HPC User Group

Meeting

2020-01-20

Jan Moren

Scientific Computing and
Data analysis section

Say hello to

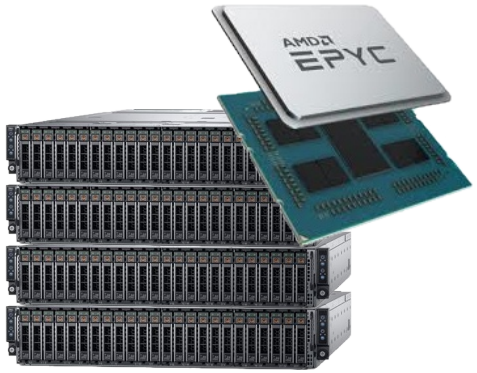
DE·i·GO

High-core AMD and Intel nodes

next-generation networking

ultra-high speed storage

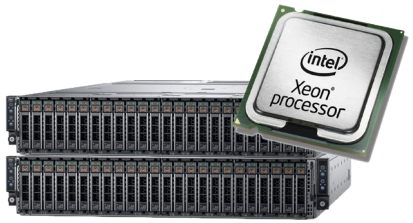
DE·i·GO



456 AMD nodes

2 x EPYC 7702 2.0GHz
128 cores
512GB memory

58368 cores



192 Intel nodes

2 x Xeon 6230 2.1GHz
40 cores
512GB memory

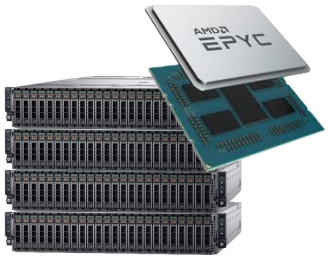
7680 cores

Sango: 9600 cores

Deigo: 66048 cores

688% more!

DE·i·GO



456 AMD nodes

2 x EPYC 7702 2.0GHz
128 cores
512GB memory

58368 cores - 88%

192 Intel nodes

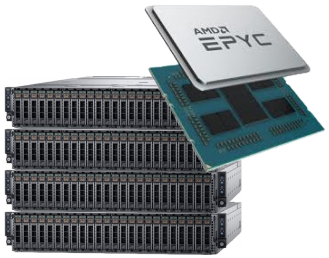
2 x Xeon 6230 2.1GHz
40 cores
512GB memory

7680 cores - 12%

Why both Intel and AMD?

- Per *core*:
  - AMD is a bit faster for integer, I/O
  - Intel is a bit faster FPU (esp. AVX512)
  - Depends *a lot* on your code
- Per *node*:
  - AMD **trounces** Intel

DE·i·GO



456 AMD nodes

2 x EPYC 7702 2.0GHz
128 cores
512GB memory

58368 cores - 88%

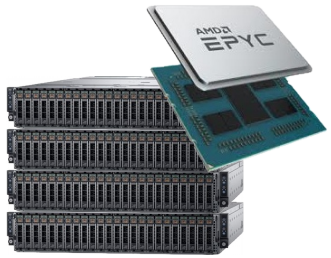
192 Intel nodes

2 x Xeon 6230 2.1GHz
40 cores
512GB memory

7680 cores - 12%

Why both Intel and AMD?

- A (very) few old, closed-source apps may run only on Intel
- A few apps don't need lots of cores and take very good advantage of AVX512
- Intel MKL runs really slow on AMD



456 AMD nodes

2 x EPYC 7702 2.0GHz
128 cores
512GB memory

58368 cores - 88%

192 Intel nodes

2 x Xeon 6230 2.1GHz
40 cores
512GB memory

7680 cores - 12%

Why both Intel and AMD?

used to run

- Intel MKL ~~runs~~ really slow on AMD
- MKL checks CPU maker, selects operations based on that.
- Can override this check with:

```
export MKL_DEBUG_CPU_TYPE=5
```

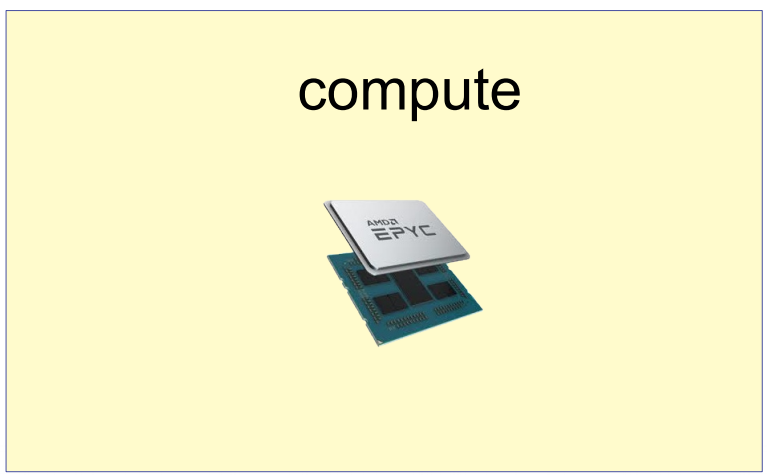
- MKL now up to 600% faster on AMD.

DE·i·GO

Work in Progress

AMD: 456 nodes, 88% cores

- General purpose compute partition
  - lots of cores, lots of users
  - **user memory and core limits**
- benefits from rebuilding your code

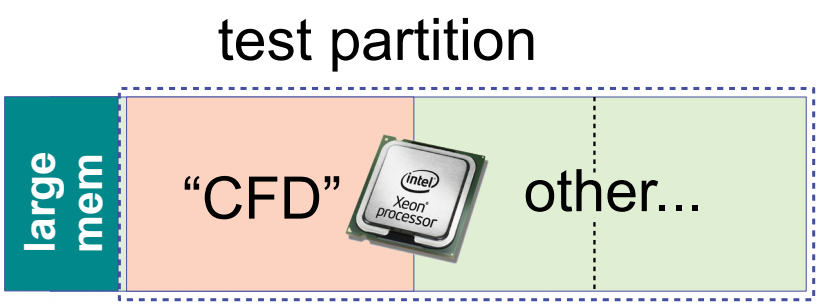


Memory and core limits

- Taking all 128 cores, or all 512GB memory waste a lot of resources

***Possible* policy:**

- ‘compute’ job limit: 64 cores and 128GB memory
- ‘fullnode’ partition can use 128/512GB

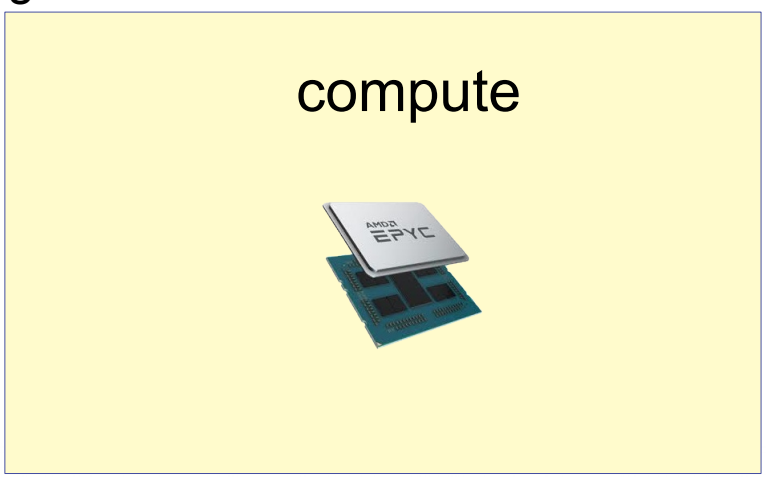


DE·i·GO

Work in Progress

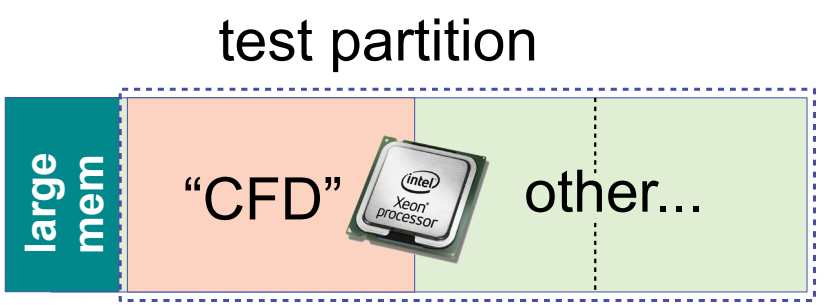
Intel: 192 nodes, 12% cores

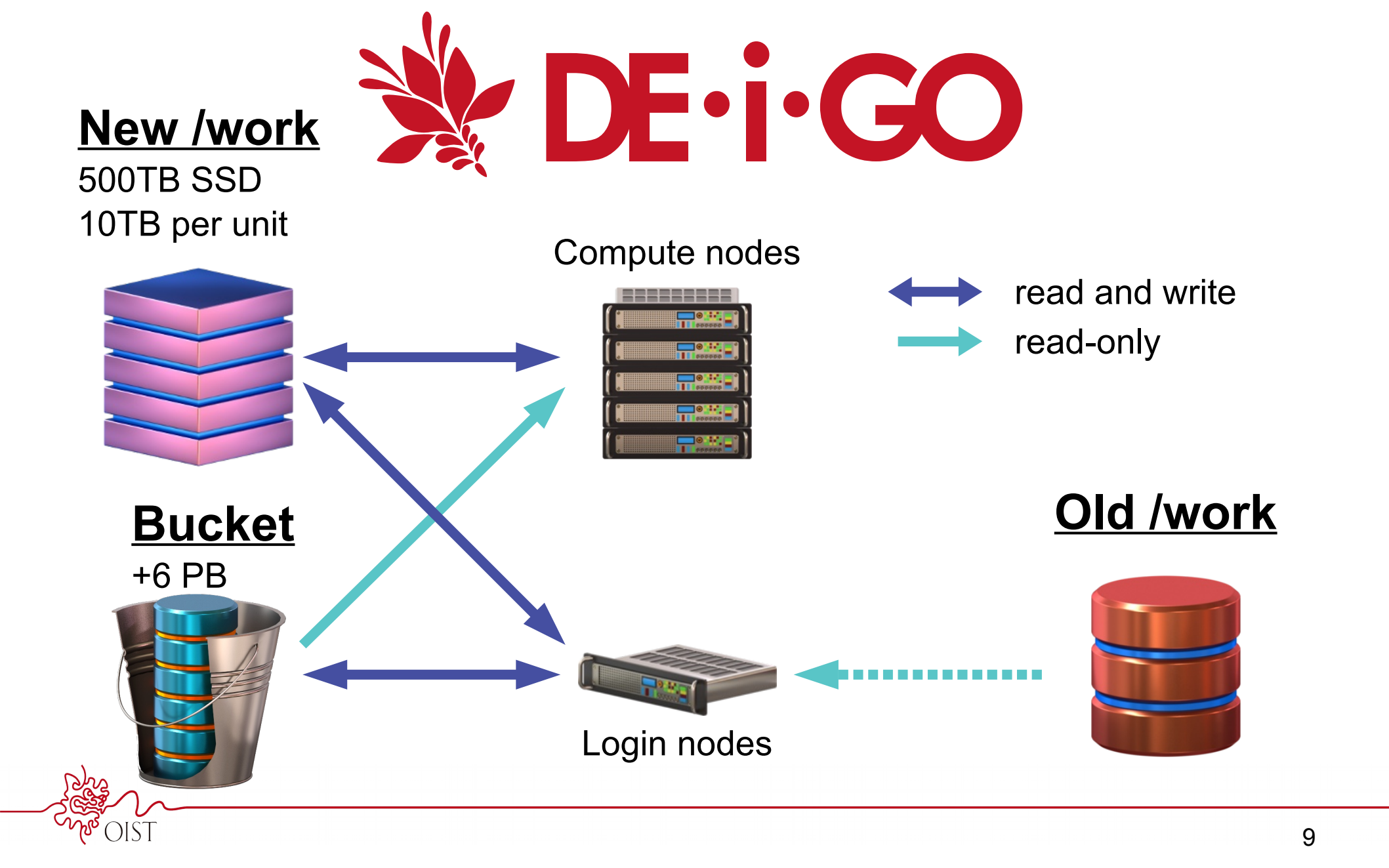
- Task-specific partitions
  - not per-user or per-unit
  - physics, MD
  - Intel-dependent code
  - largemem



Overlap with low-priority test partition

- Any user can use the test partition nodes for shorter computations.
- If a restricted task partition wants the node, the test code is stopped




New /work
500TB SSD
10TB per unit
Compute nodes
read and write
read-only
Bucket
+6 PB
Old /work
Login nodes

DE·i·GO

Sango (current) workflow:

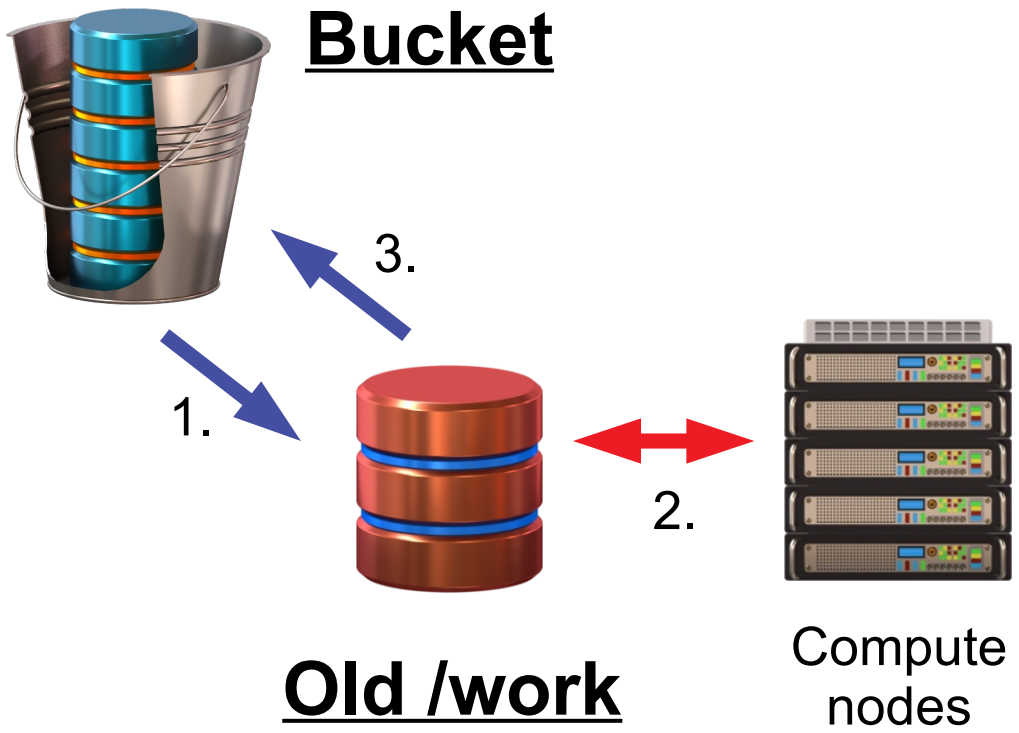
1. Copy data from Bucket to work
2. run calculation on Work only
3. Copy results back to Bucket

But:

- Inputs are large, take time to copy

→ lots of people leave data on Work

  - Work slows down
  - Lots of data ends up only on Work - where it's not backed up



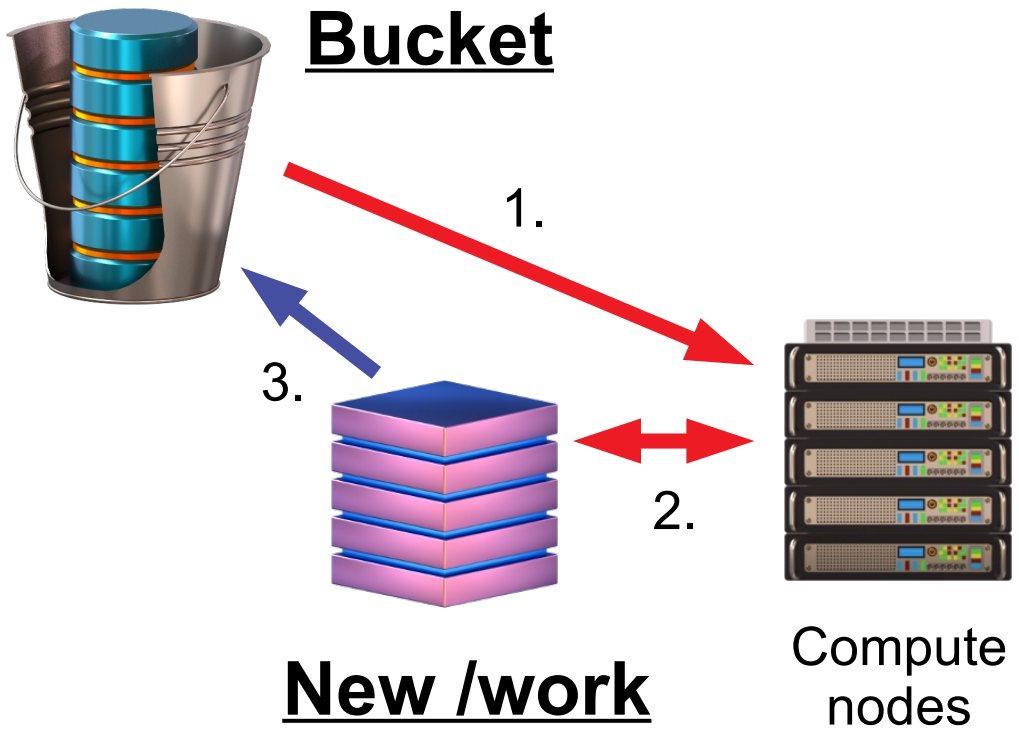
DE·i·GO

Deigo workflow:

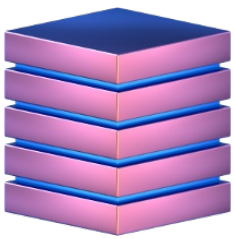
1. No preparation - use data directly from Bucket on compute nodes!
2. use /work for temporary storage
3. Copy results back to Bucket
4. Clean up /work

Why:

- Reading from Bucket is fast
- Work is *really* fast (and will stay fast)
- Data will stay backed up on Bucket



New /work
500TB SSD
10TB per unit



Compute nodes

Bucket
+6 PB

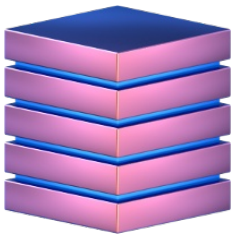
Login nodes

read and write
read-only

New workflow

- New /work:
  - /work is only scratch, **not storage**
  - 10TB per unit is a *hard limit*
- New network, new Bucket expansion
  - another 6PB storage
  - new network, new storage hardware means reading directly is fast and efficient
  - Writing would slow it down
    - → use /work for any writes during computation
    - → copy results to bucket at the end
    - → clean up /work

New /work
500TB SSD
10TB per unit



Compute nodes

Bucket
+6 PB

Login nodes

read and write
read-only

Old /work

- Will **disappear** during 2020
  - Soon out of warranty
  - Full, not expandable
- Will be available read-only
- You **must** copy data you need to Bucket
- The rest will be archived and effectively *unavailable* once shut down

Old /work

Provided software

- CentOS 8
- GCC 8 (or 9), AOCC
- BLIS, LibFLAME
- User Software (modules)
  - We will rebuild popular modules
  - Other modules will run directly or through “sango” container, available as module (best effort)

Your software

- For best results, rebuild with modern compilers, libraries
- Many will run OK unchanged
- Use “sango” container to run those that won’t:

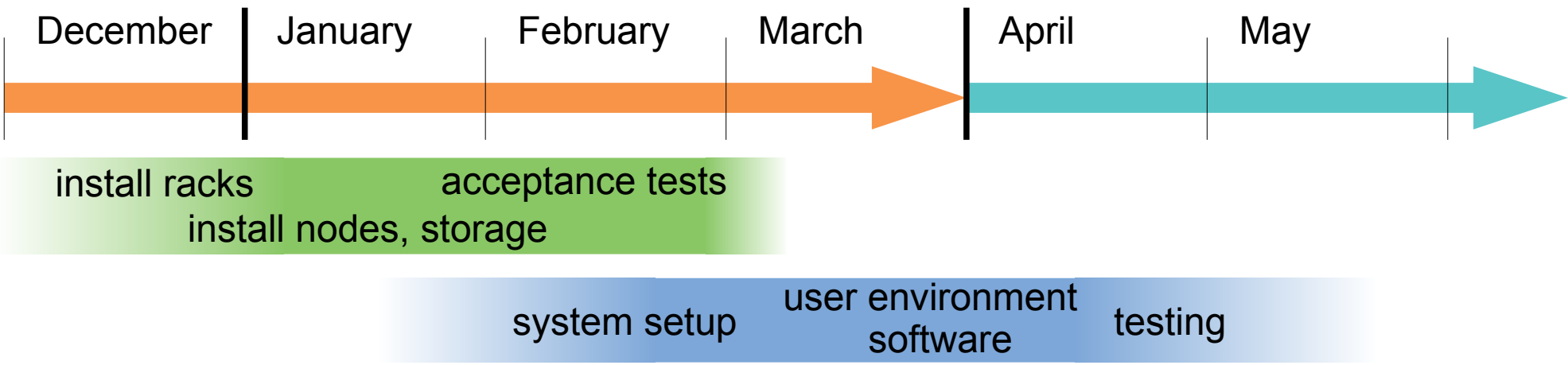
```
module load bowtie
myprog -o xyz
```

```
module load sango
sango -m bowtie myprog -o xyz
```

Preliminary

DE·i·GO

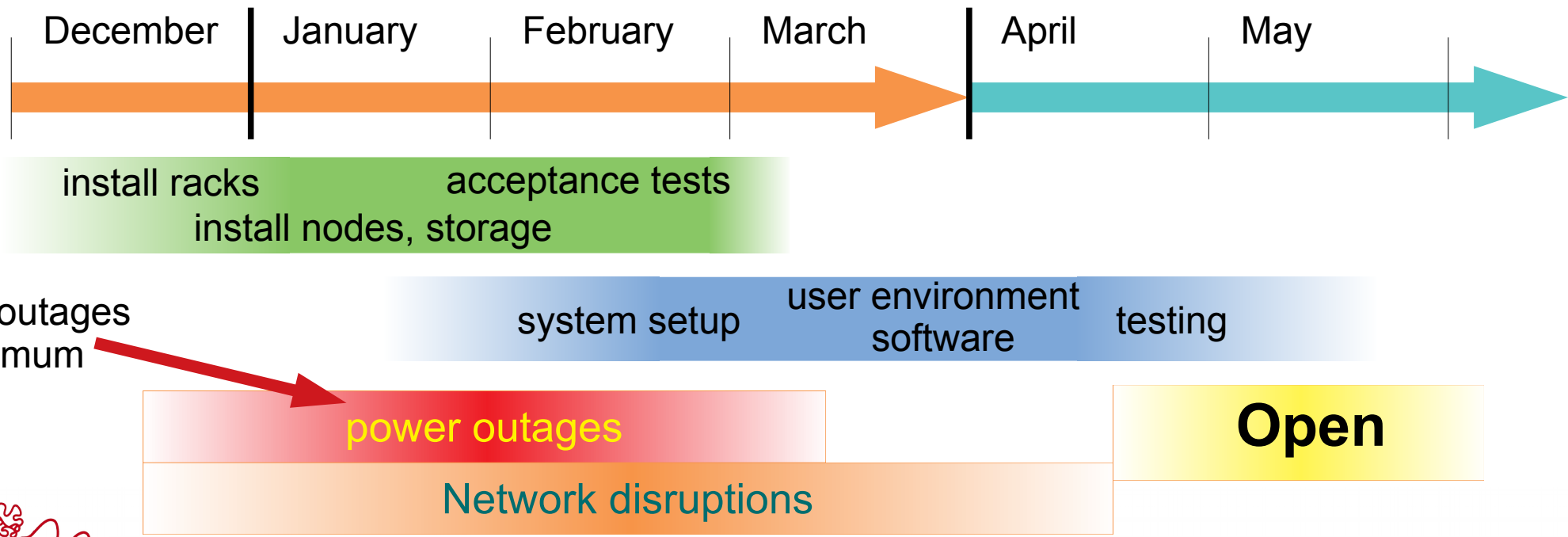
Timeline



Preliminary

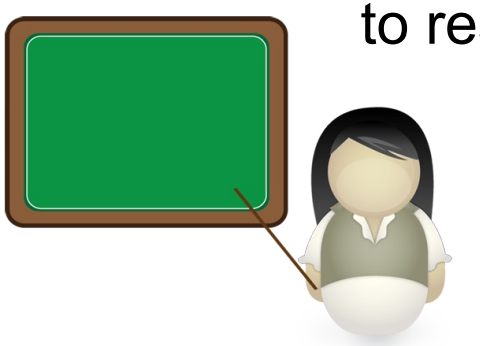
DE·i·GO

Timeline



PSA: Attribution

Scientific attribution and co-authorship rules apply to research support sections, including SCDA.



Research Support Division policy:

https://groups.oist.jp/rsd/rsd-attribution-policy

SCDA Policy:

https://groups.oist.jp/scs/attribution

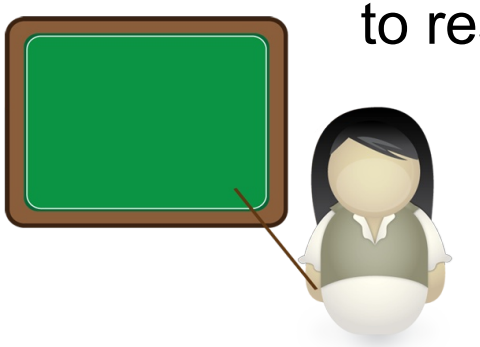
Why?

As Research Support sections we are evaluated on our contributions to OIST research

more attributions → more funding → more computing power!

PSA: Attribution

Scientific attribution and co-authorship rules apply to research support sections, including SCDA.



- If you used our systems for your research, we **require acknowledgement:**
  - You used our systems to generate data in your paper
  - You got our help for setting up computation, doing visualizations, setting up instruments and so on
- If we took an active part in the research process, we **require co-authorship:**
  - We did substantial work for you on analysis or visualization
  - Wrote substantial amount of code

You do *not* need to acknowledge us just for storing data or asking straightforward questions.

Questions, comments

Your turn, and your feedback