Codesigning Cognitive Computing Systems and Applications

Wen-mei Hwu, Co-Director with Jinjun Xiong (IBM)

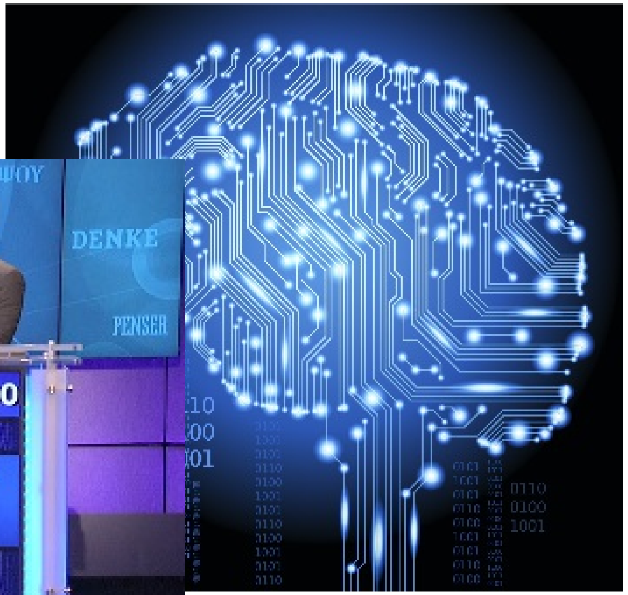
on behalf of the entire C^3SR team

University of Illinois at Urbana-Champaign

November 2, 2017

Cognitive Computing – the C3SR View

- A cognitive computing application fuses vast, **unstructured data** and vast **human knowledge base** to **extend human capabilities** by solving problems, making actionable recommendations, and producing customized learning experiences



C3SR Vision

- The rise of cognitive computing has created new opportunities to rethink all the three layers of computing systems–applications, software, and hardware.

- Dramatic enhancement in the **efficacy**, **efficiency** and **variety** of cognitive computing applications can be achieved through dramatic enhancement in the **programmability**, **throughput**, **latency**, **capacity**, and **affinity** of computing systems.

C^{3}SR faculties & students (Est. 9/2016)

Suma Bhat | Minh Do | Deming Chen | Julia Hockenmaier | Wen-mei Hwu | Nam Sung Kim | Dan Roth | Rakesh Nagi | Lav Varshney

…

The Three Pillars of C3SR:

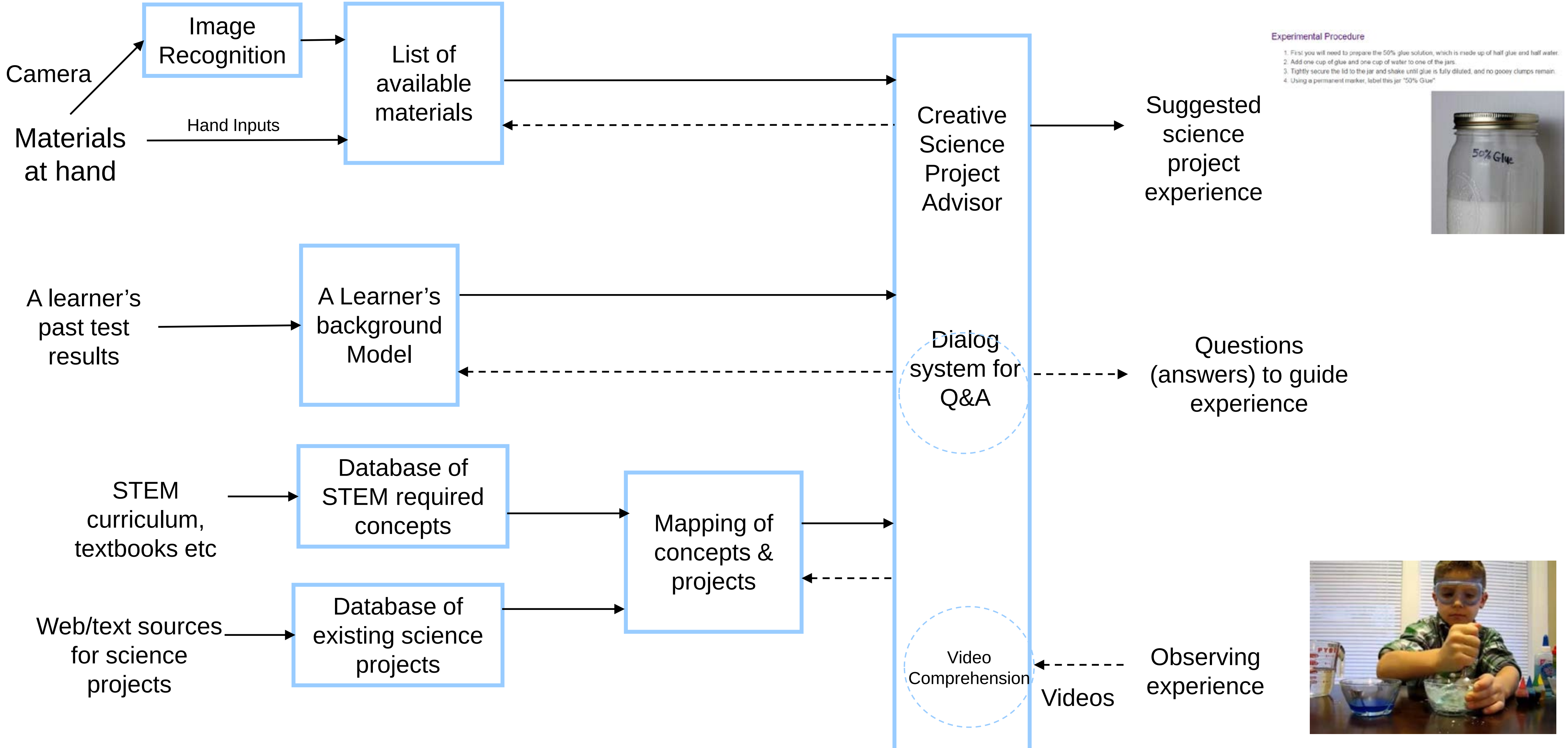
- Creative experiential learning advisor (CELA) as a grand challenge use case for cognitive capabilities
- Cognitive application builder (CAB) to make the underlying heterogeneous infrastructure easy to consume for cognitive application developers
- Cognitive systems innovations (Erudite) for workload acceleration, including Near Memory Acceleration (NMA)

A New Modality of Application Development

- Cognitive applications demand functionalities that we have failed to program
  - Computer vision, natural language dialogs, stock trading, fraud detection, …
- Use labeled data – data that come with the input values and their desired output values – to learn what the logic should be
  - Capture each labeled data item by adjusting the program logic
  - Learn by example!
- This introduces a new modality of application development
  - Training, Testing, Integration, Profiling, Debugging, etc.

Application Driver

CELA: personalized education via multi-modality data comprehension and computational creativity



Decomposition of CELA's Research Challenges

CELA

Integration (CAB)

Video/Image Understanding	Knowledge Representation	Concept Extraction	Formal Assessment	Computational Creativity	NLP/Dialog Systems
Do, Yeh, Mac, Feris Marcheret, Ratha	Varshney, Bhat, Yu Gong, Gliozzo	Lav, Bhat, Yu Gong, Sakakini	Lav, Bhat Scholoss	Varshney, Yu, Ge Bhattacharjya	Roth, Chaturvedi CHN, IBM Bluemix

Acceleration (CAB, NMA)

OpenPOWER™

OpenCAPI™ **CUDA**

Heterogeneous Infrastructure

- Requires a tool to integrate core services that are optimized for the underlying heterogeneous infrastructure

Cognitive Application Builder

A system-level challenge

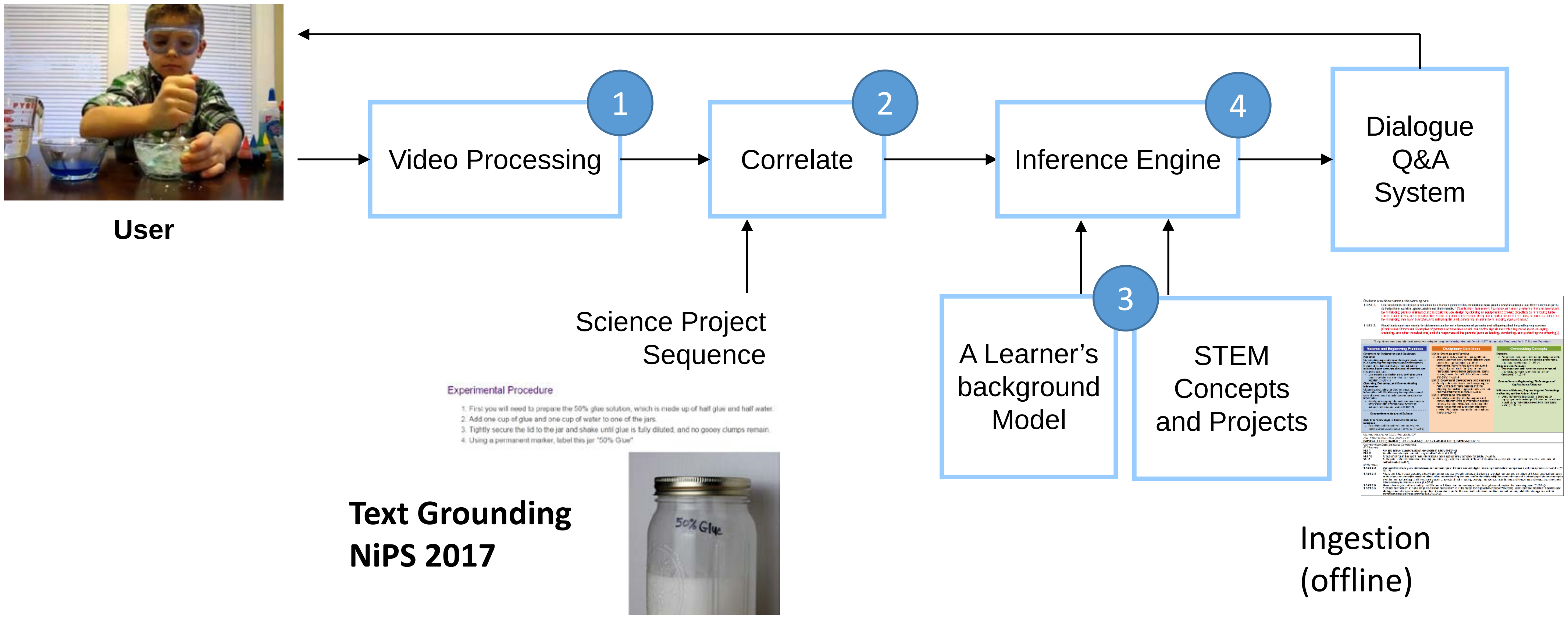
Workflow description
Innovative AI techniques,
Data, Models, Frameworks

→

High-performance, scalable,
robust applications

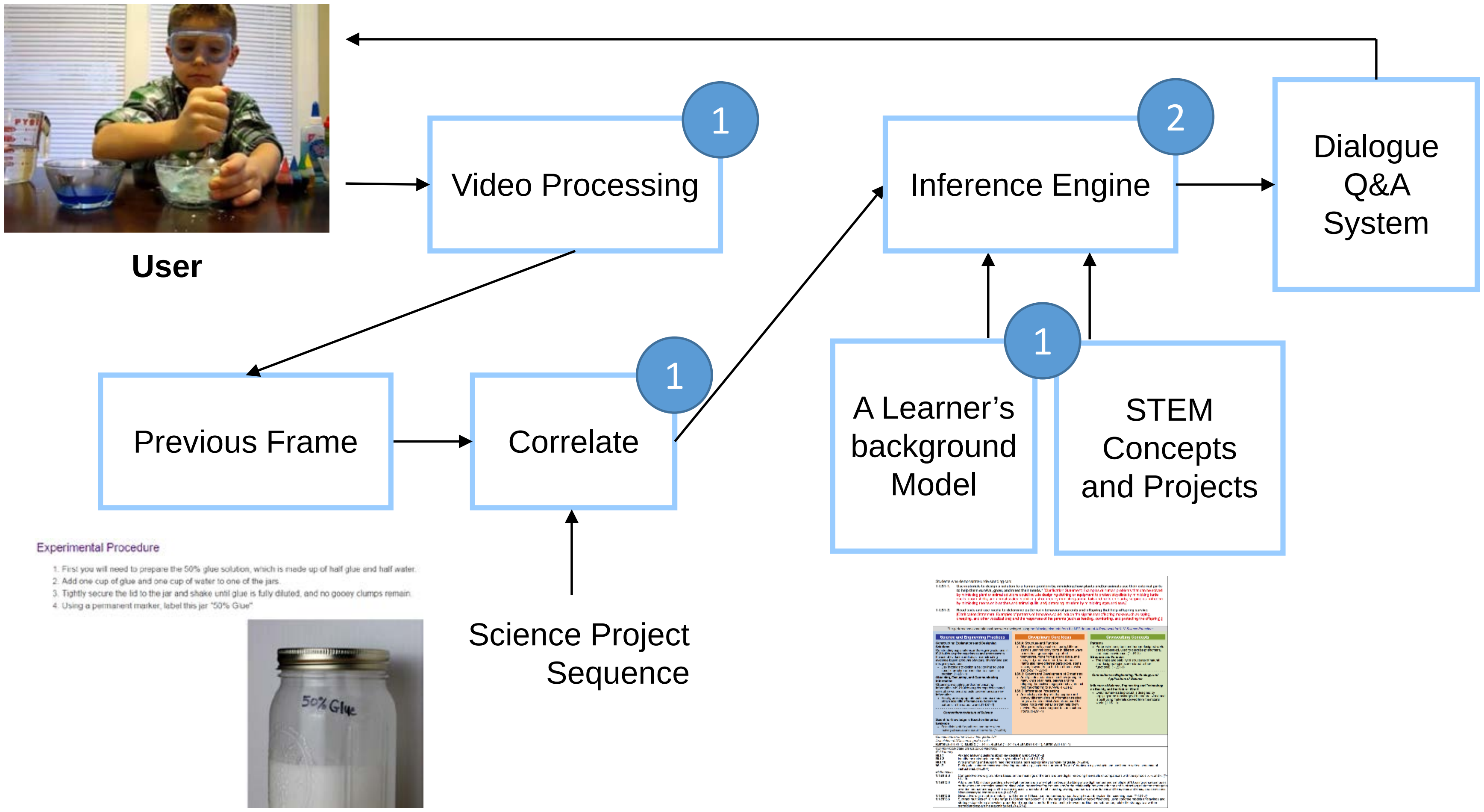
- CAB: A language, compiler, and runtime for easy development of cognitive applications
  - Software synthesis to exploit accelerators and efficient communication
  - Introspection for debugging and performance evaluation
  - Workflow profiling, optimization and orchestration for system-level performance
  - Decentralized application architecture for scalability, composability, testing, and development

CELA as a Driving Use Case for CAB



CAB simplifies component connection, workflow description, model training/selection, and iterative development

CELA as a Driving Use Case for CAB

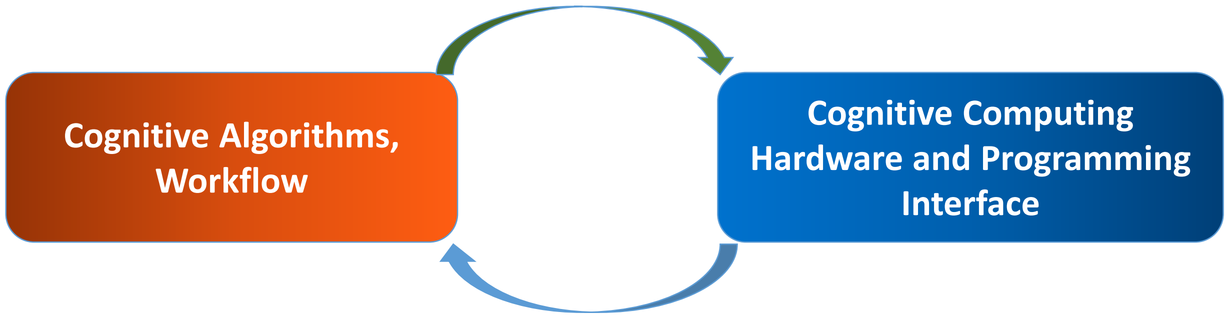


- CAB will automatically transform workflows for high-performance execution

C3SR Approach to Cognitive Computing System Design

- To develop scalable cognitive applications by co-designing
  - advanced methods and algorithms for cognitive computation, and
  - optimized heterogeneous computing systems for these workloads.

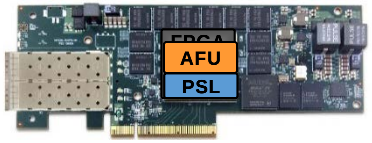
- Generations of complete prototype systems
  - **Initial** – existing methods, algorithms and workflows running on existing hardware
  - **Refined** – innovative methods, algorithms and workflows enabled by the next generation memory/storage technology and accelerators
  - **Novel** – ambitious methods, algorithms and workflows empowered by new memory and near memory/near IO acceleration technologies.



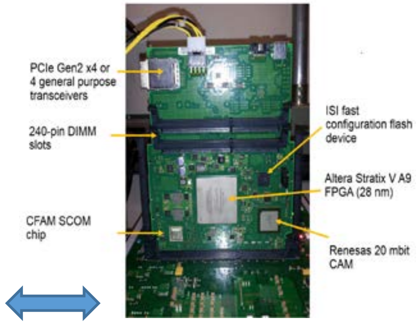
Initial Experimental Heterogeneous Infrastructure

DGX-1

2 x P8 Minsky with NVLink GPUs



FPGA CAPI
over PCIe



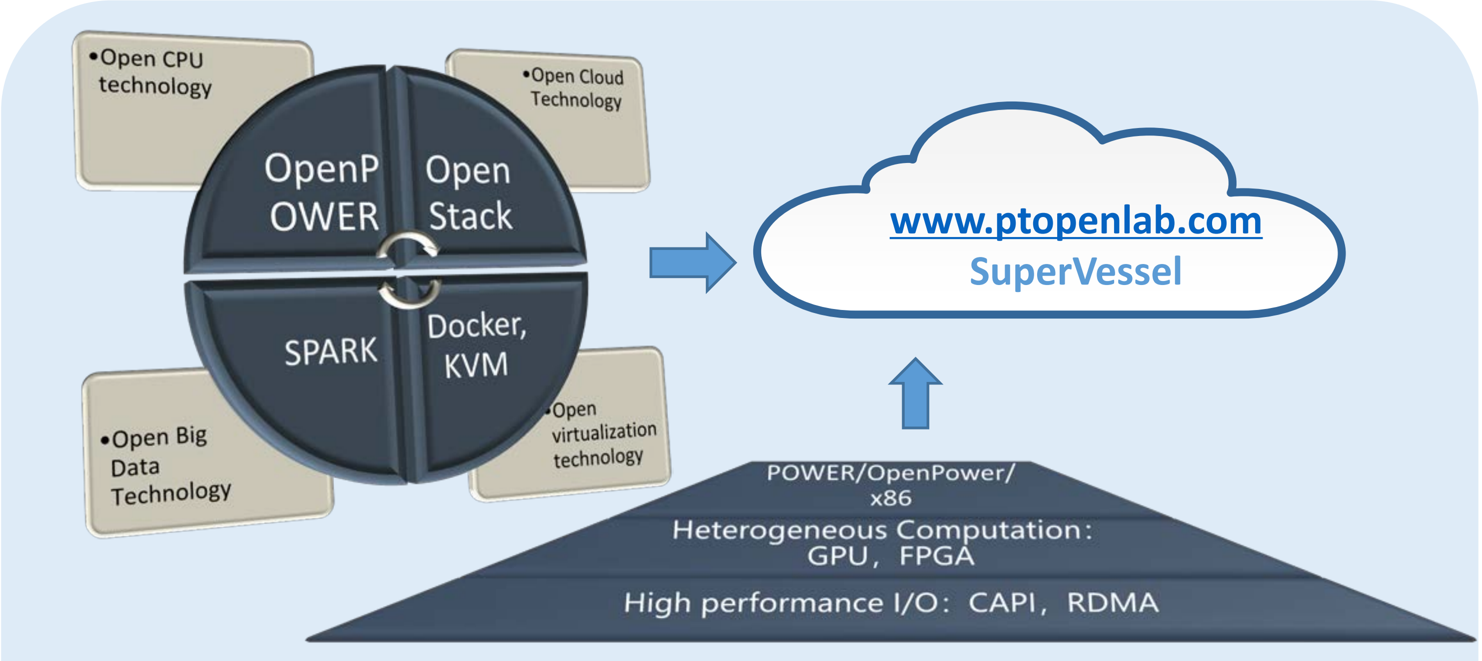
ConTutto over DMI

4 x P8 Tuleta (S824L)

IBM Bluemix™

IBM SOFTLAYER®

Watson developer cloud



Great support from Brad McCredie

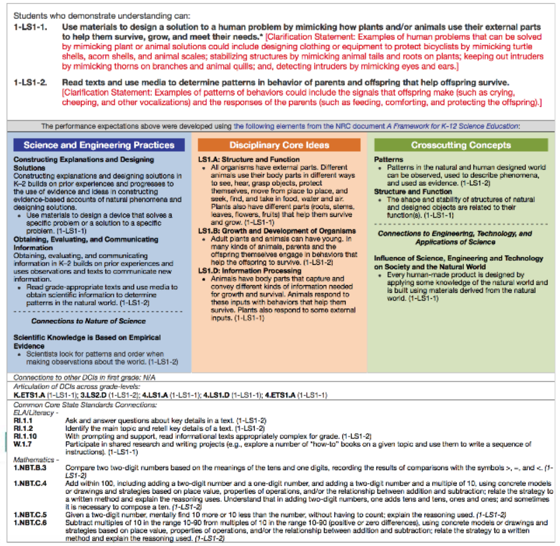
- A dedicated program manager and team (Ben Kreuz, JT Kellingon, Adam McPadden, Dean Sciacca, Jonanthan Dement)

Selected center progress highlights

- Curated datasets
- The CarML system for model development and deployment
- Workload acceleration
- The Erudite NMA system

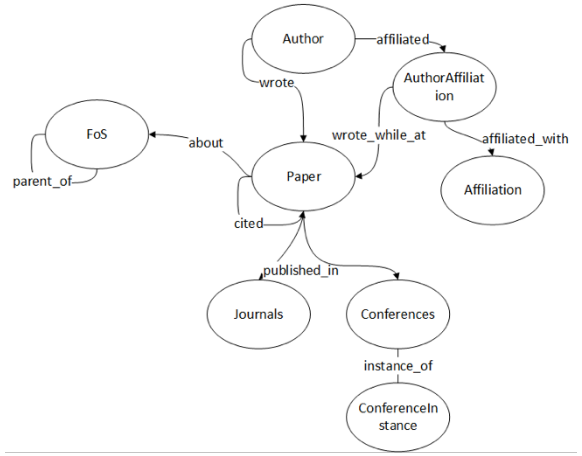
Curated Datasets

- Extracted STEM concept dependency from next generation science standard that includes
  - Performance Expectations
  - Science and Engineering Practices
  - Disciplinary Core Ideas
  - Crosscutting Concepts
  - Connections



- Extracted science projects from websites and stored as a structured data
  - Extracted all 1188 projects from ScienceBuddies.com

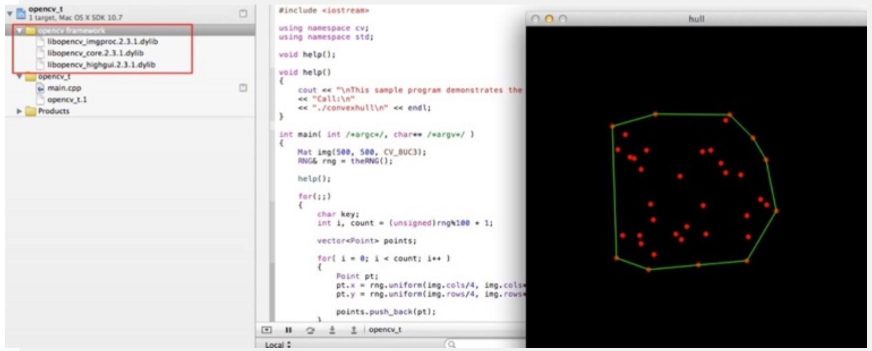
- Extracted DBLP bibliographic database for computer science and MICRO 50 years of publications (~1400)
  - All stored in a graph database (~100G) with a structure similar to the Microsoft Academic Graph



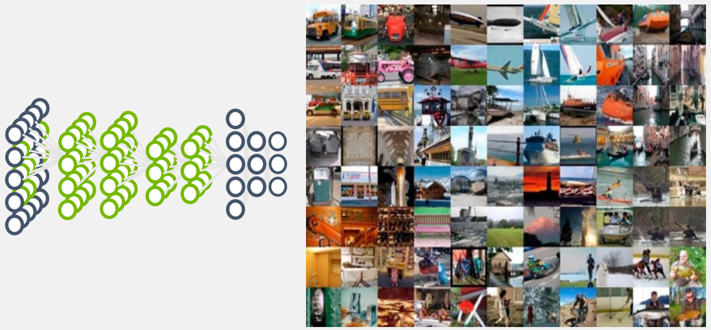
Selected center progress highlights

- Curated datasets
- **The CarML system for model development and deployment**
- Workload acceleration
- The Erudite NMA system

Deep Learning Revolution
- a humble beginning in 2010

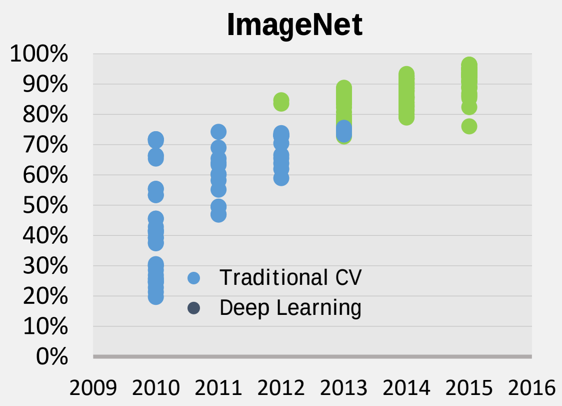


Traditional Computer Vision
Experts + Time



Deep Learning Object Detection
DNN + Data + HPC

2M training images



Deep Learning Achieves
“Superhuman” Results

Slide courtesy of Steve Oberlin, NVIDIA

CarML – Cognitive Artifacts for Machine Learning

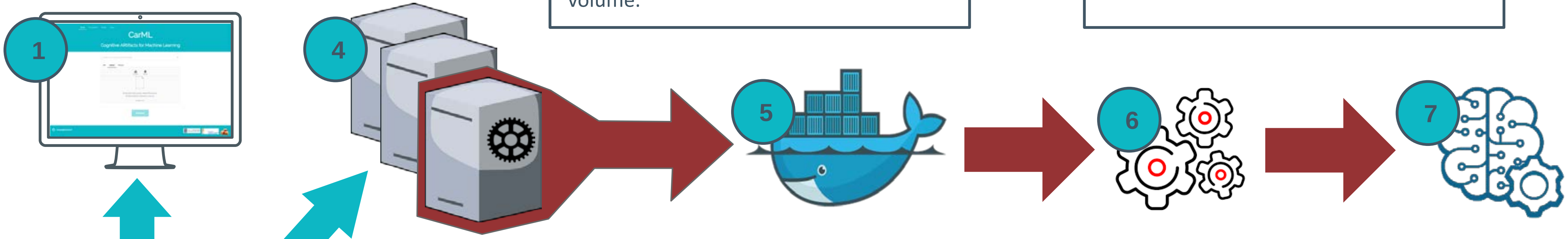
- CarML.org
- An open source distributed platform to easily deploy and benchmark machine learning frameworks and models **across hardware architectures**, through a common interface.
  - An experimentation platform for ML users
  - A deployment platform for ML developers
  - A benchmarking platform for systems architects

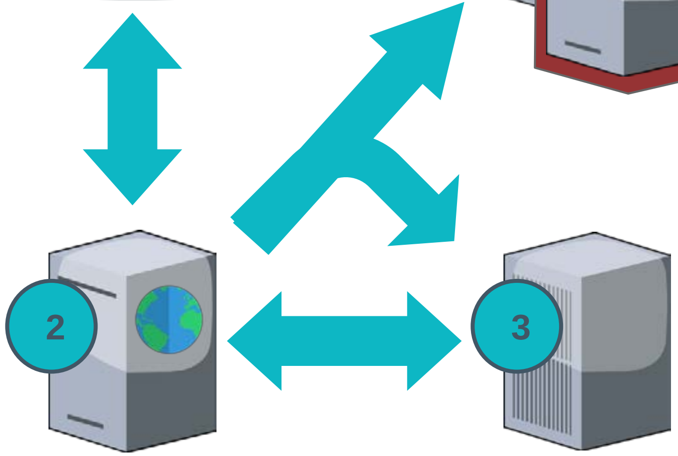
CarML.org as a Web Service

1 A use selects models and inputs through web UI or API.

5 The agent starts a docker container for the request. Data that is shared across executions are mounted as a shared volume.

6 Within the docker container, the model is downloaded, loaded into memory, and the user's inputs are preprocessed.





4 On bootup, agents advertise the models in the registry. The web server forwards the user request to the agents capable of evaluating the model

7 Inference is performed within the docker container and the results are sent back to the user.

2 The web server accepts the user inputs and interacts with registry, tracing, and agents

3 The web server queries the agent registry to retrieve the address. The tracing process is started, if profiling is enabled.

Model Catalog

- Repository contains more than 100 DL models
- Support for Tensorflow, Caffe, Caffe2, and MXNet
  - PyTorch, CNTK, Paddle, … planned
- Versioned models and frameworks
  - Allows to experiment with custom DL layers

Dataset Catalog

- Repository contains common DL datasets
  - CIFAR 10/100
  - MNIST
  - ImageNet
  - ….
- Allows one to compare DL models on validation datasets

Machine Catalog

- X86 and Power8 Systems
    - CPU only mode and/or GPU mode
- Planned to have ARM cores and integration with simulators

Tracing and Monitoring Options

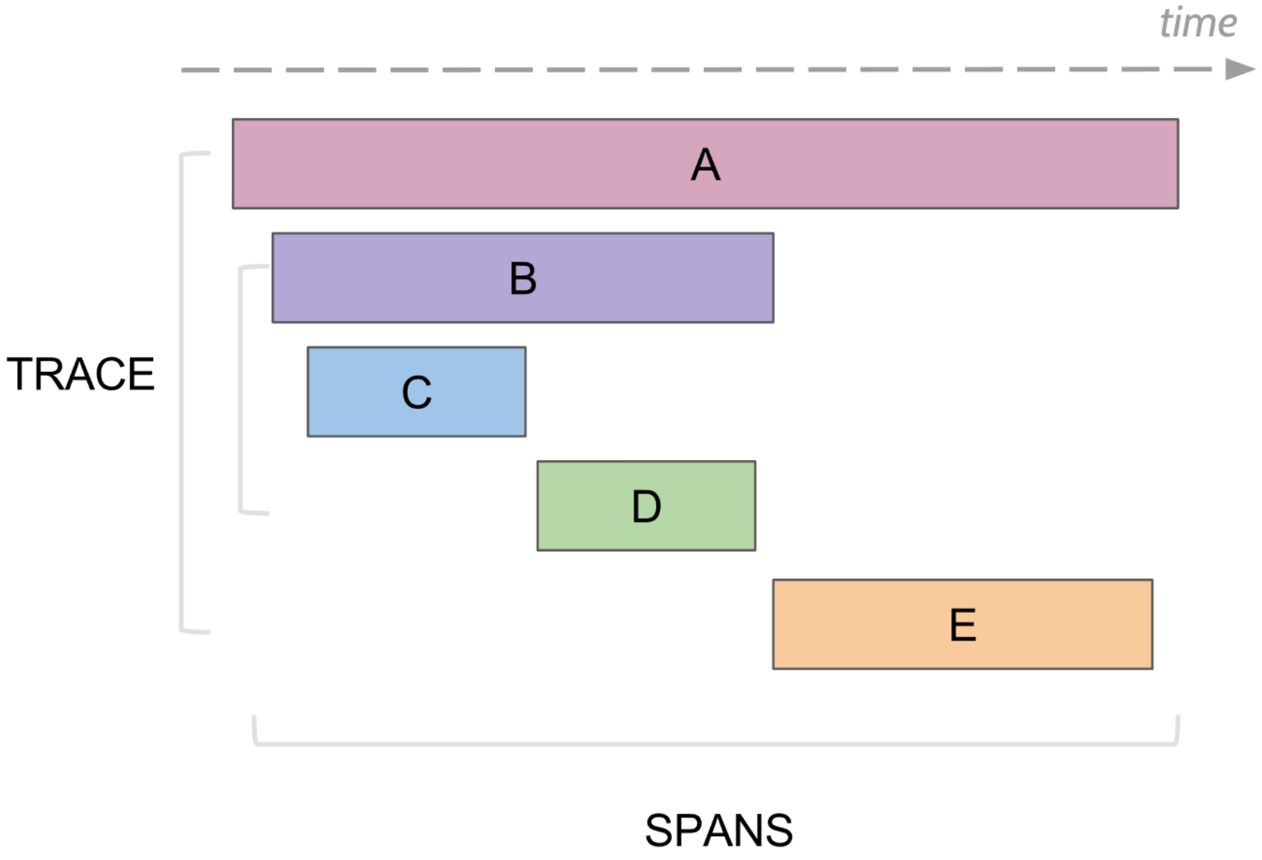
- Integration with PAPI
- Integration with Perf Events
- Integration with NVIDIA's CUPTI
- Integration with OSX's Instruments

Tracing

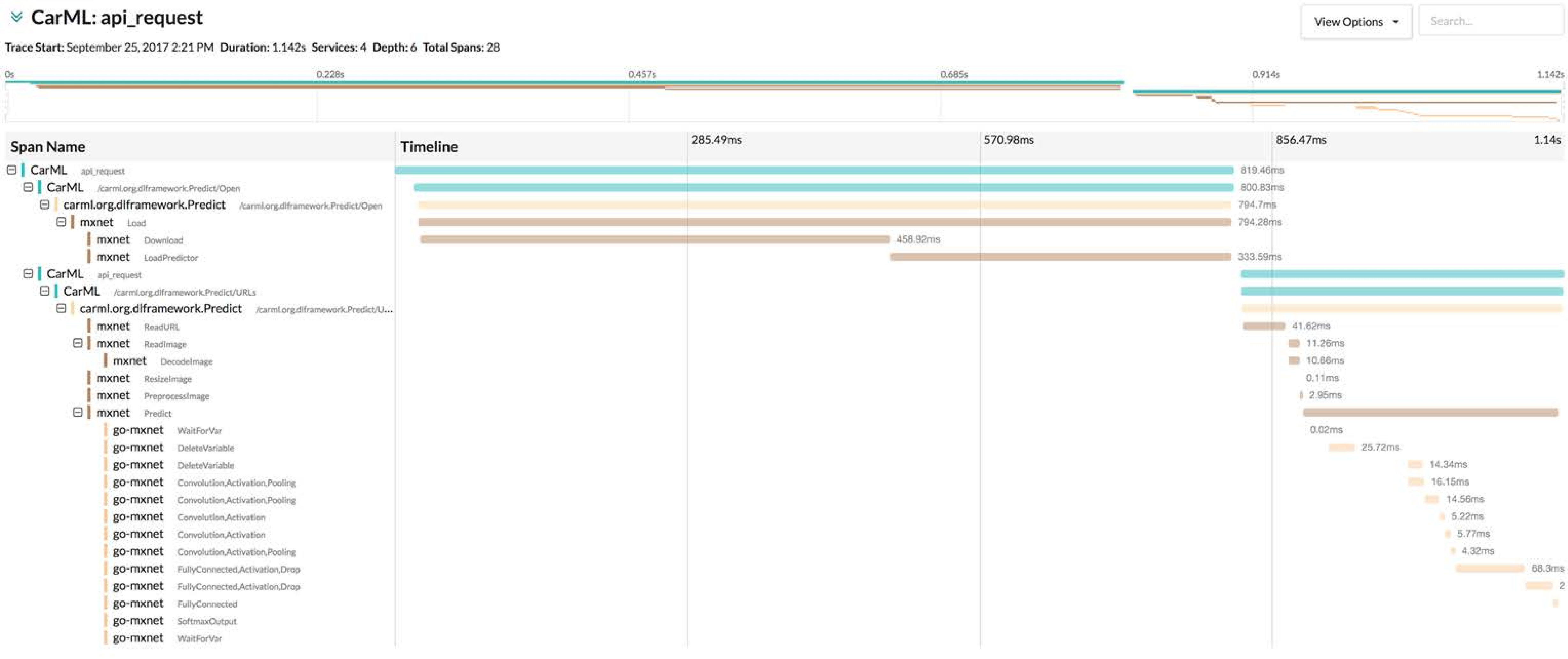
Terminology

A **Trace** is a directed acyclic graph (DAG) of **Spans**

Spans can reference one another.



Tracing


CarML: api_request
View Options
Search...
Trace Start: September 25, 2017 2:21 PM Duration: 1.142s Services: 4 Depth: 6 Total Spans: 28
0s
0.228s
0.457s
0.685s
0.914s
1.142s
Span Name
Timeline
285.49ms
570.98ms
856.47ms
1.14s
CarML api_request
CarML /carml.org.dlframework.Predict/Open
carml.org.dlframework.Predict /carml.org.dlframework.Predict/Open
mxnet Load
mxnet Download
mxnet LoadPredictor
CarML api_request
CarML /carml.org.dlframework.Predict/URLs
carml.org.dlframework.Predict /carml.org.dlframework.Predict/U...
mxnet ReadURL
mxnet ReadImage
mxnet DecodeImage
mxnet ResizeImage
mxnet PreprocessImage
mxnet Predict
go-mxnet WaitForVar
go-mxnet DeleteVariable
go-mxnet DeleteVariable
go-mxnet Convolution,Activation,Pooling
go-mxnet Convolution,Activation,Pooling
go-mxnet Convolution,Activation
go-mxnet Convolution,Activation
go-mxnet Convolution,Activation,Pooling
go-mxnet FullyConnected,Activation,Drop
go-mxnet FullyConnected,Activation,Drop
go-mxnet FullyConnected
go-mxnet SoftmaxOutput
go-mxnet WaitForVar
819.46ms
800.83ms
794.7ms
794.28ms
458.92ms
333.59ms
41.62ms
11.26ms
10.66ms
0.11ms
2.95ms
0.02ms
25.72ms
14.34ms
16.15ms
14.56ms
5.22ms
5.77ms
4.32ms
68.3ms

Observers

- Subscribe on StartSpan / EndSpan events
- Capture hardware counters for each event
  - PAPI
  - NVML
  - Perf

CUPTI

- Capture CUDA runtime & driver events
- Integrated with the CarML tracer
  - Implemented in Go
  - Declare CUPTI callback function in Go
  - Pass CUPI Go handle into C code
  - Events to capture are configurable

CUPTI

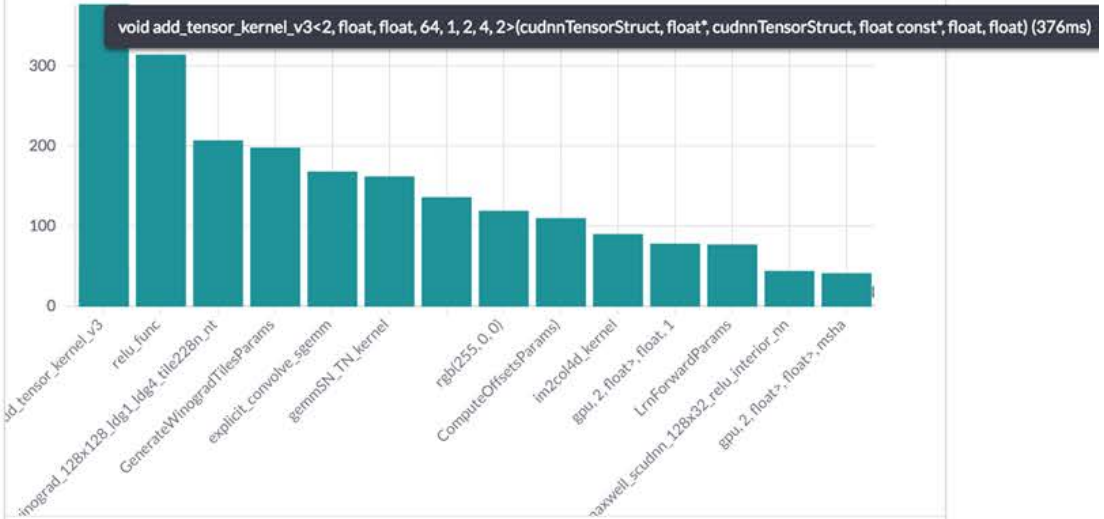
```go
func callback(userData unsafe.Pointer, domain0 C.CUpti_CallbackDomain, cbid0 C.
  handle := (*CUPTI)(unsafe.Pointer(userData))
  if handle == nil {
    log.Debug("expecting a cupti handle, but got nil")
    return
  }
  domain := types.CUpti_CallbackDomain(domain0)
  switch domain {
  case types.CUPTI_CB_DOMAIN_DRIVER_API:
    cbid := types.CUpti_driver_api_trace_cbid(cbid0)
    switch cbid {
    case types.CUPTI_DRIVER_TRACE_CBID_cuLaunchKernel:
      handle.onCULaunchKernel(domain, cbid, cbInfo)
      return
    case types.CUPTI_DRIVER_TRACE_CBID_cuMemcpyHtoD_v2,
      types.CUPTI_DRIVER_TRACE_CBID_cuMemcpyDtoH_v2,
      types.CUPTI_DRIVER_TRACE_CBID_cuMemcpyDtoD_v2,
      types.CUPTI_DRIVER_TRACE_CBID_cuMemcpyHtoDAsync_v2,
      types.CUPTI_DRIVER_TRACE_CBID_cuMemcpyDtoHAsync_v2,
      types.CUPTI_DRIVER_TRACE_CBID_cuMemcpyDtoDAsync_v2:
      handle.onCudaMemCopyDevice(domain, cbid, cbInfo)
      return
    default:
      log.WithField("cbid", cbid.String()).
        WithField("function_name", demangleName(cbInfo.functionName)).
        Debug("skipping runtime call")
      return
    }
  case types.CUPTI_CB_DOMAIN_RUNTIME_API:
    cbid := types.CUPTI_RUNTIME_TRACE_CBID(cbid0)
    switch cbid {
    case types.CUPTI_RUNTIME_TRACE_CBID_cudaDeviceSynchronize_v3020,
      types.CUPTI_RUNTIME_TRACE_CBID_cudaStreamSynchronize_v3020:
      handle.onCudaDeviceSynchronize(domain, cbid, cbInfo)
      return
    case types.CUPTI_RUNTIME_TRACE_CBID_cudaMemcpy_v3020,
      types.CUPTI_RUNTIME_TRACE_CBID_cudaMemcpyAsync_v3020:
      handle.onCudaMemCopy(domain, cbid, cbInfo)
      return
    case types.CUPTI_RUNTIME_TRACE_CBID_cudaLaunch_v3020:
      handle.onCudaLaunch(domain, cbid, cbInfo)
      return
    case types.CUPTI_RUNTIME_TRACE_CBID_cudaThreadSynchronize_v3020:
      handle.onCudaSynchronize(domain, cbid, cbInfo)
      return
    case types.CUPTI_RUNTIME_TRACE_CBID_cudaConfigureCall_v3020:
      handle.onCudaConfigureCall(domain, cbid, cbInfo)
      return
```

Current Work

Filter files

File	Size	Last Modified
caffe2_out		
mxnet_out		
BVLC-AlexNet-v1.0		
mxnet_0.11.0_BVLC-AlexNet_1.0_1	162 kB	3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_2	169 kB	3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_4	182 kB	3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_8	189 kB	3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_16	148 kB	3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_32	259 kB	3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_64	320 kB	3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_128	455 kB	3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_256	700 kB	3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_512	1.16 MB	3 hours ago
BVLC-GoogLeNet-v1.0		
mxnet_0.11.0_BVLC-GoogLeNet_1.0_1	851 kB	3 hours ago
mxnet_0.11.0_BVLC-GoogLeNet_1.0_2	860 kB	3 hours ago
mxnet_0.11.0_BVLC-GoogLeNet_1.0_4	864 kB	3 hours ago
mxnet_0.11.0_BVLC-	893	3 hours

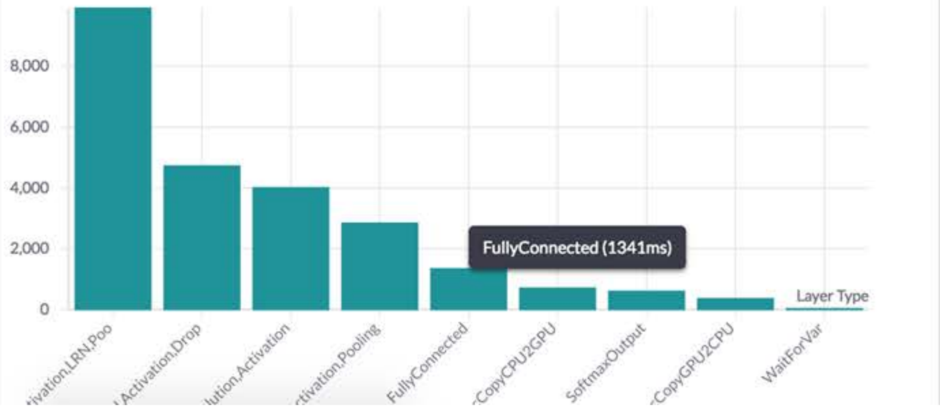
mxnet/BVLC-AlexNet::2

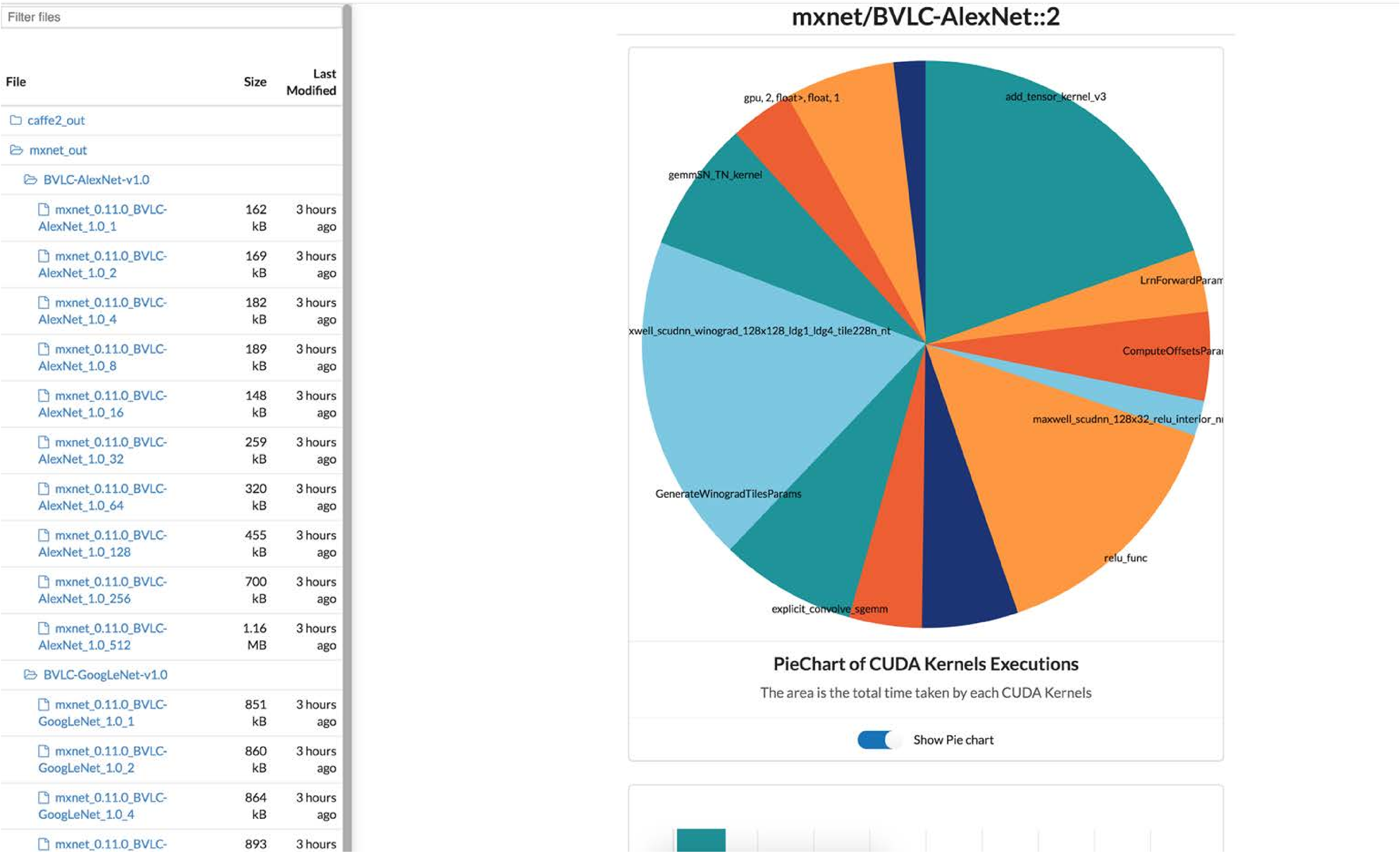


Histogram of CUDA Kernels Executions

The y-axis is the total time for each CUDA Kernels

Show Pie chart





Filter files
File
Size
Last Modified
caffe2_out
mxnet_out
BVLC-AlexNet-v1.0
mxnet_0.11.0_BVLC-AlexNet_1.0_1 162 kB 3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_2 169 kB 3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_4 182 kB 3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_8 189 kB 3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_16 148 kB 3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_32 259 kB 3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_64 320 kB 3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_128 455 kB 3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_256 700 kB 3 hours ago
mxnet_0.11.0_BVLC-AlexNet_1.0_512 1.16 MB 3 hours ago
BVLC-GoogLeNet-v1.0
mxnet_0.11.0_BVLC-GoogLeNet_1.0_1 851 kB 3 hours ago
mxnet_0.11.0_BVLC-GoogLeNet_1.0_2 860 kB 3 hours ago
mxnet_0.11.0_BVLC-GoogLeNet_1.0_4 864 kB 3 hours ago
mxnet_0.11.0_BVLC- 893 3 hours
mxnet/BVLC-AlexNet::2
gpu, 2, float>, float, 1
add_tensor_kernel_v3
gemmSN_TN_kernel
LrnForwardParam
xwell_scudnn_winograd_128x128_ldg1_ldg4_tile228n_nt
ComputeOffsetsPara
maxwell_scudnn_128x32_relu_interior_n
GenerateWinogradTilesParams
relu_func
explicit_convolve_sgemm
PieChart of CUDA Kernels Executions
The area is the total time taken by each CUDA Kernels
Show Pie chart

Model Accuracy on different machines (CPU)

	mxnet-m	mxnet-l	caffe-m	caffe-l	caffe2-m	caffe2-l
BVLC-AlexNet	0.4268	0.6764	0.4268	0.6764	0.4268	0.6764
BVLC-GoogLeNet	0.9984	0.9991	0.9984	0.9991	0.9968	0.9991
SqueezeNet-v1.0	0.8834	0.8501	0.7999	0.7999	0.7999	0.9484
SqueezeNet-v1.1	Shower Cap(0.2874)	0.6929	0.9645	0.9645	0.9645	0.9108

m: minsky
l- mac
img: cheeseburger

Selected center progress highlights

- Curated datasets
- The CarML System for Model Development and Deployment
- **Workload acceleration**
- The Erudite NMA system

Workload acceleration research at C3SR

- Focus on impactful cognitive workloads for acceleration
  - Matrix factorization on GPU
  - Long-term Recurrent Convolutional Network acceleration
  - ResNet inference acceleration
  - Neuron Machine Translation acceleration
  - DNN inference acceleration
  - Graph analytic acceleration
- In discussion with other CHN centers to collect performance critical cognitive workloads
- Plan to deliver a set of cognitive benchmarks optimized for OpenPOWER

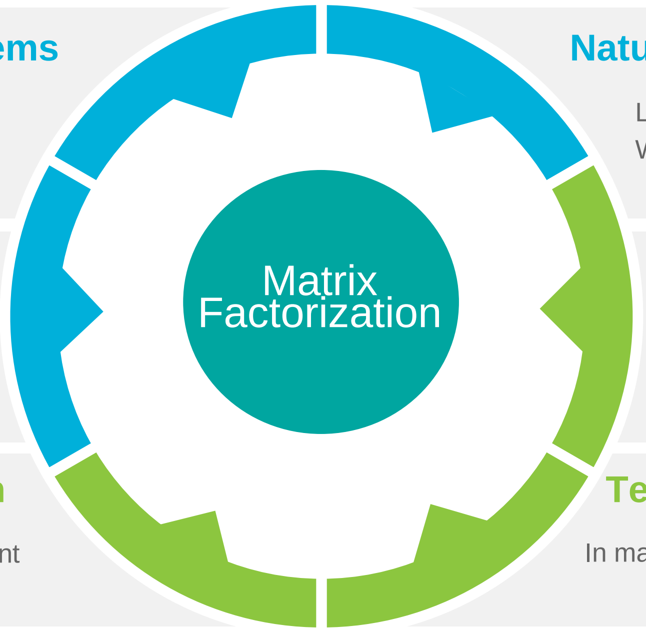
Matrix factorization: one of key workloads

Recommender systems

Predict missing ratings
Group similar users/items

Natural language processing

Latent semantic model
Word embedding as input to DNN



Complex network

Link prediction
Vertices clustering

Deep learning

Model compression
Embedding layer

Web search

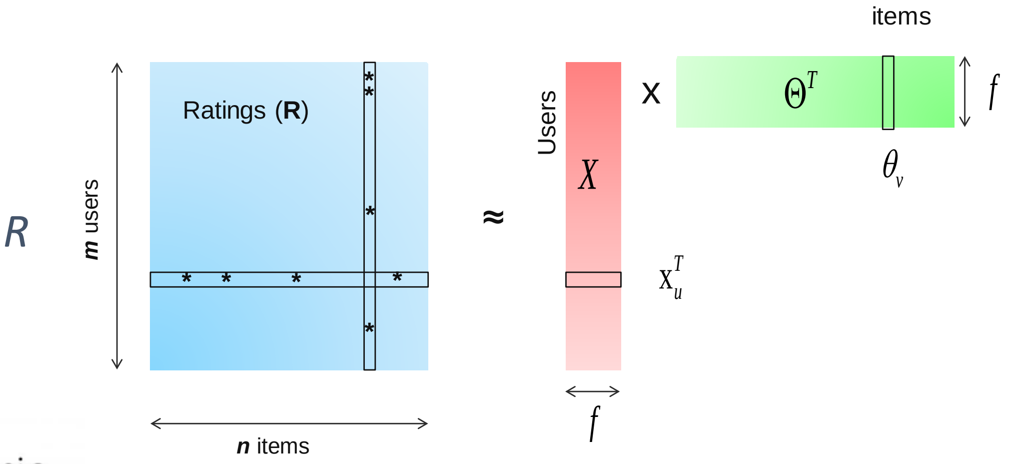
Match query and document

Tensor decomposition

In machine learning and HPC applications

2			4		5
	5	3		1	
		4		2	
1	3		3		4
			2		4
	1	3		5	

NETFLIX



cuMF acceleration

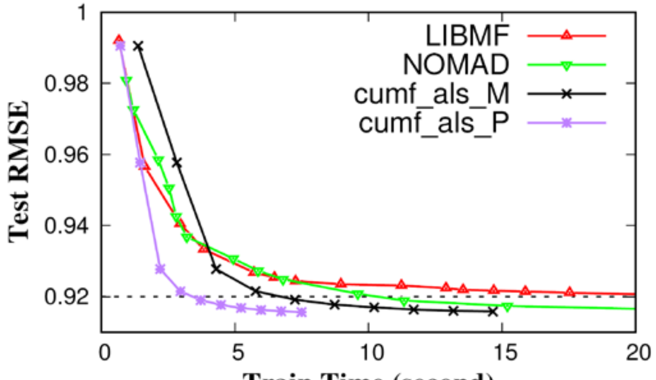
- cuMF formulation: factorize matrix R into

$$R \approx X \cdot \Theta^T$$

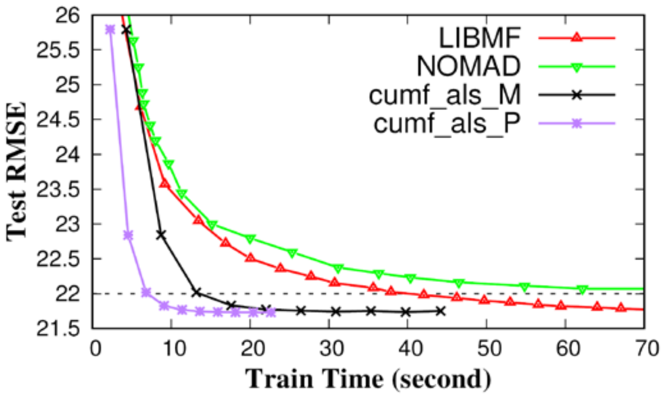
  - while minimizing the empirical lost

$$J = \sum_{u,v}(r_{uv} - \boldsymbol{x}_u^T \boldsymbol{\theta}_v)^2 + \lambda\left(\sum_u n_{x_u}||\boldsymbol{x}_u||^2 + \sum_v n_{\theta_v}||\boldsymbol{\theta}_v||^2\right)$$

- Connect cuMF to Spark MLlib via JNI
- cuMF_ALS @4 Maxwell ($2.5/hour)
  ≈ 10x speedup over SparkALS @50 nodes
  ≈ 1% of SparkALS's cost ($0.53/hour/node)
- Open source @ http://github.com/cuMF/
- **Demoed at SC'16 and GTC'16 on Minsky**
- **Presented to Jen-Hsun Huang on Feb 1, 2017**



Netflix



Yahoo!Music

- cuMF_ALS w/ FP16 on Maxwell and Pascal
- LIBMF: 1 CPU w/ 40 threads
- NOMAD
  - 32 nodes for Netflix and Yahoo
- 2-10x as fast

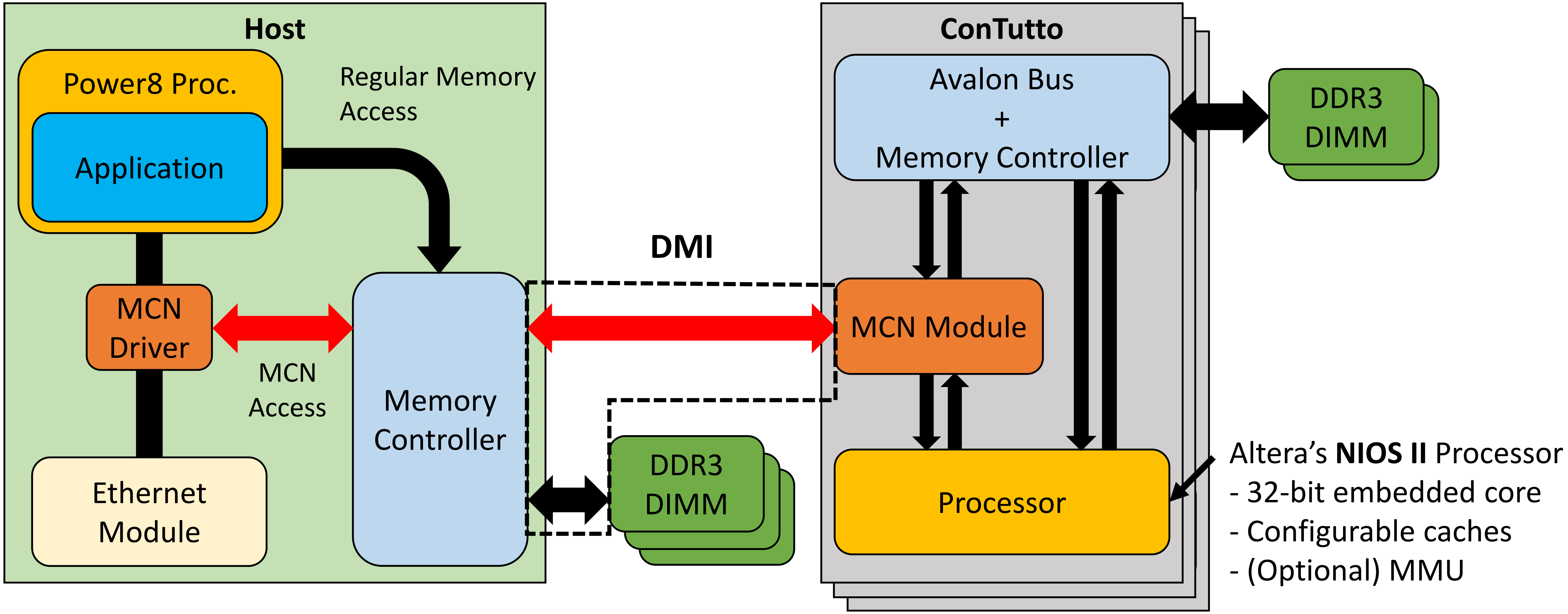
Selected center progress highlights

- Curated datasets
- The CarML System for Model Development and Deployment
- Workload acceleration
- **The Erudite NMA system**

Key Erudite Features

- Persistent objects for main stream languages (C++, Java, Python, etc.)
- Storage-Class Memory (Flash RAM).
- Near Memory Acceleration and memory-channel networking
- API for collaborative CPU/GPU/NMA execution

High-level Diagram of Current MCN Implementation



- Host requires a new kernel driver to transform TCP/IP packet to memory access and vice versa.

Summary

- Creative experiential learning advisor (CELA) as a grand challenge use case for cognitive capabilities

- Cognitive application builder (CAB) to make the underlying heterogeneous infrastructure easy to consume for cognitive application developers

- Cognitive systems innovations (Erudite) for workload acceleration, including Near Memory Acceleration (NMA)