Math Special Interest Group

Session 5

October 25, 2023

Agenda

- Welcoming remarks – 5 minutes
- UXL/Math SIG Mailing Lists - 5 minutes
- Updates from last meeting – 5 minutes
- Discussion on Discrete Fourier Transform APIs – Raphael Egan (30 minutes)
- Wrap-up and next steps – 5 minutes

We want to give the UXL/oneAPI community more shared resources.

- To make it easier to join SIGs and follow along with updates and communicate with one another between meetings
- To make it easier for the community to self-manage
- To make it easier for participants to manage their communications and preferences
- To make it easier for SIG leaders to manage groups and share the burden of responsibilities

Our Asks:

1) Sign up for the new mailing list distribution service for UXL / oneAPI SIGs at https://lists.uxlfoundation.org/g/main/subgroups

2) Create an LF ID at openprofile.dev in order to manage calendar invites and subscriptions across multiple LF projects

3) If you have any questions or issues, email info@uxlfoundation.org

We will cut over from the old email / event notification process to the new process on December 1.

Calendar notifications, discussions, and other information will be distributed via the new lists and on the GitHub repositories.

Sign up for the lists that interest you, and create an LF ID, to make sure you don't miss any news.

Updates from last meeting

- Seeking UXL Foundation Working Group Chair nominations
- oneAPI Spec:
  - Introduced oneMKL RNG Device API
  - Introduced value_or_pointer wrapper for BLAS USM scalar parameters
- Open source oneMKL interfaces updates:
  - In progress: RNG Device API to oneMKL interfaces (#382)
  - In progress: Sparse BLAS domain with MKLCPU backend (#374, #403)
  - Additional functionality supported and bugs fixed for various backends

Discussion on Discrete Fourier Transform APIs: configuring data layouts *by domain*

Outline

- Overview and introduction
  - Reminders about DFTs, jargon, and a fundamental property
  - Current configuration parameters for strides and batch distances in SYCL APIs of oneMKL
  - Examples, drawbacks and inconsistencies
- Upcoming changes
  - New configuration parameters and deprecations
  - SYCL example for batched 2D real transforms with targeted changes
  - More information and specification changes

OVERVIEW AND INTRODUCTION

Reminders about DFTs, jargon, and a fundamental property

The Discrete Fourier Transform (DFT) of a d-dimensional $n_{d-1} \times n_{d-2} \times \cdots \times n_0$ periodic sequence $\{x\} \triangleq x_{j_{d-1},\dots,j_0}, 0 \leq j_i < n_i, \forall i \in \{0, \dots, d-1\}$ is a periodic sequence $\{y\} \triangleq y_{k_{d-1},\dots,k_0}, 0 \leq k_i < n_i, \forall i \in \{0, \dots, d-1\}$ defined as

$$y_{k_{d-1},\dots,k_0} = \sum_{j_{d-1}=0}^{n_{d-1}-1} \sum_{j_{d-2}=0}^{n_{d-2}-1} \cdots \sum_{j_0=0}^{n_0-1} x_{j_{d-1},j_{d-2},\dots,j_0}\, e^{\delta 2\imath\pi\left(\sum_{\ell=0}^{d-1}\frac{j_\ell k_\ell}{n_\ell}\right)}, \imath^2 = -1.$$

- We call "**forward**" DFT the case with $\delta = -1$, referred to as $\{y\} = \mathrm{DFT}(\{x\})$.
- We call "**backward**" (or "inverse") DFT the case with $\delta = +1$, referred to as $\{y\} = \mathrm{iDFT}(\{x\})$.
- We refer to the input (resp. output) domain of forward (resp. backward) DFT as "forward domain".
- We refer to the output (resp. input) domain of forward (resp. backward) DFT as "backward domain".
- For any d-dimensional periodic sequence, one has the *fundamental roundtrip identity*

$$\frac{1}{\prod_{\ell=0}^{d-1} n_\ell}\mathrm{iDFT}\big(\mathrm{DFT}(\{x\})\big) = \{x\}.$$

- If $\{x\} \in \mathbb{R}^d$, $\{y\} = \mathrm{DFT}(\{x\})$ is a set of complex-conjugate values, *i.e.*, $\left(y_{k_{d-1},\dots,k_0}\right)^* = y_{n_{d-1}-k_{d-1},\dots,n_0-k_0}$

  → Only roughly half the backward domain's data is computed and/or stored in memory.

Current configuration parameters for strides and batch distances in SYCL APIs of oneMKL

Configuration setting	oneMKL SYCL APIs

Batch distances
(=number of elements, of implicitly-assumed types, between two successive, independent data sets)
	`dft::config_param::FWD_DISTANCE`
	`dft::config_param::BWD_DISTANCE`
	(set *by domain*)

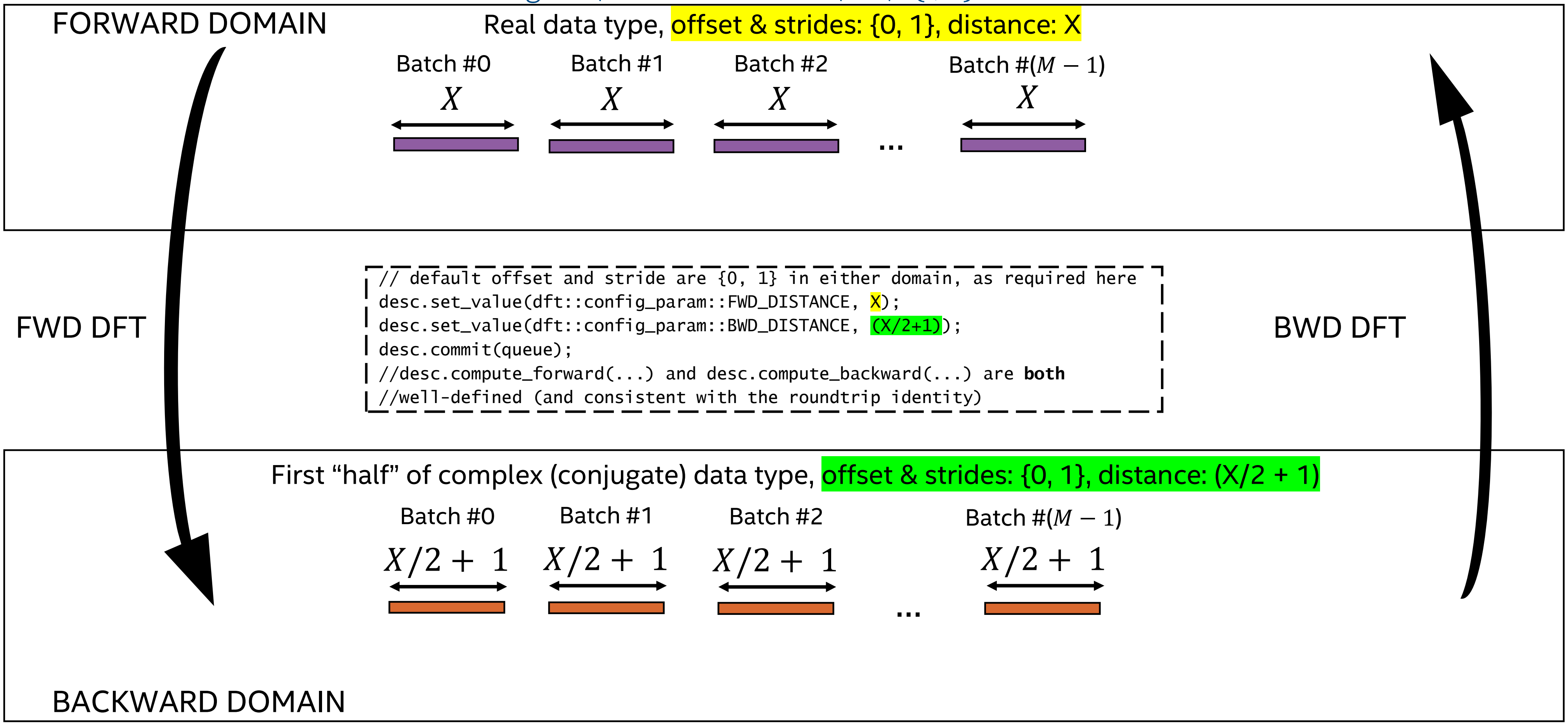
Strides
(=number of elements, of implicitly-assumed types, between two successive entries along a dimension of a given data set)
	`dft::config_param::INPUT_STRIDES`
	`dft::config_param::OUTPUT_STRIDES`
	(set by input/output)

Note:

- SYCL APIs *partially* departed from the paradigm of classic C and Fortran APIs by defining the distances by domain while strides are still to be defined by I/O. (classic APIs require strides and distances by I/O);
- Setting distances by domain makes 1D real descriptors of default unit stride capable of both compute directions;
- Strides were omitted in the design the SYCL APIs in the first place (oversight), so multi-dimensional real transforms can't handle both compute directions, resulting in
  - possible undefined behavior if invoking the ill-configured compute direction;
  - requirement to re-commit or use two separate descriptors;
  - possible commit-time overheads and larger-than-necessary memory footprint.

Examples, drawbacks and inconsistencies

Consider 1D real transforms of length X, batched M times, *i.e.*, $\mathrm{r\{f,b\}o}X * M$ via SYCL APIs



FORWARD DOMAIN

Real data type, offset & strides: {0, 1}, distance: X

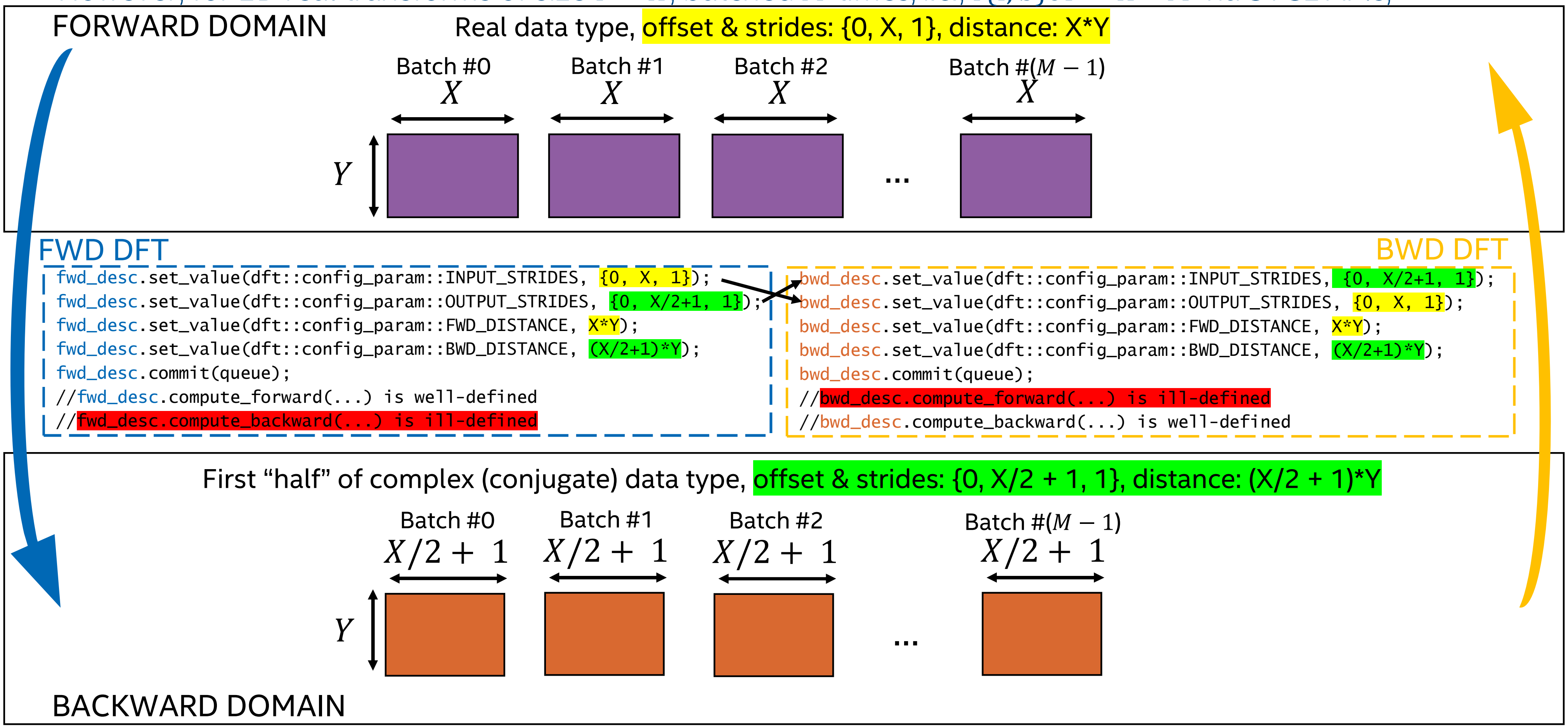
```
// default offset and stride are {0, 1} in either domain, as required here
desc.set_value(dft::config_param::FWD_DISTANCE, X);
desc.set_value(dft::config_param::BWD_DISTANCE, (X/2+1));
desc.commit(queue);
//desc.compute_forward(...) and desc.compute_backward(...) are both
//well-defined (and consistent with the roundtrip identity)
```

First “half” of complex (conjugate) data type, offset & strides: {0, 1}, distance: (X/2 + 1)

BACKWARD DOMAIN

Examples, drawbacks and inconsistencies

However, for 2D real transforms of size $Y \times X$, batched M times, *i.e.*, $\mathrm{r\{f, b\}o}Y \times X * M$ via SYCL APIs,



FORWARD DOMAIN

Real data type, offset & strides: {0, X, 1}, distance: X*Y

FWD DFT

```
fwd_desc.set_value(dft::config_param::INPUT_STRIDES, {0, X, 1});
fwd_desc.set_value(dft::config_param::OUTPUT_STRIDES, {0, X/2+1, 1});
fwd_desc.set_value(dft::config_param::FWD_DISTANCE, X*Y);
fwd_desc.set_value(dft::config_param::BWD_DISTANCE, (X/2+1)*Y);
fwd_desc.commit(queue);
//fwd_desc.compute_forward(...) is well-defined
//fwd_desc.compute_backward(...) is ill-defined
```

BWD DFT

```
bwd_desc.set_value(dft::config_param::INPUT_STRIDES, {0, X/2+1, 1});
bwd_desc.set_value(dft::config_param::OUTPUT_STRIDES, {0, X, 1});
bwd_desc.set_value(dft::config_param::FWD_DISTANCE, X*Y);
bwd_desc.set_value(dft::config_param::BWD_DISTANCE, (X/2+1)*Y);
bwd_desc.commit(queue);
//bwd_desc.compute_forward(...) is ill-defined
//bwd_desc.compute_backward(...) is well-defined
```

First "half" of complex (conjugate) data type, offset & strides: {0, X/2 + 1, 1}, distance: (X/2 + 1)*Y

BACKWARD DOMAIN

Examples, drawbacks and inconsistencies

In conclusion, any design requiring data layout configuration parameter(s) to be set for input/output data (instead of defining layouts “by domain”) faces the following, non-exhaustive, list of bottlenecks:

- impossibility to build descriptors with default sets of parameters for cases requiring different parameters in either domain as the default would depend on the intended compute direction which is unknown until compute time (affects real multi-dimensional cases, for instance);
- possibly significant commit-time overheads for any application using non-default strides;
- larger than strictly necessary library memory footprint for internal/external workspace (if using descriptor duplication to avoid frequent re-committing in applications);
- undefined behavior for one of the two compute directions;
- …

Currently, these drawbacks *affect all oneMKL SYCL users* interested in applications using different strides in either domain, *e.g.*,

- complex transform of any dimensionality using unusual (non-default) strides;
- practically any real multi-dimensional transform (including those of most conventional data layouts);
- real 1D transforms with non-unit strides.

UPCOMING CHANGES

New configuration parameters and deprecations

Configuration setting	oneMKL SYCL APIs

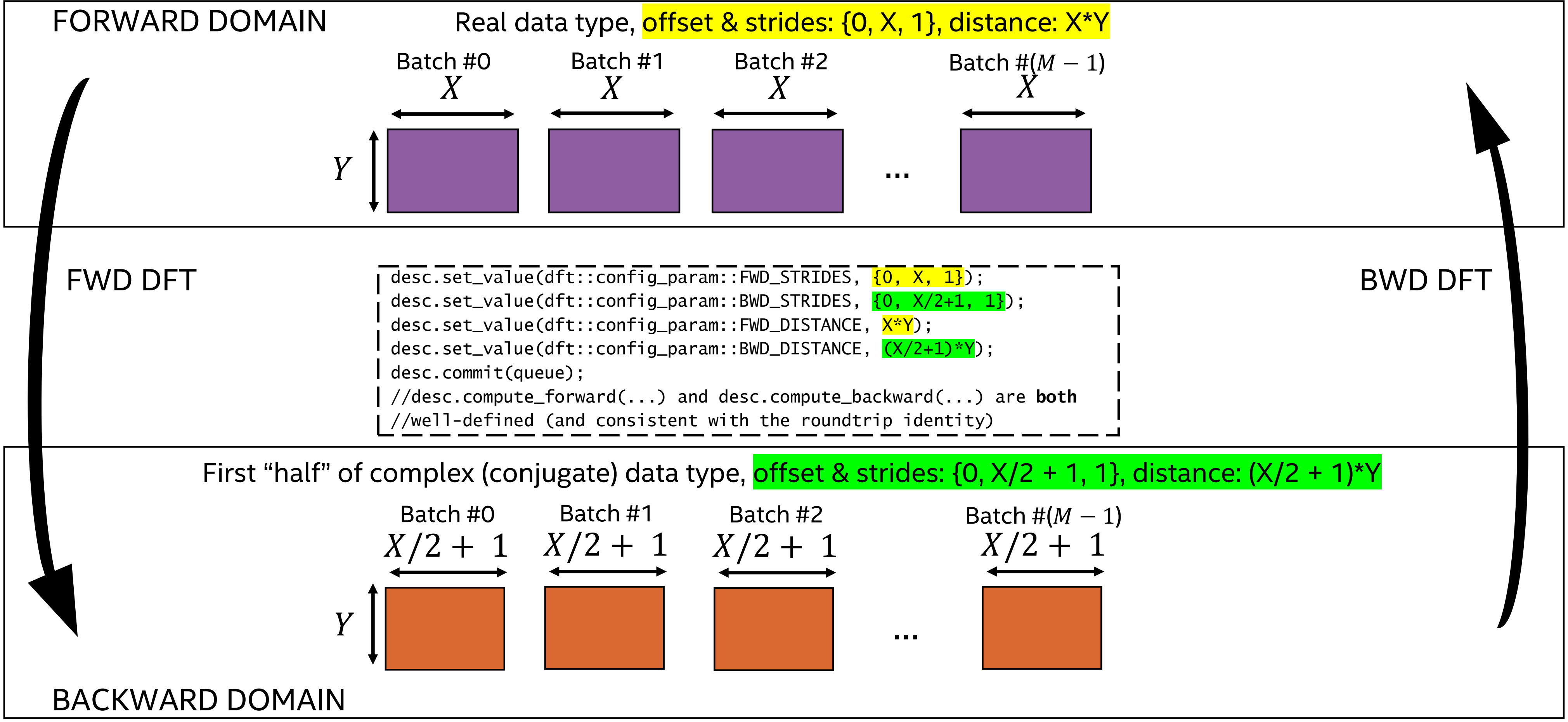
Batch distances
(=number of elements, of implicitly-assumed types, between two successive, independent data sets)
	`dft::config_param::FWD_DISTANCE`
	`dft::config_param::BWD_DISTANCE`
	(unchanged, set *by domain*)

Strides
(=number of elements, of implicitly-assumed types, between two successive entries along a dimension of a given data set)
	~~`dft::config_param::INPUT_STRIDES`~~
	~~`dft::config_param::OUTPUT_STRIDES`~~
	`dft::config_param::FWD_STRIDES`
	`dft::config_param::BWD_STRIDES`
	(~~set for I/O~~ → set *by domain*)

- The struck-through parameters will enter a *deprecation period and trigger warnings*, if used.
- Mix-n-matching new and legacy parameters would be forbidden.
- SYCL descriptors for real multi-dimensional DFT will be default-initialized unambiguously upon completion (configured as needed for in-place, unbatched transforms consistently with other relevant default parameters).

SYCL example for batched 2D real transforms with targeted changes

For 2D real transforms of size $Y \times X$, batched M times, *i.e.*, $\mathrm{r\{f,b\}o}Y \times X * M$ via SYCL APIs,



FORWARD DOMAIN

Real data type, offset & strides: {0, X, 1}, distance: X*Y

FWD DFT

BWD DFT

```
desc.set_value(dft::config_param::FWD_STRIDES, {0, X, 1});
desc.set_value(dft::config_param::BWD_STRIDES, {0, X/2+1, 1});
desc.set_value(dft::config_param::FWD_DISTANCE, X*Y);
desc.set_value(dft::config_param::BWD_DISTANCE, (X/2+1)*Y);
desc.commit(queue);
//desc.compute_forward(...) and desc.compute_backward(...) are both
//well-defined (and consistent with the roundtrip identity)
```

First “half” of complex (conjugate) data type, offset & strides: {0, X/2 + 1, 1}, distance: (X/2 + 1)*Y

BACKWARD DOMAIN

More information and specification changes

- Current effort to update the oneMKL DFT specs: PR#506.
- Targeted for oneAPI Specification version 1.3 (cutoff for all edits is 2023/11/02).

Wrap-up

Next Steps

- Focuses for next meeting(s):
  - Any topics from Math SIG members?

- Please feel free to extend invitations to others to join Math SIG
- If anyone has content that they would like posted on oneAPI.io, please let us know

Resources

- Unified Acceleration Foundation: https://uxlfoundation.org/
- oneAPI Initiative: https://www.oneapi.io/
- Latest release of oneMKL Spec (currently v. 1.2; 1.3 provisional also available): https://spec.oneapi.com/versions/latest/elements/oneMKL/source/index.html
- GitHub for oneAPI Spec: https://github.com/oneapi-src/oneAPI-spec
- GitHub for oneAPI Community Forum: https://github.com/oneapi-src/oneAPI-tab
- GitHub for open source oneMKL interfaces (currently BLAS, RNG, LAPACK, and DFT domains): https://github.com/oneapi-src/oneMKL