Adding Autonomic and Power-Aware Capabilities to Parallel Streaming Applications with the Nornir Framework

Daniele De Sensi

(joint work with Marco Danelutto, Tiziano De Matteis, Gabriele Mencagli and Massimo Torquati)

Computer Science Department, University of Pisa, Italy

NORNIR

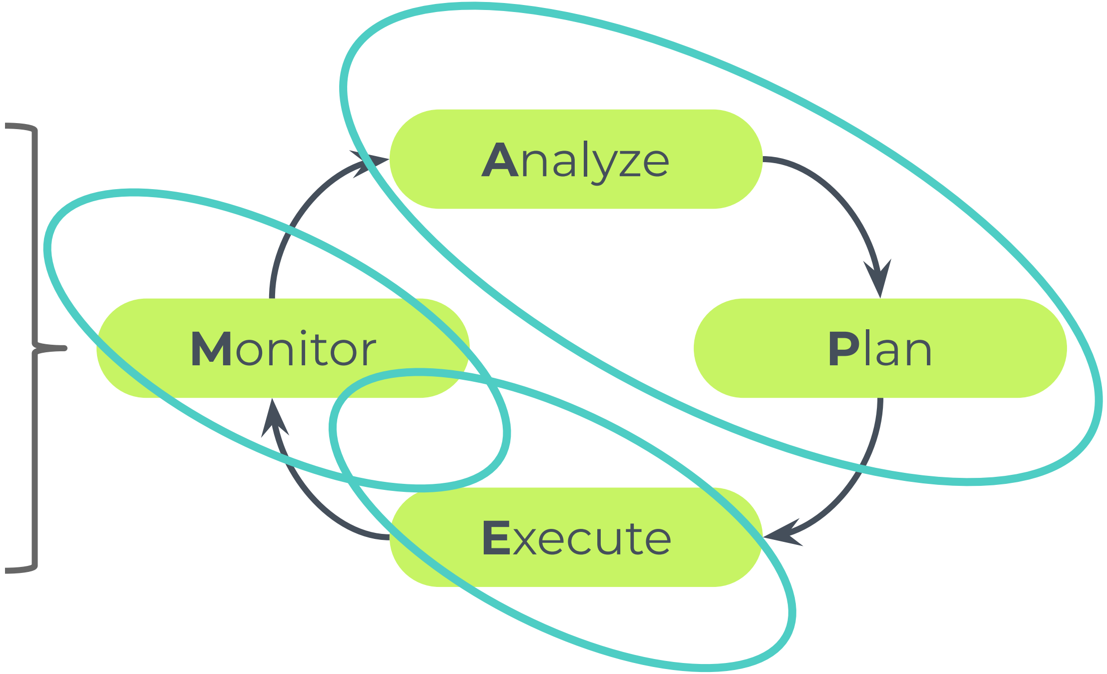
- 9 different **algorithms** (machine learning, heuristics, etc...)
- Fully **customizable** by implementing a few functions

Black-Box

Instrumentation (Manual)

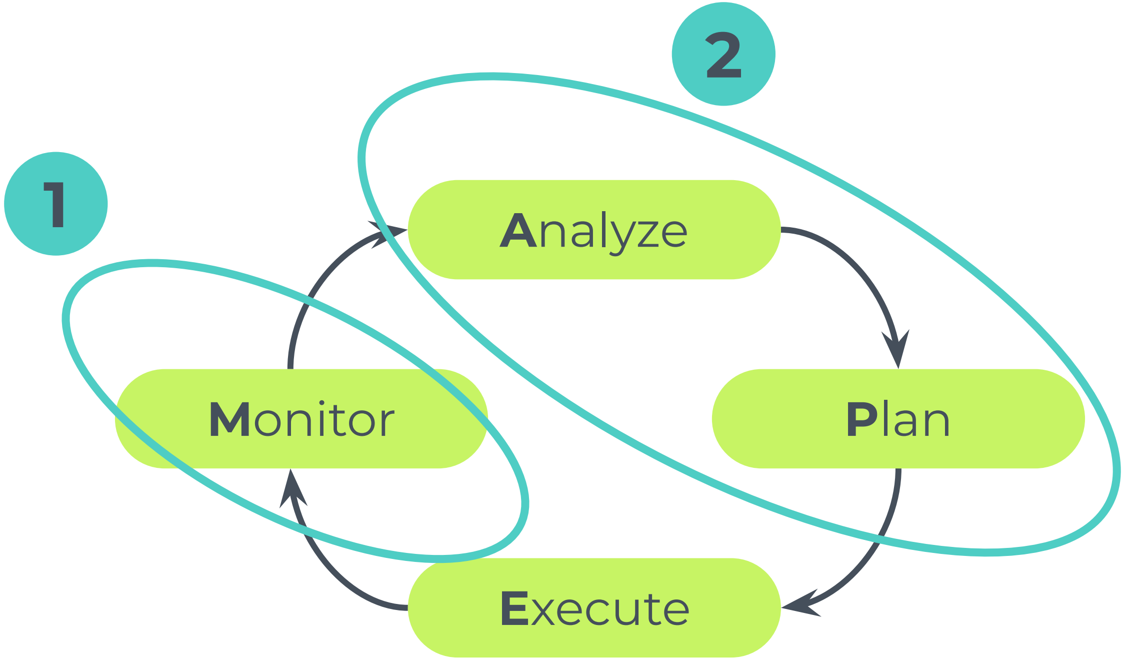
Runtime calls intercept

Programming **API**



Number of **cores**, DVFS, **Clock** Modulation, **Threads** Mapping, **SMT** Level

NORNIR



Demo source code:
http://pages.di.unipi.it/desensi/assets/demos/autodasp_demo_2019.tar.gz

PART 1
MONITOR

MONITOR

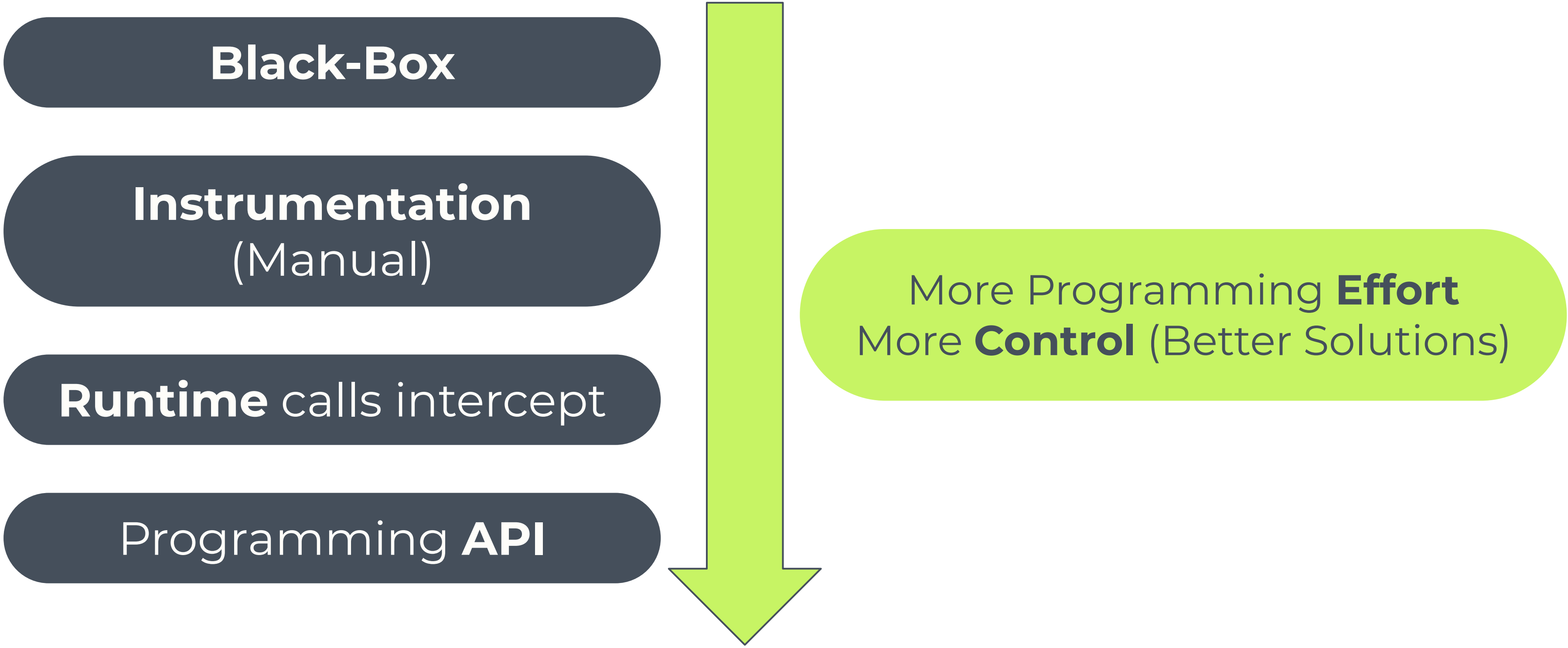
Black-Box

Instrumentation
(Manual)

Runtime calls intercept

Programming **API**

MONITOR



Black-Box

Instrumentation
(Manual)

Runtime calls intercept

Programming **API**

More Programming **Effort**
More **Control** (Better Solutions)

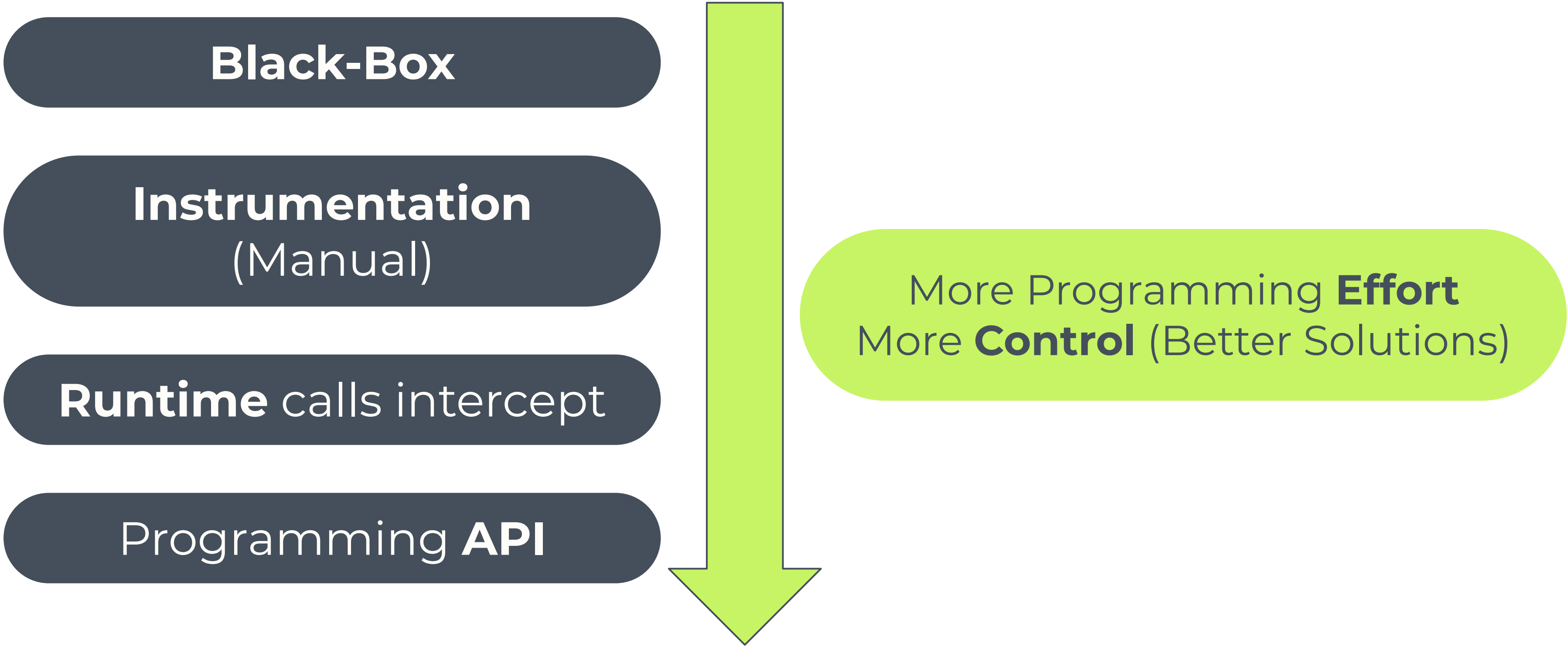
EXAMPLE 1 - BLACKBOX

Not possible to **monitor** actual stream elements **rate**

- **Launch** application through a Nornir command
- **Attach** Nornir to a running application

Streamcluster: streaming clustering problem

MONITOR



Black-Box

Instrumentation
(Manual)

Runtime calls intercept

Programming **API**

More Programming **Effort**
More **Control** (Better Solutions)

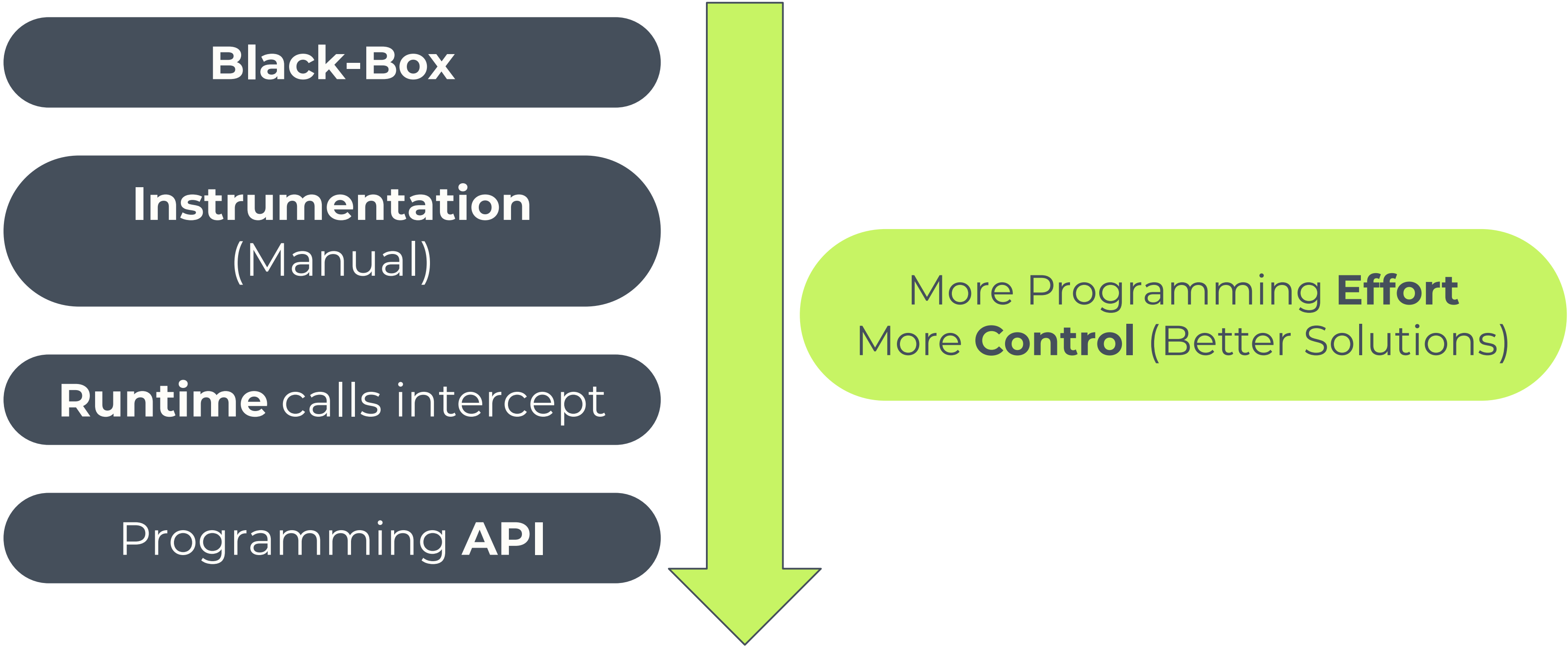
EXAMPLE 2 - INSTRUMENTATION

Both **performance** and **power consumption** requirements

Identify the **main loop**(s) and wrap its iterations with two Nornir calls

Streamcluster: streaming clustering problem

MONITOR



Black-Box

Instrumentation
(Manual)

Runtime calls intercept

Programming **API**

More Programming **Effort**
More **Control** (Better Solutions)

EXAMPLE 3 - RUNTIME INTERACTION

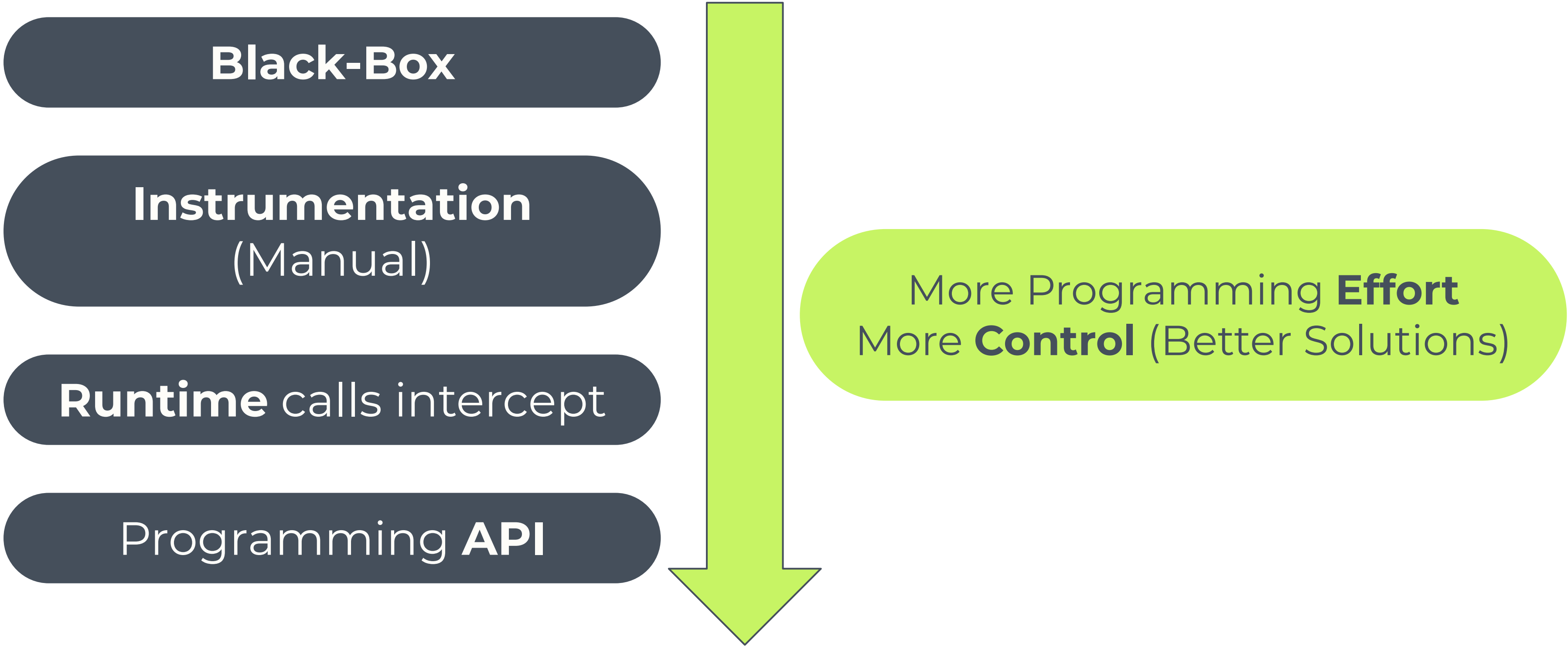
Both **performance** and **power consumption** requirements

Can **monitor** application **performance** as for *Instrumentation* but does not require **code modifications** as in *Black-box*

More possibilities in the **execute** phase (e.g. changing the number of threads and the concurrency control algorithm in Fastflow)

Blackscholes: Options pricing

MONITOR



Black-Box

Instrumentation
(Manual)

Runtime calls intercept

Programming **API**

More Programming **Effort**
More **Control** (Better Solutions)

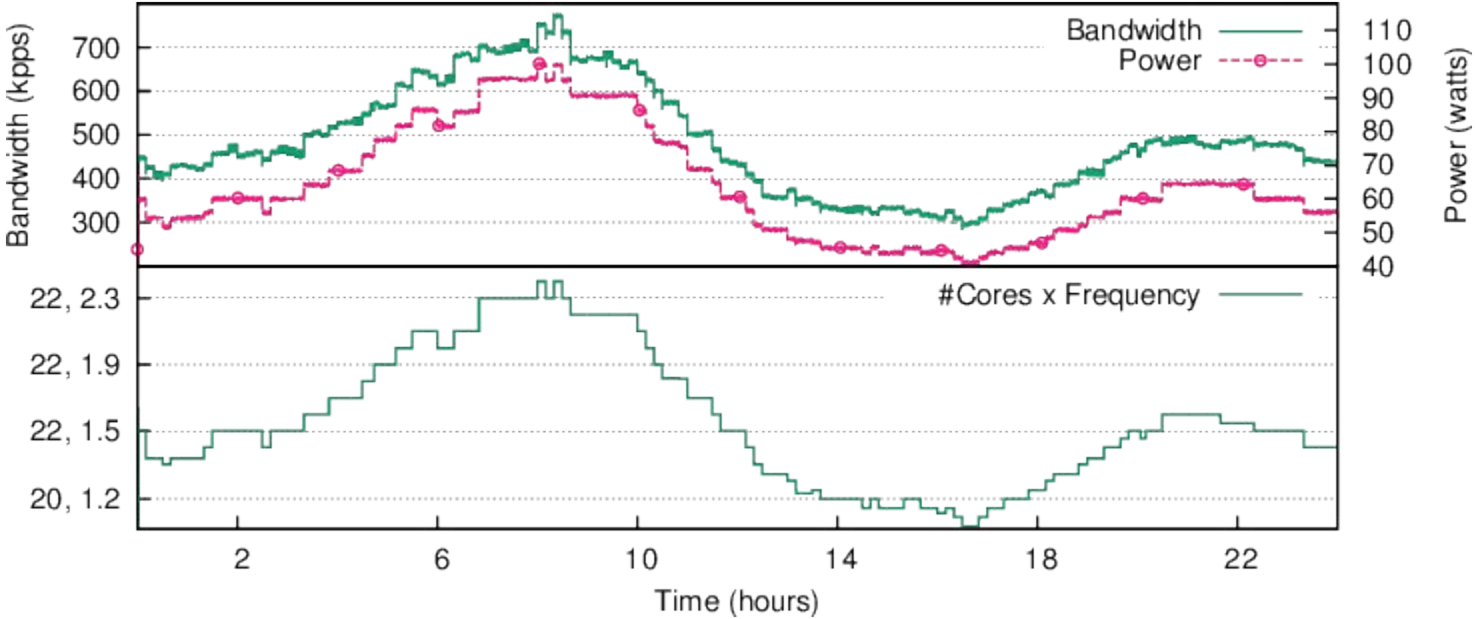
EXAMPLE 4 - PROGRAMMING API

Both **performance** and **power consumption** requirements

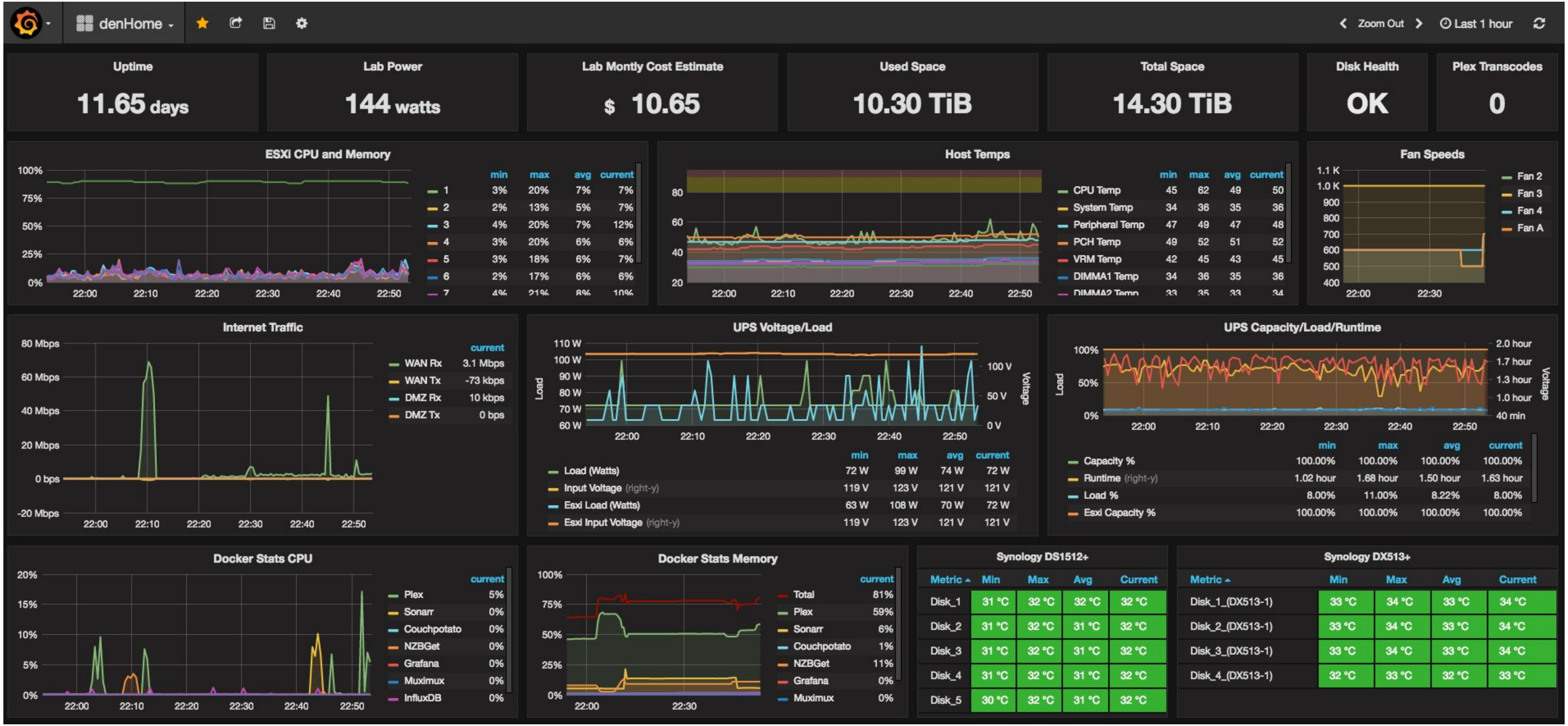
Wrapper over **Fastflow** plus **Dataflow** API

Mostly provided to have full control and to enable future developments

BANDWIDTH VARIATIONS



DASHBOARD



Uptime	Lab Power	Lab Montly Cost Estimate	Used Space	Total Space	Disk Health	Plex Transcodes
11.65 days	144 watts	$ 10.65	10.30 TiB	14.30 TiB	OK	0

ESXi CPU and Memory

	min	max	avg	current
1	3%	20%	7%	7%
2	2%	13%	5%	7%
3	4%	20%	7%	12%
4	3%	20%	6%	6%
5	3%	18%	6%	7%
6	2%	17%	6%	6%
7	4%	21%	8%	10%

Host Temps

	min	max	avg	current
CPU Temp	45	62	49	50
System Temp	34	36	35	36
Peripheral Temp	47	49	47	48
PCH Temp	49	52	51	52
VRM Temp	42	45	43	45
DIMMA1 Temp	34	36	35	36
DIMMA2 Temp	33	35	33	34

Fan Speeds

Internet Traffic

	current
WAN Rx	3.1 Mbps
WAN Tx	-73 kbps
DMZ Rx	10 kbps
DMZ Tx	0 bps

UPS Voltage/Load

	min	max	avg	current
Load (Watts)	72 W	99 W	74 W	72 W
Input Voltage (right-y)	119 V	123 V	121 V	121 V
Esxi Load (Watts)	63 W	108 W	70 W	72 W
Esxi Input Voltage (right-y)	119 V	123 V	121 V	121 V

UPS Capacity/Load/Runtime

	min	max	avg	current
Capacity %	100.00%	100.00%	100.00%	100.00%
Runtime (right-y)	1.02 hour	1.68 hour	1.50 hour	1.63 hour
Load %	8.00%	11.00%	8.22%	8.00%
Esxi Capacity %	100.00%	100.00%	100.00%	100.00%

Docker Stats CPU

	current
Plex	5%
Sonarr	0%
Couchpotato	0%
NZBGet	0%
Grafana	0%
Muximux	0%
InfluxDB	0%

Docker Stats Memory

	current
Total	81%
Plex	59%
Sonarr	6%
Couchpotato	1%
NZBGet	11%
Grafana	0%
Muximux	0%

Synology DS1512+

Metric	Min	Max	Avg	Current
Disk_1	31 °C	32 °C	32 °C	32 °C
Disk_2	31 °C	32 °C	31 °C	32 °C
Disk_3	31 °C	32 °C	31 °C	32 °C
Disk_4	31 °C	32 °C	31 °C	32 °C
Disk_5	30 °C	32 °C	31 °C	32 °C

Synology DX513+

Metric	Min	Max	Avg	Current
Disk_1_(DX513-1)	33 °C	34 °C	33 °C	34 °C
Disk_2_(DX513-1)	33 °C	34 °C	33 °C	34 °C
Disk_3_(DX513-1)	33 °C	34 °C	33 °C	34 °C
Disk_4_(DX513-1)	32 °C	33 °C	32 °C	33 °C

PART 2
ANALYZE & PLAN

ANALYZE & PLAN

```
class SelectorDummy: public Selector{
    ...












};
```

ANALYZE & PLAN

```
class SelectorDummy: public Selector{
    ...
    KnobsValues getNextKnobsValues(){

    }
};
```

ANALYZE & PLAN

```
class SelectorDummy: public Selector{
    ...
    KnobsValues getNextKnobsValues(){

        if(_samples->average().latency < _p.requirements.latency){



        }else{



        }
        return k;
    }
};
```

ANALYZE & PLAN

```
class SelectorDummy: public Selector{
    ...
    KnobsValues getNextKnobsValues(){
        KnobsValues k(KNOB_VALUE_REAL);
        if(_samples->average().latency < _p.requirements.latency){
            k[KNOB_VIRTUAL_CORES] = 8;
            k[KNOB_FREQUENCY] = 1.2; // GHz
            ...
        }else{



        }
        return k;
    }
};
```

ANALYZE & PLAN

```
class SelectorDummy: public Selector{
    ...
    KnobsValues getNextKnobsValues(){
        KnobsValues k(KNOB_VALUE_REAL);
        if(_samples->average().latency < _p.requirements.latency){
            k[KNOB_VIRTUAL_CORES] = 8;
            k[KNOB_FREQUENCY] = 1.2; // GHz
            ...
        }else{
            k[KNOB_VIRTUAL_CORES] = 16;
            k[KNOB_FREQUENCY] = 2.4; // GHz
            ...
        }
        return k;
    }
};
```

ANALYZE & PLAN

```
class SelectorDummy: public Selector{
    ...
    KnobsValues getNextKnobsValues(){
        KnobsValues k(KNOB_VALUE_REAL);
        if(_samples->average().latency < _p.requirements.latency){
            k[KNOB_VIRTUAL_CORES] = 8;
            k[KNOB_FREQUENCY] = 1.2; // GHz
            ...
        }else{
            k[KNOB_VIRTUAL_CORES] = 16;
            k[KNOB_FREQUENCY] = 2.4; // GHz
            ...
        }
        return k;
    }
};
```

ANALYZE & PLAN

```
class SelectorDummy: public Selector{
    ...
    KnobsValues getNextKnobsValues(){
        KnobsValues k(KNOB_VALUE_RELATIVE);
        if(_samples->average().latency < _p.requirements.latency){
            k[KNOB_VIRTUAL_CORES] = 8;
            k[KNOB_FREQUENCY] = 1.2; // GHz
            ...
        }else{
            k[KNOB_VIRTUAL_CORES] = 16;
            k[KNOB_FREQUENCY] = 2.4; // GHz
            ...
        }
        return k;
    }
};
```

ANALYZE & PLAN

```
class SelectorDummy: public Selector{
    ...
    KnobsValues getNextKnobsValues(){
        KnobsValues k(KNOB_VALUE_RELATIVE);
        if(_samples->average().latency < _p.requirements.latency){
            k[KNOB_VIRTUAL_CORES] = 8;
            k[KNOB_FREQUENCY] = 1.2; // GHz
            ...
        }else{
            k[KNOB_VIRTUAL_CORES] = 16;
            k[KNOB_FREQUENCY] = 2.4; // GHz
            ...
        }
        return k;
    }
};
```

ANALYZE & PLAN

```
class SelectorDummy: public Selector{
    ...
    KnobsValues getNextKnobsValues(){
        KnobsValues k(KNOB_VALUE_RELATIVE);
        if(_samples->average().latency < _p.requirements.latency){
            k[KNOB_VIRTUAL_CORES] = 25; // %
            k[KNOB_FREQUENCY] = 25; // %
            ...
        }else{
            k[KNOB_VIRTUAL_CORES] = 75; // %
            k[KNOB_FREQUENCY] = 75; // %
            ...
        }
        return k;
    }
};
```

MORE INFORMATION

http://danieledesensi.github.io/nornir

7 **conference** papers, 6 **journal** papers

Backup Slides

COLOR SCHEME

C7F464

4ECDC4

738498

454F5B