Exploring Dataflow Architectures for Improved Efficiency in Earth System Models

Justs Zarins

j.zarins@epcc.ed.ac.uk

Acknowledgements

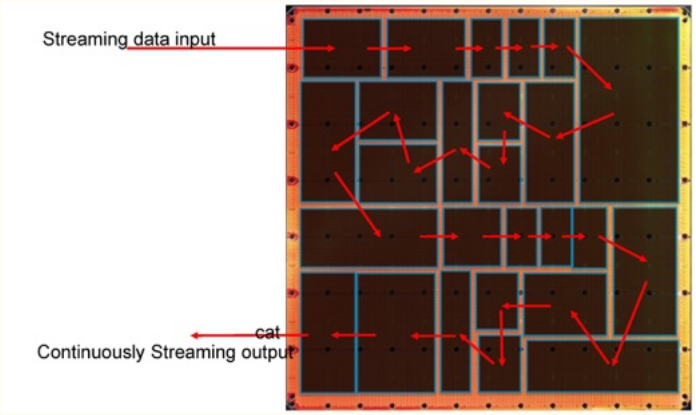
- CONTINENTS is an international collaboration project between EPCC, NCAR, and NCAS. We are aiming to improve the computational and energy efficiency of large scale computing environments and improve atmospheric simulations through directed research.
- UKRI project EP/Z531170/1

Outline

- Dataflow model
- Cerebras Wafer Scale Engine
  - hardware
  - applications
  - programming
- Modelling SWE on WSE

Dataflow model

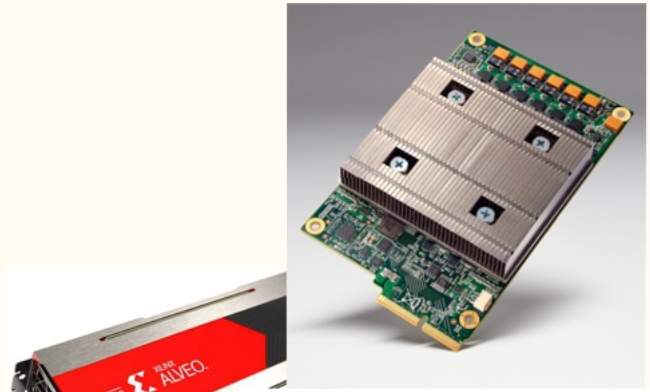
- "Dataflow machines are programmable computers of which the hardware is optimized for fine-grain data-driven parallel computation."[1]
- Characterised by spatial parallelism
  - tens of thousands of processors
  - fast on-chip network between processors
- Generally lacks large centralised memory
- Usually power efficient

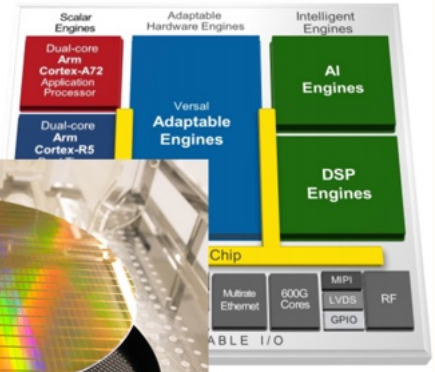


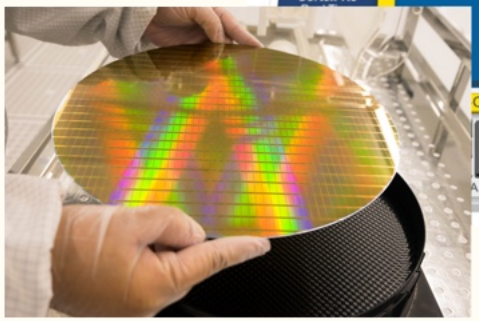
1. https://dl.acm.org/doi/10.1145/27633.28055

Hardware examples

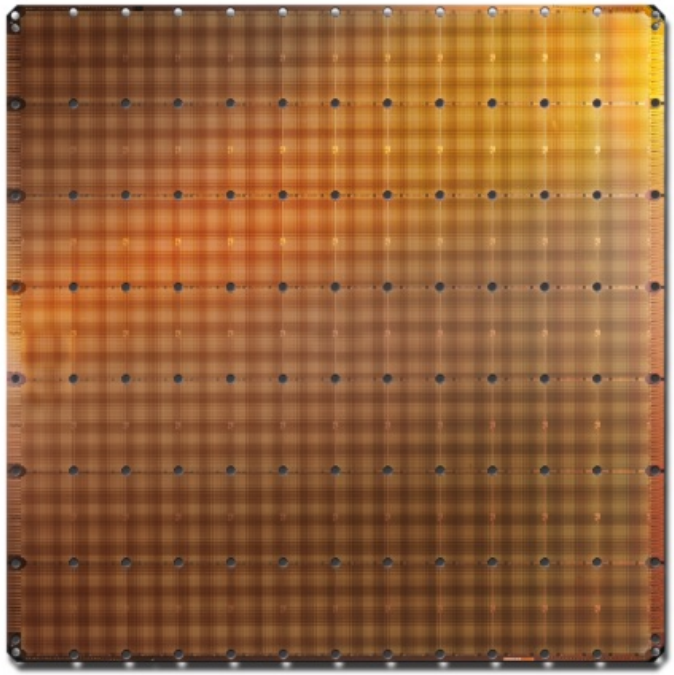
- Google TPU
- Configurable Corse Grain Array - Xilinx's ACAP
- AMD AI engine
- Cerebras Wafer Scale Engine







Cerebras Wafer Scale Engine



Cerebras WSE

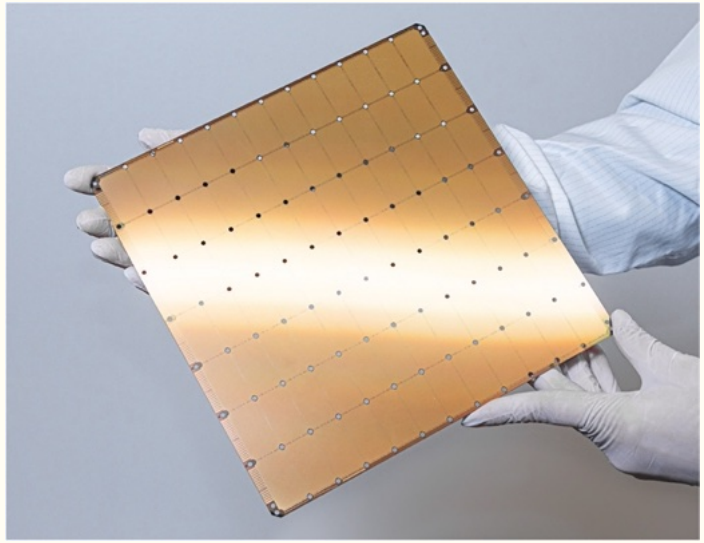
1.2 Trillion transistors
46,225 mm² silicon

Largest GPU

21.1 Billion transistors
815 mm² silicon

Wafer Scale Engine (WSE)

- Made by Cerebras
- Built upon a 5nm process technology
- The current generation WSE-3 specs:
  - 900,000 independent cores
  - 44GB on-chip SRAM memory
  - 21PB/s aggregate memory bandwidth
  - 214 Pb/s processor to processor fabric bandwidth
- The flexibility of the independent cores and the large amount of memory means that, the WSE-3 is capable of delivering the performance of many GPUs

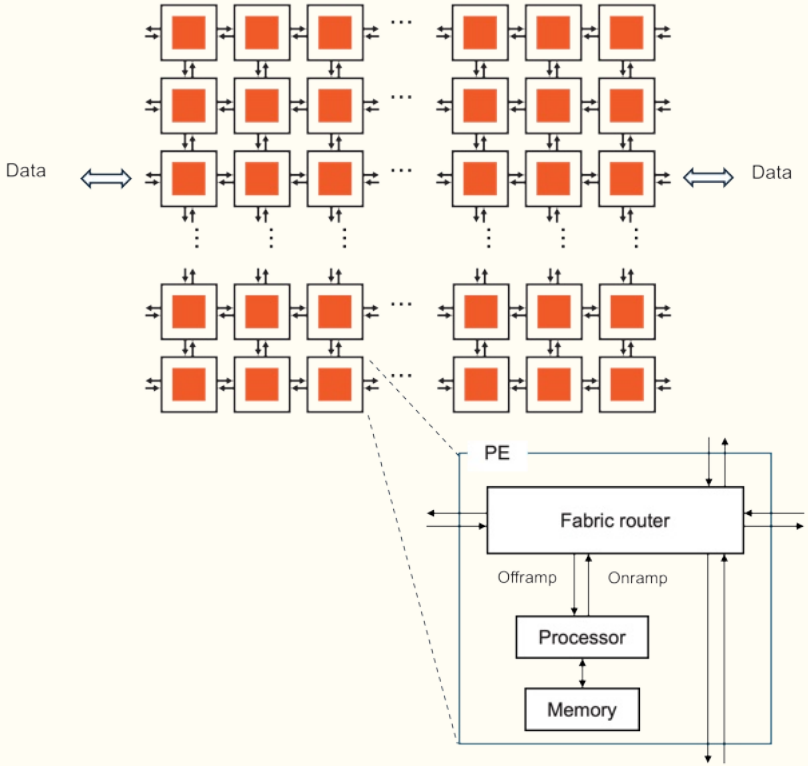


CSX

- Current generation is CS-3
- Custom cooling and power delivery for the WSE
  - draws around 23kW of power
- Uses standards-based power and network connections
  - 12x standard 100 Gigabit Ethernet links
- Multiple can be clustered together

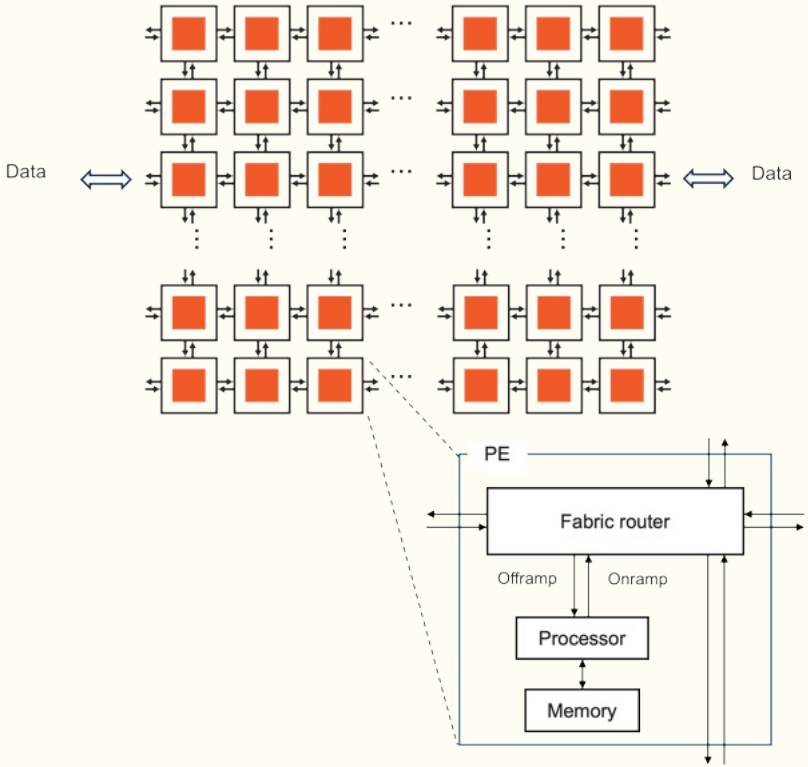
WSE details

- The Cores / Processing Elements (PEs) run independent of each other
  - e.g. have their own program counters
- PEs are connected by a 2D rectangular mesh across the chip
  - 32-bit messages (called wavelets) can be communicated with neighbours in a single cycle
- The 44GB of WSE memory is distributed amongst the PEs
  - Each PE has its own private chunk of memory
  - Access time on the order of cycles



WSE PE details

- The processor itself
  - Commonly referred to as the Compute Engine (CE)
  - Independent and private from any other
- A router
  - Connected with bidirectional links to own CE and router of four neighbours
  - Link to own CE is called the RAMP and neighbours are referred to by north, south, east and west
  - This is the only way in which PEs can communicate
- Local (private) memory
  - All data and code for the PE is stored in this memory
  - 48KB per PE



What are suitable applications?

Success stories

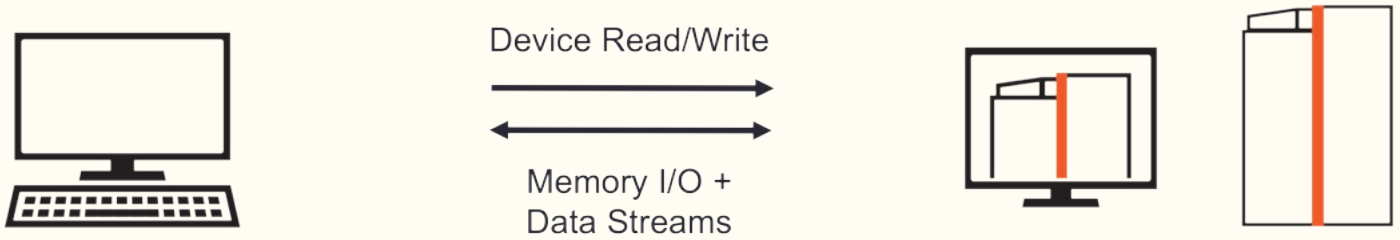
- "Wafer-Scale Fast Fourier Transforms"
  - fastest time for a usefully sized benchmark
- "Massively scalable stencil algorithm"
  - turns a memory bound problem into a compute bound problem
- "Massively Distributed Finite-Volume Flux Computation"
  - two orders of magnitude speedup over GPU
- "Scaling the "Memory Wall" for Multi-Dimensional Seismic Processing with Algebraic Compression on Cerebras CS-2 Systems"
  - Gordon Bell finalist on 48 CS-2 systems, 93 PB/s sustained memory bandwidth
- "Breaking the Molecular Dynamics Timescale Barrier Using a Wafer-Scale System"
  - 457-fold improvement in timesteps per second versus the Frontier GPU-based Exascale platform

Programming the WSE

- For machine learning - PyTorch
  - The original and primary workload
- For everything else - Cerebras Software Language (CSL)
  - Based on the Zig language
  - *High quality simulator is available*

- **Host CPU(s): Python**
- Loads program onto CSX system
- Streams in/out data from/to device
- Launches device functions

- **Device: CSL**
- CSL programs run on groups of cores on the WSE, specified by programmer
- Executes dataflow programs



Programming the WSE – execution units

- CSL program consists of tasks and functions
- Functions can be called by the host or another function on the device
- Tasks are started by the hardware, run until completion, and then at that point the hardware chooses another task to run
  - Can only be activated, can not be called by other tasks or functions
  - No return value

Programming the WSE - communication

- A wavelet is a 32-bit message communicated with a neighbour in a single cycle
- Each physical channel has 24 virtual communication channels known as colors that can be used for passing wavelets
- Each wavelet has associated with it a 5-bit identifier which defines which channel it is communicated on
    - Determines the wavelet's routing through the fabric and its consumption
    - This is a bit like a tag in MPI point-to-point communications, and similarly many messages on one color does not block messages with a different color using the same physical link
- Wavelets are consumed by tasks on a PE where a task is registered to execute when a wavelet arrives with a specific color
- Collectives and point-to-point communication libraries are available

CSL examples

```
//modules
const v1 = @import_module("m1.csl");
v1.incr();

//basic syntax
fn factorial(x : i32) i32 {
  if (x <= 2) return x;
  return x * factorial(x-1);
}

//builtins
var matrix = @zeros([4,5]f16);

//tasks
task recvTask(data: u16) void {
  globalValue = data;
}

//compile time computation
//colours and routing
comptime {
  @bind_task(recvTask, recvColor);
  @set_local_color_config(recvColor,
    .{ .rx = .{ WEST }, .tx = .{ RAMP } });
```

CSL performance features

Data Structure Descriptors (DSDs)

- Provide a mechanism to consider an array, and an access pattern, as a complete unit
- Operations using DSDs run for multiple cycles to complete an instruction on all data referenced by the DSD
- Performance and ease of use: lifts level of program to talking about whole structures, while lowering cost of computing indexing into hardware

```
const dstDsd =  @get_dsd(mem1d_dsd, .{.tensor_access = |i|{5} -> dst[i]});
const src0Dsd = @get_dsd(mem1d_dsd, .{.tensor_access = |i|{5} -> src0[i*3]});
const src1Dsd = @get_dsd(mem1d_dsd, .{.tensor_access = |i|{5} -> src1[i,i]});
```

- DSDs are a *unifying concept* that provides for complex memory reads and writes and fabric reads and writes

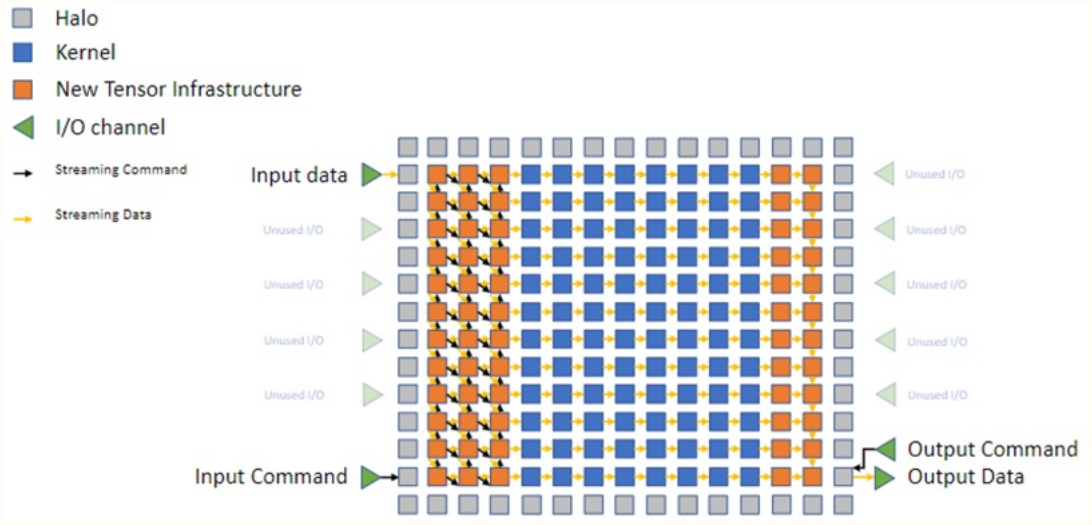
```
const fabDsd = @get_dsd(fabout_dsd, .{.fabric_color = output_color, .extent = 5});

task main_task() void {
  @faddh(dstDsd, src0Dsd, src1Dsd);
  @fmovh(fabDsd, dstDsd);
}
```

CSL libraries

Various available: collective communications, message passing, math, debug...

Memcpy library:

- Transfer data between the WSE and host
- The host and WSE network interfaces finally route the data into your kernel
  - (last step is implemented on the WSE itself to connect the I/O channel entry-points, which are in fixed locations at the edges of the WSE)
- Uses an additional halo of PEs around the user kernel and multiple columns on either side to route data



Modelling SWE for WSE

Shallow Water Equations

- The shallow water model is a simplified kernel representing the nonlinear PDE's governing geophysical fluid flow
- A mini-app with computations similar to the ones used in weather and climate modeling
- Useful in testing the suitability of hardware or programming languages

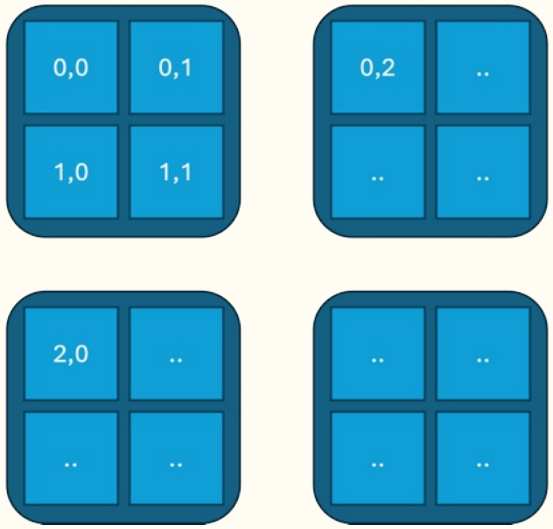
$$\left.\begin{aligned} \frac{\partial \mathbf{V}}{\partial t} + \eta \mathbf{N} \times (P\mathbf{V}) + \operatorname{grad}(P + \tfrac{1}{2}\mathbf{V}\cdot\mathbf{V}) = 0 \\ \frac{\partial P}{\partial t} + \operatorname{div}(P\mathbf{V}) = 0 \end{aligned}\right\}, \qquad (1)$$

Computational patterns

- Stencils (involves self, left, right, diagonal)
  - Mesh point updates can happen independently
- Halos (due to use of stencils and domain decomposition)
- Boundary conditions (can be cyclic)
- Operates on 2D arrays
- Code sections:
  - Data initialisation
  - Time loop
    - Stencil updates
    - Periodic continuation
    - Stencil updates
    - Periodic continuation
    - Time smoothing
  - Result output and cleanup

Data decomposition

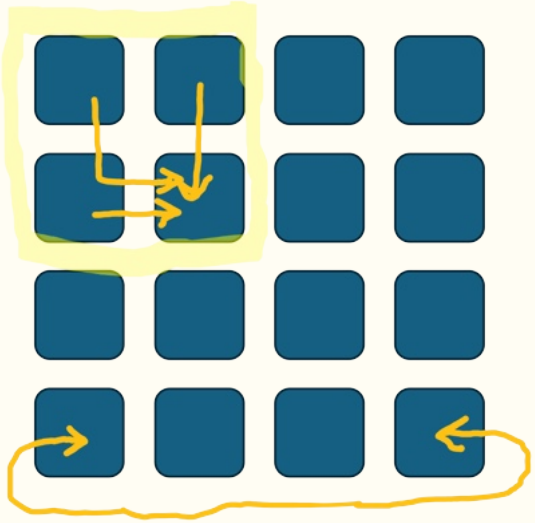
- 13 2D data arrays for u, v, p, etc.
- Match 2D mesh structure by decomposing arrays evenly



- Memory limitations:
  - 52 bytes (using 32bit floats on WSE) of storage per mesh point
  - 945 mesh points per PE (48 KB memory) MAX
  - WSE-2 has 757x996 PEs, so max problem size (round down to 30x30 elements on each PE) is 22'710 x 29'880
  - Some space needed for initialisation data and program...

Communications

- PEs need to communicate halos
  - Diagonal halo takes the longest, but should still take only ~2 cycles
  - Tiny, uniform latency should lead to perfect weak scaling
- Long range communications due to periodic boundary conditions take up to ~996 cycles
  - Local computation can hide this latency
  - The critical path between long range communications does ~40 ops per mesh point
  - Would need 25 mesh points to cover worst case latency
- Data wavelets can activate computation tasks



Performance estimates

- Should be able to run at full 800MHz when pipeline is full, parallelised to the minimal requirement to hide long range comms.
- 256x256 problem
  - Put 2x2 mesh points on each PE (hides up to 160 cycle long range comm)
  - Use 128x128 PEs
  - Compute rate is: `128*128*800MHz = 12.5` TFlop/s (at 32bit floats)
- 4096x4096 problem
  - Put 8x8 mesh points on each PE (hides up to 2560 cycle long range comm)
  - Use 512x512 PEs
  - Compute rate is: `512*512*800MHz = 200` TFlop/s (at 32bit floats)

Trends

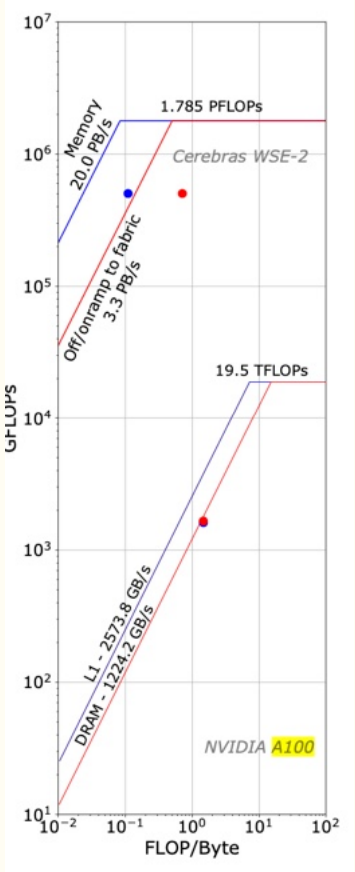
- Larger problems need more data per PE to hide the long range comms.
  - This reduces compute parallelism for a fixed problem size.
  - Limited by memory on PEs.
- Want to use minimum number of mesh points per PE to maximise parallelism.
  - But this increases the distance of long range comms.
  - Limited by number of PEs.
- Parallelisation is key. The CSX has relatively slow cores, but many more of them.
- Parallelising across multiple devices is the next challenge.
  - Similar considerations to the long-range comms problem.

PLMR model

- "WaferLLM: A Wafer-Scale LLM Inference System" (https://arxiv.org/pdf/2502.04563)
  - "PLMR model captures the unique hardware properties of wafer-scale accelerators"
- Massive **P**arallelism (P)
- Highly non-uniform memory access **L**atency (L)
- Constrained local **M**emory (M)
- Constrained **R**outing resources (R)

Power efficiency

- CSX uses up to 23kW (the whole system)
  - About 9 GFLOP/W for the 4096x4096 problem.
- Comparing to "Massively scalable stencil algorithm" (https://arxiv.org/pdf/2204.03775)
  - Reports 22 GFLOP/W for 3D stencil on WSE
  - A100 GPU achieves about 5 GFLOP/W in the same study
  - Up to 228x speedup when using WSE
- Green500 HPL benchmarks, Nov 2024
  - 1st place achieves 72 GFLOP/W
  - 10th place achieves 62 GFLOP/W



Future work

- Implementation on CS-3
- Generalisation of modelling and implementation strategy
- Extension to Tenstorrent Blackhole
- Evaluation of more complex models, such as CM1 and MPAS

Summary, SWE on WSE

- The CSX is a powerful architecture with significant raw compute and distributed memory
- We have explored the general way in which the WSE is organised and the key concepts and terminology
- Modelled the performance of shallow water equations on a dataflow architecture
  - An efficient implementation is possible
  - Likely to see performance and energy benefits