Enabling the Quantum Collisions 'PFARM' code on Xeon Phi

Refactoring MPI communications for Symmetric Mode

Michael Lysaght[1], Andrew Sunderland[2], Martin Plummer[2]

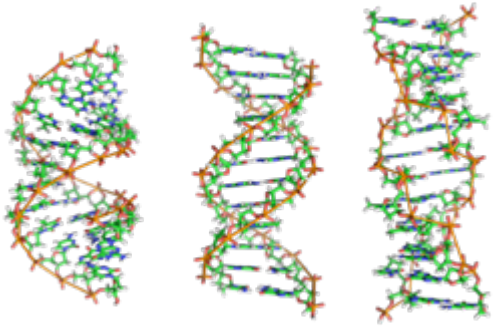
[1]Irish Centre for High End Computing (ICHEC), Ireland

[2]STFC, Daresbury, UK

The CCPQ electron-atom and electron-molecule code suites PRMAT and UKRMol

- **Used to simulate physics of Quantum Collisions**

- **Scientific applications include data for:**
  - Studies on DNA strand-breaking
  - Modeling of fusion plasmas
  - Modeling of astrophysical and industrial plasmas



- **Physics described by R-matrix Theory**
  - developed by Wigner & Eisenbud, developed for atomic physics by P. G Burke and colleagues

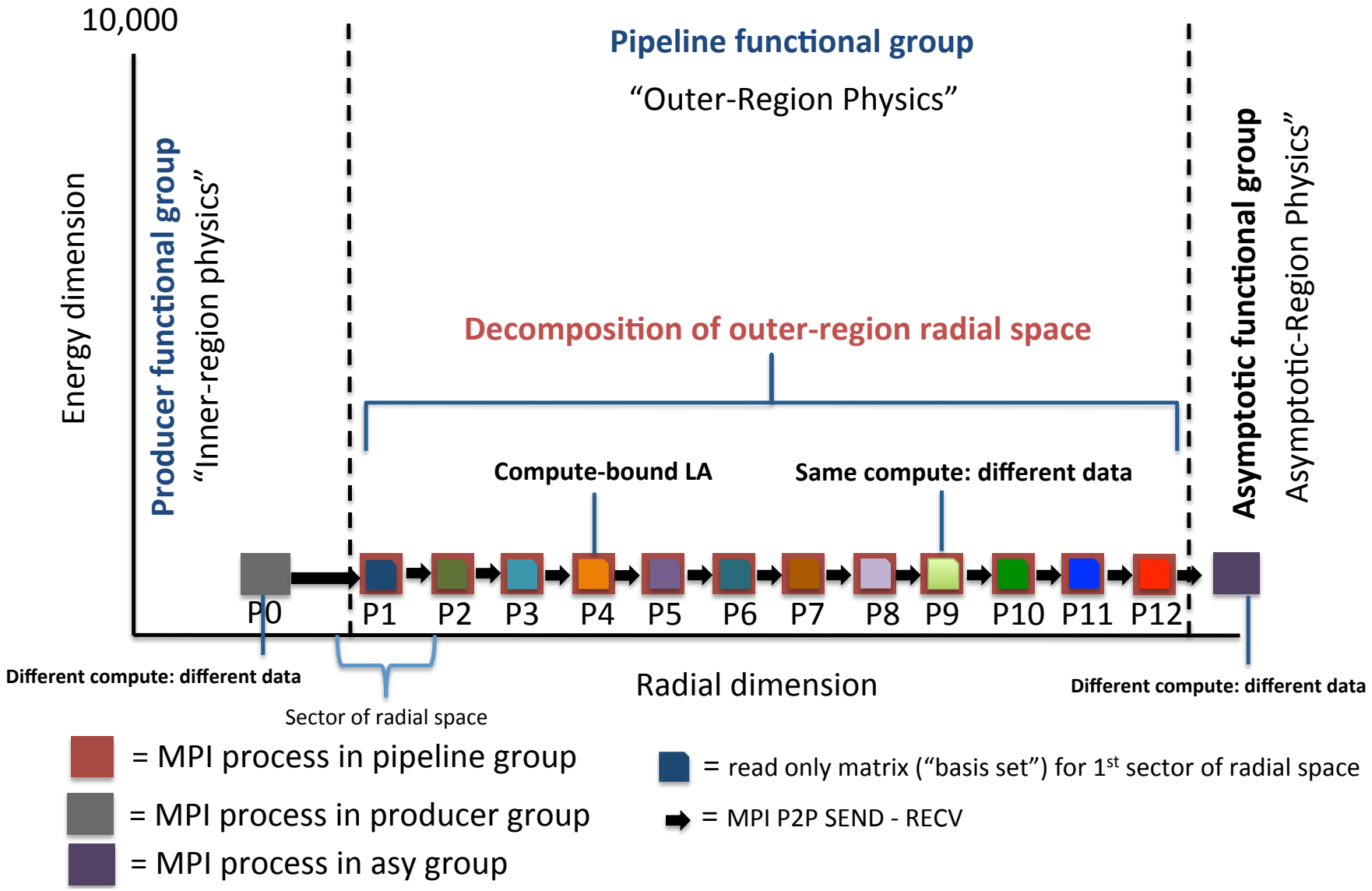
- **Codes developed variously at Queen's University Belfast, University College London, Royal Holloway London, The Open University, STFC and internationally**

- **PFARM, common to PRMAT and UKRMol, is a PRACE Benchmark Code**
  - Shown to scale well on up to ~30k cores (IBM BGQ)
  - Enabling work on EXDIG stage has resulted in significant speedup on Xeon Phi in offload mode

- **Developed using Fortran & MPI (since mid-90s), plus OpenMP, CUDA, MAGMA-MIC**

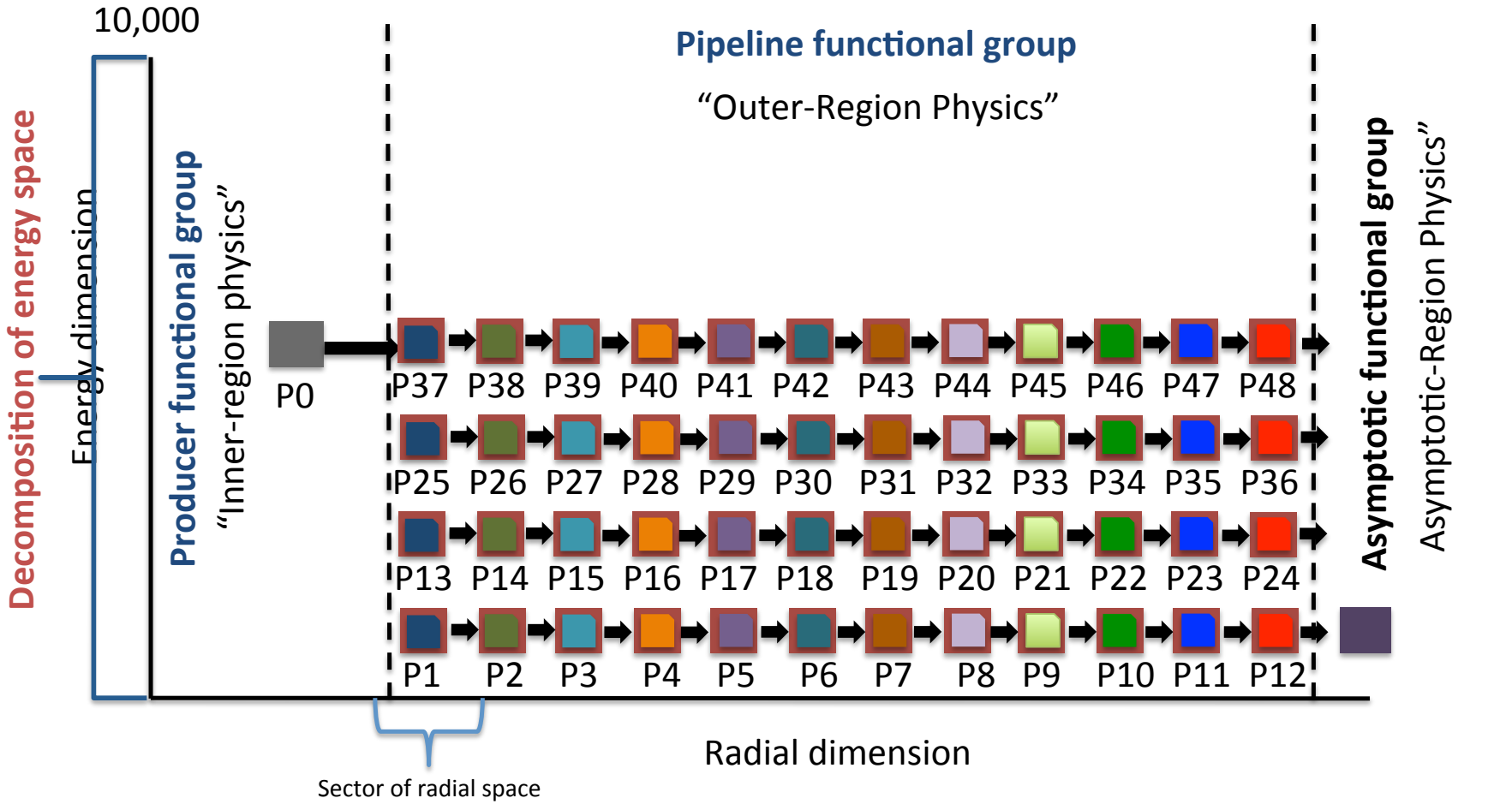
- **Available for download from CCPForge**

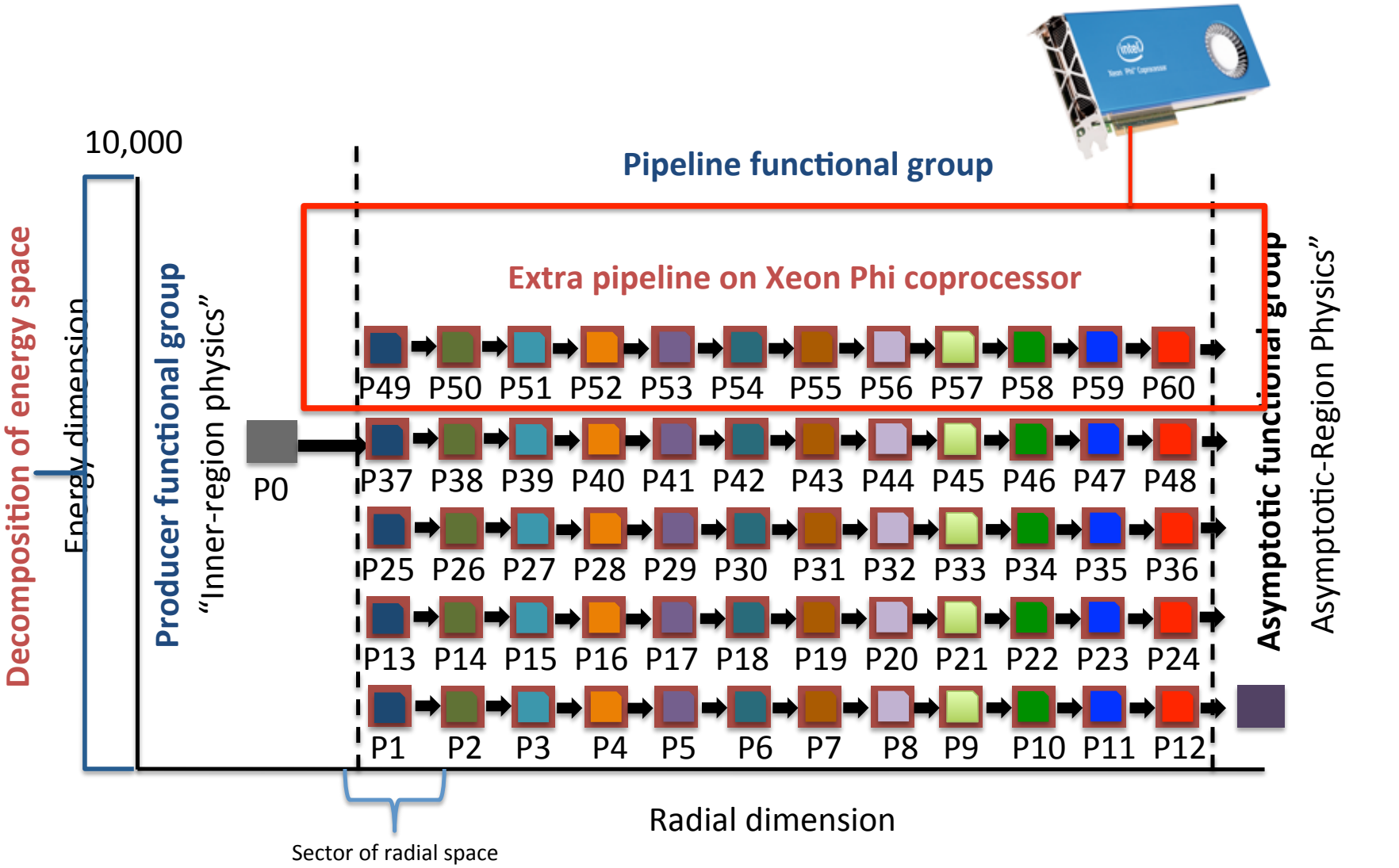
Solving coupled PDEs in multi-dimensional configuration space



How can we exploit Xeon Phi Coprocessor with this design?

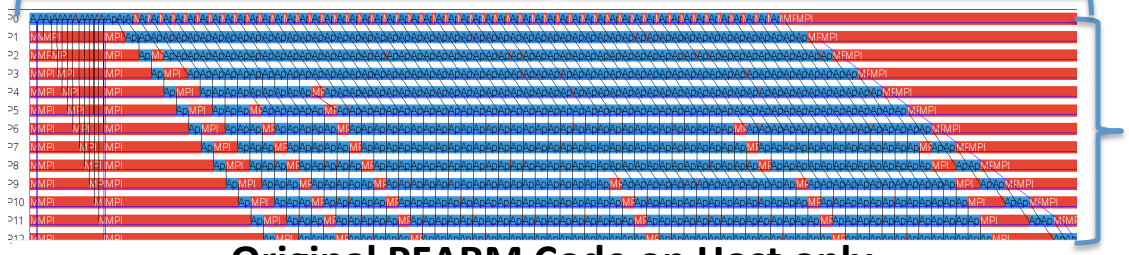
High core count on Xeon Phi suggests more pipelines possible!




10,000
Pipeline functional group
Extra pipeline on Xeon Phi coprocessor
Decomposition of energy space
Energy dimension
Producer functional group
"Inner-region physics"
Asymptotic functional group
Asymptotic-Region Physics"
P49 P50 P51 P52 P53 P54 P55 P56 P57 P58 P59 P60
P0
P37 P38 P39 P40 P41 P42 P43 P44 P45 P46 P47 P48
P25 P26 P27 P28 P29 P30 P31 P32 P33 P34 P35 P36
P13 P14 P15 P16 P17 P18 P19 P20 P21 P22 P23 P24
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12
Radial dimension
Sector of radial space
= MPI process in pipeline group
= MPI process in producer group
= MPI process in asy group
= read only matrix ("basis set") for 1st sector of radial space
= MPI P2P SEND - RECV

Exploiting ITAC for Deeper Insight

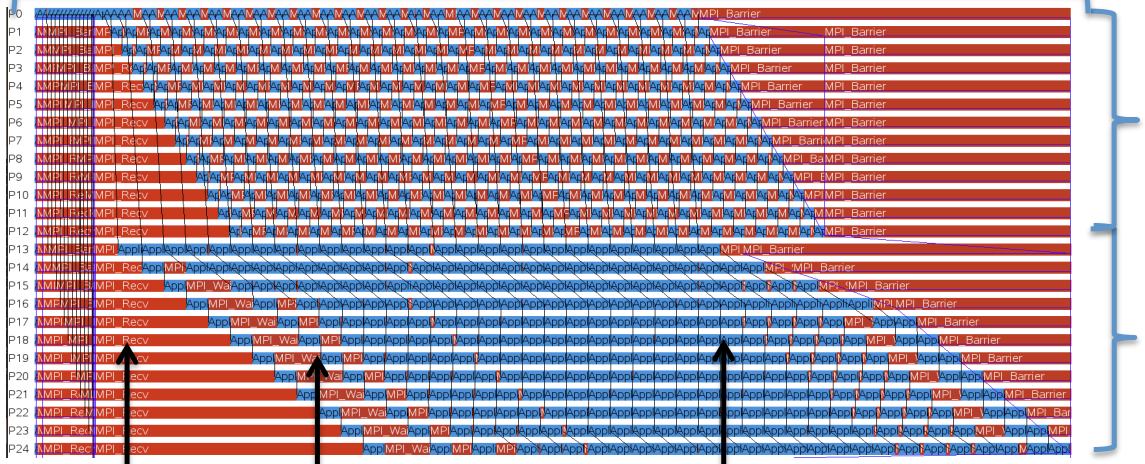
Time-to-solution: 546 secs



Host Processes (0-12)
Single pipeline

Original PFARM Code on Host only

Time-to-solution: 727 secs



Host Processes (0-12)
1st pipeline

Xeon Phi Processes (13-24)
2nd pipeline

Original PFARM Code Host + 1 x Xeon Phi Coprocessor

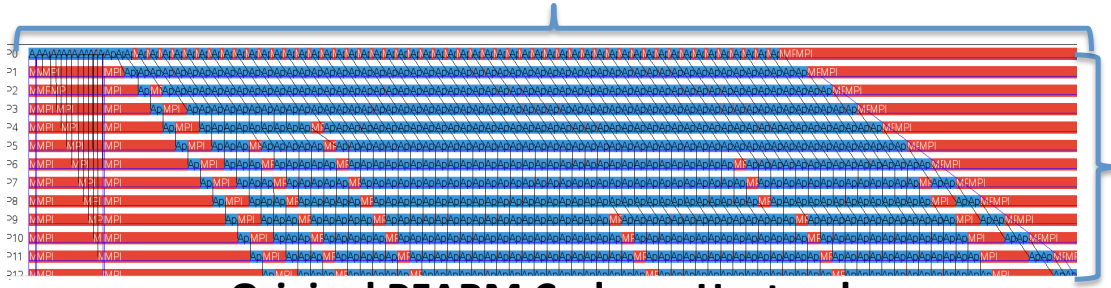
Compute phase takes longer on Xeon Phi

Higher overhead of MPI comms and MPI_WAIT on Xeon Phi

Poor load-balancing between Host and Xeon Phi

Exploiting ITAC for Deeper Insight

Time-to-solution: 546 secs



= producer process (fast compute)

= 1st pipeline stage process (fast compute)

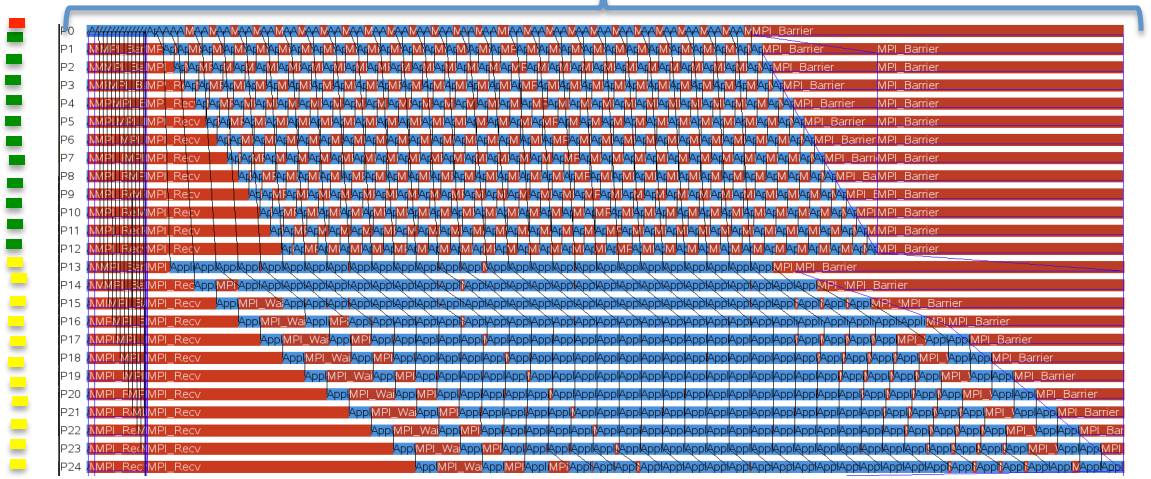
= 2nd pipeline stage process (slow compute

= fast MPI comms (on Xeon)

= slow MPI comms (on Xeon Phi)

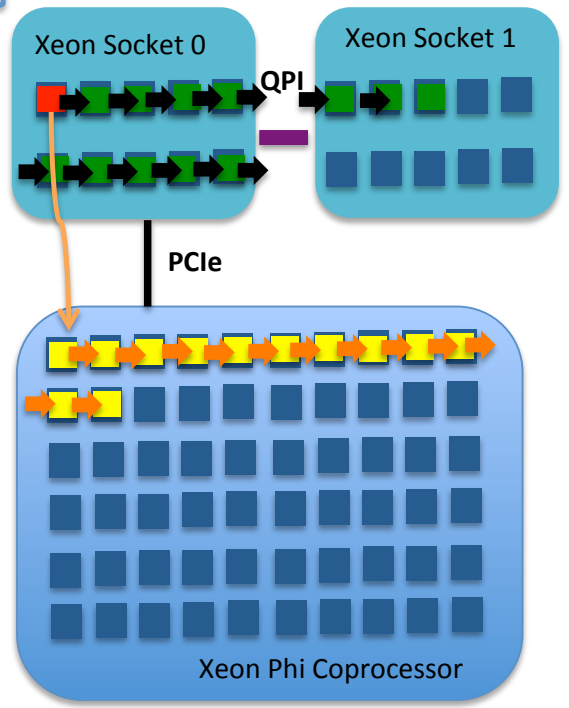
Original PFARM Code on Host only

Time-to-solution: 727 secs

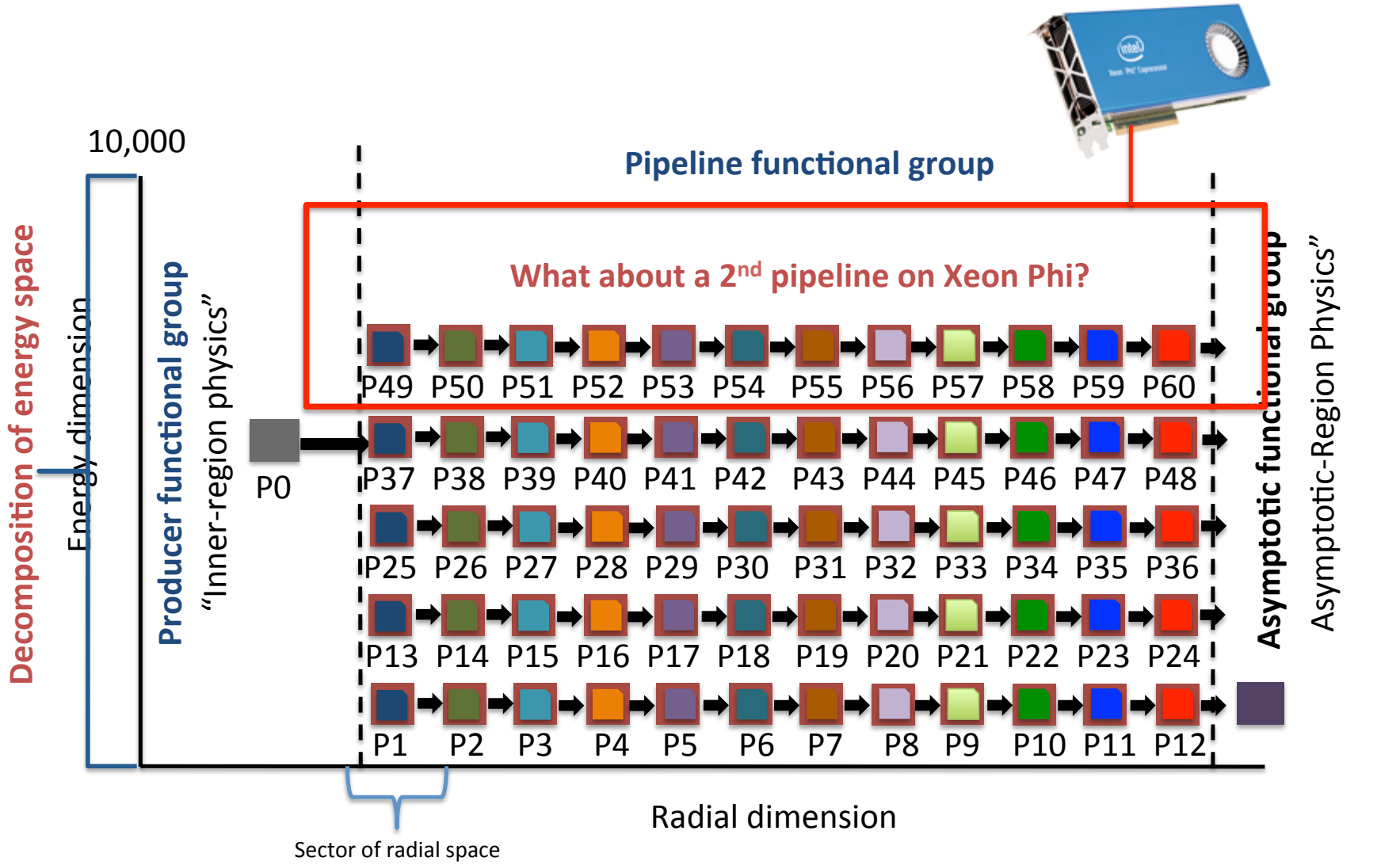


Original PFARM Code Host + 1 x Xeon Phi Coprocessor

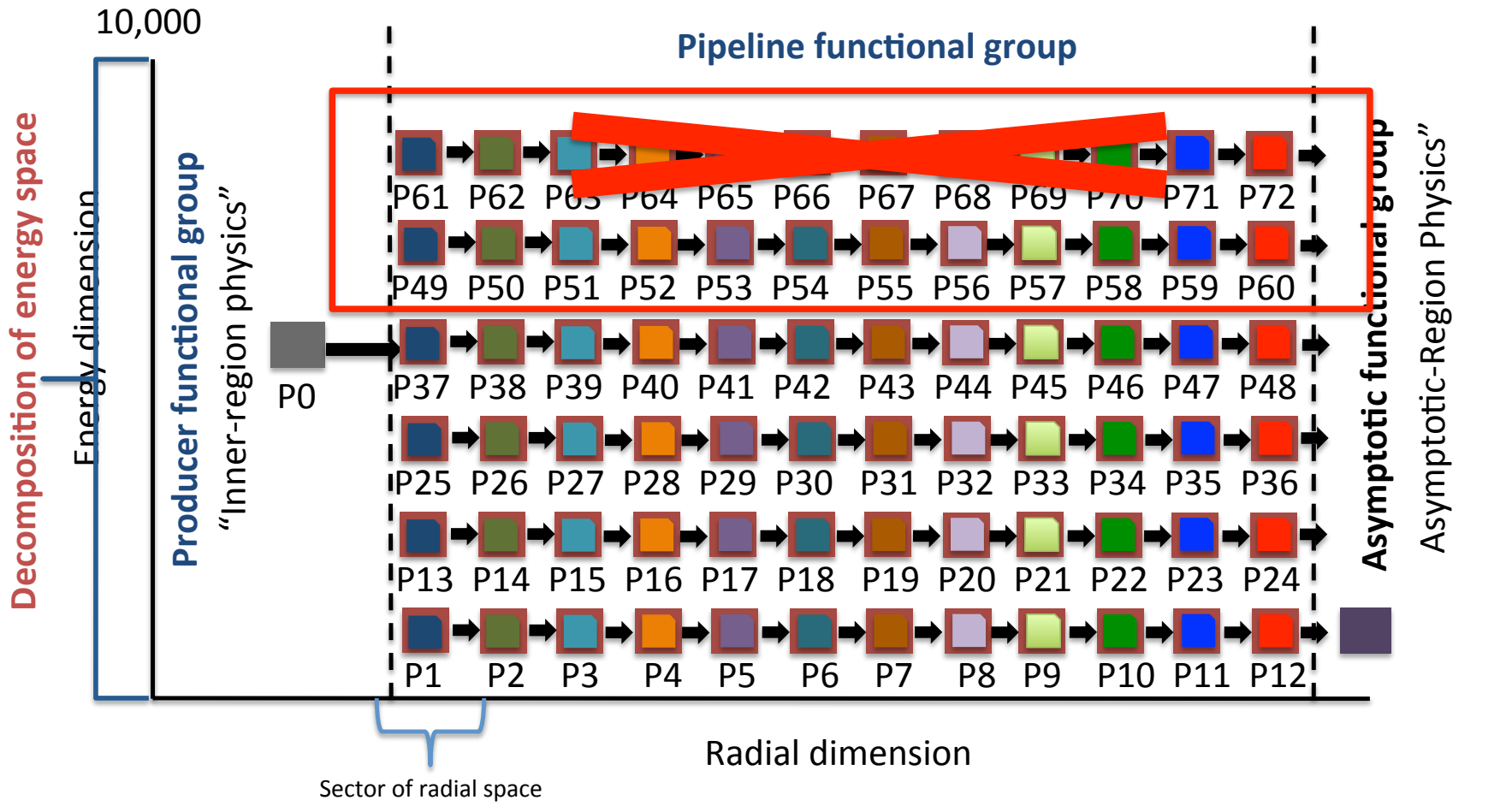
Sees resources as a homogeneous system



ITAC results strongly suggest that the only way to benefit from Xeon Phi is to increase the number of pipelines on the Xeon Phi – 1 pipeline is not enough!


10,000
Pipeline functional group
What about a 2nd pipeline on Xeon Phi?
Decomposition of energy space
Energy dimension
Producer functional group
"Inner-region physics"
Asymptotic functional group
Asymptotic-Region Physics"
P0
P49 P50 P51 P52 P53 P54 P55 P56 P57 P58 P59 P60
P37 P38 P39 P40 P41 P42 P43 P44 P45 P46 P47 P48
P25 P26 P27 P28 P29 P30 P31 P32 P33 P34 P35 P36
P13 P14 P15 P16 P17 P18 P19 P20 P21 P22 P23 P24
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12
Radial dimension
Sector of radial space
= MPI process in pipeline group
= MPI process in producer group
= MPI process in asy group
= read only matrix ("basis set") for 1st sector of radial space
= MPI P2P SEND - RECV

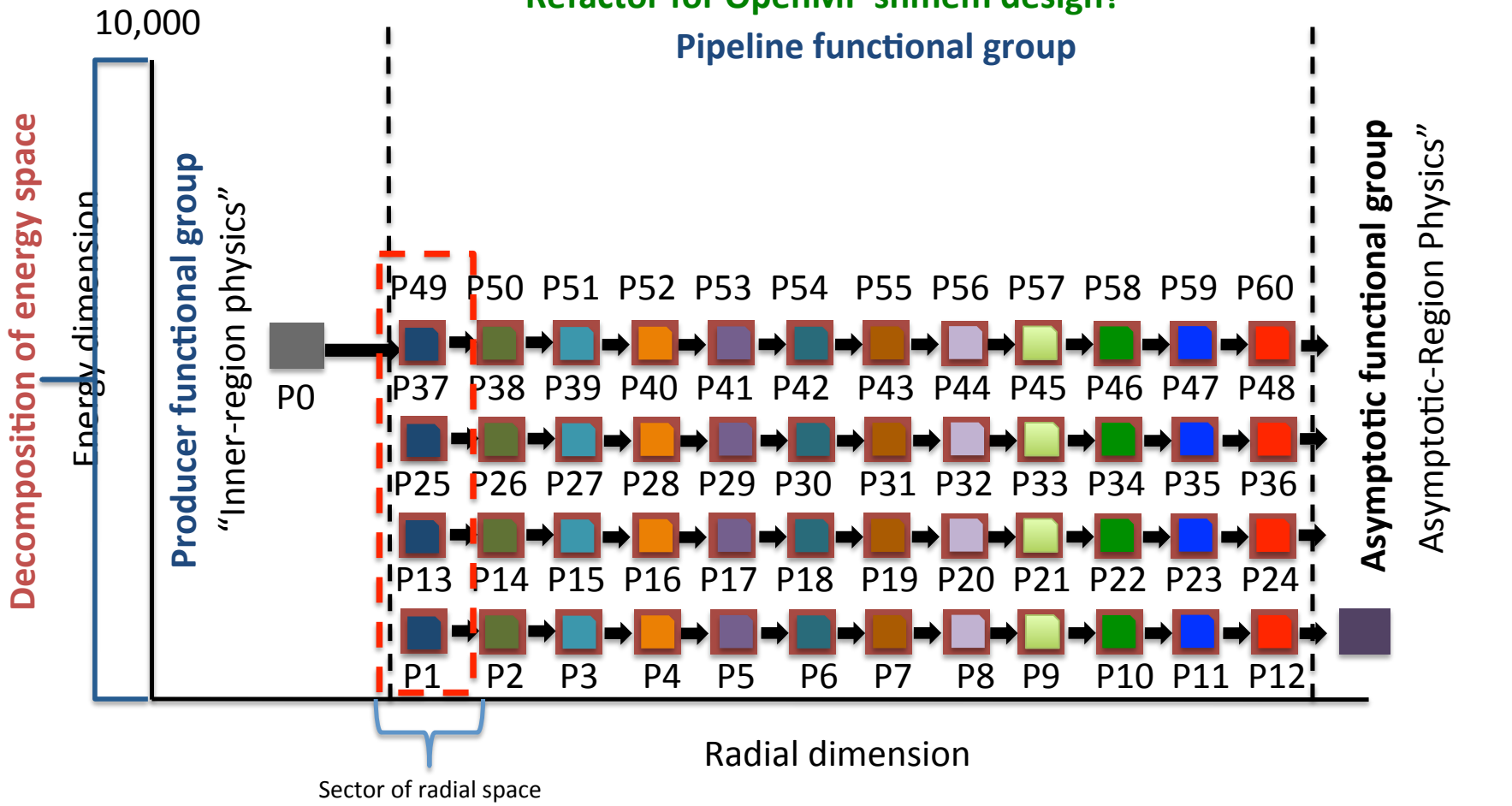
Problem: Memory footprint grows too rapidly with increase in pipeline count!



Can we share the replicated data across parallel tasks?

MPI 3.0 approach = tricky

Refactor for OpenMP shmem design?



= MPI process in pipeline group

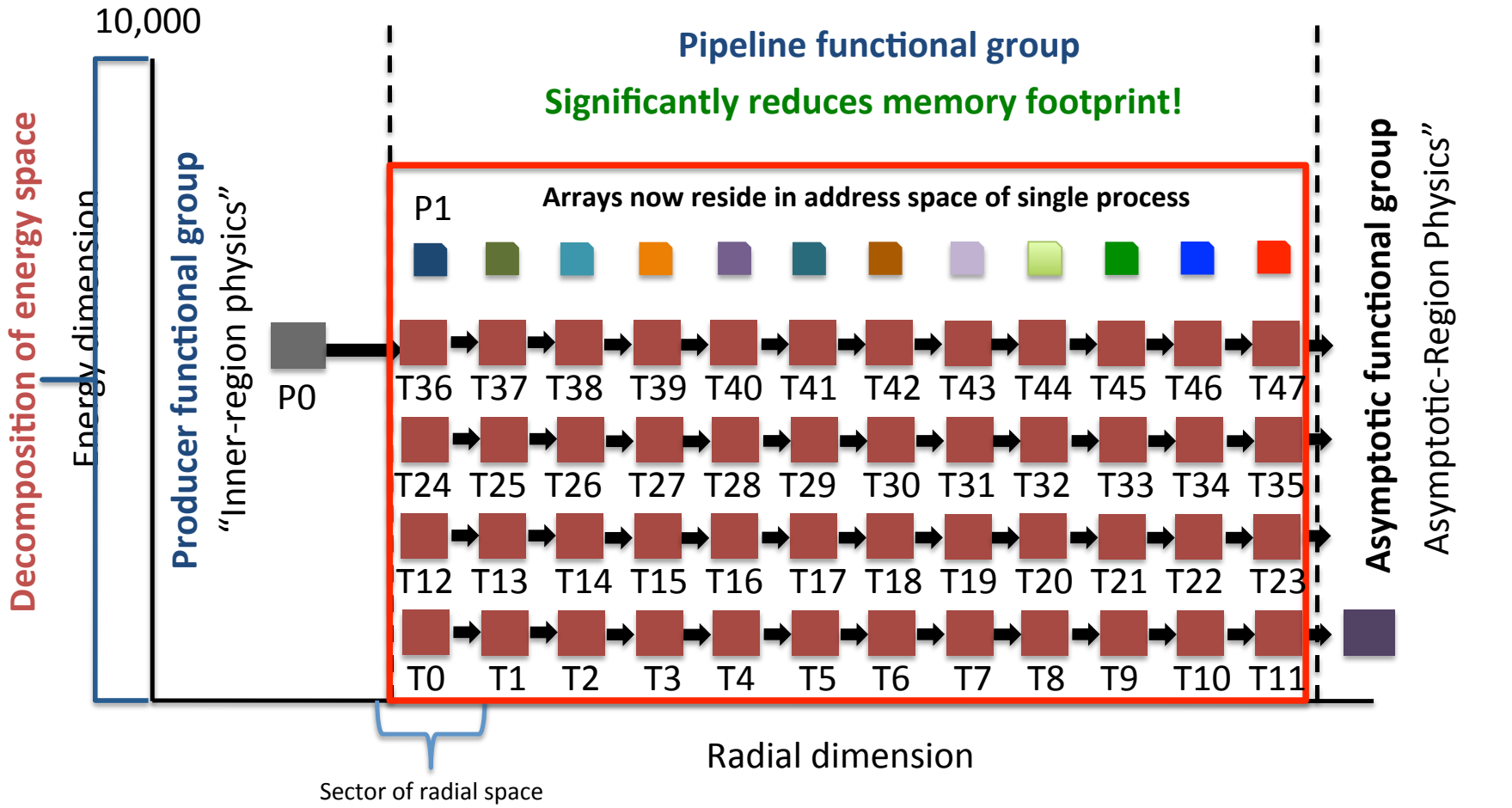
= MPI process in producer group

= MPI process in asy group

= read only matrix ("basis set") for 1st sector of radial space

= MPI P2P SEND - RECV

Can we share the replicated data across parallel tasks?
Refactoring for OpenMP shmem design



= OpenMP Task in pipeline group

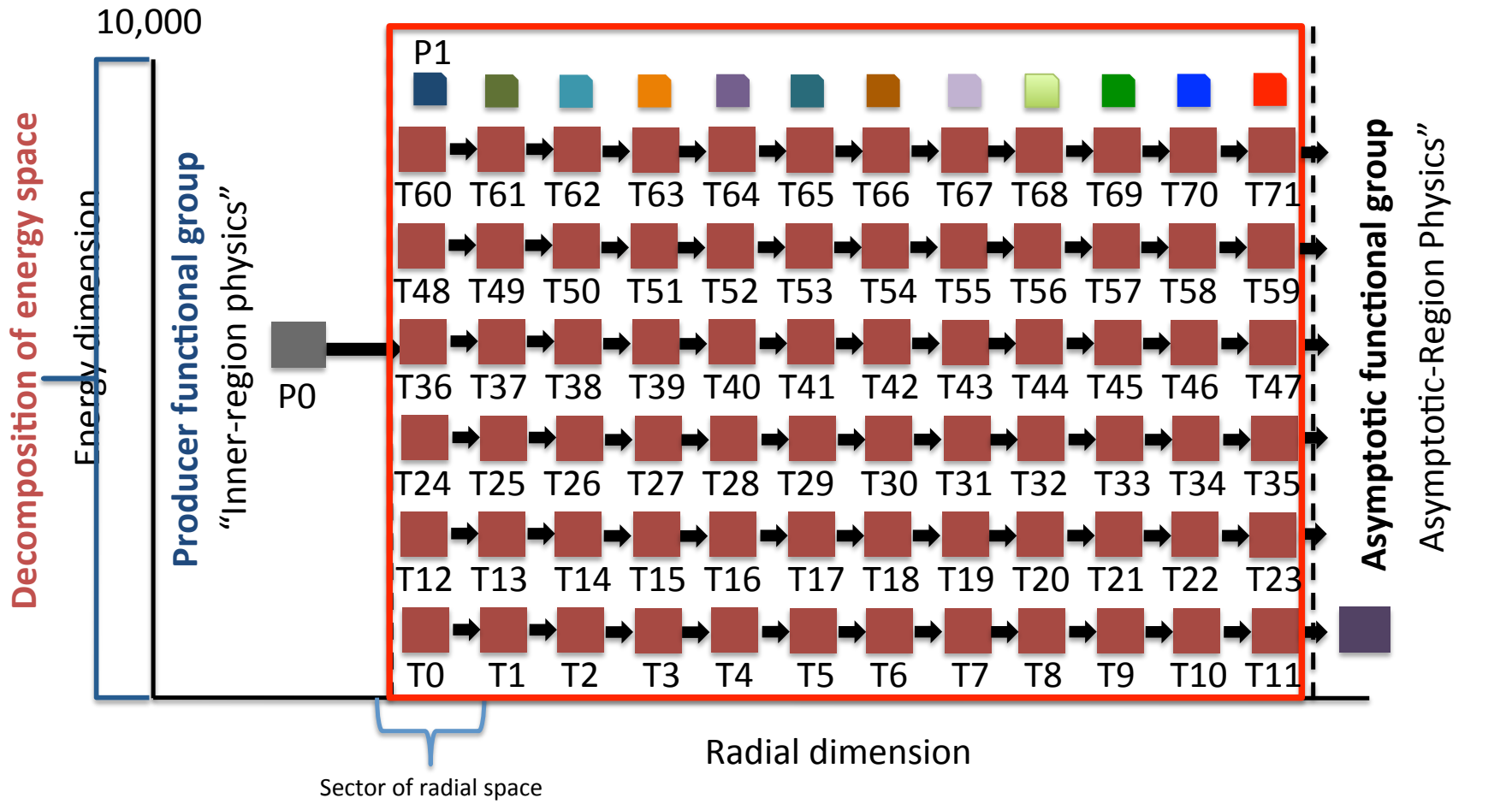
= MPI process in producer group

= MPI process in asy group

= read only matrix ("basis set") for 1st sector of radial space

= task dependency

Can we share the replicated data across parallel tasks?
Refactoring for OpenMP shmem design



= OpenMP Task in pipeline group

= MPI process in producer group

= MPI process in asy group

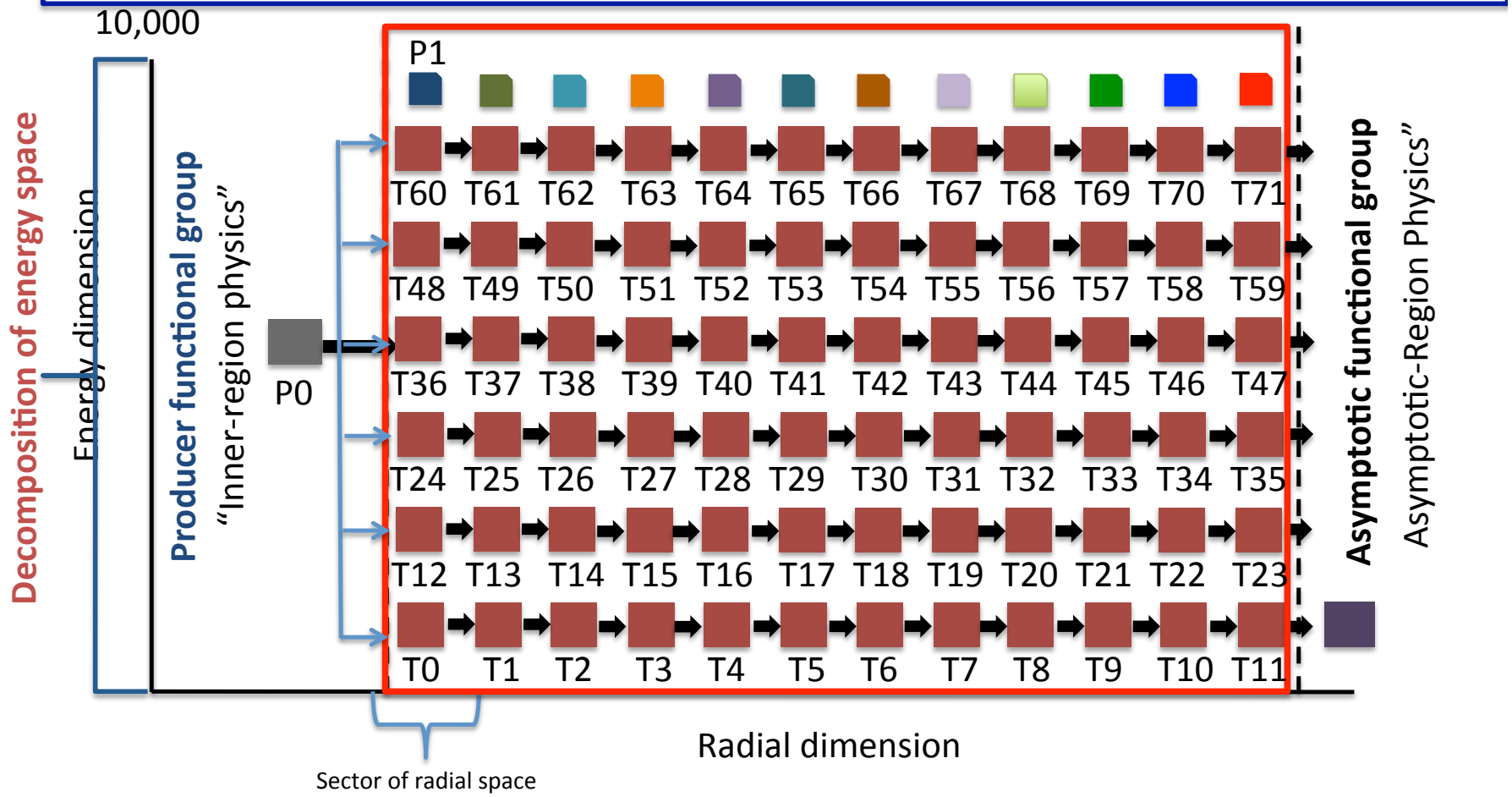
= read only matrix ("basis set") for 1st sector of radial space

= task dependency

Solution: Enable MPI_THREAD_MULTIPLE support

OpenMP tasks/threads can each call MPI_RECVs from within OMP Parallel Region (on P1)

Can maintain the MPI-based Producer/Consumer model at interface of functional groups



= OpenMP Task in pipeline group

= MPI process in producer group

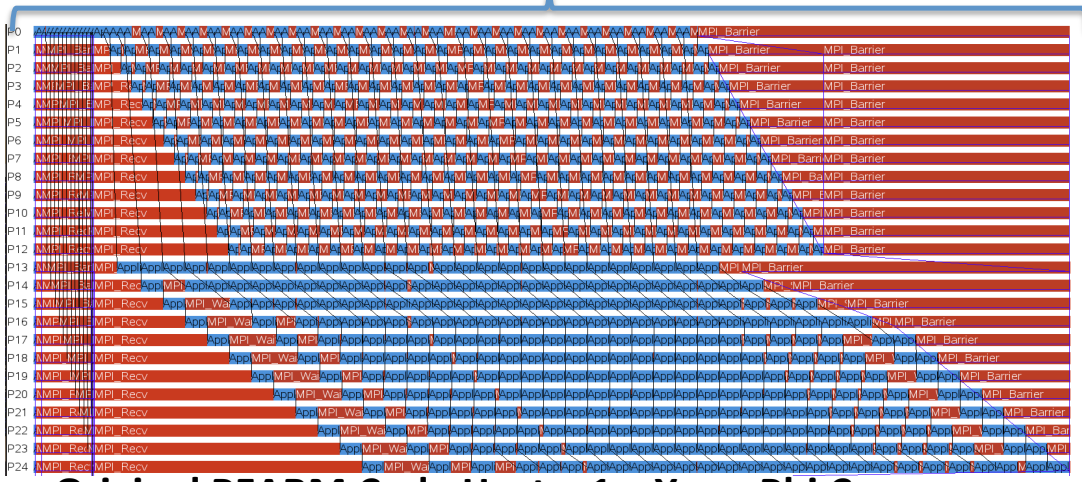
= MPI process in asy group

= read only matrix ("basis set") for 1st sector of radial space

= task dependency

Results and Deeper Insight from ITAC

Time-to-solution: 727 secs

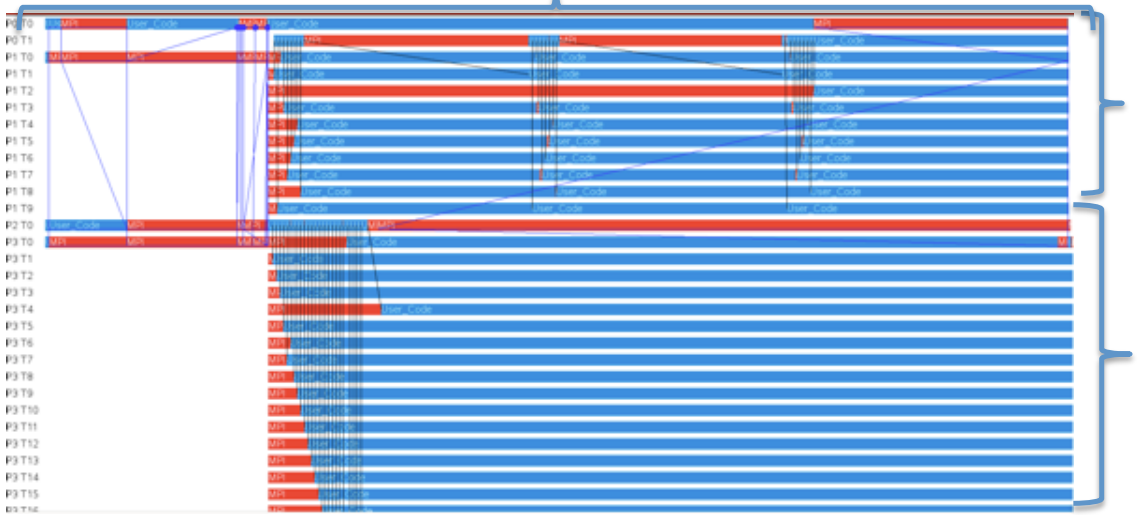


Host Processes (0-12)
1st pipeline

Xeon Phi Processes (13-24)
2nd pipeline

Original PFARM Code Host + 1 x Xeon Phi Coprocessor

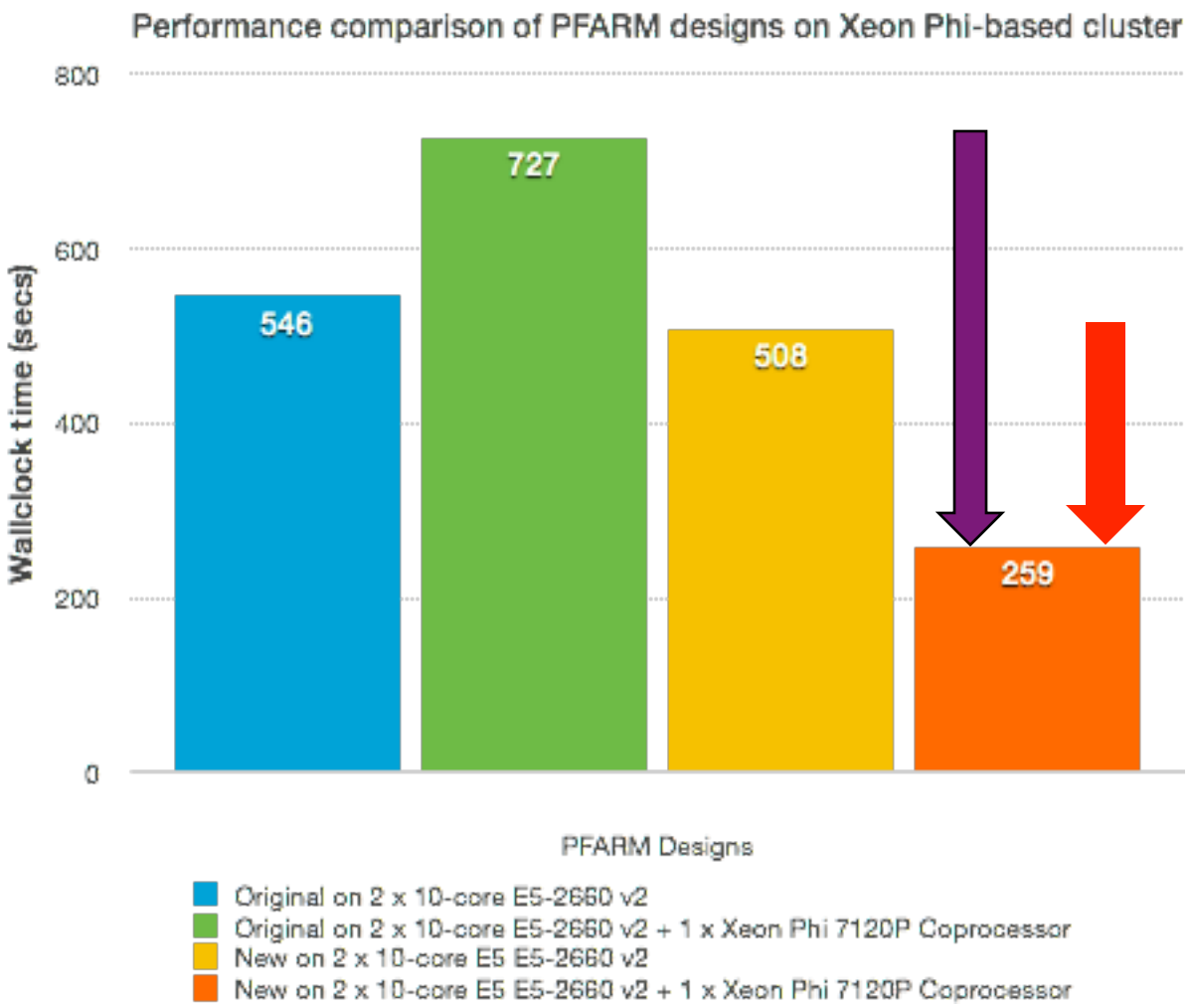
Time-to-solution: 259 secs



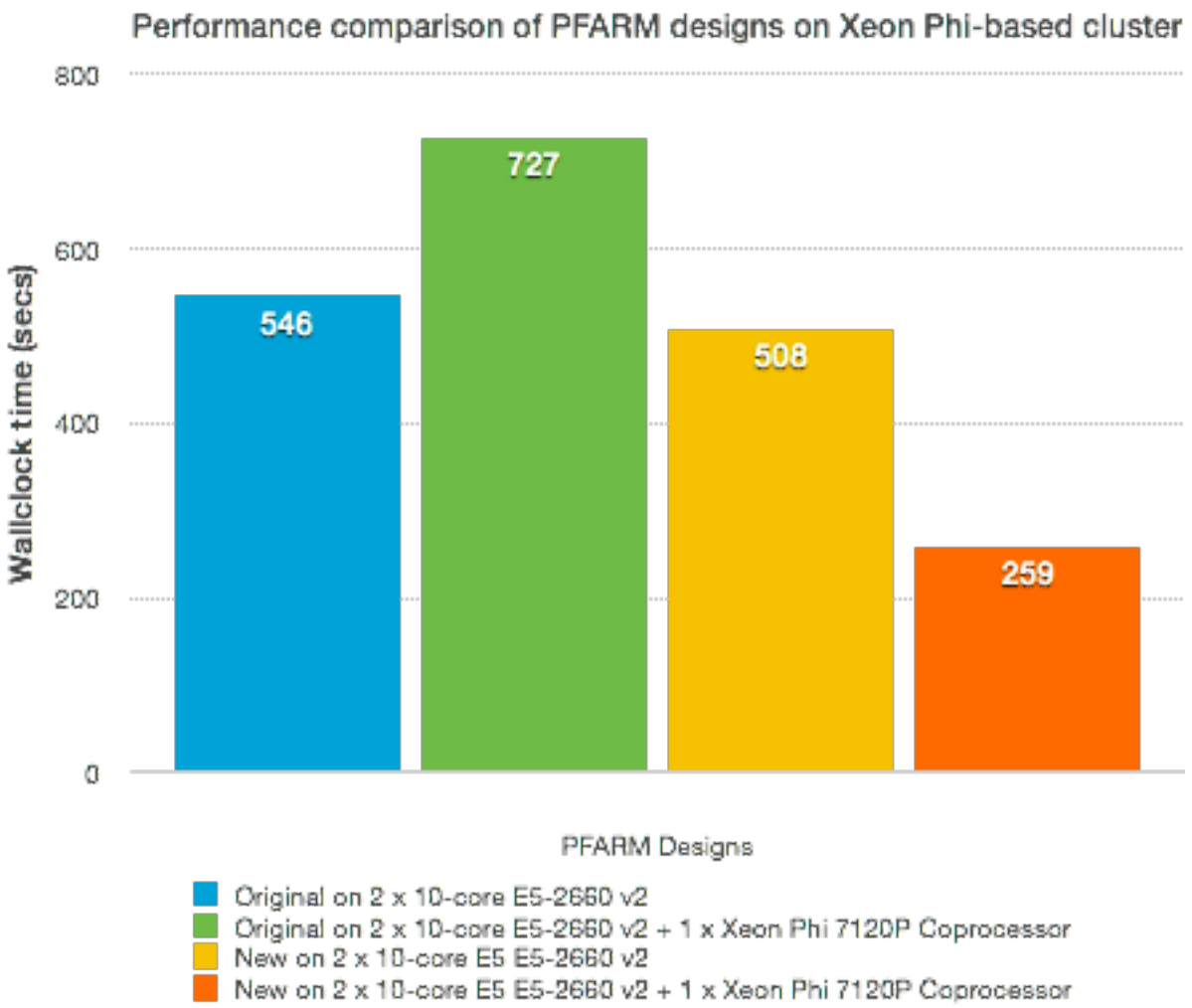
Host Threads (0-10)
1st 10 pipelines

Xeon Phi Threads (0-29)
Next 30 pipelines

New PFARM Code on Host+1 x Xeon Phi Coprocessor



- **New design leads to 2.8x speedup vs original design on Xeon + 1 x Xeon Phi coprocessor**
- **New design on 2S Xeon IVB + 1 x Xeon Phi coprocessor leads to ~2x improvement in time-to-solution vs new design on 2S Xeon IVB only**
- ***New design exposes more parallelism to further improve performance on Xeon Phi***



- ITAC can help to inform and guide the non-trivial redesign of MPI codes on heterogeneous systems
- Redesigning to exploit OpenMP shmem can lead to increase in parallelism
- MPI_THREAD_MULTIPLE support can be useful when redesigning for OpenMP shmem parallelism
- Currently exploring load-balancing issues and the exposure of nested parallelism
- Also preparing for KNL – code correctness with SDE and opportunities for HBM
- One design of many – feedback from IXPUG community welcome!

BACKUP

Why did we design the trunk version of PFARM like this ?

1. Limited memory per node on past architectures (still is a limitation for some problems). Pipelines can take full advantage of overlapping comms/comps.

2. Very flexible - parallel groups adjusted/multiplied according to target datasets.

3. It generally gets ~30% peak flops performance per node.:
http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0703.pdf