Graph Based Contractions for Calculating Correlation Functions

Redstar: Analysis Software Suite

Jie Chen, Eloy Romero Alcalde, Frank Winter and Robert Edwards

Jefferson Lab

U.S. DEPARTMENT OF ENERGY | Office of Science

JSA

Talking Points

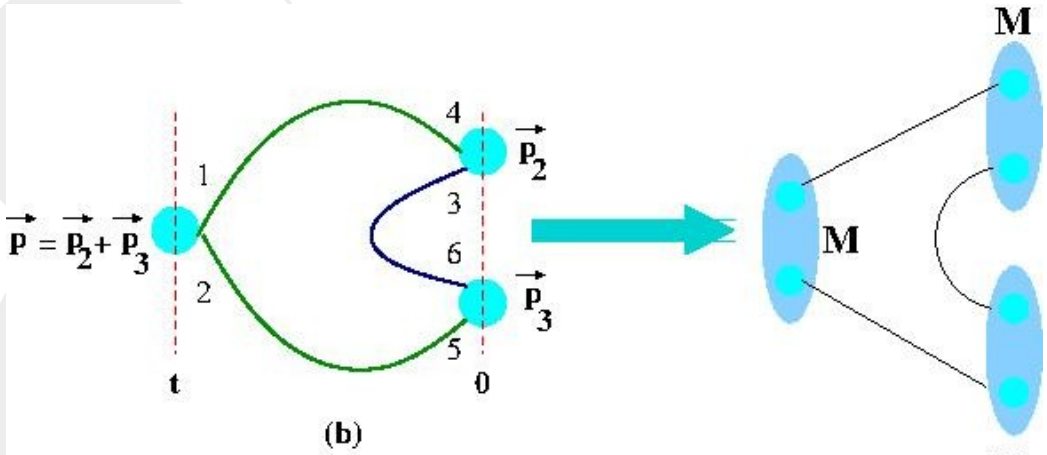
- What is graph contraction?
- ***Redstar.***
  - Current optimizations.
    - Reduce number of contraction calculations and run-time memory footprint.
  - Hadron contractions.
    - Carry out real tensor contractions on CPUs and GPUs.
- Performance
  - Mostly on GPUs.
  - Observations.
- Challenges.

Correlation Functions and Graphs

$$C(t',t) = \langle \chi(t')\chi^{\dagger}(t) \rangle \qquad \hat{H}|k\rangle = E_k|k\rangle \qquad C(t,t') = \sum_k |\langle \chi|k\rangle|^2 e^{-E_k(t'-t)}$$

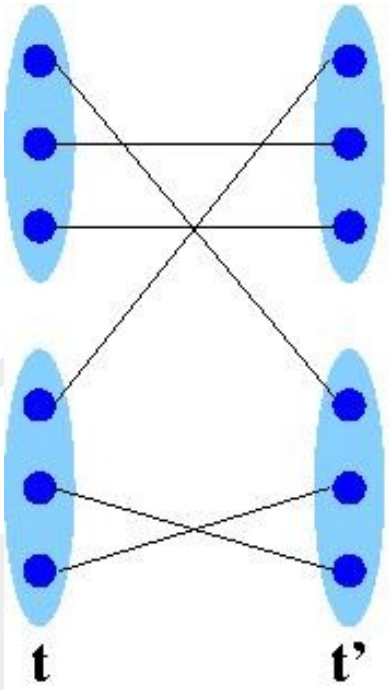
Smearing and distillation

$$C(t,0) = \sum_{p_2,p_3|p} c_{p_2,p_3} M^{12}(\vec{p},t) P^{25}(t,0) M^{56}(\vec{p}_2,0) P^{63}(0,0) M^{34}(\vec{p}_3,0) P^{41}(0,t)$$

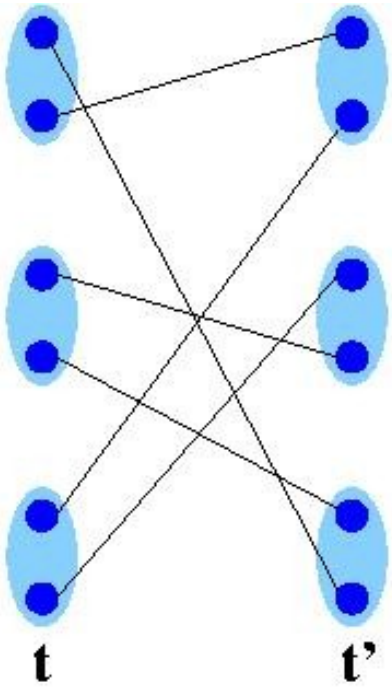


(b)

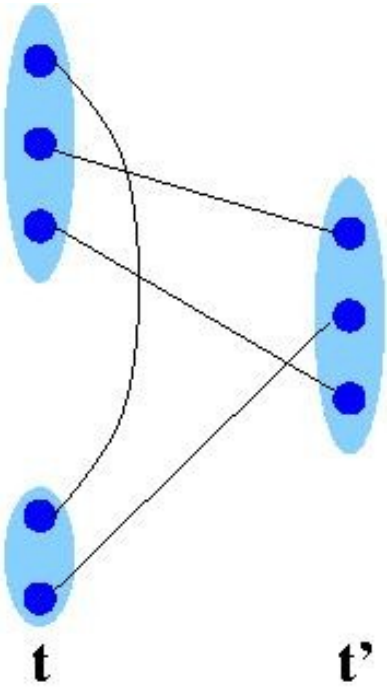
Different Types of Graphs



(a)



(b)



(c)

Graph Nodes (V)

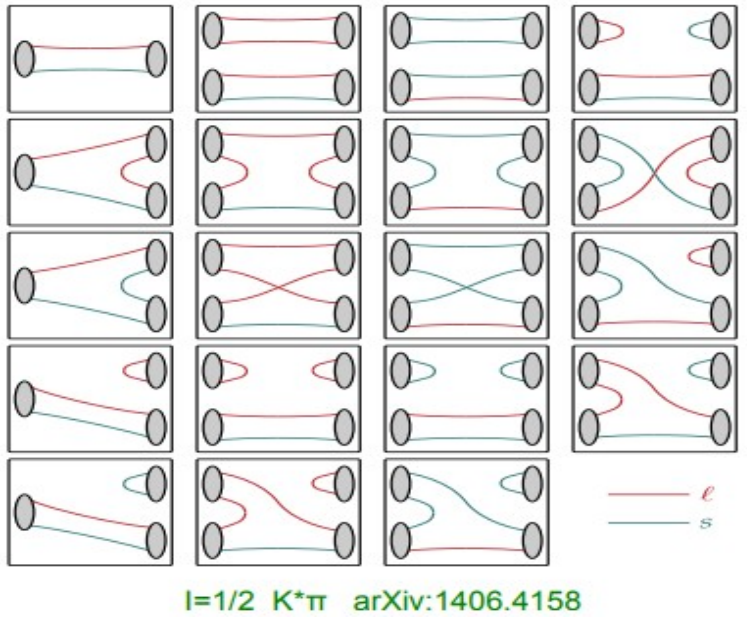
- Meson (2 quarks), Baryon (3 quarks)
- Meson
  - Each vertex $M^{ij}_{\alpha\beta}$ where α,β=0,1,2,3 (spins) and i,j ~ hundreds (distillation space).
  - e.g. each vertex occupies 384*384*16*16 = 37 MB
- Baryon
  - Each vertex $B^{ijk}_{\alpha\beta\gamma}$ where α,β,γ=0,1,2,3 (spins) and i,j,k ~ 100s (distillation space)
  - e.g. each vertex occupies 128*128*128*16*64= 2GB, each vertex occupies about 128*128*128*16*8=268 MB if only two upper spin indices are used (α,β,γ=0,1).

Graph Edges (E)

- Meson-Meson contraction $A^{ij}_{\alpha\beta}\, B^{jk}_{\beta\gamma}$
  - Batched matrix multiplication with batch size <= 64
  - $O(N^3)$ for calculations, $O(N^2)$ for memory
- Meson-Bryon contraction $A^{ij}_{\alpha\beta}\, B^{jkl}_{\beta\gamma\delta}$
  - Batched tensor contraction with batch size <= 256
  - $O(N^4)$ calculation complexity, $O(N^3)$ memory
- Baryon-Baryon Contraction
  - One index contraction $A^{ijk}_{\alpha\beta\gamma}\, B^{klm}_{\gamma\delta\varepsilon}$  Try to avoid if possible
    - $O(N^5)$ calculation complexity, $O(N^4)$ memory
    - Huge memory (e.g 128^4 * 16 * 16 = 68GB for two upper spins)
    - Batched tensor contraction with batch size up to 1024
  - Two-index contraction $A^{ijk}_{\alpha\beta\gamma} B^{jkl}_{\gamma\beta\delta}$
    - $O(N^4)$ calculation complexity, $O(N^3)$ memory.

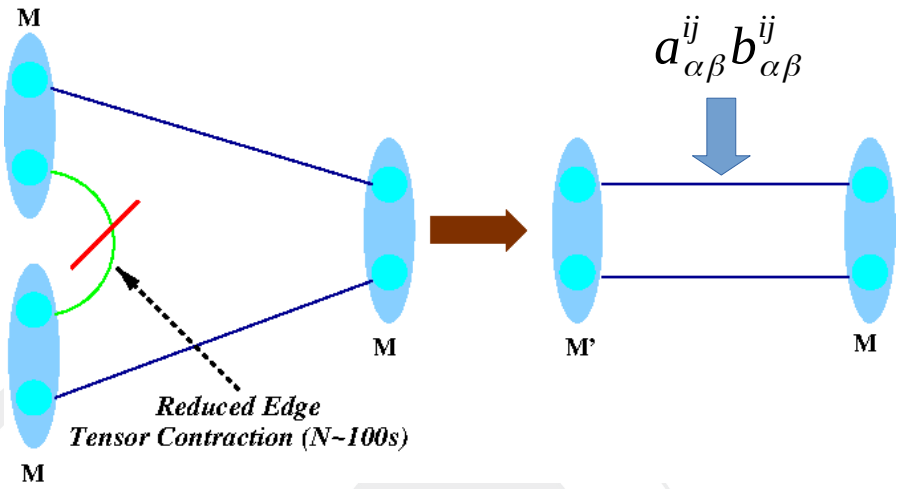
How Many Graphs ?

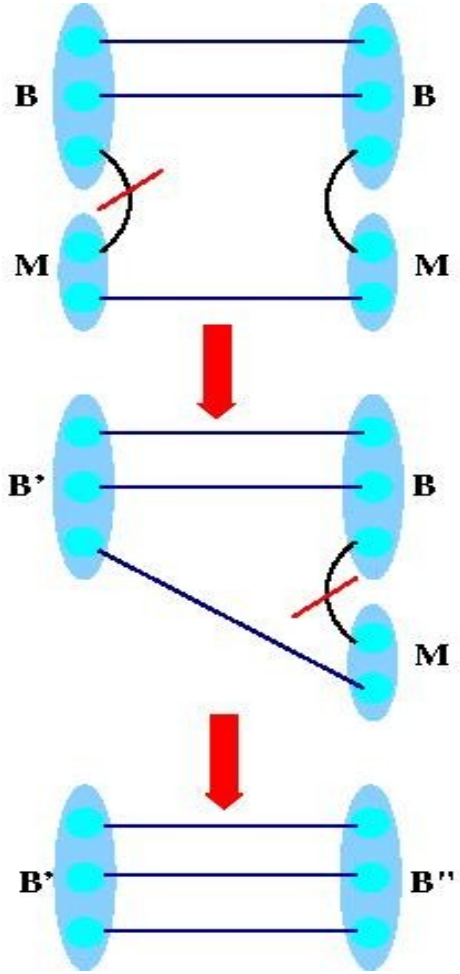
- $Number\ of\ Graphs \propto N!$
  N is the number of freedom of degrees
- Recently approaching 1M graphs
- Calculations across many time slices



Name	a_0	f_0	a_1	roper	Deuterium	Tritium
# Graphs	19,041	27,999	385,512	84,894	119,191	6,208
Type	MM	MM	MM	MB	BB	BB
Vector Size	256	256	128	64	64	32

Graph Contractions

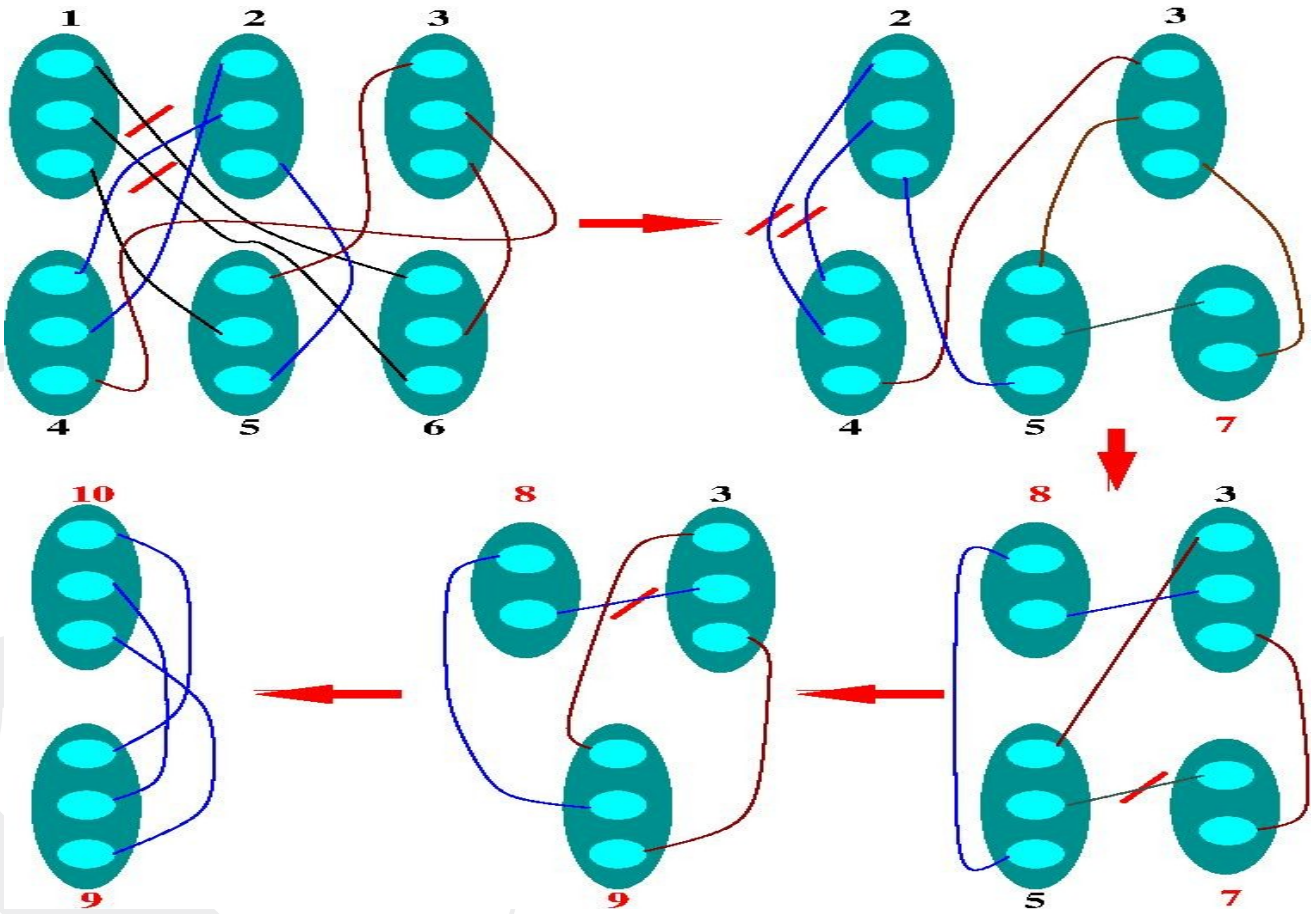




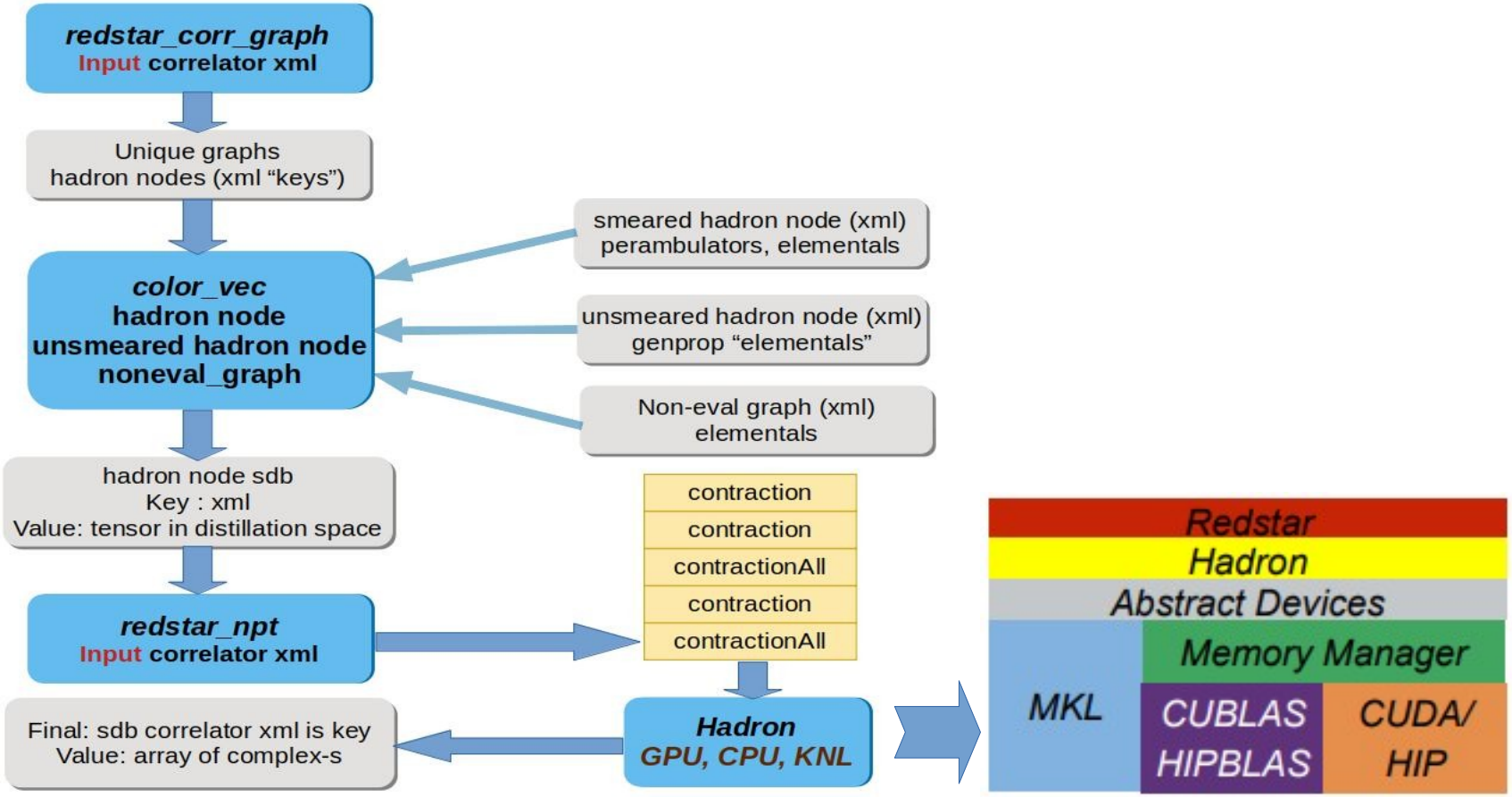
Removal of one edge after another until two hadron nodes are left

The sequence of the edge removal is important

Graph Contractions

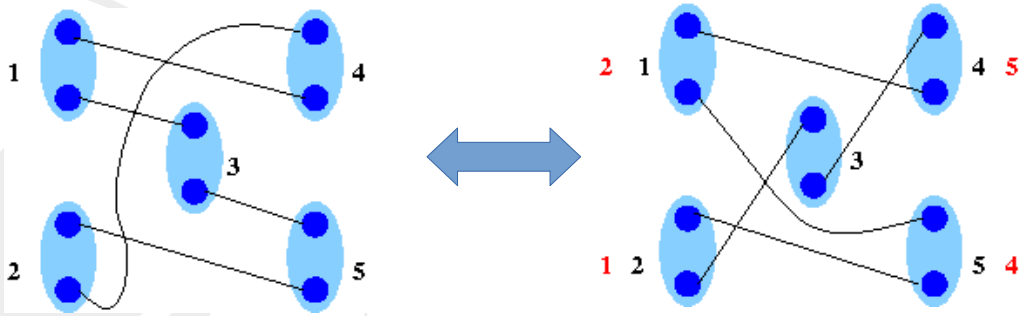


Redstar Workflow



Graph Generation and Classification (*redstar_npt*)

- Parallel contraction diagram and graph generations (*Wick Contraction*)
- Parallel graph classifications to identify unique graphs

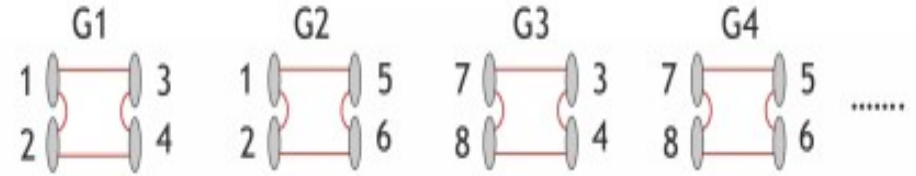


- Graph isomorphism problem.
- Perform canonical labeling of each graph in parallel.
- Identify unique graphs in O(n) time.
  - *Brendan D. McKay (Nauty)*

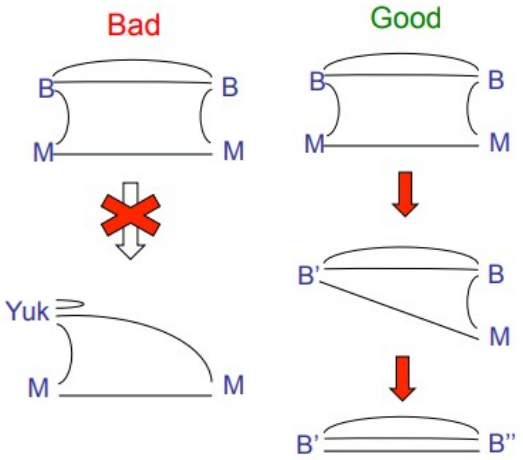
GPU ???

Optimal Graph Contraction Path and Memory Reduction

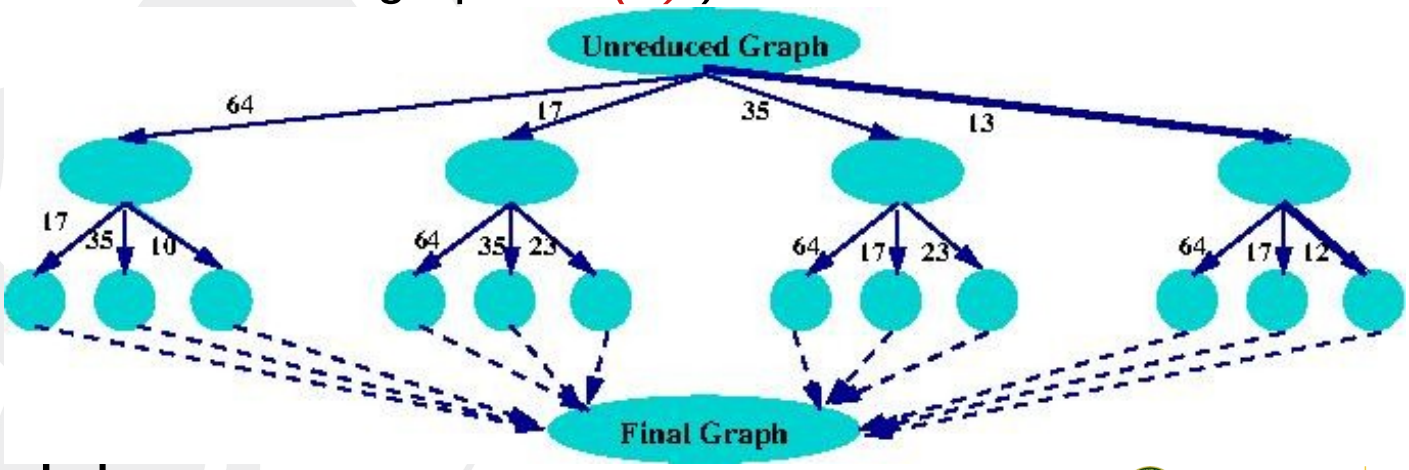
- There are many contraction paths for a graph.



An edge can appear in many graphs



- Each edge is assigned a weight (calculation cost/the number of appearances in all the graphs, *O(N)*)



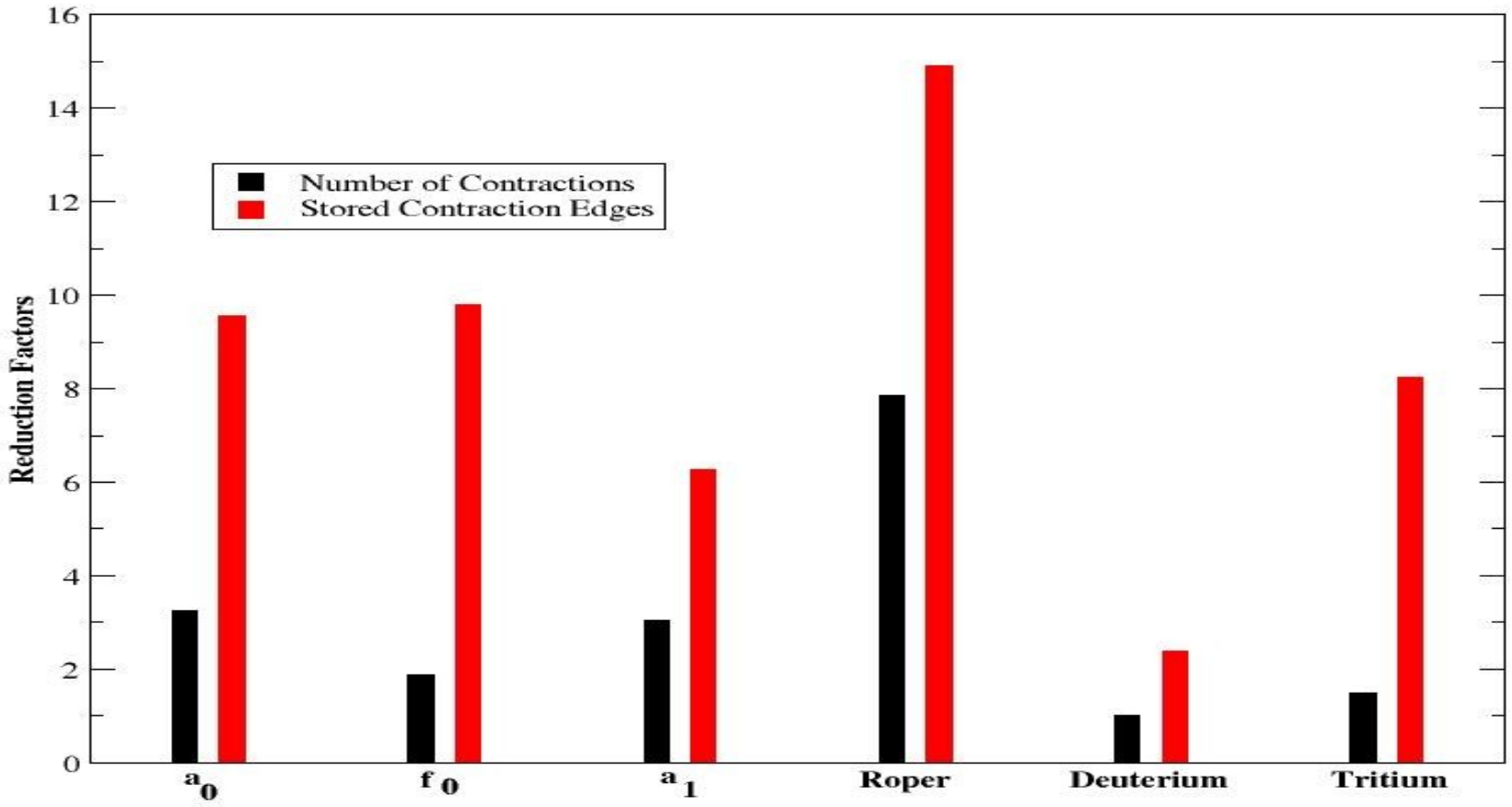
N!/2 possible paths

Optimal Graph Contraction Path and Memory Reduction

- Minimum spanning tree algorithm is used to select an optimal path to reduce the number of contractions.
  - Prefers 2-index contractions (smaller calculation cost)
  - For meson systems, the edges with most appearances are chosen first.

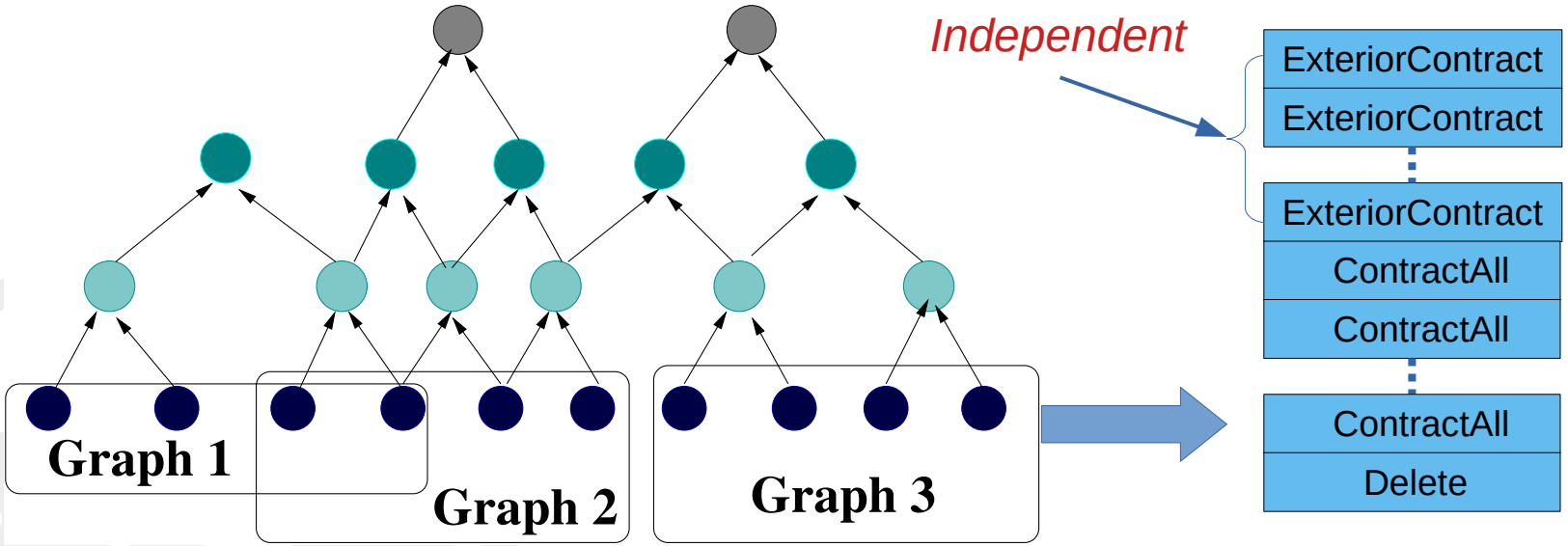
- Similar graphs are grouped together during the construction of a contraction queue to minimize the memory footprint of the intermediate contraction results.

Optimal Graph Contraction Path and Memory Reduction



Redstar and Hadron Contraction

- A gigantic *DAG* is created after the optimal contraction path is determined for each graph.



- A contraction queue is constructed from the DAG. Independent contractions can be grouped together to form arrays of contractions (***vector-form*** contractions).

Hadron Contraction

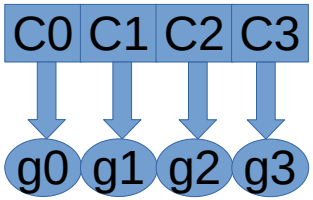
- **Batched tensor contractions on CPUs or GPUs**
  - **Batched zgemm coupled with tensor permutation if needed.**
  - **Vector-form of hadron contraction to increase batched zgemm batch size on CPUs.**
  - **Vector-form of hadron contraction to utilize multiple GPUs each could use multiple streams.**
    - **Each element in a contraction queue (an array) has no information about on which GPU it executes.**
    - **Hadron library utilizes an algorithm that decides where to put a contraction based on where the contraction sources have been.**
      - **Keep data locality high and maintain good load-balance.**

Contraction APIs

```
HadronDistOperatorRep
  hadronExteriorContract(const HadronDistOperatorRep& src1_rep,
                         int ind_j,
                         const HadronDistOperatorRep& src2_rep,
                         int ind_k);

  HadronDistOperatorRep
  hadronExteriorContract(const HadronDistOperatorRep& src1_rep,
                         const std::vector<int>& ind_j,
                         const HadronDistOperatorRep& src2_rep,
                         const std::vector<int>& ind_k);

  std::vector<HadronDistOperatorRep>
  hadronExteriorContract(const std::vector<HadronDistOperatorRep>& src1,
                         const std::vector<std::vector<int>>& ind_j,
                         const std::vector<HadronDistOperatorRep>& src2,
                         const std::vector<std::vector<int>>& ind_k);
```



More about the DAG and the Contraction Queue

- Contraction queue is constructed from the DAG by breadth-first walk through of the DAG.
  - No gpu information assigned to any contraction.
- Independent contractions are grouped together to form an array of contractions.
  - NPT_BATCH_SIZE
    - Array of contractions
  - Increase batch size for batched zgemm on KNL (or many core CPU) to take advantage of computing power of many cores.
    - Works very well for KNL nodes.
  - Each contraction in the array is carried out by a dedicated GPU if multiple GPUs are available.
    - If the sources of all contractions in the array are different, we expect a linear speedup.
    - Unfortunately, we have situation like : extcontract(a, 0, b, 1);

      extcontract(a, 1, b, 0);

      extcontract(a, 0, c, 1);

      extcontract(b, 0, c, 0);
      - In this example, a, b, c have to be copied or moved to different GPUs.
      - Multi-gpus may not provide any benefit.

Hadron Contraction on GPUs

- Performance bottlenecks
  - Data in and out of GPUs kill the performance.
    - Need to consolidate all the spin components on host side into a contiguous memory block to reduce the number of memory copies.
    - Need data reuse.
    - Data stay on a GPU as long as possible unless they are needed on the host.
    - Memory evictions due to the limited memory on a GPU.
    - Data movements from one GPU to another.
  - Many individual matrix multiplications perform poorly
    - Data movements and not enough computation on a GPU.
  - Naive tensor permutations perform terribly on GPUs
    - Stride data access using the global memory on a GPU.
  - Some matrix matrix multiplications are actually matrix vector multiplications depending on how two tensors' indices are folded.
    - Needs to identify those cases and implement batched version of **gemv**.

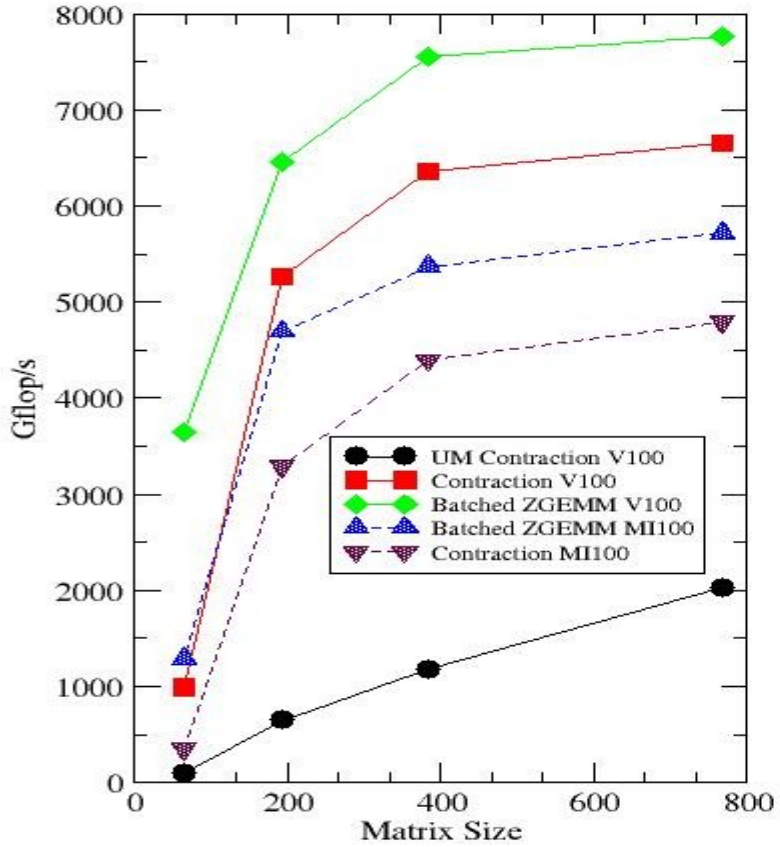
Hadron Contraction on GPUs
Current Implementation and Optimizations

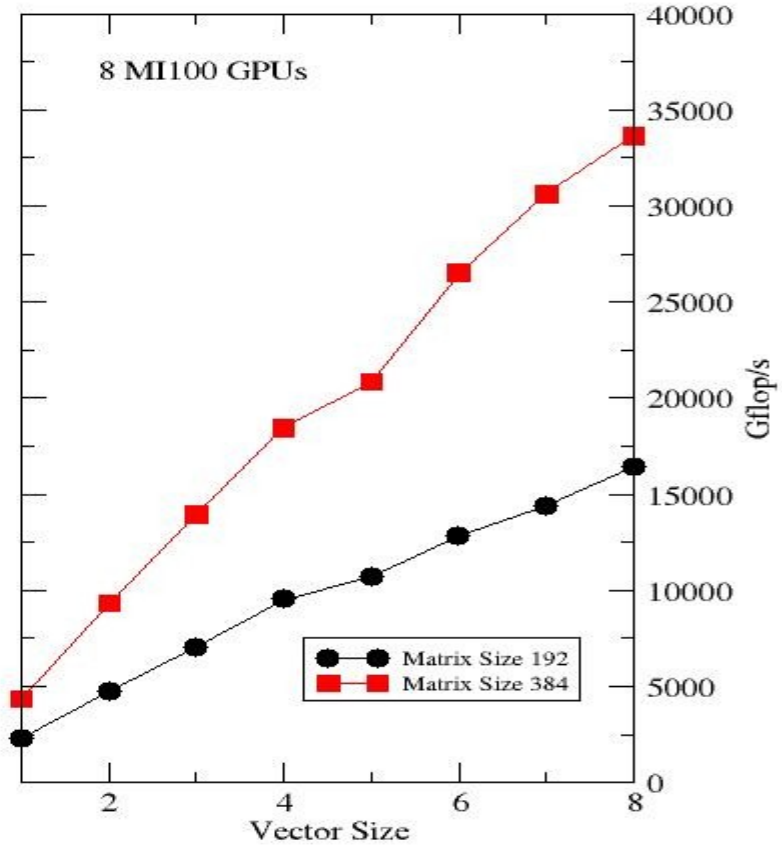
- Major components
  - **GPU memory managers**
    - One manager for one GPU.
    - cuda or hip malloc and free are going through the manager.
      - Contiguous gpu/host memory block to reduce I/O
    - Manages data evictions (improved LRU) and data reuse.
    - Intermediate contraction values stay on GPUs.
  - **GPU matrix/tensor contraction kernels**
    - Batched zgemm from cublas/hipblas.
    - Customized batched zgemv, zdotu, accumulate, contractall.
  - **GPU tensor transpose library** (***GPUTT***)
    - Fast tensor transpose using the best transpose algorithm according to the shape of a tensor and the permutation indices. Again based on *cutt* by *Dmitry I. Lyakh.*
    - Batched transpose.
    - Permuted data stay on a GPU.
  - **Smart multi-gpu data placement algorithm for vector-form contractions.**

ACM TACO Journal paper

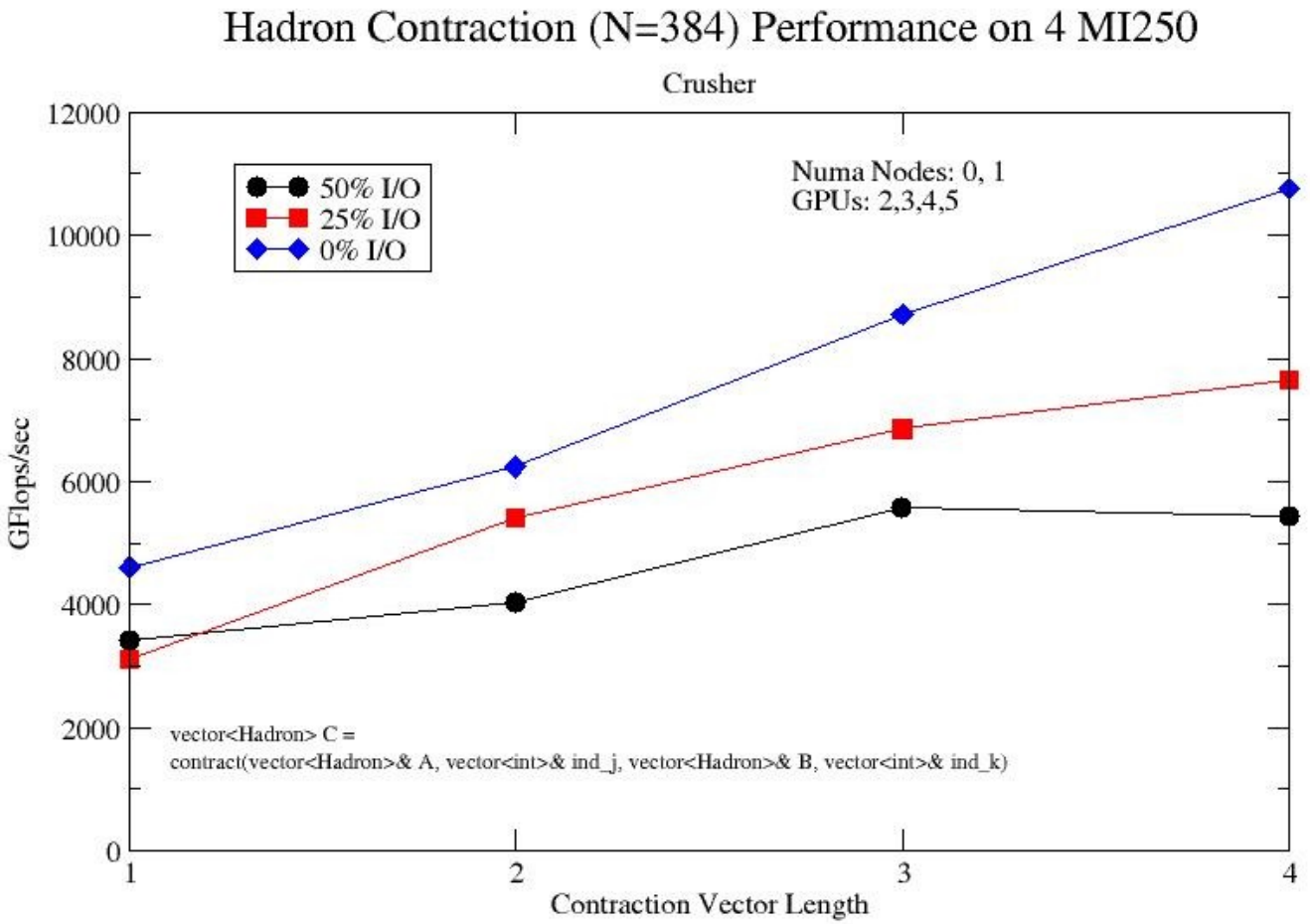
IPDPS 22

Performance of Hadron Contraction on GPUs





Performance of Hadron Contraction on GPUs



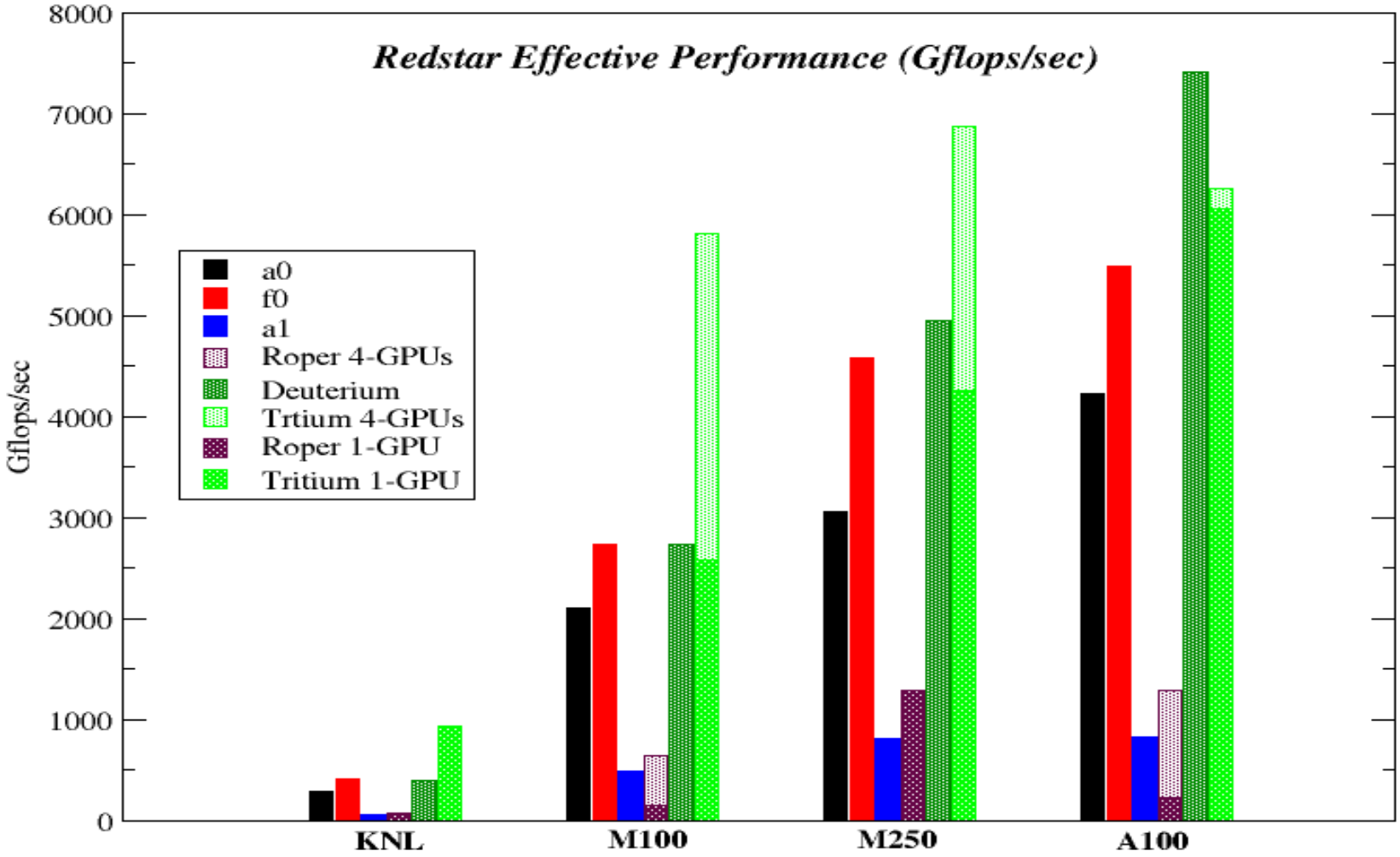
Random Inputs
Random Pairs

Redstar_npt execution time for 1-timeslice (one GPU)

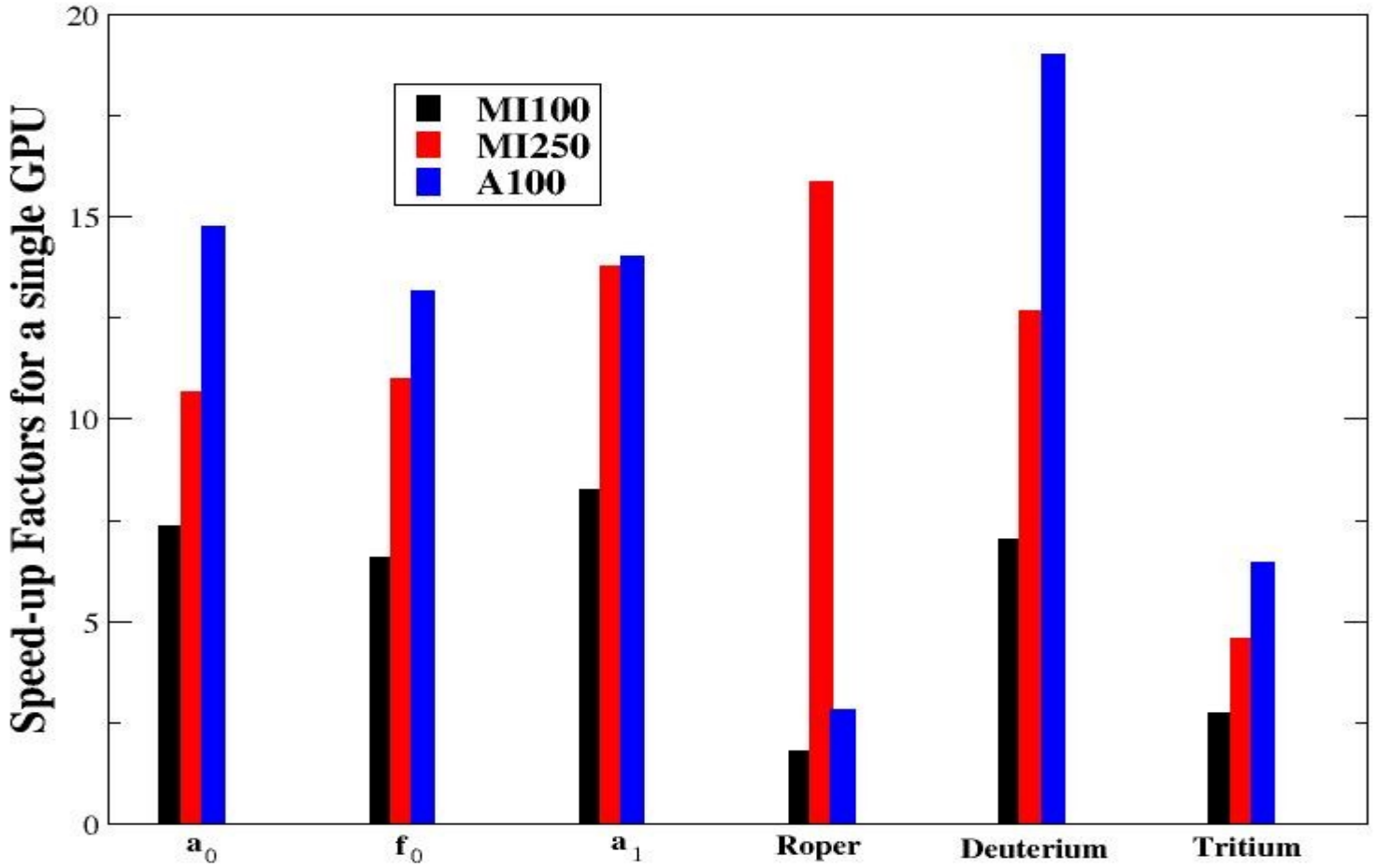
- Recent redstar_npt runs on our test configurations.
- KNL 7250 host, MI100, MI250 and A100.

#	Name	N_{ref}	T_{orig}	KNL	MI100	MI250	A100
1	a_0	256	260.7	87.95	11.94	8.23	5.96
2	f_0	256	412.2	138.89	21.14	12.63	10.55
3	a_1	128	1427.11	866.70	107.56	64.28	63.22
4	Roper	10 **64**	778.2	3488.05	1933.64	219.86	1238.44
5	Deuterium	10 **64**	1471.9	3637.48	517.21	286.58	191.21
6	Tritium	10 **32**	1735	8208.01	2970.04	1803.66	1269.93

Redstar Performance



Redstar Performance

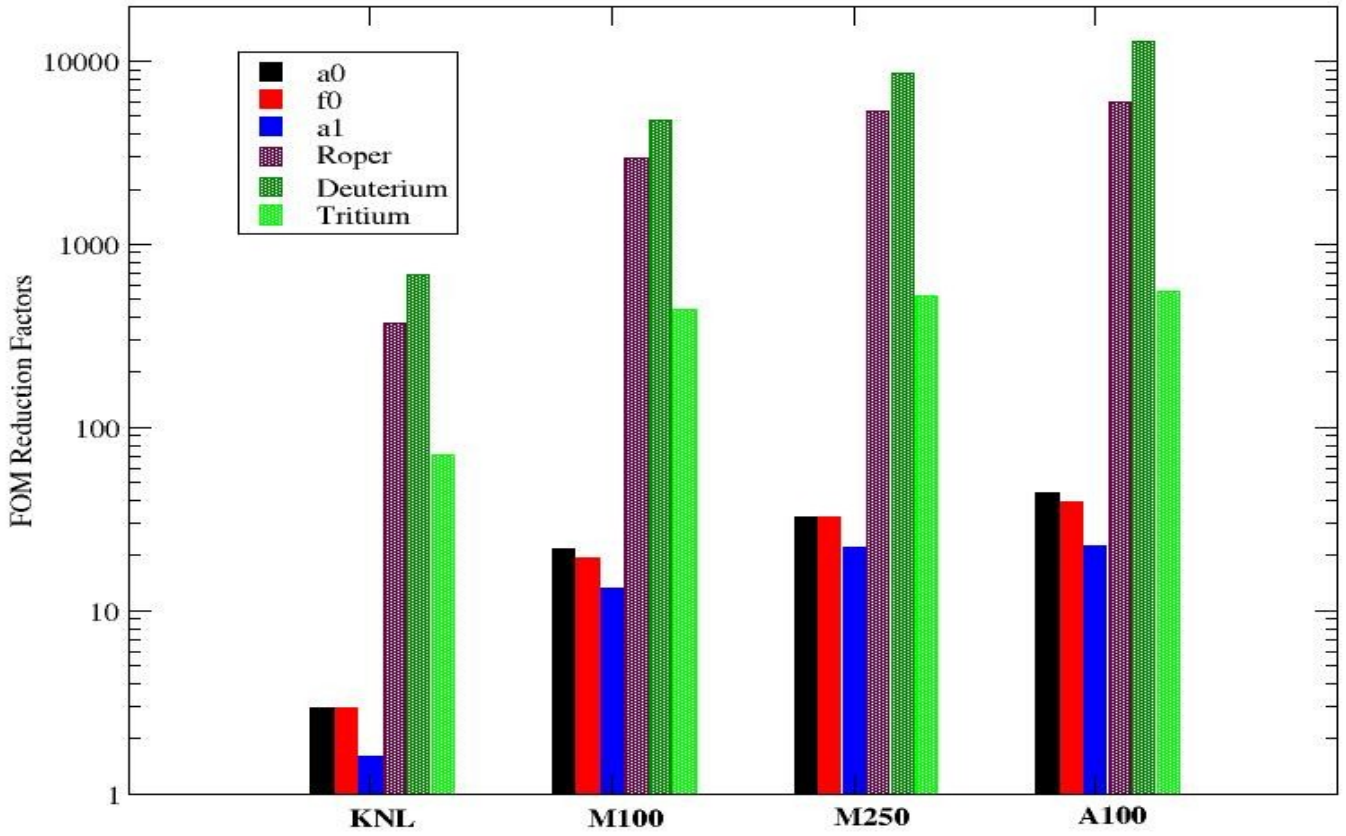


Performance speed-up values of a GPU over a single KNL host for calculating Correlation functions of multi-hadron systems.

Redstar Performance (Figure of Merit)

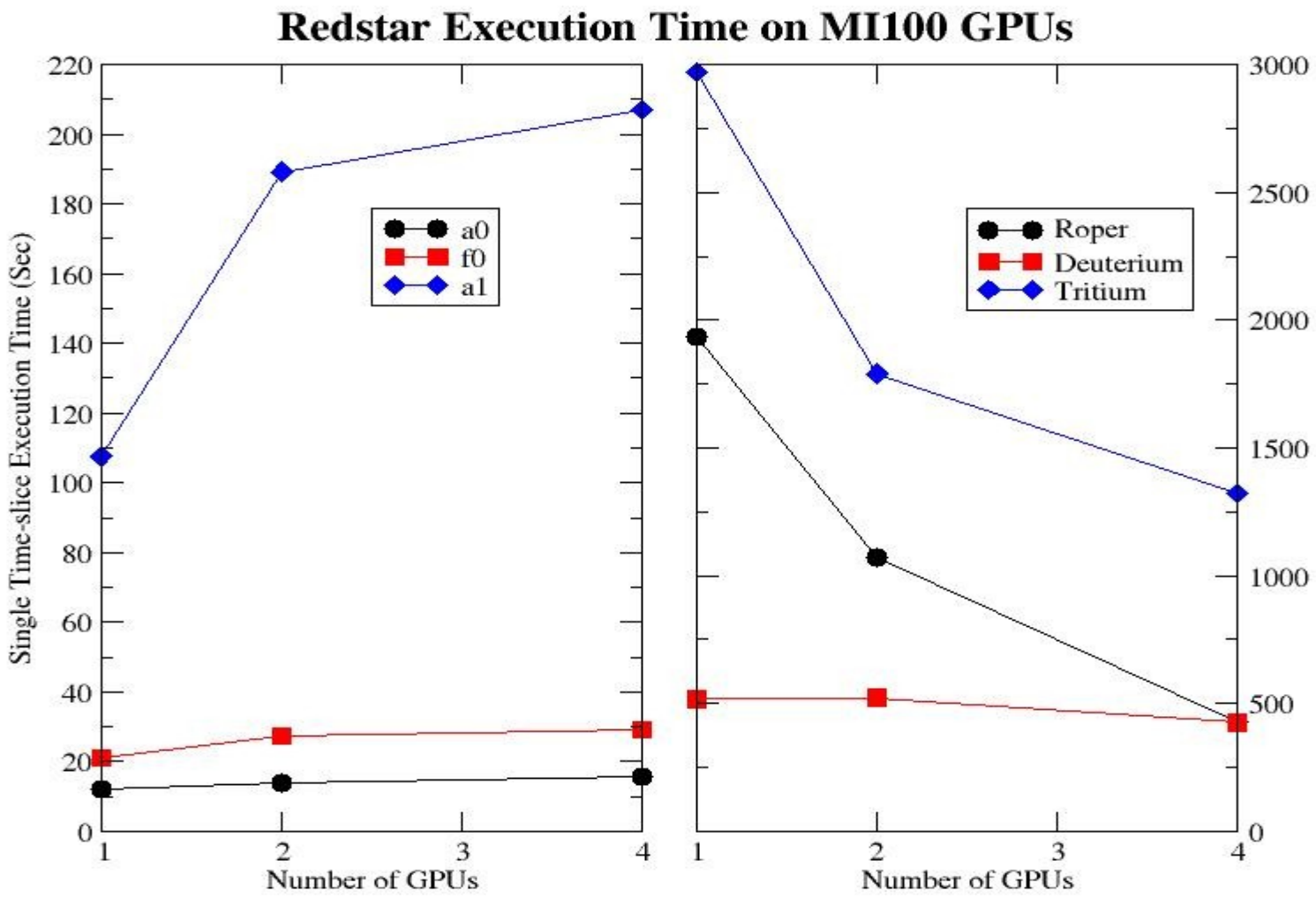
$$FOM \propto \left(\frac{N_{target}}{N_{ref}}\right)^{p} \times timing$$

P is 3 for a_0, f_0,a_1, 4 for Roper, and 5 Deuterium and Tritium
N_{target} = 256



Jefferson Lab

Redstar Performance on Multiple GPUs



NPT_BATCH_SIZE = Number of GPUs

Redstar Performance on Multiple GPUs: Impact of data movements

- For meson systems, no gpu memory evictions. The performance degradation is caused by GPU to GPU/Host movements and relative small matrix size (N=256 for a_0, f_0 and N=128 for a_1).
- Define T_s as time to calculate contractions on a single GPU, BW as the effective unidirectional bandwidth between GPUs/Host, N_{flops} number of flops for all contractions, CN_{gflops} as the gflops/s for one contraction for matrix size N, Nbytes number of bytes transferred, T_n as the execution time on n gpus.
  - $T_n = Ts - N_{flops} / (CN_{gflops} * 10\text{^}9) + N_{flops} / (CN_{gflops} * 10\text{^}9)/n + Nbytes/BW$

	a_0 2gpu	a_0 4gpu	f_0 2gpu	f_0 4gpu	a_1 2gpu	a_1 4gpu
T_n	13.75	15.78	27.21	29.01	189.28	207.14
T_s	11.94	11.94	21.14	21.14	107.56	107.56
N_{flops}	$2.52*10^{13}$	$2.52*10^{13}$	$5.79*10^{13}$	$5.79*10^{13}$	$5.26*10^{13}$	$5.26*10^{13}$
CN_{gflops}	3991	3991	3991	3991	1699	1699
Nbytes(GB)	115.63	145.19	195.66	236.20	681.93	1051.94
BW (GB/s)	23.28	22.18	14.68	12.59	7.01	8.56

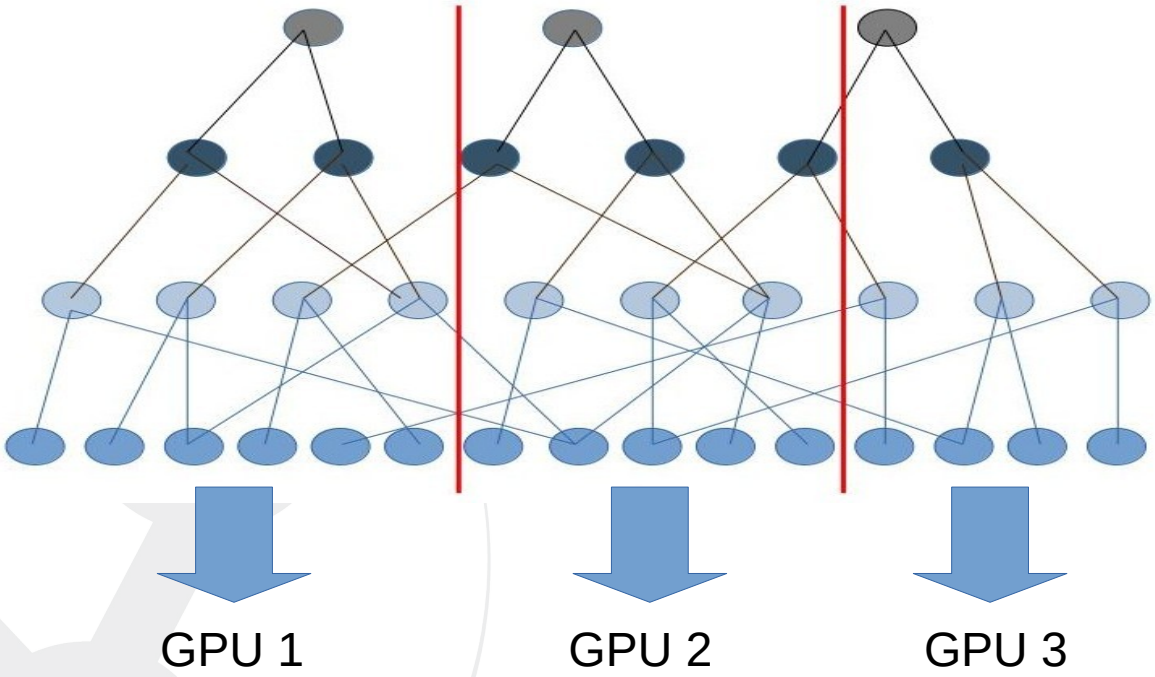
Redstar Performance on Multiple GPUs

- **For Baryon systems, data transfers and gpu memory evictions affect the performance.**

Name	# GPUs	N	# Flops	Evictions	Bytes (GB)	Time (sec)
	1			32356		1933.64
Roper	2	64	2.83×10^{14}	15290	3200	1073.32
	4			1671	5725	429.06
	1			17767		517.21
Deuterium	2	64	1.42×10^{15}	235	8986	522.56
	4			667	14552	425.58
	1			2749		2970.04
Tritium	2	32	7.69×10^{15}	2671	753	1790.77
	4			2354	1113	1323.79

Reduce Data Movements on Multi-GPUs by Partitioning the DAG

- Divide the DAG into multiple similar sized partitions with minimal number of edges crossing the borders of the partitions.



Near Term Challenges

- **Redstar and hadron contractions**
  - **Reduce data movements on multi-gpu.**
    - **Hopefully partitioning the DAG will reduce the data movements.**
    - **Each partition of contractions is handled by a GPU through a separate contraction queue.**
  - **Baryon-Baryon contractions on GPUs**
    - **One single contraction cannot fit on a single GPU.**
      - **Using multiple GPUs need to move partial GPU contraction results back to the host.**
    - **Memory evictions becomes difficult because each object is not small.**
    - ***When to split a contraction on multiple GPUs ?***
  - **Graph Isomorphism on GPUs?**
    - **Can we beat *Nauty* ?**
  - **Optimal graph contractions.**
    - **Is local optimal selection = global optimal ?**
  - **Multi-host redtar_npt through MPIs.**
    - **Still need to do partitioning the DAG (which host to do which parts of the DAG).**
  - **Intel CPU/GPUs.**