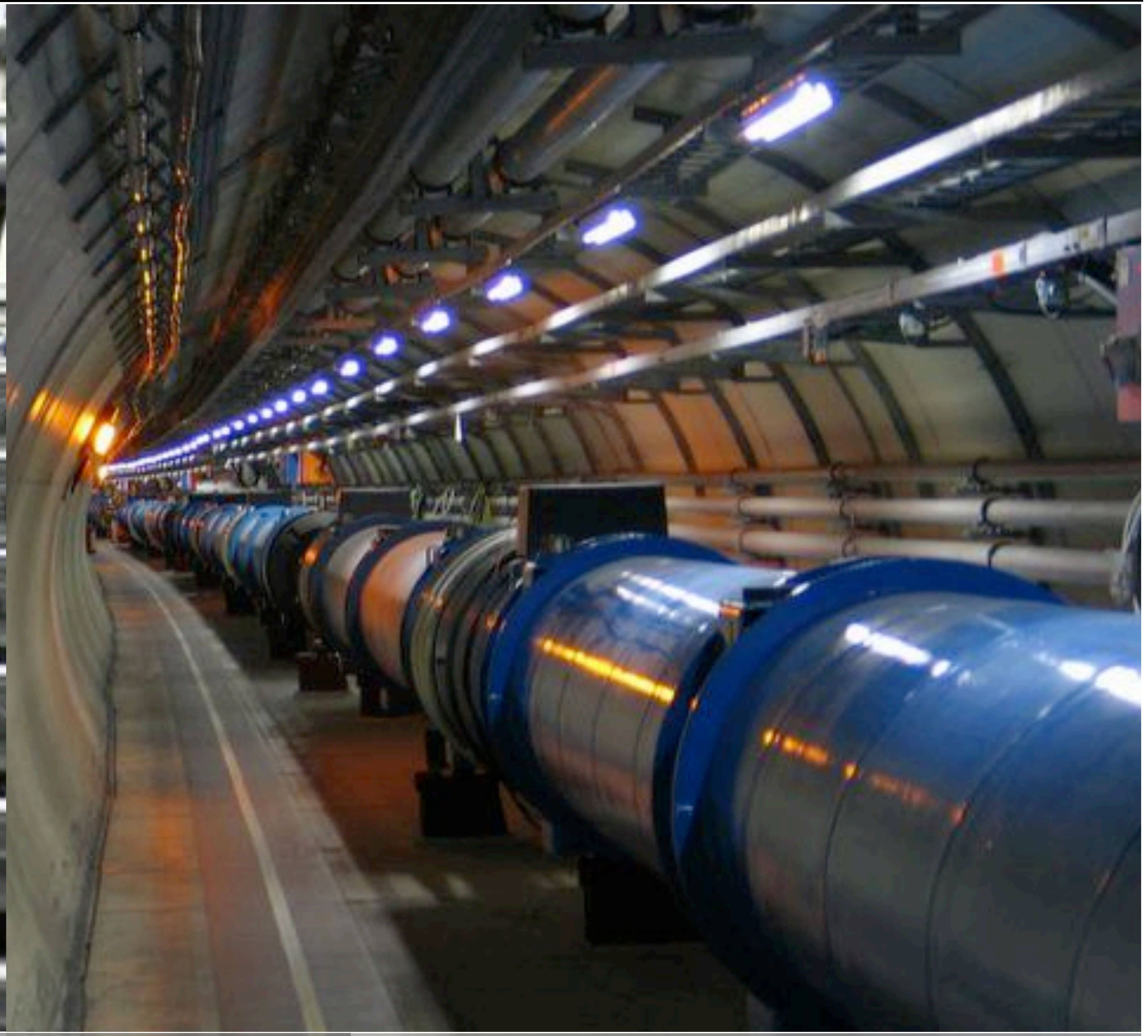
GPUs Immediately Relating Lattice QCD to Collider Experiments





Universität Bielefeld

Outline

- Quantum ChromoDynamics
- Fluctuations from Heavy-Ion experiments and lattice QCD
- Lattice QCD on GPUs and on the Bielefeld GPU cluster
- Optimizations
  - includes first experiences with Kepler architecture
- Relating Lattice Data to Collider Experiments
- Outlook

Outline

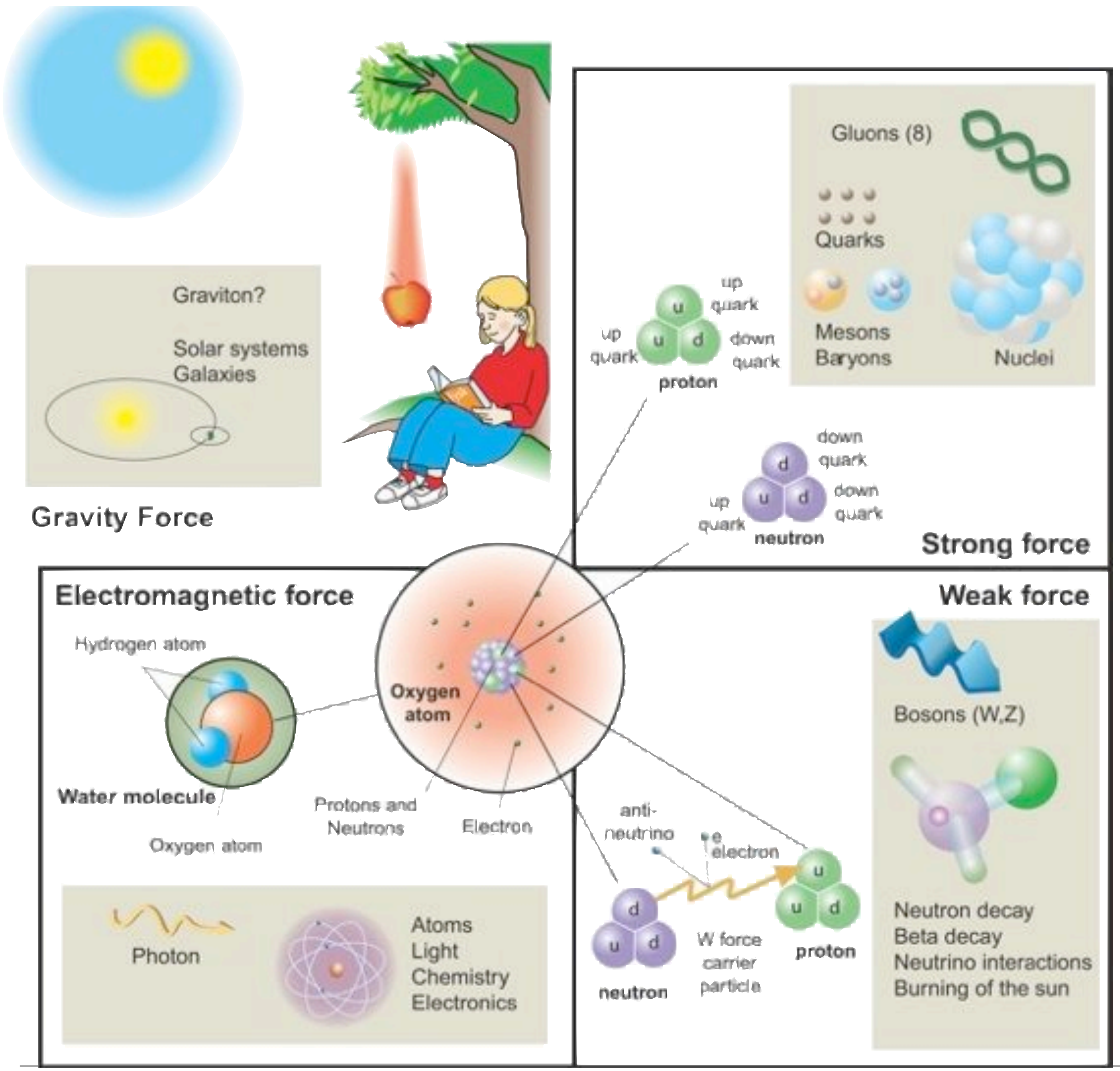
- Quantum ChromoDynamics
- Fluctuations from Heavy-Ion experiments and lattice QCD
- Lattice QCD on GPUs and on the Bielefeld GPU cluster
- Optimizations
  - includes first experiences with Kepler architecture
- Relating Lattice Data to Collider Experiments
- Outlook

→ Lattice-QCD talks by:
Frank Winter (Wed, 10:00)
Balint Joo (Wed,10:30)
Hyung-Jin Kim (Thu,16:30)

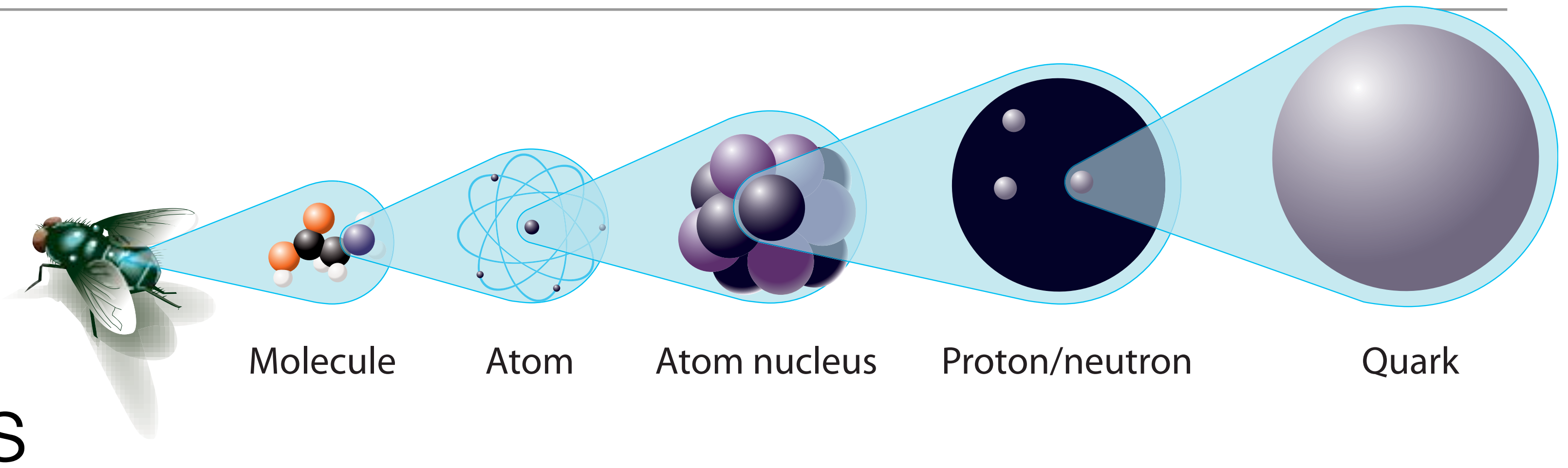
→ Lattice-QCD posters:
Hyung-Jin Kim
Richard Forster
Alexei Strelchenko

Universität Bielefeld

Strong force



- acts on quarks



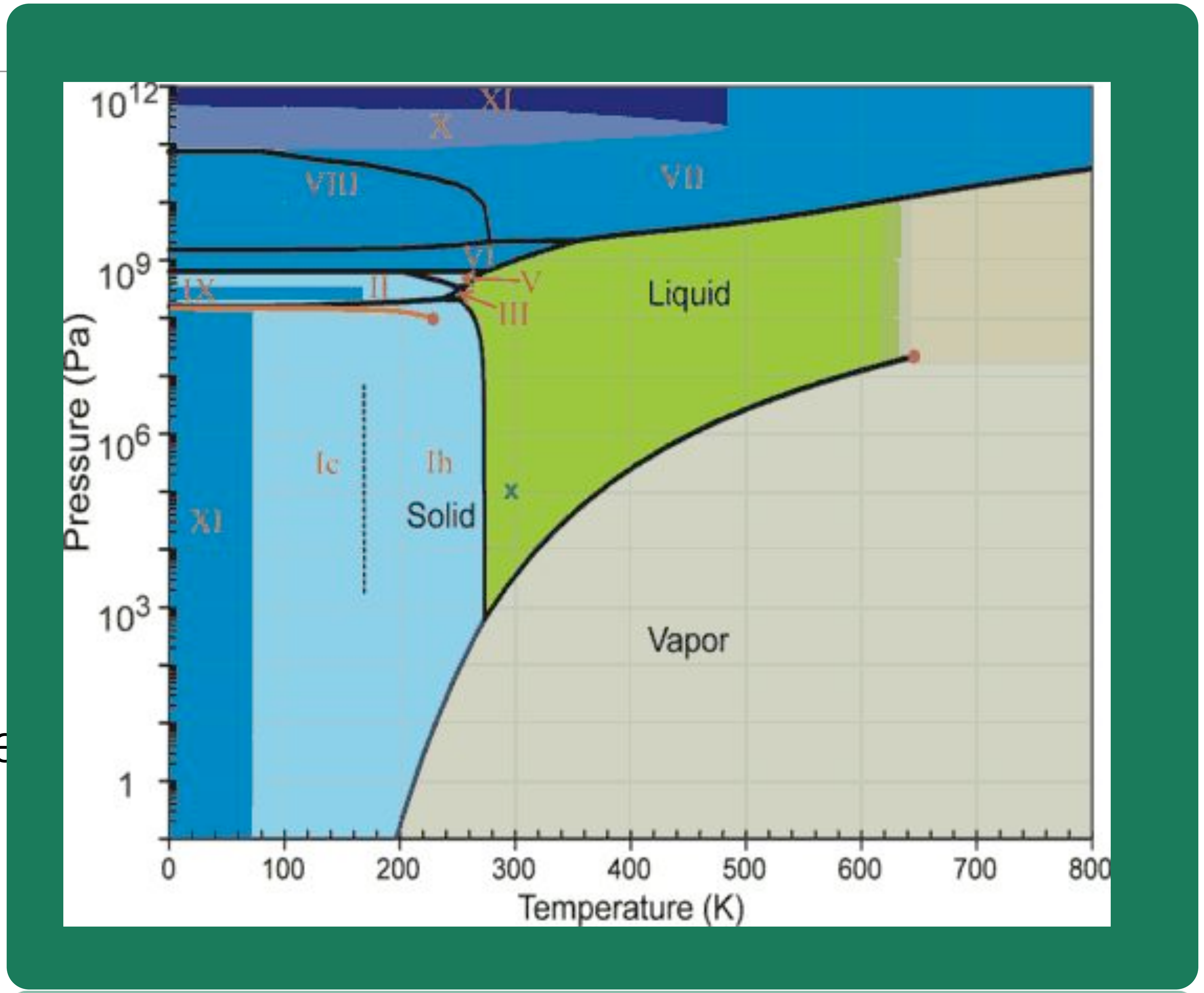
- force carriers: gluons
  (c.f. electrodynamics: photons)
- range: 10^{-15} m
- strength: 10^{38} times stronger than gravity
  10^{2} times stronger than electromagnetism
- residual interaction: nuclear force (i.e. force between nuclei in atom nucleus)
- described by Quantum ChromoDynamics (QCD)

Phase transitions

- water at different temperatures
    - ice (solid)
    - water (liquid)
    - vapor (gas)
- phase transitions occur in different ways: 1st order, 2nd order, ‘crossover’
- a ‘order parameter’ describes the change between different states
- boiling point of water depends on pressure → phase diagram

Phase transitions

- water at different temperatures
  - ice (solid)
  - water (liquid)
  - vapor (gas)
- phase transitions occur in different ways: 1st order, 2nd orde
- a ‘order parameter’ describes the change between different
- boiling point of water depends on pressure → phase diagram

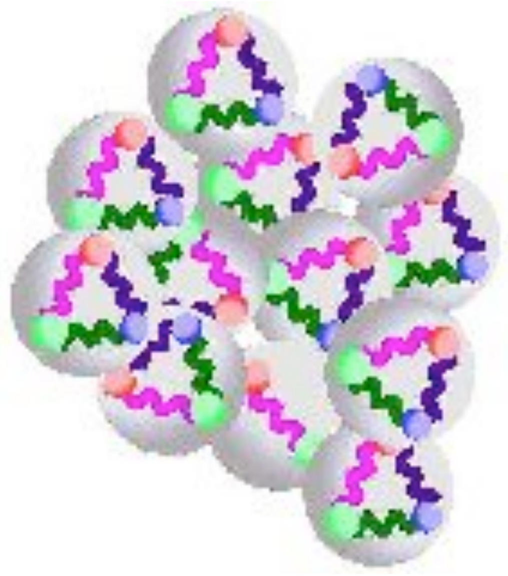


Universität Bielefeld

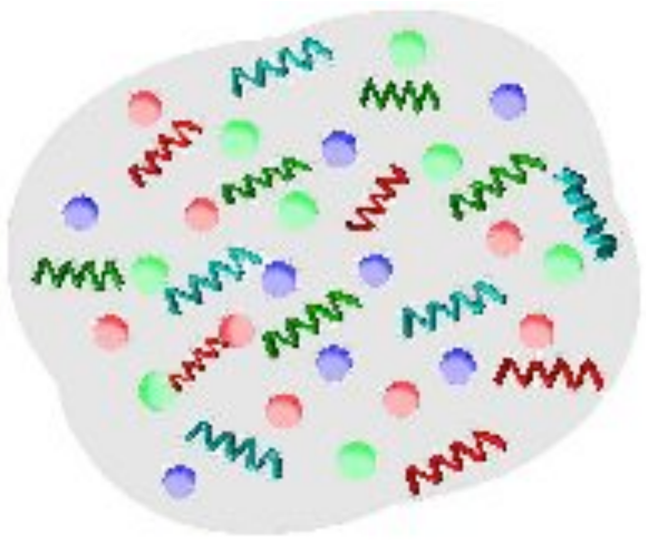
Phases of Quantum ChromDynamics

hadron gas

dense hadronic matter



quark gluon plasma



cold nuclear matter
Quarks and gluons are
confined inside hadrons

phase transition or
crossover at Tc

quarks and gluons are
the degrees of freedom
(asymptotically) free

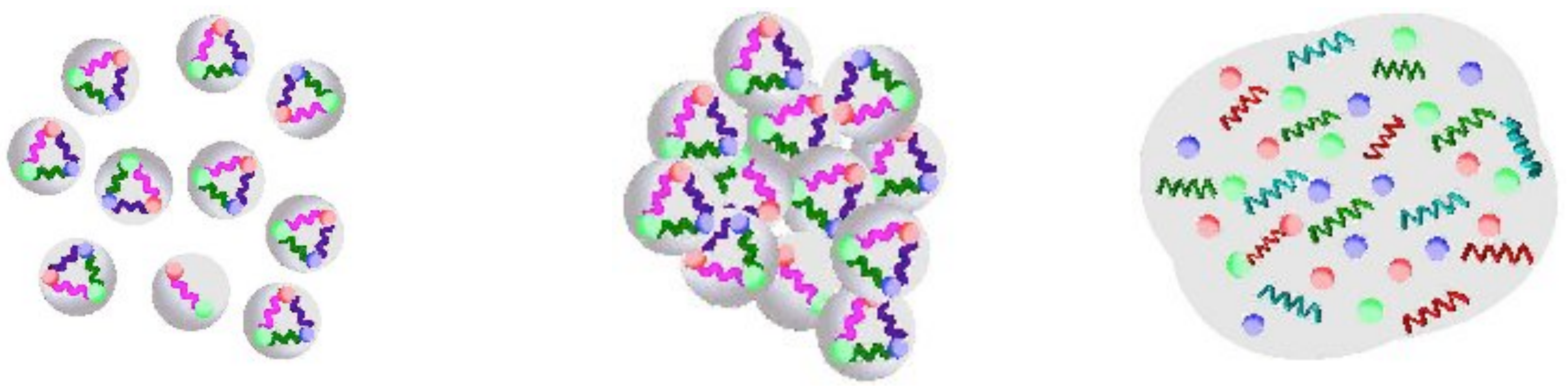
- extreme conditions (temperatures, densities) are necessary to investigate properties of QCD

- important for understanding the evolution of the universe after the Big Bang

Universität Bielefeld

Phases of Quantum ChromDynamics

hadron gas | dense hadronic matter | quark gluon plasma



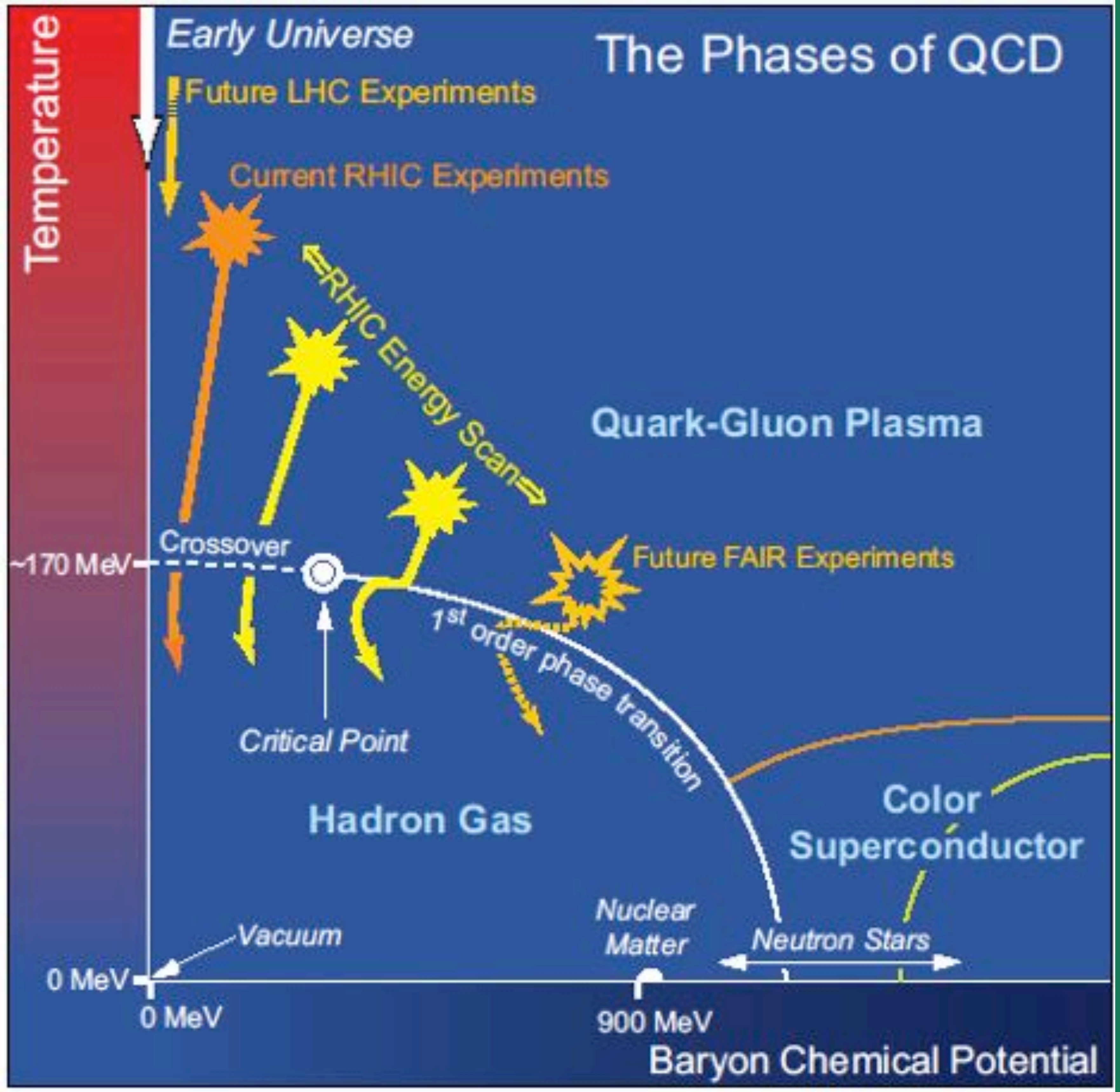
cold nuclear matter
Quarks and gluons are
confined inside hadrons

phase transition or
crossover at Tc

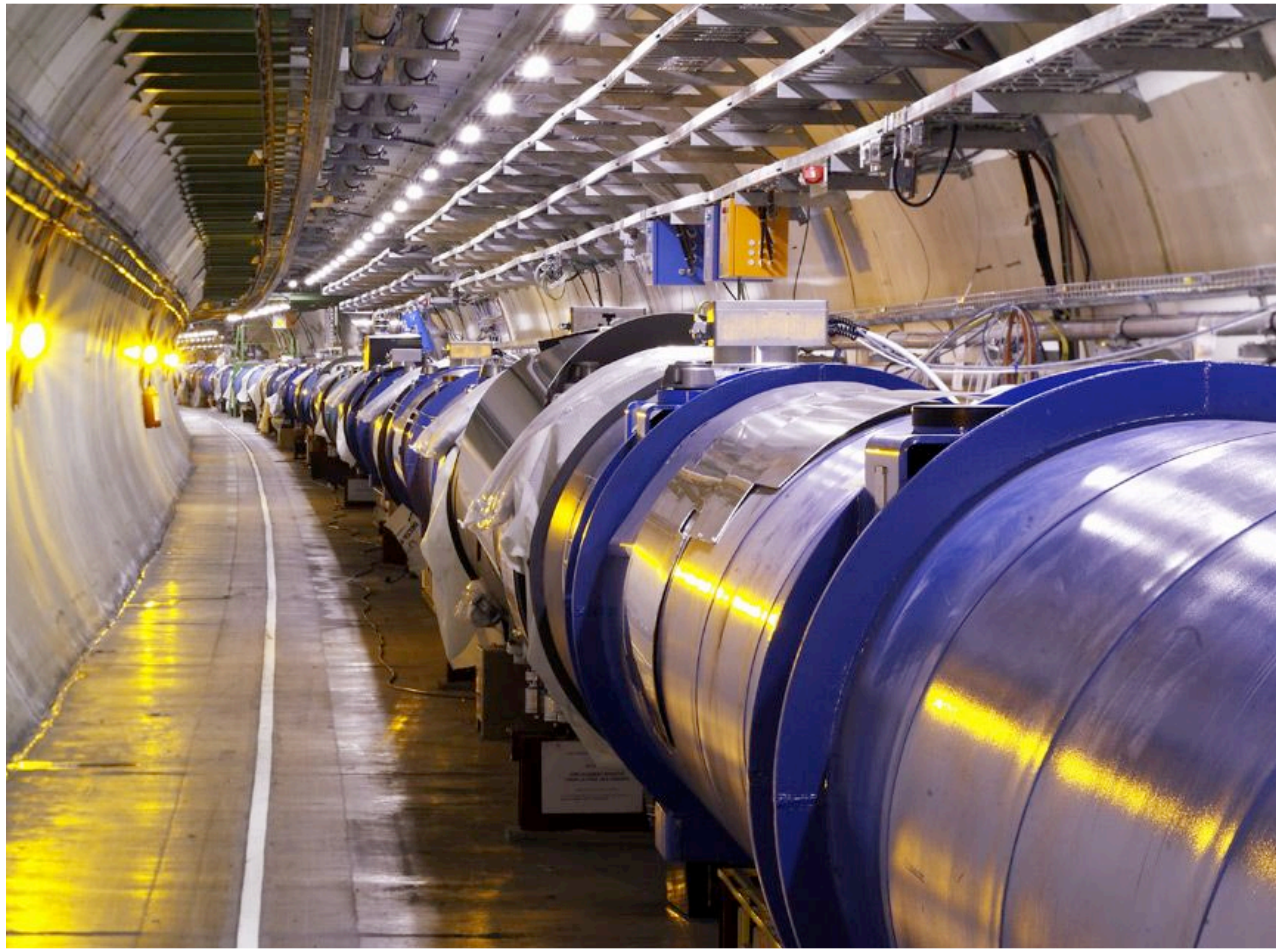
quarks and gluons are
the degrees of freedom
(asymptotically) free

- extreme conditions (temperatures, densities) are necessary to investigate properties of QCD

- important for understanding the evolution of the universe after the Big Bang



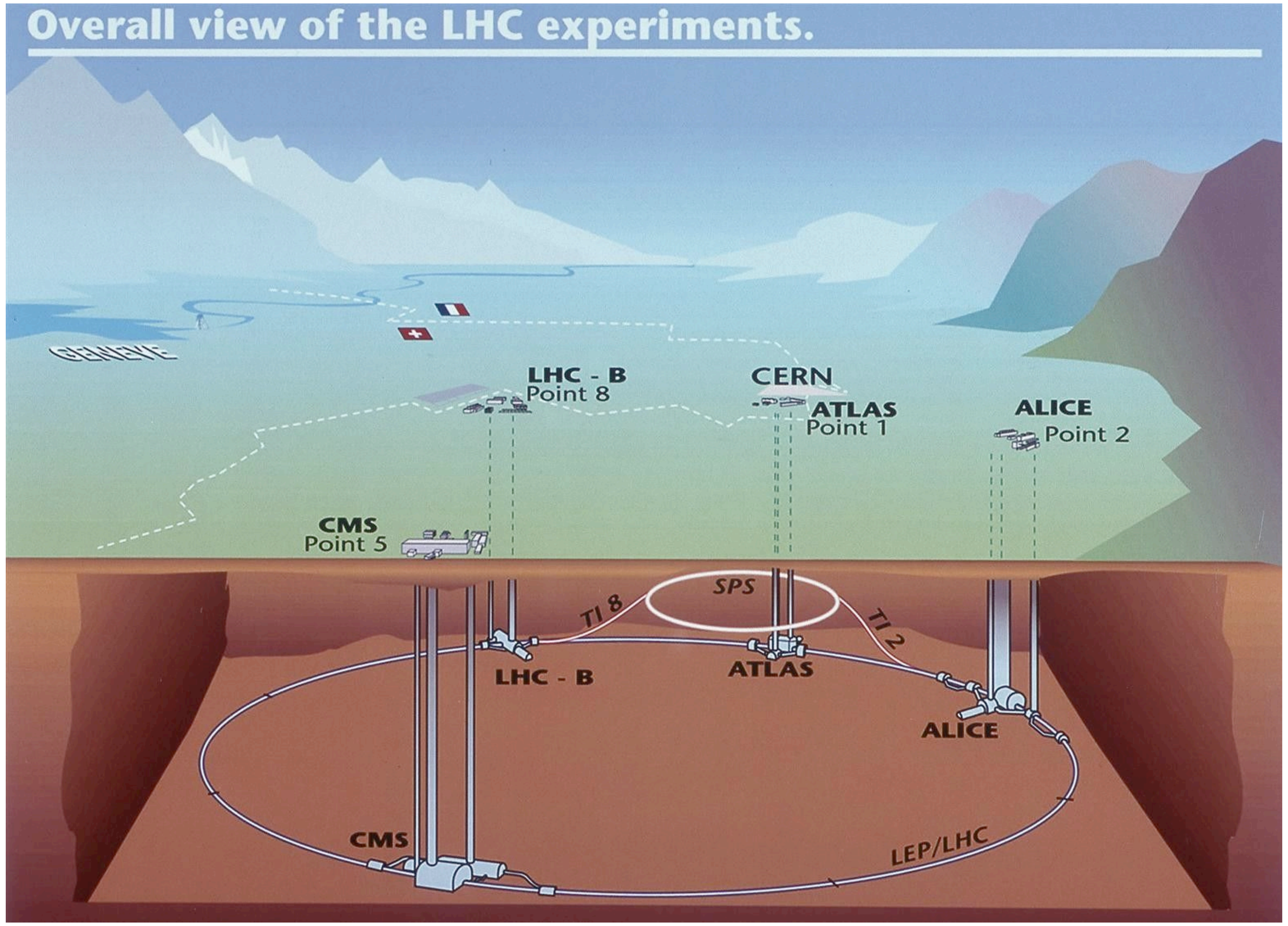
Accelerators ... the real ones



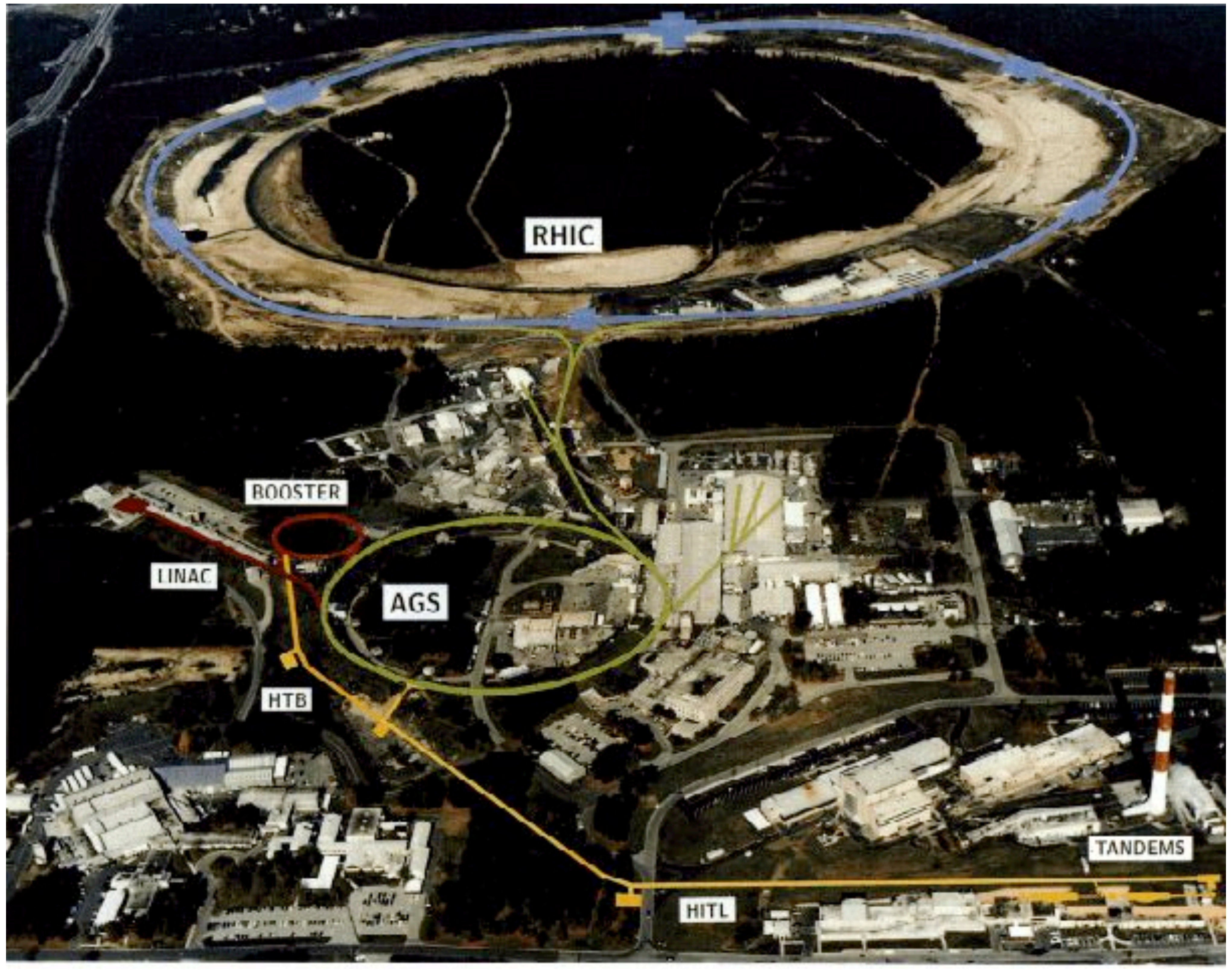
LHC @ CERN

RHIC @ Brookhaven National Lab

Accelerators ... the real ones



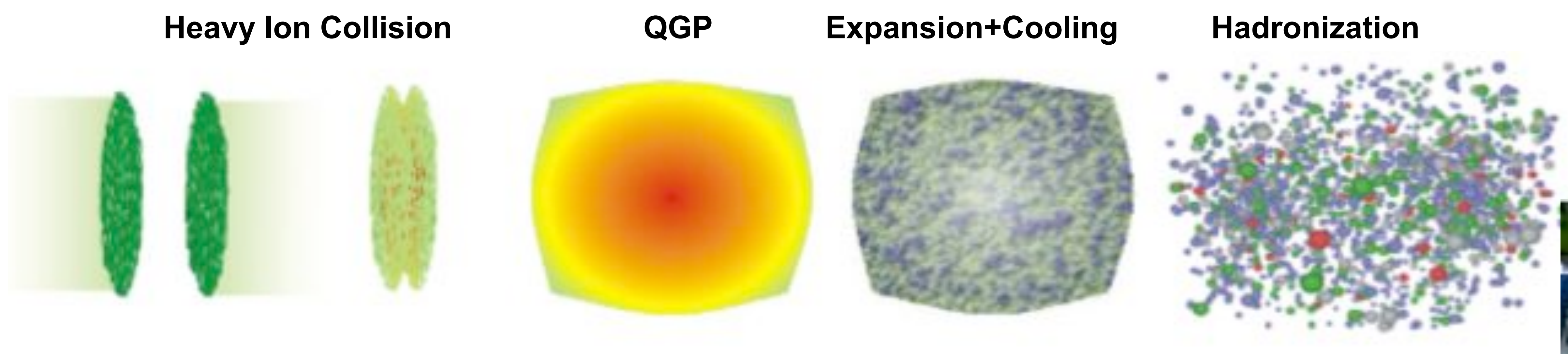
LHC @ CERN



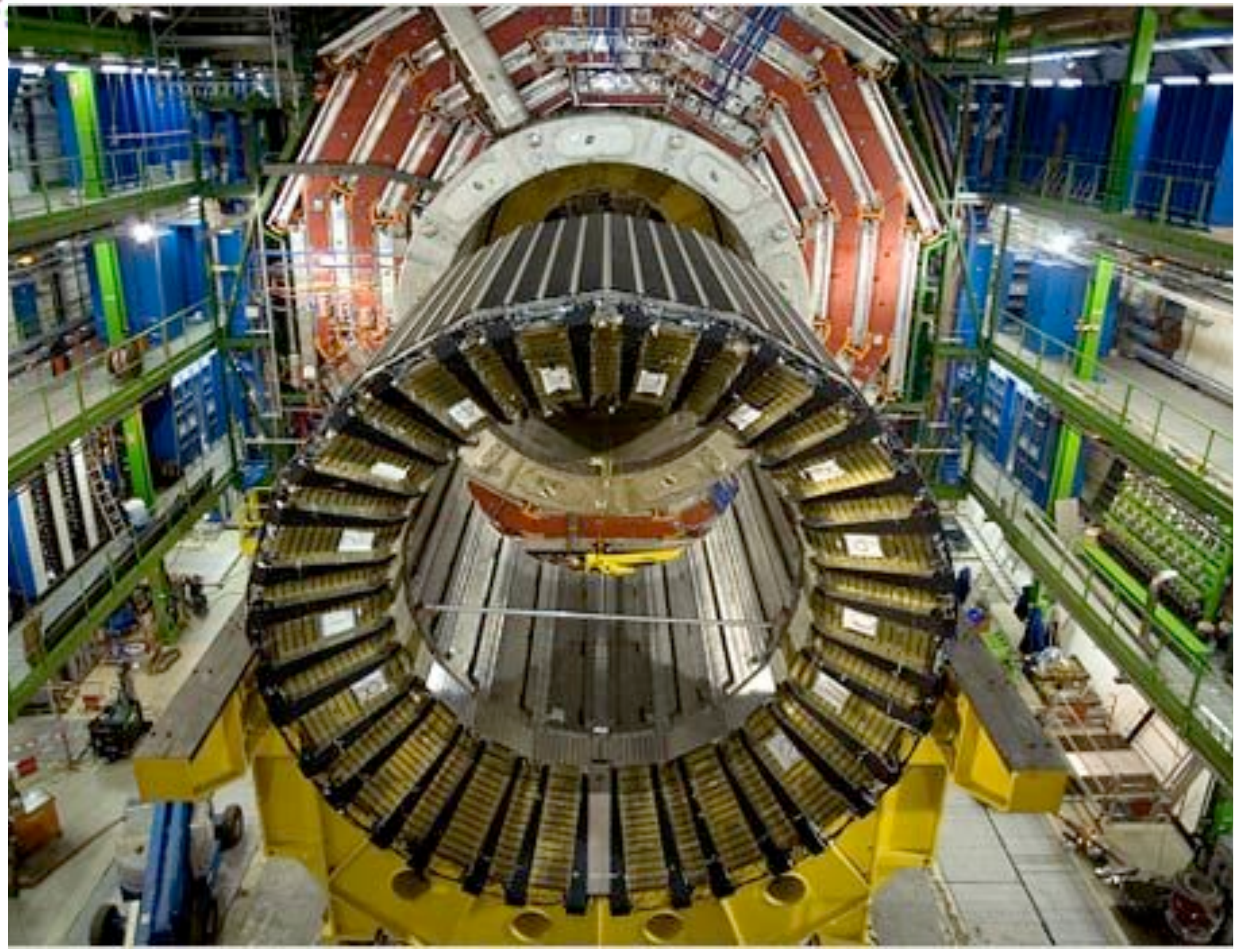
RHIC @ Brookhaven National Lab

Universität Bielefeld

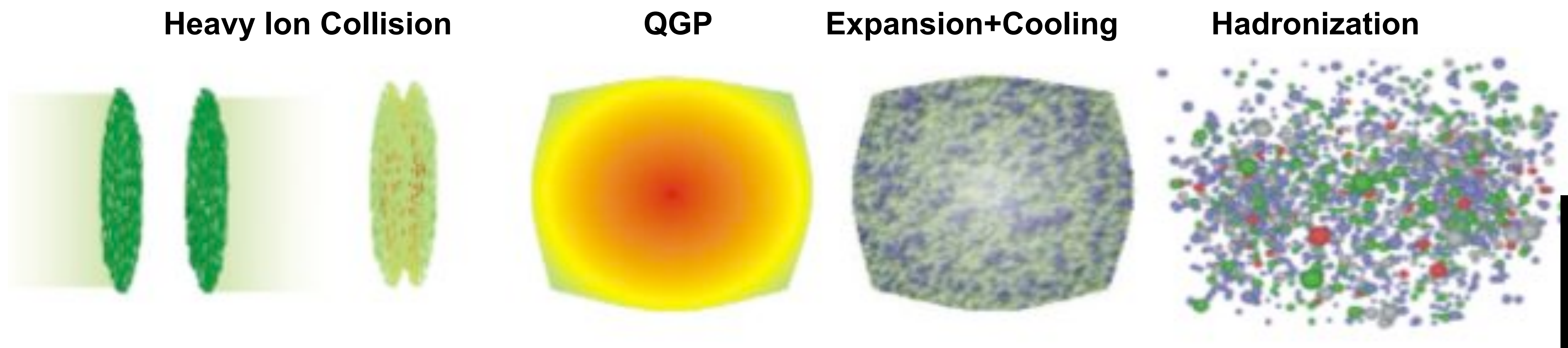
Heavy Ion Experiments



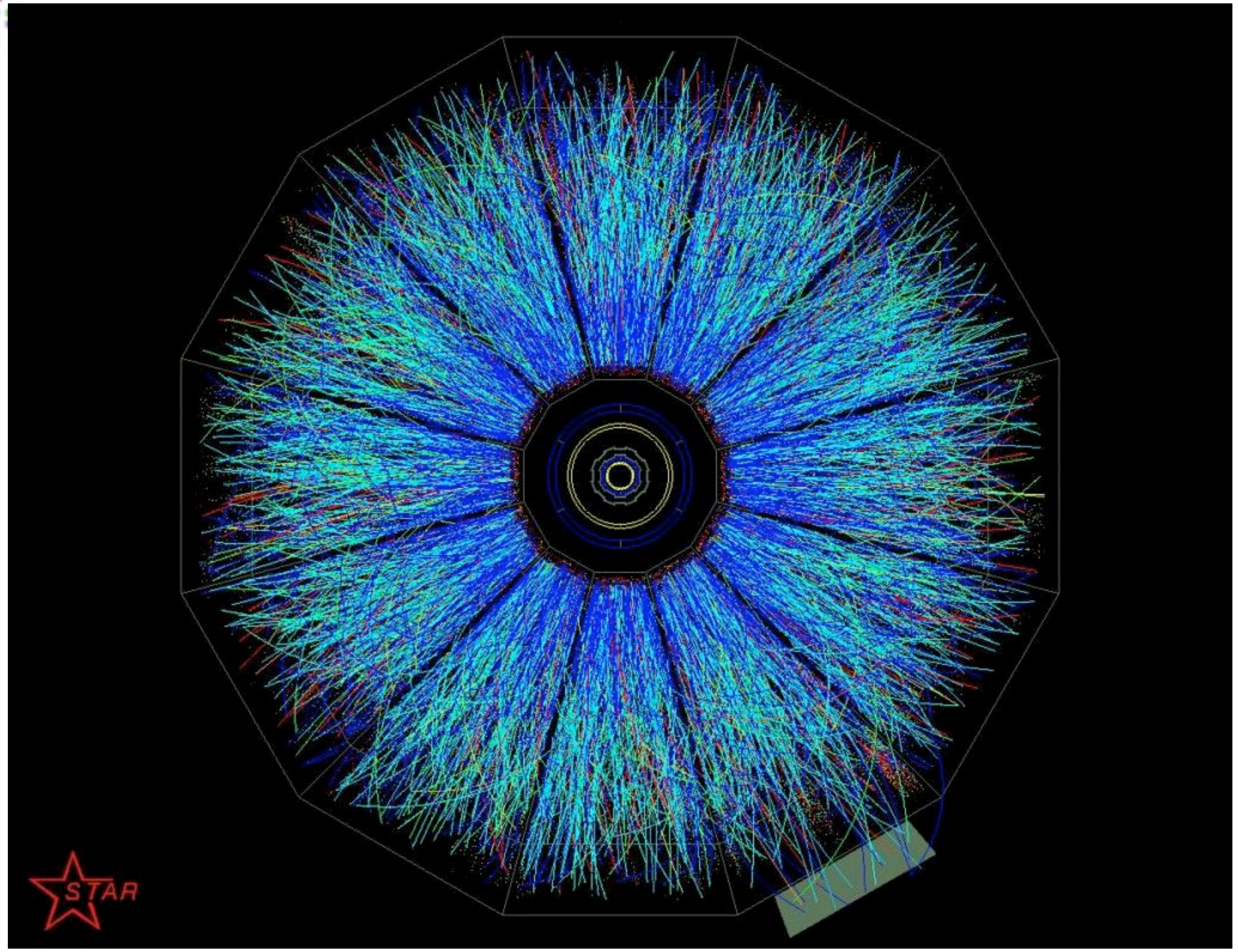
- phase transition occurs in heavy-ion collisions
- What thermometer can we use at $10^{12}\,\mathrm{K}$?
- detectors measure created particles
- to interpret the data theoretical input is required
- ab-initio approach: Lattice QCD



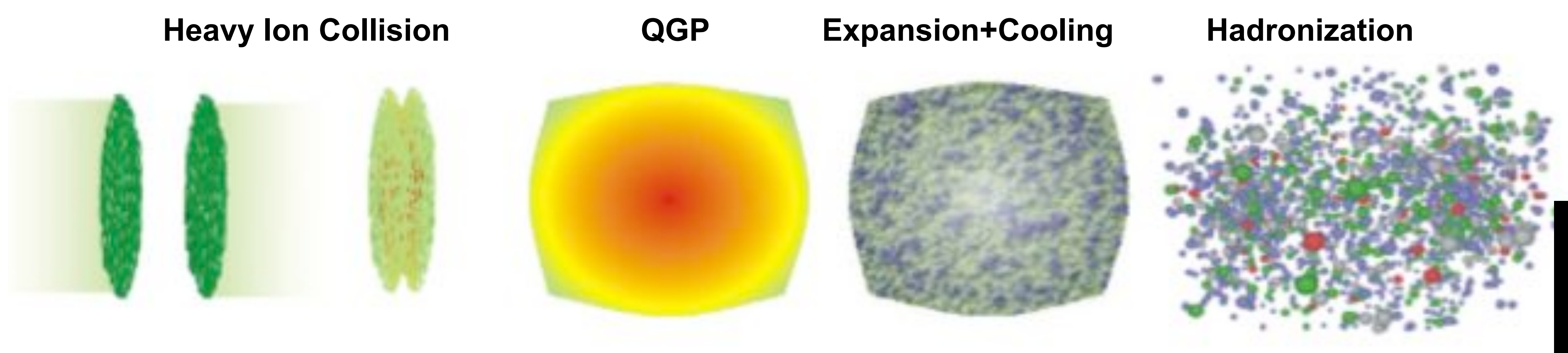
Heavy Ion Experiments



- phase transition occurs in heavy-ion collisions
- What thermometer can we use at $10^{12}\,\mathrm{K}$?
- detectors measure created particles
- to interpret the data theoretical input is required
- ab-initio approach: Lattice QCD

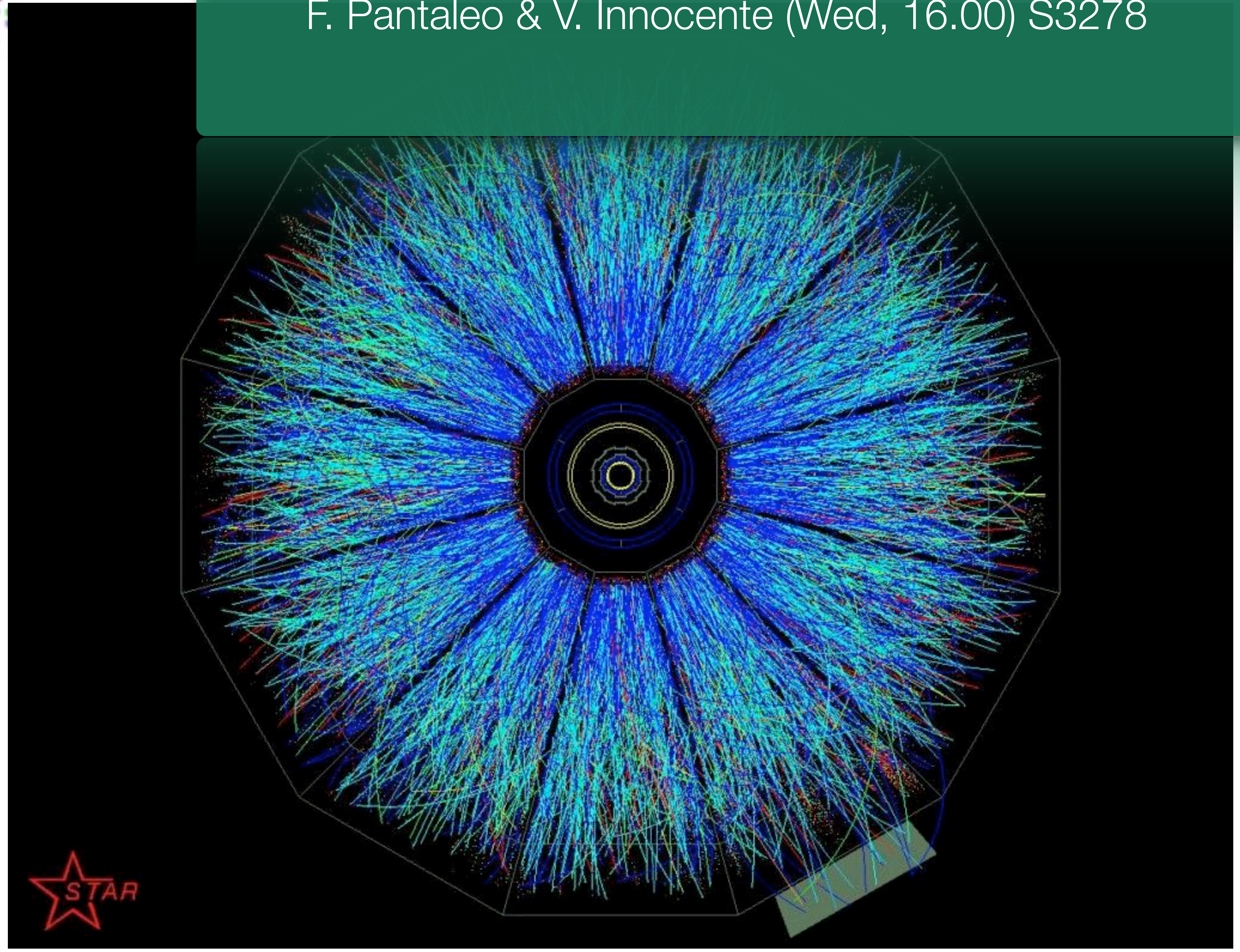


Heavy Ion Experiments



GPUs used for triggering and data processing
→ Valerie Halyo (Wed, 16.30) S3263
Alessandro Lonardo (Wed, 15.30) S3286
F. Pantaleo & V. Innocente (Wed, 16.00) S3278

- phase transition occurs in heavy-ion collisions
- What thermometer can we use at $10^{12}\,\mathrm{K}$?
- detectors measure created particles
- to interpret the data theoretical input is required
- ab-initio approach: Lattice QCD



Universität Bielefeld

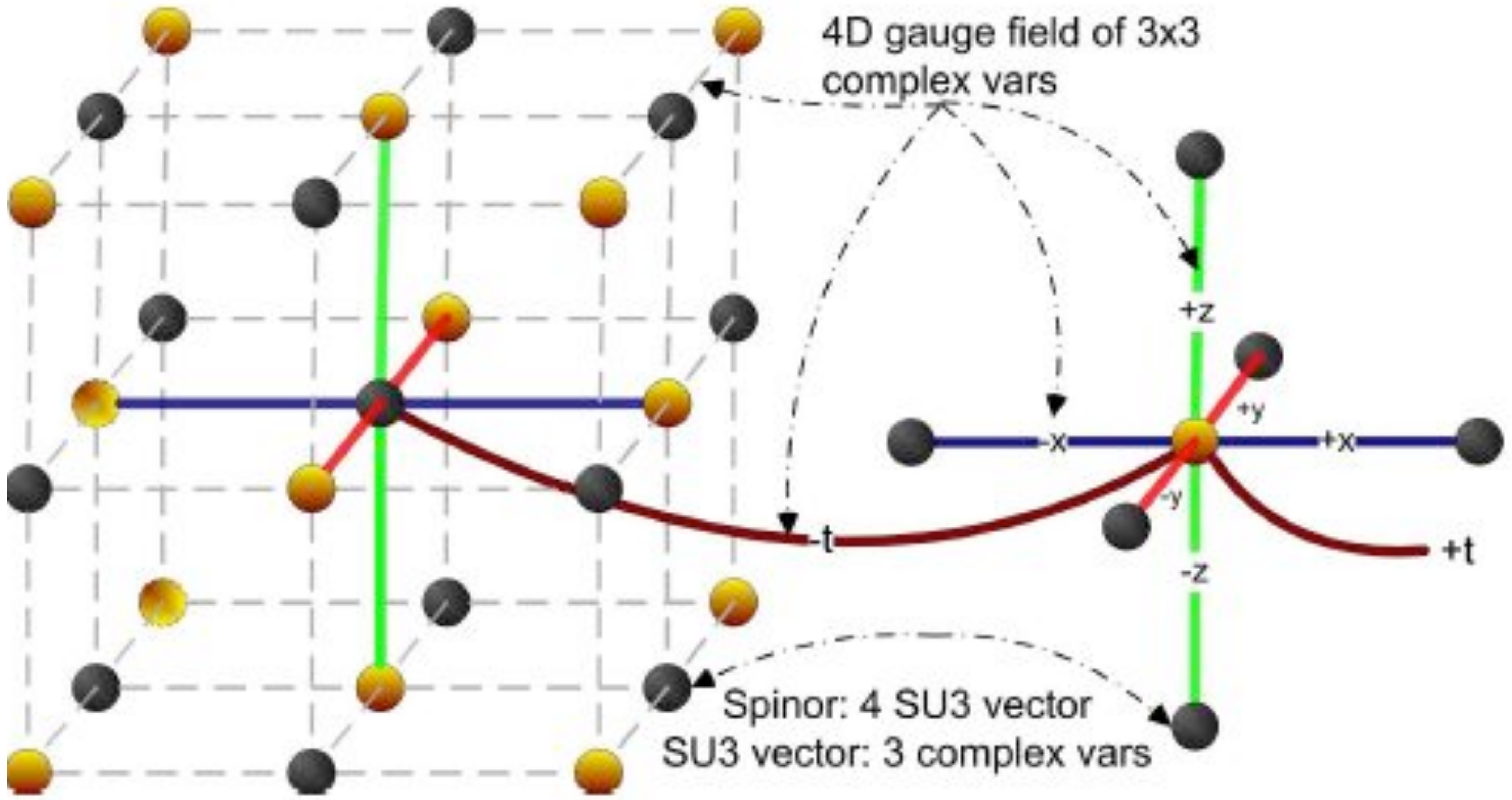
Lattice QCD

- QCD partition function

$$Z_{\text{QCD}}(T,\mu) = \int DAD\bar{\Psi}D\Psi e^{-S_E(T,\mu)}$$

includes integral over space and time

- 4 dimensional grid (=Lattice)
- quarks live on lattice sites
  - 6 or 12 complex numbers
- gluons live on the links
  - SU(3) matrices
  - 18 complex numbers
- typical sizes: 24 x 24 x 24 x 6 to 256 x 256 x 256 x 256

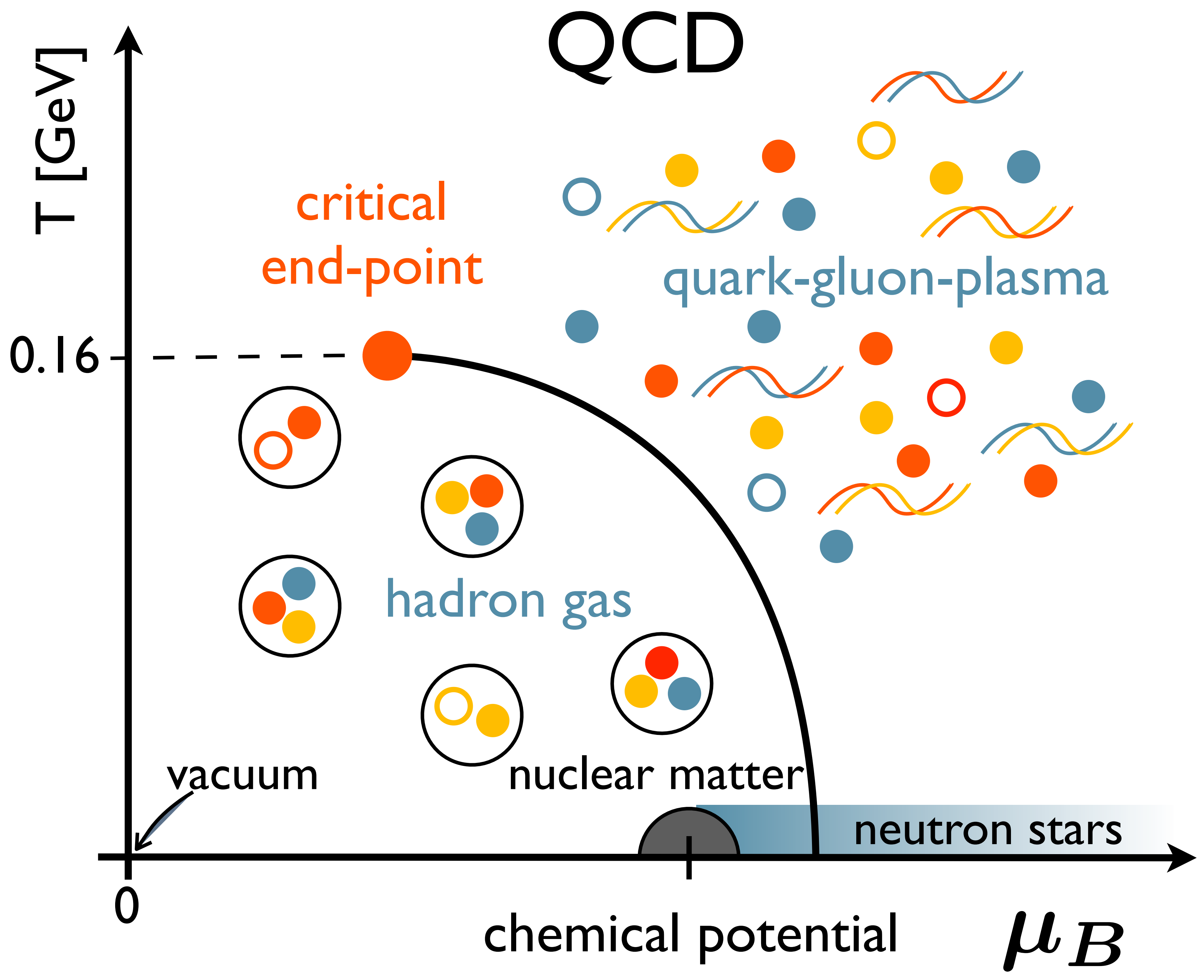


Universität Bielefeld

Fluctuations and the QCD phase diagram

- different QCD phases characterized by
  - chiral symmetry
  - confinement aspects

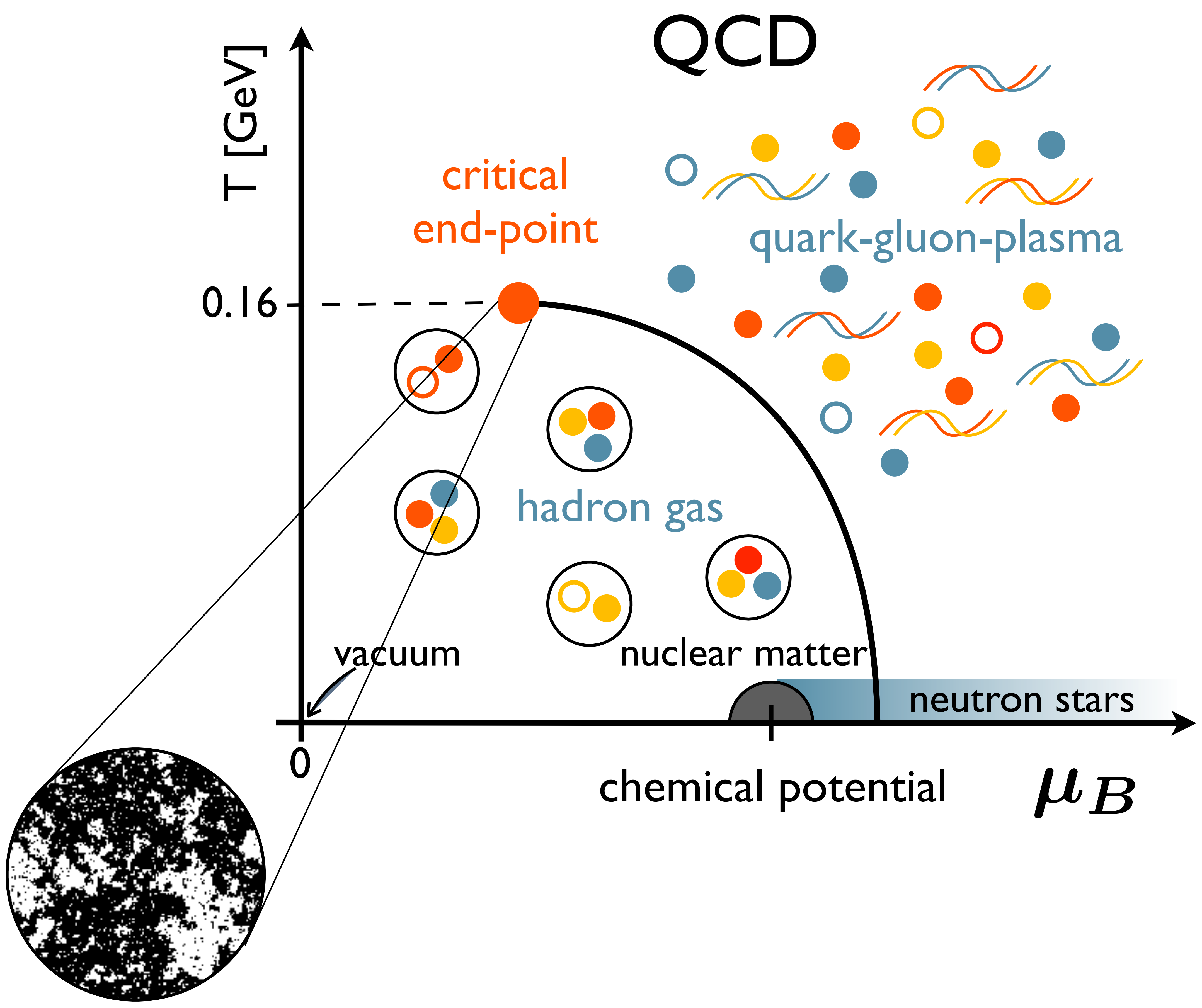
Figure from C. Schmidt



Fluctuations and the QCD phase diagram

- different QCD phases characterized by
  - chiral symmetry
  - confinement aspects
- possible critical end-point
  - 2nd order phase transition
  - divergent correlation length
  - divergent susceptibility

Figure from C. Schmidt



Universität Bielefeld

Fluctuations from Lattice QCD

- expansion of the pressure in

$$\frac{p}{T^4} = \sum_{i,j,k}^{\infty} \frac{1}{i!j!k!} \chi_{ijk}^{BQS} \left(\frac{\mu_B}{T}\right)^i \left(\frac{\mu_Q}{T}\right)^j \left(\frac{\mu_S}{T}\right)^k$$

- B,Q,S conserved charges (baryon number, electric charge, strangeness)

Fluctuations from Lattice QCD

- expansion of the pressure in

$$\frac{p}{T^4} = \sum_{i,j,k}^{\infty} \frac{1}{i!j!k!} \chi_{ijk}^{BQS} \left(\frac{\mu_B}{T}\right)^i \left(\frac{\mu_Q}{T}\right)^j \left(\frac{\mu_S}{T}\right)^k$$

- B,Q,S conserved charges (baryon number, electric charge, strangeness)

- generalized susceptibilities

$$\chi_{ijk}^{BQS} = \frac{1}{VT} \frac{\partial^i}{\partial(\mu_B/T)} \frac{\partial^j}{\partial(\mu_Q/T)} \frac{\partial^k}{\partial(\mu_S/T)} \mathcal{Z}(T,\mu)\Big|_{\mu=0}$$

Universität Bielefeld

Fluctuations from Lattice QCD

- expansion of the pressure in

$$\frac{p}{T^4} = \sum_{i,j,k}^{\infty} \frac{1}{i!j!k!} \chi_{ijk}^{BQS} \left(\frac{\mu_B}{T}\right)^i \left(\frac{\mu_Q}{T}\right)^j \left(\frac{\mu_S}{T}\right)^k$$

- B,Q,S conserved charges (baryon number, electric charge, strangeness)

- generalized susceptibilities

$$\chi_{ijk}^{BQS} = \frac{1}{VT} \frac{\partial^i}{\partial(\mu_B/T)} \frac{\partial^j}{\partial(\mu_Q/T)} \frac{\partial^k}{\partial(\mu_S/T)} \mathcal{Z}(T,\mu)\Big|_{\mu=0}$$

- related to cumulants of net charge fluctuations, e.g.

$$VT^3\chi_2^B = \langle(\delta N_B)^2\rangle = \left\langle N_B^2 - 2N_B\langle N_B\rangle + \langle N_B\rangle^2\right\rangle$$

Calculation of susceptibilities from Lattice QCD

- μ-dependence is contained in the fermion determinant

$$\mathcal{Z} = \int \mathcal{D}U (\det M(\mu))^{N_\mathrm{f}/4} \exp(-S_g),$$

- calculation of susceptibilities requires μ-derivatives of fermion determinant

$$\frac{\partial^2 \ln \mathcal{Z}}{\partial \mu^2} = \left\langle \frac{n_f}{4} \frac{\partial^2 (\ln \det M)}{\partial \mu^2} \right\rangle + \left\langle \left(\frac{n_f}{4} \frac{\partial (\ln \det M)}{\partial \mu} \right)^2 \right\rangle$$

Calculation of susceptibilities from Lattice QCD

- μ-dependence is contained in the fermion determinant

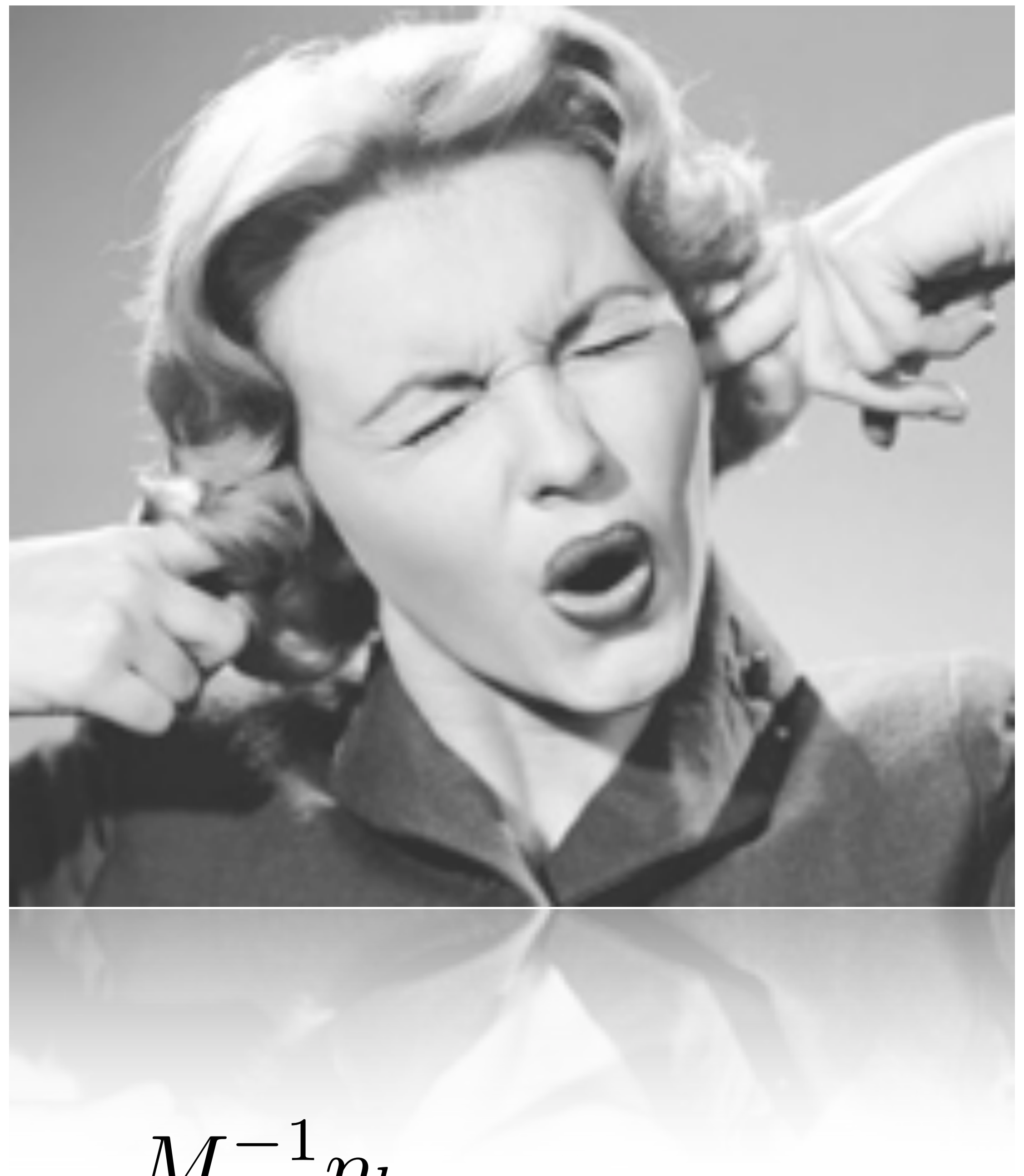
$$\mathcal{Z} = \int \mathcal{D}U (\det M(\mu))^{N_\mathrm{f}/4} \exp(-S_g),$$

- calculation of susceptibilities requires μ-derivatives of fermion determinant

$$\frac{\partial^2 \ln \mathcal{Z}}{\partial \mu^2} = \left\langle \frac{n_f}{4} \frac{\partial^2 (\ln \det M)}{\partial \mu^2} \right\rangle + \left\langle \left(\frac{n_f}{4} \frac{\partial (\ln \det M)}{\partial \mu} \right)^2 \right\rangle$$

- formulate all operator in terms of traces over space-time, color (and spin)
  - full inversion of fermion matrix is impossible: evaluate using noisy estimators
  - ensemble average → large number of configurations

Noisy estimators



- traces required for derivatives

$$\frac{\partial(\ln\det M)}{\partial\mu} = \mathrm{Tr}\left(M^{-1}\frac{\partial M}{\partial\mu}\right)$$
$$\frac{\partial^2(\ln\det M)}{\partial\mu^2} = \mathrm{Tr}\left(M^{-1}\frac{\partial^2 M}{\partial\mu^2}\right) - \mathrm{Tr}\left(M^{-1}\frac{\partial M}{\partial\mu}M^{-1}\frac{\partial M}{\partial\mu}\right)$$

- noisy estimators → large number of random vectors η (~**1500** / configuration)

$$\mathrm{Tr}\left(\frac{\partial^{n_1}M}{\partial\mu^{n_1}}M^{-1}\frac{\partial^{n_2}M}{\partial\mu^{n_2}}\ldots M^{-1}\right) = \lim_{N\to\infty}\frac{1}{N}\sum_{k=1}^{N}\eta_k^\dagger\frac{\partial^{n_1}M}{\partial\mu^{n_1}}M^{-1}\frac{\partial^{n_2}M}{\partial\mu^{n_2}}\ldots M^{-1}\eta_k$$

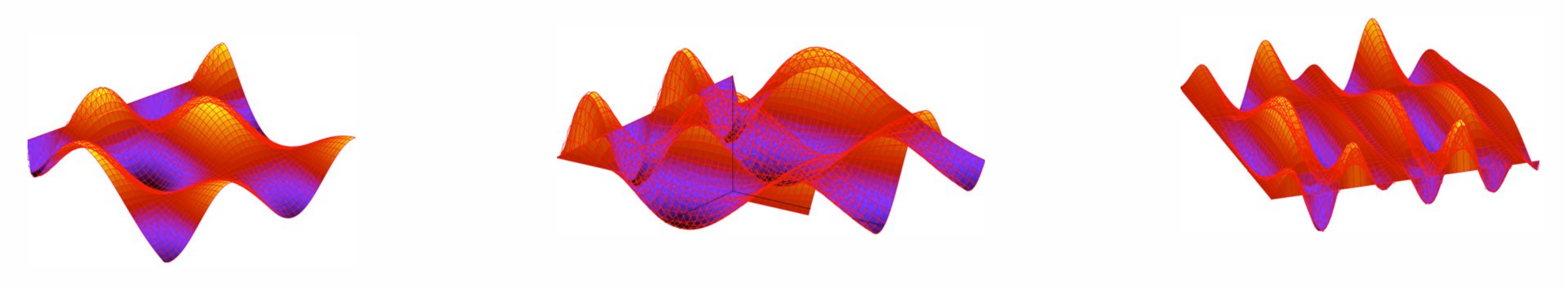
- up to **10000** configurations for each temperature

- dominant operation: fermion matrix inversion (~ 99%)

Universität Bielefeld

Configuration generation

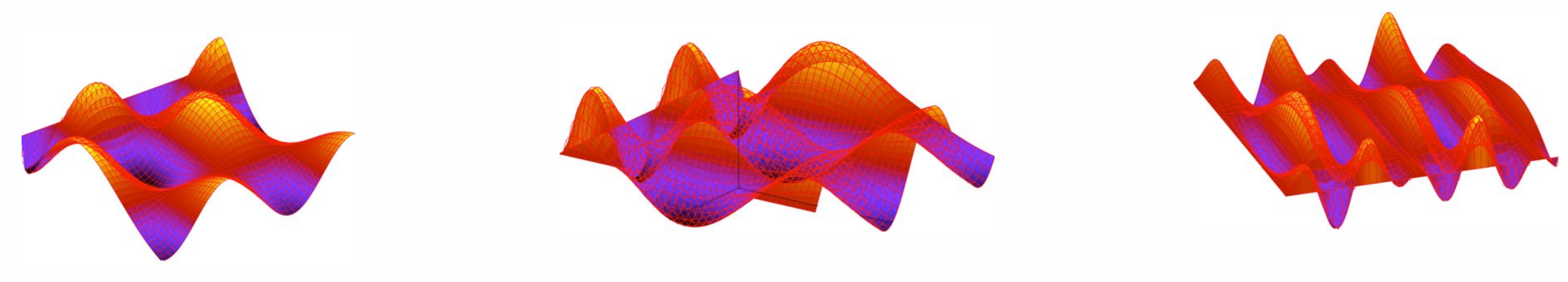
- sequential process
- use RHMC algorithm to evaluate the system in *simulation time*



$$\dot{P} = -\frac{\partial H}{\partial Q} = -\frac{\partial S}{\partial Q} = -\left(\frac{\partial S_g}{\partial Q} + \frac{\partial S_f}{\partial Q}\right) \qquad \dot{Q} = \frac{\partial H}{\partial P} = P$$

Configuration generation

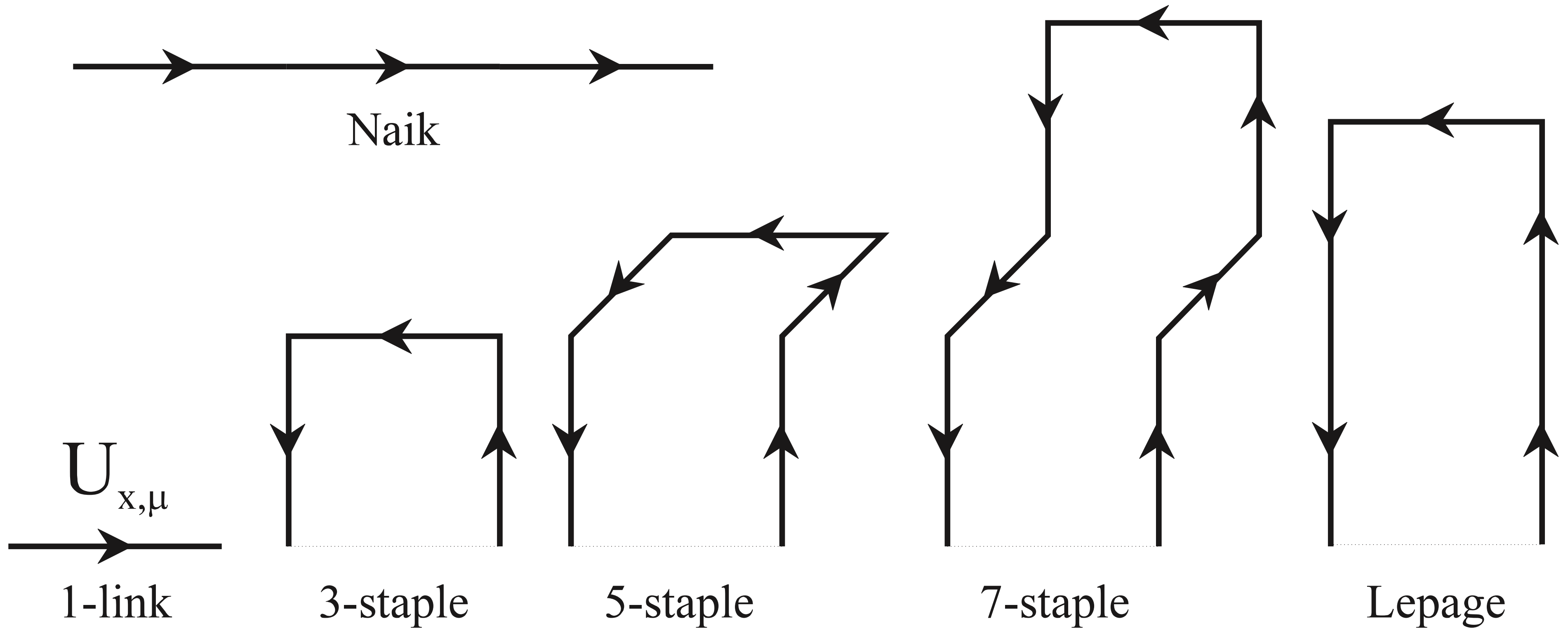
- sequential process



- use RHMC algorithm to evaluate the system in *simulation time*

$$\dot{P} = -\frac{\partial H}{\partial Q} = -\frac{\partial S}{\partial Q} = -\left(\frac{\partial S_g}{\partial Q} + \frac{\partial S_f}{\partial Q}\right) \qquad \dot{Q} = \frac{\partial H}{\partial P} = P$$

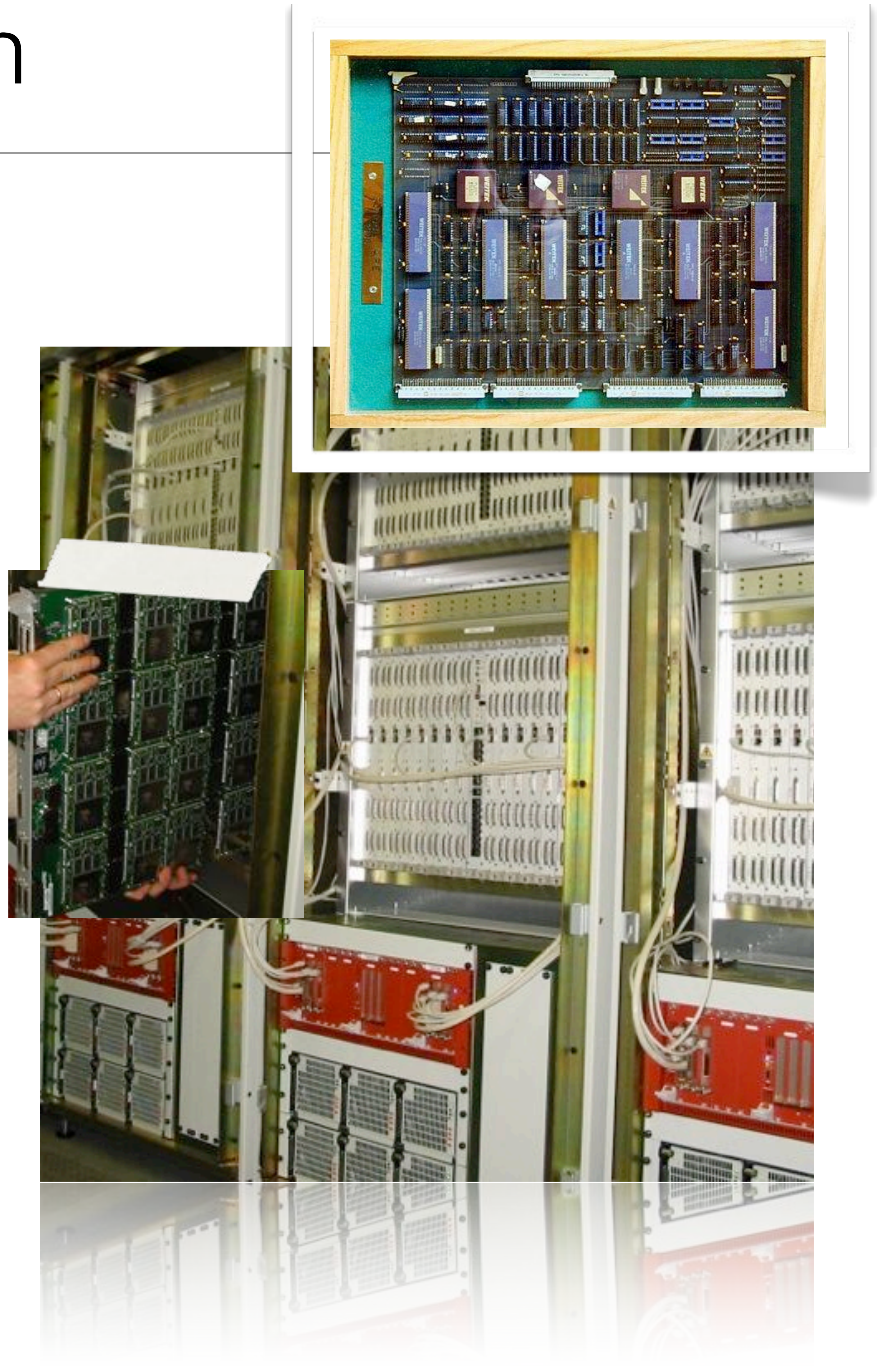
- two dominant parts of the calculation (90% of the runtime)
  - fermion force
    ~50% for improved actions (HISQ)
  - fermion matrix inversion
    ~90% for standard action



Universität Bielefeld

History of QCD Machines in BI: the APE generation

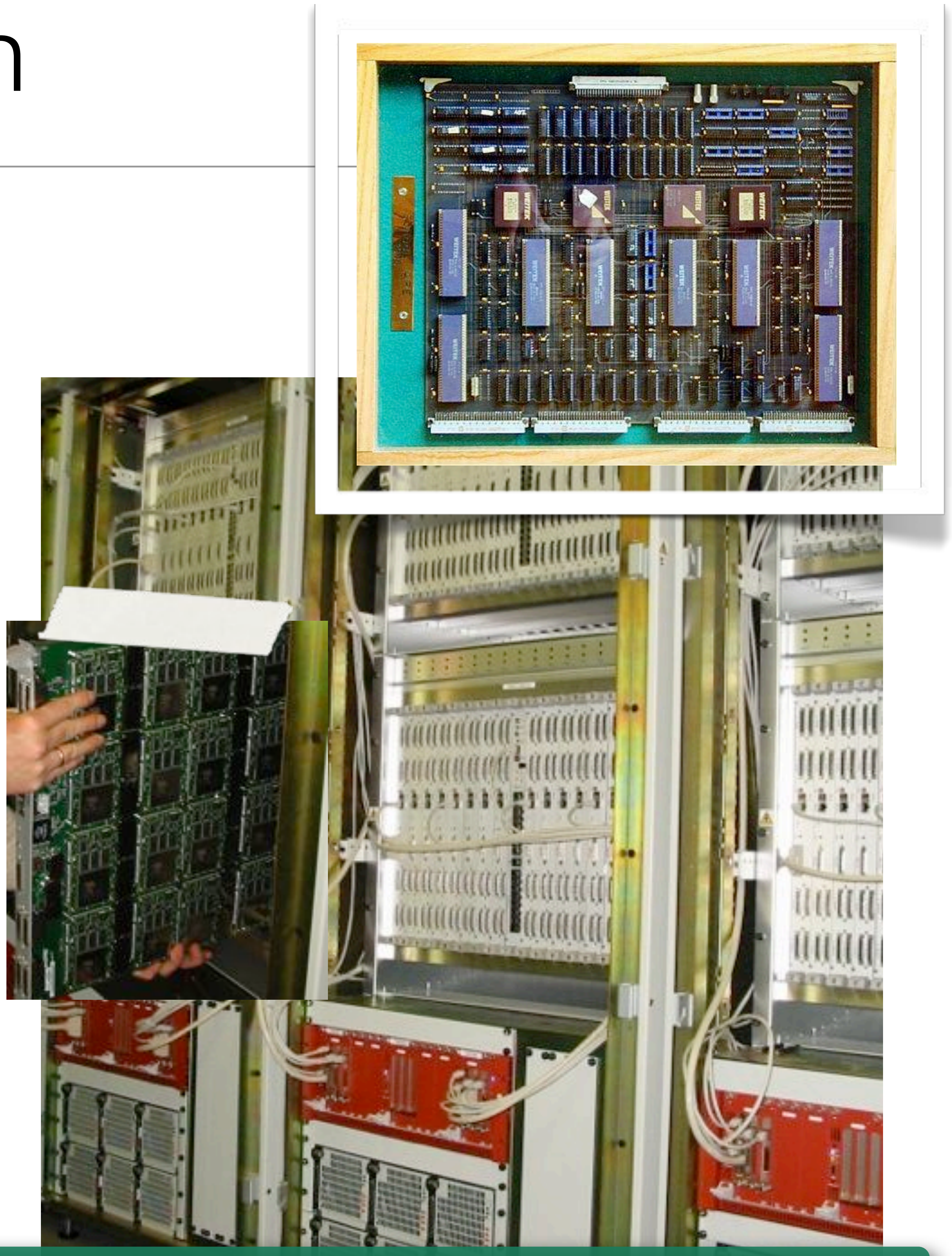
- APE = Array Processor Experiment, started mid eigthties
- SIMD architecture with lot of FPUs, VLIW
- special purpose machine build for lattice QCD
  - optimized a x b + c operation for use in complex matrix-vector multiplication
  - large register files - up to 512 64bit-registers
  - 3D network low latency: fast memory access to nearest neighbor (~ 3-4 local)
- low power consumption (latest version: ~ 1.5 GFlops @ 7 Watt)
- object-oriented programming language TAO (syntax similar to Fortran)
- controlled by host PC



Universität Bielefeld

History of QCD Machines in BI: the APE generation

- APE = Array Processor Experiment, started mid eigthties
- SIMD architecture with lot of FPUs, VLIW
- special purpose machine build for lattice QCD
  - optimized a x b + c operation for use in complex matrix-vector multiplication
  - large register files - up to 512 64bit-registers
  - 3D network low latency: fast memory access to nearest neighbor (~ 3-4 local)
- low power consumption (latest version: ~ 1.5 GFlops @ 7 Watt)
- object-oriented programming language TAO (syntax similar to Fortran)
- controlled by host PC

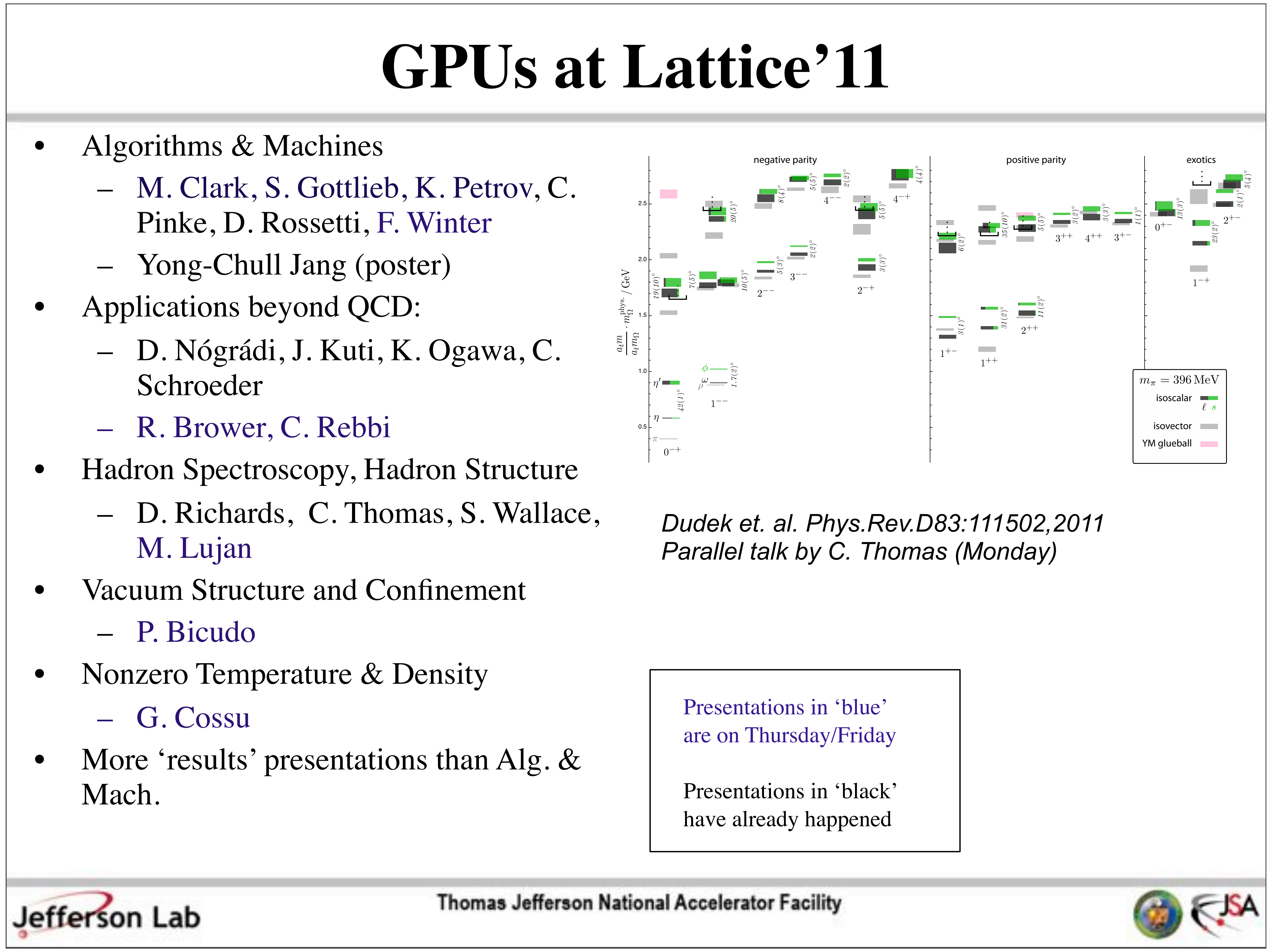


Talk on APEnet
→ Massimo Bernaschi (Tue, 16.00) S3089

Future of QCD machines in BI: the GPU era

- lattice simulations are massively parallel
- require a lot of floating point operations
- used as accelerators since 2006: ‘QCD as a video game’ (Erigi et al), coded in OpenGL
- GPUs become standard ‘tool’ of Lattice QCD
- widely used by various groups
- libraries available (e.g. QUDA)

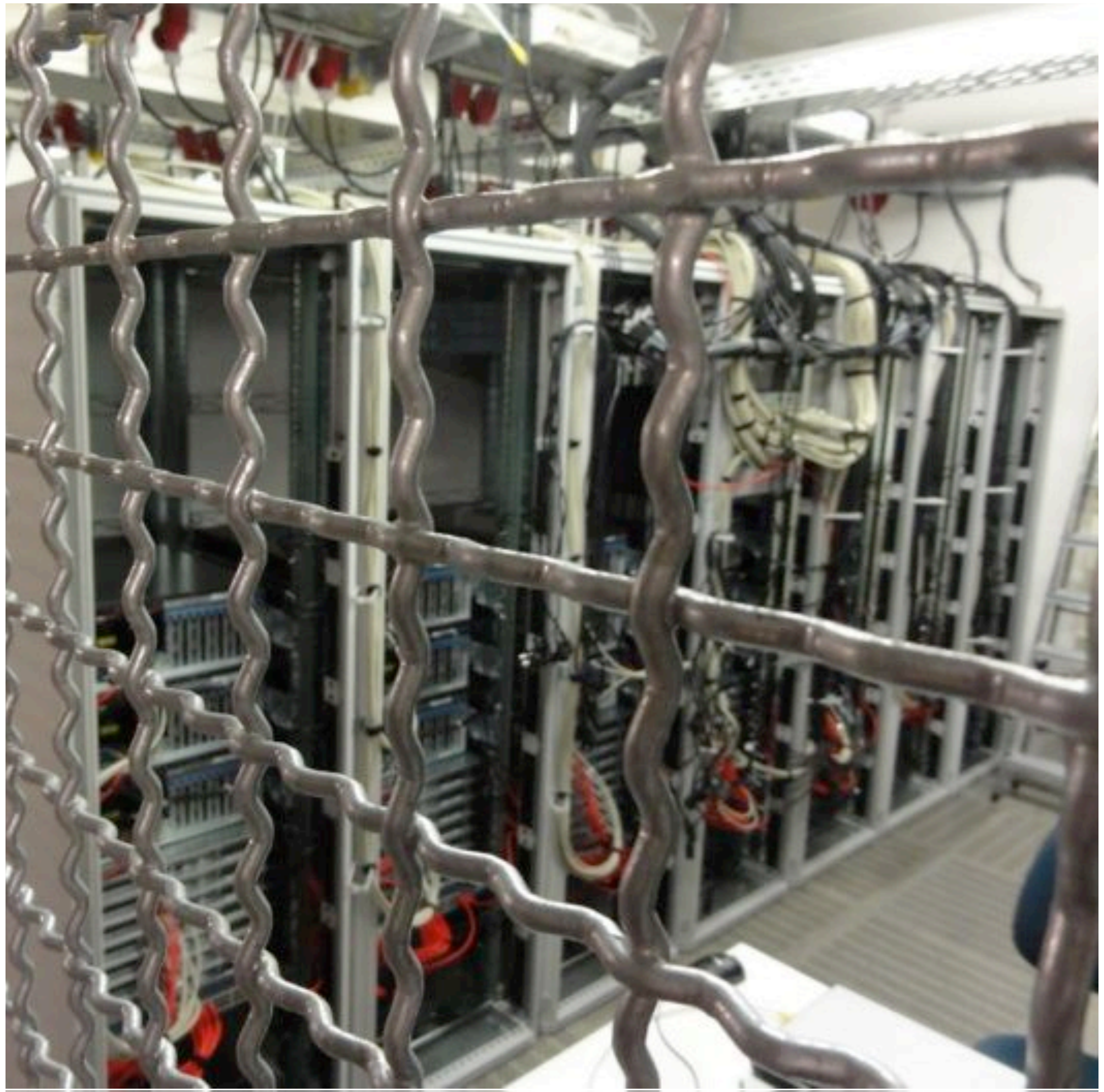
Slide from Balint Joo, Plenary talk at Lattice 2011 conference



Thursday, July 14, 2011

The Bielefeld GPU cluster

- hybrid GPU / CPU cluster
- 152 compute nodes in 14x19” racks
  - 48 nodes with 4 GTX 580
  - 104 nodes with 2 Tesla M2075
  - 304 CPUs (1216 cores) with 7296 GB memory
- 7 storage nodes / 2 head nodes
- 1.1 million € founded with federal and state government funds
- dedicated exclusively to Lattice QCD

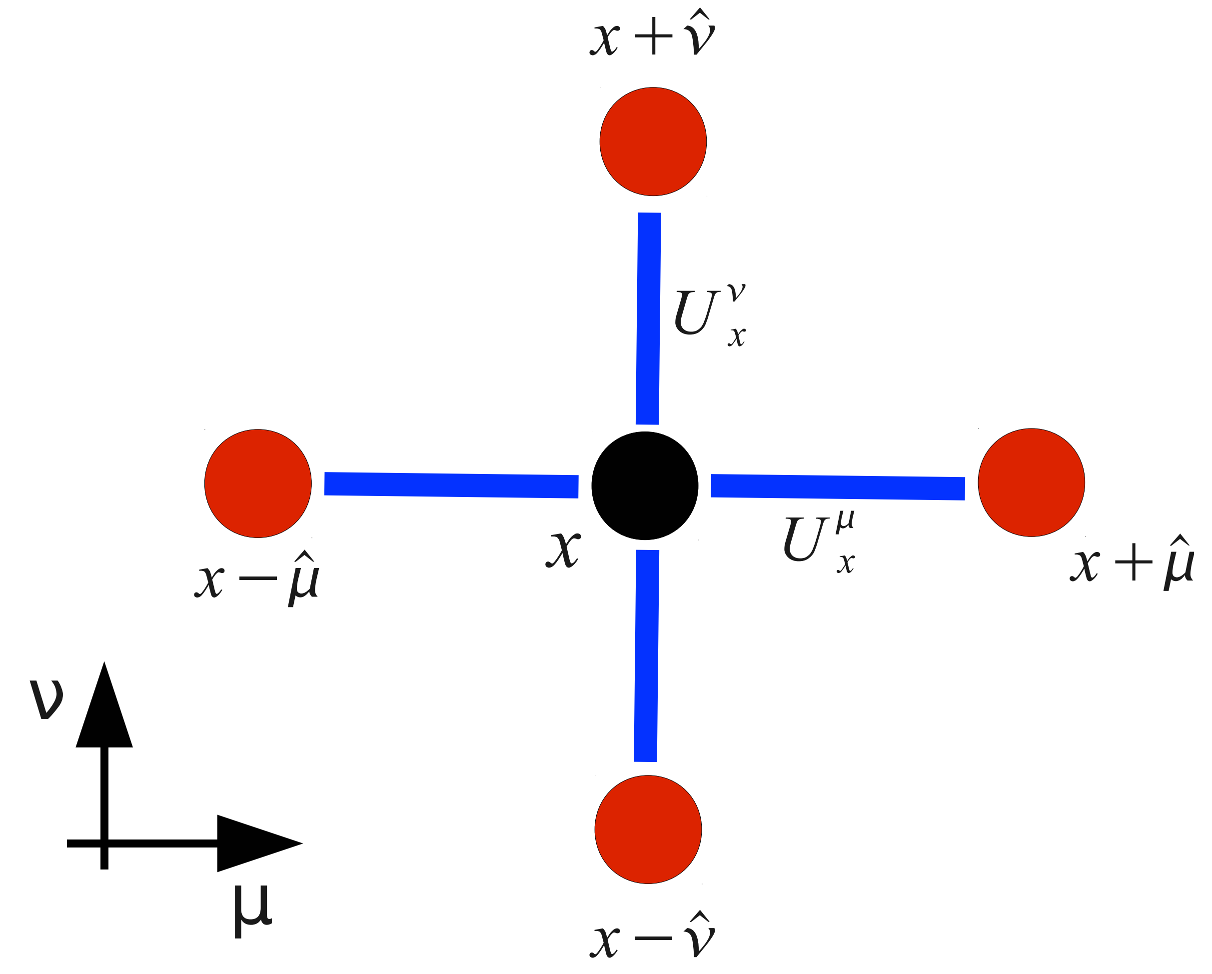


Universität Bielefeld

Standard staggered Fermion Matrix (Dslash)

- Krylov space inversion of fermion matrix dominates runtime
- within inversion application of sparse Matrix dominates (>80%)

$$w_x = D_{x,x'} v_{x'} = \sum_{\mu=0}^{3} \left\{ U_{x,\mu} v_{x+\hat{\mu}} - U^\dagger_{x-\hat{\mu},\mu} v_{x-\hat{\mu}} \right\}$$

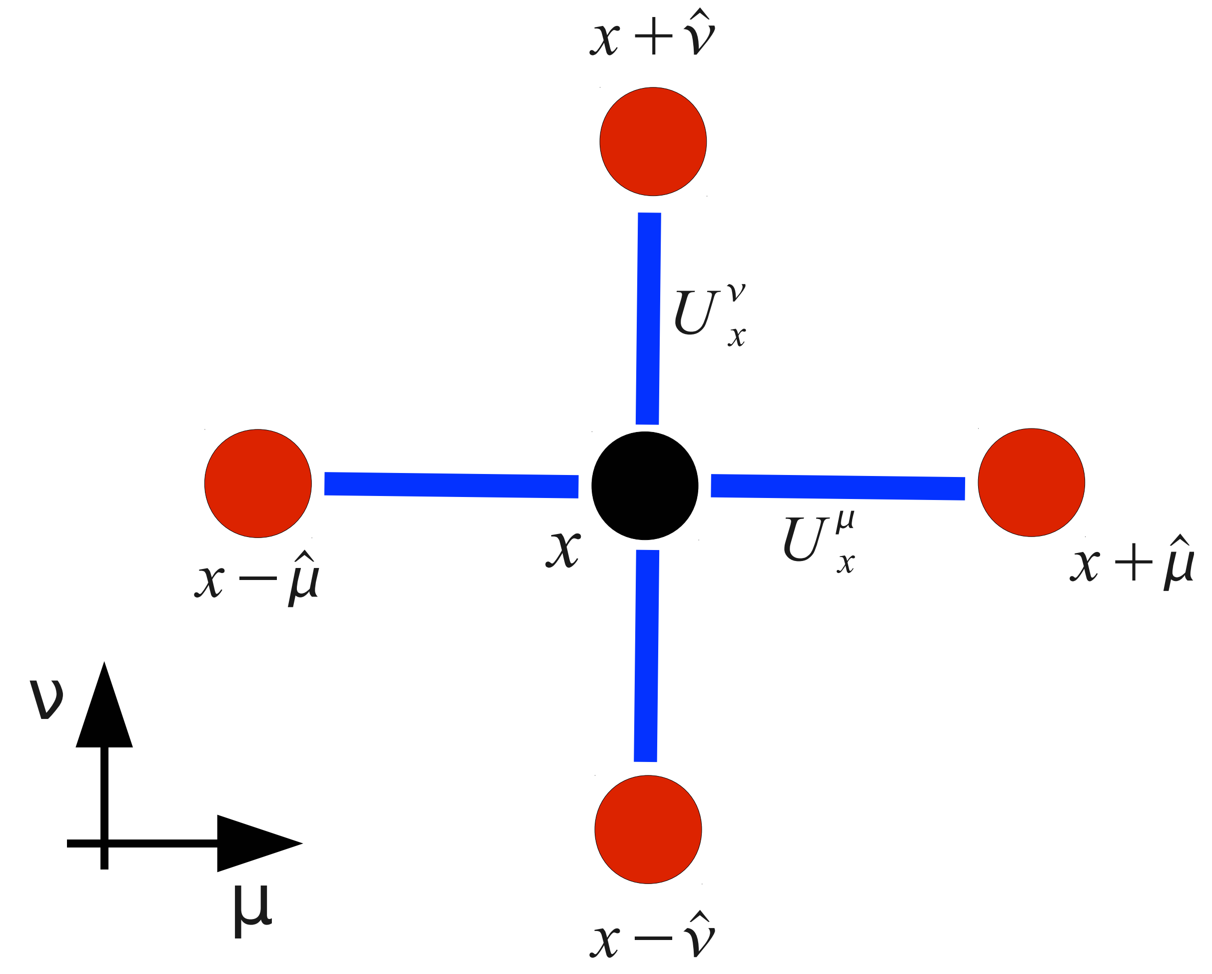


Standard staggered Fermion Matrix (Dslash)

- Krylov space inversion of fermion matrix dominates runtime
- within inversion application of sparse Matrix dominates (>80%)

$$w_x = D_{x,x'} v_{x'} = \sum_{\mu=0}^{3} \left\{ U_{x,\mu} v_{x+\hat{\mu}} - U^{\dagger}_{x-\hat{\mu},\mu} v_{x-\hat{\mu}} \right\}$$

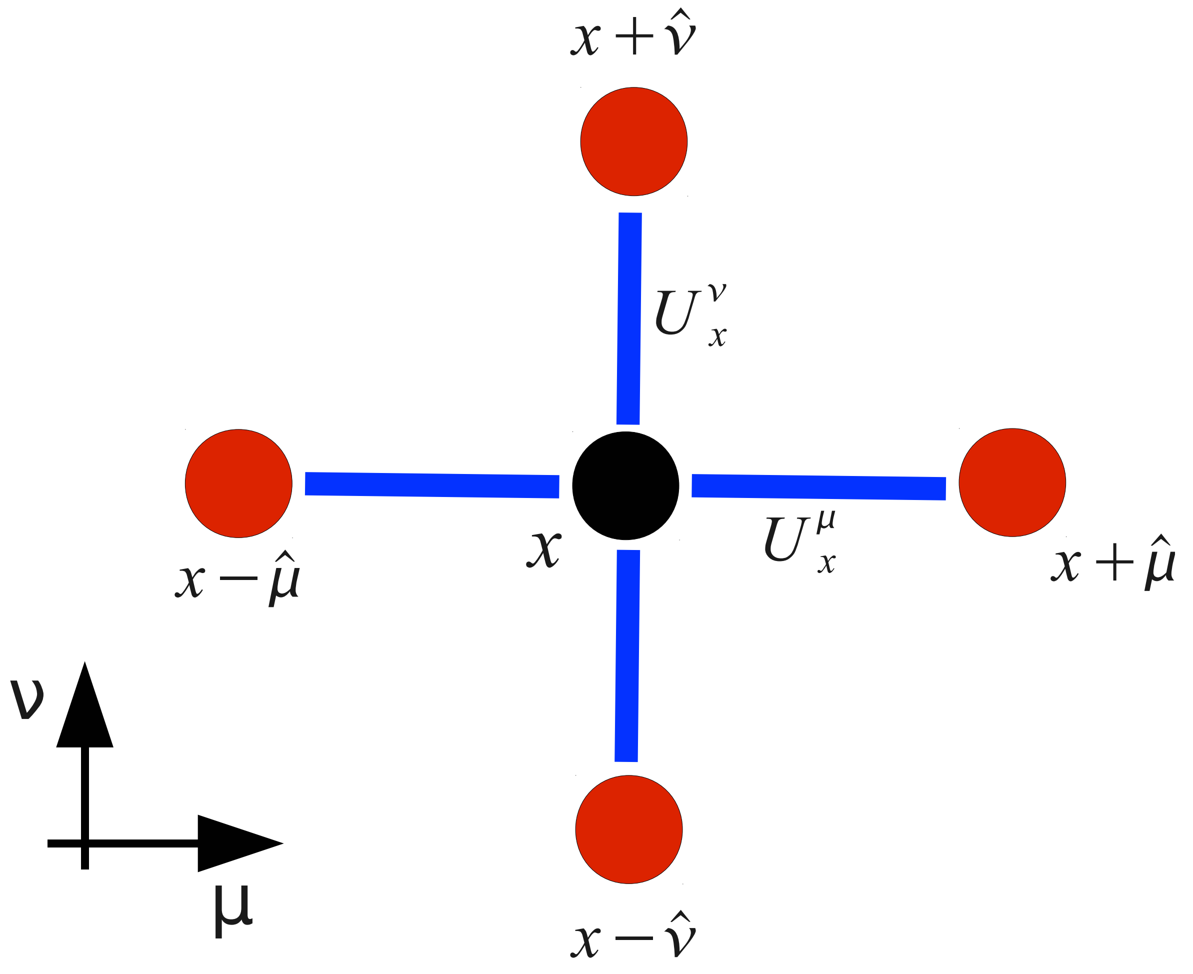
- memory: 8 SU(3) matrices input, 8 color vectors input, 1 color vector output
  - 8 x (72 + 24) + 24 bytes = 792 bytes (1584 for double precision)



Standard staggered Fermion Matrix (Dslash)

- Krylov space inversion of fermion matrix dominates runtime
- within inversion application of sparse Matrix dominates (>80%)

$$w_x = D_{x,x'} v_{x'} = \sum_{\mu=0}^{3} \left\{ U_{x,\mu} v_{x+\hat{\mu}} - U^{\dagger}_{x-\hat{\mu},\mu} v_{x-\hat{\mu}} \right\}$$



- memory: 8 SU(3) matrices input, 8 color vectors input, 1 color vector output
    - 8 x (72 + 24) + 24 bytes = 792 bytes (1584 for double precision)
- Flops: (CM = complex mult, CA = complex add)
    - 4 x (2 x 3 x (3 CM + 2 CA) + 3 CA) + 3 x 3 CA = 570 flops
- flops / byte ratios: 0.72

Universität Bielefeld

Bandwidth bound

- memory bandwidth is crucial
- GTX cards are always faster
  - even for double precision calculations
- linear algebra has an even worse flop / byte ratio
  - vector addition c = a + b
    - 48 bytes in, 24 bytes out, 6 flops →0.08 flops/byte
- flops are free - but registers are limited
- Dslash efficiency Tesla M2075: 0.72 flop/byte * 144 Gbytes/s = 103 Gflops (10% peak)

Bandwidth bound

Card	GFlops (32 bit)	GFlops (32 bit)	GBytes/s	Flops / byte	Flops/ byte
GTX 580	1581	198	192	8.2	1.03
Tesla M2075	1030	515	144	7.2	3.6

- memory bandwidth is crucial
- GTX cards are always faster
  - even for double precision calculations
- linear algebra has an even worse flop / byte ratio
  - vector addition c = a + b
    - 48 bytes in, 24 bytes out, 6 flops →0.08 flops/byte
- flops are free - but registers are limited
- Dslash efficiency Tesla M2075: 0.72 flop/byte * 144 Gbytes/s = 103 Gflops (10% peak)

Optimizing memory access

- use coalesced memory layout: structure of arrays (SoA) instead of AoS
- one can reconstruct a SU(3) matrix also from 8 or 12 floats
  - improved actions result in matrices that are no longer SU(3):
    must load 18 floats

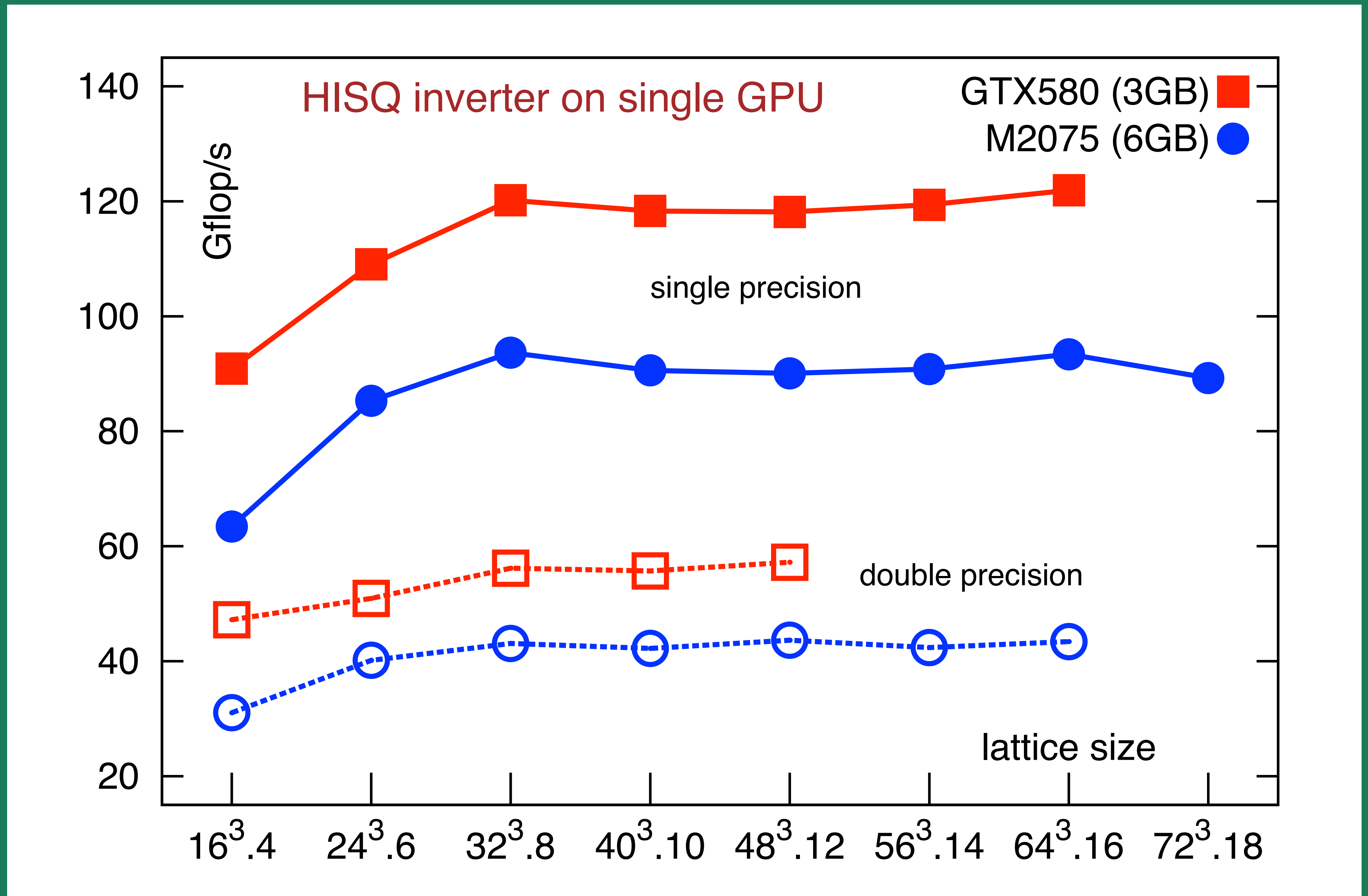
Optimizing memory access

- use coalesced memory layout: structure of arrays (SoA) instead of AoS
- one can reconstruct a SU(3) matrix also from 8 or 12 floats
    - improved actions result in matrices that are no longer SU(3):
      must load 18 floats
- exploit texture access: near 100% bandwidth
- ECC hurts (naive 12.5%, real world ~ 20-30 %)
- do more work with less bytes:
  → mixed precision inverters (QUDA libray, Clark et al, CPC.181:1517,2010)
  → multiple right hand sides

Universität Bielefeld

Optimizing memory access

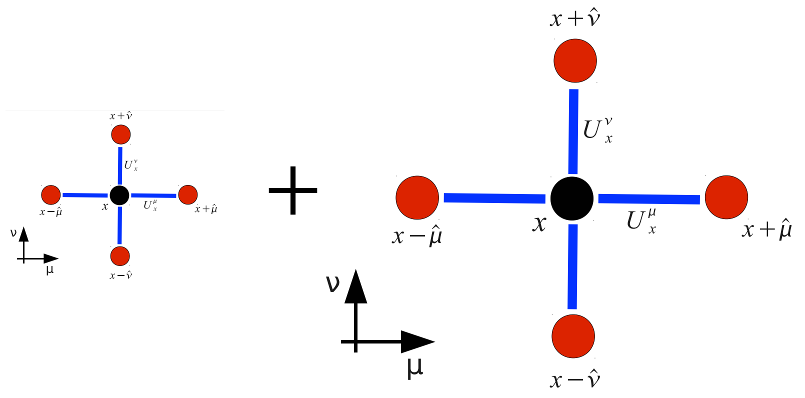
- use coalesced memory layout: structure of arrays (SoA)
- one can reconstruct a SU(3) matrix also from 8 or 12 flo
  - improved actions result in matrices that are no longer
    must load 18 floats
- exploit texture access: near 100% bandwidth
- ECC hurts (naive 12.5%, real world ~ 20-30 %)
- do more work with less bytes:
  → mixed precision inverters (QUDA libray, Clark et al, CPC.181:1517
  → multiple right hand sides



Universität Bielefeld

Solvers for multiple right hand sides

- consider single precision for improved (HISQ) action



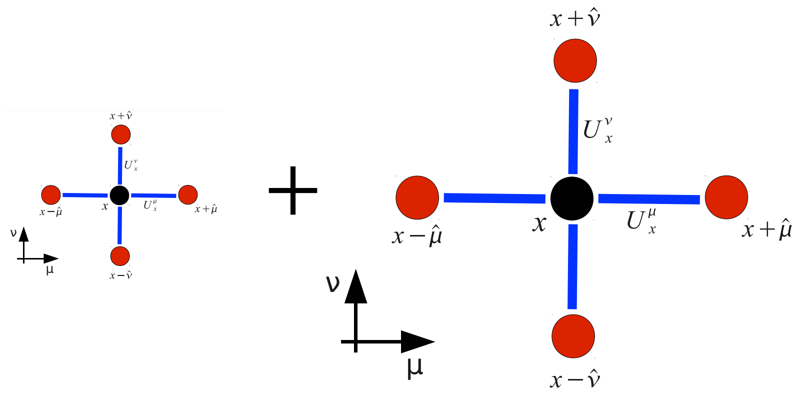
- need inversions for many (1500) ‘source’-vectors for a fixed gauge field (matrix)

- Bytes for n vectors $16 \cdot (72 + n \cdot 24)\,\text{bytes} + n \cdot 24\,\text{bytes} = 1152\,\text{bytes} + 408\,\text{bytes} \cdot n\ .$

- Flops for n vectors $1146\,\text{flops} \cdot n$

# r.h.s.	1	2	3	4	5
flops/ byte	0.73	1.16	1.45	1.65	1.8

Solvers for multiple right hand sides



- consider single precision for improved (HISQ) action
- need inversions for many (1500) 'source'-vectors for a fixed gauge field (matrix)
- Bytes for n vectors $16 \cdot (72 + n \cdot 24)\,\text{bytes} + n \cdot 24\,\text{bytes} = 1152\,\text{bytes} + 408\,\text{bytes} \cdot n$.
- Flops for n vectors $1146\,\text{flops} \cdot n$

# r.h.s.	1	2	3	4	5
flops/ byte	0.73	1.16	1.45	1.65	1.8

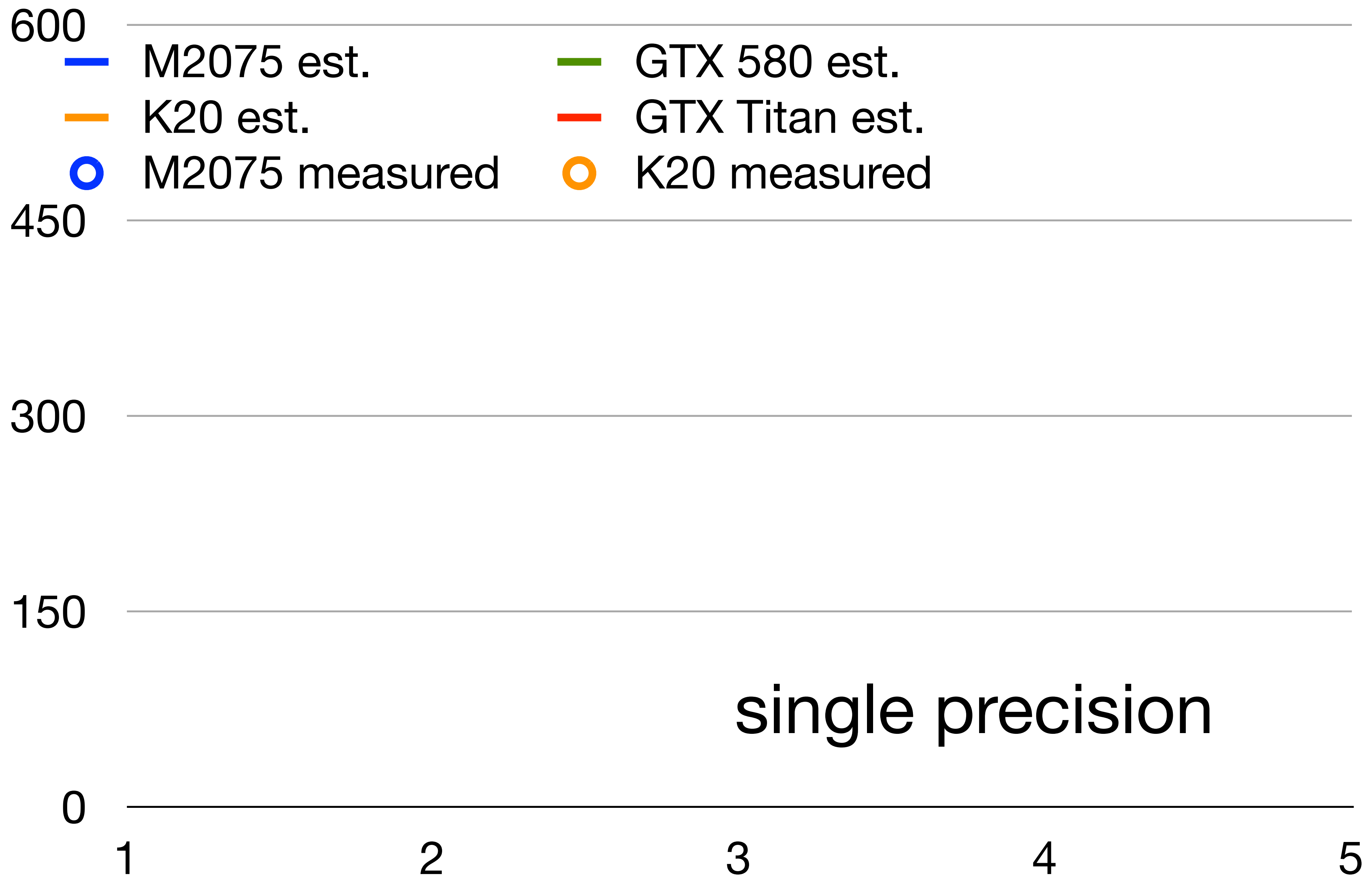
- Issue: register usage and spilling
  - spilling for more than 3 r.h.s. with Fermi architecture
  - already for more than 1 r.h.s. in double precision

#	registers	stack frame	spill stores	spill loads	SM 3.5 reg
1	38	0	0	0	40
2	58	0	0	0	60
3	63	0	0	0	65
4	63	40	76	88	72
5	63	72	212	216	77

Dslash-performance

- estimate performance from flop/byte ratio and available memory bandwidth
- full inversion should be roughly 10-15% lower

card	M2075	GTX 580	K20	GTX Titan
Bandwidth [GB/s]	150	192	208	288

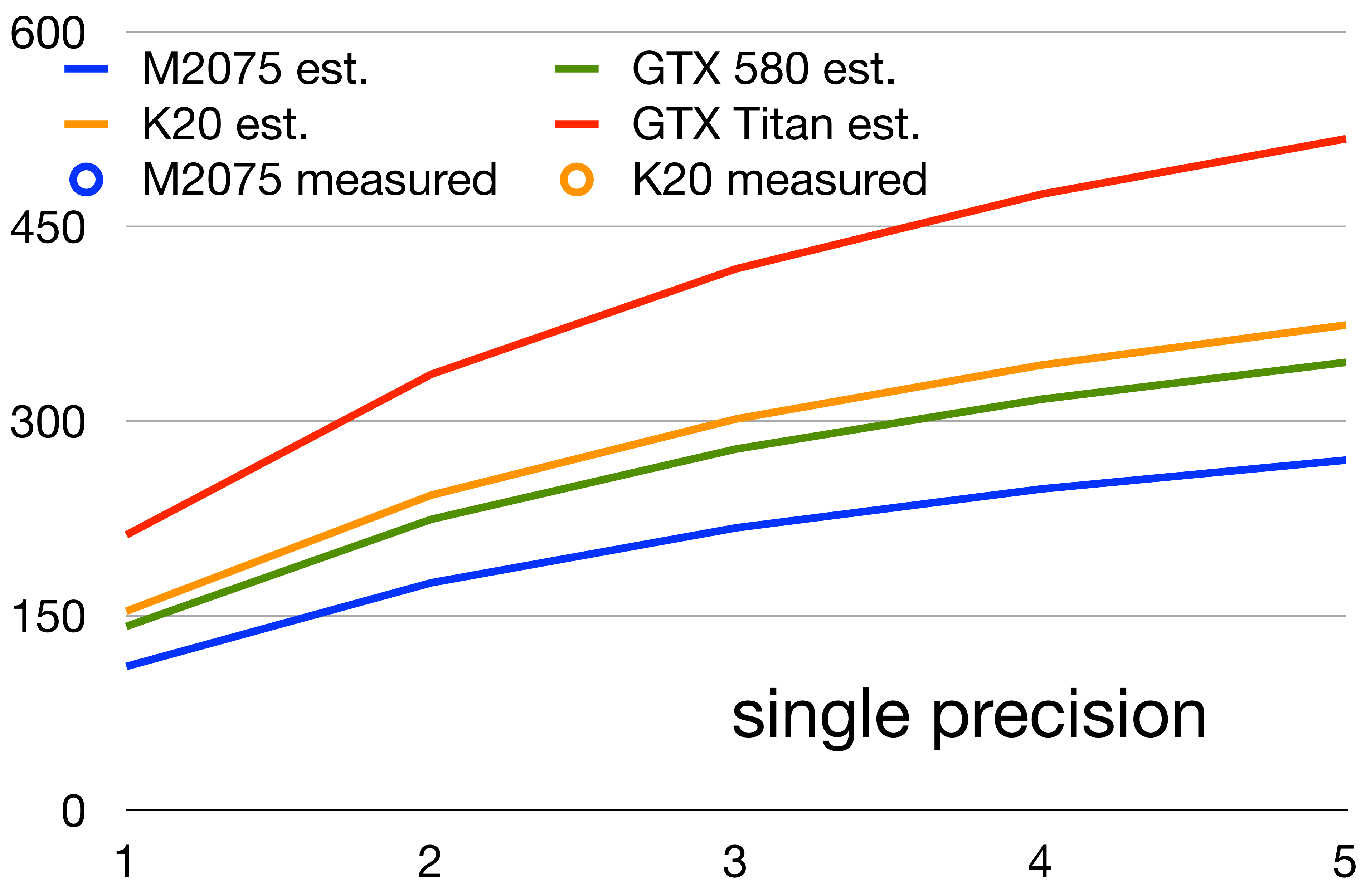


Universität Bielefeld

Dslash-performance

- estimate performance from flop/byte ratio and available memory bandwidth
- full inversion should be roughly 10-15% lower

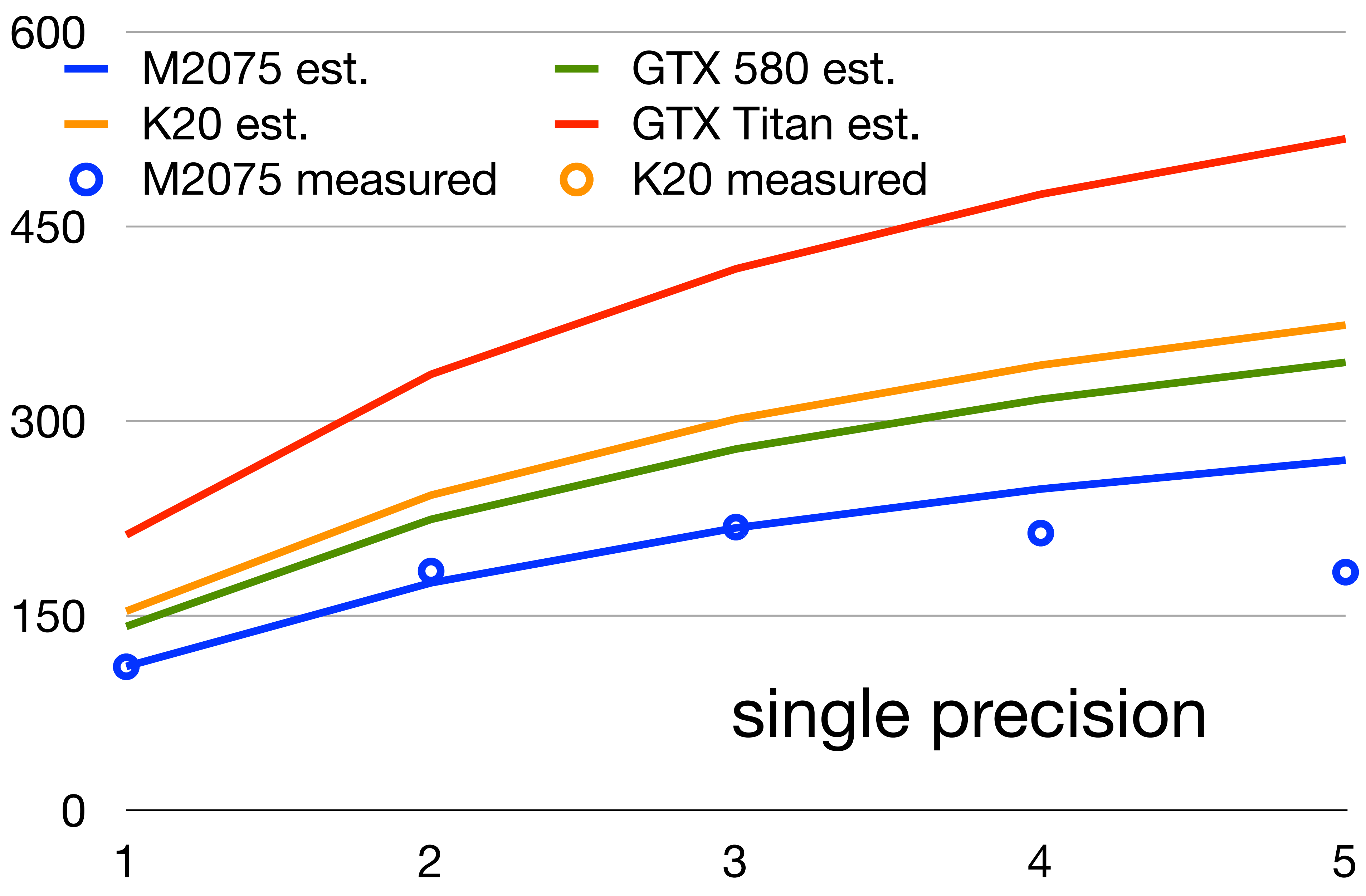
card	M2075	GTX 580	K20	GTX Titan
Bandwidth [GB/s]	150	192	208	288



Dslash-performance

- estimate performance from flop/byte ratio and available memory bandwidth
- full inversion should be roughly 10-15% lower

card	M2075	GTX 580	K20	GTX Titan
Bandwidth [GB/s]	150	192	208	288

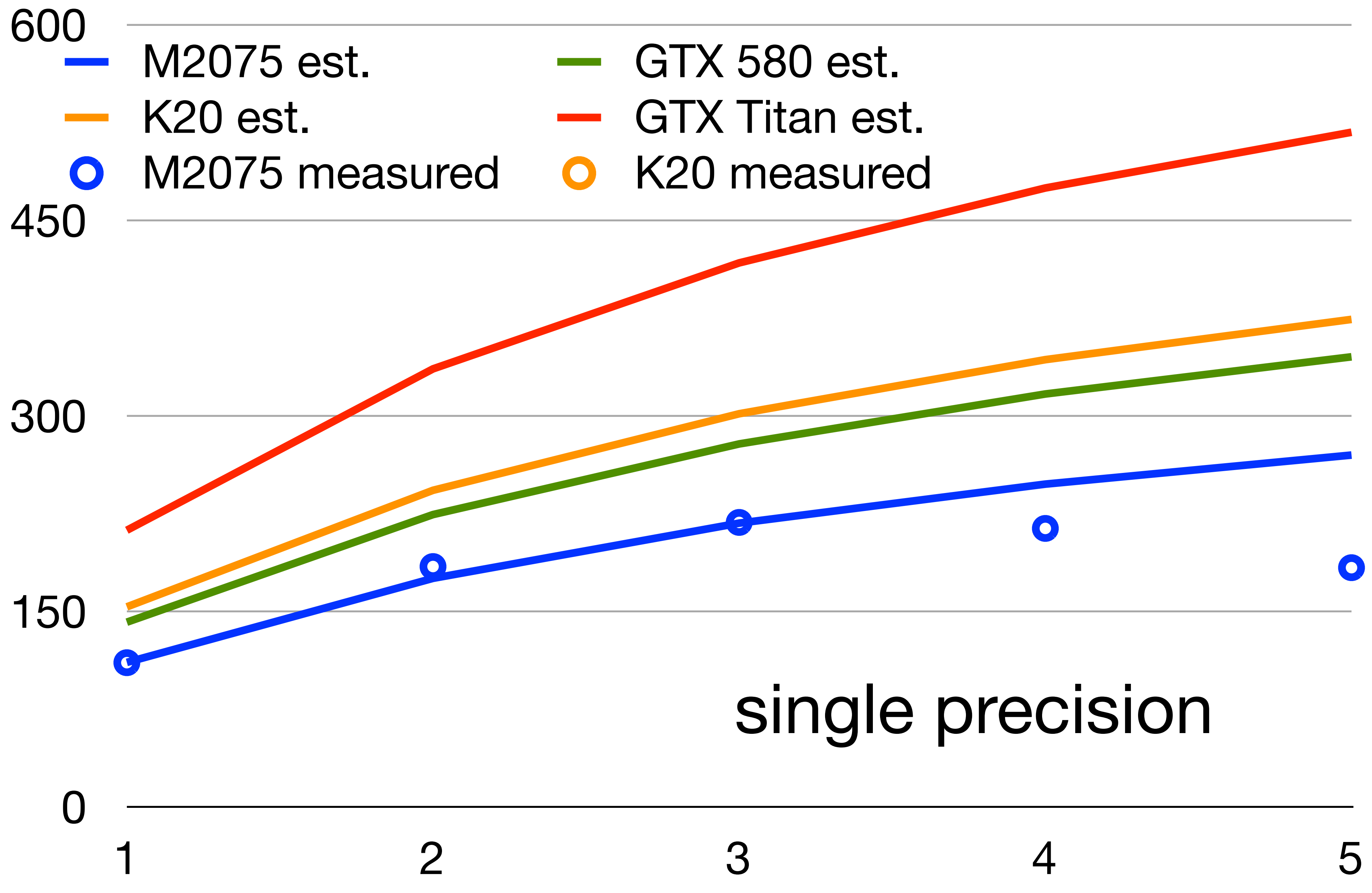


Universität Bielefeld

Dslash-performance

- estimate performance from flop/byte ratio and available memory bandwidth
- full inversion should be roughly 10-15% lower

card	M2075	GTX 580	K20	GTX Titan
Bandwidth [GB/s]	150	192	208	288

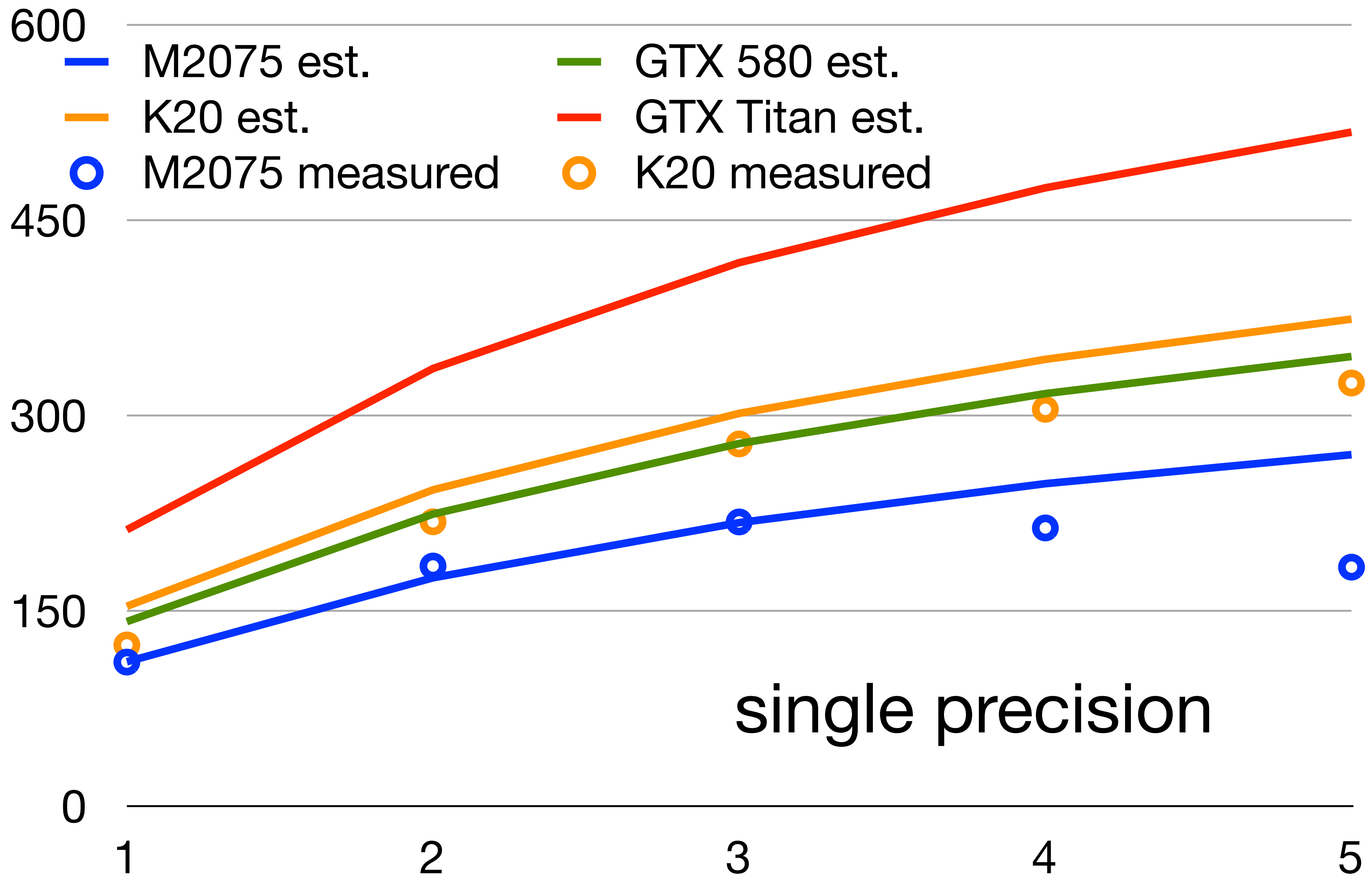


#	registers	stack frame	spill stores	spill loads	SM 3.5 reg
1	38	0	0	0	40
2	58	0	0	0	60
3	63	0	0	0	65
4	63	40	76	88	72
5	63	72	212	216	77

Dslash-performance

- estimate performance from flop/byte ratio and available memory bandwidth
- full inversion should be roughly 10-15% lower

card	M2075	GTX 580	K20	GTX Titan
Bandwidth [GB/s]	150	192	208	288

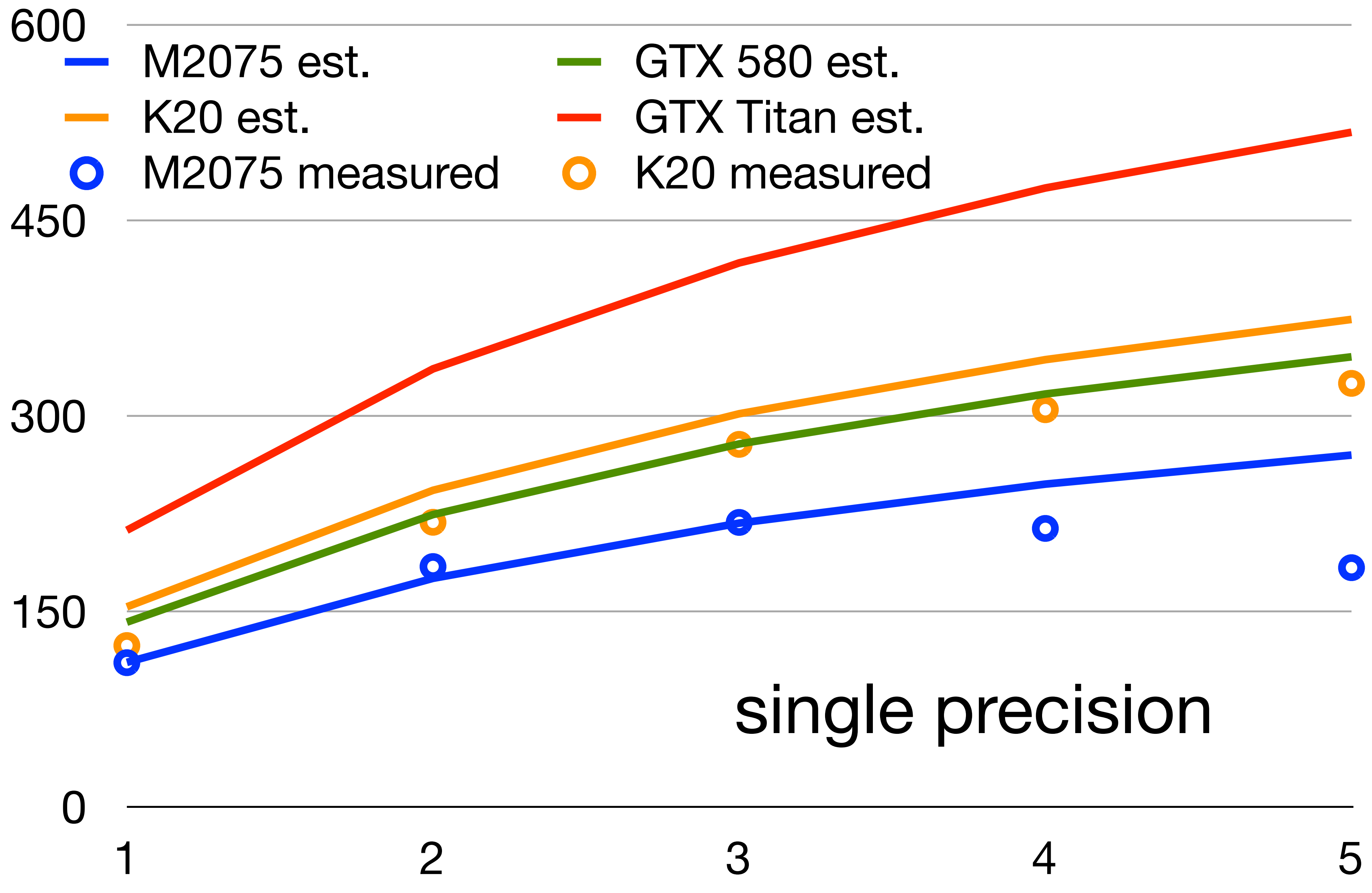


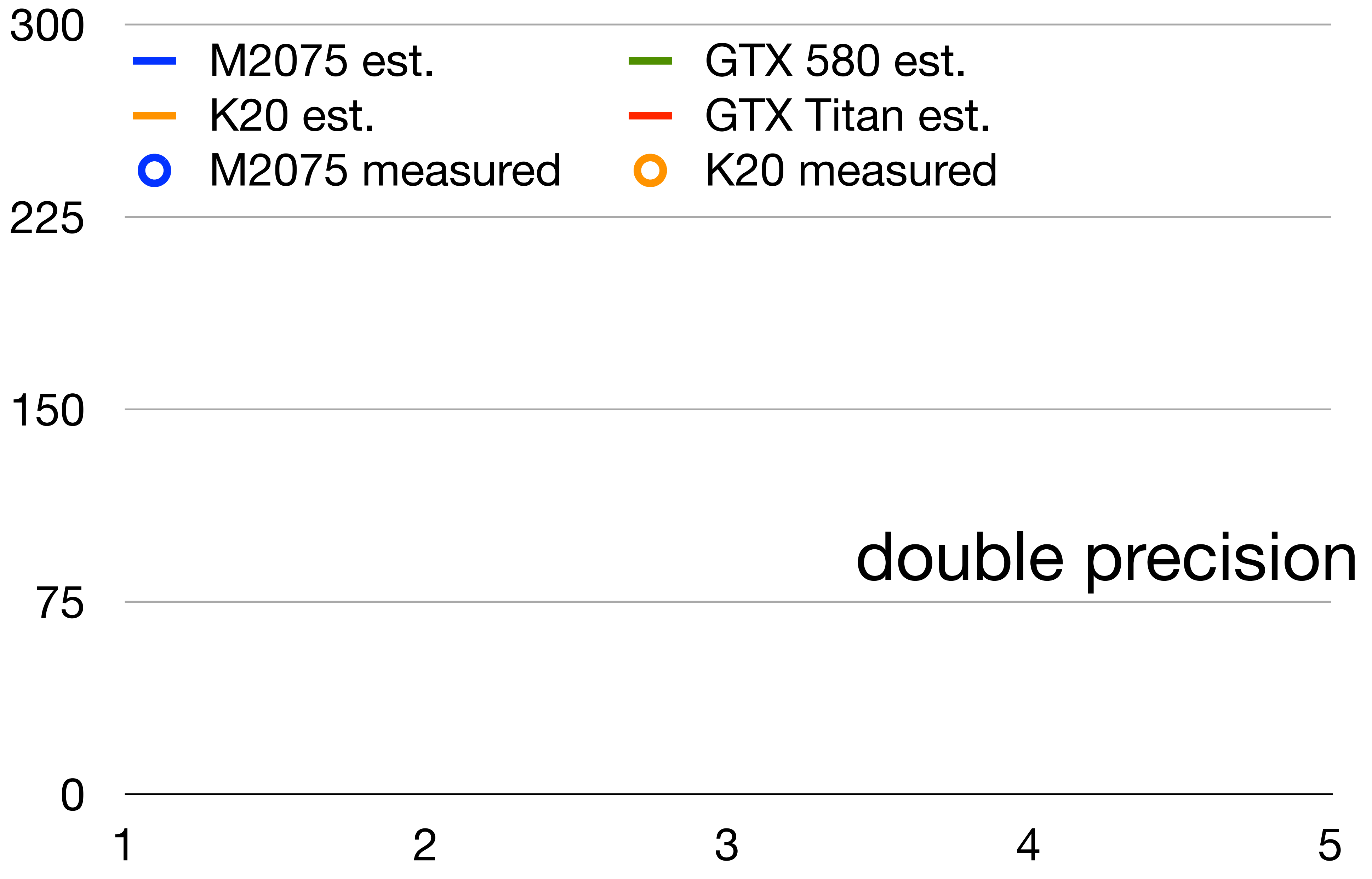
#	registers	stack frame	spill stores	spill loads	SM 3.5 reg
1	38	0	0	0	40
2	58	0	0	0	60
3	63	0	0	0	65
4	63	40	76	88	72
5	63	72	212	216	77

Dslash-performance

- estimate performance from flop/byte ratio and available memory bandwidth
- full inversion should be roughly 10-15% lower

card	M2075	GTX 580	K20	GTX Titan
Bandwidth [GB/s]	150	192	208	288



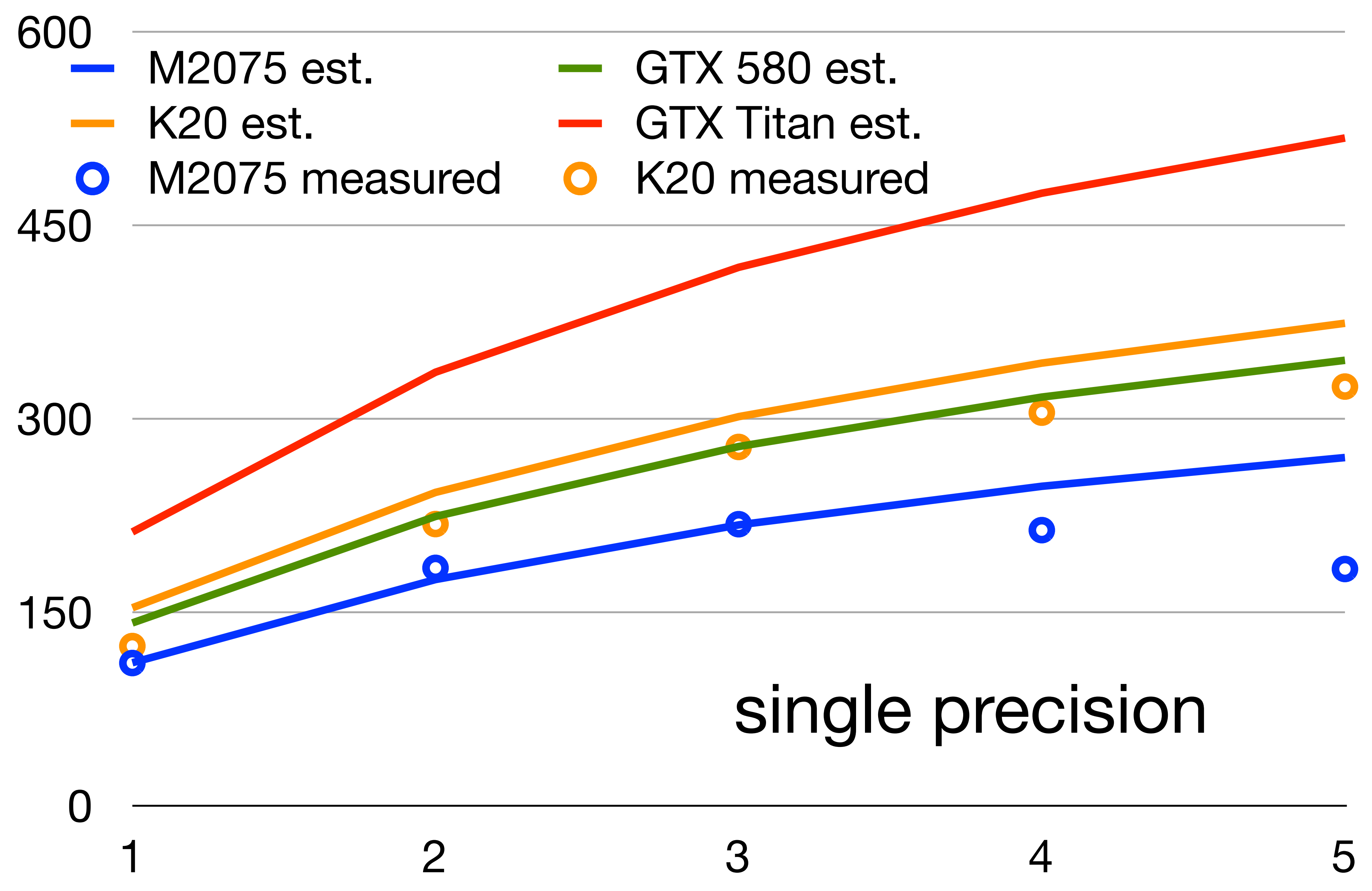


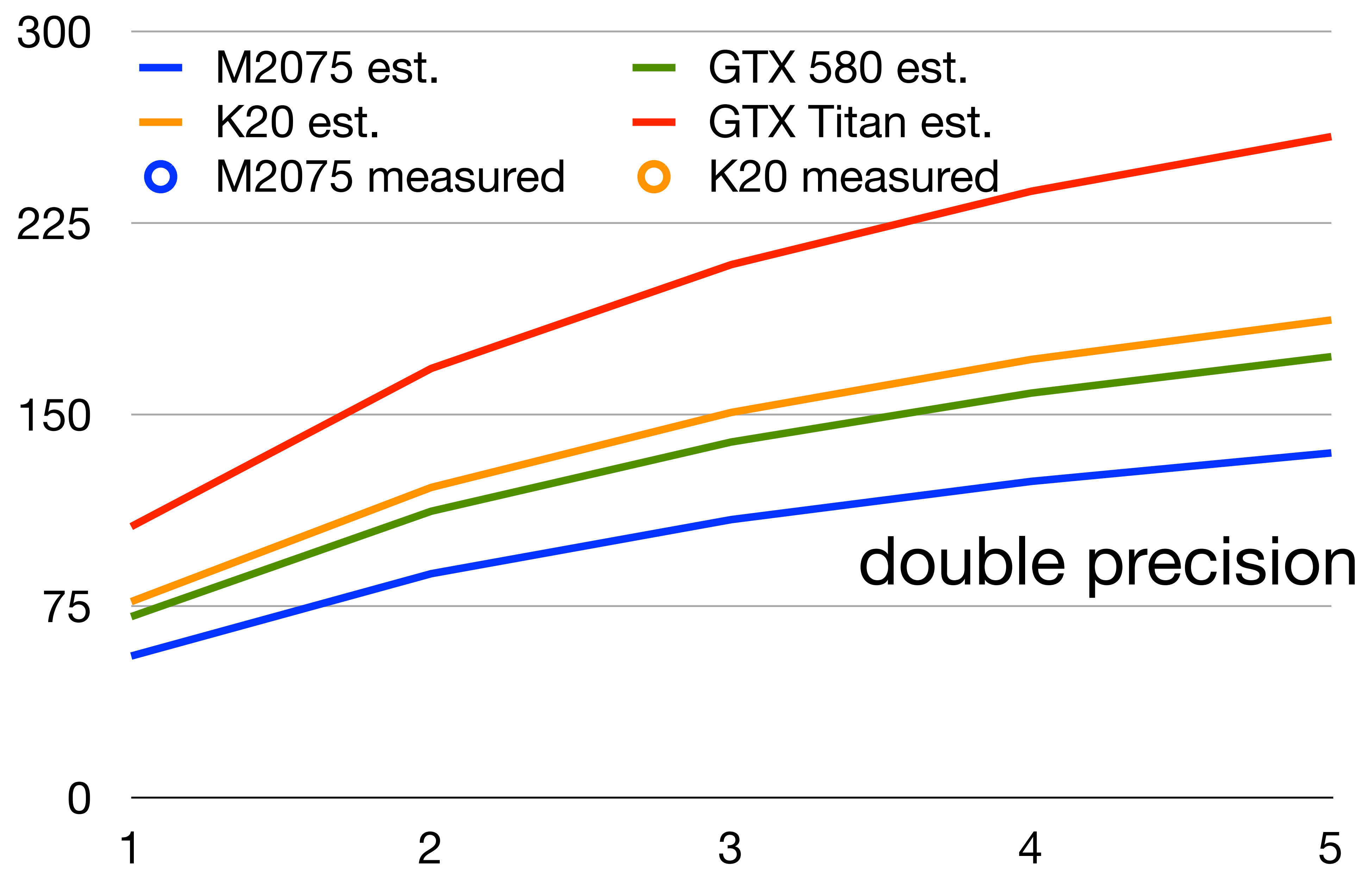
Universität Bielefeld

Dslash-performance

- estimate performance from flop/byte ratio and available memory bandwidth
- full inversion should be roughly 10-15% lower

card	M2075	GTX 580	K20	GTX Titan
Bandwidth [GB/s]	150	192	208	288



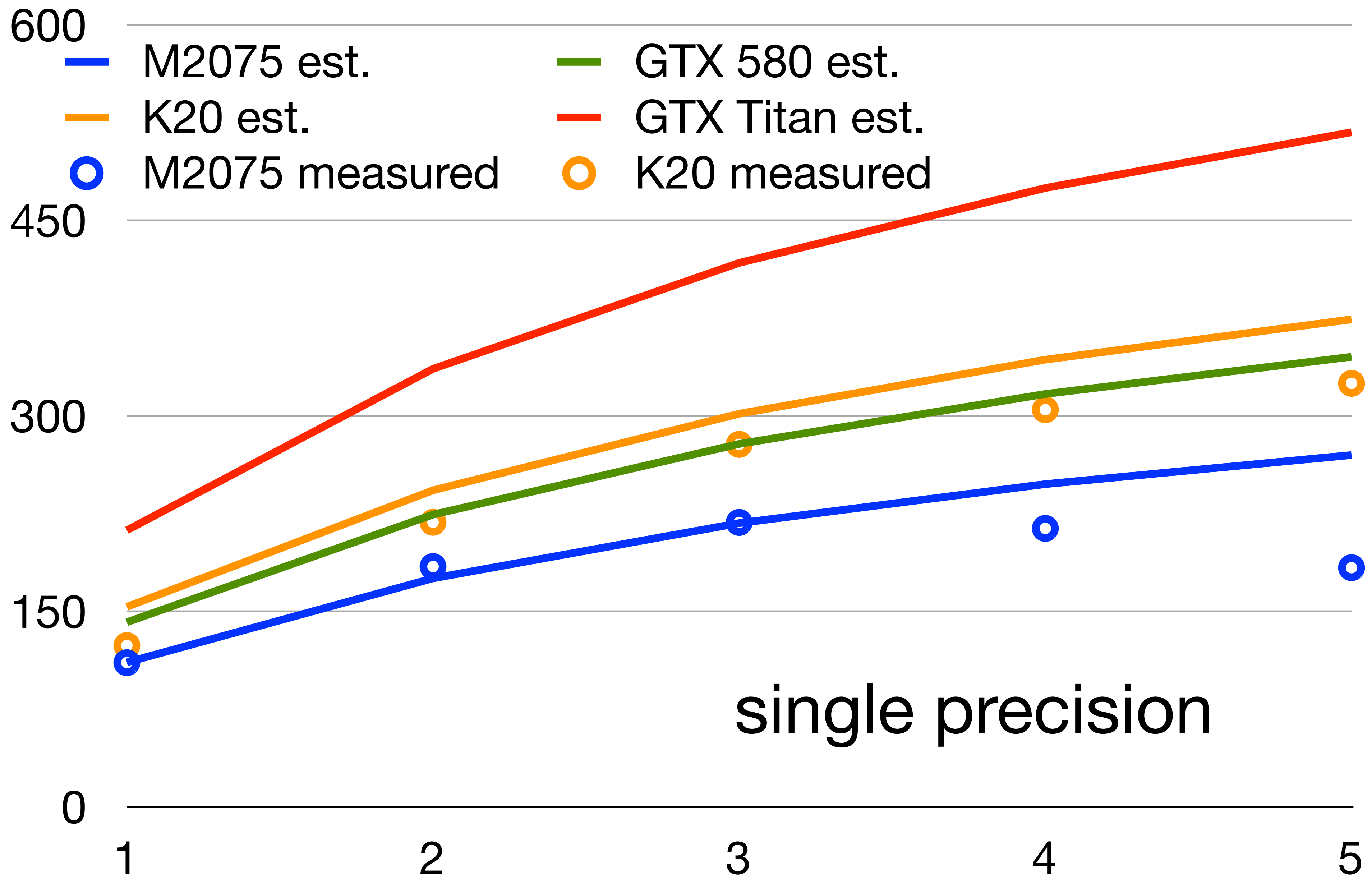


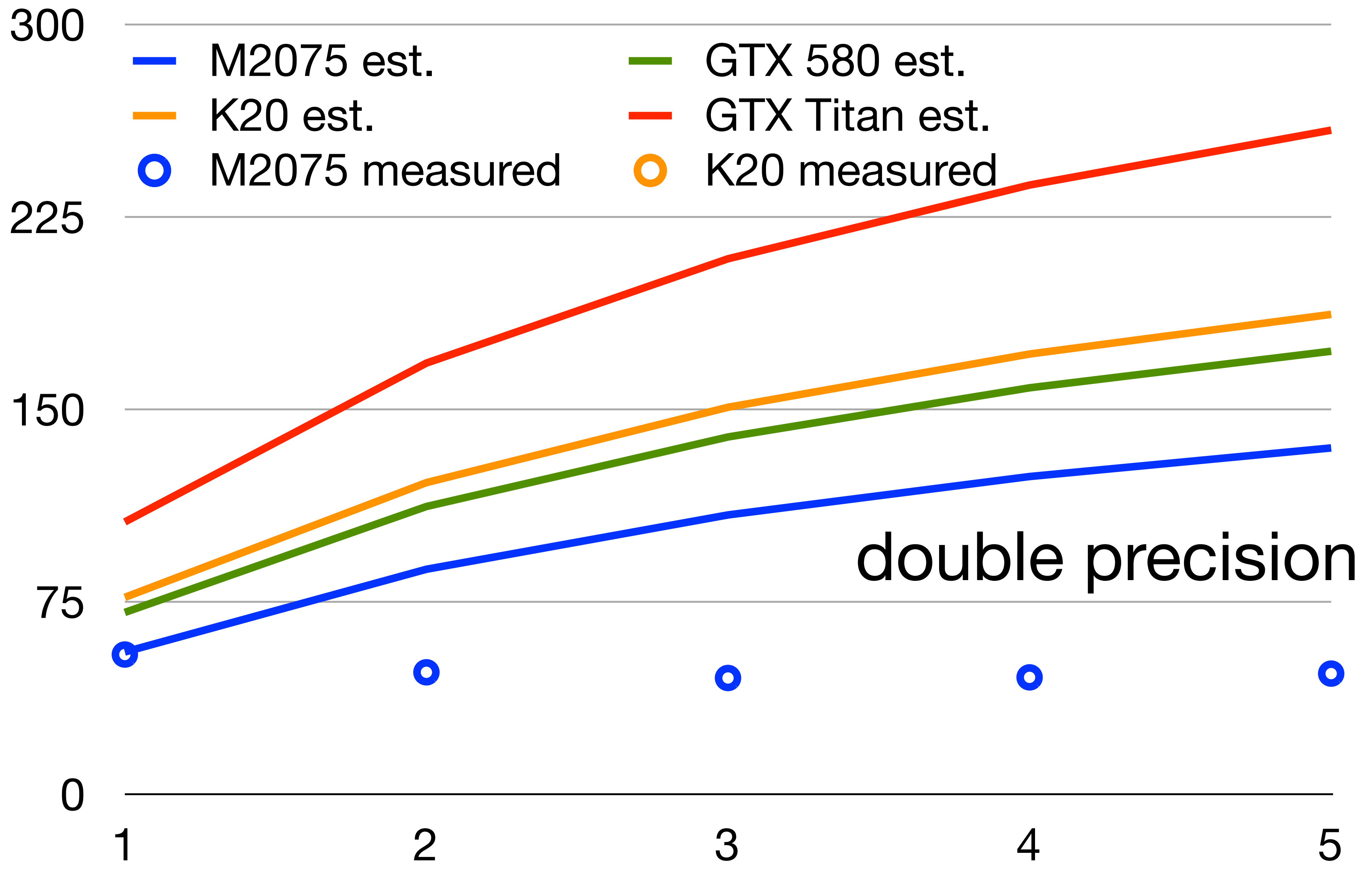
Universität Bielefeld

Dslash-performance

- estimate performance from flop/byte ratio and available memory bandwidth
- full inversion should be roughly 10-15% lower

card	M2075	GTX 580	K20	GTX Titan
Bandwidth [GB/s]	150	192	208	288



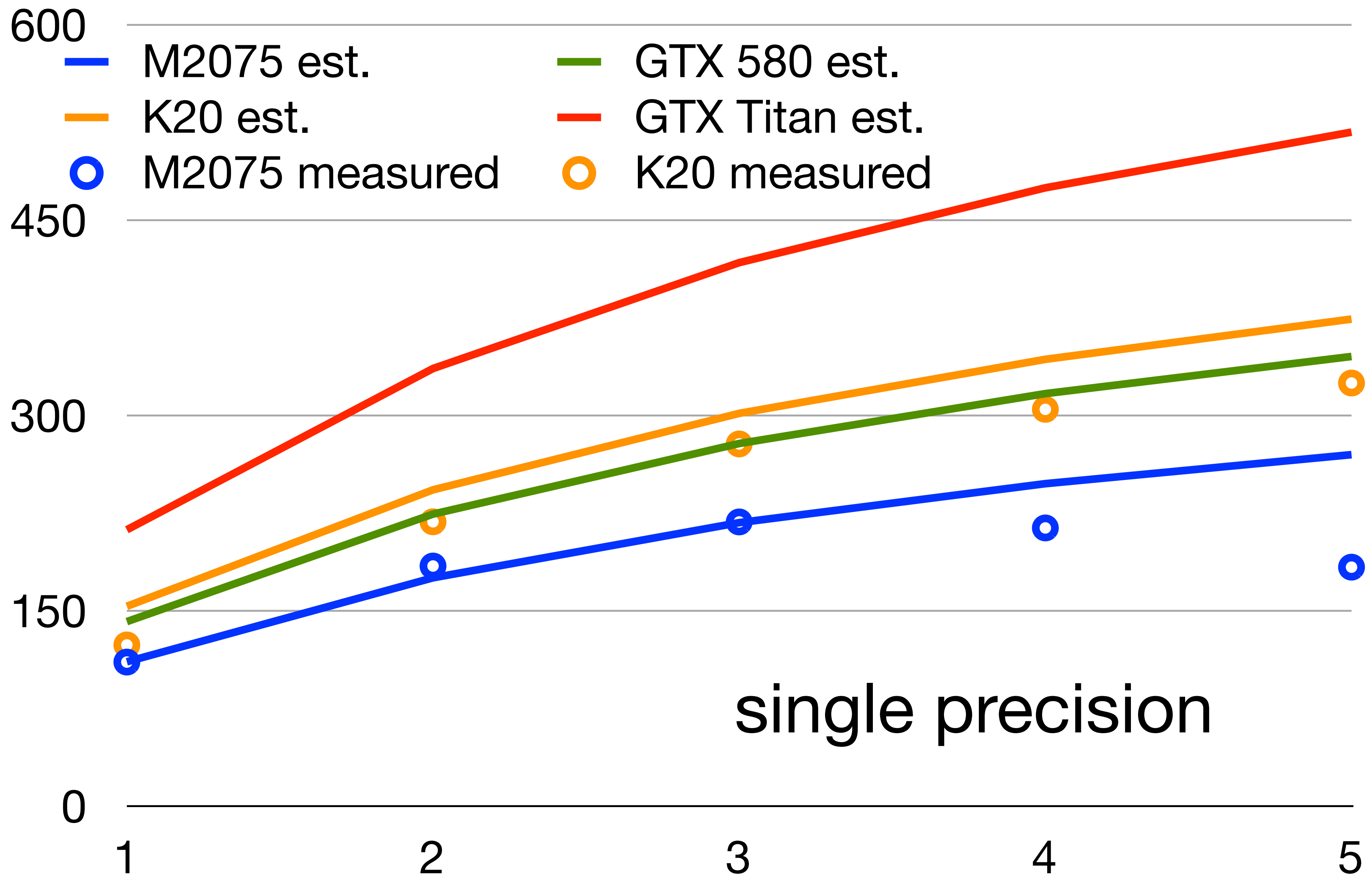


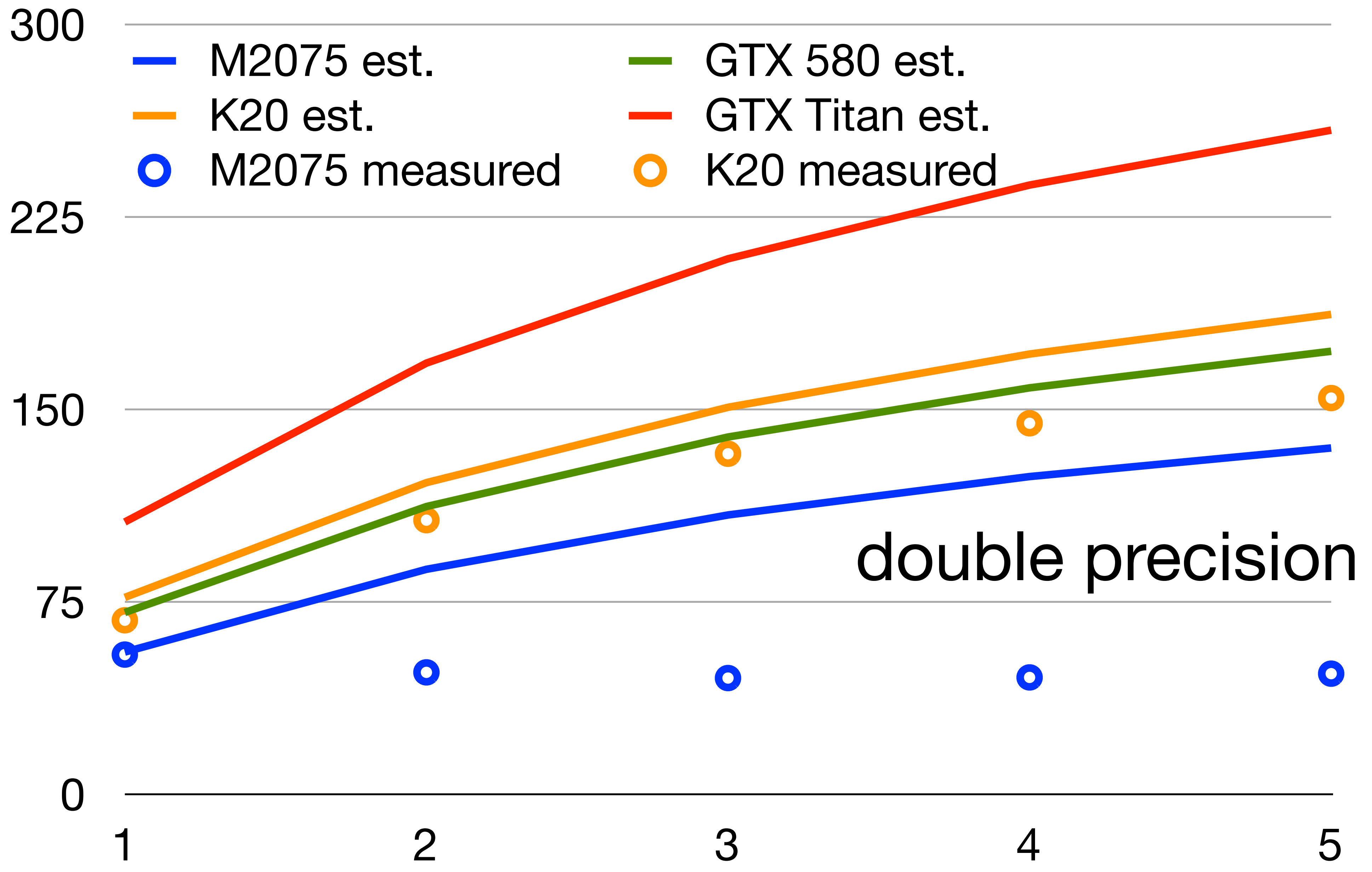
Universität Bielefeld

Dslash-performance

- estimate performance from flop/byte ratio and available memory bandwidth
- full inversion should be roughly 10-15% lower

card	M2075	GTX 580	K20	GTX Titan
Bandwidth [GB/s]	150	192	208	288





Universität Bielefeld

Linear algebra becomes relevant

- matrix operation (Dslash) for multiple r.h.s.
- linear algebra operations cannot
  - float * vector + vector
  - norms
- linear algebra scales linear #r.h.s.

Initialization

loop

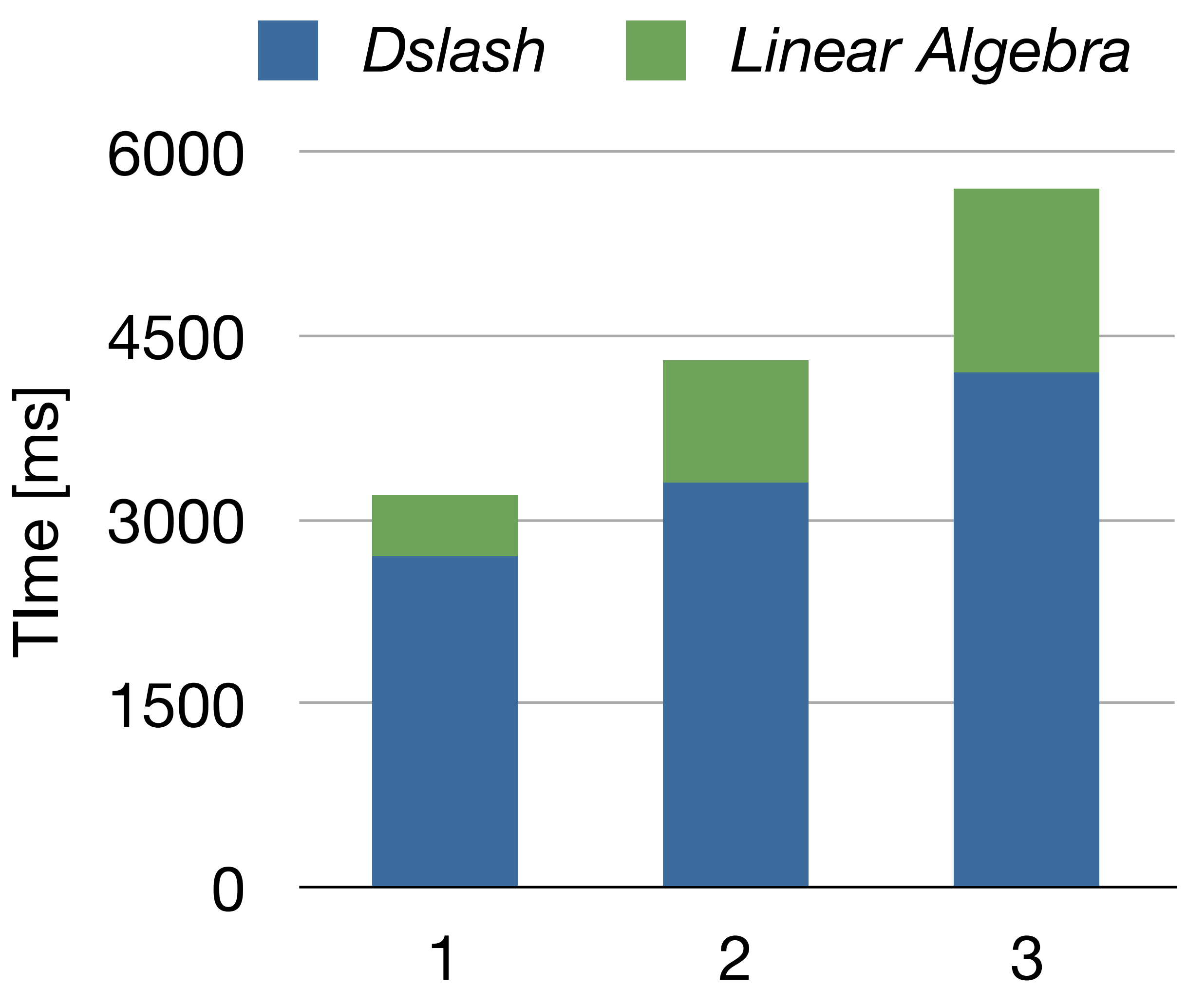
Fermion Matrix

$$\alpha = \sum_i |\vec{p} A \vec{p}|_i$$

$$\vec{r} = \vec{r} - \omega \vec{p}$$
$$\vec{x} = \vec{x} + \omega \vec{p}$$
$$\lambda_i = |r_i|$$
$$\lambda = \sum_i \lambda_i$$

$$\vec{p} = \vec{r} + \gamma \vec{p}$$

Linear algebra becomes relevant



- matrix operation (Dslash) for multiple r.h.s.
- linear algebra operations cannot
  - float * vector + vector
  - norms
- linear algebra scales linear #r.h.s.

Initialization

loop

Fermion Matrix

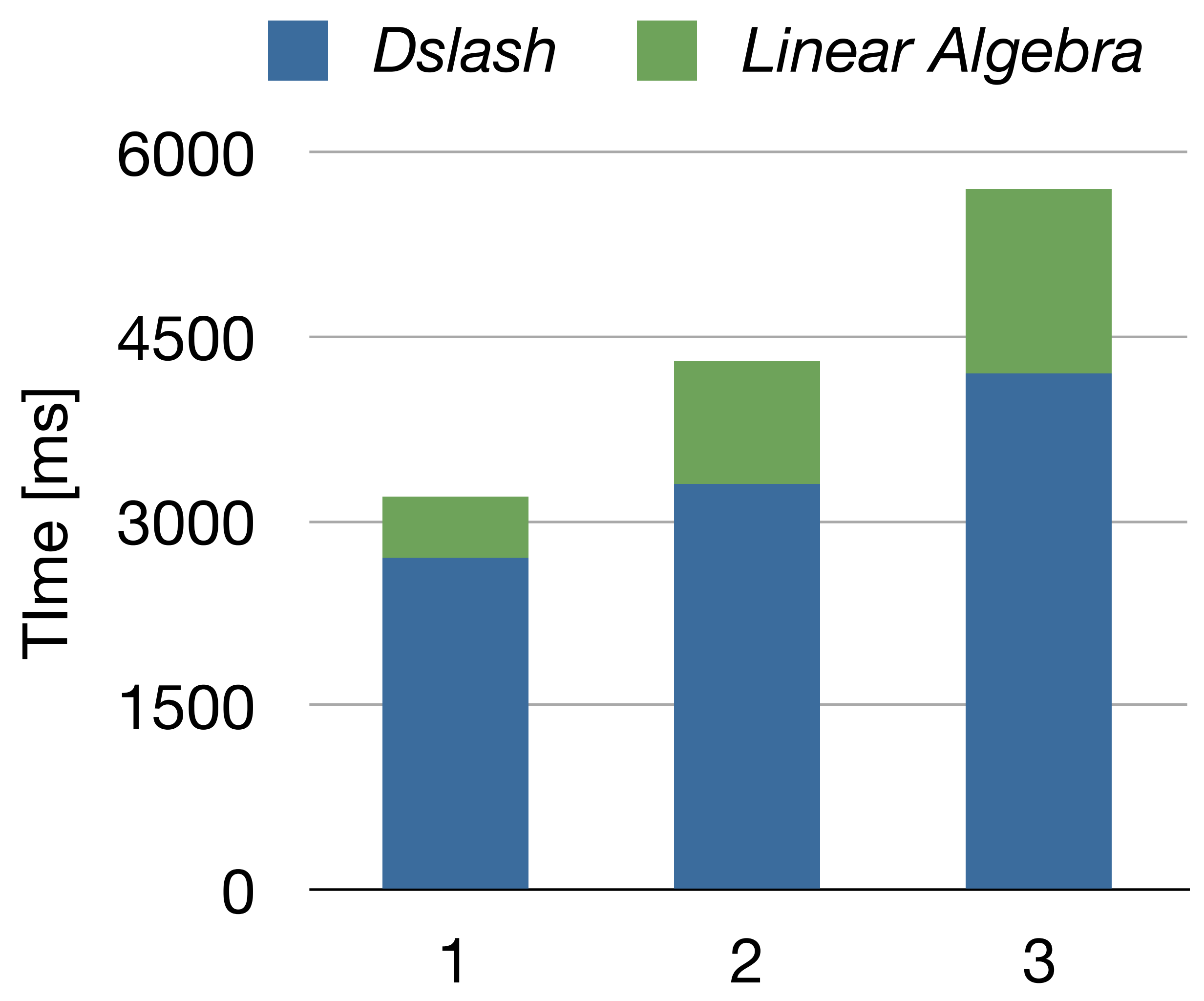
$$\alpha = \sum_i |\vec{p}A\vec{p}|_i$$

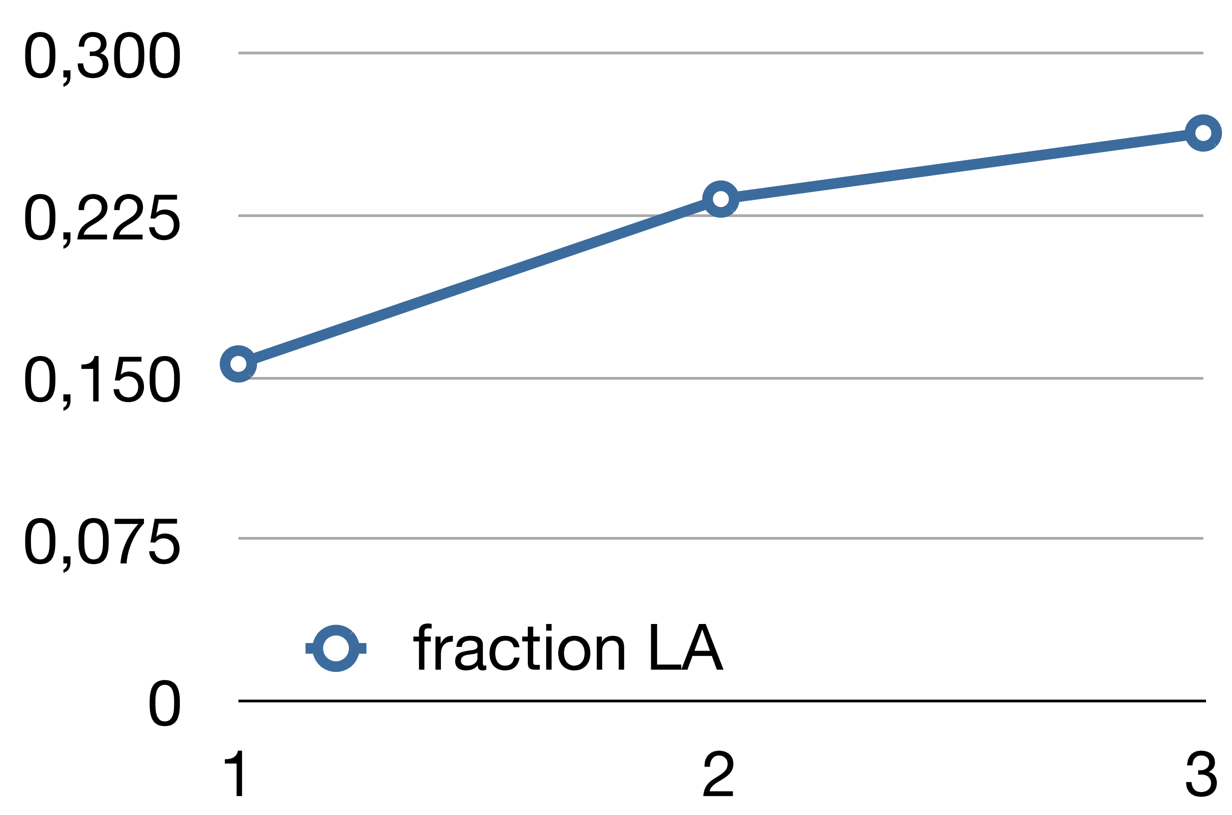
$$\vec{r} = \vec{r} - \omega\vec{p}$$
$$\vec{x} = \vec{x} + \omega\vec{p}$$
$$\lambda_i = |r_i|$$
$$\lambda = \sum_i \lambda_i$$

$$\vec{p} = \vec{r} + \gamma\vec{p}$$

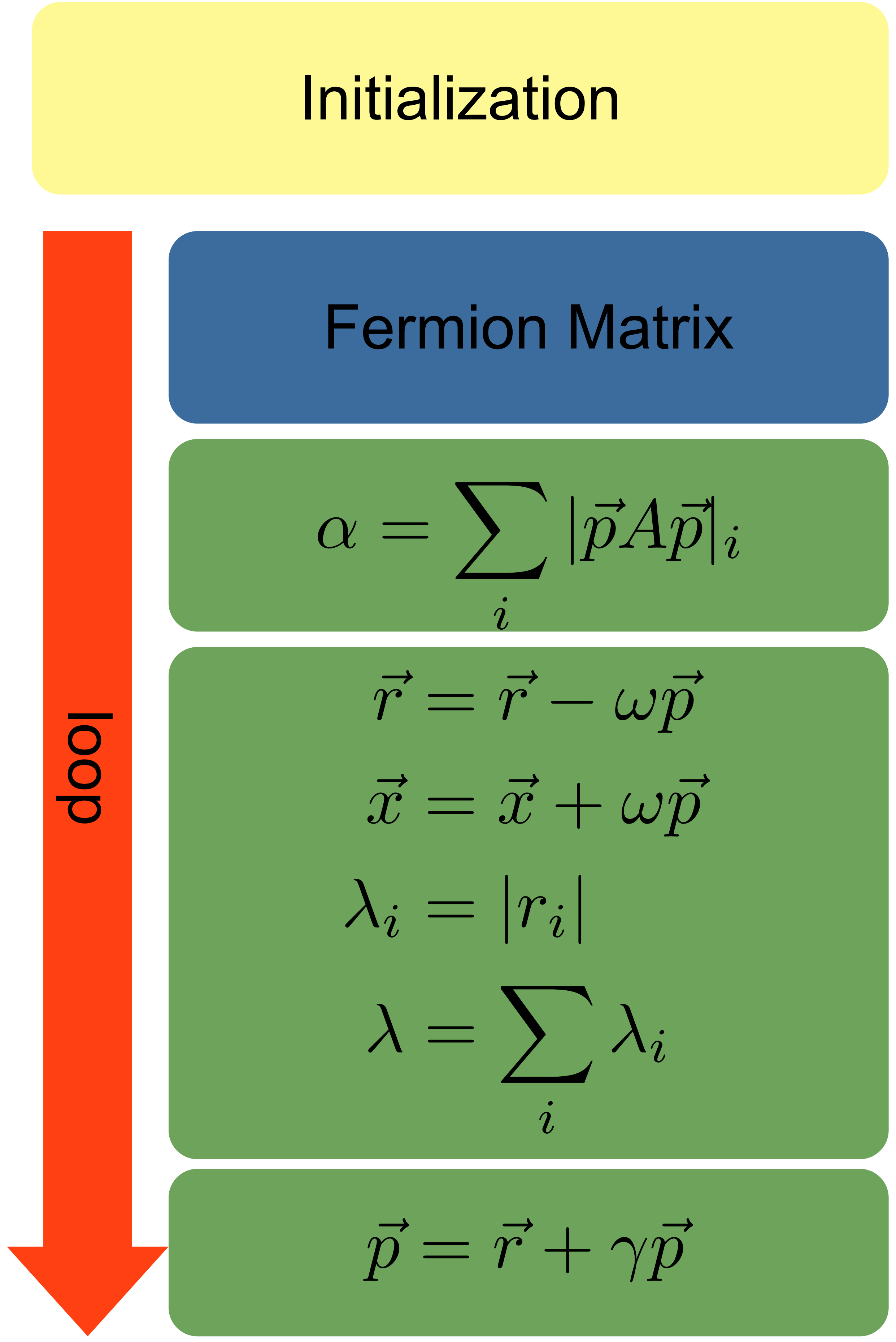
Universität Bielefeld

Linear algebra becomes relevant



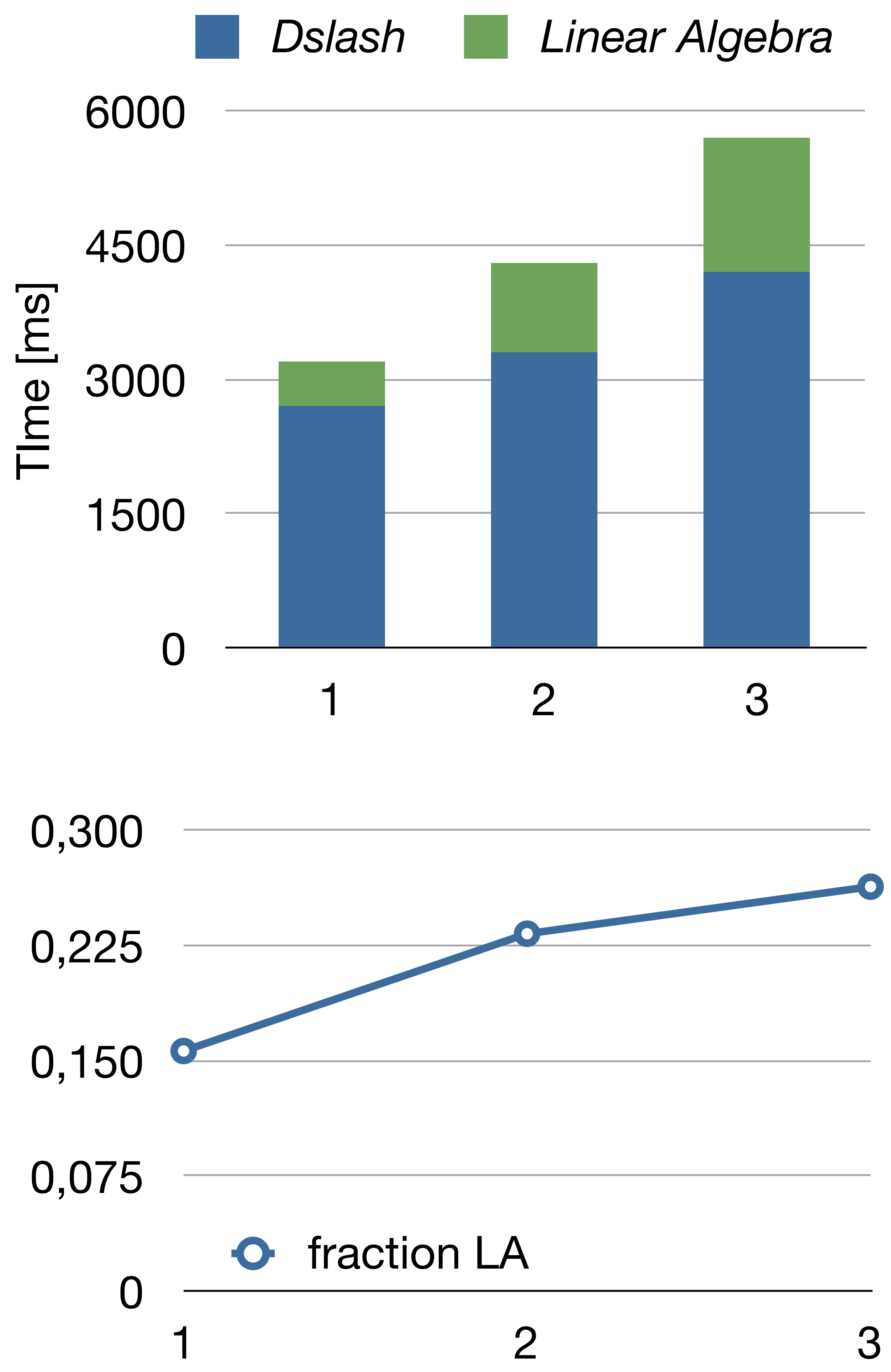


- matrix operation (Dslash) for multiple r.h.s.
- linear algebra operations cannot
  - float * vector + vector
  - norms
- linear algebra scales linear #r.h.s.
- for three r.h.s up to 25% of the runtime

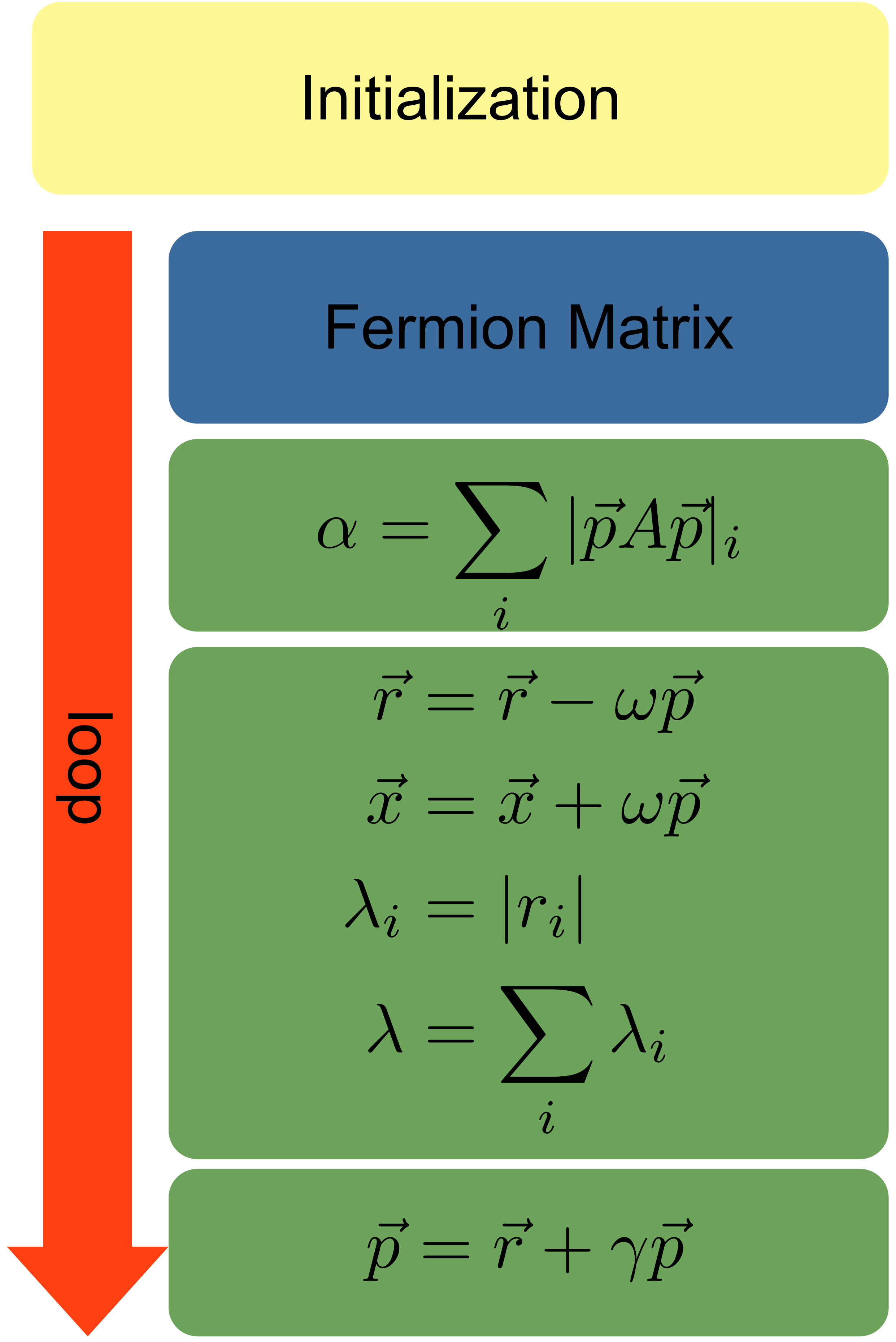


Universität Bielefeld

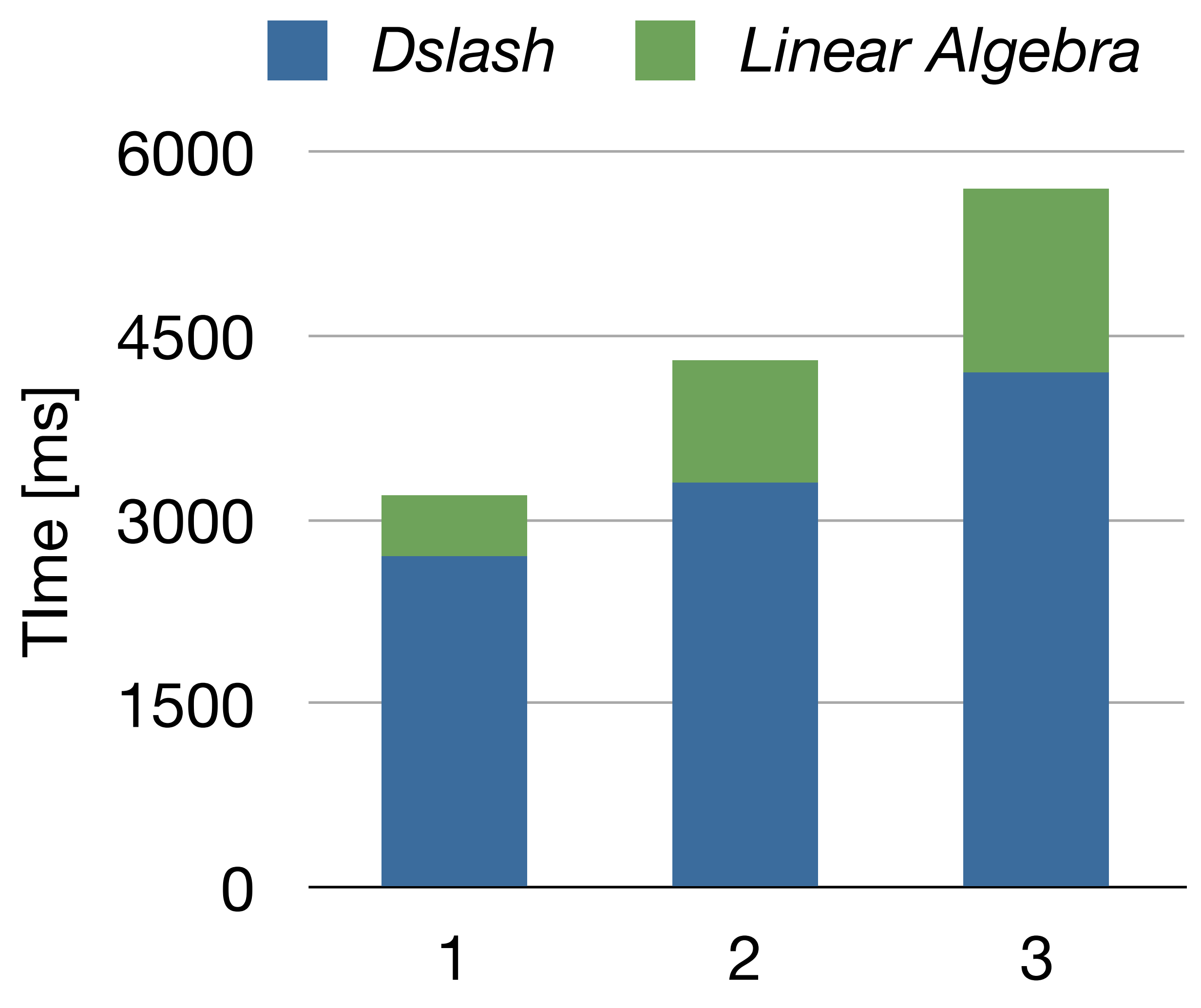
Linear algebra becomes relevant

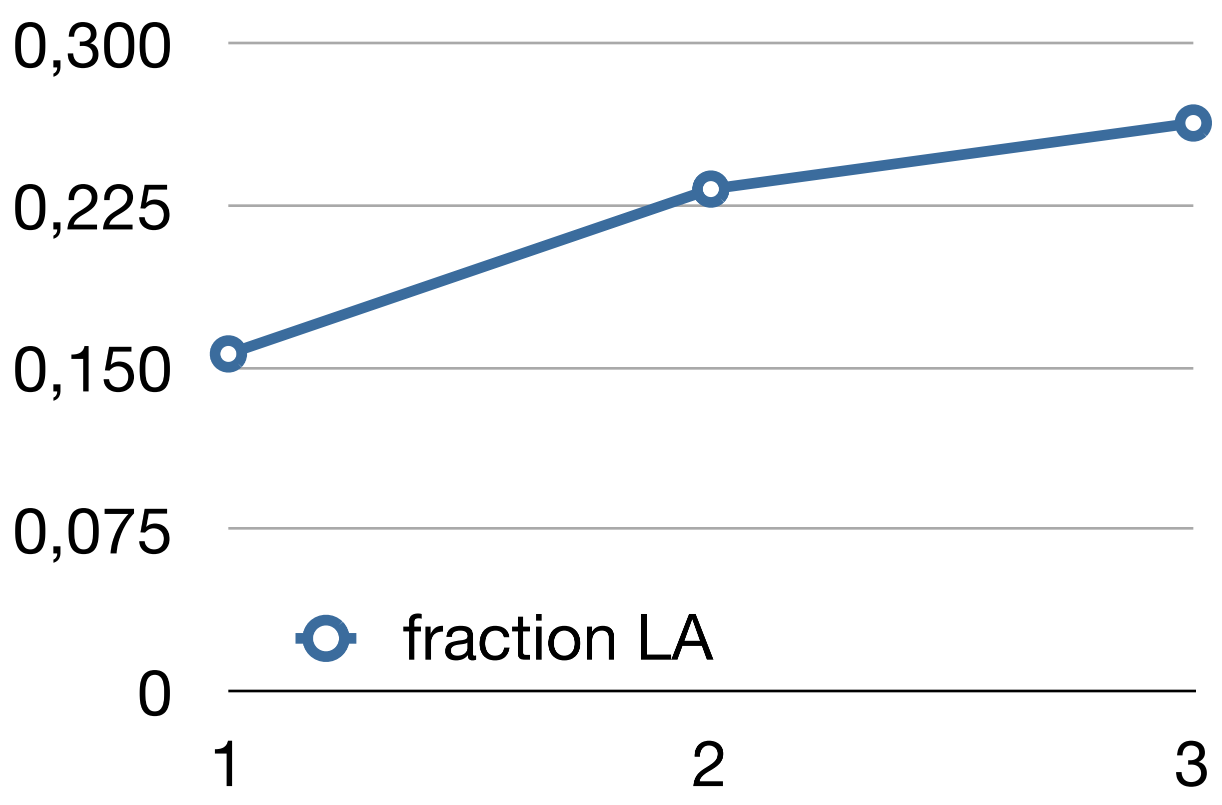


- matrix operation (Dslash) for multiple r.h.s.
- linear algebra operations cannot
  - float * vector + vector
  - norms
- linear algebra scales linear #r.h.s.
- for three r.h.s up to 25% of the runtime
- more crucial for ‘cheaper’ matrix operations



Linear algebra: reducing PCI latencies





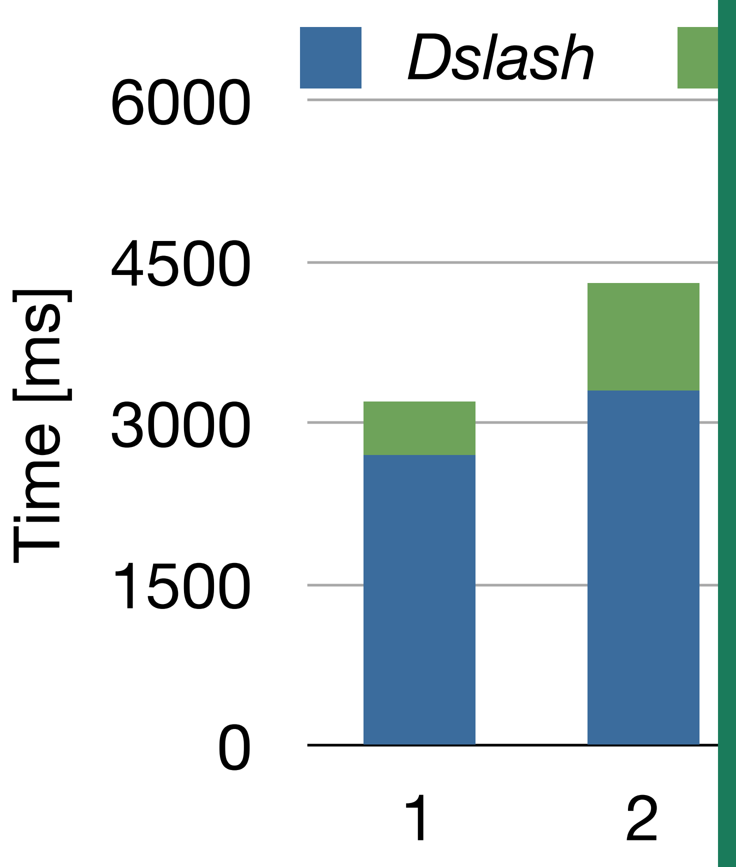
- Kernel calculates for each component i of each r.h.s. x: $\alpha_i^{(x)} = |r_i^{(x)}|$
- need to do reduction (→ see CUDA samples, M. Harris) for each r.h.s.

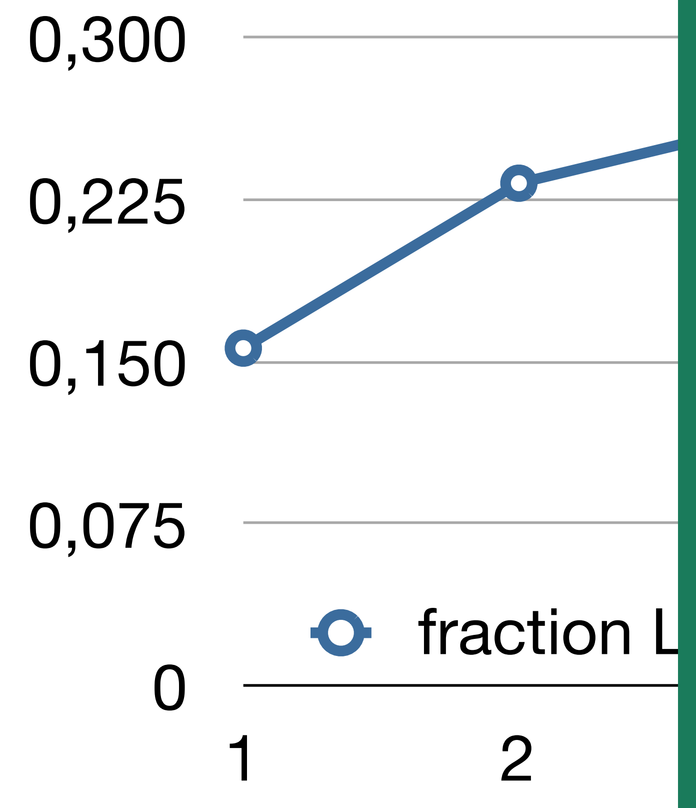
$$\alpha_j'^{(x)} = \sum_{\text{some } i} \alpha_i^{(x)}$$

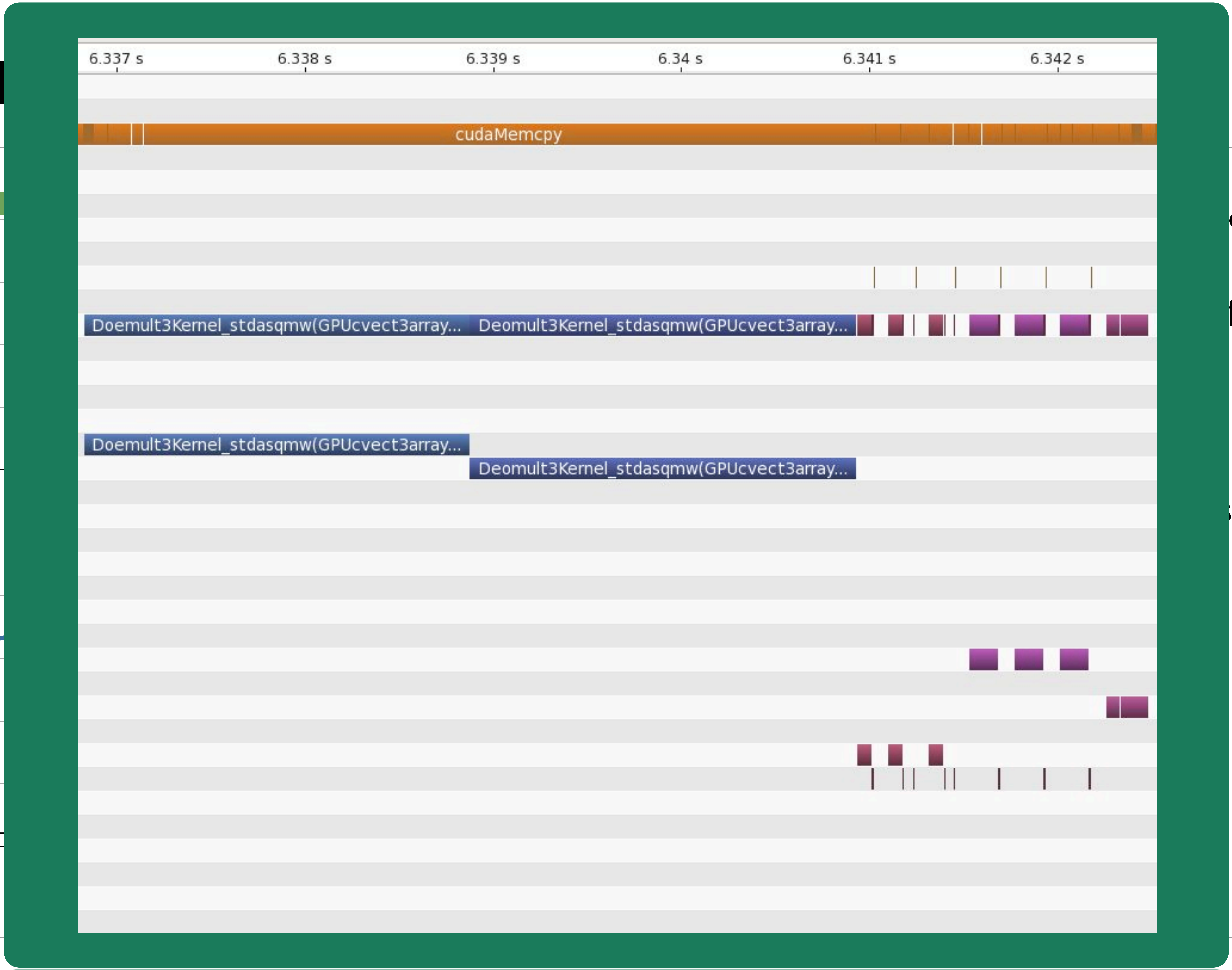
- copy data to host (one device to host copy for each r.h.s)

Universität Bielefeld

Linear algel







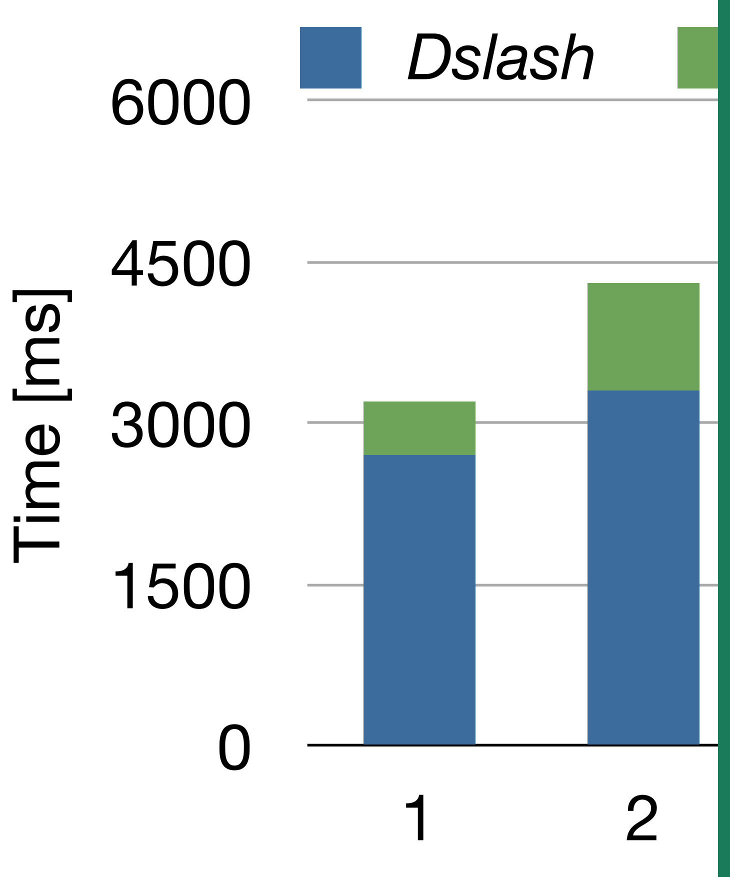
$\alpha_i^{(x)} = |r_i^{(x)}|$

for each r.h.s.

)

Universität Bielefeld

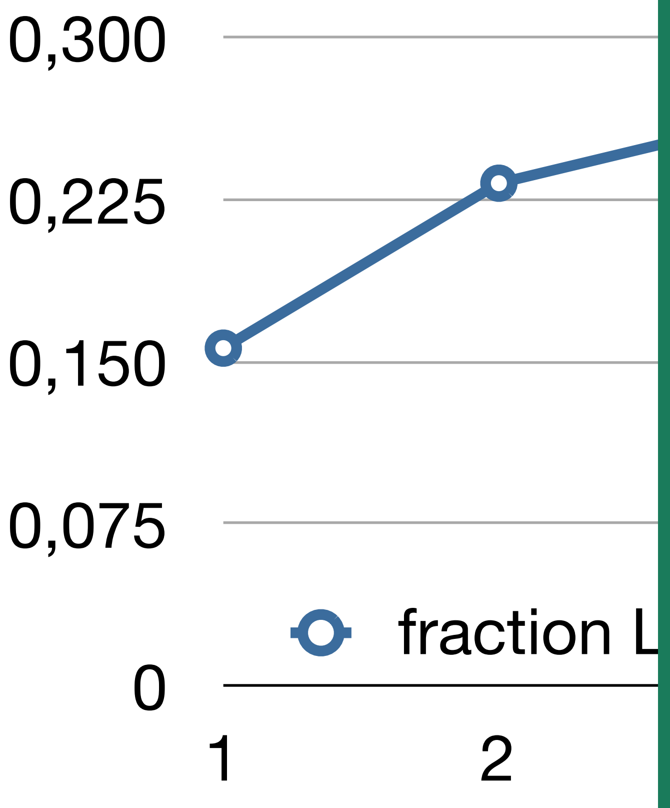
Linear algel

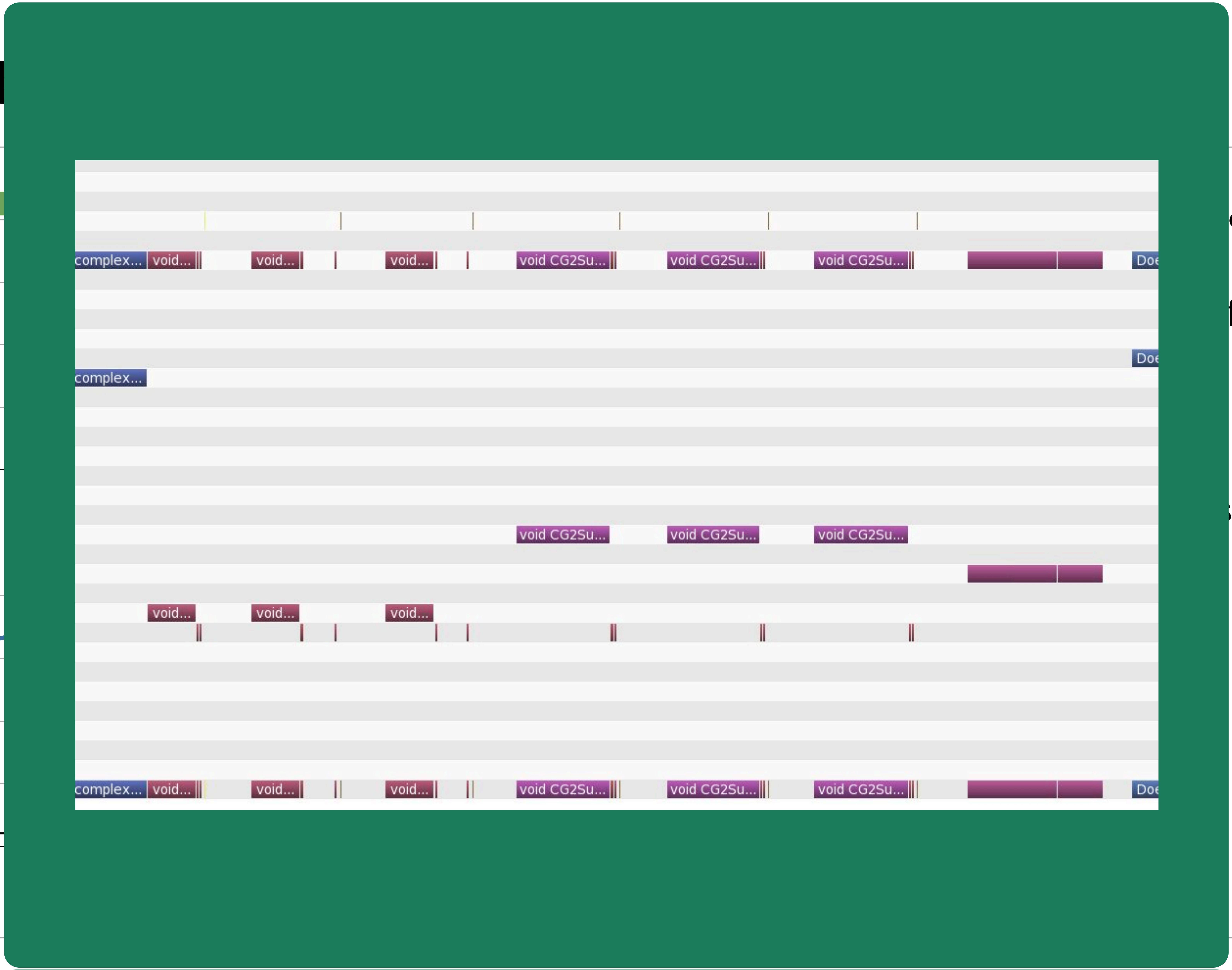


$\alpha_i^{(x)} = |r_i^{(x)}|$

for each r.h.s.

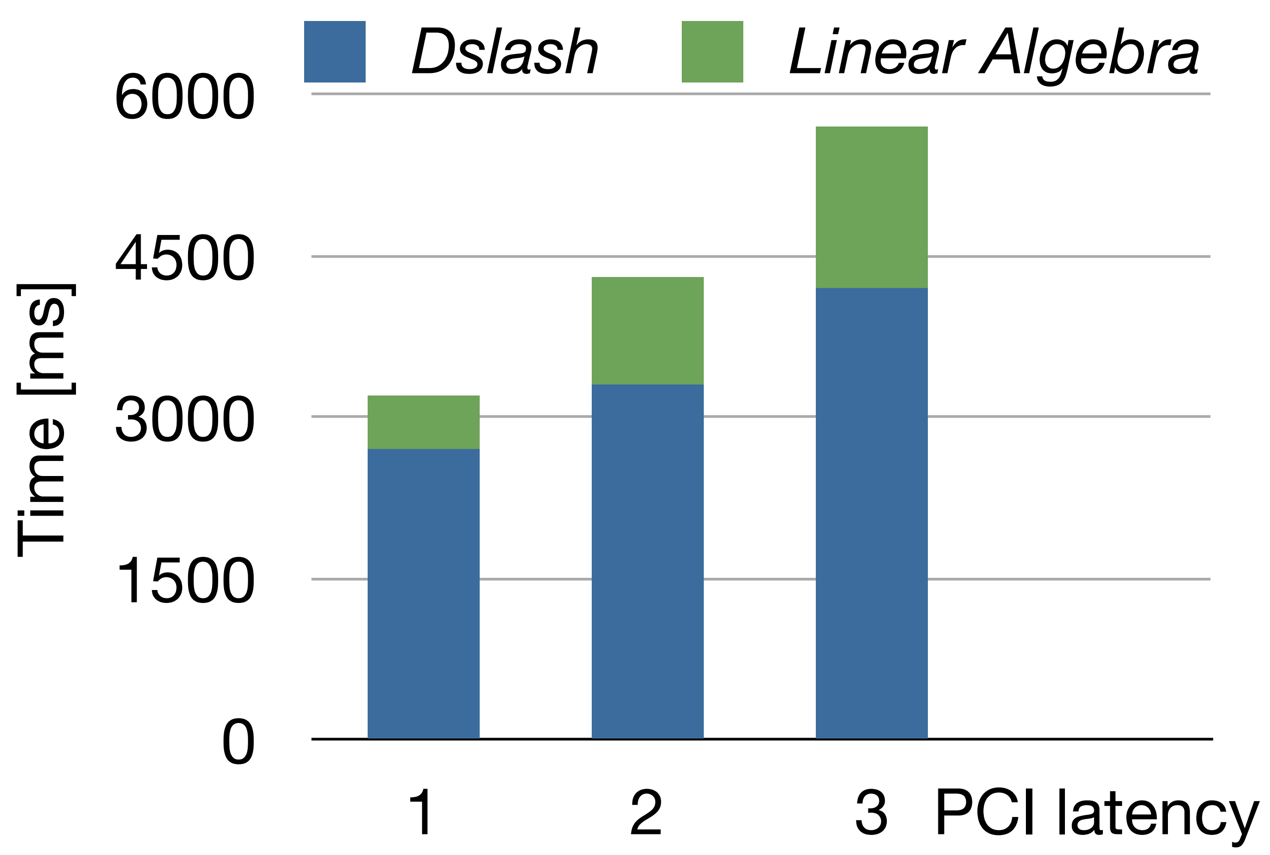
)

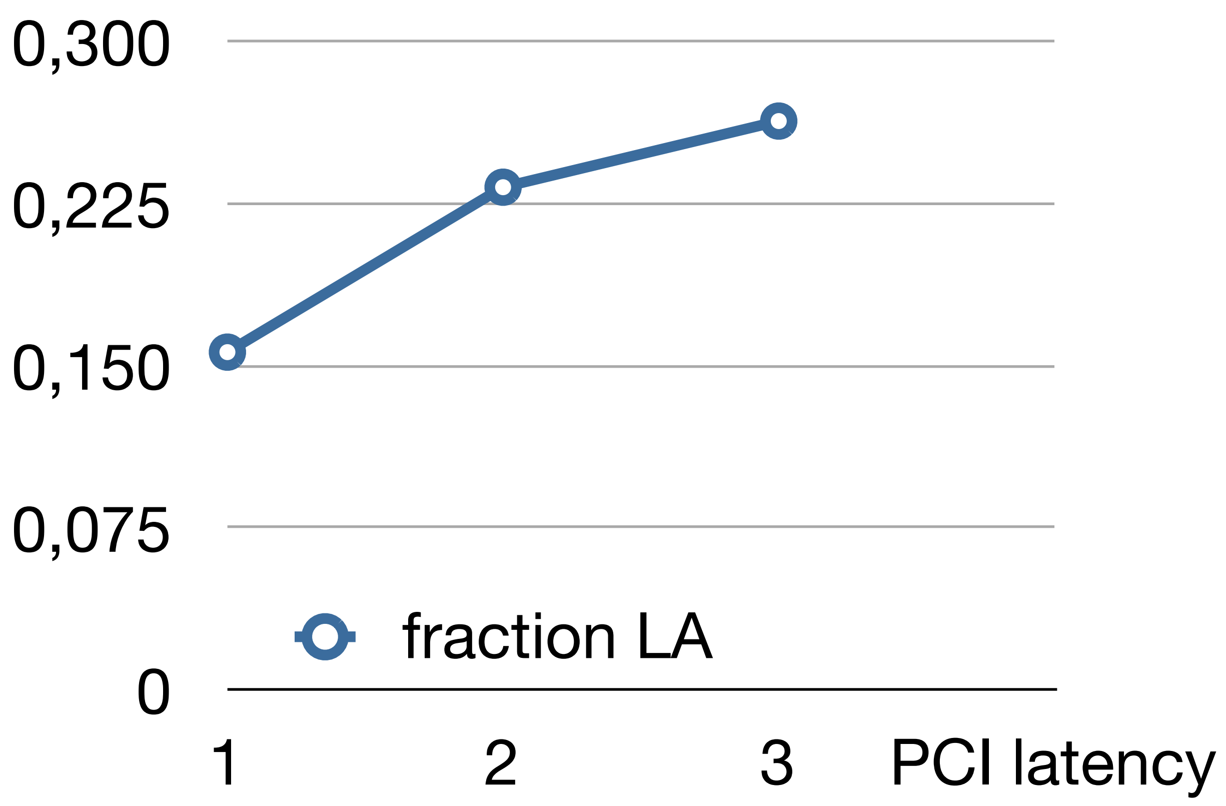




Universität Bielefeld

Linear algebra: reducing PCI latencies





- Kernel calculates for each component i of each r.h.s. x: $\alpha_i^{(x)} = |r_i^{(x)}|$
- need to do reduction (→ see CUDA samples, M. Harris) for each r.h.s.

$$\alpha_j'^{(x)} = \sum_{\text{some } i} \alpha_i^{(x)}$$

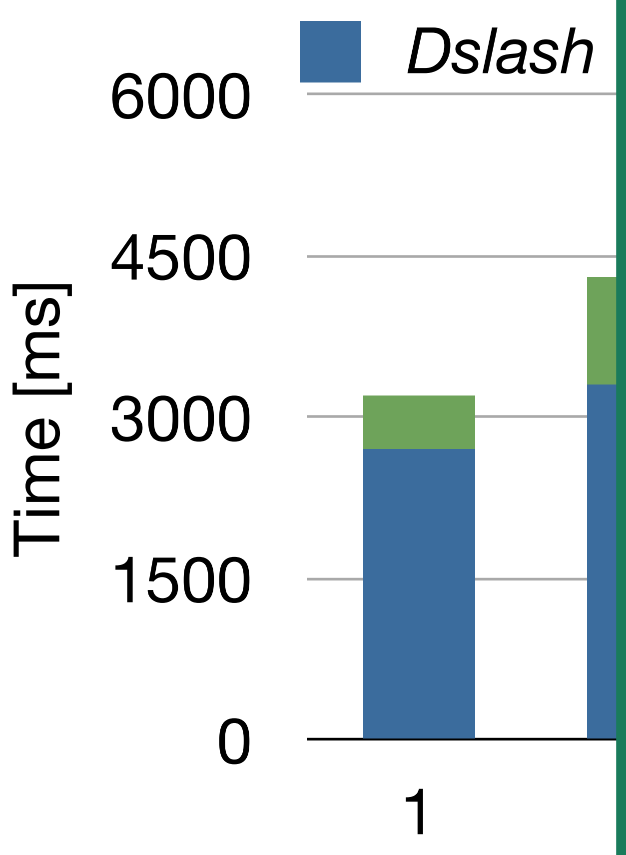
- copy data to host (one device to host copy for each r.h.s)
- combine device to host copies to one for all r.h.s.

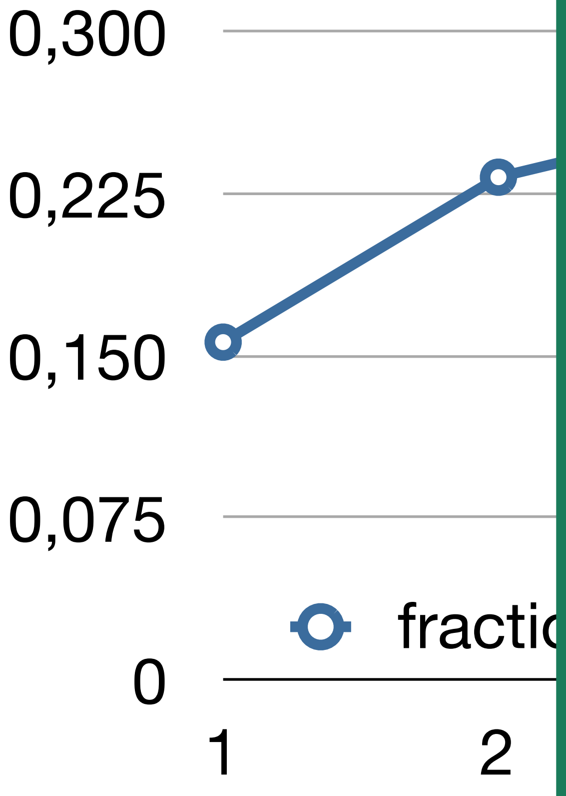
$$\alpha' = \left(\alpha_j'^{(x=0)}, \ldots, \alpha_j'^{(x=N)}\right)$$

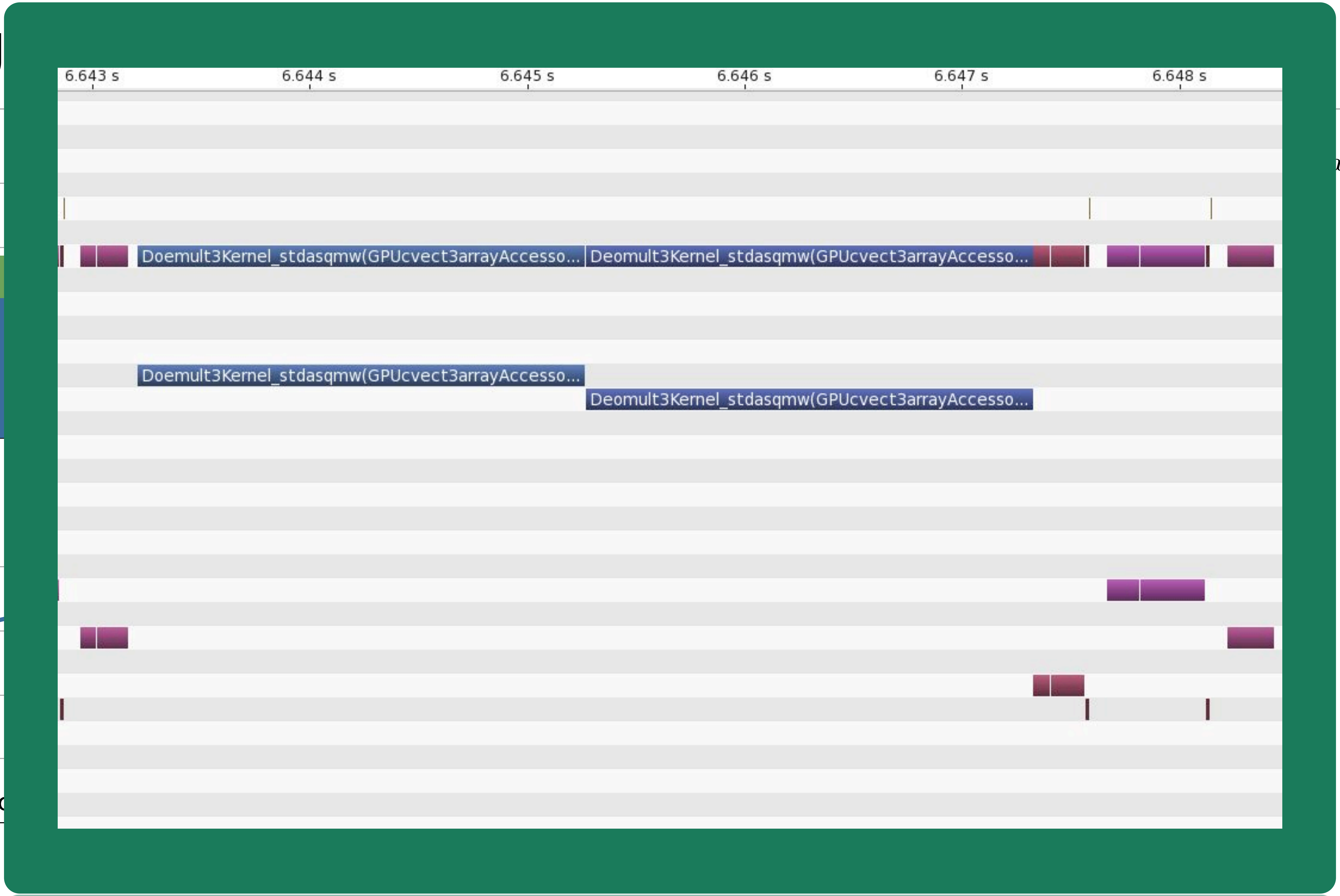
  - last reduction step can be done on CPU or GPU

Universität Bielefeld

Linear alg





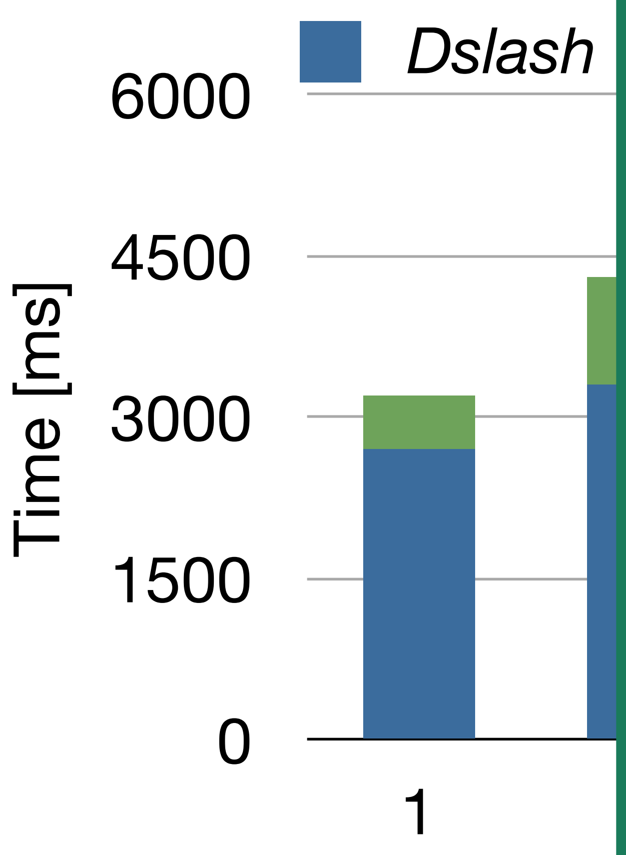


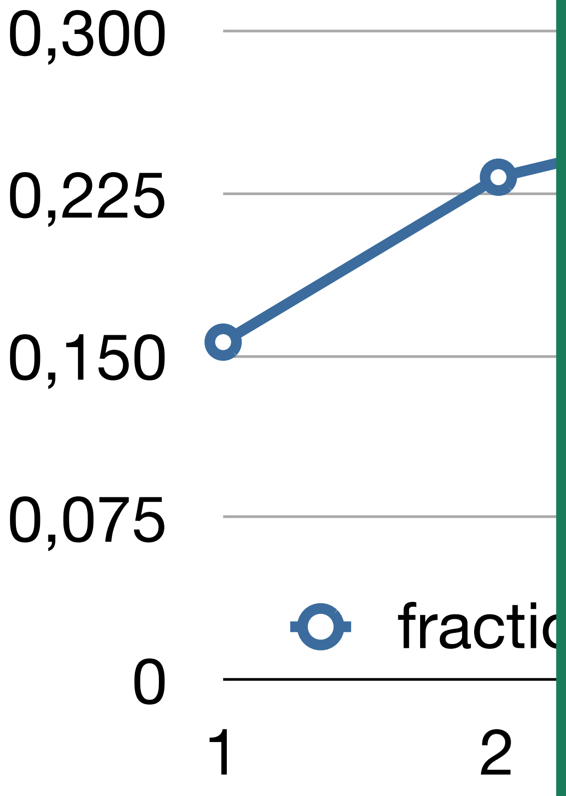
$x) = |r_i^{(x)}|$

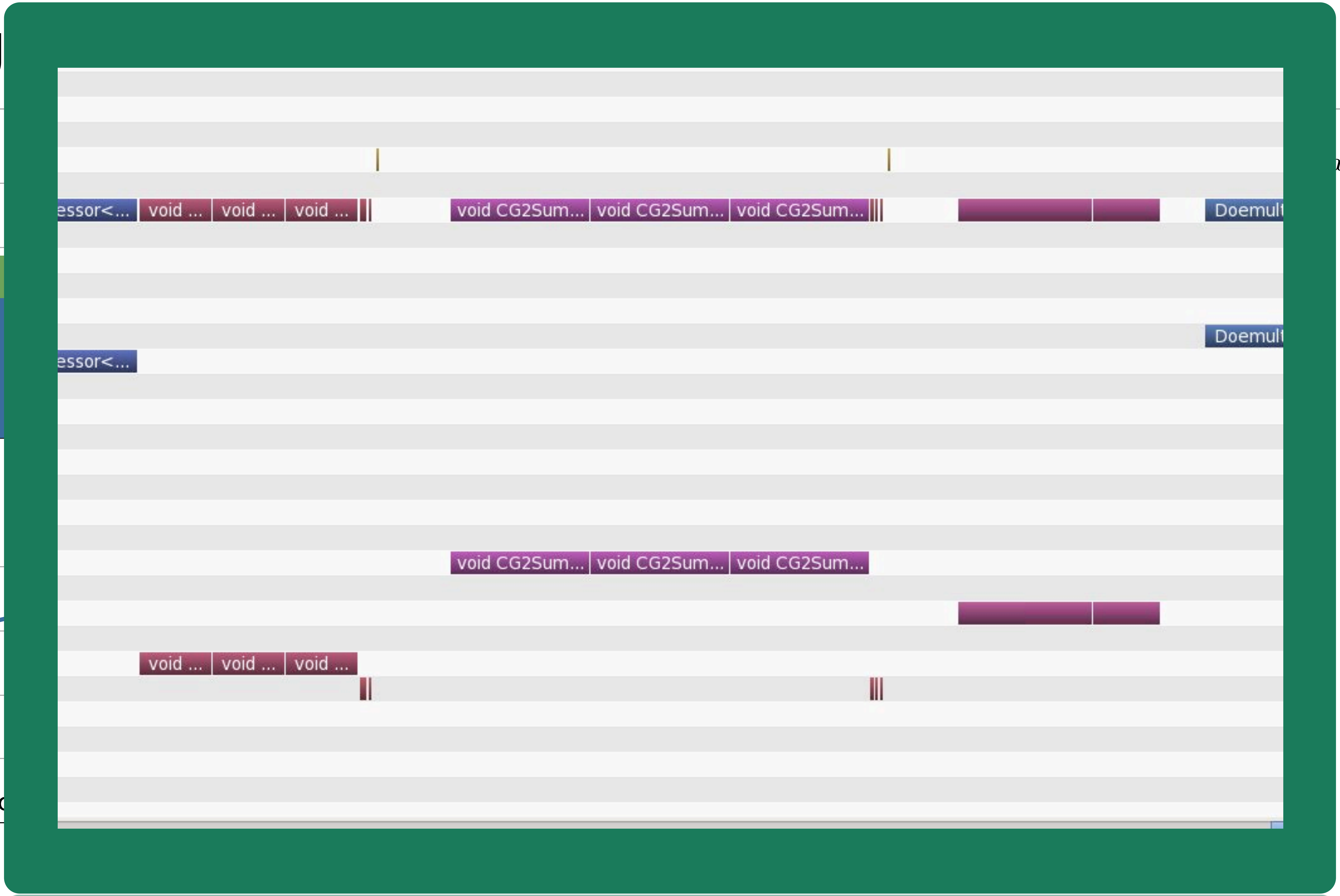
each r.h.s.

Universität Bielefeld

Linear alg





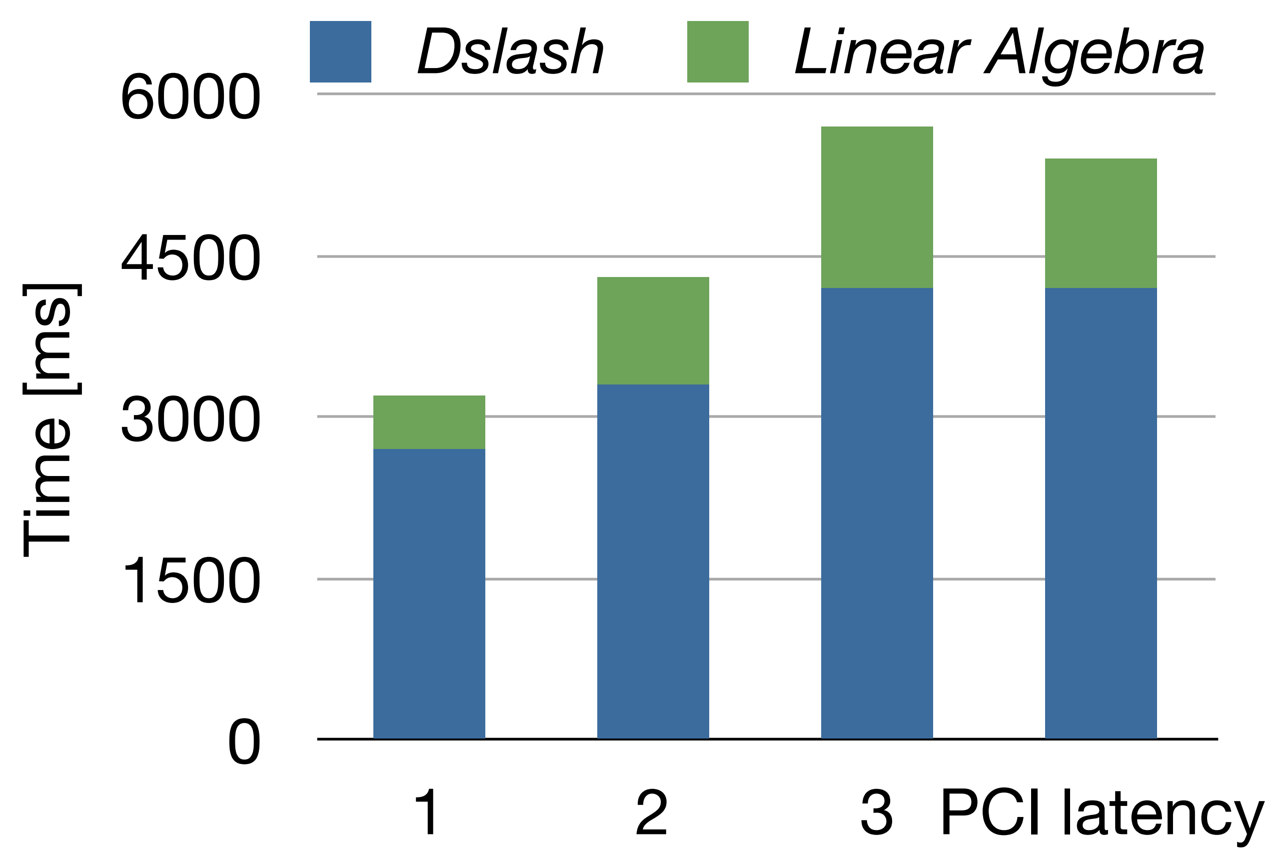


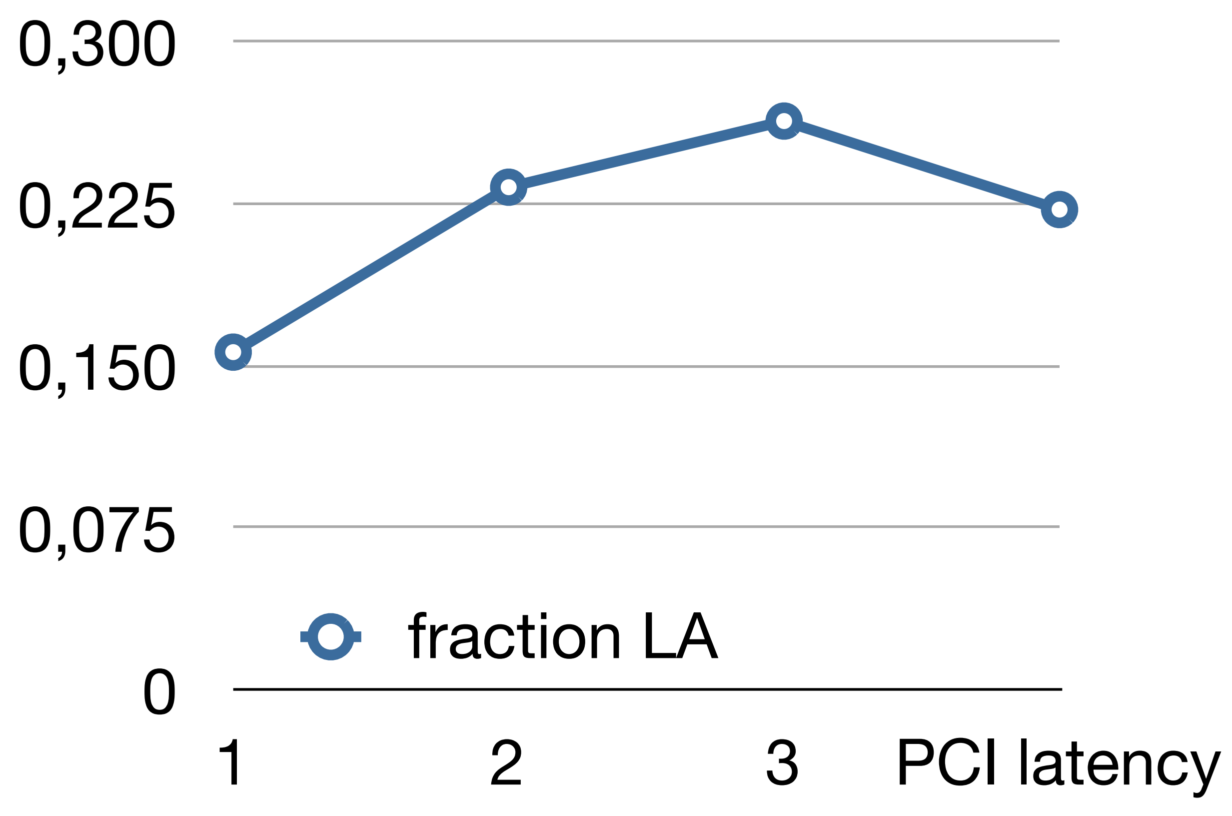
$x) = |r_i^{(x)}|$

each r.h.s.

Universität Bielefeld

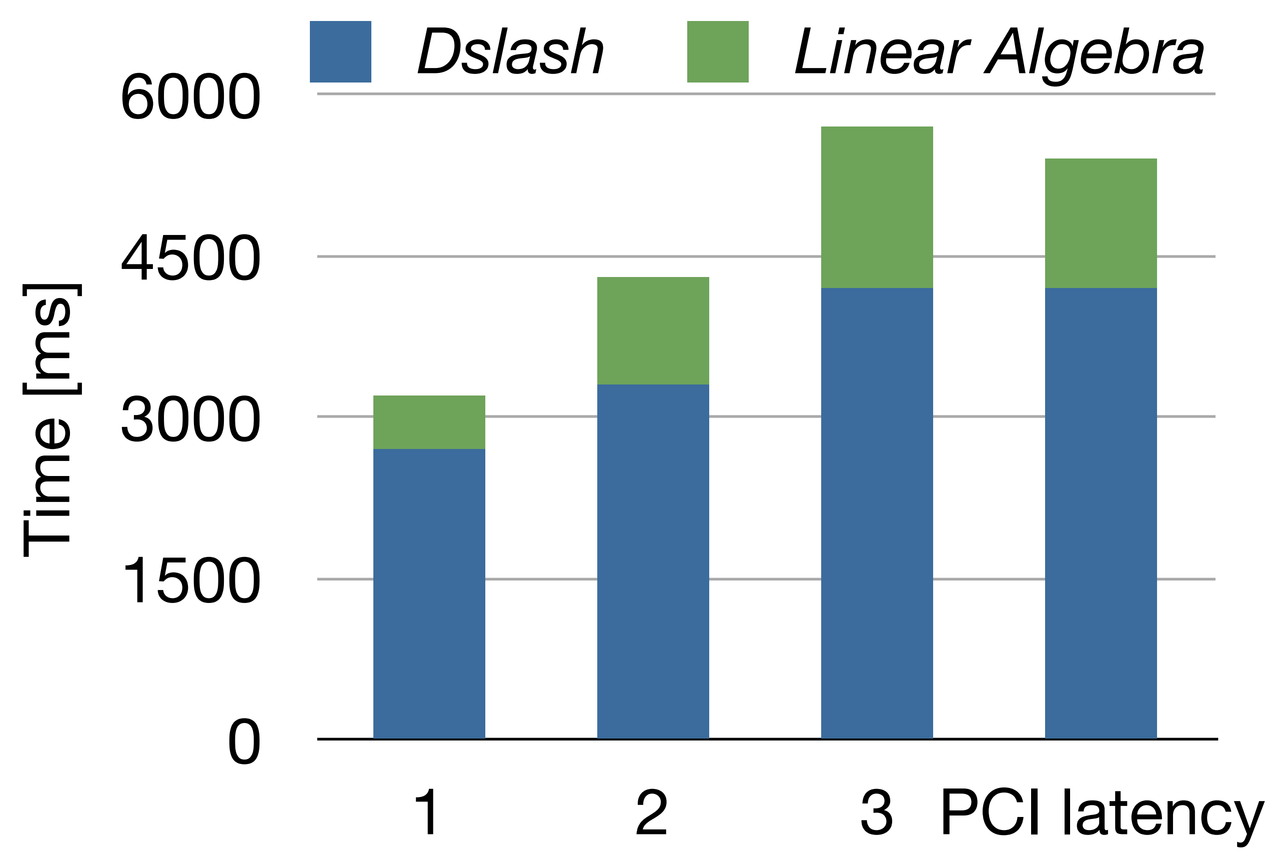
Linear algebra: improve reduction

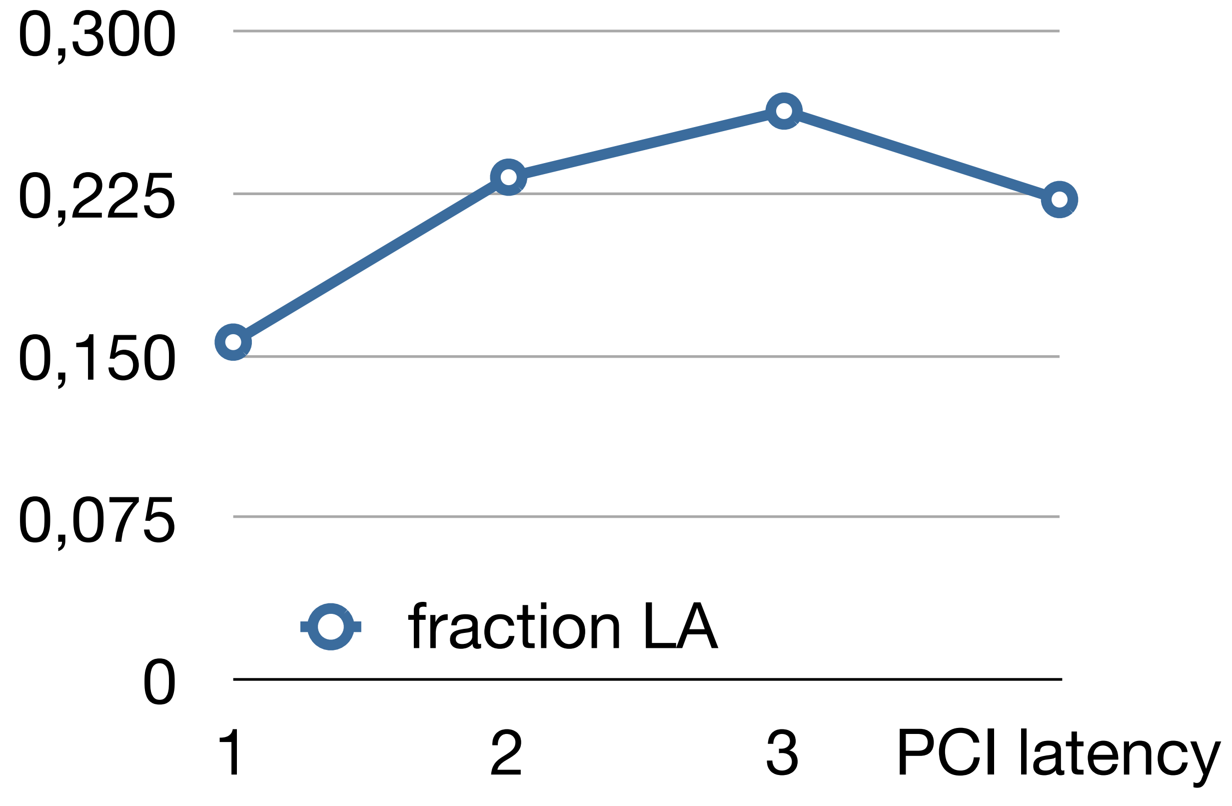




- standard way of doing reduction
    - calculate floating point numbers that shall be reduced + reduction
- but data are already created on GPU

Linear algebra: improve reduction



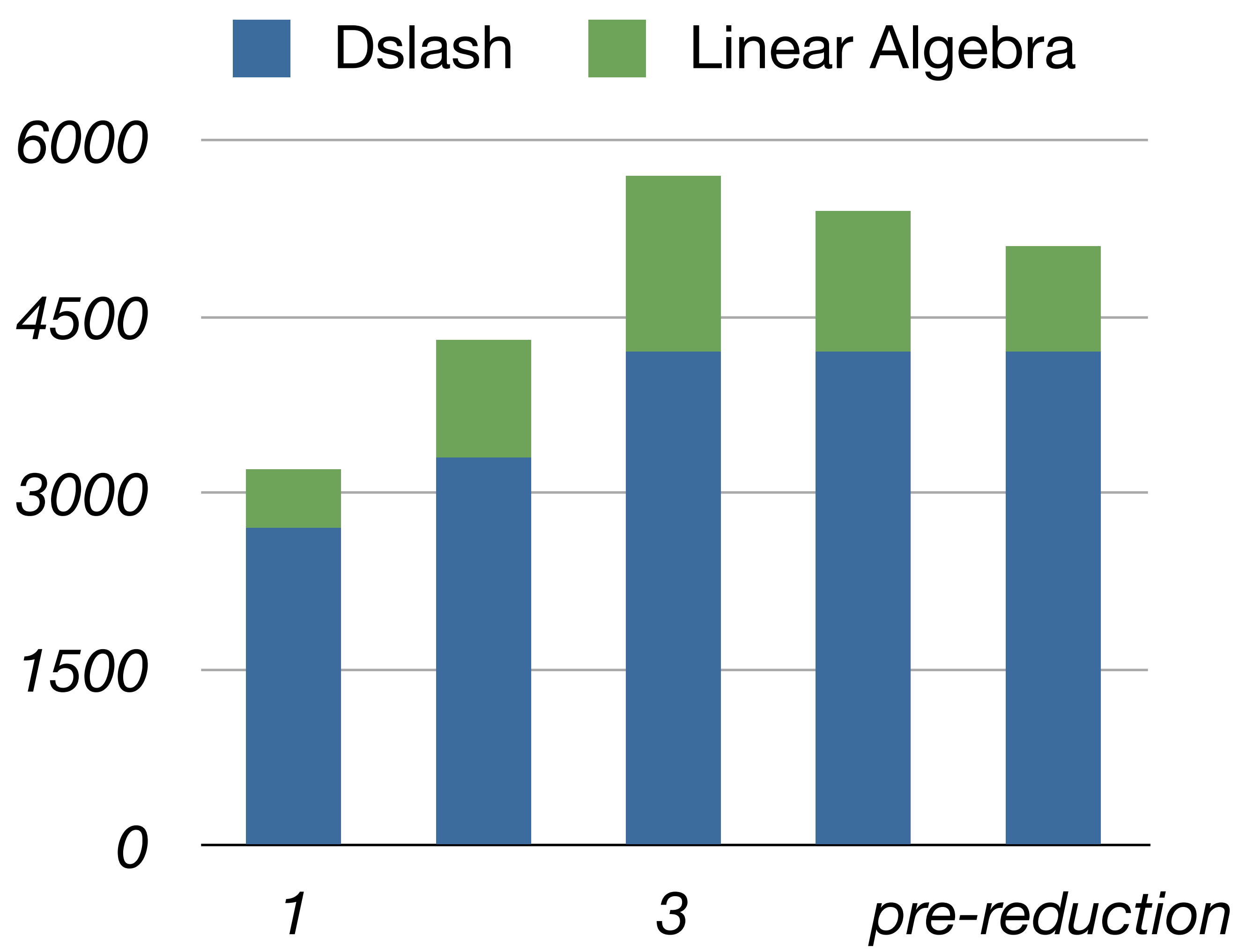


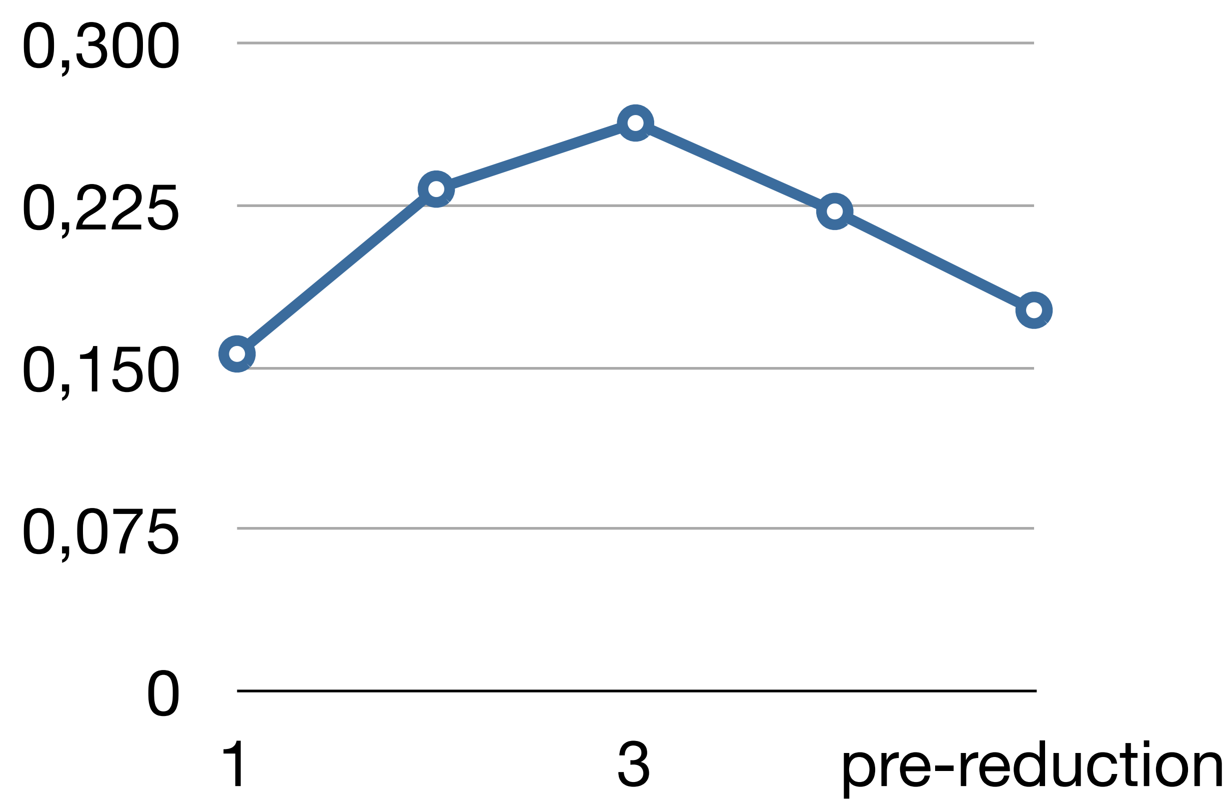
- standard way of doing reduction
  - calculate floating point numbers that shall be reduced + reduction
- but data are already created on GPU
- pre-reduction during ‘creation’
  - does not affect runtime
  - faster reduction (4x)
  - tune parameter `rc` (enough threads)

```
template<int rc>
__global__ void Kernel( vector p, vector loc_s, double *alpha, float mass )
{
  const int x = rc*blockDim.x * blockIdx.x + threadIdx.x;
  double a =0.;
#pragma unroll
  for(int r=rc-1; r>=0; r--){
      const int xr=x+r*blockDim.x;
      if( xr < c_latticeSize.sizeh() ){
          pi = p[xr];;
          temp  = mass2*pi - s[i];
          a  += (double) dot_prodf(pi,temp);
          s[xr]=temp;
          if (r==0)
            alpha[blockDim.x * blockIdx.x + threadIdx.x]=a;
      }
  }
}
```


Universität Bielefeld

Linear algebra: improve reduction





- standard way of doing reduction
  - calculate floating point numbers that shall be reduced + reduction
- but data are already created on GPU
- pre-reduction during 'creation'
  - does not affect runtime
  - faster reduction (4x)
  - tune parameter `rc` (enough threads)

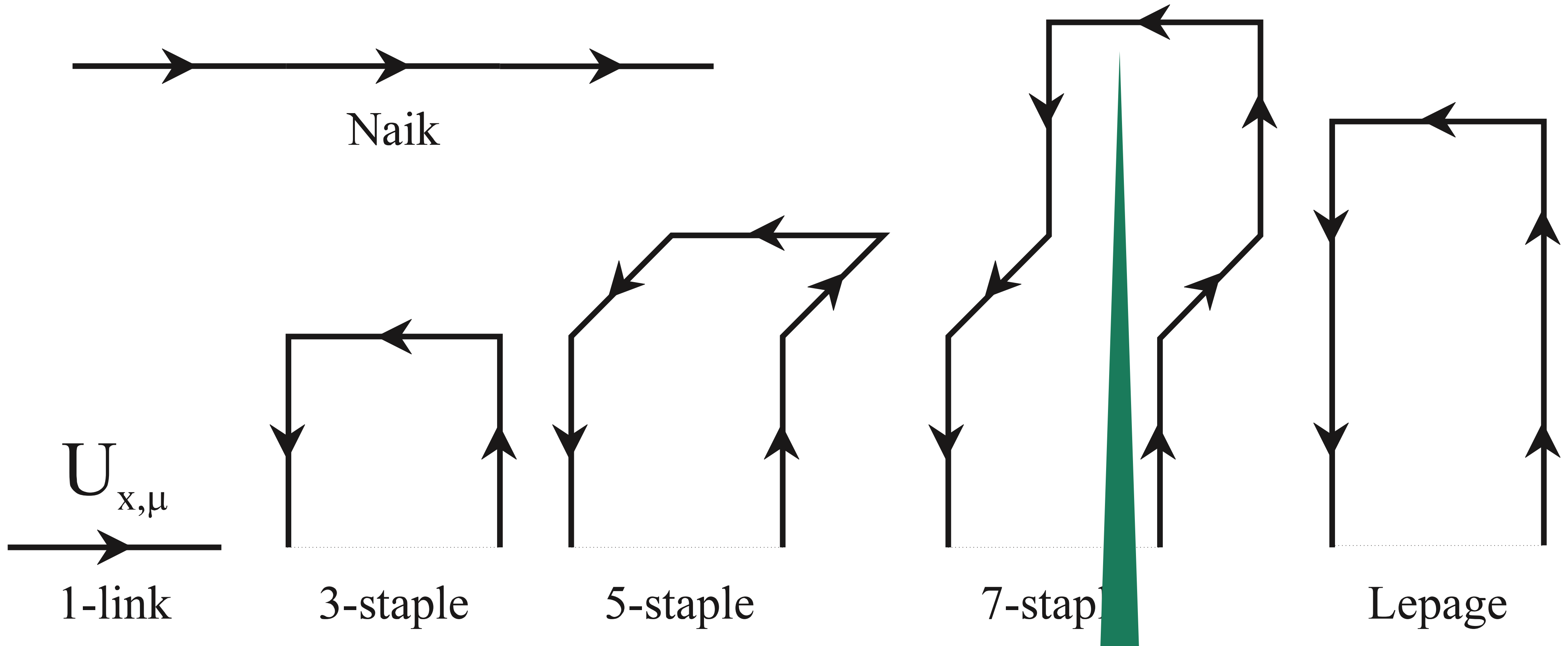
```
template<int rc>
__global__ void Kernel( vector p, vector loc_s, double *alpha, float mass )
{
  const int x = rc*blockDim.x * blockIdx.x + threadIdx.x;
  double a =0.;
#pragma unroll
  for(int r=rc-1; r>=0; r--){
      const int xr=x+r*blockDim.x;
      if( xr < c_latticeSize.sizeh() ){
          pi = p[xr];;
          temp  = mass2*pi - s[i];
          a  += (double) dot_prodf(pi,temp);
          s[xr]=temp;
          if (r==0)
            alpha[blockDim.x * blockIdx.x + threadIdx.x]=a;
      }
  }
}
```

Configuration generation on GPUs

- we use a full hybrid-monte Carlo simulation on GPU (HISQ action)
- no PCI bus bottleneck
- current runs with lattice size 32^3 x 8 in single precision
- ECC reduces memory bandwidth: costs roughly 30% performance
- lattices up to 48^3 x 12 fit on one Tesla cards with 6GB (double precision)
  - runtime is an issue - at least use several GPUs in one node
- larger lattices (64^3 x 16) → use compute time on capacity computing machines (BlueGene)
- we aim at getting the best scientific output out of limited resources (#GPUs, available supercomputer time)

Registers pressure

- improved fermion action use smeared links
  - require sum over products of up to 7 SU(3) matrices
  - SU(3) Matrix: 18 / 36 registers

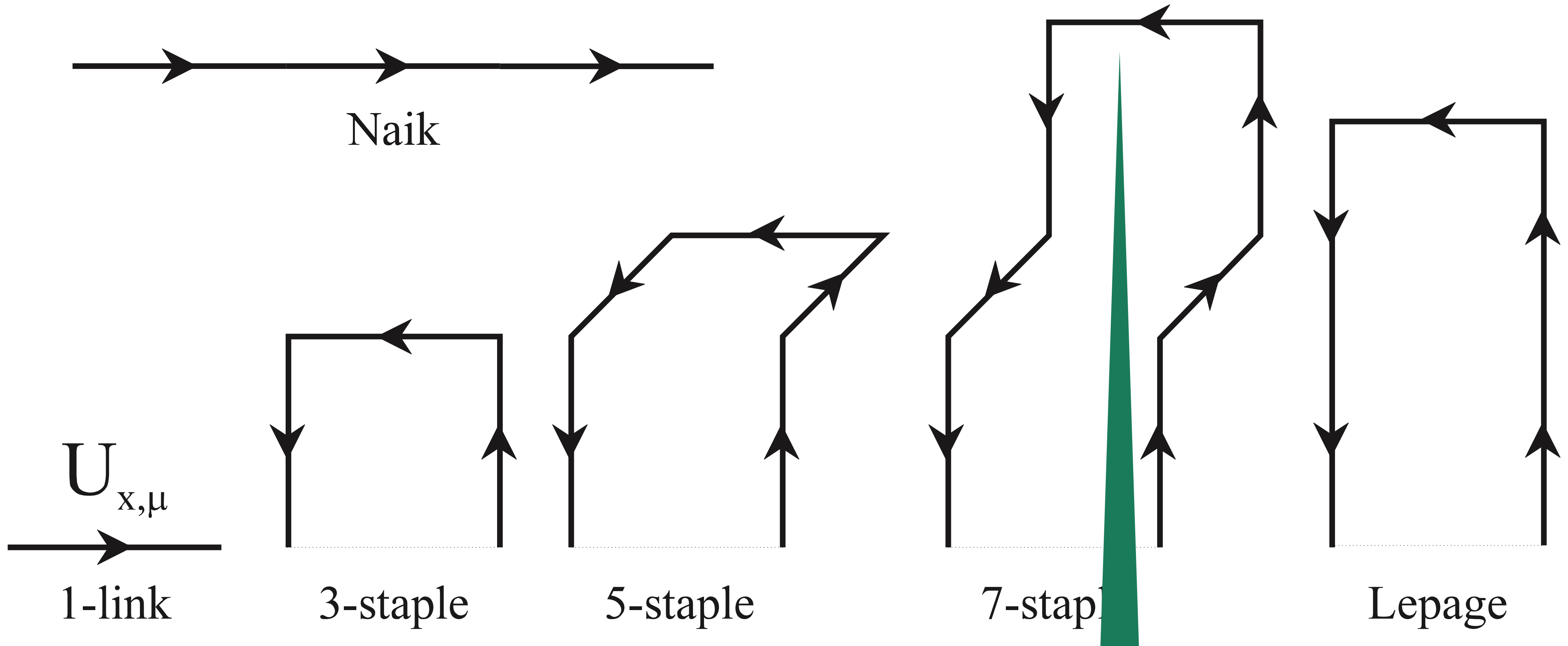


Fermion force in MD
→ take derivatives of smeared links with respect to ‘original’ links

Universität Bielefeld

Registers pressure

- improved fermion action use smeared links
  - require sum over products of up to 7 SU(3) matrices
  - SU(3) Matrix: 18 / 36 registers
- Fermi architecture: 63 registers / thread
- optimize SU(3) *= SU(3) operation for register usage
- spilling causes significant performance drop for bandwidth bound kernels
  - however: spilling is often better than shared memory ➝ 48kB L1 cache
- precomputed products help but must be stored somewhere



Fermion force in MD
→ take derivatives of smeared links with respect to 'original' links

Optimizing register usage / reduce spilling

- e.g. force for the 7 link term consists of 56 products of 7 SU(3) matrices (x 24 for ‘rotations’)
- limited GPU memory: store precomputed products ?

v201203: initial version

v201204: optimized matrix mult, split into servel Kernels

v201207: minor changes for memory access

v201211: reorganized split up Kernel

v201303: reconstruction of matrices from 14 floats

Optimizing register usage / reduce spilling

- e.g. force for the 7 link term consists of 56 products of 7 SU(3) matrices (x 24 for ‘rotations’)

- limited GPU memory: store precomputed products ?

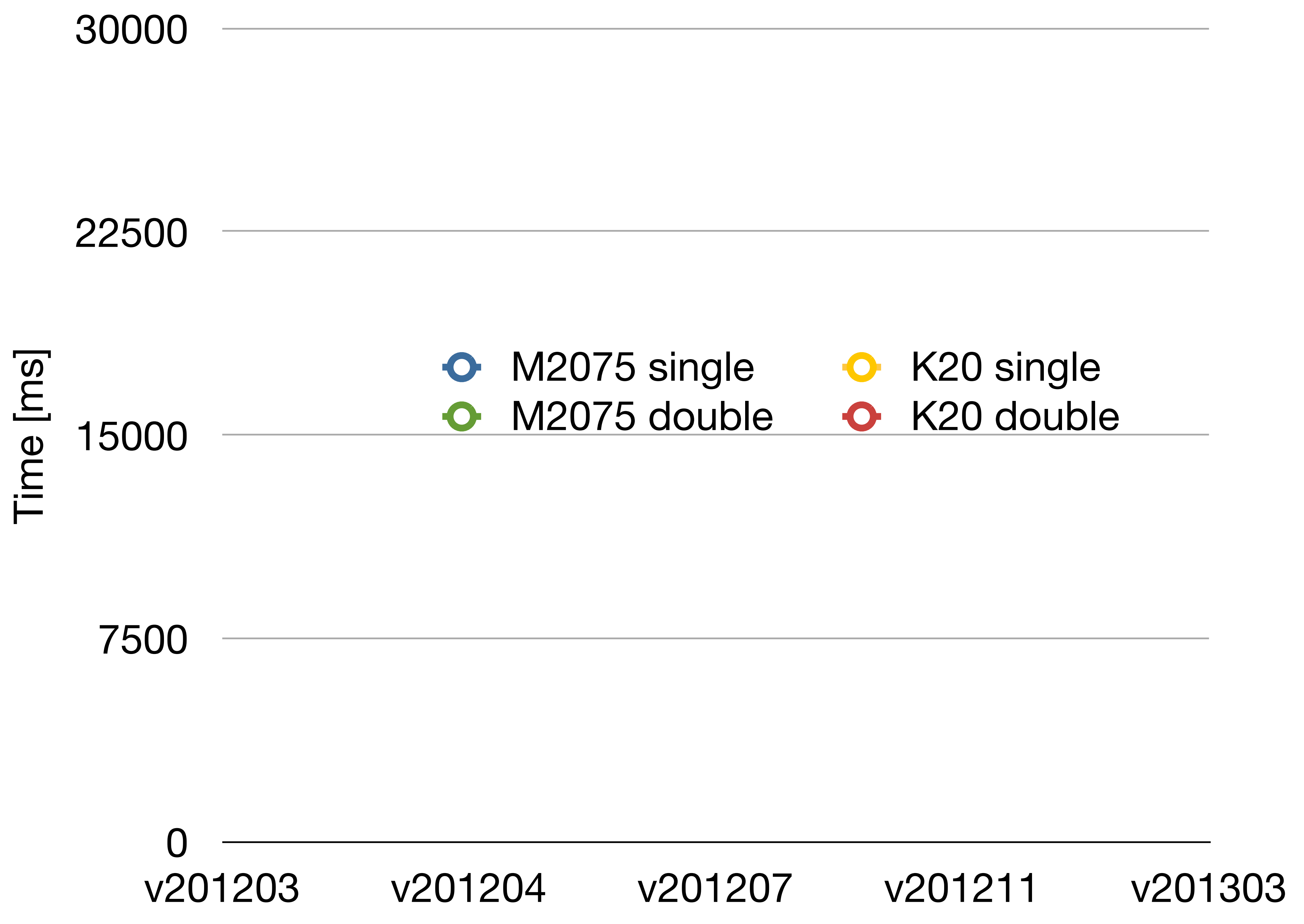
v201203: initial version

v201204: optimized matrix mult, split into servel Kernels

v201207: minor changes for memory access

v201211: reorganized split up Kernel

v201303: reconstruction of matrices from 14 floats



Optimizing register usage / reduce spilling

- e.g. force for the 7 link term consists of 56 products of 7 SU(3) matrices (x 24 for ‘rotations’)

- limited GPU memory: store precomputed products ?

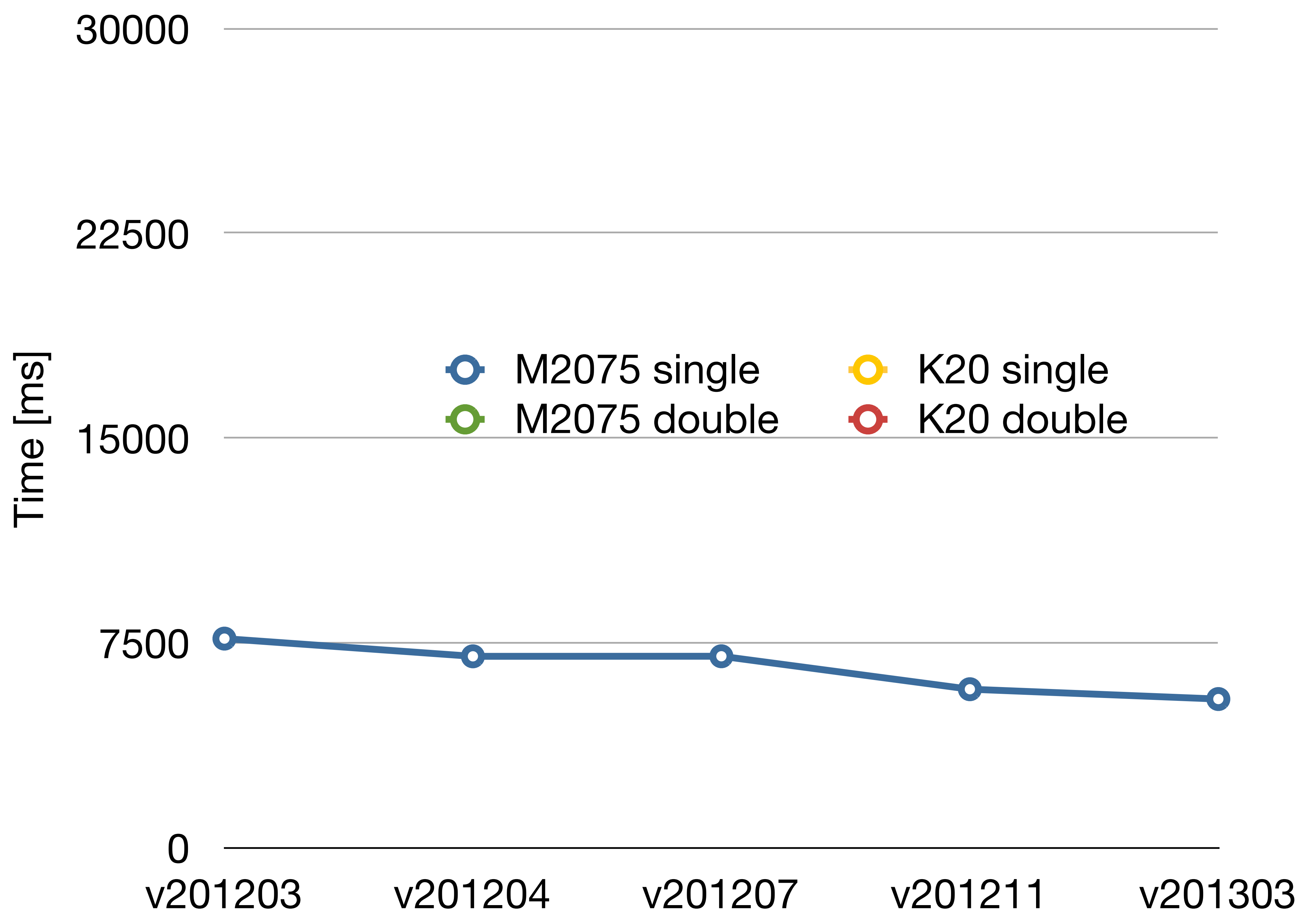
v201203: initial version

v201204: optimized matrix mult, split into servel Kernels

v201207: minor changes for memory access

v201211: reorganized split up Kernel

v201303: reconstruction of matrices from 14 floats



Universität Bielefeld

Optimizing register usage / reduce spilling

- e.g. force for the 7 link term consists of 56 products of 7 SU(3) matrices (x 24 for ‘rotations’)
- limited GPU memory: store precomputed products ?

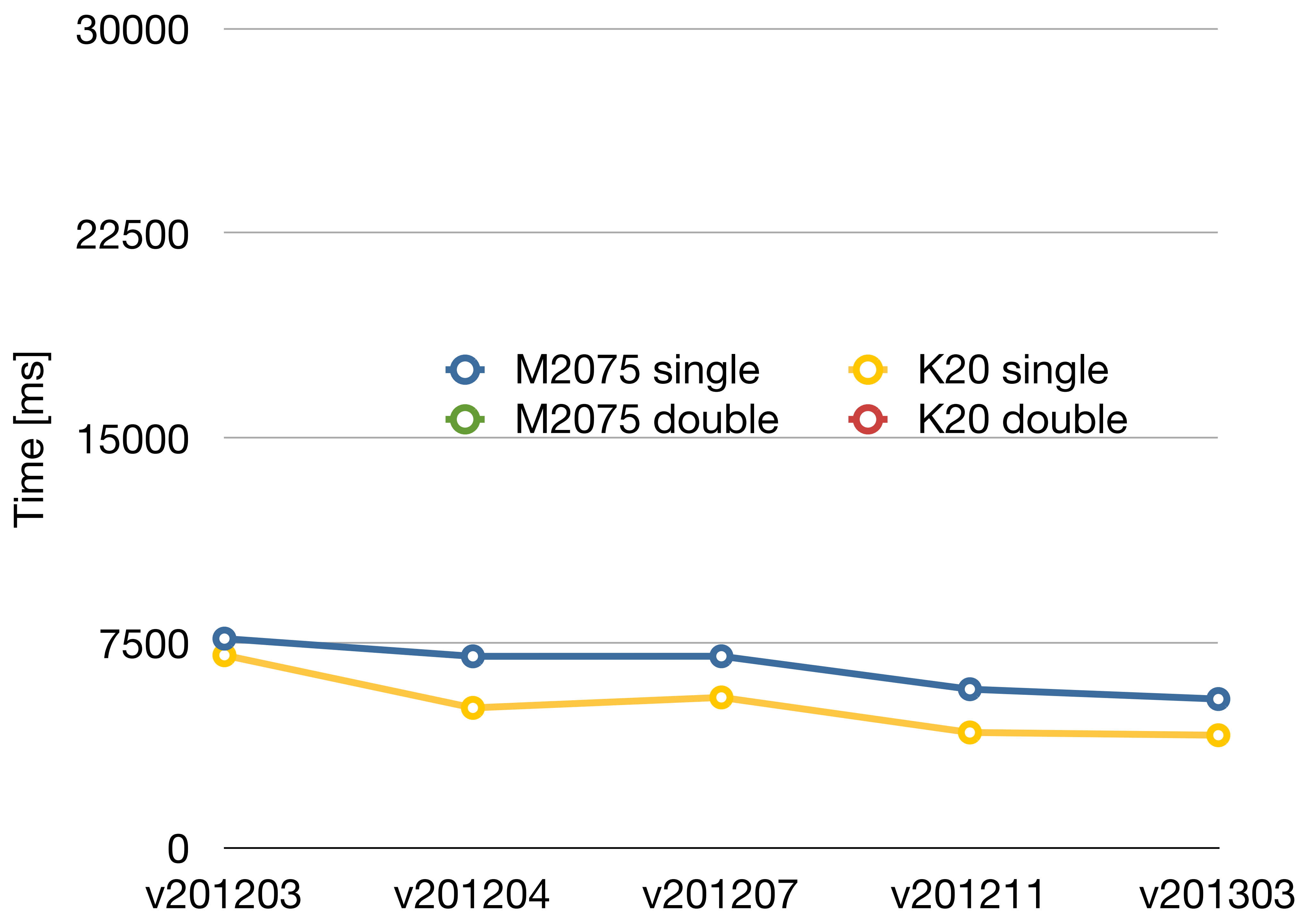
v201203: initial version

v201204: optimized matrix mult, split into servel Kernels

v201207: minor changes for memory access

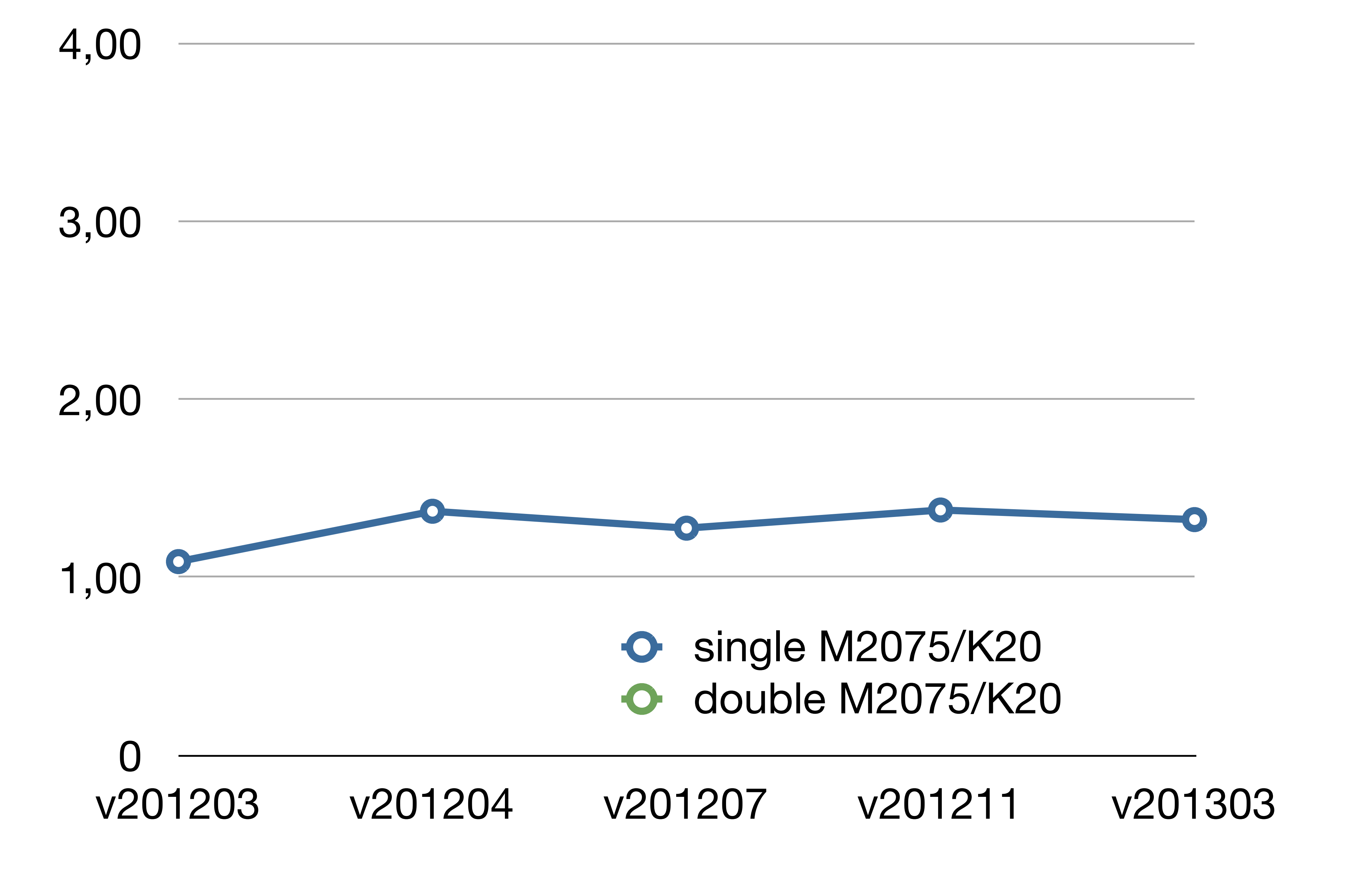
v201211: reorganized split up Kernel

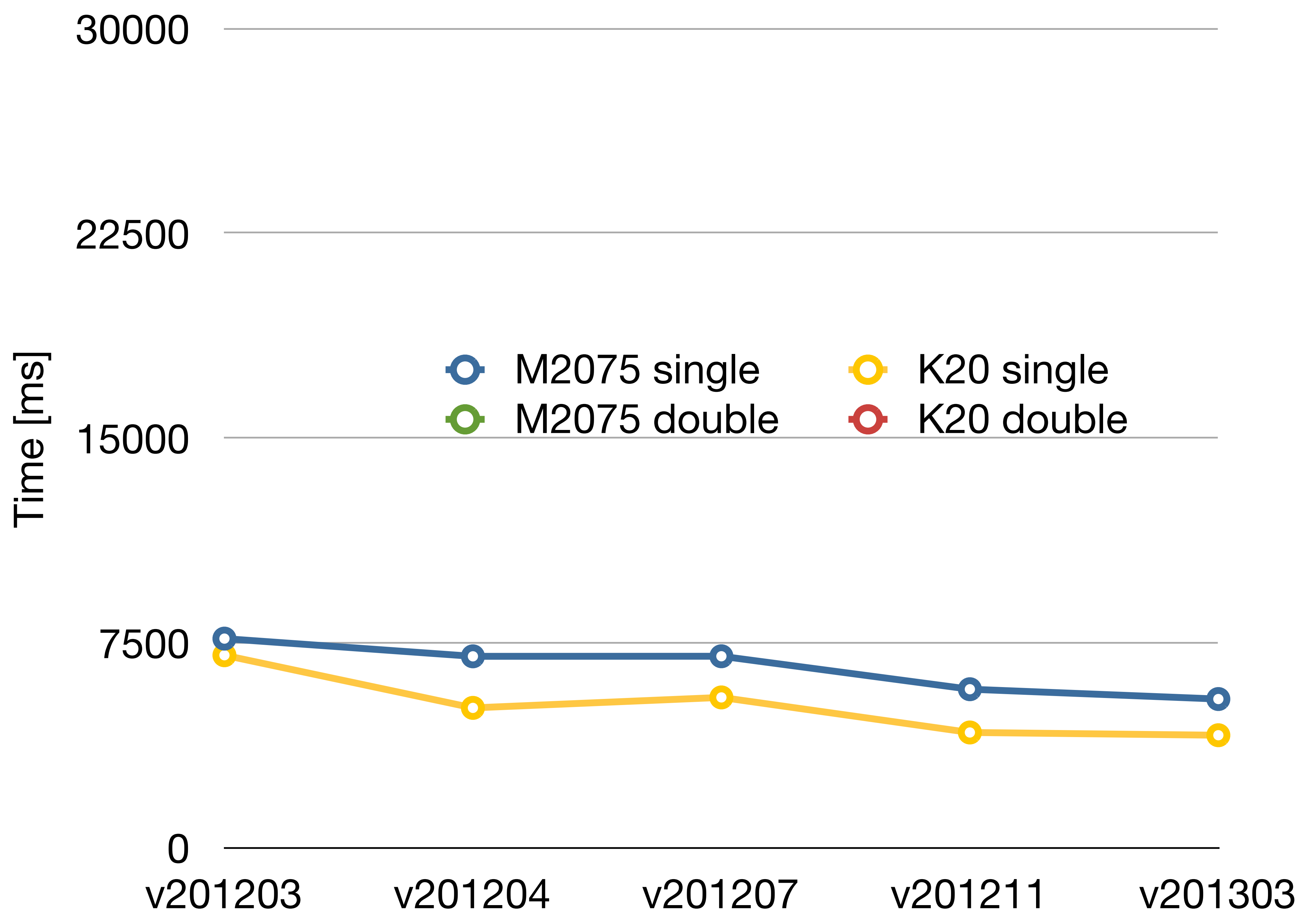
v201303: reconstruction of matrices from 14 floats



Optimizing register usage / reduce spilling

- e.g. force for the 7 link term consists of 56 products of 7 SU(3) matrices (x 24 for ‘rotations’)

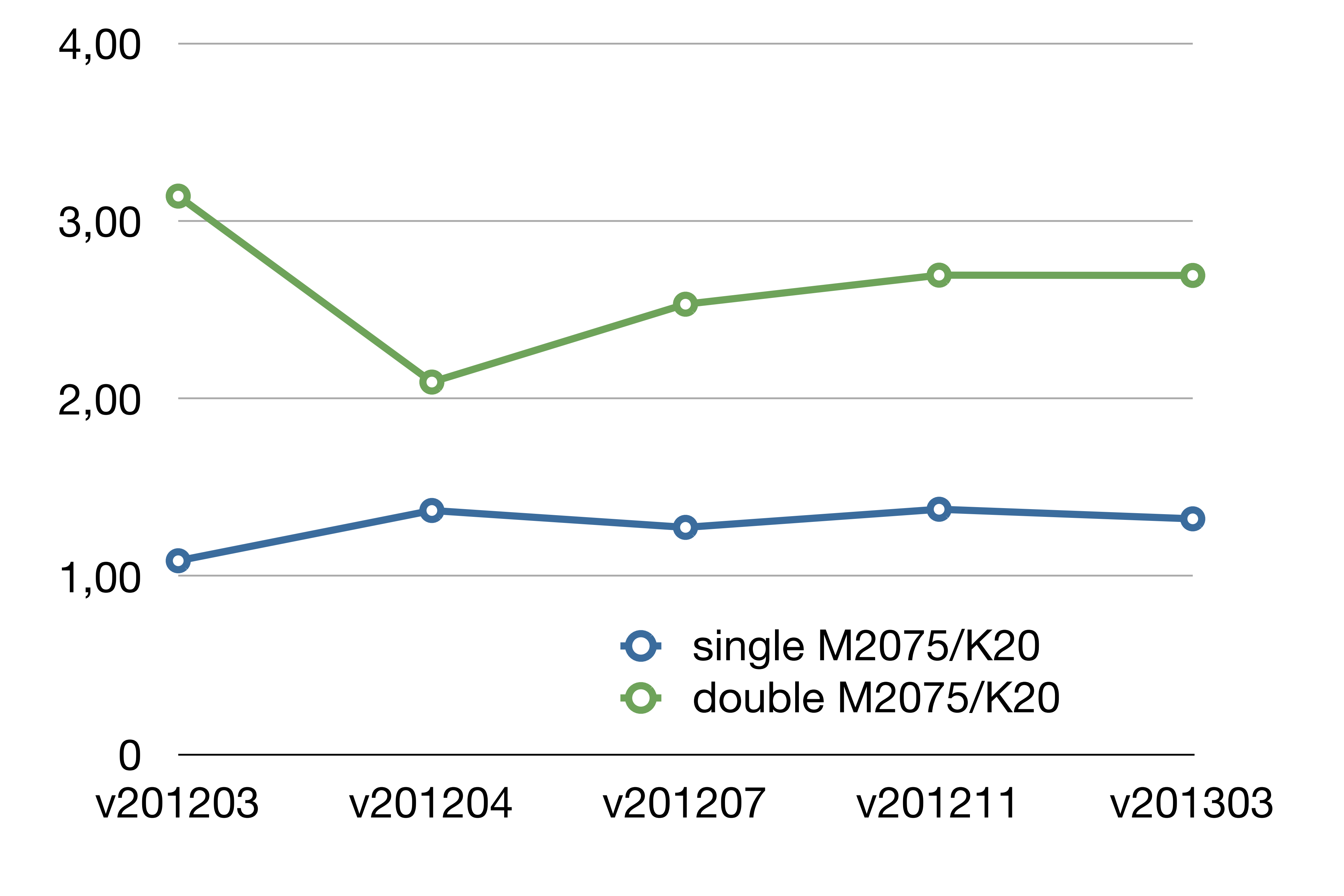


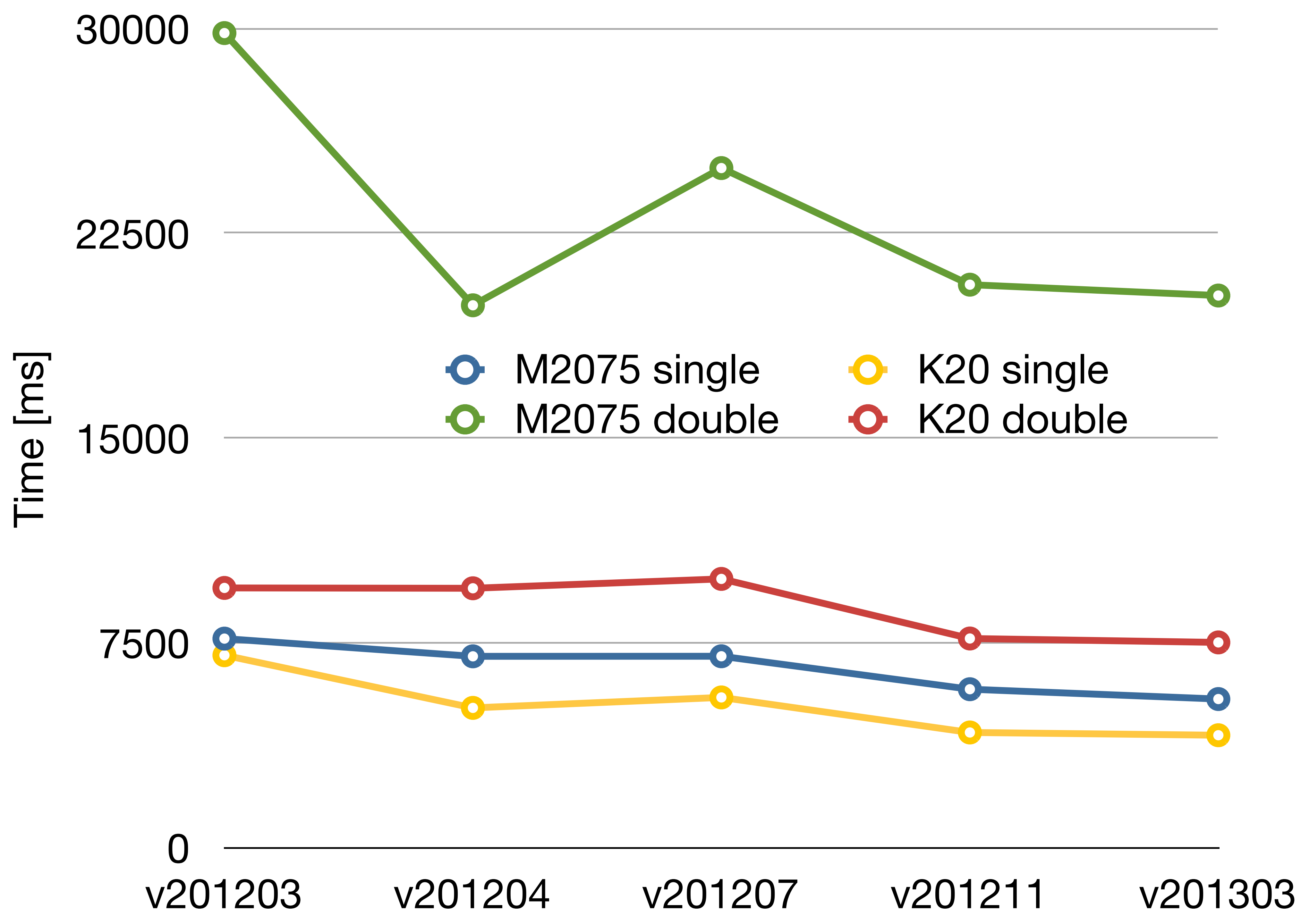


Universität Bielefeld

Optimizing register usage / reduce spilling

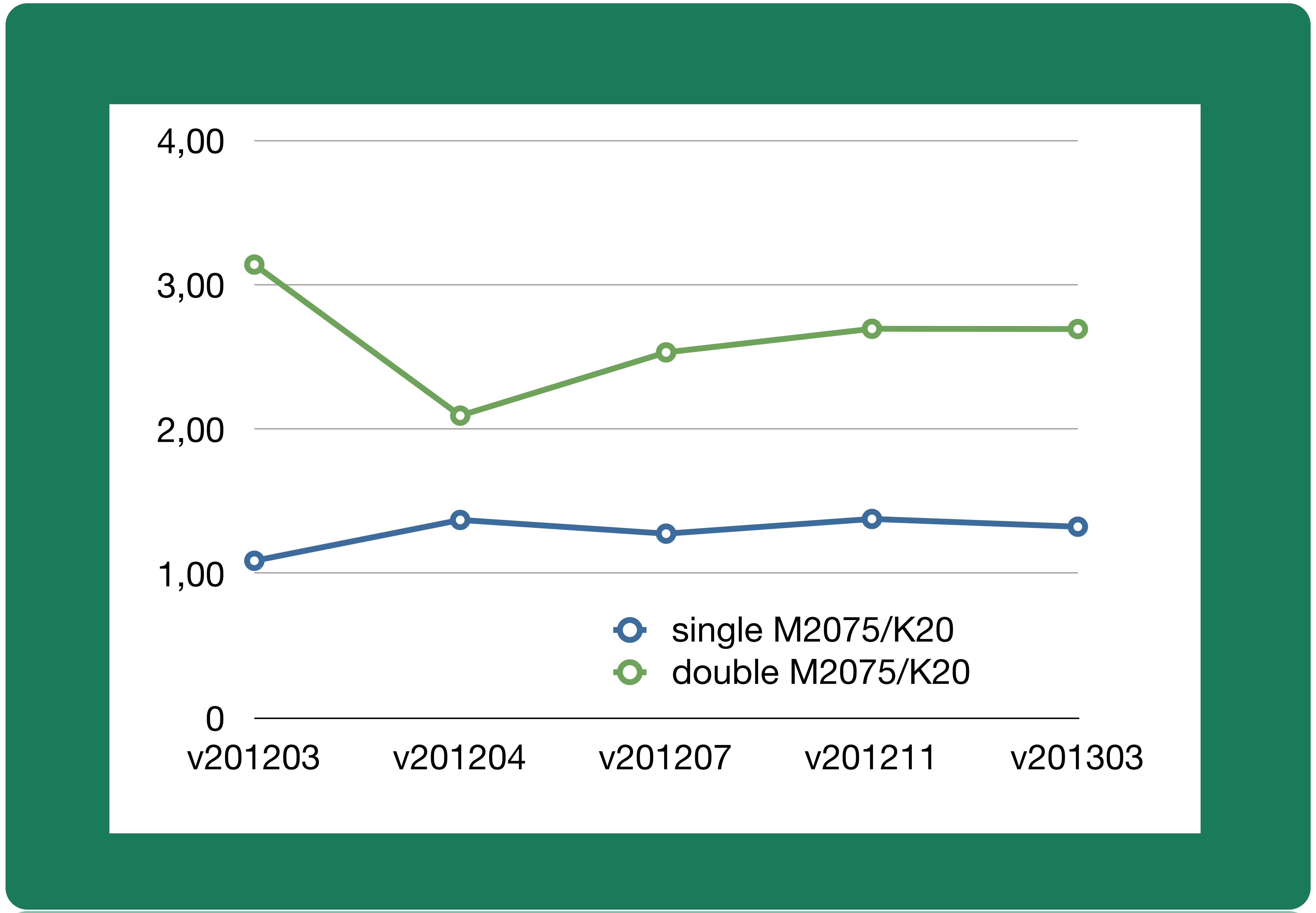
- e.g. force for the 7 link term consists of 56 products of 7 SU(3) matrices (x 24 for ‘rotations’)

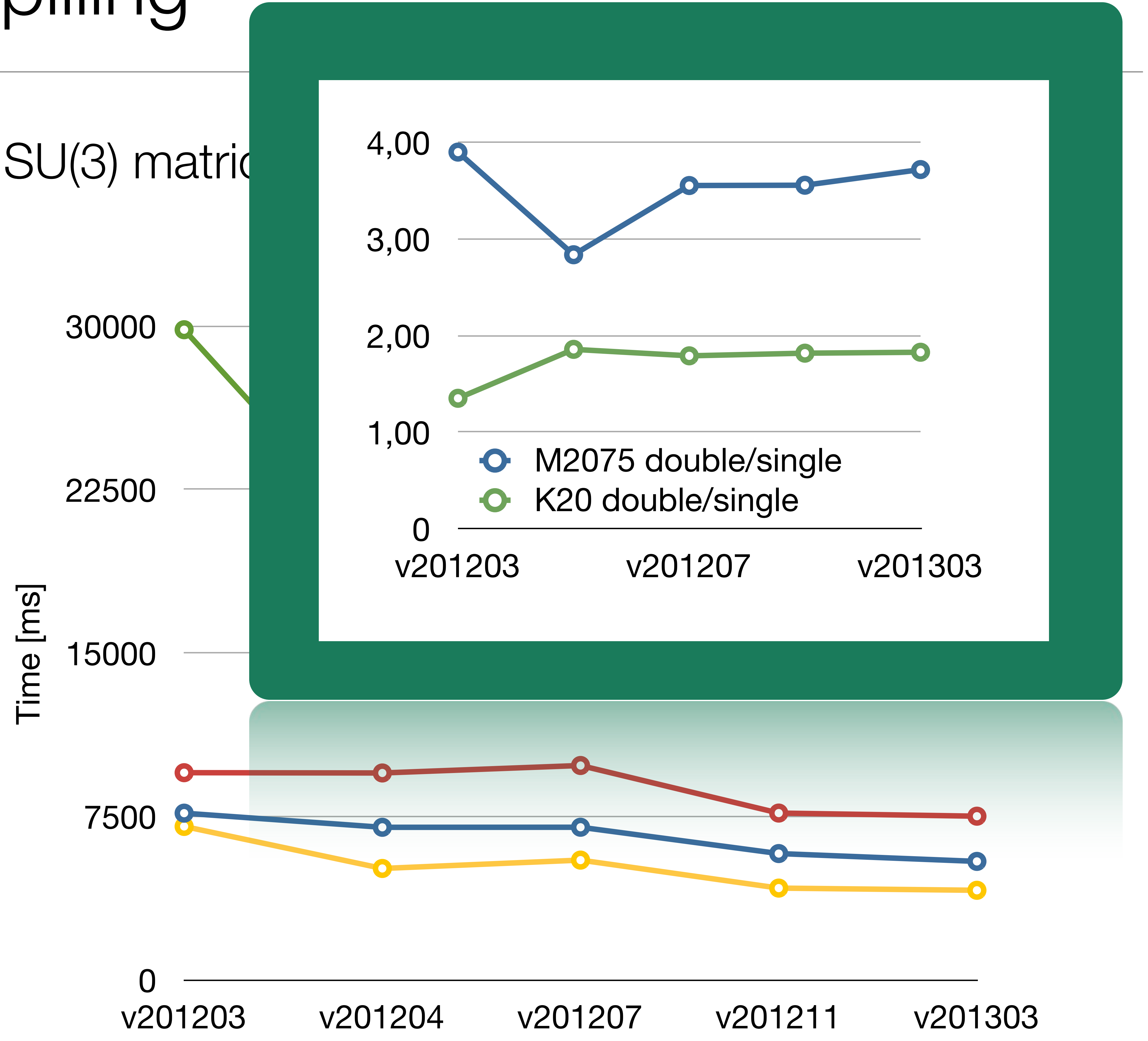




Optimizing register usage / reduce spilling

• e.g. force for the 7 link term consists of 56 products of 7 SU(3) matric

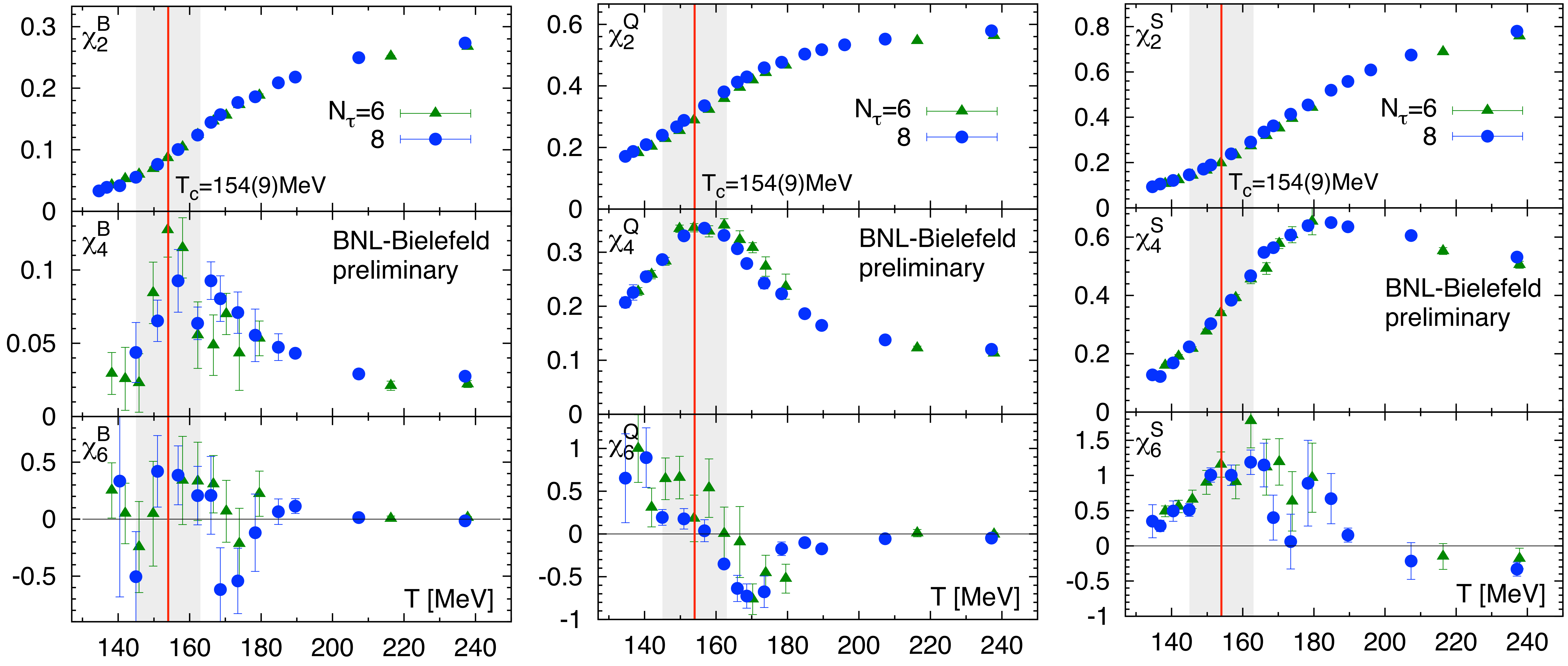




Universität Bielefeld

Status of lattice data

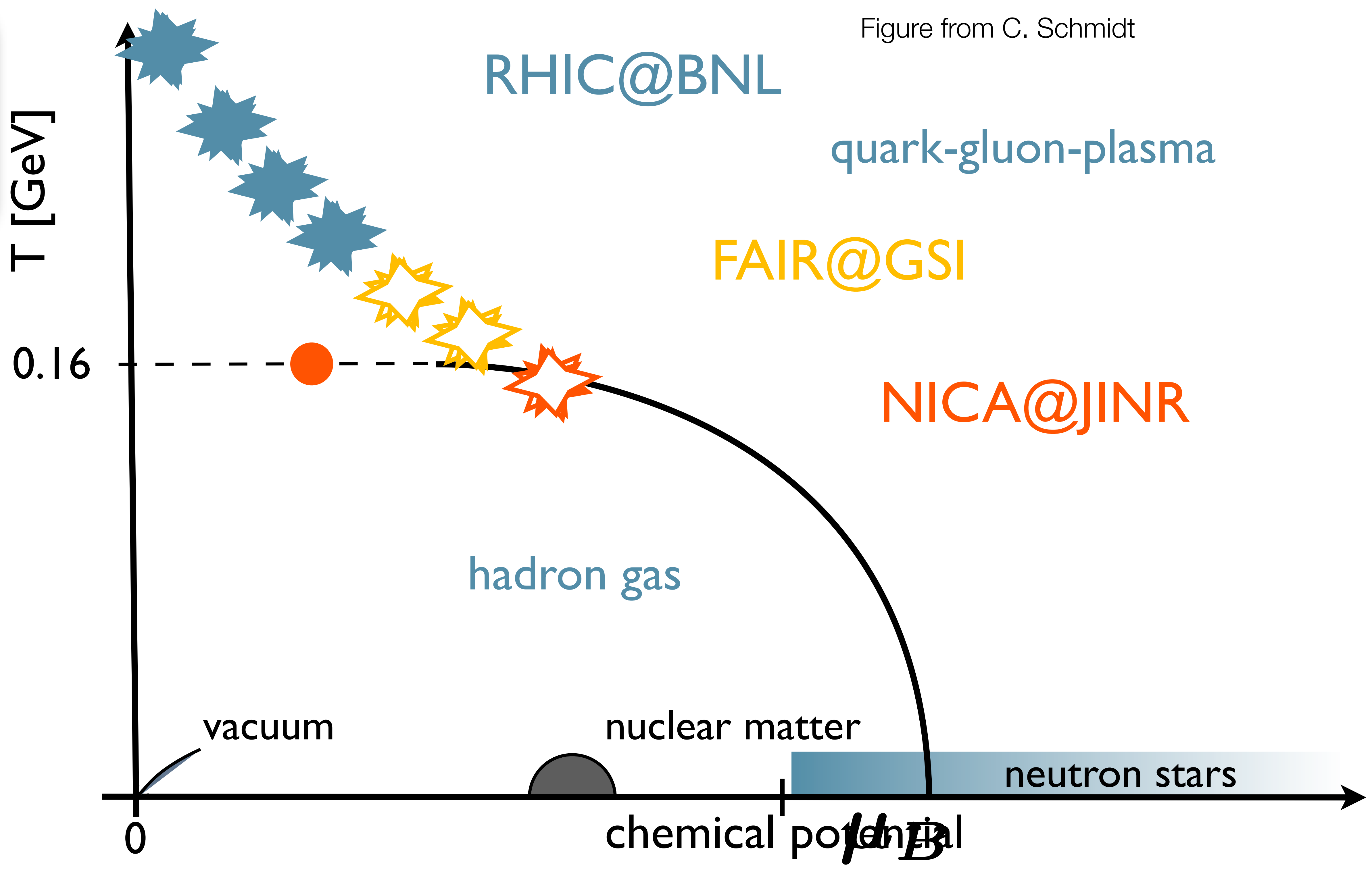
- highly-improved staggered quarks, close to physical pion mass ($m_l/m_s = 1/20$)



Universität Bielefeld

Freeze-out curve from heavy-ion collision

initial conditions



Freeze-out curve from heavy-ion collision

initial conditions

↓

evolution

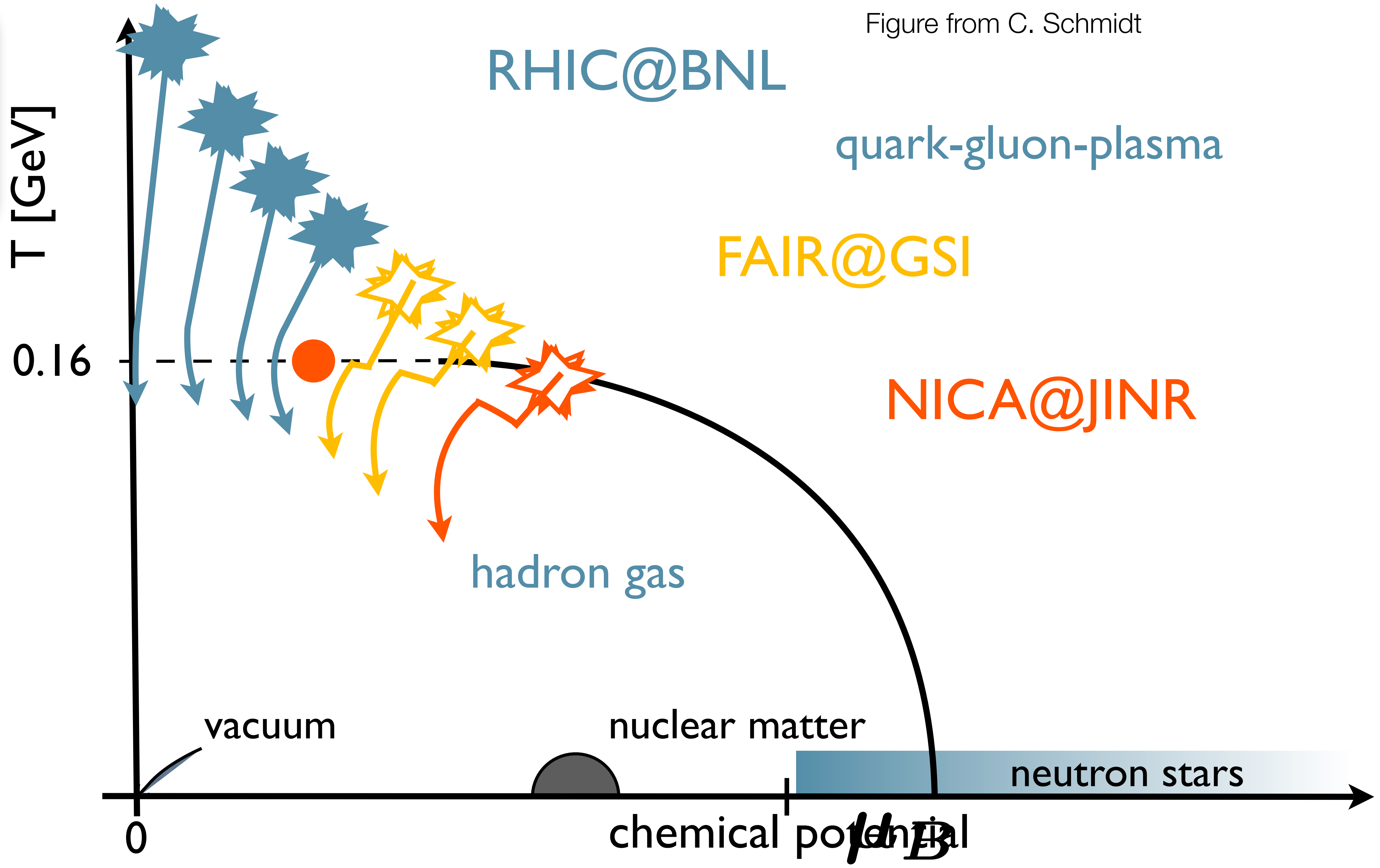
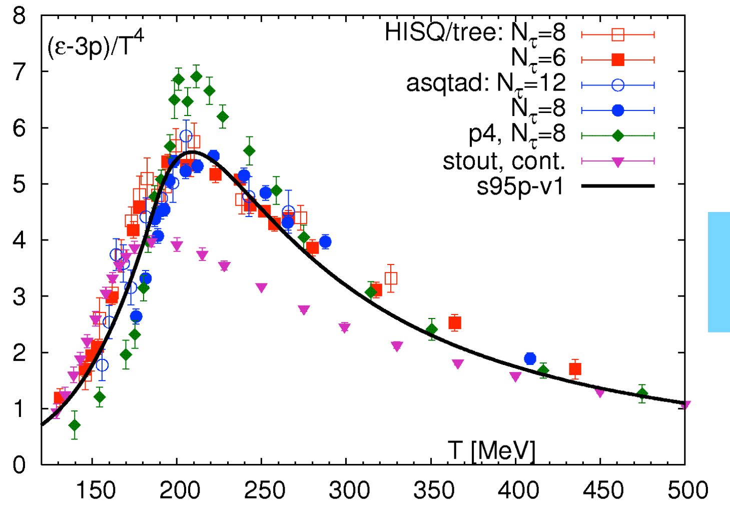


Figure from C. Schmidt

Freeze-out curve from heavy-ion collision

Lattice EoS



initial conditions

evolution

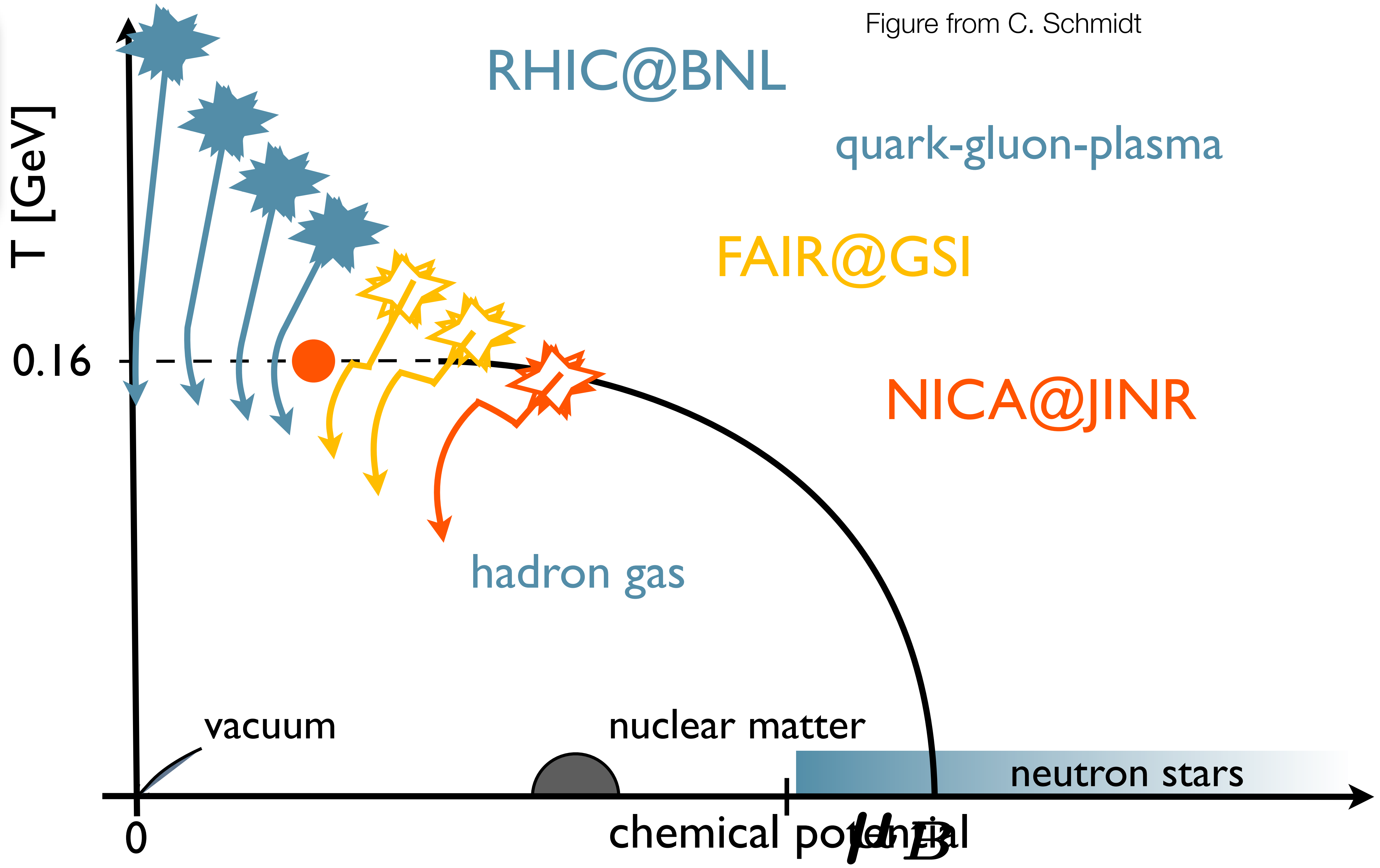
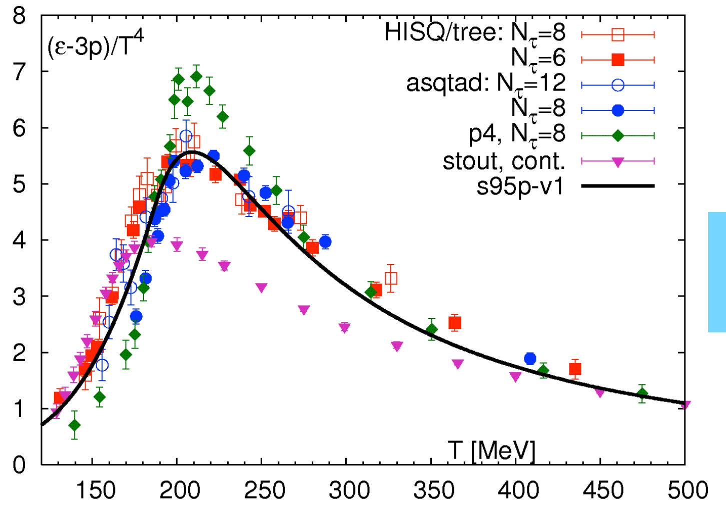
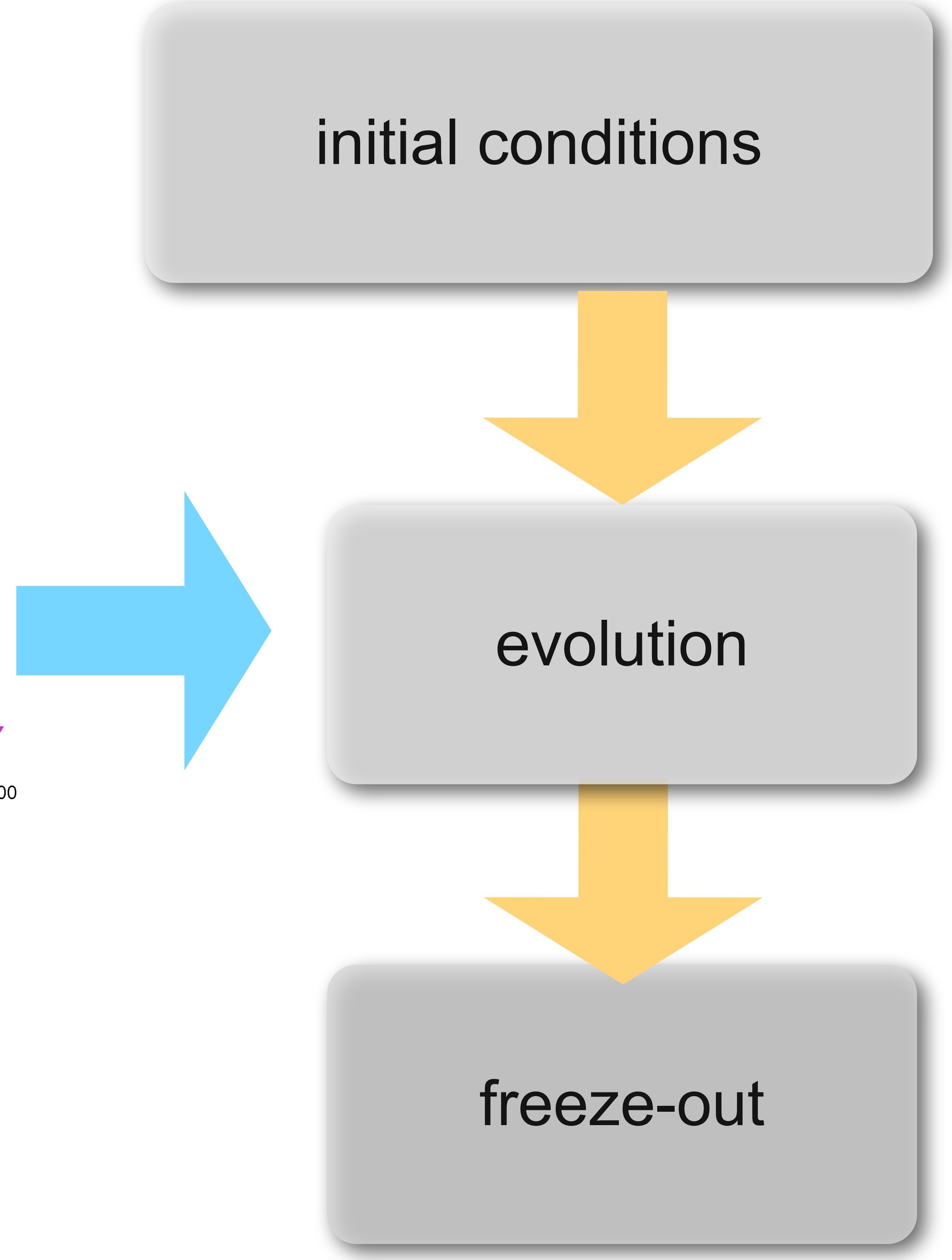


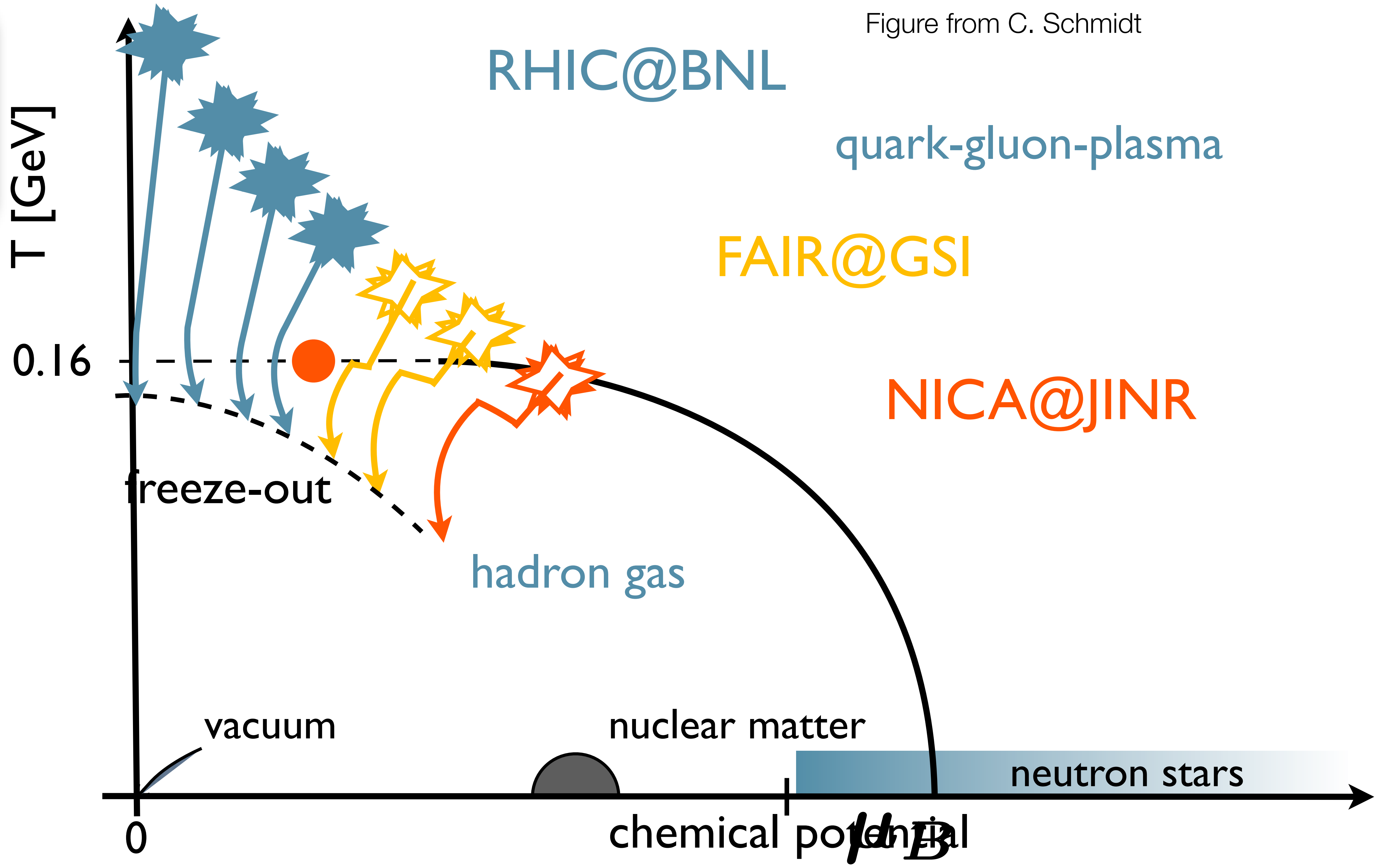
Figure from C. Schmidt

Freeze-out curve from heavy-ion collision

Lattice EoS

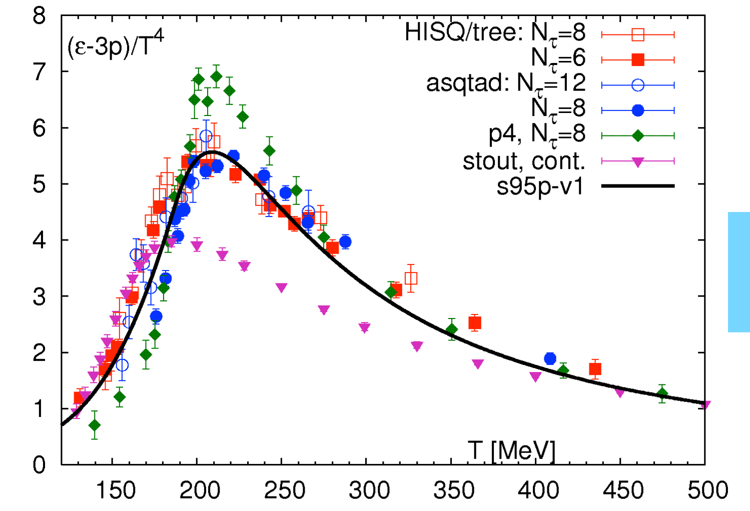




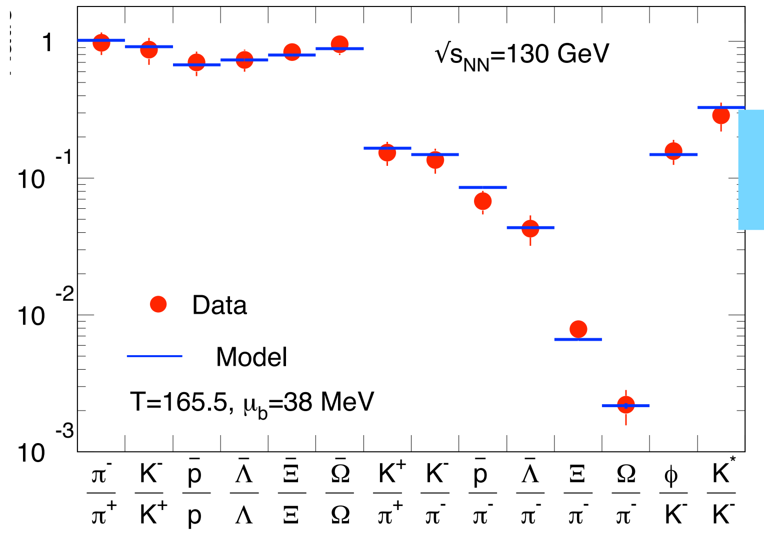


Freeze-out curve from heavy-ion collision

Lattice EoS



Hadron abundances from HRG model



initial conditions

evolution

freeze-out

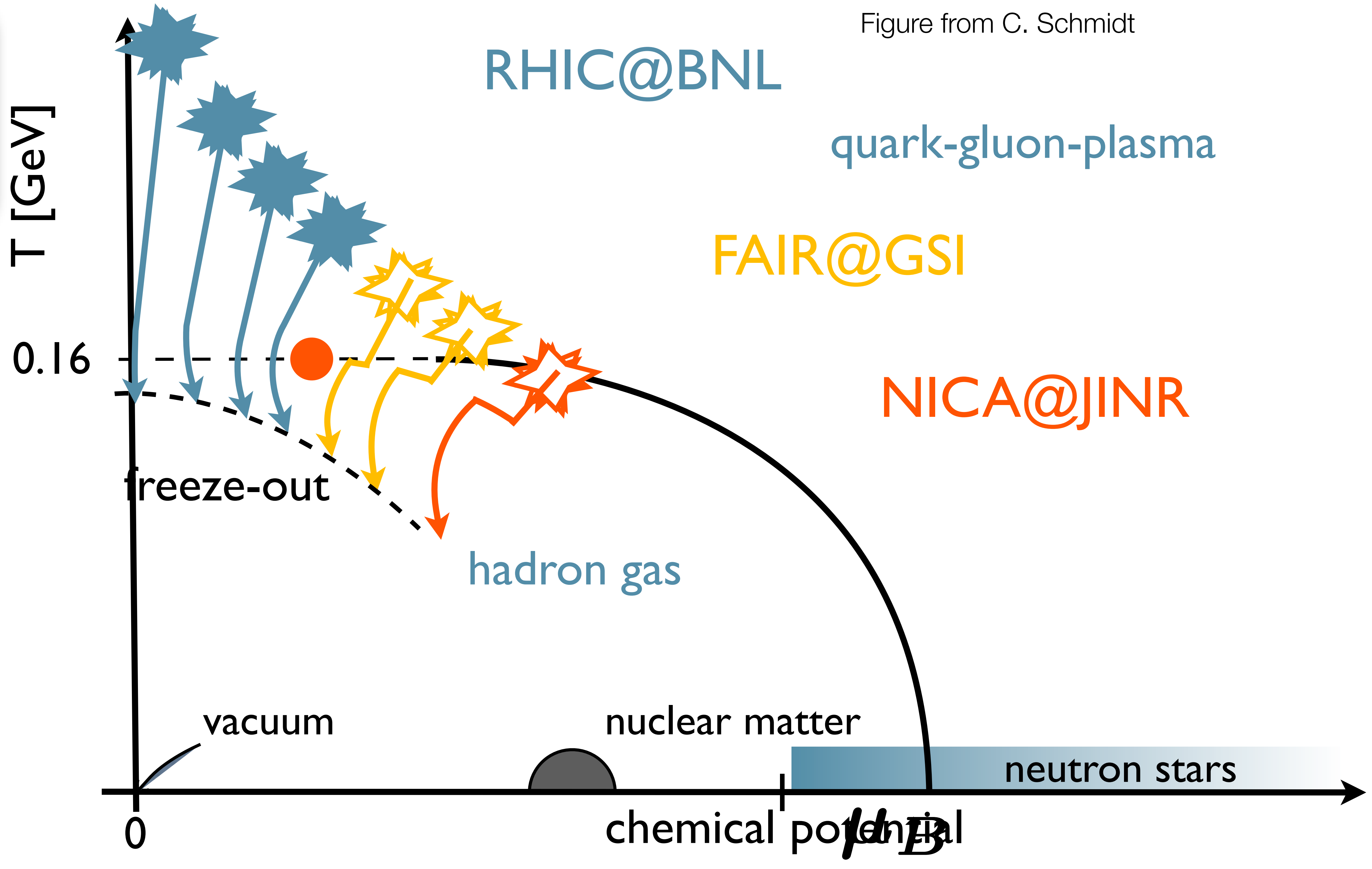
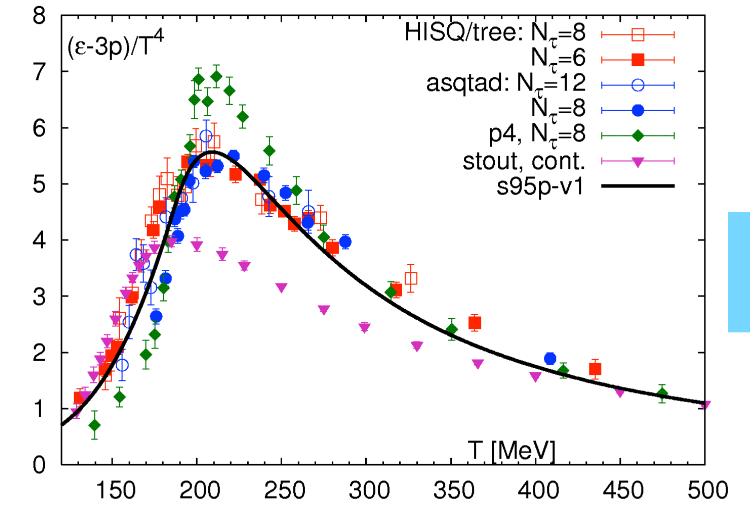


Figure from C. Schmidt

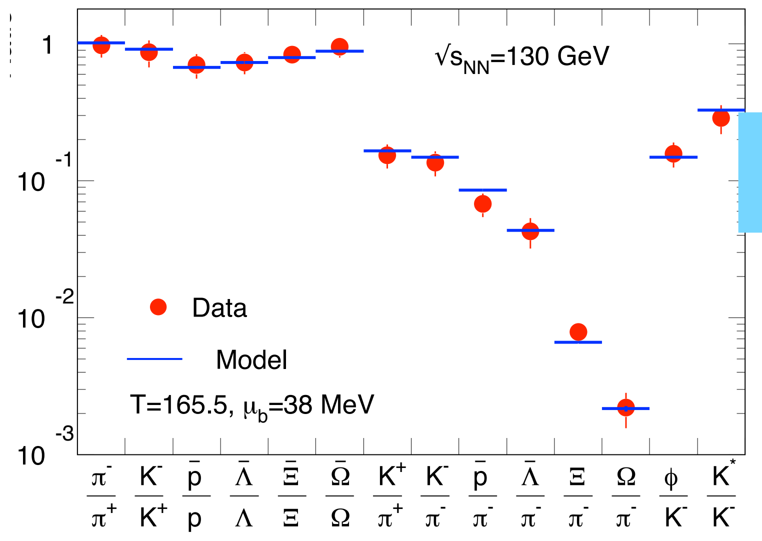
Universität Bielefeld

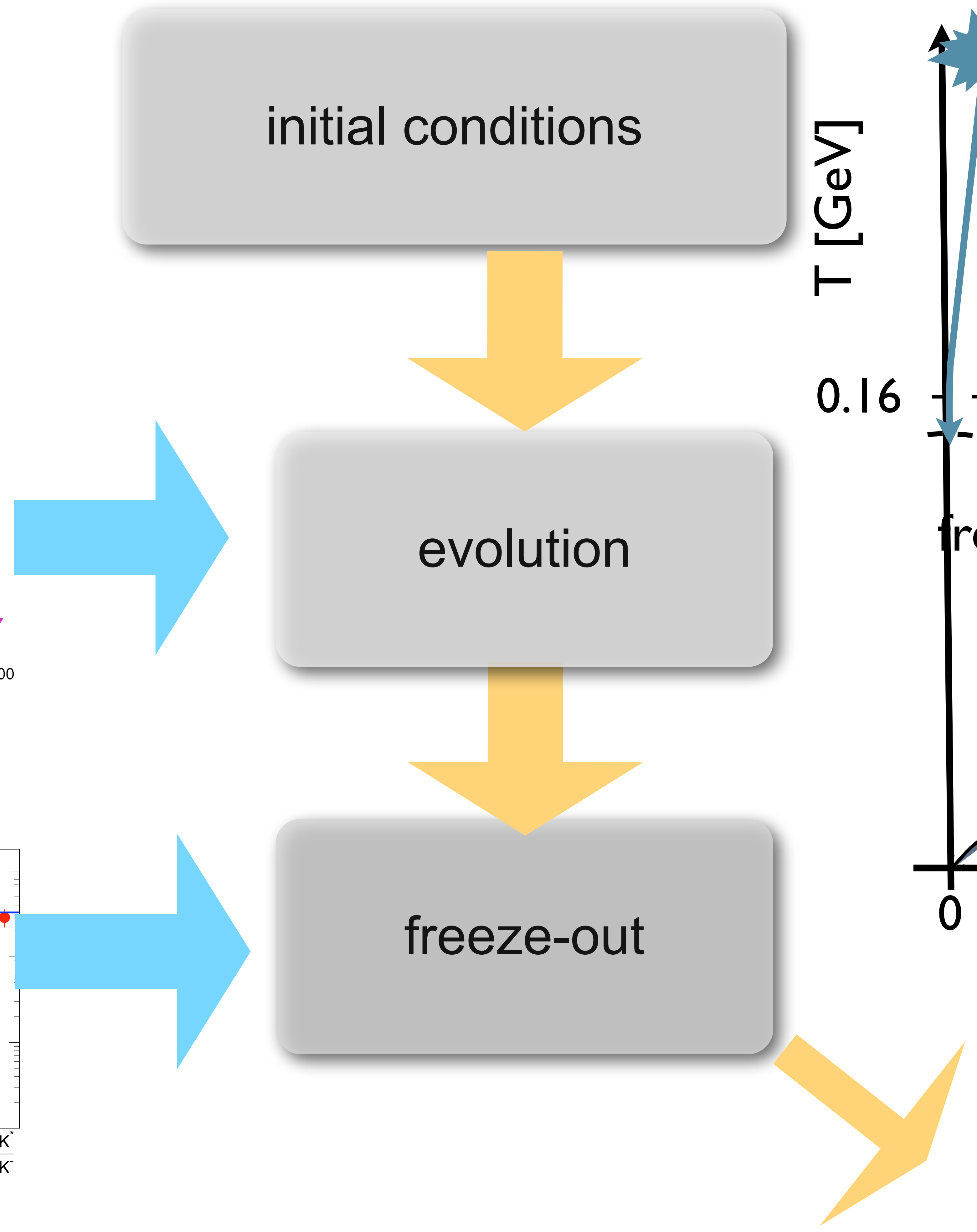
Freeze-out curve from heavy-ion collision

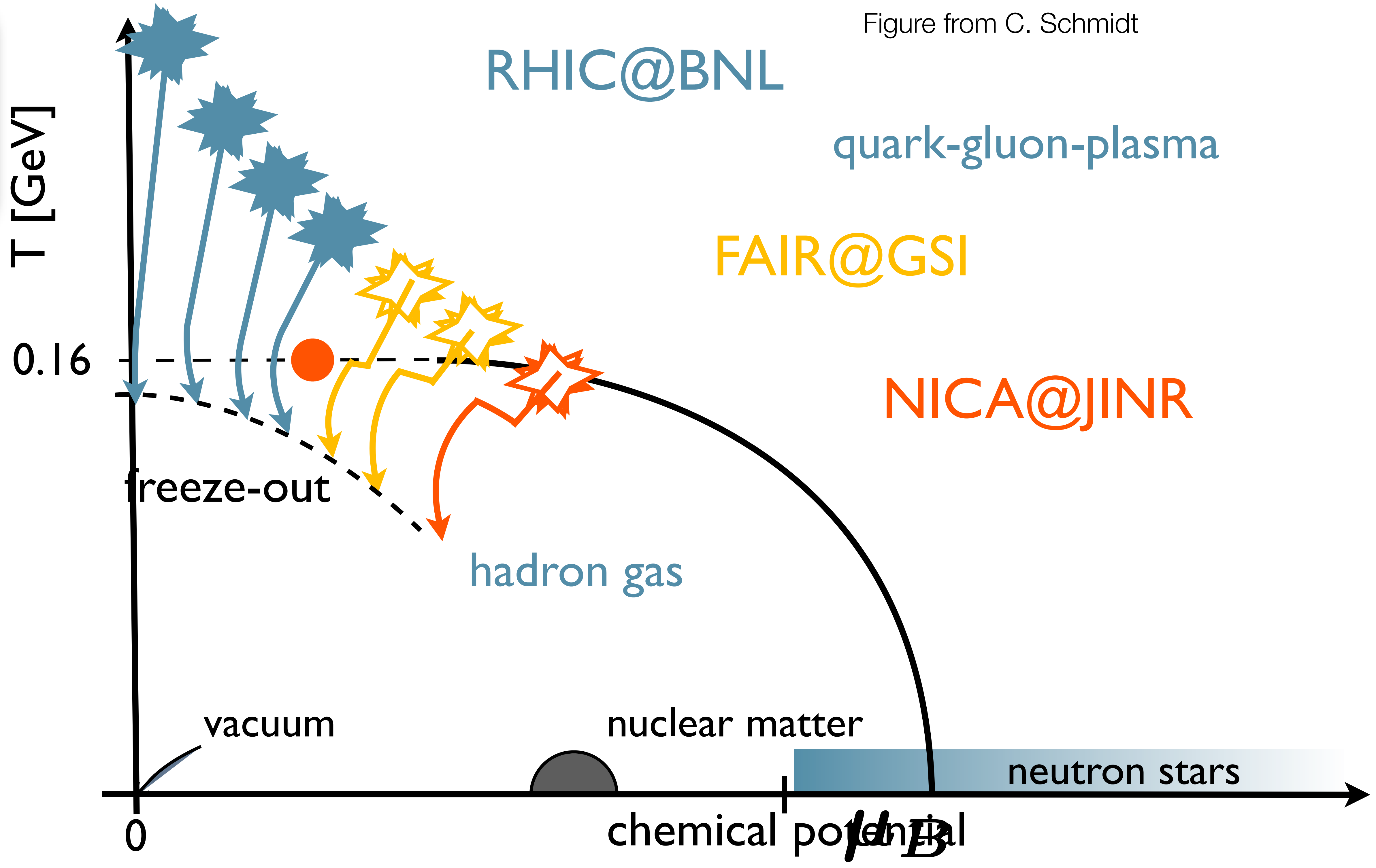
Lattice EoS



Hadron abundances from HRG model



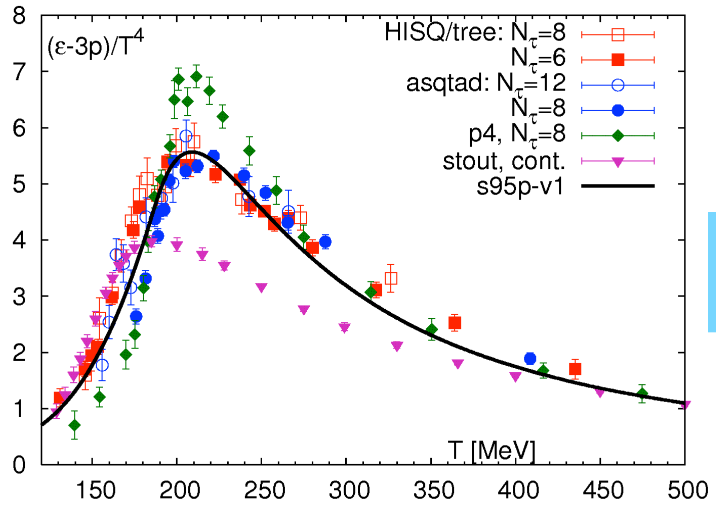




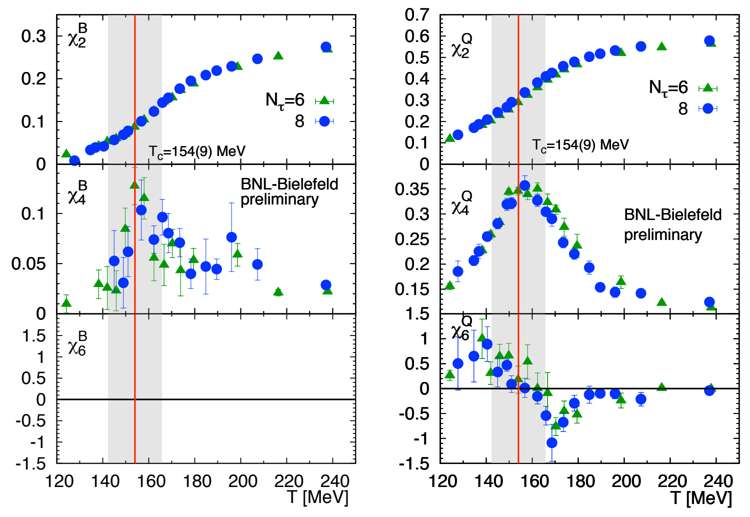
$$T_f(\sqrt{s}), \mu_f(\sqrt{s})$$

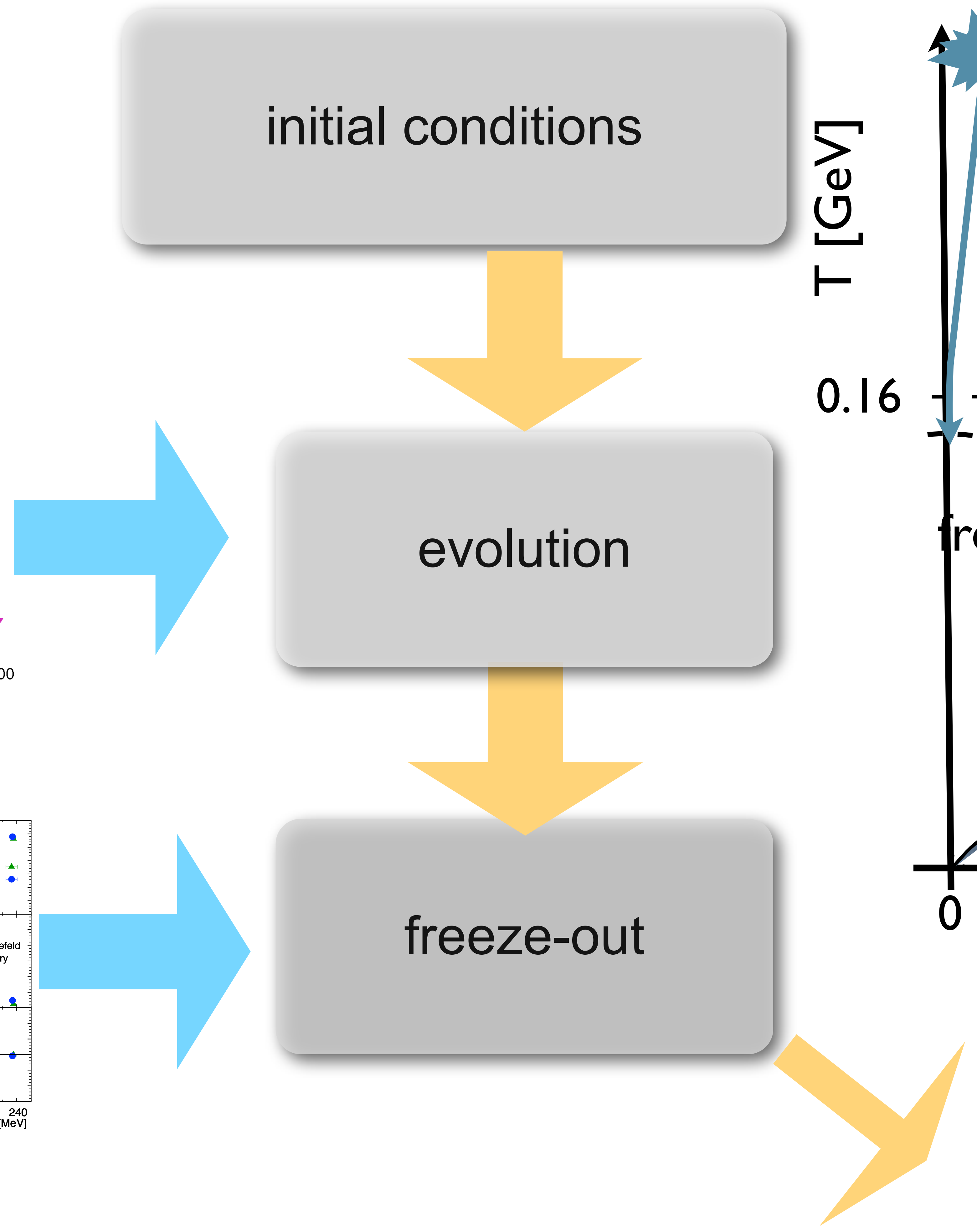
Freeze-out curve from heavy-ion collision

Lattice EoS



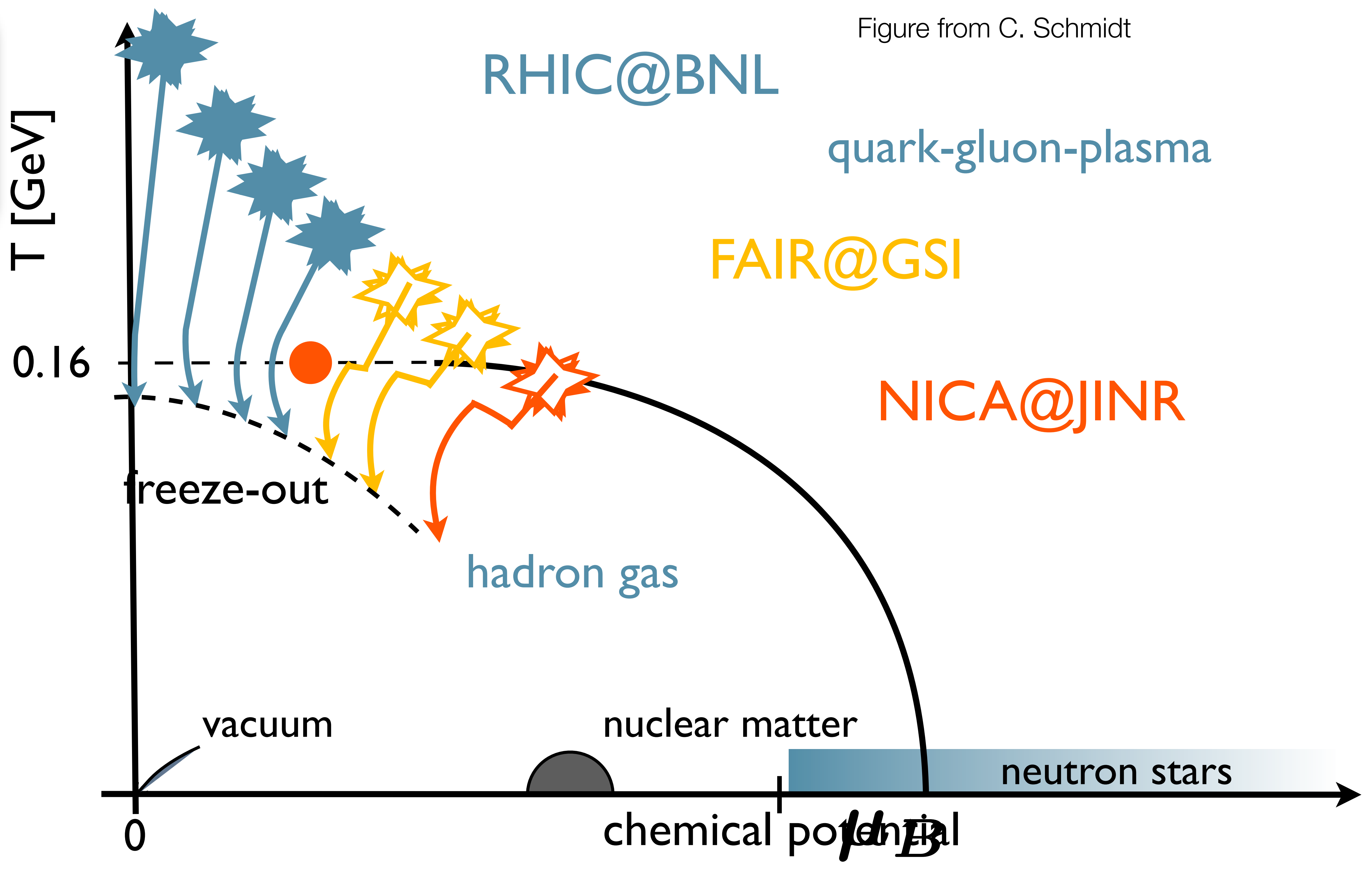
Fluctuations from
Lattice QCD





$T_f(\sqrt{s}), \mu_f(\sqrt{s})$

Figure from C. Schmidt



Universität Bielefeld

Pinning down the freeze-out parameters

- need two experimental ratios to determine (T^f, μ_B^f)
- baryon number fluctuations are not directly accessible in experiments
- we consider ratios of electric charge fluctuations

$$\frac{M_Q(\sqrt{s})}{\sigma_Q^2(\sqrt{s})} = \frac{\langle N_Q\rangle}{\langle(\delta N_Q)^2\rangle} = \frac{\chi_1^Q(T,\mu_B)}{\chi_2^Q(T,\mu_B)} = R_{12}^{Q,1}\hat{\mu}_B + R_{12}^{Q,3}\hat{\mu}_B^3 + \cdots = R_{12}^Q(T,\mu_B)$$

LO linear in μ_B **fixes** μ_B^f

$$\frac{S_Q(\sqrt{s})\sigma_Q^3(\sqrt{s})}{M_Q(\sqrt{s})} = \frac{\langle(\delta N_Q)^3\rangle}{\langle N_Q\rangle} = \frac{\chi_3^Q(T,\mu_B)}{\chi_1^Q(T,\mu_B)} = R_{31}^{Q,0} + R_{31}^{Q,2}\hat{\mu}_B^2 + \cdots = R_{31}^Q(T,\mu_B)$$

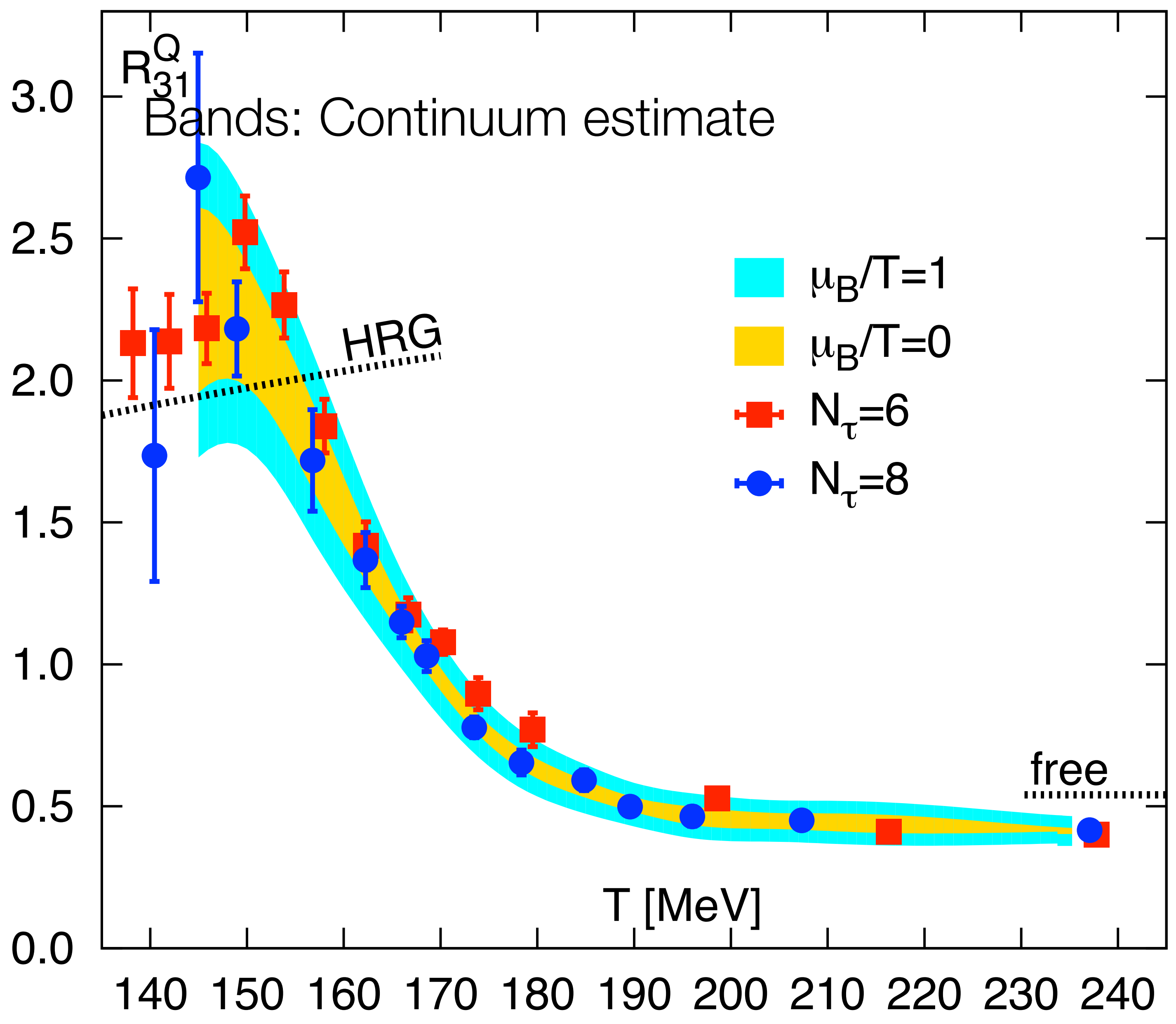
LO independent of μ_B **fixes** T^f

M: mean
σ: variance
S: skewness

Determination of freeze-out temperature

$$R_{31}^{Q}(T,\mu_B) = R_{31}^{Q,0} + R_{31}^{Q,2}\hat{\mu}_B^2$$

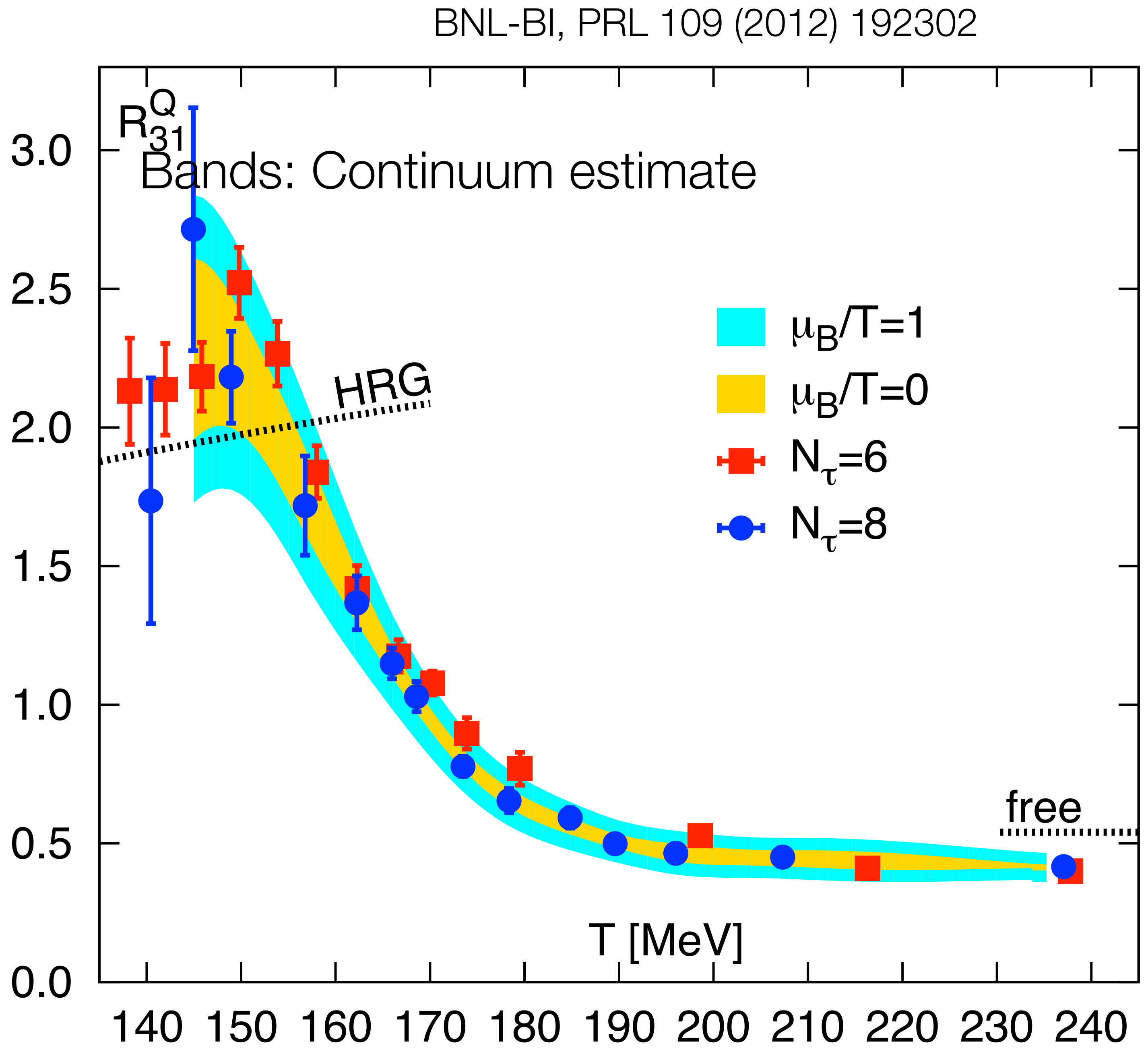
BNL-BI, PRL 109 (2012) 192302



- small cutoff effects
- small NLO corrections (<10%) for $\mu/T < 1.3$

Determination of freeze-out temperature

$$R_{31}^{Q}(T,\mu_B) = R_{31}^{Q,0} + R_{31}^{Q,2}\hat{\mu}_B^2$$



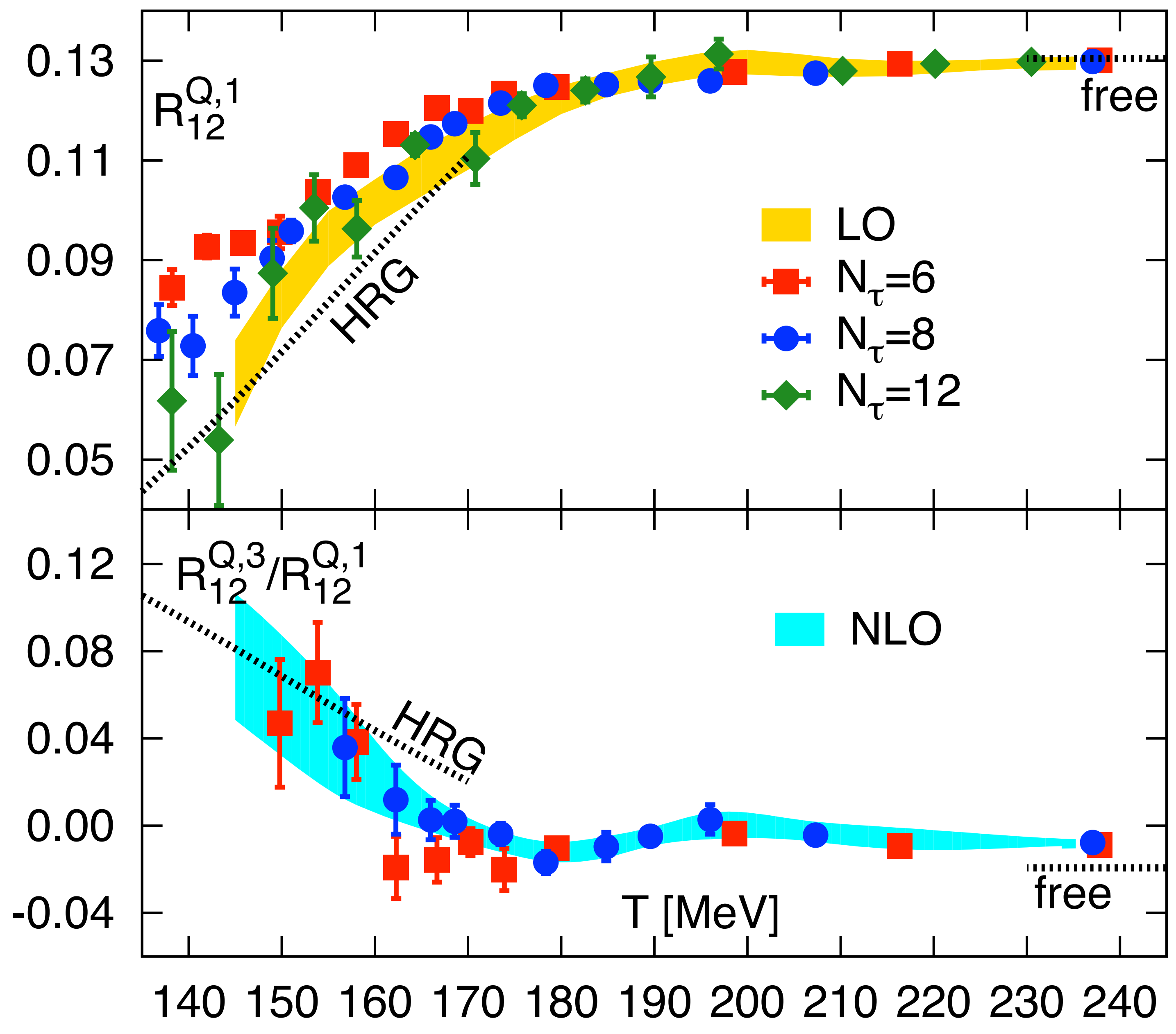
- small cutoff effects
- small NLO corrections (<10%) for $\mu/T < 1.3$

$S_Q\sigma_Q^3/M_Q$	$T^f\,[MeV]$
$\gtrsim 2$	$\lesssim 155$
~ 1.5	~ 160
$\lesssim 1$	$\gtrsim 165$

Determination of freeze-out chemical potential

$$R_{12}^{Q}(T,\mu_B) = R_{12}^{Q,1}\hat{\mu}_B + R_{12}^{Q,3}\hat{\mu}_B^3$$

BNL-BI, PRL 109 (2012) 192302



Bands: LO Continuum extrapolation
NLO Continuum estimate

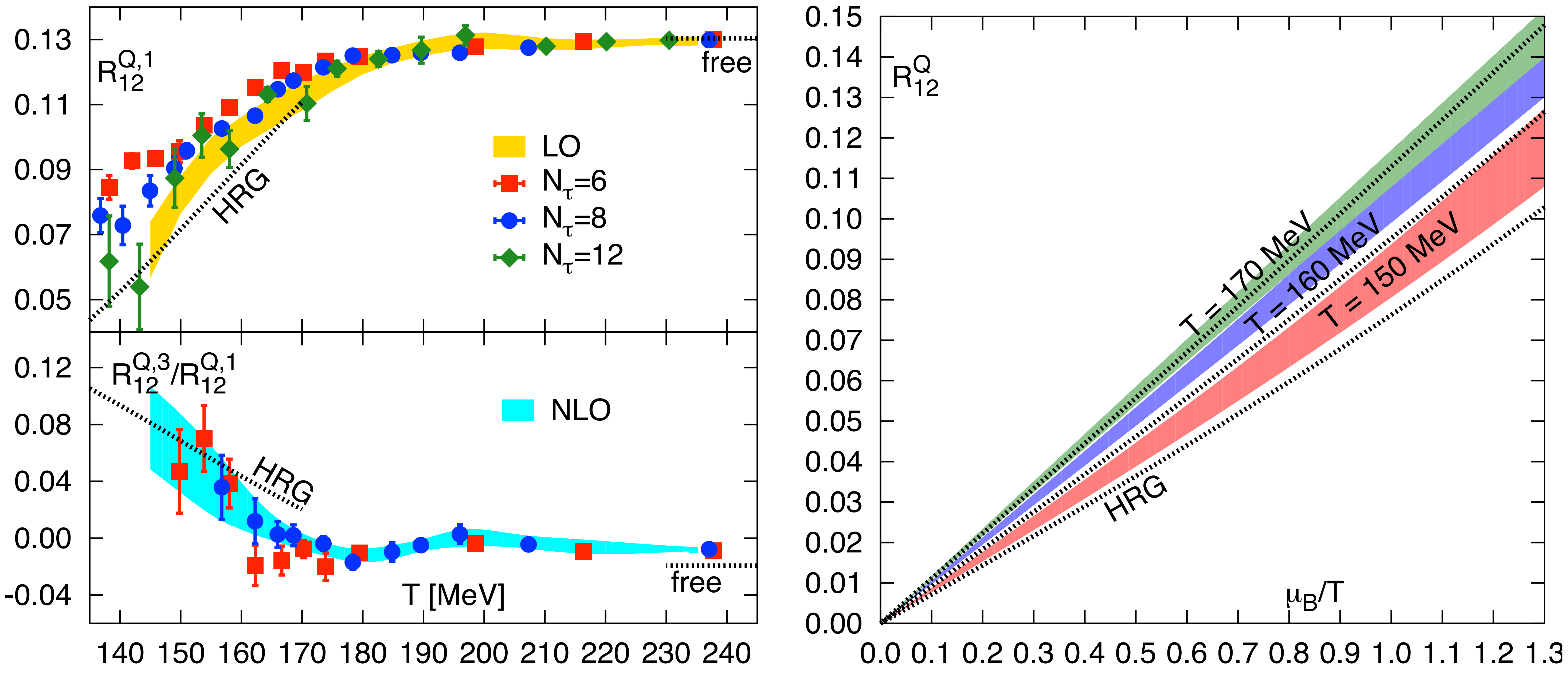
- small cutoff effects at NLO
- small NLO corrections (<10%) for $\mu/T < 1.3$

Universität Bielefeld

Determination of freeze-out chemical potential

$$R_{12}^{Q}(T,\mu_B) = R_{12}^{Q,1}\hat{\mu}_B + R_{12}^{Q,3}\hat{\mu}_B^3$$

BNL-BI, PRL 109 (2012) 192302



Bands: LO Continuum extrapolation
NLO Continuum estimate

- small cutoff effects at NLO
- small NLO corrections (<10%) for $\mu/T < 1.3$

M_Q/σ_Q^2	μ_B^f/T^f
0.01-0.02	0.1-0.2
0.03-0.04	0.3-0.4
0.05-0.08	0.5-0.7

(for $T^f \sim 160\ MeV$)

Universität Bielefeld

Summary

- GPUs enable breakthroughs in Lattice QCD
- Experiences with Lattice QCD on the Bielefeld GPU cluster
- Tuning single GPU performance for staggered fermion
- Lattice QCD is bandwidth bound

Summary

- GPUs enable breakthroughs in Lattice QCD
- Experiences with Lattice QCD on the Bielefeld GPU cluster
- Tuning single GPU performance for staggered fermion
- Lattice QCD is bandwidth bound

- multi-GPU for larger systems
- Kepler provides a major speedup for double precision (thanks to registers)
  - GTX Titan should allow for > 500 GFlops in single precision (>250 GFlops double)
- running production on CPUs and do ‘live-measurements’ on the GPU for Titan

Accelerating Lattice QCD simulations with brain power

- Bielefeld Group
  Edwin Laermann
  Frithjof Karsch
  Olaf Kaczmarek
  Markus Klappenbach
  Mathias Wagner
  Christian Schmidt
  Dominik Smith
  Hiroshi Ono
  Sayantan Sharma
  Marcel Müller
  Thomas Luthe
  Lukas Wresch

- collaborators
  Wolfgang Söldner (Regensburg)
  Piotr Bialas (Krakow)

- Brookhaven Group
  Peter Petreczky
  Swagato Mukherjee
  Alexei Bazavov
  Heng-Tong Ding
  Prasad Hegde
  Yu Maezawa

- supporters
  Mike Clark (Nvidia)

  Matthias Bach (FIAS)

Contact: mwagner@physik.uni-bielefeld.de

Universität Bielefeld