ICML
International Conference
On Machine Learning

UF

SPARe: Stacked Parallelism with Adaptive Reordering for Fault-Tolerant LLM Pretraining Systems with 100k+ GPUs

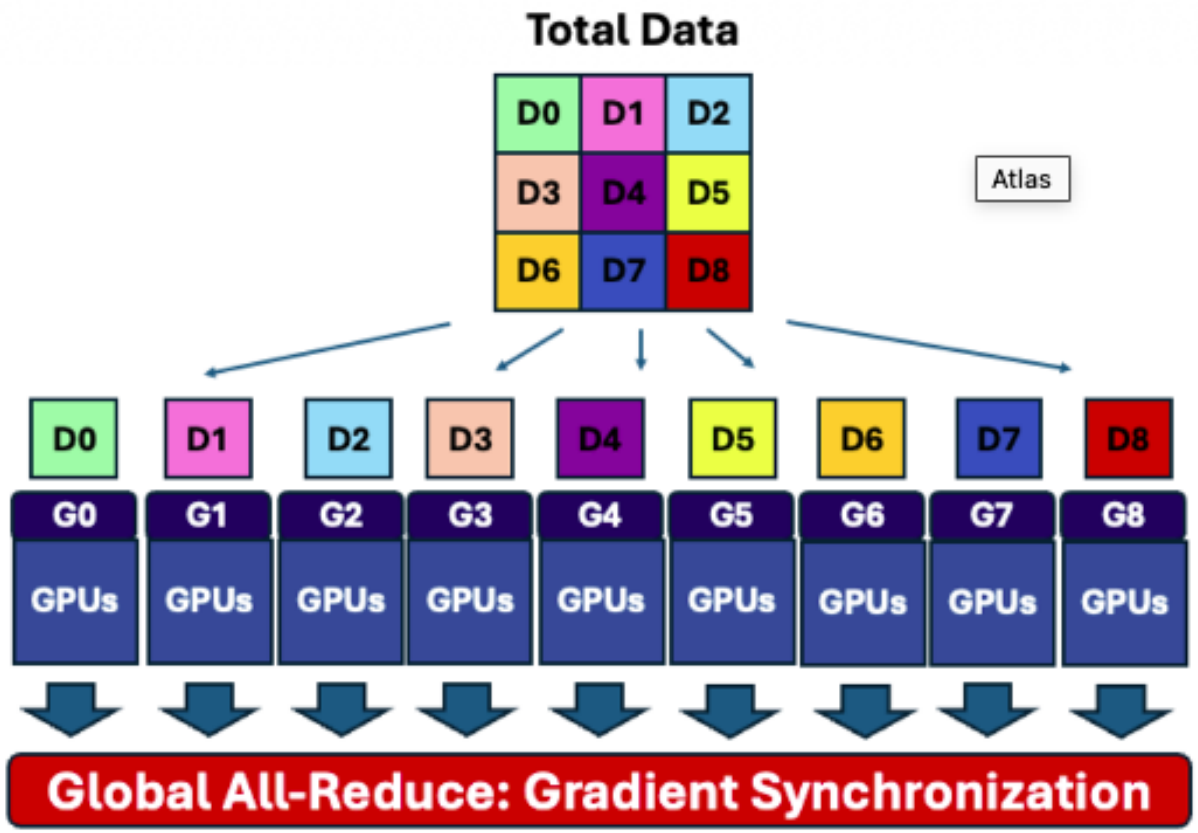
Jin Lee[1,2] Zhonghao Chen[3] Xuhang He[3] Robert Underwood[2] Bogdan Nicolae[2]
Franck Cappello[2] Xiaoyi Lu[3] Sheng Di[2] Zheng Zhang[1]

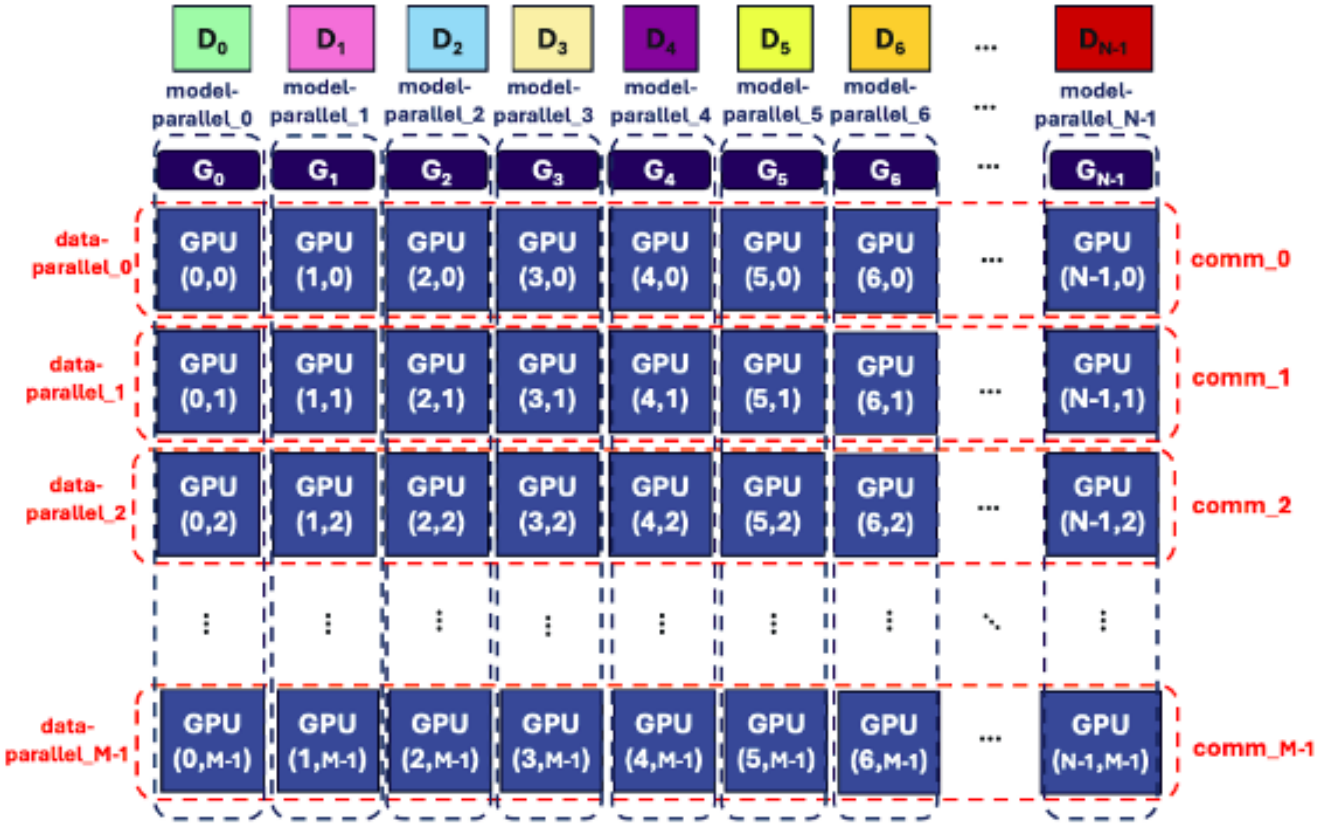
[1]University of California, Santa Barbara [2]Argonne National Laboratory [3]University of Florida

Motivation: Restart-Dominant Regime

- ❑ Failure rate scales up linearly by #GPU
- ❑ Restart cost scales up linearly or worse by #GPU
- ➢ Restart cost is projected to dominate wall-clock training time in near future: Restart-dominant regime is imminent for 100k+ GPUs

System Model: Synchronous Data Parallelism





- ❑ All types of partial gradients, each from each data shard, must be collected for one optimizer step to be completed
- ❑ Loss of any model-parallel group (DP replica) leads to a restart

Traditional (naïve) Replication

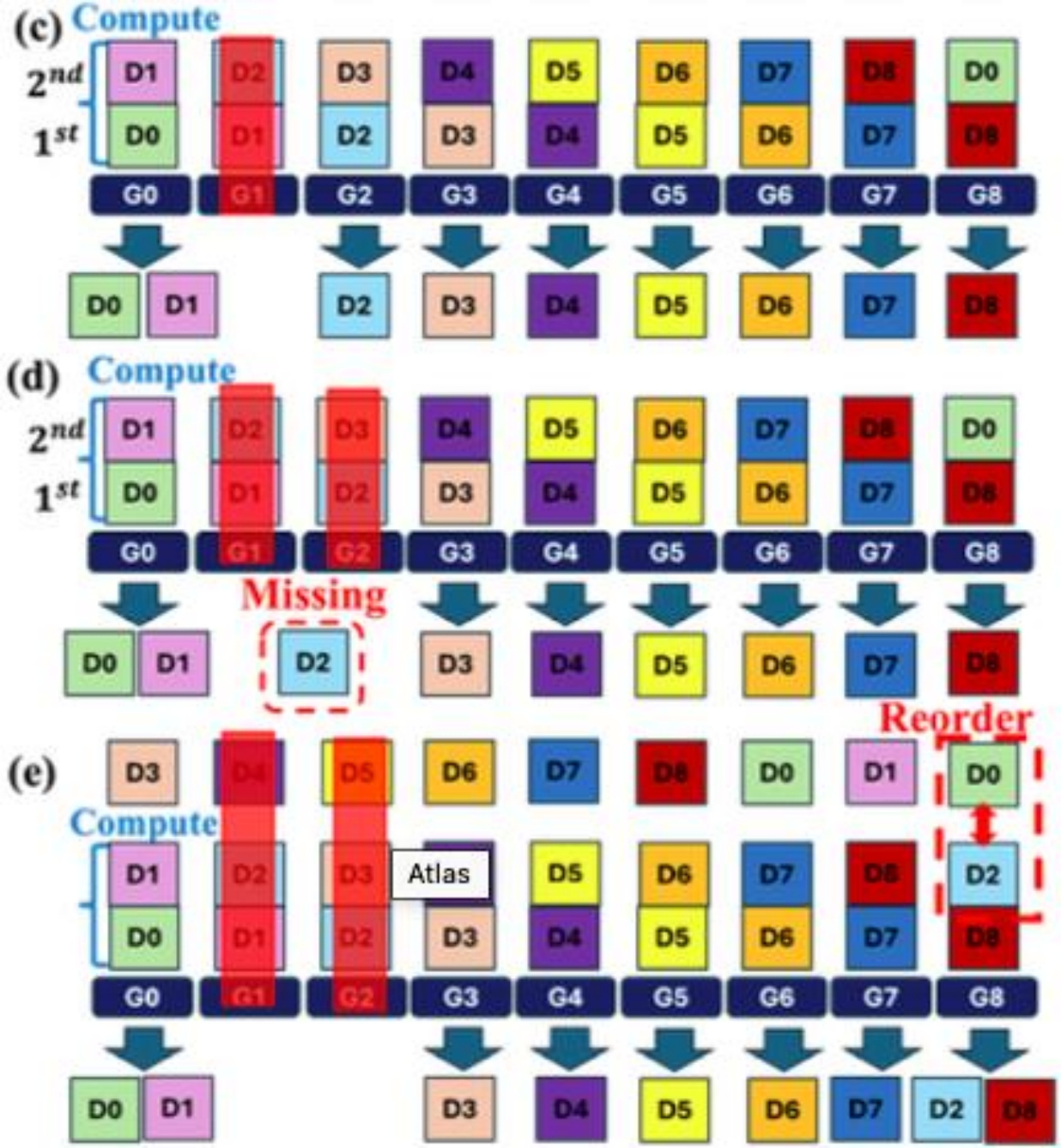
- Naïve replication endures a large number of failures by masking replica losses until extinction, massively improving system availability
- However, naïve replication incurs linear computation overhead, prohibitive in practice: redundancy $r \Rightarrow r\times$ overhead

Key Challenge

Can we mask frequent failures with redundant computation while keeping the overhead down to be near-constant?

SPARe: Stacked Parallelism with Adaptive Reordering

- SPARe replicates *shards* of computation (data) instead of fully replicating each group
- SPARe stacks them across parallelism groups so that all types of partial gradients can be collected *before* computing all assigned stacks
- SPARe adaptively reorders the shard stacks by failures so that each training step can be completed within the *minimal* number of stacks



Theoretical Analysis of SPARe

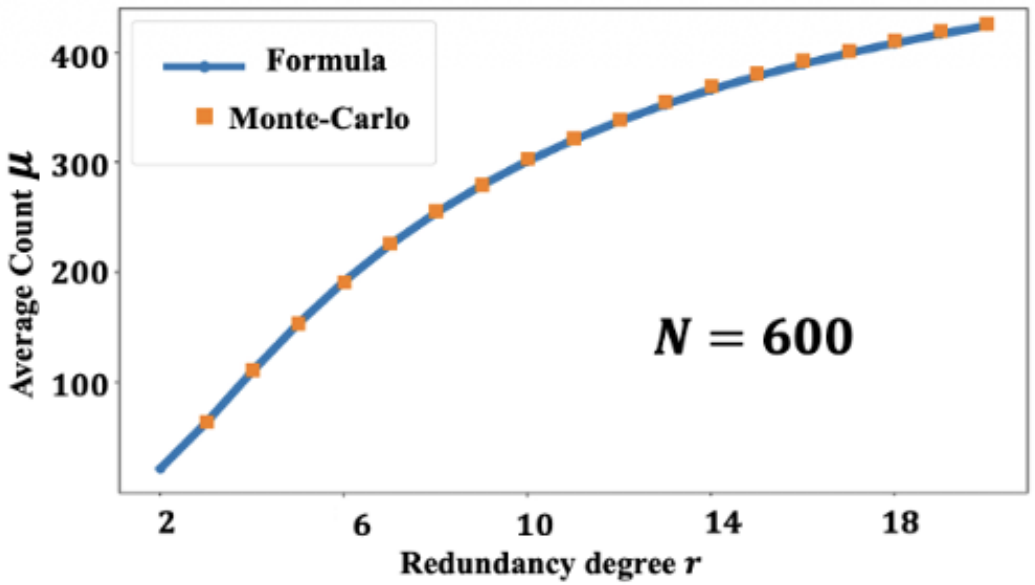
Q1) How many failures can SPARe endure before a restart?
Q2) How much computation overhead SPARe needs to pay?
Q3) What is the optimal redundancy for minimal time-to-train when merged with checkpointing?

A1) SPARe can asymptotically endure as many failures as traditional replication: Theorem 4.1

Theorem 4.1 (Average Failure Count). *The average failure count $\mu(N, r)$ SPARe can mask before the first wipe-out is asymptotically:*

$$\mu(N, r) \approx \boxed{\frac{\Gamma(1/r)}{r} N^{1-1/r}}, \quad (3)$$

where Γ refers to the Gamma function in NIST (2025).



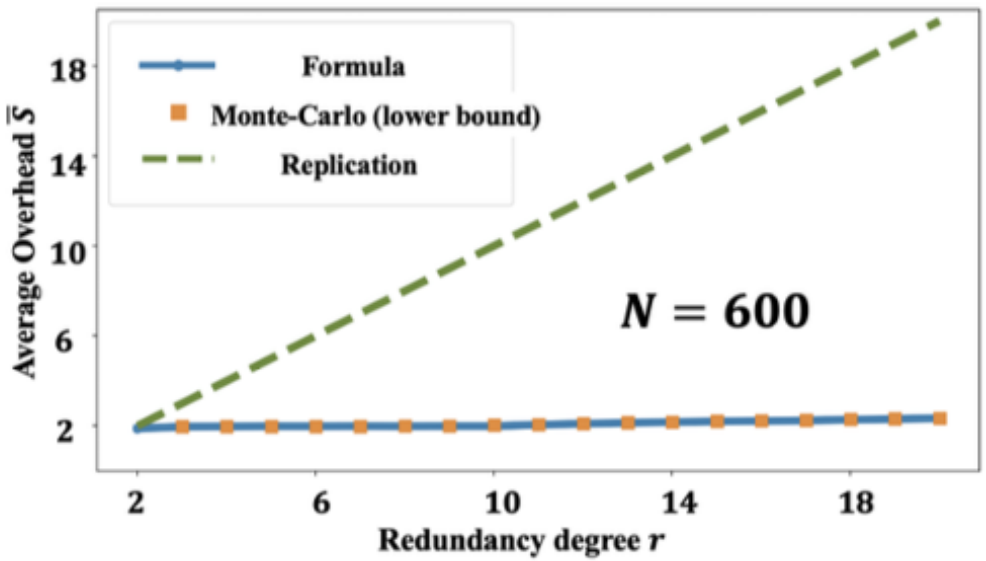
➢ SPARe improves availability as much as naïve replication

A2) SPARe can achieve near-constant computation overhead close to its theoretical lower bound: Theorem 4.2

Theorem 4.2 (Average Computation Overhead). *The average computation overhead $\overline{S}(N,r)$ SPARe needs to pay before the first wipe-out is approximately*

$$\overline{S}(N,r) \approx \boxed{\frac{1}{\lfloor \mu \rfloor} \sum_{k=0}^{\lfloor \mu \rfloor - 1} \Big(c(k) + \rho_k \Big)}, \qquad (5)$$

where μ is the average failure count of Thm. (4.1) and $c(k)$ is the lower-bound of all-reduce stack: $c(k) := \left\lceil \frac{N}{N-k} \right\rceil$. ρ_k is the probability of patch compute at k failures, $\rho_k := \max\{0,\, 2N - n_k\}/n_k$, where $n_k := c(k)(N-k)$.



➢ **SPARe retains 2~3× overhead even for large redundancy**

A3) Under exponential distribution: Theorem 4.3

Theorem 4.3 (Optimal $r^\star$ for minimal time-to-train). *Using checkpointing period of Eq. (1), SPARe+CKPT achieves minimal time-to-train at optimal redundancy $r^\star$:*

$$r^\star \approx \boxed{\lfloor \log_2 N + 0.833 \rfloor}. \qquad (8)$$

- **For simulations, realistic Weibull distribution is used**

Discrete-Event Simulations

- Evaluated using core components of **FedDES**, which is a **discrete-event simulation** toolkit built on top of **SimGrid**

Realistic system parameters for a projected 600k H100 cluster

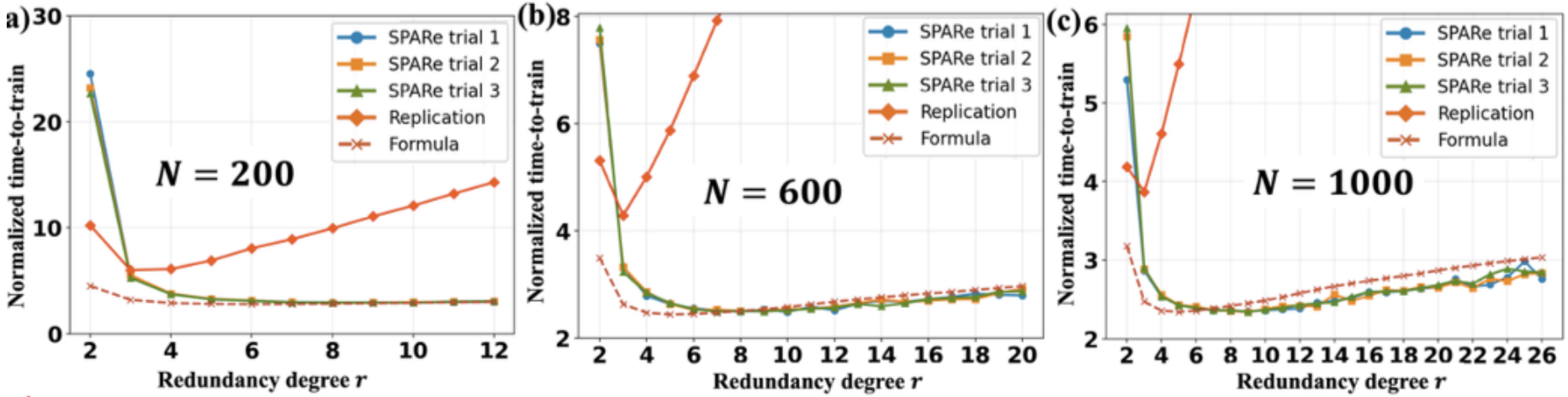
Parameter	Setting
Failure	MTBF 300s, $k = 0.78$ (Weibull)
Global Restart	$T_r = 3600$s
Model size	10T params (20TB)
DP groups	$N \in \{200, 600, 1000\}$
Data/stack	256M Tokens (4×64M)
Compute/stack	$T_{comp} = 64$s per stack
Train Horizon	$T_0 = 10{,}000$ steps $\times (T_{comp} + T_a)$
All-Reduce	$T_a = 2, 6, 10$s at each N
Failed All-Reduce	50% of All-Reduce, $0.5 \times T_a$
Comm. Shrink	0.1s
CKPT time	$T_s = 60$s
Event Jitter	$\times \mathcal{N}(1, 0.05^2)$ on all events

Simulation Results

N	Rep+CKPT		SPARe+CKPT			Gain
	time-to-train/T_0	Availability	time-to-train/T_0	$r^\star$	Availability	[%]
200	6.07	61.74%	2.92	9	87.00%	51.9%
600	4.27	79.89%	2.49	8	93.90%	41.7%
1000	3.88	84.41%	2.34	9	96.54%	39.6%

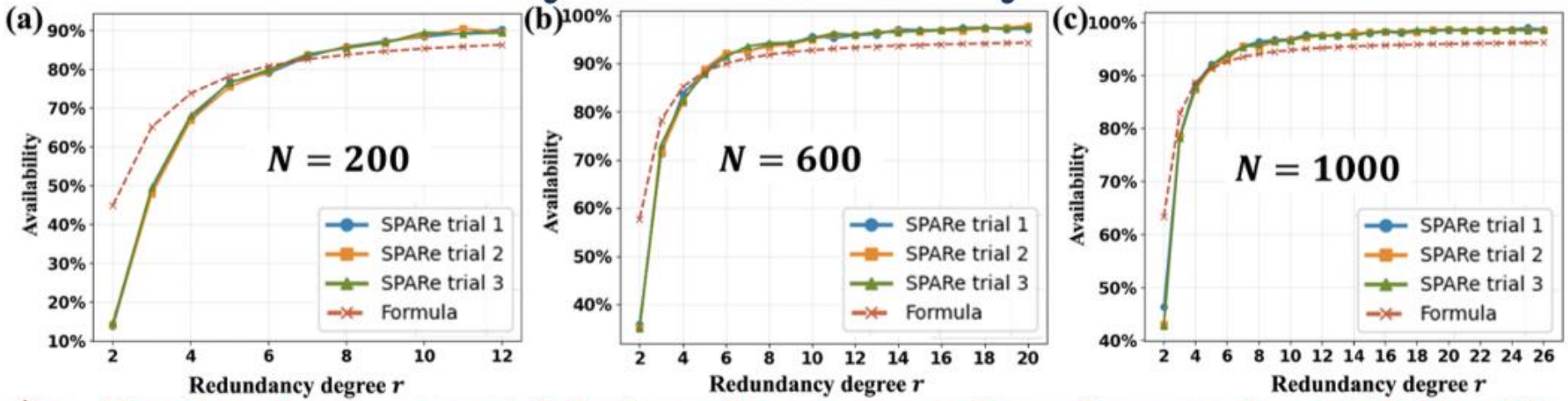
➢ **SPARe reduces TTT by 40~50% w.r.t. naïve replication**

Time-to-train (Normalized)



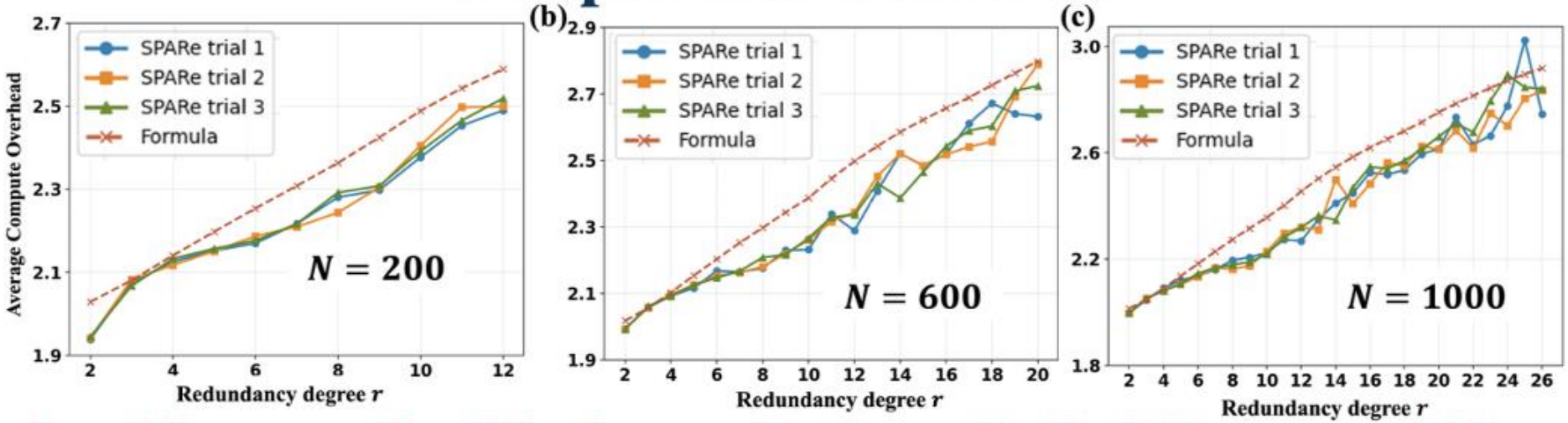
➢ **Better/worse at higher/lower redundancy by Weibull**

System Availability



➢ **Better/worse at higher/lower redundancy by Weibull**

Computation Overhead



➢ **Aligns well with theoretical analysis (Theorem 4.2)**