



STAPL Executor

Alin Julia

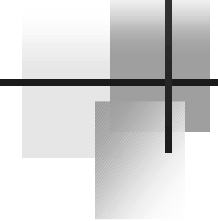
PARASOL Research Lab

Texas A&M University



What Is the Executor?

- Parallel tool for executing statically scheduled tasks
- Supports both distributed and shared memory systems (OpenMP and MPI)
- MIMD model of execution
- Standard Template Adaptive Parallel Library (STAPL) component



Why Do We Need the Executor?

- Exploit maximum amount of parallelism
 - Overlap communication/computation
- Execute on a heterogeneous architectural topology
 - distributed memory systems
 - shared memory systems

What Does It Do?

- **Input** : a static order of task execution
 - Provided by graph partitioning/scheduling
 - Quanta of work before communication
 - Data dependence graphs (DDG)
- **Output** :
 - parallel execution of work functions on data dependence graphs
 - overlapping communication/computation



How Does It Work?

- Each processor has
 - A list of partial DDGs
- Executor performs a scheduling policy (e.g round-robin) among partial DDGs
- Checks a task for data dependencies satisfiability
- If ALL data dependencies satisfied
 - Execute work function with that task



Algorithm Overview

- Initialization
 - Assign partial DDGs to processors
- While (unexecuted tasks)
 - Check for ready-to-execute tasks
 - For ready-to-execute tasks
 - Satisfy data dependence (receive)
 - Execute ready tasks
 - For executed tasks
 - Satisfy data dependence (send)



Executor Usage

- **ASCI** – particle transport calculations
 - For **ASCI** it uses **SPMD** model of execution (the same work function - e.g. sweep, scattering routines - for all **DDGs**)
- **STAPL** execution mechanism

Example

Space Decomposition

P1 P2

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

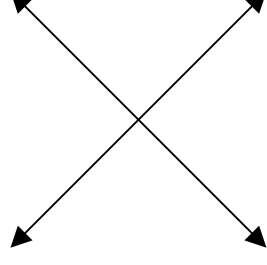
P3 P4

Legend

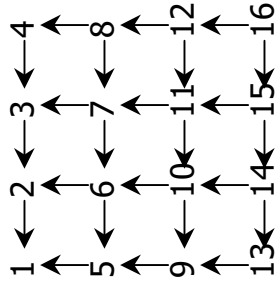
—	processor boundary
—	task boundary
□	task

Example (cont)

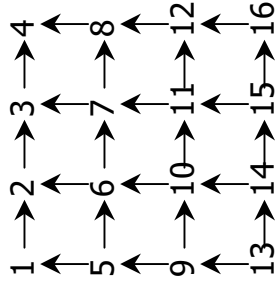
- For a given set of anglesets
- Create the Data Dependence Graphs



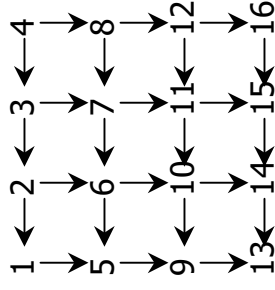
Angleset 1



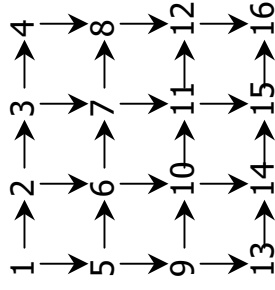
Angleset 2



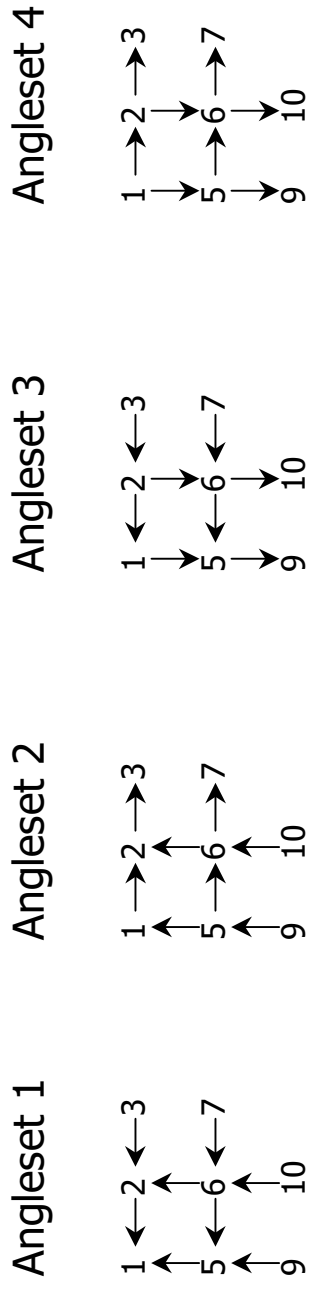
Angleset 3



Angleset 4



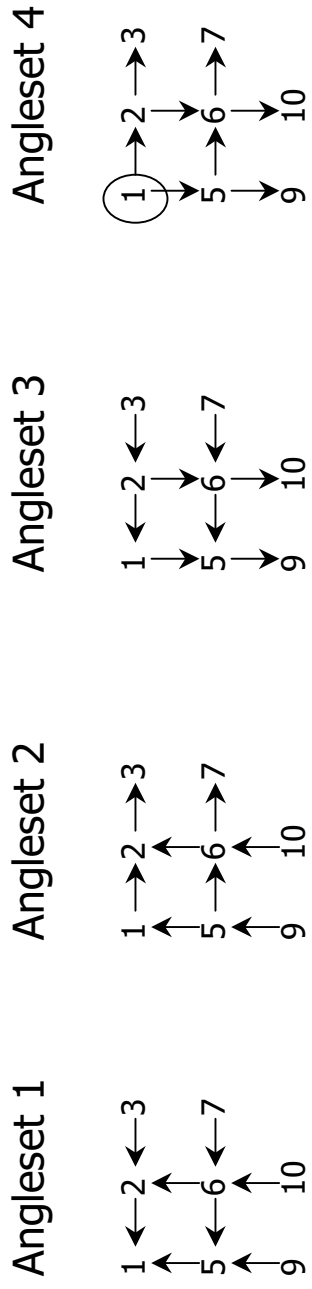
Example (execution on processor 1)



Ready-to-compute Tasks = { null }

Executed Tasks = { null }

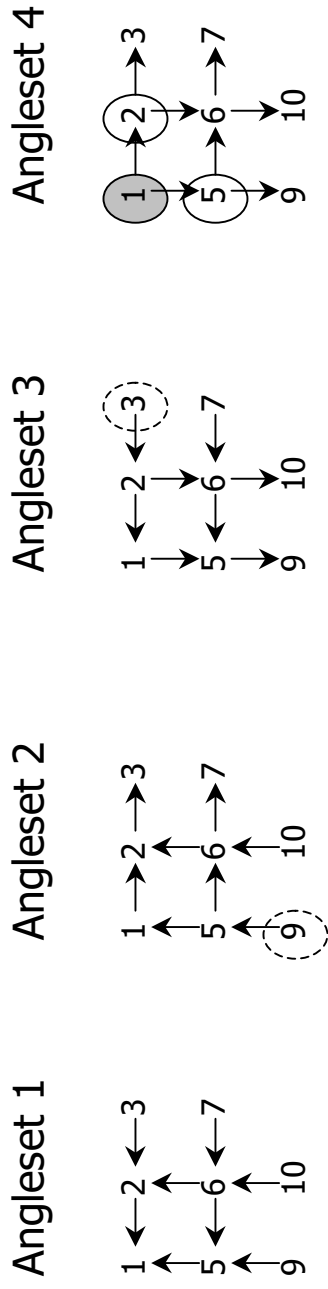
Example (execution on processor 1)



Ready-to-compute Tasks = { ① }

Executed Tasks = { null }

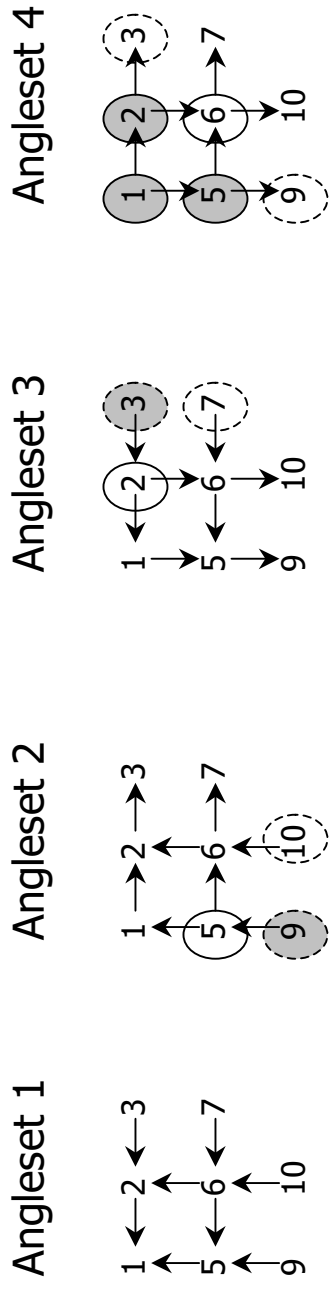
Example (execution on processor 1)



Ready-to-compute Tasks = { 2, 5 }

Executed Tasks = { 1 }

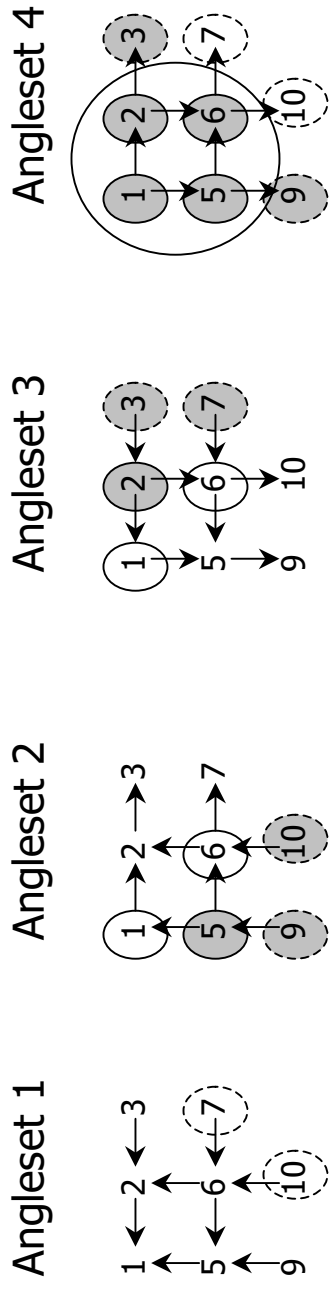
Example (execution on processor 1)



Ready-to-compute Tasks = { 6 2 5 }

Executed Tasks = { 1 2 5 }

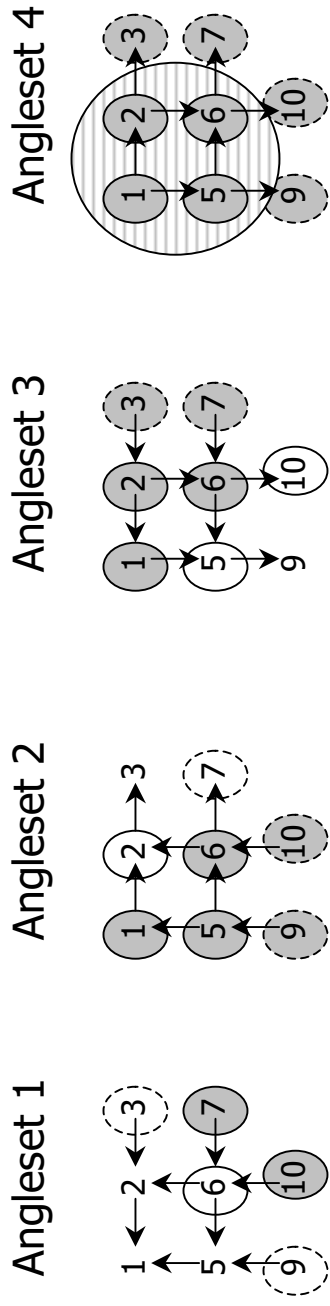
Example (execution on processor 1)



Ready-to-compute Tasks = { 1 6 1 6 }

Executed Tasks = { 1 2 5 6 2 5 }

Example (execution on processor 1)

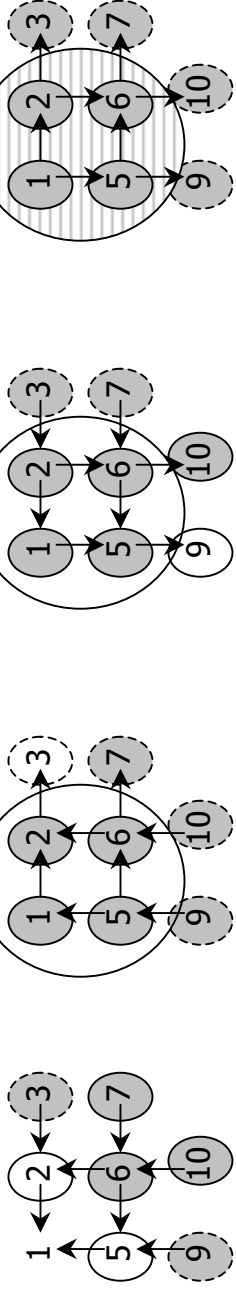


Ready-to-compute Tasks = { 5 2 6 }

Executed Tasks = { 1 2 5 6 2 5 1 6 1 6 }

Example (execution on processor 1)

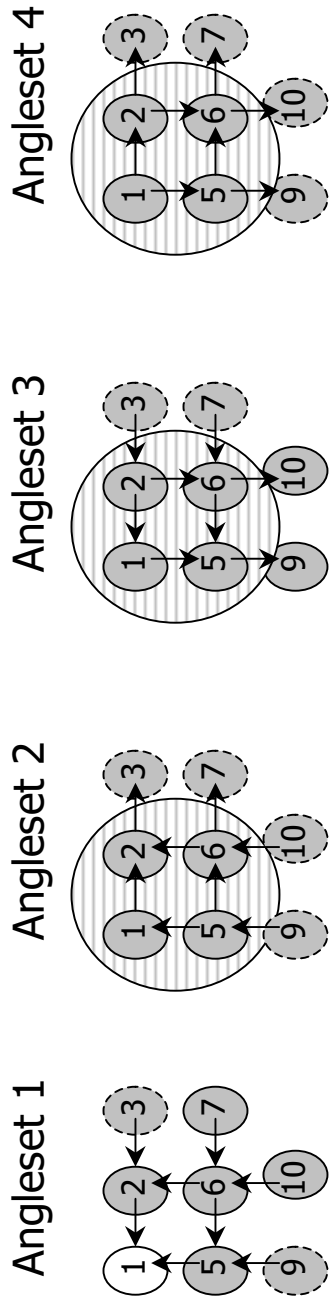
AngleSet 1 AngleSet 2 AngleSet 3 AngleSet 4



Ready-to-compute Tasks = { 2 5 }

Executed Tasks = { 1 2 5 6 2 5 1 6 1 6 2 5 6 }

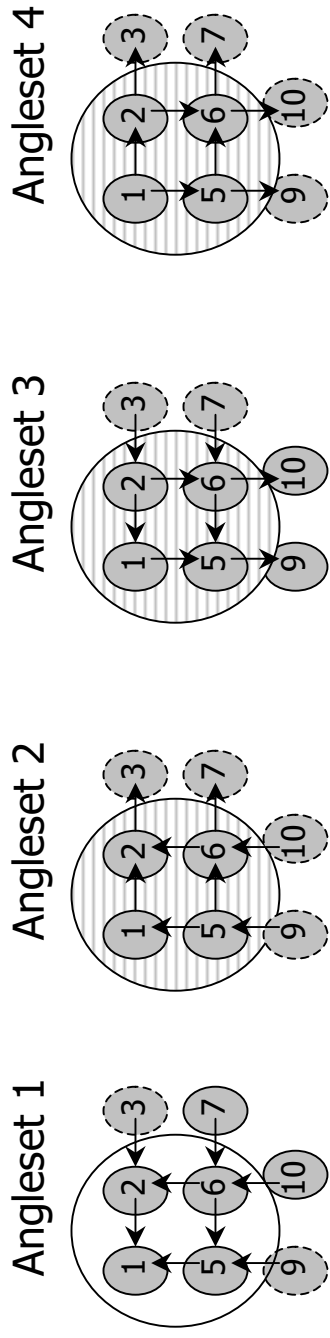
Example (execution on processor 1)



Ready-to-compute Tasks = { 1 }

Executed Tasks = { 1 2 5 6 2 5 1 6 1 6 2 5 6 2 5 }

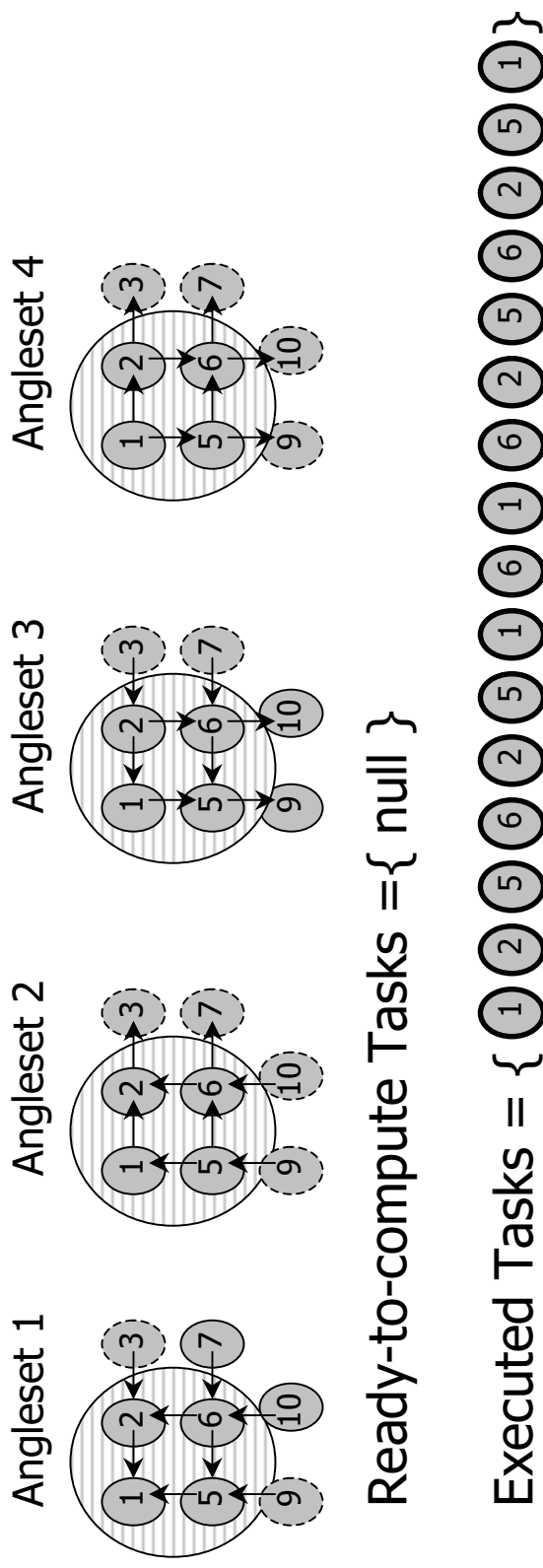
Example (execution on processor 1)



Ready-to-compute Tasks = { null }

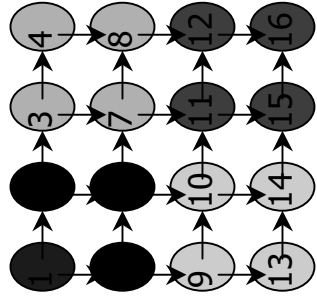
Executed Tasks = { 1 2 5 6 2 5 1 6 1 6 2 5 6 2 5 1 }

Example (execution on processor 1)

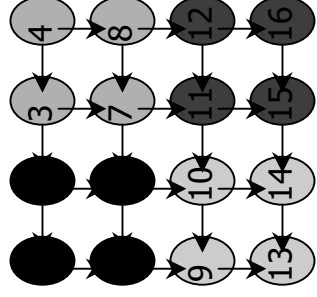


Execution on processor 1 finalizes

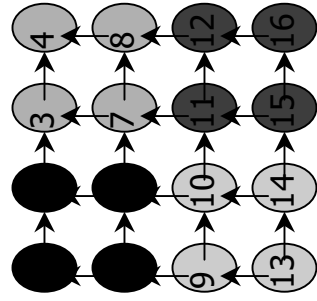
Example (overall execution)



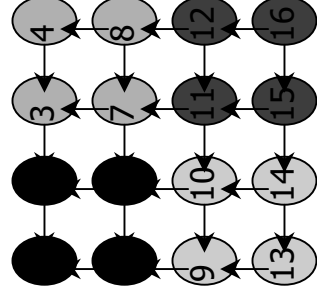
Angleset 4



Angleset 3



Angleset 2



Angleset 1



Executor Status (completed)

- MPI and OpenMP implementations
- STAPL working component

Current Work

- Automatically choose MPI/OpenMP

