Resource Management and Interoperability with the StarPU Task-Based Runtime System

Olivier Aumage

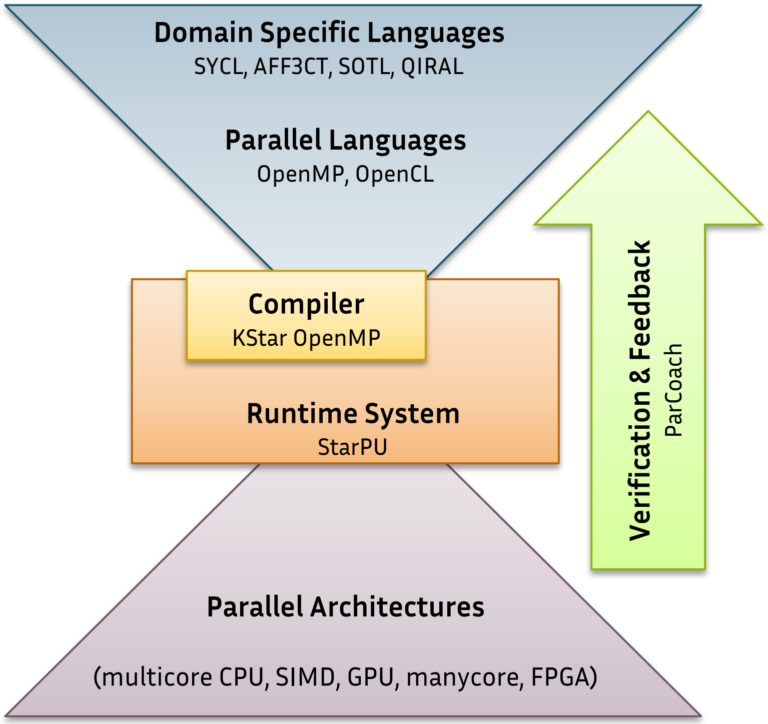
Inria — LaBRI
Bordeaux, France

RADR 2021

Team STORM

STatic Optimizations, Runtime Methods

- **Joint Team in Bordeaux, France**
  - > Inria Bordeaux Research Center
  - > LaBRI Laboratory
- **Parallelism in HPC**
  - > Express
  - > Adapt
  - > Optimize



High-Performance Computing

Supercomputers Hardware Evolution

- **Fast paced**
  - > Short lifetime: 5 – 10 years
- **Increasing complexity**
  - > RIKEN Fugaku Computer: ~160K nodes, ~7M cores
- **Increasing heterogeneity**
  - > Accelerators devices, FPGA, processing offload
- **Increasingly diverse purposes and designs**
  - > Graph / Green / Top 500, HPCG

Name	Start year	Performance (PFLOPS)[note 1]	TOP500 ranking	CPU/GPU vendor	CPU
Fugaku	2020	415	June 2020 1st	Fujitsu	A64FX
Summit	2018	148	June 2018 to November 2019 1st	IBM, NVIDIA	POWER9,
Sierra	2018	94	November 2018 to November 2019 2nd		
Sunway TaihuLight	2016	93	June 2016 to November 2017 1st	NRCPC	Sunway SW
K	2011	10	June 2011 – November 2011 1st	Fujitsu	SPARC64

CPU
A64FX
POWER9, Tesla
Sunway SW26010
SPARC64 VIIIfx

Wikipedia.org: Fugaku vs some former rank #1 Top500 supercomputers

Task-based runtime systems

Performance portability

- **Separate multiple concerns**
  - General application algorithmics
  - Low-level task kernel optimization
  - **Resource management and work assignment**
    - Task scheduling algorithmics
- **Concentrate porting efforts**
  - [Machine- | Device-] specific routines
    - **== Tasks**
    - Short term adaptation & optimization effort
  - Mostly fixed application structure
    - Long term stability
- **Many active projects**
  - Launched over last decade

- **StarPU**
  - **Inria / LaBRI, Bordeaux, 2009**
- **DuctTeip / SuperGlue**
  - University of Uppsala, 2013
- **HPX**
  - Louisiana State University, 2013
- **OCR**
  - Specification, 2014
  - Several implementations
    - Intel+Rice University
    - University of Vienna
- **OmpSs**
  - BSC, 2008 (StarSs)
- **PaRSEC**
  - ICL / UTK, 2012 (DaGUE)
- **Regent / Legion**
  - Stanford, 2012
- *. . . and many others . . .*

The StarPU runtime system

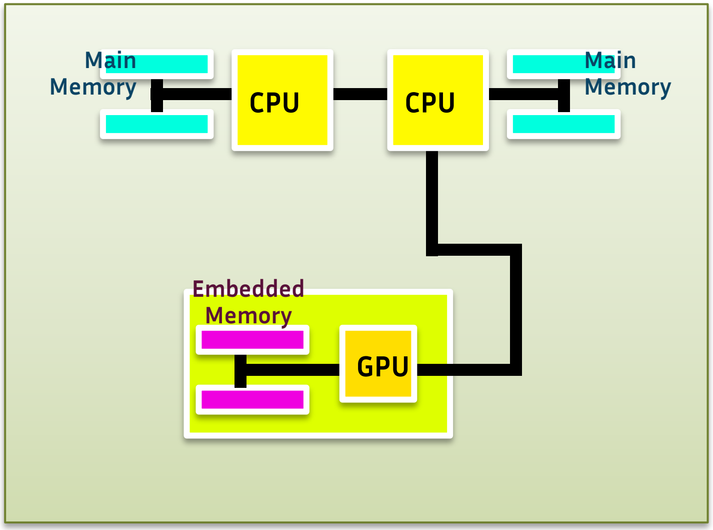
Task-based Computing Runtime System

- **Initiated in 2009**
  - PhD Cédric Augonnet

The StarPU runtime system

Task-based Computing Runtime System

- **Initiated in 2009**
  - PhD Cédric Augonnet
- **Task scheduling on a heterogeneous, accelerated node**
  - General purpose CPU cores
  - Specialized accelerators
    - Discrete board + embedded memory

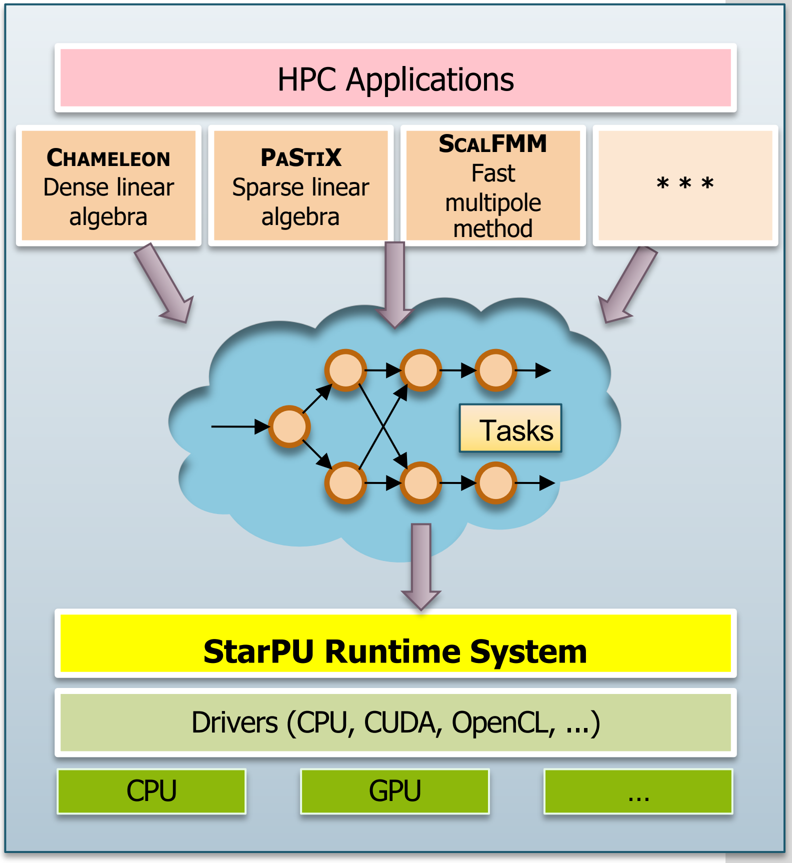


Heterogeneous computing node

The StarPU runtime system

Task-based Computing Runtime System

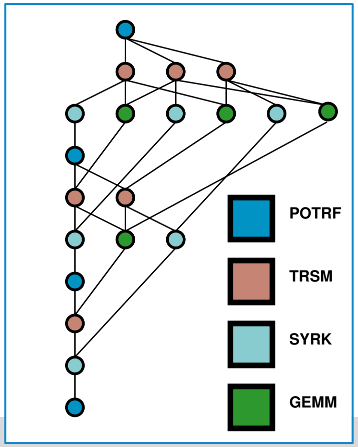
- **Initiated in 2009**
  - > PhD Cédric Augonnet
- **Task scheduling on a heterogeneous, accelerated node**
  - > General purpose CPU cores
  - > Specialized accelerators
    - Discrete board + embedded memory
- **Usage**
  - > Direct programming from application
    - C, C++, Fortran
  - > Compiler / Language
    - OpenMP, Julia, Python, SkePU
  - > Parallel numerical library



Sequential Task Flow

StarPU programming model

- **Tasks submitted sequentially**
  - Deferred execution
- **Dependence graph built incrementally**
  - Vertex == task
  - Edge == data dependence



Dependence graph

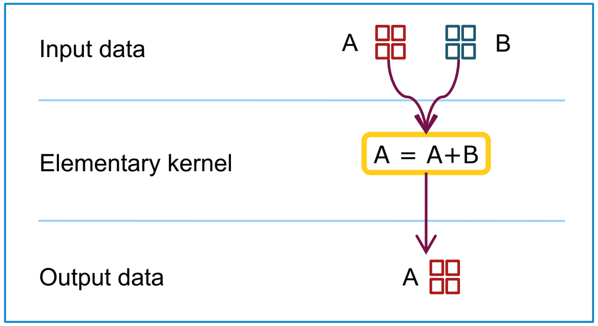
```
for (j = 0; j < N; j++)
{
    starpu_task_insert( POTRF (RW,A[j][j]) );

    for (i = j+1; i < N; i++)
        starpu_task_insert( TRSM (RW,A[i][j], R,A[j][j]) );

    for (i = j+1; i < N; i++)
    {
        starpu_task_insert( SYRK (RW,A[i][i], R,A[i][j]) );

        for (k = j+1; k < i; k++)
            starpu_task_insert( GEMM (RW,A[i][k], R,A[i][j], R,A[k][j]) );
    }
}

starpu_task_wait_for_all();
```

Flow of task submissions

Sequential Task Flow

Model assumptions

- **Tasks**
  - > Annotated kernels
  - > → **Potential parallelism**
- **Data dependences**
  - > Set of constraints
    - Input needed
    - Output produced
  - > → **Degrees of freedom**
- **No hidden dependence**
  - > Any task schedule fulfilling data dependence constraints is correct
  - > **Assume pure, stateless task functions**

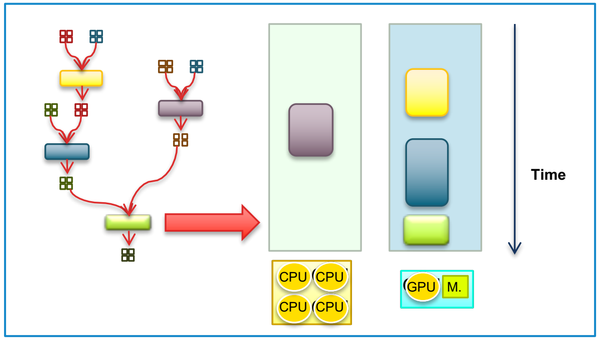


Task == kernel + data dependences

Making hardware dependent decisions on behalf of the programmer

StarPU execution model

- **Distributed Shared Memory (DSM) engine**
  - > Data management
  - > Data replication and consistency
- **Performance modeling engine**
  - > Task execution time inference
  - > Data transfer time inference
- **Scheduling engine**
  - > Programmable policies
    - Theoretical algorithmic corpus
  - > Task mapping
    - Reactive (== work stealing)
    - **Anticipative (== planning)**

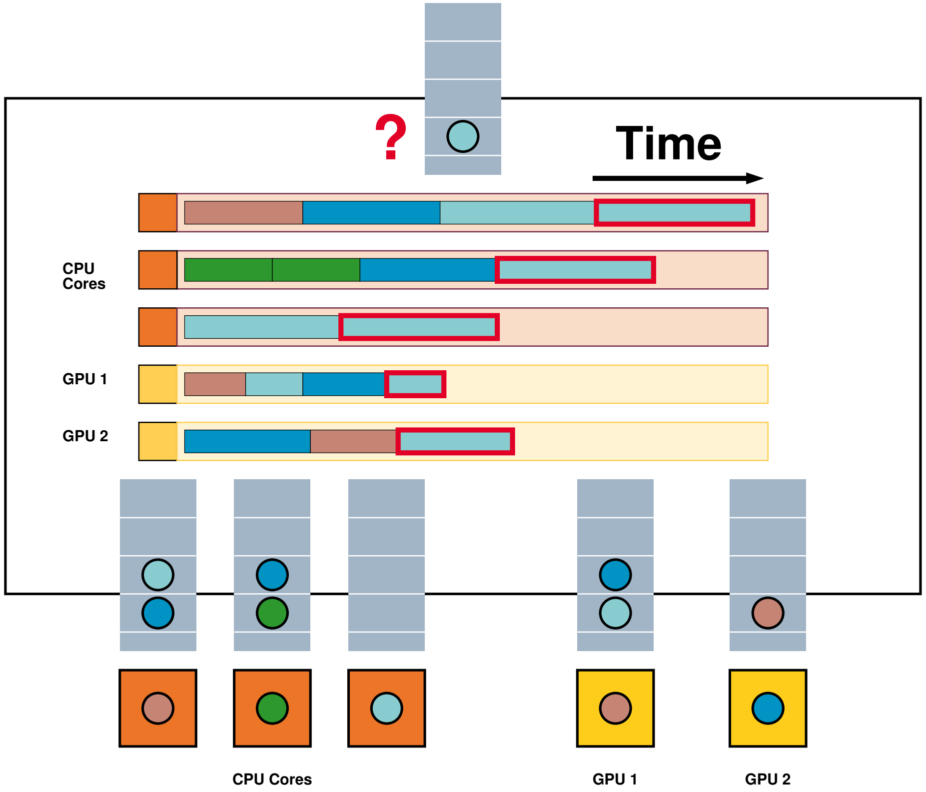


Mapping a task graph on hardware resources

Heterogeneous processing resource management

Dynamically planned execution

- **Performance estimation**
  - > Per kernel
  - > Per device
  - > Per implementation
- **Task execution time inference**
  - > History-based
  - > Custom cost function
- **Data transfer time inference**
  - > Bus sampling



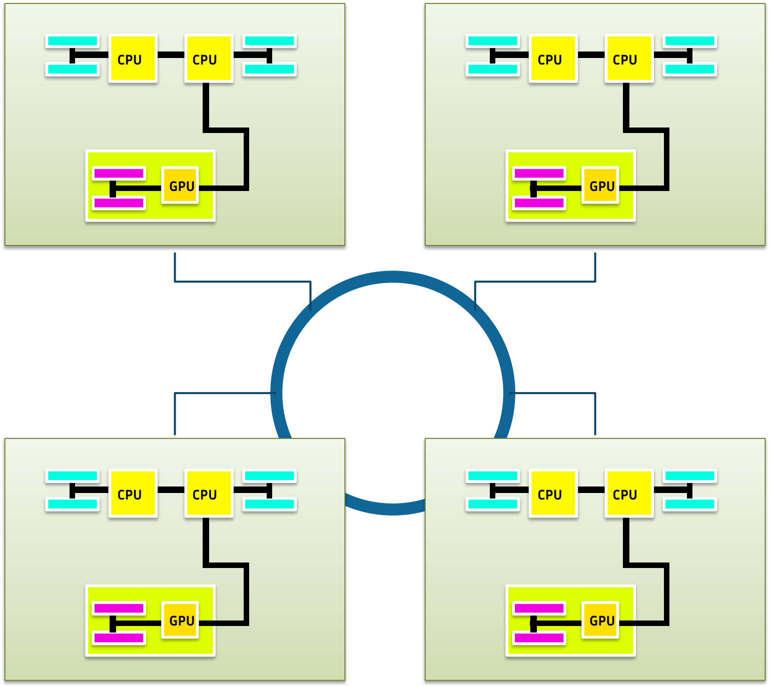
C. Augonnet, S. Thibault , R. Namyst, P.-A. Wacrenier
StarPU: a unified platform for task scheduling on heterogeneous multicore architectures
CCPE, Wiley, 2011.

Distributed processing management

StarPU-MPI

- **Contributions**
  - Early prototype by Cédric Augonnet
  - PhD Marc Sergent

- **2 models supported**
  - Master – workers
  - **Fully distributed**

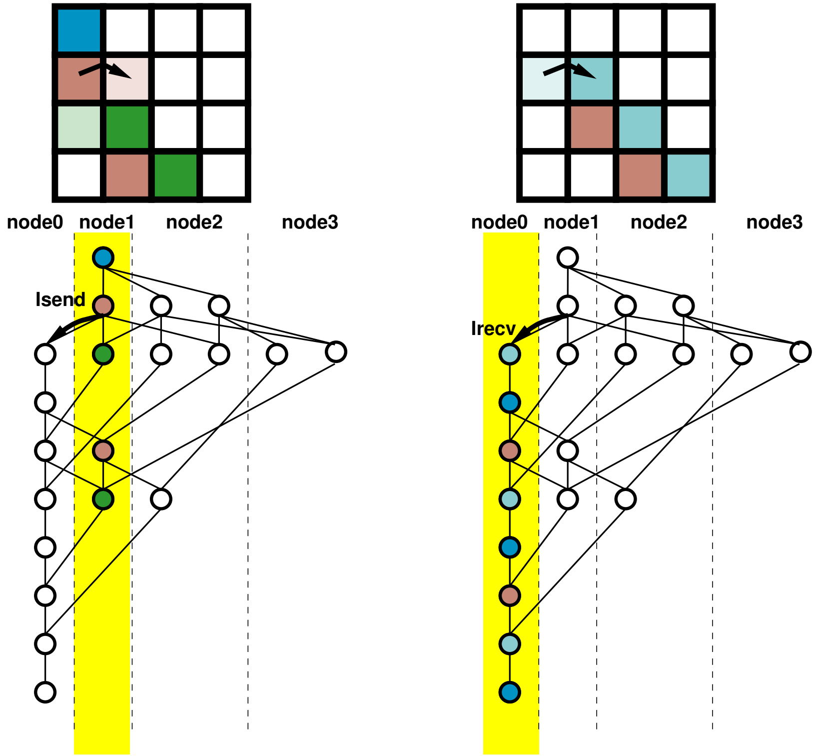


Cluster of heterogeneous nodes

Fully-distributed model

No master node

- **Local task graph discovery**
  > Whole graph discovered on every node
  > Initial data distribution given by application
- **Local decisions**
  > Task execution
    - Data ownership
  > Data transfers
    - Internode edges

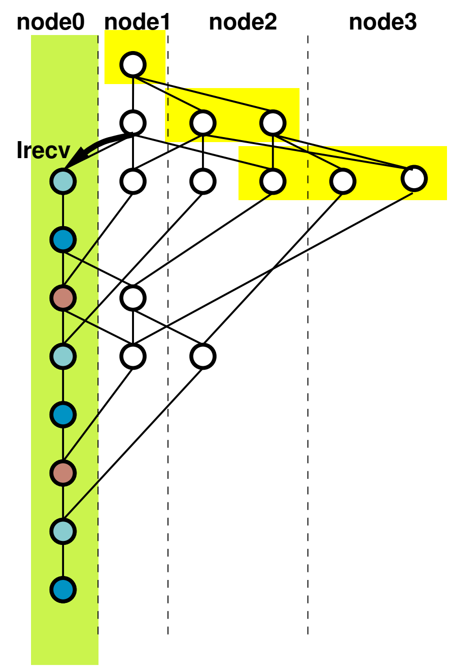


E. Agullo, O. Aumage, M. Faverge, N. Furmento, F. Pruvost, M. Sergent, S. Thibault
Achieving High Performance on Supercomputers with a Sequential Task-based Programming Model
IEEE Transactions on Parallel and Distributed Systems, 2017.

Fully-distributed model

No master node

- **Local task graph discovery**
  - > Whole graph discovered on every node
  - > Initial data distribution given by application
- **Local decisions**
  - > Task execution
    - Data ownership
  - > Data transfers
    - Internode edges
- **Task graph pruning**
  - > Scalable processing



Pruning tasks on node 0

E. Agullo, O. Aumage, M. Faverge, N. Furmento, F. Pruvost, M. Sergent, S. Thibault
Achieving High Performance on Supercomputers with a Sequential Task-based Programming Model
IEEE Transactions on Parallel and Distributed Systems, 2017.

Computing resource clustering

Parallel Tasks

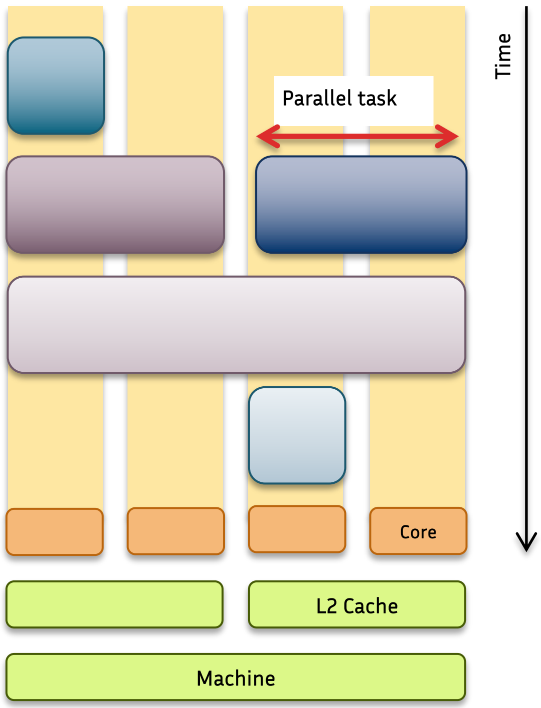
- **Contribution**
  - PhD Terry Cojean

T. Cojean, A. Guermouche, A. Hugo, R. Namyst, P.-A. Wacrenier
Resource aggregation for task-based Cholesky Factorization on top of modern architectures
Parallel Computing, Elsevier, 2018.

Computing resource clustering

Parallel Tasks

- **Contribution**
  - PhD Terry Cojean
- **Enable multi-core tasks**
  - **Control scheduling cost on large multicores**
    - Use less tasks
  - **Enhance affinity**
    - Topology-based mapping
  - **Reduce heterogeneous performance gap**
    - GPU task vs multicore CPU task



T. Cojean, A. Guermouche, A. Hugo, R. Namyst, P.-A. Wacrenier
Resource aggregation for task-based Cholesky Factorization on top of modern architectures
Parallel Computing, Elsevier, 2018.

Resource management for multiple task graphs

Scheduling contexts

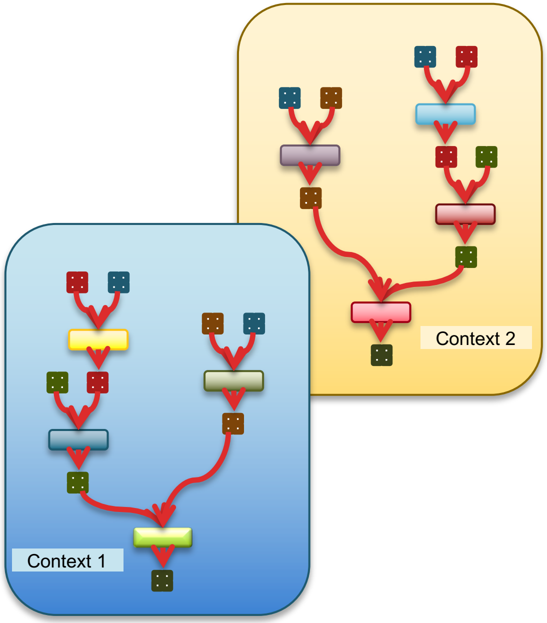
- **Contribution**
  - PhD Andra Hugo

A. Hugo, A. Guermouche, P.-A. Wacrenier, R. Namyst
Composing multiple StarPU applications over heterogeneous machines: A supervised approach
IJHPCA, SAGE Publications, 2014.

Resource management for multiple task graphs

Scheduling contexts

- **Contribution**
  - > PhD Andra Hugo
- **Single StarPU instance**
  - > Multiple task graphs
  - > Concurrent StarPU-based routines

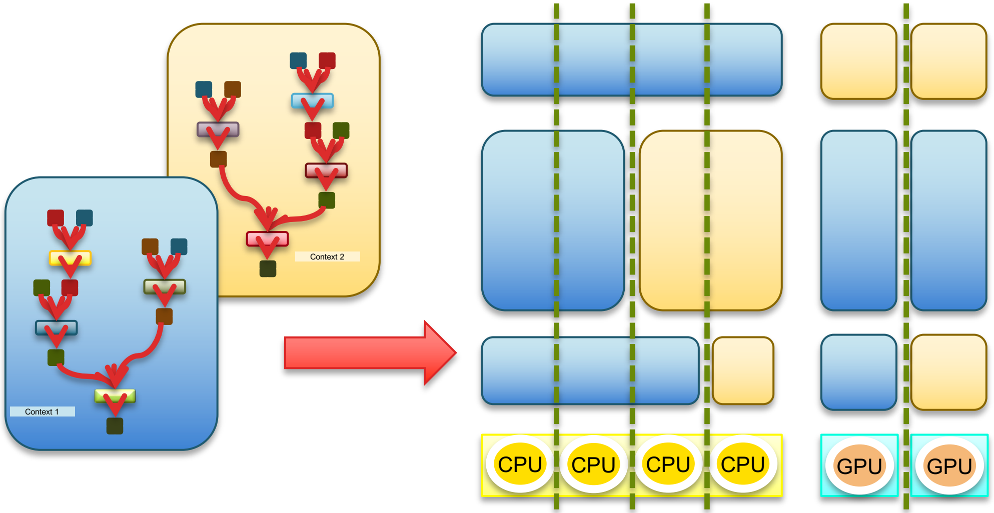


A. Hugo, A. Guermouche, P.-A. Wacrenier, R. Namyst
Composing multiple StarPU applications over heterogeneous machines: A supervised approach
IJHPCA, SAGE Publications, 2014.

Resource management for multiple task graphs

Scheduling contexts

- **Contribution**
  - > PhD Andra Hugo
- **Single StarPU instance**
  - > Multiple task graphs
  - > Concurrent StarPU-based routines
- **Dynamic resource assignment**
  - > Malleability



A. Hugo, A. Guermouche, P.-A. Wacrenier, R. Namyst
Composing multiple StarPU applications over heterogeneous machines: A supervised approach
IJHPCA, SAGE Publications, 2014.

Resource management among multiple task-based runtimes

Leverage multiple libraries & runtimes

- **Multiple codes competing for CPU cores**
  - Application threads
  - Parallel numerical libraries threads
  - Runtime systems threads
  - Communication library threads

Inria

Resource management among multiple task-based runtimes

Leverage multiple libraries & runtimes

- **Multiple codes competing for CPU cores**
  - Application threads
  - Parallel numerical libraries threads
  - Runtime systems threads
  - Communication library threads

- **Interference leads to resource over-subscription or under-subscription**
  - **Interoperability?**
  - European Project **H2020 INTERTWinE** (2015 – 2018)
    - Resource sharing APIs
    - http://www.intertwine-project.eu/

Resource management among multiple task-based runtimes

Scenarios

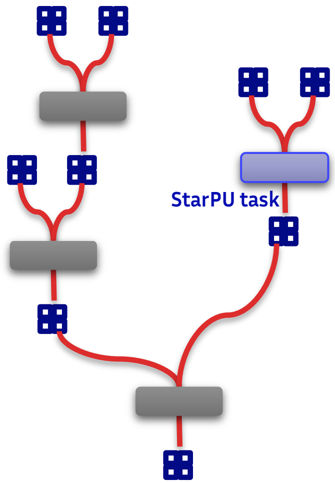
- **Nested interoperability**
  - Host runtime
    - Task-parallel application or library
  - Guest runtime
    - Parallel implementation of host's tasks

- **Concurrent interoperability**
  - Host runtime
    - Parallel application or library
  - Guest runtime
    - Concurrent parallel library

Resource management among multiple task-based runtimes

Nested composition

- **Task parallel application or library + parallel kernel tasks**
  - Offload and resource enforcement API

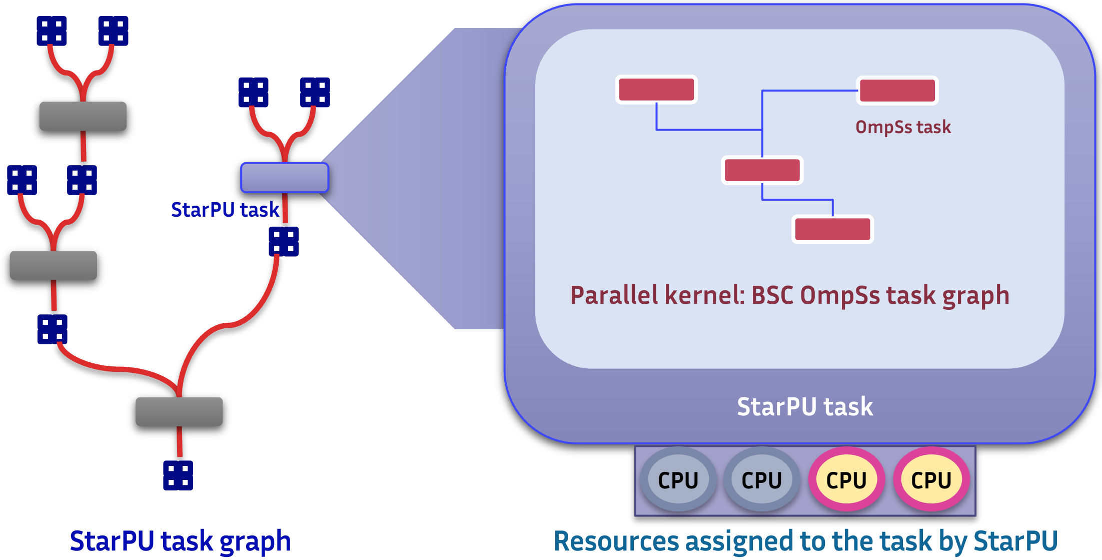


StarPU task graph

Resource management among multiple task-based runtimes

Nested composition

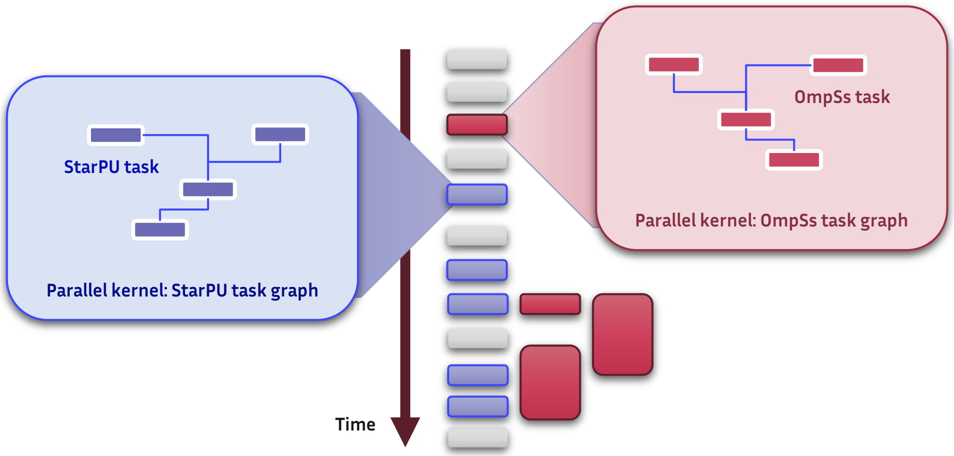
- **Task parallel application or library + parallel kernel tasks**
  > Offload and resource enforcement API



Resource management among multiple task-based runtimes

Concurrent composition

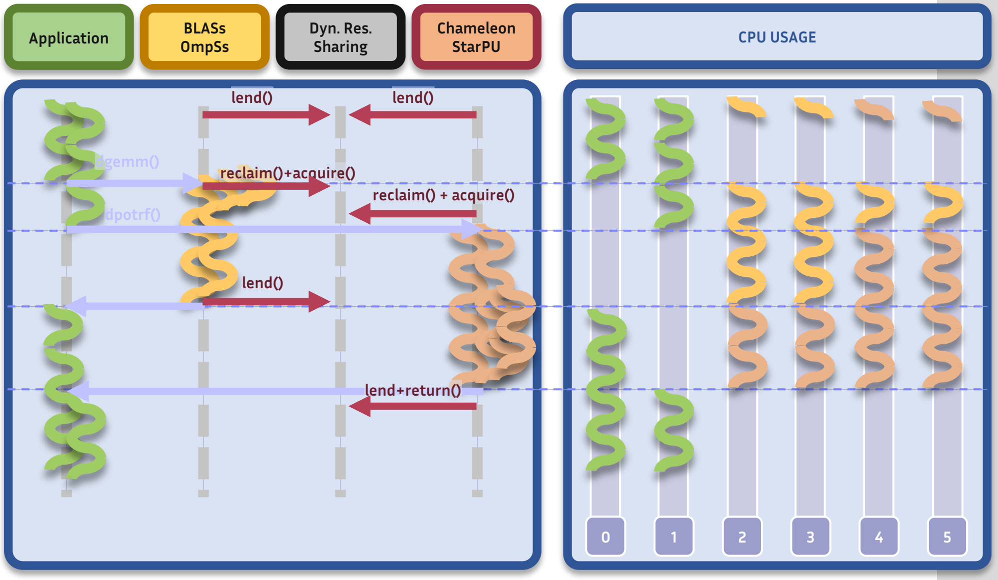
- **Parallel application or library // parallel library**
  > Dynamic Resource Sharing (DRS) API



Resource management among multiple task-based runtimes

Concurrent composition

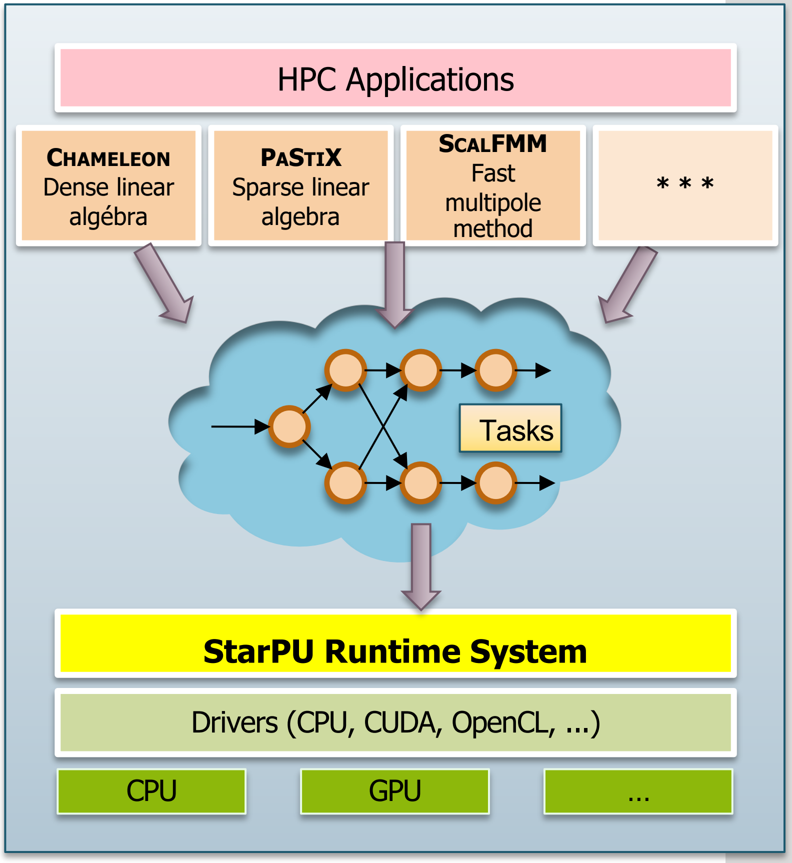
- **Parallel application or library // parallel library**
  > Dynamic Resource Sharing (DRS) API
- **Direct interfacing**
  > StarPU
  > OmpSs
  > Same process computing resource sharing
- **Interfacing through external component**
  > DLB (Dynamic Load Balancing) framework
    - Developed at BSC
    - Library + external daemon
  > Same process or multi-processes computing resource sharing
- **http://www.intertwine-project.eu/**



Conclusion

The StarPU task-based runtime system

- **Comprehensive in-app resource management**
  - > Heterogeneous processing units: **CPU**, **GPU**, . . . , ***PU**
    - Planned + work stealing task scheduling
    - Performance modeling
  - > Heterogeneous memory resource management
    - Data replication + memory consistency
    - NUMA, HBM, on-device memory, out-of-core
- **Ecosystem friendly resource management**
  - > Interoperability, composability, malleability
    - StarPU parallel code + StarPU parallel code
    - StarPU parallel code + external parallel code
- **https:// starpu . gitlabpages . inria . fr /**



Thanks!