SAND2020-1981C

A Distributed-Memory Generalized Sparse Tensor Decomposition

Karen Devine, Sandia National Laboratories

Grey Ballard, Wake Forest University

Tammy Kolda, Sandia National Laboratories

With thanks to

Eric Phipps, Chris Siefert, Mark Hoemmen, James Elliott (SNL)

Srinivas Eswar, Rich Vuduc (GA Tech)

Shaden Smith (SPLATT)

February 15, 2020

Supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Sandia National Laboratories

Exceptional service in the national interest

Motivation: decomposition of large-scale tensors

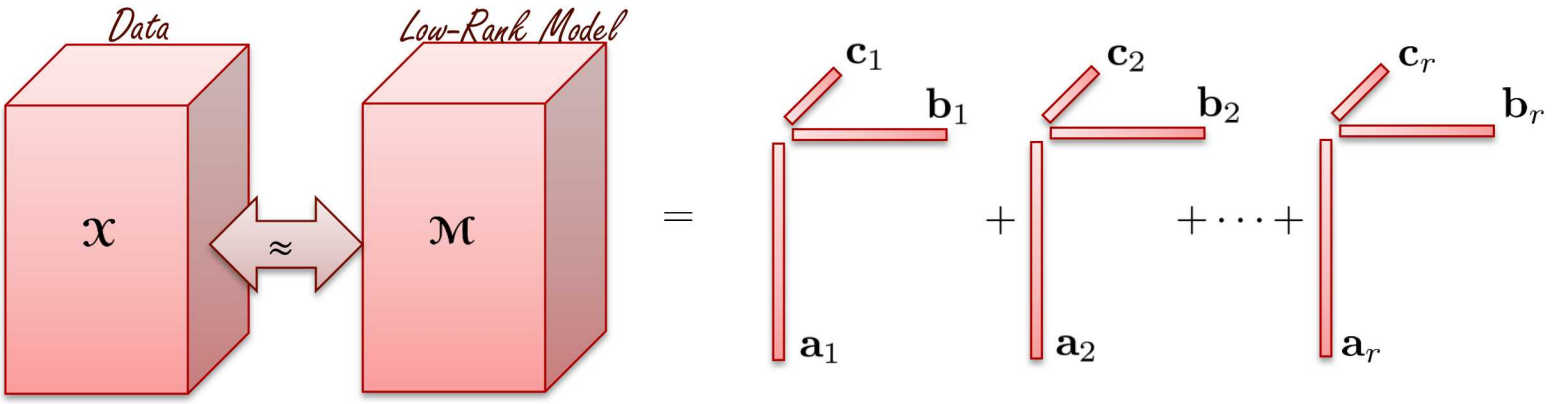
Sandia National Laboratories

- Generalized canonical polyadic (GCP) tensor decomposition
  - D. Hong, T. Kolda, J. Duersch, *SIAM Review,* 2019
  - Finds low-rank approximation of sparse tensors
  - Uses general loss function to better represent variety of tensor data
- Implementations available
  - TensorToolbox (Bader and Kolda) – Matlab
  - GenTen (Phipps and Kolda) – multicore CPU and GPU (via Kokkos)
- Very large tensors can exceed available memory in single-node systems
- SIAM CSE19: Scalable, distributed-memory, MPI-based implementation of CP-ALS using the Trilinos solver framework
- Today: Distributed memory, MPI-based implementation of GCP building from Trilinos/CP-ALS implementation

Canonical Polyadic (CP) Tensor Decomposition

Sandia National Laboratories

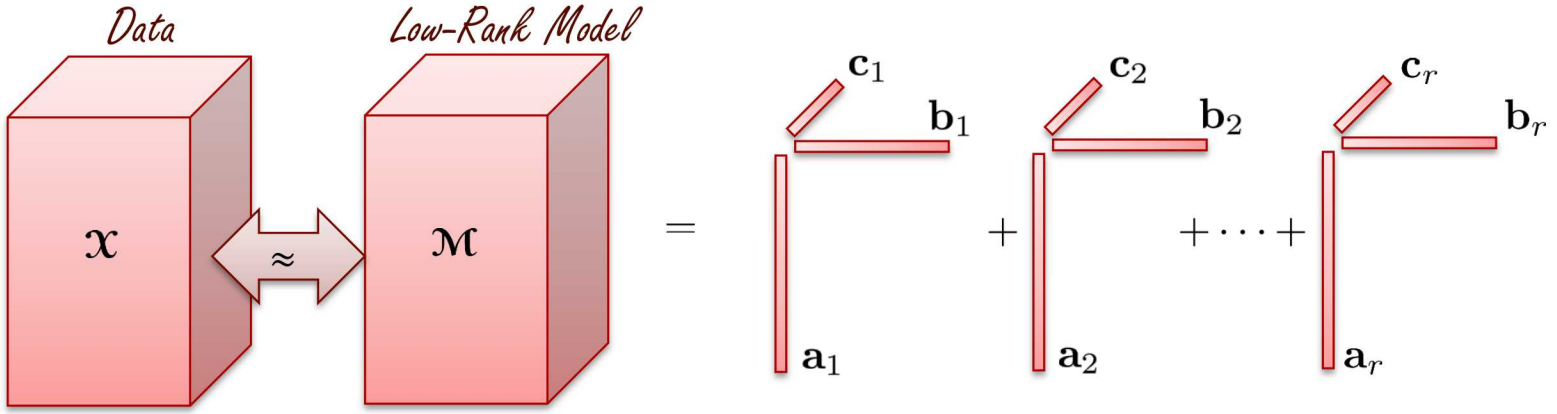
- CANDECOMP / PARAFAC (CP) decomposition
- F. Hitchcock (1927); J.D. Carroll & J-J Chang (1970); R. Harshman (1970)
- Seek low-rank approximation of tensor data



$$\underbrace{\mathcal{X} \approx \mathcal{M}}_{m \times n \times p} = \sum_{\alpha=1}^{r} \mathbf{a}_\alpha \circ \mathbf{b}_\alpha \circ \mathbf{c}_\alpha = [\![\underbrace{\mathbf{A}}_{m \times r}, \overbrace{\mathbf{B}}^{n \times r}, \underbrace{\mathbf{C}}_{p \times r}]\!]$$

Find CP decomposition by solving optimization problem

- Generalized CP (GCP) decomposition (Hong, Kolda, Duersch, 2018) takes user-defined loss function $f(x_{ijk}, m_{ijk})$



$$\mathcal{X} \approx \mathcal{M} = \sum_{\alpha=1}^{r} \mathbf{a}_\alpha \circ \mathbf{b}_\alpha \circ \mathbf{c}_\alpha = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

$f(x, m)$ provided by user

Optimization Problem: $\min \|\mathcal{X} - \mathcal{M}\|^2 \equiv \sum_i \sum_j \sum_k f(x_{ijk}, m_{ijk})$

Generalized loss functions represent variety of tensor data

Sandia National Laboratories

- For example, CP-ALS (alternating least squares) uses Normal loss function; good for Gaussian data:

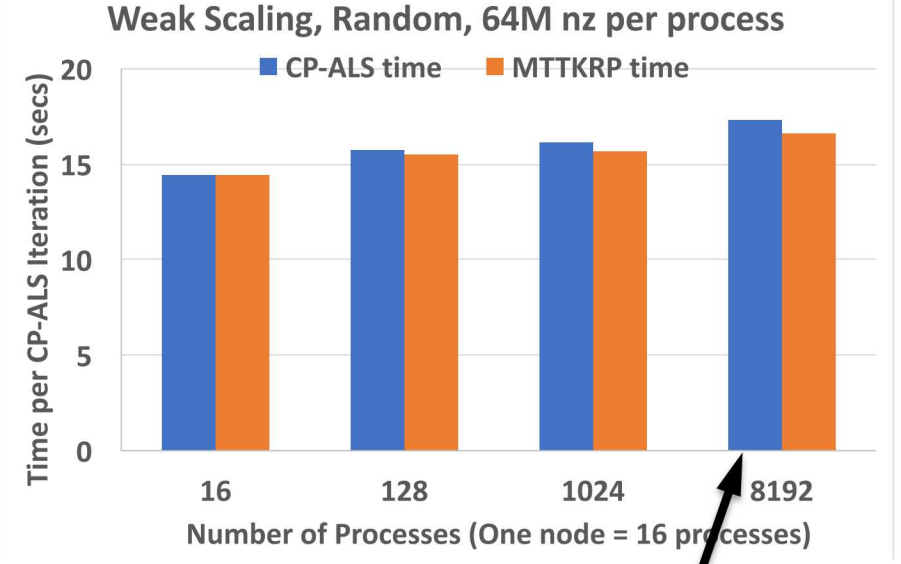
Optimization Problem: $\min \|\mathcal{X} - \mathcal{M}\|^2 \equiv \sum_i \sum_j \sum_k (x_{ijk} - m_{ijk})^2$

- Other loss functions better represent other types of data:
  - Binary data
  - Count data
  - Non-negative data

Previous work: Distributed-memory CP-ALS using Trilinos

Sandia National Laboratories

- Trilinos Solver Framework: Software for scientific applications with MPI+X parallelism
  - Linear algebra; linear, nonlinear, eigen solvers; discretization; meshing; load balancing; optimization
  - Built on Kokkos for on-node performance portability
- Tpetra parallel matrix, vector, communication classes
  - MultiVectors: dense storage, norms, etc. for factor matrices
  - Maps: parallel distribution of tensor and factor matrix
  - Import/Export: communication of factor matrix entries



***12.6 Terabyte** tensor on 8192 MPI processes (524 B nonzeros)*

MTTKRP uses Trilinos' SpMV communication pattern

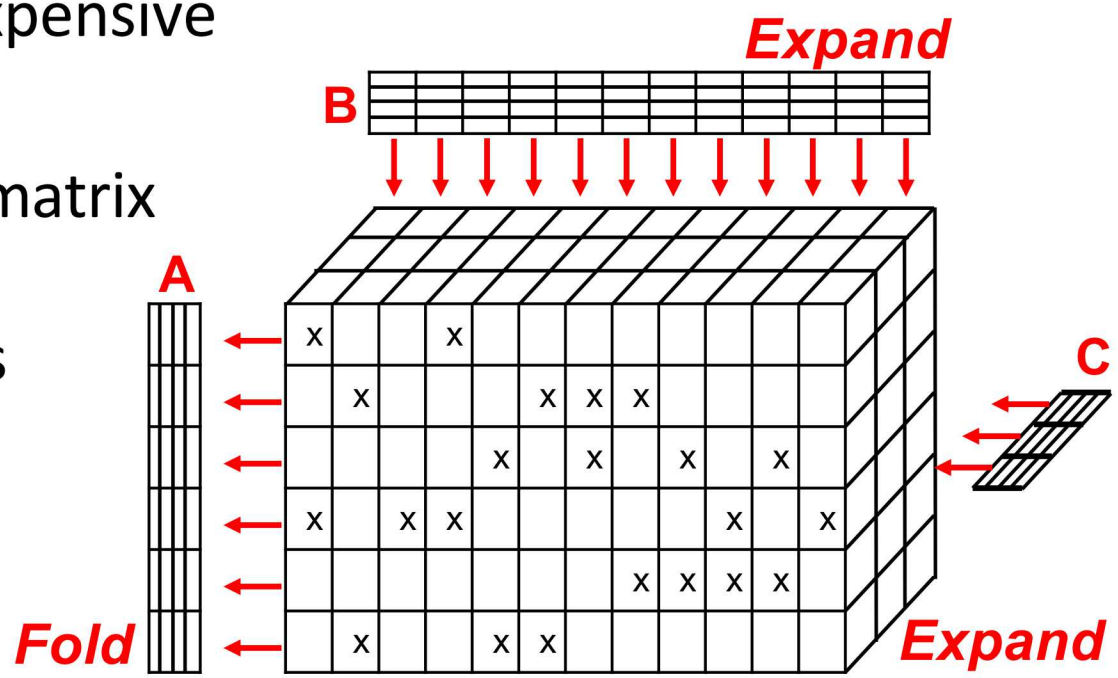
Sandia National Laboratories

$$\mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B}) \Rightarrow \sum_{ijk} x_{ijk}\, b_{jr}\, c_{kr}, r = 1, \ldots, R$$

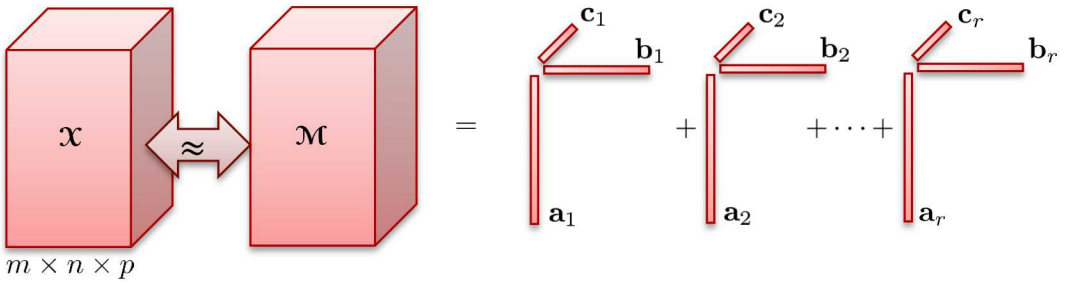
For each local tensor nonzero x_{ijk} …

…use R entries of factor matrices B and C to compute R contributions to MTTKRP

- Matricized Tensor Times Khatri-Rao Product (MTTKRP) is most expensive part of CP-ALS computation
- Expand: Sends input factor matrix entries to processors with corresponding tensor entries
- Fold: accumulates local contributions into output factor matrix



MTTKRP is key kernel of GCP direct optimization



Goal is to minimize: $F(\mathbf{A},\mathbf{B},\mathbf{C}) \equiv \sum_{ijk} f(x,m)$

Repeat until converged…

$$[\mathbf{A},\mathbf{B},\mathbf{C}] \leftarrow [\mathbf{A},\mathbf{B},\mathbf{C}] - \alpha[\frac{\partial F}{\partial \mathbf{A}}, \frac{\partial F}{\partial \mathbf{B}}, \frac{\partial F}{\partial \mathbf{C}}]$$

Gradients

$$\frac{\partial F}{\partial \mathbf{A}} = f'(\mathbf{X}_{(1)}, \mathbf{M}_{(1)})(\mathbf{C} \odot \mathbf{B})$$

$$\frac{\partial F}{\partial \mathbf{B}} = f'(\mathbf{X}_{(2)}, \mathbf{M}_{(2)})(\mathbf{C} \odot \mathbf{A})$$

$$\frac{\partial F}{\partial \mathbf{C}} = f'(\mathbf{X}_{(3)}, \mathbf{M}_{(3)})(\mathbf{B} \odot \mathbf{A})$$

Gradient tensor f' is the derivative of loss function f with respect to m, evaluated elementwise

MTTKRP using gradient tensor

Solve GCP Optimization via Stochastic Gradient Descent

Sandia National Laboratories

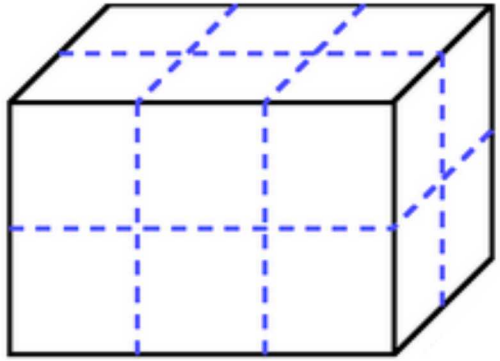
- Sample ***both*** tensor nonzeros and tensor zeros
  - Stratified sampling (sampling without replacement)
    Ensure an entry (nonzero or zero) is sampled at most once per iteration
  - Semi-stratified sampling (sampling with replacement)
    Allow duplicate samples of an entry
- Sampling to compute loss for convergence test
  - Sample <10% of tensor entries

$$\hat{F} \equiv \sum_{x_{ijk} \neq 0} \frac{\eta}{p} f\left(x_{ijk}, m_{ijk}\right) + \sum_{x_{ijk}=0} \frac{\zeta}{q} f\left(x_{ijk}, m_{ijk}\right)$$

  - η/p = number of nonzeros in tensor / number of nonzero samples
  - ζ/q = number of zeros in tensor / number of zero samples
- Sampling to compute stochastic gradients
  - Sample <1% of tensor entries in each iteration

Processor bounding boxes simplify sampling in distributed memory

Sandia National Laboratories



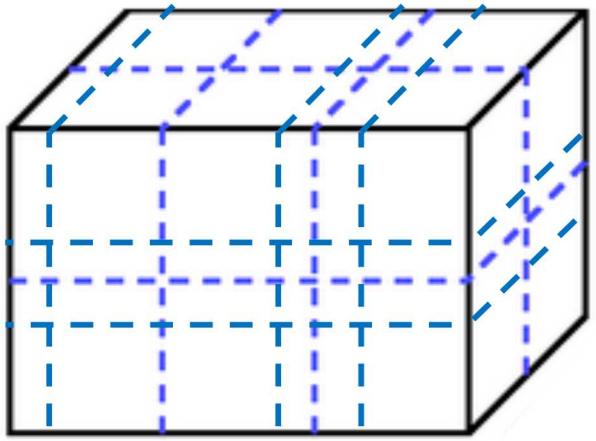
p x q x r = 3 x 2 x 2 processor layout

- Assume block-based distribution of tensor
  - Bounding boxes do not overlap
  - Bounding boxes cover entire index space of tensor
  - Each processor knows bounding box of its block
- Processors sample within their bounding boxes
  - Enables efficient searching for duplicate samples in stratified sampling
  - Maintains aligned communication pattern for expand/fold
- For *s* samples globally, select *s/np* indices on each of *np* processors (for load balance)
- Sample nonzeros proportionally to number of nonzeros in box (to prevent skew)
- Note: CP-ALS does not require bounding boxes; can use arbitrary tensor distributions

Uniform boxes prevent skew in sampling

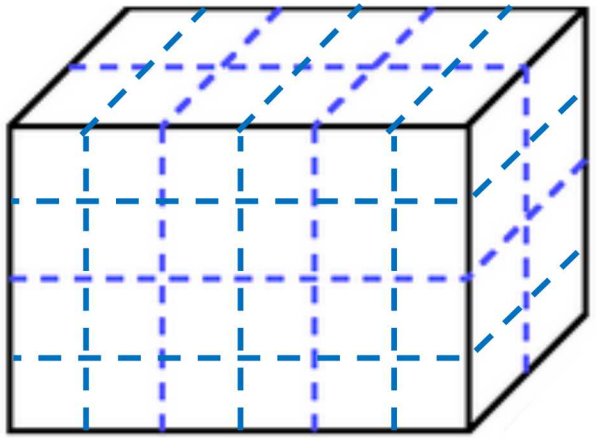
Sandia National Laboratories

- Medium grain distribution:
  - Each box has equal number of nonzeros, unequal number of zeros – great for CP-ALS
  - For uniform sampling, sample equal number of nonzeros in each box
  - For load balance, sample equal number of zeros in each box
  - Can over/under sample zeros in each box



Tensor distributions with p x q x r = 6 x 4 x 2 processor layout

- Uniform boxes:
  - Equal number of indices (nonzero + zero) per box
  - Within a box, sample proportionally to number of nonzeros and zeros in the box



Error computation is adapted to bounding boxes

Sandia National Laboratories

$$\hat{F} \equiv \sum_{x_{ijk} \neq 0} \frac{\eta}{p} f\left(x_{ijk}, m_{ijk}\right) + \sum_{x_{ijk}=0} \frac{\zeta}{q} f\left(x_{ijk}, m_{ijk}\right)$$

- η/p = # of nonzeros in tensor / # of nonzero samples
- ζ/q = # of zeros in tensor / # of zero samples

- Replace η, ζ, p, q with on-processor values from bounding boxes
  - Local number of nonzeros, zeros, samples
- If zeros and nonzeros are perfectly balanced across processors, ratios don't change from global values
- If imbalanced, account for density of sampling within a processor

Solve GCP Optimization via Stochastic Gradient Descent (ADAM)

Sandia National Laboratories

"ADAM: a method for stochastic optimization," D. Kingma & J. Ba, ICLR 2015

STOCHASTIC GRADIENT DESCENT

Randomly initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$

Estimate loss $\hat{F} = \sum_{\{ijk\}} f(x_{ijk}, m_{ijk})$

While not converged

$F_{old} = \hat{F}$

For τ iterations

Compute stochastic gradients $\mathbf{G}_A = \frac{\partial F}{\partial A}$; $\mathbf{G}_B$; $\mathbf{G}_C$

Compute $\mathbf{A}, \mathbf{B}, \mathbf{C}$ using $\mathbf{G}_A, \mathbf{G}_B, \mathbf{G}_C$ and step size

Estimate loss function $\hat{F} = \sum_{\{ijk\}} f(x_{ijk}, m_{ijk})$

If $\hat{F} > Fold$, backup and reduce step size

Return $\mathbf{A}, \mathbf{B}, \mathbf{C}$

Solve GCP Optimization via Stochastic Gradient Descent (ADAM)

Sandia National Laboratories

STOCHASTIC GRADIENT DESCENT

Randomly initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$

Estimate loss $\hat{F} = \sum_{\{ijk\}} f(x_{ijk}, m_{ijk})$

While not converged

$F_{old} = \hat{F}$

For τ iterations

Compute stochastic gradients $\mathbf{G}_A = \frac{\partial F}{\partial A}$; $\mathbf{G}_B$; $\mathbf{G}_C$

Compute $\mathbf{A}, \mathbf{B}, \mathbf{C}$ using $\mathbf{G}_A, \mathbf{G}_B, \mathbf{G}_C$ and step size

Estimate loss function $\hat{F} = \sum_{\{ijk\}} f(x_{ijk}, m_{ijk})$

If $\hat{F} > Fold$, backup and reduce step size

Return $\mathbf{A}, \mathbf{B}, \mathbf{C}$

Use fixed stratified sampling to estimate loss (<10% of tensor indices); construct sampled tensor only once

Solve GCP Optimization via Stochastic Gradient Descent (ADAM)

Sandia National Laboratories

STOCHASTIC GRADIENT DESCENT

Randomly initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$

Estimate loss $\hat{F} = \sum_{\{ijk\}} f(x_{ijk}, m_{ijk})$

While not converged

$F_{old} = \hat{F}$

For τ iterations

Compute stochastic gradients $\mathbf{G}_A = \frac{\partial F}{\partial A}$; $\mathbf{G}_B$; $\mathbf{G}_C$

Compute $\mathbf{A}, \mathbf{B}, \mathbf{C}$ using $\mathbf{G}_A, \mathbf{G}_B, \mathbf{G}_C$ and step size

Estimate loss function $\hat{F} = \sum_{\{ijk\}} f(x_{ijk}, m_{ijk})$

If $\hat{F} > Fold$, backup and reduce step size

Return $\mathbf{A}, \mathbf{B}, \mathbf{C}$

COMPUTE STOCHASTIC GRADIENTS

Sample nonzero and zero indices $\{ijk\}$

Construct gradient tensor $\boldsymbol{\mathcal{Y}}$ with

$y_{ijk} = \frac{\partial f}{\partial m}(x_{ijk}, m_{ijk})$ for samples $\{ijk\}$

$$\mathbf{G}_A = \frac{\partial F}{\partial A} = \text{MTTKRP}(\boldsymbol{\mathcal{Y}}, \boldsymbol{B}, \boldsymbol{C})$$

$$\mathbf{G}_B = \frac{\partial F}{\partial B} = \text{MTTKRP}(\boldsymbol{\mathcal{Y}}, \boldsymbol{C}, \boldsymbol{A})$$

$$\mathbf{G}_C = \frac{\partial F}{\partial C} = \text{MTTKRP}(\boldsymbol{\mathcal{Y}}, \boldsymbol{A}, \boldsymbol{B})$$

Solve GCP Optimization via Stochastic Gradient Descent (ADAM)

Sandia National Laboratories

Use stratified or semi-stratified sampling (<1% of tensor indices)

STOCHASTIC GRADIENT DESCENT

Randomly initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$

Estimate loss $\hat{F} = \sum_{\{ijk\}} f(x_{ijk}, m_{ijk})$

While not converged

$F_{old} = \hat{F}$

For τ iterations

Compute stochastic gradients $\mathbf{G}_A = \frac{\partial F}{\partial A}$; $\mathbf{G}_B$; $\mathbf{G}_C$

Compute $\mathbf{A}, \mathbf{B}, \mathbf{C}$ using $\mathbf{G}_A, \mathbf{G}_B, \mathbf{G}_C$ and step size

Estimate loss function $\hat{F} = \sum_{\{ijk\}} f(x_{ijk}, m_{ijk})$

If $\hat{F} > Fold$, backup and reduce step size

Return $\mathbf{A}, \mathbf{B}, \mathbf{C}$

COMPUTE STOCHASTIC GRADIENTS

Sample nonzero and zero indices $\{ijk\}$

Construct gradient tensor $\mathcal{Y}$ with

$$y_{ijk} = \frac{\partial f}{\partial m}(x_{ijk}, m_{ijk}) \text{ for samples } \{ijk\}$$

$$\mathbf{G}_A = \frac{\partial F}{\partial A} = \text{MTTKRP}(\mathcal{Y}, \boldsymbol{B}, \boldsymbol{C})$$

$$\mathbf{G}_B = \frac{\partial F}{\partial B} = \text{MTTKRP}(\mathcal{Y}, \boldsymbol{C}, \boldsymbol{A})$$

$$\mathbf{G}_C = \frac{\partial F}{\partial C} = \text{MTTKRP}(\mathcal{Y}, \boldsymbol{A}, \boldsymbol{B})$$

Distributed memory GCP Optimization via Stochastic Gradient Descent

Sandia National Laboratories

Sample within bounding boxes;
no communication needed

Communicate (expand)
A, B, C entries corresponding
to samples in $\mathcal{Y}$

Communicate (fold) in MTTKRP

COMPUTE STOCHASTIC GRADIENTS

Sample nonzero and zero indices $\{ijk\}$

Construct gradient tensor $\boldsymbol{\mathcal{Y}}$ with

$$y_{ijk} = \frac{\partial f}{\partial m}(x_{ijk}, m_{ijk}) \text{ for samples } \{ijk\}$$

$$\mathbf{G}_A = \frac{\partial F}{\partial A} = \text{MTTKRP}(\boldsymbol{\mathcal{Y}}, \boldsymbol{B}, \boldsymbol{C})$$

$$\mathbf{G}_B = \frac{\partial F}{\partial B} = \text{MTTKRP}(\boldsymbol{\mathcal{Y}}, \boldsymbol{C}, \boldsymbol{A})$$

$$\mathbf{G}_C = \frac{\partial F}{\partial C} = \text{MTTKRP}(\boldsymbol{\mathcal{Y}}, \boldsymbol{A}, \boldsymbol{B})$$

Majority of communication is in computing stochastic gradients

Distributed memory GCP Optimization via Stochastic Gradient Descent

Sandia National Laboratories

STOCHASTIC GRADIENT DESCENT

Randomly initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$

Estimate loss $\hat{F} = \sum_{\{ijk\}} f(x_{ijk}, m_{ijk})$

While not converged

$F_{old} = \hat{F}$

For τ iterations

Compute stochastic gradients $\mathbf{G}_A = \frac{\partial F}{\partial A}$; $\mathbf{G}_B$; $\mathbf{G}_C$

Compute $\mathbf{A}, \mathbf{B}, \mathbf{C}$ using $\mathbf{G}_A, \mathbf{G}_B, \mathbf{G}_C$ and step size

Estimate loss function $\hat{F} = \sum_{\{ijk\}} f(x_{ijk}, m_{ijk})$

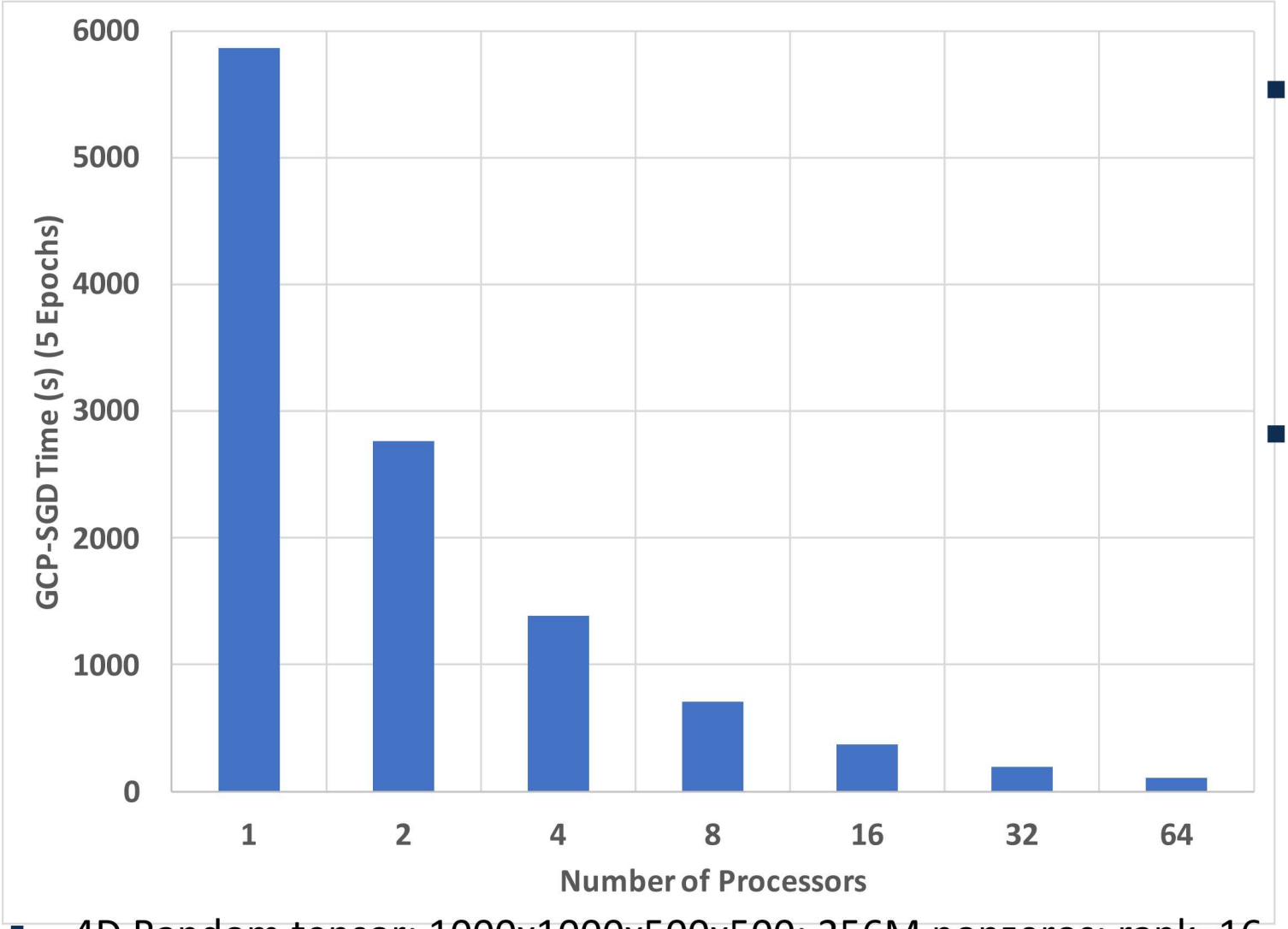
If $\hat{F} > Fold$, backup and reduce step size

Return $\mathbf{A}, \mathbf{B}, \mathbf{C}$

Communicate (expand) A, B, C for fixed samples; Allreduce

Distribute G_A, G_B, G_C in same way as A, B, C; *no communication needed here*

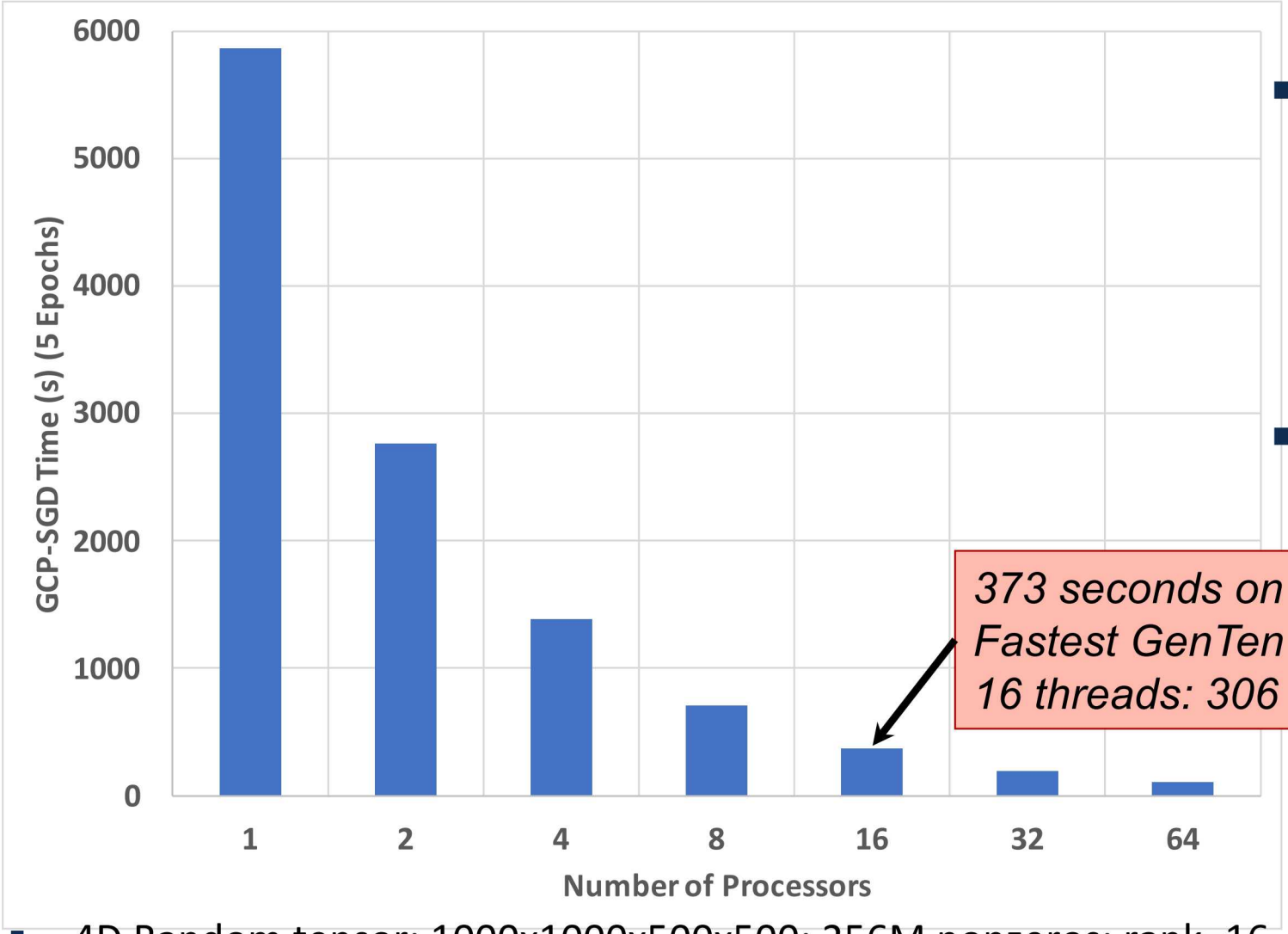
Best-case strong scaling is good



- Best case: perfect balance, alignment; small factor matrices
- 52.7x speedup on 64 processors

- 4D Random tensor: 1000x1000x500x500; 256M nonzeros; rank=16
- 5M samples in Fixed tensor; 1.5M samples in Stoc Grad tensor
- SkyBridge cluster (2.6 GHz Intel Sandy Bridge with Infiniband)

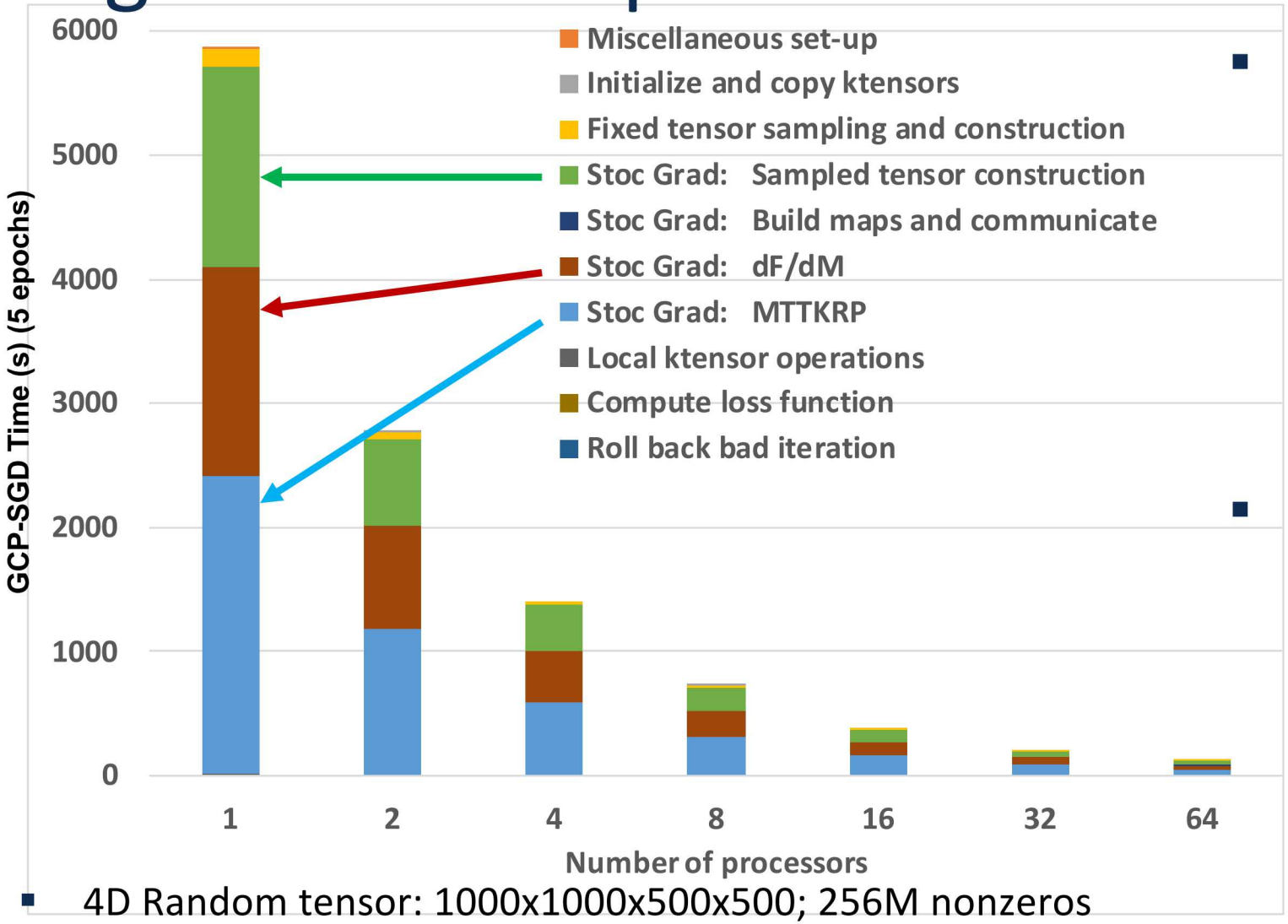
Best-case strong scaling is good



- Best case: perfect balance, alignment; small factor matrices
- 52.7x speedup on 64 processors

- 4D Random tensor: 1000x1000x500x500; 256M nonzeros; rank=16
- 5M samples in Fixed tensor; 1.5M samples in Stoc Grad tensor
- SkyBridge cluster (2.6 GHz Intel Sandy Bridge with Infiniband)

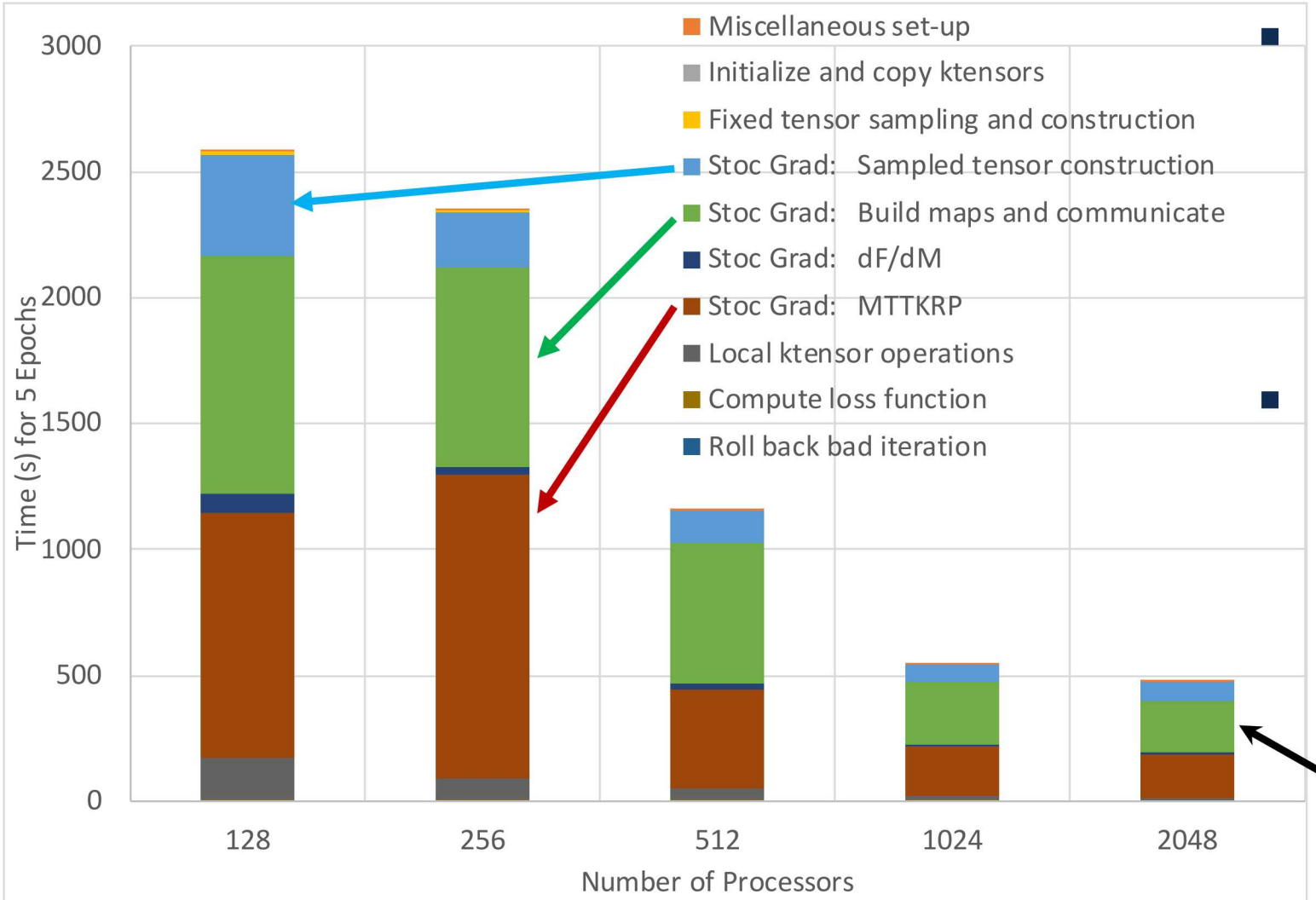
Most time spent in stochastic gradient computation



- Many kernels scaling well
  - Sampled tensor construction
  - MTTKRP
  - Derivative computation
- Costs for building maps and communicating are small but grow with number of processors

- 4D Random tensor: 1000x1000x500x500; 256M nonzeros
- 5M samples in Fixed tensor; 1.5M samples in Stoc Grad tensor
- SkyBridge cluster (2.6 GHz Intel Sandy Bridge with Infiniband)

More variability seen in real tensors

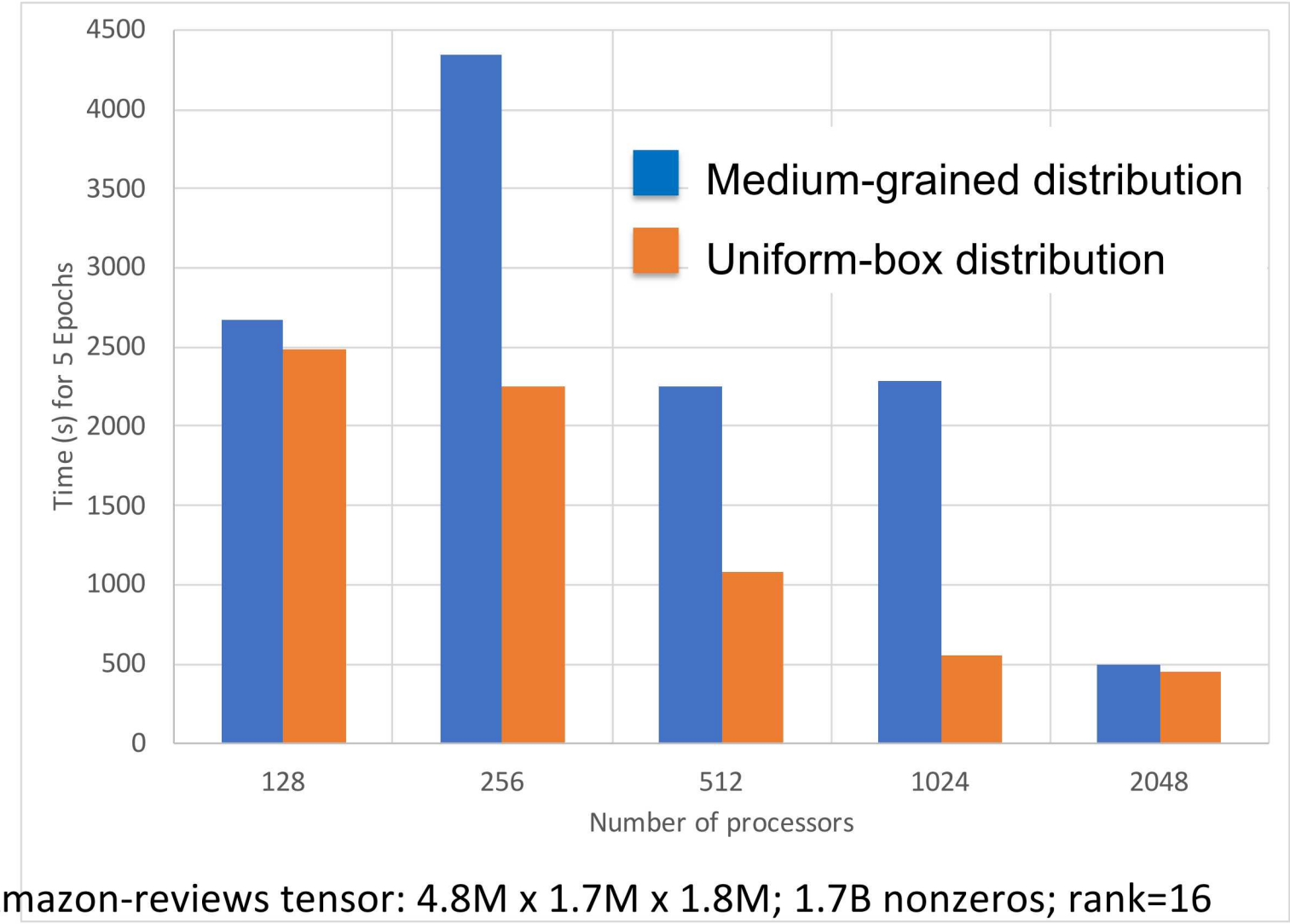


- Benefit from more processors; tensor too large for single shared-memory node
- Building maps, communicating are more expensive

With 2048 procs, ~4K samples/proc

- 3D Amazon-reviews tensor: 4.8M x 1.7M x 1.8M; 1.7B nonzeros; rank=16
- 83M samples in Fixed tensor; 8.3M samples in Stoc Grad tensor
- SkyBridge cluster (2.6 GHz Intel Sandy Bridge with Infiniband)

Uniform-box distribution benefits performance



- 3D Amazon-reviews tensor: 4.8M x 1.7M x 1.8M; 1.7B nonzeros; rank=16
- 83M samples in Fixed tensor; 8.3M samples in Stoc Grad tensor
- SkyBridge cluster (2.6 GHz Intel Sandy Bridge with Infiniband)

Conclusions and Future Work

Sandia National Laboratories

- Distributed memory implementation of GCP-SGD is scalable and feasible for decomposition of very large tensors
  - Bounding box of processor's indices aids in efficient distributed-memory sampling
  - Trilinos' Tpetra data structures manage communication and factor matrix operations
  - Small sample sizes challenge strong scaling
- Next step: integration of on-node parallelism
  - Use capabilities in GenTen (Phipps, Kolda), such as on-node MTTKRP
- MPI-based code soon to be released on gitlab.com: GentenMPI