DSAT Documentation

Release 0.6

julien@tayon.net

May 04, 2015

Contents

DSAT is a framework for handling and connecting distributed writers and producers.

It also provides an integration with circus a way to monitor and handle your processes.

CHAPTER 1

Changelog

Version 0.5 Alpha

Version 0.6 adding parse_mesg migrating to tornado event loops in ssat requires change in DSAT

1.1 Changelog and roadmap

1.1.1 Changelog

0.5.1

- GPG signing package, setting up the build chain (read the docs, pypi ...)
- making sure README.rst is both used in docs and pypi description

0.5.0 Initial release.

1.1.2 Convention:

version x.y.z

while in beta convention is :

- **x** = 0
- **y** = API change
- **z** = bugfix and/or improvement

and then

- **x** = API change
- **y** = improvement
- **z** = bugfix

1.1.3 Roadmap

0.6.1

- introducing a garbage collector that can serialize buffer on disk
- going tornado

0.6 Make a full functionnal testing framework so people can build their own topology and measure impacts of choices. When it works for me (tm) switch to beta.

1.1.4 TODO limitations

- OOP to reduce the repeating of args in function calls
- in the tests with carbon clean hierarchy of metrics
- implements a simple router
- binary star on more than one server orchester
- find a way to deploy and package satellites set
- **find a consistent naming (power of analogy will go full retarded with** astronomy)
- ease the multi hosts satellites connection
- invent a way to modify the code in the orchester for routing by using

  PUB/SUB for rerouting informations simplified coroutines (going the bright side of 10th greenspun law and assumming to borrow the concepts of CLISP)
- **make it easy to change the serialization format (utlimately find a way** to use pack and unpack to go full speed and ease the transmission of data coming from C)
- **promote the use of badass archery library and its confusing naming because** I hate humanity (some people wants to watch the world burn)
- reduce code, improve reliability, reduce functionnalities to have a better code
- **Maybe draw attention to developers that they are totally acculturated and** re explain some basics of networks/system so that they are aware that their ignorance may cause security issues. Highlight the fact that powerfull tools are a way to shoot yourself a nuclear missile in the knee
- **ERROR HANDLING even I is not totally top notch on the topic, find a subset** of best practices that prevents a DOS by repeated failures of do something at the orchester/tracker level to actively modify routing/ alert when something goes wrong
- **Use pytables to store efficiently messages in a hierarchical way for messages** that are critical and need to be replayed
- **write the limitations** [THIS CANNOT BE ACID BY DEFINITION, like any] distributed systems. (can be improved later by using PAIR of REP/REQ for statefull connections that should make it reliable enough if you had a on_happy end hook)
- write a tracker that takes full use of the FSM and implement the time_out stuff
- **put a notice about the halting problem and how we cannot differenciate a code** in an infinite loop and a processing that takes a lot of time
- put a notice on how to design asynch system: bottom half architecture: satellites are bottom halfs.
- **put a link to the fact that a complex system is by nature non deterministic** (yes it is chaotic, but compared to systemd my architecture embrace complexity and the positive property of complex systems such as the resilience to perturbations)
- **a big chapter on clock and time and distributed system, and how** you don't really need to read the indigests pedantic INRIA papers on vector clock to make a distributed systems works correctly without clock (hint : time is the accident of the accidents)