QCD ON THE MODULAR SUPERCOMPUTER

June 17, 2019 | Eric B. Gregory | JSC

OVERVIEW

- Modular computing —
  one vision of supercomputing for the (near?) future
- Modular SC at Jülich Supercomputing Centre
- How can we arrange a LQCD simulation to exploit a modular computer?
- QMOD: a toy project based on USQCD software.

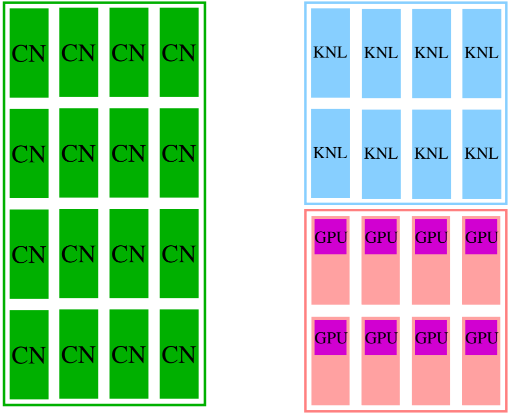
SOME QUESTIONS...

Modular supercomputing:

- What is it?
- Why do it?
- How to use it?

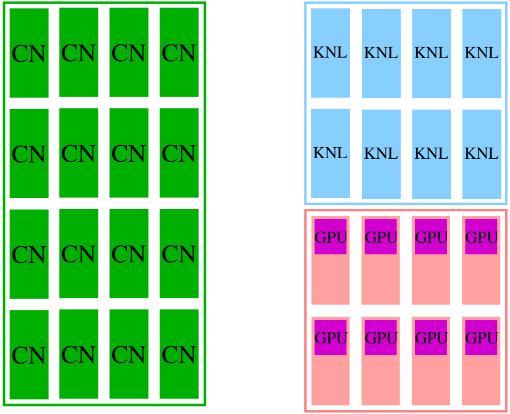
Part I: What is Modular Supercomputing?

YOUR NEIGHBORHOOD SUPERCOMPUTING CENTER



Maybe you have a choice of architectures.

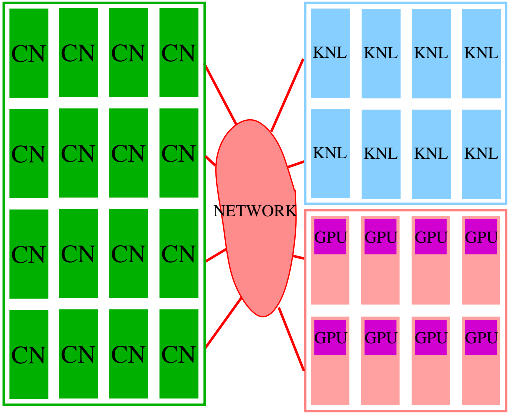
YOUR NEIGHBORHOOD SUPERCOMPUTING CENTER



Maybe you have a choice of architectures.
Run application on the hardware:

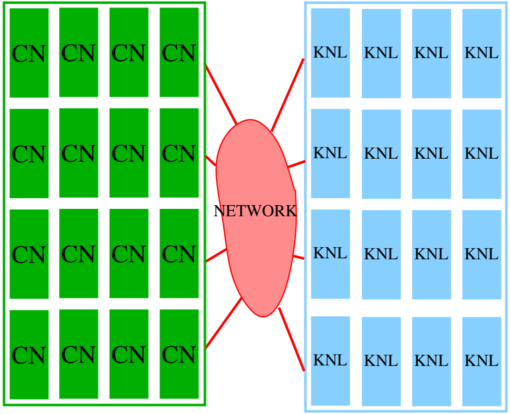
- on which it performs best
- for which it has been built
  (pre-installed packages, legacy codes,..)

MODULAR SUPERCOMPUTING



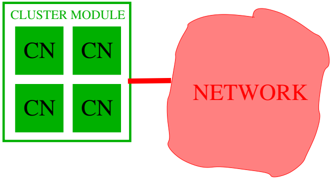
Why choose?

JURECA @ JSC



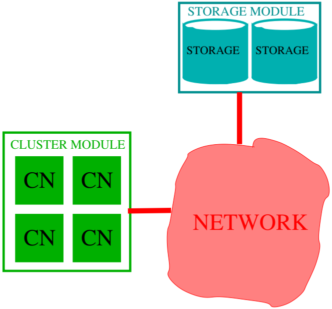
Working modular system at Jülich.

GENERIC MODULAR SUPERCOMPUTER



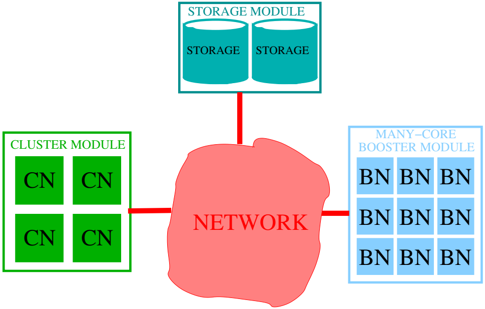
Development of a prototype modular supercomputer.

GENERIC MODULAR SUPERCOMPUTER



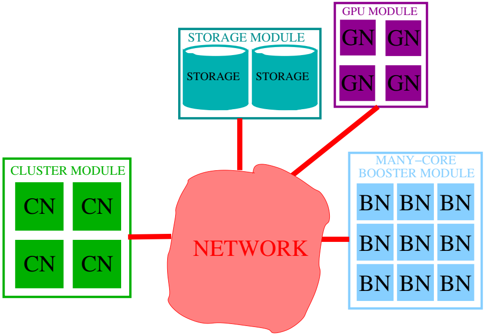
Development of a prototype modular supercomputer.

GENERIC MODULAR SUPERCOMPUTER



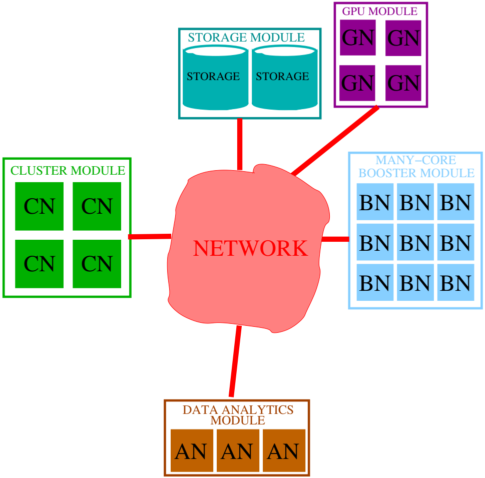
Development of a prototype modular supercomputer.

GENERIC MODULAR SUPERCOMPUTER



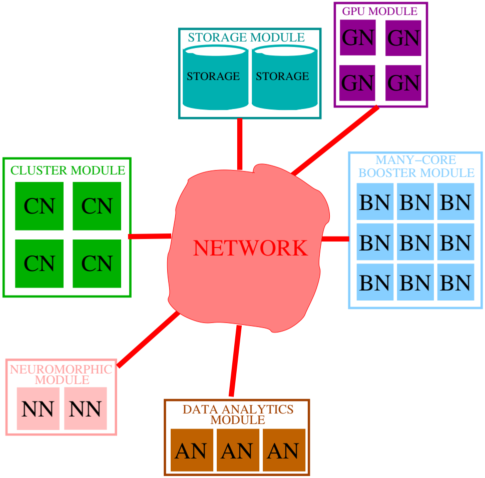
Development of a prototype modular supercomputer.

GENERIC MODULAR SUPERCOMPUTER



Development of a prototype modular supercomputer.

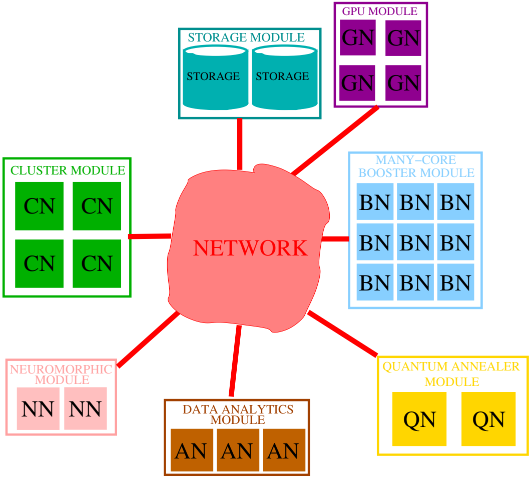
GENERIC MODULAR SUPERCOMPUTER



Development of a prototype modular supercomputer.
See:

- DEEP-EST project
  `https://www.deep-projects.eu/`
- Planned expansion of JSC's *JUWELS* system.

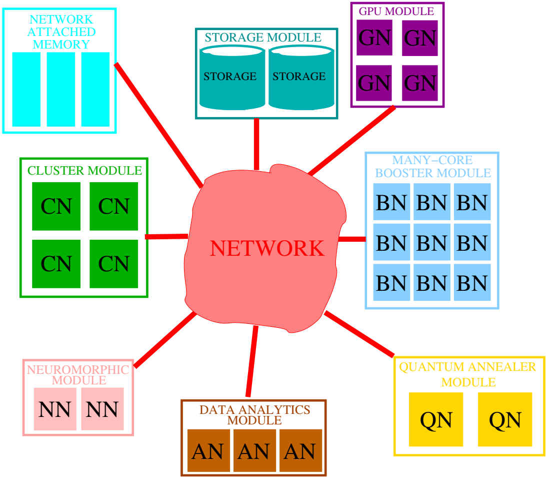
GENERIC MODULAR SUPERCOMPUTER



Development of a prototype modular supercomputer.
See:

- DEEP-EST project
  `https://www.deep-projects.eu/`
- Planned expansion of JSC's *JUWELS* system.

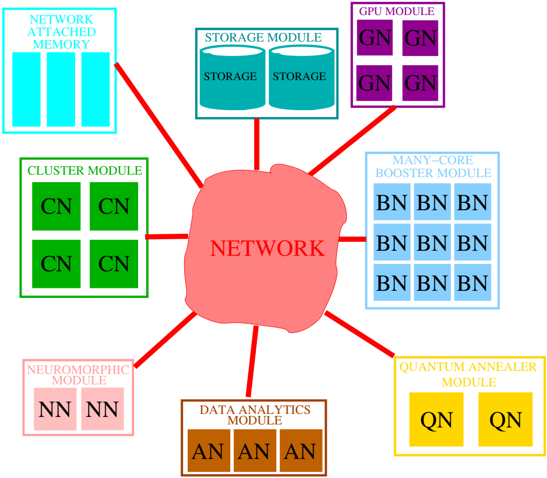
GENERIC MODULAR SUPERCOMPUTER



Development of a prototype modular supercomputer.
See:

- DEEP-EST project
  `https://www.deep-projects.eu/`
- Planned expansion of JSC's *JUWELS* system.

GENERIC MODULAR SUPERCOMPUTER

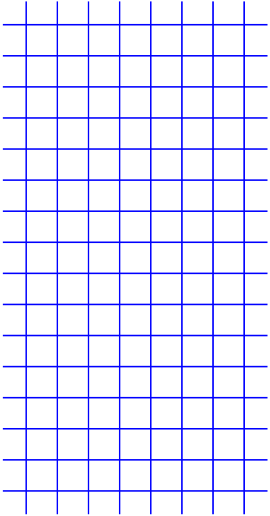


Themes:

- Heterogeneous architecture
- Flexible:
  user chooses hardware mix
- Mix changes during the run?
- Dis-aggregated hardware systems

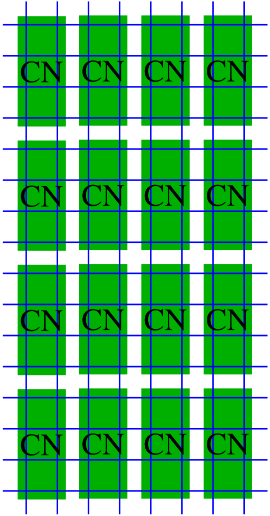
Part II: QCD and the modular supercomputer

IS QCD SUITABLE FOR A MODULAR SYSTEM?



- QCD is a *very* homogeneous problem

DOES QCD NEED A MODULAR SYSTEM?



- QCD is a *very* homogeneous problem
- Lots of room for tasks, threads, SIMD lanes to do the same operation on different parts of the data.

LQCD & MODULAR COMPUTING

How could we use a modular system for a LQCD simulation?

- Speculative — what hardware will be available in 5-10 years?
- ASSUME: network between modules is sufficiently fast.

LQCD & MODULAR COMPUTING

How could we use a modular system for a LQCD simulation?

- Speculative — what hardware will be available in 5-10 years?
- ASSUME: network between modules is sufficiently fast.

Useful exercise

- inspire consideration of any special hardware wish lists for calculation elements
- Identify further concurrencies to exploit on contemporary machines
- Look for tasks where strict ordering is not necessary.

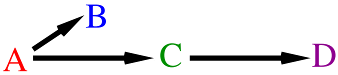
LQCD & MODULAR COMPUTING

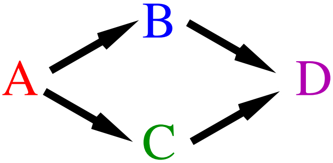
How could we use a modular system for a LQCD simulation?

- Speculative — what hardware will be available in 5-10 years?
- ASSUME: network between modules is sufficiently fast.

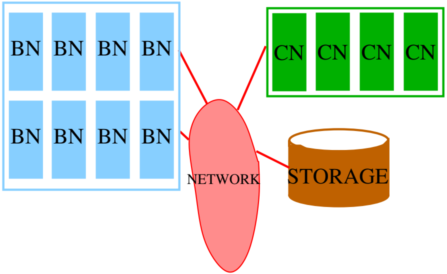
Useful exercise

- inspire consideration of any special hardware wish lists for calculation elements
- Identify further concurrencies to exploit on contemporary machines
- Look for tasks where strict ordering is not necessary.



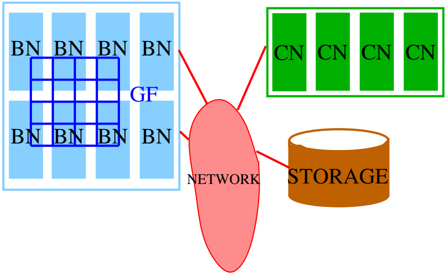


EXAMPLE 1: I/O



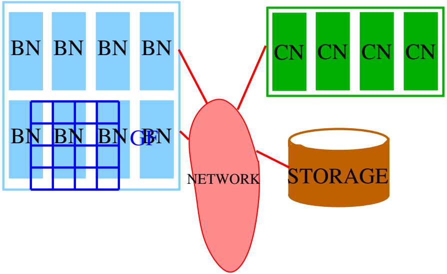
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



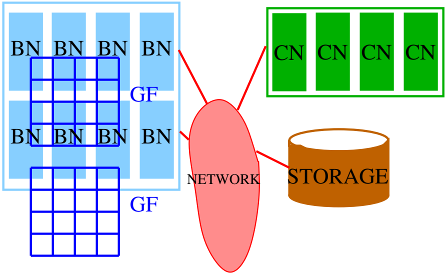
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



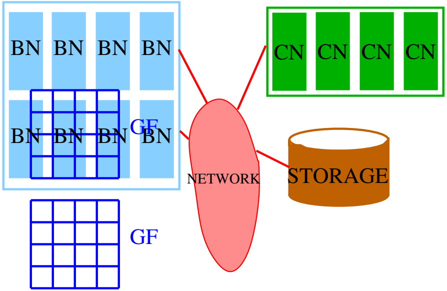
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



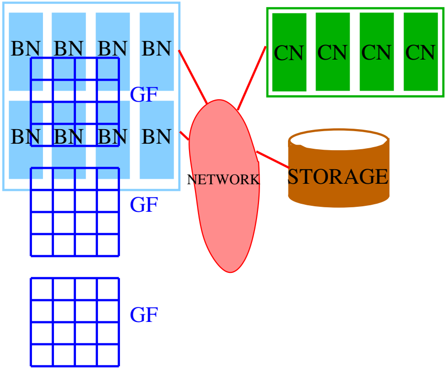
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



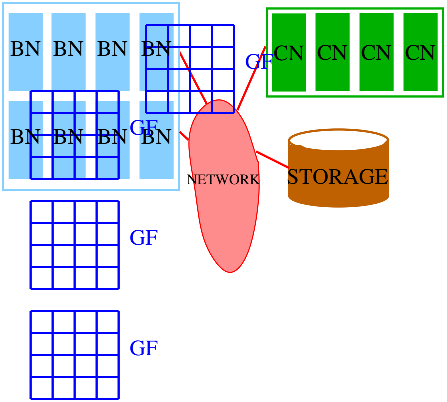
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



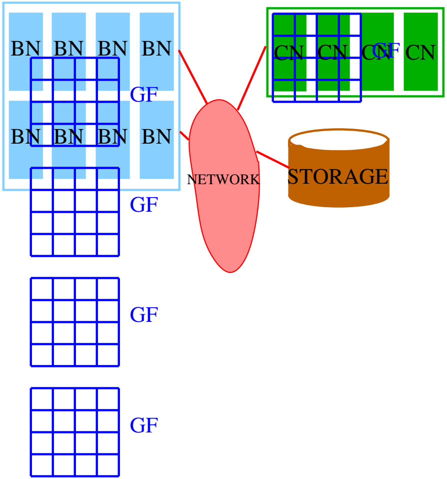
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



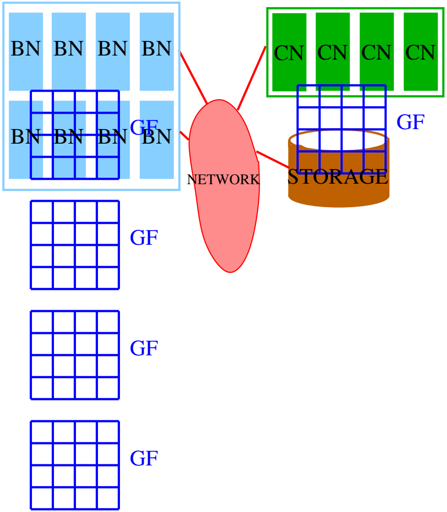
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



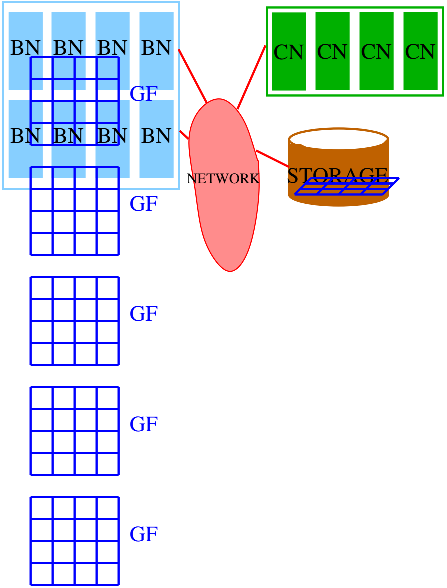
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



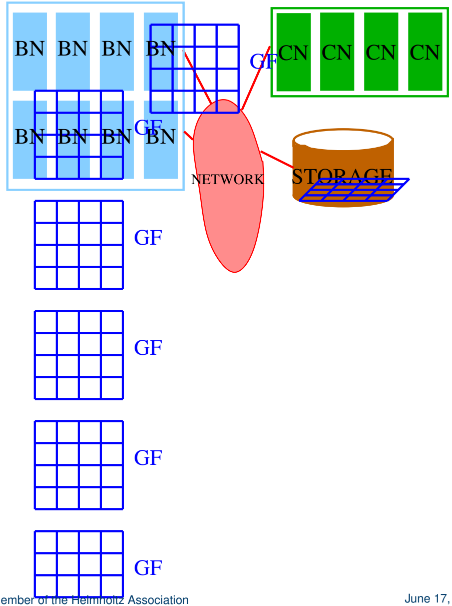
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



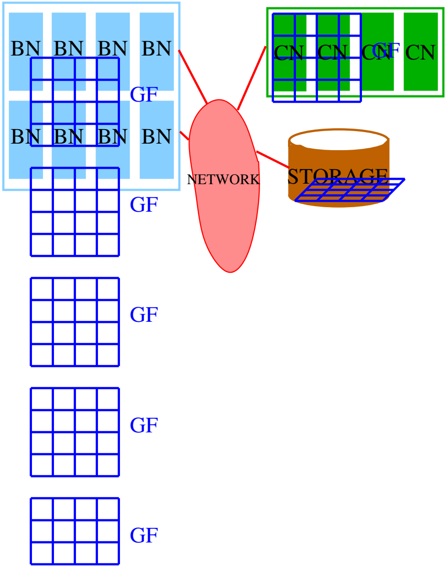
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



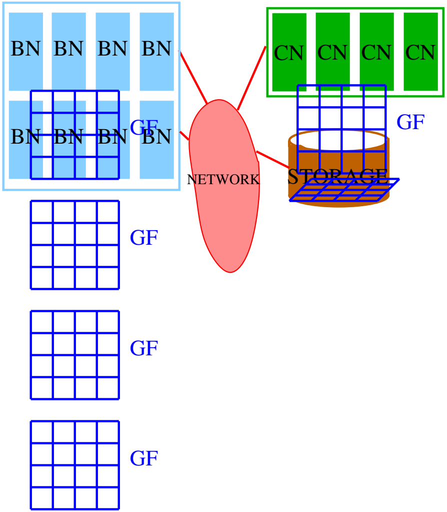
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



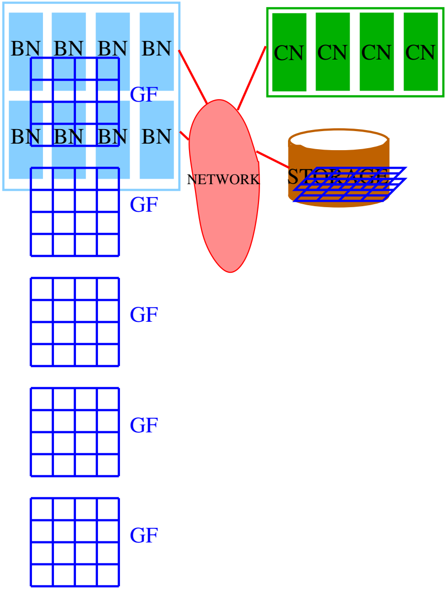
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



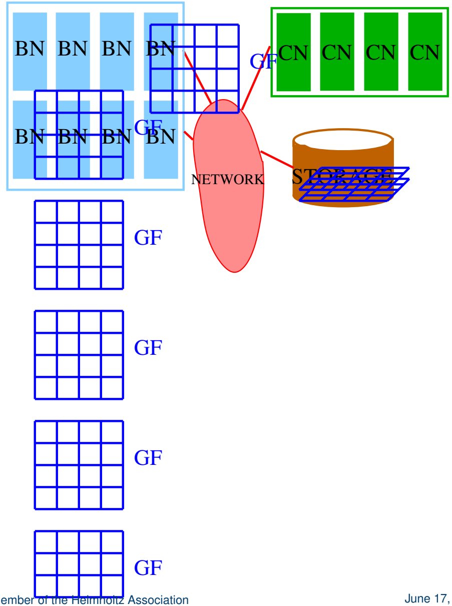
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



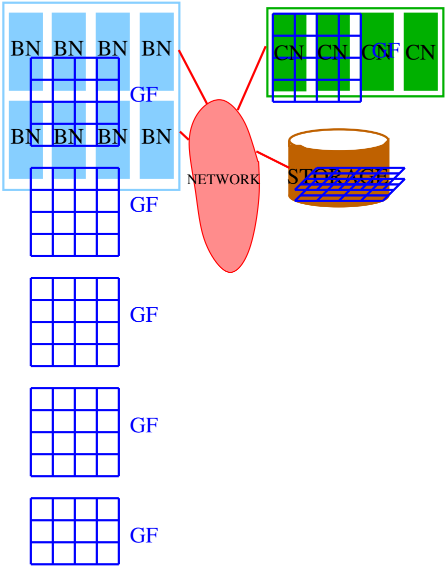
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



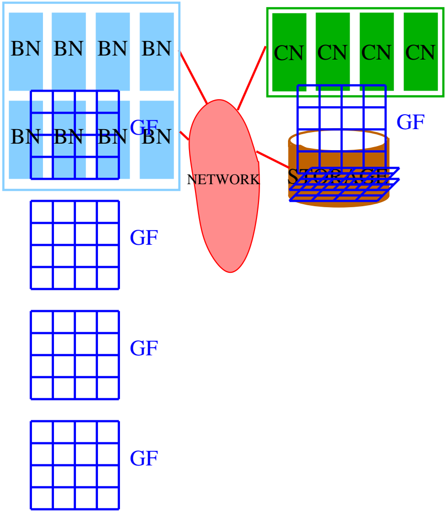
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



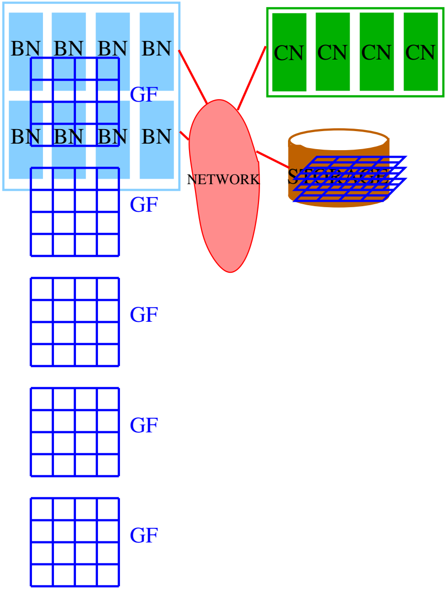
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



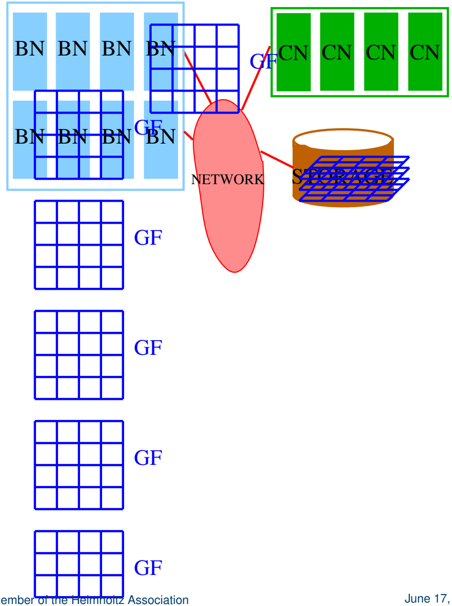
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



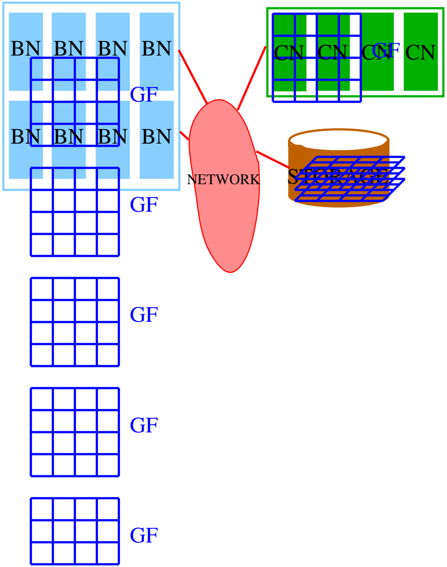
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



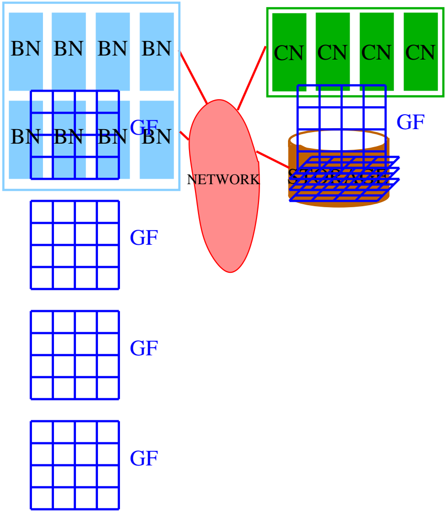
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



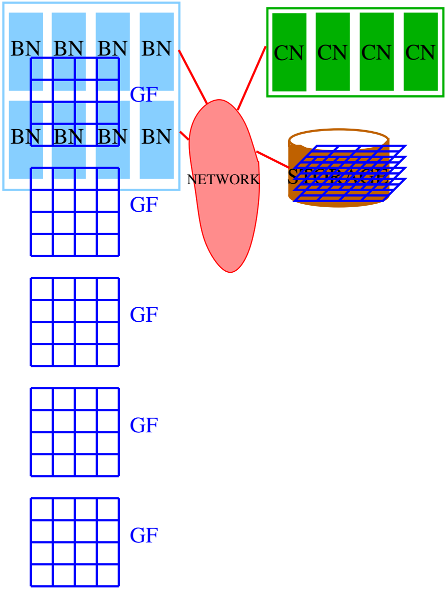
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



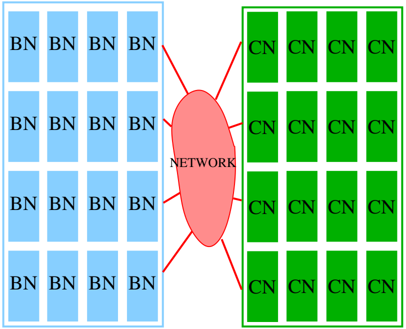
HMC gauge field generator ⟶ check-pointing

EXAMPLE 1: I/O



HMC gauge field generator ⟶ check-pointing

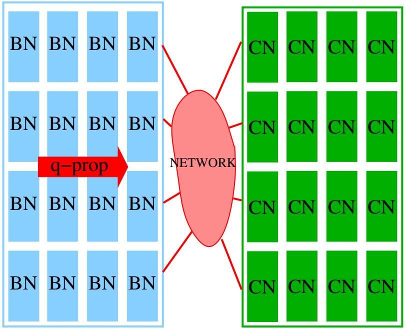
EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

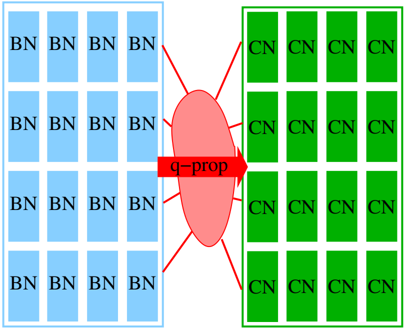
EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

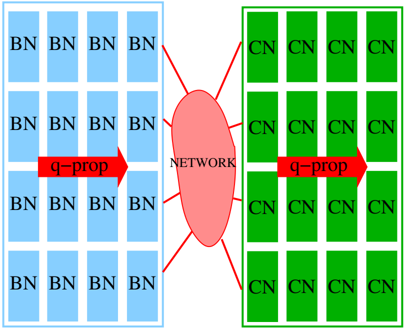
EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

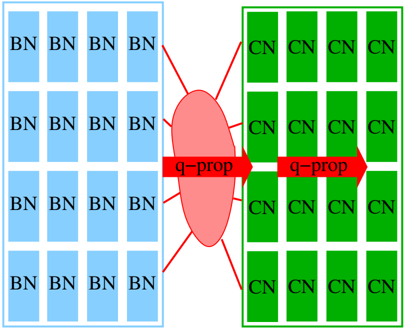
EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

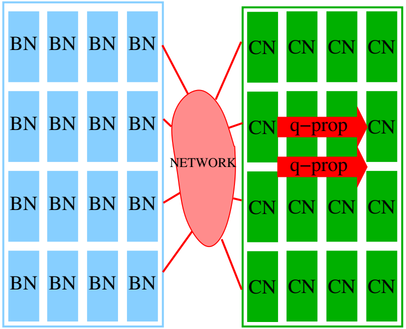
EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

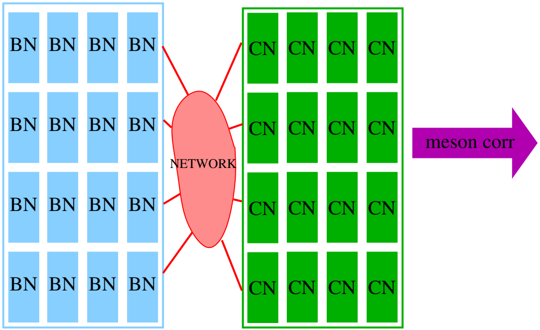
EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

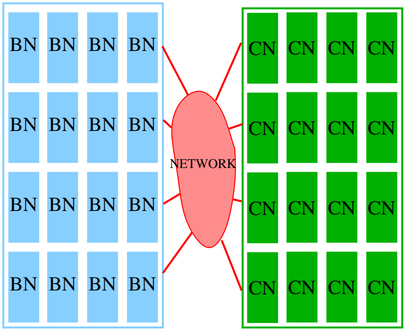
EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

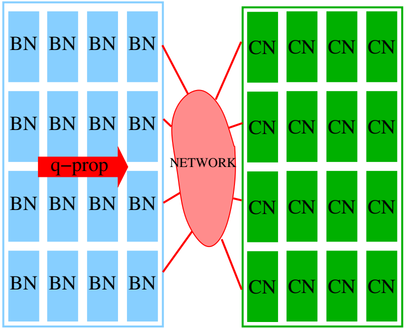
EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

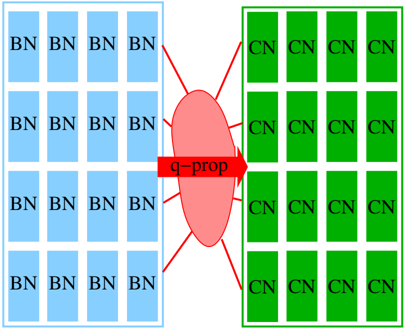
EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

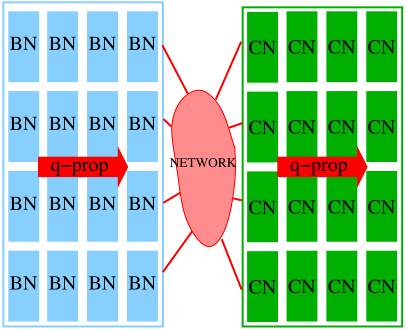
EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

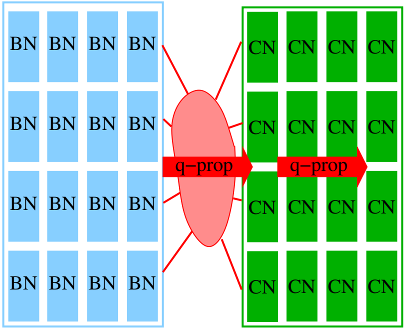
EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

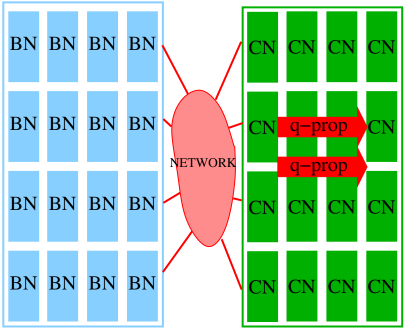
EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

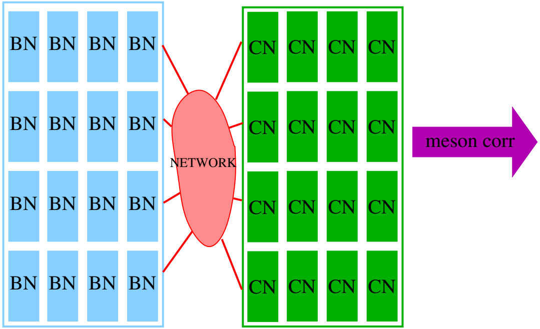
EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

EXAMPLE 2: PROPAGATORS & CONTRACTIONS



Need lots of propagators & contractions?
E.g.:

- Stochastic sources
- Multi-nucleon correlators

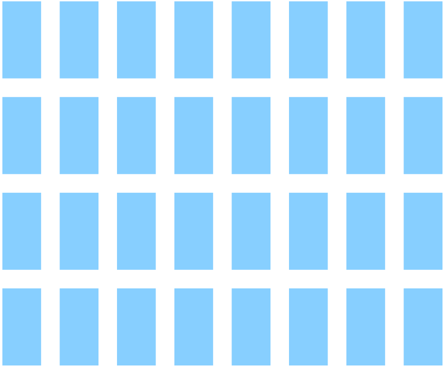
Part III: QMOD project

QMOD PROJECT

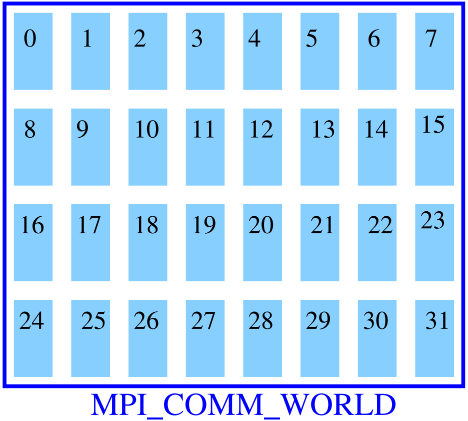
Goal:

Develop software libraries to enable to splitting a lattice simulation into concurrent tasks for separate hardware groups.

STEP 1: SPLIT THE SIMULATION



STEP 1: SPLIT THE SIMULATION



0 1 2 3 4 5 6 7

8 9 10 11 12 13 14 15

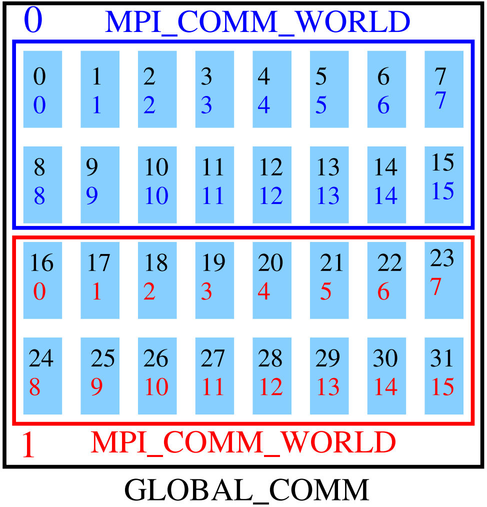
16 17 18 19 20 21 22 23

24 25 26 27 28 29 30 31

MPI_COMM_WORLD

- Split the global communicator MPI_COMM_WORLD

STEP 1: SPLIT THE SIMULATION



0 MPI_COMM_WORLD

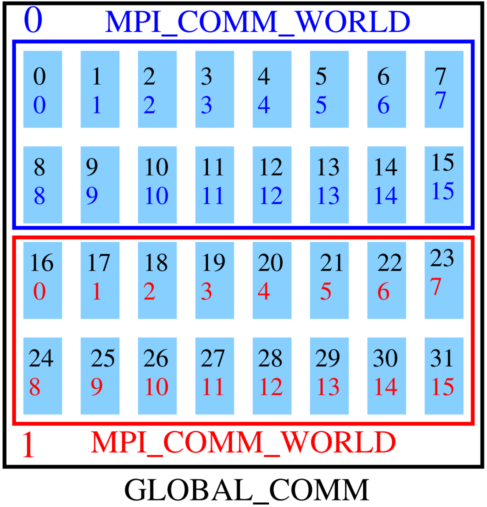
0 0	1 1	2 2	3 3	4 4	5 5	6 6	7 7
8 8	9 9	10 10	11 11	12 12	13 13	14 14	15 15
16 0	17 1	18 2	19 3	20 4	21 5	22 6	23 7
24 8	25 9	26 10	27 11	28 12	29 13	30 14	31 15

1 MPI_COMM_WORLD

GLOBAL_COMM

- Split the global communicator MPI_COMM_WORLD

STEP 1: SPLIT THE SIMULATION



0 MPI_COMM_WORLD

0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7

8 9 10 11 12 13 14 15
8 9 10 11 12 13 14 15

16 17 18 19 20 21 22 23
0 1 2 3 4 5 6 7

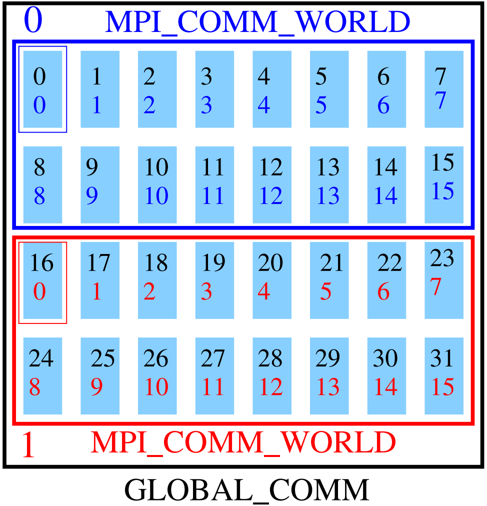
24 25 26 27 28 29 30 31
8 9 10 11 12 13 14 15

1 MPI_COMM_WORLD

GLOBAL_COMM

- Split the global communicator MPI_COMM_WORLD
- Make new global communicator: “Inter-communicator”

STEP 1: SPLIT THE SIMULATION



0 MPI_COMM_WORLD

0 0	1 1	2 2	3 3	4 4	5 5	6 6	7 7
8 8	9 9	10 10	11 11	12 12	13 13	14 14	15 15
16 0	17 1	18 2	19 3	20 4	21 5	22 6	23 7
24 8	25 9	26 10	27 11	28 12	29 13	30 14	31 15

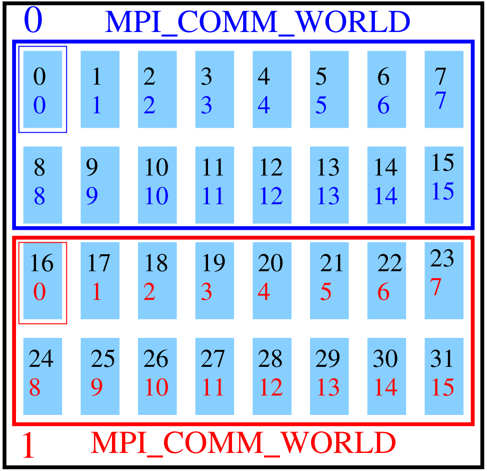
1 MPI_COMM_WORLD

GLOBAL_COMM

- Split the global communicator MPI_COMM_WORLD
- Make new global communicator: “Inter-communicator”
- Broadcast global node IDs of root nodes

[0, 16, ...]

STEP 1: SPLIT THE SIMULATION



0 MPI_COMM_WORLD

0 0	1 1	2 2	3 3	4 4	5 5	6 6	7 7
8 8	9 9	10 10	11 11	12 12	13 13	14 14	15 15
16 0	17 1	18 2	19 3	20 4	21 5	22 6	23 7
24 8	25 9	26 10	27 11	28 12	29 13	30 14	31 15

1 MPI_COMM_WORLD

GLOBAL_COMM

- Split the global communicator MPI_COMM_WORLD
- Make new global communicator: “Inter-communicator”
- Broadcast global node IDs of root nodes

[0, 16, ...]

Alternate path:

- `MPI_Comm_spawn()`
- `MPI_Connect()`

STEP 1: COMMUNICATION

Send lattice fields between the different partitions.

STEP 1: COMMUNICATION

Send lattice fields between the different partitions.
Use binary I/O as a model:

STEP 1: COMMUNICATION

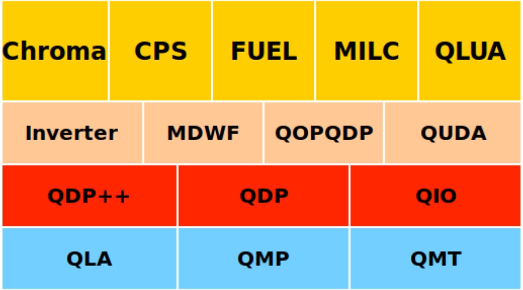
Send lattice fields between the different partitions.
Use binary I/O as a model:

- [binary write lattice field] $\longrightarrow$ [send lattice field]
- [binary read lattice field] $\longrightarrow$ [receive lattice field]

USQCD SOFTWARE

+ + +

- Open-source
- Widely used
- Architecture-specific back ends, e.g., QUDA, QPHIX
- Versatile: can be used by several high-level simulation codes



Chroma CPS FUEL MILC QLUA

Inverter MDWF QOPQDP QUDA

QDP++ QDP QIO

QLA QMP QMT

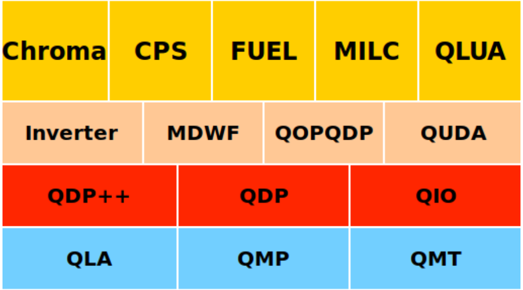
USQCD SOFTWARE

+ + +

- Open-source
- Widely used
- Architecture-specific back ends, e.g., QUDA, QPHIX
- Versatile: can be used by several high-level simulation codes

— — —

- It's complicated!



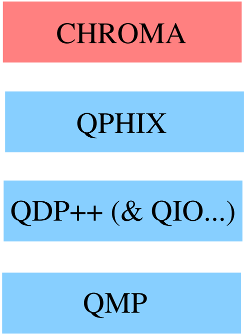
Chroma CPS FUEL MILC QLUA

Inverter MDWF QOPQDP QUDA

QDP++ QDP QIO

QLA QMP QMT

USQCD SET-MENU A



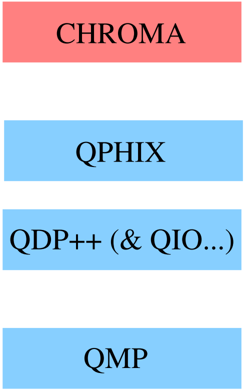
CHROMA

QPHIX

QDP++ (& QIO...)

QMP

QMOD



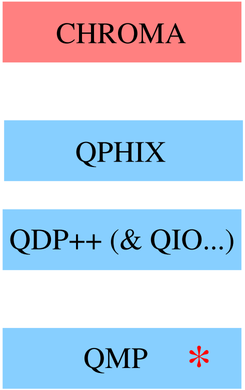
CHROMA

QPHIX

QDP++ (& QIO...)

QMP

QMOD



CHROMA

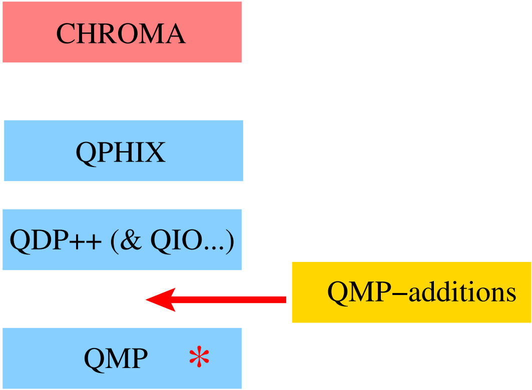
QPHIX

QDP++ (& QIO...)

QMP *

← one-line edit to QMP

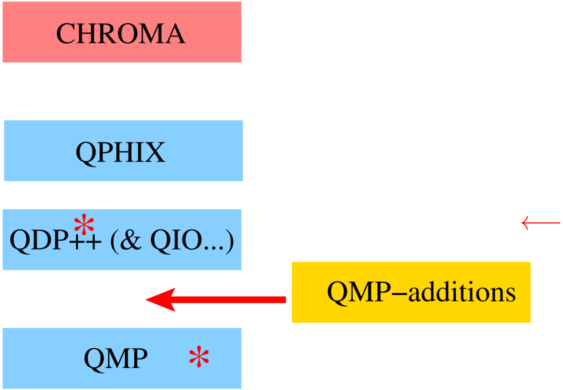
QMOD



← small library handles:

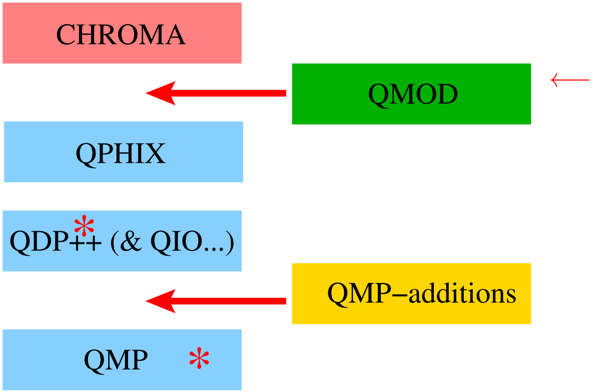
- COMM split
- passing basic info about seperate partitions

QMOD



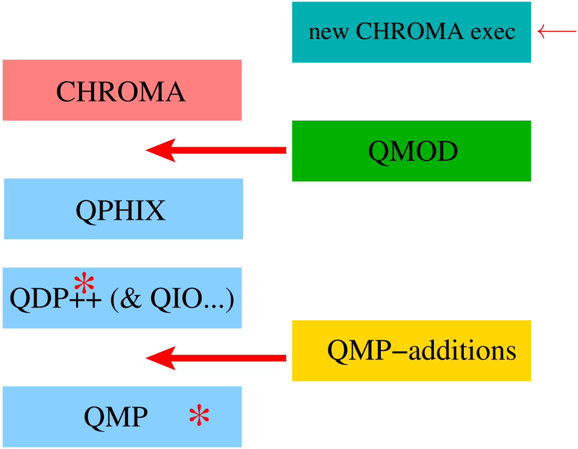
- read command line flag `-modcolor <int>` to identify new COMM
- split COMM *before* layout init

QMOD



- send lattice field to partition y
- receive lattice field from partition x

QMOD



- links to `libchroma.a`
- acts on XML input like:

```
<elem>
   <Name>QMOD_SEND_NAMED_OBJECT</Name>
   <NamedObject>
      <object_id>sh_prop_1</object_id>
      <object_type>LatticePropagator</object_t
    </NamedObject>
   <Transfer>
      <dest_partition>1</dest_partition>
      <transfer_mode>0</transfer_mode>
    </Transfer>
 </elem>
```

QMOD SHIPPING CONTAINERS

- Binary files have headers
- One can `fseek` to find a desired field element.
- Not so with MPI (though handle has some info)

QMOD SHIPPING CONTAINERS

- Binary files have headers
- One can `fseek` to find a desired field element.
- Not so with MPI (though handle has some info)

To be sure out-of-order data goes to the right place, data is packaged with a manifest.

QMOD SHIPPING CONTAINERS

- Binary files have headers
- One can `fseek` to find a desired field element.
- Not so with MPI (though handle has some info)

To be sure out-of-order data goes to the right place, data is packaged with a manifest.

```
typedef struct {
  uint32_t  send_partition;
  uint32_t  send_node;
  uint32_t  id;
  uint32_t  start_site;
  uint32_t  buf_sites;
  uint32_t  shipping_done;
  char data[];
} shippingContainer;
```

QMOD SHIPPING CONTAINERS

- Binary files have headers
- One can `fseek` to find a desired field element.
- Not so with MPI (though handle has some info)

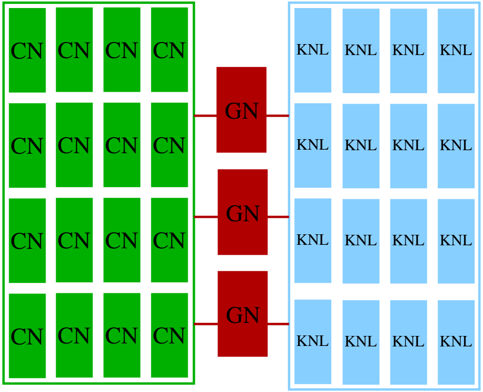
To be sure out-of-order data goes to the right place, data is packaged with a manifest.
Add info as needed.

```
typedef struct {
  uint32_t  send_partition;
  uint32_t  send_node;
  uint32_t  id;
  uint32_t  start_site;
  uint32_t  buf_sites;
  uint32_t  shipping_done;
  char data[];
} shippingContainer;
```

QMOD TESTING

Jureca cluster and KNL booster at JSC

- Cluster: InfiniBand network
- Booster: Omni-Path network
- gateway nodes joining



QMOD TESTING

To run:

- compile separate executables for each architecture
- submit as Slurm “packjob” — multiple simultaneous jobs
- single `srun` launcher line with instructions to load separate environment modules for each architecture
- still a little finicky

Slurm batch script excerpt

```
srun -n 2 \
xenv -L Intel -L ParaStationMPI/5.2.2-1-mt -L
pscom-gateway -L imkl -L libxml2/.2.9.9 -L GMP env
OMP_NUM_THREADS=24 \
$EXEC_DIR/sendtest_hsw -i unprec_clover_recv.ini-8888.0.xml
-o outhsw.xml -by 4 -bz 4 -c 24 -sy 1 -sz 1 -pxy 1 -pxyz 0
-minct 1 -geom 1 1 1 2 -modcolor 0 :  \
-n 2 xenv -L Intel -L ParaStationMPI/5.2.2-1-mt -L imkl -L
libxml2/.2.9.9 -L GMP \
env OMP_NUM_THREADS=68 env PSP_PSM=1 \
$EXEC_DIR/sendtest_knl -i unprec_clover_send.ini-8888.1.xml
-o outknl.xml -by 4 -bz 4 -c 68 -sy 1 -sz 1 -pxy 1 -pxyz 0
-minct 1 -geom 1 1 1 2 -modcolor 1
```

QMOD TESTING

To run:

- compile separate executables for each architecture
- submit as Slurm “packjob” — multiple simultaneous jobs
- single `srun` launcher line with instructions to load separate environment modules for each architecture
- still a little finicky

Slurm batch script excerpt

```
srun -n 2 \
xenv -L Intel -L ParaStationMPI/5.2.2-1-mt -L
pscom-gateway -L imkl -L libxml2/.2.9.9 -L GMP env
OMP_NUM_THREADS=24 \
$EXEC_DIR/sendtest_hsw -i unprec_clover_recv.ini-8888.0.xml
-o outhsw.xml -by 4 -bz 4 -c 24 -sy 1 -sz 1 -pxy 1 -pxyz 0
-minct 1 -geom 1 1 1 2 -modcolor 0 :  \
-n 2 xenv -L Intel -L ParaStationMPI/5.2.2-1-mt -L imkl -L
libxml2/.2.9.9 -L GMP \
env OMP_NUM_THREADS=68 env PSP_PSM=1 \
$EXEC_DIR/sendtest_knl -i unprec_clover_send.ini-8888.1.xml
-o outknl.xml -by 4 -bz 4 -c 68 -sy 1 -sz 1 -pxy 1 -pxyz 0
-minct 1 -geom 1 1 1 2 -modcolor 1
```

QMOD TESTING

To run:

- compile separate executables for each architecture
- submit as Slurm “packjob” — multiple simultaneous jobs
- single `srun` launcher line with instructions to load separate environment modules for each architecture
- still a little finicky

Slurm batch script excerpt

```
srun -n 2 \
xenv -L Intel -L ParaStationMPI/5.2.2-1-mt -L
pscom-gateway -L imkl -L libxml2/.2.9.9 -L GMP env
OMP_NUM_THREADS=24 \
$EXEC_DIR/sendtest_hsw -i unprec_clover_recv.ini-8888.0.xml
-o outhsw.xml -by 4 -bz 4 -c 24 -sy 1 -sz 1 -pxy 1 -pxyz 0
-minct 1 -geom 1 1 1 2 -modcolor 0 :  \
-n 2 xenv -L Intel -L ParaStationMPI/5.2.2-1-mt -L imkl -L
libxml2/.2.9.9 -L GMP \
env OMP_NUM_THREADS=68 env PSP_PSM=1 \
$EXEC_DIR/sendtest_knl -i unprec_clover_send.ini-8888.1.xml
-o outknl.xml -by 4 -bz 4 -c 68 -sy 1 -sz 1 -pxy 1 -pxyz 0
-minct 1 -geom 1 1 1 2 -modcolor 1
```

THE TINY QMP EDIT

```
QMP_assert(sourceNode >= 0);
```

THE TINY QMP EDIT

```
// QMP_assert(sourceNode >= 0);
QMP_assert(sourceNode != -1 );
```

THE TINY QMP EDIT

```
// QMP_assert(sourceNode >= 0);
QMP_assert(sourceNode != -1 );
```

Means:

```
<src_partition>-2<src_partition>
```

THE TINY QMP EDIT

```
// QMP_assert(sourceNode >= 0);
QMP_assert(sourceNode != -1 );
```

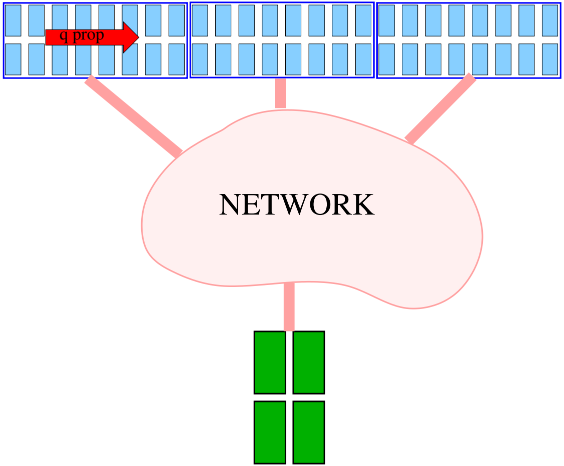
Means:

```
<src_partition>-2<src_partition>
```

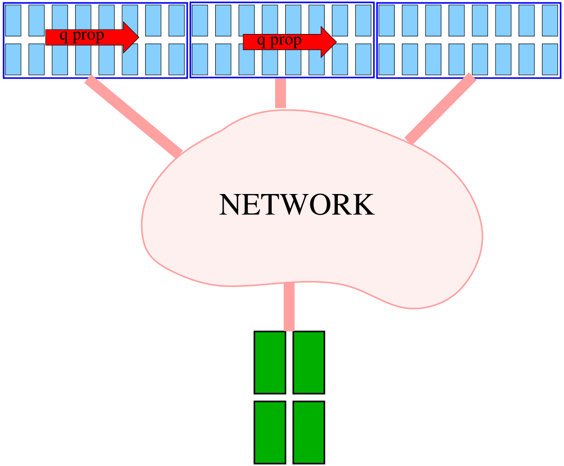
is interpreted as :

```
<src_partition>MPI_ANY_SOURCE</src_partition>
```

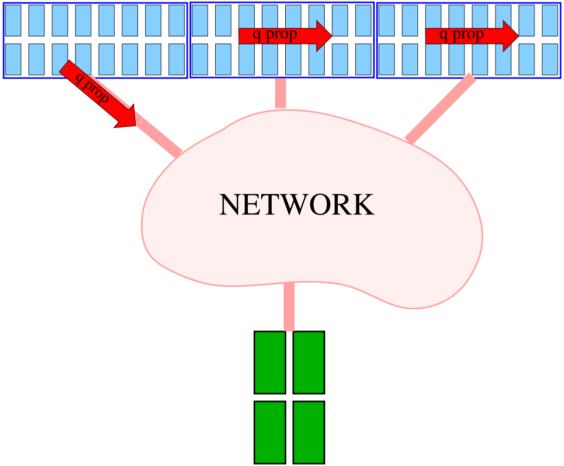
QMOD



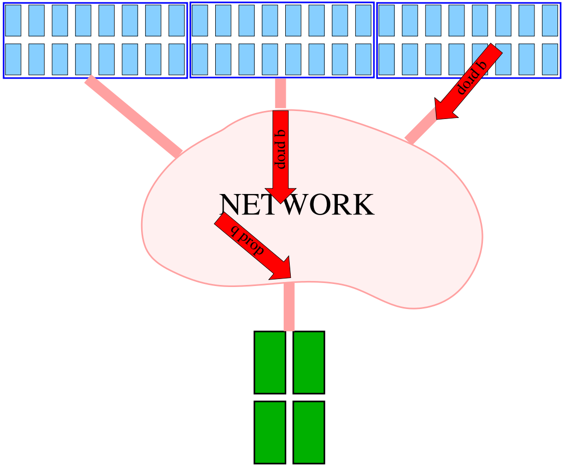
QMOD



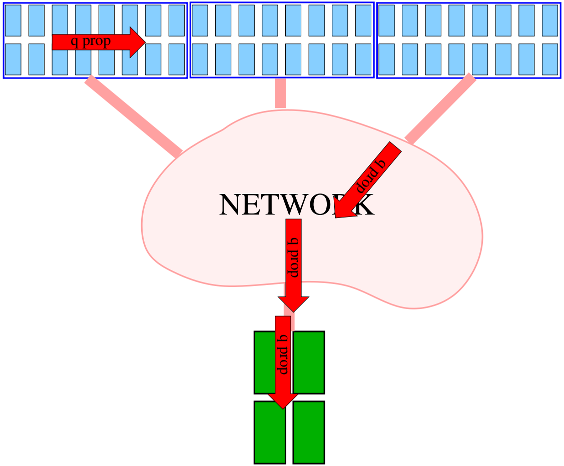
QMOD



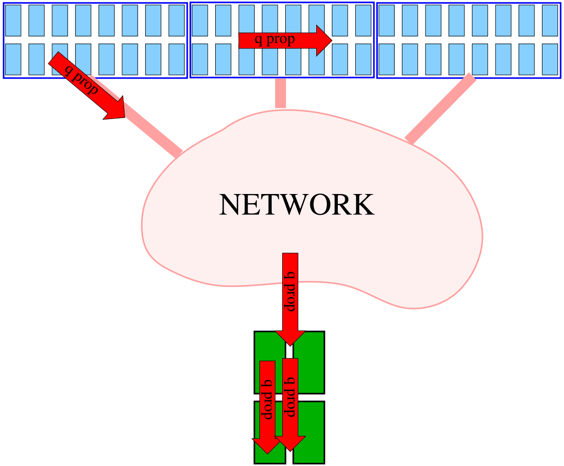
QMOD



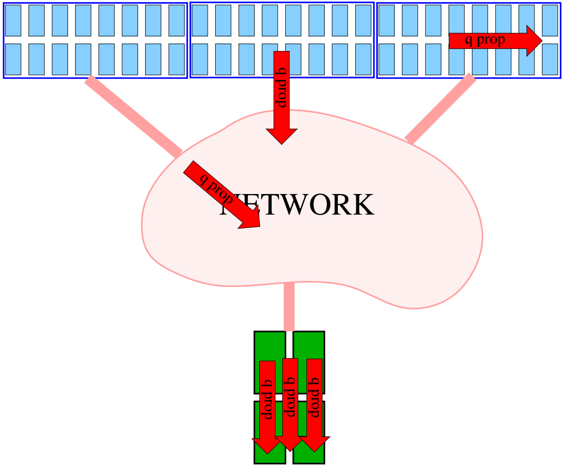
QMOD



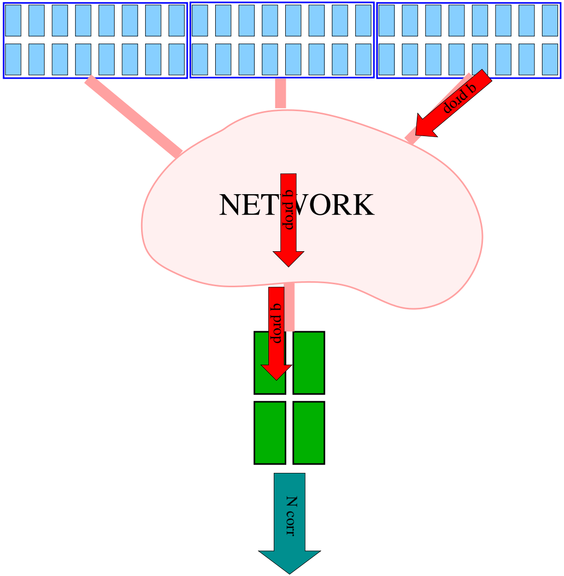
QMOD



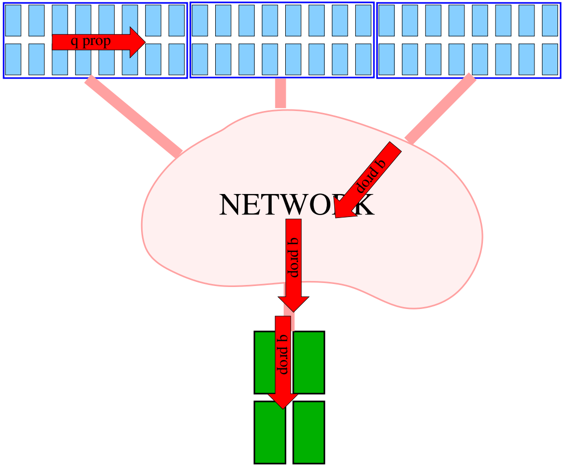
QMOD



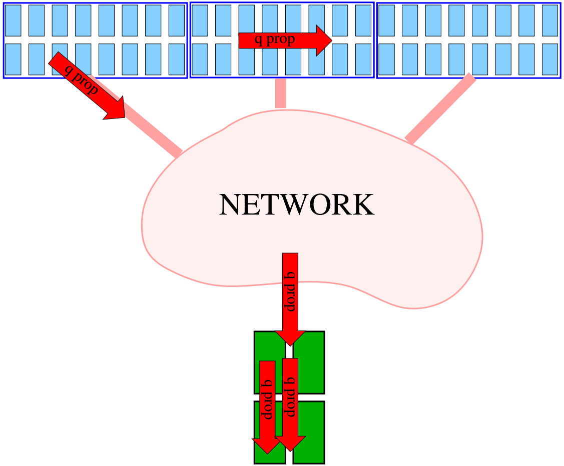
QMOD



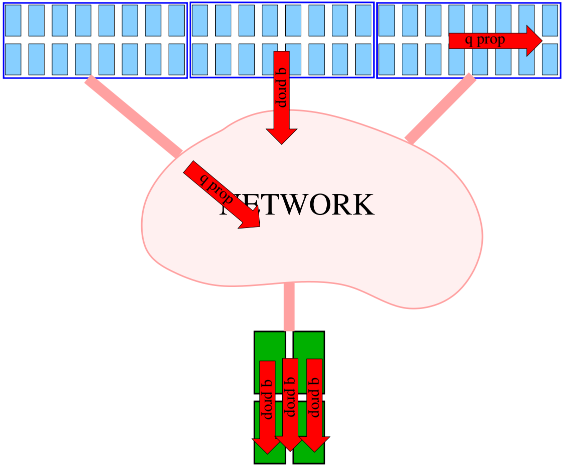
QMOD



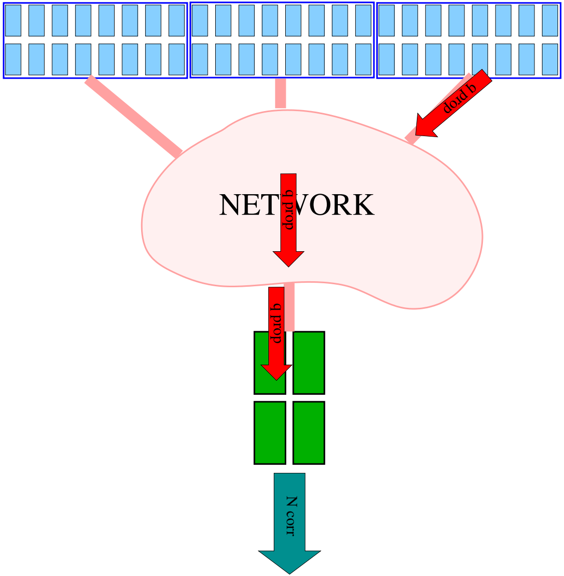
QMOD



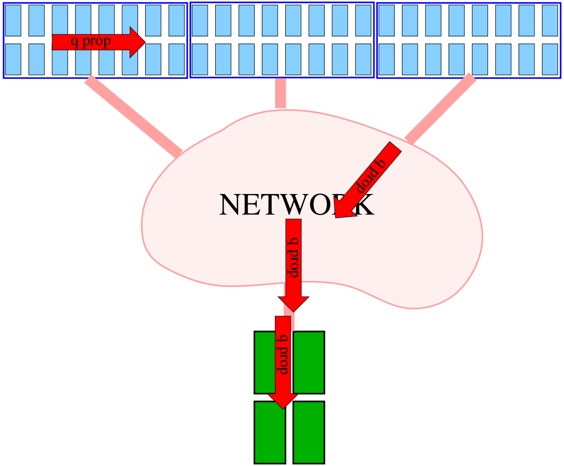
QMOD



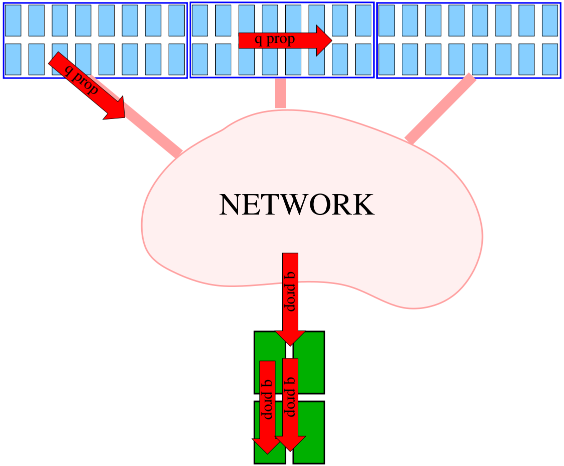
QMOD



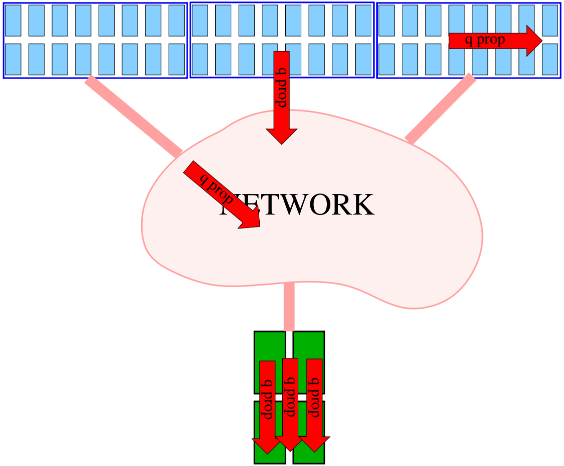
QMOD



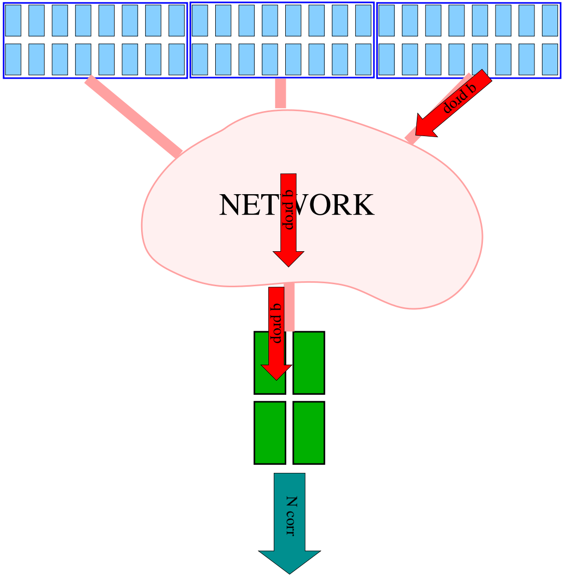
QMOD



QMOD



QMOD



TO-DO LIST

- More testing
- Pass other lattice objects
- Clean up and share
- more sophisticated transfer modes
  - more interface nodes (now just 1 per partition)
  - all-to-all communication?

CONCLUSIONS

- Should speculate about the future of supercomputers in order to influence it
- Consider modularity in LQCD code design
- Many thanks to USQCD developers, whose work I borrowed from heavily
- Special thanks to B. Joo and J. Osborn for fielding many questions