Pôle 3 / RUNTIME

Action d'envergure C2S@Exa

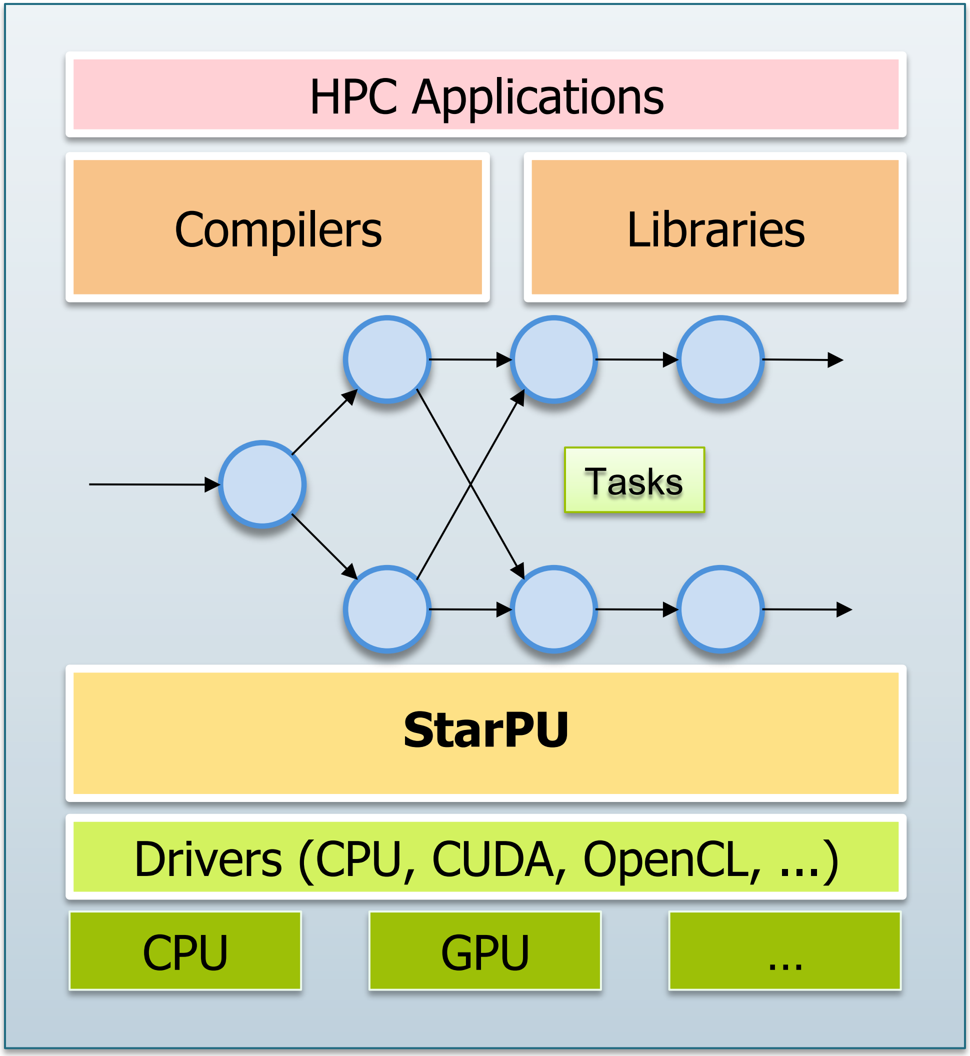
Olivier Aumage
EPI RUNTIME
Inria Bordeaux – Sud-Ouest

Sophia
12/2013

Overview of StarPU

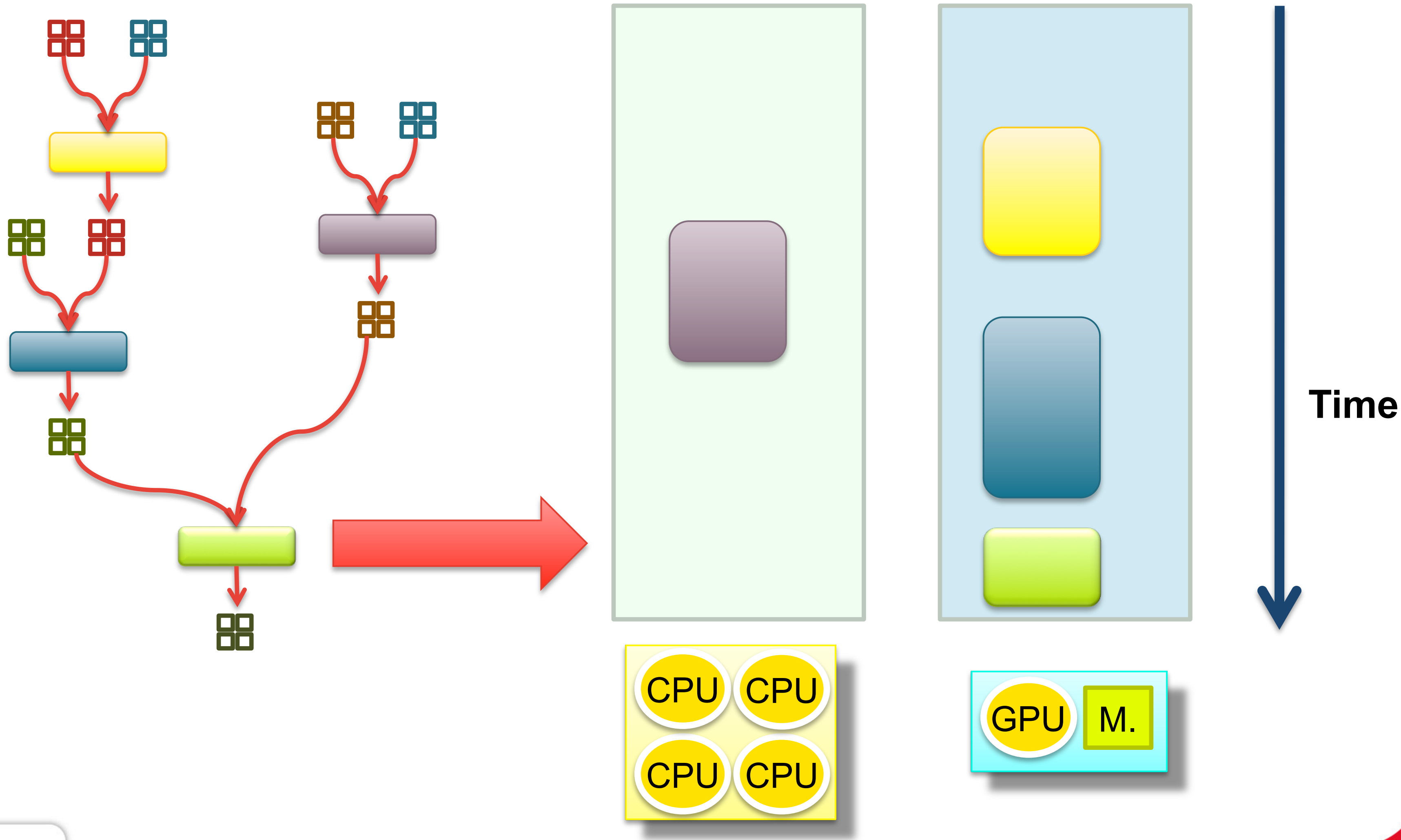
Information needed from applications

- Tasks
  - Implementation(s)
    - CPU
      - Regular
      - MMX, SSE, AVX, ...
    - Cuda, OpenCL
- Data
  - Type, layout
    - Vector, matrix, ...
  - Partioning
- Task/Data Relationships
  - Dependencies
    - R, W, R/W, reduction, ...
- (Optional) Custom Scheduler
  - Open Scheduling Platform

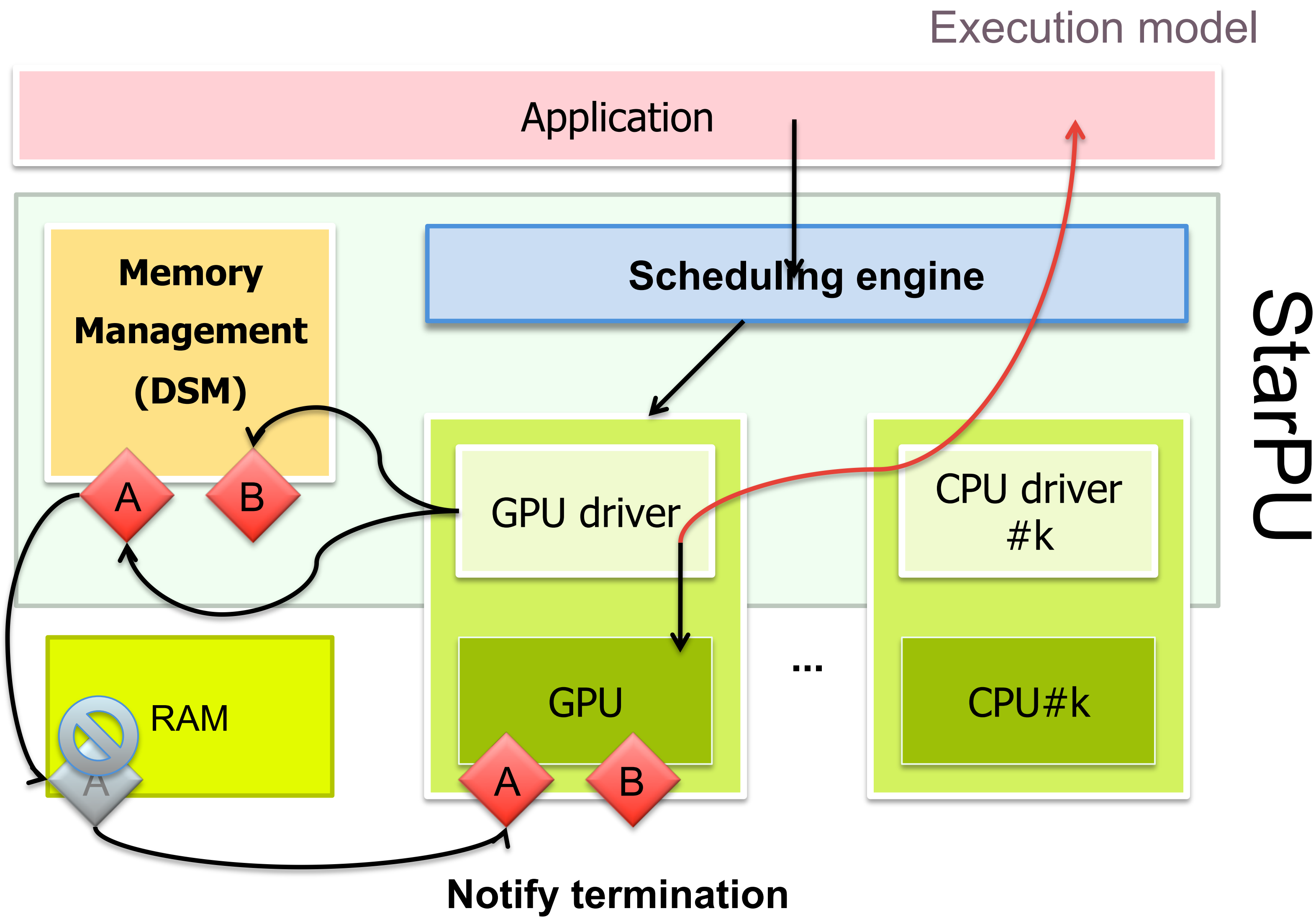


Heterogeneous Task Scheduling

Mapping a graph of tasks (DAG) on a heterogeneous platform



The StarPU runtime system



Recent and On-going Developments

- StarPU / MPI cooperation
  - Parallel vs distributed relationship support
- Out of core
  - Disks as memory nodes
- Support for Manycores
  - Intel Xeon PHI, Intel SCC
- StarPU as an OpenCL backend
  - Use StarPU from OpenCL API
- Composition
  - Scheduling contexts

StarPU interfacing with MPI

- Automatically handles task/communications dependencies
  - Automatically handles all needed CPU/GPU transfers
- Automatically overlaps MPI communications, CPU/GPU communications, and CPU/GPU computations
  - Thanks to the data transfer requests mechanism
- StarPU/MPI support could be useful to XcalableMP/StarPU port in simplifying the interface between node-local management and distributed management
- People
  - Samuel Thibault, Nathalie Furmento, Olivier Aumage (permanents)
  - Marc Sergent (Master M2 completed, now PhD 1st year)

Out of core

Using disks as StarPU memory nodes

- Large workloads
  - Enable StarPU to evict temporarily unused data to disk
- Integration with general StarPU's memory management layer
  - StarPU data handles
- Multiple disk drivers supported
  - Legacy stdio/unistd methods
  - Google's LevelDB (key/value database library)
- Readily available in StarPU's SVN repository for the FP3C project
- People
  - Samuel Thibault (permanent)
  - M1 internship completed, proposal for a M2 internship

Support for Manycores

Intel Xeon Phi & SCC, Kalray MPPA

- Common support for shared aspects of Xeon Phi and SCC
  - Source = the main CPU node
  - Sink = the manycore device node

- Scheduling performed by the main StarPU instance

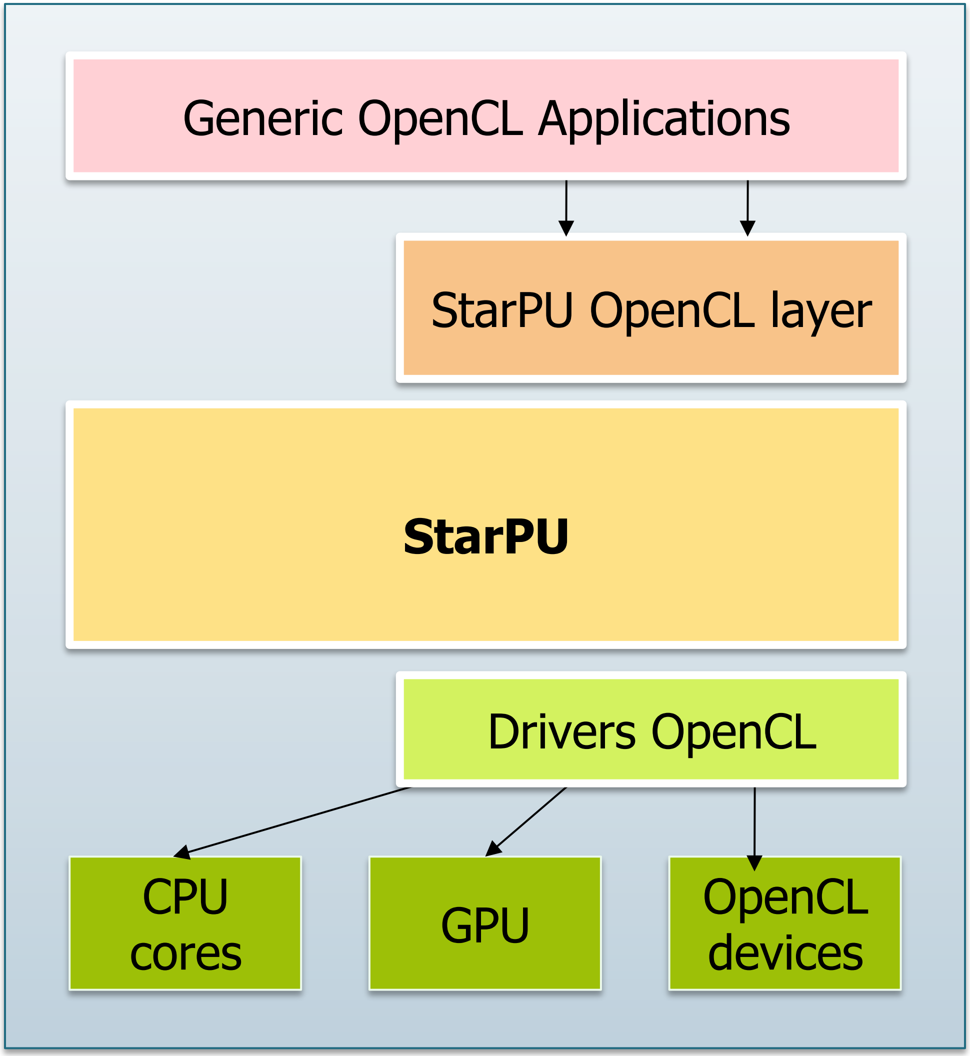
- Tested successfuly on University of Tokyo's KNCC cluster of Intel MICs, within the context of FP3C project

- People
  - Samuel Thibault
  - Completed M1 internships (Intel Xeon Phi, SCC, and also Kalray MPPA)

StarPU as backend for OpenCL

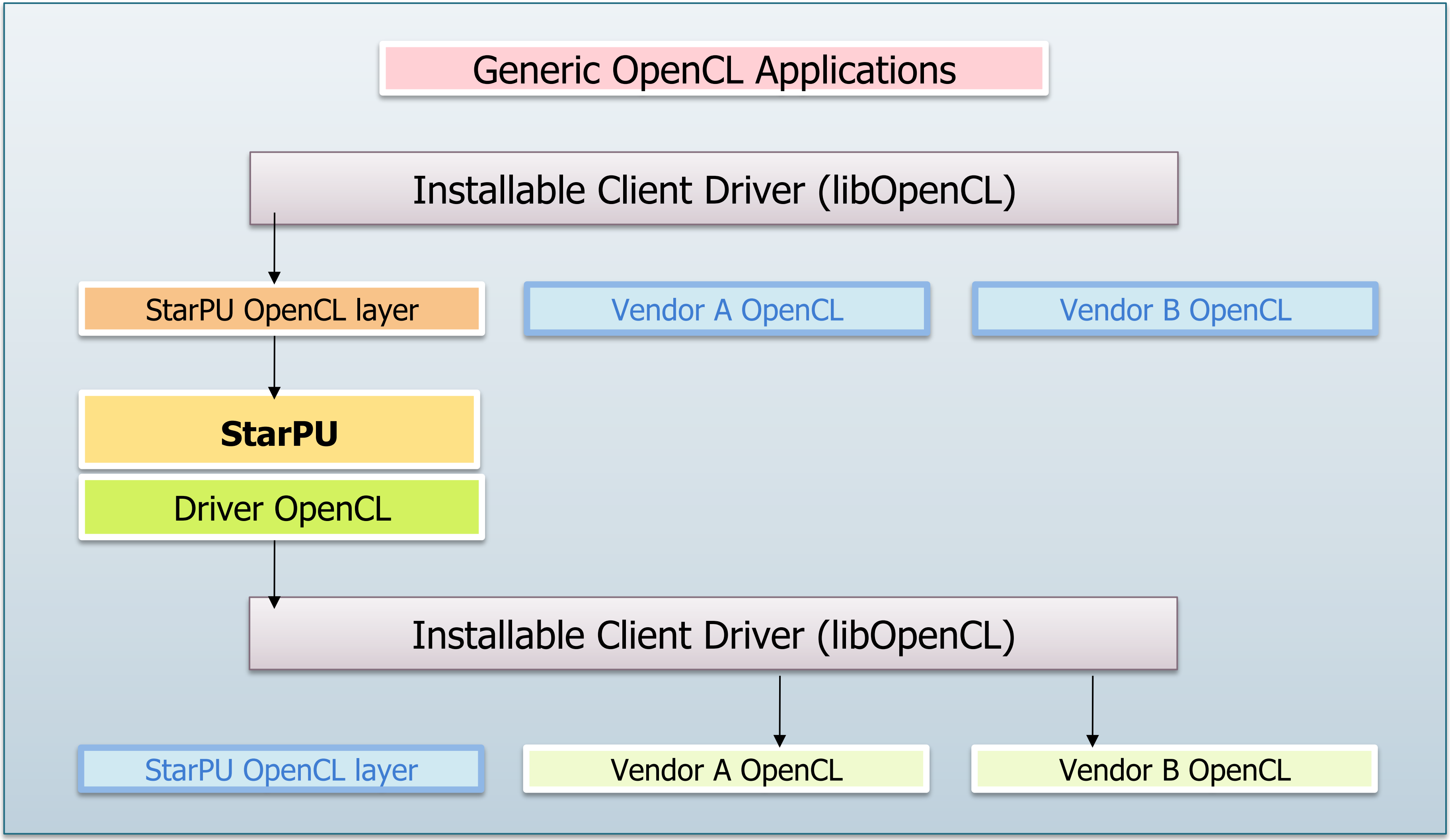
Portably programming StarPU through OpenCL

- Run generic OpenCL codes on top of StarPU



Built on OpenCL's ICD extension (`cl_khr_icd`)

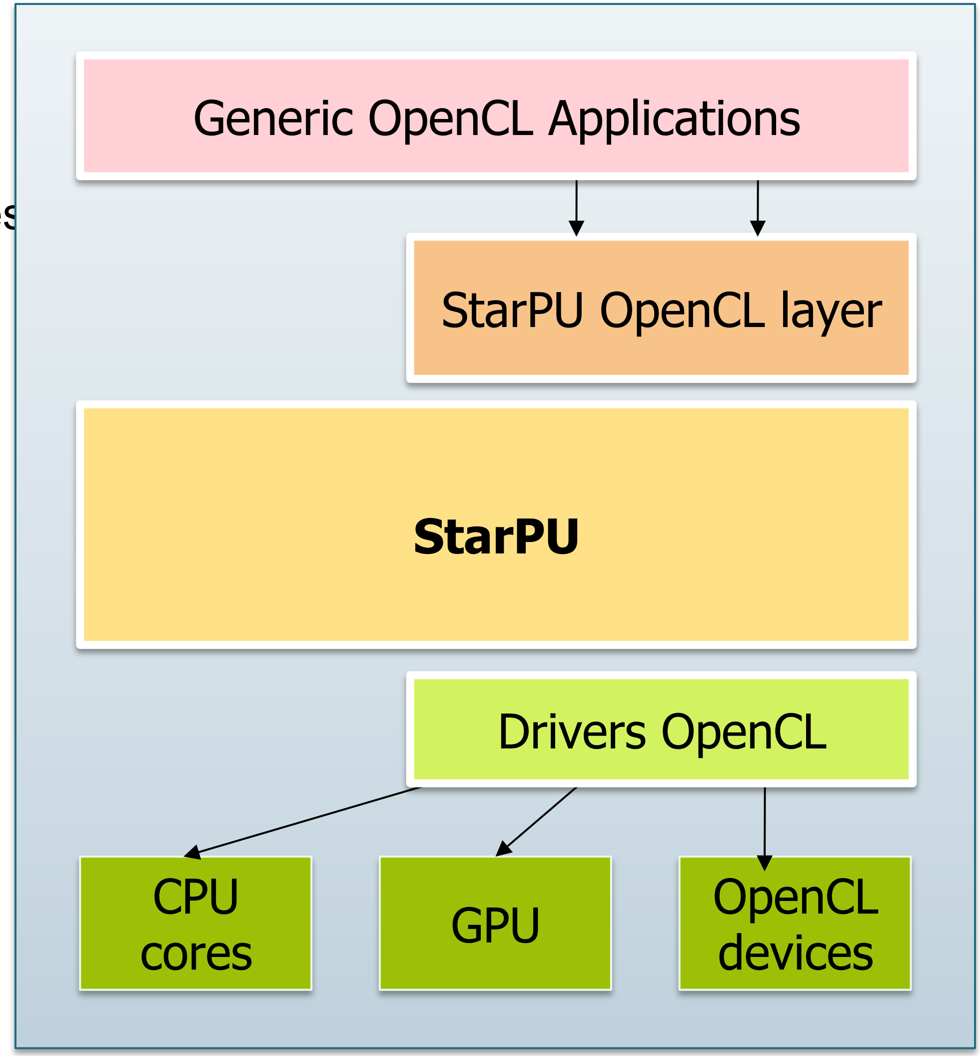
ICD = Installable Client Driver



StarPU as backend for OpenCL

Portably programming StarPU through OpenCL

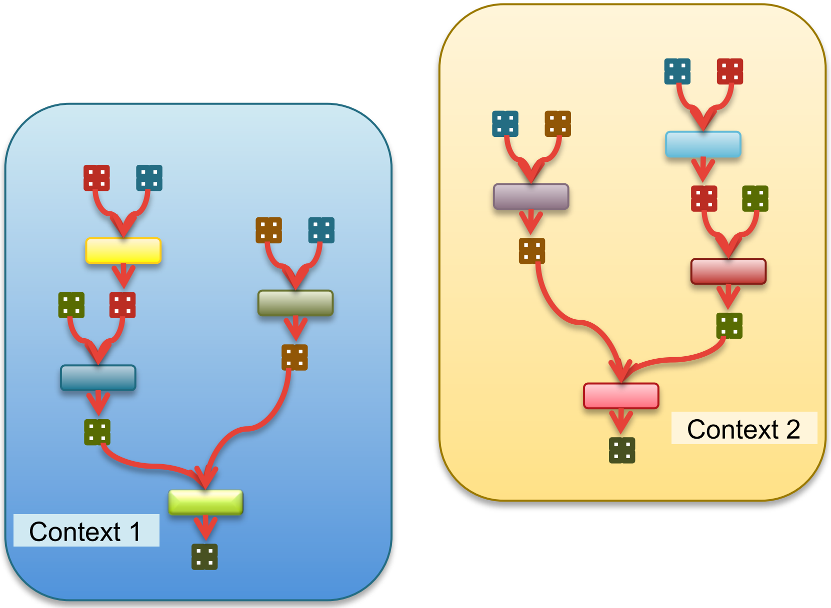
- Run generic OpenCL codes on top of StarPU
- Events can be used between all devices
  - Synchronizations
- Buffers can be shared by all devices
  - Data transfers handled by StarPU
- Compatibility with StarPU's contexts
  - Dedicated command queues
  - Dedicated schedulers
- People
  - Sylvain Henry, PhD completed Nov, 2013



Composition

Scheduling Contexts

- Managing computing resources for multiple kernels simultaneously
  - Map kernels on subsets of computing units
  - Isolate competing kernels or library calls
    - OpenMP kernel, Intel MKL, etc.
  - Select scheduling policy per context
- Two flavors
  - Explicit worker set mapping
    - Initial worker set
    - Worker add/remove ops
  - Managed worker set
    - Hypervisor



- Compatible with StarPU as OpenCL backend
- Can be use to interoperate XcalableMP codes with existing OpenCL codes

- People
  - Raymond Namyst, Pierre-André Wacrenier, Abdou Guermouche
  - CDD: Andra HUGO, Phd. Thesis (starting 3rd year)

Actions

ADT conjointe MOAIS (Pole 4) – RUNTIME (Pole 3)

- ADT « K'Star »
  - Portée par Thierry Gautier, EPI MOAIS
  - Inria Grenoble (Thierry Gautier) + Inria Bordeaux (Olivier Aumage, Samuel Thibault, Nathalie Furmento)
- Compilation *source-to-source* d'un langage standard...
  - OpenMP 4.0
- ... vers des routines XKaapi ou StarPU

- Objectif pour l'AE HPC
  - Faciliter l'adaptation portable des codes au-dessus des supports exécutifs
- Objectif pour XKaapi & StarPU
  - Simplifier l'utilisation
  - Augmenter la visibilité
- CDD
  - Pierrick Brunet, ingénieur (2 ans) à Montbonnot
  - + 1 ingénieur IJD (1an) à recruter à Bordeaux lors de la campagne 2014

Actions

ANR SOLHAR (Pole 1 + Pole 3)

- ANR « SOLHAR »
  - Portée par Abdou Guermouche, EPI HIEPACS
  - Inria Bordeaux (HIEPACS, RUNTIME), Inria Rhone-Alpes (ROMA)
  - CNRS, EADS, CEA
- Solveurs directs en algèbre linéaire creuse
- ... au dessus du support exécutif **StarPU**

- Intérêt pour l'AE HPC
  - Mise en œuvre directe du modèle de l'AE
    - Runtime + Solveur + Méthode numérique + Application

- CDD, 1 nouveau doctorant sur la thématique de l'adaptation de granularité des tâches

- Projet Hi-BOX (DGA), présenté par Guillaume Sylvand
  - EADS, IMACS, INRIA HIEPACS & RUNTIME

- Projet connexe ANR ANEMOS
  - INRIA HIEPACS (Pierre Ramet, Xavier Lacoste), CEA (Guillaume Latu)
  - ➢ Nous sommes intéressés pour distribuer les tâches de la nouvelle version « light » de Gysela sur CPU+MIC avec StarPU !
  - ➢ Voir également le logiciel Hwloc pour les questions d'affinité threads/mémoire

En résumé, au niveau de StarPU

2 axes de travail

- Faciliter l'adaptation de codes au-dessus de StarPU
  - Programmation par OpenCL et OpenMP 4.0 : avec pôle 4
  - Coopérations avec les solveurs (algèbre linéaire) : avec pôle 1

- Apporter de nouvelles fonctionnalités
  - Ordonnancement de tâches sur machines distribuées et many-cores
    - exploiter des clusters hétérogènes
  - Out-of-core
    - travailler sur de grands volumes de données
  - Composition
    - associer des codes ou des kernels parallèles
  - Granularité
    - adapter la charge de calcul demandée aux ressources de calcul disponibles

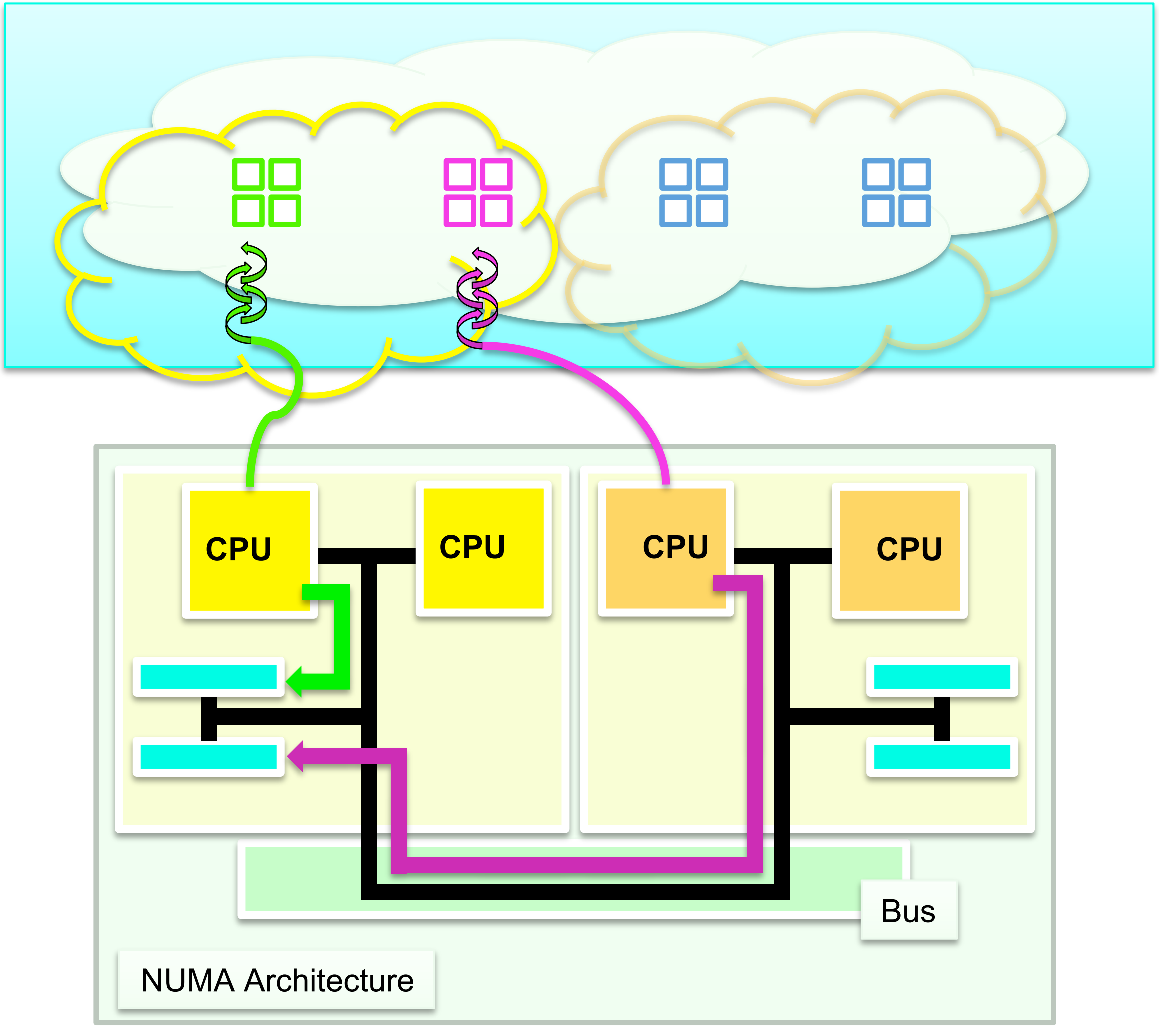
- Interlocuteurs au niveau de RUNTIME
  - Raymond Namyst, Olivier Aumage

Hwloc – Hardware Locality

Discovering Hardware Topology
Binding Computing Resources

Olivier Aumage
RUNTIME group

Why is thread binding needed?



Why thread binding may be difficult?

- Lack of generic, uniform interface
  - OS specific
  - Distribution specific
  - Low-level APIs
- Undocumented, non deterministic behaviour
  - Resource identification scheme
  - Identification stability over time
    - BIOS dependent settings
    - Reboot dependent settings
- Evolving technology
  - E.g.: AMD Bulldozer “half-cores”, Intel SCC “tiles” (2 cores, Hz) & groups (8 cores, V)
- Need generic tools & abstract API
  - Logical resource identification

Hwloc

Hardware Locality

- Brice Goglin, Samuel Thibault, EPI RUNTIME
- Joint development
  - Runtime Group + Open MPI/Cisco
    - libtopology
    - PLPA
- Two parts
  - Set of command line tools
    - lstopo
    - hwloc-bind, hwloc-calc, etc.
  - API + library

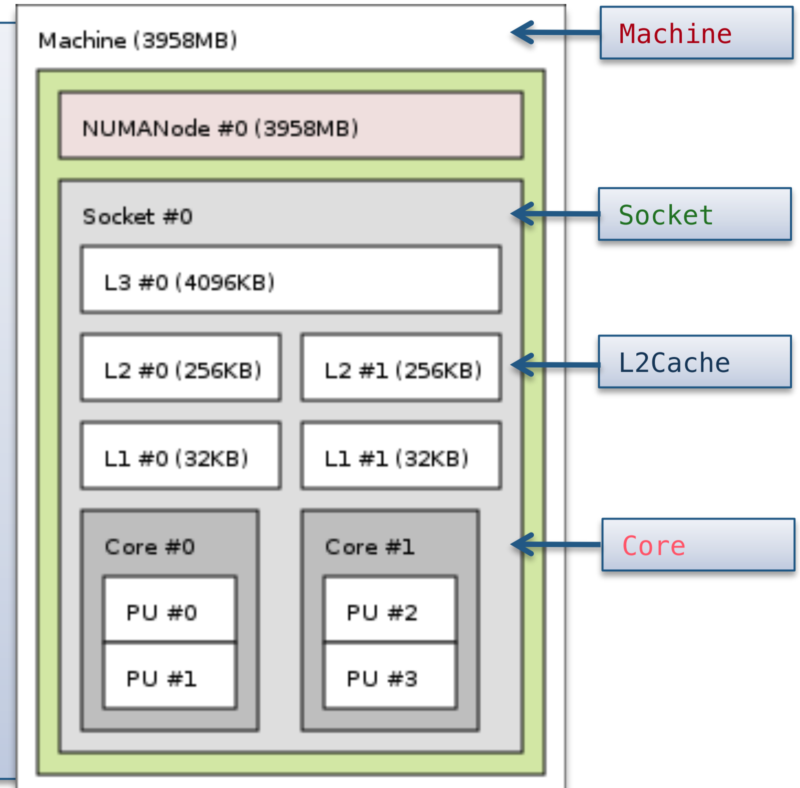
- `http://www.open-mpi.org/projects/hwloc/`

Hwloc – Lstopo

Displays the hierarchical topology map of the current system

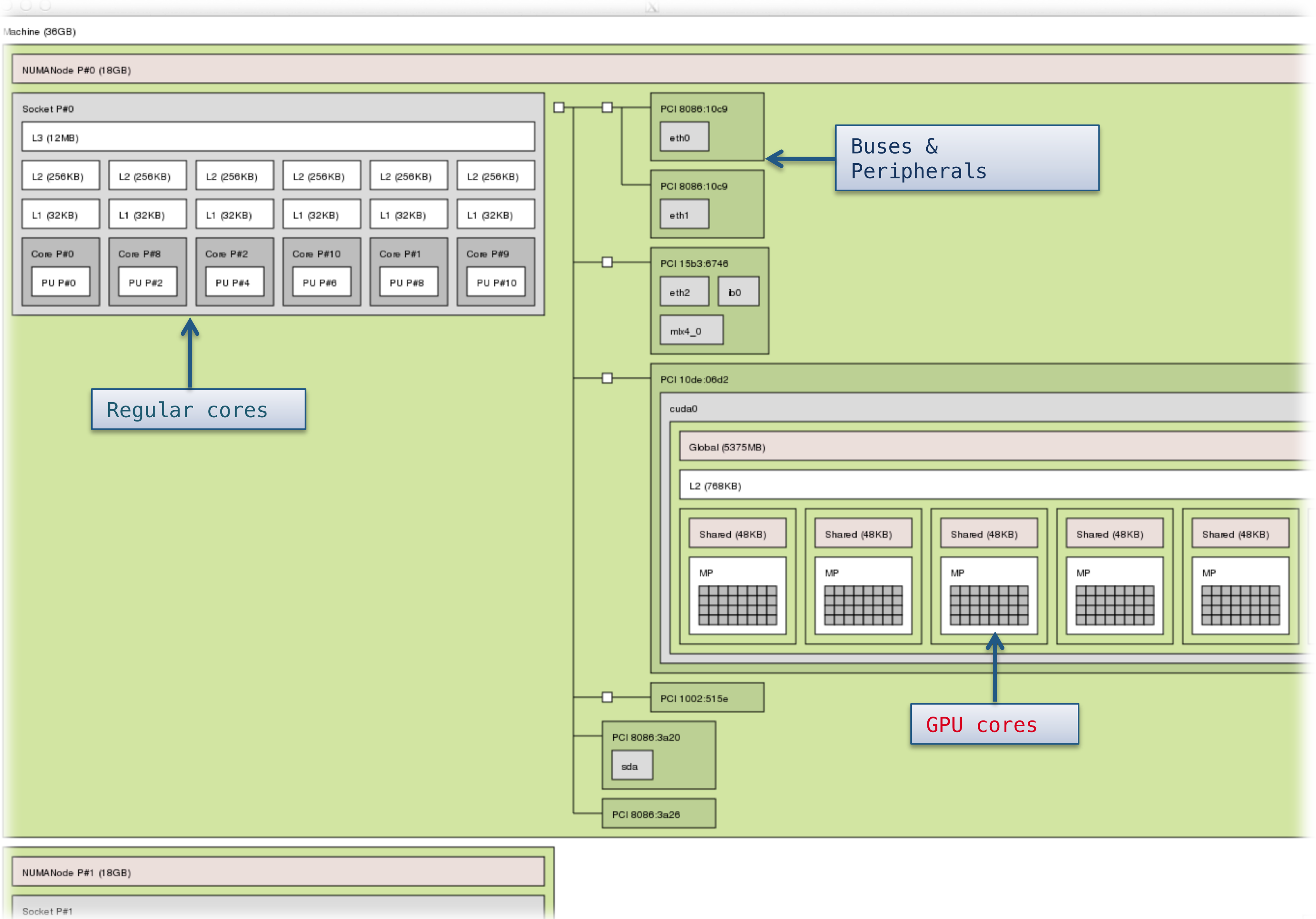
```
Machine                 (total=4052948KB)
 NUMANode #0            (phys=0 local=4052948KB)

   Socket #0            (phys=0)
    L3Cache #0          (4096KB)

      L2Cache #0        (256KB)
      |L1Cache #0       (32KB)
      | Core #0         (phys=0)
      |   PU #0         (phys=0)
      |   PU #1         (phys=2)
      |
      L2Cache #1        (256KB)
       L1Cache #1       (32KB)
        Core #1         (phys=2)
          PU #2         (phys=1)
          PU #3         (phys=3)
```

Textual representation



Graphical representation

Hwloc – Lstopo



Hwloc – `hwloc-*`

Other command line tools

- hwloc-bind
  - Binds processes to specific hardware objects
- hwloc-calc
  - Creates bitmap strings to pass to hwloc-bind
- hwloc-distrib
  - Generates a set of uniformly distributed bitmap strings
- hwloc-ps
  - Displays the bindings of currently running processes
- hwloc-gather-topology
  - Saves the relevant topology files of the current machine
  - Allows offline simulation or debugging
  - Linux only

Hwloc – API

Programming interface, run-time library

- Topology information
  - CPU, caches, memory, NUMA nodes, I/O devices
  - Query, traversal
- CPU binding
  - Processes binding
  - Threads binding
- Memory binding
  - Get/set memory binding policy
  - Allocation of memory bound to specific nodes
- Bitmap API
  - Sets of resources
  - Common set operations