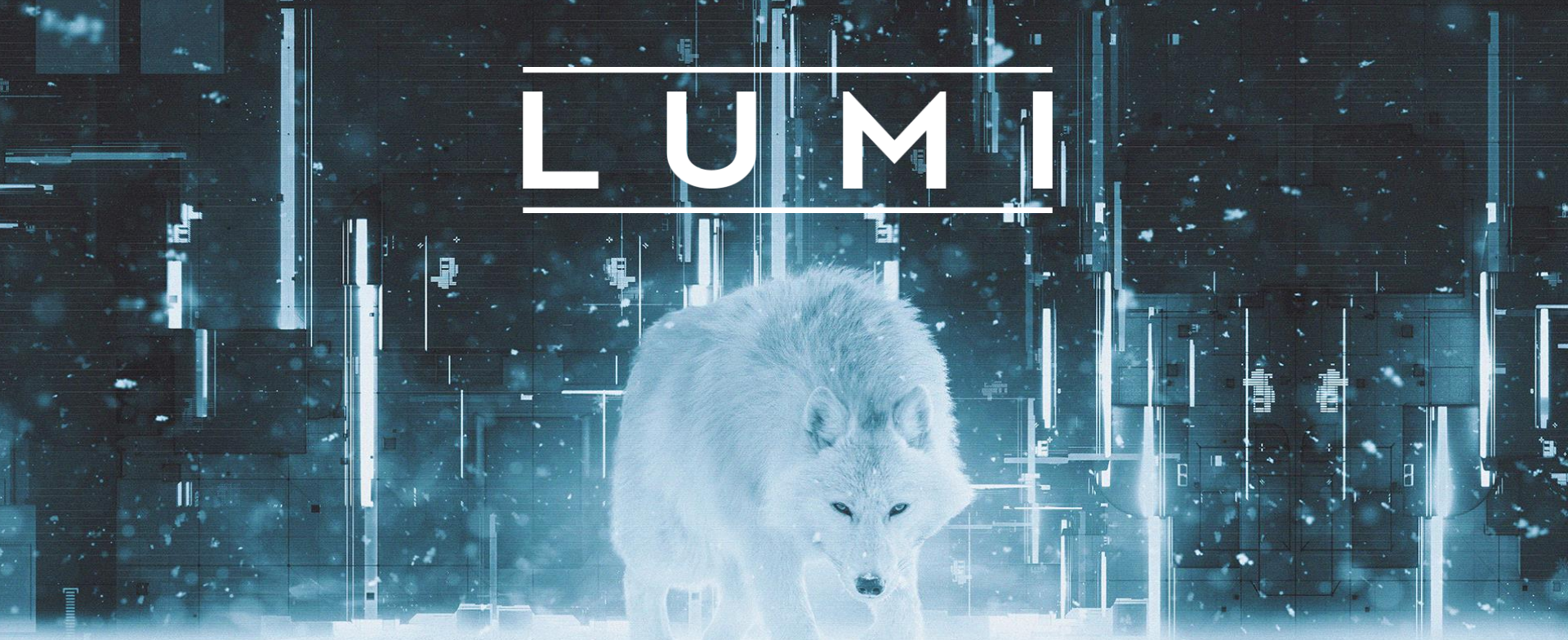


Programming Environment of Europe's flagship supercomputer, LUMI

Dr. Pekka Manninen
Director, LUMI
CSC – IT Center for Science, Finland

LUMI

LUMI: one of the fastest supercomputers in the world

- LUMI is an **HPE Cray EX** supercomputer manufactured by **Hewlett Packard Enterprise**
- HPL performance over **309 petaflop/s** makes the system #3 in the world (Top500)
  - #3 on HPCG, #2 on HPL-MxP

1 system

309

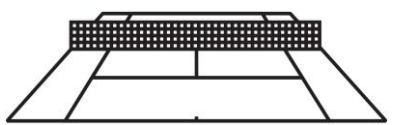
Pflop/s

Sustained performance

Modern platform for

High-performance computing, Artificial intelligence, Data analytics

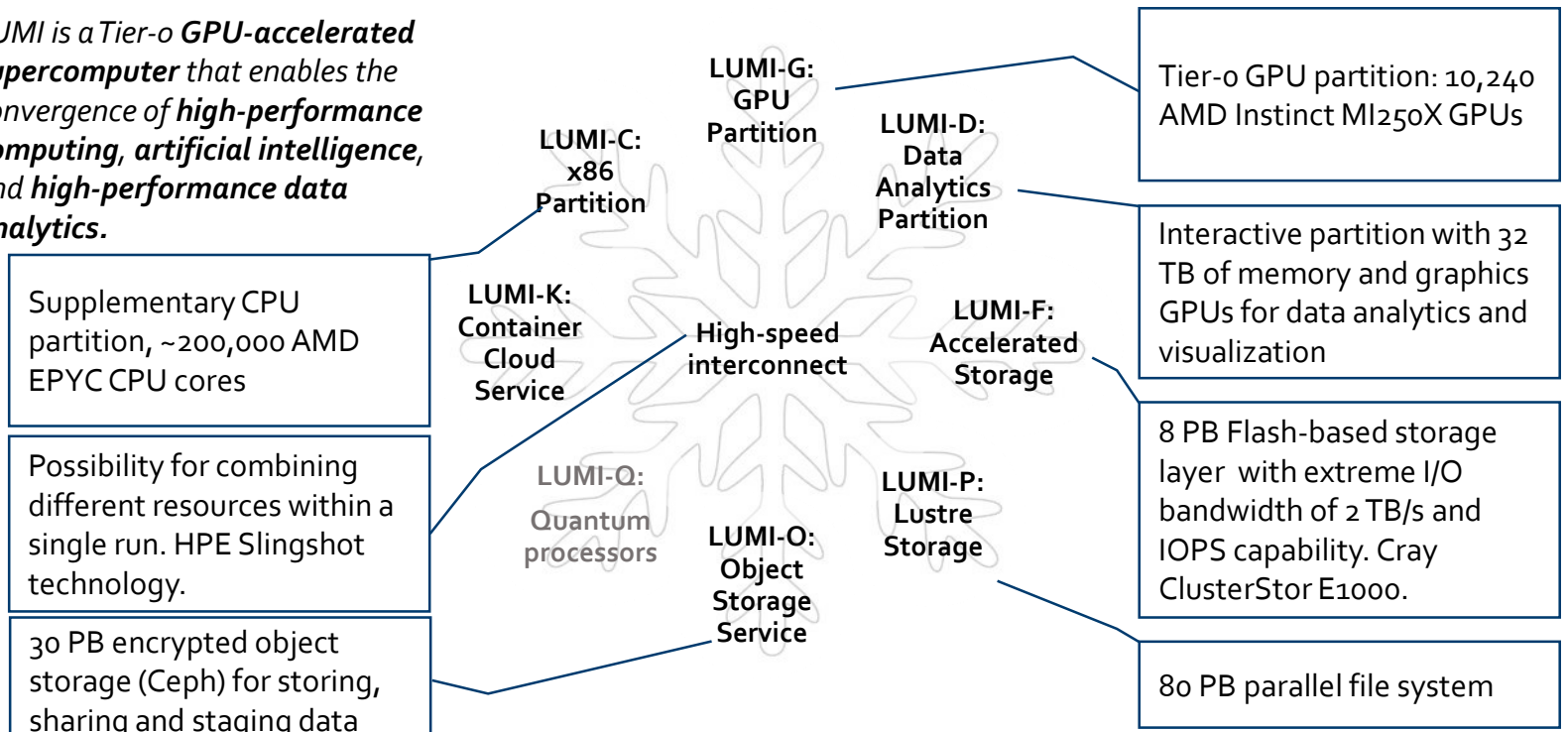
Based on GPU technology



Size of two tennis courts

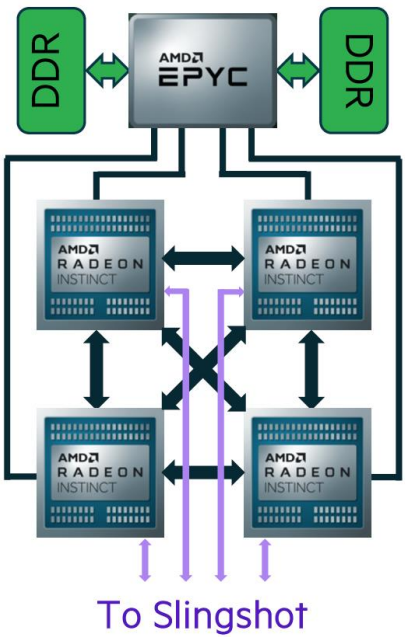
LUMI, the Queen of the North

LUMI is a Tier-0 ***GPU-accelerated supercomputer*** *that enables the convergence of* ***high-performance computing***, ***artificial intelligence***, *and* ***high-performance data analytics.***

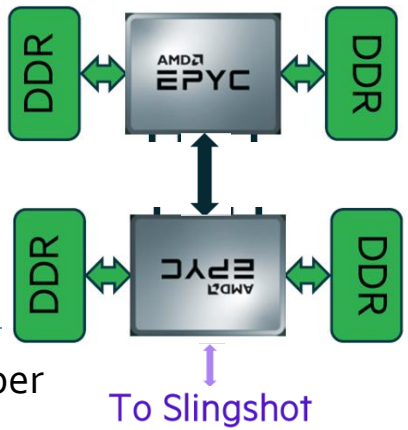


LUMI compute node configurations

LUMI-G



2560 nodes with 4 x MI250X + 1 x AMD Trento processor, 512 GB host memory and 512 GB device memory (HBM2)
4 x 200 Gbit/s NIC
Infinity Fabric



2x 64-core AMD Milan processors per node
1376 nodes with 256 GB, 128 with 512 GB and 32 with 1 TB
1 x 200 Gbit/s NIC

LUMI-C

LUMI timeline

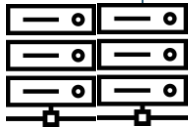
2020

2021

2022

System procurement:
November 2019 – August 2020

Data center preparation ready Q1/2021



1st phase
Q4/2021
- LUMI-C
- storage
- Early Access Platform

Q2-3/2022
- Gradual deployment of LUMI-G
- LUMI-C NIC upgrade

Q4/2022 **Final Configuration**
- LUMI-O General availability

In customer use 01/01/22

LUMI programming environment

- ROCm (Radeon Open Compute)
  - Usual set of accelerated scientific libraries (BLAS, FFT etc)
  - Usual machine learning frameworks and libraries (Tensorflow, PyTorch etc)
  - Compilers for the GPUs (AOCC)
  - Performance analysis tools
- Cray Programming Environment (CPE) stack
  - Cray Compiling Environment
  - GNU compilers
  - LibSci libraries, performance analysis tools, debuggers,…
- LUMI stack
  - Allows software installed in the user's space through EasyBuild in a way that is 100% compatible with the system stack
- More information:
  https://www.lumi-supercomputer.eu/may-we-introduce-lumi/

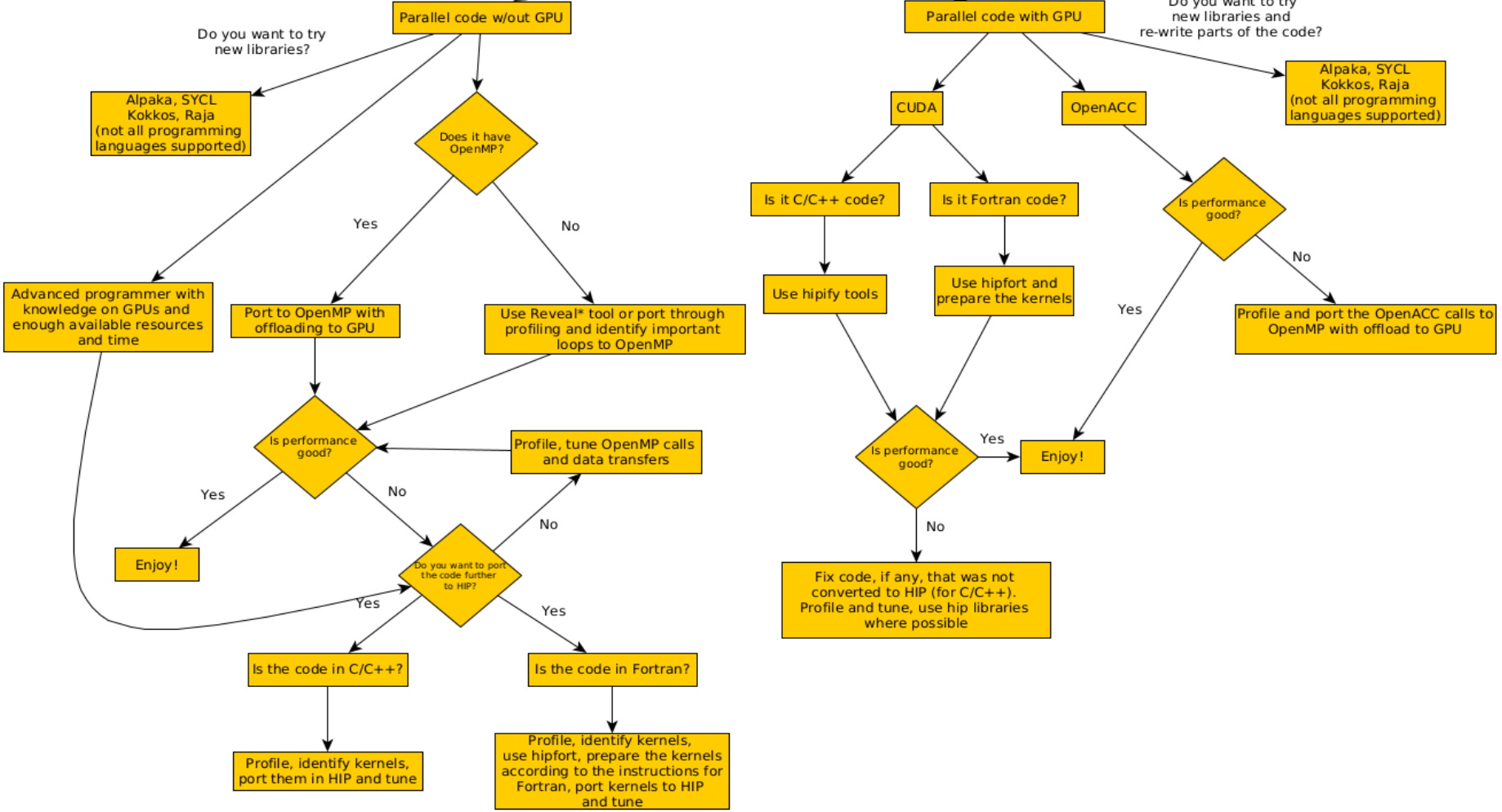
LUMI programming environment

- Traditional HPC programming models & languages supported
  - C, C++, Fortran, Python
- Parallel programming
  - MPI, OpenMP
  - PGAS (Fortran Coarrays, UPC, OpenSHMEM)
- GPU programming
  - OpenMP 5.1 offload
  - OpenACC: 3.0 for Fortran, no proper support in C/C++
  - HIP (Heterogeneous Interface for Portability)
  - hipSYCL
- Performance analysis tools
  - CrayPAT, Reveal
  - Tau
  - ROCprof
  - OmniPerf/OmniTrace
  - SCORE-P/Vampir
- Debuggers
  - ARM Forge
  - CPE debuggers (CCDB, gdb4hpc,…)
- Software installation and management
  - EasyBuild, Spack

Preparing applications and workflows for LUMI

LUMI

- Possibility of combining CPU and GPU nodes within one job – perhaps only part of the application needs to be GPU-enabled
- CUDA codes needs to be converted to HIP
  - HIPify tools can automatize the effort (~25% code needs manual work)
- Recommended to port C/C++ OpenACC codes to OpenMP offload
- In case of major rewrites: Consider writing your application on top of modern frameworks and libraries
  - Kokkos, Alpaka etc, or domain-specific frameworks e.g. GridTools


Parallel code w/out GPU
Do you want to try new libraries?
Alpaka, SYCL Kokkos, Raja (not all programming languages supported)
Does it have OpenMP?
Yes
No
Advanced programmer with knowledge on GPUs and enough available resources and time
Port to OpenMP with offloading to GPU
Use Reveal* tool or port through profiling and identify important loops to OpenMP
Is performance good?
Profile, tune OpenMP calls and data transfers
Yes
No
Enjoy!
Do you want to port the code further to HIP?
No
Yes
Yes
Is the code in C/C++?
Is the code in Fortran?
Profile, identify kernels, port them in HIP and tune
Profile, identify kernels, use hipfort, prepare the kernels according to the instructions for Fortran, port kernels to HIP and tune
Parallel code with GPU
Do you want to try new libraries and re-write parts of the code?
Alpaka, SYCL Kokkos, Raja (not all programming languages supported)
CUDA
OpenACC
Is it C/C++ code?
Is it Fortran code?
Is performance good?
No
Use hipify tools
Use hipfort and prepare the kernels
Yes
Profile and port the OpenACC calls to OpenMP with offload to GPU
Is performance good?
Yes
Enjoy!
No
Fix code, if any, that was not converted to HIP (for C/C++). Profile and tune, use hip libraries where possible