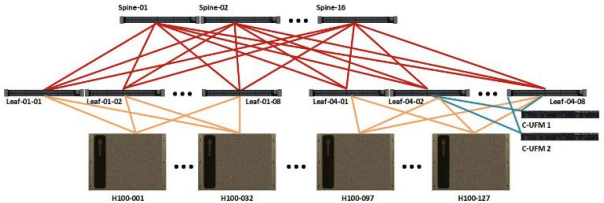
Gaining more control over node scheduling with the Topology/Block Plugin

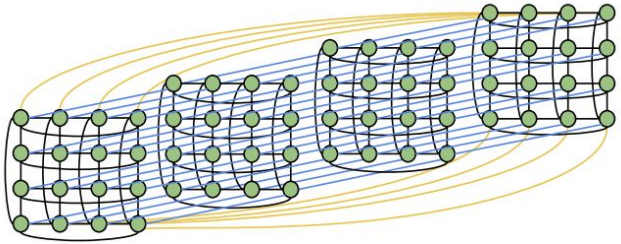
Vasileios Karakasis
Felix Abecassis
Craig Tierney
Douglas Wightman

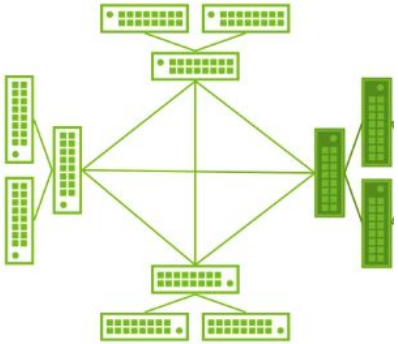
Slurm User Group '24 | September 2024

Optimizing System Usage By Understanding Network Topology

- Slurm has provided many methods to optimize scheduling with understanding of the network topology
  - Hierarchical Networks (ex: fat-tree, dragonfly+)
  - Three-dimensional Topologies (ex: torus)
- These plugins help Slurm select the “best” nodes at the time.
  - It does not implement a hard requirement or guarantees regarding node selection



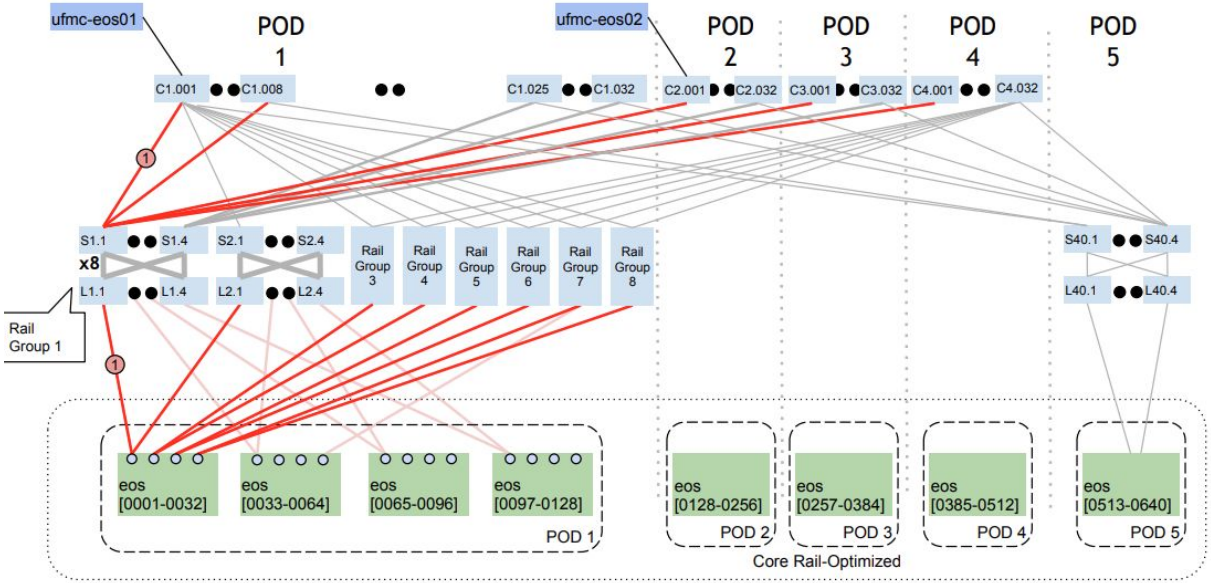




Fat Tree example is from: https://docs.nvidia.com/https:/docs.nvidia.com/dgx-superpod-reference-architecture-dgx-h100.pdf
Dragonfly example is from: https://docs.nvidia.com/networking/display/mlnxgwv814000/deployment+scenarios

Limitations to the Topology Plugins

- The plugins cannot guarantee any specific behavior.
- In the rail-optimized fat-tree below (8 rails per server), the topology plugin (topology/tree) cannot guarantee that a group of nodes would be allocated to a job that are connected to a single leaf.
- For achieving peak performance, this level of control is sometimes necessary



NVIDIA GB200 NVL72

- NVIDIA GB200 NVL72 is NVIDIA's scale out product
  - *"The system is the data center"*

- NVIDIA GB200 NVL72 Rack
  - 18 nodes, each with 2 Grace CPUs and 4 Blackwell GPUs
  - NVLink-Chip-to-Chip (C2C) interface between CPU and GPU provides coherent access to a larger memory space.
  - Each GPU has a dedicated CX7

- All the GPUs in the tack are connected via external NVLink
  - Each GPU has 18 NVLink ports @ 100 GB/s each
  - The NVLink network provides several benefits
    - ~ 5x increase in bandwidth over InfiniBand
    - GPUs can directly access the memory of other GPUs

- Groups of NVL domains can be connected via InfiniBand or Ethernet

Complexities of Scheduling Systems with multiple networks

- While not the most optimal, applications can communicate across InfiniBand
- However, applications relying on maximum bandwidth between GPUs must be scheduled within a single NVLink domain
  - Hybrid communication methods are possible (fast over NVLink, slow over InfiniBand) but allocations must be grouped as required
- Applications requiring maximum bandwidth between GPUs
  - Tensor parallel communications in LLM models
  - Large model AI inference need large GPU memory footprints to achieve peak performance
  - Fast Fourier Transform
  - AI Recommender

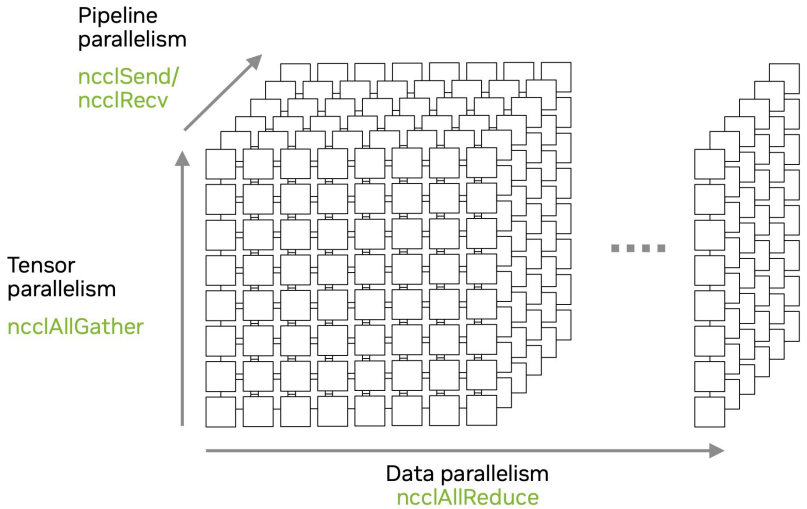


Image from ***Training Deep Learning Models at Scale: How NCCL Enables Best Performance on AI Data Center Networks***
https://www.nvidia.com/en-us/on-demand/session/gtc24-s62129/

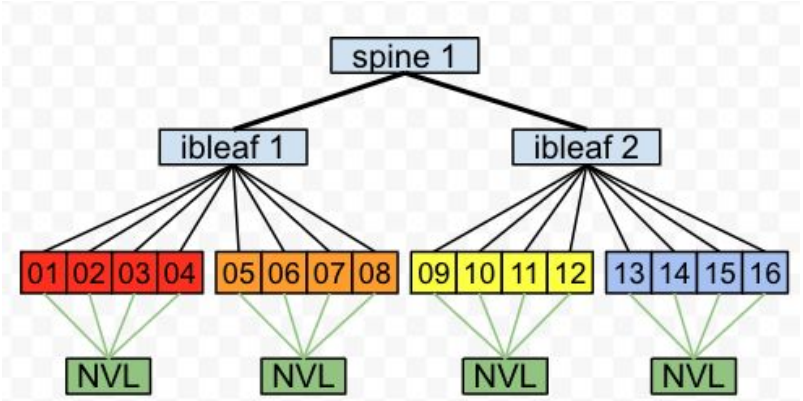
Topology/Block

- Topology/block plugin was introduced in Slurm 23.11
  - The plugin was updated and redesigned in 24.05

https://slurm.schedmd.com/topology.conf.html#SECTION_topology/block

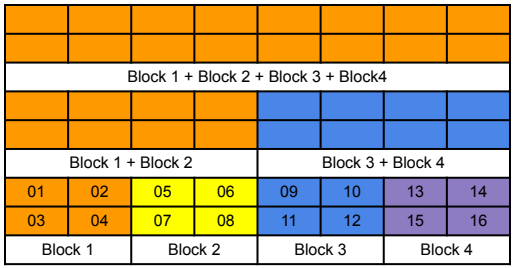
- The Block plugin provides hierarchical scheduling across blocks of nodes
- For the example system shown, the block plugin could be defined as:

```
################################################################
# Slurm's network topology configuration file for use with
# the topology/block plugin
################################################################
BlockName=block1 Nodes=node[01-04]
BlockName=block2 Nodes=node[05-08]
BlockName=block3 Nodes=node[09-12]
BlockName=block4 Nodes=node[13-16]
BlockSizes=4,8
```



Topology/Block – What is a block?

- A block is a consecutive range of nodes
- Blocks cannot overlap with each other
- All nodes in a block are allocated to a job before the next block is used
- The planning block size is the smallest block size configured
  - In the example on the previous slide, the planning block size is 4 nodes
- Every higher block level size is a power of two than the previous one



Rules and Strategies for Defining Blocks

- Block is defined as a list of non-overlapping **`Nodes`** with a **`BlockName`**
  - Not all nodes need to be listed as apart of blocks (and will be scheduled without block consideration, ex. CPU nodes)
  - Only one topology plugin can be specified at the time
- **`BlockSizes`** defines the sizes of blocks
  - First block size is the planning block size
  - Higher level blocks must be a power of two of the planning block size
- The number of nodes in a block can be greater than the size of the planning block
  - Blocks may have different sizes
- It is assumed that nodes are always allocated as `--exclusive` (never shared)

```
###############################################################
# Slurm's network topology configuration file for use with
# the topology/block plugin
###############################################################
BlockName=block1 Nodes=node[01-04]
BlockName=block2 Nodes=node[05-08]
BlockName=block3 Nodes=node[09-12]
BlockName=block4 Nodes=node[13-16]
BlockSizes=4,8
```

Control node allocation

Job requests with sbatch

`--segment=<nodes>`

- Specify the number of nodes to group together
- The size of the segment must be less or equal to the planning block size
- Use of --segment does not guarantee that segments will be placed on different blocks

`--exclusive=topo`

- Jobs can request that no other jobs be placed on the same block
- When combined with `--segment`, it does not guarantee that segments will be placed on different blocks
- This is useful for benchmarking and other application performance work
- When used, nodes left idle are not accounted for against the job

Controlling Job Placement Examples

- We will walk through some examples on the expected behavior of the block scheduler
- The specs on the example system are:
  - Nodes are connected by InfiniBand and NVlink
  - **18 nodes** per NVLink Domain
- We are only defining a block level to represent the entire NVLink domain
  - There is no higher level block definition for the IB network.

- Example topo.cfg

```
######################################################################
# Slurm's network topology configuration file for use with the
# topology/block plugin
######################################################################
BlockName=block01 Nodes=node[001-018]
BlockName=block02 Nodes=node[019-036]
BlockName=block03 Nodes=node[037-054]
BlockName=block04 Nodes=node[055-072]
BlockName=block05 Nodes=node[073-090]
BlockName=block06 Nodes=node[091-108]
BlockName=block07 Nodes=node[109-126]
BlockName=block08 Nodes=node[127-144]
BlockSizes=18
```

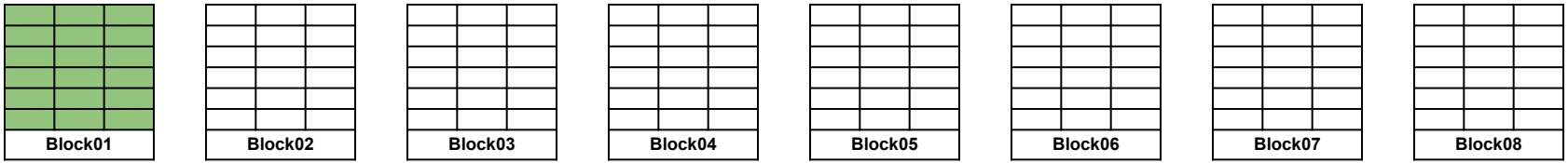
The following examples do not show the definitive choices of what the Slurm scheduler will do. They do represent one version of the 'best case" that the topology/block plugin will do.

Scheduling Examples

Allocate job which is the size of the planning block on an empty system

```
# sbatch -N18 ...
```

Full block allocated, any block could be selected



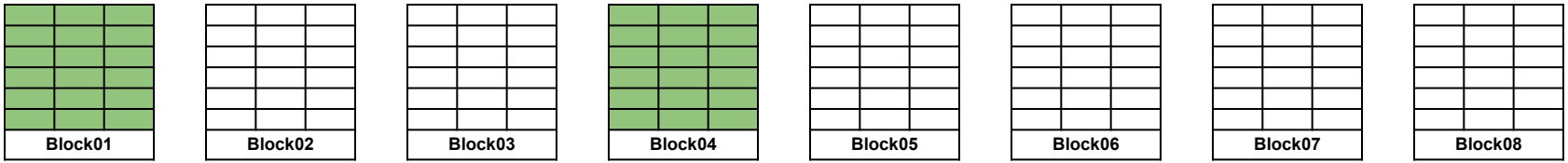
Color	Definition
	Idle
X	drained/down or otherwise unavailable
	Example job
Other	Unique colors per existing job

Scheduling Examples

Allocate job larger than the planning block size

```
# sbatch -N36 ...
```

Two blocks allocated, not consecutive since only single BlockSizes specified. **`BlockSizes=18`**



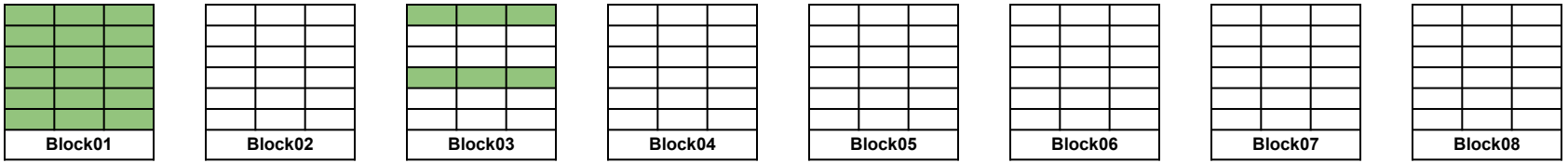
Color	Definition
	Idle
X	drained/down or otherwise unavailable
	Example job
Other	Unique colors per existing job

Scheduling Examples

Allocate job larger than the planning block size

```
# sbatch -N24 ...
```

Full block is filled before 2nd block is used



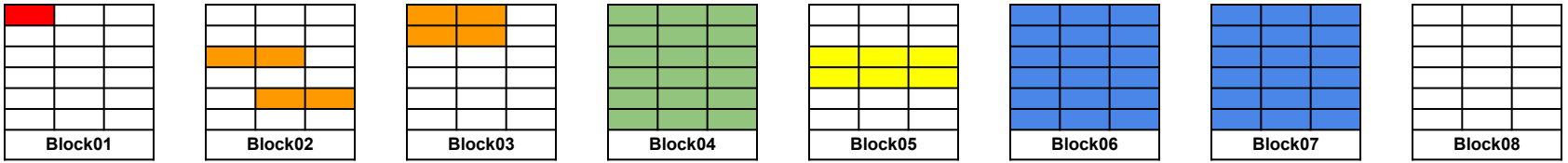
Color	Definition
	Idle
X	drained/down or otherwise unavailable
	Example job
Other	Unique colors per existing job

Scheduling Examples

Allocate job that is the size of the planning block with existing jobs

```
# sbatch -N18 ...
```

Job has to fit onto full block



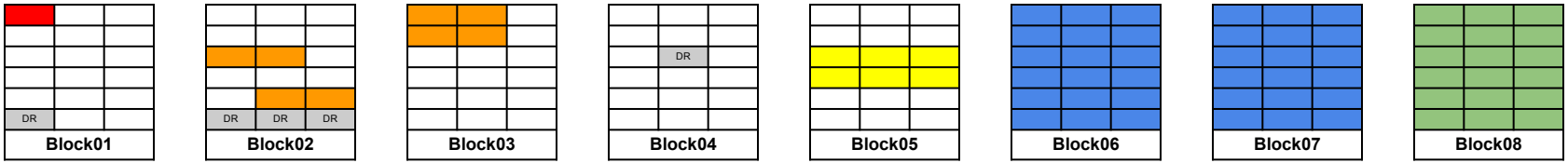
Color	Definition
	Idle
X	drained/down or otherwise unavailable
	Example job
Other	Unique colors per existing job

Scheduling Examples

Allocate job that is the size of the planning block with existing jobs, with unavailable nodes

```
# sbatch -N18 ...
```

Job has to fit on full block, with no down/drained nodes.



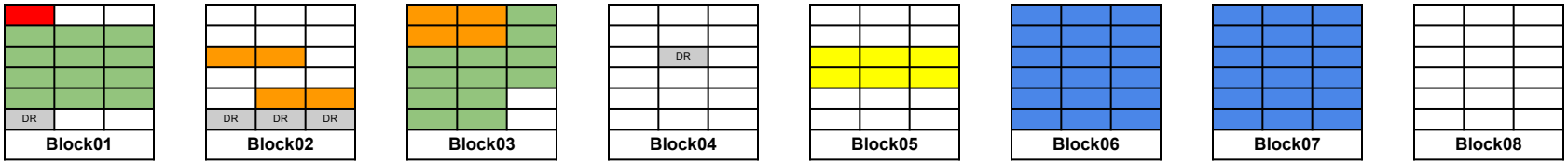
Color	Definition
	Idle
DR	drained/down or otherwise unavailable
	Example job
Other	Unique colors per existing job

Scheduling Examples

Allocate job with --segment, with unavailable nodes

```
# sbatch -N24 --segment=12 ...
```

With --segment=12, job can be placed on blocks with existing jobs



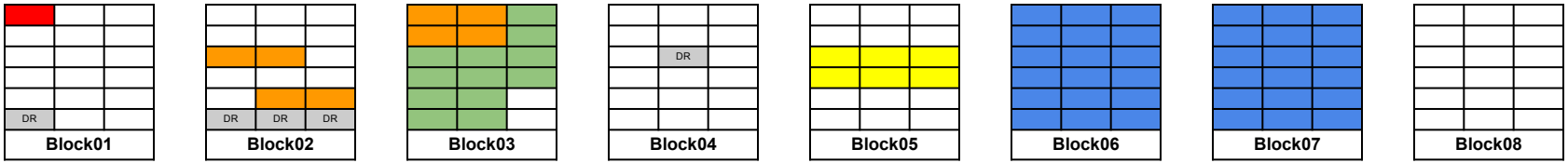
Color	Definition
	Idle
DR	drained/down or otherwise unavailable
	Example job
Other	Unique colors per existing job

Scheduling Examples

Allocate job with --segment, with unavailable nodes

```
# sbatch -N12 --segment=6 ...
```

Two segments may be placed on the same block



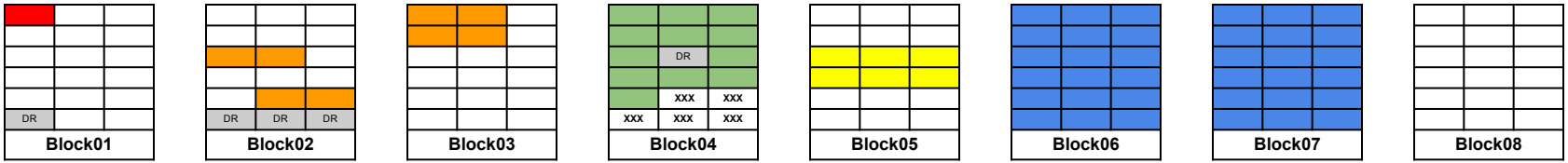
Color	Definition
	Idle
DR	drained/down or otherwise unavailable
	Example job
Other	Unique colors per existing job

Scheduling Examples

Allocate job with --exclusive=topo

```
# sbatch -N12 --exclusive=topo ...
```

With `--exclusive=topo`*, job must be placed on block with no other jobs.*



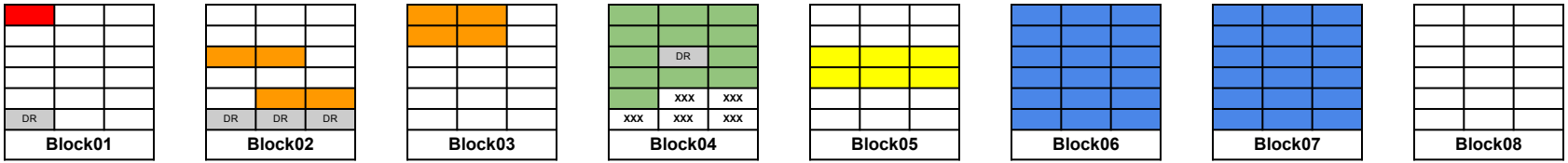
Color	Definition
	Idle
DR	drained/down or otherwise unavailable
	Example job
Other	Unique colors per existing job

Scheduling Examples

Allocate job with --exclusive=topo and --segment

```
# sbatch -N12 --segment=6 --exclusive=topo ...
```

With `--exclusive=topo`, segments from the same job may still be placed on the same block



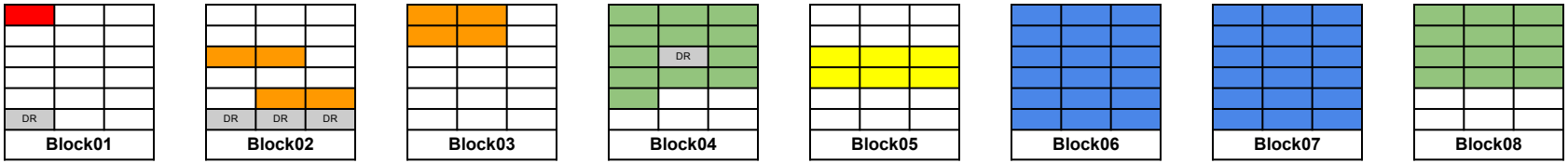
Color	Definition
	Idle
DR	drained/down or otherwise unavailable
	Example job
Other	Unique colors per existing job

Scheduling Examples

Allocate job with --exclusive=topo and --segment

```
# sbatch -N24 --segment=12 --exclusive=topo ...
```

With `--exclusive=topo` *and* `--segment`*, blocks are not shared but segments can fit where nodes may be drained*



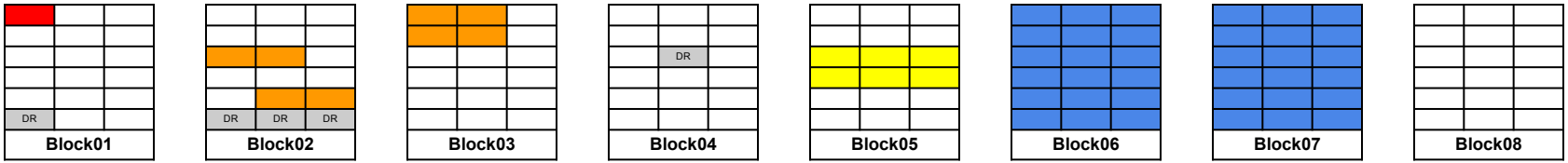
Color	Definition
	Idle
DR	drained/down or otherwise unavailable
	Example job
Other	Unique colors per existing job

Scheduling Examples

Block scheduling can lead to additional fragmentation

```
# sbatch -N36 --exclusive=topo ...
```

This job cannot be scheduled since there not 2 full blocks available for scheduling.



Color	Definition
	Idle
DR	drained/down or otherwise unavailable
	Example job
Other	Unique colors per existing job

High Availability Scheduling Options

- Option 1: A job can use --segment and not depend on every node in a block being available
  - Pro
    - Users are given more flexibility to ensure their jobs will run
    - Unused nodes can be used by small jobs
  - Cons
    - Using --segment means higher order block sizes beyond the planning block size are not supported
- Option 2: Set BlockSizes that are smaller the the full node range. Ex:

```
BlockName=block01 Nodes=node[001-018]
BlockSizes=16
```

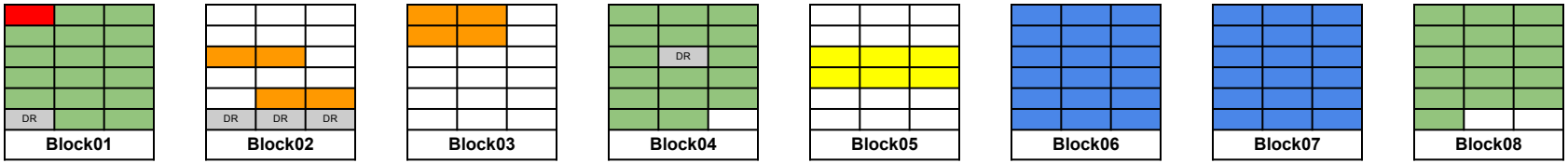
  - Pros
    - Use is transparent to the users
    - Unused nodes can be used by smaller jobs
  - Cons
    - The user may need to know the size of the block sizes to get the behavior they want
    - ***This prevents all of the nodes in the NVLink domain from being used by a single job***

Scheduling Examples

HA-like options using the block scheduler

```
# sbatch -N48 --segment=16 ...
```

Using `--segment`*, multi-block jobs can fit around downed and allocated nodes*



Color	Definition
	Idle
DR	drained/down or otherwise unavailable
	Example job
Other	Unique colors per existing job

Conclusions & Next Steps

- The Topology/block plugin guarantees that nodes will be allocated based on the defined network topology
- Applications requiring maximum bandwidth and shared memory access across GPUs are guaranteed to get an optimal placement
- The `--segment` option can mitigate the fragmentation inherent to block scheduling and increase cluster utilization
- The `--segment` option gives flexibility to users in the trade-off between absolute performance and quicker job scheduling times

Next steps:

- Continue to understand the behavior of the plugin and better optimize its use and overall system utilization
- Continue to work with SchedMD to add more features to the plugin to improve flexibility and utilization

Acknowledgements

- Thanks to SchedMD collaborators
  - Tim Wickberg
  - Jess Arrington
  - Ben Roberts
  - Dominik Bartkiewicz
  - And the rest of the SchedMD team that made this happen