On Program Restructuring for Cluster-based Highly Parallel Embedded Architectures

CISTER - Research Center in
Real-Time & Embedded Computing Systems

Ismael Marín Carrión – Luís Lino Ferreira – Luís Miguel Pinho
{ismac,llf,lmp}@isep.ipp.pt

Motivation

- **Embedded systems** more and more require to process large amounts of data coming from multiple sensors
- **Many-core processors** are starting to appear in the embedded domain, e.g. the Kalray Multi-Purpose Processor Array (**MPPA**) many-core
- The transformation of current single-core applications to parallel multi-threaded applications is a challenge
- This work evaluates strategies to **parallelize real-time applications** into many-cores
- **Goal**: To take a embedded application and parallelize it into the MPPA many-core to compare and evaluate different approaches for parallelization

Application Use Case

- A parallel and distributed **traffic simulator**: It computes the vehicle movements across lanes during T time steps
- 3 application levels: **External Linux-based**, **I/O** and **Compute Cluster applications**
- **Master/Slave** scheme: The master process running on the I/O spawns slave processes on the compute clusters that run the simulation
- **I/O application** starts-up the compute clusters, sends a dataset to work on and gathers the results
- **Cluster application** runs the simulation with the received data and sends back the results to the I/O

Algorithm 1 IO-level application

```
receive the input data file from Linux-based application
open and prepare the communication and synchronization connectors
starts up the compute clusters
wait until the clusters are ready (global sync)
send data to compute clusters (portal transfers)
wait until the clusters open the channels (global sync)
unblock each single cluster (individual sync)
wait for the results
show results
```

Algorithm 2 Cluster-level application

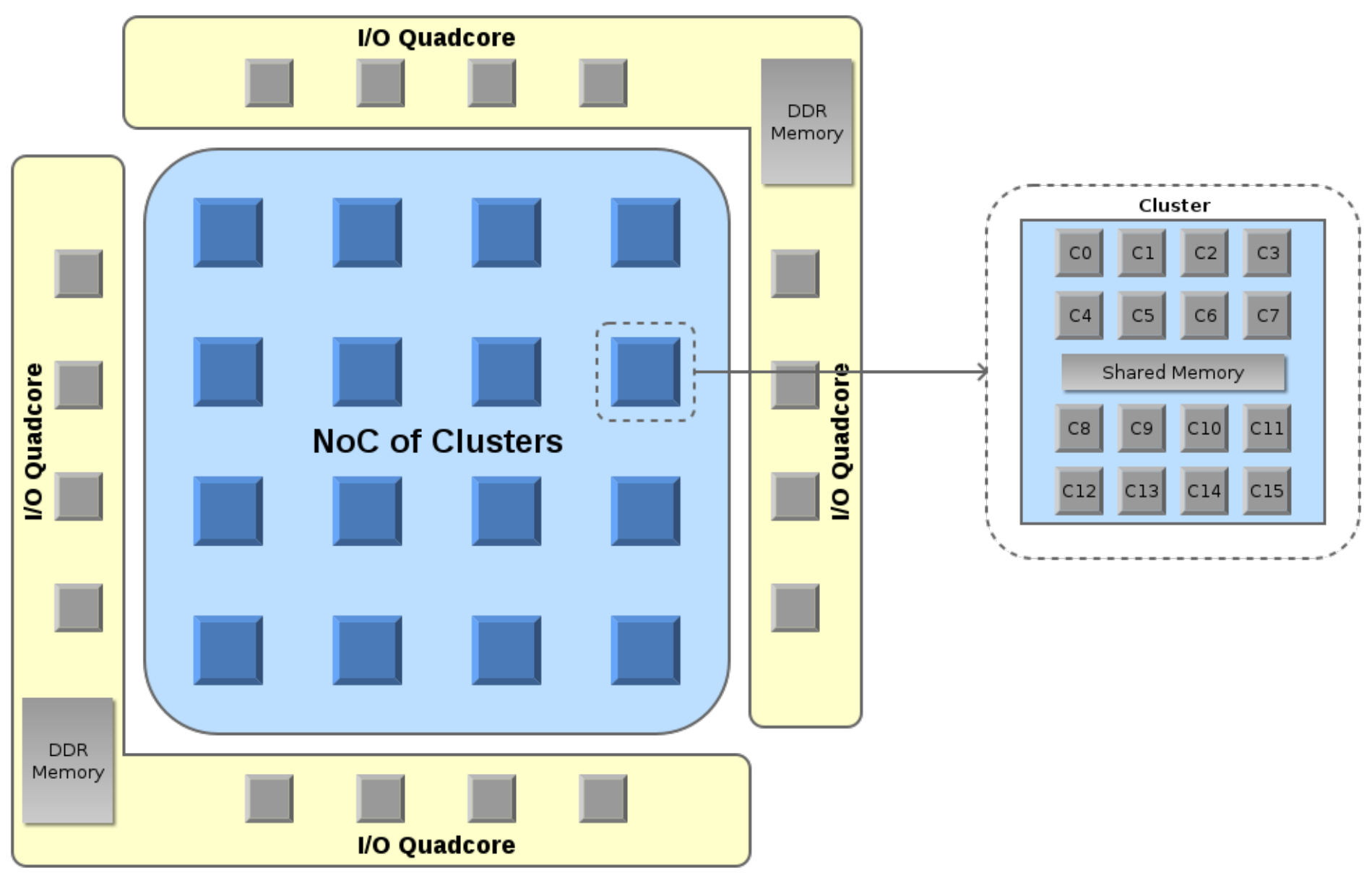
```
open and prepare the communication and synchronization connectors
contribute to unblock the master process (global sync)
wait for data
open the R/W channels
notify to the I/O that it has already opened its channels (global sync)
wait for the others (individual sync)
for $i = 0 \rightarrow SIMULATION\_TIME$ do
    run one-step simulation
    exchange data on frontiers with other clusters (channels)
end for
send partial results (portal transfer)
```

- They use **portals** (multipoint transfers), **channels** (point-to-point) and **sync** (synchronization) connectors for communication and synchronization
- **OpenMP** is used to exploit the thread-level parallelism

ARTEMIS/0002/2010 JU grant nr. 269354 (ENCOURAGE) and FCOMP-01-0124-FEDER-020447 (REGAIN). Co-financed by:

MPPA Architecture

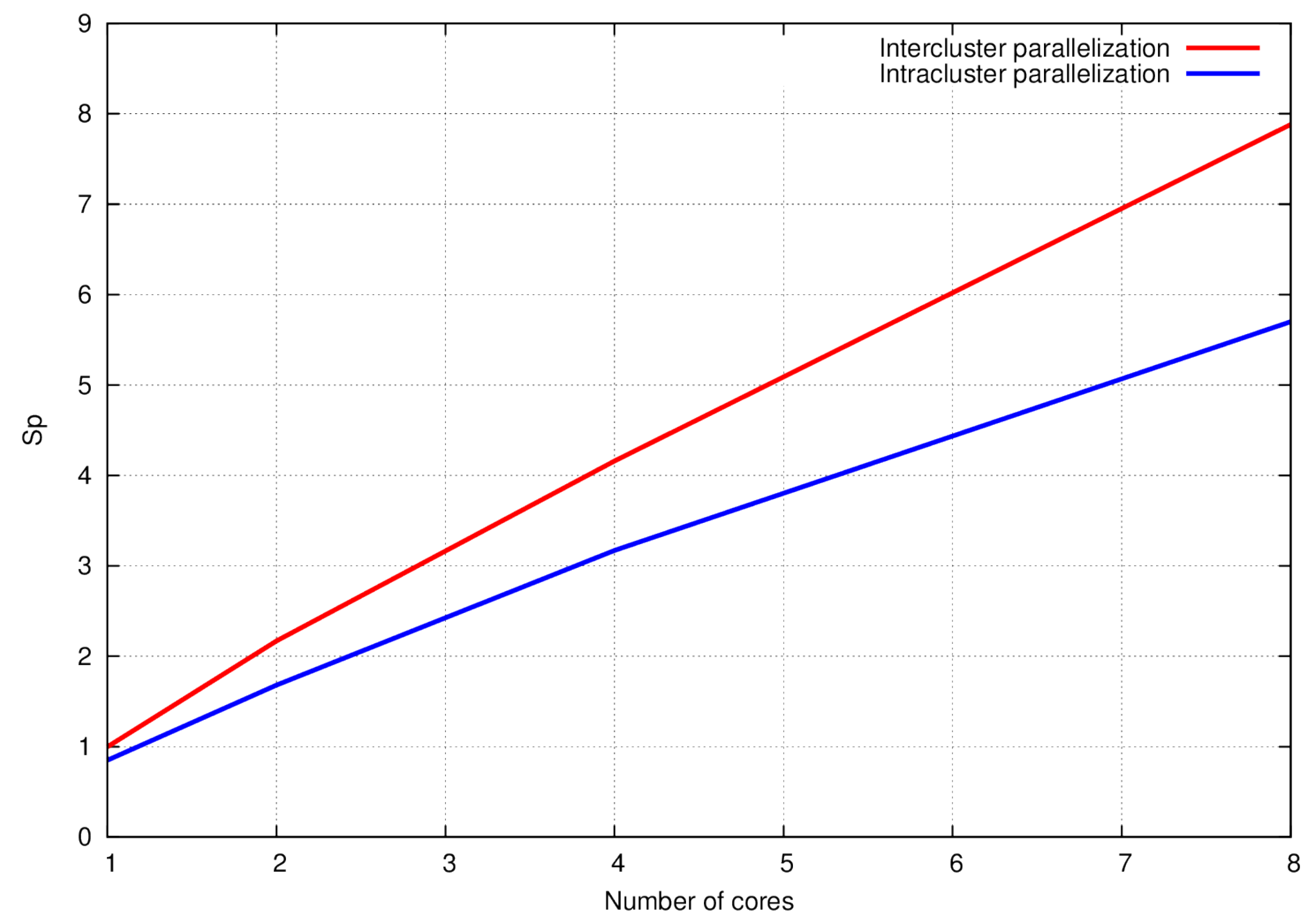
- 16 **Compute Clusters** connected through a high-throughput Network-on-Chip with 16 cores each one
- 4 **I/O subsystems** enable the access to the many-core processor



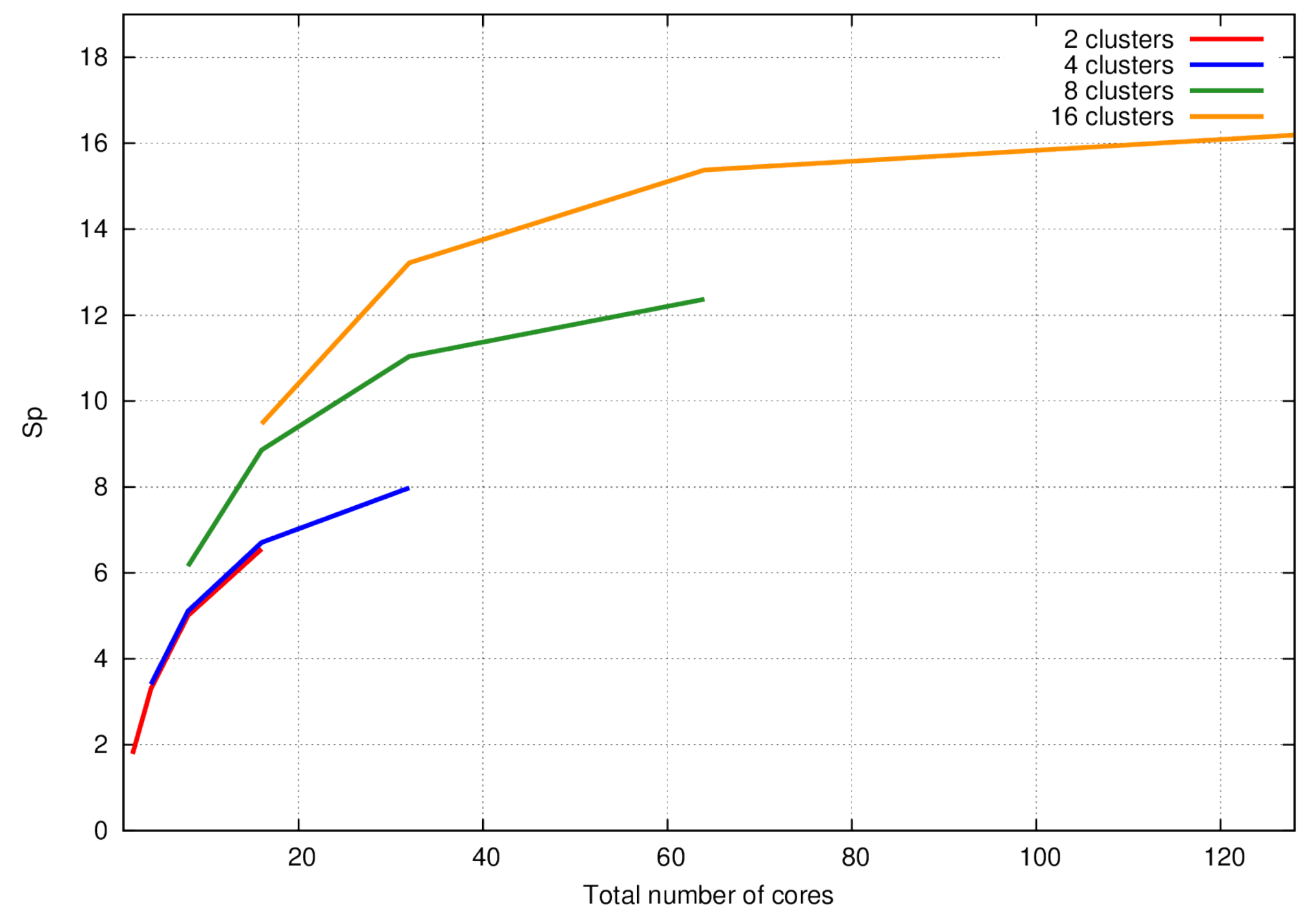
- It uses message passing (**MPPA IPC API**) and shared memory programming models (OpenMP and Pthreads)
- The MPPA IPC API provides connectors for communication and synchronization of processes

Performance Evaluation

- **Inter-cluster** parallelization (N clusters with 1 core) vs. **Intra-cluster** parallelization (N cores at 1 cluster)



- **Inter- and intra-cluster parallelization** are jointly used (N cores at M clusters)



CISTER Research Centre/INESC-TEC
ISEP, Polytechnic Institute of Porto
Rua Dr. Antº Bernardino de Almeida, 431
4200-072 PORTO Portugal
tel: +351-228340502
fax: +351-228340509
http://www.cister.isep.ipp.pt
cister-info@isep.ipp.pt