QCD in the Unified European Application Benchmark Suite

Jacob Finkenrath and Giannis Koutsou

CaSToRC, The Cyprus Institute

Outline:

Unified European Application Benchmark Suite: http://www.prace-ri.eu/ueabs/

- PRACE
- UEABS Benchmark Suite: CPU case
  - QCD kernels
  - Results
- UEABS Benchmark Suite: Accelerator based
  - QCD kernels
  - Results
- Conclusion

Here: talk about part of our work in PRACE 4IP-WP7

PRACE: Partnership for Advanced Computing in Europe

Mission of PRACE:

- enable high impact scientific discovery and engineering research and development across all disciplines

→to enhance European competitiveness for the benefit of society

- offering world class computing and data management resources and services through a peer review process

- with a strong interest in improving energy efficiency of computing systems and reducing their environmental impact

- PRACE has 24 members, representing European Union Member States and Associated Countries (+2 observers)

- computer systems and their operations accessible through PRACE

- provided by 5 PRACE members (by Tier 0 Systems: BSC, CINECA, CSCS, GCS and GENCI).

UEABS Benchmark Suite: CPU Case

Unified European Application Benchmark Suite: http://www.prace-ri.eu/ueabs

Idea: Suite for Machine administrator in order to use application codes of end-users

Unified European Application Benchmark Suite (UEABS) is a set of 12 application codes taken from the pre-existing PRACE and DEISA application benchmark suites

Objective: providing a set of scalable, currently relevant and publically available codes which can realistically be run on large systems

Setup by Task 7.4 “Unified European Applications Benchmark Suite for Tier-0 and Tier-1” in the PRACE Second Implementation Phase (PRACE-2IP) project

Here we will present the update and maintainace in the Fourth Implementation Phase

Application codes:

- ALYA
- Code_Saturne
- CP2K
- GADGET
- GENE
- GPAW
- GROMACS
- NAMD
- NEMO
- **QCD**
- Quantum Espresso
- SPECFEM3D.

UEABS Benchmark Suite: QCD

Five kernels, representative of code-base of European lattice QCD users

- **Kernel A:** From BQCD (Berlin QCD). Software suit of QCDSF collaboration.
- **Kernel B:** From an application to Higgs physics. 3D volume instead of 4D.
- **Kernel C:** From DD-HMC (OpenQCD). Solver algorithm which includes multiple sub block-updates to precondition global problem. Software suit of ALPHA collaboration/CLS
- **Kernel D:** tmLQCD (twisted mass). Software suit of European Twisted Mass collaboration
- **Kernel E:** Conjugate Gradient solver. Contributed by members of the BMW collaboration

UEABS Benchmark Suite QCD case: HowTo

Jube environment
managed by XML-files
Source files:
/applications/QCD/src

Starting on a new platform:
several xml-files has to edit
→ for compilation and submitting

Platform files:
/platform/platform.xml
/platform/NEWPLATFORM/*

Execution files:
/applications/*
(execute.xml, scaling.xml, compile.xml)

Submition:
perl ../../bench/jube scaling.xml

Example: SUPERMUC @ LRZ Munich
execute.xml:

```
<execute cname="supermuc">
   <input files="$pdir/supermuc/ibm_llsubmit.job.in" />
   <substitute infile="ibm_llsubmit.job.in"
outfile="ibm_llsubmit.job">
      <sub from="#OUTDIR#"          to="$outdir" />
      <sub from="#STDOUTLOGFILE#"   to="$stdoutlogfile" />
…
```

platform.xml:

```
platforms>
   <platform name="supermuc">
      <params
        make          = "make"
        rm            = "rm -f"
        …
```

More infos in the README-files

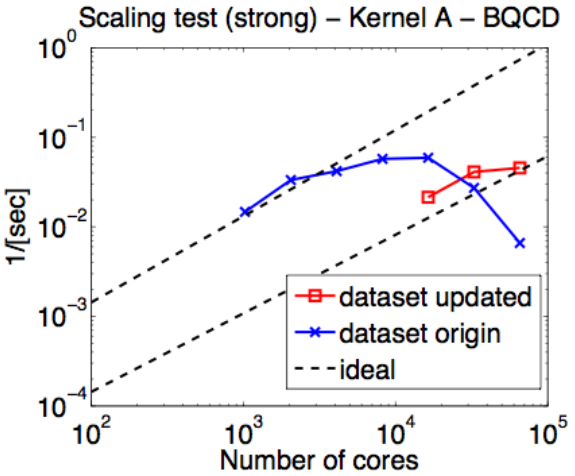
- Sandy Bridge Partition on SuperMUC: Intel Haswell-EP processor
- 16 procs per node

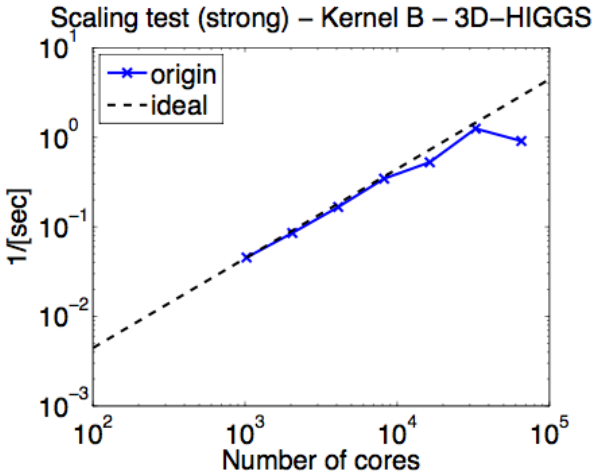
UEABS QCD: Results

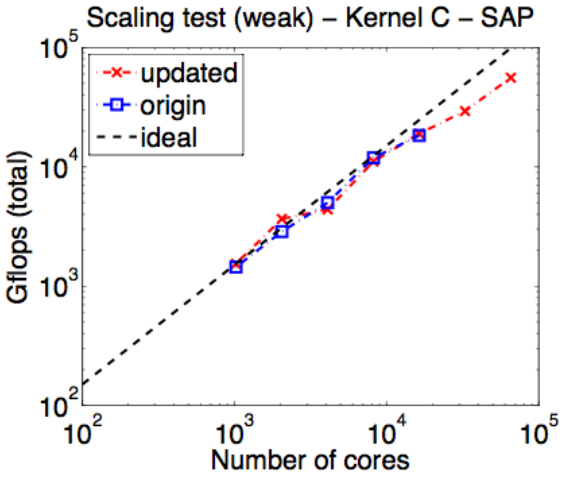
Kernel A (BQCD)

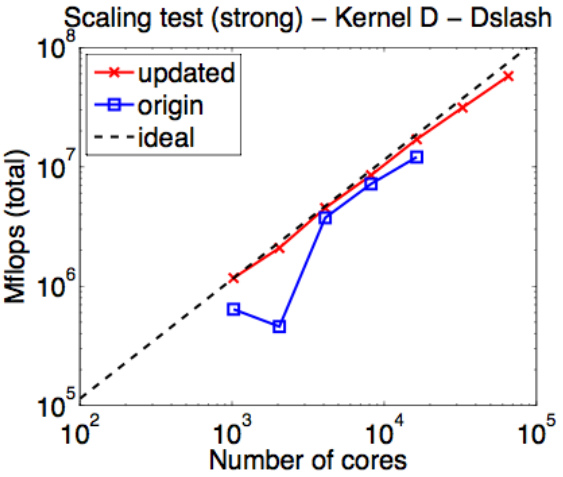
- L=32^2x64^2 → L=128x64^3

- Kernel B (SU3-3dim)
  - V=256^3
- Kernel C (openQCD)
  - (updated to openQCD)
  - Local V=8^4
- Kernel D (tmLQCD)
  - Updated to a newer version
  - V = 64^4



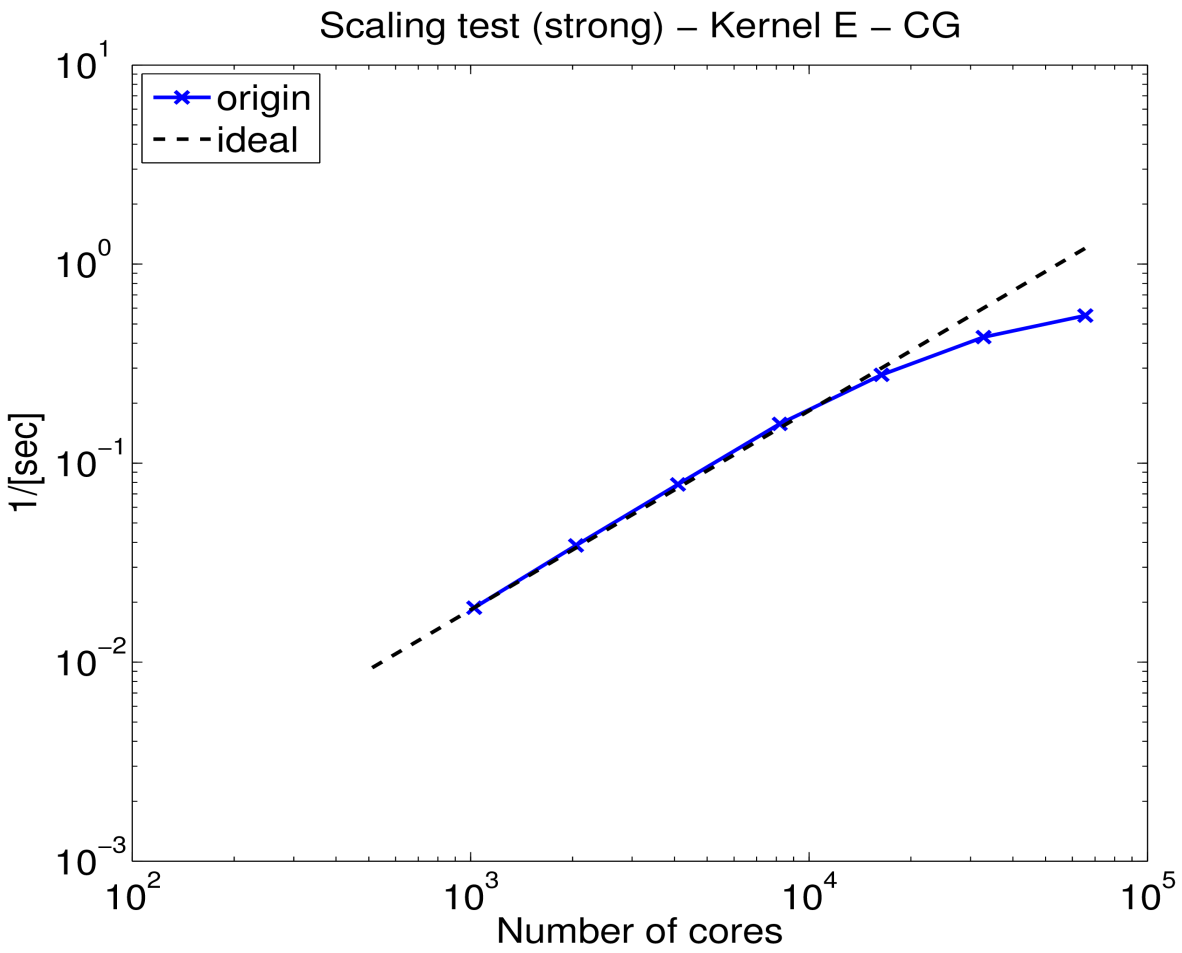






UEABS QCD: Results

- Kernel E:
  (based on Code of members of the BMWc)
- CG with V=64^4



- Scales up to 65 000 cores on Sandy Bridge Partition on SuperMUC if lattice size is adopted, like 64^3x128
  - 65 000 = 16 procs per 4096 nodes
  - Scaling from 32 nodes to 4096 nodes

Accelerator Benchmarks (Part 2)

GPUs: QUDA[1]

- Implements common lattice QCD kernels in CUDA
- Mostly an offload model approach, almost everything happens in the GPU
- Parallelized in all four directions
- Actively community developed. Main developers at NVIDIA
- https://github.com/lattice/quda

Xeon Phi: QPhiX[2]

- Similar kernels to QUDA, but optimised for wide SIMD
- Can also be compiled for various vector widths (e.g. SSE,AVX)
- Parallelised for multi-node Xeon Phi clusters
- https://github.com/JeffersonLab/qphix

- Accelerator Benchmarks Part 1:
  - Based on Kernel E of UEABS Benchmark Suite
  - Done by Alan Gray

1. R. Babbich, M. Clark and B. Joo, “Parallelizing the QUDA Library for Multi-GPU Calculations in Lattice Quantum Chromodynamics” SC 10 (Supercomputing 2010)
2. B. Joo, D. D. Kalamkar, K. Vaidyanathan, M. Smelyanskiy, K. Pamnany, V. W. Lee, P. Dubey, W. Watson III, “Lattice QCD on Intel Xeon Phi”, International Supercomputing Conference (ISC’13), 2013

Accelerator Benchmarks (Part 2): Setup

Setup:
provide bash-scripts for different test-cases

Download kernel from developer pages:

QUDA: https://github.com/lattice/quda
QPhiX: https://github.com/JeffersonLab/qphix

Configure and Compile:
→ depends on the platform
(see for instruction in the README)

Edit bash-scripts:
- run_sca.sh
- submit_job.sh.template

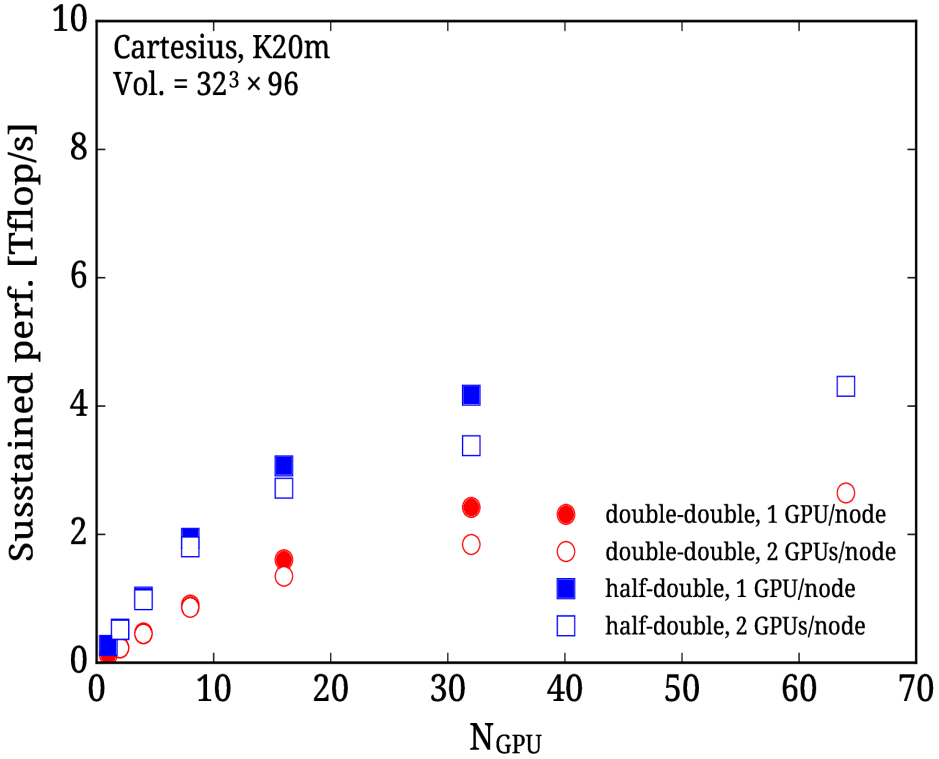
Example:
run_sca.sh

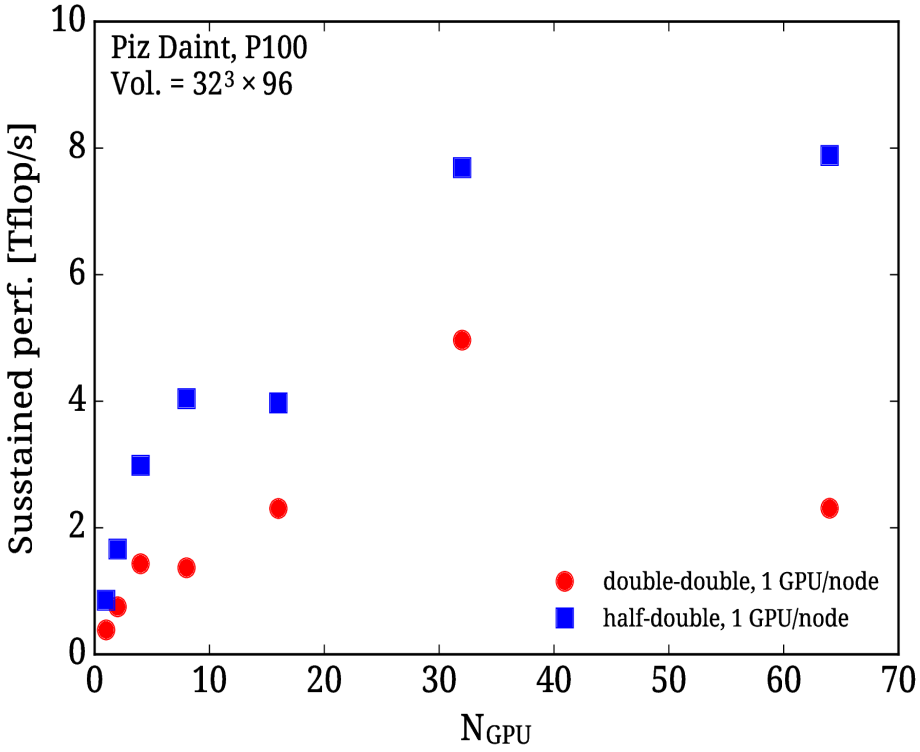
```
#!/bin/bash
EXE=$PATH2EXE
## Set scaling-mode: Strong or Weak
sca_mode="Strong"
## sca_mode="Weak"
## lattice size (size strong 1)
gx=32
gt=96
...
```

submit_job.sh.template: (KNL case)

```
#!/bin/bash
#SBATCH --cpus-per-task=68
#SBATCH --ntasks=#NODES#
#SBATCH --exclusive
#SBATCH --mem=150GB
export OMP_NUM_THREADS=68
....
```

Strong Scaling: GPU Benchmark

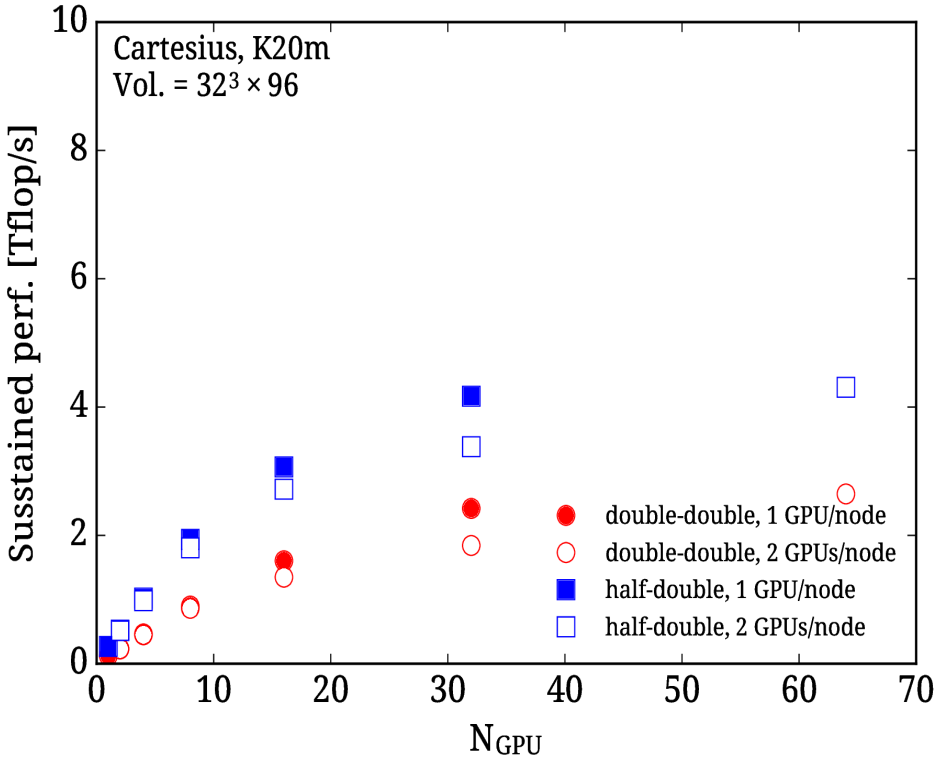


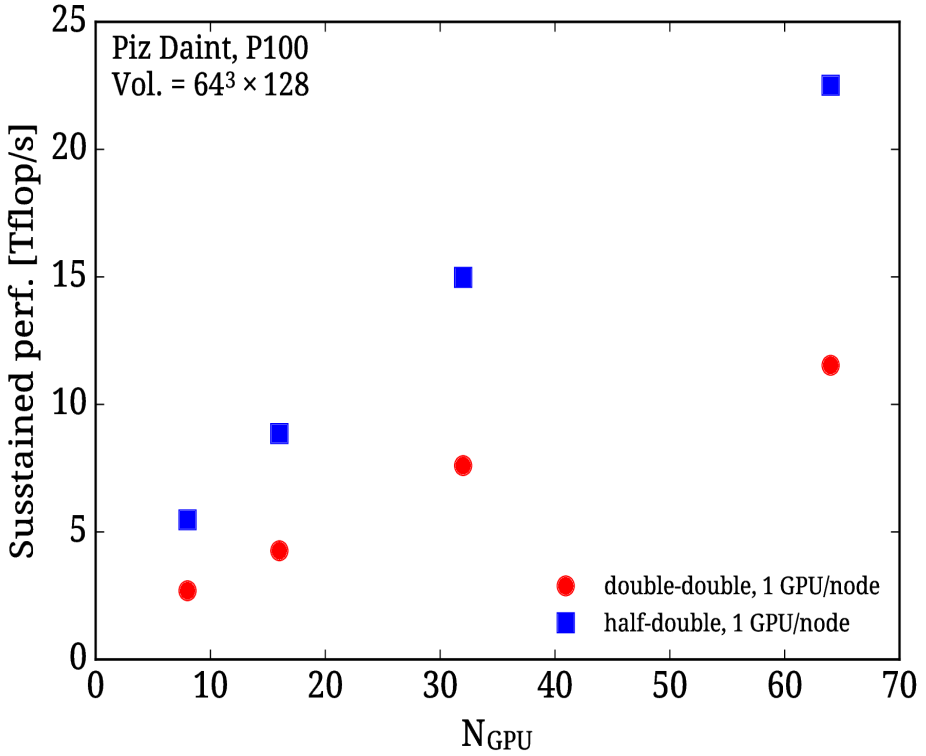


- Cartesius (SURFsara) – Kepler GPUs
- Piz Daint (CSCS) – Pascal GPUs

- Double, single or half precision solve as preconditioner
- Pascal more efficient at larger problem sizes

Strong Scaling: GPU Benchmark

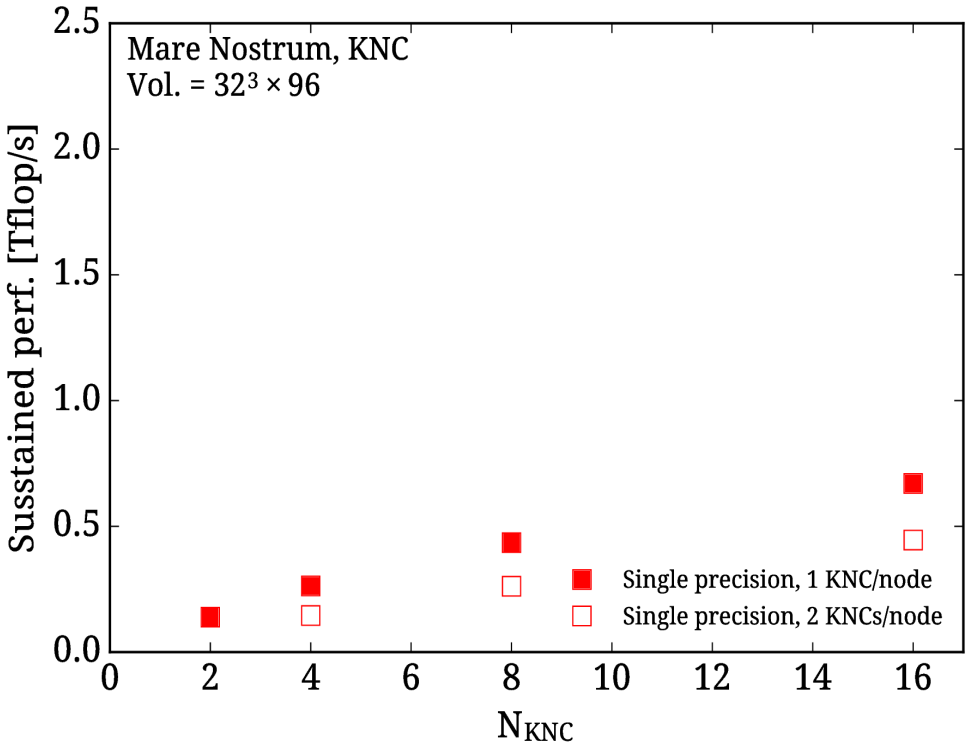


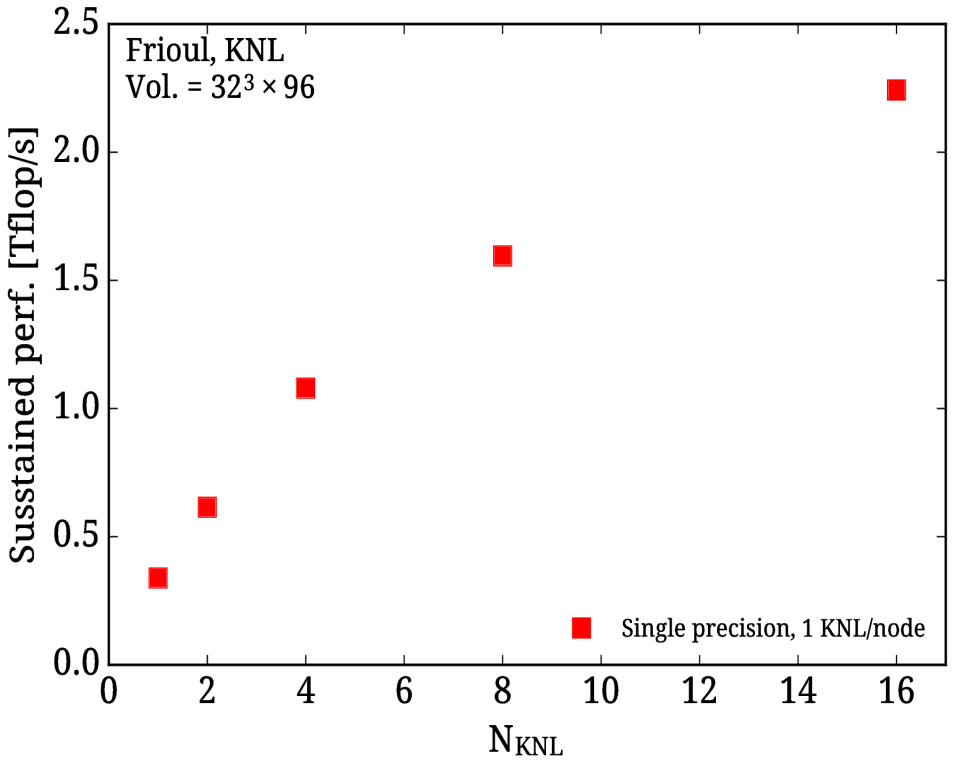


- Cartesius(SurfSARA) – Kepler GPUs
- Piz Daint (CSCS) – Pascal GPUs

- Double, single or half precision solve as preconditioner
- Pascal more efficient at larger problem sizes

Strong Scaling: Xeon Phi Benchmark

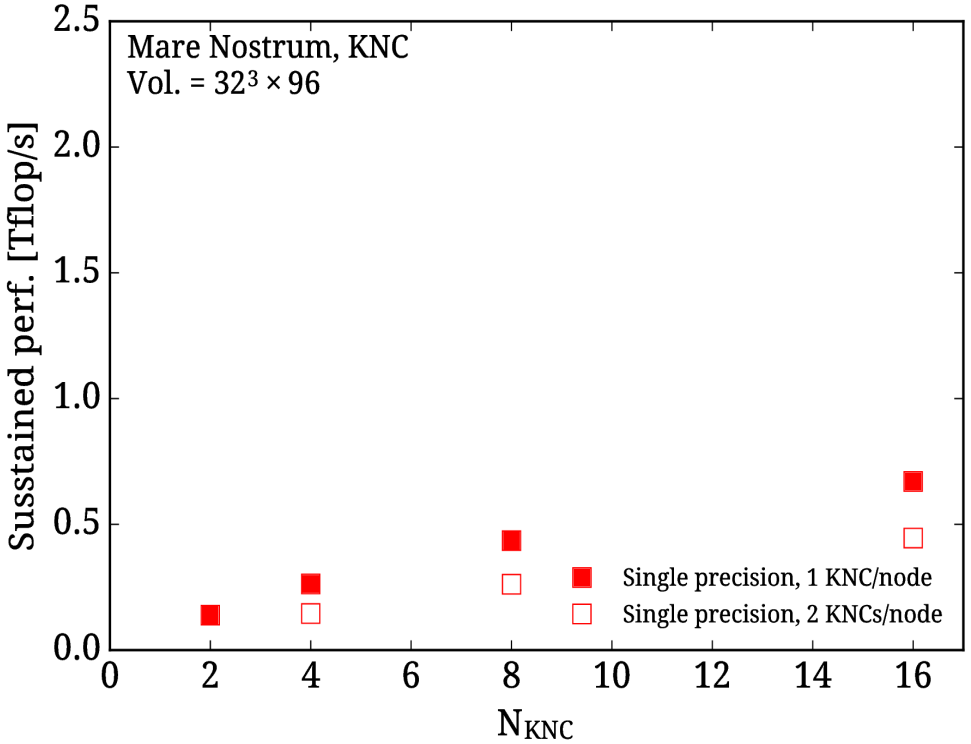


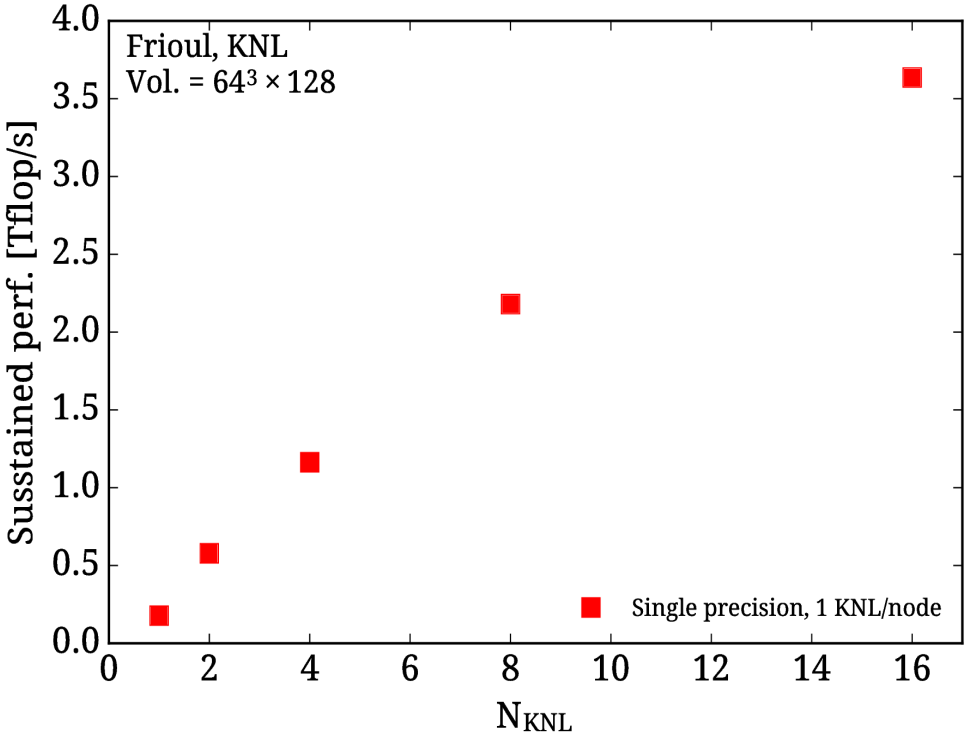


- MareNostrum (BSC) – KNCs (native mode)
- Frioul (CINES) – KNLs (cache mode, quadrant)

- Same problem size as GPU benchmark
- KNL system about 4 times more efficient

Strong Scaling: Xeon Phi Benchmark





- MareNostrum (BSC) – KNCs
  60 openOP threads per card
- Frioul (CINES) – KNLs
  68 openMP threads per card

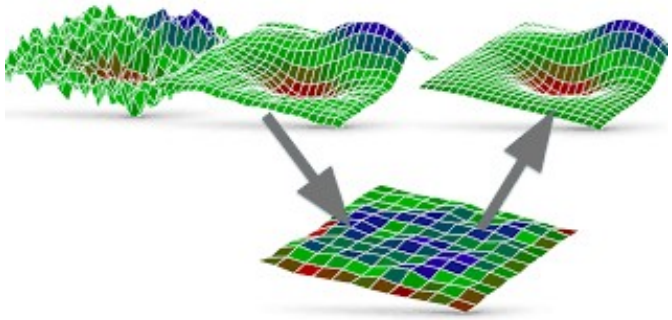
- Same problem size as GPU benchmark
- KNL system about 4 times more efficient

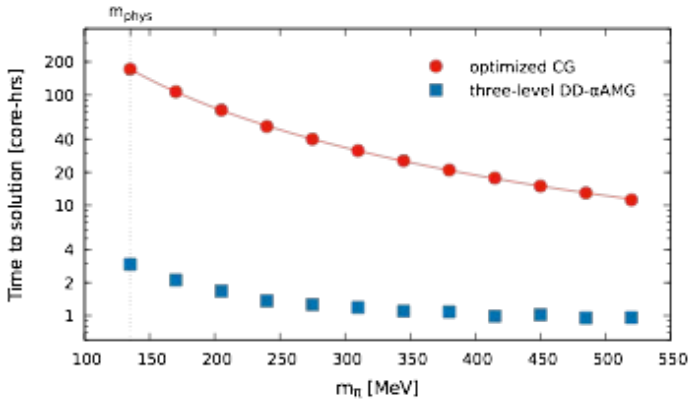
Weak scaling on V = 24 x 48^3 - Sustained Performance [Gflops/s]

Number of Cards	P100 - double	KNL - double	P100 - single	KNL - single
1	376.6	172.3	766.0	348.3
2	729.0	320.7	1473.0	616.7
4	1453.5	629.8	2865.6	1214.8
8	2884.4	1228.8	5421.3	2425.5
16	5004.5	2310.6	9373.8	4404.6
32	8744.1	---	17995.1	---
64	14053.0	---	27219.8	---

Outlook: possible Benchmark application in the future

Multigrid solver methods bringing x10-x100 speed-up in solver time





Being adopted widely by lattice QCD communities.

Computational/implementation challenges

- Majority of computation spent in coarse operator solves
- Poor strong scaling
- Difficult to distribute across thousands of cores in current implementations
- Issues are more pronounced on GPUs

Conclusion:

Unified European Application Benchmark Suite: http://www.prace-ri.eu/ueabs/

- UEABS Benchmark Suite: CPU case
  - Updated Kernel C and D
  - Scales up to 65 000 processes (4096 nodes)
- GPU: QUDA
  - Pascal scales better by using larger lattice sizes
- Xeon Phi: QphiX
  - KNL performs 4 times better than KNCs

THANK YOU FOR YOUR ATTENTION

www.prace-ri.eu