

Scalable In-situ Indexing For Fast Trajectory Queries

Qing Zheng, Brad Settlemyer, Chuck Cranor, George Amvrosiadis
Garth Gibson, Greg Ganger, Gary Grider, Fan Guo

2018 USRC Annual Symposium

Carnegie Mellon University
Los Alamos National Laboratory

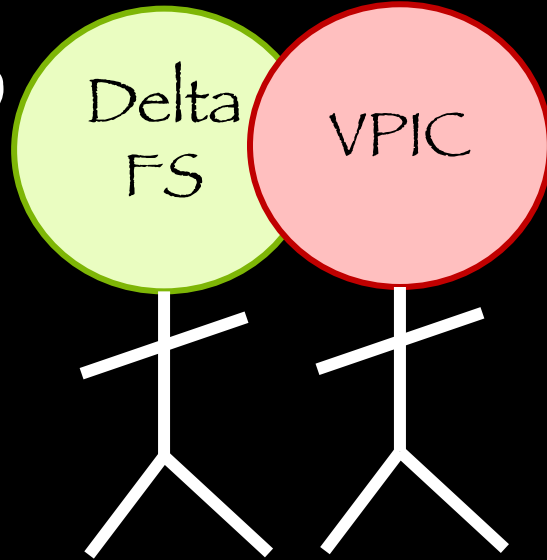
LA-UR-18-27284



Ultrascale Systems
Research Center

DeltaFS / VPIC

DeltaFS is awesome but we need an app!



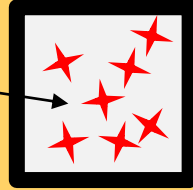
Really want to study 1 trillion particles!

What's VPIC?

Top 4 most used applications at LANL

VPIC problem
space

Particle
32 bytes – 64 bytes



Cell

Millions of particles per cell

Each process has N cells
Millions of processes.

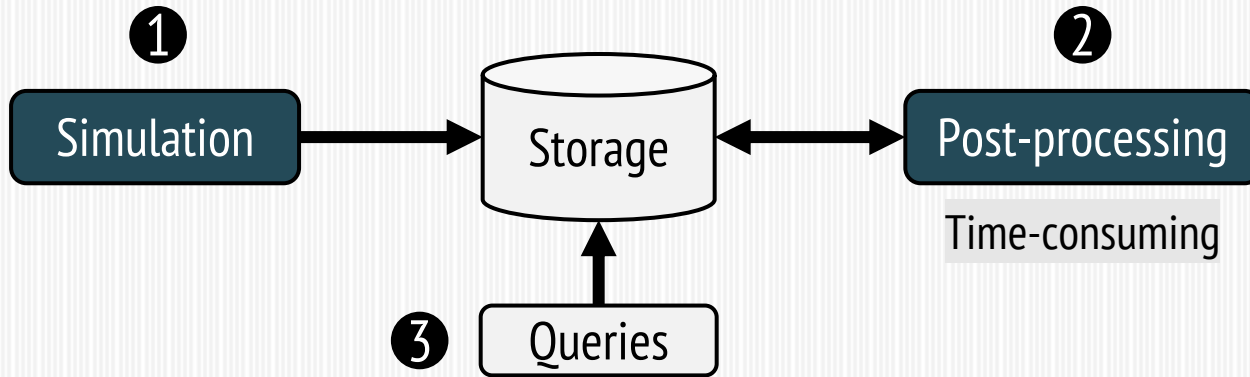
- Alternates between compute and I/O for every few timesteps
- Each timestep dump consists of the state of all particles

Need fast trajectory query over trillions of particles

★ I/O challenges

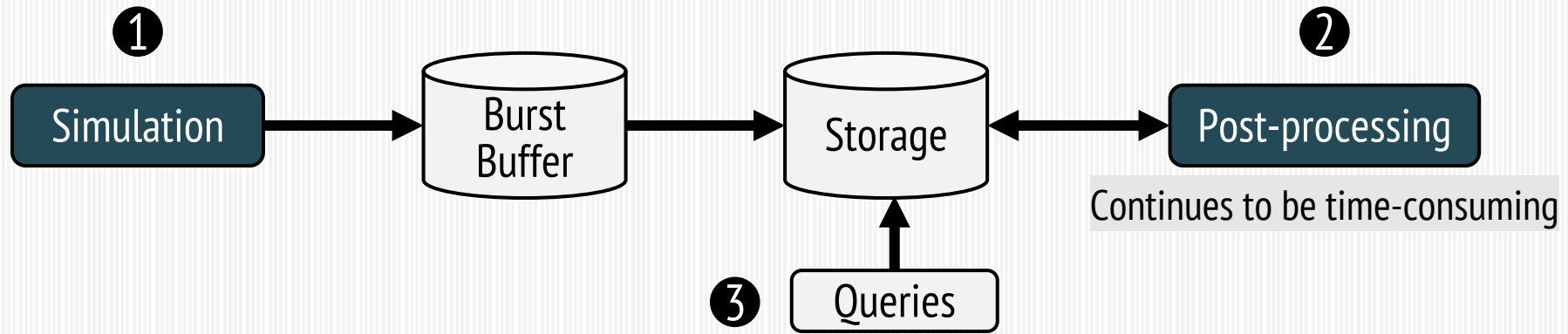
- ❑ **Write** - utilizing all available I/O bandwidth
- ❑ **Read** - fast post-analysis queries

Current state-of-the-art: Indexing during post-processing



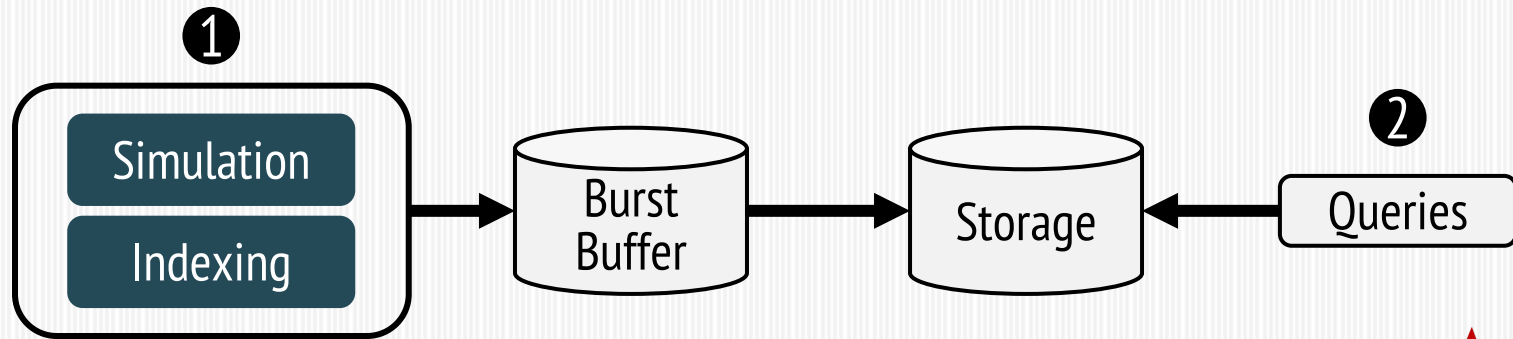
Queries run slowly until post-processing done

Fast burst-buffer storage is not going to help post-processing



Limited buffer space cannot always hold all data

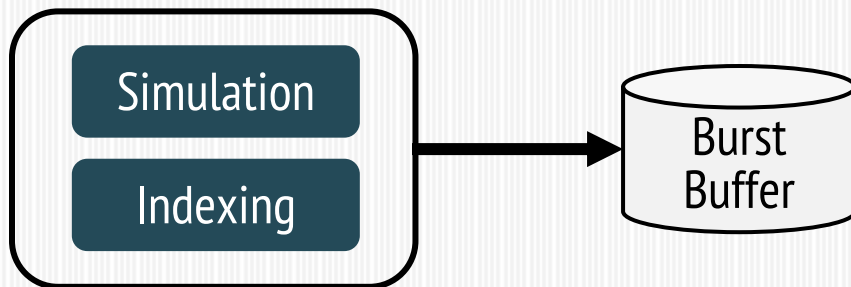
Our work is to index data as data is written to storage



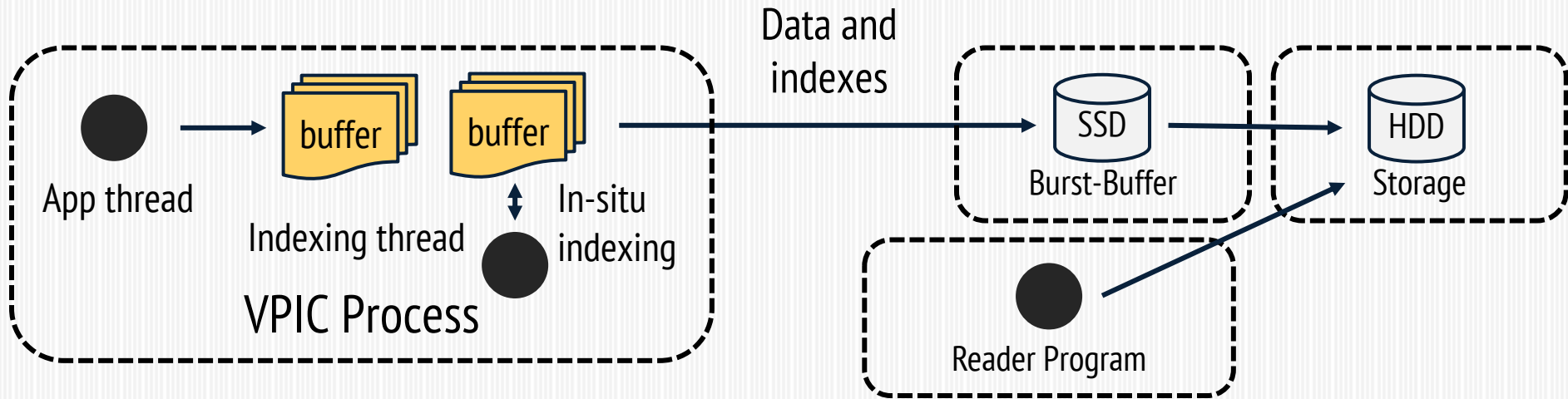
Little post-processing is needed

Key Idea

Reuse idle computing resources during simulation I/O for in-situ indexing



Initially, it works like this...



An example of our output

```
# ls /users/qingzhen/jobs/deltafs.51165/deltafs_P16384M_C1024_N32/out/particle
L-0001.dat L-0001.idx L-0002.dat L-0002.idx L-0003.dat L-0003.idx
L-0004.dat L-0004.idx L-0005.dat L-0005.idx L-0006.dat L-0006.idx
L-0007.dat L-0007.idx L-0008.dat L-0008.idx L-0009.dat L-0009.idx
...
```

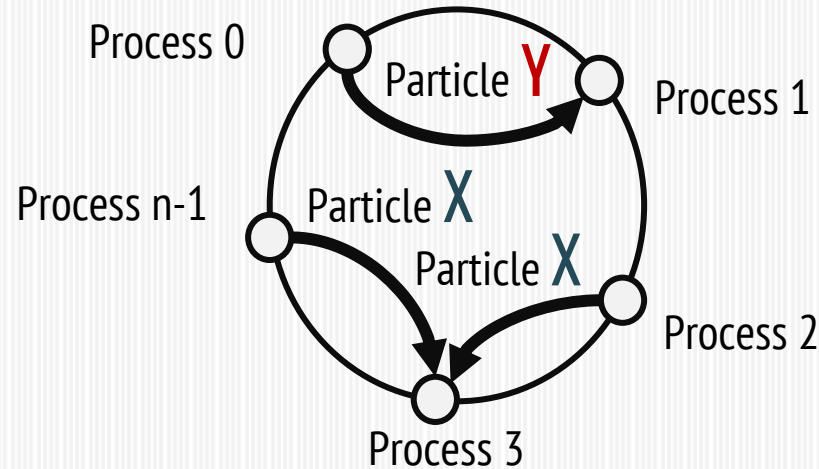


There are two problems

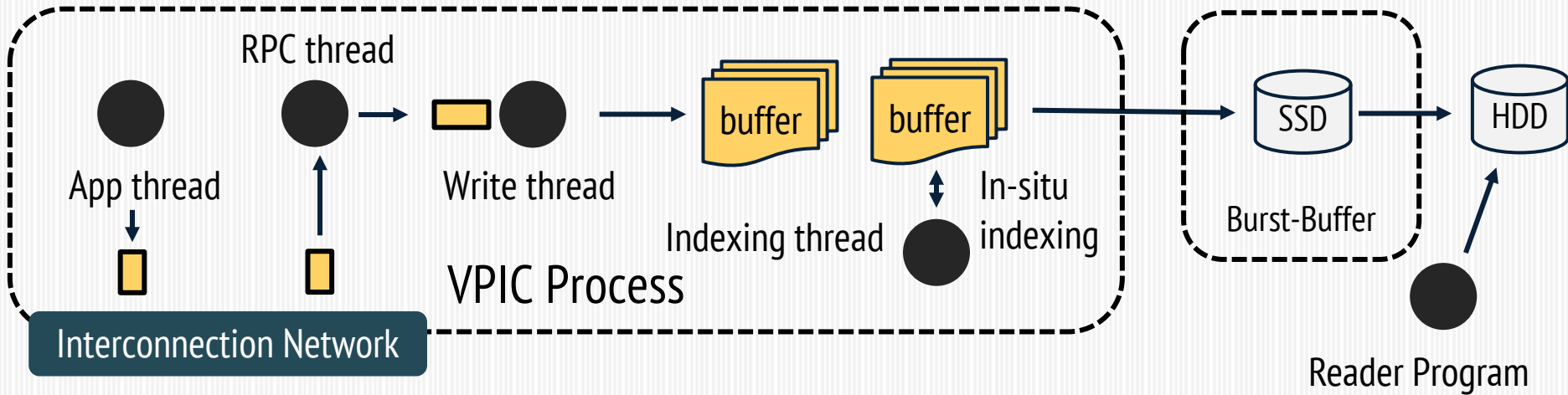
- 1) Need to read all **.idx** files to find the trajectory of any particle
- 2) Load imbalance: some **.dat** larger than others

Patch A

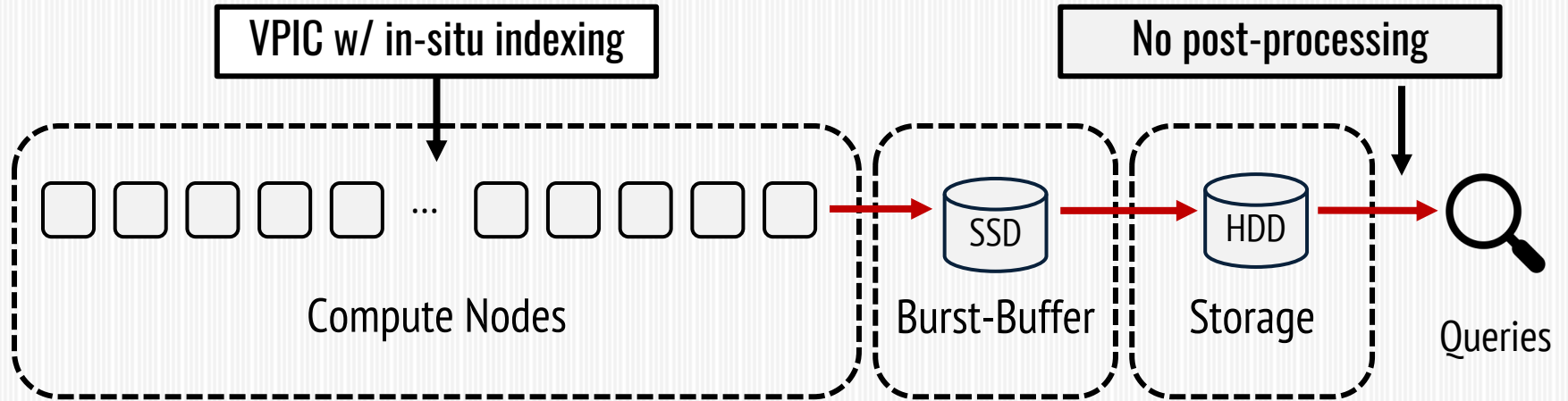
Moving particles around so data from a same particle goes to a same process



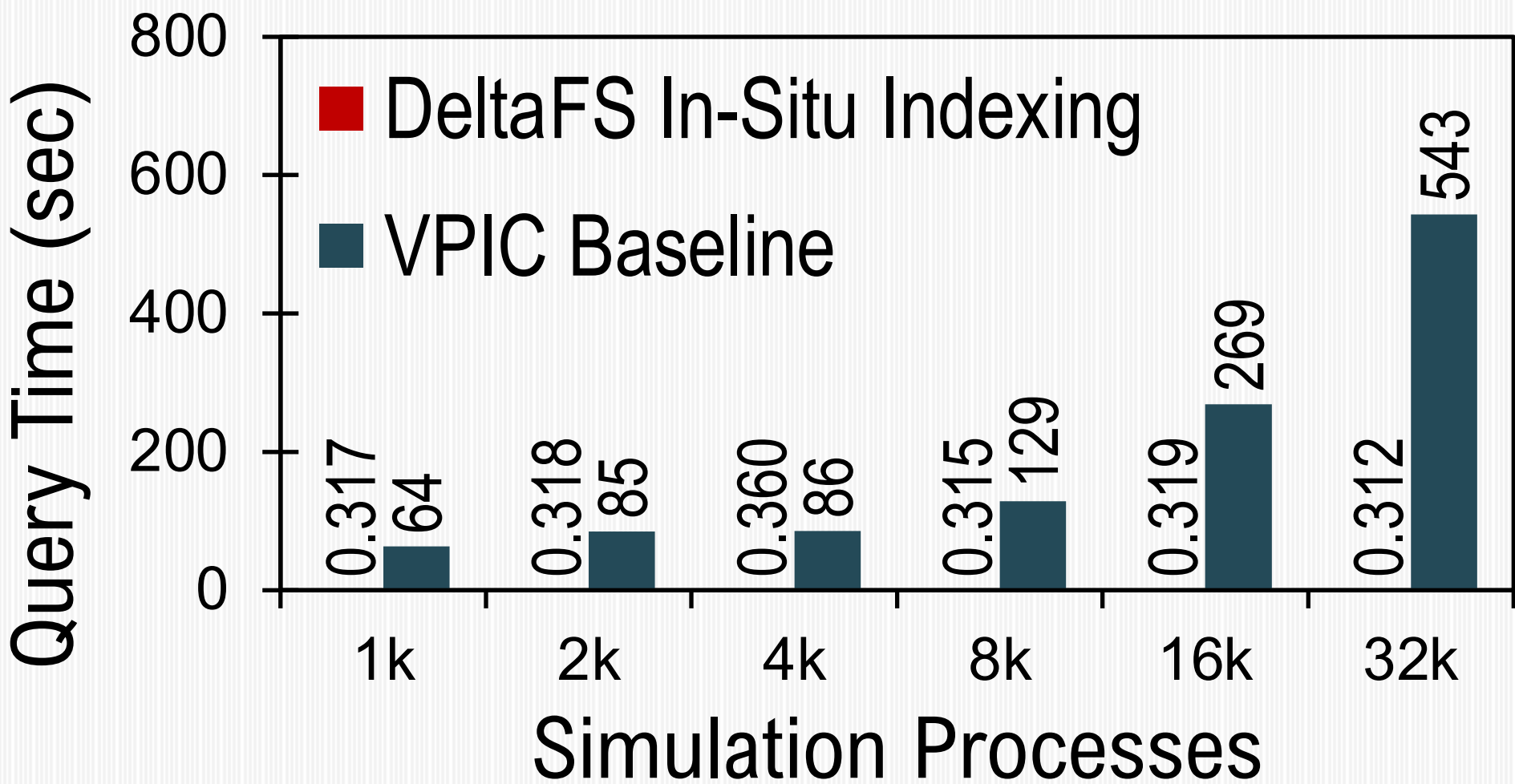
Version 2

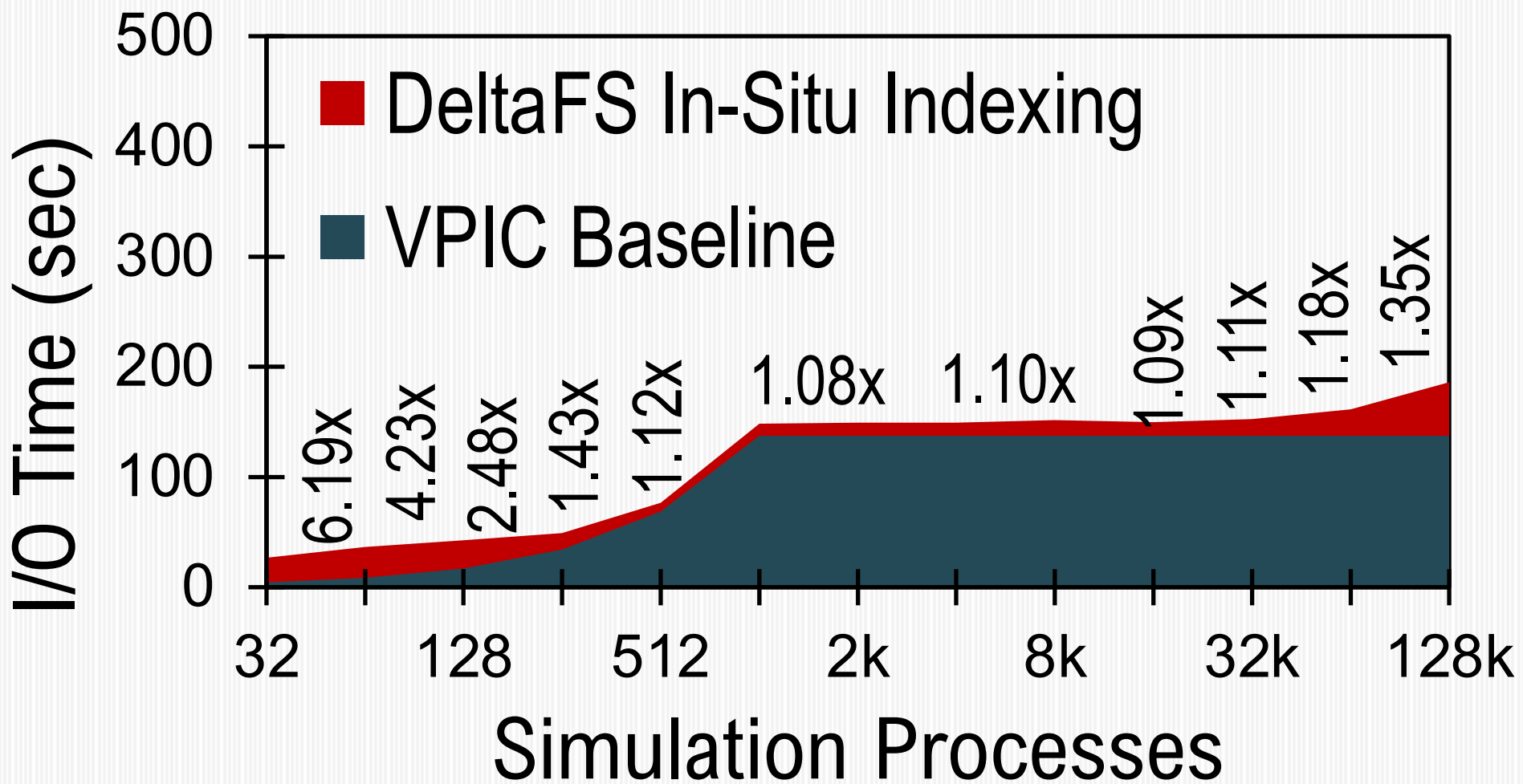


LANL Trinity Experiments



Up to **4096** compute nodes, **131,072** cores, **2** trillion particles






We work hard

	Initial code	Apr17	Jul17	Sep17	Today
I/O overhead		+64%	+35%	+15%	+10%
Max #nodes	Garbage	32	32	96	2048

Especially during this period



We have a new problem!

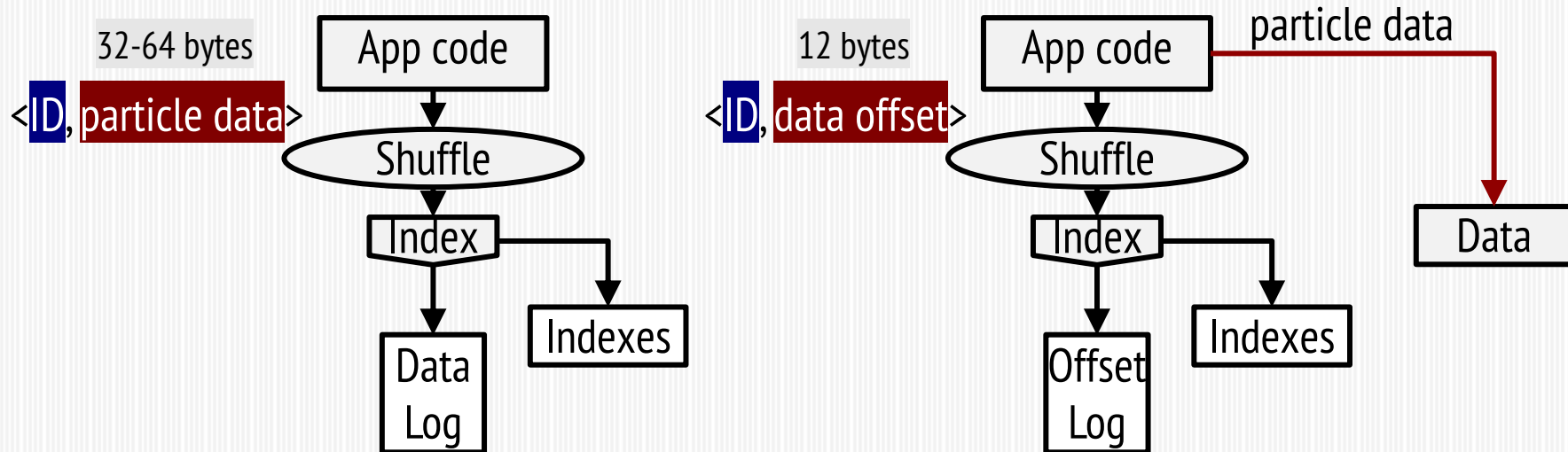
Awesome
performance
with Haswell
nodes!!

Delta
FS

Garbage on
Knights
Landing nodes

Patch B

Separate particle IDs from particle data



Conclusion

- Today's data analysis is sped up through careful post-processing
- In-situ indexing reduces post-processing and improve time-to-insight
- KNL needs more work

