ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

IKS Information and Communication Systems Research Group

Autonomic Execution of Web Service Compositions

Cesare Pautasso, Thomas Heinis, Gustavo Alonso
Department of Computer Science
ETH Zurich, Switzerland

pautasso@inf.ethz.ch

Context: Process-based Composition

- Web services built as **process-based compositions** of other Web services

Scalability on Clusters of Computers

- Process Management Infrastructure needs to scale (many clients, many conversations)
- Web Service Composition Engines run on **cluster** of computers to **handle large workloads** [IJEC'04]

The Problem: How to Configure the Engine?

- The distributed engine needs to be configured:
  - Based on its **current (unpredictable) workload**
  - Based on the **available resources** of the cluster
- How many resources of the cluster should be assigned to the engine?
- Difficult to configure the engine *apriori*
- Difficult to manage the system *manually*

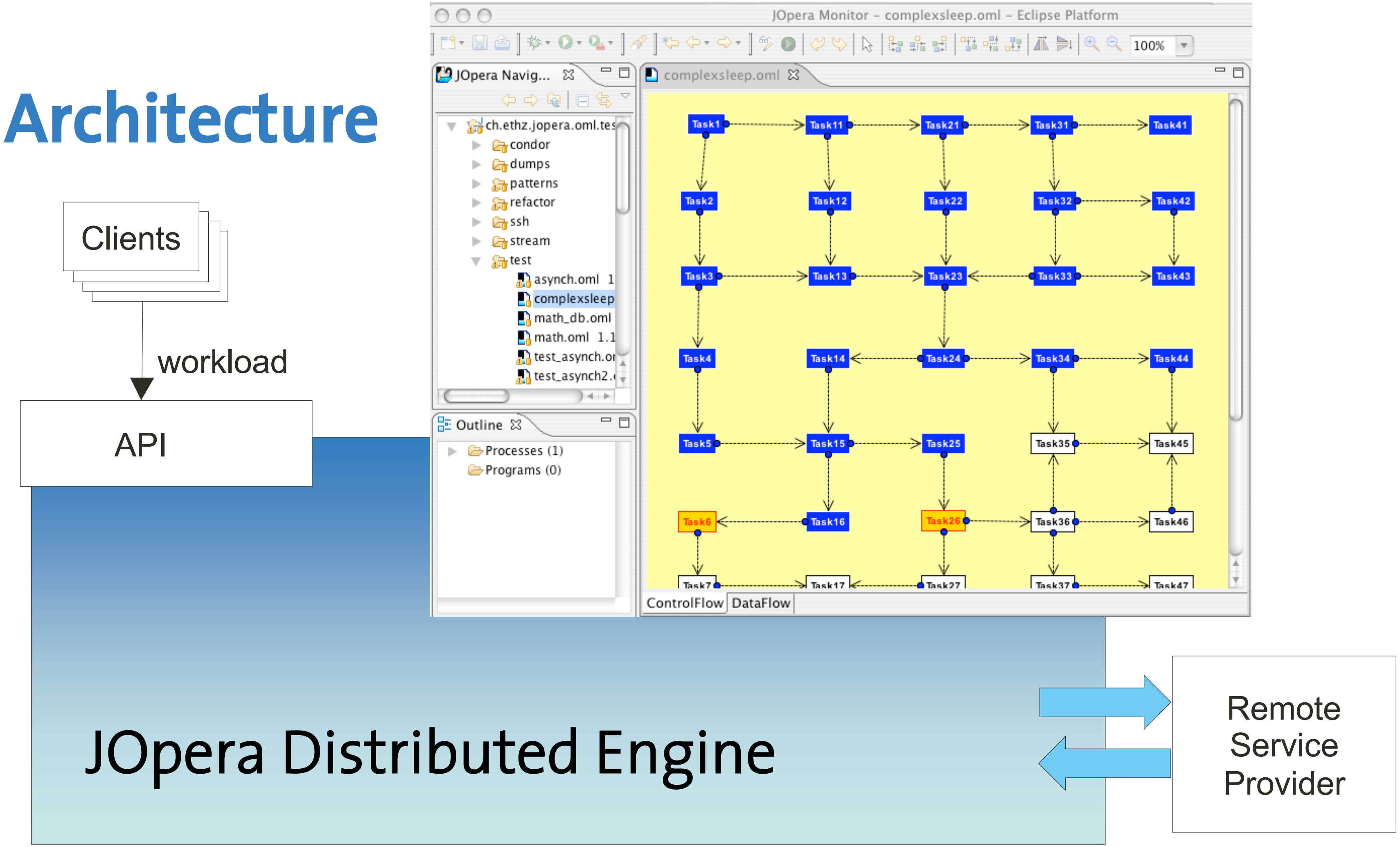
The Solution: Autonomic Computing

- The engine should configure itself
- Trade-off between two goals:
  - *Best Performance (response time, throughput, ...)*
  - *Best Resource Allocation (size of the cluster)*
- Requirements for the distributed engine design:
  - Support on-the-fly reconfiguration
  - Provide access to internal performance metrics
  - Expose an API for controlling the configuration

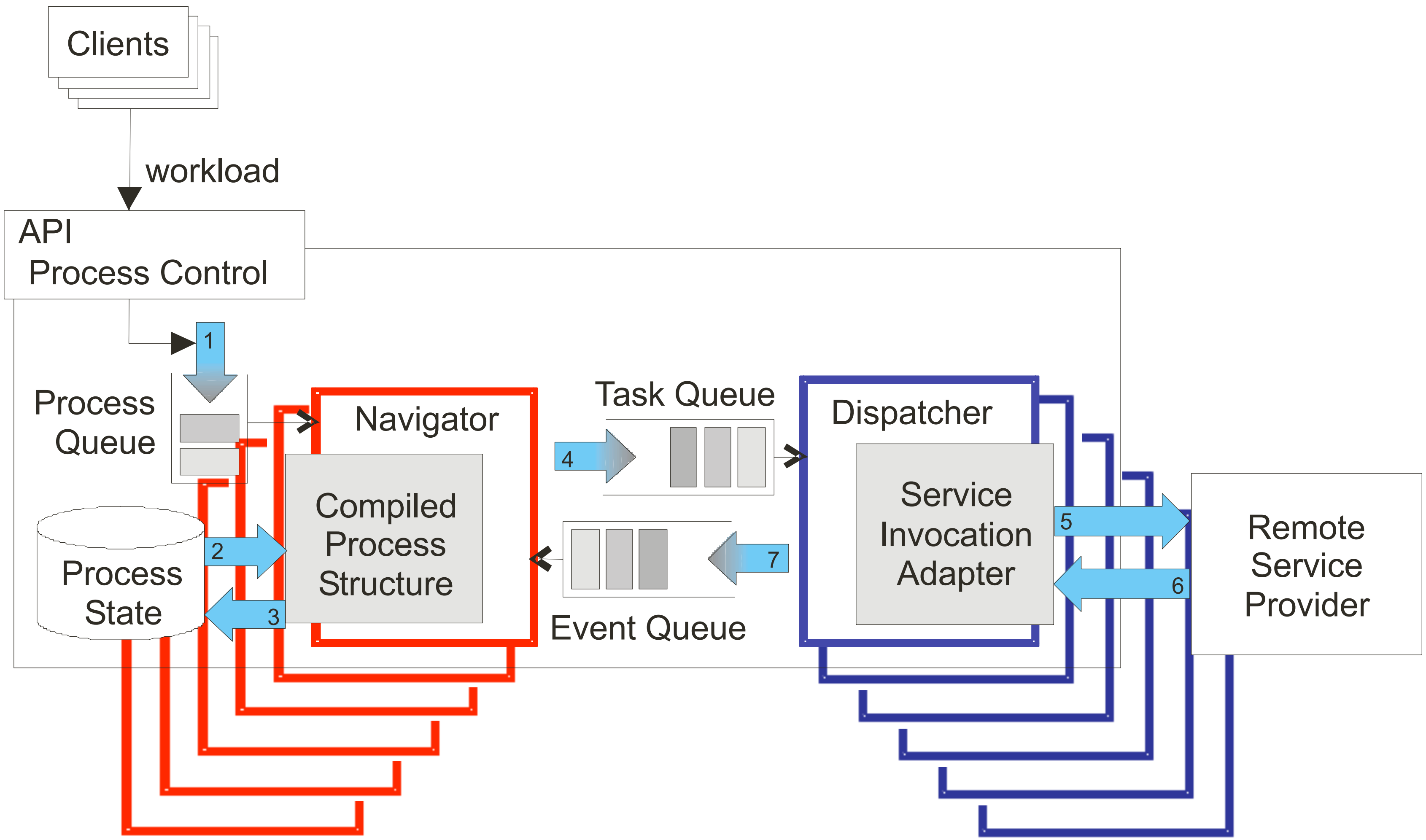
About JOpera

- **Modeling** service composition behavior
  - Graph-based **composition language** (Visual & XML)
  - Development and Debugging tools for Eclipse
  - Composition not limited to Web services
- **Execution** of the composition models
  - Efficiency (compiled to Java bytecode)
  - Dynamic (late binding, introspection)
  - Scalability (run-time is distributed on a cluster with autonomic self-configuration)
  - Extensibility (Eclipse plug-ins to provide custom service invocation adapters)

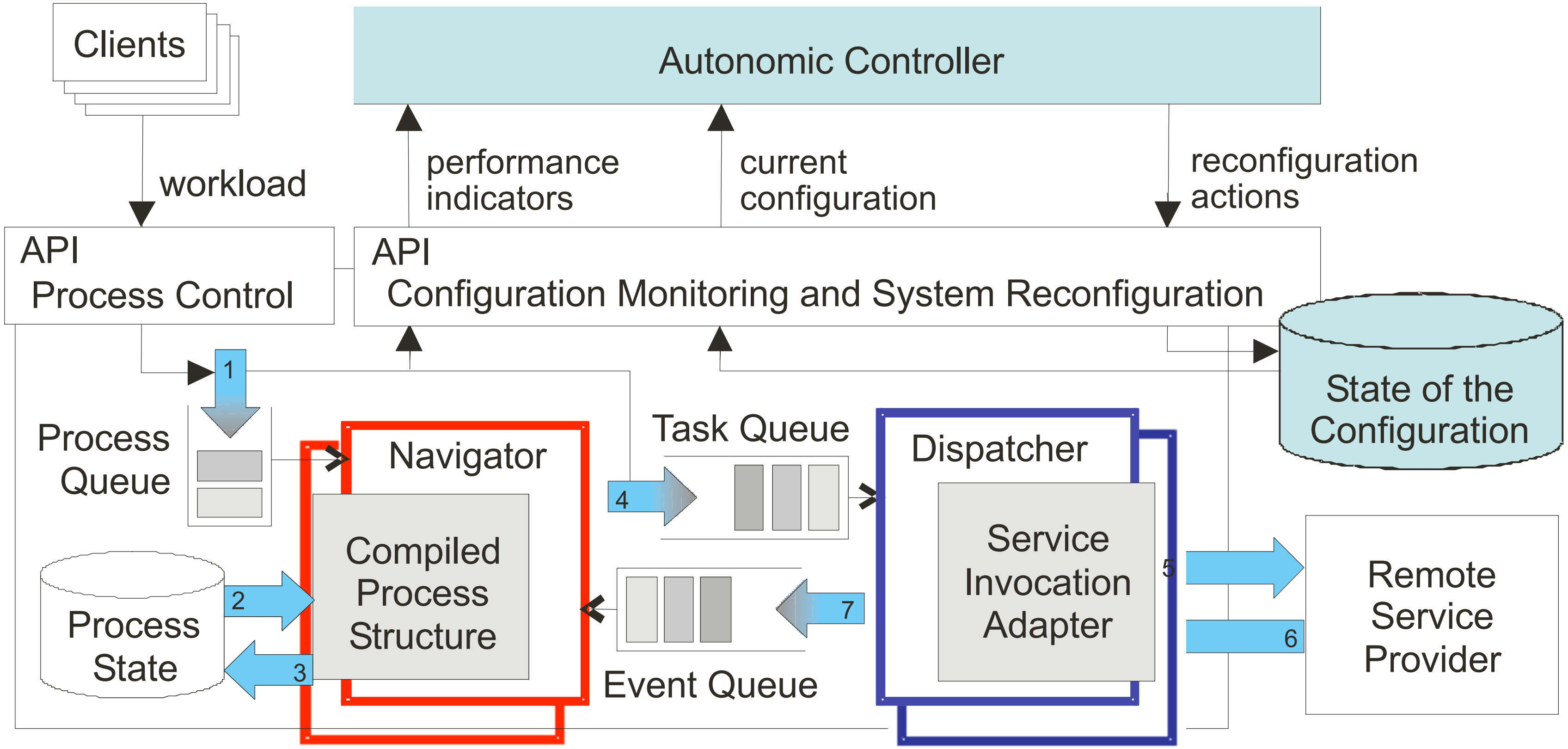
Architecture



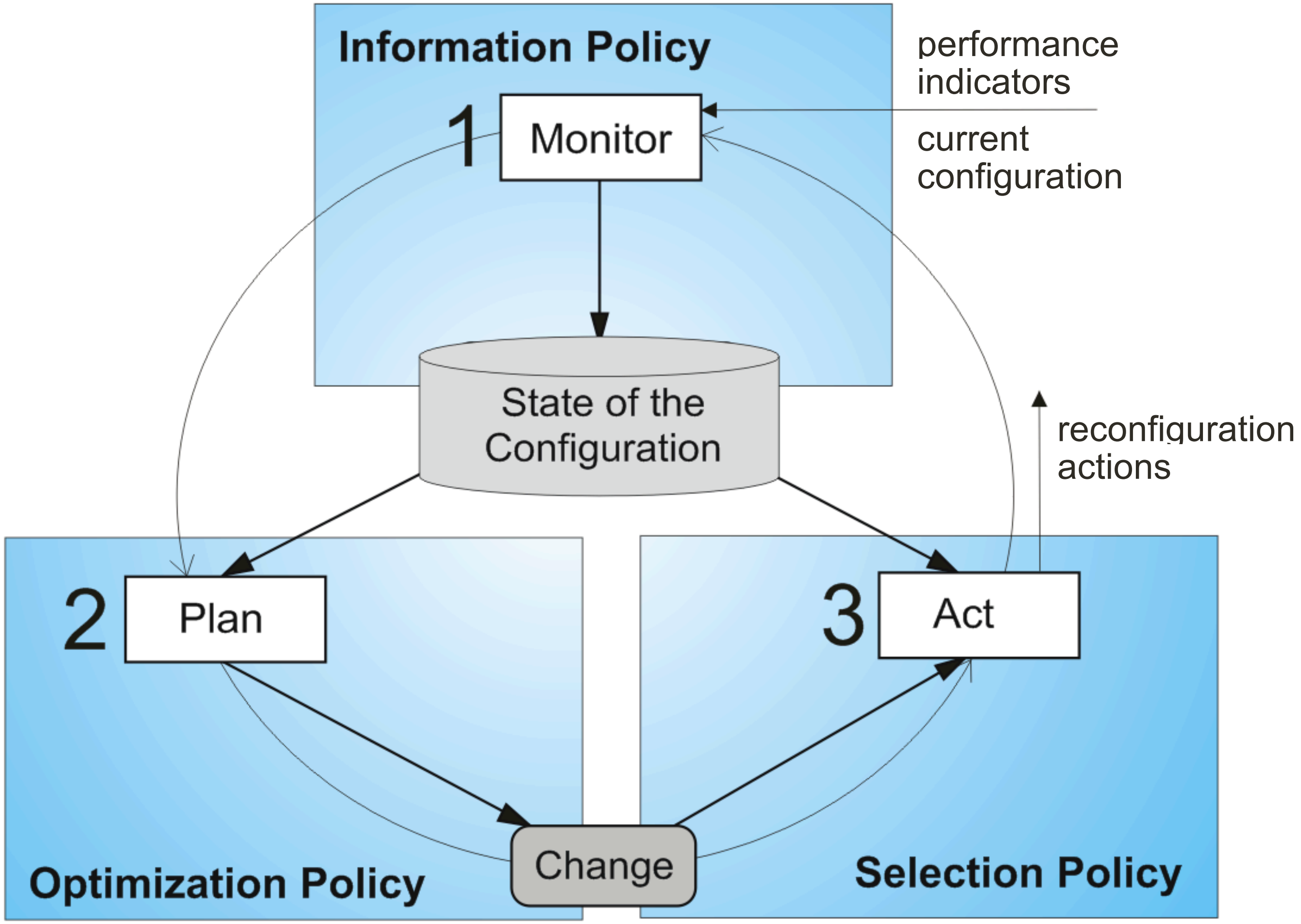
Architecture



Architecture



Autonomic Controller Algorithm



Autonomic Controller Policies

- Information Policy
  - Define which variables should be monitored

  Queue Length, Number of Navigator/Dispatcher Threads

- Optimization Policy
  - Map Monitored Variable to Reconfiguration Actions

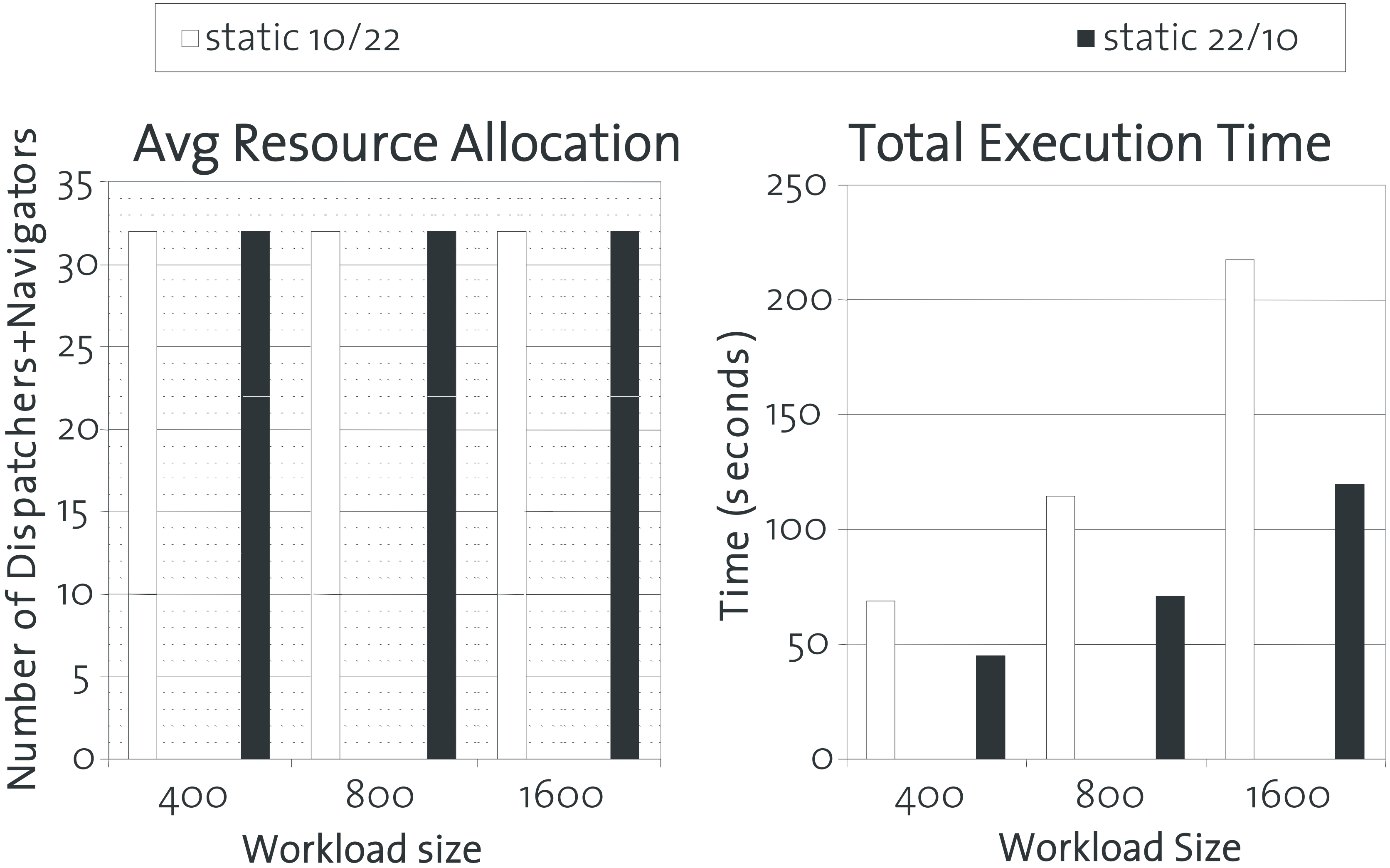
    1. Simple Threshold Policy
    2. Differential Policy
    3. Proportional Policy

- Selection Policy
  - Choose how to implement a reconfiguration plan

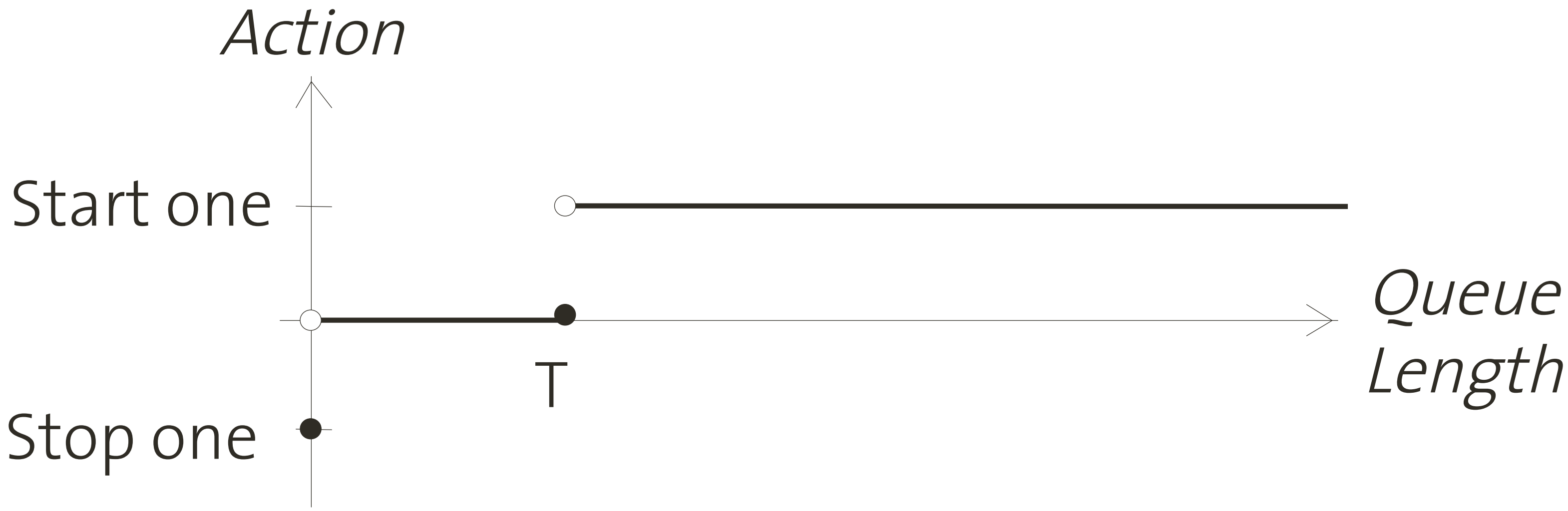
Evaluation of the Control Policies

- Workload: **Peak Response Benchmark**
  - 800 concurrent processes initiated at the same time
- Performance Indicators:
  - Total Execution Time
  - Average Resource Allocation
- 32 node cluster environment (one thread/node)
- Baseline: **Static Manual Configuration**
  - Fast: 10 Navigators, 22 Dispatchers
  - Slow: 22 Navigators, 10 Dispatchers

Baseline: Slow/Fast Static Configuration

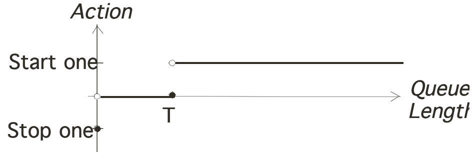


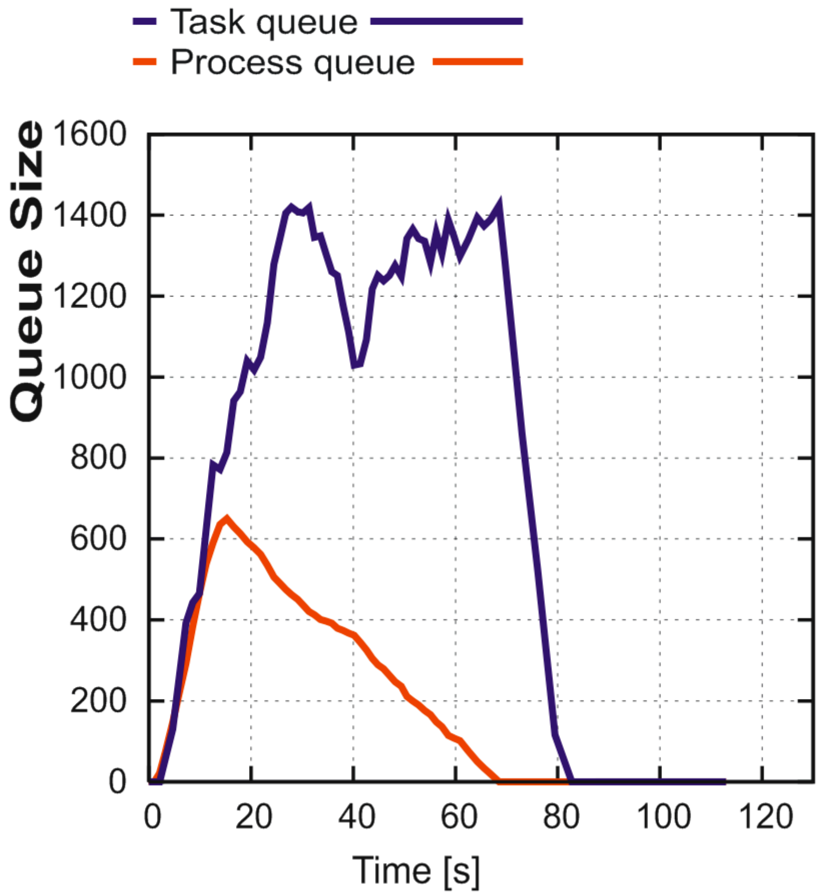
1. Simple Threshold Policy

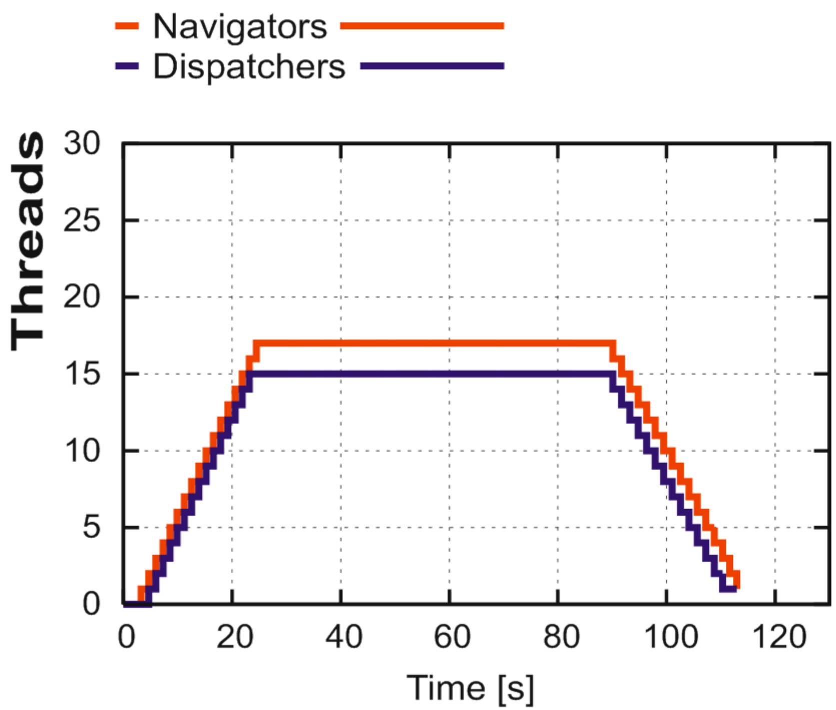


- Start one thread if Queue Length > T
- Stop one thread if Queue Length = 0

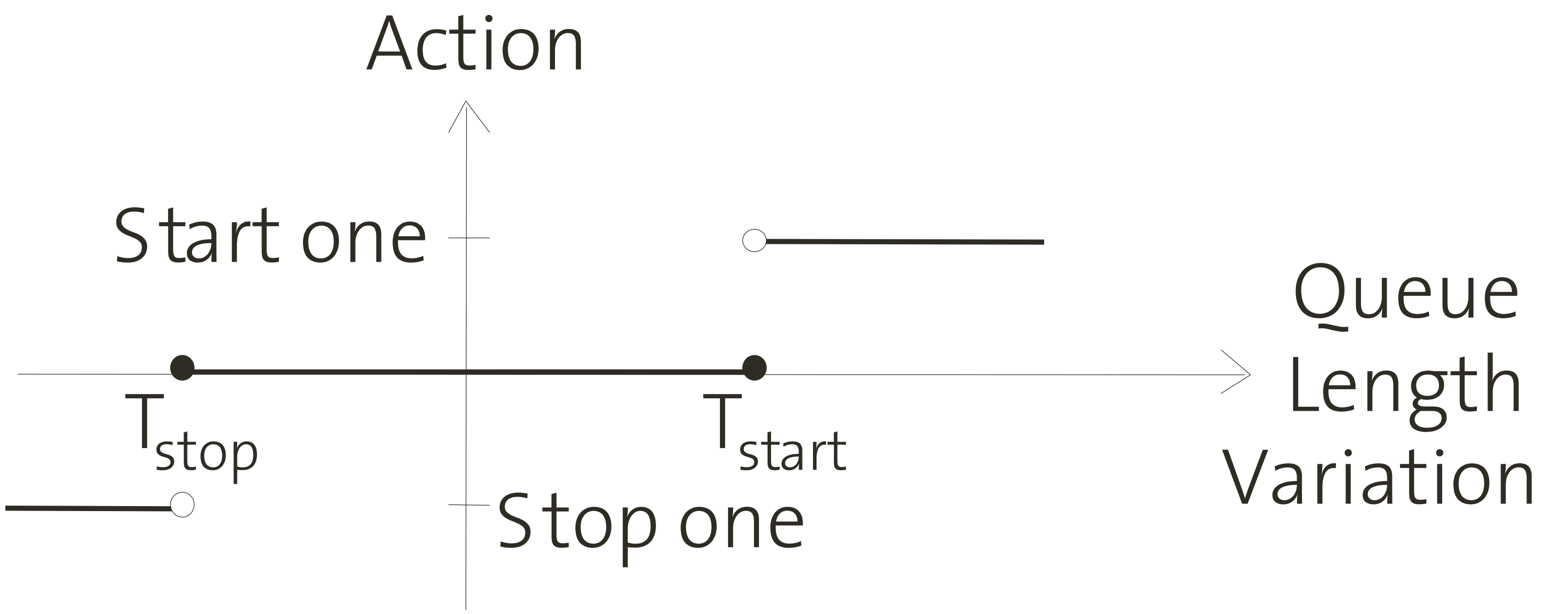
Tracing the Simple Threshold Policy





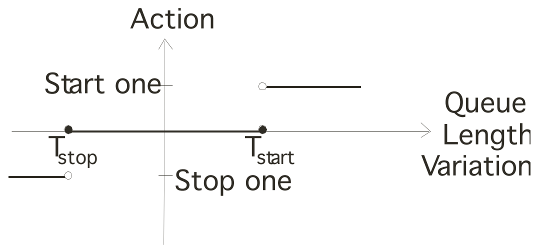


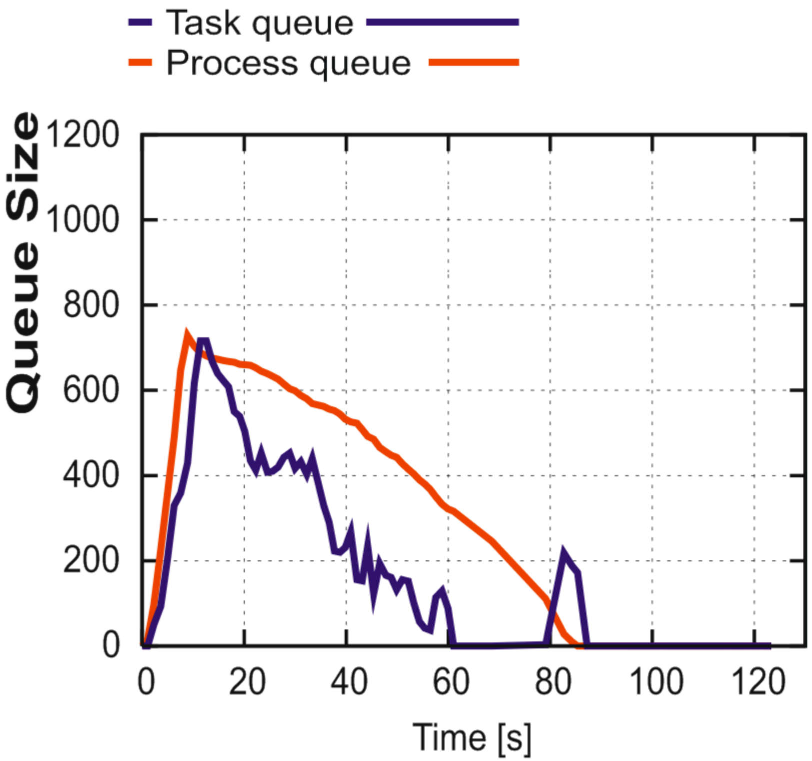
2. Differential Policy

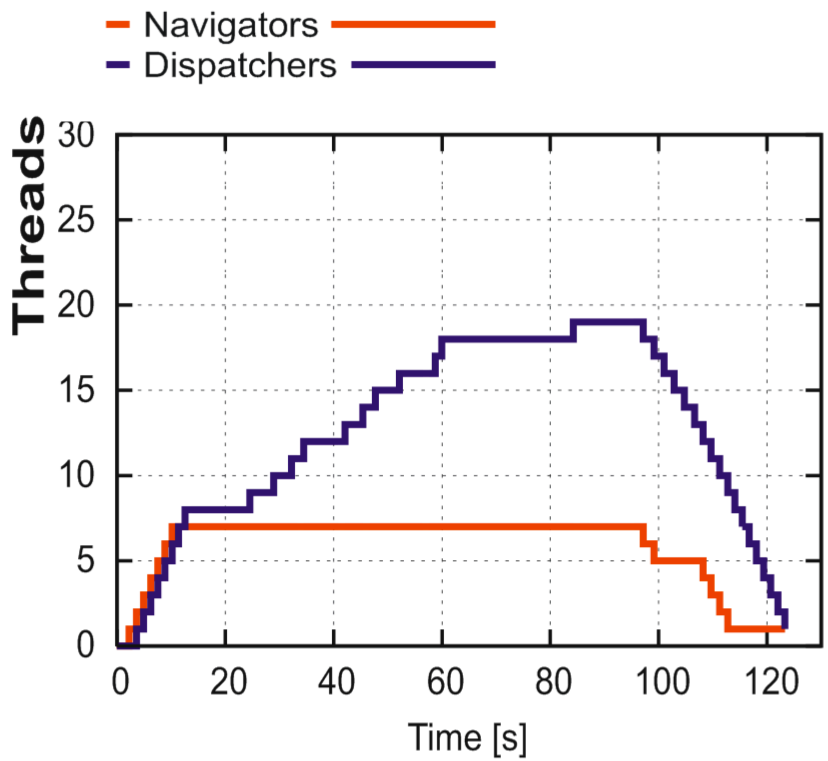


- Start one thread if Queue Length Variation > T_{start}
- Stop one thread if Queue Length Variation < T_{stop}

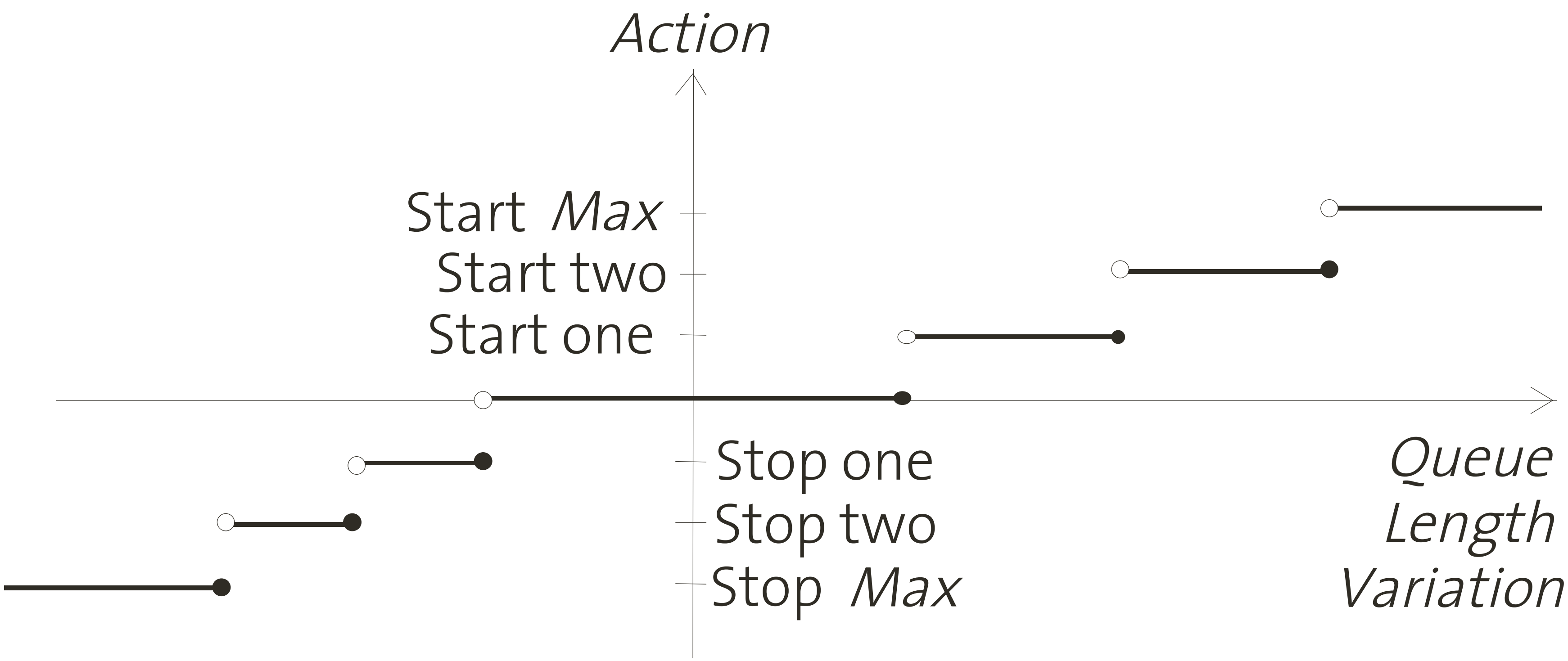
Tracing the Differential Policy





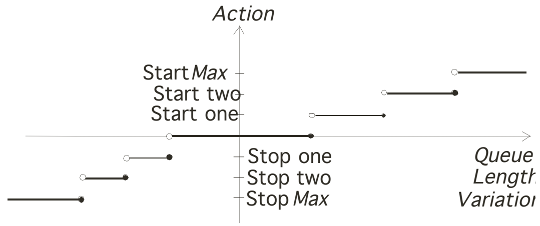


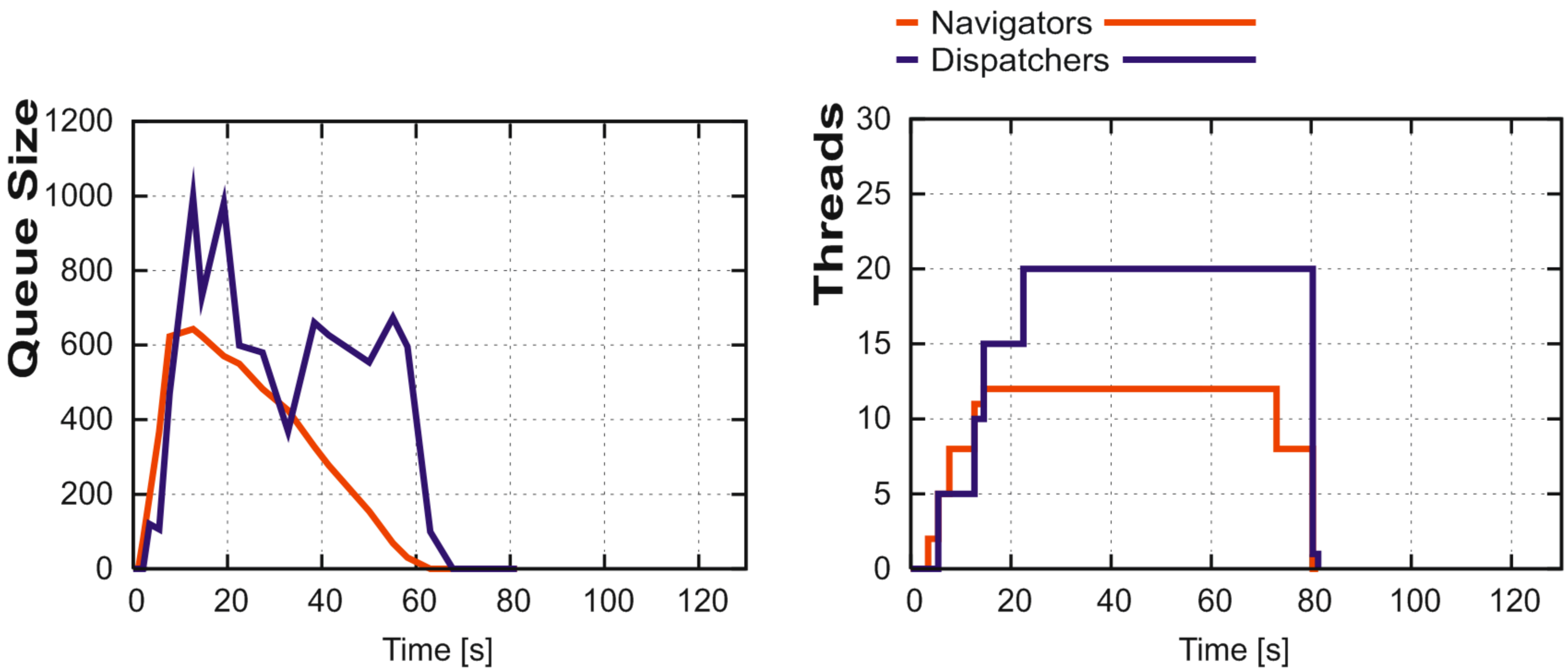
3. Proportional Policy



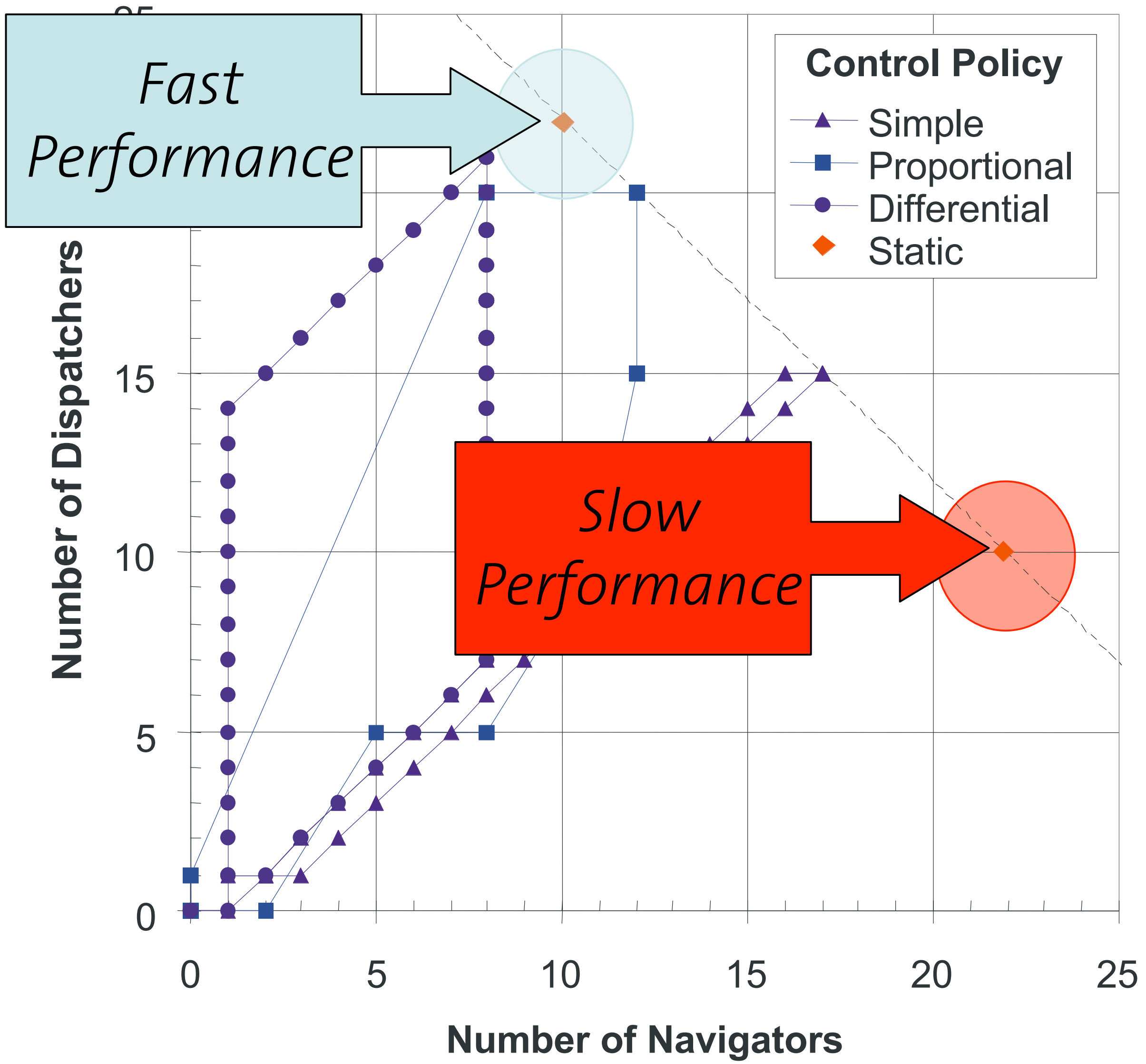
- Start/Stop *N* threads,
  proportional to the Queue Length Variation

Tracing the Proportional Policy

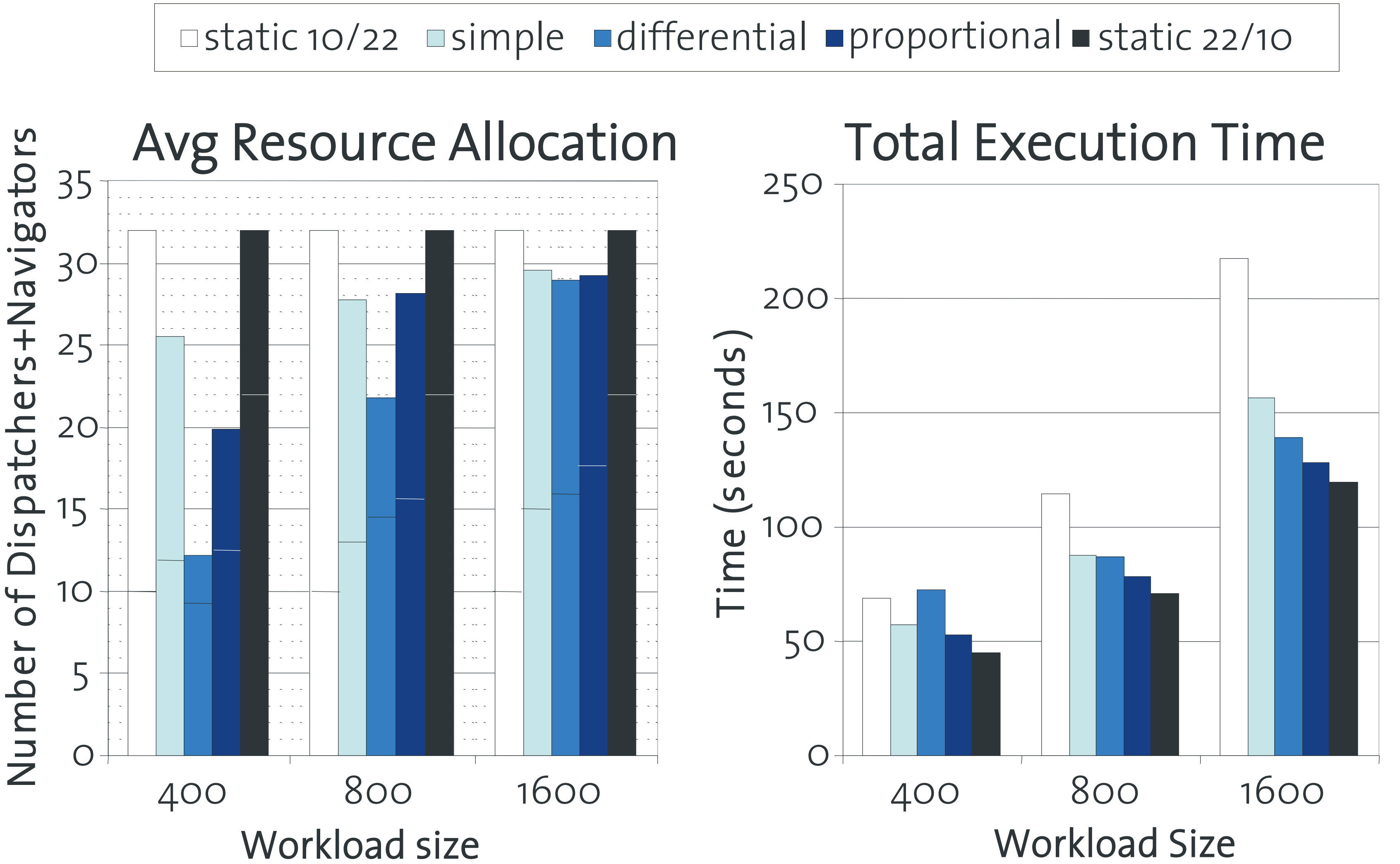




State Space Comparison of the Policies



Performance Comparison of the Policies



Conclusion

- **Manual** configuration & management of a distributed process-based Web service composition engine is difficult and expensive
- To address this problem, we have shown how to apply **autonomic computing** techniques
- Our evaluation indicates that different control policies can be used to explore the trade-off between performance vs. resource utilization

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

IKS
Information and
Communication Systems
Research Group

Autonomic Execution of Web Service Compositions

Cesare Pautasso, Thomas Heinis, Gustavo Alonso

Department of Computer Science
ETH Zurich, Switzerland

pautasso@inf.ethz.ch - **www.jopera.org**