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?
Maybe you have a choice of architectures.
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?
Working modular system at Jülich.
Development of a prototype modular supercomputer.
Development of a prototype modular supercomputer.
Development of a prototype modular supercomputer.
Development of a prototype modular supercomputer.
Development of a prototype modular supercomputer.
Development of a prototype modular supercomputer. See:

- DEEP-EST project
  https://www.deep-projects.eu/
- Planned expansion of JSC’s JUWELS system.
Development of a prototype modular supercomputer.
See:
- DEEP-EST project
  https://www.deep-projects.eu/
- Planned expansion of JSC’s JUWELS system.
Development of a prototype modular supercomputer.
See:
- DEEP-EST project
  https://www.deep-projects.eu/
- Planned expansion of JSC’s JUWELS system.
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 $\rightarrow$ checkpointing
EXAMPLE 1: I/O

HMC gauge field generator → checkpointing
EXAMPLE 1: I/O

HMC gauge field generator $\rightarrow$ check-pointing
EXAMPLE 1: I/O

HMC gauge field generator \(\rightarrow\) checkpointing
EXAMPLE 1: I/O

HMC gauge field generator $\rightarrow$ check-pointing
EXAMPLE 1: I/O

HMC gauge field generator $\rightarrow$ checkpointing
EXAMPLE 1: I/O

HMC gauge field generator → check-pointing
EXAMPLE 1: I/O

HMC gauge field generator $\rightarrow$ check-pointing
EXAMPLE 1: I/O

HMC gauge field generator $\rightarrow$ check-pointing
EXAMPLE 1: I/O

HMC gauge field generator $\rightarrow$ 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 $\rightarrow$ checkpointing
EXAMPLE 1: I/O

HMC gauge field generator $\rightarrow$ 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

Member of the Helmholtz Association

June 17, 2019
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 $\rightarrow$ check-pointing
EXAMPLE 1: I/O

HMC gauge field generator \(\rightarrow\) checkpointing
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
Goal:

Develop software libraries to enable to splitting a lattice simulation into concurrent tasks for separate hardware groups.
STEP 1: SPLIT THE SIMULATION

Split the global communicator MPI COMM WORLD
Make new global communicator: "Inter-communicator"
Broadcast global node IDs of root nodes

Alternate path:
MPI Comm spawn()
MPI Connect()
STEP 1: SPLIT THE SIMULATION

- Split the global communicator
  MPI_COMM_WORLD

```c
MPI_Comm_split(MPI_COMM_WORLD, &split, 2, &new_comm);
```

 alternate path:

```c
MPI_Comm_spawn(new_comm, &root_nodes, 2, 0, &new_comm);
```
**STEP 1: SPLIT THE SIMULATION**

- Split the global communicator MPI_COMM_WORLD

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
</tr>
</tbody>
</table>

**GLOBAL_COMM**

Member of the Helmholtz Association
June 17, 2019
Slide 14
STEP 1: SPLIT THE SIMULATION

- Split the global communicator MPI_COMM_WORLD
- Make new global communicator: “Inter-communicator”
### STEP 1: SPLIT THE SIMULATION

- **Split the global communicator**
  - MPI_COMM_WORLD

- **Make new global communicator**:
  - "Inter-communicator"

- **Broadcast global node IDs of root nodes**
  - \([0, 16, \ldots]\)

<table>
<thead>
<tr>
<th>0</th>
<th>MPI_COMM_WORLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16</th>
<th>MPI_COMM_WORLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>15</td>
</tr>
</tbody>
</table>

GLOBAL_COMM

Member of the Helmholtz Association

June 17, 2019
**STEP 1: SPLIT THE SIMULATION**

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

\[ [0, 16, \ldots] \]

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] → [send lattice field]
- [binary read lattice field] → [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

<table>
<thead>
<tr>
<th>Chroma</th>
<th>CPS</th>
<th>FUEL</th>
<th>MILC</th>
<th>QLUUA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverter</td>
<td>MDWF</td>
<td>QOPQDP</td>
<td>QUDA</td>
<td></td>
</tr>
<tr>
<td>QDP++</td>
<td>QDP</td>
<td>QIO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QLA</td>
<td>QMP</td>
<td>QMT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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!
USQCD SET-MENU A

- CHROMA
- QPHIX
- QDP++ (& QIO...)
- QMP
QMOD

CHROMA

QPHIX

QDP++ (& QIO...)

QMP *← one-line edit to QMP
QMOD

CHROMA

QPHIX

QDP++ (& QIO...)

QMP–additions

QMP

← small library handles:
- COMM split
- passing basic info about separate partitions
read command line flag `-modcolor <int>` to identify new COMM
- split COMM before layout init
- send lattice field to partition y
- receive lattice field from partition x
QMOD

CHROMA

QMOD

QMP−additions

new CHROMA exec

* links to libchroma.a

* acts on XML input like:

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

QPHIX

QDP++ (& QIO...)

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

define 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;
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;
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 -Lpscom-gateway -L imkl -L libxml2/.2.9.9 -L GMP envOMP_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

```bash
srun -n 2 \n  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 \n  $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  -n 2 xenv -L Intel -L ParaStationMPI/5.2.2-1-mt -L imkl -L libxml2/.2.9.9 -L GMP \n  env OMP_NUM_THREADS=68 env PSP_PSM=1 \n  $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 \n  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 \n  $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  -n 2 xenv -L Intel -L ParaStationMPI/5.2.2-1-mt -L imkl -L libxml2/.2.9.9 -L GMP \n  env OMP_NUM_THREADS=68 env PSP_PSM=1 \n  $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```

Member of the Helmholtz Association  
June 17, 2019  
JÜLICH Forschungszentrum
qmp_assert(sourceNode >= 0);
// 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

NETWORK

q prop q prop

Member of the Helmholtz Association

June 17, 2019

Slide 29
QMOD

NETWORK

Member of the Helmholtz Association

June 17, 2019
QMOD

NETWORK

doub l b

doub l b

doub l b

doub l b

q prop

q prop

q prop

q prop
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