index