EPiGRAM

Exascale ProGRAmming Models

Towards Exascale Programming Models
HPC Summit, Prague
Erwin Laure, KTH

Exascale Programming Models

- With the evolution of HPC architecture towards exascale, new approaches for programming these machines need to be found - EPiGRAM focuses on exploring programming models for the exascale era.

- Intense discussion whether existing models can be improved to exascale or whether disruptive changes are needed.

Plan A

- Devise a new programming model
  - Ideally high level to increase productivity
  - Including autotuning and adaptivity
  - Deals efficiently with heterogeneous hardware
    - Combination of compiler/runtime system
- These are important research questions one should (and people actually do) work on
  - But will take a long time before usable in real applications

EPiGRAM

Plan B

- Work on improving existing, widely used models
  - MPI
  - OpenMP
  - Recently PGAS has also gained momentum
  - Cuda/OpenCL/OpenACC

EPiGRAM Focus

- MPI and PGAS
  - Proven petascale technologies
  - MPI still most widely used
- Challenges
  - Reduction of memory consumption in communication
  - Efficient collective operations
  - Reduced need for synchronization
  - Interoperability

EPiGRAM

Key Objectives of the Project

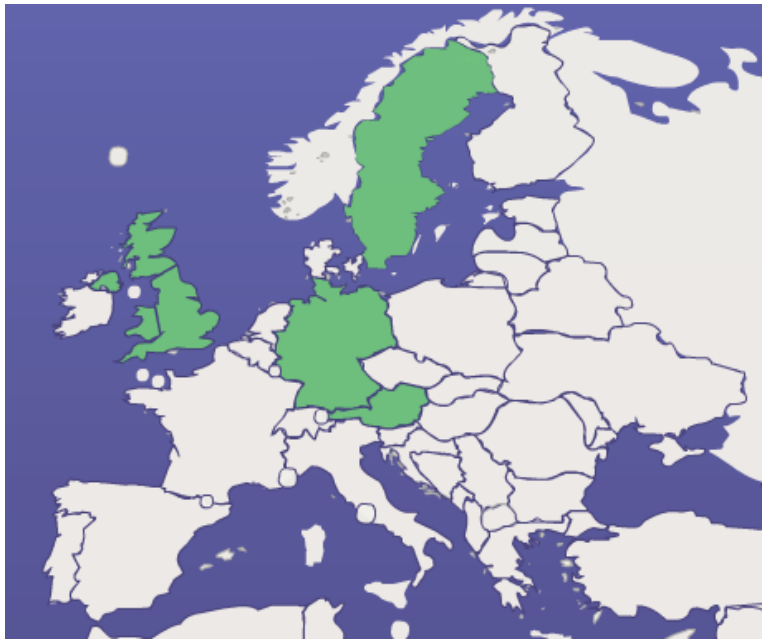
- Address the **scalability** (performance and memory consumption) problem for MP and PGAS models.
- Propose **GASPI/GPI as the European PGAS** approach to exascale.
- Design a **hybrid MP-PGAS** programming model that combines the best features of the two approaches.
- Contribute to **standardization** efforts
- Prepare two **applications** to exascale by redesigning and implementing their communications kernels.

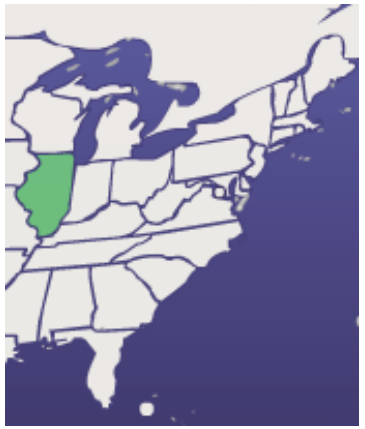
EPiGRAM

Key Players and Their Main Focus

- KTH: management (WP1), applications (WP6)
- TUW: exascale MP (WP2)
- FRAUNHOFER: exascale PGAS (WP3)
- CRAY UK: programming models for diverse memory spaces (WP3)
- EPCC: PGAS-based MPI (WP4)

- External Contributor: UNIVERSITY OF ILLINOIS: exascale MP (WP2)





EPiGRAM

Main Achievements

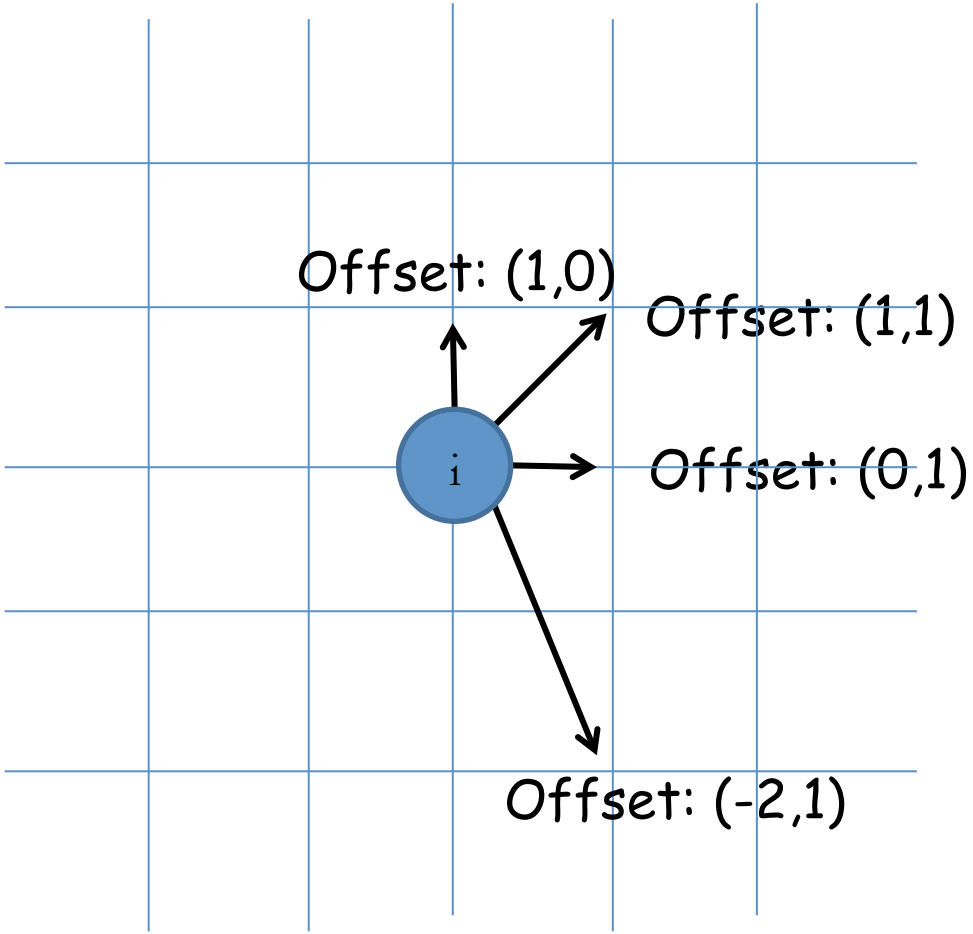
- Improvements of collectives and memory consumption in MPI
- Scalable GPI-2
- Interoperability of MPI and GASPI/GPI
  - PGAS-based MPI
  - Prototype of MPI-Endpoints
- Application validation

Exascale Message Passing

1. Dealing with limited and slower memory:
   - in-depth analysis of MPI **derived datatype** mechanism for saving copy-operations;
   - Space efficient representation of derived datatypes
   - analysis of MPI **collective interface** specification with suggestions for improvement
2. Collective communication at scale:
   - proposal for specification of homogeneous stencils, towards improved (homogeneous, regular) sparse collectives
3. A streaming model for MPI
4. Other issues to be addressed:
   - collective communication in sparse networks
   - Multi-threaded MPI
   - MPI with other models (threads, PGAS, extended message-passing models)

EPiGRAM

Neighborhood Collectives



Sparse neighborhood: list of offsets for target processes

Isomorphic: all processes give same list

Useful for stencil computations, much more general than possible with Cartesian MPI topologies, and much more lightweight and dynamic than with MPI graph topologies

EPiGRAM

New, lightweight sparse collective operation for MPI

- MPI processes in regular torus
- Each MPI process communicates with small number of neighbors
- Collective communication: exchange with all neighbors, reduce over all neighbors, …
- All processes have the same kind of neighborhood

Light-weight set-up function:
Iso_create_neighborhood(int s, int relative_offsets[],…,comm);

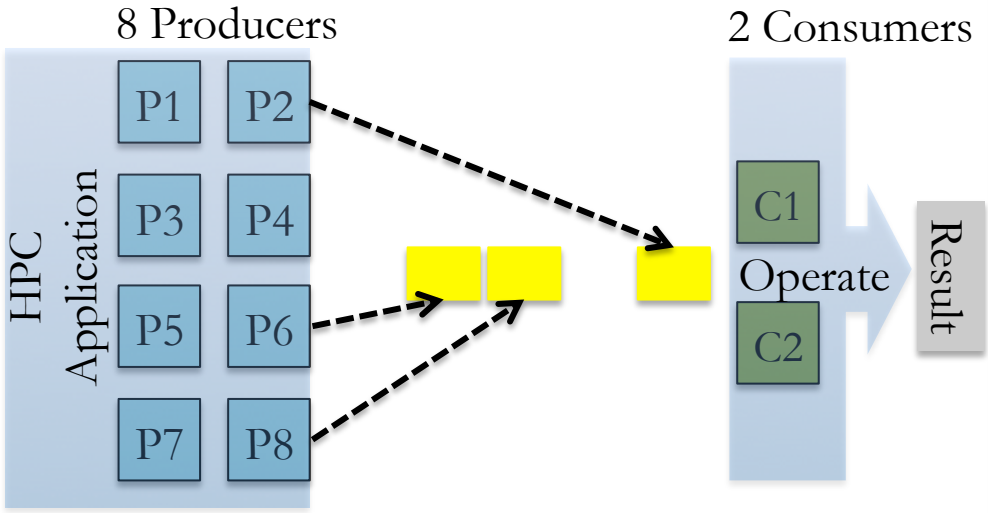
Advantage:
More leverage for easily pre-computing efficient communication schedules

Collective operations:
- Iso_alltoallw(sendbuf,…,recvbuf,…,comm);
- Iso_allgather(sendbuf,…,recvbuf,…,comm);
- Iso_Reduce_scatter(sendbuf,…,recvbuf,…,comm);

EPiGRAM

A Streaming Approach for MPI

- MPI processes are data producers or consumers
- Data producers carry out HPC applications and stream out data (in unit of **Stream Element**) to consumers
- Data consumer process each stream element according to the **Operation** attached to the stream on **first-come-first-served basis**



EPiGRAM

Exascale PGAS

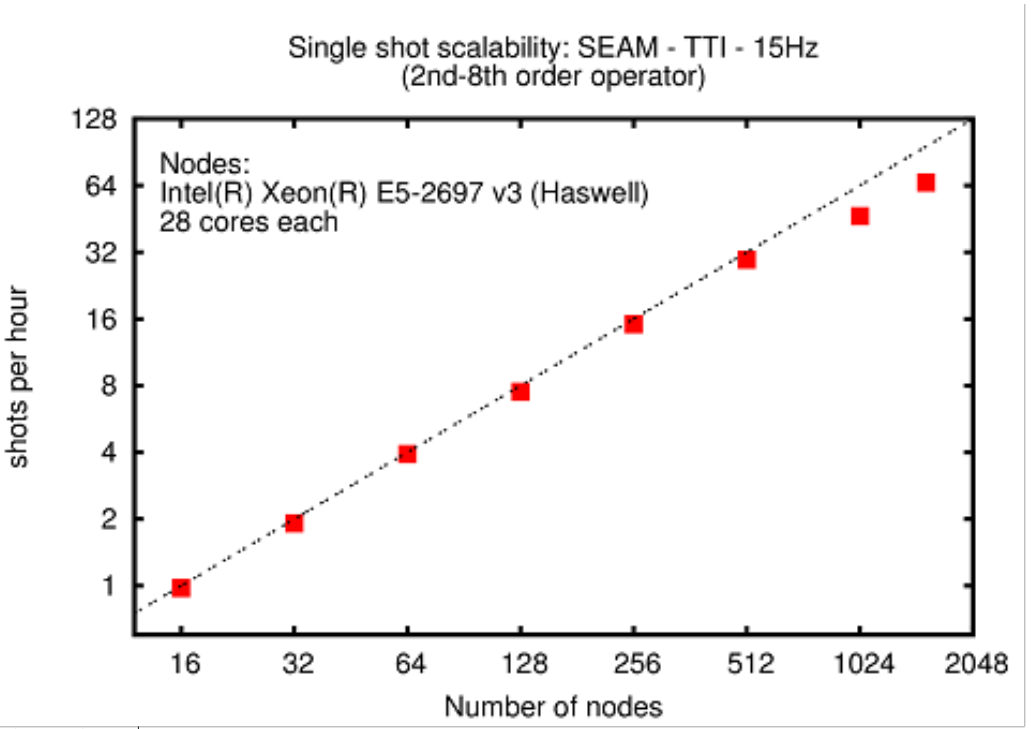
- Increase scalability of collective operations and synchronization in GASPI/GPI
- Improve exploitation of diverse and hierarchical memory spaces in PGAS
- Improved interoperability
- Standardization contributions in the GASPI Forum

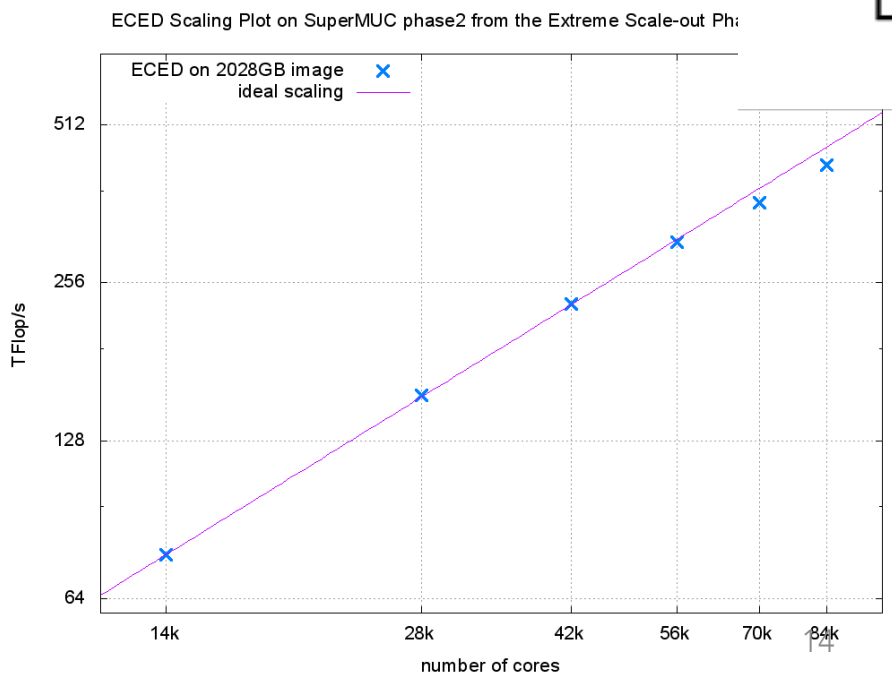
EPiGRAM

GPI Scalability

Scalability tests on SuperMUC

- Seismic imaging based on Reverse time migration, strong scaling





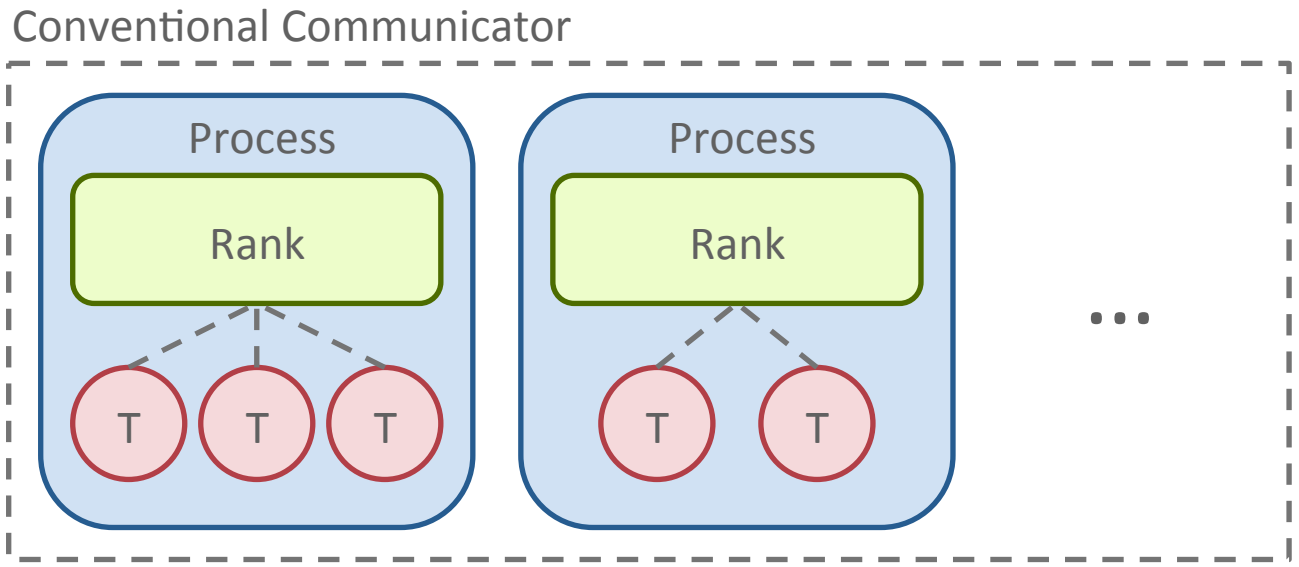
- De-noising seismic images using the ECED filter
  - About 0.4 PFs bound by memory bandwidth

EPiGRAM

PGAS-based MPI

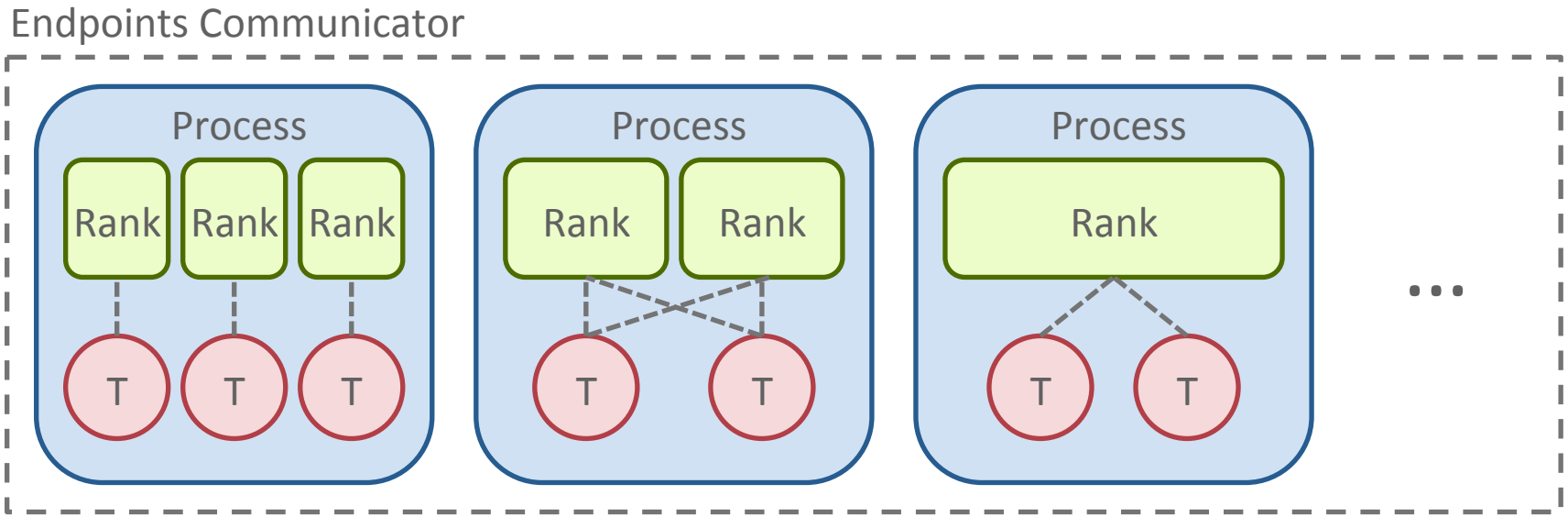
- Development of EMPI4Re as research vehicle
  - Based on T3DMPI
  - Investigate different design choices
    - Eg. memory consumption vs. performance
  - Pilot implementation of MPI-Endpoints

MPI Endpoints – a Way Forward for MPI+X



- MPI provides a 1-to-1 mapping of ranks to processes
- This was good in the past, but usage models have evolved
  - Programmers use many-to-one mapping of threads to processes
    - E.g. Hybrid parallel programming with OpenMP/threads
  - Other programming models also use many-to-one mapping
    - Interoperability is a key objective, e.g. with Charm++, etc…

Endpoints: Flexible Mapping of Ranks to Processes



- Provide a many-to-one mapping of ranks to processes
  - Allows threads to act as first-class participants in MPI operations
  - Improve programmability of MPI + node-level and MPI + system-level models
  - Potential for improving performance of hybrid MPI + X
- A rank represents a communication "endpoint"
  - Set of resources that supports the independent execution of MPI communications
- Note: Figure demonstrates many usages, some may impact performance

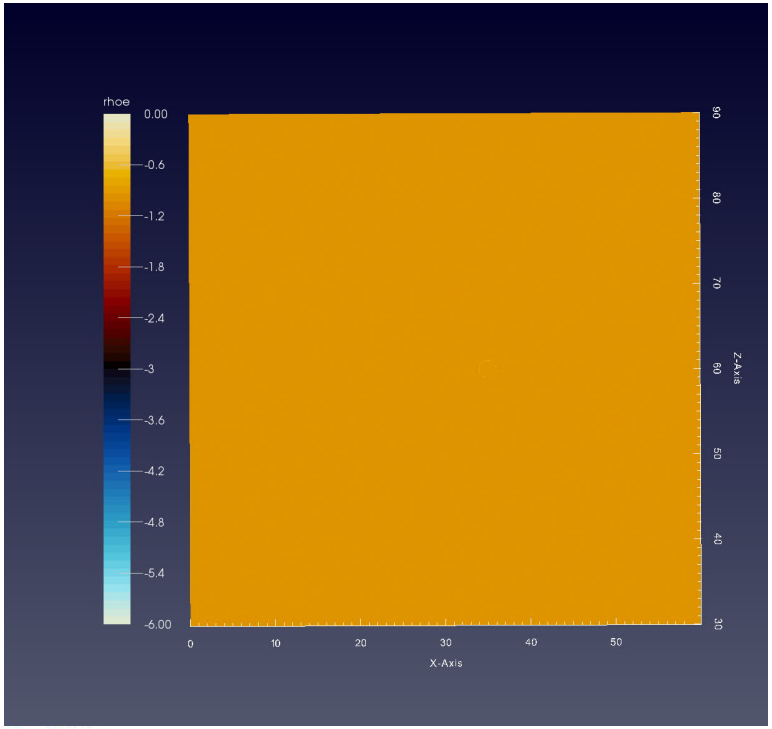
EPiGRAM MPI Endpoints Contributions

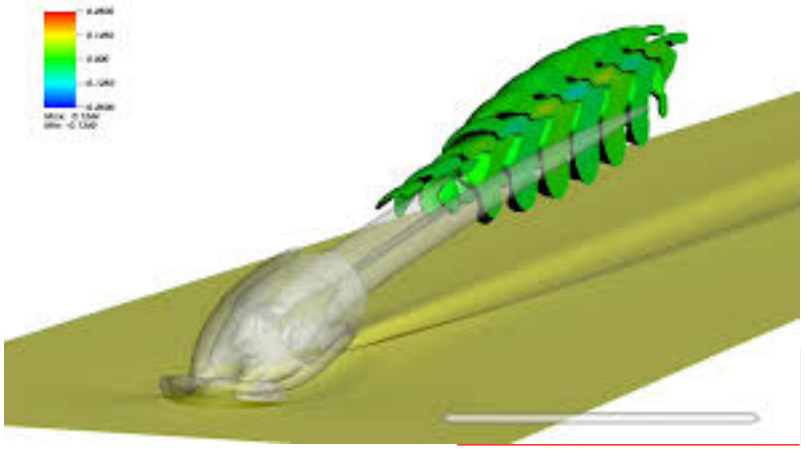
- Significant contributions to MPI Forum
  - Effects on group manipulation functions discovered and addressed
  - Communicator query function modified to identify endpoints communicator
- Paper regarding new context id allocation
  - Current algorithms in MPICH and OpenMPI will not work with multiple local endpoints
  - Fixes identified and implemented in McMPI

EPiGRAM

Two EPiGRAM Pilot Applications

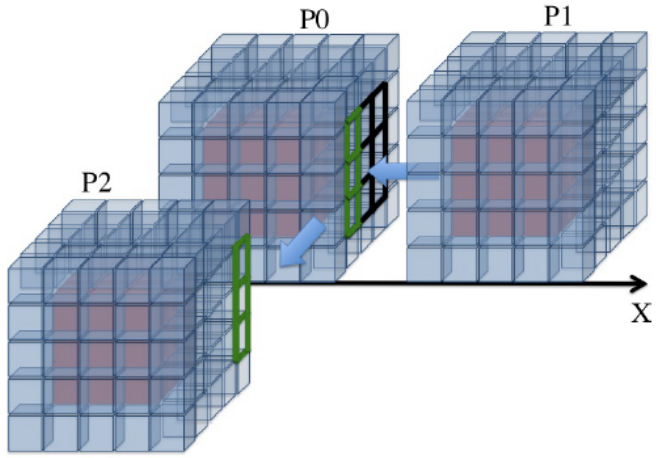
- iPIC3D:
  - *Particle-in-Cell Application for space weather prediction*
- Nek5000:
  - Spectral code for incompressible CFD
  - NekBone (mini-app)
- IFS (**NEW**)
  - Weather forecast, ECMWF

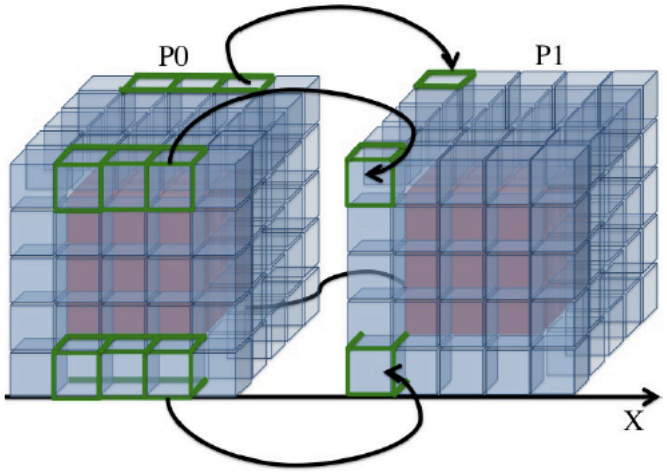




iPIC3D - non-blocking P2P and derived data types

- iPIC3D code is now fully using non-blocking point-to-point communication.
- MPI derived data types are used to avoid user-defined buffers for halo exchange and to decrease memory usage by the iPIC3D application.



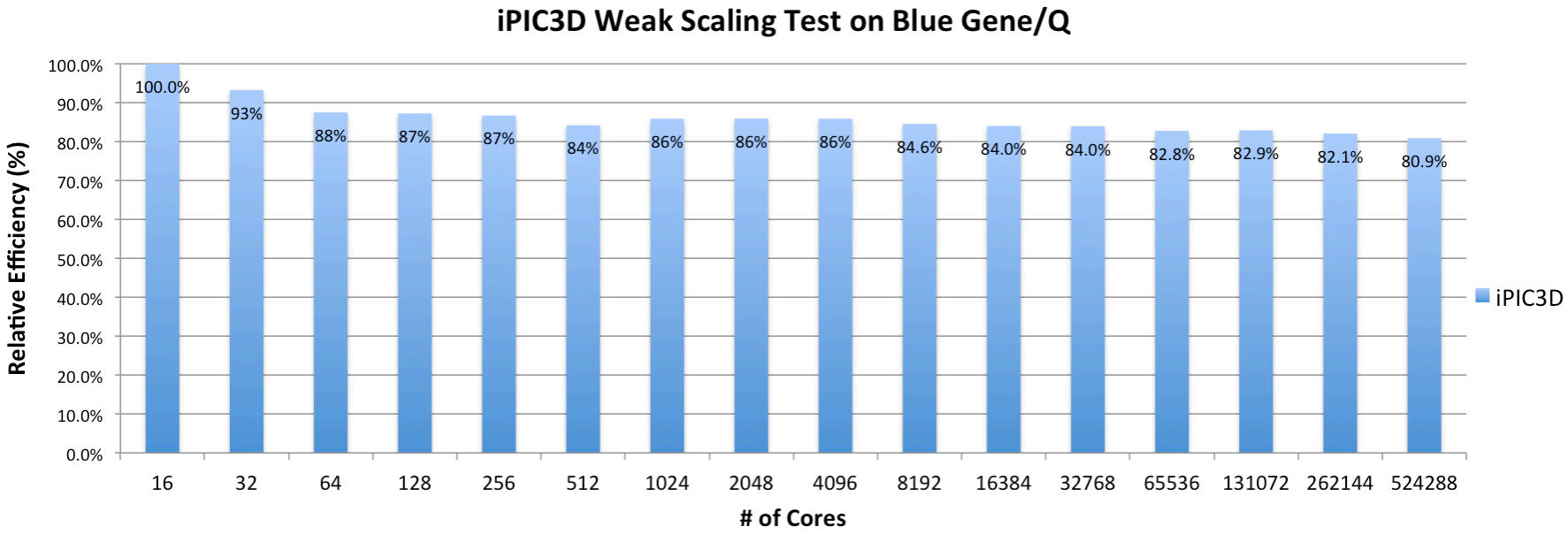


EPiGRAM

iPIC3D - non-blocking collective I/O

- Used non-blocking collective I/O.
- This allows us to overlap I/O with other work.
- Redesigned iPIC3D workflow to overlap I/O and computation.
- Using non-blocking collective I/O leads to decreased execution time
  - E.g. we decrease the execution time of the typical production simulation on 2,048 cores by 6%

Now scaling at half million cores!
(before EPiGRAM only at 8K)



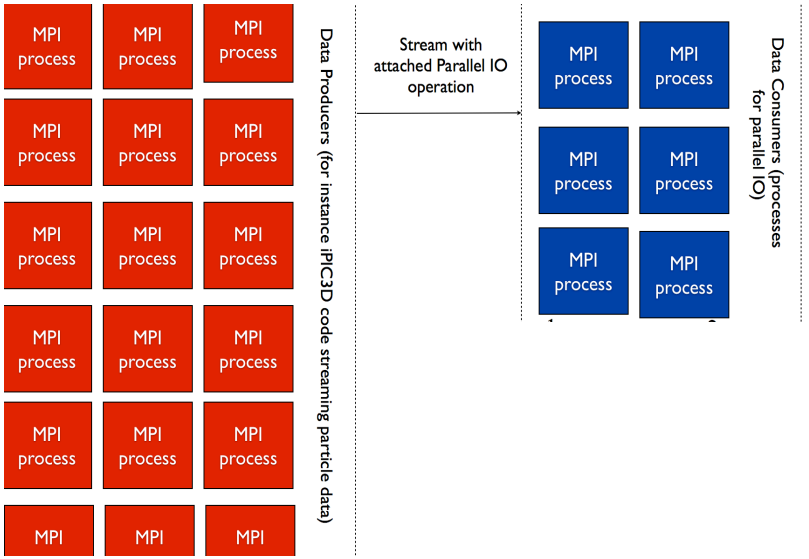
Scaling tests on VESTA BG/Q at ANL

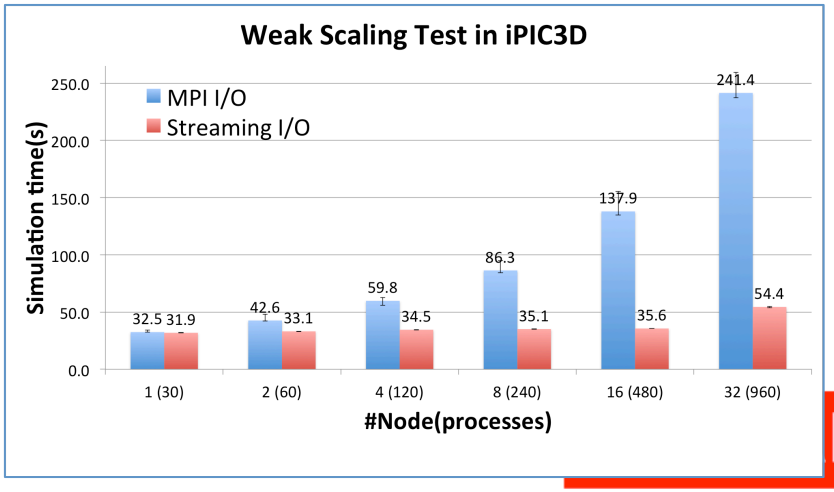
EPiGRAM

iPIC3D – Streaming Communication

We deployed the MPIStream library, developed in WP2, in the iPIC3D application :

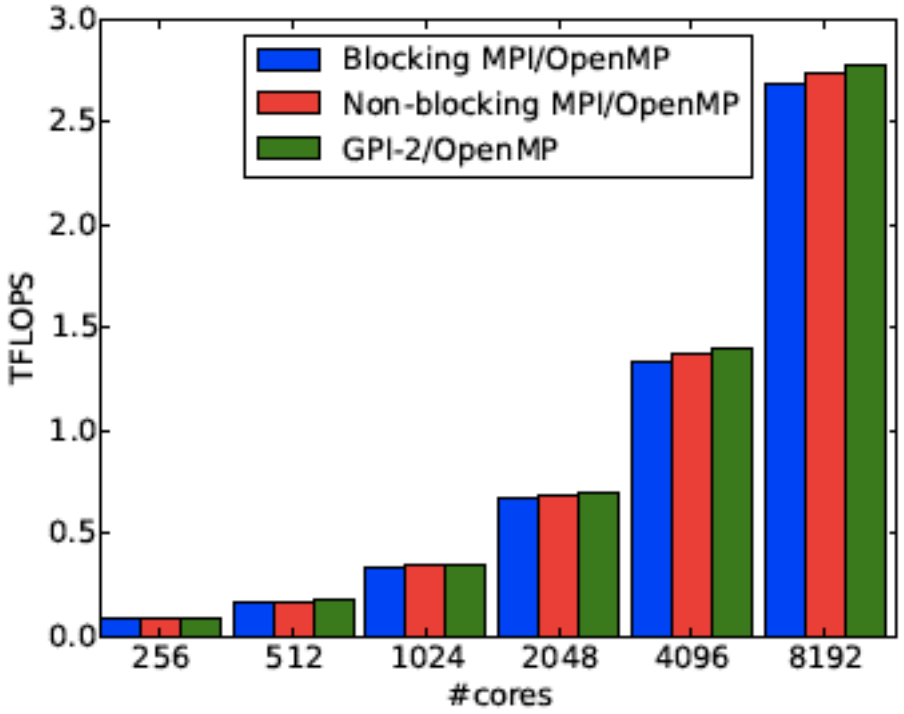
- I/O
- Communication of particles
- Monitor load imbalance





MPI/GPI+ OpenMP Communication in Nek5000

- Implemented new communication kernel in blocking/ non-blocking MPI, GPI + OpenMP
- Work presented at WRAP workshop @ CLUSTER2015



EPiGRAM

Summary

- Exascale technologies pose new challenges on efficient programming
  - Hybrid forms of parallelism, deep and limited memory hierarchies, dynamic behavior, fault tolerance, etc.
  - Not only relevant for exascale systems but for **all** systems
- Evolutionary path that builds on existing, widely used components is likely more promising than designing a new programming model
- MPI and PGAS are good starting points for such an evolutionary path
  - Interoperability is a key issue!
  - Higher abstractions can (**should**) be built on top

EPiGRAM