# Sliding Substitution of Failed Nodes

Atsushi Hori, Kazumi Yoshinaga,
Yutaka Ishikawa
RIKEN AICS
Thomas Herault, Aurélien Bouteiller,
George Bosilca
University of Tennessee, ICL

# Motivation

- Having spare node set seems to be the last resort
  - “in such case, spare node can be used.”

- Having spare node is not the answer, but new research issue

# Fault Resilience

- Fault tolerance in Exa-flops era
  - High failure rate
  - High I/O bandwidth requirement

- User-level fault resilience
  - Less I/O bandwidth required
  - e.g., ULFM (User-Level Fault Mitigation)

> *ULFM is not a recovery strategy, but a minimalistic set of building blocks for more complex recovery strategies.*

- **We need a recovery strategy !!**

# Survival from Node Failure

- Jobs with dynamic load balancing
  - e.g., Task bag, PIC, ...
  - Job shrinking to exclude failed nodes
  - Tasks running on failed node(s) are migrated to live nodes

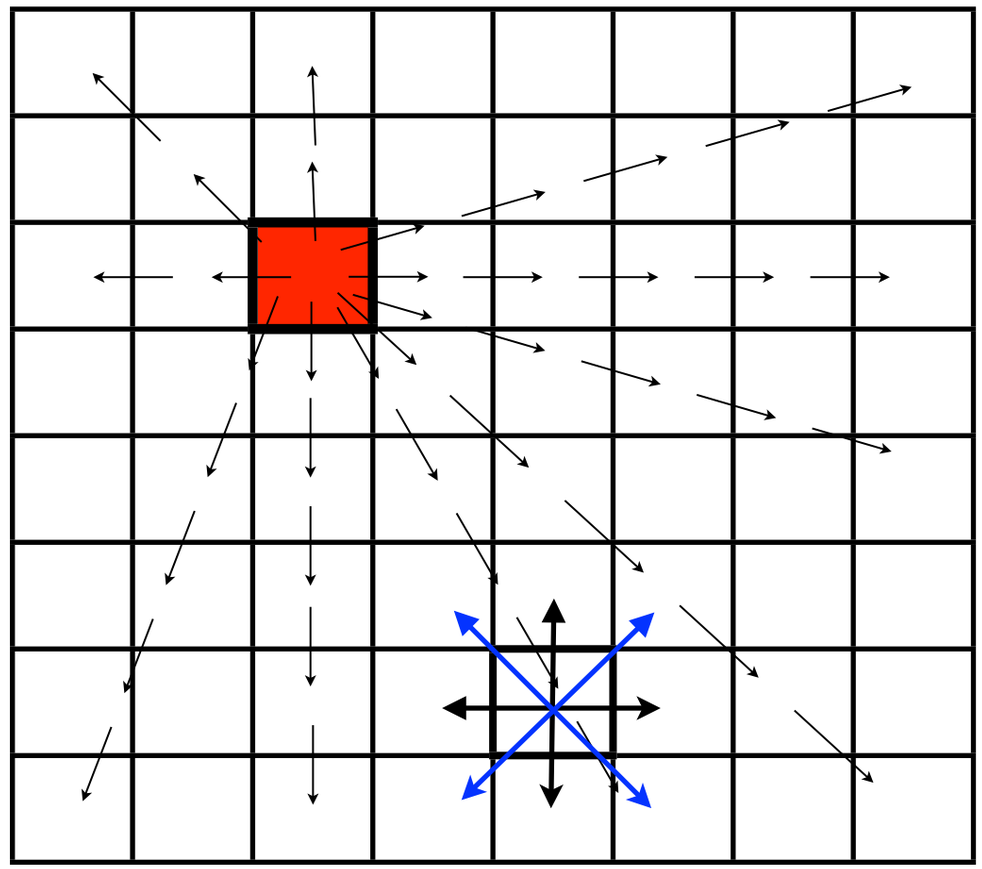
- Jobs without dynamic load balancing
  - e.g., Stencil computation, ...
    - Very difficult to balance load
  - Having spare nodes seems to be the answer ...

# Stencil Computation

- Survival from a node failure
  - Load balancing
  - Preserving communication pattern
  - Less code modification

Shift the load on to healthy nodes

New complex communication pattern



# Spare Node

- In an error handler (of ULFM, for example)
  - create a new MPI communicator to
    - exclude the failed node, and
    - include a spare node.
  - then, migrate the task running on the failed node to the spare node
  - No change in the kernel part of application

- However, at the network level, the regular stencil communication pattern can be lost !

# Is spare node really the answer ?

- Our scope
  - Is there any penalty? If any, how much?
  - How spare nodes should be allocated?
  - How many spare nodes should be allocated?
  - How failed nodes should be substituted be spare nodes?
- Out of scope
  - How (soft/hard) errors are detected
  - How checkpoints are taken
  - How tasks are migrated

# Spare Node Penalty (1)

- Spare node allocation and node utilization

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 | 32 | 33 | 34 | 35 | | |
| | | | | | | | |
| | | | | | | | |

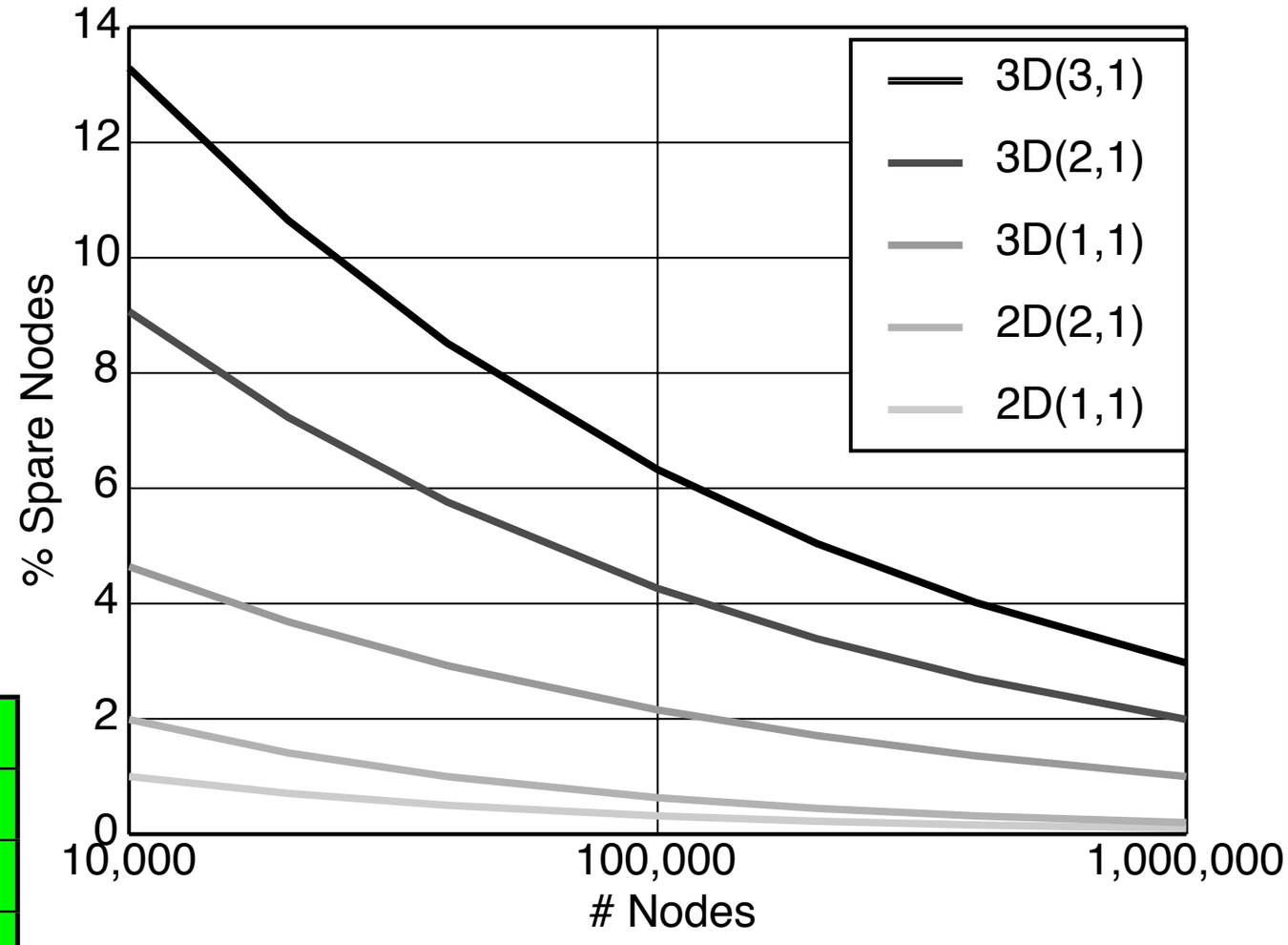
**2D(2,2)**

| | | | | | | |
|---|---|---|---|---|---|---|
| 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 | 32 | 33 | 34 | 35 | |
| | | | | | | |

**2D(2,1)**

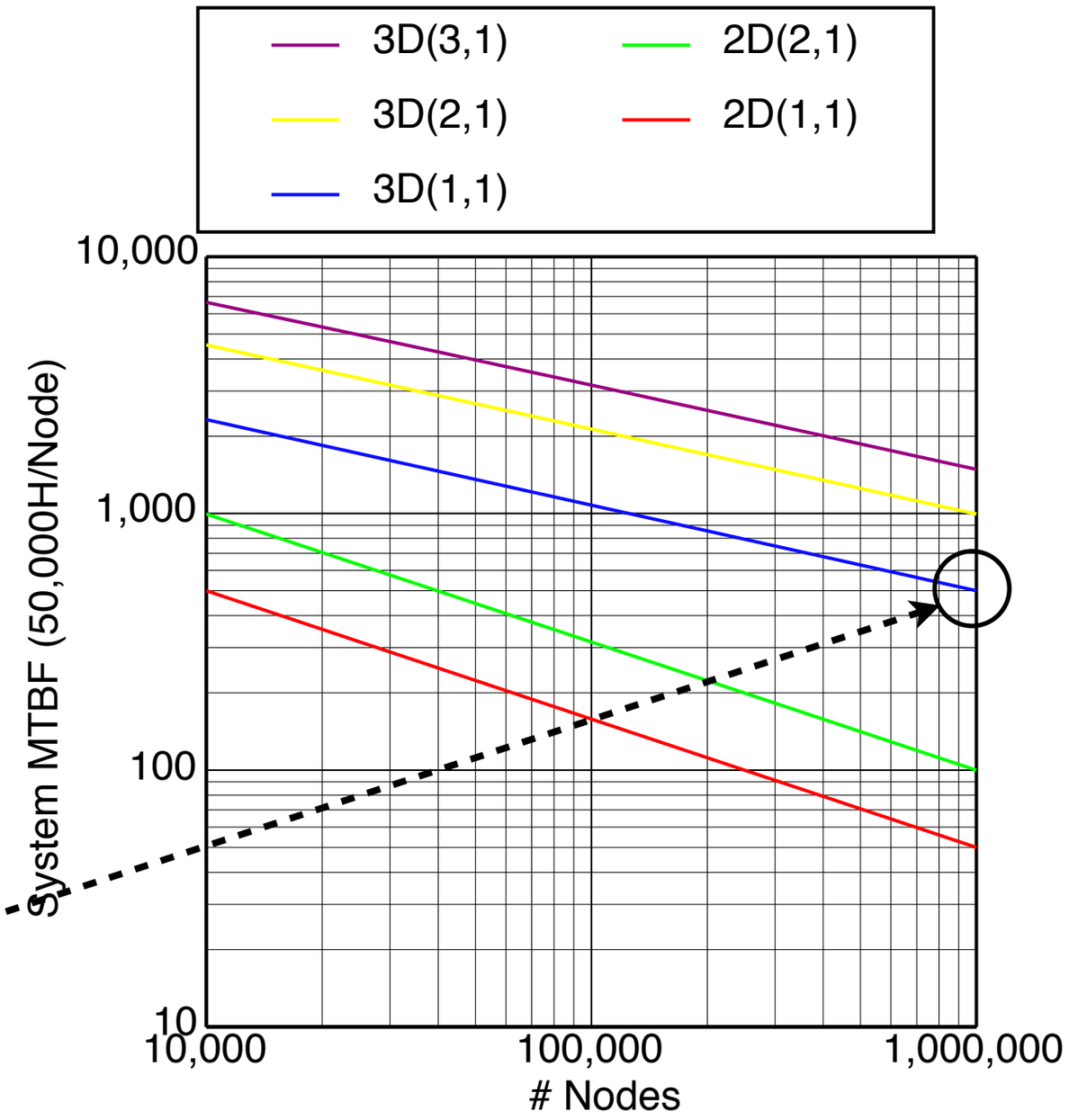
| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | |
| 6 | 7 | 8 | 9 | 10 | 11 | |
| 12 | 13 | 14 | 15 | 16 | 17 | |
| 18 | 19 | 20 | 21 | 22 | 23 | |
| 24 | 25 | 26 | 27 | 28 | 29 | |
| 30 | 31 | 32 | 33 | 34 | 35 | |

**2D(1,1)**



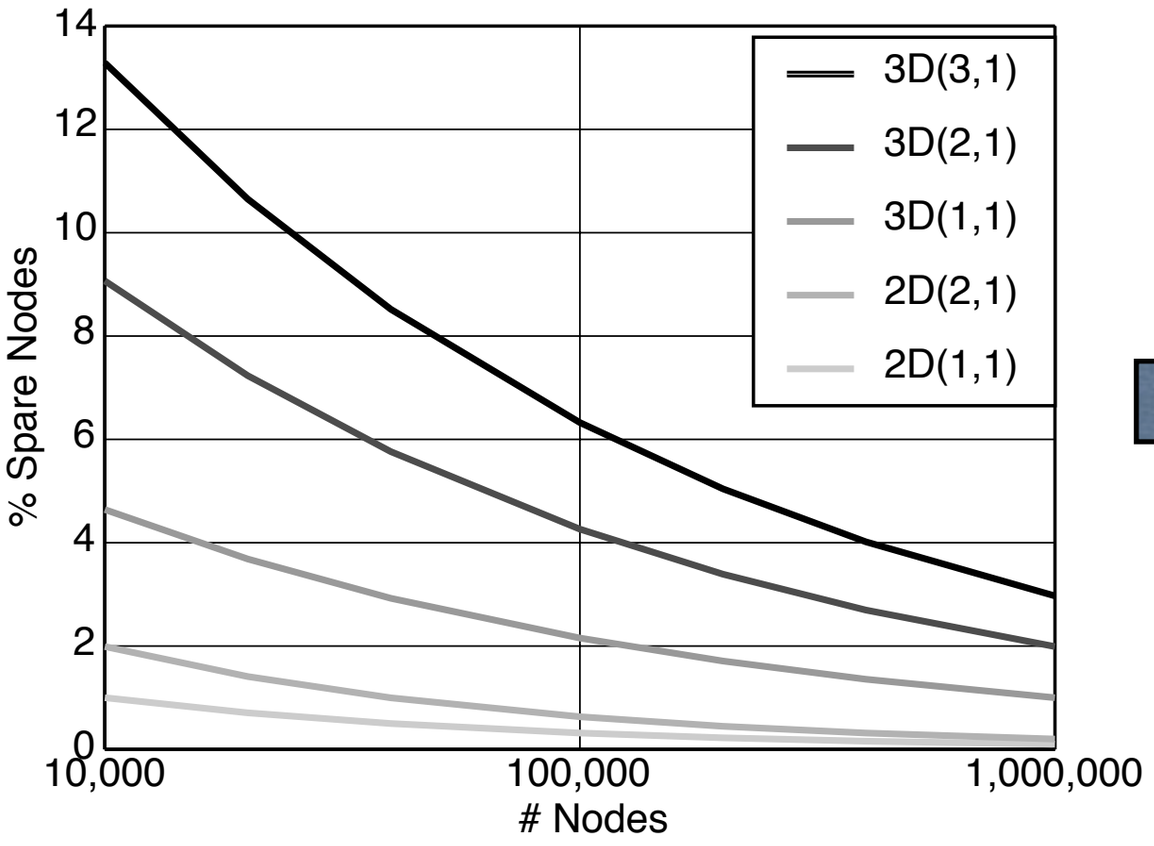
# How many spare nodes ?

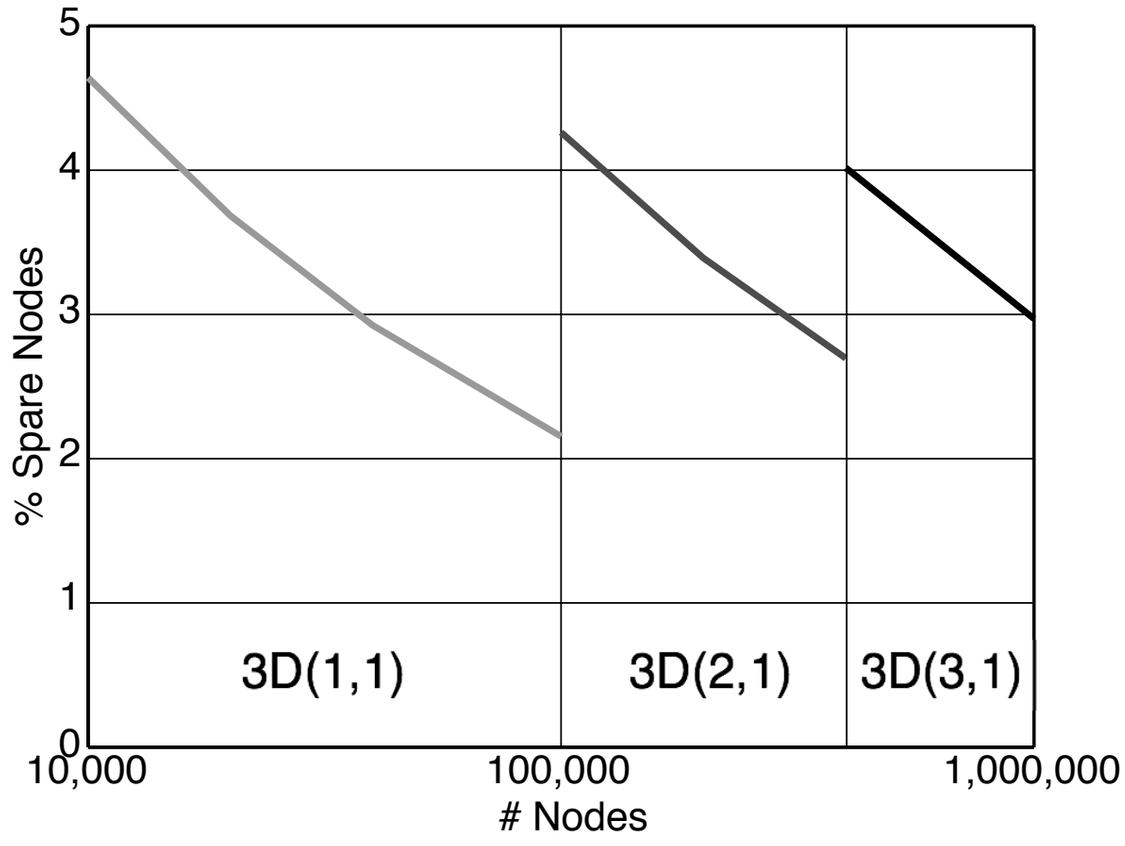
- MTBF of a node
  - 50,000 Hr. ≈ 5 Years
- MTBF of Exa ($10^6$ nodes)
  - 0.05 Hr. = 3 Min.
- #Spare = 10,000 (1%)
  - 500 Hr. ≈ 20 Days
- $10^4$ out of $10^6$



# Spare Node Allocation

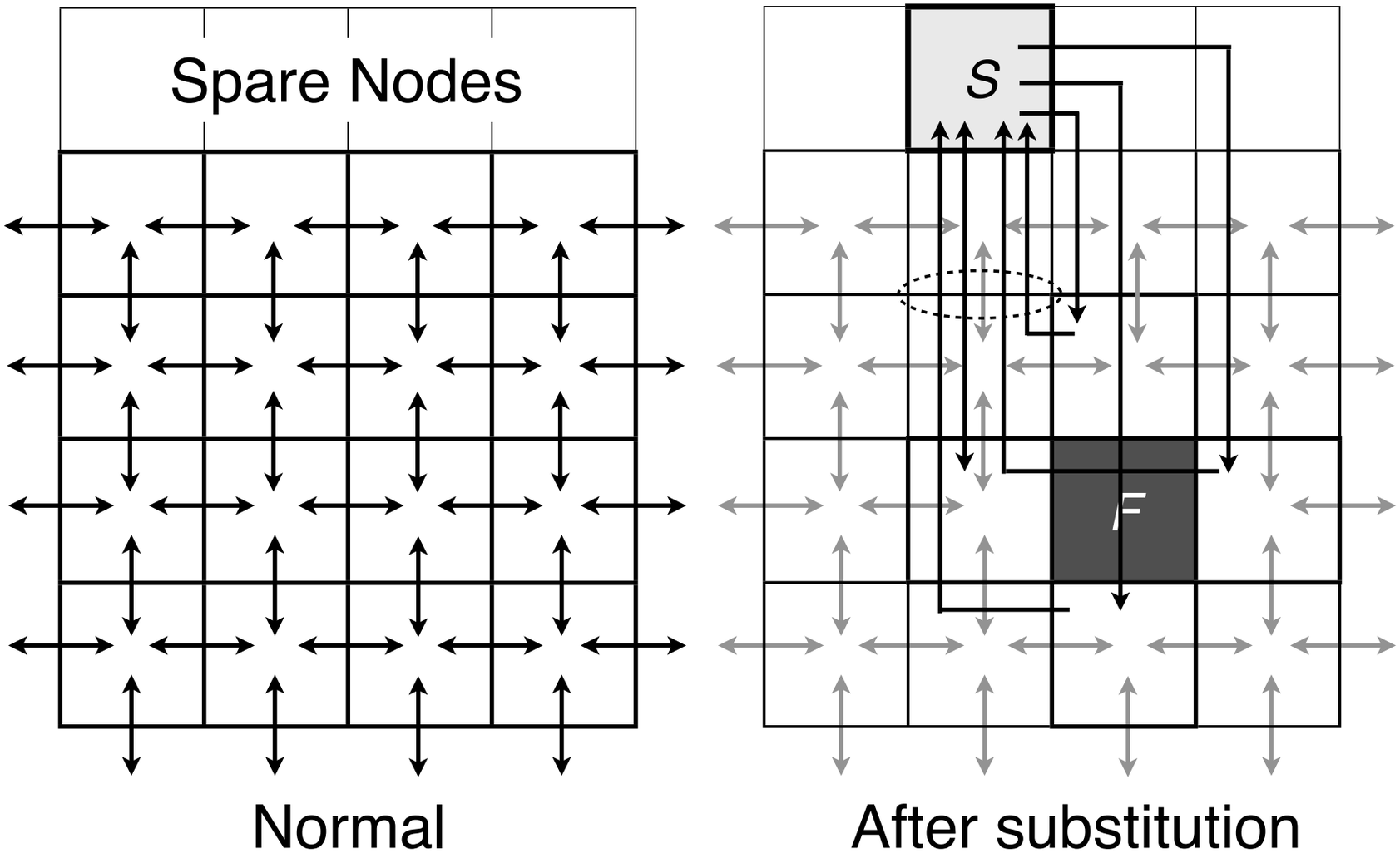
- Changing spare node allocation method according to the number of nodes



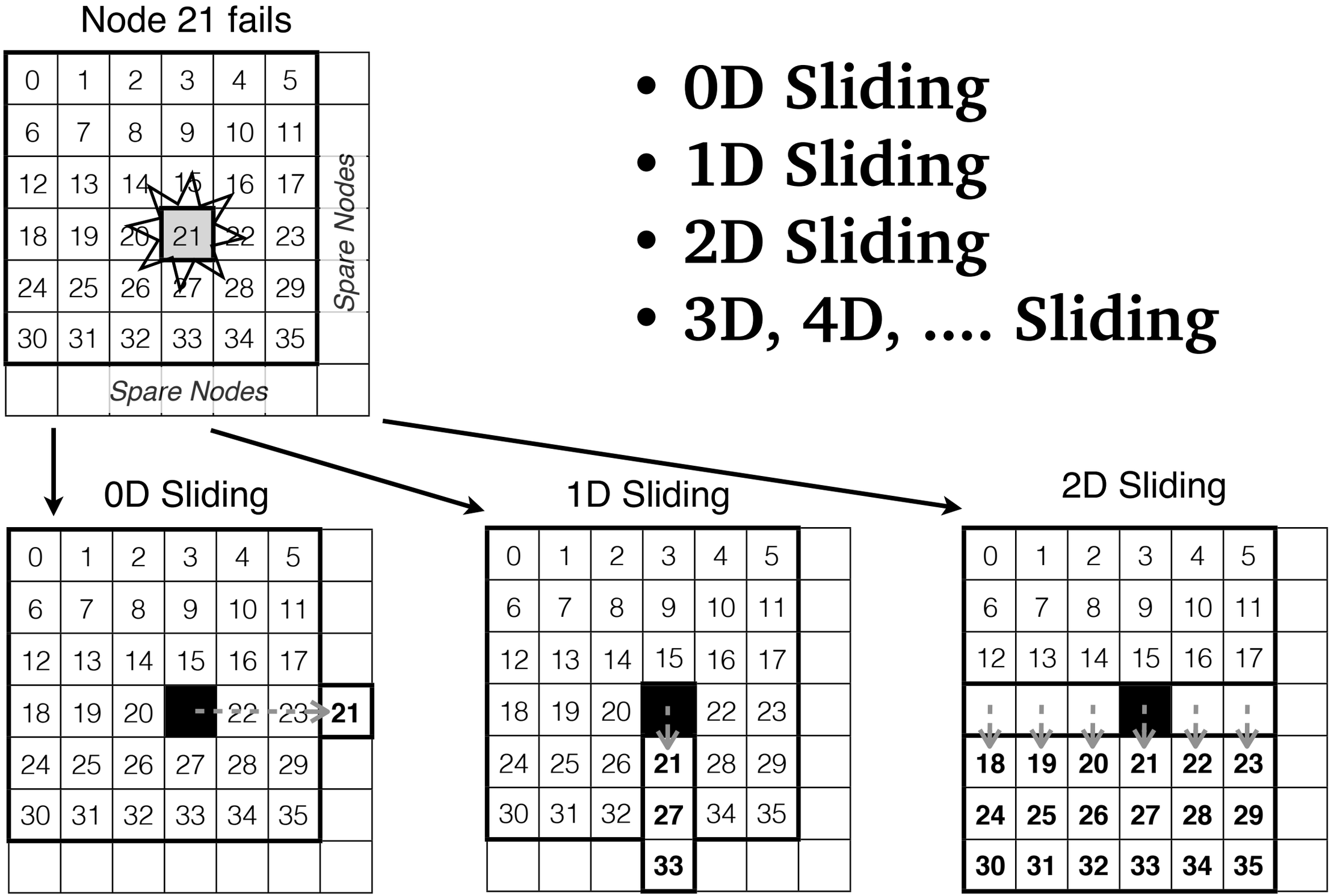
# Spare Node Penalty (2)

- Possibility of communication performance degradation
  - 5P Stencil communication pattern



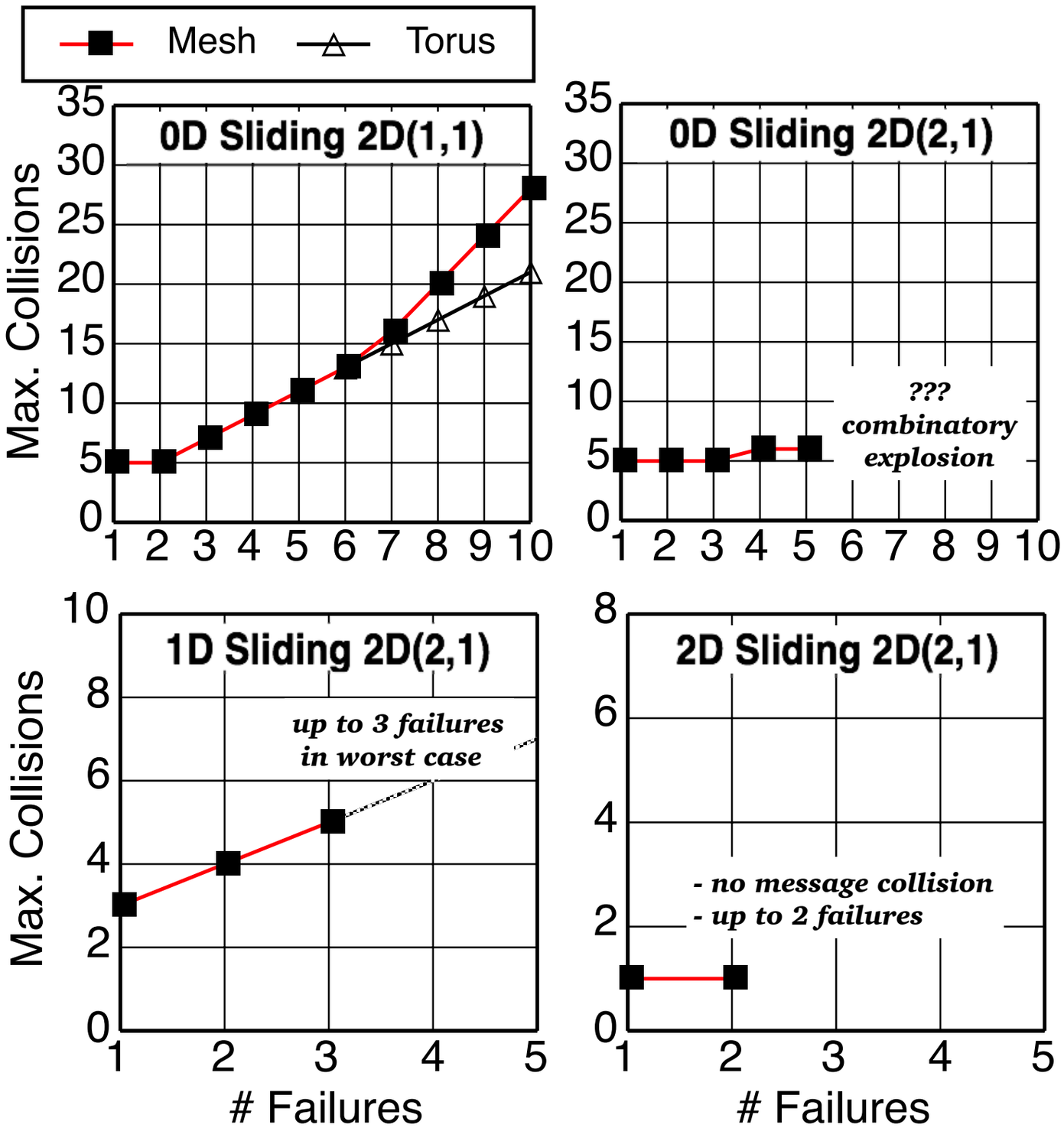

***2D Cartesian Network and XY Routing***

# Sliding Substitution




- 0D Sliding
- 1D Sliding
- 2D Sliding
- 3D, 4D, .... Sliding

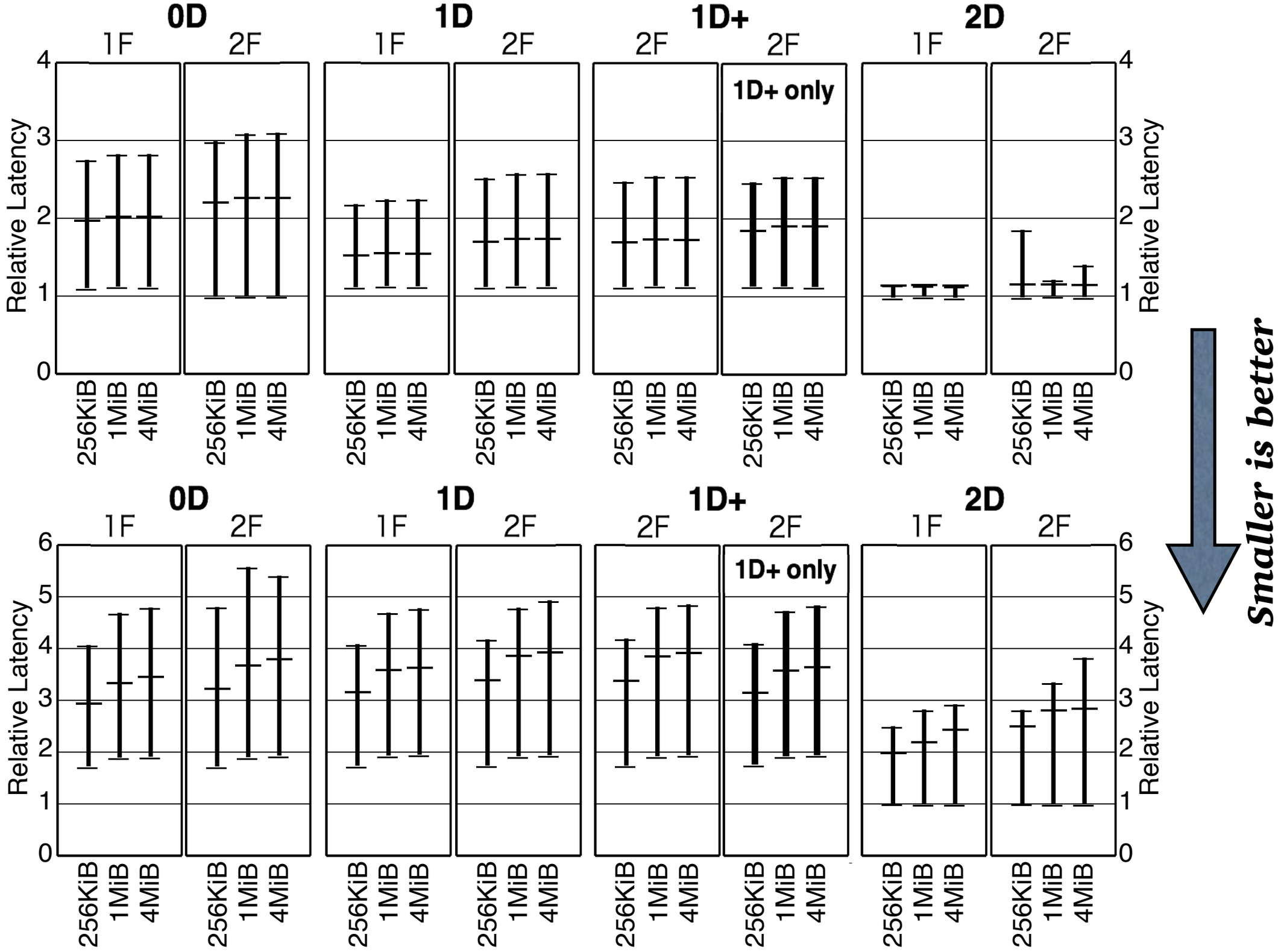
# 5P Stencil on 2D Network




- **Simulated Results**
- **Spare Allocation**
  - **2D(2,1) > 2D(1,1)**
- **Max. Failure**
  - **0D: up to #Spare**
  - **1D: 3 (or more)**
  - **2D: up to 2 (2D Cart. Topo.)**
- **Comm. Perf.**
  - **2D > 1D > 0D**

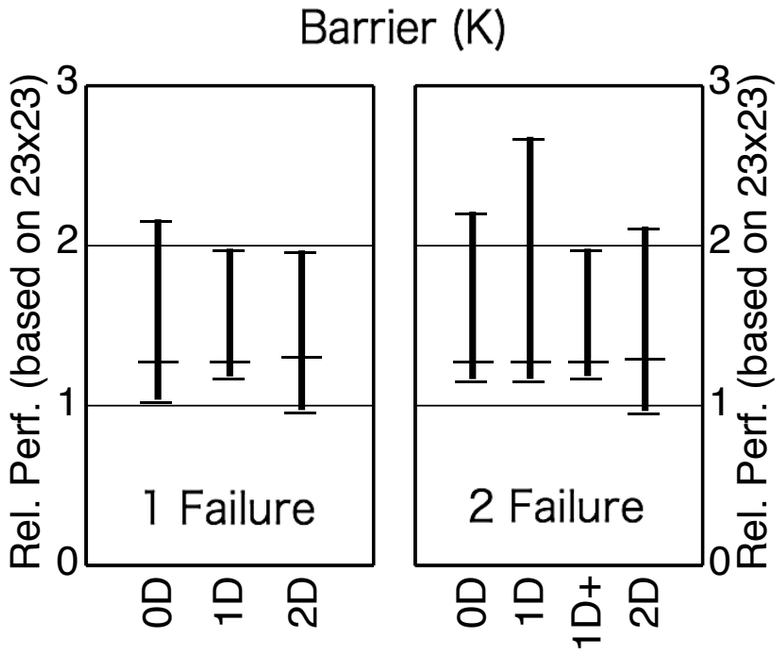
# 5P Stencil Comm. Perf.

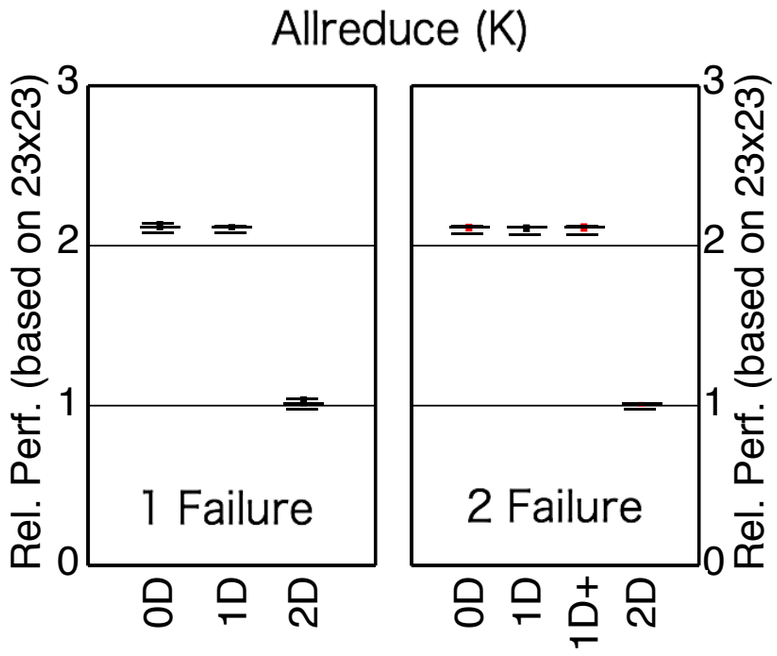
the K

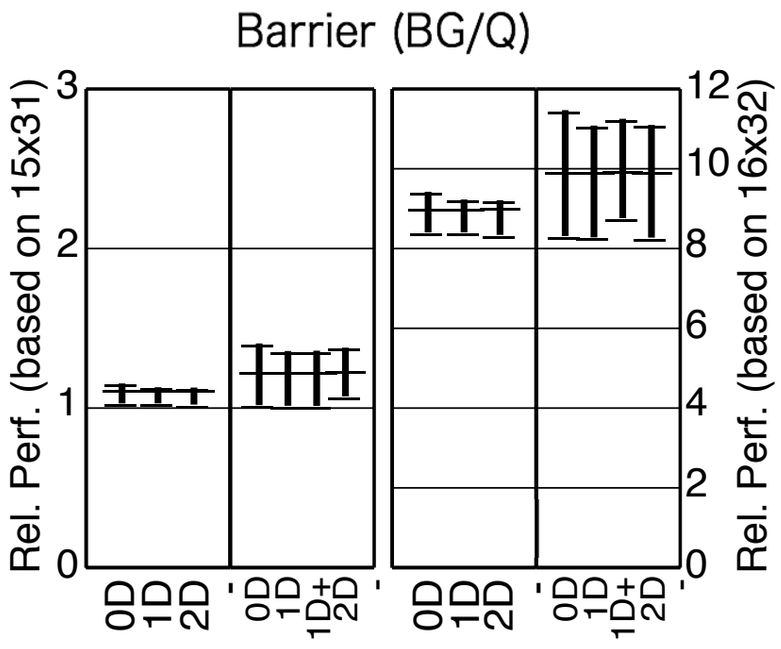


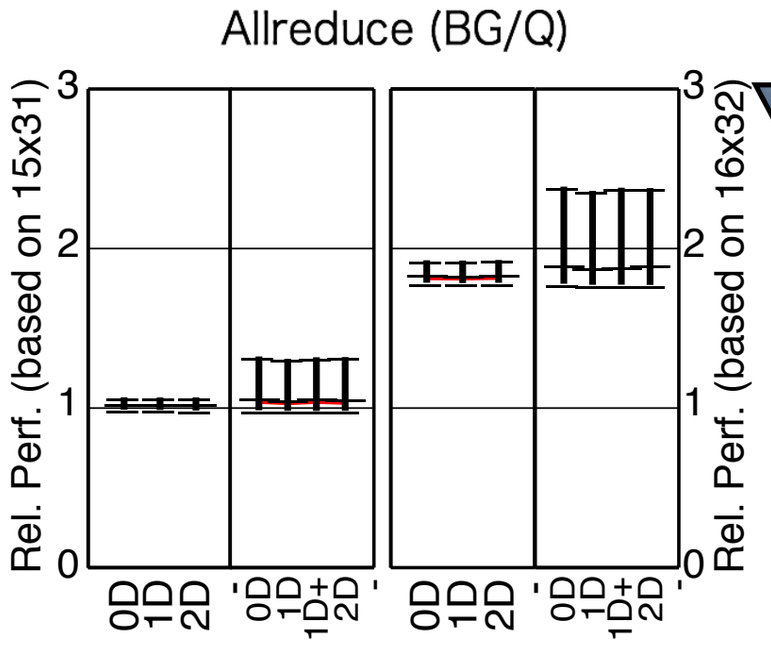

BG/Q

# Collective Performance




*Smaller is better*

- On K and BG/Q, collective ops are optimized for their network.
- Having spare nodes makes the optimization very difficult.
- BG/Q' optimization works only with **MPI_COMM_WORLD**

# Summary

- Study on spare node substitution has just begun

- Comm. perf. degradation is observed
  - 5P stencil :
    - Simulation: up to 100 times larger latency
    - Experiment: < 20 times larger latency
  - Collective : up to 12 times larger latency

# Current and Future Work

- Evaluations with real applications
- Node-Rank re-mapping algorithms, or better substitution methods
- Dragonfly and/or Fat-tree network ?
  - Experiments using Tsubame 2.5 (Fat-tree) is scheduled

- At this moment, it is still unclear if having spare nodes is a promising technique

# Acknowledgement

Thank to

**Dr. Norbert Attig**

**at Jülich Supercomputing Center**

**to give us a chance to use JUQUEEN.**