



DATA ANALYSIS AND MACHINE LEARNING WITH INTEL[®] ARCHITECTURE

One-Day Workshop

Andrey Vladimirov and Ryo Asai
Colfax International — colfaxresearch.com

February 2017

WELCOME

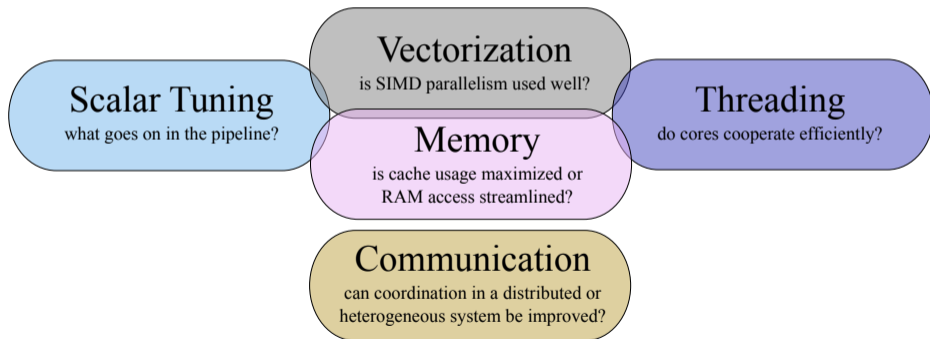
While best efforts have been used in preparing this training, Colfax International makes no representations or warranties of any kind and assumes no liabilities of any kind with respect to the accuracy or completeness of the contents and specifically disclaims any implied warranties of merchantability or fitness of use for a particular purpose. The publisher shall not be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused, or alleged to have been caused, directly or indirectly, by the information or programs contained herein. No warranty may be created or extended by sales representatives or written sales materials.



§1. QUICK REVIEW OF YESTERDAY

Code Modernization

Optimizing software to better utilize features available in modern computer architectures.





THE "HOW" SERIES TRAINING

DEEP DIVE

WITH CODE MODERNIZATION EXPERTS

It's free

→ [HOWSERIES.COM](https://howseries.com)

*10x 2-hour sessions | 24-hour 2-weeks remote access to a system

COLFAX RESEARCH

Log In/Out or Register

READ
WATCH
LEARN
CONNECT
JOIN

To search, type and hit enter

Popular

The Hands-On Tutorials (HOT) webinars: details on efficient programming for Intel architecture

The Hands-On Workshop (HOW) Series

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation

Parallel Programming Book

Introduction to parallel programming, deep discussion of optimization techniques, exercises. © 2015, Colfax International, 508 pages.

Research and Educational Publications

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation

Software Developer's Introduction to the HGST Ultrastar Archive H800 SMR Drives

Optimization Techniques for the Intel MIC Architecture, Part 3 of 3: False Sharing and Padding

Optimization Techniques for the Intel MIC Architecture, Part 2 of 3: Strip-Mining for Vectorization

Optimization Techniques for the Intel MIC Architecture, Part 1 of 3: Multi-Threading and Parallel Reduction

Performance to Power and Performance to Cost Ratios with Intel Xeon Phi Coprocessors (and why ix Acceleration May Be Enough)

Featured Video

See Research material on vectorization in a streaming mode

[http://dx.doi.org/10.1007/978-1-4939-9999-9_11](#)

Consulting

[Share](#)

Colfax offers consulting services for enterprises, research help you:

- Optimize your existing application to take advantage of parallelism, from vectors to cores to clusters and
- Future-proof your application for upcoming innovations
- Accelerate your application using coprocessor tech
- Investigate the potential system configurations that satisfy your cost, power, performance requirements.
- Take a clean slate to develop a novel approach to reduce your computing pro

All Video Content - CPU VSI - Chapter 1 - Episode 1.1

Episode 2.1 — Purpose of the MIC architecture

[Share](#)

Software Developer's Introduction to the HGST Ultrastar Archive H800 SMR Drives

In this paper we will discuss the new HGST Ultrastar Archive H800 SMR drives, their architecture, and how they compare to other high capacity storage solutions. These drives are well suited for large scale archival applications in a wide range of applications. The paper is intended for system architects.

The HGST Ultrastar Archive H800 SMR drives are well suited for large scale archival applications in a wide range of applications. The paper is intended for system architects.

Fluid Dynamics with Fortran on Intel Xeon Phi coprocessors

In this demonstration, a Colfax researcher presents a solution to the Navier-Stokes equations for a 2D flow problem. The solution is implemented in Fortran and runs on an Intel Xeon Phi coprocessor. The results are compared to those of a serial Fortran implementation. The video shows the Fortran code and the performance results.

Configuration and Benchmarks of Peer-to-Peer Communication over Gigabit Ethernet and InfiniBand in a Cluster with Intel Xeon Phi Coprocessors

A few weeks ago we conducted a conversation with James Reinders, the Director of Intel's Parallel Computing Group. He discussed the state of the parallel programming and hardware in general, might not only be interested about the hardware, but also about the software and the challenges of high performance computing.

Interview with James Reinders: future of Intel MIC architecture, parallel programming, education

A few weeks ago we conducted a conversation with James Reinders, the Director of Intel's Parallel Computing Group. He discussed the state of the parallel programming and hardware in general, might not only be interested about the hardware, but also about the software and the challenges of high performance computing.

http://colfaxresearch.com/



§2. ML IN THE CONTEXT OF HPC

THREE APPROACHES

Low Level Approach

Apply code modernization techniques to frameworks/applications.

- ▶ Colfax Research Website, HOW series, Intel Modern Code page etc.

High Level Approach

Use high level libraries that are pre-optimized for modern architectures.

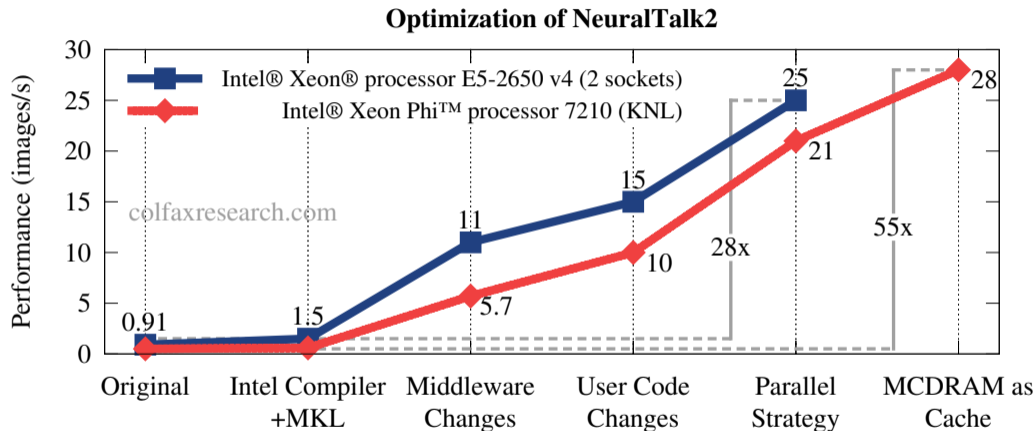
- ▶ IntelCaffe, TensorFlow, Scikit-learn etc.

Middle Ground Approach

Integrate pre-optimized kernels into frameworks/applications.

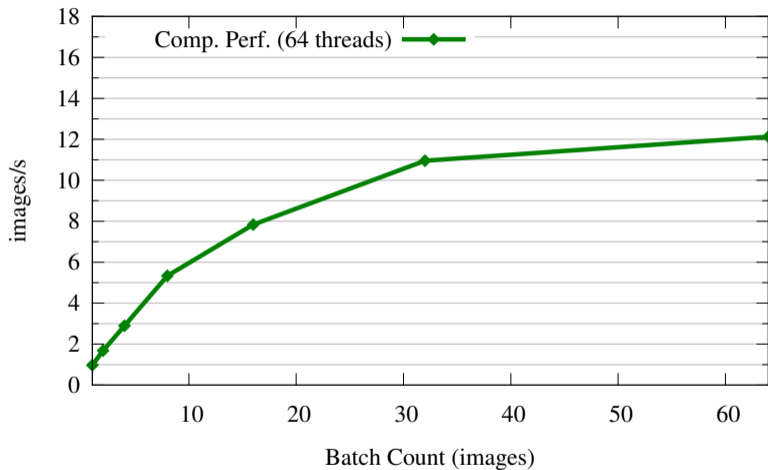
- ▶ Intel[®] MKL DNN primitives, Intel[®] DAAL, Intel[®] MKL DNN etc.

CASE STUDY: VGG-NET ON TORCH



Colfax Research [Summary Paper](#)

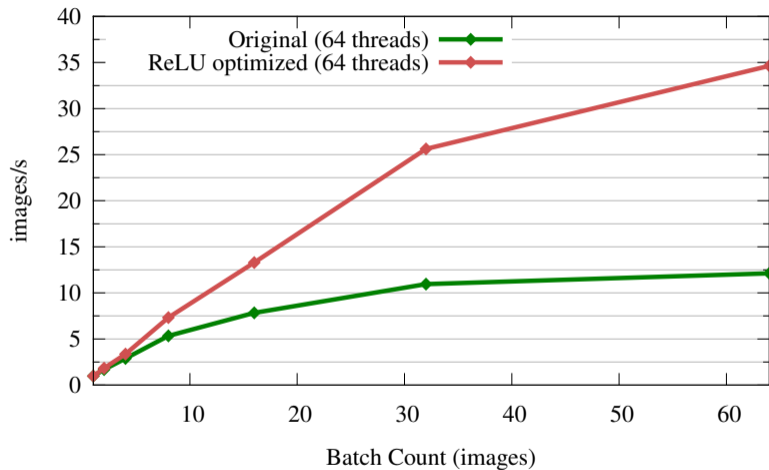
BASE TORCH PERFORMANCE



By Layer:

- ▶ ReLU: 66%
- ▶ Conv: 30%
- ▶ MaxPool: 3%
- ▶ Other: <1%

PERFORMANCE AFTER RELU OPTIMIZATION



ReLU -> 160x boost

By Layer:

- ▷ ReLU: 1%
- ▷ Conv: 85%
- ▷ MaxPool: 11%
- ▷ Other: 3%

FALCON LIBRARY: FAST IMAGE CONVOLUTION IN NEURAL NETWORKS ON INTEL ARCHITECTURE

WINOGRAD'S MINIMAL FILTERING ALGORITHM
APPLIED TO CONVOLUTION IN MACHINE LEARNING



<https://colfaxresearch.com/falcon-library/>



§3. INTEL-OPTIMIZED FRAMEWORKS



INTEL MKL: THE MAGIC PILL

Intel[®] Math Kernel Library (MKL) — standard mathematical functions optimized for Intel architecture.

Dense Linear Algebra

BLAS + PBLAS
LAPACK + ScaLAPACK
Extended eigensolver

Fourier Transform

Multi-threaded
Cluster mode
1D and multi-dimensional

Statistics and Probability

Random number generators
Convolution and correlation
Summary statistics

Sparse Linear Algebra

SpBLAS
Iterative, direct solvers
Preconditioners

Numerical Analysis

Partial differential equations
Nonlinear optimization
Data fitting. Vector math.

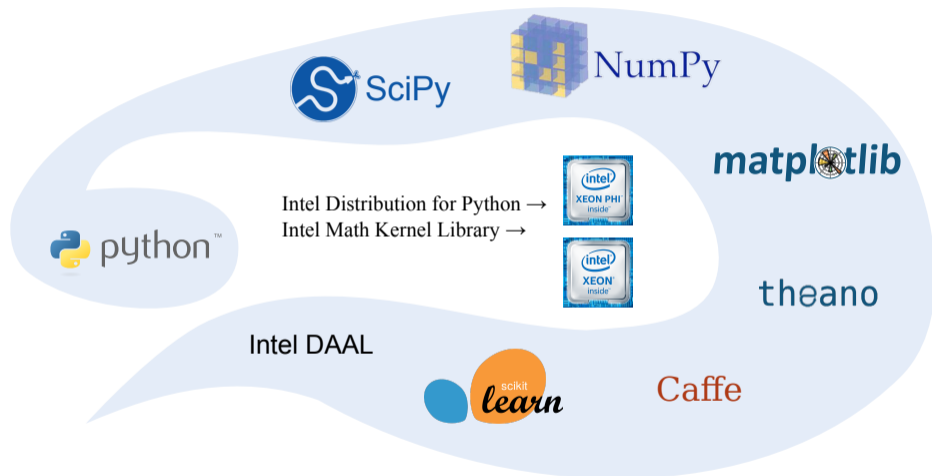
Machine Learning

New

DNN Primitives:
Convolution. Inner product.
ReLU, LRNC, etc.



INTEL PYTHON



Portal: software.intel.com/intel-distribution-for-python. See also: [CR paper](#).

DGEMM WITH SCIPLY

```
import numpy as np
from numpy.random import rand as rn
from scipy.linalg.blas import dgemm as dgemm

m=5000
n=5000
k=5000

// Using column-major format with order="F"
A = np.array(np.random.rand(m, n), dtype = np.double, order="F");
B = np.array(np.random.rand(n, k), dtype = np.double, order="F");

// Scipy matrix-matrix multiplication
C=dgemm(alpha=1.0, a=A, b=B, c=C, overwrite_c=1, trans_b=1)
```

DGEMM WITH NUMPY

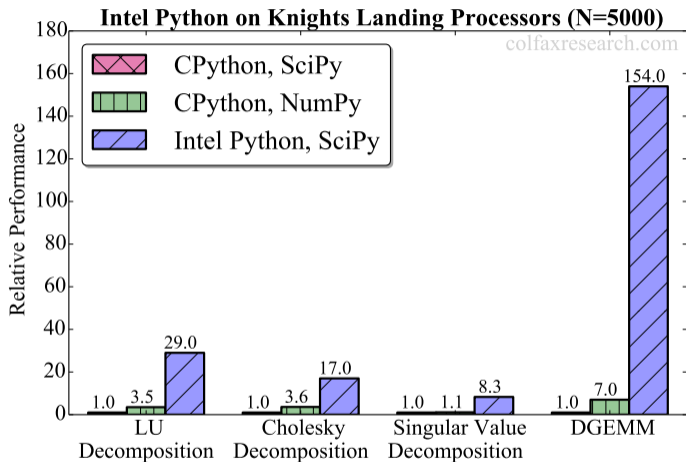
```
import numpy as np
from numpy.random import rand as rn

m=5000
n=5000
k=5000

// Using row-major by default (you can also set order="C")
A = np.array(np.random.rand(m, n), dtype = np.double);
B = np.array(np.random.rand(n, k), dtype = np.double);

// Numpy matrix-matrix multiplication
C = np.dot(A, B)
```

INTEL PYTHON PERFORMANCE



Portal: software.intel.com/intel-distribution-for-python. See also: [CR paper](#).

MISC. SCIPY WORKLOADS

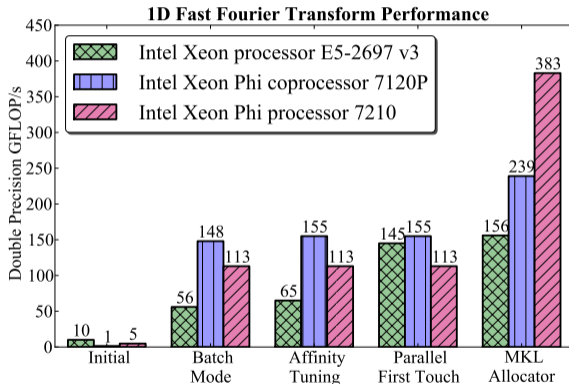
```
from scipy.linalg import lu as lu;
# ... #
A = np.array(np.random.rand(k, k), dtype = np.double, order='F')
P,L,U = lu(A, permute_l=False, overwrite_a=True, check_finite=False)
```

```
from scipy.linalg import cholesky as cholesky
# ... #
A = np.array(tempA, dtype=np.double, order='F')
L = cholesky(A, overwrite_a=True, check_finite=False)
```

```
from scipy.linalg import svd as svd
# ... #
A = np.array(np.random.rand(k, k), dtype = np.double, order='F');
U, s, V = svd(A)
```

TUNING MAY BE NECESSARY

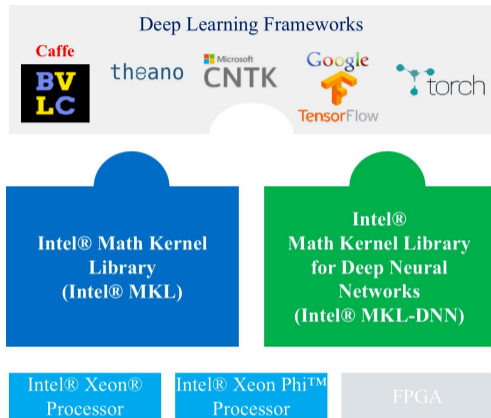
$2 \cdot 10^5$ FFTs of size 2048. See [HOW “KNL”](#) for details.



A solid green vertical bar is positioned on the left side of the slide, extending from the top to the bottom.

§4. MIDDLE GROUND: ML PERFORMANCE LIBRARIES

INTEL MKL AND INTEL MKL-DNN



Intel® MKL	Intel® MKL-DNN
DNN primitives + wide variety of other math functions	DNN primitives
C DNN APIs (C++ future)	C/C++ DNN APIs
Binary distribution	Open source DNN code ¹
Free community license. Premium support available as part of Intel® Parallel Studio XE	Apache 2.0 license
Broad usage DNN primitives; not specific to individual frameworks	Multiple variants of DNN primitives as required for framework integrations
Quarterly update releases	Rapid development ahead of Intel MKL releases

¹ GEMM building blocks are binary.

slide credit: Intel corp.

STAND-ALONE EXAMPLE: CONVOLUTION

```
1 // Creating MKL DNN primitive object
2 dnnPrimitive_t convFwd;
3 dnnConvolutionCreateForward_F32(&convFwd, NULL, dnnAlgorithmConvolutionDirect,
4                                 dim, input_dims, output_dims, filter_dims,
5                                 conv_strides, padding, dnnBorderZeros);
6
7 // Creating the needed data buffer
8 void* conv_res[dnnResourceNumber];
9 conv_res[dnnResourceSrc] = (void*) input;
10 conv_res[dnnResourceFilter] = (void*) filter;
11 conv_res[dnnResourceDst] = (void*) output;
12
13 // Execute the workload
14 dnnExecute_F32(pConvFwd, conv_res);
```

For more: [Intel MKL documentation on DNN primitives](#)



EXAMPLE WITH INTEL CAFFE

EXAMPLE INTEGRATION: INTEL CAFFE

GitHub link: <https://github.com/intel/caffe/>

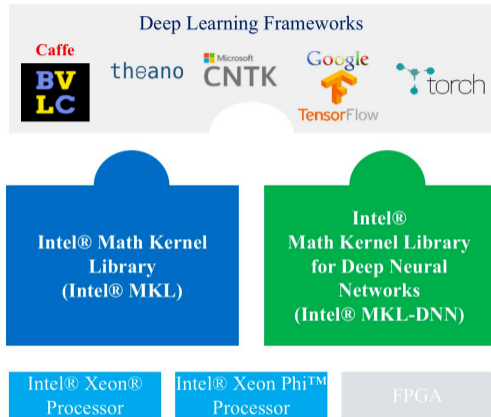
Example layer implementations: `caffe/src/caffe/layers/mkl_*.cpp`

```
1 // Grabbing parameters from Caffe Layers
2 PoolingParameter pool_param = this->layer_param_.pooling_param();
3 channels_ = bottom[0]->channels();
4 height_ = bottom[0]->height();
5 width_ = bottom[0]->width();
6 num_ = bottom[0]->num();
7 // ... //
8 kernel_h_ = pool_param.kernel_h(); kernel_w_ = pool_param.kernel_w();
9 // ..... //
10
11 // Creating the math kernel from these parameters
12 status = dnnPoolingCreateForward<Dtype>( /* ... */ );
```



§5. HIGH LEVEL: ML FRAMEWORKS

INTEL MKL AND INTEL MKL-DNN



Intel® MKL	Intel® MKL-DNN
DNN primitives + wide variety of other math functions	DNN primitives
C DNN APIs (C++ future)	C/C++ DNN APIs
Binary distribution	Open source DNN code ¹
Free community license. Premium support available as part of Intel® Parallel Studio XE	Apache 2.0 license
Broad usage DNN primitives; not specific to individual frameworks	Multiple variants of DNN primitives as required for framework integrations
Quarterly update releases	Rapid development ahead of Intel MKL releases

¹ GEMM building blocks are binary.

slide credit: Intel corp.

(CURRENTLY) AVAILABLE LIBRARIES

Machine Learning frameworks that have been optimized for Intel Architectures

- ▶ Intel Caffe - Python - Deep Learning
[GitHub](#)
- ▶ Scikit-learn - Python - Classical ML
Part of [Intel Python](#)
- ▶ Intel Theano/Keras - Python - Deep Learning
Github [Theano/Keras/](#)
- ▶ Intel Torch - Lua - Classical ML, Deep learning
[GitHub](#)
- ▶ Intel DAAL - Java/C++/Python - Classical ML, Deep Learning

... and more to come!

USING LIBRARIES ON THE CLUSTER

▷ Intel Caffe

```
~> /opt/intel/caffe/build/caffe
```

▷ Scikit-learn

```
import sklearn
```

▷ Theano/Keras

```
import theano  
import keras
```

▷ Intel DAAL

```
1 #include "daal"
```



§6. DISTRIBUTED DEEP LEARNING

"FLOPS ARE CHEAP"?

Theoretical estimates, Intel[®] Xeon E5-2697 V3 processor

Performance = 28 cores \times 2.7 GHz \times (256/64) vec.lanes \times 2 FMA \times 2 FPU \approx 1.2 TFLOP/s

Required Data Rate = 1.2 TFLOP/s \times 8 bytes \approx 10 TB/s

OPA Max Bandwidth = 12.5 GB/s \approx 0.01 TB/s

Ratio = 10/0.01 \approx 1000 (FLOPs)/(Memory Transferred)

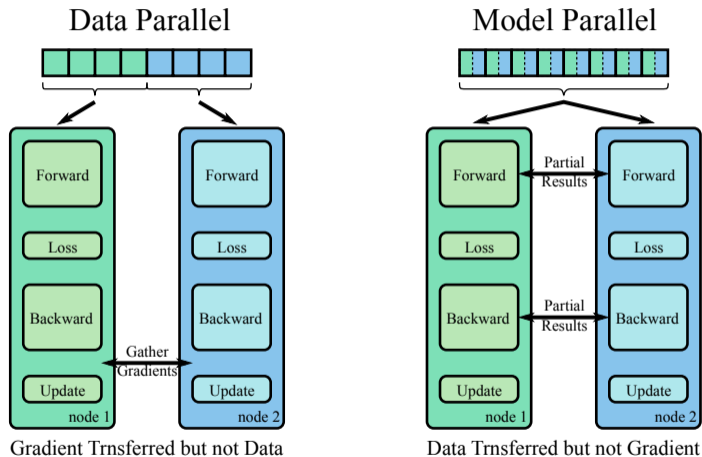
To put it short...

Difficulty of Distributed Computation

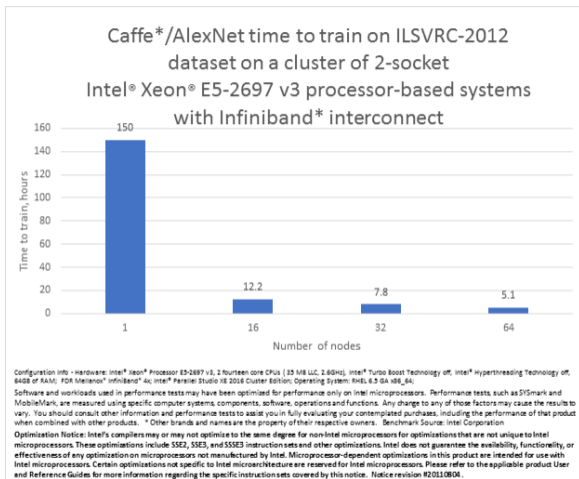
In the time it takes to transfer one data element, processors can do thousands of operation on one data element.

DISTRIBUTED COMPUTATION FOR NEURAL NETWORKS

Because of issue with large mini-batches, it is not easy to scale DNN.



Intel Machine Learning Scaling Library



Source: Intel® Corporation. (Caffe* Training on Multi-node Distributed-memory Systems Based on Intel® Xeon® Processor E5 Family)



MACHINE LEARNING FRAMEWORK: INTEL[®] DAAL

Analysis

- Low Order Moments
- Quantile
- Correlation and Variance
- Cosine Distance Matrix
- Correlation Distance Matrix
- K-Means Clustering
- Principal Component Analysis
- Cholesky Decomposition
- Singular Value Decomposition
- QR Decomposition
- Expectation-Maximization
- Multivariate Outlier Detection
- Univariate Outlier Detection
- Association Rules
- Kernel Functions
- Quality Metrics

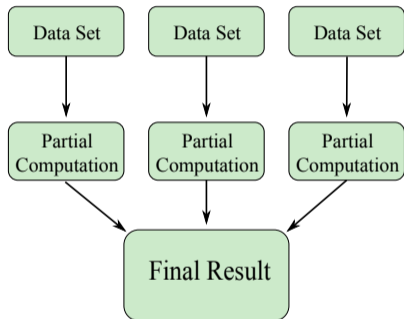
Training & prediction

- Regression
 - Linear/Ridge Regression
- Classification
 - Naive Bayes Classifier
 - Boosting
 - SVM
 - Neural Networks
 - Multi-Class Classifier

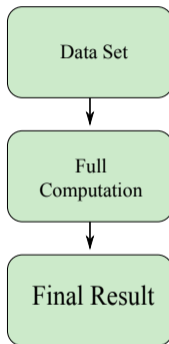
Portal: [DAAL page](#). See also: [intro article](#), [CR papers](#).

ALGORITHMS IN DAAL

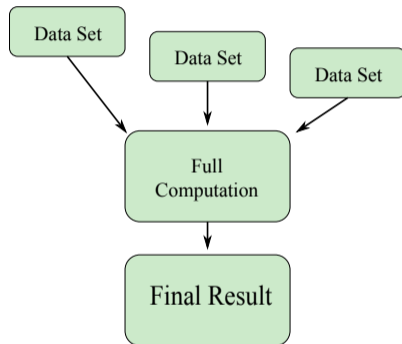
Distributed Mode



Batch Mode



Online Mode



Portal: [DAAL page](#). See also: [intro article](#), [CR papers](#).



COMMUNICATION FRAMEWORK: MPI

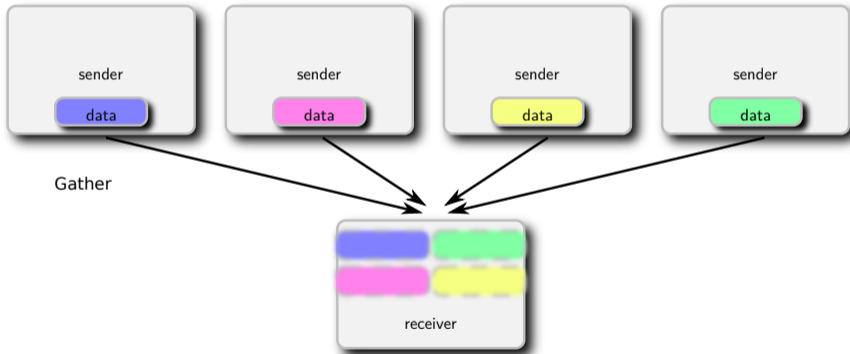
STRUCTURE OF MPI APPLICATIONS: HELLO WORLD

```
1 #include "mpi.h"
2 #include <stdio>
3 int main (int argc, char *argv[]) {
4     MPI_Init (&argc, &argv); // Initialize MPI environment
5     int rank, size, namelen;
6     char name[MPI_MAX_PROCESSOR_NAME];
7     MPI_Comm_rank (MPI_COMM_WORLD, &rank); // ID of current process
8     MPI_Get_processor_name (name, &namelen); // Hostname of node
9     MPI_Comm_size (MPI_COMM_WORLD, &size); // Number of processes
10    printf ("Hello World from rank %d running on %s!\n", rank, name);
11    if (rank == 0) printf("MPI World size = %d processes\n", size);
12    MPI_Finalize (); // Terminate MPI environment
13 }
```

MPICH site contains a list of **MPI 3.2 routines**

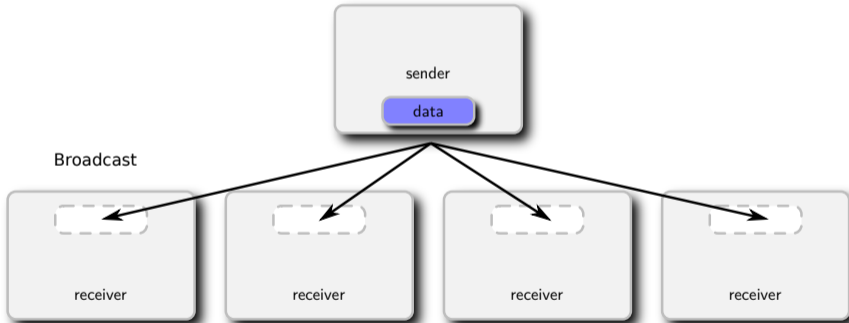
COLLECTIVE COMMUNICATION: GATHER

```
1 int MPI_Gather(void *sendbuf, int sendcnt, MPI_Datatype sendtype,  
2 void *recvbuf, int recvcnt, MPI_Datatype recvtype, int root, MPI_Comm comm);
```



COLLECTIVE COMMUNICATION: BROADCAST

```
1 int MPI_Bcast( void *buffer, int count, MPI_Datatype datatype,  
2 int root, MPI_Comm comm );
```





IMPLEMENTATION

EXAMPLE DISTRIBUTED IMAGE PROCESSING: DAAL

- ▶ Algorithm <step1Local> is responsible for the forward/backward propagation.

```

1 training::Distributed<step1Local> local_net;           // local net algorithm
2 local_net.compute();                                 // forward/backward
3 part_res = local_net.getPartialResult();             // getting partial result
4 local_net.input.get(training::inputModel)
5                                     ->setWeightsAndBiases(wb); // Update the weights/bias

```

- ▶ Algorithm <step2Master> is responsible for accumulating the gradient.

```

1 training::Distributed<step2Master> master_net;       // master net algorithm
2 master_net.input.add(training::partialResults,      // Add partial result
3                       0, part_res);
4 master_net.compute();                               // Accumulate gradients
5 wbModel = master_net.getPartialResult()             // Get Current Model
6           ->get(training::resultFromMaster)
7           ->get(training::model);
8 wb = wbModel->getWeightsAndBiases();                // Extract weights/bias

```

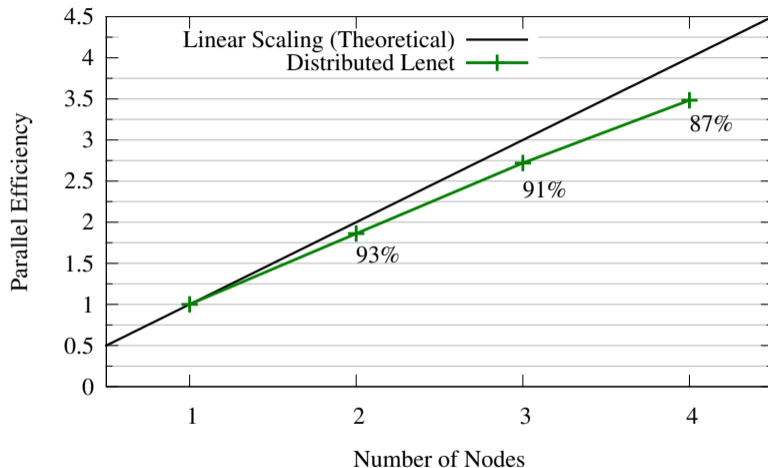
EXAMPLE DISTRIBUTED IMAGE PROCESSING (PART 1)

```
1 // Computation part of the node with the master net
2 // Local forward and backward propagation
3 local_net.compute();
4 part_res[master_node_id] = local_net.getPartialResult();
5
6 // ... Code to store the result into a buffer (char *) ... //
7
8 // Send the result to the master node
9 MPI_Gather(...);
10
11 // ... Code to reconstruct the partial result from the buffer... //
12
13 // accumulate the partial result from nodes
14 for(int i = 0; i < num_nodes; i++)
15     master_net.input.add(training::partialResults, node, part_res[i]);
16 master_net.compute();
```

EXAMPLE DISTRIBUTED IMAGE PROCESSING (PART 2)

```
1 // ... Continuing on the master compute ... //
2
3 // Extract the weight/bias from the master net
4 training::ModelPtr wbModel = master_net.getPartialResult()
5     ->get(training::resultFromMaster)
6     ->get(training::model);
7 NumericTablePtr wb = wbModel->getWeightsAndBiases();
8
9 // ... Code to store weights/bias into a buffer (char*) ... //
10
11 // Broadcast the weights/bias to all nodes //
12 MPI_Bcast(...);
13
14 // ... Code to reconstruct the weights/bias from buffer ... //
15
16 // Update the weights on local node
17 local_net.input.get(training::inputModel)->setWeightsAndBiases(wb);
```

PARALLEL EFFICIENCY



Further performance optimizations and model parallelism are coming soon...



§7. FINAL WORDS

COLFAX RESEARCH
CONTRIBUTING TO INNOVATIONS IN COMPUTING
Log In/Out or Register

READ WATCH LEARN CONNECT JOIN



To search, type and hit enter

Popular

The Hands-On Tutorials (HOT) webinars: details on efficient programming for Intel architecture

The Hands-On Workshop (HOW) Series

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation

Parallel Programming Book

Introduction to parallel programming, deep discussion of optimization techniques, exercises.

© 2015, Colfax International, 508 pages.

Featured Video

See Research material on vectorization in a training video

Optimization Techniques for the Intel MIC Architecture, Part 1 of 3: Multi-Threading and Parallel Reduction

Performance to Cost Ratios with Intel Xeon Phi Coprocessors (and why ix Acceleration May Be Enough)

Research and Educational Publications

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation

Optimization Techniques for the Intel MIC Architecture, Part 3 of 3: False Sharing and Padding

Software Developer's Introduction to the HGST Ultrastar Archive H700 SMR Drives

Optimization Techniques for the Intel MIC Architecture, Part 2 of 3: Strip-Mining for Vectorization



Optimization Techniques for the Intel MIC Architecture, Part 2 of 3: Strip-Mining for Vectorization

Parallel Programming Book

Optimization Techniques for the Intel MIC Architecture, Part 2 of 3: Strip-Mining for Vectorization

Performance to Cost Ratios with Intel Xeon Phi Coprocessors (and why ix Acceleration May Be Enough)

Consulting


Colfax offers consulting services for enterprises, research help you to:

- Optimize your existing application to take advantage of parallelism, from vectors to cores to clusters and
- Future-proof your application for upcoming innovations
- Accelerate your application using coprocessor tech
- Investigate the potential system configurations that satisfy your cost, power, performance requirements.
- Take a clean slate to develop a novel architecture to realize your computing pro

Episode 2.1 — Purpose of the MIC architecture

Parallel Computing in the Search for New Physics at LHC

Software Developer's Introduction to the HGST Ultrastar Archive H700 SMR Drives




Colfax offers consulting services for enterprises, research help you to:

- Optimize your existing application to take advantage of parallelism, from vectors to cores to clusters and
- Future-proof your application for upcoming innovations
- Accelerate your application using coprocessor tech
- Investigate the potential system configurations that satisfy your cost, power, performance requirements.
- Take a clean slate to develop a novel architecture to realize your computing pro

Parallel Computing in the Search for New Physics at LHC

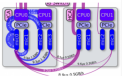
Fluid Dynamics with Fortran on Intel Xeon Phi coprocessors




Colfax offers consulting services for enterprises, research help you to:

- Optimize your existing application to take advantage of parallelism, from vectors to cores to clusters and
- Future-proof your application for upcoming innovations
- Accelerate your application using coprocessor tech
- Investigate the potential system configurations that satisfy your cost, power, performance requirements.
- Take a clean slate to develop a novel architecture to realize your computing pro

Configuration and Benchmarks of Peer-to-Peer Communication over Gigabit Ethernet and InfiniBand in a Cluster with Intel Xeon Phi Coprocessors



Interview with James Reinders: future of Intel MIC architecture, parallel programming, education



http://colfaxresearch.com/

Thank you for your Attention!

