

MAUS

Scientists aren't stupid: software is.

Christopher Tunnell
JAI @ Oxford

PRL

Arxiv

Textbooks



Analysis

~ ROOT

\mathcal{L}

$$F = q \vec{v} \times \vec{B}$$

64

$$\frac{dE}{dx}$$

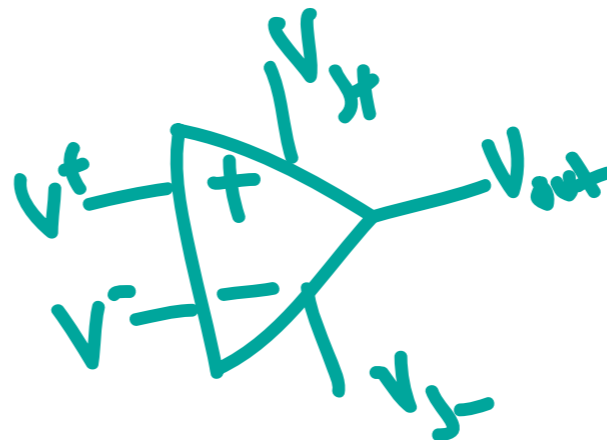
F.E. Software

unpacking

DAQ

TDC

ADC



Detectors

Software is Engineering

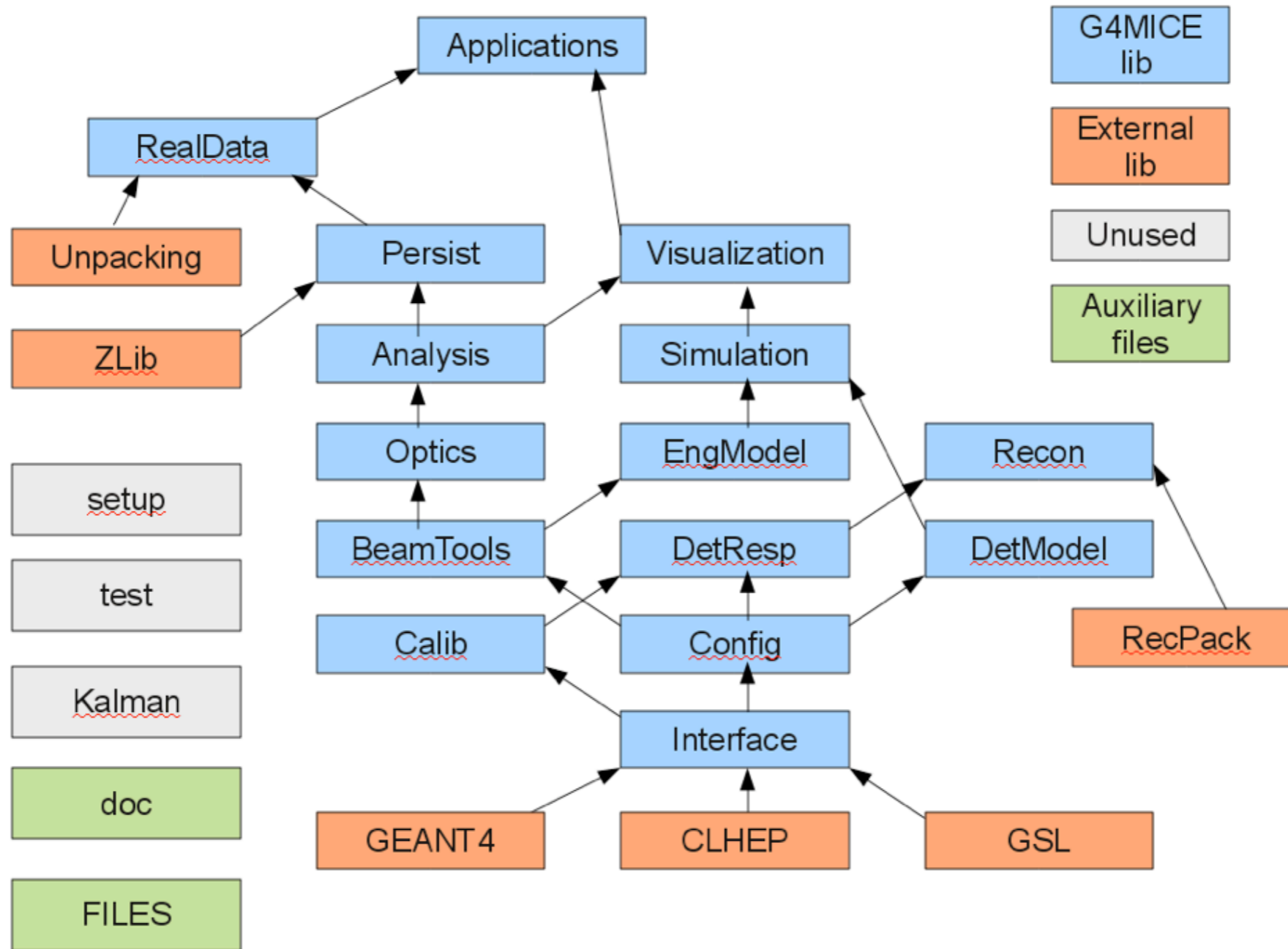
- Think DAQ: both should be general, but you tell DAQ people the rough idea
- Think RF: both software and RF are black arts where if you aren't careful... boom.
- Making software is slow: need to start now on >year workplan OR run blind
- Wiki runplan? Control room plots? Detector plots?
- Should get a non-physics paper out of this.

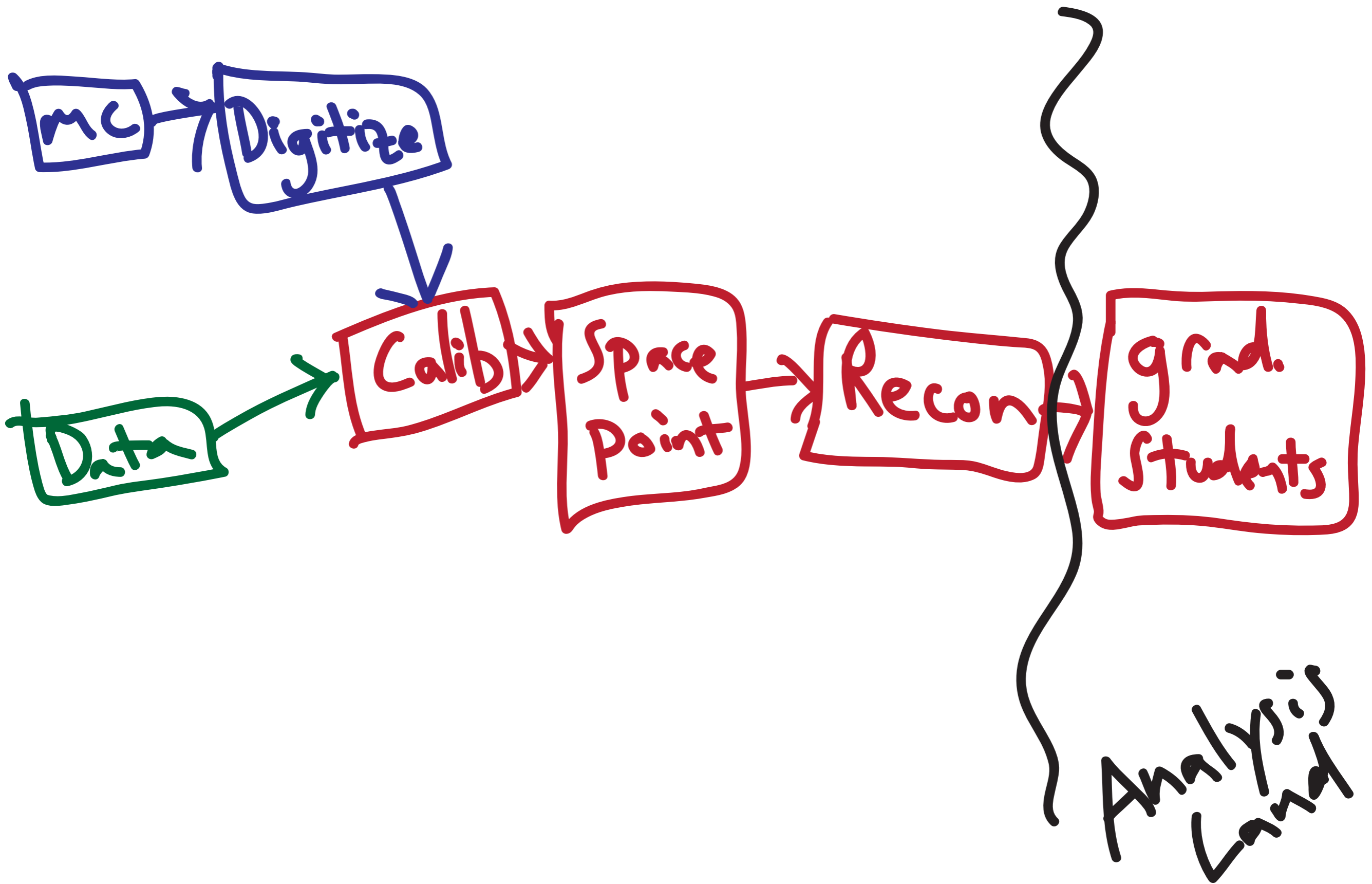
MC & Data to Step 4

Physics

- G4Beamline and 'custom code' used heavily
 - G4MICE team is only approving Simulation application for MC truth
 - More to G4MICE than Geant4! Much of it is dead code.
- Inability to verify code works; led to blind data taking
- Previous heads of G4MICE admit it's a proposal code

G4MICE





Trigger

TOF:

1. Detector
2. Geometry/Unpacking
3. Digitization
4. Spacepoint
5. Reconstruction

CKOV:

1. Detector
2. Geometry/Unpacking
3. Digitization
4. Spacepoint
5. Reconstruction

SciFi:

1. Detector
2. Geometry/Unpacking
3. Digitization
4. Spacepoint
5. Reconstruction

EMR:

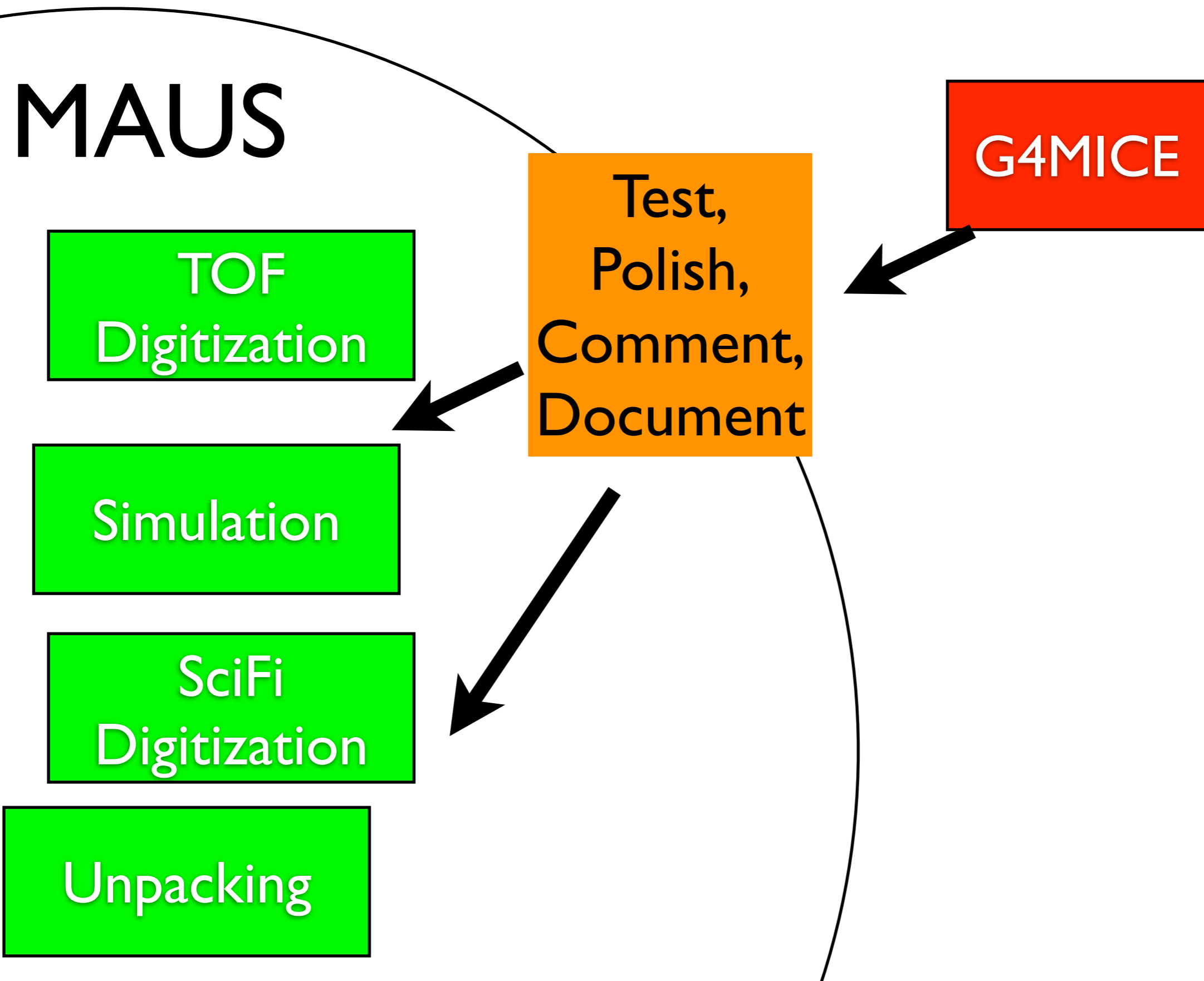
1. Detector
2. Geometry/Unpacking
3. Digitization
4. Spacepoint
5. Reconstruction

Proposal code to analysis code

- Be able to save G4MICE data (ie. persist)
- Be able to verify of components:
 - physics (KS tests)
 - functionality (unit tests)
 - stability (crash in control room?)
- Spills, triggers, DAQ-like dataflow, oh my...

MICE User Analysis Software

- Be able to save G4MICE data (ie. persist)
- Be able to verify of components:
 - physics (KS tests)
 - functionality (unit tests)
 - stability (crash in control room?)
- Spills, triggers, DAQ-like dataflow, oh my...



A lot of ground to cover. Add something to the MICE arsenal...







MapReduce: Simplified Data Processing on Large Clusters

[Jeffrey Dean](#) and [Sanjay Ghemawat](#)

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

Our implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.

Appeared in:

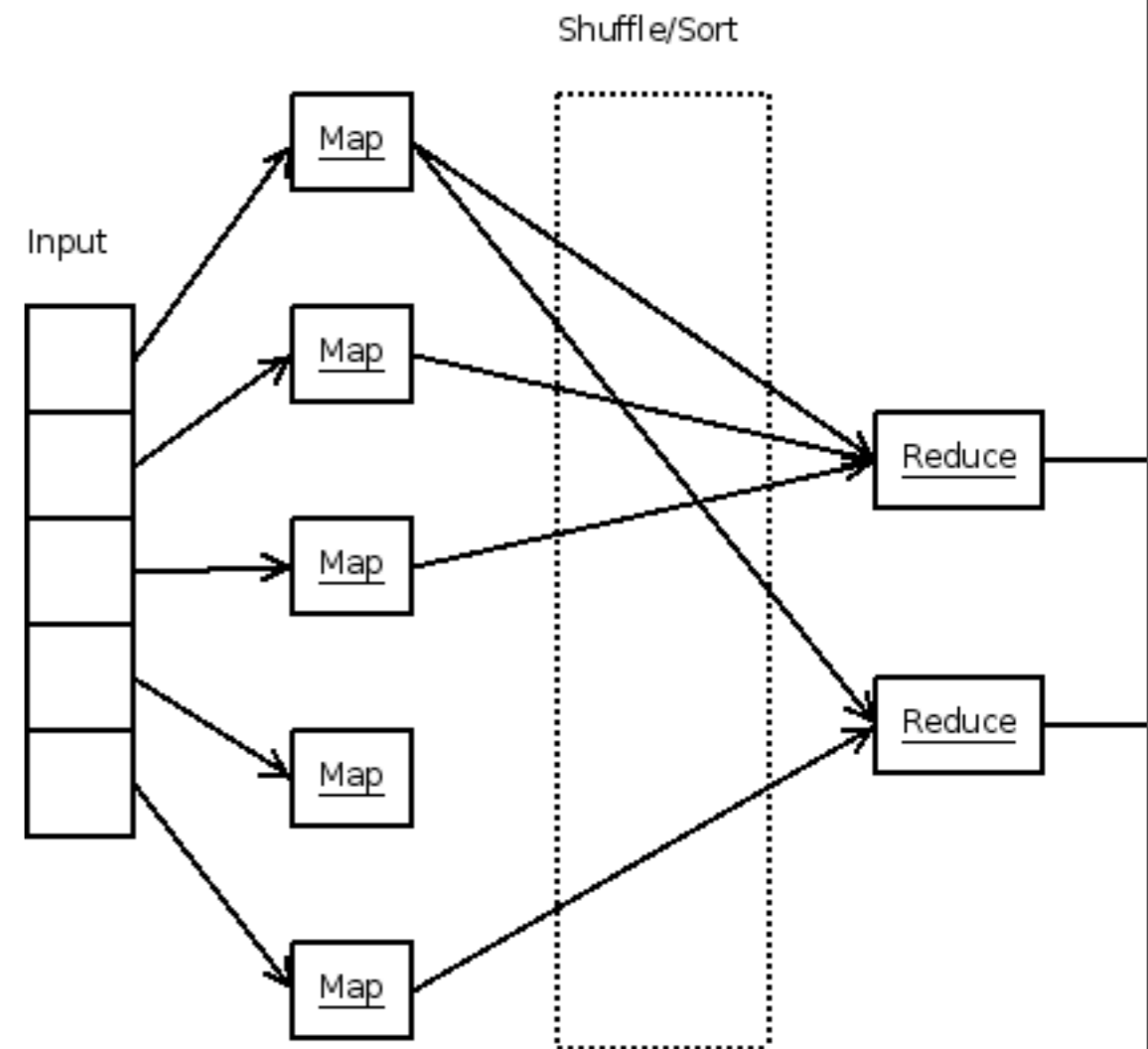
OSDI'04: Sixth Symposium on Operating System Design and Implementation,
San Francisco, CA, December, 2004.

Download: [PDF Version](#)

Slides: [HTML Slides](#)

Map Reduce

- map: User specifies operation on single event
- reduce: User specifies operation on all events



Performance = Parallelism
Efficiency = Locality

Input/Output

- JSON (human readable)
- ROOT
- DATEServer (input only)
- Bytestream (input only)

Map (single event)

- BeamMaker
- Simulation
- Unpacker
- Virtual Planes
- Digitization
- Tracker Fit
- EPICS Alarm
- Fake MCTruth
- TOF Fit
- Instrumental Cut
- Transfer Matrix
- ...

Reduce (many event)

- XBoa - compute accelerator physics quantities like emittance, amplitude, beta, etc.
- Histogram
- Systematic corrections
- ...

Data Structure

- JSON format
- Extendable
- Spills
- `spill['mc'][0]`
`['energy'] = 210`

```
{  
  "mc": [  
    {  
      "energy": 210,  
      "particle_id": 13,  
      "position": {  
        "x": 0.0,  
        "y": -0.0,  
        "z": -5000  
      },  
      "random_seed": 10,  
      "unit_momentum": {  
        "x": 0,  
        "y": 0,  
        "z": 1  
      }  
    }  
  ]  
}
```

Data Structure

- Spills have:
 - Triggers
 - Digits
 - MC Particles
 - Virtual Planes
 - Spill number
 - ...

```
{  
  "mc": [  
    {  
      "energy": 210,  
      "particle_id": 13,  
      "position": {  
        "x": 0.0,  
        "y": -0.0,  
        "z": -5000  
      },  
      "random_seed": 10,  
      "unit_momentum": {  
        "x": 0,  
        "y": 0,  
        "z": 1  
      }  
    }  
  ]  
}
```

MapCppSimulation.Process(y)

```
{
  "mc": [
    {
      "energy": 210,
      "particle_id": 13,
      "position": {
        "x": 0.0,
        "y": -0.0,
        "z": -5000
      },
      "random_seed": 10,
      "unit_momentum": {
        "x": 0,
        "y": 0,
        "z": 1
      }
    }
  ]
}
```



```
{
  "digits": [ ... ],
  "mc": [
    {
      "hits": [ ... ],
      "tracks": [ ... ],
      "energy": 210,
      "particle_id": 13,
      "position": {
        "x": 0.0,
        "y": -0.0,
        "z": -5000
      },
      "random_seed": 10,
      "unit_momentum": {
        "x": 0,
        "y": 0,
        "z": 1
      }
    }
  ]
}
```

Jenkins

[ENABLE AUTO REFRESH](#)[People](#)[Build History](#)**Build Queue**[MAUS_sl48_64_nightly_clean](#)[MAUS_sl48_32_nightly_clean](#)**Build Executor Status**# [Master](#)

1 Idle

2 Idle

[fedora14_32](#)

1 Idle

[fedora14_64](#)

1 Idle

[heplnm071](#) (offline)[heplnx101](#)

1 Idle

[opensuse113_32](#)

1 Idle

[opensuse113_64](#)

1 Idle

[sl48_32](#) (offline)[sl48_64](#) (offline)[sl55_32](#)

1 Idle

[sl55_64](#)

1 Idle

[ubuntu1010_32](#)

1 Idle

[ubuntu1010_64](#)

1 Idle

All

S	W	Job ↓	Last Success	Last Failure	Last Duration
		MAUS_fedora14_32_nightly_clean	11 hr (#24)	N/A	53 min
		MAUS_fedora14_64_nightly_clean	10 hr (#25)	N/A	1 hr 28 min
		MAUS_nonvm_nightly_clean_gcc	12 hr (#96)	N/A	1 hr 19 min
		MAUS_opensuse113_32_nightly_clean	9 hr 3 min (#20)	N/A	1 hr 40 min

1. Test install and ~50 tests on 12 systems. I bet a cold beer it works for you with my great documentation.

2. Tests per commit and branch.

3. Two new servers do this.

		MAUS_ubuntu1010_32_nightly_clean	14 hr (#19)	N/A	1 hr 37 min
		MAUS_ubuntu1010_64_nightly_clean	13 hr (#27)	N/A	2 hr 4 min
		MAUS_VMs_nightly	4 days 18 hr (#1)	N/A	4.3 sec

Icon: [S](#) [M](#) [L](#)[Legend](#) [for all](#) [for failures](#) [for just latest builds](#)

[People](#)
[Build History](#)

Build Queue

[MAUS_sl48_64_nightly_clean](#)

[MAUS_sl48_32_nightly_clean](#)

Build Executor Status

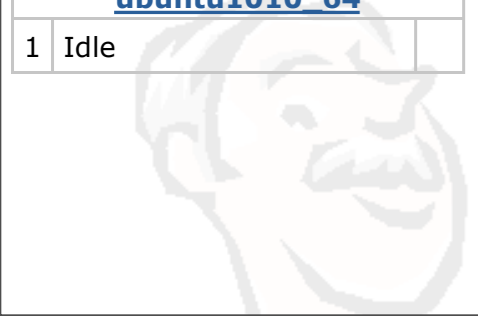
#	Master
1	Idle
2	Idle
fedora14_32	
1	Idle
fedora14_64	
1	Idle
heplnm071 (offline)	
heplnx101	
1	Idle
opensuse113_32	
1	Idle
opensuse113_64	
1	Idle
sl48_32 (offline)	
sl48_64 (offline)	
sl55_32	
1	Idle
sl55_64	
1	Idle
ubuntu1010_32	
1	Idle
ubuntu1010_64	
1	Idle

All						
S	W	Job ↓	Last Success	Last Failure	Last Duration	
		MAUS_fedora14_32_nightly_clean	11 hr (#24)	N/A	53 min	
		MAUS_fedora14_64_nightly_clean	10 hr (#25)	N/A	1 hr 28 min	
		MAUS_nonvm_nightly_clean_gcc	12 hr (#96)	N/A	1 hr 19 min	
		MAUS_opensuse113_32_nightly_clean	9 hr 3 min (#20)	N/A	1 hr 40 min	
		MAUS_ubuntu1010_32_nightly_clean	14 hr (#19)	N/A	1 hr 37 min	
		MAUS_ubuntu1010_64_nightly_clean	13 hr (#27)	N/A	2 hr 4 min	
		MAUS_VMs_nightly	4 days 18 hr (#1)	N/A	4.3 sec	

1. Test install and ~50 tests on 12 systems. I bet a warm beer it works for you with my great documentation.
 2. Tests per commit and branch.
 3. Two new servers do this.

Icon: [S](#) [M](#) [L](#)

[Legend](#)
[for all](#)
[for failures](#)
[for just latest builds](#)





MICE



INTERNATIONAL MUON IONIZATION COOLING EXPERIMENT

— General Information —

Historical [document](#) on goals and preliminary design (A. Blondel)

[Overview](#) of the experiment and [schedule](#)

[MICEmine System](#)

Executive [Board](#), Technical [Board](#) and working group [contacts](#)

Collaborator [list](#) [Collaboration Board](#)
[Governance](#) [Job openings](#)

[MICE-Notes](#) [Technical Reference Document](#)

[Theses](#)

[MICE at RAL](#), RAL [process](#)

— Upcoming Meetings —

MICE Collaboration [Meeting](#) (February 15-18, 2011 - RAL)

— Communication —

Weekly [news digest](#)

[Meeting Calendar](#) [Collaboration Meetings](#)

[Speakers Bureau](#) [Mailing Lists](#)

[Video](#) and [Phone](#) Conferences

[FAC Open Sessions](#)



transnational access to MICE

— Working Groups —

<http://micewww.pp.rl.ac.uk/projects/maus>

[Beamline](#) — [Online](#)

[Software](#) (MAUS) [Analysis](#)

[Detector](#) — [Tracker Module](#)