
# Pole 3 Optimization of Performance

## *Runtime Support*

**IPL C2S@Exa Mid-Term Evaluation**

*Olivier Aumage*, RUNTIME / STORM Team
INRIA BORDEAUX – SUD-OUEST

Jan. 13, 2015

# Pole 3

## Optimization of Performance

- Numerical data set processing support
  - Partitioning, repartitioning, re-meshing
  - See F. Pellegrini talk
- **Runtime support**
  - Running and scheduling applications on heterogeneous platforms

# Pole 3 / Runtime Support

## Context and objectives

- Task scheduling on heterogeneous platforms
  - Accelerated nodes, clusters
- Distributed data management
- Scalability
- Portability of performance

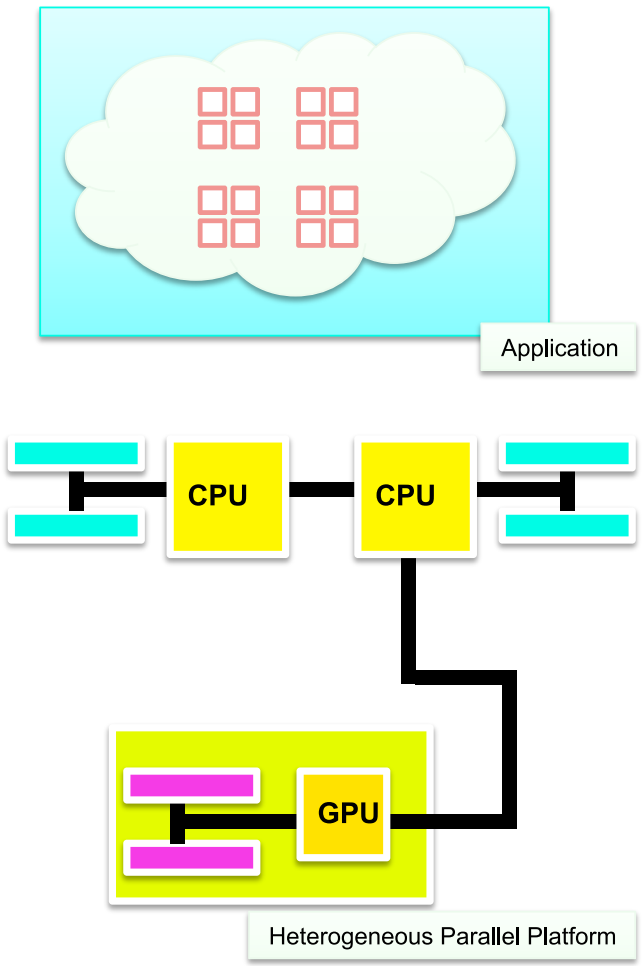
# Heterogeneous Parallel Platforms

Heterogeneous Association

- General purpose processor
- Specialized accelerator

Generalization

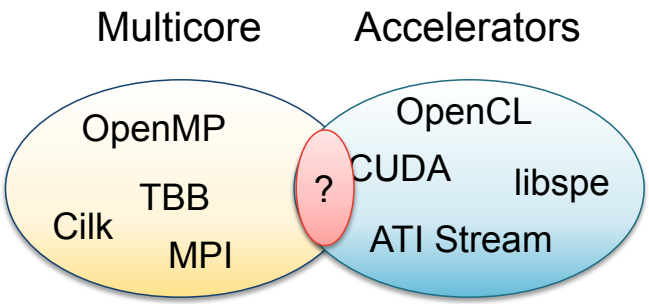
- Combination of various units
  - Latency-optimized cores
  - Throughput-optimized cores
  - Energy-optimized cores
- Distributed cores
  - Standalone GPUs
  - Intel Xeon Phi (MIC)
  - Intel Single-Chip Cloud (SCC)
- Integrated cores
  - Intel Haswell
  - AMD Fusion
  - nVidia Tegra

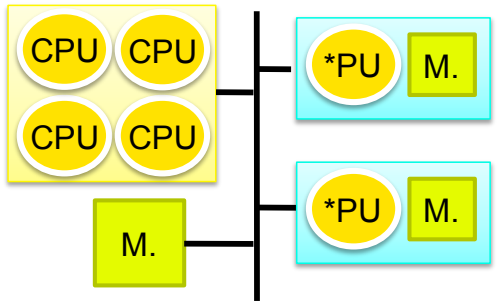


# Runtime Support for Heterogeneous Platforms?

- Multicores
  - pthreads, OpenMP, TBB, ...
- Accelerators
  - Consensus on OpenCL?
  - (Often) Pure offloading model

- Hybrid models?
  - The **StarPU** runtime system



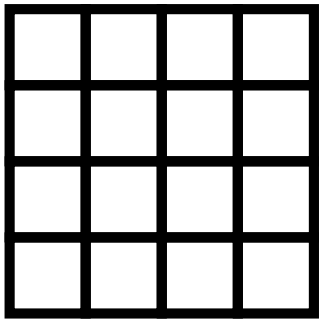
# StarPU Programming Model: Sequential Task Submission

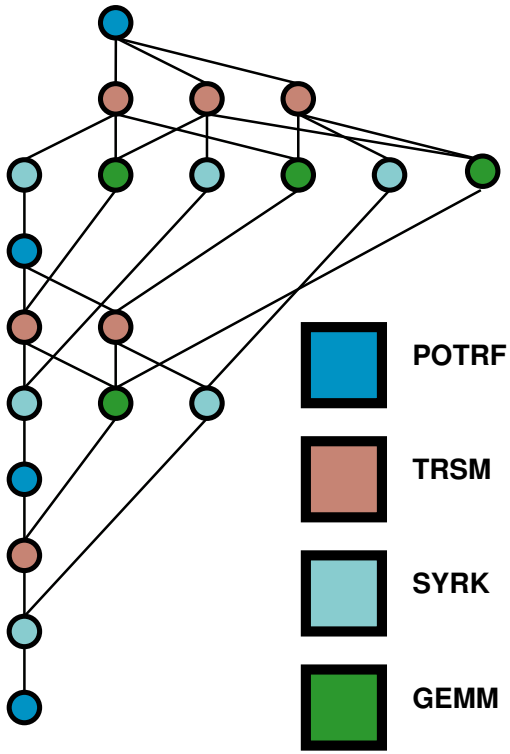
- Express parallelism...
- ... using the natural program flow

- **Submit** tasks asynchronously, in the sequential order of the program...
- ... let the runtime schedule the tasks in parallel on heterogeneous computing units

# Ex.: Sequential **Task-Based** Cholesky Decomposition

```
for (j = 0; j < N; j++) {
  POTRF (RW,A[j][j]);
  for (i = j+1; i < N; i++)
    TRSM (RW,A[i][j], R,A[j][j]);
  for (i = j+1; i < N; i++) {
    SYRK (RW,A[i][i], R,A[i][j]);
    for (k = j+1; k < i; k++)
      GEMM (RW,A[i][k],
            R,A[i][j], R,A[k][j]);
  }
}
__wait__();
```






- Kernel tasks submitted asynchronously
- Data dependences determined implicitely
- A graph of tasks is built
- The graph of tasks is executed

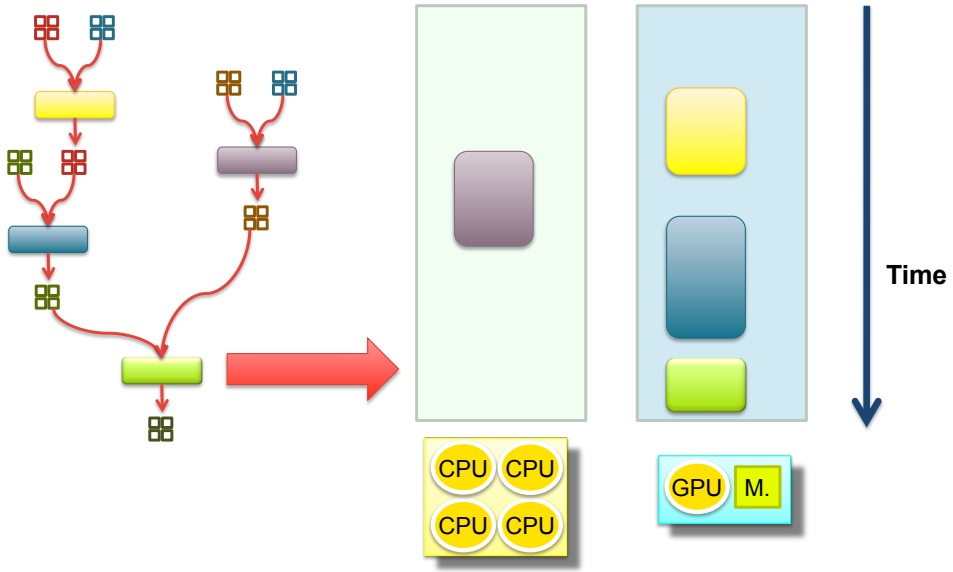
# StarPU Execution Model: Task Scheduling

Mapping the graph of tasks (DAG) on the hardware

- Allocating computing resources
- Enforcing dependency constraints
- Handling data transfers

Adaptiveness

- A single DAG enables multiple schedulings
- A single DAG can be mapped on multiple platforms



# Development and Results

## Programming Support

- OpenMP 4.0 compiler: Klang-OMP
  - Inria RUNTIME and MOAIS
- OpenCL back-end
  - Planned cooperation Inria RUNTIME and TONUS

## Platform Support

- Distributed Computing: StarPU-MPI
  - Inria RUNTIME and HIEPACS, CEA CESTA
- Out-of-core support
- Simulation support
  - Inria RUNTIME, MESCAL and CEPAGE/REALOPT

## Scheduling Support

- Composition and scheduling contexts
  - Inria RUNTIME and HIEPACS
- Component models
  - Inria RUNTIME and AVALON, CEA Cadarache, Maison de la Simulation

# Klang-omp OpenMP C/C++ Compiler

**Translate directives into runtime system API calls**

- StarPU Runtime System
- XKaapi Runtime System (INRIA Team MOAIS)
- See also Thierry Gautier's talk (MOAIS, Pole 4)

# Klang-omp OpenMP C/C++ Compiler

**Translate directives into runtime system API calls**

- StarPU Runtime System
- XKaapi Runtime System (INRIA Team MOAIS)
- See also Thierry Gautier's talk (MOAIS, Pole 4)

**ADT K'Star**

- Engineer Pierrick Brunet (start 9/2013, 2 year, Montbonnot)
- Engineer Philippe Virouleau (end 11/2014, 1 year, Montbonnot)
- Engineer Samuel Pitoiset (start 11/2014, 1 year, Bordeaux)
  - port of Inria HIEPACS ScalFMM application on top of the compiler

# Klang-omp OpenMP C/C++ Compiler

**Translate directives into runtime system API calls**

- StarPU Runtime System
- XKaapi Runtime System (INRIA Team MOAIS)
- See also Thierry Gautier's talk (MOAIS, Pole 4)

**ADT K'Star**

- Engineer Pierrick Brunet (start 9/2013, 2 year, Montbonnot)
- Engineer Philippe Virouleau (end 11/2014, 1 year, Montbonnot)
- Engineer Samuel Pitoiset (start 11/2014, 1 year, Bordeaux)
  - port of Inria HIEPACS ScalFMM application on top of the compiler

**Status**

- LLVM-based source-to-source compiler
  - Builds on open source Intel compiler `clang-omp`
- OpenMP 3.1
  - Virtually full support
- OpenMP 4.0
  - Dependent tasks
  - Heterogeneous targets (on-going work)
- K'Star project website – **http://kstar.gforge.inria.fr/**

# SOCL Layer – StarPU as an OpenCL Backend

## SOCL Rationale

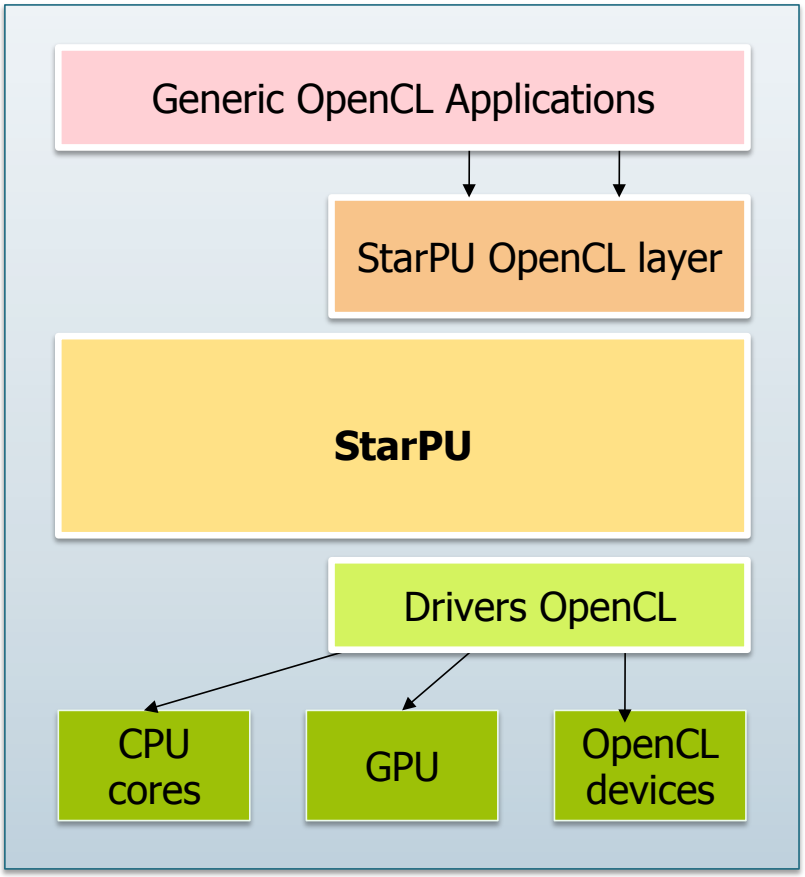
- Run generic OpenCL codes. . .
- . . . on top of StarPU

## Technical details

- StarPU as an OpenCL backend
  - ICD: Installable Client Driver
- Redirects OpenCL calls. . .
- . . . to StarPU routines

## Kernels

- SOCL can itself use OpenCL Kernels




## Partnership

- Planned cooperation with Inria Team TONUS (Pole 2) to port the CLAC code on SOCL

# Distributed Computing: StarPU-MPI

Summary: Interoperability between StarPU and MPI

- On-going work
- Ph.D thesis Marc Sergent (INRIA RUNTIME + CEA CESTA, Region Aquitaine Grant)

Related partnerships and works

- ADT HPC Collective (engineer Florent Pruvost)
- MORSE associated team (Inria HIEPACS, Inria RUNTIME, UTK)
  - Chameleon library port on top of StarPU-MPI
- ANR SOLHAR
- DGA RAPID Hi-BOX with Airbus Group and Imacs

# Distributed Computing: StarPU-MPI

## Extending StarPU's Paradigm on Clusters

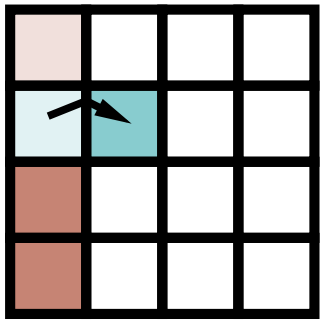
No global scheduler

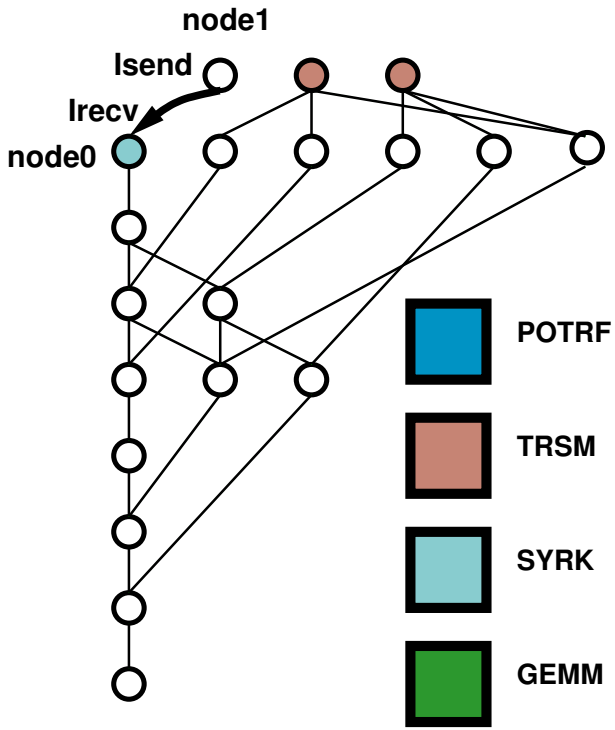
Task $\leftrightarrow$ Node Mapping

- Provided by the application
- Can be altered dynamically

Communications

- Inferred from the task graph
  - **Dependencies**
- Automatic Isend and Irecv calls





# Out-of-Core

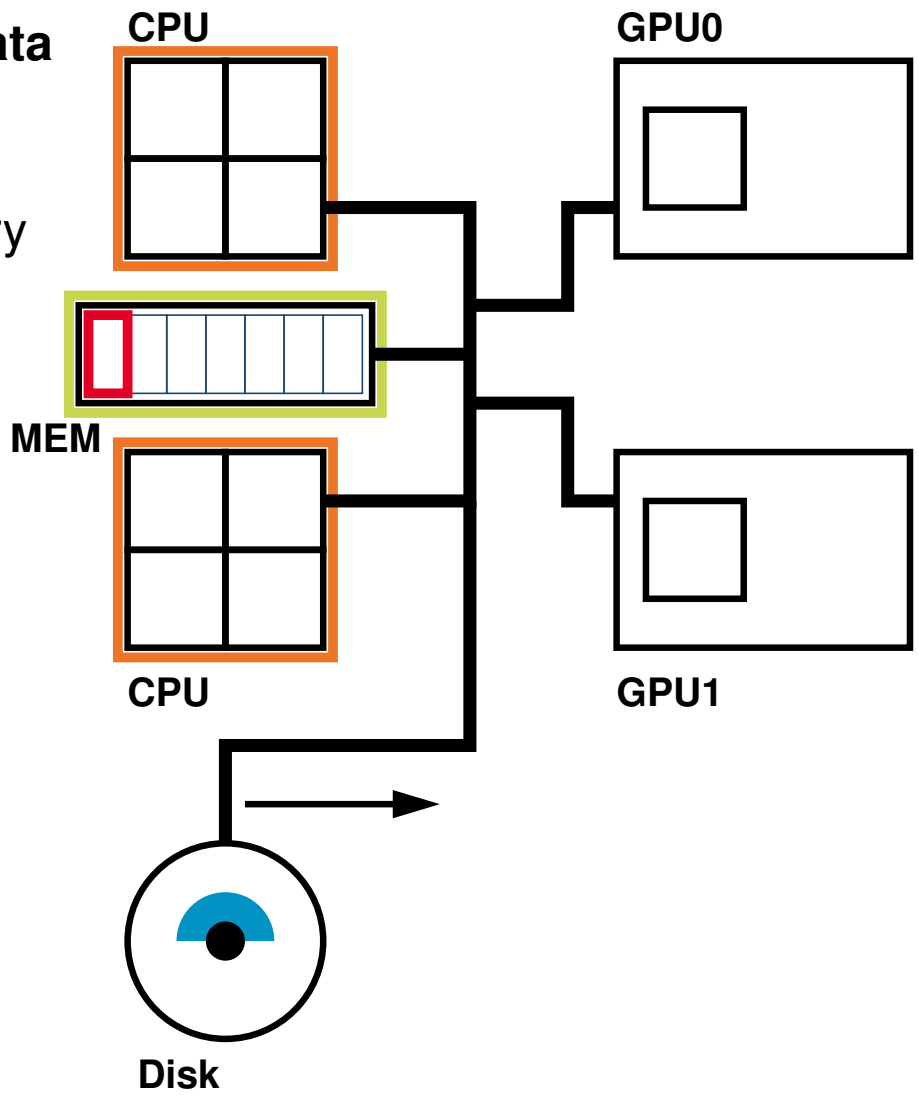
**Storing temporarily unused StarPU data on disk**

Integration with general StarPU's memory management layer

- StarPU data handles
- Task dependencies
  - Data reloaded automatically

Multiple disk drivers supported

- Legacy stdio/unistd methods
- Google's LevelDB
  - (key/value database library)



# Simulation with SimGrid

## Scheduling without executing kernels

- Requires the SimGrid simulation environment
- Enables simulating large-scale scenarios
  - Large data sets
  - Large simulated hardware plaform
- Relies on **real** performance models...
- ...collected by StarPU on a real machine
- Enables fast experiments when designing application algorithms
- Enables fast experiments when designing scheduling algorithms

## Partnerships

- Inria RUNTIME, MESCAL and CEPAGE/REALOPT
- ANR SONGS

# Composition: Scheduling contexts

## Rationale

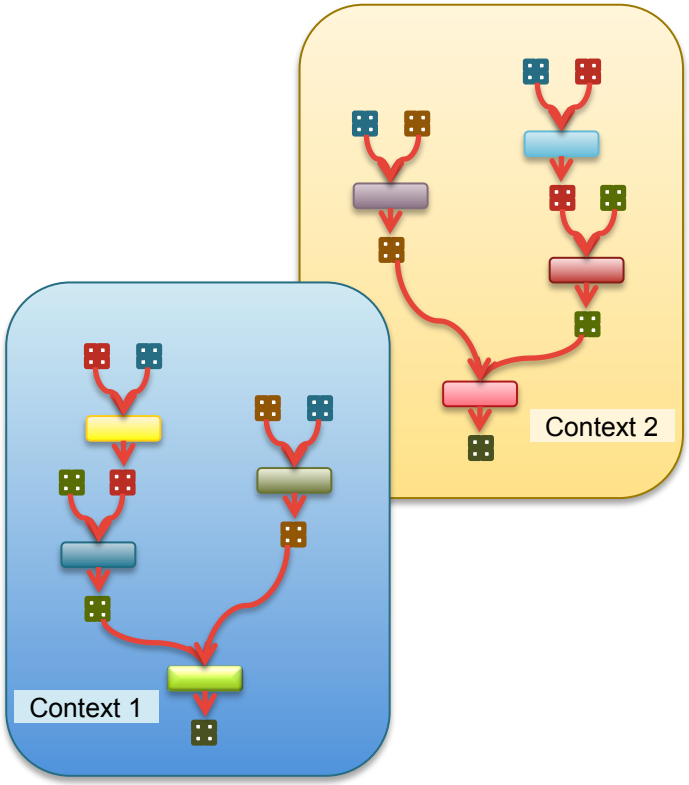
- Sharing computing resources. . .
- . . . among multiple DAGs
- . . . simultaneously
- Composing codes, kernels

## Scheduling contexts

- Map DAGs on subsets of computing units
- Isolate competing kernels or library calls
  - OpenMP kernel, Intel MKL, etc.
- Select scheduling policy per context

## Partnerships

- Inria RUNTIME, HIEPACS
- ANR SOLHAR



# Components: new C2S@Exa Ph.D.

## Objectives

- Software component model with task scheduling
- . . . for many-core based parallel architectures
- . . . applied to the **Gysela5D** code
- Ph.D. student Jérôme Richard (started 11/2014)

## Participants

- Inria AVALON and RUNTIME
- Maison de la Simulation, CEA Cadarache

# Conclusion: C2S@Exa – Pole 3 / Runtime Support

**StarPU Design**

- Sequential task submission
- Inferred dependencies
- Heterogeneous, parallel scheduler
- Distributed shared memory

# Conclusion: C2S@Exa – Pole 3 / Runtime Support

**StarPU Design**

- Sequential task submission
- Inferred dependencies
- Heterogeneous, parallel scheduler
- Distributed shared memory

**Results**

- Runtime support for high level programming
- Runtime support for heterogeneous nodes, clusters and out-of-core
- Runtime support for scheduler composition and code coupling
- Runtime support for scheduling algorithm design and simulation

# Conclusion: C2S@Exa – Pole 3 / Runtime Support

**StarPU Design**

- Sequential task submission
- Inferred dependencies
- Heterogeneous, parallel scheduler
- Distributed shared memory

**Results**

- Runtime support for high level programming
- Runtime support for heterogeneous nodes, clusters and out-of-core
- Runtime support for scheduler composition and code coupling
- Runtime support for scheduling algorithm design and simulation

**On-going work**

- with Pole 1: scalability, support for numerical algebra, interoperability
- with Pole 4: OpenMP compiler, component model, programmability
- Planned application ports
  - Gysela5D (C2S@Exa driving application)
  - ScalFMM
  - CLAC

# Thanks for your attention.

**StarPU**

Web Site: http://runtime.bordeaux.inria.fr/starpu/

SVN Repository: http://gforge.inria.fr/projects/starpu/

LGPL License

Open to external contributors