Medical Compute with ChRIS on the MOC

PowerPC & x86_64 GPU Usage & Benchmarking

Elizabeth Slade | Shineun Yoon | Bowen Jia | Haoyang Wang | Kefan Zhang

Quick Review

An open source platform for medical analysis

The goal is to democratize application development for medical analysis applications

Sprint4 Overview:

Matrix Multiplication App for x86 & PowerPC

Object Detection App for x86

Deployment on MOC

What we have learned:

- How to run Plugin on PowerPC
- Docker workflow
- pfurl for pman & pfioh
- ChRIS Instance

Last Sprint:
PROBLEMS NEED ATTENTION

- ✓ CUDA driver
- ✓ Hardware incompatible on VM
- ✓ Building pman and pfioh images on Power9 machine

Dockerized `Matmul` on Power9 & X86

Driver :

The ubuntu docker image provided by FNNDSC actually not contains a cuda driver.

->We use the base docker image provided by nVidia.

Hardware:

✓ Asked Power 9 admin for help to run a nvidia container

✓ Use -e NVIDIA_VISIBLE_DEVICES=x to specify the GPU id on x86

Power 9

Facts:

- All c/c++ based python libraries have to be complied on Power 9 for `pip3 install` command.
- Compile Environment.
- Dependency issues.

Power 9 version plugin

```
#
#   docker run -ti -e HOST_IP=$(ip route | grep -v docker | awk '{if(NF==11) print $9}') --entrypoint /bin/bash local/pl-MatMultiply
#

FROM nvidia/cuda-ppc64le:10.1-devel
LABEL maintainer "NVIDIA CORPORATION <cudatools@nvidia.com>"
ENV APPROOT="/usr/src/matmultiply"
COPY ["matmultiply", "${APPROOT}"]
COPY ["requirements.txt", "${APPROOT}"]
COPY ["dependencies.txt","${APPROOT}"]
WORKDIR $APPROOT


RUN apt update
RUN apt install -y build-essential python3 llvm-7
RUN ln -s /usr/bin/llvm-config-7 /usr/bin/llvm-config
RUN apt install -y python3-pip
RUN pip3 install -r dependencies.txt
RUN pip3 install -r requirements.txt
ENTRYPOINT ["python3"]
CMD ["matmultiply.py", "--help"]
```

Compile Environment

2 Steps of pip install

Demo for Power 9

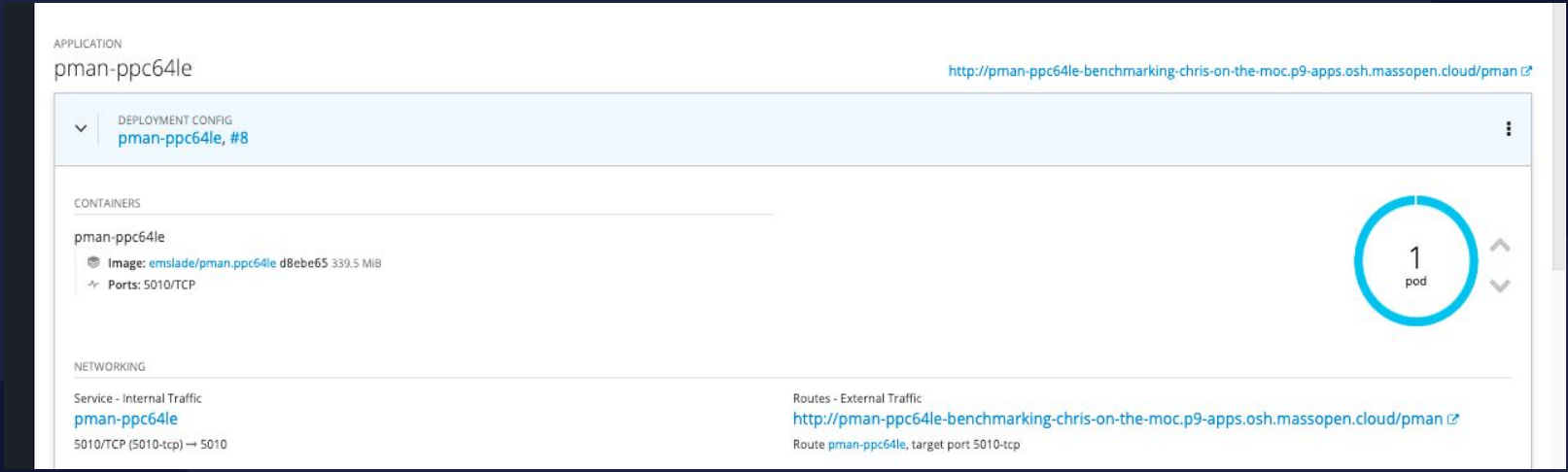
```
[root@bu-21-9 ~]# docker run  --security-opt label=type:nvidia_container_t pupiltong/pl-matmultiply-ppc64le:latest matmultiply.py in out
0.6105351448059082
0.6105434894561768
[root@bu-21-9 ~]# nvidia-smi
Tue Mar 31 20:43:32 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.39       Driver Version: 418.39       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000004:04:00.0 Off |                    0 |
| N/A   40C    P0    40W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000004:05:00.0 Off |                    0 |
| N/A   44C    P0    39W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-SXM2...  On   | 00000035:03:00.0 Off |                    0 |
| N/A   39C    P0    37W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100-SXM2...  On   | 00000035:04:00.0 Off |                    0 |
| N/A   43C    P0    38W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
[root@bu-21-9 ~]# uname -a
Linux bu-21-9.infra.massopen.cloud 4.14.0-115.18.1.el7a.ppc64le #1 SMP Wed Jan 29 11:49:09 EST 2020 ppc64le ppc64le ppc64le GNU/Linux
[root@bu-21-9 ~]#
```

DEMO

Powerful Cluster

Ppc64le Arch

pman on MOC's OpenShift



Process Manager, pman on OpenShift

p9-openshift.osh.massopen.cloud:8443/console/project/benchmarking-chris-on-the-moc/browse/rc/pman-ppc64le-1?tab=logs

OPENSHIFT CONTAINER PLATFORM Application Con...

Medical Compute using ChRIS on the MOC -- PowerPC and x86_64 GPU usage and benchmarking

Overview
Applications
Builds
Resources
Storage
Monitoring
Catalog

Deployments › pman-ppc64le › #1

pman-ppc64le-1 created 4 days ago

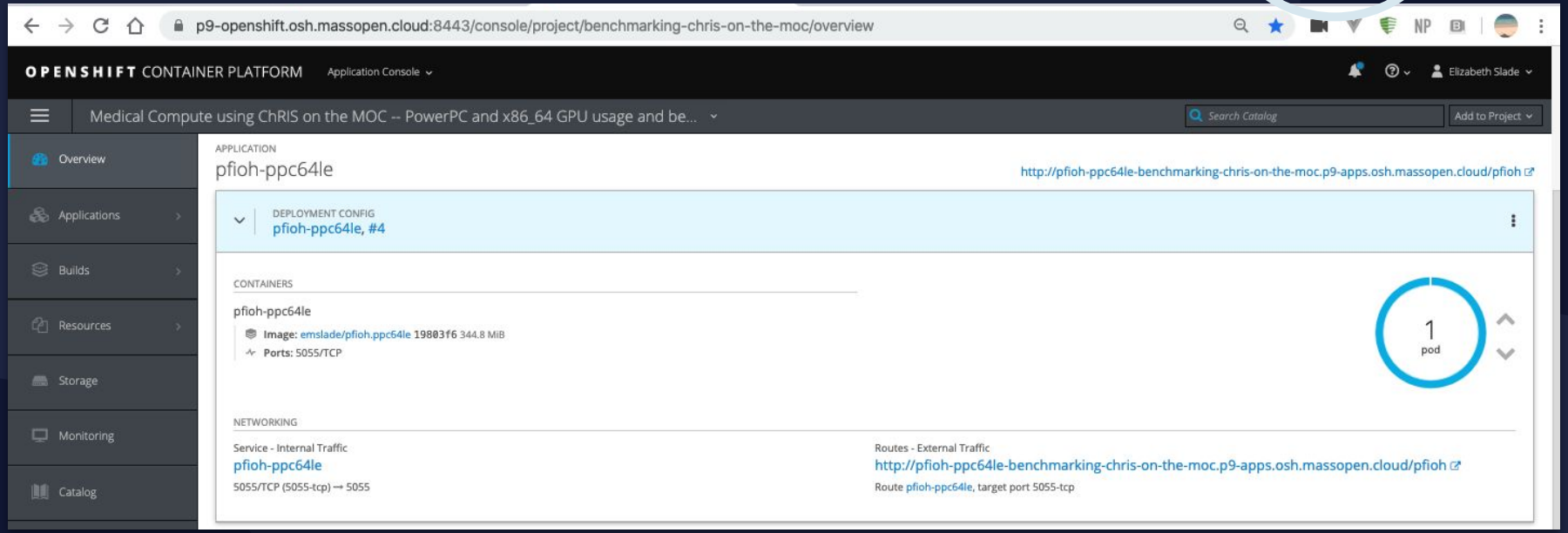
app pman-ppc64le openshift.io/deployment-config.name pman-ppc64le

Details Environment Metrics Logs Events

Status: Active

```
2020-03-26 16:51:03  | pman-ppc64le-1-q9cr4 |                    pman.py:pman.__init__() |

 _ __  _ __ ___   __ _ _ __
| '_ \| '_ ` _ \ / _` | '_ \
| |_) | | | | | | (_| | | | |
| .__/|_| |_| |_|\__,_|_| |_|
| |
|_|


font generated by:
http://patorjk.com/software/taag/#p=display&f=Doom&t=pman


                              Process-MANager

          A simple compute process tracking and management framework.

                           -- version 2.0.0.10 --

   'pman' is a server that spawns and tracks processes executed on an underlying
   shell/system. Internally, 'pman' provides services using Zero MQ and not
   http-based services.

   'pman' listens for REST comms and provides facilities for running, stopping,
   querying, and managing command-line processes.

   'pman' is also container-aware and can run containerized compute
   using either swarm or openshift (assuming suitable installed underlying
   infrastructure).



             +-------------------------------------------------------+
             | Use '--enableTokenAuth' and '--tokenPath <tokenPath>'  |
             |         arguments for secure communication.            |
             +-------------------------------------------------------+
```

pfioh on MOC's OpenShift



Status of Plugin Development

Last Sprint	Object Detection	Matrix Multiplication
x86	✓	🕒
Power9	🕒	🕒

Status of Plugin Development

This Sprint	Object Detection	Matrix Multiplication
x86	✓	✓
Power9	🕒	✓

Modification: Parameter option

Examples

-c start, step, stop

```
docker run --runtime=nvidia                                          \
           -e NVIDIA_VISIBLE_DEVICES=1                               \
           -v $(pwd)/in:/incoming -v $(pwd)/out:/outgoing            \
           fnndsc/pl-matrixmultiply                                  \
           python3 matmultiply.py                                    \
           -c 32,32,128                                              \
           /incoming /outgoing
```

Matrix size = (COE x TPB) ^2, TPB fixed as 32

Modification: Output as csv file

robertmorrislike — chris-local@titan (linux-gnu) pts/14 ~/kefan/kefan7/out — ssh -X -p 7639..

Matrix_Size	Start_Time	Finish_Time	Elapse_Time
1048576.0	1585700450.1756785	1585700450.8636434	0.687964916229248
4194304.0	1585700450.8645773	1585700451.027952	0.163374662399292
9437184.0	1585700451.028828	1585700451.4861693	0.45734143257141113
16777216.0	1585700451.487207	1585700452.4652367	0.978029727935791

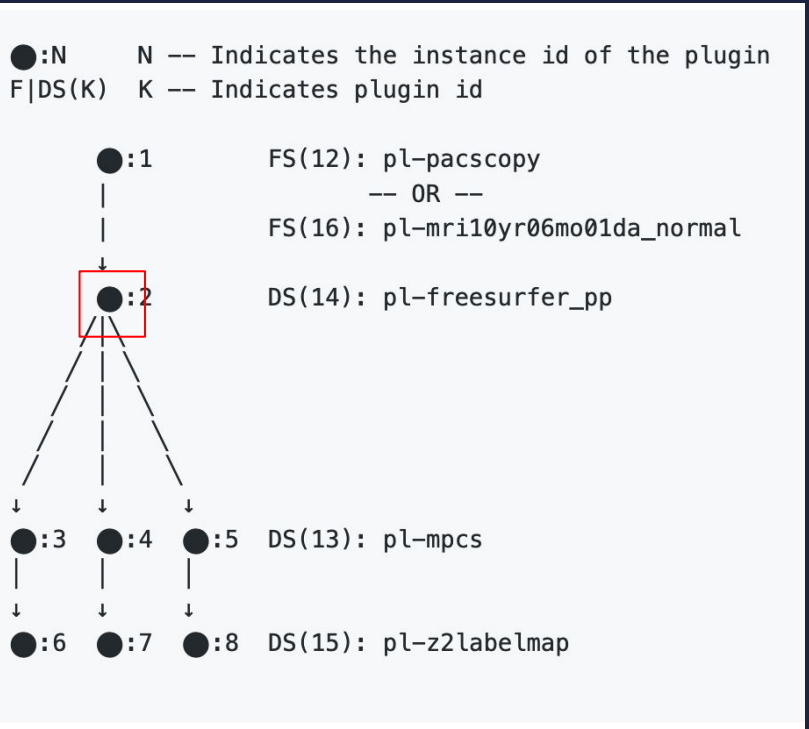
(END)

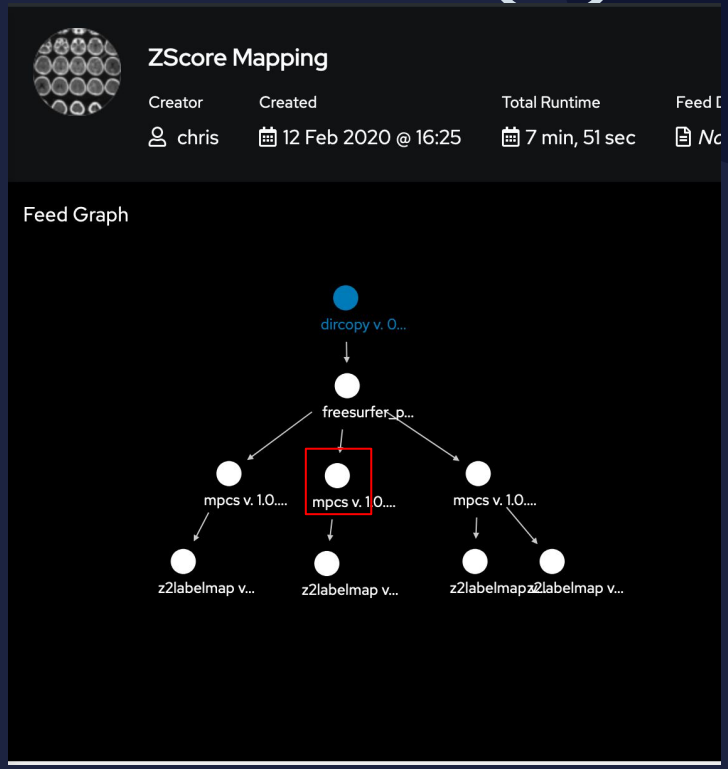
Elapse = Finish - Start

Now, we can benchmark with Matrix multiplication plugin

MatMul demo

Big Picture: a “node” on ChRIS





Recap: 2 ways to run a plugin

- Run it through Docker containers
  - Successfully build 3 images(local, x86_moc, ppc_moc).
  - maintained on Dockerhub under FNNDSC repo.
- Run it through Chris Instance
  - What does that means?

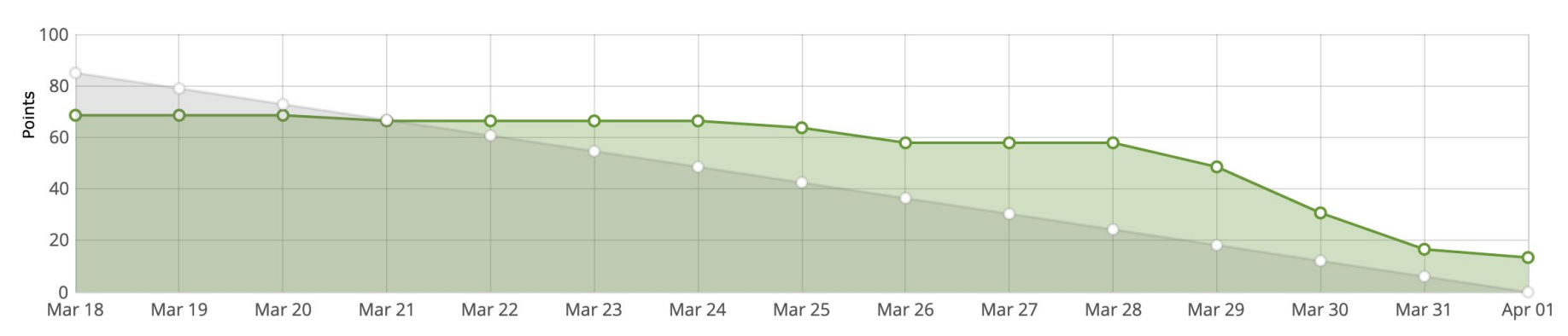
ChRIS Instance

- Register runnable plugin to Chris store.
- Fire up a ChRIS instance.
  - instantiated as Django-mysql project
  - offering a collection+json REST API.

Next Step

- Integrating **Object-Detection** to ChRis plugin
- Test plugins through **ChRIS instance**
- Test plugins through **ChRIS GUI**
- **Benchmarking** plugin result between x86 vs PowerPC

BURNDOWN CHART



THANKS

Any questions?