Parallelism Taxonomy

Deep Dive into SGLang — Chapter 22

Yin Yang

Foundation Books Project

Where this chapter sits

Parallel serving spreads model state, token work, and communication across ranks — but the user still expects one ordered request stream.

- A throughput claim is meaningful only after the runtime records *where state lives*, *where work runs*, *what communication was introduced*, and *when* shard-local results become one step.
- TP, PP, DP, CP, and EP differ most by ownership: who owns state, who communicates, where the logical step becomes whole again.
- The taxonomy separates *capacity scaling* from the *correctness obligations* placement brings.

Goal: read a parallel deployment and name what was split, where it rejoins, and which request pays if the ranks disagree.

Roadmap

Placement and ownership

Groups, execution, invariants

Placement and ownership

Parallelism is a placement problem

A serving optimization is a placement plan

$$\Gamma = (\mathcal{Q}, \rho_{\text{state}}, \rho_{\text{work}}, \rho_{\text{comm}})$$

assigning model state, token work, and communication to ranks $\mathcal{Q}$ *while preserving one request-visible step*.

- ρ_{state}: which weights, KV, experts, or metadata a rank owns.
- ρ_{work}: which token rows, spans, experts, or layers a rank runs.
- ρ_{comm}: the collectives and waits before rejoin.

The plan is the unit of reasoning. Five modes are just different answers to "which object was cut, and where does it become one value again?"

The failure the whole chapter guards against

Two decode rows A, B reach a row-parallel layer on two TP ranks. Each rank holds a partial sum; the rejoin is an all-reduce:

$$Y = \underbrace{\begin{bmatrix} y_A^{(0)} \\ y_B^{(0)} \end{bmatrix}}_{\text{rank 0}} + \underbrace{\begin{bmatrix} y_A^{(1)} \\ y_B^{(1)} \end{bmatrix}}_{\text{rank 1}} = \begin{bmatrix} y_A^{(0)} + y_A^{(1)} \\ y_B^{(0)} + y_B^{(1)} \end{bmatrix}.$$

If rank 1 stores $[y_B^{(1)}, y_A^{(1)}]$ instead, the shape still matches — the collective just sums partials from *different* logical requests.

Shape agreement is not row agreement. Every rank must map local batch position i to the same request.

Five modes, one contract

Mode	Owns (rank-local)	Rejoin condition
TP tensor	tensor / head slice	all-reduce or gather a layer
PP pipeline	layer interval	activation crosses a stage in order
CP context	sequence segment	restore token order, visible KV len
EP expert	physical experts + routed rows	all-to-all returns rows
DP data	a full replica + local requests	routing / metrics align

Communication follows the rejoin condition, not the acronym: TP all-reduces partials; CP re-orders positions; EP dispatches then combines; DP adds routing before any layer collective is even interpretable.

Groups, execution, invariants

Rank spaces become process groups — some are aliases

`initialize_model_parallel` builds the TP group first, then derives attention-CP, attention-TP, MoE-DP/EP/TP, and PP from the same rank space.

- A conceptual label may *alias* an existing communicator when it reuses the same ordered ranks for the same tensor meaning.
- Alias validity needs $\mathcal{Q}_a = \mathcal{Q}_b$ *and* matching collective semantics — sharing a rank count is not enough.
- TP $= 2$ and EP $= 2$ are *not* the same group if EP moves different request rows.

Constructing a communicator has cost, so equivalent roles reuse one; the accessors encode both the label and the communicator that owns the ranks.

The rank-local execution boundary

Launch fixes labels; then a scheduler batch $\mathcal{B}_t$ is converted to

$$X_q = (T_q, \Gamma_q, \mu_{t,q}, \mathrm{mode}_t),$$

the tensors, rank labels, groups, and forward-mode metadata one model runner can execute without changing the global plan.

- **Request-row alignment**: batch index i names the same $(r_i, \ell_i, \mathrm{kv}_i, \mathrm{sample}_i, \mathrm{out}_i)$ on every rank.
- The scheduler thinks in request rows; each worker sees a rank-local tensor program — the boundary must not lose the row identity.

Not every shard is a smaller tensor: a DP replica owns full weights but only rank-local requests, cache pages, and metric contribution.

Two invariants that keep the account honest

Invariant (Parallel step-alignment)
All ranks in one request agree on request order, sequence lengths, and collective membership before the step begins.

Invariant (Rank-local work equivalence)
Rank-local compute, communication, synchronization, metadata, ordering, and sequence-length agreement add back to the same logical step as the unsharded execution.

Splitting work is not an optimization unless the recomposed step is equivalent. Everything downstream — cost, speedup, capacity — is measured against that equivalence.

From concept to code: named groups and the TP rejoin

The placement tuple is runtime state, not one file. Groups are built and named; the TP rejoin is a collective on *that* group.

from **python/sglang/srt/distributed/parallel_state.py**

```
if attn_cp_size == tensor_model_parallel_size:
    _ATTN_CP = _TP                  # alias: same ranks, same meaning
else:
    _ATTN_CP = init_model_parallel_group(group_ranks, ..., group_name="attn_cp")
```

from **python/sglang/srt/distributed/communication_op.py**

```
def tensor_model_parallel_all_reduce(input_):
    return get_tp_group().all_reduce(input_)   # rejoin on the TP group
```

The collective is correct only for the group that owns the ranks.

Cost account and placement audit

One logical step decomposes as

$$C_{\text{step}} = \max_q C_{\text{local}}^{(q)} + C_{\text{comm}} + C_{\text{sync}},$$

critical-path local compute, cross-rank movement, and agreement cost.

- Partitioning lowers *per-rank memory* but adds collectives, metadata, and failure surfaces — modes fail in different places.
- A lower footprint is not a speedup if C_{sync} dominates the decode step.

A **placement audit** $A_{\text{place}} = (\Gamma, \mathcal{G}, S, \mu, \chi, C)$ records rank map, groups, owned state, request metadata, collective membership, and per-rank cost before any speedup or capacity claim counts.

What to carry forward

1. Parallelism is **placement**: Γ assigns state, work, and communication while preserving one request-visible step.
2. Name the mode by **ownership and rejoin**, never the acronym alone; the communication pattern follows the rejoin condition.
3. **Groups can alias** only on identical ordered ranks and tensor meaning; a rank-count match is not enough.
4. Two invariants gate everything: **step-alignment** and **rank-local work equivalence**.
5. A **placement audit** — not a bare throughput number — is the chapter's deliverable.

Next: Section 1 makes the placement plan concrete on two TP ranks — the shape matches, but only the rows carry the request.