The ABS Language

Simulating the Cost of Cloud Deployment

Rudolf Schlatte, University of Oslo
rudi@ifi.uio.no

April 19, 2018

What do we want?

- Model and simulate concurrent and distributed systems
- Make qualitative and quantitative predictions on QoS, resource usage, deployment cost, ...
  - Before starting the implementation

What do we want?

- Model and simulate concurrent and distributed systems
- Make qualitative and quantitative predictions on QoS, resource usage, deployment cost, ...
  - Before starting the implementation

What do we provide?

- Integrated behavioral and deployment model
- Simulation and static analysis tools
- A Java-like modeling language with Actor-based semantics
  - Easy to learn, easy to use

What do we want?

- Model and simulate concurrent and distributed systems
- Make qualitative and quantitative predictions on QoS, resource usage, deployment cost, ...
  - Before starting the implementation

What do we provide?

- Integrated behavioral and deployment model
- Simulation and static analysis tools
- A Java-like modeling language with Actor-based semantics
  - Easy to learn, easy to use
- Developed by EU research projects HATS, Envisage
- Maintained by UiO with external contributors
- `http://docs.abs-models.org`
  `https://github.com/abstools/abstools`

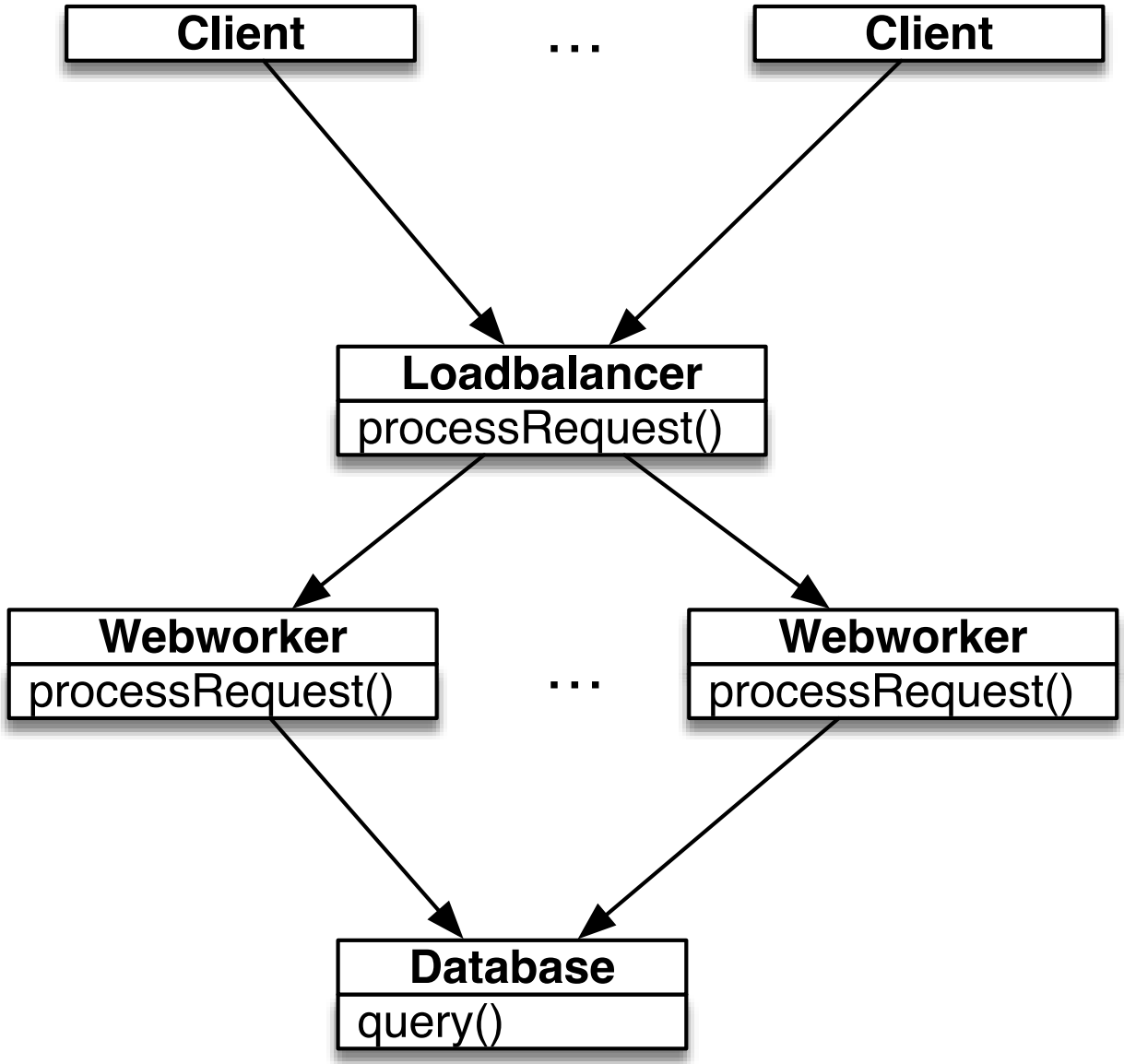
Behavioral Modeling with ABS

- A language to model OO and distributed software systems
- Java-like syntax: interfaces, classes
- Object-local processes communicate via asynchronous method calls and futures
- Pure functional datatype layer
  `data Person = Person(String name, Int age);`

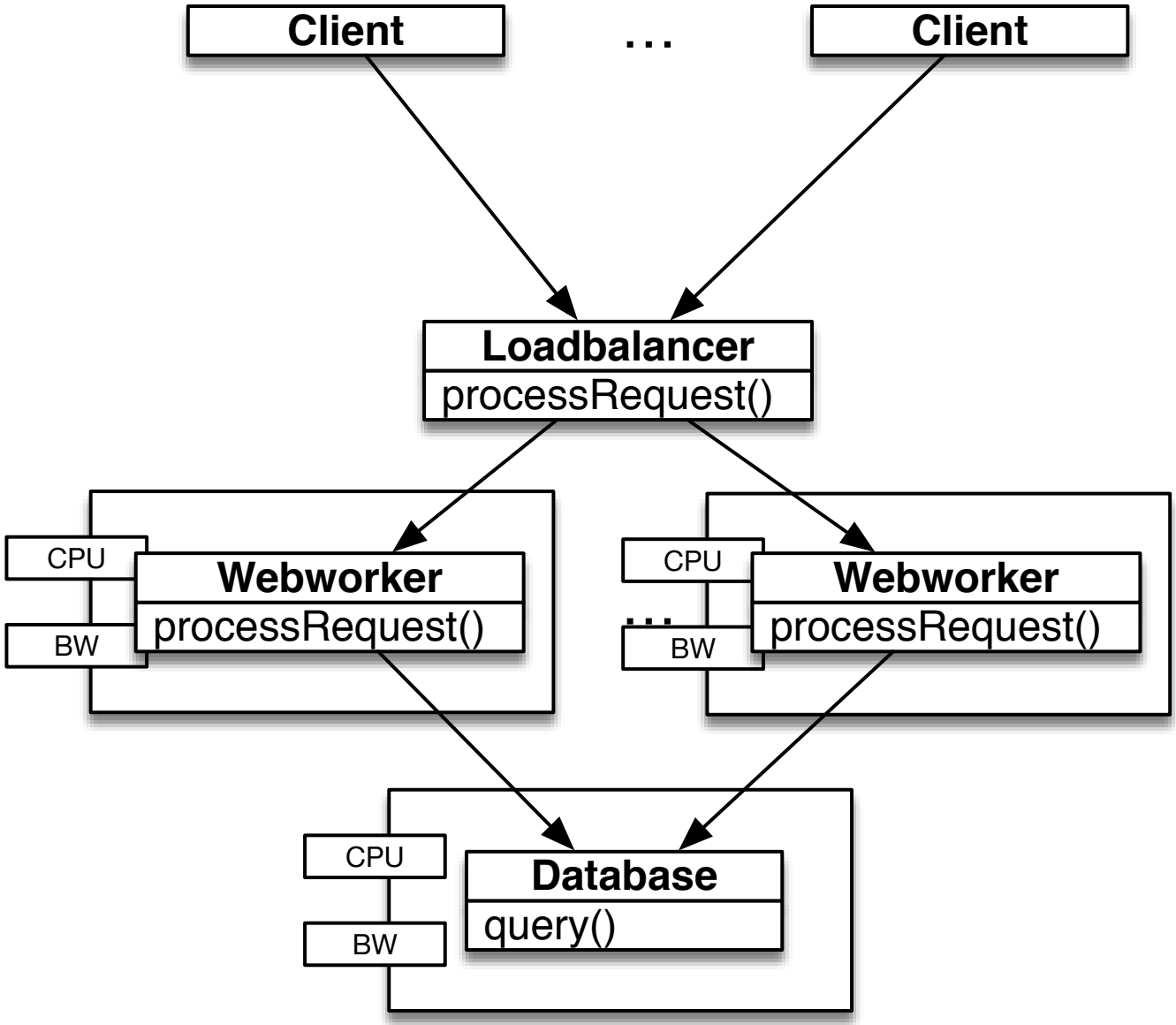
Catchphrases

- Safe object-based multithreading via cooperative scheduling
- Shared-nothing message-based parallelism between object groups

Example: Behavioral Model of a Web App Architecture



Example: Adding Resource and Deployment Model



Resource and Deployment Modeling

- Objects get a location $\Rightarrow$ Deployment Component
  `[DC: server1] new Worker();`
- Deployment components carry resources (Speed, Bandwidth, Memory)
- Code gets annotated with Resource usage information
  `[Cost: 15] x = x + 1;`
- No functional changes, minimal code changes

Resource and Deployment Modeling

- Objects get a location ⇒ Deployment Component
  `[DC: server1] new Worker();`
- Deployment components carry resources (Speed, Bandwidth, Memory)
- Code gets annotated with Resource usage information
  `[Cost: 15] x = x + 1;`
- No functional changes, minimal code changes

Changes required

- 2 lines modified: add cost annotations
- 3 lines modified: add location annotations
- 5 lines added: set up deployment in main block

Advanced Resource Management

- Dynamic load balancing
- Dynamic deployment scenarios (horizontal scaling)
- Adapt, transfer resources (vertical scaling)
- Dynamic cost model

The Model API

- Access running model from outside
- Based on HTTP requests, JSON data
- Inspect object state, call methods (and get the result)
- Used in various case studies
  - driving models from external sources, e.g., replaying real system logs on model
  - custom, model-specific visualizations

More Information

- Language reference: `http://docs.abs-models.org`
- Open source at: `http://github.com/abstools/abstools`
- Contact: rudi@ifi.uio.no