A Changing Landscape

Jason Cong[2], Konstantis Daloukas[1], Nate Earnest-Noble[3], Kostas Kafousas[1], Sophia Kolak[4], Yorgos Koutsoyannopoulos[1], Bob Lucas[1], Hamed Mohamedbagherpoor[3], Francois-Henry Rouet[1],and Linghao Song[2]

[1]Ansys, [2]UCLA, [3]IBM, [4]CMU

June 22, 2023

Past era of innovation

- The mid-1980s saw the emergence of CMOS as a viable semiconductor technology
  - Low cost, low power, and high-volume components led to “commercial off the shelf” (COTS) strategy
- Unbounded demand for computing in science and national security
  - Public funding for computer science research
- A decade of innovation in computer architecture and parallel programming
  - Communicating sequential processes, shared memory, data parallel, systolic, dataflow, etc.
- Thomas Sterling ruined it for all of us
  - He invented the Linux cluster
    - Cheap hardware
    - Free software

Déjà vu: we have entered a new era of innovation

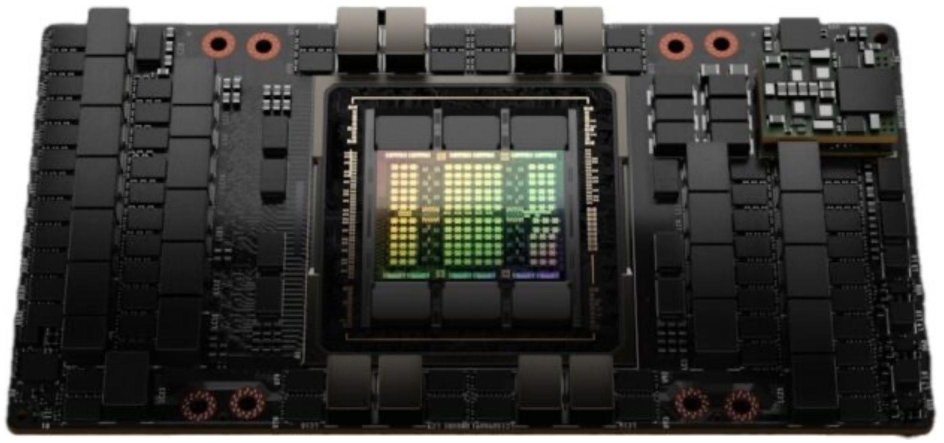
- Dennard scaling has ended
- Moore’s Law is slowing
- Yet, demand is growing
- Specialized systems are increasingly attractive
  - D.E. Shaw’s Anton (drug design)
  - “General purpose” GPUs
  - TPUs and DPUs
  - Neuromorphic (IBM True North)
  - Annealers for optimization (Fujitsu, Toshiba and D-Wave)
  - ML start ups including:
    - Cerebras (wafer scale integration)
    - Samba Nova (reconfigurable data flow)
  - Quantum computers
- Which of these can we exploit?

Anton

D-Wave

Graphical Processing Units (GPUs)

- Familiar to this community
  - MUMPS uses them
- More computing power than their host
  - Slower clocks, but an order-of-magnitude more ALUs
  - Off-load DGEMM and other compute-bound functions
    - O(N^2) data transfer versus O(N^3) floating point operations
    - IBM's ESSL does this transparently
  - Write more sophisticated functions in CUDA (or HIP, or SYCL)
- More memory bandwidth than their host
  - HBM provides an order-of-magnitude more main memory bandwidth
    - An order of magnitude less memory volume
  - Off-load memory bandwidth bound functions
    - E.g., iterative solvers dominated by sparse matrix multiply

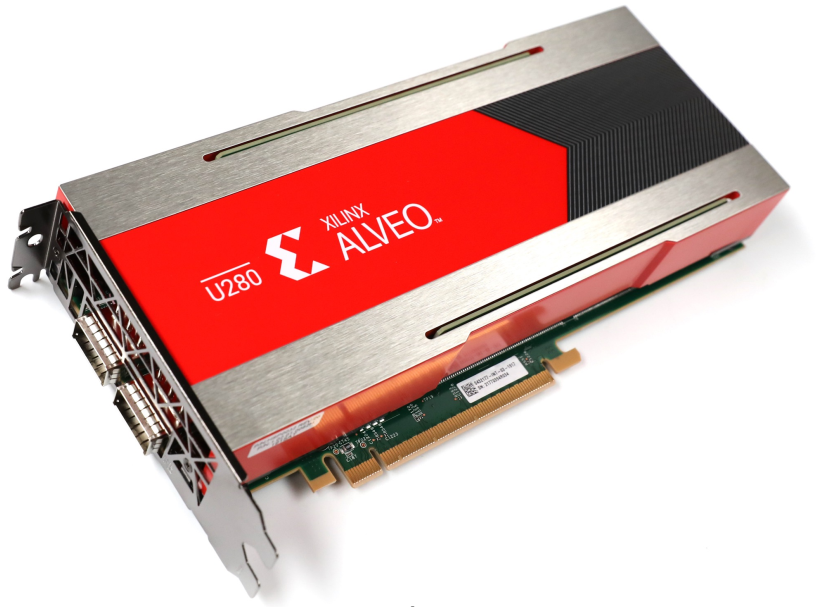

Nvidia Hopper

Accelerated Processing Units (APUs)

- AMD devices mixing CPU and GPU
  - On the same for game consoles
  - In the same socket for HPC
- Only one physical memory
  - Upcoming MI300A CPUs and GPUs will share 128 GB of HBM
    - Will be launched at SC23
  - No DDR for the CPUs
  - Should eliminate the CPU <-> GPU data copying bottleneck
- Open questions
  - Will this enable exploiting finer-grained computations on GPUs?
  - Will we be able to use OpenMP 5 directives effectively?
  - Will kernel launch times or other overheads still limit broader utility of GPUs?

Field Programmable Gate Array (FPGA)

- "Sea of gates"
  - First brought to market in 1985 by Xilinx
  - User programs logical units (LUTS) and their interconnect
    - Create your own custom circuit
    - Slower and more power-hungry than an ASIC
  - Widely used in networking systems
    - Reconfigure hardware in the field with software patches
- Programmable Array of Memory
  - World's first attempt to make an FPGA-based accelerator
  - DEC Paris Research Center
- SPLASH
  - First attempt in the US
  - My wife was one of four people who could program it

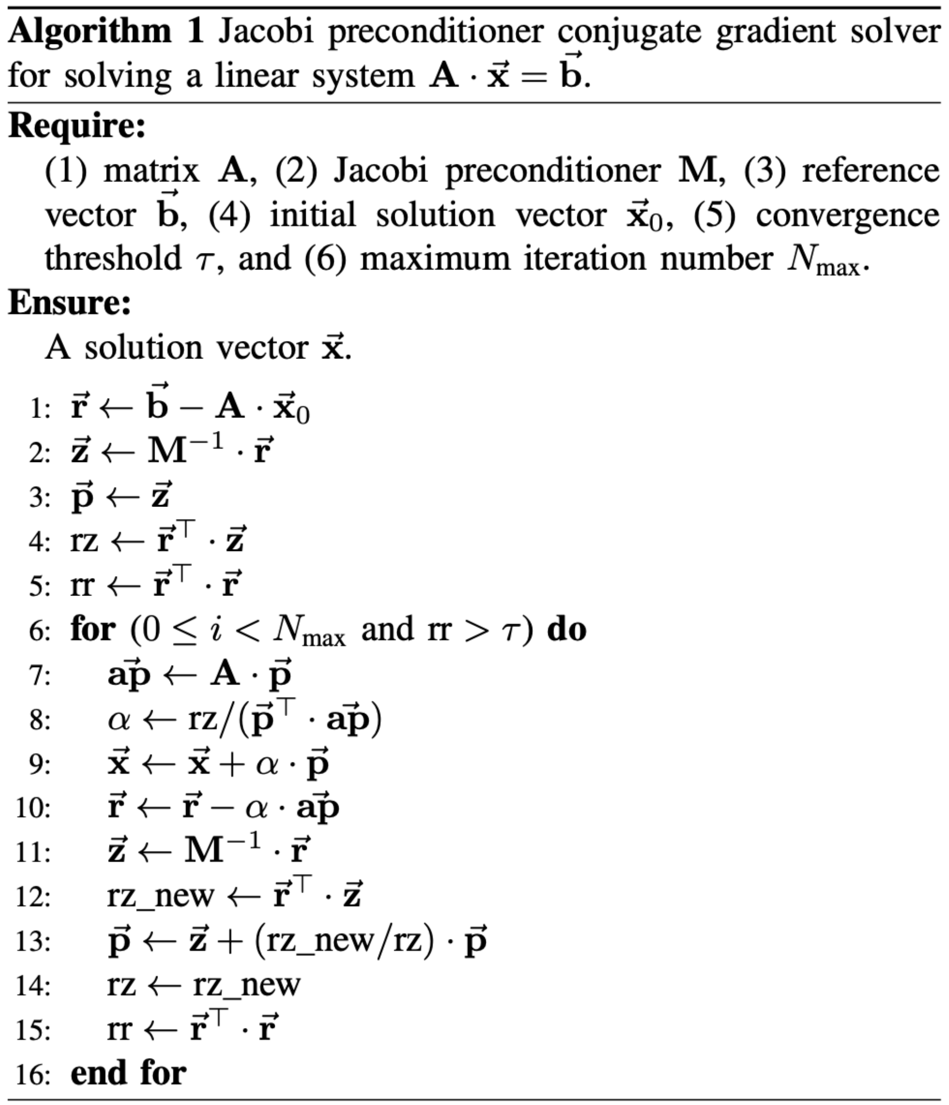


Xilinx U280

How can an FPGA help?

- GPUs were initially used for compute intensive functions
  - Factor large frontal matrices
- Now GPU memory bandwidth is often more important
  - Sparse matrix – vector multiplication
  - Algebraic multigrid
- FPGAs are cheaper and have lower power consumption than GPUs
- FPGAs now have HBM too
  - Xilinx U280 has 8 GB and 460 GB/s
  - Ansys joined UCLA Prof. Jason Cong's Center for Domain-Specific Computing (CDSC)
  - Exploring preconditioned Conjugate Gradients

The Jacobi Preconditioned Conjugate Gradient (JPCG) Solver



Algorithm 1 Jacobi preconditioner conjugate gradient solver for solving a linear system $\mathbf{A} \cdot \vec{\mathbf{x}} = \vec{\mathbf{b}}$.

Require:
(1) matrix $\mathbf{A}$, (2) Jacobi preconditioner $\mathbf{M}$, (3) reference vector $\vec{\mathbf{b}}$, (4) initial solution vector $\vec{\mathbf{x}}_0$, (5) convergence threshold τ, and (6) maximum iteration number $N_{\max}$.

Ensure:
A solution vector $\vec{\mathbf{x}}$.

```
1:  $\vec{\mathbf{r}} \leftarrow \vec{\mathbf{b}} - \mathbf{A} \cdot \vec{\mathbf{x}}_0$
2:  $\vec{\mathbf{z}} \leftarrow \mathbf{M}^{-1} \cdot \vec{\mathbf{r}}$
3:  $\vec{\mathbf{p}} \leftarrow \vec{\mathbf{z}}$
4:  $\mathrm{rz} \leftarrow \vec{\mathbf{r}}^{\top} \cdot \vec{\mathbf{z}}$
5:  $\mathrm{rr} \leftarrow \vec{\mathbf{r}}^{\top} \cdot \vec{\mathbf{r}}$
6:  for $(0 \leq i < N_{\max}$ and $\mathrm{rr} > \tau)$ do
7:      $\mathrm{a}\vec{\mathbf{p}} \leftarrow \mathbf{A} \cdot \vec{\mathbf{p}}$
8:      $\alpha \leftarrow \mathrm{rz}/(\vec{\mathbf{p}}^{\top} \cdot \mathrm{a}\vec{\mathbf{p}})$
9:      $\vec{\mathbf{x}} \leftarrow \vec{\mathbf{x}} + \alpha \cdot \vec{\mathbf{p}}$
10:     $\vec{\mathbf{r}} \leftarrow \vec{\mathbf{r}} - \alpha \cdot \mathrm{a}\vec{\mathbf{p}}$
11:     $\vec{\mathbf{z}} \leftarrow \mathbf{M}^{-1} \cdot \vec{\mathbf{r}}$
12:     $\mathrm{rz\_new} \leftarrow \vec{\mathbf{r}}^{\top} \cdot \vec{\mathbf{z}}$
13:     $\vec{\mathbf{p}} \leftarrow \vec{\mathbf{z}} + (\mathrm{rz\_new}/\mathrm{rz}) \cdot \vec{\mathbf{p}}$
14:     $\mathrm{rz} \leftarrow \mathrm{rz\_new}$
15:     $\mathrm{rr} \leftarrow \vec{\mathbf{r}}^{\top} \cdot \vec{\mathbf{r}}$
16: end for
```

- Solve A*x=b where A and b are known and x is unknown
- Iteratively refine errors and approach a solution
- Widely used in scientific and engineering computing, including LS-DYNA
  - Default for thermal modeling

Callipepla

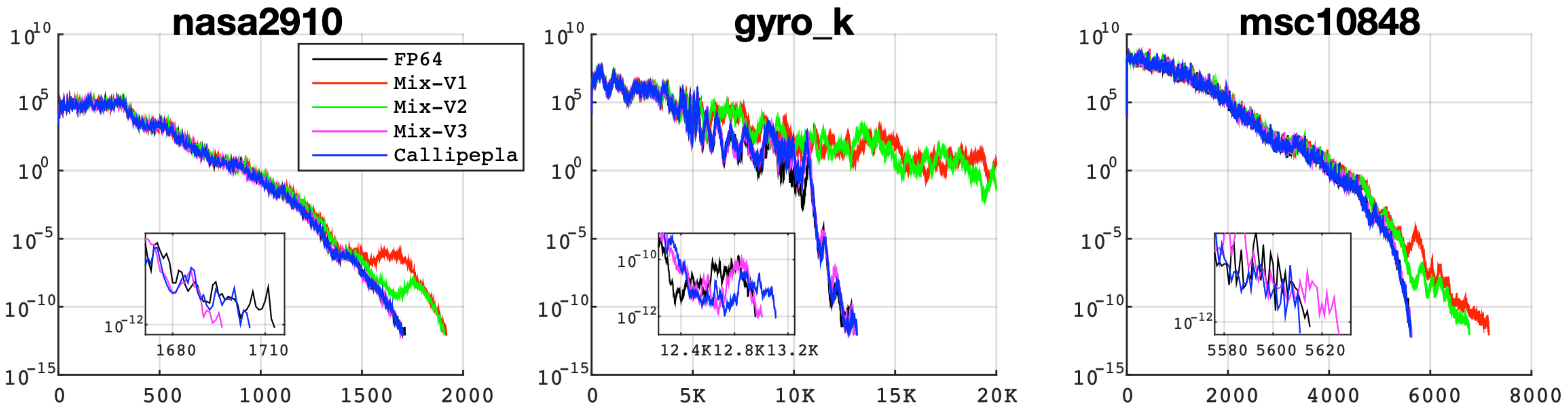
- High-Level Synthesis can take days to run
  - Therefore, a PCG accelerator cannot be uniquely designed for each sparse matrix
- Callipepla (California state bird) is streaming architecture for PCG
  - Operates on an arbitrary sparse matrix structure
- Research questions included:
  - How to support an arbitrary problem and terminate accelerated processing on the fly?
    - Main loop termination condition unknown until run time
  - How to coordinate long-vector data flow among processing modules?
    - Off-chip accesses: may waste memory bandwidth
    - On-chip accesses: which vectors and which modules can be kept ina very limited memory
  - How to reduce off-chip memory bandwidth yet maintain the double precision (FP64) accuracy?
    - SpMV dominates the memory bandwidth consumed

Mixed-precision sparse matrix-vector multiplication

- Bandwidth
  - FP32 is better
- Accuracy
  - FP64 is better

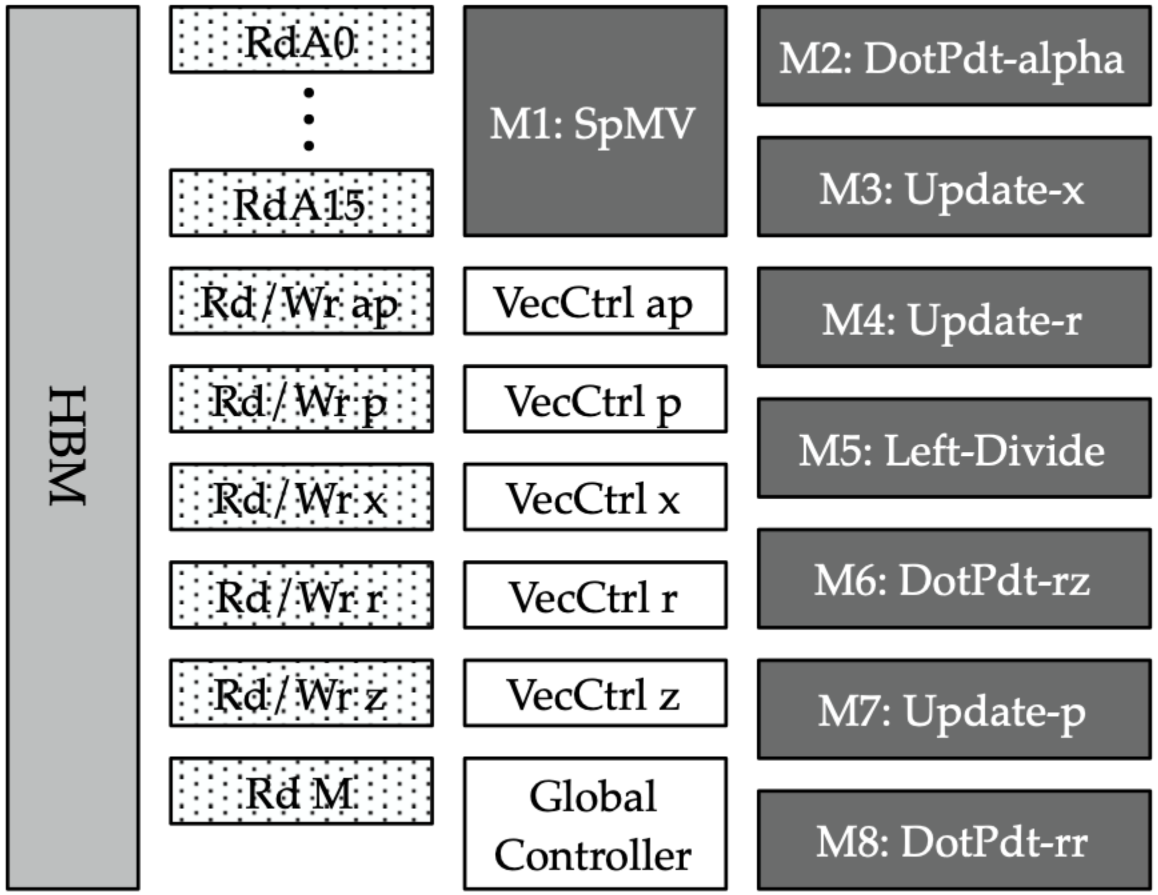
THREE MIXED-PRECISION SCHEMES FOR SPMV $\vec{\mathbf{y}} = \mathbf{A} \cdot \vec{\mathbf{x}}$.

	$\mathbf{A}$	$\vec{\mathbf{x}}$	$\vec{\mathbf{y}}$
Default FP64	FP64	FP64	FP64
Mixed-V1	FP32	FP32	FP32
Mixed-V2	FP32	FP32	FP64
Mixed-V3	FP32	FP64	FP64



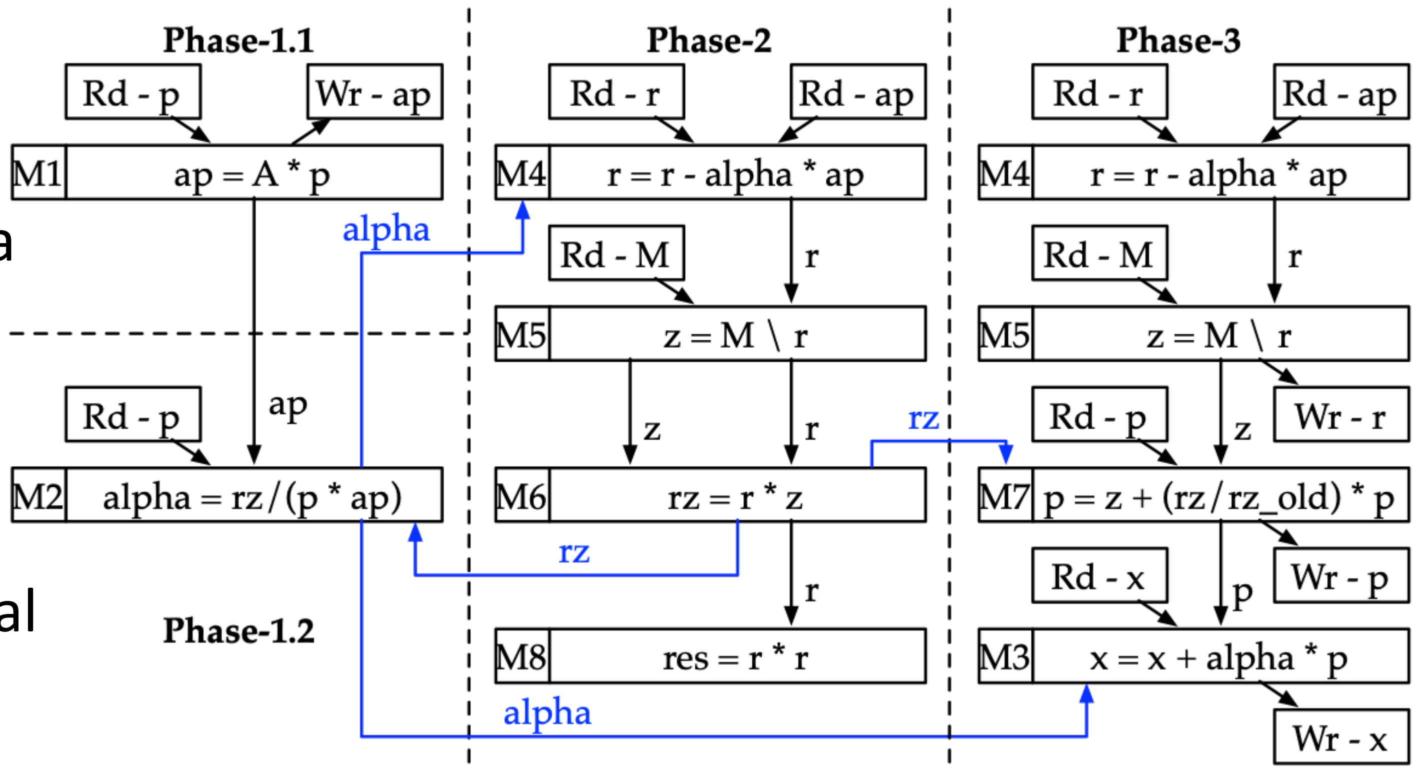
Callipepalla architectural components

- U280 HBM FPGA
- Memory modules -> read/write
- Vector control modules -> control vector flows
- Computation modules -> computations, M1-M8
- SpMV: based on Serpens (DAC'22)



Dependency & Three Computation Phases

- The computation modules -> three phases
  - A scalar dependency separates two phases
  - Each vector only rd/wr once within a phase
  - All modules share the same vectors within a phase
- Vector streaming reuse
  - To trigger the processing of individual modules
  - To overlap computation to save processing time
  - To share vectors among modules to save off-chip bandwidth



Early results in LS-DYNA

- AWE nested cylinders benchmark
- AMD Host with Xilinx Alveo

Solver	1st step	2nd step
Multifrontal	849	2.1
Ref. JPCG	408	N/A
FPGA JPCG	43	50

- Intel Skylake (two AVX512 ALUs), first load step

Solver	1 CPU	8 MPI
Multifrontal	562	113
JPCG	265	39
ICCG	164	34

FPGA timing:

Preprocessing	28.43
Downloading	0.77
Solving	13.96

Future directions for FPGA R&D

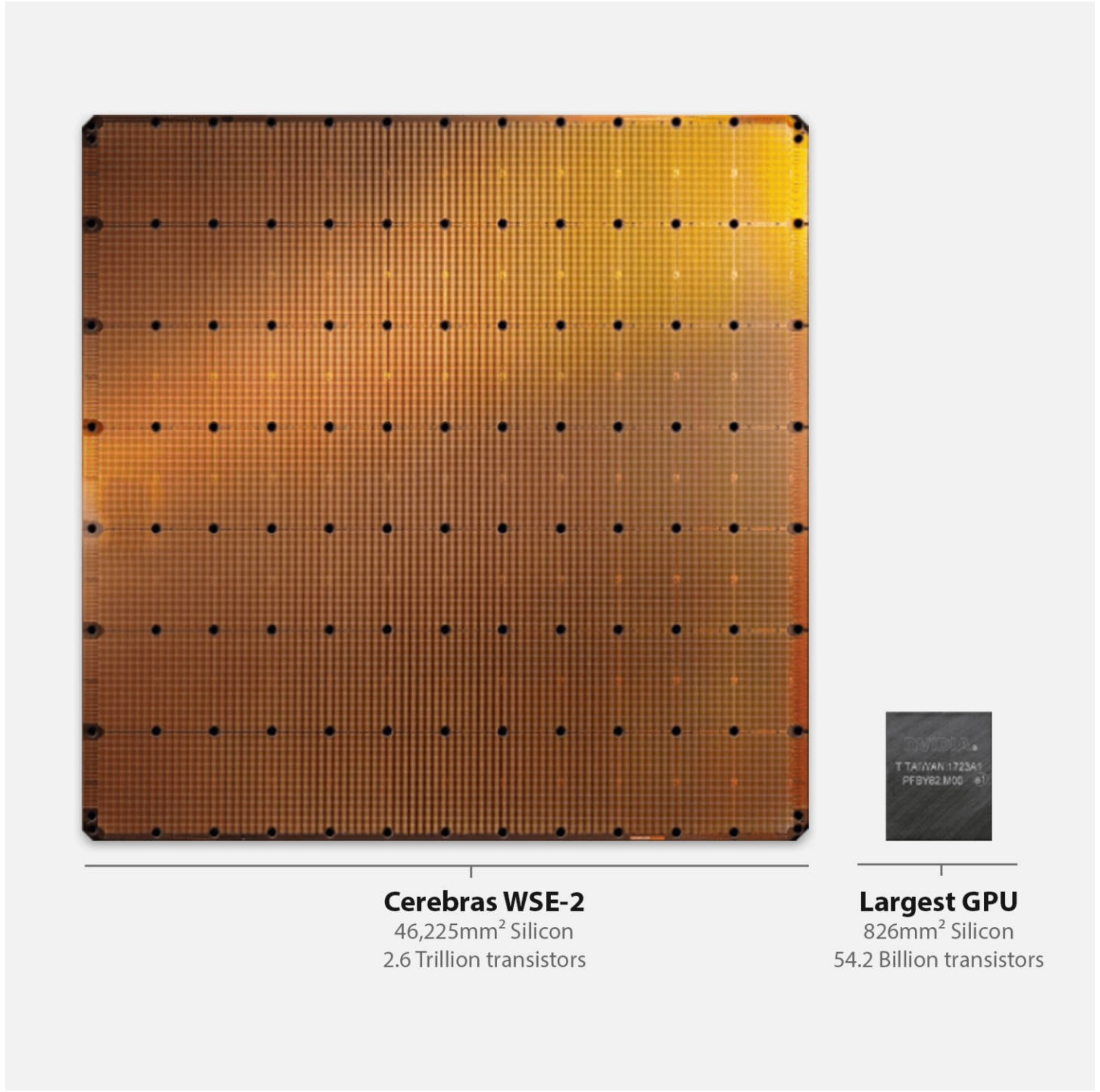
- Multiple FPGAs per host node
  - How many PCI-e channels do you have?
  - Multiple distributed memory hosts
    - Like Nvidia GPUS, Xilinx FPGAs can talk to each other without involving their hosts
- ICCG
  - Requires triangular solves
  - Work in progress with UCLA

Cerebras machine learning startup

- One of over a hundred hardware startups targeting training for machine learning
- Cerebras is unique in that they've gone wafer-scale!
  - Obvious good idea for decades
    - Not easy though. Gene Amdahl tried and failed
  - TSMC is their fab
- Data movement on-chip takes more energy than arithmetic.
  - 100 pJ for a floating-point operation
  - 120 pJ to move an operand 2 cm
- Data movement off-chip takes an order-of-magnitude more energy
  - 2000 pJ to move a bit to DRAM
  - The Cerebras WSE doesn't have multiple chips

Cerebras CS-2 Wafer-Scale Engine

- Good
  - Largest "chip" in the World at 46,225 mm^2
  - 2.6 trillion transistors
  - 850,000 32-bit cores, with ~2 PFlop/s peak
  - 20 PB/s of memory bandwidth
  - 220 Pb/s on-chip network
- Less Good
  - 40 GB of SRAM
    - Not much state at each vertex in a neural net
  - No high-level programming language
    - TensorFlow and PyTorch
  - 15 RU, 23 KW
    - More energy efficient than GPU-based systems



Collaboration with Cerebras and National Energy Technology Lab

- Focus on iterative solvers
  - Not enough memory for MUMPS
- Initially looking at structured, 3D grids
  - First two dimensions map easily to Cerebras processor grid
    - 3rd dimension is local memory
  - Performance of approximately 1 PFlop/s when all processors are busy
- Unstructured grids is work-in-progress

Ansys and quantum computing

- Ansys is involved in quantum computing today.
  - Our Computer Aided Engineering (CAE) tools are being used to design quantum computers.
  - E.g., HFSS is used to design resonators for transmons

- Ansys needs to know if we can use quantum computers to accelerate our CAE tools.
  - We have formed a partnership with IBM to investigate quantum computing.
  - Initial focus on graph partitioning.

- Longer term, Ansys expects quantum computers will enable new CAE markets that are inconceivable today.
  - Simulation of quantum dynamics.

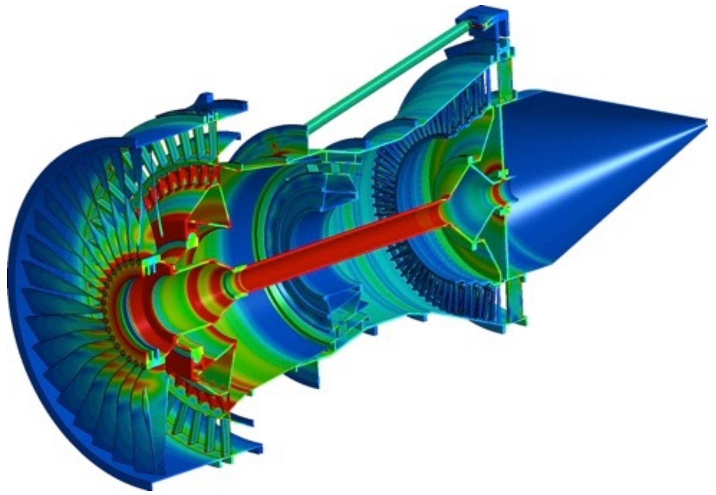
The relative importance of sparse matrix reordering

We reorder sparse matrices to reduce the cost of factorization

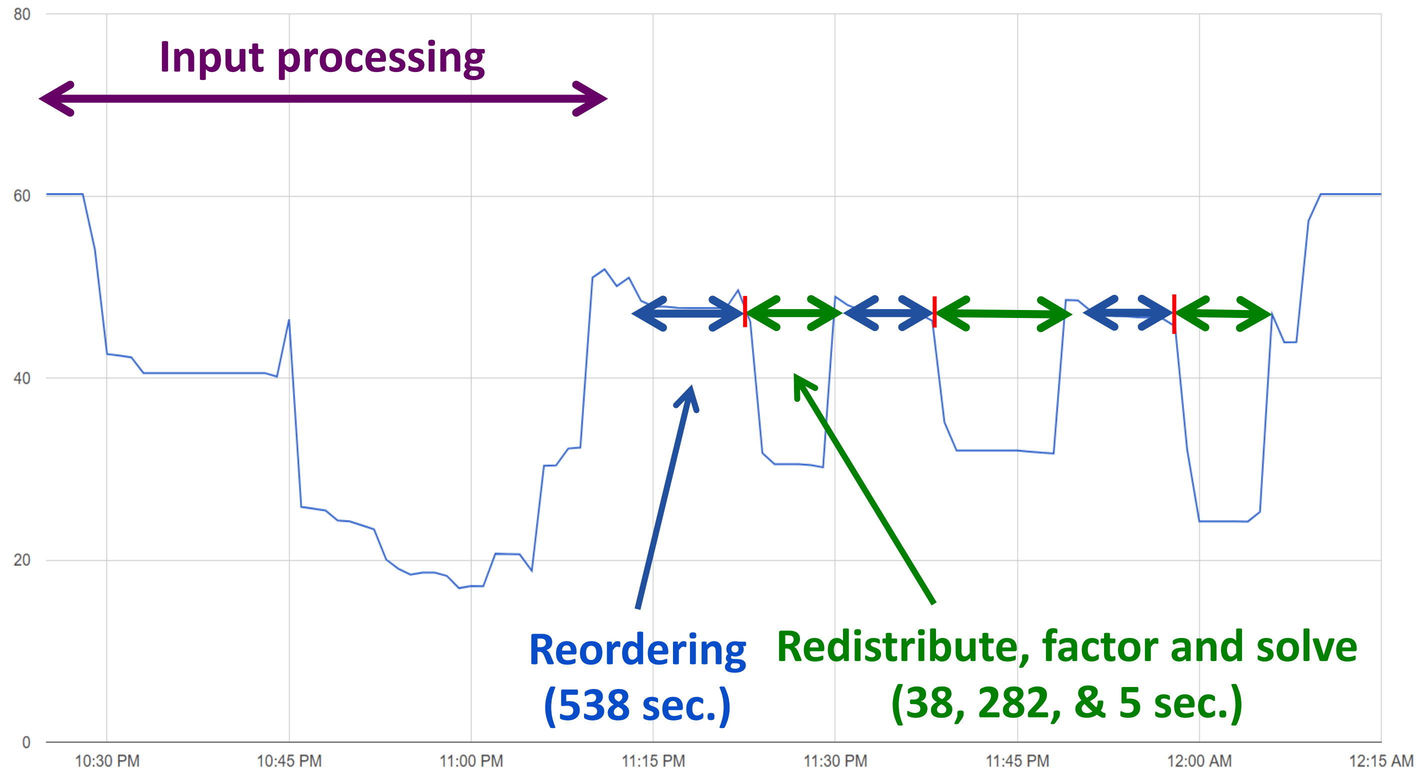
- Minimizing cost is NP-complete
- Approximate with nested dissection

As you add processors, eventually reordering and factorization dominate the run time.

- Factorization scales better


Rolls-Royce Representative Engine Model (REM)

Three simulated load steps of the Rolls-Royce REM.

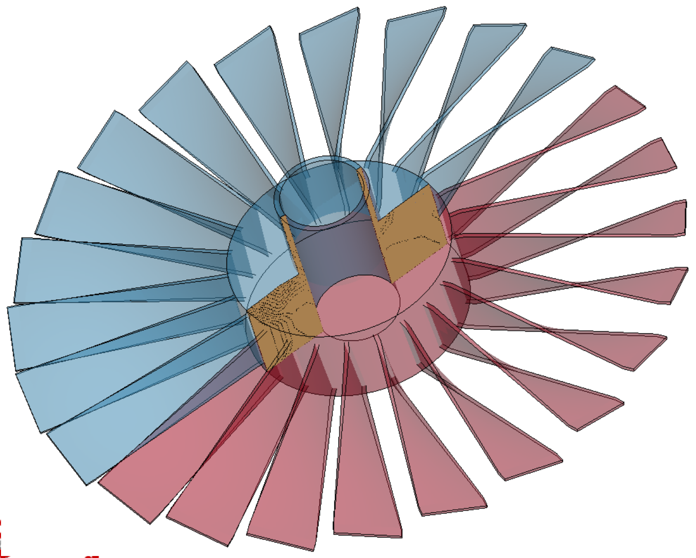


LS-DYNA memory usage (rank 0) vs. time on NCSA Blue Waters
256 MPI ranks, 8 threads each

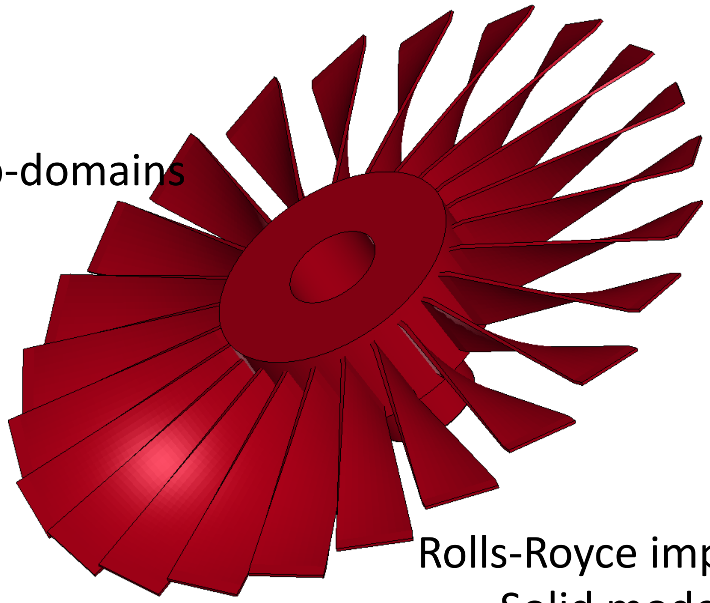
Reordering with nested dissection

LS-GPart nested dissection algorithm

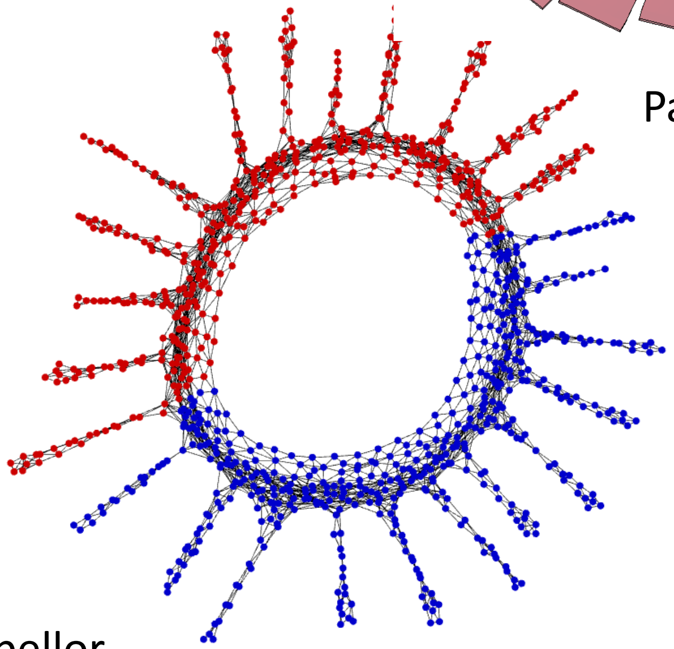
- For each domain:
  - Coarsen graph
  - Partition coarsened graph
    - NP-complete
  - Project to fine graph
  - Refine
  - Remove separator
  - Recurse on resulting sub-domains



Partitioned model



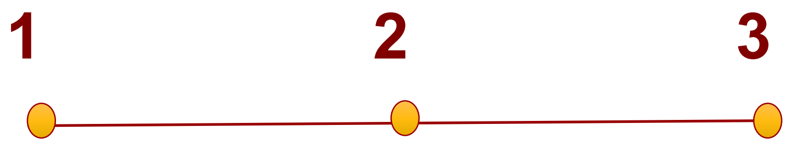
Rolls-Royce impellor
Solid model



Coarse graph

Graph partitioning as an Ising model

$$H = A \underbrace{\left(\sum_{i\in V}\sigma_i\right)^2}_{\text{penalize config. that doesn't bisect}} + B\sum_{ij\in E}\underbrace{\frac{1-\sigma_i\sigma_j}{2}}_{\text{penalty } B \text{ for each edge btwn two subsets}}$$



$$\underset{\text{Solutions}^T}{\begin{pmatrix} - & - & - \\ - & - & + \\ - & + & - \\ - & + & + \\ + & - & - \\ + & - & + \\ + & + & - \\ + & + & + \end{pmatrix}} * \underset{\mathbf{H}}{\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}} * \underset{\text{All possible solutions}}{\begin{pmatrix} - & - & - & - & + & + & + & + \\ - & - & + & + & - & - & + & + \\ - & + & - & + & - & + & - & + \end{pmatrix}} = \underset{\text{Energy}}{\begin{pmatrix} 5 \\ 1 \\ 5 \\ 1 \\ 1 \\ 5 \\ 1 \\ 5 \end{pmatrix}}$$

The objective is to minimize the product of

$$x^T * H * x$$

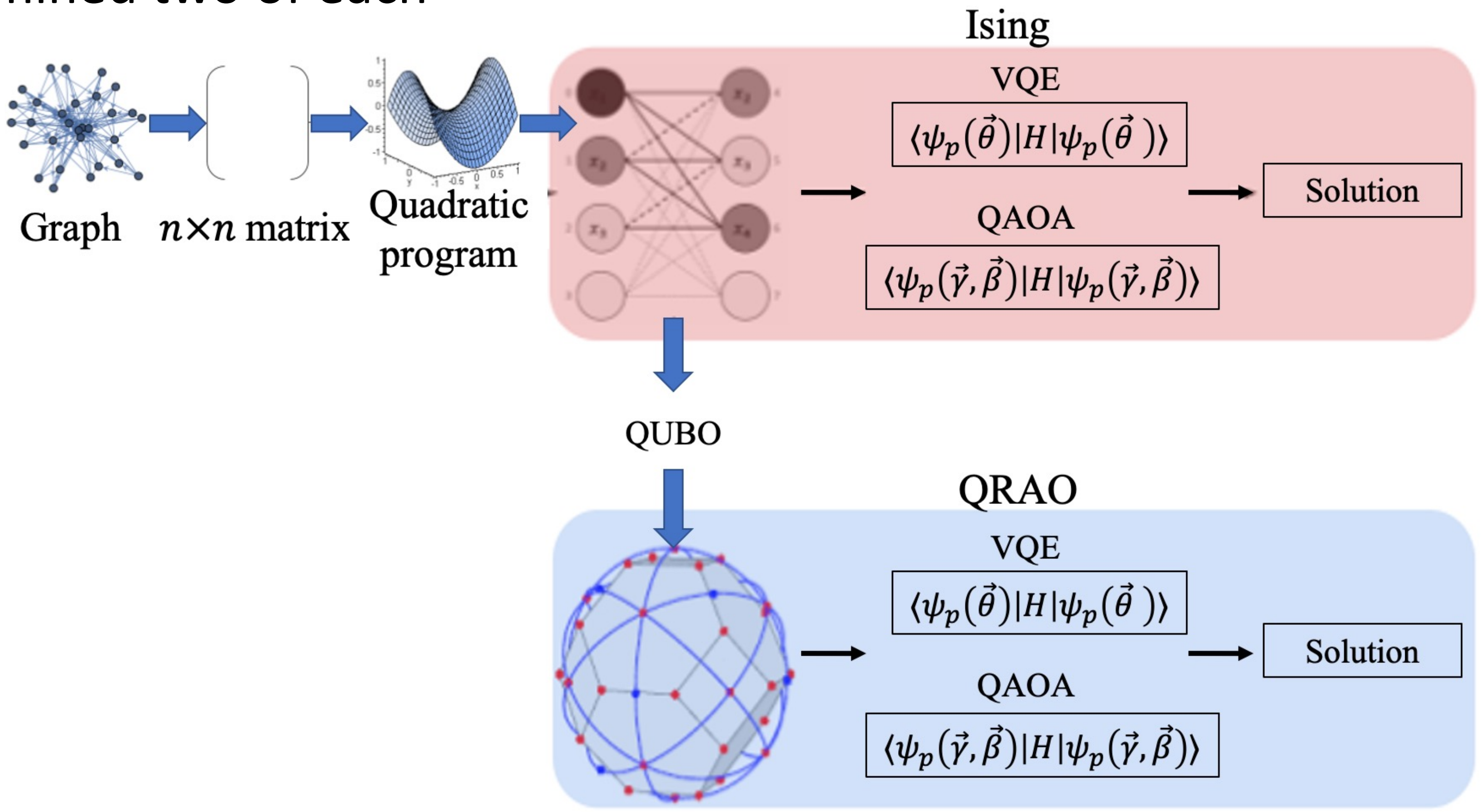
Program H so as to add energy penalties to configurations you don't want

J Phy A19 1605 (1986)
Frontiers in Physics 2, 5 (2014)

Ansys

Encoding and solutions considered

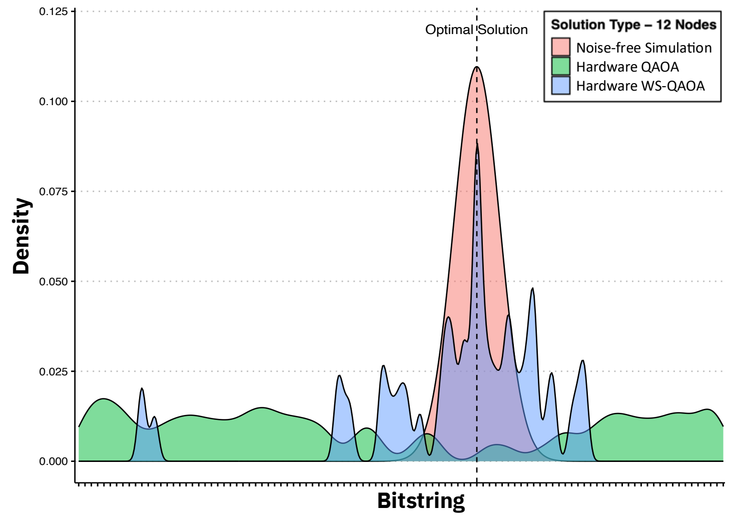
- The are multiple possible encodings and multiple possible solution algorithms
- We examined two of each


Ising
VQE
$\langle\psi_p(\vec{\theta})|H|\psi_p(\vec{\theta})\rangle$
QAOA
$\langle\psi_p(\vec{\gamma},\vec{\beta})|H|\psi_p(\vec{\gamma},\vec{\beta})\rangle$
Solution
Graph
$n\times n$ matrix
Quadratic program
QUBO
QRAO
VQE
$\langle\psi_p(\vec{\theta})|H|\psi_p(\vec{\theta})\rangle$
QAOA
$\langle\psi_p(\vec{\gamma},\vec{\beta})|H|\psi_p(\vec{\gamma},\vec{\beta})\rangle$
Solution

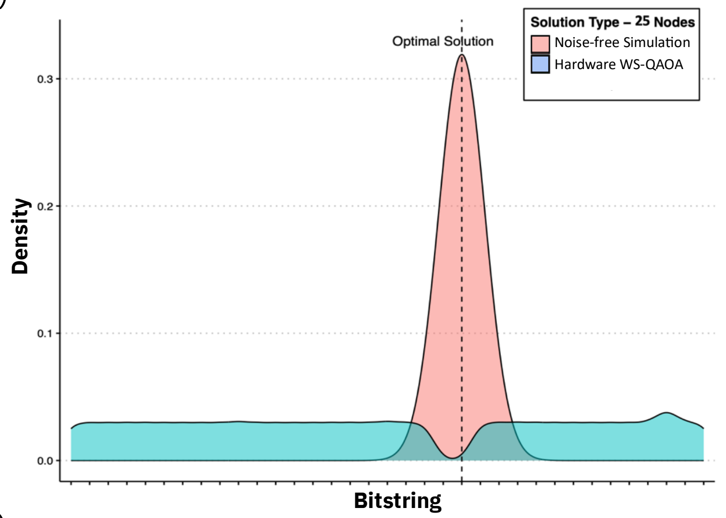
Early results

- Results on 12 and 25-qubit IBM quantum computers

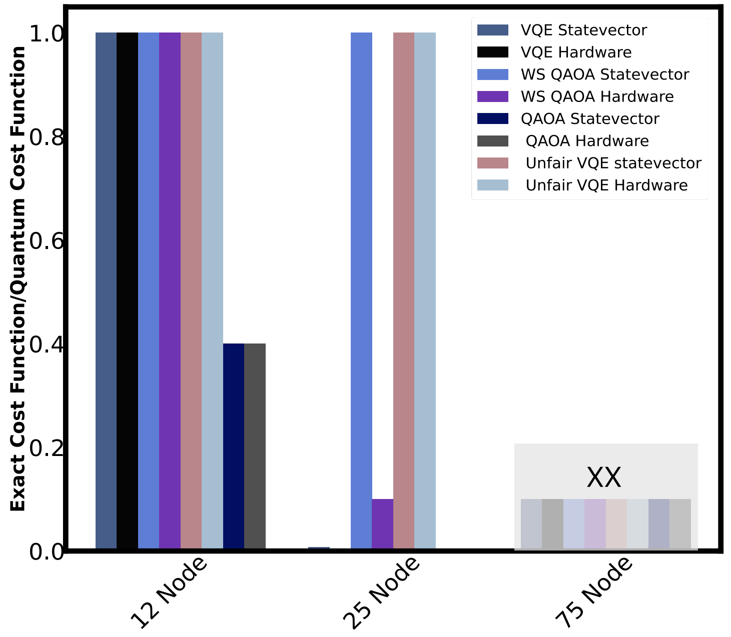
a)



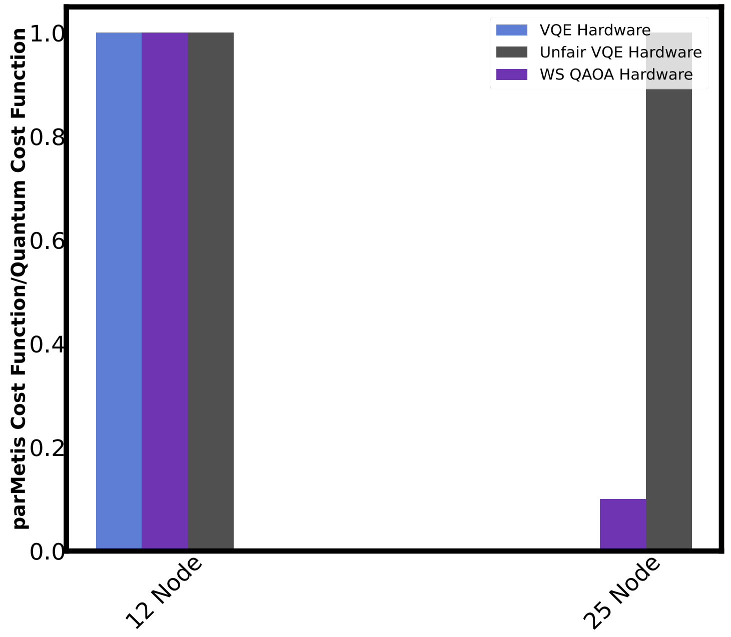
b)



c)



d)



Summary

Machine learning has increased the demand for computing power

- Demand is growing faster than Moore’s Law
- Meanwhile, Moore’s Law itself it slowing down

We have entered a new era of innovation in computing technology

- Primarily privately funded this time around
- Driven by advertising

This presents us with a rich new set of potential accelerators to explore

- Look beyond GPUs

Ansys