Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Elasticity Through Modularity

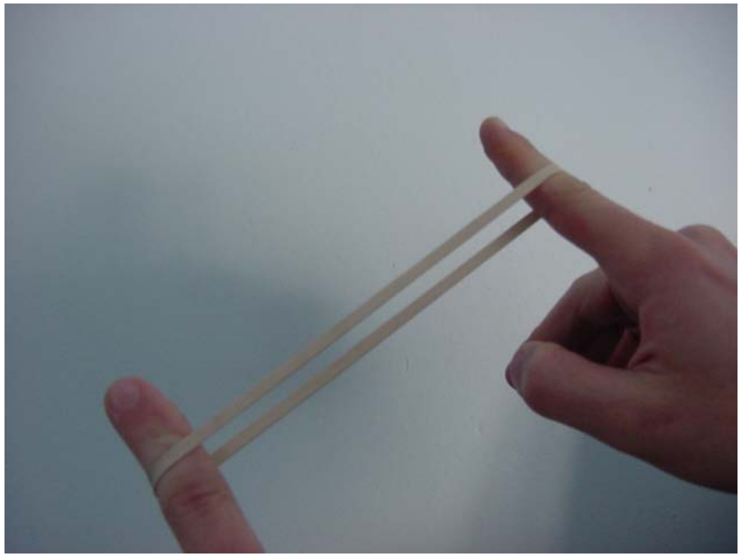
Jan S. Rellermeyer

Systems Group, ETH Zürich

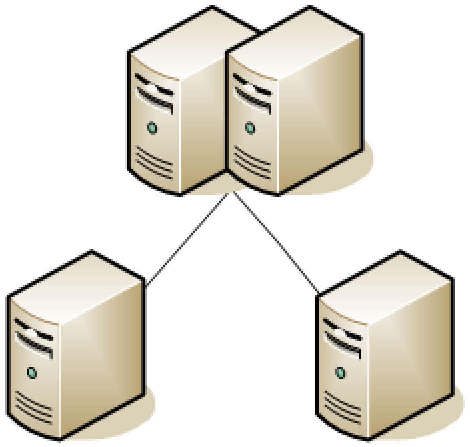
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

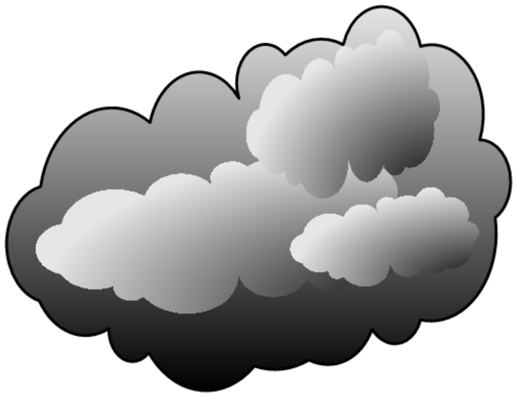
Elasticity

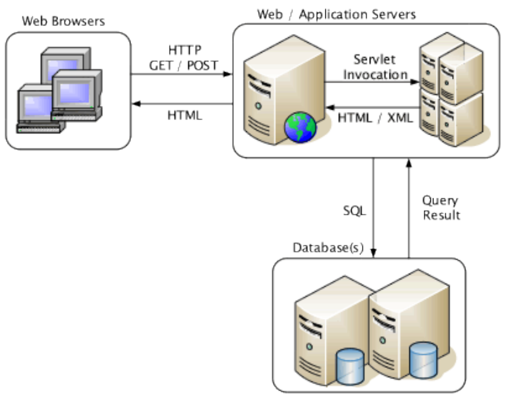
- the ability to acquire and release resources on demand
- elastic infrastructure (like EC2): virtualization
- elastic software?

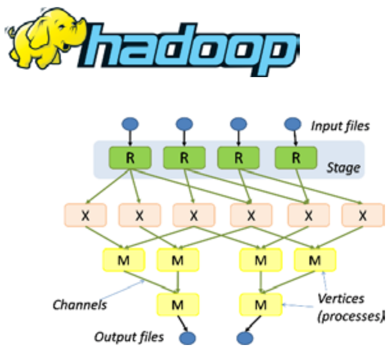


Software Elasticity



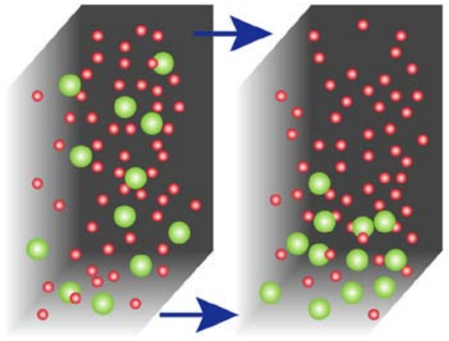




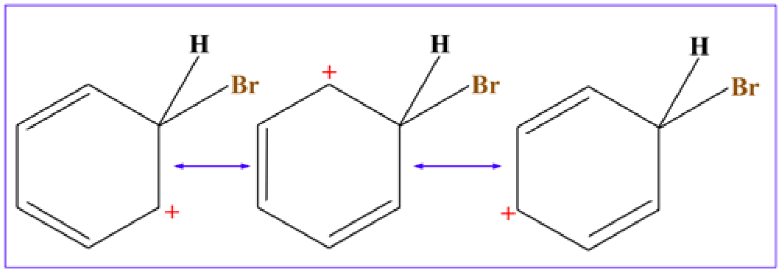


Elastic Systems

- Fluidity



- Delocalization



Modularity as a System Design Principle

- Modules as units of encapsulation
- Modules as units of deployment
- Plain old modules

- Tradeoffs are well understood in software engineering

Two components are loosely coupled, when changes in one never or rarely necessitate a change in the other

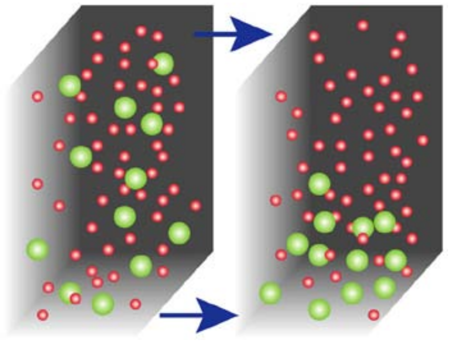
coupling

cohesion

A component exhibits high cohesion when all its functions/methods are strongly related in terms of function

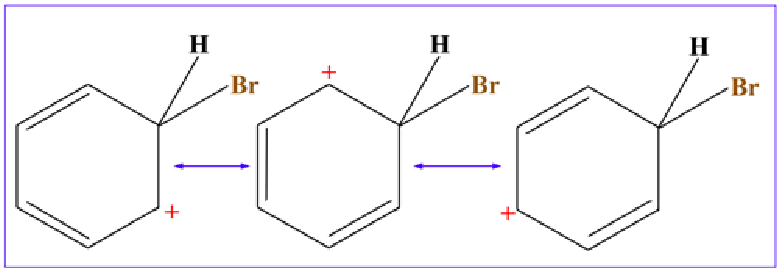
Elastic Modular Systems

- Fluidity



cohesion

- Delocalization



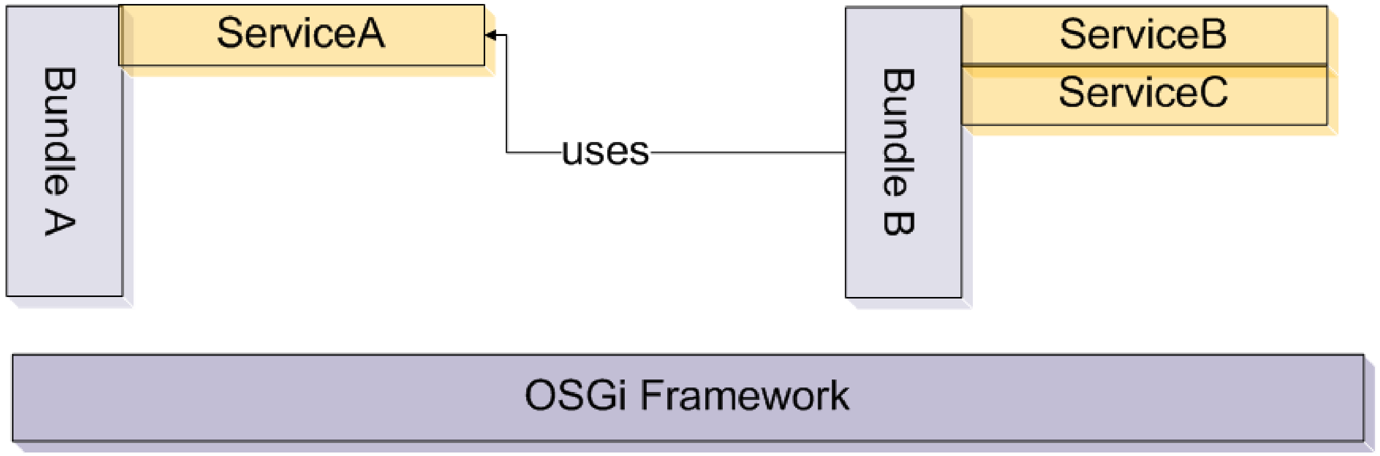
coupling

OSGi: Dynamic Modules for Java

- Open Standard, well supported by major vendors
  - App servers, Eclipse IDE, Embedded Software, Mobile Phones
- Modules are called *Bundles*
  - JAR files with additional metadata
- Runtime system: The *Framework*
  - Lifecycle management
- Bundles implement isolation and locality
- Interaction between bundles is limited
  - Shared code through package imports
    (explicit dependencies, tight coupling)
  - Inter-bundle calls through services
    (loose coupling)
  - Monitoring system state through events

OSGi™ Alliance

OSGi



- Lifecycle of each Bundles can be controlled individually
- Services are registered and retrieved through a central service registry (in-VM SOA)
- The system is dynamic

Software Modules for the Cloud

- Life-Cycle Management
  - Provision components, update components
- Composition
  - Make components communicate
- Fabric of the Cloud:
  - Distributed System
  - Potential node failures and link failures

R-OSGi

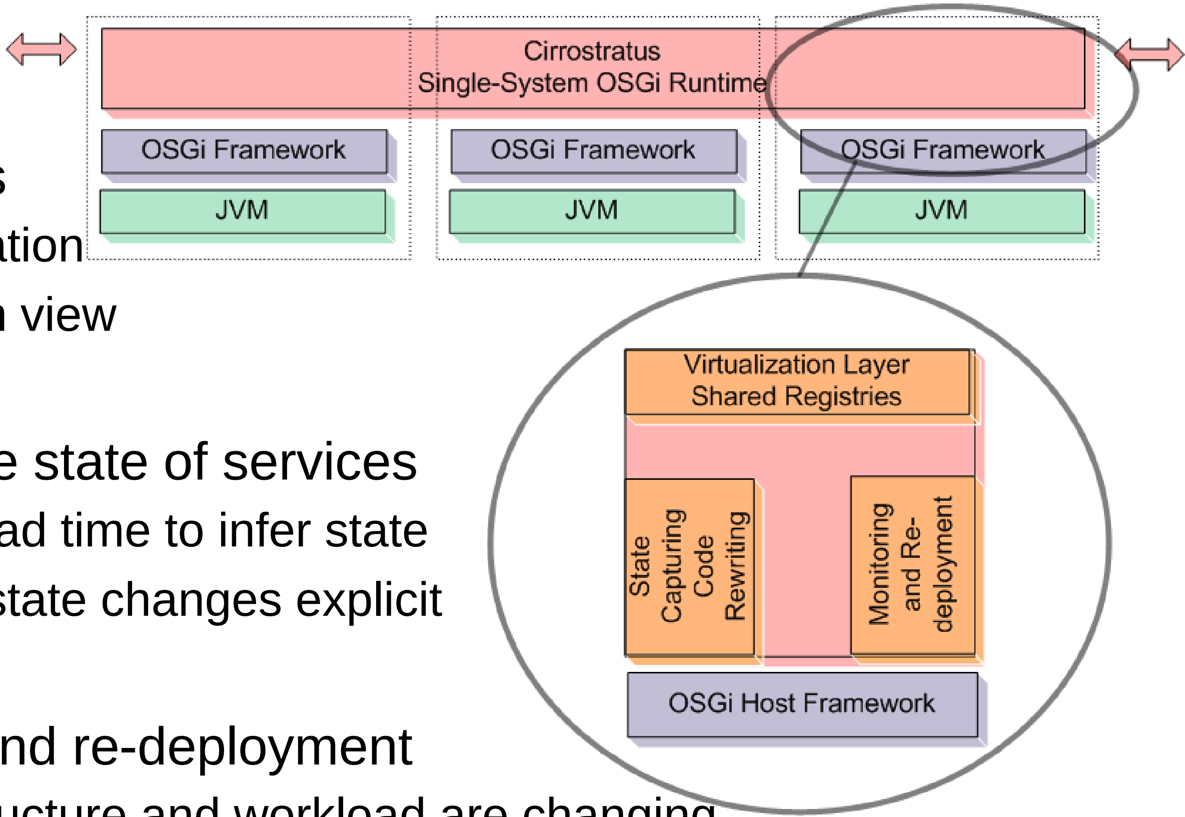
- Approach: Assimilate Complexity into a Runtime System

[J.S. Rellermeyer, G. Alonso, T. Roscoe: *R-OSGi - Distributed Applications through Software Modularization*. In: Middleware 2007]
[J.S. Rellermeyer, M. Duller, and G. Alonso: *Engineering the Cloud from Software Modules*. In: ICSE-Cloud 2009].

Cirrostratus

A Runtime System for Elastic Modules

- Provide a "Single System Image" for modular applications
  - Single OSGi Framework
- Virtual Modules,Services
  - For migration and replication
  - Provide a global, uniform view
- Capture and replicate the state of services
  - Symbolic execution at load time to infer state
  - Code rewriting to make state changes explicit
- Continuous monitoring and re-deployment
  - Optimize despite infrastructure and workload are changing



Cirrostratus

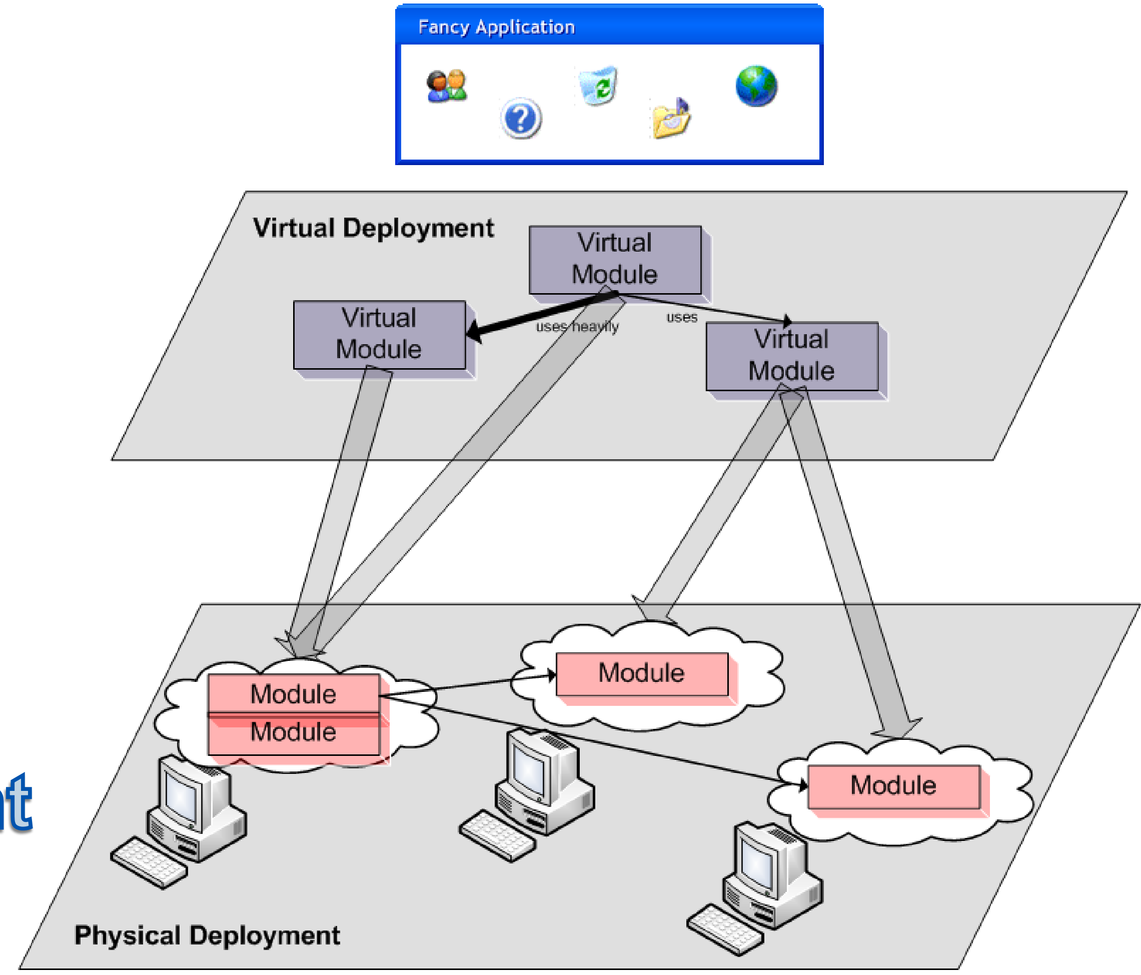
virtual deployment

↓

+ implicit requirements
+ non-functional requirements

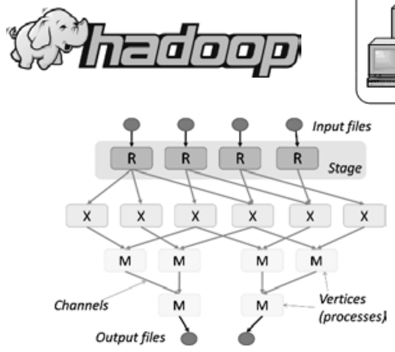
↓

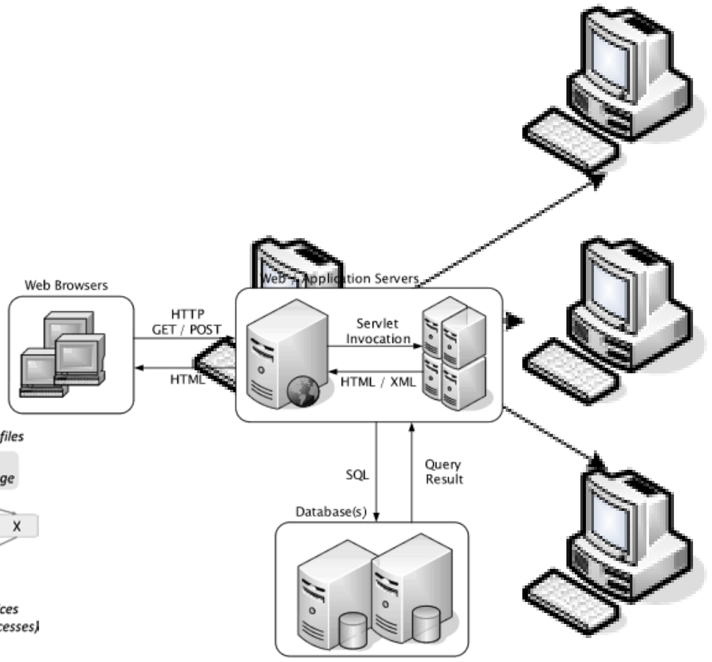
physical deployment



Elasticity and The Problem of State

- Idea: replicate services on demand
- Problem: It makes a difference if you have one service or ten services
- State!





Inferring State: Symbolic Execution

- For each service, perform an abstract interpretation when the module is loaded the first time
- Interprete the code in terms of symbols rather than concrete values
- Determine how state propagates through the system
- Capture the state through bytecode-rewriting

```
add(I)I
   L0
    ALOAD 0
    DUP
    GETFIELD test/Simple.state : I
    ILOAD 1
    IADD
    PUTFIELD test/Simple.state : I
   L1
    ALOAD 0
    GETFIELD test/Simple.state : I
    IRETURN
   L2
    LOCALVARIABLE this Ltest/Simple;
    L0 L2 0
    LOCALVARIABLE i I L0 L2 1
    MAXSTACK = 3
    MAXLOCALS = 2
```

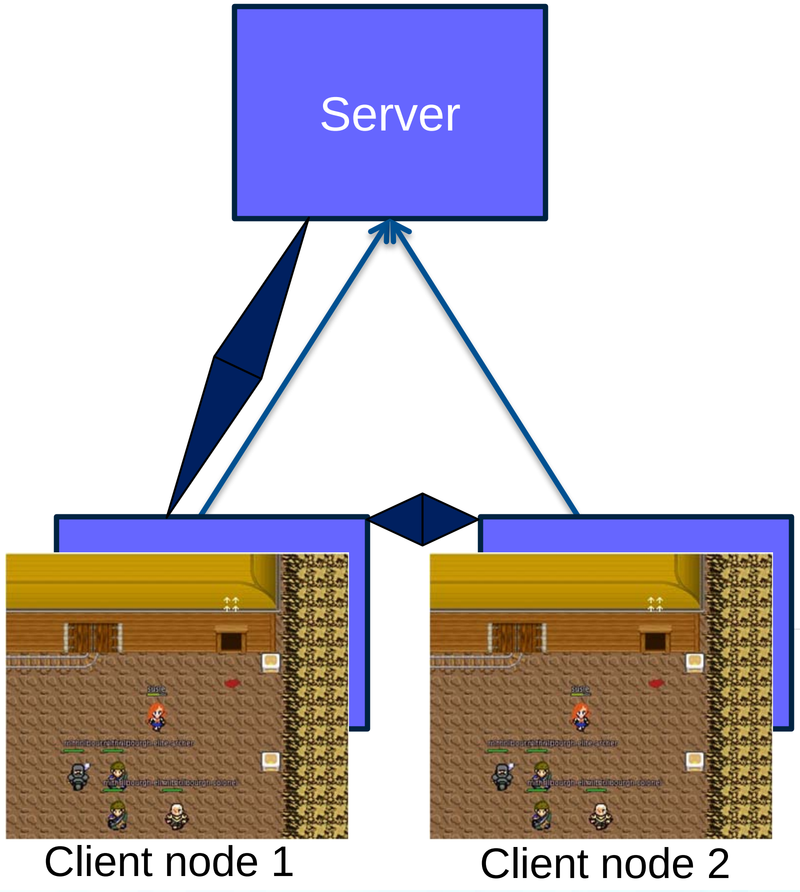
Monitoring and Re-deployment

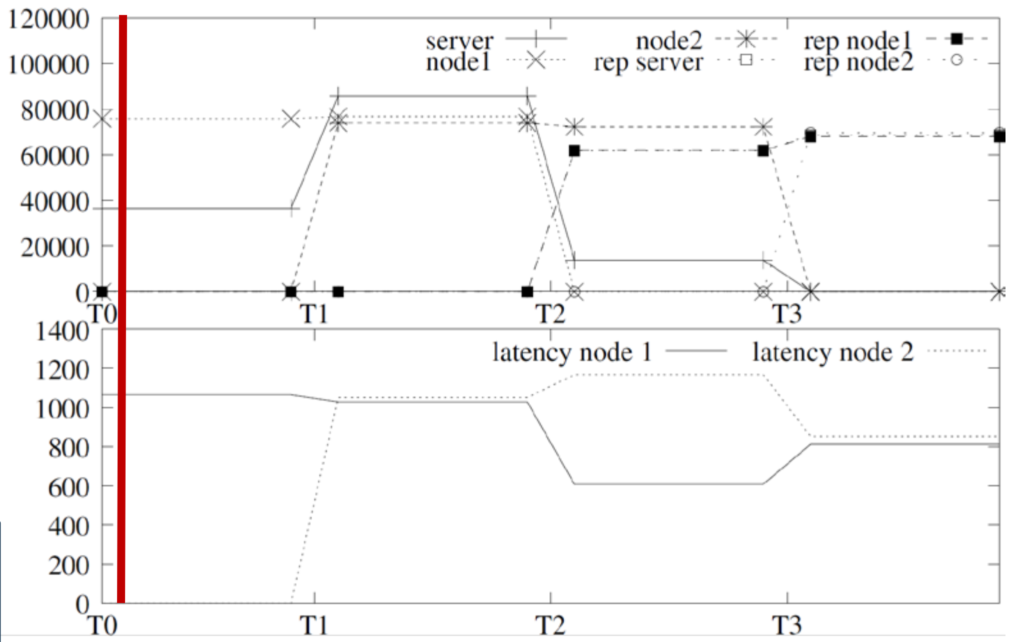
- System inserts performance probes into the code
- Controllers can sense the running application
- System provides interfaces to trigger actions
  - e.g., eigrate a service, replicate a service, drop a replica, rebind a service

- Controllers are typically provided by the application
- Have application-specific knowledge
- Know non-functional requirements

Use Case: Stendhal

- Client/server online game





Future Work

- Generalizing the ideas of modularity as a systems design principle beyond Java and OSGi
  - We did it for services in C, nesC, through R-OSGi
  - OSGi-kind of runtime for the .net CLR
- Build interesting applications
  - Porting .net CF to Lego Mindstorms NXT for swarms of robots
- Supported by the Microsoft Innovation Cluster for Embedded Software (ICES)
- Graduate ☺

CONCLUSIONS

- Software elasticity is challenging
- Modularity is key to facilitating elastic deployments of software
- The arising complexity such as the problem of state replication can be mitigated by an intelligent runtime system like Cirrostratus