Runnan QIN, Zhen YANG, Xiaodong PENG, Wenming XIE, Xiao ZHENG, Jingyi REN, Zeyu FAN, 2026. A microservice framework for data–model fusion in situational awareness simulations. *ENGINEERING Information Technology & Electronic Engineering*, 27(7):250154. https://doi.org/10.1631/ENG.ITEE.2025.0154

A microservice framework for data–model fusion in situational awareness simulations

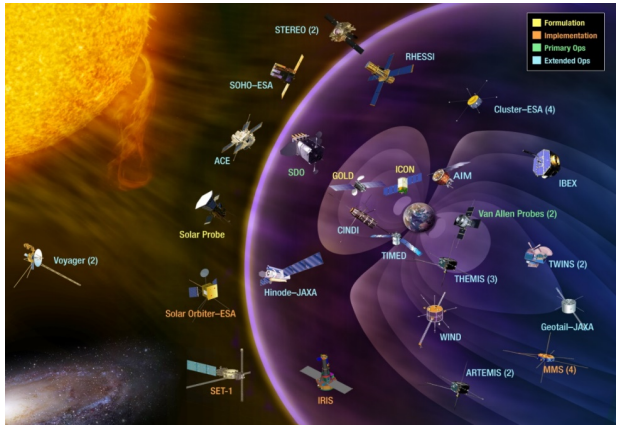
Key words: Data–model fusion; High-performance simulation computing; Scheduling strategy; Node load prediction; Microservice

Runnan QIN
E-mail: qinrunnan@nssc.ac.cn
ORCID: https://orcid.org/0009-0007-7668-2385

Motivation

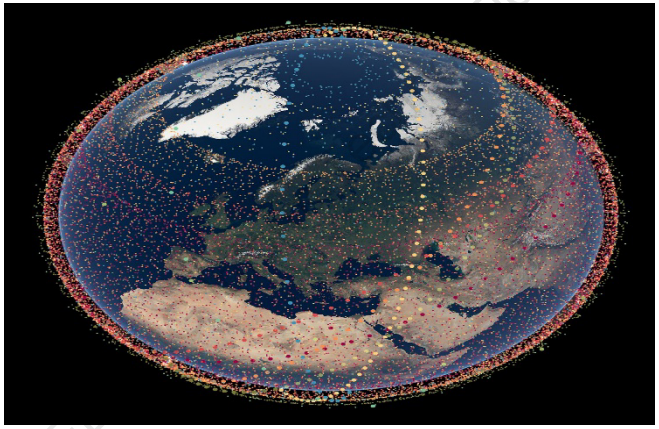
Background

As space missions continue to grow in originality, systematicity, complexity, precision requirements, and technical difficulty, the need for complex system simulation has become increasingly urgent.



Space Science

- High technical complexity



Large Satellite Constellation

- Complex coordination processes

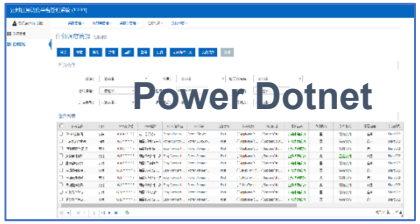
Space Reconnaissance/Resistance

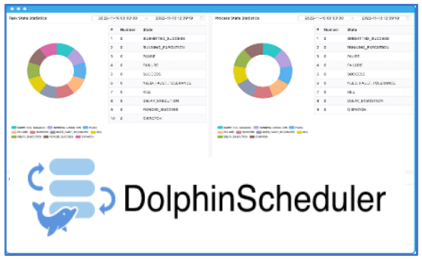
- High precision requirements

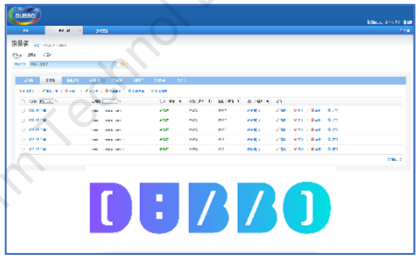
Situational awareness simulation requires systematic modeling of space entities, scenarios, and environments, calling for **data–model fusion** to integrate heterogeneous data and multidisciplinary models under stringent accuracy and real-time constraints.

Motivation

Main solutions: task scheduling systems





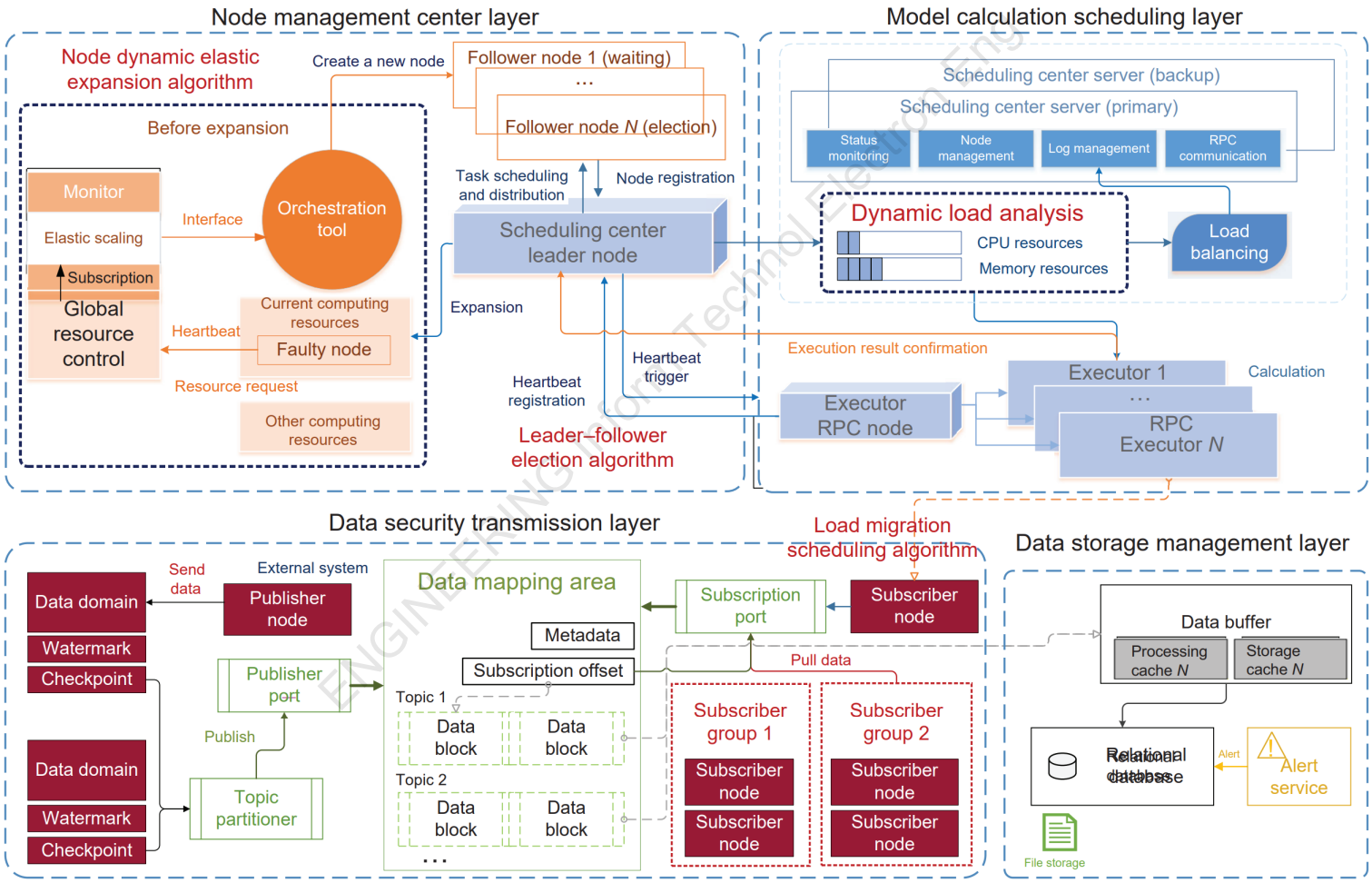


Three critical challenges

- **Management instability**
  Frequent leader elections cause service interruptions.
- **Burst load handling**
  Be unable to predict/proactively scale for large-scale computing tasks → node crashes.
- **Resource oscillation**
  Reactive load migration triggers unstable resource allocation and load oscillations.

Design

Proposed solution: MP-DMC framework

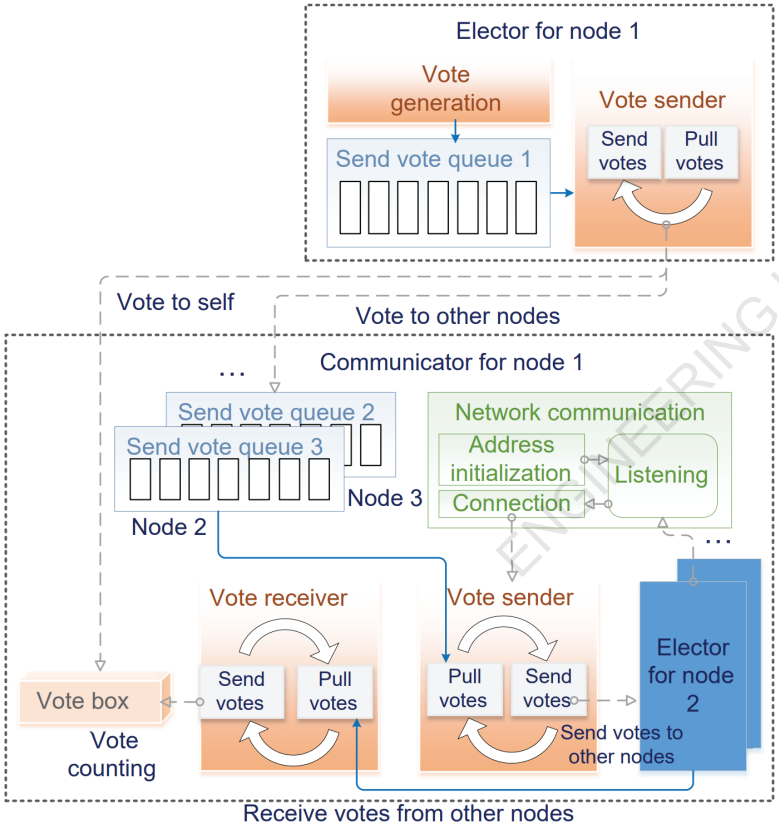


Overall architecture of the proposed microservice platform for data–model computing (MP-DMC), which adopts a four-layer distributed architecture where similar functional structures are expressed using the same color

Design

Distributed data–model fusion management and the scheduling strategy

(1) Leader–follower election algorithm



Node election process

Algorithm 1 Leader–follower election

Require: $N, Q_n\ (n = 1, 2, \ldots, N)$, θ, L_{thr}

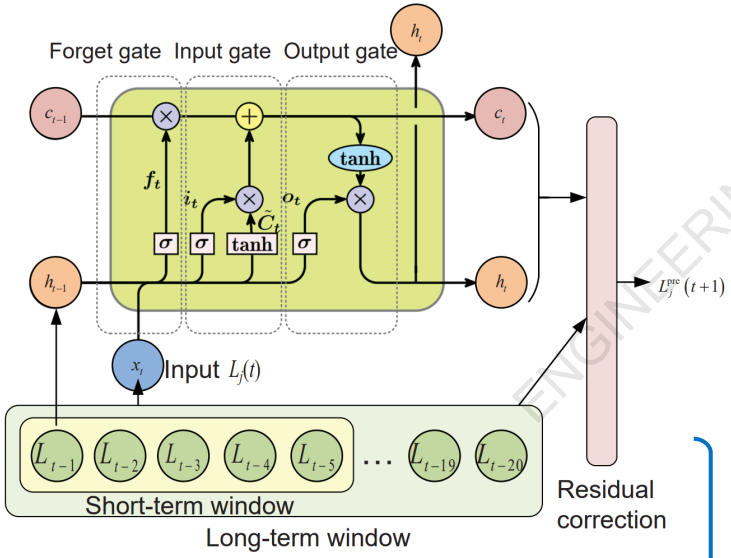
Initialize A_{election} and A_{wait}
Sort $Q_i\ (i = 1, 2, \ldots, N)$ by priority or load in ascending order
for $i = 1$ to N **do**
  if $|A_{\text{election}}| > \theta$ **then**
    Add Q_i to A_{wait}
  else if $|A_{\text{wait}}| > \theta$ **then**
    Add Q_i to A_{election}
  end if
end for
Timer → Multicast(Q_i, heartbeat message)
if $L_i > L_{\text{thr}}$ **then**
  Add Q_i to A_{wait}
end if
if Q_i has priority **then**
  Multicast(move over message) and wait for acknowledgments
  if Number of acknowledgments $> N/2$ **then**
    Q_i becomes the leader node
  else
    Multicast(end message)
  end if
end if
return Q_n

Design

Distributed data–model fusion management and the scheduling strategy

(2) Node dynamic elastic expansion algorithm

① DMA-LSTM model
-> Predict node load



Structure of DMA-LSTM model

② Dynamic node expansion

Algorithm 2 Dynamic elastic expansion algorithm

Require: $N, Q_n \ (n = 1, 2, \ldots, N)$

Sum $\leftarrow 0$
for $i = 1$ to N **do**
  Predict $L_i^{\text{pre}}(t+T)$ for node i using DMA-LSTM
  Sum $\leftarrow$ Sum $+ L_i^{\text{pre}}(t+T)$
end for
if Sum $> N$ **then**
  Create a new node Q_{new}
  Add Q_{new} to the expanding queue
end if
return $\left(Q_n \cup \{Q_{\text{new}}\},\ N+1\right)$

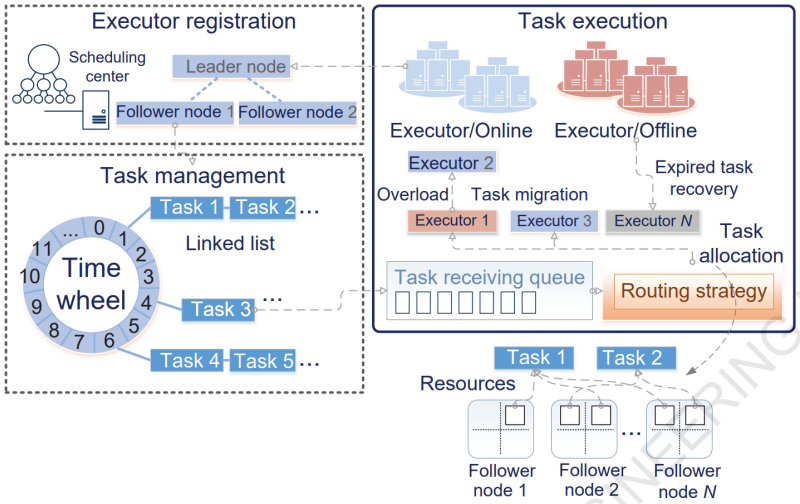
Lower (DMA):
Short-term (5 steps) + long-term (20 steps) moving averages → smooth baseline trends

Upper (LSTM):
Model prediction residuals → short-term corrections

Design

Distributed data–model fusion management and the scheduling strategy

(3) Load migration scheduling algorithm



Computation scheduling process

Algorithm 3 Load migration scheduling

Require: N, $L_n(t)$, $Q_n(t)$, and $T_n(t)$ $(n = 1, 2, \ldots, N)$

for all $0 \le t \le T$ **do**
  $u_{\text{thr}}(t) = \text{mean}\{1 - L_{i,j}(t)\},\ i, j = 1, 2, \ldots, N$
  $\Phi\{\text{desc}\} \rightarrow \text{sort}\ L_{i,j}(t)$
  for all $i, j = 1, 2, ..., N$ **do**
    $u_{i,j}(t) = (L_{i,j}(t) - L_{i,j}(t-1)) \,/\, (Q_i(t)T_j(t))$
    while $\Phi\{\text{inx}\}$ by $1, 2, \ldots, N$ **do**
      if $L_{i,j}(t) + u_{i,j}(t) \ge u_{\text{thr}}(t)$ **then**
        $u_{i,j}(t) = 0,\ h_{i,j}(t) = 1.0$
        $\rightarrow$ transfer to $u_{i+1,j+1}(t)$ by $\Phi\{\text{inx} + 1\}$
      else
        $h_{i,j}(t) = 0.5$
      end if
    end while
    $U(t)$ += $u_{i,j}(t)$
  end for
  $U^*(t) \leftarrow Q_i(t), L_{i,j}(t), T_j(t)$
end for
return $U^*(t)$

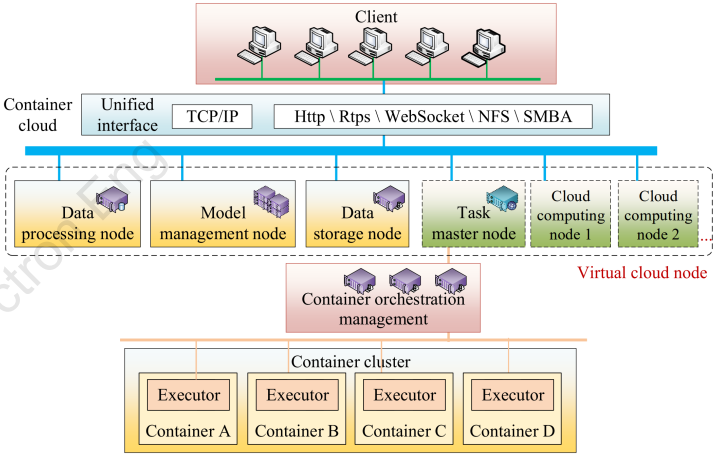
Key concepts:

- **TSW (Task Scheduler Weight):**
  $W_{i,j}(t) = h_{i,j}(t-1)\xi_i \left(L_j(t) - L_j(t-1)\right) / Q_i$
  (Lower TSW → node neglected → migration target)
- **Loss Function Minimization:**
  $U^*(t) = U(t-1) - \eta/2W(t)Q(t)T(t)$

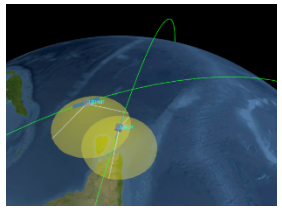
Experiments

Experimental environment

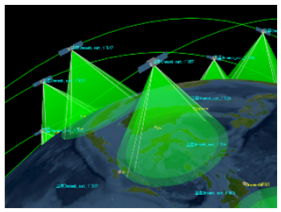
- **Hardware:** a cluster of 10 Zhongke Tenglong TL550F-B servers
- **Software:** China Galaxy Kirin Advanced Server Operating System V10 & China Shenzhou General Database V7.0
- **Architecture:** a distributed container cloud with dedicated core service and computational nodes



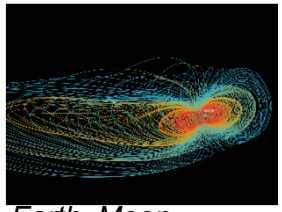
Hardware environment deployment diagram



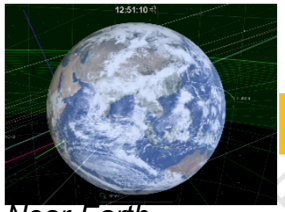
Spacecraft close-in



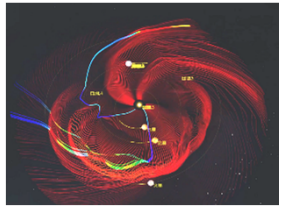
Spacecraft detection



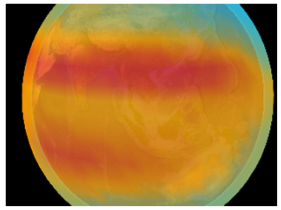
Earth–Moon magnetic field



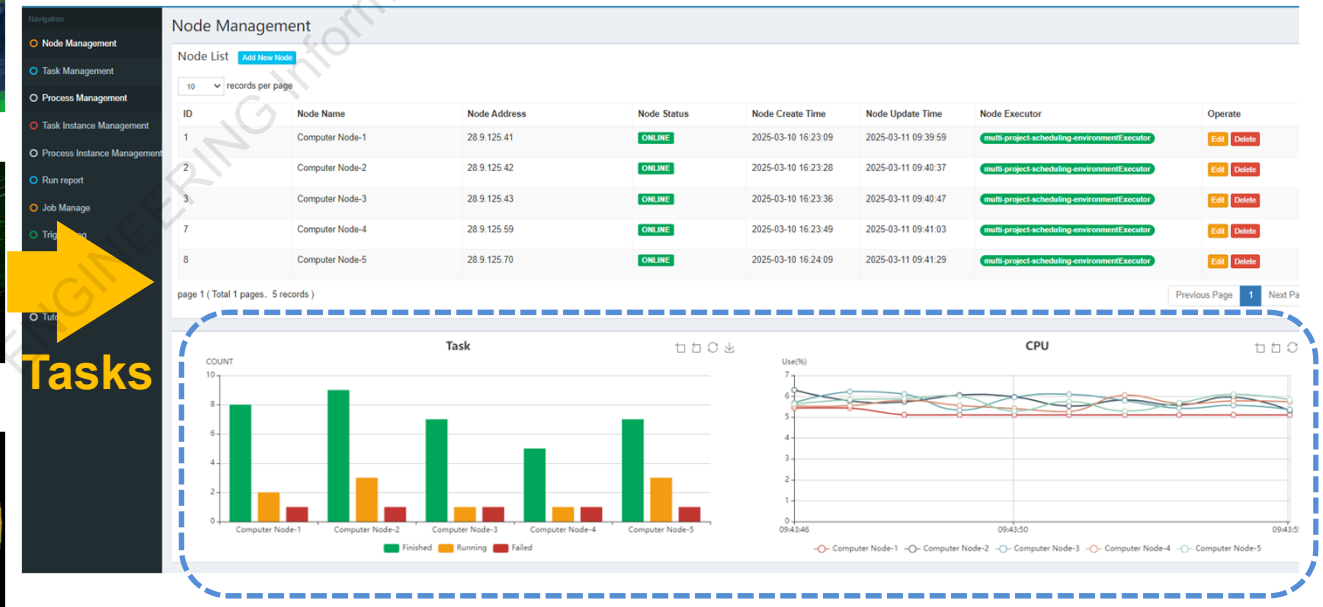
Near-Earth atmosphere



Earth–Moon solar wind



Near-Earth ionosphere



Client management interface of MP-DMC

Real-time load of nodes

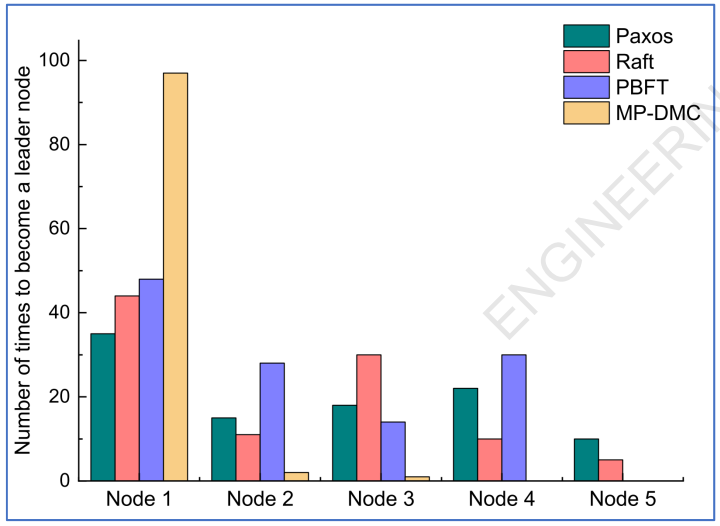
Experiments

Dynamic election analysis

Experiment setup

Table S1 Configuration of election algorithm parameters for the five-node experiment

Algorithm	Key parameter	Value
Paxos	Proposer timeout	200 ms
Raft	Election timeout	1500 ms
	Heartbeat interval	100 ms
PBFT	View-change timeout	1500 ms
	Quorum size	$3f+1$ (f=1 for 5 nodes)
MP-DMC	Heartbeat interval	100 ms



Comparison of election algorithms in terms of leader election stability

Table 1 Reliability enhancement effects of leader–follower election algorithms after different running times, from 24 h to 168 h

Node number	System	Number of service interruptions						
		24	48	72	96	120	144	168
5	MP-DMC w/o election	0	0	1	2	2	3	3
	MP-DMC*	0	0	0	0	0	0	0
20	MP-DMC w/o election	0	0	1	1	2	2	3
	MP-DMC*	0	0	0	0	0	0	0
50	MP-DMC w/o election	0	0	2	2	2	4	5
	MP-DMC*	0	0	0	0	0	0	1
100	MP-DMC w/o election	0	1	1	2	4	5	6
	MP-DMC*	0	0	0	0	1	1	1

w/o: without. MP-DMC* denotes the proposed framework with the election algorithm

Key findings:

- MP-DMC maintains node 1 as leader with 97% stability.
- Conventional algorithms distribute leadership probabilistically.
- Algorithm scales effectively to 100+ nodes.

Experiments

Elastic expansion performance

Experiment setup

Table S2 Hyperparameter configurations for baseline prediction models

Model	Key parameter	Value
Historical mean	Sliding window size	20 timesteps
SVM	Kernel	RBF
	Regularization coefficient	1.0
	γ	0.01
	Epsilon	0.1
LSTM	Layer	1 LSTM layer with 128 units
	Optimizer	Adam (lr=0.001)
	Batch size	64
	Number of epochs	100
	Input sequence length	20 timesteps
DMA	Short-term MA window	5 time steps
	Long-term MA window	20 time steps

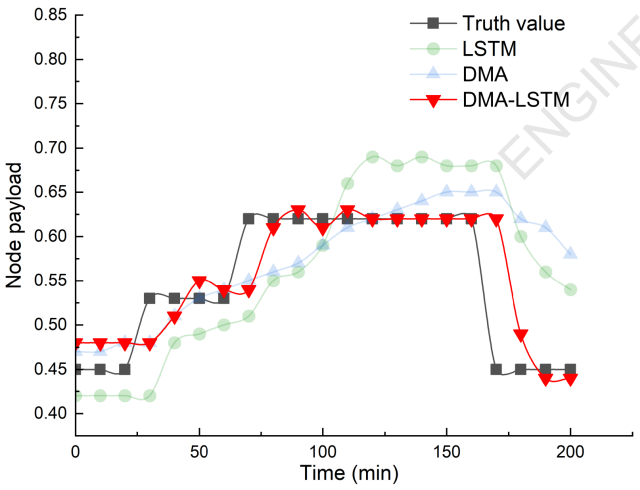
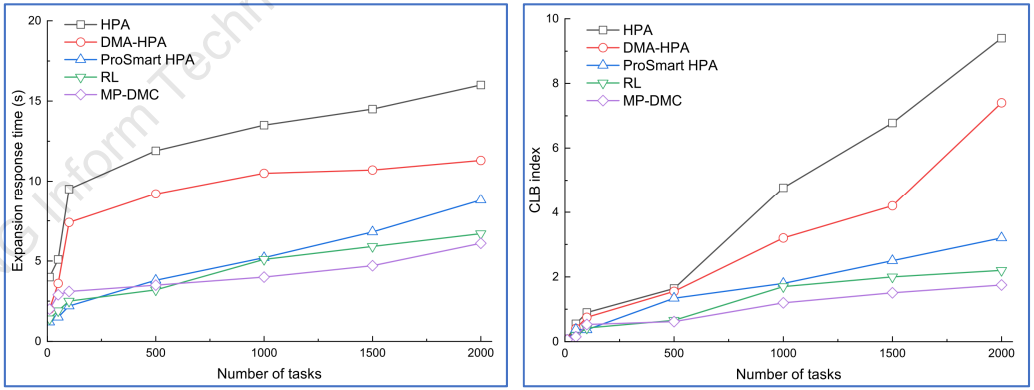


Table 2 Load prediction accuracy of models

Model	RMSE (%)	MAE (%)	MAPE (%)
Historical mean	12.56	9.45	13.23
SVM	6.84	5.22	7.41
LSTM	5.21	4.15	5.66
DMA	7.01	6.71	7.92
DMA-LSTM	3.82 ± 0.16	3.68 ± 0.11	4.51 ± 0.23



Comparison of elastic expansion performance under varying task workloads: (a) expansion response time; (b) CLB index

Key findings:

- DMA-LSTM captures spikes and fluctuations missed by baselines.
- ProSmart HPA and RL excel at small scales (<100 tasks); MP-DMC excels in large-scale, bursty workload scenarios.

Experiments

Migration scheduling effectiveness

Experiment setup

Table S5 Configuration of baseline scheduling algorithms

Algorithm	Key parameter	Value
Round-robin	Fixed time quantum	10 ms
	Queue order	FIFO queue
Purely random	Probability distribution	Uniform random
Weighted random	Weight assignment	Proportional to RLP
MP-DMC	Migration threshold	85% of RLP
	Migration decision	10 ms

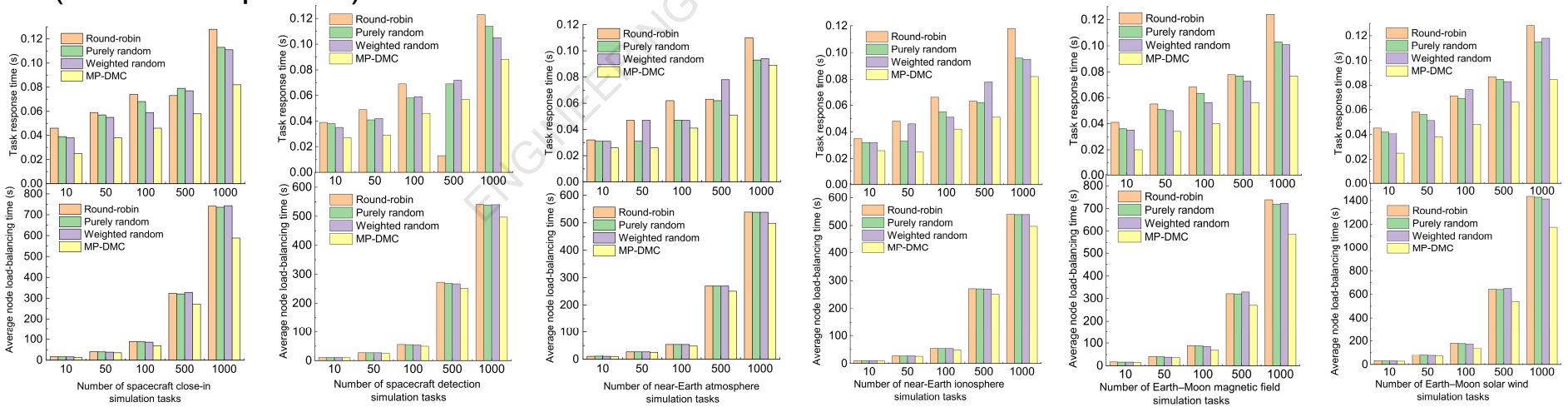
Table 4 Contributions of core MP-DMC components for different numbers of tasks

Configuration	Response time (s)			Load-balancing time (s)		
	10	100	1000	10	100	1000
Full	0.03	0.05	0.08	14.12	69.89	640.47
–E	0.03	0.05	0.11	16.15	84.27	752.13
–S	0.02	0.05	0.09	19.41	105.34	925.21
–M	0.02	0.05	0.11	24.39	132.52	1180.88

All values represent the mean over five independent runs. The standard deviations for response time and load-balancing time range from ±0.001 s to ±0.004 s and ±0.5 s to ±4.2 s, respectively (increasing proportionally with the task count)

Key finding:

Without migration: +84% load-balancing time (critical component)



Performance comparison of task scheduling algorithms. Each graph compares the task response time and node load-balancing time of scheduling algorithms under different simulation tasks

Conclusions

The proposed MP-DMC framework tackles cloud-simulation challenges through priority-based dynamic election that reduces leader-node changes by 97%, DMA-LSTM elastic scaling that achieves response times below 6 s for 2000 concurrent tasks, and TSW-prioritized migration that stabilizes load balancing quickly. Its predictor, which decouples trend and residual learning, is domain-agnostic and suitable for industrial IoT and cloud data centers. Future work will extend the framework to deep-space mission simulations and incorporate predictive failure models to enable proactive leader handover for enhanced robustness under extreme conditions.

Biography

Runnan QIN received the MS degree in Instrumentation Engineering from Beihang University, Beijing, China, in 2019. She is currently a development engineer in National Space Science Center, working toward the PhD degree at the University of Chinese Academy of Sciences. Her main research directions include software architecture, cloud computing, computer simulation technology, and space intelligence applications.