Runtime Administration

Milad Jason Daivandy, Bernd Schuller, Bastian Demuth

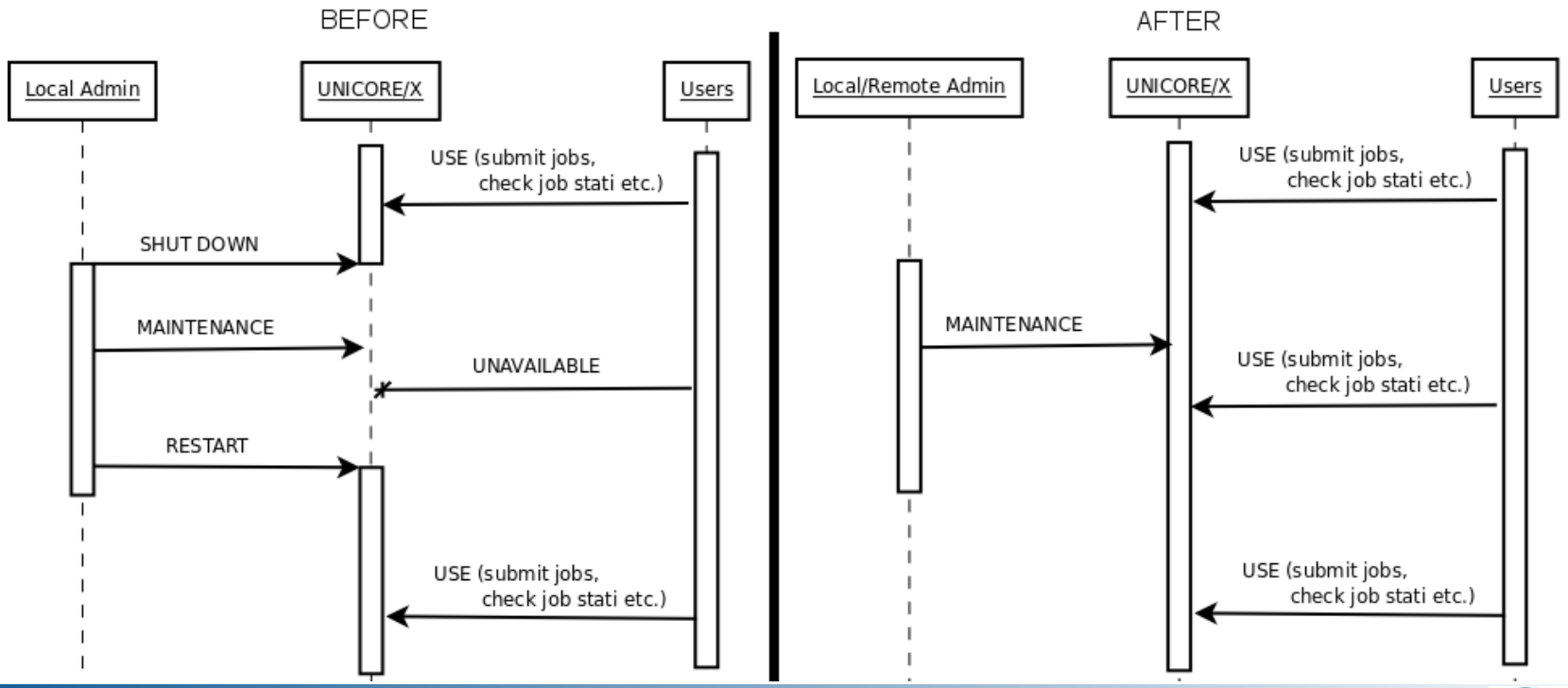
Jülich Supercomputing Centre (JSC)

Runtime Administration – Why, How?

- Why?
  - Minimize service interruption due to maintenance and management actions
- How?
  - Dynamic UNICORE service deployment
  - Reconfiguration at runtime
  - System health monitoring via metrics

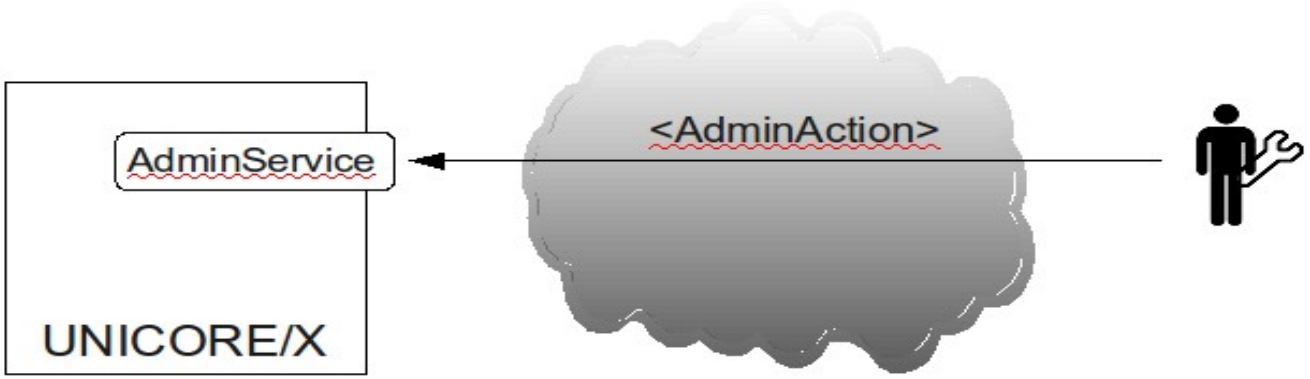
Runtime Administration – Benefits

- Reduce service interruption due to maintenance & management actions
  - Service deployment, configuration change
- Remote admin access via Web service
  - Leverages existing UNICORE security



Runtime Administration – The Big Picture

- Runtime administration capabilities exposed via Web service: *AdminService*



AdminAction

- services: list, (un)deploy
- WS-Resources: list, delete
- modify configuration
- retrieve metrics

Solution – Overview

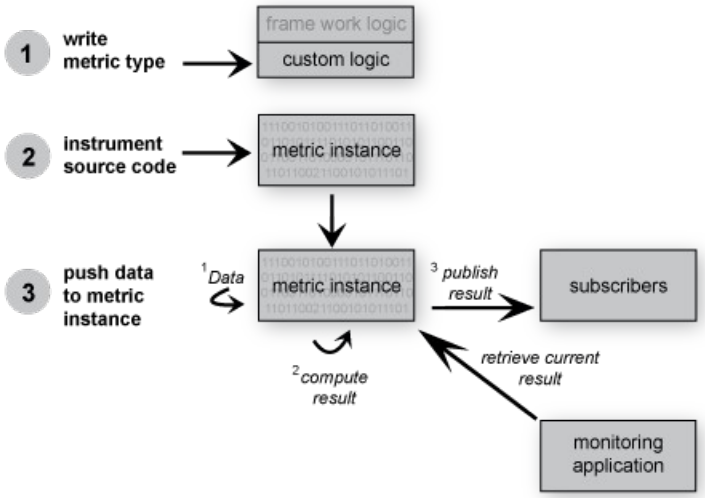
- Dynamic UNICORE service deployment
  - Provide a deployment package (JAR file + deployment descriptions)
- Reconfiguration at runtime
  - Modify remotely accessible properties
  - Optional: property value transitions trigger predefined actions
- Metrics for system health
  - Source code instrumentation of UNICORE
  - Using a lightweight Java metric framework
- Usability of the above
  - UNICORE Rich Client Plugin: *AdminDashboard*

Solution – Details 1/3

- Dynamic UNICORE service deployment
  - Two modes of deployment
    - Remotely via *AdminService*
    - Locally via *AutomaticDeployment* thread (periodically checks for new deployment packages in local directory)
- Distinction in two service types
  - Core: UNICORE Atomic Services; can't be (un)deployed at runtime
  - Plugin: can be (un)deployed at runtime
- Reconfiguration at runtime
  - Modify property value
  - Optional: add PropertyChangeListener per property (via config file)
    - Contains logic to react on value changes (essentially a state machine)

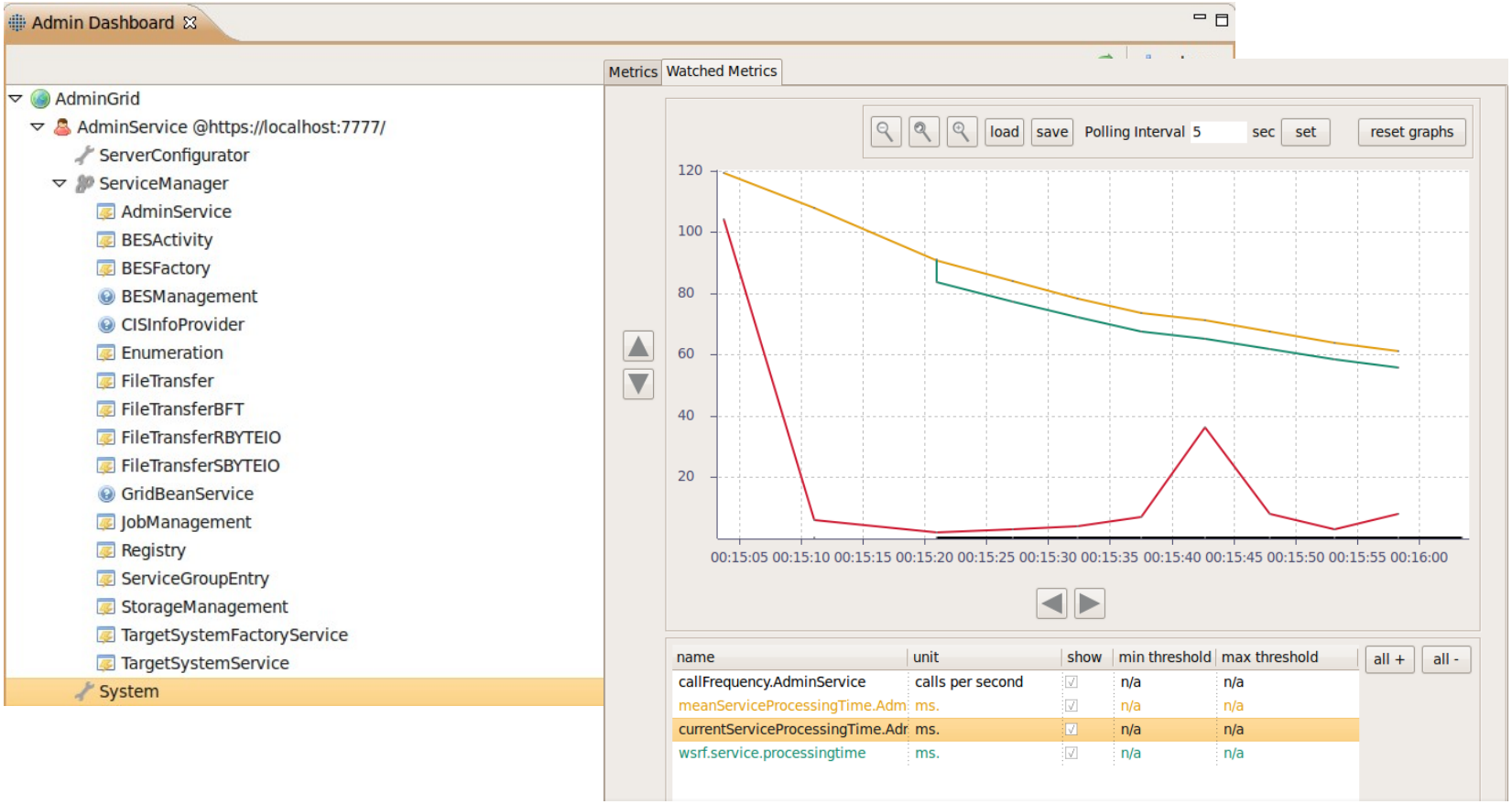
Solution – Details 2/3

- Metrics
  - Employing *metriX* (a lightweight in-house Java metric framework)
  - Write custom metric types, only focus on actual processing logic
  - Source code instrumentation
    - Deploy metric instances
    - Optional: tag with categories
  - Retrieve metric instances by ID and/or categories via *AdminService*



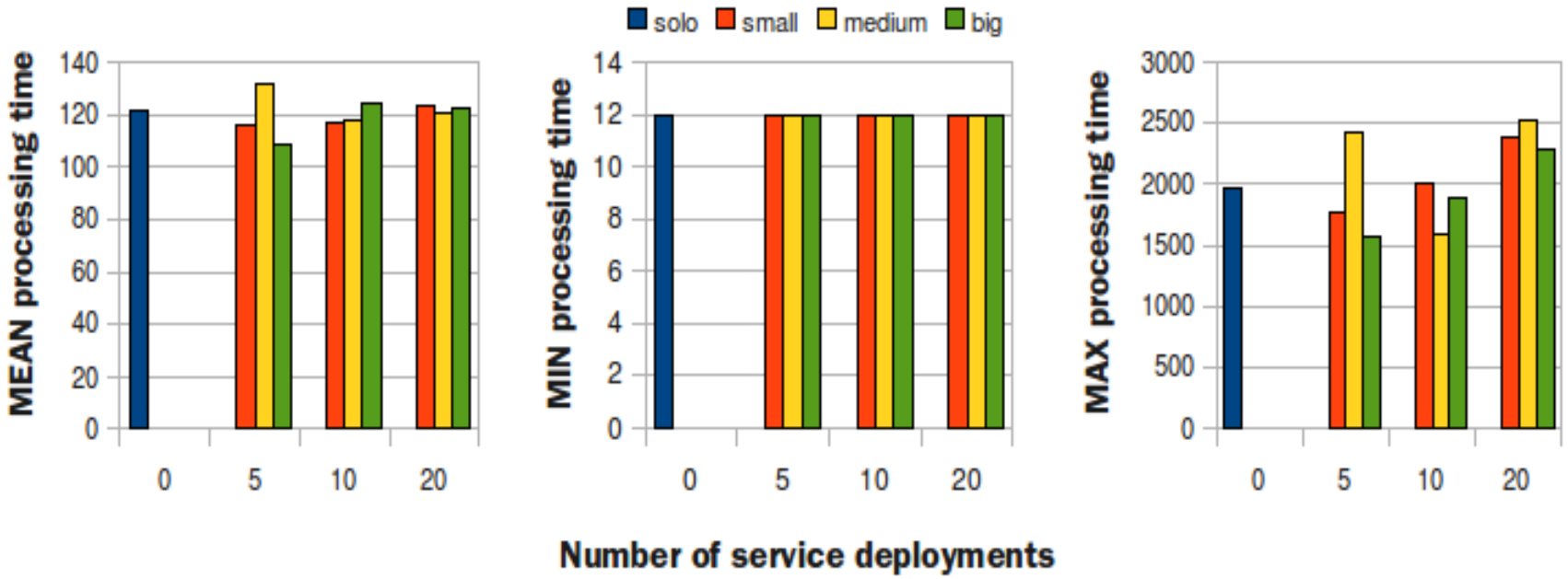
Solution – Details 3/3

- AdminDashboard
  - Powerful user interface
  - Exposes *AdminService* functionality (see demo)



Solution – Results 1/2

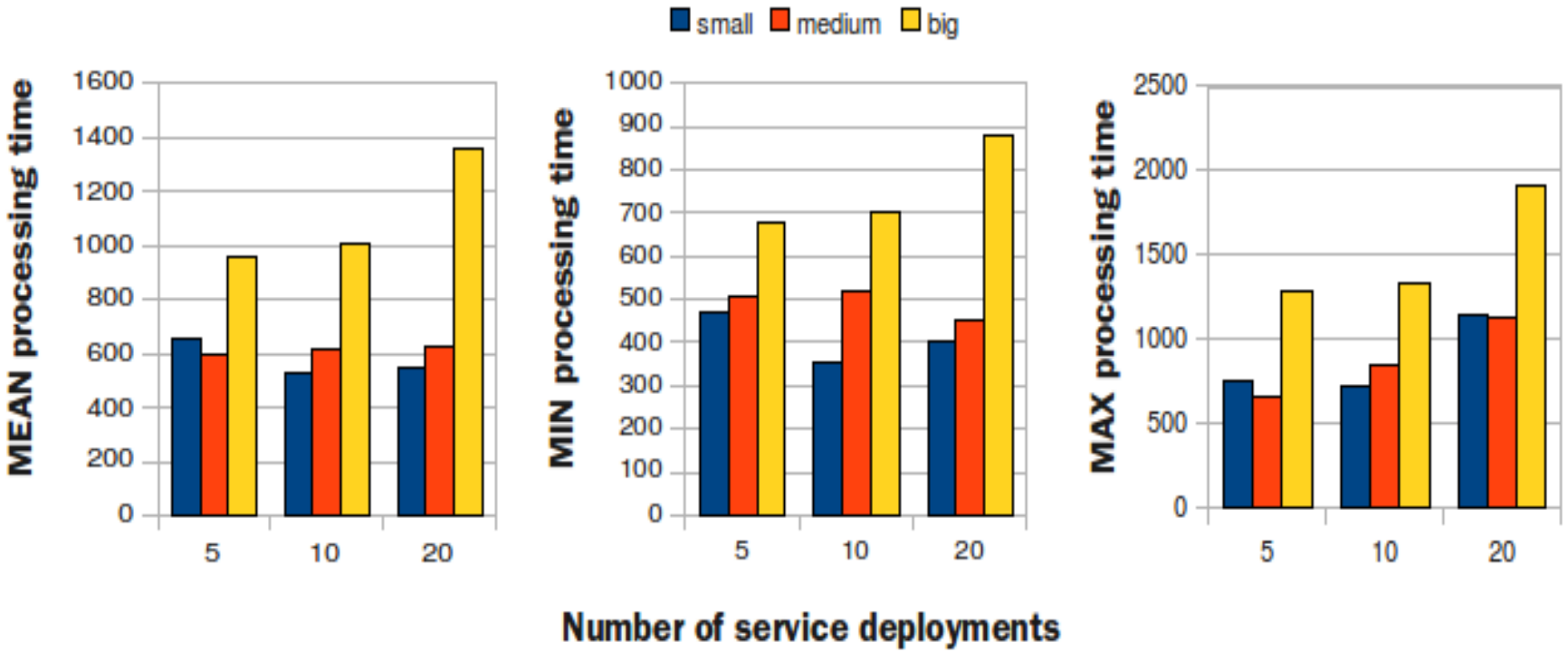
- Testing by instrumenting UNICORE with metrics
  - MEAN, MIN, MAX processing times of TargetSystemService.getResourceProperty()
- Performance impact of dynamic service deployment on UNICORE
  - Load generator stressing TSS with 10000 read requests



Solution – Results 2/2

- Testing by instrumenting UNICORE with metrics
  - MEAN, MIN, MAX processing times of AdminService.deployService()

- Scalability of dynamic service deployment on idle UNICORE



Solution – Conclusion & Future Work

- Dynamic service deployment
  - Service package attached to SOAP message of deployment request
  - Sub-optimal: more than ~ 4 MB / package can cause Heap Space Errors
- Solution
  - Decouple service package upload from deployment request
    - Use datastream-based mechanisms
  - Proposal: make dynamic deployment request three-stepped
    - 1. notify UNICORE of deployment intention
    - 2. upload service package
    - 3. deploy