Until UCC is Here - UCG Status Update

Alex Margolin

UCF Annual Workshop, December 2020

HUAWEI

Outline

1. UCG status (and how did we get to it)

2. UCG in the software stack

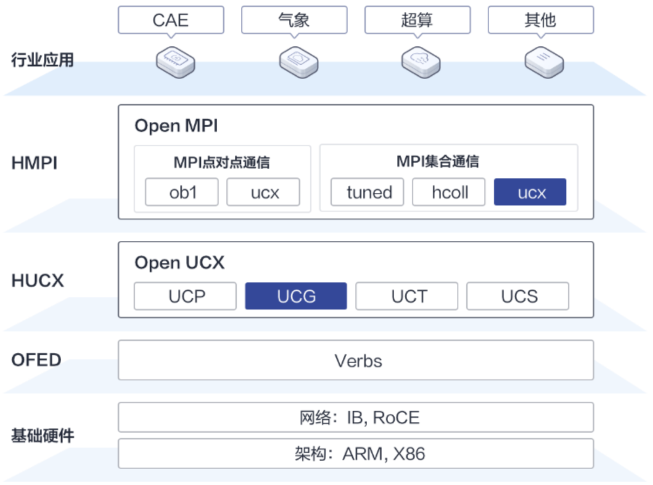
3. UCG & UCC – where is this going?

4. Huawei’s roadmap for collective operations

A Brief history of UCG (*G for Groups*) and UCC

- [Aug. 2018] UCG Started (just me, soon after joining Huawei)
- [Dec. 2018] UCG API submitted for upstreaming (~~#3091~~#3602)
- [Nov. 2019] Talks about upstreaming UCG (#4545) – didn't work out…
- [Dec. 2019] UCF formed the UCC working-group
- [Sep. 2020] Huawei Cloud officially publishes its

  "High-Performance Communication Library"
- [Sep. 2020] UCC's external API finalized (#1)
- [Dec. 2020] … this update.



*__Any__ UCX version?! Yes. See next slide.

Current Status

Right now - there are too many versions!

~~1. my personal UCG repo~~ (deprecated, finally ☺)

2. xUCG (OpenUCX github)

   - This is my "master" branch – should fit **ANY*** UCX version, not 100% stable…

3. Huawei's xUCG (Huawei's github)

   - Recently created by the team in China – assumes UCX v1.6, **very stable**

4. Huawei's internal UCG (Steady release schedule - within Hyper-MPI)

   - Includes some proprietary extensions, but mostly just experimental code

***Any** UCX version?! Yes. See next slide.

UCG Today

Supports:

- a range of collectives (Bcast, Reduce, Allreduce, Barrier, Scatter, Gather, Allgather),
- any datatype Open-MPI does,
- any transports UCT does,

~~• any protocol UCP does~~, (not even close… but we’re working on it!)

- various hardware platforms.

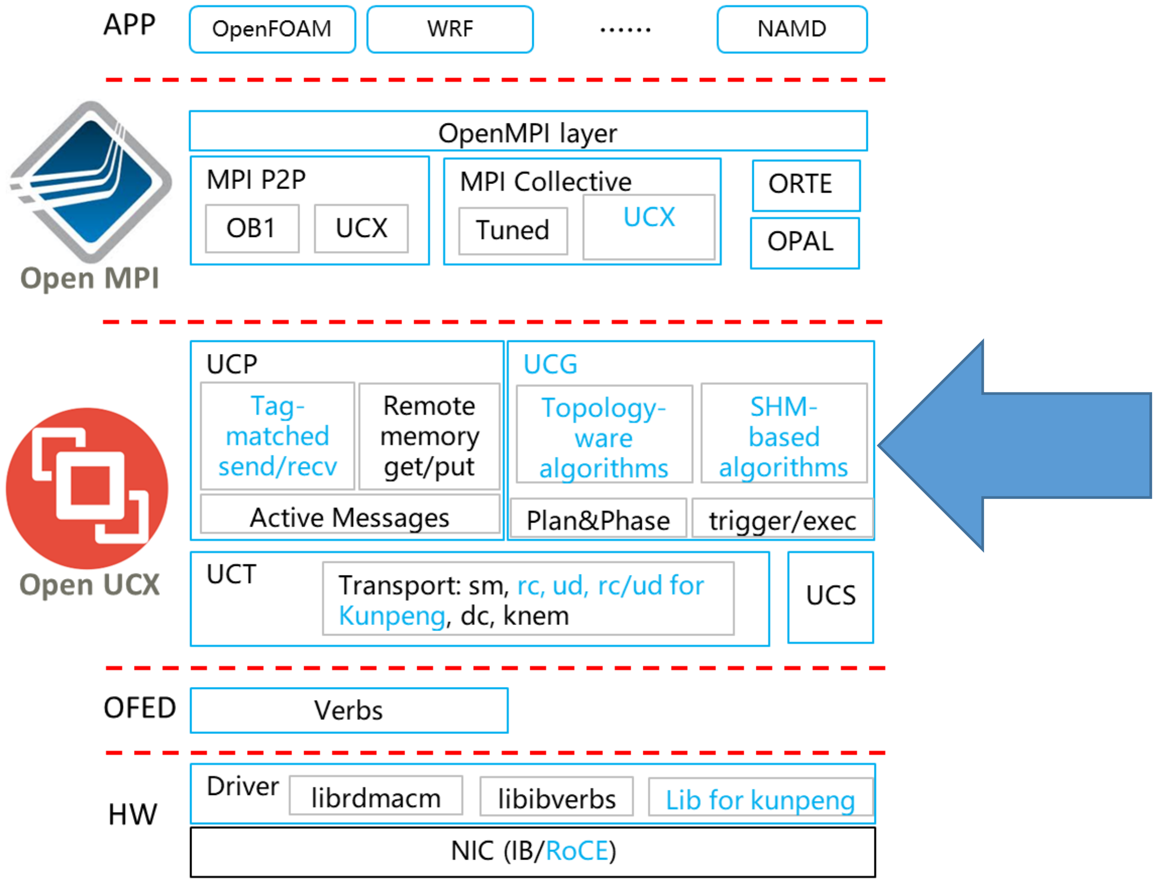
Also: tested up to 256 (x86) nodes and up to 256 (ARM) cores on a host.

Outline

1. UCG status (and how did we get to it)
2. <u>UCG in the software stack</u>
3. UCG & UCC – where is this going?
4. Huawei’s roadmap for collective operations

TEL AVIV
RESEARCH
CENTER

UCG in the software stack



Making UCG work with ANY UCX version

Challenge*: keep using UCX internal APIs (UCS / UCT) as it evolves

Incident #1 - [Change-ID 8da6a5be2e](#):

```
- typedef void (*uct_completion_callback_t)(uct_completion_t *self,
-                                           ucs_status_t status);
+ typedef void (*uct_completion_callback_t)(uct_completion_t *self);
```

Incident #2 - [Change-ID fca960826a](#):

```
- #define UCS_CONFIG_REGISTER_TABLE_ENTRY(_entry) \
+ #define UCS_CONFIG_REGISTER_TABLE_ENTRY(_entry, _list) \
...
```

Making UCG work with ANY UCX version

Challenge*: keep using UCX internal APIs (UCS / UCT) as it evolves

*Not as hard as it sounds, actually.

Incident #1 - Change-ID 8da6a5be2e:

```
AC_COMPILE_IFELSE([AC_LANG_PROGRAM([[#include "uct/api/uct.h"]],
                                   [[uct_completion_callback_t func = NULL;]
                                   [func(NULL, UCS_OK);]])],...
```

Incident #2 - Change-ID fca960826a:

```
AC_COMPILE_IFELSE([AC_LANG_PROGRAM([[#include "ucs/config/parser.h"]],
                                   [[#undef UCS_CONFIG_REGISTER_TABLE_ENTRY]
                                    [#define UCS_CONFIG_REGISTER_TABLE_ENTRY(a, b)]
                                    [UCS_CONFIG_REGISTER_TABLE(NULL, NULL, NULL, ...
```

APIs

- "Northbound":
  - Generic (src/ucg/api/ucg.h): ucg_group_create, ucg_collective_create
  - MPI-specific (src/ucg/api/ucg_mpi.h): MPI_Reduce(),
  - Versioning (src/ucg/api/ucg_version.h): similar to ucp_version.h

- "Southbound":
  - Plan Components (src/ucg/api/ucg_plan_components.h):

    - for accommodating multiple implementations (similar to OMPI's MCA COLL)
  - Expected components:"builtin" (day-1), hicoll, NCCL?

Northbound API, side-by-side (Blocking)

MPI Application (or MPI-specific API)	UCX Groups (UCG) Equivalent (Generic API)

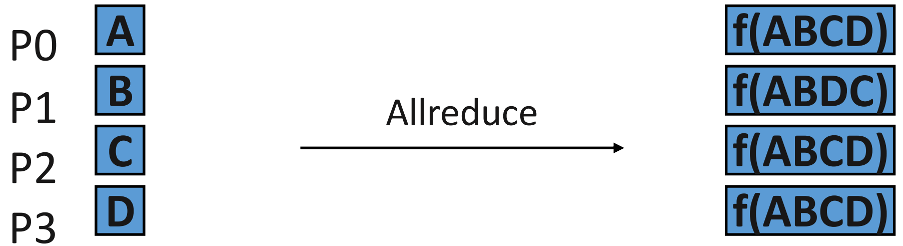
`MPI_Bcast(bufA,...,rowcomm)`	`paramsA.buf = bufA;`
	`paramsA.type.modifiers = BCAST_MODIFIERS;`
	`...`
	
	`ucg_collective_create(rowGroup, &paramsA, &collA);`
	`ucg_collective_start_nb(collA, &reqA);`
	
	`MPI_Wait(reqA, &status);`

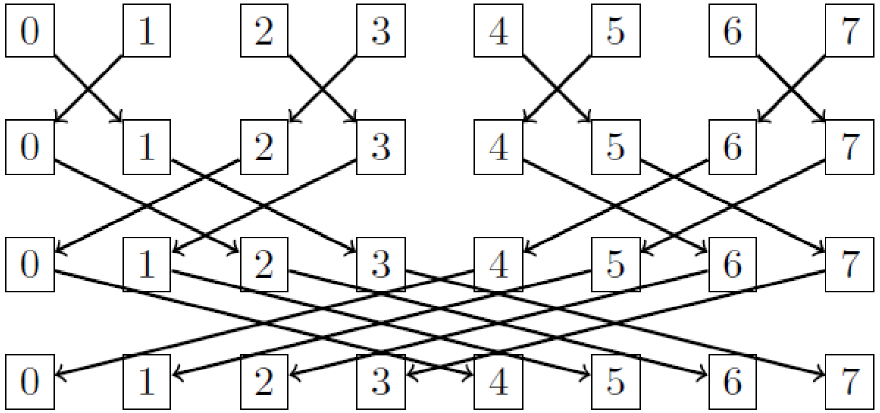
Northbound API, side-by-side (Non-blocking)

MPI Application (or MPI-specific API)	UCX Groups (UCG) Equivalent (Generic API)
`MPI_Ibcast_init(..., &req);`	`ucg_collective_create(Group, ..., &coll);`
`for (i=0; i<MAXITER; i++) {`	`for (i=0; i<MAXITER; i++) {`
`    compute(buf);`	`    compute(buf);`
`    MPI_Start(req);`	`    ucg_collective_start_nb(coll, &req);`
`    MPI_Wait(req, &status);`	`    MPI_Wait(req, &status);`
	`}`
	`ucg_request_free(req);`
`}`	
`MPI_Request_free(req);`	

TEL AVIV RESEARCH CENTER

Decoupling Planning and Execution – Recursive Doubling





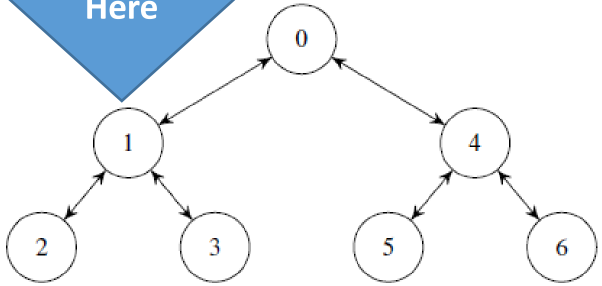
Phase #	Ranks	Action
1	0	Send a message and also wait for one
2	3	Send a message and also wait for one
3	5	Send a message and also wait for one

Decoupling Planning and Execution – Tree-based

Phase #	Ranks	Action
1	2,3	Wait for a message
2	0	Send a message
3	0	Wait for a message
4	2,3	Send a message

Phase #	Ranks	Parameters
1	2,3	**Action:** Wait for a message from each node **Incoming buffer:** temporary buffer of 24 bytes which starts with the user's input **Incoming length:** 16 bytes **Incoming offset:** 8 bytes
2	0	**Action:** Send a message to each node **Outgoing buffer:** the temporary buffer from the previous step **Outgoing length:** 24 bytes **Outgoing offset:** 0 bytes

You Are Here



8-byte call to MPI_Gather()

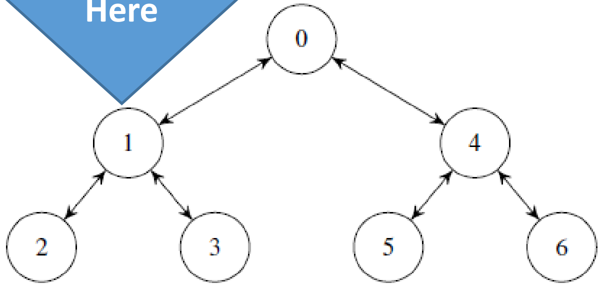
*Is tree the best choice for MPI_Gather? Not always…

Decoupling Planning and Execution – Tree-based

Phase #	Ranks	Action
1	2,3	Wait for a message
2	0	Send a message
3	0	Wait for a message
4	2,3	Send a message

Phase #	Ranks	Parameters
3	0	**Action:** Wait for a message from each node **Incoming buffer:** destination buffer **Incoming length:** 8 bytes **Incoming offset:** 0 bytes
4	2,3	**Action:** Send a message to each node **Outgoing buffer:** destination buffer **Outgoing length:** 8 bytes **Outgoing offset:** 0 bytes

You Are Here



8-byte call to MPI_Bcast()

Outline

1. UCG status (and how did we get to it)
2. UCG in the software stack
3. <u>UCG & UCC – where is this going?</u>
4. Huawei's roadmap for collective operations

My point of view on UCC

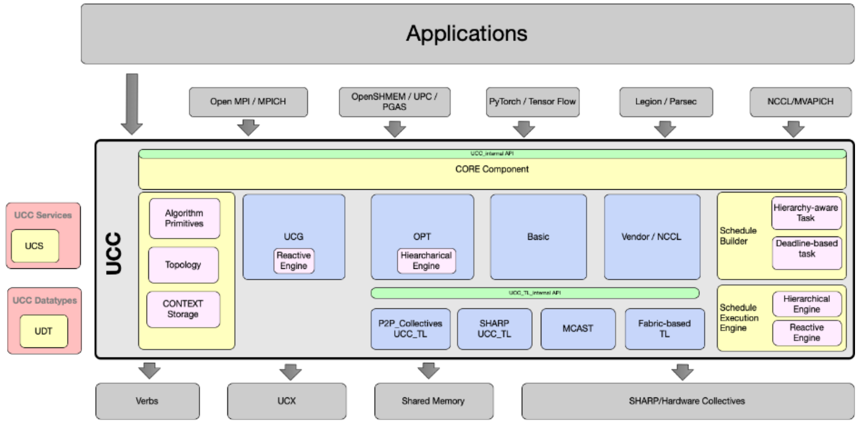
UCC is coming about, slowly – it'll be a while before we can use it, and…

My point of view on UCC

UCC is coming about, slowly – it’ll be a while before we can use it, and…

UCC looks *to me* like a big performance risk!

- UCC adds levels of abstraction, possibly returning issues from Open-MPI
  - Example #1: UCC is limited to UCP API… how/when to use atomics for reduction?
  - Example #2: progress is not aware whether SHARP is used at the moment.



My point of view on UCC

UCC is coming about, slowly – it'll be a while before we can use it, and…

UCC looks *to me* like a big performance risk!

- UCC adds levels of abstraction, possibly returning issues from Open-MPI
  - Example #1: UCC is limited to UCP API… how/when to use atomics for reduction?
  - Example #2: progress is not aware whether SHARP is used at the moment.
- Development is "breadth-fist" instead of "depth-first"
  - Preliminary performance results are not expected any time soon.
- UCC is practically just two participants: Mellanox and Huawei.
  - Missing representation from other vendors and potential users.

Challenges, looking forward

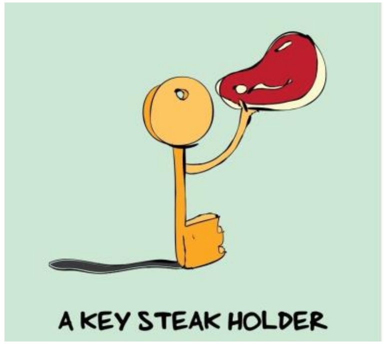
1. Must show the approach is beneficial - before investing much effort

   - I don't think xCCL's or xUCG's individual performance is proof enough…

Challenges, looking forward

1. Must show the approach is beneficial - before investing much effort
   - I don't think xCCL's or xUCG's individual performance is proof enough…

2. Community involvement:
   - More contributing parties
   - More representation for use-cases
   - More exposure, feedback, stakeholders…



Challenges, looking forward

1. Must show the approach is beneficial - before investing much effort
   - I don't think xCCL's or xUCG's individual performance is proof enough…

2. Community involvement:
   - More contributing parties
   - More representation for use-cases
   - More exposure, feedback, stakeholders…

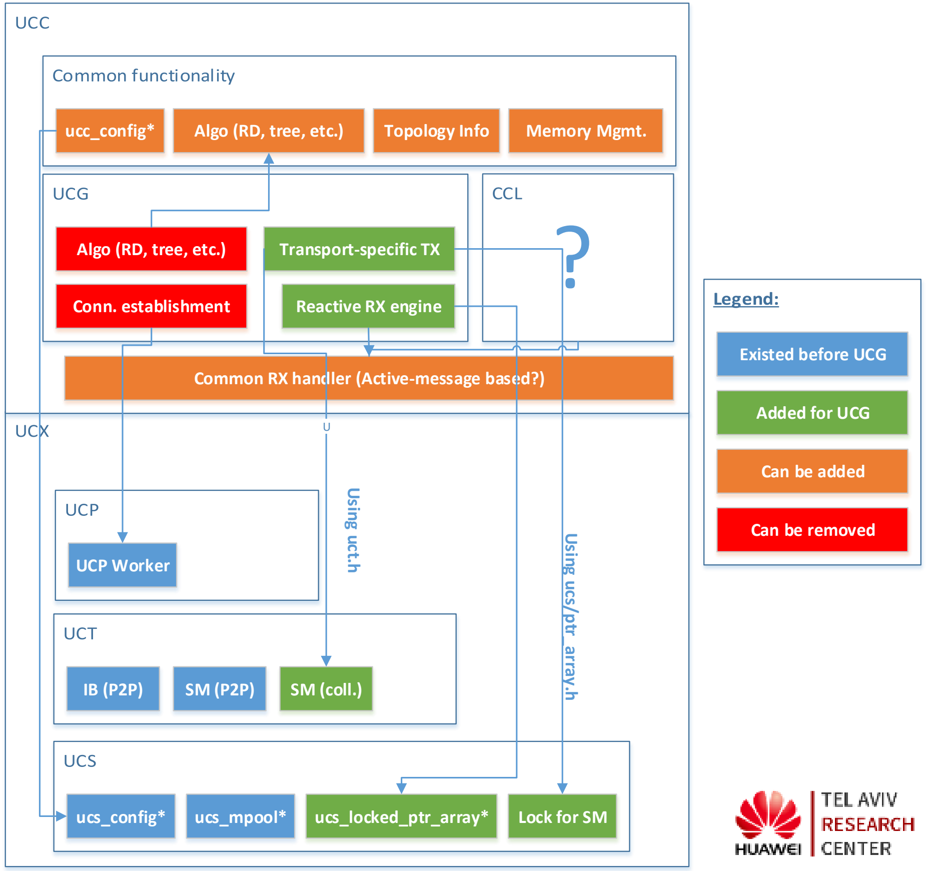
3. Delivering relevant, high-quality software.

UCG

Not going anywhere, anytime soon…
no open-source alternative (yet)

<u>3 possible paths</u>:

1. Merge into UCC
2. Merge into [Huawei's UCX package]
3. Merge into upstream UCX



TEL AVIV RESEARCH CENTER

Outline

1. UCG status (and how did we get to it)
2. UCG in the software stack
3. UCG & UCC – where is this going?
4. <u>Huawei’s roadmap for collective operations</u>

Development Roadmap

- Hardware-specific capabilities
    - Kunpeng CPU
    - Storage product-lines
    - Atlas AI accelerator product-lines

- Integration with related HPC Software
    - HPC Storage and parallel filesystems, e.g. MPI I/O
    - Batch scheduler, e.g. *SLURM* (for job information)
    - Builders, e.g. *Spack* (provide collectives for other packages)

- Custom acceleration hints (through MPI?)
    - Persistent operations
    - Read-only / write-only buffers

HUAWEI

www.huawei.com