

# High Performance Parallel Input/Output for Cluster Environments



**Thomas Ludwig**

Ruprecht-Karls-Universität Heidelberg

Computer Science Department

Parallel and Distributed Systems

**[t.ludwig@computer.org](mailto:t.ludwig@computer.org)**



# Storage in 2004

---

- Supercomputers in the TOP500-list
  - Current No. 1: NEC's Earth Simulator  
Compute power: 35 TFlop/s  
**10** TByte main memory; **700** TByte disk space
- Berkeley Report „How Much Information“
  - Increase of **5** Exabytes (about  $5 \times 10^{18}$  bytes) of information stored in 2002
  - 92% of it on magnetic media, mainly hard disks



# Single Disk Performance

---

## Storage performance

- A single disk makes about 50MByte/s
- To store 1 GByte takes 20 seconds
- To store 1 TByte takes 20.000 seconds (~5 hours)  
Or 20 seconds with 1000 disks

Storage performance is a crucial factor



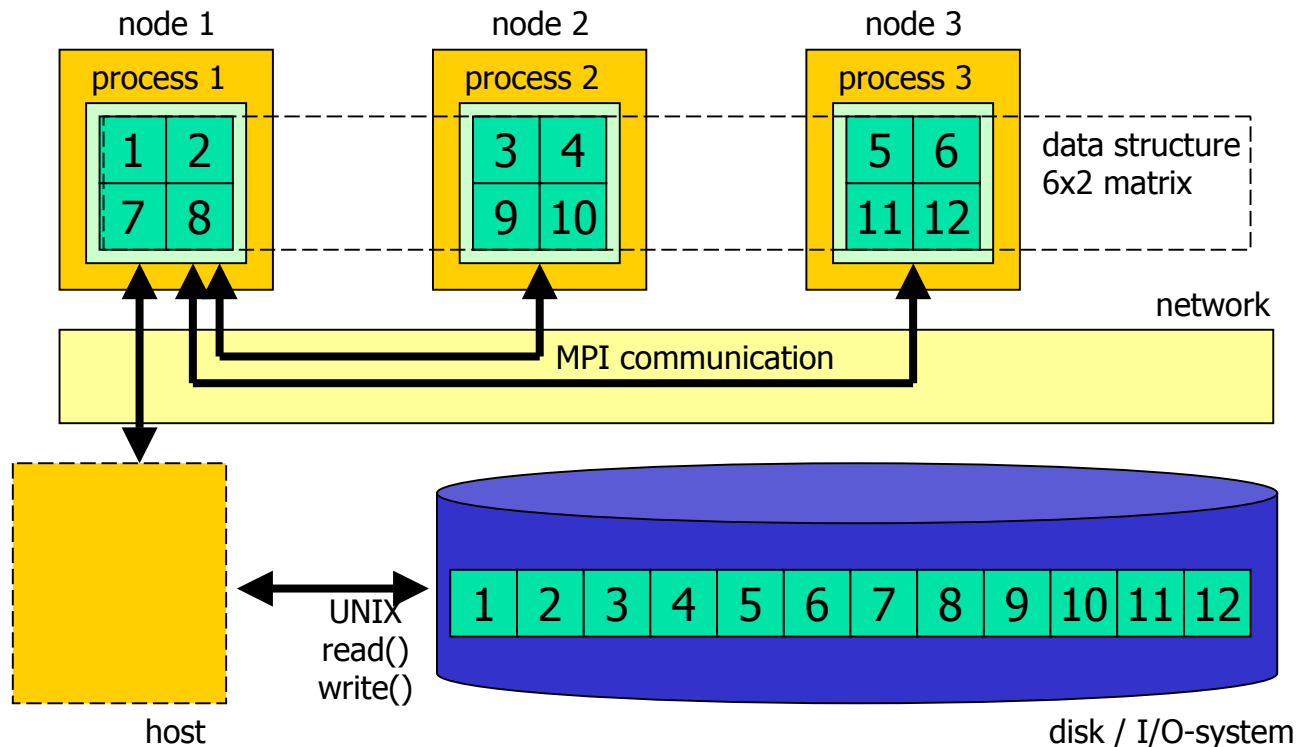
# Abstraction Layers

---

- Program level
  - How do I access my data?
  - Single process / multiple processes
- System level
  - Where is my data?
  - Single disk / multiple disks
  - Powerful modern I/O-systems are available
    - RAID, SAN, NAS
- Concepts at program and at system level are independent of each other!

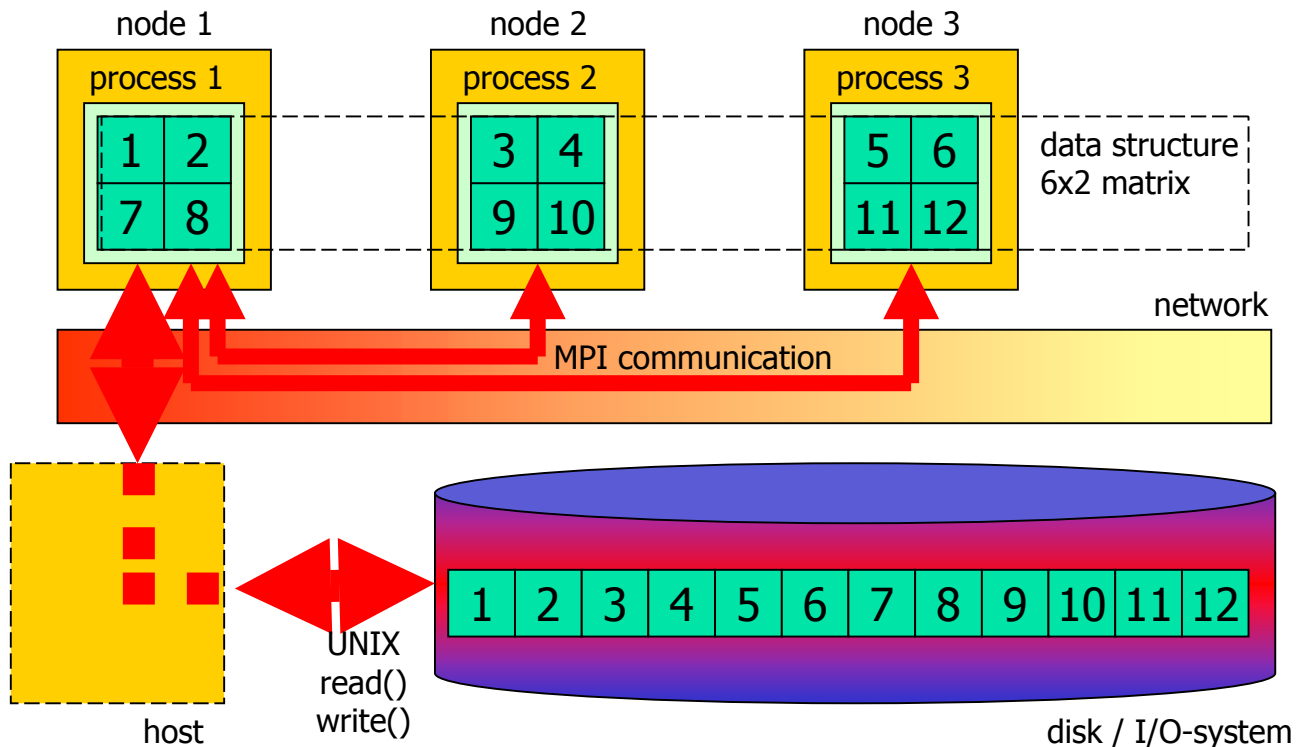
# Traditional Approach

## Parallel program with MPI communication



# Traditional Approach: Problems

## Parallel program with MPI communication





# Modern I/O-Concepts

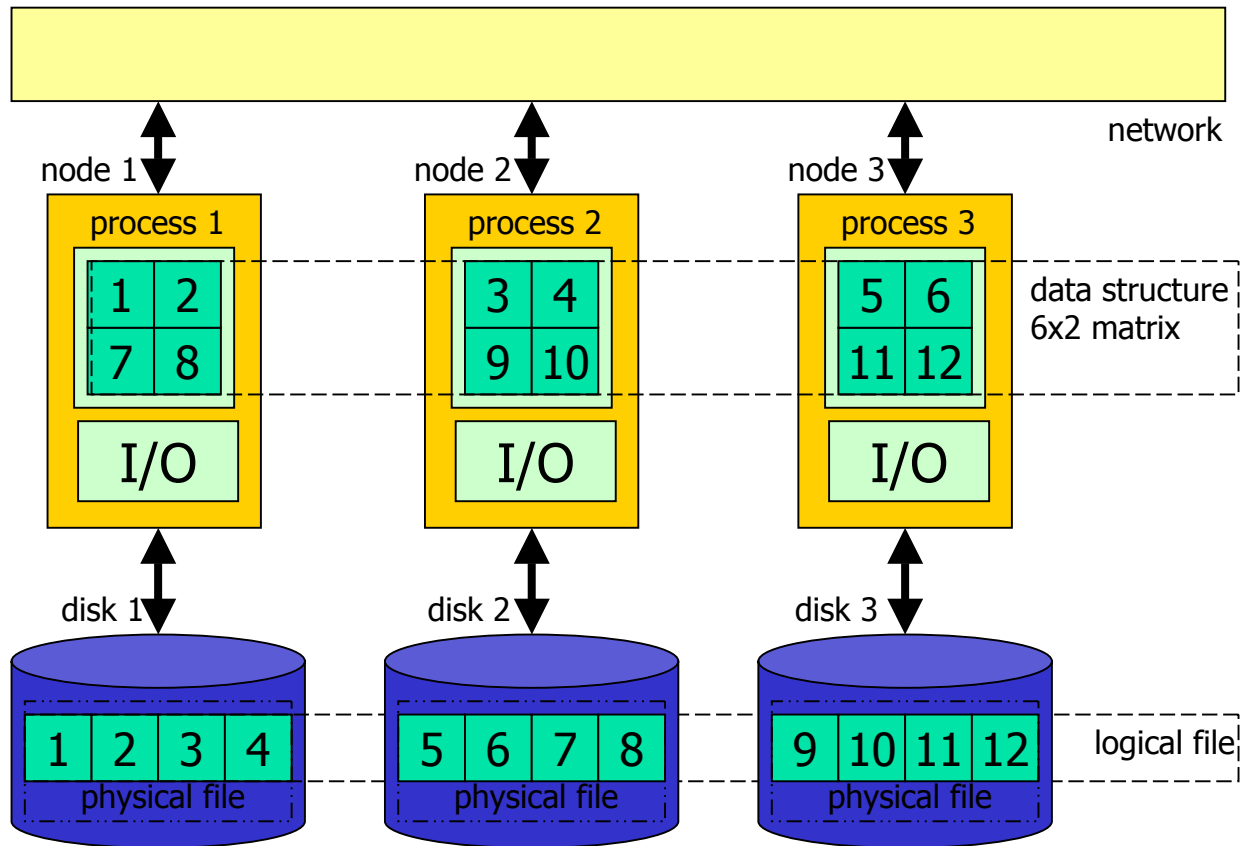
---

## Parallelization of I/O

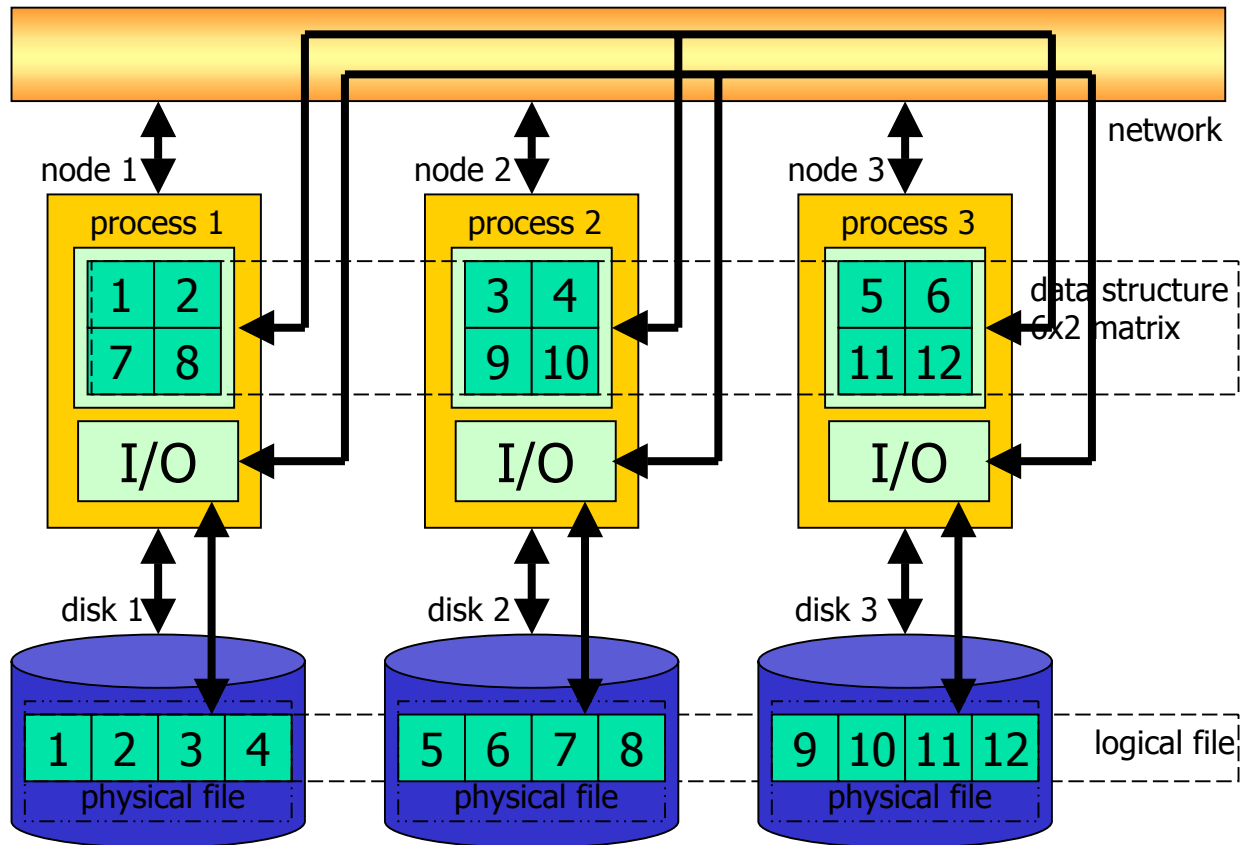
- At program level (Parallel file I/O)
  - Each process can directly make I/O calls
  - I/O calls from different processes may address identical files (even identical bytes)
- At system level (Parallel file system)
  - We use more than one disk
  - A single file may be spread over many disks
  - The size of a file may exceed the size of a disk

Both together is called "Parallel I/O"

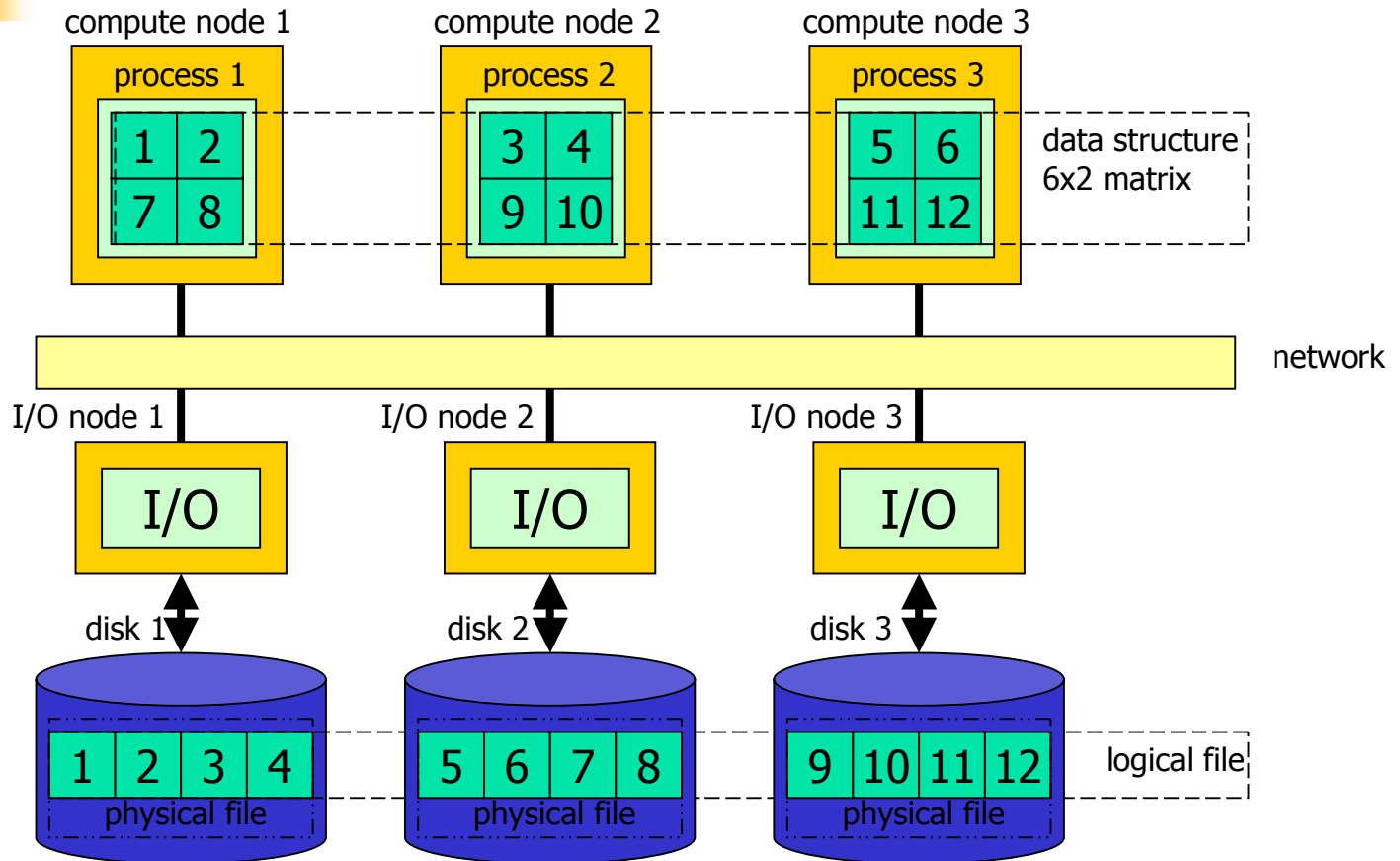
# The Concept of Parallel I/O



# The Concept of Parallel I/O...



# The Concept of Parallel I/O...





# Parallel I/O Libraries

---

- Parallel programming with message passing on large supercomputers/clusters
  - Use send()- and receive()-routines
  - MPI is the current standard for this
- Parallel I/O: defined by MPI-IO
  - Defines read()-operation similar to receive()
  - Defines write()-operation similar to send()
  - Keeps most other semantic details of MPI message passing



# Parallel File Systems

---

- Parallel file system
  - Distributes a single file over several disks
- Not too many systems available
- A selection
  - GPFS (IBM): older proprietary approach
  - PVFS: current popular open source system
  - Lustre: future open source system



# Open Research Questions

---

- Usability
  - Abstraction levels with user libraries
- Increase of performance
  - Metadata server is bottleneck
  - **Mapping of logical file to physical bytes is crucial**
- Improved availability
  - How to keep a file on 1000 disks?
  - Fault tolerance is mandatory

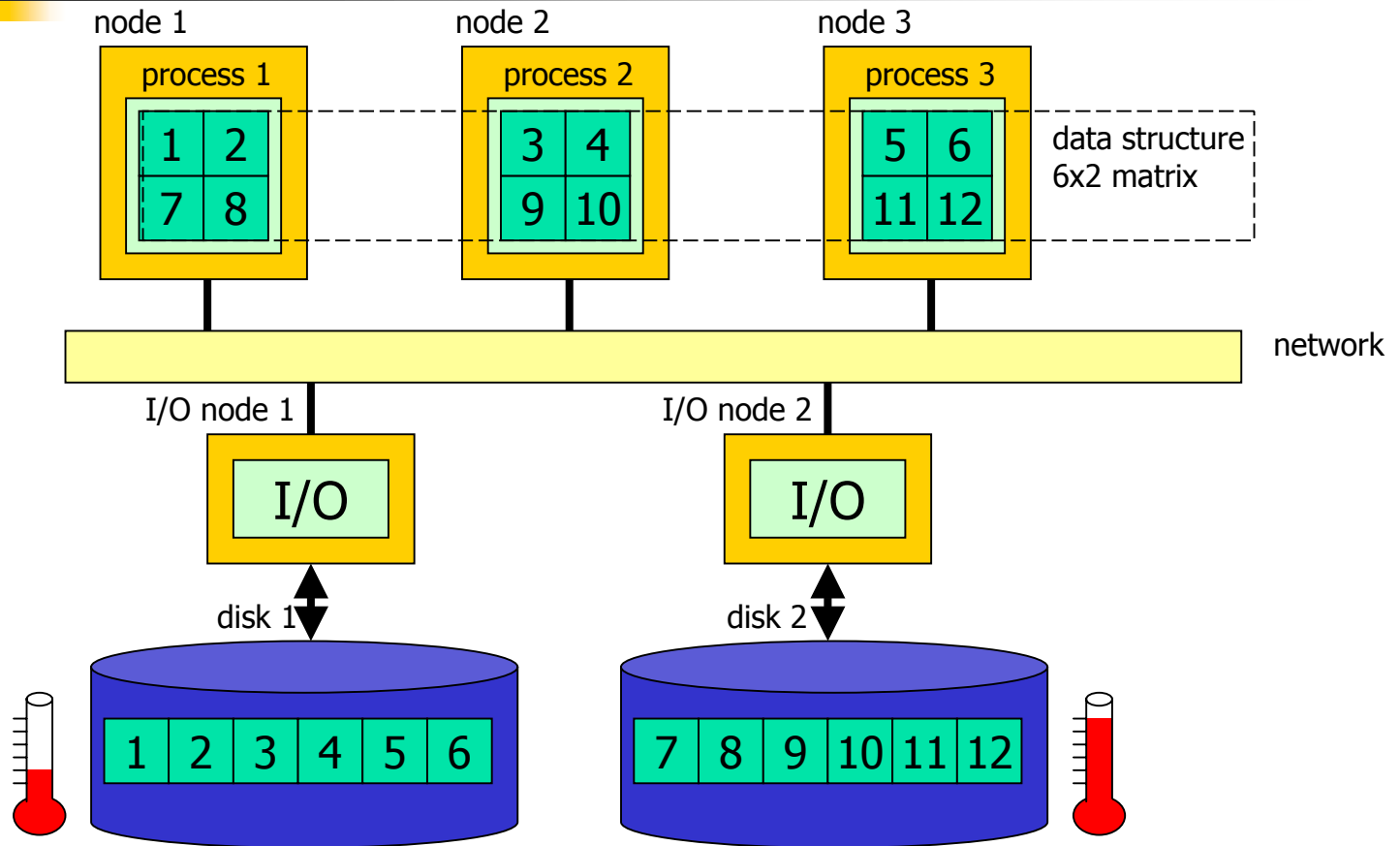


# Own Research

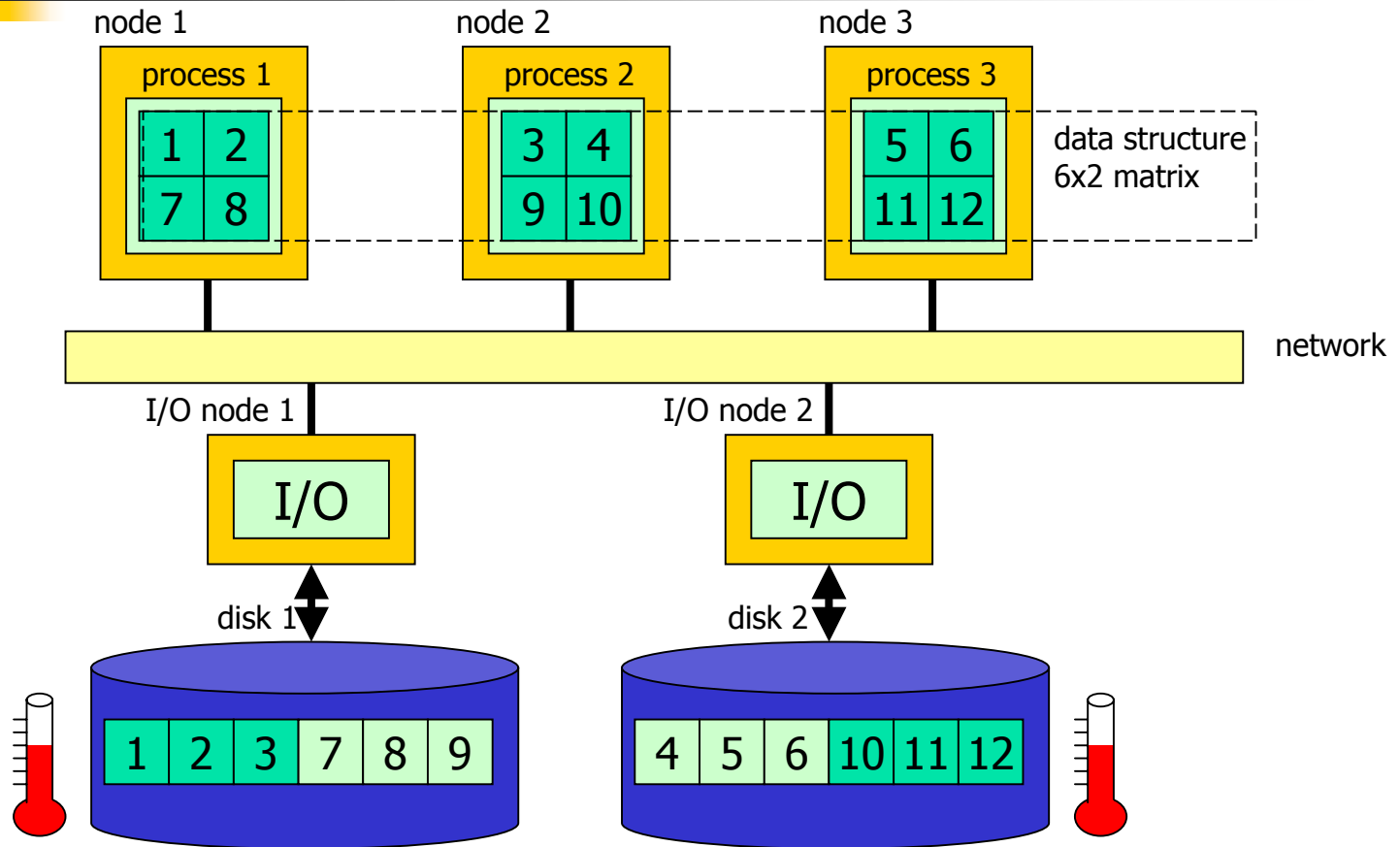
---

- Deployment and optimization of parallel I/O in cluster environments
  - Performance measurement
  - **Load balancing**
  - Deployment in production environments

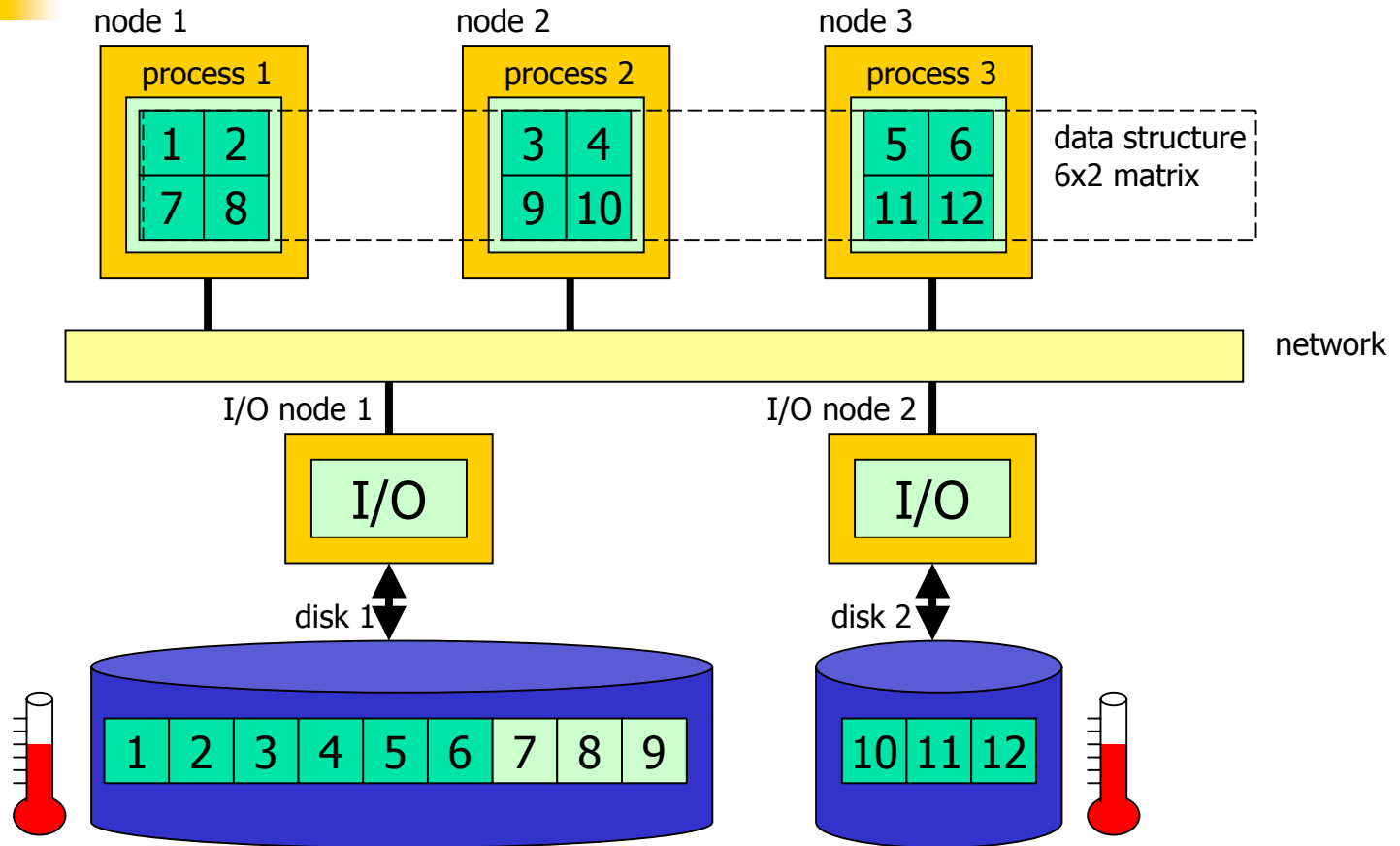
# Load Balancing



# Load Balancing by Redistribution



# Load Balancing by Migration





# The Future

---

- Integration of own mechanisms into parallel file systems
  - Load balancing
  - Load measurement
- Cooperation with other interested researchers and users