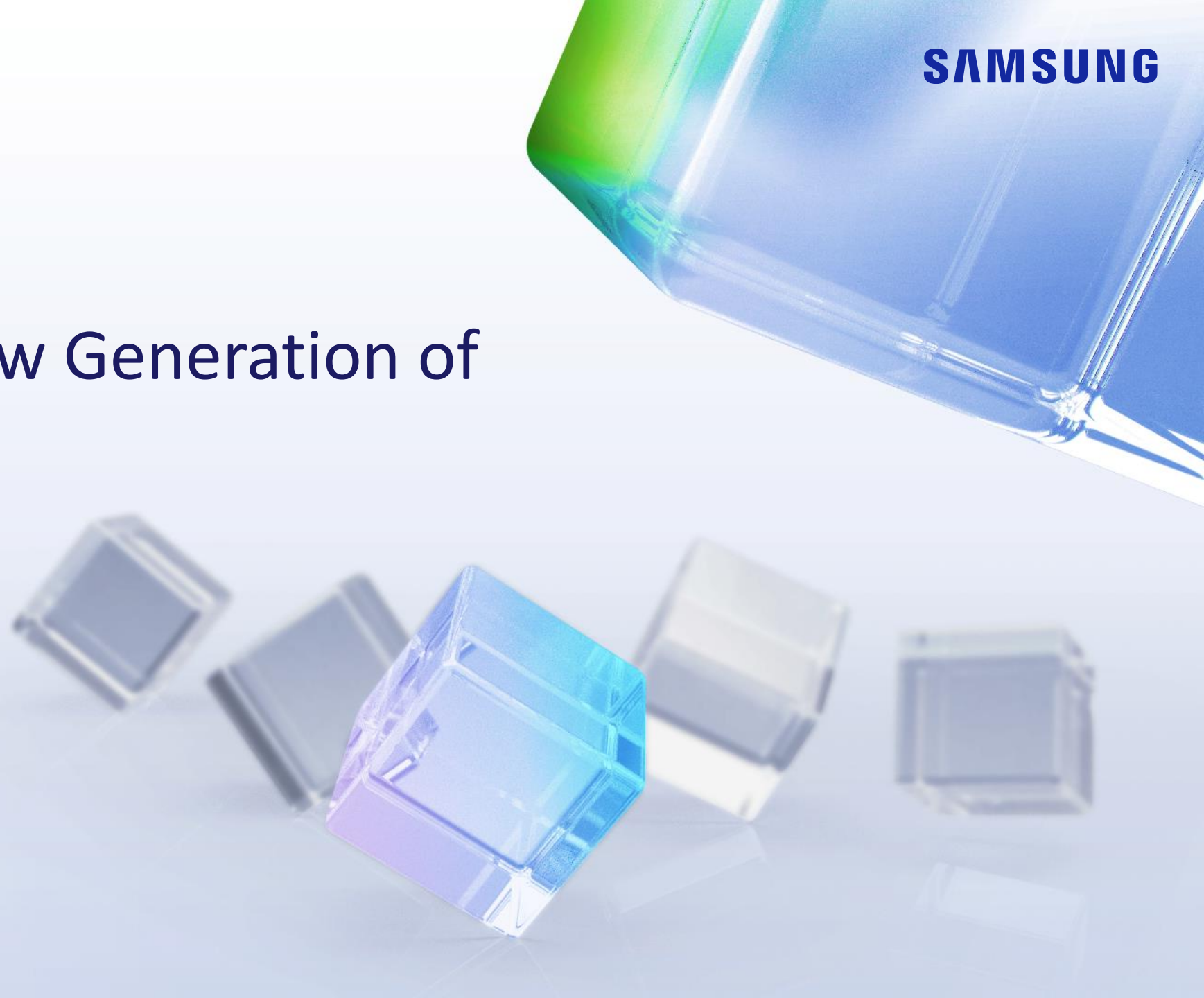


Storage for a New Generation of AI/ML

Somnath Roy

Principle Engineer,
Memory Solutions Lab,
Samsung

August 4th 2022



Current State of AI/ML

Focus on Large-Scale AI/ML (at least > 1PB storage for training data)

- Large-Scale Use cases:
 - Fraud prevention and risk analysis
 - Natural Language Processing
 - Real-time price optimization
 - Autonomous driving

Compute has evolved rapidly with new algorithms and GPUs

- In fact with the advent of GPU direct, NVIDIA is claiming bottleneck is on storage

Can large-scale storage keep up with compute?

- High read BW requirement (>1TB/s per rack) for running AI training at scale with thousands of GPUs in parallel

DSS: Performant & Scalable Object Storage

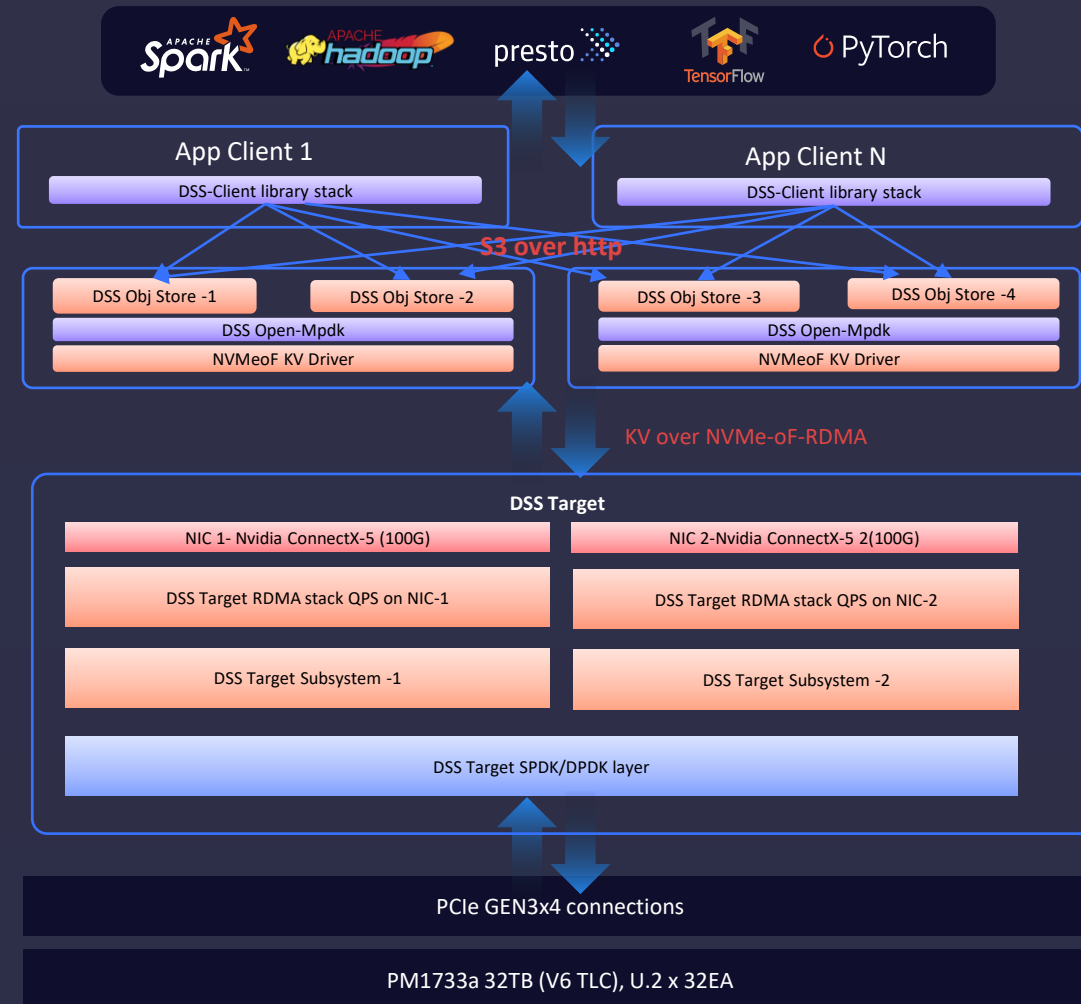
Disaggregated Storage Solution(DSS)

Services

- Samsung developed – open sourced <https://github.com/OpenMPDK/DSS>
- NVMeoF based S3 Service
- High Read Throughput Object Storage
- Disaggregated Storage and compute
- Shared everything architecture
- Zero copy Key-Value transfer
- Easy Scaling at Exabytes

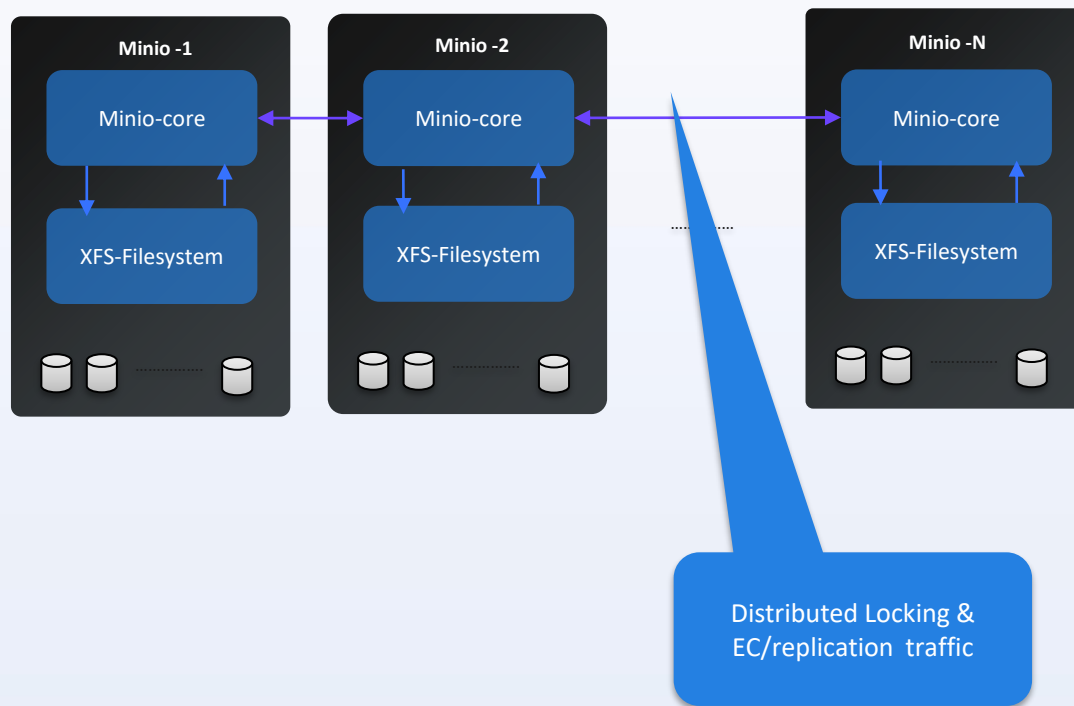
Use Cases

- Large scale high READ throughput AI training
- Image Analytics
- Audio/Video AI
- Metaverse

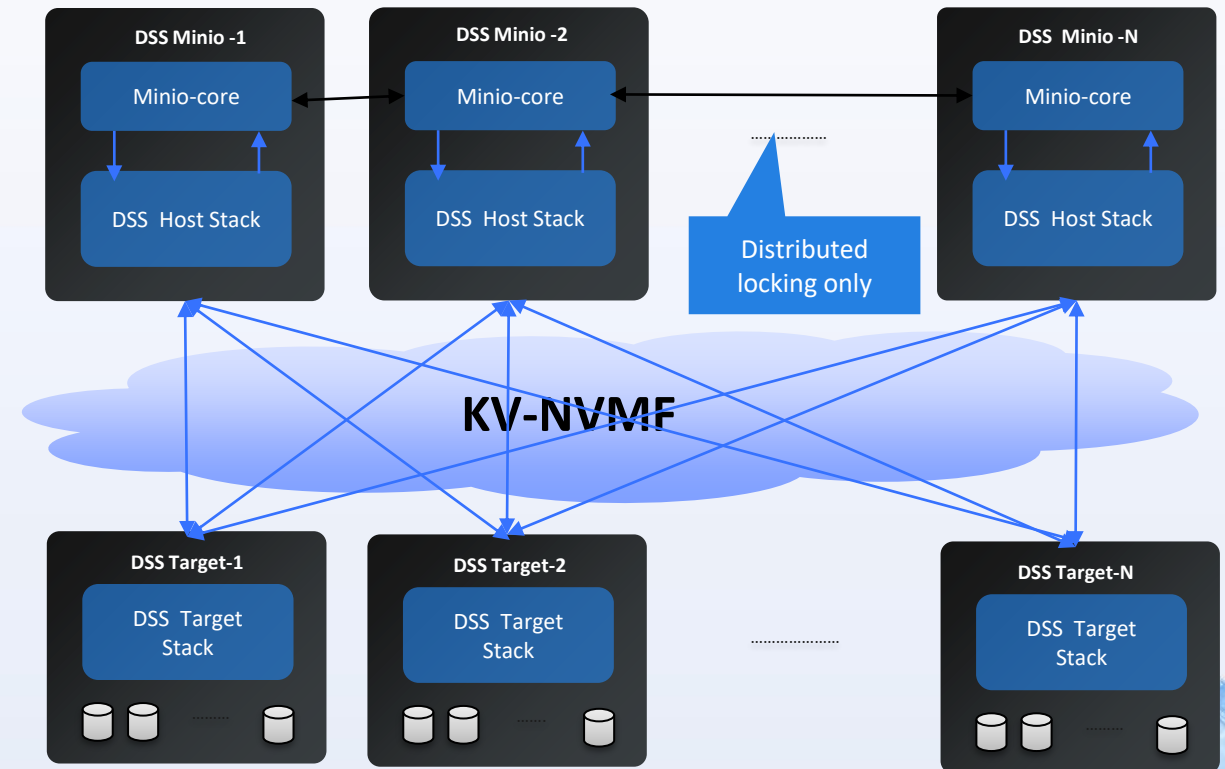


DSS Enhanced Minio Object-Store

Stock Minio Shared-nothing architecture (Compute has to grow along with storage)



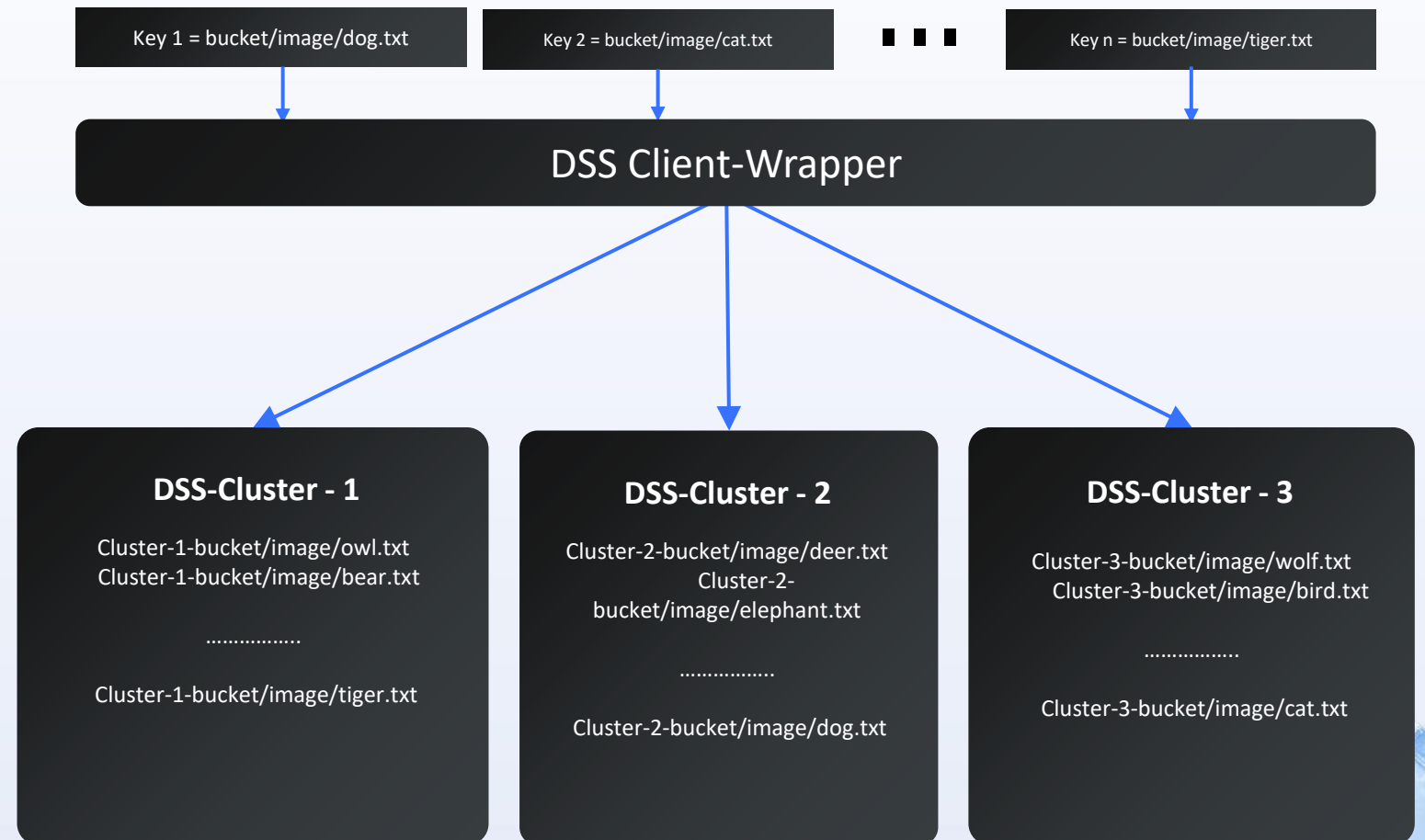
DSS Minio disaggregated, Shared-everything architecture (Compute and storage can grow independently)



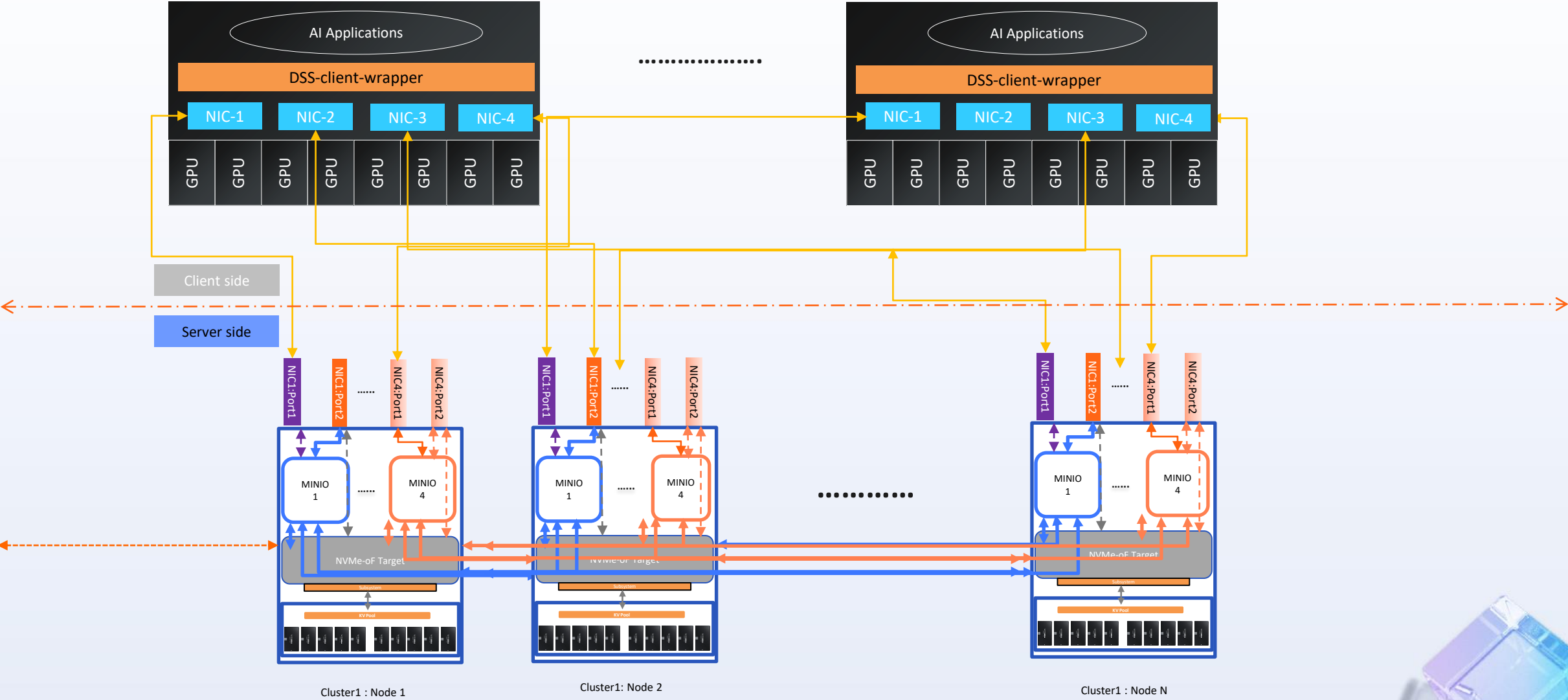
DSS Client Wrapper



- Bucket Abstraction
- Key Distribution
- Cluster Expansion
- Rebalance
- Standard S3 Operations (Get/Put etc.)



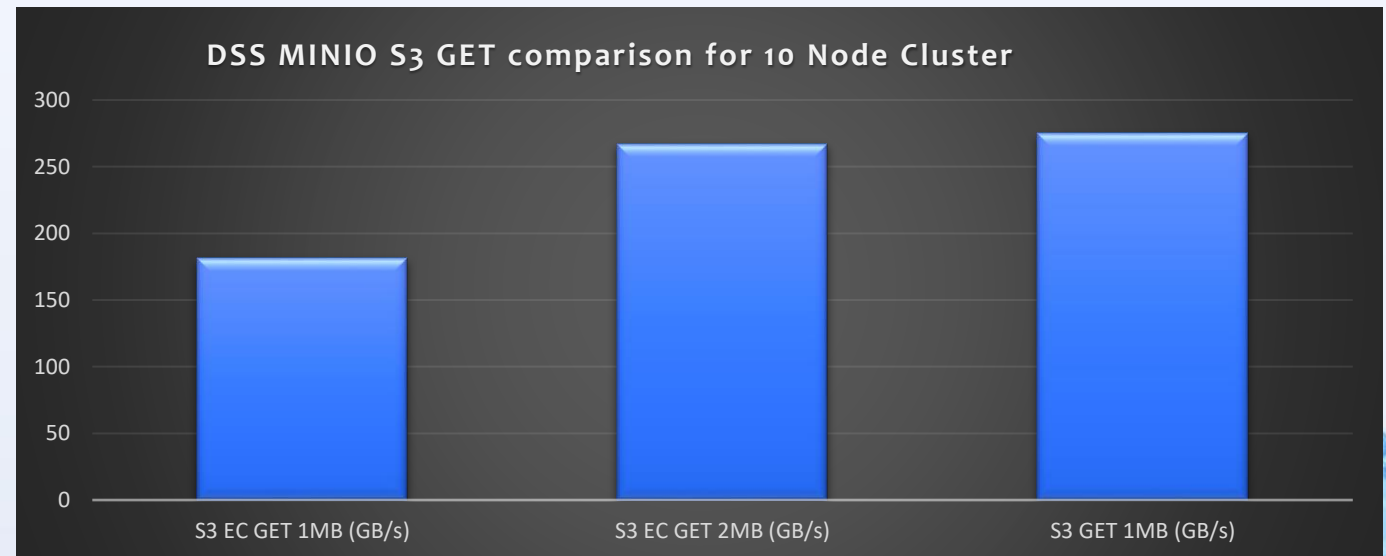
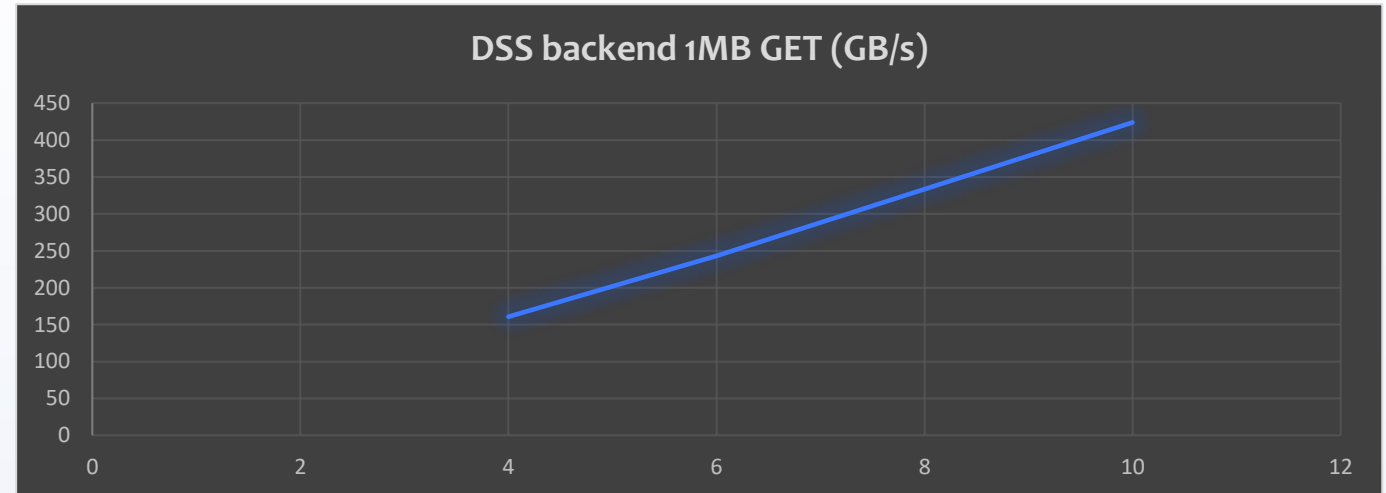
DSS Deployment View



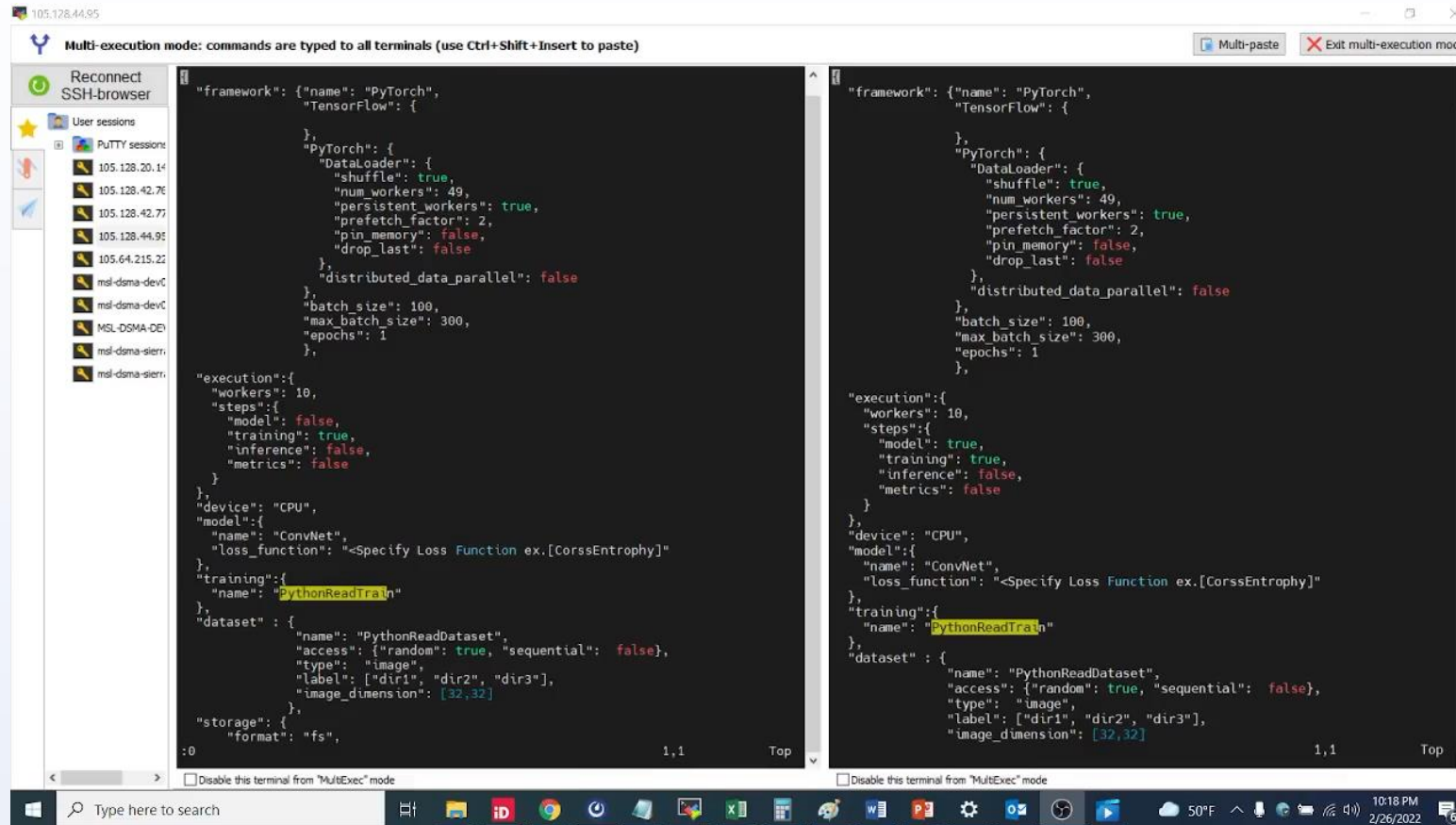
DSS GET Performance

Setup

- Client - 16x Dell PowerEdge R6525, 2 x DGX A100
- **DSS S3 Server**
 - 10x Dell PowerEdge R7525 Gen4 servers
 - Dual socket AMD EPYC 7742 64-Core
 - 1TB physical memory
 - 4xMellanox Dual port 200Gb (ConnectX-6)
- SSD - 16x PM1733 4TB Gen4 NVMe SSD per DSS S3 server
- Total data set generated during test ~400TB
- Top chart is just DSS backend performance across 10 node, no S3 involved
- Tool used home grown dss test cli
- Bottom one with DSS optimized Minio
- Tool used standard S3-benchmark



AI Benchmarking Tool



The screenshot displays the AI Benchmarking Tool interface. On the left, a sidebar shows a list of user sessions and Putty sessions. The main area contains two terminal windows, each displaying a JSON configuration for a PyTorch training job. The configuration includes details about the framework (PyTorch), data loader (DataLoader), execution (workers, steps), device (CPU), model (ConvNet), training (PythonReadTrain), dataset (PythonReadDataset), and storage (fs). The left terminal window shows a configuration with "training": true and "inference": false. The right terminal window shows a configuration with "training": true and "inference": false. Both configurations specify a batch size of 100, a max batch size of 300, and 1 epoch. The interface also includes a "Multi-execution mode" button and a "Multi-paste" button.

```
{
  "framework": {
    "name": "PyTorch",
    "TensorFlow": {
      "name": "TensorFlow",
      "DataLoader": {
        "name": "DataLoader",
        "shuffler": true,
        "num_workers": 49,
        "persistent_workers": true,
        "prefetch_factor": 2,
        "pin_memory": false,
        "drop_last": false
      },
      "distributed_data_parallel": false
    },
    "batch_size": 100,
    "max_batch_size": 300,
    "epochs": 1
  },
  "execution": {
    "workers": 10,
    "steps": {
      "model": false,
      "training": true,
      "inference": false,
      "metrics": false
    }
  },
  "device": "CPU",
  "model": {
    "name": "ConvNet",
    "loss_function": "<Specify Loss Function ex.[CorssEntropy]"
  },
  "training": {
    "name": "PythonReadTrain"
  },
  "dataset": {
    "name": "PythonReadDataset",
    "access": {
      "random": true,
      "sequential": false
    },
    "type": "image",
    "label": ["dir1", "dir2", "dir3"],
    "image_dimension": [32, 32]
  },
  "storage": {
    "format": "fs",
    "name": "fs"
  }
}
```

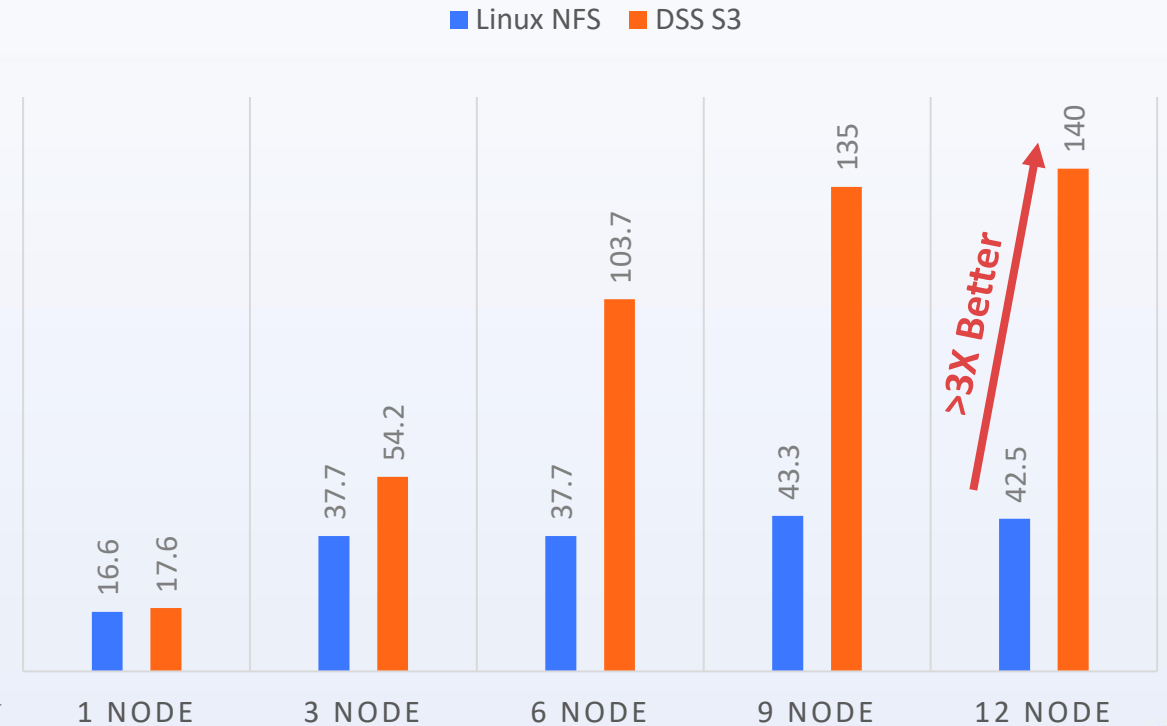
- Benchmarking various storage solution based on NFS, S3 at AI training level
- Platform where developers can add their ML framework, custom data set, training method, models and storage backend
- Demo is showing a custom training with a custom data set that is only capturing data load time and BW from storage servers on NFS/S3

DSS S3 vs Standard NFS

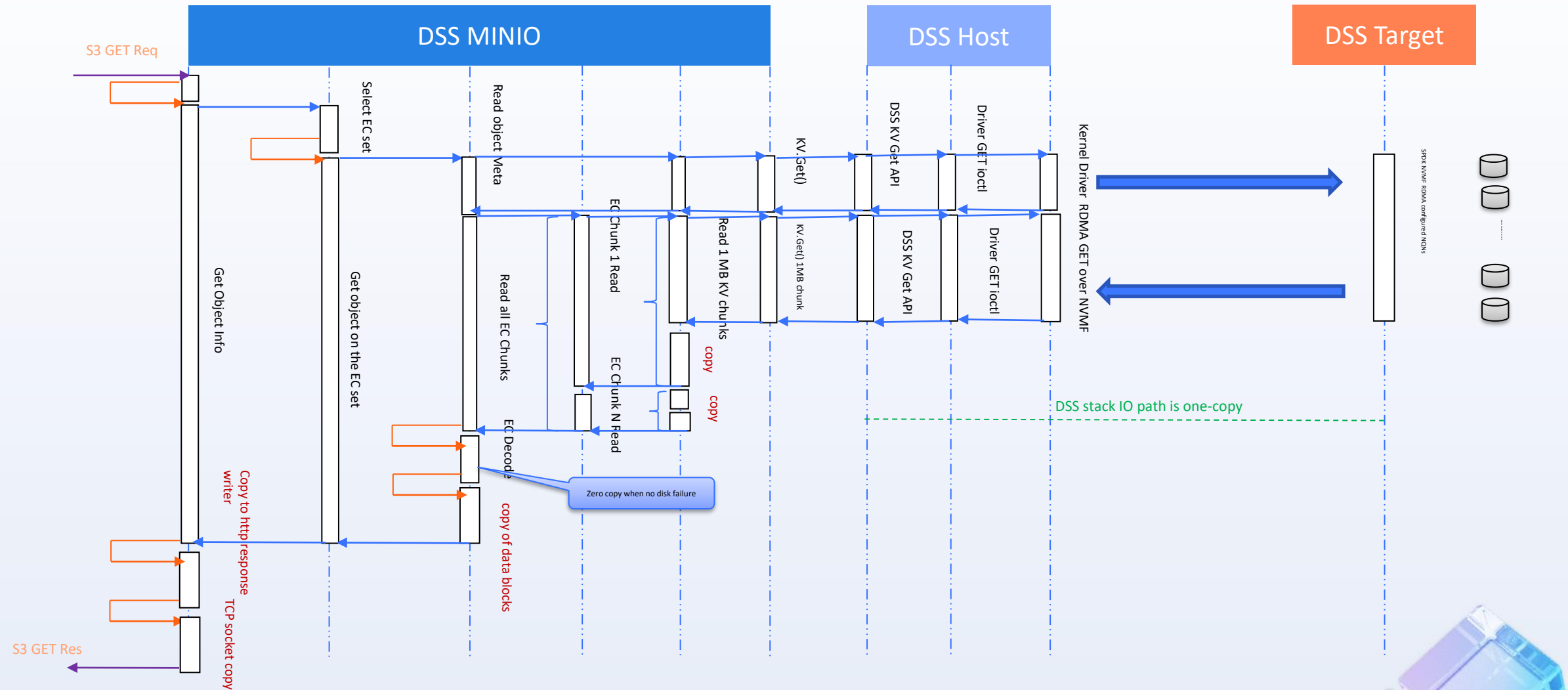
Setup

- Client – 12xDell PowerEdge 740xd
- DSS S3 Server
 - 6x Dell PowerEdge R7525
 - AMD EPYC 7742 64-Core
 - Mellanox Dual port 200g (ConnectX-6)
- NFS server –
 - 6xDell PowerEdge R6525
 - AMD EPYC 7742 64-Core
 - Mellanox Dual port 200g (ConnectX-6)
- SSD - PM1733 4TB NVMe SSD

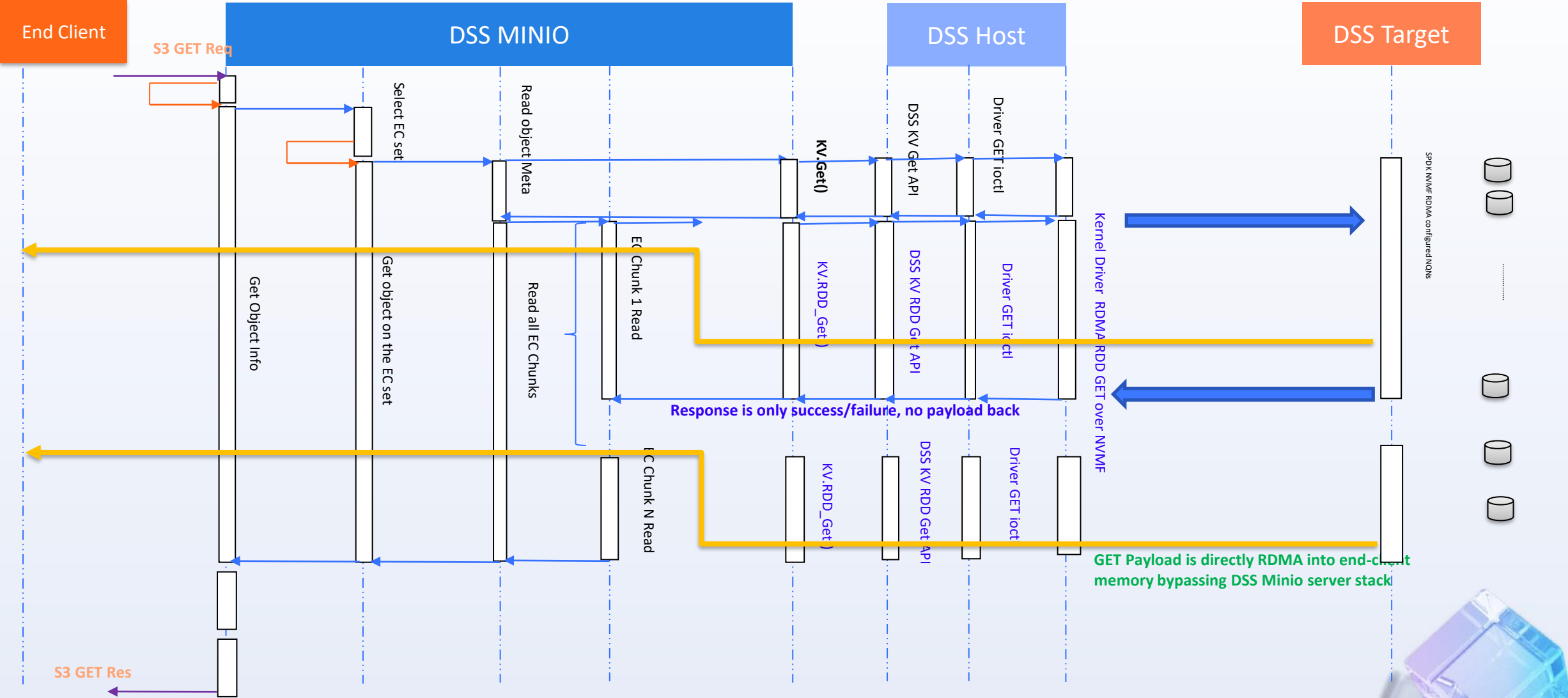
Scaling client
nodes



IO flow during S3 GET request



IO flow during S3 GET request for next Gen DSS



DSS Availability

Open Source Announcement <https://github.com/OpenMPDK/DSS>

- <https://github.com/OpenMPDK/dss-sdk>
- <https://github.com/OpenMPDK/dss-ansible>
- <https://github.com/OpenMPDK/dss-minio>
- <https://github.com/OpenMPDK/dss-ecosystem>

Complete Ecosystem

- AI Benchmarking Framework supporting user preferred training and models
- Client Wrappers supporting Pytorch and Tensorflow
- Host and Target Stack



Thank You

(som.roy@samsung.com)



Back up

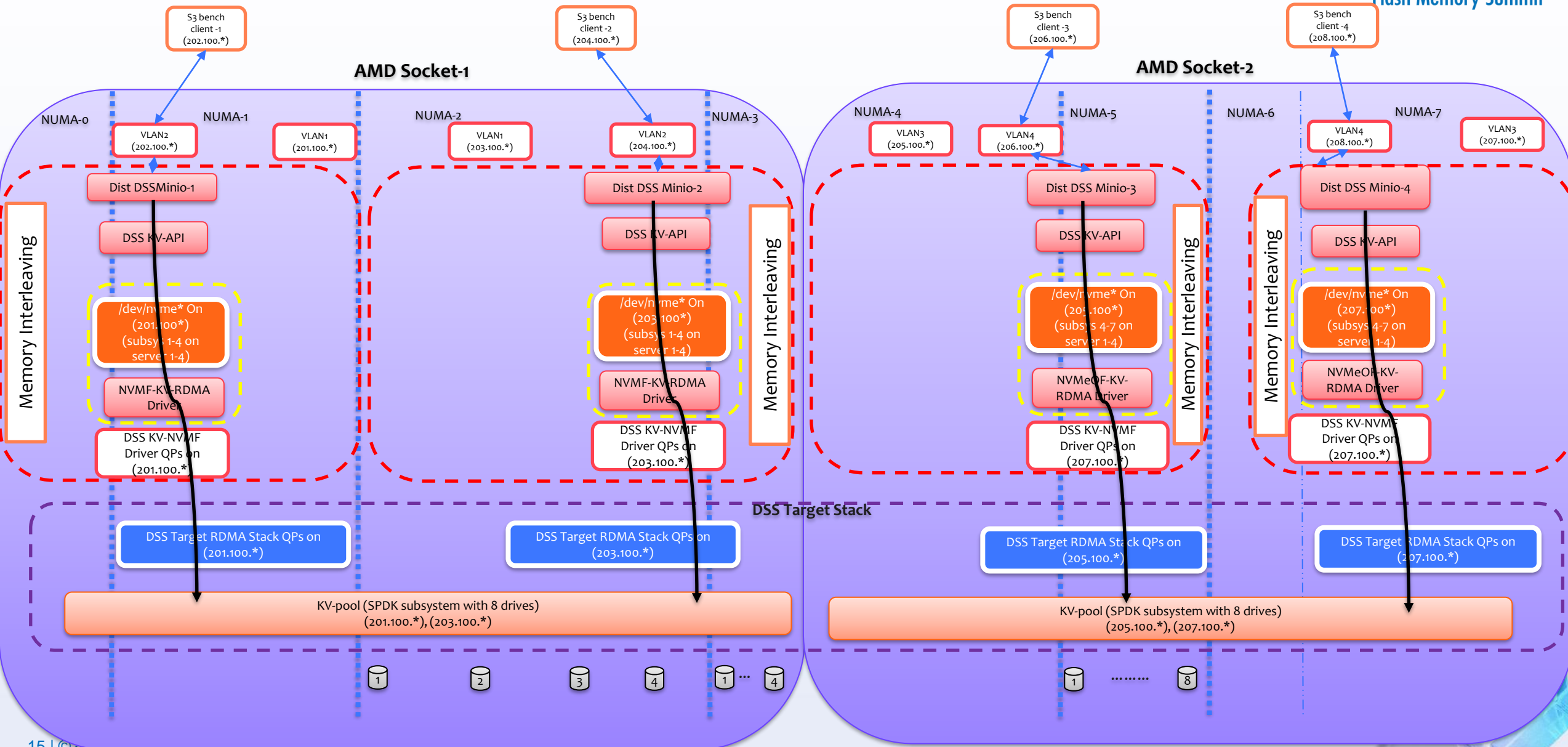
(if time permits)



Reference Minio + DSS deployment model for AMD



Flash Memory Summit



SAMSUNG

