Everest: A Cloud Platform for Computational Web Services

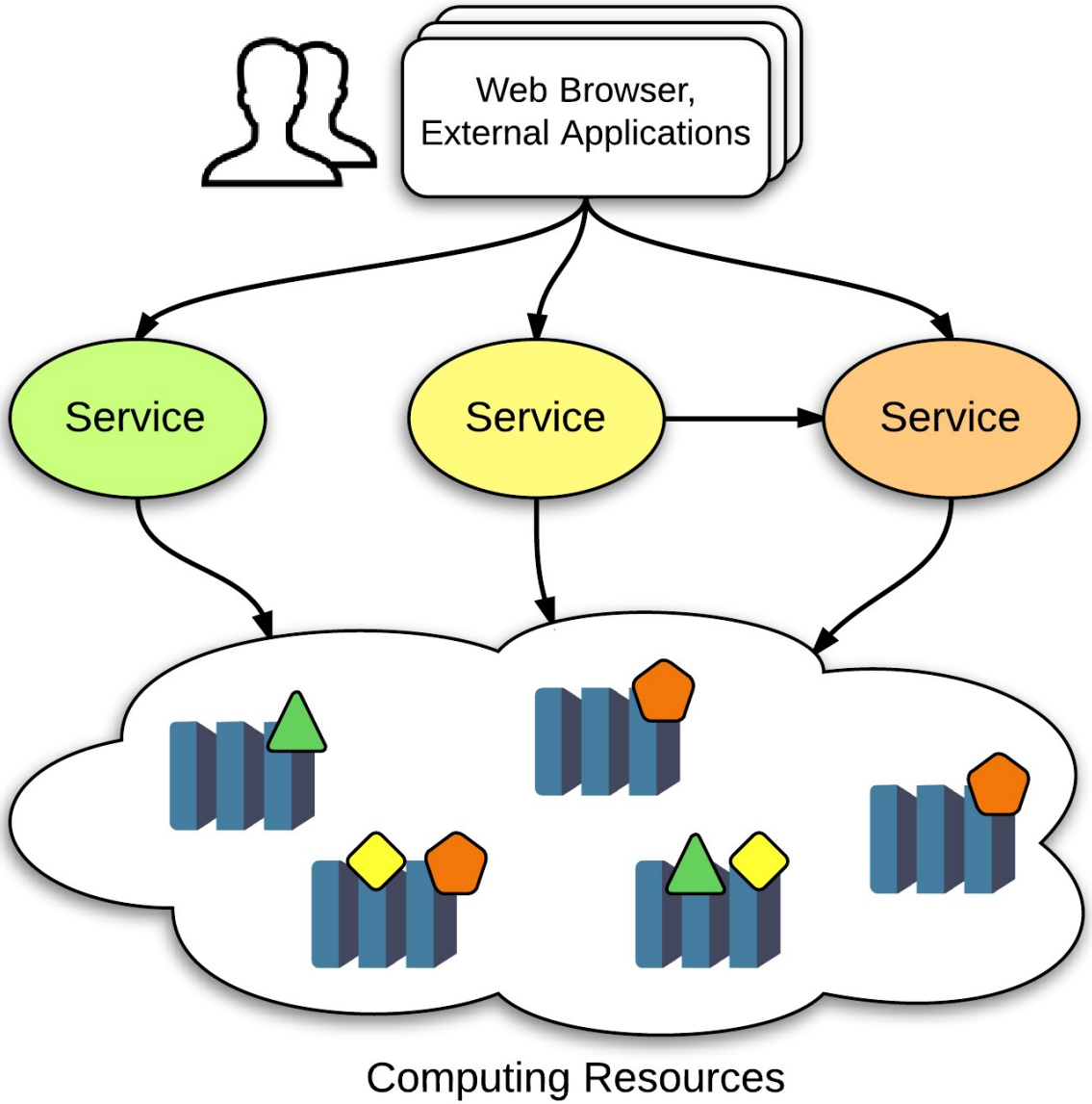
Oleg Sukhoroslov, Alexander Afanasiev,
Anton Rubtsov, Sergey Volkov

Institute for Information Transmission Problems (Moscow, Russia)

Agenda

- Scientific Application as a Service
- Service Model and Unified Interface
- Everest Architecture and Implementation

Scientific Application as a Service



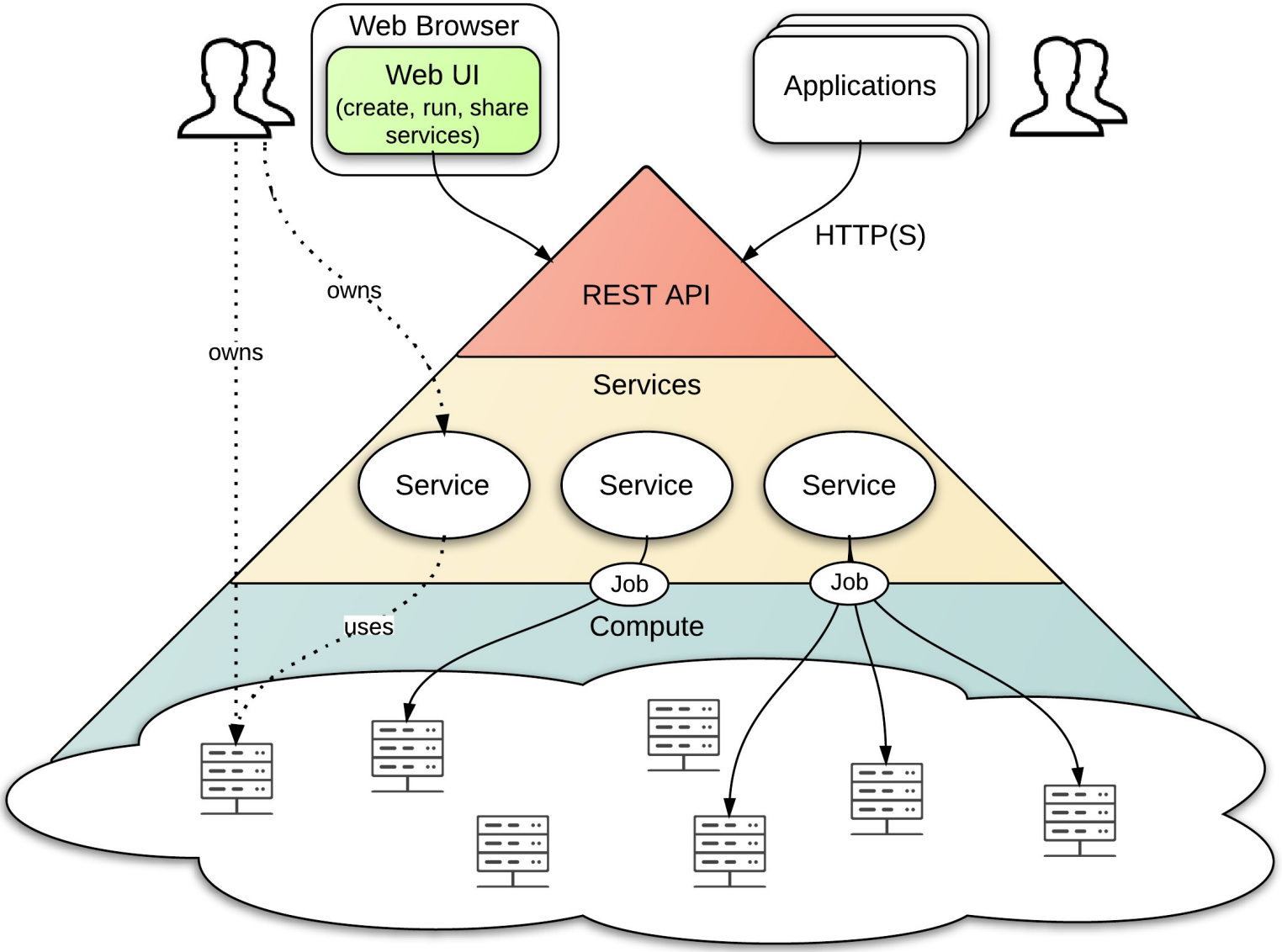
Related Approaches

- Computational Grids
  - Globus Toolkit, gLite, UNICORE...
  - Generic web service interfaces to computing resources
  - Low-level, hard to use for unskilled researchers
- Scientific Portals
  - P-GRADE, HubZero, Galaxy...
  - Convenient web user interfaces to applications and computing resources
  - Do not expose applications as services or provide programming interfaces
- Web Service Toolkits
  - GEMLCA, Opal, MathCloud...
  - Tools for exposing scientific applications as web services
  - Ad-hoc, no common practices, require an infrastructure for hosting services

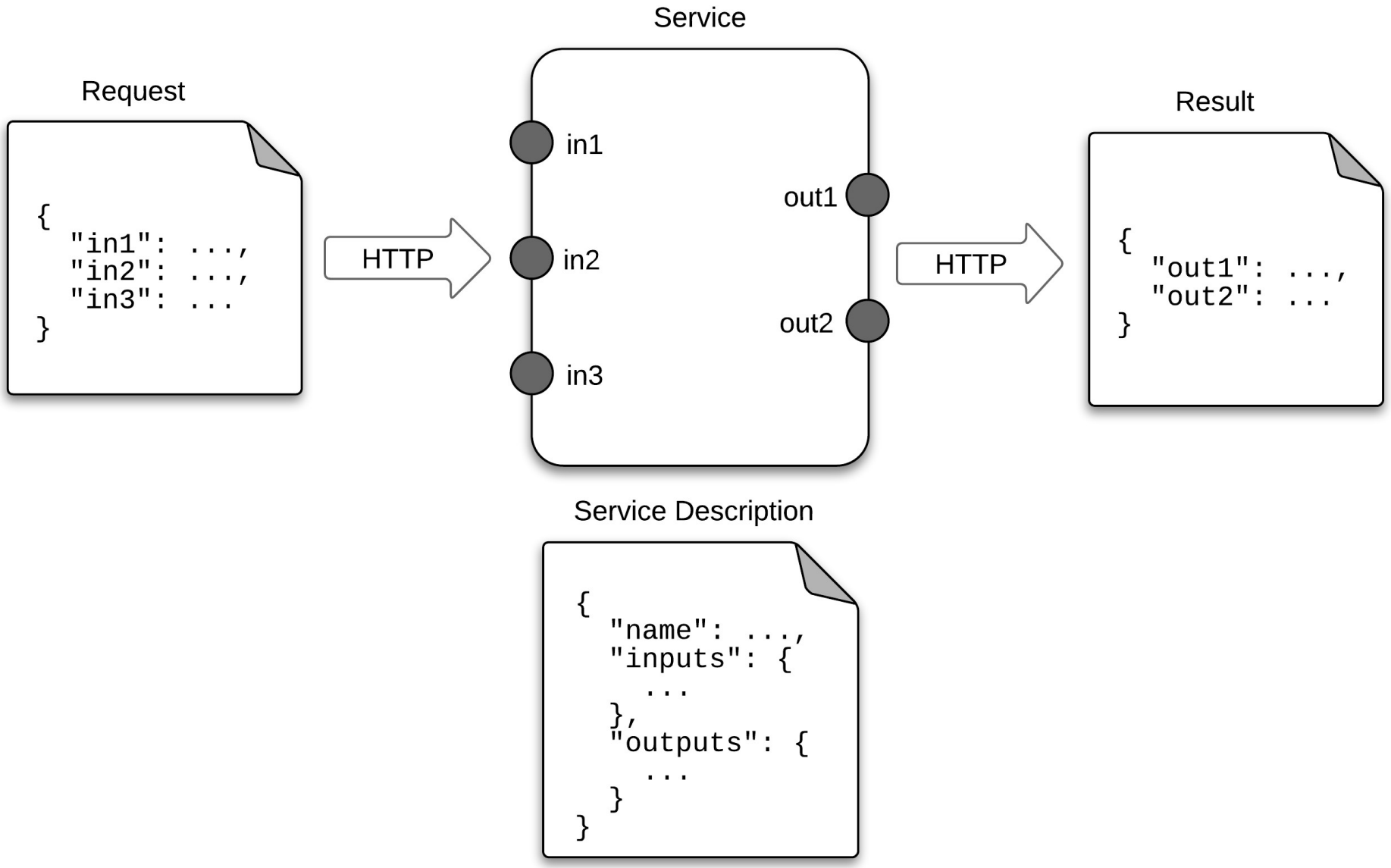
Everest

- Platform as a Service for development and hosting of scientific web services
  - Accessible via web browser
  - Supports multiple users
  - Enables users to create, share and access services
  - Integrates with external computing resources
- Combination of existing approaches + PaaS
  - Uniform service interface
  - Web UI for service development
  - Automatic generation of web UI for service invocation

Everest


Web Browser
Web UI
(create, run, share services)
Applications
HTTP(S)
REST API
owns
owns
Services
Service
Service
Service
Job
Job
uses
Compute
External Computing Resources (attached by users)

Service Model



POV-Ray

About | Parameters | Submit Job

Inputs

	Title	Name	Type	Values	Default	Description
✔	Scene file	scene	string		http://bit.ly/Pel8kW	
	INI file	ini	string			
	Include files	includes	array[string]			
✔	Output format	format	string	C N8 N16 P T	N8	
✔	Image width	width	integer	[1, 2048]	320	
✔	Image height	height	integer	[1, 2048]	240	
✔	Image quality	quality	integer	[0, 9]	9	0 = rough, 9 = full

Outputs

	Title	Name	Type	Description
✔	Output image	image	string	
✔	CPU utilization histogram	cpuHist	string	
✔	POV-Ray log	log	string	

Uniform Interface (REST API)

	GET	POST	PUT	DELETE
SERVICE	Get service description	Invoke service	Modify service	Delete service
JOB	Get job state/results	N/A	Modify job	Delete job
FILE	Download file	N/A	N/A	Delete file
SERVICES	List services	Create service	N/A	N/A
FILES	List files	Upload file	N/A	N/A

POV-Ray

About | Parameters | Submit Job

Scene file http://bit.ly/Pel8kW + Add file...

INI file + Add file...

Include files + Add file... +

Output format PNG 8 bits/color

Image width 320

Image height 240

Image quality 9

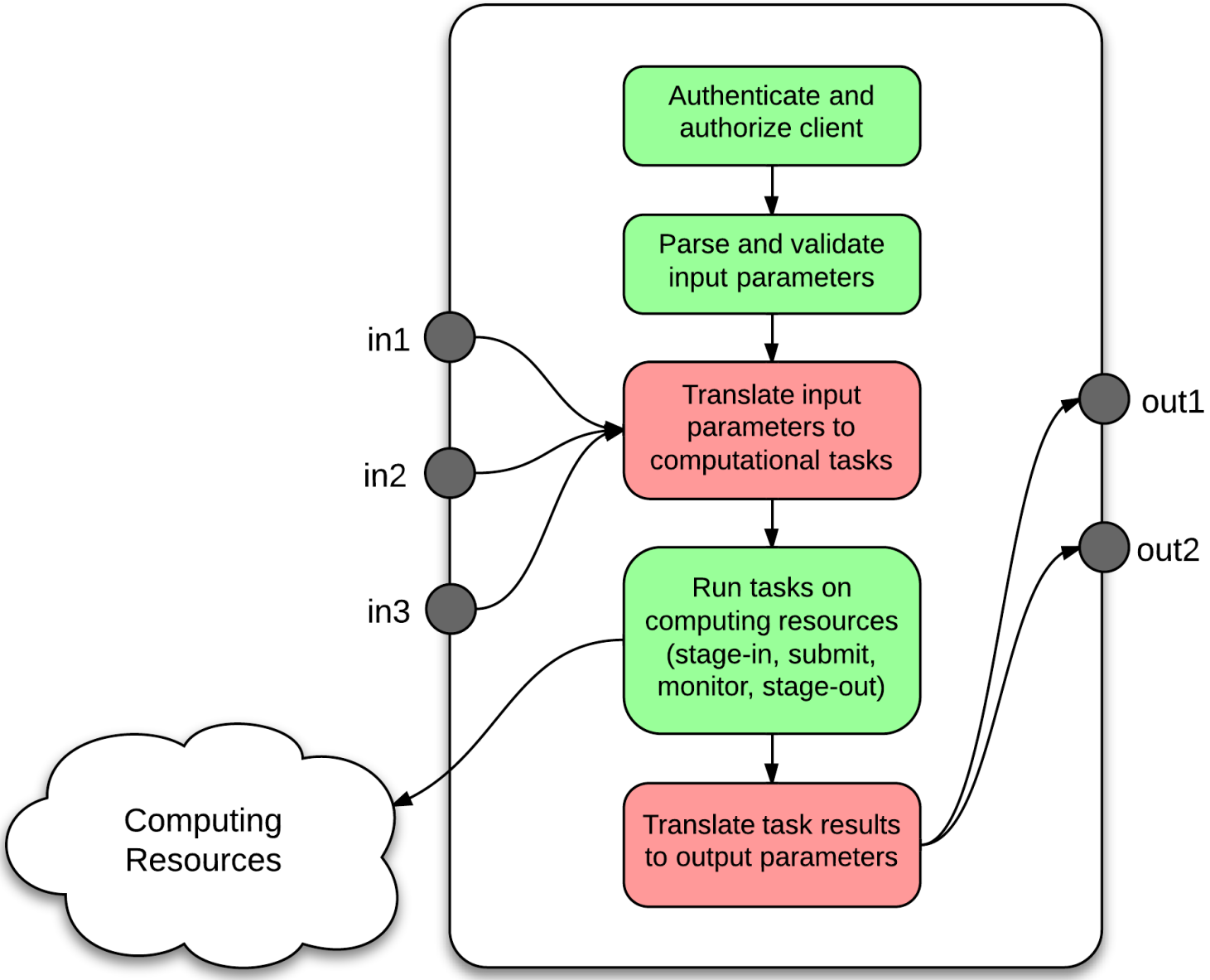
0 = rough, 9 = full

Request JSON

▶ Submit

```
{
  "inputs": {
    "scene": "http://bit.ly/Pel8kW",
    "includes": [],
    "format": "N16",
    "width": 320,
    "height": 240,
    "quality": 9
  }
}
```

Service Implementation



Request

```
{
  "in1": ...,
  "in2": ...,
  "in3": ...
}
```

Task

```
executable
arguments
input files
output files
resources
```

Request

```
{
  "out1": ...,
  "out2": ...
}
```

POV-Ray

Metadata | Inputs | Outputs | Configuration | Files | Resources | Access

Command

```
povray +Iscene.pov +F${format} +W${width} +H${height} +Q${quality} -D +HTN -Oimage
```

Refer to input values as ${param}

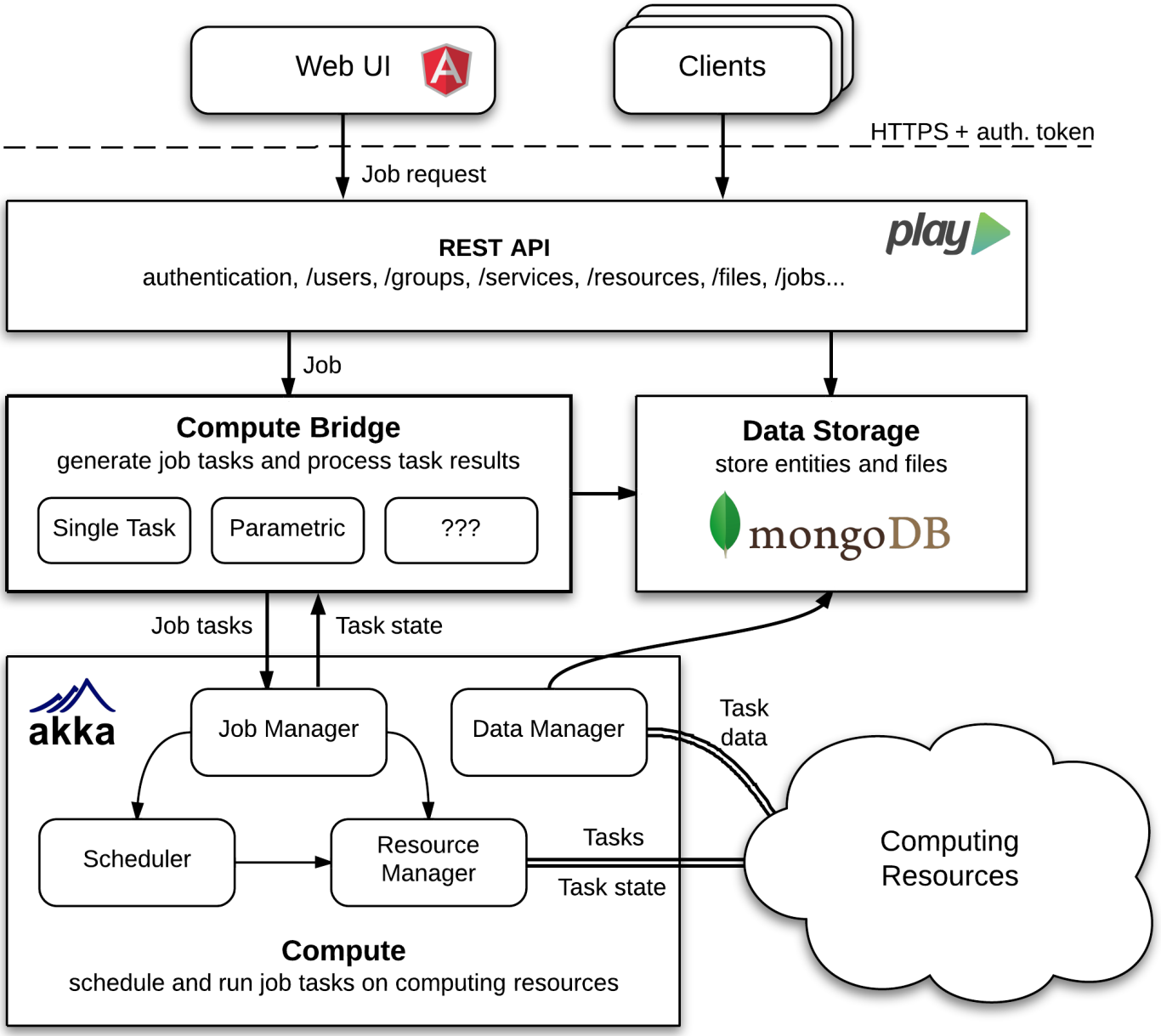
Input Mappings

Input	File	Pattern	
scene	scene.pov		🗑
ini	povray.ini		🗑
			+

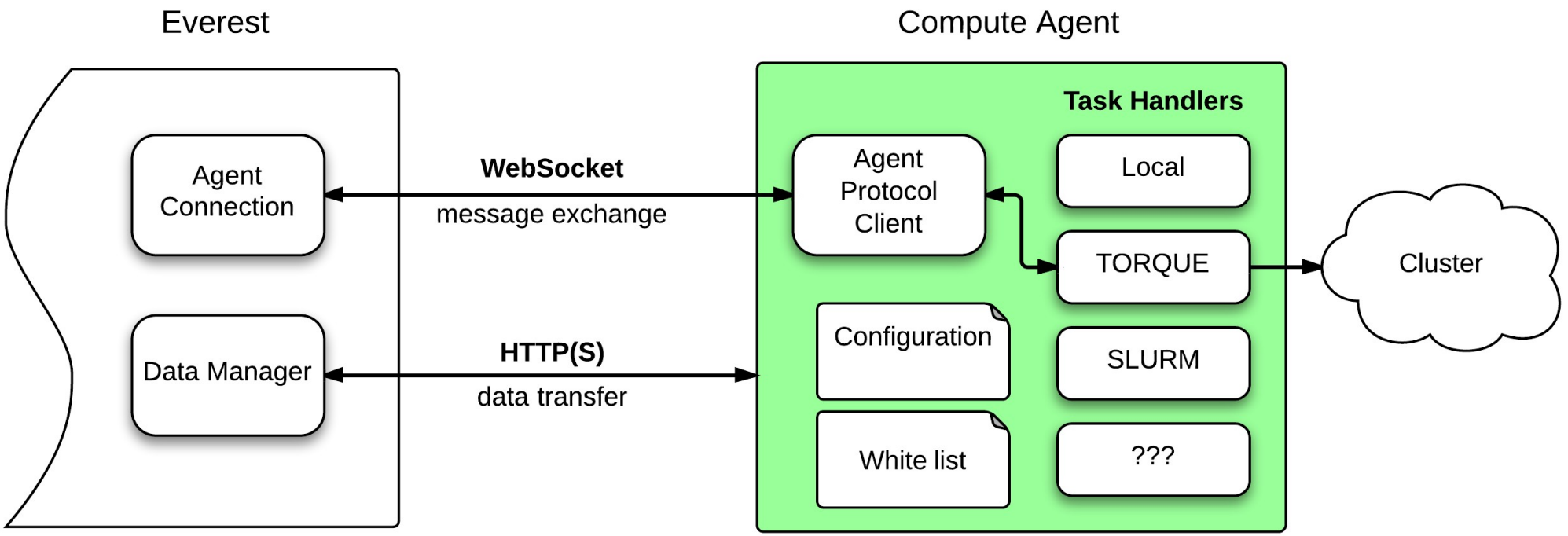
Output Mappings

Output	File	Pattern	
image	image.*		🗑
cpuHist	histgram.png		🗑
log	stderr		🗑
			+

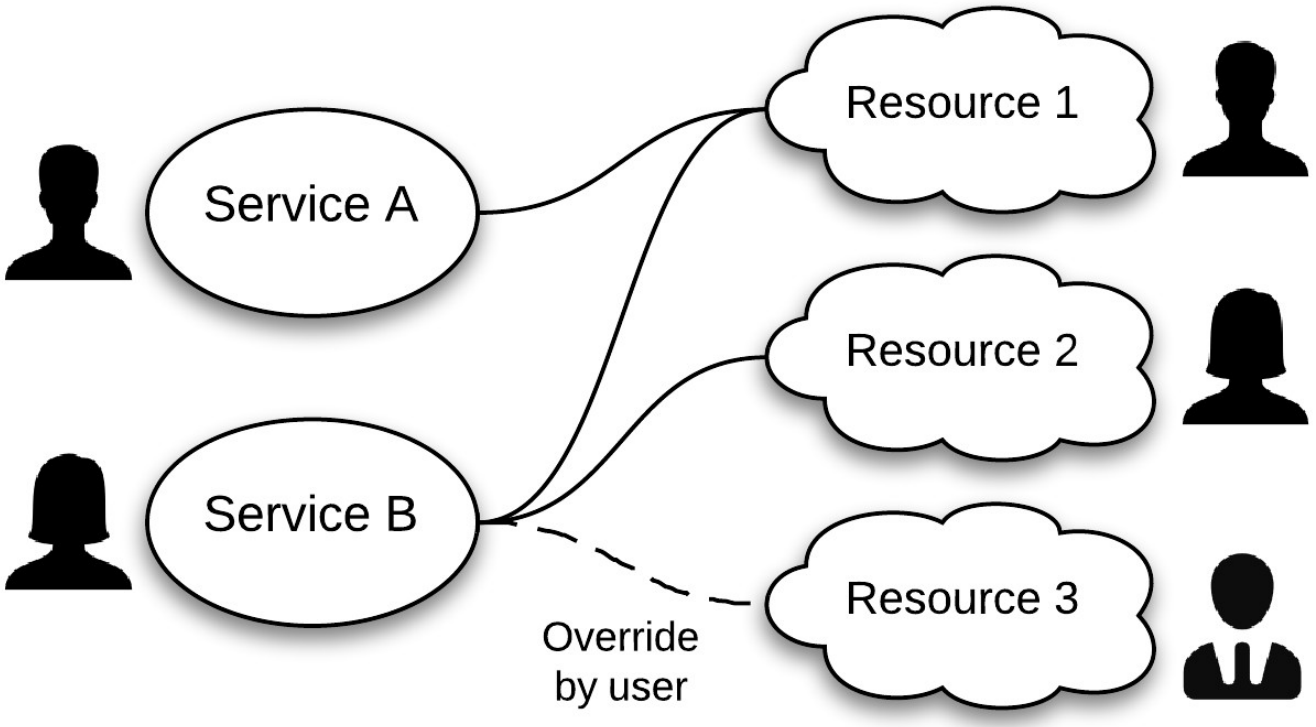
Everest Architecture



Integration with Computing Resources



Mapping Services to Resources



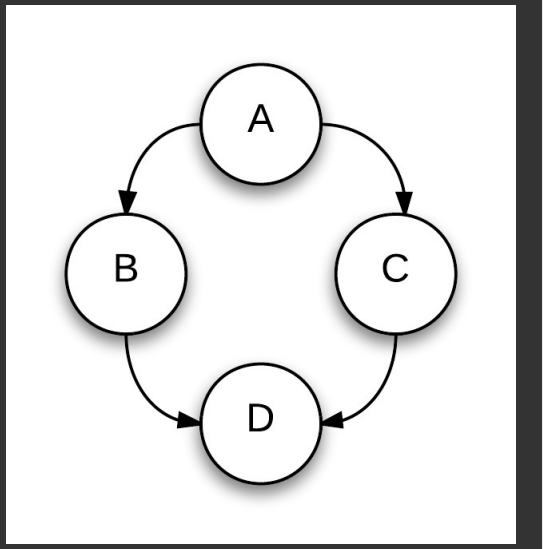
Python API

```
session = everest.Session(
  'https://mc2.distcomp.org',
  user = '...',
  password = '...'
)

serviceA = everest.Service('52b1d2d13b...', session)
serviceB = everest.Service('...', session)
serviceC = everest.Service('...', session)
serviceD = everest.Service('...', session)

jobA = serviceA.run({'a': '...'})
jobB = serviceB.run({'b': jobA.output('out1')})
jobC = serviceC.run({'c': jobA.output('out2')})
jobD = serviceD.run({'d1': jobB.output('out'), 'd2': jobC.output('out')})

print(jobD.result())
```



Applications

- Personal services
  - Ubiquitous access to applications + resources
  - Automate repetitive tasks
- Sharing services with colleagues
  - Collaborative workflows
  - Publication of results
  - Reproducibility
- Education

Conclusion

- Cloud Platform for Computational Web Services
  - Uniform service interface
  - Use of PaaS model (Web UI + REST API)
  - Flexible mapping of services to external computing resources
  - Pilot deployment
- Future work
  - Experimental evaluation, application case studies
  - Task scheduling across multiple resources
  - Integration with grid infrastructures
  - Integration with scientific workflow systems
  - Optimization of data transfer

Thank you for your attention!

https://mc2.distcomp.org/ demo:demo

oleg.sukhoroslov@gmail.com