Approaching the Fifth Paradigm

Nikolay Malitsky

#SAISEco4

Outline

- Paradigm Shift
- Spark-MPI Approach
- MPI-Based Deep Learning Applications
- Next: Reinforcement Learning Applications

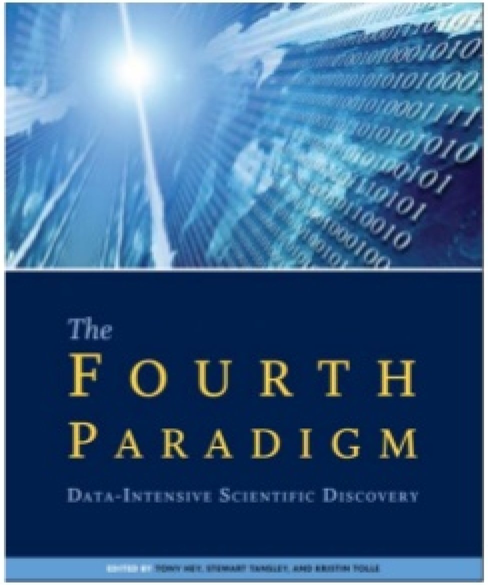
Four Science Paradigms*

1. **Experimental:** describe empirical facts and test hypotheses
   since: thousand years ago

2. **Theoretical:** explain and predict natural phenomena using models and abstractions
   since: several hundred years ago

3. **Computational:** simulate theoretical models using computers
   since: second half of the 20th century

4. **Data-Intensive:** scientific discoveries based on Big Data analytics
   since: around 15 years ago



*Jim Gray and Alex Szalay, eScience – A Transformed Scientific Method, NRC-CSTB, 2007

Paradigm Shift

- The fourth paradigm of data-intensive science rapidly became a major conceptual approach for multiple application domains encompassing and generating large-scale scientific drivers such as fusion reactors and light source facilities.
- The success of data-intensive projects subsequently triggered an explosion of numerous machine learning approaches addressing a wide range of industrial and scientific applications such as computer vision, self-driving cars, and brain modelling.
- The next generation of artificial intelligent systems clearly represents a paradigm shift from data processing pipelines towards cognitive **knowledge-centric** applications.
- As shown in Fig. 1, AI systems **broke the boundaries of computational and data-intensive paradigms** and began to form a new ecosystem by merging and extending existing technologies.

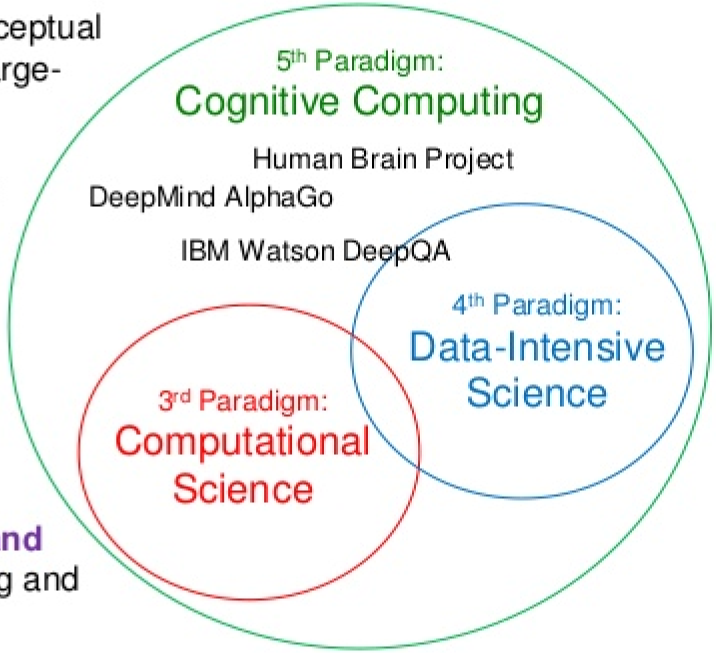


Figure 1: The Fifth Paradigm*

*N. Malitsky, R. Castain, and M. Cowan, Spark-MPI: Approaching the Fifth Paradigm of Cognitive Applications, arXiv:1806.01110, 2018

Knowledge

- In his original talk, Jim Gray discussed "objectifying" knowledge within the field of ontology for providing a structured representation of abstract concepts and physical entities. This direction is related with the development of structured knowledge bases and associated technologies such as the Semantic Web and Linked Data.
- Existing structured resources however only capture a tiny subset of available information. Therefore, advanced question-answering (QA) systems* augmented them with corpora of raw text and processing pipelines consisting of multiple stages that combine hundreds of different cooperating algorithms from various fields.

As a result, emerging AI-oriented applications imply a more general and practical knowledge definition:

Knowledge is a multifacet substance distributed among heterogeneous information networks and associated processing platforms. The structure and relationship between different components of such a composite representation is dynamic, continuously shaped and consolidated by machine learning processes.

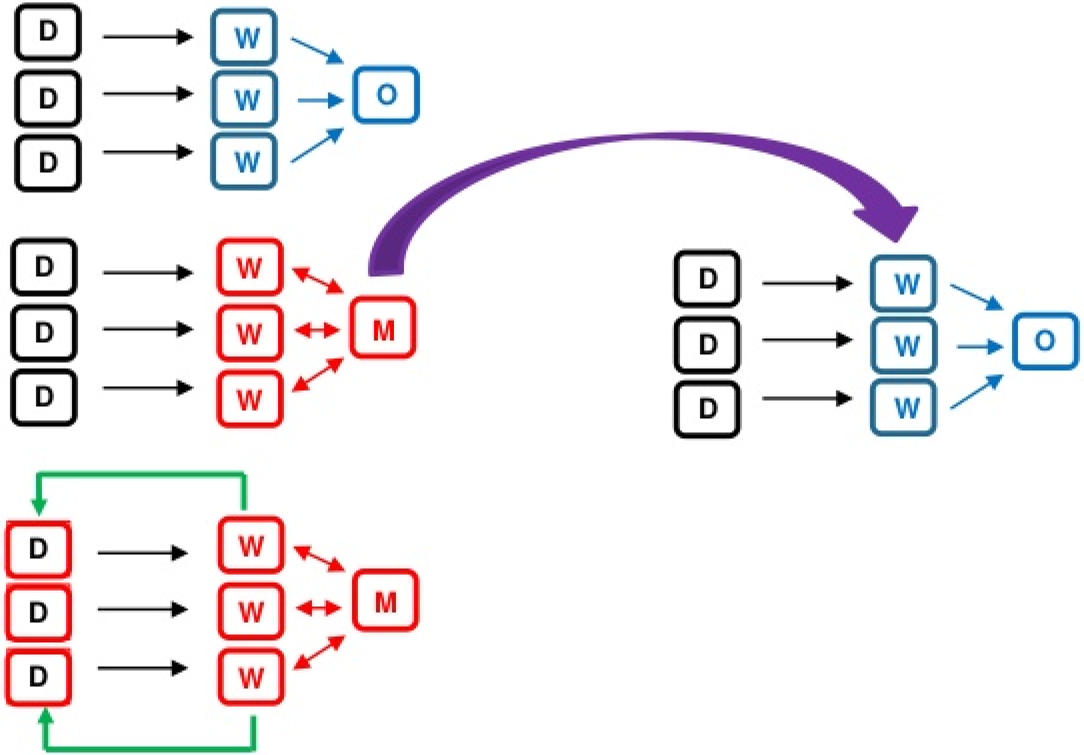
*D. A. Ferrucci, Introduction to "This is Watson", IBM Journal of Research and Development, 2012

From Processing Pipelines to Rational Agents

Data-intensive
processing pipelines

Deep learning
model-centric applications

Reinforcement learning
agent-oriented applications



Approaching the Fifth Paradigm of Cognitive Applications

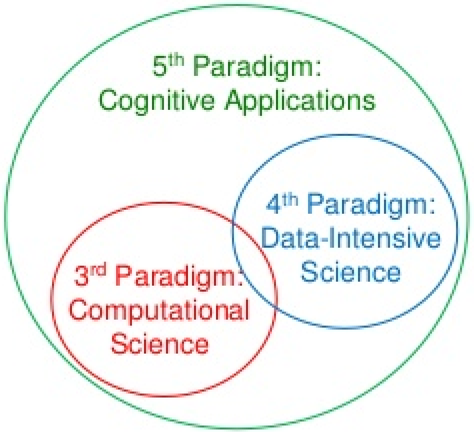


Figure 1: The Fifth Paradigm

The consolidation of HPC and Big Data machine learning technologies represents the prerequisite for developing the next paradigm of cognitive applications

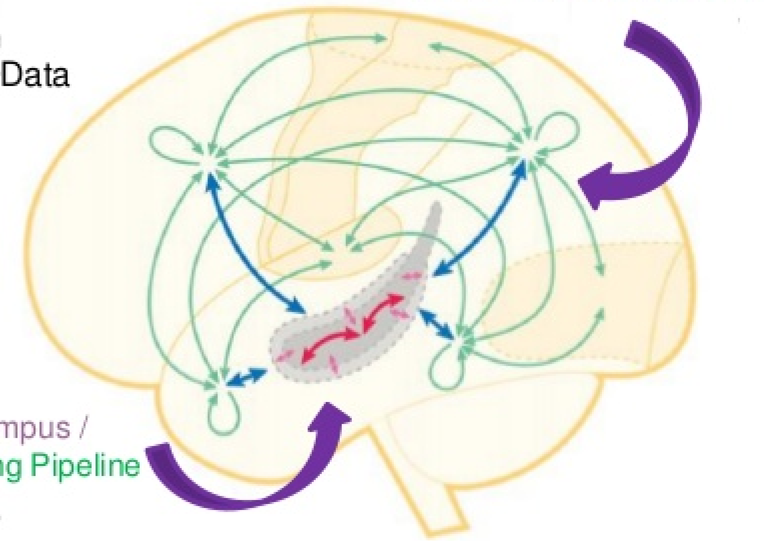
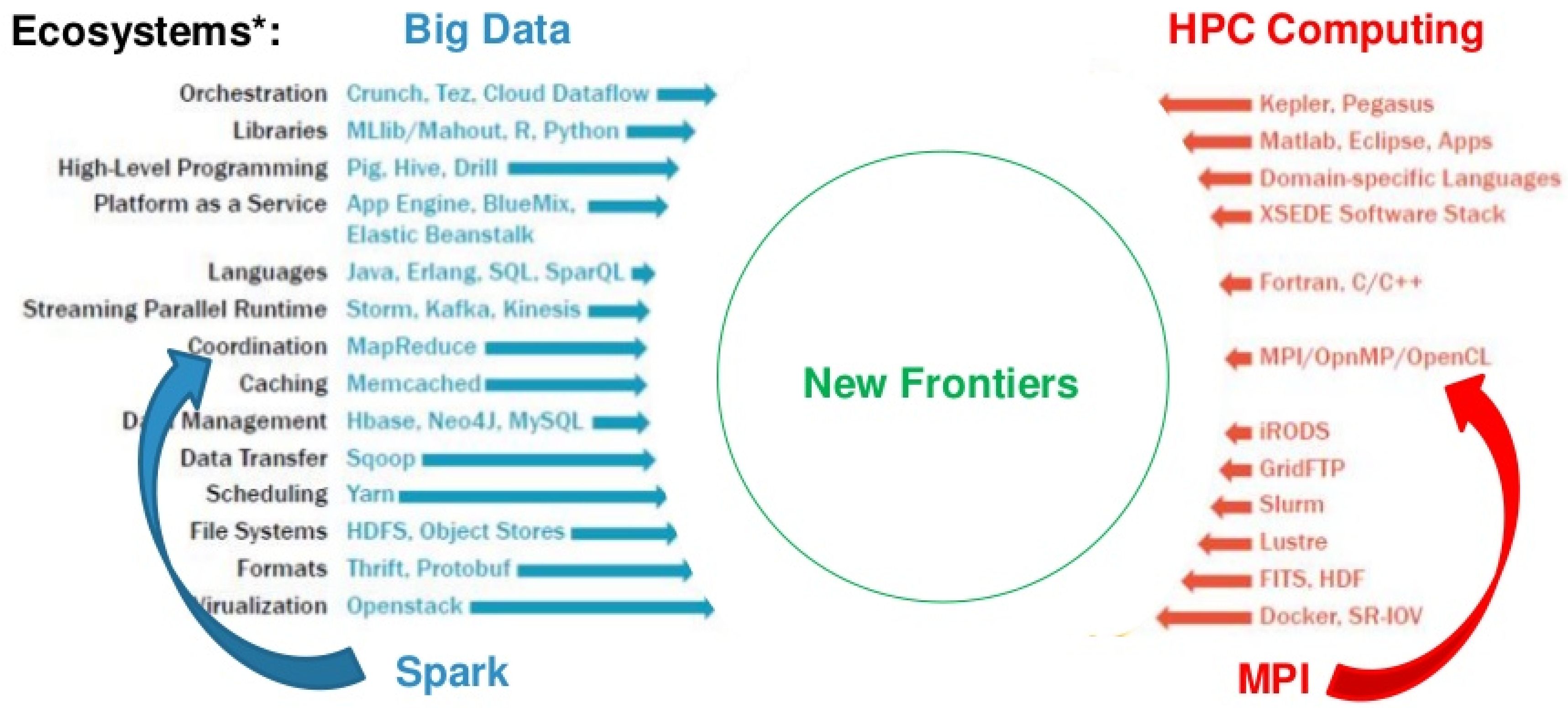


Figure 2: Complementary Learning Systems*

*Dharshan Kumaran, Demis Hassabis, and James L. McClelland, What Learning Systems do Intelligent Agents Need? Complementary Learning Systems, Trends in Cognitive Sciences, 2016

Spark-MPI Approach

Closing the gap between Big Data and HPC computing



*Geoffrey Fox et al. HPC-ABDC High Performance Computing Enhanced Apache Big Data Stack, CCGrid, 2015

MPI: Message Passing Interface

Application Programming Interface:

- peer-to-peer: allreduce
- master-workers: scatter, gather, reduce
- point-to-point: send, receive
- remote memory access: put, get

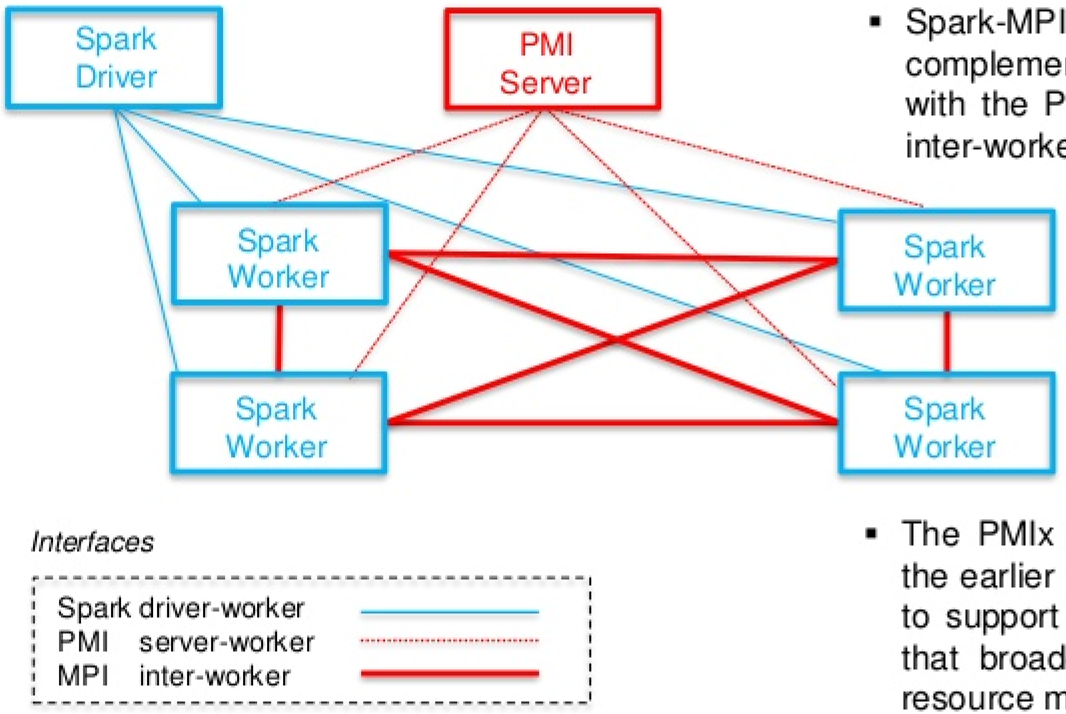
Portable Access Layer for various communication protocols:

- RDMA
- GPUDirect RDMA
- TCP/IP
- shared memory

Process Management Interface:

- address exchange service
- …

PMI-based Spark-MPI Approach



- Spark-MPI[1] encompasses three interfaces. Specifically, it complements the Spark conventional driver-worker model with the PMI server-worker interface for establishing MPI inter-worker communications.
  - Process Management Interface (PMI): originally developed by the MPICH team[2] and used for exchanging wireup information among processes.
  - PMI-Exascale (PMIx): created by the Open MPI team[3] in response to the ever-increasing scale of supercomputing clusters.
- The PMIx community has therefore focused on extending the earlier PMI work, adding flexibility to existing APIs (e.g., to support asynchronous operations) as well as new APIs that broaden the range of interactions with the resident resource manager.

---

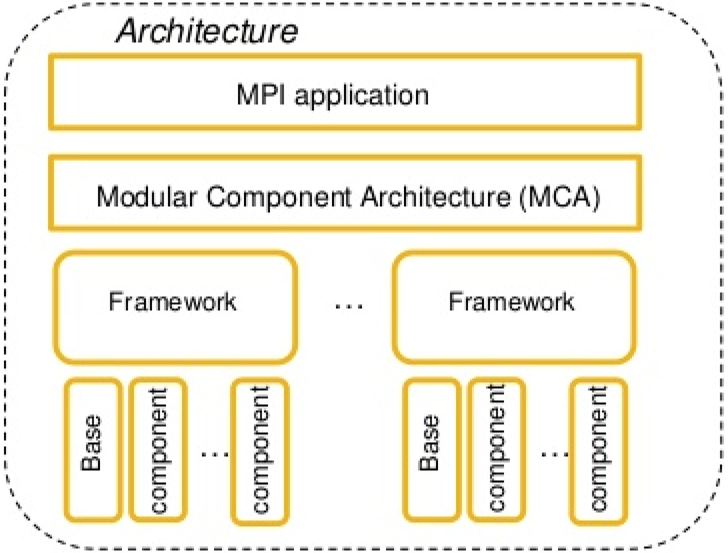
(1) N. Malitsky et al. Building Near-Real-Time Processing Pipelines with the Spark-MPI platform, NYSDS, 2017

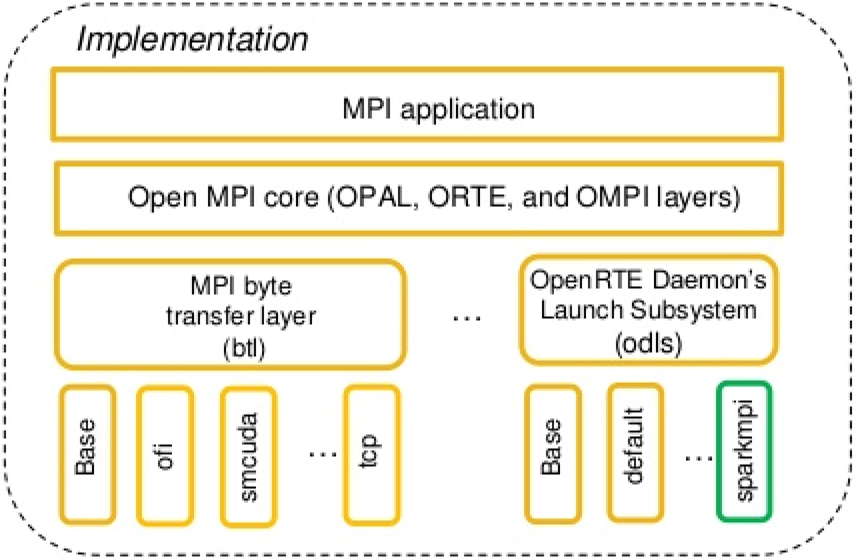
(2) P. Balaji et al. PMI: A Scalable Parallel Process-Management Interface for Extreme-Scale Systems, 2010

(3) R. Castain, D. Solt, J. Hursey, and A. Bouteiller, PMIx: Process Management for Exascale Environment, 2017

Open MPI*

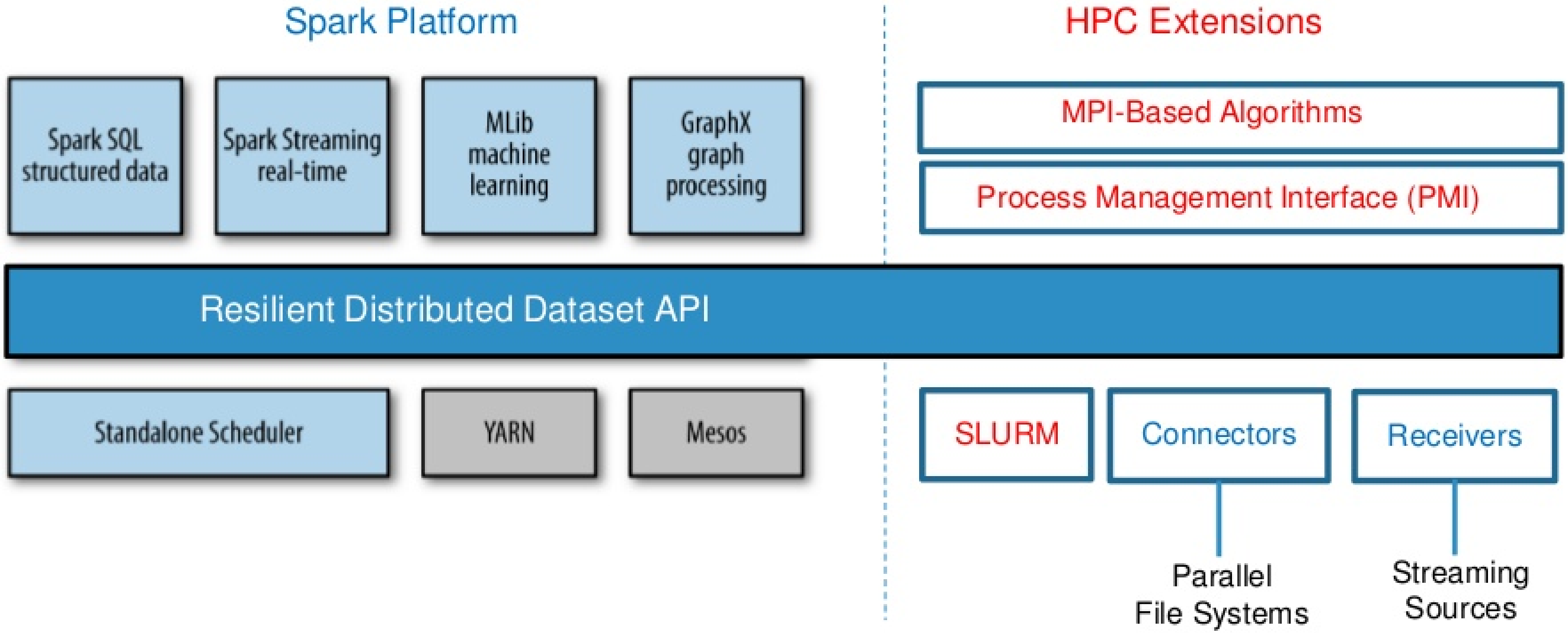
Open MPI was derived as a generalization of four projects bringing together over 40 frameworks. It introduced a Modular Component Architecture (MCA) that utilized components (a.k.a. plugins) to provide alternative implementations of key functional blocks such as message transport, mapping, algorithms, and collective operations.





*E.Gabriel, G.E. Fagg, G. Bosilca, T. Anhskun, J. J. Dongarra, J. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation, 2004

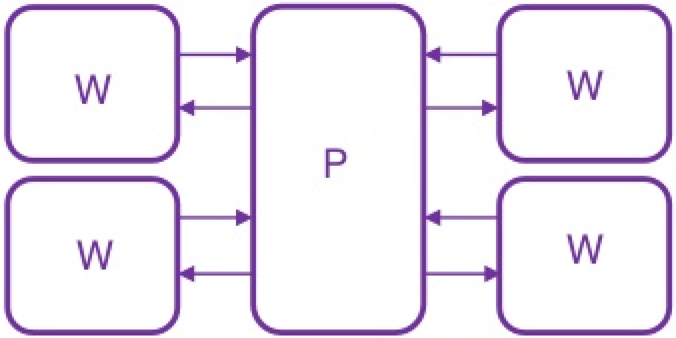
Spark-MPI Integrated Platform



N. Malitsky, R. Castain, and M. Cowan, Spark-MPI: Approaching the Fifth Paradigm of Cognitive Applications, arXiv:1806.01110, 2018

MPI-Based Deep Learning Applications

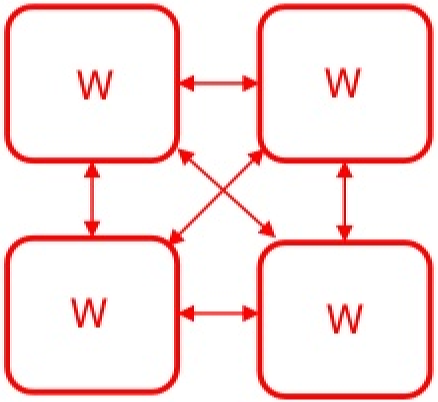
Deep Learning Training as a Third Paradigm Computational Application



Parameter Server-based Data Parallel Model*

P: Parameter Server
W: DL Worker

All-Reduce Model



*Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, 2015

(Some of the) MPI DL Projects

- **CNTK**[1]: Microsoft Cognitive Toolkit
- **TensorFlow-Matex**[2]: added two new TensorFlow operators, Global_Broadcast and MPI_Allreduce
- **S-Caffe**[3]: scaled Caffe with the MPI level hierarchical reduction design
- **Horovod**[4]: adopted Baidu's approach based on the ring-allreduce algorithm and further developed its implementation with NVIDIA's NCCL library for collective implementation
- **CPE ML Plugin**[5]: Cray Programming Environment Machine Learning Plugin

---

(1) A. Agarwal et al. An Introduction to Computational Networks and Computational Network Toolkit, 2014
(2) A. Vishnu et al. User-transparent distributed TensorFlow, 2017
(3) A. A. Awan et al. S-Caffe: co-designing MPI runtime and Caffe for scalable deep learning on modern GPU clusters, 2017
(4) A. Segeev and M. Del Balso. Horovod: fast and easy distributed deep learning in TensorFlow, 2018
(5) P. Mendygral. Scaling Deep Learning, 2018

Spark-MPI-Horovod

The Horovod MPI-based training framework replaces the TensorFlow parameter servers with the ring-allreduce approach for averaging gradients among TensorFlow workers.

For users, the corresponding integration consists of two primary steps as illustrated by the script: (1) initializing Horovod with *hvd.init()* and (2) wrapping TensorFlow worker's optimizer with *hvd.DistributedOptimizer().*

The Spark-MPI pipelines enable to process the Horovod training on Spark workers with Map operations. To establish MPI communication among the Spark workers, the Map operation (e.g. *train()*) needs only to define PMI-related environmental variables (such as PMIX_RANK and a port number).

```python
def train(pid, parts):

    import tensorflow as tf
    import horovod.tensorflow as hvd
    import mnist_app

    log_string = mnist_app.get_log_string(1024)

    # define the MPI environmental variables
    os.environ["PMIX_RANK"] = str(pid)
    for env in parts:
        for key in env:
            os.environ[key] = env[key]
```

Initialize the PMI environmental variables

```python
    # initialize Horovod
    hvd.init()
```

Initialize Horovod and MPI

```python
    # Extract the MNIST dataset
    learn = tf.contrib.learn
    mnist = learn.datasets.mnist.read_data_sets('MNIST-data-%d' % hvd.rank())
```

Extract the MNIST dataset

```python
    # Build model...
    with tf.name_scope('input'):
        image = tf.placeholder(tf.float32, [None, 784], name='image')
        label = tf.placeholder(tf.float32, [None], name='label')
    predict, loss = mnist_app.conv_model(image, label, tf.contrib.learn.ModeKeys.TRAIN)

    global_step = tf.train.get_or_create_global_step()
```

Build the DL model

```python
    # Horovod: add Horovod Distributed Optimizer.
    opt = tf.train.RMSPropOptimizer(0.001 * hvd.size())
    opt = hvd.DistributedOptimizer(opt)
    train_op = opt.minimize(loss, global_step=global_step)
```

Crate the TF optimizer

Wrap TF with Horovod

…

```python
    with tf.train.MonitoredTrainingSession(checkpoint_dir=checkpoint_dir,
                                           hooks=hooks,
                                           config=config) as mon_sess:
        while not mon_sess.should_stop():
            # Run a training step synchronously.
            image_, label_ = mnist.train.next_batch(100)
            mon_sess.run(train_op, feed_dict={image: image_, label: label_})

    log_contents = log_string.getvalue()
    log_string.close()

    yield log_contents
```

Run the Horovod MPI-based training

```python
log_contents = rdd.mapPartitionsWithIndex(train).collect()
```

Run the Horovod training on Spark workers

Next

Deep Reinforcement Learning

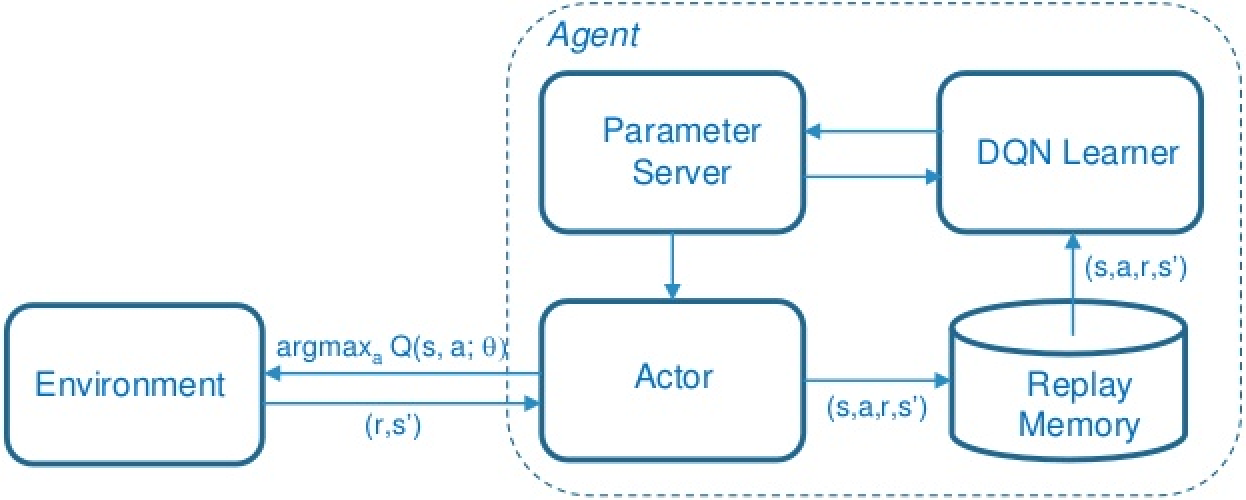


Figure 1: Gorila* (General Reinforcement Learning Architecture)

System Requirements**:

- Low latency
- High throughput
- Dynamic task creation
- Heterogeneous tasks
- Arbitrary dataflow dependencies
- Transparent fault tolerance
- Debuggability and profiling

---

*A. Nair et al. Massively Parallel Methods for Deep Reinforcement Learning, ICML, 2015

**R. Nishihara et. al. Real-Time Machine Learning: The Missing Pieces, arXiv 1703.03924, 2017

(Some of the) RL Applications*

- Atari Games[1]
- AlphaGo[2]
- Robotics
- Self-driving vehicles
- Autonomous UAVs

…

range

"*Pterodactylus antiquus*, the first pterosaur species to be named and identified as a flying reptile … 150.8–148.5 million years ago" (Wikipedia)

(1) V. Mnih et al. Playing Atari with Deep Reinforcement Learning, NIPS, 2013
(2) D. Silver et al. Mastering the game of Go with deep neutral networks an tree search, Nature, 2016

Summary

- Emerging AI projects represent a paradigm shift from data processing pipelines towards the fifth paradigm of cognitive knowledge-centric applications.

- Knowledge is a multifacet substance distributed among heterogeneous information networks and associated processing platforms. The structure and relationship between different components is dynamic, continuously shaped and consolidated by machine learning processes.

- The new generation of AI composite applications requires the integration of Big Data and HPC technologies. For example, MPI was originally introduced within the computational paradigm ecosystem for developing HPC scientific applications. But recently, MPI was successfully applied for extending the scale of deep learning applications.

- Spark-MPI addresses this strategic direction by extending the Spark platform with MPI-based HPC applications using the Process Management Interface (PMI).