Deep Learning/AI Lifecycle with Dell EMC and bitfusion

Bhavesh Patel Dell EMC Server Advanced Engineering

Abstract

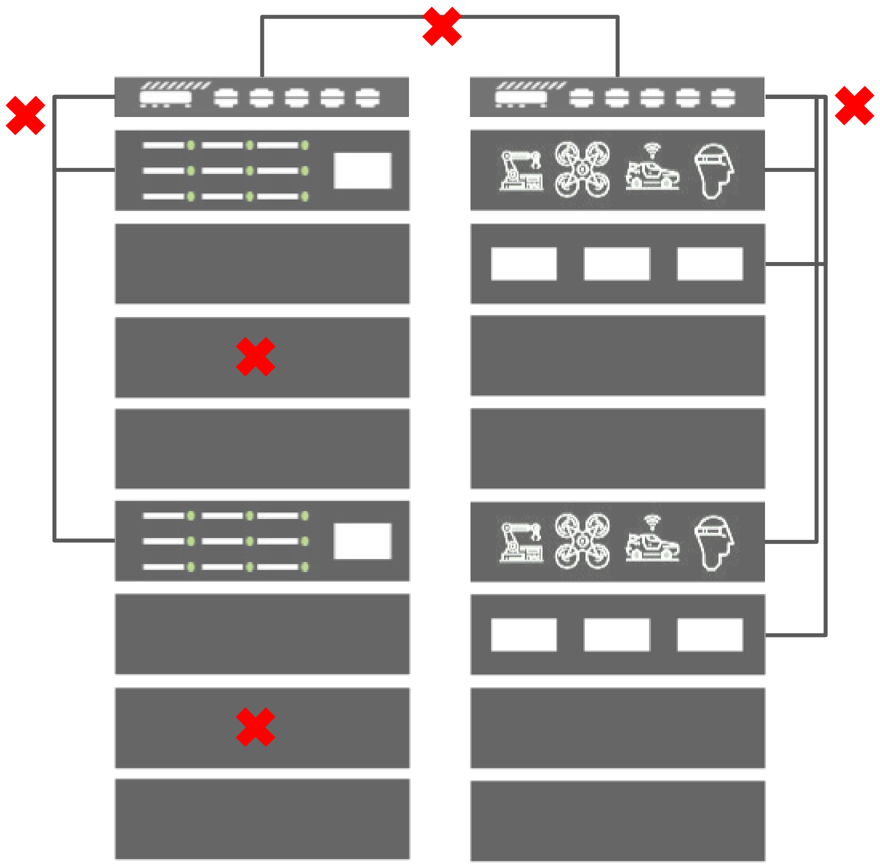
This talk gives an overview of the end to end application life cycle of deep learning in the enterprise along with numerous use cases and summarizes studies done by Bitfusion and Dell on a high performance heterogeneous elastic rack of DellEMC PowerEdge C4130s with Nvidia GPUs. Some of the use cases that will be talked about in detail will be ability to bring on-demand GPU acceleration beyond the rack across the enterprise with easy attachable elastic GPUs for deep learning development, as well as the creation of a cost effective software defined high performance elastic multi-GPU system combining multiple DellEMC C4130 servers at runtime for deep learning training.

Deep Learning and AI Are being adopted across a wide range of market segments

Industry/Function	AI Revolution
ROBOTICS	Computer Vision & Speech, Drones, Droids
ENTERTAINMENT	Interactive Virtual & Mixed Reality
AUTOMOTIVE	Self-Driving Cars, Co-Pilot Advisor
FINANCE	Predictive Price Analysis, Dynamic Decision Support
PHARMA	Drug Discovery, Protein Simulation
HEALTHCARE	Predictive Diagnosis, Wearable Intelligence
ENERGY	Geo-Seismic Resource Discovery
EDUCATION	Adaptive Learning Courses
SALES	Adaptive Product Recommendations
SUPPLY CHAIN	Dynamic Routing Optimization
CUSTOMER SERVICE	Bots And Fully-Automated Service
MAINTENANCE	Dynamic Risk Mitigation And Yield Optimization

…but few people have the time, knowledge, resources to even get started

PROBLEM 1: HARDWARE INFRASTRUCTURE LIMITATIONS



- Increased cost with dense servers
- TOR bottleneck, limited scalability
- Limited multi-tenancy on GPU servers (limited CPU and memory per user)
- Limited to 8-GPU applications
- Does not support GPU apps with:
    - High storage, CPU, Memory requirements

PROBLEM 2: SOFTWARE COMPLEXITY OVERLOAD

Software Management
GPU Driver Management
Framework & Library Installation
Deep Learning Framework Configuration
Package Manager
Jupyter Server or IDE Setup

Model Management
Code Version Management
Hyperparameter Optimization
Experiment Tracking
Deployment Automation
Deployment Continuous Integration

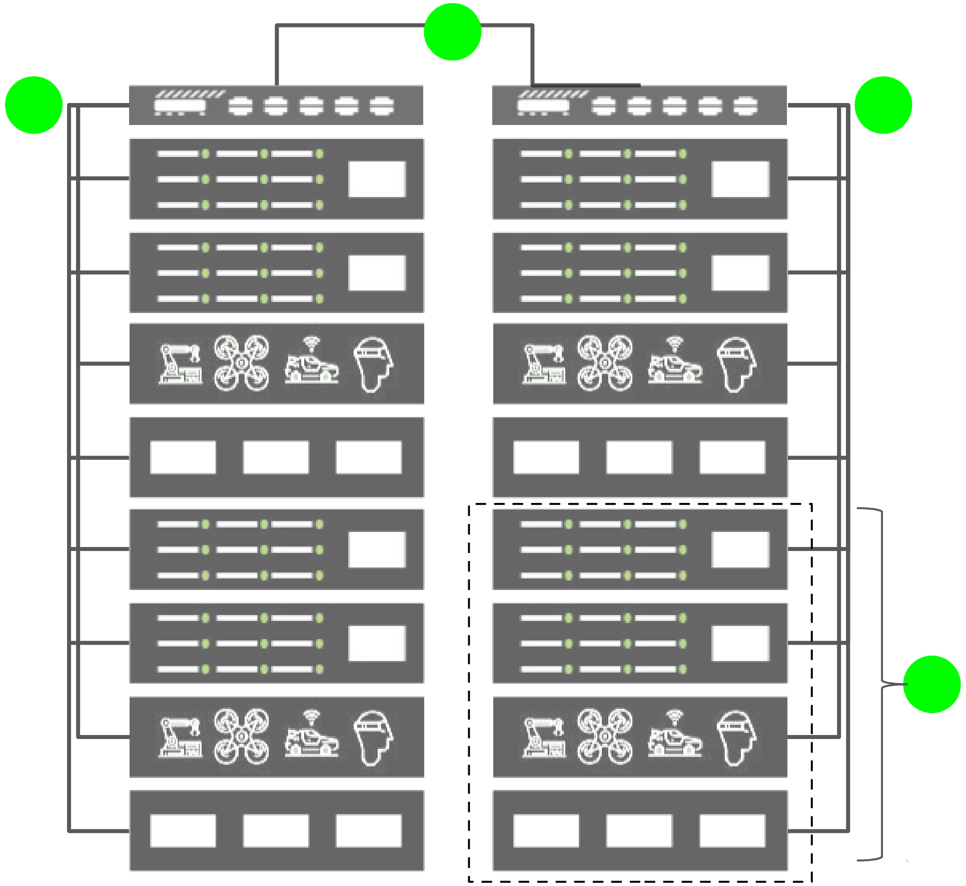
Infrastructure Management
Cloud or Server Orchestration
GPU Hardware Setup
GPU Resource Allocation
Container Orchestration
Networking Direct Bypass
MPI / RDMA / RPI / gRPC
Monitoring

Data Management
Data Uploader
Shared Local File System
Data Volume Management
Data Integrations & Pipelining

Workload Management
Job Scheduler
Log Management
User & Group Manageme
Inference Autoscaling

Need to Simplify and Scale

SOLUTION 1/2: CONVERGED RACK SOLUTION

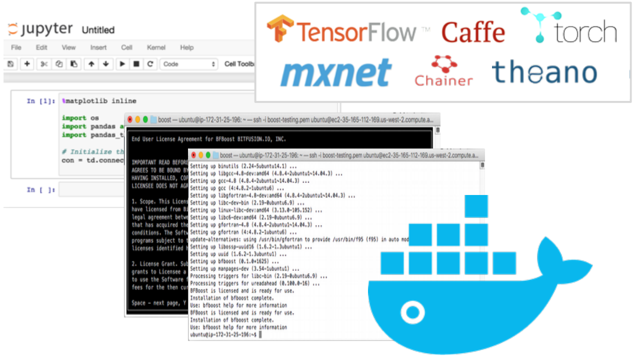


Composable compute bundle

- Up to 64 GPUs per application
- GPU applications with varied storage, memory, CPU requirements
- 30-50% less cost per GPU
- > {cores, memory} / GPU
- >> intra-rack networking bandwidth
- Less inter-rack load
- Composable - Add-as-you-go

SOLUTION 2/2: COMPLETE, STREAMLINED AI DEVELOPMENT

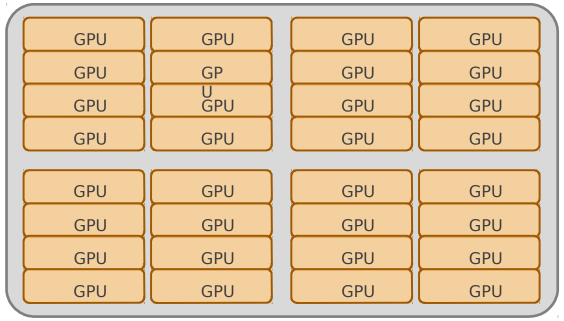
1 DEVELOP



Develop on pre-installed, quick start deep learning containers.

- Get to work quickly with workspaces with optimized pre-configured drivers, frameworks, libraries, and notebooks.
- Start with CPUs, and attach Elastic GPUs on-demand.
- All your code and data is saved automatically and sharable with others.

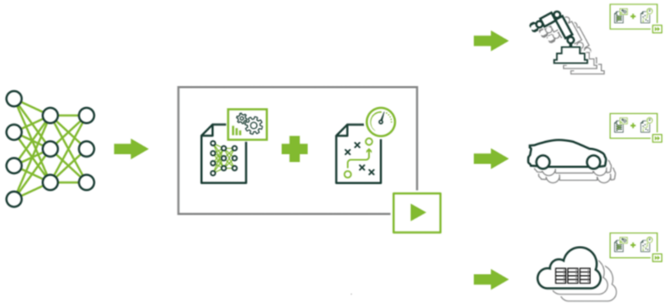
2 TRAIN



Transition from development to training with multiple GPUs.

- Seamlessly scale out to more GPUs on a shared training cluster to train larger models quickly and cost-effectively.
- Support and manage multiple users, teams, and projects.
- Train multiple models in parallel for massive productivity improvements

3 DEPLOY

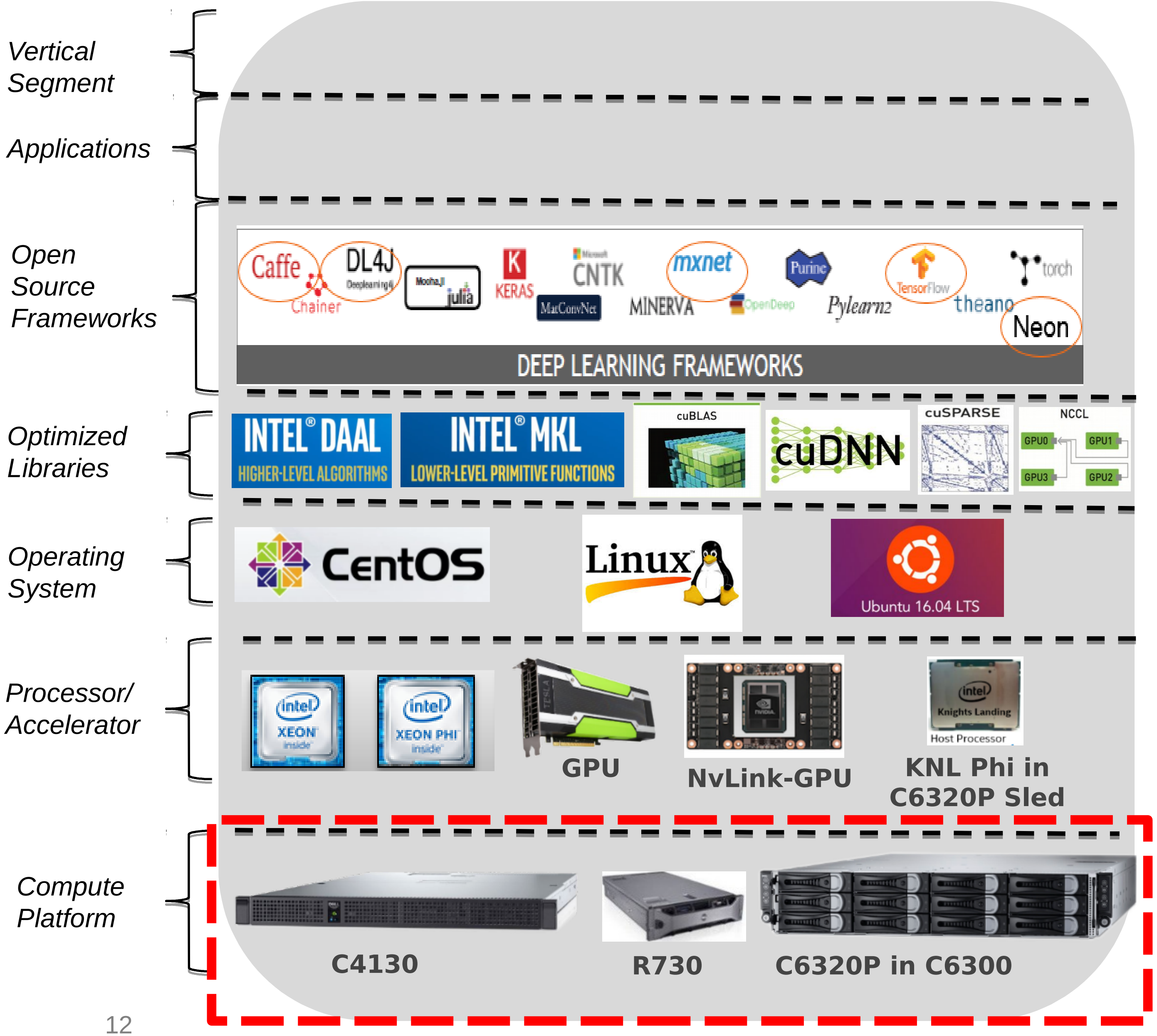


Push trained, finalized models into production.

- Deploy a trained neural network into production and perform real-time inference across different hardware.
- Manage multiple AI applications and inference endpoints corresponding to different trained models.

DELLEMC

Dell EMC Deep Learning Optimized servers



DELL EMC

C4130 DEEP LEARNING Server

Front

Back

Dual SSD boot drives

IDRAC NIC

2x 1Gb NIC

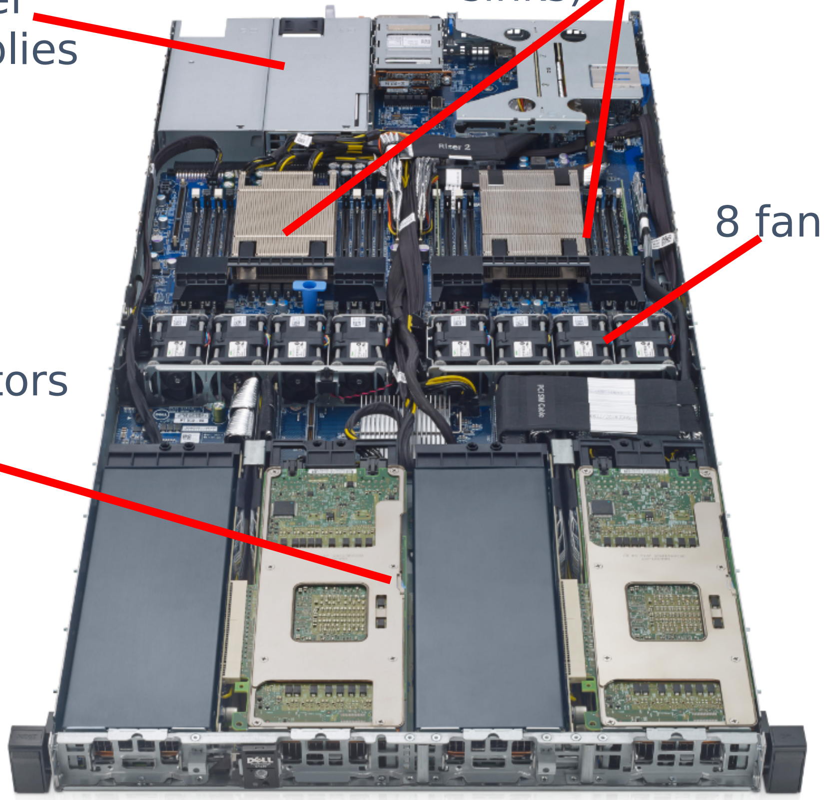
(optional) Redundant Power Supplies

Power Supplies

CPU sockets (under heat sinks)

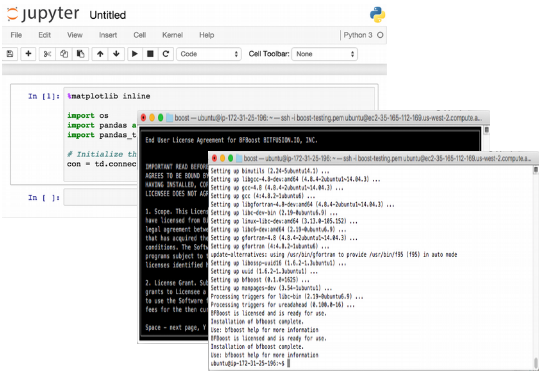
8 fans

GPU accelerators (4)



Front

GPU DEEP LEARNING RACK SOLUTION



bitfusion

- Pre-Built App Containers
- GPU and Workspace Management
- Elastic GPUs across the Datacenter
- Software defined Scaled out GPU Servers

R730

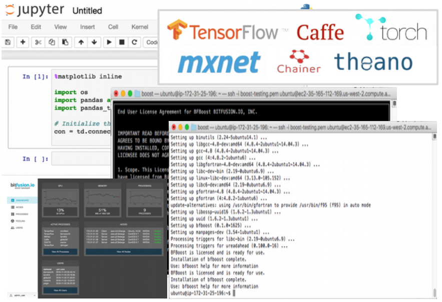
C4130

Configuration Details

Features	R730	C4130
CPU	E5-2669 v3@2.1GHz	E5-2630 v3@ 2.4Ghz
Memory	4GB	1TB/node; 64G DIMM
Storage	Intel PCIe NVME	Intel PCIe NVME
Networking IO	CX3 FDR InfiniBand	CX3 FDR InfiniBand
GPU	NA	M40-24GB
TOR Switch	Mellanox SX6036- FDR Switch	
Cables	FDR 56G DCA Cables	

DELLEMC

GPU DEEP LEARNING RACK SOLUTION

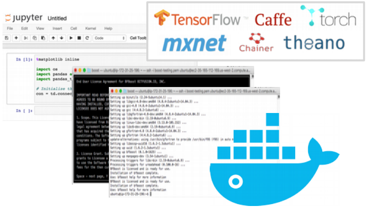


bitfusion

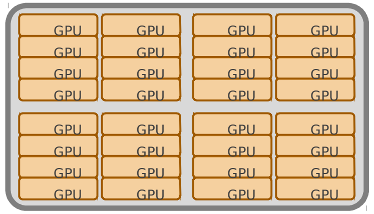
- Pre-Built App Containers
- GPU and Workspace Management
- Elastic GPUs across the Datacenter
- Software defined Scaled out GPU Servers

End to End Deep Learning Application Life Cycle

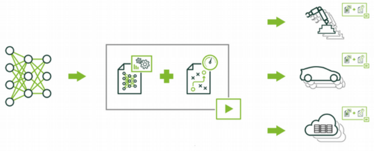
1 Develop

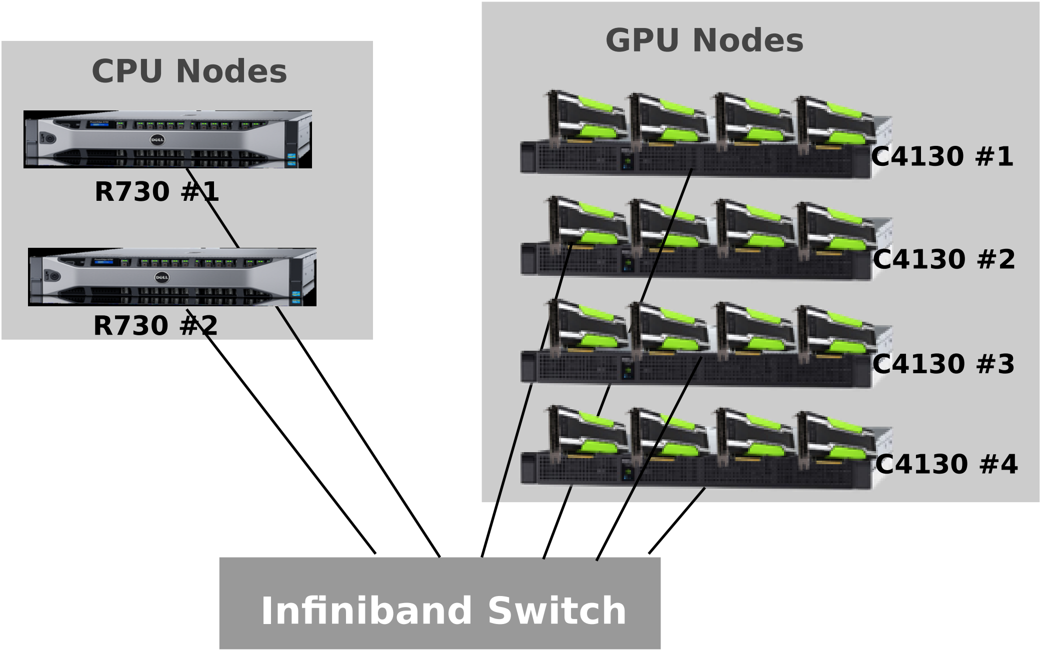


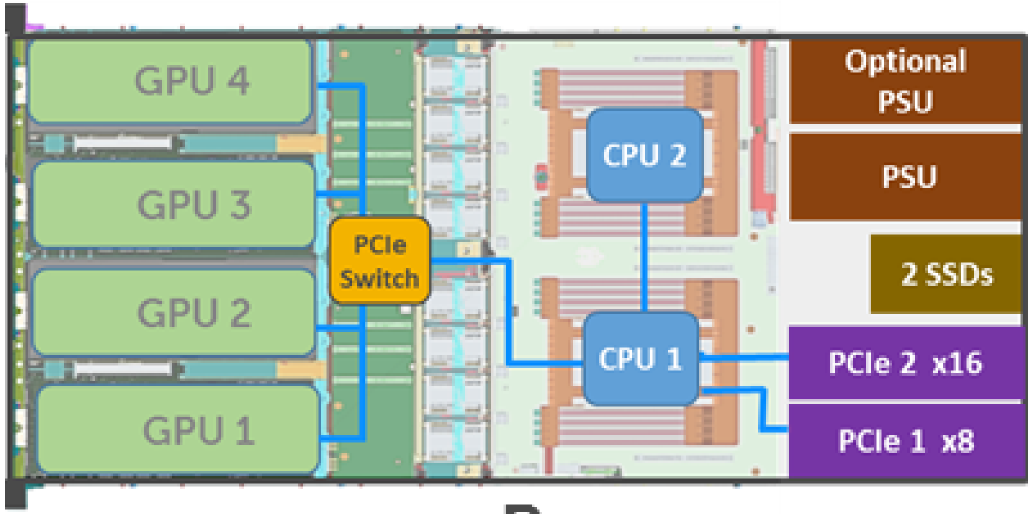
2 Train



3 Deploy







B

DELLEMC

...but wait, 'converged compute' requires network attached GPUs...

DELLEMC

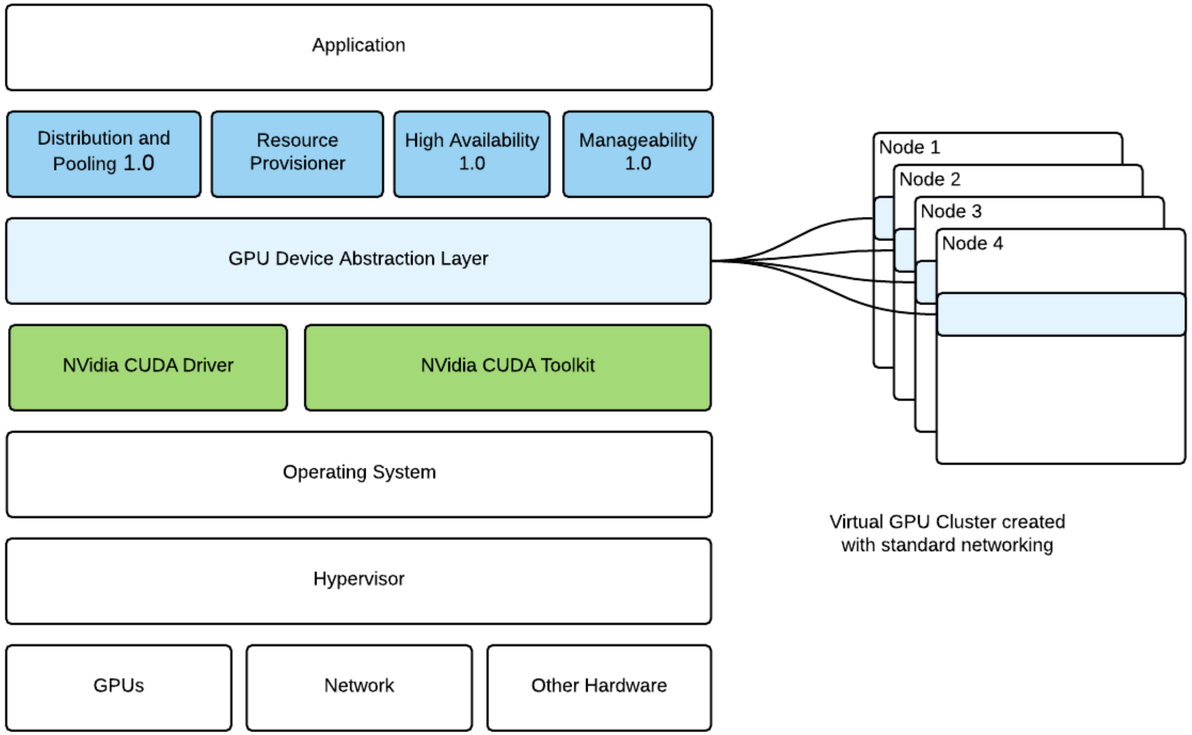
BITFUSION CORE VIRTUALIZATION

GPU Device Virtualization

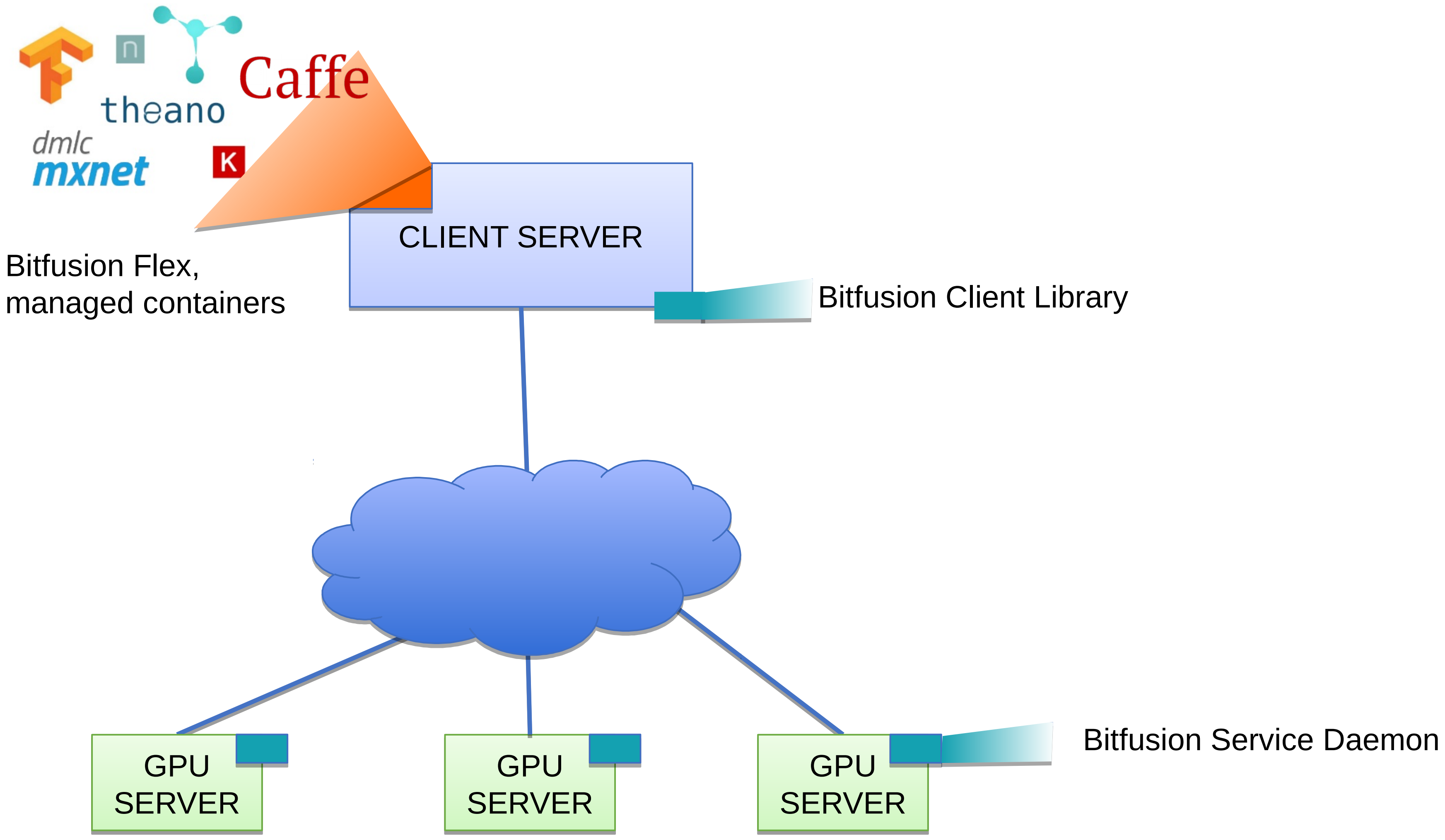
- Allows dynamic GPU attach on a per-application basis

Features

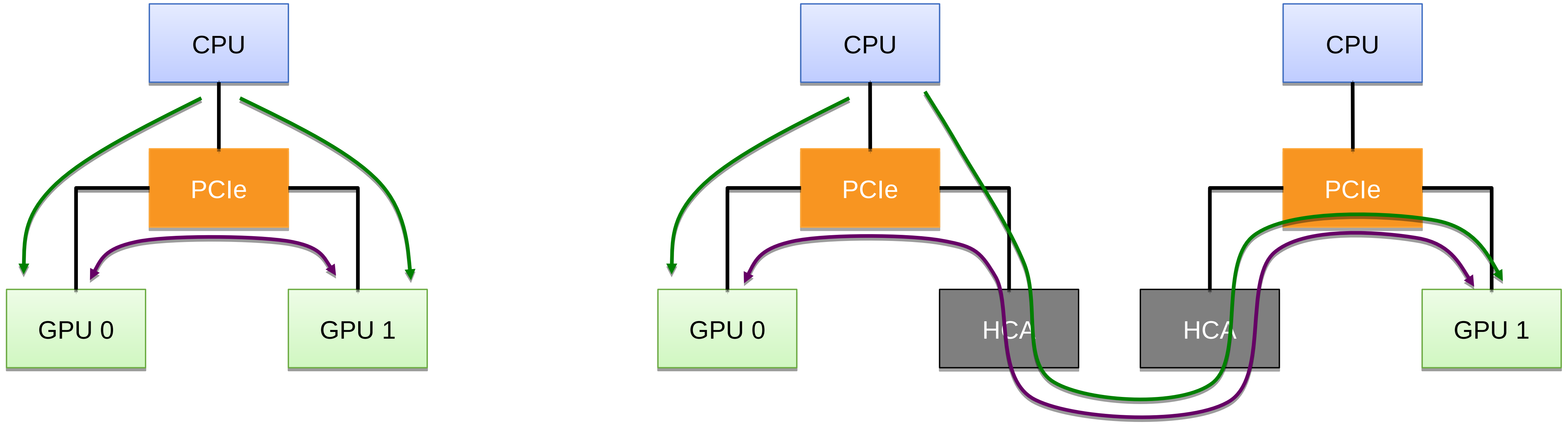
- APIs: CUDA, OpenCL
- Distribution: scale-out to remote GPUs
- Pooling: Oversubscribe GPUs
- Resource Provisioning: Fractional vGPUs
- High Availability: Automatic DMR
- Manageability: Remote nvidia-smi
- *Distributed* CUDA Unified Memory
- Native support for IB, GPUDirect RDMA
- Feature complete with CUDA 8.0



PUTTING IT ALL TOGETHER



NATIVE VS. REMOTE GPUs



Completely transparent: All CUDA Apps see local and remote GPUs as if directly connected

Results

REMOTE GPUs - LATENCY AND BANDWIDTH

- Data movement overheads is the primary scaling limiter

- Measurements done at application level – cudaMemcpy

Fast Local GPU copies

PCIe Intranode copies

Native GPUs

Bandwidth Matrix (GB/s)

	0	1	2	3
H	11.4	10.9	10.9	11.5
0	94.4	5.5	5.7	5.7
1	5.5	94.4	5.4	5.4
2	5.4	5.5	94.7	5.7
3	5.4	5.4	5.7	98.9

Latency Matrix (us)

src\dst	0	1	2	3
H	3	3	3	3
0	7	24	14	27
1	31	7	24	21
2	19	27	7	29
3	20	15	22	6

16 GPU virtual system: Naive implementation w/ TCP/IP

TCP/IP over IPoIB
Bandwidth Matrix (GB/s)

	src\dst	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	H	11.4	10.6	10.9	10.8	11.3	11.1	11.6	11.3	10.7	0.1	10.7	11.1	11.3	11.2	0.0	11.1
node 0	0	94.1	5.3	5.7	5.8	0.0	0.0	0.2	0.2	0.0	0.0	0.0	0.0	0.3	0.0	0.2	0.5
	1	5.4	95.0	5.3	5.4	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.2	0.0	0.2	0.0
	2	5.5	5.5	93.3	5.7	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.5	0.0
	3	5.3	5.3	5.5	99.2	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.1	0.0	0.3	0.2	0.3
node 1	4	0.0	0.2	0.0	0.3	####	5.7	5.5	5.7	0.4	0.0	0.4	0.2	0.2	0.0	0.3	0.3
	5	0.3	0.5	0.3	0.1	5.6	94.4	5.6	5.4	0.2	0.0	0.0	0.0	0.3	0.1	0.1	0.1
	6	0.3	0.3	0.3	0.2	5.5	5.6	97.7	5.7	0.2	0.2	0.2	0.0	0.1	0.1	0.0	0.3
	7	0.2	0.0	0.3	0.3	5.6	5.2	5.3	99.2	0.0	0.3	0.0	0.1	0.1	0.0	0.2	0.3
node 2	8	0.5	0.3	0.3	0.0	0.2	0.0	0.0	0.0	99.5	5.7	5.6	5.6	0.5	0.0	0.0	0.1
	9	0.1	0.3	0.3	0.0	0.4	0.4	0.4	0.0	5.6	99.8	5.3	5.4	0.3	0.0	0.0	0.3
	10	0.0	0.0	0.2	0.5	0.4	0.0	0.0	0.0	5.3	5.4	99.2	5.5	0.0	0.0	0.0	0.3
	11	0.3	0.2	0.0	0.0	0.0	0.0	0.2	0.2	5.4	5.7	5.7	99.5	0.0	0.5	0.3	0.3
node 3	12	0.0	0.0	0.3	0.3	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	####	5.6	5.8	5.7
	13	0.3	0.3	0.5	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.7	99.2	5.7	5.7
	14	0.4	0.0	0.3	0.3	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.2	5.5	5.6	####	5.7
	15	0.0	0.0	0.3	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	5.6	5.7	5.7	####

Latency Matrix (us)

	src\dst	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	H	11.9	8.93	6.56	9.34	12.9	12.9	5.86	7.81	12.2	9.54	8.99	5.6	7.84	9.31	5.76	10.4
	0	7.36	130	153	139	218	195	198	198	221	236	187	186	203	153	193	155
	1	150	7.17	137	141	196	184	188	195	201	197	188	217	164	179	172	152
	2	175	140	7.62	127	211	220	221	219	215	215	190	219	162	162	165	182
	3	106	134	138	7.42	208	209	215	213	203	204	212	191	160	177	166	157
	4	165	169	172	177	7.14	151	149	151	214	198	208	219	168	153	160	176
	5	168	162	162	153	149	7.2	128	152	198	207	231	221	177	167	166	170
	6	172	168	163	169	145	143	7.26	149	207	199	200	219	163	155	162	182
	7	156	167	178	178	150	156	142	8.29	192	190	237	212	173	166	171	160
	8	163	155	170	167	223	216	217	219	7.17	146	146	159	161	169	166	156
	9	164	166	193	179	199	228	225	207	152	6.24	115	152	166	160	157	158
	10	173	191	209	172	206	257	183	203	159	109	7.42	154	162	162	197	159
	11	164	175	171	164	214	222	220	211	151	142	145	7.14	161	172	167	164
	12	161	162	156	189	245	246	212	223	203	206	211	248	7.01	147	148	162
	13	186	165	157	154	206	195	191	213	237	216	249	196	140	6.24	139	139
	14	162	158	159	193	205	210	183	257	222	198	196	202	133	133	6.46	167
	15	195	188	197	201	238	244	205	218	219	191	201	210	134	139	136	7.04

Native GPUs
Bandwidth Matrix (GB/s)

	0	1	2	3
H	11.4	10.9	10.9	11.5
0	94.4	5.5	5.7	5.7
1	5.5	94.4	5.4	5.4
2	5.4	5.5	94.7	5.7
3	5.4	5.4	5.7	98.9

Fast local GPU copies

Intranode copies via PCIe

Latency Matrix (us)

src\dst	0	1	2	3
H	3	3	3	3
0	7	24	14	27
1	31	7	24	21
2	19	27	7	29
3	20	15	22	6

Low BW, High Latency remote copies

OS Bypass needed to avoid primary TCP/IP overheads
AI apps are very **latency** sensitive

16 GPU virtual system: Bitfusion optimized transport and runtime

- Same FDRx4 transport, but drop IPoIB
- Replace remote calls with native IB
- Runtime selection of intranode RDMA or cudaMemcpy
- Multi-rail communications where available
- Runtime optimizations: pipelining, execution, distributed caching & ev...
- …

IB+RDMA attached GPUs

Bandwidth Matrix (GB/s)

src\dst	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
H	11.6	11.1	10.8	10.6	9.5	11.1	10.5	9.0	10.8	11.3	10.7	11.3	10.9	11.2	10.9	10.8
0	94.4	5.7	5.6	5.5	3.9	4.0	3.9	3.9	3.9	3.9	3.9	3.8	3.8	3.7	3.8	3.8
1	5.5	99.5	5.7	5.7	3.9	3.9	3.9	3.9	4.0	3.9	3.9	3.8	3.7	3.7	3.8	3.7
2	5.3	5.3	99.2	5.6	3.9	3.8	3.9	4.0	3.9	3.9	3.9	3.7	3.8	3.7	3.8	3.8
3	5.4	5.3	5.4	98.9	3.8	4.1	3.9	3.9	3.9	4.0	4.0	3.9	3.9	4.0	4.0	3.8
4	3.8	3.7	3.7	3.7	91.4	4.8	4.8	5.6	3.9	3.9	3.9	3.9	3.9	3.8	[illegible]	3.8
5	3.8	3.7	3.8	3.7	4.8	91.1	4.8	4.7	3.8	3.9	3.8	4.0	3.8	3.9	3.8	3.7
6	3.8	3.7	3.7	3.8	4.7	5.0	####	5.2	3.9	3.9	3.9	3.9	3.8	3.9	3.9	3.7
7	3.8	3.8	3.9	3.8	4.8	4.8	5.0	94.7	3.8	4.0	4.0	4.0	3.9	3.9	3.9	3.9
8	3.8	3.8	3.8	3.8	3.8	3.8	3.8	3.8	89.8	5.6	5.6	5.6	4.0	3.9	3.9	3.9
9	3.8	3.9	3.8	3.8	3.7	3.8	3.8	3.8	5.4	93.6	5.6	5.5	4.0	3.9	3.9	3.9
10	3.8	3.8	3.8	3.7	3.7	3.8	3.9	3.8	5.5	5.5	99.5	5.7	3.9	3.8	[illegible]	3.8
11	3.7	3.8	3.9	3.8	3.9	3.9	4.0	3.8	5.2	5.7	5.7	####	4.0	3.9	4.0	4.0
12	3.7	3.8	3.9	3.8	3.9	3.8	3.8	3.8	3.8	3.8	3.7	3.8	94.4	5.3	5.8	5.7
13	3.8	3.8	3.8	[illegible]	3.7	3.7	3.7	3.7	3.7	3.8	4.0	3.8	5.4	92.5	5.6	5.8
14	3.7	3.8	3.9	3.8	3.7	3.9	3.7	3.9	3.8	3.7	3.9	4.0	5.6	5.6	####	5.7
15	3.9	3.8	3.9	3.9	4.0	4.0	3.8	4.0	3.9	3.9	4.0	3.9	5.6	5.5	5.8	####

Latency Matrix (us)

src\dst	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
H	8	5	7	9	8	6	12	6	13	5	12	6	6	9	5	6
0	7	13	14	14	14	13	11	14	13	13	14	14	14	13	12	14
1	14	7	12	15	13	11	11	12	12	12	12	12	14	14	14	14
2	13	11	7	14	12	12	13	12	12	13	13	14	12	12	13	14
3	14	13	13	7	14	15	13	14	14	12	13	13	14	14	13	[illegible]
4	13	13	12	13	8	12	12	14	12	14	13	13	12	12	13	13
5	14	13	14	14	14	8	12	13	13	15	13	12	11	11	12	13
6	14	14	13	16	13	15	7	12	11	14	13	13	12	13	12	12
7	13	14	12	13	12	13	12	7	13	14	11	14	14	11	13	13
8	14	14	11	12	14	15	11	14	7	13	14	13	15	12	[illegible]	[illegible]
9	12	14	14	13	13	12	15	12	23	7	14	13	15	12	12	13
10	14	14	11	14	13	14	12	12	14	12	8	15	14	13	12	14
11	13	12	13	13	12	12	14	23	12	13	14	8	14	22	12	13
12	13	13	13	12	13	12	13	13	13	14	12	13	6	14	12	13
13	12	13	12	13	12	14	12	13	12	12	13	14	15	7	12	14
14	13	14	12	14	13	11	12	14	12	11	12	13	16	14	7	14
15	12	12	14	13	13	13	11	13	12	15	12	14	15	14	15	7

TCP/IP over IPoIB

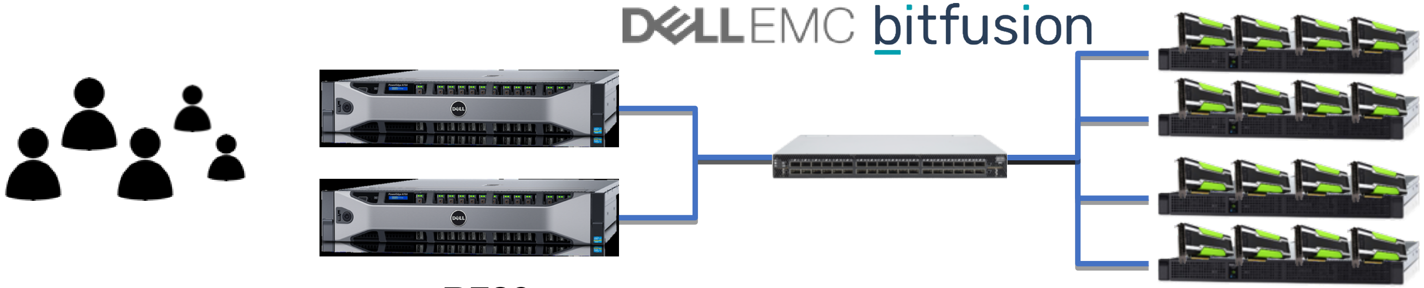
Bandwidth Matrix (GB/s)

src\dst	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
H	11.4	10.6	10.9	10.8	11.3	11.1	11.6	11.3	10.7	0.1	10.7	11.1	11.3	11.2	0.0	11.1
0	94.1	5.3	5.7	5.8	0.0	0.0	0.2	0.2	0.0	0.0	0.0	0.0	0.3	0.0	0.2	0.5
1	5.4	95.0	5.3	5.4	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.2	0.0	0.2	0.0
2	5.5	5.5	93.3	5.7	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.5	0.0
3	5.3	5.3	5.5	99.2	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.1	0.0	0.3	0.2	0.3
4	0.0	0.2	0.0	0.3	####	5.7	5.5	5.7	0.4	0.0	0.4	0.2	0.2	0.0	0.3	0.3
5	0.3	0.5	0.3	0.1	5.6	94.4	5.6	5.4	0.2	0.0	0.0	0.0	0.3	0.1	0.1	0.1
6	0.3	0.3	0.3	0.2	5.5	5.6	97.7	5.7	0.2	0.2	0.2	0.0	0.1	0.1	0.0	0.3
7	0.2	0.0	0.3	0.3	5.6	5.2	5.3	99.2	0.0	0.3	0.0	0.1	0.1	0.0	0.2	0.3
8	0.5	0.3	0.3	0.0	0.2	0.0	0.0	0.0	99.5	5.7	5.6	5.6	0.5	0.0	0.0	0.1
9	0.1	0.3	0.3	0.0	0.4	0.4	0.4	0.0	5.6	99.8	5.3	5.4	0.3	0.0	0.0	0.3
10	0.0	0.0	0.2	0.5	0.4	0.0	0.0	0.0	5.3	5.4	99.2	5.5	0.0	0.0	0.0	0.3
11	0.3	0.2	0.0	0.0	0.0	0.0	0.2	0.2	5.4	5.7	5.7	99.5	0.0	0.5	0.3	0.3
12	0.0	0.0	0.3	0.3	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	####	5.6	5.8	5.7
13	0.3	0.3	0.5	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.7	99.2	5.7	5.7
14	0.4	0.0	0.3	0.3	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.2	5.5	5.6	####	5.7
15	0.0	0.0	0.3	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	5.6	5.7	5.7	####

Latency Matrix (us)

src\dst	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
H	11.9	8.93	6.56	9.34	12.9	12.9	5.86	7.81	12.2	9.54	8.99	5.6	7.84	9.31	5.76	10.4
0	7.36	130	153	139	218	195	198	198	221	236	187	186	203	153	193	155
1	150	7.17	137	141	196	184	188	195	201	197	188	217	164	179	172	152
2	175	146	7.62	127	211	220	221	219	215	215	190	219	162	162	165	182
3	106	134	138	7.42	208	209	215	213	203	204	212	191	160	177	166	157
4	165	169	172	177	7.14	151	149	151	214	198	208	219	168	153	160	176
5	168	162	162	153	149	7.2	128	152	198	207	231	221	177	167	166	170
6	172	168	163	169	145	143	7.26	149	207	199	200	219	163	155	162	182
7	156	167	178	178	150	156	142	8.29	192	190	237	212	173	166	171	160
8	163	155	170	167	223	216	217	219	7.17	146	146	159	161	169	166	156
9	164	166	193	179	199	228	225	207	152	6.24	115	152	166	160	157	158
10	173	191	209	172	206	257	183	203	159	109	7.42	154	162	162	197	159
11	164	175	171	164	214	222	220	211	151	142	145	7.14	161	172	167	164
12	161	162	156	189	245	246	212	223	203	206	211	248	7.01	147	148	162
13	186	165	157	154	206	195	191	213	237	216	249	196	140	6.24	139	139
14	162	158	159	193	205	210	183	257	222	198	196	202	133	133	6.46	167
15	195	188	197	201	238	244	205	218	219	191	201	210	134	139	136	7.04

SLICE & DICE - MORE THAN ONE WAY TO GET 4 GPUs

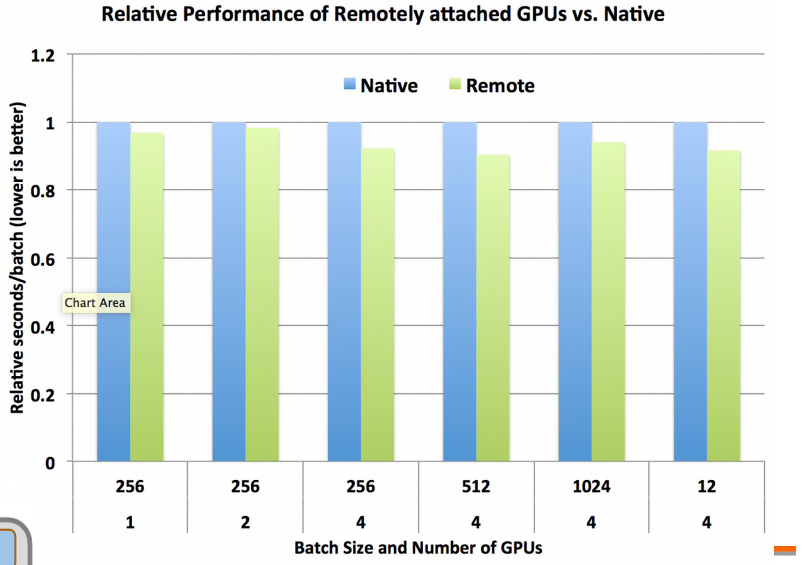


R730

C4130

***Native* GPU performance with network attached GPUs**

Run time comparison (lower is better) →

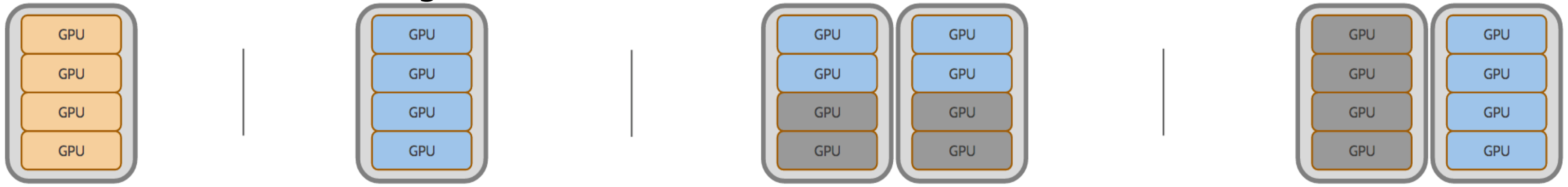


Caffe GoogleNet

TensorFlow Pixel-CNN

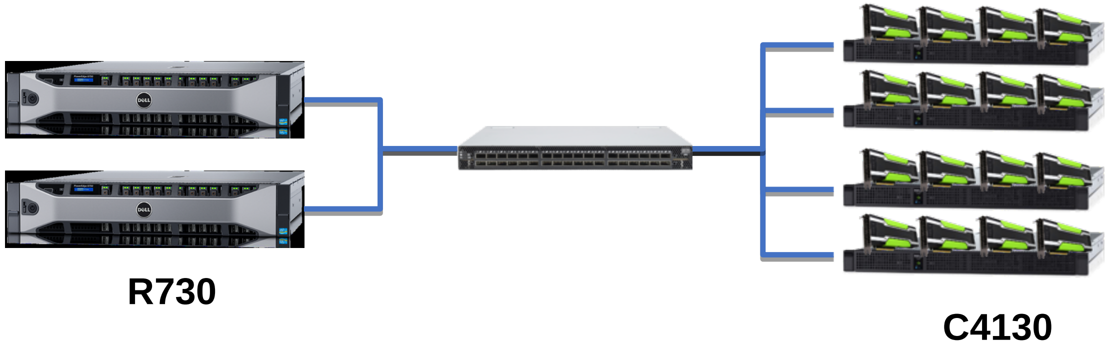
Multiple ways to create a virtual 4 GPU node, *with native efficiency*

(secs to train Caffe GoogleNet, batch size: 128)

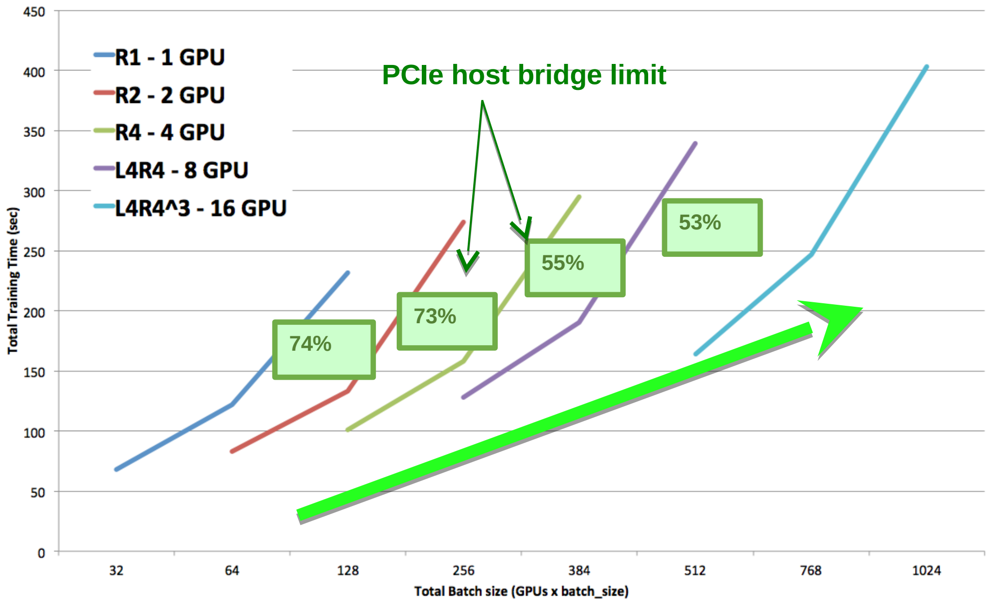


N4	L4	L2R2	R4
Native 4 GPUs (w/o Boost)	*4 Local GPUs*	*2 Local, 2 Remote GPUs*	*4 Remote GPUs*
323	**301**	**295**	**293**

TRAINING PERFORMANCE

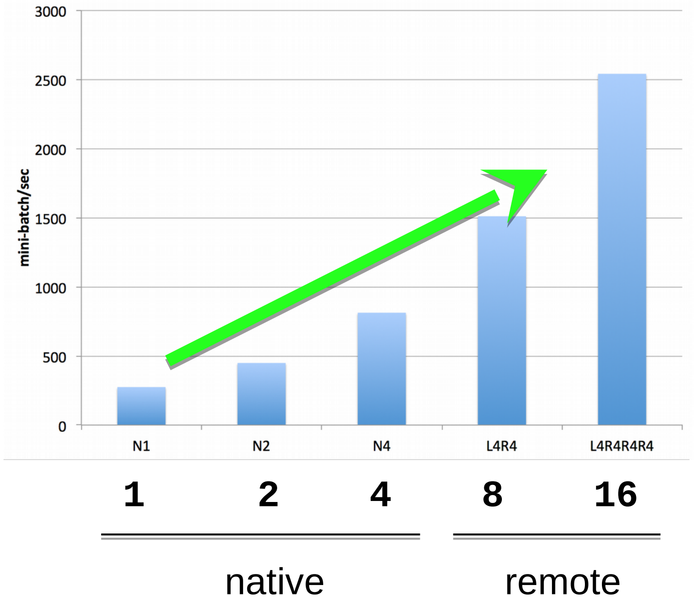


Continued Strong Scaling
Caffe GoogleNet

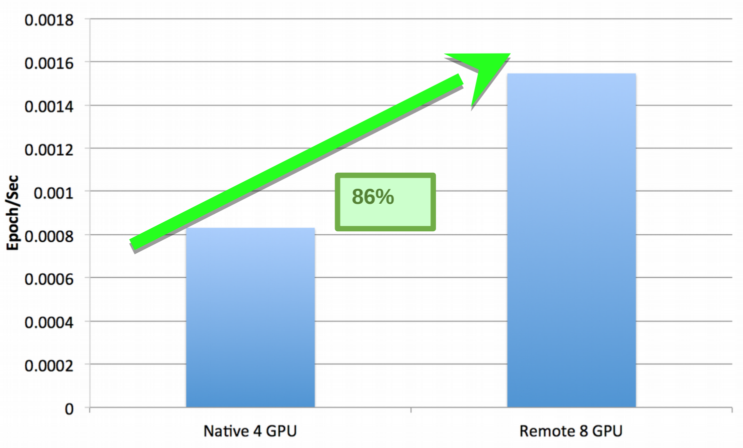


Weak-scaling
Accelerate Hyper parameter Optimization

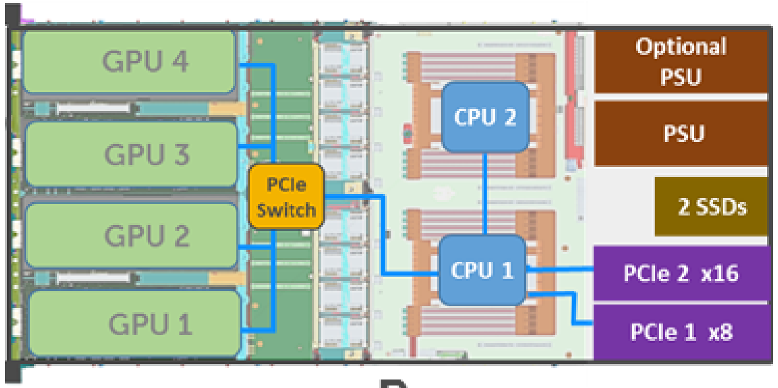
Caffe GoogleNet



TensorFlow 1.0 with Pixel-CNN

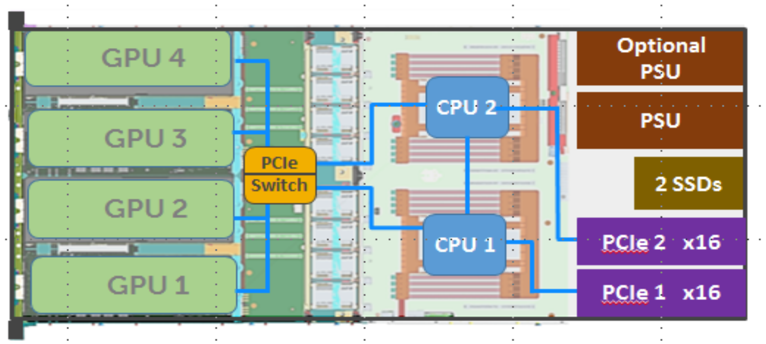


Other PCIe GPU Configurations Available



B

Currently Testing



Config 'G'

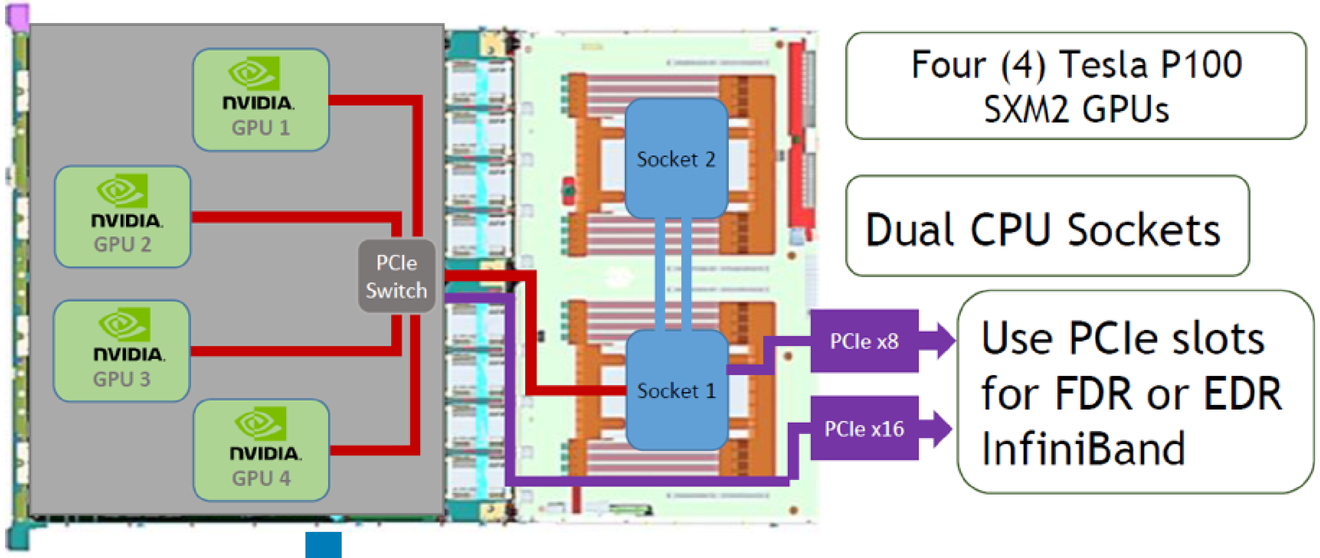
Further reading:

http://en.community.dell.com/techcenter/high-performance-computing/b/general_hpc/archive/2016/11/11/deep-learning-performance-with-p100-gpus

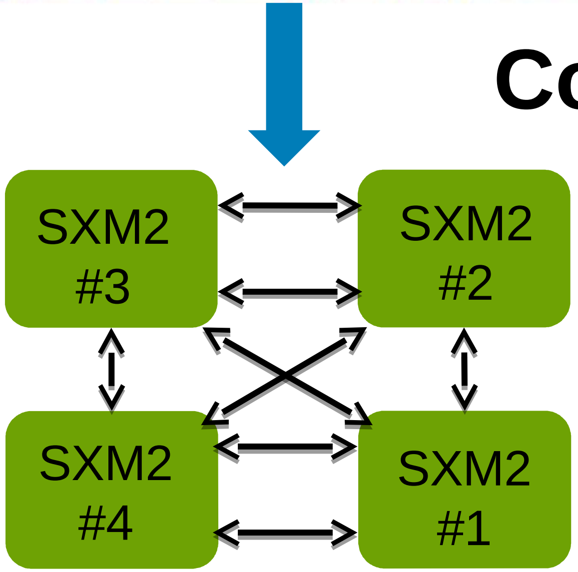
http://en.community.dell.com/techcenter/high-performance-computing/b/general_hpc/archive/2017/03/22/deep-learning-inference-on-p40-gpus

DELL EMC

NvLink Configuration

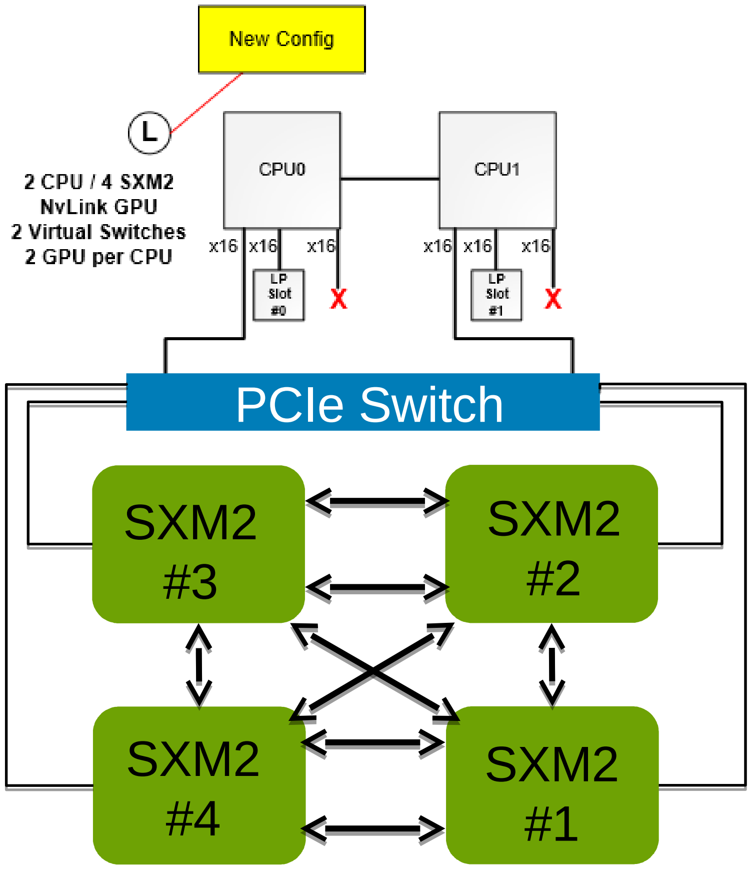


Config 'K'



- 4 P100-16GB SXM2 GPU
- 2 CPU
- PCIe switch
- 1 PCIe slot – EDR IB

NvLink Configuration



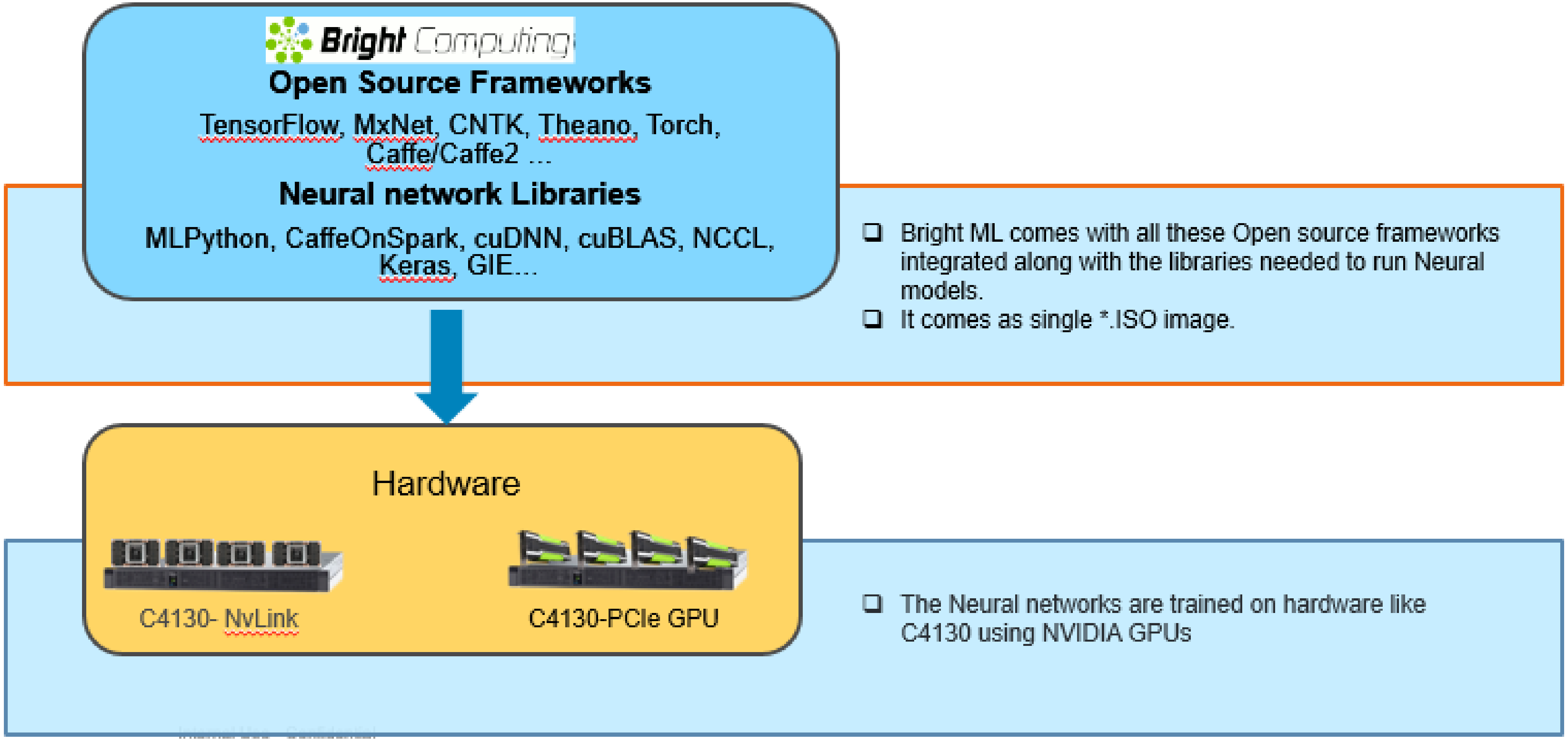
Config 'L'

- **4 P100-16GB SXM2 GPU**
- **2 CPU**
- **PCIe switch**
- **1 PCIe slot – EDR IB**
- **Memory : 256GB w/16GB @ 2133**
- **OS: Ubuntu 16.04**
- **CUDA: 8.1**

Software Solutions

Overview – Bright ML

- Dell EMC has partnered with Bright Computing to offer their Bright ML package as the software stack on Dell EMC Deep learning hardware solution.



Internal Use - Confidential

D&LLEMC

Bright ML Overview

Bright 8.0 Features

- Bright View administrator web interface
- Cloud bursting support for Azure
- New monitoring subsystem
- Ubuntu 16.04 LTS support
- OpenStack Newton
- Mesos integration (+ Marathon)
- Improved Kubernetes integration
- Updated and new machine learning packages
- NVIDIA DCGM integration
- CephFS support
- Job based metrics enabled by default

FRAMEWORKS

Caffe / (Caffe2)

TensorFlow

Theano

Torch

(CNTK)

(MXNet)

(Caffe-MPI)

LIBRARIES

MLPython

cuDNN

DIGITS

CaffeOnSpark

NCCL

(GIE)

(Keras)

DELL EMC

Machine Learning in Seismic Imaging Using KNL + FPGA – Project # 1

Bhavesh Patel – Server Advanced Engineering

Robert Dildy - Product Technologist Sr. Consultant, Engineering Solutions

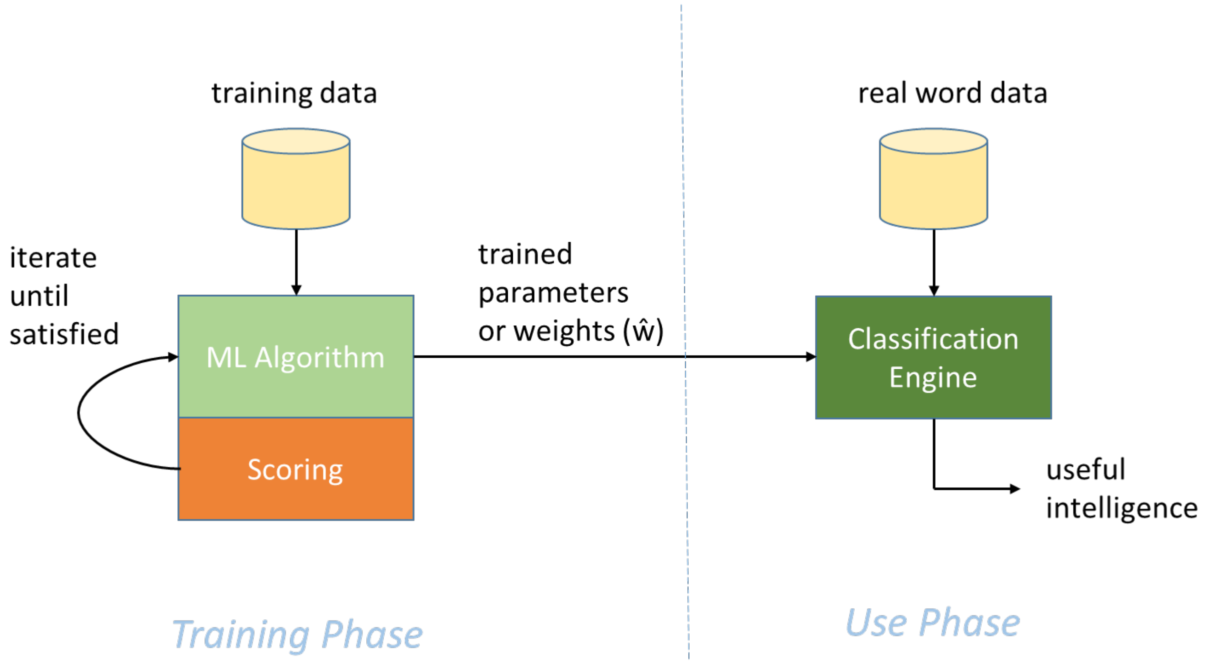
Abstract

This paper is focused on how to apply Machine Learning to seismic imaging with the use of FPGA as a co-accelerator.

It will cover 2 hardware technologies: 1) Intel KNL Phi 2) FPGA and also address how to use Machine learning for seismic imaging.

There are different types of accelerators like GPU, Intel Phi but we are choosing to study how we can use i-ABRA platform on KNL + FPGA to train the neural network using Seismic Imaging data and then doing the inference.

Machine learning in a broader sense can be divided into 2 parts namely : Training and Inference.



DELL EMC

Background

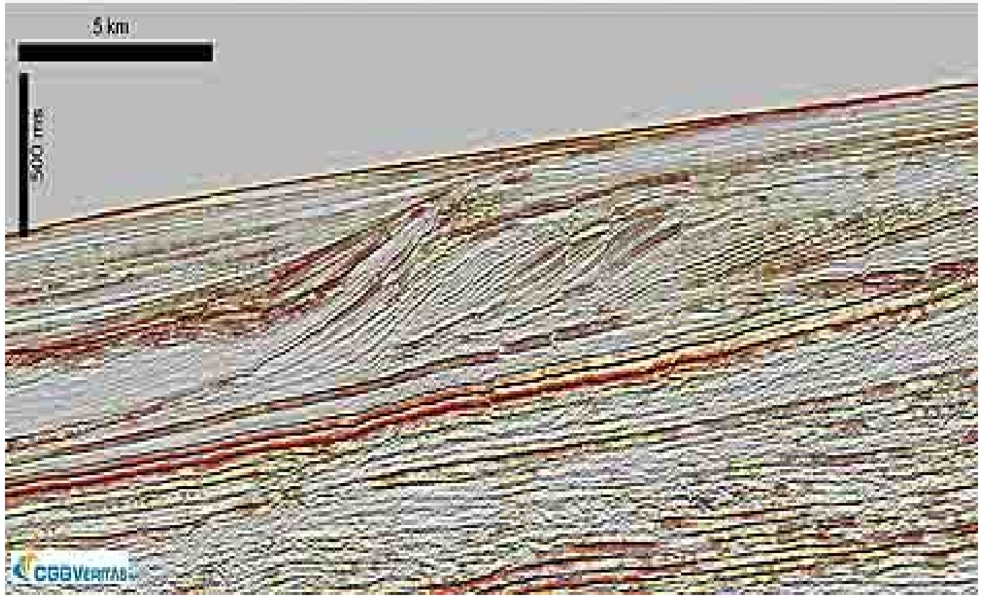
Seismic Imaging is a standard data processing technique used in creating an image of subsurface structures of the Earth from measurements recorded at the surface via seismic wave propagations captured from various sound energy sources.

There are certain challenges with Seismic data interpretation like 3D is starting to replace 2D for seismic interpretation.

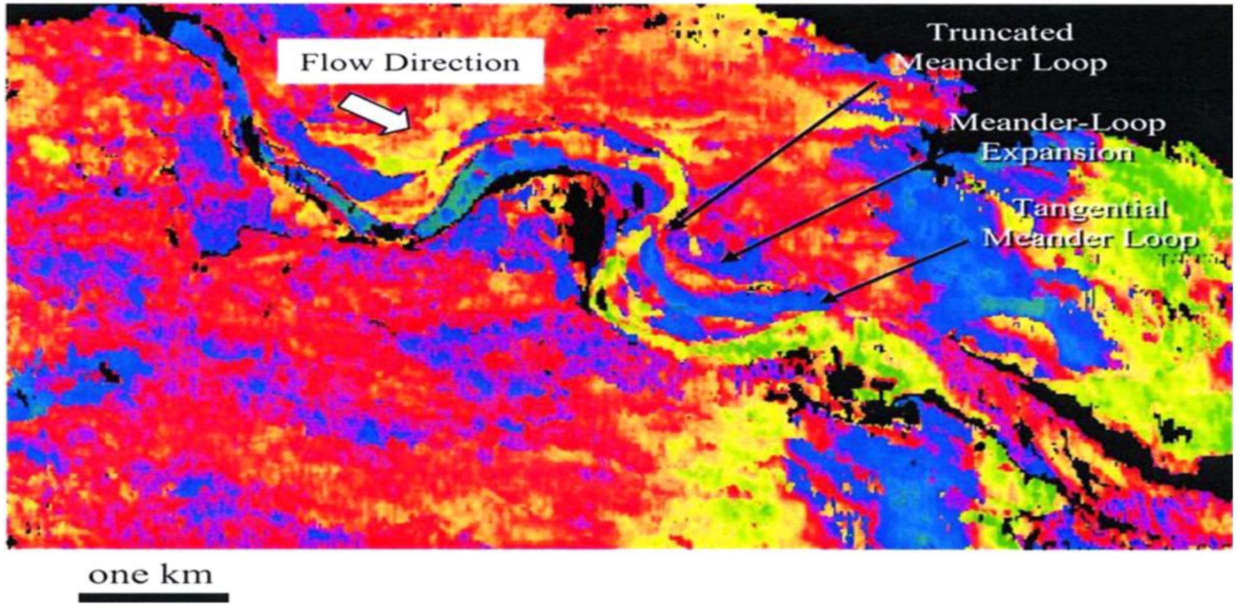
There has been rapid growth in use of computer vision technology & several companies developing image recognition platforms. This technology is being used for automatic photo tagging and classification. The same concept could be applied to identify geometric patterns in the data and generate image captions/descriptions. We can use Convolutional Neural Networks (CNN) to learn visual concepts using massive amounts of data which would help in doing objective analysis of it.

The use of machine learning and image processing algorithms to analyze, recognize and understand visual content would allow us to analyze data both in Supervised neural networks(SNN) and unsupervised neural networks (UNN) like CNN.

Observing both plane and cross-section



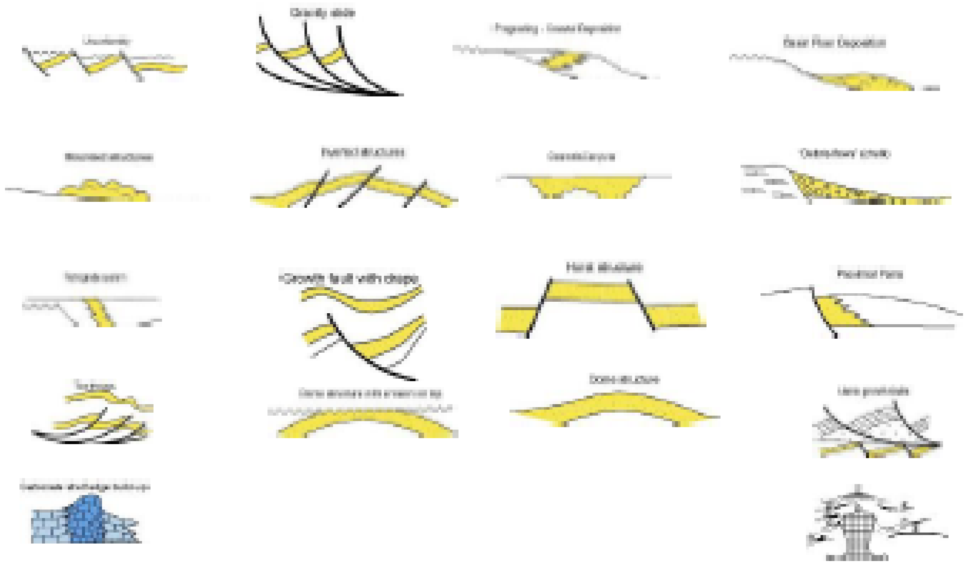
Seismic Stratigraphic image learning



Seismic Geomorphology image learning

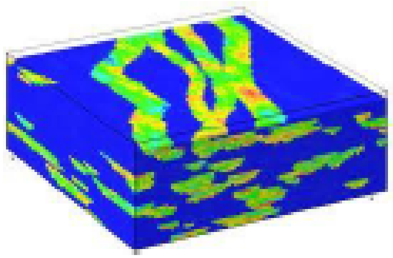
Models in plane and cross-section

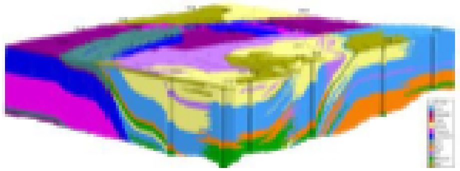
Seismic Stratigraphic image models



Train the data to recognize geometrical patterns and utilization of “iPhoto” and “Facebook” technology and methodology to interact with the training.

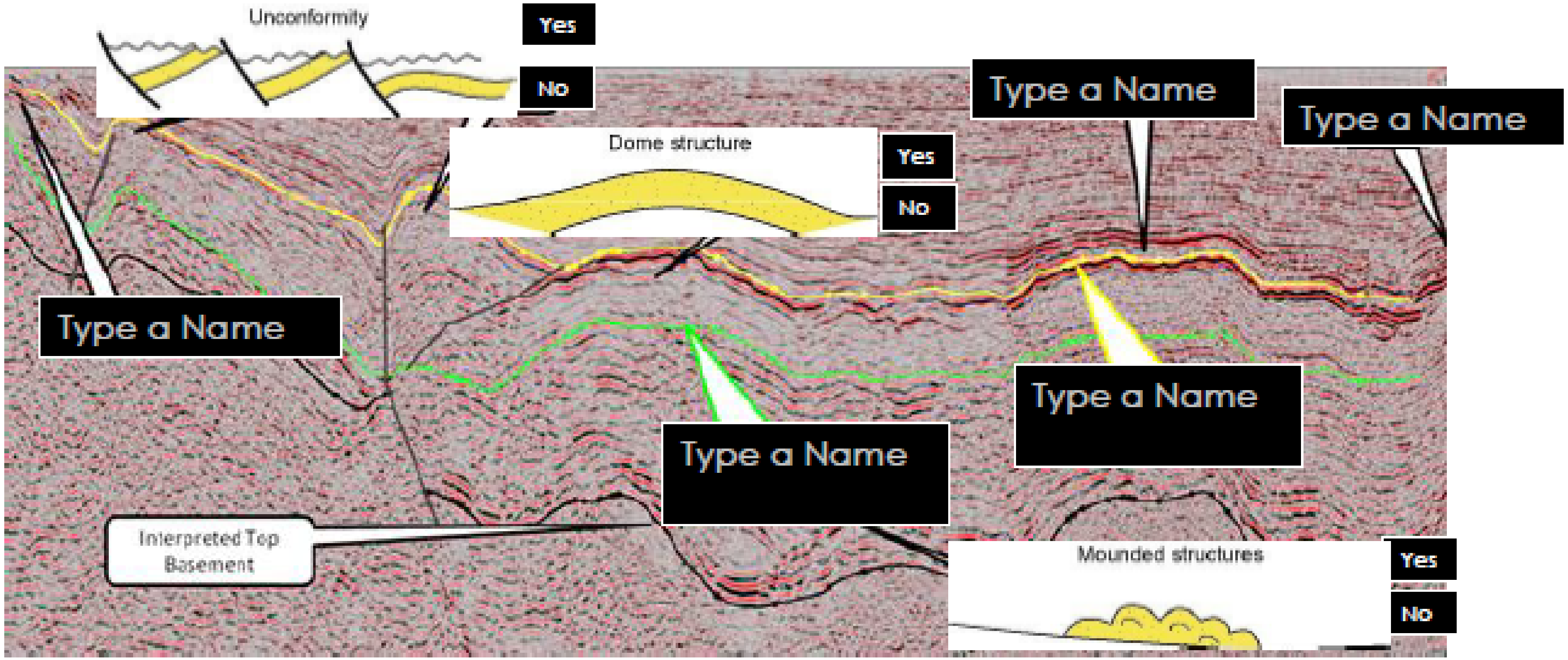
Seismic Geomorphology image models





Algorithms already established in geological modeling software.
Require some guidance with a low frequency surface model in data to mimic dips and curvatures in stratigraphic response of data

Tags with 'facies' recognition



You give input to the unsupervised training of your data. It will automatically identify similar ones and/or give you a choice of places it finds similar, and you choose to tell its right or wrong.

Solution

For this paper we will be using the following Hardware and Software platforms:

Hardware Platform:

- C6320P Sleds with Intel KNL Phi + Intel Arria 10 (A10PL4) FPGA adapter.

Software Platform:

- i-ABRA Deep learning framework

This will be a joint collaboration with :

- Dell EMC
- Intel
- i-ABRA
- Seismic Imaging firm - TBD

Thank You