HTCondor Week 2018

Prioritizing vanilla and grid jobs from local users on a Tier-3 condor cluster

Kenyi Hurtado
Université de Notre Dame du Lac

May 22, 2018

Introduction

- The Notre Dame CMS group operates an HTCondor cluster at the scale of +1K cores.
  - We additionally have +20K cores on campus that we use opportunistically, (but that's a different story).

Introduction

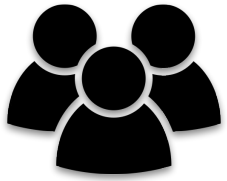
- Users in this group at Notre Dame submit jobs to the Tier-3 in two different ways:
  - Locally, through vanilla jobs

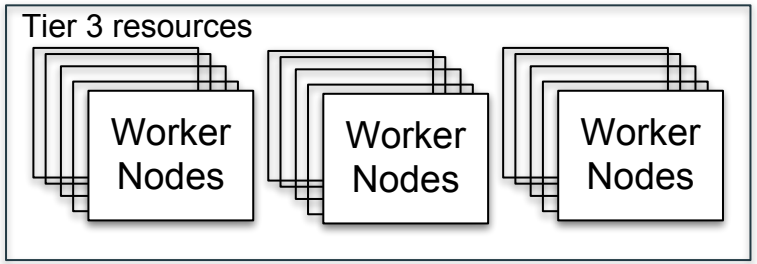
Introduction

- Users in this group at Notre Dame submit jobs to the Tier-3 in two different ways:
  - Locally, through vanilla jobs
  - Through the "grid"
    - Actually, not (directly) using the grid universe.
    - Rather, users submit jobs off-campus to a global pool factory that then submits glideins to the pool through an HTCondor-CE at ND.
      - So, the factory takes care of the grid submission.
      - (See previous presentation for details).

ND CMS Tier 3 pool

ND users

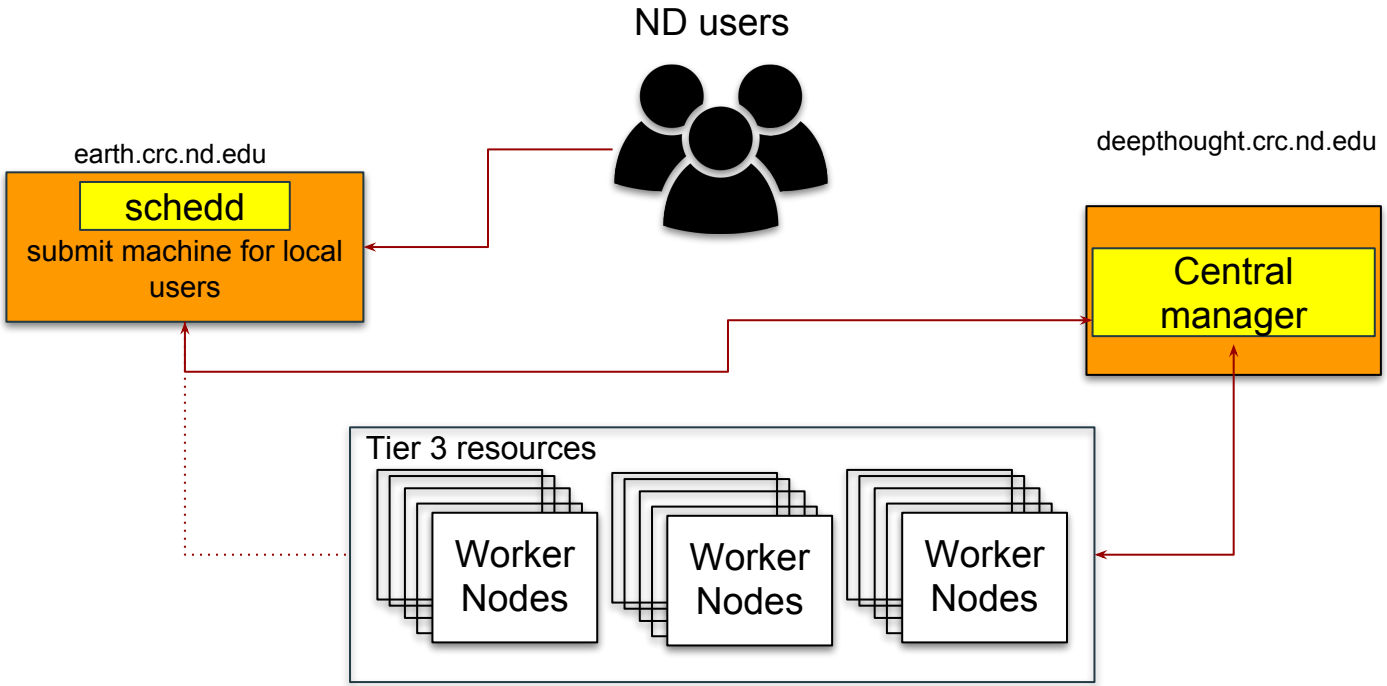




1,344 cores

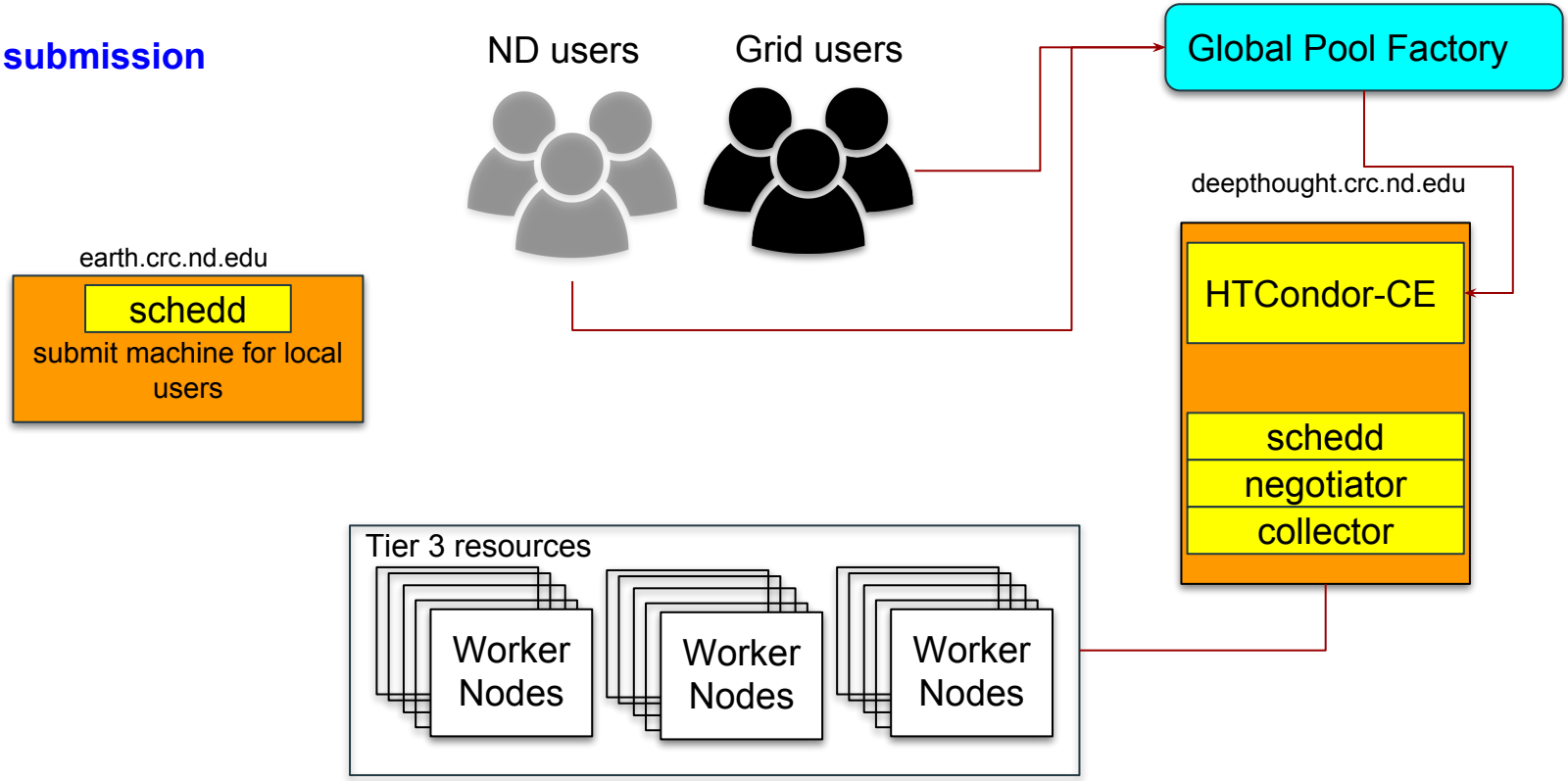
ND CMS Tier 3 pool

Local submission

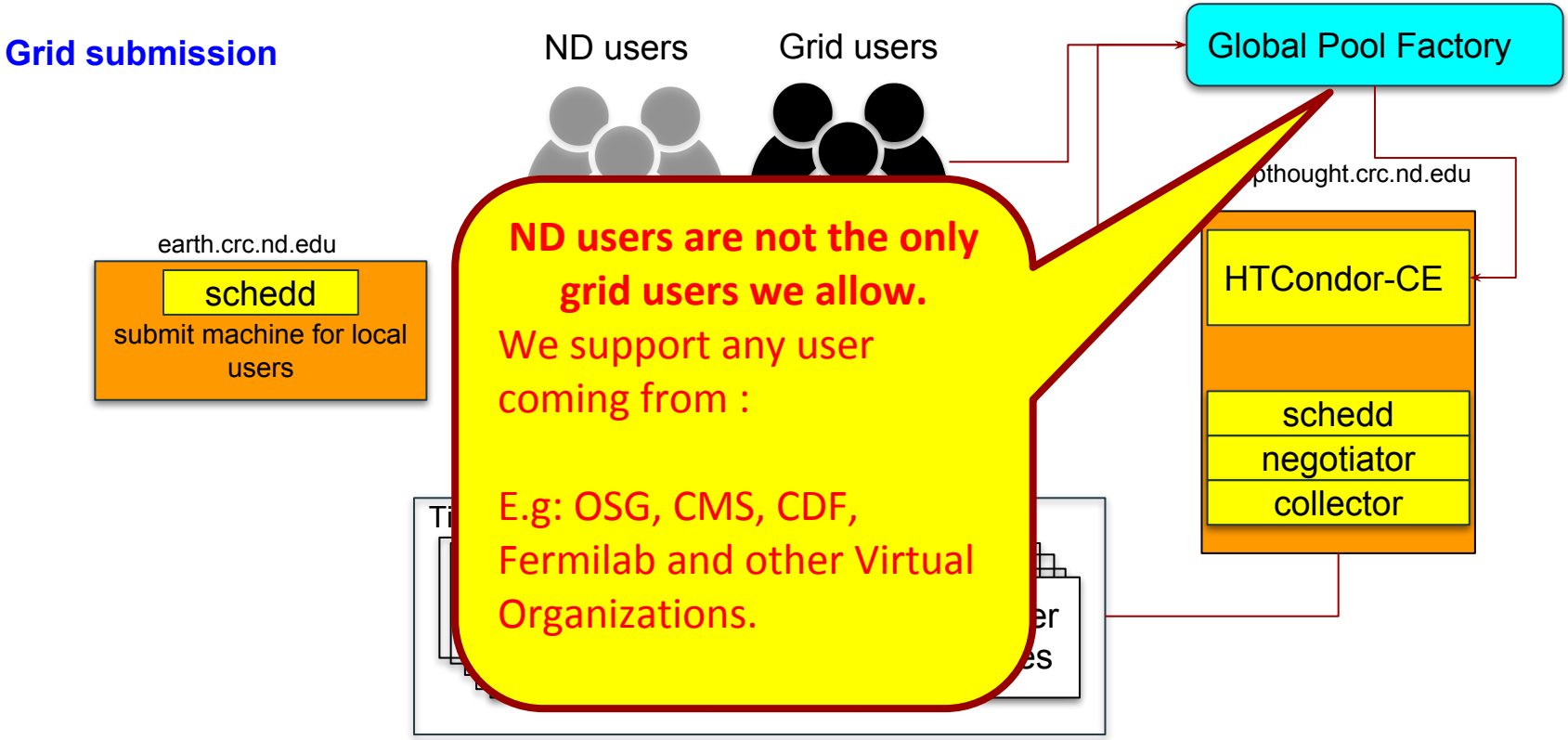


ND CMS Tier 3 pool

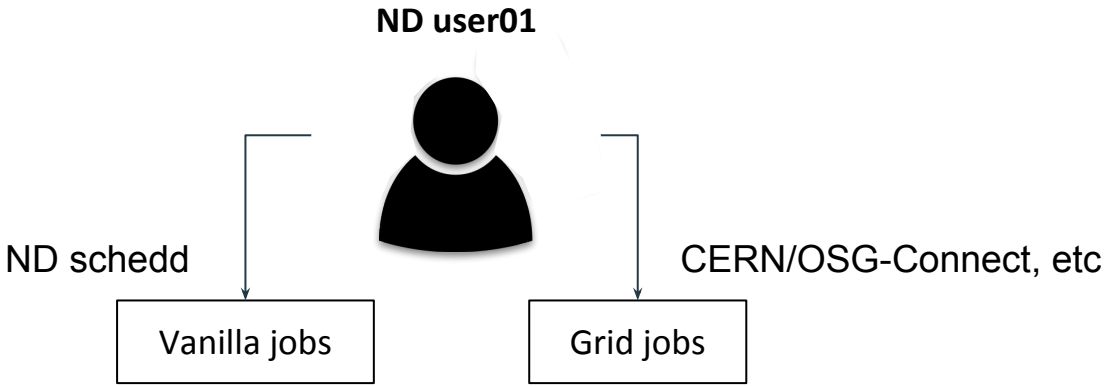
Grid submission



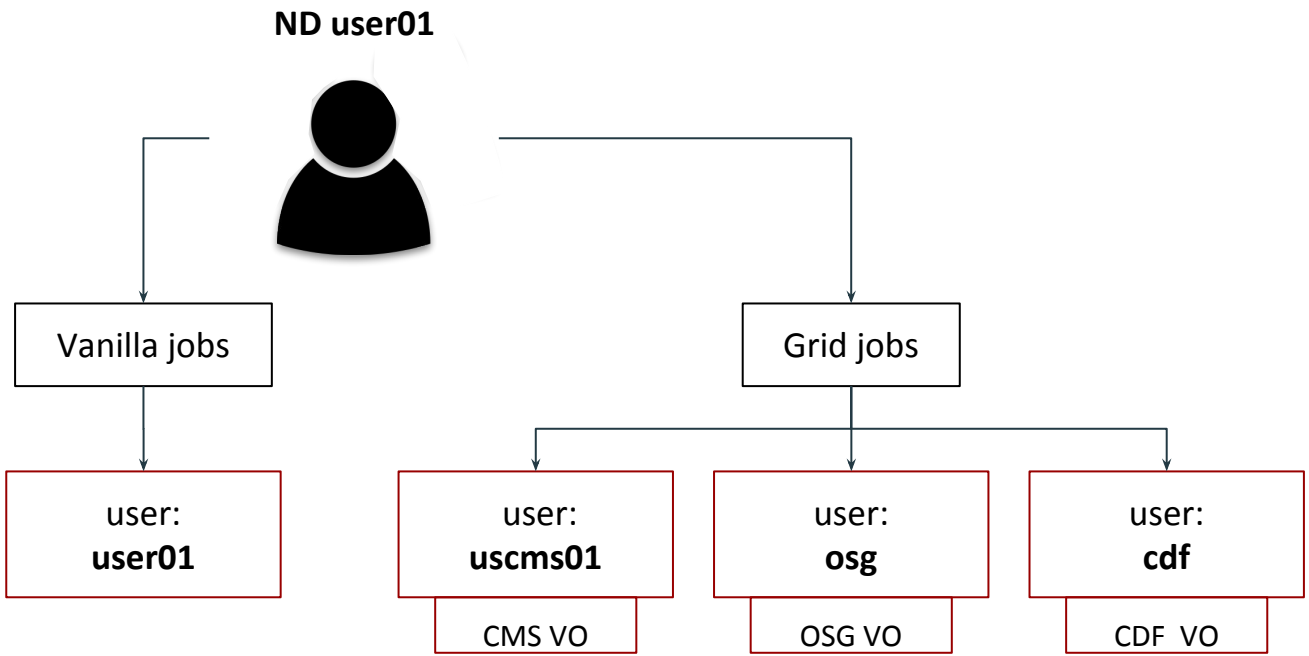
ND CMS Tier 3 pool



Job Owners (Payload)

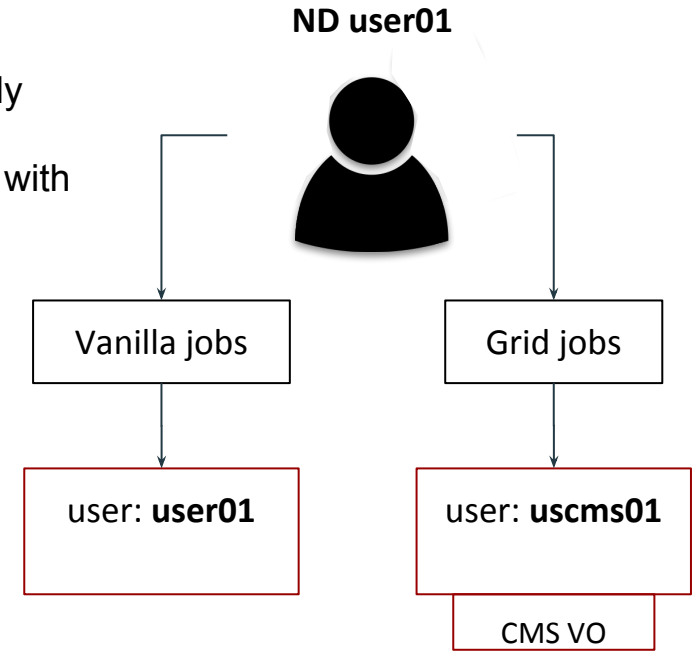


Job Owners (Payload)

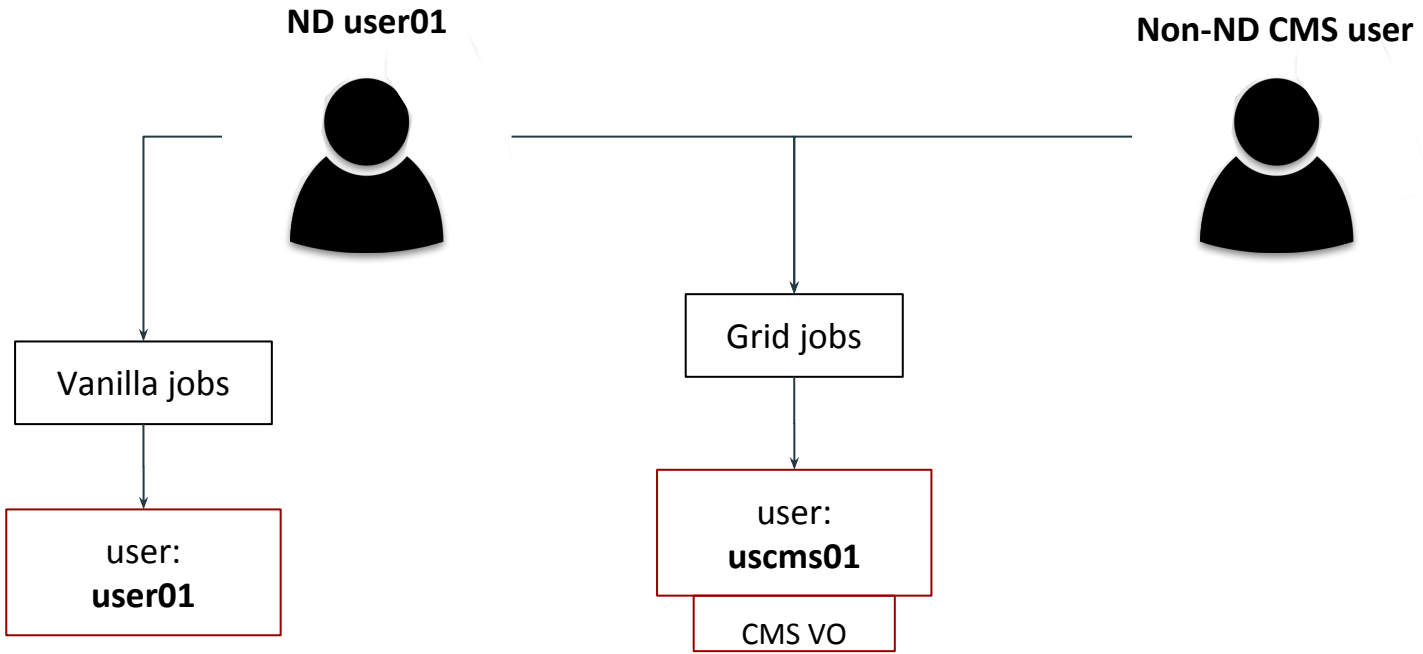


Job Owners (Payload)

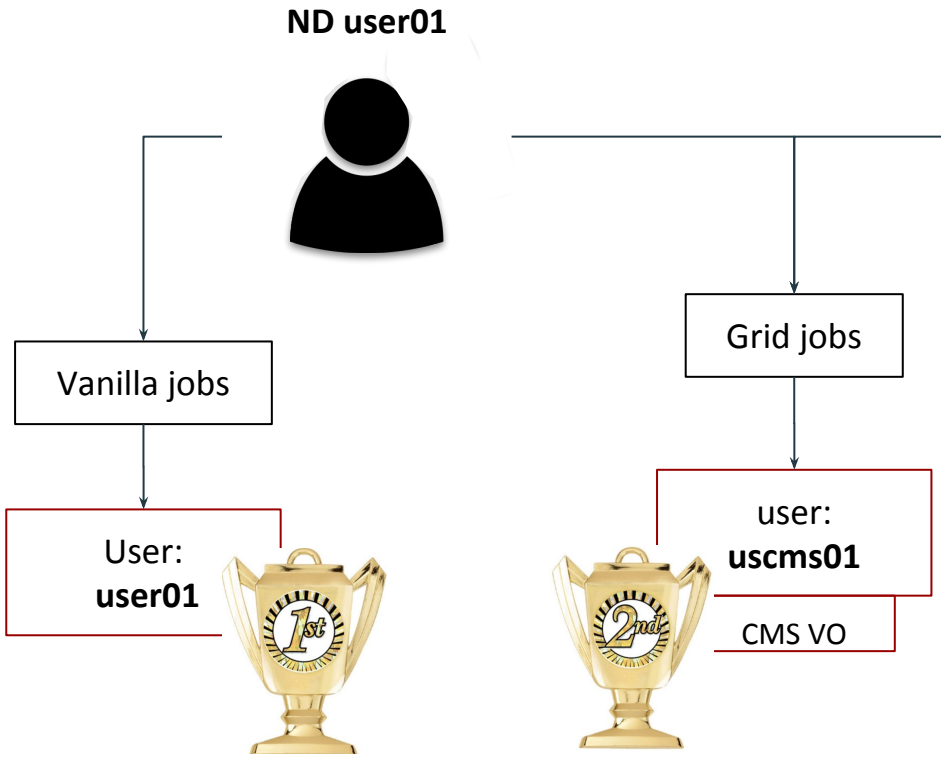
In practice, our users only submit either locally or through through the grid with CMS VO



So, who runs first?

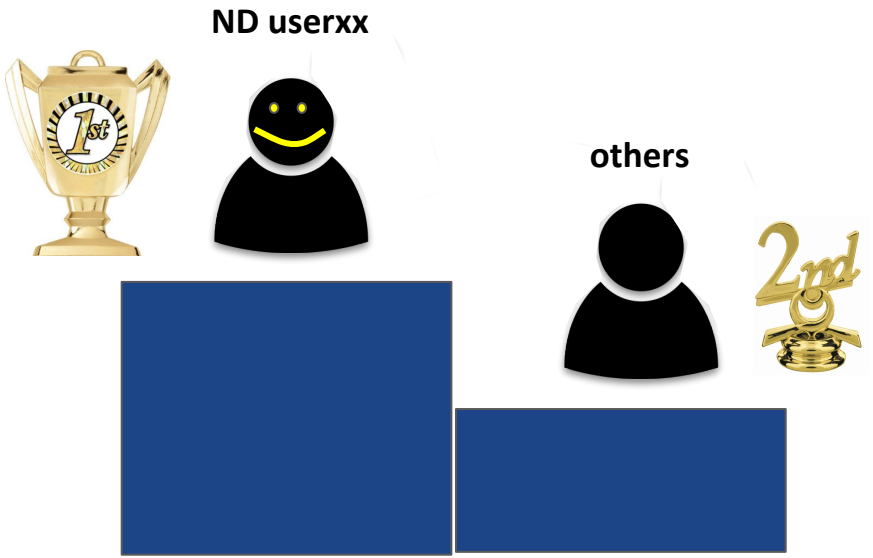


So, who runs first?

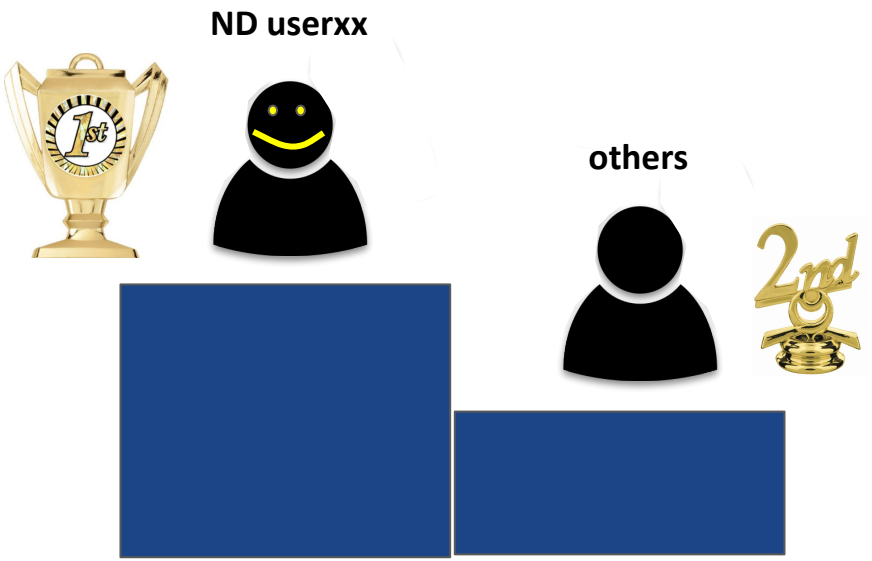


- Both ND user01 and cmsuser01 would have the same priority while running through the grid.
- User01 would have better priority with vanilla jobs.

What we want in terms of resource priorities?



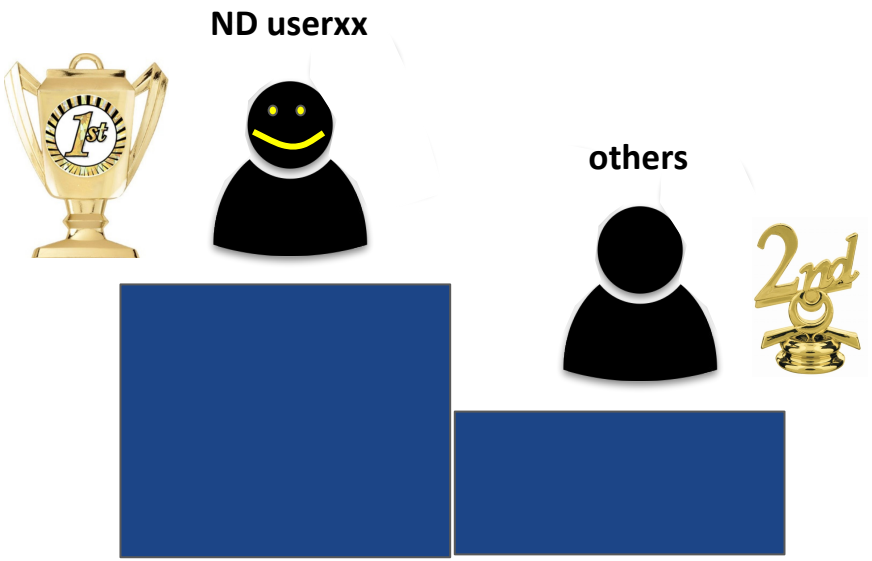
What we want in terms of resource priorities?



ND group should have priority on resources.

1. If a resource is taken by others, ND-user jobs should be able to take over those resources at all times
2. ND users can't preempt between each other.

What we want in terms of resource priorities?



ND group should have priority on resources.

1. If a resource is taken by others, ND-user jobs should be able to take over those resources at all times.
2. ND users can't preempt between each other.

Solution alternatives?

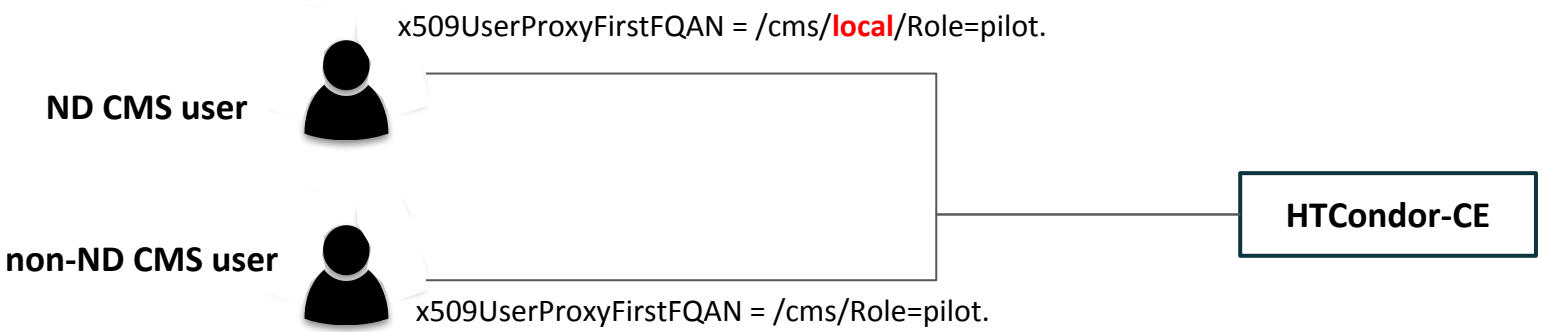
1. Accounting groups
   - Allows to put all ND users in a specific group to give it better priority.
2. Preemption
   - To replace a job with one with better priority.
3. Defragmentation
   - Drain machines periodically so jobs with better priority can take over.

Problem: How to discriminate between CMS users coming from ND or not? Remember both submit using the same user account.

Solution alternatives?

Problem: How to discriminate between CMS users coming from ND or not? Remember both submit using the same user account.

- HTCondor-CE to the rescue!
  - ND vs non-ND CMS users use a different grid proxy



Solution alternatives?

Problem: How to discriminate between CMS users coming from ND or not? Remember both submit using the same user account.

- HTCondor-CE to the rescue!
  - ND vs non-ND CMS users use a different grid proxy
    - Can map ND users to a different user account (e.g uscms01_local)
      - Requires creating another account

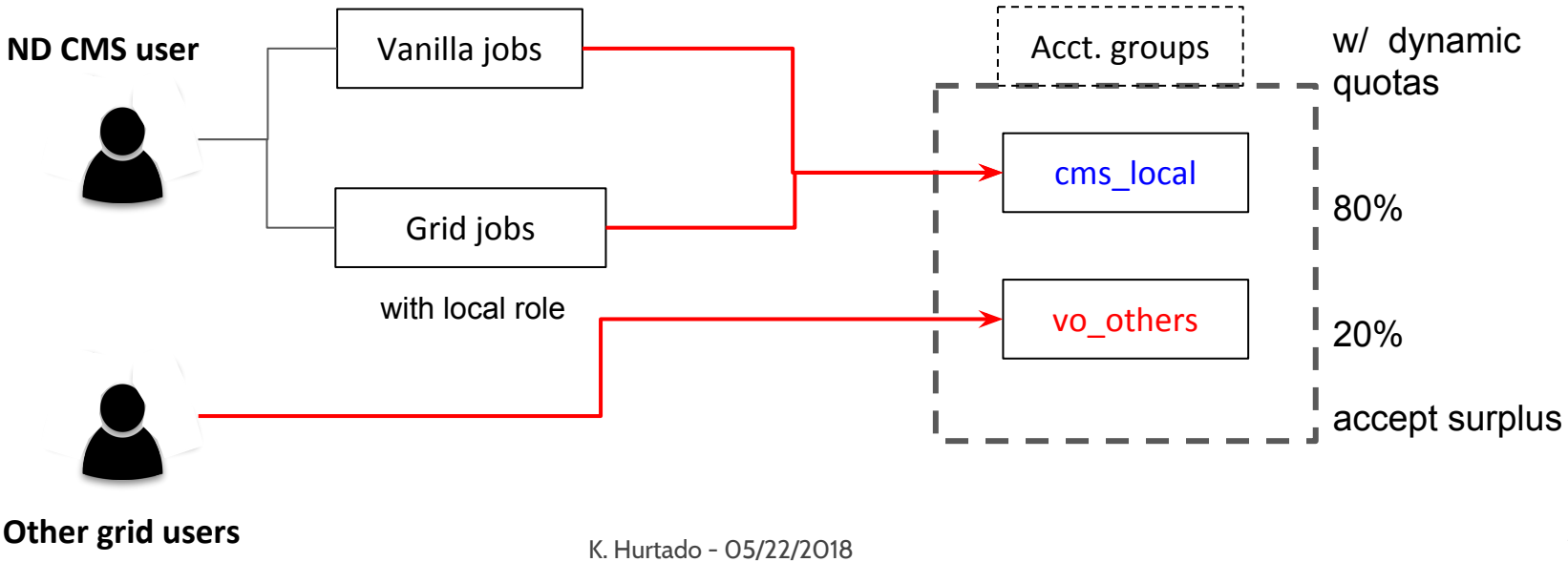
Solution alternatives?

Problem: How to discriminate between CMS users coming from ND or not? Remember both submit using the same user account.

- HTCondor-CE to the rescue!
  - ND vs non-ND CMS users use a different grid proxy
    - Can map ND users to a different user account (e.g uscms01_local)
      - Requires creating another account
    - Can directly assign ND CMS jobs to a condor accounting group
      - All other ND user accounts can also be included in that group.

First solution attempt....

- Combine accounting groups and preemption
  - ND user accounts and grid jobs with "/cms/local." proxy role



First solution attempt….

- Combine accounting groups and preemption
  - ND user accounts and grid jobs with "/cms/local." proxy role
  - Make jobs that belong to the:
    - "vo_others" group preemptable.
    - "cms_local" group non preemptable
- Test completely filling the pool with grid jobs and submit a few vanilla and grid jobs (with local role) to see if they preempt properly.

Working! But something seems wrong…

```
-- Schedd: earth.crc.nd.edu : <x.x.x.x:9618?... @ 05/21/18 18:26:51
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
 256.0   khurtado        5/21 18:18   0+00:05:25 R  0    0.0 analisis.sh
 256.1   khurtado        5/21 18:18   0+00:04:23 R  0    0.0 analisis.sh
 256.2   khurtado        5/21 18:18   0+00:03:23 R  0    0.0 analisis.sh
 256.3   khurtado        5/21 18:18   0+00:02:22 R  0    0.0 analisis.sh
 256.4   khurtado        5/21 18:18   0+00:01:22 R  0    0.0 analisis.sh
 256.5   khurtado        5/21 18:18   0+00:00:23 R  0    0.0 analisis.sh
 256.6   khurtado        5/21 18:18   0+00:00:00 I  0    0.0 analisis.sh
 256.7   khurtado        5/21 18:18   0+00:00:00 I  0    0.0 analisis.sh
```

Wait, something is wrong… can you guess what?

```
-- Schedd                                    :9618?... @ 05/21/18 18:26:51
 ID                                      RUN_TIME ST PRI SIZE CMD
 256.0                                   +00:05:25 R  0    0.0 analisis.sh
 256.1                                   +00:04:23 R  0    0.0 analisis.sh
 256.2                                   +00:03:23 R  0    0.0 analisis.sh
 256.3                                   +00:02:22 R  0    0.0 analisis.sh
 256.4                                   +00:01:22 R  0    0.0 analisis.sh
 256.5                                   +00:00:23 R  0    0.0 analisis.sh
 256.6                                   +00:00:00 I  0    0.0 analisis.sh
 256.7                                   +00:00:00 I  0    0.0 analisis.sh
```

There are no other local users in this example. We should be ideally getting all jobs running at once (no minimum promised runtime for jobs in vo_others group)

Wait, something is wrong… can you guess what?

Wait, something is wrong… can you guess what?

```
-- Schedd                                  x.x.x.x:9618?... @ 05/21/18 18:26:51
 ID                                    TTED     RUN_TIME ST PRI SIZE CMD
 256.0                                 8:18   0+00:05:25 R  0    0.0 analisis.sh
 256.1                                 8:18   0+00:04:23 R  0    0.0 analisis.sh
 256.2                                 8:18   0+00:03:23 R  0    0.0 analisis.sh
 256.3                                 8:18   0+00:02:22 R  0    0.0 analisis.sh
 256.4                                 8:18   0+00:01:22 R  0    0.0 analisis.sh
 256.5                                 8:18   0+00:00:23 R  0    0.0 analisis.sh
 256.6                                 8:18   0+00:00:00 I  0    0.0 analisis.sh
 256.7                                 8:18   0+00:00:00 I  0    0.0 analisis.sh
```

The negotiator in our pool starts a new cycle every minute (the default value in condor)

Wait, something is wrong…

Problem: Only 1 preemption per negotiation cycle. Can take hours to take the whole pool (~3 to 24h, multi-core dependant).

Wait, something is wrong…

Problem: Only 1 preemption per negotiation cycle. Can take hours to take the whole pool (~3 to 24h, multi-core dependant).

Solution alternatives:

1. Report problem to HTCondor team
   - Team identified this as a bug in the Negotiatior
   - Fix in progress...
   - … but fixing this is not trivial and takes time (months-term).

2. Look for alternatives in the meantime.

Looking for alternatives

- Defragmentation
  - Done periodically
  - Drain whole X machines per hour.
  - Doesn't look at demand but startds.
  - Need to evaluate
  - So, it's a good general solution, but not as fast as we wanted.

Looking for alternatives

- Try to fill whole machines with jobs from one group or the other (segregation?) :
  - Change NEGOTIATOR_PRE_JOB_RANK to sort resources matching a request in a way that ND CMS jobs prefer to run on machines already running other jobs in that acct. group.
  - Make Machine RANK to prefer vanilla & grid jobs with local role.

Looking for alternatives

- Try to fill whole machines with jobs from one group or the other (segregation?) :
  - Change NEGOTIATOR_PRE_JOB_RANK to sort resources matching a request in a way that ND CMS jobs prefer to run on machines already running other jobs in that acct. group.
  - Make Machine RANK to prefer vanilla & grid jobs with local role.
- Use a basic draining on demand method
  - Write script with python bindings to query demand of vanilla/grid local jobs periodically (each ~15 minutes)
  - Calculate number of nodes needed
  - Drain X machines

Looking for alternatives

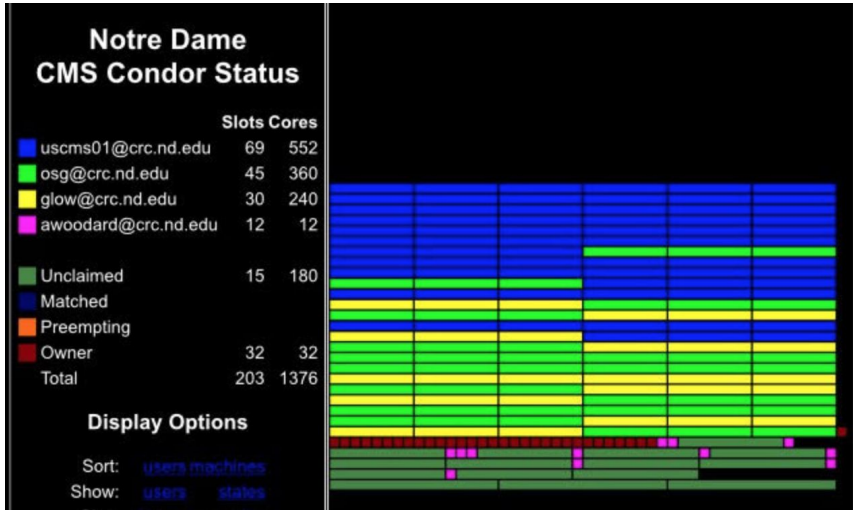
- Try to fill whole machines with jobs from one group or the other (segregation?) :
- Use a basic draining on demand method
- Keep preemption in place too (actually faster for small number of local jobs).

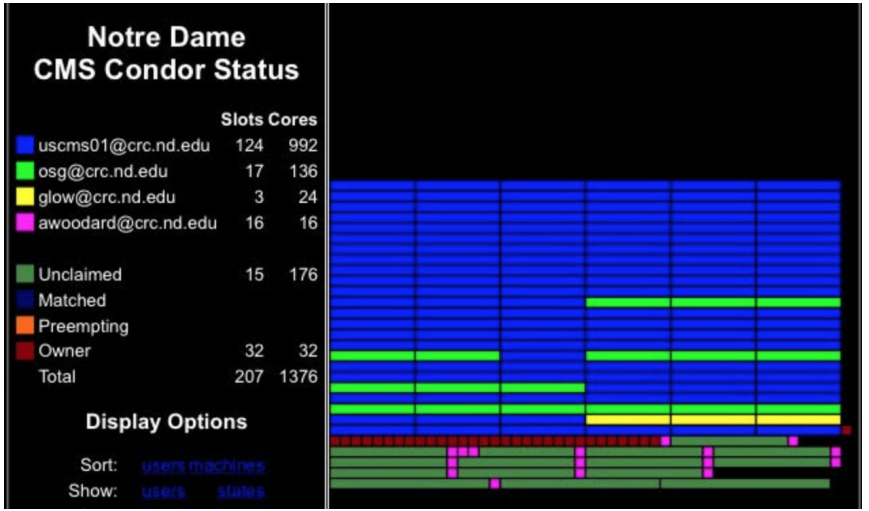
Can't replace a bug-free slot-based preemption solution, but it works fine at our scale and for our needs.

Reduces waiting time for ND users from hours to minutes (to take over the whole pool if there enough local demand).

Looking for alternatives

(~10 minutes later)





Conclusions

- Python bindings, accounting groups, preemption and draining of machines are powerful tools/features in condor.

- This presentation illustrates a use-case in which we happen to use all of that.
  - Looking forward to get the negotiator issue fixed, so draining+python bindings become unnecessary though.