Case Studies on Accelerating Scientific Computing Applications with TPUs

Tianjian Lu and Yi-Fan Chen
June 2021

Google Research

Scientific Computing on TPUs

- The recent success of deep learning has spurred the new wave of hardware accelerators.
- One such example is Google's Tensor Processing Unit (TPU).
- TPU has strength in tensor operations.
- In witnessing how scientific computing applications benefit from the advancement of hardware accelerators, it is tempting to ask whether TPU be useful for scientific computing.
- We seek an answer to this question through the case studies.

Motivation

Formulation decisions

- Formulate the problem based on the hardware architecture.

Tensor operations

- Design the building blocks of the applications such as non-uniform Fourier transform, sparsifying transform, encoding of sensitivity profiles all as **tensor operations**.

Data decomposition and communication strategy

- Select highly parallelizable methods, **ADMM**, **CG**
- Minimize communication (high parallel efficiency).
- Design data decomposition to **localize tensor operations** on individual cores.

Load balancing

- Determine the size of operands for the localized tensor operations.

Hardware architecture (TPU v3) overview

- Designed as a co-processor on the I/O bus;
- four chips per board, two cores per chip;
- each board pairing up with one CPU host;
- a total number of 2048 cores in a Pod.

MXU

- Bulk of the computing power
- with 16 K multiply-accumulate (MAC) operations per clock cycle.

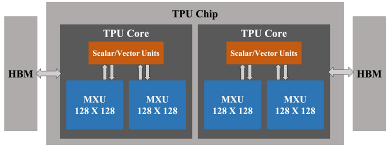
Interconnect topology

- 2D torus
- dedicated and on-device, not going through host CPU
- high-speed and low-latency

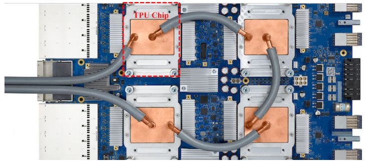
Memory

- high-bandwidth memory (HBM)

Chip/Package level



Board level



System level

Case studies

- Fourier transform
  - Discrete Fourier transform (DFT)[1]
  - Fast Fourier transform (FFT)[1]
  - Nonuniform fast Fourier transform (NUFFT)[2]
- Linear system solver
  - Conjugate gradient (CG) method[3]
- Numerical optimization
  - Alternating direction method of multipliers (ADMM)[3]
- The applications in medical imaging[3]

1. Lu, Tianjian, Yi-Fan Chen, Blake Hechtman, Tao Wang, and John Anderson. "Large-scale discrete Fourier transform on TPUs." *IEEE Access* (2021).
2. Lu, Tianjian, Thibault Marin, Yue Zhuo, Yi-Fan Chen, and Chao Ma. "Nonuniform Fast Fourier Transform on Tpus." In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pp. 783-787. IEEE, 2021.
3. Lu, Tianjian, Thibault Marin, Yue Zhuo, Yi-Fan Chen, and Chao Ma. "Accelerating MRI Reconstruction on TPUs." In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1-9. IEEE, 2020.

Case Study 1: Discrete Fourier Transform on TPUs

- DFT is critical in many scientific and engineering applications.
- General form of DFT

$$X_k \triangleq X(z_k) = \sum_{n=0}^{N-1} x_n z_k^{-n}$$

- Matrix Form

$$\{X\} = [V]\,\{x\}\,,$$

where

$$\begin{aligned} \{X\} &= (X(z_0), X(z_1), \cdots, X(z_{N-1}))^{\mathrm{T}}\,, \\ \{x\} &= (x_0, x_1, \cdots, x_{N-1})^{\mathrm{T}}\,, \end{aligned}$$

and

$$[V] = \begin{pmatrix} 1 & z_0^{-1} & z_0^{-2} & \cdots & z_0^{-(N-1)} \\ 1 & z_1^{-1} & z_1^{-2} & \cdots & z_1^{-(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & z_{N-1}^{-1} & z_{N-1}^{-2} & \cdots & z_{N-1}^{-(N-1)} \end{pmatrix}.$$

Case Study 1: Discrete Fourier Transform on TPUs

Google Research

- Three-dimensional (3D) DFT

$$X(z_{1k}, z_{2k}, z_{3k}) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \sum_{n_3=0}^{N_3-1} x(n_1, n_2, n_3) z_{1k}^{-n_1} z_{2k}^{-n_2} z_{3k}^{-n_3}$$

- Matrix Form

$$\{X\} = [V_1] \otimes [V_2] \otimes [V_3] \{x\} \,,$$

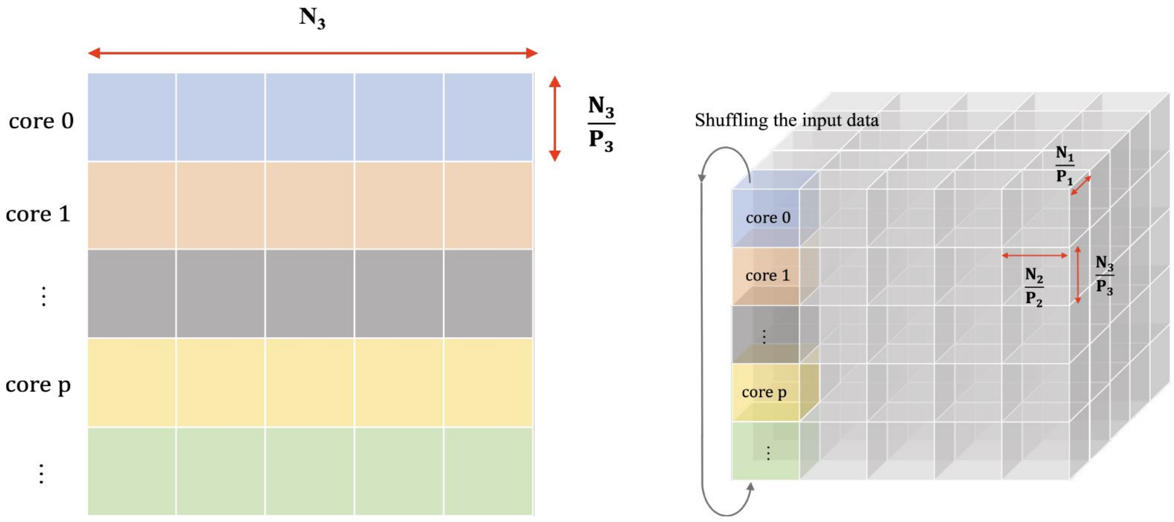
where

$$[V_j] \;=\; \begin{pmatrix} 1 & z_{j0}^{-1} & z_{j0}^{-2} & \cdots & z_{j0}^{-(N_j-1)} \\ 1 & z_{j1}^{-1} & z_{j1}^{-2} & \cdots & z_{j1}^{-(N_j-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & z_{j,N_j-1}^{-1} & z_{j,N_j-1}^{-2} & \cdots & z_{j,N_j-1}^{-(N_j-1)} \end{pmatrix}$$

$j \in \{1, 2, 3\}$.

Case Study 1: The One-Shuffle Algorithm on TPUs

- Advantages of the one-shuffle scheme:
  - tensor operations are **localized** on individual cores;
  - communication (sending and receiving data among cores) takes place **along the same direction** on the interconnect network;
  - and it achieves **high parallel efficiency**.

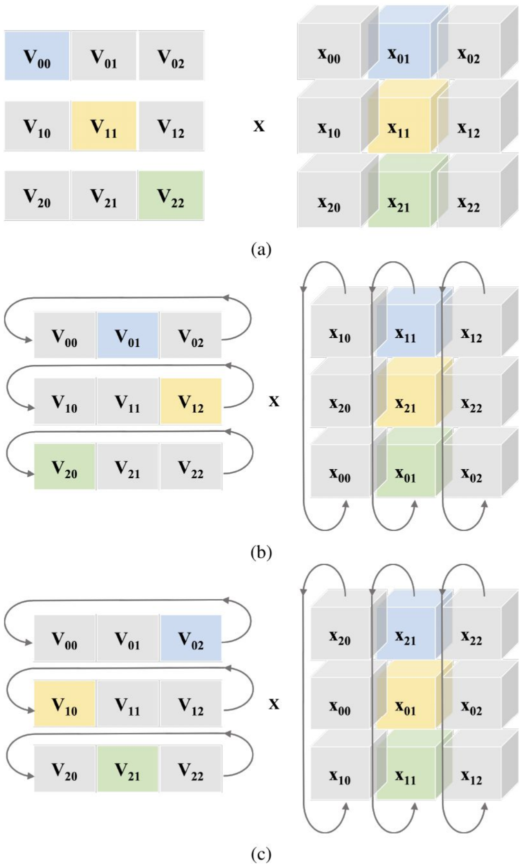


(a) (b)

Case Study 1: The One-Shuffle Algorithm on TPUs

Algorithm 1 The one-shuffle scheme

function ONE_SHUFFLE(v, x, core_idx, num_cores, src_tgt_pairs)
    iteration_idx ← 0
    slice_idx ← core_idx
    x_out ← `einsum`(v[slice_idx], x)
    slice_idx ← `mod`(slice_idx + 1, num_cores)
    while iteration_idx < num_cores − 1 **do**
        x ← `collective_permute`(x, src_tgt_pairs)
        x_out ← x_out + `einsum`(v[slice_idx], x)
        slice_idx ← `mod`(slice_idx + 1, num_cores)
        iteration_idx ← iteration_idx + 1
    return x_out

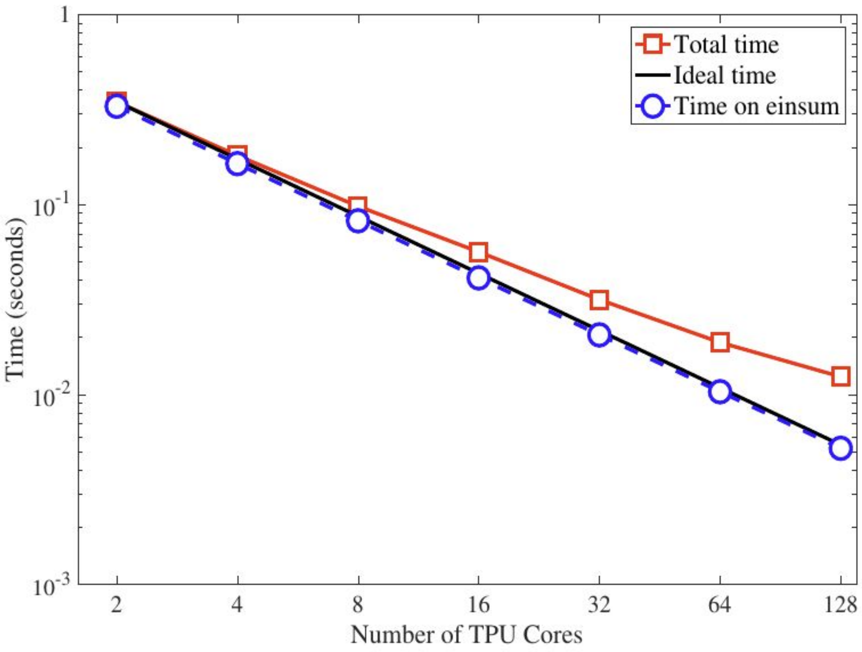


Case Study 1: Discrete Fourier Transform on TPUs

- Strong scaling analysis
- Define ideal time of linear scaling as a reference

$$\text{ideal time} = \frac{T_2}{\frac{N_{\text{core}}}{2}}$$

- The total computation time has a **close-to-linear scaling**.

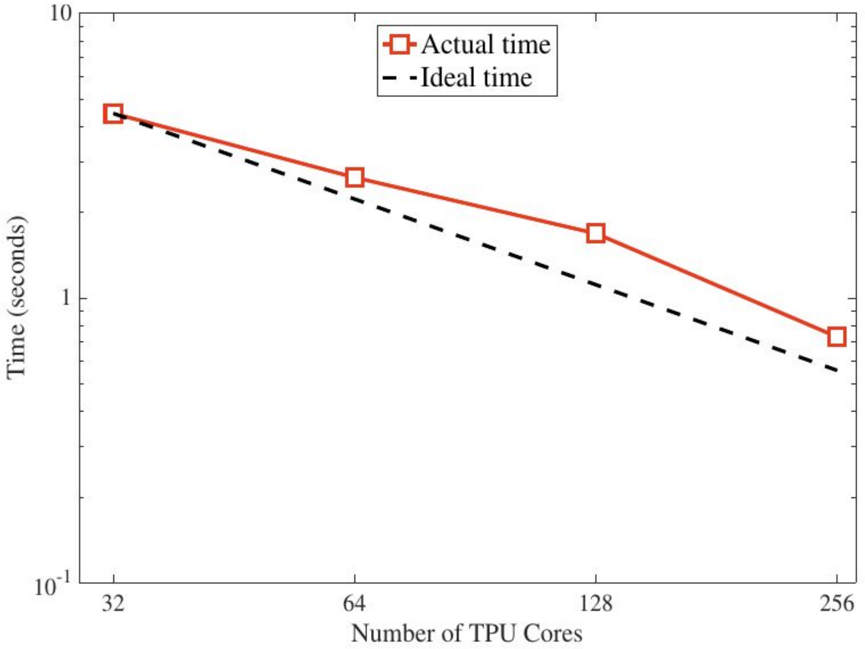


2D DFT, fixed problem size of 8192 x 8192; up to 128 TPU cores being used.

- Strong scaling analysis:
- Define ideal time of linear scaling as a reference

$$\text{ideal time} = \frac{T_{32}}{\frac{N_{\text{core}}}{32}}$$

- The total computation time has a **close-to-linear scaling**.



3D DFT, fixed problem size of 2048 x 2048 x 2048, and up to 256 TPU cores being used.

Case Study 2: Fast Fourier Transform on TPUs

Google Research

- The FFT formulation starts with

$$X_k \triangleq \sum_{n=0}^{N-1} x_n e^{-j2\pi \frac{nk}{N}}.$$

- The global index n can be expressed as

$$n = Pl + \beta,$$

where $l = 0, 1, \cdots, \frac{N}{P} - 1$ and $\beta = 0, 1, \cdots, P - 1$.

- Rewrite as **phase adjustment** and **localized transform**

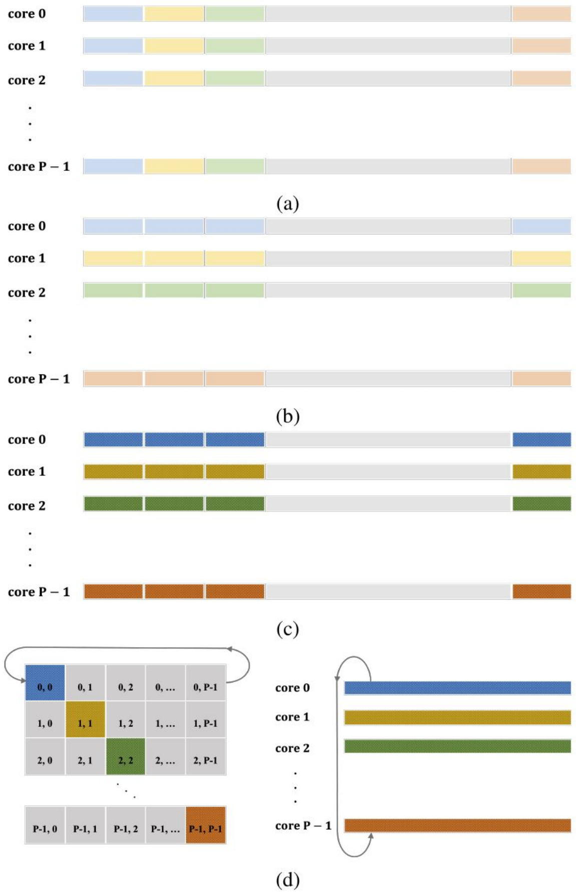
$$X_k \triangleq \sum_{n=0}^{N-1} x_{(Pl+\beta)} e^{-j2\pi \frac{(Pl+\beta)k}{N}}$$

$$= \boxed{\sum_{\beta=0}^{P-1} e^{-j2\pi \frac{\beta k}{N}}} \boxed{\left(\sum_{l=0}^{\frac{N}{P}-1} x_{(Pl+\beta)} e^{-j2\pi \frac{lk}{\frac{N}{P}}} \right)}.$$

Case Study 2: Fast Fourier Transform on TPUs

The parallel algorithm

- (a) Data decomposition.
- (b) The gathering of input for in-order transform.
- (c) The transform performed locally on individual cores.
- (d) Applying phase adjustment with the **one-shuffle algorithm**.

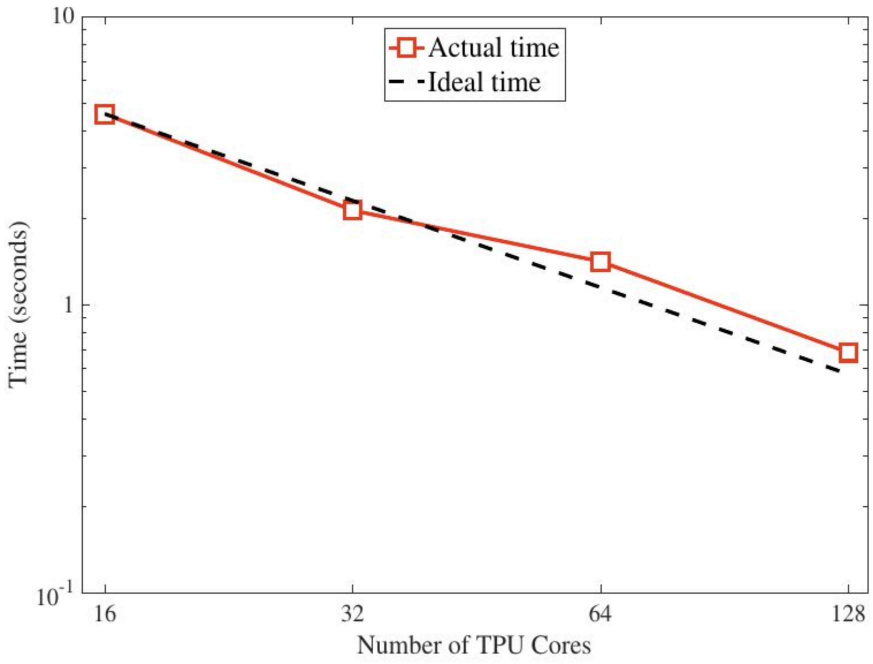


Case Study 2: Fast Fourier Transform on TPUs

- Strong scaling analysis.
- Define ideal time of linear scaling as a reference

$$\text{ideal time} = \frac{T_{16}}{\frac{N_{\text{core}}}{16}}$$

- The total computation time has a **close-to-linear scaling**.



3D FFT, fixed problem size of 2048 x 2048 x 2048, and up to 128 TPU cores being used.

Case Study 2: Fast Fourier Transform on TPUs

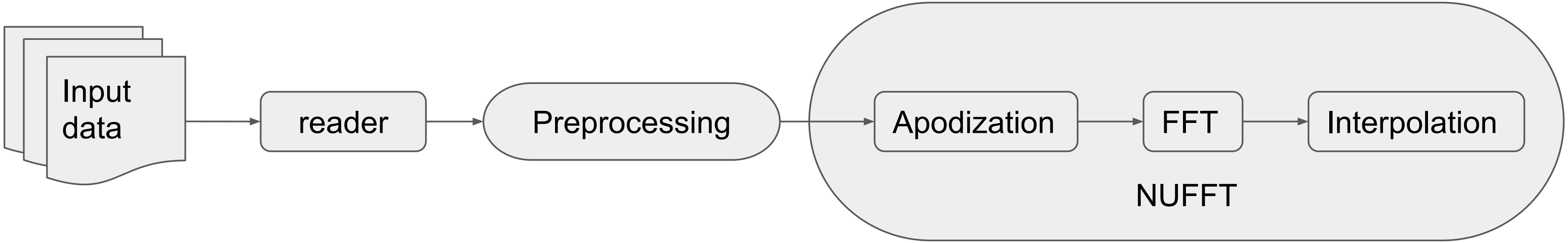
- The computation time of a few 3D DFT and FFT examples on a full TPU v3 Pod with 2048 cores.
- The runtimes reported in the table are for **complex** transforms.
- As a reference, the runtime of a **real** FFT for the problem size 8192 x 8192 x 8192 on **CPUs**: 2048 nodes of Fujitsu PRIMERGY CX1640 M1 cluster is 5.36 seconds (converted from 10 TFlops, D. Takahashi 2019).

No.	Problem size	Time (seconds)	
		DFT	FFT
1	$8192 \times 8192 \times 8192$	12.66	8.30
2	$4096 \times 4096 \times 4096$	1.07	1.01
3	$2048 \times 2048 \times 2048$	0.120	0.118
4	$1024 \times 1024 \times 1024$	0.0220	0.0148

D. Takahashi, "Implementation of parallel 3-D real FFT with 2-D decomposition on Intel Xeon Phi clusters," in *International Conference on Parallel Processing and Applied Mathematics.* Springer, 2019, pp. 151–161.

Case Study 3: NUFFT on TPUs

Google Research



- NUFFT

$$s\left(k_{x,m}, k_{y,m}\right) = \sum_{n=1}^{N} \rho_n e^{-i2\pi\left(k_{x,m}x_n + k_{y,m}y_n\right)}$$

where $\left(k_{x,m}, k_{y,m}\right), m = 1, 2, \cdots, M$ represents the k-space coordinates on a nonuniform grid, $\left(x_n, y_n\right), n = 1, 2, \cdots, N$ represents the spatial coordinates on a uniform grid, and ρ_n denotes the image intensity on grid $\left(x_n, y_n\right)$.

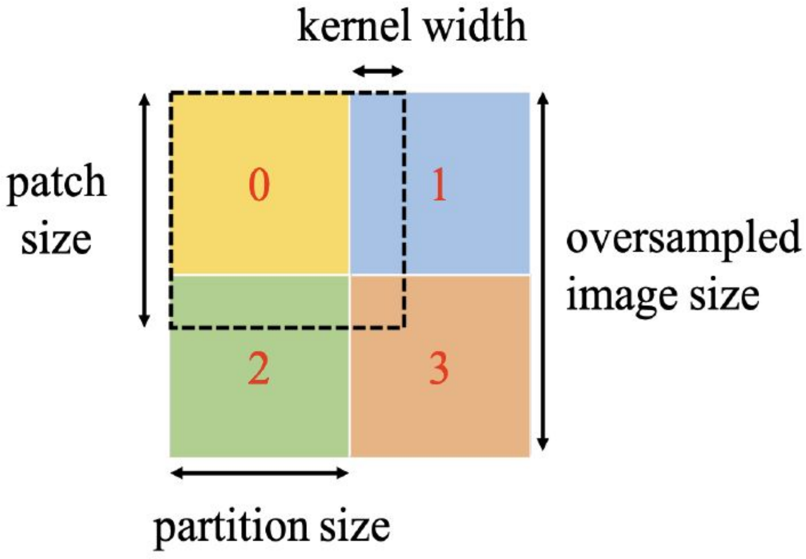
- Matrix form

$$\mathbf{s} = \mathbf{CFD}\boldsymbol{\rho}$$

where $\mathbf{D}$ is the apodization operator, $\mathbf{F}$ denotes the FFT operator, and $\mathbf{C}$ represents the interpolation operator.

Google Research

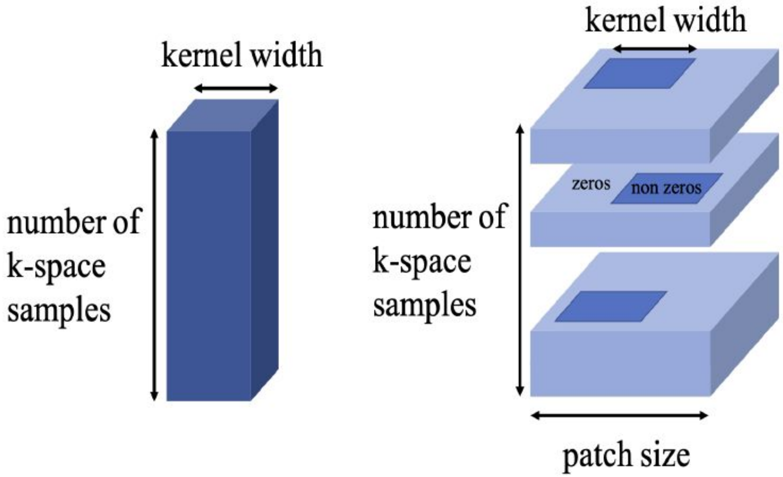
Preprocessing



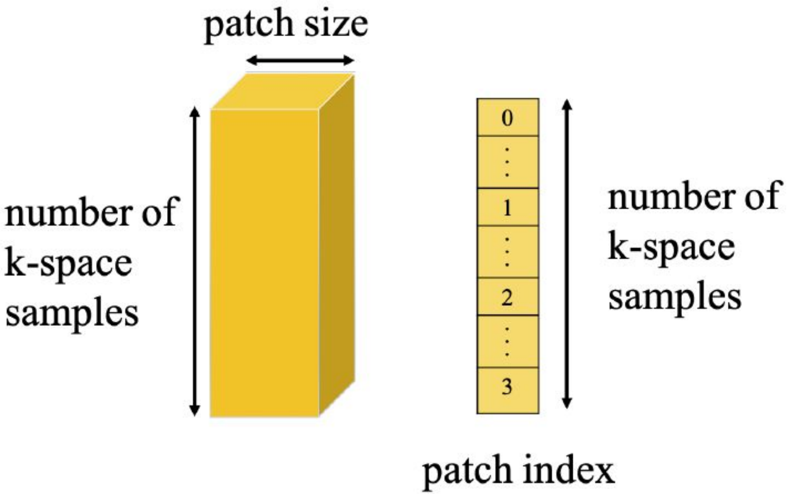
Perform checker-board partition to an oversampled image into patches.

Google Research

Preprocessing



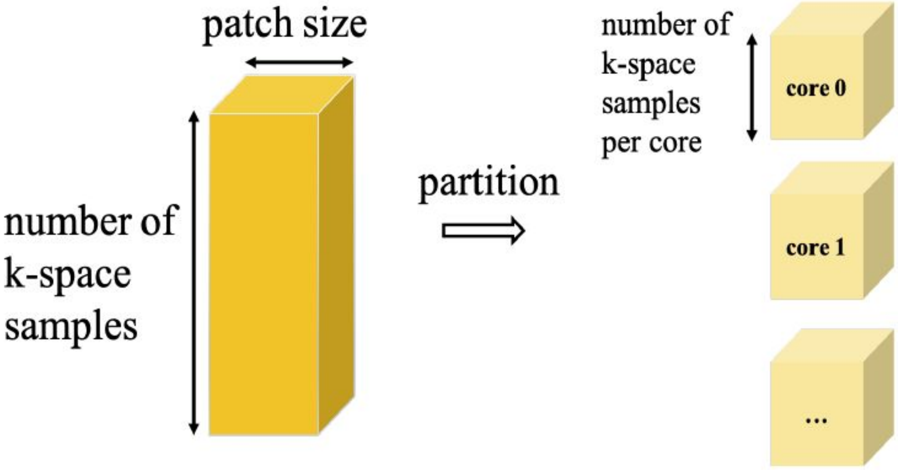
Pad tensors of kernel coefficients with zeros.



Shuffle kernel coefficients along the k-space dimension.

Google Research

Load balancing

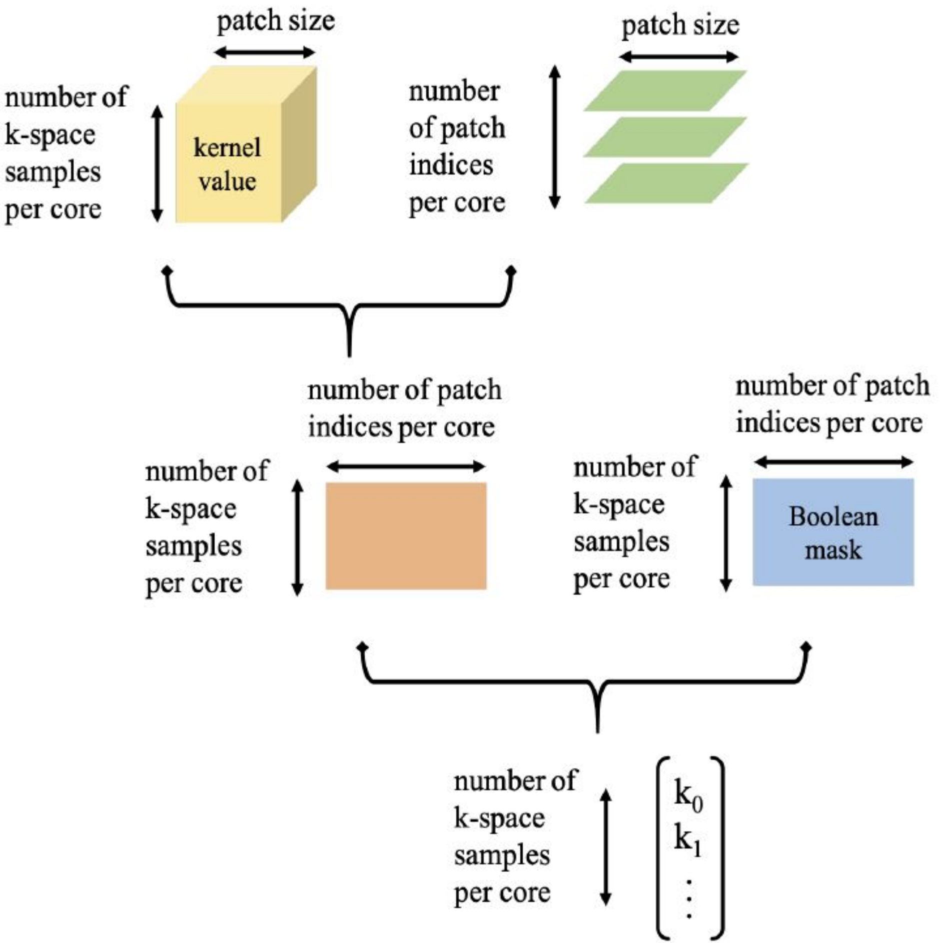


Each TPU core contains partial k-space information.

Interpolation in Forward NUFFT

Transform onto nonuniform grids:

- **Tensor contraction** between kernel coefficients and patches along patch dimension.
- **Matrix multiplication** with a Boolean mask.
- **Reduce sum** along the patch dimension.



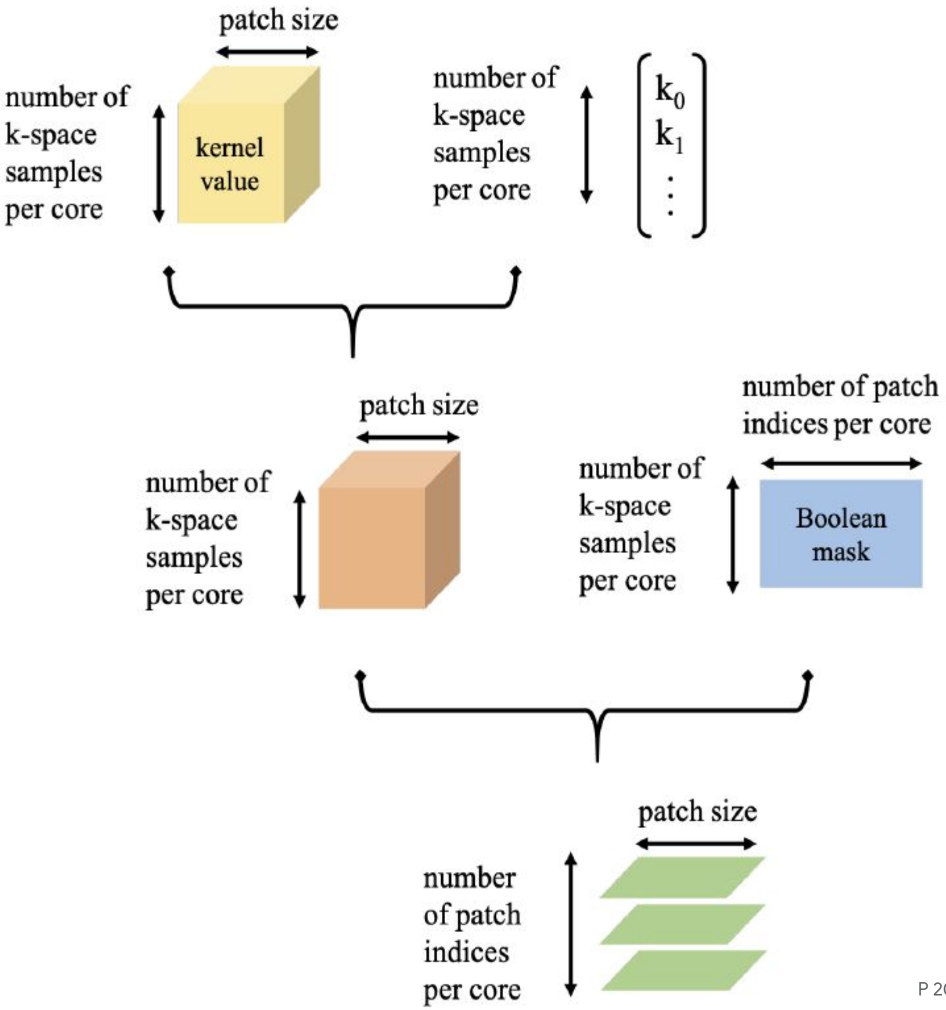
Case Study 3: NUFFT on TPUs

Interpolation in Adjoint NUFFT

Transform onto the uniform grids:

- **Scale** the kernel coefficients with k-space data
- **Tensor contraction** with a Boolean mask along the k-space dimension.

Patches are assembled back to the image.



Case Study 3: NUFFT on TPUs

Computation on three types of hardware for forward NUFFT:

- CPU: Intel(R) Xeon(R) Silver 4110 8-core 2.10 GHz
- GPU: Nvidia V100
- TPU: one TPU v3 unit (eight cores).

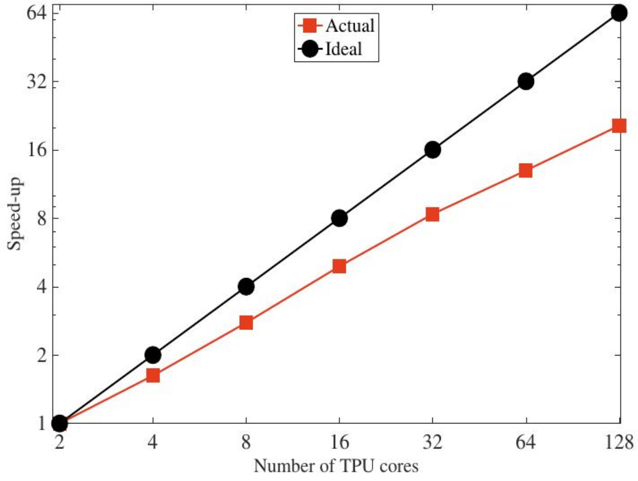
	Time (ms)		
Image size	CPU	GPU	TPU
64 x 64	18.97	2.89	0.11
128 x 128	75.04	3.08	0.36
256 x 256	250.5	3.02	1.23
512 x 512	1135.19	3.04	5.41

Case Study 3: NUFFT on TPUs

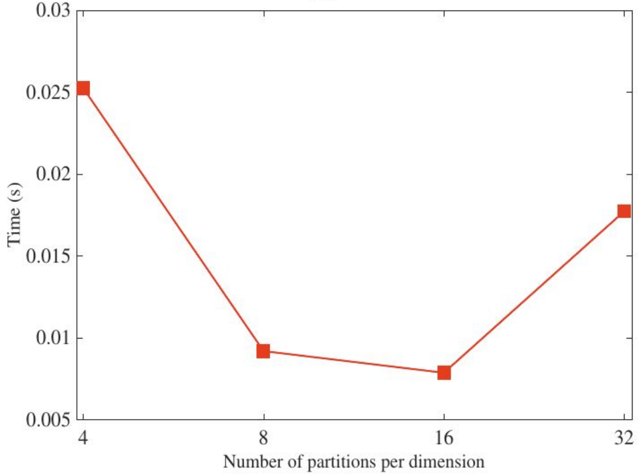
Google Research

Adjoint NUFFT

- Image size: 512 x 512
- Oversampling factor: 2
- Number of points in k-space: 412,160
- Strong scaling:
  - Number of TPU cores: 2 to 128
  - Number of partitions: 16 along each dimension
- Computation time versus partitions
  - 16 TPU cores



(a)



(b)

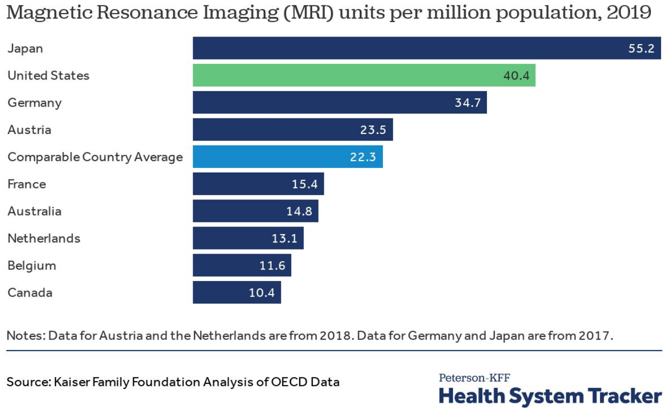
Case Study 4: ADMM and CG on TPUs

Google Research

Magnetic resonance imaging (MRI) is a **powerful** medical imaging modality:

- non-invasive
- excellent soft-tissue contrast
- high spatial resolution

MRI has revolutionized the field of medical imaging since its invention in 1970s.



Computation in MR image reconstruction is now the new **bottleneck**:

- MR data acquisition speed is approaching the physical limits.
- Further acceleration of MR requires breaking the Nyquist sampling criterion by sparse sampling and constrained image reconstruction.
- However, the state-of-the-art MR image reconstruction methods often build upon **large-scale**, **iterative**, **optimization** algorithms
  - with extensive usage of **non-uniform Fourier transform**
  - **computationally infeasible** for practical clinical use.

Case Study 4: ADMM and CG on TPUs

- MRI signal model

$$
\begin{aligned}
d_{\kappa,\gamma} &= \sum_n s_{n,\gamma} \rho_n e^{-i2\pi \mathbf{k}_\kappa \cdot \mathbf{r}_n} \\
&= [\mathbf{F}(\boldsymbol{\rho})]_{\kappa,\gamma}
\end{aligned}
$$

- In compressed sensing, one reconstructs an image from the undersampled k-space data by solving

$$
\min_{\boldsymbol{\rho}} \; \underbrace{\|\mathbf{F}(\boldsymbol{\rho}) - \mathbf{d}\|_2^2}_{\text{Data fidelity}} + \lambda \underbrace{\|\boldsymbol{\Theta}(\boldsymbol{\rho})\|_1}_{\text{Sparsity constraint}}
$$

- We use the Alternating Direction Method of Multipliers (**ADMM**) to solve the large-scale convex optimization problem

$$
\begin{aligned}
&\min_{\boldsymbol{\rho}} \quad \|\mathbf{F}(\boldsymbol{\rho}) - \mathbf{d}\|_2^2 + \lambda \|\boldsymbol{\mu}\|_1 \\
&\text{s.t.} \quad \boldsymbol{\Theta}(\boldsymbol{\rho}) - \boldsymbol{\mu} = 0
\end{aligned}
$$

- ADMM consists of three updates:

$$
\begin{aligned}
\boldsymbol{\mu}^{m+1} &= \operatorname*{argmin}_{\boldsymbol{\mu}} \; \lambda \|\boldsymbol{\mu}\|_1 + \frac{\beta}{2} \|\boldsymbol{\Theta}(\boldsymbol{\rho}^m) - \boldsymbol{\mu} + \boldsymbol{\eta}^m\|_2^2 \\
\boldsymbol{\rho}^{m+1} &= \operatorname*{argmin}_{\boldsymbol{\rho}} \; \|\mathbf{F}(\boldsymbol{\rho}) - \mathbf{d}\|_2^2 + \frac{\beta}{2} \|\boldsymbol{\Theta}(\boldsymbol{\rho}) - \boldsymbol{\mu}^{m+1} + \boldsymbol{\eta}^m\|_2^2 \\
\boldsymbol{\eta}^{m+1} &= \boldsymbol{\eta}^m + \boldsymbol{\Theta}(\boldsymbol{\rho}^{m+1}) - \boldsymbol{\mu}^{m+1}
\end{aligned}
$$

Case Study 4: ADMM and CG on TPUs

- The update of the auxiliary variable has a closed-form solution

$$\boldsymbol{\mu}^{m+1} = S_{\frac{\lambda}{\beta}}\left(\boldsymbol{\Theta}\left(\boldsymbol{\rho}^{m}\right) + \boldsymbol{\eta}^{m}\right)$$

and the element-wise soft thresholding can be written as

$$S_{\frac{\lambda}{\beta}}(\alpha) = \begin{cases} \alpha - \frac{\lambda}{\beta}, & \alpha > \frac{\lambda_1}{\beta} \\ 0, & |\alpha| \leq \frac{\lambda}{\beta} \\ \alpha + \frac{\lambda}{\beta}, & \alpha < -\frac{\lambda_1}{\beta} \end{cases}$$

- The update of the primal variable (complex image intensities) can be considered as a regularized least square problem.
- The necessary and sufficient optimality condition is

$$\mathbf{A}\boldsymbol{\rho}^{m+1} = \mathbf{b}$$

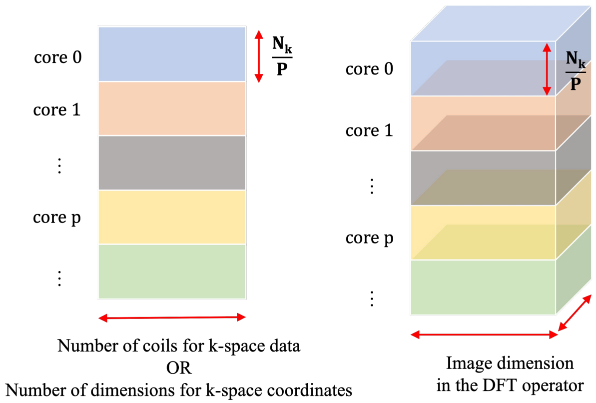
where

$$\mathbf{A} = \mathbf{F}^{\mathrm{H}}\mathbf{F} + \frac{\beta}{2}\boldsymbol{\Theta}^{\mathrm{H}}\boldsymbol{\Theta},$$

$$\mathbf{b} = \mathbf{F}^{\mathrm{H}}\mathbf{d} + \frac{\beta}{2}\boldsymbol{\Theta}^{\mathrm{H}}\left(\boldsymbol{\mu}^{m+1} - \boldsymbol{\eta}^{m}\right)$$

- This is solved iteratively by using the **conjugate gradient (CG)** method.

Case Study 4: ADMM and CG on TPUs



Data decomposition applied to data and DFT operator.

- The data decomposition is applied to the k-space.
- DFT and sparsifying transform operations
  - The DFT operation and its adjoint are formulated as **tensor contractions** (tf.einsum).
  - The sparsifying transform operation and its adjoint are formulated as **convolutions** (tf.nn.conv1d).

Algorithm 1 The generation of the DFT operator on TPUs

```
function MAP_TO_UNIT_CIRCLE(k_r_product)
    return exp(−i 2π k_r_product)

function GEN_DFT_OPERATOR(k_coord, image_coord)
    kr_dim0 ← map_to_unit_circle(
                  einsum(image_coord[0], k_coord[:, 0]))
    kr_dim1 ← map_to_unit_circle(
                  einsum(image_coord[1], k_coord[:, 1]))
    dft_op ← einsum(kr_dim0, kr_dim1)
    return dft_op
```

Case Study 4: ADMM and CG on TPUs

- Communication strategy
- ADMM has three updates per iteration:
  - The update of the **auxiliary** variable is **local**.
  - The update of the **dual** variable is **local**.
  - The update of the **primal** variable is through CG solver, **requiring communication** (tf.cross_replica_sum) to sum the partial images across TPU cores such that all cores start the new CG iteration with the same image.

Algorithm 4 ADMM on TPUs

```
function ADMM_STEP(η, ρ, ρ0)
    μ_next ← update_auxiliary_var(ρ, η)
    ρ_next ← update_primal_var(μ_next, ρ0, η)
    α ← get_relative_diff(ρ_next, ρ)
    η_next ← update_dual_var(μ_next, ρ, η)
    return η_next, ρ_next, α
function ADMM_RECONSTRUCT(ρ0, max_iterations, rtol)
    ▷ ρ0 contains initial values of ρ and rtol is the relative
    tolerance in terms of the squared norm of the residual.
    η, ρ ← get_initial_value(ρ0)
    i ← 0
    α ← 1.0
    while i < max_iterations & α > square(rtol) do
        η, ρ, α ← admm_step(η, ρ, ρ0)
        i ← i + 1
    return ρ
```

Algorithm 2 The conjugate gradient method

```
function CG_STEP(linear_op, r, d, x, τ)    ▷ d is the
conjugate vector and r is the the residual vector.
    a_d ← linear_op(d)
    α ← divide(τ, dot_product(d, a_d))
    x_next ← x + α d
    r_next ← r − α a_d
    τ_next ← dot_product(r_next, r_next)
    β ← divide(τ_next, τ)
    d_next ← r_next + β d
    return r_next, d_next, x_next, τ_next
function CONJUGATE_GRADIENT(linear_op, b, x0,
                            max_iterations, atol)
    ▷ x0 contains initial values of x and atol is the absolute
    tolerance in terms of the norm of the residual vector.
    x ← x0
    r ← linear_op(x)
    d ← r
    τ ← dot_product(r, r)
    i ← 0
    while i < max_iterations & τ > square(atol) do
        r_next, d_next, x_next, ρ_next ← cg_step(
        linear_op, r, d, x, τ)
        i ← i + 1
    return x
```

Case Study 4: ADMM and CG on TPUs

:h

Accuracy benchmark

- The k-space data were retrospectively undersampled with an undersampling factor of eight to demonstrate the capability of compressed sensing in accelerating MR.
- The total number of k-space measurements was 19,968 (1,664 samples per coil and 12 coils in total).
- The images were reconstructed on a 128 x 64 uniform grid.
- The relative difference is about 1% for the voxels within the phantom, which is satisfactory.

ADMM				CG	
Regularization parameter	Augmented Lagrangian parameter	Relative tolerance	Maximum number of iterations	Absolute tolerance	Maximum number of iterations
1e-7	1.0	1e-4	5	1e-6	20

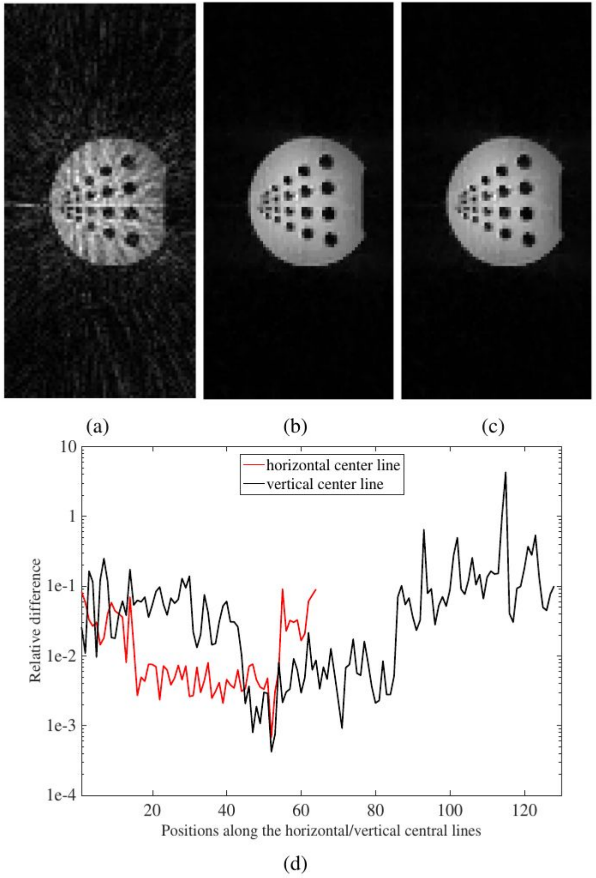


Fig. Reconstructed phantom images by using retrospectively undersampled data (a) with a single inverse DFT operation and (b) by the ADMM algorithm on CPU and (c) TPUs; and (d) the relative difference between the two images in (b) and (c) along the horizontal and vertical center lines of the image.

Case Study 4: ADMM and CG on TPUs

Parallel Efficiency

- The **strong scaling** analysis was adopted to understand the parallel efficiency.
- Phantom data were acquired by using 804 radial readouts, each with 1024 samples.
- The fully sampled k-space data were then retrospectively undersampled by a factor of eight, resulting in a total number of **1,241,076** k-space measurements (103,424 measurements per coil, 12 coils in total).
- Runtimes on CPU (**Intel(R) Xeon(R) Silver 4110** 8 core 2.1 GHz) and GPU (**NVIDIA V100** SXM2).

		Computation time (seconds)			
Hardware		CPU	GPU	TPU (number of TPU units)	
Non-uniform Fourier Transform		NUFFT	NUFFT	DFT	
Image Size	128 X 64	2.38	1.17	0.14 (1/4 unit)	0.036 (two units)
	1024 X 512	139.88	2.24	3.39 (16 units)	0.29 (128 units)

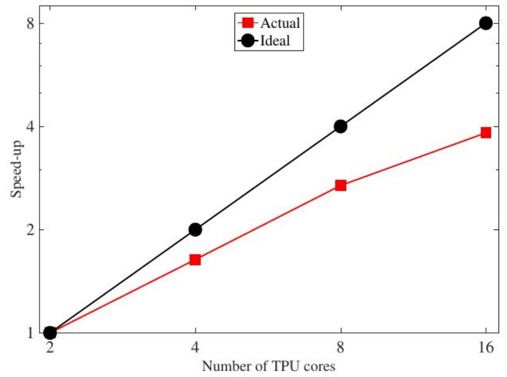


Fig. 5: The speed-up of reconstructing an image of size 128×64 with up to 16 TPU cores. The number of k-space measurements is 19,968 with 1,664 samples for each coil and 12 coils in total.

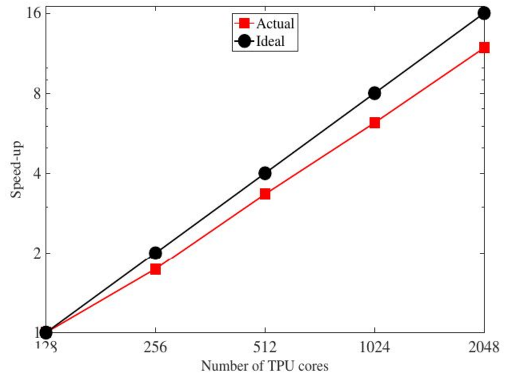


Fig. 6: The speed-up of reconstructing an image of size 1024×512 with up to 2048 TPU cores. The number of k-space measurements is 1,241,076 with 103,424 samples for each coil and 12 coils in total.

Conclusion

Google Research

- Through the few case studies, we explore using TPU for scientific computing.
- The case studies include Fourier transform (DFT, FFT, NUFFT), linear system solver (CG), numerical optimization (ADMM), and their applications in medical imaging.
- We formulate the problem and design the algorithms in accordance with TPU’s strength in tensor operations and its high-speed interconnect network.
- TPU achieves good acceleration for these scientific computing applications.

Thank You

tianjianlu@google.com