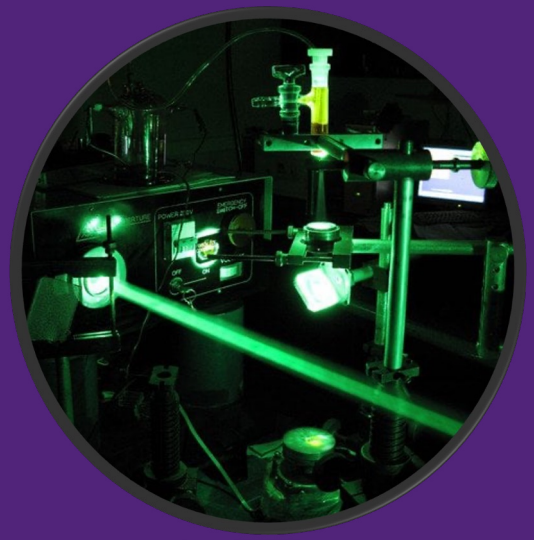
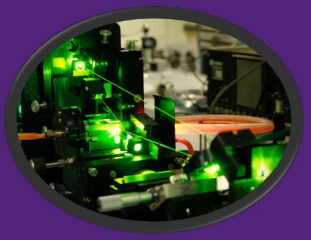
A site update from the University of Queensland, Australia

Things we're seeing, things we've broken, things we're doing next…





Jake Carroll
University of Queensland, Australia
jake.carroll@uq.edu.au
ORCID ID: 0000-0002-7765-5772

Introduction

- In 20 minutes or less we'll cover:
  - **Accelerated computing quirks** – deeper analysis and impact of GPU differentiation – odd things going on we just noticed.
  - **FAL chaos** – our recent journey through the windshield with File Audit Logging.
  - **National scale data fabric dreams** using Storage Scale.

A top 50 global university

36

U.S. News Best Global Universities (2023)

37

NTU Performance Ranking of Scientific Papers (2023)

40

QS World University Ranking (2025)

63

Academic Ranking of World Universities (2024)

70

Times Higher Education World University Ranking (2024)

Who we are

55,000+
students

21,500+
international students

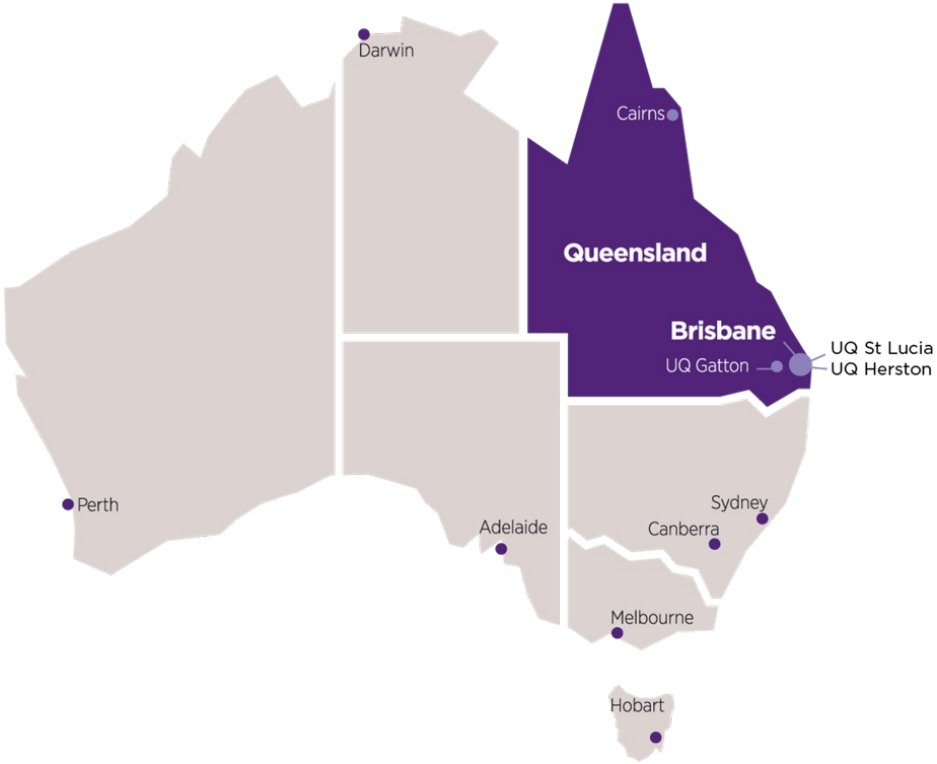
20,000+
postgraduate students

17,700+
PhD graduates

8,500
staff

340,000+
graduates

Where we are



4
campuses

6
faculties

40+
teaching and research sites

8
research institutes

UQ produces quality research

#1 in Australia
Nature Index tables
(2023)

95%
of UQ research above world standard (ERA, 2018)

$437.7 million
in research funding
(2022)

31% increase
in global partners since 2013

213 fellows
of learned academies are UQ staff and emeritus professors

17,700+
PhDs awarded since 1952

Overall winner Work & Careers Education Best Universities Ranking

Here's the overall winner in the AFR's Best Universities Ranking

For the second year in a row, UQ has taken out pole position in The Australian Financial Review Best Universities Ranking.

UQ named Australia's best in AFR ranking

14 November 2024

The University of Queensland has topped the Australian Financial Review's (AFR) Best Universities Ranking for the second consecutive year.

The ranking measures Australian universities across four pillars - teaching, research, equity and career impact.

Vice-Chancellor Professor Deborah Terry AC said the result reaffirmed UQ as one of Australia's most comprehensive universities, with tangible impact across education, research and communities.

UQ has been named the best university for the second consecutive year in the Australian Financial Review's 2024 Best Universities Ranking.

"We are proud to have Australia's most awarded university teachers deliver our high-quality programs that equip students with the knowledge and skills to transition into their chosen industry," Professor Terry said.

"Through education, we can transform the lives of individuals and enrich communities, and this purpose drives our mission to improve equity and access to higher education through our Queensland Commitment initiatives."

CRICOS code 00025B

UQ Research Computing Centre

Deputy Vice-Chancellor (Research & Innovation)

Digital Research Infrastructure

- High-speed "Medici" data fabric – powered by IBM Storage Scale and HPE's DMF7
- IBM HCI Fusion
- "Bunya" supercomputer
- Research cloud (ARDC Nectar QLD Node)
- Workflow platforms

Expertise

- Researchers with years of experience providing extensive support
- Supporting all UQ researchers who need computing resources
- Domain experts in fields including biosciences, physics, computational chemistry, computer science, machine learning
- Systems engineers with a broad range of experience

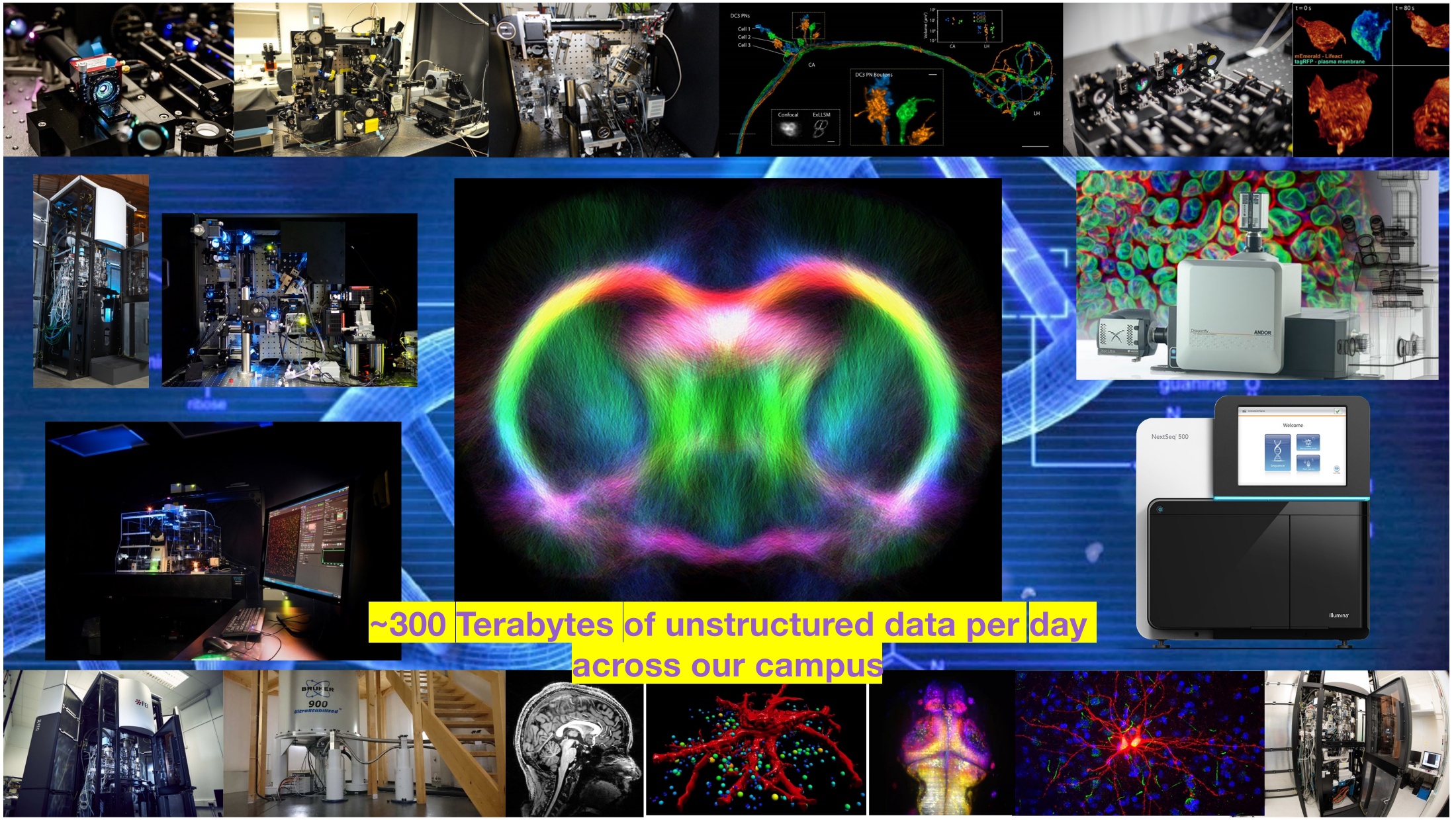
The University Of Queensland
AUSTRALIA

80 Petabytes Research Data Storage capacity

Best-in-class storage economics, sustainability and scalability

Over 4000 Researchers Used HPC Bunya as a primary resource in 2024

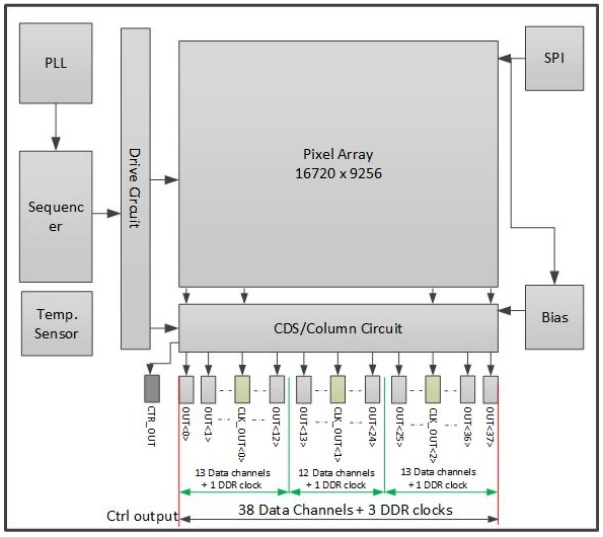
Over 11 million jobs have run on HPC Bunya since its deployment

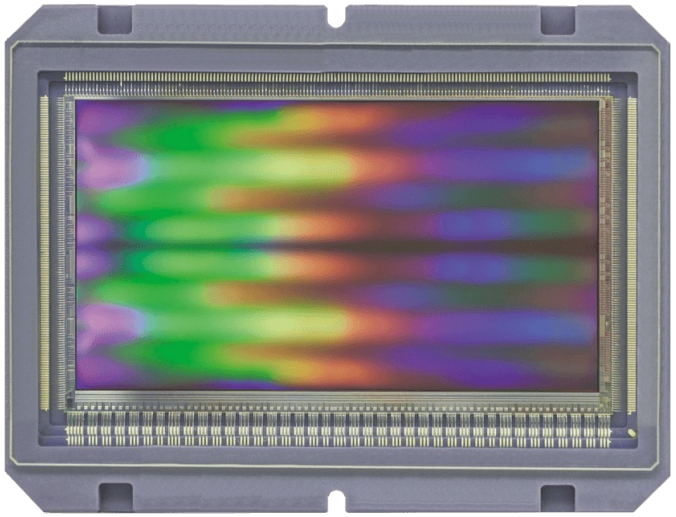

DC3 PNs
Cell 1
Cell 2
Cell 3
CA
LH
DC3 PN Boutons
Confocal
ExLLSM
t = 0 s
t = 80 s
mEmerald - Lifeact
tagRFP - plasma membrane
ANDOR
NextSeq 500
Welcome
Illumina
~300 Terabytes of unstructured data per day
across our campus

March 2024: a 152MP sCMOS landed in the labs at UQ.

152MP (megapixels) per sCMOS

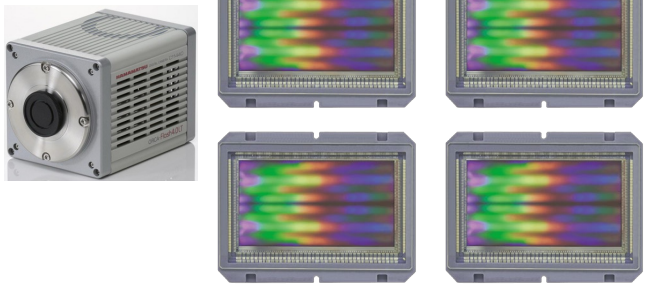
16720x9256.

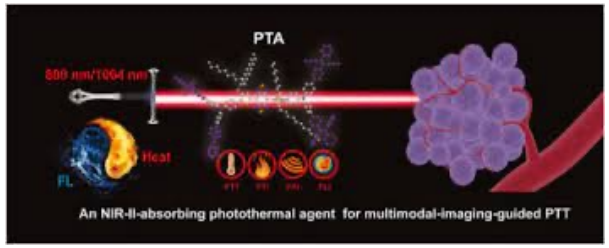


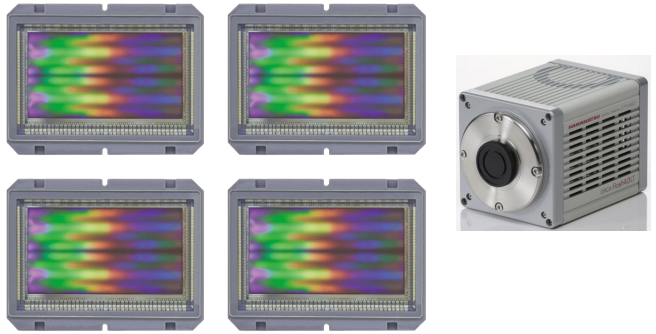


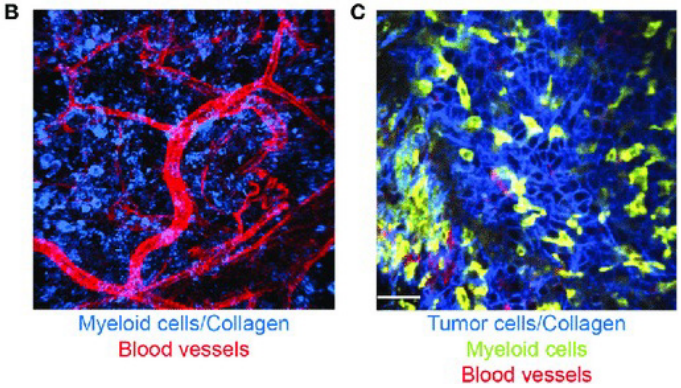
CRICOS code 00025B

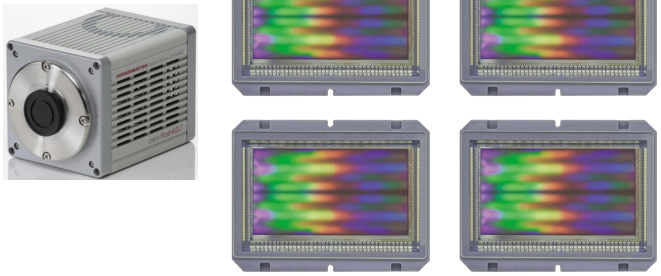
April 2024: Now put an array of 16 of these together…

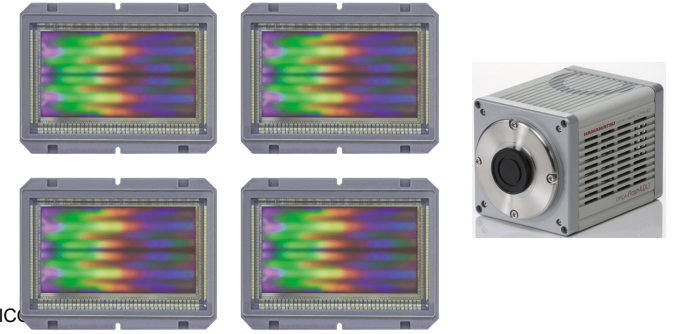












2,432MP (megapixels)

256Gbps from the block-face chain

Temporal + Spatial Resolution win.

Actual conversation

Researcher: Man, did you see what we did with those sensors? HOW COOL IS THAT? SO MUCH GOOD TEMPORAL AND SPATIAL RESOLUTION, YO!

Me:

ʘ_ʘ

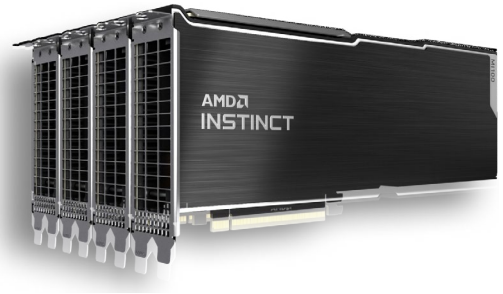
(ﾉ°□°) ﾉ ⁀ ┻━┻

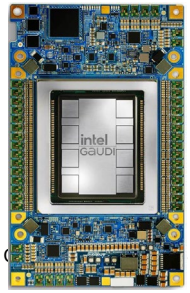
GPU differentiation and its impact on storage.

CRICOS code 00025B

This isn’t your grandma’s GPU, *mate.*

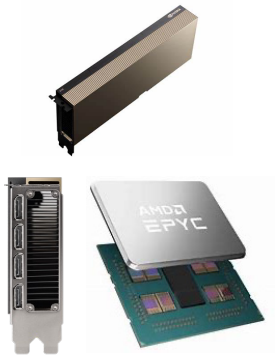
- UQ is on a GPU diversity mission. Plenty of NVIDIA stuff in the tank, but it is critical to look at other things, too.
  - OpenAI have apparently started using a different GPU en masse for inference serving, recently [*AMD MI300x*]. Isn’t that an interesting sign?
- We’re in MI300x, MI210 and Gaudi3 territory now, as well. There is a lot going on.





The path of Fusion at UQ

Q1 2023: Fusion Building Block #1 lands at UQ

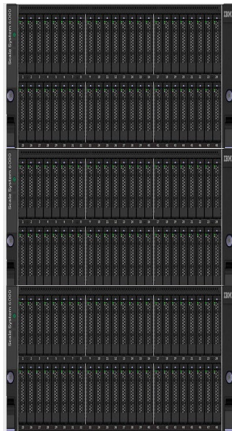


Q2 2023: Fusion gets GPUs and more CPU cores.

Q1 2024: Fusion gets AMD MI210x, IBM SPEED Program

Q2 2024: ESS 6000 Storage Scale in Fusion

Q3 2024: FlashSystem 5300



Q1 2025: IBM ESS 6000 JBOD Expansions

Continuum of sponsorship and enablement from IBM to UQ

Where has Fusion taken UQ to date?

The IBM @ UQ partnership stands as one of the largest deployments of capability between the two sectors in the Asia Pacific.

We now have:

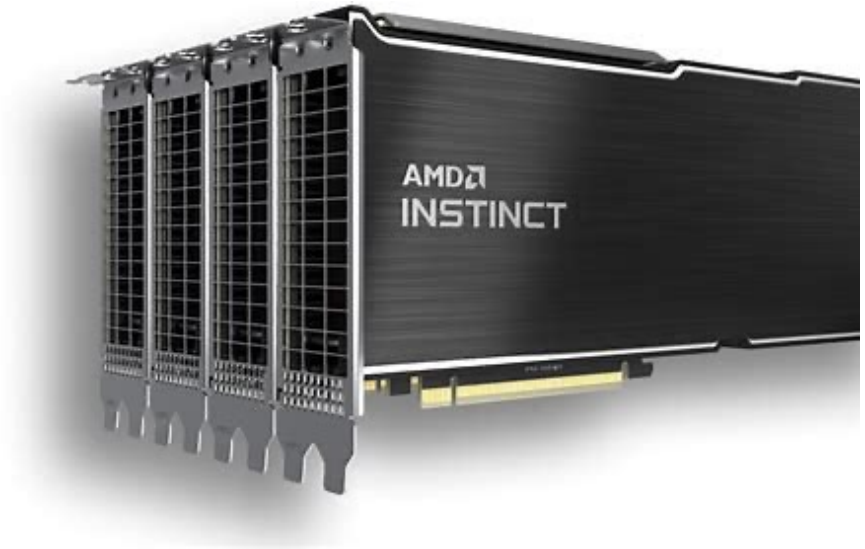
- Cutting edge parallel filesystem capabilities.
- Cutting edge GPU capabilities.
- Converged Infrastructure.
- Best of breed block storage

...all in place, due to the partnership.

FS5300 – FCM4 FlashCore Modules…

The Scale Storage System 6000

IBM@UQ's SPEED Program and Sponsorship: AMD GPU for Fusion HCI, Watsonx, Watsonx.ai



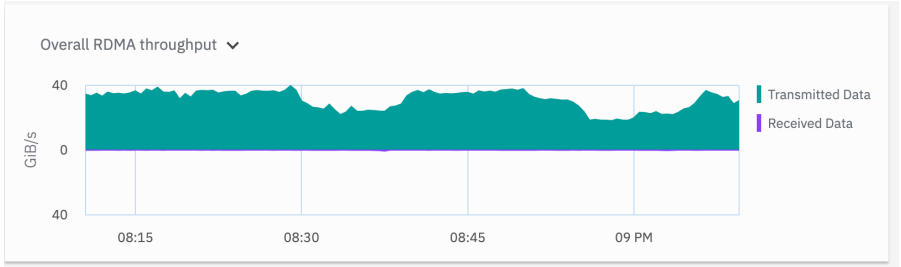
Why is this important? UQ and IBM worked together to qualify the Watson platform for AMD GPU enablement and compatibility – a world first, allowing the IBM Watsonx models to run natively, *agnostically* on NVIDIA *or* AMD GPU platforms.

Different GPUs create varied storage conditions with the same workload. We're trying to characterize it.

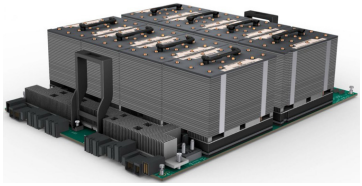
Workload: Model warm-up: oLlama 3.1 405b; 233GB
Command: `ollama model run llama3.1:405b`

Workload: Model warm-up: granite-3-guardian 8b; 2.7GB
Command: `ollama model run granite-3-guardian:8b`

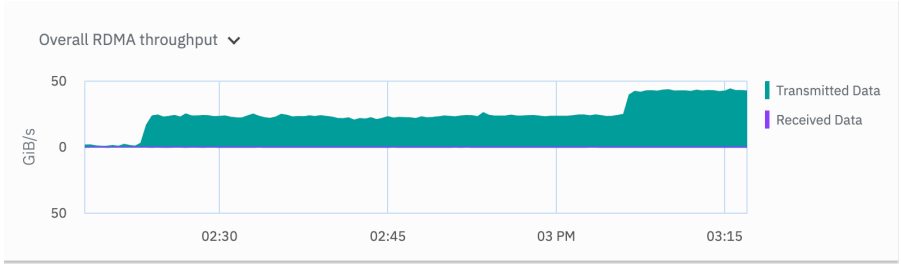
NVIDIA 4*H100



Slower and less consistent model load into into HBM3, peak IO @ 40GB/sec.



AMD 8*MI300x



Graduated IO into HBM3, peak IO @ 50GB/sec.

CRICOS code 00025B

Warming up the node…

```
time=2024-11-18T03:17:47.189+10:00 level=INFO source=.:0 msg="Server listening on 127.0.0.1:35359"
llama_model_loader: loaded meta data with 36 key-value pairs and 362 tensors from /home/uqjcarr1/.ollama/models/blobs/sha256-
fc93e012ac809ee2e55858a9104ee9e2ad49139aefff5a372cb5101f4d3a1db4 (version GGUF V3 (latest))
llama_model_loader: Dumping metadata keys/values. Note: KV overrides do not apply in this output.
llama_model_loader: - kv   0:                       general.architecture str              = granite
llama_model_loader: - kv   1:                               general.type str              = model
llama_model_loader: - kv   2:                               general.name str              = Granite Guardian 3.0 8b
llama_model_loader: - kv   3:                          general.basename str              = granite-guardian-3.0
llama_model_loader: - kv   4:                         general.size_label str              = 8B
llama_model_loader: - kv   5:                            general.license str              = apache-2.0
llama_model_loader: - kv   6:                               general.tags arr[str,1]       = ["text-generation"]
llama_model_loader: - kv   7:                          general.languages arr[str,1]       = ["en"]
llama_model_loader: - kv   8:                        granite.block_count u32              = 40
llama_model_loader: - kv   9:                     granite.context_length u32              = 8192
llama_model_loader: - kv  10:                   granite.embedding_length u32              = 4096
llama_model_loader: - kv  11:                granite.feed_forward_length u32              = 12800
llama_model_loader: - kv  12:               granite.attention.head_count u32              = 32
llama_model_loader: - kv  13:            granite.attention.head_count_kv u32              = 8
llama_model_loader: - kv  14:                     granite.rope.freq_base f32              = 10000.000000
llama_model_loader: - kv  15:   granite.attention.layer_norm_rms_epsilon f32              = 0.000010
llama_model_loader: - kv  16:                           general.file_type u32              = 17
llama_model_loader: - kv  17:                       granite.vocab_size u32              = 49155
llama_model_loader: - kv  18:                 granite.rope.dimension_count u32              = 128
llama_model_loader: - kv  19:            tokenizer.ggml.add_space_prefix bool             = false
llama_model_loader: - kv  20:                    granite.attention.scale f32              = 0.007812
llama_model_loader: - kv  21:                   granite.embedding_scale f32              = 12.000000
llama_model_loader: - kv  22:                    granite.residual_scale f32              = 0.220000
llama_model_loader: - kv  23:                       granite.logit_scale f32              = 16.000000
llama_model_loader: - kv  24:                      tokenizer.ggml.model str              = gpt2
llama_model_loader: - kv  25:                        tokenizer.ggml.pre str              = refact
llama_model_loader: - kv  26:                      tokenizer.ggml.tokens arr[str,49155]   = ["<|end_of_text|>", "<fim_prefix>", "...
llama_model_loader: - kv  27:                  tokenizer.ggml.token_type arr[i32,49155]   = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, ...
llama_model_loader: - kv  28:                      tokenizer.ggml.merges arr[str,48891]   = ["Ġ Ġ", "ĠĠ ĠĠ", "ĠĠĠĠ ĠĠ...
llama_model_loader: - kv  29:                tokenizer.ggml.bos_token_id u32              = 0
llama_model_loader: - kv  30:                tokenizer.ggml.eos_token_id u32              = 0
llama_model_loader: - kv  31:            tokenizer.ggml.unknown_token_id u32              = 0
llama_model_loader: - kv  32:            tokenizer.ggml.padding_token_id u32              = 0
llama_model_loader: - kv  33:               tokenizer.ggml.add_bos_token bool             = false
llama_model_loader: - kv  34:                    tokenizer.chat_template str              = {%- set risk_bank = ({"social_bias": ...
llama_model_loader: - kv  35:               general.quantization_version u32              = 2
llama_model_loader: - type  f32:   81 tensors
llama_model_loader: - type q5_K:  240 tensors
llama_model_loader: - type q6_K:   41 tensors
```

```
ggml_cuda_init: found 1 ROCm devices:
  Device 0: AMD Radeon Graphics, compute capability 9.4, VMM: no
llm_load_tensors: ggml ctx size =    0.34 MiB
llm_load_tensors: offloading 40 repeating layers to GPU
llm_load_tensors: offloading non-repeating layers to GPU
llm_load_tensors: offloaded 41/41 layers to GPU
llm_load_tensors:      ROCm0 buffer size =  5527.21 MiB
llm_load_tensors:        CPU buffer size =   157.51 MiB
llama_new_context_with_model: n_ctx      = 8192
llama_new_context_with_model: n_batch    = 2048
llama_new_context_with_model: n_ubatch   = 512
llama_new_context_with_model: flash_attn = 0
llama_new_context_with_model: freq_base  = 10000.0
llama_new_context_with_model: freq_scale = 1
llama_kv_cache_init:      ROCm0 KV buffer size =  1280.00 MiB
llama_new_context_with_model: KV self size  = 1280.00 MiB, K (f16):  640.00 MiB, V (f16):  640.00 MiB
llama_new_context_with_model:  ROCm_Host  output buffer size =     0.81 MiB
llama_new_context_with_model:      ROCm0 compute buffer size =   560.00 MiB
llama_new_context_with_model:  ROCm_Host compute buffer size =    24.01 MiB
llama_new_context_with_model: graph nodes  = 1368
llama_new_context_with_model: graph splits = 2
time=2024-11-18T03:17:55.094+10:00 level=INFO source=server.go:601 msg="llama runner started in 7.94
seconds"
[GIN] 2024/11/18 - 03:17:55 | 200 |   7.97522246s |       127.0.0.1 | POST     "/api/generate"
```

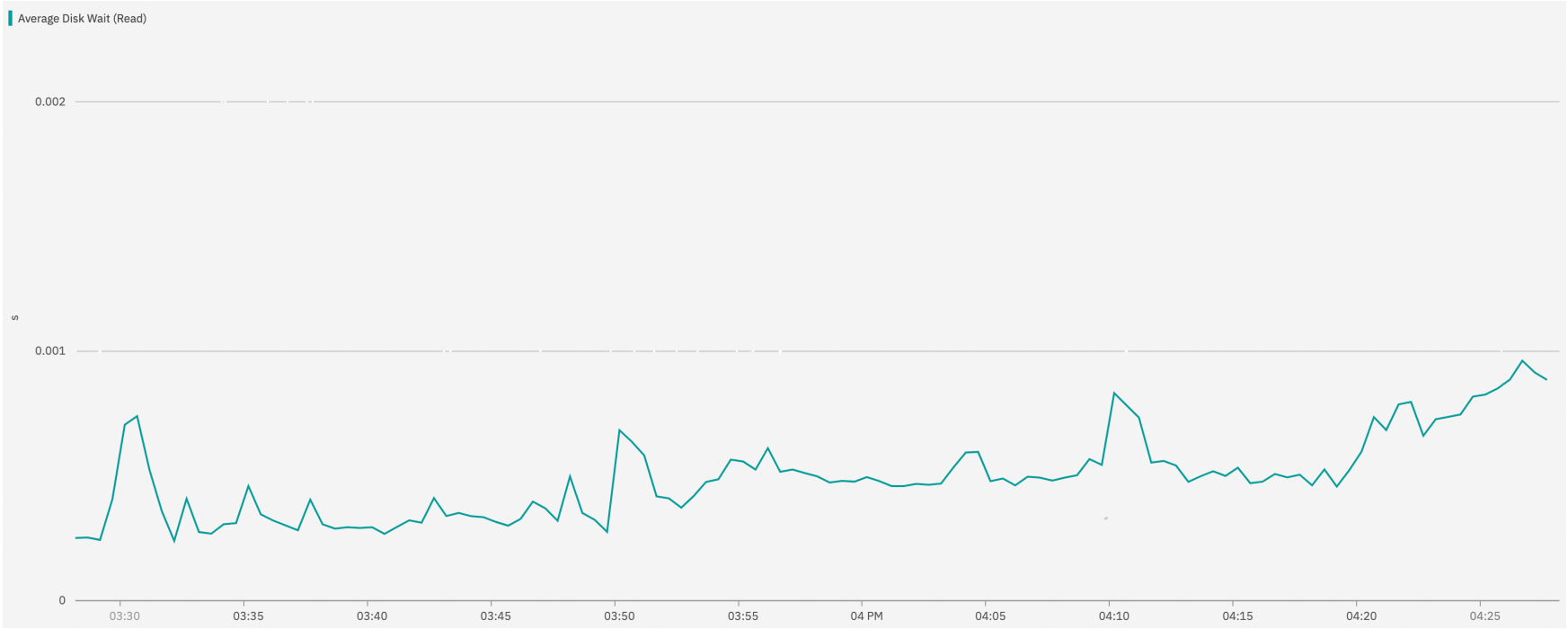
Granite warm, after about 8 seconds…

Similar for Llama 3.1-405 billion

```
llm_load_tensors: offloading non-repeating layers to GPU
llm_load_tensors: offloaded 127/127 layers to GPU
llm_load_tensors:      ROCm0 buffer size = 27362.00 MiB
llm_load_tensors:      ROCm1 buffer size = 27362.00 MiB
llm_load_tensors:      ROCm2 buffer size = 27362.00 MiB
llm_load_tensors:      ROCm3 buffer size = 27362.00 MiB
llm_load_tensors:      ROCm4 buffer size = 27362.00 MiB
llm_load_tensors:      ROCm5 buffer size = 27362.00 MiB
llm_load_tensors:      ROCm6 buffer size = 27362.00 MiB
llm_load_tensors:      ROCm7 buffer size = 25585.72 MiB
llm_load_tensors:        CPU buffer size =  1127.25 MiB
llama_new_context_with_model: n_ctx      = 8192
llama_new_context_with_model: n_batch    = 2048
llama_new_context_with_model: n_ubatch   = 512
llama_new_context_with_model: flash_attn = 0
llama_new_context_with_model: freq_base  = 500000.0
llama_new_context_with_model: freq_scale = 1
llama_kv_cache_init:      ROCm0 KV buffer size =   512.00 MiB
llama_kv_cache_init:      ROCm1 KV buffer size =   512.00 MiB
llama_kv_cache_init:      ROCm2 KV buffer size =   512.00 MiB
llama_kv_cache_init:      ROCm3 KV buffer size =   512.00 MiB
llama_kv_cache_init:      ROCm4 KV buffer size =   512.00 MiB
llama_kv_cache_init:      ROCm5 KV buffer size =   512.00 MiB
llama_kv_cache_init:      ROCm6 KV buffer size =   512.00 MiB
llama_kv_cache_init:      ROCm7 KV buffer size =   448.00 MiB
llama_new_context_with_model: KV self size  = 4032.00 MiB, K (f16): 2016.00 MiB, V (f16): 2016.00 MiB
llama_new_context_with_model:  ROCm_Host  output buffer size =     2.21 MiB
llama_new_context_with_model: pipeline parallelism enabled (n_copies=4)
llama_new_context_with_model:      ROCm0 compute buffer size =  2368.01 MiB
llama_new_context_with_model:      ROCm1 compute buffer size =  2368.01 MiB
llama_new_context_with_model:      ROCm2 compute buffer size =  2368.01 MiB
llama_new_context_with_model:      ROCm3 compute buffer size =  2368.01 MiB
llama_new_context_with_model:      ROCm4 compute buffer size =  2368.01 MiB
llama_new_context_with_model:      ROCm5 compute buffer size =  2368.01 MiB
llama_new_context_with_model:      ROCm6 compute buffer size =  2368.01 MiB
llama_new_context_with_model:      ROCm7 compute buffer size =  2368.02 MiB
llama_new_context_with_model:  ROCm_Host compute buffer size =    96.02 MiB
llama_new_context_with_model: graph nodes  = 4038
llama_new_context_with_model: graph splits = 9
```

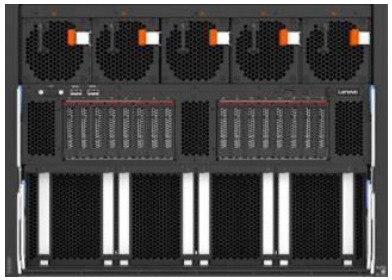
Stranger again – the difference in disk latency depending upon the GPU + IO-Framework? *[Nvidia H100]*

4 * H100 SXM5

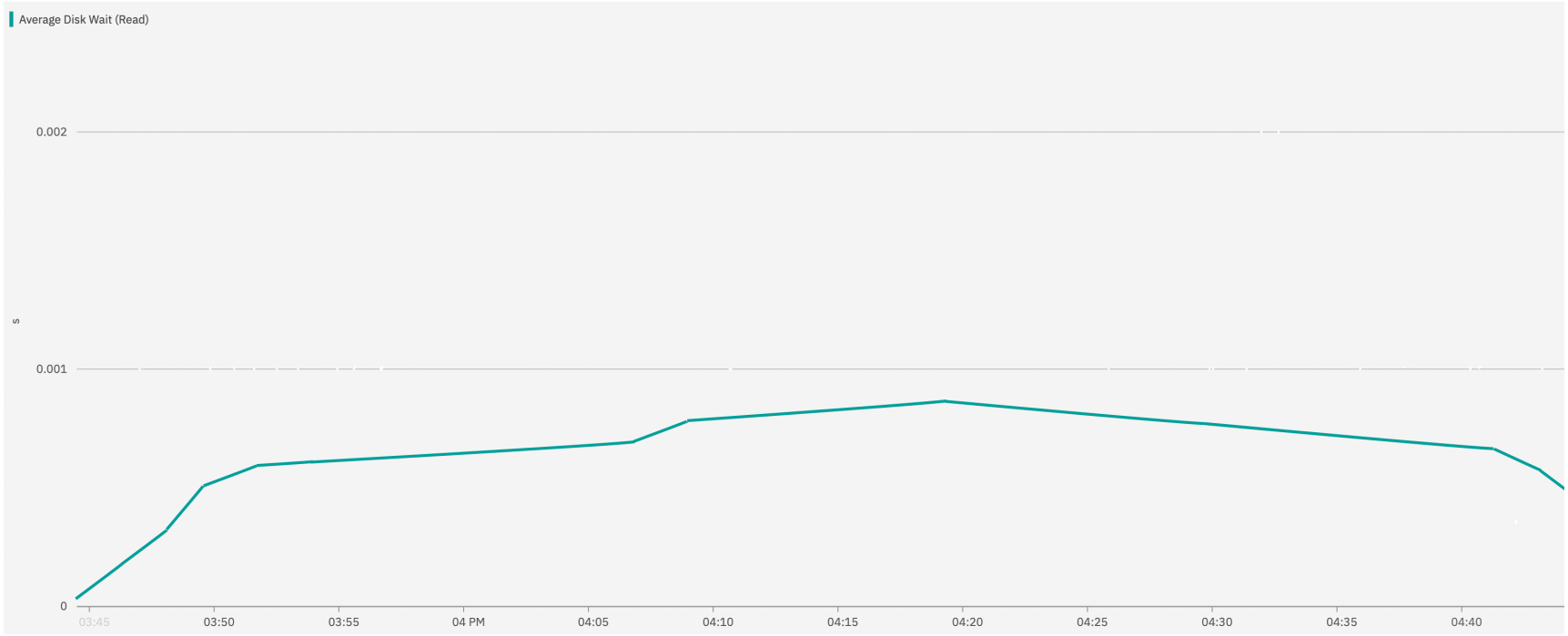


CRICOS code 00025B

Stranger again – the difference in disk latency depending upon the GPU + IO-Framework? *[AMD MI300x]*



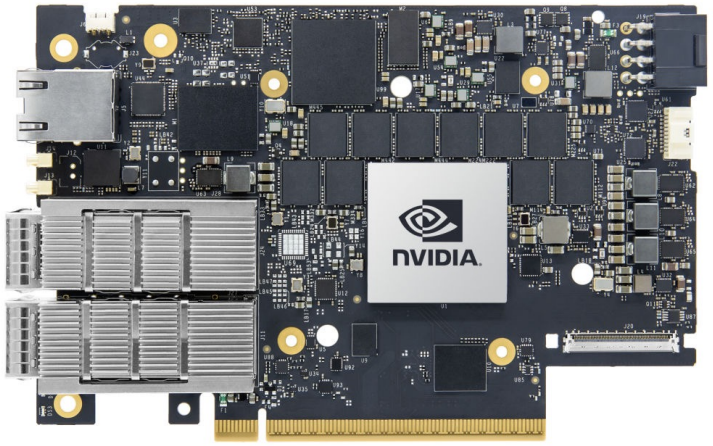
4 * MI300x OAM



CRICOS code 00025B

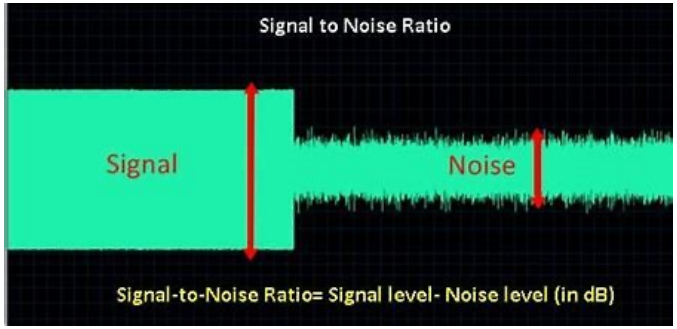
What's going on?

- Short answer: we don't know yet.
  - Could be related to the new BlueField-3 DPU versus generic HDR CX-6 cards? [Both running at HDR-200G].
  - Could be related to the way the IO libraries work in getting data out of storage scale and into the MI300x pad?



Implications

- Noise on the fabric. Get enough rough IO on a large deployment – and what was once predictable storage reliability turns into a trouble shooting exercise.
- At scale, what do latency lumps and inconsistencies look like?
- Would GDS on the NVIDIA GPU have made a difference?



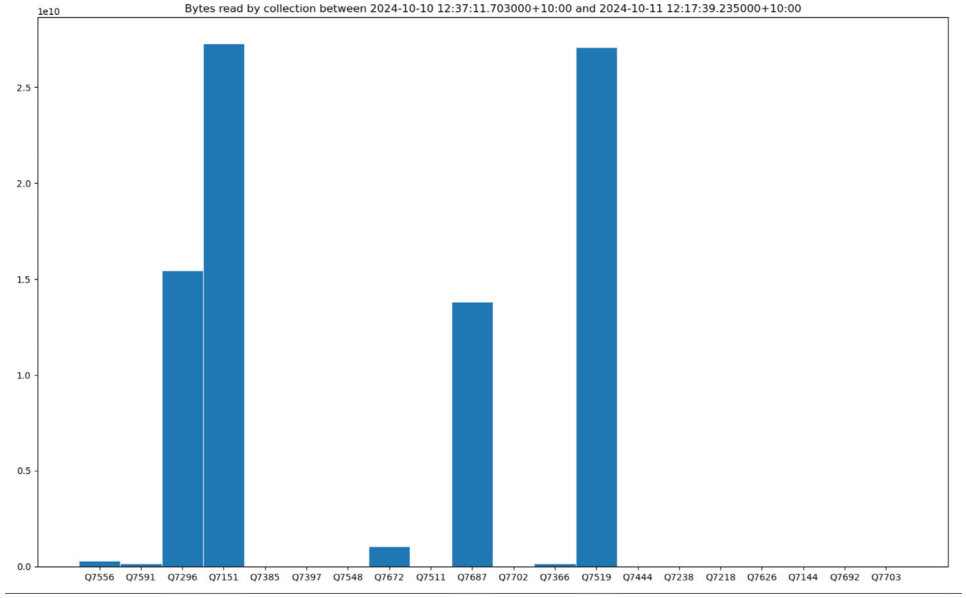
CRICOS code 00025B

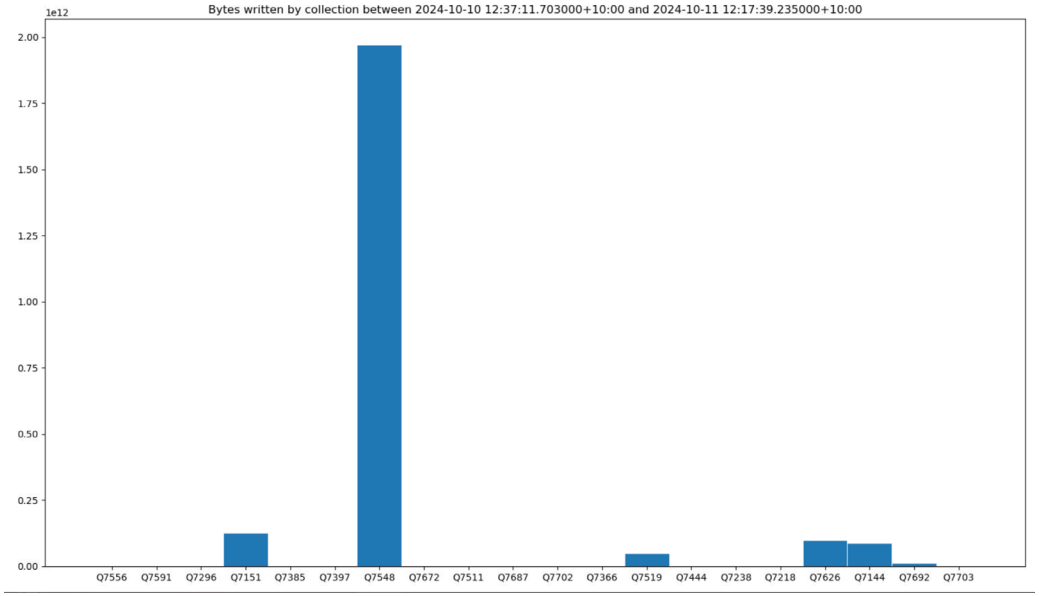
The University Of Queensland
AUSTRALIA

Taming your pet AI enchanted supercomputer, one AFM fileset at a time with FAL.

Hunting for noisy top-talkers in AFM filesets on ESS 3500/ESS 6000

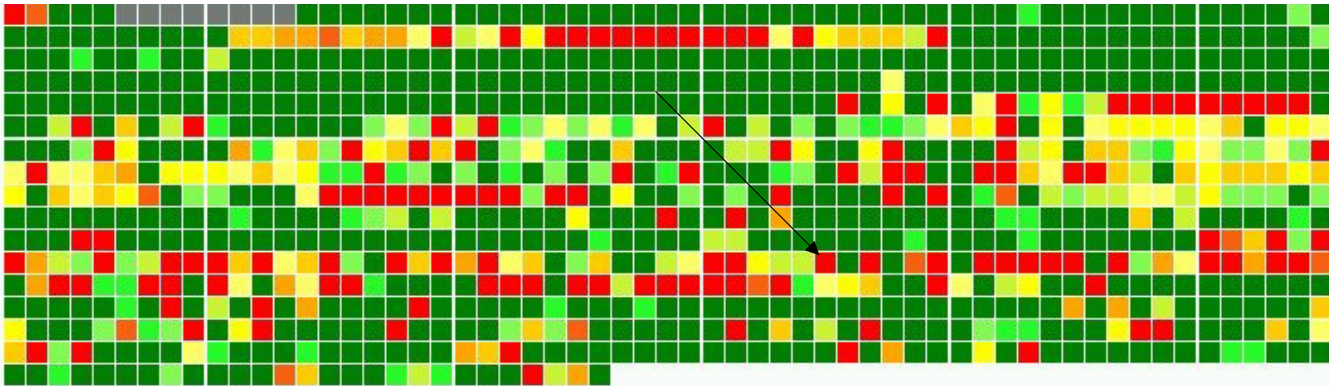
- Who here is using FAL? [mmaudit]
- We're doing something a little bit challenging with it. Perhaps you do as well? Perhaps you do **moreso?**
- Imagine running mmaudit and FAL on a 6.8 billion object Storage Scale cluster – and having telemetry feed out of that in real time, all day long, so you can determine what is "hot" in your AI-enchanted supercomputer.





Heatmap generation: New problems for our team (mostly Sarah) to make sense of for us…

Metrics ingest out of mmaudit text logs from the mmaudit fileset



Q7519

Q7296

Q7687

Q7672

`/afm/Q08/fileset/x`

Here, we colour each fileset as it appears in the frequency map of I/O activity from the supercomputer, as detected in path by mmaudit logging.

You might wonder why it matters?

- It's not enough to know how many IOPS or GB/sec your building block is carrying in and out at anymore
- It's not enough to know the latency of service you're getting from it, either. That's just the beginning.

Here is the crux of it:

- When you enable 1000's of people to talk to a data fabric ubiquitously, which is AFM backed, which is in turn backed by an HSM capability under the covers (DMF/HPSS etc) - that's a fantastic active archive, but people tend to do *dumb stuff*.
- Examples of **dumb stuff** includes treating the AFM fileset mount like a scratch disk without an appropriate cache or flushing policy in place – or drowning a very large bit of fileset infrastructure in IO (because you can) from 100's of compute nodes or GPUs.
- Understanding whom is doing what inside a supercomputer and to which storage can be hard so we devised a way to observe fileset IO using FAL outputs from mmaudit logs, put it into a big data-science soup maker and found a way to look at "hot" collections of significance inside the storage cluster.
- Different than zimon, not the same as what Storage Insights offers – as it targets things mounted on the HPC side.

Turns out we broke it all horrifically, and we needed...

Hewlett Packard Enterprise

IBM

FAL itself wasn't broken in isolation. It was interaction with DMF7 that needed work from Storage Scale DEV team.

- As some know, we use Storage Scale as a top level filesystem to act as our cache for DMF7 information lifecycle management (giant HSM).
- This is a common play now. Some use Lustre these days for the same task. We don't like Lustre much. (ಠ_ಠ)
- Turns out – DMF7 uses Ganesha to detect events to consume for filesystem ingest/tracking/archiving, as well as DMAPI.
- Also turns out – that the Ganesha API was, in interaction with the way HPE's DMF7 was consuming it – acting very strangely (full assert and crash of GPFS cluster) with **endian flipping on the wire up to calculate a PATH.**

What was eventually discovered by FAL-DEV

- dmf-gpfs-scanne calls kxGanesha with a file handle, and the file handle looks like this

- `0.989635332 3190200 TRACE_VNODE: ganesha_fh_to_dentry sbP 0xFFFF93668859E000 type 10 len 10: 01000001 00000000 00000000 BBF53800 00000000 2F689833 00000FFF 00000000`

- These contents represent this struct:

```
typedef struct nfsFileHandleCommon_t
{
UInt32 flags; /* nfs file handle flag fields */
UInt32 hash; /* Hash of the file dentry name */
UInt64 inode; /* InodeNumber for file */
UInt32 sid; /* Snapshot id for file */
UInt32 gen; /* File generation number for file */
UInt64 pinode; /* InodeNumber for parent file */
UInt32 psid; /* Snapshot id for parent file */
UInt32 pgen; /* File generation number for parent file */
}
```

CRICOS code 00025B

What was eventually discovered

- With the new updates to GPFS FAL module, now have to decode the file handle, and the first 5 fields appear to be in order, the flag 01000001 indicates that the format is in BE, and the fields are appropriately converted to LE format, and the call succeeds.

- Now, there are calls to `gpfs_fh_to_parent`, which decodes the last three fields (the fields with a 'p' prefix). When decoding these fields, we are again, looking at the flag 01000001, and since the file handle is in BE format, we byte swap the fields to get LE format.

- The problem now appears to be that the p-inode field is in LE format, even though the file handle indicates that the fields are in BE format. This causes GPFS code (before the new fixes) to change 00000FFF > FF0F0000, which is again, an invalid inode number. When we try to look it up, we recognize it's a (-) inode number, and **assert**.

Eventual outcome:

- GPFS can now send its Ganesha events downstream to be consumed by DMF7 with appropriate endianness using an in place correction routine upon detection - as we have a BE to LE conversion routine in-situ.
- Yes, there is now *intelligence* and *determinism* in the way the FAL code sends path creation/calculation out, depending on the consumer!
- If anyone out there ever runs this type of infrastructure and you need mmaudit – you won't meet this problem, because of this work. Lucky you!

Eventual outcome:

- How it happened at all: There are multiple ways to consume events from Storage Scale. Ganesha's API was internal and never intended for this – but this is how HPE chose to implement their consumer out of GPFS.
- **Despite all this:** HPE and IBM worked directly together over the course of a week or so to make it work for us (and not crash an entire GPFS remote mount cluster…).

So, cheers to you, IBM and HPE, for fixing a thing that was complicated, niche and a corner case – but you did it anyway and helped us. We sincerely appreciate the effort and it is what makes these communities special. This doesn't happen in every high-technology sector enclave.

National scale data fabrics

We have arrived at a place where there is asymmetric thinking in different organisations about what research data storage “is”

Some organisations look at storage and think:

Yet others think:

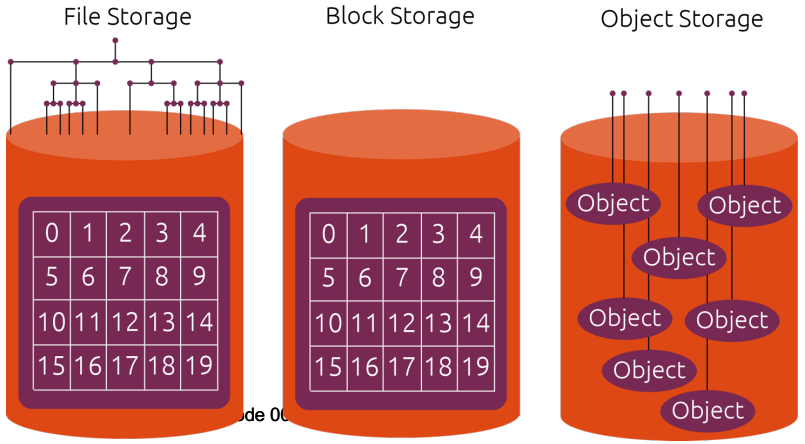
The problem: it isn't enough to store blocks or files anymore and the era of "the network share" has well and truly ended.

- A static SMB/NFS mount doesn't mean a great deal anymore in the era of multi-modal and multi-step experimentation and science. Stop being passive providers.

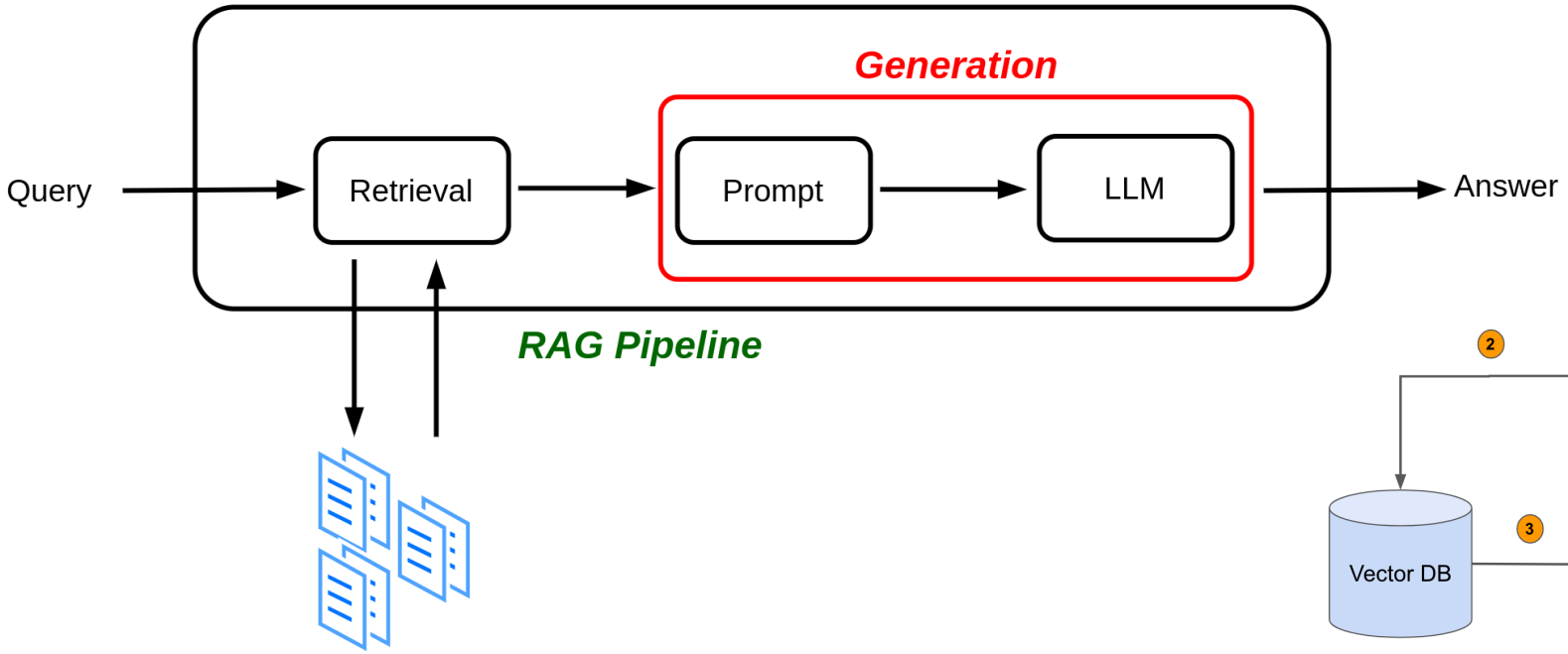
- It isn't that the concept of the network share is no longer valuable – it is that due to the way science and research now works, the ideals of a "dumb NAS" are not flexible nor do they necessarily offer what research now needs.

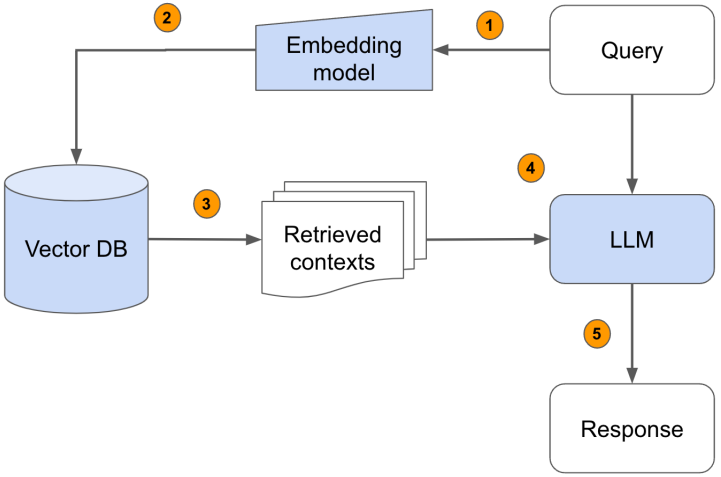
The has conversation shifted to:

- Fabrics
- AI-Enablement (workflows, movement, management)
- How you **participate** in research. It is **about facilitating scientific technique**. Not "storing stuff".



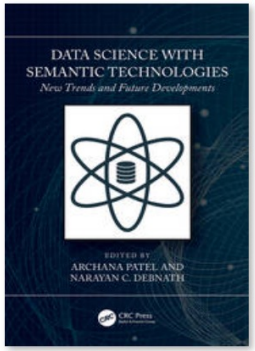
How do you make your storage go from a flat static thing, to a participant in the workflow and pipeline?





Data Fabric type technologies are now turning up everywhere. It seems to be well established that they are now here to stay – and it's the new way of data management for a modern era.

https://doi.org/10.1201/9781003310785



Chapter

Interoperability Frameworks

Data Fabric and Data Mesh Architectures

By Michael DeBellis, Livia Pinera, Christopher Connor

Book Data Science with Semantic Technologies

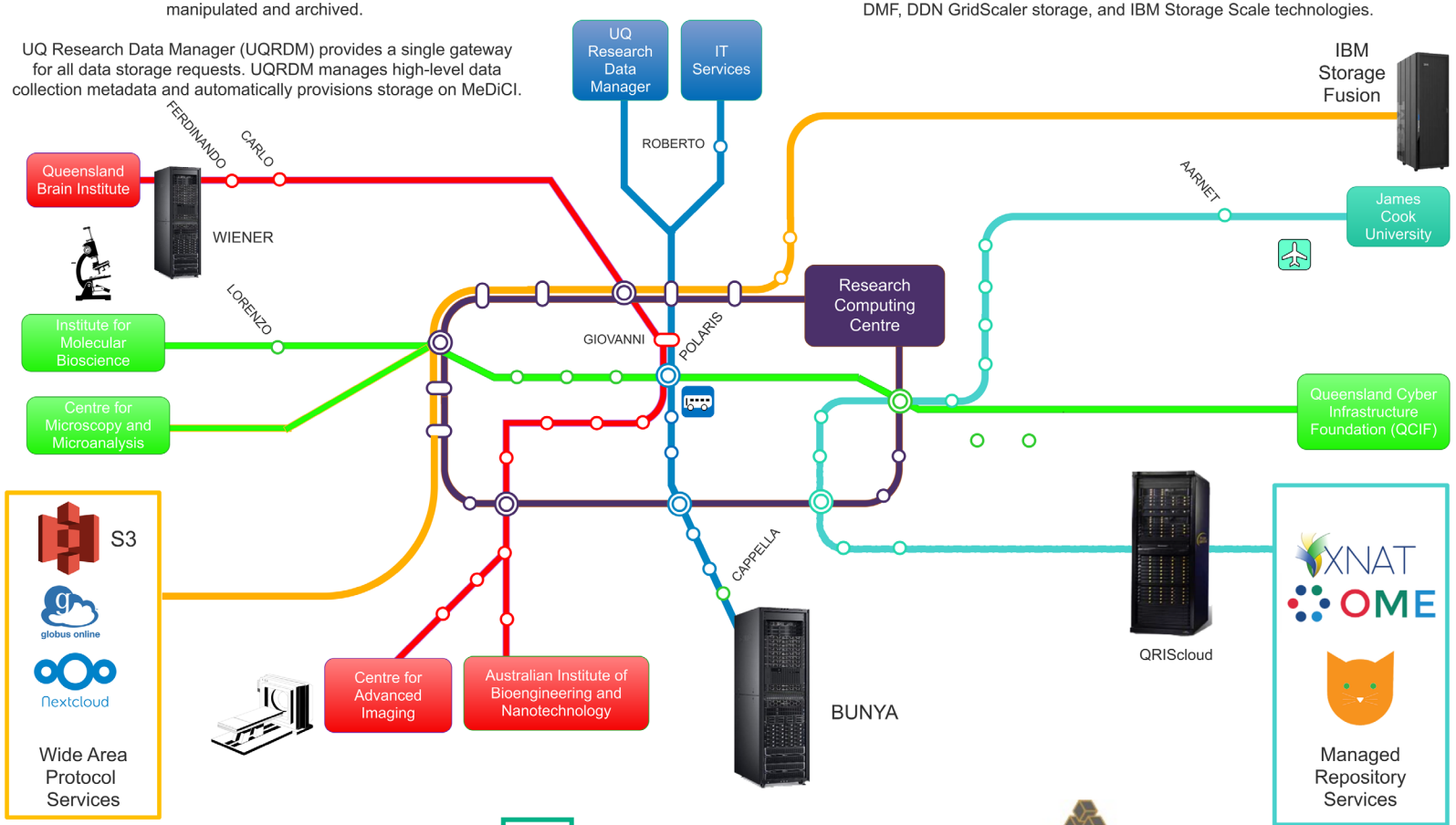
Edition	1st Edition
First Published	2023
Imprint	CRC Press
Pages	20
eBook ISBN	9781003310785

MeDiCI: UQ's Data Fabric of Choice (2024)

The University of Queensland's Metropolitan Data Caching Infrastructure (MeDiCI) spans urban Brisbane and provides seamless access to data regardless of where it is created, manipulated and archived.

UQ Research Data Manager (UQRDM) provides a single gateway for all data storage requests. UQRDM manages high-level data collection metadata and automatically provisions storage on MeDiCI.

MeDiCI holds copies of data on campus. Data is moved between on- and off-campus storage caches on demand without user involvement. MeDiCI is underpinned by HPE DMF, DDN GridScaler storage, and IBM Storage Scale technologies.



Benefits Of Using Data Fabric

1

Process and analyze data in real time

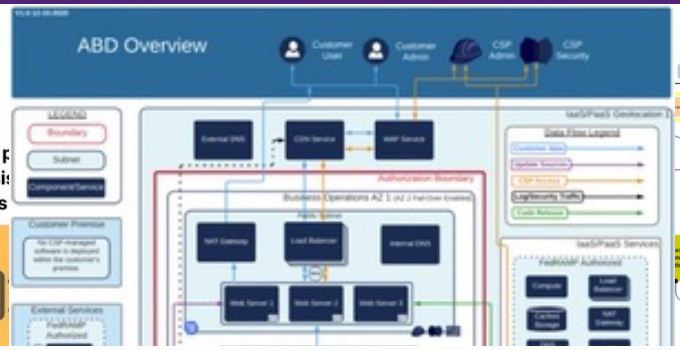
3

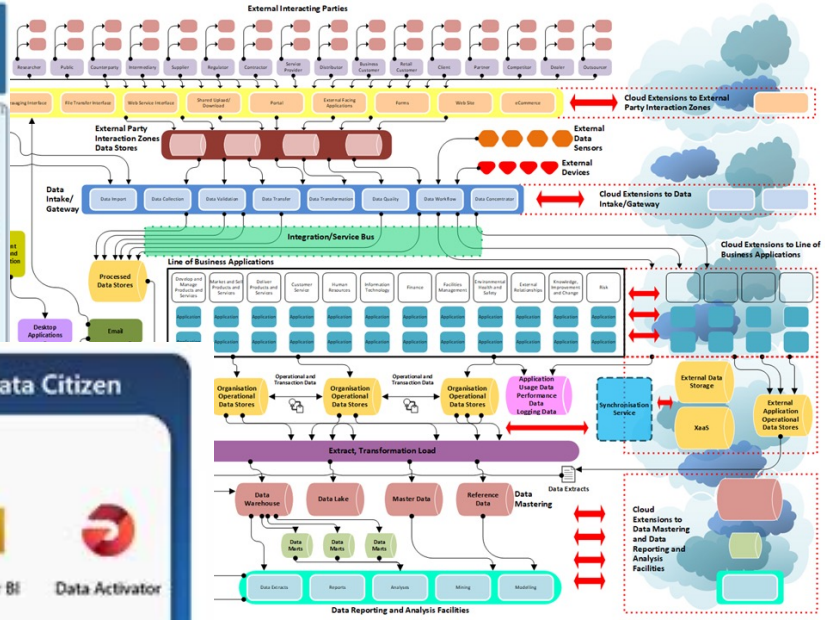
Scale data p and analysi capabilities

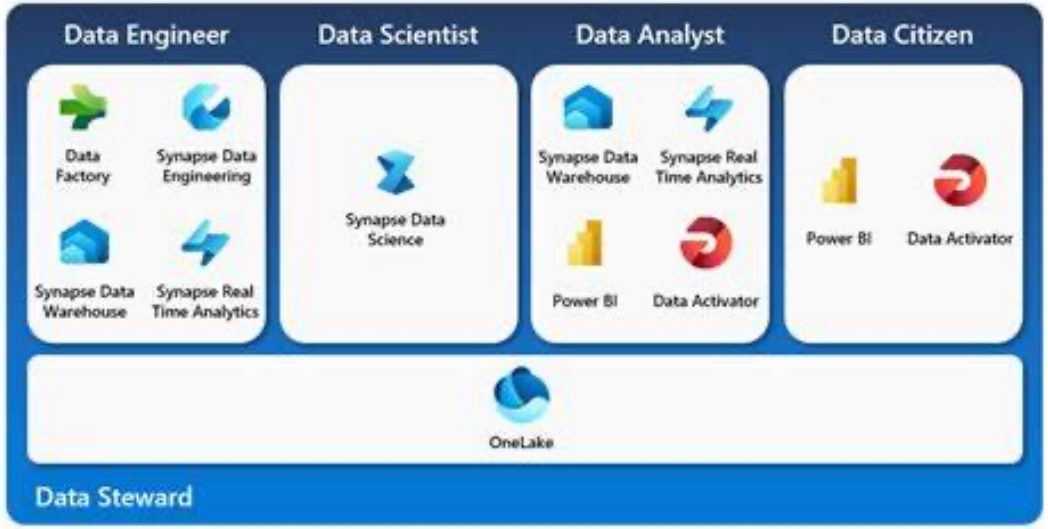
Consolidate data from multiple sources onto a single platform

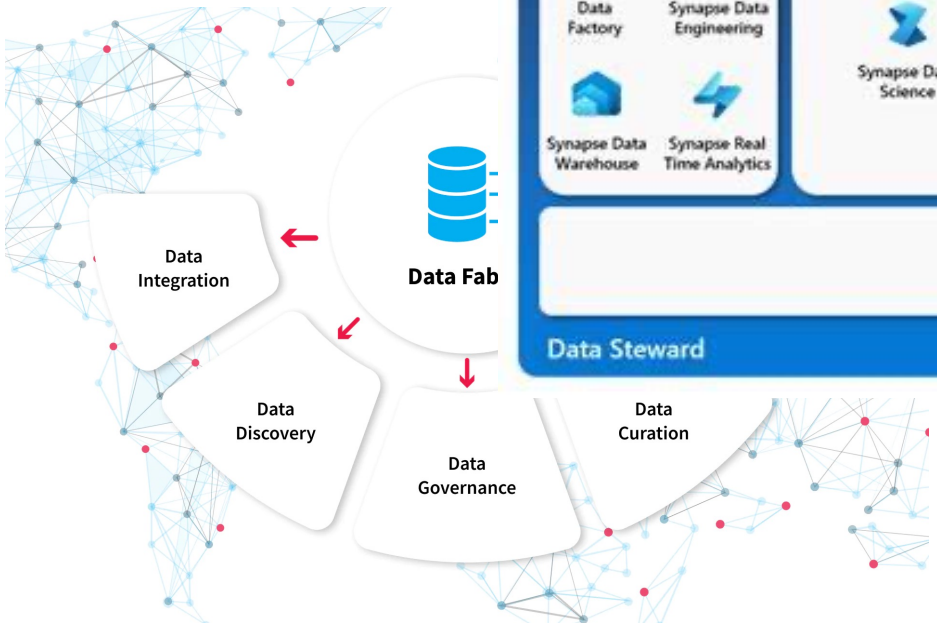
2

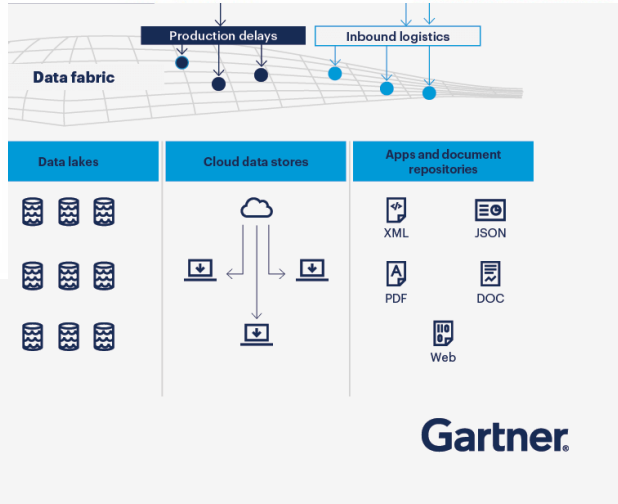
Access d anywhere, at









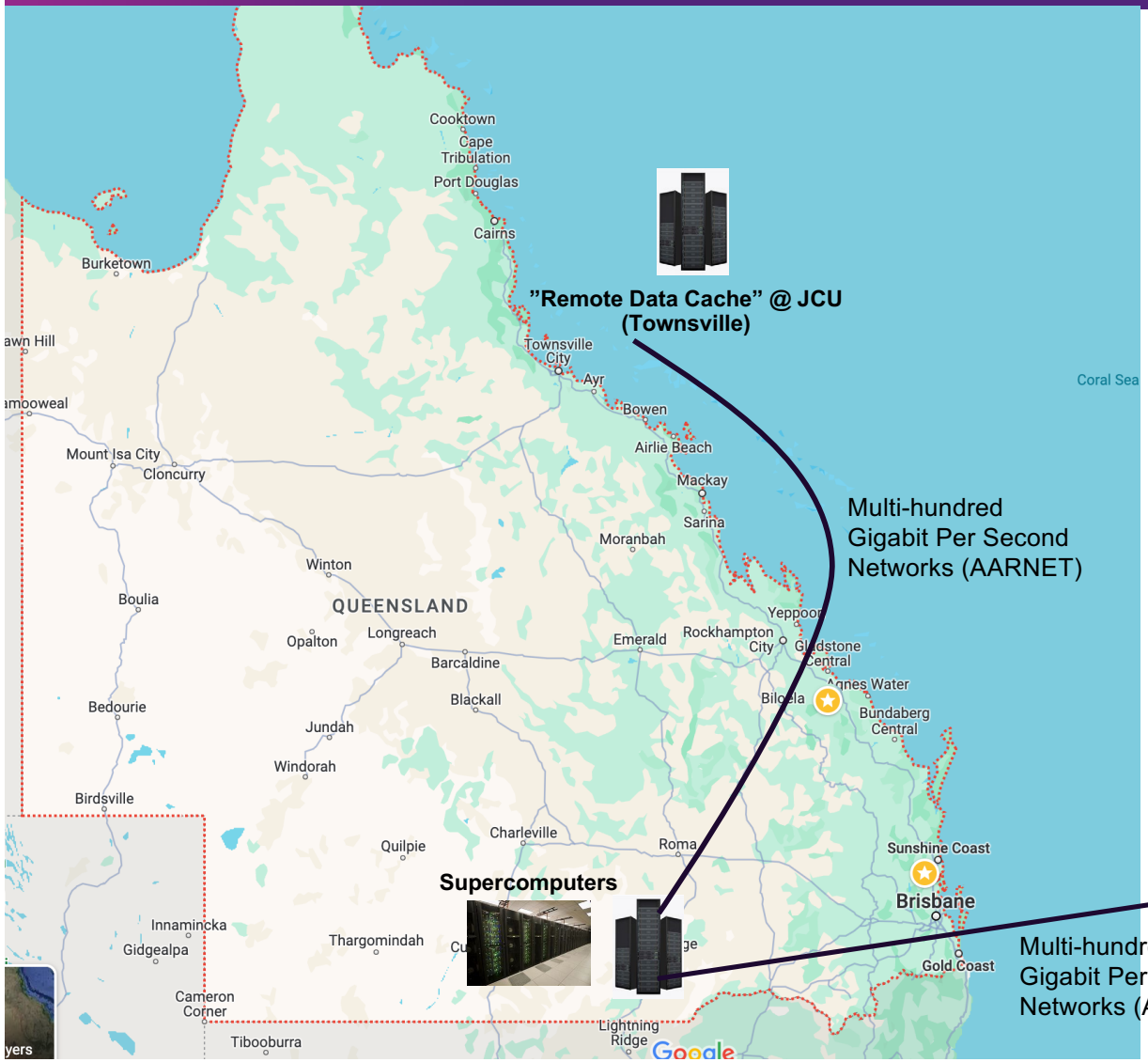


Source: Gartner

Gartner

How does it work?

- UQ uses high speed, high-capacity data caches at the remote organisation linked together via AFM.
- …coupled to Australia's fastest networks to provide near instantaneous data movement to supercomputing, cloud, container and interactive resources.
- Researchers at the remote institution experience their normal workflows, access their data as if it were next to them – but the data is automatically transported to the "home" sites, where it is preserved and available on the advanced computing infrastructure.

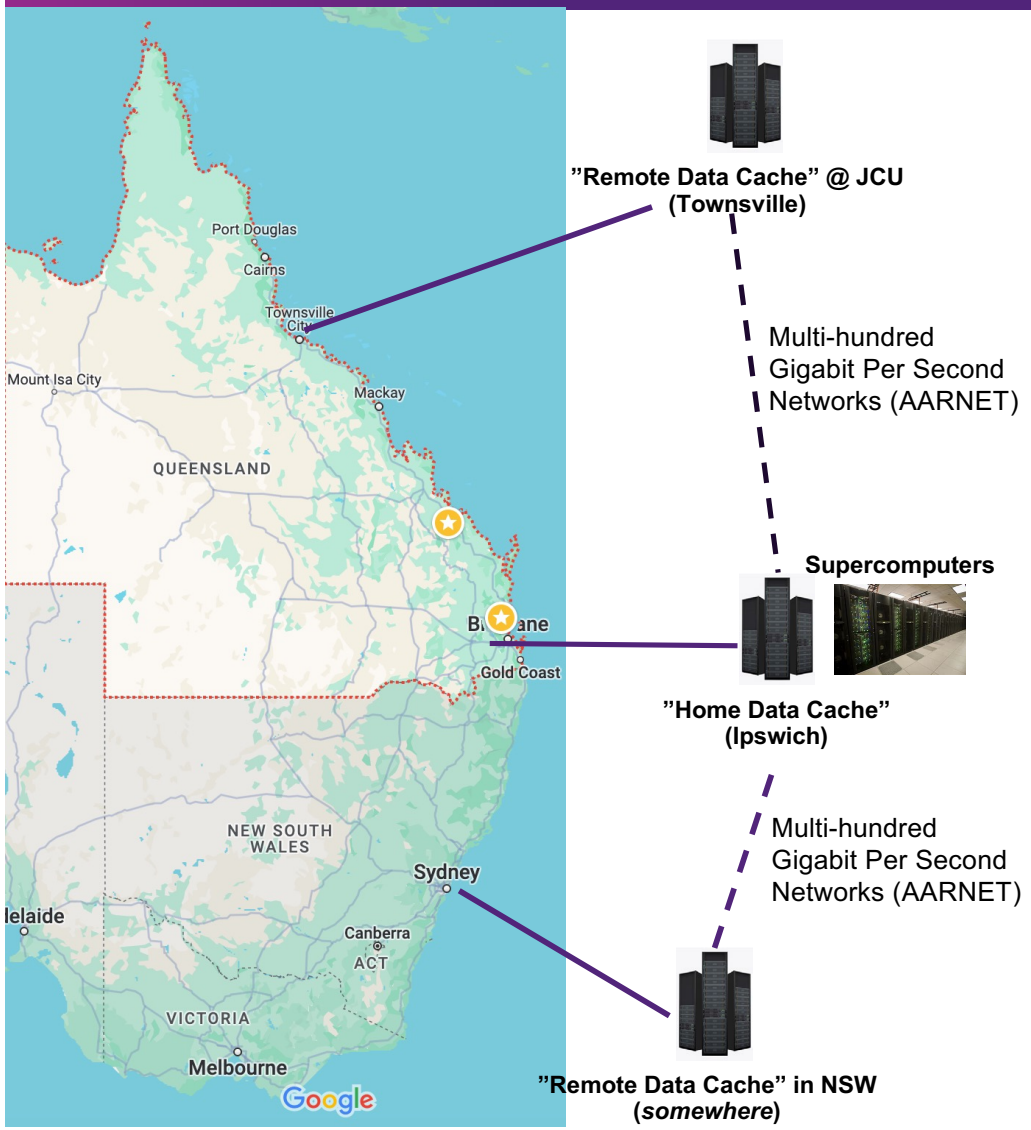


Multi-hundred Gigabit Per Second Networks (AARNET)

Multi-hundred Gigabit Per Second Networks (AARNET)

UQ's AFM based data fabric already spans the east coast of QLD, between James Cook University and Brisbane, providing JCU unprecedented access to UQ's supercomputing and award-winning research storage infrastructure and supercomputers.

UQ Campus (Brisbane), instruments, systems, data generating research infrastructure…



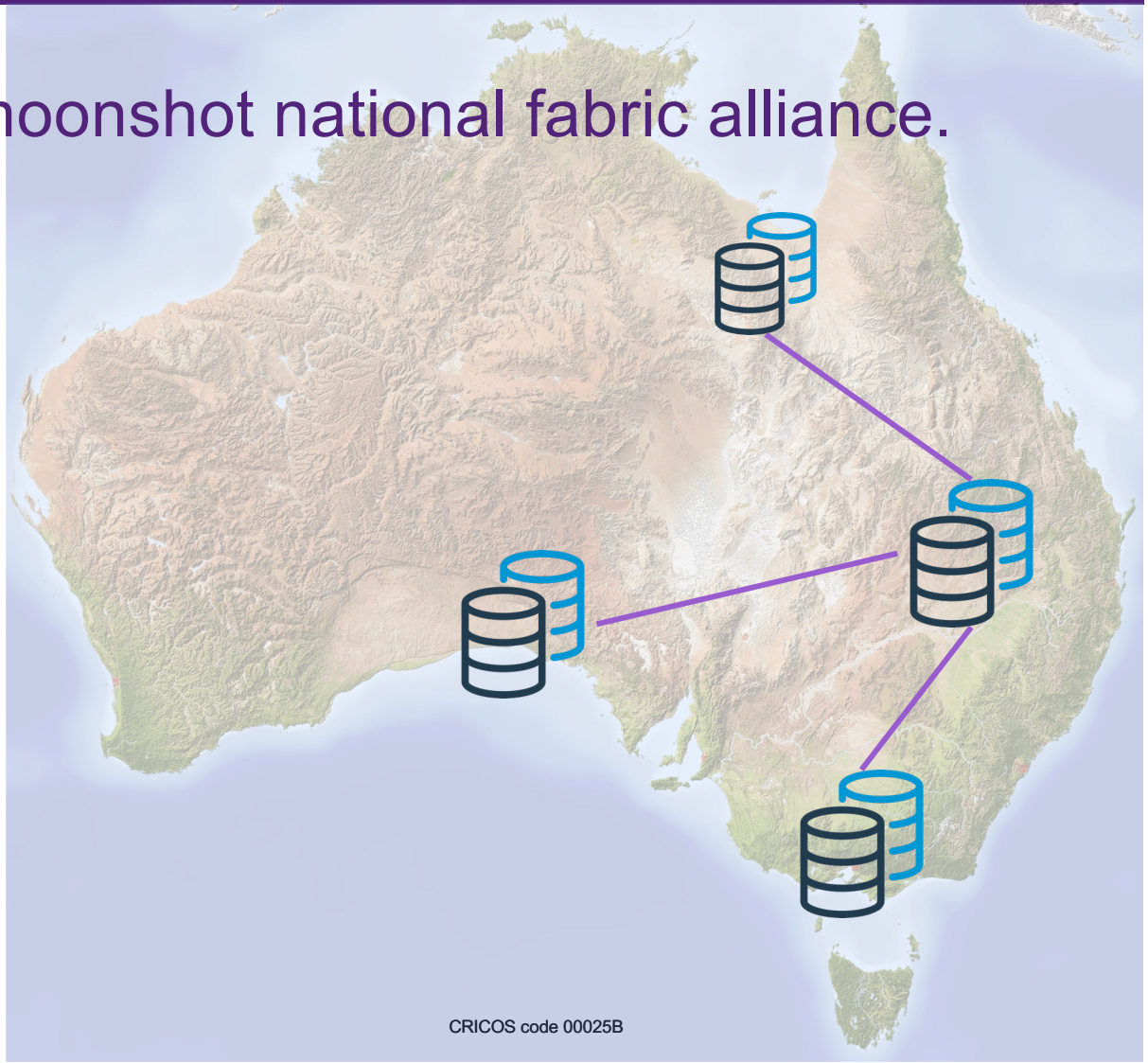
- For UQ to deploy a similar capability to the Southern States it is now relatively trivial.
- It requires a fast network and a data cache at the remote institution.
- How we glue it together in terms of the back end and front end? Thinking…

What's coming next? Our moonshot national fabric alliance.

A robust national scale capability, spanning states, institutions and organisations to achieve an economy of scale, resiliency and distribution unlike anything else Australia has ever seen.

Initial scope:

- A group of IBM ESS building blocks in different institutions.
- S3 → AFM backends.
- Landing zones in very large archival storage platforms eventually.



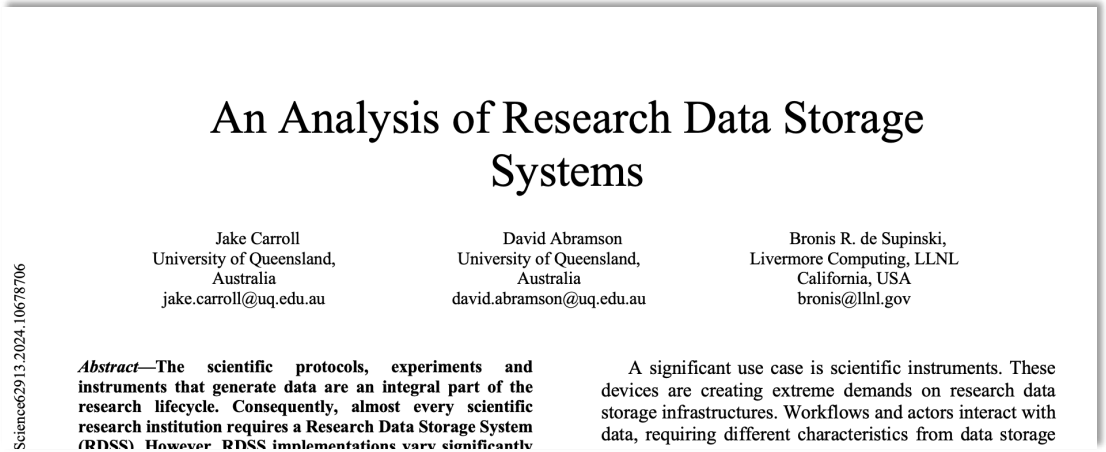
Are you out of **your mind?** Nobody has done that before.

1. Yes. Clearly.
2. I know nobody has done this before. Not at this scale. I'm not scared. *Perhaps I should be.*
3. Our federal government seems keen to get behind this, initially. We found a model that work well for the state. Maybe it can work for the whole country?
4. We will meet many new challenges in bringing a national Storage Scale AFM platform together over a country so big. It is our hope IBM will be there for us, as they have been on the past – on a truly national journey.

If you're interested in the work behind this – or the architectural diligence and principles behind all of this…

DOI: 10.1109/e-Science62913.2024.10678706

- IEEE eScience conference talk, "An Analysis of Research Data Storage Systems" - presented at the NRDPISI-1 Near Real Time Data Processing Storage Infrastructures Workshop in Osaka, Japan on the 17th of September 2024.



An Analysis of Research Data Storage Systems

Jake Carroll
University of Queensland,
Australia
jake.carroll@uq.edu.au

David Abramson
University of Queensland,
Australia
david.abramson@uq.edu.au

Bronis R. de Supinski,
Livermore Computing, LLNL
California, USA
bronis@llnl.gov

***Abstract*—The scientific protocols, experiments and instruments that generate data are an integral part of the research lifecycle. Consequently, almost every scientific research institution requires a Research Data Storage System (RDSS). However, RDSS implementations vary significantly**

A significant use case is scientific instruments. These devices are creating extreme demands on research data storage infrastructures. Workflows and actors interact with data, requiring different characteristics from data storage

Science62913.2024.10678706

Thank you. Time for Q's? If not – find me later if you'd like to chat.