## NAME

**racluster** – aggregate **argus(8)** data files.

## SYNOPSIS

**racluster** [**–f** *conf*] [**–m** *agr(s)*] [**–M** *mode(s)*] [**raoptions**] [**--** *filter-expression*]

## DESCRIPTION

**Racluster** reads **argus** data from an *argus-data* source, and clusters/merges the records based on the flow key criteria specified either on the command line, or in a racluster configuration file, and outputs a valid *argus-stream*. This tool is primarily used for data mining, data management and report generation.

The default action is to merge status records from the same flow and argus probe, providing in some cases huge data reduction with limited loss of flow information. Racluster provides the ability to modify the flow model key, either using the "-m" option, or in the racluster.conf file, allowing records to be clustered based on any number of attributes. This supports the development of important reports, such as MPLS LSP usage statistics, DiffServe flow marking policy verification, VLAN group behavior, IP distance related measurements, routing loop detection, traceroute path data recovery, and complex availability/reachability reports, to name just a few useful applications.

Please see racluster.5 for detailed information regarding racluster configuration.

## OPTIONS

Racluster, like all ra based clients, supports a number of **ra options** including filtering of input argus records through a terminating filter expression, and the ability to specify the output style, format and contents for printing data. See **ra(1)** for a complete description of **ra options**. **racluster(1)** specific options are:

**–m** *aggregation object*

Supported aggregation objects are:

| | |
|---|---|
| **none** | use a null flow key. |
| **srcid** | argus source identifier. |
| **smac** | source mac(ether) addr. |
| **dmac** | destination mac(ether) addr. |
| **soui** | oui portion of the source mac(ether) addr. |
| **doui** | oui portion of the destination mac(ether) addr. |
| **smpls** | source mpls label. |
| **dmpls** | destination label addr. |
| **svlan** | source vlan label. |
| **dvlan** | destination vlan addr. |
| **saddr/[l\|m]** | source IP addr/[cidr len \| m.a.s.k]. |
| **daddr/[l\|m]** | destination IP addr/[cidr len \| m.a.s.k]. |
| **matrix/l** | sorted src and dst IP addr/cidr len. |
| **proto** | transaction protocol. |
| **sport** | source port number. Implies use of 'proto'. |
| **dport** | destination port number. Implies use of 'proto'. |
| **stos** | source TOS byte value. |
| **dtos** | destination TOS byte value. |
| **sttl** | src -> dst TTL value. |
| **dttl** | dst -> src TTL value. |
| **stcpb** | src -> dst TCP base sequence number. |
| **dtcpb** | dst -> src TCP base sequence number. |
| **inode[/l\|m]]** | intermediate node IP addr/[cidr len \| m.a.s.k], source of ICMP mapped events. |
| **sco** | source ARIN country code, if present. |
| **dco** | destination ARIN country code, if present. |
| **sas** | source node origin AS number, if available. |

**das** destination node origin AS number, if available.
**ias** intermediate node origin AS number, if available.

**−M** *modes*
Supported modes are:

**correct** Attempt to correct the direction of flows by also searching the reverse flow key, if a match isn't found in the cache. This mode is on by default when using the default full 5-tuple flow key definitions.

**nocorrect** Turn off flow correction for direction. This mode is used by default if the flow key has been changed.

**norep** Do not generate an aggregate statistic for each flow. This is used primarily when the output represents a single object. Primarily used when merging status records to generate single flows that represent single transactions.

**rmon** Generate data suitable for producing RMON types of metrics.

**ind** Process each input file independantly, so that after the end of each inputfile, racluster flushes its output.

**replace** Replace each inputfile contents, with the aggregated output. The initial file compression status is maintained

**−V** Verbose operation, printing a line of output for each input file processed. Very useful when using the ra() -R option.

## INVOCATION

A sample invocation of **racluster(1)**. This call reads **argus(8)** data from **inputfile** and aggregates the TCP protocol based **argus(8)** data. By default, **racluster(1)** merges using the standard 5-tuple flow key. This method is used to merge multiple status records into a single flow record per transaction.

```
% ra -r argus.tcp.2012.02.13.12.20.00
    StartTime      Dur Trans      Flgs  Proto        SrcAddr  Sport   Dir        DstAddr  Dport  TotPkts State
 12:23:07.268    0.997     1  e i         tcp   192.168.0.68.59016     ->  208.59.201.75.http        298   CON
 12:23:08.294    1.000     1  e           tcp   192.168.0.68.59016     ->  208.59.201.75.http        111   CON
 12:23:09.294    0.991     1  e d         tcp   192.168.0.68.59016     ->  208.59.201.75.http        637   CON
 12:23:10.331    0.330     1  e           tcp   192.168.0.68.59016     ->  208.59.201.75.http         89   CON
 12:23:32.183    0.010     1  e           tcp   192.168.0.68.59016     ->  208.59.201.75.http          3   FIN

% racluster -r argus.tcp.2012.02.13.12.20.00
    StartTime      Dur Trans      Flgs  Proto        SrcAddr  Sport   Dir        DstAddr  Dport  TotPkts State
 12:23:07.268   24.925     5  e d         tcp   192.168.0.68.59016     ->  208.59.201.75.http       1138   FIN
```

A sample invocation of **racluster(1)**. This call reads **argus(8)** data from **inputfile** and aggregates the TCP protocol based **argus(8)** data, based on the source and destination address matrix and the protocol. It reports the metrics as a percent of the total.

```
% racluster -r argus.2012.02.13.17.20.00 -m saddr/16 daddr proto -% \
       -s stime dur trans proto saddr dir daddr pkts state - tcp and port https

    StartTime      Dur   pTrans  Proto        SrcAddr  Dir        DstAddr  pTotPkts State
 17:49:54.225    8.101   33.333    tcp 192.168.0.0/16   ->   17.154.66.18    23.372   FIN
 17:48:42.607  179.761   13.333    tcp 192.168.0.0/16   ->  17.172.224.25    31.052   FIN
 17:50:01.113    0.803    6.667    tcp 192.168.0.0/16   -> 17.250.248.161     5.676   FIN
 17:49:54.525    1.153    6.667    tcp 192.168.0.0/16   ->  64.12.173.137     5.509   FIN
 17:50:35.411  101.133   26.667    tcp 192.168.0.0/16   ->  184.28.150.87    19.199   RST
 17:49:56.061   73.415    6.667    tcp 192.168.0.0/16   ->   205.188.8.47    11.018   RST
 17:49:55.677    0.434    6.667    tcp 192.168.0.0/16   -> 205.188.101.10     4.174   FIN
```

## COPYRIGHT

Copyright (c) 2000-2014 QoSient. All rights reserved.

**SEE ALSO**

**racluster(5), ra(1), rarc(5), argus(8),**

**FILES**

**AUTHORS**

Carter Bullard (carter@qosient.com).

**BUGS**