RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

Exploring Software-Defined Environments for Science

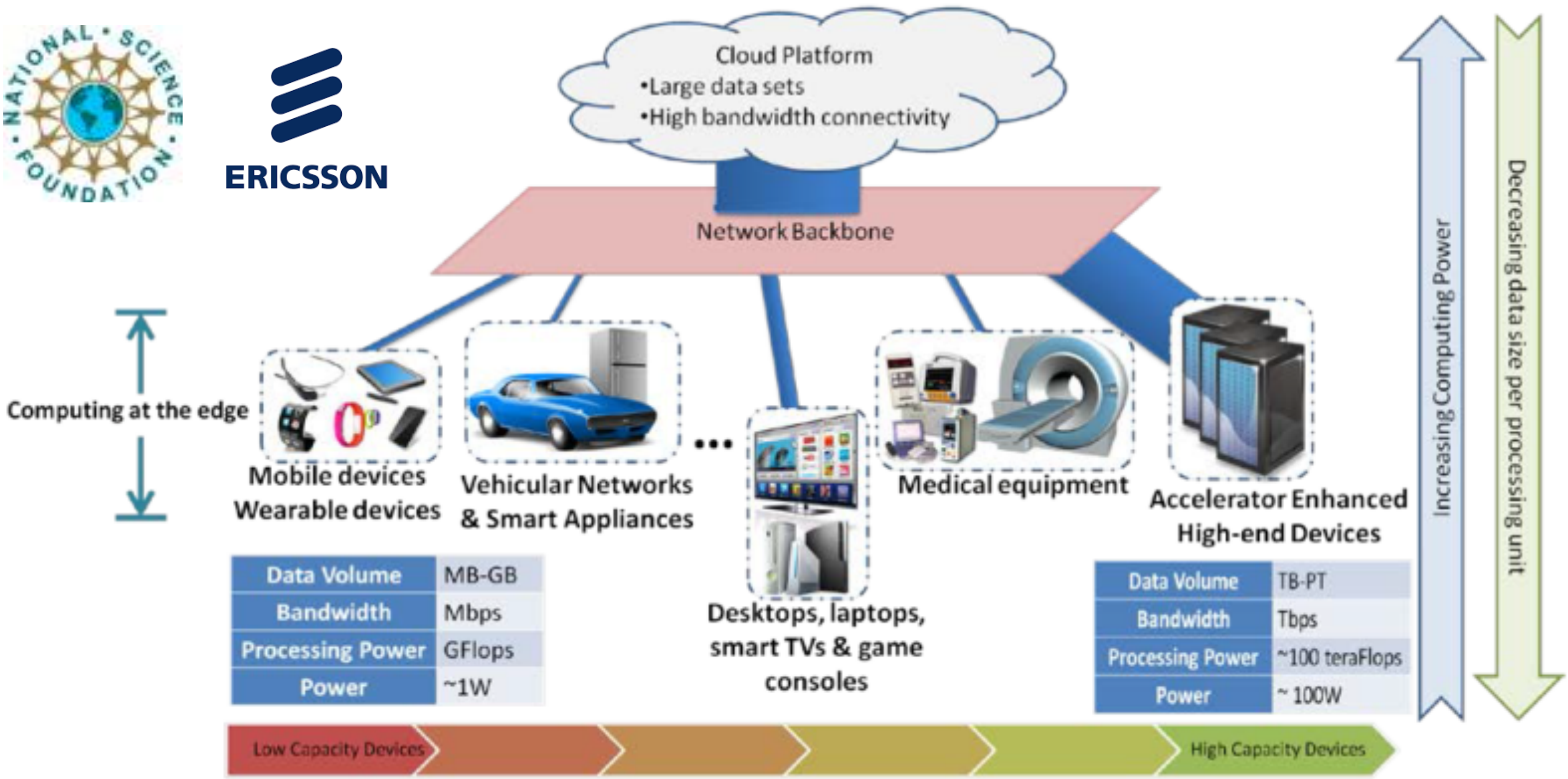
Moustafa AbdelBaky, Javier Diaz, Manish Parashar

Rutgers Discovery Informatics Institute (RDI^2)
parashar@rutgers.edu

CCDSC 2016

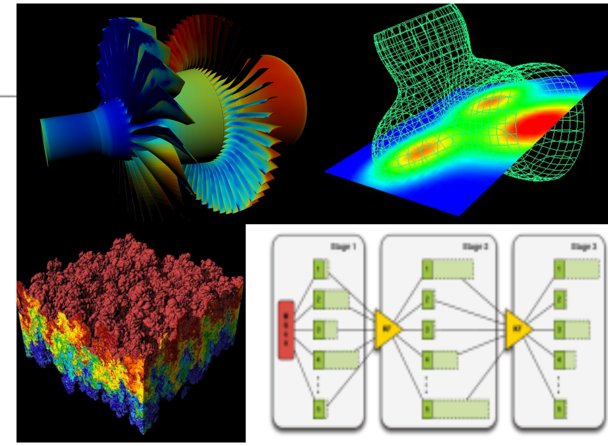
RUTGERS

Service Compositions for IoT / Emerging Data Ecosystems

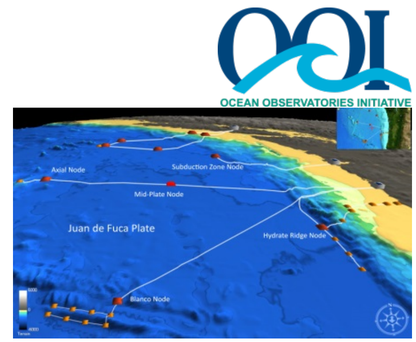


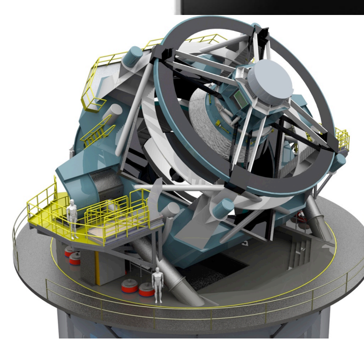
Online approximation **Computing in the continuum** **Living workflows**

Evolving workflows in science

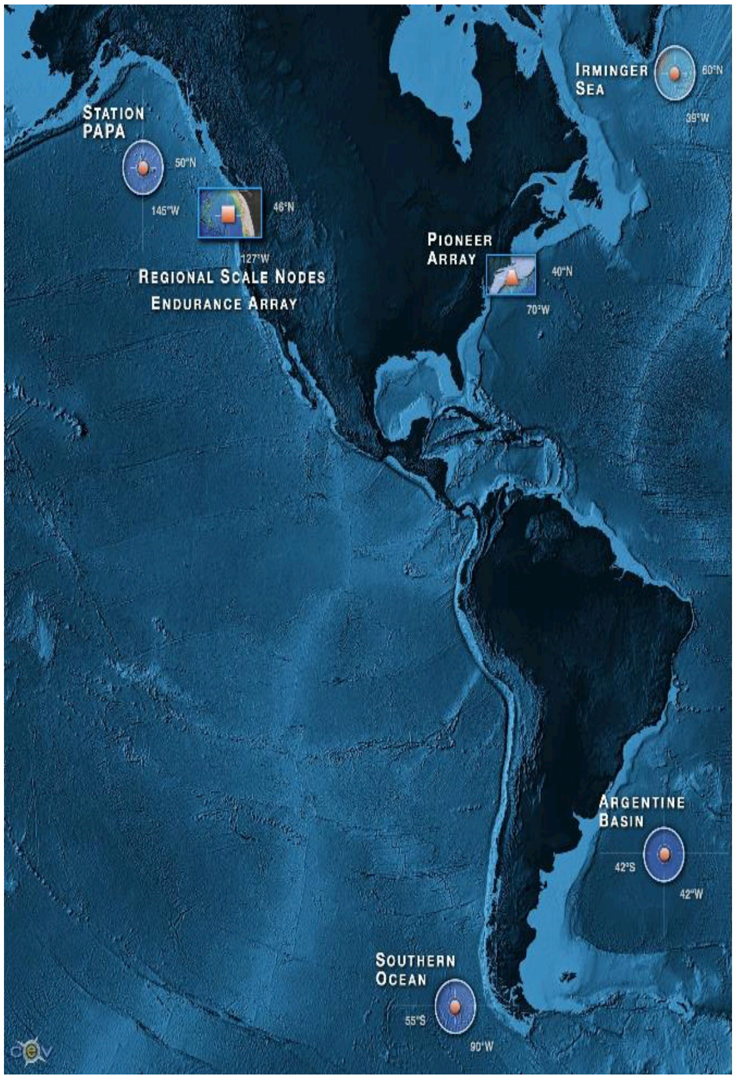


- Unprecedented amounts of data from experiments, observations, simulations, devices, etc.
  - Large size, heterogeneous in nature, and distributed across geographic locations
- Application workflows
  - Heterogeneous and dynamic
  - Dynamic demands for resources
  - Various (and changing) QoS requirements
    - Throughput, budget, time
- Use cases span climate, precision medicine, smart infrastructure, instrumented oilfields, disaster management, etc.





OOI by the Numbers



7 **Arrays**
4 Global, Pioneer, Endurance, Cabled

50 **Sites**
Moorings, Profilers, Nodes

33 **Mobile Assets**
Gliders, AUVs

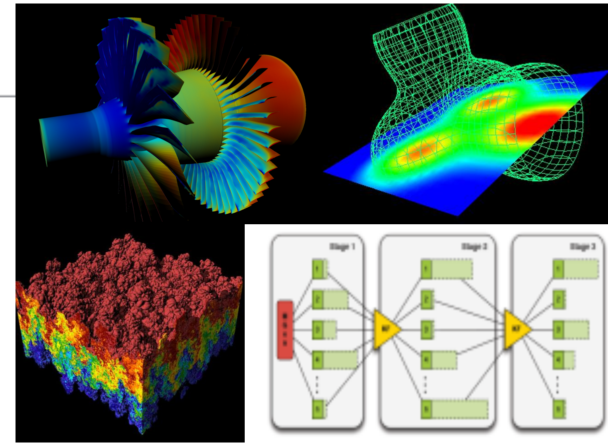
833 **Instruments**

>2500 **Science Data Products**

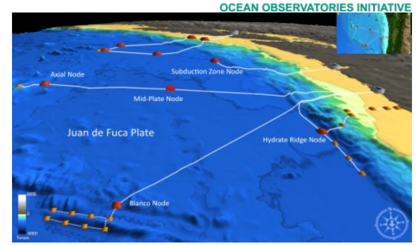
>100K **Science/Engineering Data Products**

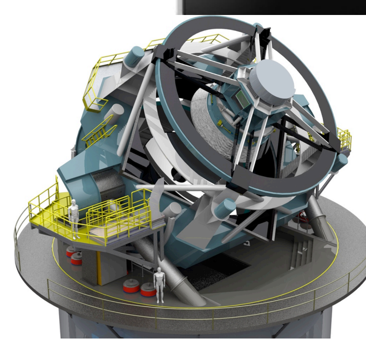
RUTGERS

Evolving workflows in science



OCEAN OBSERVATORIES INITIATIVE

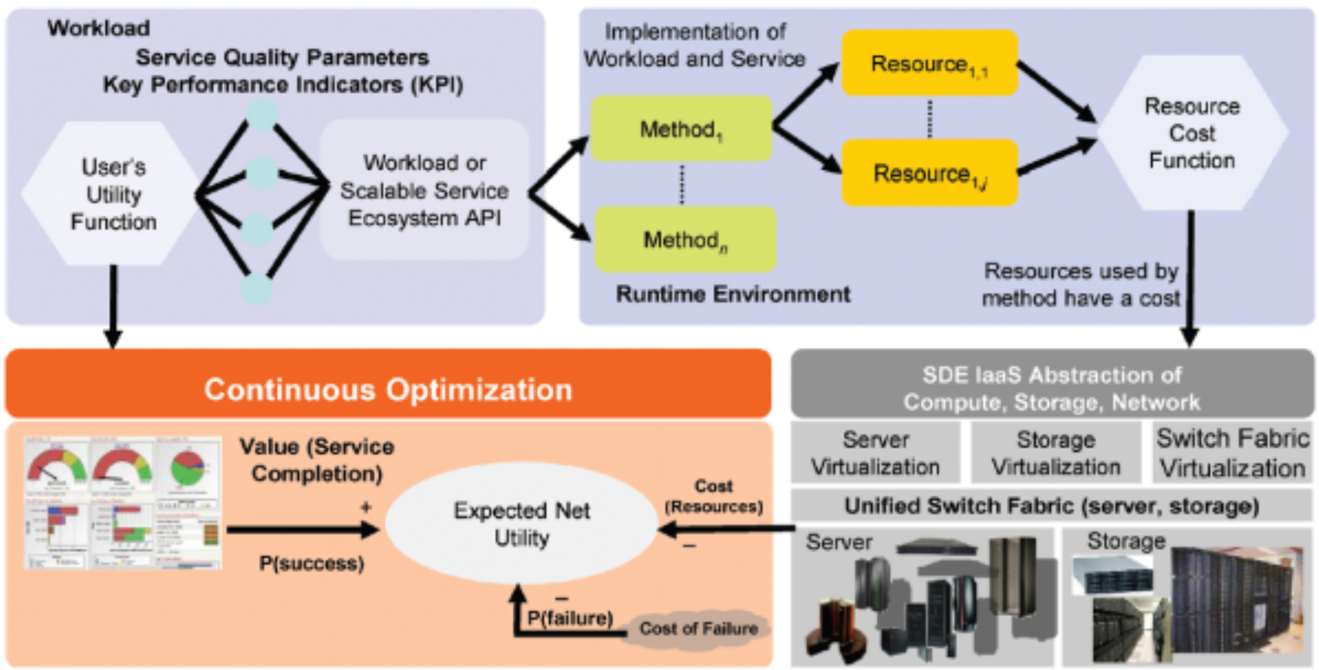




- Unprecedented amounts of data from experiments, observations, simulations, devices, etc.
  - Large size, heterogeneous in nature, and distributed across geographic locations
- Application workflows
  - Heterogeneous and dynamic
  - Dynamic demands for resources
  - Various (and changing) QoS requirements
    - Throughput, budget, time
- Use cases span climate, precision medicine, smart infrastructure, instrumented oilfields, disaster management, etc.

Software-Defined Environments (SDE)

- *Software can be used to adjust the entire infrastructure to match the workload through defined policies that control the configuration of compute, storage, and networking components, and optimize application execution*
  - **Integrated**: Built on open standards, making it quicker and easier to adapt
  - **Modular**: Cost-effective - scale up, down, or out as needed
  - **Automated**: Simplifies IT operations and service delivery



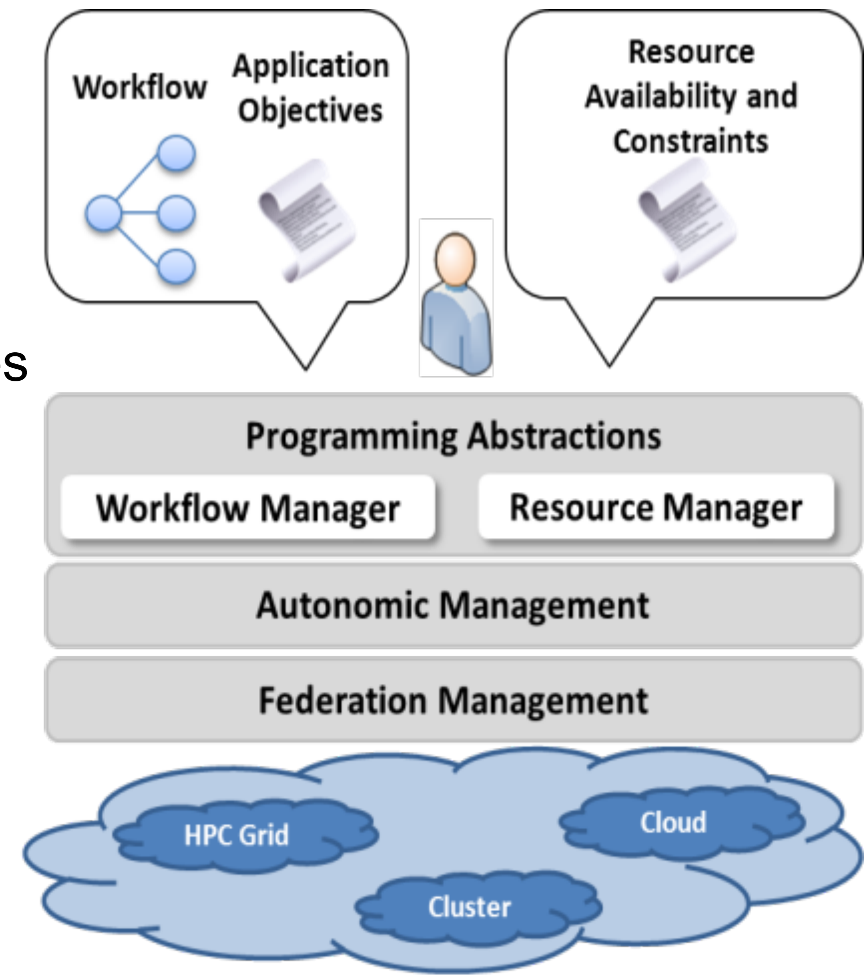
Picture source:

Li, C.; Brech, B.L.; Crowder, S.; Dias, D.M.; Franke, H.; Hogstrom, M.; Lindquist, D.; Pacifici, G.; Pappe, S.; Rajaraman, B.; Rao, J.; Ratnaparkhi, R.P.; Smith, R.A.; Williams, M.D., "Software defined environments: An introduction," in IBM Journal of Research and Development , vol.58, no.2/3, 2014

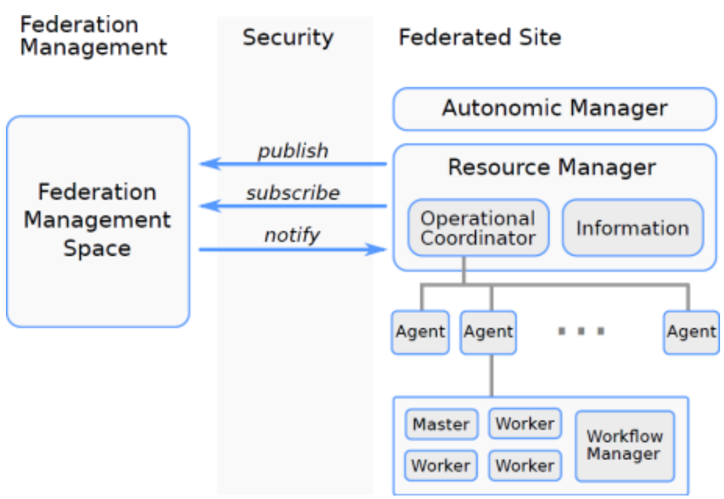
RUTGERS

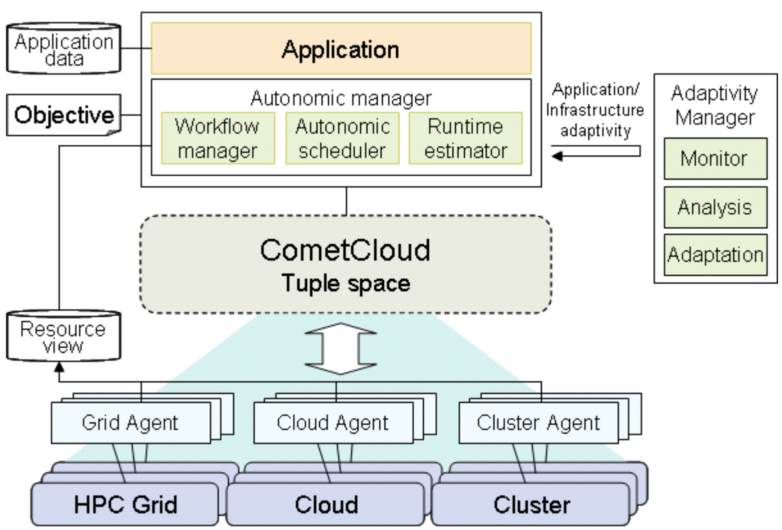
Software-Defined Environments for Science

- Combine cloud/service abstractions with concepts from software-defined environments
- Create a nimble and programmable environment that autonomously evolves in time and space, adapting to:
  - Changes in the infrastructure
  - Application requirements
- Enable efficient data processing
  - Allocate computation close to data sources
  - Process data in-situ and/or in-transit
- Independent management of applications and resources

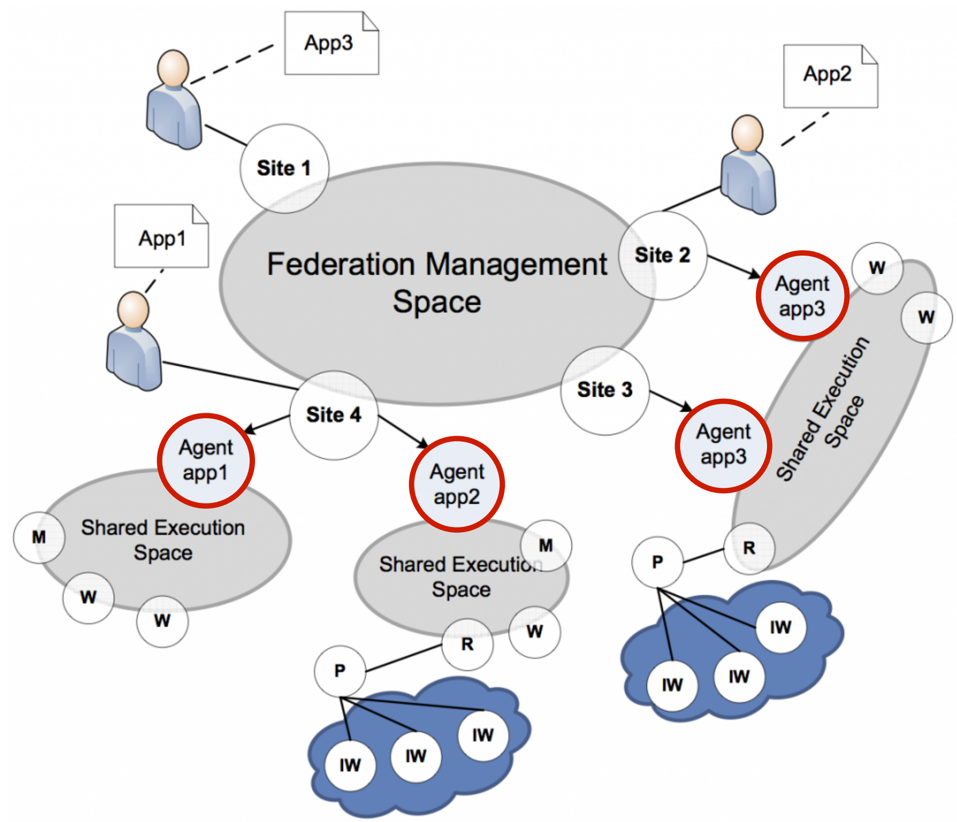


Builds on CometCloud



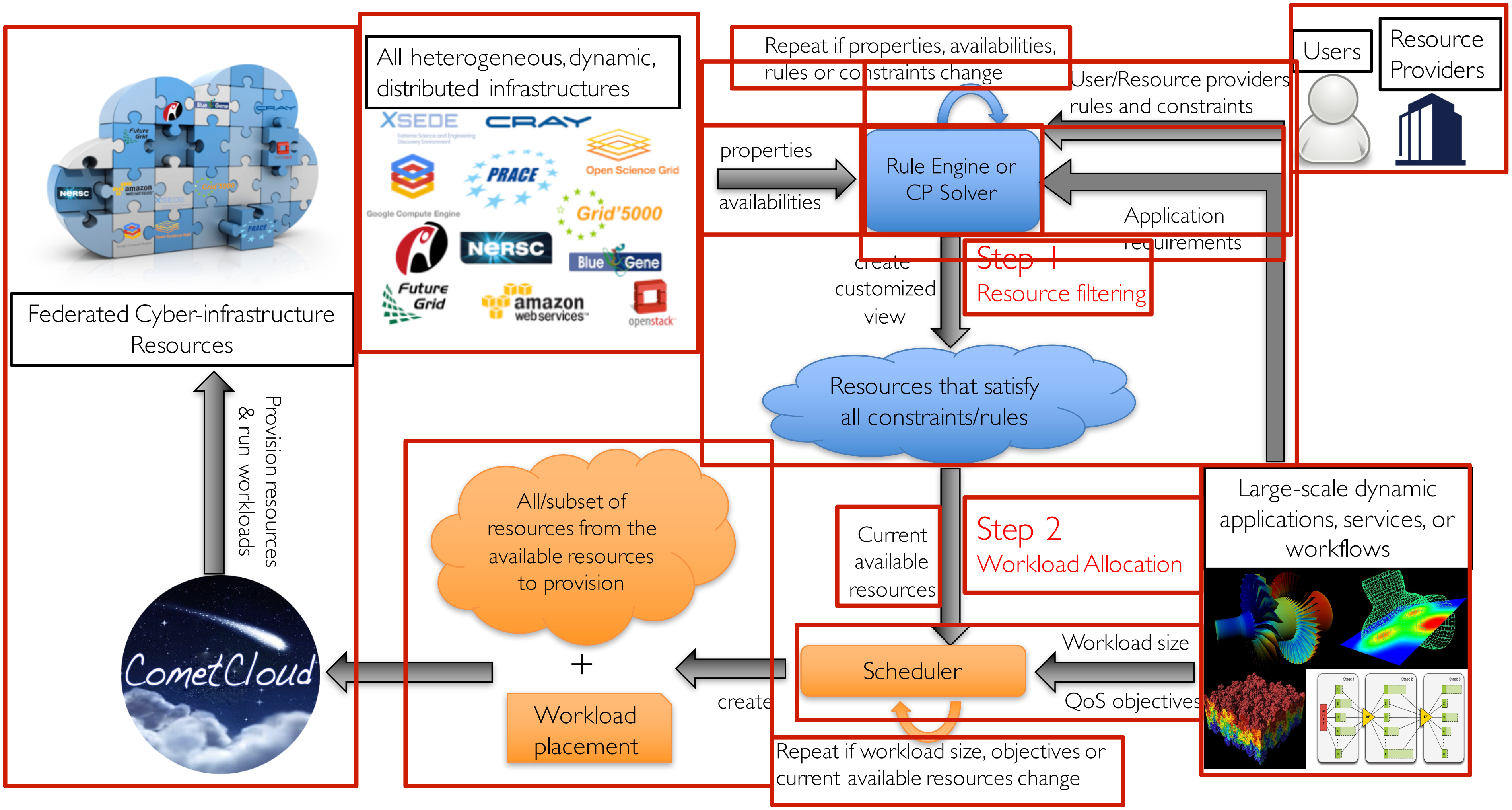


Federated (hybrid) computing infrastructure



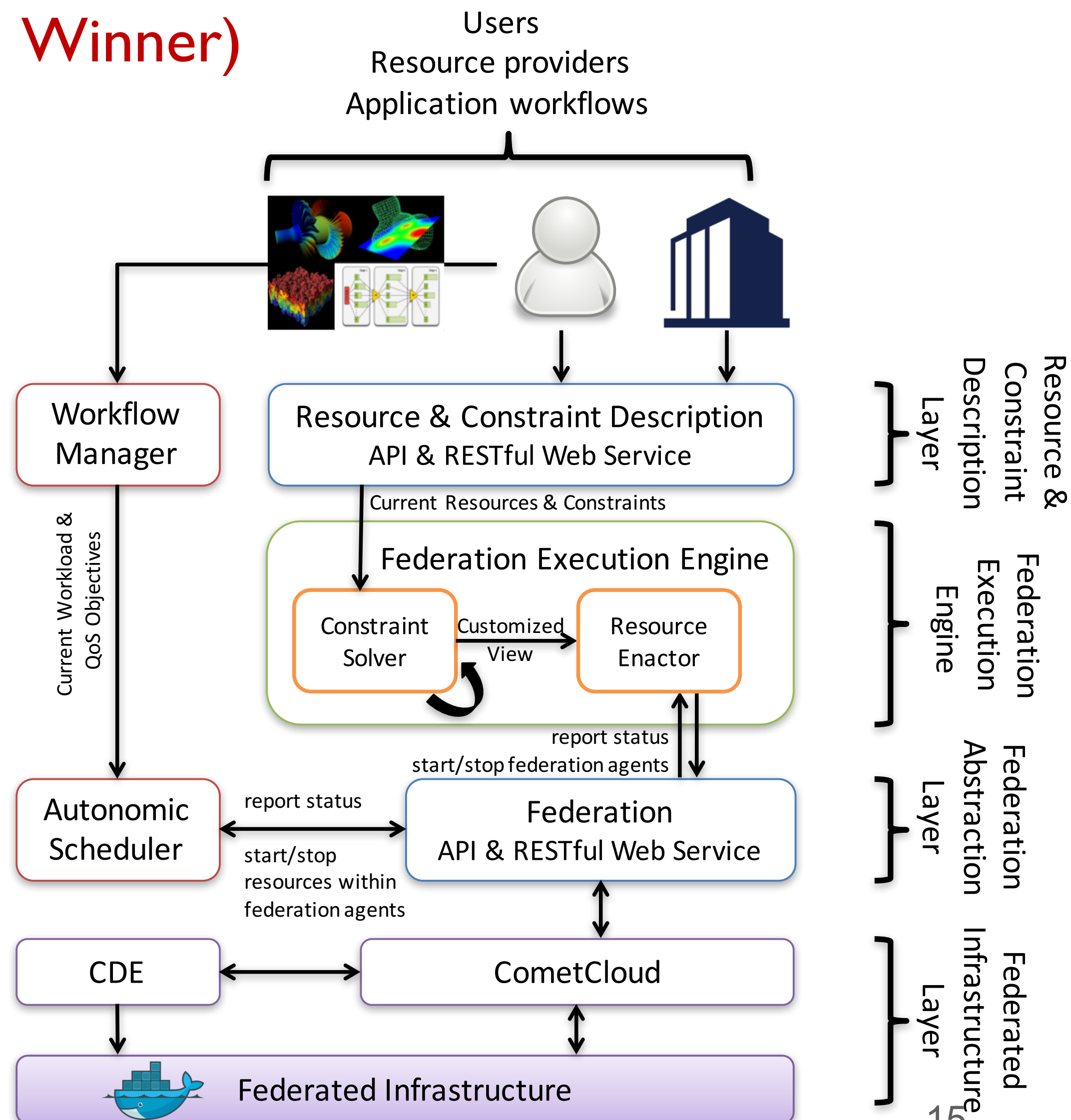
http://cometcloud.org

Overview of the Approach (Using Infra. Services)



Prototype CP-based Architecture
(with IBM, UCC Challenge'15 Winner)

- Use CP to describe the environment state
  - Specify the min. requirements for a resource class/site to be included
  - Reduce the search space for the scheduling step
- The AS decides the number/type of resources based on application QoS objectives
- The entire process is continuous to allow for dynamic reconfiguration
- Leverages Docker containers



Constraints Formulation

$$x_{ij} = \begin{cases} 1 & \text{if } i^{th} \text{ site's } j^{th} \text{ resource class} \\ & \text{satisfies all the constraints} \\ 0 & \text{otherwise} \end{cases}$$

where $i = \{1, 2, \ldots, n\}$ and $j = \{1, 2, \ldots, m_i\}$

for $i = \{1, 2, \ldots, n\}$ and $j = \{1, 2, \ldots, m_i\}$

$$\text{maximize} \sum_{i=1,j=1}^{n,m_i} x_{ij} \text{ subject to}$$

$$x_{ij} \leq av_{ij} \; \forall j \; \forall i$$

$$x_{ij} \cdot CP \leq cp_{ij} \; \forall j \; \forall i$$

$$x_{ij} \cdot AL \leq al_{ij} \; \forall j \; \forall i$$

$$x_{ij} \cdot PF \leq pf_{ij} \; \forall j \; \forall i$$

$$x_{ij} \cdot u_{ij} \leq U \; \forall j \; \forall i$$

$$x_{ij} \cdot c_{ij} \leq C \; \forall j \; \forall i$$

$$x_{ij} \cdot pw_{ij} \leq PW \; \forall j \; \forall i$$

$$x_{ij} \cdot o_{ij} \leq O \; \forall j \; \forall i$$

$$x_{ij} \leq (1 - sc_{ij}) \cdot (1 - SC) + (SC \cdot sc_{ij}) \; \forall j \; \forall i$$

$$x_{ij} \leq (1 - ao_{ij}) \cdot (1 - AO) + (AO \cdot ao_{ij}) \; \forall j \; \forall i$$

$$x_{ij} \in \{0, 1\}$$

Property	**Description**
Availability (av)	Whether a resource class is operational
Capacity (cp)	Number of instances (e.g. nodes, VMs) in a resource class
Allocation (al)	Number of compute hours available for a shared resource class
Performance (pf)	Average performance of an instance of a resource class
Utilization (u)	Load of a resource class as a percentage of available capacity (0-100%)
Cost (c)	Price per hour for an instance of a resource class. We assume the cost per instance includes both CPU and memory costs
Power (pw)	Power consumption of a resource class
Overhead (o)	Time required to allocate an instance of a resource class
Security (sc)	Whether a resource class is secure or not
Always-on (ao)	Whether a resource class is provisioned on demand or always on

Preliminary Evaluation of the CP-based Approach

- Run across 5 different clouds in 8 different regions using 15 different types of resource classes, 110 VMs
- Deployed up to 7000 containers across the federation
- Varying workloads
- Varying resource availabilities & constraints

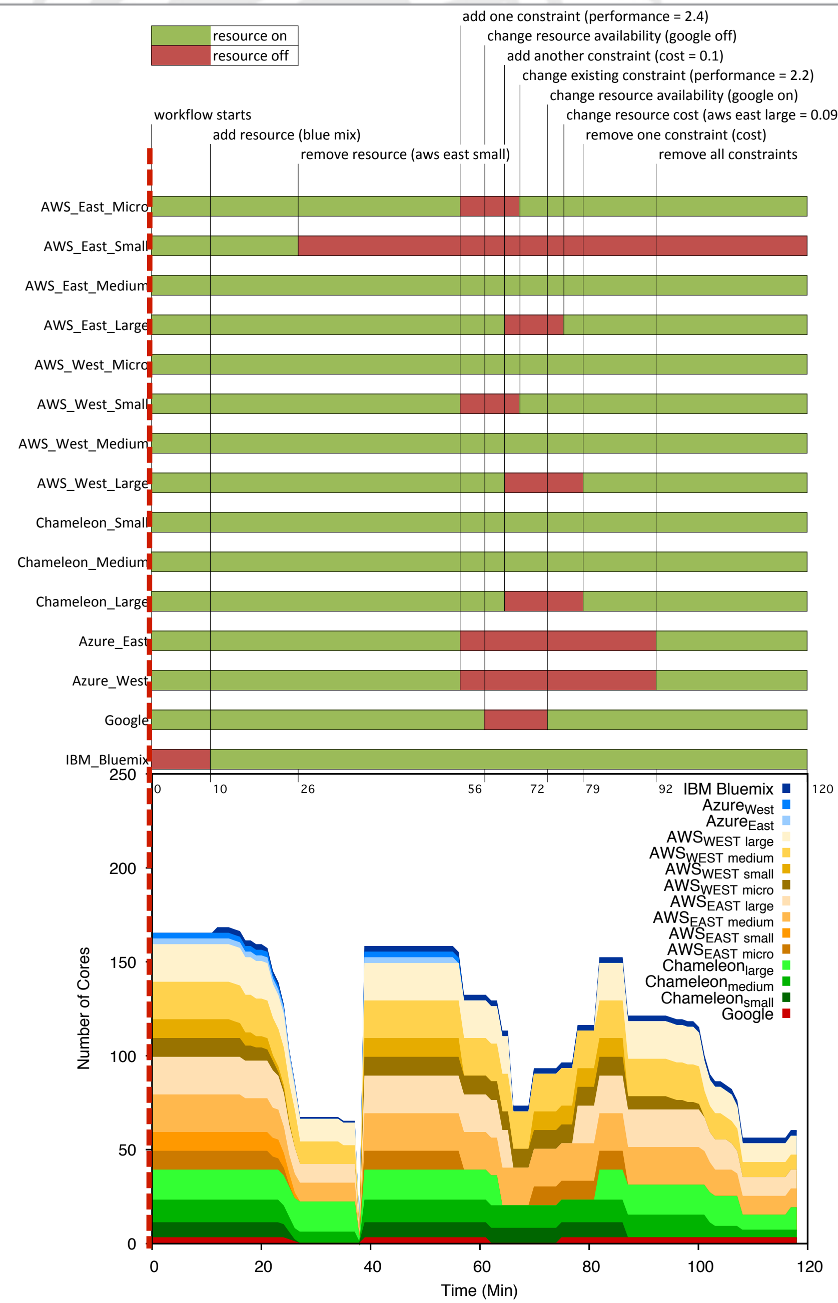
Site Name & VM Type	# Cores	Max. VMs†	Speedup	Cost ⇑
AWS east t2.micro	1	10	2.39	0.013
AWS east t2.small	1	10	2.39	0.026
AWS east t2.medium	2	10	3.35	0.052
AWS east t2.large	2	10	3.47	0.104
AWS west t2.micro	1	10	2.52	0.013
AWS west t2.small	1	10	2.33	0.026
AWS west t2.medium	2	10	3.45	0.052
AWS west t2.large	2	10	3.47	0.104
Chameleon m1.small	1	8	2.49	0.026
Chameleon m1.medium	2	6	3.99	0.052
Chameleon m1.large	4	4	5.87	0.209
Azure east Standard-A1	1	3	1.00	0.044
Azure west Standard-D1	1	3	1.70	0.077
Google east n1-standard-1	1	3	2.40	0.05
IBM Bluemix	N/A	3∗	N/A	0.028
Dell cluster	8	12	N/A	N/A

Note: The # of containers per instance = # of cores per instance.
∗ Max number of containers for Bluemix.
† – Maximum number of available VMs per type.
⇑ – Real cost ($) per hour for all cloud providers except Chameleon, which was estimated base on AWS pricing.

Triggering Event

Details

workload size



Time (min)

Available Resource Classes

Selected Resource Classes

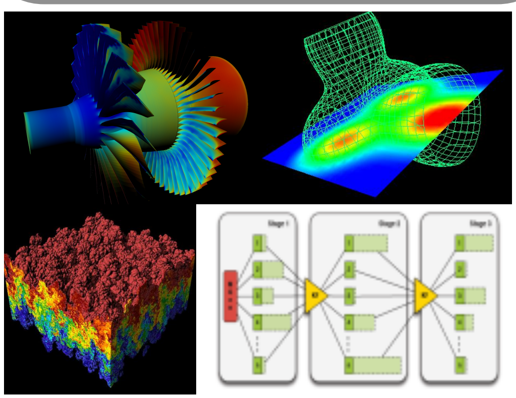
Current Total Number of Cores

Summary

Applications & Workflows

- Workflow definition
- QoS Objectives (deadline, budget)
- App requirements (type of resources, memory, I/O rate)
- Defined in terms of science (e.g., precision, resolution)

- varies at runtime -



Autonomic Management

- Identify utility of the composition
- Negotiate with application
- Ensure applications objectives and constraints are always met
- Adapt and reconfigure resources on the fly

User/Resource Provider

Define service composition programmatically using rules and/or constraints

- Availability
- Capacity & Capability
- Cost
- Location
- Access policy

- varies at runtime -

Synthesize a space-time ACI

Exposed as a cloud to application/workflow

Elastic Cyberinfrastructure

Next steps

- QoS Modeling & Quantification
  - Quantify the composition of services and model the collective performance and behavior at any given time
  - Create models to translate resource/service capabilities and availabilities into application-level utilities (e.g., throughput, performance, etc.)
- Science as a Service Platform / Application Malleability
  - Allow information-driven applications to detect and adapt to changes in the execution environment
  - Initiate a bidirectional negotiation between the workflow management framework and the underlying software-defined service composition

RUTGERS

The CometCloud Team

- Ph.D. Students
  - Moustafa AbdelBaky, Dept. of Electrical & Computer Engr.
  - Mengsong Zou, Dept. of Computer Science
  - Ali Reza Zamani, Dept. of Computer Science

- Faculty
  - Javier Diaz-Montes, Ph.D. Rutgers Discovery Informatics Institute (RDI^2)
  - Esma Yildirim, Ph.D. Rutgers Discovery Informatics Institute (RDI^2)
  - Manish Parashar, Ph.D. Dept. of Computer Science and RDI

Omer Rana, Ion Petri, and many other collaborators….

CometCloud: http://cometcloud.org

Thank You!

PS: We are hiring

- Postdocs/Research Associates
- Research programmers