EPiGRAM-HS: Programming Models for Heterogeneous Systems at Exascale

Erwin Laure

PDC, KTH Royal Institute of Technology

18th Workshop on High Performance Computing in Meteorology

The End of Historic Scaling

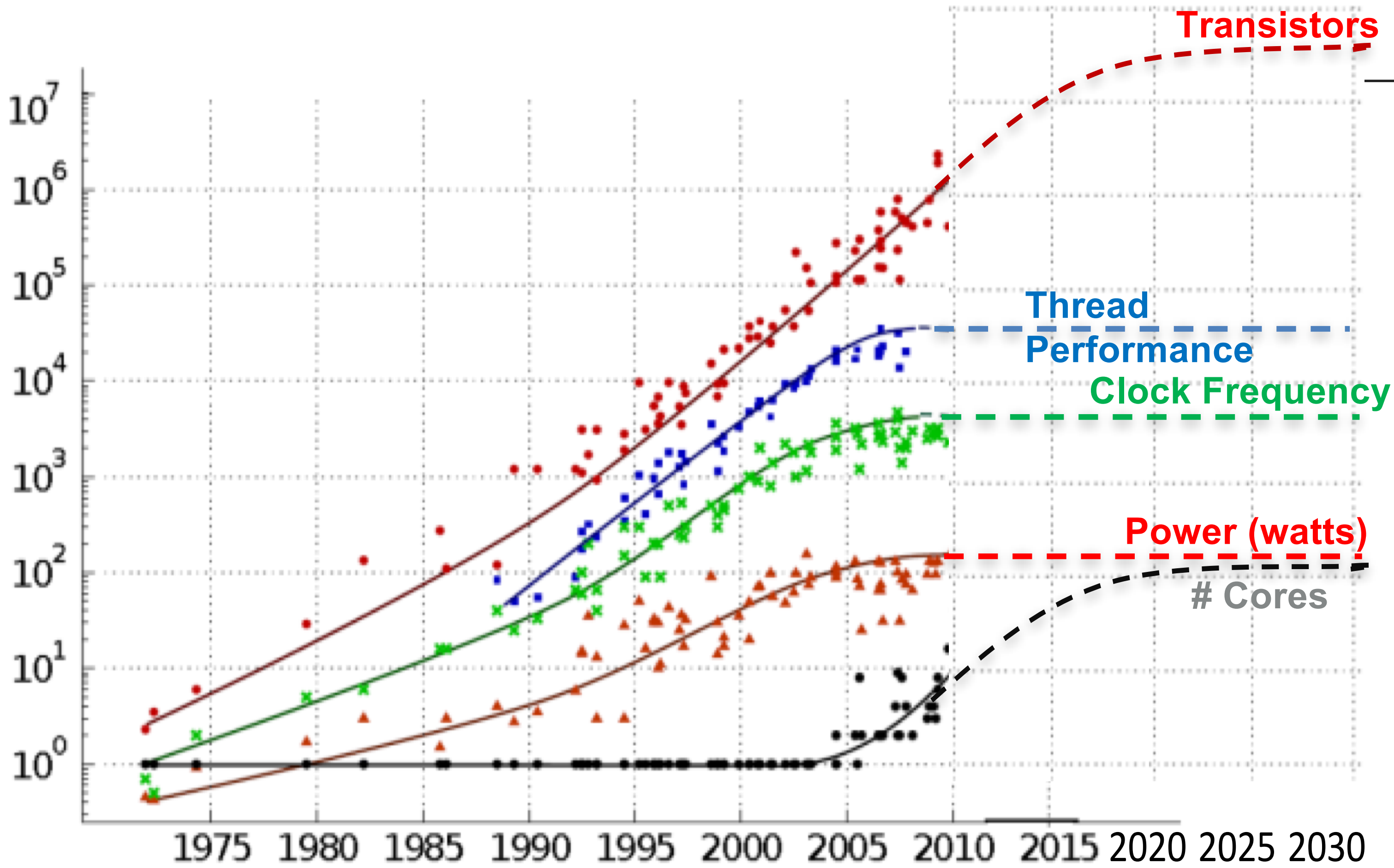
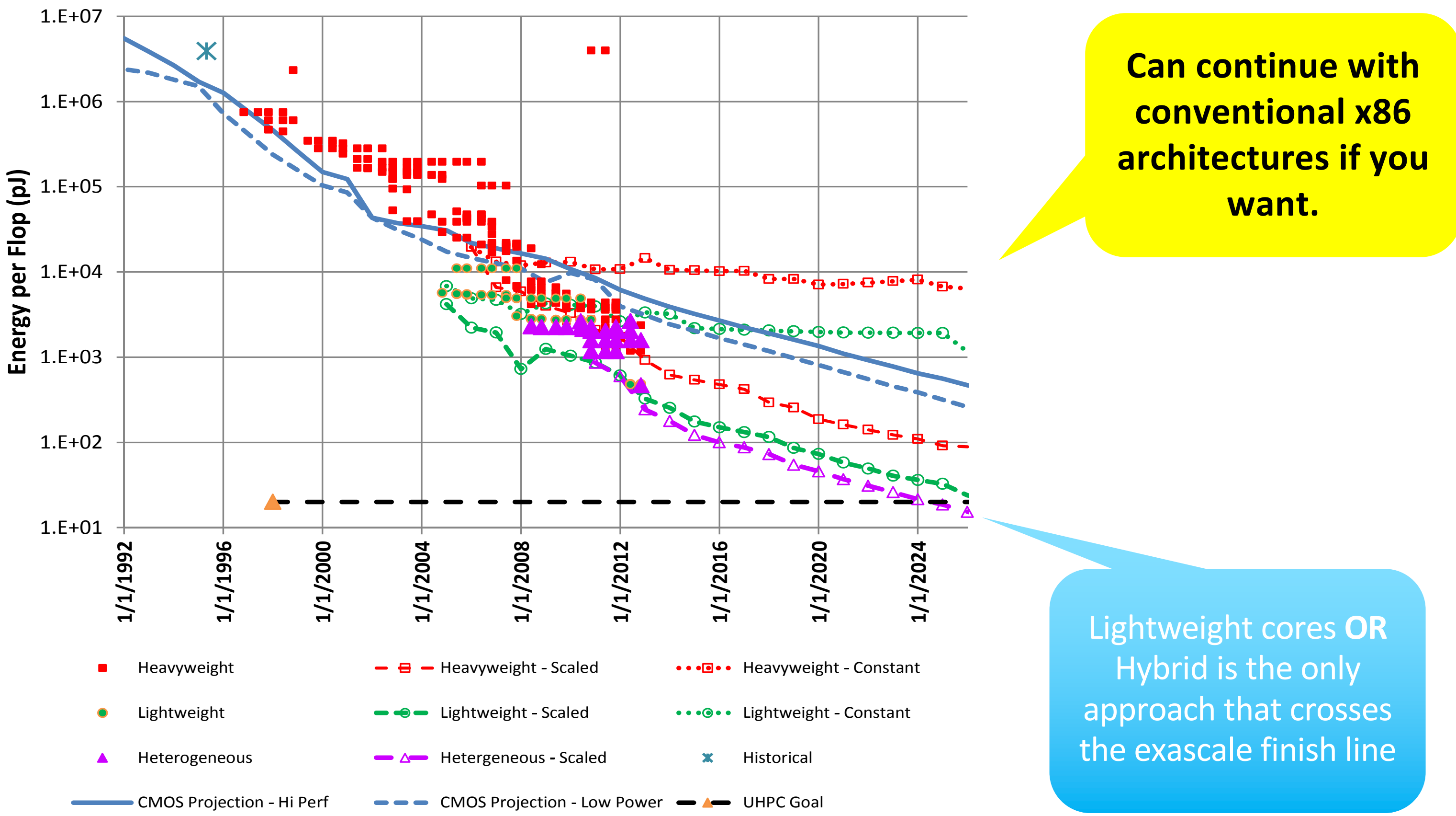


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

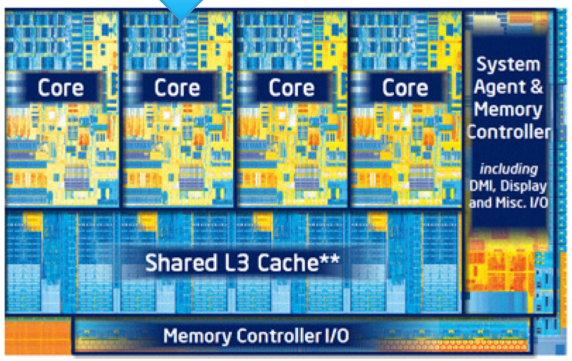
But Mere Multi-Core is NOT good enough!
(need to go to simpler cores)



Slide curtesy John Shalf

Heterogeneous Future (LOCs and TOCs)

Big cores (very few)

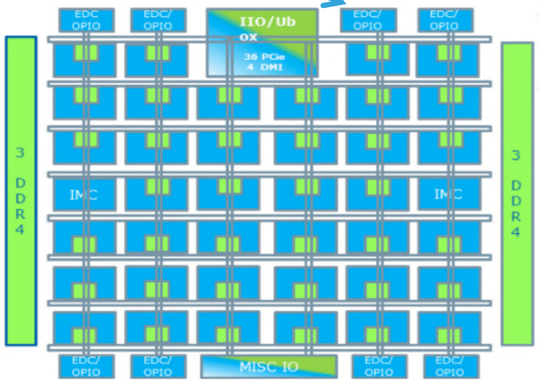


Tiny core

Lots of them!

0.23mm

0.2 mm



Latency Optimized Core (LOC)

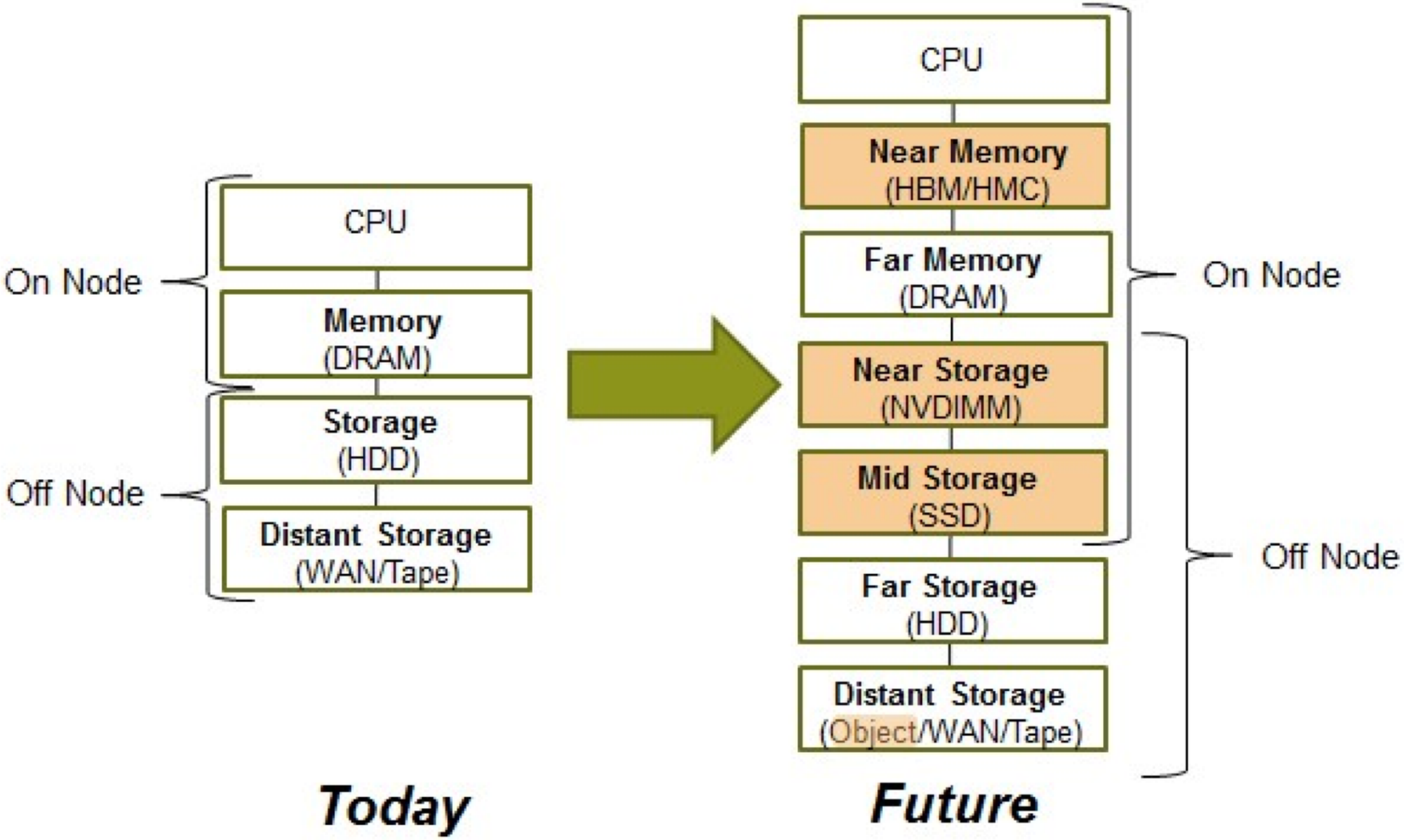
Most energy efficient if you don't have lots of parallelism

Throughput Optimized Core (TOC)

Most energy efficient if you DO have a lot of parallelism!

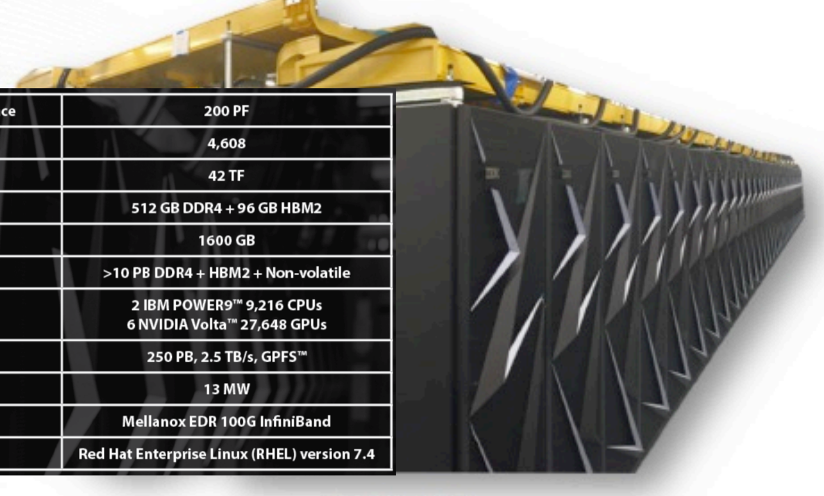
Slide curtesy John Shalf

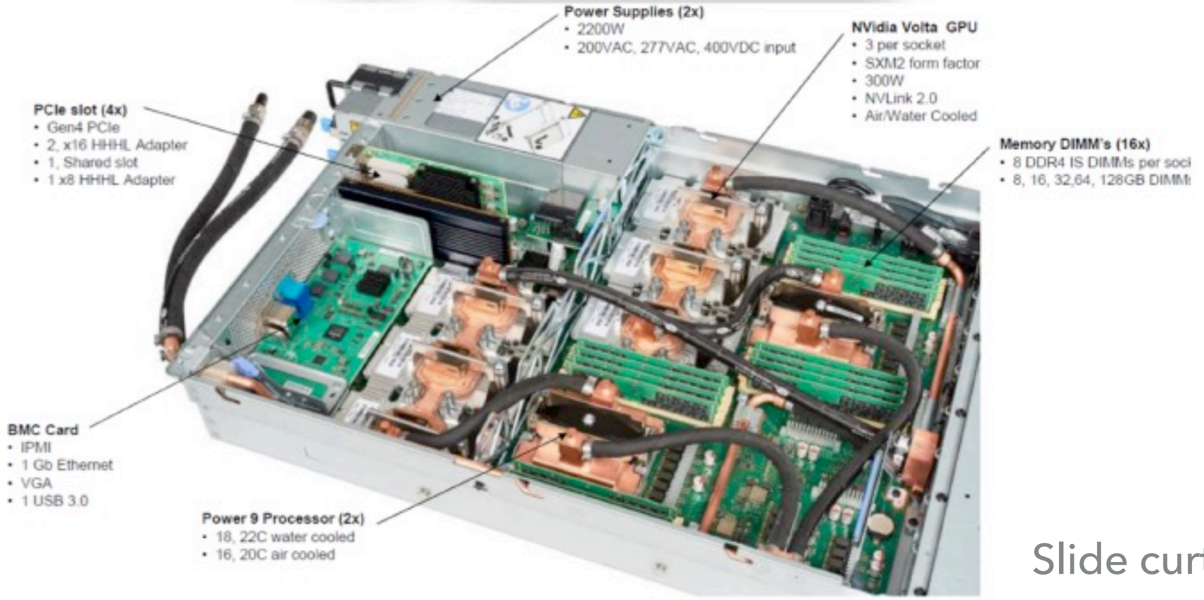
Trends in the Memory/Storage Subsystem

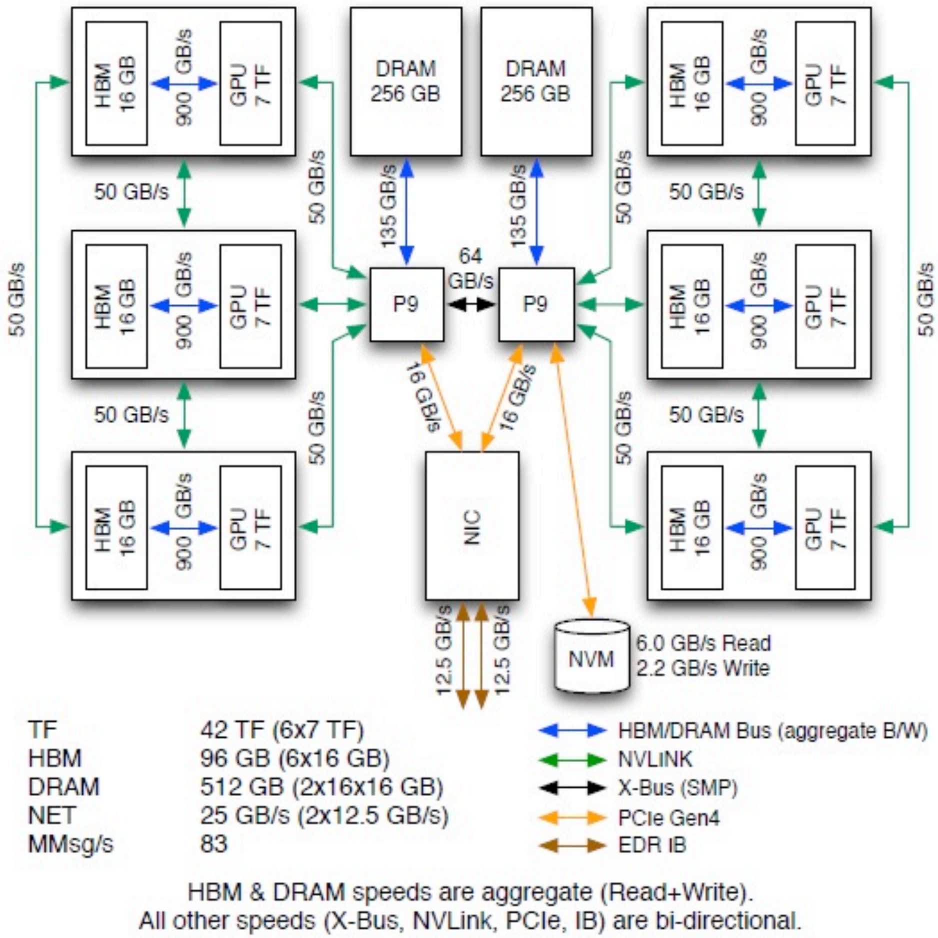


Summit Node Overview

Application Performance	200 PF
Number of Nodes	4,608
Node performance	42 TF
Memory per Node	512 GB DDR4 + 96 GB HBM2
NV memory per Node	1600 GB
Total System Memory	>10 PB DDR4 + HBM2 + Non-volatile
Processors	2 IBM POWER9™ 9,216 CPUs 6 NVIDIA Volta™ 27,648 GPUs
File System	250 PB, 2.5 TB/s, GPFS™
Power Consumption	13 MW
Interconnect	Mellanox EDR 100G InfiniBand
Operating System	Red Hat Enterprise Linux (RHEL) version 7.4







Slide curtesy Jeffrey Vetter

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

How to Program these Systems?

- 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

What Applications Want

- HPC System Architecture and Components
  - Efficient use of memory and I/O hierarchies - Balance Compute, I/O and Storage Performance
  - Efficient interaction between “fat” and “thin” (GPU) cores
- System Software and Management
  - Software standards (C++17 and Fortran 2015 in particular, but also OpenMP 4.5, MPI 3.1, OpenCL 2.2,...)
- Programming Environment
  - (Dynamic) environments for task parallelism.

Plan B

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

- This was the focus of the FP7 project (2013-2016)

EPiGRAM Focus

- EPiGRAM believes in the incremental approach and that the most promising parallel programming environments can be scaled to exascale:
- 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

Exascale Message Passing

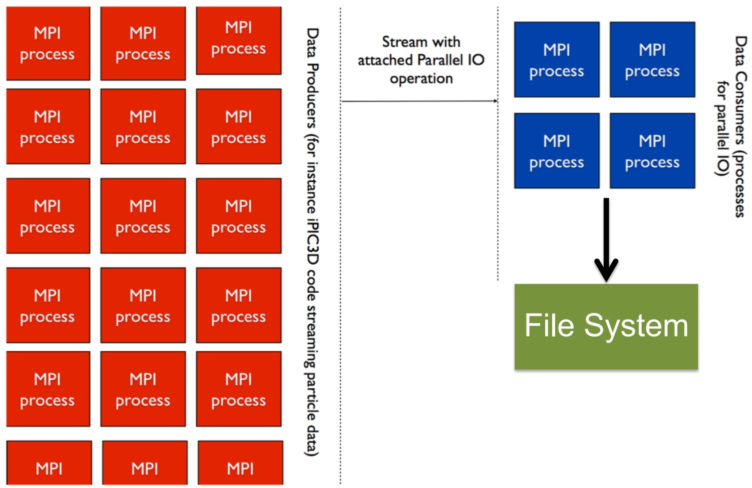
1. Dealing with limited and slower memory:
   - in-depth analysis of MPI **derived datatype** mechanism for saving copy-operations;
   - Space efficient representation of derived datatypes
   - analysis of MPI **collective interface** specification with suggestions for improvement
2. Collective communication at scale:
   - proposal for specification of homogeneous stencils, towards improved (homogeneous, regular) **sparse (isomorphic) collectives**
3. New models:
   - Streaming in MPI
   - MPI interoperability with other models (OpenMP, PGAS)

EPiGRAM

MPIStream for Irregular I/O

- Conventional MPI I/O approach calls reduction operations to find each process's position in the shared file, then call MPI collective I/O -> buffering is not feasible due to large number of particles

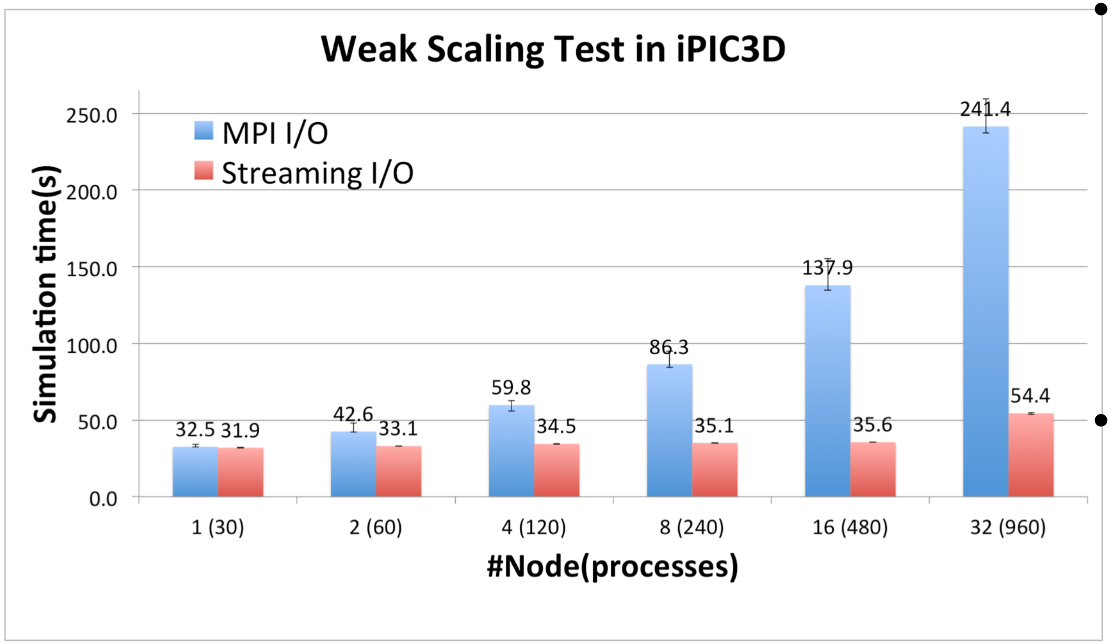
- Streaming I/O enables data producers to stream out data during computation and only data consumers carry out I/O operations



Ivy Bo Peng et al.

EPiGRAM

MPIStream for Irregular I/O in HPC Application



- Streaming I/O: Data producers stream out particle information during computation. Data Consumers perform I/O operations (15 : 1)
- MPI I/O: each MPI process perform I/O operations

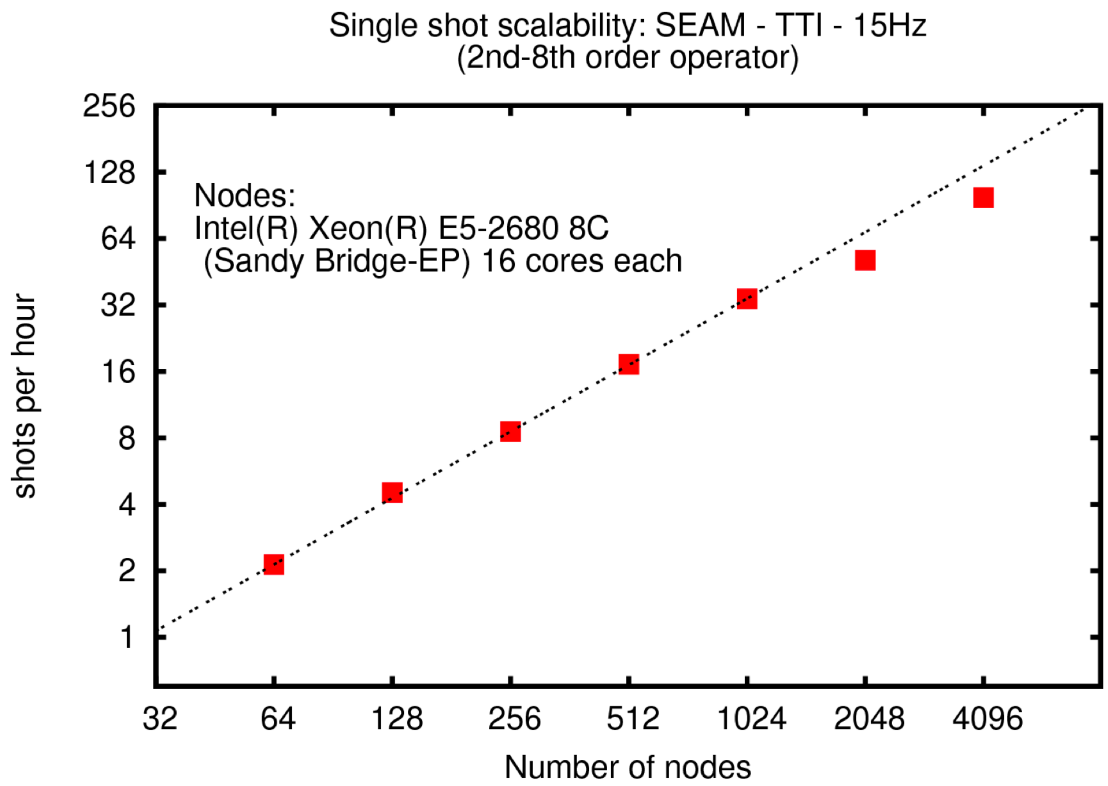
Tests carried out on Beskow supercomputer, a Cray XC40 system based on Intel Haswell processors and Cray Aries interconnect network with Dragon Topology, Cray C compiler version 5.2.40 and the Cray MPICH2 library version 7.0.4)).

Ivy Bo Peng et al.

EPiGRAM

Exascale PGAS

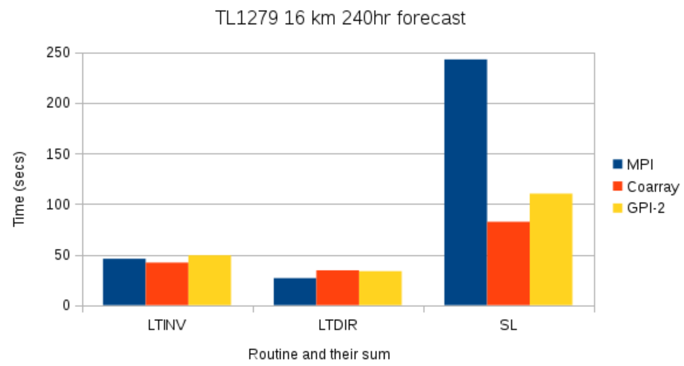
- Increase scalability of collective operations and synchronization in GPI
- Support fault-tolerance in GPI
- Improve exploitation of diverse and hierarchical memory spaces in PGAS
- Isolation of libraries and user managed memory
- Interoperability
  - MPI+GPI-2; migration path
- GASPI Forum

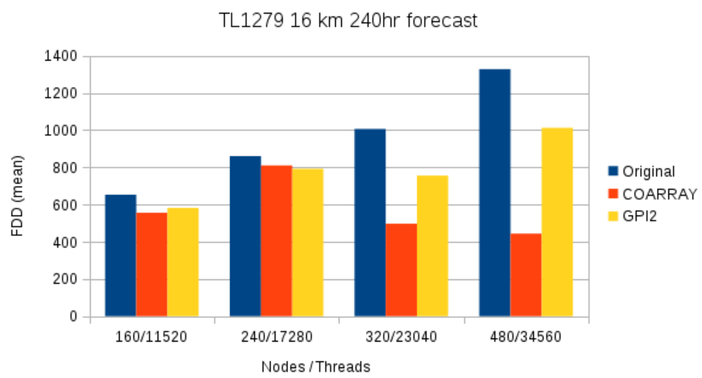


GPI in IFS: Results

- Due to the size and complexity of the complete code, porting efforts have been done incrementally. Currently three main routines have a GPI-2 implementation:
  - inverse Legendre transform (LTINV)
  - direct Legendre transform (LTDIR)
  - semi-Langragian (SL) scheme.

- Existing coarray implementation from the CRESTA project was starting point.





Cray XC30/40

EPiGRAM

EPiGRAM-HS is motivated by the increasing presence of **heterogeneous technologies** on pre-exascale **supercomputers** and by the need of **porting key** HPC and emerging **applications** to these systems **on time for exascale**

Exascale is at Door: will Applications use the ExaFLOPS?

- The race to an ExaFLOPS-capable supercomputer will likely end up in 2020 – 2021
  - That leaves us only 2-3 years for software development and application porting!

- Most of large-scale HPC applications either don't use heterogeneous systems or have limited support in experimental branches
  - Major effort needed for running production-quality simulations from day one of the exascale era

Four Main Project Teams

Network

Heterogenous
Memory

Heterogenous
Compute

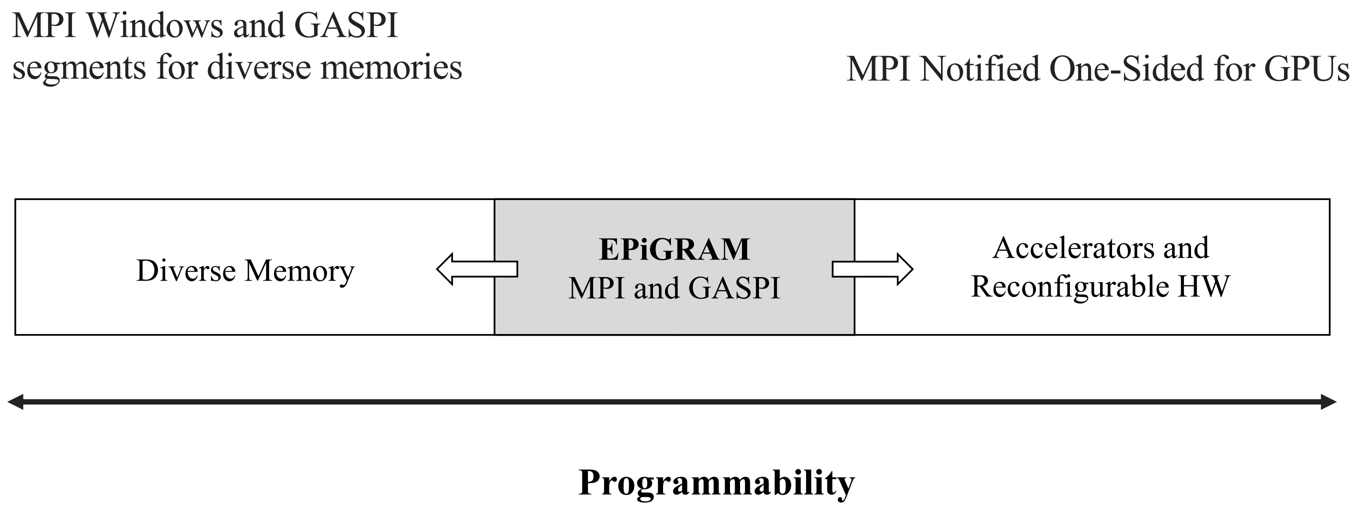
Applications

EPiGRAM-HS Applications

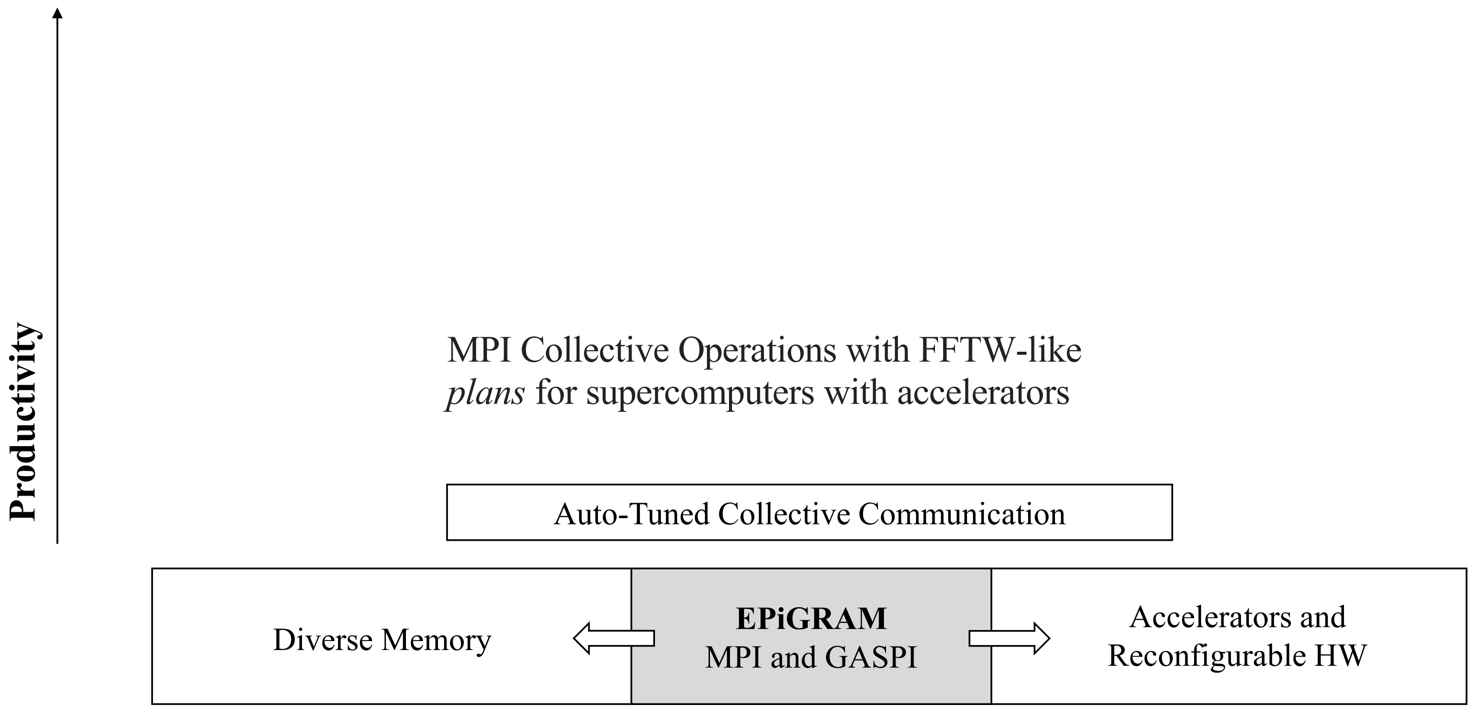
- Traditional HPC Applications
  - **IFS** – Weather Forecast – ECMWF
  - **Nek5000** – CFD – KTH PDC
  - **iPIC3D** – Space Physics – KTH PDC
- Emerging AI Applications
  - **Lung Cancer Detection** – Caffe / TensorFlow – Fraunhofer
  - **Malware Detection** – Caffe / TensorFlow – Fraunhofer

EPiGRAM-HS is developing a **programming environment,** enabling HPC and emerging **applications** to run on large-scale heterogeneous systems at maximum performance

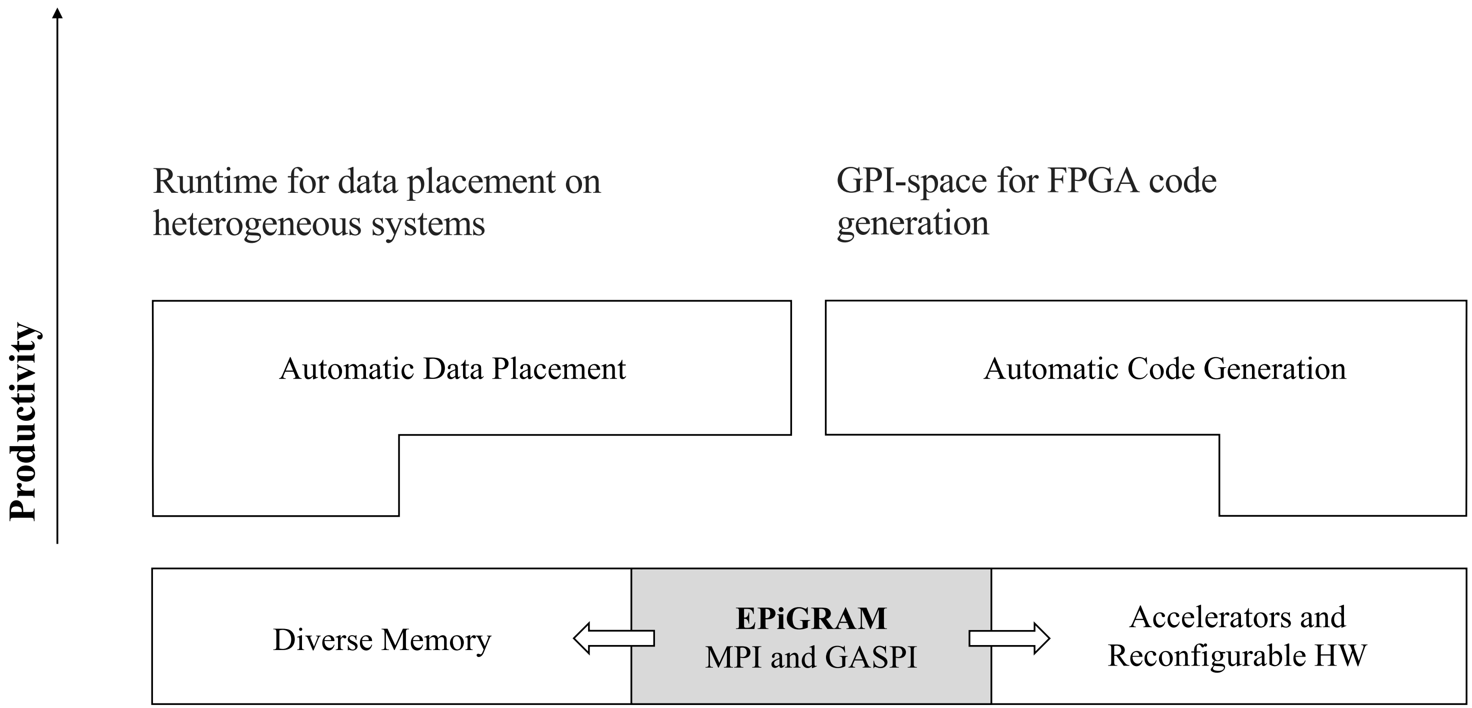
Extending MPI and GASPI Programmability



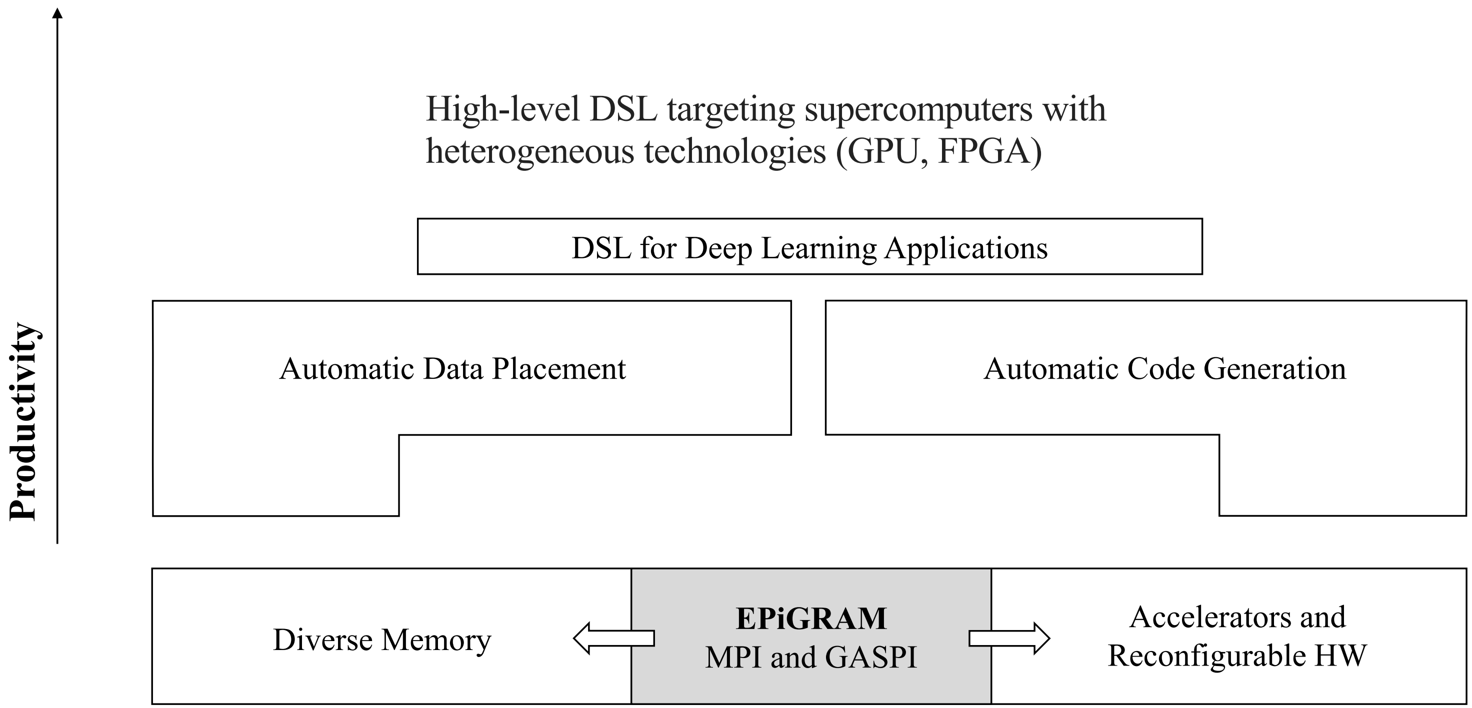
Automation for Productivity: MPI "Planned" Collectives



Automation for Productivity: Runtimes for Data Plac. and FPGAs



Automation for Productivity: DSL for DL on Distributed Het. Systems



Standardization

- MPI Forum
- GASPI Forum (EPiGRAM was founding member)

Project Fact Sheet

- EPiGRAM-HS = **E**xascale **P**ro**GRA**mming **M**odels for **H**eterogenous **S**ystems
  - Continuation of a first EC-funded EPiGRAM project 2013-2016
- EC Call: H2020-FETHPC-2017
  - Sub-topic: a) High productivity programming environments for exascale
- Total Budget: 3,998,741 €
  - Six Partners with KTH as coordinating team
- Started on September 1st 2018 with a duration of three years

Conclusion

- EPiGRAM-HS is motivated by the increase of heterogenous compute and memory systems on pre-exascale supercomputers and porting applications to these systems on time for exascale
- EPiGRAM-HS is a three-year EC-funded project to develop programming models for these systems
- EPiGRAM-HS is developing a programming environment, based on MPI and GASPI, for enabling applications to run on large-scale heterogeneous systems at maximum performance

Funding for the work is received from the European Commission H2020 program Grant Agreement No. 801039 (https://epigram-hs.eu/)

Contact: Stefano Markidis, markidis@pdc.kth.se