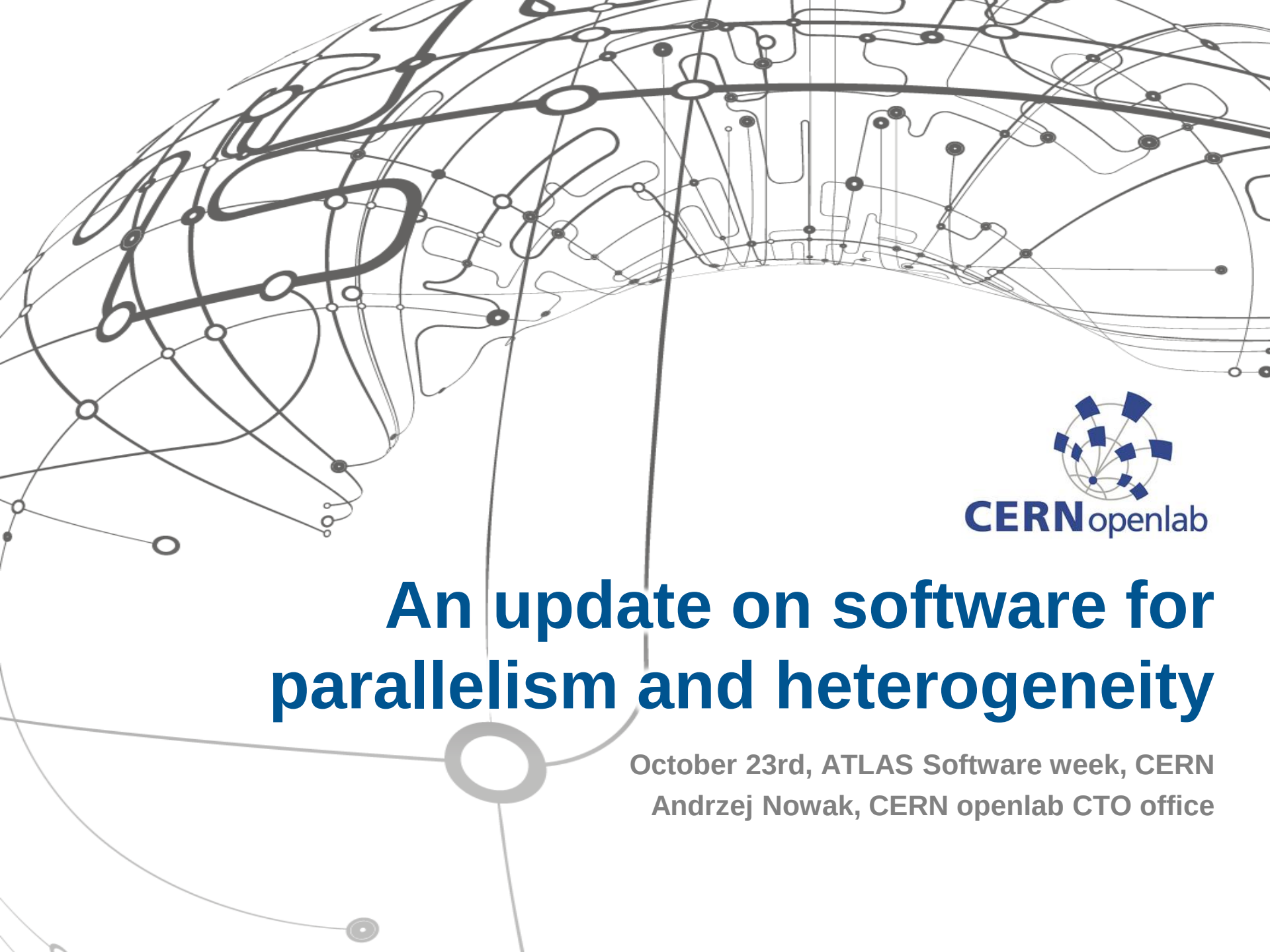


An update on software for parallelism and heterogeneity

October 23rd, ATLAS Software week, CERN

Andrzej Nowak, CERN openlab CTO office

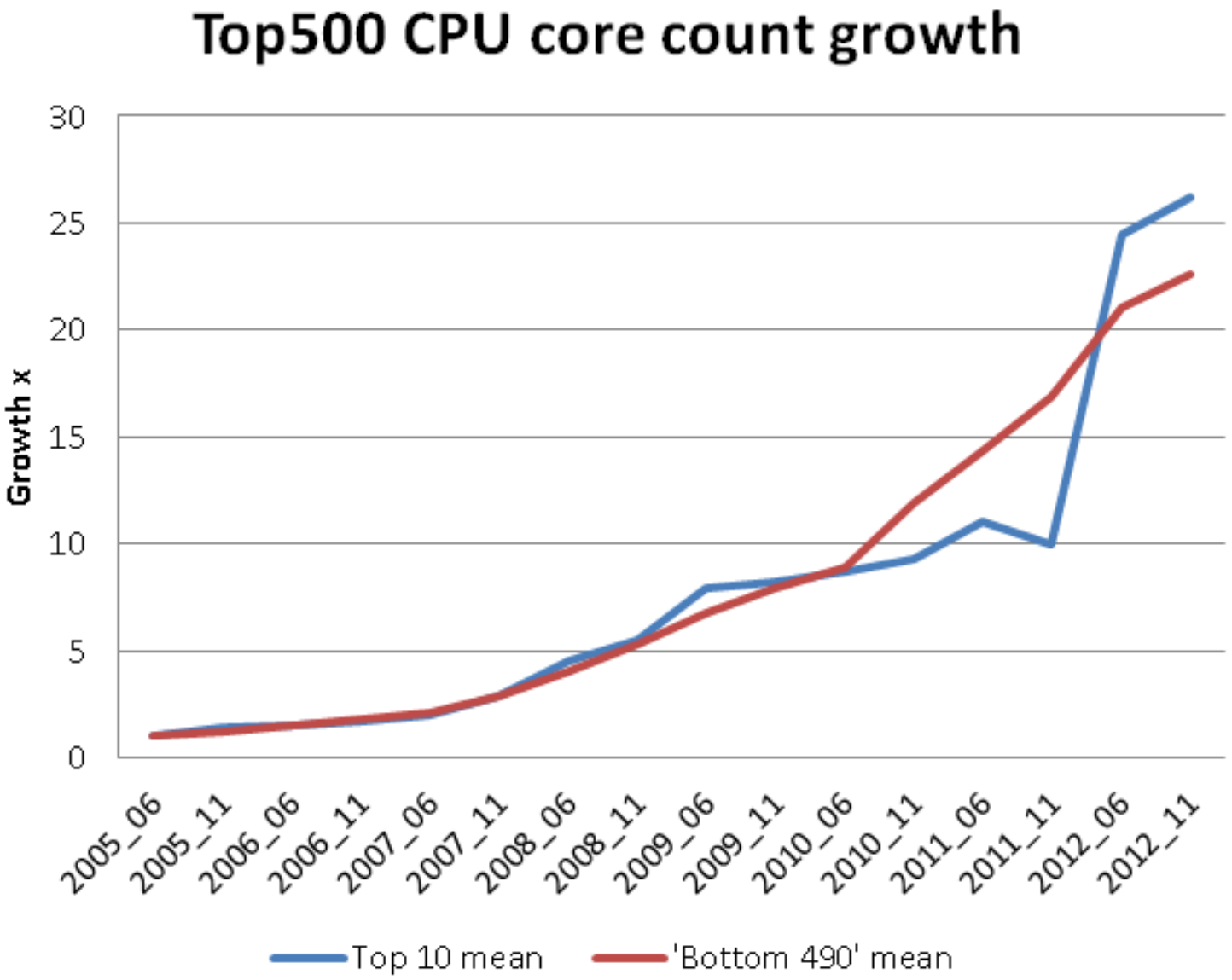
CERN openlab

- CERN openlab is a framework for evaluating and integrating cutting-edge IT technologies or services in partnership with industry
- The Platform Competence Center (PCC) has worked closely with Intel for the past decade and focuses on:
  - many-core scalability
  - performance tuning and optimization
  - benchmarking and thermal optimization
  - teaching

Partners

Contributors

Associates



Modeled after Lehto, Manninen, von Alfthan

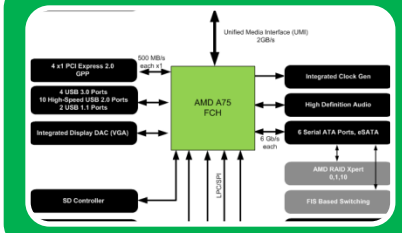
Heterogeneity scale and scenarios

Cluster level

- Non-homogeneous nodes
- Large scale, expensive interconnect

Node level

- Non-homogeneous components of a node
- Standard platform interconnect



Chip level

- Non-homogeneous components in a package/chip
- On-chip interconnect or standard bus

Heterogeneity

- A whole new set of problems
- How is heterogeneity expressed in hardware?
- How far from one node to another?
- Will the floating point results match?
- How to express heterogeneity in code?
- What coding standards to use? Will code compile anywhere? Will it perform well?
- How to split up the workload?

Intel MIC programming models

Native mode

workload runs entirely on a co-processor system (networked via PCIe)

Offload

Co-processor as an accelerator where host gets weak

Balanced

Co-processor and host work together

Cluster

application distributed across multiple cards (possibly including host)

Intel MIC progamming models

- Native (GCC/ICC)
  - Compile on the card or cross-compile on the host
- Offload
  - OpenMP
  - #pragma offload
- Balanced (symmetric) – need a careful balance and locality
  - Topology-aware MPI interfaces (example: TACC)
- Cluster
  - MPI or MPI + ?

New features in OpenMP4

- OpenMP: parallelism standard for SHM
- Compiler and runtime parallelize regions, tasks supported
- OpenMP can play along with autovec
- OpenMP 4.0
  - Vectorization: omp simd, simd functions
  - Offload regions (omp target)
  - thread teams, task groups; much like in CUDA or MPI – can control distribution
  - Data mapping (to/from)
  - OMP_PROC_BIND
  - User reductions

Cilk+

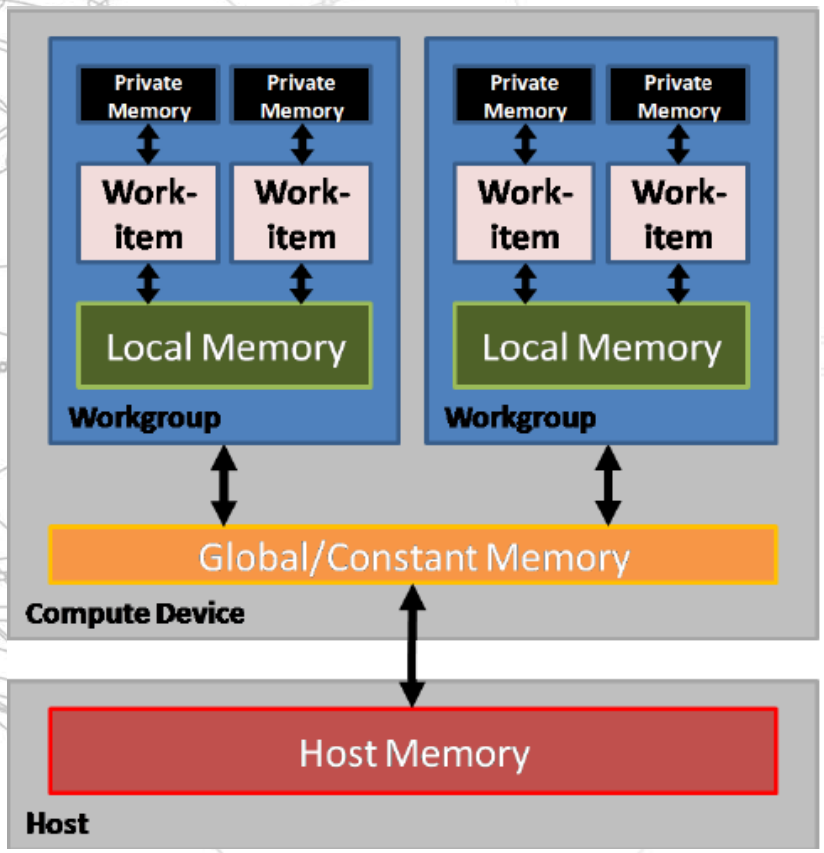
- Supports „for loop” parallelism
  - Functions, spawns, syncs
- Supports explicit array vector syntax
  - Cool syntax that we know from Matlab, Python, but will it perform?
- Compiler support
  - Both ICC 13+ and GCC branch 4.8+
  - ICC support more mature but lacking optimization
  - GCC support still some way from the standard
- Practical experiments show:
  - Alignment is very important
  - Sometimes need attributes
  - Good vectorization obtained with simple syntax

Cilk+ syntax examples

Simple assignments	`A[:] = 5;`
Range assignment	`A[0:7] = 5;`
Assignment w/ stride	`A[0:5:2] = 5;`
Increments	`A[:] = B[:] + 5;`
2D arrays	`C[:][:] = 12;`
	`C[0:5:2][:] = 12;`
Function calls	`func (A[:]);`
	`A[:] = pow(c, B[:])`
	operators
Conditions	`if (5 == a[:])`
	`  results[:] = „y”`
	`else`
	`  results[:] = „n”`
Reductions	`__sec_reduce_mul (A[:])`
Gather	`C[:] = A[B[:]]`
Scatter	`A[B[:]] = C[:]`

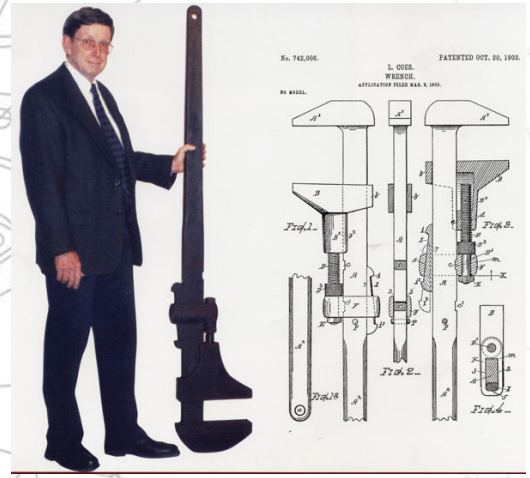
http://cilkplus.org

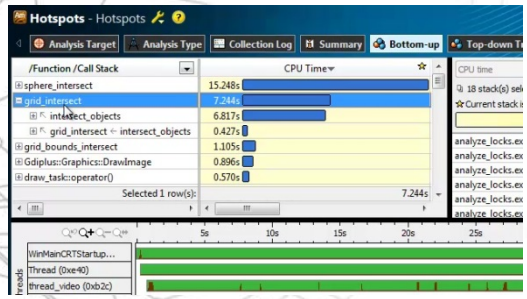
OpenCL status

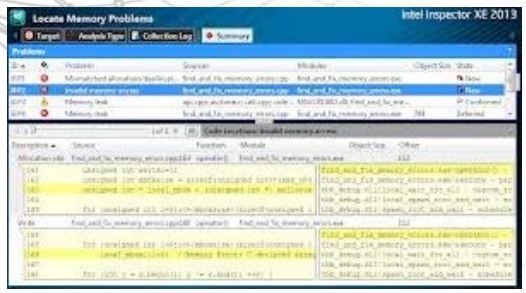


- Compute API for acceleration and co-processing
- Best suited for GPUs, CPU implementations still lacking
- Supported by AMD, Intel, even FPGA vendors
- Need extensions for FP double (1.2)
- Unclear future, challenges of portable performance (as elsewhere)

Working with parallel code







- Use the right tools. Intel examples:
  - Inspector – correctness
  - VTune – performance (OpenMP support)
  - Debugger – particularly interesting on MIC, with vector registers, OpenMP supported
  - If you have a CERN account, you can use Intel tools
- GDB
  - info threads (gdb has its own numbering – you can use **`pthread_setname_np()`**)
  - thread num
  - break line thread num
  - Breakpoints make syscalls return
  - No way to lock-step all threads
  - OpenMP regions as functions
- MPI debugging – an art (better use Intel MPI tools, TotalView or DDT)

Thank you

Andrzej.Nowak@cern.ch