Raytheon

HTCondor Week 2018

Managing Caffe Deep Learning with HTCondor

Integrated Defense Systems

Michael V. Pelletier, Principal Engineer

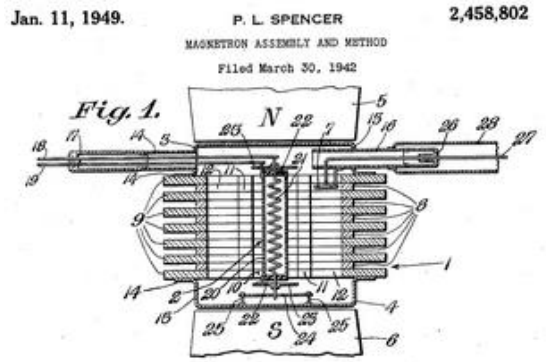
May 2018

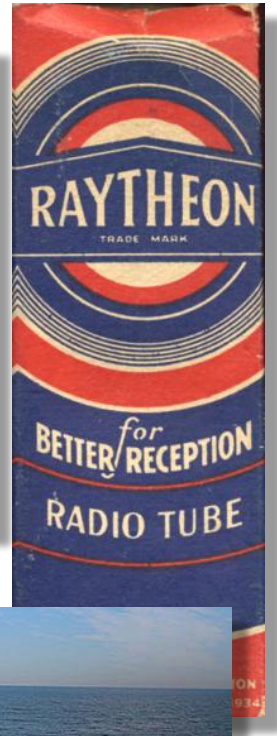
Approved under eTPCR IDS-14060

About Raytheon

Raytheon
Integrated Defense Systems

- **Raytheon Company**
  - Founded in 1922
  - $27.7 billion in 2017 sales
  - 64,000 employees worldwide
- **Integrated Defense Systems**
  - Headquartered in Tewksbury, Mass.
  - Broad portfolio of weapons, sensors, and integration systems across multiple mission areas including **air and missile defense radars**; early warning radars; naval ship operating systems; command, control, communications; air traffic systems
- **Information Technology**
  - 2018 IT Goal: “Identify and implement new innovations and best practices that yield business productivity and cost optimization, without sacrificing quality.”





Cmacd @ Wikipedia, GFDL

US Navy

Nearly a century of technology and innovation

About Michael V. Pelletier

Raytheon
Integrated Defense Systems

Raytheon

- Joined Raytheon IT in 2009
- 2009-2015: Sysadmin at Raytheon Missile Defense Center
- 2015-present: Program Execution & Business Work Environments
- Highlights
  - 2017 & 2018 Excellence in Information Technology Awards
  - 2013 Excellence in Engineering & Technology Award
    - In recognition of HTCondor implementation
  - 2011-2013 Technical Honors Peer Recognition
  - 2010 Authors & Inventors Award
  - 2009 Excellence in Information Solutions Award
- Prior Roles
  - Nortel: 2000-2008 – Global infrastructure services
  - Taos: 1998-2000 – Unix and Linux consulting services
  - TechTeam Global: 1992-1998 – Ford Motor Company IT services
  - University of Michigan Engineering: 1988-1992

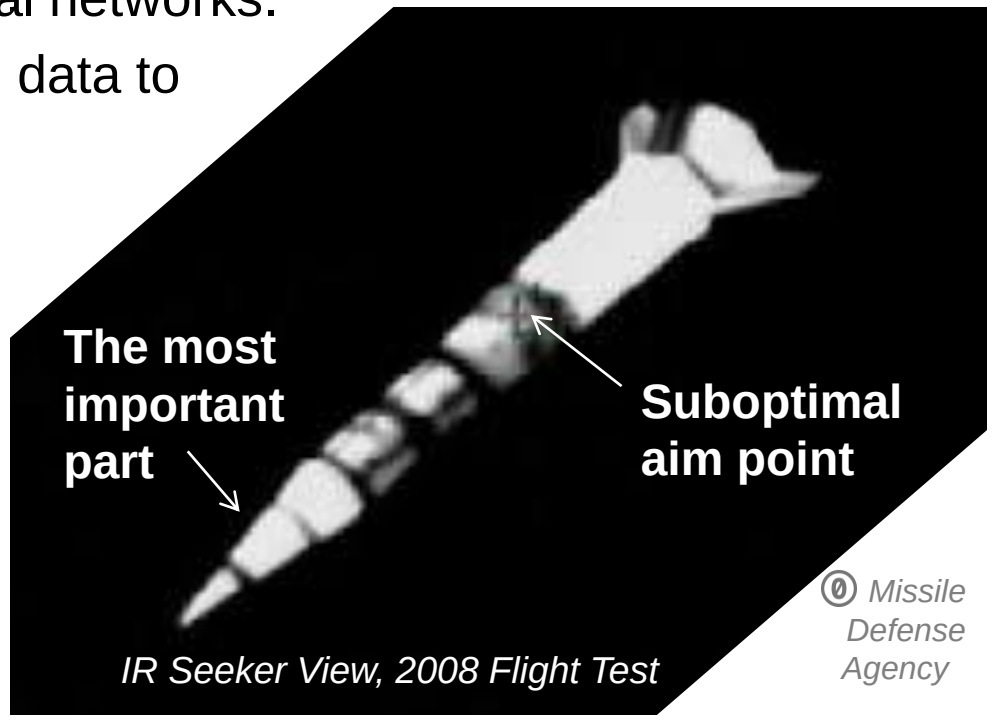
Raytheon Advanced Media, March 2017

Caffe Deep Learning Framework

Raytheon
Integrated Defense Systems

- Deep Learning is Large Neural Networks*
  - "…a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks."
  - "…we now have fast **enough computers** and enough data to actually train large neural networks."
- Caffe | Deep Learning Framework
  - A tool used to describe and deploy neural networks in a fast, modular, expressive, open way
    - Developed by Yangqing Jia at the Berkeley Artificial Intelligence Research Lab
- **Image Recognition and Classification**
  - Has important applications in sensor technology

Shortcomings of past discrimination software:



Better aim is now done by conventional algorithms, but AI can do it better, faster, cheaper

*Jason Brownlee, "What is Deep Learning," MachineLearningMastery.com, 16 Aug 2016

Neural Network Training

Raytheon
Integrated Defense Systems

© US Navy

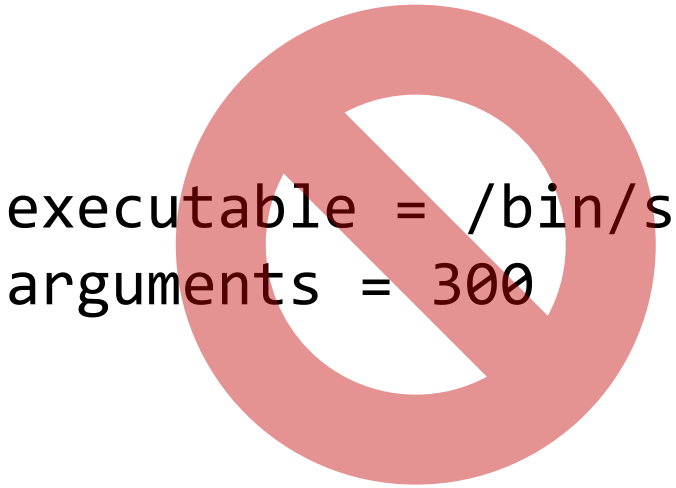
© Missile Defense Agency

That is: More Machines, CPUs, Memory, Disk, GPUs

Challenges of Complex Software

Raytheon
Integrated Defense Systems

- Command-line arguments are no longer sufficient to fully define a job




```
name: "LogReg"
layer {
  name: "mnist"
  type: "Data"
  top: "data"
  top: "label"
  data_param {
    source: "input_leveldb"
    batch_size: 64
  }
}
layer {
  name: "ip"
  type: "InnerProduct"
  bottom: "data"
  top: "ip"
    num_output: 2
  }
}
```

A simple logistic regression classifier, from the Caffe site's net layer blob tutorial https://tinyurl.com/yabempnq

Complex software often requires complex configuration

Why Not Just Transfer_Input_Files?

Raytheon
Integrated Defense Systems

- Input configuration can't be examined unless something reads it


```
executable = caffe
arguments = train --solver=solve.prototxt
should_transfer_files = True
transfer_input_files = solve.prototxt
```

- Unknowns:
  - Does the job need a GPU? If so, how many?
  - Does the job need other input files? Which ones?
  - Where will the output be written?

`condor_submit` needs help answering these questions!

We need to run examination code before or during condor_submit

Providing Submit Assistance

- Up to 2015 and HTCondor 8.2: Job submission scripts
  - Read and sanity-check the input file and build a submit description based on its contents
    - Queue, Change Something, Repeat 1000 times
    - Inexperienced authors tended to create administrator headaches
      - Sanity checking of inputs is an acquired taste
    - A variety of different scripting languages
      - CSH, SH, Bash, Perl, Python… MATLAB?
    - Created another custom piece of the workflow that needed to be maintained by a suitable subject-matter expert

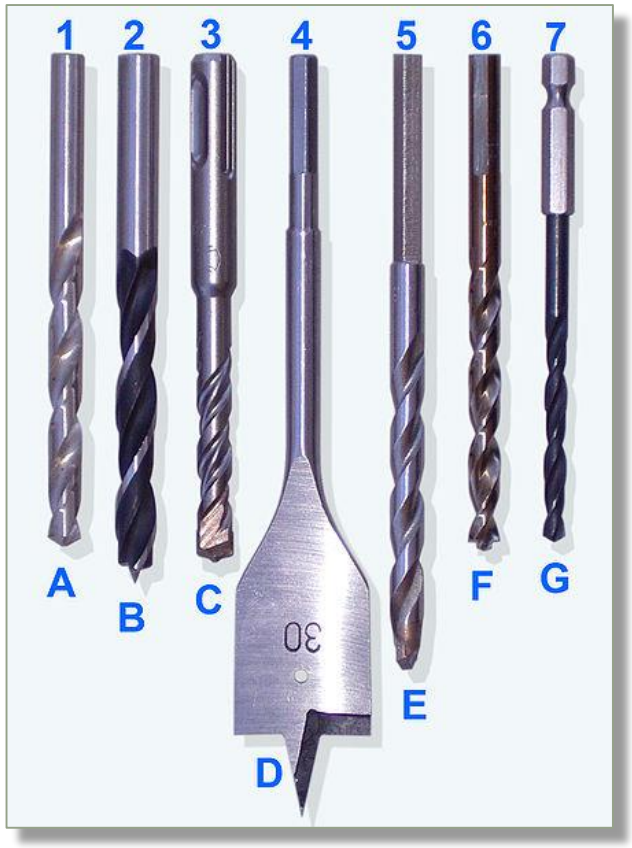
Torsten Bätge, Wikimedia Commons, GFDL, CC-BY-SA

Adds complexity on top of complexity

Modular Submit Assistance in v8.4+

Raytheon
Integrated Defense Systems

- `Include command : mycommand`
  - "Run a command and include its output in the submit description"
  - **Encourages Modularity** (Unix Rule 1)
    - `include command : lookup_email -submit $ENV(USERNAME)`
      `→ notify_user = michael.v.pelletier@raytheon.com`
  - **Encourages Simplicity** (Unix Rule 5)
    - No need to generate an entire submit description and command line, just generate one simple piece of the whole
  - **Encourages Transparency** (Unix Rule 7)
    - Clearly illustrates the thought process of the submitter, and uses key-value-pair format that makes it easy to identify valid output and allows use of "error:" and "warning:" pragmas



Ssawka @ pl.Wikipedia, GFDL, CC-BY-SA

Rule of Modularity
Eric S. Raymond's 17 Unix Rules
https://en.wikipedia.org/wiki/Unix_philosophy

"Do one thing and do it well." – Doug McIlroy, Unix Pipe Inventor

Caffe Job Submit Assistance

- Solver Attribute Definitions

```
#!/usr/bin/perl
# Convert solver prototxt definitions to HTCondor submit lines
open(F, "$ARGV[0]") or print "error : could not open solver file\n"
    and exit;
while(<F>) {
    next if m{^\s*#};
    s{^\s*([^:]+)\s*"?([^"]+)"?}
        {MY.Caffe_$1 = "\$(CAFFE_$1)"\nCAFFE_$1 = $2\n};
    s{(CAFFE_solver_mode = GPU)}{$1\nCAFFE_NEEDS_GPU = True}i;
    print;
}
```

Unix Rule 6: Parsimony

```
net: "train_val.prototxt"
test_iter: 256
test_interval: 256
base_lr: 0.001
lr_policy: "step"
stepsize: 4096
display: 16
max_iter: 61440
iter_size: 1
type: "SGD"
momentum: 0.1
gamma: 0.25
weight_decay: 0.005
snapshot: 4096
snapshot_prefix: "native_train"
solver_mode: GPU
```

```
Include command : condor_caffe_prototxt.pl $(SOLVER)
```

```
MY.CAFFE_net = "$(CAFFE_net)"
CAFFE_net = train_val.prototxt
MY.CAFFE_test_iter = "$(CAFFE_test_iter)"
CAFFE_test_iter = 256
# … and so on …
```

All solver values are now in both submit and job ClassAd attributes

Acting on Job Configuration Data

Raytheon
Integrated Defense Systems

- GPU Allocation
  - Submit helper script detected the need for a GPU in the solver.prototxt, so we need a request:

```
if $(CAFFE_NEEDS_GPU)
    request_gpus = $(GPUS:1)
endif
```

  - User might specify GPUS=2 on the command line, so we convert it to an argument for Caffe:

```
REQGPU_INT = $INT(request_gpus)
GPU_ORDINAL = $CHOICE(REQGPU_INT, "", "0", "0,1", "0,1,2", "0,1,2,3", "too_many_gpus_requested")
GPU_ARG = --gpu=$SUBSTR(GPU_ORDINAL, 1, -1)
```

$INT since request_gpus could be an expression

CUDA_VISIBLE_DEVICES is set by Condor, and GPUs appear to the job in sequential order.

Removes the quotation marks from the list of GPUs

  - Work it… Make it… Do it… Makes us…

```
requirements = (CUDACapability >= 7.0) && $(requirements:True)
rank = (CUDAComputeUnits * CUDACoresPerCU + CUDAClockMHz) + $(rank:0)
```

Harder
Better
Faster
Stronger

Daft Punk -TNS Sofres @ flickr CC-BY

NVIDIA Volta Tensor Cores are supported in CUDA Capability 7.0

Reading Nested Configuration Files

- Peeling the configuration onion
  - Is the required “net” parameter in the solver config?
  - Does the training net file exist?
  - Read neural network name for job description:

```
if defined CAFFE_net
  include command : " /usr/bin/perl -e ' \
  open(F, ""$(CAFFE_net)"") or print "error : open failed\n"; \
  while(<F>) { \
  next unless m{^\s*name:}i; \
  s{^\s*([^:]+)\s*:\s*""?([^""]+)""?} \
    {MY.Caffe_$(DOLLAR)1 = ""\$(DOLLAR)(CAFFE_$(DOLLAR)1)""\n \
        CAFFE_$(DOLLAR)1 = $(DOLLAR)2}; \
  print; last;} ' "
else
    error : No net file is defined in $(SOLVER)
endif
description = $(CAFFE_name)
```

solver.prototxt

```
net: "train_val.prototxt"
test_iter: 256
test_interval: 256
base
lr_p
step
disp
max_
iter
type
nome
gamm
weig
snap
snap
solv
```

train_val.prototxt

```
name: "LogReg"
layer {
  name: "mnist"
  type: "Data"
  top: "data"
  top: "label"
  data_param {
    source: "input_leveldb"
    batch_size: 64
  }
}
layer {
  name: "ip"
  type: "InnerProduct"
```

Inline commands keep the helpers with the submit, but conversion is tricky!

Preventing “condor_exec” in Log Names

- “*Unfortunately, some programs read argv[0] expecting their own program name and get confused if they find something unexpected like condor_exec.*” – HTCondor Manual § 2.15.1
- My first Caffe submission’s info log file name:

```
condor_exec.node04.proj.ray.com.pelletm.log.INFO.20180713-180043.28218
```

- The undocumented workaround? Wrap it in a shell exec:

```
executable = /bin/sh
transfer_executable = False
arguments = " -c 'exec ./$BASENAME(CAFFE_EXE) train --solver=$(SOLVER) $(GPU_ARG:)' "
transfer_input_files = $(CAFFE_EXE) $(transfer_input_files:)
```

The argv[0] becomes “caffe” and the log names return to normal

Tracking Progress

Raytheon
Integrated Defense Systems

- Job parameters which were parsed from the solver file help construct a progress meter in the job description

```
MY.WantIOProxy = True
MY.HookKeyword = "Checkfile"
MY.CheckFileDataRegexp = " Iteration (\d+), "
MY.CheckFileDataAttributeName = "Caffe_last_logged_iter"
MY.Caffe_last_logged_iter = -1
MY.JobDescription = strcat( \
    "$(CAFFE_name) $(CAFFE_solver_mode) ", \
    Caffe_last_logged_iter, "/$(CAFFE_max_iter)" )
```


```
$ condor_q -nobatch pelletm
-- Schedd: had-schedd@ : <10.0.0.100:9618?... @ 05/03/18 02:20:0
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
7067.0   pelletm         5/2  17:40   0+08:39:48 R  0    944 (LogReg GPU 47400/61440)

1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```

Refer to last year's presentation for more details on this technique.

Preventing Double Transfer of Logs

Raytheon
Integrated Defense Systems

- Caffe links its four log file names to short names:

```
caffe.INFO ->
caffe.node04.proj.ray.com.pelletm.log.INFO.20170713-180043.28218
```

  - Condor output transfers follow symbolic links, so we would get two copies of all the logs
  - A simple inline PostCmd cleanup prevents this:

```
should_transfer_files = Yes
MY.PostCmd = "../../../../../../../../../bin/rm"
MY.PostArguments = " -f $BASENAME(CAFFE_EXE).INFO $BASENAME(CAFFE_EXE).WARNING \
                     $BASENAME(CAFFE_EXE).ERROR $BASENAME(CAFFE_EXE).FATAL "
```

  - PostCmd paths are rooted at the scratch directory, so in order to use the exec node's `/bin/rm` command you need to climb out of however deep the exec node's scratch directory is to reach it.

Default output transfer is all files in the top level scratch directory.

Caffe Snapshot Layout

Raytheon
Integrated Defense Systems

- Caffe snapshot structure
  - `$(CAFFE_snapshot_prefix)_iter_4096.caffemodel`
    - The neurons, the connections and their weights, and the propagation functions, at iteration 2000
  - `$(CAFFE_snapshot_prefix)_iter_4096.solverstate`
    - The state of the solver of the neural network at iteration 2000
  - The size of the files depend on the complexity of the neural network which is being solved. Even a simple net easily reaches 20 MB per file, or 40 MB per snapshot.
- Command line for resuming from a snapshot
  - `caffe -snapshot snapshot_iter_4096.solverstate -solver solver.prototxt`

The submit description's `arguments` command gets a bit more tricky…

On-Exit and Periodic Snapshots

- Caffe takes a snapshot when manually terminated with Control-C
  - Configure the same signal (SIGINT) for the exec node to use, and give it plenty of time to write the snapshot:

```
kill_signal = 2
max_job_vacate_time = 300
```

- Periodic snapshots need plenty of disk
  - This solver configuration will generate 15 snapshots totaling about 800 megabytes of disk space (for this model)

```
SNAP_SIZE_KB = 54000
NUM_SNAPS = $(CAFFE_max_iter) / $(CAFFE_snapshot)
request_disk = $INT(NUM_SNAPS) * $(SNAP_SIZE_KB)
transfer_output_files = $DIRNAME(CAFFE_snapshot_prefix) \
  caffe.INFO caffe.WARNING caffe.ERROR caffe.FATAL
```

solver.prototxt

```
net: "train_val.prototxt"
test_iter: 256
test_interval: 256
base_lr: 0.001
lr_policy: "step"
stepsize: 4096
display: 16
max_iter: 61440
iter_size: 1
type: "SGD"
momentum: 0.1
gamma: 0.25
weight_decay: 0.005
snapshot: 4096
snapshot_prefix: "native_train"
solver_mode: GPU
```

When you specify one thing in output transfers, you must specify everything.

Resuming from a Snapshot

- User sets a new run's iteration using `ITER=32768` on command line to cause a file transfer into scratch, **or** periodic and exit snapshots are spooled into scratch after a vacate and resume:

```
if defined ITER
    transfer_input_files = $(transfer_input_files), \
        $(CAFFE_snapshot_prefix)_iter_$INT(ITER).solverstate, \
        $(CAFFE_snapshot_prefix)_iter_$INT(ITER).caffemodel
endif

arguments = " -c 'export CAFFE_NEWEST_SNAP=""$(DOLLAR)(/bin/ls -rv
$(CAFFE_snapshot_prefix)_iter_*.solverstate 2>/dev/null | echo --snapshot=$(/bin/head -1))""; \
  exec ./$BASENAME(CAFFE_EXE) $(COMMAND:train) --solver=$(SOLVER) $(GPU_ARG:) \
  $(DOLLAR)CAFFE_NEWEST_SNAP' "
```

Input-transfer file lists require commas, since filenames might have spaces on Windows

Evaluated at job startup on the exec node

Launching in a shell adds some useful capabilities along with an argv[0] fix

Accelerating the Virtually Impossible

Raytheon
Integrated Defense Systems

Xkcd.com #1425 – CC-BY-NC 2.5

- Configuration parsing eliminates need to manually maintain consistency between the Caffe solver config and the submit description.
- Automatic HTCondor GPU allocation eliminates manual GPU scheduling and GPU runtime collisions for multi-member teams
- Adapting Caffe snapshot mechanisms to HTCondor makes it easier for users to manage and utilize their snapshots

Questions and Comments