Distributed Software Monitoring

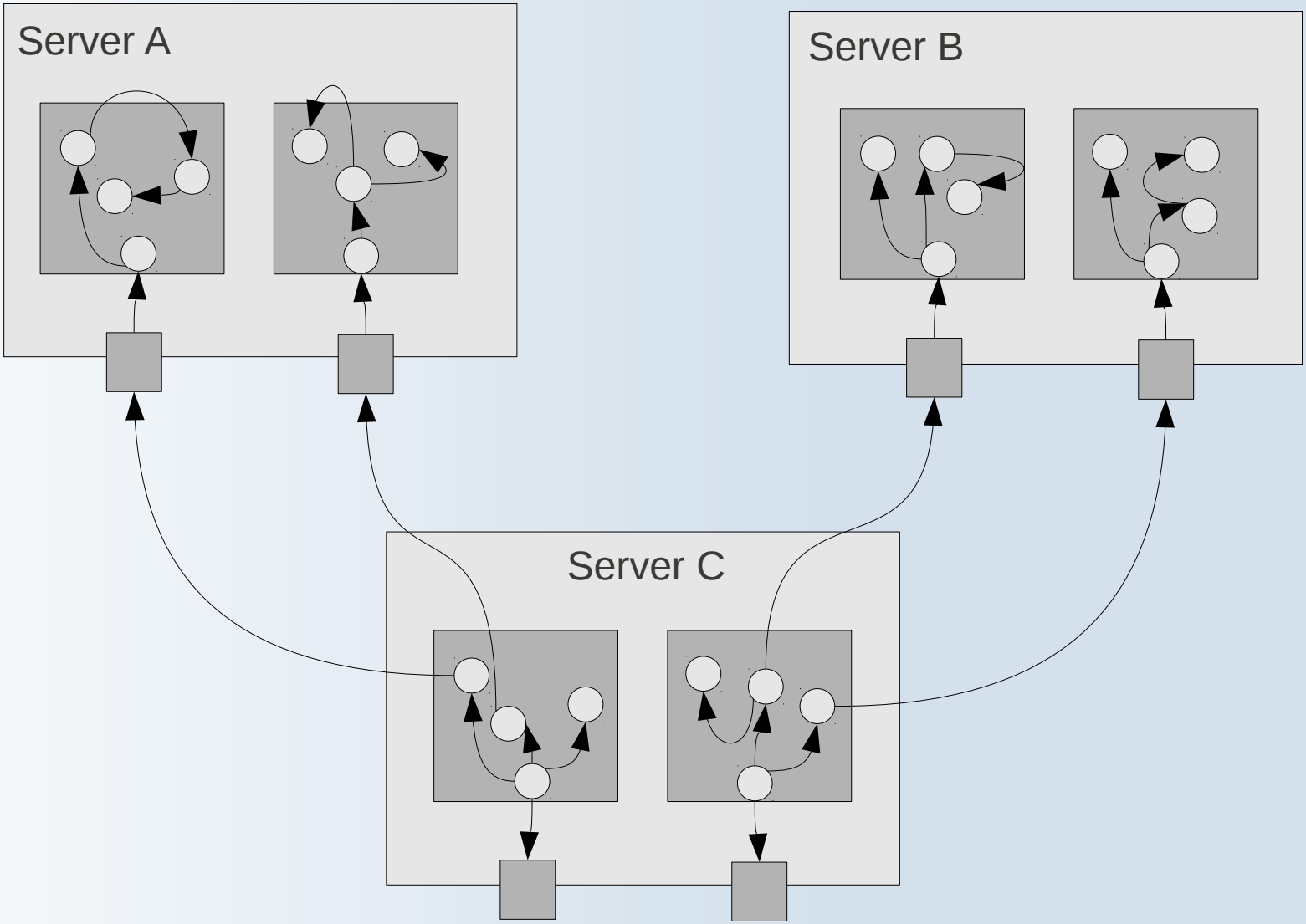
by

Rafael Sobek
1&1 Internet AG

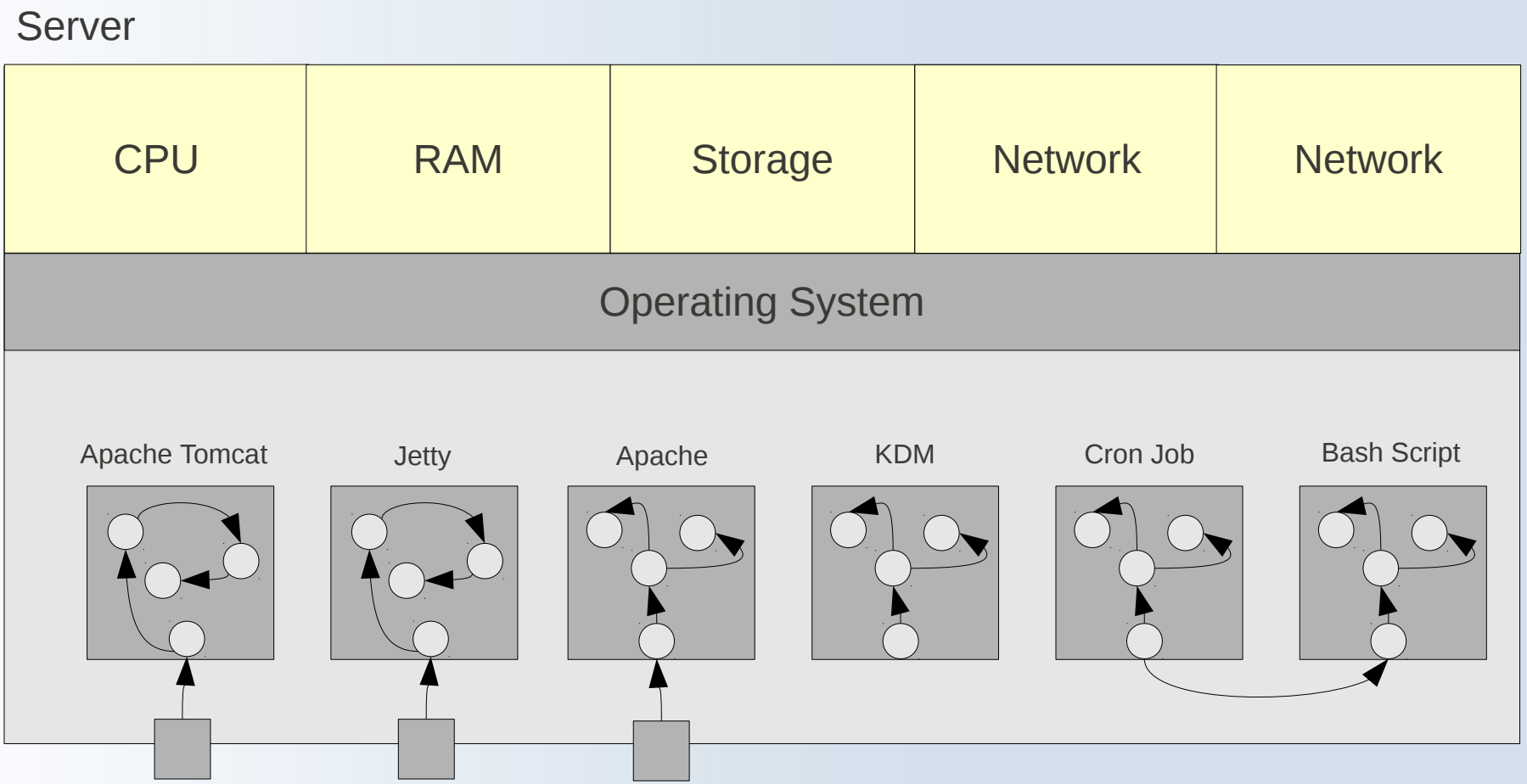
Distributed Software Monitoring – Contents

1. Distributed Complexity
2. Separation from Hardware Monitoring
3. Monitoring Techniques and Methods
3. Target Groups – Outage Activities and Different Views
4. Monitoring Requirements
5. OpenSource Tool Simple Java Monitor
    5.1 Main Architecture
    5.2 In Memory Data Holding
    5.3 Data Model
    5.4 Heuristical Threshold Analysis
    5.5 Dashboard View, Cluster View, Method View, ...
    5.6 Example
6. Other Tools
  6.1 Perf4J,
  6.2 JaMon

Distributed Software Monitoring – Distributed Complexity

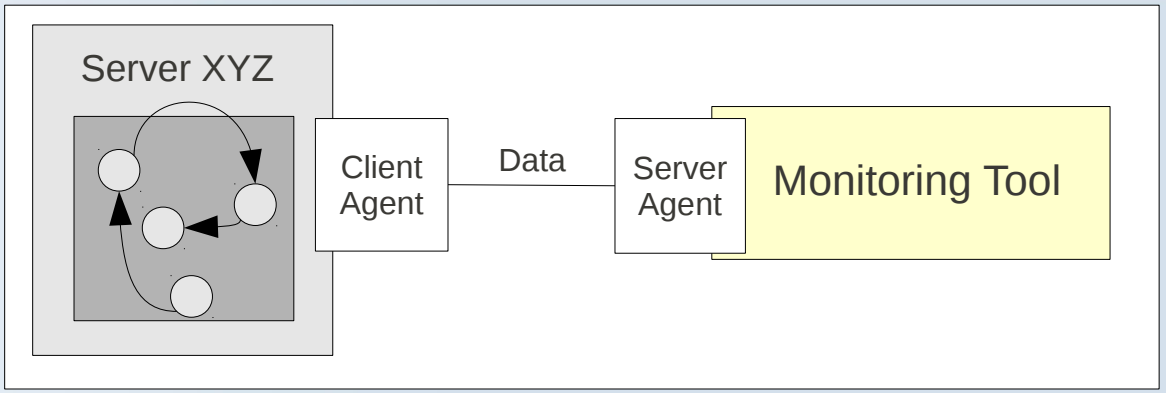


Distributed Software Monitoring - Separation from Hardware Monitoring

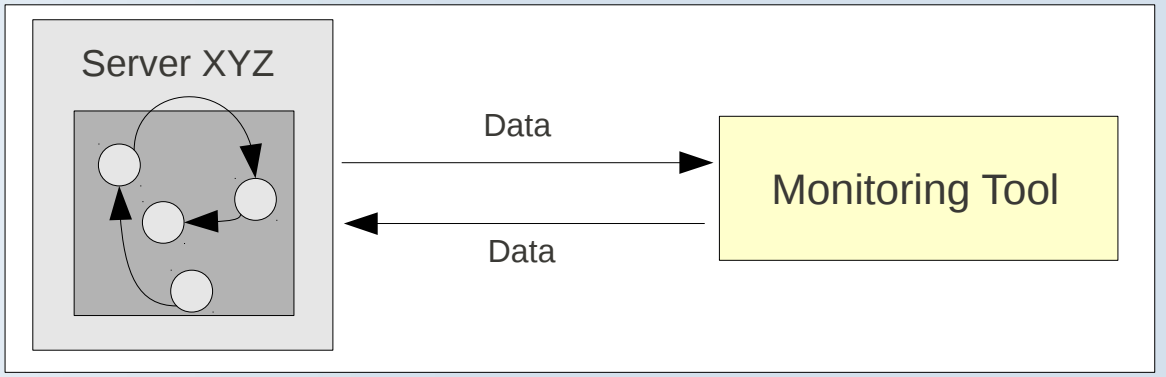


Distributed Software Monitoring – Monitoring Techniques and Methods

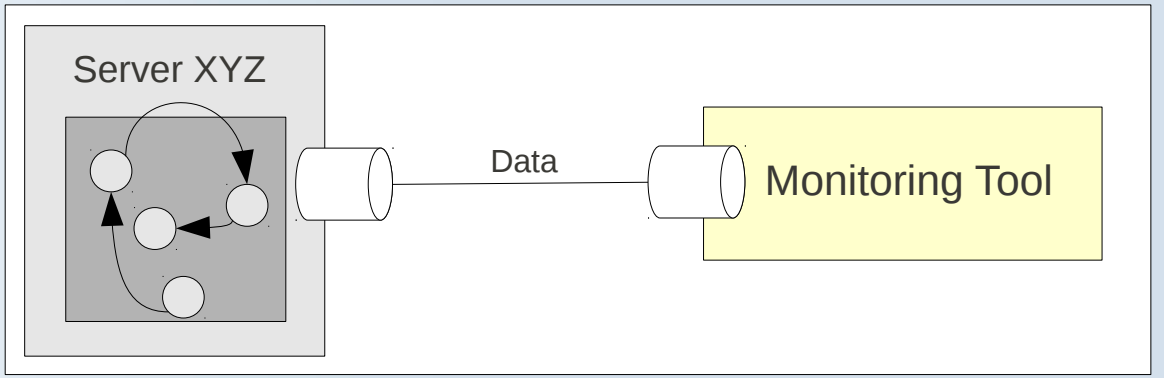
Usage of Agents



Polling or Pushing of Measurements



Usage of Queues



Distributed Software Monitoring – Monitoring Techniques and Methods

Manually Data Acquisition

```
StopWatch stopWatch = new LoggingStopWatch("codeBlock1");

Thread.sleep((long)(Math.random() * 1000L));

stopWatch.stop();
```

Annotated Data Acquisition

```
    @Monitored
    public static void doSomething() throws Exception {
        ....
    }
```

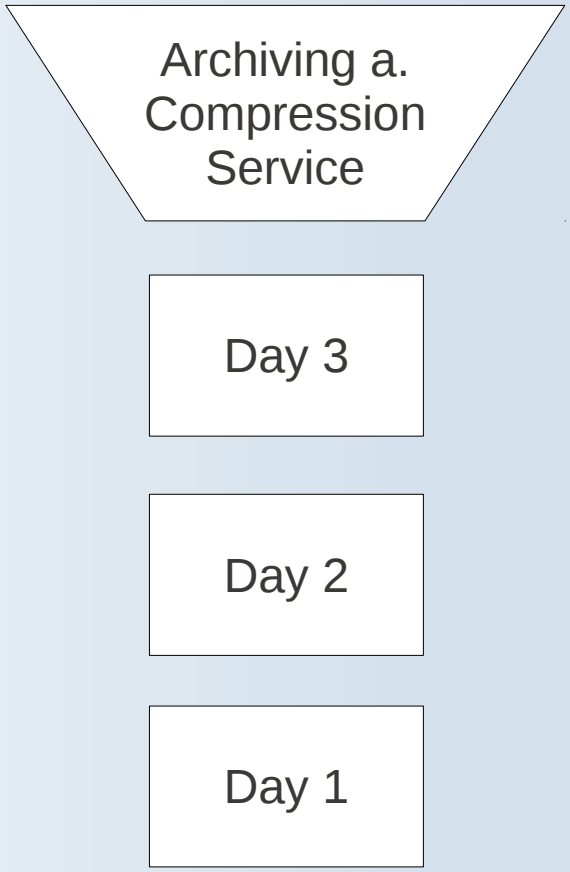
AOP based Data Aquisation

```
@Aspect
public class ProfilingAspect {

    @Around("methodsToBeProfiled()")
    public Object profile(ProceedingJoinPoint pjp) throws Throwable {
        StopWatch sw = new StopWatch(getClass().getSimpleName());
        try {
            sw.start(pjp.getSignature().getName());
            return pjp.proceed();
        } finally {
            sw.stop();
            System.out.println(sw.prettyPrint());
        }
    }

    @Pointcut("execution(public * *.*(..))")
    public void methodsToBeProfiled(){}
}
```

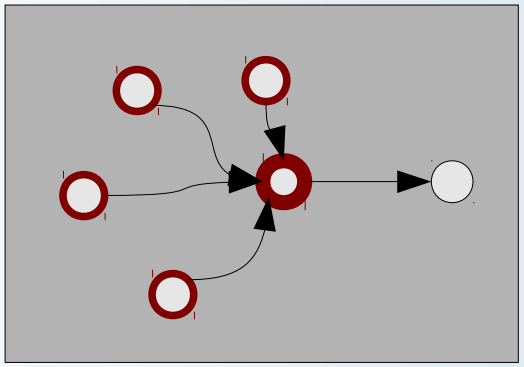
Data Archiving and Compression



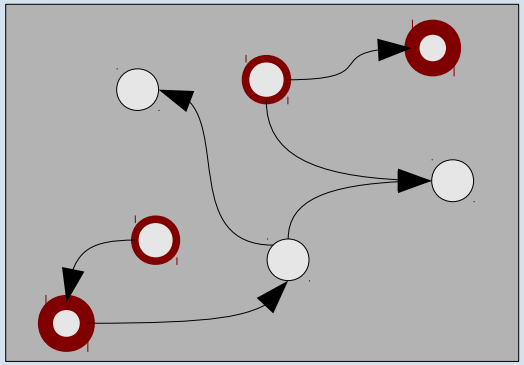
Distributed Software Monitoring – Software Outage Scenarios (Within Component)

Bottleneck

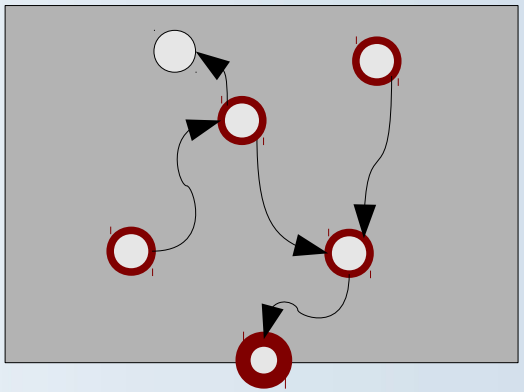
(inappropriate synchronize block)



Bugs



Outage External Service



Distributed Software Monitoring
Software Outage Scenarios (Outside of Component)

Growth of User Retries → Amplifies Outage

Server A (HTML-Frontend)

Server B (HTML-Frontend)

Server C (RMI-Service)

Server D (MySQLs)

Convolutions Intensifier Problem

Sporadic Webpage Timeouts
(e.g. HTTP 500 Error)

Sporadic Service Timeouts
(e.g. RMI Timeout)

Sporadic JDBC Timeouts
(e.g. Bottleneck)

Distributed Software Monitoring – Target Groups – Outage Activities

Which service is affected?

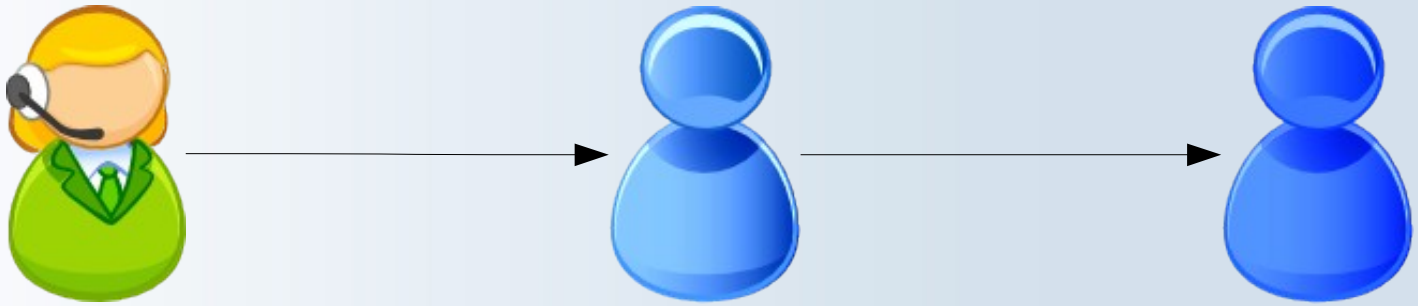
→ Easily States
→ RUNNING, FAILED, ..
→ Service Description
→ Service Dependencies
→ Contact Persons
→ Customer Communication

Analysis of the problem

→ Historical Analysis
→ Cluster Analysis
→ Hardware Problem
→ Software Problem
→ Bug, Bottleneck
→ Security Issue

Identification of concerned program logic

→ Analysis Program Flow Control
→ Analysis Data Flow
→ Exception Analysis
→ Contact Persons



User Help Desk
First-Level-Support

Second-Level-Support

Third-Level-Support
Specialist Department
Developers

In-Time Alarming

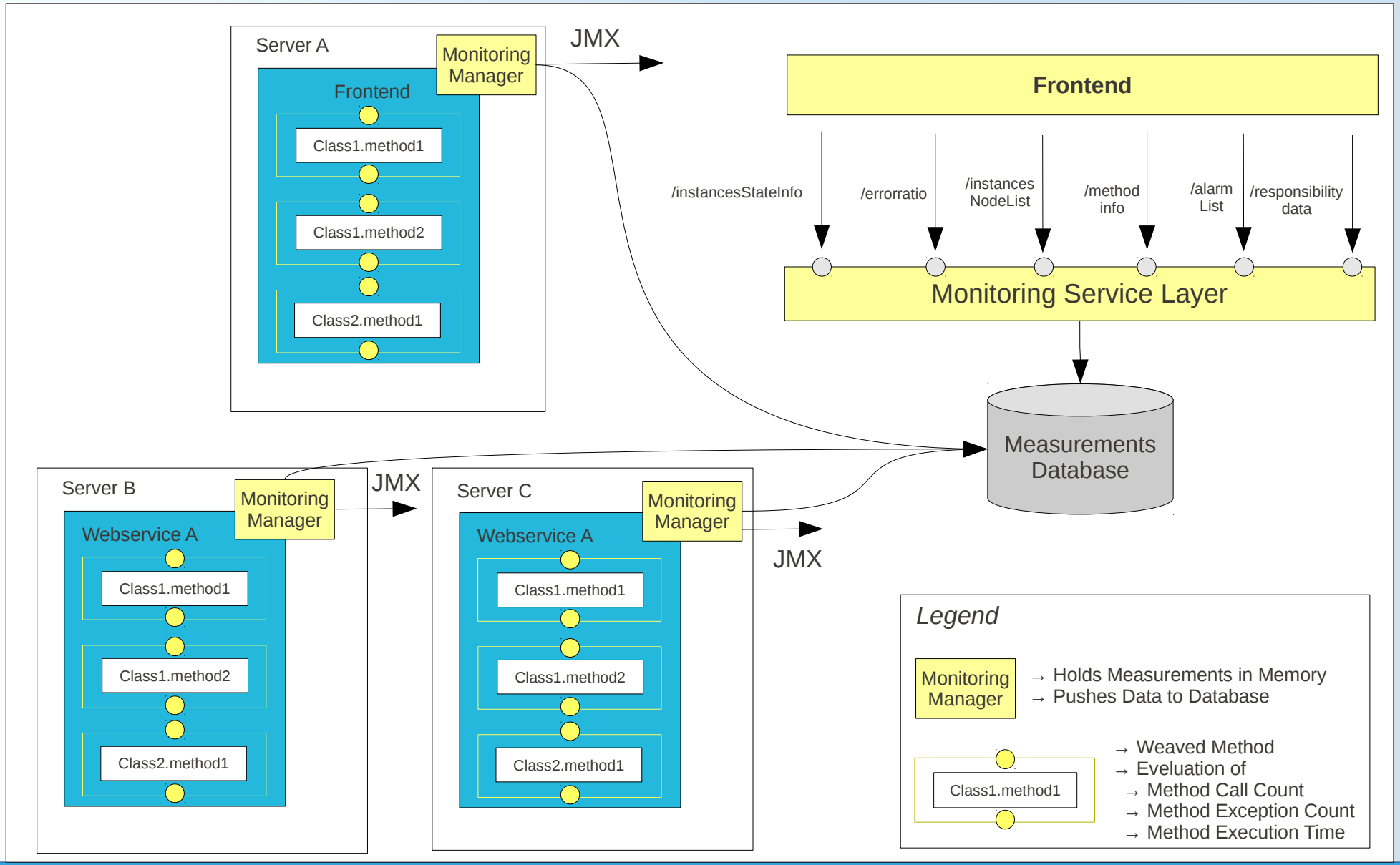
Distributed Software Monitoring – Requirements

- Easy-to-Use → Cost-Effective (e.g. Maven Plugin)
- Low Implementation Effort → Cost-Effective (e.g. AOP)
- Unique Software Component Identifiers → (e.g. Maven GroupId, ArifactId, Version)
- Aggregration and Archiving of Measurement → Resource-Saving
- No Runtime-Effects!
- Target Group-Orientated Views of Measurements (e.g. Developer, Operation, ...)
- Service Layer → Customization and Preparation of Measurements (e.g. Evaluation of SLAs, Management Reports, ...)

Distributed Software Monitoring – Requirements

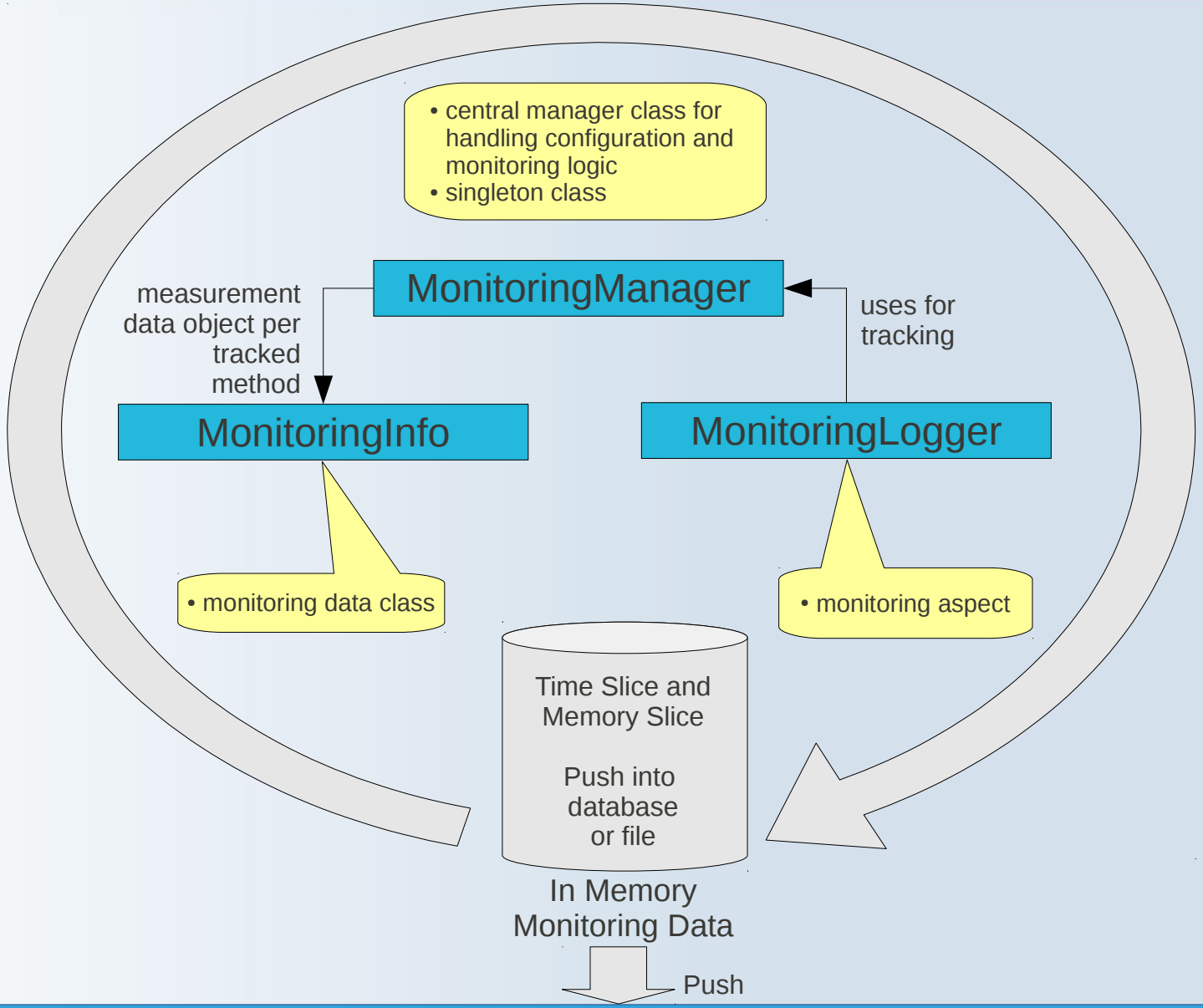
- Reactive Monitoring → On Time Alarming
- Local and Central Holding of Measurements enabled
- Additional Metadata → Software Description, Department, Contact Persons (e.g. Usage of POM description, developers, ...)
- High Availability (e.g. Master-Slave based SQL-Database Cluster, NoSQL Storage Cluster)
- Acquire Detailed Data
  - Count of Method Calls
  - Average Time of Method Calls
  - Count of Exceptions

Distributed Software Monitoring - OpenSource Tool - Simple Monitoring
Main Architecture



Distributed Software Monitoring - OpenSource Tool - Simple Monitoring

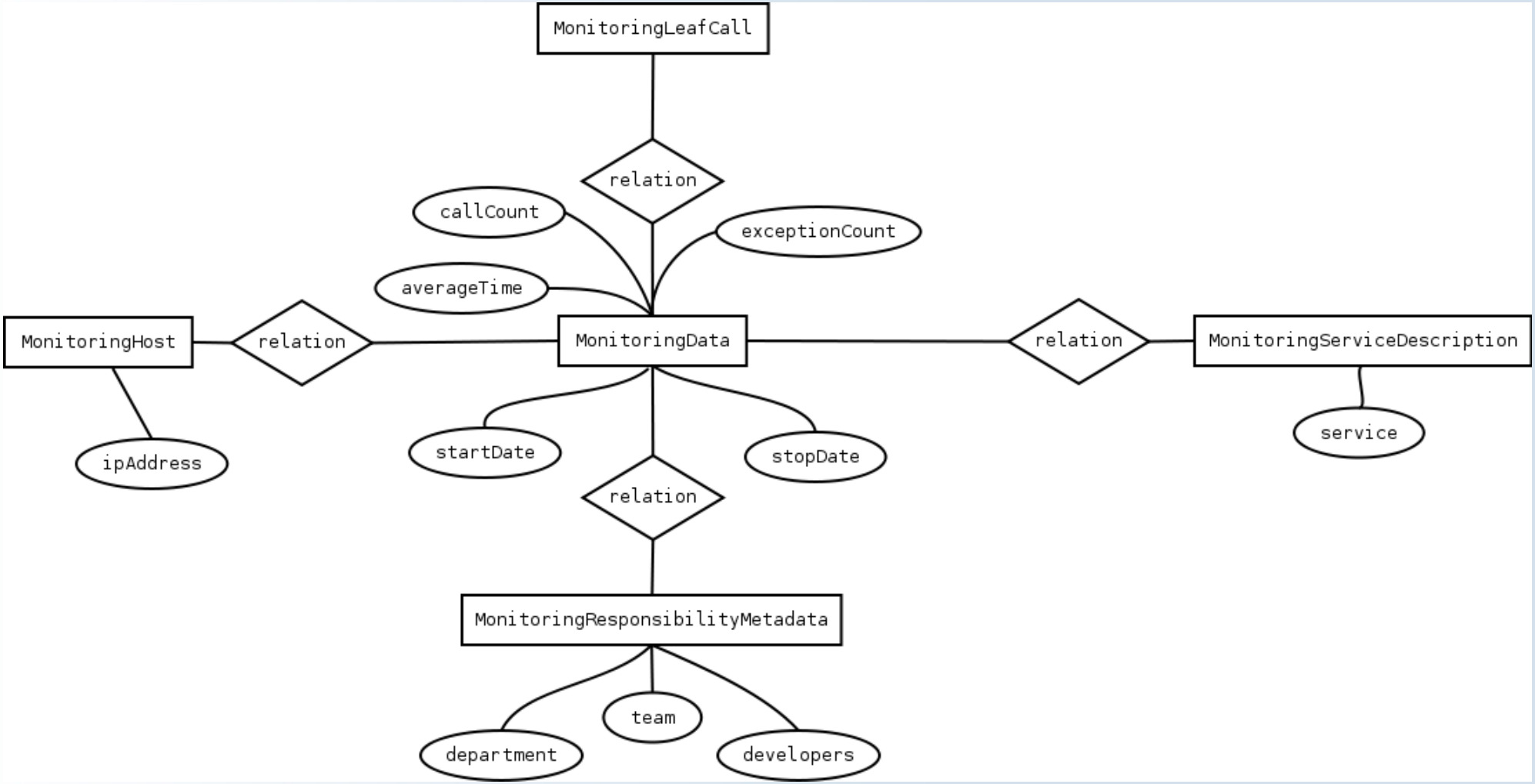
In Memory Data Holding



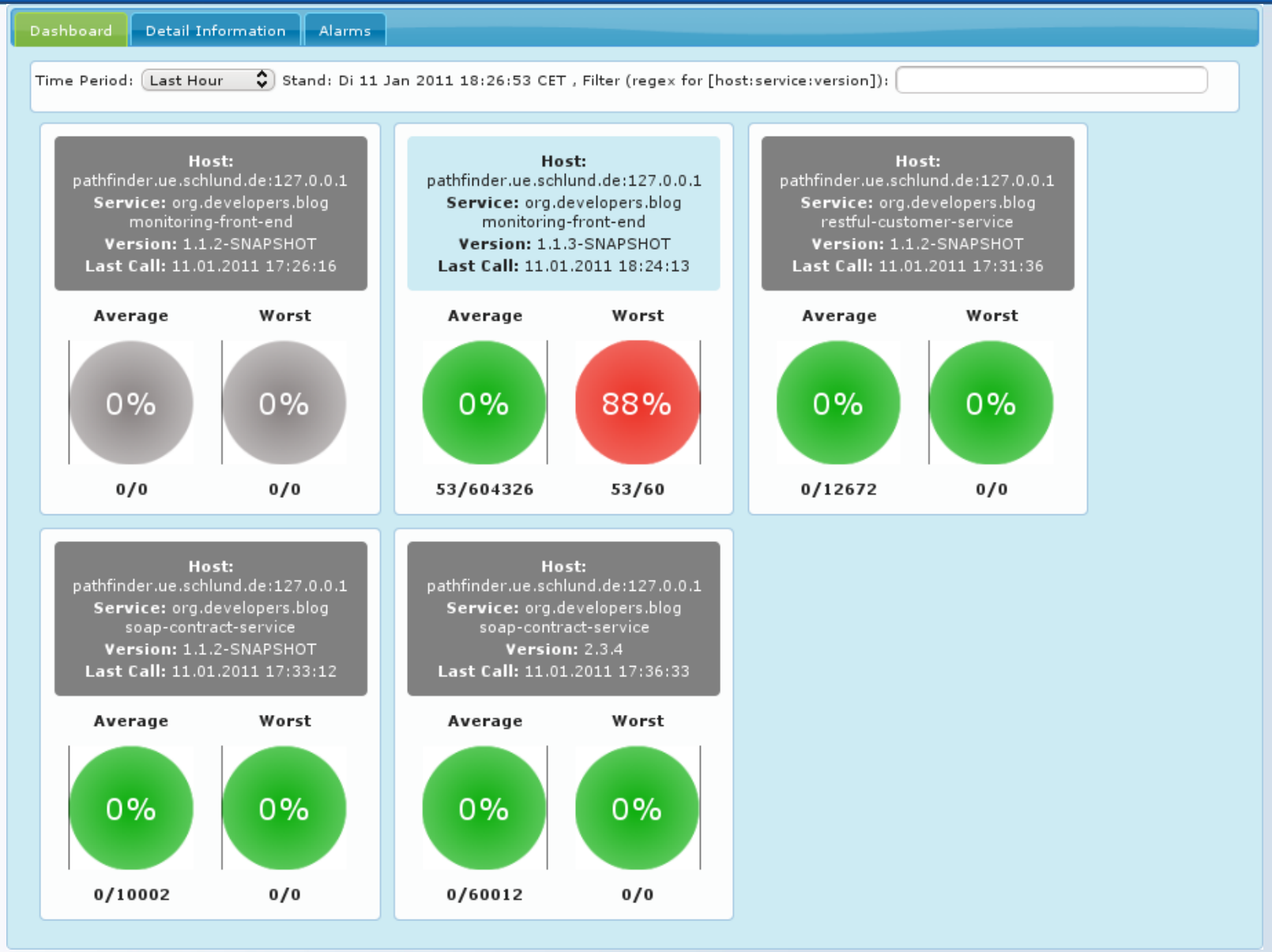
Distributed Software Monitoring - OpenSource Tool – Simple Monitoring Configuration

```
<plugin>
    <groupId>org.developers.blog</groupId>
    <artifactId>monitoring-plugin</artifactId>
    <version>1.1.1</version>
    <configuration>
        <aopExpression>execution(* *.*(..))</aopExpression>
        <useDB>true</useDB>
        <department>My Test Department</department>
        <team>Very Skilled Team</team>
        <developers>f.mercury@domain.com</developers>
        <complianceLevel>1.5</complianceLevel>
        <verbose>true</verbose>
        <debug>true</debug>
        <showWeaveInfo>true</showWeaveInfo>
        <memorySlice>50</memorySlice>
        <timeSlice>20000</timeSlice>
        <maxConnections>10</maxConnections>
        <startThreadsCount>5</startThreadsCount>
        <maxThreadsCount>20</maxThreadsCount>
        <maxQueueCapacity>300</maxQueueCapacity>
        <maxWaitTimeForConnection>60</maxWaitTimeForConnection>
        <connectionPoolDataSourceClass>
           com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource
        </connectionPoolDataSourceClass>
        <dbProperties>
            <dbProperty>
                <name>url</name>>
                <value>jdbc:mysql://localhost/monitordb</value>
            </dbProperty>
            <dbProperty>
                <name>user</name>
                <value>dbuser</value>
            </dbProperty>
            <dbProperty>
                <name>password</name>>
                <value>secret</value>
            </dbProperty>
        </dbProperties>
    </configuration>
</plugin>
```

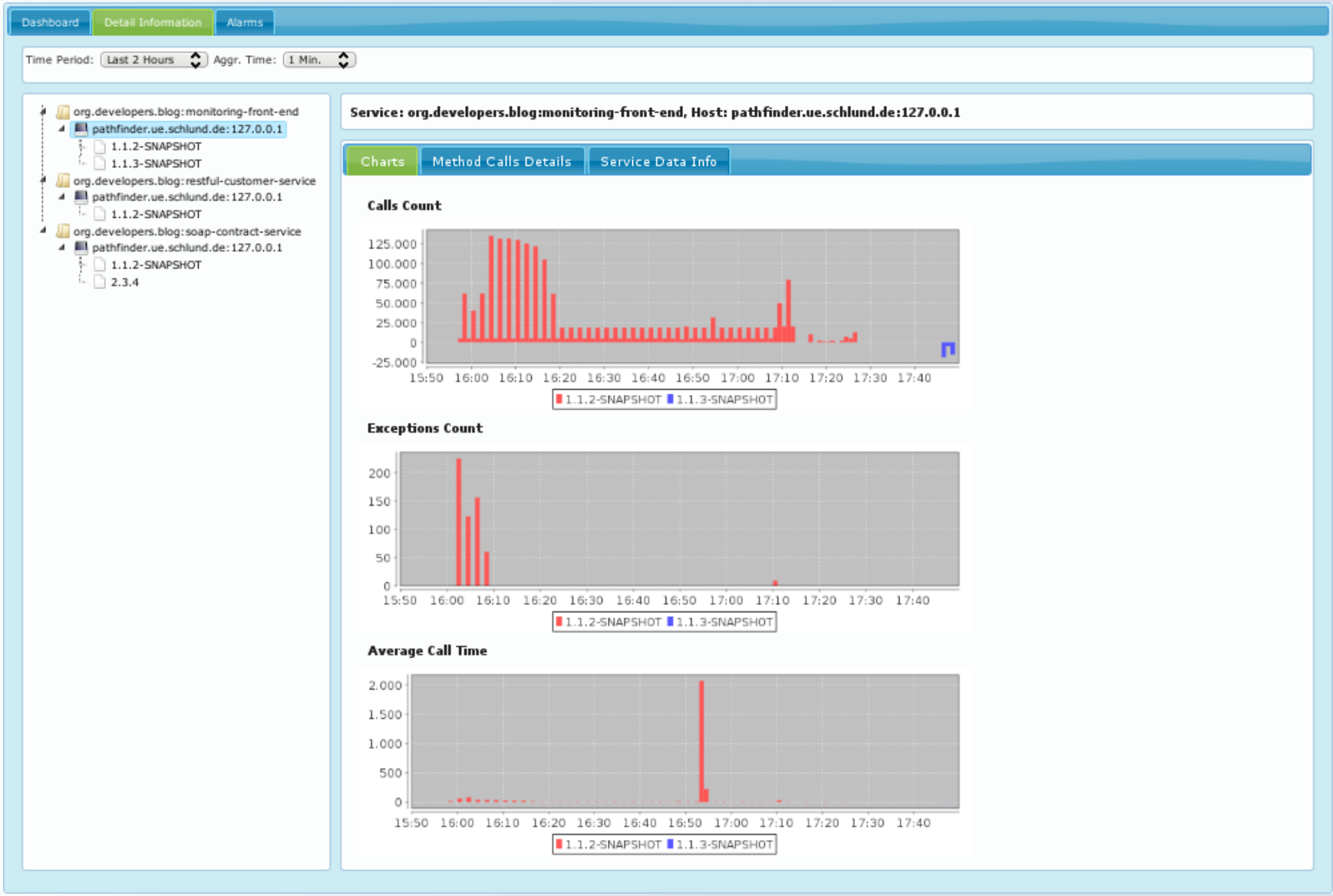
Distributed Software Monitoring - OpenSource Tool – Simple Monitoring
Data Model



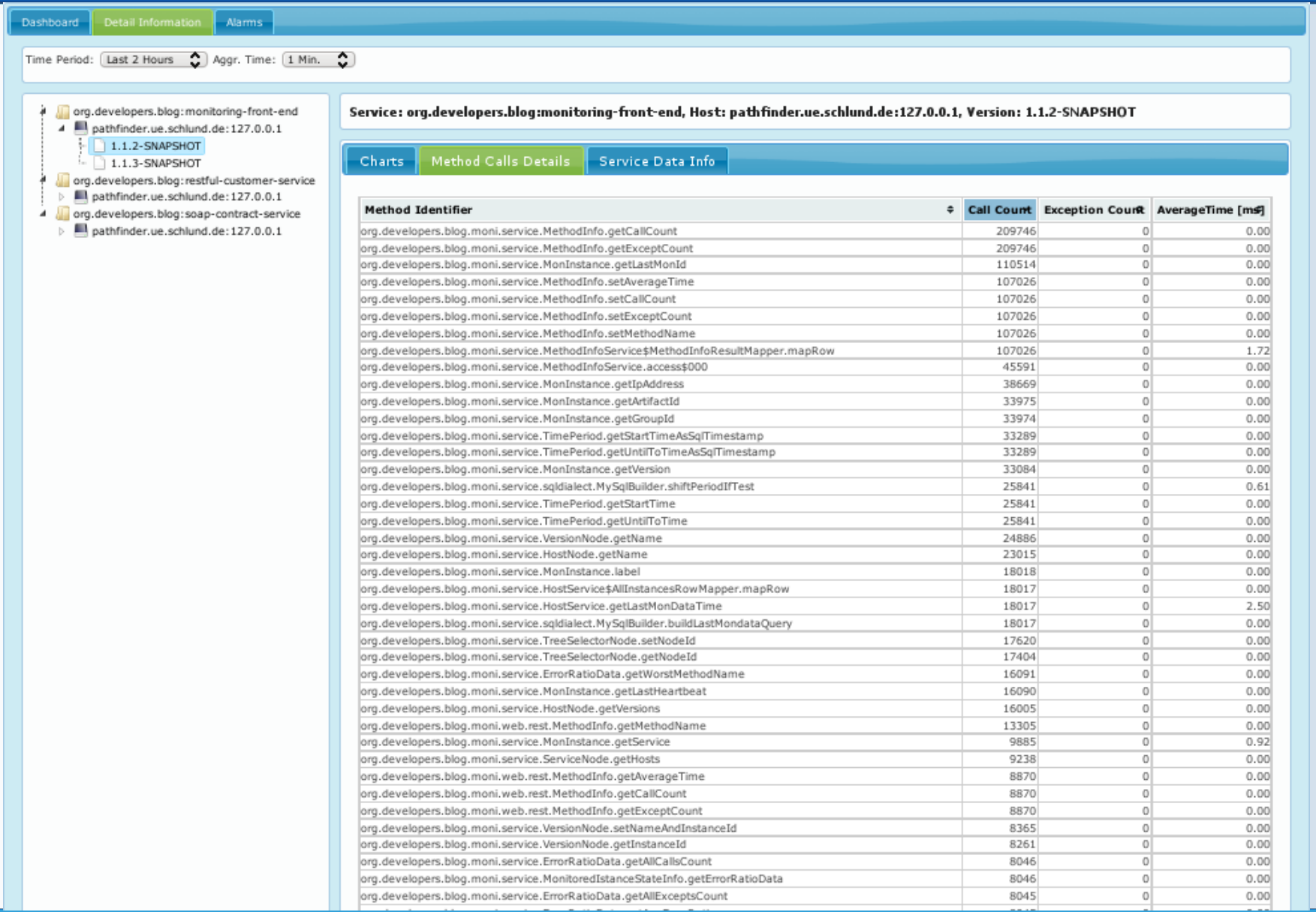
Distributed Software Monitoring - OpenSource Tool – Simple Monitoring Dashboard View



Distributed Software Monitoring - OpenSource Tool – Simple Monitoring
Chart View

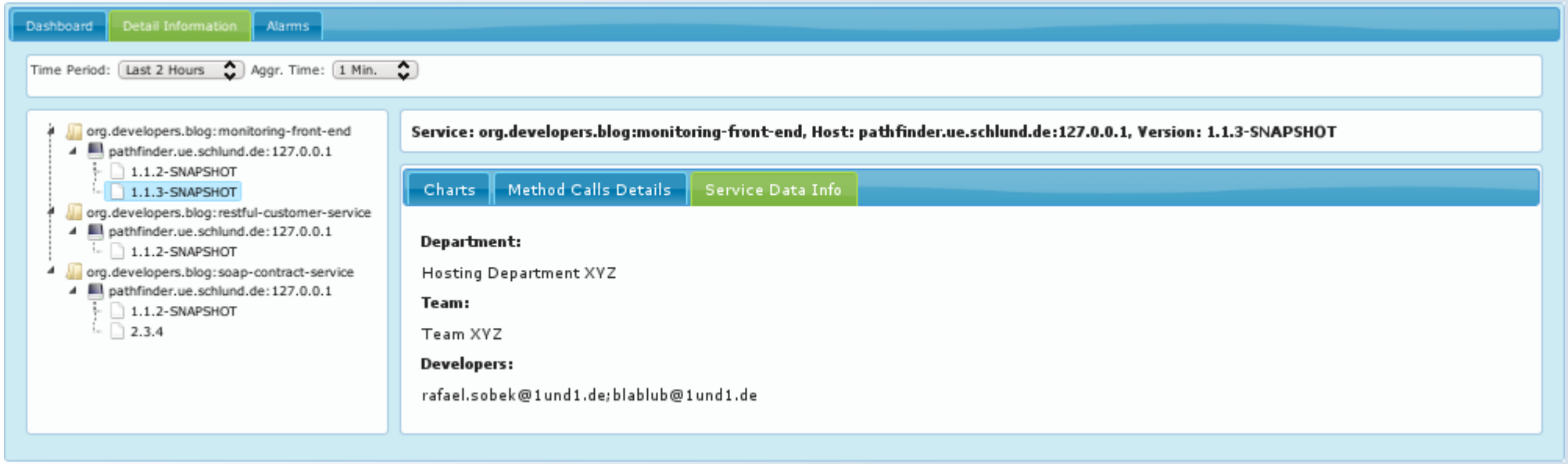


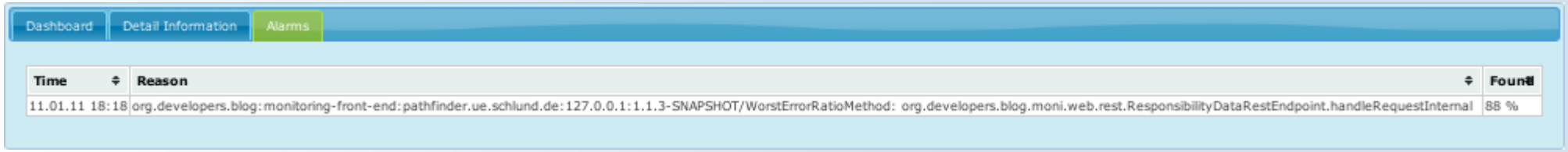
Distributed Software Monitoring - OpenSource Tool – Simple Monitoring
Method View (Call Count, Exception Count, Average Time)



Method Identifier	Call Count	Exception Count	AverageTime [ms]
org.developers.blog.moni.service.MethodInfo.getCallCount	209746	0	0.00
org.developers.blog.moni.service.MethodInfo.getExceptCount	209746	0	0.00
org.developers.blog.moni.service.MonInstance.getLastMonId	110514	0	0.00
org.developers.blog.moni.service.MethodInfo.setAverageTime	107026	0	0.00
org.developers.blog.moni.service.MethodInfo.setCallCount	107026	0	0.00
org.developers.blog.moni.service.MethodInfo.setExceptCount	107026	0	0.00
org.developers.blog.moni.service.MethodInfo.setMethodName	107026	0	0.00
org.developers.blog.moni.service.MethodInfoService$MethodInfoResultMapper.mapRow	107026	0	1.72
org.developers.blog.moni.service.MethodInfoService.access$000	45591	0	0.00
org.developers.blog.moni.service.MonInstance.getIpAddress	38669	0	0.00
org.developers.blog.moni.service.MonInstance.getArtifactId	33975	0	0.00
org.developers.blog.moni.service.MonInstance.getGroupId	33974	0	0.00
org.developers.blog.moni.service.TimePeriod.getStartTimeAsSqlTimestamp	33289	0	0.00
org.developers.blog.moni.service.TimePeriod.getUntilToTimeAsSqlTimestamp	33289	0	0.00
org.developers.blog.moni.service.MonInstance.getVersion	33084	0	0.00
org.developers.blog.moni.service.sqldialect.MySqlBuilder.shiftPeriodIfTest	25841	0	0.61
org.developers.blog.moni.service.TimePeriod.getStartTime	25841	0	0.00
org.developers.blog.moni.service.TimePeriod.getUntilToTime	25841	0	0.00
org.developers.blog.moni.service.VersionNode.getName	24886	0	0.00
org.developers.blog.moni.service.HostNode.getName	23015	0	0.00
org.developers.blog.moni.service.MonInstance.label	18018	0	0.00
org.developers.blog.moni.service.HostService$AllInstancesRowMapper.mapRow	18017	0	0.00
org.developers.blog.moni.service.HostService.getLastMonDataTime	18017	0	2.50
org.developers.blog.moni.service.sqldialect.MySqlBuilder.buildLastMondataQuery	18017	0	0.00
org.developers.blog.moni.service.TreeSelectorNode.setNodeId	17620	0	0.00
org.developers.blog.moni.service.TreeSelectorNode.getNodeId	17404	0	0.00
org.developers.blog.moni.service.ErrorRatioData.getWorstMethodName	16091	0	0.00
org.developers.blog.moni.service.MonInstance.getLastHeartbeat	16090	0	0.00
org.developers.blog.moni.service.HostNode.getVersions	16005	0	0.00
org.developers.blog.moni.web.rest.MethodInfo.getMethodName	13305	0	0.00
org.developers.blog.moni.service.MonInstance.getService	9885	0	0.92
org.developers.blog.moni.service.ServiceNode.getHosts	9238	0	0.00
org.developers.blog.moni.web.rest.MethodInfo.getAverageTime	8870	0	0.00
org.developers.blog.moni.web.rest.MethodInfo.getCallCount	8870	0	0.00
org.developers.blog.moni.web.rest.MethodInfo.getExceptCount	8870	0	0.00
org.developers.blog.moni.service.VersionNode.setNameAndInstanceId	8365	0	0.00
org.developers.blog.moni.service.VersionNode.getInstanceId	8261	0	0.00
org.developers.blog.moni.service.ErrorRatioData.getAllCallsCount	8046	0	0.00
org.developers.blog.moni.service.MonitoredIstanceStateInfo.getErrorRatioData	8046	0	0.00
org.developers.blog.moni.service.ErrorRatioData.getAllExceptsCount	8045	0	0.00

Distributed Software Monitoring - OpenSource Tool – Simple Monitoring
Metadata Information and Alarming View





Distributed Software Monitoring
Other Tools

Perf4J 0.9.13 http://perf4j.codehaus.org/

Usage:

```
StopWatch stopWatch = new LoggingStopWatch();
try {
    // the code block being timed - this is just a dummy example
    long sleepTime = (long)(Math.random() * 1000L);
    Thread.sleep(sleepTime);
    if (sleepTime > 500L) {
        throw new Exception("Throwing exception");
    }
    stopWatch.stop("codeBlock2.success", "Sleep time was < 500 ms");
} catch (Exception e) {
    stopWatch.stop("codeBlock2.failure", "Exception was: " + e);
}
```

Log4J Output:

```
INFO: start[1230493236109] time[447] tag[codeBlock2.success] message[Sleep time was < 500 ms]
INFO: start[1230493236719] time[567] tag[codeBlock2.failure] message[Exception was: java.lang.Exception: Throwing exception]
INFO: start[1230493237286] time[986] tag[codeBlock2.failure] message[Exception was: java.lang.Exception: Throwing exception]
INFO: start[1230493238273] time[194] tag[codeBlock2.success] message[Sleep time was < 500 ms]
INFO: start[1230493238467] time[463] tag[codeBlock2.success] message[Sleep time was < 500 ms]
INFO: start[1230493238930] time[310] tag[codeBlock2.success] message[Sleep time was < 500 ms]
```

Distributed Software Monitoring
Other Tools

http://jamonapi.sourceforge.net/

- SQL/JDBC Monitoring – No Code Changes Required!
- Servlet Filter – No Code Changes Required!
- Interface/Exception Monitoring – One line of Code per Interface
- ...

```
MyInterface myObj = new MyImplementation();
myObj = (MyInterface) MonProxyFactory.monitor(myObj);
myObj.myMethod(); // monitored!
```

```
Monitor mon=MonitorFactory.start("myPage.jsp");
…page code…
mon.stop();
```

Questions?