QPACE 2
and
Domain Decomposition on the KNC

Tilo Wettig

Department of Physics
University of Regensburg

Lattice 2014, Columbia University, New York

Outline

- Introduction
- QPACE 2
- Domain Decomposition on the KNC
- Outlook

Outline

- Introduction
- QPACE 2
- Domain Decomposition on the KNC
- Outlook

Custom machines for lattice QCD?

- long history starting in the 1980s:
  - ACPMAPS, GF11, Columbia-256, QCDSP, QCDOC
  - QCDPAX, CP-PACS, PACS-CS
  - APE, APE100, APEmille, apeNEXT
  - QPACE 1
- very sensible at the time: more performant/scalable and more cost-effective ("science per dollar") than commercial supercomputers
- in the last ∼10 years things have changed:
  - chip design has become too expensive for academic projects
  - commercial supercomputers (in particular BlueGene) provide high LQCD performance at reasonable cost and energy efficiency
  - standard "PC" clusters with accelerators (GPU, KNC) are fine if scalability is not a must (analysis, thermodynamics, parameter scans)

Columbia-256

We are trying very hard to get the 256-node machine built and get good physics out of it.

–Norman Christ, Columbia

this year that will have a theoretical peak speed of 16 gigaflops, and 256 megabytes of memory, expandable to 2 gigabytes. "We are trying very hard to get the 256-node machine built and to get good physics out of it," says Christ.

a toroidal, or doughnut, topology. Each node consists of a processor and memory. The machines operate in a lockstep, synchronous mode, with each node communicating locally to its nearest neighbor. A Digital VAX 11/780 serves as a host

grained," to a few very powerful general purpose, or "coarse-grained" ones.

At 256 nodes, the proposed Columbia machine represents a medium-grained approach, with a moderate number of powerful nodes. By

source: Peter Batacan, New Computer Architectures for Lattice QCD, Computers in Physics 3 (1989) 17

Custom machines for lattice QCD!

- nevertheless, custom designs can still beat commercial machines in cost efficiency and energy efficiency (most recently QPACE 1)
- basic approach: combine commercially available components in an innovative way (network, cooling, energy efficiency)
- metrics for success:
  - scalability (preferably strong scaling)
  - cost-performance (Dollar per GFlop/s sustained)
  - power-performance (Watt per GFlop/s sustained)
  - development costs must be amortized
- co-design with industrial partners essential
  - combine strengths of industry and academia
  - innovative concepts often come from academic side
- nowadays, LQCD “custom designs” are much more general-purpose
- NB: development of machines like BlueGene or K Computer relies on Gigadollar-scale government funding (not always forthcoming)

Outline

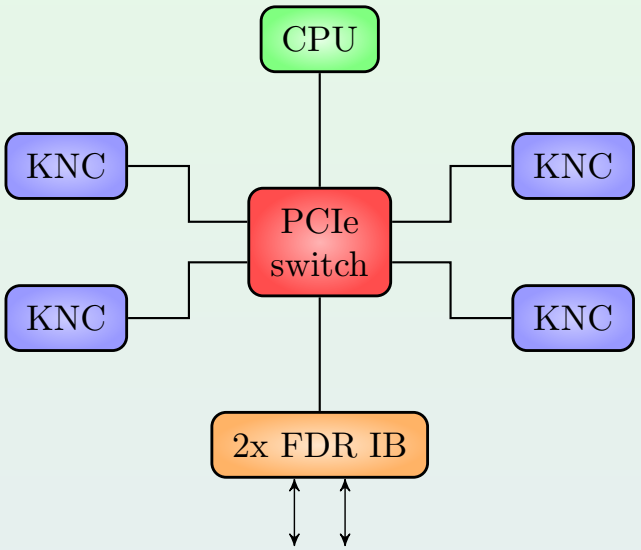
- Introduction
- QPACE 2
- Domain Decomposition on the KNC
- Outlook

QPACE 2: Overview

- QPACE now stands for “QCD Parallel Computing Engine”
- funded by German Research Foundation DFG (SFB/TR-55)
- collaboration of Regensburg University with Eurotech (Italy) and Intel, with contributions from Jülich Research Center and Wuppertal University
- initial collaboration with T-Platforms (Russian company) was cut short by US Department of Commerce
- schedule:
  - start of design (with Eurotech): May 2013
  - begin of bringup and debugging: spring 2014
  - machine available by end of summer/early fall 2014 (hopefully)

QPACE 2: High-level design

- machine consists of identical compute nodes:
  - 4 Intel Xeon Phi a.k.a. Knights Corner (KNC) coupled via PCIe switch (PLX 8796)
  - weak CPU (Xeon E3-1230L v3) for root complex functionality (PCIe master)
  - dual-FDR Infiniband card (Mellanox Connect-IB)

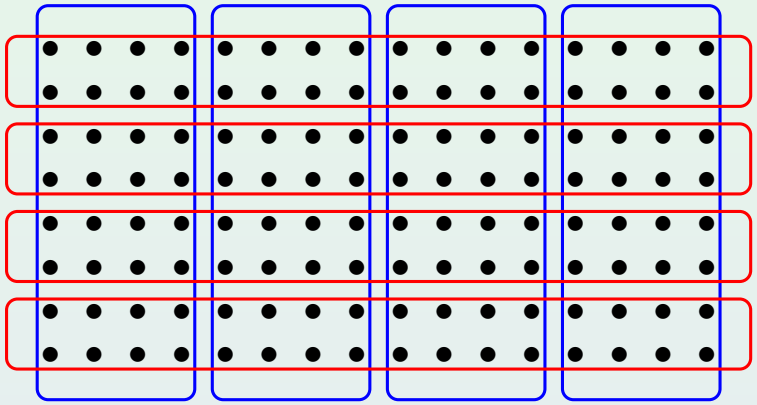


- communication
  - within node via PCIe (8 GB/s between each KNC and PCIe switch)
  - out of node via Infiniband (13.5 GB/s per node)

- Intel management:
  - "I'm glad to see someone did this. I'd actually proposed a machine like this using the large fast buffers on the switch >2 years ago. Look forward to seeing the machine."

QPACE 2: Network topology

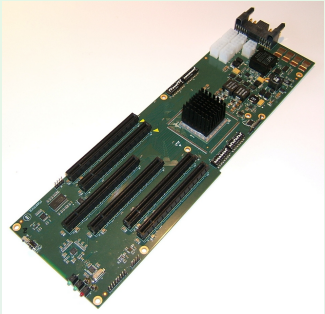
- Hyper-crossbar (copied from CP-PACS),[1] here based on Infiniband
  - 2-dim. example for 16×8 nodes and switches with 32 ports:



  - higher connectivity than torus:
    - full connectivity in every single dimension with one switch hop
    - all-to-all connectivity with (few) hops between switches
  - generalizes to higher dimensions
    or map higher-dimensional application to lower-dimensional hyper-crossbar
- IB edge switches have 36 ports
  → use 4 extra ports for connection to storage system and for torus of switches (Torus-2QoS by Mellanox)

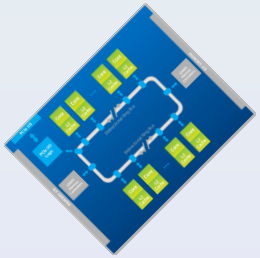
[1] thanks to Peter Boyle for suggesting this

QPACE 2: Major components

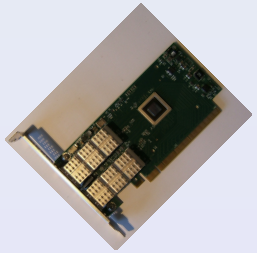


- midplane (MARS)
  - designed together with Eurotech (Italy)
  - contains PCIe switch PEX 8796 from PLX (96 lanes Gen3) and 6 PCIe slots (16 lanes each)

- CPU card (JUNO)
  - designed together with Advanet (Japan)
  - Intel Xeon E3-1230L v3 (25W TDP) with 16 GB memory
  - CPU only needed for booting and PCIe root complex
  - also contains Board management controller (BMC)
  - 2 Gigabit Ethernet interfaces for booting and control



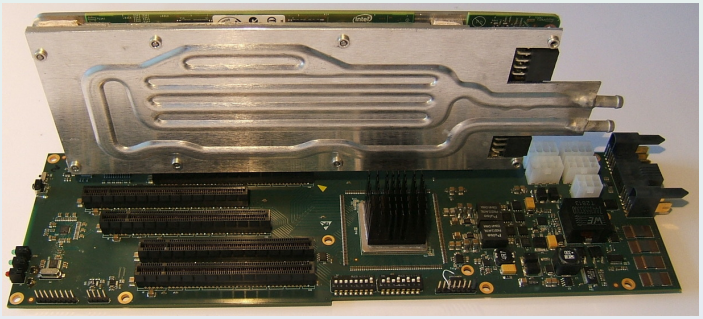
- KNC (from Intel)
  - workhorse for computations
  - we use version 7120X (61 cores at 1.2 GHz, 16 GB memory)



- Connect-IB Infiniband card (from Mellanox)
  - 16 lanes Gen 3 to PCIe switch
  - 2 FDR ports to network

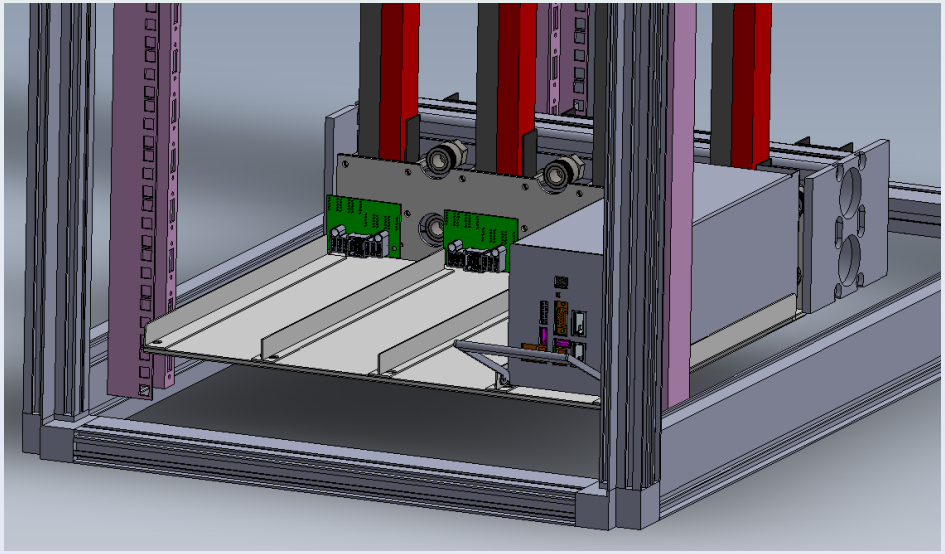
QPACE 2: System design

- system design uses new "brick" concept:
  (patented with Eurotech)
  - midplane with PCIe switch (96 lanes Gen3)
  - 6 PCIe slots (16 lanes each) for CPU card, ConnectIB, and 4 KNCs
  - KNCs could be replaced by GPUs
  - innovative liquid cooling (roll-bond), coldplates attached to each card
  - brick height 3U, 4 + 4 bricks per 3U in 19" rack (front and back)



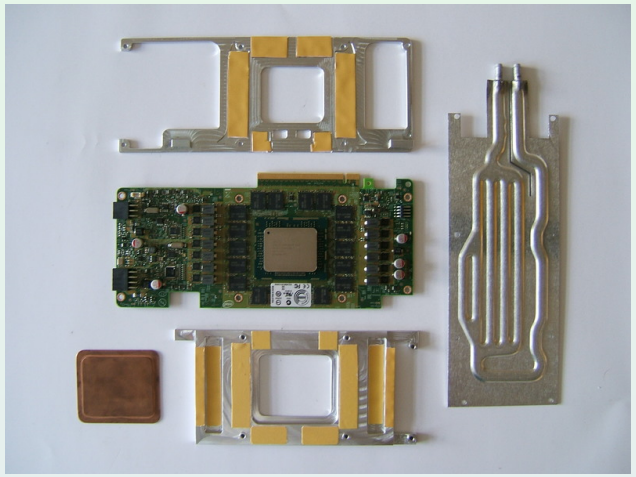
QPACE 2: System design

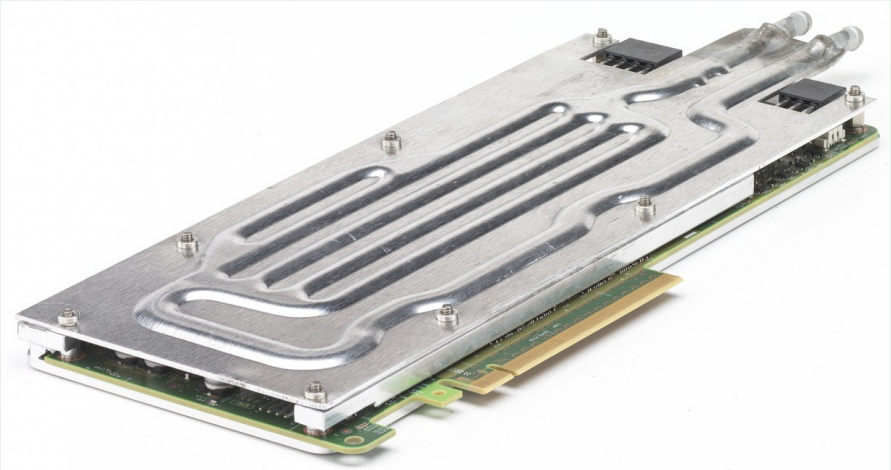
- power distribution:
  - 380V AC to 12V DC conversion by 2kW Platinum PSUs ($\sim$ 95% efficiency)
  - 5+1 redundancy for 8 bricks
- rack design:
  - standard 19” rack (height 42U)
  - 64 bricks (256 KNCs) in 24U
  - rest for PSUs, switches (4x IB and 3x GigE), management/login server
  - 310 TFlop/s DP peak per rack (KNCs only) at $\sim$ 75 kW



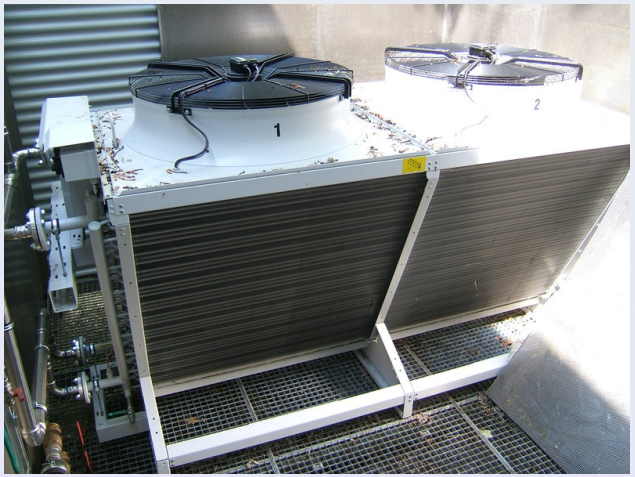
QPACE 2: System design

- cooling concept: coldplates + roll-bond plate

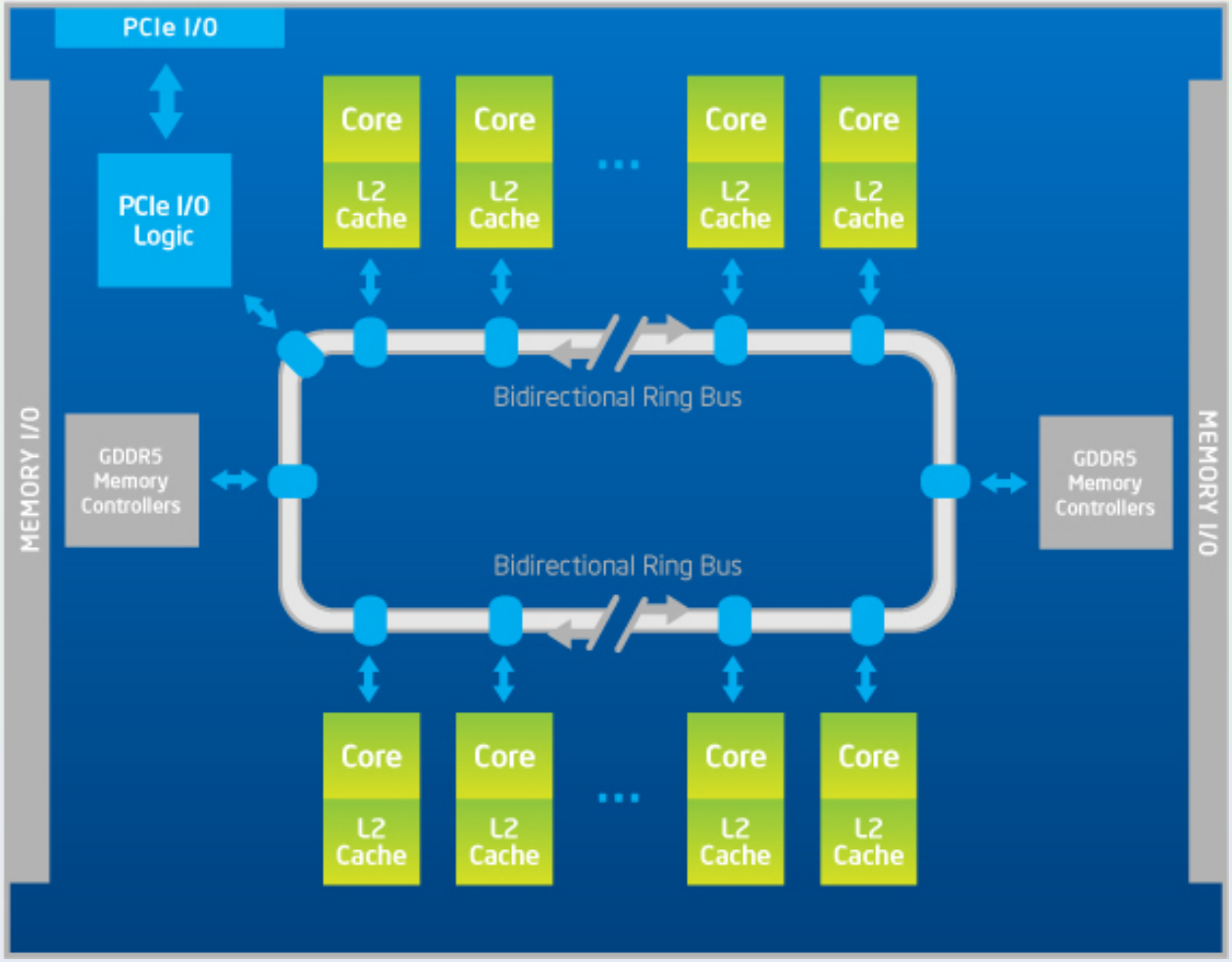




- water cooling up to 50°C (at least); free cooling year-round
  → very energy efficient (no chillers needed)



KNC architecture



source: intel.com

- each core has 512-bit wide vector unit
- 512 kB private L2 cache per core (unified with distributed tag directory)

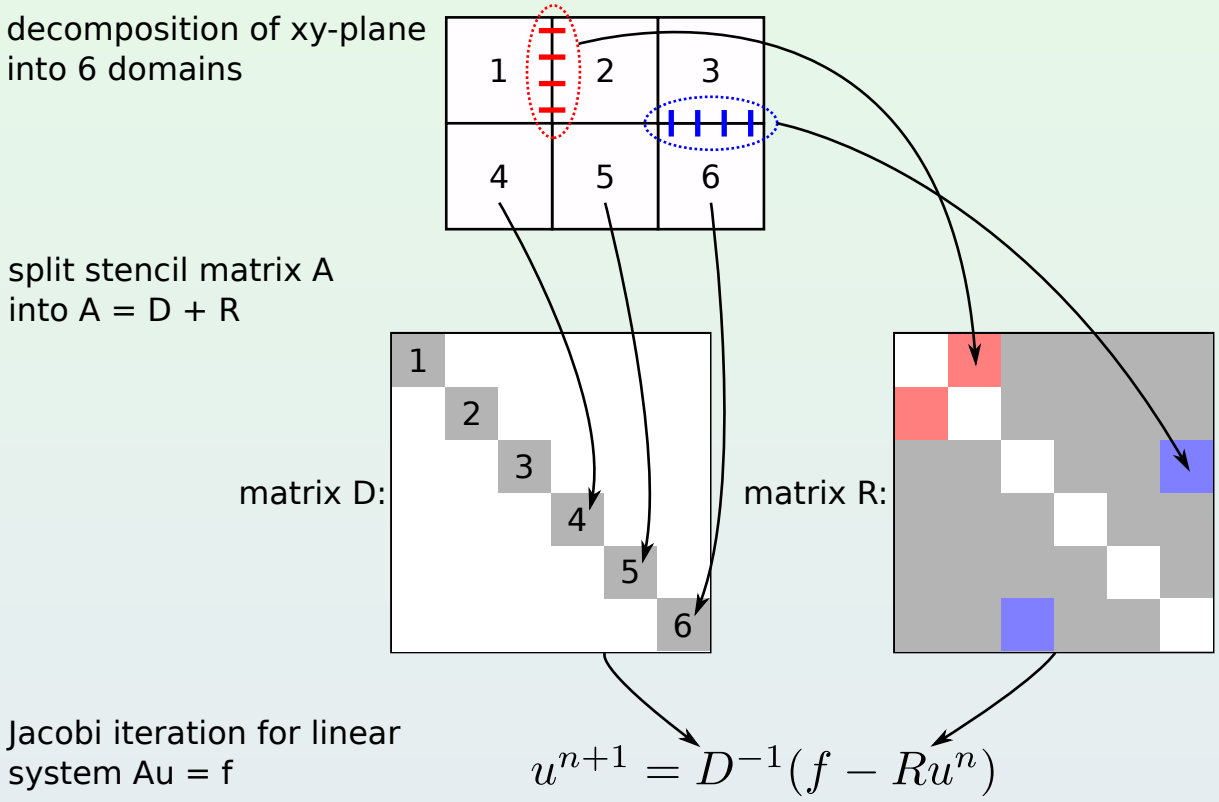
KNC architecture

- compute cores
  - $\sim$ 61 cores running at about 1.2 GHz
  - each core can do FMA on 512-bit vector $\rightarrow$ 1.2 TFlop/s peak (DP)
  - instructions for single and double precision
  - up to 4 (hyper-) threads per core (with their own register set, but shared execution units and caches)
  - in-order execution (load can stall execution for 10 – 100 cycles if data are not in cache)
  - instructions from a particular thread can only be issued every other cycle
    $\rightarrow$ need at least 2 threads to get 1 instruction/cycle and to hide pipeline stalls due to in-order execution
    $\rightarrow$ complex program structure
- memory and cache
  - max. sustained memory bandwidth $\sim$ 170 GB/s (read + write) (advertised bandwidth 352 GB/s)
  - accessing caches of other cores faster than memory access, but not by much

Outline

- Introduction
- QPACE 2
- Domain Decomposition on the KNC
- Outlook

Reminder: Domain Decomposition



- main idea: Schwarz (1870), Lüscher (2004)
  - subdivide lattice into domains and reorder indices → block-diagonal + rest
  - inversion only on domains (no communication required, ideally from cache)
  - rest (application of R) needs comms but does not occur frequently

→ less communication, better latency tolerance, more cache reuse

Implementation overview

- main person: Simon Heybrock
  Intel collaborators: Dhiraj Kalamkar, Misha Smelyanskiy, Karthik Vaidyanathan
- all details in paper submitted to SC14 (Heybrock et al.)
- Dirac operator: Wilson clover
- algorithms:
  - outer solver: flexible GMRES with deflated restarts
    Frommer, Nobile, Zingler, arXiv:1204.5463
  - preconditioner: multiplicative Schwarz (= SAP)
    inversion on domains: minimial residual (MR) with e/o preconditioning
- code written in C++
- performance-relevant parts vectorized using icc intrinsics
- threading with OpenMP
- multi-node implementation with Intel MPI
- benchmark runs on TACC Stampede cluster:
  - KNC version 7110P (1.1 GHz, 8 GB memory, using 60 cores)
  - FDR Infiniband with 7 GB/s peak bandwidth
  - using Intel-provided proxy to communicate medium/large packets via CPU to circumvent hardware issues unrelated to KNC

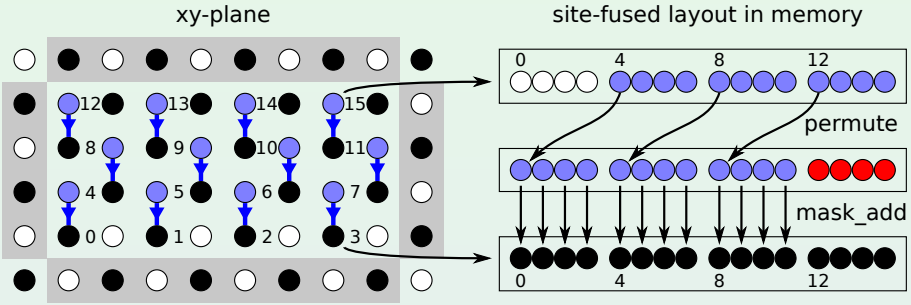
Data layout and mapping to SIMD

- goals:
  - avoid loading cache lines that are only partially needed
  - use all SIMD elements
  - avoid instruction overhead due to permutations
- best solution:
  - structure-of-array (SOA) format, i.e., all 24 floating-point components of a spinor are stored in 24 separate registers and cache lines
  - this leads to "site fusing": 16 sites in one 512-bit register (for SP)
    in our case: 4×4 sites per register in x and y direction
- computation of hopping terms:
  - straightforward in z and t direction
  - in x and y, use permute/mask → wastes 12.5% (25%) of SIMD units in x (y)
  - in site-fused dimensions, hopping terms between domains would give large overhead:
    - need to load cache line with neighbor's boundary data
    - but this cache line contains extra data that are not needed

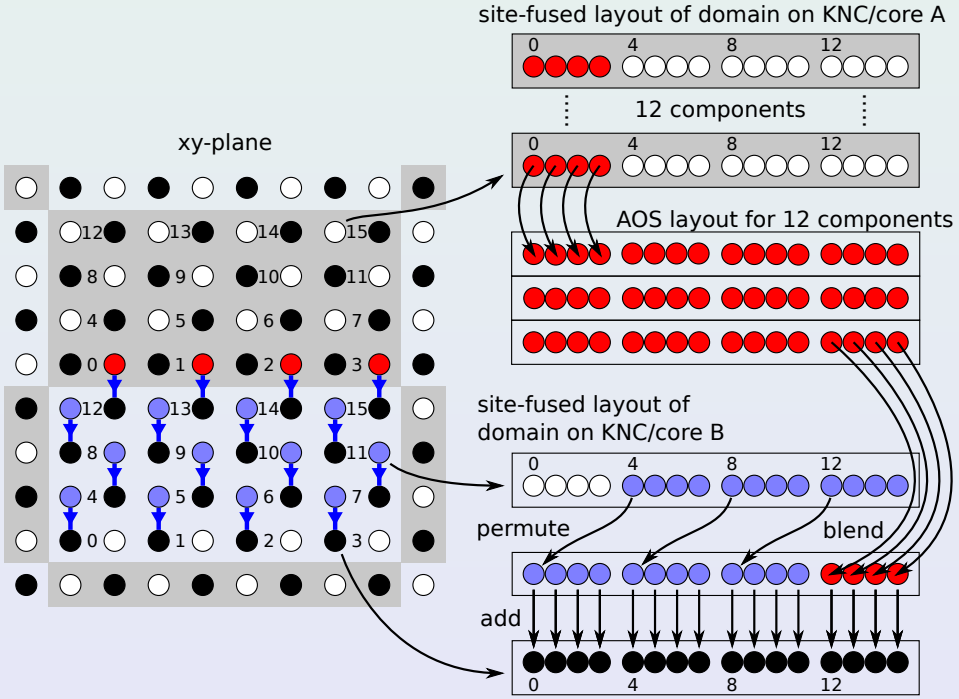
    → additionally store boundary data in array-of-structure (AOS) format

Permuting/masking and repacking of boundary data

- permuting/masking:



- repacking of boundary data:



Cache management and prefetching

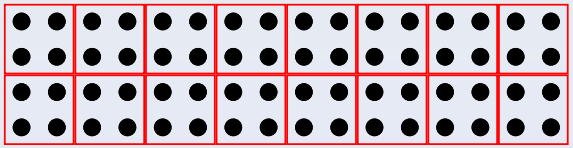
- one domain per core since L2 is not shared
- cache size (512 kB/core) restricts domain size to 8×4^3 (in SP)
- KNC can do up- and down-conversion
  $\longrightarrow$ store (some) domain data in half precision
  $\longrightarrow$ reduced working set and reduced bandwidth requirements
  - to ensure stability, spinors are kept in single precision
  - gauge links and clover matrices in half precision
  - no noticeable impact on iteration count of outer solver
- prefetching:
  - no L1 hardware prefetcher
  - L2 hardware prefetcher only for streaming access
  - compiler-generated software prefetches often not good enough
  - manual L1 and L2 prefetches essential (using intrinsics)
  - fine-tuning of prefetches rather time-consuming

Intra-core threading

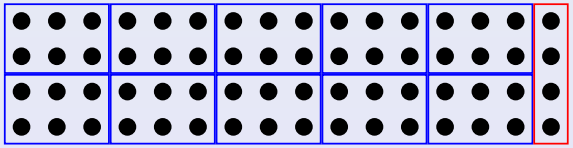
- need at least two threads per core for full pipeline utilization
- we assign threads to alternating time slices within domain
- we see no significant differences between two or four threads per core:
  - two threads: more stalls due to latency of L1 or L2 misses
  - four threads: working set exhausts L1 size
    $\rightarrow$ threads evict each other's data more frequently

Inter-core parallelization

- using OpenMP
- recall: one domain per core
- synchronization between cores only necessary after MR block solve
  $\longrightarrow$ cost of barrier has no significant impact
- load-balancing issues with standard lattice sizes (2^n) on 60 cores (some cores would be unused)
  - simple issue but significant impact on performance
  - possible workarounds:
    - use prime factors of 3 and 5 in lattice sizes (for new lattices)
    - non-uniform partitioning of the lattice
      example: processors with 6 cores each, 4×16 lattice



$2 \cdot 8 = 16$ processors
$(16 \cdot 2 = 32$ cores unused)



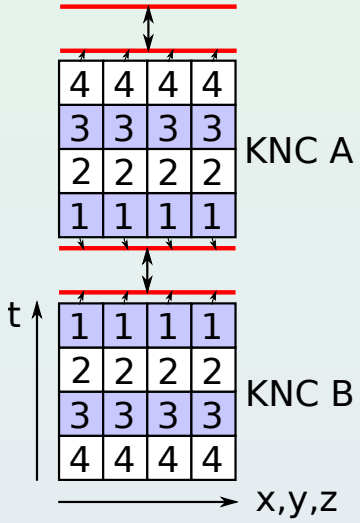
$2 \cdot 5 + 1 = 11$ processors
(2 cores unused)

Multi-node implementation

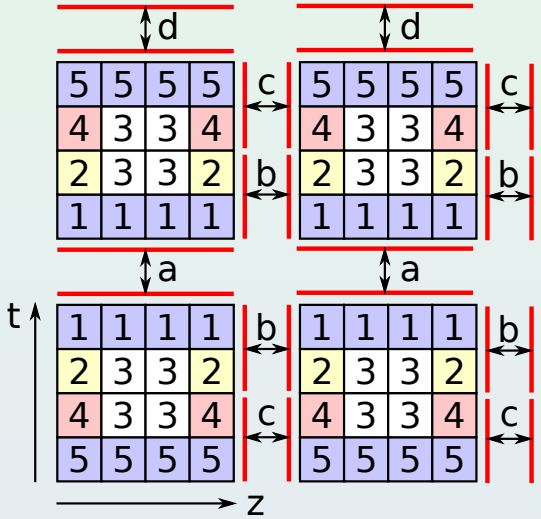
- could have each thread issue its own MPI calls, but:
  - typically high overhead for MPI calls from several threads
  - message sizes too small for efficient network utilization
- better:
  - combine surface data of all domains and communicate them using a single thread
  - needs explicit on-chip synchronization
- hiding communication behind computation is important (even for DD)
  - standard method (divide local volume into interior and surface) does not work for us since most domains would be on the surface
  - instead, send boundary data when half of them are ready

Communication latency hiding for DD

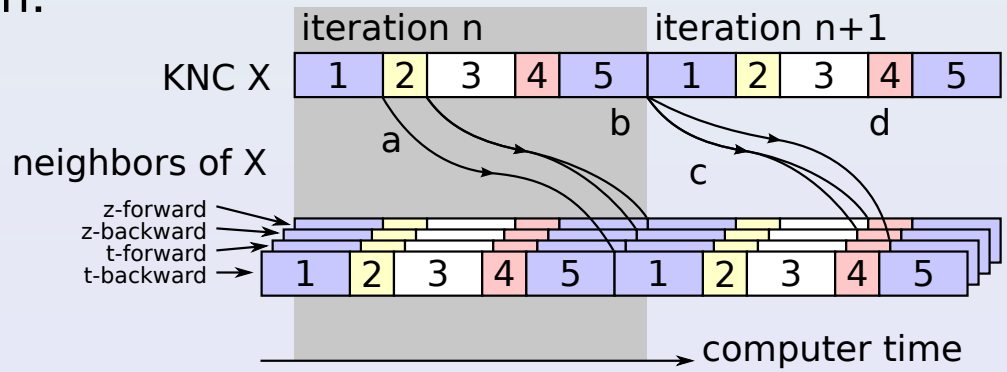
- boxes represent domains, numbers represent order of execution, small letters represent order of communication
- bad:



good:



- linear representation:

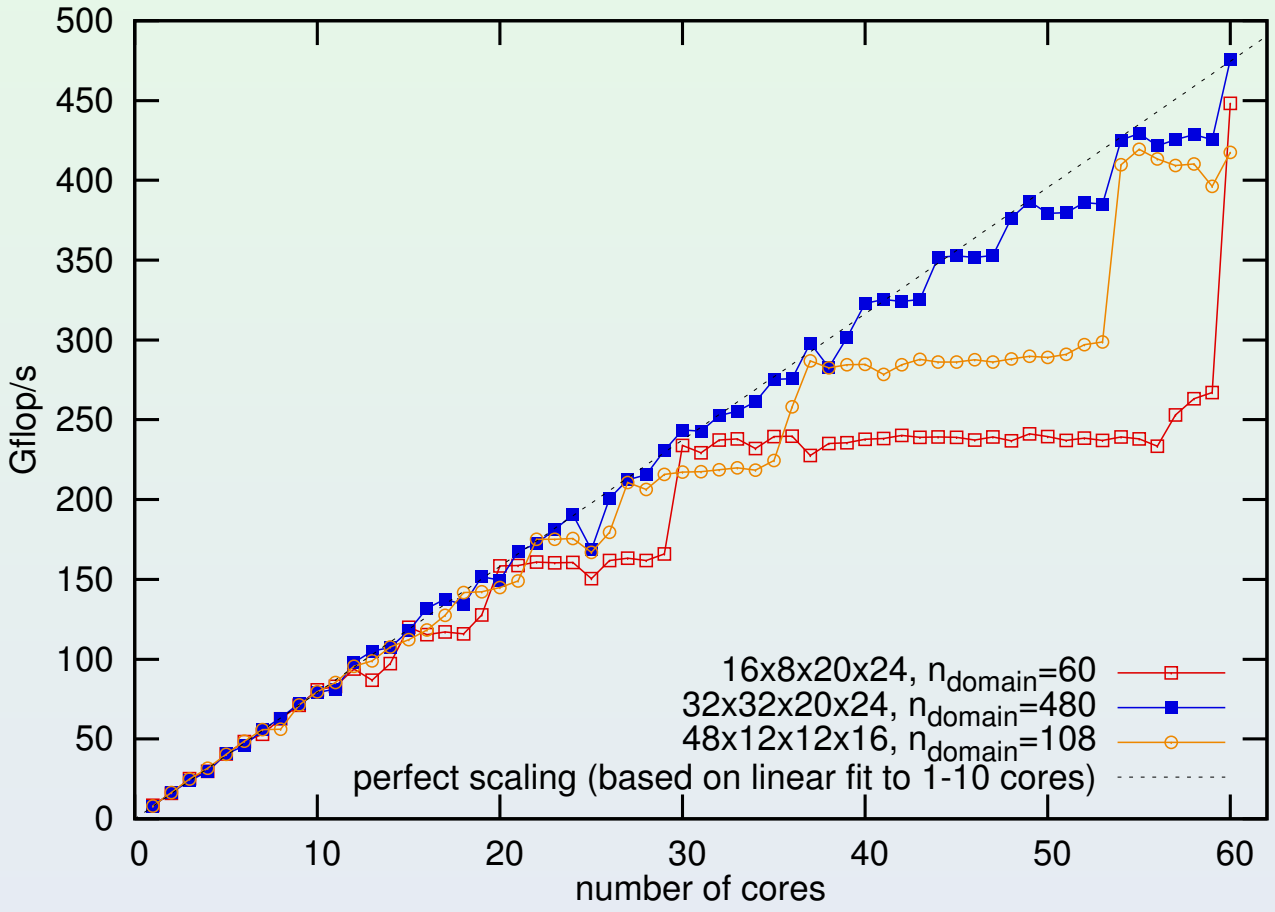


Results: Single core

- theoretical performance model predicts 56% of peak = 20 GFlop/s/core
- actual MR performance $\sim$ 12 GFlop/s on single core
  main culprit (VTune): stalls due to outstanding L1 prefetches
- optimal number of MR iterations (for minimal time-to-solution) is only 4$\sim$5
  $\rightarrow$ other parts of Schwarz method contribute significantly
  $\rightarrow \sim$ 8 GFlop/s/core
- single-core performance in GFlop/s:

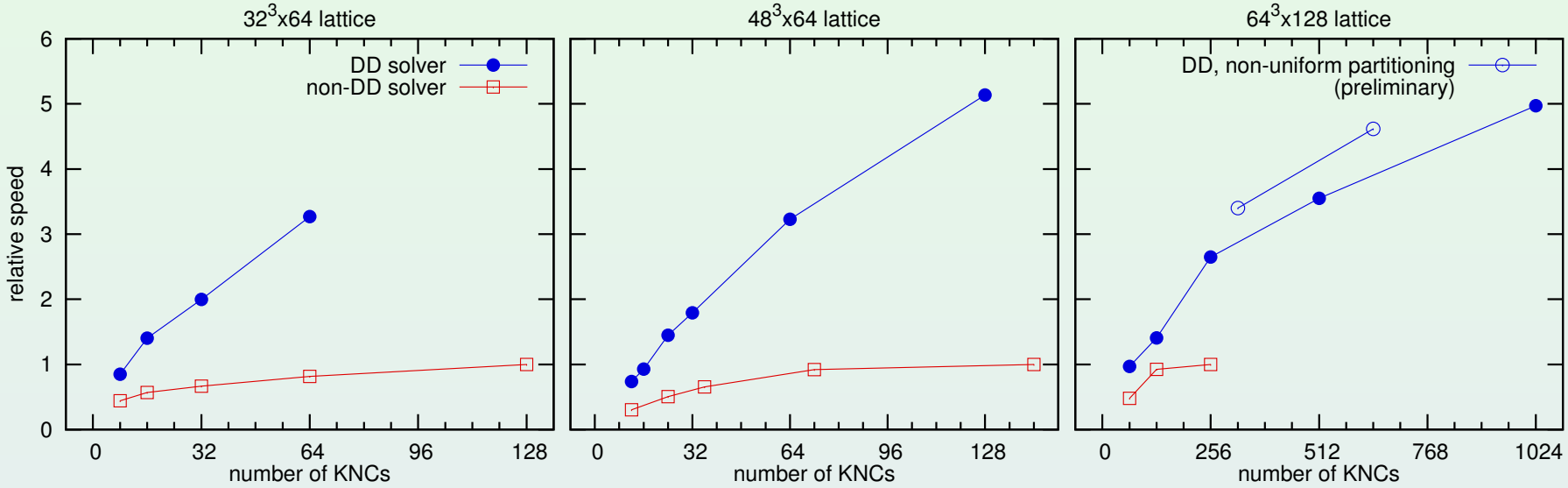
	MR iteration		DD method	
	single	half	single	half
no software prefetching	5.4	7.9	4.1	5.9
L1 prefetches	9.2	11.8	5.8	7.7
L1+L2 prefetches	9.1	11.8	6.3	8.4

Results: Single KNC



- almost perfect scaling (except for load imbalance):
  - cores can work independently during MR inversion
  - almost no competition for memory access since MR runs from cache

Results: Strong scaling on many nodes (HMC)

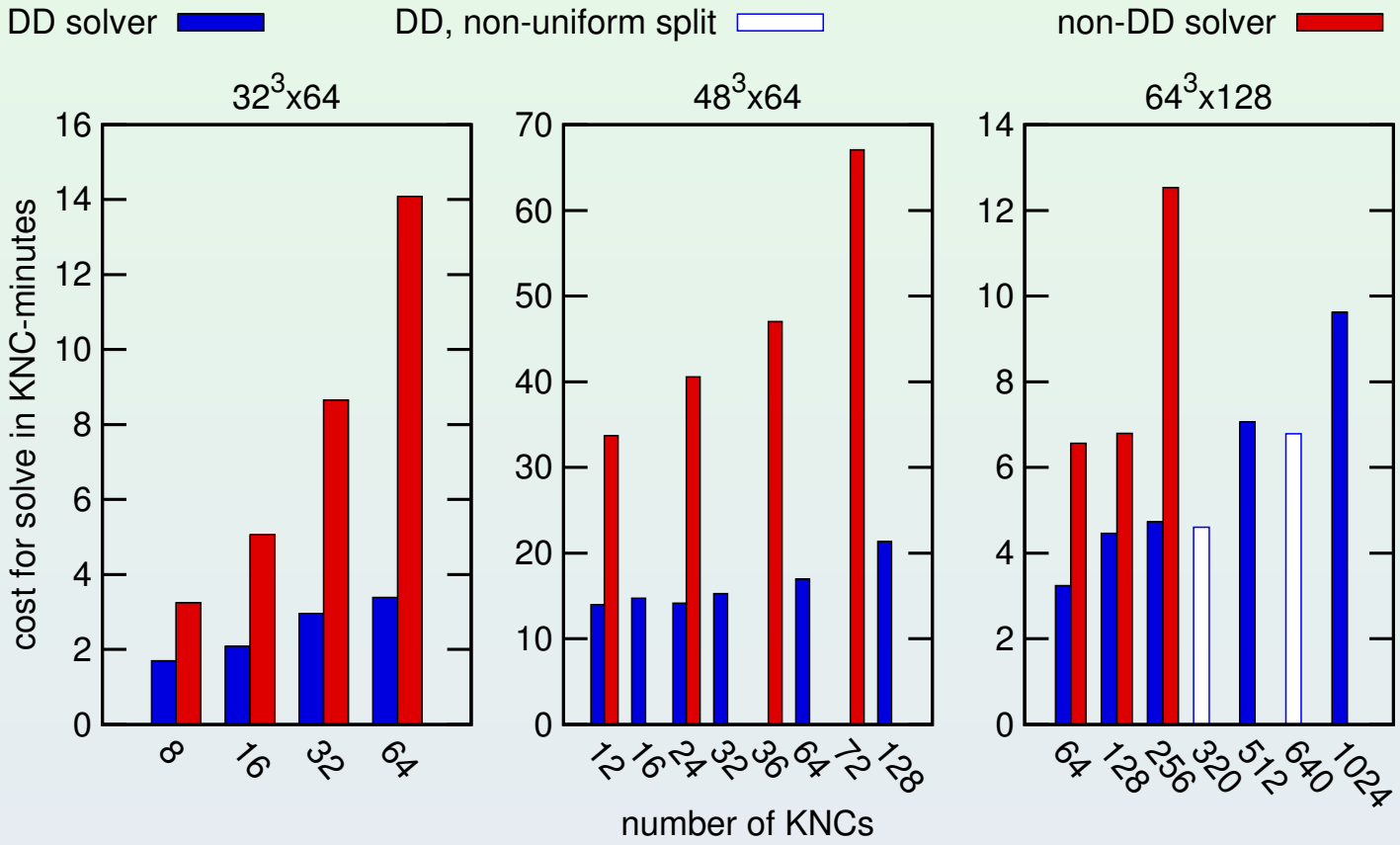


- $m_\pi = 290$ MeV, 150 MeV (QCDSF), SU(3) point ($\sim$ 800 MeV) (USQCD)
- results normalized to minimum time-to-solution for non-DD solver (BiCGstab)
- DD strong-scales to more nodes (also better for equal number of nodes)
- performance drop for large number of nodes:
  - overlapping communication with computation becomes harder/impossible
  - message sizes are smaller $\rightarrow$ less efficient network communication
- max. number of nodes is determined by local volume:
  - if domains too small $\rightarrow$ DD less efficient
  - we don't split domains over cores (no shared L2)

Results: Strong scaling details

KNCs	n_{domain}	load	percent of total time A	M	GS	other	Gflop/s/KNC A	M	GS	other	iterations	Tflop/s M	Tflop/s total	[s]	#global-sums	comm./KNC
$48^3 \times 64$, DD, parameters $m = 16$, $k = 6$, $I_{\text{Schwarz}} = 16$, $I_{\text{domain}} = 5$																
24	288	96%	4.3	85.8	7.8	2.1	66	299	56	143	198	7.0	6.3	35.4	423	15593
32	216	90%	4.0	86.5	7.3	2.2	67	276	55	127	198	8.6	7.8	28.6	423	13156
64	108	90%	4.5	85.9	6.8	2.7	52	250	53	92	198	15.6	14.0	15.9	423	8040
128	54	90%	5.3	83.4	7.0	4.4	35	199	40	42	198	24.9	21.6	10.3	423	5116
$64^3 \times 128$, DD, parameters $m = 5$, $k = 0$, $I_{\text{Schwarz}} = 16$, $I_{\text{domain}} = 5$																
64	512	95%	4.7	89.4	3.5	2.3	64	300	29	24	10	18.8	17.1	3.34	27	488
128	256	85%	4.4	90.0	4.0	1.5	50	221	19	27	10	27.6	25.3	2.3	27	293
256	128	71%	4.5	90.2	3.8	1.5	45	204	19	26	10	51.0	46.8	1.22	27	171
*320	112/64	85%	4.8	89.5	4.0	1.7	48/27	230/131	20/11	26/15	10	67.3	59.9	0.95	27	152/98
512	64	53%	3.9	91.1	3.6	1.4	35	135	13	18	10	67.5	62.7	0.91	27	98
*640	56/32	85%	4.7	88.5	5.0	1.8	33/19	158/90	11/6	17/10	10	92.4	81.4	0.70	27	98/61
1024	32	53%	5.9	86.7	4.5	2.8	16	100	7	6	10	100.0	88.4	0.65	27	61
$48^3 \times 64$, non-DD: double-precision BiCGstab, SOA-length $= 8$ (performs better than mixed-precision solver, since half limited to SOA $= 8$)																
12	-	-	-	-	-	-	entire solver: 105				4781	-	0.82	168.5	23,907	188,272
24	-	-	-	-	-	-	entire solver: 87				4777	-	1.36	101.4	23,887	115,556
36	-	-	-	-	-	-	entire solver: 57				4802	-	1.77	78.4	24,012	91,848
72	-	-	-	-	-	-	entire solver: 39				4760	-	2.46	55.9	23,802	48,200
144	-	-	-	-	-	-	entire solver: 21				4728	-	2.66	51.4	23,642	26,598
$64^3 \times 128$, non-DD: mixed-precision Richardson inverter — outer solver: double (SOA $= 8$) — inner solver BiCGstab: residual 0.1, single stored as half (SOA $= 16$)																
64	-	-	-	-	-	-	entire solver: 101				$\approx 12 \cdot 23$	-	6.3	6.1	1408	2500
128	-	-	-	-	-	-	entire solver: 94				$\approx 12 \cdot 22$	-	11.7	3.2	1353	1314
256	-	-	-	-	-	-	entire solver: 56				$\approx 12 \cdot 24$	-	14.1	2.9	1473	948

Results: Minimum cost (for data analysis)



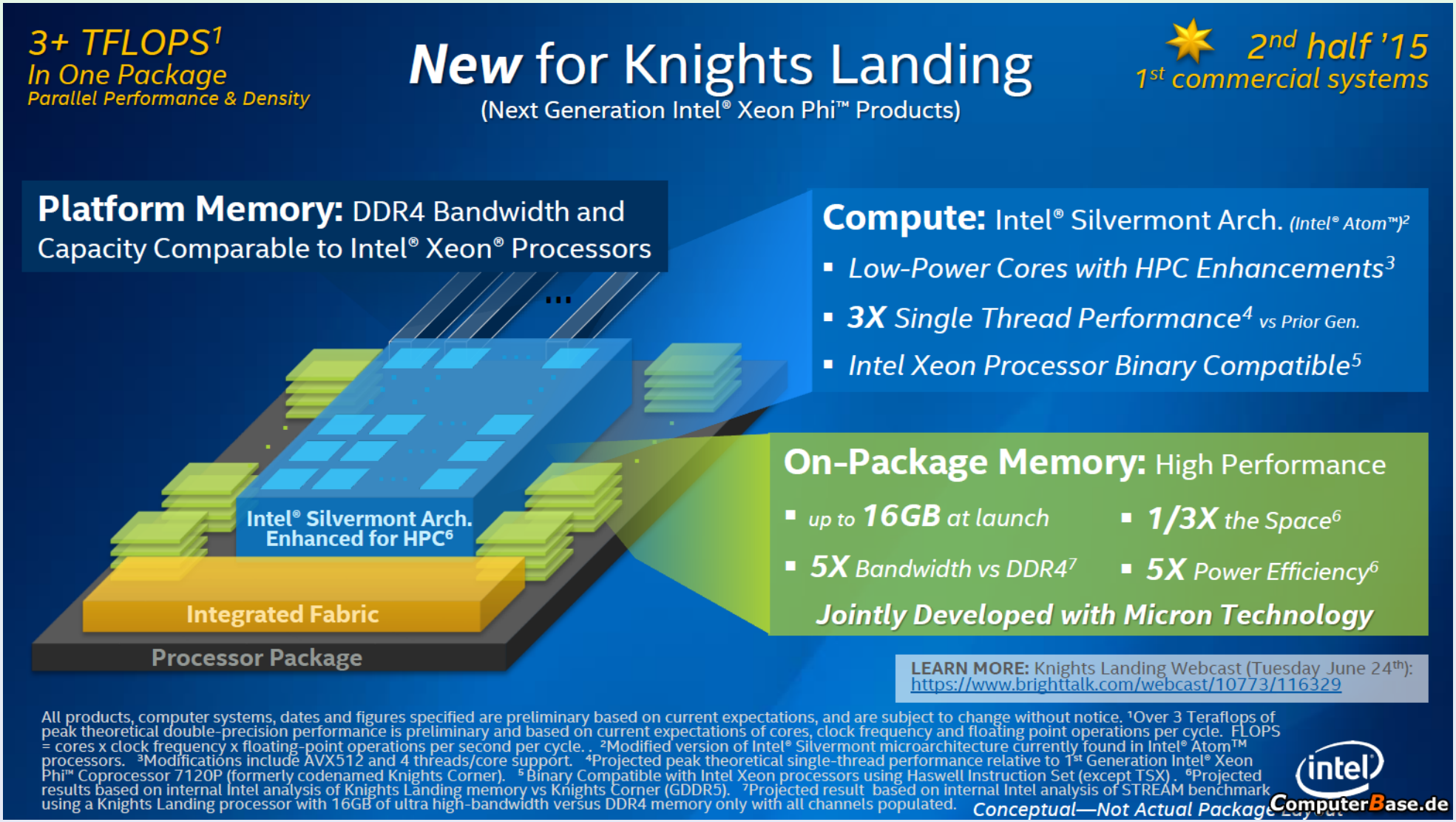
- run on as few nodes as memory footprint allows
  → minimizes negative impact of network communication
  → lowest cost
- in all cases, DD costs about half as much as non-DD

Outline

- Introduction
- QPACE 2
- Domain Decomposition on the KNC
- Outlook

Outlook

- QPACE 2 will hopefully be up and running in the next few months
  - one prototype rack with 256 KNCs at Regensburg
- multigrid:
  - Simon Heybrock is currently porting Wuppertal algebraic multigrid code (arXiv:1303.1377, 1307.6101) to KNC
  - outer solver still fGMRES
  - preconditioner now multigrid: consists of V-cycle + smoother (= DD code)
- follow-up project (QPACE 3) in planning/early development
  - will use KNL (Knights Landing) instead of KNC
    - 3× peak performance per chip at same/less power
    - better cores (out of order, hardware prefetcher)
    - high-bandwidth memory on package (Micron HMC)
    - network controller integrated on KNL-F (Omni Scale)
  - several network options (Infiniband, Omni Scale, or ExpressFabric)
  - plan is to have 4 racks ($\sim$ 4 PFlop/s DP peak) at Jülich



from http://www.computerbase.de/2014-06/xeon-phi-knights-landing-mit-72-kernen-und-on-package-dram

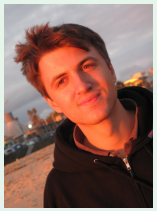
Thanks

- Regensburg team: Jacques Bloch, Benjamin Glässle, Simon Heybrock, Robert Lohmayer, Simon Mages, Bernhard Mendl, Nils Meyer, Florian Rappl, Stefan Solbrig







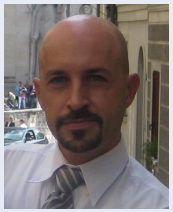




- Regensburg machine shop: Johann Deinhart, Norbert Sommer


- Jülich/Regensburg: Dirk Pleiter

- Eurotech: Paul Arts, Alessio Parcianello, Mauro Rossi, Giampietro Tecchiolli, Gianpaolo Zanier





- Advanet: Yu Komatsubara