Un retour sur les activités INRIA menées sur le MIC

Inria Bordeaux Sud Ouest

Journées MCIA
Université de Pau et des Pays de l'Adour

1. Portable Hardware Locality - hwloc

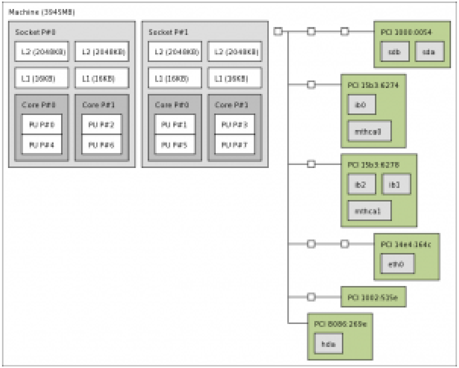
2. StarPU: Hybrid CPU/GPU Task Programming

3. Nbody Problem on Xeon Phi

4. conclusion

1

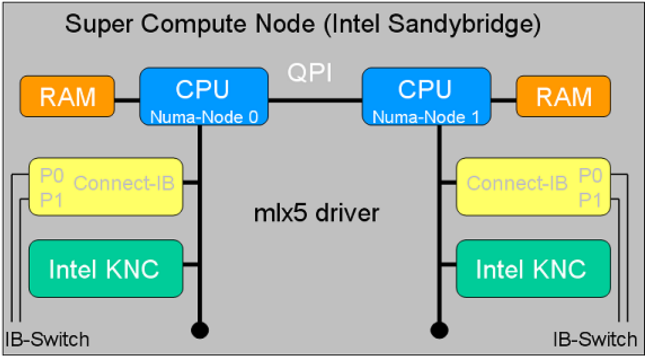
Portable Hardware Locality - hwloc



Portable Hardware Locality - hwloc

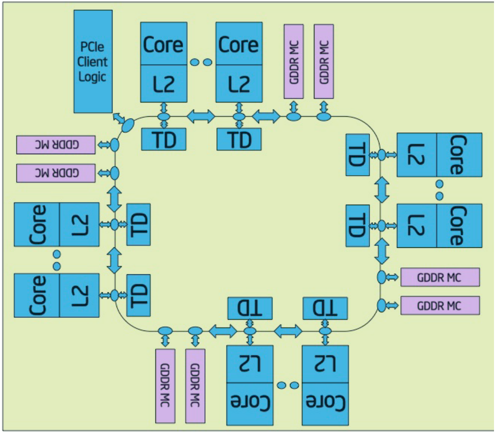
Respecter la localité des Xeon Phi pour améliorer leur utilisation

- Mise en place de stratégies pour connaître les coeurs et bancs mémoire proches de chaque Xeon Phi ⇒ Communication améliorées entre hôte et accélérateur
- Implémenté dans hwloc



Respecter la localité dans les Xeon Phi

- Architecture à 57-61 coeurs reliés en anneau
- Etude de l'impact de cette topologie sur les communications inter-coeur ⇒ L'anneau semble bien dimensionné, peu d'influence notée

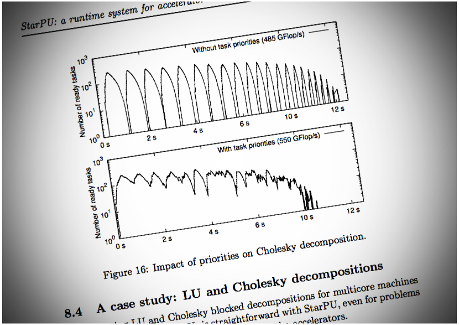


Etude des pragmas Intel LEO (Language Extensions for Offload)

- Offload une partie du code sur le Phi
  - Peut-être combiné avec de l'OpenMP à l'intérieur
- Pragmas avancés pour spécifier les transferts de données
  - Ex: garder des données sur le Phi entre deux sections offloadées
- Pragmas avancés pour réduire la synchronisation
  - Exécuter un noyau offloadé en tâche de fond
    - L'hote peut faire autre chose en attendant
    - Puis tester la terminaison de l'offload plus tard ⇒ Pragmas intéressants mais complexes
  - Si les pragmas avancés sont nécessaires à l'obtention de perfs, ca va etre dur pour les non-informaticiens...

2

StarPU: Hybrid CPU/GPU Task Programming



StarPU: a runtime system for accelerat

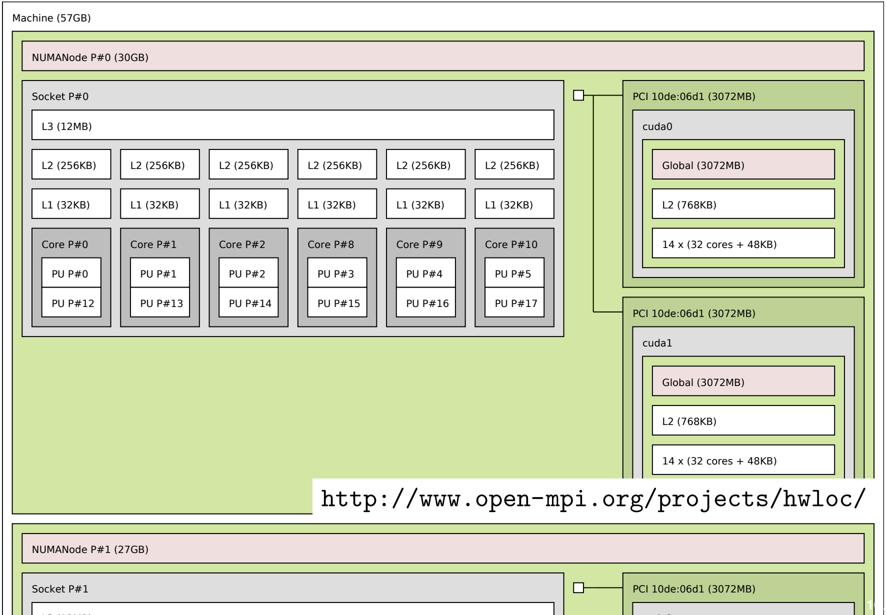
Figure 16: Impact of priorities on Cholesky decomposition.

8.4 A case study: LU and Cholesky decompositions

LU and Cholesky blocked decompositions for multicore machines straightforward with StarPU, even for problems accelerators.

StarPU: Hybrid CPU/GPU Task Programming

What today’s machines really look like



Machine (57GB)

NUMANode P#0 (30GB)

Socket P#0

L3 (12MB)

L2 (256KB) | L2 (256KB) | L2 (256KB) | L2 (256KB) | L2 (256KB) | L2 (256KB)

L1 (32KB) | L1 (32KB) | L1 (32KB) | L1 (32KB) | L1 (32KB) | L1 (32KB)

Core P#0: PU P#0, PU P#12

Core P#1: PU P#1, PU P#13

Core P#2: PU P#2, PU P#14

Core P#8: PU P#3, PU P#15

Core P#9: PU P#4, PU P#16

Core P#10: PU P#5, PU P#17

PCI 10de:06d1 (3072MB)

cuda0

Global (3072MB)

L2 (768KB)

14 x (32 cores + 48KB)

PCI 10de:06d1 (3072MB)

cuda1

Global (3072MB)

L2 (768KB)

14 x (32 cores + 48KB)

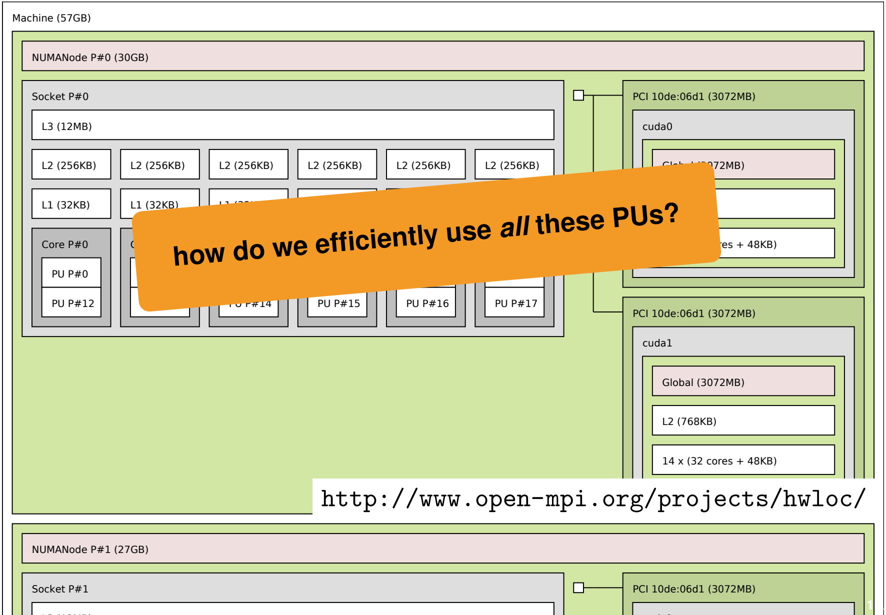
NUMANode P#1 (27GB)

Socket P#1

PCI 10de:06d1 (3072MB)

http://www.open-mpi.org/projects/hwloc/

What today’s machines really look like



how do we efficiently use *all* these PUs?

http://www.open-mpi.org/projects/hwloc/

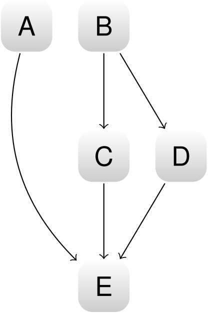
StarPU: runtime support to **schedule tasks** over all the available **processing units**

StarPU: runtime support to **schedule tasks** over all the available **processing units**

- C library, LGPLv2.1+
- started in 2009

In a nutshell

DAG of tasks



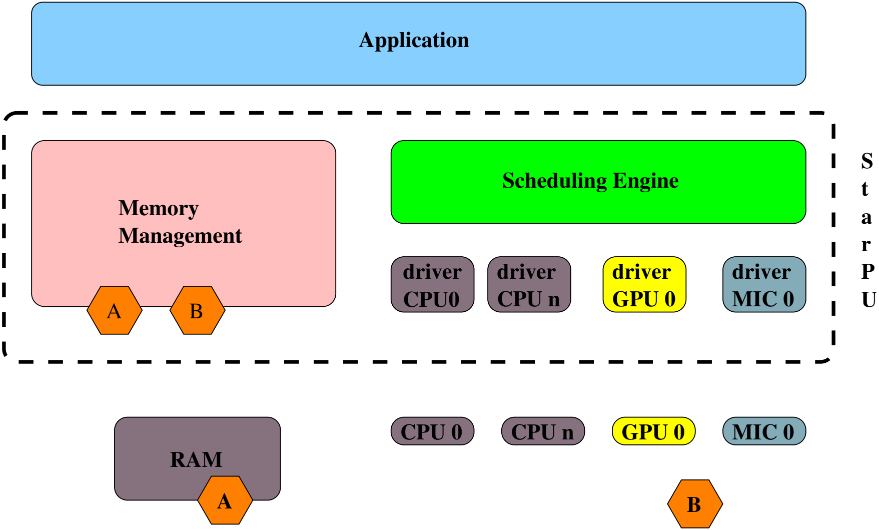
StarPU's runtime

scheduler

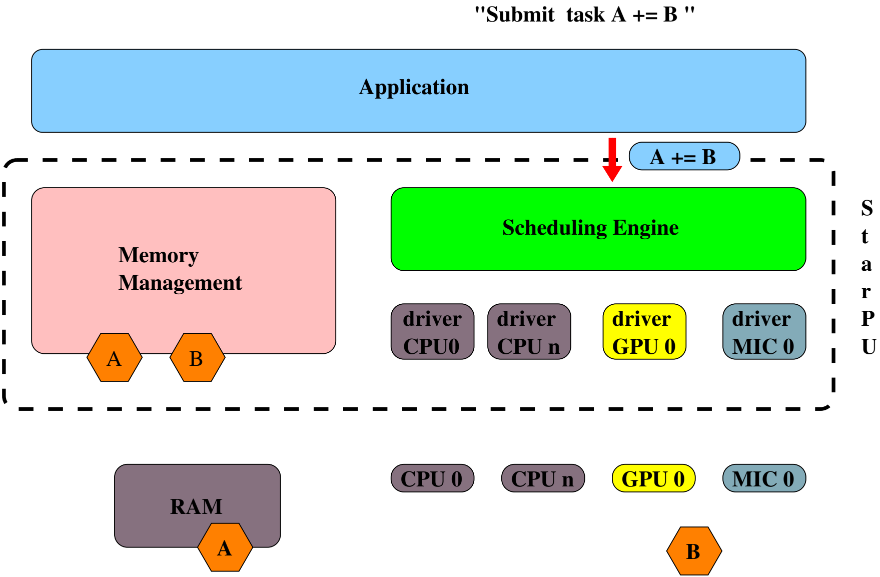
memory mgmt

CPU 0
CPU 1
CPU 2
CPU 3
GPU 0
GPU 1
MIC

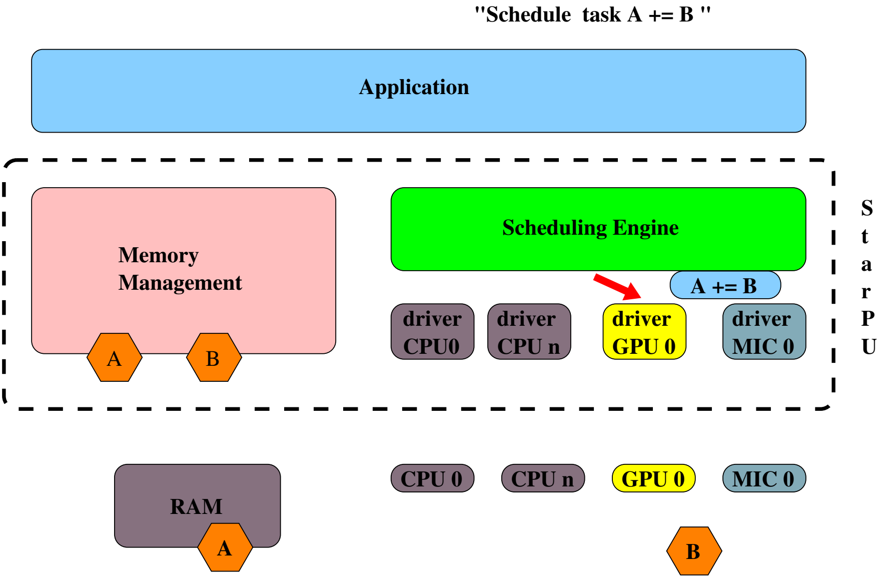
Execution model



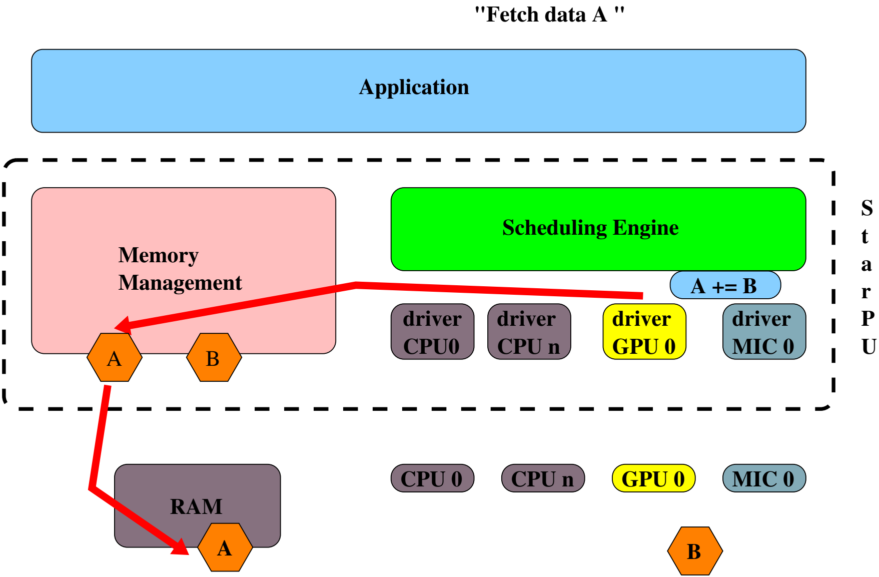
Execution model



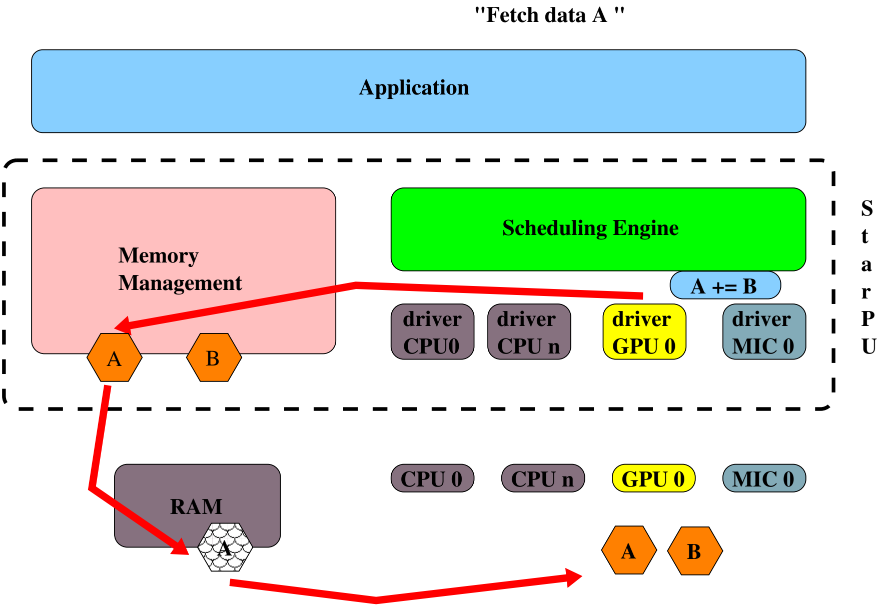
Execution model



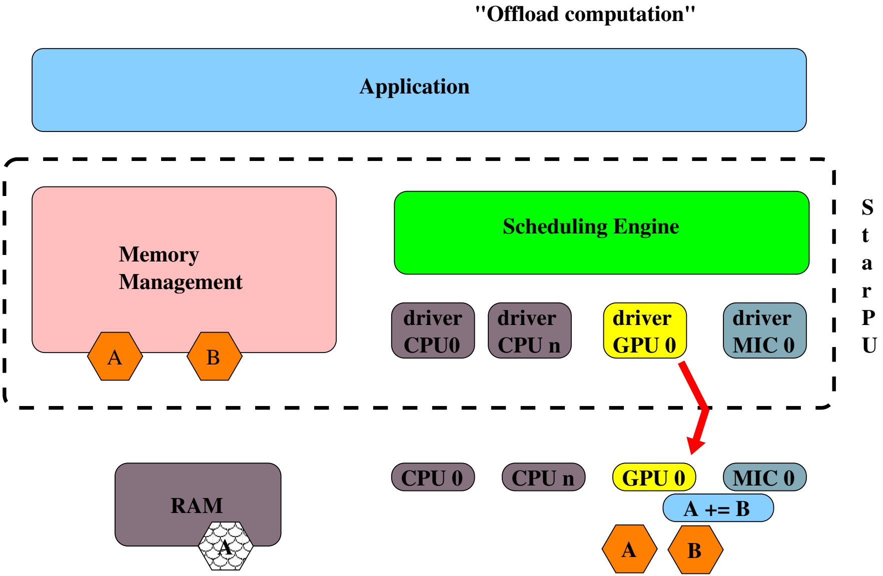
Execution model



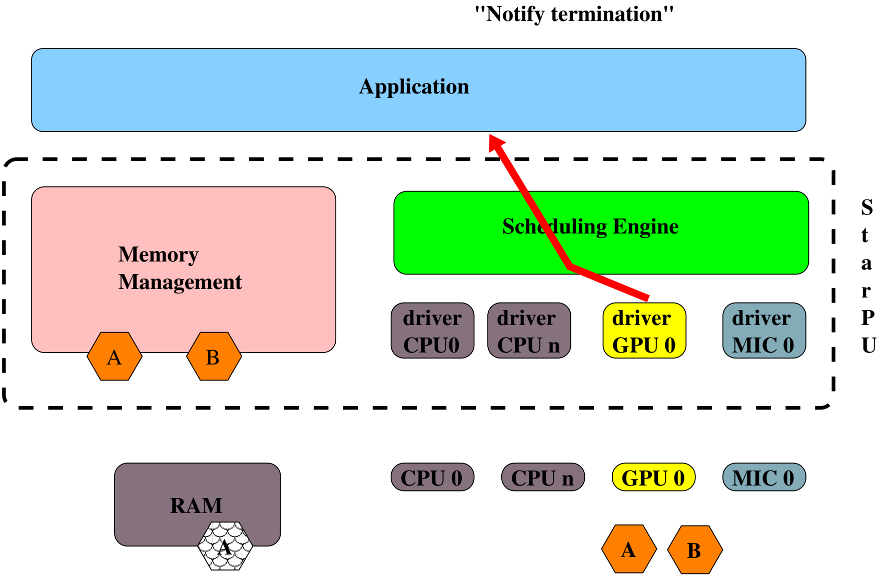
Execution model



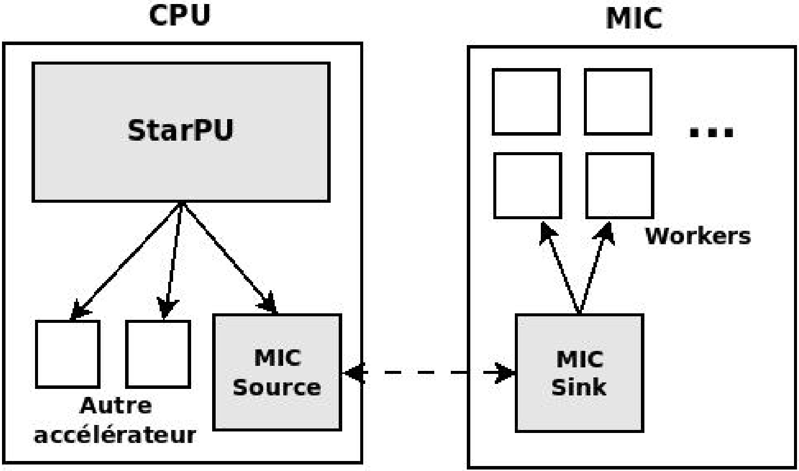
Execution model



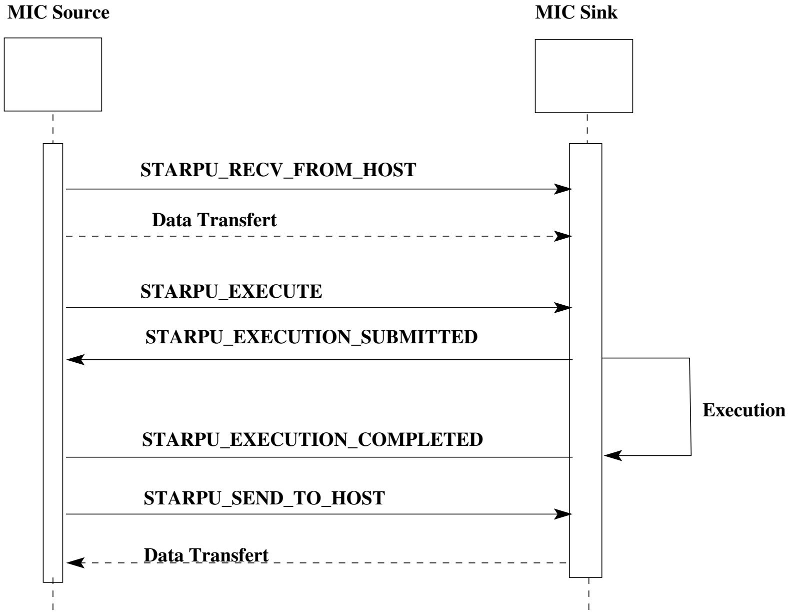
Execution model



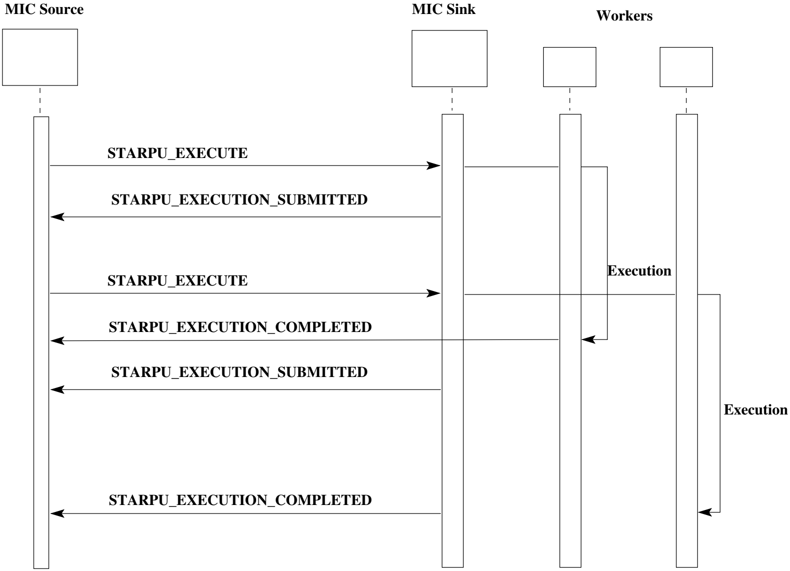
Interaction StarPU-MIC



Interaction StarPU-MIC



Interaction StarPU-MIC


MIC Source
MIC Sink
Workers
STARPU_EXECUTE
STARPU_EXECUTION_SUBMITTED
Execution
STARPU_EXECUTE
STARPU_EXECUTION_COMPLETED
STARPU_EXECUTION_SUBMITTED
Execution
STARPU_EXECUTION_COMPLETED

Definition of the MIC function

The same StarPU code is compiled both for the host and for the MIC. When executing a codelet on the MIC, StarPU will use a lookup mechanism to find out the function to be executed.

```
struct starpu_codelet cl =
{
.cpu_funcs = {cpu_codelet, NULL},

.cuda_funcs = {cuda_codelet, NULL},
.opencl_funcs = {opencl_codelet, NULL},
....
};
```

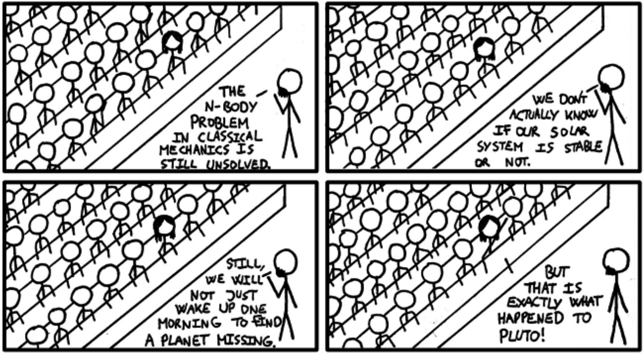
Definition of the MIC function

The same StarPU code is compiled both for the host and for the MIC. When executing a codelet on the MIC, StarPU will use a lookup mechanism to find out the function to be executed.

```
struct starpu_codelet cl =
{
.cpu_funcs = {cpu_codelet, NULL},
.cpu_funcs_name = {"cpu_codelet", NULL},
.cuda_funcs = {cuda_codelet, NULL},
.opencl_funcs = {opencl_codelet, NULL},
....
};
```

3

Nbody Problem on Xeon Phi

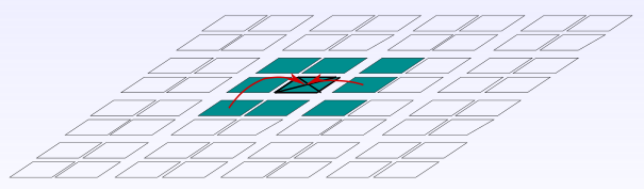


THE N-BODY PROBLEM IN CLASSICAL MECHANICS IS STILL UNSOLVED.
WE DON'T ACTUALLY KNOW IF OUR SOLAR SYSTEM IS STABLE OR NOT.
STILL, WE WILL NOT JUST WAKE UP ONE MORNING TO FIND A PLANET MISSING.
BUT THAT IS EXACTLY WHAT HAPPENED TO PLUTO!

Nbody Problem on Xeon Phi

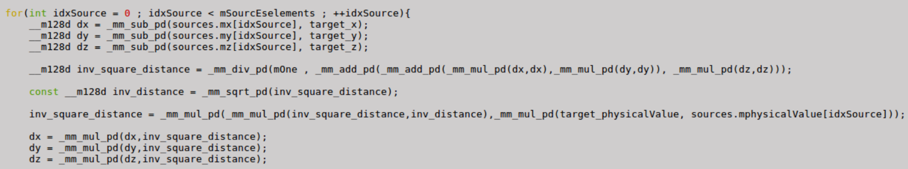
FMM on xeon phi

- Far field part (M2L): generally less efficiently implemented than near field part (P2P)
- Near field (P2P): direct N-body computation
- Based on vectorized code (sse) and/or OpenMP implemented



FMM on xeon phi: offloading attempt

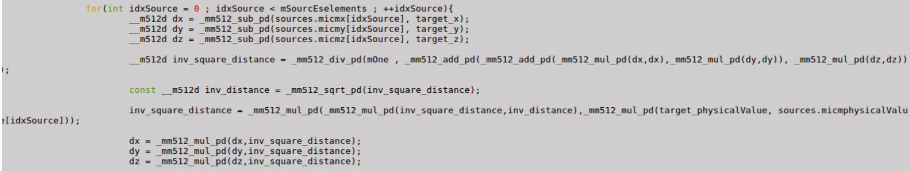
- sse to (similar) avx2 implementation
- OpenMP offloading compilation for the target pragmas
- Only applies to Intel® MIC Architecture



```
for(int idxSource = 0 ; idxSource < mSourcEselements ; ++idxSource){
    __m128d dx = _mm_sub_pd(sources.mx[idxSource], target_x);
    __m128d dy = _mm_sub_pd(sources.my[idxSource], target_y);
    __m128d dz = _mm_sub_pd(sources.mz[idxSource], target_z);

    __m128d inv_square_distance = _mm_div_pd(mOne , _mm_add_pd(_mm_add_pd(_mm_mul_pd(dx,dx),_mm_mul_pd(dy,dy)), _mm_mul_pd(dz,dz)));

    const __m128d inv_distance = _mm_sqrt_pd(inv_square_distance);

    inv_square_distance = _mm_mul_pd(_mm_mul_pd(inv_square_distance,inv_distance),_mm_mul_pd(target_physicalValue, sources.mphysicalValue[idxSource]));

    dx = _mm_mul_pd(dx,inv_square_distance);
    dy = _mm_mul_pd(dy,inv_square_distance);
    dz = _mm_mul_pd(dz,inv_square_distance);
```

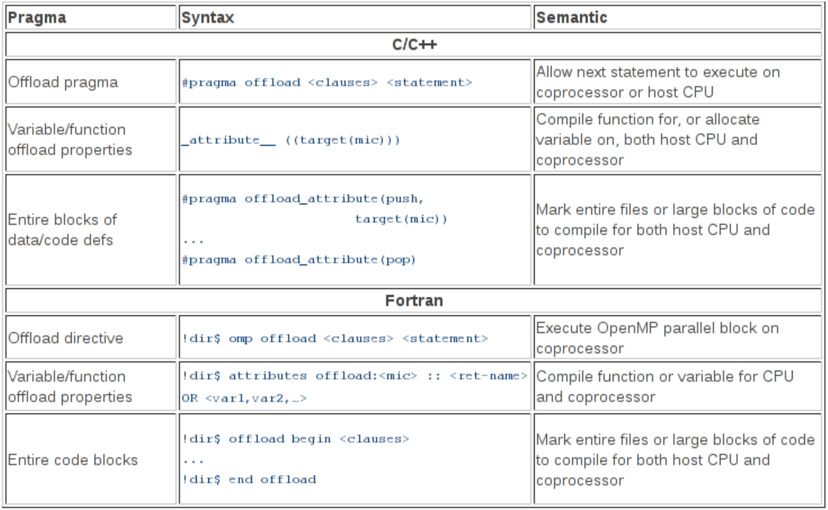
FMM on xeon phi: offloading attempt

- sse to (similar) avx2 implementation
- OpenMP offloading compilation for the target pragmas
- Only applies to Intel® MIC Architecture



```
                for(int idxSource = 0 ; idxSource < mSourcEselements ; ++idxSource){
                        __m512d dx = _mm512_sub_pd(sources.micmx[idxSource], target_x);
                        __m512d dy = _mm512_sub_pd(sources.micmy[idxSource], target_y);
                        __m512d dz = _mm512_sub_pd(sources.micmz[idxSource], target_z);

                        __m512d inv_square_distance = _mm512_div_pd(mOne , _mm512_add_pd(_mm512_add_pd(_mm512_mul_pd(dx,dx),_mm512_mul_pd(dy,dy)), _mm512_mul_pd(dz,dz))
);

                        const __m512d inv_distance = _mm512_sqrt_pd(inv_square_distance);

                        inv_square_distance = _mm512_mul_pd(_mm512_mul_pd(inv_square_distance,inv_distance),_mm512_mul_pd(target_physicalValue, sources.micmphysicalValu
e[idxSource]));

                        dx = _mm512_mul_pd(dx,inv_square_distance);
                        dy = _mm512_mul_pd(dy,inv_square_distance);
                        dz = _mm512_mul_pd(dz,inv_square_distance);
```

FMM on xeon phi: offloading attempt



Pragma	Syntax	Semantic
C/C++		
Offload pragma	`#pragma offload <clauses> <statement>`	Allow next statement to execute on coprocessor or host CPU
Variable/function offload properties	`_attribute__ ((target(mic)))`	Compile function for, or allocate variable on, both host CPU and coprocessor

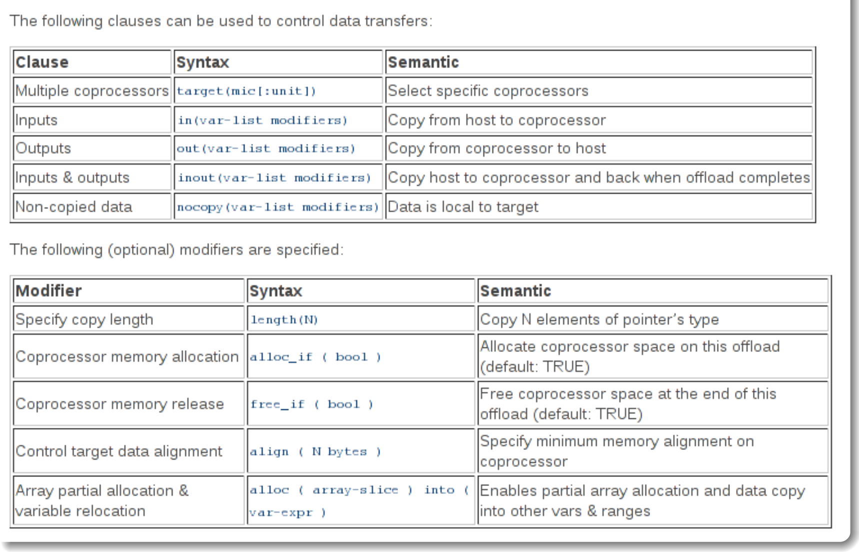
Entire blocks of data/code defs	`#pragma offload_attribute(push,`
	`target(mic))`
	`...`
	`#pragma offload_attribute(pop)`
		Mark entire files or large blocks of code to compile for both host CPU and coprocessor

Fortran		
Offload directive	`!dir$ omp offload <clauses> <statement>`	Execute OpenMP parallel block on coprocessor

Variable/function offload properties	`!dir$ attributes offload:<mic> :: <ret-name>`
	`OR <var1,var2,…>`
		Compile function or variable for CPU and coprocessor

Entire code blocks	`!dir$ offload begin <clauses>`
	`...`
	`!dir$ end offload`
		Mark entire files or large blocks of code to compile for both host CPU and coprocessor

FMM on xeon phi: offloading attempt



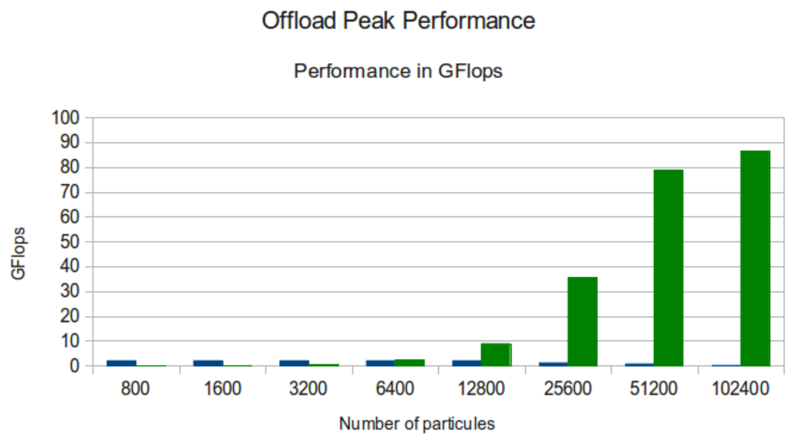
The following clauses can be used to control data transfers:

Clause	Syntax	Semantic
Multiple coprocessors	`target(mic[:unit])`	Select specific coprocessors
Inputs	`in(var-list modifiers)`	Copy from host to coprocessor
Outputs	`out(var-list modifiers)`	Copy from coprocessor to host
Inputs & outputs	`inout(var-list modifiers)`	Copy host to coprocessor and back when offload completes
Non-copied data	`nocopy(var-list modifiers)`	Data is local to target

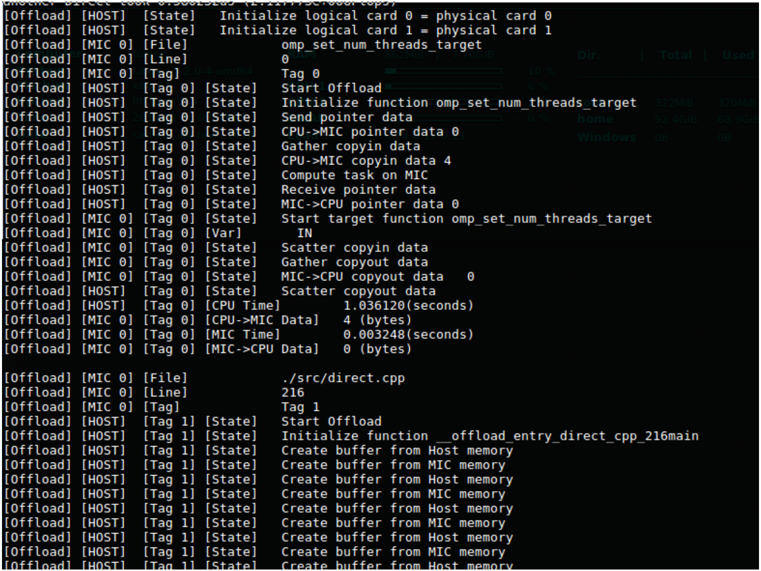
The following (optional) modifiers are specified:

Modifier	Syntax	Semantic
Specify copy length	`length(N)`	Copy N elements of pointer's type
Coprocessor memory allocation	`alloc_if ( bool )`	Allocate coprocessor space on this offload (default: TRUE)
Coprocessor memory release	`free_if ( bool )`	Free coprocessor space at the end of this offload (default: TRUE)
Control target data alignment	`align ( N bytes )`	Specify minimum memory alignment on coprocessor
Array partial allocation & variable relocation	`alloc ( array-slice ) into ( var-expr )`	Enables partial array allocation and data copy into other vars & ranges

FMM on xeon phi: offloading attempt

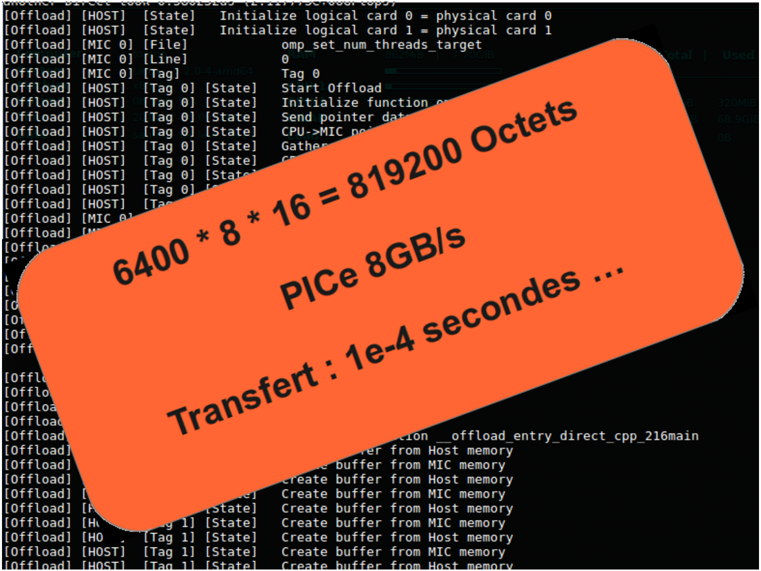


FMM on xeon phi: offloading attempt



```
[Offload] [HOST]  [State]   Initialize logical card 0 = physical card 0
[Offload] [HOST]  [State]   Initialize logical card 1 = physical card 1
[Offload] [MIC 0] [File]            omp_set_num_threads_target
[Offload] [MIC 0] [Line]            0
[Offload] [MIC 0] [Tag]             Tag 0
[Offload] [HOST]  [Tag 0] [State]   Start Offload
[Offload] [HOST]  [Tag 0] [State]   Initialize function omp_set_num_threads_target
[Offload] [HOST]  [Tag 0] [State]   Send pointer data
[Offload] [HOST]  [Tag 0] [State]   CPU->MIC pointer data 0
[Offload] [HOST]  [Tag 0] [State]   Gather copyin data
[Offload] [HOST]  [Tag 0] [State]   CPU->MIC copyin data 4
[Offload] [HOST]  [Tag 0] [State]   Compute task on MIC
[Offload] [HOST]  [Tag 0] [State]   Receive pointer data
[Offload] [HOST]  [Tag 0] [State]   MIC->CPU pointer data 0
[Offload] [MIC 0] [Tag 0] [State]   Start target function omp_set_num_threads_target
[Offload] [MIC 0] [Tag 0] [Var]       IN
[Offload] [MIC 0] [Tag 0] [State]   Scatter copyin data
[Offload] [MIC 0] [Tag 0] [State]   Gather copyout data
[Offload] [MIC 0] [Tag 0] [State]   MIC->CPU copyout data   0
[Offload] [HOST]  [Tag 0] [State]   Scatter copyout data
[Offload] [HOST]  [Tag 0] [CPU Time]        1.036120(seconds)
[Offload] [MIC 0] [Tag 0] [CPU->MIC Data]   4 (bytes)
[Offload] [MIC 0] [Tag 0] [MIC Time]        0.003248(seconds)
[Offload] [MIC 0] [Tag 0] [MIC->CPU Data]   0 (bytes)

[Offload] [MIC 0] [File]            ./src/direct.cpp
[Offload] [MIC 0] [Line]            216
[Offload] [MIC 0] [Tag]             Tag 1
[Offload] [HOST]  [Tag 1] [State]   Start Offload
[Offload] [HOST]  [Tag 1] [State]   Initialize function __offload_entry_direct_cpp_216main
[Offload] [HOST]  [Tag 1] [State]   Create buffer from Host memory
[Offload] [HOST]  [Tag 1] [State]   Create buffer from MIC memory
[Offload] [HOST]  [Tag 1] [State]   Create buffer from Host memory
[Offload] [HOST]  [Tag 1] [State]   Create buffer from MIC memory
[Offload] [HOST]  [Tag 1] [State]   Create buffer from Host memory
[Offload] [HOST]  [Tag 1] [State]   Create buffer from MIC memory
[Offload] [HOST]  [Tag 1] [State]   Create buffer from Host memory
[Offload] [HOST]  [Tag 1] [State]   Create buffer from MIC memory
[Offload] [HOST]  [Tag 1] [State]   Create buffer from Host memory
```

FMM on xeon phi: offloading attempt



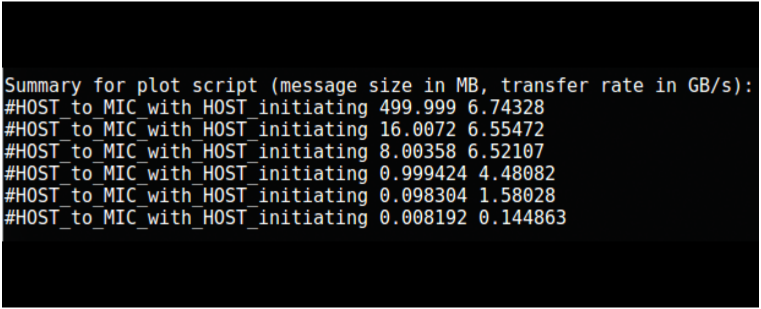
```
[Offload] [HOST]  [State]   Initialize logical card 0 = physical card 0
[Offload] [HOST]  [State]   Initialize logical card 1 = physical card 1
[Offload] [MIC 0] [File]            omp_set_num_threads_target
[Offload] [MIC 0] [Line]            0
[Offload] [MIC 0] [Tag]             Tag 0
[Offload] [HOST]  [Tag 0] [State]   Start Offload
[Offload] [HOST]  [Tag 0] [State]   Initialize function o
[Offload] [HOST]  [Tag 0] [State]   Send pointer dat
[Offload] [HOST]  [Tag 0] [State]   CPU->MIC po
[Offload] [HOST]  [Tag 0] [State]   Gather
[Offload] [HOST]  [Tag 0] [State]   C
[Offload] [HOST]  [Tag 0] [State
[Offload] [HOST]  [Tag 0]
[Offload] [HOST]  [Ta
[Offload] [MIC 0
[Offload] [M
[Offlo
[O
[
[
[O
[Of
[Of
[Off

[Offl
[Offlo
[Offloa
[Offload
[Offload                             tion __offload_entry_direct_cpp_216main
[Offload]                           fer from Host memory
[Offload]                        e buffer from MIC memory
[Offload]                       Create buffer from Host memory
[Offload] [                  e]   Create buffer from MIC memory
[Offload] [H             [State]   Create buffer from Host memory
[Offload] [H        g 1] [State]   Create buffer from MIC memory
[Offload] [HO      [Tag 1] [State]   Create buffer from Host memory
[Offload] [HOST]  [Tag 1] [State]   Create buffer from MIC memory
[Offload] [HOST]  [Tag 1] [State]   Create buffer from Host memory
```

6400 * 8 * 16 = 819200 Octets

PICe 8GB/s

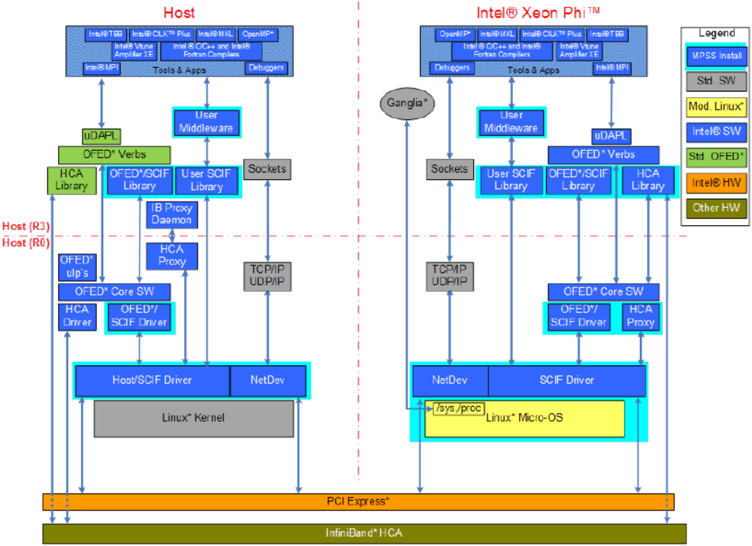
Transfert : 1e-4 secondes ...

FMM on xeon phi: driver version

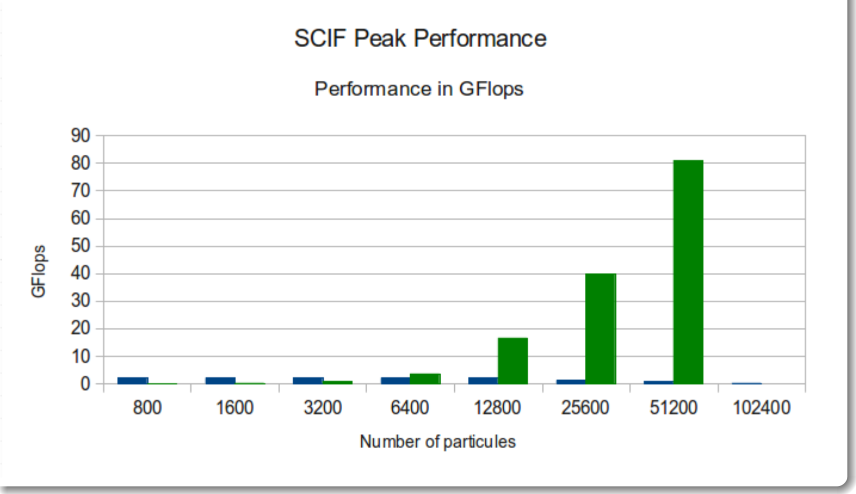


```
Summary for plot script (message size in MB, transfer rate in GB/s):
#HOST_to_MIC_with_HOST_initiating 499.999 6.74328
#HOST_to_MIC_with_HOST_initiating 16.0072 6.55472
#HOST_to_MIC_with_HOST_initiating 8.00358 6.52107
#HOST_to_MIC_with_HOST_initiating 0.999424 4.48082
#HOST_to_MIC_with_HOST_initiating 0.098304 1.58028
#HOST_to_MIC_with_HOST_initiating 0.008192 0.144863
```

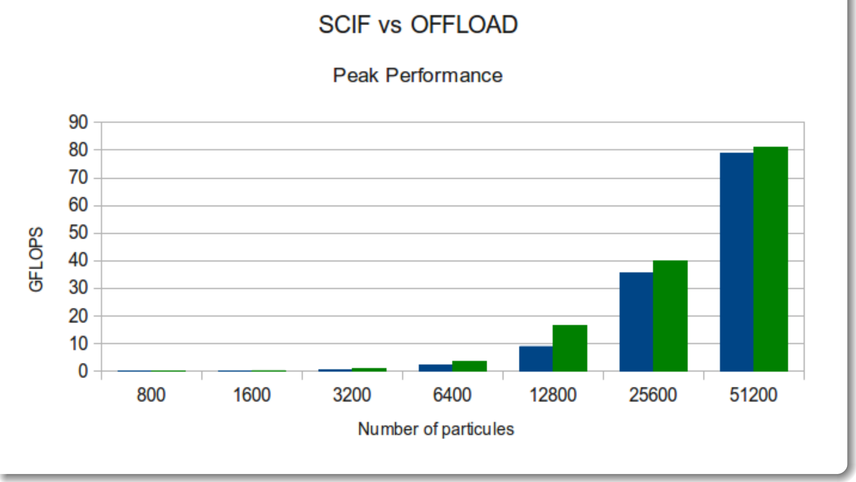
FMM on xeon phi: driver version



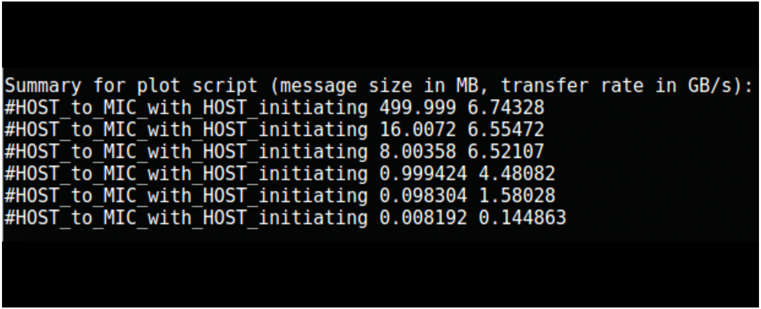
FMM on xeon phi: driver version



FMM on xeon phi: driver version

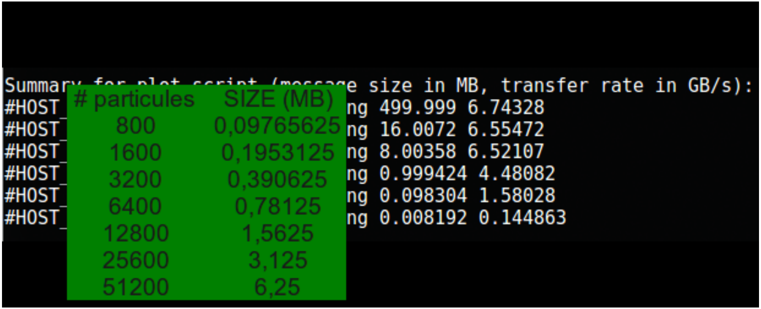


FMM on xeon phi: driver version



```
Summary for plot script (message size in MB, transfer rate in GB/s):
#HOST_to_MIC_with_HOST_initiating 499.999 6.74328
#HOST_to_MIC_with_HOST_initiating 16.0072 6.55472
#HOST_to_MIC_with_HOST_initiating 8.00358 6.52107
#HOST_to_MIC_with_HOST_initiating 0.999424 4.48082
#HOST_to_MIC_with_HOST_initiating 0.098304 1.58028
#HOST_to_MIC_with_HOST_initiating 0.008192 0.144863
```

FMM on xeon phi: driver version



```
Summary for plot script (message size in MB, transfer rate in GB/s):
#HOST_                          ng 499.999 6.74328
#HOST_                          ng 16.0072 6.55472
#HOST_                          ng 8.00358 6.52107
#HOST_                          ng 0.999424 4.48082
#HOST_                          ng 0.098304 1.58028
#HOST_                          ng 0.008192 0.144863
```

# particules	SIZE (MB)
800	0,09765625
1600	0,1953125
3200	0,390625
6400	0,78125
12800	1,5625
25600	3,125
51200	6,25

4

conclusion