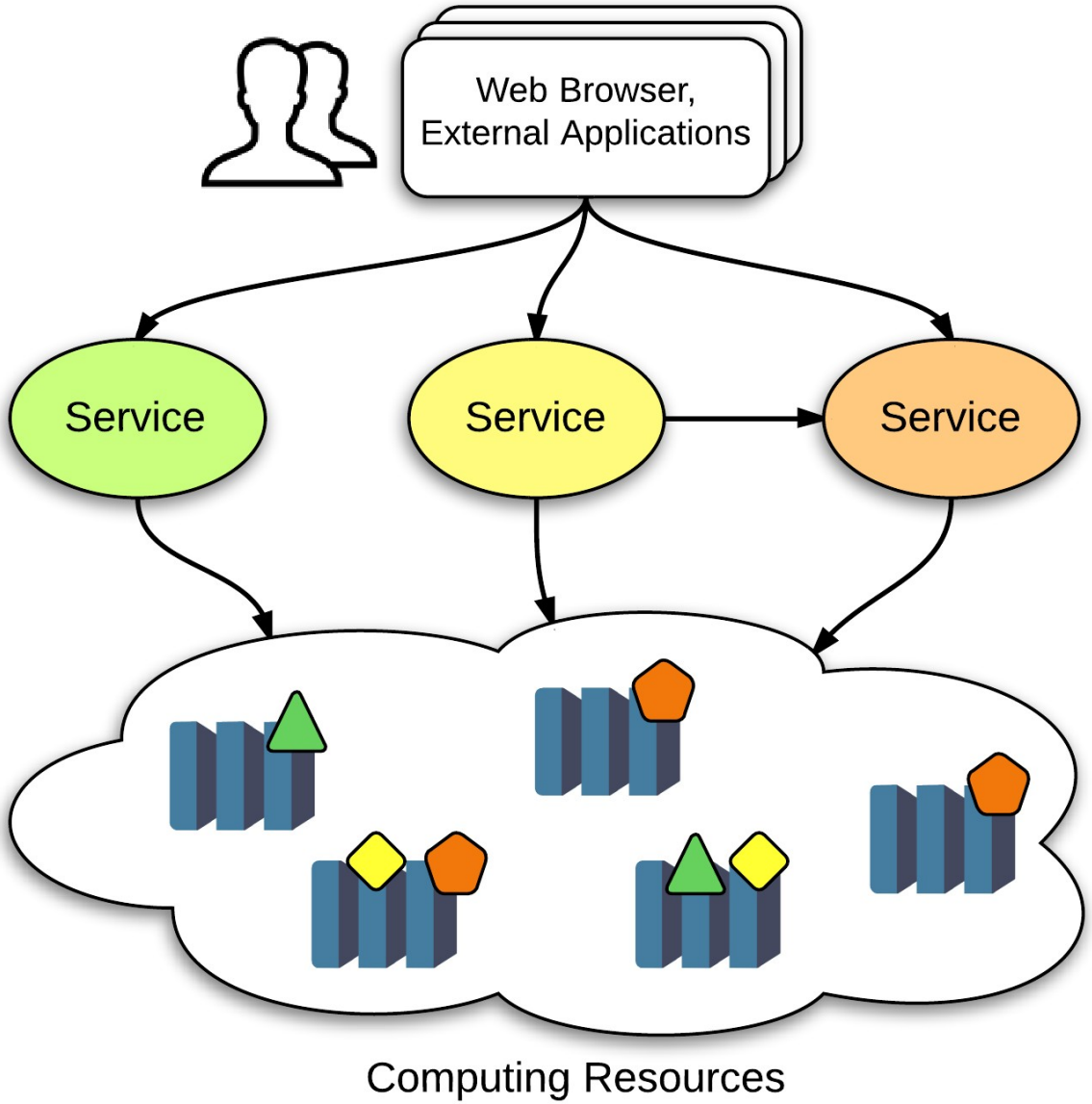
6th International Conference
"Distributed Computing and Grid-technologies in Science and Education" (GRID'2014)

Development of Distributed Computing Applications and Services with Everest Cloud Platform

Oleg Sukhoroslov, Anton Rubtsov, Sergey Volkov

Institute for Information Transmission Problems (Moscow, Russia)

Scientific Application as a Service



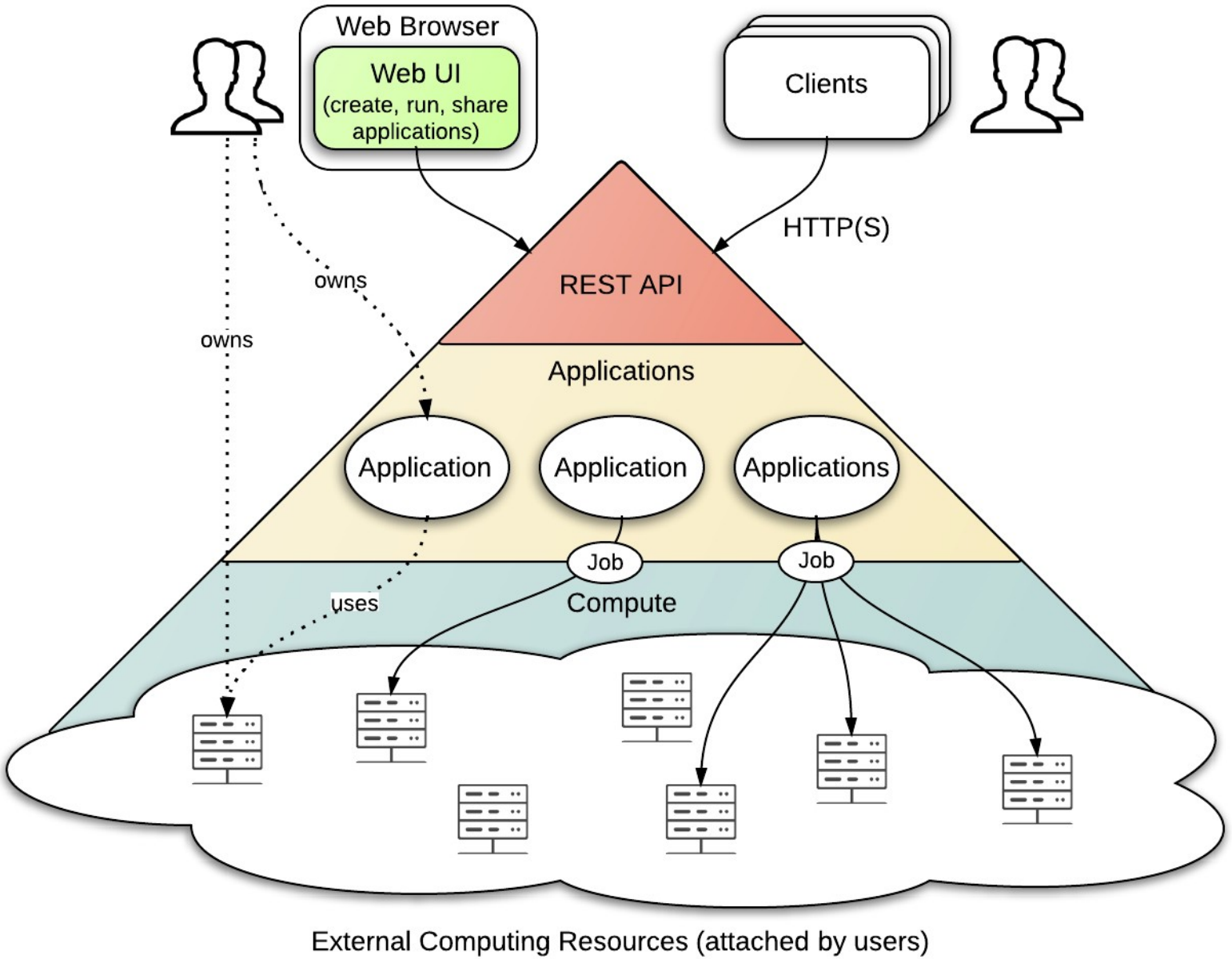
Related Approaches

- 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 supporting publication, execution and composition of applications running across distributed computing resources
  - Describe and expose applications as services
  - Bind computing resources to applications
  - Run applications on arbitrary sets of resources
  - Share applications and resources
- Platform as a Service
  - Accessible via web browser and REST API
  - No installation is required
- Combination of existing approaches + PaaS
  - Uniform REST interface for accessing applications
  - Web UI for application description
  - Automatic generation of web UI for application invocation

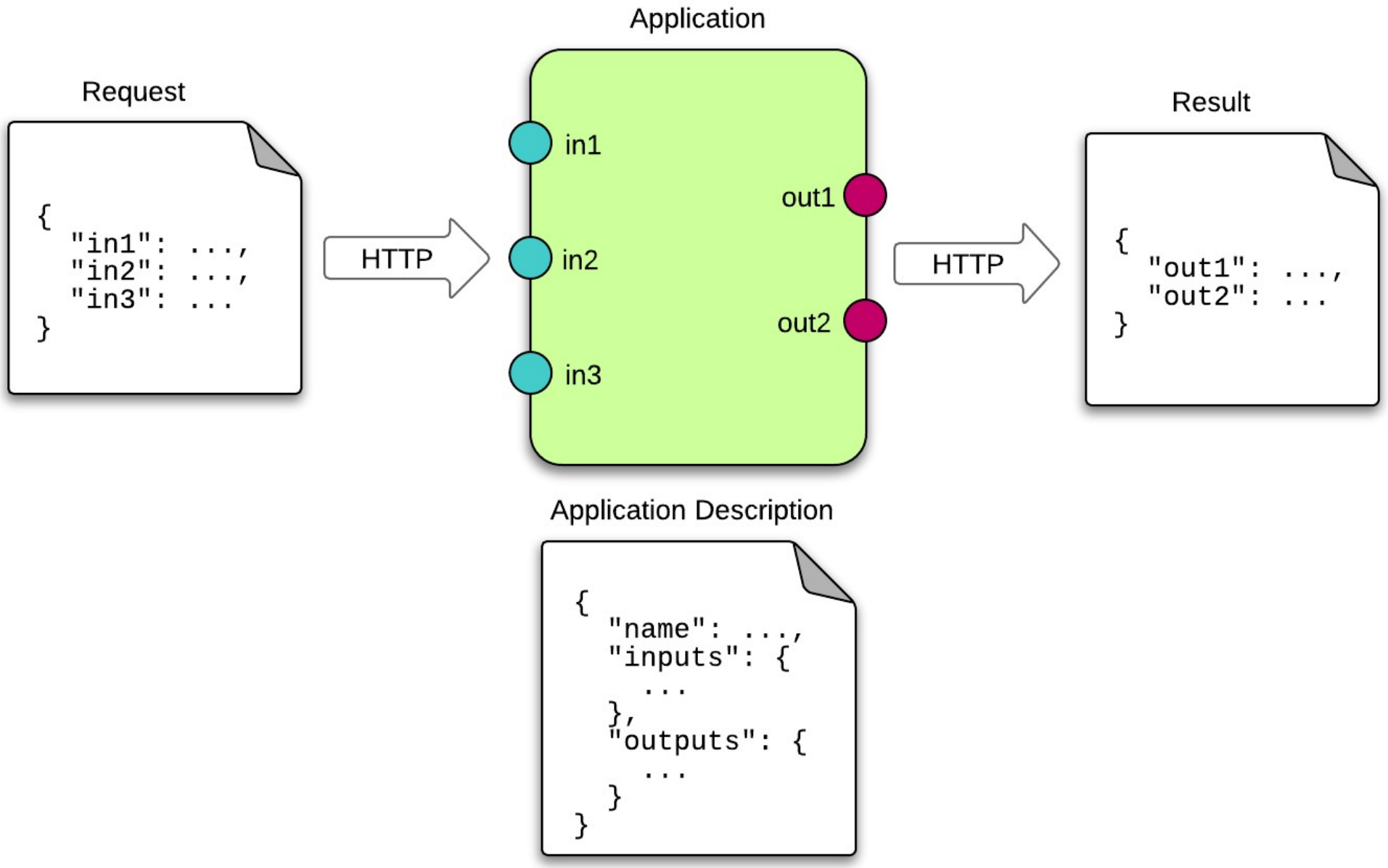
Everest


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

REST API

URI	GET	POST	PUT	DELETE
/api/apps/	List applications	Create application		
/api/apps/:id	Get application description	Invoke application (submit job)	Modify application	Delete application
/api/jobs/:id	Get job state/results		Modify job	Delete job
/api/files/(path)	List files	Upload file		
/api/files/:id/:name	Download file			Delete file
/auth/access_token		Create access token (log in)		Delete access token (log out)

Application: Interface



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	

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

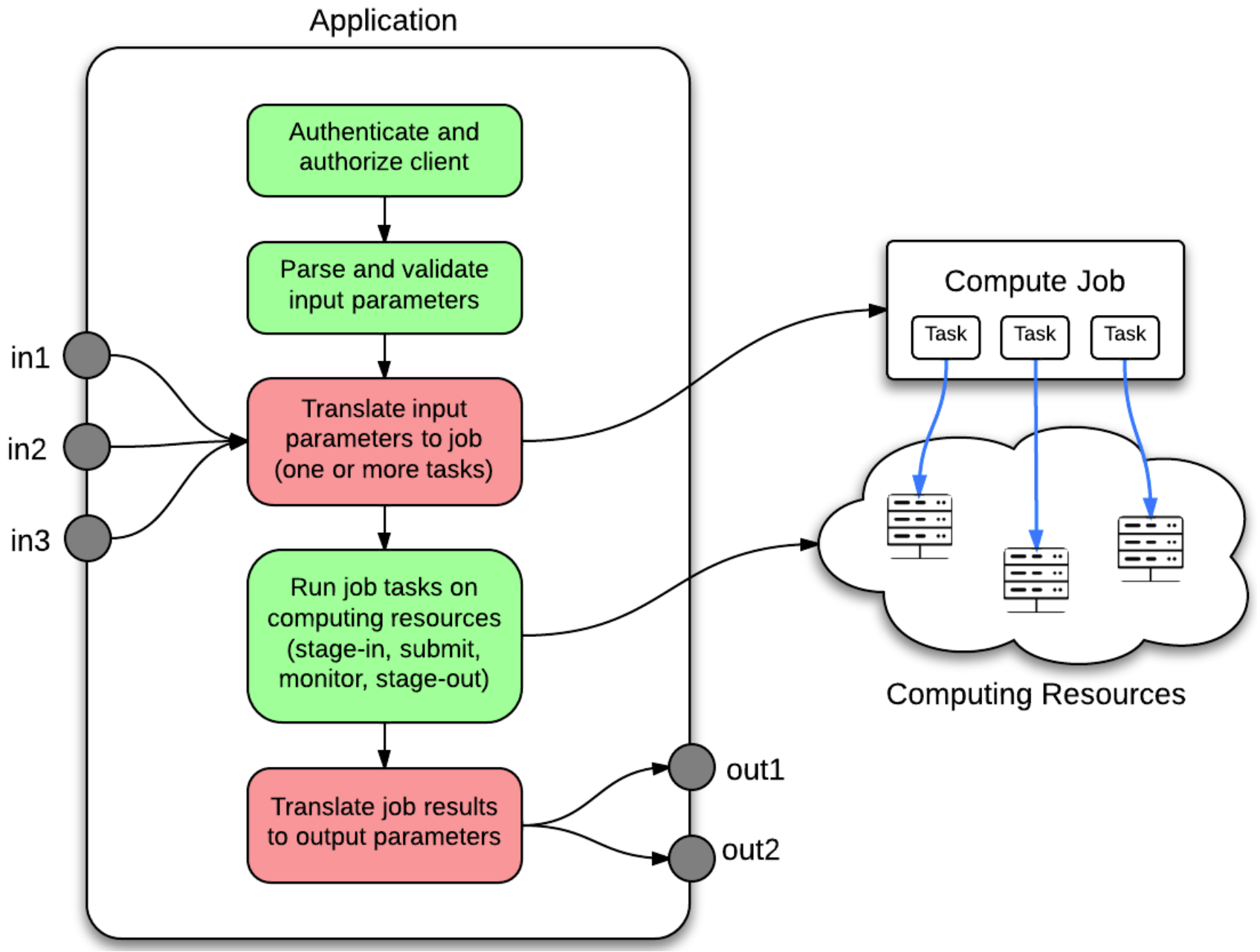
0 = rough, 9 = full

Request JSON

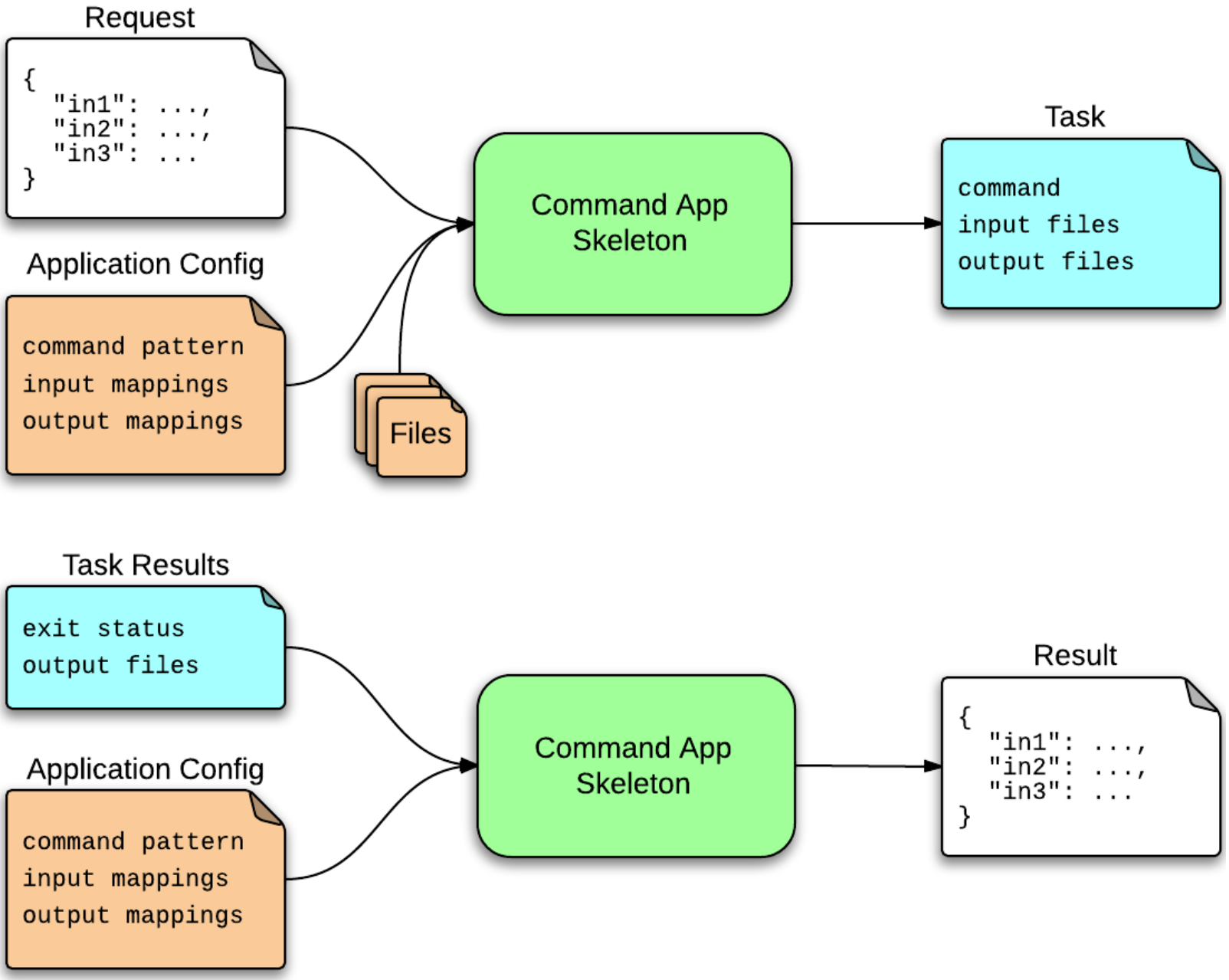
▶ Submit

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

Application: Implementation



Command Application Skeleton



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}

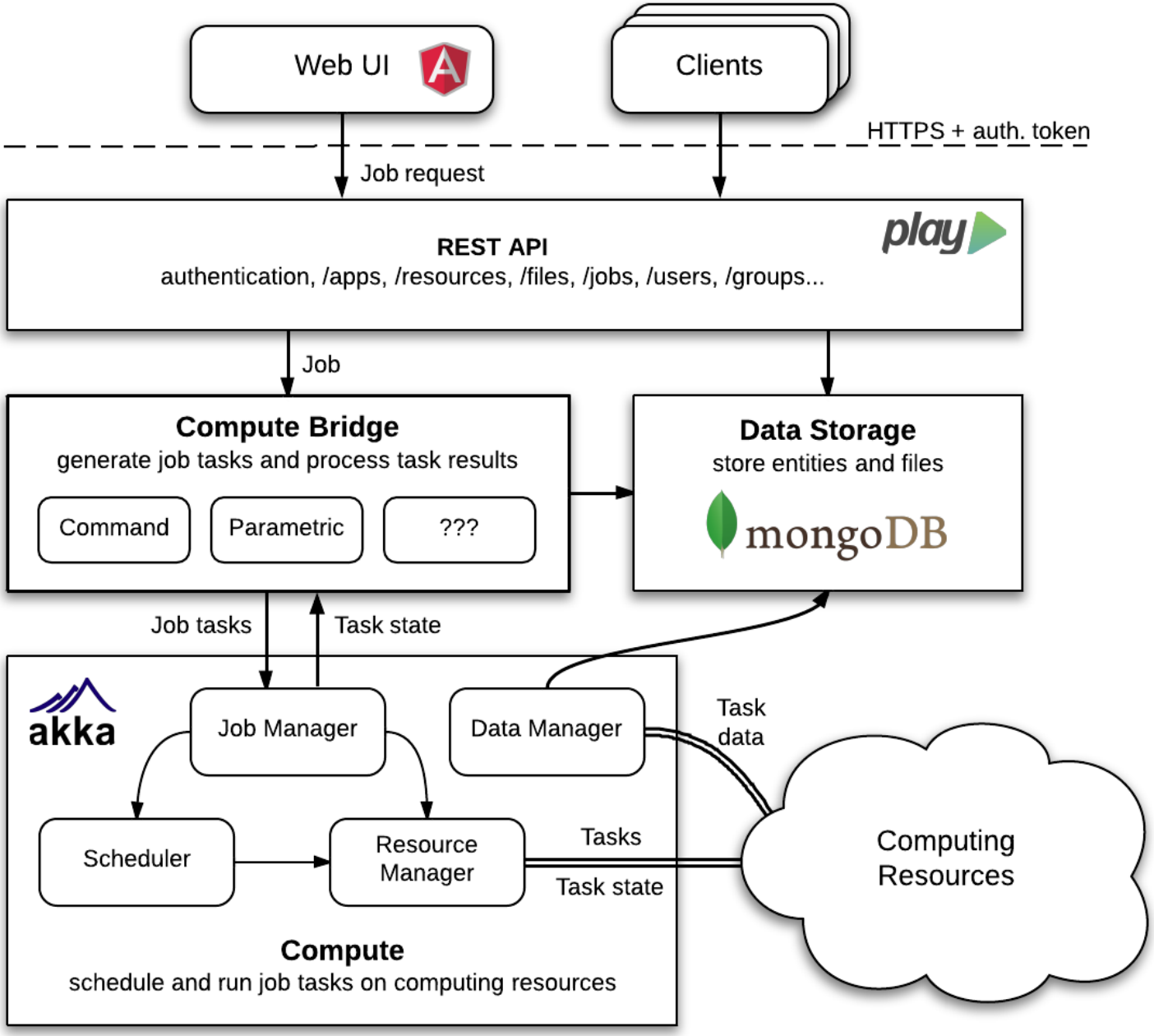
Input Mappings

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

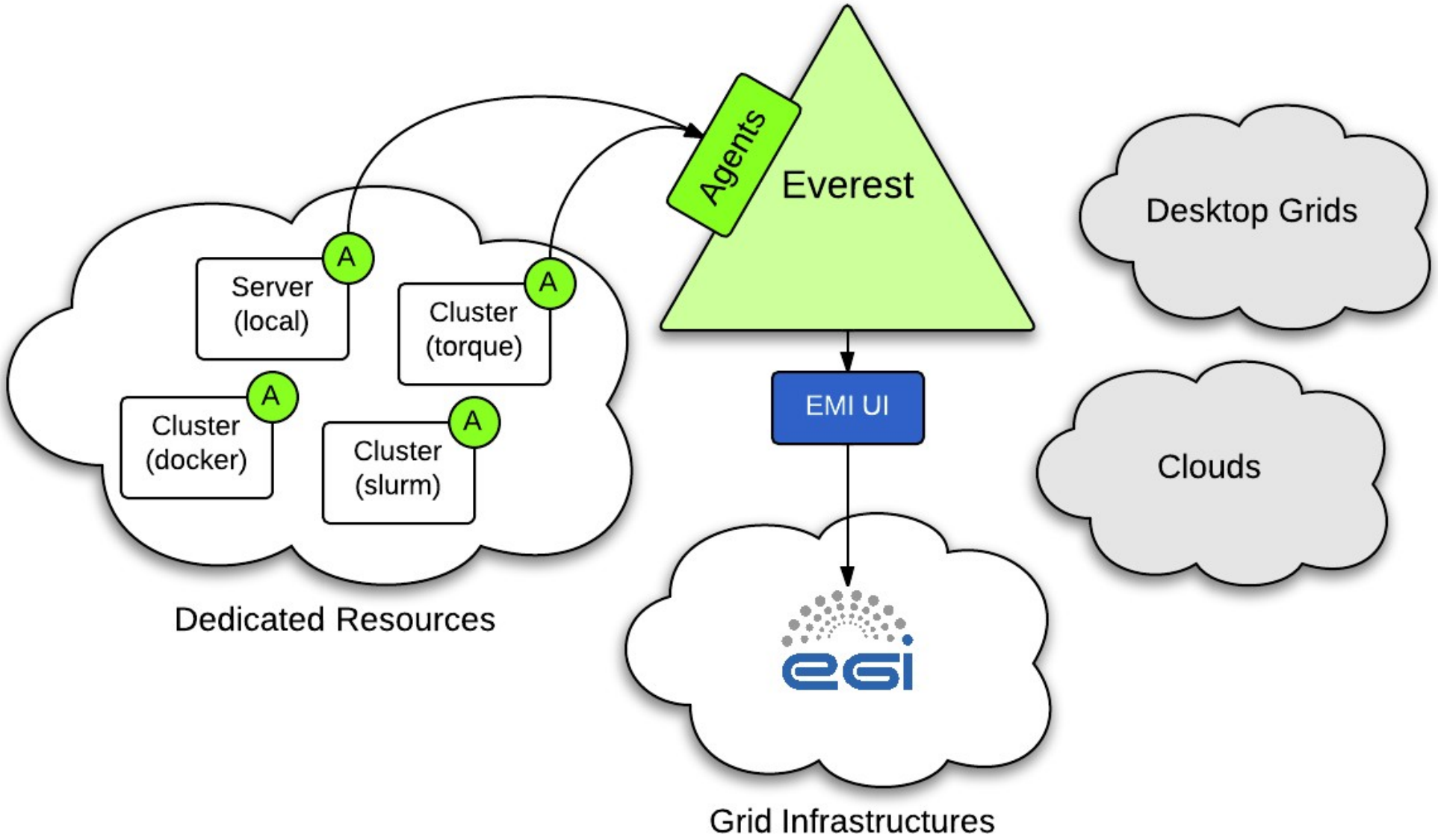
Output Mappings

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

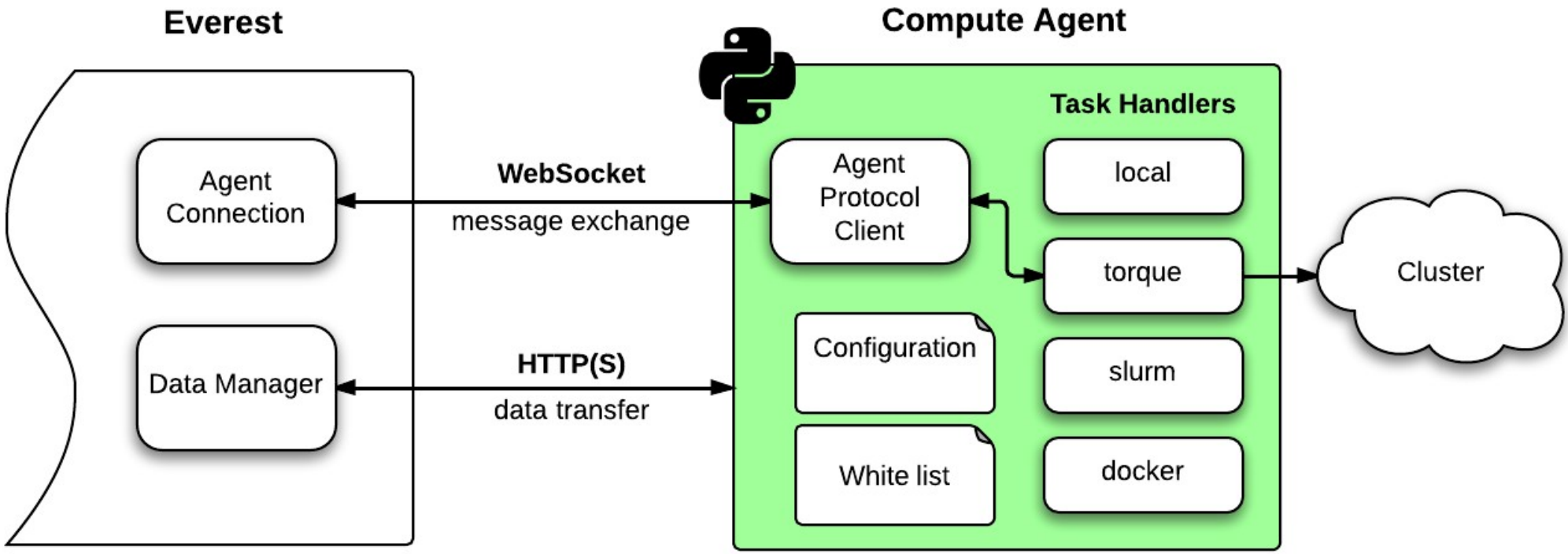
Everest Architecture



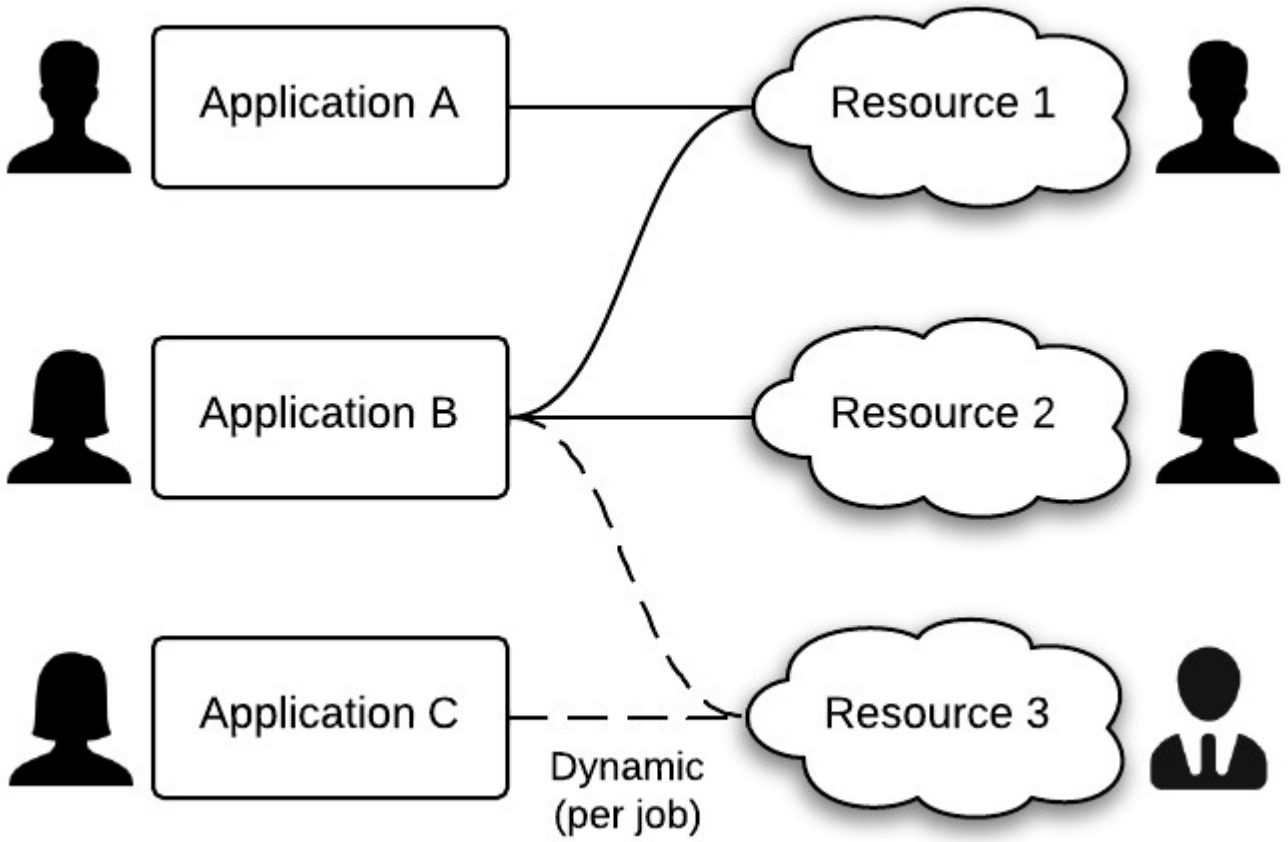
Integration with Computing Resources



Compute Agent



Binding Resources to Applications



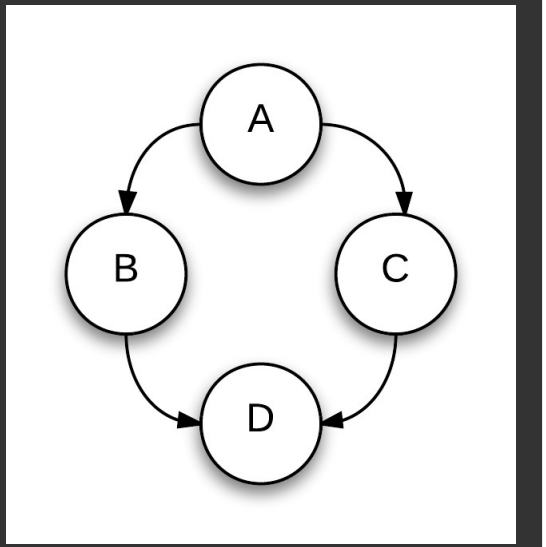
Python API

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

appA = everest.App('52b1d2d13b...', session)
appB = everest.App('...', session)
appC = everest.App('...', session)
appD = everest.App('...', session)

jobA = appA.run({'a': '...'})
jobB = appB.run({'b': jobA.output('out1')})
jobC = appC.run({'c': jobA.output('out2')})
jobD = appD.run({'d1': jobB.output('out'), 'd2': jobC.output('out')})

print(jobD.result())
```



Using Everest

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

Conclusion

- Distributed Computing Platform as a Service
  - Publication of applications as RESTful web services
  - Flexible mapping of applications to external computing resources
  - Use of PaaS model (Web UI + REST API)
  - http://everest.distcomp.org/
- Future Work
  - Experimental evaluation, application case studies
  - Advanced scheduling across multiple resources
  - Support for parallel applications
  - Integration with other types of computing resources
  - Optimization of data transfer