BriCS
Bristol Centre for Supercomputing

Slingshot RDMA on Kubernetes: Serving LLMs with KServe on Isambard-AI

David John

AI Supercomputing Infrastructure Engineer

Bristol Centre for Supercomputing

ACIT Hub Co-design Day

16/09/26

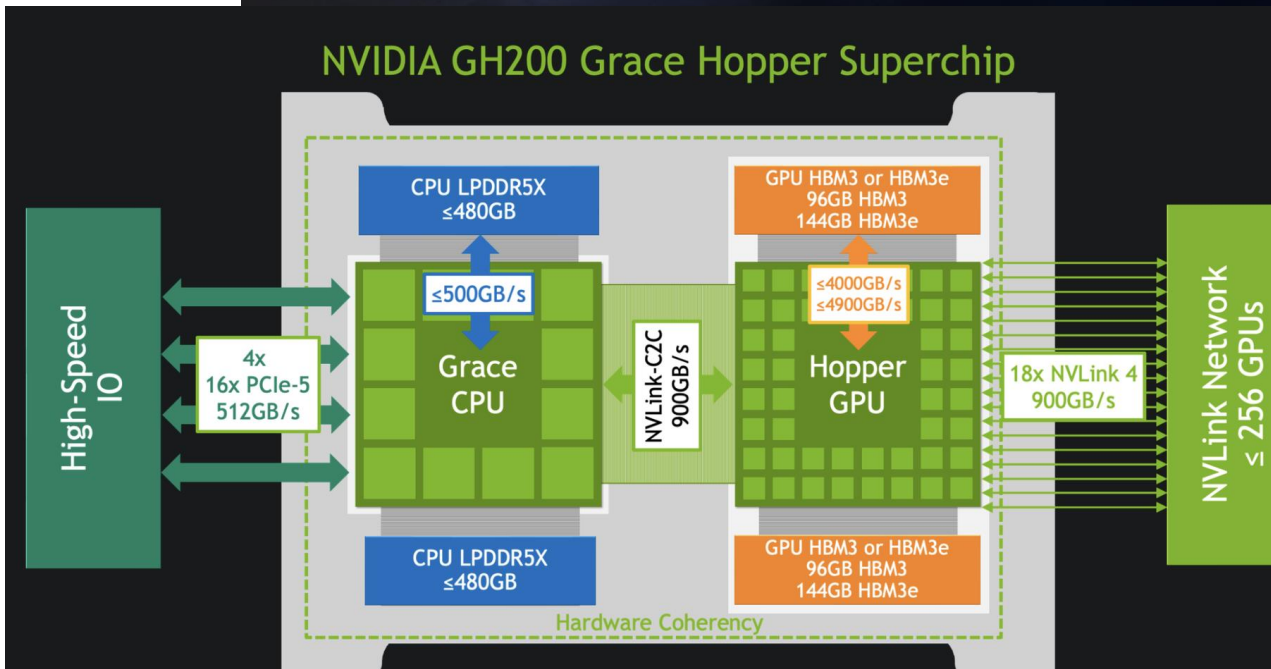
University of BRISTOL

Introduction to Isambard-AI

- Phase 1 → 42 nodes (168 GH200 superchips)
- Phase 2 → 1320 nodes (5280 GH200 superchips)
- Slingshot 11, 4 x 200Gb/s CXI NIC per node
- £225m hardware
- 13th in TOP500 benchmark
- >1000 projects, 4000 users – ranging from vaccine development to AI security research

See our YouTube channel for more case studies.
@BriCS-UoB





Slingshot Cassini 11 and RDMA

NCCL talks GPUDirect RDMA through our aws-ofi-nccl plugin over libfabric to the CXI driver.

Most container images, particularly Nvidia GPU Cloud (NGC) images, bundle support for InfiniBand (IB) verbs. Modifications are required to enable Slingshot RDMA.

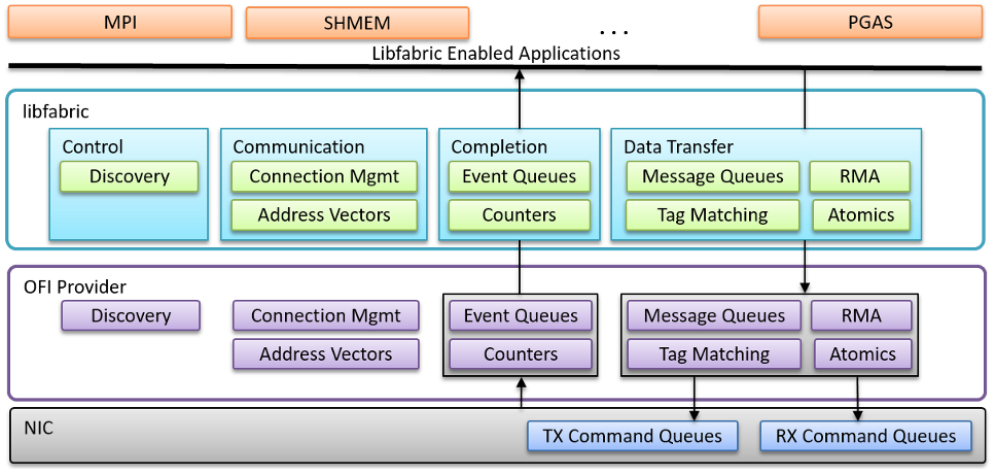
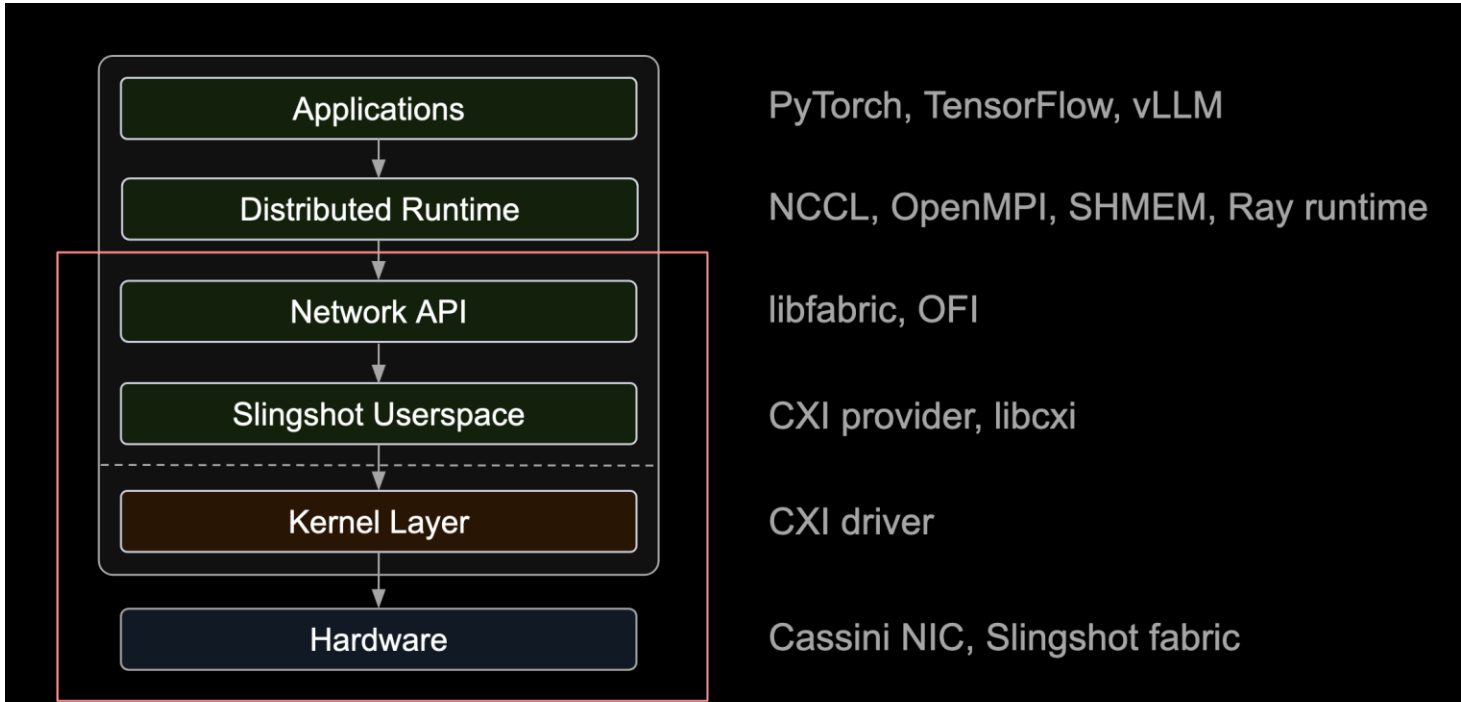


Image from AWS Blog

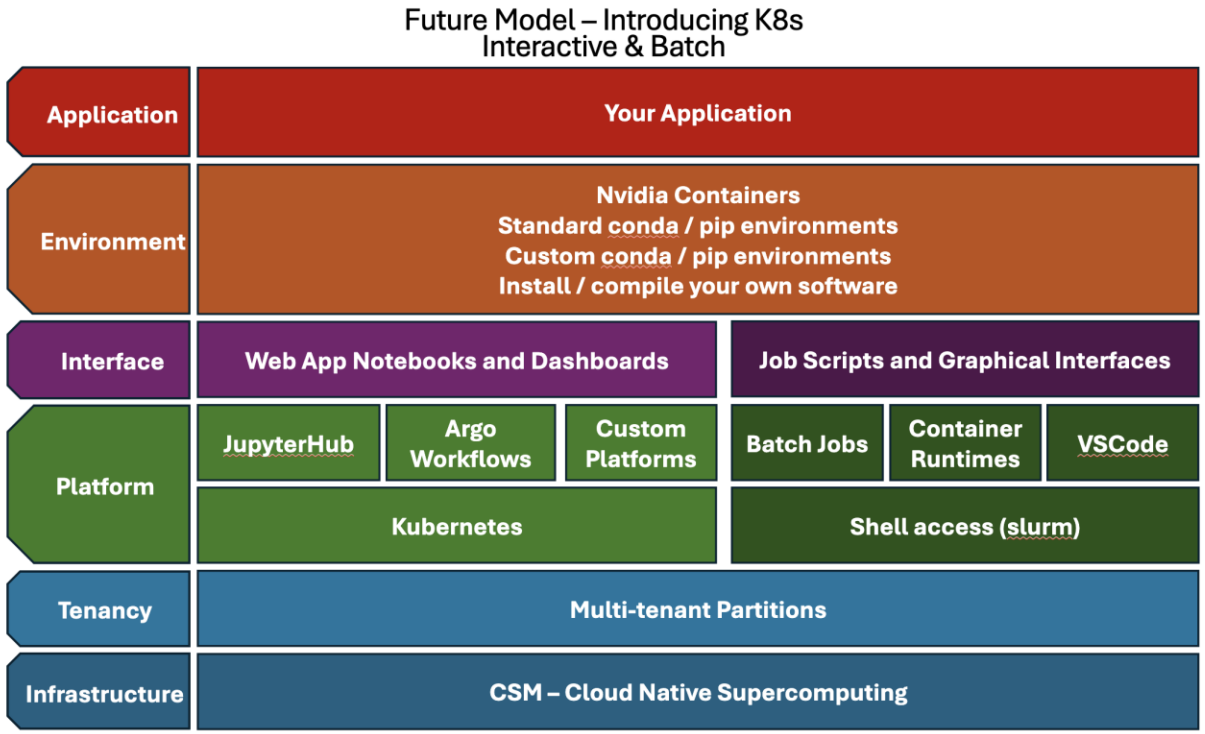


NCCL – NVIDIA Collective Communications Library
RDMA – Remote Direct Memory Access
OFI – Open Fabrics Interface
CXI – Slingshot Cassini 11 Network Interface

Why Kubernetes?

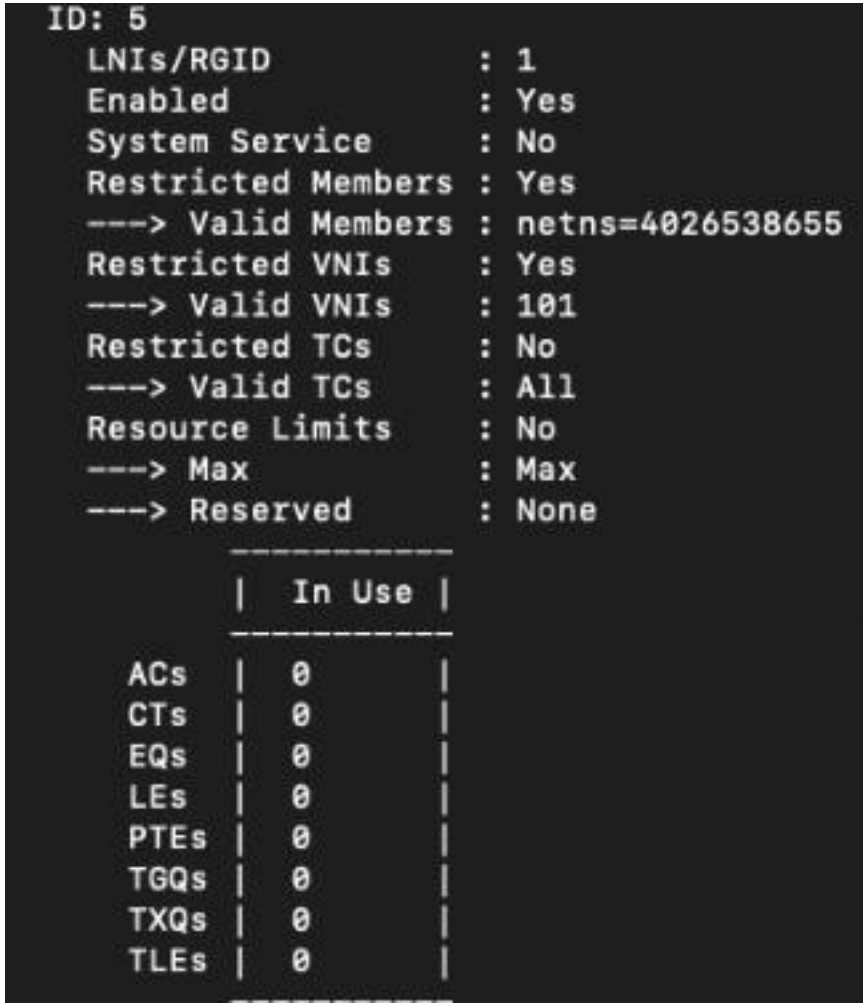
Important Note
Kubernetes on Isambard-AI is still in the early stages of development.

- Isambard-AI users increasingly come from the cloud, not traditional HPC, and want portability across infrastructure (i.e. cloud-native).
- LLM Inference is a highly-requested use-case.
- Provides a basis for us, BriCS, to deliver more platforms for users.



RDMA-backed inference with KServe

- Co-developed with HPE, we built a CXI DRA driver which enables isolated RDMA traffic per namespace; allocating an RDMA domain with dedicated VNI. ***Slurm handles this automatically.***
- It uses container device interfaces (CDIs) to inject required host libraries (aws-ofi-nccl, libfabric, libcxi, etc.) into pods and configures required NCCL and libfabric environment variables.
- Users just need to add ***ResourceClaims*** for GPUs and CXI devices – RDMA is configured automatically.



RDMA-backed inference with KServe (continued)

- KServe ***LLMInferenceService*** CRD provides an abstraction for inference, with configurations to handle:
  - **Multi-node inference** - using ***LeaderWorkerSets***
  - **Prefill-decode (PD) disaggregation** - runs prefill and decode as separate pools of pods, each scaled and scheduled independently, matching their different compute vs. memory-bandwidth profiles
  - **Tensor and pipeline parallelism** - configured declaratively
  - **Autoscaling** - scales replicas based on inference load (built on Knative-style autoscaling)
  - **Load balancing / routing** - a router-scheduler component directs requests across prefill and decode pools
- Combined with the CXI DRA driver, this means a user can request any RDMA-backed inference with a ***ResourceClaims*** block and a ***LLMInferenceService*** spec.

```yaml
apiVersion: serving.kserve.io/v1alpha2
kind: LLMInferenceService
metadata:
  name: gpt-oss-120b-2node8gpu
  namespace: kserve
spec:
  model:
    name: "openai/gpt-oss-120b"
    uri: "pvc://hf-model-cache/openai/gpt-oss-120b"
  replicas: 1
  parallelism:
    tensor: 4
    pipeline: 2
  template:
    resourceClaims:
      - name: cxi
        resourceClaimTemplateName: cxi-slingshot-hpe-com-4-whole
      - name: gpus
        resourceClaimTemplateName: grace-hopper-quad-gpu
    containers:
      - name: main
        resources:
          claims:
            - name: cxi
            - name: gpus
        env:
          - name: TIKTOKEN_ENCODINGS_BASE
            value: /mnt/models/tiktoken
  worker:
    resourceClaims:
      - name: cxi
        resourceClaimTemplateName: cxi-slingshot-hpe-com-4-whole
      - name: gpus
        resourceClaimTemplateName: grace-hopper-quad-gpu
    containers:
      - name: main
        resources:
          claims:
            - name: cxi
            - name: gpus
        env:
          - name: TIKTOKEN_ENCODINGS_BASE
            value: /mnt/models/tiktoken
```

Example: Qwen3 PD-Disagg

Qwen3-235B-A22B-Thinking-2507-FP8.

- Running across two Isambard-AI Phase 1 nodes (2 x 4 GH200s).
- Prefill-decode (PD) disaggregation (one node dedicated to prefill, one to decode).
- Model split across all four GPUs on each node using tensor parallelism (TP).

```
qwen3-235b-a22b-pd-single-eager-kserve-7b7f4b58f5-pwmvn          ●  2/2   Running   10.42.31.77   nid001025  18d
qwen3-235b-a22b-pd-single-eager-kserve-prefill-59c5dbd4c5-7kx46  ●  1/1   Running   10.42.11.128  nid001009  18d
qwen3-235b-a22b-pd-single-eager-kserve-router-scheduler-86fqwn5  ●  1/1   Running   10.42.9.87    nid001003  18d
```

See KServe documentation on PD-Disagg.

NCCL All Reduce (K8s vs. Bare Metal)

Between two nodes, single GPU+NIC to single GPU +NIC

Size	Bare Metal (GB/s)	K8s (GB/s)	Delta
128 KiB	1.55	0.44	-72%
512 KiB	3.58	1.67	-53%
1 MiB	6.64	6.12	-8%
4 MiB	17.80	17.03	-4%
16 MiB	20.97	20.73	-1%
64 MiB	22.75	22.69	-0.3%
256 MiB	23.08	23.10	+0.1%
1 GiB	23.21	22.35	-4%
4 GiB	23.01	22.41	-3%
8 GiB	23.13	22.56	-2%

K8s overhead is <5% for messages ≥1 MiB.
This is important for multi-node LLM inference, due to large tensor collectives such as KV-cache transfers.

Final Notes

- RDMA-backed inference on Kubernetes has comparable performance to bare-metal.
- CXI DRA driver is the key enabler for RDMA on Kubernetes.
- Kubernetes on compute development is still in the early stages on Isambard-AI.