JÜLICH
FORSCHUNGSZENTRUM

Performance engineering for Lattice QCD applications

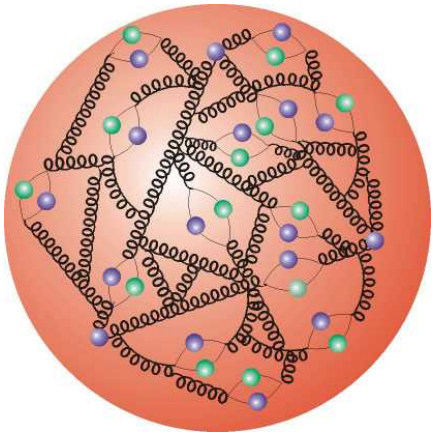
Outline

- Application area description
- Application performance signatures
- Selected performance results
- Summary and conclusions

Application Area Description

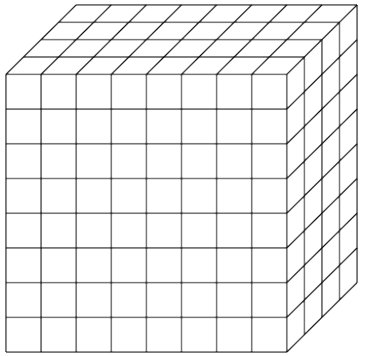
Quantum Chromodynamics

Theory of strong interactions: Quantum Chromodynamics



- Quarks are the constituents of matter which strongly interact exchanging gluons
- Particular phenomena:
  - Confinement
  - Asymptotic freedom (Nobel Prize 2004)

Formulation of QCD on the lattice → Lattice QCD



- Discretization of theory enables numerical, **ab-initio** simulations
- State-of-the-art projects need O(100) TFlop*year

Computational problem

Computation of physical observables (continuum):

$$\langle O \rangle = \int \mathcal{D}\phi \mathcal{D}\phi^* \mathcal{D}U \exp\left[-\phi^* \mathbf{M[U]}^{-1} \phi - S_G[U]\right]$$

- After discretization: Finite, but high-dimensional integral

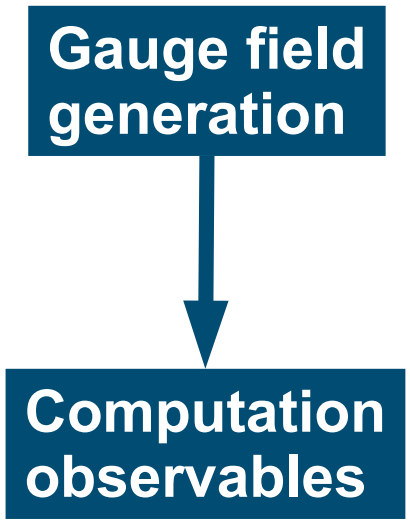
Problem can be formulated as statistical problem with following work-flow:

- Generate chain of gauge field configurations using Monte-Carlo methods:

$$U_0 \rightarrow U_1 \rightarrow \ldots \rightarrow U_N$$

  ☞ **need for scalable compute resources**

- Compute physical observables on the configurations U_i
  - Can be parallelized trivially



Computational kernel

Computational challenge: Solve linear equation

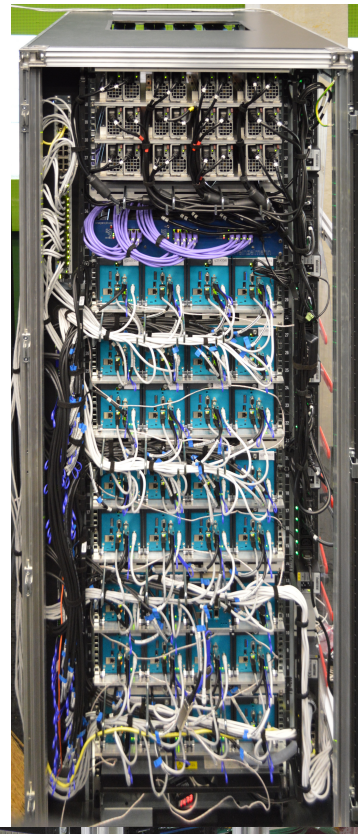
$$M\,\chi = s$$

- Matrix M huge, but sparse
- Typical size for size for state-of-the-art projects: $O(10^8)$

Standard approach: Iterative, Krylov-space based methods

Application Performance Signatures

Application optimised architectures



Goal

- Enable costs efficient solutions enable significant increase of the computational resources available for LQCD research

Example: QPACE

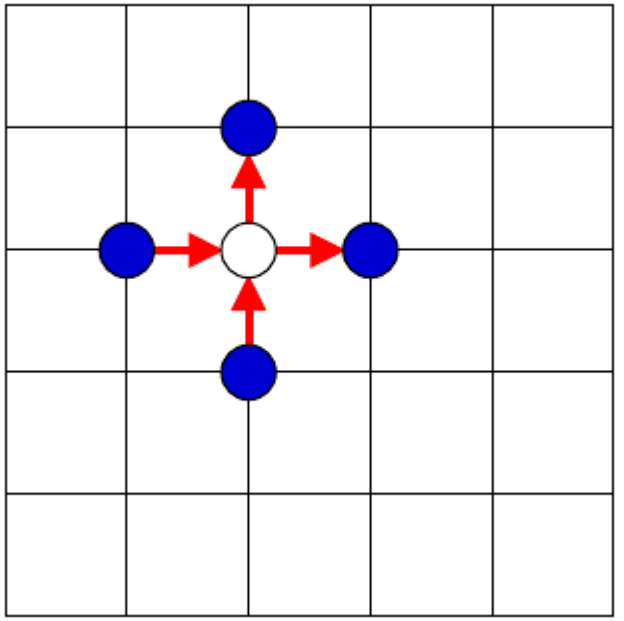
- Based on compute devices with exceptional performance
- Optimised network architectures
- Custom, costs-optimised integration

Application performance signature

Sparse matrix-vector multiplication dominates, e.g. Wilson-Dirac operator:

$$\psi_x = \sum_{\mu=0}^{3} \left\{ (1+\gamma_\mu) U_{x,\mu} \phi_{x+\hat{\mu}} + (1-\gamma_\mu) U^{+}_{x-\hat{\mu},\mu} \phi_{x-\hat{\mu}} \right\}$$
$$= D[U]_{xy} \phi_y$$

- U … 3 x 3 complex matrix
- ψ, ϕ … 3 x 4 complex matrix



Application performance signature (cont.)

Complex arithmetics dominate

Regular control flow

- Predictable behaviour

Parallelization based on 1-4d domain decomposition

- Exploitation of different levels of parallelism feasible
- Halo exchange
  - Communication requirements reduce when local lattice volume increases
- Nearest-neighbour communication patterns

Information flow analysis

Any computation requires flow of information between storage devices

- E.g., external and on-chip memory

Architectural parameters

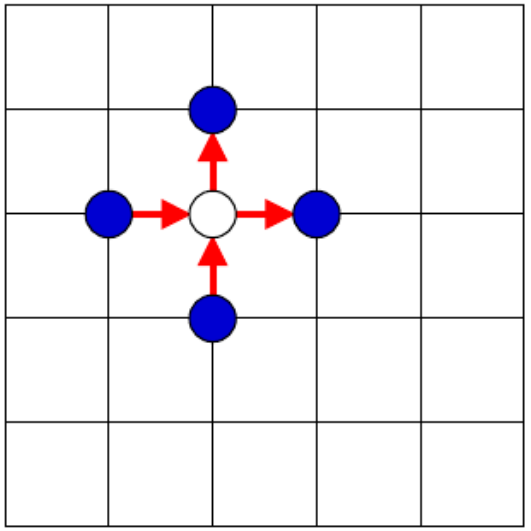
- Storage capacity C_x
- Bandwidth or throughput $B_{x \rightarrow y}$

Wilson-Dirac operator

- I_{fp} = 1320 Flop / site
- I_{mem} = 1.4 kiByte (SP) / site

Simple modelling ansatz

- $\Delta t_x(I_x) = \lambda + I_x / \beta$



Balanced architecture: B_{fp} vs. B_{mem}

Naive balance condition $I_{fp} / I_{mem} = B_{fp} / B_{mem}$

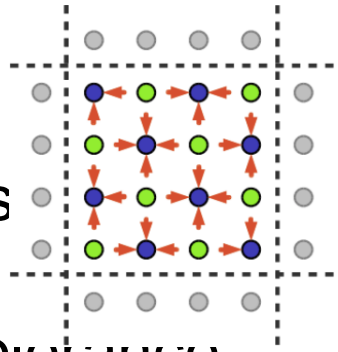
- Main kernel: $B_{fp} / B_{mem} = 0.9$ (SP)

Balance condition cannot be fulfilled (anymore) using discrete compute and memory devices

- Near-memory computing could change this

Different strategies to improve data locality

- Optimized implementations
  - E.g., optimize for re-use of stencil points
- Algorithms optimized for data locality
  - E.g., SAP-based approaches where sub-blocks are processed without communication
  - Minimizes data transport + increases latency tolerance



Digression: device level strong scaling limits

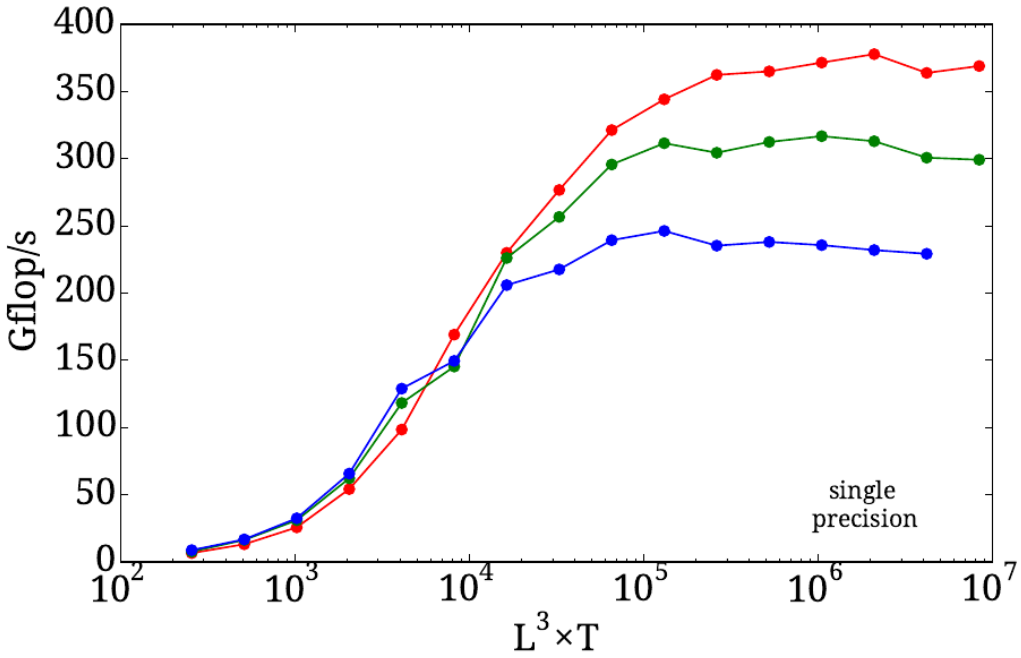
End of Dennard scaling → Increase performance = increase of parallelism

- More Flop/cycle
- Need larger problems to feed all pipelines

Example:

- Single-precision Dslash on NVIDIA K40m
- $V_{min} = 16^4 \rightarrow$ need at least ~32 sites per FMA pipeline

[G. Koutsou, 2014]



Balanced architecture: C_{mem}

Strong scaling limit defines minimal memory footprint for kernel

- Kernel requires <5 kiByte/site
- $C_{mem} > V_{min} * 5$ kByte
  - NVIDIA P100: $C_{mem} > 32\ N_{FMA} * 5$ kiByte = 450 MiByte

Caveat: Dependence of minimal footprint on algorithm

- Communication avoiding algorithms become inefficient for too small block sizes
- O(10) MiByte per block sufficient

Overall application memory footprint 10-100x larger

- Strongly depends on details of application and the used algorithms

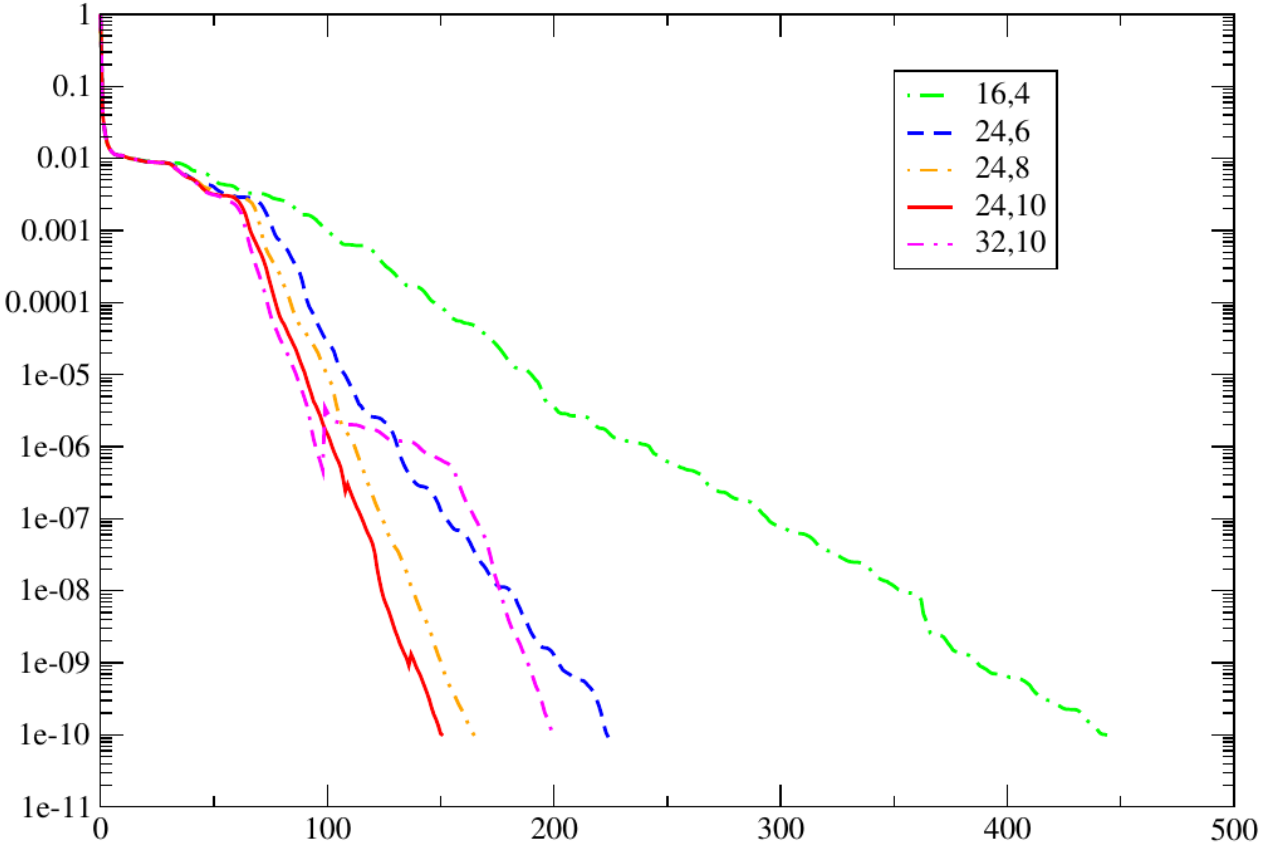
☞ Application typically not memory capacity limited

- O(100) TiByte for largest lattices

Memory capacity can matter

Case of the FGMRES-DR algorithm

- Need for re-orthogonalisation of vectors
- Need to restart if vectors of previous iteration cannot be stored



[A. Nobile et al., 2012]

Network performance

Communication pattern

- Regular nearest neighbour communication
  → favour torus network topology
- Global reduction operations

Information flow analysis d-dimensional torus network

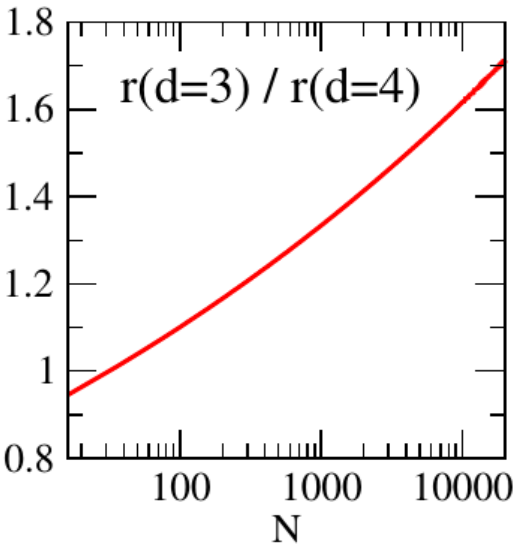
- Assume d-dimensional torus network ($d \leq 4$) such that local volume

$$v = L^{4-d} \left(\frac{L}{N^{1/d}} \right)^d = L^{4-d}\, l^d$$

- Surface to (local) volume ratio r:

$$r = \frac{2d\, L^{4-d}\, l^{d-1}}{v} = 2d\, l^{-1} = 2d\, \frac{N^{1/d}}{L}$$

- For large N large d slightly more favourable



Machine vs. algorithmic performance

Machine performance

Set of relevant performance numbers which can be measured during execution of a computational task in terms of a hardware related metric

- Examples: flops per time unit, port occupation rate

Algorithmic performance

Number of atomic sub-tasks, e.g., matrix-vector multiplications, to solve a particular problem

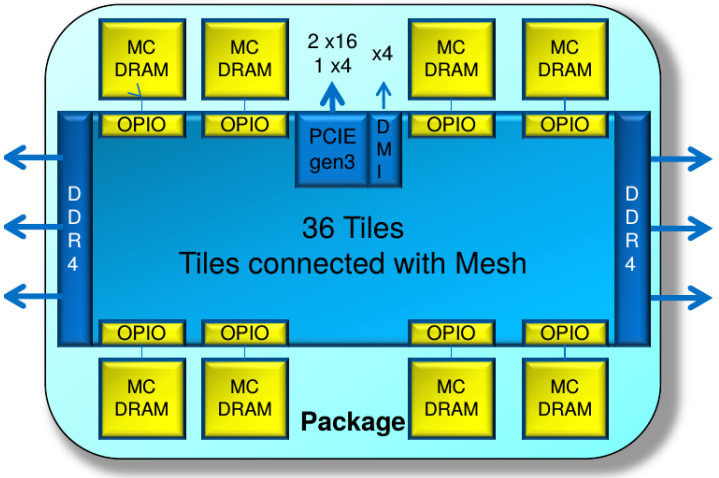
- Example: number of solver iterations

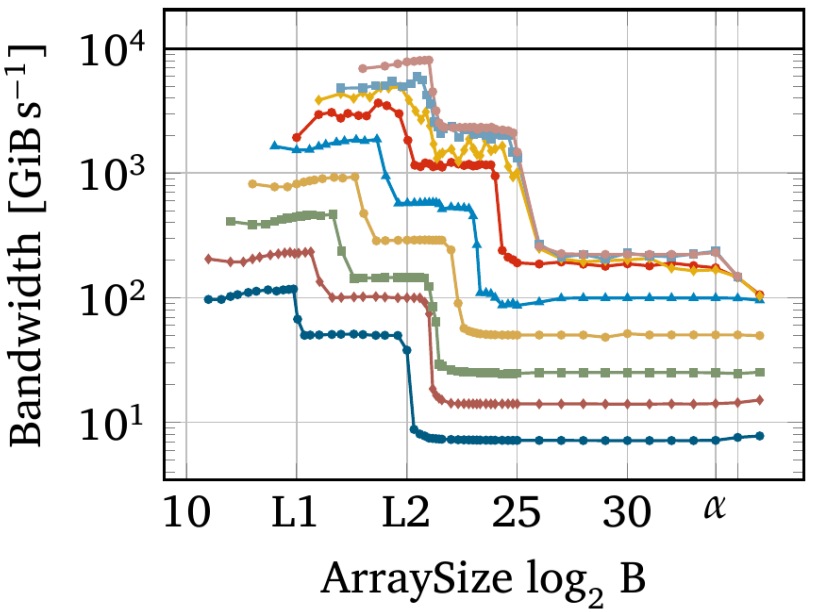
Past LQCD performance improvements both due to improved machine and algorithmic performance

Selected results on state-of-the-art architectures

Intel Knights Landing: Properties

B_{fp} (DP/SP) [Flop/cycle]		~2048 / ~4096
f [GHz]		1.3-1.5
B_{mem} [GByte/s]	DDR MCDRAM	~110 >400
C_{mem}	[GByte] DDR MCDRAM	96 (typical) 16





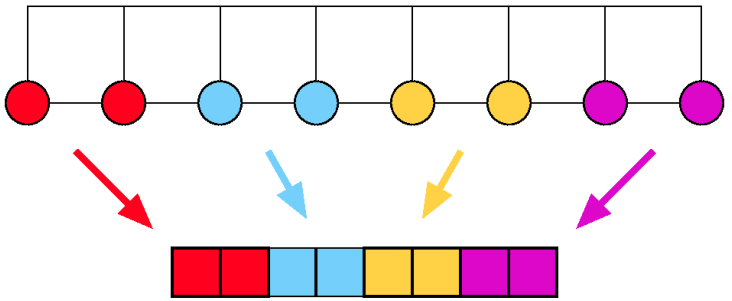
Intel Knights Landing: Selected optimisations

Data layout / SIMDsation

- Domain decomposition for parallelisation
- Typical layout: `float spinor[nvec][3][4][2][SOA]`
  - SOA < 16 requires gather/scatter (for SP)

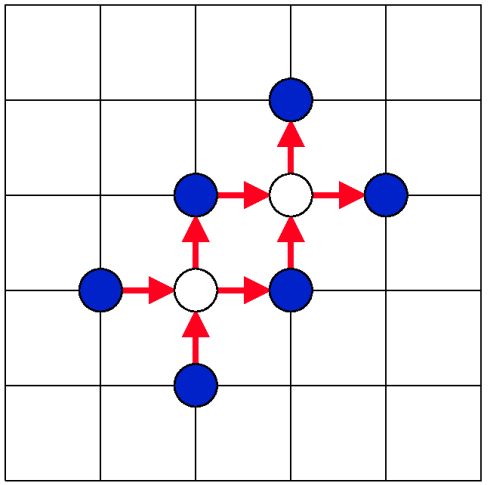
Maximize data re-use

- Cache-blocking and kernel fusion

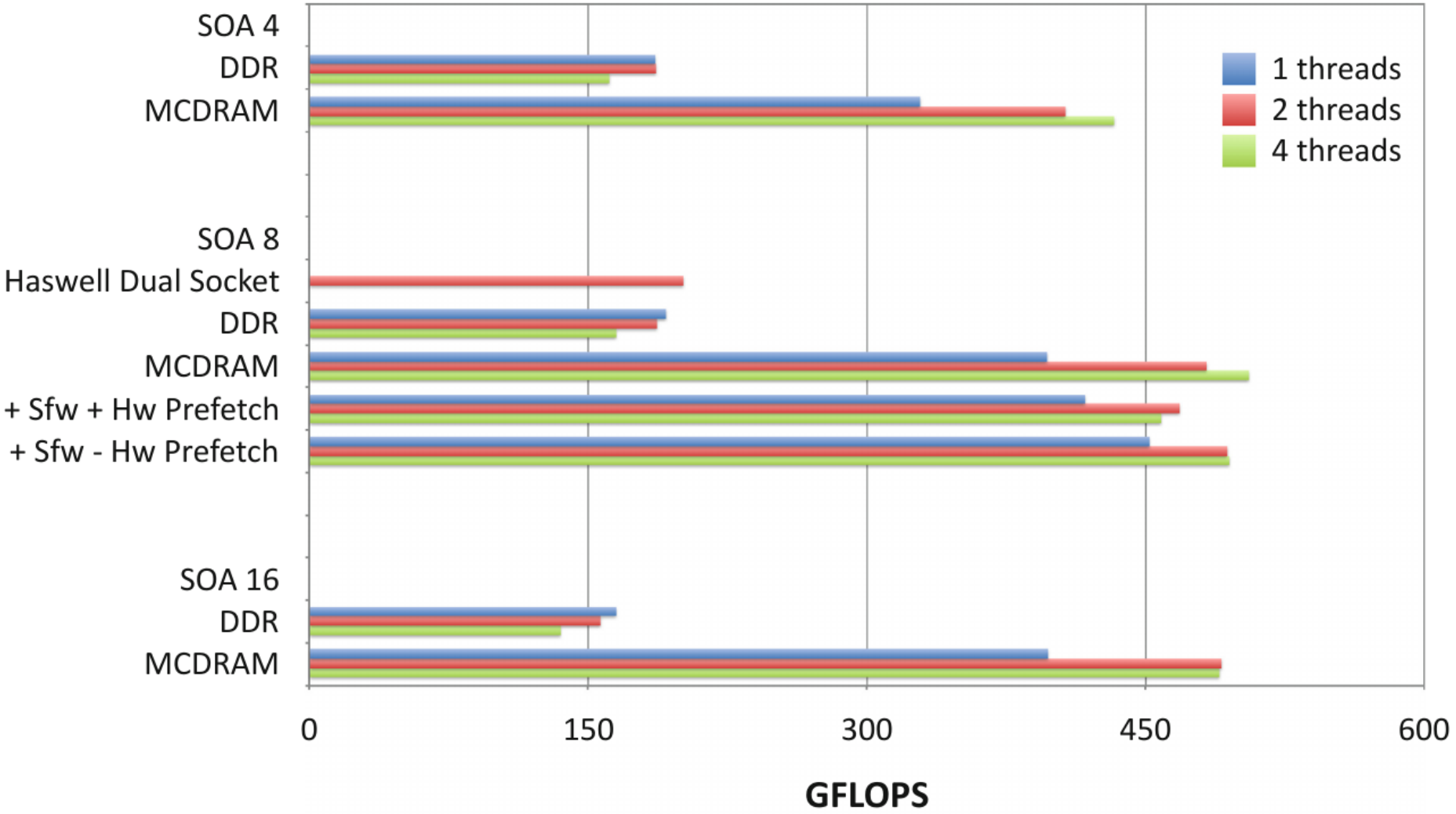


Memory pre-fetching

- hardware only vs.
  software assisted hardware vs.
  software only



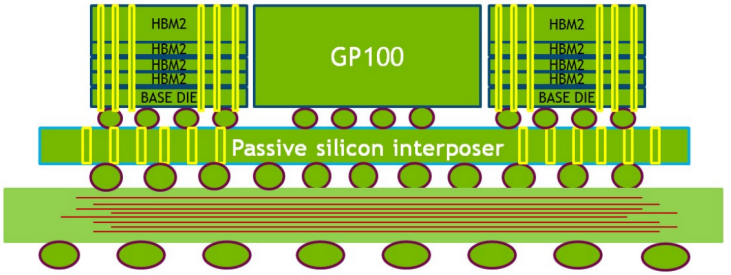
KNL: Performance results based on QPhiX



[B. Joó et al., 2016]

https://github.com/JeffersonLab/qphix

NVIDIA K80 / P100: Properties



	K80	P100
B_{fp} (DP/SP) [Flop/cycle]	3328 / 9984	3584 / 7168
f [GHz]	0.56	1.3
B_{mem} [GByte/s]	2*240	720
C_{mem} [GByte]	2*12	16

NVIDIA K80 / P100: Optimisation

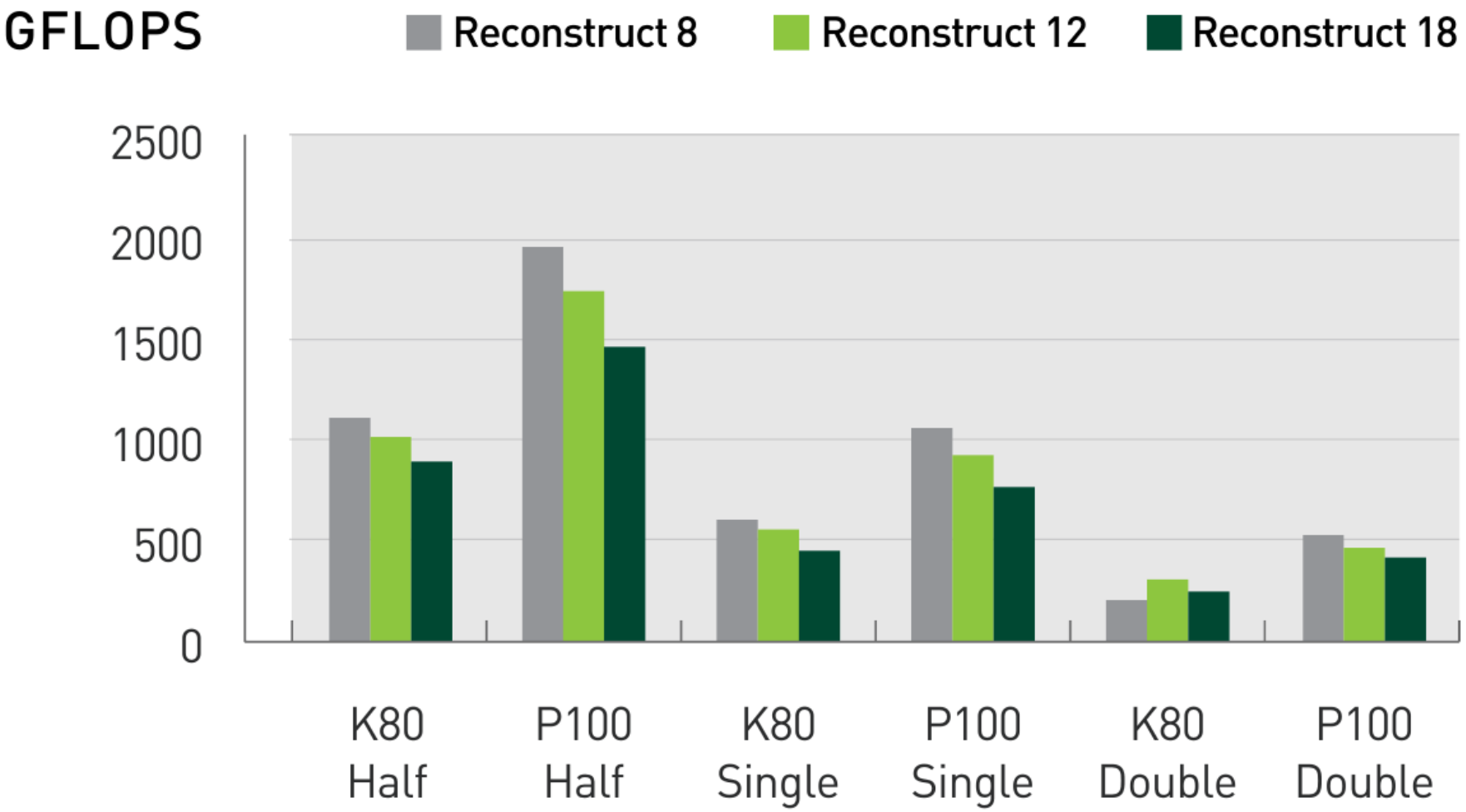
Reduce data to be loaded

- Exploiting mathematical properties allows to store only truncated data objects
- Reconstruction of objects after load into GPU
  - `reconstruct#`: Use # floats to store 3x3 complex matrix

Exploit mixed precision

- Modern solvers allow bulk computations to be performed in lower precision without compromising on final precision
- Lower precision may have negative effect on algorithmic performance

K80 / P100: Performance results based on QUDA



[M. Wagner, 2016]

http://lattice.github.com/quda

Summary and Conclusions

Summary and conclusions

Analysis of application performance signatures established in Lattice QCD community

- Key for application optimisation and design of application optimised HPC infrastructure

Efficient exploitation of current architectures

- Strong-scaling limits per device are becoming an issue
- High-bandwidth memory is critical to balance performance
- Capacity of high-bandwidth memory sufficient

Performance comparison on state-of-the-art extremely parallel compute devices

- About 500 GFlop/s for main kernel on Intel KNL and 1000 GFlop/s on NVIDIA P100
- Performance exceeds naïve estimate due to data reuse/data reduction