A FINE-GRAINED AUTONOMIC MANAGEMENT SOLUTION FOR MULTI-LAYERED SYSTEMS

F. Seracini, M. Menarini, I. Kruger
L. Baresi, **S. Guinea**, G. Quattrocchi
sam.guinea@polimi.it

UCSD Jacobs | Computer Science and Engineering

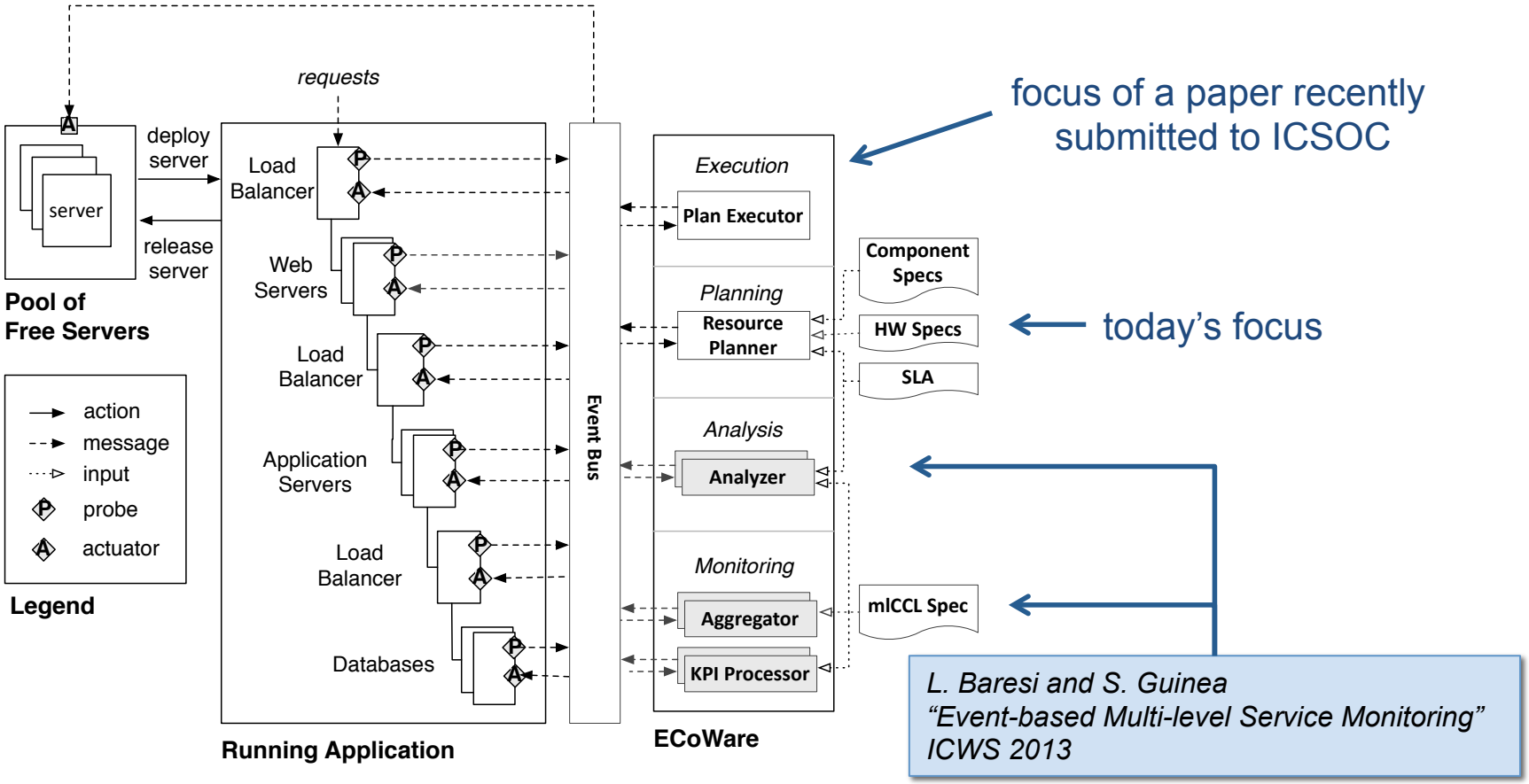


Context and Problem

- Modern Software Systems are subject to continuous changes
  - Requirements
    - Multiple stakeholders with different interests
  - Context of execution
    - Service evolution – distributed ownership
- We focus on complex Web-based systems
  - Fluctuations in the system's workload
  - Avoid over-allocation of resources
  - Provide more efficient systems
- We need fine-grained autonomic capabilities that cover all the system's facets, from their hardware resources to their software
- We exploit and suitably extend ECoWare

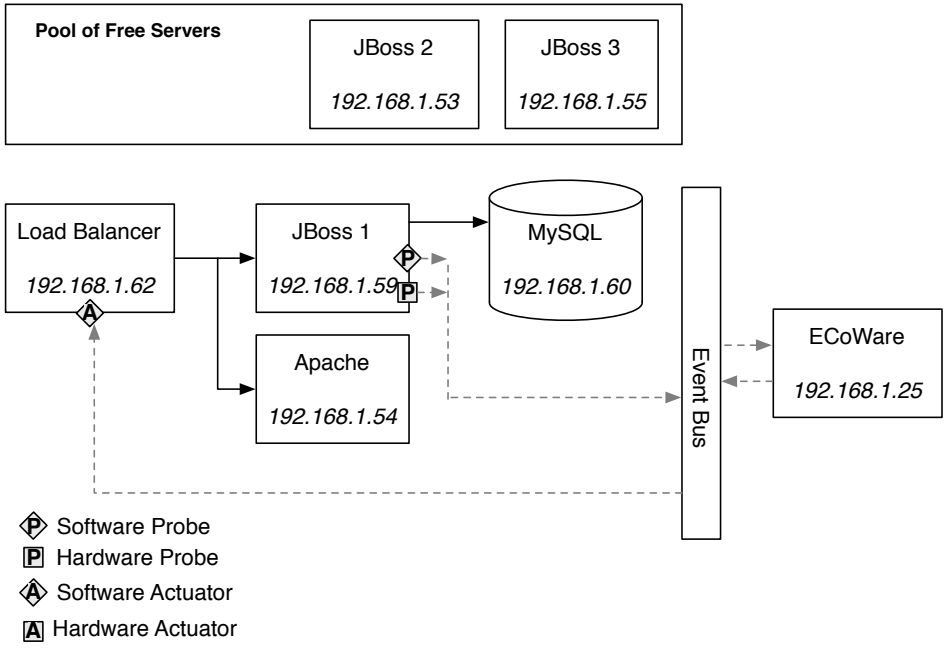
The ECoWare Initiative

Empower the development of *holistic* self-adaptive systems



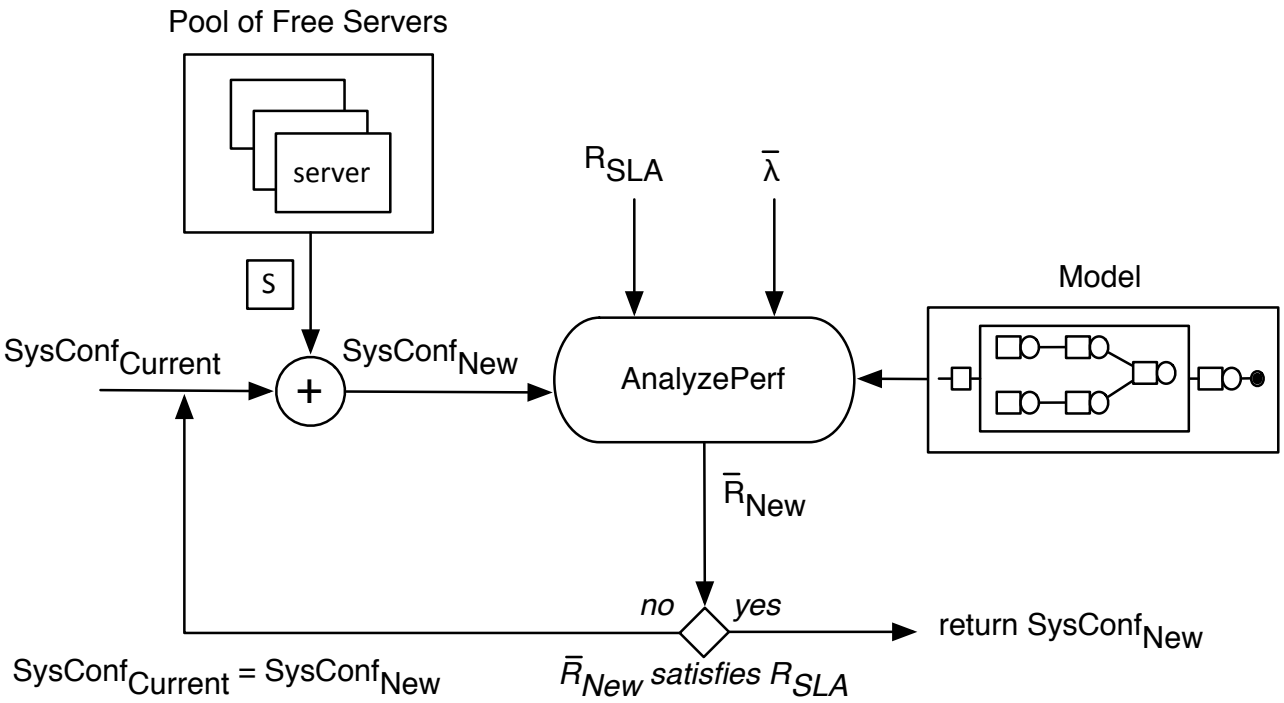
Running Example

- RUBiS
  - Simulates a common auction web site
  - HTML, Java Servlets and SQL
- Apache Server for static content
- Jboss App Servers for dynamic content
- MySQL for backend data tier
- Content-aware load balancer performs the routing



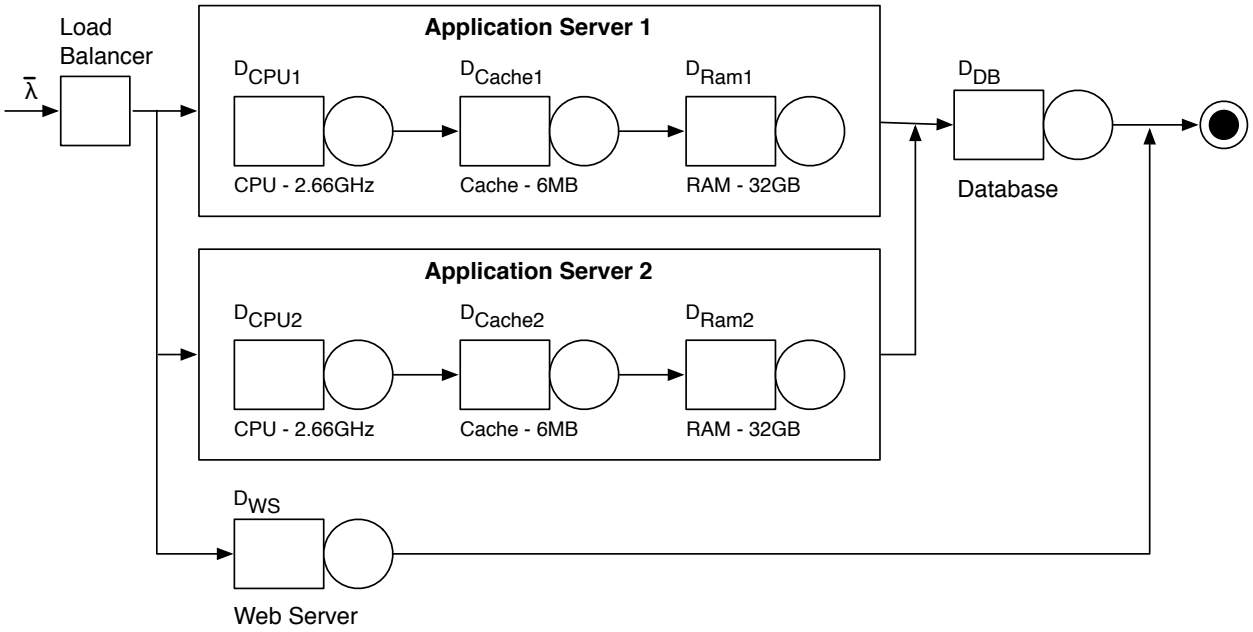
The 95th percentile response time should be within a fixed threshold over a period of time

Resource Planner



Performance Model

- The model considers the app's multiple tiers and multiple levels



We used JMT as our model solver

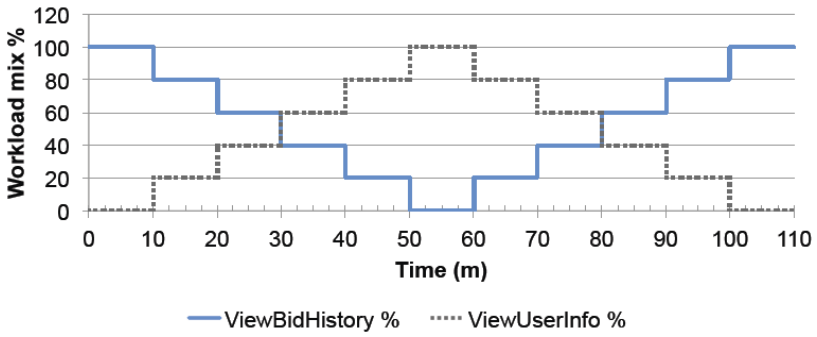
Calibration of the Performance Model

- We profiled the servlets in Rubis
  - Selected servlets with different resource usage profiles
  - Increased resource usage profiles of the servlets to make them more realistic

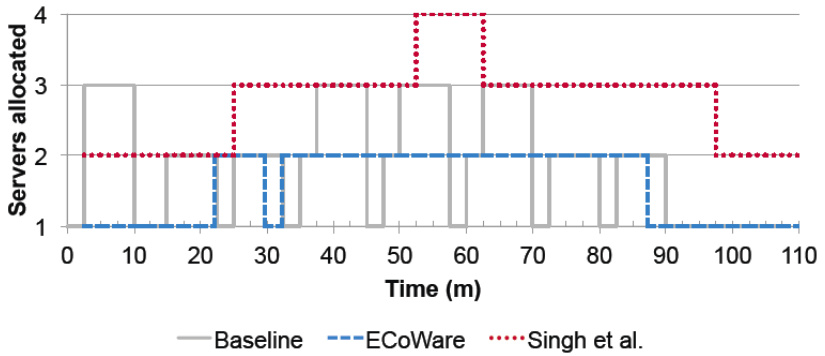
- We collected data regarding the service demands
  - Apache RT
  - Database RT
    - Wrapped Hibernate
  - Application server
    - Extended HTTPServlet class
      - Native calls for thread matching
    - Modified Linux kernel to measure
      - # of CPU cycles * CPU cycle time
      - # of LLC missed * main memory avg. access time (used a free profiler)

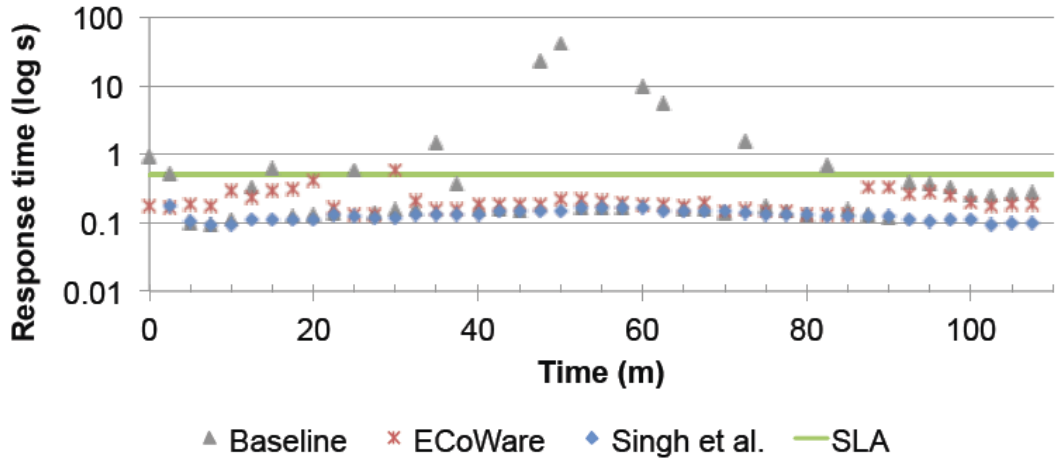
Experimental Evaluation: Workload One

Workload



Number of servers allocated





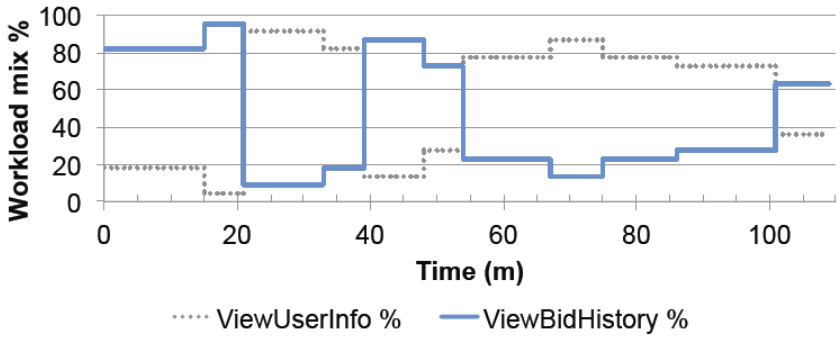
95% Percentile response time

Results:

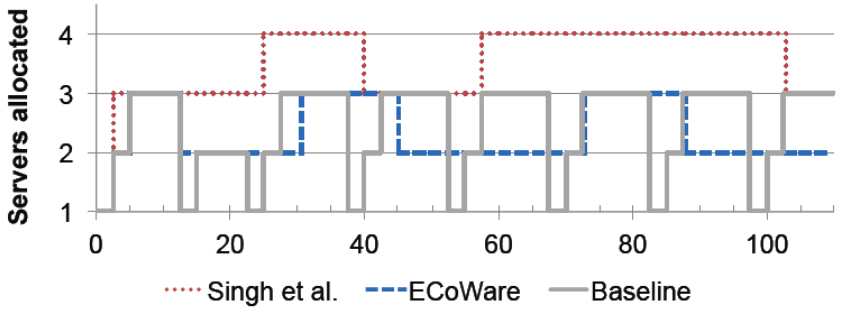
- 13.75% fewer server-minutes than baseline, and fewer violations
- 42.5% fewer server-minutes than Singh et al. [19], one minor violation*

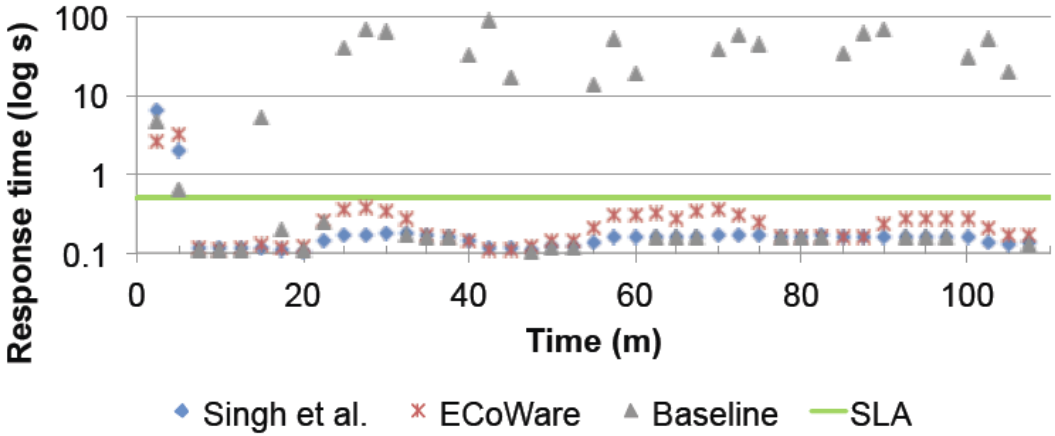
Experimental Evaluation: Workload Two

Workload



Number of servers allocated





95% Percentile response time

Results:

- 16% fewer server-minutes than baseline, and fewer violations
- 33% fewer server-minutes than Singh et al. [19], same violations*

Conclusion

- Allocation strategies that take into account the resource usage profiles of the different requests help increases the accuracy of our performance predictions
  - allows us to improve the utilization of our infrastructure

- Extend the work to consider hybrid-cloud deployments of Web applications
  - We have already begun working in this direction
  - Strong focus on distributed and coordinated adaptation