Communication Strategies for Wafer-Scale Iterative Solvers

Hayden Estes[+] Nathan Hanford* Kirk W. Cameron[+] Edgar A. León*
[+]Virginia Tech *Lawrence Livermore National Laboratory

Cerebras WSE-3 Architecture

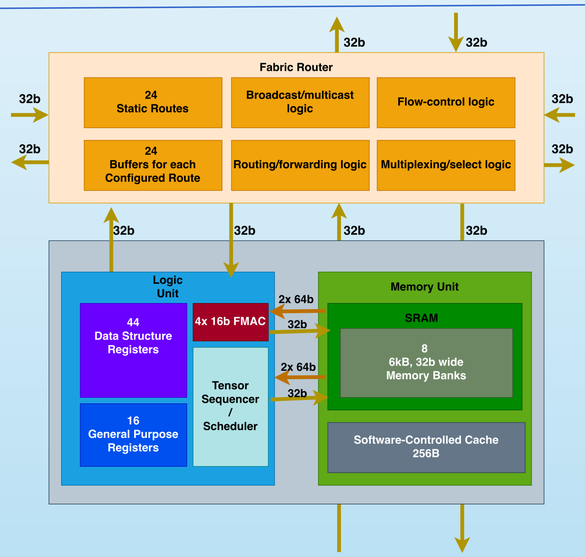
The Cerebras Wafer Scale Engine 3 (WSE-3) is a *post-von Neumann dataflow architecture* that utilizes the entire silicon wafer.

Despite being built for **AI workloads**, recent work shows promising results computing algorithms with *locality*.

The WSE-3 chip contains **900,000 Processing Elements** (PE) running at 850 MHz and **44 GB of on-chip SRAM** with a 23 kW peak energy draw.

The largest hurdles in utilizing these systems for modeling and simulation is the **ease of programmability** due to relying on low-level general programming abstractions when not using AI mode, and the **small memory size** of each PE.

Architecture of One WSE-3 PE



WSE-3 Data Flow

Utilizing the WSE-3 chip requires **spatial awareness** for all operations, as each PE has *limited memory, registers, and threads while only communicating with their neighbors.*

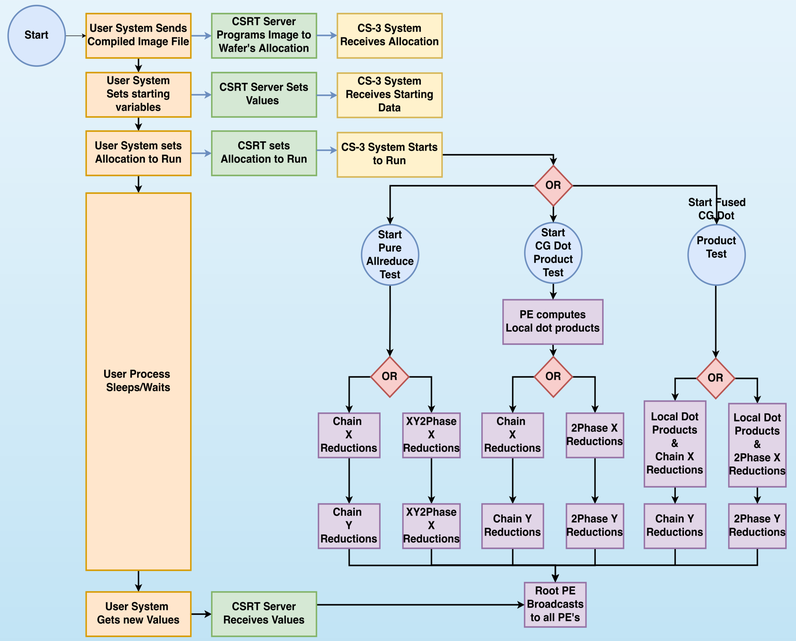
PE's communicate through **Wavelets**, special 32-bit fabric packets for sending instructions or data.

Structure of the WSE-3 Fabric Packets (Message Passing)



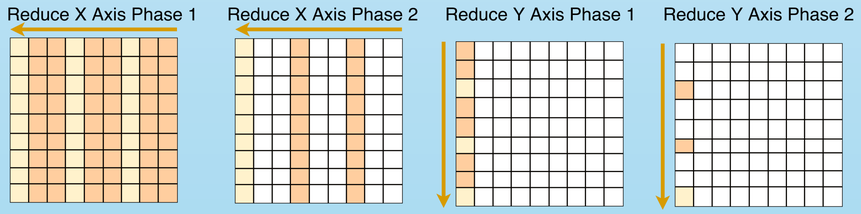
From All Reduce to CG Dot Product

We built three benchmarks of increasingly realistic workloads (pure All Reduce, CG dot product, fused CG dot product), each in Tungsten and Paint with either communication algorithm, and ran them on Kingfisher at Sandia National Laboratories on a 512×512 grid (262,144 PEs). Each PE computes B lanes over K local elements: payload is 4·B bytes per PE, and K adds local work but no communication. XY two-phase uses groups of ~23 PEs.

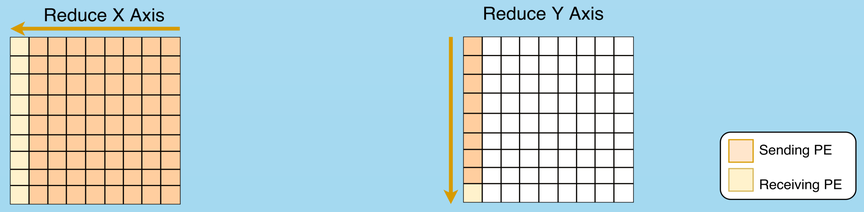


All Reduce Algorithms

XY two-phase All Reduce 10x10 Example

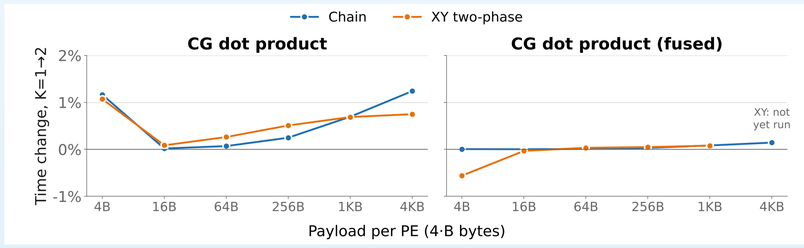


Chain All Reduce 10x10 Example

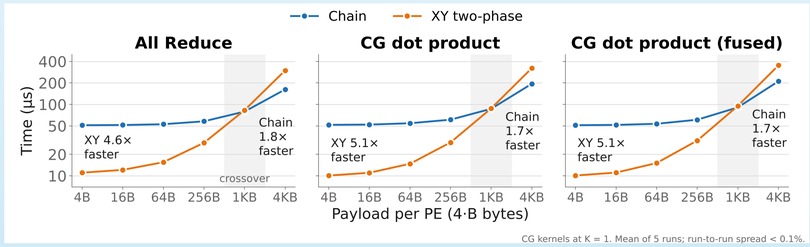


Impact of Communication on Workloads

Doubling local work per PE (K) changes timing by ≤1.3%



XY two-phase wins below ~1 KB of payload per PE; Chain wins above



CG kernels at K = 1. Mean of 5 runs; run-to-run spread < 0.1%.

```
// 1. Compute every local dot-product lane.
i in [0, B) {
    value[i] ← 0;

    k in [0, K) {
        value[i] ← value[i]
                 + matrix[i * K + k]
                 * xvec[i * K + k];
    }
}

// 2. Reduce each row from right to left.
if (W > 1) {
    i in [0, B) {
        if (x ≠ W - 1) {
            incoming ← x1[];
            value[i] ← value[i] + incoming;
        }

        if (x ≠ 0) {
            x1[] ← value[i];
        }
    }
}

// 3. Reduce column zero from bottom to root.
if (x == 0 && H > 1) {
    i in [0, B) {
        if (y ≠ H - 1) {
            incoming ← y1[];
            value[i] ← value[i] + incoming;
        }

        if (y ≠ 0) {
            y1[] ← value[i];
        }
    }
}
```

Conclusions

- No single All Reduce algorithm performs best for every payload.
- XY two-phase wins for small, latency-bound payloads: its shorter critical path has fewer serial hops.
- Chain is best for large payloads, where bandwidth matters more than latency.

Future Work

- Sweep larger K and finish the K = 2 runs.
- End-to-end CG on Matrix Market inputs and mini-apps.
- Pick the All Reduce by payload; tune XY group size.
- Compare with MPI collectives on CPU/GPU.

Acknowledgments

cerebras
Sandia National Laboratories

Code Available

LLNL-POST-2024376

Prepared by LLNL under Contract DE-AC52-07NA27344.