Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

# Analytic Modeling of Idle Waves in Parallel Programs: Communication, Cluster Topology, and Noise Impact

Supervisor: Jannek Squar

Phuong Thao Le

Presentation date: 11.01.2022

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

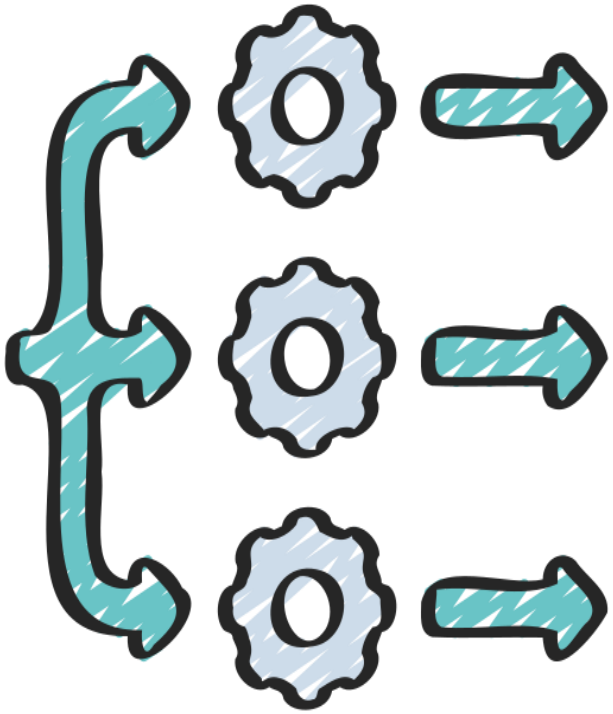
# Table of Content

# 1. Introduction

Due to the need to compute enormous computer resources (time, memory).

→ Parallel Computing:

- Solving a task by simultaneous use of multiple processors, which are components of a unified architecture.
- There might be disturbances such as system and network noise, delays caused by one-off events, etc.



# 1. Introduction

What are idle waves?

- Long one-off delays from one process cause periods of idleness, which later ripple through the system and affect other processes.
- Idleness: a state of delay, where the process needs to wait for information.

Consequences:

- Delay performance of the application
- Desynchronization among processes

→ Automatic communication overlap

# 1. Introduction

**Goals of the study:**

- Analytic modelling of the propagation speed of Idle waves in scalable code with respect to:
  - Communication topology
  - Communication concurrency.
- Interaction of idle waves with MPI collectives.
- Initiation of idle wave decay and analyse of decay rate:
  - Topological Decay
  - Noise-induced Decay

# 2. MPI

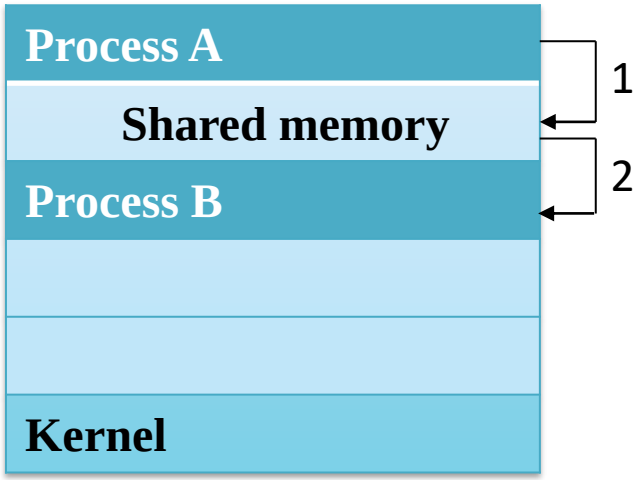
**Interprocess communication:**

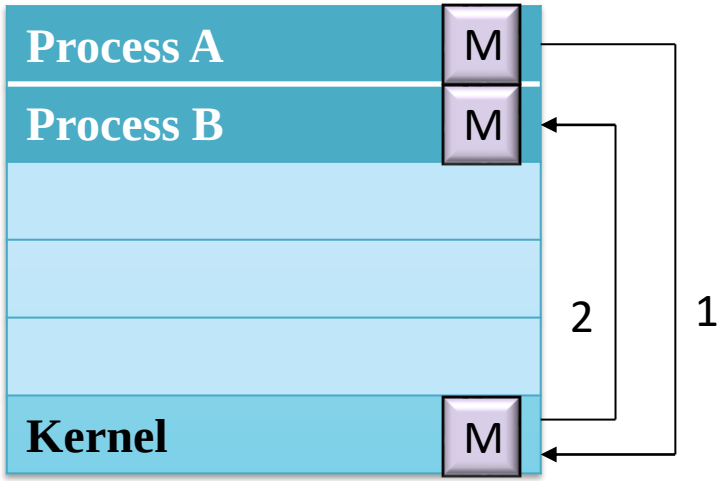
- Processes executing concurrently in the operating system may be independent or cooperating processes.
  - Independent: Processes not affecting each other
  - Cooperating: Processes share data with each other and thus have effects on each other.
- Advantages of cooperating processes: Information sharing, computation speedup, convenience, modularity.

# 2. MPI

- Cooperating processes require an interprocess communication (IPC) mechanism that allows exchanging data and information.
- Two fundamental models:
  - Shared memory
  - Message passing

# 2. MPI




**Shared Memory**: Exchanged Data are written and read on the shared memory region.

**Message Passing**: Communication takes place by means of messages exchanged between processes.

# 2. MPI

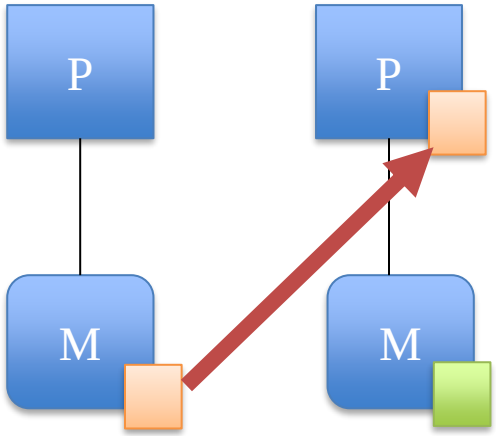
## MPI : Message-passing library interface specification

- A standardized and portable message-passing standard designed to function on parallel computing architectures.
- Useful in distributed memory environment, where communication processes located on different computer nodes.
- Language bindings: C, C++, Fortran

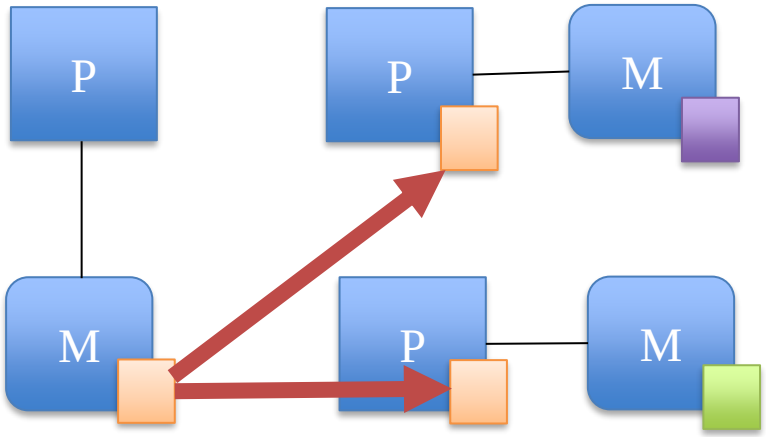
# 2. MPI

- Two main operations:
  - Send(messages) and Receive(messages)

Point to Point communication




Collective functions



# 2. MPI

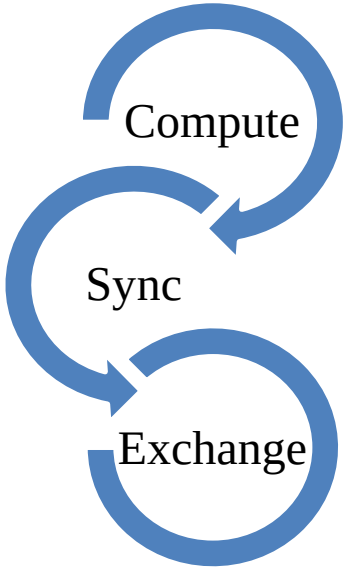
- Synchronous or asynchronous communication:
  - Synchronous (Blocking) : Send and receive must be completed before conducting another process
  - Asynchronous (Non-Blocking) : Computation and communication can be conducted at the same time ( MPI_Wait , MPI_Test  to see if the communication is finished)
- Rank is a unique identifier for each processor.

# 3. Test bed and experimental methods

- Three clusters Emmy, SuperMUC-NG and Hawk
- Process-core affinity was enforced and multiple characteristics are adjusted in order to conduct the study.
- Process scalability: latency-bound communication and compute-bound workload
- One-off idle periods are deliberately generated by massively expanding one computational phase via doing extra work on one random MPI process, usually rank 5 in this study.

# 3. Test bed and experimental methods

- Barrier free bulk synchronous parallel programs
- Distributed-memory parallel system using one MPI process per contention domain (typically ccNUMA)



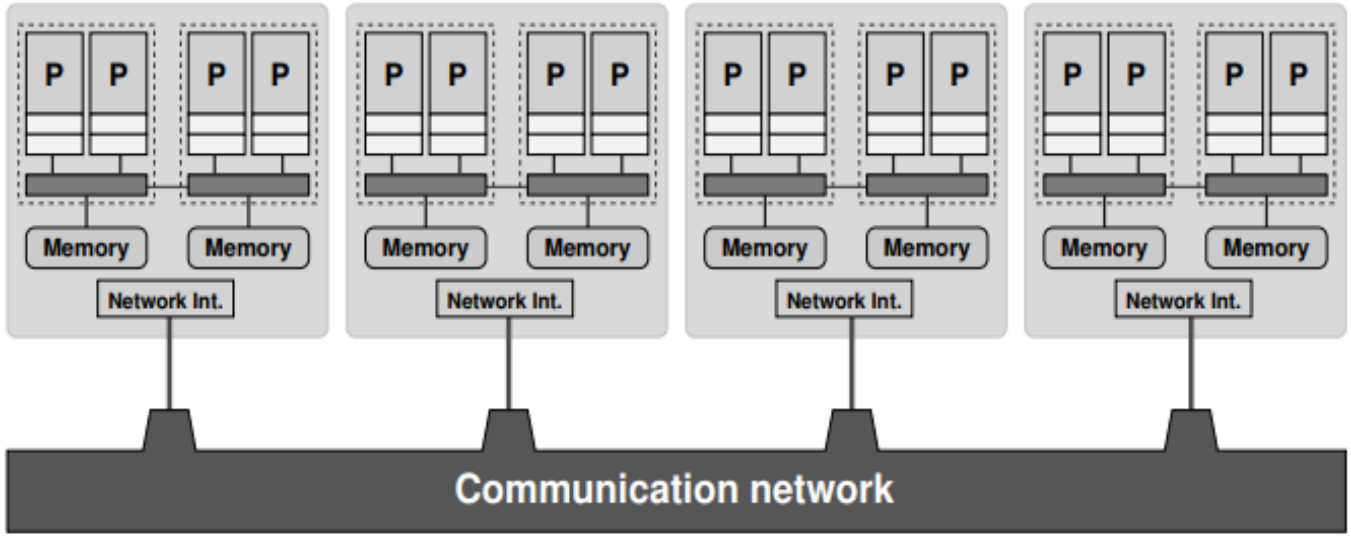




**Figure 1:** Hybrid architecture with shared-memory nodes (typically ccNUMA) (4)

# 4. Related works

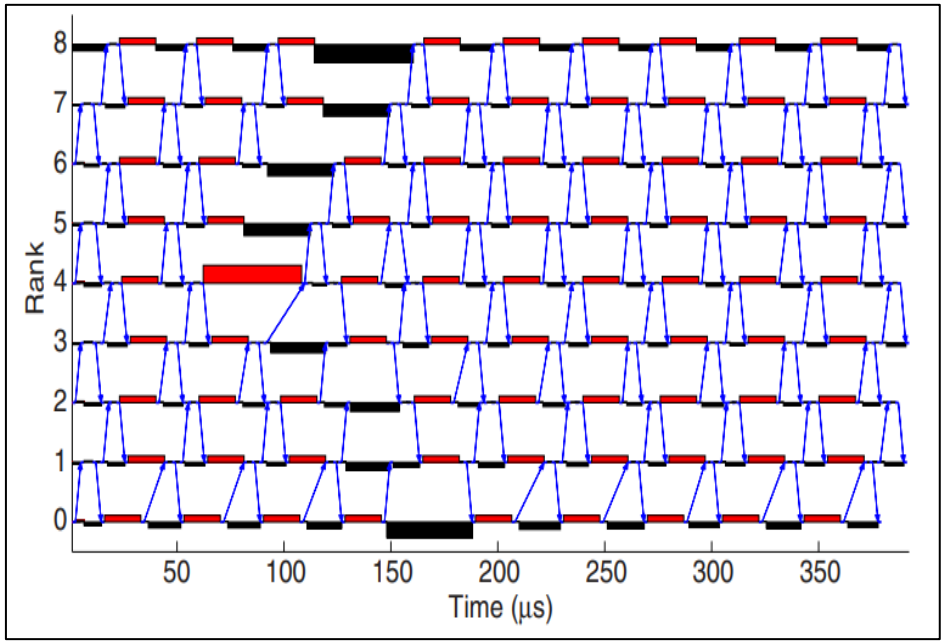



**Figure 2:** Physical mechanism for the generation of idle waves in computation in 9 processes (2)

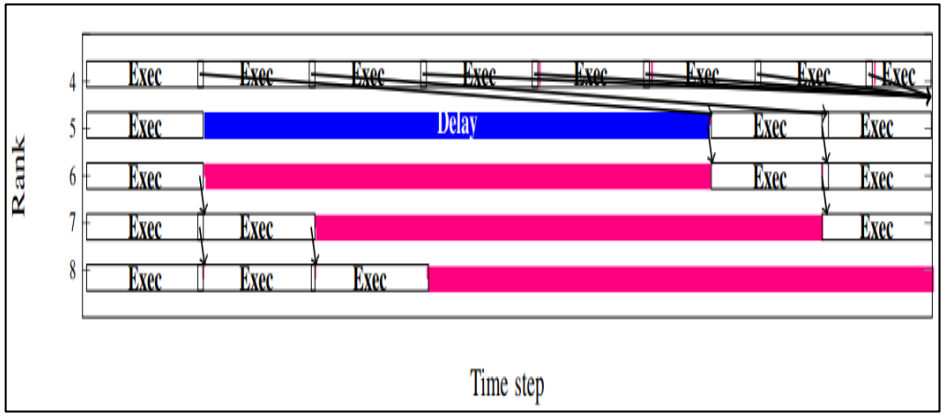



**Figure 3:** The delay propagation mechanism in the most simple setting (3)

# 5. Idle Wave Propagation Velocity for Scalable code

## 5.1. Execution Characteristics

- Traditional memory-bound algorithms such as stencil updates or SpMV with one MPI process per contention domain (ccNUMA node)
- In-core workload

# 5. Idle Wave Propagation Velocity for Scalable code

## 5.2. Categorization of communication characteristics

- Assuming P2P Homogeneous Situation

**Communication Topology:** A consequence of the physical problem underlying the numerical method and of the algorithm(discretization, geometry).

- Compact Topology:
  - Each process communicates with a dense, continuous array of neighbours with distances d (±1, ±2, ±3, …. ±12)
- Noncompact Topology:
  - Each process communicates with processes that are not continuous block d (±1,±12)

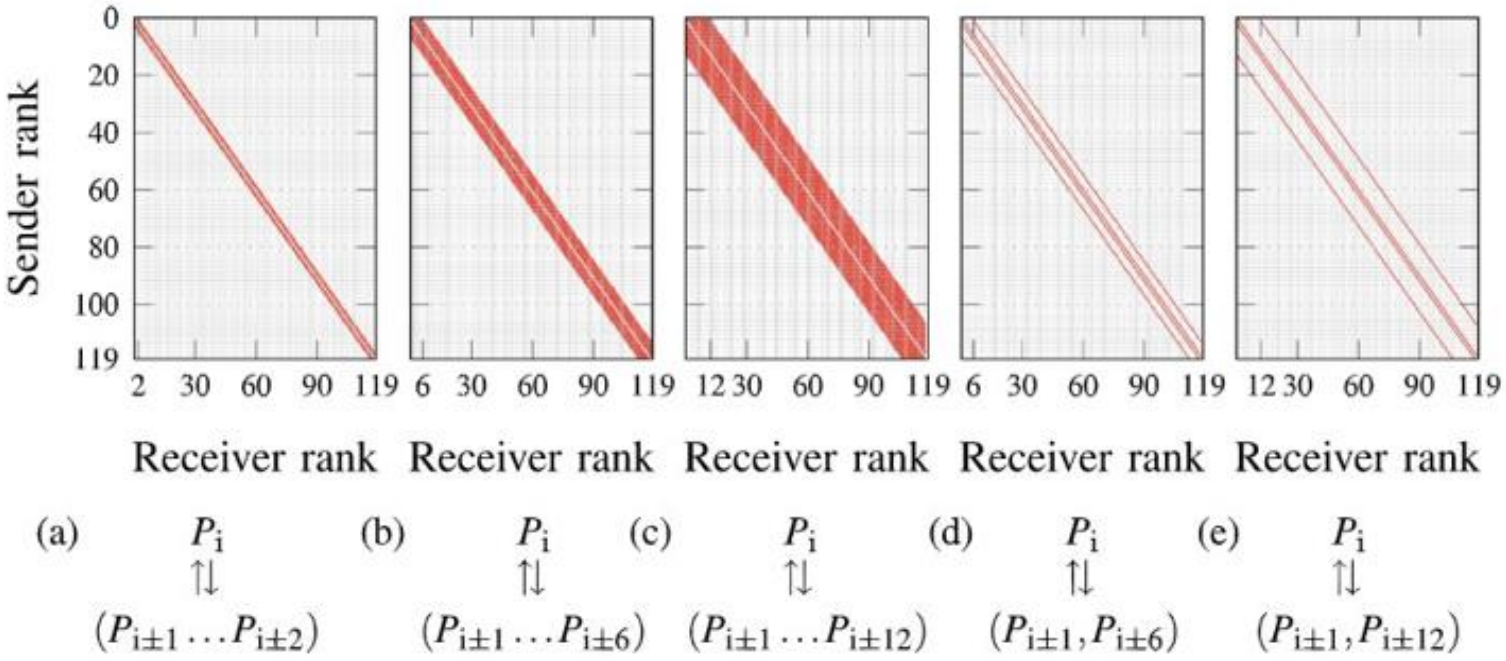



**Figure 4:** Communication topology with bidirectional open chain characteristics (1)

# 5. Idle Wave Propagation Velocity for Scalable code

## Communication concurrency:

- The number of P2P communications are grouped and subject to completion via MPI_Waitall.

Cartesian Communicator example:

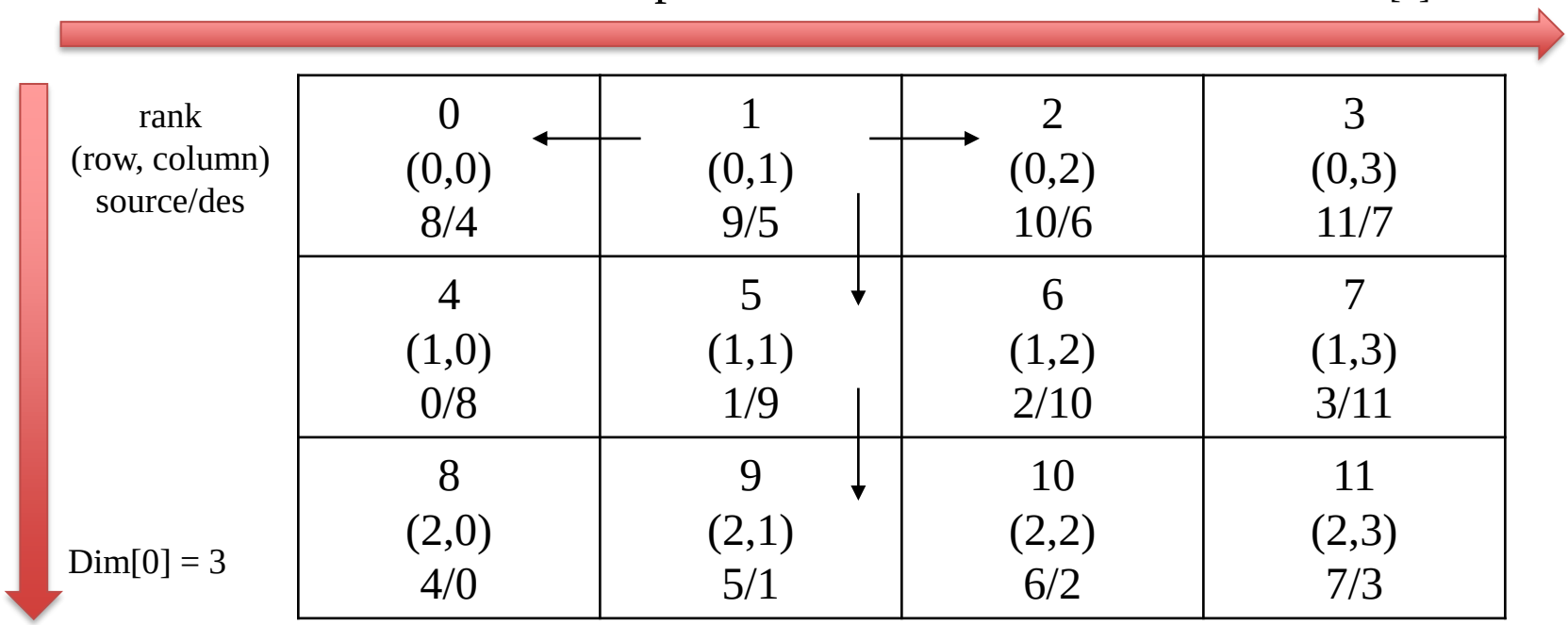


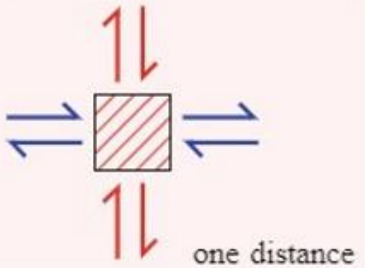

**Figure 5:** Example Cartesian communicator (6)

# 5. Idle Wave Propagation Velocity for Scalable code

Assuming that all P2P communication is nonblocking

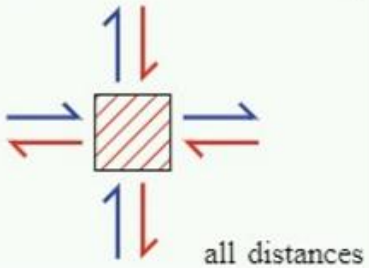
*Multi-wait, single-dimension* (MWSDim)

```
1: while d ≤ dims do
2:     while dir ≤ bi do
3:         MPI_Isend ‡
4:         MPI_Irecv §
5:     end while
6:     MPI_Waitall
7: end while
```



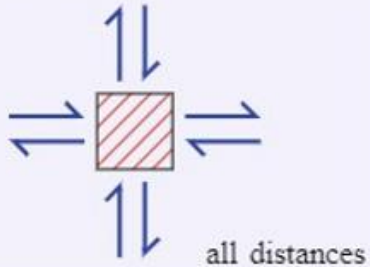

*Multi-wait, multi-dimension* (MWMDim)

```
1: while dir ≤ bi do
2:     while d ≤ dims do
3:         MPI_Isend ‡
4:         MPI_Irecv §
5:     end while
6:     MPI_Waitall
7: end while
```



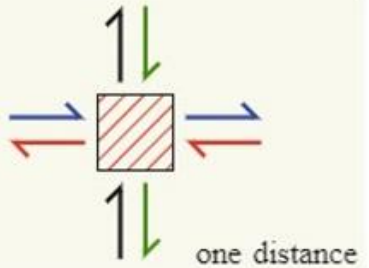

*Single-wait, multi-dimension* (SWMDim)

```
1: while d ≤ dims do
2:     while dir ≤ bi do
3:         MPI_Isend ‡
4:         MPI_Irecv §
5:     end while
6: end while
7: MPI_Waitall
```




*Multi-wait, single-direction* (MWSDir)

```
1: while d ≤ dims do
2:     while dir ≤ bi do
3:         MPI_Isend ‡
4:         MPI_Irecv §
5:         MPI_Waitall
6:     end while
7: end while
```




‡ $P_i$ send to $P_{i+dir \times d}$; § $P_i$ receive from $P_{i-dir \times d}$

**Table 1:** Selected algorithms for communication concurrency in the MPI Microbenchmarks (1)

# 5. Idle Wave Propagation Velocity for Scalable code

## 5.3. Analytical Model of Idle Wave Propagation

- Propagation speed of an idle wave is the speed, in ranks per second, with which it ripples through the system.
- Restriction: Open boundary conditions across the MPI Processes

→ Affects the survival time and not the propagation speed of the wave

# 5. Idle Wave Propagation Velocity for Scalable code

**Corner Cases:**

**Minimum:** Min Speed, Max Survival time

$$v_{\text{silent}}^{\min} = 1 \left[\frac{\text{ranks}}{\text{iter}}\right] \times \frac{1}{T_{\text{exec}} + T_{\text{comm}}} \left[\frac{\text{iter}}{\text{s}}\right].$$

- Simple direct next-neighbour communication ( d = 1 )
- $T_{exec}$ and $T_{comm}$: Execution and communication times of one iteration of the bulk-synchronous program

→ The wave survives until system boundaries.

# 5. Idle Wave Propagation Velocity for Scalable code

**Maximum:** Max Speed, The wave dying out quickly in a minimum of one time step.

$$v_{\text{silent}}^{\max} = \alpha \left[\frac{\text{ranks}}{\text{iter}}\right] \times \frac{1}{T_{\text{exec}} + T_{\text{comm}}} \left[\frac{\text{iter}}{\text{s}}\right],$$

$$\alpha = \max\left(\text{size}_{\text{comm}} - r_{\text{inject}} - 1, r_{\text{inject}} - 1\right).$$

$r_{inject}$ : The rank where the idle wave originated

$size_{comm}$: The total number of MPI processes

# 5. Idle Wave Propagation Velocity for Scalable code

## Multi-neighbour Communication:

$$v_{\text{silent}} = \kappa \cdot v_{\text{silent}}^{\min} \left[\frac{\text{ranks}}{\text{s}}\right]$$

*k* is the distance (in processes) travelled by the wave in one time step (depending on communication topology and concurrency).

*j* is the longest-distance communication partner of a process.

$$\kappa = \begin{cases} \sum_{k=1}^{j} k = \frac{j(j+1)}{2} & \text{if compact MWSDim/MWSDir/blocking} \\ \sum_{k=1,j} k = j+1 & \text{if non-compact MWSDim/MWSDir/blocking} \\ j & \text{if MWMDim/SWMDim} \end{cases}.$$

# 5. Idle Wave Propagation Velocity for Scalable code

## 5.4. Experimental validation

**Microbenchmarks:**

- One-off idle wave originated at rank 5, dark blue
- Execution time = 13ms ( light blue) and a data volume of 1 KiB per message.

# 5. Idle Wave Propagation Velocity for Scalable code

## Compact Communication:

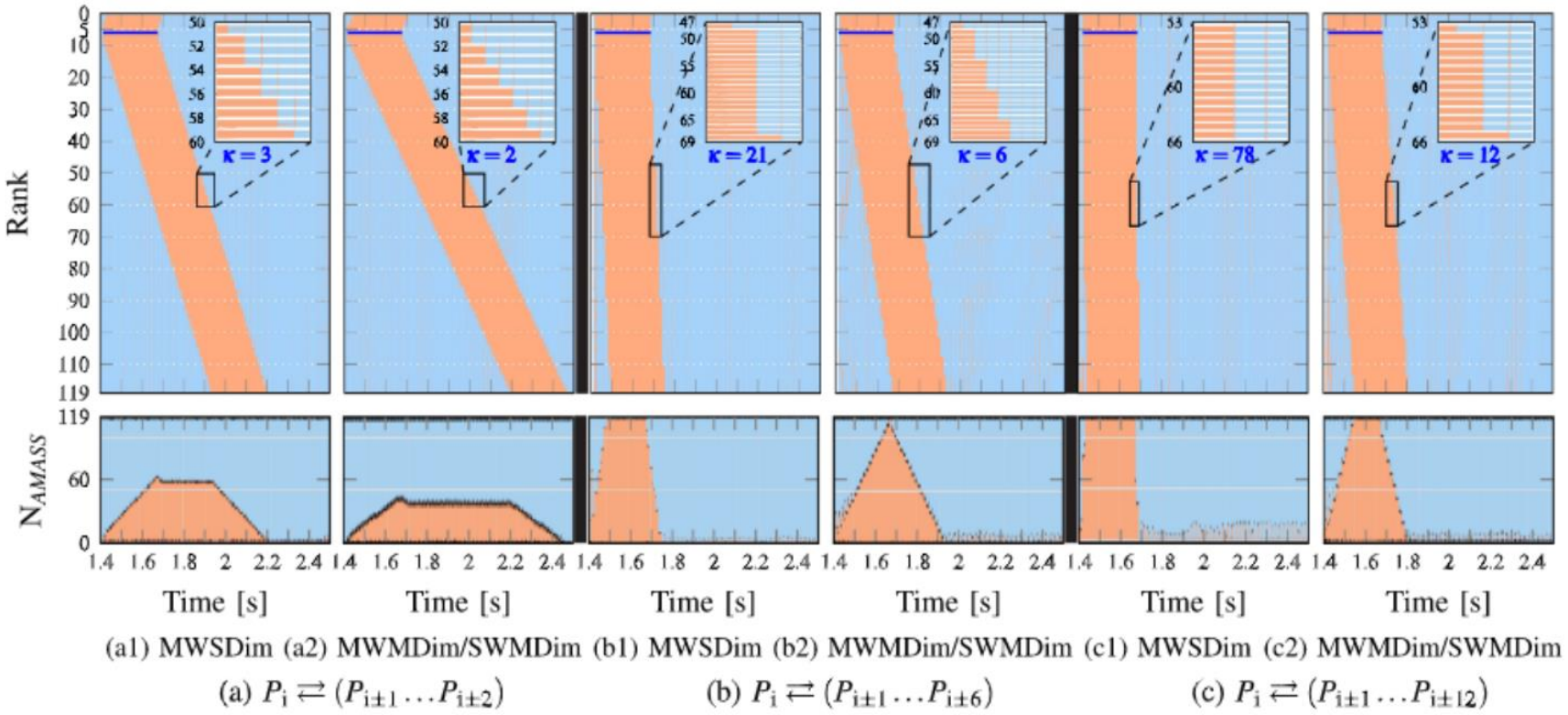



**Figure 6:** Validation for compact communication (1)

# 5. Idle Wave Propagation Velocity for Scalable code

**Compact Communication:**

- Propagation speed of idle waves is independent of the number of split-waits
- Higher speed:
  - Communication distance goes up with increasing number of communications of communication partners
  - Number of dimensions spanned within each MPI_Waitall
- Higher $k$ → higher speed → shorter survival time.
- Slower wave propagation → more waiting time

  → Resource utilization across processes.

# 5. Idle Wave Propagation Velocity for Scalable code

## Noncompact Communication

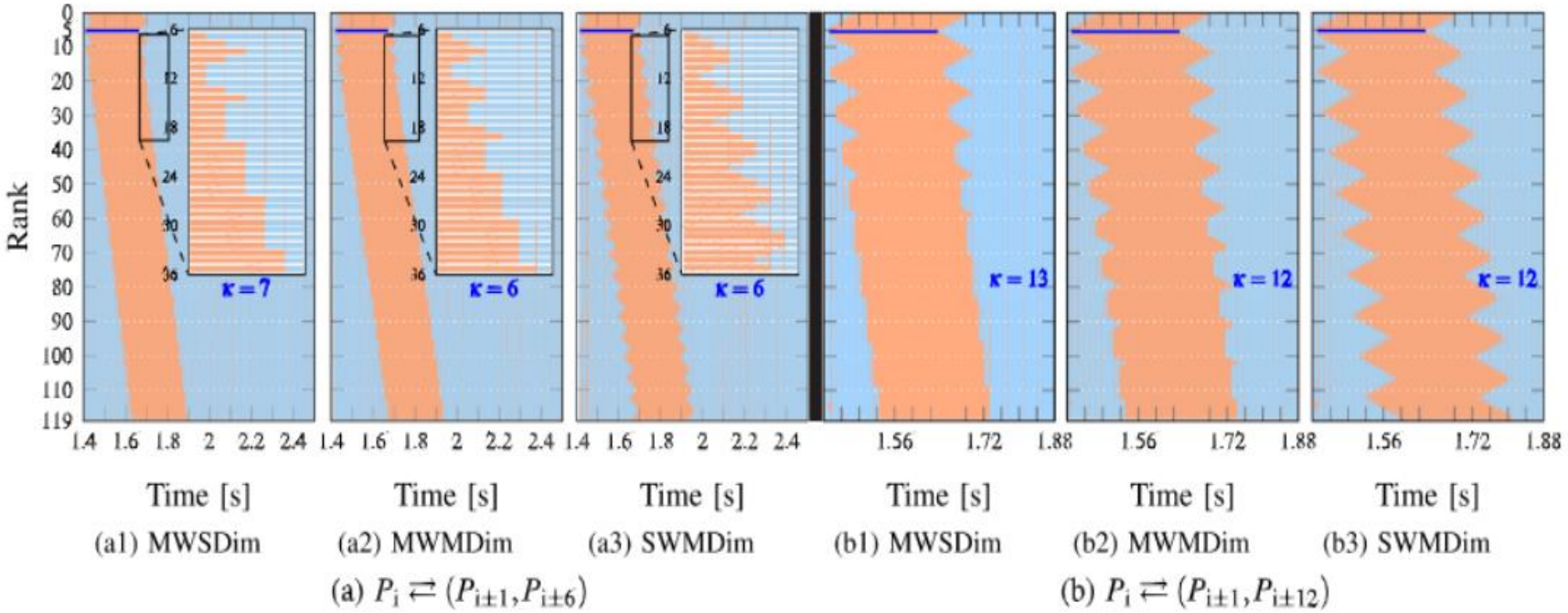



**Figure 7:** Validation for noncompact communication (1)

# 5. Idle Wave Propagation Velocity for Scalable code

## Noncompact Communication

- Difference between propagation speed of the “fast waves” and that of the “secondary waves” → Only fast waves remain.
- Higher speed of residual waves:
  - A larger number of split-waits
  - A smaller number of communication dimensions spanned by each MPI_Waitall
  - A larger longest communication distance j
- The zigzag pattern dies out for MWSDim and MWMDim but remains for SWMDim

# 5. Idle Wave Propagation Velocity for Scalable code

## Heterogeneous Communication

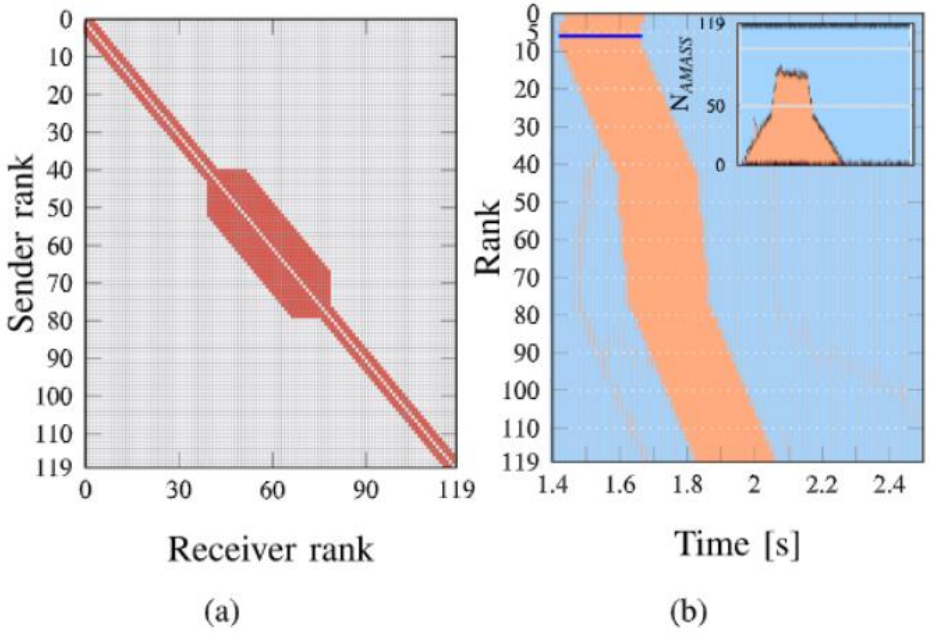



**Figure 8:** Idle wave propagation with heterogeneous compact communication characteristics (a) Topology matrix; (b) Idle wave propagation for SWMDim concurrency. (1)

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

# 6. Idle Waves Interacting with MPI Collectives

- MPI supports Point-to-Point communication and collective functions for communication among multiple computer nodes
- Not all MPI collective routines eliminate a traveling idle wave, some may be permeable to it, depending on the implementation.
- Restriction: Intel MPI on Emmy

# 6. Idle Waves Interacting with MPI Collectives

- Globally synchronizing Primitives

→ Destroy idle waves completely

- Globally Non-synchronizing Primitives
  - MPI_Reduce
  - MPI_Gather
- Implementation Variants
  - I_MPI_ADJUST_<opname>

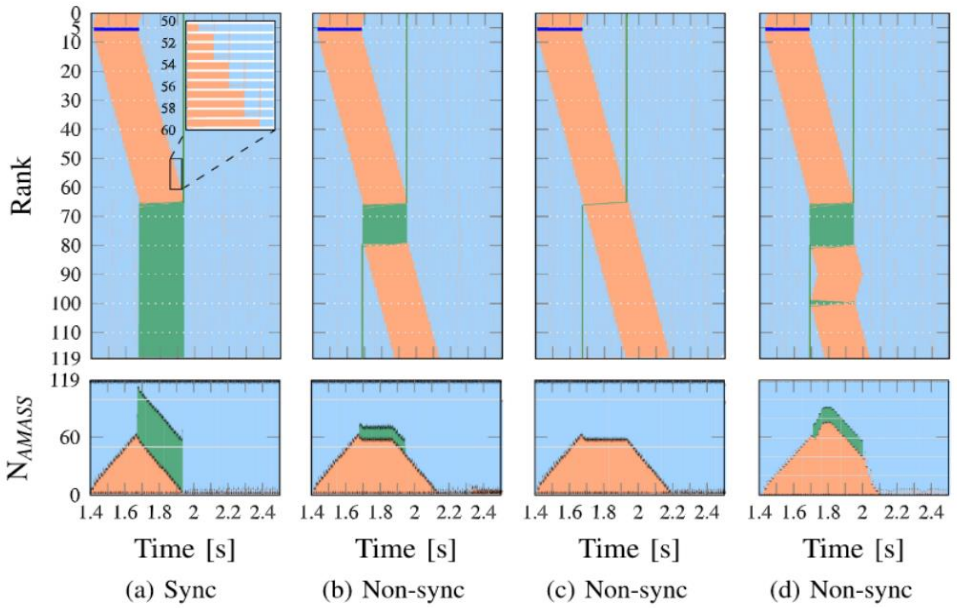



**Figure 9:** Interaction of idle waves with MPI Collectives (1)

# 7. Idle Wave Decay

Idle wave decay is the phenomenon where idle waves are damped in time.

## 7.1. Topological Decay

- Three benchmarks Hawk, Emmy, SuperMUC-NG have different features in respect to system topology.
- Communication heterogeneities→ Fine-grained noise→ Idle wave decay.
- Decay rate: Emmy has the strongest effect then Hawk and lastly SuperMUC-NG

# 7. Idle Wave Decay

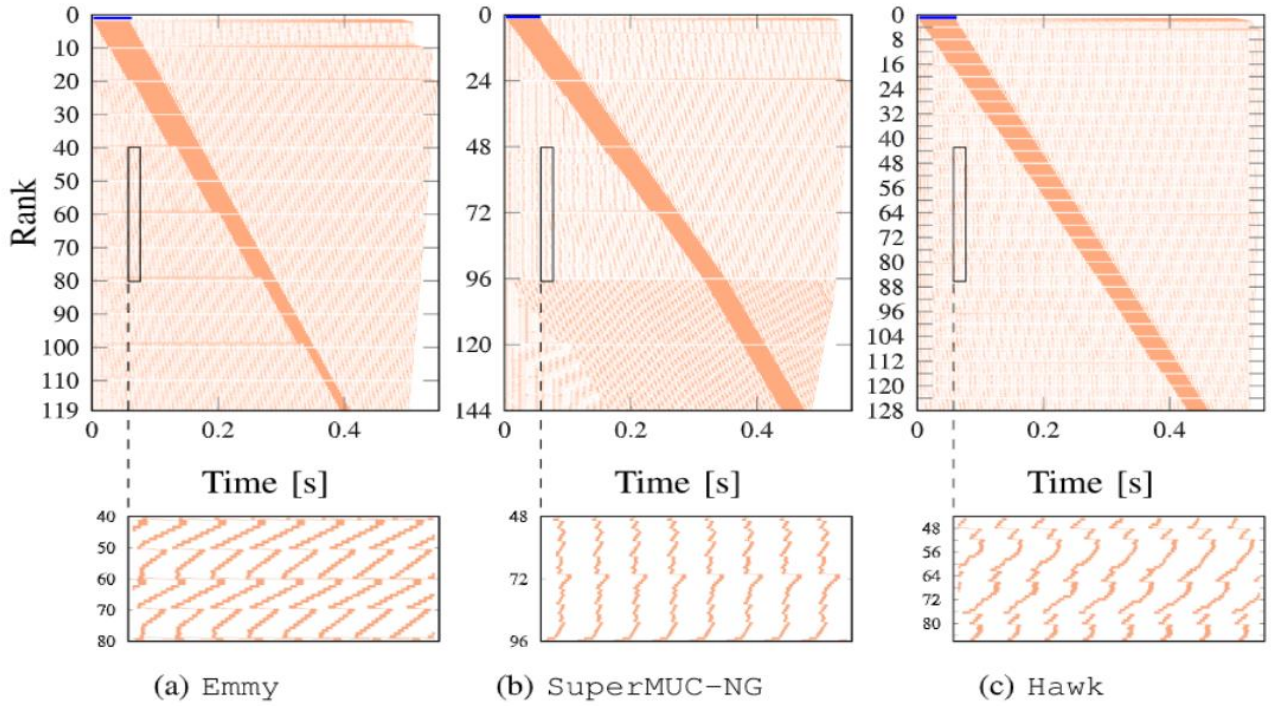



**Figure 10:** Topological idle wave decay on the benchmark systems. (1)

# 7. Idle Wave Decay

## 7.2. Noise-Induced Decay

- Fine grained noise effects on idle wave decay with resource-scalable code
- Noise eliminates the trailing edge of the wave
- A small idle period of duration $T_{noise}$ shortens the next by the exact amount of $T_{noise}$

→Cumulative process

- Only noise power and not the noise characteristics that has an impact on noise-induced decay rate.

# 7. Idle Wave Decay

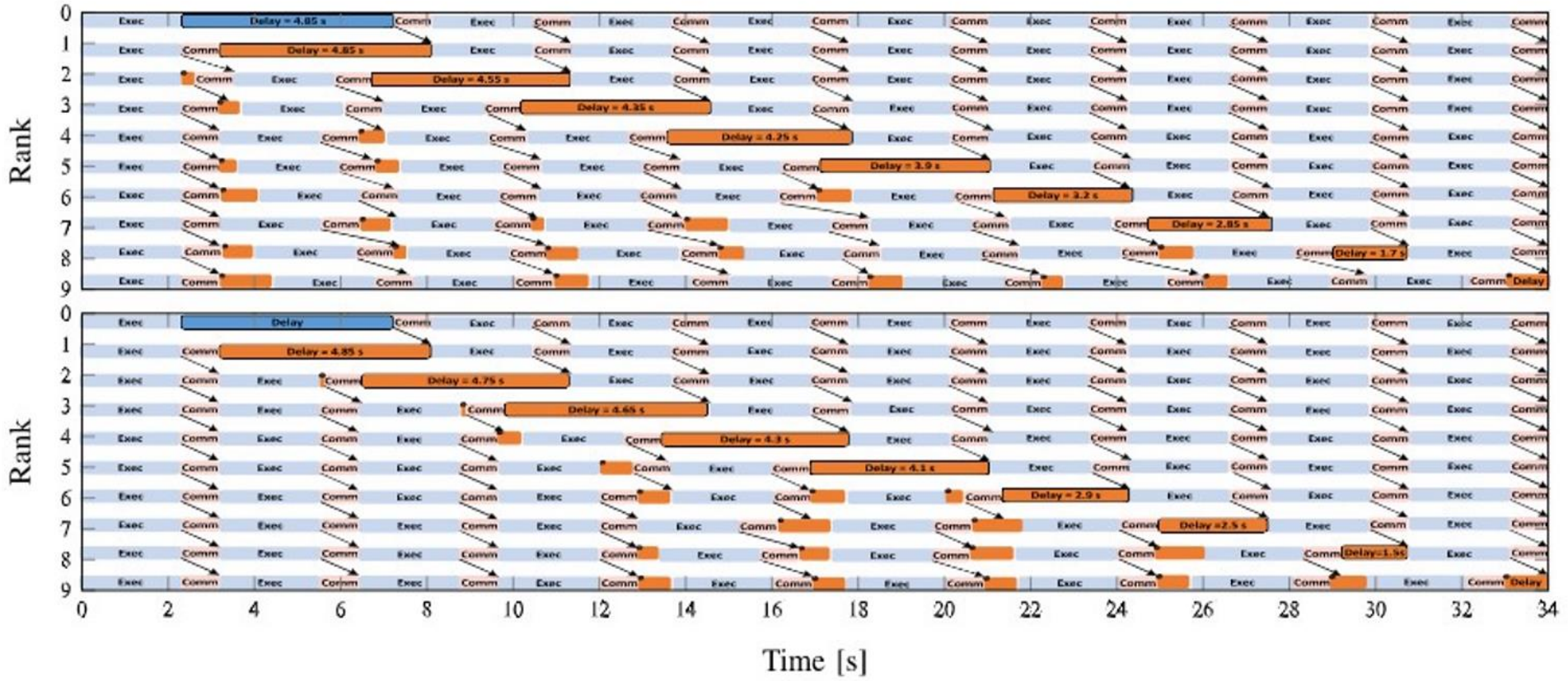



**Figure 11:** Experiment comparing the average decay rate of an idle wave for 2 different noise characteristics (1)

# 7. Idle Wave Decay

## 7.3. Experimental validation

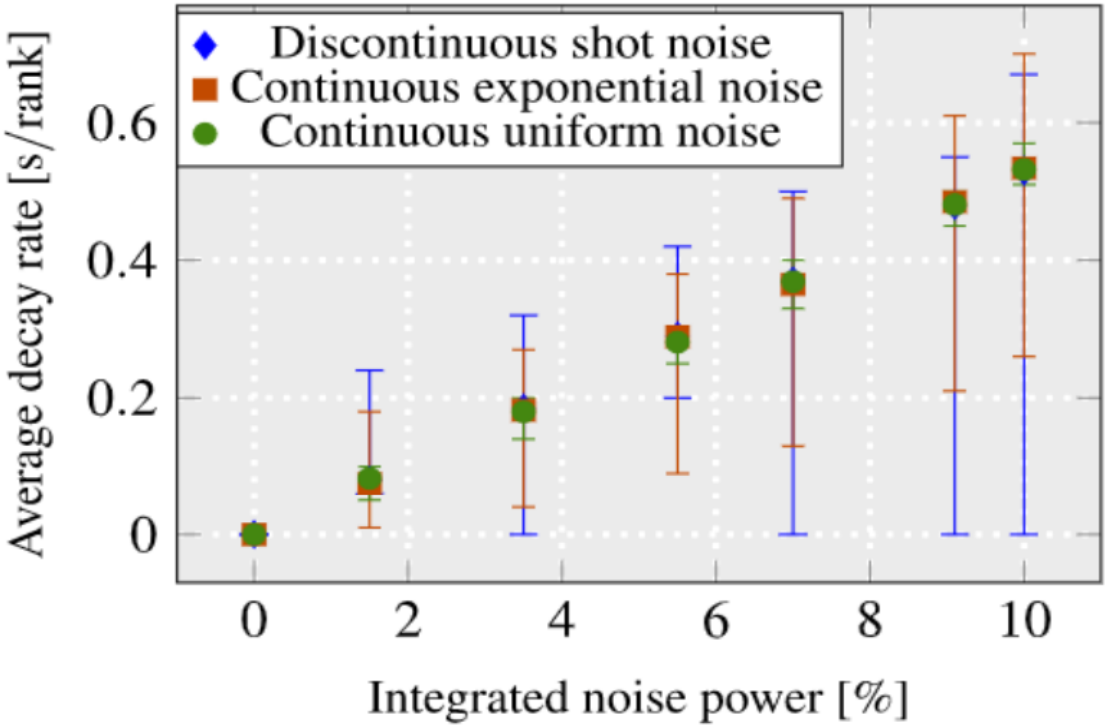



**Figure 12:** Decay rate of an idle period in s/rank, comparing 3 different noise patterns (1)

# 8. Summary

- Analytic model of idle wave propagation speed based on communication topology and concurrency of resource-scalable MPI.
- MPI collectives can be permeable to idle waves depending on which collectives we use and how we implement or adjust them.
- System Topology → Fine-grained noise → Impact on idle wave decay rate.
- Only noise power and not noise characteristics has an impact on noise-induced decay rate.

# 9. References

(1) Afzal, A., et al: Analytic Modeling of Idle Waves in Parallel Programs: Communication, Cluster Topology, and Noise Impact (2021)

(2) Markidis, S., et al: Idle Waves in high-performance computing. (2015)

(3) Afzal, A., et al: Propagation and decay of injected one-off delays on cluster: a case study. (2019)

(4) Hager, G., Wellein, G. : Introduction to High Performance Computing for Scientists and Engineers. (2010)

(5) MPI: A Message-Passing Interface Standard Version 3.1 (2015)

# 9. References

(6) Prof.Dr-Ing Riedel, Morris: 2021 High Performance Computing Lecture 2 Parallel Programming with MPI Part1. URL : 2021 High Performance Computing Lecture 2 Parallel Programming with MPI Part1 – YouTube

(7) Neso Academy: Interprocess Communication. URL : Interprocess Communication – YouTube

(8) NHR @FAU : Parallel Programming 2020: Lecture 7 - ccNUMA and wavefront parallelization with OpenMP. URL : Parallel Programming 2020: Lecture 7 - ccNUMA and wavefront parallelization with OpenMP - YouTube

# 9. References

(9) Prof.Dr-Ing Riedel, Morris: 2021 High Performance Computing Lecture 4 Advanced MPI Techniques Part1. URL: 2021 High Performance Computing Lecture 4 Advanced MPI Techniques Part1 💻 – YouTube

(10) Graphcore: Fundamentals of Bulk Synchronous Parallel Execution on the IPU

URL: Fundamentals of Bulk Synchronous Parallel Execution on the IPU - YouTube

# Thank you very much for listening !