Table of Contents

4 Cache management utility	3
CCU configuration	3
YAML format	3
Conventions to specify the parameters	4
Parameters	
Working directories	4
Events	5
Logging	5
Statistics collection	
Day categories and time classes	6
Jobs	7
Storage	14
SSD-Caching	19
Initial configuration	20

4 Cache management utility

CCU - Cache Control Utility

CCU is designed to load data into the cache, manage its content and collect statistics. The utility supports a flexible configuration which includes all the specified parameters.

CCU configuration

The main configuration file is the <u>/etc/ccu/ccu.conf</u>. The configuration in this file is specified using the <u>YAML</u> format.

YAML format

The basic elements of the YAML format are:

```
"Key-Value" pair, for example: default: offpeak
value sequence, for example:
        - "00:00:00 - 01:00:00"
        - "17:00:00 - 23:59:59"
```

It is possible to create complex data structures based on the basic elements.

One of the principles describing complex structures laid down in the YAML format is the formatting nested elements rule using indents from the beginning of the line. Elements located on the same hierarchy level must have the same number of leading spaces, for example, the following text:

```
# Time classes
time classes:
                           # Defines the classes of time
   default:
               offpeak # Specifies the default value for time classes
                           # All others items define the names of time
classes and rules for them definition
                           # Each rule is started from the name of day
category and contains
                           # the list of ranges of times in form "HH:MI:SS-
HH:MI:SS"
   peak:
       workdays:
            - "00:00:00 - 01:00:00"
            - "17:00:00 - 23:59:59"
       weekend eve:
            - "00:00:00 - 02:00:00"
```

```
- "16:00:00 - 23:59:59"

weekend:
- "00:00:00 - 02:00:00"
- "09:00:00 - 23:59:59"

holidays:
- "00:00:00 - 02:00:00"
- "09:00:00 - 23:59:59"
```

can be treated as the time_classes data structure having the following fields: default and peak. The peak field in turn is also a structure with the workdays, weekend_eve, weekendand holidays fields which are the named string series.

Given the above, the main rule when editing YAML format documents is <u>using a certain number of</u> <u>whitespaces to form indents instead of using tabs</u>.

Conventions to specify the parameters

- To specify the name of a particular field in a complex nested structure, you should use the full field name, which is an enumeration of all field names separated by a slash, for example: time classes/peak/workdays
- To specify the name of user-defined element or the name of element that refers to another element, you should use the symbolic element definition specified between the signs less and more, for example: time_classes/peak/<the_day_category>
- Unless otherwise specified, the full name is specified to be relative to the entire configuration
 file, for example: time_classes/peak/workdays, or when describing the full name, you can
 specify its base, for example: the time classes description (time_classes) are specified by the
 parameters in the <time_class>/<the_day_category>form that contain a listing of time
 intervals

Parameters

Working directories

The following two parameters specify the directories where files with running processes identifiers will be stored -pid_files_path and working files of the cache management utility - work_files_path. Among the working files, in particular, are the files that store the cache usage statistics.

```
#
# PID files path, default: /var/run/ccu
pid_files_path: /var/run/ccu
#
# Work files directory, default: /var/lib/ccu
work_files_path: /var/lib/ccu
```

Events

When the certain events occur the **ccu** can perform actions specified by the user. Actions to be executed should be specified to meet the rules of the command line interpreter. Only one event currently is supported: creation of a text file containing information about objects stored in caches. To set the response to the event, you should define the events/on_after_enumeration_creation parameter. A command to create a binary cache contents representation in the event configuration template is specified.

```
#
# Events
events:
    # Command which should be executed after caches' files enumeration
    on_after_enumeration_creation: "cut -d' ' -f2
/var/cache/ccu/data/enumerated.cs | url2dic
/var/cache/ccu/data/enumerated.bin"
```

Logging

Logging options include the path to the log files, as well as the message levels to be written to the file. The most detailed journaling level is the *debug* level, i.e. when it is enabled all possible messages will be written to the file. When the *error* journaling level is specified, only errors will be written to the file.

The logging levels are specified separately for each **ccu** command.

```
# Logging parameters
logging:
    path:
                    /var/log/ccu
                                                  # Enabled log levels for
    levels:
particular command
                                                  # Possible levels are:
                                                      error, warning, info,
diagnostic, debug
                                                  # default logging level is
"info"
        load:
                     "info"
                    "info"
        purge:
                    "warning"
        remove:
        online:
                    "info"
                    "info"
        monitor:
```

Statistics collection

The fields of the statistics/collectors/

parameter specify the node address and the port the data about the cache usage will be sent. In the future, the name of the statistical collector can be used when describing the CACHES.

Day categories and time classes

Day classes and time classes are used to specify the restrictions for data loading to the cache. In order to specify the category of the day, you should create the day_categories/<day_category> parameter, whose value is the list of days that fall into the created day category. The day list element can be both the name of the day of the week, as well as a partially or fully defined date. When determining the category of a day for a specific date, the more particular definitions are examined in the first place, after that the more general ones are examined.

```
# Day categories
day categories:
                  # Defines day categories which could be used in time
class definitions
                  # Items under "day categories" define names of
categories;
                  # To define the day categorie it is necessary to specify
list of values,
                  # value can be one of following:
                          week day
                          partially or fully defined date in form
DD.MM.YYYY
                  #
                           for example:
                           "01"
                   #
                                           - the first day of each month
                           "07.01"
                   #
                                           - the 7th of January of each year
                          "09.05.2015"
                                           - 2015, the 9th of May
   workdays:
                   [Mon, Tue, Wed, Thu]
   weekend eve:
                    [Fri]
   weekend:
                   [Sat, Sun]
   holidays:
       - "01.01"
        - "07.01"
        - "23.02"
        - "08.03"
        - "01.05"
```

```
- "09.05"
- "12.06"
- "04.11"
```

Time classes divide the day of a certain category into time intervals. To specify the time class, you should create the time_classes/<time_class> parameter, as well as a parameters set for a time class that define the day category. Its values are represented by a list of time intervals. The default time_classes/default time class parameter is specified in a special way. If the time class can not be determined using the specified parameters, then the default time class will be used instead.

In the example below, the peak time class is specified and the time intervals for each day category considered to be peaking are defined. If no rule specifying the peak time class can be used, then the default time offpeak class will be used.

```
# Time classes
                           # Defines the classes of time
time classes:
   default:
                           # Specifies the default value for time classes
                offpeak
                           # All others items define the names of time
classes and rules for them definition
                           # Each rule is started from the name of day
categorie and contains
                           # the list of ranges of times in form "HH:MI:SS-
HH:MI:SS"
   peak:
       workdays:
            - "00:00:00 - 01:00:00"
            - "17:00:00 - 23:59:59"
       weekend eve:
            - "00:00:00 - 02:00:00"
            - "16:00:00 - 23:59:59"
       weekend:
            - "00:00:00 - 02:00:00"
            - "09:00:00 - 23:59:59"
        holidays:
            - "00:00:00 - 02:00:00"
            - "09:00:00 - 23:59:59"
```

Jobs

The job parameters are specified as nested jobs parameter structures. There are the following tasks:

- jobs/monitor monitoring of the network interfaces and data cache usage;
- jobs/load data cache loading/validating;
- jobs/scan auxiliary task designed for data analysis Storages;

Further, when describing, the full parameter name will be based on the jobs parameter.

Monitoring

For the monitoring job, you should specify the incoming connection parameters (monitor/listener/host and monitor/listener/port), which will be used to receive messages from other processes. If one computer is used for all processes, then these parameters have to match the parameters defined for statistics collectors description.

In other words, the monitor/listener parameter specifies the server side of the IP connection, and the parameters specified within the statistics collectors description determine the client side one.

The monitoring job, in addition to gathering statistics of the processes operating with the cache, also gathers system information on network interfaces. In order to limit the list of network interfaces used to gather statistics, it is necessary to specify them in a list in the monitor/network_interfaces parameter.

```
monitor:
                                                 # listener collects
        listener:
information from different processes about
                                                 # incoming/outgoing traffic
                        localhost
            host:
                        1600
            port:
        network interfaces:
                                                 # list of network interfaces
which is used for getting
                                                 # incoming/outgoing traffic
statistics
                                                 # if list is empty, all
interfaces except "lo" will be used
```

Load

Load parameters are divided into generic ones, which are specified as fields of the load parameter; used in offline-loading (load/offline); used in online-loading (load/online);

Generic load parameters:

```
#
# Load jobs parameters
load:
    ip_binding:  # optional, list of local IP-adresses
which could be used
    # to binding while loading
    ignored_clients:  # list of CIDRs for source IP addresses
```

```
which should be ignored;
                                     # list could be defined internally into
configuration file as list of CIDRs
                                     # under key "cidr list" or into
external files under key "cidr_files";
                                     # if list is defined externally in
file, each line that file must contain
                                     # only one CIDR
            cidr list:
            cidr files:
        rate_limits:
                                     # defines rate limit for data loading
depending on time class;
                                     # value can be one of following:
                                     # - "unlimited"
                                         - number of bytes which defines
amount of loaded data per second
            offpeak:
                      unlimited
            peak:
                        1m
       # ... skipping ...
```

The load/ip_binding parameter specifies the IP address list to be used when loading the data, the listed addresses are used evenly.

The load/ignored_clients parameter allows to specify the list of CIDR to be ignored, the requests being sent from these CIDR should not be taken into account when loading data. CIDRs can be specified both in the configuration file (the load/ignored_clients/cidr_list list), and in the external files (the load/ignored_clients/cidr_files list).

The load/rate_limits parameter defines the loading data speed limitations based on the time classes. To set a limit on the speed of loading data at a certain time, you should to create the load / rate_limits / <time class> parameter which value is used to set the desired limitation.

Offline-load parameters:

```
#
# Load jobs parameters
load:
    # ... skipping ...
    offline:
        parallel_workers: 1 # optional, number of parallel
loading jobs, default 1
        job_awaiting_time: 10 # awaiting time for job in a queue
in seconds;
# if after this time no one job is
in the queue,
# worker finishes its execution
```

Online-load parameters can be divided into the following logical groups:

- Descriptions of IPFIX stream sources
- Description of the input IPFIX flow analysis
- Description of the requested objects collectors
- Description of data loading processes

Descriptions of IPFIX stream sources are specified in the load/online/exporters structure:

```
# Load jobs parameters
   load:
       # ... skipping ...
        online:
            exporters:
                                                # list of IPFIX exporters
                main:
                    queue size: 1000
                                                # optional, max number of
messages from exporter which is sent to analyzing queue,
                                                # must be between 1 and
100000, default: 1000
                                localhost
                                              # host name or IP-address
                    host:
                    port:
                                1500
                                               # listening port
                                                # one of possible protocols:
                    protocol:
                                udp
UDP or TCP
                    information elements:
                                                            # list of used
information elements:
                                                            # names are
reserved, values must be in form "PEN/NUM"
                        timestamp:
                                            "43823/1001"
                                                            # mandatory,
timestamp of event, must be IPFIX dateTimeSeconds
                                            "43823/1005"
                                                            # mandatory,
host name, must be IPFIX string
                                            "43823/1006"
                                                            # mandatory,
                        path:
path to resource, must be IPFIX string
                        login:
                                            "43823/1002"
                                                            # optional,
login, must be IPFIX string
                                            "43823/1003"
                                                            # optional,
                        source ip4:
source IP-address
                                            "43823/1004"
                                                            # optional,
                        destination ip4:
destination IP-address
                        referal:
                                            "43823/1007"
                                                            # optional,
referal, must be IPFIX string
                                            "43823/1008"
                                                            # optional, user
                        user agent:
agent, must be IPFIX string
                                            "43823/1009"
                                                            # optional,
                        cookie:
cookie, must be IPFIX string
       # ... skipping ...
```

To create a description of the IPFIX stream source, you should create the load/online/exporters/<IPFIX-источник>structure according to the example above. You should pay attention to specifying the following parameters:

- load/online/exporters/<IPFIX-источник>/host and load/online/exporters/<IPFIX-источник>/port - definition of the server socket parameters (IPFIX stream should be sent to this address);
- load/online/exporters/<IPFIX-источник>/protocol the definition of the protocol used to send messages;

The input IPFIX flow analysis description is specified in the load/online/analyzing structure:

```
# Load jobs parameters
    load:
       # ... skipping ...
        online:
            # ... skipping ...
            analyzing:
                                                # the analyzing processes
perform URL rewriting, filtering
                                                # and binding URLs getting
from exporters to cache's definition
                                                # each analyzzng process
sends valid URLs to collection queue
                parallel workers: 1
                                                # optional, number of
parallel analyzing jobs, default 1
                queue size:
                                 1000
                                                # optional, max number of
messages from each analyzing process
                                                # which is sent to
collecting queue, must be between 1 and 100000,
                                                # default: 1000
        # ... skipping ...
```

When specifying the analyzer parameters, you should focus initially on the load/online/analyzing/parallel_workers parameter. When there is a large number of messages from the processes receiving an IPFIX stream, the analyzer processes number should be increased. The indication to increase the number of analyzer processes is the message about the message queue overflow in the log files of the processes responsible for receiving the IPFIX stream.

The description of the requested objects collectors is specified in the load/online/collectors structure:

```
#
# Load jobs parameters
load:
    # ... skipping ...
online:
```

```
# ... skipping ...
           collectors:
                                               # list of collectors
               default:
                                               # the collector is used for
aggregation of identical URLs
                                               # and distributes them into
time windows
                                               # each event could hit no
more than 1 window according to event's timestamp
                                               # the time window is a
period of time which defines interval of time in the past
                                               # it is possible to have
many time windows, in this case each window is
                                               # connected to previous one
                                               # optional, number of
                   slots:
                                   24
windows, could not be greater than 100, default 24
                   window:
                            3600 # optional, window size in
seconds, could not be less than 60 secs, default 3600
               week by 4 hours:
                   slots:
                                   42
                   window:
                            14400
       # ... skipping ...
```

The collector of the requested objects is the main factor for deciding whether to start the object download. Logically, the collector is a collection of counters in the sliding time intervals in the past from the present. When an object is requested, the counter is incremented in a certain interval, and as soon as the total value of all object counters reaches the threshold value specified within the cache description, the object loading is permitted.

In order to create the collector description, you should create the following parameters:

- load/online/collectors/<object collector>/slots number of intervals;
- load/online/collectors/<object collector>/window the size of each interval;

In the example above, two collectors are described: default and week_by_4_hours. The default collector has 24 1-hour intervals, i.e. allows you to analyze the number of object requests during the day.

The description of the data loading processes is specified in the load/online/loading structure:

```
#
# Load jobs parameters
load:
    # ... skipping ...

online:
    # ... skipping ...
```

```
loading:
                                              # optional, number of
               parallel workers:
                                       1
parallel loading jobs, default 1
               queue size:
                                       100 # optional, max number of
messages from collecting process
                                               # which is sent to loading
queue, must be between "parallel_workers" and 10000,
                                               # default: 100
               unbuffered queue size: 10
                                               # optional, if current queue
size less or equal this value, the messages from
                                               # collecting process will
put into loading queue immediately. The messages will be buffered
                                               # and ordered before puting
into loading queue according to URLs' weight;
                                               # must be between 0 and
"queue size" - 2 * "parallel_workers", default 2 * "parallel_workers"
       # ... skipping ...
```

When specifying the load process parameters you should pay attention to the number of parallel processes. Do not significantly increase the queue size (the load/online/loading/queue_size parameter), since when using a large value for this parameter at startup, the queue will be populated with objects that satisfy the load startup conditions, but it is possible that objects that are requested more frequently will appear during the queue processing.

Scanning

Scanning is an auxiliary job that starts when loading and deleting data from the cache.

```
# Scannig jobs parameters
       workers:
                                           # the scanning processes perform
scan cache directories on demand;
                                           # each scanning process sends
information about found items to scanning queue
            parallel workers: 4
                                           # optional, number of parallel
scanning jobs, default 4
           job queue size: 5000
                                           # optional, max number of
messages in job queue, must be between 1000 and 10000
                                           # default: 5000
            result queue size: 100000
                                           # optional, max number of
messages from each scanner process
                                           # which are sent to result
scanning queue, must be between 1000 and 1000000,
                                           # default: 100000
```

The functions performed by the scan task include the following:

• initial directories scanning when the load process starts or when old objects are removed;

- checking the lifetime of object in the cache and removing it, if necessary;
- removal of objects on request from the control process;
- checking the compliance of the object stored in the cache with its original;

Storage

Storage parameters are specified as the storage_parameters parameter fields. All storage parameters are divided into the following groups:

- storage general characteristics the storage parameters/general structure
- CACHEs parameters description the storage_parameters/caches structure

General parameters specify the path to the storage root, as well as the maximum size that can be allocated to all the objects of the cache:

The parameters of a particular cache are specified as the storage_parameters/caches/<cache description> structure, for example:

```
storage_parameters:
    # ... skipping ...

caches:
    youtube.com:
    is_enabled:
    yes # optional (default yes), if

value is "no" the cache definition

# will not be used for loading

data

# ... skipping ...
```

specifies the cache description using the youtube.com name.

Cache description

The parameter names will be based on the storage_parameters/caches/<cache description> parameter when describing cache parameters in this section.

The is enabled parameter specifies whether objects will be loaded into the cache or not.

Statistics

The cache statistics are specified in the statistics structure:

```
statistics:
    group: "youtube.com" # name of statistic's

group

# all statistics in the

same group is aggregated together
    collector: local # name of statistics

collector
```

The statistics/group the parameter specifies the group where the cache operations will be taken account. The statistics/collector parameter specifies statistics collector, which will account for cache transactions.

Online-loading

For the online-loading, you need to specify the following parameters:

```
online:
    collector: "week_by_4_hours" # name of collector which
should be used # for online processing
    validating:
        interval: 24h # revalidating interval
for particular cache item; # could be used only for
caches with "general" algorithm
```

The online/collector parameter specifies the collector name of the requested objects. The online/validating/interval parameter can only be used by the cache with *general* load algorithm and specifies the time interval defining how often the cache-stored object will be checked against its original.

Loading

The following parameters are used to specify the load rules:

```
loading:
    algorithm: "youtube.com" # internal algorithm name
    required_weight: 3 # min URL's weight which is
required to start loading, default 3
```

```
urls:
              matching:
                           # list of parameters for URL matching and
rewriting;
                           # - the "key" item is used for unique
identification of object;
                           # - the "target" item specifies target value for
URL; usually target is used as
                           # as additional URL for loading
                           # - the "weight" item is used as weight addition,
default 1;
                           # all items under "sources" are used as a source
masks;
                           # the "key" and "target" can be omitted in this
case
                           # the sources URLs will not be rewritten.
                           # it is necessary to have at least one URL in
sources.
                           # It is possible to use in the key and target's
expression variables from sources;
                           # See RE syntax on
https://docs.python.org/2/library/re.html
                      key:
                                  '\1'
                                  1
                      weight:
                      sources:
                          - '^www\.youtube\.com/watch\?v=([a-zA-Z0-9 \-]+)'
                          - '^www\.youtube\.com/embed/([a-zA-Z0-9 \-]+)'
              ignoring:
                         # list of regular expr. for ignoring urls
                           # default: all matched with matching parameters
URLs will be processed
              loadable rejecting: # list of regular expr. for rejecting
"real" urls
                                     # which is obtained according to
particular URL
                                     # default: all "real" URLs will be
processed
                                     # See RE syntax on
https://docs.python.org/2/library/re.html
                  - "itag=(?!(18|22|140)(?![0-9]))"
```

- The loading/algorithm parameter defines algorithm to be used for object loading. The following algorithms exist:
 - voutube.com to download video files from youtube.com;
 - rutube.ru to download video files from rutube.ru:
 - vk.com to download video files from vk.com;
 - general general algorithm for loading the source URL;
- The loading/required_weight parameter specifies the required collector weight of the requested objects; when the weight is reached, object loading will start;

- The loading/urls/matching parameter specifies the rules for original URL processing/overwriting; This parameter is a list of the elements structure with the following fields:
 - key the object key, which is issential for the loading algorithm, for example, the object key for the youtube.com algorithm is presented by the video identifier;
 - weight the weight to be added when the requested object in the collector will be accounted;
 - target rewritten original URL;
 - sources a list of regular expressions for analyzing original URL;

When specifying the mapping rules, you can use references to the elements of the original URL. So in the example above, the key parameter value is the video id specified as the 1st group in regular expressions.

- The loading/urls/ignoring parameter specifies a list of regular expressions for URLs that passed the matching stage ('loading/urls/matching') and to be excluded from the load;
- The loading/urls/loadable_rejecting parameter specifies a list of regular expressions for loadable URLs to be excluded from the load; downloadable URLs are the links to real files obtained when the loading algorithm (loading/algorithm) is executed for original URL, for example, in order to execute the *youtube.com* algorithm for the URL specifying the video WEB-page to be viewed, a list of URLs for video and audio files will be received in all available formats.

Storage

The following parameters are used to specify the rules for storing objects:

```
storage:
          path:
                     "sites/youtube.com"
                                              # regard to
"storage parameters/general/path"
          levels:
                    "2:2"
                                              # optional, store files in
subfolders created by leading symbols
                                              # of md5 sum of file name
                                              # possible values:
                                                  "1"
                                                          - creating
subfolder by the last symbol of md5 sum
                                                  "2"
                                                          - creating
subfolder by the last two symbols of md5 sum
                                                  "1:2"
                                              #
                                                          - creating
subfolder by the last symbol of md5 sum
                                              #
                                                            and creating
subfolder by 2th and 3th symbols
                                                            from the end of
                                              #
md5 sum inside
                                                  "2:2"
                                                          - creating
subfolder by the last two symbols of md5 sum
                                                            and creating
subfolder by 3th and 4th symbols
                                                            from the end of
                                              #
md5 sum inside
```

```
max size:
                          "unlimited"
                                             # optional ("unlimited", 0 or
missed - general limit will be used),
                                             # maximum data size, possible
suffixes are:
                                             # Kb, Mb, Gb, Tb, Pb or without
trailing "b" just K, M, G, T, P
                        "30d"
                                             # optional (0 or missed -
          expiry time:
unlimited), expiry time,
                                             # default in seconds (without
suffix)
                                             # possible suffixes are:
                                                 m - minutes
                                                 h
                                                     - hours
                                             #
                                                 d
                                                     - days
```

- The storage/path parameter specifies the path for storing objects relative to storage parameters/general/path parameter value;
- The storage/levels parameter specifies the number of additional directories to be created based on md5-sum of the file name;
- The storage/max_size parameter specifies the maximum total size of all objects stored in the cache. If this restriction is not specified, then the entire repository size restriction will be used instead;
- The storage/expiry_time parameter specifies the lifetime of an object within the cache from the moment it is loaded;

Constraints

For cache objects, you can set constraints, for example:

```
constraints:
                                           # optional, specifies
constraints for loaded files
         min file size:
                           128k
                                           # optional (default 0), min file
size, possible suffixes are:
                                           # Kb, Mb, Gb, Tb, Pb or without
trailing "b" just K, M, G, T, P
         max_file_size: "unlimited"
                                           # optional (default unlimited),
max file size, possible suffixes are:
                                           # Kb, Mb, Gb, Tb, Pb or without
trailing "b" just K, M, G, T, P
         # optional, command which should be executed after file loading;
         # if command returns non-zero result, the loaded file will be
assumed as invalid an will be removed;
         # the next variables could be used in the command:
         # {cache name}
                                 - cache name
             {full file name}
                                 - fully qualified file name
         post_load_validation: "ft=`file -b {full_file_name}`; echo $ft;
echo $ft | grep -E \"WebM|ISO Media|MPEG\" 2>/dev/null 1>&2"
```

SSD-Caching

It is possible to get caching of the the most frequently requested storage objects on an SSD, to do so you should define the following parameters:

```
# SSD caching parameters
ssd caching:
    is enabled:
                                                # enable or not SSD caching,
                        no
by default SSD caching is disabled
                        "/var/cache/ccu/ssd"
                                                # optional, base path for
SSD caching area, default: /var/cache/ccu/ssd
                                                # NOTES:
                                                # - to SSD caching you
have to mount SSD on specified path (or create link /var/cache/ccu/ssd to
mount point)
                                                    - changes this parameter
should be performed in cooperation with changes in web server configuration
                                                      (it is recomended do
not change this parameter)
    max size:
                                                # maximum data size for
storing on SSD, possible suffixes are:
                                                # Kb, Mb, Gb, Tb, Pb or
without trailing "b" just K, M, G, T, P
    required weight:
                                                # optional, min URL's weight
                       10
which is required to start caching, default 10
    uri prefixes:
                                                # URI prefixes which are
used in HTTP requests to access data
        ssd cache requests:
                                                # optional, prefix for
requests to SSD cache, default: "/ssd"
        main storage requests: "/cache"
                                                # optional, prefix for
requests to main storage, default: "/cache"
                                                # optional, interval of time
    frozen time:
                        180
in seconds after putting file into cache in which that
                                                # file cannot be replaced by
another one, default: 3 * collecting window
                                                # frozen time cannot be less
than collecting window
    collector:
                                    # the collector is used for aggregation
of identical requests
                                    # and distributes them into time windows
                                    # optional, number of windows, could not
        slots:
                        60
be greater than 120, default 60
        window:
                                    # optional, window size in seconds,
                        60
could not be less than 60 secs, default 60
    workers:
                                    # the worker processes perform scan
```

```
cache directories on demand;

# each process sends information about

found items to result queue

parallel_workers: 2 # optional, number of parallel jobs,

default 2

job_queue_size: 5000 # optional, max number of messages in

job queue, must be between 1000 and 10000

# default: 5000

result_queue_size: 100000 # optional, max number of messages from

each process

# which is sent to result queue, must be

between 1000 and 1000000,

# default: 100000
```

When setting parameters, you should pay attention to the number of working processes and the collector parameters.

Initial configuration

When installing **ccu** the /etc/ccu/ccu.conf.default configuration template file will be created. To get the proper configuration, you should copy the template file to /etc/ccu/ccu.conf and edit/check the value of at least the following parameters:

- jobs/monitor/network interfaces
- jobs/load/ip_binding
- jobs/load/online/exporters/main/host
- jobs/load/online/exporters/main/port
- jobs/load/online/exporters/main/protocol
- storage parameters/general/max size