Statically and Dynamically verifiable SLA metrics

Violet Ka I Pun

Department of Informatics
University of Oslo

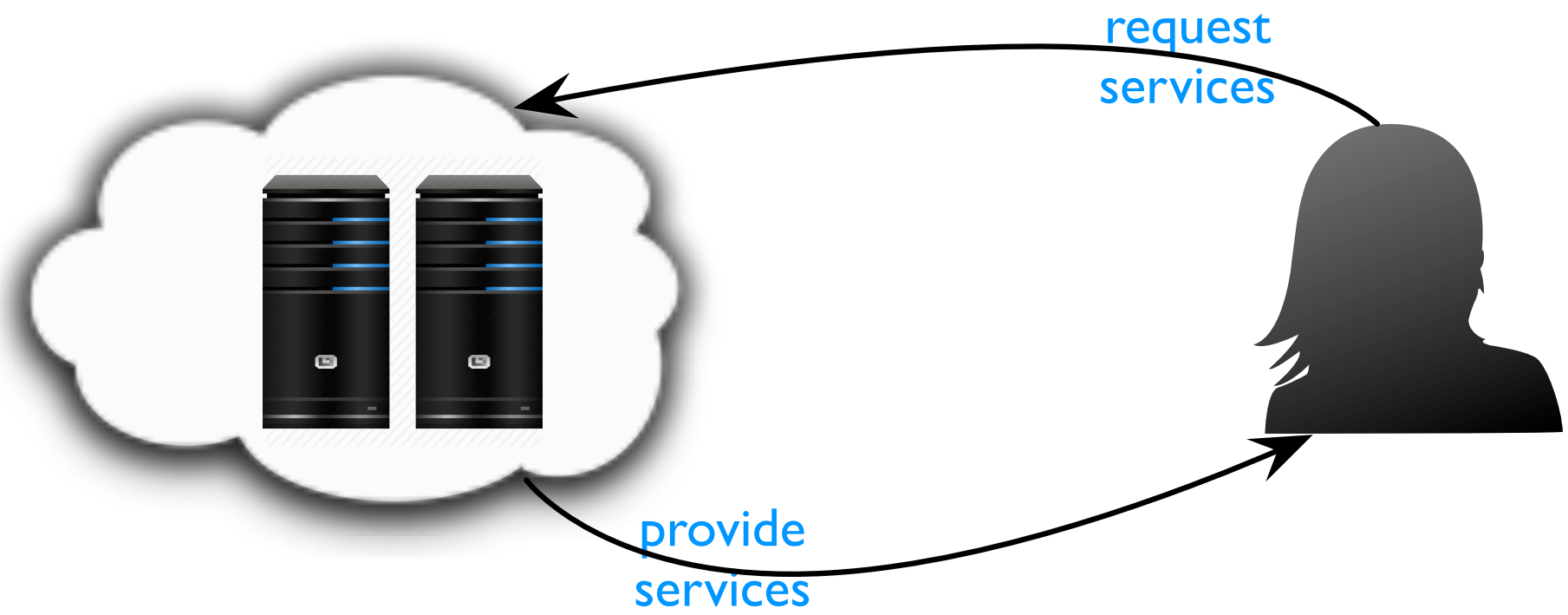
(Elena Giachino[1], Stijn de Gouw[2], Cosimo Laneve[1] and Behrooz Nobakht[2])
[1] University of Bologna
[2] SDL Fredhopper

COST ARVI meeting

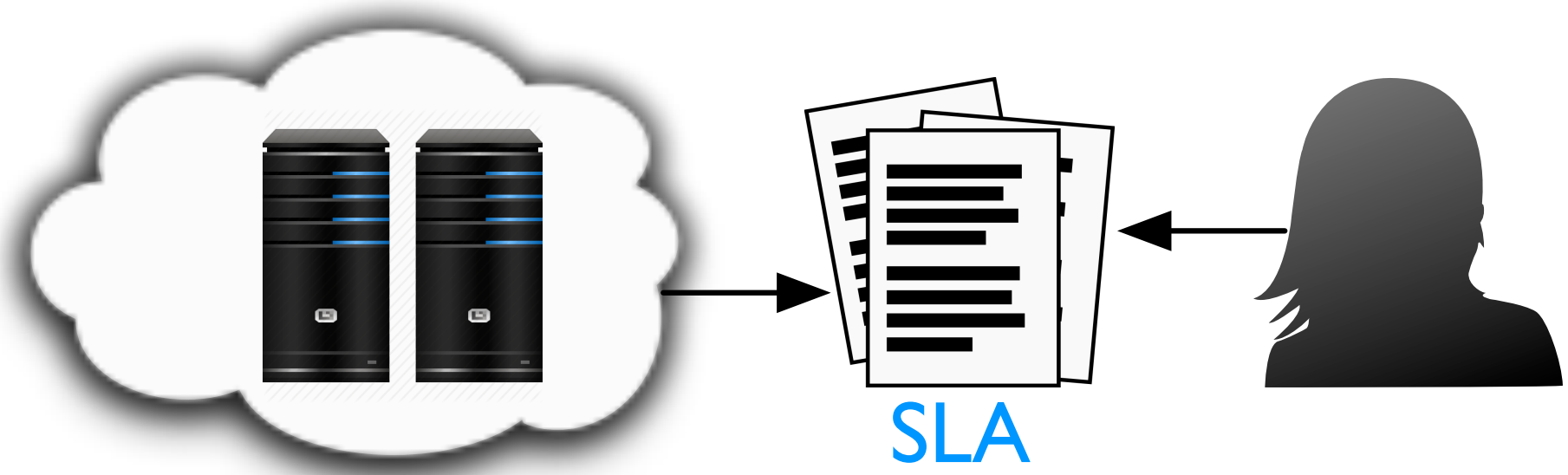
http://www.envisage-project.eu

Motivation

Motivation



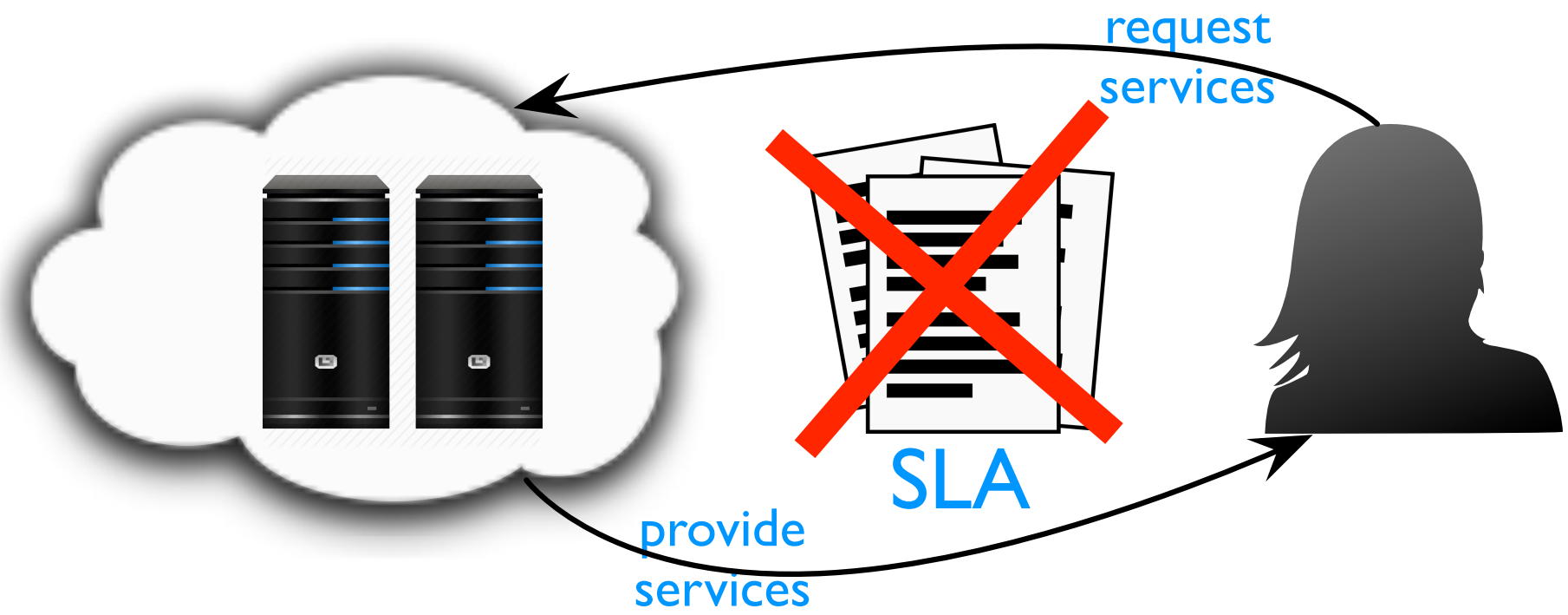
Motivation



Service Level Agreements (SLAs)

- Legal contracts between service providers and customers
- Define the quality of service
- Informal

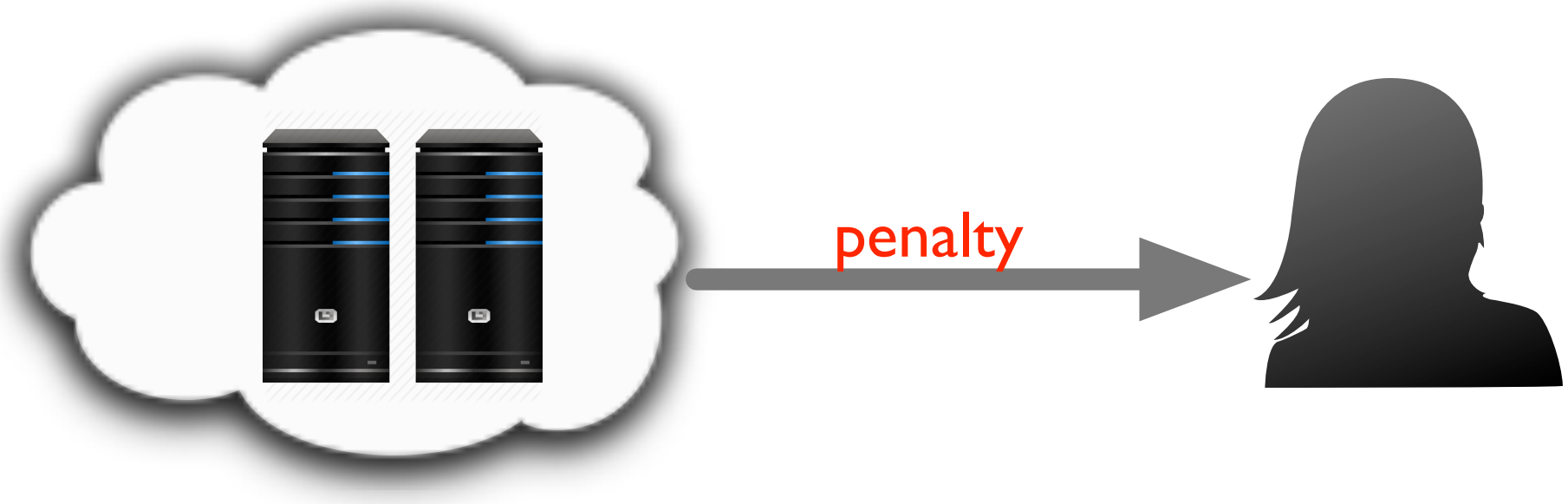
Motivation



Service Level Agreements (SLAs)

- Legal contracts between service providers and customers
- Define the quality of service
- Informal

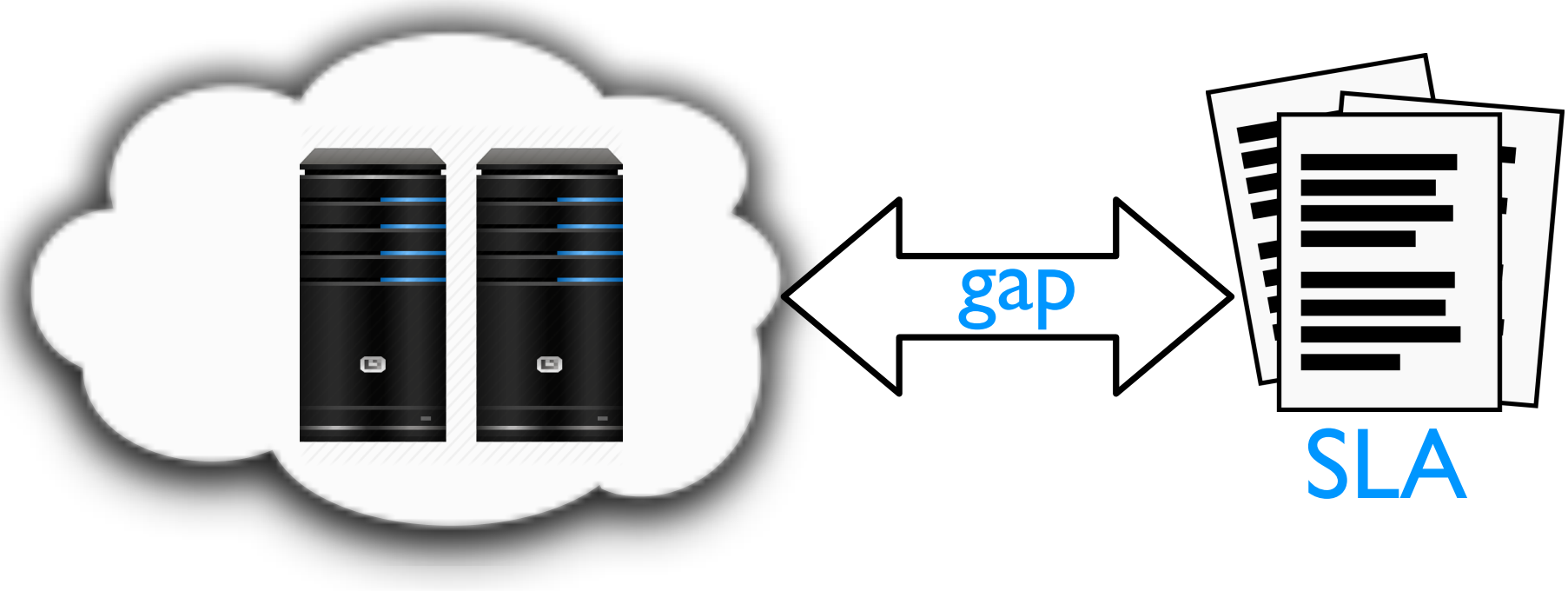
Motivation



Service Level Agreements (SLAs)

- Legal contracts between service providers and customers
- Define the quality of service
- Informal

Motivation – The gap between SLAs and services



- Difficult to assess whether a service complies with an SLA or not
- Waste of resources and money

Motivation – Bridge the gap between SLAs and services

Metrics:

1. A means to specify QoS in SLAs
   - Define the boundaries and margins of errors wrt. cloud services
2. Usage:
   - Static time: code evaluation
   - Runtime: service monitoring, balancing, or remediation

Motivation – Bridge the gap between SLAs and services

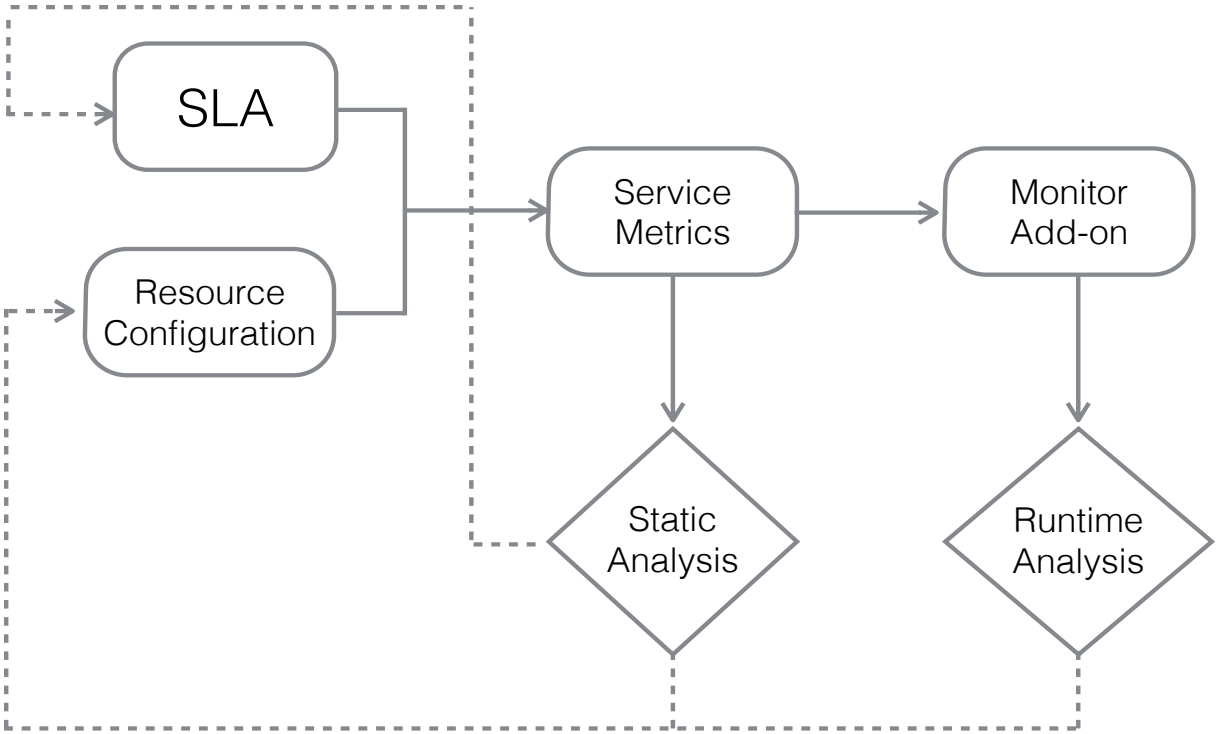
Metrics:

1. A means to specify QoS in SLAs
   - Define the boundaries and margins of errors wrt. cloud services
2. Usage:
   - Static time: code evaluation
   - Runtime: service monitoring, balancing, or remediation

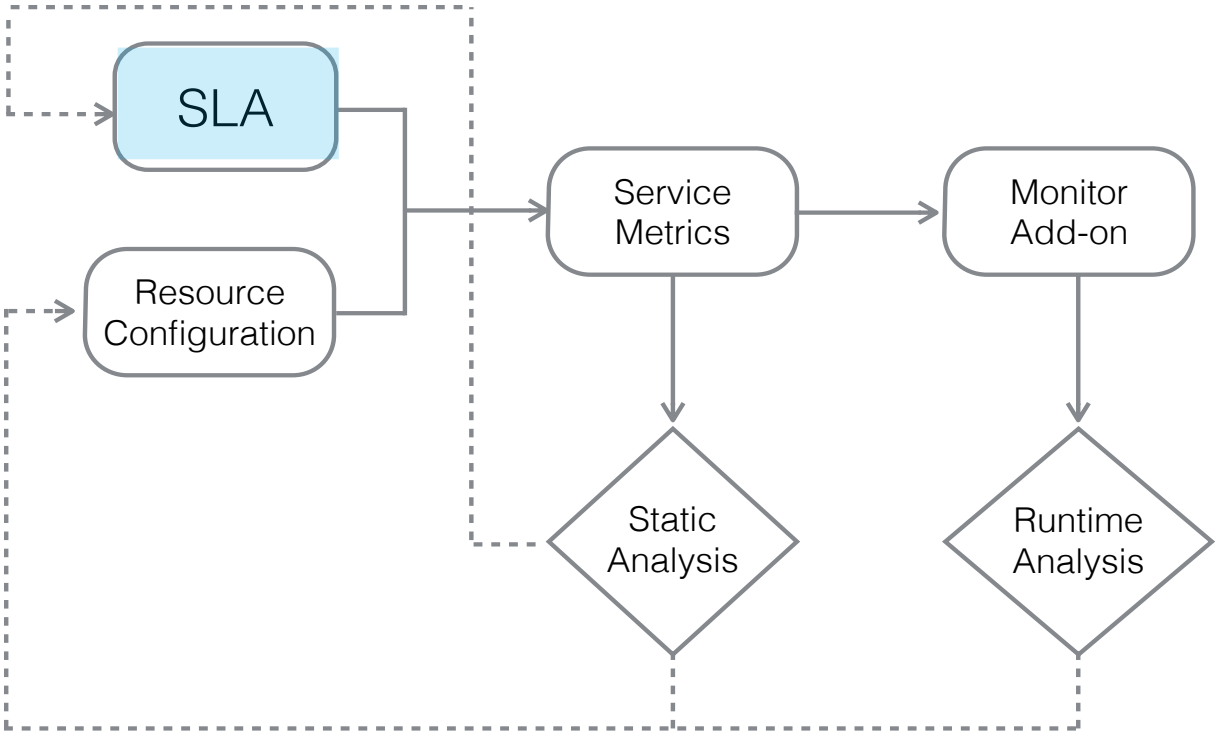
Our approach:

1. Metric functions:
   - A simple formal description of SLA
2. A mathematical framework
   - Statically derives the SLA quality levels from programs
   - Dynamically monitors service behaviours

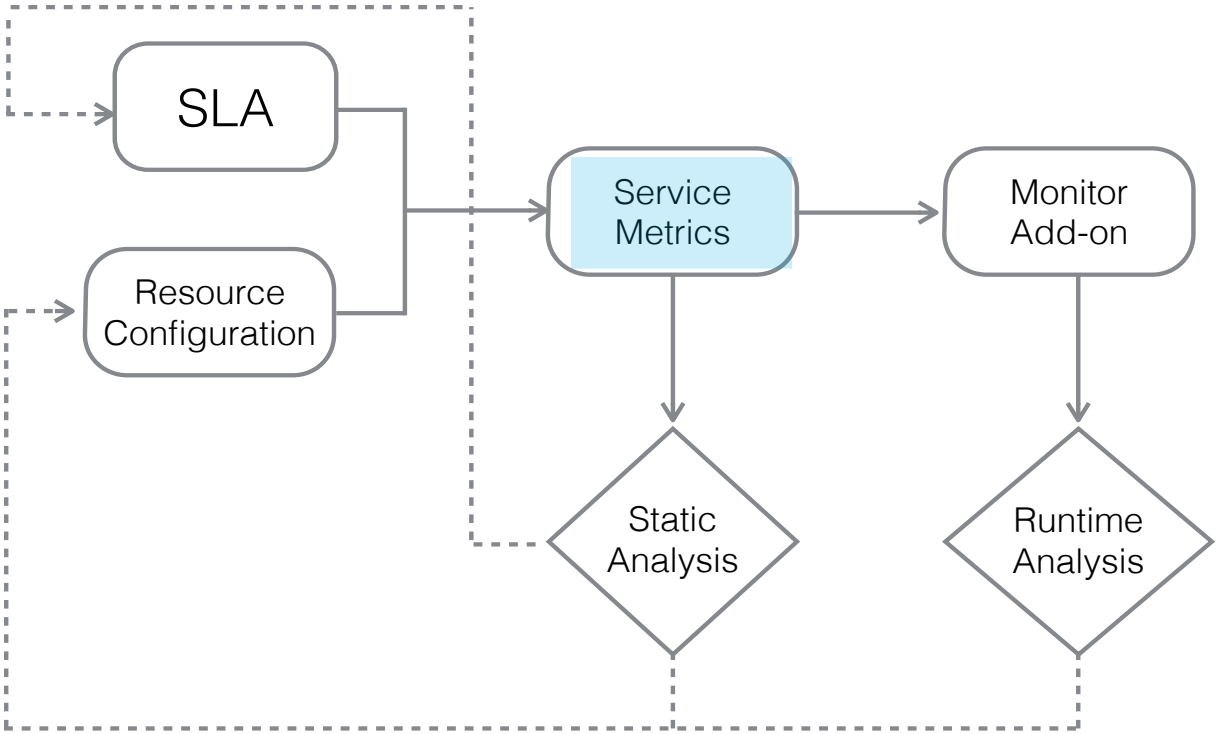
The Analysis Flow



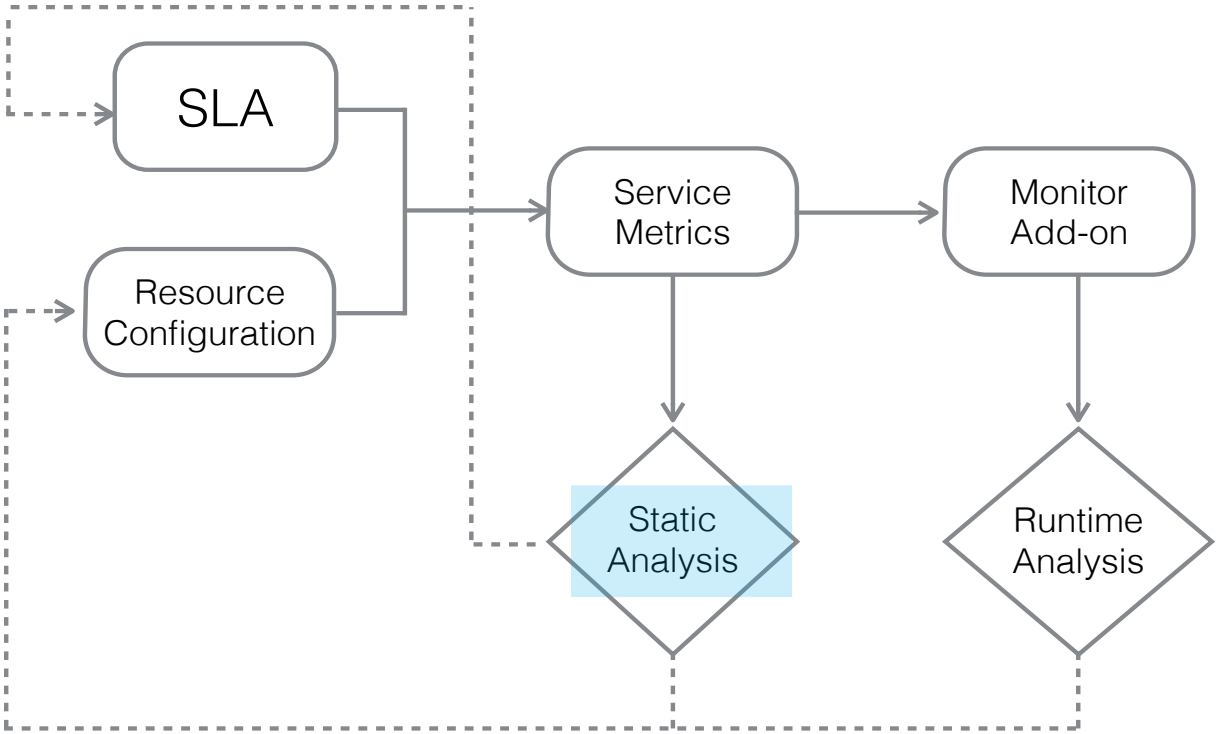
The Analysis Flow



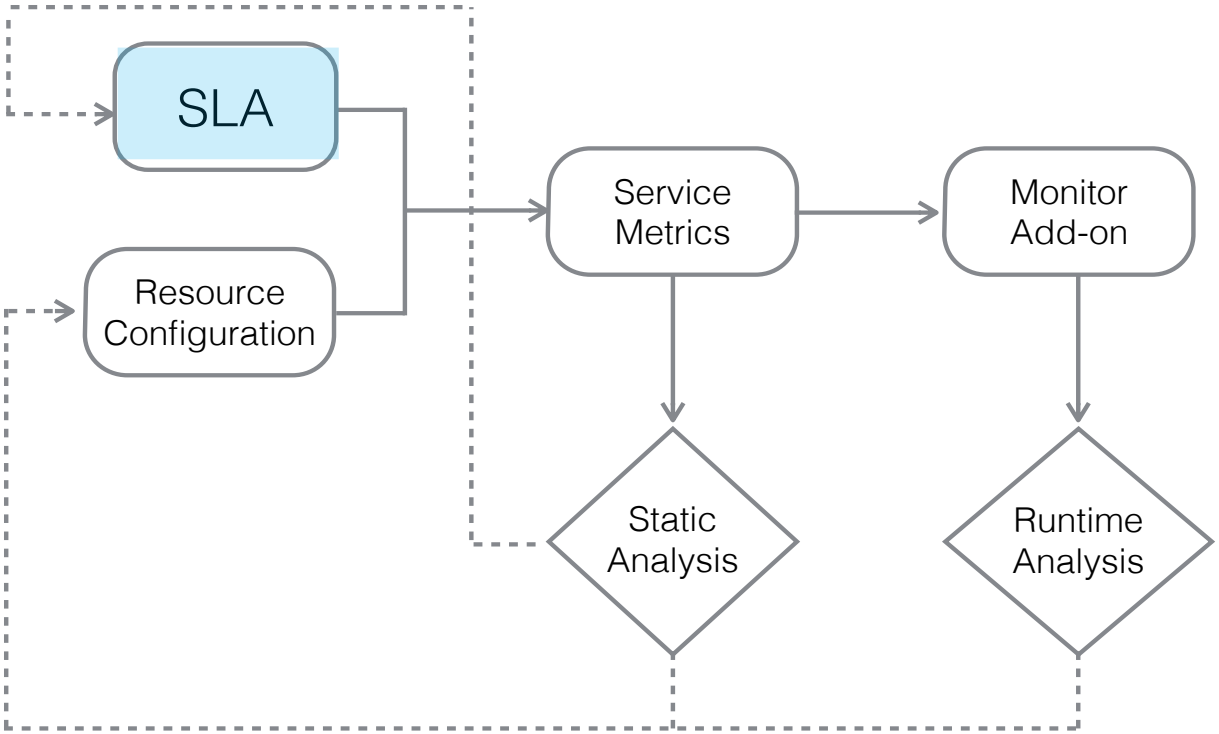
The Analysis Flow



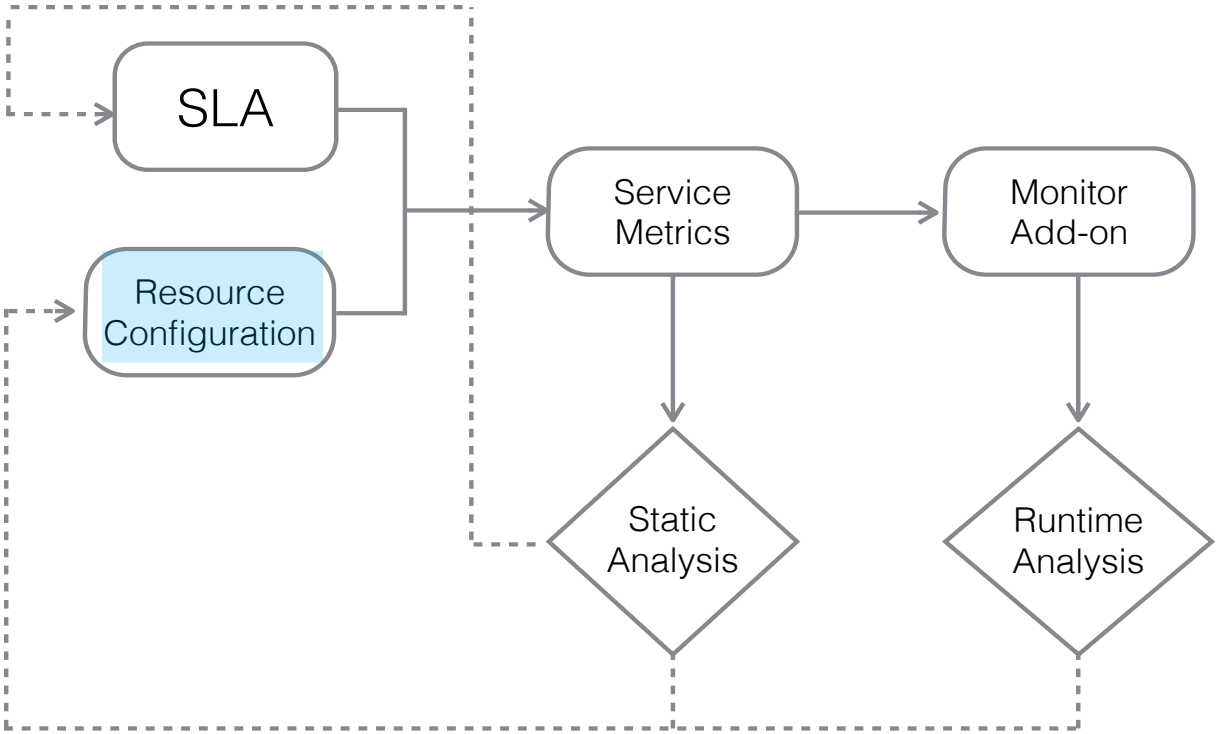
The Analysis Flow



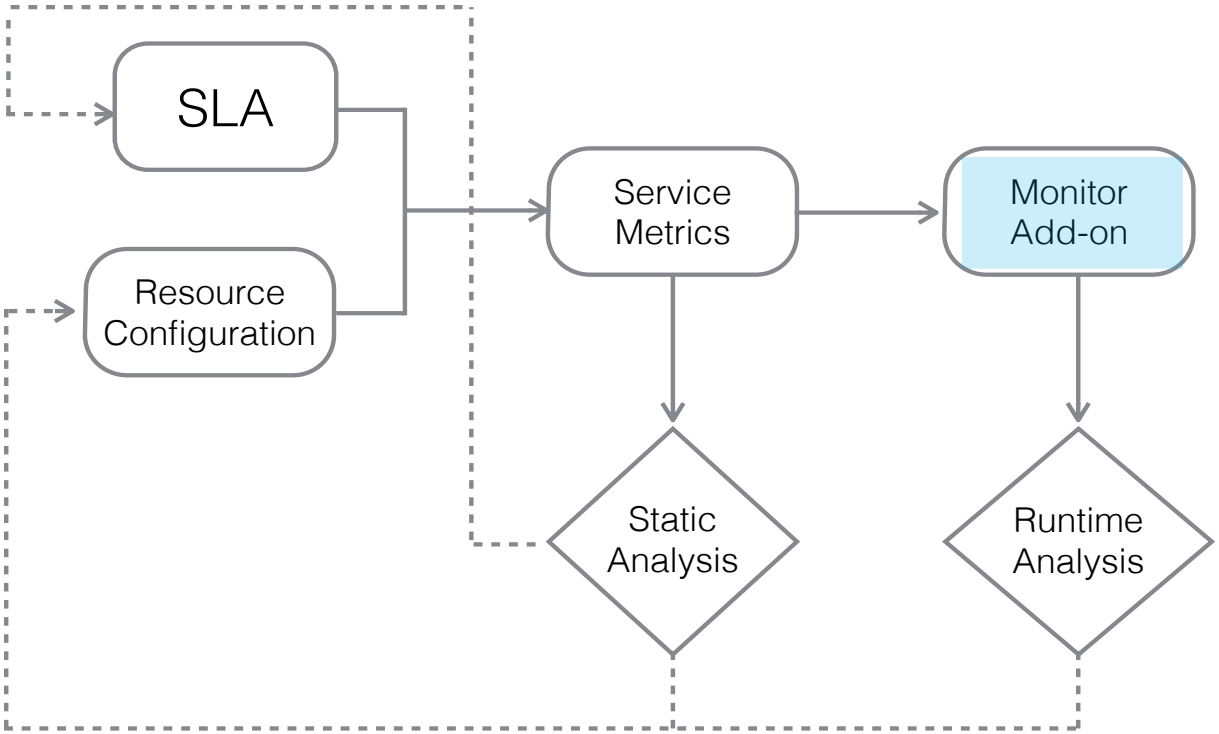
The Analysis Flow



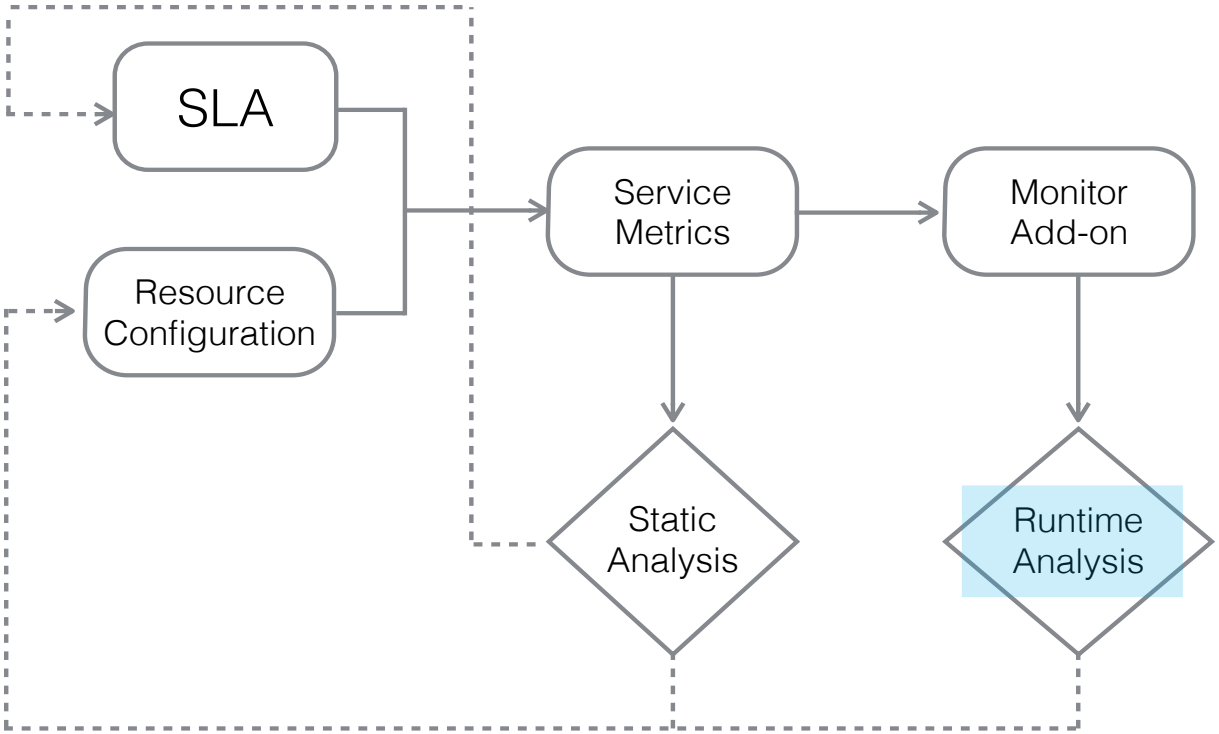
The Analysis Flow



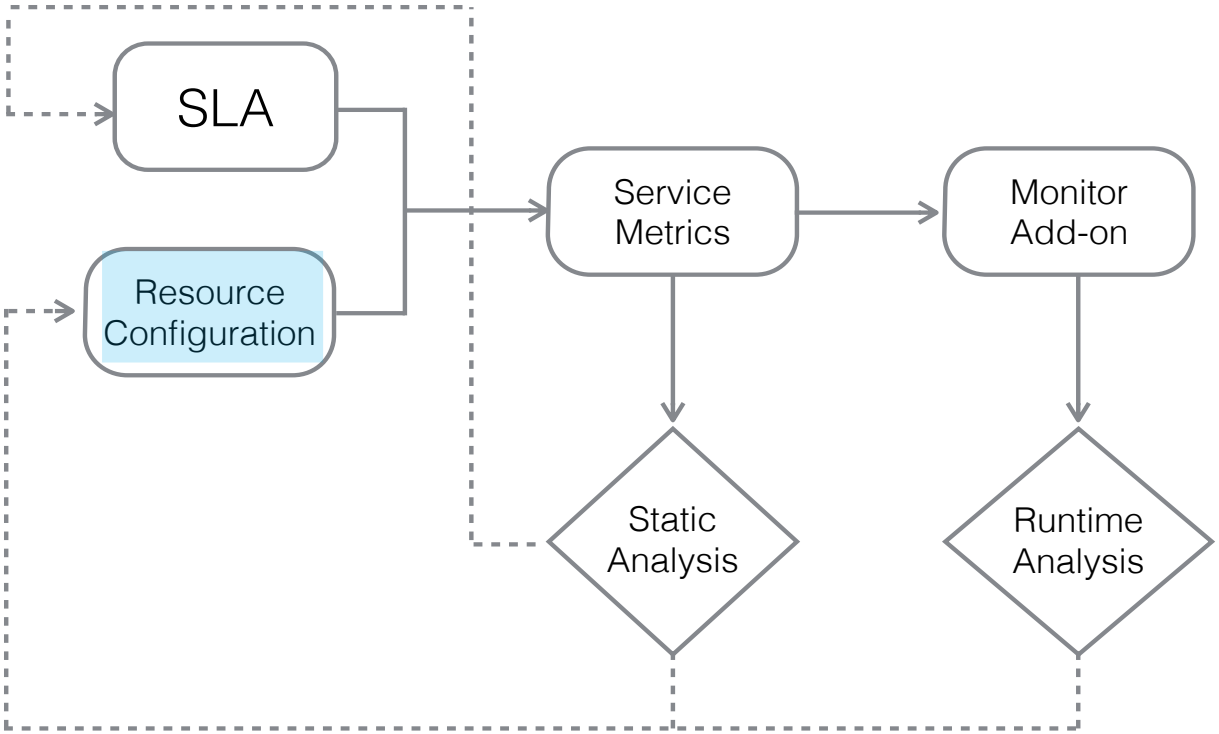
The Analysis Flow



The Analysis Flow



The Analysis Flow



SLAs and performance properties

Metrics for a Query API

"95% of requests is completed within 1 minute, 97% within 3 minutes and 98% within 5 minutes"

AVAILABILITY:

the percentage of timely service provisioning requests

"the service completes 6 queries per minute from 9:00 to 18:00 and 4 queries per minute otherwise"

CAPACITY:

the service throughput

"the service replies to a query request (with the result or with a failure) within 7 minutes"

RESPONSE TIME:

the maximum response time

Metrics for a Query API

"95% of requests is completed within 1 minute, 97% within 3 minutes and 98% within 5 minutes"

AVAILABILITY:

the percentage of timely service provisioning requests

"the service completes 6 queries per minute from 9:00 to 18:00 and 4 queries per minute otherwise"

CAPACITY:

the service throughput

"the service replies to a query request (with the result or with a failure) within 7 minutes"

RESPONSE TIME:

the maximum response time

Metrics for a Query API

"95% of requests is completed within 1 minute, 97% within 3 minutes and 98% within 5 minutes"

AVAILABILITY:

the percentage of timely service provisioning requests

"the service completes 6 queries per minute from 9:00 to 18:00 and 4 queries per minute otherwise"

CAPACITY:

the service throughput

"the service replies to a query request (with the result or with a failure) within 7 minutes"

RESPONSE TIME:

the maximum response time

Metrics for a Query API

"95% of requests is completed within 1 minute, 97% within 3 minutes and 98% within 5 minutes"

AVAILABILITY:

the percentage of timely service provisioning requests

"the service completes 6 queries per minute from 9:00 to 18:00 and 4 queries per minute otherwise"

CAPACITY:

the service throughput

"the service replies to a query request (with the result or with a failure) within 7 minutes"

RESPONSE TIME:

the maximum response time

SLA – Performance metrics

[1] Cloud service level agreement standardisation guidelines (June 2014)

SLA – Performance metrics

Other metrics [1]

security, data management, personal data protection

[1] Cloud service level agreement standardisation guidelines (June 2014)

SLA – Performance metrics

AVAILABILITY:

CAPACITY:

RESPONSE TIME:

Other metrics [1]

security, data management, personal data protection

[1] Cloud service level agreement standardisation guidelines (June 2014)

SLA – Performance metrics

AVAILABILITY:

the property of being accessible and usable upon demand

CAPACITY:

RESPONSE TIME:

Other metrics [1]

security, data management, personal data protection

[1] Cloud service level agreement standardisation guidelines (June 2014)

SLA – Performance metrics

AVAILABILITY:

the property of being accessible and usable upon demand

CAPACITY:

the maximum amount of some property of a cloud service.

RESPONSE TIME:

Other metrics [1]

security, data management, personal data protection

[1] Cloud service level agreement standardisation guidelines (June 2014)

SLA – Performance metrics

AVAILABILITY:

the property of being accessible and usable upon demand

CAPACITY:

the maximum amount of some property of a cloud service.

RESPONSE TIME:

the time interval between a cloud service customer event and a cloud service provider response event

Other metrics [1]

security, data management, personal data protection

[1] Cloud service level agreement standardisation guidelines (June 2014)

Metrics' formalisation: Service metric functions

Service metric functions

- We formalise SLA metrics by means of service metric functions
- Assume $F(\tau)$ and $G(\tau, \delta)$ are two metric functions
- τ is a time interval of the form $[\mathtt{d}.t,\ \mathtt{d}'.t']$
  - $\mathtt{d}$, $\mathtt{d}'$ are days ($\mathtt{d}, \mathtt{d}' \in \{1, \cdots, 365\}$)
  - t, t' are seconds in the day ($t, t' \in \{0, \cdots, 86399\}$)
- δ can be
  - an upper bound to the datasize (in bytes), or
  - a time bound for getting a reply, or
  - an upper bound to the number of resources used by the service.

Percentage of timely service provisioning requests (PTS)

From the SLA

95% of requests is completed within 1 minute, 97% within 3 minutes and 98% within 5 minutes.

Formally

$$\texttt{PTS}_s([1.0,\ 365.86399], x) = \begin{cases} 0,95 & \text{if } x = 60s \\ 0,97 & \text{if } x = 180s \\ 0,98 & \text{if } x = 300s \end{cases}$$

Service Throughput (ST)

From the SLA

the service completes 6 queries per minute from 9:00 to 18:00 and 4 queries per minute otherwise.

Formally

$$\mathtt{ST}_s([1.t,\ 365.t'], 60) = \begin{cases} 4 & \text{if } t = 0 \text{ and } t' = 32399 \\ 6 & \text{if } t = 32400 \text{ and } t' = 64800 // 9am - 6pm \\ 4 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

Response Time (RT)

From the SLA

the service replies to a query request (with the result or with a failure) within 7 minutes.

Formally

$$\mathtt{RT}_s([1.0,\ 365.86399]) = 420s$$

Static analysis

Static analysis

Behavioural types

- Abstract descriptions of programs
- Highlight the information for a particular property.
  - service time
  - number of virtual machines

Approach

1. Extract behavioural types from a program
2. Translate ① into abstract descriptions, e.g., equations
3. Solve ② to get, e.g., upper bound of service time

Static analysis

Behavioural types

- Abstract descriptions of programs
- Highlight the information for a particular property.
  - service time
  - number of virtual machines

Approach

1. Extract behavioural types from a program
2. Translate ① into abstract descriptions, e.g., equations
3. Solve ② to get, e.g., upper bound of service time

Static analysis

Behavioural types

- Abstract descriptions of programs
- Highlight the information for a particular property.
  - service time
  - number of virtual machines

Approach

1. Extract behavioural types from a program
2. Translate ① into abstract descriptions, e.g., equations
3. Solve ② to get, e.g., upper bound of service time

Static analysis

Behavioural types

- Abstract descriptions of programs
- Highlight the information for a particular property.
  - service time
  - number of virtual machines

Approach

1. Extract behavioural types from a program
2. Translate ① into abstract descriptions, e.g., equations
3. Solve ② to get, e.g., upper bound of service time

Example

```
String searchDB(Database DB,
    String s)
{
  String u;
  u = DB.query(s) ;
  job(h) ;
  ......
  return u;
}
```

- `job(h)` needs `h` CPU resources
- the time is $\frac{\text{needed CPU resources}}{\text{allocated CPU resources}}$

Example

```
String searchDB(Database DB,
    String s)
{
  String u;
  u = DB.query(s) ;
  job(h) ;
  ......
  return u;
}
```

Behavioural type:

```
Service.searchDB(a[x], b[y])
{


 DataBase.query(b[y]) ;
 h/x ;

} : _
```

Example

```
String searchDB(DataBase DB,
    String s)
{
  String u;
  u = DB.query(s) ;
  job(h) ;
  ......
  return u;
}

class DataBase {
 String query(String s) {
  .....
  job(k);
 }
}
```

Behavioural type:

```
Service.searchDB(a[x], b[y]){

 DataBase.query(b[y]) ;
 h/x ;

} : _

DataBase.query(b[y]) {

 k/y
} : _
```

Cost Equations

$searchDB(x,y) = query(y) + h/x$

$query(y) = k/y$

Example

```
String searchDB(DataBase DB,
     String s)
{
  String u;
  u = DB.query(s) ;
  job(h) ;
  ......
  return u;
}

class DataBase {
 String query(String s) {
  .....
  job(k);
 }
}
```

Behavioural type:

```
Service.searchDB(a[x], b[y]){

 DataBase.query(b[y]) ;
 h/x ;

} : _

DataBase.query(b[y]) {

 k/y
} : _
```

Cost Equations

$$searchDB(x,y) = query(y) + h/x$$

$$query(y) = k/y$$

Example

Cost Equations

$$\texttt{searchDB}(x,y) = \texttt{query}(y) + h/x$$

$$\texttt{query}(y) = k/y$$

Example

Cost Equations

searchDB(x,y) = query(y) + h/x

query(y) = k/y

Time estimation

$$\frac{h}{x} + \frac{k}{y}$$

Example

Cost Equations

```
searchDB(x,y) = query(y) + h/x
query(y) = k/y
```

Time estimation

$$\frac{15}{x} + \frac{5}{y}$$

Example

Cost Equations

$$\texttt{searchDB}(x,y) = \texttt{query}(y) + h/x$$

$$\texttt{query}(y) = k/y$$

Time estimation

$$\frac{15}{2} + \frac{5}{1}$$

Example

Cost Equations

$$\texttt{searchDB}(x,y) = \texttt{query}(y) + h/x$$

$$\texttt{query}(y) = k/y$$

Time estimation

$$\frac{15}{2} + \frac{5}{1} = 12.5$$

Example

Cost Equations

$\texttt{searchDB}(x,y) = \texttt{query}(y) + \texttt{h}/x$

$\texttt{query}(y) = \texttt{k}/y$

Time estimation

$$\frac{15}{2} + \frac{5}{1} = 12.5$$

$$\texttt{ST}_{\texttt{searchDB}}([1.t,\ 365.t'], 60) = \begin{cases} 4 & \text{if } t = 0 \text{ and } t' = 32399 \\ 6 & \text{if } t = 32400 \text{ and } t' = 64800 \ //9am - 6pm \\ 4 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

$$\texttt{RT}_{\texttt{searchDB}}([1.0,\ 365.86399]) = 420s$$

Example

Cost Equations

searchDB(x,y) = query(y) + h/x

query(y) = k/y

Time estimation

$$\frac{15}{2} + \frac{5}{1} = 12.5$$

$$\mathtt{ST}_{\mathtt{searchDB}}([1.t,\ 365.t'], 60) = \begin{cases} 4 & \text{if } t = 0 \text{ and } t' = 32399 \\ 6 & \text{if } t = 32400 \text{ and } t' = 64800 \ //9am - 6pm \\ 4 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

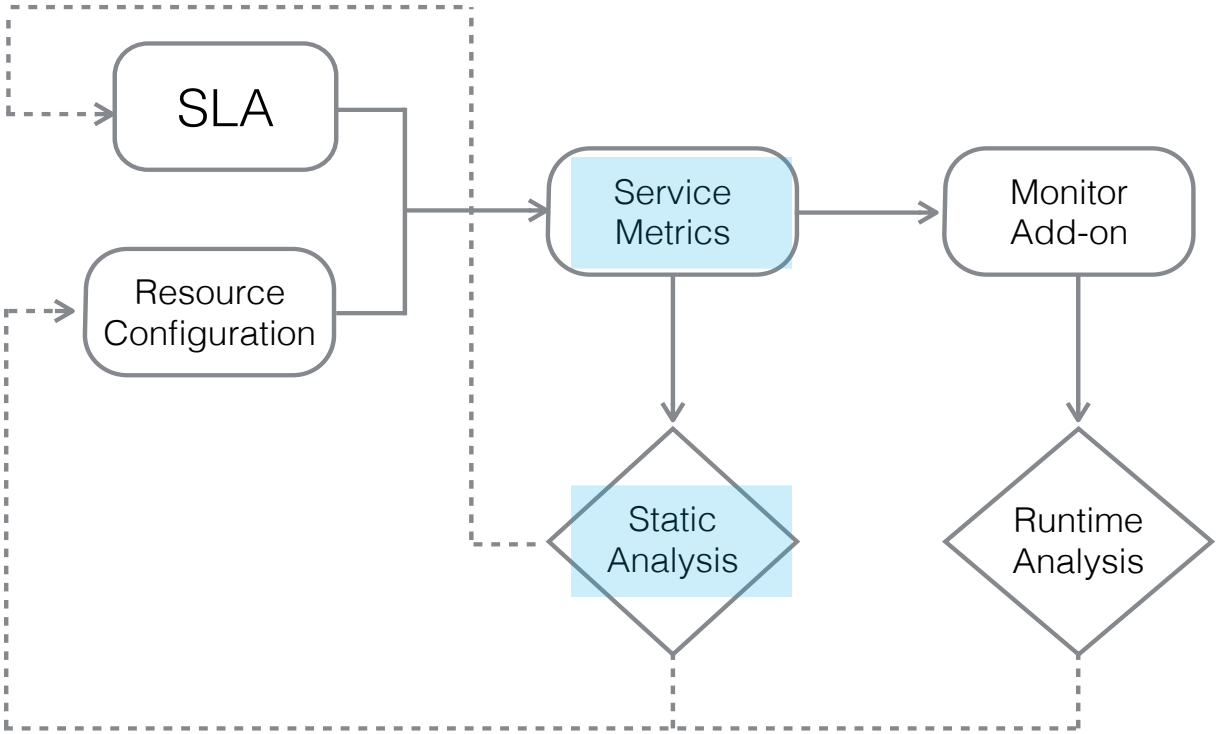
$$\mathtt{RT}_{\mathtt{searchDB}}([1.0,\ 365.86399]) = 420s$$

Break $\mathtt{ST}_{\mathtt{searchDB}}$

Adjustment

Time estimation

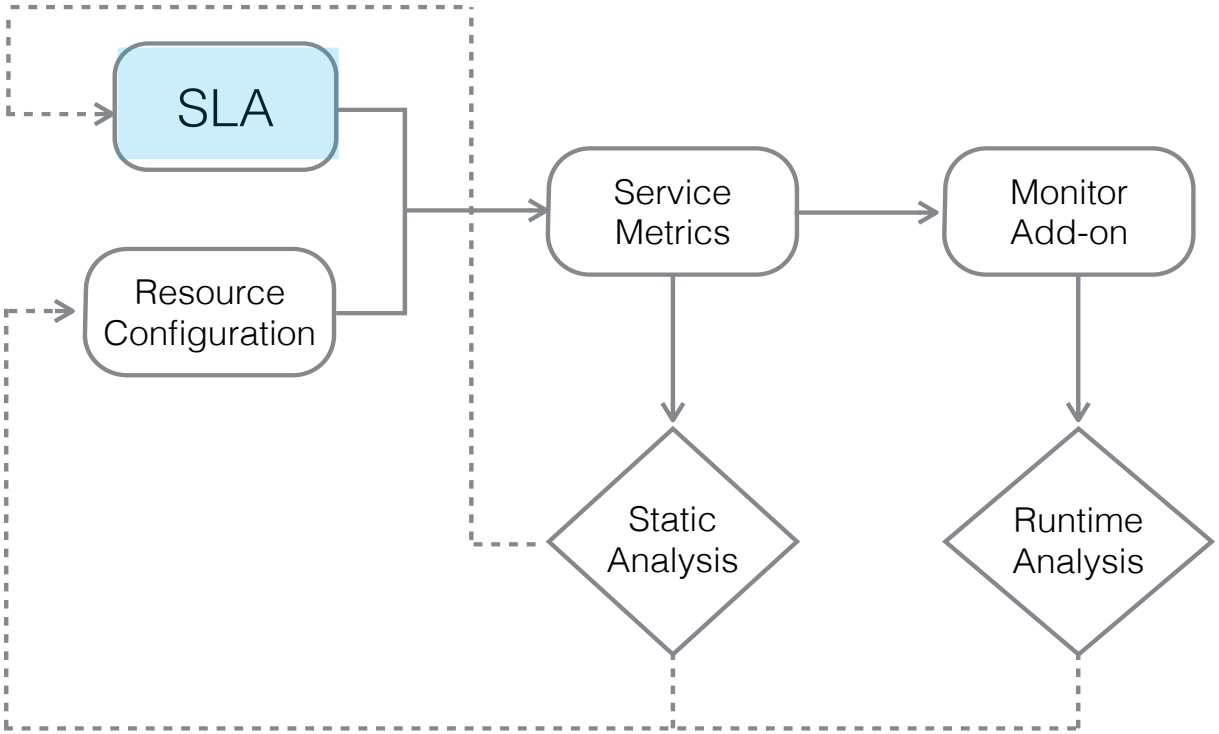
$$\frac{15}{2} + \frac{5}{1} = 12.5$$



Adjustment

Time estimation

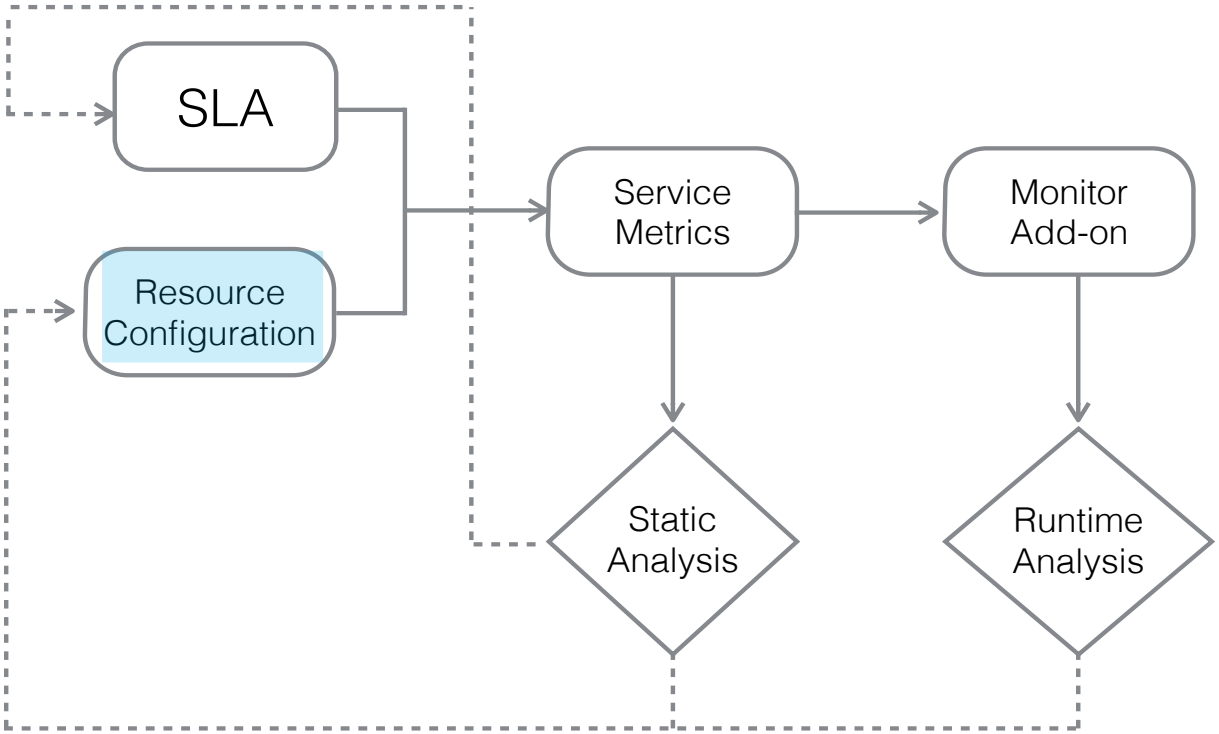
$$\frac{15}{2} + \frac{5}{1} = 12.5$$



Adjustment

Time estimation

$$\frac{15}{2} + \frac{5}{2} = 10$$



Adjustment

Time estimation

$$\frac{15}{2} + \frac{5}{2} = 10$$

$$\mathtt{ST}_{\mathtt{searchDB}}([1.t,\ 365.t'], 60) = \begin{cases} 4 & \text{if } t = 0 \text{ and } t' = 32399 \\ 6 & \text{if } t = 32400 \text{ and } t' = 64800 \ //9am - 6pm \\ 4 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

$$\mathtt{RT}_{\mathtt{searchDB}}([1.0,\ 365.86399]) = 420s$$

Adjustment

Time estimation

$$\frac{15}{2} + \frac{5}{2} = 10$$

$$\mathtt{ST}_{\mathtt{searchDB}}([1.t,\ 365.t'], 60) = \begin{cases} 4 & \text{if } t = 0 \text{ and } t' = 32399 \\ 6 & \text{if } t = 32400 \text{ and } t' = 64800 \ //9am - 6pm \\ 4 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

$$\mathtt{RT}_{\mathtt{searchDB}}([1.0,\ 365.86399]) = 420s$$

Satisfy both

Run-time analysis

Monitoring metrics

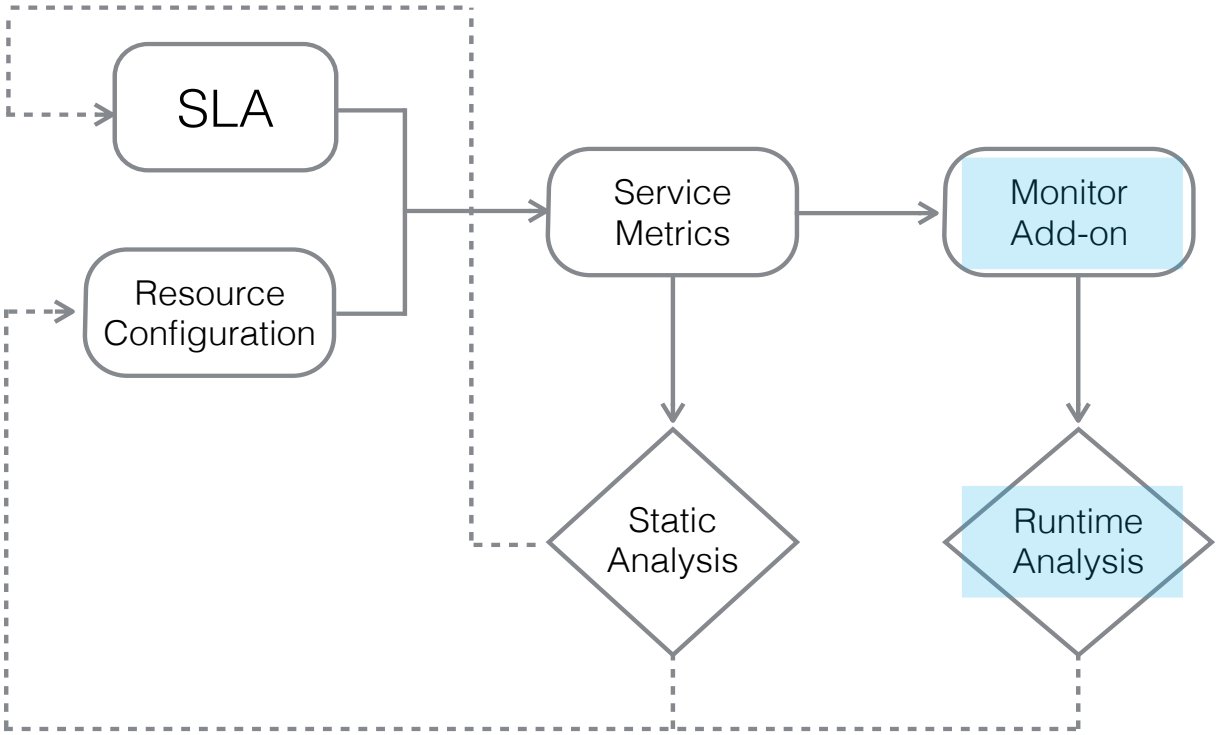
Run-time monitoring

- Observation:
- Reaction:

Monitoring metrics

Run-time monitoring

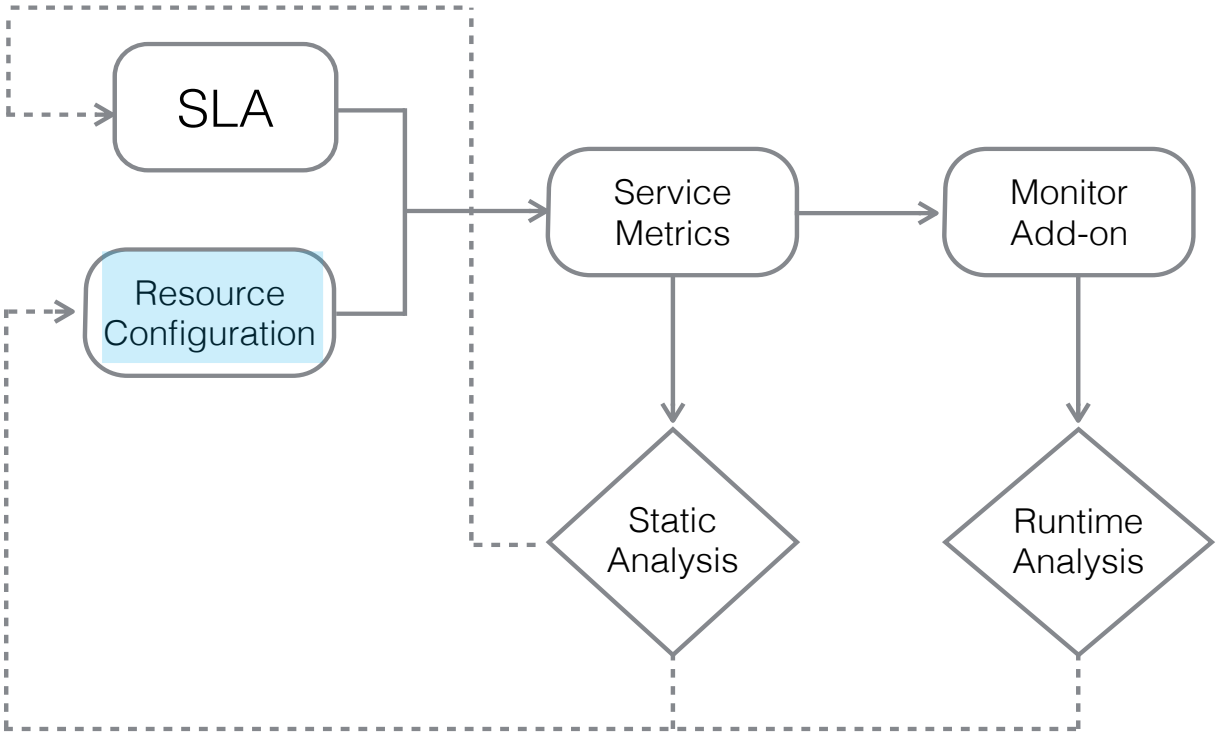
- Observation: measurements on services are taken
- Reaction:



Monitoring metrics

Run-time monitoring

- Observation: measurements on services are taken
- Reaction: adjustments for resource allocations



Observation phase

Service time monitor

- Intercepts all the HTTP invocations/replies to/from a service
- Records the time taken by every request to be completed

```
void monitor_service_time() {
   (service,method,msg,client,m_id) = HttpRequest.intercept();
   time_start = time();
   reply = service.method(msg);
   time_end = time();
   HttpResponse.send(client,reply,m_id);
   log(m_id,time_start,time_end);
}
```

Percentage of timely service provisioning requests

`percentage` method

takes as input a time window and returns `true` if the percentage of requests complies with the definition of $\texttt{PTS}_s$

```
boolean percentage(Log_file d, Time t_begin, Time t_end){
   boolean v = true ;

   /* retrieve from the log the total number of messages served in the time window */
   nmb_msg = get_total_messages(d, t_begin, t_end) ;

   /* check whether the SLA percentages of served requests
      correspond to the observed ones */
    nmb_msg_completed = find(t_begin, t_end, 60) ;
    v = v && (nmb_msg_completed/nmb_msg <= 0.95) ; //95% in 1 min
    nmb_msg_completed = find(t_begin, t_end, 180) ;
    v = v && (nmb_msg_completed/nmb_msg <= 0.97) ; //97% in 3 mins
    nmb_msg_completed = find(t_begin, t_end, 300) ;
    v = v && (nmb_msg_completed/nmb_msg <= 0.98) ; //98% in 5 mins

   return v;
}
```

Service Throughput

`throughput` method

takes as input a logfile and a time window and returns `true` if the throughput complies with the definition of ST_s

```
boolean throughput(Log_file d, Time t_begin, Time t_end){
    int daily = 0;
    int nightly = 0;

    /* collects the number of the served requests during the two
      specified time-frames */
    for each (m_id, time_init, time_end) in d {
      if ((time_init >= 32400) && (time_init <= 64800)) // 9:00—18:00
            daily = daily + 1 ;
      else nightly = nightly + 1 ;
    }

    /* return true if 6 queries per minute are completed in 9:00-18:00
       and 4 queries per minute in the remaining time */
    return ( ((daily/60*9)>6) && ((nightly/60*15)>4) );
  }
```

Example

$$\mathtt{ST}_{\mathtt{searchDB}}([1.t,\ 365.t'], 60) = \begin{cases} 4 & \text{if } t = 0 \text{ and } t' = 32399 \\ 6 & \text{if } t = 32400 \text{ and } t' = 64800 \ //9am - 6pm \\ 4 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

$$\mathtt{RT}_{\mathtt{searchDB}}([1.0,\ 365.86399]) = 420s$$

- [Static Analysis] The initial 2 CPU resources respect the SLA.
- The `throughput` method records only 4 requests per minute.

Example

$$\mathtt{ST}_{\mathtt{searchDB}}([1.t,\ 365.t'], 60) = \begin{cases} 4 & \text{if } t = 0 \text{ and } t' = 32399 \\ 6 & \text{if } t = 32400 \text{ and } t' = 64800 \ //9am - 6pm \\ 4 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

$$\mathtt{RT}_{\mathtt{searchDB}}([1.0,\ 365.86399]) = 420s$$

- [Static Analysis] The initial 2 CPU resources respect the SLA. ☺
- The `throughput` method records only 4 requests per minute. ☹

Example

$$\mathtt{ST}_{\mathtt{searchDB}}([1.t,\ 365.t'], 60) = \begin{cases} 4 & \text{if } t = 0 \text{ and } t' = 32399 \\ 6 & \text{if } t = 32400 \text{ and } t' = 64800 \ //9am - 6pm \\ 4 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

$$\mathtt{RT}_{\mathtt{searchDB}}([1.0,\ 365.86399]) = 420s$$

- [Static Analysis] The initial 2 CPU resources respect the SLA.
- The `throughput` method records only 4 requests per minute.

Example

$$\mathtt{ST}_{\mathtt{searchDB}}([1.t,\ 365.t'], 60) = \begin{cases} 4 & \text{if } t = 0 \text{ and } t' = 32399 \\ 6 & \text{if } t = 32400 \text{ and } t' = 64800 \ //9am - 6pm \\ 4 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

$$\mathtt{RT}_{\mathtt{searchDB}}([1.0,\ 365.86399]) = 420s$$

- [Static Analysis] The initial 2 CPU resources respect the SLA.
- The `throughput` method records only 4 requests per minute.

Reaction

Configure resource allocation

Reaction phase

`reaction` method

verifies every 300s whether the percentage of timely service provisioning requests is reached and, in case of failures, adds one more CPU

```
void reaction(Service s) {
  Time t ; Bool v ;
  t = time() ;
  idle(300) ;
  v = percentage(d,t, t+300) ;
  if (!v) MonitoringPlatform ! allocate(s) ;
}
```

Example

$$\texttt{ST}_{\texttt{searchDB}}([1.t,\ 365.t'], 60) = \begin{cases} 4 & \text{if } t = 0 \text{ and } t' = 32399 \\ 6 & \text{if } t = 32400 \text{ and } t' = 64800\ //9am - 6pm \\ 4 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

$$\texttt{RT}_{\texttt{searchDB}}([1.0,\ 365.86399]) = 420s$$

- [Static Analysis] The initial 2 CPU resources respect the SLA.
- The `throughput` method records only 4 requests per minute.
- The `reaction` method requests 1 additional CPU resource.
- Waste of money and resources during the night.

Example

$$\texttt{ST}_{\texttt{searchDB}}([1.t,\ 365.t'], 60) = \begin{cases} 4 & \text{if } t = 0 \text{ and } t' = 32399 \\ 6 & \text{if } t = 32400 \text{ and } t' = 64800\ //9am - 6pm \\ 4 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

$$\texttt{RT}_{\texttt{searchDB}}([1.0,\ 365.86399]) = 420s$$

- [Static Analysis] The initial 2 CPU resources respect the SLA.
- The `throughput` method records only 4 requests per minute.
- The `reaction` method requests 1 additional CPU resource.
- Waste of money and resources during the night.

Example

$$\mathtt{ST}_{\mathtt{searchDB}}([1.t,\ 365.t'], 60) = \begin{cases} 4 & \text{if } t = 0 \text{ and } t' = 32399 \\ 6 & \text{if } t = 32400 \text{ and } t' = 64800\ //9am - 6pm \\ 4 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

$$\mathtt{RT}_{\mathtt{searchDB}}([1.0,\ 365.86399]) = 420s$$

- [Static Analysis] The initial 2 CPU resources respect the SLA.
- The `throughput` method records only 4 requests per minute.
- The `reaction` method requests 1 additional CPU resource.
- Waste of money and resources during the night.

Example

Budget metric function

$$\texttt{Budget}_{\texttt{searchDB}}([1.t,\ 365.t']) = \begin{cases} 40 & \text{if } t = 0 \text{ and } t' = 32399 \\ 60 & \text{if } t = 32400 \text{ and } t' = 64800 // 9am - 6pm \\ 40 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

- Each CPU resource costs 10
- The static analysis will approve $\texttt{Budget}_{\texttt{searchDB}}$.
- When the runtime CPU reallocation is triggered by the `throughput` monitor, the nightly budget is not met anymore.
- The `budget_monitor` reacts by requiring a deallocation of half of the CPU units during the night.

Example

Budget metric function

$$\texttt{Budget}_{\texttt{searchDB}}([1.t,\ 365.t']) = \begin{cases} 40 & \text{if } t = 0 \text{ and } t' = 32399 \\ 60 & \text{if } t = 32400 \text{ and } t' = 64800 // 9am - 6pm \\ 40 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

- Each CPU resource costs 10
- The static analysis will approve $\texttt{Budget}_{\texttt{searchDB}}$.
- When the runtime CPU reallocation is triggered by the `throughput` monitor, the nightly budget is not met anymore.
- The `budget_monitor` reacts by requiring a deallocation of half of the CPU units during the night.

Example

Budget metric function

$$\texttt{Budget}_{\texttt{searchDB}}([1.t,\ 365.t']) = \begin{cases} 40 & \text{if } t = 0 \text{ and } t' = 32399 \\ 60 & \text{if } t = 32400 \text{ and } t' = 64800 // 9am - 6pm \\ 40 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

- Each CPU resource costs 10
- The static analysis will approve $\texttt{Budget}_{\texttt{searchDB}}$. 
- When the runtime CPU reallocation is triggered by the `throughput` monitor, the nightly budget is not met anymore.
- The `budget_monitor` reacts by requiring a deallocation of half of the CPU units during the night.

Example

Budget metric function

$$\texttt{Budget}_{\texttt{searchDB}}([1.t,\ 365.t']) = \begin{cases} 40 & \text{if } t = 0 \text{ and } t' = 32399 \\ 60 & \text{if } t = 32400 \text{ and } t' = 64800 // 9am - 6pm \\ 40 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

- Each CPU resource costs 10
- The static analysis will approve $\texttt{Budget}_{\texttt{searchDB}}$. 
- When the runtime CPU reallocation is triggered by the `throughput` monitor, the nightly budget is not met anymore. 
- The `budget_monitor` reacts by requiring a deallocation of half of the CPU units during the night.

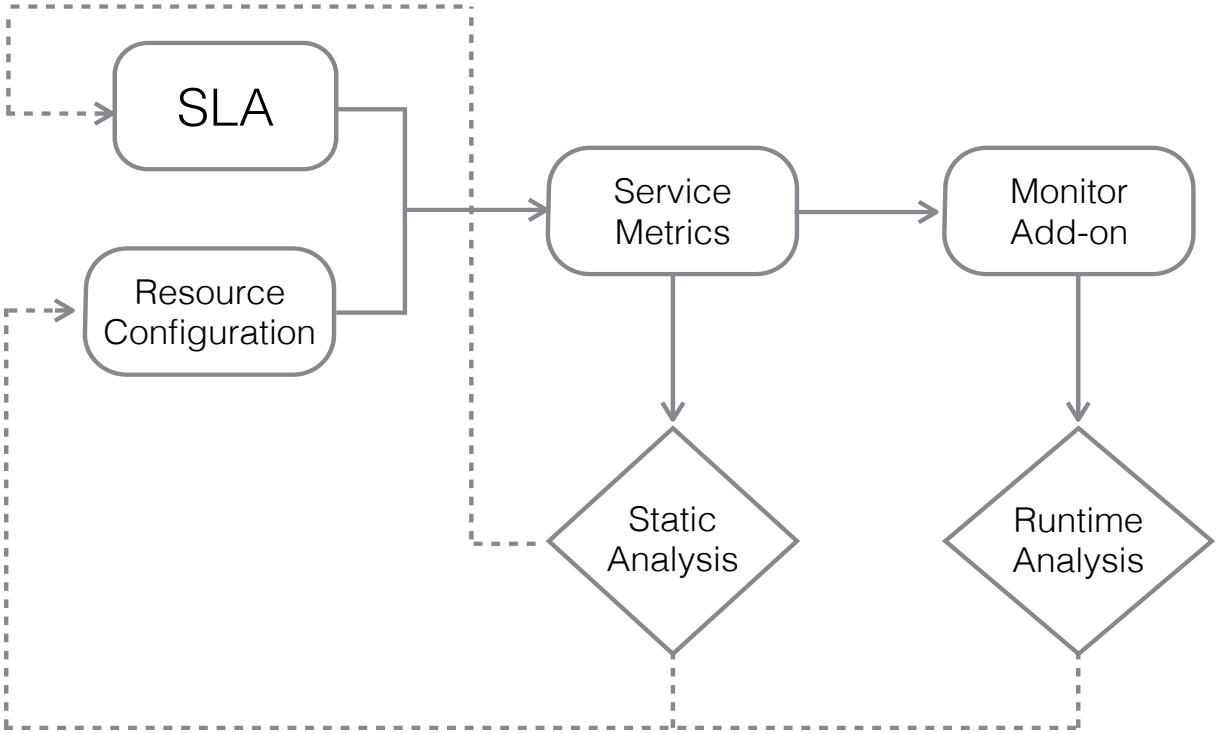
Example

Budget metric function

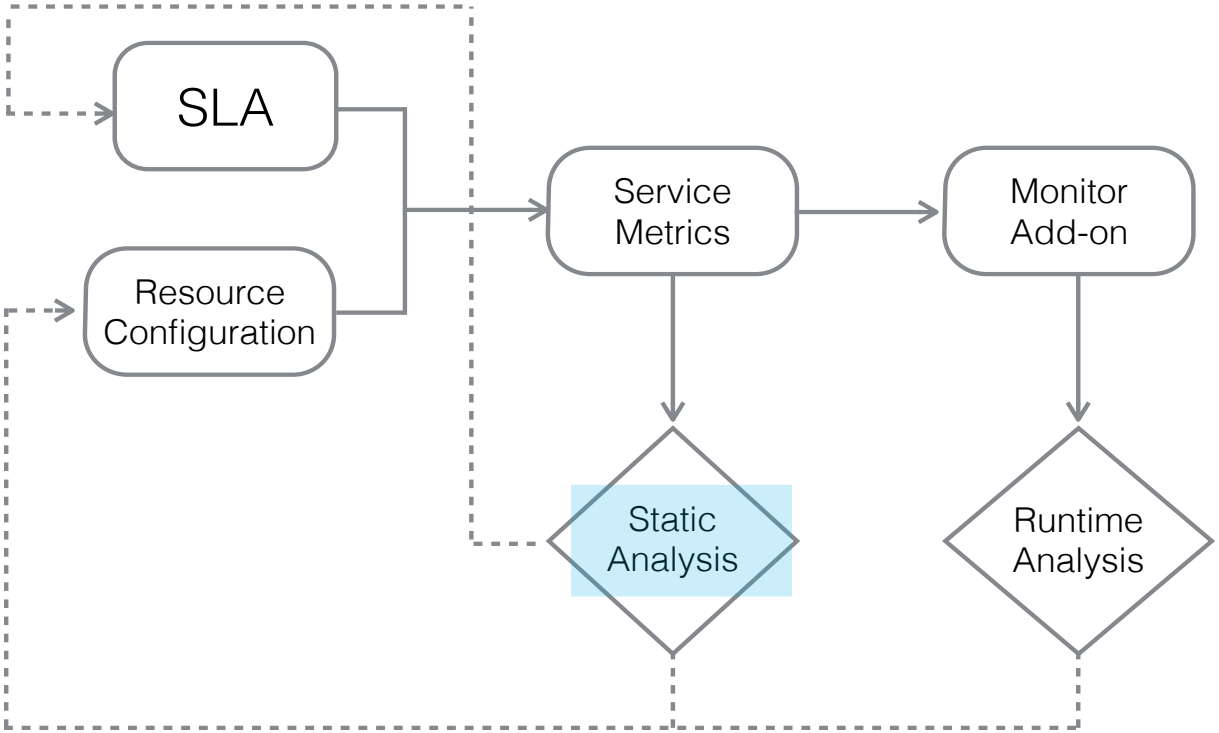
$$\texttt{Budget}_{\texttt{searchDB}}([1.t,\ 365.t']) = \begin{cases} 40 & \text{if } t = 0 \text{ and } t' = 32399 \\ 60 & \text{if } t = 32400 \text{ and } t' = 64800 // 9am - 6pm \\ 40 & \text{if } t = 64801 \text{ and } t' = 86399 \end{cases}$$

- Each CPU resource costs 10
- The static analysis will approve $\texttt{Budget}_{\texttt{searchDB}}$. ☺
- When the runtime CPU reallocation is triggered by the `throughput` monitor, the nightly budget is not met anymore. ☹
- The `budget_monitor` reacts by requiring a deallocation of half of the CPU units during the night. ☺

Summary

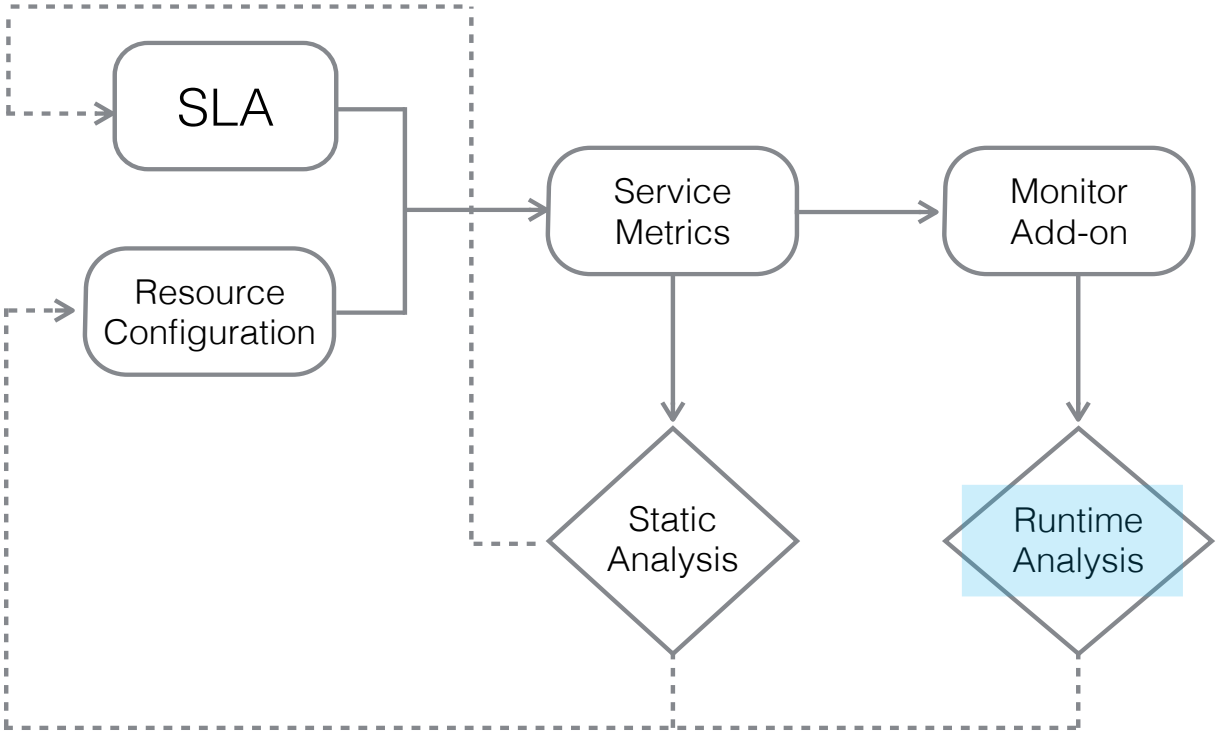

SLA
Resource Configuration
Service Metrics
Monitor Add-on
Static Analysis
Runtime Analysis

Summary



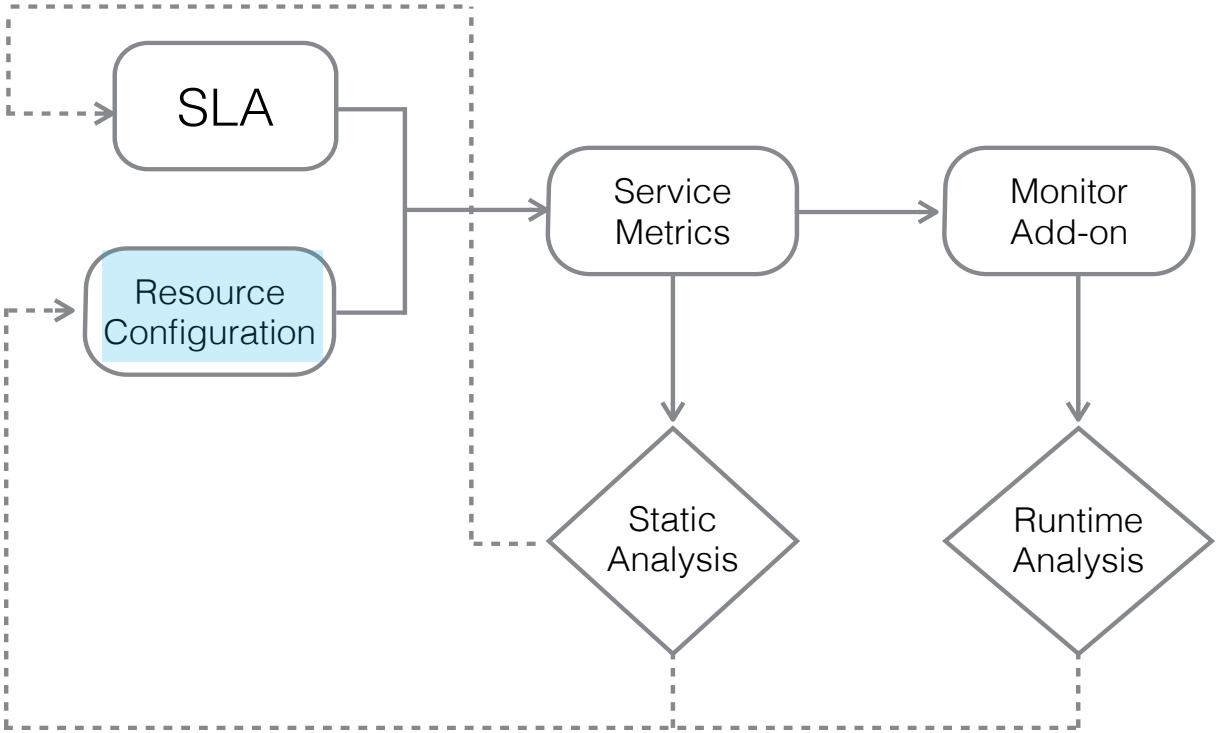
- Maximum number of virtual machines
- Upper bound of response time

Summary



- Availability
- Budget compliances

Summary



- A monitoring platform for resource (de-)allocation.

Summary

Static Analysis

- Static analysis of cloud elasticity. (PPDP'15).
  A. Garcia, C. Laneve, M. Lienhardt
- Time Complexity of Concurrent Programs. (FACS'15).
  E. Giachino, E. B. Johnsen, C. Laneve, K. I Pun

Runtime monitoring

- Formal verification of service level agreements through distributed monitoring. (ESOCC'15).
  B. Nobakht, S. de Gouw, and F. S. de Boer

SLA metric approach

- Statically and Dynamically verifiable SLA metrics.
  (Theory and Practice of Formal Methods).
  E. Giachino, S. De Gouw, C. Laneve, and B. Nobakht.