#FMS2026

FMS
FUTURE OF MEMORY AND STORAGE

Towards fault-tolerant quantum computing through the lens of quantum memory

Roman Pletka, Senior Research Scientist
Contributors: Dionysios Diamantopoulos, Haris Pozidis

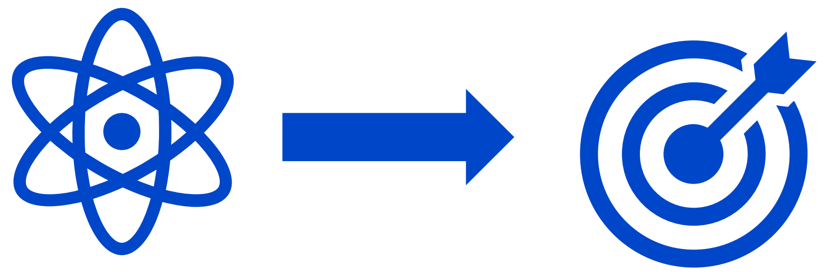
IBM **Quantum**

Towards fault-tolerant quantum computing (FTQC)

Biggest challenges in quantum computing today is quantum error correction (QEC):

1. Qubits are extremely noisy

Qubits suffer from
- Bit flips (X)
- Phase flips (Z)
- Imperfect gate operations
- Measurement errors
- Decoherence
- Leakage



$\approx 10^{-3}$ to 10^{-4} → $\approx 10^{-12}$

2. Coherence time and operating speed

- Qubits gradually lose their quantum information because of interactions with their environment.
- Longer coherence allows more computations before error correction becomes necessary.

	Typical decoherence	Operating speed
Superconducting	~50 – 300μs	50 – 100ns
Trapped ion	Milliseconds to seconds	Microseconds (~20μs)
Silicon / Spin	Microseconds to milliseconds	Nanoseconds
Neutral atoms	~ 1 – 5s	Hundred's of μs to milliseconds

3. Fast decoding

QEC requires decoding measurements in real time:
- Collecting large amount of syndrome measurements
- Identify likely error patterns
- Apply corrections quickly (within sub-microseconds)

Decoders use classical computation and must keep up with QPUs, hence build an essential part of fault-tolerant quantum computers.

Comparison of classical and quantum error correction

Error correction in classical storage/memory systems and quantum systems share high-level goal to preserve information reliably but differ profoundly in assumptions about physics, observability and allowable operations:

	DRAM	NAND Flash	Quantum
Information	Bits storing 0 or 1		Qubits can be 0 and 1 or combination of both (superposition + entanglement)
Capacity Scaling	Adding bits linearly increases storage		Adding qubits increases capacity exponentially
Error model	Bit flips, occasional burst errors		- Bit flips (X) - Phase flips (Z) - Decoherence - Leakage
Redundancy	Encode multiple bits using parity		Logical qubits encoded into many physical qubits (1:10-100)
ECC algorithms	Hamming, Hsiao, RS codes single-bit error correction and double-bit error detection	RS, BCH, LDPC codes, today correcting up to 1% error rate	Shor codes, surface codes, qLDPC codes, ... Fault-tolerance threshold 0.1 – 1%
Error detection	Non-destructive reads of parity checks, checksum mismatch, ECC syndrome bits		Measurement collapses qubit state -> Stabilizer measurements

Error correction fundamentals

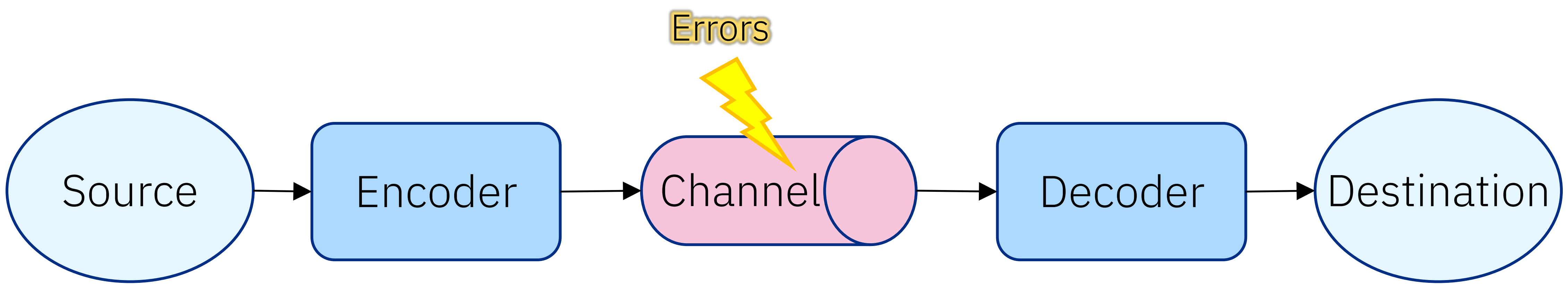
Goal: Add redundant information intelligently to detect and correct errors with probability.

Key ECC parameters [[n, k, d]]:

- **n:** Total number of bits/qubits in a code block (total length)
- **k:** Number of logical encoded bits/qbits being protected (data length)
- **d:** Code distance, a quality metric of the code denoting the minimum number of bit flips required to change one valid encoded message into another valid message

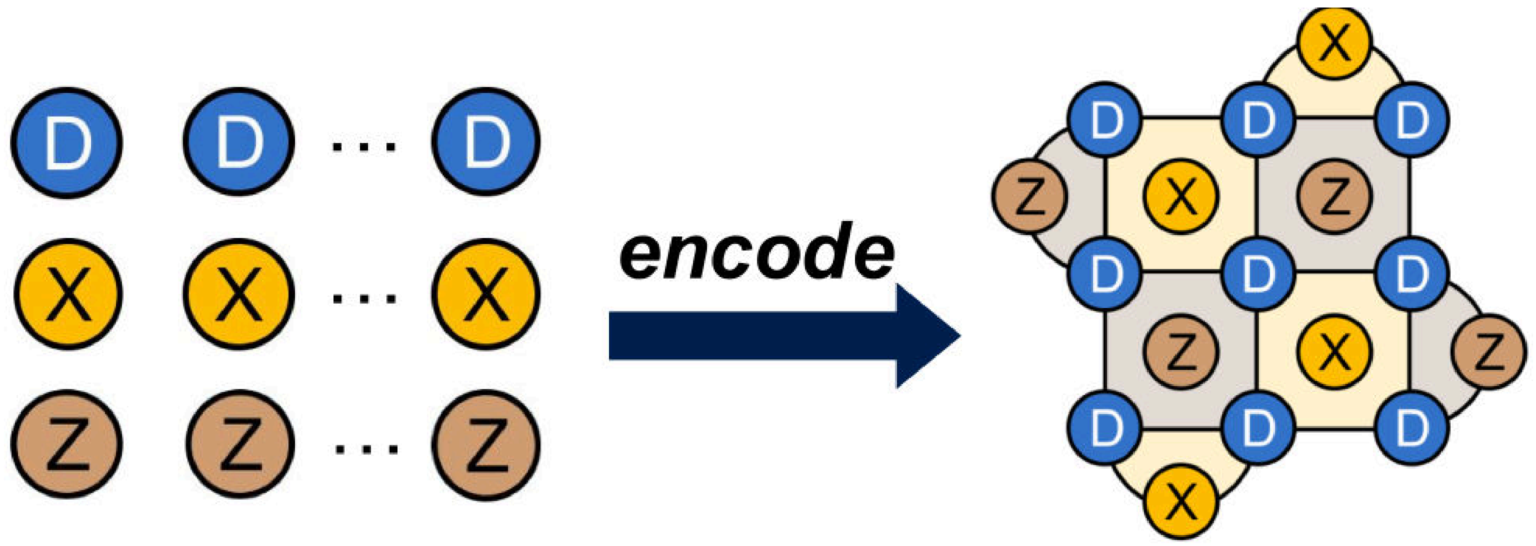
Properties

- Detection: up to $d - 1$ errors
- Correction: up to $\lfloor (d - 1) / 2 \rfloor$ errors
- Encoding rate: k / n

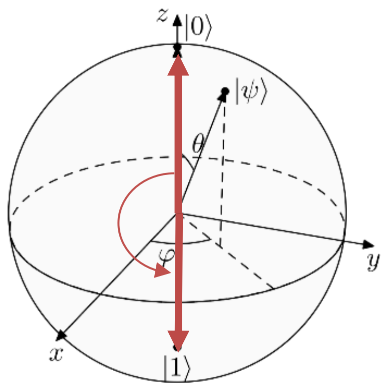


Quantum error correction fundamentals

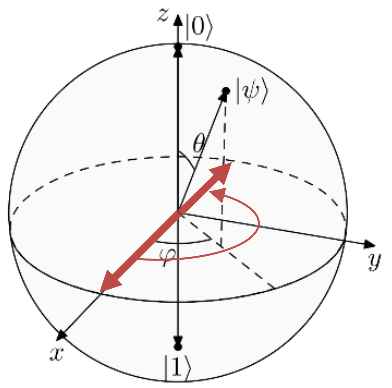
- Classical computers use basic replication to protect against errors.
- No cloning theorem in quantum:
  It is impossible to create an independent and identical copy of an arbitrary unknown quantum state.
- Therefore, quantum computers encode logical information of a single qubit into an entangled, multi-qubit state (no duplicating the qubits).



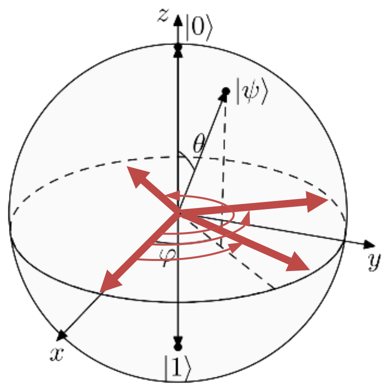
k logical data qubits,
total n physical qubits



Bit flip (X)



Phase flip (Z)



Phase shift

- No data is read, instead "syndrome" measurements are performed that reveal the type of error without destroying superposition of the *data*.

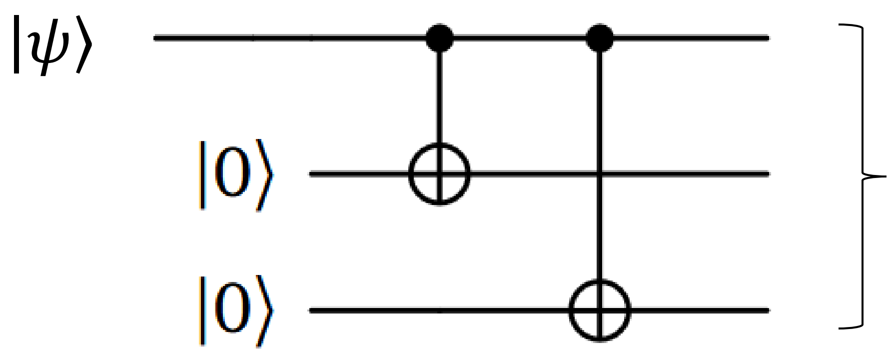
Example: 3-bit Repetition Code

Encoding

- The 3-bit repetition code encodes an input state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to

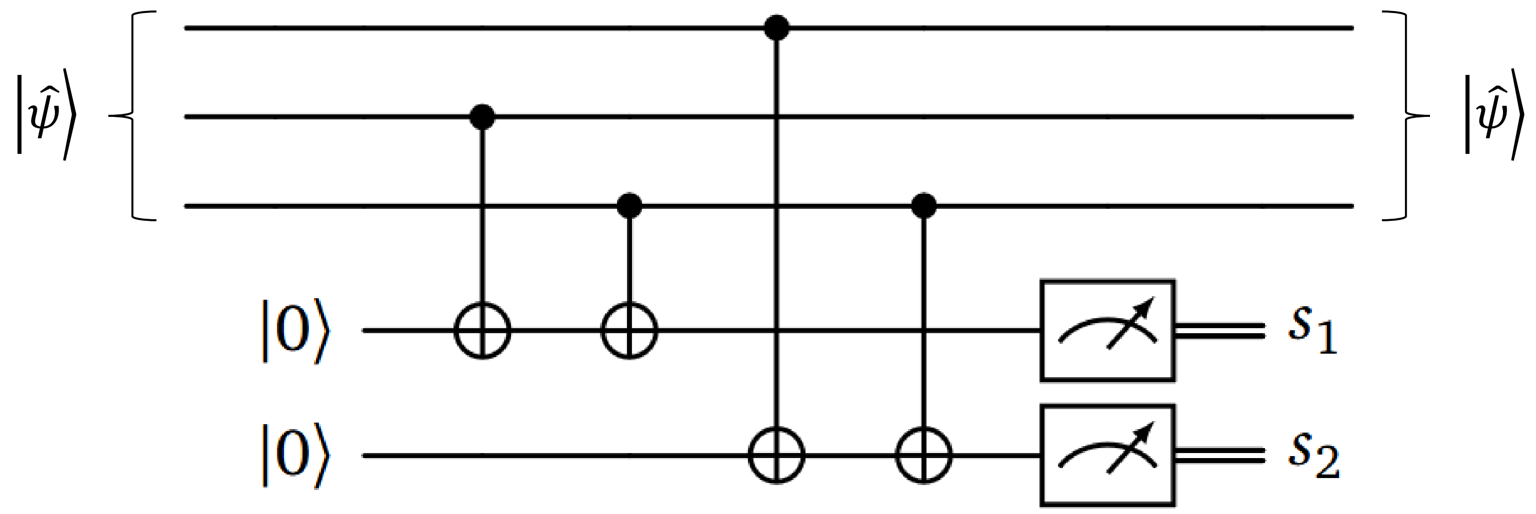
$$\left|\hat{\psi}\right\rangle = \alpha|0\rangle \otimes |0\rangle \otimes |0\rangle + \beta|1\rangle \otimes |1\rangle \otimes |1\rangle$$
$$= \alpha|000\rangle + \beta|111\rangle$$

Encoding circuit:



Decoding – Syndrome measurement

- Measure ancilla qubits to obtain the syndromes:

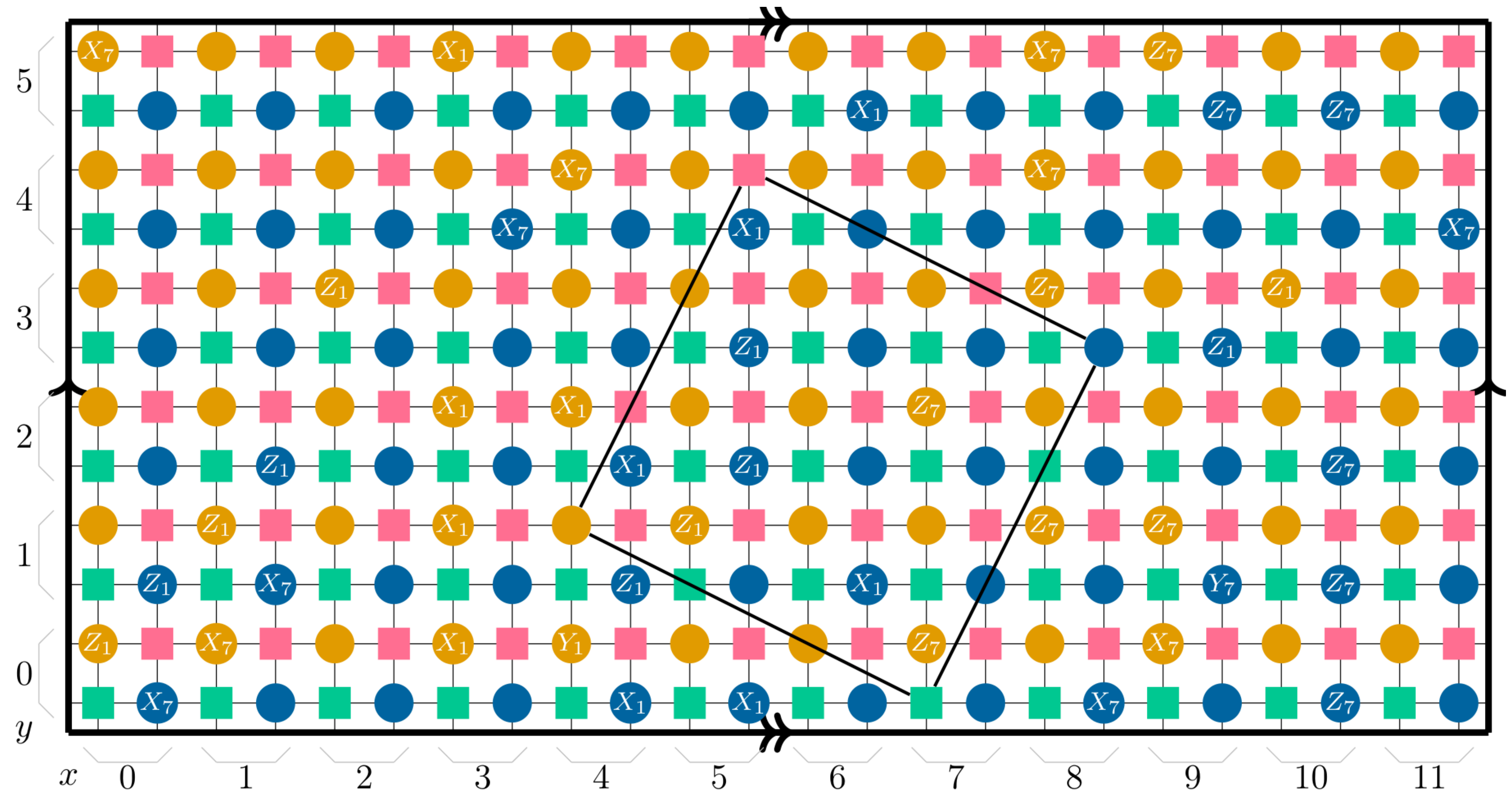


Each ancilla qubit accumulates one of the syndrome parities. Then, the value of the syndrome is read out by measurement.

- To check if $\left|\hat{\psi}\right\rangle$ is a valid encoding of a qubit, evaluate these two equations (stabilizer generators):

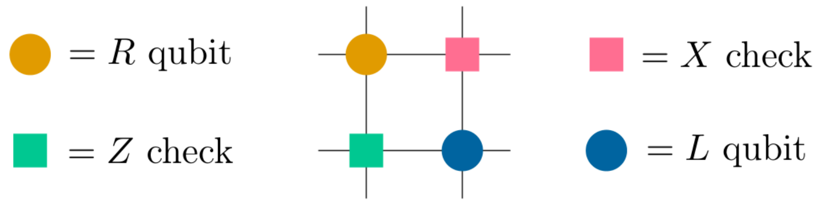
$$(\mathbf{1} \otimes \mathbb{Z} \otimes \mathbb{Z})\left|\hat{\psi}\right\rangle = \left|\hat{\psi}\right\rangle$$
$$(\mathbb{Z} \otimes \mathbf{1} \otimes \mathbb{Z})\left|\hat{\psi}\right\rangle = \left|\hat{\psi}\right\rangle$$

Quantum LDPC Codes | IBM Gross codes



(b) The [[144, 12, 12]] gross code

Rate: 12/144



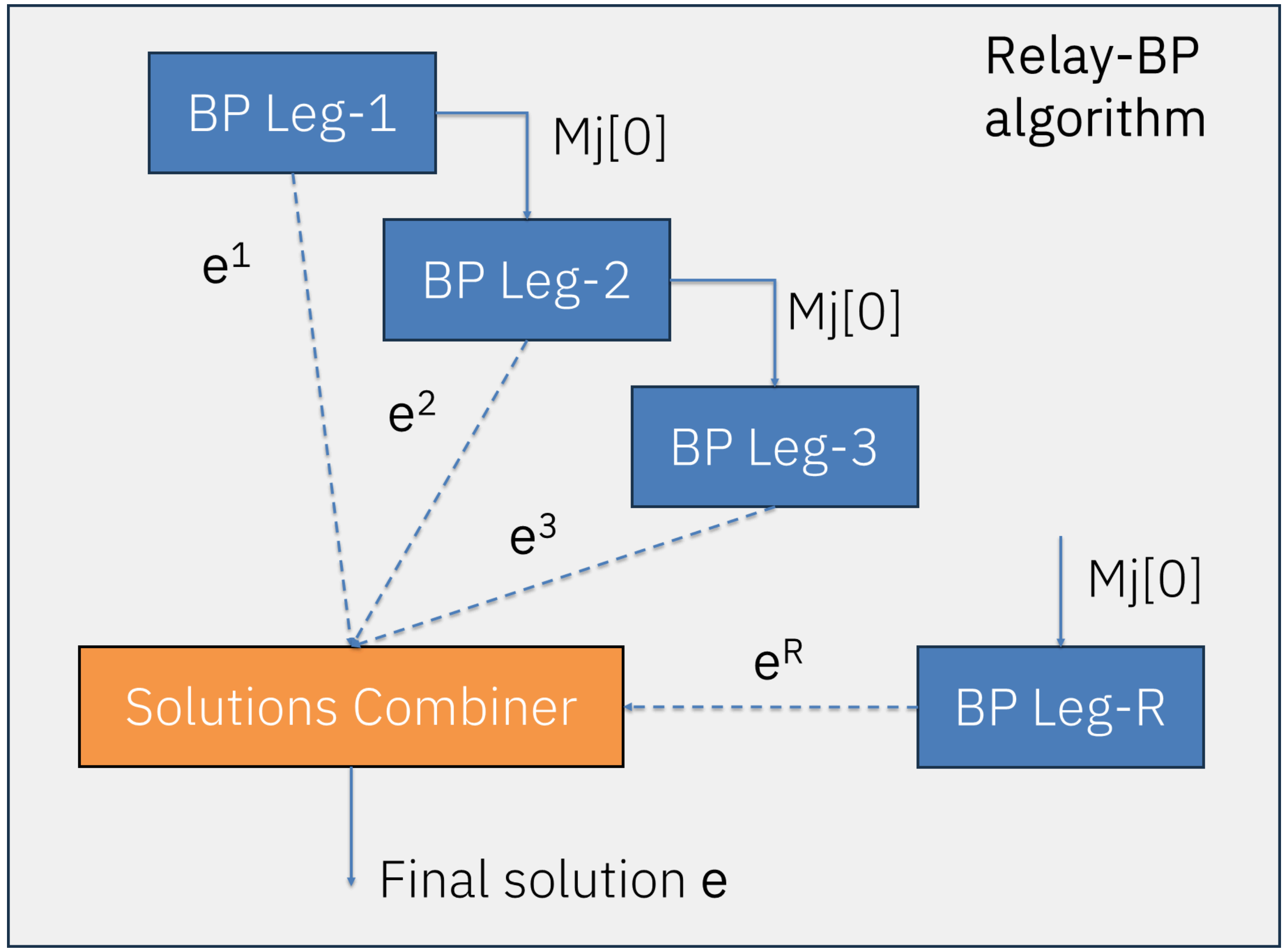
(a) A unit cell

- Encoding done by computing parity among qubit groups.
- Typical codes employ groups of nearest-neighbor qubits.
- The Gross codes introduce long-range connections, in addition to nearest-neighbor connections.
- This results in much improved code efficiency (rate).
- IBM Gross codes are 10x more efficient than surface codes of the same correction strength.

Nature 627, 778–782 (2024)

QEC Decoding | IBM Relay-BP decoder

- LDPC and qLDPC codes are decoded by iterative algorithms that exchange messages in graphs.
- These are variants of Belief Propagation (BP).
- IBM's Relay-BP is such a BP variant.
- Relay-BP is an **ensemble** decoder.

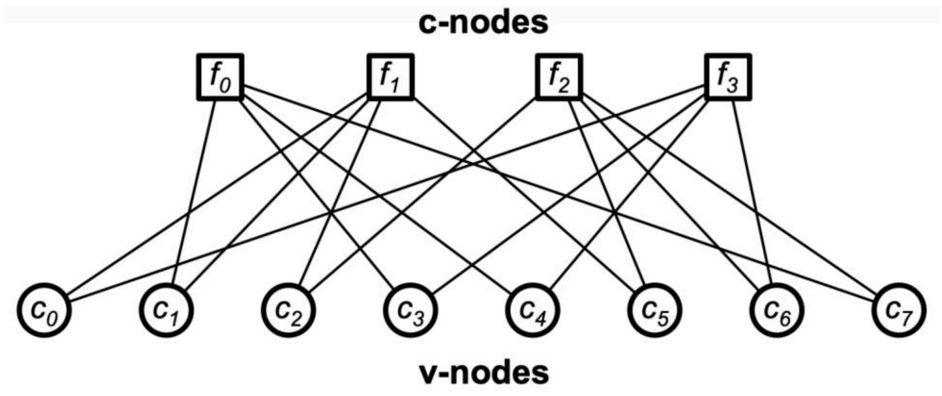


Voting among S<<R solutions

Parity-Check Matrix

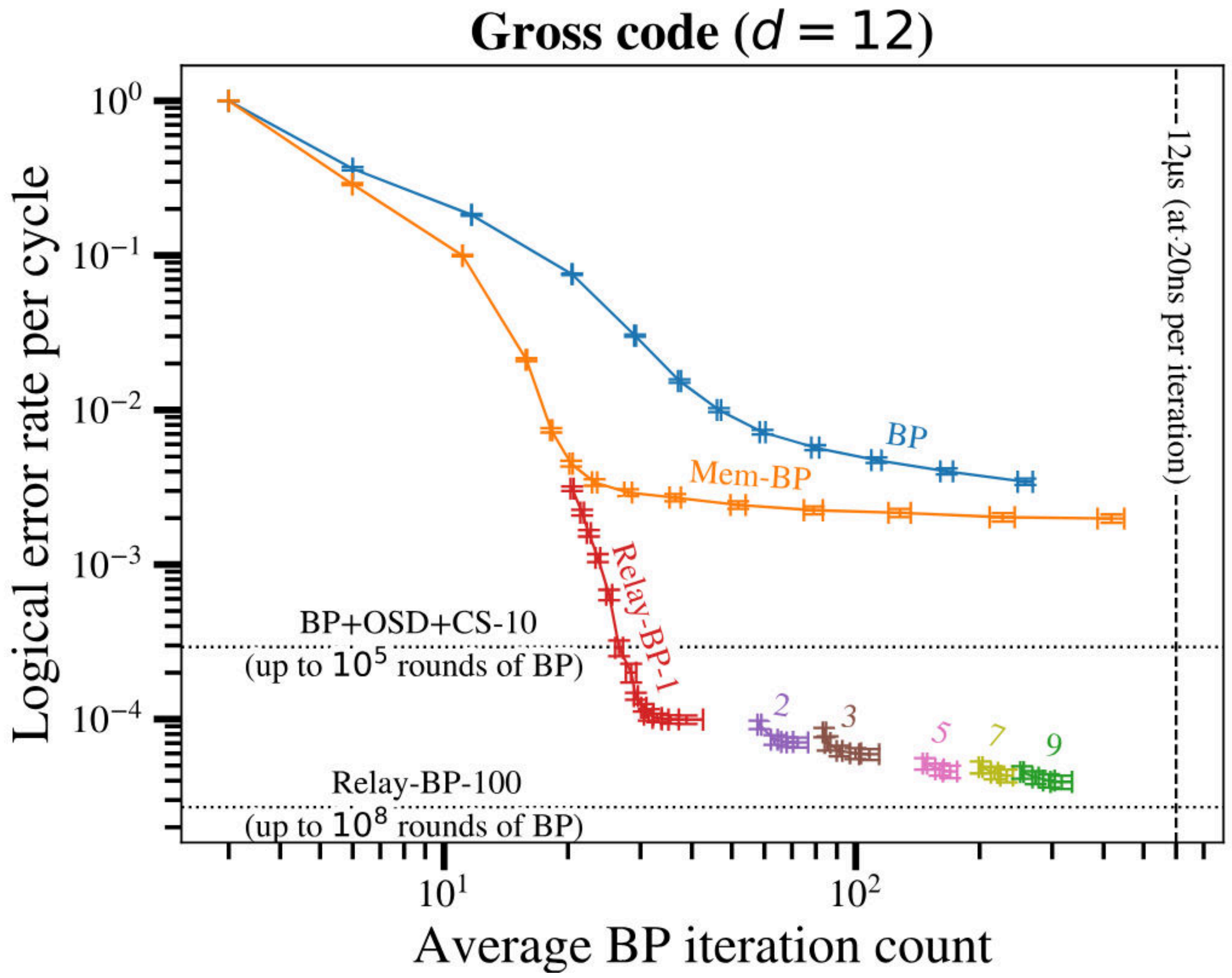
$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Tanner Graph



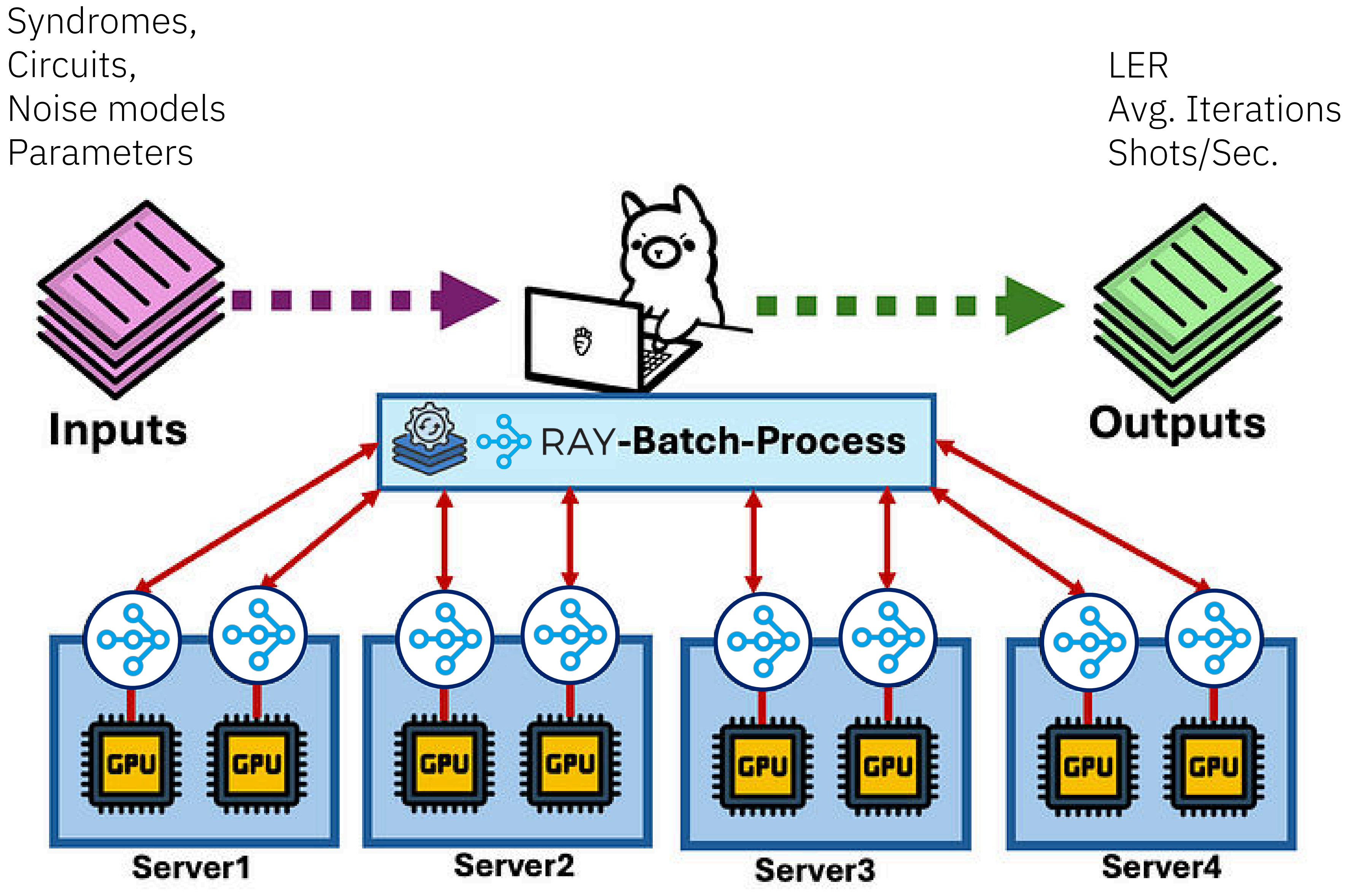
Decoding Objective:
Find e s.t. He=s

Relay BP Performance



Motivation for simulating QEC

- The goal of QEC is to reduce the logical error rate below the physical error rate. Simulations estimate quantities such as:
  - Logical failure probability
  - Memory lifetime
  - Error suppression versus code distance
  - Performance across different physical error rates
- Since logical error rates can be extremely low (e.g., $\leq 10^{-12}$), they are difficult to measure directly on current QPUs, making simulations essential.
- The decoders often determines how well a code performs in practice. Different decoders with different configurations make different trade-offs between accuracy and computational cost.
- Simulations enable the evaluation of realistic noise models



More information on high-performance GPU kernels using Triton on thousands of GPUs at scale:
D. Diamantopoulos et al., "Pêlαγos: An Adaptive GPU Decoder for Error Correction Simulations of Quantum LDPC Codes", IEEE Int. Conf. on Quantum Software (QSW), 2026

QEC Simulation Workflow

Phase 1 :
Circuit generation

Decompose a logical quantum circuits into native hardware-level instructions using a code specific instruction set architecture (ISA).

For each code block, encodes logical qubits into many physical qubits.

Phase 2 :
Code artifact creation

Add noise information and compile physical circuit into a code artifacts (detectors observables, decoding matrices, circuit data, detector error model (DEM)).

The compiled artifact files are put into the distributed object store so all workers can access it.

Phase 3 :
Sample creation

Generate syndrome samples from the DEM by applying the noise model and perform syndrome measurements (stabilizers)

Executed in parallel batches and stored in a Zarr store.

Phase 4 :
Parallel decoding

Decode batches of syndromes using specific decoder backend(s)

- Relay-BP (CPU or GPU variants using Triton, or CudaQ
- Tesseract, Chromobius, pymatching, ...

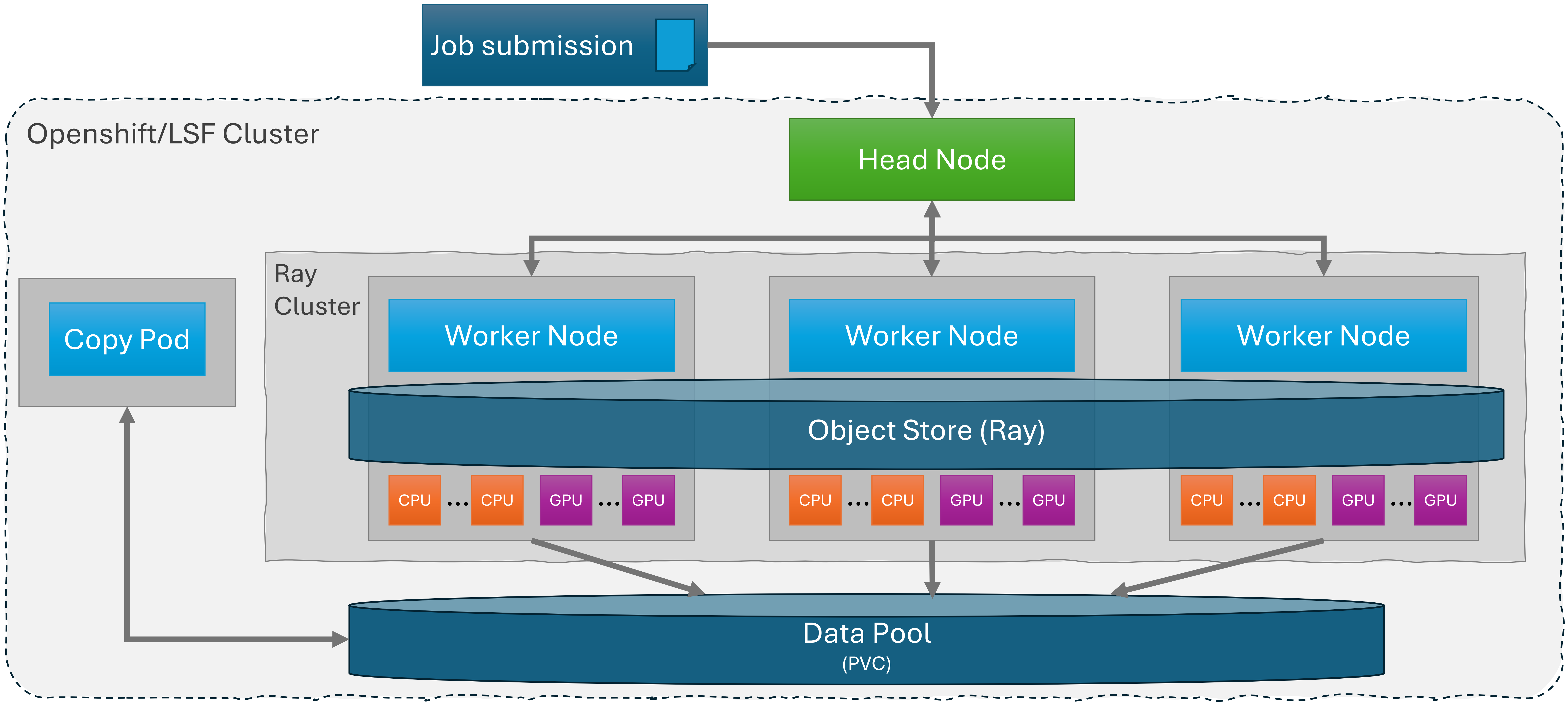
Decoders may use a different (simplified) noise model.

Collect per-batch results with predicted corrections

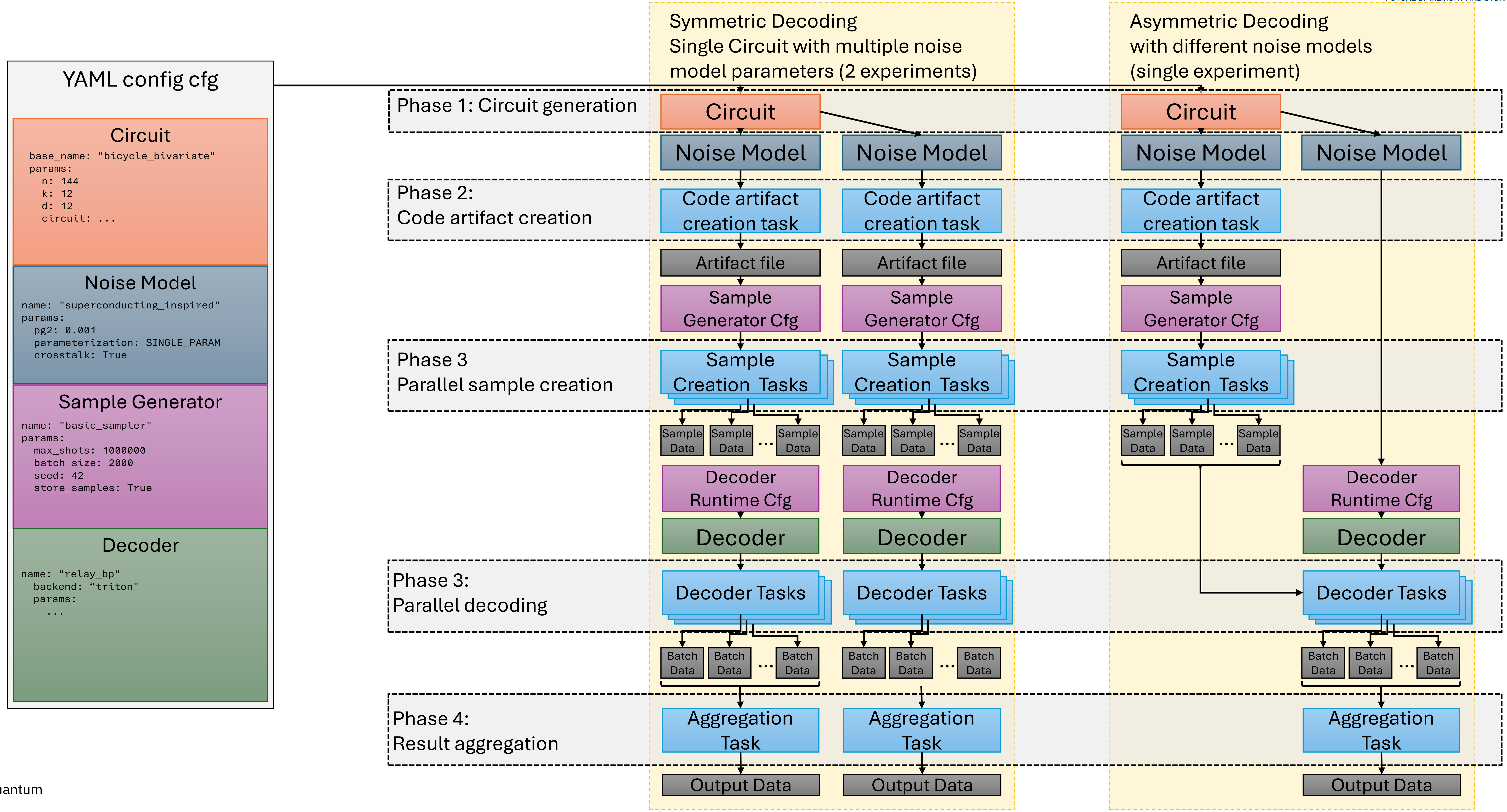
Phase 5 :
Result aggregation

Merge per-batch result files into a single aggregated JSONL with computed statistics.

Cluster overview



QEC pipeline processing



Conclusion

- High-fidelity simulation of QEC algorithms allows researchers to predict performance, optimize designs, and identify practical trade-offs before deploying them on a real Quantum Processing Unit at reasonable cost.

- QEC simulations help to guide hardware development of decoders (GPUs, FPGAs, ASICs) as classical decoding with low latency and high throughput is essential for large-scale FTQC.

- Future work: Investigate more advanced techniques including, windowed decoding, concatenated codes and morphing codes


IBM Quantum