## Application View

### **Goal**

**Survey application teams and vendors to map out current and emerging use cases, evolving requirements, and interface approaches.**

### **Session Chairs**

### **ORNL: Peter Groszkowski**

### **IBM: Antonio Corcoles**

### **Prompts**

* Which applications (chemistry, optimization, ML) are currently deployed on NISQ hardware vs FT prototypes?
* What's the workload characterization of hybrid applications? High/low/equivalent quantum/hpc? And time scales?
* What software‑stack interfaces do application groups require? How will those change with FT? (e.g., calibration information, gate/pulse level operations, equivalent to classical assembly instructions or IOCTL/kernel system calls)
* How are teams anticipating changes to hybrid applications as we move from NISQ to FT? (e.g., latency, fidelity, coherence times, error rates)
* Current SDK and programming paradigms tightly integrate algorithms and runtime. Should we be moving towards **standalone libraries** (akin to BLAS), with no dependency on a specific framework
* How significant a concern is software framework lock-in (e.g., Qiskit, PennyLane) for application developers?
  + As applications grow in scale and complexity, would they benefit from a modular, composable, interoperable, library‑centric ecosystem with minimal dependency chains?
* Are there software metrics which would be useful to expose to the application?
* What changes do we expect when we go from the NISQ era to FT? How will the software reflect that?
  + Mapping libraries from application domain → circuit
  + Ease of development
* What are things which will not change from NISQ to FT?
  + structure of what a quantum computational unit is?
  + FT can have different gates, but gates remain the
* Is quantum circuit the lowest level? Lower than the quantum circuit becomes the vendor domain?
  + What's the lowest common interface
  + FT can have different gates
  + How services can evolve as systems scale

Application View

Goal

Survey application teams and vendors to map out current and emerging use cases, evolving requirements, and interface approaches.

Session Chairs

• ORNL: Peter Groszkowski

• IBM: Antonio Corcoles

Prompts

Q: applications; how to break down the problem into pieces that are consumed by diff paradigms

identify probs amenible to quantum hardare vs tensor networks

short term SQD; (utilized hpc)

small fraction of sample being "right"; get reasonable solution (energy)

example of need for HPC;

limit in terms of "scale of this approach"

limit: how big of a classical matrix you can diagonalize

checks against state of the art classical methods

nisq world; algorithm depenence on different modalities

ground state

generally

how to select portion of problems that should go to quantum vs classical

embeding real quantum problems into usable/workable aglos

\*\*\*Group3

Q: kind of problems for noisy processors

vs fault tollerant systems

• What software‑stack interfaces do application groups require? How will those change with FT? (e.g., calibration information, gate/pulse level operations, equivalent to classical assembly instructions or IOCTL/kernel system calls)

\*\*\*Group1

compilation to pulses is expensive;

interface needs to allow to shared memory to shared memory compiled system (rigetti)

qiskit does this some extent

how do we device on the interface details

levels of APIs

inspiration from the software side;

see; MILR (e.g. language called mojo)

future take

open qasm 3; "everything"...

ultimately use cases will drive the standard

Q: does qasm 3 capture everything we need?

no; open pulse, specialized routines (see extern keyword)

FT/QEC

do we need to extend qasm 3 or other rep for FT

meta language on top of qasm 3

PUNCH LINE: not entirely clear; but possible that we'll need to augment interfaces

\*\*\*Group2

what interfaces work today vs what does not?

high level (e..g vqe algo; do it all)

vs more granularity

different users might need different

should there be levels of interfaces???

challenges:

generality vs performance

we are biased by some very specific applications (vqe?)

algorithmic skeletons (dev in europe)

Q: why should we based on classical computers

is there a dramatically bettter way to do things?

"because we've done it before" "early on, we're still exploring"

\*\*\*Group3

who owns it? who pays?

hypothesis

likely scenario: high level solvers for few key things

app deves may often need what happens behind the scenes

\* Are there specific software metrics which would be useful to expose to the application?

\*\* details of latest calibration information along with job information

\*\* more detailed error messages; when thigns go wrong, should be easier to debug

\*\* more updated information

worry: applications are reproducible

answer is totally off

application developer persopective

too difficult to run on different hardware

better education material

qnexus

lower level api; interact with job service

higher level 'library':

eg. qiskit estimator: shot information

\*\* Group4

ZX calculus

qiskit's functions / library

• Which applications (chemistry, optimization, ML) are currently deployed on NISQ hardware vs FT prototypes?

will the interface change??

connect to HPC / latency requirements / workload characterization / does it matter?

targets will expand => material science / chemistry

have to distinguish against computation intensive vs data intensive

can't expect funding to last forever

have to be thinking about how to keep funding entities (never forget where you re going)

crazy science is cool but have to connect with 'reality'

focus on applications where we can't just throw more classical compute at them.

need easily accessible hardware to be in place for exploration

more expressive interfacing for hybrid HPC/QC programming

\*\*\*Group3

• What's the workload characterization of hybrid applications? High/low/equivalent quantum/hpc? And time scales?

• What are the 2 highest priority requirements for the Application View?

\* make quantum hardware better ;-)

\* set goal posts:

concentrate on portability

define interfaces to libraries that "useful stuff" (i.e., analogous to blas, etc)

• Current SDK and programming paradigms tightly integrate algorithms and runtime. Should we be moving towards standalone libraries (akin to BLAS), with no dependency on a specific framework

\*\* important

• How are teams anticipating changes to hybrid applications as we move from NISQ to FT? (e.g., latency, fidelity, coherence times, error rates)

- if changes required will be large; to what extent should be make things general now?

• How significant a concern is software framework lock-in (e.g., Qiskit, PennyLane) for application developers?

◦ As applications grow in scale and complexity, would they benefit from a modular, composable, interoperable, library‑centric ecosystem with minimal dependency chains?

• What changes do we expect when we go from the NISQ era to FT? How will the software reflect that?

◦ Mapping libraries from application domain → circuit

◦ Ease of development

• What are things which will not change from NISQ to FT?

◦ structure of what a quantum computational unit is?

◦ FT can have different gates, but the gate concept remains.

• Is quantum circuit the lowest level? Lower than the quantum circuit becomes the vendor domain?

◦ What's the lowest common interface

◦ FT can have different gates

◦ How services can evolve as systems scale

----

- what should the next steps be?

- (rigetti asked this) what will emerge as the “industry standard” lagnuage

Other:

1) methods with provable confidence bounds

e.g.

2) algorithm approaches with \*guarantees\* solutions bounds

e.g. variational algos

3) problems with efficient classical verification

peaked circuit sampling

optimization problems

want to test QA \*rigorously\*

- layers of abstractions; simplify development, potentially integration; impact on performance

should there be different “layers”/subsets that different players may support