Julia and Singularity discussion

April 27, 2019

Discussion plan

- Overview of Julia
  - multi-architecture development platform
  - interoperability with QUDA
- Singularity containers

Julia intro

- Open-source and free programming language:
  - https://julialang.org
  - Developed since 2012 (MIT licencse)
  - Multi-platform (GNU/Linux, Mac OS X, Windows)
- Desinged for performance and usability
  - Matches Python in terms of ease of code and maintenance, while targets to achieve the speeds of C
  - Multi-core, clusters, GPUs
  - Easy learning, support dynamic typing etc. (python-like)
  - Support an interactive shell (REPL, "Read-Eval-Print-Loop")
- Interoperable with the outside world:
  - Allows to import C / Python code directly

Julia intro

- Uses JIT compilation
  - LLVM compiler framework to generate native machine code
  - @*code_llvm* macro outputs LLVM IR
- Easy to install
  - Can be extended with modules (Pkg.add("..."))
  - E.g., CUDA support via CUDAdrv and CUDAnative modules

Julia language

- Supports both dynamic and static type systems
  - All basic data type (Int8, Float64 etc.)
  - Abstract types (Numeric, Integer, Signed)
  - Composite types
- Types can take type parameters, so that type declarations can actually introduce a whole family of new types (conceptually similar to templates in C++)
- Supports multi-dim. arrays
- Supports views into arrays, reshaping, slicing etc.
- Uses multiple dispatch as a paradigm to express many object-oriented and functional programming patterns

Interoperability with C

- Main features
  - Make calls to C without any hassle
  - No overhead
  - No further processing or compilation needed before calling the C function, hence it can be used directly!
- Syntax:
  ccall((:name,"lib"), return_type, (arg1_type, arg2_type...), arg1, arg2)
- Iteroperability with C++ libs is more involved but doable (requires external modules)

Calling QUDA routines from Julia

- QUDA provides with C interface to all major computational routines
- Main interface structures (QudaInvertParam,QudaGaugeParam etc) can be mirrored in Julia

Using CUDA in Julia

- CUDA programming via CUDAnative module
- memory management, kernel generation etc.
- GPU specific memory objects: CuArray

Simple SU3xSU3 test

```
function su3_test(x, y)
   i = (blockIdx().x-1) * blockDim().x + threadIdx().x

   y_site_view = view(y,i,:,:,1)
   x_site_view = view(x,i,:,:,1)

   for r in 1:3
     for c in 1:3
       temp = ComplexF32(0.0f0, 0.0f0)
       for j in 1:3;
         temp += x_site_view[r, j]* y_site_view[j, c]
       end
       y_site_view[r, c] = temp
     end
   end
   return
 end

 #####################################################################################################################

 csGrid   = QJuliaGrid.QJuliaGridDescr_qj{ComplexF32}(QJuliaEnums.QJULIA_CUDA_FIELD_LOCATION, 0, (N,N,N,N,))
 gaugeParam  = QJuliaGrid.CreateGaugeParams( csGrid )

 cuda_su3_m1 = QJuliaCUDAFields.CreateGenericField( gaugeParam )
 cuda_su3_m2 = QJuliaCUDAFields.CreateGenericField( gaugeParam )

 cuda_su3_m1_accessor = QJuliaFields.create_field_accessor(cuda_su3_m1)
 cuda_su3_m2_accessor = QJuliaFields.create_field_accessor(cuda_su3_m2)

 accessor_dims = size( cuda_su3_m1_accessor[1] )

 len = accessor_dims[1]; nthreads = 128; nblocks= ceil.(Int, len ./ nthreads)

 @cuda blocks=nblocks threads=nthreads sun_test(cuda_su3_m1_accessor[1], cuda_su3_m2_accessor[1])
```

Singularity containers for HPC applications

- What brought us to containers
- Quick overview of Singularity
- Challenges and future perspectives

Why containers

- There are applications/workflows that difficult to maintain
  - e.g., legacy codes are working on old OS, new codes require new OS
  - e.g., applications with complicated dependencies
  - e.g., Tensorflow may need newer GLIBC than that is on the host system
- ML frameworks at FNAL are provided within containers

Singularity (basic) facts

- (starting from ver 3.1.0) Fully compliant with the Open Containers Initiative (OCI) standards
- Not Docker but supports docker containers/hub
- You can get into an image shell, execute commands, have access to local FS etc.
- Image can be running on different OS:
  - containers are kernel-independent
  - well, in some cases they does not
  - for kernel-dependent features, a container platform may not be the right solution

Singularity (basic) facts cont.

- Limits user privileges and access from within the container
- Multi-process execution
  - container and host MPI impl. and version must match
  - for multi-node performance: e.g., container must be built with the proper OFED as the host
- running on NVIDIA gpus requires –nv option for the exec command

- container and host MPI impl. and version must match
- for multi-node performance: e.g., container must be built with the proper OFED as the host

Definition file

```
# Ubuntu MILC container

BootStrap: docker
From: nvcr.io/hpc/milc:quda0.8-patch4Oct2017
IncludeCmd: false

%labels

Based on NVCloud image
Date: 2019-01-25 (A.S)

%environment
    # set these environment variables
    export PATH=$PATH:$CUDA_ROOT/bin
    export LD_LIBRARY_PATH=$CUDA_ROOT/lib64

%runscript
    # Check the current environment
    chk_nvidia_uvm=$(grep nvidia_uvm /proc/modules)
    if [ -z "$chk_nvidia_uvm" ]; then
        echo "Problem detected on the host: the Linux kernel module nvidia_uvm is not loaded"
    fi
    exec /bin/bash

%setup
    # Runs from outside the container during Bootstrap
    workdir=$(pwd)

%post
    # Runs within the container during Bootstrap
    # make lqcd filesystem mount points
    mkdir /scratch /data /project /lqcdproj
```

Building containers

Building from scratch (e.g., pre-build HPC rpm, OpenHPC)

Using pre-build docker/singularity images (docker hub or NVIDIA cloud)

- sudo singularity build test.simg deffile.def
- singularity exec –nv test.simg
- mpirun np n singularity exec test.img testprog

Conclusion

- Julia is a convenient framework for code prototyping:
  - easy to use
  - can be even optimized for performance if necessary
  - can interoperate with existing libraries
- Singularity containers needs more testing for POWER9 env.