MPSoC'17 → MPSoC'22 → MPSoC'24

Scaling a Shared Memory Manycore

→

RAN GINOSAR

TECHNION
Israel Institute
of Technology

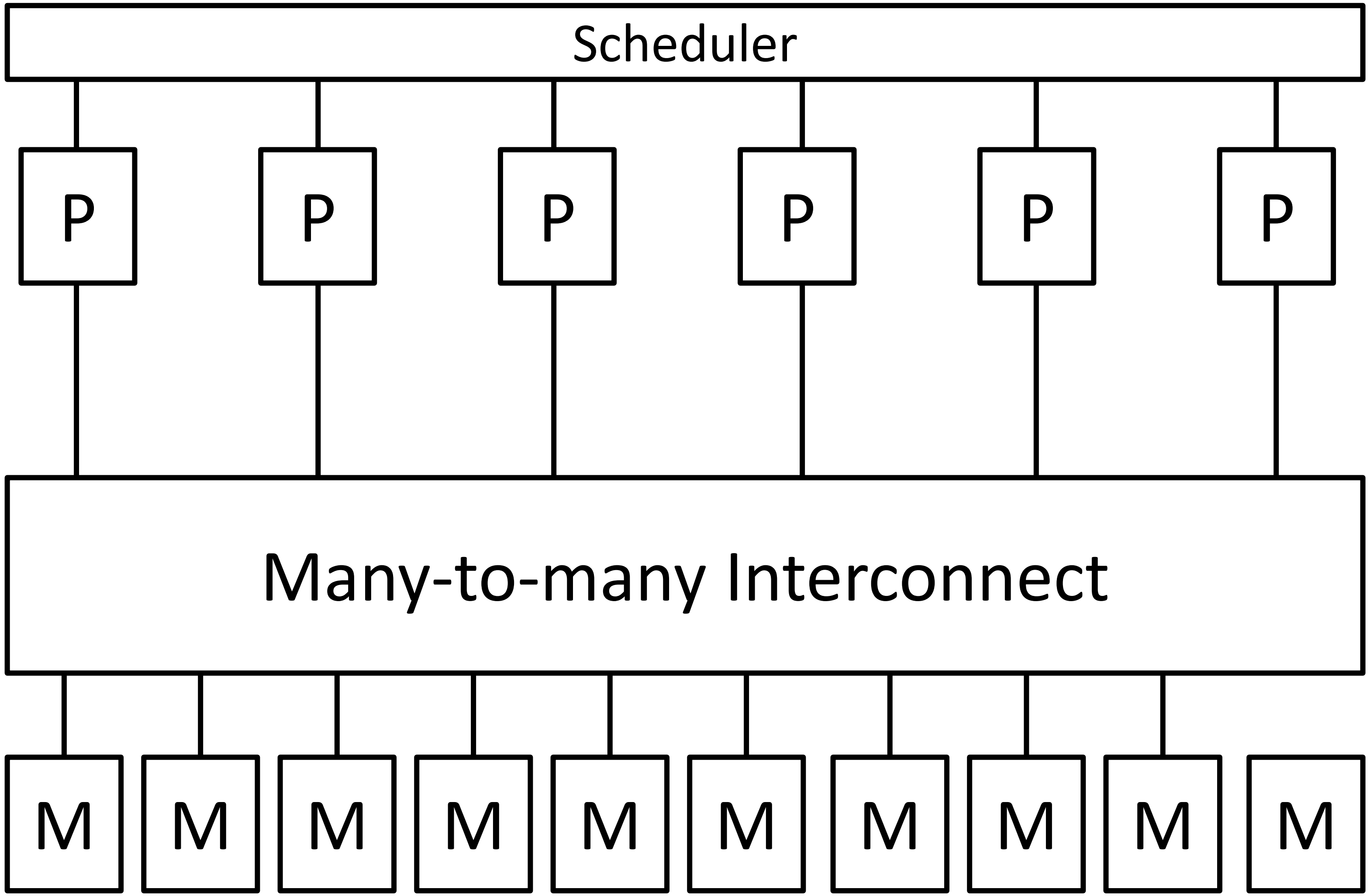
RAMON.SPACE

Contents

- Shared Memory Manycore
- Research Questions:
    - Scale Up
    - Scale Out

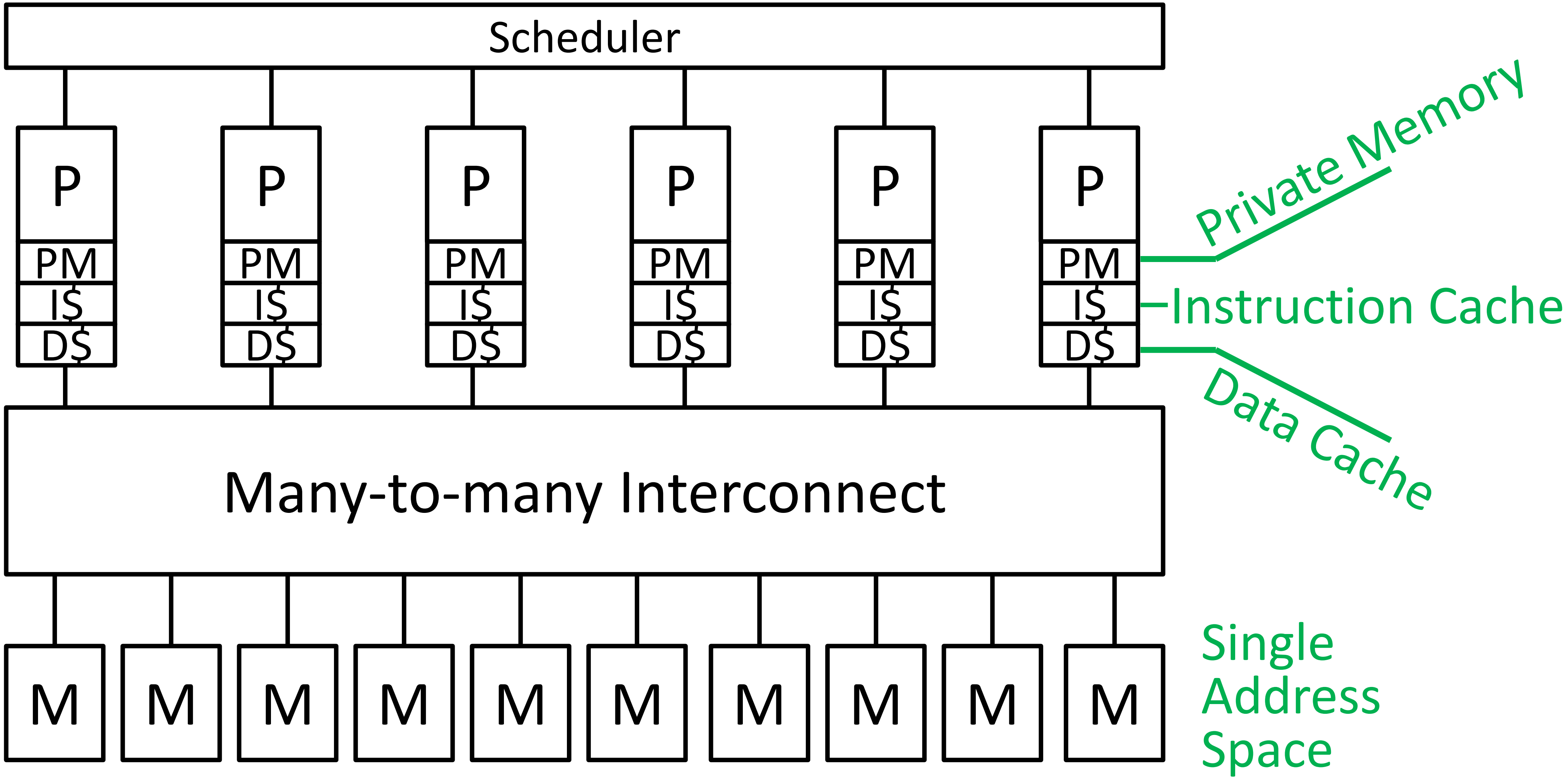
Shared Memory Manycore

RC64



Shared Memory Manycore

RC64



Scheduler

P P P P P P

PM IS DS

Many-to-many Interconnect

M M M M M M M M M M

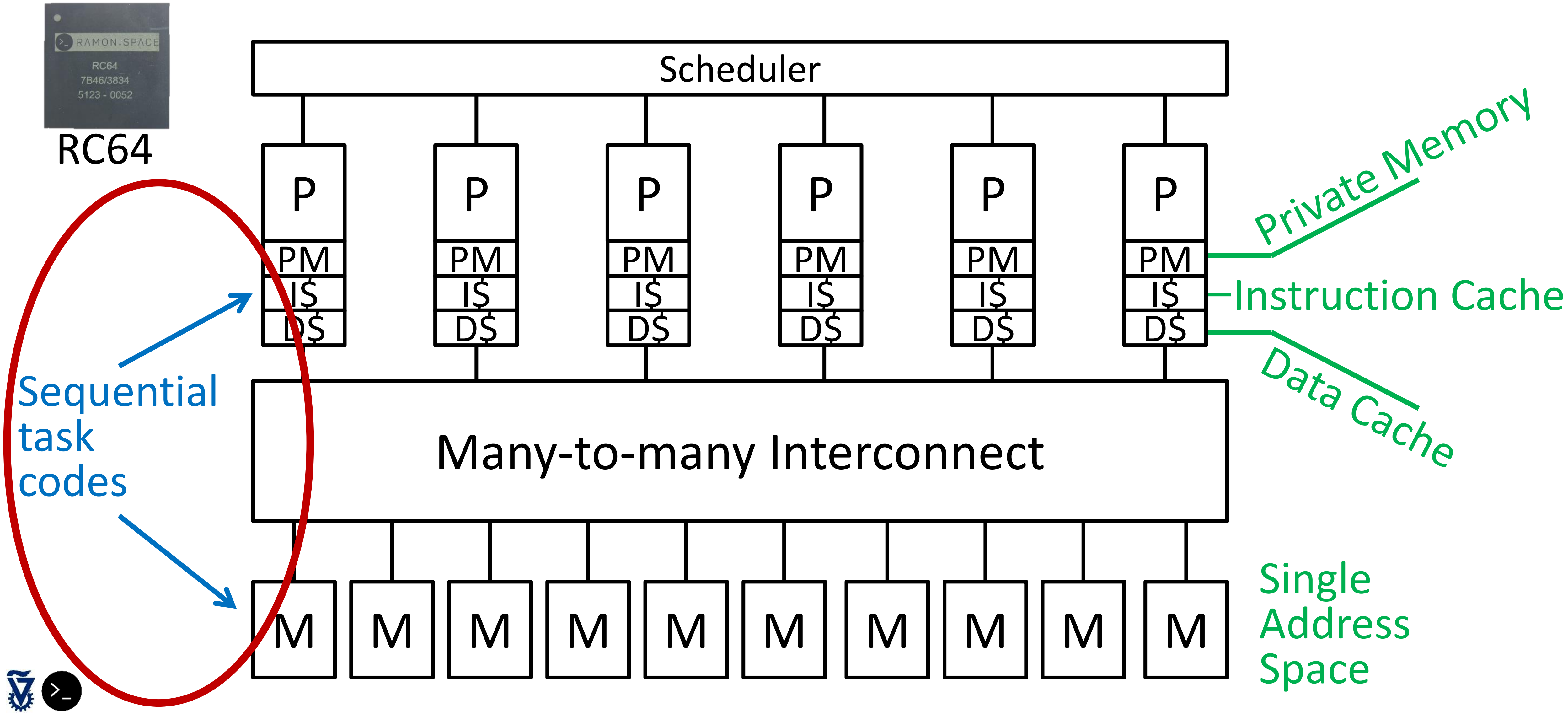
Private Memory

Instruction Cache

Data Cache

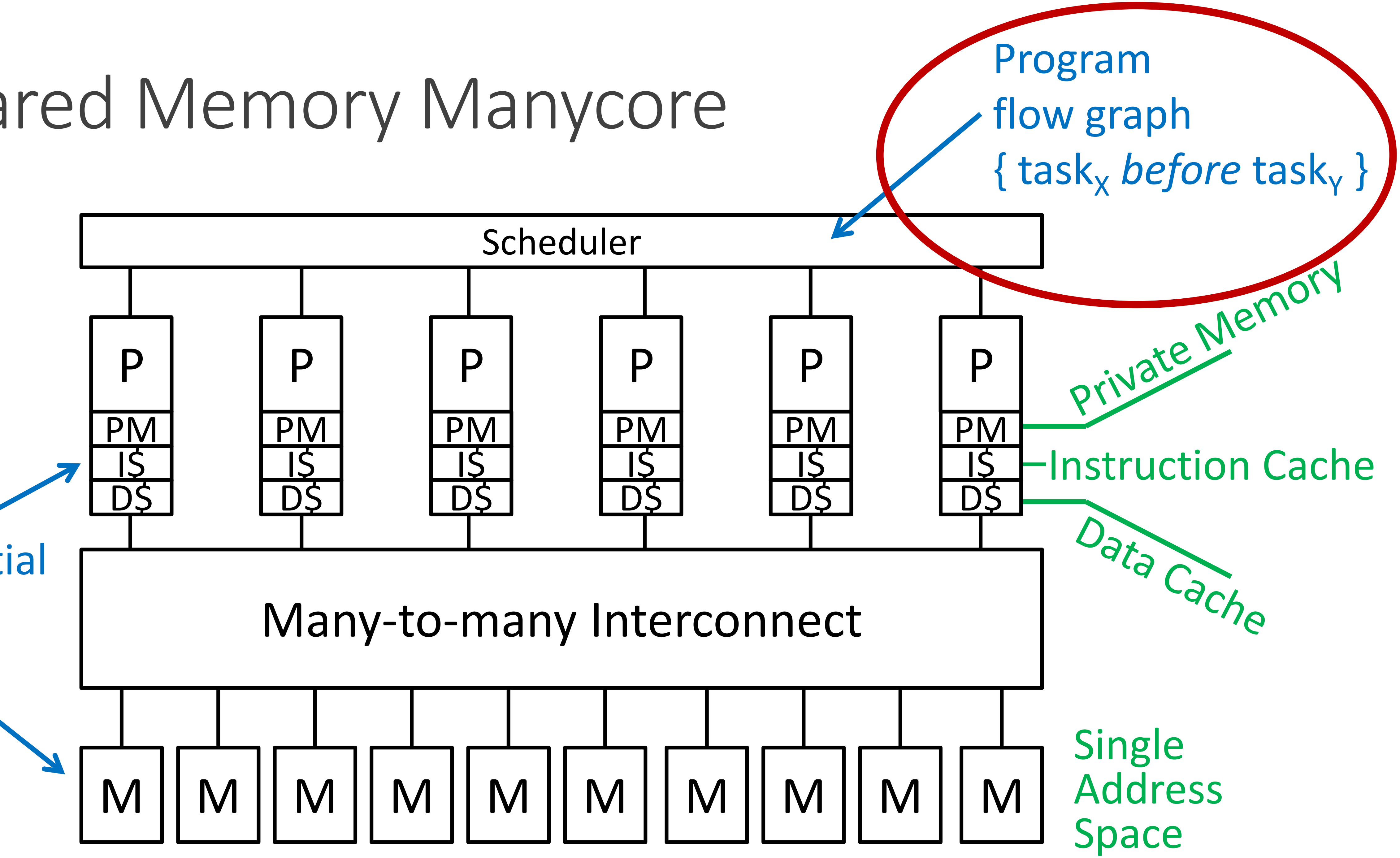
Single Address Space

Shared Memory Manycore



Shared Memory Manycore

RC64



Program flow graph { task_X *before* task_Y }

Private Memory

Instruction Cache

Data Cache

Sequential task codes

Single Address Space

Shared Memory Programming Model

- “Software is king”
- Main purpose of shared memory architecture:
  ENABLE SIMPLE SOFTWARE
- Programming model resembles PRAM-CREW
  - Parallel Random Access Machine—Concurrent Read, Exclusive Write
  - Single program execution (manycore ≠ multicore)
  - No need for cache coherency
- Sequential tasks executed by cores
  Task-dependency-graph executed by scheduler [1]

[1] R. Ginosar *et al.*, "RC64: High performance rad-hard manycore," *2016 IEEE Aerospace Conference*, doi: 10.1109/AERO.2016.7500697.

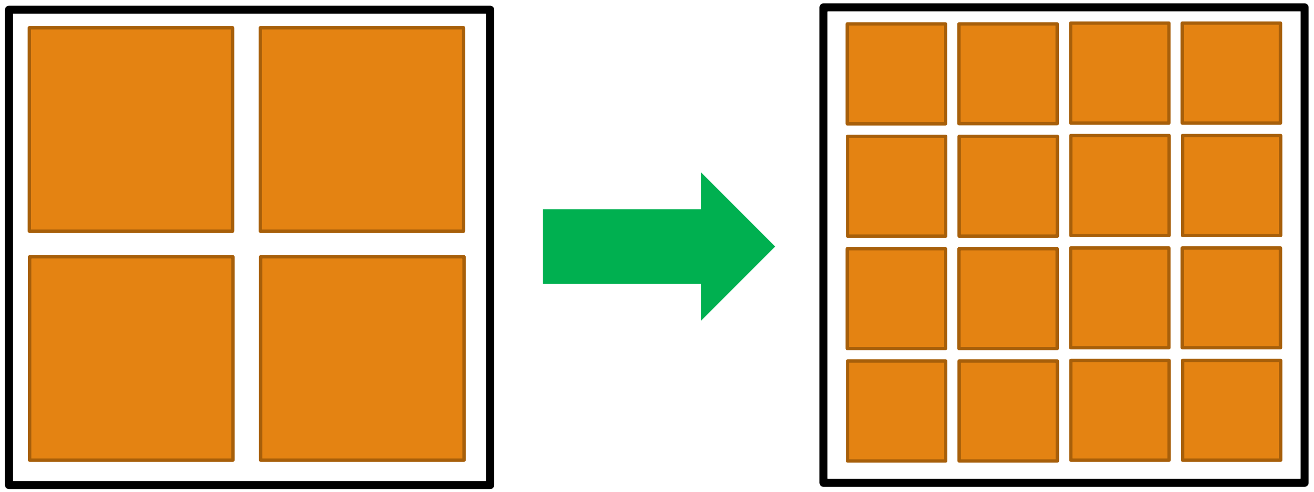
Shared Memory Observations

- Memory read = cache line fetch
- Memory write = cache line update / word update
  - Write-back vs. write-through
- Memory access time is unknown and it varies
- No data locality is assumed
  - All memory banks are *perceived* similarly close (similarly far) from every processor
- No data locality is allowed !

Advantages

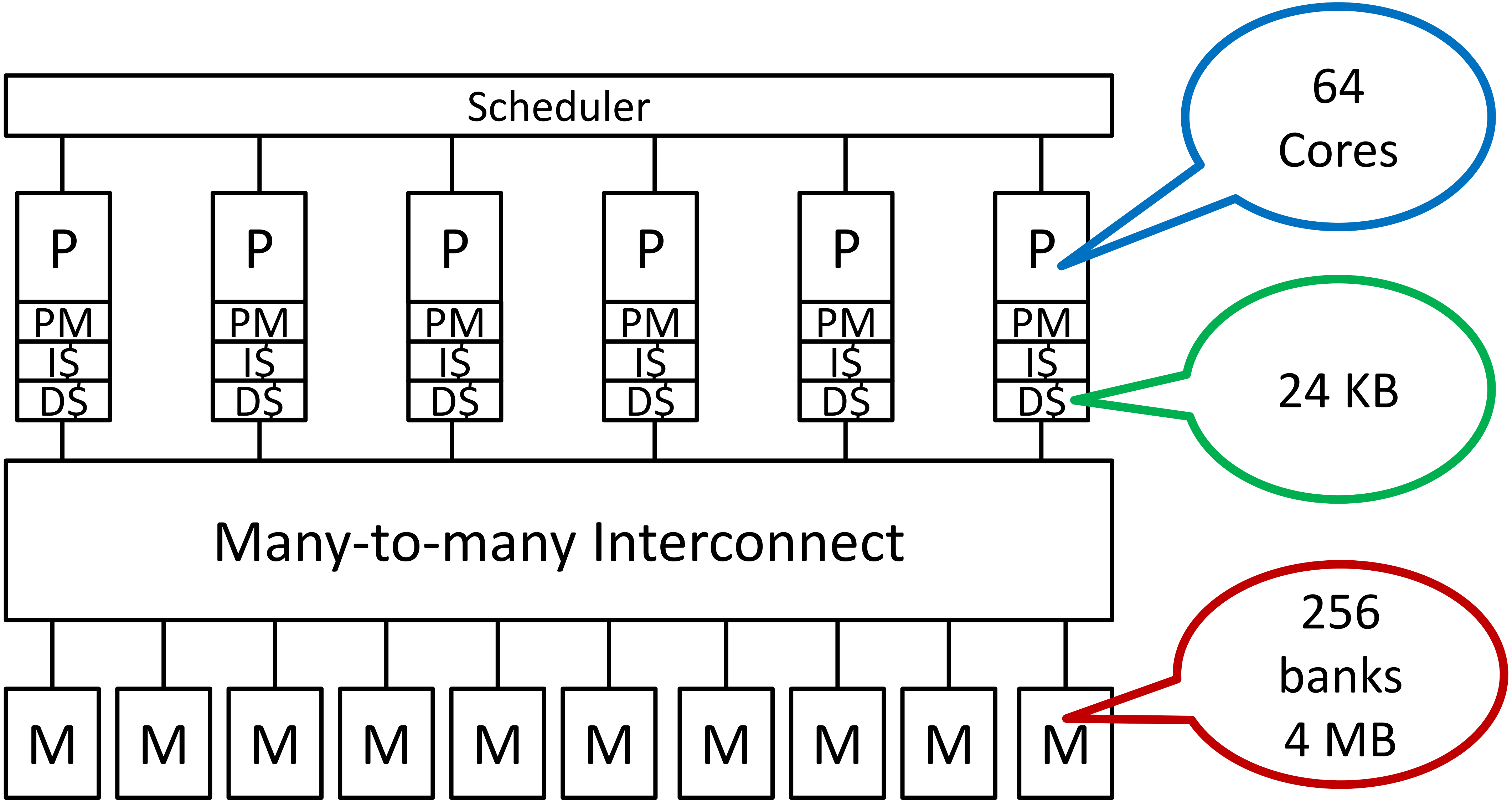
- Simple architecture
  - Simple to program
  - Simple to compile
  - Simple to implement
  - Simple to scale ← subject of this talk
- High CPU utilization
- High power efficiency

Scale Up



Baseline

RC64



Scheduler

P P P P P P

PM PM PM PM PM PM

I$ I$ I$ I$ I$ I$

D$ D$ D$ D$ D$ D$

Many-to-many Interconnect

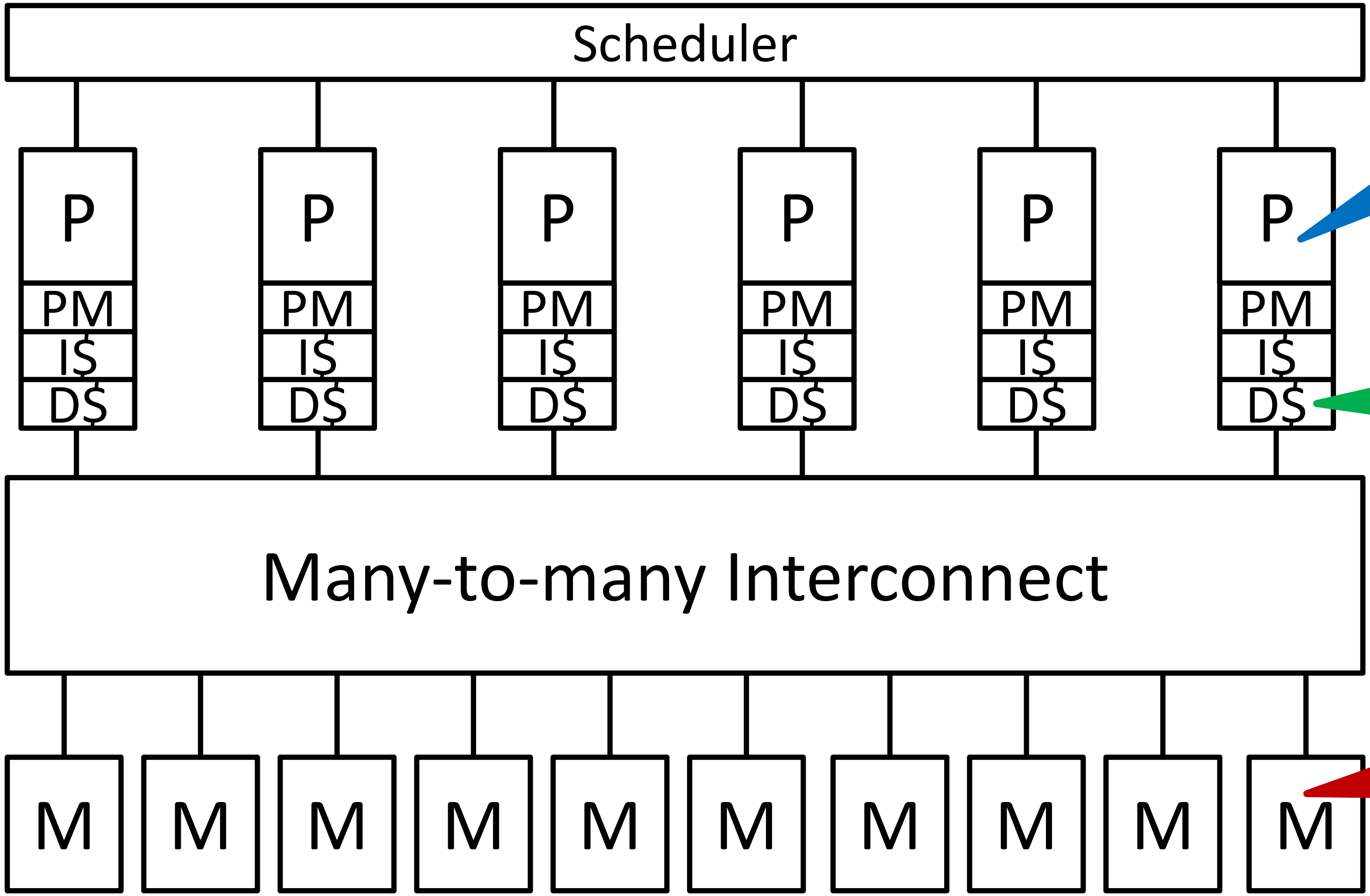
M M M M M M M M M M

64
Cores

24 KB

256
banks
4 MB

1. Scale Up: more of the same

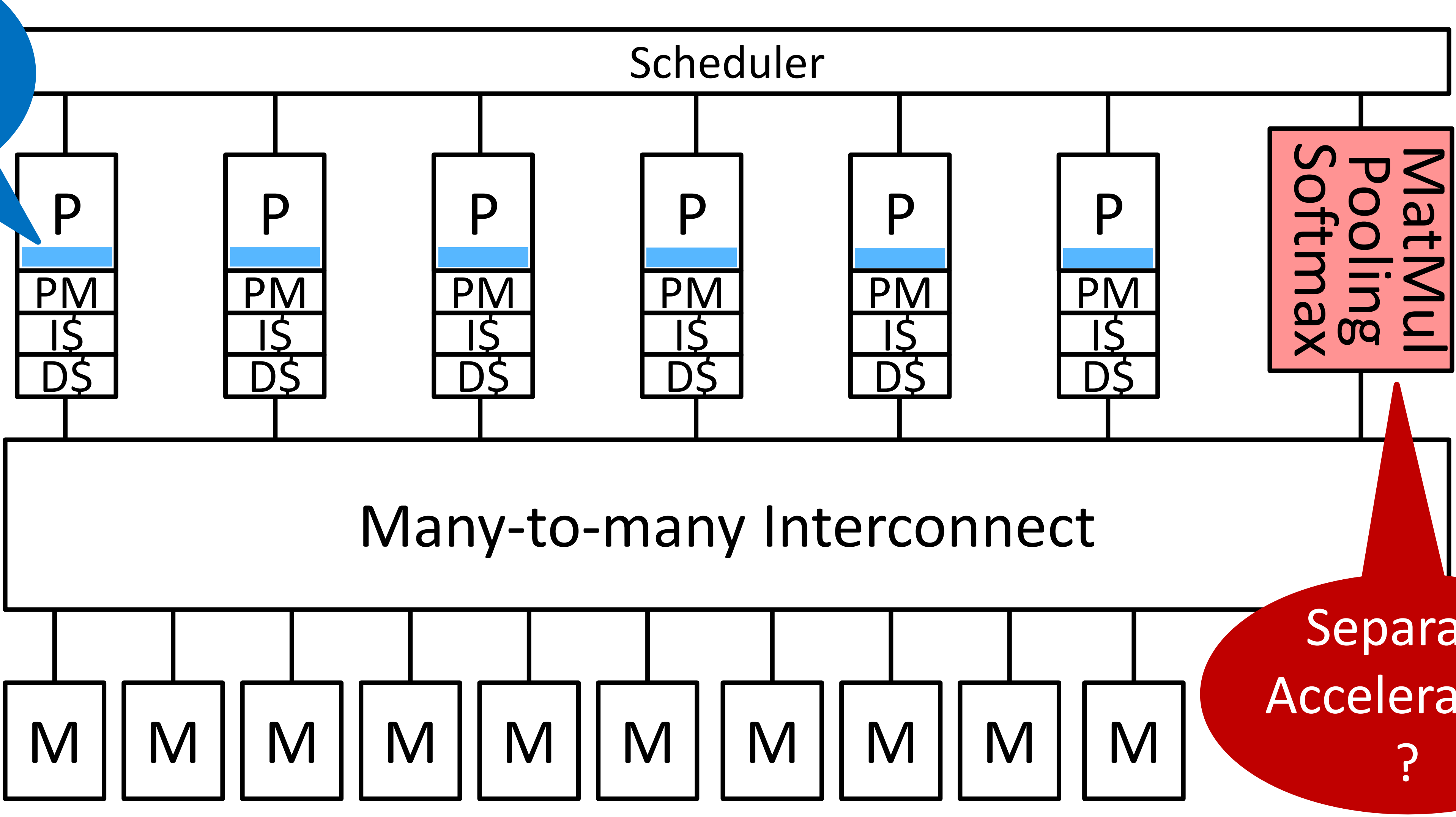


More Cores ?

Larger Cache ?

Larger Shared Memory ?

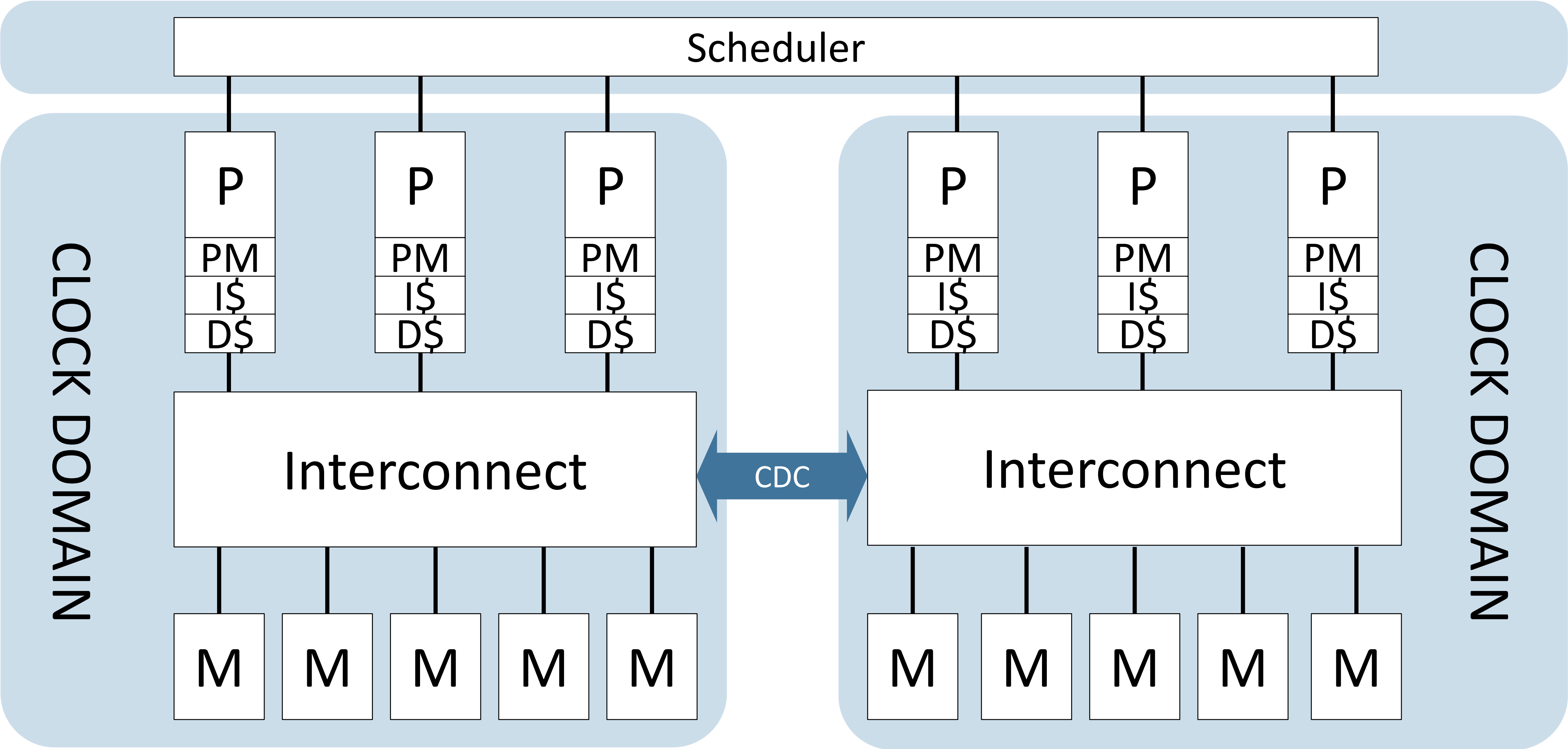
2. Scale Up: Acceleration—per core or per chip ?



Vector unit per core ?

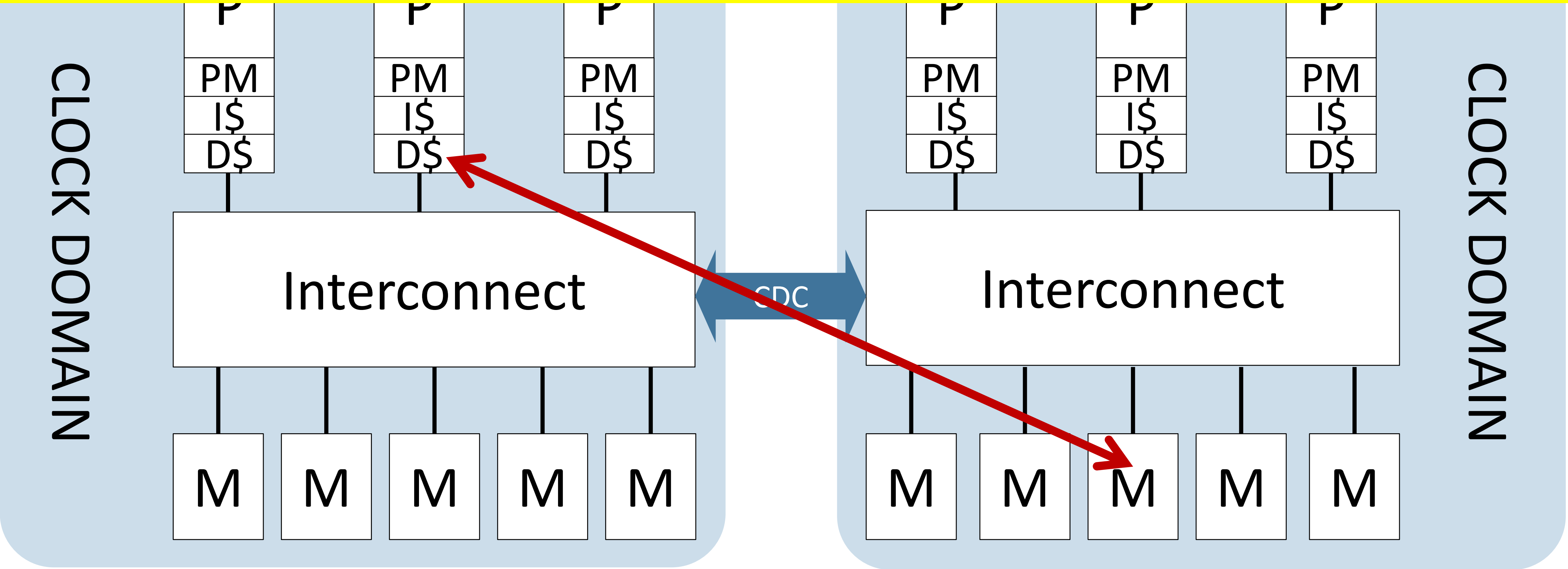
Separate Accelerators ?

3. Scale Up: GALS (transparent) Clusters ?



RECALL:

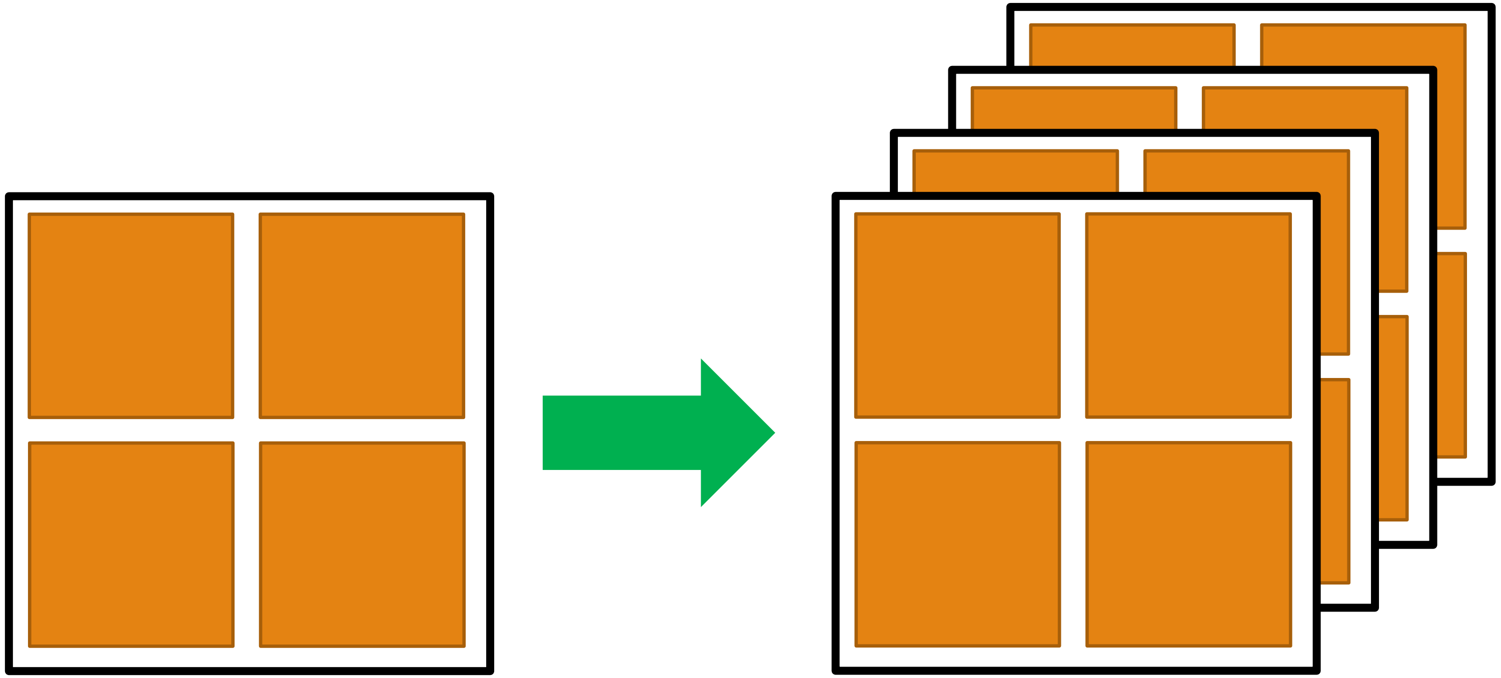
- Memory access time is unknown and it varies
- No data locality is assumed
  - All memory banks are *perceived* similarly close (similarly far) from every processor
- No data locality is allowed !



Scale Up Evaluation ?

- Architecture level simulations
- Which benchmarks?
  - CNN & LLM inference
  - FFT
  - Matrix Multiply
  - Optional 4G/5G codec: Turbo + LDPC

Scale Out



Multiple **chips**
and/or
Multiple **chiplets** in same package

Scale Out

- Maintain shared memory architecture over multiple chips/chiplets
  - “Architecture” = the view presented to software
  - Software should see a “single address space” shared over multiple chips/chiplets

RECALL:

- Memory access time is unknown and it varies
- No data locality is assumed
  - All memory banks are *perceived* similarly close (similarly far) from every processor
- No data locality is allowed !

CHALLENGE

- Transparent & efficient access to remote memory

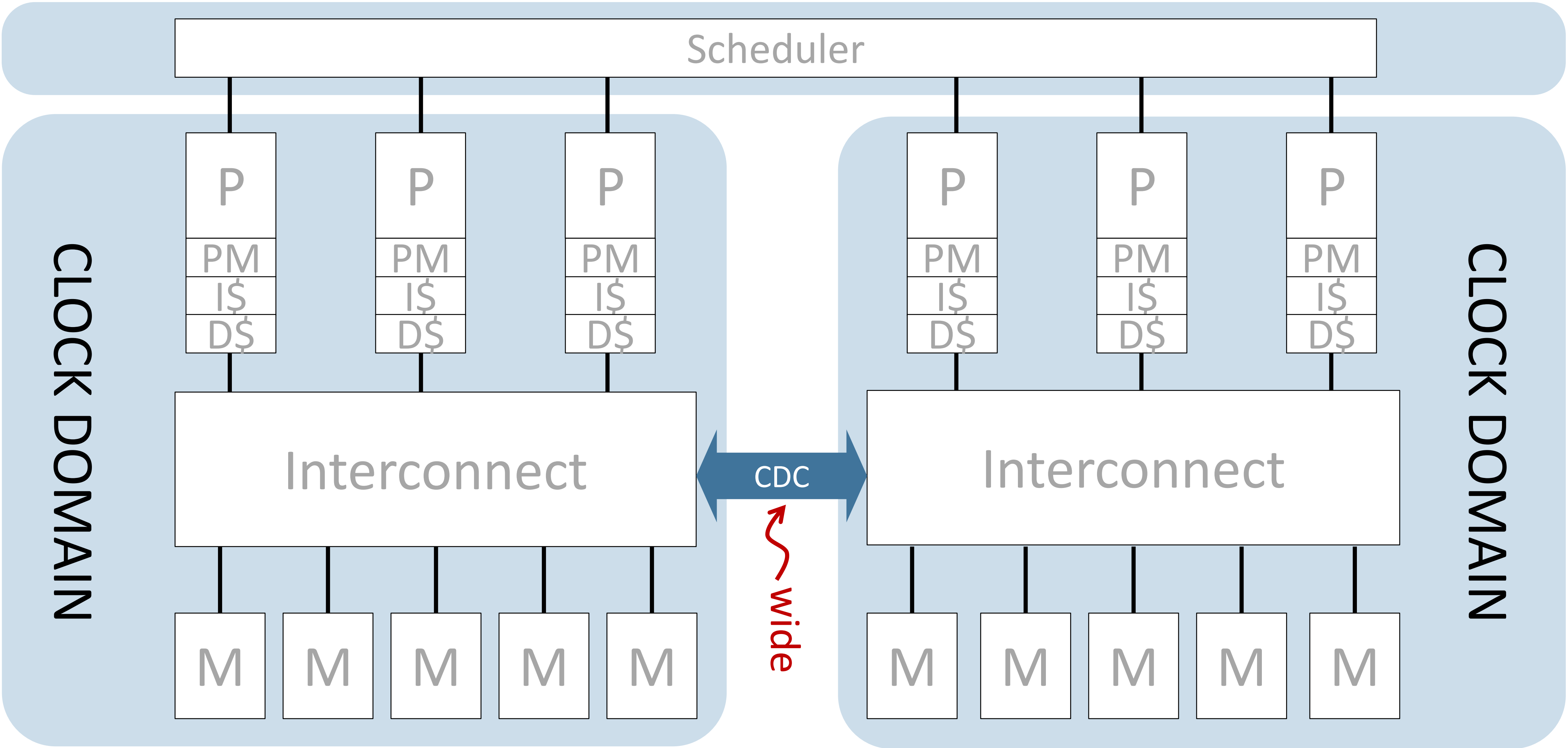
Transparent Access to Remote Memory

- Already addressed in large compute systems—datacenters & HPC
- Alternatives:
  1. Non-transparent message passing (MP)
     - SW issues SEND() & RECV(), handled by OS → context switch & long latency
  2. RDMA (infiniband, RoCE, NVlink)
     - SW employs LOAD / STORE, HW catches and executes → shorter latency
  3. Hybrid MP/RDMA
     - Two SW layers: *Application* (e.g. AI training) assumes shared memory, *Framework* selects MP or RDMA
- Too complex for our purpose

Lower Complexity Shared Memory

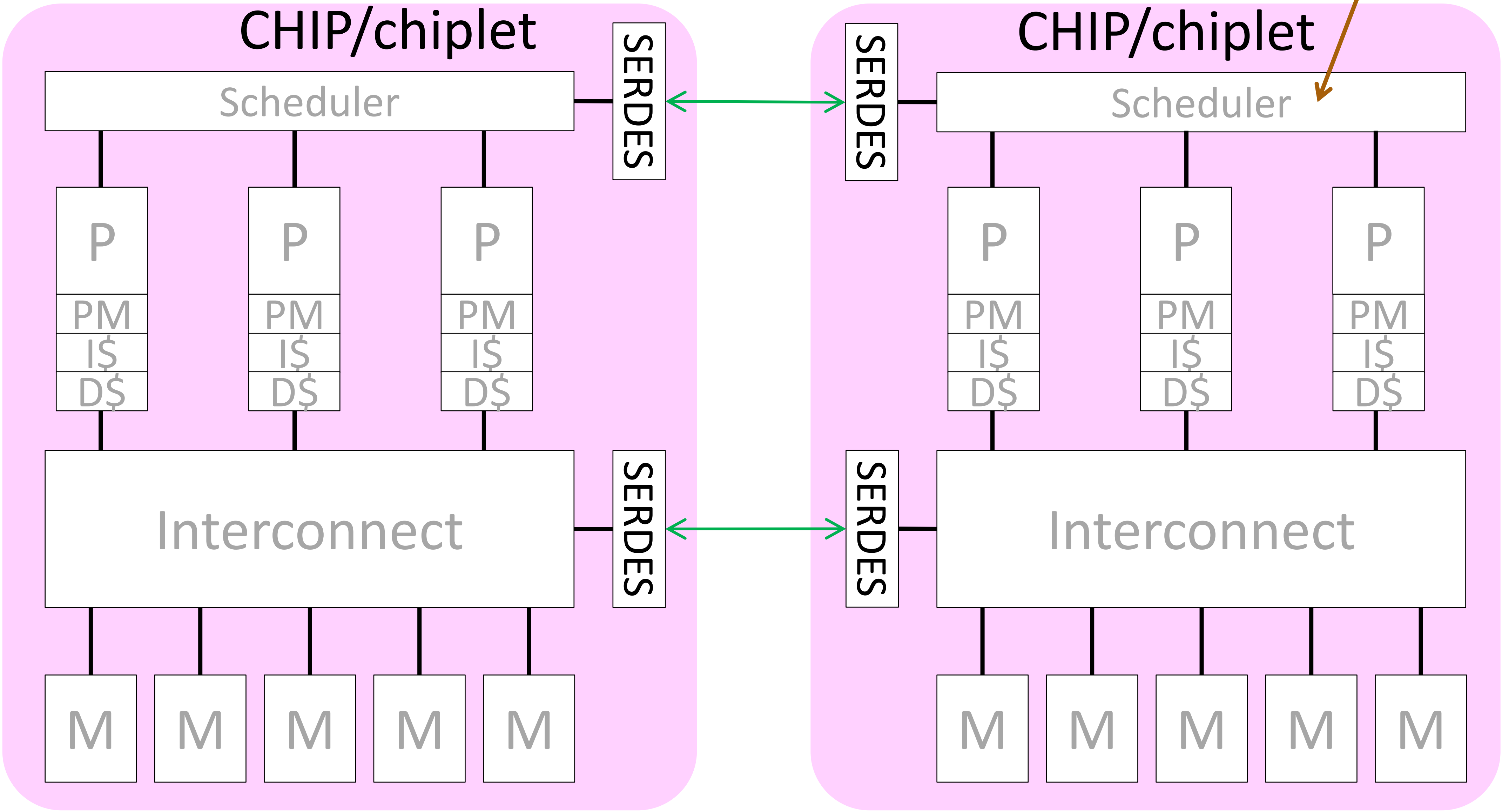
- More suitable for small systems (Edge, embedded)
- So far, we have used MP
  - Shared memory within a chip, distributed computing among multiple chips
- Extension of GALS is possible and "easy"
  - Drawback: wide datapath between chips → too many I/O pins
- Instead, use SERDES and High-Speed Serial Links (HSSL)

RECALL: GALS within a chip

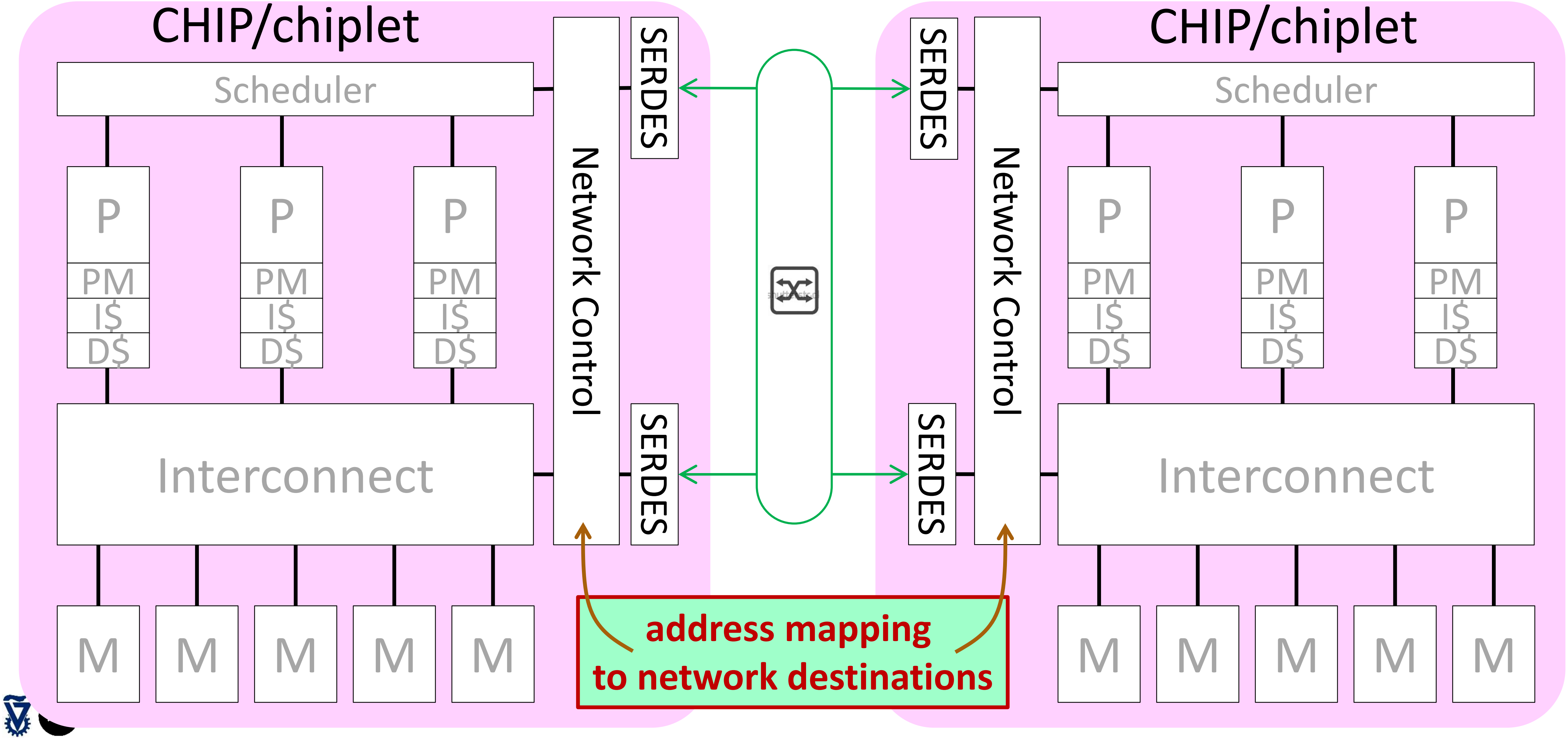


Study distributed scheduler

Extend to Multiple Chips/Chiplets

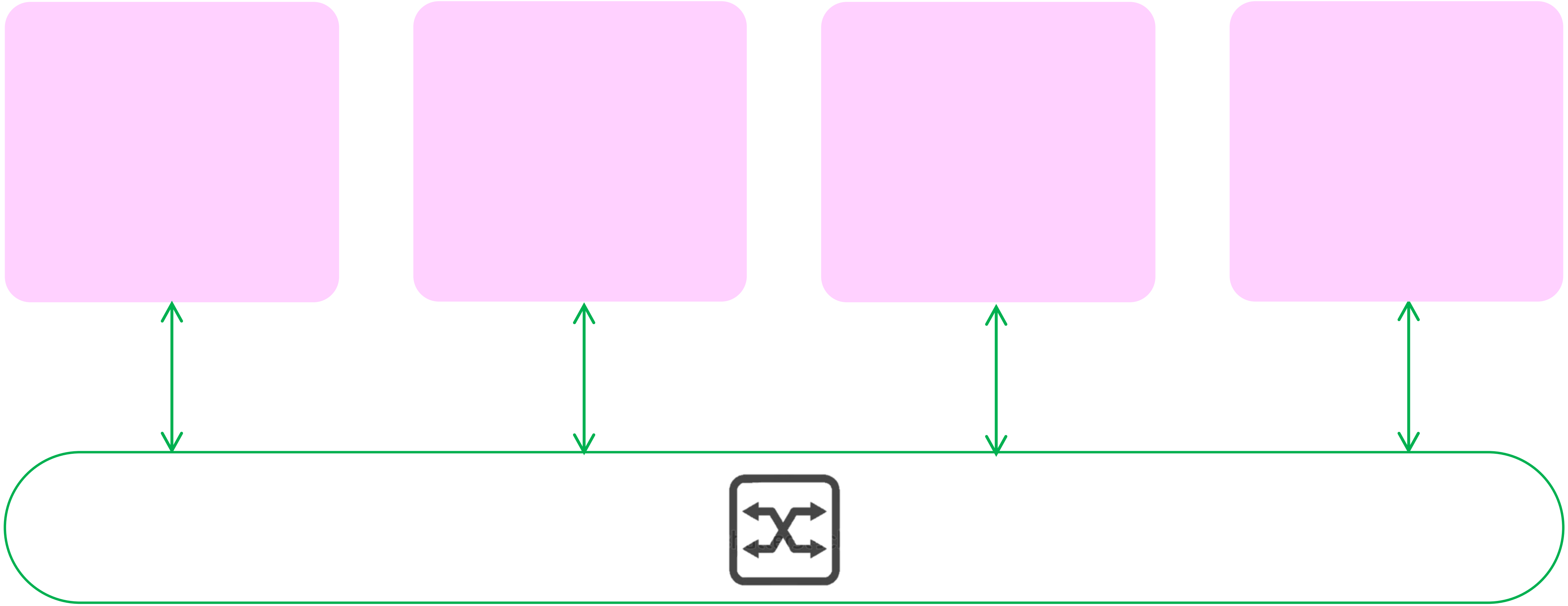


Actually, a “transparent” network is needed



address mapping
to network destinations

Network is needed to scale beyond 2 chips/chiplets



NoC extended to NoC* (spanning multiple chips/chiplets)

Scale Out Evaluation ?

- Architecture level simulations (same as for Scale Up Evaluation)
- Which benchmarks?
  - CNN & LLM inference
  - FFT
  - Matrix Multiply
  - Optional 4G/5G codec: Turbo + LDPC

Summary

- Shared memory manycore is “simple”
- As a result, it has been shown useful
- Now it is time to scale
- Research questions:
    - How to scale up?
    - How to scale out?
    - How to evaluate scaling?

TECHNION
Israel Institute
of Technology

RAMON.SPACE