



ParslFest 2023

# Multi-User Endpoints



Kevin Hunter Kesling – [kevin@globus.org](mailto:kevin@globus.org)

October 20, 2023



THE UNIVERSITY OF  
CHICAGO



Argonne  
NATIONAL LABORATORY

globus

# What is a Multi-User Compute Endpoint?



- Reminder of “normal endpoints”
  - Configure
  - Start (which we interpret at the web-services as “register”)
  - Use the printed uuid identifier in your SDK scripts
  - “Oh wait, I need to change that configuration because [reason]”
  - Log in, change the configuration, restart
  - “Oh shoot, the cluster has been restarted”
  - Log in, start the endpoint afresh
  - “Hmm ...I need more than one configuration for my various workloads”
  - Log in, and manage them ad-hoc
- ... And so forth; for many folks, it’s not uncommon to have multiple configurations, that each need to be managed

# What is a Multi-User Compute Endpoint?



- Aside ... hereafter:
  - MEP → Multi-User Endpoint
  - UEP → User Endpoint (“normal endpoint”)



- Reminder of “normal endpoints”
  - Configure
  - Start (which we interpret at the web-services as “register”)
  - Use the printed uuid identifier in your SDK scripts
  - “Oh wait, I need to change that configuration because [reason]”
  - Log in, change the configuration, restart
  - “Oh shoot, the cluster has been restarted”
  - Log in, start the endpoint afresh
  - “Hmm ...I need more than one configuration for my various workloads”
  - Log in, and manage them ad-hoc
- ... And so forth; for many folks, it’s not uncommon to have multiple configurations, that each need to be managed

# What is a Multi-User Compute Endpoint?



- Aside ... hereafter:
  - MEP → Multi-User Endpoint
  - UEP → User Endpoint (“normal endpoint”)
- In contrast to a “normal” compute endpoint, an MEP **does not run tasks**.



- Reminder of “normal endpoints”
  - Configure
  - Start (which we interpret at the web-services as “register”)
  - Use the printed uuid identifier in your SDK scripts
  - “Oh wait, I need to change that configuration because [reason]”
  - Log in, change the configuration, restart
  - “Oh shoot, the cluster has been restarted”
  - Log in, start the endpoint afresh
  - “Hmm ...I need more than one configuration for my various workloads”
  - Log in, and manage them ad-hoc
- ... And so forth; for many folks, it’s not uncommon to have multiple configurations, that each need to be managed

# What is a Multi-User Compute Endpoint?



- Aside ... hereafter:
  - MEP → Multi-User Endpoint
  - UEP → User Endpoint (“normal endpoint”)
- In contrast to a “normal” compute endpoint, an MEP **does not run tasks**.
- Instead, an MEP
  - starts UEPs
  - (Slightly more precisely, fork, drop privileges, exec)
  - Manages their lifecycle (okay, `os.fork()` and `os.waitpid()`)



- Reminder of “normal endpoints”
  - Configure
  - Start (which we interpret at the web-services as “register”)
  - Use the printed uuid identifier in your SDK scripts
  - “Oh wait, I need to change that configuration because [reason]”
  - Log in, change the configuration, restart
  - “Oh shoot, the cluster has been restarted”
  - Log in, start the endpoint afresh
  - “Hmm ...I need more than one configuration for my various workloads”
  - Log in, and manage them ad-hoc
- ... And so forth; for many folks, it’s not uncommon to have multiple configurations, that each need to be managed

# What is a Multi-User Compute Endpoint?



- Aside ... hereafter:
  - MEP → Multi-User Endpoint
  - UEP → User Endpoint (“normal endpoint”)
- In contrast to a “normal” compute endpoint, an MEP **does not run tasks**.
- Instead, an MEP
  - starts UEPs
  - (Slightly more precisely, fork, drop privileges, exec)
  - Manages their lifecycle (okay, `os.fork()` and `os.waitpid()`)
- Receives start UEP commands from the web-service



- Reminder of “normal endpoints”
  - Configure
  - Start (which we interpret at the web-services as “register”)
  - Use the printed uuid identifier in your SDK scripts
  - “Oh wait, I need to change that configuration because [reason]”
  - Log in, change the configuration, restart
  - “Oh shoot, the cluster has been restarted”
  - Log in, start the endpoint afresh
  - “Hmm ...I need more than one configuration for my various workloads”
  - Log in, and manage them ad-hoc
- ... And so forth; for many folks, it’s not uncommon to have multiple configurations, that each need to be managed

# htop screen recording



PID/USER	RES	S	CPU%	Command
1923814 root	126M	S	0.0	Globus Compute Endpoint *(290ffb16-c7f2-4799-a314-f4fd67787edd, test_mt) -
1924147 kevin	142M	S	0.0	Globus Compute Endpoint (fbcbb7eb-0251-1f36-e706-54273e576aa3, uep.290f
1924217 kevin	124M	S	1.3	Globus Compute Endpoint (fbcbb7eb-0251-1f36-e706-54273e576aa3, uep.2
1924233 kevin	123M	S	0.0	Globus Compute Endpoint (fbcbb7eb-0251-1f36-e706-54273e576aa3, uep.2
1924243 kyle	142M	S	0.6	Globus Compute Endpoint (3d872cb2-6da9-11ee-94fd-5779496bdfed, uep.290f
1924266 kyle	124M	S	0.6	Globus Compute Endpoint (3d872cb2-6da9-11ee-94fd-5779496bdfed, uep.2
1924282 kyle	123M	S	0.0	Globus Compute Endpoint (3d872cb2-6da9-11ee-94fd-5779496bdfed, uep.2
1924303 harper	142M	S	0.0	Globus Compute Endpoint (405864d8-6da9-11ee-b71e-8b2ed7f32de6, uep.290f
1924368 harper	125M	S	1.3	Globus Compute Endpoint (405864d8-6da9-11ee-b71e-8b2ed7f32de6, uep.2
1924384 harper	123M	S	0.0	Globus Compute Endpoint (405864d8-6da9-11ee-b71e-8b2ed7f32de6, uep.2
1924399 jessica	142M	S	0.0	Globus Compute Endpoint (40d3bdc2-6da9-11ee-a600-efb50f3bbdb, uep.290f
1924419 jessica	124M	S	0.6	Globus Compute Endpoint (40d3bdc2-6da9-11ee-a600-efb50f3bbdb, uep.2
1924435 jessica	123M	S	0.0	Globus Compute Endpoint (40d3bdc2-6da9-11ee-a600-efb50f3bbdb, uep.2
1924445 rowan	142M	S	0.0	Globus Compute Endpoint (41357daa-6da9-11ee-9130-a70f3801bb30, uep.290f
1924472 rowan	124M	S	1.3	Globus Compute Endpoint (41357daa-6da9-11ee-9130-a70f3801bb30, uep.2
1924488 rowan	123M	S	0.0	Globus Compute Endpoint (41357daa-6da9-11ee-9130-a70f3801bb30, uep.2

PDF NOTE: Original presentation had a live screen recording, showing the values updating in real time as “presentation-proof” that the software exists (if not yet released). See speaker notes.

Video of original presentation linked via the [ParslFest 2023 list of presentations](https://parsl-project.org/parslfest/parslfest2023.html).  
(<https://parsl-project.org/parslfest/parslfest2023.html>)



- Showing it in action on my laptop; a screen recording of htop so as “to prove” that it exists, “really,” even though still in development. (“Nearly there!!!”)
- Key point is the main process has children – forked, and not double-forked – and the children are not owned by root but by actual users on the system
- Tree is *enforced* – respect the admin, always.

# How do we *do* it?



- Admin writes the main configuration
- Configuration will be run through the Jinja template engine
- Admin may export variables via the usual Jinja syntax (`{{ variable_name|filter1|filter2|... }}`)
- User need only specify the variables *at submission time*.

# Admin Writes/Controls



```
engine:
  type: GlobusComputeEngine

provider:
  type: SlurmProvider
  partition: cpu
  account: {{ ACCOUNT_ID }}

launcher:
  type: SrunLauncher

walltime: {{ walltime|default("00:30:00") }}
```

**user\_config\_template.yaml**



- Admin writes the main configuration
- Configuration will be run through the Jinja template engine
- Admin may export variables via the usual Jinja syntax ({{ variable\_name|filter1|filter2|... }})
- User need only specify the variables *at submission time*.

# Admin Writes/Controls

# User Script



```
engine:
  type: GlobusComputeEngine

provider:
  type: SlurmProvider
  partition: cpu
  account: {{ ACCOUNT_ID }}

launcher:
  type: SrunLauncher

walltime: {{ walltime|default("00:30:00") }}
```

**user\_config\_template.yaml**

```
import globus_compute_sdk as GC

uep_conf = {
  "ACCOUNT_ID": "314159265",
  "walltime": "00:02:00"
}

with GC.Executor(
  endpoint_id=mep_id,
  user_endpoint_config=uep_conf
) as gce:
  fut = gce.submit(some_func)
  res = fut.result()
```



- Admin writes the main configuration
- Configuration will be run through the Jinja template engine
- Admin may export variables via the usual Jinja syntax ({{ variable\_name|filter1|filter2|... }})
- User need only specify the variables *at submission time*.

# Admin Writes/Controls

# User Script



```
engine:
  type: GlobusComputeEngine

provider:
  type: SlurmProvider
  partition: cpu
  account: {{ ACCOUNT_ID }}

launcher:
  type: SrunLauncher

walltime: {{ walltime|default("00:30:00") }}
```

**user\_config\_template.yaml**

```
import globus_compute_sdk as GC

uep_conf = {
  "ACCOUNT_ID": "543126688"
}

with GC.Executor(
  endpoint_id=mep_id,
  user_endpoint_config=uep_conf
) as gce:
  fut = gce.submit(some_func)
  res = fut.result()
```



- The user still needs to be aware of the configuration pieces of interest
- "The abstraction is still leaky!"
  - But less leaky.
  - The user needs to know about less (SlurmProvider can be ignored by user; only account\_id matters)
- Key point: configuration of interest is *closer to the SDK codes that user them*
  - Not attached to an opaque uuid identifier
- Side note: observe that the admin *can* specify defaults, meaning the user need not specify ALL variables. Just the required one.
  - N.B. if the user does not supply account\_id, then the UEP would still start up (valid YAML to have an empty account\_id) but the submission would fail

# Two different configurations; same user!



```
2725570 root      126M S 0.6
2800689 kevin     145M S 1.9
2800700 kevin     125M S 1.3
2800716 kevin     124M S 0.0
2801135 kevin     145M S 0.0
2801155 kevin     125M S 1.3
2801171 kevin     124M S 0.0
```

```
├─ Globus Compute Endpoint *(290ffb16-c7f2-4799-a314-f4fd67787edd, test_mt) - l
├─ Globus Compute Endpoint (8574f3e9-01c4-5628-a6c2-b2b169d3731f, uep.290ffb
├─ parsl: HTEX interchange
├─ Globus Compute Endpoint (8574f3e9-01c4-5628-a6c2-b2b169d3731f, uep.290
├─ user_config_b ── Globus Compute Endpoint (3dc6c69f-0221-7291-98a2-b67fcc23d411, uep.290ffb
├─ parsl: HTEX interchange
├─ Globus Compute Endpoint (3dc6c69f-0221-7291-98a2-b67fcc23d411, uep.290
```



# Value-Add for Users



# Value-Add for Users



- No need to maintain multiple endpoints for different configurations



# Value-Add for Users



- No need to maintain multiple endpoints for different configurations
- Specify needs at task submission



# Value-Add for Users



- No need to maintain multiple endpoints for different configurations
- Specify needs at task submission
- No need to log in to the terminal



# Value-Add for Site Administrators



# Value-Add for Site Administrators



- Templatable User Endpoint Configurations (Jinja)
  - e.g., pre-choose SlurmProvider, PBSProvider; enforce limits



# Value-Add for Site Administrators



- Templatable User Endpoint Configurations (Jinja)
  - e.g., pre-choose SlurmProvider, PBSProvider; enforce limits
- No orphaned user compute endpoints
  - Enforced process tree
  - Idle-endpoints are shutdown (per template configuration)



# Value-Add for Site Administrators



- Templatable User Endpoint Configurations (Jinja)
  - e.g., pre-choose SlurmProvider, PBSProvider; enforce limits
- No orphaned user compute endpoints
  - Enforced process tree
  - Idle-endpoints are shutdown (per template configuration)
- Standard Globus Identity Mapping



# Value-Add for Site Administrators



- Templatable User Endpoint Configurations (Jinja)
  - e.g., pre-choose SlurmProvider, PBSProvider; enforce limits
- No orphaned user compute endpoints
  - Enforced process tree
  - Idle-endpoints are shutdown (per template configuration)
- Standard Globus Identity Mapping
- **Lower barrier for users**



# Current status



- We're buttoning up a few details
- Have not yet written any documentation
- Looking for brave volunteers to give it go



# Thank You!



- **Questions?**
- **Comments?**
- **Synergistic thoughts?**

