

Optimising HPC Workflows: Three case studies from a Research Software Engineer's perspective

Dr Jacob Newman

Research & Specialist Computing

University of East Anglia

Norwich, United Kingdom

Me: Research Software Engineer

- Previously a computing researcher
- Extensive user experience of HPC
- Advocate for HPC in research
- Appointed RSE in May 2021

Outreach
Optimisation



UEA's HPC: Ada

- Centos 7
- Slurm 19.05
- About 10,000 cores
- Across 173 CPU nodes
- 57 GPUs
- Including 30 Nvidia Quadro RTX6000s
- InfiniBand 100Gb/sec
- GPU/Compute/IB/Hi-Mem/Vis partitions



Our users

- About 1000 users
- Range of disciplines
- Simulations, modelling, machine learning, data processing, code development, etc.
- Associated levels of computing proficiency





3 Case Studies

User A

Senior lecturer in Chemistry

Researching molecular dynamics

GROMACS



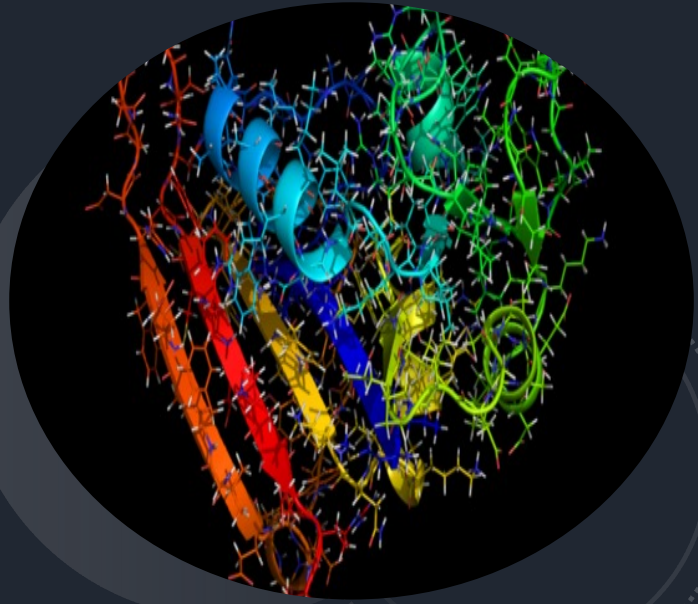
User A

“Currently I am trying to run molecular dynamics calculations using **gromacs** on the **compute** queues. These show **0.32 ns/day** on a shared node, **1.31 ns/day** on an exclusive node. Perhaps I am not using the queue properly because on my desktop **at home** (Intel i5-2500 @3.3GHz) I can achieve over **8 ns/day**. Anyway, **I need** a simulation of **200 ns** or so, hence I am wondering whether the other hardware available will offer better performance. If so, can I be given access to the appropriate queue. gromacs is **GPU aware** though whether this offers a significant improvement depends on the balance of calculations and the advice is to test the system.”

User A

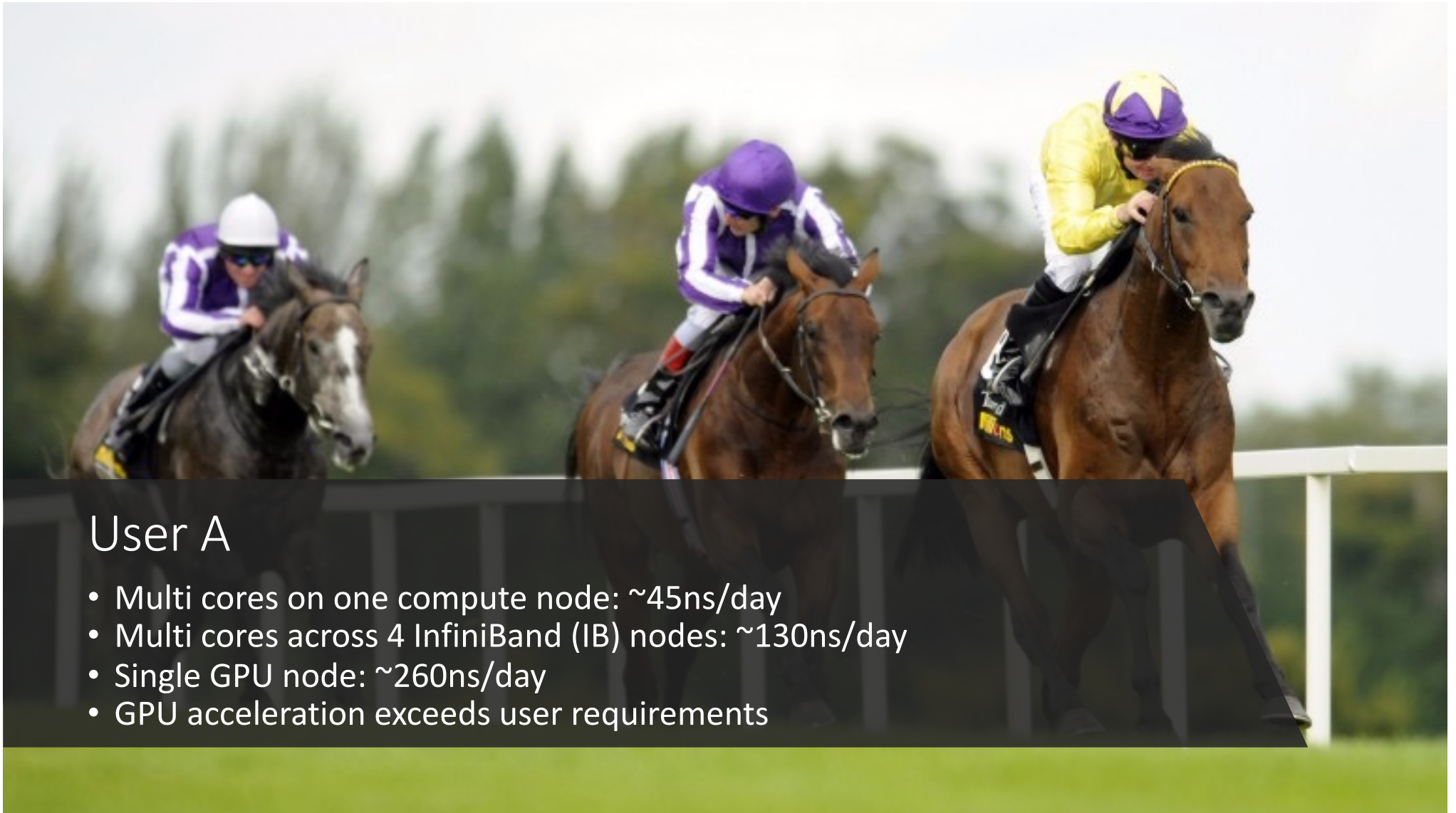
- Replicate results
- Now what?
- 2019.4
- 2020.4
- 34x speed increase
- ~45 ns/day, which is still not close to 200 ns/day

```
gromacs/2019.4/gcc (D)  
gromacs/2020.4/gcc-gpu-cuda11.0  
gromacs/2020.4/gcc-openmpi-eth (D)
```

MPI

GPU



User A

- Multi cores on one compute node: $\sim 45\text{ns/day}$
- Multi cores across 4 InfiniBand (IB) nodes: $\sim 130\text{ns/day}$
- Single GPU node: $\sim 260\text{ns/day}$
- GPU acceleration exceeds user requirements

User B

Postgraduate student in Computing

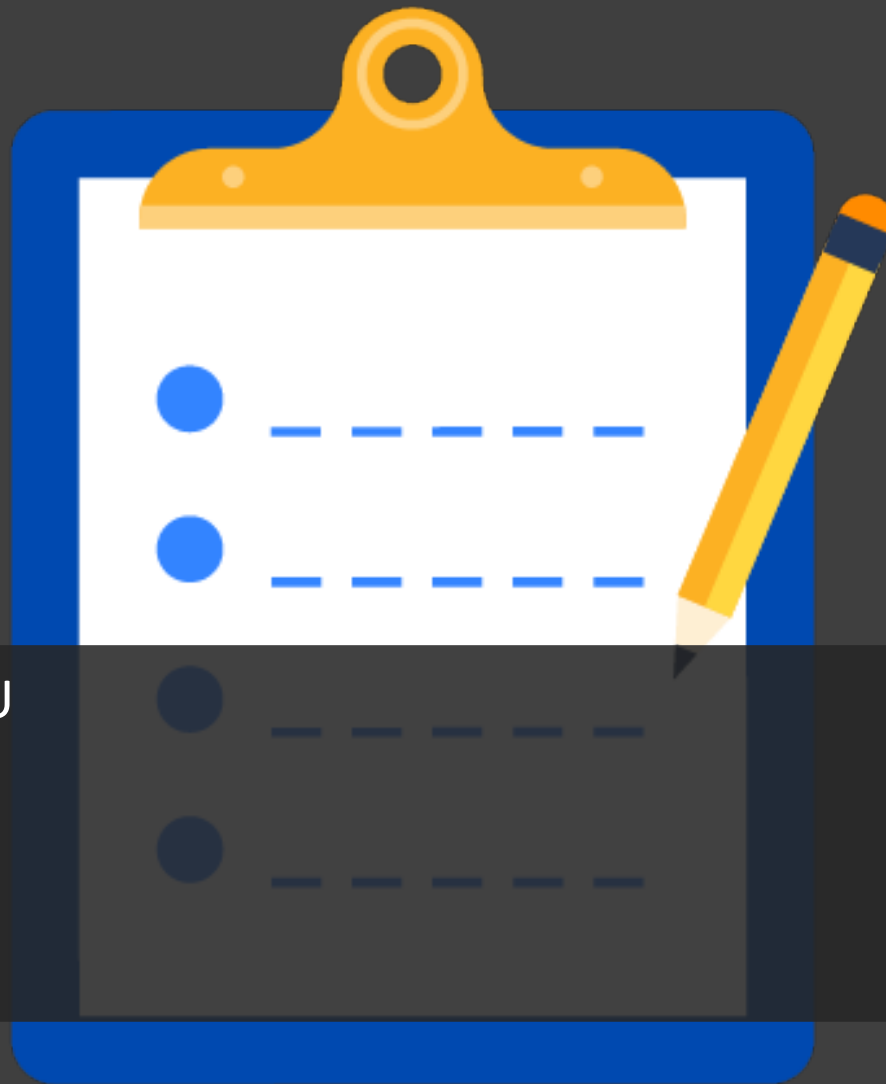
Time-series classification

TensorFlow



User B

Are there any way you can think of that might improve efficiency/speed up GPU jobs?



- Already using GPU
- Expert user
- TensorFlow
- Mixed precision



Mixed precision

```
from tensorflow.keras import mixed_precision
```

```
# The feature is experimental in TensorFlow < 2.4. >= 2.4  
it is a main feature.
```

```
mixed_precision.experimental.set_policy('mixed_float16')
```

```
# The output layer needs to be float32 for numerical  
stability
```

```
dense_1 = Dense(nclass, activation=activations.softmax,  
dtype='float32', name="dense_3_ptbdb")(dense_1)
```

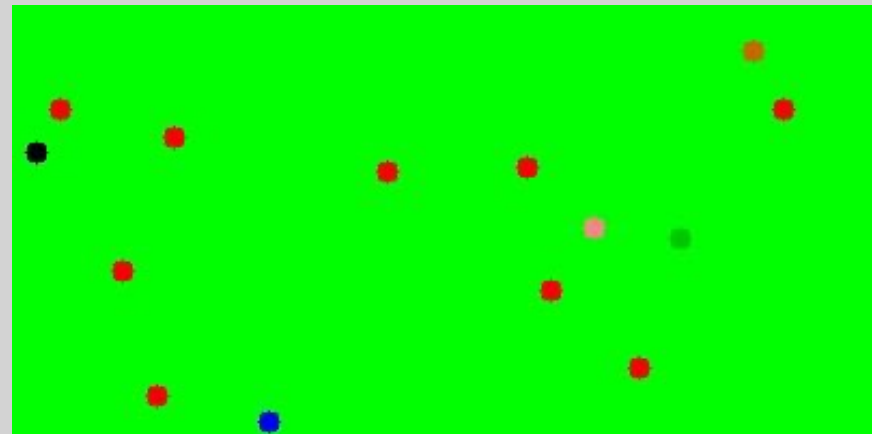
My own tests

Without the mixed policy...

```
Epoch 1/100  
1490/1490 [=====] -  
1124s 754ms/step
```

And with the mixed policy...

```
Epoch 1/100  
1490/1490 [=====] -  
485s 325ms/step
```



Did your test model policy job finish?

Are you running another with the output type corrected?

I've corrected the output type issue. But that optimization finished in 2.15 hours rather than 2.05

rather than a tick in the wall - suppose. You may find that some models work much better than others

That's about 14% improvement.

9×0.85

25/08/2021 10:50

👍 1

Which is over an hours improvement over the longer 9 hour problem - if the improvement doesn't scale too.

Type a new message

User C

Researcher in Biology

Quantifying bird abundance

R



User C

Hi Jake,

I hope you are well. I have been using the code you made for Claire to run some analysis on the cluster, however after 2 weeks it timed out.

I think the best idea is to try and split it up using the site index but I have no idea how to do 2 array indexes or if that is even possible? I have attached all the code. Do you have some time to talk about this soon?

Many thanks,
Cat

Current workflow

- For 16 datasets
 - For 15k sites
 - Process 330k records
- Takes at least two weeks to complete
- Current implementation submits a job for each dataset
- Bash script to submit an array job from idx 1 to 16
- Suggestion to process the 15k sites in smaller batches



New workflow

		Dataset ID		
		1	2	3
Site ID	1	Array ID 1	Array ID 2	Array ID 3
	2	Array ID 4	Array ID 5	Array ID 6
	3	Array ID 7	Array ID 8	Array ID 9



New workflow

```
#!/bin/bash
SBATCH_ARRAY_INX=$(ls ./datasets/*.csv | wc -l)
SBATCH_ARRAY_INX=$((SBATCH_ARRAY_INX*10))
sbatch --array=1-$SBATCH_ARRAY_INX pairwise_site.sh
```



New workflow

```
idx <- strtoi(args[1],base=10)
num_datasets <- strtoi(args[2],base=10)

dataset_idx <- ((idx-1)%num_datasets)+1
site_idx <- ceiling(idx/num_datasets)
```

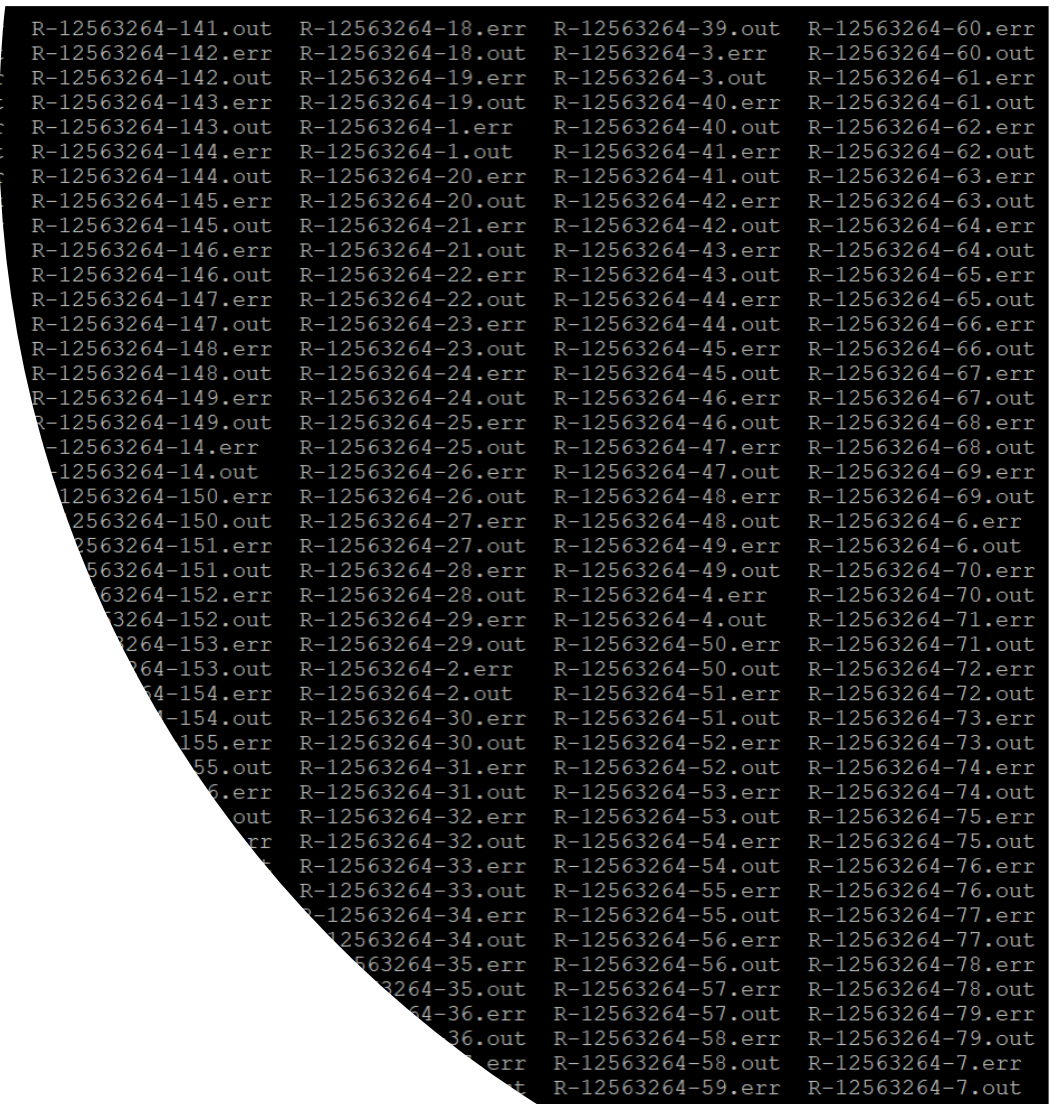
Worked example, where idx=19 and num_datasets=16:

$dataset_idx = ((19-1)\%16)+1 = 3$

$site_idx = ceiling(19/16) = ceiling(1.1875) = 2$

Results

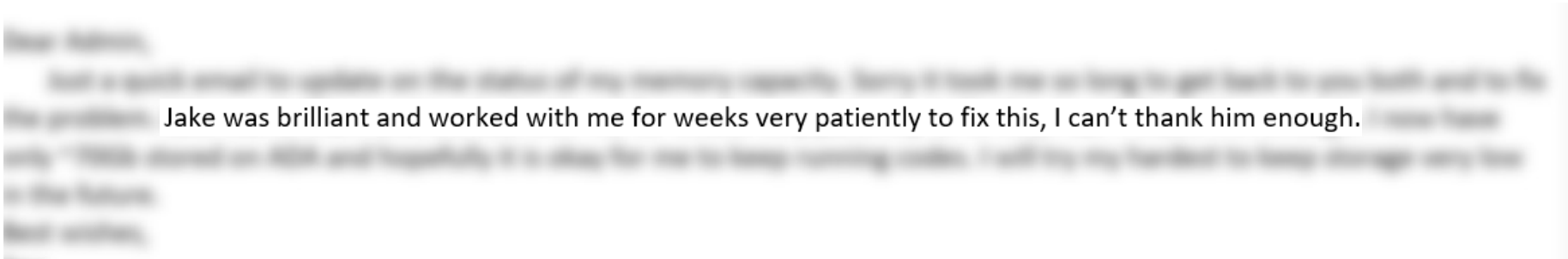
- Longest job completed in 4.5 days compared to "over two weeks"
- The cumulative time is 389 days
- But more work to be done...
- Could increase granularity
- Could ensure the jobs are distributed equally
- But in this case, five days is acceptable



R-12563264-141.out	R-12563264-18.err	R-12563264-39.out	R-12563264-60.err
R-12563264-142.err	R-12563264-18.out	R-12563264-3.err	R-12563264-60.out
R-12563264-142.out	R-12563264-19.err	R-12563264-3.out	R-12563264-61.err
R-12563264-143.err	R-12563264-19.out	R-12563264-40.err	R-12563264-61.out
R-12563264-143.out	R-12563264-1.err	R-12563264-40.out	R-12563264-62.err
R-12563264-144.err	R-12563264-1.out	R-12563264-41.err	R-12563264-62.out
R-12563264-144.out	R-12563264-20.err	R-12563264-41.out	R-12563264-63.err
R-12563264-145.err	R-12563264-20.out	R-12563264-42.err	R-12563264-63.out
R-12563264-145.out	R-12563264-21.err	R-12563264-42.out	R-12563264-64.err
R-12563264-146.err	R-12563264-21.out	R-12563264-43.err	R-12563264-64.out
R-12563264-146.out	R-12563264-22.err	R-12563264-43.out	R-12563264-65.err
R-12563264-147.err	R-12563264-22.out	R-12563264-44.err	R-12563264-65.out
R-12563264-147.out	R-12563264-23.err	R-12563264-44.out	R-12563264-66.err
R-12563264-148.err	R-12563264-23.out	R-12563264-45.err	R-12563264-66.out
R-12563264-148.out	R-12563264-24.err	R-12563264-45.out	R-12563264-67.err
R-12563264-149.err	R-12563264-24.out	R-12563264-46.err	R-12563264-67.out
R-12563264-149.out	R-12563264-25.err	R-12563264-46.out	R-12563264-68.err
R-12563264-14.err	R-12563264-25.out	R-12563264-47.err	R-12563264-68.out
R-12563264-14.out	R-12563264-26.err	R-12563264-47.out	R-12563264-69.err
R-12563264-150.err	R-12563264-26.out	R-12563264-48.err	R-12563264-69.out
R-12563264-150.out	R-12563264-27.err	R-12563264-48.out	R-12563264-6.err
R-12563264-151.err	R-12563264-27.out	R-12563264-49.err	R-12563264-6.out
R-12563264-151.out	R-12563264-28.err	R-12563264-49.out	R-12563264-70.err
R-12563264-152.err	R-12563264-28.out	R-12563264-4.err	R-12563264-70.out
R-12563264-152.out	R-12563264-29.err	R-12563264-4.out	R-12563264-71.err
R-12563264-153.err	R-12563264-29.out	R-12563264-50.err	R-12563264-71.out
R-12563264-153.out	R-12563264-2.err	R-12563264-50.out	R-12563264-72.err
R-12563264-154.err	R-12563264-2.out	R-12563264-51.err	R-12563264-72.out
R-12563264-154.out	R-12563264-30.err	R-12563264-51.out	R-12563264-73.err
R-12563264-155.err	R-12563264-30.out	R-12563264-52.err	R-12563264-73.out
R-12563264-155.out	R-12563264-31.err	R-12563264-52.out	R-12563264-74.err
R-12563264-156.err	R-12563264-31.out	R-12563264-53.err	R-12563264-74.out
R-12563264-156.out	R-12563264-32.err	R-12563264-53.out	R-12563264-75.err
R-12563264-157.err	R-12563264-32.out	R-12563264-54.err	R-12563264-75.out
R-12563264-157.out	R-12563264-33.err	R-12563264-54.out	R-12563264-76.err
R-12563264-158.err	R-12563264-33.out	R-12563264-55.err	R-12563264-76.out
R-12563264-158.out	R-12563264-34.err	R-12563264-55.out	R-12563264-77.err
R-12563264-159.err	R-12563264-34.out	R-12563264-56.err	R-12563264-77.out
R-12563264-159.out	R-12563264-35.err	R-12563264-56.out	R-12563264-78.err
R-12563264-160.err	R-12563264-35.out	R-12563264-57.err	R-12563264-78.out
R-12563264-160.out	R-12563264-36.err	R-12563264-57.out	R-12563264-79.err
R-12563264-161.err	R-12563264-36.out	R-12563264-58.err	R-12563264-79.out
R-12563264-161.out	R-12563264-37.err	R-12563264-58.out	R-12563264-7.err
R-12563264-162.err	R-12563264-37.out	R-12563264-59.err	R-12563264-7.out

Benefits of an RSE service

Since you started helping I've managed to get 1,200 gpu jobs completed



Hi Jake,

This is looking great. I think it is exactly what I want.

it work, thaaaaank you



Thanks a lot Jacob for your kind help!

A photograph of four business professionals in a meeting. A woman on the left is high-fiving a man in the center, who is also high-fiving a man on the right. A woman on the far right is partially visible, also appearing to be part of the celebration. They are all smiling and looking towards the center. The background is plain white.

Benefits of an RSE service

- Relationship building
- Recognition of RSCS
- Freeing time
- Proactive mitigation
- Jobs complete more quickly
- Improved productivity: global problems solved faster

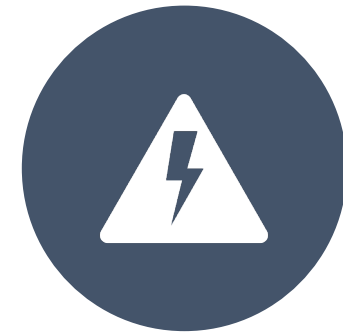
Increased efficiency?



“REDUCE POWER
CONSUMPTION”?



INCREASE RESEARCH
OUTPUT



POWER MANAGEMENT



Conclusions

- No two jobs are the same
- Impressive improvements can be made using simple solutions
- It is worth trying several approaches
- Productive users and good relationships



Thank you for
listening