

TROY

Tiered Resource OverlaY

Andre Merzky

<http://saga-project.github.com/troy/>



- ① Motivation / Use Cases
- ② Conceptual Architecture
- ③ Implementation Architecture / Design
- ④ API, Code Example

Motivation for TROY

- provide **uniform interface** to pilot frameworks (PF)

Motivation for TROY

- provide **uniform interface** to pilot frameworks (PF)
- support for application level **scheduling on** those PFs

Motivation for TROY

- provide **uniform interface** to pilot frameworks (PF)
- support for application level **scheduling on** those PFs
- support for application level **scheduling across** those PFs

Use Case (i)

- NGS data on storage archive on XSEDE/lonestar

Use Case (i)

- NGS data on storage archive on XSEDE/lonestar
- compute pilots on XSEDE

Use Case (i)

- NGS data on storage archive on XSEDE/Ionestar
- compute pilots on XSEDE
- scheduling of data allocations & transfer

Use Case (i)

- NGS data on storage archive on XSEDE/lonestar
- compute pilots on XSEDE
- scheduling of data allocations & transfer
- scheduling compute node allocation & execution

Use Case (ii)

- NGS data on storage archive on XSEDE/lonestar

Use Case (ii)

- NGS data on storage archive on XSEDE/lonestar
- compute pilots on XSEDE and EGI

Use Case (ii)

- NGS data on storage archive on XSEDE/Ionestar
- compute pilots on XSEDE and EGI
- scheduling of data allocations & transfer

Use Case (ii)

- NGS data on storage archive on XSEDE/Ionestar
- compute pilots on XSEDE and EGI
- scheduling of data allocations & transfer
- scheduling compute node allocation & execution

Use Case (iii)

- develop new scheduling algorithm / data placement policy

Use Case (iii)

- develop new scheduling algorithm / data placement policy
- run workload from UC(i) with backend scheduler (i.e. Bigjob)

Use Case (iii)

- develop new scheduling algorithm / data placement policy
- run workload from UC(i) with backend scheduler (i.e. Bigjob)
- run same workload with application level scheduler

Use Case (iii)

- develop new scheduling algorithm / data placement policy
- run workload from UC(i) with backend scheduler (i.e. Bigjob)
- run same workload with application level scheduler
- compare, analyze

Use Case (iii)

- develop new scheduling algorithm / data placement policy
- run workload from UC(i) with backend scheduler (i.e. Bigjob)
- run same workload with application level scheduler
- compare, analyze
- ...

Use Case (iii)

- develop new scheduling algorithm / data placement policy
- run workload from UC(i) with backend scheduler (i.e. Bigjob)
- run same workload with application level scheduler
- compare, analyze
- ...
- publish ;-)

TROY Placement and Scope

- **is-a** application framework
 - application defines resource requirments
 - application defines data and compute workload

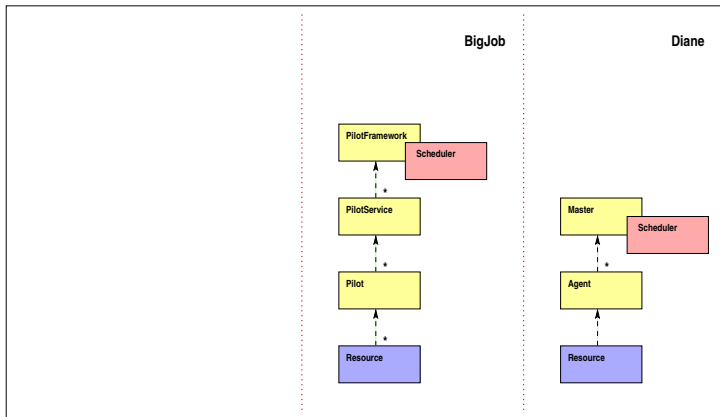
TROY Placement and Scope

- **is-a** application framework
 - application defines resource requirements
 - application defines data and compute workload
- **is-a** scheduling framework
 - hosts scheduling algorithms
 - interfaces to external schedulers
 - enacts scheduling decisions

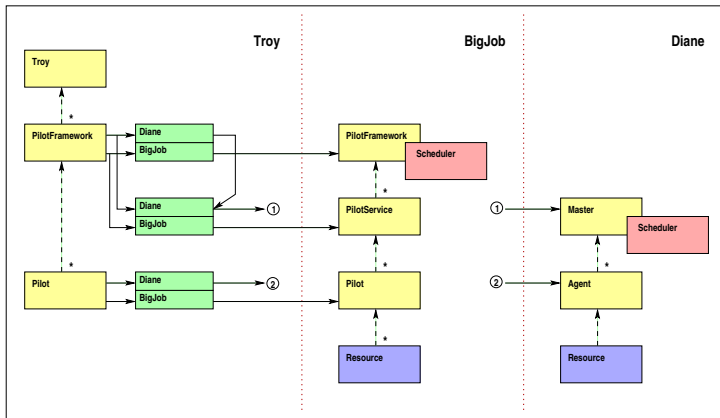
TROY Placement and Scope

- **is-a** application framework
 - application defines resource requirements
 - application defines data and compute workload
- **is-a** scheduling framework
 - hosts scheduling algorithms
 - interfaces to external schedulers
 - enacts scheduling decisions
- **interface to** pilot job frameworks
 - assumes P*, and possibly Pilot API

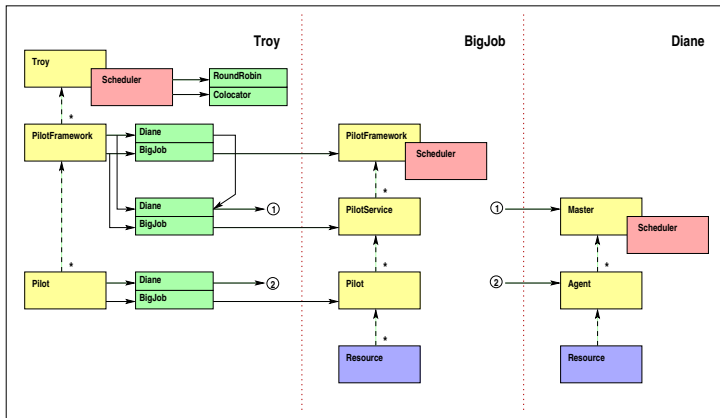
Architecture



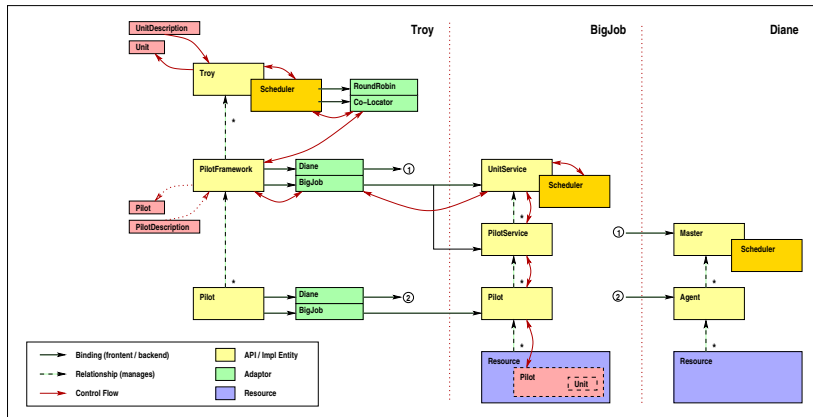
Architecture



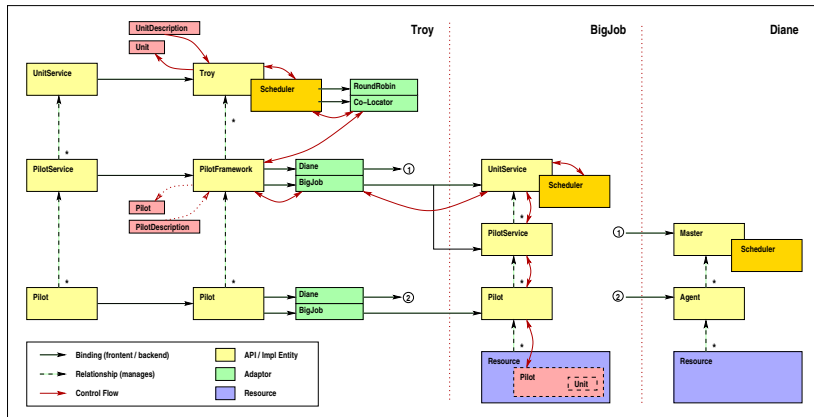
Architecture



Architecture



Architecture



Troy API Classes

Troy classes interfacing to backend pilot systems:

- `troy.Scheduler`
- `troy.PilotFramework`
interfaces to the `XXXUnitService` and `XXXPilotService`
classes of Pilot API
- `troy.ComputePilot`
- `troy.ComputeUnit`
- `troy.DataPilot`
- `troy.DataUnit`

Troy API

API Example

```
pf1 = troy.PilotFramework ('bigjob//lonestar')
pf2 = troy.PilotFramework ('bigjob//kraken')

cpd = troy.ComputePilotDescription ()
pf1.submit_pilot (cpd)
pf2.submit_pilot (cpd)

t = troy.Troy ()
t.add_pilot_framework (pf1)
t.add_pilot_framework (pf2)

s = troy.Scheduler ('Random')
t.add_scheduler (s)
```

Troy API

API Example Cont.

```
cud = troy.ComputeUnitDescription ()  
  
cud['executable'] = '/bin/sh'  
cud['arguments'] = ['-c', 'touch /tmp/hello_troy_pj && sleep 10']  
  
cu  = t.submit_unit (cud)
```

Troy API

API Example Cont.

```
while cu.state not in (troy.Done, troy.Failed) :  
  
    print "? cu %s: %s" % (cu.id, cu.state)  
    time.sleep (1)  
  
print "! cu %s: %s" % (cu.id, cu.state)  
  
pf1.cancel ()  
pf2.cancel ()
```

Troy API

Scheduler

```
def my_scheduler (troy, ud) :  
  
    pilots = []  
  
    for pf_id in troy.list_pilot_frameworks () :  
        pf = troy.PilotFramework (pf_id)  
  
        for p_id in pf.list_pilots () :  
            p = troy.ComputePilot (p_id)  
            pilots.append (p)  
  
    tgt = pilots[random.randint (0, len (pilots) - 1)]  
    return tgt.submit_unit (ud)
```


Troy API

API Example + Scheduler

```
pf1 = troy.PilotFramework ('bigjob//lonestar')
pf2 = troy.PilotFramework ('bigjob//kraken')

cpd  = troy.ComputePilotDescription ()
pf1.submit_pilot (cpd)
pf2.submit_pilot (cpd)

t    = troy.Troy ()
t.add_pilot_framework (pf1)
t.add_pilot_framework (pf2)

s = troy.Scheduler ('Random')
t.add_scheduler      (s)
```

Troy API

API Example + Scheduler

```
pf1 = troy.PilotFramework ('bigjob//lonestar')
pf2 = troy.PilotFramework ('bigjob//kraken')

cpd  = troy.ComputePilotDescription ()
pf1.submit_pilot (cpd)
pf2.submit_pilot (cpd)

t    = troy.Troy ()
t.add_pilot_framework (pf1)
t.add_pilot_framework (pf2)

t.add_scheduler      (my_scheduler)
```

Troy API

Questions?