## Summary

Current implementation of services was designed long time ago, it is rather limited and has significant limitations. Major limitations are:

- scalability: it does not scale well, also bad user experience when number of services exceeds few hundreds
- hard to configure: mapping based on triggers does not work well and requires significant effort
- very limited visualization: no graphical representation of services, no good reporting
- limited SLA calculation rules
- no alerting in case of service status changes
- user permissions: all or nothing

Proposed functionality will introduce more flexible status calculation and propagation rules.

## Use cases

1. I want complex status calculation and propagation rules to support various use cases
    a. Monitoring of load balancer cluster having nodes of different performance (weight)
    b. Monitoring of HA cluster (amount or percentage of available nodes is important)
    c. Different handling of problems of different status
    d. Monitoring of load balancing cluster with nodes of different weight

## Zabbix acceptance

Current implementation of Services must be extended to support:

1. Service must support the following new attributes:
    a. **Weight:** optional weight of the service, integer in a range of 0 (default) to 1000000
    b. **Status calculation rules:** see below
    c. **Status propagation rules:** see below
2. Status of a parent service will be determined by status of its child services based on status calculation rules
3. Advanced service status calculations rules
    a. **Status calculation rules** define how to calculate status of the service based on status of its  child services. Main rule, one of:
        i. **Most critical of child nodes**: status is determined as the most severe status of its child services (default rule)
        ii. **Most critical if all children have problems**: if all children have non-OK status then status is determined as the most severe status of its child services
        iii. **Set status to OK**: status is set to OK
    b. User may also specify any number of additional rules:
        i. If at least **N** child nodes are **Status** or greater then set to **New status**
        ii. If at least **N%** child nodes are **Status** or greater then set to **New status**
        iii. If less than **N** child nodes are **Status** or less then set to **New status**
        iv. If less than **N%** child nodes are **Status** or less then set to **New status**
            1. where:
                a. N - number of child services, for example, "4"
                b. N% - percentage of child services, for example, "50%"

            c. Status is one of: OK, Not classified, Information, Warning, Average, High or Disaster

      v. If at least **W** of child nodes weight is in **Status** or greater then set to **New status**
      vi. If at least **N%** of child nodes weight is in **Status** or greater then set to **New status**
      vii. If less than **W** of child nodes weight is in **Status** or less then set to **New status**
      viii. If less than **N%** of child nodes weight is in **Status** or less then set to **New status**

   c. If more than one rule matches conditions then the most critical status of the main and additional rules will be used

4. Advanced service status propagation rules
   a. **Service propagation rules** will define service status that is passed to its parent service. If does not affect its own status of the service. The following propagation rules must be supported, one of:
      i. **As is**: service status is seen unchanged by its parent service (default rule)
      ii. **Increase**: status is increased by a given number, '1' by default
      iii. **Decrease**: status is decreased by a given number, '1' by default
      iv. **Ignore this service**: service will be ignored by parent
      v. **Fixed status**: sets fixed service status
5. Existing attribute 'Status calculation algorithm' must be upgraded using the following rules:
   a. Do not calculate → **Set status to OK**
   b. Problem, if at least one child has a problem → **Most critical of child nodes**
   c. Problem, if all children have problems → **Most critical if all children have problems**

## Zabbix UI changes

1. Mockup of the Service configuration form
   a. Advanced configuration (additional rules, status propagation rule and weight) is not visible by default

## Examples

Parent service has 9 child services having the following statuses: 5OK, 2A (Average), 1H (high) and 1D (Disaster). Different calculation rules of the Parent service will end up to different results:

1. Example 1
    a. **Most critical of child nodes**
        i. Result: Disaster, status of the most critical child service
2. Example 2

   a. **Set status to OK**
   b. Rule 1: **Average** if at least **4** are **Average** or greater
   c. Rule 2: **Disaster** if at least **3** are **High** of greater
      i. Result: Average, because we have 4 or more child services having Average status: 2A+1H+1D. No match for Rule 2.
3. Example 3
   a. **Most critical of child nodes**
   b. Rule 1: **Average** if at least **4** are **Average** or greater
   c. Rule 2: **Disaster** if less than **75%** are in **OK status**
      i. Result: Disaster. Both rules match but we take higher status, i.e. result of Rule 2. Besides **Most critical of child nodes** rule also suggests Disaster.

## Nonfunctional requirements

1. N/A

## Decisions made

1. No support of propagation rules for weights, can be implemented in the future if needed