DE LA RECHERCHE À L'INDUSTRIE

INVENTORS FOR THE DIGITAL WORLD

# PORTING PARALLEL APPLICATIONS TO HETEROGENEOUS SUPERCOMPUTERS: LIBRARIES AND TOOLS CAN MAKE IT TRANSPARENT

**Jean-Yves VET**, DDN Storage
Patrick CARRIBAULT, CEA
Albert COHEN, INRIA

**CATC 2016**
September,15

www.cea.fr

*CEA, DAM, DIF, F-91297 Arpajon, France*

# CONTEXT (1/2)

## *HPC AND LEGACY CODES AT THE CEA*



Since the 60s, about a new machine every 5 years…

*CDC 7600 (1976)*

*CRAY 1S (1982)*

*Bull Tera-10 (2006)*

Some legacy codes >100k lines. Maintaining and porting legacy codes is a huge amount of work.

**A strong need for**:

- Increasing portability
- Reaching decent compute efficiency (HPC)
- Porting code in a cost-efficient way (libraries or transparent mechanisms, incremental changes)

# CONTEXT (2/2)

## *BACK TO 2010: TERA 100 AND EVALUATIONS OF NEW ARCHITECTURES*

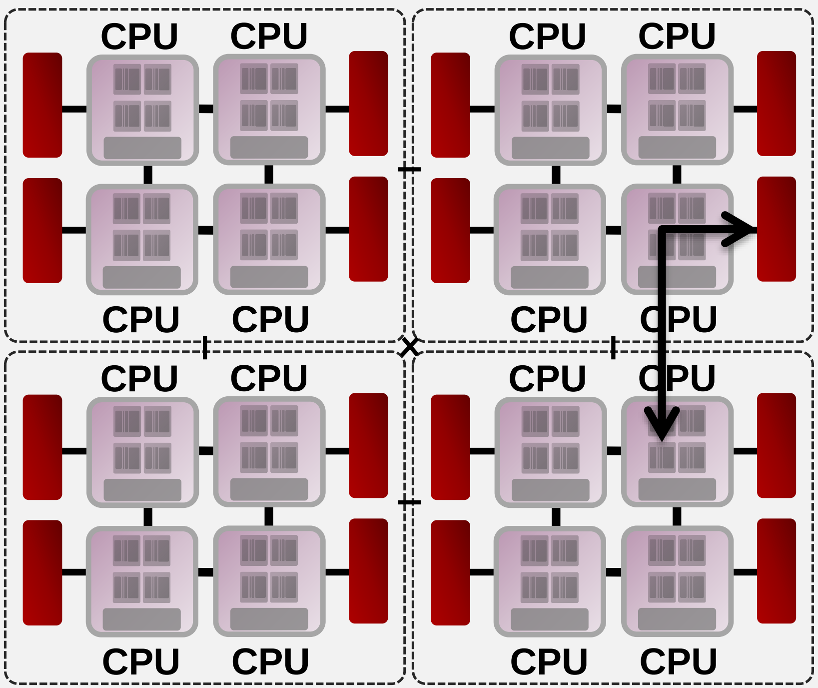


*Tera-100 (2010)*

## Tera-100 (Bull)

- 1.254 PFLOP/s (ranked 6, November 2010 TOP500)
- Mainly homogeneous (Intel Xeon)
- Codes: MPI + OpenMP

## Fat node (Bull Coherence System)



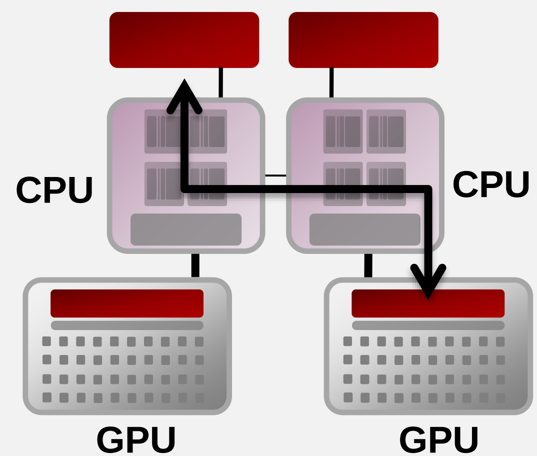

- Increased **Non-Uniform Memory Access (NUMA)** effects

- **Heterogeneous computing** (load balancing + programming models)

- **Non-hardware coherent memories** (GPU <-> CPU)

- **Non-Uniform IO Access** (NUIOA [(1)] )

## Heterogeneous node






(1) Stéphanie Moreaud, Brice Goglin, and Raymond Namyst. *Adaptive MPI multirail tuning for non-uniform input/output access*. EuroMPI'10.

# CONTRIBUTIONS

## HETEROGENEOUS LOAD BALANCING & IMPROVED DATA LOCALITY APPLIED TO LEGACY CODES

### H3LMS

*(Harnessing Hierarchy and Heterogeneity with Locality Management and Scheduling)*

- **Bulk synchronous (multi-phase with barriers) task decomposition** to deal with heterogeneity.

- **NUMA aware scheduling**: mix data centric work distribution and hierarchical work stealing.

- **Transparent coupling with MPI and OpenMP** with an implementation into a single framework.

**Common features with StarPU [(1)], XKaapi [(2)], or OmpSs [(3)]**

- Distributed Shared Memory (DSM) to handle data transfers automatically between non-hardware coherent memories in the compute node.

- Software caches to reduce memory transfers.


### COMPAS

*(Coordinate and Organize Memory Placement and Allocation for Scheduler)*

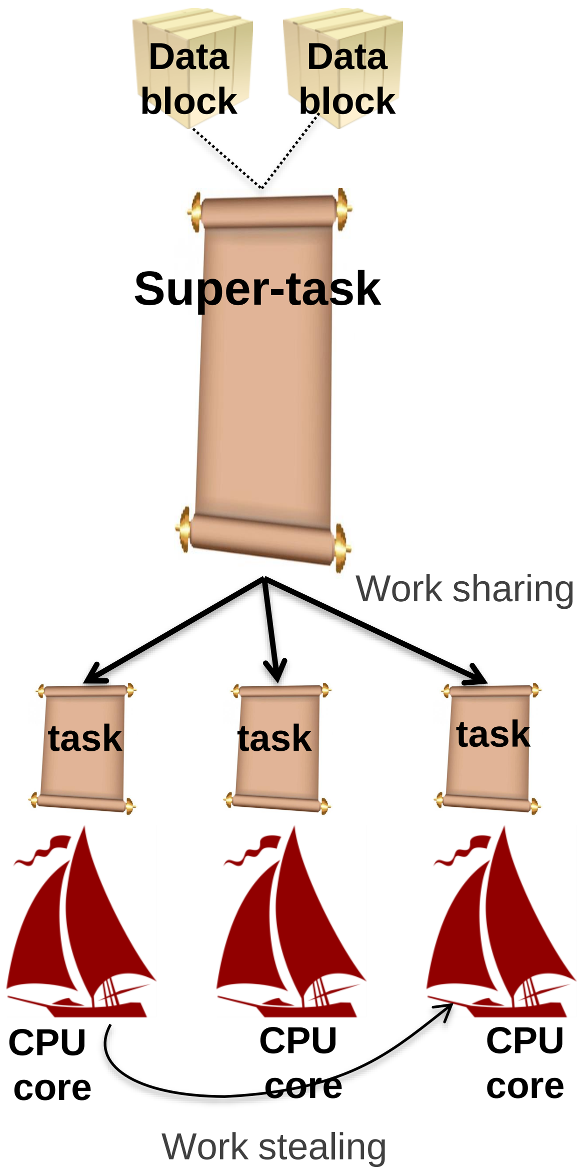
- Keep track of data residency (NUMA node) to **guide scheduling**

**Common feature with Minas [(4)]**

- Allocate memory and **distribute pages across NUMA nodes according to provided pattern**.

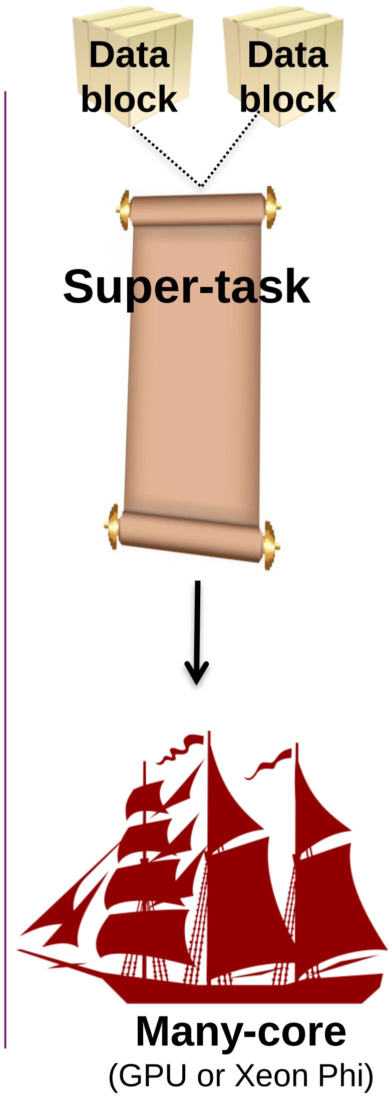
(1) Cédric Augonnetcet al., *StarPU : A unified platform for task scheduling on heterogeneous multicore architectures*, Euro-Par'09
(2) Thierry Gautier et al., *Xkaapi : A runtime system for data-flow task programming on heterogeneous architectures*, IPDPS 2013
(3) Alejandro Duran et al., *Ompss : a proposal for programming heterogeneous multi-core architectures*. Parallel Processing Letters 2011
(4) Pousa Ribeiro (C.). et al., *Minas: Memory Affinity Management Framework.* INRIA 2009.

## *HETEROGENEOUS LOAD BALANCING WITH SUPER-TASKS*




### Corner stone of the H3LMS platform

- **Multi-level load balancing. Decomposable tasks to adapt the workload to the target architecture.**
- **Increased spatial locality on targets + work stealing between units of the same type.**
- **2 scheduling modes :**
  - **For compute intensive applications** Dynamic heterogeneous load balancing by using shared queue of super-tasks.
  - **Data centric to minimize transfers** by directly using local queues.



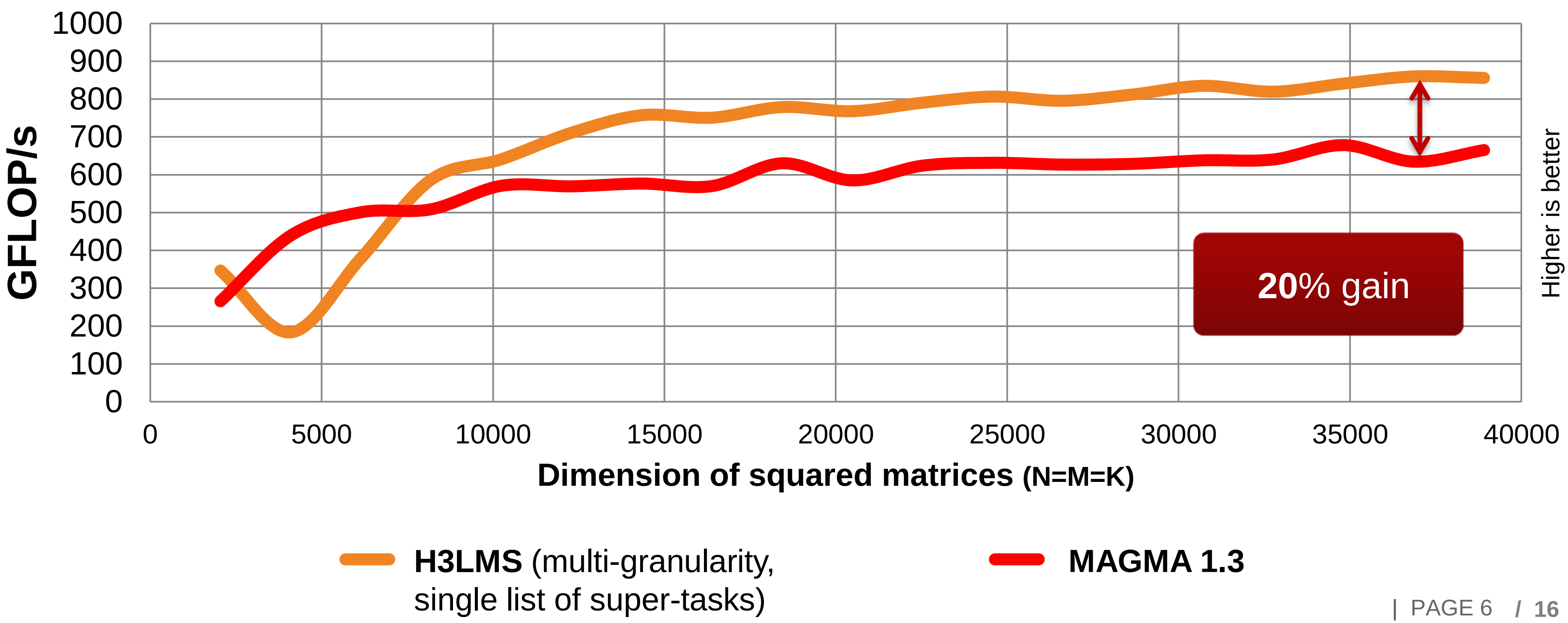
## MAGMA [(1)]

- Linear-algebra library for a heterogeneous node
- StarPU task scheduler
- Kernels: Intel MKL + Nvidia CuBLAS

**CPUs:** 2 Intel Xeon X5660 (2x 4 cores)
**GPUs:** 2 Nvidia Tesla M2050
(Blocks: 1024x1024, Sub-blocks: 256x256)

## Matrix multiply (SGEMM)

**Comparison with MAGMA** (including data transfers)






(1) Agullo (E.) et al., *Numerical linear algebra on emerging architectures : The PLASMA and MAGMA projects*. Journal of Physics, 2009

# HIERARCHICAL AFFINITIES IN H3LMS

*ABSTRACT ORGANIZATION BASED ON THE HARDWARE TOPOLOGY*

***Extended with an abstract organization***

(based on the HWLOC [1] library)

included into

**Worker Unit** (constant granularity)
- Two kinds: small and accelerators
- Execute local tasks

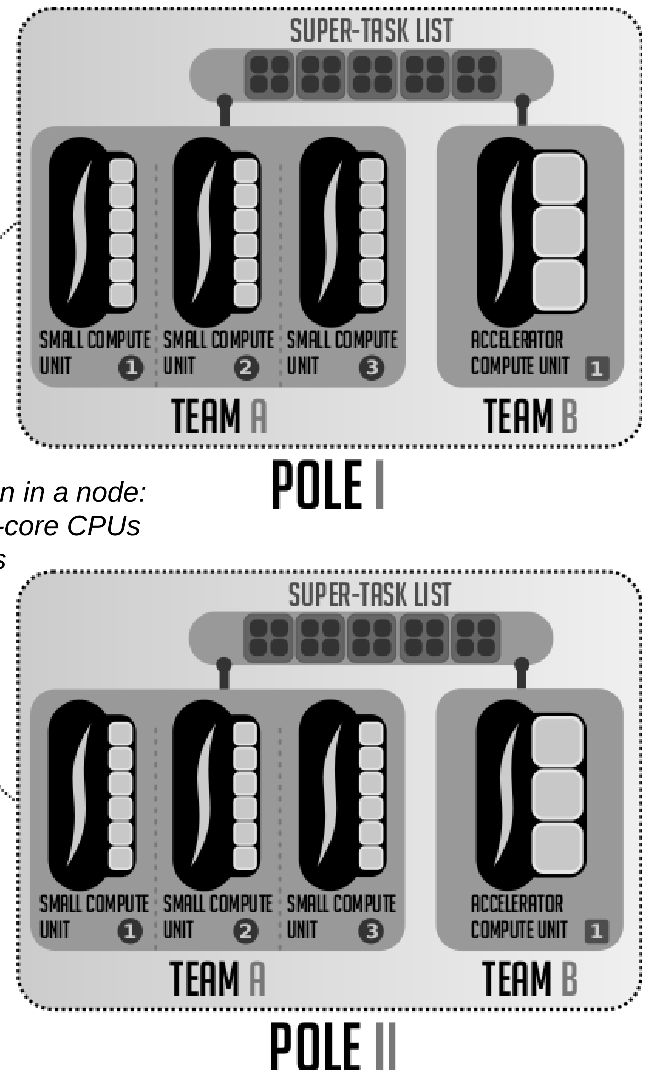
**Work Team** (shared same physical memory)
- Execute super-tasks
- First level of work-stealing

**Work Pole** (memory affinity)
- NUMA / NUIOA affinities
- As many lists of super-tasks as NUIOA nodes
- **COMPAS** to help selecting the pole and super-task-list

***Choice of the list at the begining of a bulk of super-tasks***
(~owner compute rule[2])

**Pole hierarchy**
- Poles organized in tree to map NUMA distances

*Abstract organization in a node: 2 NUMA nodes, 2x 4-core CPUs and 2 accelerators*

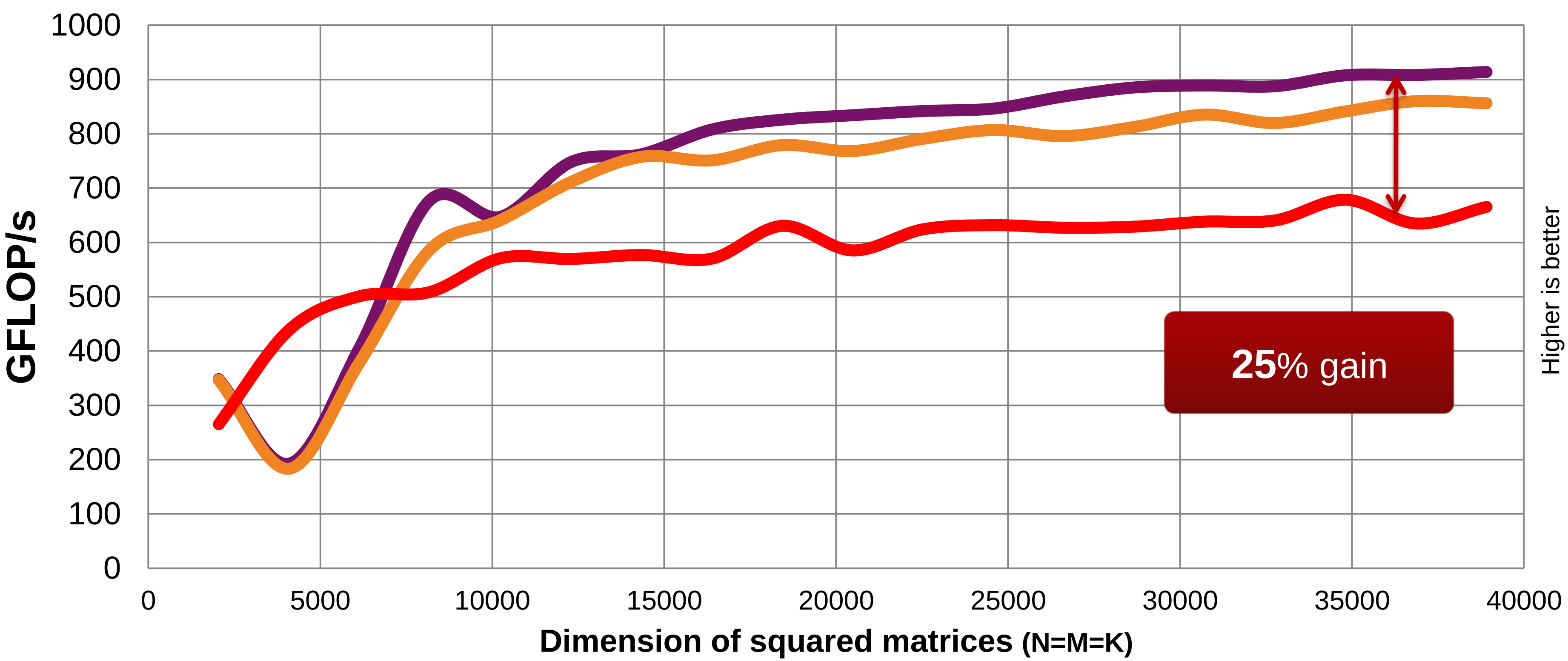





(1) F. Broquedis et al., *HWLOC : A generic framework for managing hardware affinities in HPC applications*. PDP '10.
(2) D. Callahan, et al., Compiling Programs for Distributed Memory Multiprocessors.The Journal of Supercomputing 1988.

## Matrix multiply (SGEMM)

**Comparison with MAGMA** (including data transfers)




- **H3LMS** (multi-granularity, hierarchical affinities) + **COMPAS**
- **H3LMS** (multi-granularity, single list of super-tasks)
- **MAGMA 1.3**

**CPUs:** 2 Intel Xeon X5660 (2x 4 cores)
**GPUs:** 2 Nvidia Tesla M2050
(Blocks: 1024x1024, Sub-blocks: 256x256)

# REDUCE TRANSFERS WITH COMPAS

*BENEFIT FROM THE SOFTWARE CACHES BETWEEN LIBRARY CALLS*

**MAGMA with MORSE** (1)

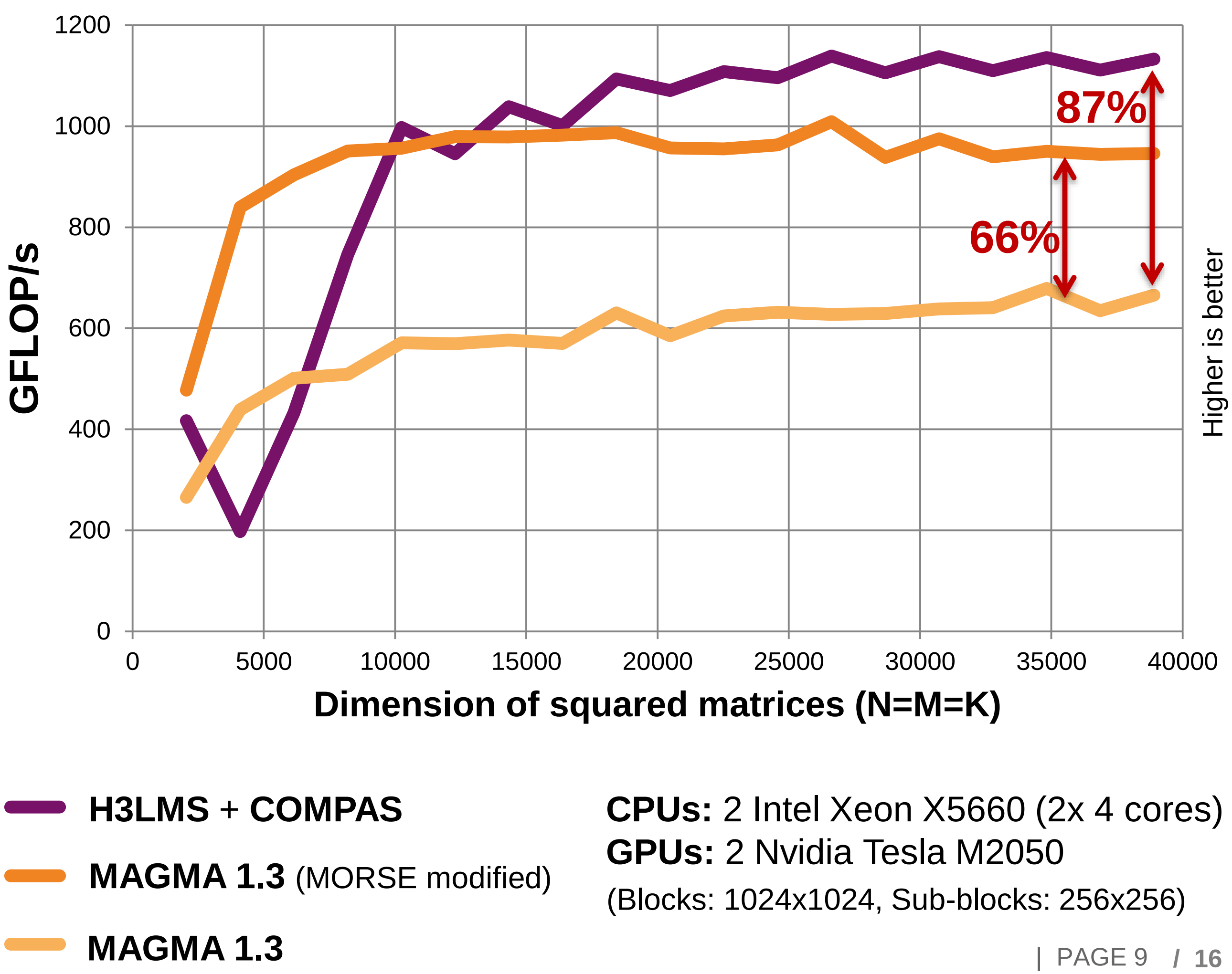
- Interface to link to linear-algebra libraries to runtime systems
- e.g. MAGMA on StarPU

**With COMPAS:**

- Keep same granularity of data blocks
- Turn off systematic flushing of software caches between library calls.

**5 SGEMMs accumulating in the same matrix** (C = A * B + C)

**Comparison with MAGMA** (including data transfers)




**CPUs:** 2 Intel Xeon X5660 (2x 4 cores)
**GPUs:** 2 Nvidia Tesla M2050
(Blocks: 1024x1024, Sub-blocks: 256x256)



(1) Matrices Over Runtime Systems @ Exascale (MORSE)

# COUPLING WITH MPI AND OPENMP CODES

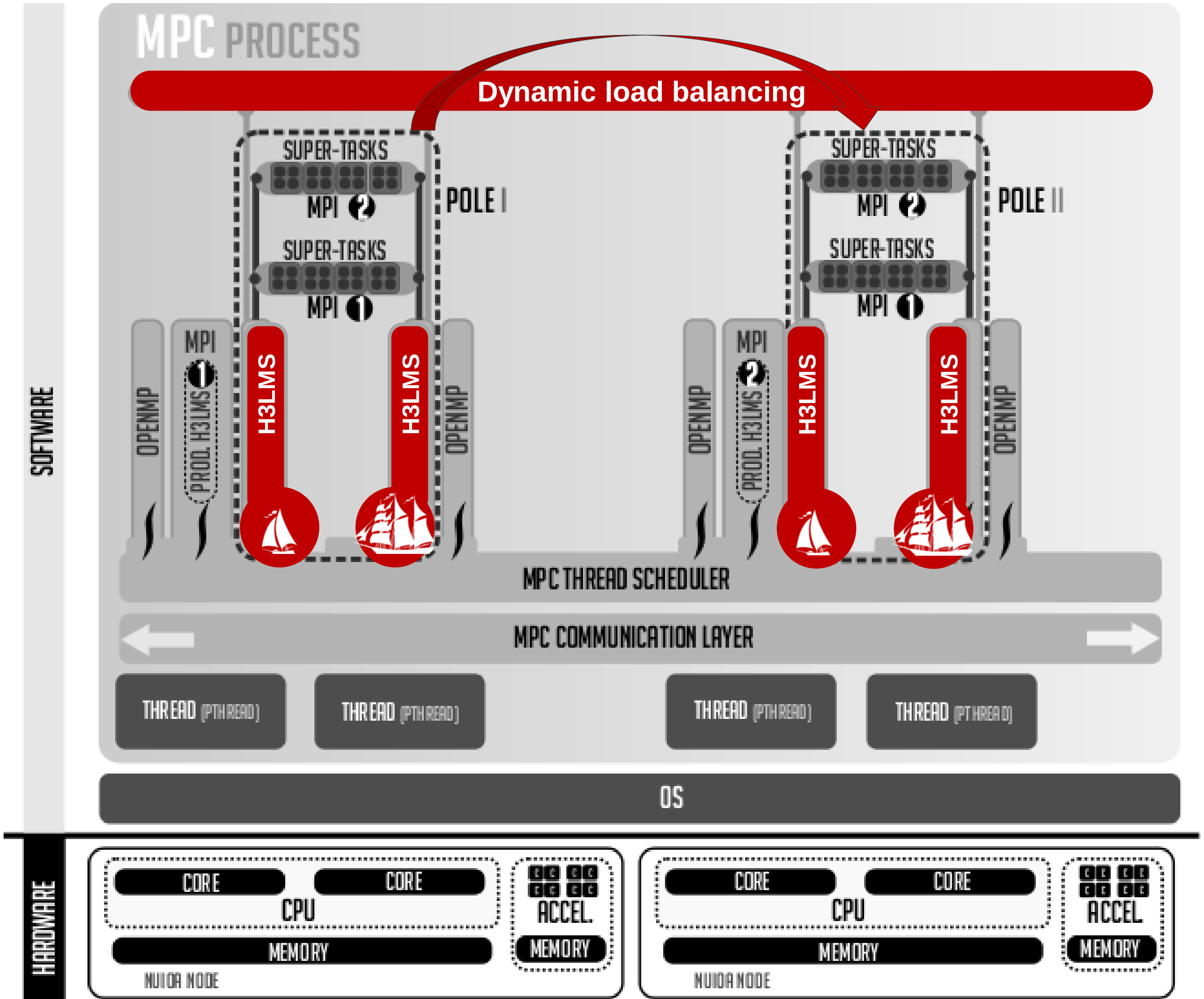
## IMPLEMENTATION INTO THE MPC FRAMEWORK



## MPC[1]

*(Multi-Processor Computing)*

- **Framework developped by the CEA and the ECR**
- **Thread based MPI tigthly coupeled to an OpenMP implementation**




## H3LMS-MPC

- **Rely on MPC for inter-node communications**
- **Load balancing with H3LMS, super-tasks generated from different MPI taks in the same compute node**



(1) M. Pérache et al., MPC: A unified parallel runtime for clusters of NUMA machines. Euro-Par '08.

# EVALUATION WITH LEGACY CODES

# LINPACK
## *HETEROGENEOUS EXECUTION OF THE HOT SPOT* (1/2)

## LINPACK (HPL 2.0)

**Need to modifiy 3 lines of code**

### 1 Allocate page-locked memory with COMPAS.

testing/ptest/HPL_pdtest.c **Before**

```
vptr = (void*)malloc(
        ((size_t)(ALGO->align)+
        (size_t)(mat.ld+1)*
        (size_t)(mat.nq))*
         sizeof(double));
```

testing/ptest/HPL_pdtest.c **After**

```
vptr = (void*)(compas_malloc(NULL,
         COMPAS_PLOCKED,
        ((size_t)(ALGO->align)+
        (size_t)(mat.ld+1)*
        (size_t)(mat.nq))*
         sizeof(double));
```

### 2-3 Call BLAS function based on H3LMS.

include/hpl_blas.h **Before**

```
#define HPL_dgemm cblas_dgemm

#define HPL_dtrsm cblas_dtrsm
```

include/hpl_blas.h **After**

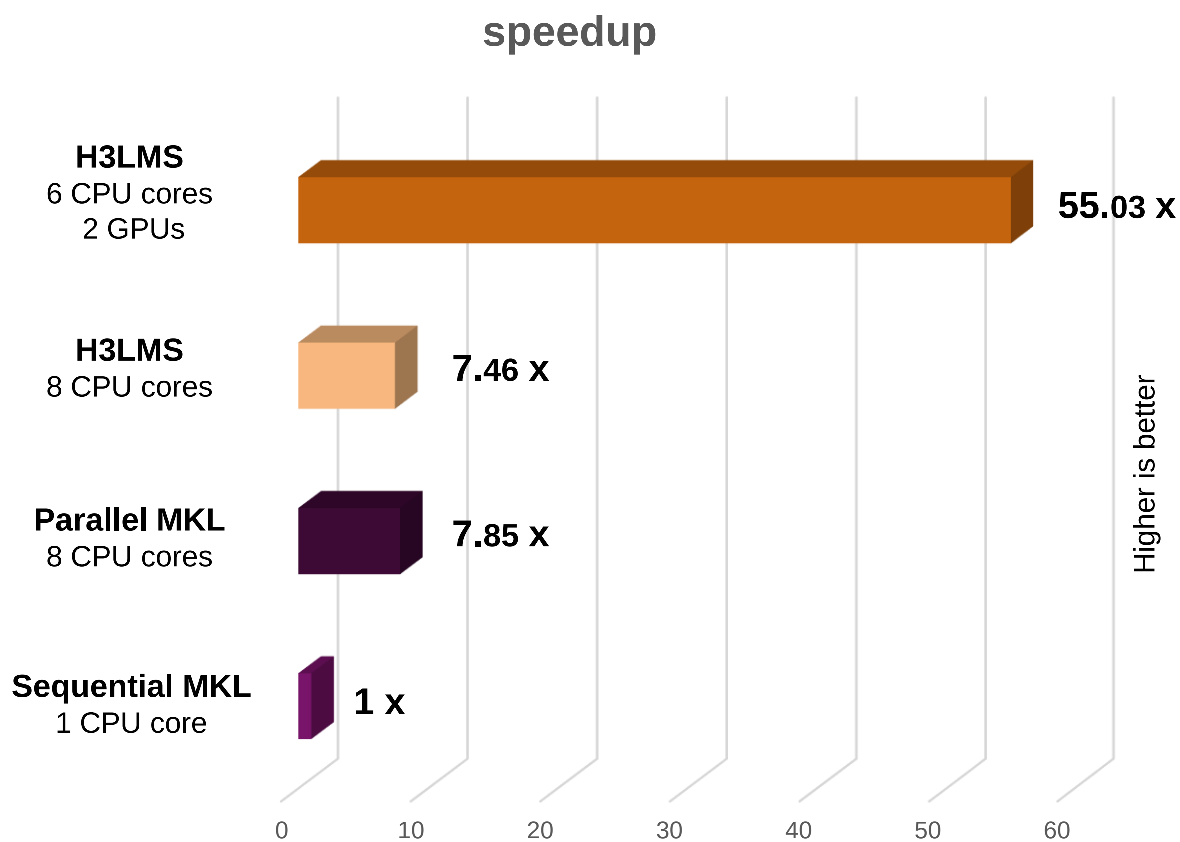
```
#define HPL_dgemm (...)
      h3lms_blas_dgemm_ext_sync(__VA_ARGS__,
       4096,-1,256,-1)
#define HPL_dtrsm h3lms_blas_dtrsm_sync
```

# LINPACK

## *HETEROGENEOUS EXECUTION OF THE HOT SPOT (2/2)*

## LINPACK – HPL 2.0

(N = 46080, N B = 512, P = 1, Q = 1, WC10L2L2)

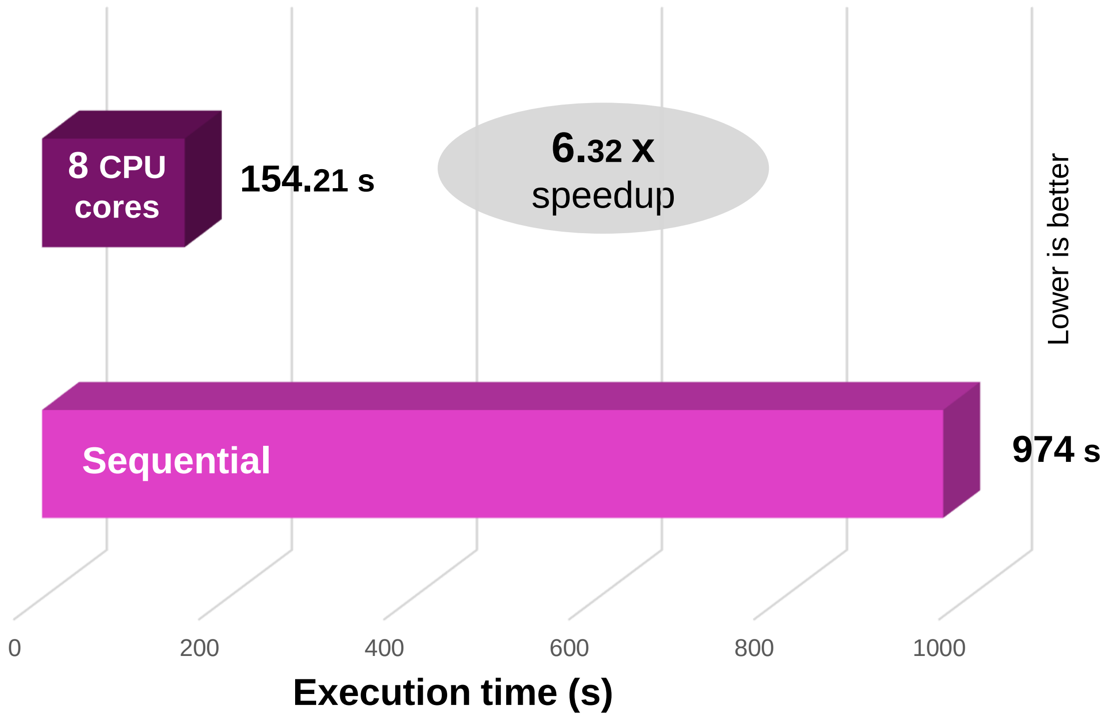



- Based on optimized libraries Intel MKL 10.1 and NVIDIA CUBLAS 4.2
- Transparent for the user with synchronous function calls
- Internally decomposed into super-tasks and tasks
- Homogeneous performance close to parallel MKL (**62.11** vs **68.94 GFLOP/s**)
- Heterogeneous performance **482.4 GFLOP/s**

*2x Intel Xeon Nehalem EP E5620 (2x 4 cores @ 2.4 GHz, peak: 2x 38.4 GFLOPS)*
*2x NVIDIA Tesla Fermi M2090 (peak: 2x 665 GFLOPS)*
*24 GB of DDR3 memory*

# PN APPLICATION
## *INCREMENTAL CHANGES TO HARNESS HETEROGENEOUS COMPUTE RESOURCES (1/3)*

- **PN: solve the linear particle transport equation with deterministic resolution based on spherical harmonics approximation** (1)
- **Hybrid MPI-OpenMP code**
- **Focus on the** *numerical_flux* **function** (~***90%*** *execution time*).




**PN: CPUs performance of *numerical_flux***

- ***Double precision***
- *Cartesian mesh :* ***1536****x****1536****, N=****15****,* ***36*** *iter.*
- *Average on* ***20*** *runs*

**Tera-100 heterogeneous compute nodes** — **CPUs**: 2x 4 cores, Intel Xeon E5620



(1) Thomas A. Brunner and James Paul Holloway. Two dimensional time dependent riemann solvers for neutron transport. J. Comput. Phys., 210(1) :386–399, November 2005.

## *numerical_flux* function on a x * z Cartesian mesh

/* Part 1 */

DGEMM [read: $A_{X(136\times136)}$, $B_{X(x\times z\times136)}$] [write: $C_{X(x\times z\times136)}$]

DGEMM [read: $A_{Z(136\times136)}$, $B_{Z(x\times z\times136)}$] [write: $C_{Z(x\times z\times136)}$]

1 « Large » matrix multiples

/* Part 2 */

DGEMM [read: $A_{X(136\times136)}$, $BL_{X(x\times136)}$] [write: $CL_{X(x\times136)}$]

DGEMM [read: $A_{Z(136\times136)}$, $BL_{Z(z\times136)}$] [write: $CL_{Z(z\times136)}$]

DGEMM [read: $A_{X(136\times136)}$, $BR_{X(x\times136)}$] [write: $CR_{X(x\times136)}$]

DGEMM [read: $A_{Z(136\times136)}$, $BR_{Z(z\times136)}$] [write: $CR_{Z(z\times136)}$]

**BARRIER**

2 Small matrix multiplies

/* Part 3 */

SUPER-TASK [read: $D_{X(1024)}$, $C_{X(x\times z\times136)}$] [write: $E_{X(x\times z\times136)}$]

SUPER-TASK [read: $D_{Z(1024)}$, $C_{Z(x\times z\times136)}$] [write: $E_{Z(x\times z\times136)}$]

**BARRIER**

SUPER-TASK [read: $D_{X(1024)}$, $CL_{X(x\times136)}$] [read-write: $E_{X(x\times z\times136)}$]

SUPER-TASK [read: $D_{Z(1024)}$, $CL_{Z(z\times136)}$] [read-write: $E_{Z(x\times z\times136)}$]

**BARRIER**

SUPER-TASK [read: $D_{X(1024)}$, $CR_{X(x\times136)}$] [read-write: $E_{X(x\times z\times136)}$]

SUPER-TASK [read: $D_{Z(1024)}$, $CR_{Z(z\times136)}$] [read-write: $E_{Z(x\times z\times136)}$]

**BARRIER**

3 Consecutive loops operating on each cell of the mesh

/* Part 4 */

DGEMM [read: $F_{X(136\times136)}$, $E_{X(x\times z\times136)}$] [write: $G_{X(x\times z\times136)}$]

DGEMM [read: $F_{Z(136\times136)}$, $E_{Z(x\times z\times136)}$] [write: $G_{Z(x\times z\times136)}$]

**BARRIER**

4 « Large » matrix multiplies

**FLUSH_SOFTWARE_CACHES**

# PN APPLICATION

## *INCREMENTAL CHANGES TO HARNESS HETEROGENEOUS COMPUTE RESOURCES (3/3)*

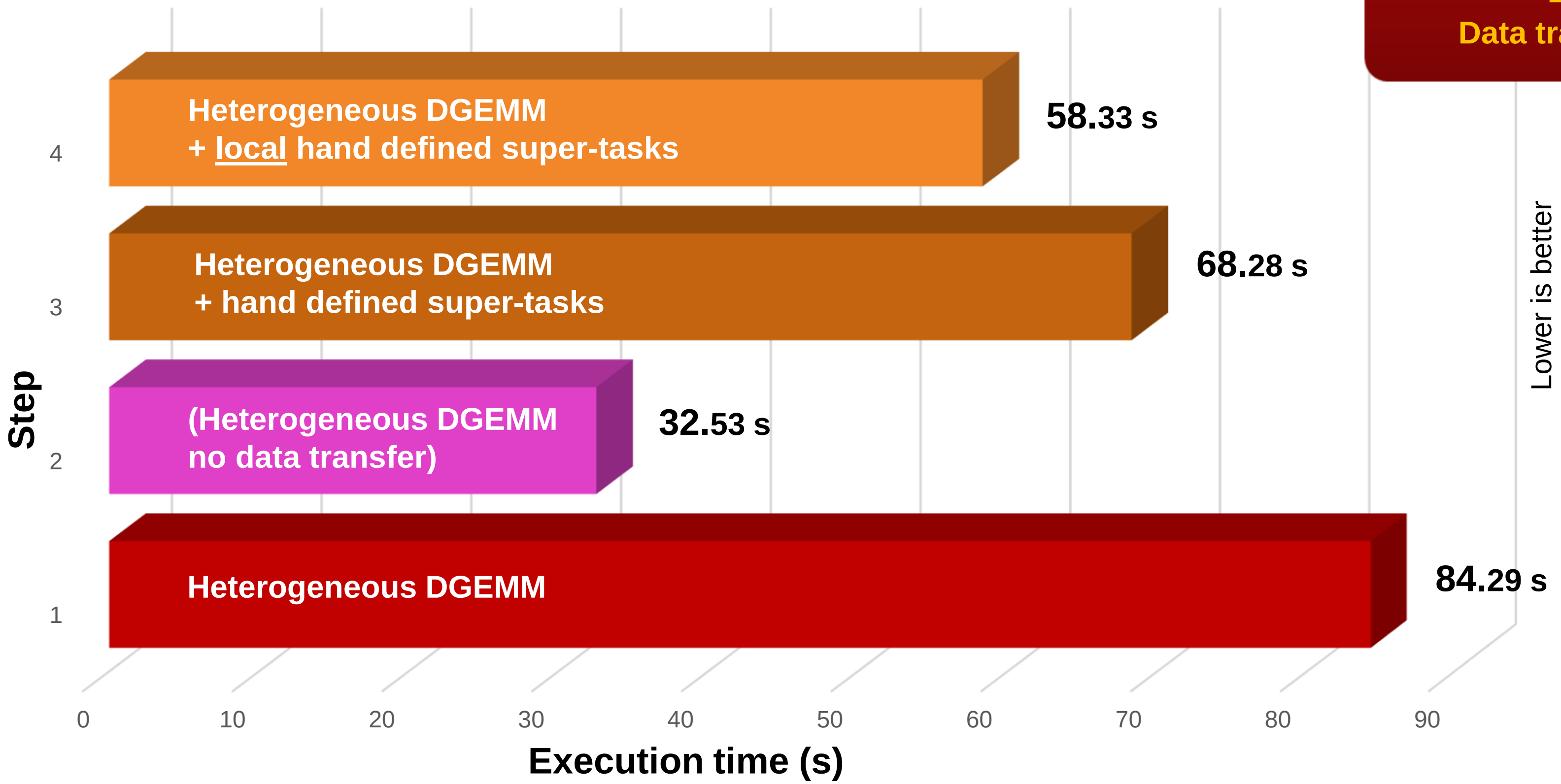
**Reference**

Homogeneous 8 cores: **154.21 s**

## PN: heterogeneous performance of *flux numerical_flux*

**H3LMS+ COMPAS: multi-granularity, spatial and temporal locality**

Final speedup
Heterogeneous vs homogeneous
**2.65 X**
**Data transfer bound**



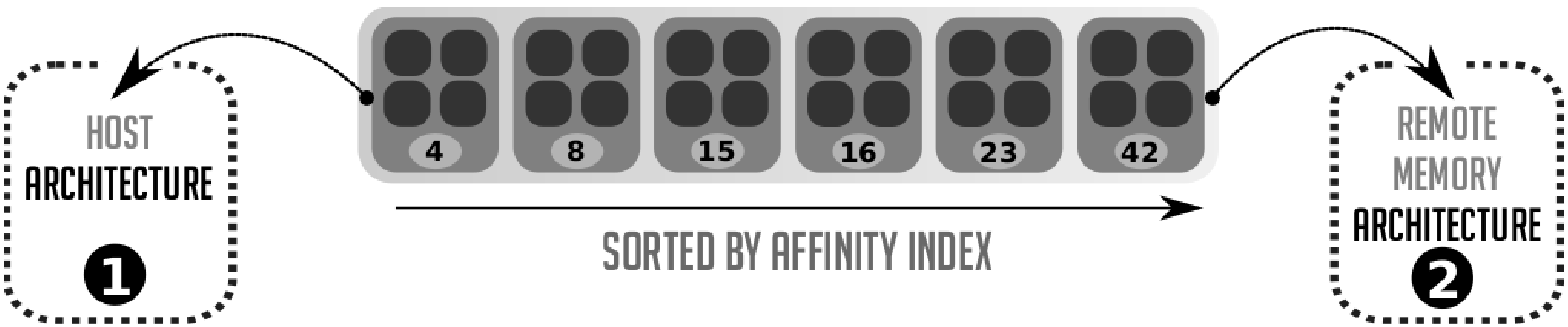

***double precision**, Cartesian mesh: **1536**x**1536**, N=**15**, **36** iter. averaged on **20** runs*

**Tera-100 heterogeneous node**

**CPUs:** 2x 4 cores Intel Xeon E5620
**Accel.:** 2x Nvidia Telsa GTX M2090

THANK YOU

## Software cache and execution order of the super-tasks[(1)]




- **Sort list of super-task at the bulk instantiation according to affinity with the corresponding memory.**
- **New scheduling policy based on cache statistics to reduce data transfers.**

**Affinity index depends on**:

- Quantity of data
- If blocks of data are already held inside the cache



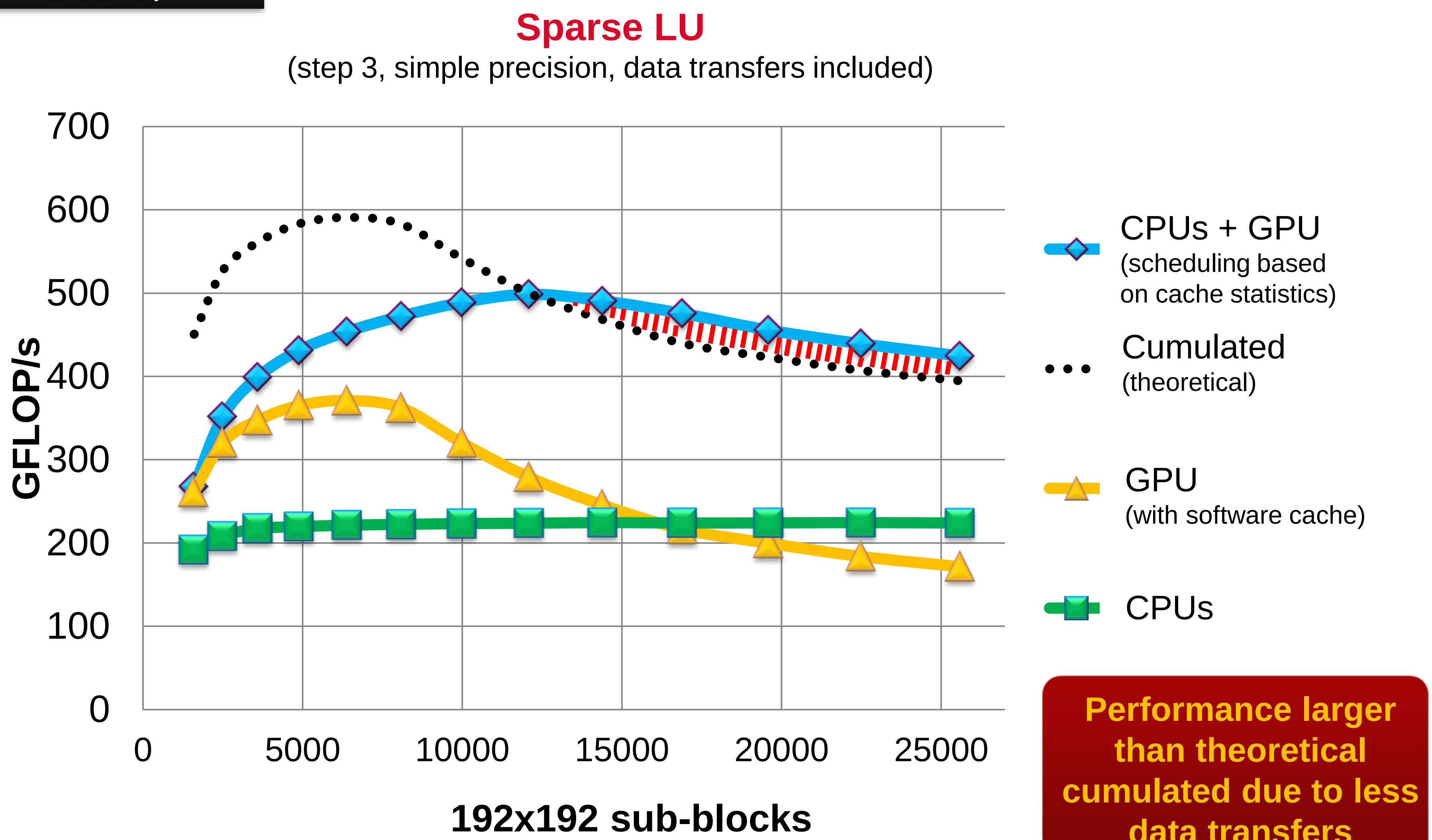
(1) Jean-Yves Vet, Patrick Carribault, and Albert Cohen. Multigrain affinity for heterogeneous work stealing. MULTIPROG-2012.

# SUPER-TASK AFFINITY

*IMPROVING TEMPORAL LOCALITY (2/2)*



Backup




**Performance larger than theoretical cumulated due to less data transfers**

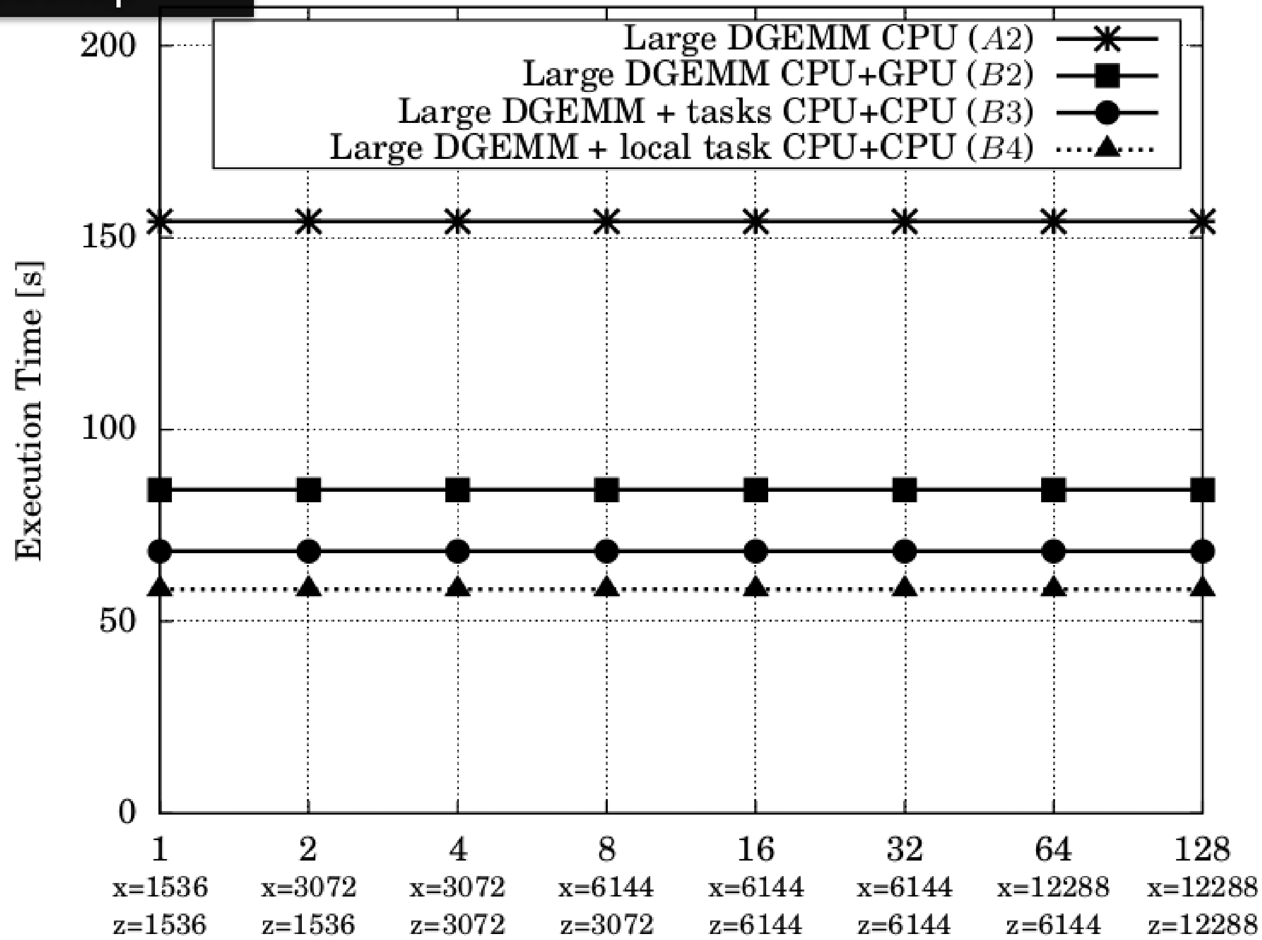
**CPUs:** 2x 12 cores AMD Opteron 6164 HE
**Accel.:** 1 GPU Nvidia Geforce GTX 470

# PN APPLICATION

## *MULTI-NODES STRONG SCALING*



Backup




**# compute nodes and mesh dimensions**

**PN: heterogeneous performance of *flux numerical_flux* on multiple compute nodes**

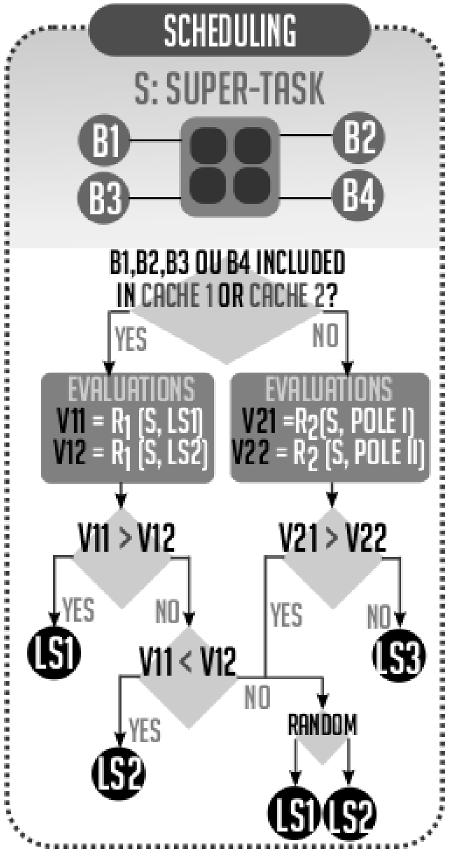
**Tera-100 heterogeneous node**

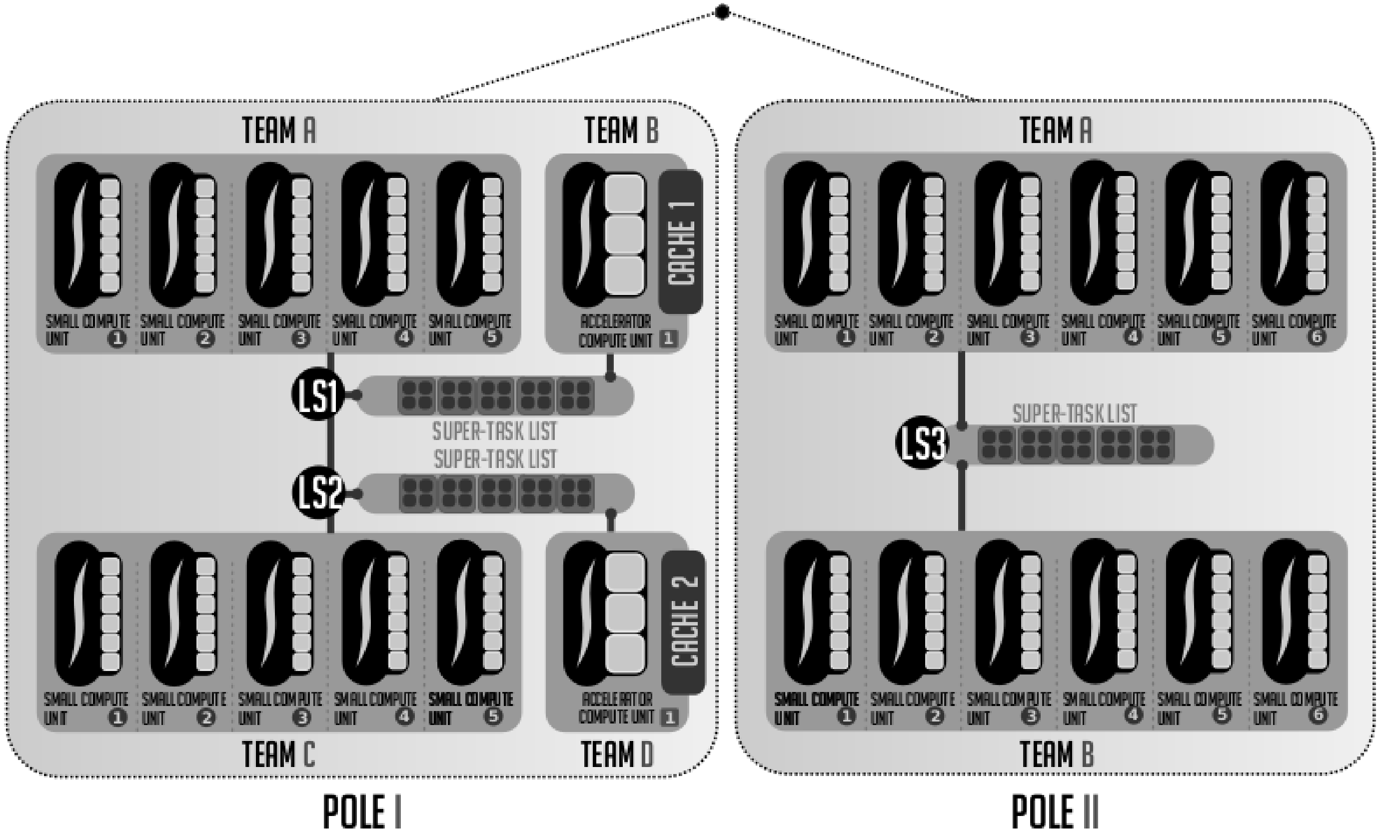
**CPUs:** 2x 4 cores Intel Xeon E5620
**Accel.:** 2x Nvidia Telsa GTX M2090

# HIERARCHICAL AFFINITIES IN H3LMS

## *CHOICE OF THE SUPER-TASK LIST*

Backup




*Abstract organization in a node:*
*2 NUMA nodes,*
*2x 12-core AMD Magny-Cours CPUs (Dual package, 1 memory controler per CPU)*
*and 2 accelerators attached to the same NUMA node*

# API - COMPAS

## Backup

### COMPAS API

```
struct compas_pages_s pages;
compas_pages_init(&pages);
pages.geometry = COMPAS_2D;
pages.pattern = COMPAS_CYCLIC;
pages.xsize = NB*BS;
pages.ysize = NB*BS;
pages.xblocksize = BS;
pages.yblocksize =
    (BS*sizeof(double))/PSIZE;
distri.elsize =
    (BS*sizeof(double))/PSIZE;

double *A = compas_malloc(&distri,
    COMPAS_PLOCKED, NB*BS*NB*BS
    *sizeof(double));
init_matrix(A);
```

### Proposal based on pragma

```
#pragma compas malloc
    elsize(sizeof(double))
    size(NB*BS,NB*BS)
    bsize((BS*sizeof(double))/PSIZE,
    (BS*sizeof(double))/PSIZE)
    pattern(cyclic)
    page(locked)

double *A = malloc(NB*BS*NB*BS *
    sizeof(double));
init_matrix(A);
```

DE LA RECHERCHE À L'INDUSTRIE

# API - H3LMS

INVENTORS FOR THE DIGITAL WORLD

## Backup

### H3LMS API

```
H3LMS_BEGIN_SUPERTASK_FUNCTION(l_bmod_cpu)
  double *A1 = GETDATA(0, double*);
  double *A2 = GETDATA(1, double*);
  double *A3 = GETDATA(2, double*);

  const int subblocks = 16;
  const int subblocksize = BS/subblocks;

   for (int i=0; i<subblocks; i++) {
     double *SA1 = A1;
     double *SA2 = &A2[i*subblocksize
         *BS*NB];
     double *SA3 = &A3[i*subblocksize
         *BS*NB];
     H3LMS_BEGIN_SUBTASK_DECLARATION
        H3LMS_REGISTER_ARGS(SA1, SA2, SA3,
            subblocksize)
        H3LMS_CPU_LAUCHER_FUNCTION(
            bmod_cpu)
     H3LMS_END_SUBTASK_DECLARATION
   }
H3LMS_END_SUPERTASK_FUNCTION
...

for (int ii=kk+1; ii<NB; ii++) {
  for (int jj=kk+1; jj<NB; jj++) {
    H3LMS_BEGIN_SUPERTASK_DECLARATION
       H3LMS_REGISTER_NB_DEP(3)
       H3LMS_REGISTER_ARGS( &A[ii*MROW +
           kk*BS], &A[kk*MROW + jj*BS],
           &A[ii*MROW + jj*BS])
       H3LMS_REGISTER_DEP_2D( &A[ii*MROW +
           kk*BS], IN,sizeof(double),
           BS,BS*NB,BS)
       H3LMS_REGISTER_DEP_2D( &A[kk*MROW +
           jj*BS], IN,sizeof(double),
           BS,BS*NB,BS)
       H3LMS_REGISTER_DEP_2D( &A[ii*MROW +
           jj*BS], INOUT,sizeof(double),
           BS,BS*NB,BS)
       H3LMS_CPU_LAUCHER_FUNCTION(
           l_bmod_cpu)
       H3LMS_ACCEL_LAUCHER_FUNCTION(
           l_bmod_accel)
       H3LMS_COMPUTE_RATIO(5)
    H3LMS_END_SUPERTASK_DECLARATION
  }
```

### Proposal based on pragmas

```
for (int ii=kk+1; ii<NB; ii++) {
  for (int jj=kk+1; jj<NB; jj++)
      {
      #pragma h3lms supertask
          depend(in:
          A[ii*BS:ii*BS+BS-1,
          kk*BS:kk*BS+BS-1],
          A[kk*BS:kk*BS+BS-1,
          jj*BS:jj*BS+BS-1], inout:
          A[ii*BS:ii*BS+BS-1,
          jj*BS:jj*BS+BS-1])
          accel(l_bmod_cuda)
          cratio(5)
      {
       const int subblocks = 16;
       const int subblocksize = BS
           /subblocks;

       for (int i=0; i<subblocks;
           i++) {
         double *SA1 = &A[ii*MROW
             + kk*BS];
         double *SA2 = &A[kk*MROW
             + jj*BS + i*
             subblocksize*BS*NB];
         double *SA3 = &A[ii*MROW
             + jj*BS + i*
             subblocksize*BS*NB];

         #pragma h3lms subtask
         bmod_cpu (SA1, SA2, SA3,
             subblocksize);
       }
     }
   }
}

#pragma h3lms barrier
```

# Glossary

**BCS** – Bull Coherence System

**BLAS** – Basic Linear Algebra Subprograms

**COMPAS** – Coordinate and Organize Memory Placement and Allocation for Scheduler

**CPU** – Central Processing Unit

**DGEMM** – Double precision matrix matrix multiplication

**DSM** – Distributed Shared Memory

**DTRSM** – Solves one of the matrix equations op( A )*X = alpha*B, or X*op( A ) = alpha*B

**GFLOPS** – Giga FLoating-point Operations Per Second

**GPU** – Graphics Processing Unit

**H3LMS** – Harnessing Hierarchy and Heterogeneity with Locality Management and Scheduling

**HPC** – High Performance Computing

**HPL** – High Performance Linpack

**LRU** – Least Recently Used

**NUMA** – Non-Uniform Memory Access

**NUIOA** – Non-Uniform Input/Output Access

**PFLOPS** – Peta FLoating-point Operations Per Second