MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

HiDL
High-Performance Deep Learning

HiBD
High-Performance Big Data

High-Performance Deep Learning with Large Pathology WSI Images

Talk at First Midwestern Consortium Workshop for Computational Pathology (Jan ’21)

by

Hari Subramoni

The Ohio State University

E-mail: subramon@cse.ohio-state.edu

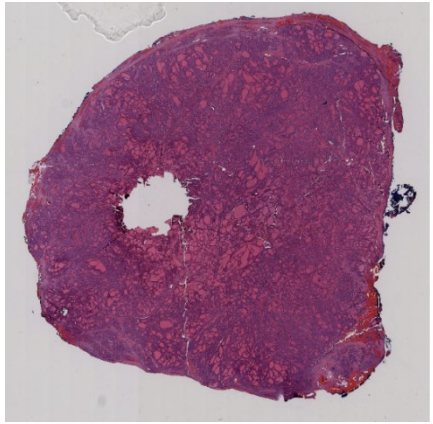
http://www.cse.ohio-state.edu/~subramon

Follow us on

https://twitter.com/mvapich

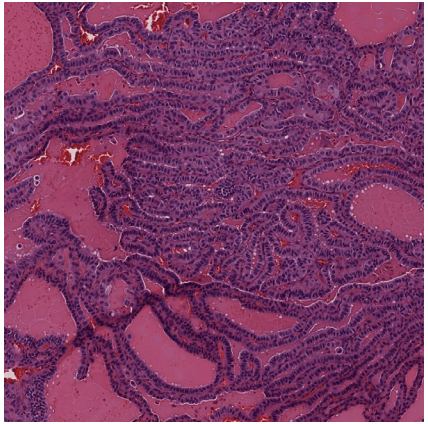
Digital Pathology

- Whole Slide Images (WSI)
  - Replacing the glass slide for diagnostic purposes
  - Typically, 100,000 X 100,000 pixels in size



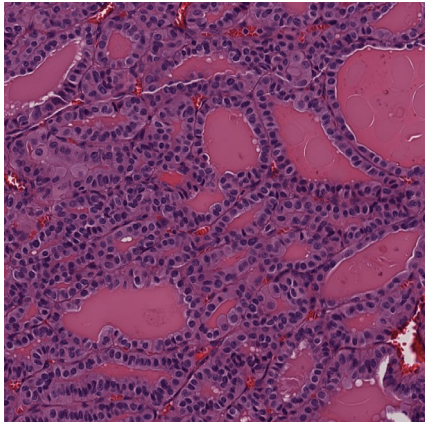
A whole slide image (WSI)

A Hematoxylin and Eosin stained whole slide image labeled as Tall Cell Variant (TCV) of the papillary thyroid cancer (PTC).



A tile at 10x magnification level

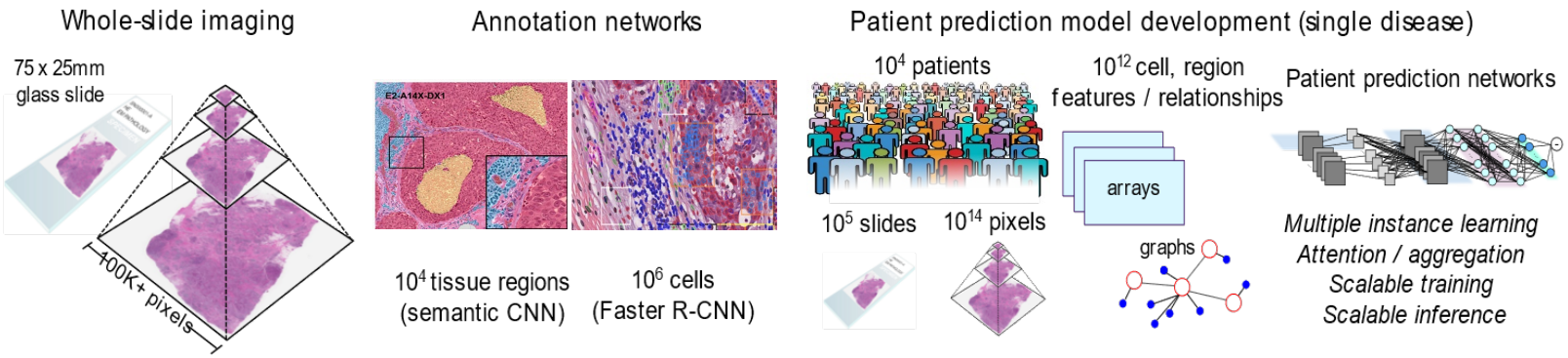
A 1024x1024 image tile at 10 magnification level shows histologic feature of elongated follicles arranged in parallel cords or tram tracks.



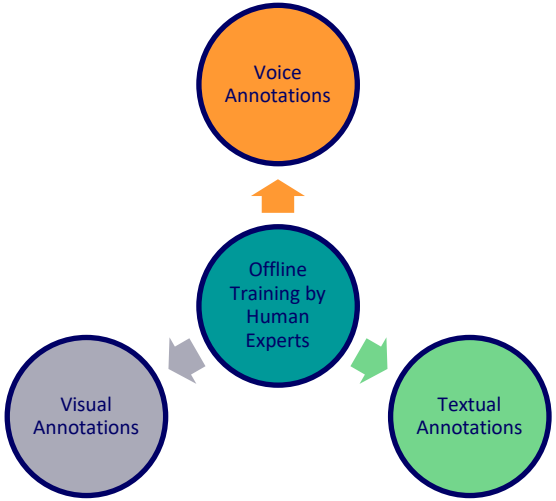
A tile at 20x magnification level

A 1024x1024 image tile at 20 magnification level shows cellular features of tall cells

Modern Computational Pathology Workflow



- Begins with digitization of glass slides to form large multi resolution, whole-slide images
  - Each slide can contain more than 10^6 cells and 10^4 tissue regions.
- Cells and tissue regions in these images can be automatically annotated using neural networks like Faster R-CNN that are trained offline using **textual, audio, and visual** annotations generated by human experts
- Developing models for diagnosis or predicting clinical outcomes for a single disease may involve **10^4 or more patients**
- Typical studies will encompass **hundreds of thousands of slides**, generating **billions of annotated cells and tissue regions**
- **Variety (audio, textual, and visual) and volume of data creates unique learning and computing challenges**

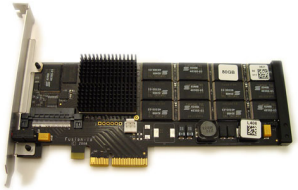


Drivers of Modern HPC Cluster Architectures

Multi-core Processors

High Performance Interconnects - InfiniBand
<1usec latency, 200Gbps Bandwidth>

Accelerators / Coprocessors high compute density, high performance/watt
>1 TFlop DP on a chip



SSD, NVMe-SSD, NVRAM

- Multi-core/many-core technologies
- Remote Direct Memory Access (RDMA)-enabled networking (InfiniBand and RoCE)
- Solid State Drives (SSDs), Non-Volatile Random-Access Memory (NVRAM), NVMe-SSD
- Accelerators (NVIDIA GPGPUs and Intel Xeon Phi)
- Available on HPC Clouds, e.g., Amazon EC2, NSF Chameleon, Microsoft Azure, etc.

Summit

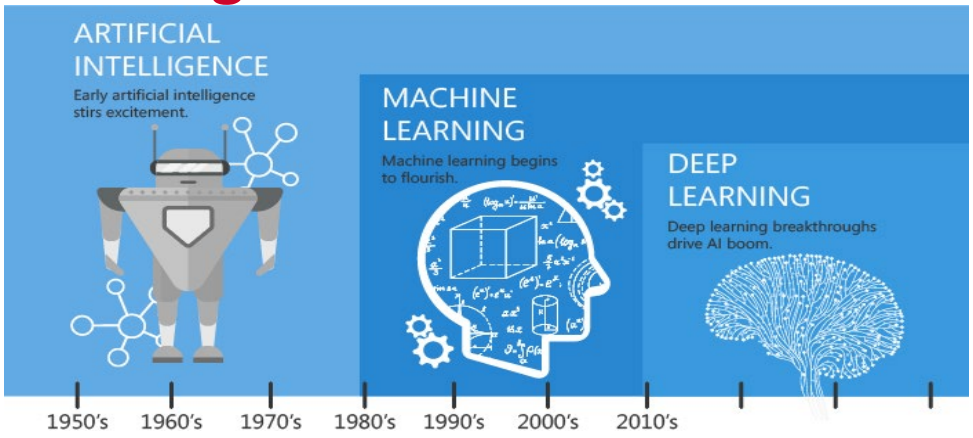
Sierra

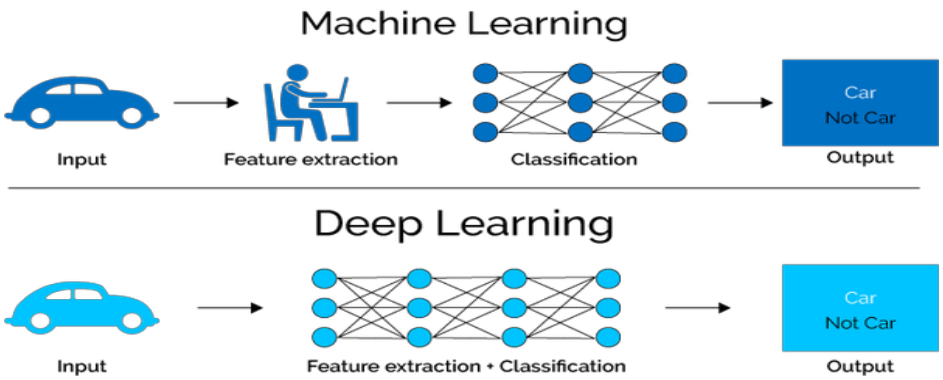
Sunway TaihuLight

K - Computer

AI, Machine Learning & Deep Learning

- Machine Learning (ML) with many traditional applications
  - K-means
  - Random Forest
  - Linear Regression
  - Nearest Neighbor
- Deep Learning (DL)
  - A subset of Machine Learning that uses Deep Neural Networks (DNNs)
- Based on learning data representation
- Examples Convolutional Neural Networks, Recurrent Neural Networks, Hybrid Networks





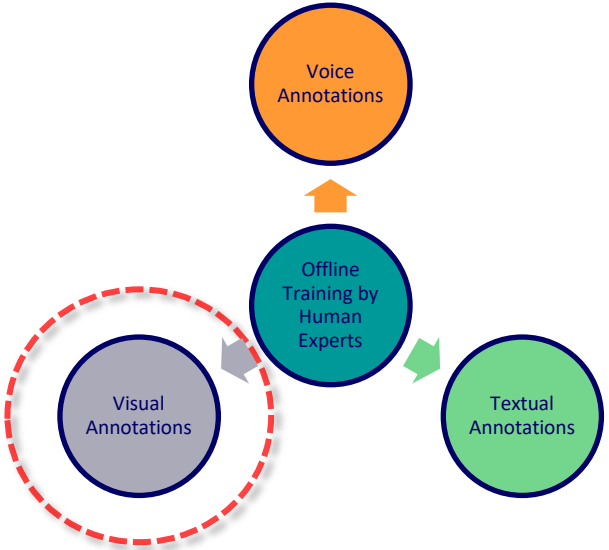
Courtesy: https://hackernoon.com/difference-between-artificial-intelligence-machine-learning-and-deep-learning-1pcv3zeg, https://blog.dataiku.com/ai-vs.-machine-learning-vs.-deep-learning

Key Phases of Deep Learning

- Deep Learning has two major tasks
  1. Training of the Deep Neural Network
  2. Inference (or deployment) that uses a trained DNN
- DNN Training
  - Training is a compute/communication intensive process – can take days to weeks
  - Faster training is necessary!
- Faster training can be achieved by
  - Using Newer and Faster Hardware – But there is a limit!
  - Can we use more GPUs or nodes?
    - The need for Parallel and Distributed Training

Broad Challenge

- Training high-level TCV classifier using data parallelism on 1024 X 1024 image tiles takes 7.25 hours on a state-of-the-art multi-GPU compute node
  - An example of training DL models based on visual annotation



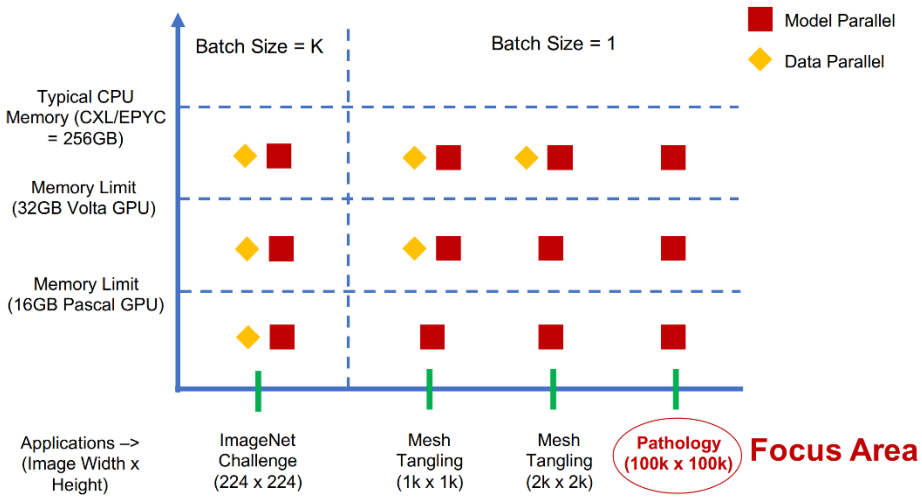
- Can we accelerate training of the TCV classifier?

How to make training faster?

- Data parallelism
  - Horovod: TensorFlow, PyTorch, and MXNet
  - TensorFlow: tf.distribute.Strategy API
  - PyTorch: torch.nn.parallel.DistributedDataParallel
- Model-parallelism and Hybrid-parallelism
  - LBANN: Only framework designed for distributed training
  - Higher-level frameworks: Gpipe, Mesh-TensorFlow, DeepSpeed, etc.
  - Model-level Support: Megatron-LM, OpenAI, etc.

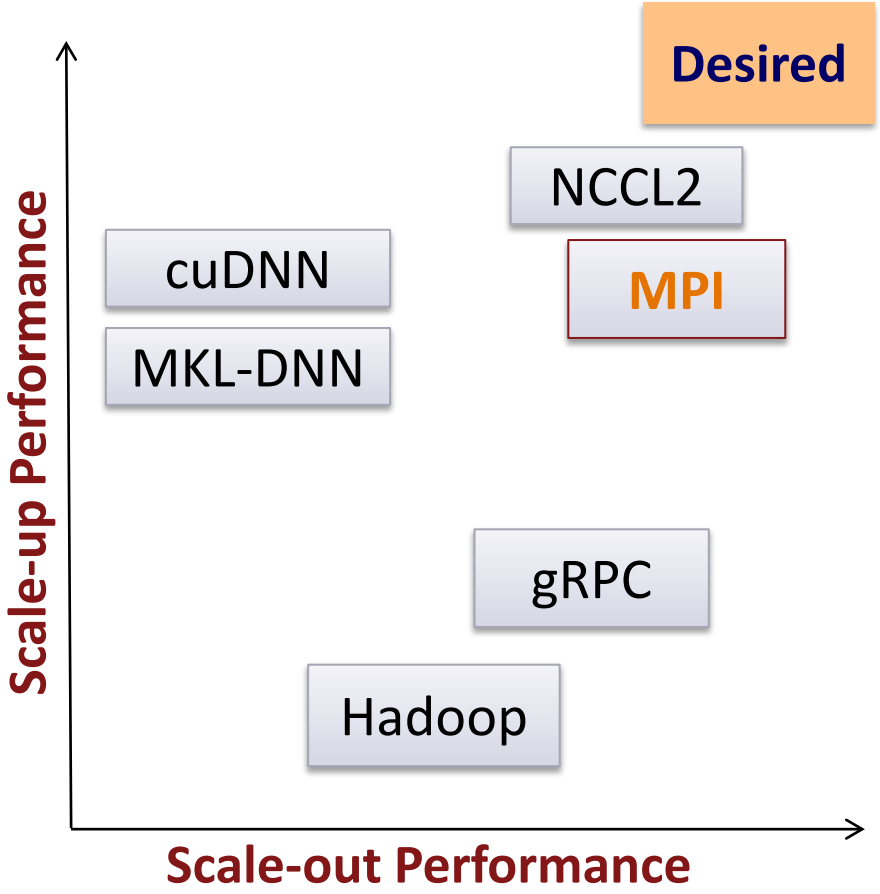
Why Model Parallelism?

- Data-Parallelism– only for models that fit the memory
- Out-of-core models
  - Deeper model → Better accuracy but more memory required!
- Model parallelism can work for out-of-core models!
- Performance is questionable!

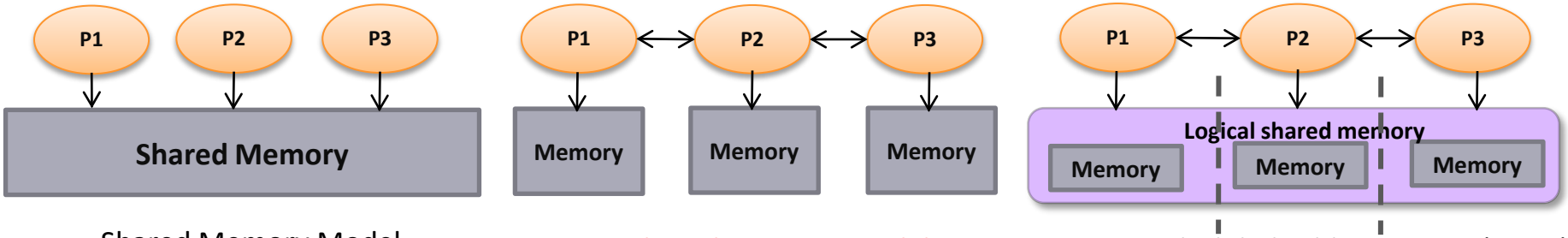


Scale-up and Scale-out

- **Scale-up**: Intra-node Communication
  - Many improvements like:
    - NVIDIA cuDNN, cuBLAS, NCCL, etc.
    - CUDA Co-operative Groups
- **Scale-out**: Inter-node Communication
  - DL and ML Frameworks – most are optimized for single-node only
  - Distributed (Parallel) Execution is an emerging trend



Parallel Programming Models Overview



Shared Memory Model

SHMEM, DSM

Distributed Memory Model

MPI (Message Passing Interface)

Partitioned Global Address Space (PGAS)

OpenSHMEM, UPC, Chapel, X10, CAF, …

- Programming models provide abstract machine models
- Models can be mapped on different types of systems
  - e.g. Distributed Shared Memory (DSM), MPI within a node, etc.
- PGAS models and Hybrid MPI+PGAS models are gradually receiving importance

Overview of the MVAPICH2 Project

- High Performance open-source MPI Library
- Support for multiple interconnects
  - InfiniBand, Omni-Path, Ethernet/iWARP, RDMA over Converged Ethernet (RoCE), and AWS EFA
- Support for multiple platforms
  - x86, OpenPOWER, ARM, Xeon-Phi, **GPGPUs (NVIDIA and AMD)**
- Started in 2001, first open-source version demonstrated at SC '02
- Supports the latest MPI-3.1 standard
- http://mvapich.cse.ohio-state.edu
- Additional optimized versions for different systems/environments:
  - MVAPICH2-X (Advanced MPI + PGAS), since 2011
  - MVAPICH2-GDR with support for NVIDIA GPGPUs, since 2014
  - MVAPICH2-MIC with support for Intel Xeon-Phi, since 2014
  - MVAPICH2-Virt with virtualization support, since 2015
  - MVAPICH2-EA with support for Energy-Awareness, since 2015
  - MVAPICH2-Azure for Azure HPC IB instances, since 2019
  - MVAPICH2-X-AWS for AWS HPC+EFA instances, since 2019
- Tools:
  - OSU MPI Micro-Benchmarks (OMB), since 2003
  - OSU InfiniBand Network Analysis and Monitoring (INAM), since 2015

- **Used by more than 3,125 organizations in 89 countries**
- **More than 1.2 Million downloads from the OSU site directly**
- Empowering many TOP500 clusters (Nov '20 ranking)
  - 4th , 10,649,600-core (Sunway TaihuLight) at NSC, Wuxi, China
  - 9th, 448, 448 cores (Frontera) at TACC
  - 14th, 391,680 cores (ABCI) in Japan
  - 21th, 570,020 cores (Nurion) in South Korea and many others
- Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, OpenHPC, and Spack)
- Partner in the 9th ranked TACC Frontera system
- **Empowering Top500 systems for more than 15 years**

Challenges in Accelerating Digital Pathology with DL

- How can we design a model parallelism solution that is
  - Memory-efficient
  - Offers better training speed compared to state-of-the-art systems
  - Supports emerging real-world use cases like digital pathology

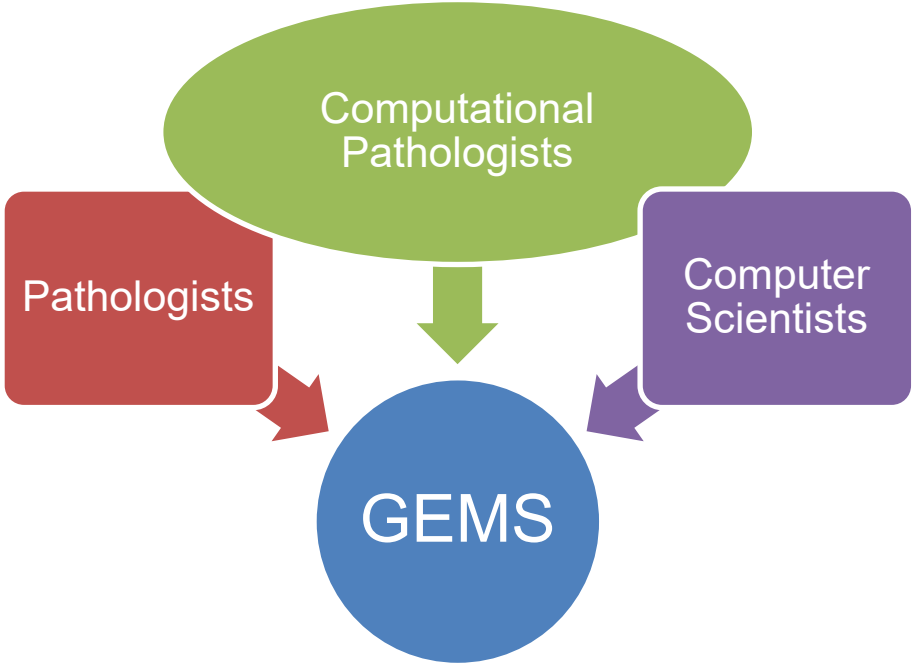
GEMS: GPU-Enabled Memory-Aware Model-Parallelism System for Distributed DNN Training

A Paper at SuperComputing '20

Computer Scientists: Arpan Jain, Ammar A. Awan, Jahanzeb M. Hashmi, Quentin G. Anthony, Hari Subramoni, and Dhabaleswar K. Panda

Computational Pathologists: Asmaa M. Aljuhani, and Raghu Machiraju

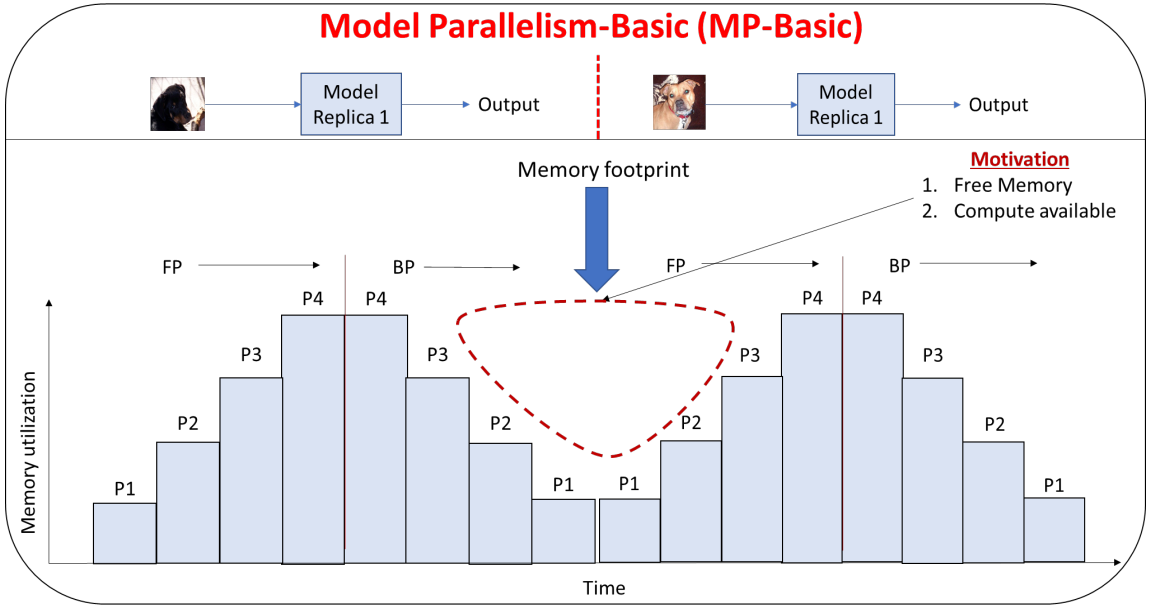
Pathologist: Anil Parwani



Problem with Model Parallelism

Why do we need Memory aware designs?

- Data and Model Parallel training has limitation!
- Maximum Batch Size depends on the memory.
- Basic Model Parallelism suffers from underutilization of memory and compute →



Memory requirement increases with the increase in image size!

Research Challenges

Challenge-1: GPU-based Communication in TensorFlow

Challenge-2: Memory management in TensorFlow

Challenge-3: Scaling Memory-Aware solutions

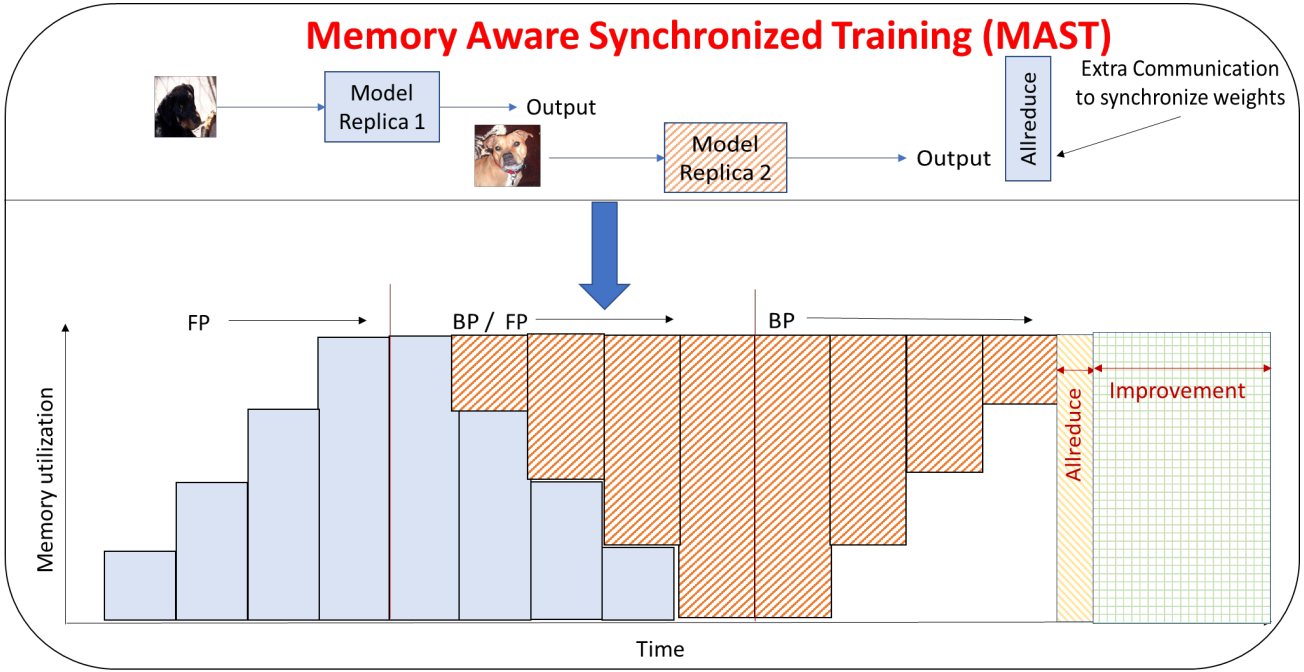
Meet GEMS!

Key Contributions

- Propose, Design, and Evaluate GEMS: an integrated system that provides memory-efficient model parallel training and scalable hybrid parallel training.
- Propose several design schemes
  - Basic Model Parallelism (GEMS-Basic)
  - Memory Aware Synchronized Training (GEMS-MAST)
  - Memory Aware Synchronized Training with Enhanced Replications (GEMS-MASTER)
  - Combination of Model and Data Parallel Training (GEMS-Hybrid)
- Enabled training of High-level TCV classifier on 1024 X 1024 image tiles
- Reduced training time from 7.25 hours to 28 minutes for out-of-core training on 128 Volta V100 GPUs.

GEMS-MAST: Memory Aware Synchronized Training

- GEMS-MAST
  - Uses free memory and compute available between training steps
  - Leverages performance of MPI pt-to-pt. and collectives for communication

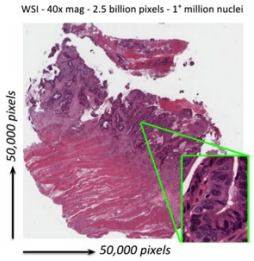


Evaluation Setup

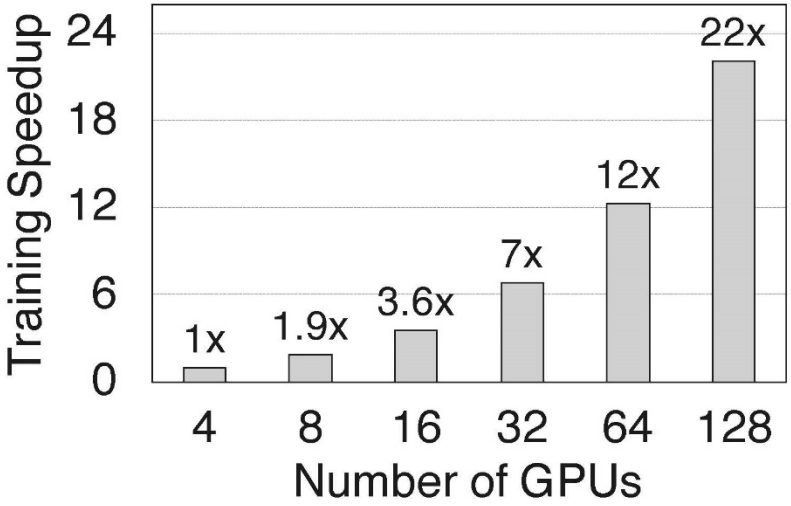
- System
  - Lassen at Lawrence Livermore National Laboratory (LLNL)
    - POWER9 processor
    - 4 NVIDIA Volta V100 GPUs per node
- Interconnect
  - X Bus to connect two NUMA Nodes
  - NVLink is used to connect GPU-GPU and GPU-Processor
  - InfiniBand EDR
- TensorFlow v1.14, MVAPICH2-GDR 2.3.3
- We use and modify model definitions for ResNet(s) from *keras.applications*

Exploiting GEMS in AI-Driven Digital Pathology

- Pathology whole slide image (WSI)
  - Each WSI = 100,000 x 100,000 pixels
  - Can not fit in a single GPU memory
  - Tiles are extracted to make training possible
- Two main problems with tiles
  - Restricted tile size because of GPU memory limitation
  - Smaller tiles loose structural information
- Reduced training time significantly
  - **GEMS-Basic: 7.25 hours (1 node, 4 GPUs)**
  - **GEMS-MAST: 6.28 hours (1 node, 4 GPUs)**
  - **GEMS-MASTER: 4.21 hours (1 node, 4 GPUs)**
  - **GEMS-Hybrid: 0.46 hours (32 nodes, 128 GPUs)**
  - **Overall 15x reduction in training time!!!!**



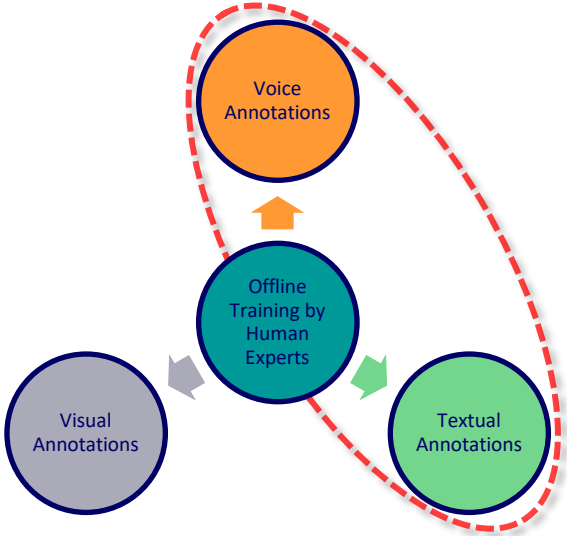
Courtesy: https://blog.kitware.com/digital-slide-archive-large-image-and-histomicstk-open-source-informatics-tools-for-management-visualization-and-analysis-of-digital-histopathology-data/



Scaling ResNet110 v2 on 1024×1024 image tiles using histopathology data

Broad Challenge

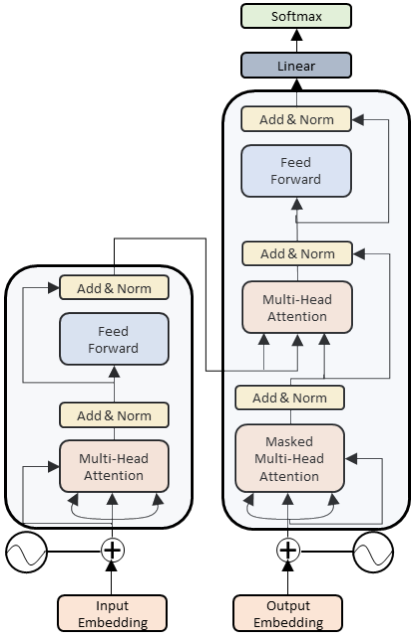
- Training a high-level Sequence Classification model using data parallelism on pathology reports (Text/Audio) can take weeks on a state-of-the-art multi-GPU compute node
  - An example of training DL models based on audio/textual annotation



- Can we accelerate the training using sub-graph parallelism?

Transformers for Audio and Textual Pathology Reports

- Transformers have continually pushed the state-of-the-art in natural language processing and have achieved impressive results in audio processing.
  - Examples: BERT, GPT, GPT-2, GPT-3, T5
  - Multi-head attention module is used by all Transformer models
- Propose SUPER: SUb-Graph Parallelism for TransformERs
  - Accelerates the training of Transformer models for audio and textual data
  - A generalized hybrid of data and sub-graph parallelism (D&SP)
  - Enhanced communication patterns to achieve scalability

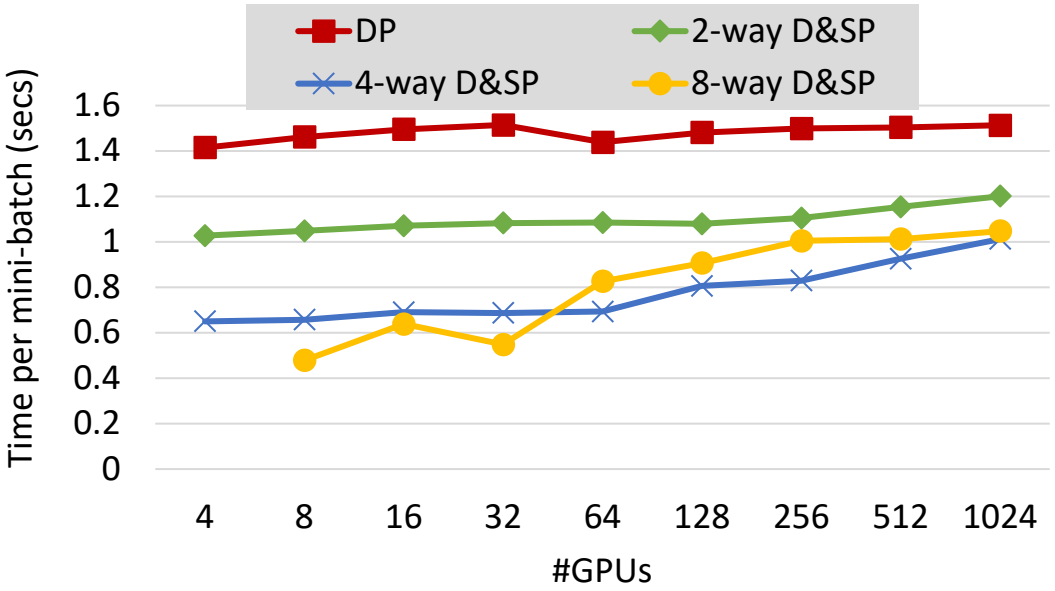


Transformer

A. Jain, T. Moon, T. Benson, H. Subramoni, S. Jacobs, D. Panda, B. Essen, "SUPER: SUb-Graph Parallelism for TransformERs", IPDPS '21 (*Accepted to be presented*)

Exploiting Sub-Graph Parallelism

- Benefits of Sub-Graph parallelism for T5-Large Transformer model
  - Proposed design (D&SP) is up to 2.22X faster than data parallelism



Conclusions

- Next-generation Computational Pathology requires support for HPC, Deep Learning, and Machine Learning
- Requires high-performance middleware designs while exploiting modern HPC technologies
- Provided a set of solutions to achieve
  - MPI (MVAPICH2)-driven solution with Deep Learning Frameworks (TensorFlow, PyTorch and MXNet)
  - Out-of-core training and Hybrid Parallelism for large pathology WSI images
- Looking forward to working with computational pathology community

Funding Acknowledgments

Funding Support by

Equipment Support by

Acknowledgments to all the Heroes (Past/Current Students and Staffs)

Current Students (Graduate)

- Q. Anthony (Ph.D.)
- M. Bayatpour (Ph.D.)
- C.-C. Chun (Ph.D.)
- A. Jain (Ph.D.)
- K. S. Khorassani (Ph.D.)
- P. Kousha (Ph.D.)
- N. S. Kumar (M.S.)
- B. Ramesh (Ph.D.)
- K. K. Suresh (Ph.D.)
- N. Sarkauskas (Ph.D.)
- S. Srivastava (M.S.)
- A. H. Tu (Ph.D.)
- S. Xu (Ph.D.)
- Q. Zhou (Ph.D.)

Current Research Scientists

- A. Shafi
- H. Subramoni

Current Senior Research Associate

- J. Hashmi

Current Software Engineers

- A. Reifsteck
- N. Shineman

Current Research Specialist

- J. Smith

Past Students

- A. Awan (Ph.D.)
- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- R. Biswas (M.S.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- S. Chakraborthy (Ph.D.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- C.-H. Chu (Ph.D.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- J. Hashmi (Ph.D.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- M. Kedia (M.S.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- K. Kulkarni (M.S.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- M. Li (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- K. Raj (M.S.)
- R. Rajachandrasekar (Ph.D.)
- D. Shankar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- N. Sarkauskas (B.S.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)
- J. Zhang (Ph.D.)

Past Research Scientists

- K. Hamidouche
- S. Sur
- X. Lu

Past Programmers

- D. Bureddy
- J. Perkins

Past Research Specialist

- M. Arnold

Past Post-Docs

- D. Banerjee
- X. Besseron
- M. S. Ghazimeersaeed
- H.-W. Jin
- J. Lin
- M. Luo
- E. Mancini
- K. Manian
- S. Marcarelli
- A. Ruhela
- J. Vienne
- H. Wang

Thank You!

panda@cse.ohio-state.edu

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

HiBD
High-Performance
Big Data

The High-Performance Big Data Project
http://hibd.cse.ohio-state.edu/

HiDL
High-Performance
Deep Learning

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/