

Tutorial 156 @ SC2018

# Application Porting & Optimization on GPU-accelerated POWER Architectures

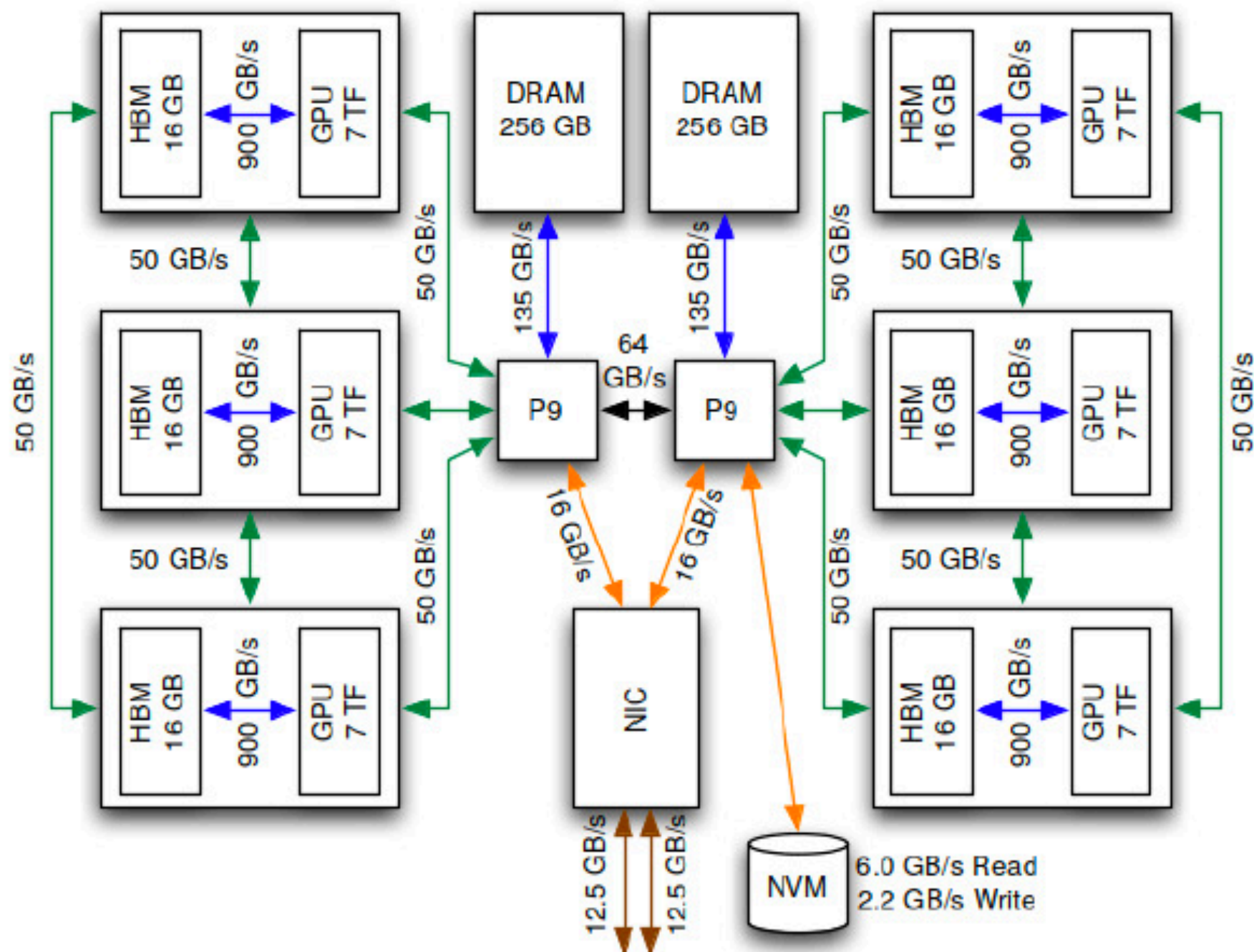
## Best practices for porting scientific applications

Christoph Hagleitner, [hle@zurich.ibm.com](mailto:hle@zurich.ibm.com)

- CPMD team (Manish Modani, Valery Weber, Teodoro Laino)
- HPCG team (Cristiano Malossi, Panos Chatzidoukas)
- SnapML team (Haris Pozidis, Thomas Parnell, Celestine Dünner, Dimitrios Sarigiannis, Nikolas Ioannou, Andreea Anghel)
- Many more working on next-gen tooling

- Recall: openPOWER for HPC - differentiating features
- Porting a complex application: CPMD
- Porting a scalable benchmark: HPCG
- Porting a cloud benchmark: SNAP-ML
- HPC application porting: Trends
  - Libraries
  - Containerization
  - Jupyter

# AC922: IBM POWER9 for HPC



- ↔ HBM/DRAM Bus (aggregate B/W)
- ↔ NVLINK
- ↔ X-Bus (SMP)
- ↔ PCIe Gen4
- ↔ EDR IB

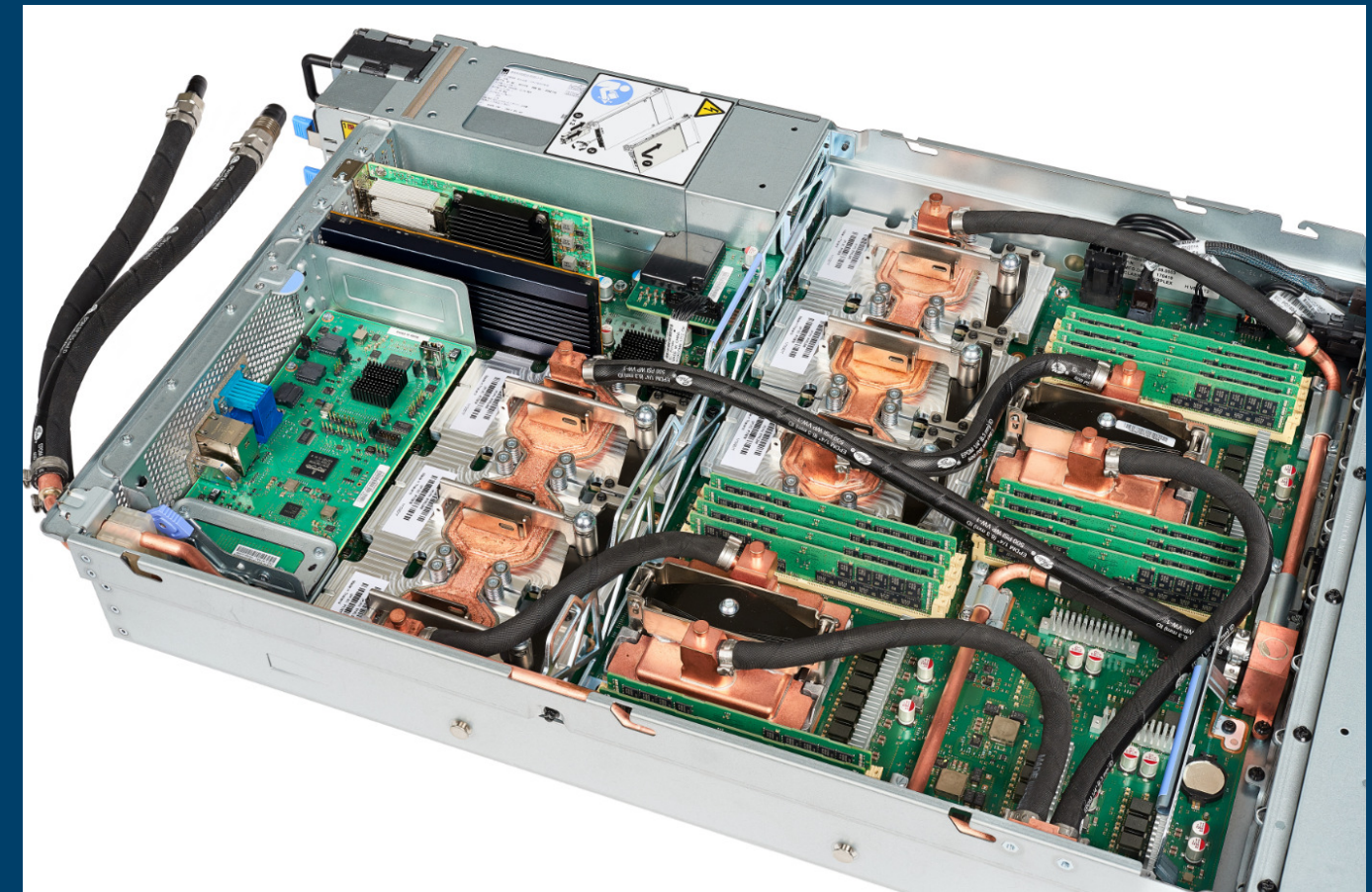
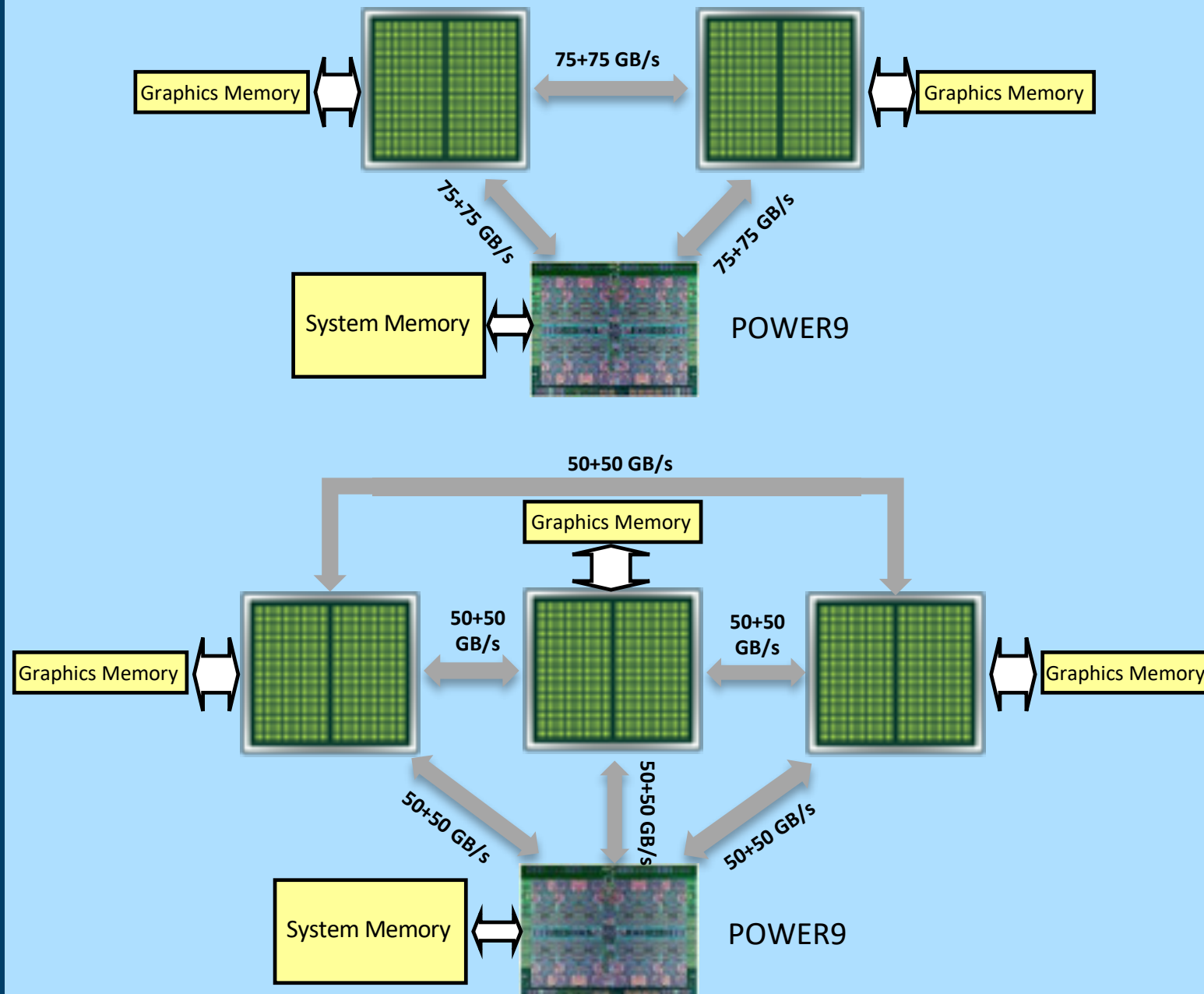
TF	42 TF (6x7 TF)
HBM	96 GB (6x16 GB)
DRAM	512 GB (2x16x16 GB)
NET	25 GB/s (2x12.5 GB/s)
MMsg/s	83

HBM & DRAM speeds are aggregate (Read+Write).  
All other speeds (X-Bus, NVLink, PCIe, IB) are bi-directional.



# AC922: GPU options

NVIDIA Volta GPU with NVLink 2.0





# AC922 w/ 4 GPUs



## PCIe slot (4x)

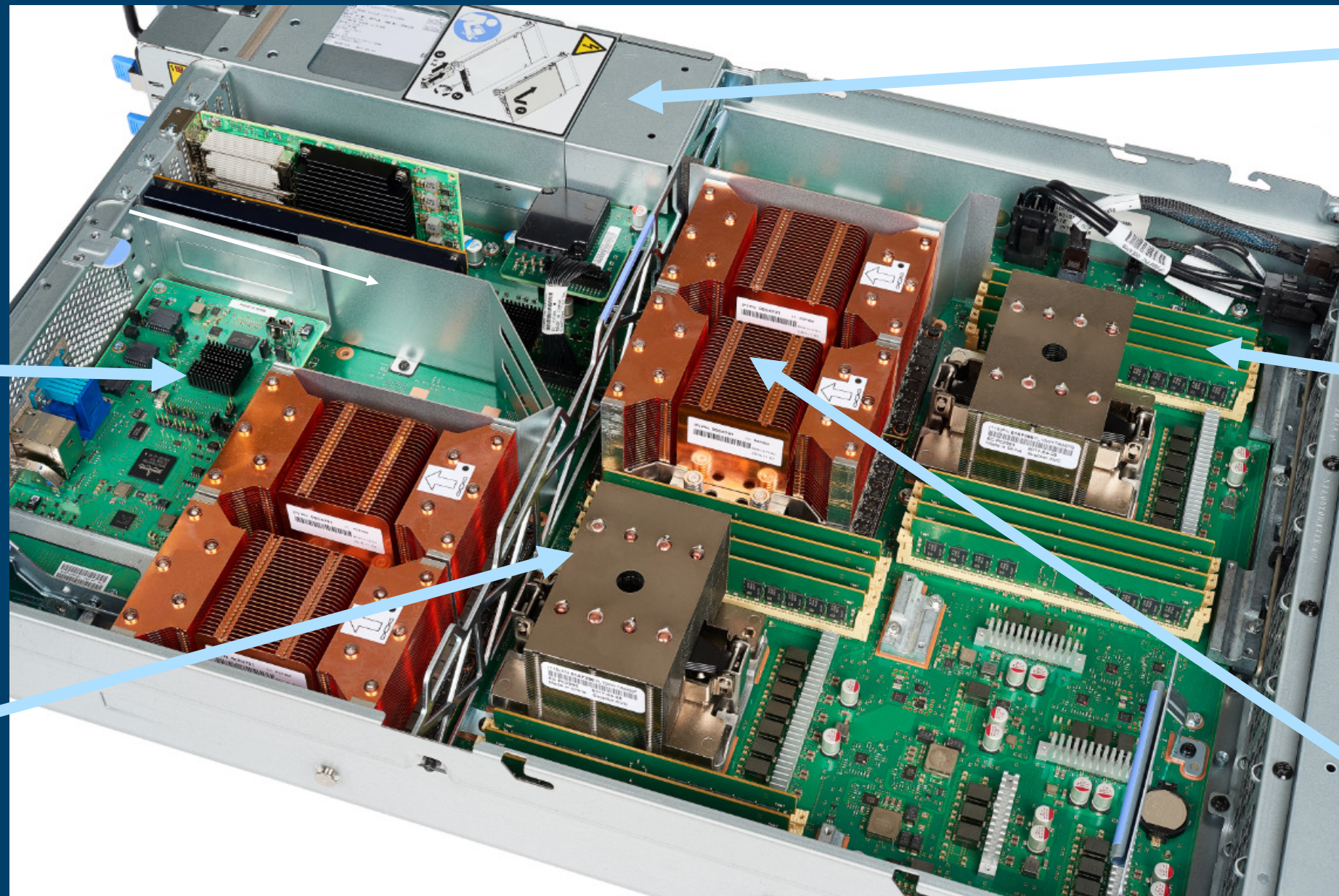
- Gen4 PCIe
- 2, x16 HHL Adapter
- 1, Shared slot
- 1 x8 HHL Adapter

## BMC Card

- IPMI
- 1 Gb Ethernet
- VGA
- 1 USB 3.0

## Power 9 Processor (2x)

- 18, 22C water cooled
- 16, 20C air cooled



## Power Supplies (2x)

- 2200W
- 200VAC, 277VAC, 400VDC input

## Memory DIMM's (16x)

- 8 DDR4 IS DIMMs per socket
- 8, 16, 32, 64, 128GB DIMMs

## NVidia Volta GPU

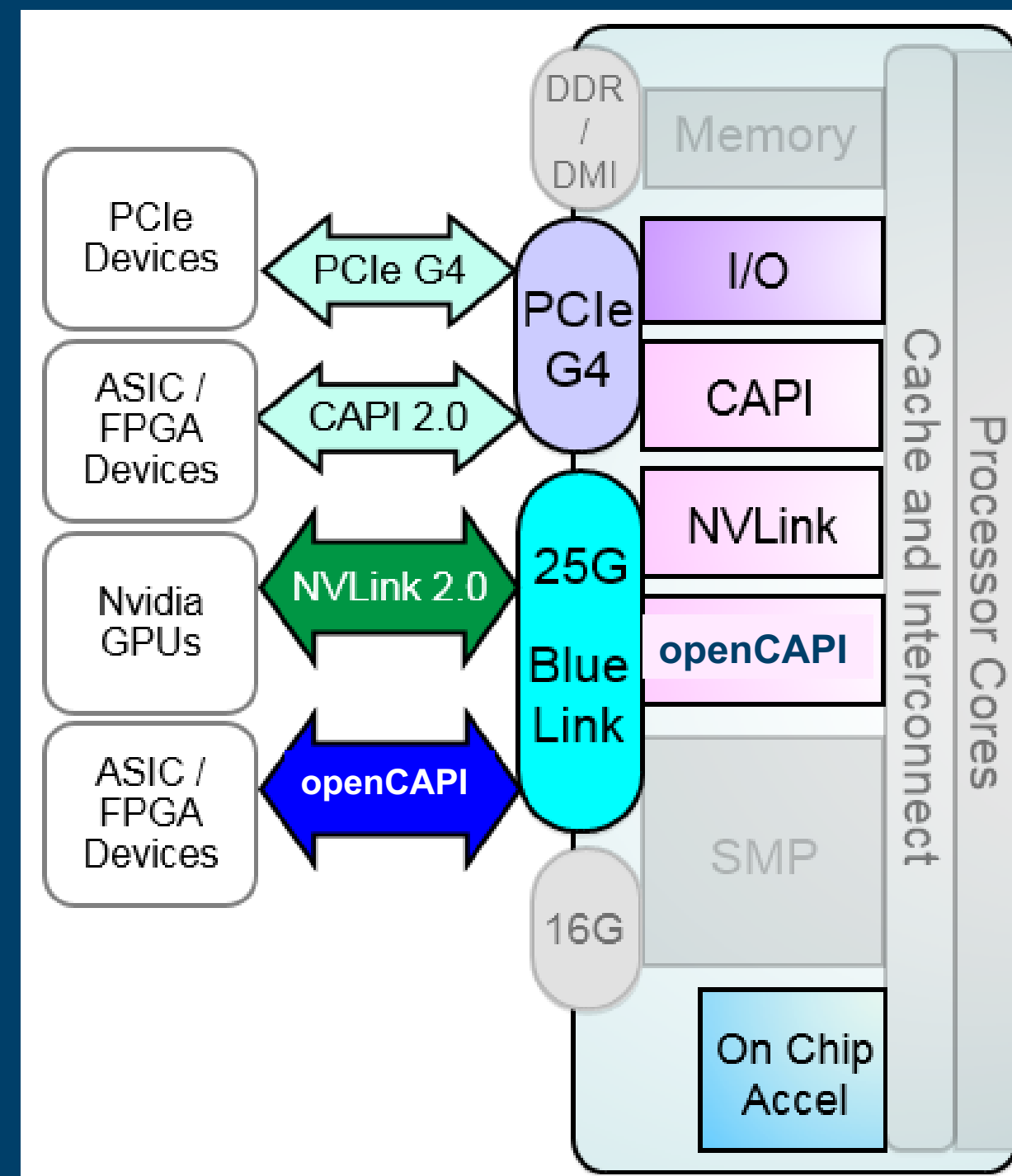
- 2 per socket
- SXM2 form factor
- 300W
- NVLink 2.0
- Air Cooled

## Extreme Accelerator Bandwidth and Reduced Latency

- PCIe Gen 4 x 48 lanes – 192 GB/s peak bandwidth (duplex)
- IBM BlueLink 25Gb/s x 48 lanes – 300 GB/s peak bandwidth (duplex)

## Coherent Memory and Virtual Addressing Capability for all Accelerators

- CAPI 2.0 - 4x bandwidth of POWER8 using PCIe Gen 4
- NVLink 2.0 – Next generation of GPU/CPU bandwidth and integration using BlueLink
- OpenCAPI – High bandwidth, low latency and open interface using BlueLink





- Recall: openPOWER for HPC - differentiating features
- Porting a complex application: CPMD
- Porting a scalable benchmark: HPCG
- Porting a cloud benchmark: SNAP-ML
- HPC application porting: Trends
  - Libraries
  - Containerization
  - Jupyter

- Heterogeneous systems (eg, CPU/GPU) are key to reach exascale
- OpenPOWER systems combining CPUs and GPUs address key issues on the road to scalable acceleration
  - Compute density
  - Data transfer density/BW
  - Coherent memory space
- Thus there is a need to port computational science codes to heterogeneous systems. This requires algorithm rethinking and code reengineering in order to fully exploit next generation of heterogeneous architectures.
- Today's showcase #1: electronic structure code CPMD

- POWER-optimized libraries & compilers

- Advanced toolchain

- [https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W51a7ffcf4dfd\\_4b40\\_9d82\\_446ebc23c550/page/IBM%20Advance%20Toolchain%20for%20PowerLinux%20Documentation](https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W51a7ffcf4dfd_4b40_9d82_446ebc23c550/page/IBM%20Advance%20Toolchain%20for%20PowerLinux%20Documentation)

- XL-compilers

- <https://www.ibm.com/developerworks/community/groups/community/xlpower/>

- ESSL

- <https://www-03.ibm.com/systems/power/software/essl/>

- GPU optimization

- CUDA

- CUDNN

- OpenGL

- PowerAI



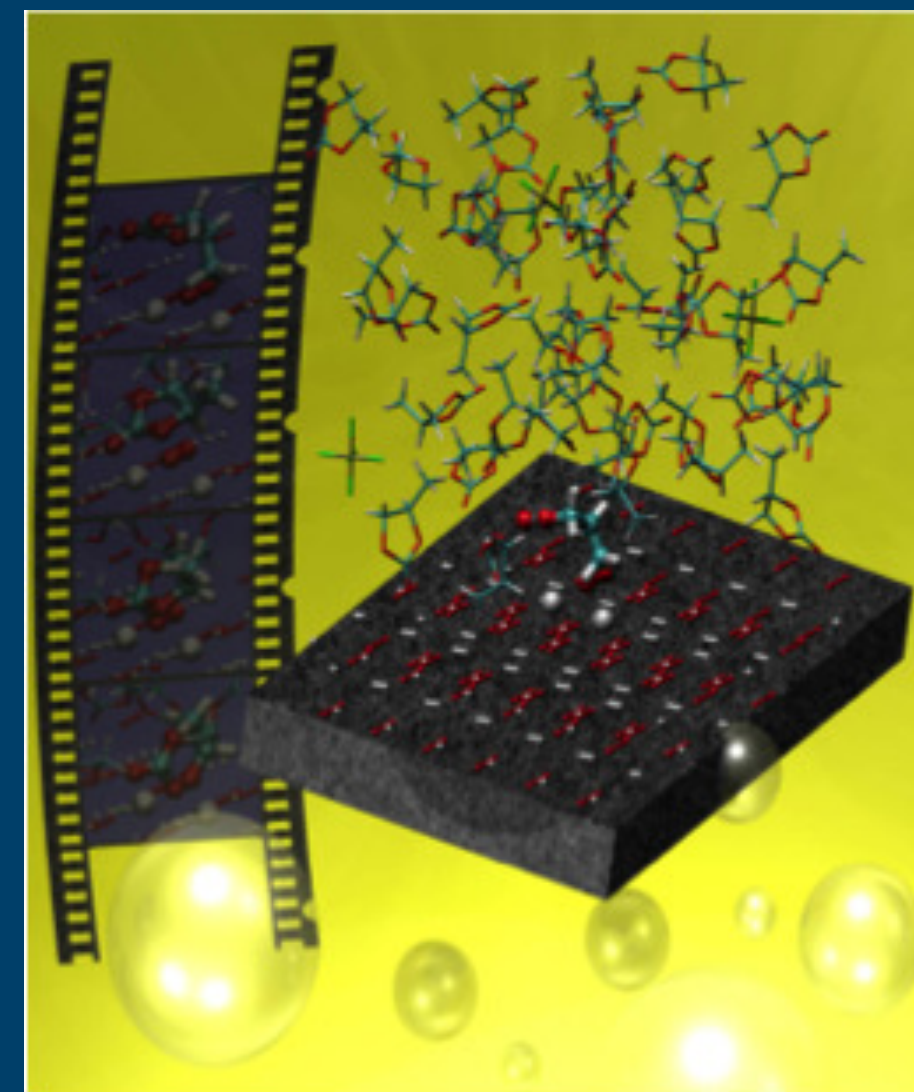
- (open)POWER for HPC: differentiating features
- Porting a complex application: CPMD
  - Introduction
  - Refactoring the code
  - Compiling the code
  - Assigning

## AI / Machine Learning

- Dense Storage
- Conclusion

# Car–Parrinello Molecular Dynamics: CPMD

- Shown to scale to very large systems
- Numerous showcases, eg, Li-Air batteries



*Simulations of  $\text{Li}_2\text{O}_2$  in Propylenecarbonate,  
T. Laino, A. Curioni, A New Piece in the  
Puzzle of Lithium/Air Batteries, Chemistry,  
DOI 10.1002/chem.201103057 (22 February  
2012)*

Observation:

- Each iteration step require at least  $N \times 3D$  FFT (inverse/forward)

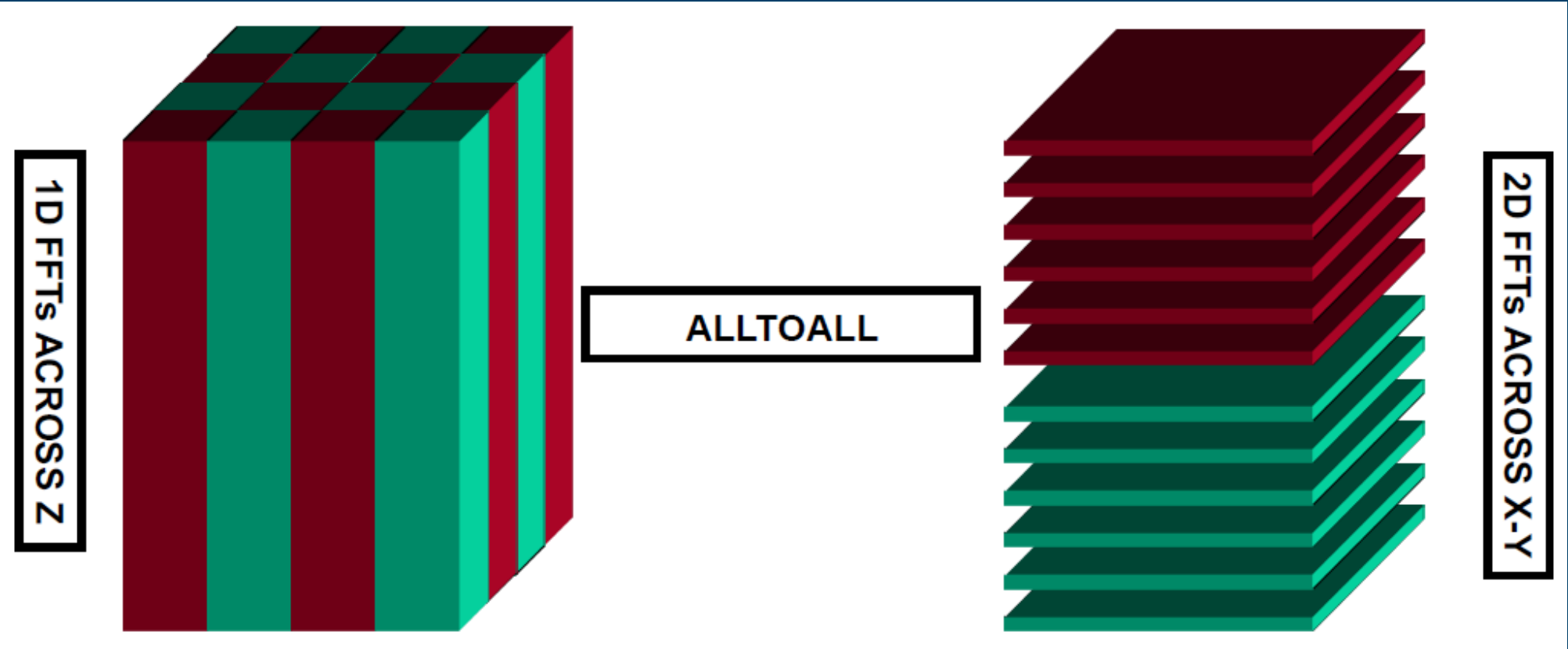
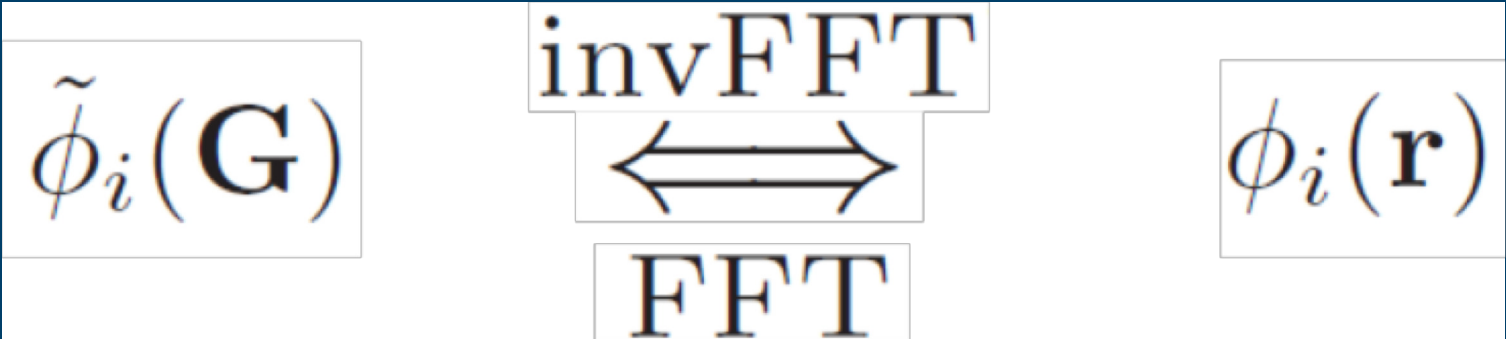
We focused on:

- Construction of the electronic density
- Applying the potential to the wavefunctions
- Orthogonalization of the wavefunctions

$$\rho(\mathbf{r}) = \sum_i^N |\phi_i(\mathbf{r})|^2$$

$$\left[ -\frac{1}{2} \nabla_i^2 + V_{\text{eff}}[\rho] \right] \phi_i(\mathbf{r}) = \epsilon_i \phi_i(\mathbf{r}),$$

$$\int \phi_i(\mathbf{r}) \phi_j(\mathbf{r}) d^3r = \delta_{ij}$$



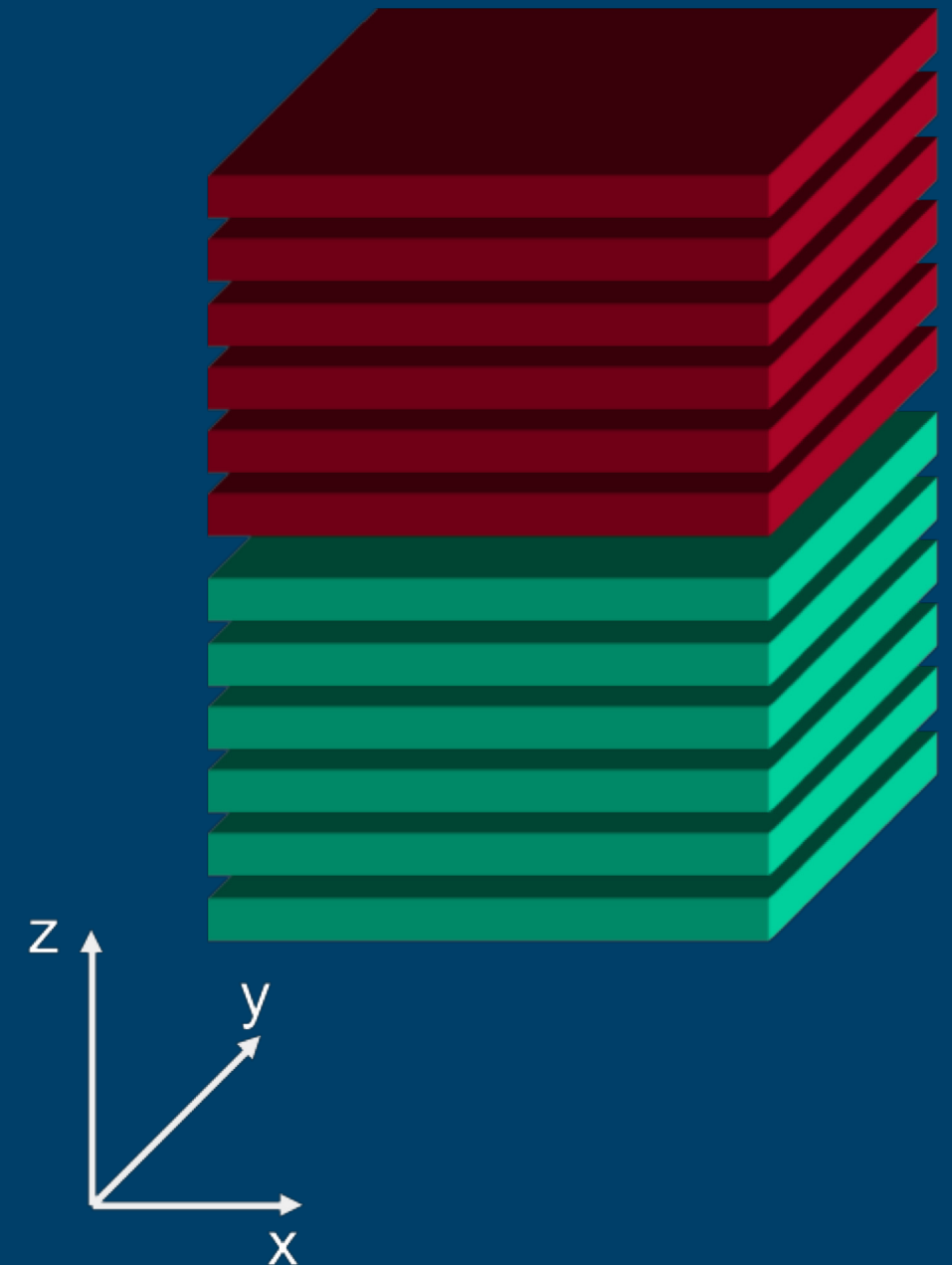
Each processor takes a number of whole planes ...

... very good scheme for small – medium sized computational platforms

... but observe that scalability is limited by the number of planes across the Z-direction!

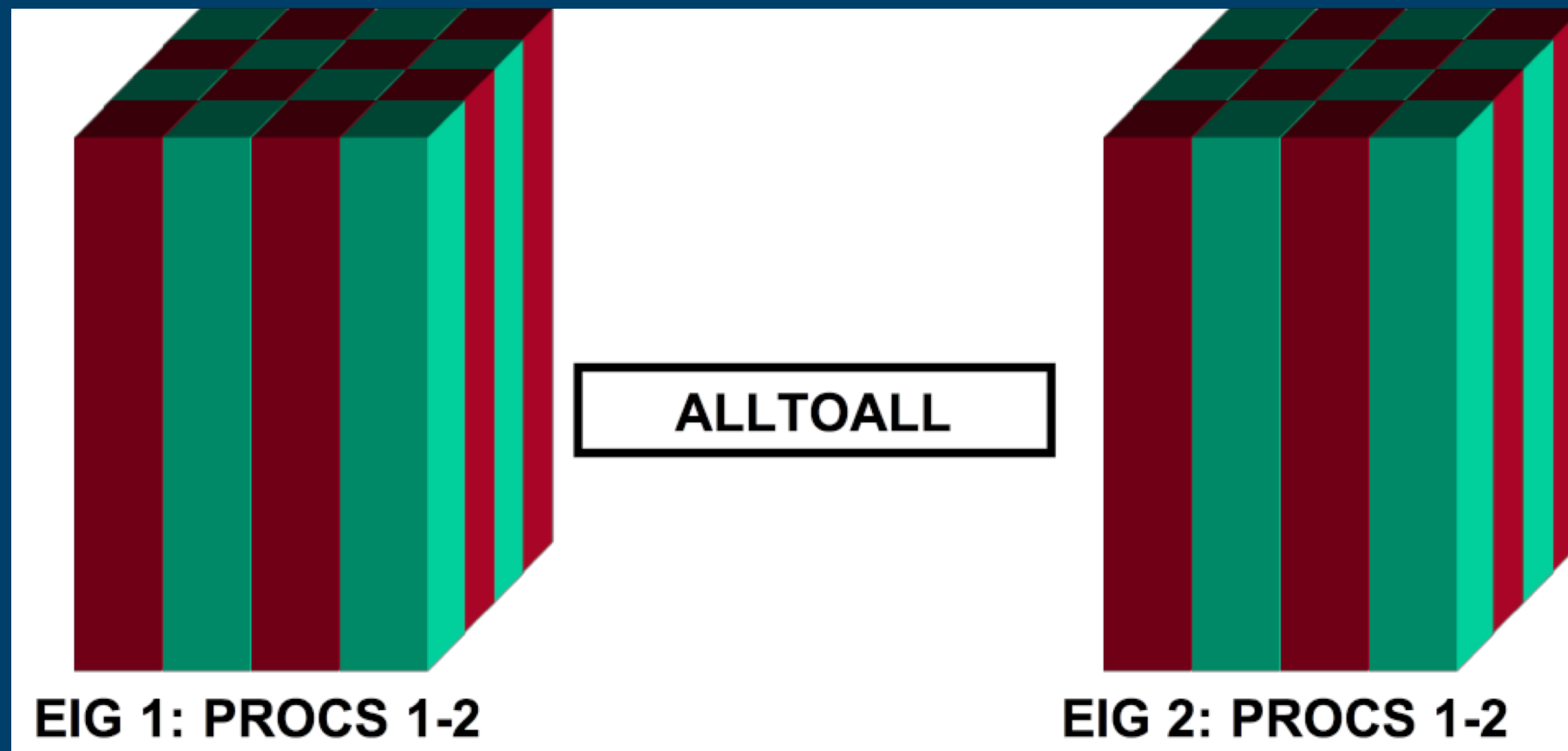
... which is in the order of a few hundred

Thus: not appropriate for a massively parallel system



$$\rho(r) = \sum_{occ} |\psi_i(r)|^2$$

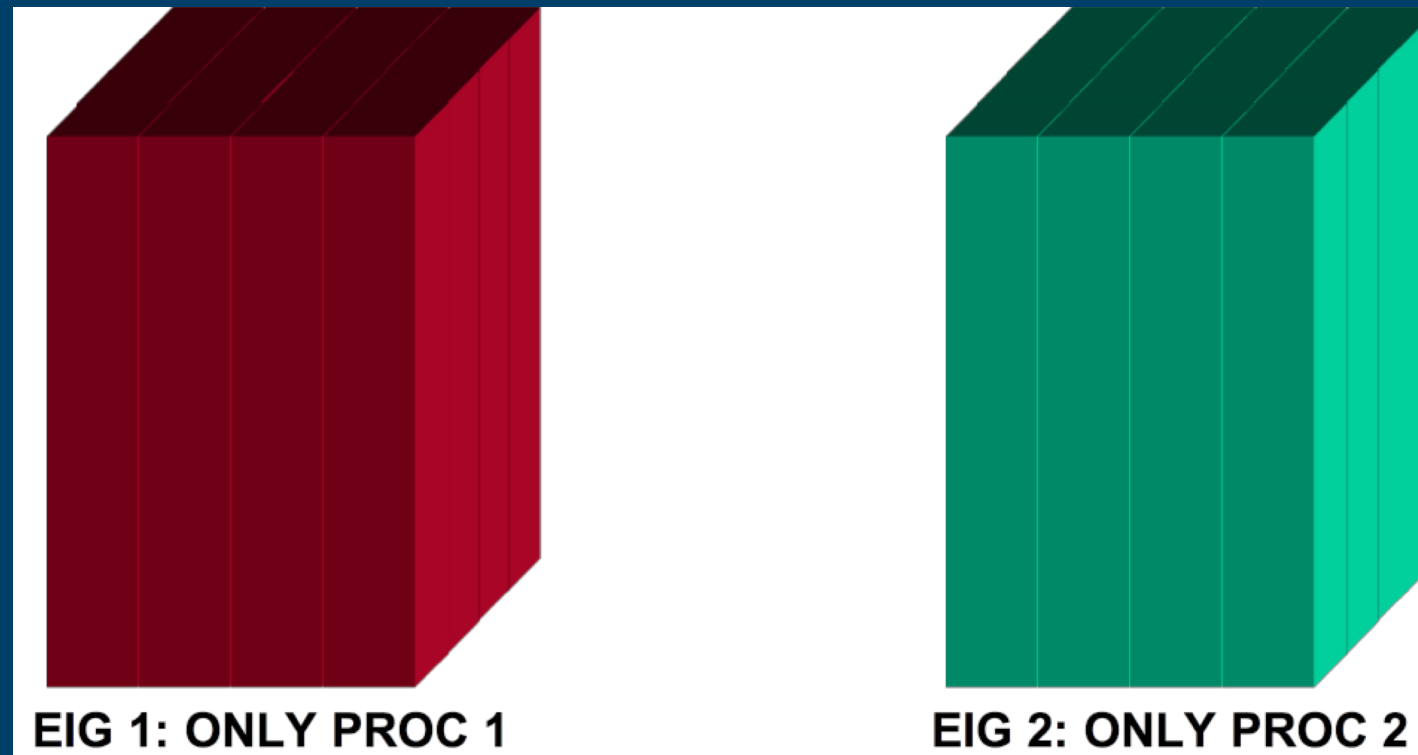
- Loop across the number of electrons. Each iteration requires 1 3D FFT.
- Hierarchical parallelism\*: Assign to each Task Group a number of iterations





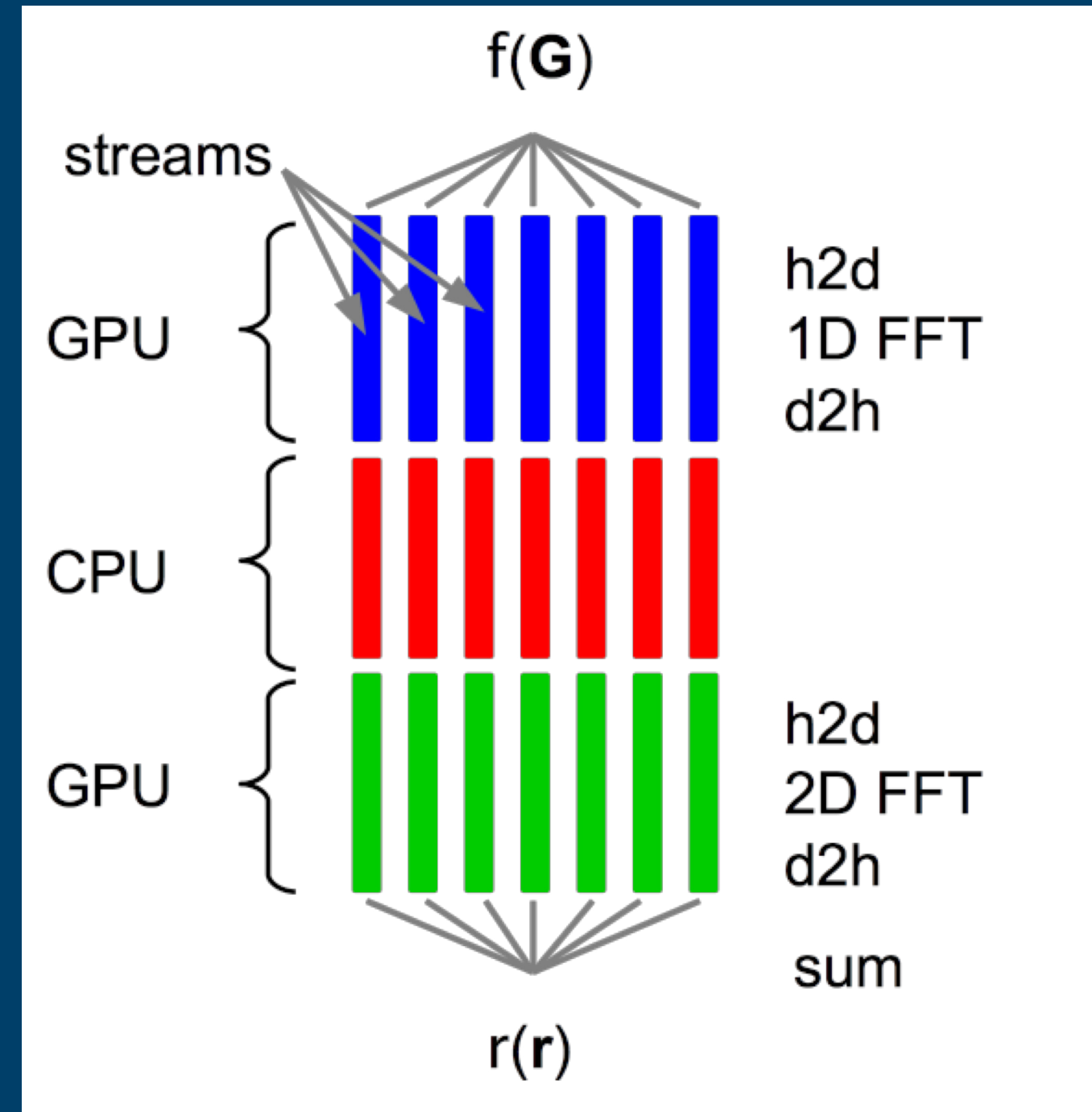
# 3D FFTs Using Task Groups

- task groups of processors will work on different eigenstates concurrently
- number of processors per group: Ideally the one that achieves the best scalability for the original parallel 3D FFT scheme



$$\phi_i(\mathbf{r}) = \text{invFFT}(\tilde{\phi}_i(\mathbf{G}))$$
$$\rho(\mathbf{r}) = \sum_i^N |\phi_i(\mathbf{r})|^2$$

- The reverse Fourier transform of the N states  $\phi(\mathbf{G})$  is distributed over the NS streams that work concurrently.
- Each stream is assigned to a CPU thread.
- Each stream transforms a state  $\phi(\mathbf{G})$  to the corresponding density (1D FFT – all2all – 2D FFT)



- The reverse and forward Fourier transforms as well as the application of the potential  $V$  to the  $N$  states are distributed over  $NS$  streams that work concurrently.
- Each stream is assigned to a CPU thread.
- Each stream transforms a state  $\phi(\mathbf{G})$  to  $\phi(\mathbf{r})$  (1D FFT – all2all – 2D FFT). The potential is applied and the result back transformed (2D FFT – all2all – 1D FFT).

$$\phi_i(\mathbf{r}) = \text{invFFT}(\tilde{\phi}_i(\mathbf{G}))$$

$$V(\mathbf{r})\phi_i(\mathbf{r})$$

$$(\widetilde{V\phi_i})(\mathbf{G}) = \text{FFT}((V\phi_i)(\mathbf{r}))$$

- we seek the orthogonalized coefficient matrix
- the coefficients of the expansion of  $\varphi(\mathbf{G})$  on the plane-wave basis is block-partitioned column-wise into  $n$  blocks of size  $b$ .
- the block Gram–Schmidt scheme loops over the  $n$  blocks  $C_i$  and orthogonalizes them one after the other

$$\tilde{C} = \text{ortho}(C)$$

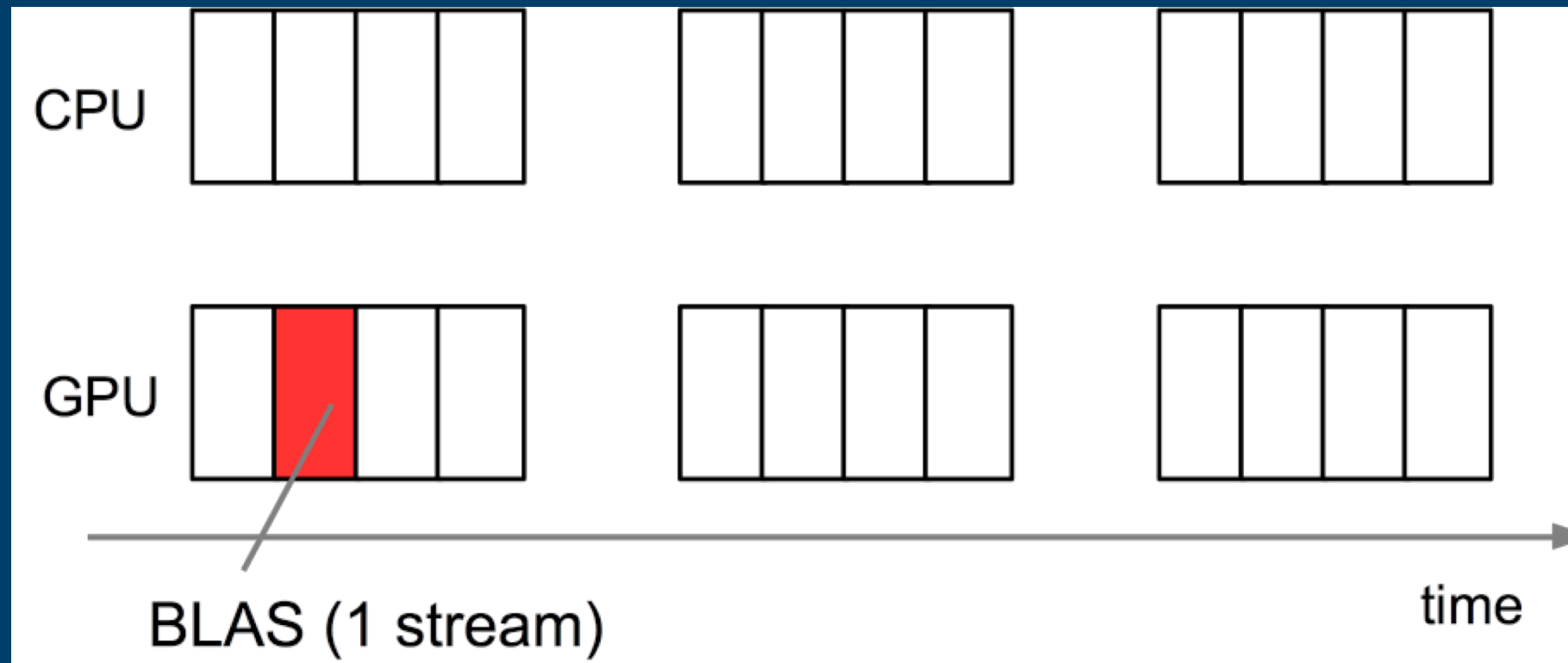
$$C = [C_1, C_2, \dots, C_n]$$

$$[\tilde{C}_1, \dots, \tilde{C}_{i-1}, C_i, \dots, C_n]$$

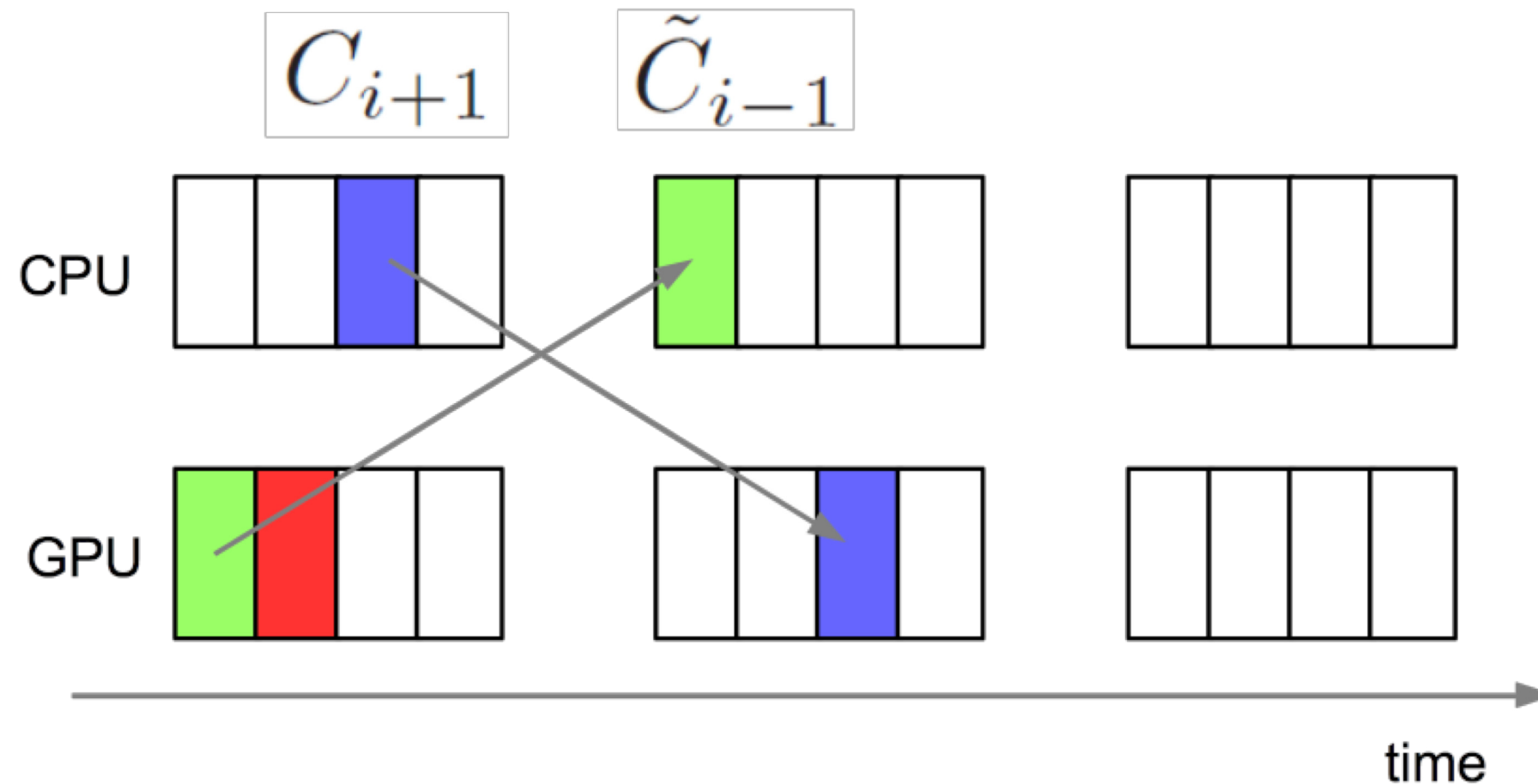
$$\tilde{C}_i = \text{ortho}((I - \sum_{j=1}^{i-1} \tilde{C}_j \tilde{C}_j^T) C_i)$$

$$[\tilde{C}_1, \dots, \tilde{C}_{i-1}, C_i, \dots, C_n]$$

$$\tilde{C}_i = \text{ortho}((I - \sum_{j=1}^{i-1} \tilde{C}_j \tilde{C}_j^T) C_i)$$



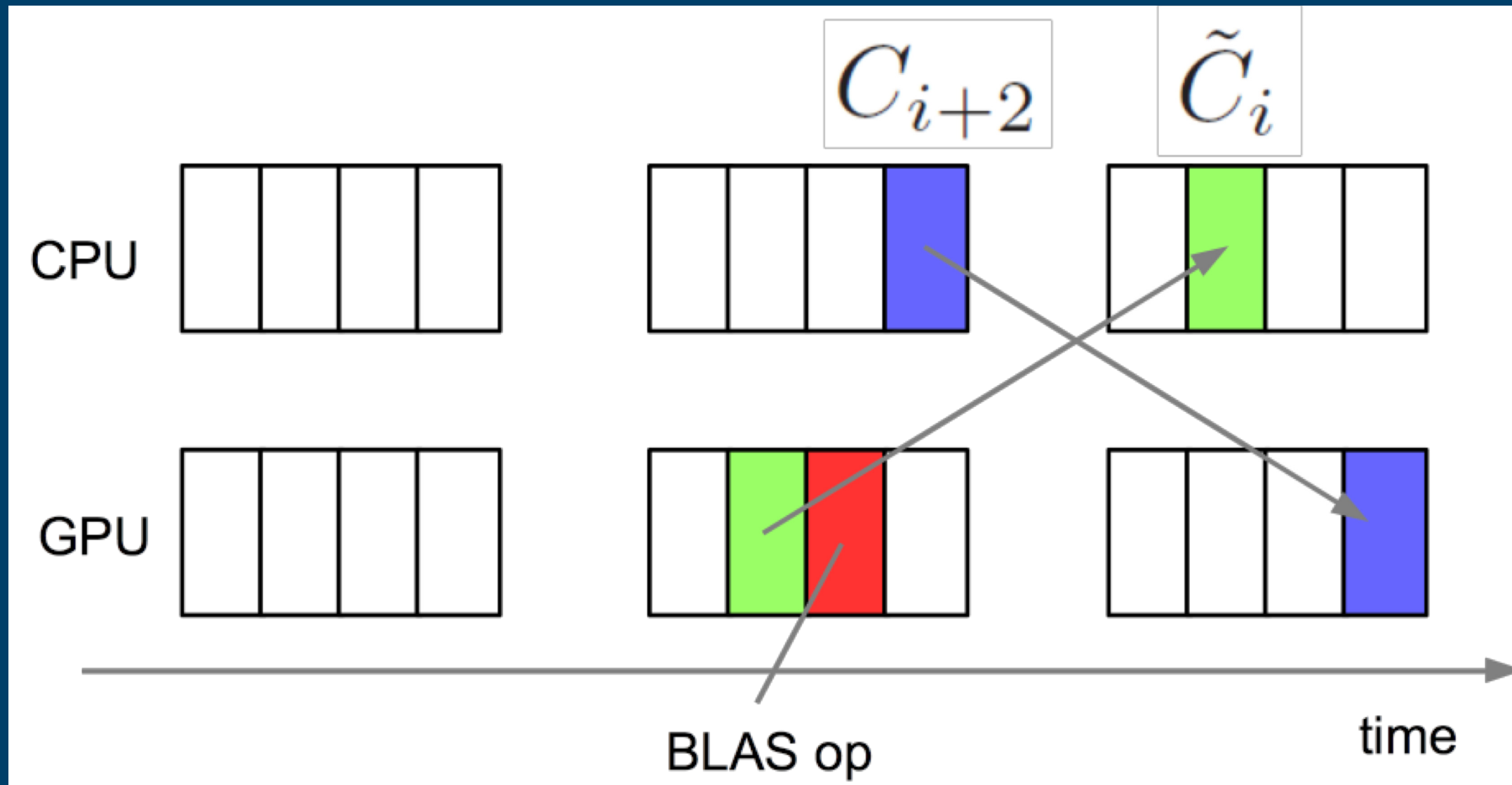
Two streams take care of D2H and H2D communication, respectively.





# GPU Porting: Orthogonalization via block Gram-Schmidt

$$\tilde{C}_{i+1} = \text{ortho}((I - \sum_{j=1}^i \tilde{C}_j \tilde{C}_j^T) C_{i+1})$$

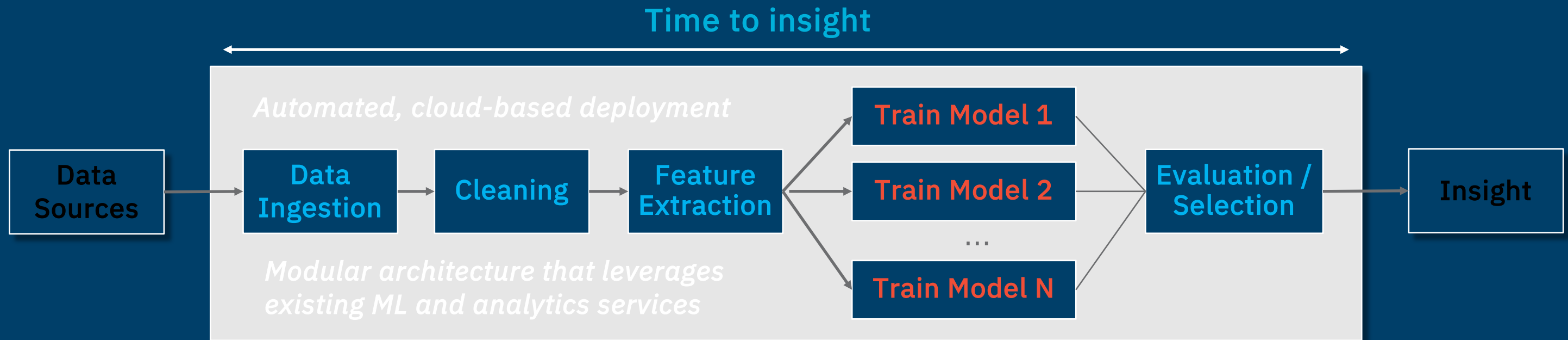


- Recall: openPOWER for HPC - differentiating features
- Porting a complex application: CPMD
- Porting a scalable benchmark: HPCG
- Porting a cloud benchmark: SNAP-ML
- HPC application porting: Trends
  - Libraries
  - Containerization
  - Jupyter

- [hpcg-benchmark.org](http://hpcg-benchmark.org)
- High Performance Conjugate Gradient (HPCG).
- Solves  $Ax=b$ ,  $A$  large, sparse,  $b$  known,  $x$  computed.
- An optimized implementation of PCG contains essential computational and communication patterns that are prevalent in a variety of methods for discretization and numerical solution of PDEs
- Patterns:
  - Dense and sparse computations.
  - Dense and sparse collective.
  - Multi-scale execution of kernels via MG (truncated) V cycle.
  - Data-driven parallelism (unstructured sparse triangular solves).
- Strong verification and validation properties (via spectral properties of PCG).

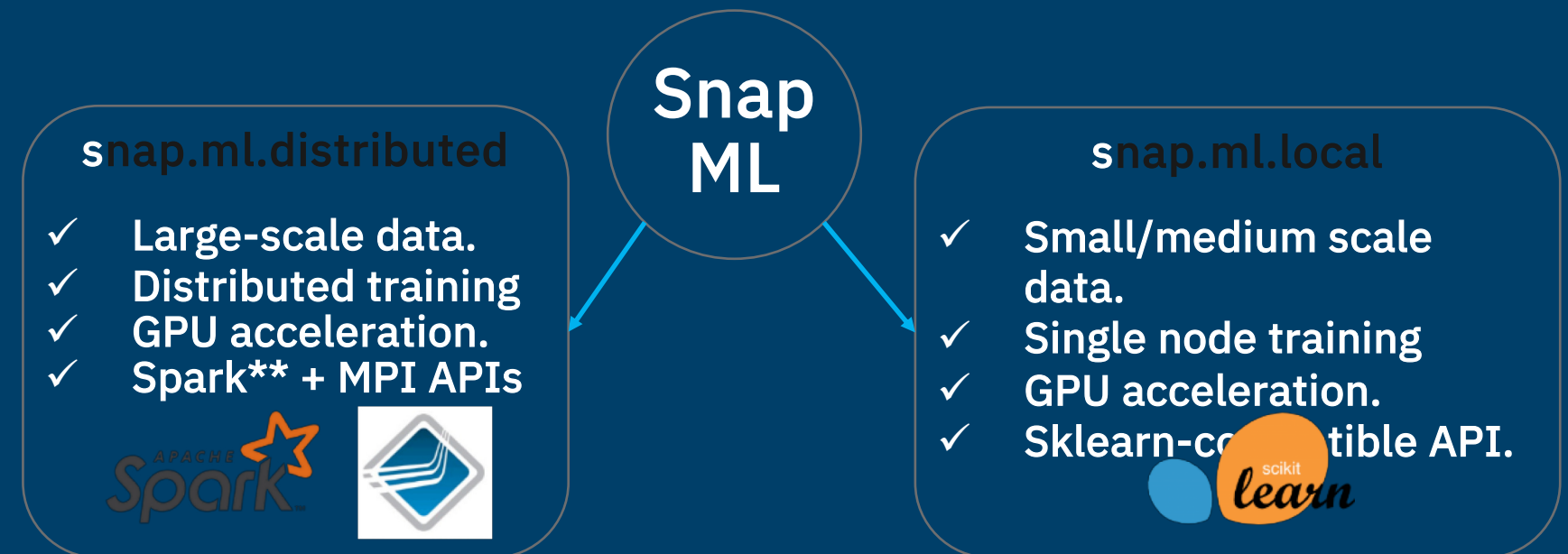


- Recall: openPOWER for HPC - differentiating features
- Porting a complex application: CPMD
- Porting a scalable benchmark: HPCG
- Porting a cloud benchmark: SNAP-ML
- HPC application porting: Trends
  - Libraries
  - Containerization
  - Jupyter



Model training time may dominate time to insight in several cases:

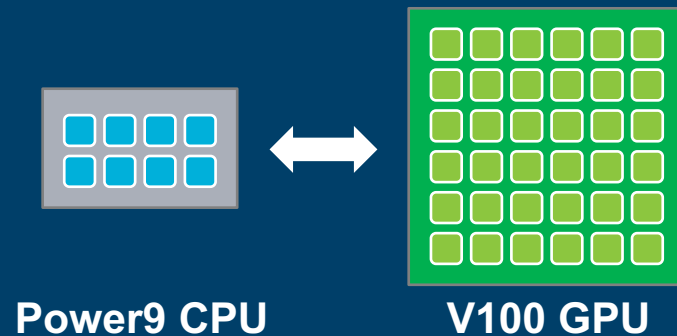
- A. **frequent re-training** is needed, to adapt to events in real time
- B. **many TB's of data** are ingested per day
- C. **large ensembles of models** are needed for best accuracy



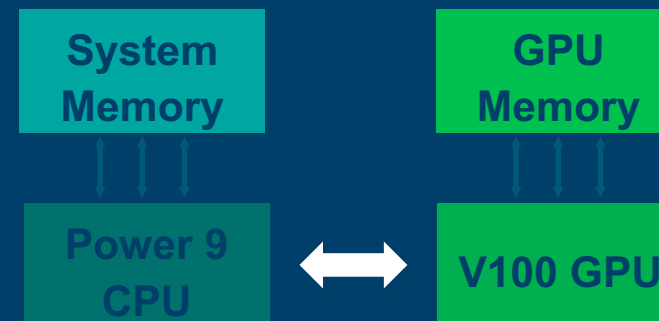


# 3 Key Breakthroughs

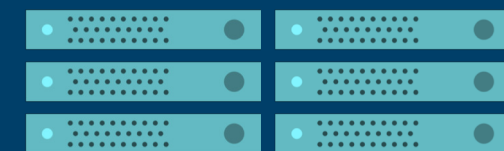
## GPU Acceleration



## Dynamic Optimized Memory Management



## Efficient Cluster Scaling



- C. Duenner, S. Forte, M. Takac, and M. Jaggi. "Primal-Dual Rates and Certificates." In *International Conference on Machine Learning (ICML 2016)*, pp. 783-792. 2016.
- T. Parnell, C. Duenner, K. Atasu, M. Sifalakis and H. Pozidis, "Large-scale stochastic learning using GPUs," *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, Lake Buena Vista, FL, 2017, pp. 419-428.
- C. Duenner, T. Parnell, K. Atasu, M. Sifalakis and H. Pozidis, "Understanding and Optimizing the Performance of Distributed Machine Learning Applications on Apache Spark", *poster presentation at NIPS 2016 ML Systems workshop, IEEE Big Data 2017*
- C. Duenner, T. Parnell, M. Jaggi, "Efficient Use of Limited-Memory Resources to Accelerate Linear Learning", *proceedings of 2017 Neural Information Processing Systems (NIPS 2017)*

## Criteo Releases Industry's Largest-Ever Dataset for Machine Learning to Academic Community

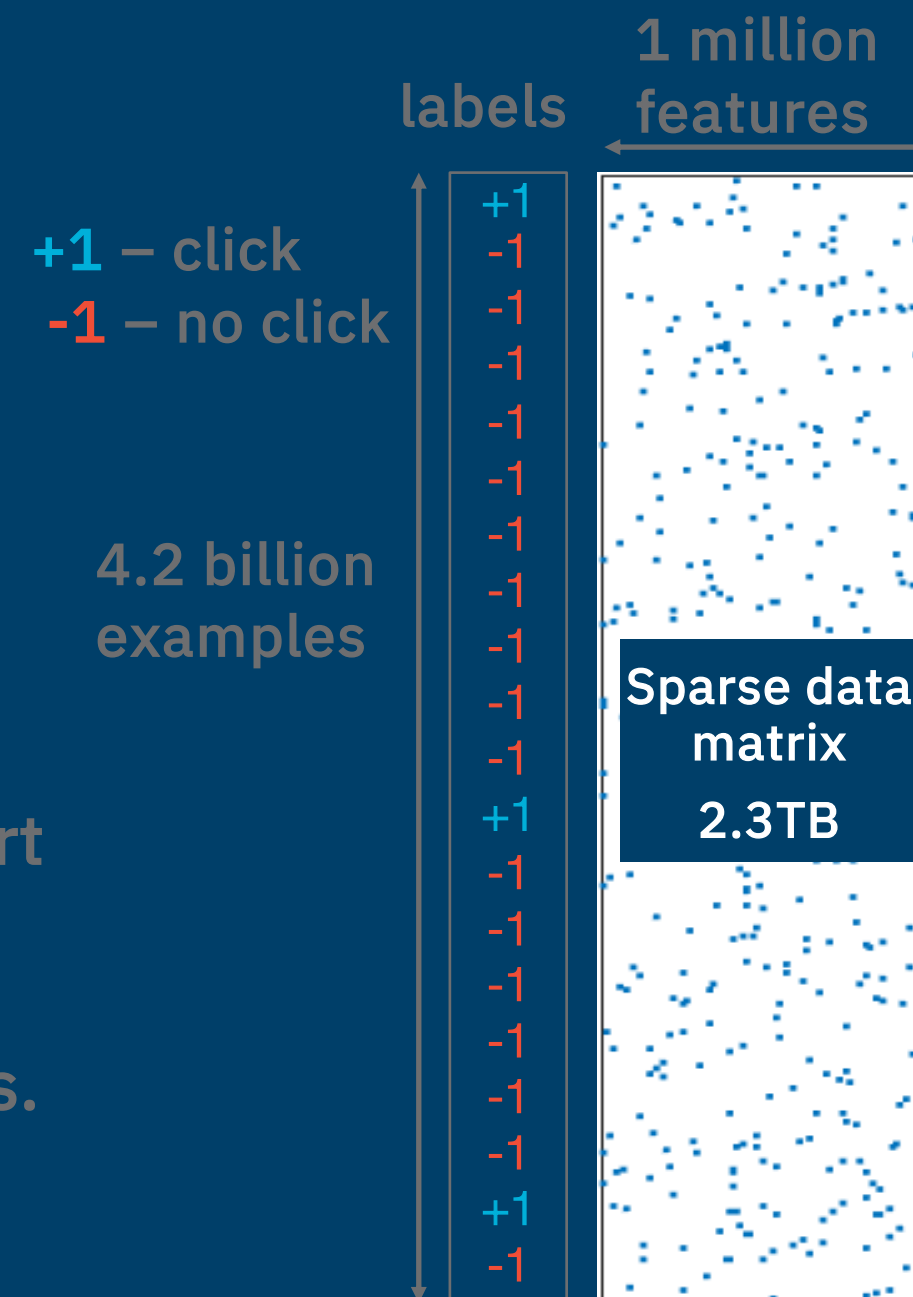
New York – June 18, 2015 – Criteo (NASDAQ: CRTO), the performance marketing technology company, today announced the release of the largest public machine learning dataset ever issued to the open source community, with the goal of supporting academic research and innovation in distributed machine learning algorithms.

\* Criteo Labs. 2015. Criteo Releases Industry's Largest-Ever Dataset for Machine Learning to Academic Community. <https://www.criteo.com/news/press-releases/2015/07/criteo-releases-industrys-largest-ever-dataset/>

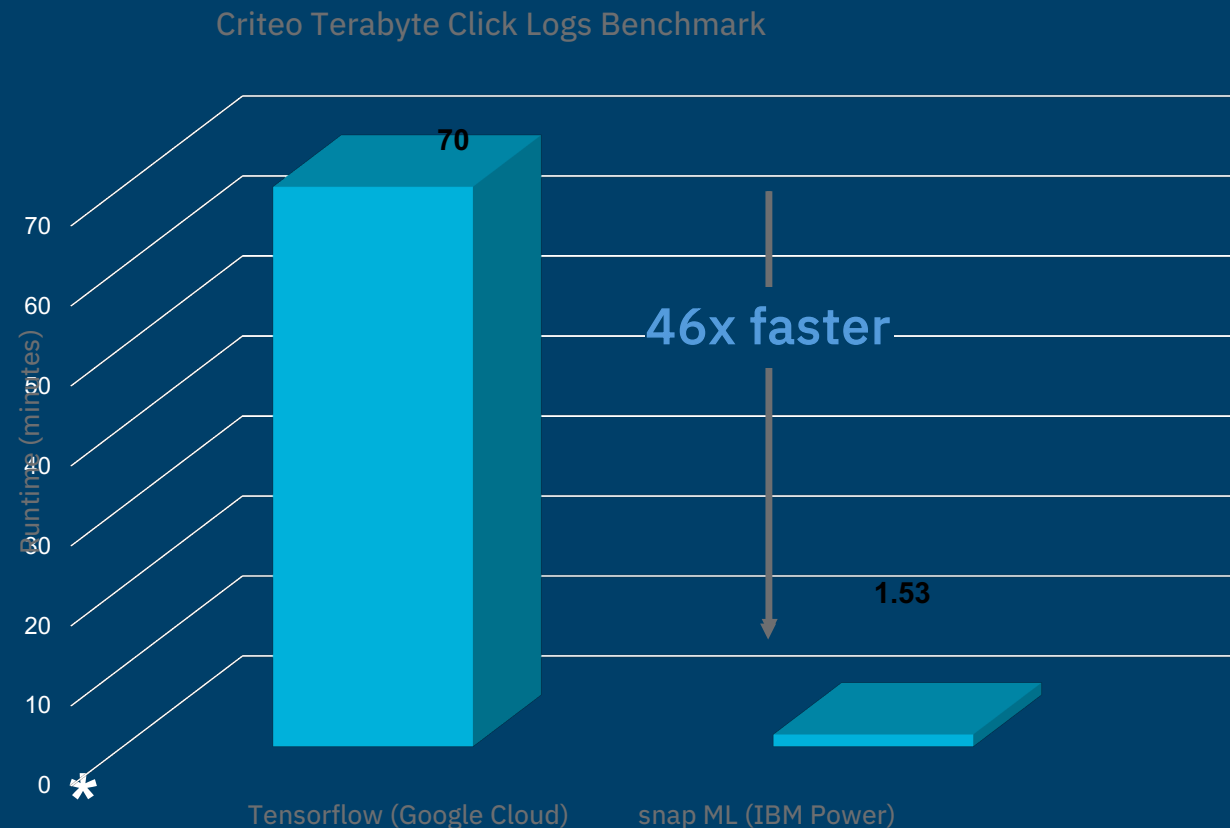
**Goal:** Predict whether a user will click on a given advert based on an anonymized set of features.

**Train:** Fit model parameters using 4.2 billion examples.

**Inference:** Evaluate model on 180 million unseen examples.



# Snap ML: Tera-scale ML benchmark



\* <https://cloud.google.com/blog/big-data/2017/02/using-google-cloud-machine-learning-to-predict-clicks-at-scale>

Comparison of Tensorflow\*\* on Google Cloud with SNAP ML on POWER9\* (AC922) cluster

Workload: Click-through-rate prediction for computational advertising, using Logistic Regression

Dataset: Criteo Terabyte Click Logs (<http://labs.criteo.com/2013/12/download-terabyte-click-logs/>)

Dataset: 4.2 billion training examples, 1 million features

Model: **Logistic Regression**

Test LogLoss: 0.1293 (Tensorflow), 0.1292 (snap ML)

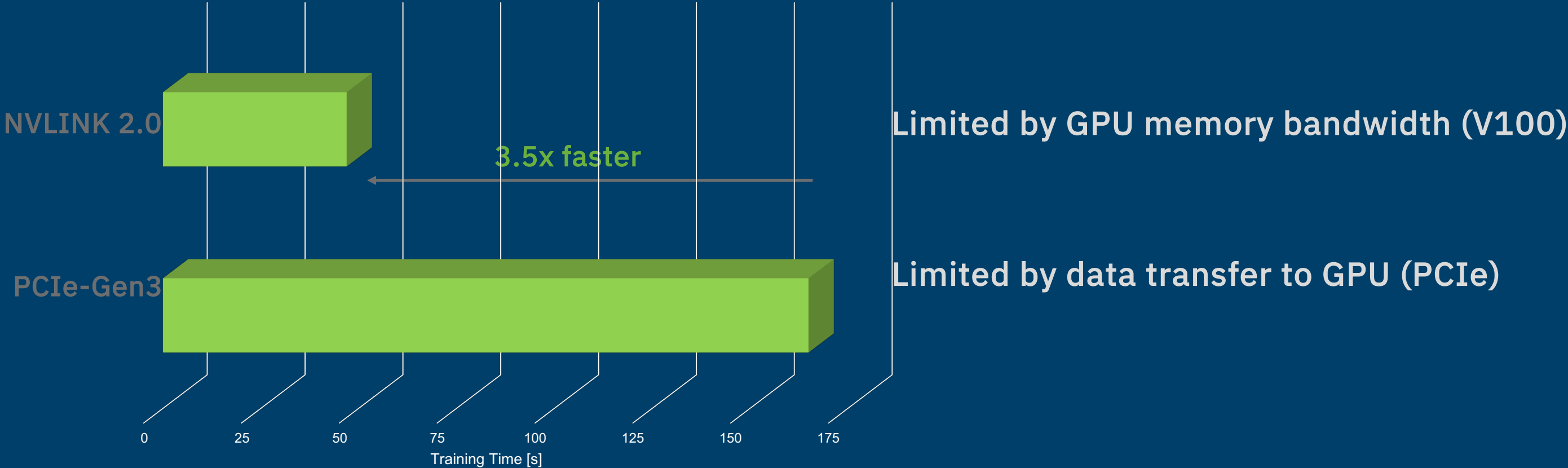
Platform: 89 machines (Tensorflow),

8 Power9 CPUs+16 NVIDIA® Tesla™ V100 GPUs (snap ML)

# Snap ML single-GPU performance

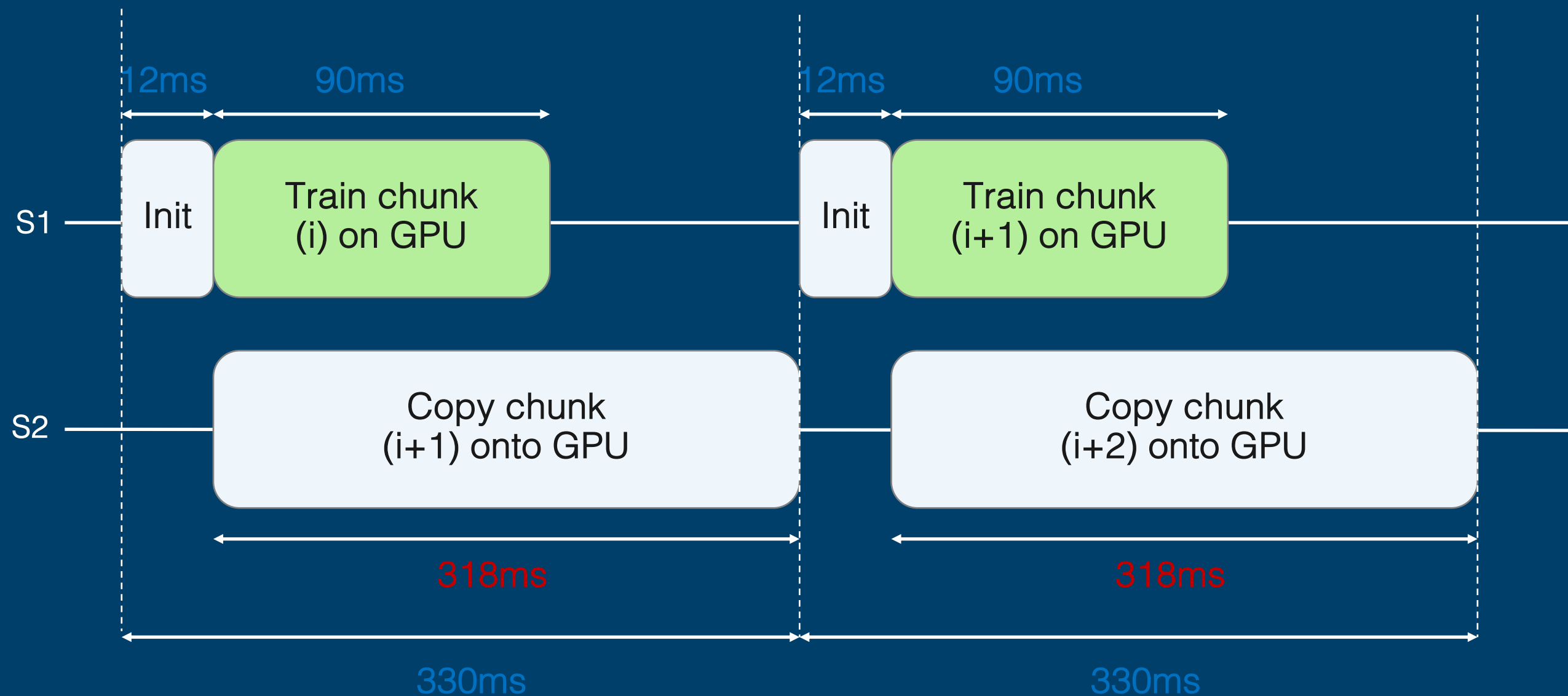


Snap ML on PCI-Gen3 vs. NVLINK CPU-GPU link



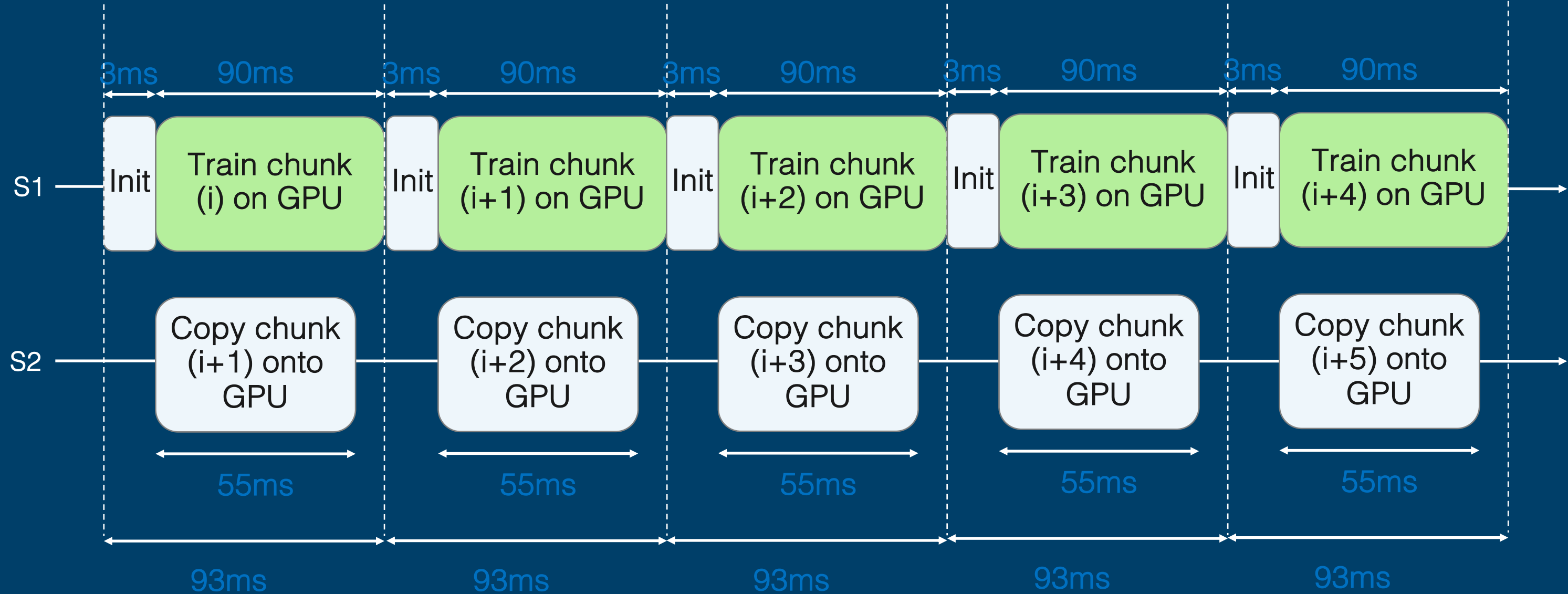
Dataset: 200 million training examples, 1 million features  
Model: **Logistic Regression**  
Test LogLoss: 0.131 (in all cases)  
Platform: Single node experiment. 1x NVIDIA Tesla V100 GPU  
PCIe-Gen3: Intel(R) Xeon(R) Gold 6150 CPU (SkyLake)  
NVLink2.0: Power9 CPU (AC922 server)

# Profile (Intel x86\*\* + Tesla™ V100 + PCIe Gen3)



**Each iteration takes 330ms and the bottleneck is the time to copy next chunk onto GPU**

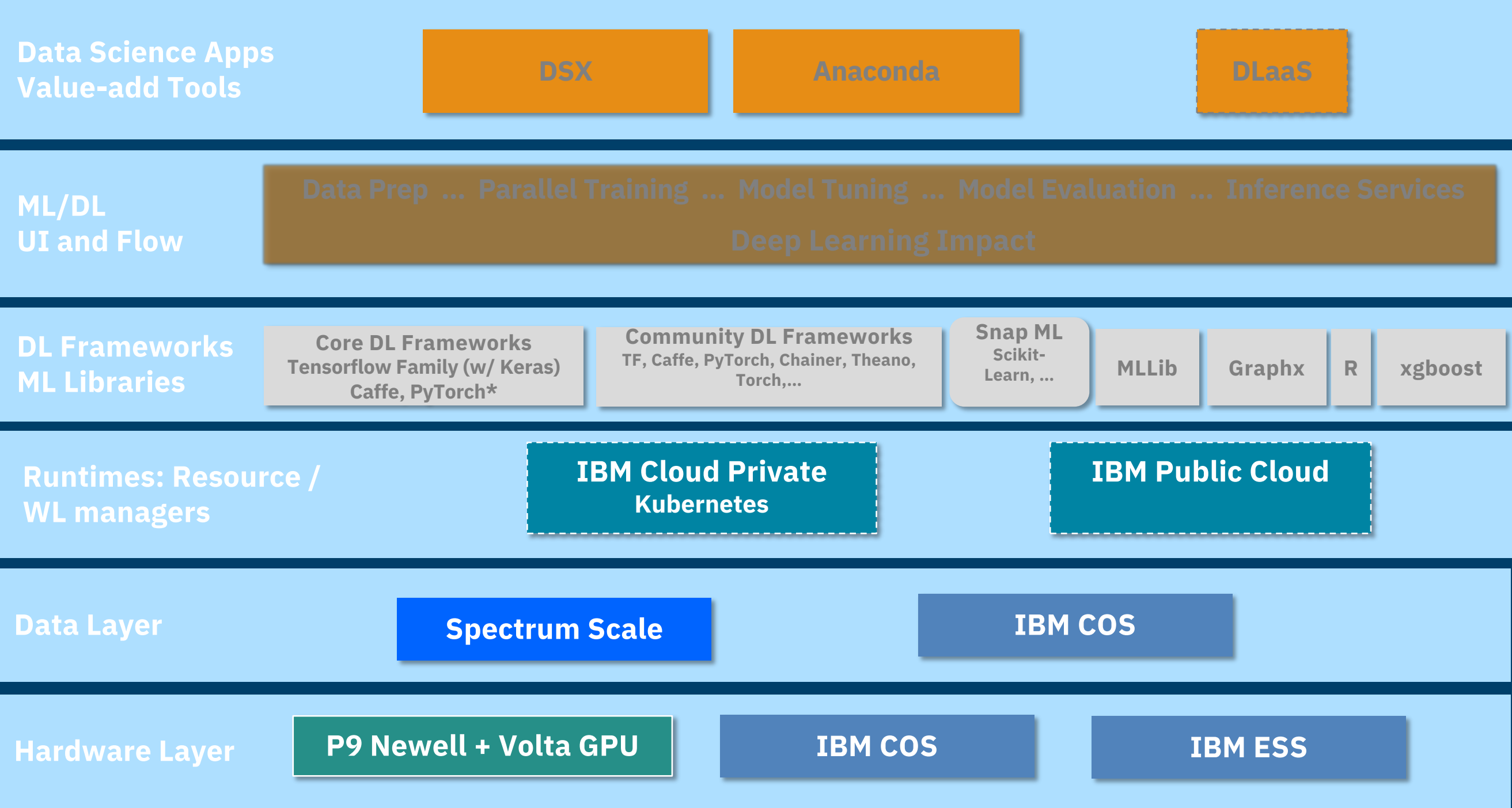
# Profile (Power9\* + Tesla™ V100 + NVLINK 2.0)



**Copy time completely hidden → Each iteration takes 93ms (3.5x faster)**



- Recall: openPOWER for HPC - differentiating features
- Porting a complex application: CPMD
- Porting a scalable benchmark: HPCG
- Porting a cloud benchmark: SNAP-ML
- HPC application porting: Trends
  - Libraries
  - Containerization
  - Jupyter



- DEMO

THINK

BIG

BIG

but be willing to be incremental on the way there