Barcelona Supercomputing Center
Centro Nacional de Supercomputación

The Current Mess (and Power) of Heterogeneity in HPC

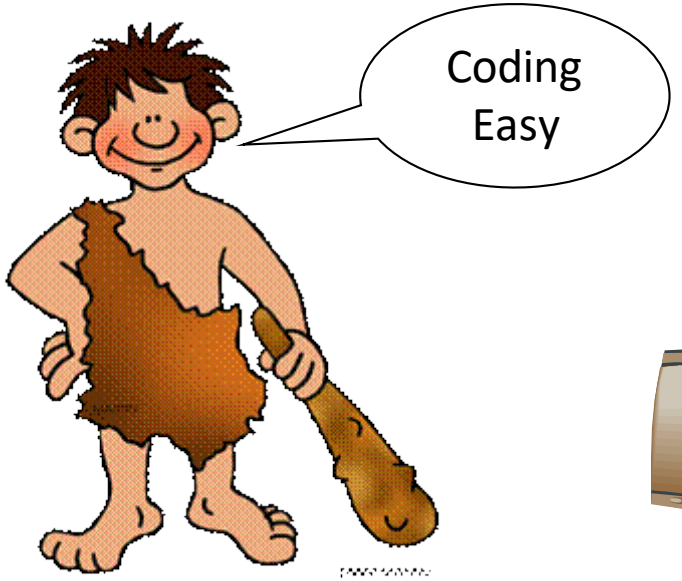
Antonio J. Peña

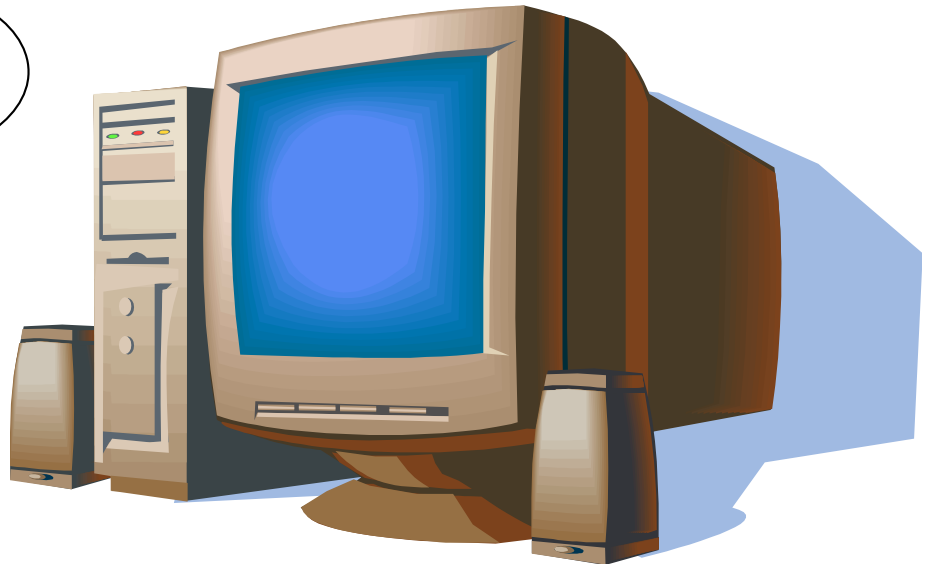
Leading Researcher and Group Manager

Apr. 22, 2024

AccelCom
Accelerators & Communications for HPC

Back in the Computers Stone Age...

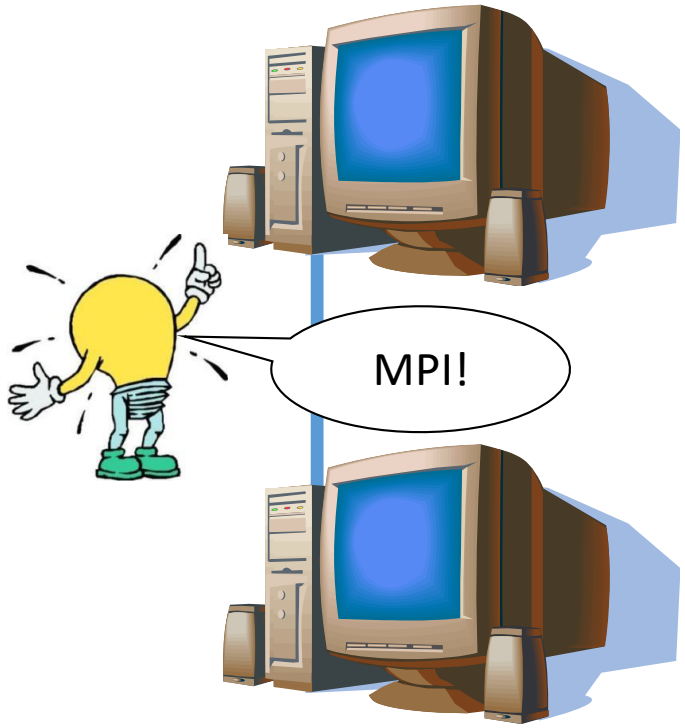


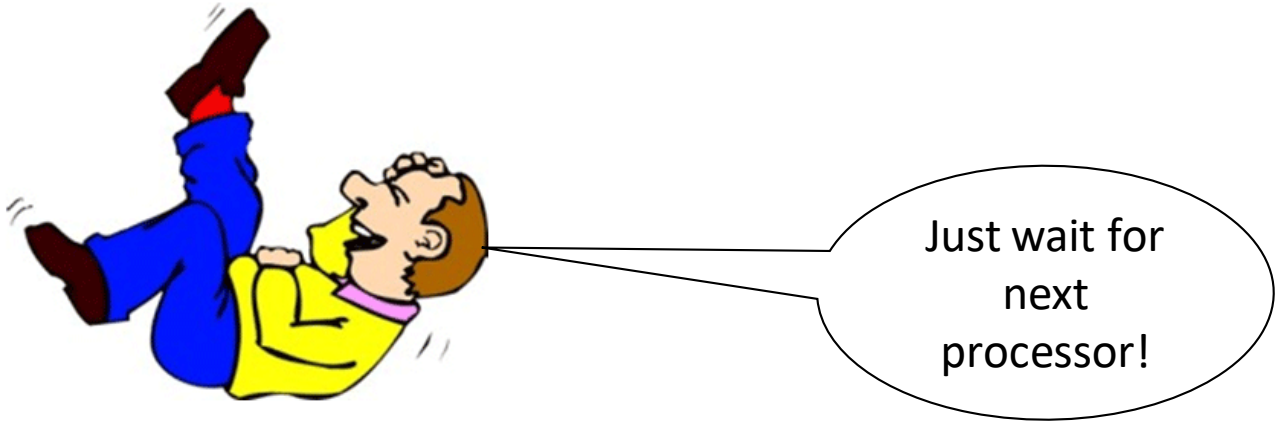


- Multi-what?
- What is a core?

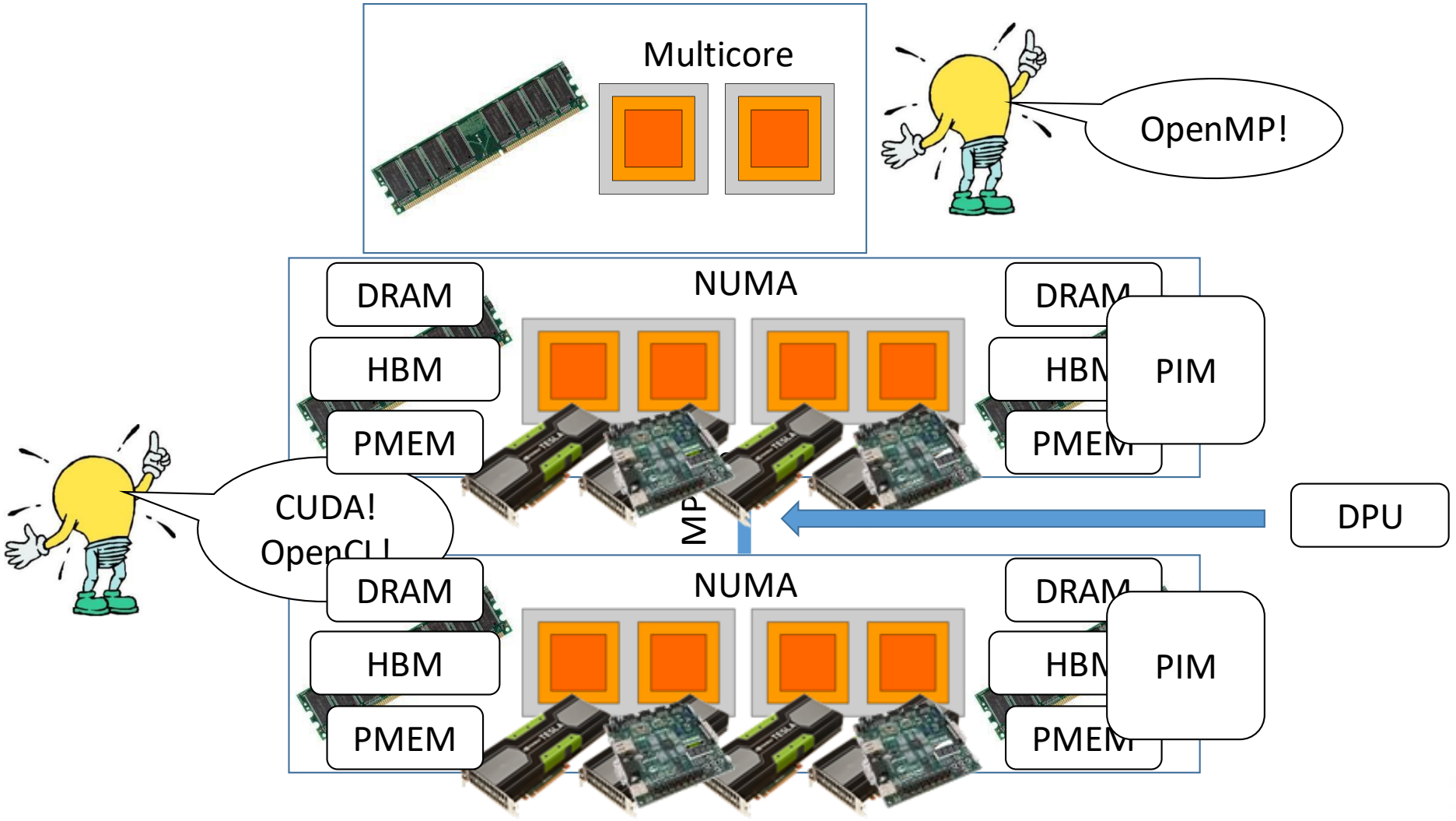
By 1994…


Hmmm… me need more performance
What if…
Damn! Coding not so easy!


MPI!


Just wait for next processor!

With the end of Frequency Scaling…



DPU e.g.: NVIDIA BlueField III

- Overview HW features
  - 16 x (weak) ARM cores
  - DDR5 Memory
  - Network Controller (of course)
  - PCIe switch

- Many possible benefits:

N. Sarkauskas, *et al.*, "Large-message nonblocking MPI_Iallgather and MPI Ibcast offload via BlueField-2 DPU," *HiPC* 2021

MPI Acceleration
DL
…
Other Offloads?

A. Jain *et al.*, "Optimizing distributed DNN training using CPUs and BlueField-2 DPUs," *IEEE Micro* 2022

PIM

- Quite some papers…
- So far UPMEM's is the only commercially available
  - Here DPU = DRAM Processing Unit ☺
  - Weak compute power
    - Best for memory-bound apps
    - Best for simple arithmetics (e.g., bitwise, int add…)
    - Not good at multiply

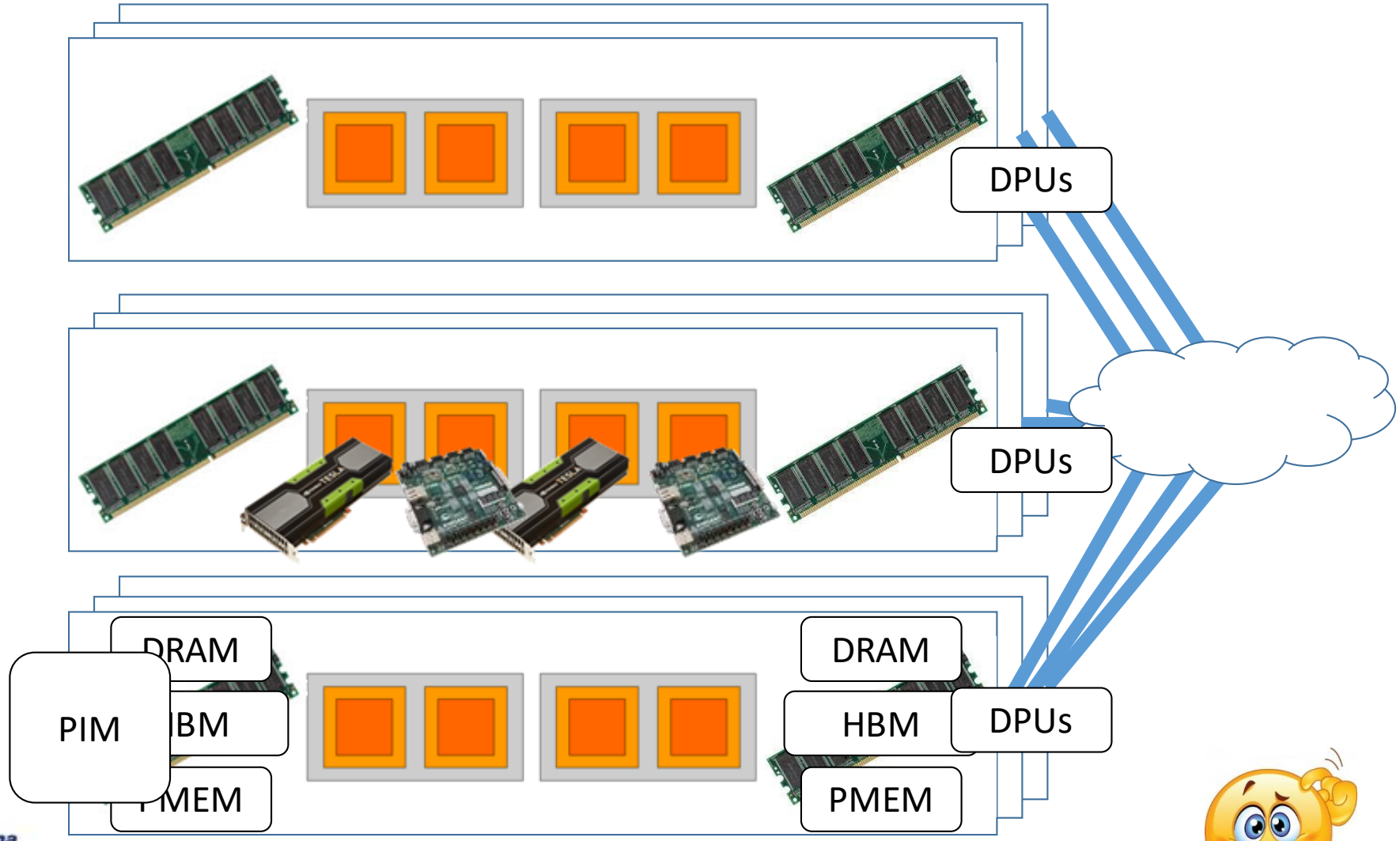
J. Gomez-Luna *et al.*, "Benchmarking a new paradigm: An experimental analysis of a real processing-in-memory Architecture", arXiv:2105.03814

- We'll need to work on the programming model…

```
#include <assert.h>
#include <dpu.h>
#include <dpu_log.h>
#include <stdio.h>

#ifndef DPU_BINARY
#define DPU_BINARY "./helloworld"
#endif

int main(void) {
 struct dpu_set_t set, dpu;

 DPU_ASSERT(dpu_alloc(1, NULL, &set));
 DPU_ASSERT(dpu_load(set, DPU_BINARY, NULL));
 DPU_ASSERT(dpu_launch(set, DPU_SYNCHRONOUS));

 DPU_FOREACH(set, dpu) {
  DPU_ASSERT(dpu_log_read(dpu, stdout));
 }

 DPU_ASSERT(dpu_free(set));

 return 0;
}
```

Source: https://sdk.upmem.com/2021.3.0/02_HelloWorld.html

Hold on… Even More Fun!



Power? Nightmare?

- Pros
  - Specialized HW
    - Awesome performance
    - More energy friendly
    - Exascale (and beyond…)
- Challenges
  - Programming models
  - Programmability
  - Coding productivity
  - Code maintainability
  - Portability
  - Performance
  - Performance Portability
  - HW design / architectures

Agenda

The Heterogeneity of the Homogeneity

Heterogeneous Processors

Heterogeneous Memory Systems

Distributed Heterogeneity

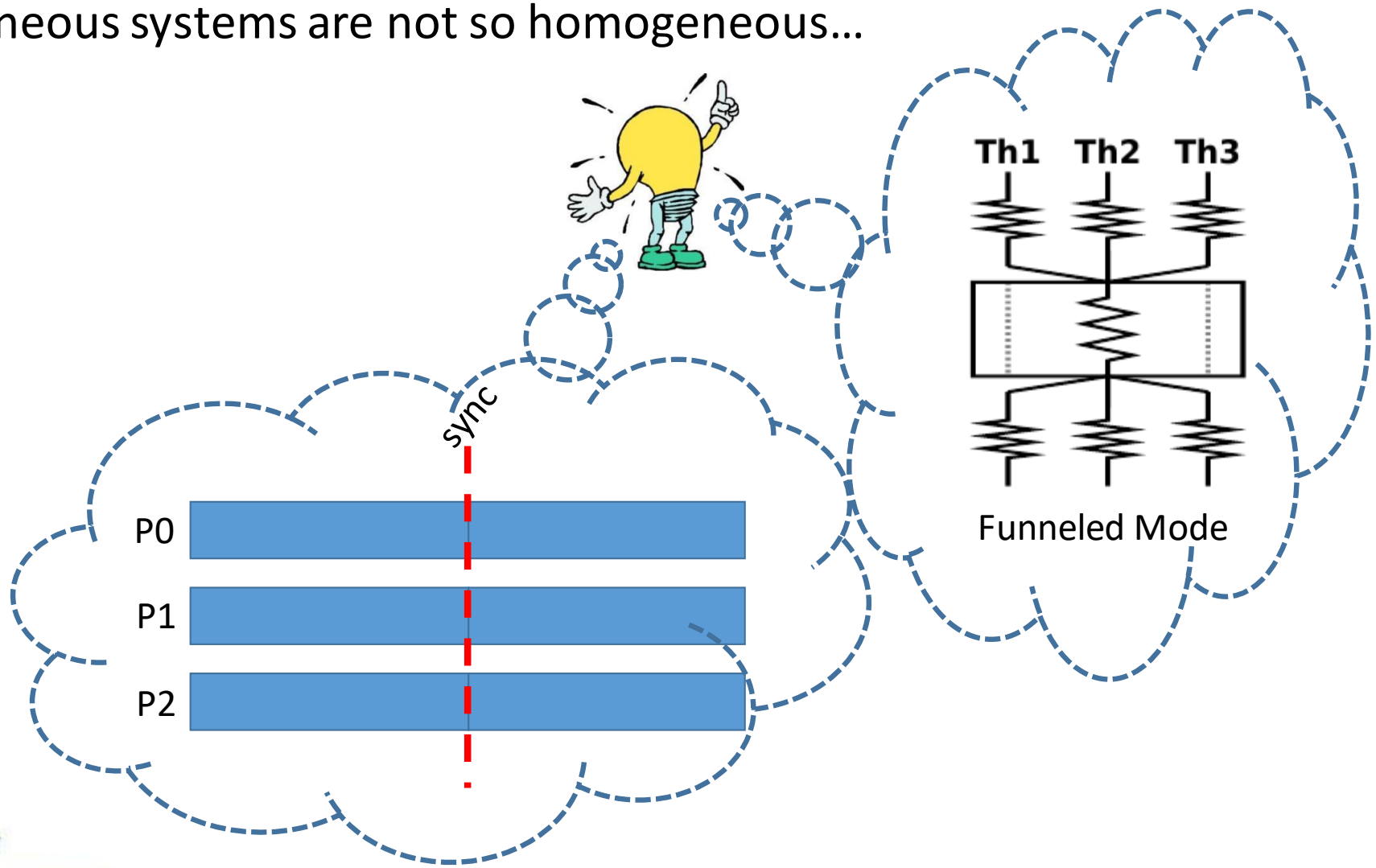
Putting it All Together?

Summary

The Heterogeneity of the Homogeneity

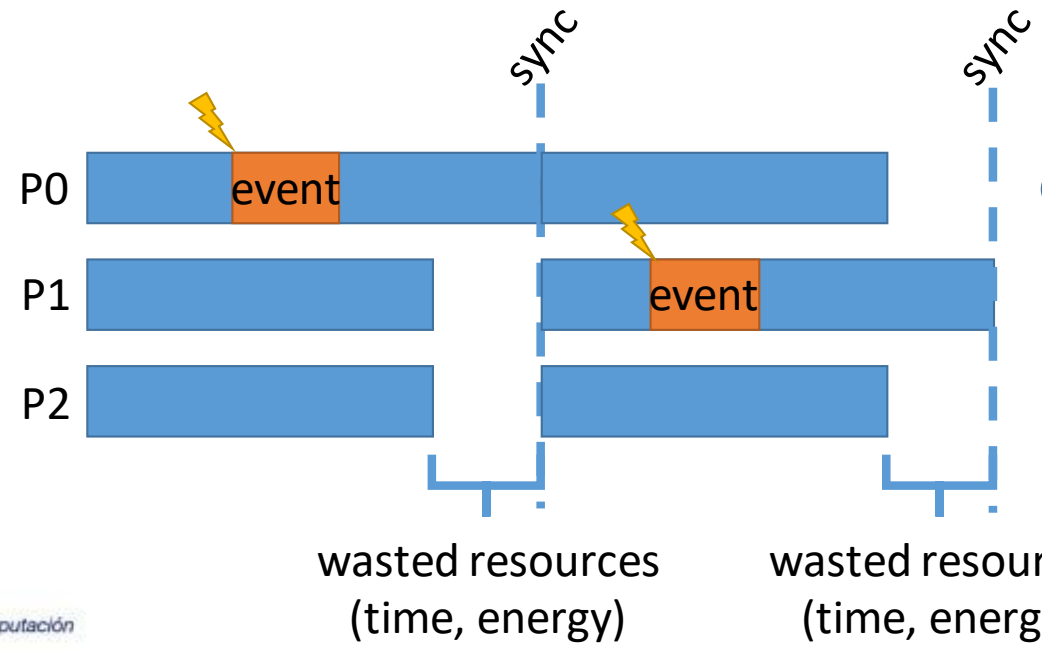
Performance Heterogeneity

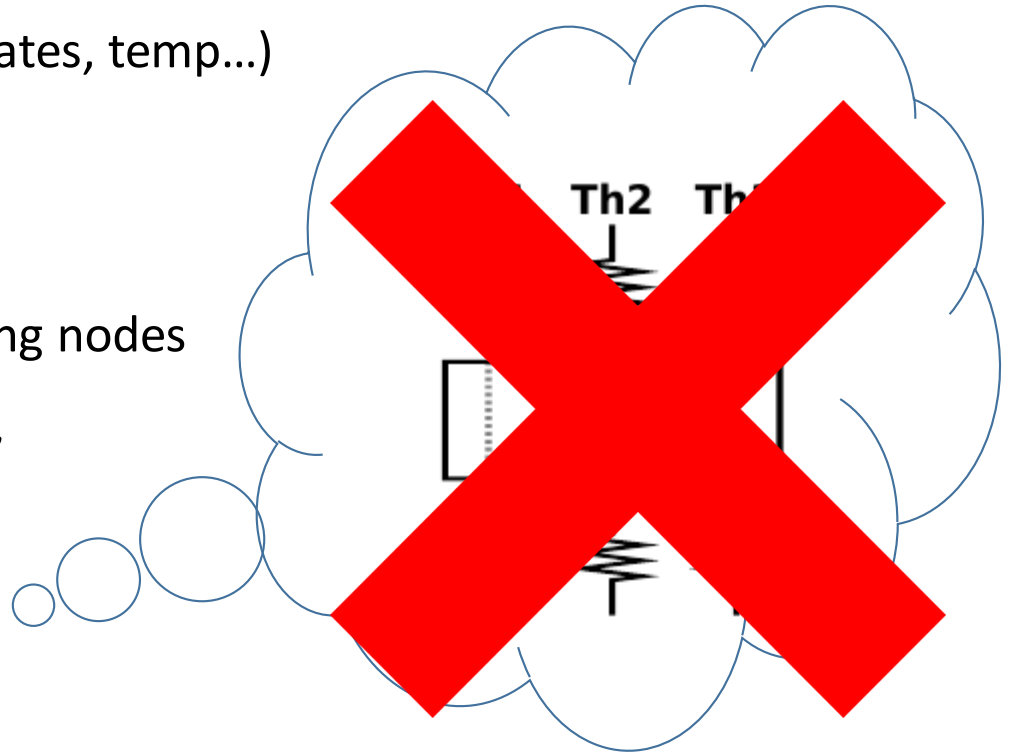
- Homogeneous systems are not so homogeneous…



Performance Heterogeneity

- Homogeneous systems are not so homogeneous…
  - OS noise (interrupts, running processes, energy saving states, temp…)
  - Manufacturing deviations
  - Eventual recoverable faults (ECC)
  - Resource contention (memory, network, …)
  - Irregular network congestion (3D Torus) or distance among nodes





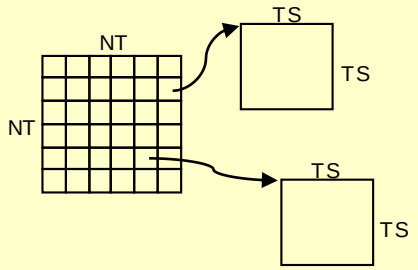
D. Skinner, W. Kramer, "Understanding the causes of performance variability in HPC workloads", IISWC 2005

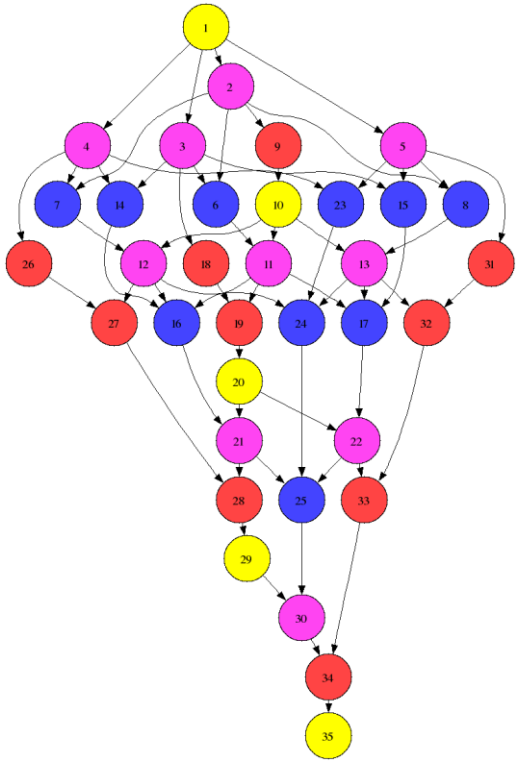
Performance Heterogeneity

- Smart SW may prevent performance issues derived from user's ideal view
- E.g., task decomposition, dynamic scheduling, malleability
- Scheduling left to the system SW / runtime: will adapt to timing variability
  - Runtime system for OpenMP task scheduling
  - Global resource manager (e.g., SLURM) for MPI malleability
  - …
- Main benefit: no need for "ninja programming"
  - Programming productivity
  - Performance
  - Portability
  - Runtime management overhead often pays off

Task-Based Programming

```
void Cholesky( float *A[NT][NT] ) {
int i, j, k;
for (k=0; k<NT; k++) {
  #pragma omp task inout (A[k][k])
  spotrf (A[k][k]) ;
  for (i=k+1; i<NT; i++) {
    #pragma omp task in (A[k][k]) inout (A[k][i])
    strsm (A[k][k], A[k][i]);
  }
  for (i=k+1; i<NT; i++) {
    for (j=k+1; j<i; j++) {
      #pragma omp task in (A[k][i], A[k][j]) inout (A[j][i])
      sgemm( A[k][i], A[k][j], A[j][i]);
    }
    #pragma omp task in (A[k][i]) inout (A[i][i])
    ssyrk (A[k][i], A[i][i]);
  }
}
```

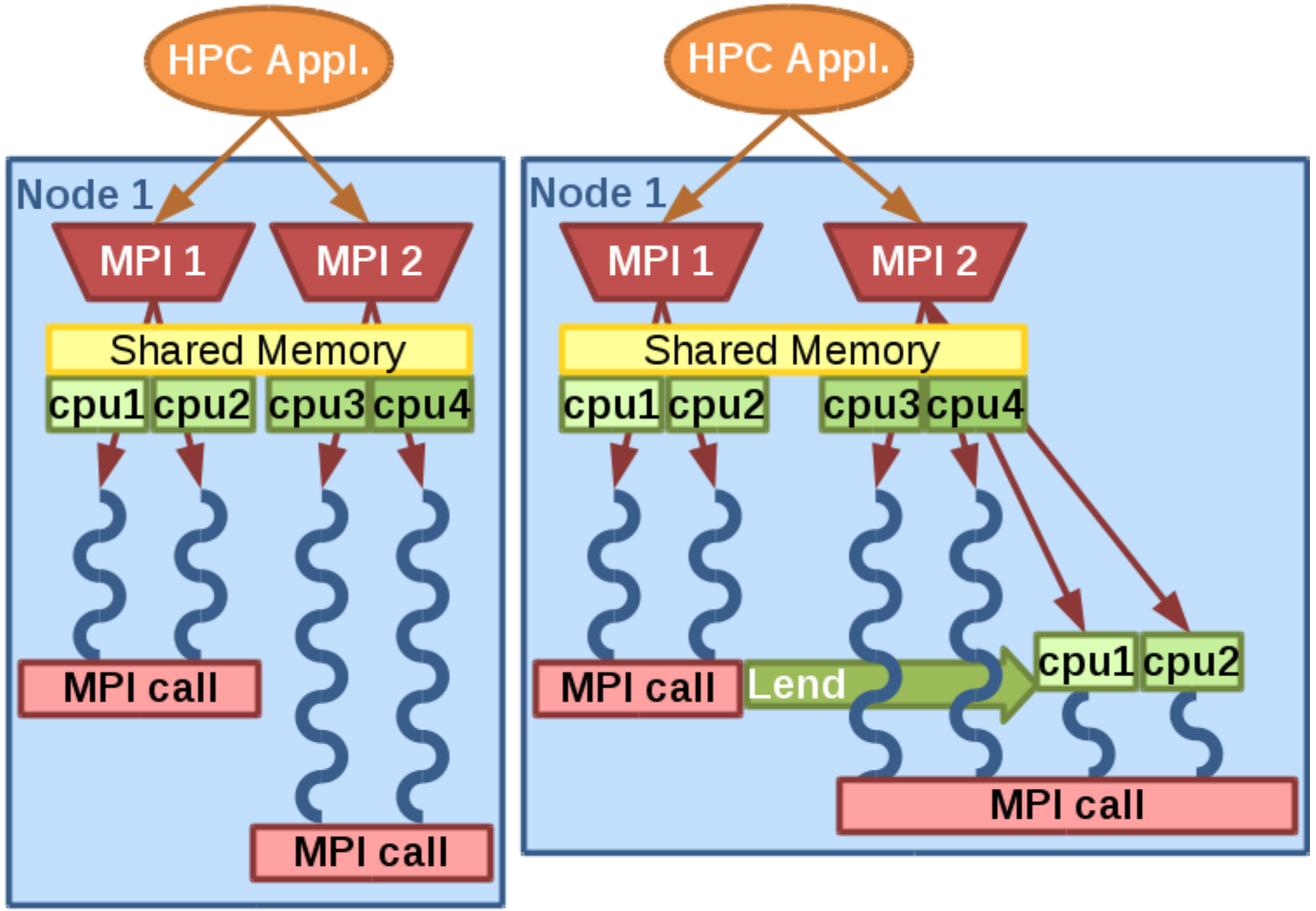




Decouple how we write/think (sequential) from how it is executed

A. Duran, E. Ayguadé, R. M. Badia, J. Labarta, L. Martinell, X. Martorell, J. Planas. "OmpSs: A proposal for programming heterogeneous multi-core architectures", PPL 2011

Dynamic Load Balancing



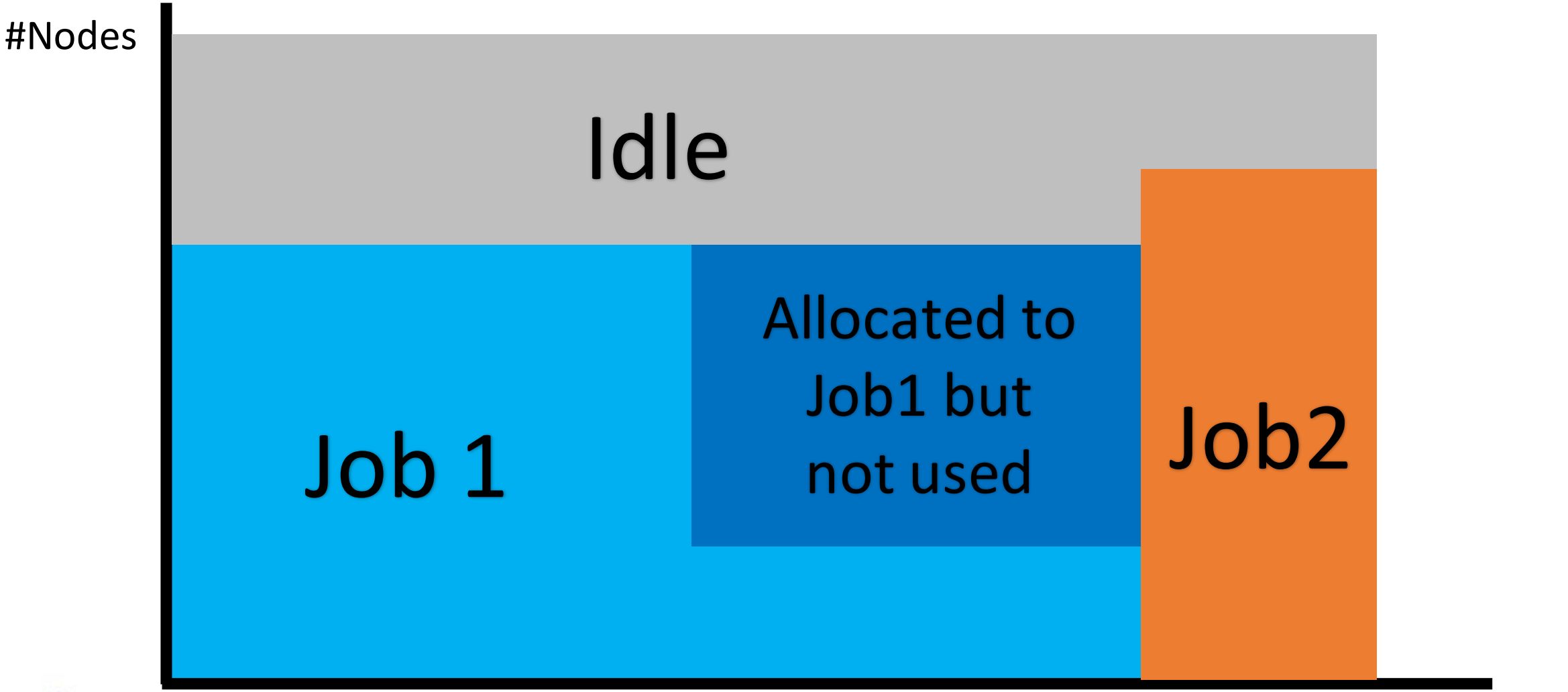
a) Unbalanced MPI application

b) Unbalanced MPI application balanced with DLB

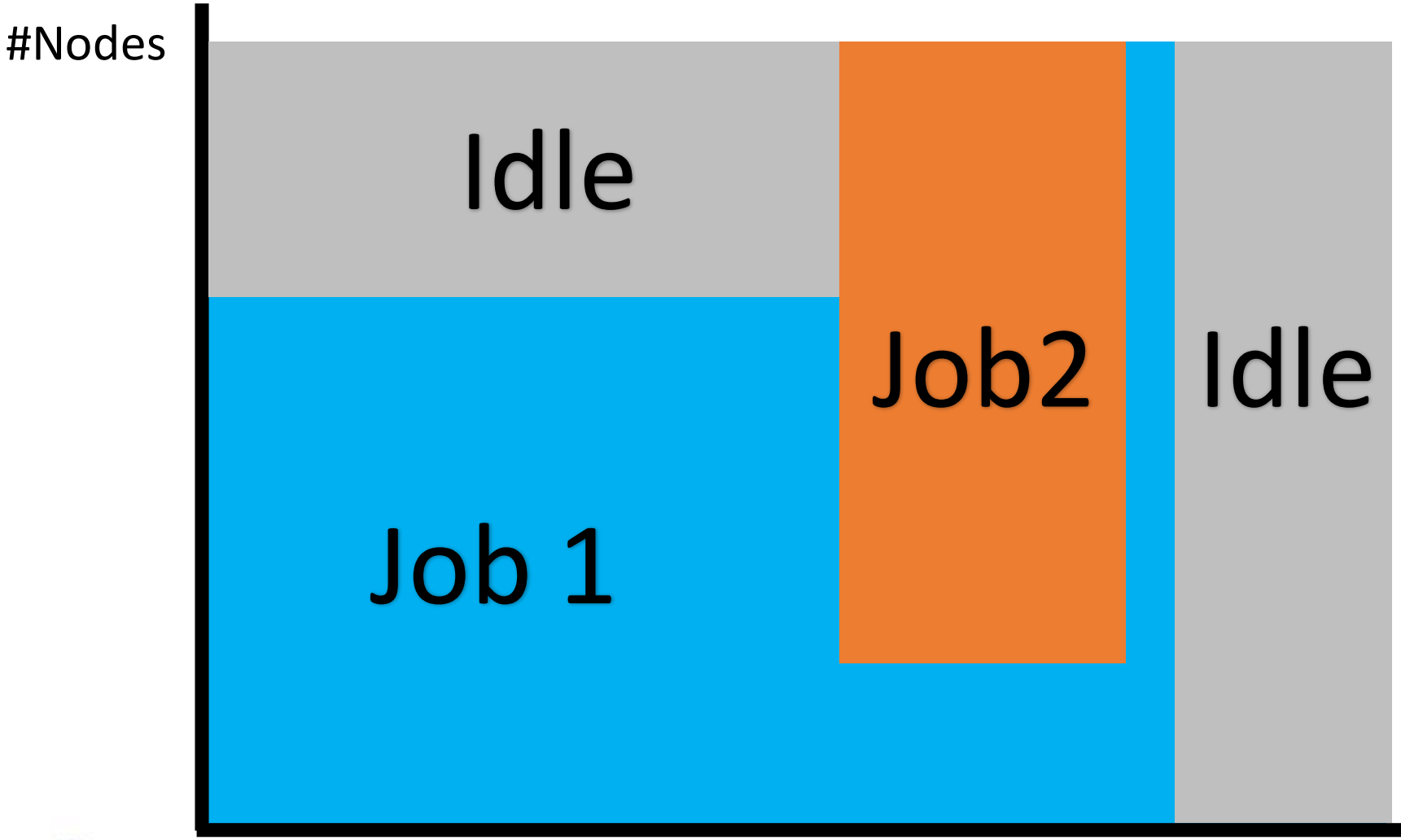
M. Garcia, J. Labarta, J. Corbalan, "Hints to improve automatic load balancing with LeWI for hybrid applications", JPDC, 2014

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

MPI Malleability



MPI Malleability



S. Iserte, R. Mayo, E. S. Quintana-Orti, V. Beltran, A. J. Peña, "DMR API: Improving cluster productivity by turning applications into malleable", Parallel Computing, 2018

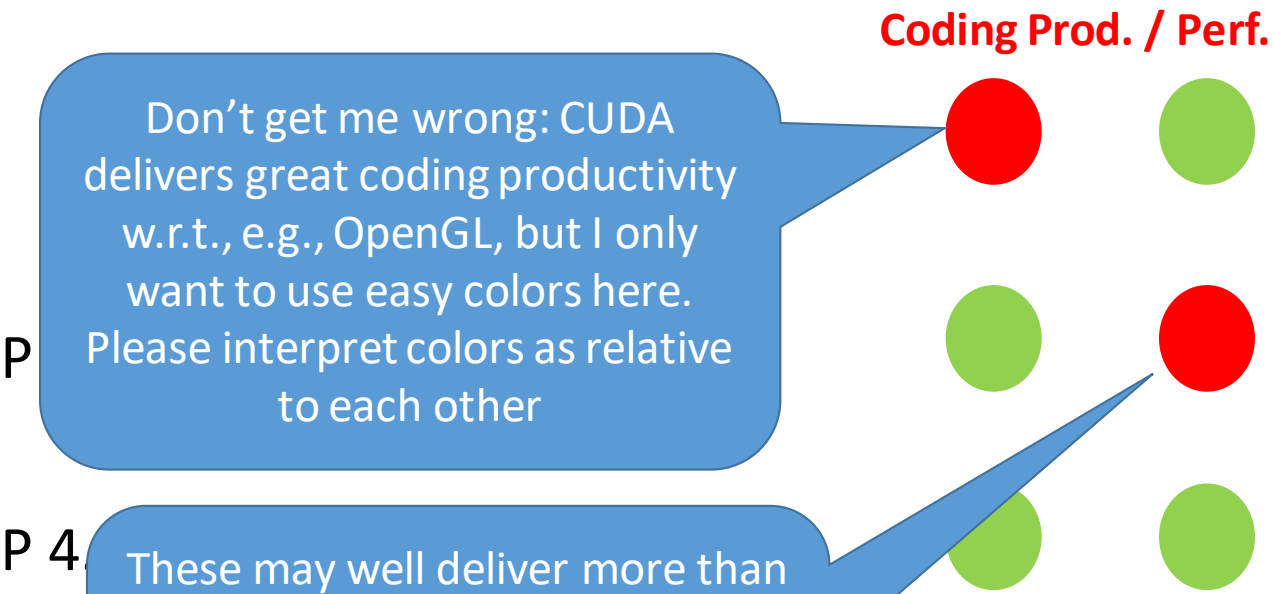
S. Iserte, R. Mayo, E. S. Quintana-Orti, A. J. Peña, "DMRlib: Easy-coding and efficient resource management for job malleability", IEEE Trans. Comput., 2021

Heterogeneous Processors: Accelerators

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

GPU Programming

- CUDA
- OpenACC / OpenMP
- OpenACC / OpenMP 4.



Others: OpenCL, SYCL…

OmpSs@CUDA

1 Port kernel to CUDA 2 Annotate device (cuda) 3 Complete device (smp)

main.c

```
#include <kernel.h>

int main(int argc, char *argv[])
{
   float a=5, x[N], y[N];

   // Initialize values
   for (int i=0; i<N; ++i)
      x[i] = y[i] = i;

   // Compute saxpy algorithm (1 task)
   saxpy(N, a, x,  y);
   #pragma oss taskwait

   //Check results
   for (int i=0; i<N; ++i){
      if (y[i]!=a*i+i) perror("Error\n")
   }

   message("Results are correct\n");
}
```

kernel.h

```
3
#pragma oss task in([n]x) inout([n]y) device(smp) copy_deps
void saxpy(int n, float a, float* x, float* y);
```

kernel.c

```
void saxpy(int n, float a, float *X, float *Y)
{
   for (int i=0; i<n; ++i)
      Y[i] = X[i] * a + Y[i];
}
```

2 implementations for the runtime to choose

kernel.cuh

```
2
#pragma oss task device(cuda) copy_deps ndrange(1,n,128) \
                 in([n]x) inout([n]y)
__global__ void saxpy(int n, float a, float* x, float* y);
```

kernel.cu

```
1
__global__ void saxpy(int n, float a, float* x, float* y)
{
   int i = blockIdx.x * blockDim.x + threadIdx.x;
   if(i < n) y[i] = a * x[i] + y[i];
}
```

OmpSs@OpenACC

```
#pragma oss task device(openacc) in(rho, sxptr, syptr, szptr) inout(vptr)
#pragma acc parallel loop deviceptr(rho, sxptr, syptr, szptr, vptr)
for (int y=ny0; y < nyf; y++) {
   for (int x=nx0; x < nxf; x++) {
      for (int z=nz0; z < nzf; z++) {
         …code…
      }
   }
}
```

EPEEC PRODUCTIVITY AT EXASCALE

https://epeec-project.eu

Benefits:

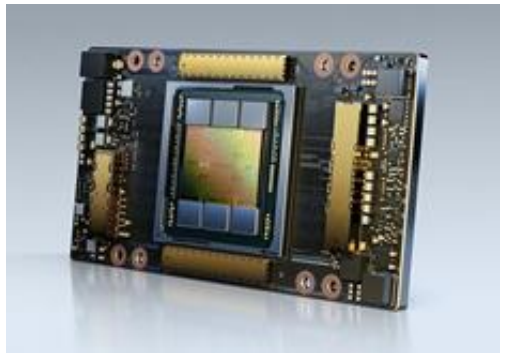
- Portability
- Programming productivity
- Automatic asynchronicity
- Automatic multi-GPU

O. Korakitis, S. García de Gonzalo, N. Guidotti, J. Barreto, J. Monteiro, and A. J. Peña, "OmpSs-2 and OpenACC interoperation", WACCPD 2022

ODOS: Democratizing Network Co-processors (a.k.a. SmartNICs/DPUs)

```
int main ()
{
        /*************/
        /* exec in host */
         /*************/

        #pragma omp target device (  )
        {
                /***************/
                /* exec in target */
                /***************/
        }
}
```

GPU

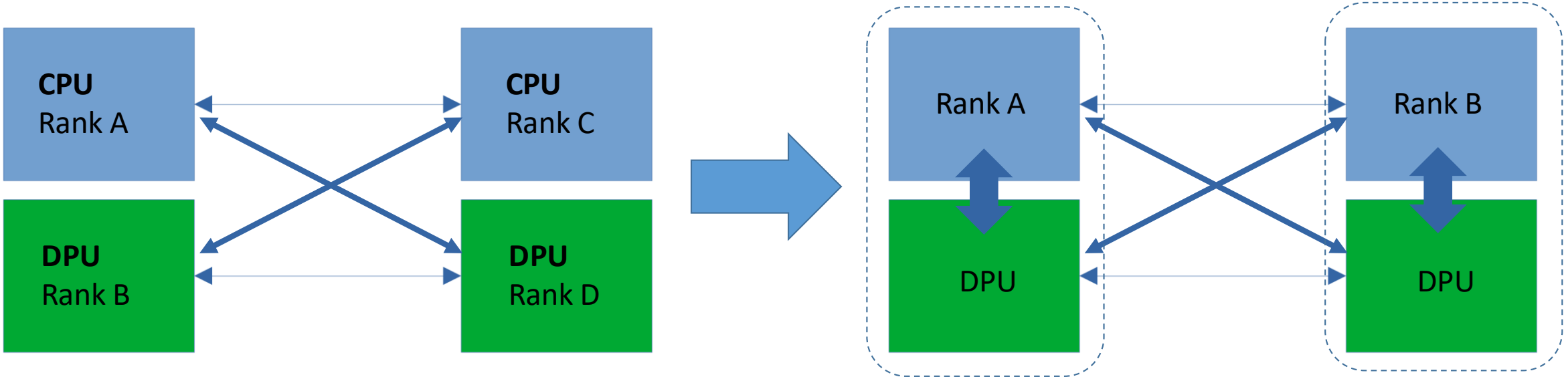


DPU

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

ODOS: Democratizing Network Co-processors (a.k.a. SmartNICs/DPUs)

- Currently CPU and DPU feature different MPI Rank numbers
  - You may have to use an MPI intercommunicator (connect/accept) or MPMD (mpirun's ":" syntax)



M. Usman *et al.*, "DPU offloading programming with the OpenMP API", in LLVM-HPC SC23 Workshop

```c
void main( int argc, char* argv[] ) {
    // Data initialization
    MPI_Init( &argc, &argv );
    MPI_Comm_rank( MPI_COMM_WORLD, &dpurank );
    if ( dpurank )      // DPU rank receives data
        MPI_Recv( &buf, SIZE, MPI_INT, 0, … );
    else                // Host rank sends data
        MPI_Send( &buf, SIZE, MPI_INT, dpurank, … );
    for( int t = 1; t <= TIMESTEPS; t++ ) {
        if ( dpurank );         // Offloaded compute
        else;                   // Host compute
    }
    if ( dpurank )      // DPU rank send data
        MPI_Send( &buf, SIZE, MPI_INT, 0, … );
    else                // Host rank receive data
        MPI_Recv( &buf, SIZE, MPI_INT, dpurank, … );
    MPI_Finalize();
}
```

```c
void main( int argc, char* argv[] ) {
    // Data initialization
    #pragma omp target
                data map(tofrom:buf[0:SIZE]))
    {
        for( int t = 1; t <= TIMESTEPS; t++ ) {
            #pragma omp target
            {
                // Offloaded compute
            }
            // Host compute
        }
    }
}
```

JACC: Just-in-Time OpenACC

AUTOMATIC...
- Asynchronous execution
- On-the-fly kernel optimization
- Multi-GPU utilization

K. Matsumura, S. Garcia de Gonzalo, A. J. Peña, "JACC: An OpenACC runtime framework with kernel-level and multi-GPU parallelization", HiPC 2021

Input Code

```
#pragma acc data copyout(x[0:N])
#pragma acc parallel loop
for (int i=0; i<N; i++)
  x[i]=y[i] * y[i];
```

① Routine Use

JACC Code

```
/* Entry of #pragma acc data */
jacc_create(x, N * sizeof (float));

/* #pragma acc parallel loop */
jacc_kernel_push(
  "#pragma acc parallel present (x, y)\n"
  "#pragma acc loop\n"
  "for (int i=0; i<N; i++) /* ... */",
  /* args */ , /* flags */ );

/* Exit of #pragma acc data */
jacc_copyout(x, N * sizeof(float));
```

② Exec

Dynamic Code

```
void kernel0(f
             float *y, size_t N) {
#pragma acc parallel present(x, y)
#pragma acc loop
for (int i=0; i<N; i++)
    x[i]=y[i] * y[i];
}
```

③ Generate Code

④ Compile (NVHPC/GCC)
Link
Caching Binary
Execution

JACC Runtime

Data Management
- Array Mapping
- CPU-to/from-Device Communication

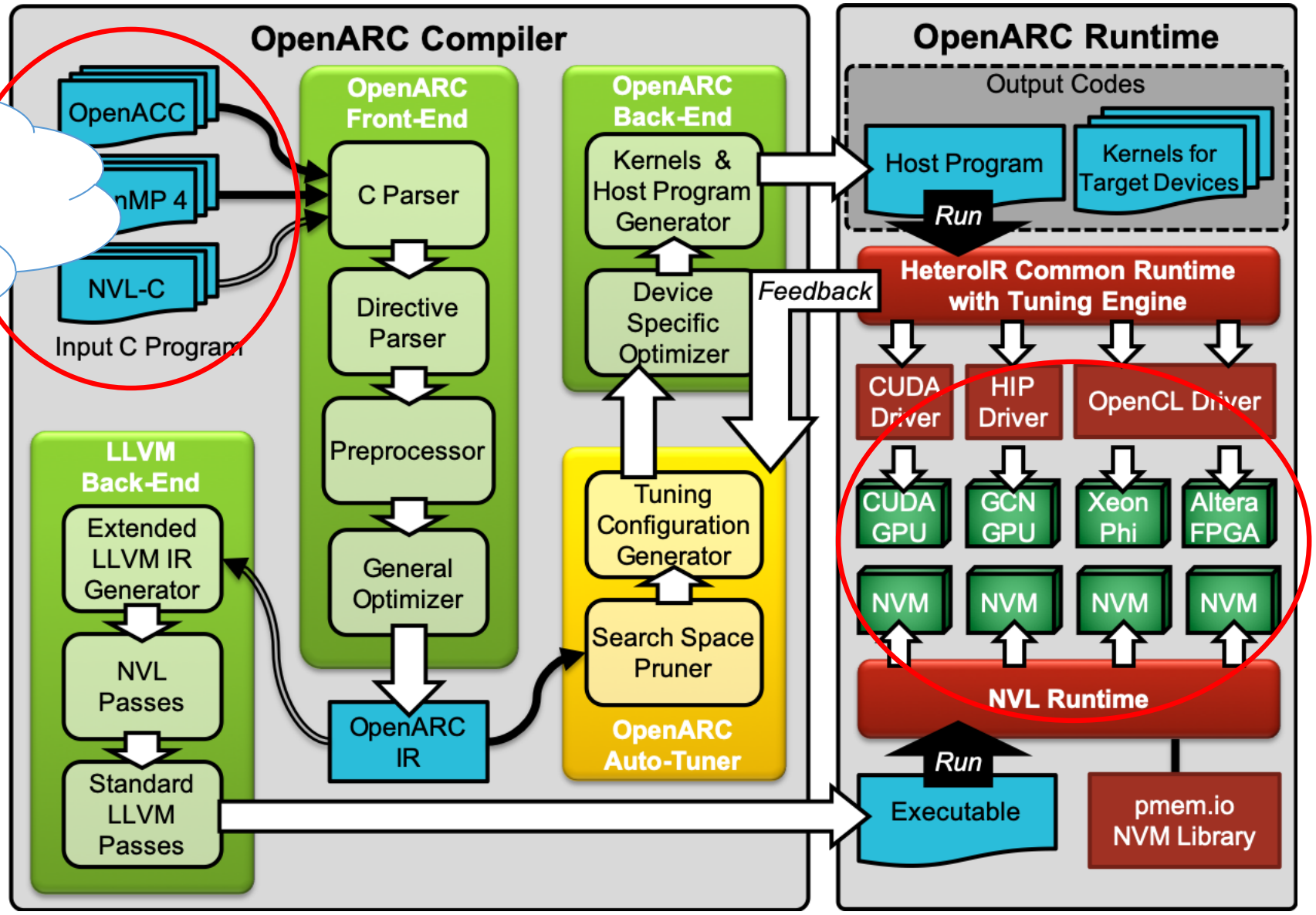
Kernel Execution
- Code Gen/Compile
- Parameter Calculation
- Extended Execution

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

https://www.bsc.es/discover-bsc/organisation/scientific-structure/accelerators-and-communications-hpc

Multi-device: OpenARC

OmpSs@OpenACC + OpenARC?



S. Lee, J. S. Vetter, "OpenARC: Open accelerator research compiler for directive-based, efficient heterogeneous computing", HPDC 2014

S. Lee, J. S. Vetter, "OpenARC: Extensible OpenACC compiler framework for directive-based accelerator programming study", WACCPD 2014

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Source: https://csmd.ornl.gov/project/openarc-open-accelerator-research-compiler

Trendy: C++

- No API calls
- Single source file
- Heavy compiler work
- Heavily based in newest C++ standards (C++11/C++17)
  - Memory model
  - Concurrent execution model
  - Concurrency library
  - Execution policies
- Inherently supports multi-device + heterogeneous devices
- E.g.:
  - SYCL / OneAPI – DPC++
  - NVC++

    nvc++ -stdpar program.cpp –o program

```
#include <CL/sycl.hpp>
#include <iostream>

constexpr int num=16;
using namespace sycl;

int main() {
 auto r = range{num};
 buffer<int> a{r};

 queue{}.submit([&](handler& h) {
  accessor out{a, h};
  h.parallel_for(r, [=](item<1> idx) {
   out[idx] = idx;
  });
 });

 host_accessor result{a};
 for (int i=0; i<num; ++i)
  std::cout << result[i] << "\n";
}
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Code Source: https://www.intel.com/content/www/us/en/develop/documentation/oneapi-programming-guide/top/oneapi-programming-model/data-parallel-c-dpc.html

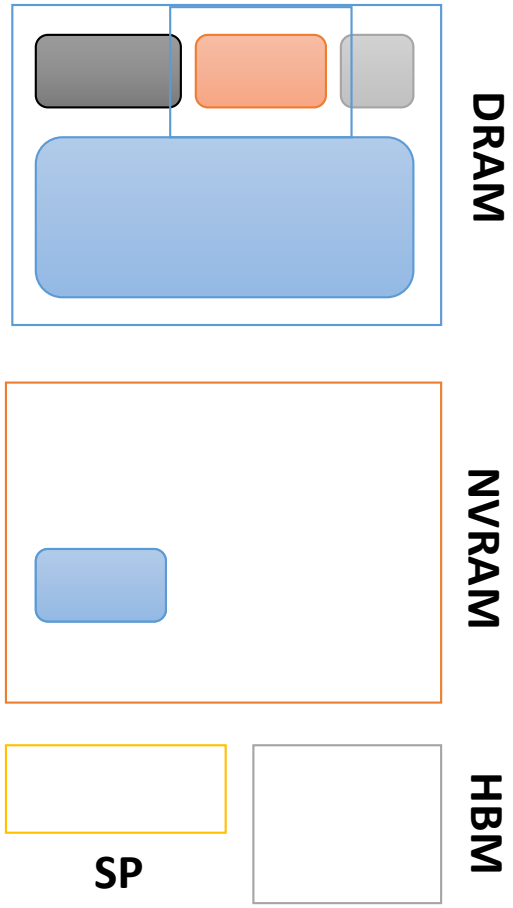
Heterogeneous Memory Systems

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Heterogeneous Memory Systems

- Heterogeneous memory systems
  - KNL: DRAM + MCDRAM (↑ BW, ↑ Lat.) → R.I.P.
  - CPUs + HBM (e.g., Sapphire Rapids)
  - Byte-addressable NVRAM / Persistent Memory (PMEM)
    - HP's The Machine (R.I.P.?)
    - Intel's Optane (3D XPoint)
  - Also GPUs (and more with UVM)

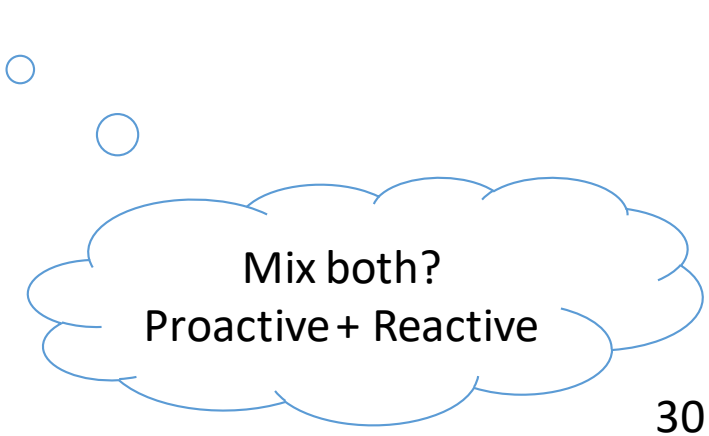
- Goal
  - Assess optimal data distribution
    - Maximize performance
    - Minimize energy
    - …



Heterogeneous Memory Systems

- Deep memory != Heterogeneous Memory
  - Rationale: “deep” implies hierarchy (i.e., cache-like)
  - (own view – many authors don’t follow this distinction)
  - In some cases deep memory works well (high locality)
- Methodologies
  - Page level (reactive)
    - Leverages OS’s view
    - Can monitor hot vs. cold pages, # of allocations, total size, global status
    - Easy migrations
  - Object granularity (proactive)
    - (object: variable, static array, heap buffer, etc.)
    - Leverage object semantics
    - Usually same access pattern across entire object
    - User-friendly – user may hint / control

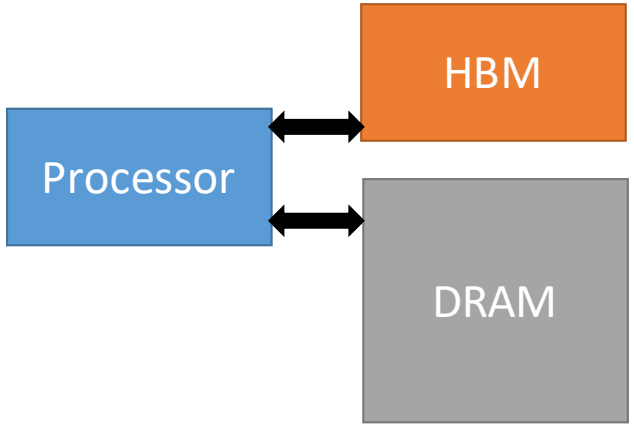
M. Marques, I. Kuzmin, J. Barreto, J. Monteiro, R. Rodrigues, “Dynamic Page Placement on Real Persistent Memory Systems”, arXiv:2112.12685, 2021



Heterogeneous Memory Systems

- Intel Xeon Max with HBM + DRAM

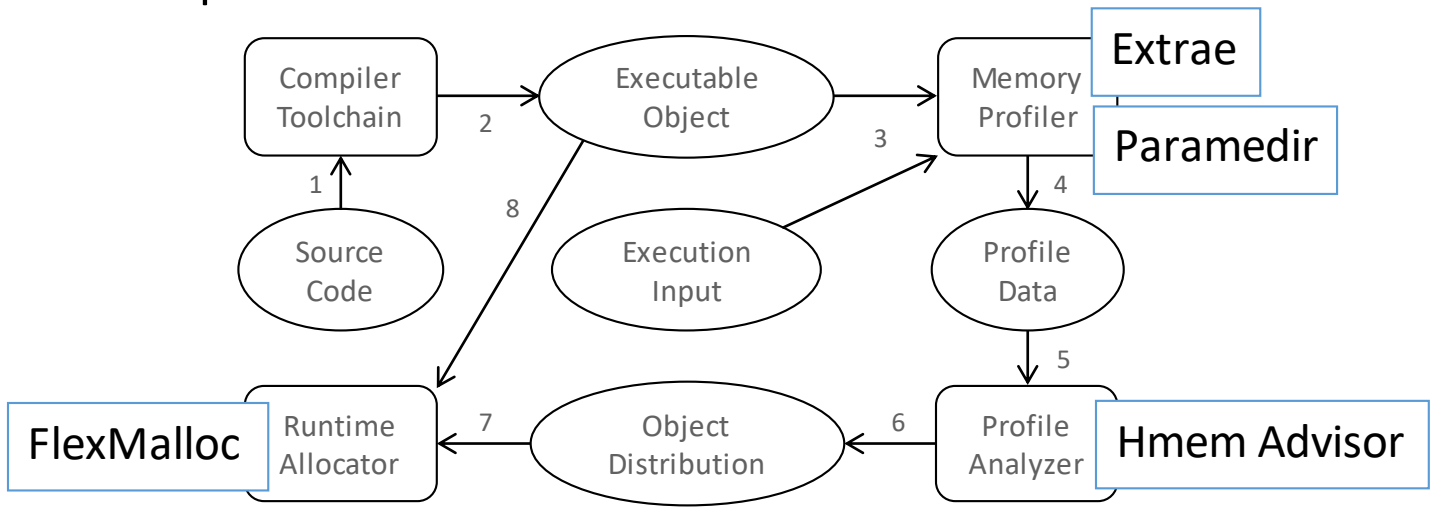
- Heter. Memory (a.k.a. *flat mode* or *app. direct*)
  - DRAM and HBM are both available
  - More overall memory available
  - Software managed (applications need to handle themselves)



- Deep Memory (a.k.a. *cache mode* or *memory mode*)
  - HBM as cache for DRAM
  - Only DRAM address space
  - Done in hardware (applications don't need to be modified)

Heterogeneous Memory Systems

- Data-oriented profiling
  - Instrumenting-based solutions (slow)
    - MemSpy, SLO, MACPO, Intel Advisor, **EVOP**
  - HW-based solutions (lossy)
    - Sun ONE Studio, HPCToolkit, Intel Vtune Amplifier, MemAxes, **Extrae**
- These give data access information to developers
  - Anything more automated possible?

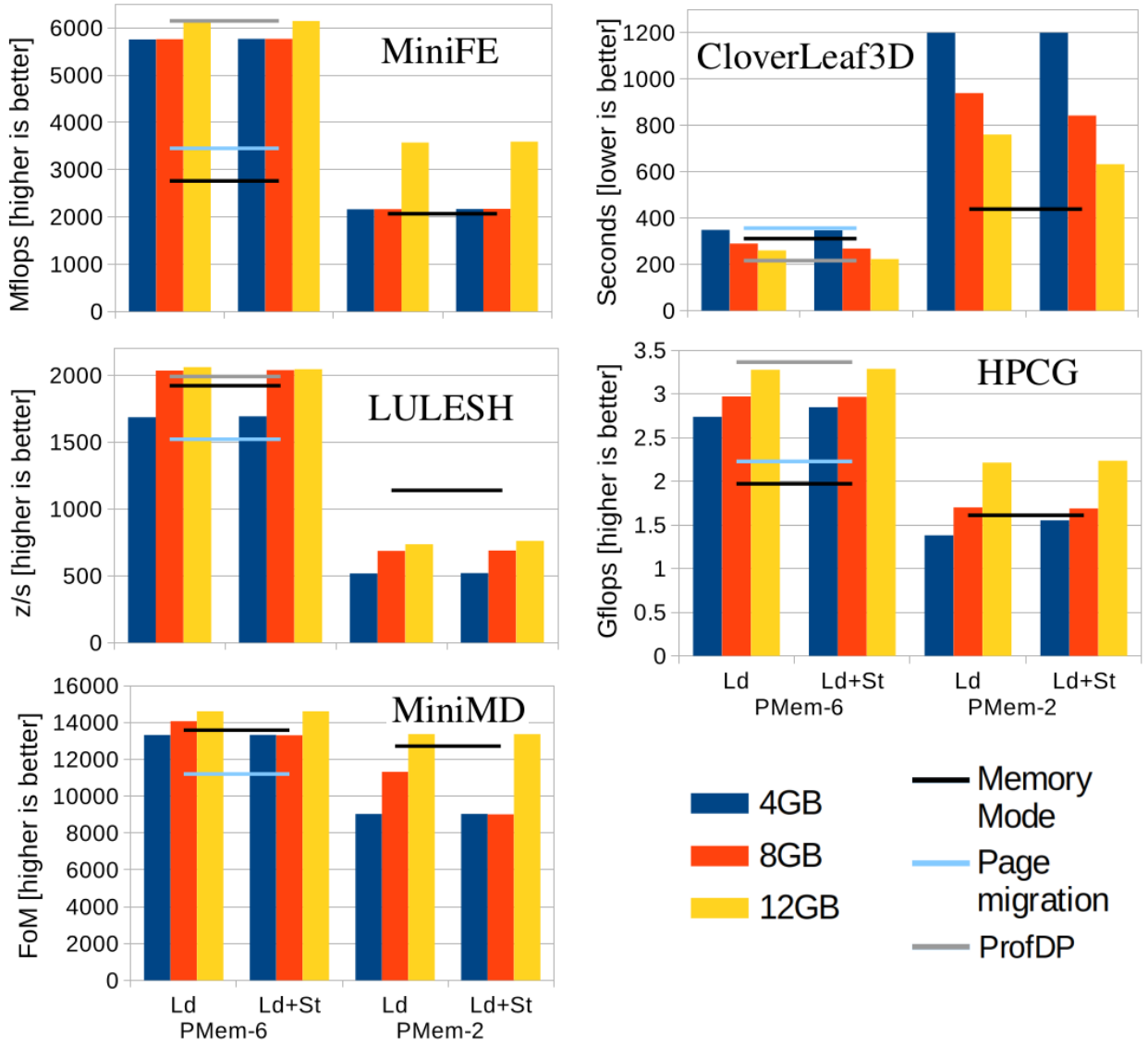


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Evolved version of: A. J. Peña, P. Balaji, "Toward the efficient use of multiple explicitly managed memory subsystems," IEEE Cluster 2014

Some Results on Optane PMEM

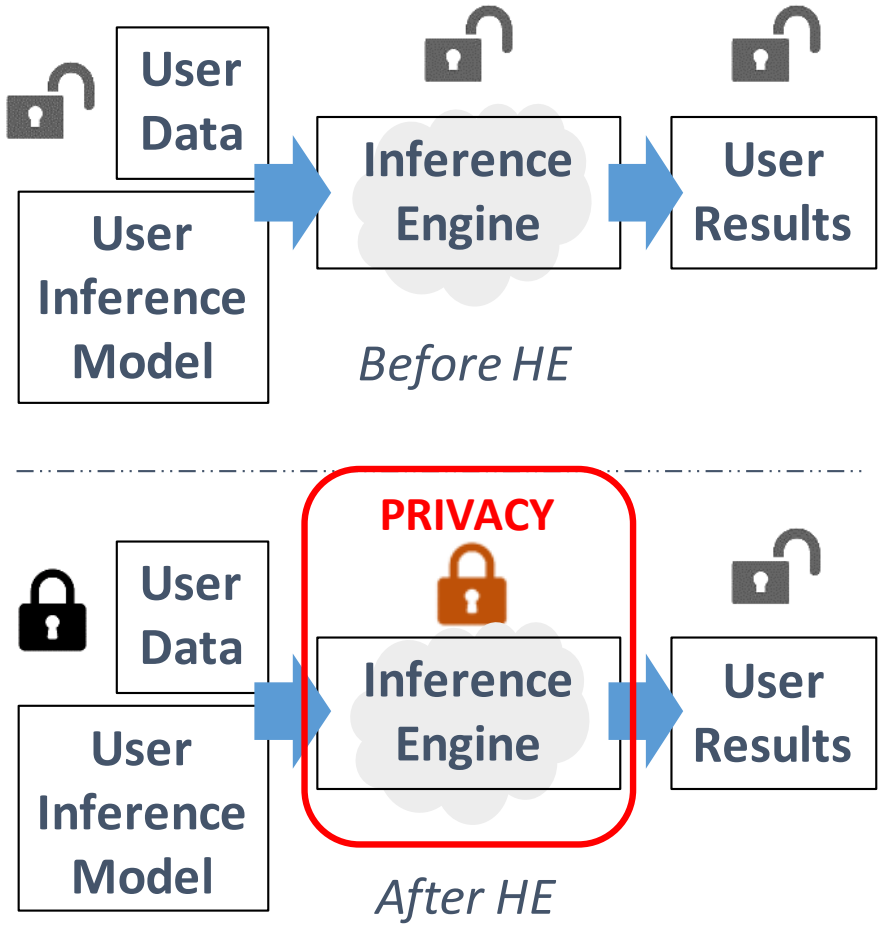
- Comparison with memory-mode and kernel page migration (KMP)
- ecoHMEM outperforms MM and KPM
  - Up to 2x in MiniFE
  - In some of them even using 4x less DRAM
- Full apps:
  - LAMMPS: Similar performance
  - OpenFOAM: ~6% speedup
  - **BSIT: ~2x w.r.t. I/0**
- Performance in pair with SotA ProfDP
- **Sapphire Rapids (HBM):**
  - Seamless use of HBM with >10% gains wrt DRAM



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

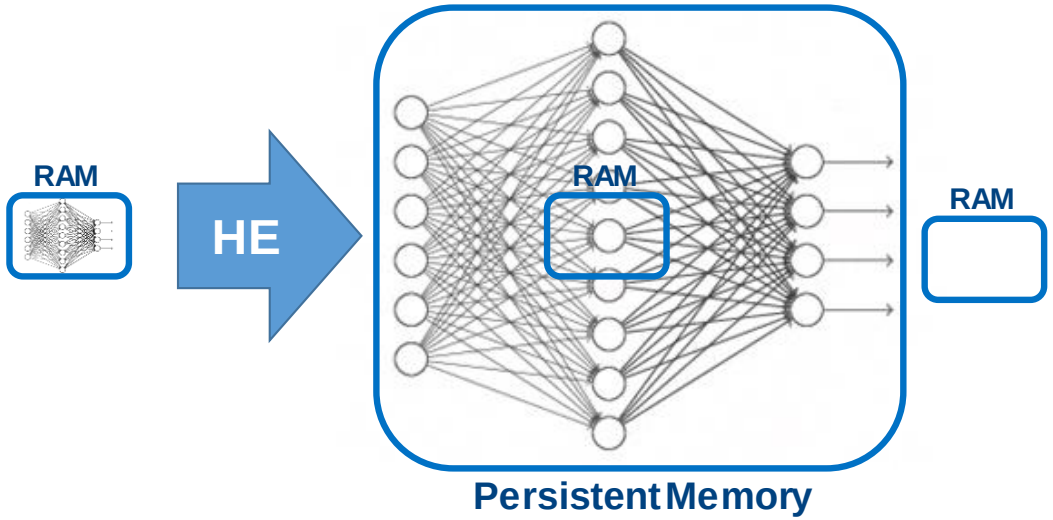
Jordà *et al.*, "ecoHMEM: Improving object placement methodology for hybrid memory systems in HPC", in IEEE Cluster 2022 – Best Paper Finalist

Homomorphic Encryption (HE) & DL



x

100x
10,000x



HomE enabling private DL

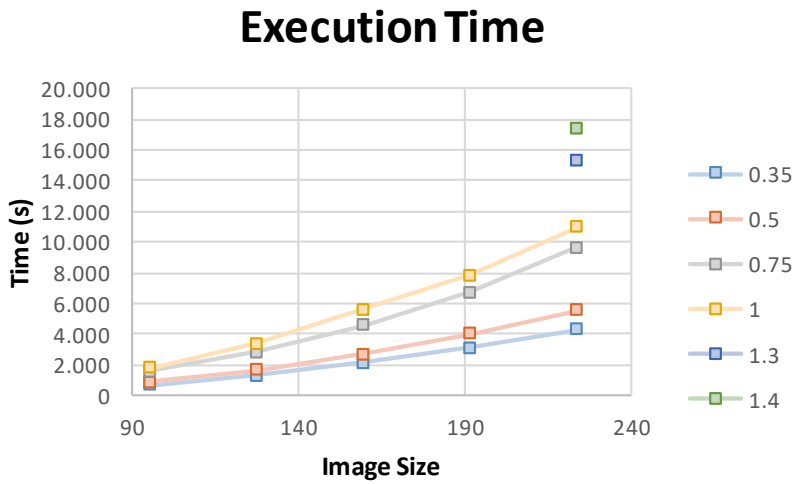
A. J. Peña, ERC Consolidator Grant "HomE", 2022-2027

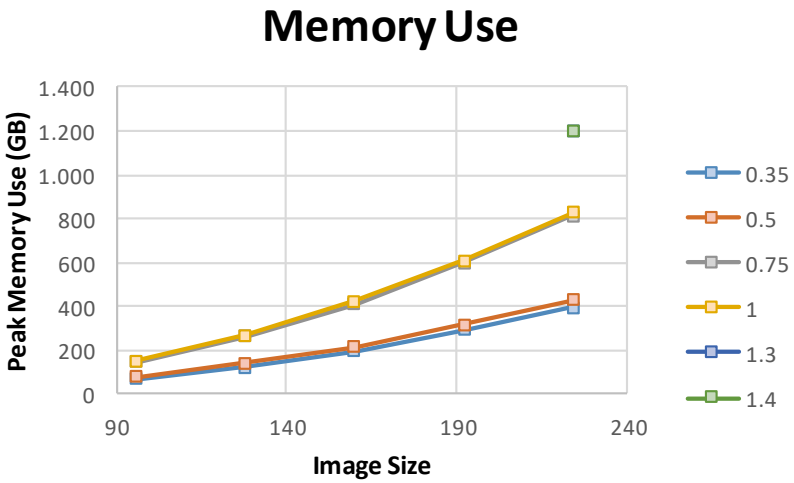
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

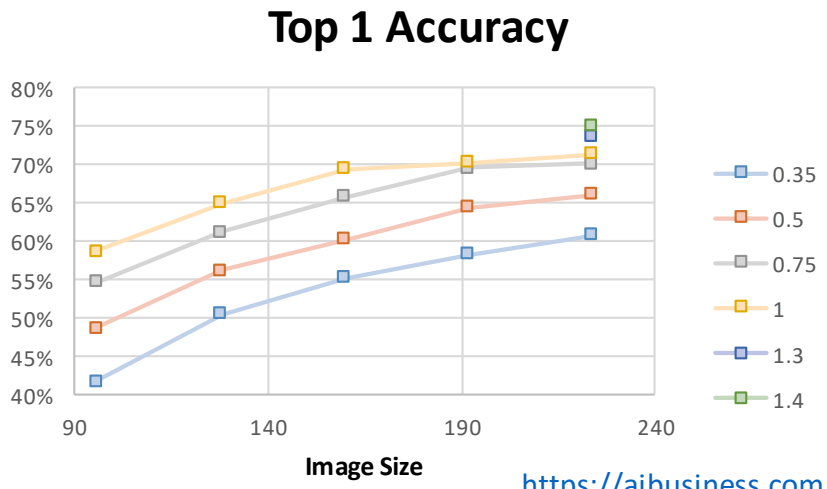
https://www.hpcwire.com/off-the-wire/bsc-researcher-antonio-j-pena-awarded-erc-consolidator-grant-for-the-home-project

HW-Based Memory Mode (DEEP-MEMORY) Works Great

- We have run the largest (plaintext) model to date using HE input data
- Biggest to date was MobileNetV2 at expansion factor of 0.35
- Limited by DRAM sizes
- We run MobileNetV2 at max. expansion factor (x3 previous size)
- Also ResNet-50, 1st non-toy DNN
- Thanks to Intel Optane DIMMs
- **Memory mode yields high efficiency**







Barcelona Supercomputing Center
Centro Nacional de

G. Lloret-Talavera *et al.*, "Enabling homomorphically encrypted inference for large DNN models", IEEE Trans. Comput., 2022

https://aibusiness.com/ml/intel-and-barcelona-supercomputing-center-succeed-in-encrypting-large-neural-networks

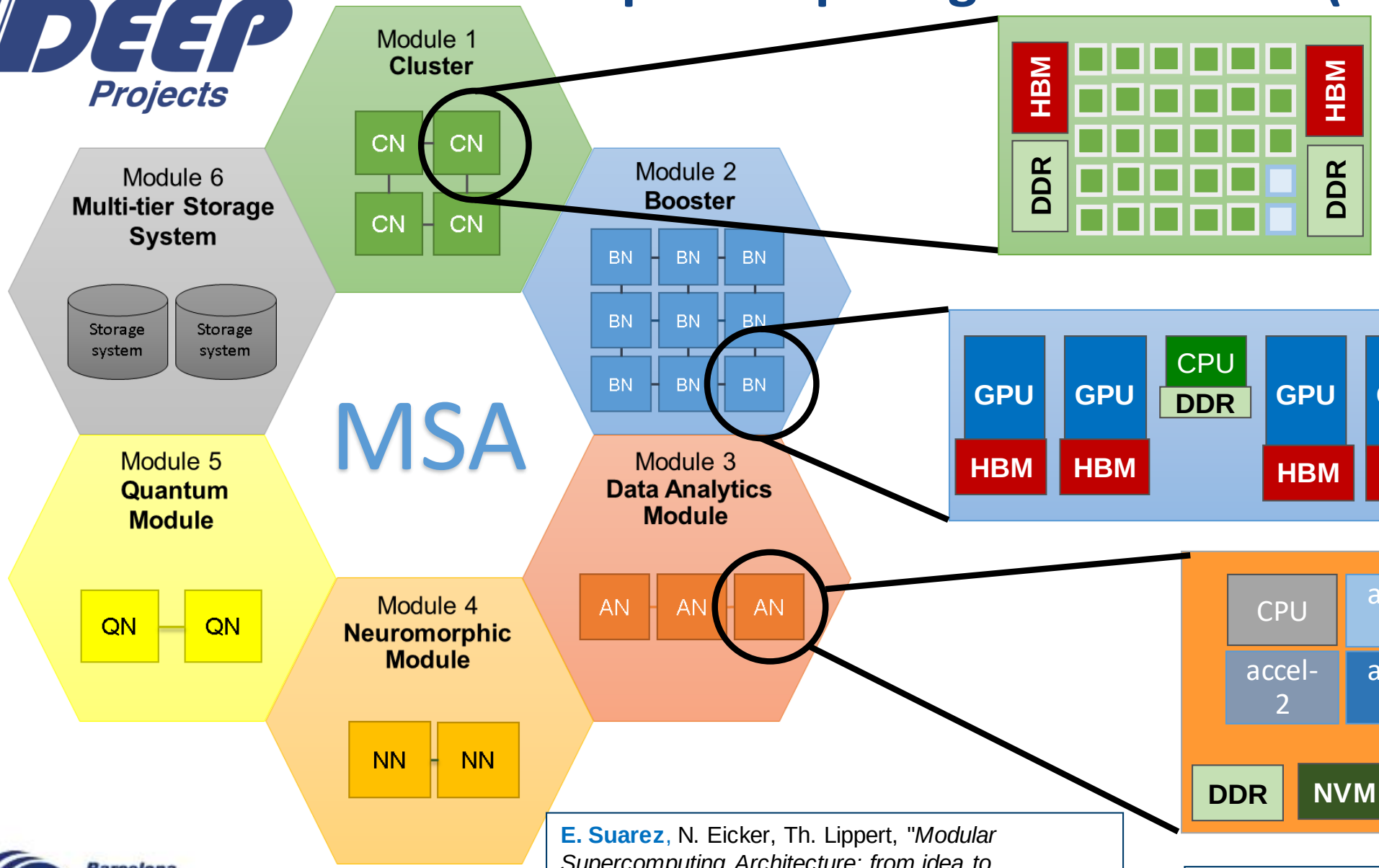
Distributed Heterogeneity

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Heterogeneous Distributed Devices

- Increasing heterogeneity at compute node level (e.g., TOP500):
  - BSC MareNostrum 5:
    - ACC (#8): Intel Sapphire Rapids + NVIDIA Hopper
    - GPP (#19): Intel Sapphire Rapids + some nodes with HBM+DRAM
  - NERSC Perlmutter (#12): CPU nodes + GPU-accelerated nodes + Workflow nodes + high-memory
  - JÜLICH JEWELS:
    - CLUSTER Module: 2,271 CPU + 240 large memory + 56 GPU V100
    - BOOSTER Module (#18): 936 GPU A100
  - CSCS Piz Daint (#37): 5,704 nodes w/GPUs + 1,813 nodes just CPU
  - DOE Trinity (#38): 9,436 Xeon Nodes + 9,984 KNL Nodes
  - DOE Cori (#60): 2,388 Xeon Nodes + 9,688 KNL Nodes
- In practice
  - People use a set of homogeneous nodes (i.e., "partition") → Like separate clusters
  - Considering those alltogether good for TOP500, though ☺
- A few not-(yet?)-so-commonly-used SW solutions...

Modular Supercomputing Architecture (MSA)



EPI node

GPU-centric node

Highly het. node

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

E. Suarez, N. Eicker, Th. Lippert, "*Modular Supercomputing Architecture: from idea to production*", Chapter 9 in Contemporary High Performance Computing: from Petascale toward Exascale, Volume 3, p 223-251, CRC Press. (2019)

E. Suarez, N. Eicker, and Th. Lippert, "Supercomputer Evolution at JSC", Proceedings of the 2018 NIC Symposium, Vol.49, p.1-12, (2018)

Global MPI for MSA

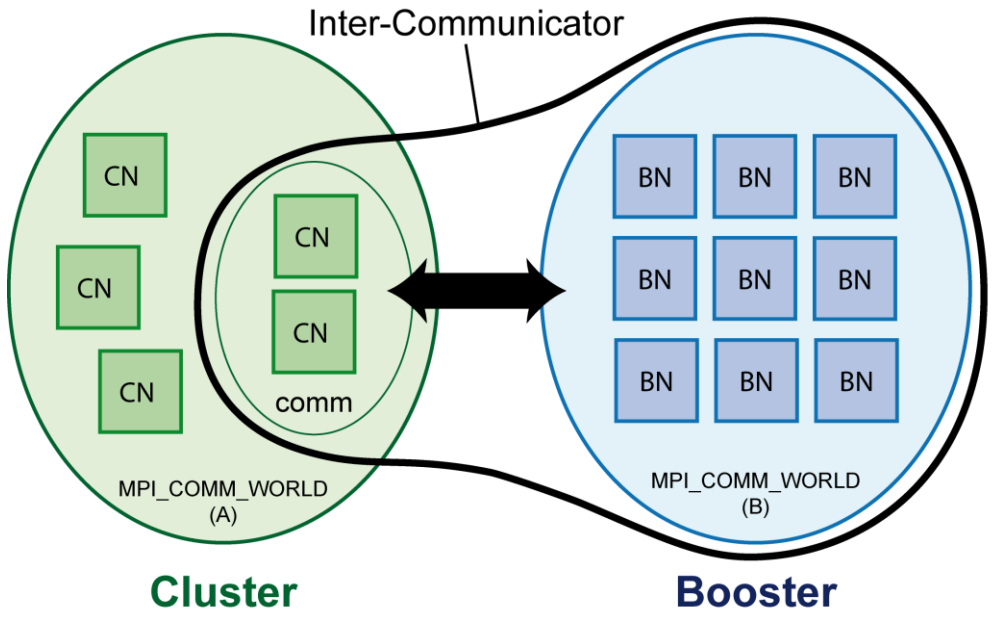
E. Suarez, "Modular Supercomputing: a system-wide orchestration of heterogeneous resources", ADAC 2022

ParaStation MPI

- An MPI application can run:
  - Using only Cluster nodes
  - Using only Booster nodes
  - Distributed over Cluster and Booster
    - *In this case two executables are created*
    - *Collective offload process*
    - *Transparent data exchange via MPI*

- One can also start two parts of a code and connect them via **`MPI_Connect()`**
- Or have one single common **`MPI_COMM_WORLD`** and split it into subcommunicators via **`MPI_Comm_Split()`**

- ParaStation Global MPI
  - Uses **`MPI_Comm_spawn()`**
    - *Collective spawn groups of processes from Cluster to Booster (or vice-versa)*
  - Inter-communicator
    - *Connects the 2 MPI_COMM_WORLD*
    - *Contains all parents on one side and all children on the other*
      - *Returned by MPI_Comm_spawn for the parents*
      - *Returned by MPI_Get_parent by the children*

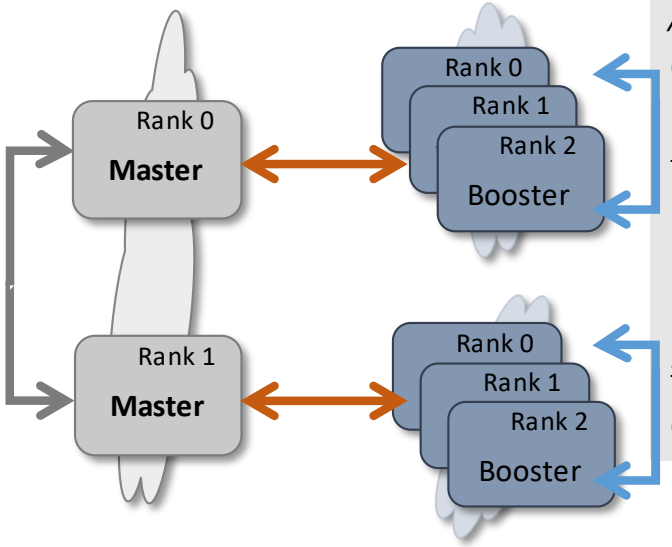



```
salloc --partition=cluster -N 4
       : --partition=booster -N 12
srun --het-group=1 -N 4 -n 8
       ./app_booster
```

Clauss et al., "Dynamic Process Management with Allocation-internal Co-Scheduling towards Interactive Supercomputing", COSH@HiPEAC, (2016)

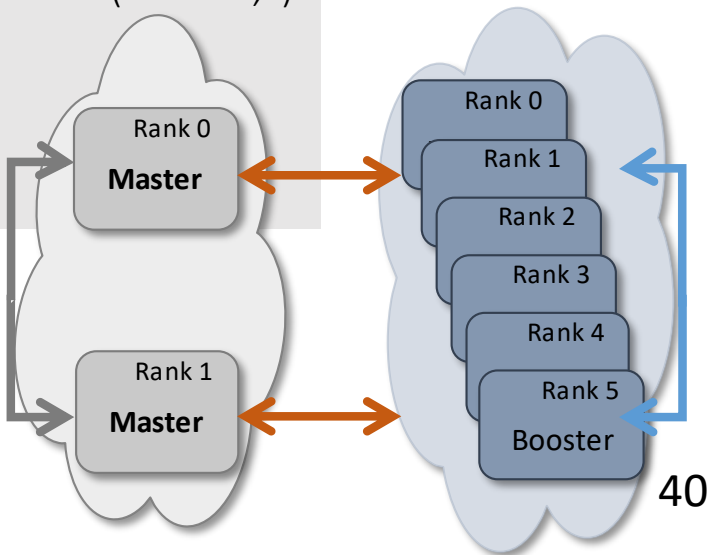
Intel & OmpSs Offload Semantics

- Let offload work to remote compute node
- Enables leveraging good architecture-algorithm matching
- Not transparent: developers have to select target nodes & offload tasks
- Great for mixed CPU + accelerator clusters (e.g., MSA)




```
// Allocates 'ppn ' processes in each of 'n' hosts
deep_booster_alloc(MPI_COMM_WORLD , n,
                                  ppn , &worker);
for (int i=0; i<n*ppn; i++) {
        #pragma omp task onto (worker , i)
        offloaded_task();
}
#pragma omp taskwait
deep_booster_free (worker);
```

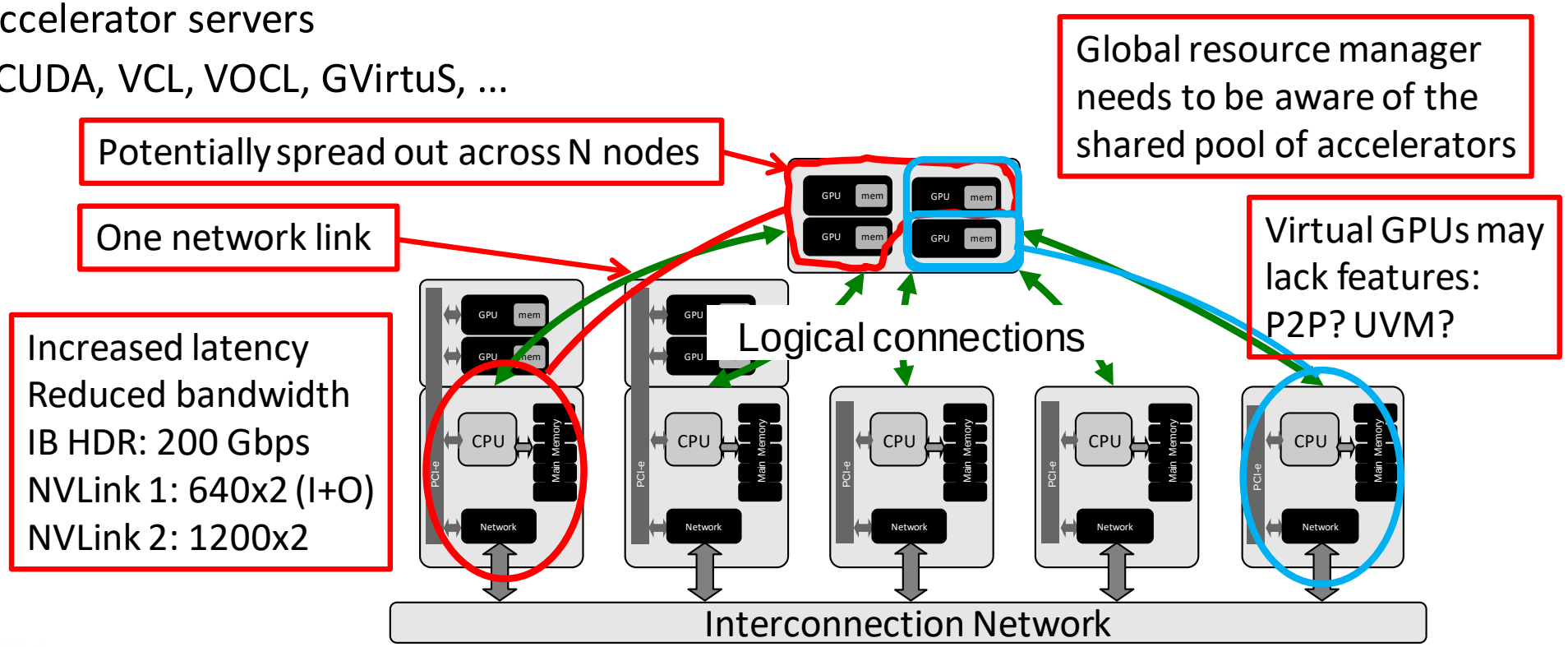
A. J. Peña, V. Beltran, C. Clauss, T. Moschny, "Supporting automatic recovery in offloaded distributed programming models through MPI-3 techniques", ICS 2017



F. Sainz *et al*, "Collective offload for heterogeneous clusters", *HiPC* 2015

Heterogeneous Distributed Devices

- Virtualized remote accelerators:
  - Transparent* to the user – no programming model extensions
  - Runtime system intercepts accelerator API (CUDA/OpenCL) and forwards commands to accelerator servers
  - rCUDA, VCL, VOCL, GVirtuS, …

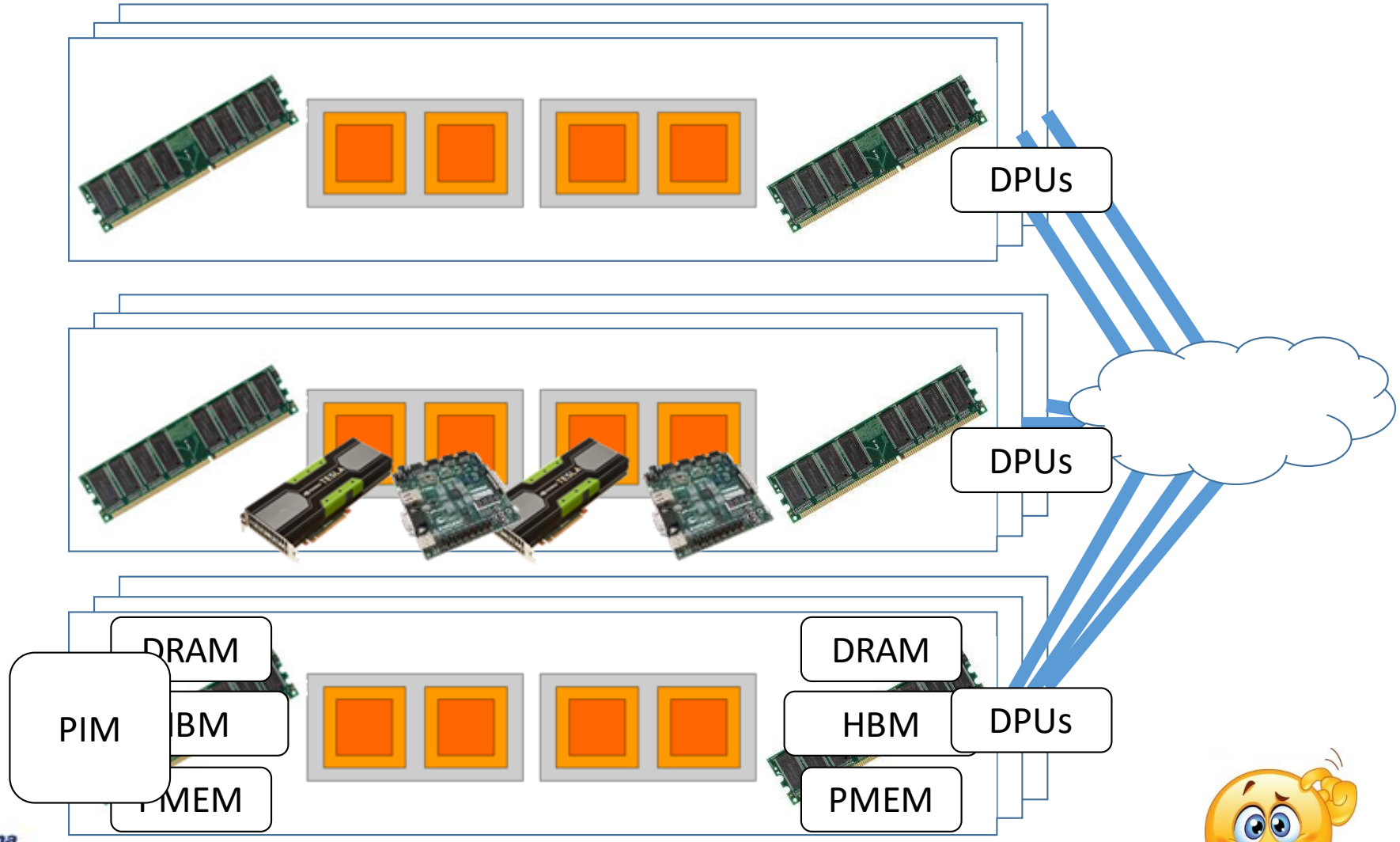


A. J. Peña, C. Reaño, F. Silla, R. Mayo, E. S. Quintana-Ortí, J. Duato, "A complete and efficient CUDA-sharing solution for HPC clusters", Parallel Computing, 2014

Putting it All Together?

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Plenty of Q's, no Perfect A's – Up to Us to Have Fun!



Conclusion

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Summary

- Heterogeneity is here for good
- Much more than hybrid CPU + GPU…
  - Processing elements everywhere (memory, network)
  - Also memory and distributed devices
- Programming models, runtimes, OS, languages… here to help
  - But no one-fits-all
    - Combinations not always straightforward
  - Ease programmers' life but no free lunch
  - Still a lot to improve to fully banish ninja programmers

AccelCom's Current Members

- Sergio Iserte, Sr. Researcher
- Sergi Rovira, Postdoc

- Mariano Benito, Sr. Postdoc
- Michele Esposito, Postdoc

- Thomas Spendlhofer, Postdoc
- Tathagata Barik, PhD Student

- Marc Jordà, Sr. Research Engineer
- Lena Martens, Sr. Research Engineer

- Muhammad Usman, Research Engineer
- Priyam Gupta, Research Engineer

- Zaid Mughal, Research Engineer
- Petter Sandas, Jr. Research Engineer

- **Looking for more PhD Students…**

- External Collaborators:
  - Adrián Castelló (Universitat Politècnica de València)
  - Kazuaki Matsumura (NVIDIA)
  - Francisco Rodríguez (Universitat Oberta de Catalunya)
  - Marc Luis Ponce Sochnov (Universitat Politècnica de Catalunya)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Funding Acknowledgements

- Intel-BSC Exascale Laboratory SoW 5.1
- European Union's H2020 Marie Sklodowska-Curie grant agreement No. 749516
- European Union's H2020 Project EPEEC under grant agreement No. 801051
- European Commission's EuroHPC project DEEP-SEA under grant agreement 955606
- European Commission's EuroHPC project EPI SGA2 under grant agreement 101036168
- The Spanish Ministerio de Ciencia e Innovación—Agencia Estatal de Investigación

This project has received funding from the European High Performance Computing Joint Undertaking (JU) under Framework Partnership Agreement No 800928 and Specific Grant Agreements No 955606 (DEEP-SEA) and No 101036168 (EPI SGA2). The JU receives support from the European Union's Horizon 2020 research and innovation programme and from Croatia, France, Germany, Greece, Italy, Netherlands, Portugal, Spain, Sweden, and Switzerland.

Funding Acknowledgements

Partially funded by the European Union (ERC, HomE, 101043467). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

HomE enabling private DL

European Research Council
Established by the European Commission

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Funding Acknowledgements

National Support

Project PCI2021-121958 funded by:

MINISTERIO DE CIENCIA E INNOVACIÓN

Financiado por la Unión Europea NextGenerationEU

Plan de Recuperación, Transformación y Resiliencia

AGENCIA ESTATAL DE INVESTIGACIÓN

Authors are grateful for the support form the Department of Research and Universities of the Generalitat de Catalunya to Research Group AccMem (Code: 2021 SGR 00807)

Generalitat de Catalunya
Departament de Recerca i Universitats

Grant RYC2020-030054-I funded by:

MINISTERIO DE CIENCIA E INNOVACIÓN

UNIÓN EUROPEA
Fondo Social Europeo
El FSE invierte en tu futuro

AGENCIA ESTATAL DE INVESTIGACIÓN

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Thank you

The Current Mess (and Power) of Heterogeneity in HPC

antonio.pena@bsc.es