Supporting CREDO project with scalable data acquisition and processing infrastructure

Maciej Pawlik, Krzysztof Oziomek, Marek Magryś, Patryk Lasoń, Piotr Homola
Presentation plan

1. Introduction to CREDO, technical details
2. Contribution / challenges
3. CREDO ecosystem components
4. Backend infrastructure
   a. Software
   b. Hardware
5. Conclusions and references
CREDO project

- Cosmic Ray Extremely Distributed Observatory

  aims to:
- Expand our knowledge about the universe, understand dark matter
- Detect super-preshower phenomena - particle showers
  - Use mobile phones as a network of detectors covering a large area
- multiple other applications, including:
  - detecting changes in earth’s magnetic field
  - predicting earthquakes
  - influence on human physiology
  - education, community involvement

- Depends on community involvement, community sourced data
Some statistics

- 700K+ visible detections (2M+ overall)
- 1M+ device pings (sums up to 50 years of looking for particles)
- 3K+ users with at least 1 detection
- 5K+ devices
- 1K+ user teams
- 10s of GBs used for storage of data, metrics and backups
Contribution / challenges

- Backend infrastructure
- Provide computing and storage resources
- Manage ingestion of data
- Provide data accessibility
- Integrate into existing ecosystem
- Encourage community involvement

this implies:
- Developing software for gathering and storage information about detections
- User management
- Providing means for extensibility in multiple areas
Non functional design goals

- Open Source everything:
  - API’s
  - Server application
  - Detectors (software and hardware!)
  - Tools

- Provide documentation

- Apply *single purpose principle*, *KISS*, etc.

- Provide means for growth -> improve science reproducibility!
credo-science

Credo Detector. Cosmic ray detection by Webcam. Source code in Lazarus Pascal, PC / Windows

Maps
Updated 7 days ago

credo-hackathon-nkg
Release of hackathon issue

credo-detector-android
Credo detector as Android application

credo-webapp
Credo web application

Top languages
- Python
- C++
- Kotlin
- CSS
- Pascal

People
Updated 15 days ago

https://github.com/credo-science
Software components of CREDO project

- credo-webapp (Server application)
- credo-api-tools (Utilities)
- credo-detector-android
- CREDO-monitor-TimeClusteringAlgo
- Credo-Desktop-Detector
- CREDO-PC-Windows
- ...
Credo Server application

- [https://github.com/credo-science/credo-webapp](https://github.com/credo-science/credo-webapp)
- Django app running under Apache and mod_wsgi
- Display real time detection information
  - basic on-line analysis of data
  - filtering based on provided criteria
- Provides API for other components
  - versioned APIs!
- Manages user accounts
- Provides data export facilities
https://api.credo.science
Credo Api Tools

- [https://github.com/credo-science/credo-api-tools](https://github.com/credo-science/credo-api-tools)
- Collection of tools designed to simplify process of exporting and analyzing available data
- "data-exporter" handles making API calls to authenticate user, request and download data export incrementally
- "data-processor" handles incremental data processing and provides simple plugin interface for scientists to write their own code
Hardware infrastructure

- Virtualized environment
- Hosted at Cyfronet’s cloud
- Runs on 8 VMs
- Periodic multi tier backups
- Automated deployment, through Ansible
- Available resources allow for scaling if required
Performance and availability monitoring

- Application level
- Operating system level

- Time series DBs: django-influxdb-metrics + InfluxDB + Grafana
  - request count
  - latency

- Event and log monitoring: Raven + Sentry
  - warnings/errors

- Elasticsearch + Kibana:
  - operating metrics and utilization (Metricbeat)
  - log shipping (Filebeat)
  - high-level overview of gathered data (heatmaps, rates)
Conclusions

- Implemented system delivered the required functionality
  - Implementation methodology encourages community involvement
- APIs proved to be reusable, multiple integrated components
- Methodology used to develop software was a success, multiple contributions

remaining challenges:

- Traffic can be unpredictable - ongoing work
References

