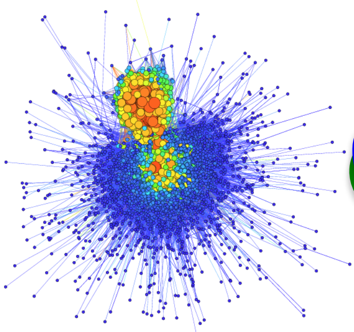
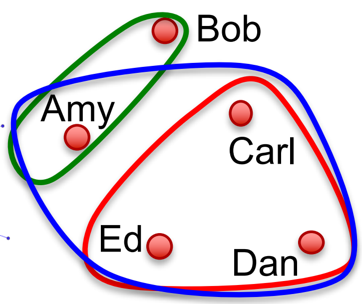
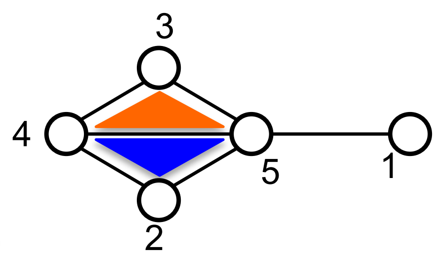
Exceptional service in the national interest

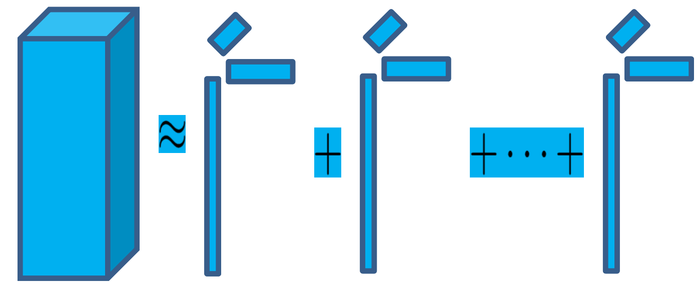
Sandia National Laboratories

CCR Center for Computing Research









High-Performance Portable Data Analytics Software Using the Kokkos Ecosystem

kokkos

Michael Wolf

CLSAC 2018 (October 30, 2018)

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is 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-NA0003525.

SAND2018--12353 C

Acknowledgements

- Kokkos Ecosystem
  - Slides from Christian Trott, Siva Rajamanickam
- Grafiki (previously TriData)
  - Danny Dunlavy, Rich Lehoucq, Jon Berry, Nathan Ellingwood, Daniel Bourgeois (Rice)
- Tensor Work (slides from Dunlavy's TRICAP talk)
  - Data Analytics: Danny Dunlavy, Keita Teranishi, Rich Lehoucq, Richard Barrett, Tammy Kolda, Chris Forster (NVIDIA), Karen Devine
  - Related Work: Eric Phipps
- Kokkos Kernels for Triangle-Based Analytics
  - Siva Rajamanickam, Mehmet Deveci (Google), Jon Berry, Abdurrahman Yassar (GT), Umit Catalyurek (GT)

Performance Portability Motivation

Intel Multicore

NVIDIA GPU

IBM Power

Intel Manycore

AMD Multicore/APU

ARM

- **Several many/multi-core architectures central to DOE HPC**
- **Applications struggle to obtain good performance on all of these**

Performance Portability Motivation

- Example: Architecture Change NVIDIA Pascal to Volta
  - Warps can arbitrarily, permanently diverge, and branches can now interleave
  - **2 man months** to fix in Kokkos for just **3 code positions**
  - Without abstraction layer: **~400 places** in SNL's Trilinos framework (excluding Kokkos) would need fixes
- Timeline for Architectures
  - In Bold: requires new approach for performance for the first time

2012
IBM BGQ (Sequoia, Mira)
NVIDIA Kepler (Titan)

2016
Intel KNL (Trinity, Cori)

2018
NVIDIA Volta (Summit, Sierra)
ARM (Astra)

2021
Intel A21?
AMD GPU?
NVIDIA GPU?

Decade of DOE HPC will have seen 4-5 "new" paradigms!

Performance Portability or Bust?

- Optimistic estimate: 10% of application needs to get rewritten for adoption of Shared Memory Parallel Programming Model
- Typical Apps: 300k – 600k Lines
  - Uintah: 500k, QMCPack: 400k, LAMMPS: 600k; QuantumEspresso: 400k
  - Typical App Port thus 2-3 Man-Years
  - Sandia maintains a couple dozen of those
- Large Scientific Libraries
  - E3SM: 1,000k Lines x 10% => 5 Man-Years
  - Trilinos: 4,000k Lines x 10% => 20 Man-Years

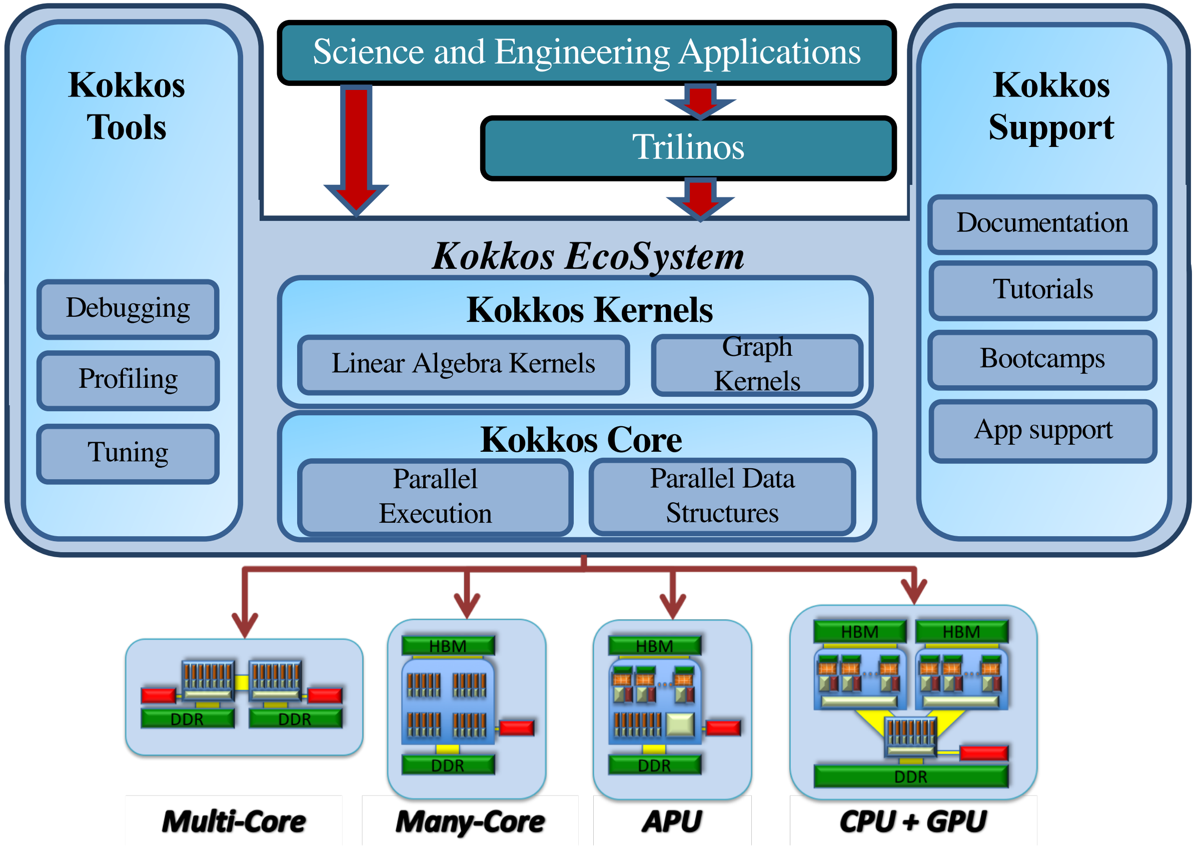
10 LOC / hour ~ 20k LOC / year

Sandia alone: 50-80 Man Years

Convincing applications to adopt even one MPI+X programming model challenging

Kokkos Ecosystem for Performance Portability

Sandia National Laboratories



Kokkos Core: parallel patterns and data structures, supports several execution and memory spaces

Kokkos Kernels: performance portable BLAS, sparse, and graph algorithms and kernels

Kokkos Tools: debugging and profiling support

Kokkos Ecosystem addresses complexity of supporting numerous many/multi-core architectures that are central to DOE HPC enterprise

Why Kokkos?

- **Support multiple back-ends**
  - OpenMP, CUDA, Qthreads, Pthread, …
  - Work closely with hardware vendors
- **Support multiple data layout options**
  - Column vs Row Major; Device/CPU memory
- **Support different parallelism**
  - Nested loop support; vector, threads, warps, etc.
  - Task parallelism
- **Growing Kokkos Support**
  - Community: ORNL, LANL, CSCS, Juelich, Slack Channel (80+ members)
  - Kokkos abstractions migrating to C++ standard

Kokkos team eager to engage with new customers to support new applications and architectures

DOE ECP Kokkos Users

We don't actually know who all is using Kokkos. Partial ECP List:

Application	State
SNL ATDM Apps	Base (SPARC, EMPIRE, Nimble,…)
LANL ATDM Apps	In Parts
EXAALT	Base Code
QMCPack	Evaluation
ExaWind	Base Code
ExaAM	Experimenting
LatticeQCD	Experimenting
ProxyApp	Base Code (in parts)
COPA	Base Code
ExaGraph	Base Code (in parts)
ExaLearn	Committed (in parts)

Software Technology	State
SNL ATDM PMR	This is Kokkos ;-)
LANL ATDM PMR	Experimenting
KokkosSupport	
SNL ATDM DevTools	Base Code (in parts)
ExaPapi	Integrates KokkosTools
SNL ATDM Math	Base Code
ForTrilinos	Base Code
PEEKS	Base Code

Additionally:

- Many ASC applications at Sandia are porting or using Kokkos in their base code
- Many applications leverage Kokkos through Trilinos framework's solvers

Kokkos has a growing DOE user base

Kokkos and Greater HPC Community

- Many Institutions outside of DOE started experimenting with Kokkos or have projects that are already committed
- Additional institutions leveraging Kokkos indirectly via Trilinos solvers

What is Kokkos?

- **Templated C++ Library**
  - Goal: Write algorithms once, run everywhere (almost) optimally
  - Serve as substrate layer of sparse matrix and vector kernels
- **Kokkos::View() accommodates performance-aware multidimensional array data objects**
  - Light-weight C++ class
- **Parallelizing loops using C++ language standard**
  - Lambda, Functors
- **Extensive support of atomics**
- **Substantial DOE investment**
  - **ECP/ATDM software technology (Ecosystem ~$3M/year)**
  - **Many DOE ECP and ASC applications use Kokkos**

Parallel Loops with Kokkos

Serial

```
for (size_t i = 0; i < N; ++i)
{
  /* loop body */
}
```

OpenMP

```
#pragma omp parallel for
for (size_t i = 0; i < N; ++i)
{
  /* loop body */
}
```

Kokkos

```
parallel_for (( N, [=], (const size_t i)
{
  /* loop body */
});
```

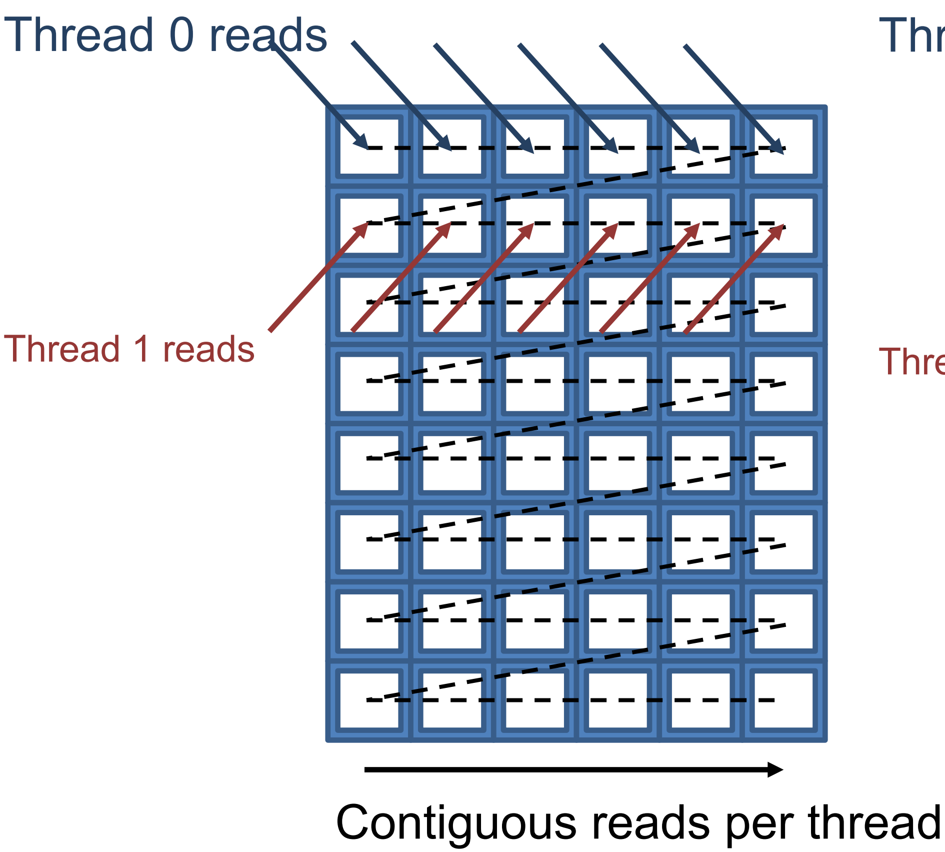
- Provide parallel loop operations using C++ language features
- Conceptually, the usage is no more difficult than OpenMP. The annotations just go in different places.

Array Access with Kokkos

```
Kokkos::View<double **, Layout, Space>
```

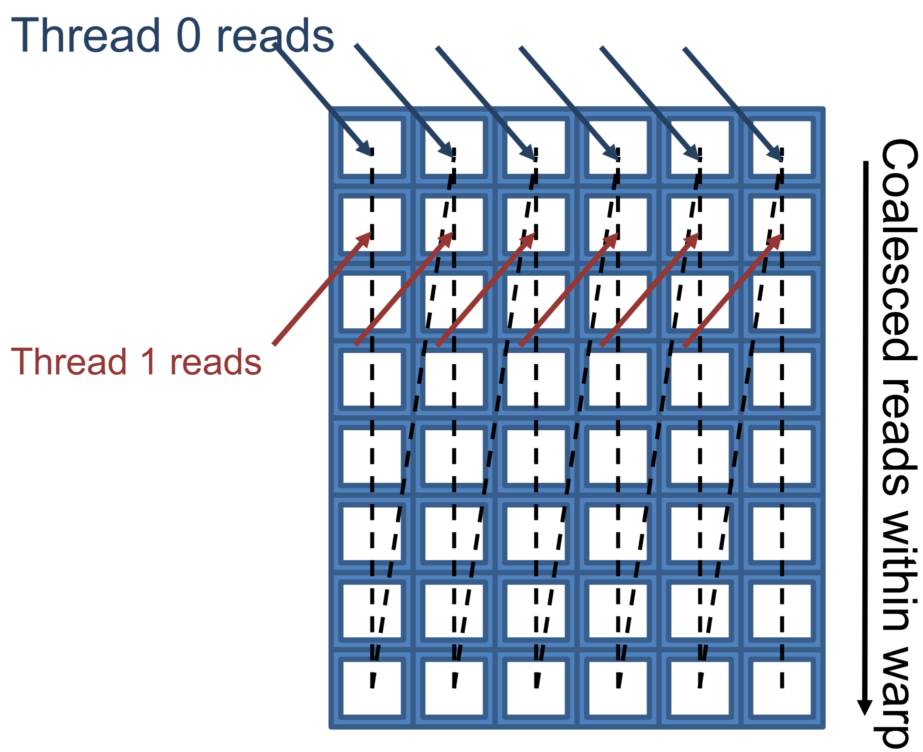
```
View<double **, Right, Host>
```

Row-major

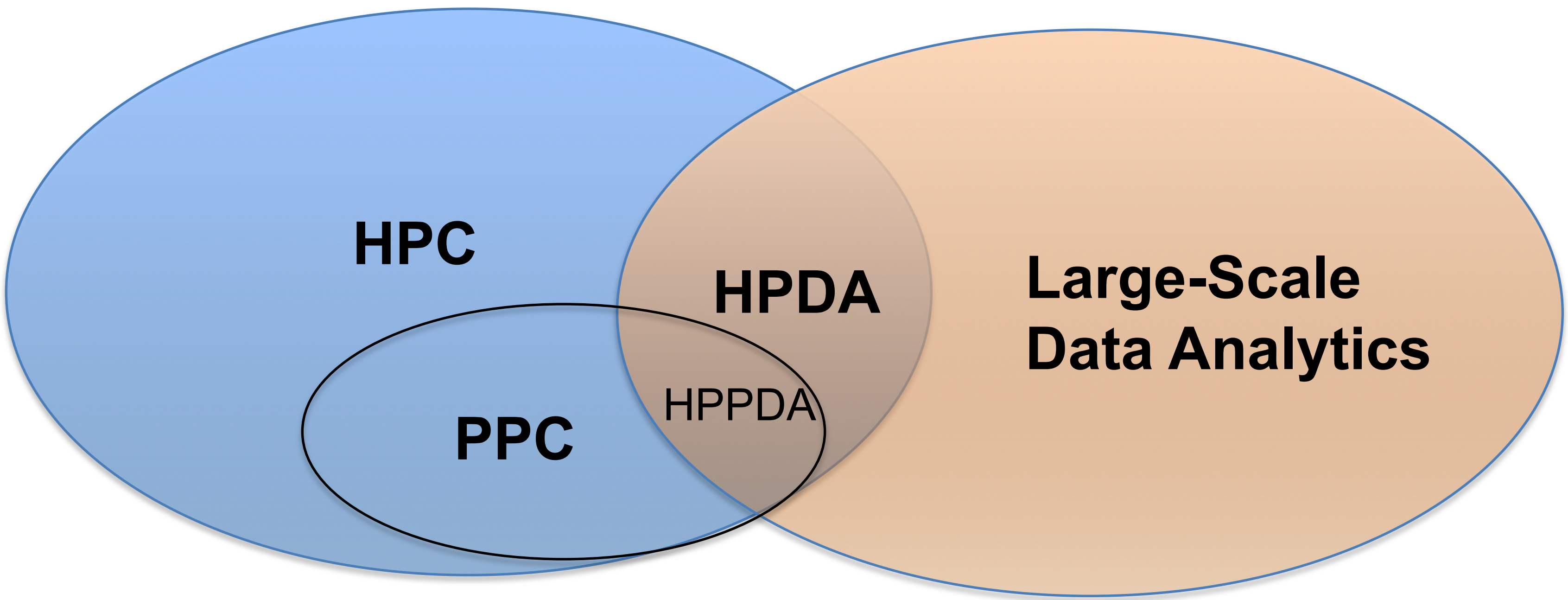



```
View<double **, Left, CUDA>
```

Column-major



HPPDA and Kokkos



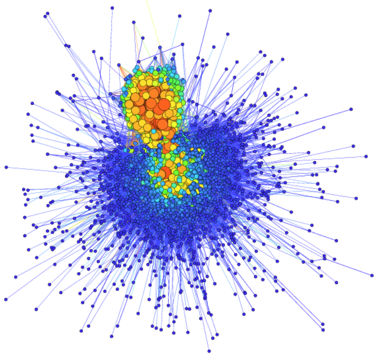
- **Performance-Portable Computing (PPC) (*Sandia: Kokkos*)**
- **High-Performance Data Analytics (HPDA)**
  - Use HPC to compute big data analytics faster
- **High-Performance-Portable Data Analytics (HPPDA)**
  - Use PPC to enable HPDA on DOE platforms (***Sandia: Kokkos***)

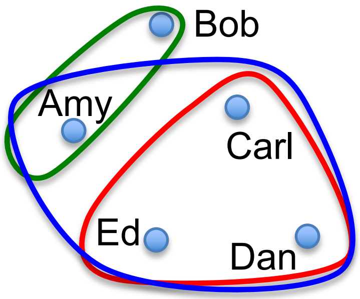
Leverage significant DOE investment in performance portability computing to impact large-scale data analytics

Use Case 1: Grafiki

- Formerly TriData – Trilinos for Large-Scale Data Analysis
  - Leverages Trilinos Framework (Sandia National Labs)
    - High performance linear algebra, traditional focus on CSE
    - High performing eigensolvers, linear solvers
    - Scales to billions of matrix rows/vertices
- Grafiki Vision: Sparse Linear Algebra-Based Data Analysis
  - Apply sparse linear algebra techniques to data analysis
  - Target: very large data problems
  - Target: distributed memory and single node HPC architectures
- Additionally
  - Vehicle for improving how Trilinos can be leveraged for data analytics (e.g., submatrix extraction, preconditioning, load-balancing)
  - Support GraphBLAS-like linear algebra analysis techniques
- Focus: Graph and Hypergraph Analysis

Grafiki Capabilities

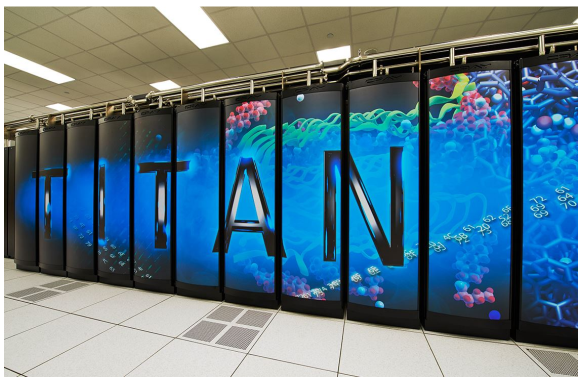




- Eigen solver based capabilities
  - Spectral Clustering, Vertex/Edge eigencentrality (graphs, hypergraphs)
  - Supports several eigensolvers (through Trilinos): LOBPCG, TraceMin-Davidson, Riemannian Trust Region, Block Krylov-Schur
- Linear solver based capability
  - Mean hitting time analysis on graphs
  - Support for different linear solvers (typically use CG) and preconditioners
- Other
  - K-means++, metrics (conductance, modularity, jaccard index)
  - Random graph and hypergraph models, hypothesis testing techniques/infrastructure for evaluation of clustering software

Grafiki Approach

Grafiki



Distributed Memory (DM)

- Clusters, supercomputers
- Tpetra (MPI, DM data structures)
- **Kokkos** (node level parallelism)

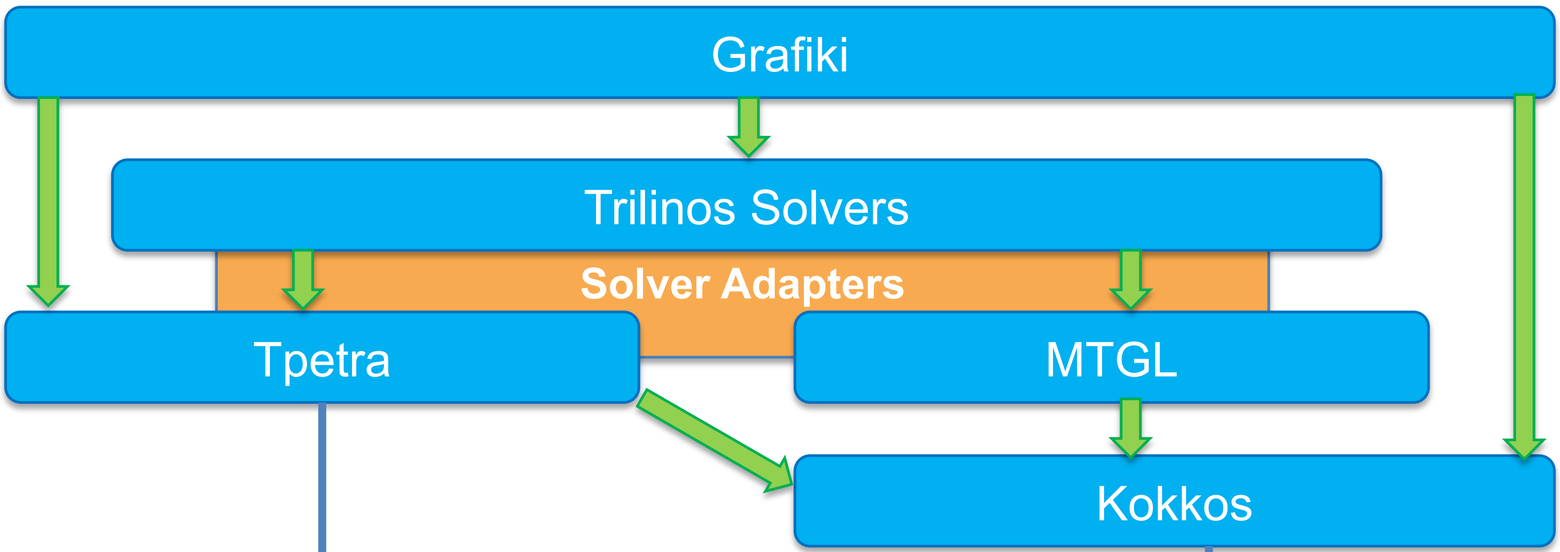
Workstation

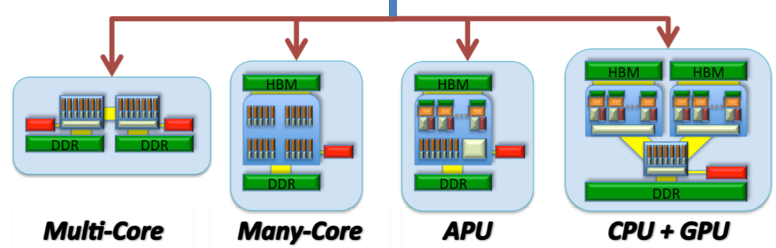
- CPUs, GPUs, KNLs, …
- **Kokkos**
- MTGL

Goal: Write algorithms once, run on both types of architectures

MTGL=MultiThreaded Graph Library

Grafiki Software Stack





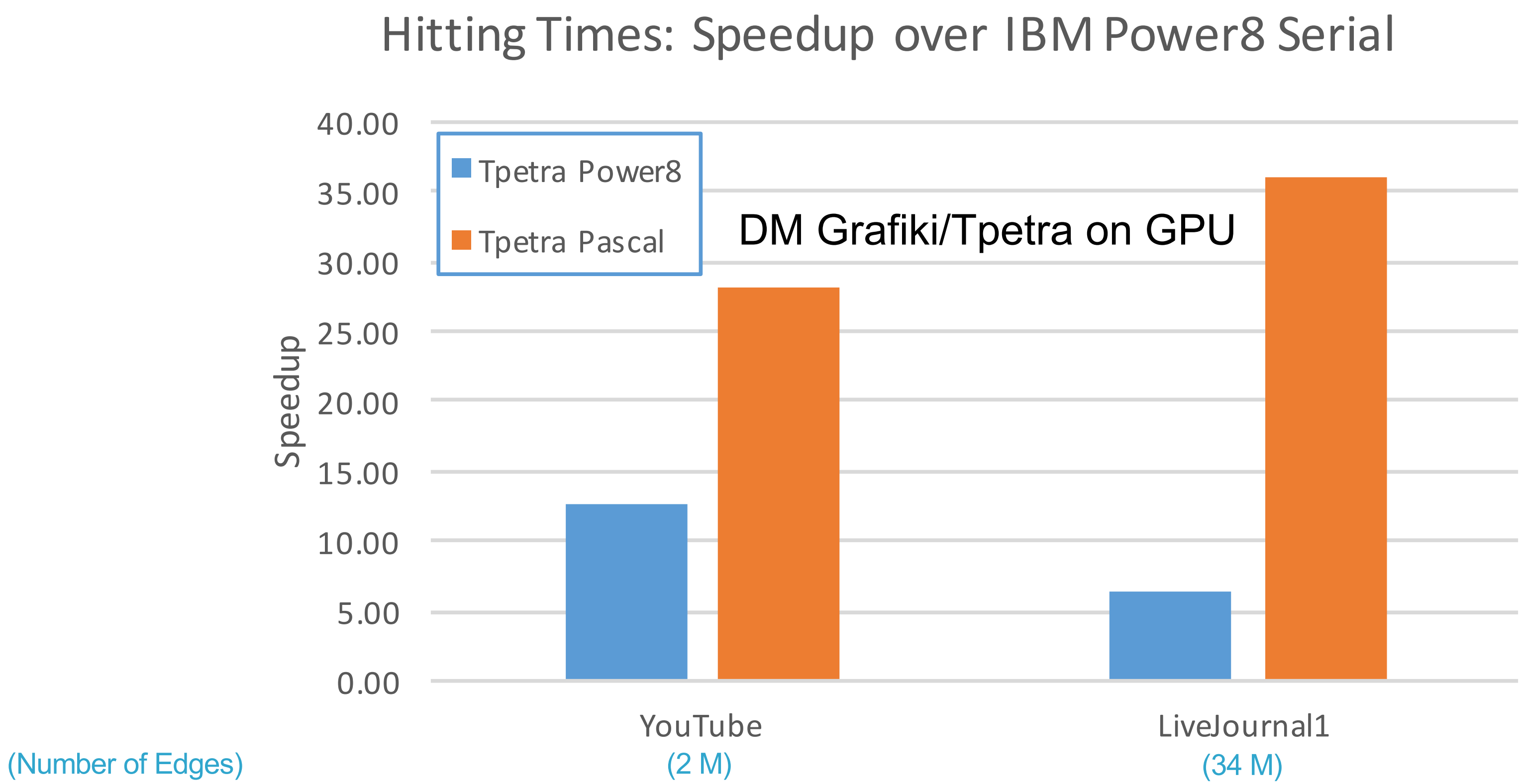
Distributed memory computations

Portable on-node performance

Flexible solver adapters enable solution for both architectures

Mean Hitting Time Results

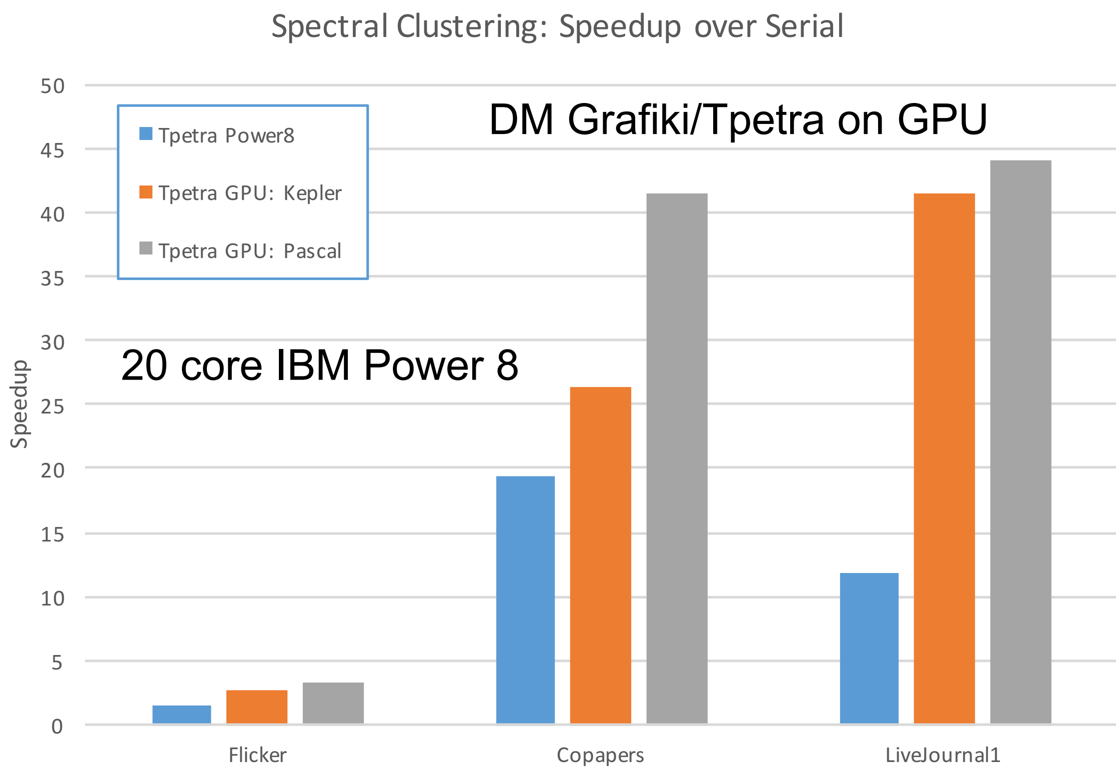
MHT: Linear solver based analytic



- **Solver/Kokkos stack allows analytic to be written in architecture agnostic manner**
- **GPU computation is up to 35x speedup over host serial**

Grafiki Spectral Clustering Results

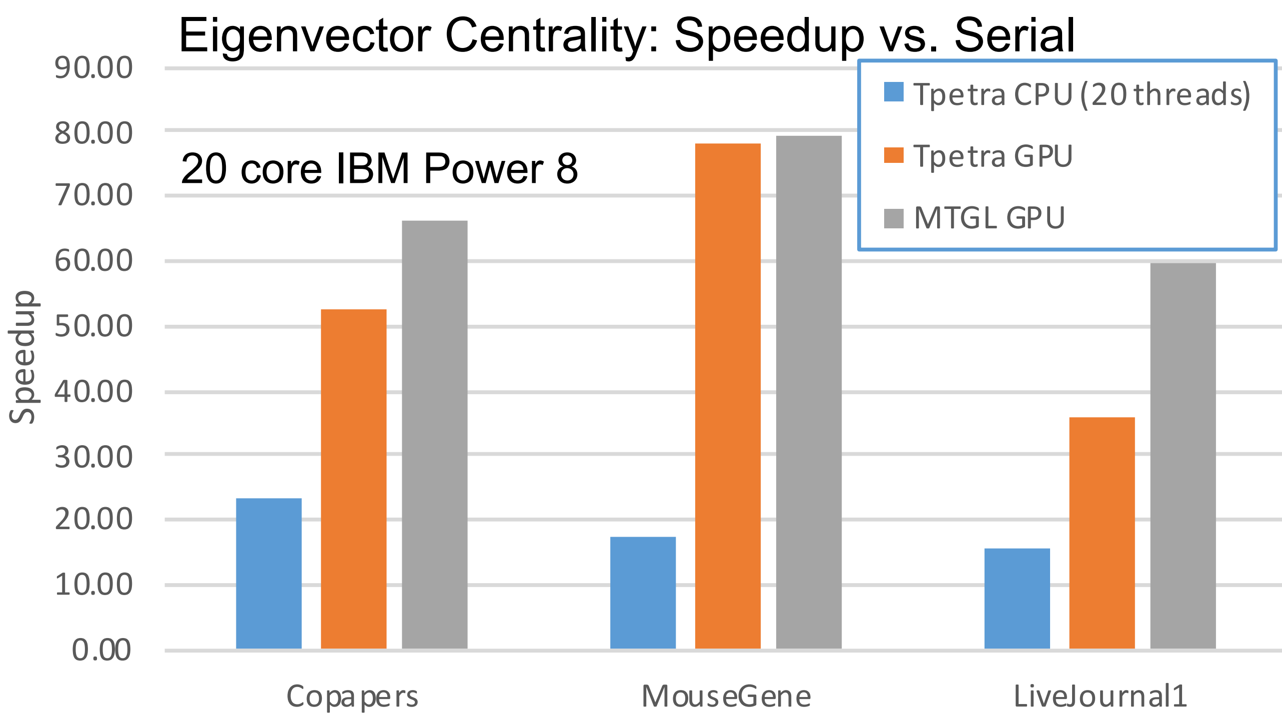
Spectral clustering: Eigensolver based analytic



- **Solver/Kokkos stack allows analytic to be written in architecture agnostic manner**
- **GPU computation up to 45x speedup over host serial**

Grafiki Centrality Results: Tpetra and MTGL

EV centrality: Eigensolver based analytic



(Number of Edges) (15 M) (14 M) (43 M)

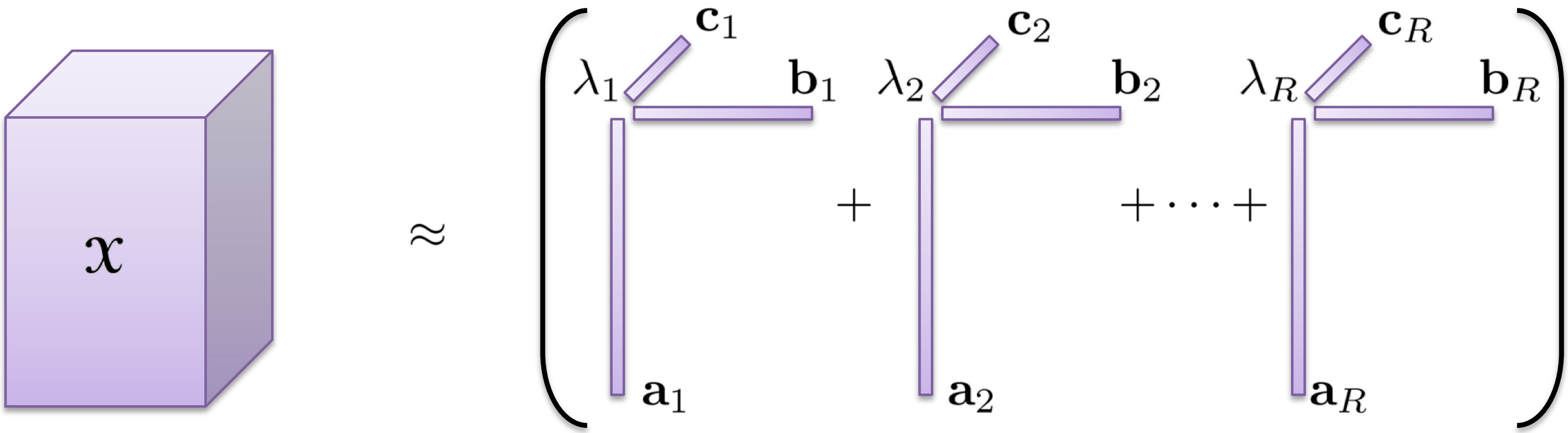
- **Solver/Kokkos stack allows analytic to be written in architecture agnostic manner**
- **GPU computation is up to 80x speedup over host serial**

Use Case 2: Scalable Tensor Factorizations

- **Motivation: Count Data**
  - Network analysis
  - Term-document analysis
  - Email analysis
  - Link prediction
  - Web page analysis
- **Large, Sparse Data**
  - Number of dimensions = 4, 5, 6, …
  - Example tensor size: 10^4 x 10^4 x 10^6 x 10^6 x 10^7
  - Example densities: 10^{-8} to 10^{-16}
- Targeting several multi/many-core architectures
  - Intel CPU, Intel MIC, NVIDIA GPU, IBM Power 9, etc.

CP Tensor Decomposition

CANDECOMP/PARAFAC (CP) Model



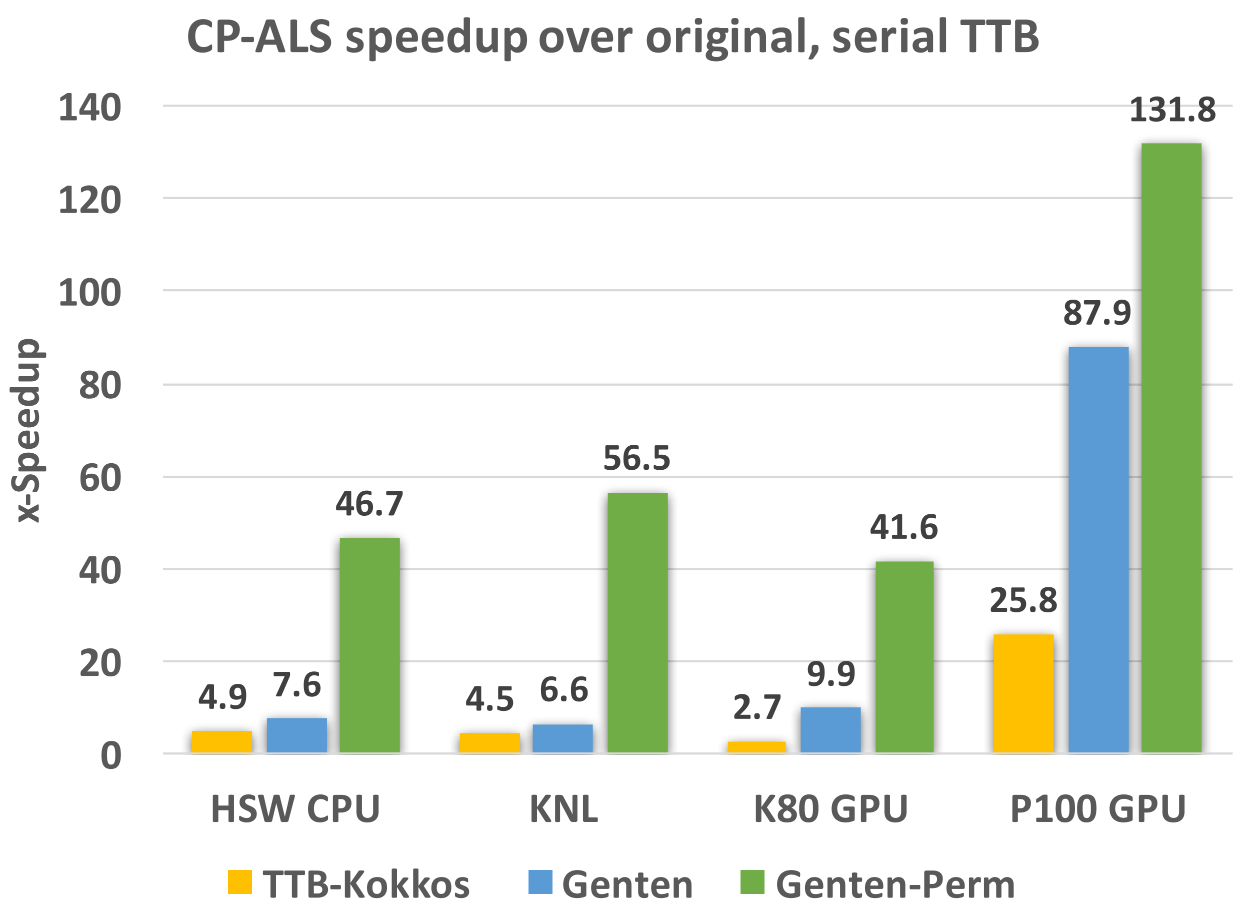
Model: $\mathcal{M} = \sum_r \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$

$$x_{ijk} \approx m_{ijk} = \sum_r \lambda_r \, a_{ir} \, b_{jr} \, c_{kr}$$

- Express the important feature of data using a small number of vector outer products

Hitchcock (1927), Harshman (1970), Carroll and Chang (1970)

CP-ALS using Kokkos



Random, synthetic data

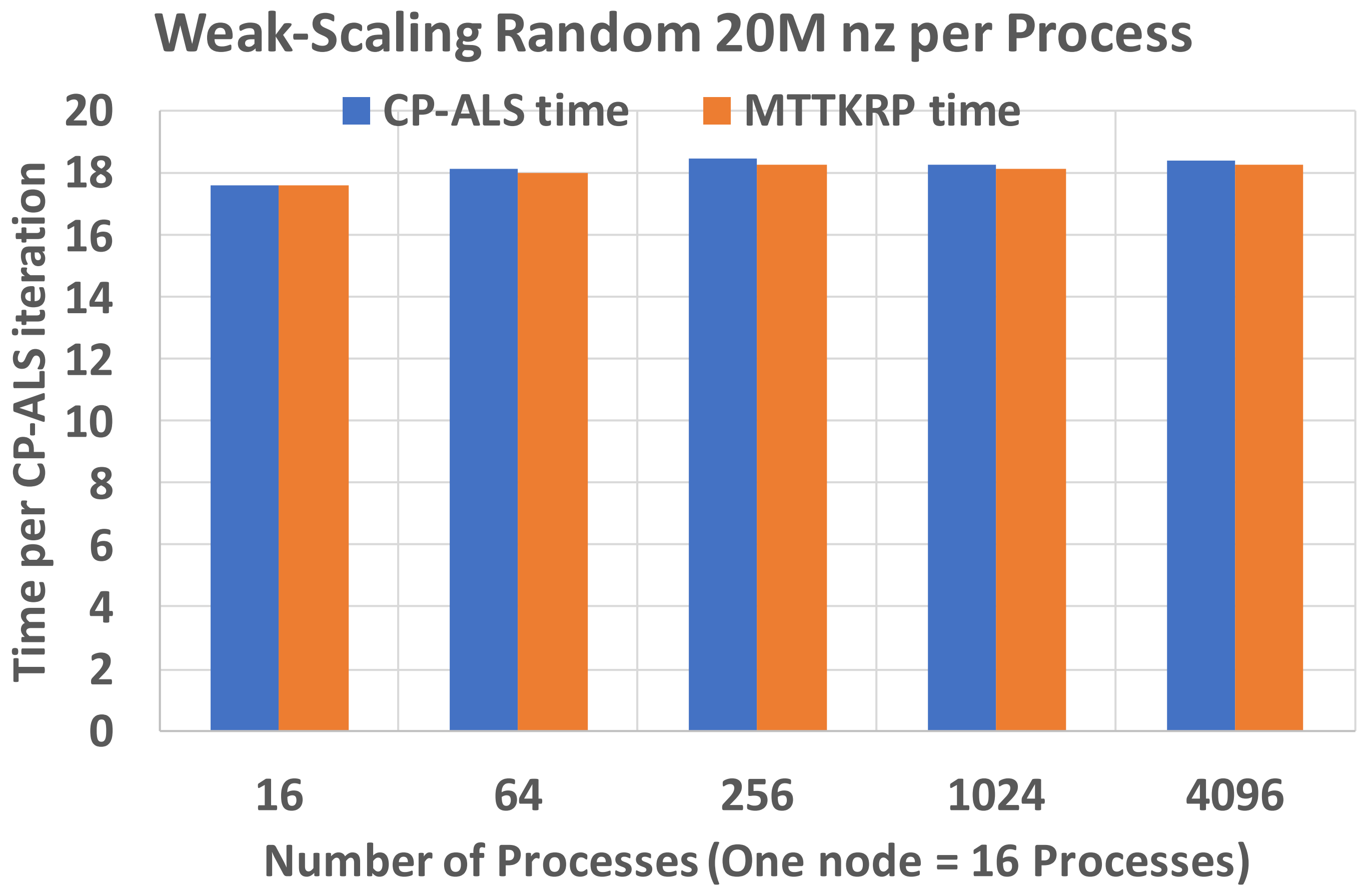
For continuous, real data (Gaussian model), we can use CP-ALS

Genten software

POC: Eric Phipps (etphipp@sandia.gov)

CP-ALS using Kokkos + Trilinos

- CP-ALS for **huge** sparse tensors in distributed memory
- 1.6TB tensor (82B nonzeros) on 4096 cores



POC: Karen Devine (kddevin@sandia.gov)

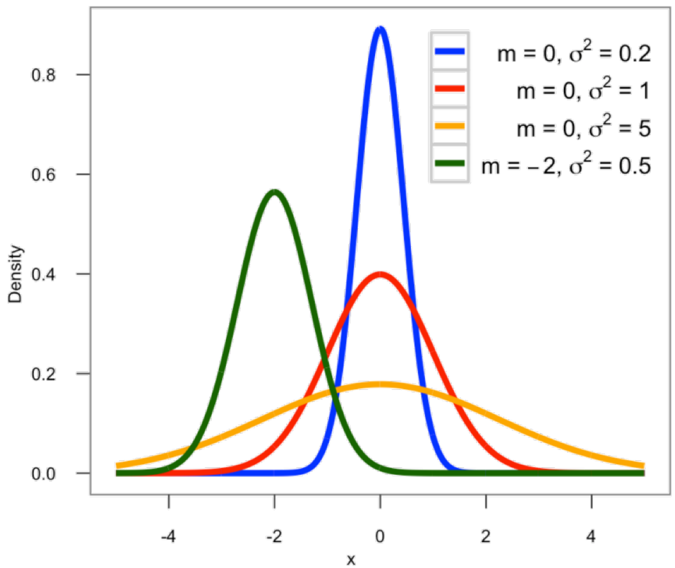
Poisson for Sparse Count Data

Gaussian (typical)

The random variable x is a continuous real-valued number.

$$x \sim N(m, \sigma^2)$$

$$P(X = x) = \frac{\exp(-\frac{(x-m)^2}{2\sigma^2})}{\sqrt{2\pi\sigma^2}}$$

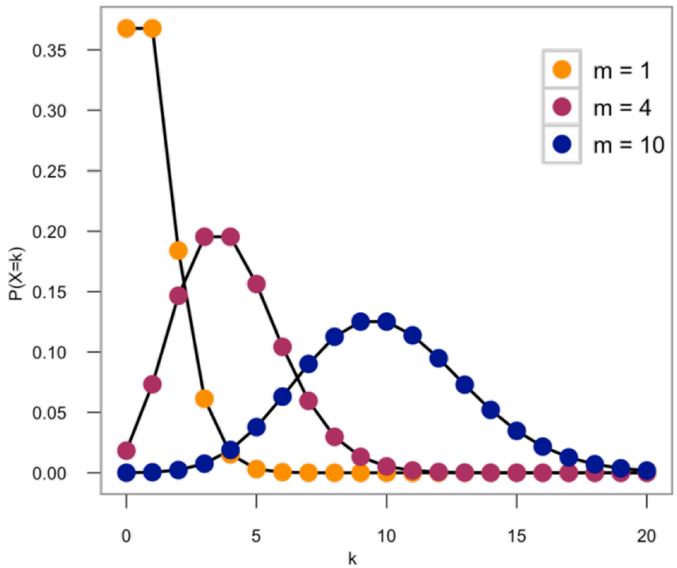


Poisson

The random variable x is a discrete nonnegative integer.

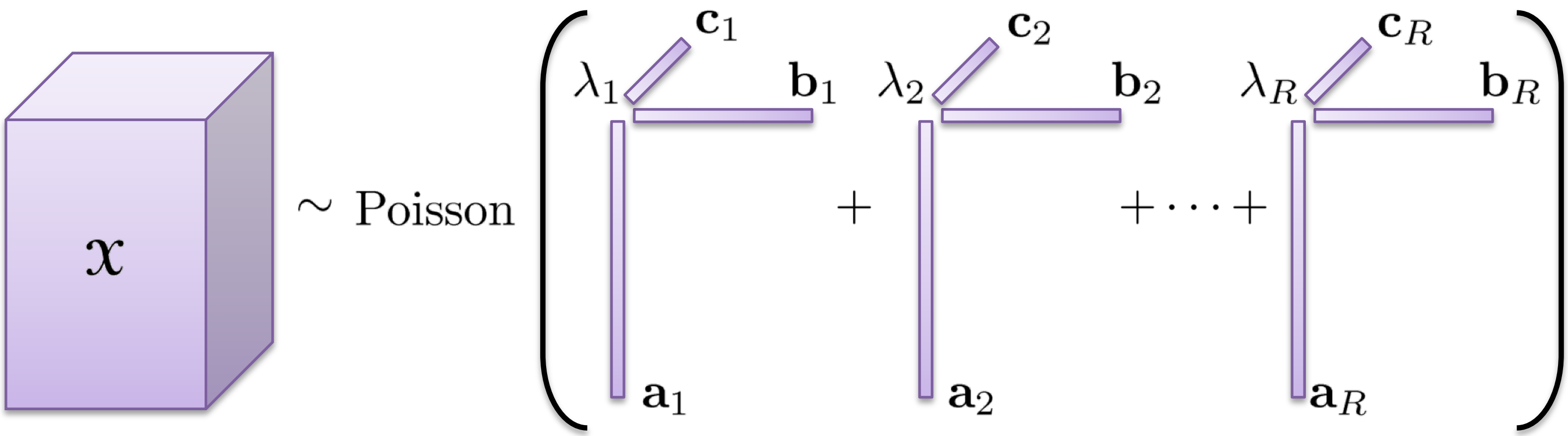
$$x \sim \text{Poisson}(m)$$

$$P(X = x) = \frac{\exp(-m)m^x}{x!}$$



Sparse Poisson Tensor Factorization

Sandia National Laboratories



<u>Model</u>: Poisson distribution (nonnegative factorization)

$$x_{ijk} \sim \text{Poisson}(m_{ijk}) \text{ where } m_{ijk} = \sum_r \lambda_r \, a_{ir} \, b_{jr} \, c_{kr}$$

- Nonconvex problem!
- Constrained minimization problem (decomposed vectors are non-negative)
- Alternating Poisson Regression (Chi and Kolda, 2011)
  - Assume (d-1) factor matrices are known and solve for the remaining one

- **Multiplicative Updates (CP-APR-MU)** by Chi and Kolda (2011)
- **Projected Damped Newton using Row-subproblems (CP-APR-PDNR)** by Hansen, Plantenga and Kolda (2014)

Parallel CP-APR-MU

Algorithm 1: CP-APR-MU in source

CP-APR-MU X, M, R;

Input : Sparse N-mode Tensor X of size $I_1 \times I_2 \times \ldots I_N$ and the number of components R

Output: Kruskal Tensor $\mathcal{M} = [\lambda; A^{(1)} \ldots A^{(N)}]$

initializeBuffer(X, R)

$\mathcal{E} \leftarrow$ computeIndexMap(X)

repeat

  for $n = 1, \ldots, N$ **do**

    $M \leftarrow$ offset(M, n) (Remove inadmissible zeros)

    $M \leftarrow$ distribute(M, n) (Scale the elements of A^n by λ)

    $\Pi^{(n)} \leftarrow$ computePi$(M, \mathcal{E}^{(n)})$

    for $i = 1, \ldots, 10$ **do**

      $\Phi_i^{(n)} \leftarrow$ computePhi$(A_i^{(n)}, \Pi^{(n)}, \mathcal{E}^{(n)})$

      $A_{i+1}^{(n)} \leftarrow A_i^{(n)} \Phi_i^{(n)}$

    end

    $M \leftarrow$ normalize(M, A, n)

  end

until *all mode subproblems converged*;

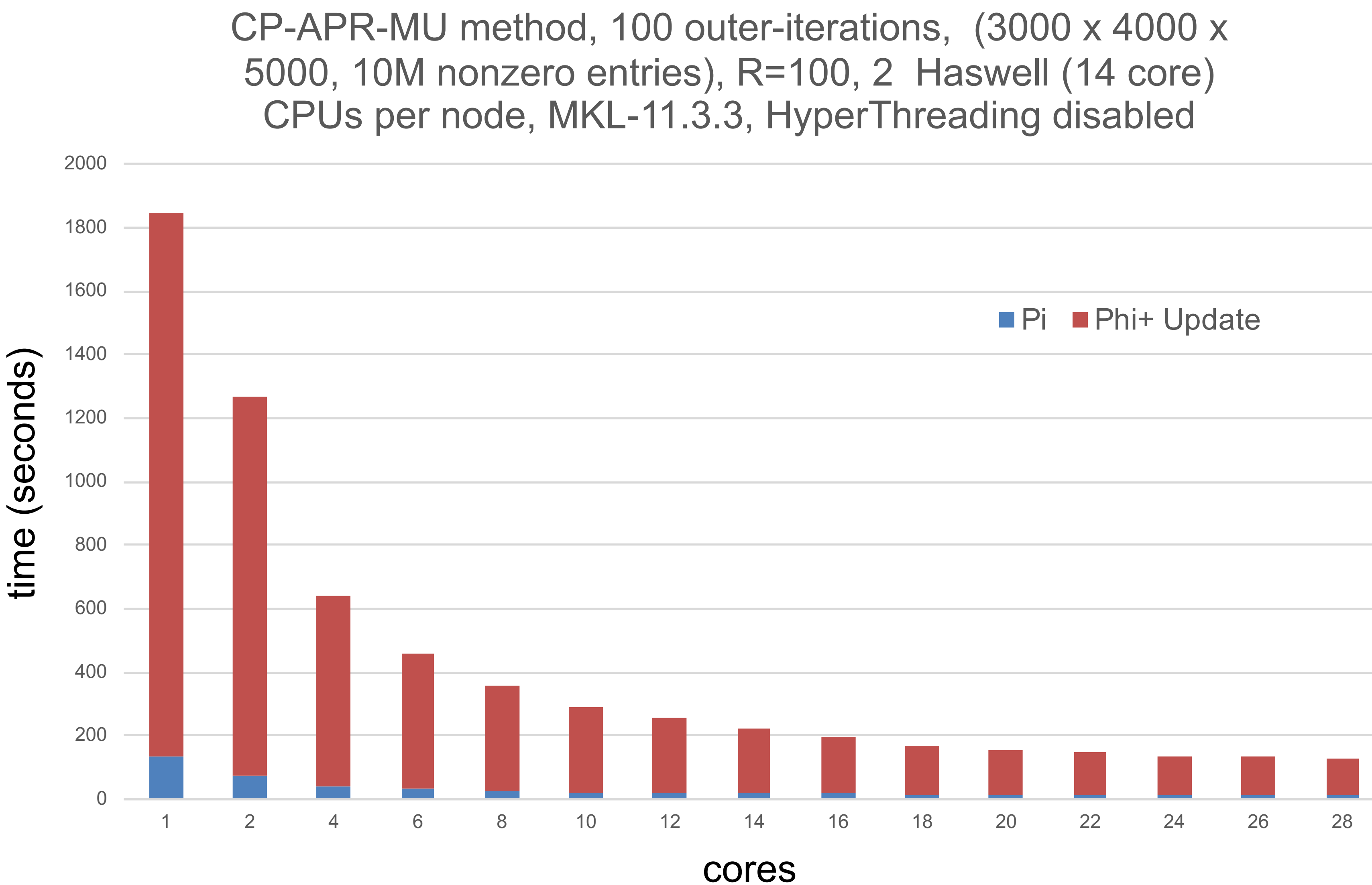
Data Parallel

CP-APR-MU Performance Test

- Strong Scalability
  - Problem size is fixed
- Random Tensor
  - 3K x 4K x 5K, 10M nonzero entries
  - **100 outer iterations**
- Realistic Problems
  - Count Data (Non-negative)
  - Available at http://frostt.io/
  - **10 outer iterations**
- Double Precision

Data	Dimensions	Nonzeros	Rank
LBNL	2K x 4K x 2K x 4K x **866K**	1.7M	10
NELL-2	12K x 9K x 29K	77M	10
NELL-1	3M x 2M x 25M	144M	10
Delicious	500K x **17M** x 3M x **1K**	140M	10

CP-APR-MU on CPU (Random)



Results: CPAPR-MU Scalability

Data	CPU 1-core		KNL (Cache Mode) 68-core CPU		NVIDIA P100 GPU		NVIDIA V100 GPU	
	Time(s)	Speedup	Time(s)	Speedup	Time(s)	Speedup	Time(s)	Speedup
Random	1849*	1	84	22.01	44.76	41.31	30.05	**61.53**
LBNL	39	1	33	1.18	2.99	13.04	2.09	**18.66**
NELL-2	1157	1	100	11.02	47.17	24.52	28.80	**40.17**
NELL-1	3365	1	257	10.86				
Delicious	4170	1	3463	1.41				

100 outer iterations for the random problem
10 outer iterations for realistic problems
* Pre-Kokkos C++ code on 2 Haswell CPUs: 1-core, 2136 sec

Kokkos based code runs on several architectures

Parallel CP-APR-PDNR

Sandia National Laboratories

Algorithm 1: CP-APR-PDNR in source

CP-APR-PDNR X, M, R;

Input : Sparse N-mode Tensor X of size $I_1 \times I_2 \times \ldots I_N$ and the number of components R

Output: Kruskal Tensor $\mathcal{M} = [\lambda; A^{(1)} \ldots A^{(N)}]$

initializeBuffer(X, R)

$\mathcal{E} \leftarrow$ computeIndexMap(X)

repeat

  for $n = 1, \ldots, N$ **do**

    $M \leftarrow$ distribute(M, n) (Scale the elements of A^n by λ)

    $\Pi^{(n)} \leftarrow$ computePi$(A, \mathcal{E}^{(n)})$

    parallel_for $i = 1, \ldots, I_n$ **do**

      $a_i^n \leftarrow$ rowSolvePDNR$(a_i^n, X^n, \Pi^n, \mathcal{E}_i^{(n)})$

    end

    $M \leftarrow$ normalize(M, A, n)

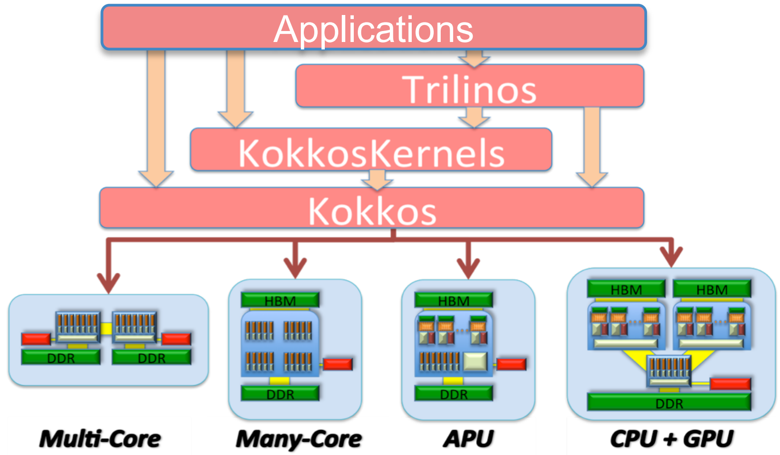
  end

until *all mode subproblems converged*;

Data Parallel

Task Parallel

Use Case 3: Finding Triangles with Kokkos Kernels for Node-Level Performance

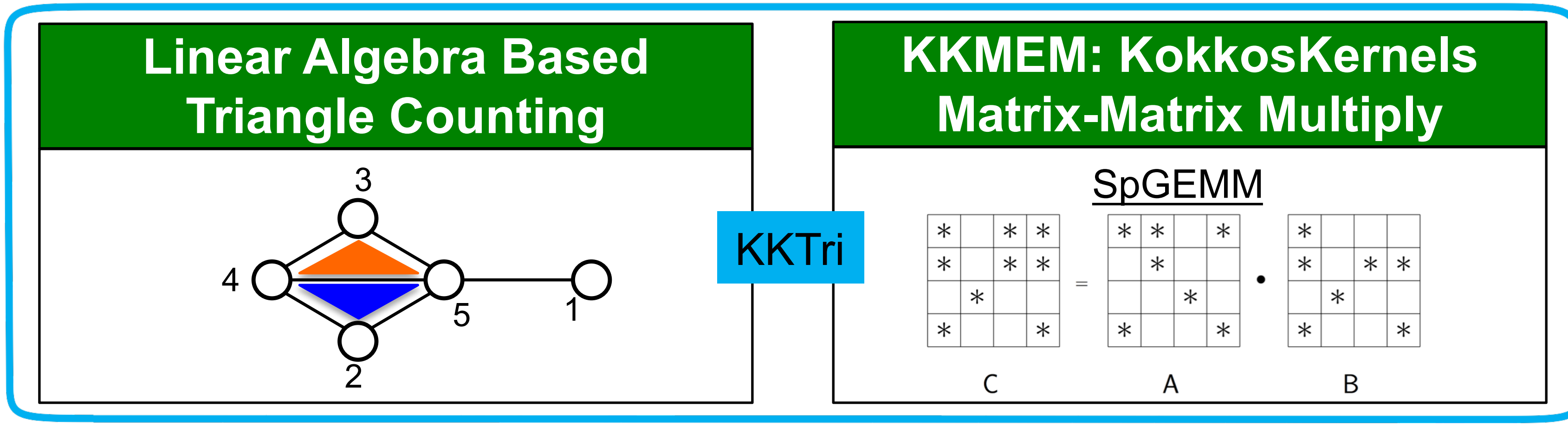


- Kokkos
  - Tools for performance portable node-level parallelism
  - Manages data access patterns, execution spaces, memory spaces
  - Performance portability not trivial for sparse matrix and graph algorithms
- Kokkos Kernels
  - Layer of performance-portable kernels for high performance
  - Sparse/Graph: SpMV, SpGEMM, triangle enumeration

Kokkos Kernels for performance-portable graph kernels

Kokkos Kernels KKTri and 2017 Graph Challenge

Sandia National Laboratories



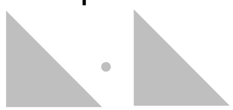
- 2017 MIT/Amazon/IEEE Graph Challenge Submission
  - Wolf, Deveci, Berry, Hammond, Rajamanickam: “Fast Linear Algebra-Based Triangle Counting with KokkosKernels.”
  - **Triangle Counting Champion** (focus: single node)
  - Counted 35B triangles in 1.2B edge graph in 43 secs (Twitter2010)

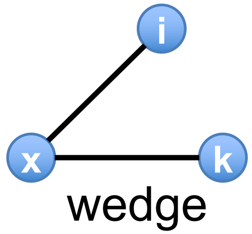
Vision: Build software on top of highly optimized KokkosKernels kernels (e.g., KKTri) to impact applications

Linear Algebra-Based Triangle Counting

C=L*L

Matrix-matrix multiplication

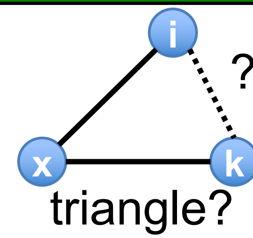




- C(i,k) = # of wedges with endpoints i,k

D=C.*L

Element-wise multiplication

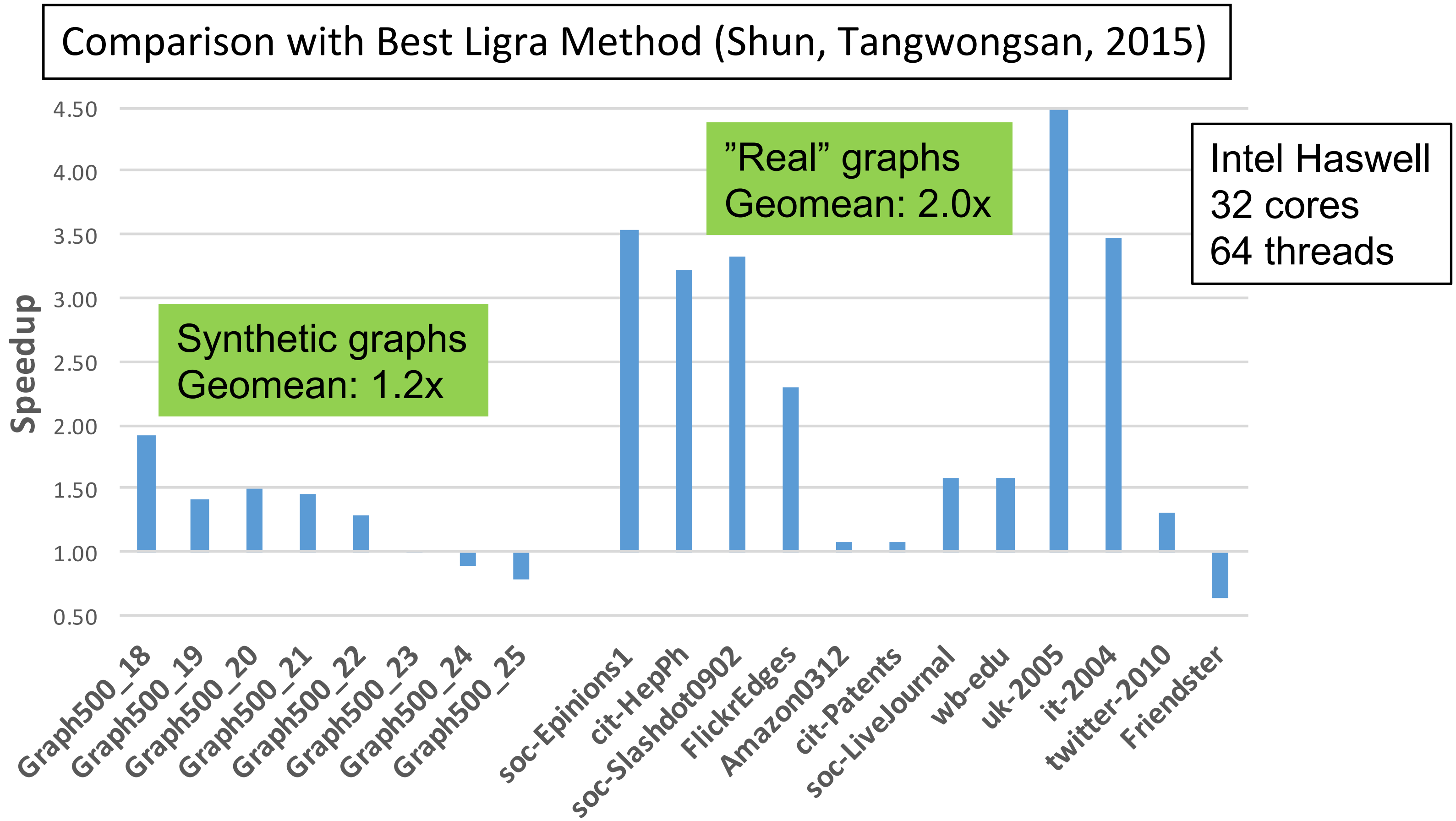


- D(i,k) = # of triangles with vertices i,k
- Filters out wedges i-x-k when edge i,k doesn't exist

- New linear algebra-based triangle counting method
  - Uses lower triangle part of adjacency matrix, L
  - Method: (L*L).*L
  - "Visits" each triangle/wedge once
- Once triangle is "visited," C++ functor/lambda used to count triangles
  - Other operations can be performed on each triangle

Functor enables "Visitor Pattern," which can add more flexibility to linear algebra approach

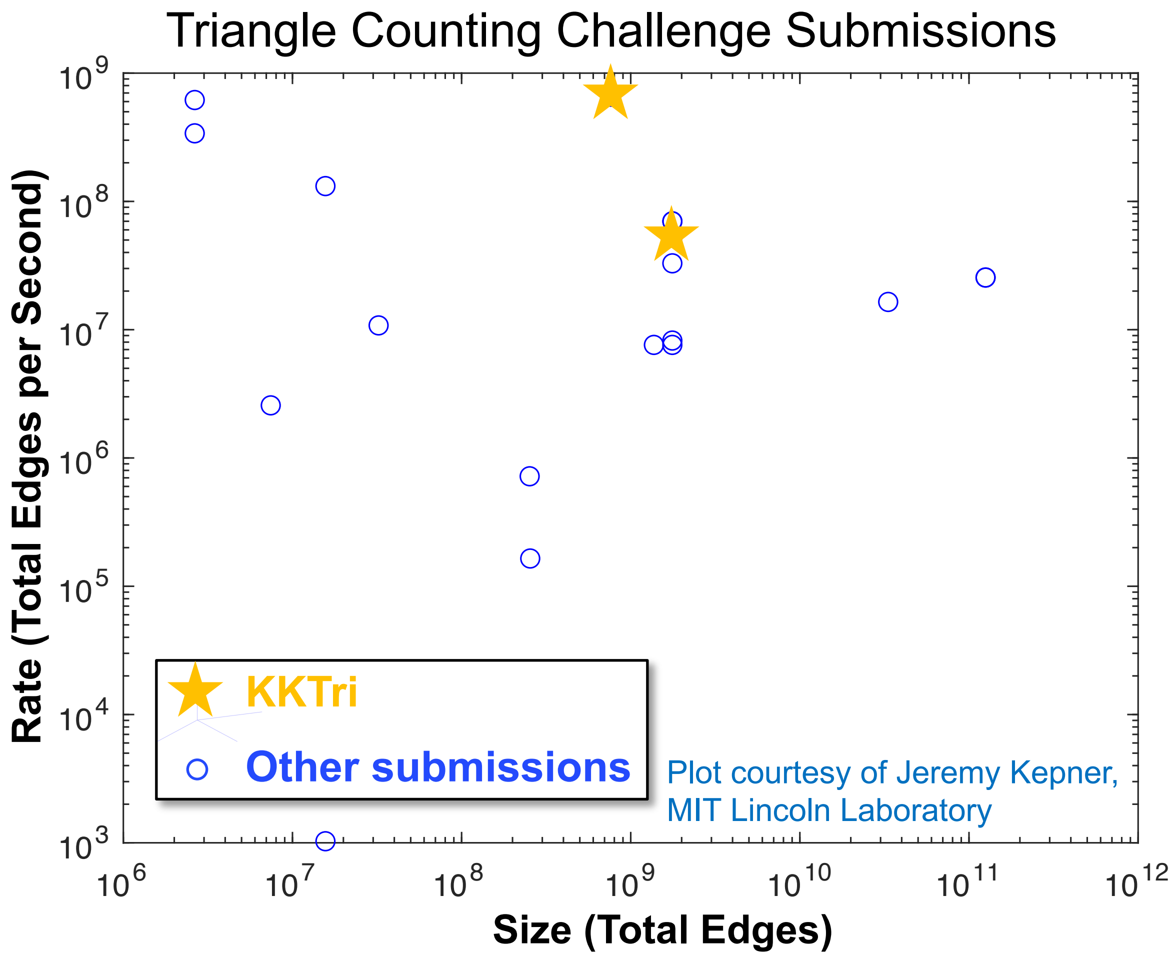
KKTri Speedup Relative to State of the Art



KKTri's linear algebra-based triangle counting outperforms state-of-the-art graph-centric method

Graph Challenge 2017: Lessons Learned

Sandia National Laboratories



Lessons Learned

- Linear algebra approach can be competitive
- Kokkos Kernels yields high performance triangle counting algorithm
- Visitor pattern can add more flexibility to linear algebra approach

Linear algebra-based KKTri as good as or better than other state-of-the-art methods

GraphChallenge 2018

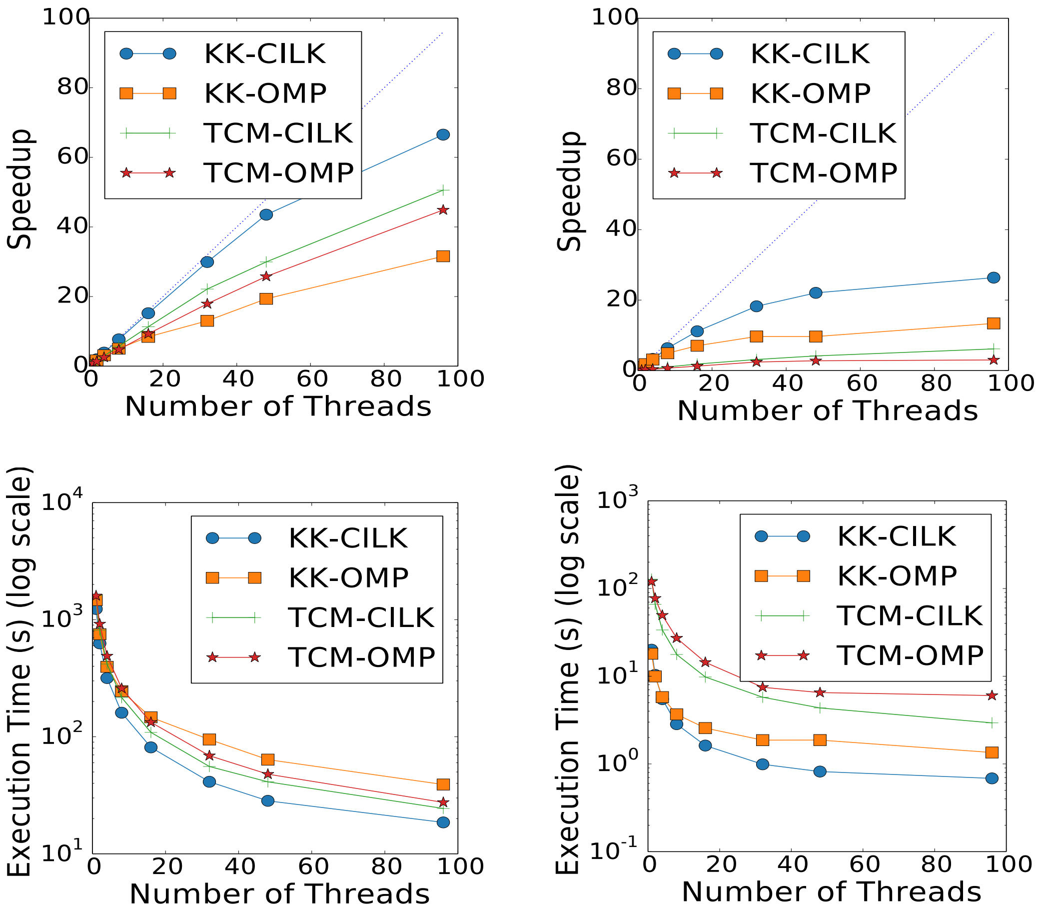
Sandia National Laboratories

- Kokkos Kernels-based triangle counting KKTri-Cilk
  - Replaced Kokkos/OpenMP with Cilk
  - Demonstrated improved usage of hyperthreading
  - Faster than Kokkos/OpenMP implementation on 63 of 78 instances
  - **Related: Abdurrahman Yasar, "Fast Linear Algebra-Based Triangle Analytics with Kokkos Kernels," Poster, W 7:30**
- KKTri-Cilk surpasses 10^9 rate on single multicore node
- **2018 MIT/Amazon/IEEE Graph Challenge Champion**

Example of HPDA driving Kokkos development
- **Improving hyperthread usage**
- **Cilk backend**

Graph Challenge: 2018 Strong Scaling

Friendster | UK-2005



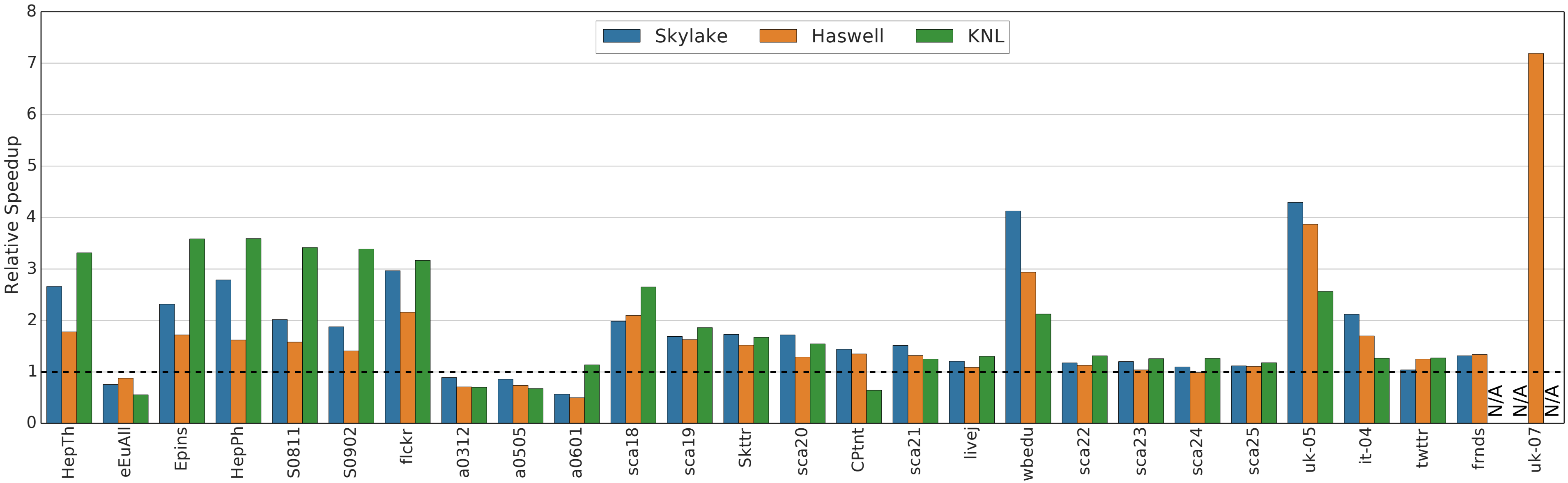
KKTri-Cilk scales the best in both problems

uk-2005 graph has a very good ordering: highly local computations (best rate).

Friendster graph has best scalability

Scaling is with respect to the best sequential execution time.

Graph Challenge 2018: Relative Speedup

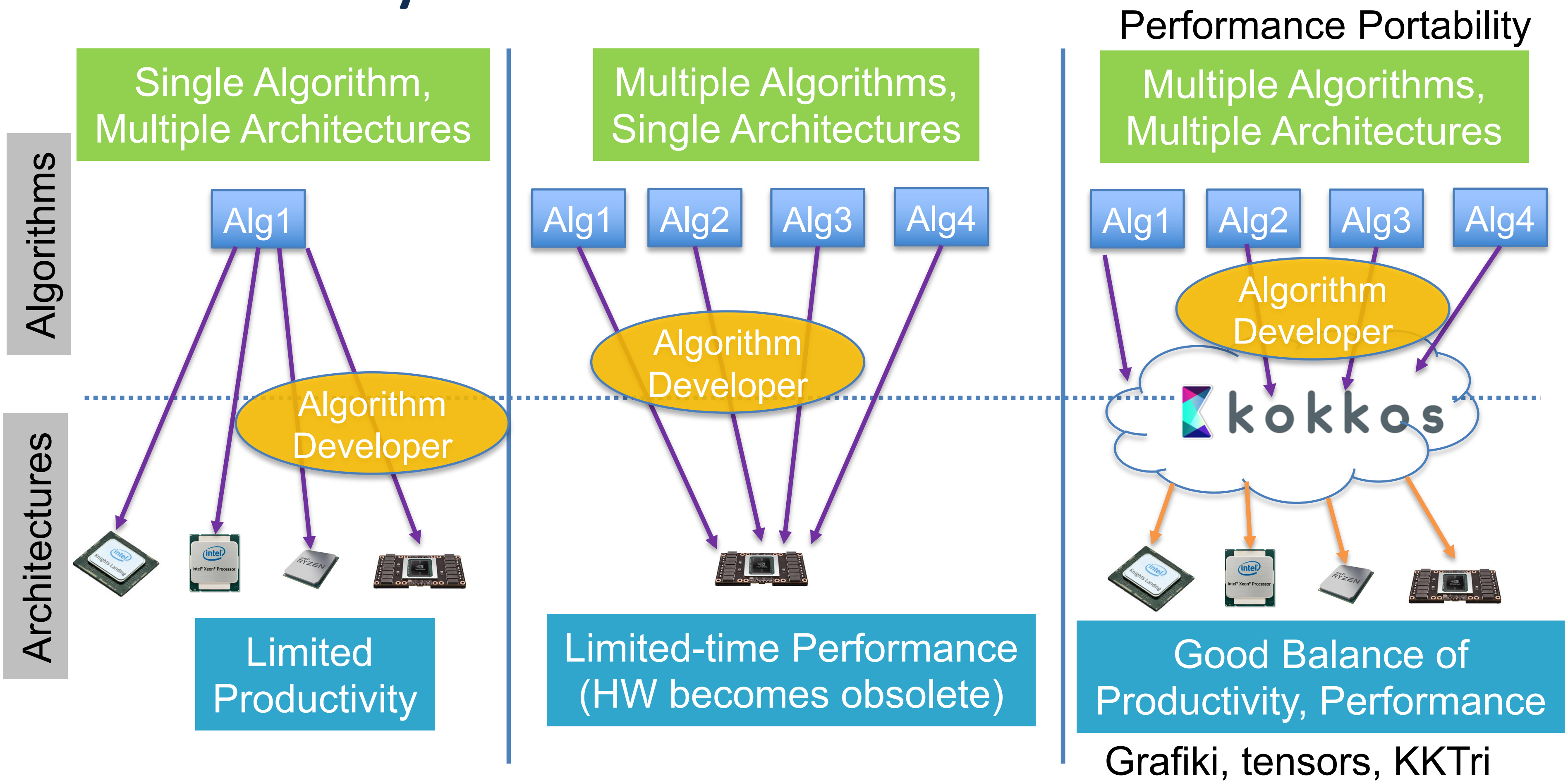


- Comparisons of KKTri-Cilk with TCM, a state-of-the-art graph library [Shun *et al.*]
- KKTri outperforms TCM in 23 of 27 cases
- KKTri can achieve up to 7x speedup on graphs that have a good natural ordering such as wb-edu, uk-2005, and uk-2007

Kokkos Kernels used to develop high optimized graph algorithm

Summary

Sandia National Laboratories



- **Choice where to focus HPDA efforts**
- **Performance-Portable Kokkos enables productivity of algorithm developer and performance on several architectures**

Conclusions

- Results show promise of Kokkos Ecosystem for HPDA
- Improvements to Kokkos (based on HPDA experience) will yield additional performance improvements
- **More work ahead**
  - Algorithms, optimized kernels, integration, architectures, …
  - Machine learning – **ECP ExaLearn Co-Design Center**
- **Much software is available**
  - Kokkos: https://github.com/kokkos/kokkos
  - Kokkos Kernels: https://github.com/kokkos/kokkos-kernels
  - GenTen: https://gitlab.com/tensors/genten
  - SparTen: https://gitlab.com/tensors/sparten
  - Triangle Counting: https://github.com/Mantevo/miniTri
  - Coming soon: Grafiki

Thank You

Questions?

- Contact: mmwolf@sandia.gov