The Alan Turing Institute

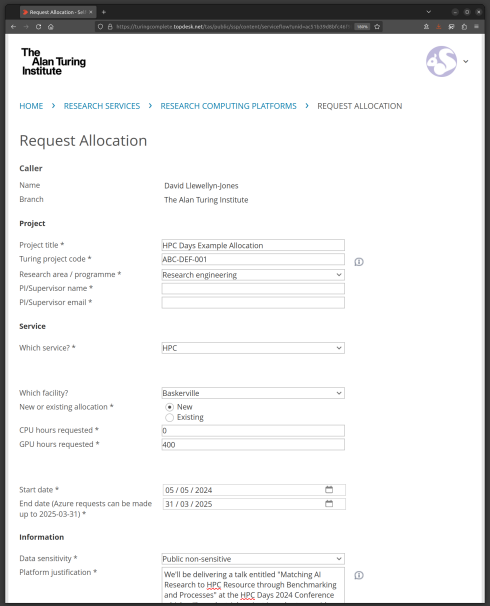
Matching AI Research to HPC Resource through Benchmarking and Processes

David Llewellyn-Jones, Tomas Lazauskas

The Problem

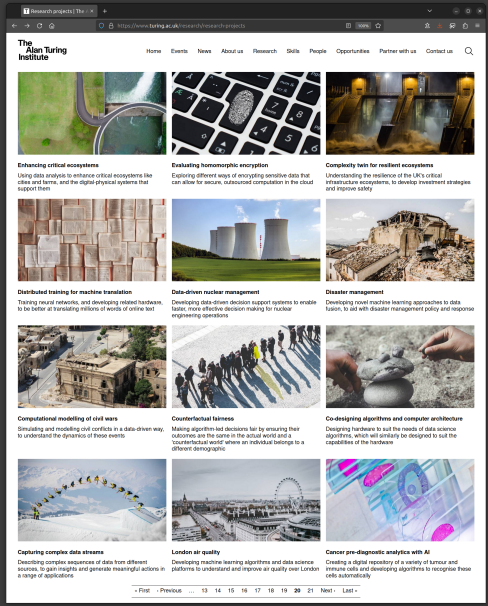
Matching Projects to Systems

1. Many diverse users and projects
2. Many diverse systems and characteristics
3. Researchers aren't familiar with the system
4. Research Computing isn't involved with the project
5. Communication is through ticketing system



A Diverse Institute

1. Over 400 researchers
2. Data science, machine learning, AI
3. Grand challenges
   - 3.1 Defence and national security
   - 3.2 Environment and sustainability
   - 3.3 Transformation of health
4. Digital society and policy



Core Capabilities

1. Research software engineering capability
   Growing our core research software engineering capability to continue to contribute skills in research software engineering and data science in support of national priorities.
2. Open-source infrastructure
   Expanding our work in the development and provision of open-source infrastructure that is accessible to all.


Bull


Lenovo
UNIVERSITY OF BIRMINGHAM
BASKERVILLE
diamond


UNIVERSITY OF CAMBRIDGE
DELL Technologies
intel
UK Atomic Energy Authority
UKRI
UK Research and Innovation
StackHPC

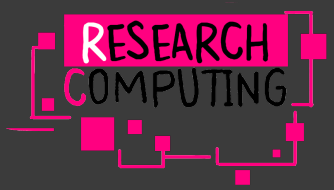

COSMA
DELL EMC

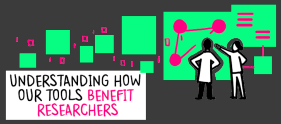
Our Approach

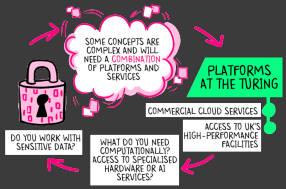
Our Approach

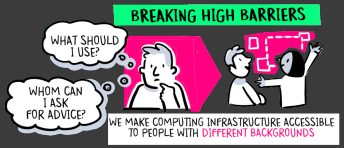
Four-pronged approach

1. Knowledge base and training
2. Structured onboarding
3. Trial access
4. Embedding in projects







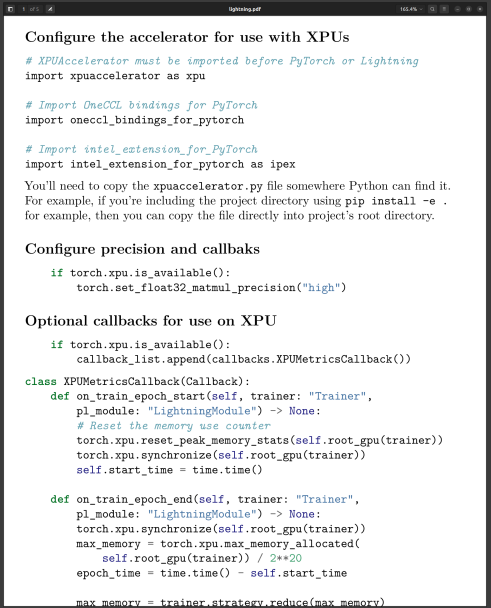


Knowledge Base and Training

1. Walkthroughs
2. Periodic training
3. Developing benchmarking results

Walkthroughs

1. Most available systems have excellent docs
2. System-specific, but can't possibly cover all tools
3. The Turing has a narrower focus
4. Different tools have (mostly) excellent docs
5. But rarely HPC-specific (let alone system-specific)



lightning.pdf

Configure the accelerator for use with XPUs

```
# XPUAccelerator must be imported before PyTorch or Lightning
import xpuaccelerator as xpu

# Import OneCCL bindings for PyTorch
import oneccl_bindings_for_pytorch

# Import intel_extension_for_PyTorch
import intel_extension_for_pytorch as ipex
```

You'll need to copy the `xpuaccelerator.py` file somewhere Python can find it. For example, if you're including the project directory using `pip install -e .` for example, then you can copy the file directly into project's root directory.

Configure precision and callbaks

```
    if torch.xpu.is_available():
        torch.set_float32_matmul_precision("high")
```

Optional callbacks for use on XPU

```
    if torch.xpu.is_available():
        callback_list.append(callbacks.XPUMetricsCallback())

class XPUMetricsCallback(Callback):
    def on_train_epoch_start(self, trainer: "Trainer",
        pl_module: "LightningModule") -> None:
        # Reset the memory use counter
        torch.xpu.reset_peak_memory_stats(self.root_gpu(trainer))
        torch.xpu.synchronize(self.root_gpu(trainer))
        self.start_time = time.time()

    def on_train_epoch_end(self, trainer: "Trainer",
        pl_module: "LightningModule") -> None:
        torch.xpu.synchronize(self.root_gpu(trainer))
        max_memory = torch.xpu.max_memory_allocated(
            self.root_gpu(trainer)) / 2**20
        epoch_time = time.time() - self.start_time

        max_memory = trainer.strategy.reduce(max_memory)
```

System-Tooling Matrix

Tool	JADE2	Baskerville	COSMA8	Azure
PyTorch	✓	✓	✓	
Lightning	✓	✓	✓	
⚡ Fabric				
Deepspeed	✓	✓	✓	
FSDP	✓	✓	✓	
Tensor Parallel				
MPI	✓	✓	✓	
oneCCL				
Accelerate				
AzureML	✗	✗	✗	✓

Periodic Training

1. Annual training from the Baskerville Team
2. Bespoke in-house training
3. Knowledge-sharing tech-talks and reading groups

Developing Benchmarking Results

Benchmarking different HPC systems

1. Tasks relevant for AI workloads
2. Develop walkthroughs in parallel

Help users and projects select appropriate systems

1. Hard to compare based on individual systems' websites
2. Different systems have quite different characteristics for different workloads
3. Will give some results later

Structured Onboarding

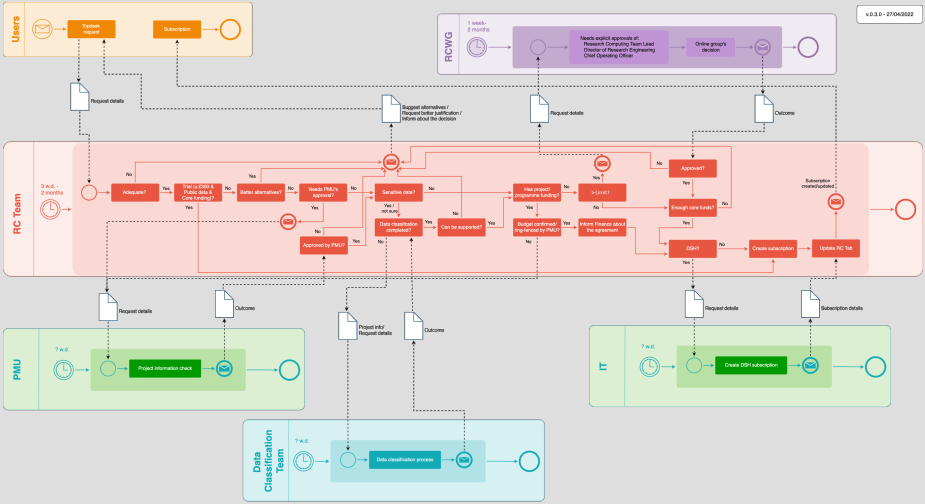
Flowchart processes

1. Modelled on a successful internal allocation flowchart
2. Built around a set of Intranet pages
3. Originally a wall of text

Ticketing system

1. Works okay but introduces project ping-pong
2. Want to avoid by providing more accessible material up-front

Backed up with drop-in sessions


v.0.3.0 - 27/04/2022
Users
Topdesk request
Subscription
RCWG
1 week - 2 months
Needs explicit approvals of: Research Computing Team Lead Director of Research Engineering Chief Operating Officer
Online group's decision
Request details
Suggest alternatives / Request better justification / Inform about the decision
Request details
Outcome
RC Team
3 w.d. - 2 months
Adequate?
Trial i.e. ESD & Public data & Core funding??
Better alternatives?
Needs PMU's approval?
Approved by PMU?
Sensitive data?
Yes / not sure
Data classification completed?
Can be supported?
Has project programme funding?
>1 unit?
Budget confirmed/ ring-fenced by PMU?
Inform Finance about the agreement
Approved?
Enough core funds?
DSH?
Create subscription
Update RC ToS
Subscription created/updated
Request details
Outcome
Project info/ Request details
Outcome
Request details
Subscription details
PMU
7 w.d.
Project information check
IT
7 w.d.
Create DSH subscription
Data Classification Team
7 w.d.
Data classification process

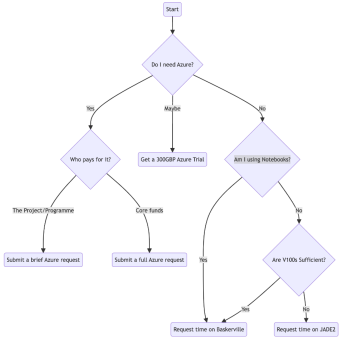
Flowchart Processes

1. Developed separate user-facing flowchart
2. Plan to develop into an interactive flow
3. Plus drop-in sessions

Applying for allocations

Requests should be submitted through Turing Complete (https://turingcomplete.topdesk.net/: Research Services -> Research Computing Platforms -> Request Allocation).

If it is your first time completing the request form, you can refer to this image to get some idea of which service to apply for. These are only recommendations and you may apply for access to any computing service. Each decision corresponds to a subheading further down the page.



Do I need Azure?

For your computing needs, you have the option of using cloud computing (namely, Microsoft Azure) and/or HPC (namely, Baskerville and JADE2) systems.

Microsoft Azure is a cloud computing platform, which provides services such as virtual machines, virtual networks and databases as well as services targeted at specific fields, such as analytics, machine learning, and internet of things. Microsoft Azure is particularly suitable if you need full control of the resource. For example, to host a publicly available web service, when software installation requires root access and for long-term storage of large data volumes. If you will be working with sensitive data, see Trusted Research Environments.

Trial Access

All Turing users can request trial access

1. Minimal justification
2. Restricted to 400 GPU hours
3. Aimed at helping scope and specify requirements
4. Production systems

Can be converted to full subscriptions

Benchmarking

HPC Benchmarking

1. Explore real-world training performance
2. Use PyTorch Lightning for multi-GPU strategies
   `https://github.com/Lightning-Universe/lightning-GPT`
3. Focus on GPT-2 (minGPT)

Model	Hidden layers	Attention heads	Embedding dim	Parameters (M)	16 bit Size (MB)
GPT2	12	12	768	85.21	170.51
GPT2-M	24	16	1024	302.51	605.16
GPT2-L	36	20	1280	708.64	1417.45
GPT2-XL	48	25	1600	1475.87	2951.96
GPT2-XXL	60	30	1920	2656.08	5312.43
GPT2-XXXL	84	40	2560	6609.33	12219.00

HPC Systems

Service	Name	Type	Accelerator	GB	Interface	Launched
JADE 2	J-V100-32	GPU	Nvidia V100	32	SXM2	06-2017
Baskerville	B-A100-40	GPU	Nvidia A100	40	SXM4	06-2020
Baskerville	B-A100-80	GPU	Nvidia A100	80	SXM4	06-2020
Stanage	S-H100-80	GPU	Nvidia H100	80	PCIe 4.0	03-2023
COSMA8	C-MI100-32	GPU	AMD MI100	32	PCIe 4.0	11-2020
COSMA8	C-MI210-64	GPU	AMD MI210	64	PCIe 4.0	03-2022
Graphcore	IPU-POD 16	IPU	IPU-M2000	14.4	RoCEv2	03-2021
Dawn	D-MX1550-128	GPU	Intel Max 1550	128	PCIe 5.0	03-2023

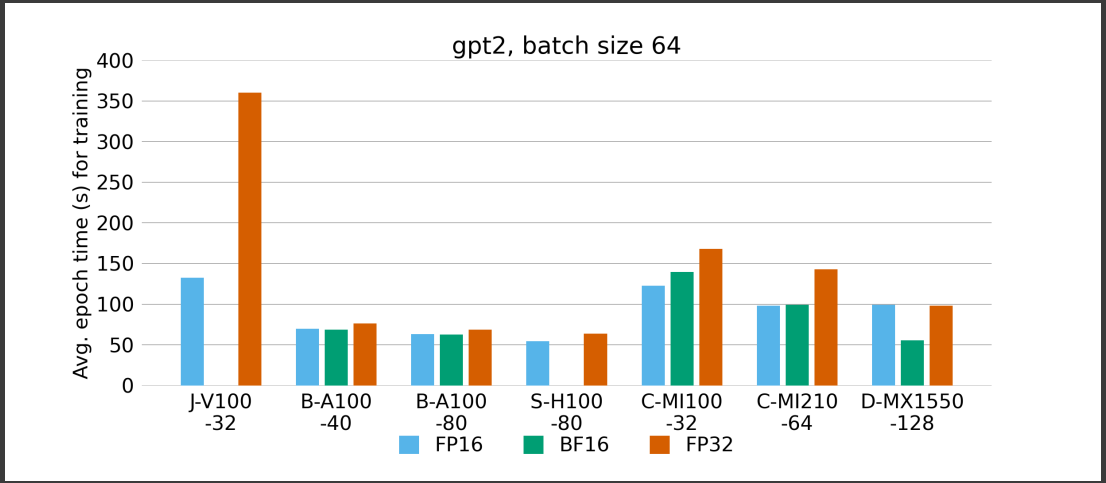
HPC Peak Performance on Paper (TFLOPs)

Service	Name	GB	FP16	BF16	FP32	FP64
JADE 2	J-V100-32	32	31.33	N/A	15.7	7.8
Baskerville	B-A100-40	40	312	312	19.5	9.7
Baskerville	B-A100-80	80	312	312	19.5	9.7
Stanage	S-H100-80	80	1513	1513	51	26
COSMA8	C-MI100-32	32	184.6	92.3	23.1	11.5
COSMA8	C-MI210-64	64	181	181	22.6	22.6
Graphcore	IPU-POD 16	14.4	3994	N/A	998	N/A
Dawn	D-MX1550-128	128	52.43	832	52.43	52.43

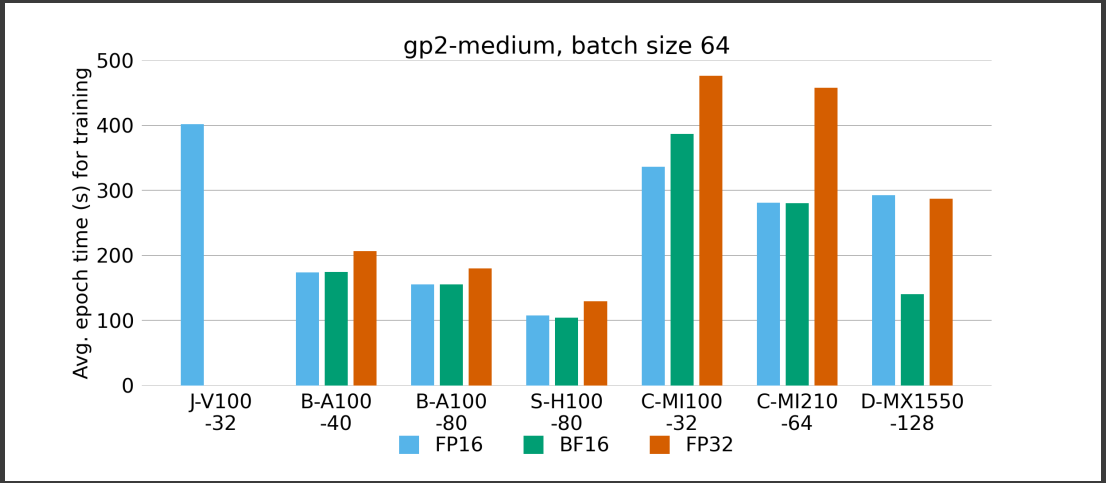
Strategies

1. Distributed Data Parallel
2. DeepSpeed ZeRO
3. Fully Sharded Data Parallel
4. Pipelined Execution
5. Sharded Execution

Single Accelerator Comparison



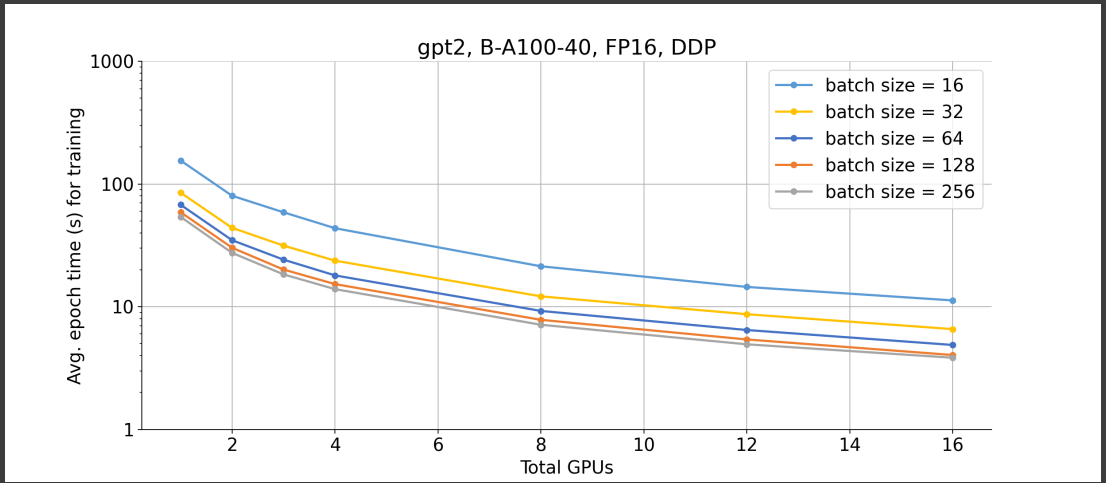
Single Accelerator Comparison



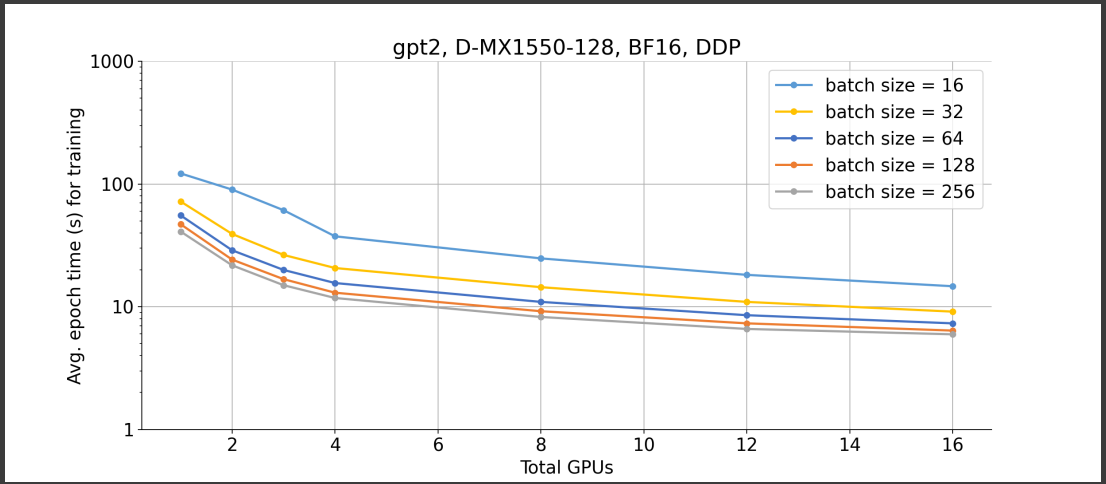
Single Accelerator Comparison - Observations

1. Nvidia H100 80 GB is the fastest GPU, theoretically and actually
2. Performance gap between H100, A100, Max 1550 not as large as expected
3. Difference between 16 and 32 bit less significant for smaller models, except for V100
4. Increased model size significantly increases training time
5. GPT2-M and a batch size of 128 too large for 40 GB

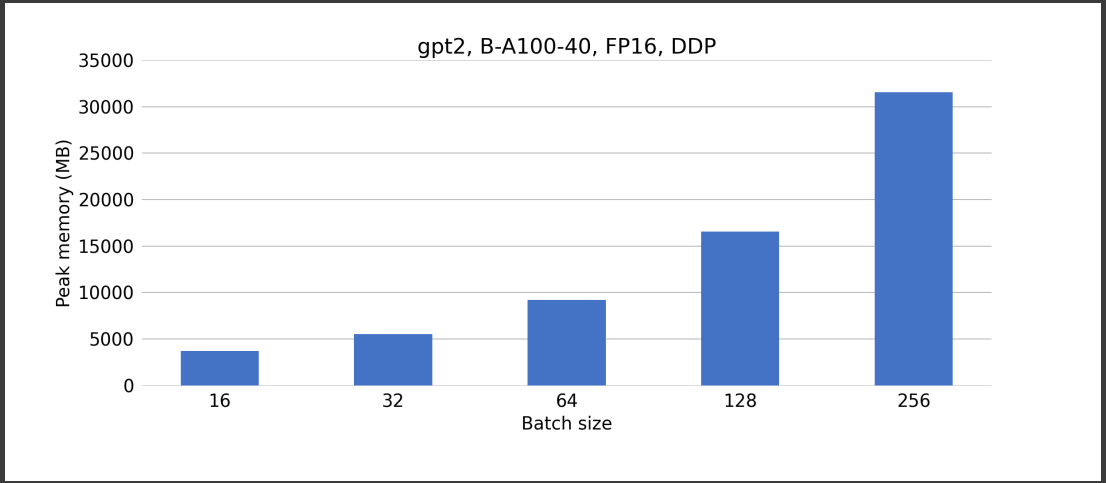
Scaling Up and Out with DDP



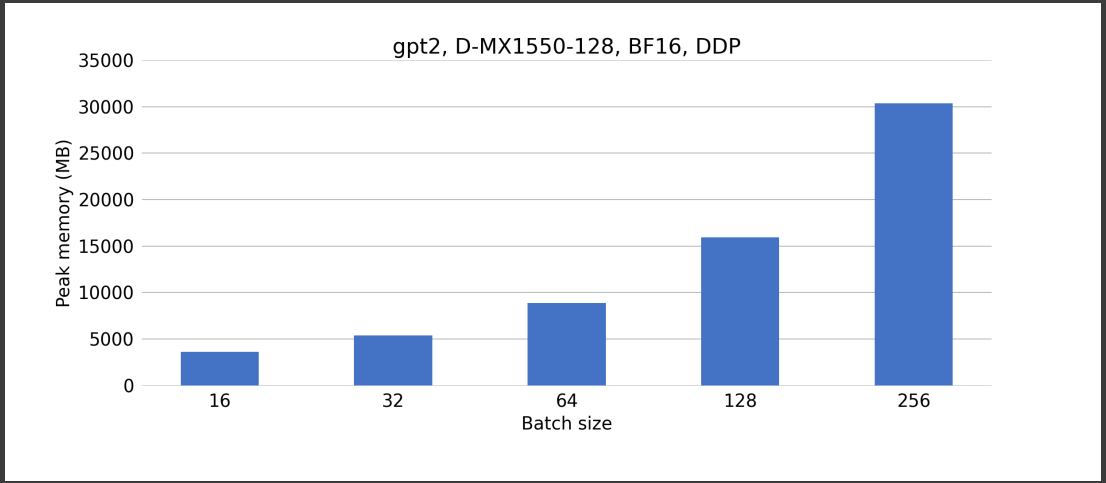
Scaling Up and Out with DDP



Scaling Up and Out with DDP



Scaling Up and Out with DDP



Scaling Up and Out with DDP - Observations

1. Scaling between 1 and 16 GPUs marginally sub-linear
2. Batch size 64 to 128 decreases training time by 15%
3. Batch size 64 to 256 reduces training time by 22%
4. Fixed model size, limiting performance factor is batch size and GPU memory
5. Doubling batch size increases peak memory usage by a factor of 1.5

Conclusions

Conclusions - Benchmarks

1. BFLOAT16 peak performance better indicator for AI workloads than FP16 or FP32
2. MI100 32 GB and MI210 64 GB potentially more suitable for traditional HPC tasks
3. FSDP faster than DeepSpeed, but DeepSpeed Stage 3 more memory-efficient for largest models
4. Balanced consideration of memory and time is needed especially for larger models

Conclusions - Process

1. Understanding HPC trade-offs is difficult for researchers
2. We use four approaches to try to help
   2.1 Knowledge base and training
   2.2 Structured onboarding
   2.3 Trial access
   2.4 Embedding in projects
3. These are all still work-in-progress

Acknowledgements

1. With thanks to Edwin Brown, Sheffield and Turing
2. Funded by The Alan Turing Institute under the EPSRC grant EP/N510129/1
3. Partially supported by Baskerville, a national accelerated compute resource under the EPSRC Grant EP/T022221/1
4. Partially supported by JADE: Joint Academic Data Science Endeavour - 2 under the EPSRC Grant EP/T022205/1, and The Exascale Computing: Algorithms and Infrastructures Benefiting UK Research (ExCALIBUR) program, which is funded under Wave 2 of the Strategic Priorities Fund (SPF)
5. Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (`https://www.csd3.cam.ac.uk`), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/T022159/1)
6. The University of Sheffield for the provision of services for High Performance Computing
7. The Mandelbrot system at the UCL Centre for Advanced Research Computing and associated support services (`https://www.ucl.ac.uk/advanced-research-computing/advanced-research-computing-centre`)
8. DiRAC@Durham facility managed by the Institute for Computational Cosmology on behalf of the STFC DiRAC HPC Facility (`https://www.dirac.ac.uk`). The equipment was funded by BEIS capital funding via STFC capital grants ST/P002293/1, ST/R002371/1 and ST/S002502/1, Durham University and STFC operations grant ST/R000832/1. DiRAC is part of the National e-Infrastructure