Experiences from the edge of Exascale

Andrew Emerson

Exascale Projects

- DEEP/DEEP-ER
  - Focus on software+hardware technologies based on Intel MIC KNC/KNL processors and other innovative technologies (e.g. NVM, NAM, etc)
- Mont Blanc/Mont Blanc 2
  - Emphasis on energy saving technologies with ARM chips
- Oprecomp
  - “Approximate” computing with trans-precision approaches.

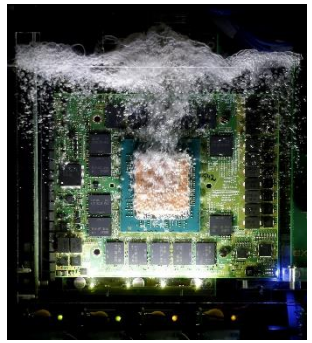
The DEEP project

Dynamical Exascale Entry Platform

Project acronym:	DEEP (Dynamical Exascale Entry Platform)
Contract no.:	ICT-287530
Project type:	IP/STREP
Start date:	1 December 2011
Duration:	45 months
End date:	31 August 2015
Total budget:	18 500 000 €
Funding from the EC:	8 300 000 €

The DEEP project

- Objective of project to build a heterogeneous cluster as a prototype for an Exascale computer based on many core chips (Intel MIC), innovative cooling technologies and a carefully designed runtime software stack.
- The main idea in DEEP is the Cluster-Booster concept where the computer is divided into two parts:
    1. The "Cluster" with conventional multi-core chips (e.g. SandyBridge)
    2. The "Booster" based on many-core chips, e.g. Intel KNC chips.
- The reasoning is that for most HPC applications, not all parts of the code are highly parallel so while the most parallel kernels will be run on the Booster, the more serial-like parts can stay on the Cluster.
- Final peak performance ~500 Tflops.
- Also experimentation with a smaller Booster (32 nodes) using immersive cooling technology.



DEEP GreenICE Booster

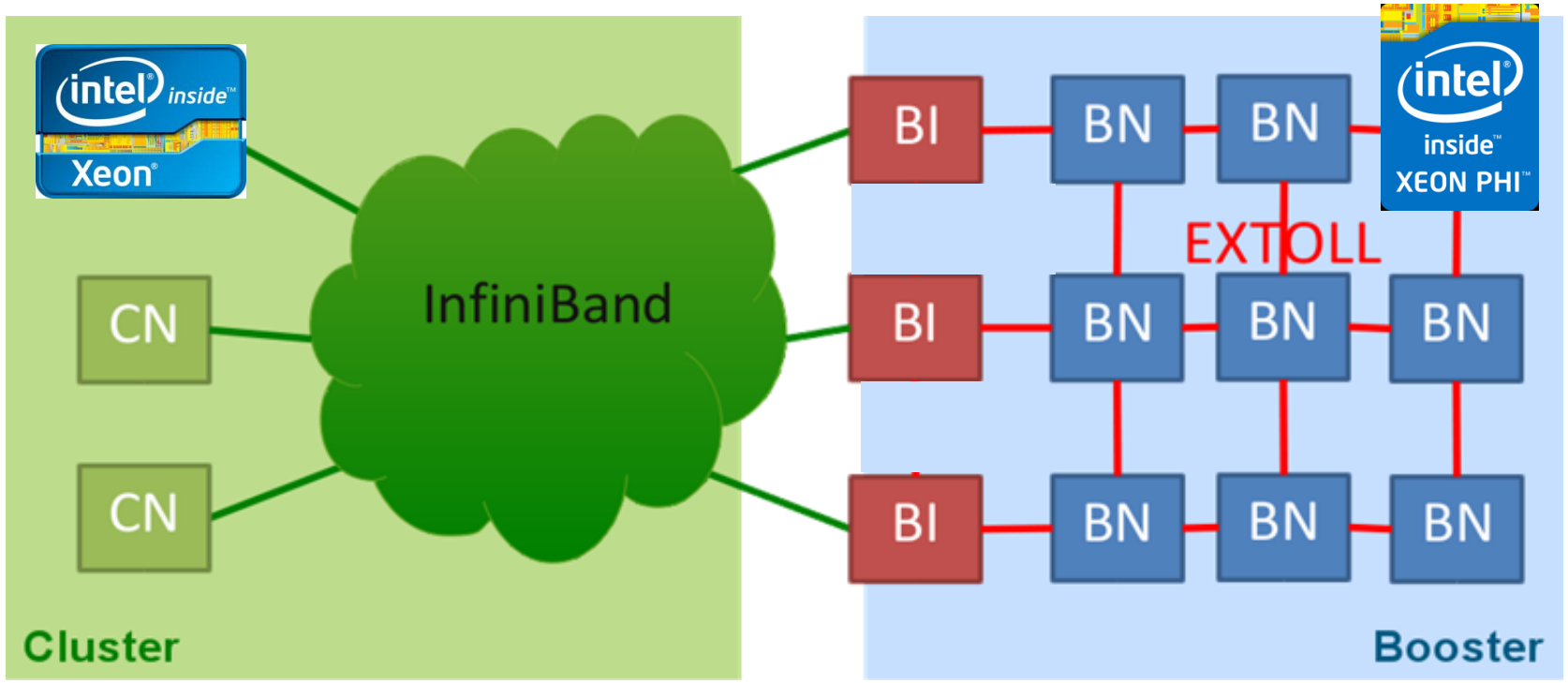
Important engineering effort by Eurotech and Intel. Remember that "standard" KNC chips are co-processors and not self-bootable.

Cluster-Booster architecture

The strategy is to design an architecture where applications can be divided into highly scalable and low or medium scalable code portions.

128 Xeon (Sandy Bridge)

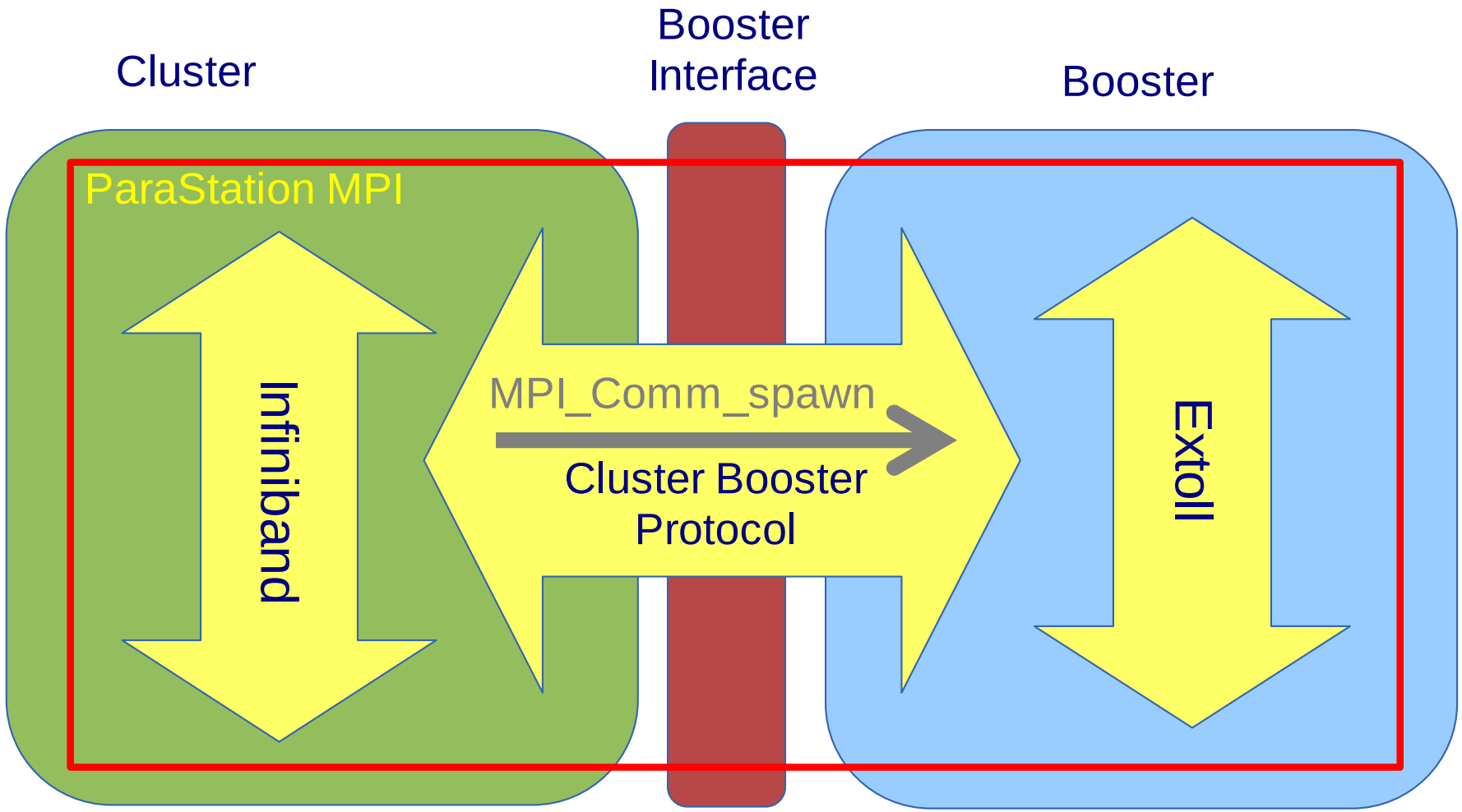
384 Xeon Phi (KNC)



Low/Medium scalable code parts

Highly scalable code parts

Programming environment



OmpSs on top of MPI provides pragmas to ease the offload process

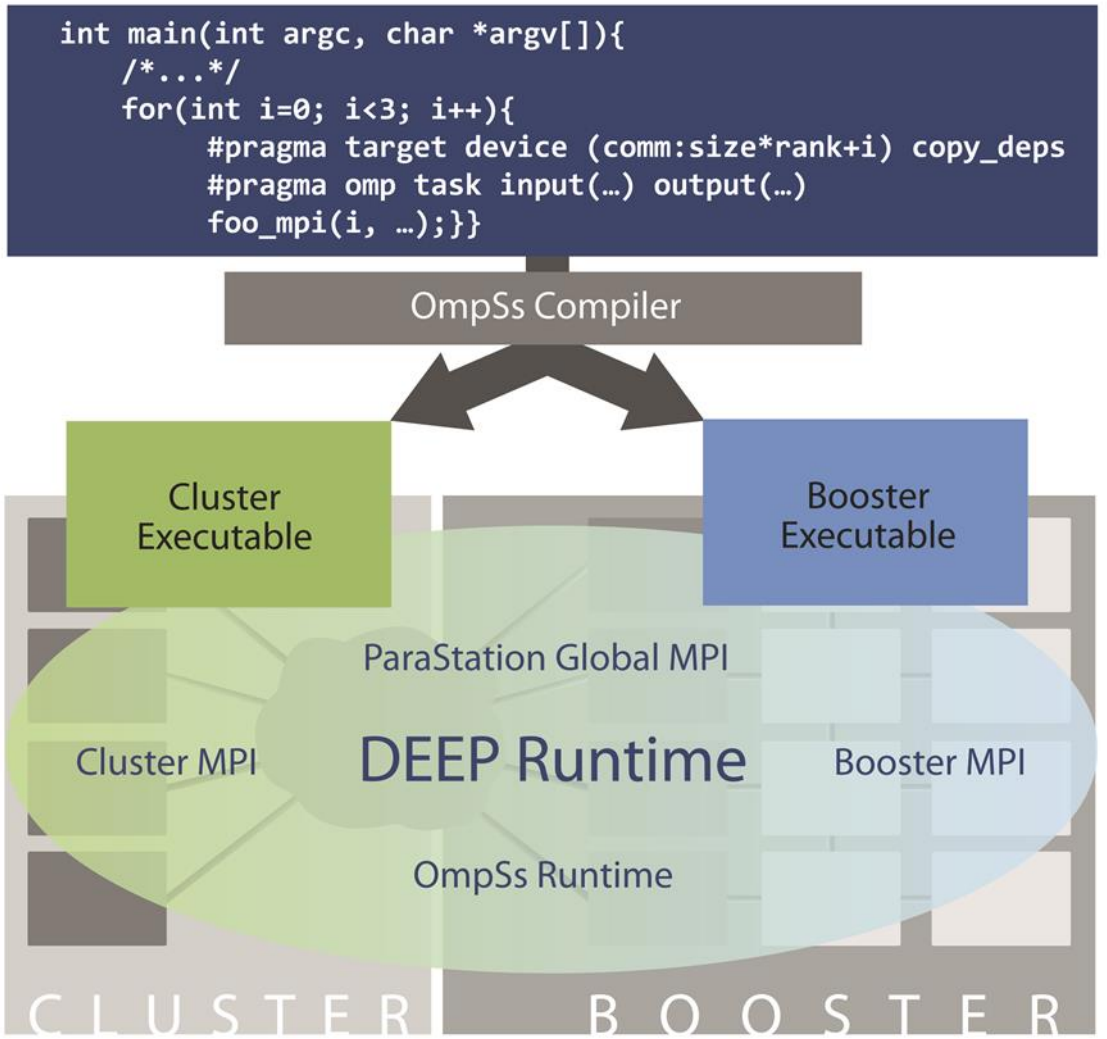
Application running on DEEP

Source code

Compiler

Application binaries

DEEP Runtime



OmpSs

Directive-based task programming model developed by the Barcelona Supercomputing Centre (BSC).

Allows tasks to be executed asynchronously according to the *data flow* of the program. Programmers indicate data dependencies of functions (i.e. *tasks*) via OpenMP-like directives.

Programs are compiled with the Mercurium compiler (*source-to-source compiler*) which generates tasks for the Nanos++ runtime.

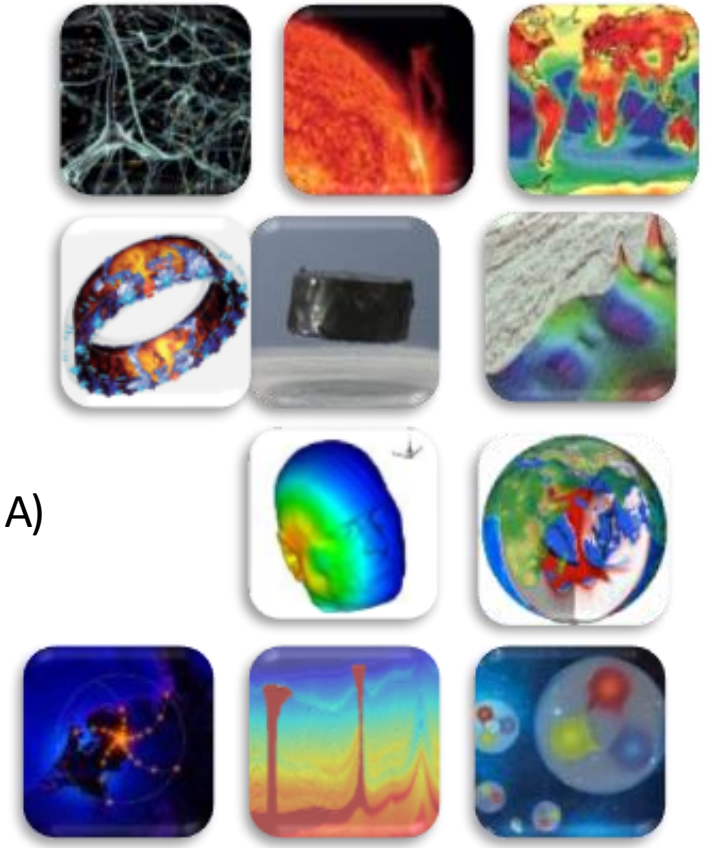
In DEEP/-ER extended version of OmpSs with offload facility.

Similarities and some convergence with the latest OpenMP versions (4.x) or Intel offloading directives but OmpSs is currently more flexible, e.g. OmpSs offload allows MPI calls within offloaded kernels (unlike Intel MPI).

```
/* test.c */
#include <stdio.h>
 int main(int argc, char *argv[])
 {
 int x = argc;
#pragma omp task inout(x)
 {
x++;
 }
#pragma omp task in(x)
{
printf("argc + 1 == %d\n", x);
}
#pragma omp taskwait
return 0;
 }
$ mcc -o test --ompss test.c
Nanos++ prerun
Nanos++ phase
```

Application enabling for the DEEP architecture

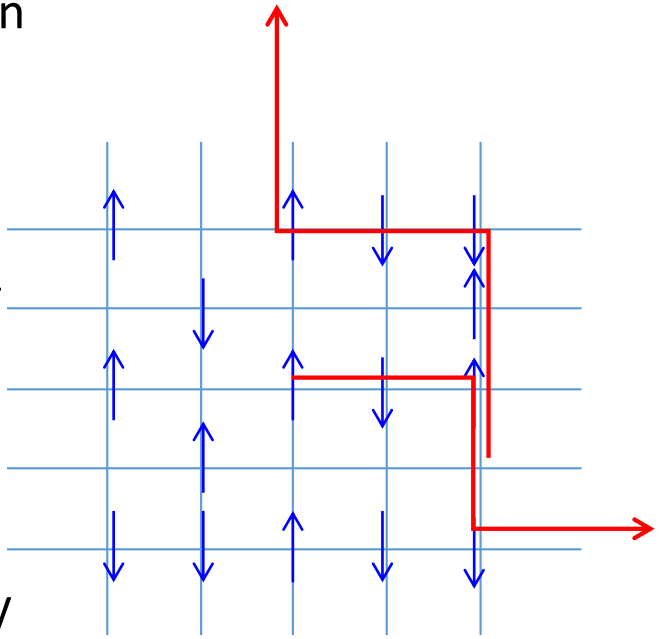
- **DEEP+DEEP-ER applications:**
  - Brain simulation (EPFL)
  - Space weather simulation (KULeuven)
  - Climate simulation (CYI)
  - Computational fluid engineering (CERFACS)
  - **High temperature superconductivity (CINECA)**
  - Seismic imaging (CGG)
  - Human exposure to electromagnetic fields (INRIA)
  - Geoscience (BADW-LRZ)
  - Radio astronomy (Astron)
  - Oil exploration (BSC)
  - Lattice QCD (UREG)



Wide range of application codes covering many areas of science and engineering.

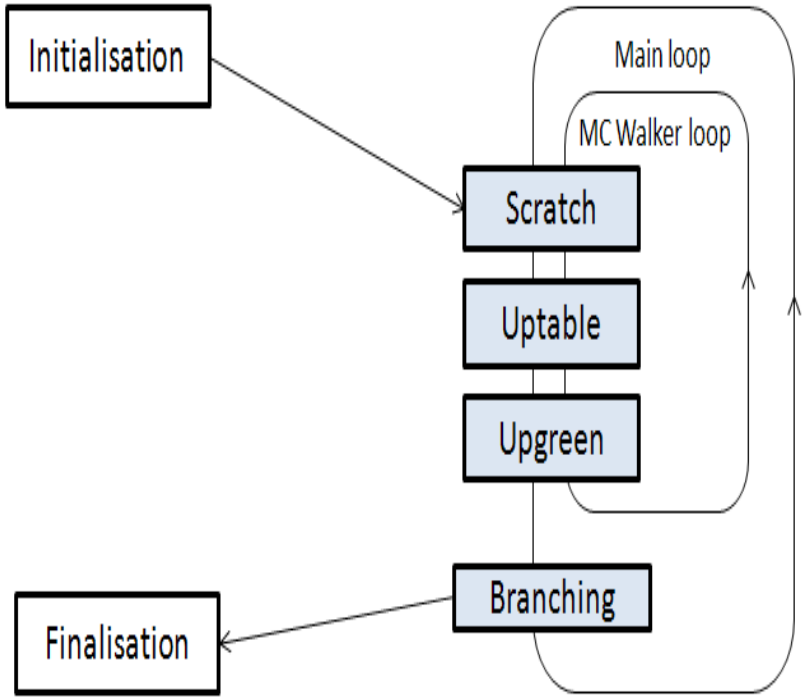
TurboRVB -Quantum Monte Carlo code

- TurboRVB is a Quantum Monte Carlo code which can be used to study quantum phenomena such as ferromagnetism and superconductivity.
- The calculations are based on different “walkers” that sample the space in an independent manner (# walkers defined by user).
- At user defined intervals walkers communicate to exchange relative weights.
- Every N steps, the values of the energy computed by each walker are reduced and written to a file.
- Written and developed by S. Sorella and others at SISSA (Trieste)



TurboRVB characteristics

- “Fortran90”, pure MPI code. Typical scientific academic code, few modern software design principles (many global variables).
- Execution time dominated by linear algebra routines (e.g MKL) performed by each walker (typically 90%).
- Highly parallel due to weak coupling between walkers. Normally 1 MPI task =1 walker, but walkers can share MPI tasks (strong scaling). Weak scaling if no. of. MPI tasks = no. of. walkers.
- Low memory, low I/O for normal system sizes.



Program structure – pseudo code

Pseudo code:

```
CALL initialization

DO imain  = 1, number_of_mc_steps
        DO j = 1,number_of_walkers
                CALL move_walkers_and_compute_sth
        ENDDO
        CALL mix_walkers_weights
        IF(write_flag)THEN
                CALL write_on_file
        ENDIF
ENDDO
CALL finalization
```

Initialize variables; calculate and write lattice positions; Print to stdout simulation's info

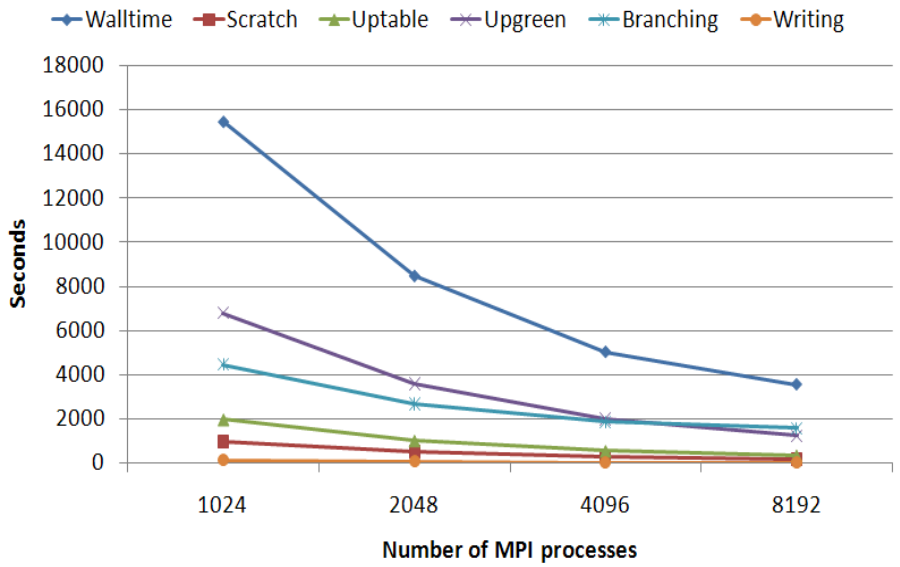
Computationally intensive; Massive use of dgemm.

Mixing of the walkers' weights. Based on p2p communications

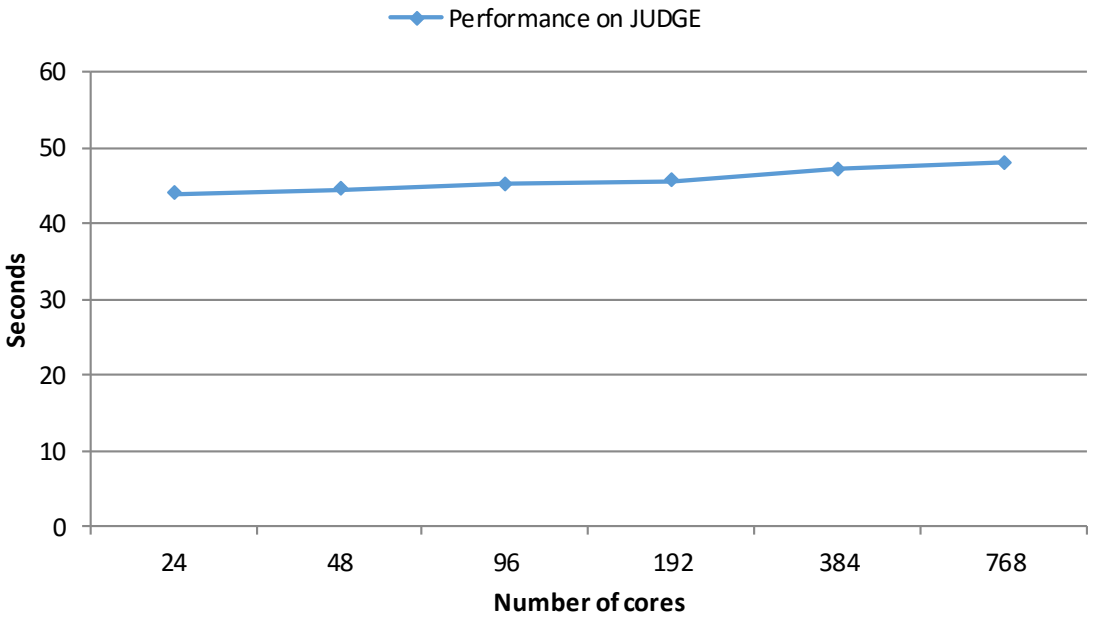
Dumping some information to file

Dumping final information to file

Application behaviour



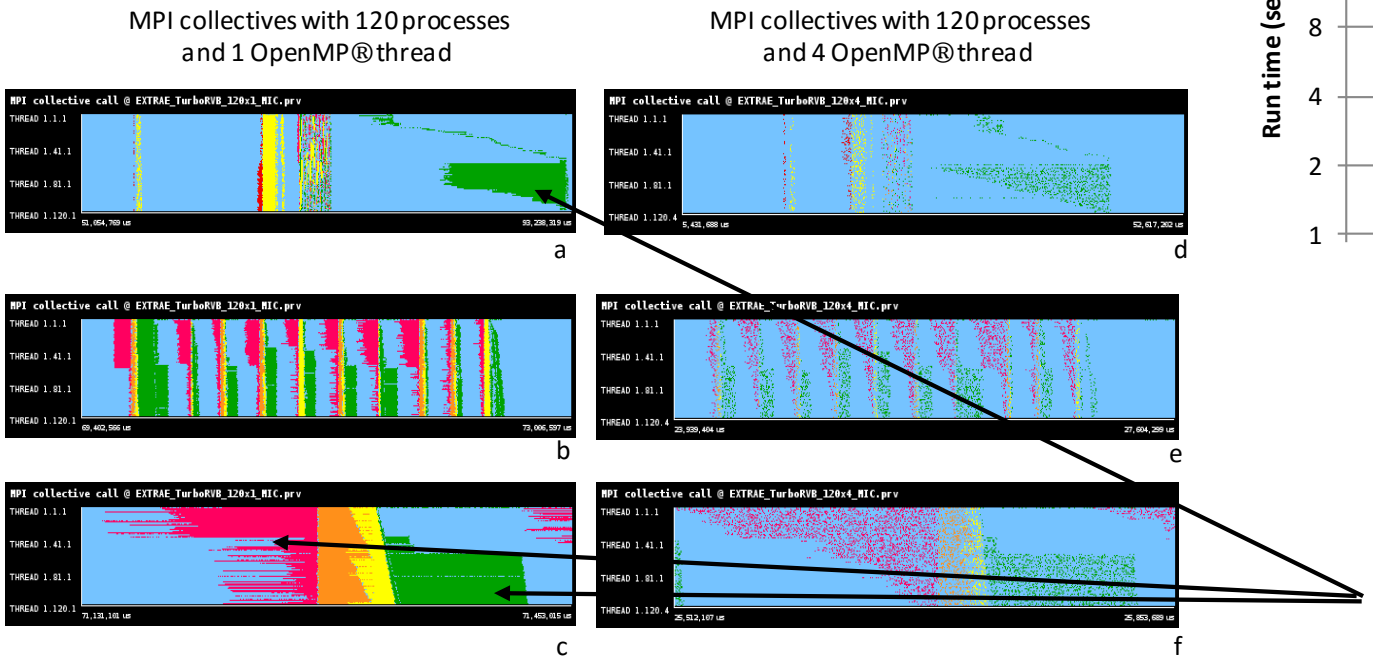
Strong scaling on Fermi (BG/Q) for 200 electrons.
No. of walkers=8192.
IBM XL Fortran + ESSL (2 OpenMP/MPI task)



Weak Scaling on Judge (Juelich). 1 MPI task=1 MC walker.

Weak Application behaviour on Xeon Phi

- The application does not scale well over multiple Xeon Phi nodes.
- Analysis with a profiler/trace tool (e.g. Extrae, Tau, Scalasca etc) can indicate problems with the communication pattern.



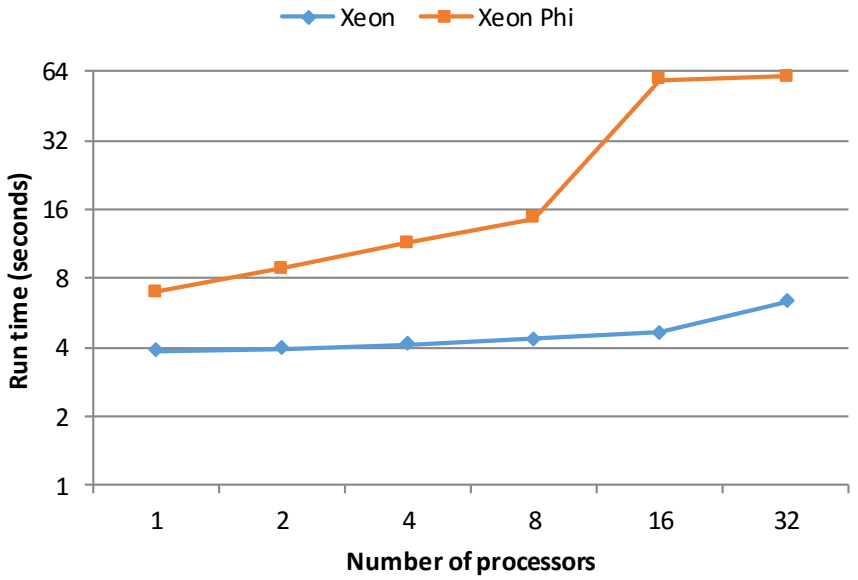


Figure 27: Weak scaling of TurboRVB. Xeon® setups used 8 MPI processes, 2 OpenMP® threads and 64 walkers per processor, whereas Xeon Phi™ used 60 MPI processes, 4 OpenMP® threads and 60 walkers per coprocessor (CINECA)

Porting of TurboRVB to the DEEP architecture - 1

CINECA SCAI
SuperComputing Applications and Innovation

Optimisation on Xeon Phi.

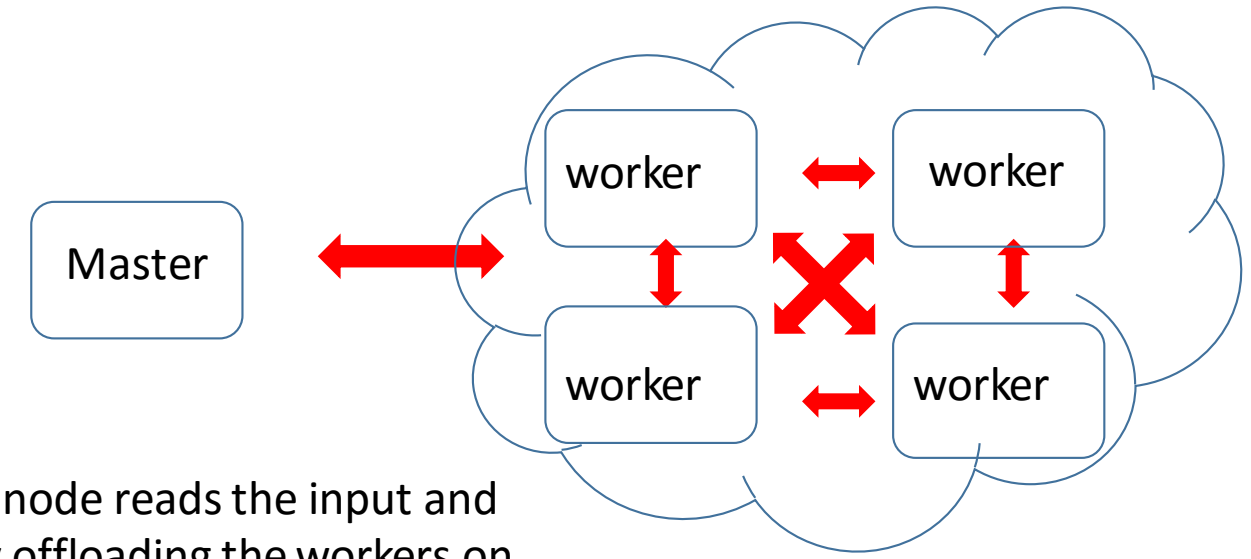
- Vectorisation.
  - Few loops have been identified which would benefit from enhanced vectorisation.
- OpenMP support.
  - Since MPI performance within MIC was expected to be poor this should bring benefits but the program style made this difficult and the effort was abandoned for DEEP (but continued for DEEP-ER).
- Reduce I/O.
  - Not really a great worry for QMC, although task-level restart files are inconvenient. Parallel I/O strategies (e.g. SIONLIB) being pursued instead in DEEP-ER.

But in any case for many problems up to 90% of time may be spent in the linear algebra libraries so gains may be limited.

Porting of TurboRVB to the DEEP architecture - 2

Focus in DEEP on restructuring code to allow offload via OmPss from the MPI-only version.

walkers are spawned on booster nodes via ompss tasks



- The single MASTER node reads the input and starts the program by offloading the workers on the booster nodes.
- Each worker is really one MPI task, but is wrapped by OmpSS.

Porting of TurboRVB to the DEEP architecture - 3

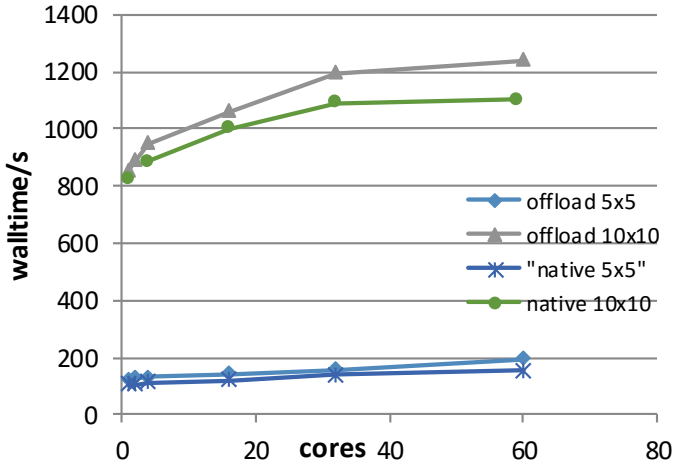
- Porting was not trivial since not only does the code need re-arranging but OmPss requires that **`allocatable`** arrays are allocated before being offloaded (in the original code the allocations were spread out through the program).
- But the process with OmpSS still easier than with OpenMP/MPI or Intel directives.

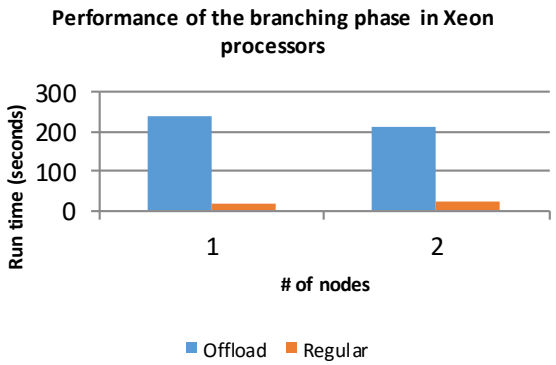
Wraps MPI_Spawn

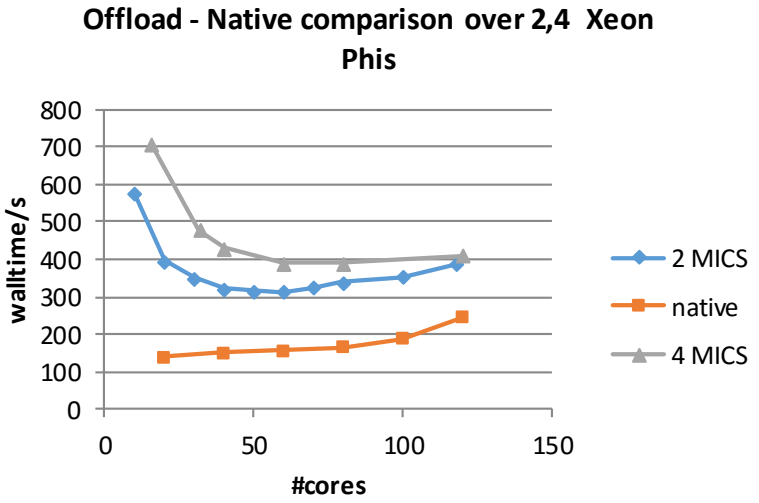
```
CALL READ_INPUT
CALL DEEP_BOOSTER_ALLOC (MPI_COMM_SELF, nhosts, nboosters, comm_boosters)
!$OMP TASK in(itest,iopt,ngen,nw,nscra,iseedr,nelup,neldo,&
!$OMP nx,ny,nxp,nyp,pbcx,pbcy,tabflag,krepm,ncheck,errmax,release,&
!$OMPavlast,power,trotter,ltab,U,lambda,gammat,gshift,mu,muwf,tprime,tprimewf,&
!$OMPcuteloc,ifstop,delta,deltaf,gutz,dtg,rmshort,swave,mesbcs,yesrew,epsreg,epsmach, &
....
!$OMPncore,iread,iskip,evalopar,yestwo,tpar,iboot,nweight,epsi,epsdgel,ieser,tbra,etria
l,&
!$OMPktrial,btrial,nbra,nbrag,nbrat,typereg,recbra,fngutz,epscut,epscutg,cutrelease)
ONTO(BOOSTERS,I)

call offloadcode

!$OMP end task
ENDDO
!$OMP taskwait
CALL DEEP_BOOSTER_FREE(comm_boosters)
CALL nanos_mpi_finalizef()
```

Offload TurboRVB results on DEEP

- The DEEP Cluster/Booster was not available before the end of the project so the applications were tested instead on the MIC partition on Mare Nostrum.
- OmPss offload compared well with Native mode in one node – but no so well over multiple nodes.
- Analysis revealed bottlenecks in the branching routines (collectives + point-to-point).
- Without the DEEP cluster/booster difficult to draw conclusions other than the approach appears feasible.







DEEP Summary

- Engineering delays unfortunately meant that the machine was not ready before the end of the project (one application did run shortly after the project finished).
- The TurboRVB application probably wouldn't have ported too well on he DEEP cluster given the poor performance over multiple KNC nodes. But much was learnt about the application and the communication patterns affecting performance. An OpenMP version might have been useful but would have required too much effort.
- The OmpSs offload mechanism was successful although inefficient on KNC.
- Other applications benefited significantly from the efforts in optimisation and vectorisation and some probably would have exploited well the Cluster/Booster design.
- Currently, the machine is up and (sometimes) running at the Juelich Supercomputer Centre.

DEEP-ER (DEEP – Extended Reach)

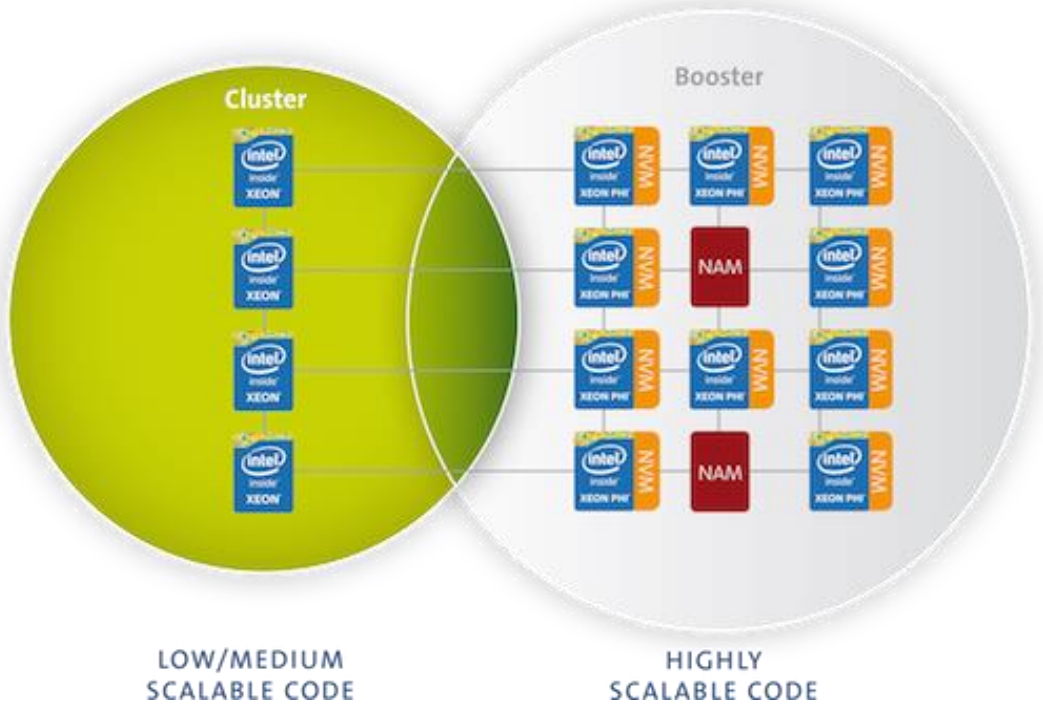
Similar goals to DEEP: construction and testing of a Booster-Cluster architecture (this time based on KNL), but more emphasis on experimenting with hardware and software technologies.

Strong promotion for the idea of *Co-Design*, i.e. design hardware, middleware and software applications together to improve integration.

DEEP-ER Cluster and Applications

7 Applications

- Human Exposure to Electromagnetic Fields (Inria)
- Radio Astronomy (Astron)
- Earthquake Source Dynamics (LRZ)
- Enhancing Oil Exploration (BSC)
- Lattice QCD (UREG)
- Space Weather (KU Leuwen)
- High Temperaure Superconductivity (Quantum Monte Carlo, Cineca)



DEEP-ER technologies

- Cluster-Booster based on KNLs and EXTOLL network.
- Non-Volatile Memory(NVM) and Network Attached Memories (NAM)
- BeeGFS filesystem and SIONlib + Exascale 10 libraries for fast parallel I/O.
- Task based resiliency with OmpPs and SCR (scalable checkpoint and restart)
- Parastation MPI
- Application tools such as JUBE (benchmarking), Extrae/Paraver (profiling and traces)

To allow hardware, software and application developers to do the co-design effectively a small prototype (Software Development Vehicle or SDV) was made available early on in the project.

Xeon part (or SDV-Cluster)
16 dual-socket Intel® Xeon® E5-2680 nodes
16 NVMe cards Intel DC P3400, 400 GB each (one in each server)

Xeon Phi part (or SDV-Booster)
8 Intel® Xeon Phi (KNL) nodes
2 NVMe cards Intel DC P3400 (integrated in two of the KNL boards)
Storage
2 storage servers (spinning disks, 57 TB) and
1 metadata server (SSDs)
Additionally, the SDV has also one NAM (network attached memory) board integrated.

Cineca Applications for DEEP-ER

TURBORVB

- As for DEEP but concentrate on technologies for I/O and resilience (SIONLIB, SCR)

2degas (“the mock-up”)

- Because of poor performance on Xeon Phi Cineca was asked to try an alternative QMC code.
- Chose 2degas, a similar application to TURBORVB but with a simpler communication pattern

2degas Mock-up

2Degas is a QMC for the simulation of a two-dimensional electron gas based on a Jastrow-Slater basis function.

Developed in the Physics dept. of University of Rome “La Sapienza” by Moroni and coworkers.

Written in plain Fortran77+MPI. Very scalable (minimal MPI communications), but only weak-scaling (unlike Turbo). Each MC walker is represented by 1 MPI process.

Advantages

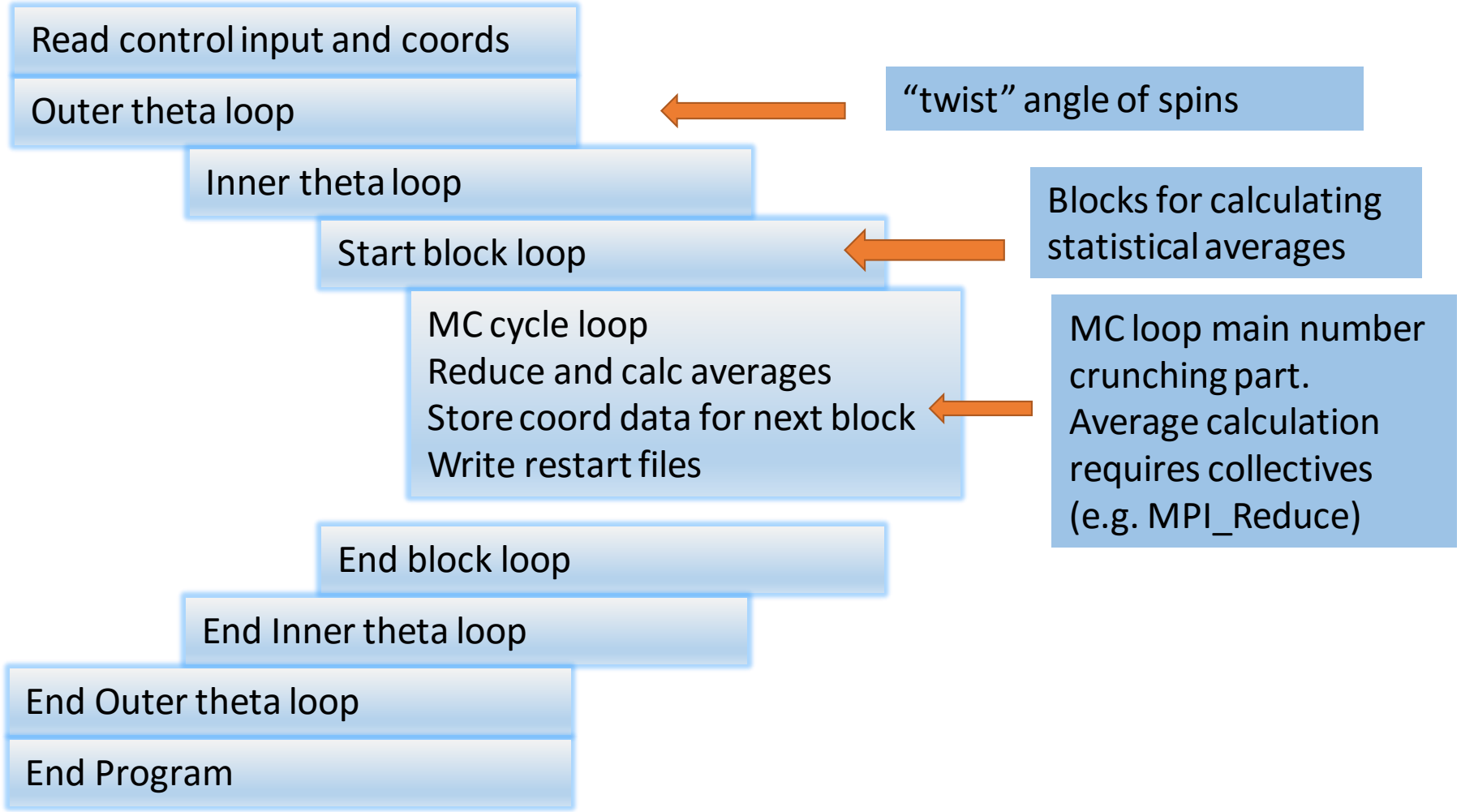
- Smaller and easier code.
- The internal kernel is lightweight and the communication pattern is simpler.

Disadvantages

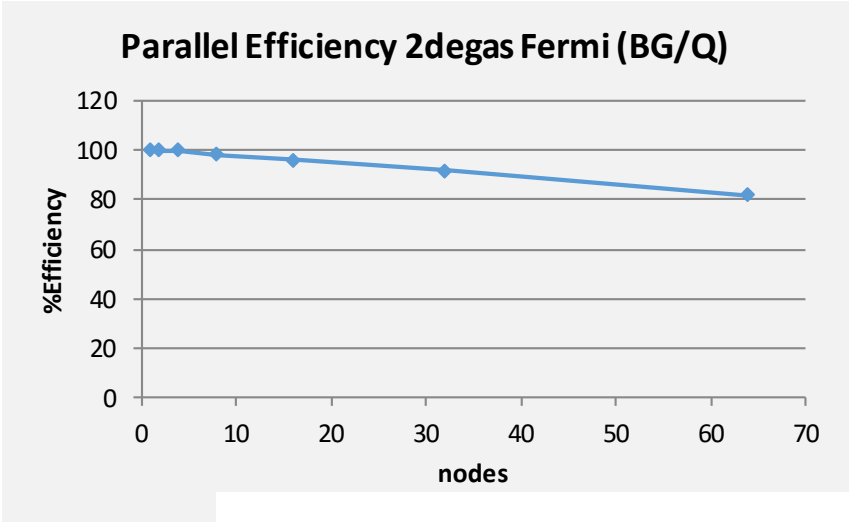
- Quite an “old” code(12 yrs), certainly requires some modernisation.

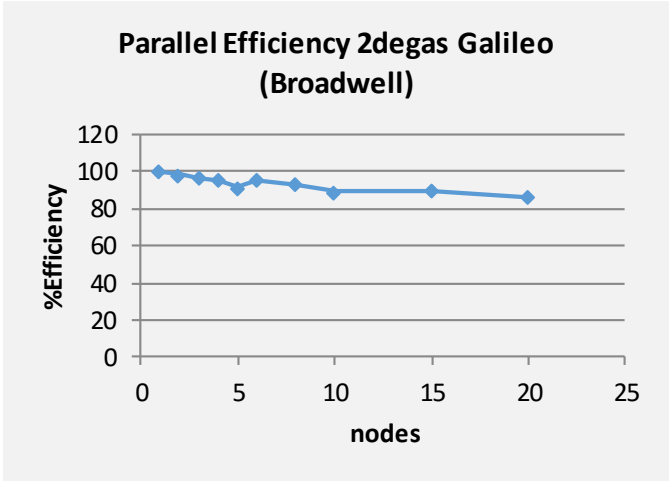
No OpenMP version

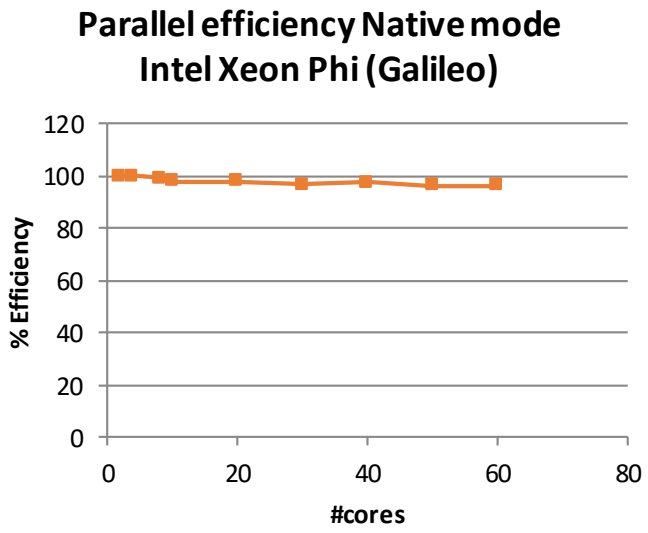
2degas program structure for standard input



Performance of mock-up on various architectures







Very high parallel efficiency, even with MPI on KNC.

Porting of mock-up to SDV

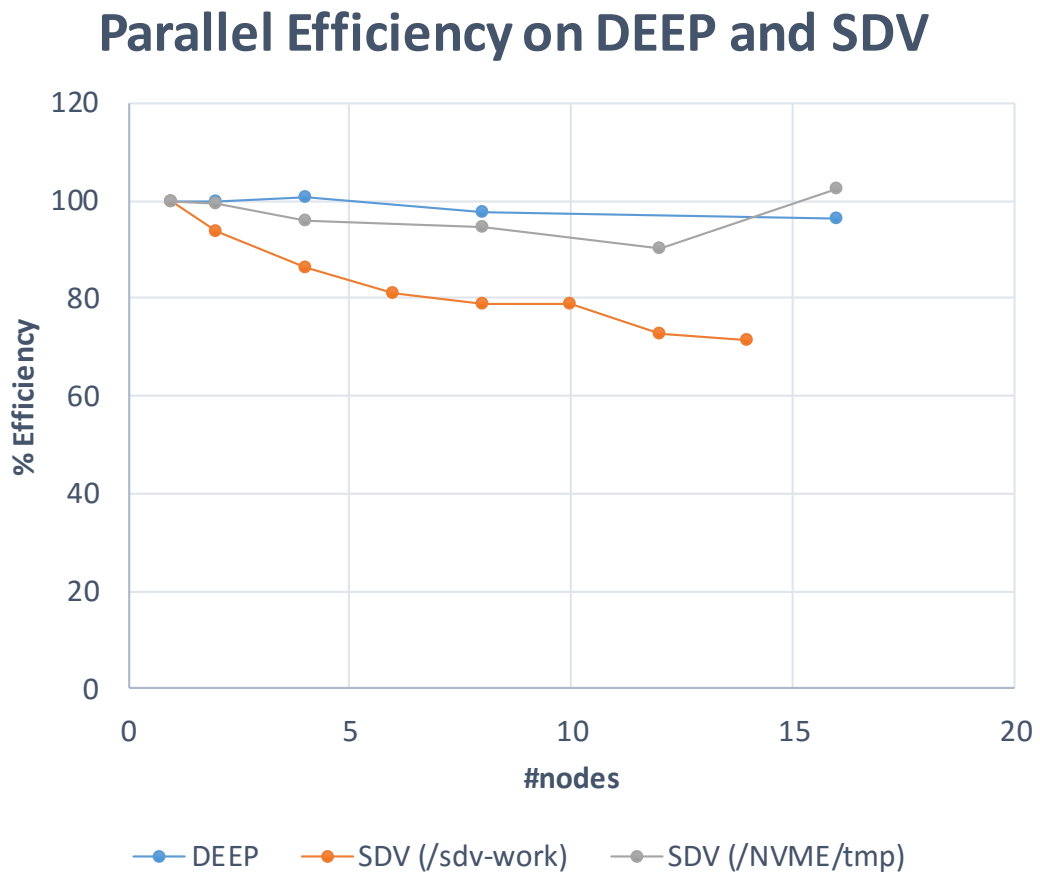
Original intention to port directly the F77 version onto the SDV and modernise the code later.

Decided to bring forward the modernisation activity to a partial F90 version:

- F77 version did not compile on MICs due to unaligned COMMON blocks. Rather than re-align COMMONs, decided to replace with F90 modules.
- Some hard limits in the number of walkers were best resolved with F90 constructs.
- Original version used inbuilt LAPACK and LINPACK routines for matrix determinants. Replaced with LAPACK calls for compatibility with MKL.

Little performance difference, but code now easier to modify.

Performance of MPI mock-up on SDV



Poor parallel scaling of mockup on SDV was attributed to the **/sdv-work** filesystem on the SDV. Changing directories to the **/NVME/tmp** alternative filesystem (requiring code changes for task-local directories)
removed the performance problem.

OpenMP Port of 2degas

Top Hotspots

This section lists the most active functions in your application performance.

Function	CPU Time
slater_excitations	2034.279s
__intel_avx_rep_memset	1000.038s
makeg	409.321s
ZGETRF	364.830s
dotg	270.306s
[Others]	1490.877s

Initialising a large array

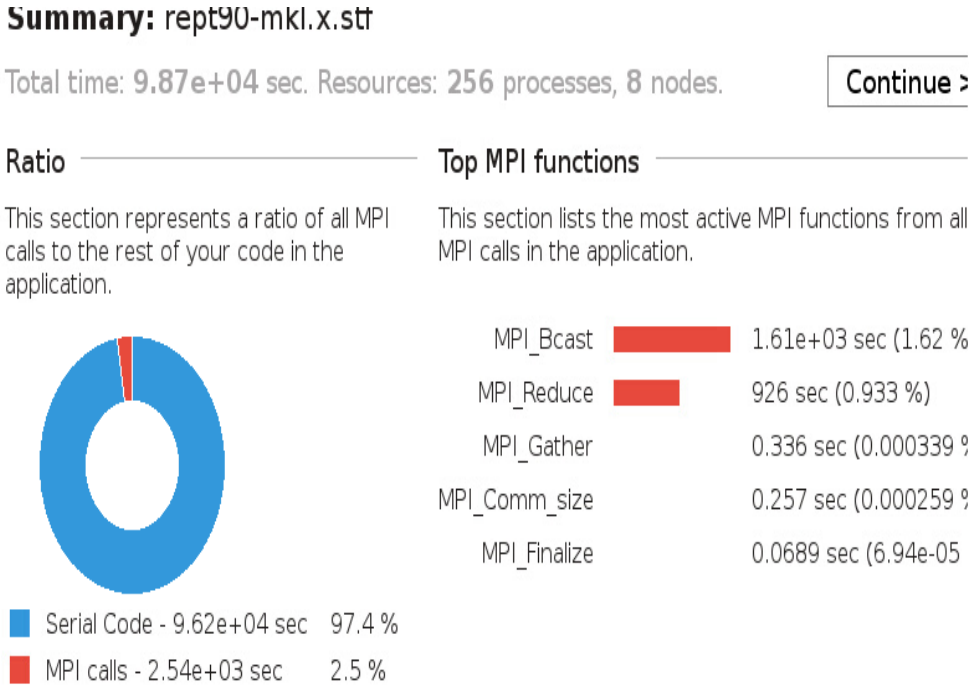
Callees	CPU Time: Total — Effective Time by Utilization (Idle / Poor / Ok / Ideal / Over)	Spin Ti: Imb..	Spin Ti: Lo
trial_function	99.8%	0.0%	0.0
slater_excitations	60.1%	0.0%	0.0
three_body	39.1%	0.0%	0.0
two_body	0.6%	0.0%	0.0
__intel_avx_rep_me	0.0%	0.0%	0.0
r_set	0.0%	0.0%	0.0
_intel_fast_memcp	0.0%	0.0%	0.0
r_set	0.0%	0.0%	0.0
[MKL BLAS]@xzger	0.0%	0.0%	0.0
__intel_avx_rep_me	0.0%	0.0%	0.0
[MKL BLAS]@xzswa	0.0%	0.0%	0.0

- One reason for adopting the 2degas mock-up was to investigate QMC on MIC with OpenMP, something deemed to difficult with TurboRVB (complex communication pattern).
- Initial Vtune analysis of MPI version on Haswell looked promising…

..but this is misleading because this is an average over 800,000 MC cycles

OpenMP port of 2degas

- A hybrid approach of threading each MPI task (i.e. each MC walker) with OpenMP was abandoned:
  - Most loops are a function of the number of electrons (in our case 26 in total), so are quite short and often very nested.
  - The sequential nature of the MC algorithm does not easily allow OpenMP tasks within each walker.
- Opted instead to convert pure MPI version to pure OpenMP version. Not expected to gain much in the short term, even on MIC, because of low MPI communications but (maybe) useful for MIC implementations.



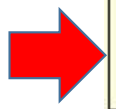
ITAC Analysis

Re-factoring of 2degas code

- For OpenMP conversion decided to do major refactoring of the code, involving subroutines used in the benchmark input, converting everything to f90.
- Used Photran re-factoring tools + custom scripts to perform the conversion.

```
[aemerson@node168 2degas-orig]$ tree -
.
├── a
├── docs
│   └── tesi.ps.gz
├── ewald.h
├── fake_mpi
│   ├── mpif.f
│   └── mpif.h
├── main.f
├── reptation.f
└── setup.f
```

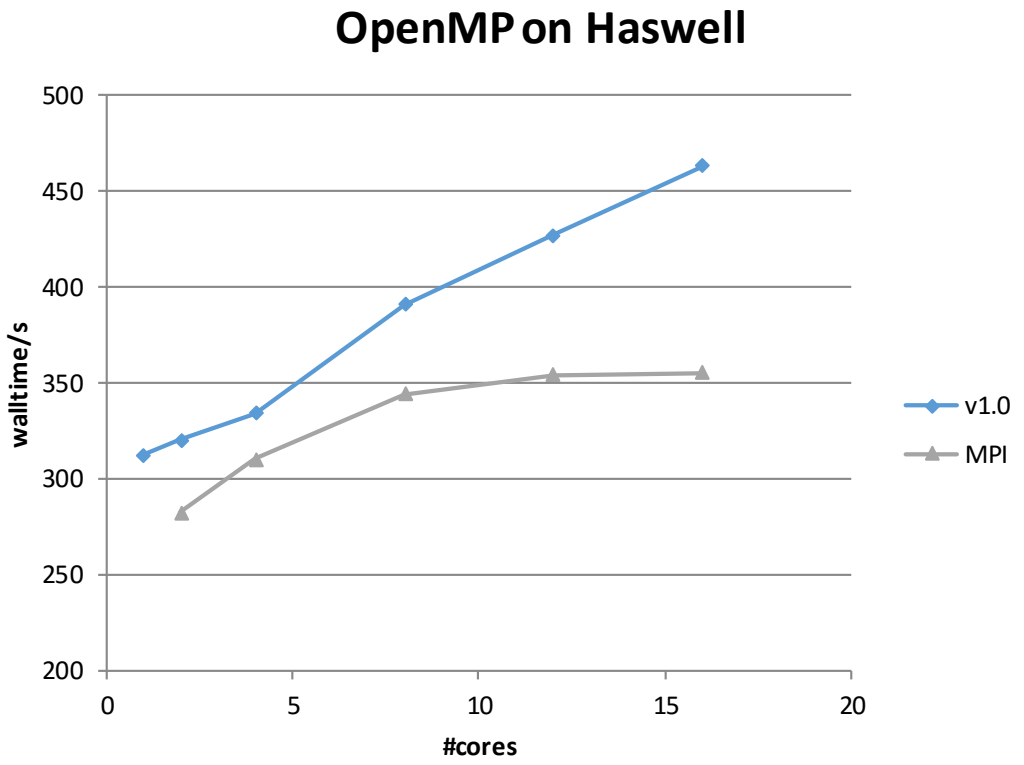
reptation.f (10K lines) → reptation.f90 (5859 lines)



```
.
├── compile-fermi.sh
├── compile-mic.sh
├── comp-mpi.sh
├── fake_mpi
│   ├── mpif.f
│   └── mpif.h
├── jube
│   ├── deep-2degas.xml
│   ├── fermi-2degas.xml
│   └── galileo-2degas.xml
├── makedefs
│   ├── make.defs.fermi
│   ├── make.defs.galileo
│   ├── make.defs.mic
│   └── make.defs.scalasca
├── make.defs
├── makefile
├── rept90-mpi-mic.x
├── src
│   ├── ewald.f90
│   ├── main.f90
│   ├── reptation.f90
│   ├── setup.f90
│   ├── three_body.f90
│   └── tools.f90
├── test
└── tools
    ├── f77conv.pl
    ├── sdv-copy.sh
    └── sdv-rm.sh
```

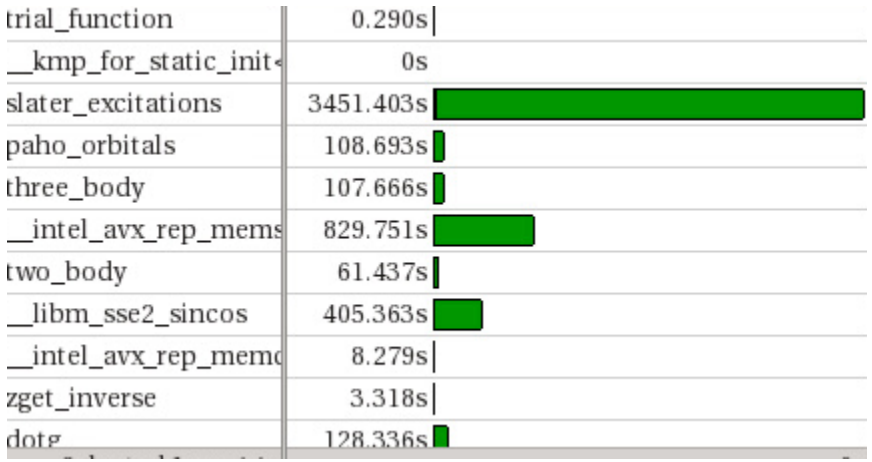
OpenMP mock-up v1.0

- Final seg fault resolved mid- August (thanks to Totalview)
- Initial tests on Galileo (Haswell) node disappointing when compared to MPI version.



OpenMP v1.0 – Vtune Analysis

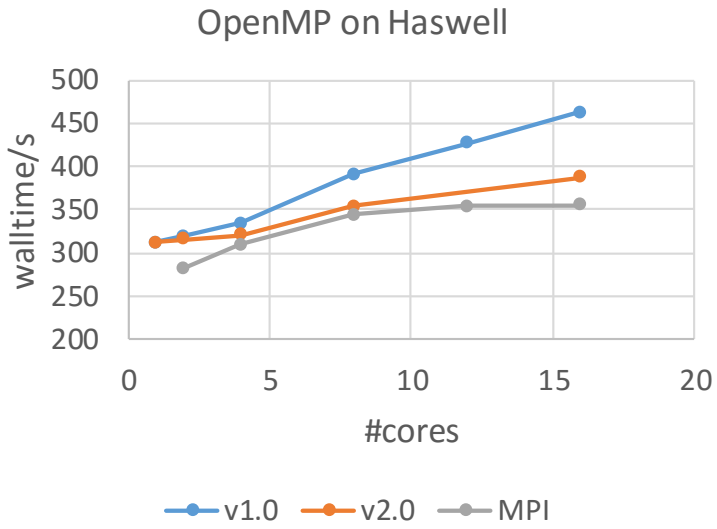
Function / Call Stack	CPU Time: Effective Time by Utilization (Idle Poor Ok Ideal Over)	CPU Time: Spin Time	CPU Time: Overhead Time: Cre...	Sch..	Red..	Other	Module	Function (Full)
__kmpc_critical_with_	0s	700.780s	0s	0s	0s	0s	libiomp5.so	__kmpc_critical_with_hint
write_conf ← vmc ←	0s	683.270s	0s	0s	0s	0s	rept.x	write_conf
restart ← averages ←	0s	9.659s	0s	0s	0s	0s	rept.x	restart
read_conf ← vmc ←	0s	7.851s	0s	0s	0s	0s	rept.x	read_conf
__kmpc_barrier	0s	236.569s	0s		0s	0.022s	libiomp5.so	__kmpc_barrier
write	0s	54.604s	0s	0s		0s	libpthread-2.17.so	write

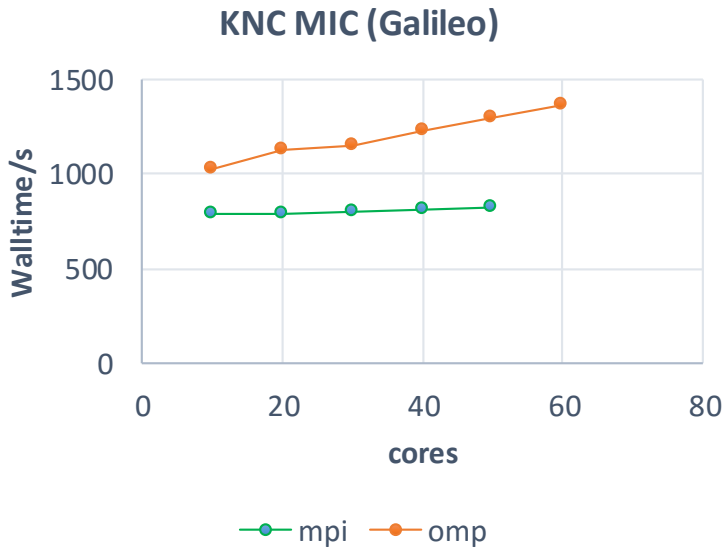


- Vtune confirmed poor performance and scaling due to omp critical.
- In particular in the routine which uses scratch files to store coordinates between Monte Carlo trial moves

OpenMP v2.0

- Scratch files “removed” to memory.
- Better performance and scaling on Haswell, but performance and scaling still poor on KNC.





OMP v2.0 on KNL

- Same story on SDV/KNL.
- Vtune analysis again points the finger at an OMP critical region.
- But this time in the routine which writes restart files

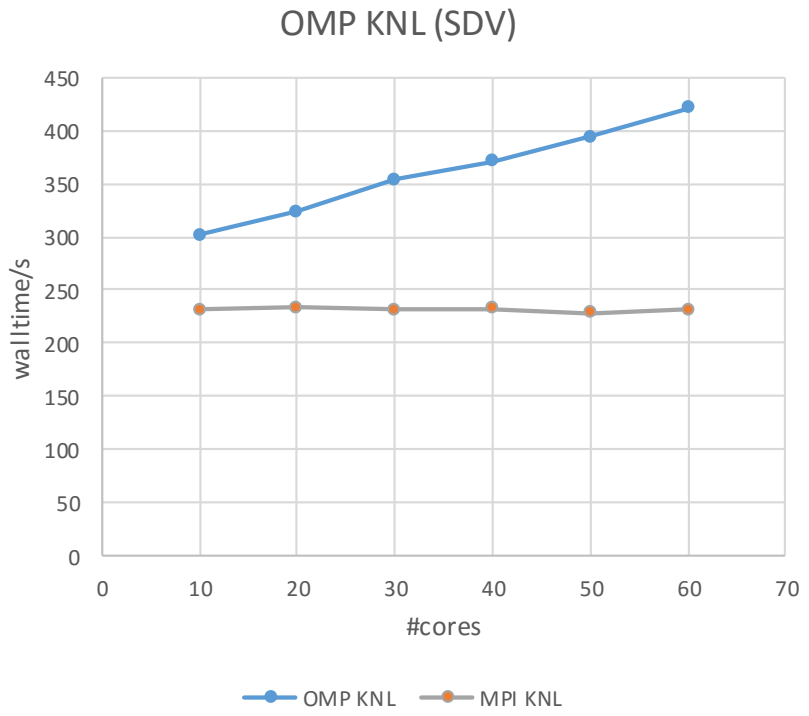
Elapsed Time: 401.262s

CPU Time:	23224.785s
Effective Time:	18364.625s
Spin Time:	4815.873s

A significant portion of CPU time is spent waiting. Use this metric spinning. Consider adjusting spin wait parameters, changing the backing off then descheduling), or adjusting the synchronization gran

Imbalance or Serial Spinning (OpenMP): 2519.223s

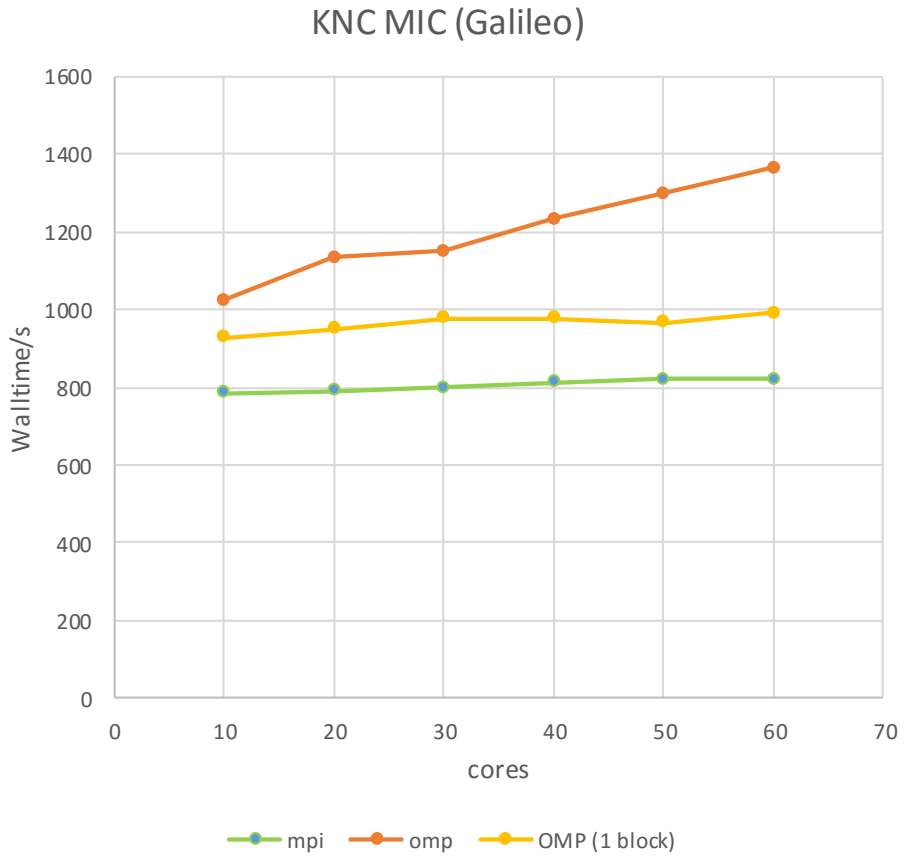
CPU time spent waiting on an OpenMP barrier inside of a parallel r Where relevant, try dynamic work scheduling to reduce the i execution (Serial - outside any region) may signal significant



10 blocks, 500 MC cycles/block

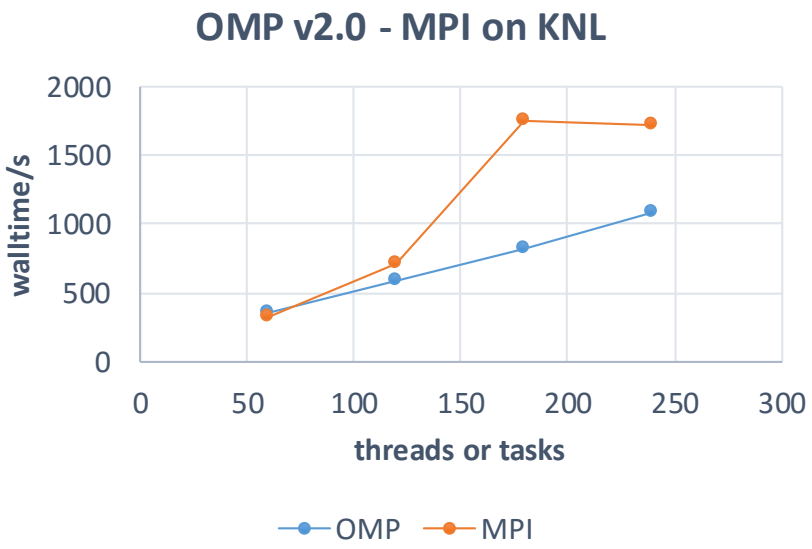
OMP v2.0 on KNC

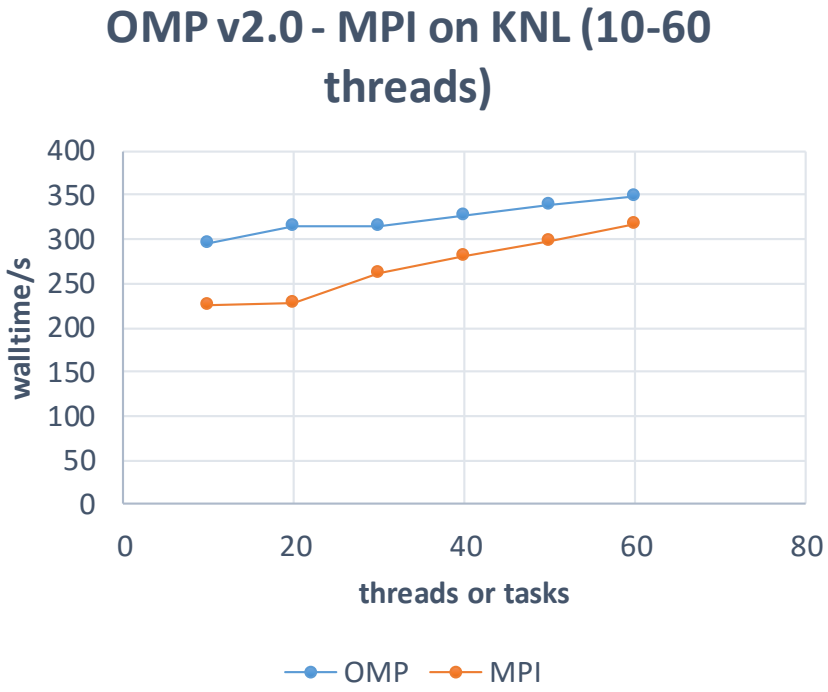
- In FORTRAN, need to use unit numbers which are distinct for each thread.
- If this is done can remove the **`$OMP critical`**.
- Scaling much better, although performance still less than MPI.



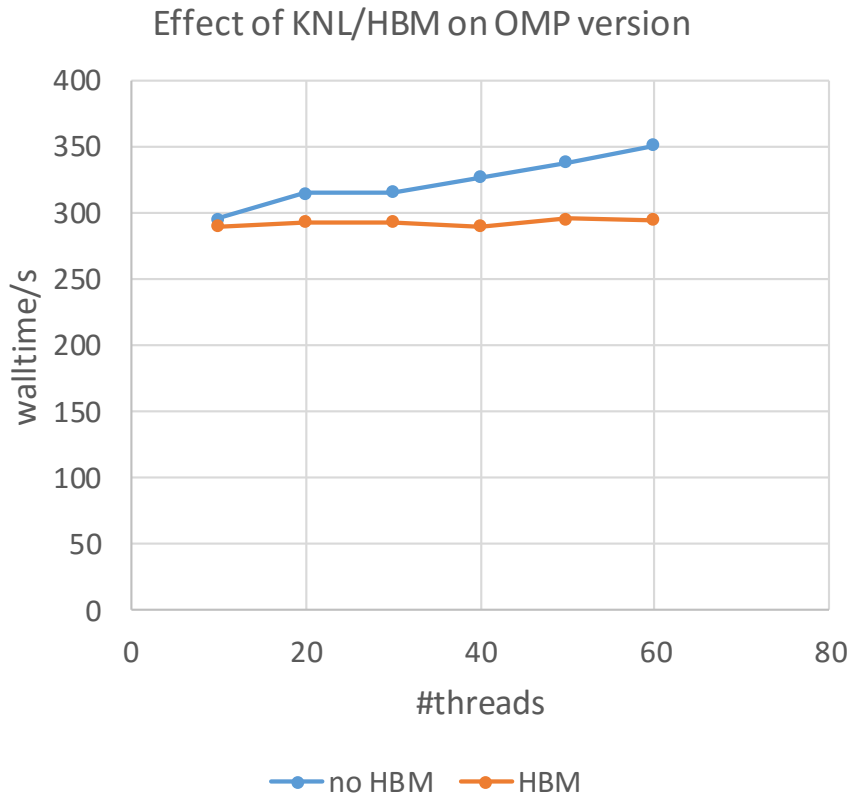
OMP v2.0 on KNL

- Same story on KNL, but if we push up the number of threads, OpenMP starts to beats MPI (single node)





KNL optimisations



- Because of short loops, little scope for vectorisation.
- Using the MCDRAM (High Bandwidth memory)
- HBM used with numactl but could consider memkind library.
- Small influence of HBM, not suprising given low memory footprint of this input.
- Other applications have reported greater speed-ups (e.g. 2x).

```
numactl --membind 1 ./rept.x
```

OpenMP conversion summary

- OpenMP threading of original MPI Walkers not viable due to small loops and lack of tasking possibilities (sequential algorithm).
- Decided instead to convert MPI tasks into OpenMP threads. No great speedup expected with current input on MIC because MPI communications low.
- Initial performance results disappointing due to two omp critical regions. One can be removed by using memory instead of scratch files. The second removed but localising FORTRAN input unit numbers.
- Performance and scaling broadly similar to MPI version upto 60 threads/tasks at which point OpenMP becomes much better.

Beyond OpenMP

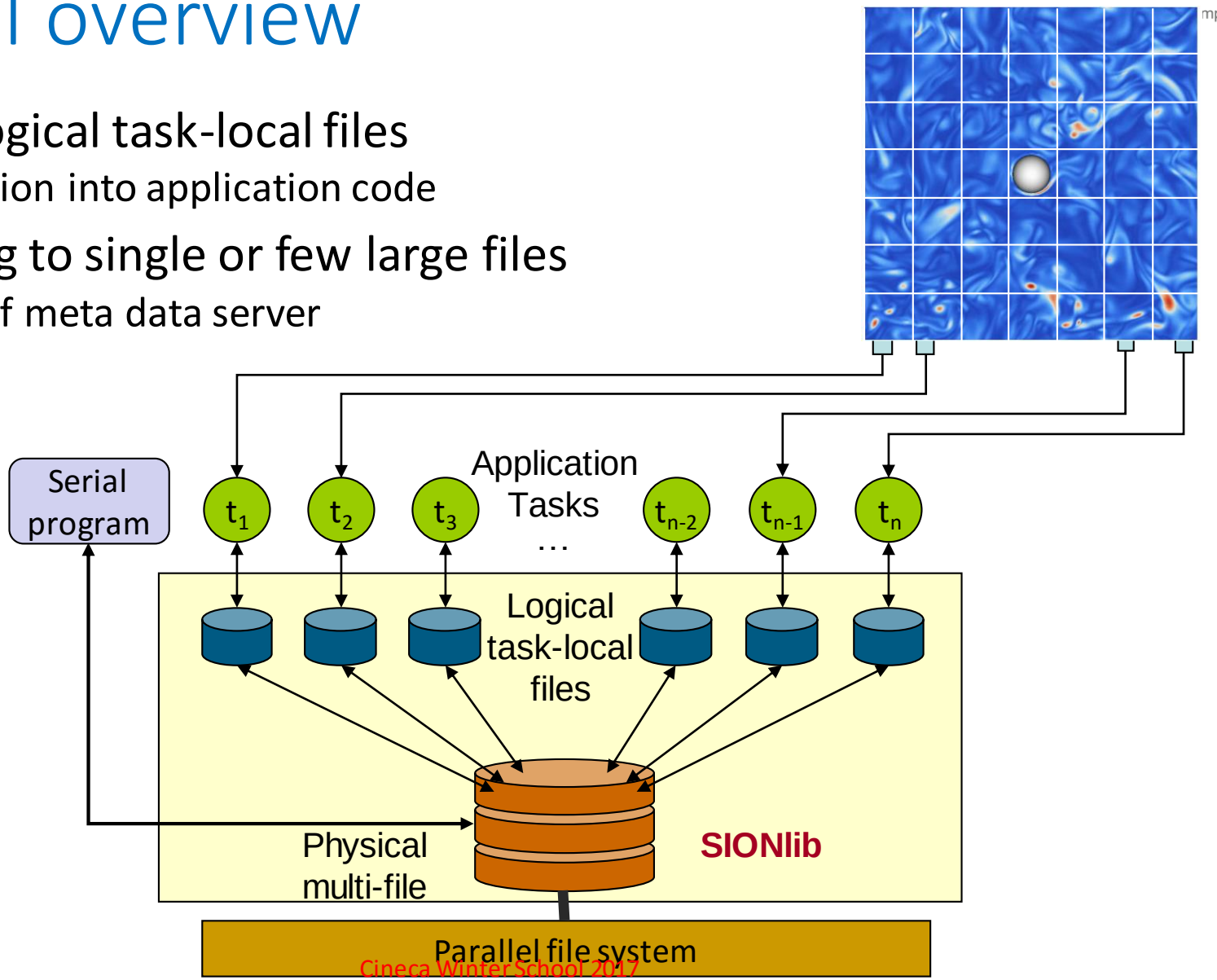
- . Would need to re-introduce MPI in order to re-enable internode communications (e.g. MPI_Bcast at the beginning and MPI_Reduce during the run).
- Alternatively, use the NAM which acts as a fast, shared memory for all the nodes in the SDV. At the moment, only low-level API available and a planned MPI_Reduce wrapper for the NAM will not be ready in time.
- Unfortunately, OmpSs cannot be used because $OMP threadprivate variables not supported

Resilience technologies and TURBORVB - SIONLib

- Library developed by Juelich Supercomputing Centre and and others to enable the reading and writing of binary data to/from many task-local files to one or a small number of files.
- Very useful if you have MPI programs where each rank writes/reads its own file (task local). Avoids having thousands of restart files.
- For parallel access, opening and closing files are collective calls but reads/writes can be asynchronous. Also possible to use a serial interface.
- Simple API which mimics usual C or FORTRAN I/O commands: **`fopen -> sion_open, fwrite -> sion_fwrite`**, etc.

SIONLIB API overview

- API resembles logical task-local files
  - Simple integration into application code
- Internal mapping to single or few large files
  - Reduces load of meta data server

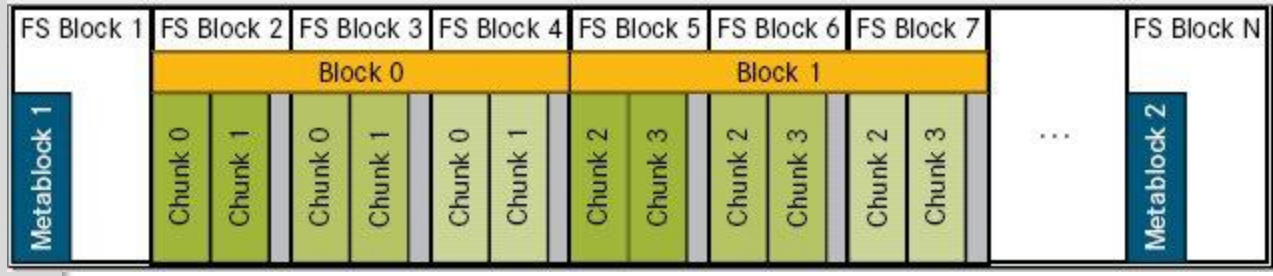


SIONlib API overview: code example

- For basic cases direct translation of serial task-local calls to calls using MPI
- Interfaces for MPI, OpenMP, hybrid (MPI + OpenMP) and generic communication
  - Individual communication call backs can be used

C

```
/* fopen() → */
sid=sion_paropen_mpi( filename , "bw",
                      &numfiles, &chunksize,
                      gcom, &lcom, ...);

/* fwrite(bindata,1,nbytes, fileptr) → */
sion_fwrite(bindata, 1, nbytes, sid);

/* fclose() → */
sion_parclose_mpi(sid)
```



chunksize – defines the block range in the file over which each task reads or writes. Used to ensure contention-free access. Normally set as the largest variable in a read/write.

SIONLib and TurboRVB

Original version of the code does not checkpoint – task-local restart files only written at the end of program.

First task – modify program to allow restarts (either task-local or Sionlib) to be dumped at user-selected intervals.

Then replace Fortran calls with SIONlib equivalents.

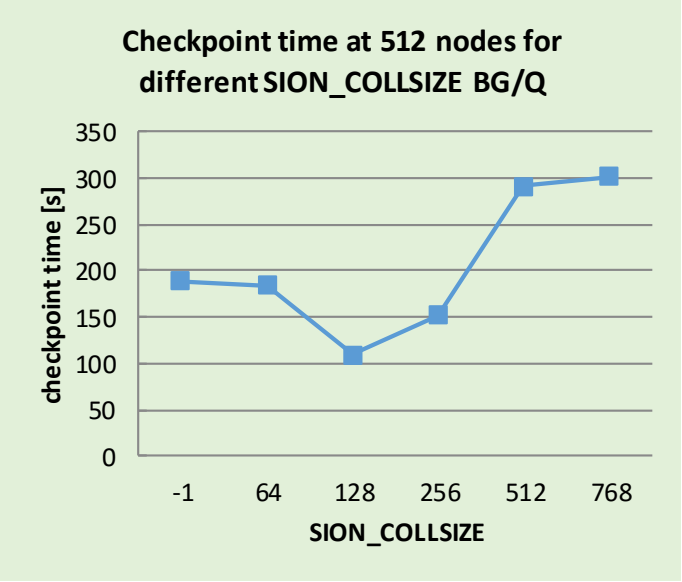
```
/
&checkpoint
idumpfreq=10
checkp_enabled=.true.
siondump=.false.
/
```

```
 call fsion_paropen_mpi(trim(checkpfilebase),
mode, nfiles, comm, lComm,chunksize, fsblksize,
rank, newfname, sid)
isize=kind(irece)
nelem=1
call fsion_write(irece,isize,nelem,sid,ierr)
call fsion_parclose_mpi(sid,ierr)
```

SIONlib and TURBORVB

- SIONlib utility provides compile and load flags for the make.
- Program is run as usual but env variables can control execution and provide debug info
  - SION_COLLSIZE
    - No. of collectors, i.e. no. of tasks involved in collecting data and writing the file (cf. MPI-IO).
  - SION_DEBUG
    - Debug information
- Other utilities also available
  - sionsplit, sioncat, siondump, siondefrag.

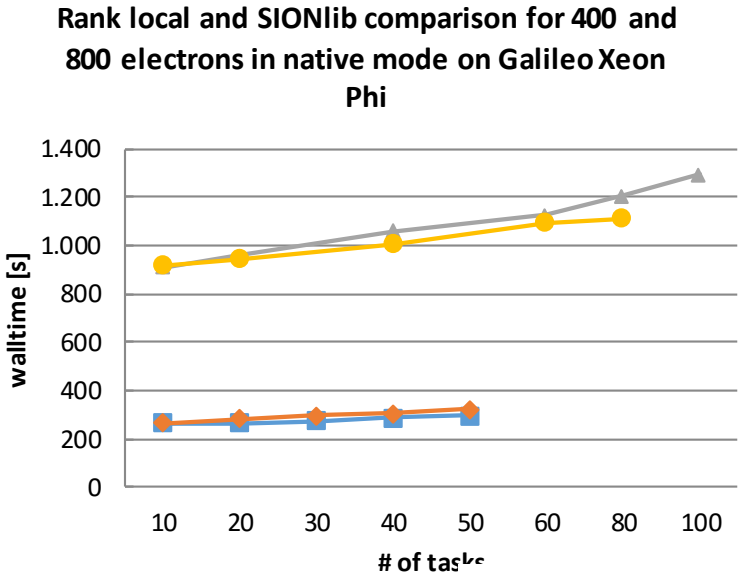
```
SIONCONFIG=/usr/local/bin/sionconfig
SION_FLAGS=$(shell $(SIONCONFIG) --mpi --cflags --f77 --32)
SION_CFLAGS=$(shell $(SIONCONFIG) --mpi --cflags --gcc --32)
SION_LIBS=$(shell $(SIONCONFIG) --mpi --libs --f77 --32)
```

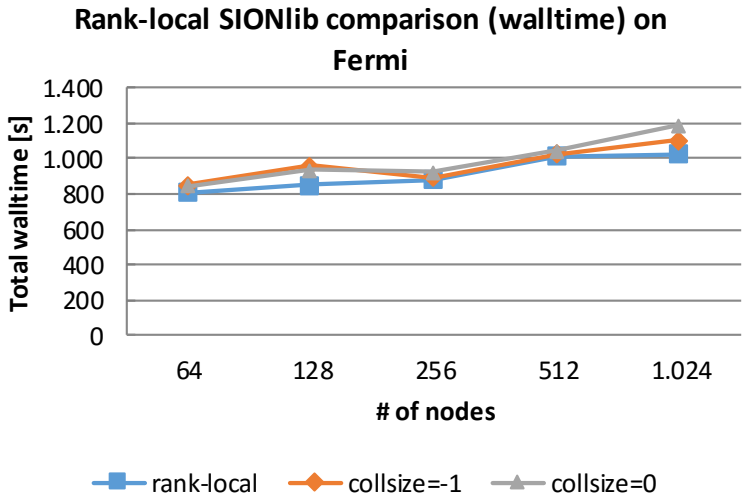


SIONlib and TURBORVB

- Turbo with Sionlib when tested on various clusters showed no particular performance speed-up when writing checkpoints, but since Turbo is not I/O intensive did not expect this.
- Main advantage for Turbo is to minimise the number of restart files needed, although some partners did report significant speed-ups.

Time taken to do `time ls` in a results directory for n=32768		
	Sionlib	Rank local
Real	0m0.003s	0m33.794s
User	0m0.001s	0m0.896s
Sys	0m0.002s	0m1.506s





SIONLib restarts

SIONLib on the other hand greatly reduces the time needed to perform restarts

	Direct SSD	GPFS SSD	GPFS
SIONlib	0.07	3.18	1.60
Rank local	1.11	551.00	515.00

	restart time/s	
nodes	Sionlib	Rank-local
1	0.43	18.01
2	0.47	144.53
3	1.97	299.29
4	1.35	508.78
5	2.42	501.32

Using SSD disks (PICO) can lead to further improvements in start up times.
Note that I/O results vary quite considerably so more correct to do many runs and estimate an average value.

Scalable Checkpoint and Restart (SCR)

In DEEP-ER based on the library developed at the Lawrence Livermore National Library (LLNR).

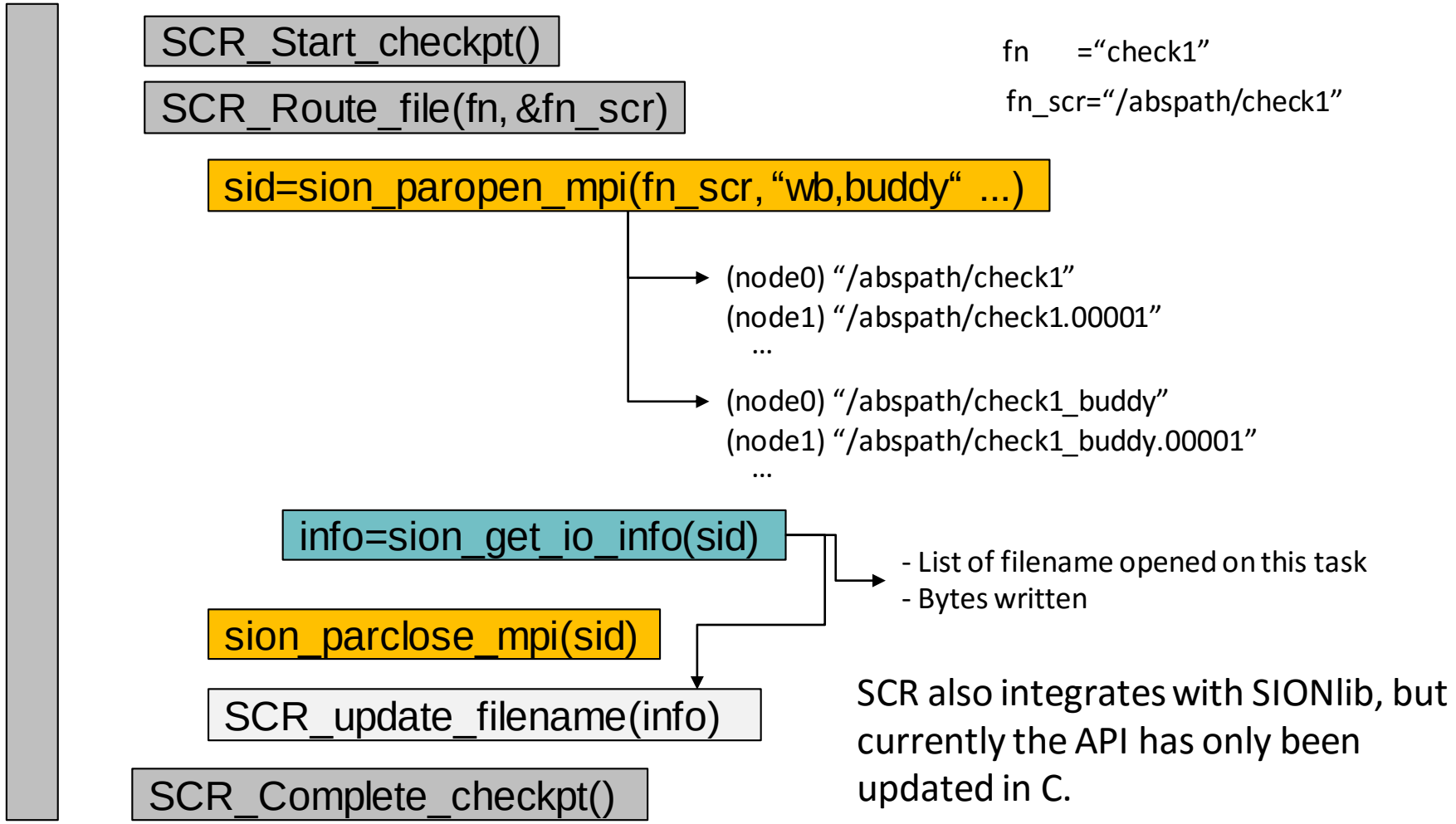
Based on observations that often only 1 node fails and that task-based checkpointing for example does not scale on global filesystems such as GPFS.

By inserting commands in the source on what is to be checkpointed, SCR can store checkpoints on local disks which can be automatically and transparently migrated to the GPFS (or other filesystem).

```
77
78 !   SCR checkpointing
79        call scr_start_checkpoint(ierr)
80        checkpfile_tmp=trim(checkpfile)//".ckpt"
81        call scr_route_file(checkpfile_tmp,checkpfile,ierr)
82        valid=1
83
84        open(icheckpunit,file=checkpfile, form='unformatted',status='unknown')
85
86        WRITE(icheckpunit) irece,irecmin,nw,in1
87
88 ! does write_cont have all the info ?
89
90        CALL write_cont(icheckpunit)
91        WRITE(icheckpunit) (wconf(i),i=ist,ien),(timerelease(i),i=1,in1)
92        IF(yesskip) THEN
93           WRITE(icheckpunit) wbraupo,jbraup,signwup,econfo,wro,signwupo
94        ENDIF
95        IF (rank.EQ.0) WRITE(*,*) 'Checkpoint: continuation file dumped&
96             & at step ',istep
97        close(icheckpunit)
98
99        call scr_complete_checkpoint(valid,ierr)
src/checkpoint.F90" 369L, 10516C
```

```
SCR_USER_NAME=deep70
SCR_JOB_NAME=cp
SCR_FLUSH=10 #SCR flush node-local dir to global fs every X
checkpoints
SCR_FETCH=1 # enable or disable fetching from global file system
during restart.
SCR_CACHE_SIZE=2
STORE=/tmp
STORE=/mnt/beeond COUNT=10 TYPE=BEEGFS
CKPT=1 STORE=/mnt/beeond INTERVAL=1 TYPE=SINGLE
```

SIONlib and SCR



SCR also integrates with SIONlib, but currently the API has only been updated in C.

Non-volatile Memory (NVM)

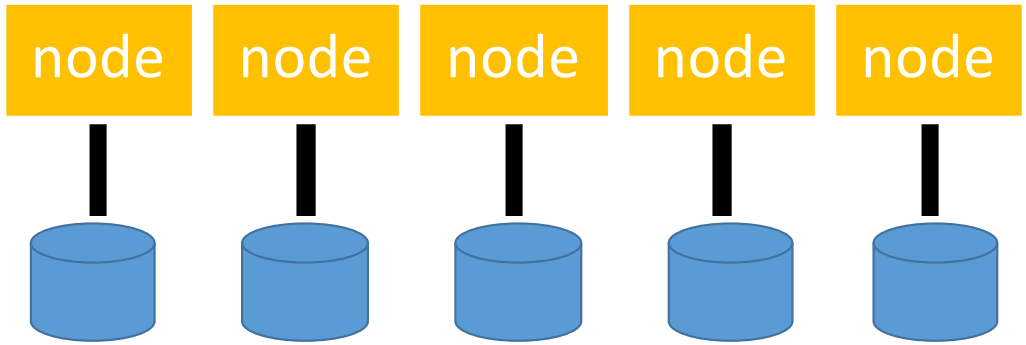
Memory than can retain information even after being powered-off.

Typical examples include Flash memory or standard hard disks (“spinning disks”) but usually these technologies are not suitable for replacing DDR RAM due to performance (disks) or low write endurance (Flash).

Now available NVM devices to replace disks in HPC clusters.

In DEEP-ER many SDV nodes have been equipped with NVM disks and NVMe for local storage. Can be combined with SCR. Aim is to connect to Booster nodes.

TurboRVB is not I/O intensive so only modest speedups obtained but other partners reported performance improvements.



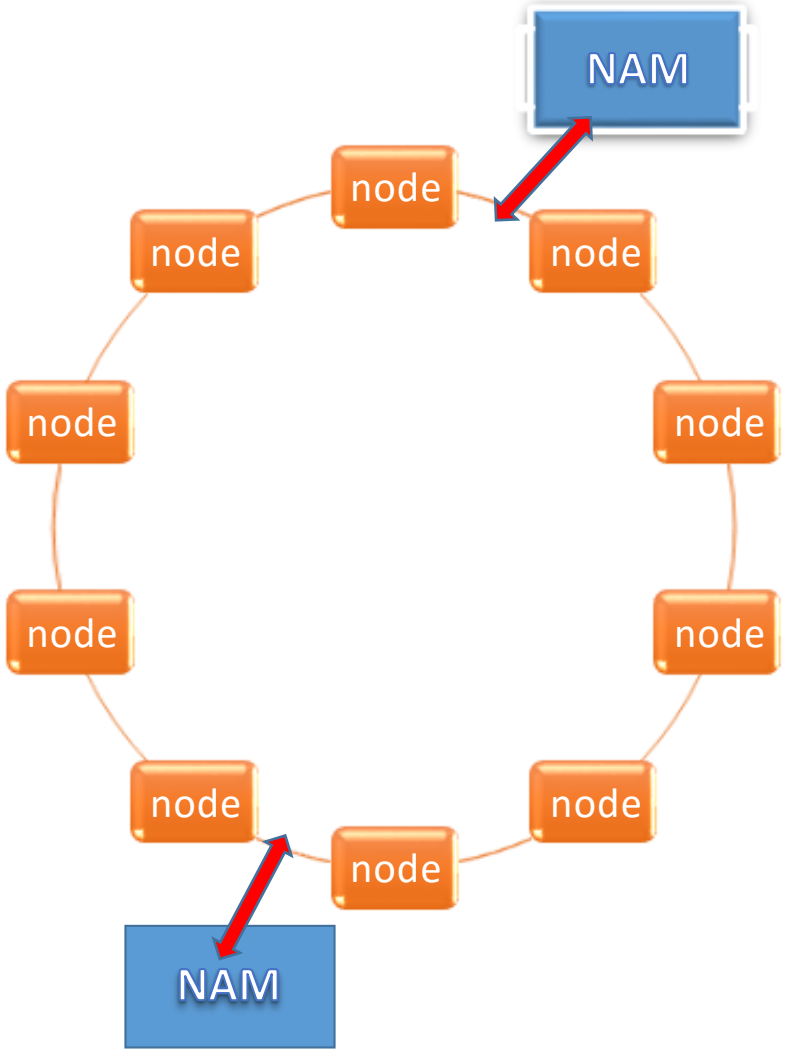
2 x CPUs	Intel Xeon E5-2680v3, 12-Core (Haswell)
RAM	128 GB (DDR4 , PC2133 ECC)
1 x SSD	400 GB NVMe Intel DC P3700
2 x HDD	1.0 TB S-ATA 6 Gb/s

Network Attached Memory (NAM)

Recent technology which allows memory to be directly attached to the network (as opposed to a node with a processor).

The idea is to reduce the HPC bottleneck of moving data around the system to be processed by processing data as it passes through the network → *near data computing*.

In the DEEP-ER project the NAM is based on PCIe with FPGA and a hybrid memory cube (4Gb in total).



NAM usage

As well as being memory, the NAM can perform limited operations.

For DEEP-ER the NAM is predicted to have two main uses:

1. **Checkpoints and Restarts**. The NAM can be used to store (incremental) system and checkpoints for restarting in case of processor fails.
2. For **MPI Global operations** such as SUM, MIN, MAX etc. (e.g. MPI_Reduce). The NAM thus acts as a sort of global, shared memory on which simple operations can be applied.

The complication is that the current API in DEEP-ER is based on low-level network (i.e. EXTOLL) GET and PUT requests. To simplify the life of C/FORTRAN programmers a library (libnam) is being developed.

```
double op[] = {1.0, 2.0, 3.0, 4.0, 5.0};
double result;
nam_allocation_t *my_alloc;
my_alloc = nam_malloc(sizeof(double) * 6);
nam_put
(op, 0, sizeof(double) * 5, my_alloc);
nam_reduce_op
( 0, 5, 5, &result, DOUBLE , SUM, NULL,
NAM_RETURN, my_alloc);
nam_free(my_alloc);
```

JUBE - Juelich Benchmark Environment

A Python/XML based system for systematically performing benchmarks developed by JSC.

http://www.fz-juelich.de/ias/jsc/EN/Expertise/Support/Software/JUBE/_node.html

The philosophy is that the platform-dependent configuration files (e.g. hardware details such as nodes/core or batch system) are separated from application and benchmark run parameters.

In this was easy to benchmark the same application on a different platform, or a different application on the same platform.

As well as running the benchmark runs also allows the application performance data to be analysed, collected and displayed.

JUBE - benchmarking

Particularly convenient the facility to automatically loop over any benchmark parameter (e.g. node, threads, etc) thus avoiding complicated loops in shell scripts.

Also useful the ability to define new variables and more complicated (python-based) scripted manipulations.

The main difficulty lies in the initial customization of the application and platform files.

this will run benchmarks for 1,2,4,6,8, and 10 nodes.

```
<!-- Job configuration -->
    <!-- no. of nodes, tasks, etc -->
    <parameterset name="systemParameter" init_with="platform.xml">
    <parameter name="taskspernode" type="int">64</parameter>
    <parameter name="ncores" type="int">68</parameter>
    <parameter name="mem" type="string">90GB</parameter>
    <parameter name="accountno" >cin_staff</parameter>
    <parameter name="executable">$mdrun mdrun </parameter>
    <parameter name="npme" mode="python" type="int">$tasks//2</parameter>
    <parameter name="args_executable">-v -s topol.tpr -npme $npme</parameter>
    <parameter name="nodes" type="int">1,2,4,6,8,10</parameter>
    <!-- <parameter name="tasks" type="int">2</parameter> -->
    <parameter name="timelimit">00:30:00</parameter>

    <parameter name="threads">1</parameter>
    <parameter name="ngpus"></parameter>
    <parameter name="mcdram">cache</parameter>
    <parameter name="OMP_NUM_THREADS" export="true">$threads</parameter>
    <parameter name="KMP_AFFINITY" export="true">scatter</parameter>
    <parameter name="I_MPI_PIN_MODE" export="false">lib</parameter>
    </parameterset>

    <!-- running the job, llsubmit, runjob , etc -->
    <parameterset name="executeset" init_with="platform.xml">
    </parameterset>

    <!-- files needed incl job file and input files -->
    <fileset name="jobfiles" init_with="platform.xml">
```

DEEP/DEEP-ER Summary

- Many of technologies used and developed in the project are freely available:
  - OmPSs
  - SIONlib
  - SCR
  - JUBE etc.
- The DEEP-ER Cluster/Booster itself is expected to be available via PRACE in late 2017.

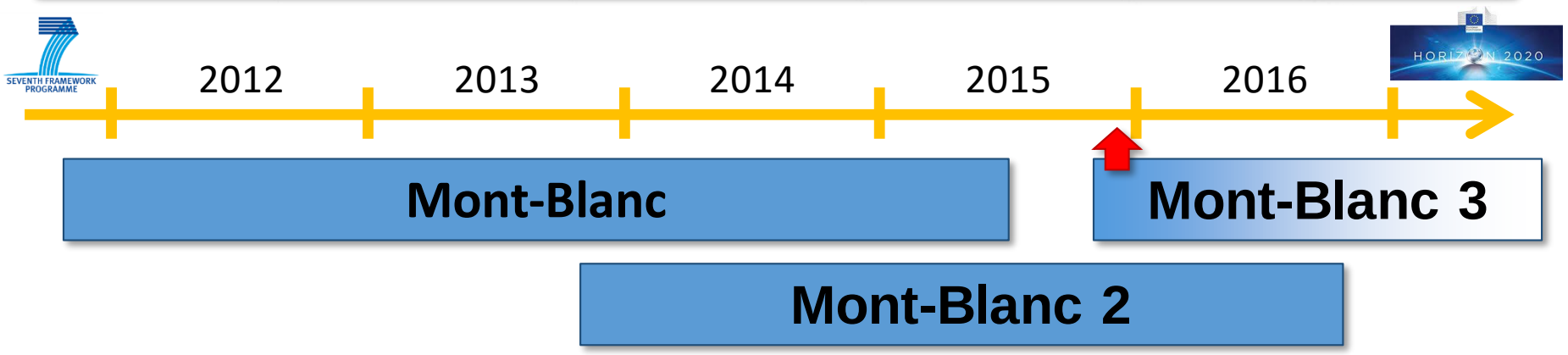
Mont Blanc - 1,2 and 3

Exascale project with the aim of building Energy efficient Exascale systems, for example using processors normally used in mobile systems such as ARM .

Extensive use of OmPSs as the programming model.

Mont-Blanc projects in a glance

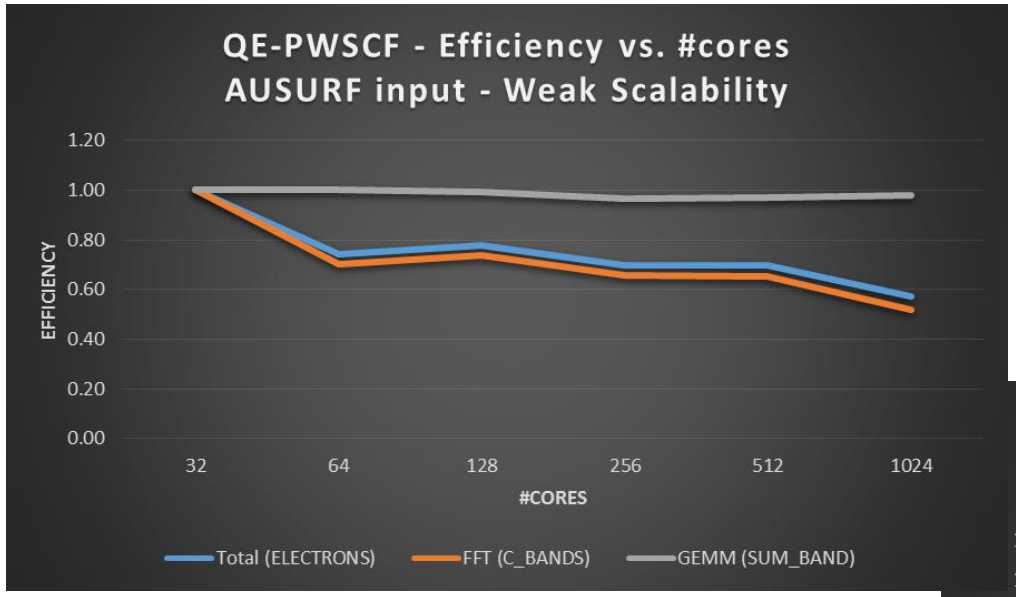
Vision: to leverage the fast growing market of mobile technology for scientific computation, HPC and non-HPC workload.

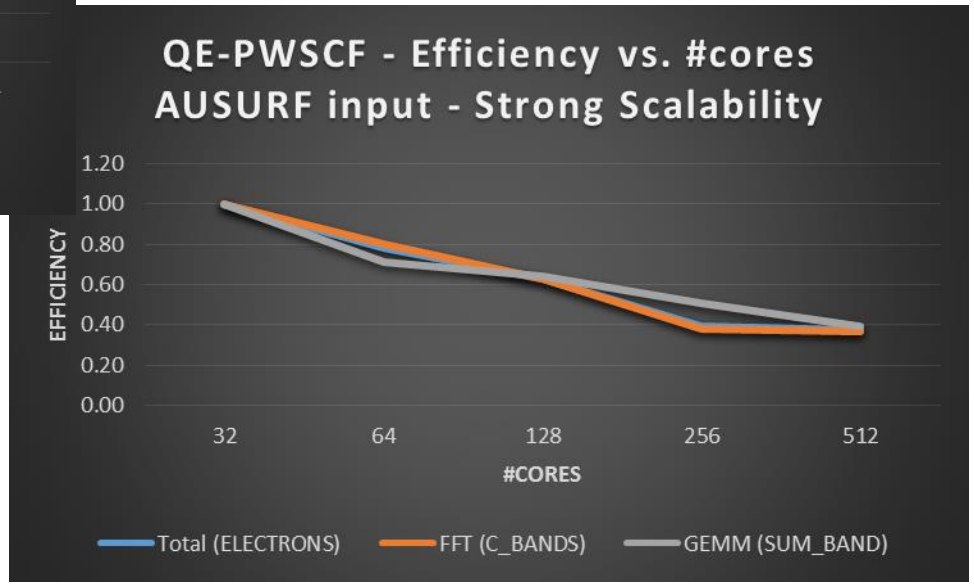


Barcelona Supercomputing Center
Centro Nacional de Supercomputación
allinea
UNI GRAZ
Bull atos technologies
ARM
JÜLICH FORSCHUNGSZENTRUM
CINECA
Inria INVENTEURS DU MONDE NUMÉRIQUE
University of BRISTOL
Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities
HLRS
ETH Zürich
cea
UC UNIVERSIDAD DE CANTABRIA
AVL
UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES
cnrs dépasser les frontières

QuantumESPRESSO

Weak & Strong scalability on MB prototype – ***OmpSs(SMP)***

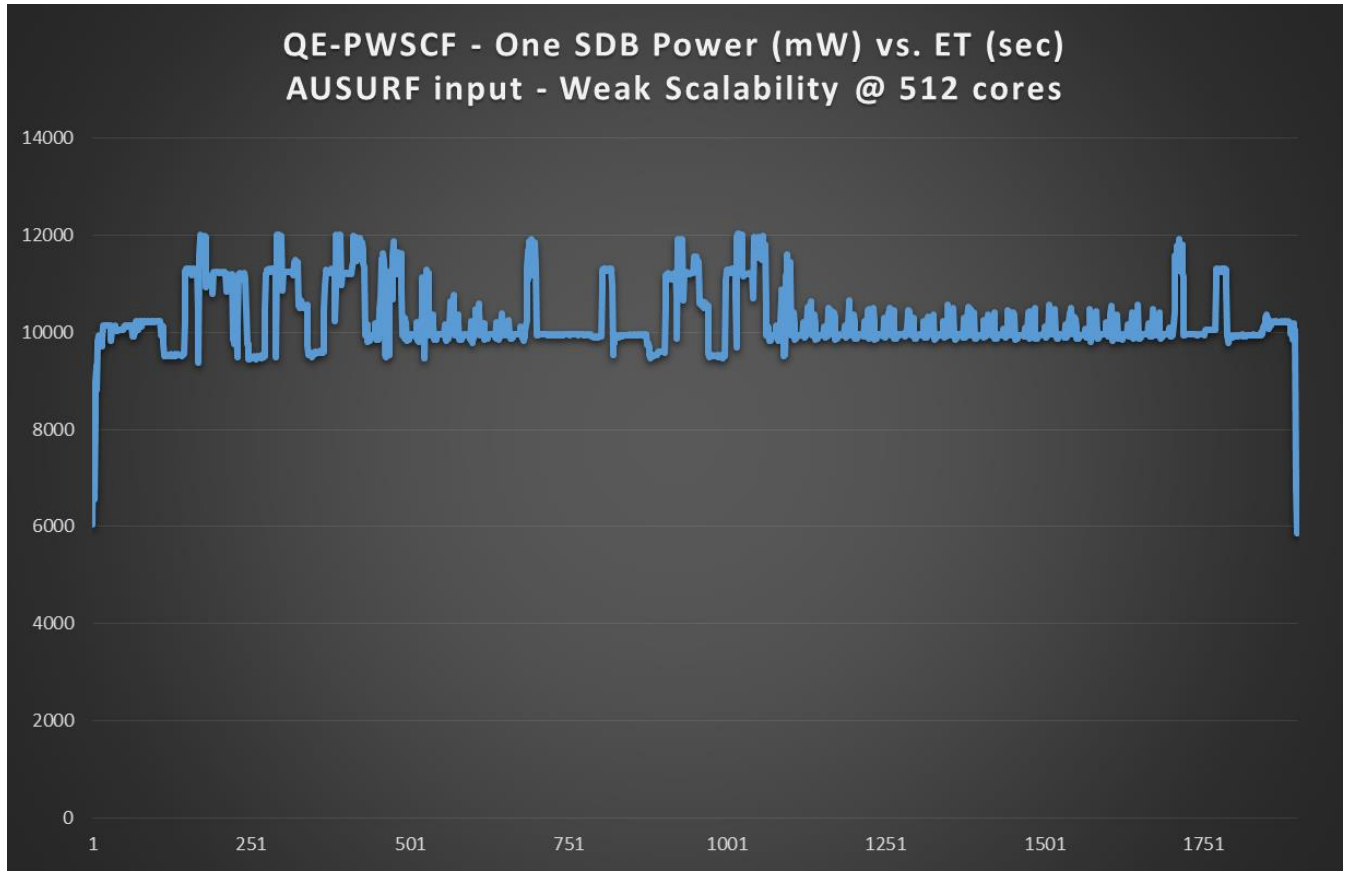




- ✓ <60% efficiency above ca. 512 cores
- ✓ With the new ETH driver:
  - ✓ ***Higher limit in scalability***
  - ✓ ***2-2.5x better perfomance***

QuantumESPRESSO

Power consumption on MB prototype – ***OmpSs(SMP)***



- ✓ Reproducible power consumption mesurement up to 512 SDBs
- ✓ 5.5 kJ @ 256 SDBs
- ✓ ***E2S decrease as #core increase***

Oprecomp – OPen TransPrecision COMPuting

Again main emphasis to reduce the energy required in order to do computing.

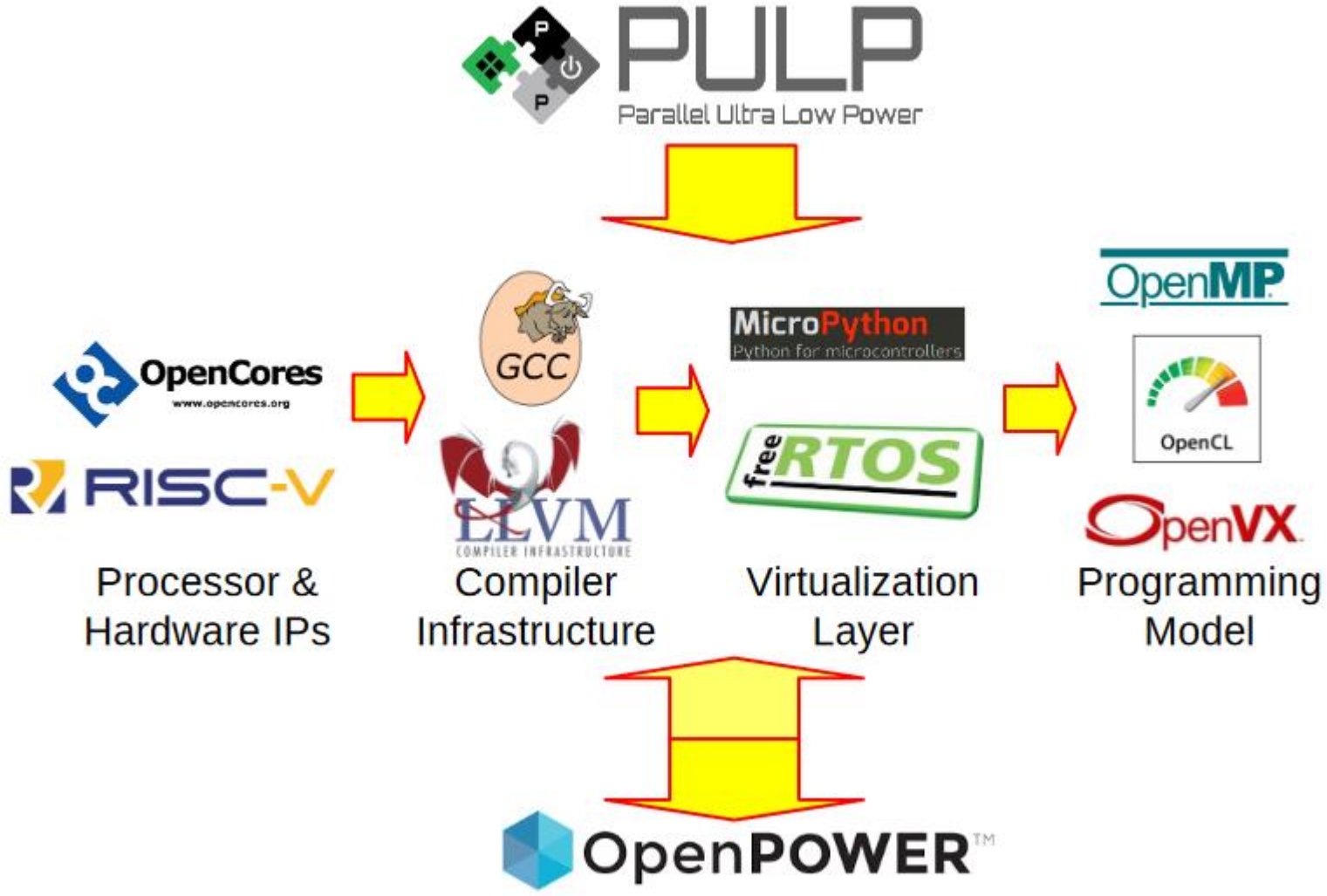
But here among some of the key ideas will be the use of *transprecision computing,* in other words using appropriate floating point accuracy (not only double precision!).

Algorithms and (micro)benchmarks will be tested on two different classes of hardware:

- mW – PULP processors
- MW – IBM Power

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732631.

Oprecomp – hardware + software



PULP – Parallel Ultra Low Power Processor.
32 bit chip RISC-V core developed at ETH Zurich and Universita' di Bologna.

Normally used in embedded systems: sensors, low-resolution cameras, accelerometers etc.

Summary

Many Exascale projects are concentrating on reducing the power required to perform HPC, combining innovative hardware and software approaches.

The important feature relies on *co-design*, develop hardware, middleware and application software together.

For application developers good software design is important: in the DEEP/ER projects much time was lost re-factoring code.

Communication is important, particularly between hardware and software engineers.