Generalized Hypercube (GHC)

A topology plugin

M. Clayer
A. Faure

September 25, 2018

- HyperCube
- Generalized HyperCube
- Slurm configuration
- Examples

Hypercube topology

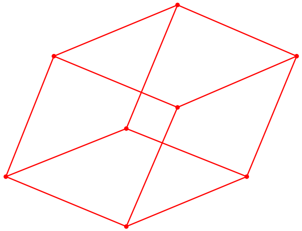
Unit Hypercube:

A **n***-dimensional* ***unit hypercube*** *is defined by* 2^n *point, which coordinates are composed by* 0 *or* 1*.*
These points represent the corners of the unit hypercube.
For $n = 2$*: a square,* $n = 3$*: a cube.*

Hypercube topology

Unit Hypercube:

A **n***-dimensional* ***unit hypercube*** *is defined by* 2^n *point, which coordinates are composed by* 0 *or* 1*.*
These points represent the corners of the unit hypercube.
For $n = 2$*: a square,* $n = 3$*: a cube.*

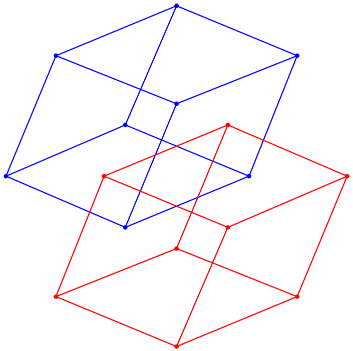


a cube, $n = 3$

Hypercube topology

Unit Hypercube:

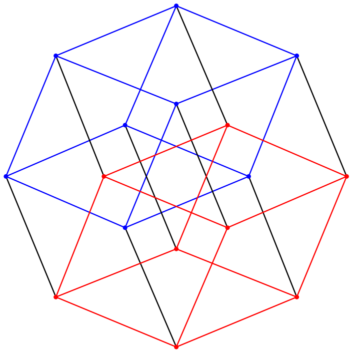
A **n***-dimensional* ***unit hypercube*** *is defined by* 2^n *point, which coordinates are composed by* 0 *or* 1*.*
These points represent the corners of the unit hypercube.
For $n = 2$*: a square,* $n = 3$*: a cube.*


two cubes, $n = 3$

Hypercube topology

Unit Hypercube:

A **n***-dimensional* ***unit hypercube*** *is defined by* 2^n *point, which coordinates are composed by* 0 *or* 1*.*
These points represent the corners of the unit hypercube.
For $n = 2$*: a square,* $n = 3$*: a cube.*

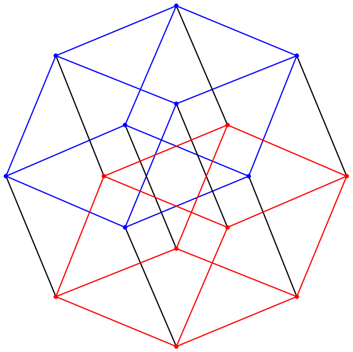


a tesseract, $n = 4$

Hypercube topology

Hypercube Topology:

- *Each corner represent a switch*

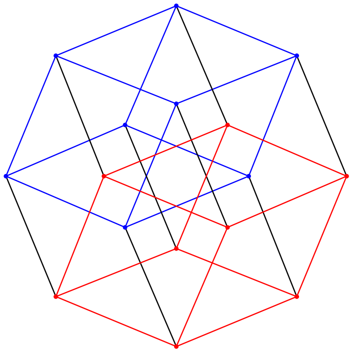


a tesseract, $n = 4$

Hypercube topology

Hypercube Topology:

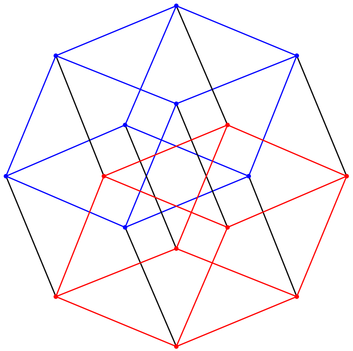
- *Each corner represent a switch*
- *we connect $\mathcal{T}$ terminals on each switches*


a tesseract, $n = 4$

Hypercube topology

Hypercube Topology:

- *Each corner represent a switch*
- *we connect $\mathcal{T}$ terminals on each switches*
  - *Maximum of $\mathcal{T} \times 2^{n}$ terminals*

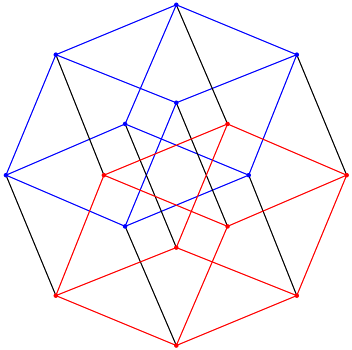


a tesseract, $n = 4$

Hypercube topology

Hypercube Topology:

- *Each corner represent a switch*
- *we connect $\mathcal{T}$ terminals on each switches*
  - *Maximum of $\mathcal{T} \times 2^n$ terminals*
- *number of hops between 2 terminals: $2 + n$*



a tesseract, $n = 4$

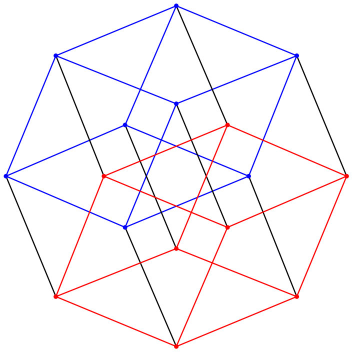
Hypercube topology

Hypercube Topology:

- *Each corner represent a switch*
- *we connect $\mathcal{T}$ terminals on each switches*
    - *Maximum of $\mathcal{T} \times 2^{\boldsymbol{n}}$ terminals*
- *number of hops between 2 terminals: $2 + n$*

Limitation

Hypercube have a strong constraint: the number of switches: $2^{\boldsymbol{n}}$


a tesseract, $n = 4$

Hypercube topology

Hypercube Topology:

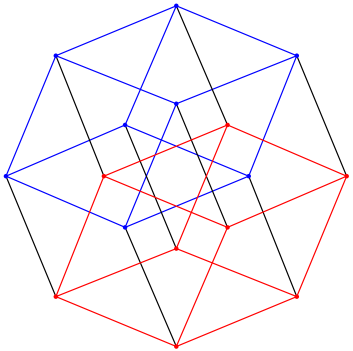
- *Each corner represent a switch*
- *we connect $\mathcal{T}$ terminals on each switches*
  - *Maximum of $\mathcal{T} \times 2^{\boldsymbol{n}}$ terminals*
- *number of hops between 2 terminals: $2 + n$*

Limitation

Hypercube have a strong constraint: the number of switches: $2^{\boldsymbol{n}}$

Solution

A similar topology avoid this constraint:
*the **G**eneralized **H**yper**C**ube topology (GHC)*



a tesseract, $n = 4$

Generalized HyperCube

GHC

Defining a n-dimensional GHC topology by:

- *a number of switches for each dimension:* $\mathcal{S}_i$

  $\Rightarrow$ *number total of switches:* $\prod_{i=1}^{n} S_i$

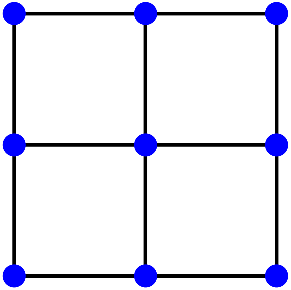
Generalized HyperCube

GHC

Defining a n-dimensional GHC topology by:

- *a number of switches for each dimension:* $\mathcal{S}_{\boldsymbol{i}}$

  $\Rightarrow$ *number total of switches:* $\prod_{i=1}^{n} S_i$



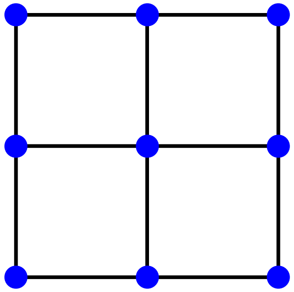
Generalized HyperCube

GHC

Defining a n-dimensional GHC topology by:

- *a number of switches for each dimension:* $\mathcal{S}_i$

  $\Rightarrow$ *number total of switches:* $\prod_{i=1}^{n} S_i$



$n = 2,\ \mathcal{S} = (3, 3)$

Generalized HyperCube

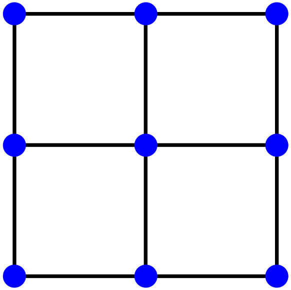
GHC

Defining a n-dimensional GHC topology by:

- *a number of switches for each dimension:* $\mathcal{S}_i$

  $\Rightarrow$ *number total of switches:* $\prod_{i=1}^{n} S_i$

- *a number of terminal* $\mathcal{T}$ *per switch*



$n = 2,\ \mathcal{S} = (3, 3)$

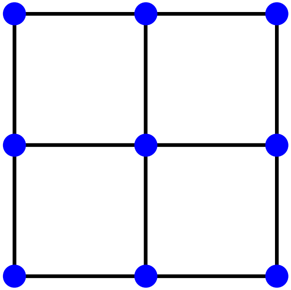
Generalized HyperCube

GHC

Defining a n-dimensional GHC topology by:

- *a number of switches for each dimension: $\mathcal{S}_i$*

  $\Rightarrow$ *number total of switches:* $\prod_{i=1}^{n} S_i$
- *a number of terminal $\mathcal{T}$ per switch*
- *in each dimension, switches are fully connected*



$n = 2,\ \mathcal{S} = (3, 3)$

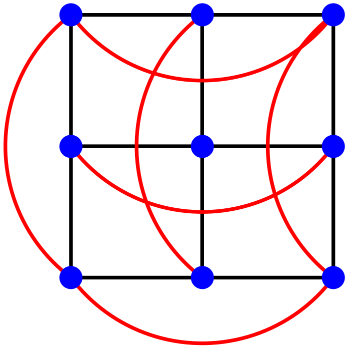
Generalized HyperCube

GHC

Defining a n-dimensional GHC topology by:

- *a number of switches for each dimension:* $\mathcal{S}_i$

  $\Rightarrow$ *number total of switches:* $\prod_{i=1}^{n} S_i$
- *a number of terminal* $\mathcal{T}$ *per switch*
- *in each dimension, switches are fully connected*



$$n = 2,\ \mathcal{S} = (3, 3)$$

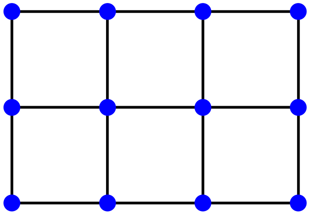
Generalized HyperCube

GHC

Defining a n-dimensional GHC topology by:

- *a number of switches for each dimension: $\mathcal{S}_i$*

  $\Rightarrow$ *number total of switches:* $\prod_{i=1}^{n} S_i$
- *a number of terminal $\mathcal{T}$ per switch*
- *in each dimension, switches are fully connected*



$n = 2$, $\mathcal{S} = (4, 3)$

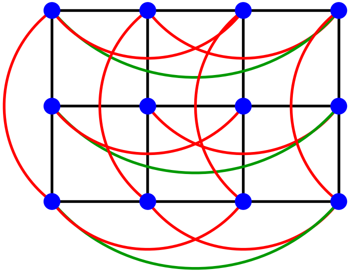
Generalized HyperCube

GHC

Defining a n-dimensional GHC topology by:

- *a number of switches for each dimension: $\mathcal{S}_i$*

  $\Rightarrow$ number total of switches: $\prod_{i=1}^{n} S_i$
- *a number of terminal $\mathcal{T}$ per switch*
- *in each dimension, switches are fully connected*



$n = 2$, $\mathcal{S} = (4, 3)$

Generalized HyperCube

GHC

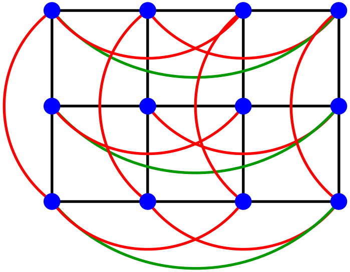
Defining a n-dimensional GHC topology by:

- *a number of switches for each dimension:* $\mathcal{S}_{\boldsymbol{i}}$

  $\Rightarrow$ *number total of switches:* $\prod_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} S_{\boldsymbol{i}}$
- *a number of terminal* $\mathcal{T}$ *per switch*
- *in each dimension, switches are fully connected*

Note

GHC topology can be represented in a n-dimensional euclidian space



$n = 2,\ \mathcal{S} = (4, 3)$

Generalized HyperCube

GHC

Defining a n-dimensional GHC topology by:

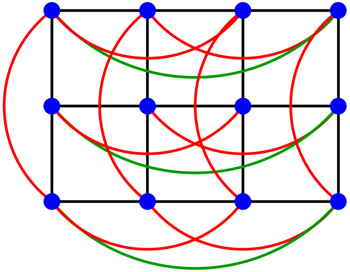
- *a number of switches for each dimension: $\mathcal{S}_i$*

  $\Rightarrow$ number total of switches: $\prod_{i=1}^{n} S_i$

- *a number of terminal $\mathcal{T}$ per switch*
- *in each dimension, switches are fully connected*

Note

GHC topology can be represented in a n-dimensional euclidian space
$\Rightarrow$ switches have coordinates!



$n = 2$, $\mathcal{S} = (4, 3)$

Generalized HyperCube

GHC

Defining a n-dimensional GHC topology by:

- *a number of switches for each dimension: $\mathcal{S}_{\boldsymbol{i}}$*

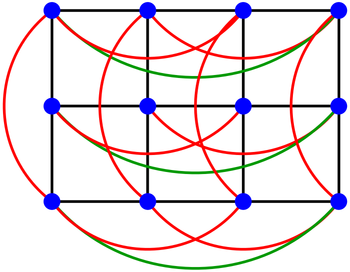
  $\Rightarrow$ number total of switches: $\prod_{i=1}^{n} S_i$
- *a number of terminal $\mathcal{T}$ per switch*
- *in each dimension, switches are fully connected*

Note

GHC topology can be represented in a n-dimensional euclidian space
$\Rightarrow$ switches have coordinates!

Proposition

2 switches are linked $\Leftrightarrow$ Their coordinates differ by only one coordinate



$n = 2,\ \mathcal{S} = (4, 3)$

Framework

topology.conf

```
SwitchName=sw1 Nodes=n0 Switches=sw2,sw4
SwitchName=sw2 Nodes=n1 Switches=sw1,sw3
SwitchName=sw3 Nodes=n2 Switches=sw2,sw4
SwitchName=sw4 Nodes=n3 Switches=sw1,sw3
```

Framework

topology.conf

```
SwitchName=sw1 Nodes=n0 Switches=sw2,sw4
SwitchName=sw2 Nodes=n1 Switches=sw1,sw3
SwitchName=sw3 Nodes=n2 Switches=sw2,sw4
SwitchName=sw4 Nodes=n3 Switches=sw1,sw3
```

Initialisation

topology.conf permit to compute:

- topology parameters (dimension n and $\mathcal{S}$)
- set up coordinates on switches

Framework

topology.conf

```
SwitchName=sw1 Nodes=n0 Switches=sw2,sw4
SwitchName=sw2 Nodes=n1 Switches=sw1,sw3
SwitchName=sw3 Nodes=n2 Switches=sw2,sw4
SwitchName=sw4 Nodes=n3 Switches=sw1,sw3
```

slurm.conf

```
TopologyPlugin=topology/ghc
SelectType=select/linear
```

Initialisation

topology.conf permit to compute:

- topology parameters (dimension n and $\mathcal{S}$)
- set up coordinates on switches

Framework

topology.conf

```
SwitchName=sw1 Nodes=n0 Switches=sw2,sw4
SwitchName=sw2 Nodes=n1 Switches=sw1,sw3
SwitchName=sw3 Nodes=n2 Switches=sw2,sw4
SwitchName=sw4 Nodes=n3 Switches=sw1,sw3
```

slurm.conf

```
TopologyPlugin=topology/ghc
SelectType=select/linear
```

Initialisation

topology.conf permit to compute:

- topology parameters (dimension n and $\mathcal{S}$)
- set up coordinates on switches

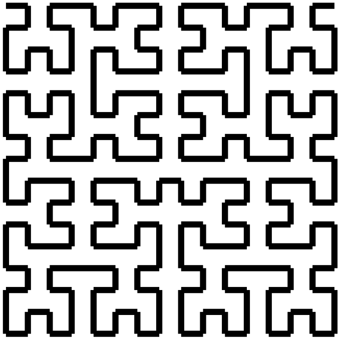
select linear

- use of Slurm best fit algorithm
  $\Rightarrow$ linear path across the GHC topology

linear Path & switches selection

How to get a linear path

- Hilbert's curve
  - map the switches to *n*-dimensional space into a 1-dimensional space
  - achieve a high degree of locality for jobs



$2\mathcal{D}$ Hilbert's curve

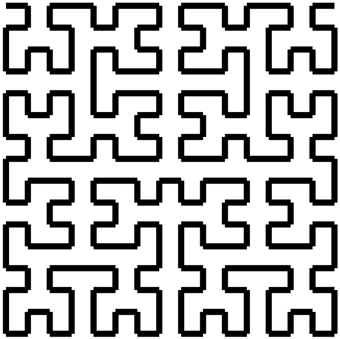
linear Path & switches selection

How to get a linear path

- Hilbert's curve
  - map the switches to *n*-dimensional space into a 1-dimensional space
  - achieve a high degree of locality for jobs

switches selection

- loop through the Hilbert curve
  - create a cluster:
    of neighboring nodes
  - compute the variance for this cluster:
    based on the distance set, between each cluster's nodes.
- choose the cluster with the lowest variance



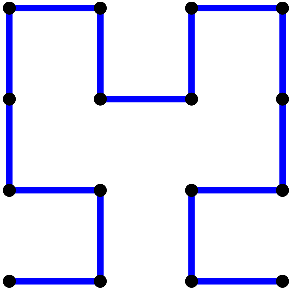
$2\mathcal{D}$ Hilbert's curve

Example - scontrol show topology

Example - scontrol show topology

$n = 2$, $\mathcal{S} = (4, 4)$, with 1 node per switch

```
slurm$ scontrol show topology

GHC NbSwitches: 16 Dimensions: 2

Dimension 1: 4 Dimension 2: 4

SwitchName=sw1 NodeCount=2 Nodes=node[0-1]

      Switches=sw2,sw3,sw4,sw5,sw9,sw13
```



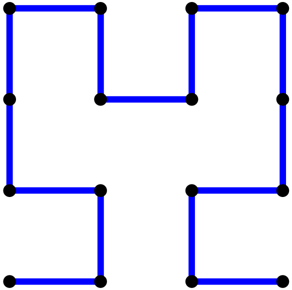
$2\mathcal{D}$, $\mathcal{S} = (4, 4)$

Example - jobs allocation

Example - launch of 4 tasks

$n = 2$, $\mathcal{S} = (4, 4)$, with 1 node per switch
launch multiple srun -n4:

```
slurm$ srun -n4 sleep 50 &
slurm$ squeue
JOBID  NAME  USER ST TIME  NODES NODELIST
153    sleep slurm R  0:02  4    node[1-4]
```



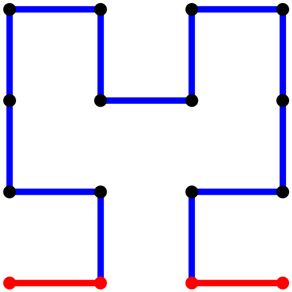
$2\mathcal{D}$, $\mathcal{S} = (4, 4)$

Example - jobs allocation

Example - launch of 4 tasks

$n = 2$, $\mathcal{S} = (4,4)$, with 1 node per switch
launch multiple srun -n4:

```
slurm$ srun -n4 sleep 50 &
slurm$ squeue
JOBID  NAME  USER ST TIME  NODES NODELIST
153    sleep slurm R  0:02  4    node[1-4]
```



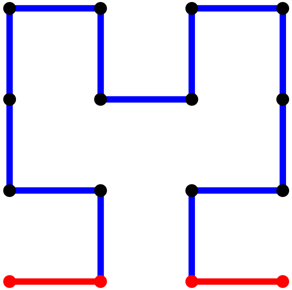
$2\mathcal{D}$, $\mathcal{S} = (4,4)$

Example - jobs allocation

Example - launch of 4 tasks

$n = 2$, $\mathcal{S} = (4, 4)$, with 1 node per switch
launch multiple srun -n4:

```
slurm$ srun -n4 sleep 50 &
slurm$ squeue
JOBID  NAME  USER ST TIME  NODES NODELIST
154    sleep slurm R  0:02  4     node[5-8]
153    sleep slurm R  0:07  4     node[1-4]
```



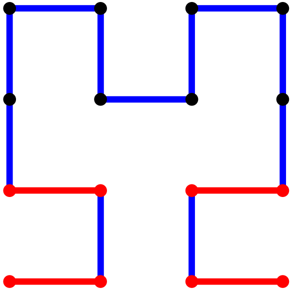
$2\mathcal{D}$, $\mathcal{S} = (4, 4)$

Example - jobs allocation

Example - launch of 4 tasks

$n = 2$, $\mathcal{S} = (4, 4)$, with 1 node per switch
launch multiple srun -n4:

```
slurm$ srun -n4 sleep 50 &
slurm$ squeue
JOBID  NAME  USER ST TIME  NODES NODELIST
154    sleep slurm R  0:02  4    node[5-8]
153    sleep slurm R  0:07  4    node[1-4]
```



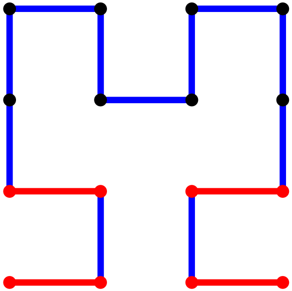
$2\mathcal{D}$, $\mathcal{S} = (4, 4)$

Example - jobs allocation

Example - launch of 4 tasks

$n = 2$, $\mathcal{S} = (4, 4)$, with 1 node per switch
launch multiple srun -n4:

```
slurm$ srun -n4 sleep 50 &
slurm$ squeue
JOBID  NAME  USER ST TIME  NODES NODELIST
155    sleep slurm R  0:02  4    node[9-10,13-14]
154    sleep slurm R  0:07  4    node[5-8]
153    sleep slurm R  0:12  4    node[1-4]
```



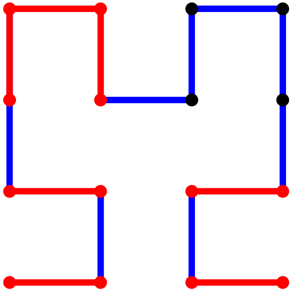
$2\mathcal{D}$, $\mathcal{S} = (4, 4)$

Example - jobs allocation

Example - launch of 4 tasks

$n = 2$, $\mathcal{S} = (4,4)$, with 1 node per switch
launch multiple srun -n4:

```
slurm$ srun -n4 sleep 50 &

slurm$ squeue

JOBID  NAME  USER ST TIME  NODES NODELIST

155    sleep slurm R  0:02  4    node[9-10,13-14]

154    sleep slurm R  0:07  4    node[5-8]

153    sleep slurm R  0:12  4    node[1-4]
```



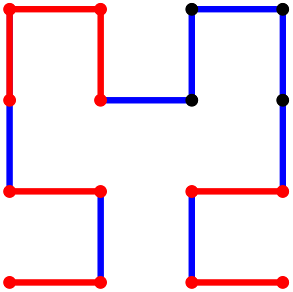
$2\mathcal{D}$, $\mathcal{S} = (4,4)$

Example - jobs allocation

Example - launch of 4 tasks

$n = 2$, $\mathcal{S} = (4, 4)$, with 1 node per switch
launch multiple srun -n4:

```
slurm$ srun -n4 sleep 50 &
slurm$ squeue
JOBID  NAME  USER ST TIME  NODES NODELIST
156    sleep slurm R  0:02  4    node[11-12,15-16]
155    sleep slurm R  0:07  4    node[9-10,13-14]
154    sleep slurm R  0:12  4    node[5-8]
153    sleep slurm R  0:17  4    node[1-4]
```



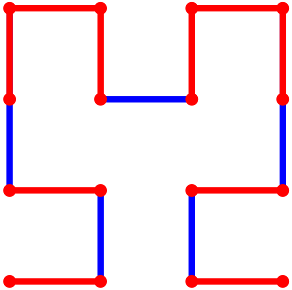
$2\mathcal{D}$, $\mathcal{S} = (4, 4)$

Example - jobs allocation

Example - launch of 4 tasks

$n = 2$, $\mathcal{S} = (4,4)$, with 1 node per switch
launch multiple srun -n4:

```
slurm$ srun -n4 sleep 50 &
slurm$ squeue
JOBID  NAME  USER ST TIME  NODES NODELIST
156    sleep slurm R  0:02  4    node[11-12,15-16]
155    sleep slurm R  0:07  4    node[9-10,13-14]
154    sleep slurm R  0:12  4    node[5-8]
153    sleep slurm R  0:17  4    node[1-4]
```



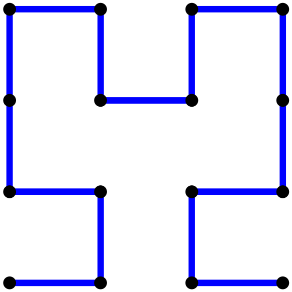
$2\mathcal{D}$, $\mathcal{S} = (4,4)$

Example - jobs allocation

Example - launch of 3 tasks

$n = 2$, $\mathcal{S} = (4,4)$, with 1 node per switch
launch multiple srun -n3:

```
slurm$ srun -n3 sleep 50 &
slurm$ squeue
JOBID NAME  USER ST TIME  NODES NODELIST
 197  sleep slurm R 0:01  3 node[1,3-4]
 196  sleep slurm R 0:05  3 node[7,11,15]
 195  sleep slurm R 0:09  3 node[8,12,16]
 194  sleep slurm R 0:11  3 node[2,6,14]
 193  sleep slurm R 0:15  3 node[5,9,13]
```



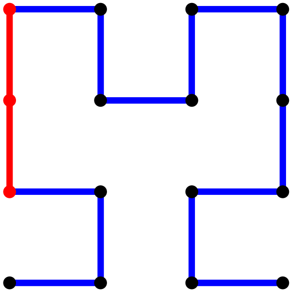
$2\mathcal{D}$, $\mathcal{S} = (4,4)$

Example - jobs allocation

Example - launch of 3 tasks

$n = 2$, $\mathcal{S} = (4, 4)$, with 1 node per switch
launch multiple srun -n3:

```
slurm$ srun -n3 sleep 50 &
slurm$ squeue
JOBID NAME  USER ST TIME  NODES NODELIST
 197  sleep slurm R 0:01  3 node[1,3-4]
 196  sleep slurm R 0:05  3 node[7,11,15]
 195  sleep slurm R 0:09  3 node[8,12,16]
 194  sleep slurm R 0:11  3 node[2,6,14]
 193  sleep slurm R 0:15  3 node[5,9,13]
```



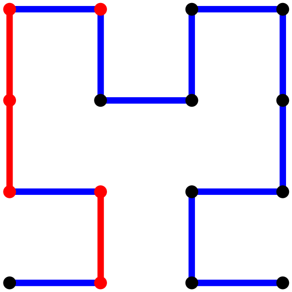
$2\mathcal{D}$, $\mathcal{S} = (4, 4)$

Example - jobs allocation

Example - launch of 3 tasks

$n = 2$, $\mathcal{S} = (4,4)$, with 1 node per switch
launch multiple srun -n3:

```
slurm$ srun -n3 sleep 50 &
slurm$ squeue
JOBID NAME  USER ST TIME  NODES NODELIST
 197  sleep slurm R 0:01  3 node[1,3-4]
 196  sleep slurm R 0:05  3 node[7,11,15]
 195  sleep slurm R 0:09  3 node[8,12,16]
 194  sleep slurm R 0:11  3 node[2,6,14]
 193  sleep slurm R 0:15  3 node[5,9,13]
```



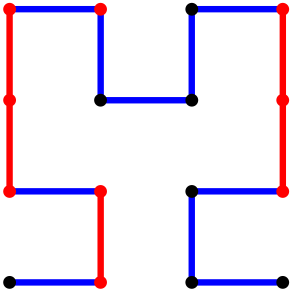
$2\mathcal{D}$, $\mathcal{S} = (4,4)$

Example - jobs allocation

Example - launch of 3 tasks

$n = 2$, $\mathcal{S} = (4, 4)$, with 1 node per switch
launch multiple srun -n3:

```
slurm$ srun -n3 sleep 50 &
slurm$ squeue
JOBID NAME  USER ST TIME  NODES NODELIST
 197  sleep slurm R 0:01  3 node[1,3-4]
 196  sleep slurm R 0:05  3 node[7,11,15]
 195  sleep slurm R 0:09  3 node[8,12,16]
 194  sleep slurm R 0:11  3 node[2,6,14]
 193  sleep slurm R 0:15  3 node[5,9,13]
```



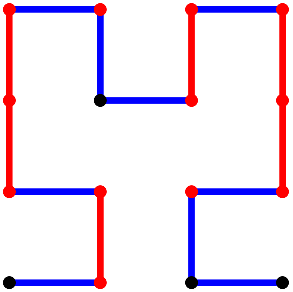
$2\mathcal{D}$, $\mathcal{S} = (4, 4)$

Example - jobs allocation

Example - launch of 3 tasks

$n = 2$, $\mathcal{S} = (4, 4)$, with 1 node per switch
launch multiple srun -n3:

```
slurm$ srun -n3 sleep 50 &
slurm$ squeue
JOBID NAME  USER ST TIME  NODES NODELIST
 197  sleep slurm R 0:01  3 node[1,3-4]
 196  sleep slurm R 0:05  3 node[7,11,15]
 195  sleep slurm R 0:09  3 node[8,12,16]
 194  sleep slurm R 0:11  3 node[2,6,14]
 193  sleep slurm R 0:15  3 node[5,9,13]
```



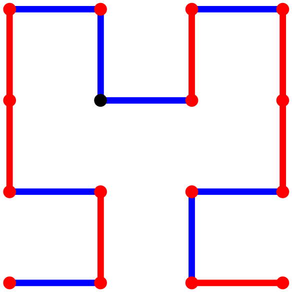
$2\mathcal{D}$, $\mathcal{S} = (4, 4)$

Example - jobs allocation

Example - launch of 3 tasks

$n = 2$, $\mathcal{S} = (4,4)$, with 1 node per switch
launch multiple srun -n3:

```
slurm$ srun -n3 sleep 50 &
slurm$ squeue
JOBID NAME  USER ST TIME  NODES NODELIST
 197  sleep slurm R 0:01  3 node[1,3-4]
 196  sleep slurm R 0:05  3 node[7,11,15]
 195  sleep slurm R 0:09  3 node[8,12,16]
 194  sleep slurm R 0:11  3 node[2,6,14]
 193  sleep slurm R 0:15  3 node[5,9,13]
```



$2\mathcal{D}$, $\mathcal{S} = (4,4)$

Example - jobs allocation

Example - launch of 800 tasks

$n = 6$, $\mathcal{S} = (2, 3, 3, 3, 5, 5)$ (1350 switches), with 1 node per switch
launch srun -n800 sleep 120&:

```
slurm$ srun -n800 sleep 120 &

slurm$ squeue

JOBID NAME  USER ST TIME  NODES NODELIST

200  sleep slurm R 0:50  800   node[0-11,18-29,36-39,42-45,54-65,72-83,90-93,96-99,

108-119,126-137,144-147,150-153,162-173,180-191,198-201,204-207,216-227,234-245,

252-255,258-261,270-281,288-299,306-309,312-315,324-335,342-353,360-363,366-369,

378-389,396-407,414-417,420-423,432-443,450-461,468-471,474-477,486-497,504-515,

522-525,528-531,540-551,558-569,576-579,582-585,594-605,612-623,630-633,636-639,

648-659,666-677,684-687,690-693,702-713,720-731,738-741,744-747,756-767,774-785,

792-795,798-801,810-821,828-839,846-849,852-855,864-875,882-893,900-903,906-909,918-929,

936-947,954-957,960-963,972-983,990-1001,1008-1011,1014-1017,1026-1037,1044-1055,1062-1065,

1068-1071,1080-1091,1098-1109,1116-1119,1122-1125,1134-1145,1152-1163,1170-1173,1176-1179,
```

Future work

- GHC with select cons_res
- Scalability and Efficiency evaluation
- Validate on a physical cluster
- Push to the community

Thanks for your attention!

Any questions?

Contact: mathis.clayer@atos.net