Providing Fault-tolerant Execution of Web-service–based Workflows within Clouds

Johannes Behl, Tobias Distler, Rüdiger Kapitza, Florian Heisig, Matthias Schunter

CloudCP 2012, Bern, April 10th 2012

From Clouds to TClouds

Cloud Computing is … “Everything as a Service” (XaaS)

- Computing power, storage, software, …

Deficiencies regarding trustworthiness

- Inhibit critical applications like financial or medical services

Motivation for TClouds

- Targets provisioning of dependable and secure cloud infrastructures
- EU-funded project comprising 14 partners

Web-Service Orchestration

Combination and coordination of Web services

- Integral part of cloud computing

We need

- Way to express (critical) workflows based on Web services
- Fault-tolerant platform to execute these workflows

Web-Service Orchestration

Combination and coordination of Web services

- Integral part of cloud computing

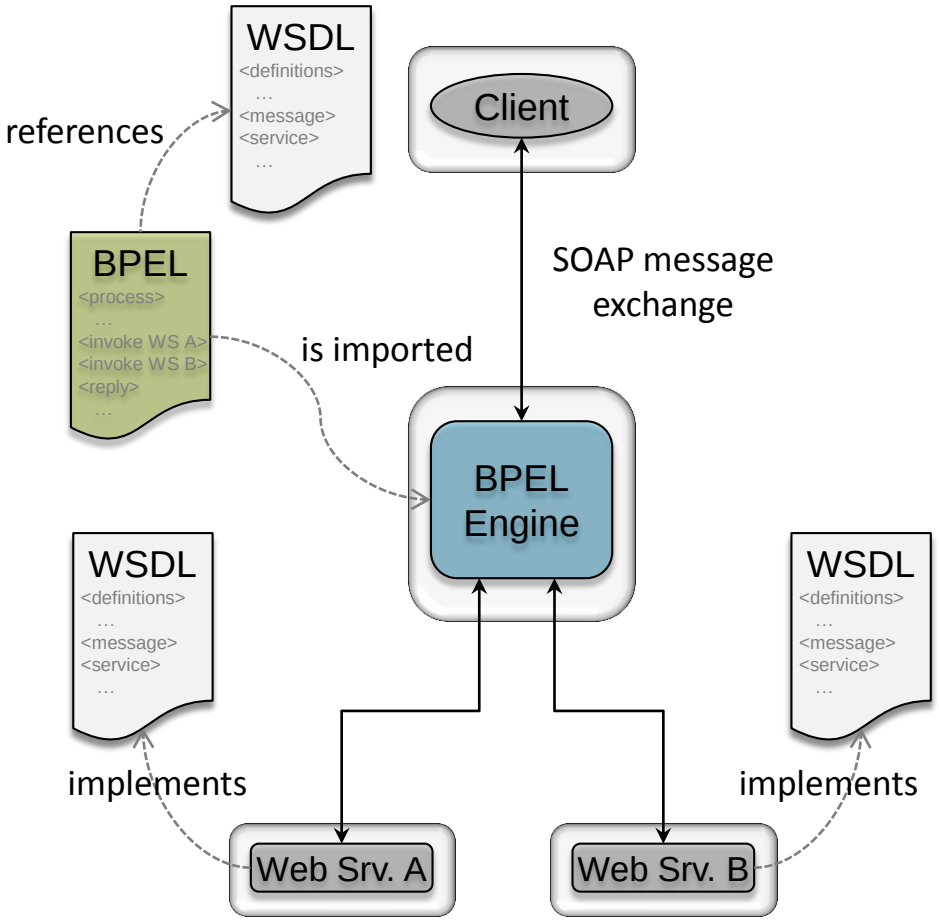
We need

- Way to express (critical) workflows based on Web services
- Fault-tolerant platform to execute these workflows

Web Service Business Process Execution Language (WS-BPEL)

- Standardized XML-based language …
- … to describe and execute Web-service–based workflows
- Tools to create, execute, and manage workflows

Standard BPEL Infrastructure



BPEL process definition

- Specifies workflow

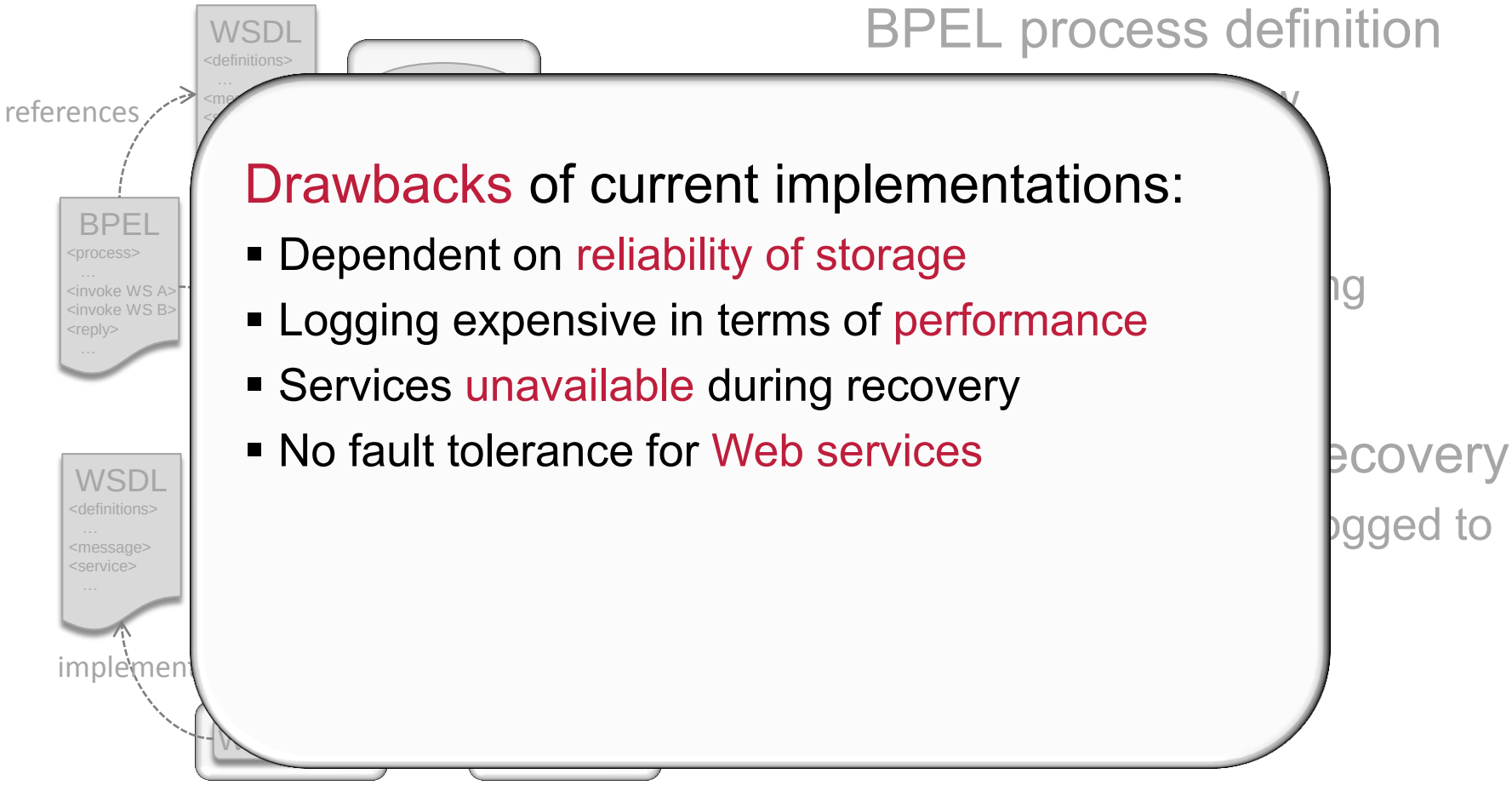
BPEL engine

- Platform for executing BPEL processes

Logging for crash recovery

- State changes are logged to stable storage

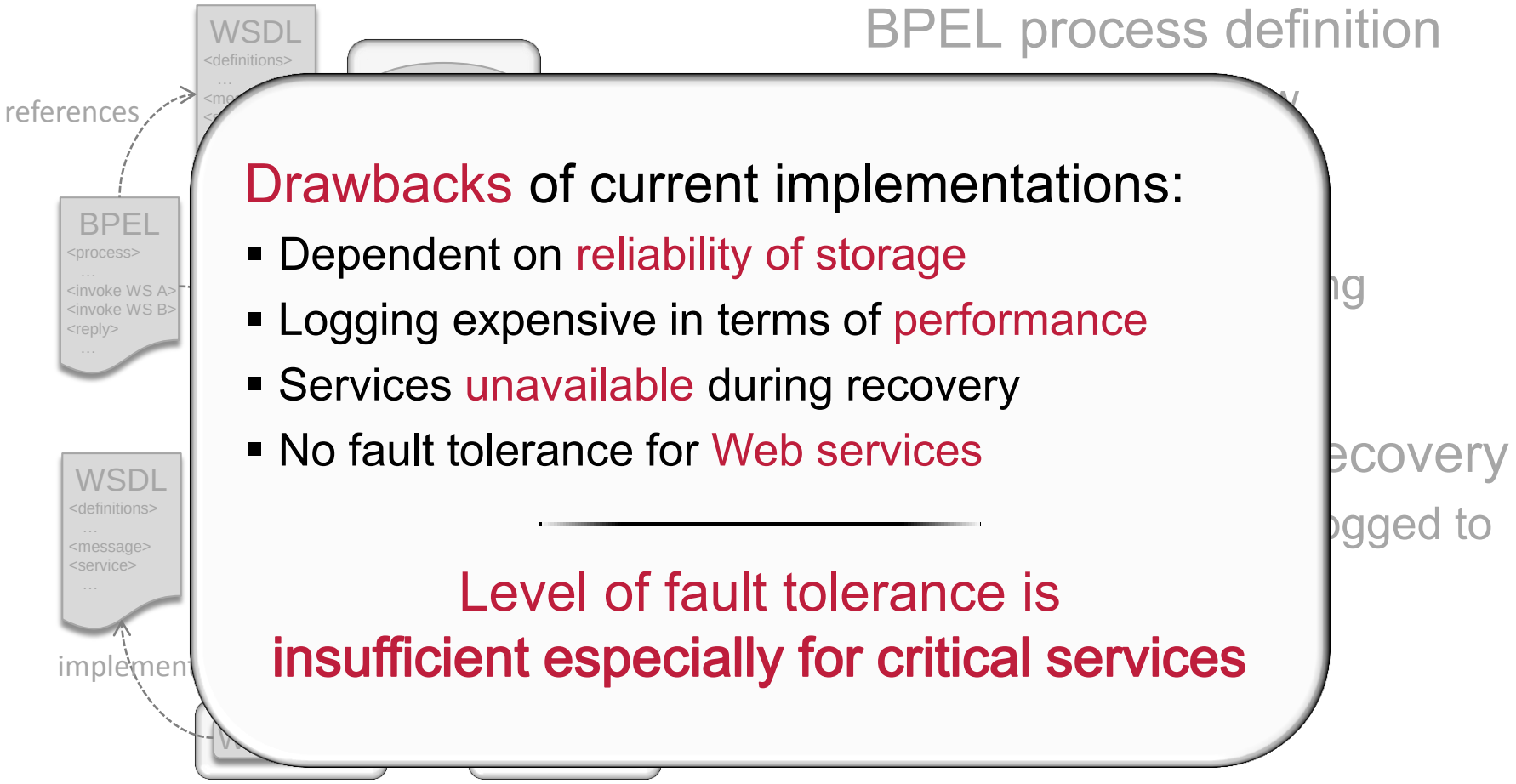
Standard BPEL Infrastructure



Drawbacks of current implementations:

- Dependent on reliability of storage
- Logging expensive in terms of performance
- Services unavailable during recovery
- No fault tolerance for Web services

Standard BPEL Infrastructure



Drawbacks of current implementations:

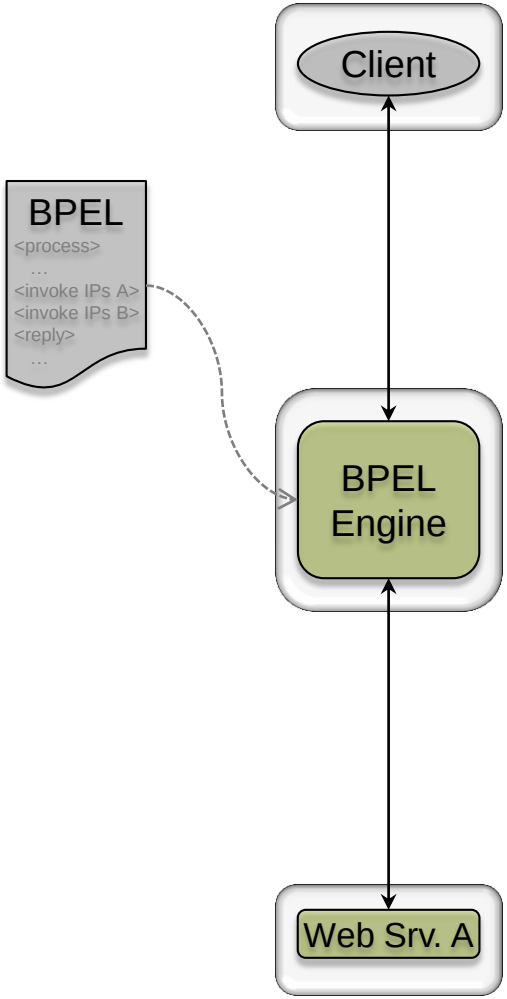
- Dependent on **reliability of storage**
- Logging expensive in terms of **performance**
- Services **unavailable** during recovery
- No fault tolerance for **Web services**

Level of fault tolerance is
insufficient especially for critical services

Reliable BPEL Infrastructure

- Motivation
- Reliable BPEL Infrastructure
- Evaluation
- Outlook and Conclusion

Reliable BPEL Infrastructure



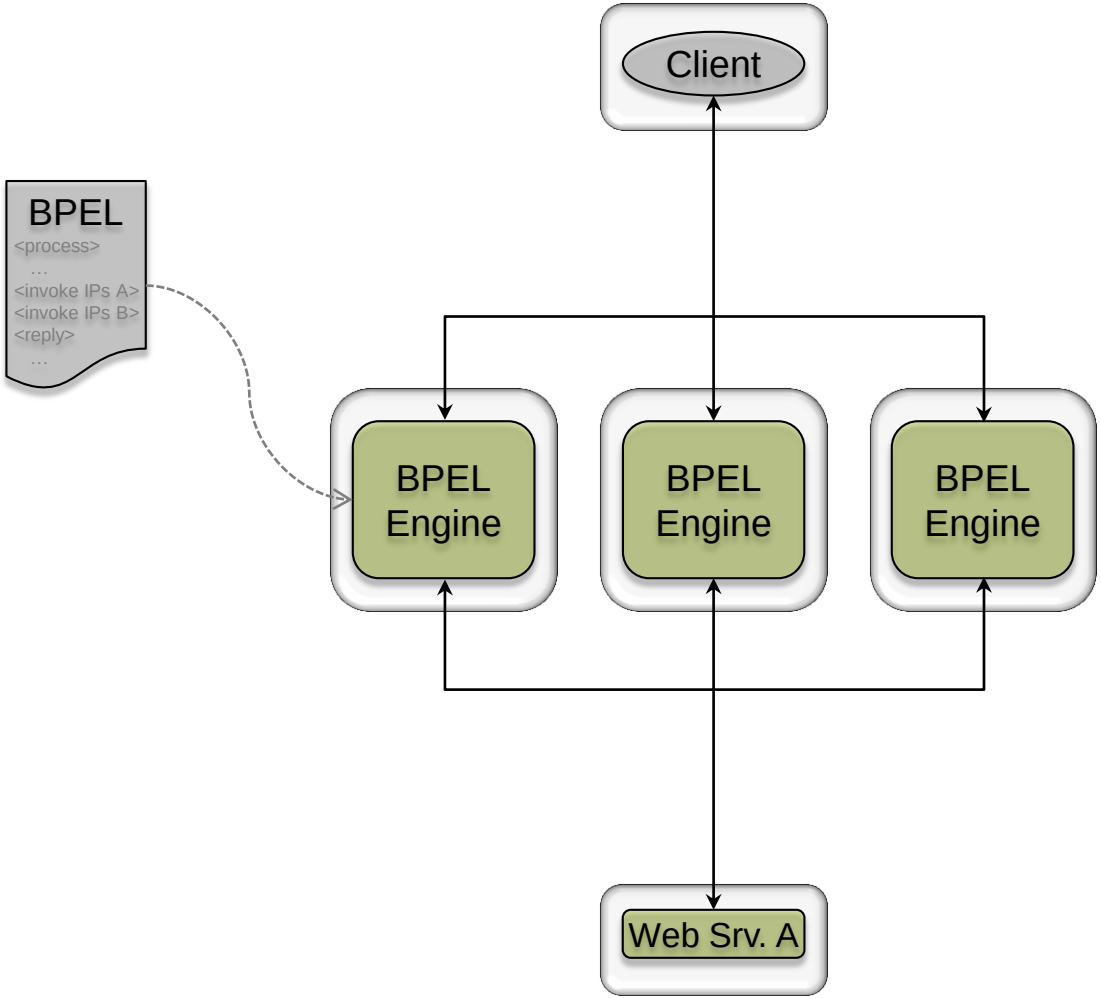
Starting Point:

Reuse current BPEL systems

- BPEL engines
- Tools

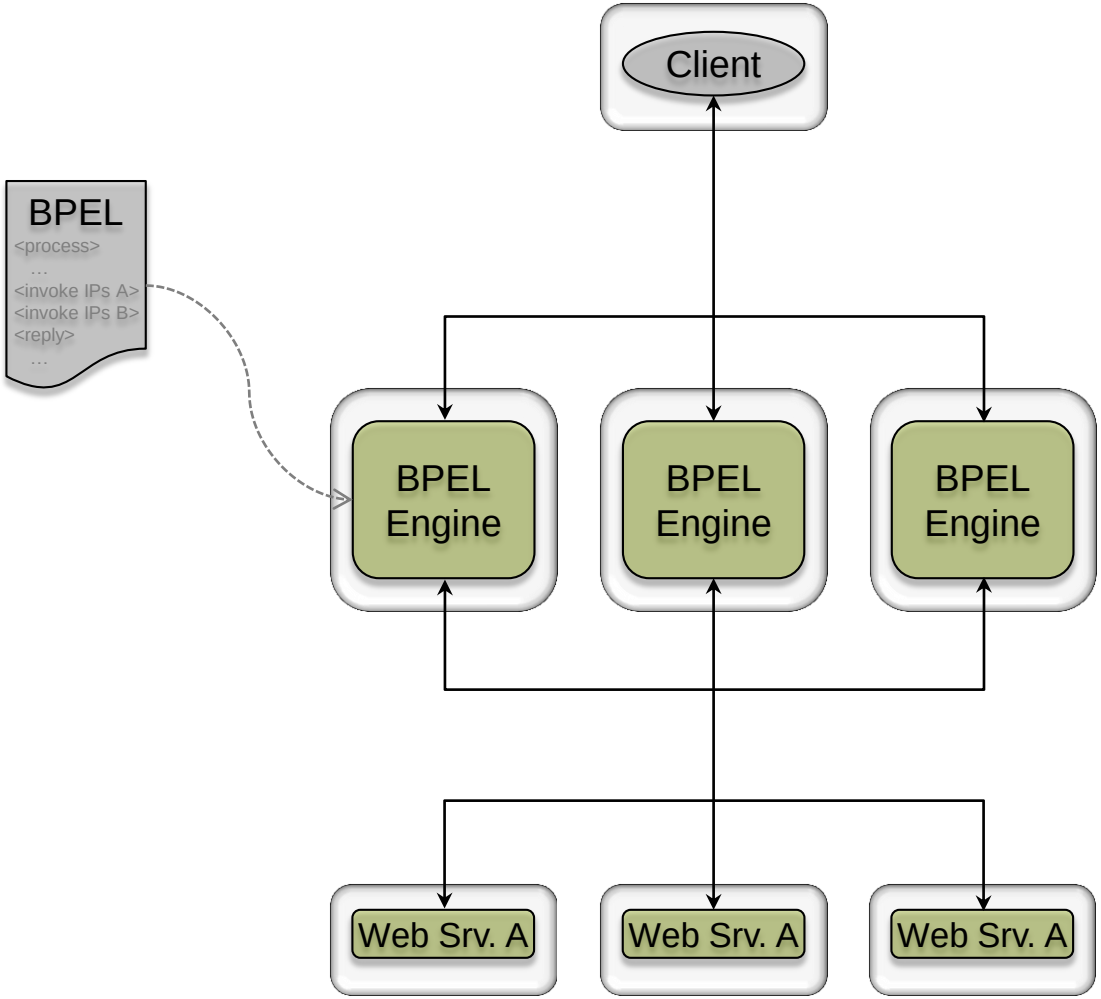
Replace standard mechanism for crash recovery

Reliable BPEL Infrastructure



Active replication for high availability

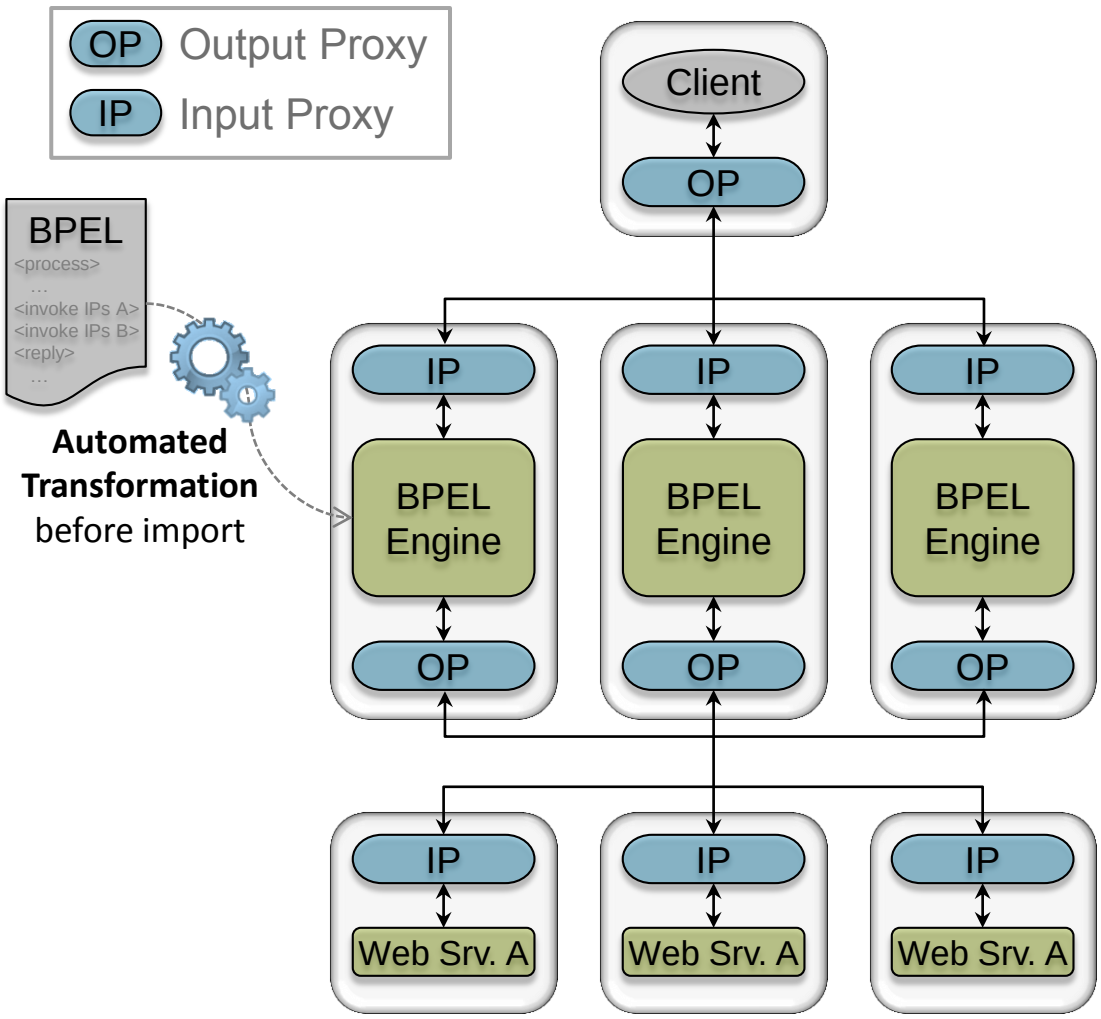
Reliable BPEL Infrastructure



Active replication for high availability

- BPEL engines
- Web services

Reliable BPEL Infrastructure

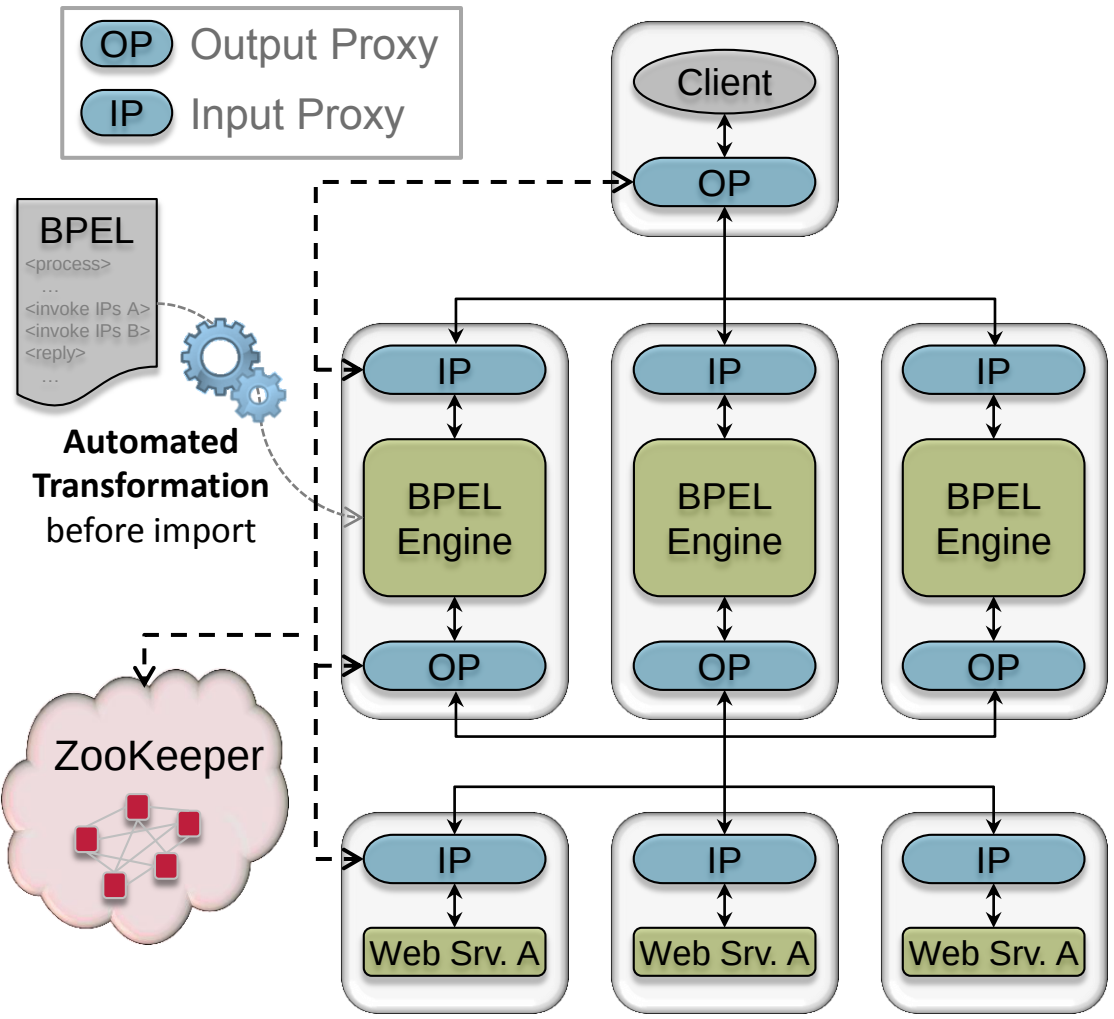


Active replication for high availability

- BPEL engines
- Web services

Generic proxies and automated transformation for transparency

Reliable BPEL Infrastructure



Active replication for high availability

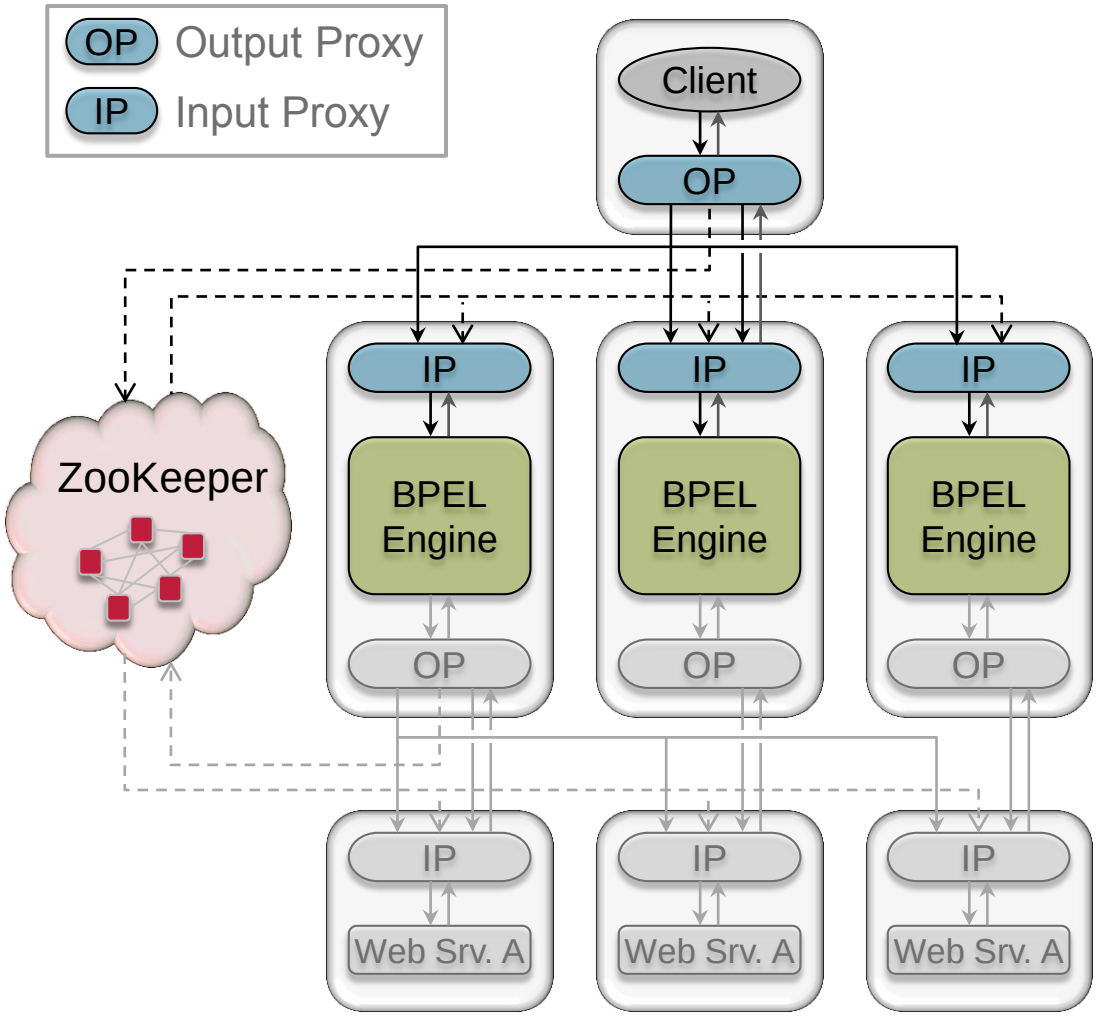
- BPEL engines
- Web services

Generic proxies and automated transformation for transparency

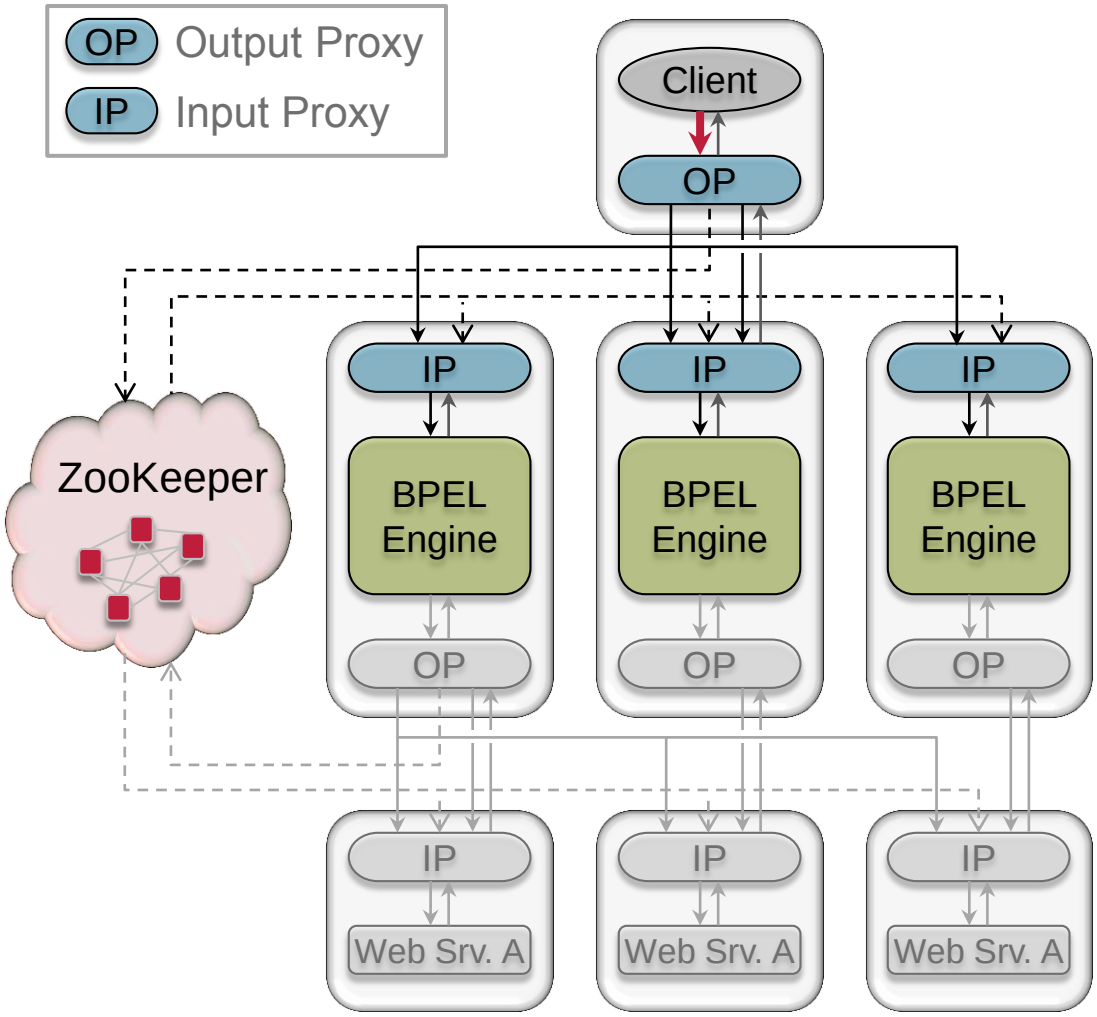
Coordination service for simplified configuration and implementation

- Apache ZooKeeper for:
- Leader election
- Crash detection
- Dynamic configuration
- Request ordering

Details – Client / BPEL Stage

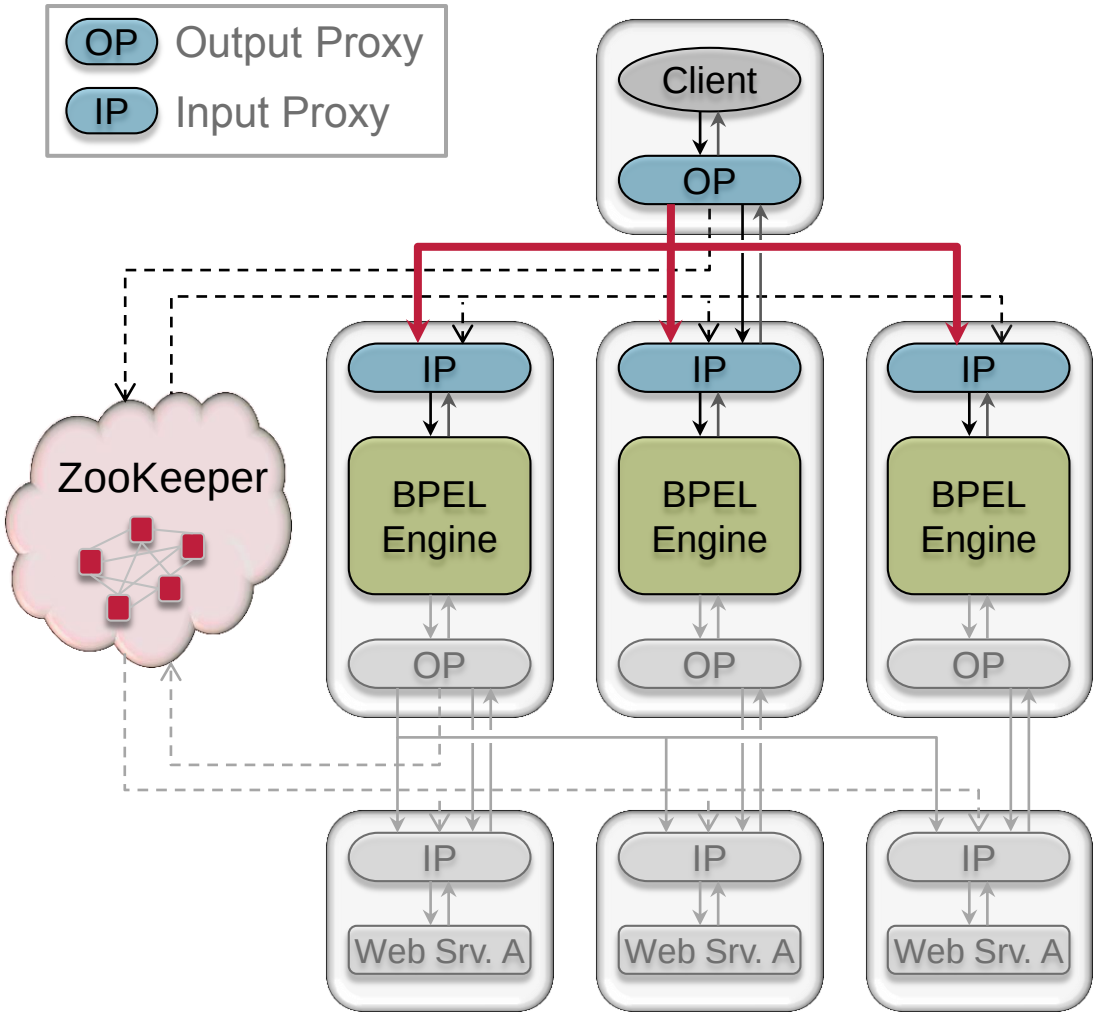


Details – Client / BPEL Stage



(1) Web-service request is passed to local output proxy

Details – Client / BPEL Stage

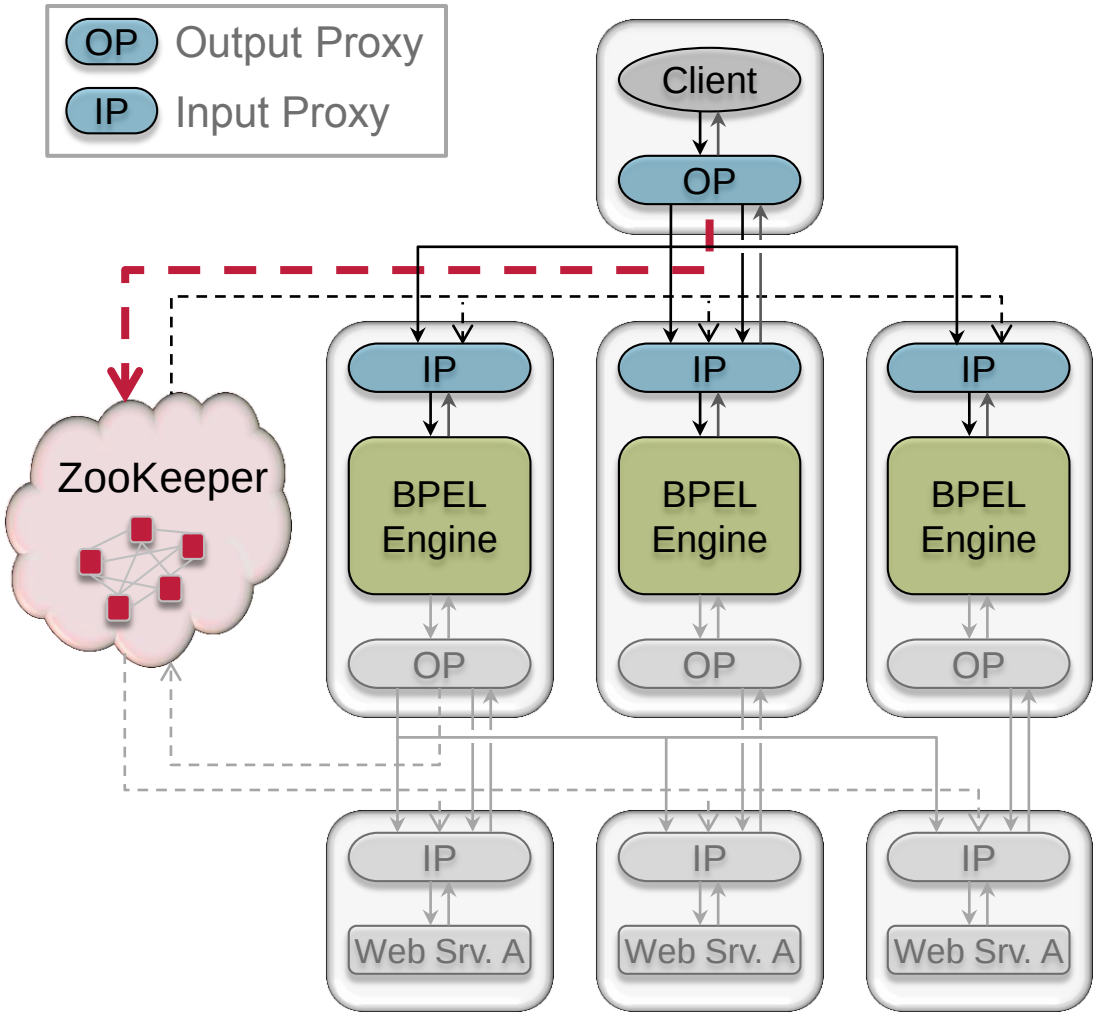


(1) Web-service request is passed to local output proxy

(2) Request data and request ID are sent to input proxies

- ZooKeeper for list of active replicas

Details – Client / BPEL Stage



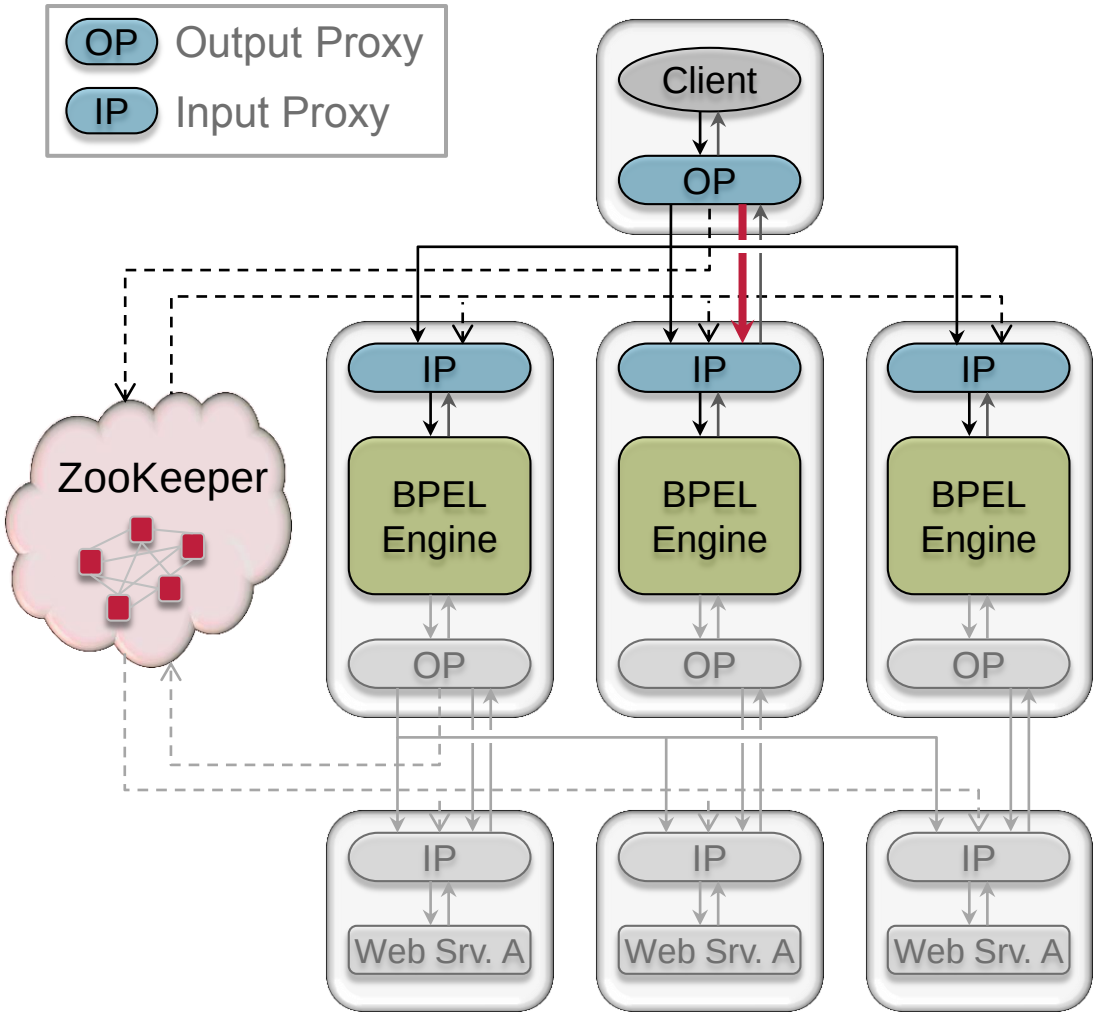
(1) Web-service request is passed to local output proxy

(2) Request data and request ID are sent to input proxies

- ZooKeeper for list of active replicas

(3) Request is registered at ZooKeeper

Details – Client / BPEL Stage



(1) Web-service request is passed to local output proxy

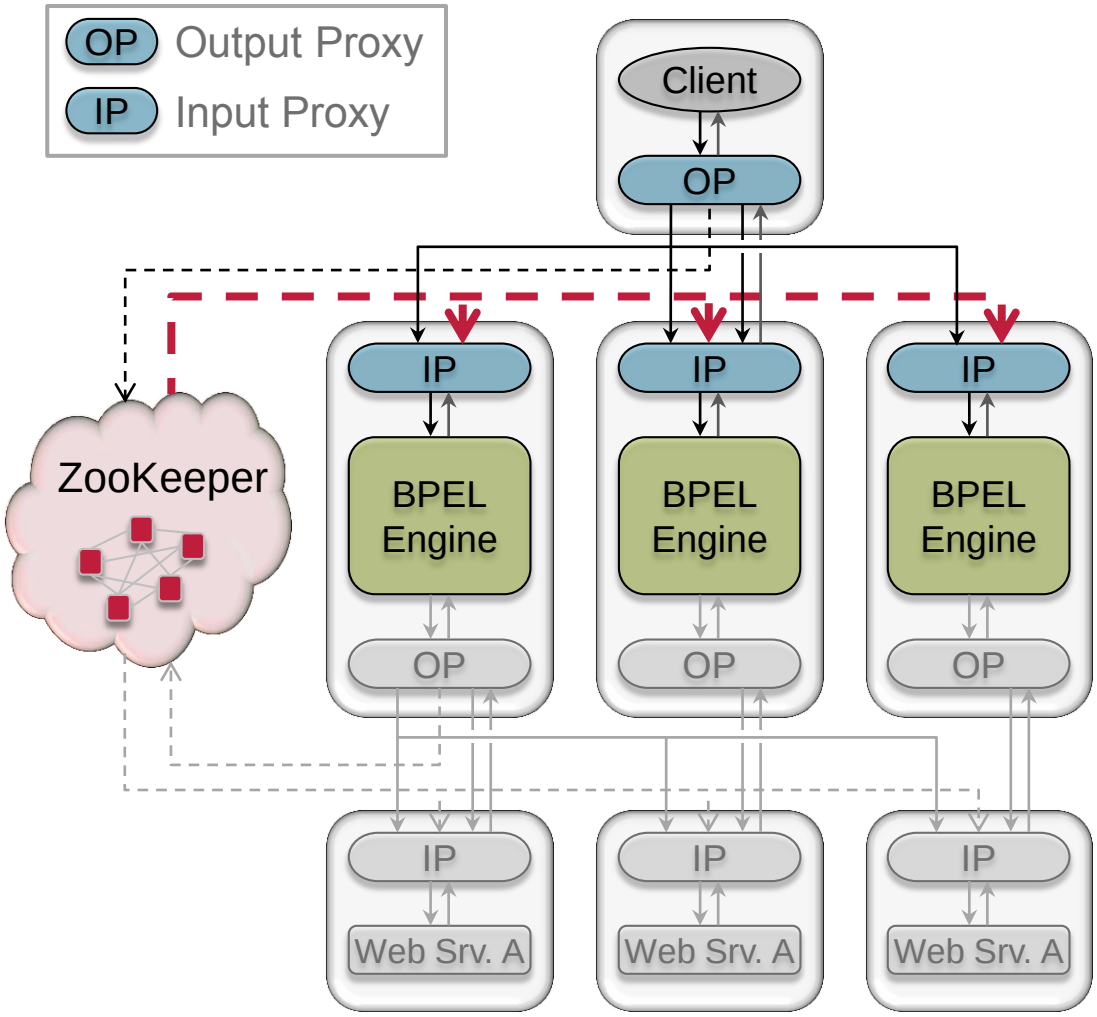
(2) Request data and request ID are sent to input proxies

- ZooKeeper for list of active replicas

(3) Request is registered at ZooKeeper

(4) Output proxy waits for reply

Details – Client / BPEL Stage



(1) Web-service request is passed to local output proxy

(2) Request data and request ID are sent to input proxies

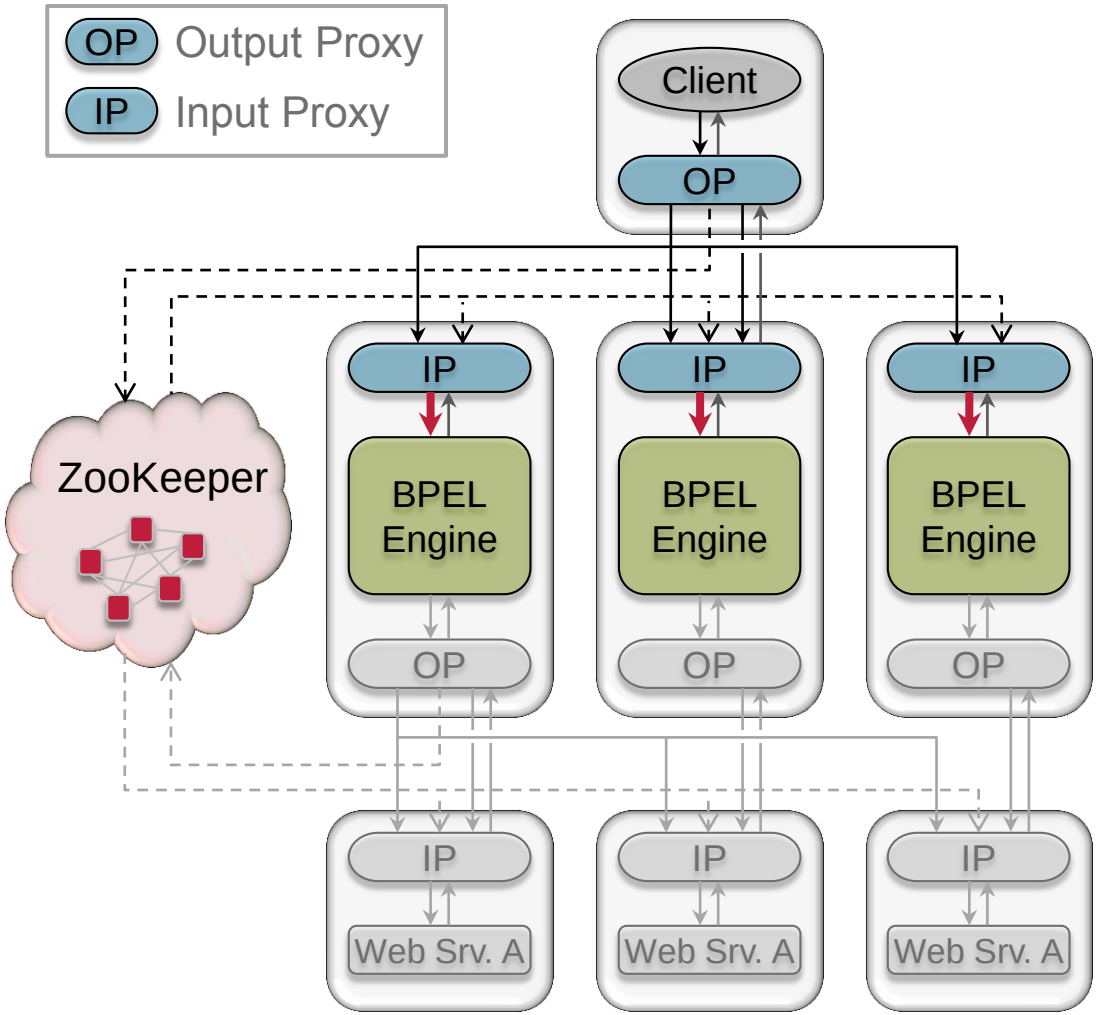
- ZooKeeper for list of active replicas

(3) Request is registered at ZooKeeper

(4) Output proxy waits for reply

(5) Input proxies are informed about the next request

Details – Client / BPEL Stage



(1) Web-service request is passed to local output proxy

(2) Request data and request ID are sent to input proxies
- ZooKeeper for list of active replicas

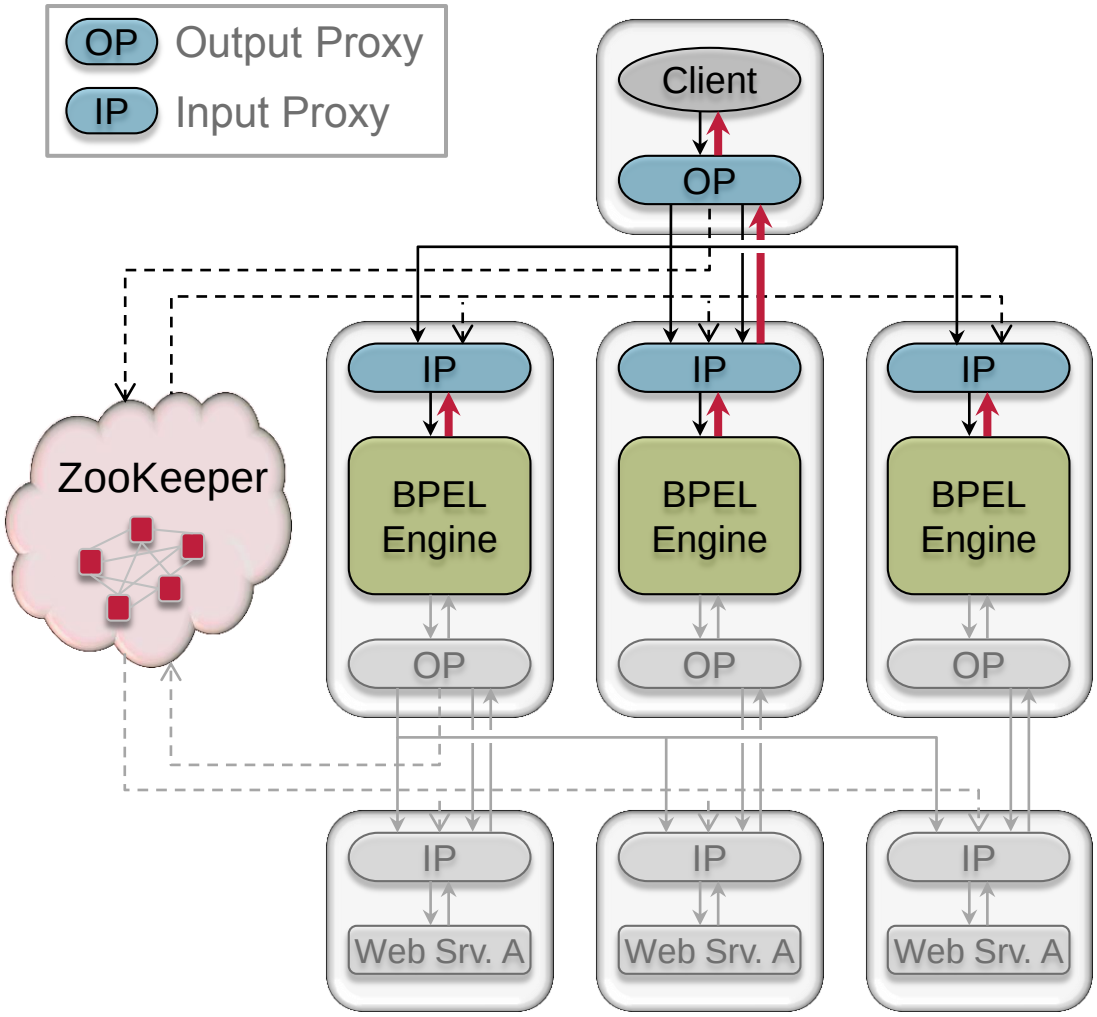
(3) Request is registered at ZooKeeper

(4) Output proxy waits for reply

(5) Input proxies are informed about the next request

(6) Request is delivered to engines
- Transformed interface for request ID

Details – Client / BPEL Stage



(1) Web-service request is passed to local output proxy

(2) Request data and request ID are sent to input proxies
- ZooKeeper for list of active replicas

(3) Request is registered at ZooKeeper

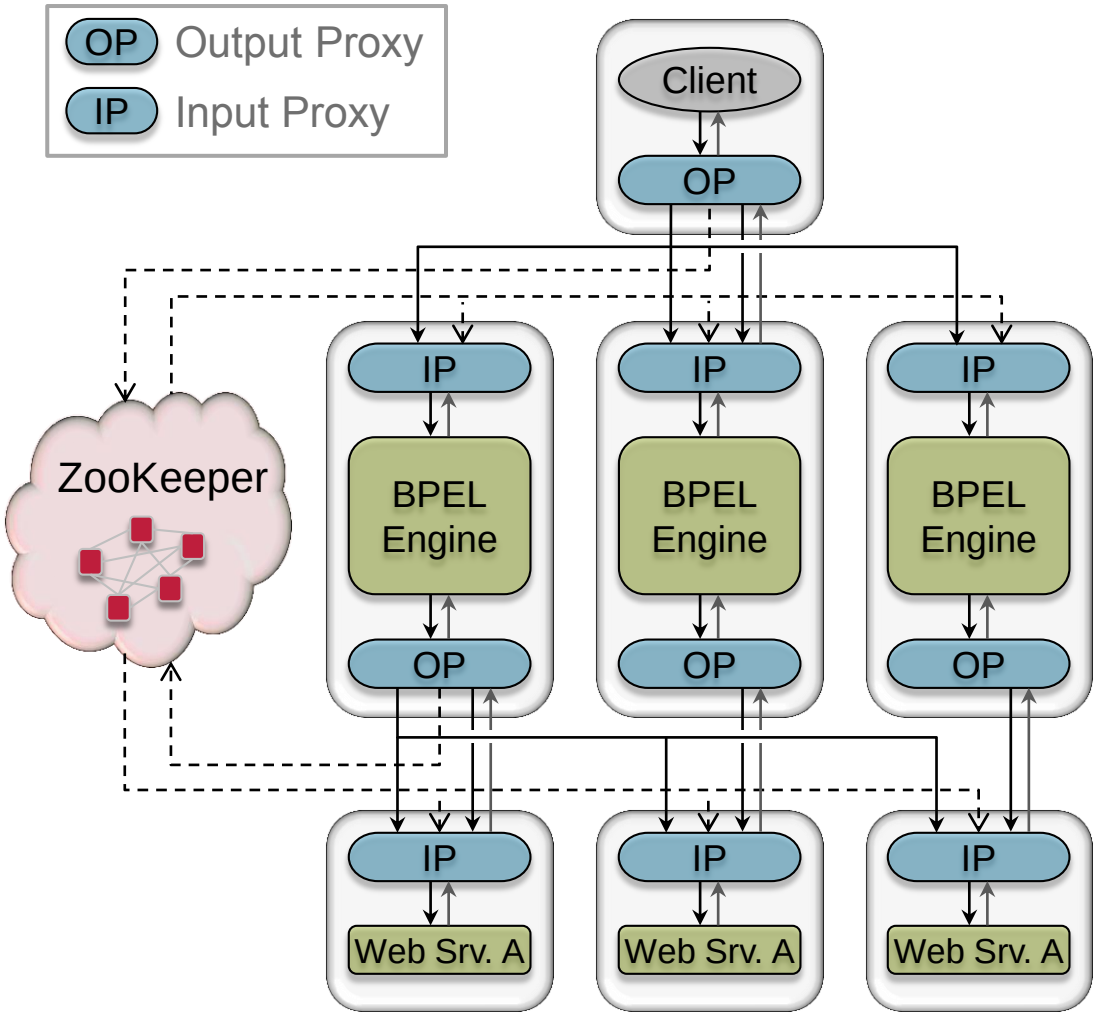
(4) Output proxy waits for reply

(5) Input proxies are informed about the next request

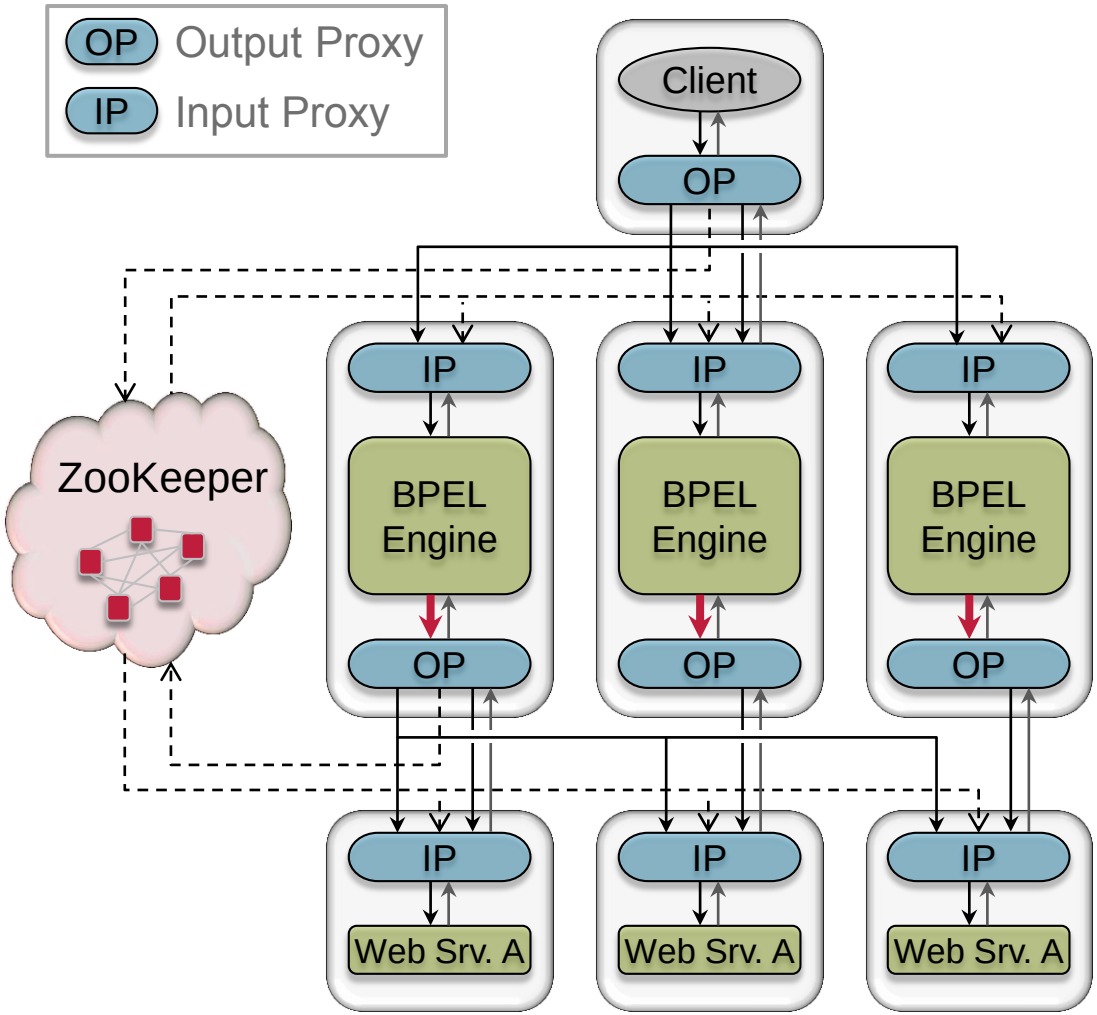
(6) Request is delivered to engines
- Transformed interface for request ID

(7) Results are returned

Details – BPEL / Web-Service Stage



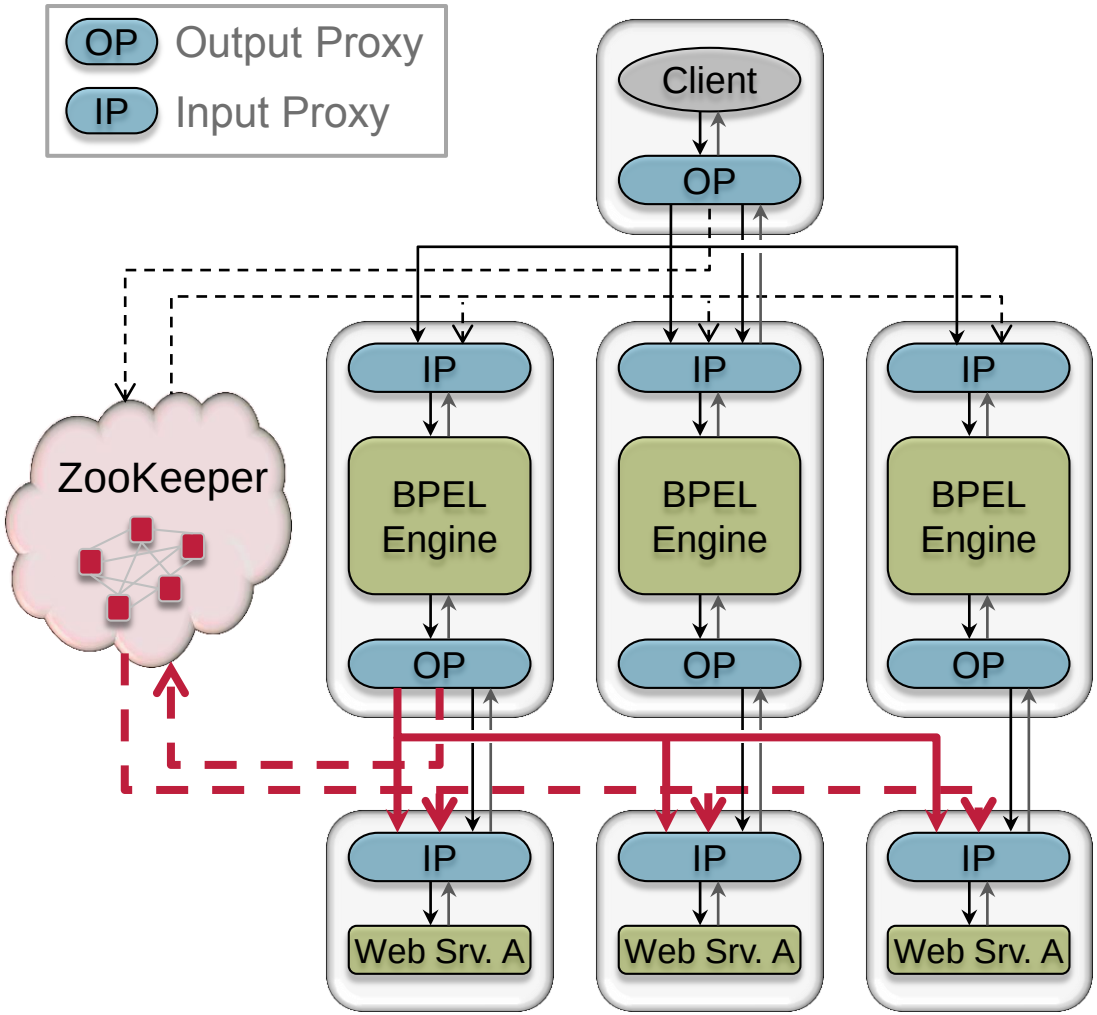
Details – BPEL / Web-Service Stage



(1) Web-service calls are redirected through automated transformation

- Transformation of process definition is conducted only once
- Call ID (request ID + counter) added

Details – BPEL / Web-Service Stage



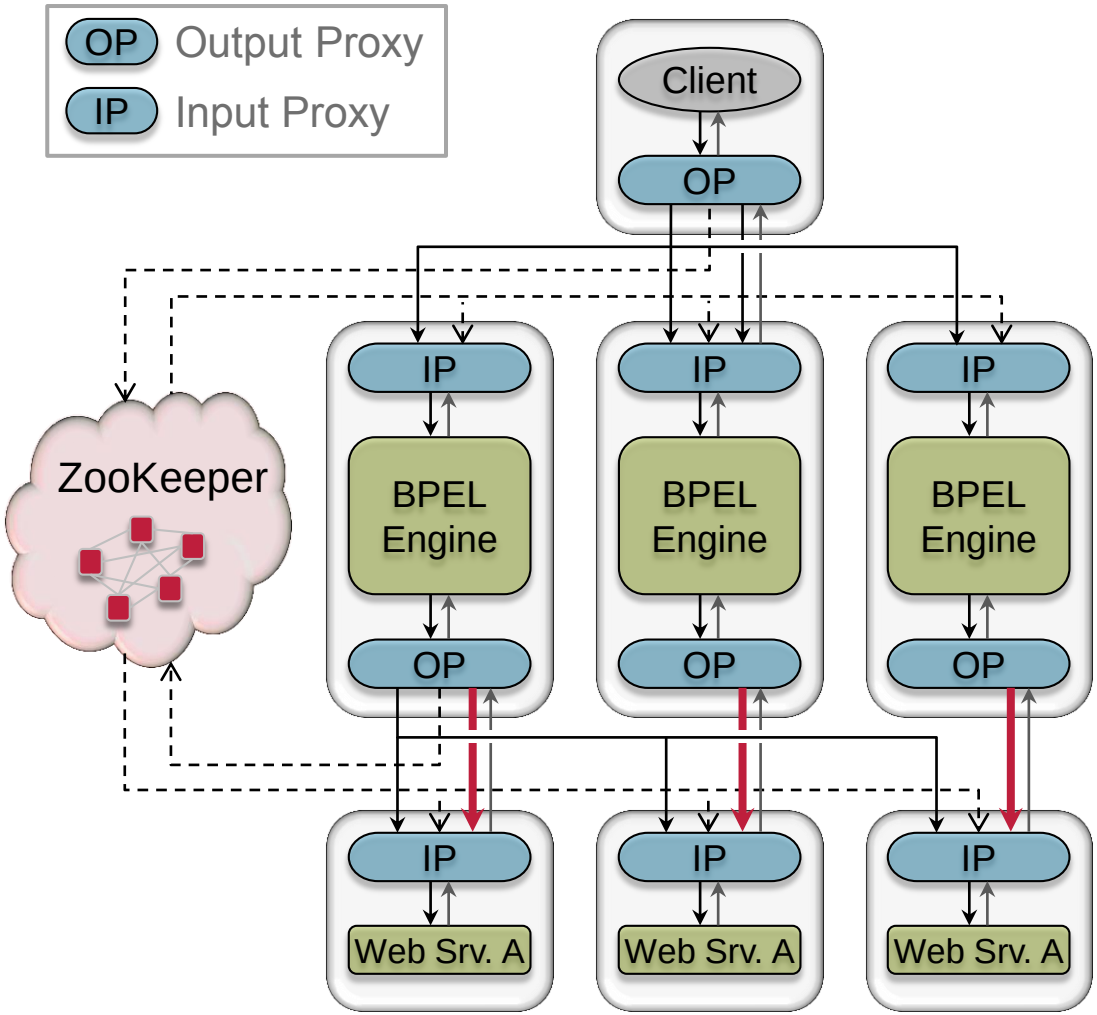
(1) Web-service calls are redirected through automated transformation

- Transformation of process definition is conducted only once
- Call ID (request ID + counter) added

(2) Only leader delivers calls

- ZooKeeper for leader election and crash detection
- Delivery as in first stage

Details – BPEL / Web-Service Stage



(1) Web-service calls are redirected through automated transformation

- Transformation of process definition is conducted only once
- Call ID (request ID + counter) added

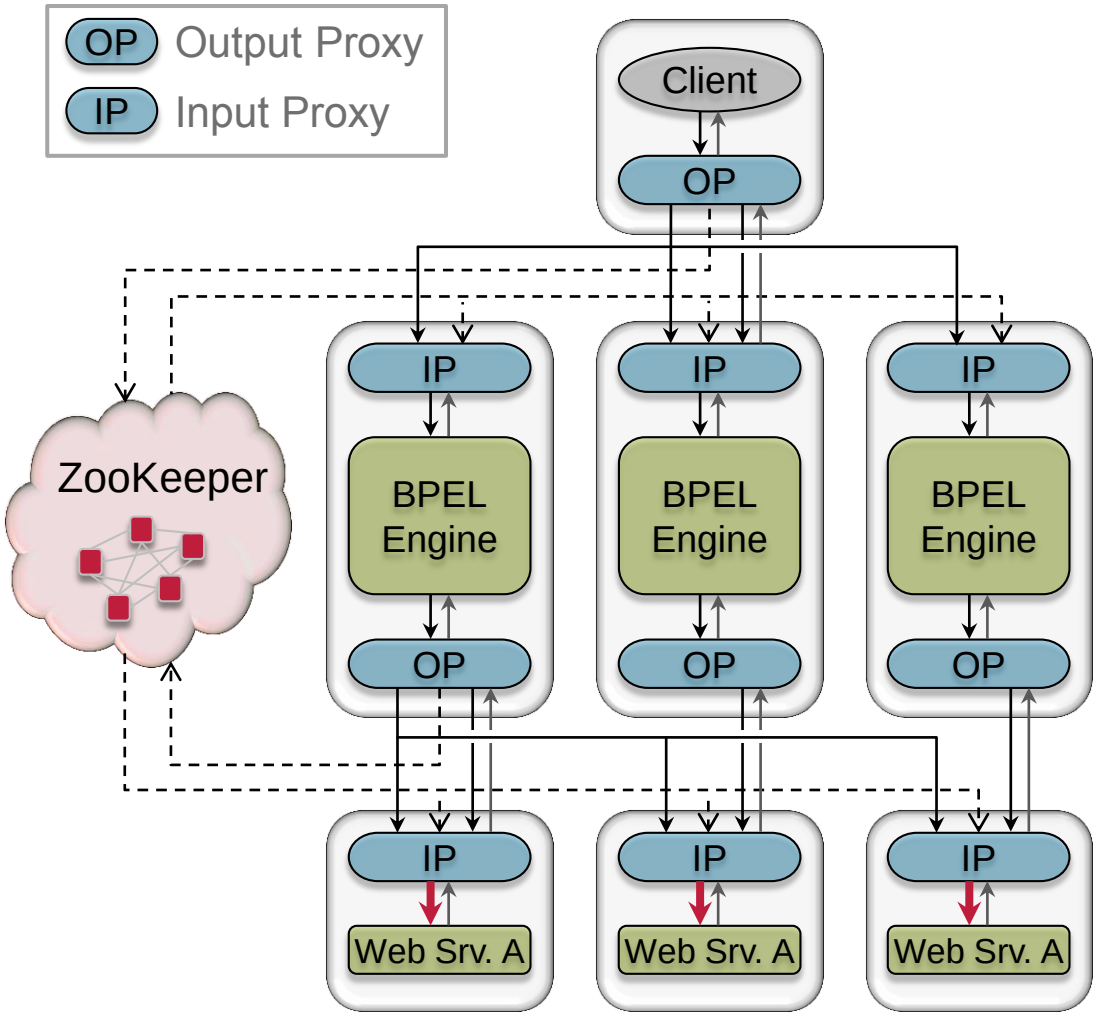
(2) Only leader delivers calls

- ZooKeeper for leader election and crash detection
- Delivery as in first stage

(3) All output proxies wait for reply

- Utilizing call ID

Details – BPEL / Web-Service Stage



(1) Web-service calls are redirected through automated transformation

- Transformation of process definition is conducted only once
- Call ID (request ID + counter) added

(2) Only leader delivers calls

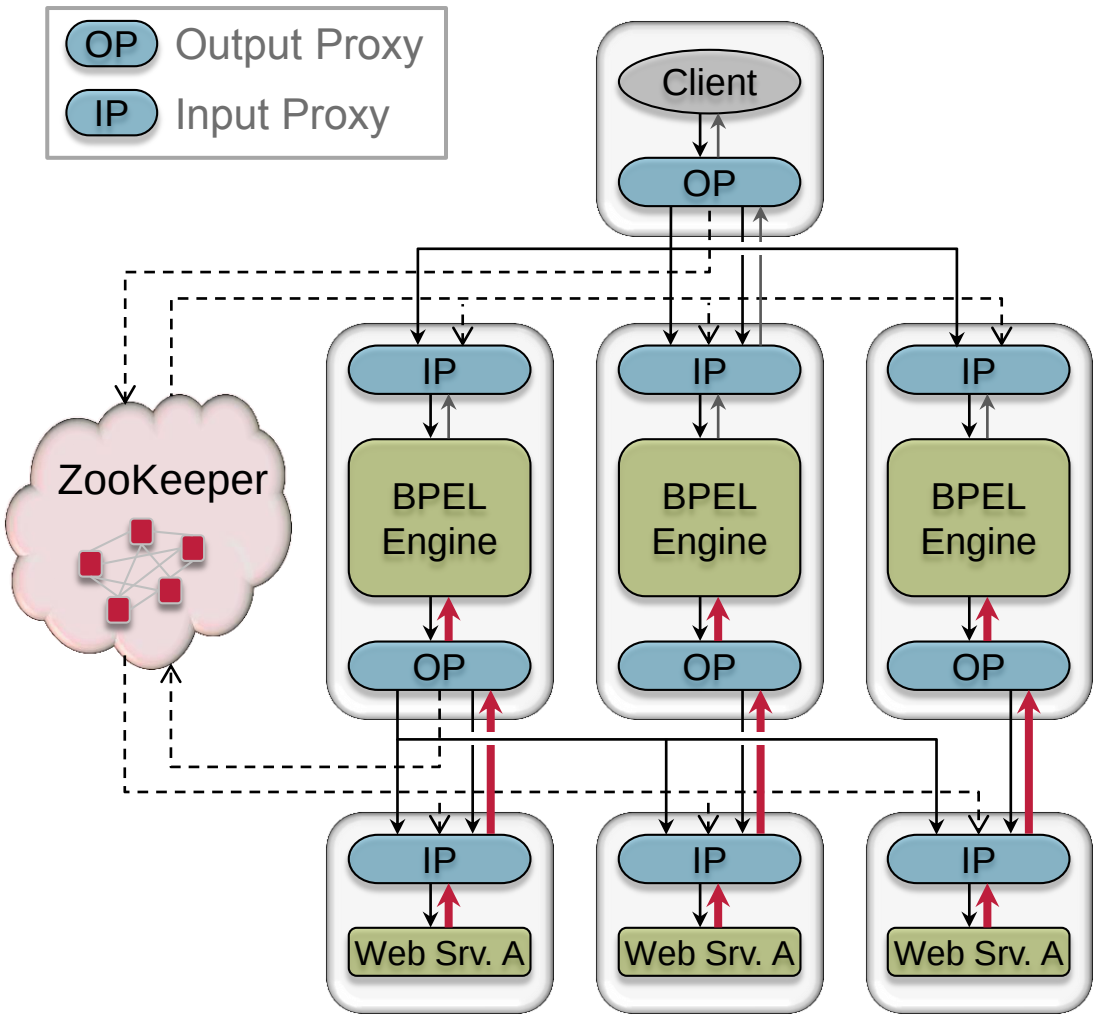
- ZooKeeper for leader election and crash detection
- Delivery as in first stage

(3) All output proxies wait for reply

- Utilizing call ID

(4) Web-service replicas are invoked

Details – BPEL / Web-Service Stage



(1) Web-service calls are redirected through automated transformation

- Transformation of process definition is conducted only once
- Call ID (request ID + counter) added

(2) Only leader delivers calls

- ZooKeeper for leader election and crash detection
- Delivery as in first stage

(3) All output proxies wait for reply

- Utilizing call ID

(4) Web-service replicas are invoked

(5) Results are returned

Reliable BPEL Infrastructure

- Motivation
- Reliable BPEL Infrastructure
- Evaluation
- Outlook and Conclusion

Evaluation

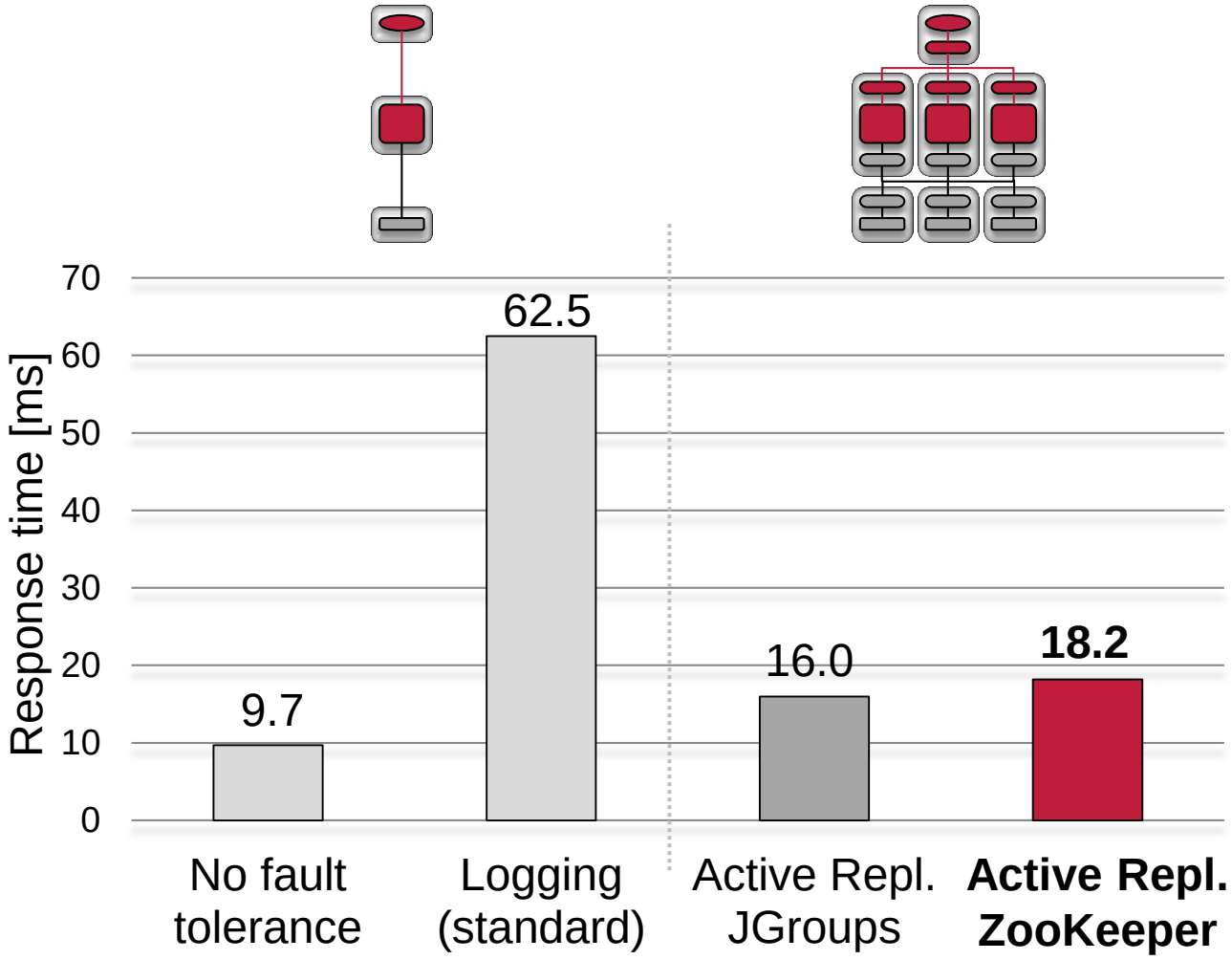
Test Setting

- Prototype implementation using Java
- 16 machines connected over switched Gigabit Ethernet
  - 1 client, 5-times replicated BPEL engine, ZooKeeper, and Web service
- Apache software stack
  - Tomcat, Axis2, ODE

Comparison of response times

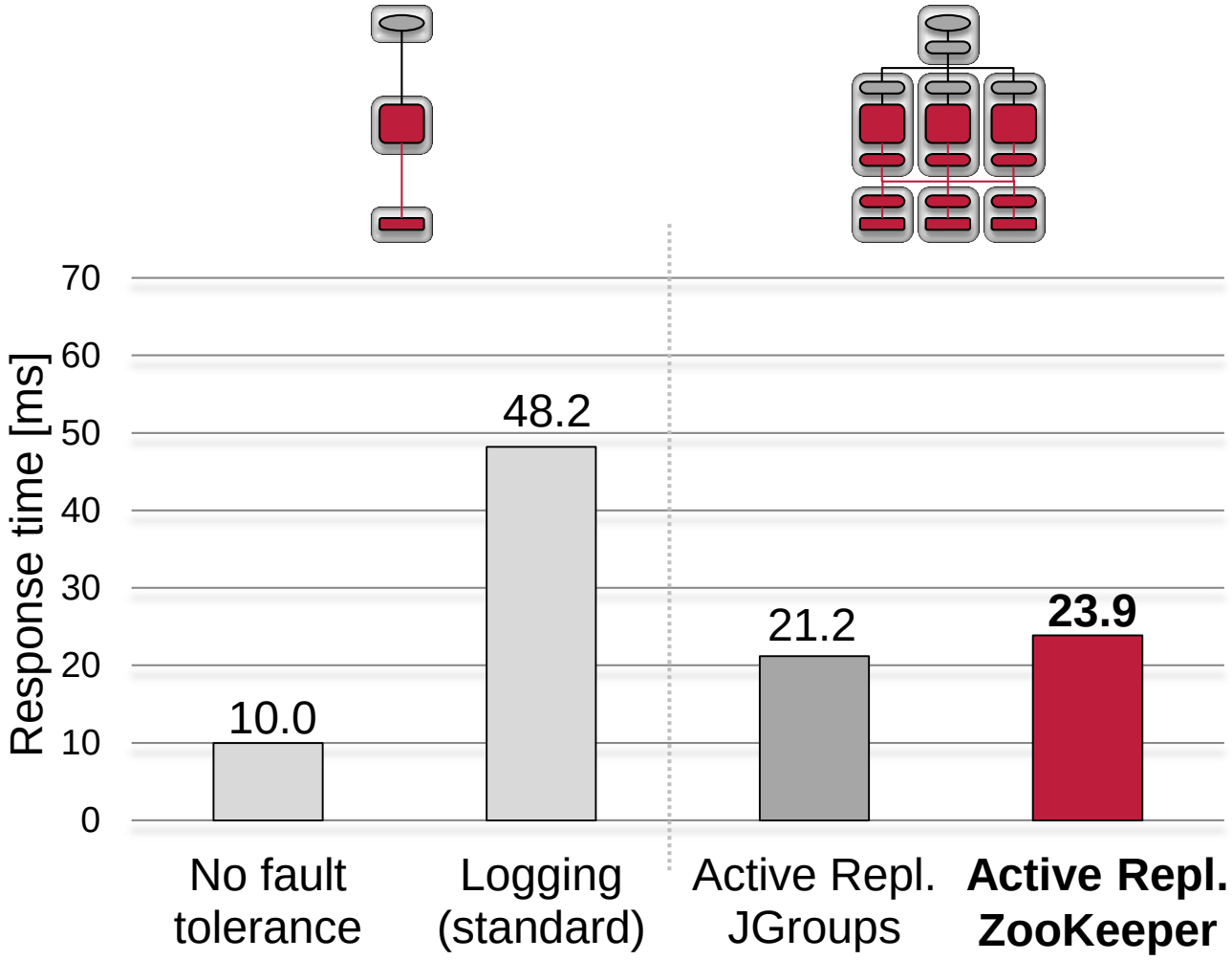
- Unreplicated
  - No crash recovery (without any fault tolerance)
  - Crash recovery enabled (standard system)
- Actively replicated
  - Integrated coordination (group communication over JGroups)
  - Externalized coordination (ZooKeeper)

Evaluation – Client / BPEL Stage



- Request from client to BPEL process without Web service invocation
- Active replication with external coordination 3.4 times faster than standard system
- Moderate overhead of 14% for externalized coordination

Evaluation – BPEL / Web-Service Stage



- Request from BPEL process to Web service
- Active replication with external coordination 2.0 times faster than standard system
- Moderate overhead of 13% for externalized coordination

Reliable BPEL Infrastructure

- Motivation
- Reliable BPEL Infrastructure
- Evaluation
- Outlook and Conclusion

Outlook

Tolerating arbitrary faults (Byzantine faults)

- Hardware errors
- Malicious attacks

Distribute replicas over multiple clouds (cloud-of-clouds)

- Independence from a single cloud provider

Cloud infrastructure management over coordination service

- Platform and resource allocation
- Failure detection and dynamic reconfiguration
- Job control

Conclusion

Fault-tolerant execution of Web-service–based workflows in context of cloud computing is currently an open issue

BPEL would be a perfect supply for the demand…

- …but level of fault tolerance is not sufficient

Our proposed reliable BPEL infrastructure:

- Active replication of all components
- Transparency through automated transformation of workflows
- Utilizing of external coordination

Thank you for your attention!

Providing Fault-tolerant Execution of Web-service–based Workflows within Clouds

Johannes Behl
(behl@ibr.cs.tu-bs.de)

TClouds

http://www.tclouds-project.eu/

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement number ICT-257243.