Monkey Server

Eduardo Silva
@edsiper
eduardo@treasure-data.com

Embedded Linux Conference Europe

About Me

TREASURE DATA

Eduardo Silva

- Github & Twitter @edsiper
- Personal Blog http://edsiper.linuxchile.cl

Treasure Data

- Open Source Engineer
- Fluentd / Fluent Bit http://github.com/fluent

Projects

- Monkey HTTP Server http://monkey-project.com
- Duda I/O http://duda.io

HTTP Everywhere

Use cases

- Home Automation
- Big Data
- Data Collection
- Internet of Things
- Wearables
- Automotive

basically everywhere...

Monkey HTTP Server

Monkey HTTP Server

Highlights

- Event driven
- Multi-thread
- Zero copy strategy
- Memory optimization
- Secure
- Optimized for Linux, but compatible with others.
- Embedded Linux / Yocto
- Open Source: Apache License v2.0

Monkey HTTP Server

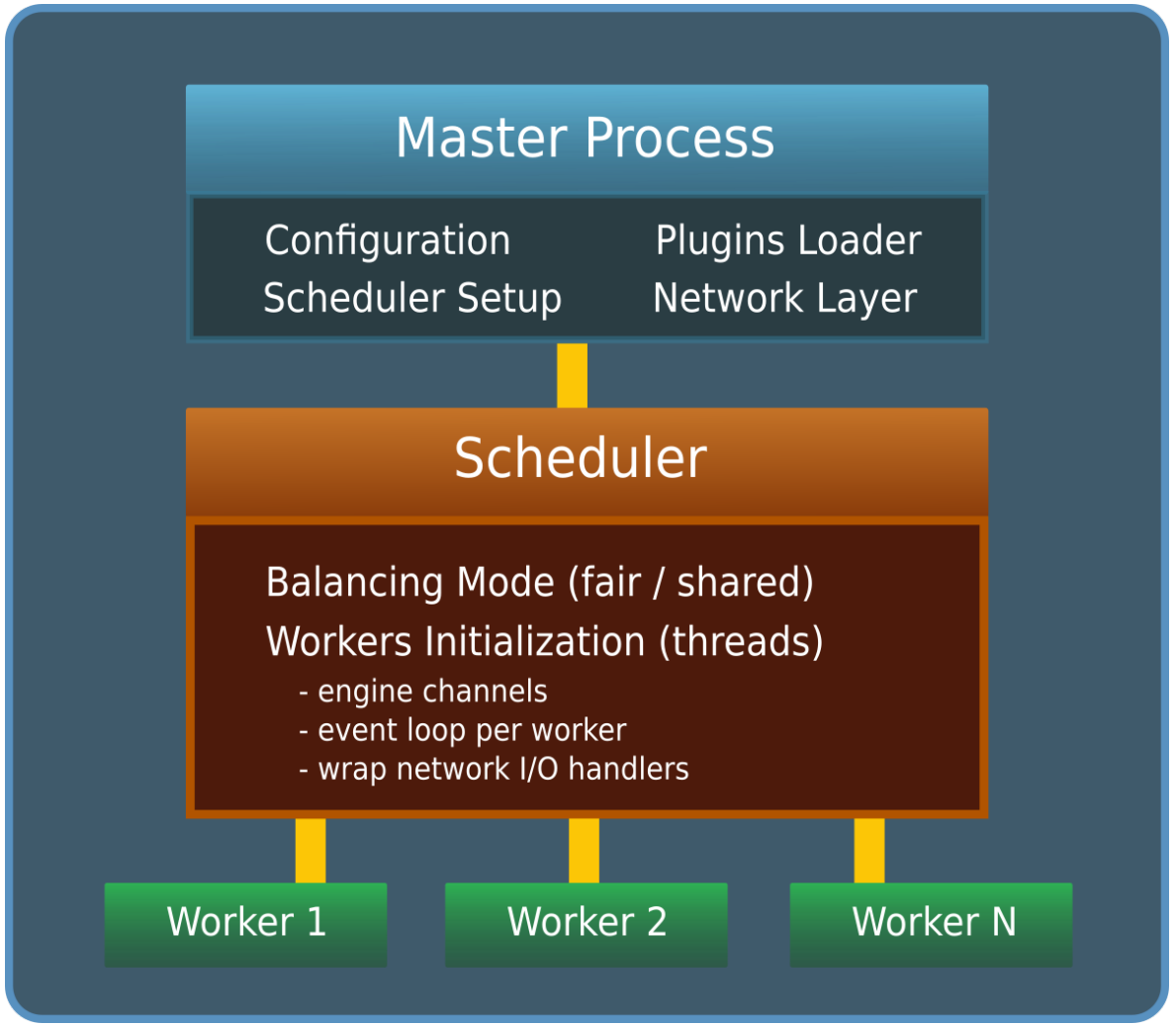
Modular Design

- **mk_core**: agnostic runtime for handling of strings, memory, event loop, configuration reader, vectors and utilities within others.
- **mk_server**: TCP server, data streams, protocols, plugins manager, scheduler and general runtime setup.
- **plugins**: networking layer for I/O operations, stage handlers and content handlers.

Architecture

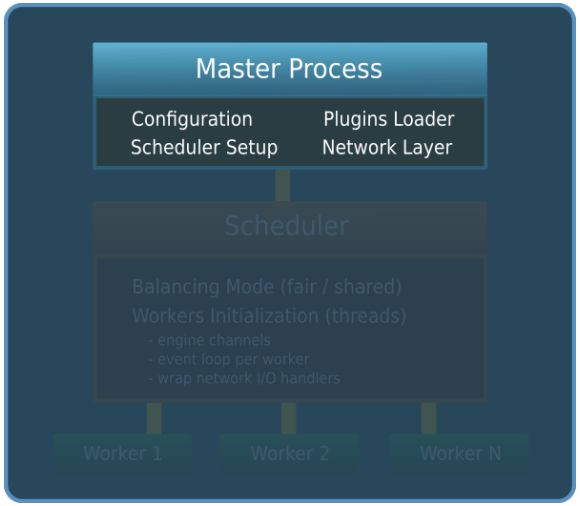
Monkey HTTP Server

Architecture



Monkey HTTP Server

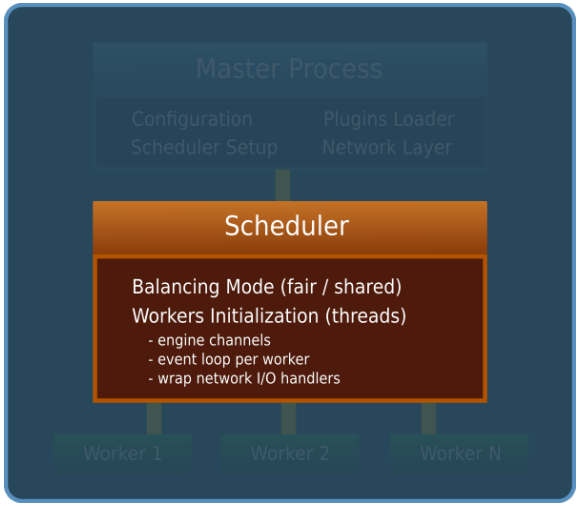
Master Process



- Prepare the environment.
- Read and *process* configuration.
- Load *plugins* and initialize them.
- Create a *Scheduler* instance.
- All things required before joining the event loop.

Monkey HTTP Server

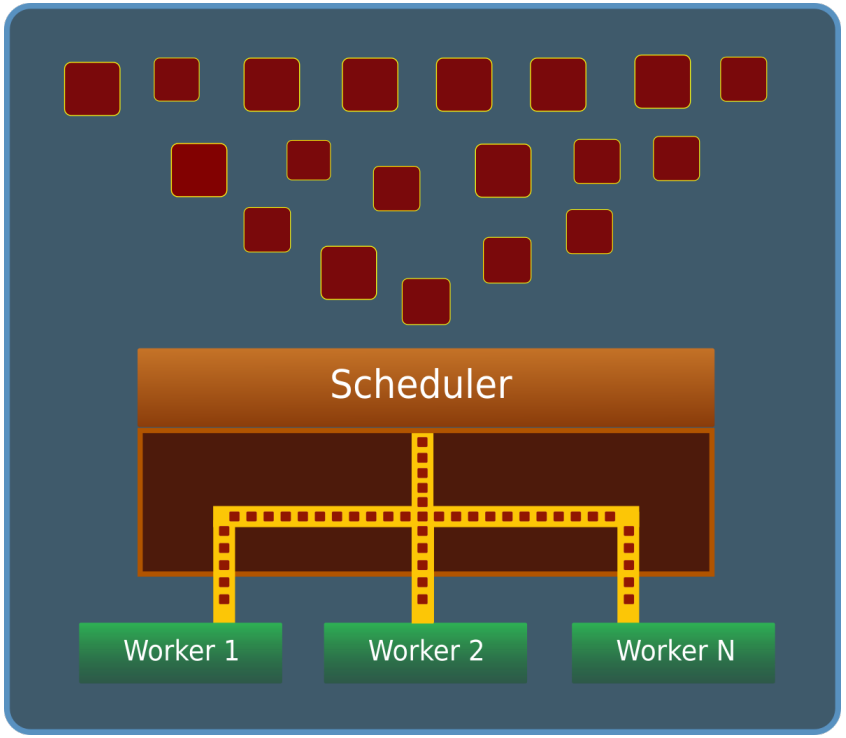
Scheduler



- Balancing mode / OS check:
  - Fair balancing
  - Shared TCP ports (SO_REUSEPORT)
- Setup Listeners.
- Create the event loop for each worker.
- Handle incoming connections.

Monkey HTTP Server

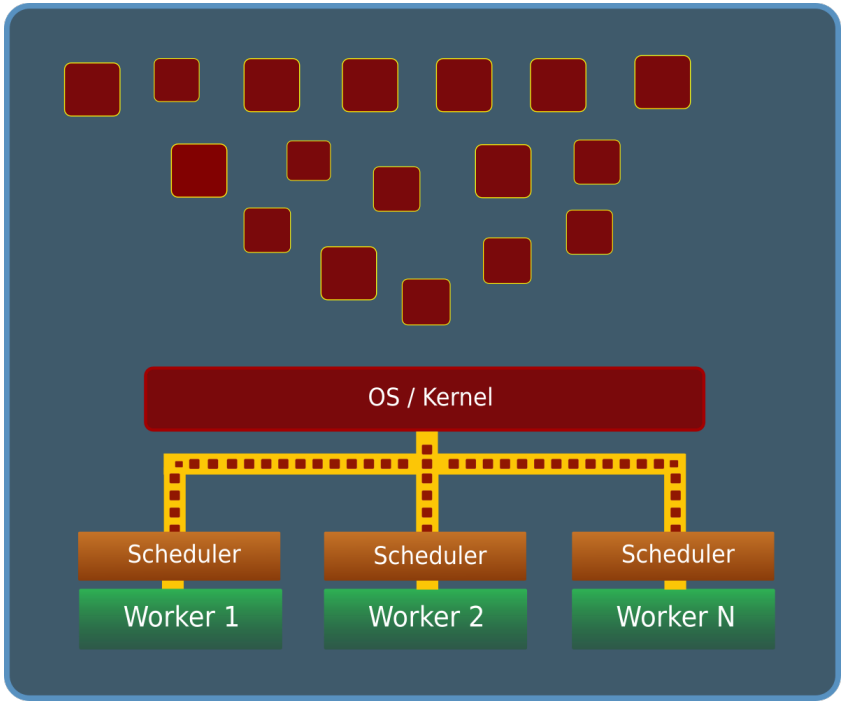
Scheduler mode: Fair balancing



- One scheduler instance.
- It accepts all connections.
- Distribute to the less busy worker.
- Old fashion mechanism.

Monkey HTTP Server

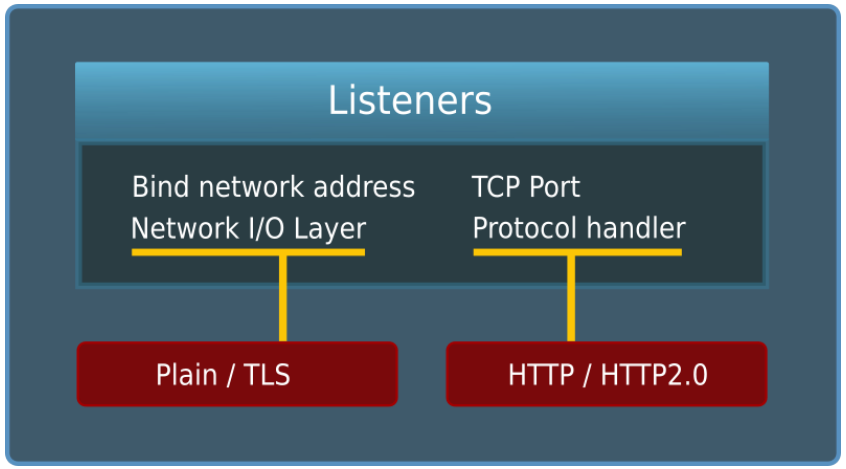
Scheduler mode: Shared TCP ports



- Scheduler handler, one per worker.
- Connections are distributed by the Kernel.
- High performance, less contention.
- SO_REUSEPORT

Monkey HTTP Server

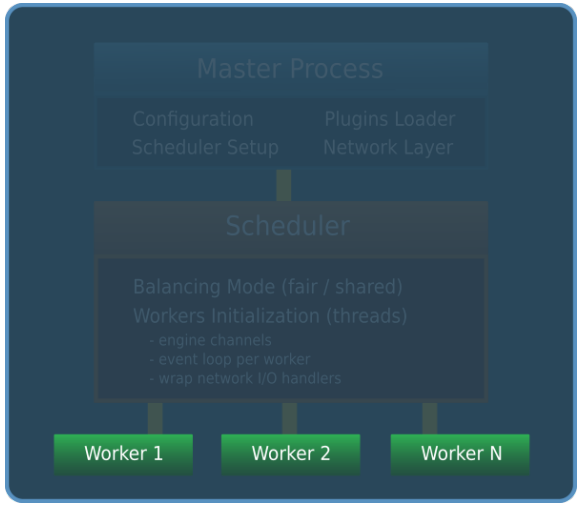
Scheduler: Listeners



- Bind a network address to listen for connections.
- TCP port
- Associate Network I/O layer handler plugin.
- Associate protocol handler for each connection.

Monkey HTTP Server

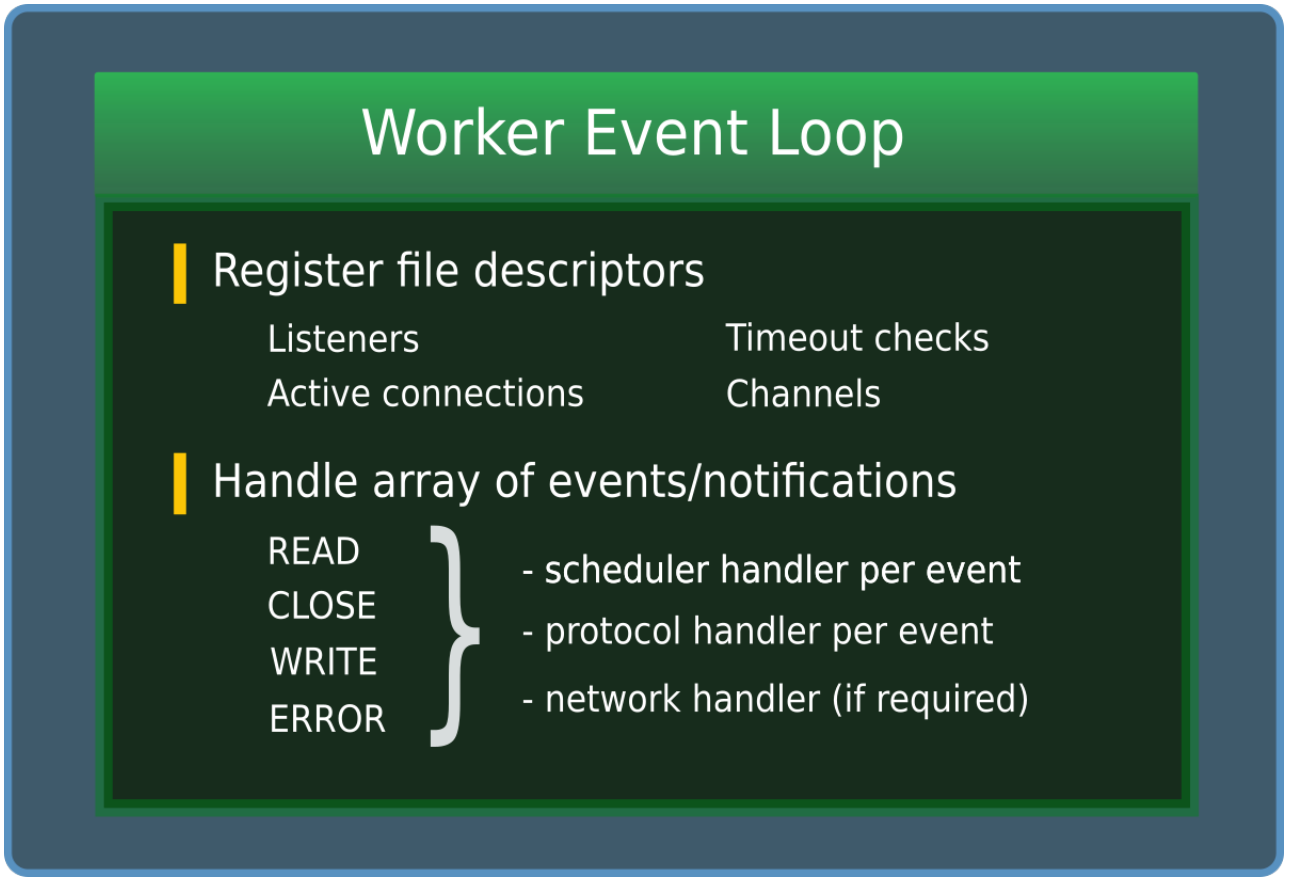
Workers



- Each worker have one event loop.
- Associate Listeners with connections.
- Handle protocol operations based on events notified.
- Timeout connections.
- Cleanup sessions.

Monkey HTTP Server

Workers: event loop



Plugins

Monkey HTTP Server

Plugins

- Monkey **core** provides only basic HTTP processing

- **Plugins** provides interfaces for:
  - Network I/O operations
  - Security
  - Content Handling
  - HTTP Stages / Hooks

Monkey Plugins

Network Layer: I/O

- **Liana**: Basic socket I/O handling, for short plain sockets.
- **TLS**: built on top of mbedTLS, provides SSL and TLS encryption capabilities when writing/reading data over sockets.

"Monkey core and content handling plugins are not aware about how the network I/O is being handled, unless they ask for"

Monkey Plugins

Plugins: stages

Each stage represents a phase of an incoming request cycle inside the server:

STAGE 10	Connection accepted.
STAGE 20	Request have been received.
STAGE 30	Content handler.
STAGE 40	Response completed.
STAGE 50	Connection closed.

Monkey Plugins

Plugins: STAGE 10

Every time a connection is accepted by the scheduler, the plugins associated with this stage are triggered.

The security plugin **mandril**, hooks to this stage and provides restrictions by IP.

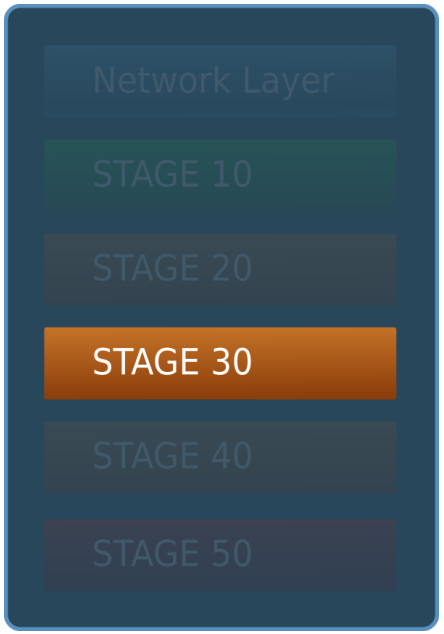
Monkey Plugins

Plugins: STAGE 20

This stage triggers the plugins callbacks every time a *request* have arrived successfully to the server.

Monkey Plugins

Plugins: STAGE 30



Well known as the **content handler**, owns the request and process a response. Some plugins that works at this level are:

- FastCGI
- CGI
- Directory Listing
- Duda I/O

Monkey Plugins

Plugins: STAGE 40

Once a request have finished like completed successfully or due to an exception, this stage is triggered.

The **logger** plugin makes use of this stage to make sure to register all kind of events.

Monkey Plugins

Plugins: STAGE 50

This stage triggers the associated callbacks every time a TCP connection is closed by a protocol, scheduler timeout or due to any network anomaly found.

Monkey HTTP Server

Memory Handling

- Just **one** memory allocation per TCP connection.
- Jemalloc memory allocator built-in by default, if desired it can be disabled at build time.

Monkey HTTP Server

System calls / Tweaks

- sendfile(2) static content delivery
- writev(2) write multiple buffers as an array (headers)
- splice(2) move data between file descriptors (logger)
- epoll(7) I/O event notification on Linux
- kqueue(2) I/O event notification on OSX & BSD
- accept4(2) accept connection and set option (non block)

- TCP_CORK send only full packets (on-demand)
- SO_REUSEPORT shared TCP ports (if available)
- SOCK_NONBLOCK non-blocking sockets

Monkey HTTP Server

Clock thread: cached date

The server spawns a thread named *clock*. It basically update a local memory buffer with a formatted version of the current time. This info is mandatory in the response header.

```
HTTP/1.1 200 OK
Server: Monkey/1.7.0
Date: Wed, 06 Oct 2015 10:30:00 GMT_
Last-Modified: Mon, 12 Jul 2014 17:06:23 GMT
Connection: KeepAlive
```

note: if you get 1000 request per second, you don't want to do a string formatting 1000 times for the same time.

Monkey HTTP Server

Indented Configuration

Force people to use an indented format, avoid a spaghetti configuration.

```
[SERVER]
    Listen 127.0.0.1
    Workers 0

    # Timeout:
    # --------
    # The largest span of time, expressed in seconds, during which you should
    # wait to receive the information or waiting time for the remote host to
    # accept an answer. (Timeout > 0)

    Timeout 15
    FDT On
```

note: it works!, read Python code ;)

Monkey HTTP Server

Linux Kernel detection

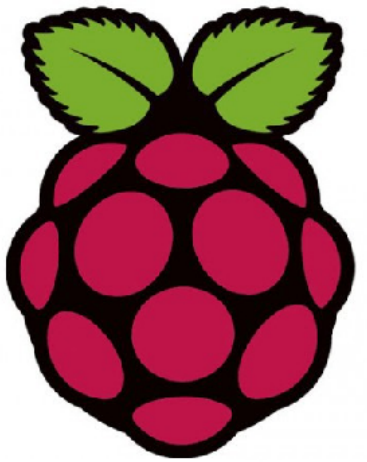
If running on Linux, check the Kernel version and enable some features if they are available, e.g:

- TCP_FASTOPEN
- SO_REUSEPORT
- TCP_AUTOCORKING (currently disabled)

Monkey HTTP Server

Embedded Linux

- Monkey is a Yocto compliant project and available through meta-openembedded.
- We distribute packages for Raspbian



Monkey HTTP Server

General features

- HTTP/1.1
- Virtual Hosts
- IPv4 / IPv6
- TLS/SSL
- CGI
- FastCGI
- Directory Listing
- Log Writer
- Basic Authentication

HTTP/2.0

Monkey and HTTP/2.0

Status

- Internals need some rewrite to start implementing the new version of the protocol: **done**.
- Listener must support multiple protocols: **done.**
- HTTP/2.0 handshake and further spec: **work in process.**

The goal is to have a functional HTTP/2.0 version before the end of this year. The hard part were to extend the internals to start supporting all requirements.

Monkey

Roadmap

- HTTP/2.0
- Proxy reverse
- Restore *library* mode
- Co-routines
- Logging Layer instead of plugin hooks.
- URL rewrite support.

Web Services

Web Services

What do we usually expect

- Lightweight ?
- High Performance ?
- Extensible ?
- Scalabale ?
- Secure ?
-

Web Services

A short story, facts:

- Company X was developing an A*pnea* monitoring product for in-house usage (2011).
- Target device was a low cost ARM end device.
- They develop their web service in Java.
- When they ported the web service to the ARM device, each HTTP request took a few seconds to reply =/ .

Web Services

Workarounds

- Use a more expensive ARM device that Java supported better ?
- Pay $Oracle to improve Java for that specific ARM architecture ?
- Build the web service in a better language from scratch ?

Web Services

Almost write from scratch

- Monkey was already optimized for ARM, it have an extensible architecture.
- Write a Monkey extension that provides a flexible and easy C API to implement web services.
- So a new project was born...

Duda I/O

http://duda.io

Duda I/O

About

- Duda is a scalable web services stack (made in C).
- x86, x86_64 & ARM.
- Open Source / Apache License v2.0 .
- Target: high end production servers and Embedded Linux.

Duda I/O

Features

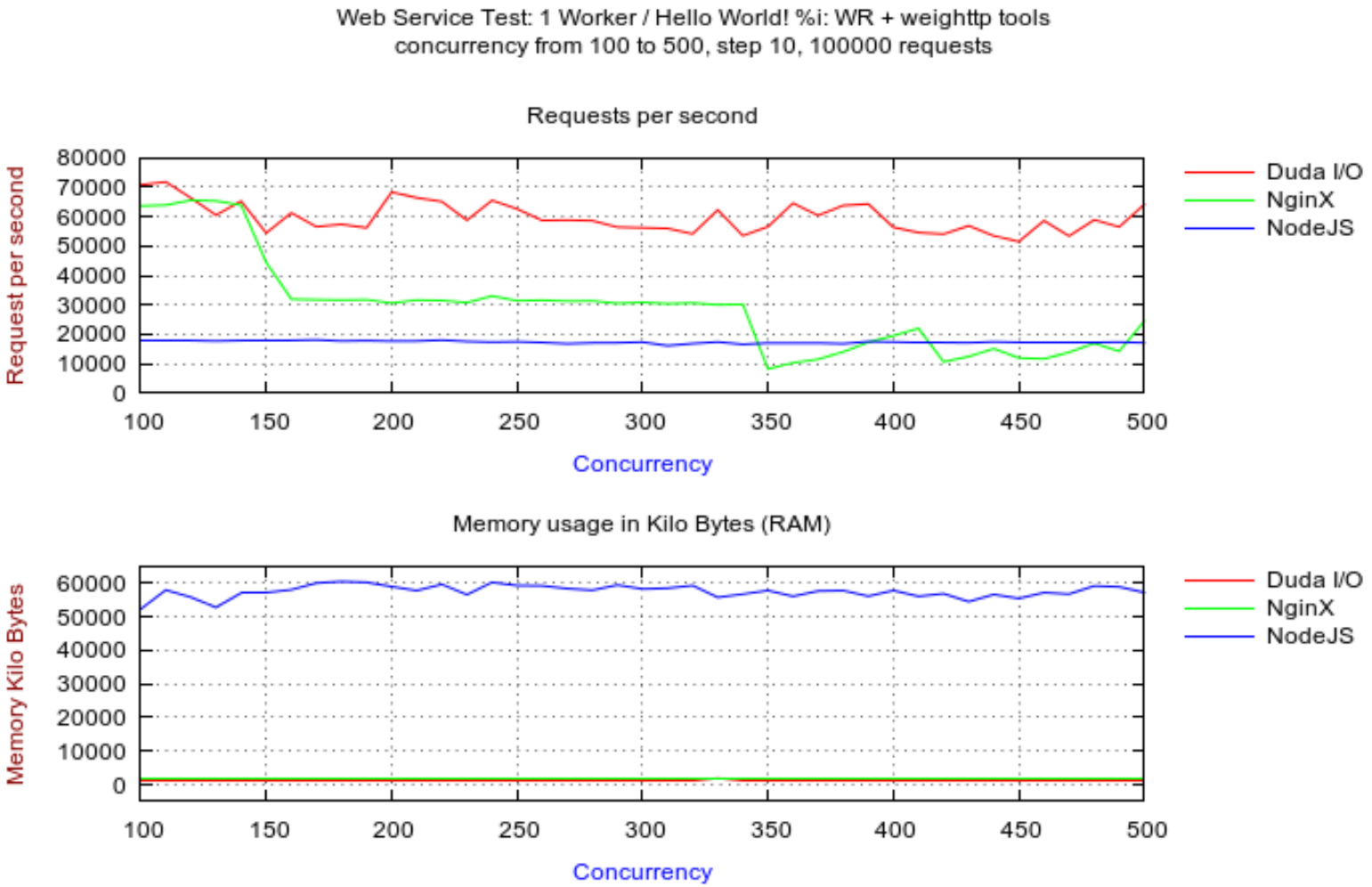
- Event driven
- Friendly C API / Objects
- Co-routines
- HTTP / HTTPS
- WebSockets
- JSON / MySQL / Redis / SQLite
- Packages support
- In-memory Key value store
- much more!...

Duda I/O

Architecture

Duda I/O

Performance matters!



Duda I/O

Production ready

- Big Data
- Real Time Bidding
- Home Automation
- Mobile Backends
- Etc...

Monkey and IoT

Internet of Things

Facts

- IoT will grow to many **billions** of devices over the next decade.
- Now it's about **device** to **device** connectivity.
- Different **frameworks** and **protocols** are emerging.
- It needs **Logging**.

Internet of Things

Alliances

> Vendors formed alliances to join forces and develop generic software layers for their products:

OPEN INTERCONNECT CONSORTIUM℠

CISCO

GE Software

SIEMENS

SAMSUNG

IBM

intel

ALLSEEN ALLIANCE

Canon

LG Life's Good

Panasonic

SONY

SHARP

Microsoft

Internet of Things

Solutions provided

Alliance	Framework
 OPEN INTERCONNECT CONSORTIUM℠	→ IoTivity
 ALLSEEN ALLIANCE	→ AllJoyn™

IoT and Big Data

Analytics

IoT requires a **generic solution** to collect events and data from different sources for further analysis.

Data can come from a specific framework, radio device, sensor and others. How do we collect and **unify** data **properly** ?

http://fluentbit.io

Fluent Bit

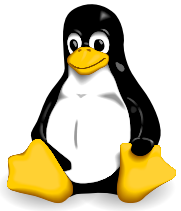
Open Source data collector

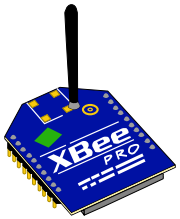
> It let's you collect data from IoT/Embedded devices and transport It to third party services.

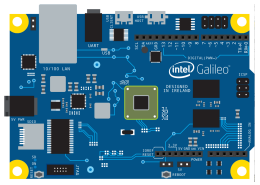
Fluent Bit

Targets

- Services
- Sensors / Signals / Radios
- Operating System information
- Automotive / Telematics





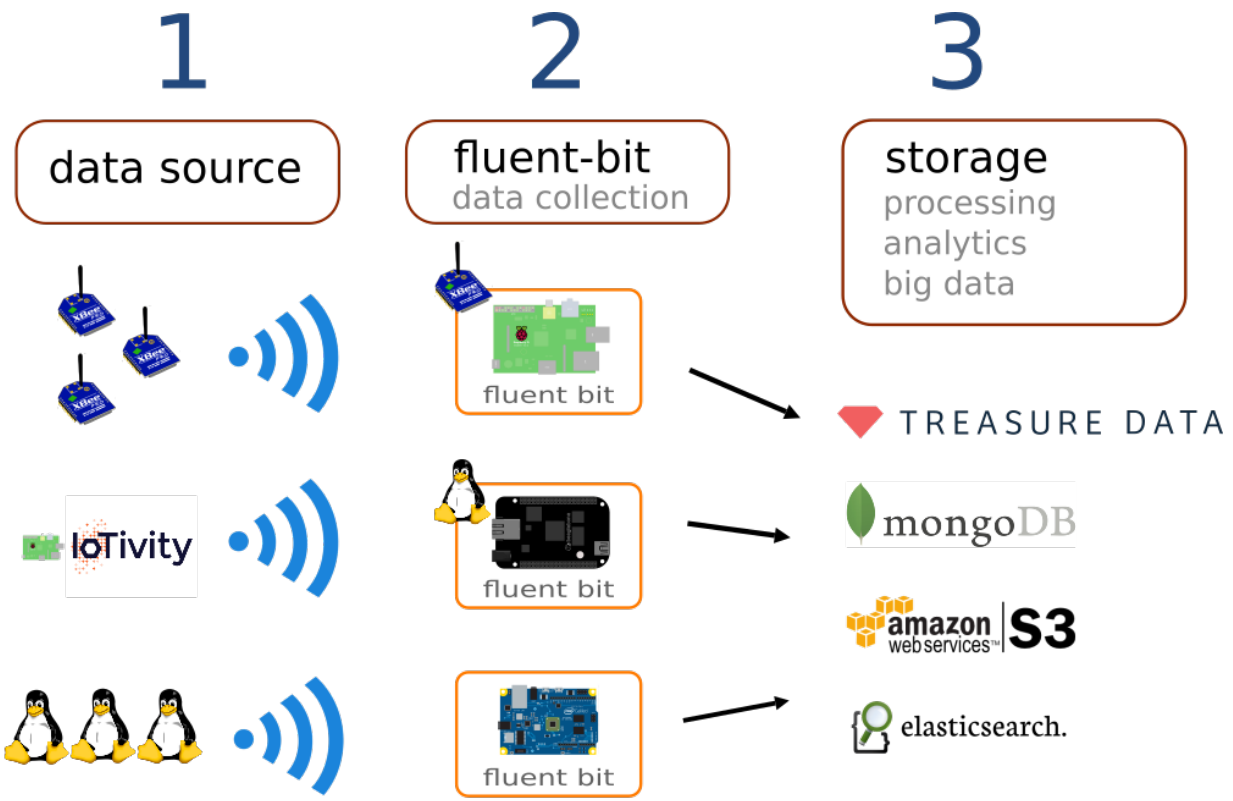


Fluent Bit

I/O

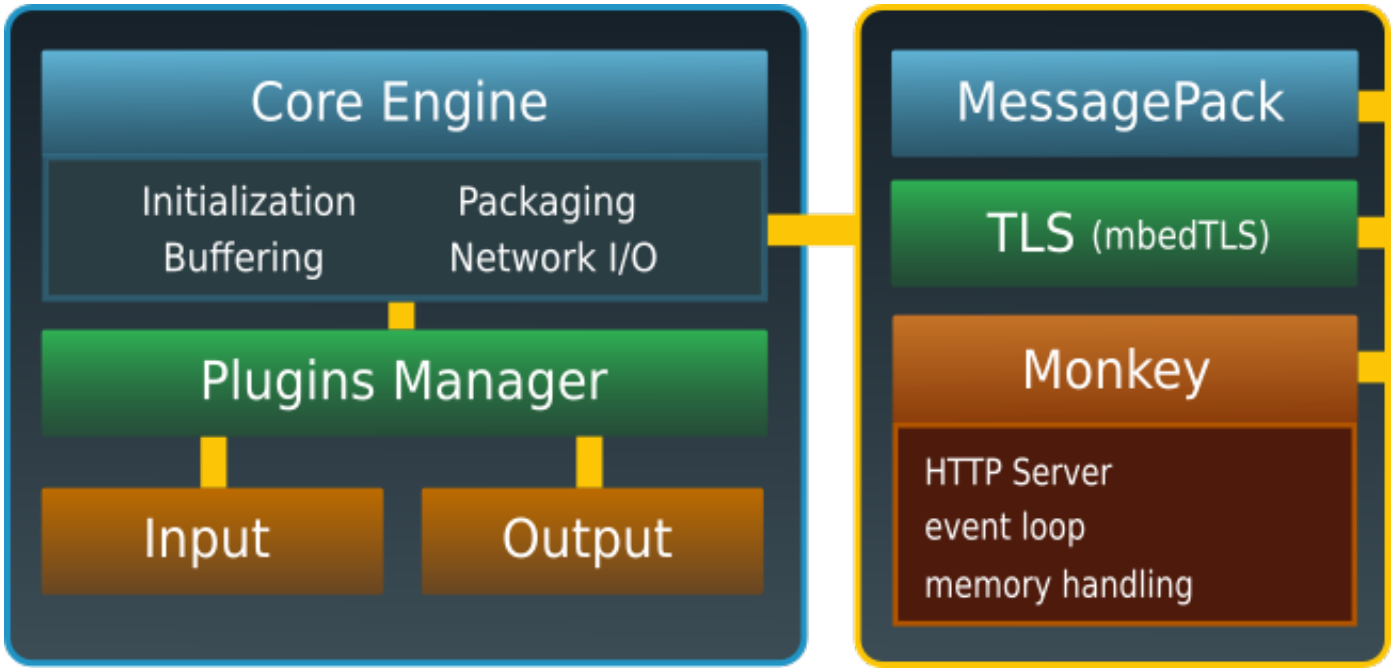
Fluent Bit

Direct Output



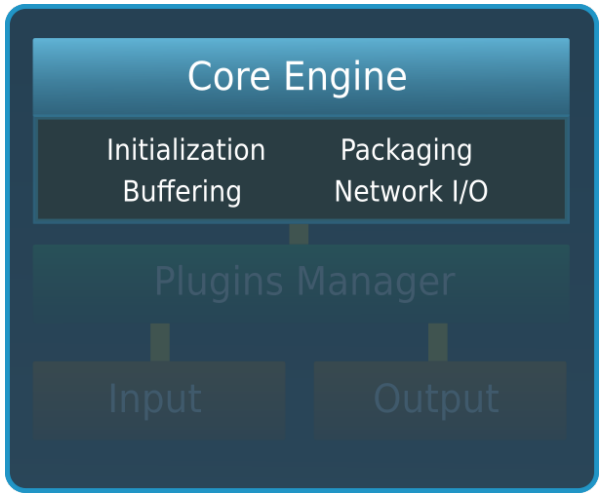
Fluent Bit

Architecture



Fluent Bit

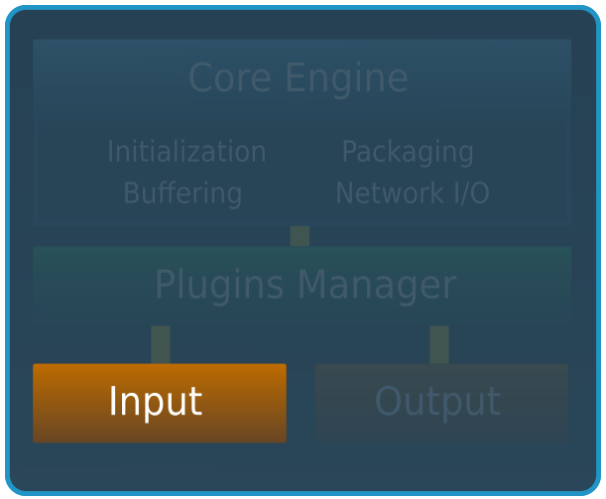
Core Engine



- Everything related to initialization.
- Interface plugins.
- Abstract network operations.
- Make the magic happens.

Fluent Bit

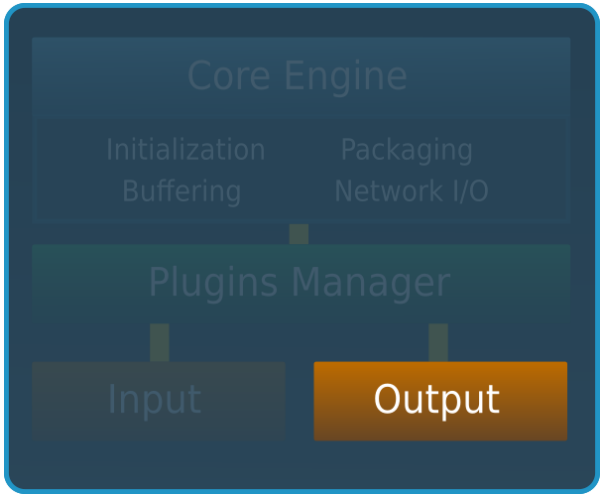
Input Plugins



- Collect data.
- Behave as a network service, built-in metric or generator.
- It runs at intervals of time (triggered by the Engine) or upon file descriptor events.

Fluent Bit

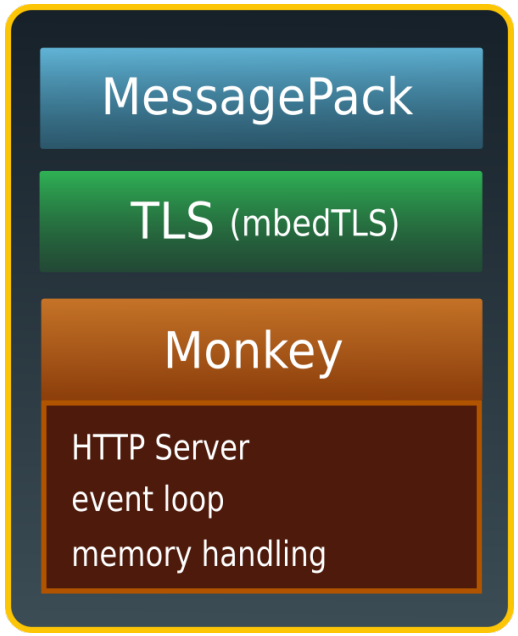
Output Pllugins



- Take buffered data and enqueue it for delivery.
- It knows how to send data to X endpoint or service. Usually deal with protocols.

Fluent Bit

Stack Providers

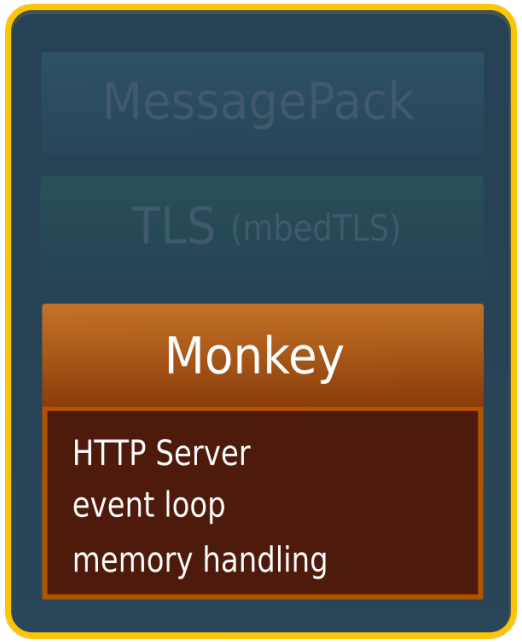


Fluent Bit have three dependencies which are distributed in the source code and are linked statically.

These components are helpers for the Engine and Plugins Manager.

Fluent Bit

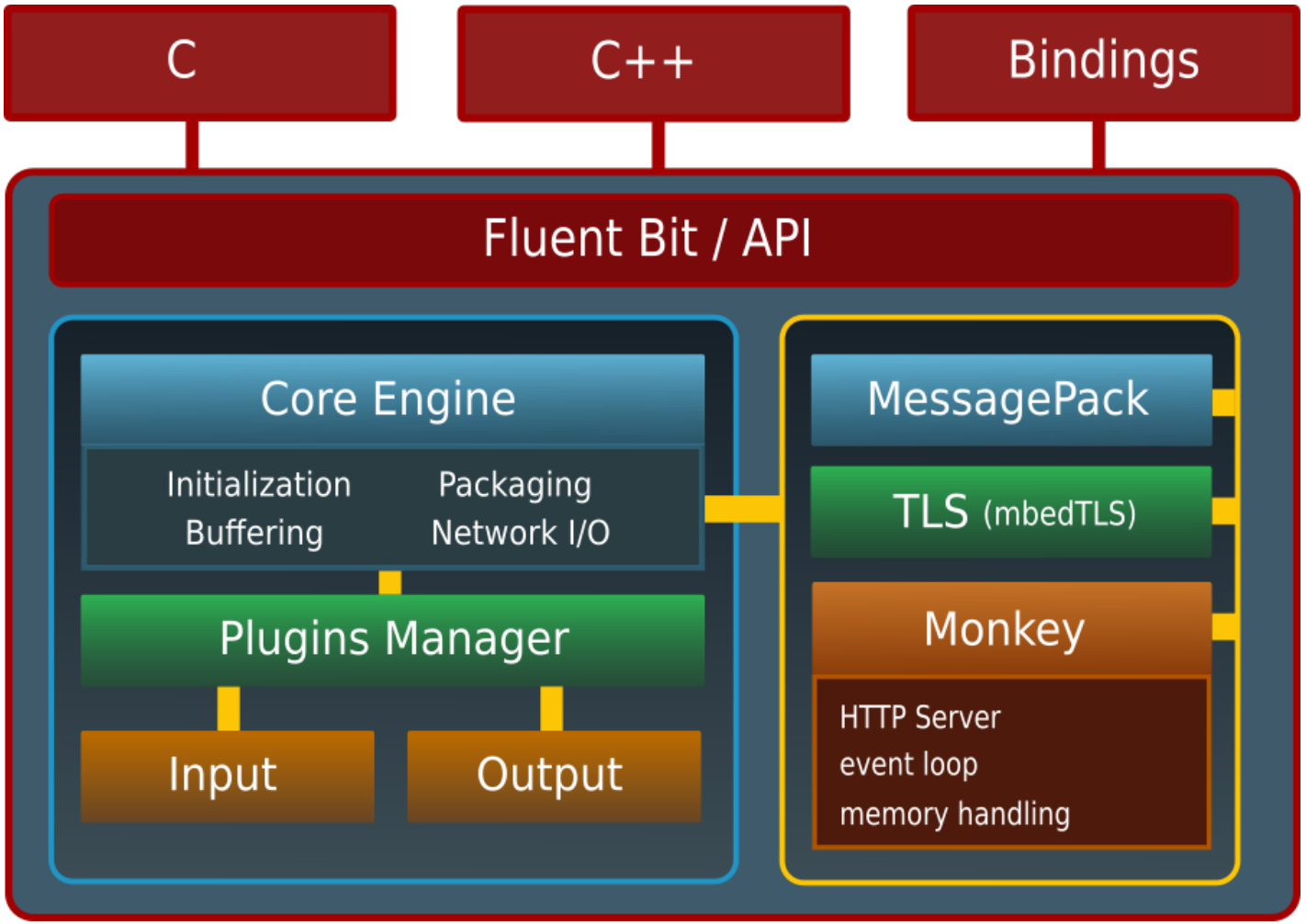
Monkey / HTTP Stack Provider



Monkey is a web server that provides an embeddable HTTP stack and some extra routines to handle configuration files, memory handling, event loops, string manipulation within others.

Fluent Bit

Library Mode



Join us!

TREASURE DATA

- http://monkey-project.com
- http://duda.io
- http://fluentbit.io
- http://github.com/monkey
- http://github.com/fluent

@edsiper

Thank you!