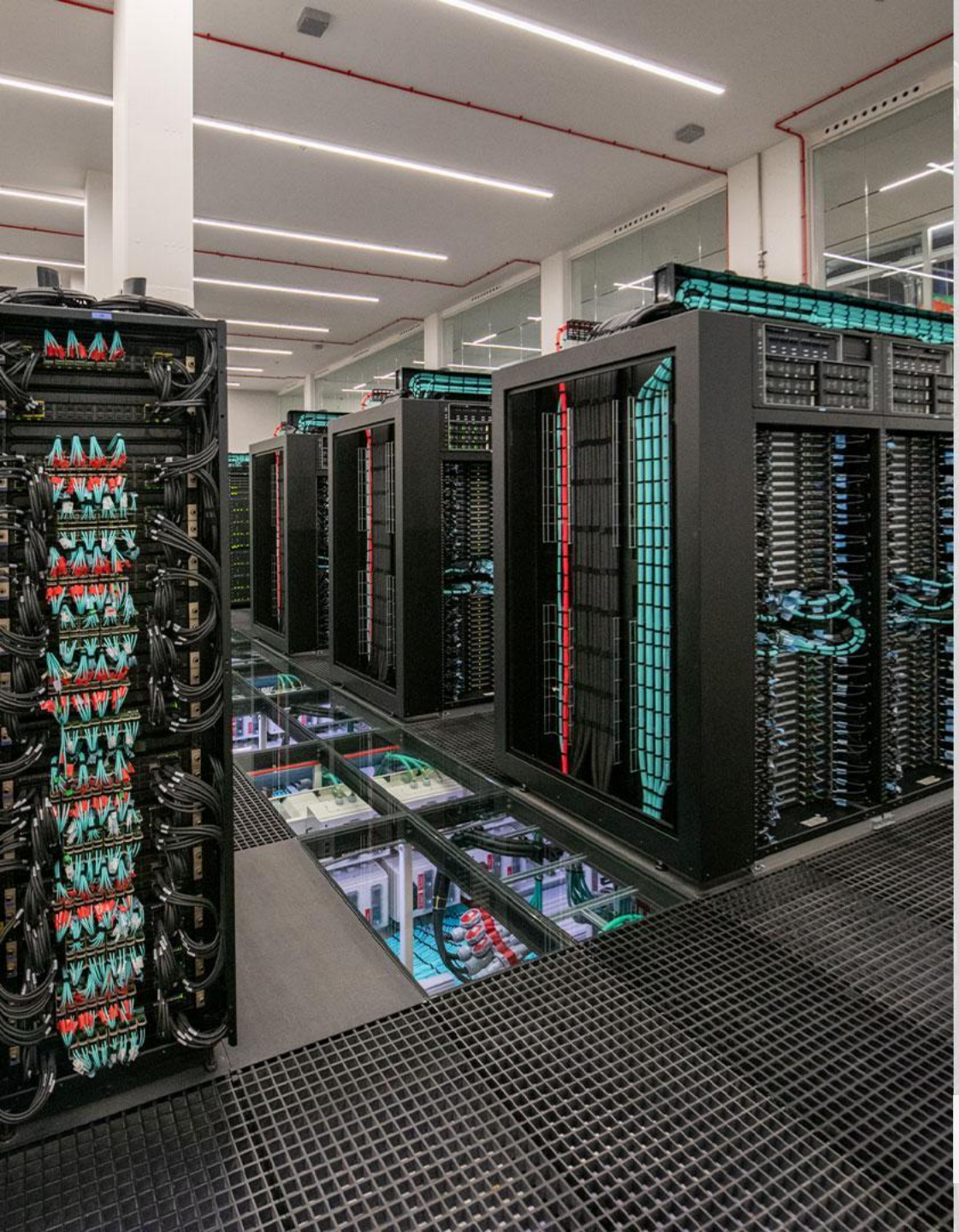


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Decentralized Scheduling of Scientific Workflows

Xavier Casas-Moreno[1], Komal Thareja[2], Anirban Mandal[2], Ewa Deelman[3], Rosa M Badia[1], Francesc Lordan[1]

[1] Computer Sciences Department, Barcelona Supercomputing Center
[2] Renaissance Computing Institute, University of North Carolina - Chapel Hill
[3]Information Sciences Institute, University of Southern California

SC26 Workshops · WORKS · November 2026

Motivation: why decentralize workflow scheduling?

CENTRALIZED WMS

One scheduler maintains a global view and dispatches tasks.

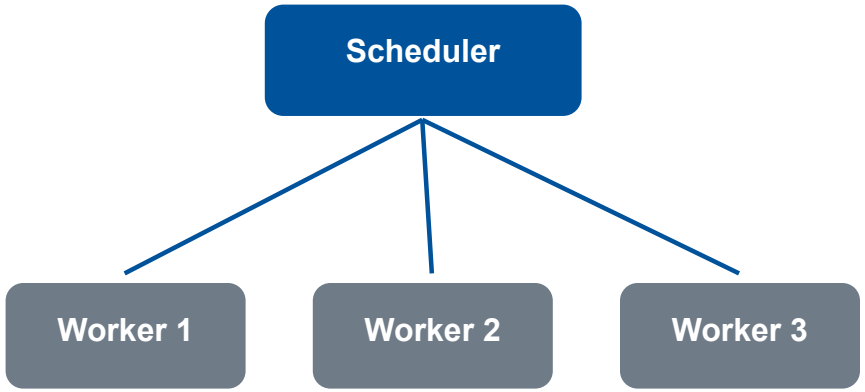
DISTRIBUTED WORKFLOWS

Nested workflows can generate tasks across multiple runtimes.

COMPUTE CONTINUUM

HPC, cloud and edge resources are heterogeneous and distributed.

Traditional master–worker model



Why this becomes a problem

- All scheduling decisions pass through one component.
- Distributed task generation increases coordination overhead.
- Centralization limits scalability and creates a single point of failure.

Research question: can agents allocate tasks collaboratively, without a central coordinator?

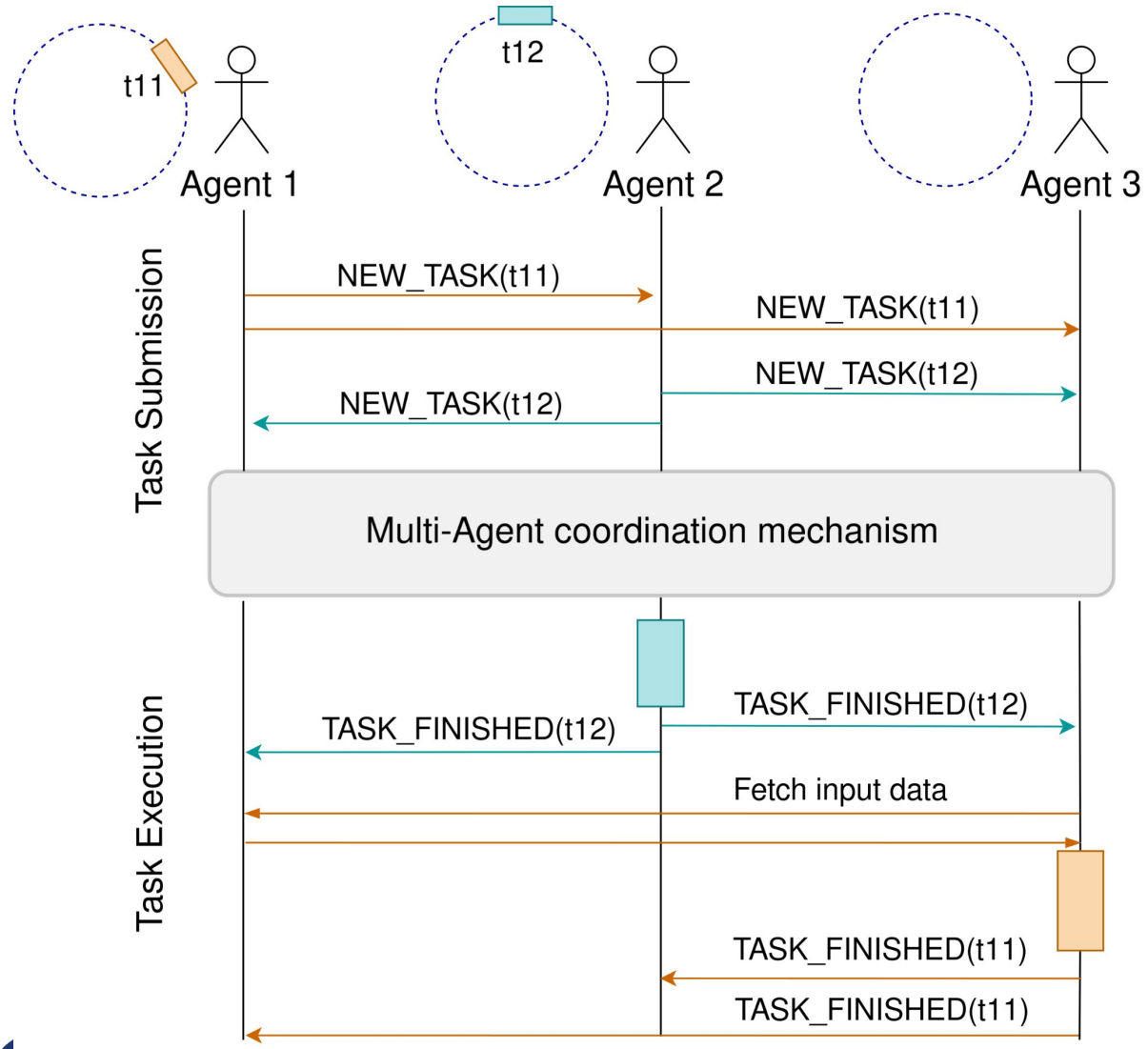
Research gap: from centralized to fully decentralized allocation

APPROACH	WHAT IT PROVIDES	REMAINING LIMITATION
Centralized WMS	Global workflow view; mature scheduling	Central bottleneck and single point of failure
Hierarchical offloading	Distributed task submission and execution	Tasks at the lowest level cannot be redistributed
SWARM consensus	Collaborative allocation through agreement	Original approach relies on a shared global database

OUR CONTRIBUTION

Integrate SWARM-inspired consensus into COMPSs and replace the shared database with message-based coordination.

A general approach for decentralized task scheduling



Three stages

1 DISTRIBUTED SUBMISSION

An agent detects a ready task and broadcasts NEW_TASK.

2 COLLABORATIVE ALLOCATION

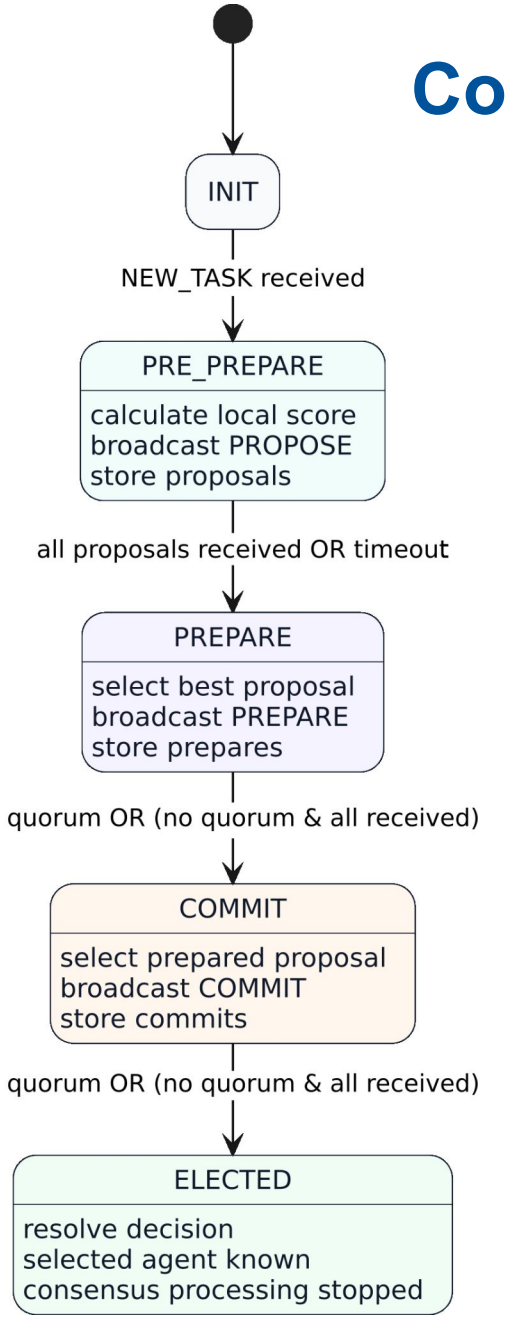
Agents exchange proposals and agree on an execution agent.

3 REMOTE EXECUTION

The selected agent retrieves inputs, executes the task and broadcasts TASK_FINISHED.

No central coordinator · No shared global database

Consensus-based task allocation



PRE-PREPARE

Each agent computes a local scheduling cost and broadcasts a PROPOSE message.

PREPARE

Agents select the lowest-cost proposal and exchange votes. Quorum: more than half of the agents.

COMMIT

Agents confirm the selected proposal. Once quorum is reached, the decision is finalized.

If no quorum: after all messages arrive, apply the same deterministic lowest-cost rule.

Scheduling cost: resource awareness and data locality

Resource utilization

$$u_r(a,t) = q_r(t) / c_r(a)$$

- $q_r(t)$: resource r required by task t
- $c_r(a)$: resource r currently available at agent a
- Maximum utilization across required resources determines the resource-related cost.
- Agents unable to satisfy all requirements are discarded.

DATA LOCALITY

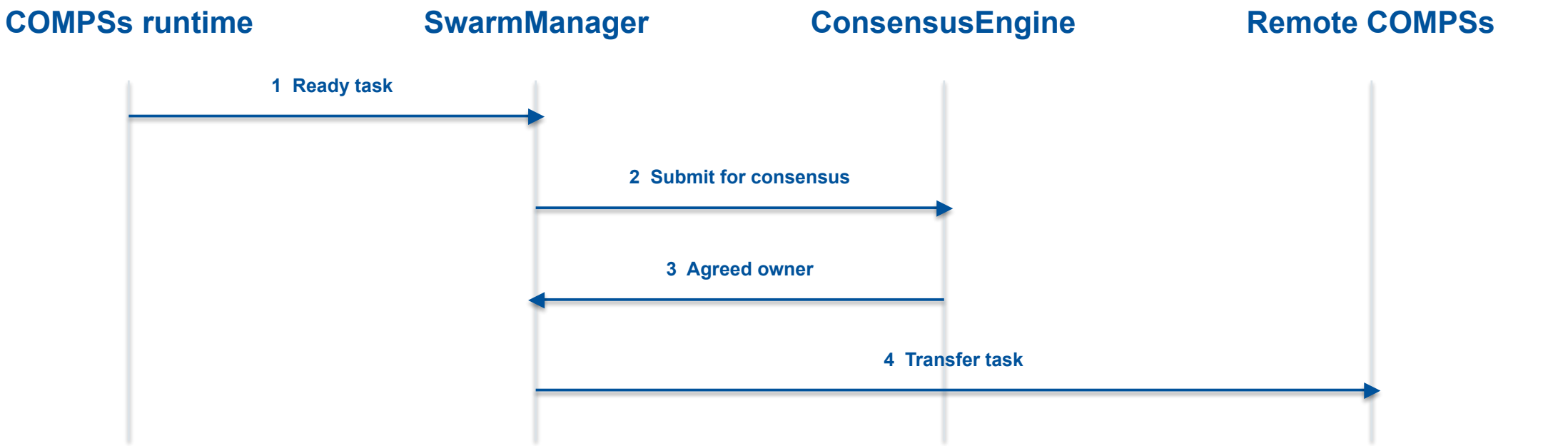
If input data is not locally available, cost is multiplied by a configurable locality factor λ.

EXPERIMENTAL SETTING

λ = 1.001 — a slight preference for local execution, while allowing better resource matches elsewhere.

Consensus is independent of the cost function: other scheduling objectives can be used without changing the agreement procedure.

SwarmTS: integrating decentralized scheduling into COMPSs



SwarmManager

Coordinates task ownership, communication and the remote-task lifecycle.

SwarmCommunicationManager

Maintains peer connections and exchanges messages through COMPSs NIO.

ConsensusEngine

Runs one consensus state machine per task and returns the final decision.

After allocation, the selected agent injects the task into its local COMPSs scheduler; execution status is propagated to peers.

Experimental evaluation on MareNostrum 5

INFRASTRUCTURE

MareNostrum 5

General-purpose partition

COMPUTE RESOURCES

3–20 nodes / agents

112 CPU cores · 256 GB RAM per node

WORKLOAD

Critical Support Vector Machine (CSVM)

Nested GridSearch

Hyperparameter optimization

Workflow characteristics

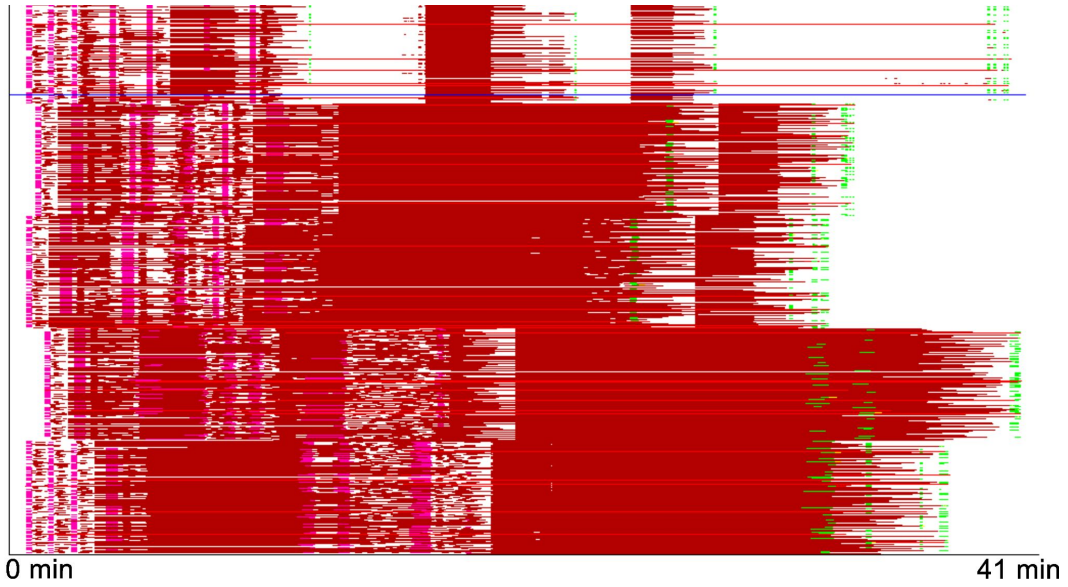
- First-level tasks dynamically generate finer-grained tasks.
- Multiple agents generate workload during execution.
- Largest configuration: 21,800 tasks.

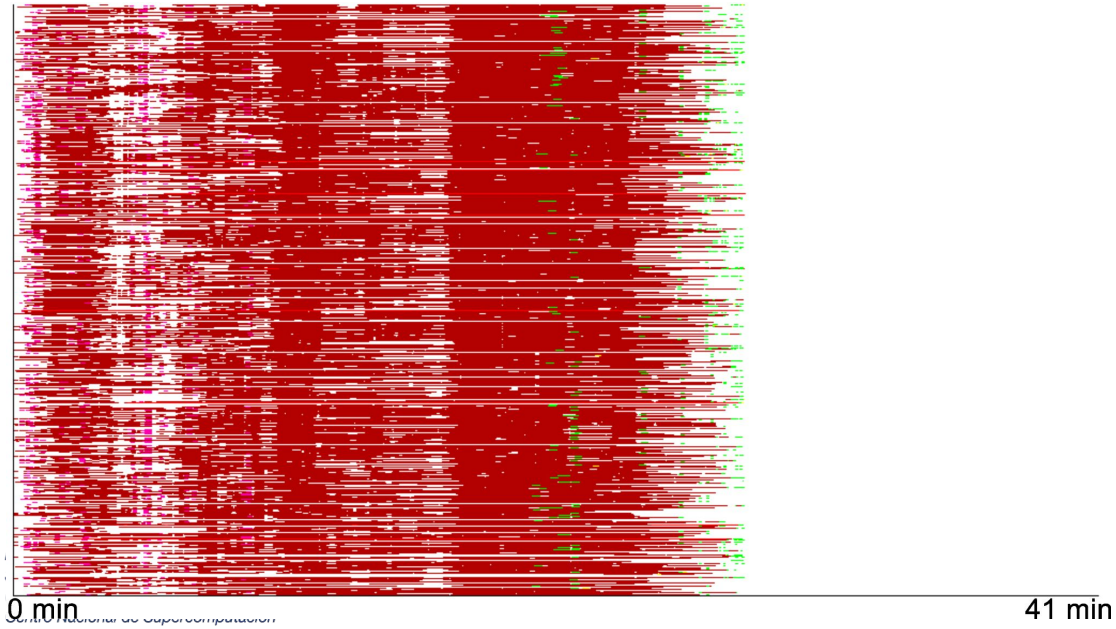
Comparison and configuration

- Consensus vs. hierarchical task-offloading baseline.
- Data blocks: 30, 50 and 100.
- Five repetitions per configuration.

Artifact available: https://github.com/xcasas/sc26_artifact
COMPSs Workflow Management System

Main result: faster execution and better load balance





WORKFLOW EXECUTION TIME

40 min 52 s → 32 min 32 s

20.4% reduction with consensus

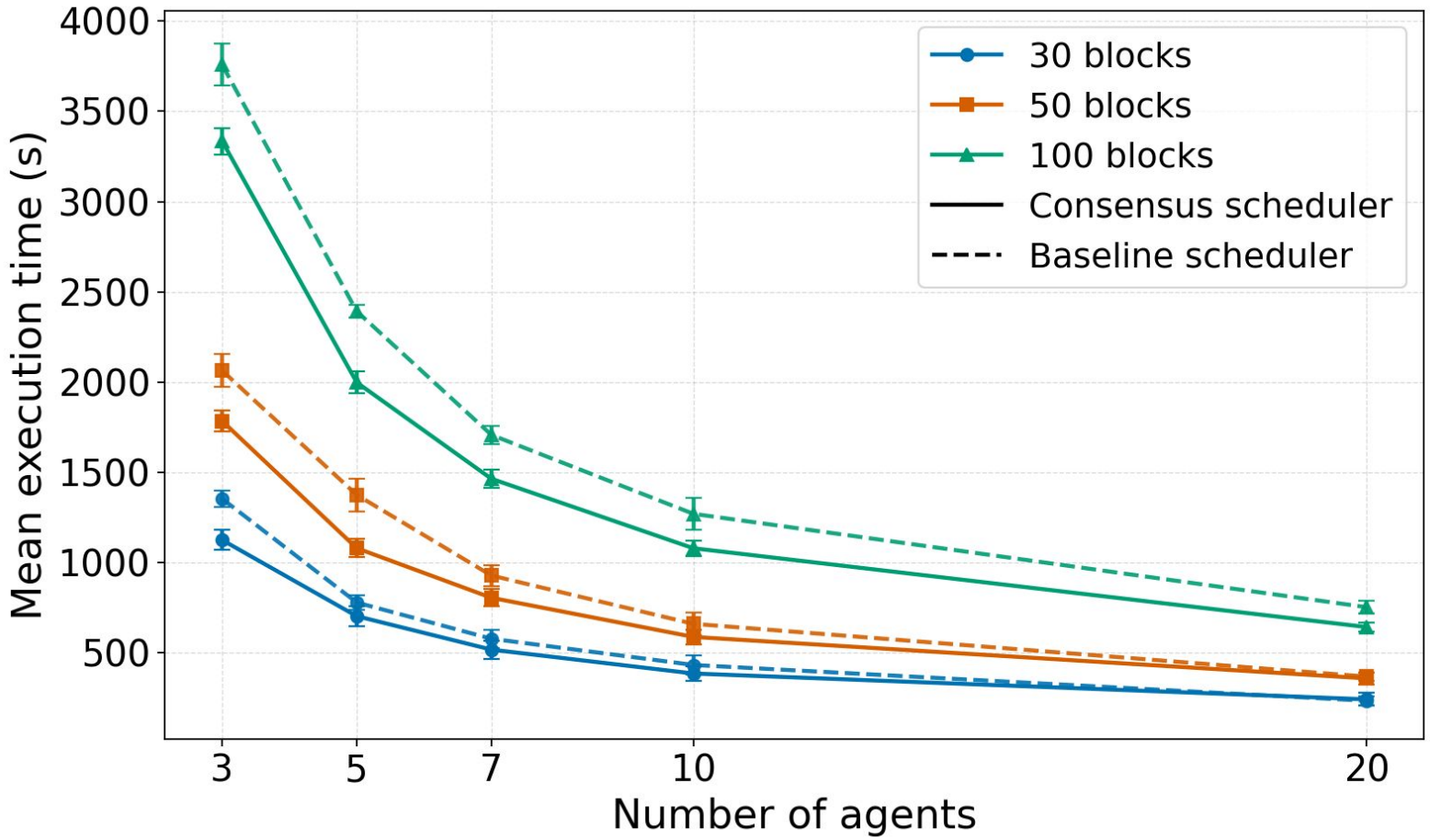
TASK DISTRIBUTION

19.1% → 3.1%

Difference between most- and least-loaded agents

Hierarchical offloading can leave agents idle because tasks at the bottom of the hierarchy cannot be redistributed. Consensus allocates across the full group.

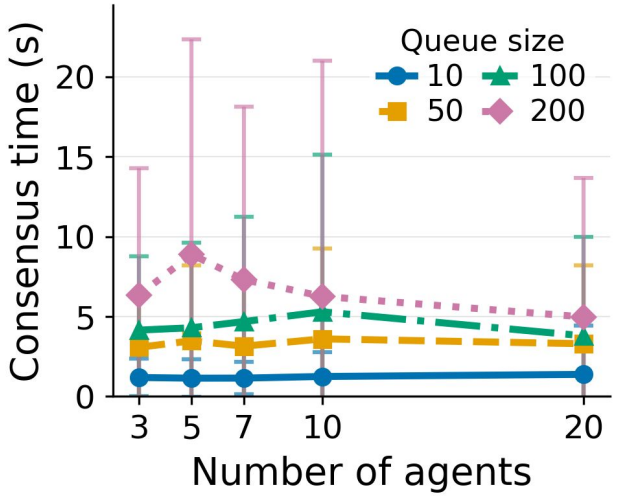
Scalability: execution time as the system grows

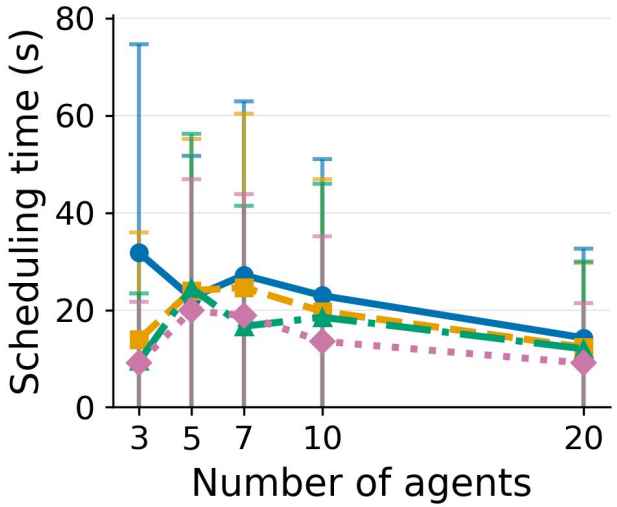


Mean execution time decreases as more agents participate. The consensus scheduler remains faster than the baseline across the tested data-block configurations.

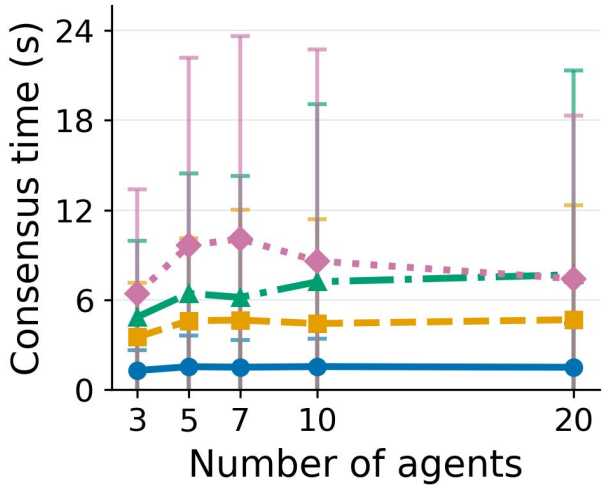
Consensus and scheduling overhead

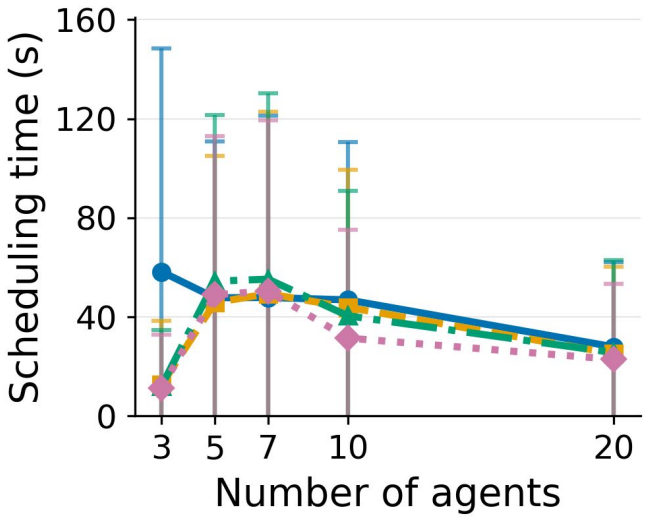
30 blocks (6717 tasks)



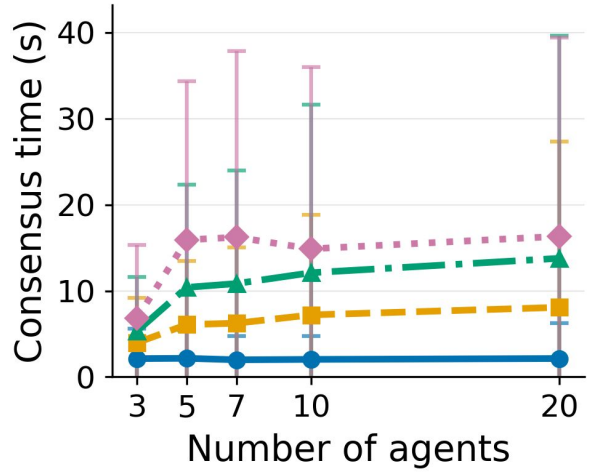


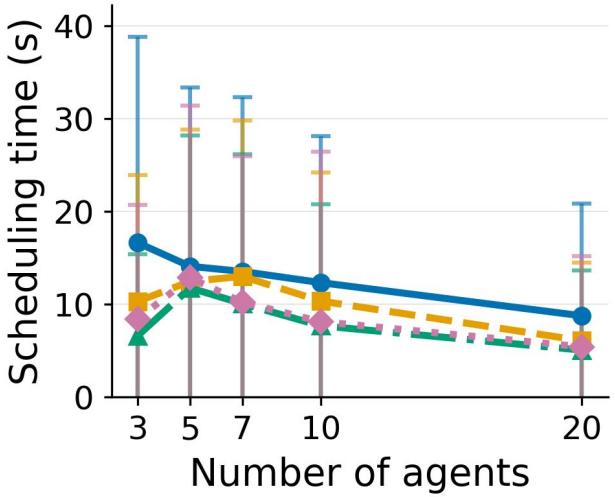
50 blocks (11076 tasks)





100 blocks (21800 tasks)





Conclusions and future directions

1. A general decentralized scheduling approach

Distributed task submission, collaborative allocation and remote execution without a central coordinator.

2. A consensus-based implementation in COMPSs

SwarmTS replaces shared global state with direct message-based coordination.

3. Experimental evidence

20.4% lower execution time and substantially improved workload balance.

FUTURE WORK

Parallel communication and batching · Hybrid coordination mechanisms · Heterogeneous Compute Continuum evaluation

Thank you! Questions?

This work is supported by the J**uan de la Cierva grant (JDC2023-051290-I)**. This work is also partially supported by Horizon Europe project **ODISEE (grant number 101188332)** and projects **CEX2021-001148-S** and **PID2023-147979NB-C21** funded by MCIN/AEI and **MICIU/AEI/10.13039/501100011033**, by FEDER, UE, and by the Departament de Recerca i Universitats de la Generalitat de Catalunya through the research group **MPiEDist (2021 SGR 00412).**

This work is supported in part by the U.S. Department of Energy under the Distributed Resilient Systems for Science program, **grant DE-SC0024387** and the National Science. Foundation under **grant #2513101**.



New version available!
COMPSs v4.0 Queen's wreath