UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik

Introducing Scalileo
A Java Based Scaling Framework

Tilmann Rabl, Christian Dellwo, Harald Kosch
Chair of Distributed Database Systems

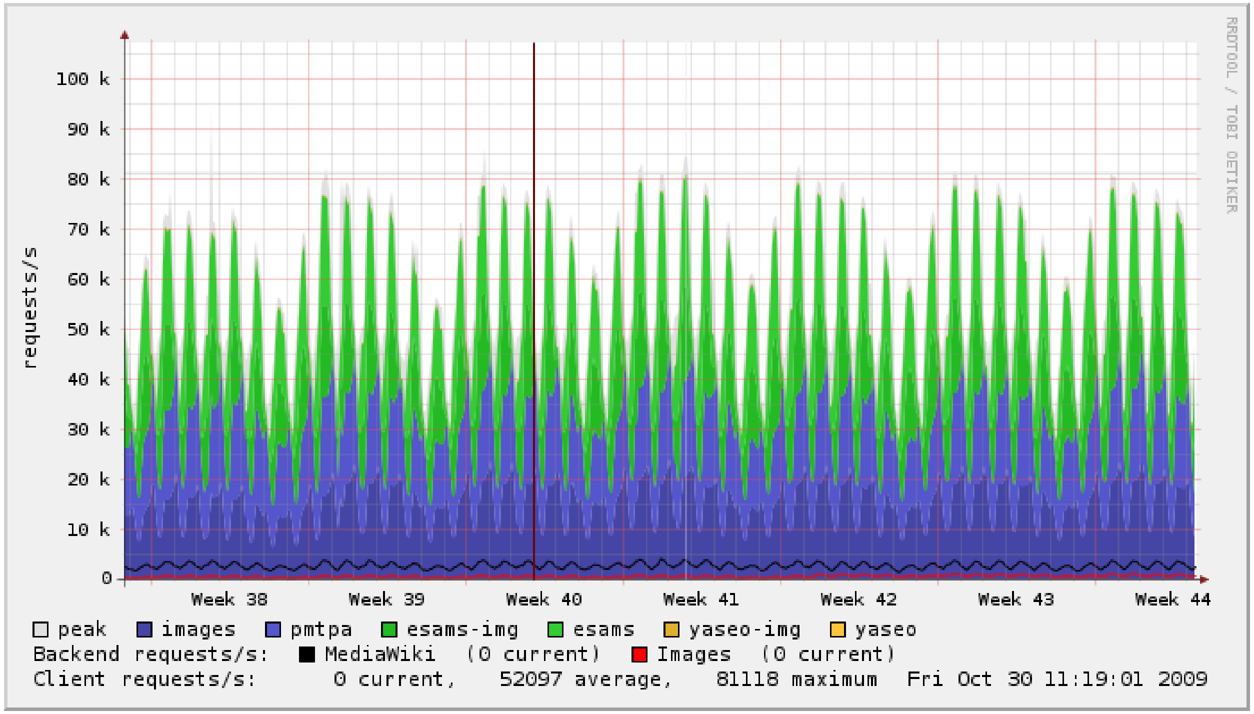
First International Conference on Energy-Efficient Computing and Networking
April 15, 2010

Motivation

- Application workloads grow
  - Web application
  - Database
  - Beyond single processor / node
- Hardware: horizontal scaling
  - More hardware (usually shared nothing)
  - Scales good
  - Cheap
- Software Scaling
  - Distributed applications
  - Usually manual / semi-automatic scaling
  - **Expensive**
- Common practice: vastly underloaded system
  - **Energy-inefficient**

Real-World Workloads I

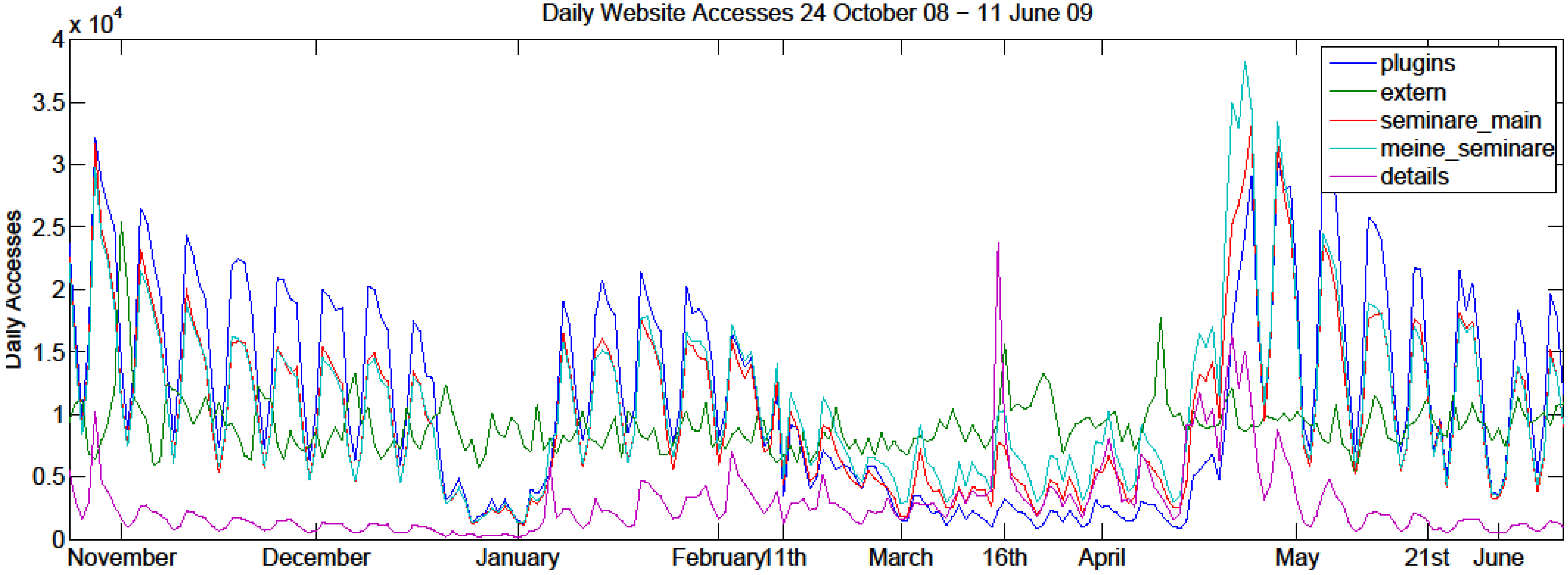
- Homogeneous workload
  - Daily and weekly patterns



- Need for ***automatic*** scaling

Real-World Workloads II

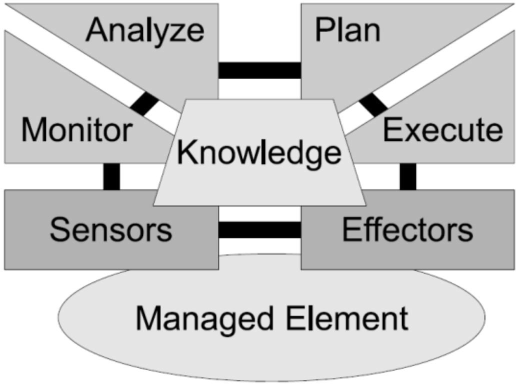
- Special purpose workloads
  - Daily and weekly patterns
  - Workload classes / trends
  - Outliers



- Need for ***autonomic*** scaling

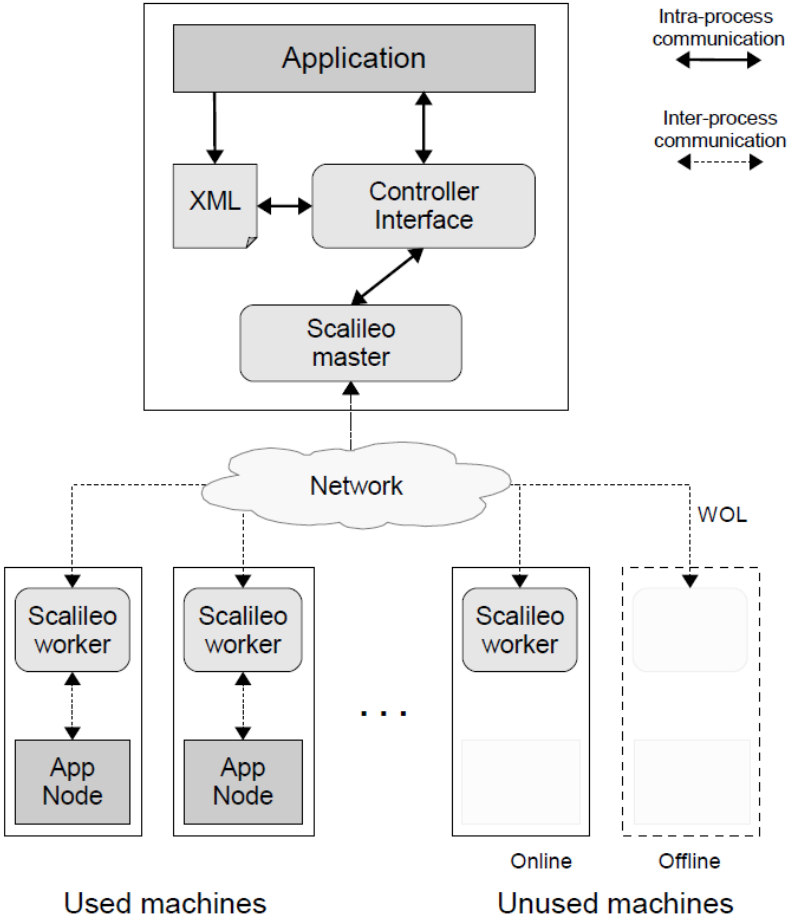
Scalileo

- Scaling framework
  - Easy integration of scaling
  - Java based
- Self-scaling
  - Autonomic computing
  - Online feedback control loop
  - MAPE
- **Energy-efficiency**
  - On-off policy
  - Wake-on-LAN



Architecture

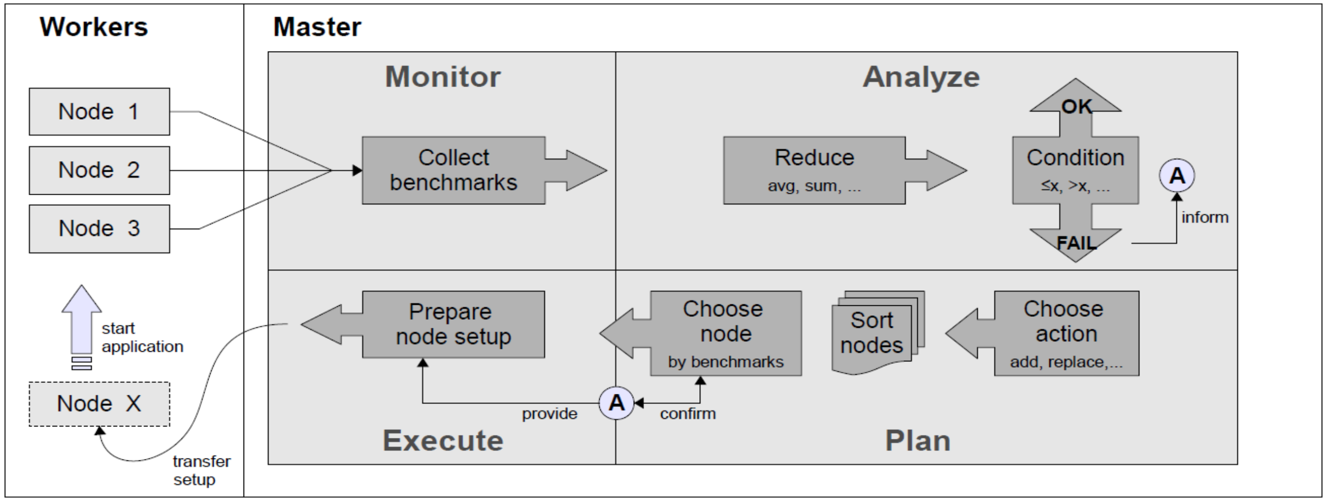
- Application
  - Hierarchical distributed system
  - Supports scaling
- Controller
  - Interface between application and Scalileo
- Master
  - Monitor the system
  - Coordinate the nodes
- Worker
  - Start and stop application
  - Run benchmarks



Components

- Login Method
  - Remote access to nodes
  - Login, process start, file transfer
- Benchmark
  - Measuring node performance and status
- Reduction
  - Reducing measurements to a single value
- Condition
  - Valid domain for benchmark results
- Defined by XML file
- Dynamically loaded with Java Reflection API

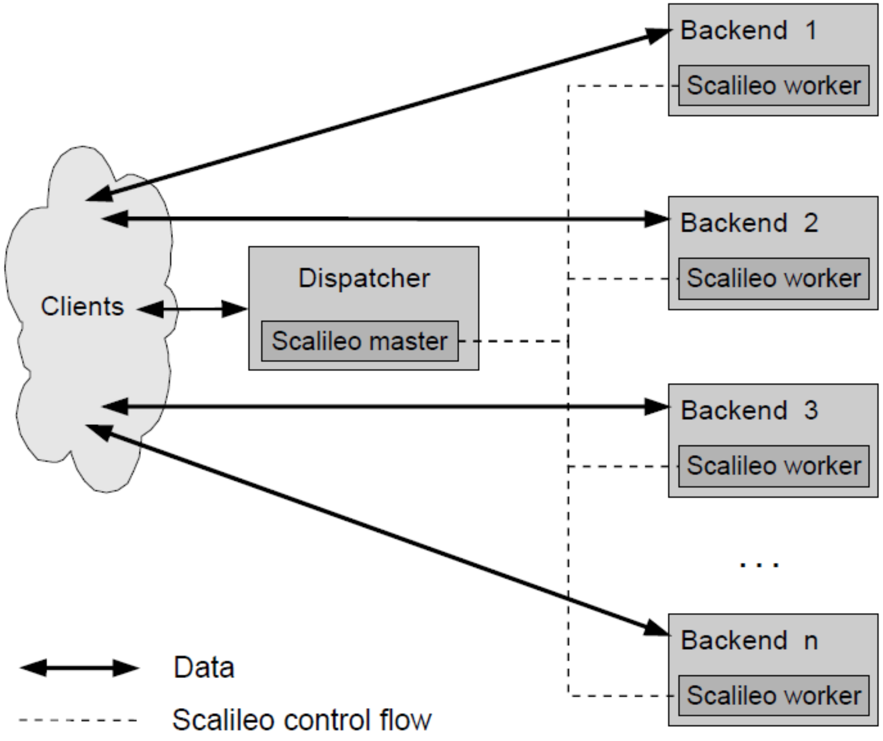
Functional Principle



- MAPE model – online feedback control loop
  - Monitor: benchmark nodes
  - Analyze: reduce measurements, check conditions
  - Plan: choose action, choose nodes
  - Execute: prepare node setup, start/stop node

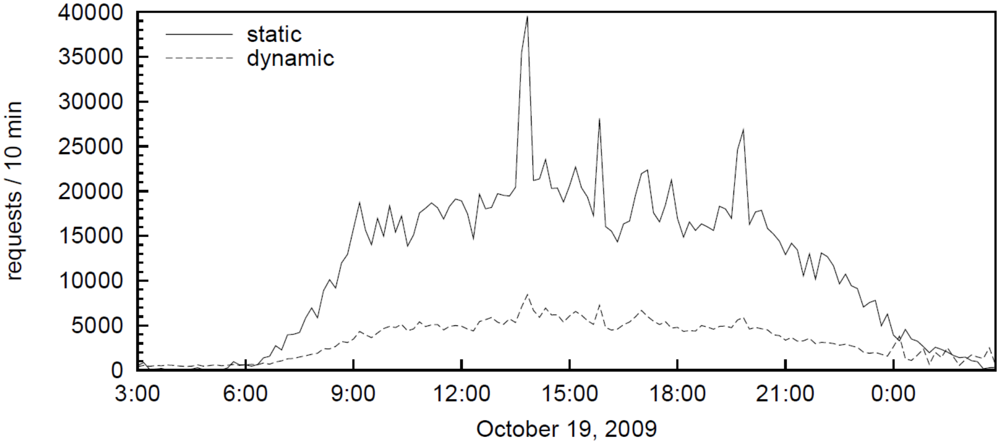
Evaluation

- Distributed web server
  - Simulated
  - Dynamic and static requests
- Central dispatcher
  - HTTP 302 redirect (Found)
- 1 – 4 workers
  - Workstation
    - 3 GHz Pentium D
    - 3 GB RAM
  - Energy consumption
    - 91 W idle
    - 200 W booting
    - 2 W off

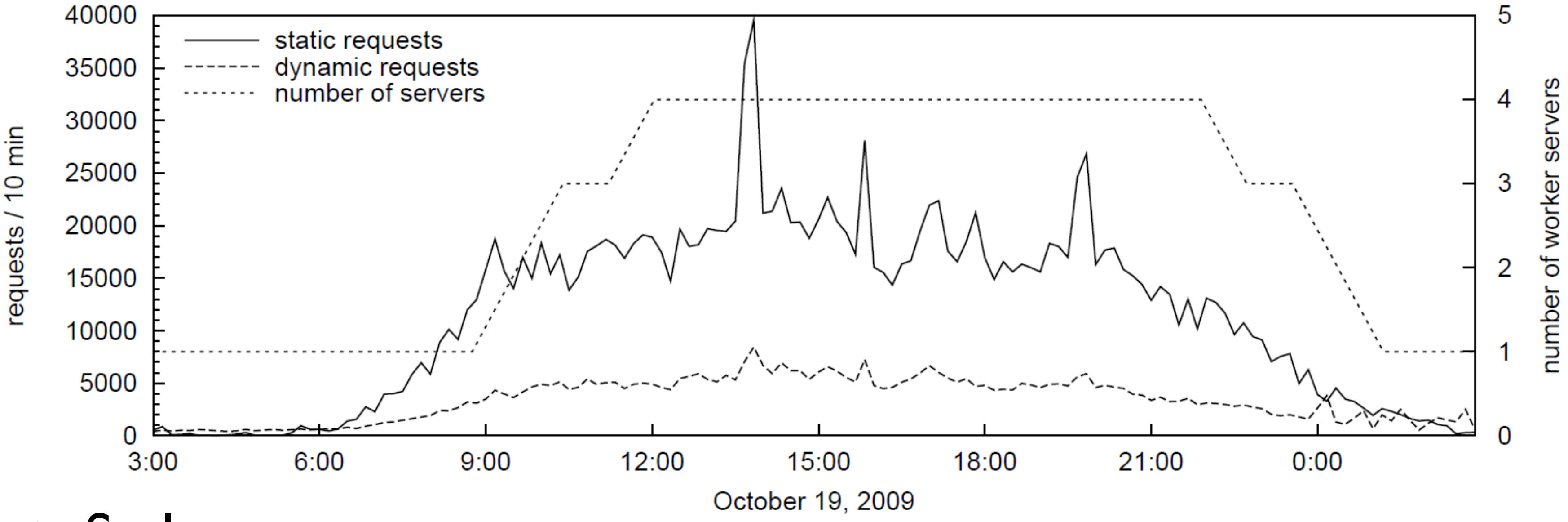


Workload

- Stud.IP: online eLearning management system
  - University of Passau
  - 15000 Users
- Apache Log
  - First day of lecture period
  - Static and dynamic HTTP requests
- Replayed at 48x speed
  - Only every 20th request
  - Static vs. dynamic ~ 1:3

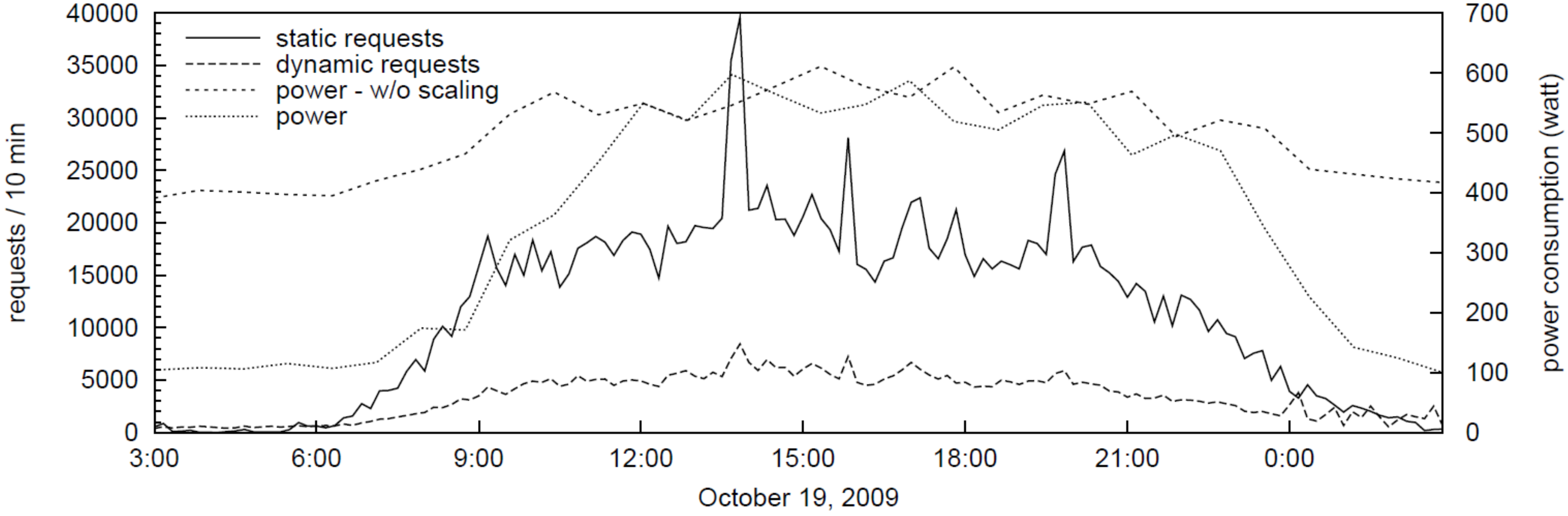


Evaluation: Scaling



- Scale up
  - Sliding window of CPU usage
  - 25 sec (~16 min) over 45% CPU (66% of samples)
- Conservative scale down
  - 35 sec (~28 min) lower than 20% CPU
- Relative booting time: 50 minutes
  - With maximum energy consumption

Evaluation: Energy Consumption



- Total 175 Wh vs. 250 Wh
  - 175 Wh w scaling
  - 250 Wh w/o scaling
  - 30 % savings

Conclusion

- Scalileo
  - Autonomic scaling framework
  - Adaptable – XML configuration
  - Extensible – Java interfaces
  - Energy savings (w/o extensive optimization): **30%**
- Future Work
  - Better benchmarks: time series analysis
  - Local optimizations: dynamic voltage scaling
  - Scaling databases

Thank you

- Questions?

Extensibility / Adaptability – Example

- Java Interface Definition
  - All components:

```
public <constructor>(String id, Map<String,Object> parameters) {}
public abstract boolean hasValidParameters();
```

  - Benchmark:

```
public abstract double run() throws BenchmarkException;
```

- XML config file

```
<benchmarks>
  <benchmark id="ExampleBenchmark"
      class="example.package.ExampleBenchmark" />
  <benchmark id="OtherBenchmark"
      class="example.package.OtherBenchmark" />
</benchmarks>
```

```
<node ...>
  ...
  <use-benchmark type="ExampleBenchmark">
    <parameter key="interval" type="java.lang.Integer"
       value="10000"/>
  </use-benchmark>
  <use-benchmark type="OtherBenchmark">
  ...
</node>
```

Controller Interface

- constraintViolated
  - First information for application
- beforeScale
  - Scaling necessary
- chooseNode
  - Choose best node or no scaling
- getNodeSetup
  - Command and files for the node setup
- getNodeShutdown
  - Command and files for the node shutdown
- afterScale
  - Result of scaling
- Other methods
  - Special events
  - Default implementation