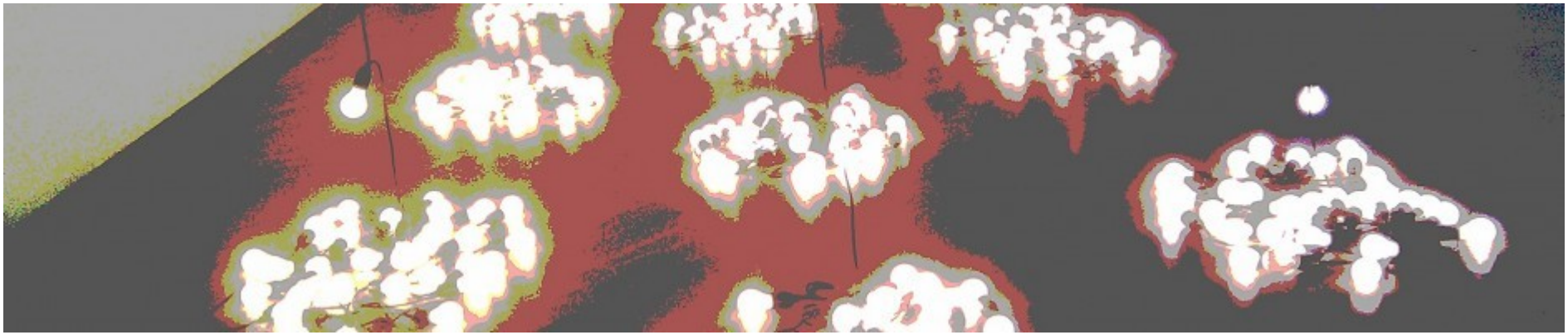
dynamic & adaptive thresholds

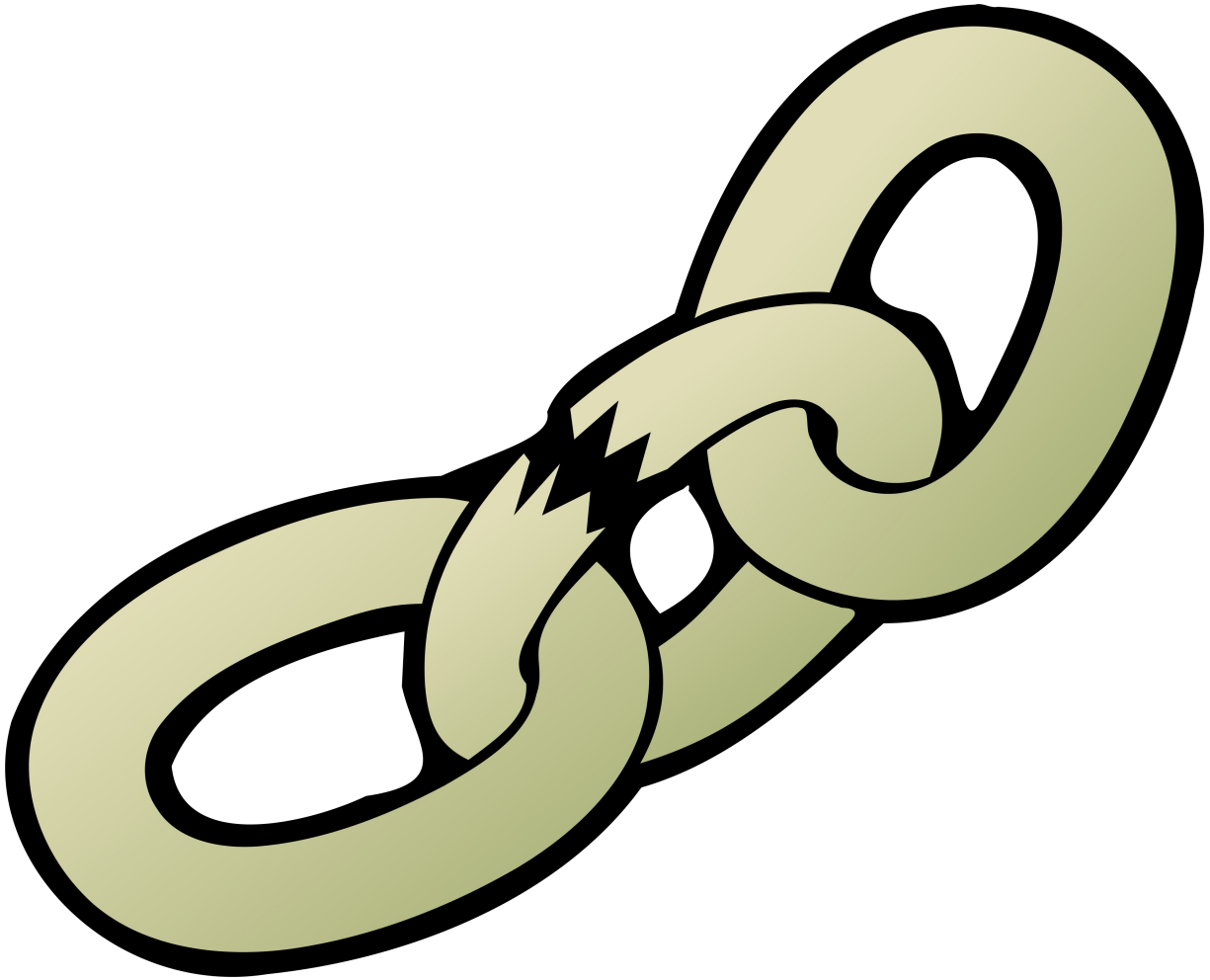
anders håål, ingenjörsbyn ab

www.ingby.com

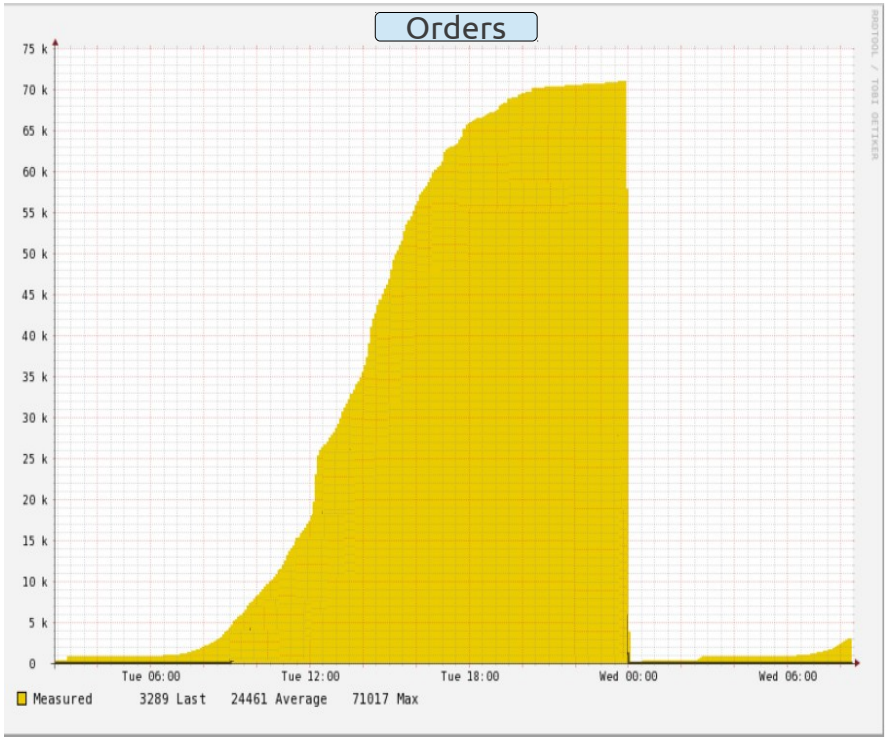


www.bischeck.org

OPEN SOURCE MONITORING CONFERENCE



Where it all started



Different **expectation** depending on the time of day

Different **expectation** depending on day in week and month

Thresholds the Nagios way

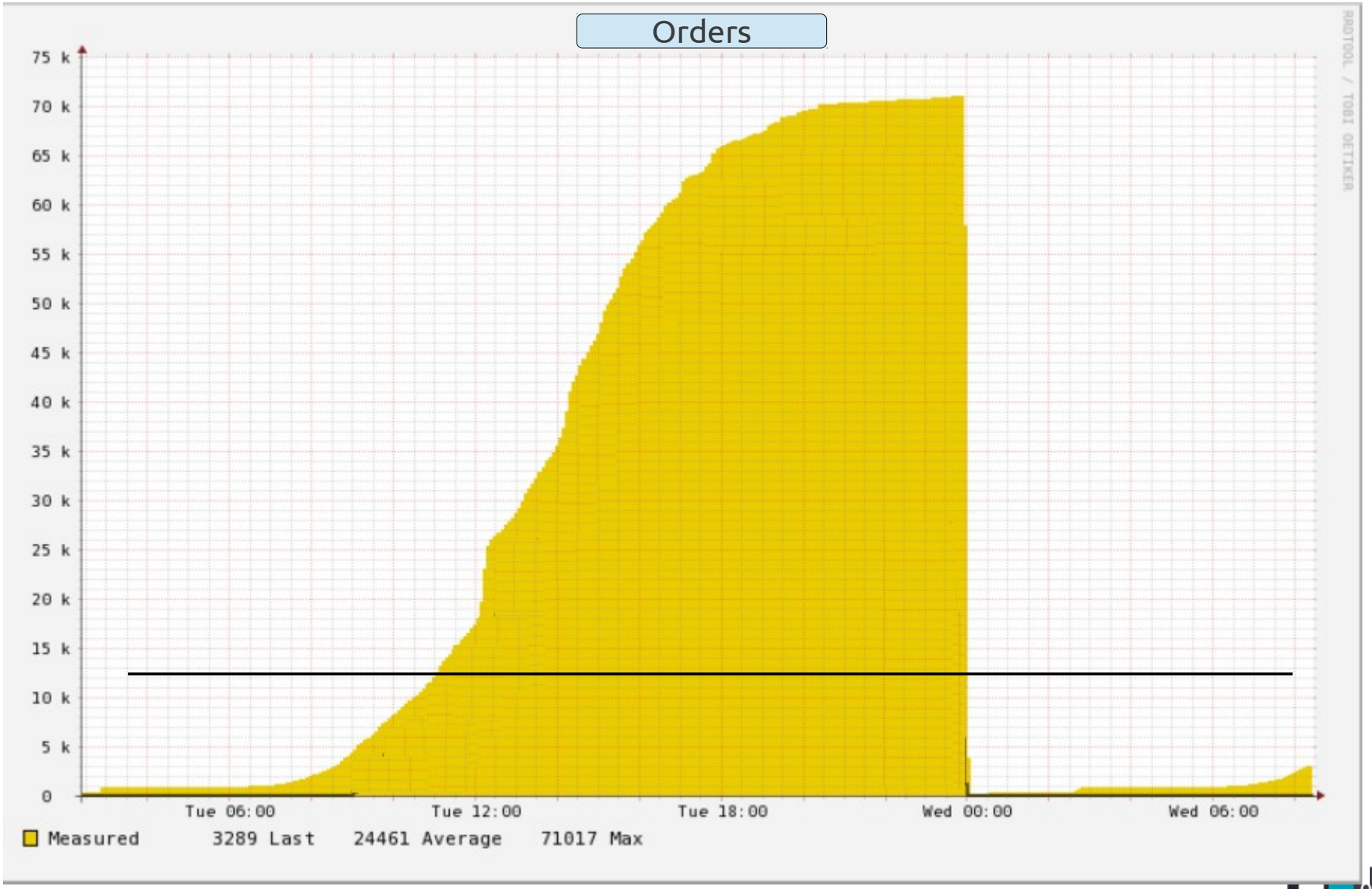
“DISK UTLIZATION HIGHER THEN 90%”

“PING TIME HIGHER THEN 150 MS”

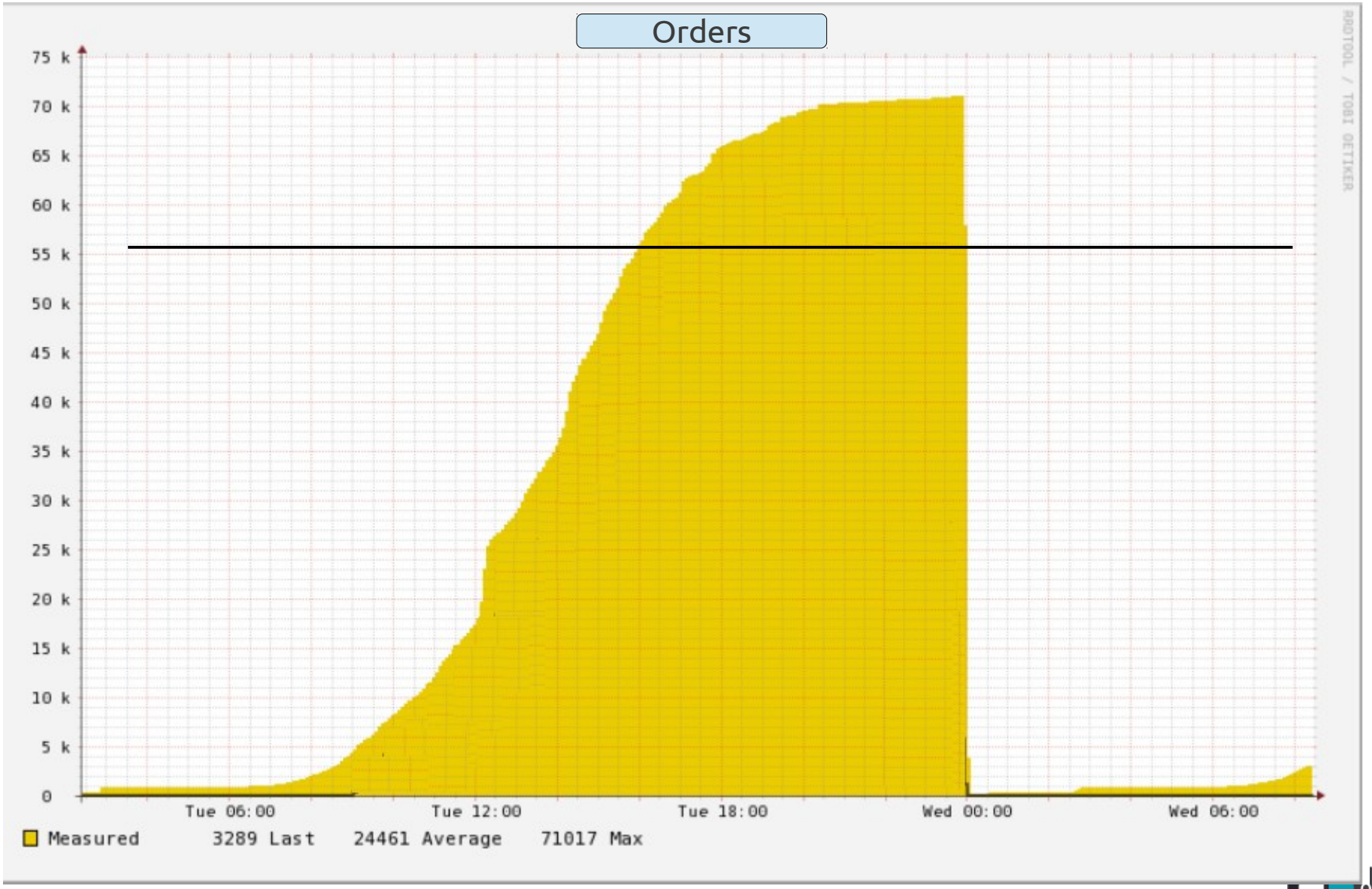
“MORE THEN 100 DATABASE EXECUTES/SECOEND“

Static threshold

The static way will not work



The static way will not work

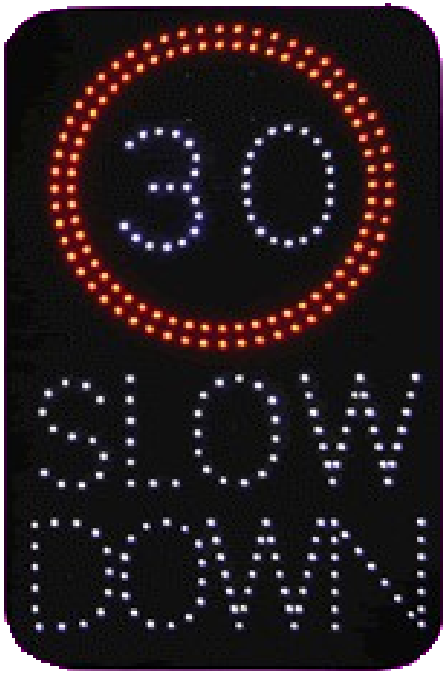


Static thresholds are not enough

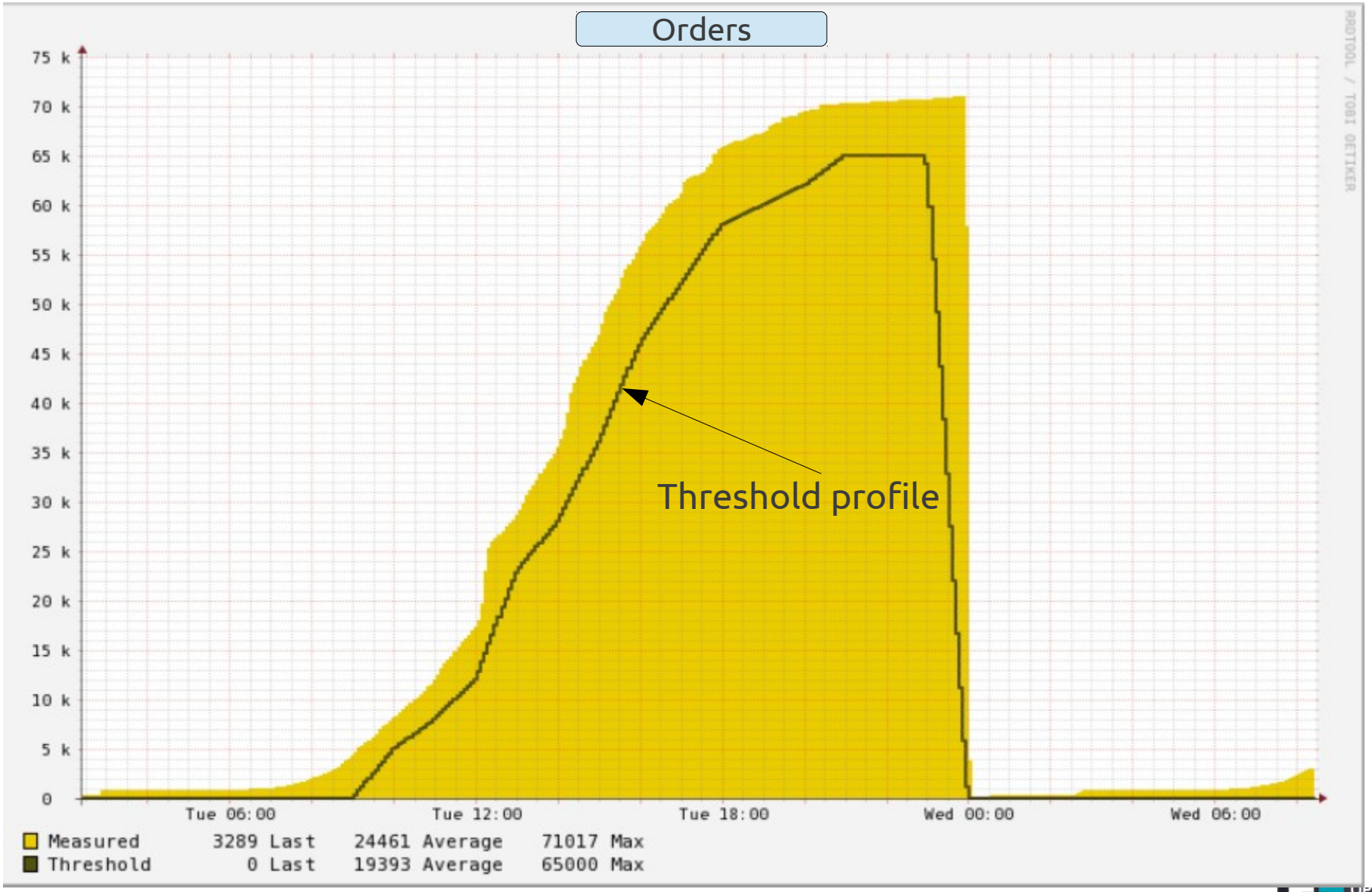
- Business “load” is not static
- Thresholds need to be different throughout the day, week and month
- Need calculated threshold based on historical data
- Threshold related to other services data
- Too many or too fee alarms

From static to dynamic & adaptive thresholds

From static to dynamic & adaptive thresholds

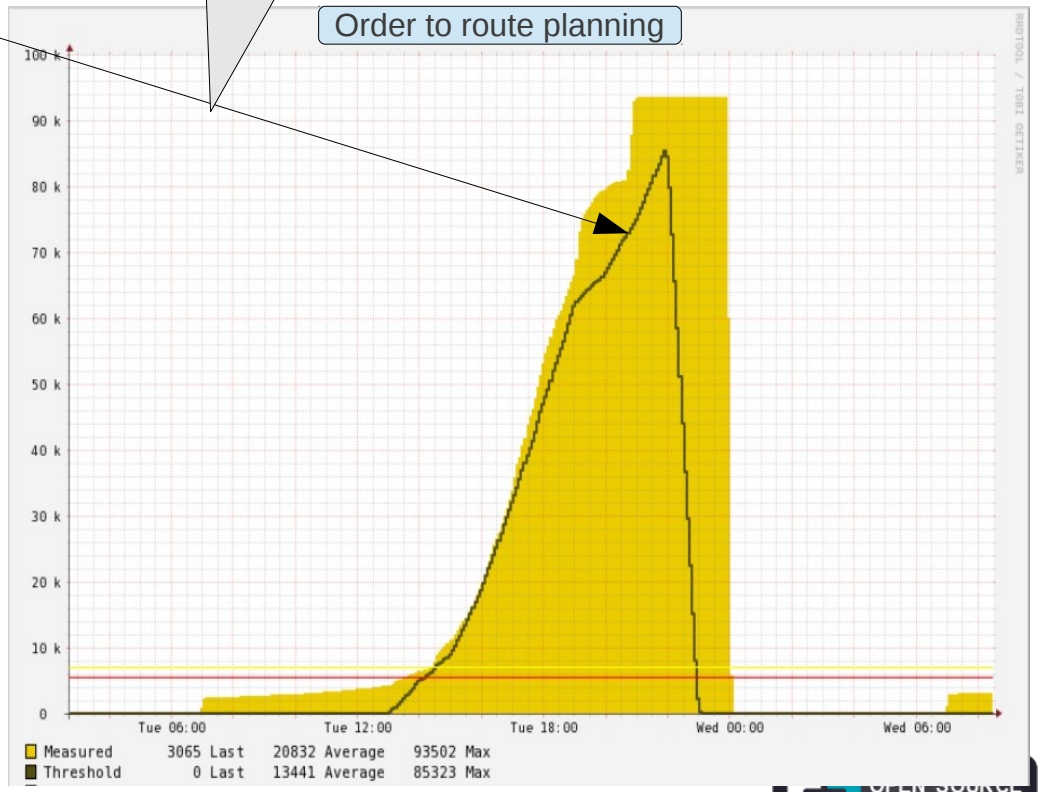


Dynamic threshold



Adaptive threshold

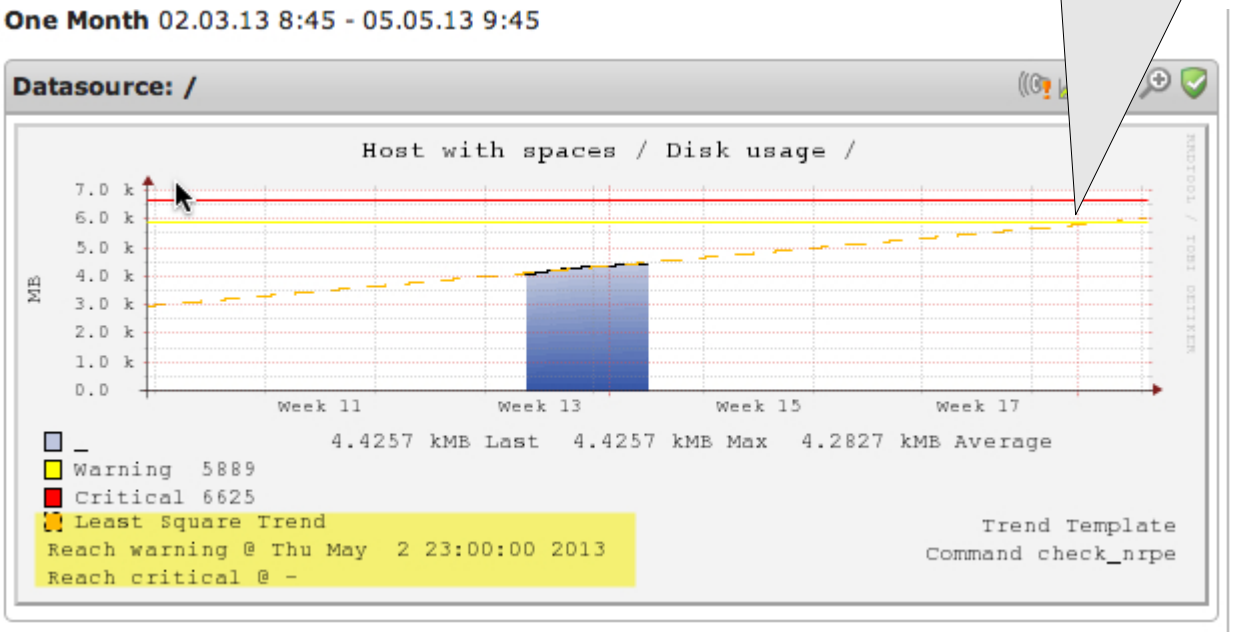
Threshold = function(*Orders*)



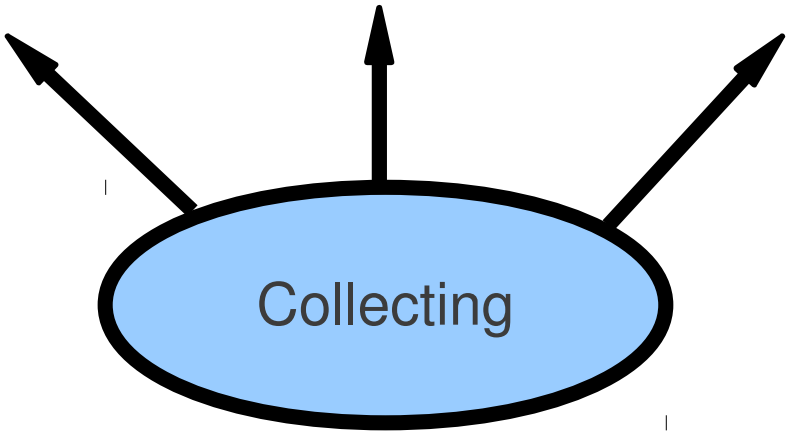
Process driven thresholds!

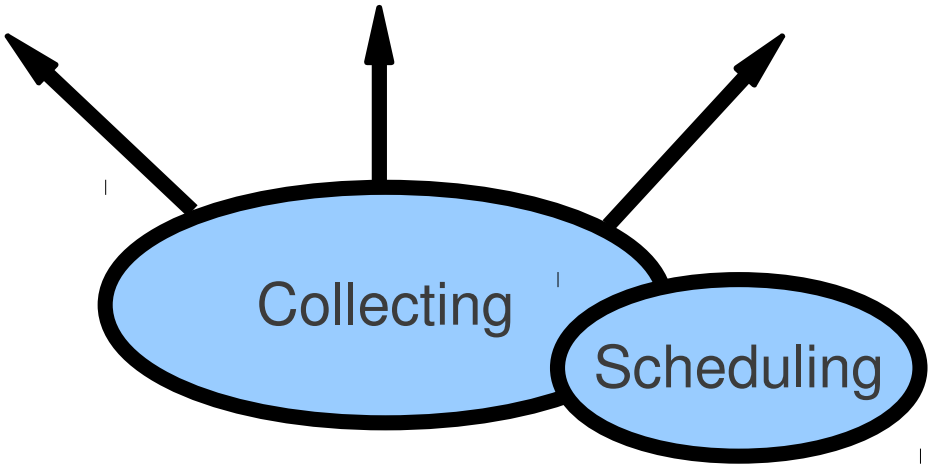
Predictive threshold

Threshold > 6k in 45 days

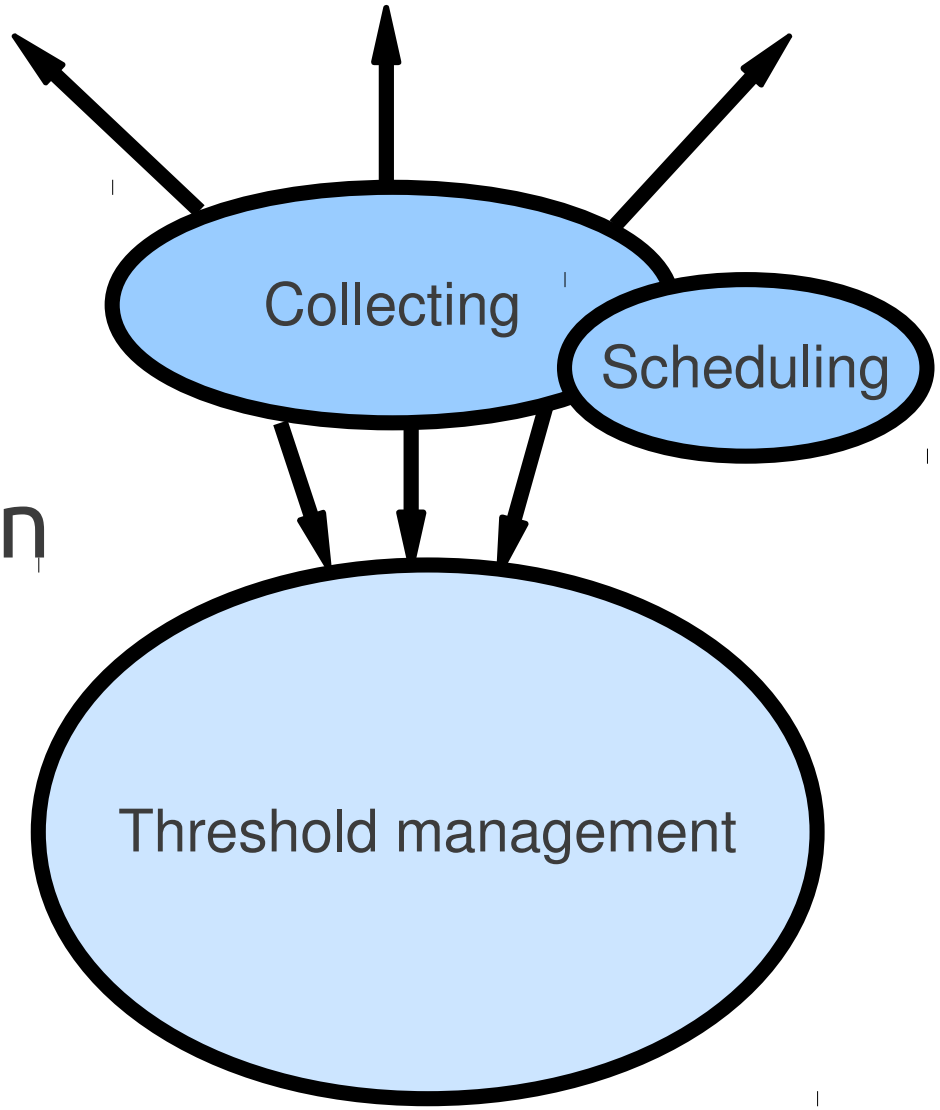


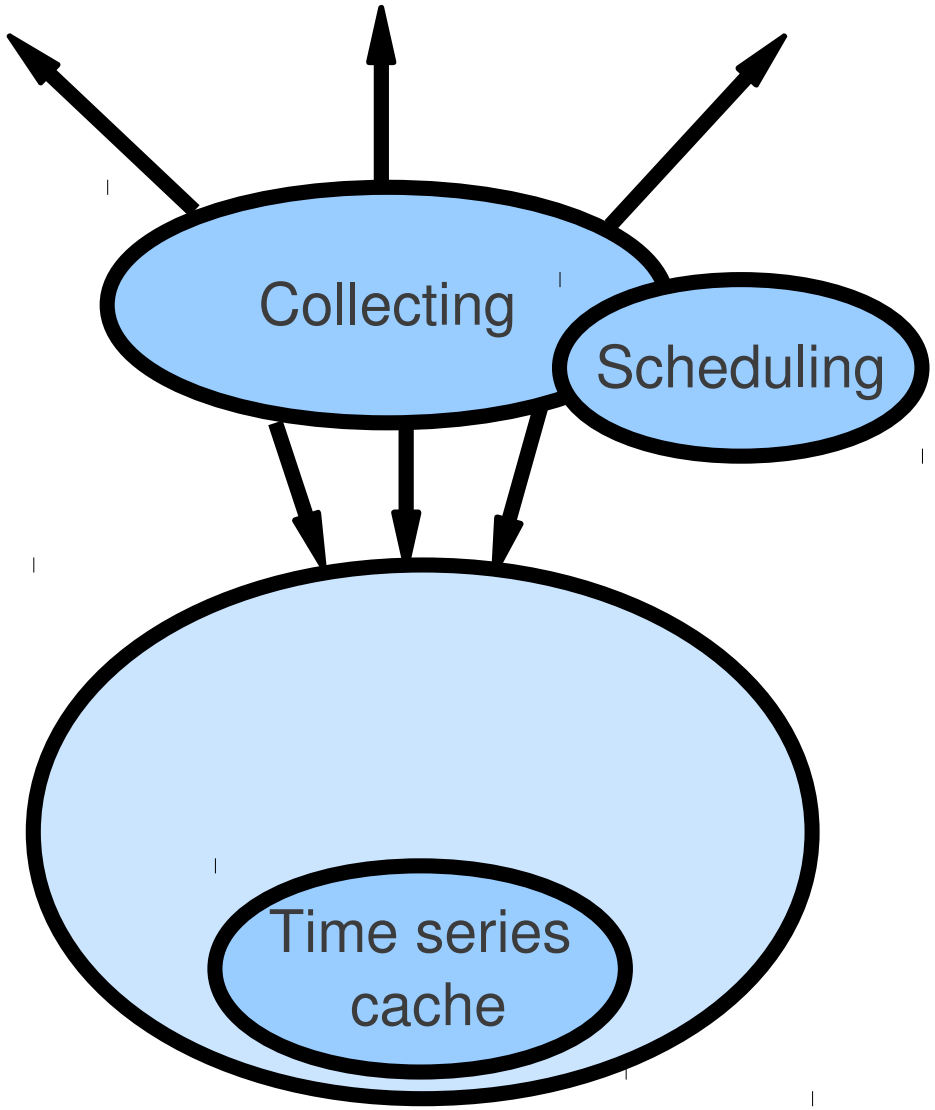
OPEN SOURCE MONITORING CONFERENCE

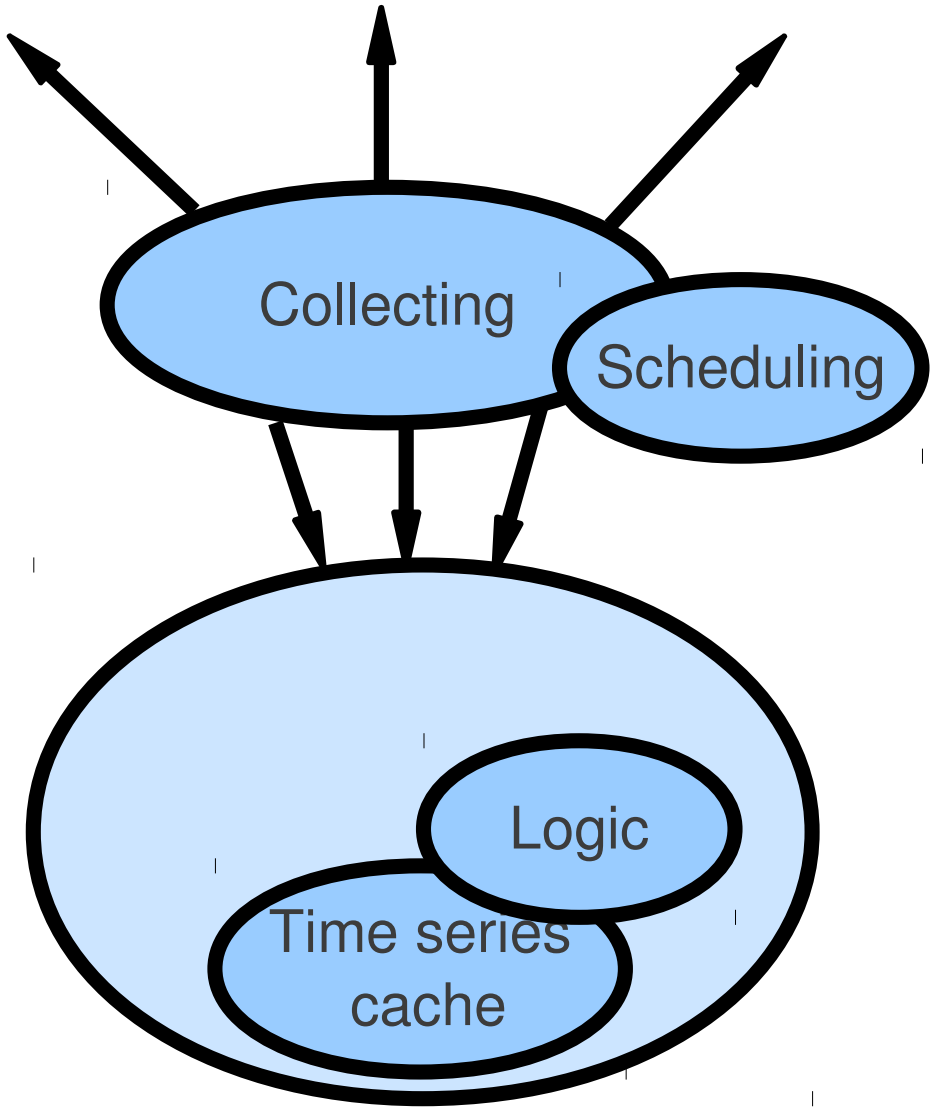


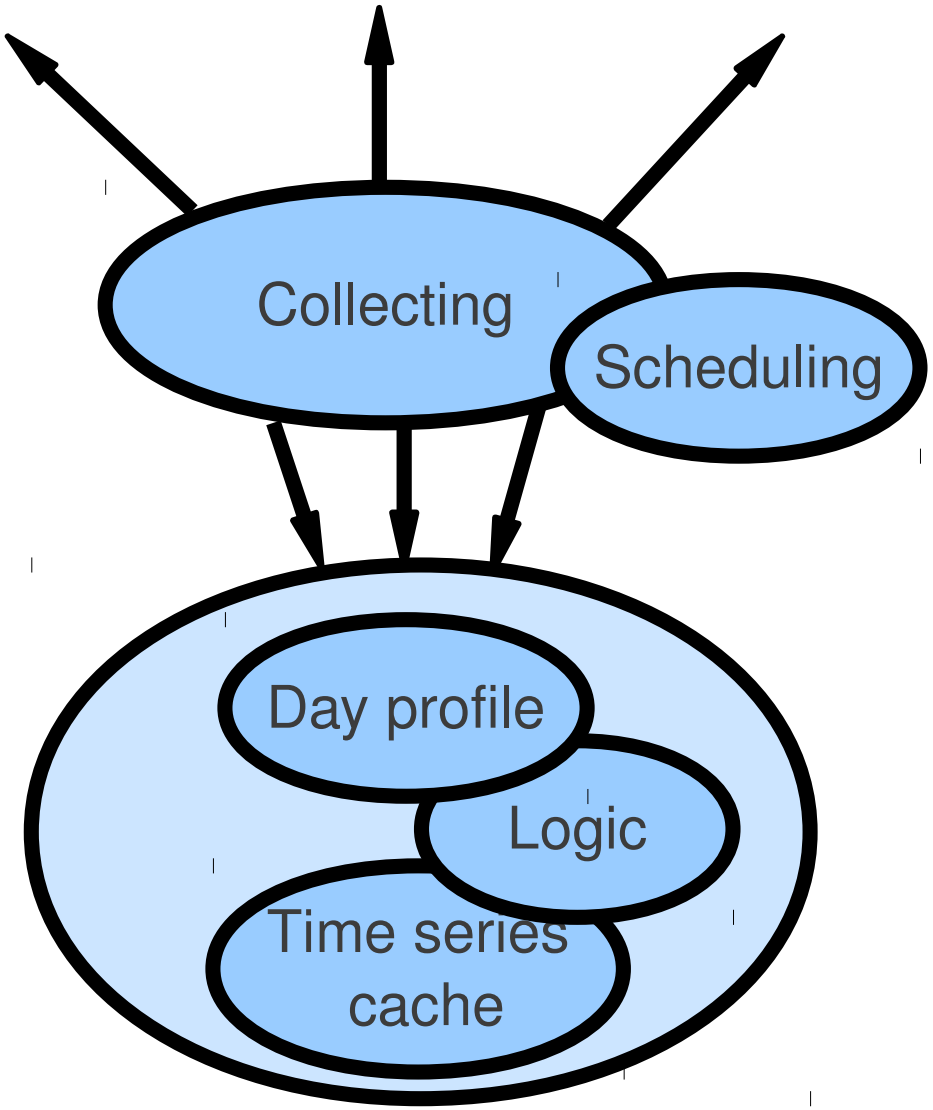


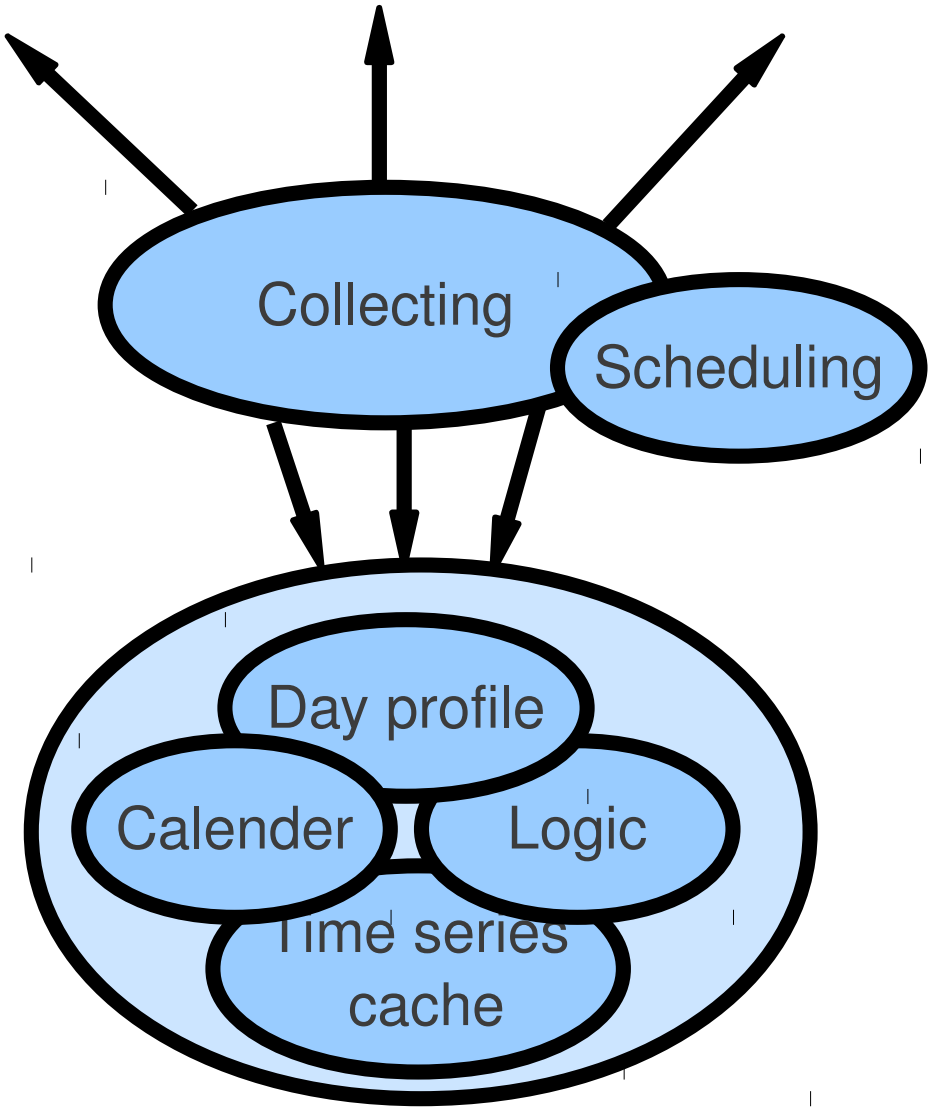
Separation

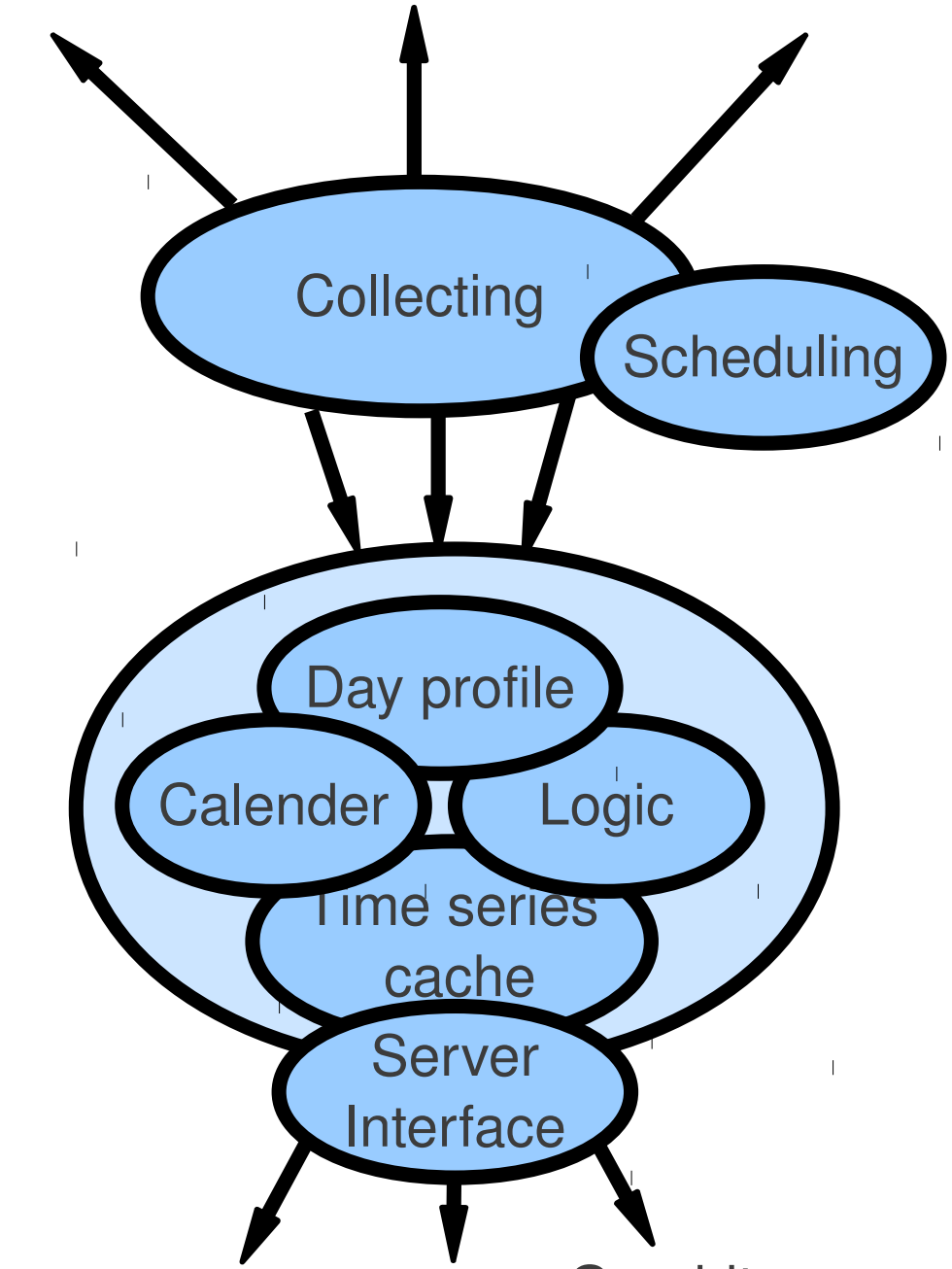






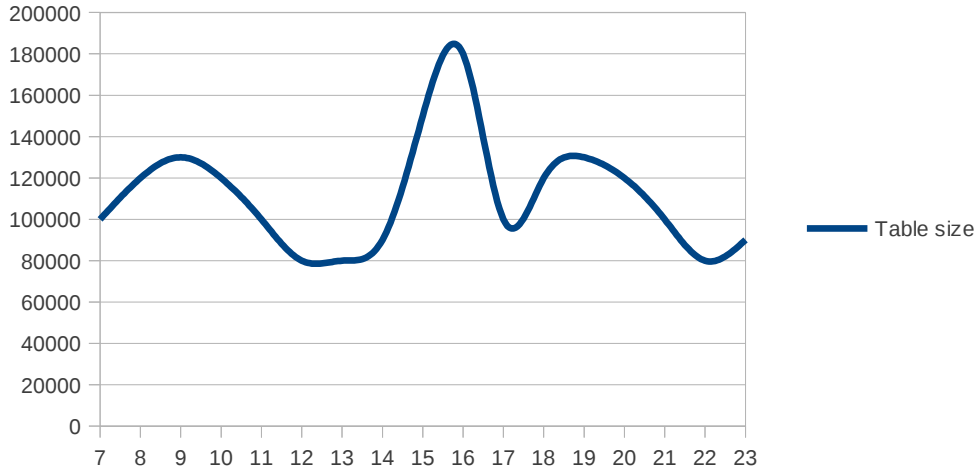






{example 1}

“Database table size should not be bigger then 5 % of yesterdays max size “



{example 1} ✓

“Database table size should not be bigger then 5 % of max size yesterday“

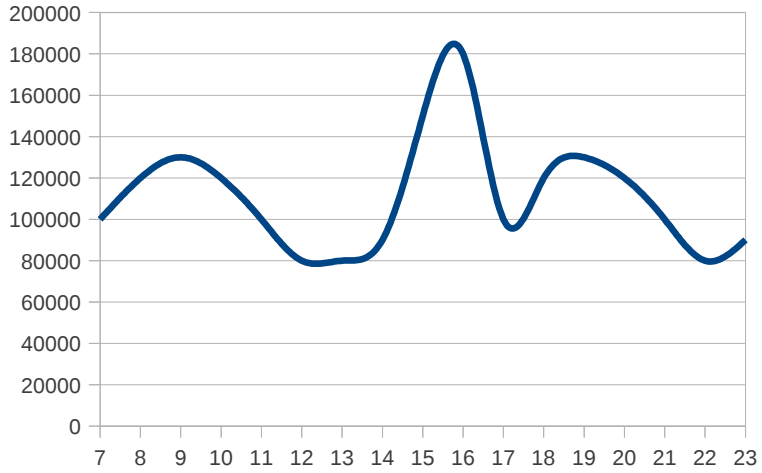


table size < max(yesterday)*1.05

Yesterday

Today

{example 2}

“Number of on-line users should not be more then 10 % higher then the average number of on-line users for the same hour yesterday”

{example 2} ✓

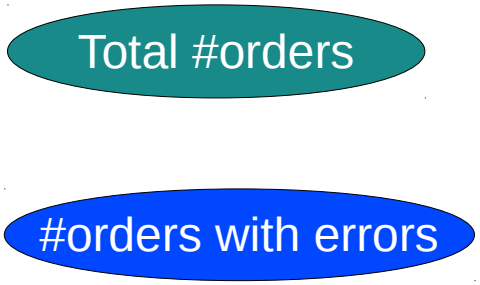
“Number of on-line users should not be more then 10 % higher then the average number of on-line users for the same hour yesterday”

$$users < avg(X_{-24h})*1.1$$

Where X is the historical on-line users data points

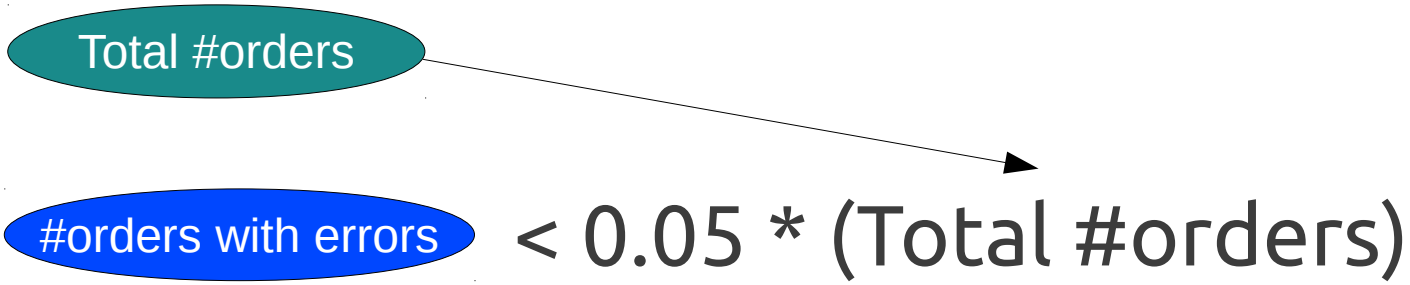
{example 3}

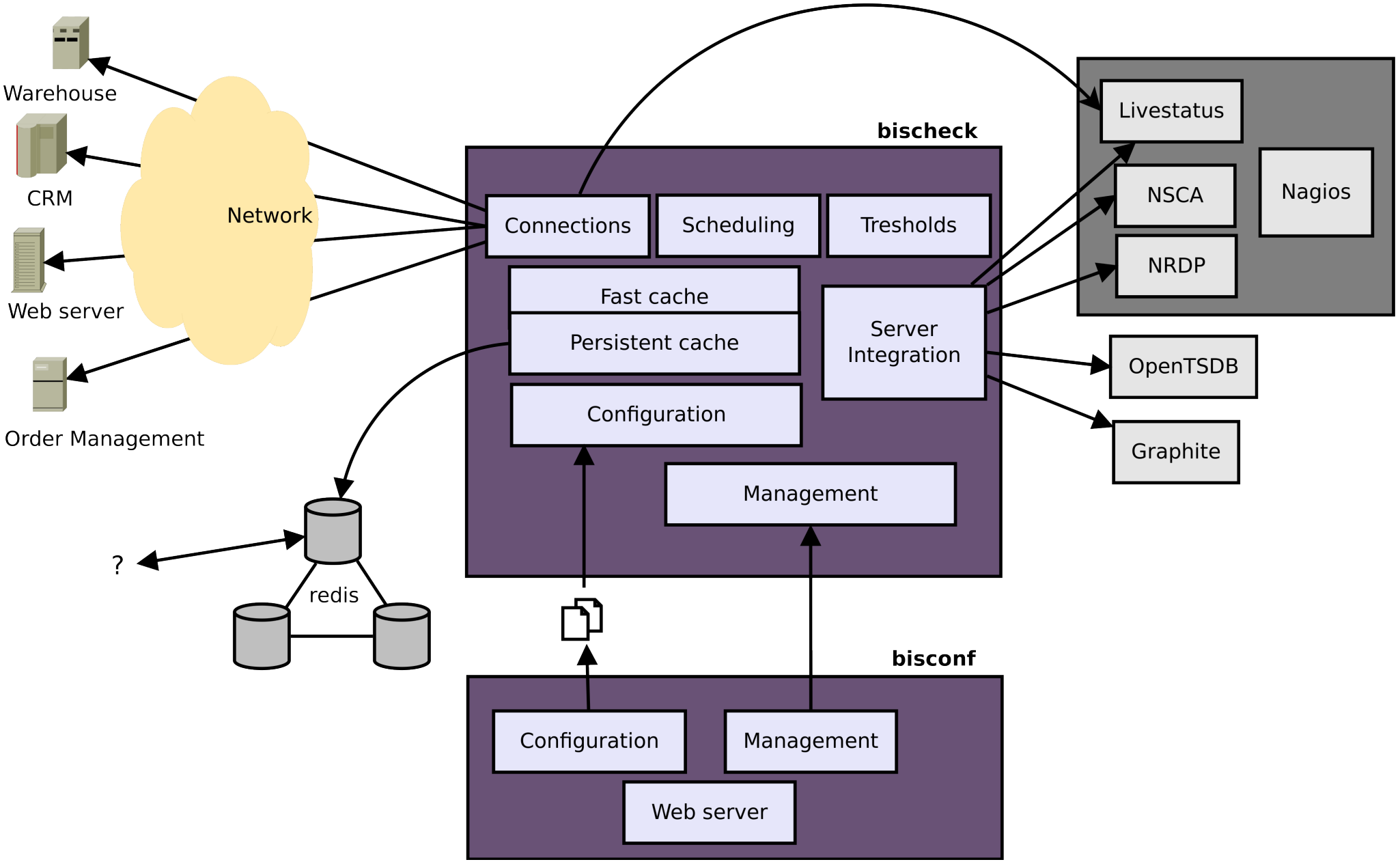
"The number of orders with errors should be lower then 5% of the total number of registered orders"



{example 3} ✓

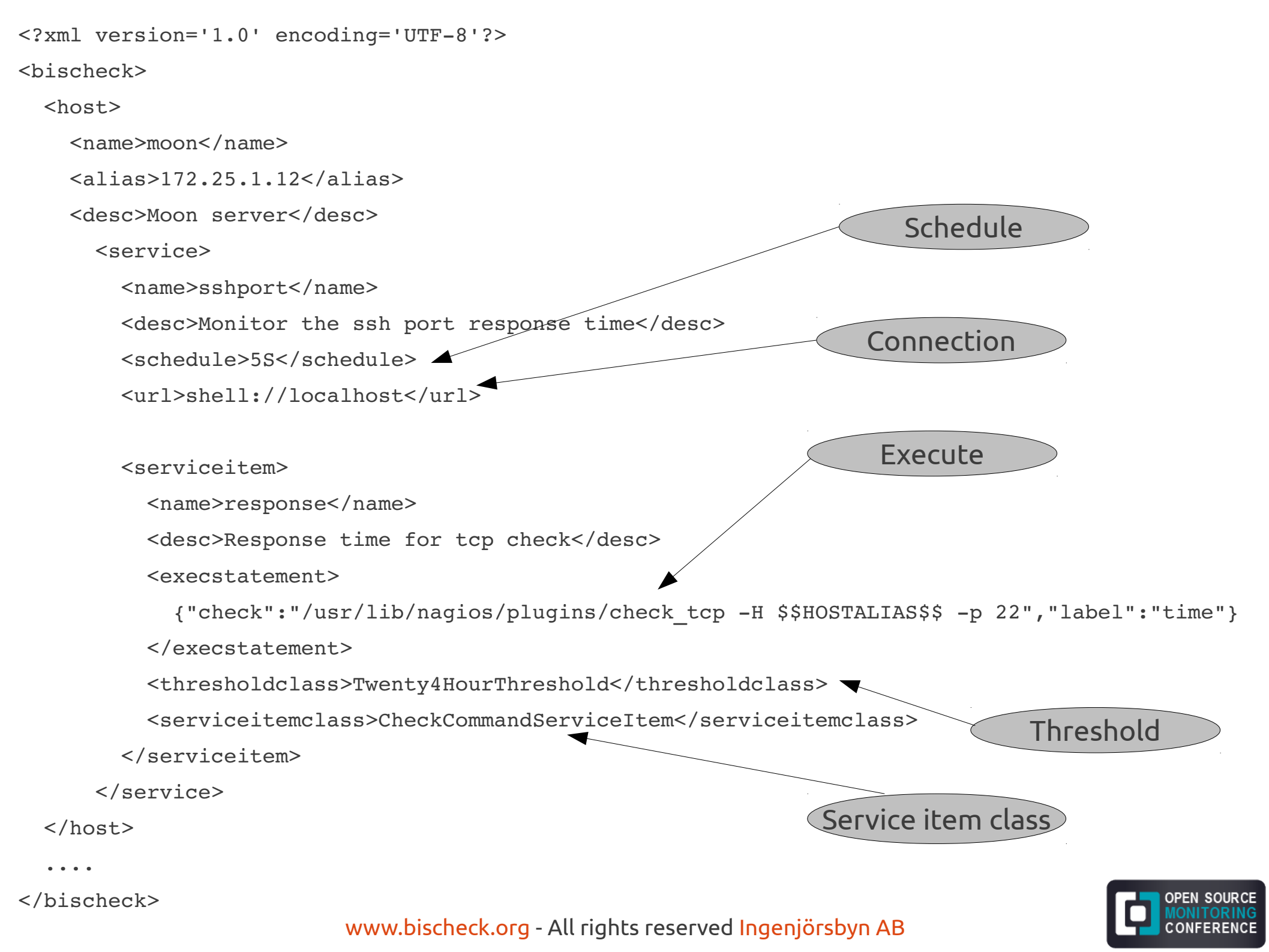
“The number of orders with errors should be lower then 5% of the total number of registered orders”





- Configuration “like” Nagios – host, service but also service item
  - Host is just a container of Services
  - Service specify the connection and scheduling
  - Service item specify the how data will be collected, execute and the threshold class to use
- Host and service name must be the same as in the Nagios configuration to enable passive check

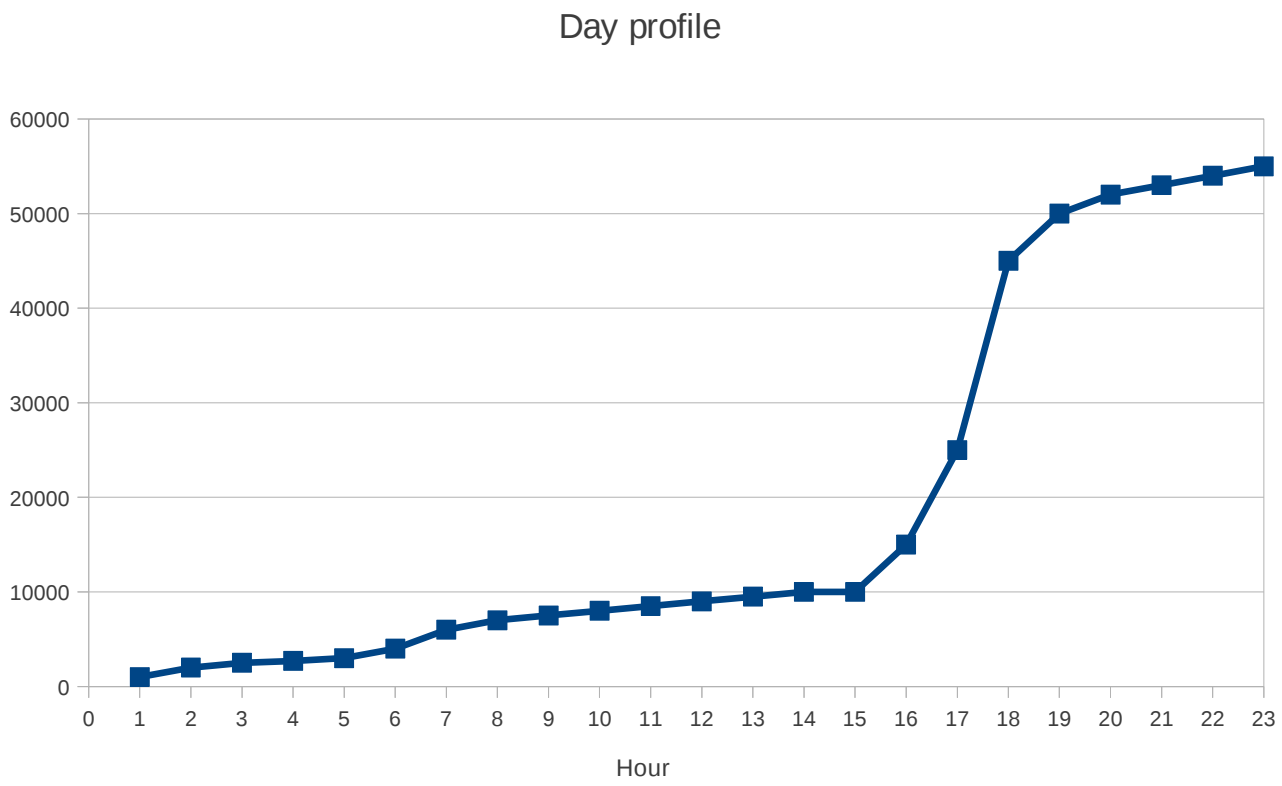
```
<?xml version='1.0' encoding='UTF-8'?>
<bischeck>
  <host>
    <name>moon</name>
    <alias>172.25.1.12</alias>
    <desc>Moon server</desc>
      <service>
        <name>sshport</name>
        <desc>Monitor the ssh port response time</desc>
        <schedule>5S</schedule>
        <url>shell://localhost</url>

        <serviceitem>
          <name>response</name>
          <desc>Response time for tcp check</desc>
          <execstatement>
            {"check":"/usr/lib/nagios/plugins/check_tcp -H $$HOSTALIAS$$ -p 22","label":"time"}
          </execstatement>
          <thresholdclass>Twenty4HourThreshold</thresholdclass>
          <serviceitemclass>CheckCommandServiceItem</serviceitemclass>
        </serviceitem>
      </service>
  </host>
  ....
</bischeck>
```



Cache query

- host-service-serviceitem
  - E.g. *moon-sshport-response*
- Where X is:
  - Single value query:
    - Index – *moon-sshport-response[1]*, 0 is the last collected value
    - Offset time back in time – *moon-sshport-response[-30M]*
  - List of value query:
    - Index from:to – *moon-sshport-response[10:20]*
    - Time from:to – *moon-sshport-response[-24H:-48H]*

Threshold profiles



```
<hours hoursID="0">
 <hourinterval>
    <from>06:00</from>
    <to>11:00</to>
    <threshold>
      0.160
   </threshold>
  </hourinterval>

 <hourinterval>
    <from>12:00</from>
    <to>23:00</to>
    <threshold>
      avg(moon-sshport-response[10:19])
   </threshold>
  </hourinterval>
</hours>
```

```
<hours hoursID="101">
 <hourinterval>
   <from>06:00</from>
    <to>11:00</to>
    <threshold>
      avg(moon-sshport-response[10:19])
    </threshold>
 </hourinterval>

 <hourinterval>
    <from>12:00</from>
    <to>23:00</to>
    <threshold>
      avg(moon-sshport-response[-1H:-2H])* 0.9
    </threshold>
 </hourinterval>
</hours>
```

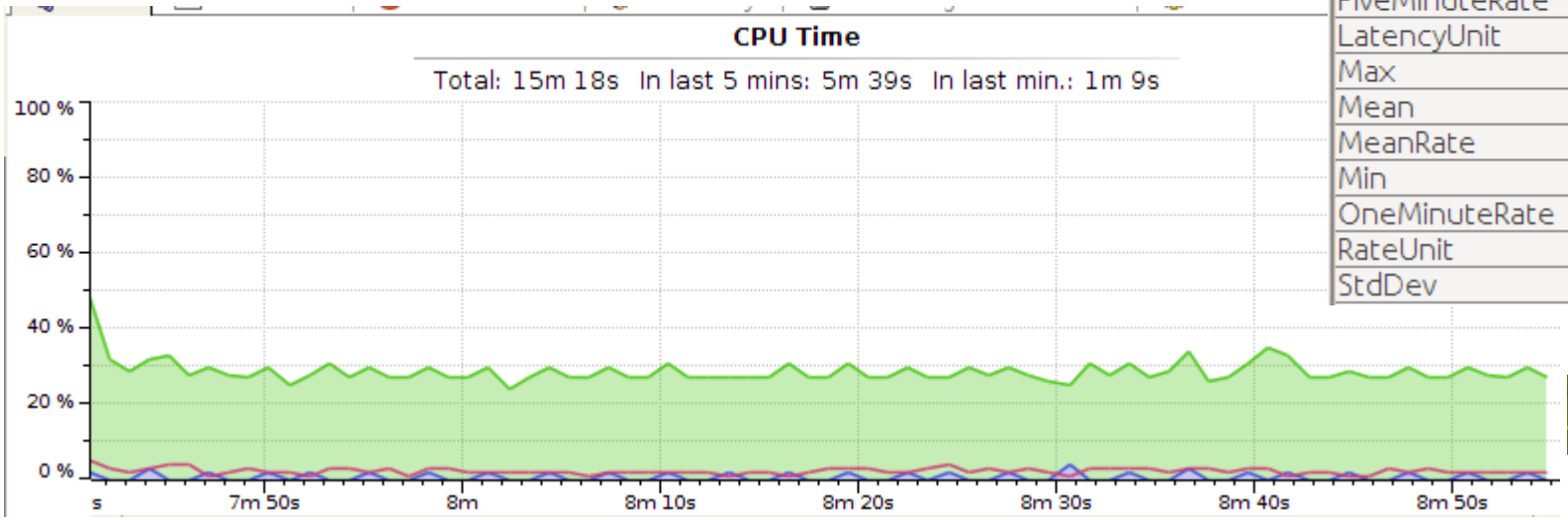
```
<hours hoursID="101">
  <hourinterval>
    <from>12:00</from>
    <to>23:00</to>
    <threshold>
      max(avg(moon-sshport-response[-24H:-48H]),
          avg(sun-sshport-response[-24H:-48H]))
    </threshold>
  </hourinterval>
</hours>
```

Bischeck core* performance – current trunk

4 core Intel i7 2.9 GHz, 8Gbyte memory

- 2000 host-service-serviceitems
- 1 sec interval schedule
- 2000 data points/sec writes to Redis
- 2000 avg calculations per sec, each with 50 values
- 1000 max calculations per sec, each with 100 values
- 2 millions data points in Redis
- Every 60 sec cache eviction Redis

Bischeck ~30 % cpu
Redis ~4 % cpu



ServiceJob timer

Name	Value
50thPercentile	**0.7310205**
75thPercentile	**1.151654**
95thPercentile	**2.479539749999998**
98thPercentile	**3.823695679999989**
999thPercentile	**15.755260498000009**
99thPercentile	**5.2313463200000125**
Count	**1475805**
EventType	calls
FifteenMinuteRate	**1168.0676879931345**
FiveMinuteRate	**1859.4106214855358**
LatencyUnit	MILLISECONDS
Max	**4743.457865**
Mean	**1.8650097561195416**
MeanRate	**2012.4881826283595**
Min	**0.09106**
OneMinuteRate	**2020.3115603703284**
RateUnit	SECONDS
StdDev	**14.745070924519183**

YourKit

*) Internally generated data and no server integration but all NSCA formatting

OPEN SOURCE MONITORING CONFERENCE

Bischeck features

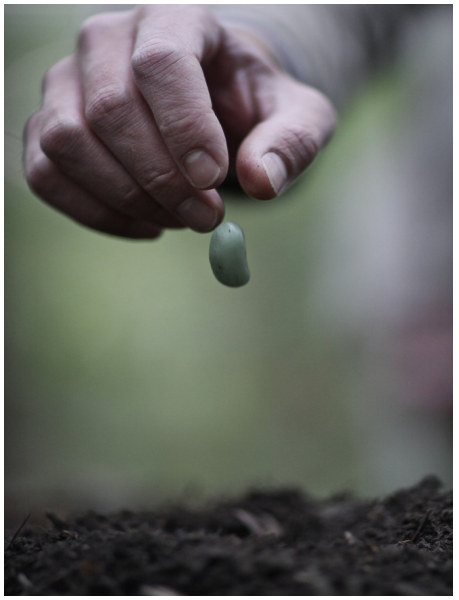
- Multi-threaded and multi-scheduling per service definition
- Data collection – jdbc, livestatus, nagios-plugins, internal cache
- Virtual services – based on cached data
- Cache aggregations on Hour, Day, Week, Month
- Time series cache queries – time, index and ranges
- Time and calendar threshold profiles
- Math functions including prediction
- Configuration templates
- Macros support – configuration and runtime
- Extendable – implement java interfaces
- XML configuration supported with WEBui (beta)
- JMX instrumentation
- GPL 2 license

New monitor opportunities

- Complement to your “normal” thresholds
  - Baseline
  - Dynamic
  - Predictive
  - Adaptive
- Monitor applications and business data
  - Process monitoring
  - BAM – business activity monitoring
  - OBI - operational business intelligence
- Business SPOC
  - Enable business users with dashboards, notifications and alarms that make sense

Q&A

anders håål, ingenjörsbyn ab
www.ingby.com
anders.haal@ingby.com
+46 70 575 35 46
@thenodon



www.bischeck.org
bischeck@ingby.com

Please test the new bischeck 1.0.0 release candidate

Pictures – Creative Commons

www.flickr.com/photos/kclifford/3980451175
www.flickr.com/photos/catatronic/2397319483
www.flickr.com/photos/dtrimarchi/6815004766
www.flickr.com/photos/bikeracer/6740232